Anaconda Platform Documentation

Release 5.2.0-0

Anaconda Inc.

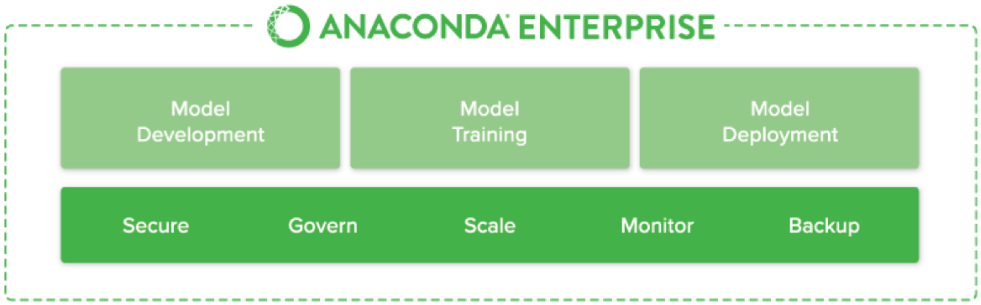
Sep 06, 2018

Contents

Anaconda Enterprise is an enterprise-ready, secure and scalable data science platform that empowers teams to govern data science assets, collaborate and deploy data science projects.

With Anaconda Enterprise, you can:

- **Develop**: ML/AI pipelines in a central development environment that scales from laptops to thousands of nodes
- **Govern**: Complete reproducibility from laptop to cluster with the ability to configure access control
- **Automate**: Model training and deployment on scalable, container-based infrastructure



CHAPTER 1

Installing and configuring AE

1.1 Installation requirements

Anaconda Enterprise can be installed on one to five nodes during the initial installation. After the initial installation, you can add or remove nodes from the Anaconda Enterprise cluster any time. For more information, see *Adding and removing nodes*.

A rule of thumb for each project session or deployment is 1 CPU, 1 GB of RAM and 5 GB of disk space.

For more information about sizing for a particular component, see the following minimum requirements:

- *Hardware requirements*
- *Disk IOPS requirements*
- *Operating system requirements*
- *Browser requirements*
- *Network requirements*
- *DNS requirements*
- *Kernel module requirements*
- *Security requirements*
- *TLS/SSL certificate requirements*
- *Verifying system requirements*

To configure Spark or Hadoop data sources, refer to the *requirements and process here*.

Hardware requirements

Minimum and recommended specs of the master and worker nodes and the total cluster:

Master node	Minimum	Recommended
CPU (cores)	8	16
RAM (GB)	16	32
Disk space in /opt/anaconda (GB)	100	500*
Disk space in /var/lib/gravity (GB)	100	100
Disk space in /tmp or $TMPDIR (GB)	30	30

Worker nodes	Minimum	Recommended
CPU (cores)	8	16
RAM (GB)	16	32
Disk space in /var/lib/gravity (GB)	100	100
Disk space in /tmp or $TMPDIR (GB)	30	30

Cluster totals	Minimum
CPU (cores)	16
RAM (GB)	32

*Note that the recommended disk space in `/opt/anaconda` includes project and package storage (including mirrored packages).

*Note that currently `/opt/anaconda` **must** be a supported filesystem such as `ext4` or `xfs` and **cannot** be an NFS mountpoint. Subdirectories of `/opt/anaconda` may be mounted through NFS.

*If you are installing Anaconda Enterprise on an `xfs` filesystem, then it needs to support `d_type` to work properly. XFS filesystems do not support `d_type` if they have been formatted with the `-n ftype=0` option. Before installing Anaconda Enterprise, recreate the filesystem using the correct parameter for XFS, or use another supported filesystem.

Disk IOPS requirements

Cloud providers report concurrent disk input/output operations per second (IOPS). Master and worker nodes require a minimum of 3000 concurrent IOPS. Fewer concurrent IOPS than 3000 will fail.

Hard disk manufacturers report sequential IOPS, which are different than concurrent IOPS. On-premises installations require servers with disks that support a minimum of 50 sequential IOPS, typically 7200 revolution per minute (RPM) disks or faster.

Operating system requirements

NOTE: Anaconda Enterprise cannot be installed with heterogeneous versions in the same cluster. Before installing, verify that all cluster nodes are operating the same version of the OS.

Linux versions:

- RHEL/CentOS 7.2, 7.3, 7.4, 7.5
- Ubuntu 16.04
- SUSE 12 SP2, 12 SP3
- Hosted vSphere such as Rackspace or OVH

NOTE: On SUSE, set `DefaultTasksMax=infinity` in `/etc/systemd/system.conf`.

GPU requirements

To use GPUs with Anaconda Enterprise, you'll need to install the NVIDIA CUDA 9.0 or higher drivers from the Nvidia runfile on the host operating system *of any GPU worker nodes*.

Software requirements

The programs Docker (and `dockerd`), `dnsmasq`, and `lxd` will conflict with Anaconda Enterprise. If any of these have been installed on the master node or any worker nodes, remove them from all nodes before installing Anaconda Enterprise.

Browser requirements

- Edge 14+
- Chrome 39+
- Firefox 49+
- Safari 10+
- Internet Explorer 11+ (Windows 10)

The minimum browser screen size for using the platform is 800 pixels wide and 600 pixels high.

Network requirements

The following network ports for Anaconda Enterprise are externally accessible:

- 80 - HTTP
- 443 - HTTPS
- 32009 - Operations center
- 61009 - Install wizard (only used during cluster installation)

The following network ports for Anaconda Enterprise must be open *internally*, between cluster nodes:

- 53 - Internal cluster DNS
- 2379 - Etcd server communication
- 2380 - Etcd server communication
- 3008 - Internal Anaconda Enterprise service
- 3009 - Internal Anaconda Enterprise service
- 3010 - Internal Anaconda Enterprise service
- 3011 - Internal Anaconda Enterprise service
- 3012 - Internal RPC agent
- 3022 - Teleport internal SSH control panel
- 3023 - Teleport internal SSH control panel
- 3024 - Teleport internal SSH control panel
- 3025 - Teleport internal SSH control panel
- 3080 - Teleport web UI
- 4001 - Etcd server communication
- 5000 - Docker registry
- 6443 - Kubernetes API server
- 7001 - Etcd server communication
- 7373 - Peer-to-peer health check
- 7496 - Peer-to-peer health check
- 8472 - VXLAN (Overlay network)

- 10248 - Kubernetes components
- 10249 - Kubernetes components
- 10250 - Kubernetes components
- 10255 - Kubernetes components

If you plan to use online package mirroring, you'll need to whitelist the following domains:

- repo.continuum.io
- anaconda.org
- conda.anaconda.org
- binstar-cio-packages-prod.s3.amazonaws.com

IPv4 forwarding on servers is required for internal load balancing and must be turned on. Anaconda Enterprise performs pre-flight checks and only allows installation on nodes that have the required kernel modules and other correct configuration.

To enable IPv4 forwarding, run:

```
sudo sysctl -w net.ipv4.ip_forward=1
```

To persist this setting on boot, run:

```
sudo echo -e "# Enable IPv4 forwarding\nnet.ipv4.ip_forward=1" >> /etc/sysctl.d/99-
↪ipv4_forward.conf
```

If any Anaconda Enterprise users will use the local graphical program *Anaconda Navigator* in online mode, they will need access to these sites, which may need to be whitelisted in your network's firewall settings.

- https://repo.anaconda.com (or for older versions of Navigator and Conda, https://repo.continuum.io)
- https://conda.anaconda.org if any users will use conda-forge and other channels on Anaconda Cloud (anaconda.org)
- https://vscode-update.azurewebsites.net/ if any users will install Visual Studio Code
- google-public-dns-a.google.com (8.8.8.8:53) to check internet connectivity with Google Public DNS

DNS requirements

Web browsers use domain names and web origins to separate sites so they cannot tamper with each other, and a system such as Anaconda includes many deployments from different users.

If these deployments had addresses such as `https://anaconda.yourdomain.com/apps/001` and `https://anaconda.yourdomain.com/apps/002`, then one app could access the cookies of the other, and JavaScript in one app could access the other app.

To prevent this security risk, Anaconda gives deployments addresses such as `https://uuid001.anaconda.yourdomain.com` and `https://uuid002.anaconda.yourdomain.com`.

"yourdomain.com" would be replaced with your domain name, and "uuid001" and "uuid002" would be replaced by dynamically generated universally unique identifiers (UUIDs). Other details of the address might be different than this example as well.

This system requires the use of wildcard DNS entries that apply to a set of domain names such as `*.anaconda.yourdomain.com`.

For example, if you are using the fully qualified domain name (FQDN) anaconda.yourdomain.com with a master node IP address of 12.34.56.78, the DNS entries would be as follows:

```
  anaconda.yourdomain.com IN A 12.34.56.78
*.anaconda.yourdomain.com IN A 12.34.56.78
```

The wildcard subdomain's DNS entry points to the Anaconda Enterprise master node. **Kernel module requirements**

The following kernel modules are required for Kubernetes to function properly.

br_netfilter module

The bridge netfilter kernel module is required for Kubernetes iptables-based proxy to work correctly.

The bridge kernel module commands are different for different versions of CentOS.

To find your operating system version run `cat /etc/*release*` or `lsb-release -a`.

On RHEL/CentOS 7.2 the bridge netfilter module name is `bridge` and on other operating systems and other versions of CentOS the module name is `br_netfilter`.

To check if the module is loaded run:

```
# For RHEL/CentOS 7.2
lsmod | grep bridge

# For all other supported platforms
lsmod | grep br_netfilter
```

If the above commands did not produce any result, then the module is not loaded. Run the following command to load the module:

```
# For RHEL/CentOS 7.2
sudo modprobe bridge

# For all other supported platforms
sudo modprobe br_netfilter
```

Now run:

```
sudo sysctl -w net.bridge.bridge-nf-call-iptables=1
sudo sysctl -w net.bridge.bridge-nf-call-ip6tables=1
```

To persist this setting on boot, run:

```
sudo echo "# Enable bridge module" >> /etc/sysctl.d/99-bridge.conf
sudo echo "net.bridge.bridge-nf-call-iptables=1" >> /etc/sysctl.d/99-bridge.conf
sudo echo "net.bridge.bridge-nf-call-ip6tables=1" >> /etc/sysctl.d/99-bridge.conf
```

overlay module

The overlay kernel module is required to use overlay or overlay2 Docker storage driver.

To check that overlay module is loaded:

```
lsmod | grep overlay
```

If the above command did not produce any result, then the module is not loaded. Use the following command to load the module:

```
sudo modprobe overlay
```

ebtables module

The ebtables kernel module is required to allow a service to communicate back to itself via internal load balancing when necessary.

To check that ebtables module is loaded:

```
lsmod | grep ebtables
```

If the above command did not produce any result, then the module is not loaded. Use the following command to load the module:

```
sudo modprobe ebtables
```

iptable_filter

The iptable_filter kernel module is required to make sure firewall rules that Kubernetes sets up function properly.

To check that iptable_filter module is loaded:

```
lsmod | grep iptable_filter
```

If the above command did not produce any result, then the module is not loaded. Use the following command to load the module:

```
sudo modprobe iptable_filter
```

iptable_nat

The iptable_nat kernel module is required to make sure firewall rules that Kubernetes sets up function properly.

To check that iptable_nat module is loaded:

```
lsmod | grep iptable_nat
```

If the above command did not produce any result, then the module is not loaded. Use the following command to load the module:

```
sudo modprobe iptable_nat
```

NOTE: During installation, the Anaconda Enterprise installer alerts you if any of these modules are not loaded.

NOTE: If your system does not load modules at boot, add the following to ensure they are loaded in case the machine gets rebooted:

```
sudo bash -c "echo 'overlay' > /etc/modules-load.d/overlay.conf"
sudo bash -c "echo 'br_netfilter' > /etc/modules-load.d/netfilter.conf"
sudo bash -c "echo 'ebtables' > /etc/modules-load.d/ebtables.conf"
sudo bash -c "echo 'iptable_filter' > /etc/modules-load.d/iptable_filter.conf"
sudo bash -c "echo 'iptable_nat' > /etc/modules-load.d/iptable_nat.conf"
```

Mount settings

Many linux distributions include the kernel setting *fs.may_detach_mounts = 0*. This can cause conflicts with the docker daemon and Kubernetes will show pods as stuck in the terminating state if docker is unable to clean up one of the underlying containers.

If the installed kernel exposes the option *fs.may_detach_mounts*, we recommend always setting this value to 1:

For CentOS and RHEL:

```
sudo sysctl -w fs.may_detach_mounts=1
sudo bash -c "echo 'fs.may_detach_mounts = 1' >> /usr/lib/sysctl.d/99-containers.conf"
```

For Ubuntu:

```
sudo sysctl -w fs.may_detach_mounts=1
sudo bash -c "echo 'fs.may_detach_mounts = 1' >> /etc/sysctl.d/10-may_detach_mounts.
↪conf"
```

Security requirements

For CentOS and RHEL:

- Disable SELinux on all of the cluster nodes.
- Various tools may be used to configure firewalls and open required ports, including `iptables`, `firewall-cmd`, `susefirewall2`, and others.

  Make sure that the firewall is permanently set to keep the required ports open, and will save these settings across reboots. Then restart the firewall to load these settings immediately.
- Sudo access.

TLS/SSL certificate requirements

Anaconda Enterprise must use TLS/SSL to operate. Self-signed certificates are used during the initial installation, and organizational TLS/SSL certificates can be configured after installation.

Certificates may either be purchased commercially, or generated from an internal organization public key infrastructure (PKI) system. Either configuration will include:

- a certificate for the root certificate authority (root CA),
- an intermediate certificate chain,
- a server certificate, and
- a private server key (called "server key" in our configuration file).

When using an internal PKI signed setup, the CA certificate is inserted into the Kubernetes secret.

Web browsers use domain names and web origins to separate sites so they cannot tamper with each other, and a system such as Anaconda includes many deployments from different users.

If these deployments had addresses such as `https://anaconda.yourdomain.com/apps/001` and `https://anaconda.yourdomain.com/apps/002`, then one app could access the cookies of the other, and JavaScript in one app could access the other app.

To prevent this security risk, Anaconda gives deployments addresses such as `https://uuid001.anaconda.yourdomain.com` and `https://uuid002.anaconda.yourdomain.com`.

"yourdomain.com" would be replaced with your domain name, and "uuid001" and "uuid002" would be replaced by dynamically generated universally unique identifiers (UUIDs). Other details of the address might be different than this example as well.

This system requires the use of wildcard TLS/SSL certificates that apply to a set of domain names such as `*.anaconda.yourdomain.com`.

In a typical configuration one main (primary) certificate will have the hostname, and include the wildcard name as a SAN entry, with no separate wildcard certificate. It is also possible to have one main (primary) certificate with the hostname only, and a separate wildcard certificate with the wildcard entry only.

Verifying system requirements

Anaconda Enterprise performs system checks during the install to verify CPU, RAM and other system requirements. The system checks can also be performed manually before the installation using the following commands from the installer directory, `~/anaconda-enterprise-<installer-version>`.

NOTE: You can perform this check after downloading and extracting the installer.

To perform system checks on a master node, run the following command as sudo or root user:

```
sudo ./gravity check --profile ae-master
```

To perform system checks on a worker node, run the following command as sudo or root user:

```
sudo ./gravity check --profile ae-worker
```

If all of the system checks pass and all requirements are met, the output from the above commands will be empty. If the system checks fail and some requirements are not met, the output will indicate which system checks failed.

Storage requirements

To check your available disk space, use the built-in Linux `df` utility with the -h parameter for human readable format:

```
df -h /var/lib/gravity

df -h /opt/anaconda

df -h /tmp
# or
df -h $TMPDIR
```

Memory requirements

To show the free memory size in GB, run:

```
free -g
```

CPU requirements

To check the number of cores, run:

```
nproc
```

1.2 Installing on-premises

Before you begin:

- Verify that your system meets all *Anaconda Enterprise installation requirements*.
- Configure your A record in DNS for the master node with the actual domain name you will use for your Anaconda Enterprise installation.
- Confirm that there is sufficent free space on each node. Approximately 30GB of available free space is required for the Anaconda Enterprise installer to temporarily decompress files to the `/tmp` directory during the installation process.
- Verify that the clocks of each of the nodes is in sync prior to starting the installation process, to avoid potential issues when installing Anaconda Enterprise on a system with multiple nodes

If adequate free space is not available in the `/tmp` directory, you can specify the location of the temporary directory to be used during installation by setting the `TMPDIR` environment variable to a different location.

Example:

```
sudo TMPDIR=/tmp2 ./gravity install
```

NOTE: When using sudo to install, the temporary directory must be set explicitly in the command line to preserve TMPDIR.

The master node and each worker node all require a temporary directory of the same size and should each use the `TMPDIR` variable as needed.

Or, you can install as root.

- By default, Anaconda Enterprise installs using a service account with the user ID (UID) 1000. You can change the UID of the service account by using the `--service-uid` option or the `GRAVITY_SERVICE_USER` environment variable at installation time.

For example, to use UID 1001, use `sudo ./gravity install --service-uid=1001` or `sudo GRAVITY_SERVICE_USER=1001 ./gravity install`.

- Optionally create a new directory and set `TMPDIR`. User 1000 (or the UID for the service account) needs to be able to write to this directory. This means they can read, write and execute on the $TMPDIR. For example, to give write access to UID 1000, run `sudo chown 1000 -R $TMPDIR`.

Now you can install Anaconda Enterprise using one of the following methods:

- *Using a web browser (recommended)*
- *Using a command line*
- *Installing in air gapped environments*

Using a web browser

On the master node, download and decompress the installer, replacing `<location_of_installer>` with the location of the installer, and `<version>` with your installer version.:

```
curl -O <location_of_installer>.tar.gz
tar xvzf anaconda-enterprise-<version>.tar.gz
cd anaconda-enterprise-<version>
```

NOTE: The tarball file is 7GB, so it may take some time to download.

On the master node, run the pre-installation system checks as sudo or root user before proceeding with the installation:

```
sudo ./gravity check --profile ae-master
```

After doing the pre-installation system checks, run the installer on the master node as sudo or root user:

```
sudo ./gravity wizard
```

```
* [0/100] starting installer
confirm the config:

* IP address: 1.1.1.1

confirm (yes/no):
yes
* [0/100] starting installer
* [0/100] preparing for installation... please wait
* [0/100] application: AnacondaEnterprise:5.1.1
* [0/100] starting web UI install wizard
--------------------------------------------------------------------------------
↪-------
```

(continues on next page)

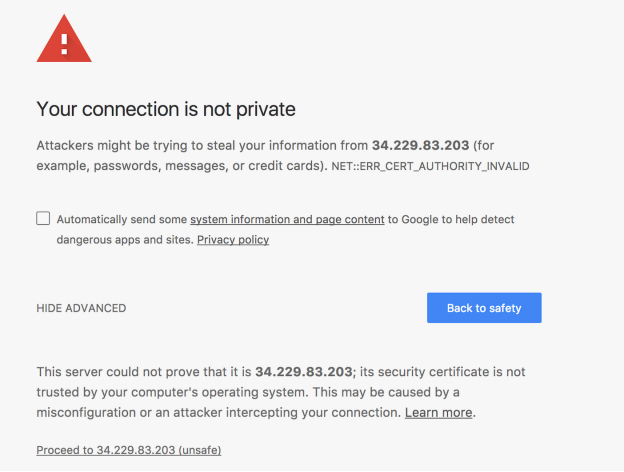
(continued from previous page)

```
OPEN THIS IN BROWSER: https://1.1.1.1:61009/web/installer/new/gravitational.io/
↪AnacondaEnterprise/5.1.0?install_token=0a39d9a2f16036fc6583e78b502e7cae
------------------------------------------------------------------------------------
↪-------
```

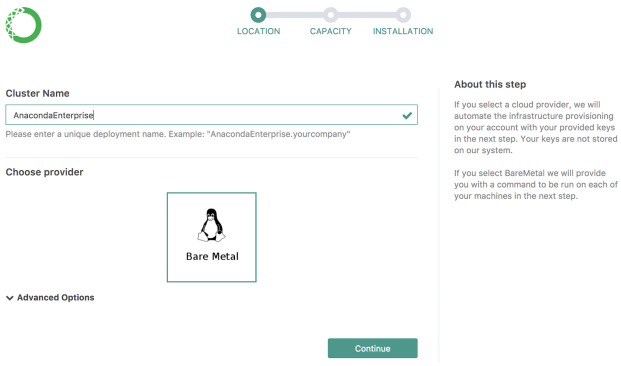
To start the browser-based install, copy the example URL into your browser, replacing the IP `1.1.1.1` and URL above with your actual URL. Ensure that you are connecting to the public network interface.

NOTE: If you are unable to connect to the URL due to security measures in place at your organization, select **File > New Incognito Window** to launch the installer.

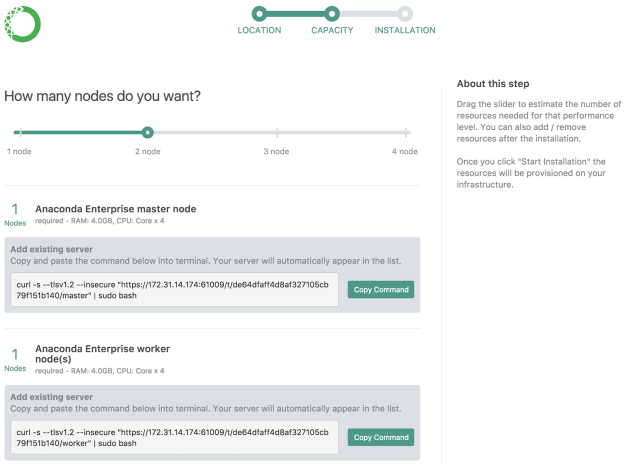
The installer will install a self-signed TLS/SSL certificate, so you can click the link at the bottom of this warning message to proceed:



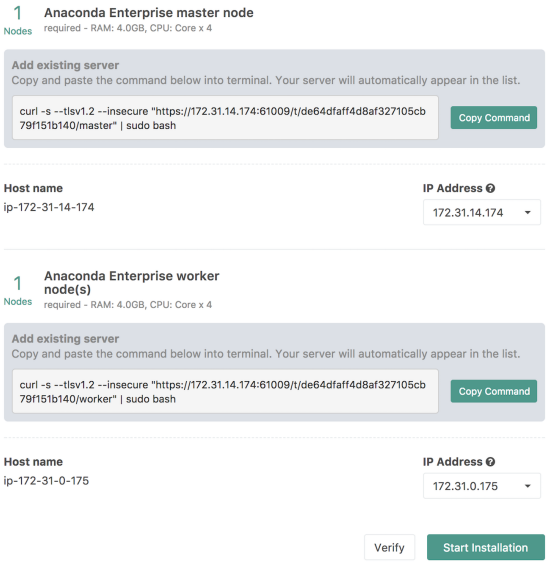
Enter the name to use for your deployment in the **Cluster Name** field. For on-premised installations, the **Bare Metal** option is already selected, so you can click **Continue**.



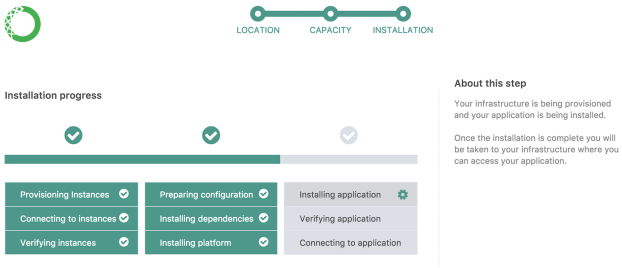
Select the number of nodes that you want to install in the cluster. One node will act as the master node, and any remaining nodes will be worker nodes. See *Fault tolerance* for more information on how to size your cluster.



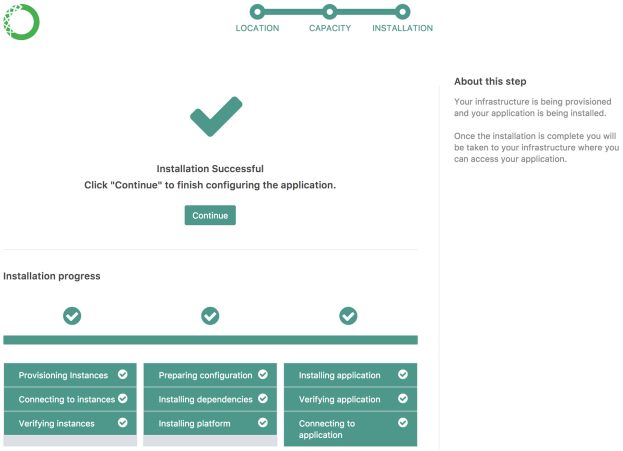
On each node you plan to install Anaconda Enterprise, copy and run the command that's provided as it applies to the master node and any worker nodes. As you run the command on each node, the host name of the node is listed in the installer.



After all nodes are list, click **Start Installation**. This process takes approximately 20 minutes to complete.



When the installation is complete, the following screen is displayed.



Click **Continue** to proceed to *Post-Install configuration*.

NOTE: The installer running in the terminal will note that installation is complete and that you can stop the installer process. **Do not do so** until you have completed the post-install configuration.

Using a command line

If you cannot connect to the server from a browser–because you're installing from a different network, for example–you can install Anaconda Enterprise using a command line.

On each node in the cluster, download and decompress the installer, replacing `<location_of_installer>` with the location of the installer, and `<version>` with your installer version:

```
curl -O <location_of_installer>.tar.gz
tar xvzf anaconda-enterprise-<version>.tar.gz
cd anaconda-enterprise-<version>
```

On the **master node**, run the pre-installation system checks—as sudo or root user—before proceeding with the installation:

```
sudo ./gravity check --profile ae-master
```

Create a file named `values.yaml` with the following values, replacing `HOSTNAME` with the fully-qualified domain name (FQDN) of the host server:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: anaconda-enterprise-install
data:
  values: |
    hostname: HOSTNAME
    generateCerts: true
    keycloak:
      includeMasterRealm: true
```

After running the pre-installation system checks and creating the YAML file, run the following command *on the master node* as sudo or root user, where you replace:

- the `advertise-addr` IP address with the address you want to be visible to the other nodes
- `CLUSTERNAME` with a name, otherwise a random cluster name will be assigned
- `/path/to/values.yaml` with the path to the `values.yaml` file you created

For `flavor`, choose from the following options the one that represents the number and type of nodes you want to install in the cluster:

- `small`: installs a single-node cluster (one ae-master node). This is the default flavor.
- `medium`: installs three nodes (one ae-master node and two ae-worker nodes)
- `large`: installs five nodes (one ae-master node, two k8s-master nodes and two ae-worker nodes):

```
sudo ./gravity install --advertise-addr=192.168.1.1 --token=anaconda-enterprise --
↪cluster=CLUSTERNAME --flavor=small --config /path/to/values.yaml
```

The command line displays the installer's progress:

```
* [0/100] starting installer
* [0/100] preparing for installation... please wait
* [0/100] application: AnacondaEnterprise:5.1.1
* [0/100] starting non-interactive install
* [0/100] still waiting for 1 nodes of role "worker" to join
* [0/100] still waiting for 1 nodes of role "worker" to join
* [0/100] still waiting for 1 nodes of role "worker" to join
* [0/100] initializing the operation
* [20/100] configuring packages
* [50/100] installing software
```

On each **worker node**, run the following command, replacing the `advertise-addr` IP address with the address you want to be visible to the other nodes:

```
sudo ./gravity join 192.168.1.1 --advertise-addr=192.168.1.2 --token=anaconda-
↪enterprise --role=worker
```

The command line displays the installer's progress:

```
* [0/100] joining cluster
* [0/100] connecting to cluster
* [0/100] connected to installer at 192.168.1.1
* [0/100] initializing the operation
* [20/100] configuring packages
* [50/100] installing software
```

This process takes approximately 20 minutes.

After you've finished installing Anaconda Enterprise, you'll need to create a local user account and password to log into the Anaconda Enterprise Operations Center.

First, enter the Anaconda Enterprise environment *on any of the master or worker nodes*:

```
sudo gravity enter
```

Then, run the following command to create a local user account and password for the Anaconda Enterprise Operations Center, replacing <your-email> and <your-password> with the email address and password you want to use.

NOTE: Passwords must be at least six characters long.:

```
gravity --insecure user create --type=admin --email=<your-email> --password=<your-
↪password> --ops-url=https://gravity-site.kube-system.svc.cluster.local:3009
```

Installing in air gapped environments

If the cluster where you will install AE cannot connect to the internet, follow these instructions:

1. Download the installer tarball file to a jumpbox or USB key.
2. Move the installer tarball file to a designated head node in the cluster.
3. Untar the installer file and run `sudo ./gravity wizard` for browser-based installation or `sudo ./gravity install` for CLI-based installation.

Installation and post-install configuration steps are the same on air-gapped and internet-connected installations, so you can continue the installation process from this point.

There are two methods to install Anaconda Enterprise:

- Browser installation (must be on the same network as the target machines)
- Unattended command-line interface (CLI) installation

With both methods, you can create any number of nodes from one to four nodes. You can also add or remove nodes at any time after installation. For more information, see *Adding and removing nodes*.

Post-install configuration

After completing either installation path, complete the *post-install configuration steps*.

1.3 Installing on Amazon Web Services (AWS)

The process for installing AE on AWS is the same as *Installing on-premises* with one key difference. Due to etcd's sensitivity to disk latency, only use EC2 instances with a minimum of 3000 IOPS. We recommend an instance type no smaller than the `m4.4xlarge` for both master and worker nodes.

When running the `sudo ./gravity install` command, the installer autodetects EC2 and uses the VPC-based flannel backend instead of VXLAN. To force the use of VXLAN, use the `--cloud-provider generic` option in the installation command.

1.4 Installing on Microsoft Azure

The process for installing AE on Microsoft Azure is mostly the same as *Installing on-premises*, with the following considerations specific to Azure:

VM requirements:

To achieve CPU and Disk I/O requirements, the minimum VM size selected should be Standard D16s v3 (16 VCPUs, 64 GB memory).

Other Azure caveats:

For RHEL/CentOS, ensure `dnsmasq` is stopped and disabled:

- Stop dnsmasq: `sudo systemctl stop dnsmasq`
- Disable dnsmasq: `sudo systemctl disable dnsmasq`
- Verify dnsmasq is disabled: `sudo systemctl status dnsmasq`

NOTE: See known issues on Azure, including clock drift.

1.5 Installing on Google Cloud Platform (GCP)

The process for installing AE on Google Cloud Platform follows the standard instructions for *Installing on-premises*.

1.6 Post-install configuration

There are a few platform settings that need to be updated *after installing* Anaconda Enterprise, before you can begin using it. Follow the instructions below, based on whether you used *a web browser* or *a command-line* to install the platform. Then you'll be ready to test your installation and perform additional configuration, specific to your organization.

1.6.1 Browser-based instructions

If you installed Anaconda Enterprise using a web browser, a UI will guide you through some post-install configuration steps.

NOTE: It may take a moment for the **Post-Install Setup** screen to appear. If you see an error immediately after clicking **Continue** at the end of the installation process, please refresh your browser after a few seconds to display the UI.

1. Enter the cluster Admin account credentials that you will use to log in to the Anaconda Enterprise Operations Center initially and click **Next**.

NOTE: The installer will generate self-signed SSL certificates that you can use temporarily to get started. See *Managing TLS/SSL certificates* for information on how to change them later, if desired.

2. Enter the fully-qualified domain name (FQDN) where the cluster will be accessed and click **Finish Setup**.
3. Log in to the Anaconda Enterprise Operations Center using the cluster Admin credentials you provided in Step 1, and follow the instructions below to *update additional default settings* within the Operations Center and the Authorization Center.

1.6.2 Command-line instructions

If you performed an unattended installation using *the command-line instructions*, follow the instructions below to generate self-signed SSL certificates that you can use temporarily to get started. See *Managing TLS/SSL certificates* for information on how to change them later, if desired.

1. *On the master node* for your Anaconda Enterprise installation, run the following commands to save your secrets file to a location where Anaconda Enterprise can access it, replacing `YOUR_FQDN` with the fully-qualified domain name of the cluster on which you installed Anaconda Enterprise.:

```
cd path/to/Anaconda/Enterprise/unpacked/installer
cd DIY-SSL-CA
bash create_noprompt.sh YOUR_FQDN
cp out/DESIRED_FQDN/secret.yaml /var/lib/gravity/planet/share/secrets.yaml
```

Now `/var/lib/gravity/planet/share/secrets.yaml` is accessible as `/ext/share/secrets.yaml` within the Anaconda Enterprise environment.

2. Run the following command to enter the Anaconda Enterprise environment:

```
sudo gravity enter
```

3. Replace the default secrets cert with the contents of your `secrets.yaml` file by running the following commands from within the Anaconda Enterprise environment:

```
$ kubectl delete secrets certs
secret "certs" deleted
$ kubectl create -f /ext/share/secrets.yaml
secret "certs" created
```

4. Run the following command to determine the site name:

```
SITE_NAME=$(gravity status --output=json | jq '.cluster.token.site_domain' -r)
```

5. Run the following command to complete the installation:

```
gravity --insecure site complete $SITE_NAME --ops-url=https://gravity-site.kube-
↪system.svc.cluster.local:3009
```

Now you are ready to follow the instructions below to *test your installation*.

1.6.3 Test your installation

1. Access the Anaconda Enterprise console by entering the URL of your AE server in a web browser: `https://anaconda.example.com`, replacing `anaconda.example.com` with the fully-qualified domain name of the host server.
2. Login with the default username and password `anaconda-enterprise` / `anaconda-enterprise`. After testing your installation, update the credentials for this default login. See *Configuring authentication* for more information.

You can test your install by doing any or all of the following:

- *Creating* a new project and starting an editing session
- *Deploying* a project
- *Generating a token* from a deployment

NOTE: Some of the sample projects can only be deployed after *mirroring the package repository*. To test your installation without doing this first, you can deploy the "Hello Anaconda Enterprise" sample project.

1.6.4 Next steps

Now that you've completed these essential steps, you can do the following:

- *Set TLS/SSL certificates* for the UI and for the Operations Center.
- *Authorize platform users*, including *System Admins*.
- *Configure channels and packages*, including *mirroring package repositories*
- *Configure Hadoop or Spark clusters*
- *Generate custom installers*

1.7 Advanced configuration

After installing Anaconda Enterprise, there are a couple of default settings that you may want to update with information specific to your installation: the password for the database and the redirect URLs for the AE platform.

To update the database password:

1. Run the following command to determine the id of the postgres pod:

```
kubectl get pod | grep postgres
```

2. Run the following command to connect to the postgres pod, where `<id>` represents the id of the pod:

```
kubectl exec -it anaconda-enterprise-postgres-<id> /bin/sh
```

3. Run this `psql` command to connect to the database:

```
psql -h localhost -U postgres
```

4. Set the password by running the following command:

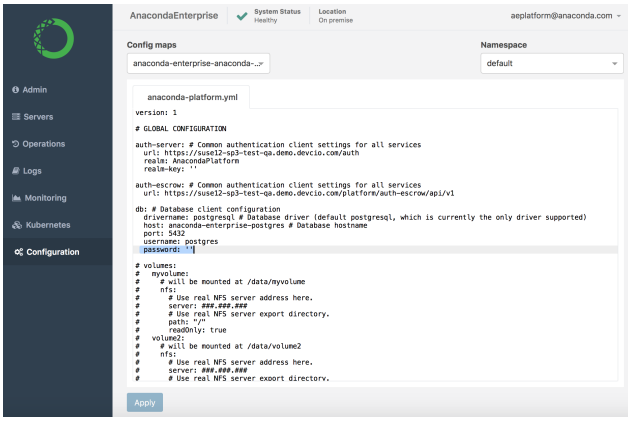
```
ALTER USER user_name WITH PASSWORD 'new_password';
```

5. Restart all the service pods using the following command:

```
kubectl get pods | grep ap- | cut -d' ' -f1 | xargs kubectl delete pods
```

To update the platform settings with the database password of the host server:

1. Access the Anaconda Enterprise Operations Center by entering this URL in your browser: `https://anaconda.example.com:32009`, replacing `anaconda.example.com` with the FQDN of the host server.
2. Login with the default username and password: `aeplatform@yourcompany.com/aeplatform`. You'll be asked to change the default password when you log in.
3. Click **Configuration** in the left menu to display the Anaconda Enterprise Config map.
4. In the GLOBAL CONFIGURATION section of the configuration file, locate the `db` section and enter the password you just set:



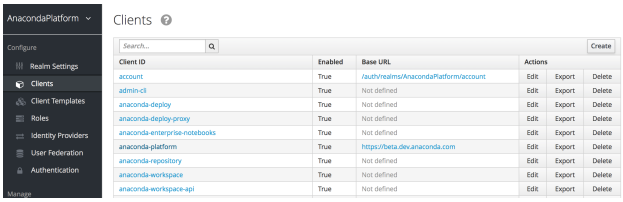
5. Click **Apply** to update the platform with your changes.

To edit the redirect URLs for Anaconda Enterprise:

1. Access the Anaconda Enterprise Authentication Center by entering this URL in your browser: `https://anaconda.example.com/auth/`, replacing `anaconda.example.com` with the fully-qualified domain name of the host server.

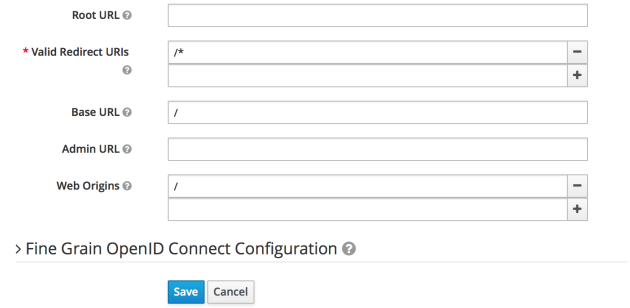
2. Login with the default username and password `admin` / `admin`. It's important that you change this password after completing this initial configuratiopn, as this account will be used to authorize access to the platform. See *Managing System Administrators* for instructions.

3. Verify that `AnacondaPlatform` is displayed as the current realm, then select **Clients** from the **Configure** menu on the left.



4. In the **Clients** list, click `anaconda-platform` to display the platform settings.

5. On the **Settings** tab, update all URLS in the following fields with the FQDN of the Anaconda Enterprise server, or the following symbols:



NOTE: If you choose to provide the FQDN of your AE server, be sure each field also ends with the symbols shown. For example, the **Valid Redirect URIs** would look something like this: `https://server-name.domain.com/*`.

6. Click **Save** to update the server with your changes.

1.8 Migrating from AE 4

1.8.1 Migrating Anaconda Repository 4.x packages

To migrate from Anaconda Repository 4.x to Anaconda Enterprise involves exporting a site-dump of all packages, then importing into Anaconda Enterprise.

Export all packages

You can use the command `anaconda-server-admin export-site` to create a dump of all the packages and information regarding owners and permissions of those packages.

This command creates a directory structure containing all files and user information from Repository.

Example:

```
site-dump/
├── anaconda-user-1
│   ├── 59921152446b5703f430383f--moto
│   ├── 5992115f446b5703fa30383e--pysocks
│   └── meta.json
├── anaconda-organization
│   ├── 5989fbd1446b575b99032652--future
│   ├── 5989fc1d446b575b99032786--iso8601
│   ├── 5989fc1f446b575b990327a8--simplejson
│   ├── 5989fc26446b575b99032802--six
│   ├── 5989fc31446b575b990328b0--xz
│   ├── 5989fc35446b575b990328c6--zlib
│   └── meta.json
└── anaconda-user-2
    └── meta.json
```

Each subdirectory of `site-dump` contains the contents of a user. For example `anaconda-user-1` has two packages, *moto* and `pysocks`. Inside the package directories, which are prefixed with the id of the database, are the package files. There is a `meta.json` in the user directories and the package directories with different metadata. The user's `meta.json` contains information regarding which groups the user belongs to (end users) or what groups the user has (organizations).

NOTE: Other files included in the site-dump such as projects and envs are NOT imported by the package import tool.

Importing packages

You can choose to import packages by username or directory, by all packages or by organization.

For all methods, log into the command line interface:

```
anaconda-enterprise-cli login
```

Then follow the instructions for the method you want to use.

Import by username or directory

The packages for each user are in a separate directory in the site-dump file, so the import process is the same for each username or directory.

Import a single directory from the `site-dump` with the command:

```
anaconda-enterprise-cli admin import site-dump/NAME
```

Where NAME is the name of the directory you want to import.

Import all packages

Then to import all packages, run the command:

```
anaconda-enterprise-cli admin import site-dump/*
```

As you can see from the glob operator (`site-dump/*`) you must pass a list of users you want to import.

What this script does:

For every user in a 4.x Repository, it will create a new channel for each label the user has, using the username as a prefix.

EXAMPLE:

Let's say the user `anaconda-user-1` has the following packages:

- `moto-0.4.31-2.tar.bz2` with label "main"
- `pysocks-1.6.6-py35_0.tar.bz2` with label "test"

For this user, the script creates the following channels:

- `anaconda-user-1` with file `moto-0.4.31-2.tar.bz2`
- `anaconda-user-1/test` with file `pysocks-1.6.6-py35_0.tar.bz2`

The default label "main" is dropped in preference of just keeping the username as channel name.

Import an organization

For each organization you want to import, the script adds the organization groups to the channel name it creates.

EXAMPLE:

Let's say `anaconda-organization` has a group called "Devs" and there are packages there such as `xz-5.2.2-1.tar.bz2` which has label "Test".

This script creates the following channels:

- anaconda-organization - This contains everything the Owner group has
- anaconda-organization/Devs - This contains everything that is available for the Dev group
- anaconda-organization/Devs/Test - This contains everything in the Dev group with label "Test".

So for an organization, group, and label a new channel is created. The default labels "main" and group "Owner" are dropped in preference of shortening names.

After everything is uploaded, the channels are shared with the users. That is the channel `anaconda-user-1` is made read-writable by `anaconda-user-1`. The members of a group are given read permission on the organization's channel.

1.8.2 Migrating AE Notebook 4.x Projects

If you have been an Anaconda Enterprise Notebooks user, you are likely to have some 4.x projects that you would like to use in Enterprise v5.

Before you start

If your project contains several notebooks, check that they all are using the same kernel/environment.

NOTE: These instructions are for notebooks that use the same kernel or environment only.

On your AE Notebook server

1. Log onto your Notebooks server.
2. Open a Terminal window and activate the environment that contains your project.
3. Install anaconda project in the environment:

   ```
   conda install anaconda-project=0.6.0
   ```

   If you get a `not found` message, install it from Anaconda.org:

   ```
   conda install -c anaconda anaconda-project=0.6.0
   ```

4. Export your environment to a file:

   ```
   `conda env export -n default -f _env.yml`
   ```

   `<default>` is the name of the environment where the notebook runs.

5. Check the format of the environment to be sure it looks like this, without any warning exclamation points:

   ```
   yaml
   channels:
   - wakari
   - r
   - https://conda.anaconda.org/wakari
   - defaults
   - anaconda-adam
   prefix: /projects/anaconda/MigrationExample/envs/default
   dependencies:
   - _license=1.1=py27_1
   - accelerate=2.3.1=np111py27_0
   - accelerate_cudalib=2.0=0
   - alabaster=0.7.9=py27_0
   # ... etc ...
   ```

   NOTE: Check to see if your file contains warning exclamation marks like this example:

```
yaml
name: default
channels:
- wakari
- r
- https://conda.anaconda.org/wakari
- defaults
- anaconda-adam
dependencies:
- _license=1.1=py27_1'
# ...
```

If you see any warning exclamation points, run this script to modify the encoding and remove the warnings:

```
import ruamel_yaml
with open("_env.yml") as env_fd:
    env = ruamel_yaml.load(env_fd)
with open("environment.yml", "w") as env_fd:
    ruamel_yaml.dump(env, env_fd, Dumper=ruamel_yaml.RoundTripDumper)
```

6. Create a Project compatible for Enterprise v5:

```
anaconda-project init
```

7. Remove the platforms, Windows and macOS from the file `anaconda-project.yml` since AEN is linux only. Run:

```
anaconda-project remove-platforms win-64 osx-64
```

Verify using:

```
anaconda-project list-platforms
```

8. Lock the project dependencies:

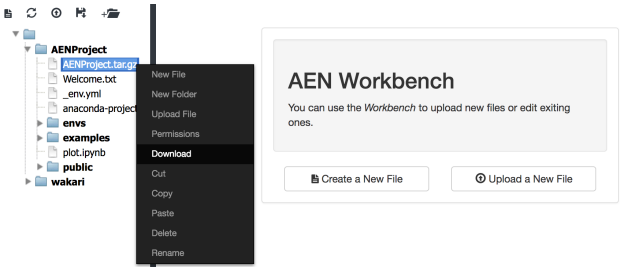
```
anaconda-project lock
```

9. Compress your project:

```
anaconda-project archive WAKARI_PROJECT_NAME.tar.gz
```

Note: If you get a permission denied error for `.indexer.pid`, add `/.indexer.pid` to the `.projectignore` file. To remove all `.git` directories, add `.git` to the `.projectignore` file.

10. In Anaconda Enterprise Notebooks, from your project home page, open the Workbench. In the file list, locate your file `WAKARI_PROJECT_NAME.tar.gz`. To download the project, right-click the file name and select Download.



Upload to Enterprise version 5

Log onto the Enterprise v5 interface and upload FILENAME.tar.gz:

On the Projects page, click the Add (+) button, click on "Upload project" and select your file.

Troubleshooting

- Does your project refer to channels in your on-premises repository or other channels in anaconda.org? Those channels should be migrated to Enterprise by your system administrator.
- Does your project use packages which are not in anaconda? If so, upload those packages to AE v5.
- Does your notebook refer to multiple kernels/environments? Try setting the kernel to just one environment.

1.9 Configuring Spark/Hadoop

To support your organization's data analysis operations, Anaconda Enterprise enables you to optionally connect to remote Apache Hadoop or Spark clusters.

Connection requirements

These are the supported configurations for connecting to remote Hadoop and Spark clusters with Anaconda Enterprise.

software	version
Hadoop and HDFS	2.6.0+
Spark and Spark API	1.6+ and 2.X
Sparkmagic	0.12.5
Livy	0.5
Hive	1.1.0+
Impala	2.11+

Anaconda Enterprise connects to Spark with Livy, authenticated and authorized by MIT Kerberos 5 or the Kerberos version installed on the cluster. AE also authenticates to HDFS with Kerberos. Kerberos Impersonation must be enabled.

The Hive metastore may be Postgres or MySQL.

Installing Livy

This is a recipe for installing Livy into an existing Spark cluster. This recipe is specific to a Red Hat-based Linux distribution, with a Hadoop installation based on Cloudera CDH. To use other systems, you will need to look up corresponding commands and locations.

The Livy server must run on an "edge node" or client in the Hadoop/Spark cluster. The spark-submit command and/or the spark `repl` commands must be known to work on this machine.

Locate the directory that contains Anaconda Livy. Typically this will be `anaconda-enterprise-X.X.X-X.X/anaconda-livy-0.5.0`, where `X.X.X-X.X` are the Anaconda Enterprise version numbers.

Copy the entire directory that contains Anaconda Livy to an edge node on the Spark/Hadoop cluster.

To start the Livy server, set the following environment variables. These can be put into a user's `.bashrc` file, or within the file `conf/livy-env.sh` in the livy directory.

These values are accurate for a Cloudera install of Spark with Java version 1.8:

```
export JAVA_HOME=/usr/java/jdk1.8.0_121-cloudera/jre/
export SPARK_HOME=/opt/cloudera/parcels/CDH/lib/spark/
export SPARK_CONF_DIR=$SPARK_HOME/conf
export HADOOP_HOME=/etc/hadoop/
export HADOOP_CONF_DIR=/etc/hadoop/conf
```

The Livy server itself is configured by editing the file `conf/livy.conf`. Many options can be configured in this file; see the in-line documentation. For example, the port that is defined here (parameter `livy.server.port`) is the same which will generally appear in the Sparkmagic user configuration, above.

The minimum required parameter is `livy.spark.master`. Other possible values include `local[*]` (for testing), `yarn-cluster` for using with the YARN resource allocation system, or a full spark URI like `spark://masterhost:7077` if the spark scheduler is on a different host.

Example with YARN:

```
livy.spark.master = yarn-cluster
```

The YARN deployment mode is set to `cluster` for Livy. The file `livy.conf`, typically located in `$LIVY_HOME/conf/livy.conf`, may include settings similar to the following:

```
livy.server.port = 8998
# What spark master Livy sessions should use: yarn or yarn-cluster
livy.spark.master = yarn
# What spark deploy mode Livy sessions should use: client or cluster
livy.spark.deployMode = cluster

# Kerberos settings

livy.server.auth.type = kerberos
livy.impersonation.enabled = true

# livy.server.launch.kerberos.principal = livy/$HOSTNAME@ANACONDA.COM
# livy.server.launch.kerberos.keytab = /etc/security/livy.keytab
# livy.server.auth.kerberos.principal = HTTP/$HOSTNAME@ANACONDA.COM
# livy.server.auth.kerberos.keytab = /etc/security/httplivy.keytab

# livy.server.access_control.enabled = true
# livy.server.access_control.users = livy,hdfs,zeppelin
# livy.superusers = livy,hdfs,zeppelin
```

Finally, start the Livy server:

```
./bin/livy-server
```

This may be done within some process control mechanism, to ensure that the server is reliably restarted in the event of a crash.

Configuring Livy to use HTTPS

If you want to use Sparkmagic to communicate with Livy via https, you need to do the following to configure Livy as a secure endpoint:

- Generate a keystore file, certificate, and truststore file for the Livy server.
- Update Livy with the keystore details.
- Update your Sparkmagic configuration.

- Restart the Livy server.

1. Generate a keystore file for Livy server using the following command:

```
keytool -genkey -alias <host> -keyalg RSA -keysize 1024 -dname CN=<host>,OU=hw,
↪O=hw,L=paloalto,ST=ca,C=us -keypass <keyPassword> -keystore <keystore_file> -
↪storepass <storePassword>
```

2. Create a certificate:

```
keytool -export -alias <host> -keystore <keystore_file> -rfc -file <cert_file> -
↪storepass <StorePassword>
```

3. Create a truststore file:

```
keytool -import -noprompt -alias <host> -file <cert_file> -keystore <truststore_
↪file> -storepass <truststorePassword>
```

4. Update `livy.conf` with the keystore details. For example:

```
livy.keystore = /home/centos/livy-0.5.0-incubating-bin/keystore.jks
livy.keystore.password = anaconda
livy.key-password = anaconda
```

5. Update `~/.sparkmagic/config.json`. For example:

```
  "kernel_python_credentials" : {
    "username": "",
    "password": "",
    "url": "https://35.172.121.109:8998",
    "auth": "None"
  },
"ignore_ssl_errors": true,
```

NOTE: In this example, `ignore_ssl_errors` is set to `true` because this configuration uses self-signed certificates. Your production cluster setup may be different.

CAUTION: If you misconfigure a `.json` file, all Sparkmagic kernels will fail to launch. You can test your Sparkmagic configuration by running the following Python command in an interactive shell:

```
python -m json.tool config.json
```

If you have formatted the JSON correctly, this command will run without error. Additional edits may be required, depending on your Livy settings. See *Working with Spark/Hadoop* for more information on working with Sparkmagic.

6. Restart the Livy server.

The Livy server should now be accessible over https. For example, `https://<livy host>:<livy port>`.

To test your SSL-enabled Livy server, run the following Python code in an interactive shell to create a session:

```
livy_url = "https://<livy host>:<livy port>/sessions"
data = {'kind': 'spark', 'numExecutors': 1}
headers = {'Content-Type': 'application/json'}
r = requests.post(livy_url, data=json.dumps(data), headers=headers,␣
↪auth=HTTPKerberosAuth(mutual_authentication=REQUIRED,     sanitize_mutual_error_
↪response=False), verify=False)
r.json()
```

Run the following Python code to verify the status of the session:

```
session_url = "https://<livy host>:<livy port>/sessions/0"
headers = {'Content-Type': 'application/json'}
r = requests.get(session_url, headers=headers, auth=HTTPKerberosAuth(mutual_
↪authentication=REQUIRED, sanitize_mutual_error_response=False), verify=False)
r.json()
```

Then submit the following statement:

```
session_url = "https://<livy host>:<livy port>/sessions/0/statements"
data ={"code": "sc.parallelize(1 to 10).count()"}
headers = {'Content-Type': 'application/json'}
r = requests.get(session_url, headers=headers, auth=HTTPKerberosAuth(mutual_
↪authentication=REQUIRED, sanitize_mutual_error_response=False), verify=False)
r.json()
```

Using Kerberos

If the Hadoop cluster is configured to use Kerberos, to allow Livy to access the services do the following:

Generate 2 keytabs for Apache Livy using `kadmin.local`.

IMPORTANT: These are hostname and domain dependent. Edit according to your Kerberos settings:

```
$ sudo kadmin.local

kadmin.local:  addprinc livy/ip-172-31-3-131.ec2.internal
WARNING: no policy specified for livy/ip-172-31-3-131.ec2.internal@ANACONDA.COM;␣
↪defaulting to no policy
Enter password for principal "livy/ip-172-31-3-131.ec2.internal@ANACONDA.COM":
Re-enter password for principal "livy/ip-172-31-3-131.ec2.internal@ANACONDA.COM":
kadmin.local:  xst -k livy-ip-172-31-3-131.ec2.internal.keytab livy/ip-172-31-3-131.
↪ec2.internal@ANACONDA.COM
...

kadmin.local:  addprinc HTTP/ip-172-31-3-131.ec2.internal
WARNING: no policy specified for HTTP/ip-172-31-3-131.ec2.internal@ANACONDA.COM;␣
↪defaulting to no policy
Enter password for principal "HTTP/ip-172-31-3-131.ec2.internal@ANACONDA.COM":
Re-enter password for principal "HTTP/ip-172-31-3-131.ec2.internal@ANACONDA.COM":
kadmin.local:  xst -k HTTP-ip-172-31-3-131.ec2.internal.keytab HTTP/ip-172-31-3-131.
↪ec2.internal@ANACONDA.COM
...
```

This will generate two files: `livy-ip-172-31-3-131.ec2.internal.keytab` and `HTTP-ip-172-31-3-131.ec2.internal.keytab`.

NOTE: You must change the permissions of these two files so they can be read by the `livy-server`.

Enable Kerberos authentication and reference these two keytab files in the `conf/livy.conf` configuration file, as shown:

```
livy.server.auth.type = kerberos
livy.impersonation.enabled = false  # see notes below

# principals and keytabs to exactly match those generated before
livy.server.launch.kerberos.principal = livy/ip-172-31-3-131@ANACONDA.COM
livy.server.launch.kerberos.keytab = /home/centos/conf/livy-ip-172-31-3-131.keytab
livy.server.auth.kerberos.principal = HTTP/ip-172-31-3-131@ANACONDA.COM
livy.server.auth.kerberos.keytab = /home/centos/conf/HTTP-ip-172-31-3-131.keytab
```

(continues on next page)

(continued from previous page)

```
# this may not be required when delegating auth to kerberos
livy.server.access_control.enabled = true
livy.server.access_control.users = livy,zeppelin,testuser
livy.superusers = livy,zeppelin,testuser

livy.server.launch.kerberos.principal = livy/ip-172-31-3-131@ANACONDA.COM
livy.server.launch.kerberos.keytab = /home/centos/conf/livy-ip-172-31-3-131.keytab
livy.server.auth.kerberos.principal = HTTP/ip-172-31-3-131@ANACONDA.COM
livy.server.auth.kerberos.keytab = /home/centos/conf/HTTP-ip-172-31-3-131.keytab
```

NOTE: The hostname and domain are not the same; verify that they match your Kerberos configuration.

Impersonating users

Impersonation allows you to log in as another user to troubleshoot a problem they may be having.

If impersonation is enabled, any user executing a Spark session must be able to log in on every machine in the Spark cluster, so it must exist in all the nodes.

If impersonation is not enabled, the user executing the livy-server (`livy`) must exist on every machine.

You can add this user to each machine by running this command on each node:

```
sudo useradd -m livy
```

NOTE: If you have any problems configuring Livy, try setting the log level to `DEBUG` in the `conf/log4j.properties` file.

1.10 Upgrading between versions of AE5

The upgrade process varies slightly, depending on your current version and which version you're installing. To update an existing Anaconda Enterprise installation to a newer version, follow the process that corresponds to your particular scenario:

- *Upgrading from AE 5.1.2 or 5.1.3 to 5.2.0, or 5.2.0 to 5.2.x*
- *Upgrading from AE 5.1.0 to 5.1.2*
- *Upgrading from AE 5.0.x to 5.1.x*

CAUTION: After the back up process has begun, it won't be possible to back up data for any open sessions or deployments. We therefore recommend that you ask all users to save their work, stop any sessions and deployments, and log out of the platform during the upgrade window. If they don't, they will lose any usaved work. They may also encounter a 404 error after the upgrade. The workaround for the error message is to stop and restart the session or deployment that generated the error, but there is no way to retrieve lost data.

1.10.1 Upgrading from AE 5.1.2 or 5.1.3 to 5.2.x

In-place upgrades from 5.1.x to 5.2.x are not supported. To update an existing Anaconda Enterprise installation to 5.2.x, you'll need to follow this process:

The specific steps for each stage in the process are outlined in the sections below:

1. *Back up your Anaconda Enterprise configuration and data files*.
2. *Uninstall *all* nodes—master and workers*.

3. *Install Anaconda Enterprise 5.2.x*.
4. *Restore your Anaconda Enterprise configuration and data files from the backup*.
5. *Verify that all AE pods are healthy*.
6. *Update the Anaconda Enterprise server settings URLs*.
7. *Verify that your installation was successful*.

Stage 1 – Back up Anaconda Enterprise

Before you begin any upgrade, you must back up your Anaconda Enterprise configuration and data files.

NOTE: *After installing AE 5.2.x*, you'll need to re-configure your SSL certificates, so ensure all certificate-related information—including the private key—is accessible at that point in the process.

All of the following commands should be run on the *master* node.

1. Copy the `backup.sh` script from the location where you saved the installer tarball to the Anaconda Enterprise environment using the following command:

```
sudo cp backup.sh /opt/anaconda
```

2. Back up Anaconda Enterprise by running the following commands:

```
sudo gravity enter
cd /opt/anaconda
bash backup.sh
```

NOTE: The number of channels and packages being backed up will impact the amount of free space and time required to perform the backup, so ensure you have sufficient free space and time available to complete the process.

The following backup files are created and saved to `/opt/anaconda`:

```
ae5-data-backup-${timestamp}.tar
ae5-state-backup-${timestamp}.tar.gz
```

3. Move the backup files to a remote location to preserve them, as the `/opt/anaconda` directory will be deleted in future steps. *After uninstalling AE*, you'll copy `ae5-data-backup-${timestamp}.tar` back to your local filesystem.
4. Exit the Anaconda Enterprise environment by typing `exit`.

If your existing configuration includes Spark/Hadoop, perform these additional steps to migrate configuration information specific to your cluster:

1. Run the following command to retrieve configuration info. from the 5.1.x server, and generate the `anaconda-config-files-secret.yaml` file:

```
kubectl get secret anaconda-config-files -o yaml > <path-to-anaconda-config-files-
↪secret.yaml>
```

2. Move this file to a remote location to preserve it, as it will be deleted in future steps. Ensure that you can access this file from the server where you're installing the newer version of AE 5.2.x.
3. Open the `anaconda-config-files-secret.yaml` file, locate the `metadata` section, and delete everything under it *except for the following*: `name: anaconda-config-file`.

For example, if it looks like this to begin with:

```
apiVersion: v1
data:
  xxxx
kind: Secret
metadata:
  creationTimestamp: 2018-07-31T19:30:54Z
  name: anaconda-config-files
  namespace: default
  resourceVersion: "981426"
  selfLink: /api/v1/namespaces/default/secrets/anaconda-config-files
  uid: 3de10e2b-94f8-11e8-94b8-1223fab00076
type: Opaque
```

It will look like this afterwards:

```
apiVersion: v1
data:
  xxxx
kind: Secret
metadata:
  name: anaconda-config-files
type: Opaque
```

Stage 2 – Uninstall Anaconda Enterprise

1. Uninstall *all worker nodes*, by running the following commands from a shell *on each worker node*:

```
sudo gravity leave --force
sudo killall gravity
sudo killall planet
```

2. Now you can uninstall *the master node*, by running the following commands:

```
sudo gravity system uninstall
sudo killall gravity
sudo killall planet
sudo rm -rf /var/lib/gravity /opt/anaconda
```

3. Reboot *all nodes* to ensure that any Anaconda Enterprise state is flushed from your system.

Stage 3 – Install Anaconda Enterprise

CAUTION: You must use the same FQDN used in your 5.1.x installation for your 5.2.x installation.

1. Download the latest installer file using the following command:

```
curl -O <link-to-installer>
```

2. Follow the *installation instructions*—including the post-install configuration steps—to install Anaconda Enterprise.

If you encounter any errors, refer to *Installation requirements* for the instructions to update your environment so that it meets the AE installation requirements, and restart the install.

NOTE: After ensuring your environment meets all requirements, if you still see a `Cannot continue` error, restart the install.

3. Before restoring your data, log in to the platform to verify that the installation completed successfully, then log out again.

Stage 4 – Restore data

Copy the `restore.sh` script from the location where you saved the installer tarball to the Anaconda Enterprise environment using the following command:

```
sudo cp restore.sh /opt/anaconda
```

When upgrading from 5.1.x to 5.2.x, we recommend restoring backup *data only*, as new state information will be generated during the installation of 5.2.x.

In the terminal, run the following commands:

```
sudo gravity enter
cd /opt/anaconda/
bash restore.sh <path-to-data-backup-file>
```

NOTE: Replace `path-to-data-backup-file` with the path to the data backup file generated when you ran the Anaconda Enterprise backup script in step 1 of Stage 1 above.

For help, run the `bash restore.sh -h` command.

Stage 5 – Verify pod status

Restoring AE data deletes and restarts the associated pods. Follow this process to ensure the new pods are healthy:

1. Login to the Anaconda Enterprise Operations Center with the same username and password you used for the 5.1.x Operations Center.
2. Select **Kubernetes** in the left menu and click on the **Pods** tab.
3. Verify that the **Status** of all pods says `Running`.

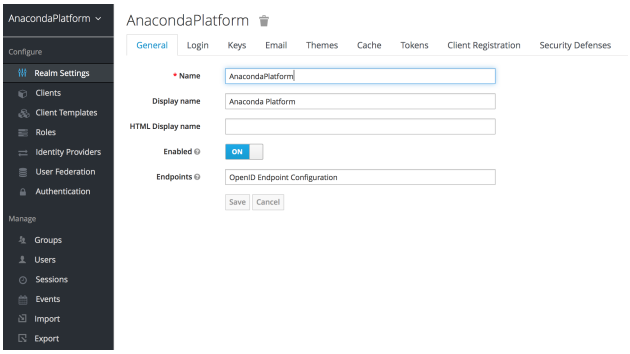
You can also use the following command to watch for status updates:

```
watch kubectl get pods --all-namespaces
```

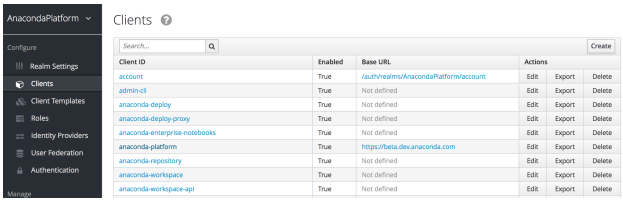
NOTE: It may take up to 15 minutes for their status to change from `Pending` to `Running`.

Stage 6 – Edit redirect URLs

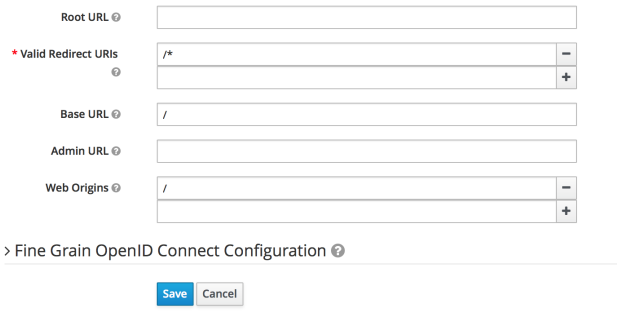
1. When all pods are running, access the Anaconda Enterprise Authentication Center by visiting this URL in your browser: `https://example.anaconda.com/auth/`—replacing `example.anaconda.com` with the FQDN of your server—and clicking the **Administration Console** link.
2. Login with the same username and password you used for the 5.1.x Authentication Center.



3. Verify that `AnacondaPlatform` is displayed as the current realm, then select **Clients** from the **Configure** menu on the left.



4. In the **Client** list, click `anaconda-platform` to display the platform settings.
5. On the **Settings** tab, update all URLs in the following fields with the FQDN of the Anaconda Enterprise server, or the following symbols:



NOTE: If you provide the FQDN of your AE server, be sure each field still ends with the symbols shown. For example, the **Valid Redirect URIs** would look something like this: `https://server-name.domain.com/*`

6. Click **Save** to update the server with your changes.

Stage 7 – Verify installation

After you've verified that all pods are running and updated the Anaconda Enterprise URLs, you can confirm that your upgrade was successful by doing the following:

1. Return to the Authentication Center and select **Users** in the **Manage** menu on the left.
2. Click **View all users** and verify that all user data has also been restored.
3. Access the Anaconda Enterprise user console by visiting this URL in your browser: `https://example.anaconda.com/`—replacing `example.anaconda.com` with the FQDN of your server—and logging in using the same credential you used in your previous installation.
4. Review the **Projects** list to verify that all project data has been restored.

NOTE: If you didn't configure SSL certificates as part of the post-install configuration, do so now. See *Managing TLS/SSL certificates* for more information.

If you're upgrading a Spark/Hadoop configuration:

After you successfully *restore your Anaconda Enterprise data*, run the following commands *on the master node* of the newly-installed Anaconda Enterprise server:

```
kubectl replace -f <path-to-anaconda-config-files-secrets.yaml>
```

To verify that your configuration upgraded correctly:

1. Log in to Anaconda Enterprise.
2. *If your configuration uses Kerberos authentication*, open a Hadoop terminal and authenticate yourself through Kerberos using the same credentials you used previously. For example, `kinit <username>`.
3. Open a Jupyter Notebook that uses Sparkmagic, and verify that it behaves as expected. For example, run the `sc` command to connect to Sparkmagic and start Spark.

1.10.2 Upgrading from AE 5.1.0 to 5.1.2

This in-place upgrade process is recommended, as it requires almost no downtime.

1. Download the 5.1.2 installer file.
2. Add OS settings required for 5.1.2:

   ```
   sudo sysctl -w fs.may_detach_mounts=1
   sudo sysctl -w net.bridge.bridge-nf-call-iptables=1
   sudo sysctl -w net.ipv4.ip_forward=1
   ```

   Add settings to `/etc/sysctl.conf`:

   ```
   net.ipv4.ip_forward = 1
   net.bridge.bridge-nf-call-iptables = 1
   fs.may_detach_mounts = 1
   ```

3. Run `sudo ./upgrade`
4. Update the version of the app images in the configmap.

   First, edit the configmap:

   ```
   sudo gravity enter
   kubectl edit cm
   ```

   Next, in the configmap, update the app images to the new version of the images in the installer:

   ```
   data:
    anaconda-platform.yml
      images:
        app: apiserver:5000/ap-app:5.1.2-0
        app_proxy: apiserver:5000/ap-app-proxy:5.1.2-0
        editor: apiserver:5000/ap-editor:5.1.2-0
   ```

5. Restart all Anaconda Enterprise pods:

   ```
   kubectl get pods | grep ap- | cut -d' ' -f1 | xargs kubectl delete pods
   ```

1.10.3 Upgrading from AE 5.0.x to 5.1.x

Upgrading your Anaconda Enterprise installation from version 5.0.x to 5.1.x requires the following:

1. Backup all AE data.
2. Uninstall the current version of AE.
3. Install the newer version of AE.

Stage 1 – Backup Anaconda Enterprise

NOTE: All of the following commands should be run on the *master* node.

1. Back up the Anaconda Enterprise configuration:

```
sudo gravity backup anaconda-enterprise-backup.tar.gz
```

2. Ensure all users have saved their work and logged out. To prevent any database transactions, stop the AE database with:

```
sudo gravity enter
kubectl delete deploy postgres
```

3. Exit the Anaconda Enterprise environment by typing `exit`.
4. All of the persistent data in AE is stored on the master node in `/opt/anaconda/storage`, you can backup your data by running the following command:

```
sudo tar -zcvf anaconda-data.tar.gz /opt/anaconda/
```

5. Restart the AE database:

```
sudo gravity enter

kubectl apply -f /var/lib/gravity/local/packages/unpacked/gravitational.io/
↪AnacondaEnterprise/*/resources/postgres.yaml

# Restart service pods
kubectl get pods | grep ap- | cut -d' ' -f1 | xargs kubectl delete pods
```

6. Exit the Anaconda Enterprise environment by typing `exit`.

Stage 2 – Uninstall Anaconda Enterprise

1. To uninstall Anaconda Enterprise on a healthy master node, run:

```
sudo gravity system uninstall
sudo killall gravity
sudo killall planet
sudo rm -rf /var/lib/gravity
```

   If `/var/lib/gravity` is present after the uninstallation, you should reboot your machine and retry the `sudo gravity system uninstall` command.

2. Reboot, to ensure that any Anaconda Enterprise state is flushed from your system.

Stage 3 – Install Anaconda Enterprise

1. Download the installer file for the newer AE version.
2. Follow the *installation instructions* to install Anaconda Enterprise, which will use the existing data in `/opt/anaconda`.
3. Update the Anaconda Enterprise configuration to match the latest *configuration schema*. Note that we do not currently version the schema of the `anaconda-platform.yml`, so there may be incompatible changes between versions.

Check the logs for each service for errors about new or missing fields. If you see any errors, manually update the configuration to match the new schema.

Significant known schema changes, with the version they were added in, are detailed below:

5.1.x

The field format for specifying passive license information has changed. The field `license.client-id` is now `license.number`, and the field `license.client-certificate` is now `license.key`.

4. Ensure that your SSL certificate filenames are correct.

   In Anaconda Enterprise 5.1.0 and newer, the default SSL certificate filenames provided by the installer are different than in previous versions. It is recommended that you update any Kubernetes secrets you created and update the Anaconda Enterprise configuration to match the new filenames.

Previous	Updated
`rootca.pem`	`rootca.crt`
`cert.pem`	`server.crt`
`privkey.pem`	`server.key`
`tls.crt`	`wildcard.crt`
`tls.key`	`wildcard.key`

   NOTE: the `keystore.jks` filename is unchanged.

5. Add roles and associate them with the appropriate users (if upgrading from 5.0.x):

```
ae-admin
ae-creator
ae-deployer
ae-uploader
```

6. Restart all Anaconda Enterprise pods:

```
kubectl get pods | grep ap- | cut -d' ' -f1 | xargs kubectl delete pods
```

1.10.4 Troubleshooting an upgrade

In-place upgrades from a version other than 5.1.0 to 5.1.2

If an attempt was made to perform an in-place upgrade from a version other than 5.1.0 to 5.1.2, the service pods will be in the `ImagePullBackOff` state.

To recover, execute the following command with the correct original version:

```
kubectl get deployments -n default -o yaml | sed \"s/:original-version/:5.1.2-0/g\" |␣
↪kubectl replace -f - && kubectl get pods -n default | grep ap- | cut -d' ' -f1 |␣
↪xargs kubectl delete pods -n default
```

1.11 Backing up and restoring Anaconda Enterprise

Before you begin any upgrade, you must back up your Anaconda Enterprise configuration and data files. You may also choose to back up AE regularly, based on your organization's disaster recovery policies.

CAUTION: After the back up process has begun, it won't be possible to back up data for any open sessions or deployments. We therefore recommend that you ask all users to save their work, stop any sessions and deployments,

and log out of the platform during the upgrade window. If they don't, they will lose any usaved work. They may also encounter a 404 error after the upgrade. The workaround for the error message is to stop and restart the session or deployment that generated the error, but there is no way to retrieve lost data.

If you are performing backing up Anaconda Enterprise as part of an upgrade, note that *after installing AE 5.2.x*, you'll need to re-configure your SSL certificates, so ensure all certificate-related information—including the private key—is accessible at that point in the process. See *upgrading between versions for AE5* for the complete upgrade process.

Backing up Anaconda Enterprise

The number of channels and packages being backed up will impact the amount of free space and time required to perform the backup, so ensure you have sufficient free space and time available to complete the process.

All of the following commands should be run on the master node.

1. Copy the `backup.sh` script from the location where you saved the installer tarball to the Anaconda Enterprise environment using the following command:

```
sudo cp backup.sh /opt/anaconda
```

2. Back up Anaconda Enterprise by running the following commands:

```
sudo gravity enter
cd /opt/anaconda
bash backup.sh
```

The following backup files are created and saved to `/opt/anaconda`:

```
ae5-data-backup-${timestamp}.tar
ae5-state-backup-${timestamp}.tar.gz
```

NOTE: The `/opt/ancaconda` directory will be deleted in future steps, so **move the backup files to a remote location to preserve them**. *After uninstalling AE*, you'll copy `ae5-data-backup-${timestamp}.tar` back to your local filesystem.

3. Exit the Anaconda Enterprise environment by typing `exit`.

If your existing configuration includes Spark/Hadoop, perform these additional steps to migrate configuration information specific to your cluster:

1. Run the following command to retrieve configuration info. from the 5.1.x server, and generate the `anaconda-config-files-secret.yaml` file:

```
kubectl get secret anaconda-config-files -o yaml > <path-to-anaconda-config-files-
↪secret.yaml>
```

NOTE: This file will be deleted in future steps, so **move it to a remote location to preserve it**, and ensure that you can access this file from the server where you're installing the newer version of AE 5.2.x.

2. Open the `anaconda-config-files-secret.yaml` file, locate the metadata section, and delete everything under it except for the following: `name: anaconda-config-file`.

For example, if it looks like this to begin with:

```
apiVersion: v1
data:
  xxxx
kind: Secret
metadata:
  creationTimestamp: 2018-07-31T19:30:54Z
```

(continues on next page)

(continued from previous page)

```
  name: anaconda-config-files
  namespace: default
  resourceVersion: "981426"
  selfLink: /api/v1/namespaces/default/secrets/anaconda-config-files
  uid: 3de10e2b-94f8-11e8-94b8-1223fab00076
type: Opaque
```

It will look like this afterwards:

```
apiVersion: v1
data:
  xxxx
kind: Secret
metadata:
  name: anaconda-config-files
type: Opaque
```

Restoring Anaconda Enterprise

If you backed up your Anaconda Enterprise installation, you can restore configuration information from the backup files. The restore script restores data, and can be optionally used to restore state information.

NOTE: When upgrading from 5.1.x to 5.2.x, we recommend restoring only *data* from the backup, and using the *state* generated during installation of 5.2.0. See *upgrading between versions for AE5* for the complete upgrade process.

Copy the `restore.sh` script from the location where you saved the installer tarball to the Anaconda Enterprise environment using the following command:

```
sudo cp restore.sh /opt/anaconda
```

To restore only data, run:

```
sudo gravity enter
cd /opt/anaconda/
bash restore.sh <path-to-data-backup-file>
```

NOTE: Replace `path-to-data-backup-file` with the path to the data backup file generated when you ran the Anaconda Enterprise backup script.

To restore data and state, run:

```
sudo gravity enter
cd /opt/anaconda/
bash restore.sh <path-to-data-backup-file> <path-to-state-backup-file>
```

For help, run the `bash restore.sh -h` command.

After recovery, manually stop and restart all active sessions and deployments and job runs with the UI.

1.12 Uninstalling AE

Before using the following instructions to uninstall Anaconda Enterprise, be sure to follow the steps to *backup your current installation* so you'll be able to restore your data from the backup after installing Anaconda Enterprise 5.2.

To uninstall Anaconda Enterprise on a healthy cluster worker nodes, run:

```
sudo gravity leave --force
sudo killall gravity
sudo killall planet
```

To uninstall Anaconda Enterprise on a healthy cluster master node, run:

```
sudo gravity system uninstall
sudo killall gravity
sudo killall planet
sudo rm -rf /var/lib/gravity /opt/anaconda
```

To uninstall a failed or faulty cluster node, run:

```
sudo gravity remove --force
```

To remove an offline node that cannot be reached from the cluster, run:

```
sudo gravity remove <node>
```

<node> - specifies the node to be removed and can be either the node's assigned hostname or its IP address (the one that was used as an "advertise address" or "peer address" during install) or its Kubernetes name (can be obtained via kubectl get nodes).

1.13 Implementation examples

1.13.1 Creating a Kerberized EMR cluster for use with AE 5

Objective

In this exercise an AWS EMR cluster with Kerberos will be set up and configured to be used with Anaconda Enterprise v5 for testing of Spark, Hive and HDFS access.

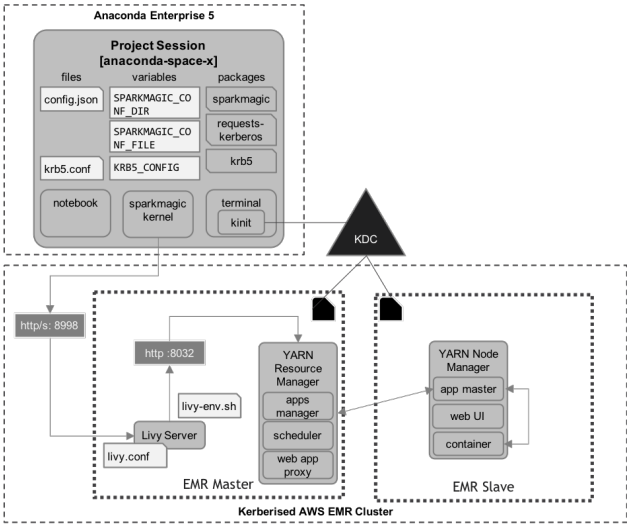
Applicability

- Amazon EMR
- Anaconda Enterprise 5.1.x on AWS

Due to the way public and private addresses are handled in AWS, on premises installs and installs on different cloud providers will have somewhat different configuration.

Implementation

The figure below illustrates the main components of this solution.



Consequences

- This setup uses internal KDC for Kerberos. It is possible to configure cross domain trust with ActiveDirectory.
- This setup requires definition of users on all nodes in the cluster for delegation tokens.
- The described configuration scope is per project. It can be extended to site wide.

Creating the AWS EMR Cluster

Set up AWS EMR security configuration

Go to the Amazon EMR Console: https://console.aws.amazon.com/elasticmapreduce

Confirm that your default home region is selected.

In the left hand pane select “Security configurations”.

Create a security configuration with these options:

- Name: my-emr-seccfg
- Encryption: Leave all of these blank.
- Authentication: Select Kerberos, leave Ticket lifetime at 24 hours and leave Cross-realm trust blank.
- IAM roles for EMRFS: Leave blank. The default roles will be used for the purposes of this exercise.

Click Create. Confirm that the new security configuration is shown in the list.

Create the AWS EMR cluster

Go to the Amazon EMR Console: https://console.aws.amazon.com/elasticmapreduce

Confirm that your default home region is selected.

In the left hand pane select “Clusters”.

Select “Create Cluster” and “Advanced Options”.

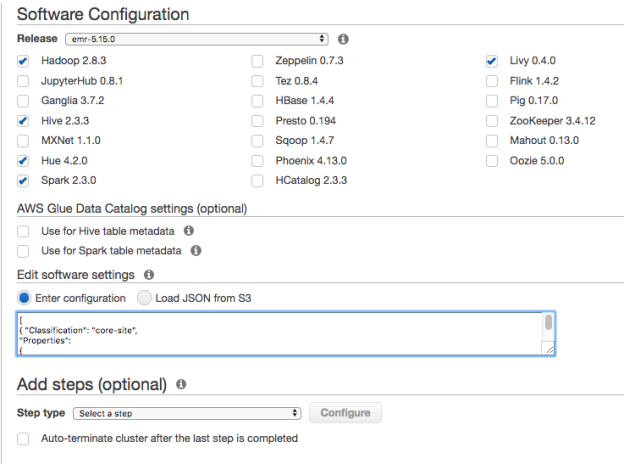
Software

Release: emr-5.15.0

Applications: Hadoop 2.8.3, Hive 2.3.3, Hue 4.2.0, Spark 2.3.0, Livy 0.4.0

Under “Edit software settings” insert this JSON snippet:

```
[
{ "Classification": "core-site",
"Properties":
{
"hadoop.proxyuser.yarn.groups":"*",
"hadoop.proxyuser.yarn.hosts":"*",
"hadoop.proxyuser.livy.groups":"*",
"hadoop.proxyuser.livy.hosts":"*"
}
},
{"Classification":"hadoop-kms-site",
"Properties":
{"hadoop.kms.proxyuser.livy.users":"*",
"hadoop.kms.proxyuser.livy.hosts":"*",
"hadoop.kms.proxyuser.livy.groups":"*"}
}
]
```



Click Next.

Hardware

Instance group configuration: Left as Uniform instance groups.

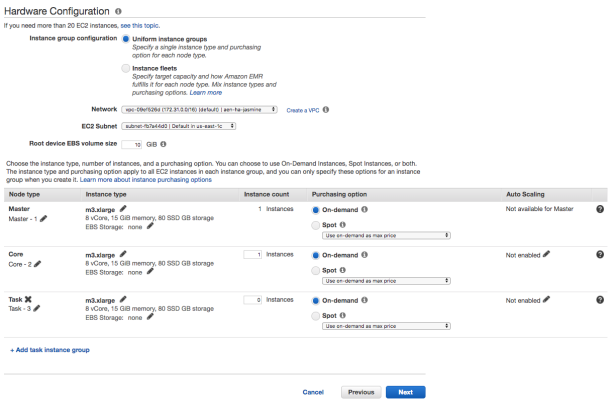
Network: Select your preferred VPC in this box. Save the inbound rules to this VPC for AE5 access later.

EC2 Subnet: Select a subnet from the selected VPC.

Root device EBS volume size: Left at 10GB. This is for test purposes only with the storage in S3 so large volumes should not be required.

Master configuration: Left as default. Instance type is calculated depending on what applications were assigned.

Core configuration: Reduced to 1 instance. This is for connectivity testing primarily.



Click Next.

General cluster settings

General Options: Cluster name set to “my-emr-cluster”.

All others left as default.

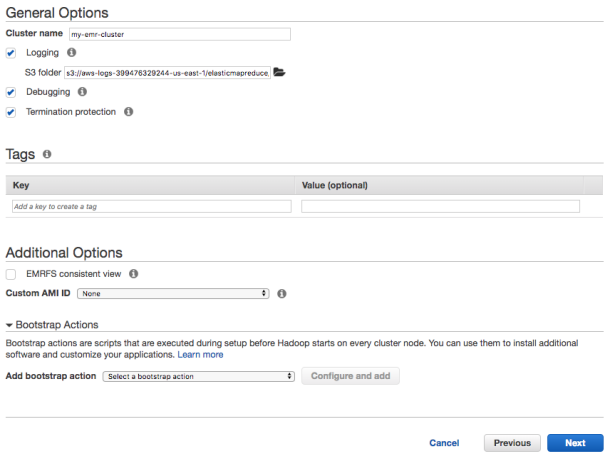
Click Next.

Security

Security Options: Specify your EC2 key pair and leave the option to make “Cluster visible to all IAM users in account”.

Permissions: Set to “Default” which automatically configures the following roles:

- EMR role: EMR_DefaultRole
- EC2 instance profile: EMR_EC2_DefaultRole



- Auto Scaling role: EMR_AutoScaling_DefaultRole

Authentication and encryption: Enter the name of the security configuration defined in *Set up AWS EMR security configuration* which was "my-emr-seccfg".

Realm: MYEMRREALM.ORG

KDC admin password: <kdcpassword>. This is required to add end users to kerberos in the next step.

Create a security group. This group will be used for inbound livy access on port 8998 (the default livy port).

Name tag: emr-livy-sg

Group name: emr-livy

Description: Inbound access for Livy from AE5

VPC: Same as in Hardware section above.

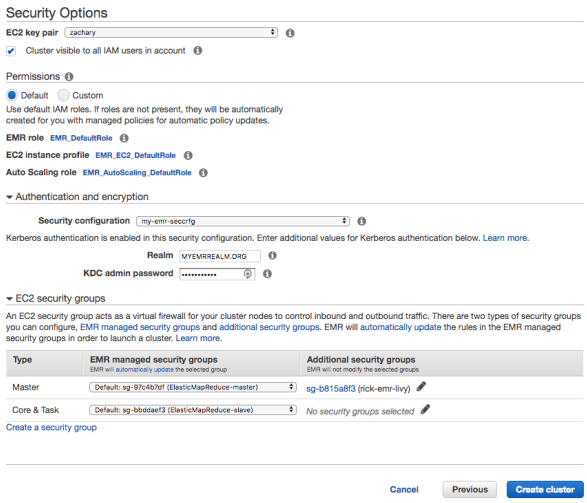
Create the following Inbound Rules:

- Type: Custom TCP Rule
- Protocol: TCP
- Port Range: 8998
- Source: This can be either a port range or a security group. In this case the security group of the AE 5 cluster was used.
- Description: Inbound access for Livy from AE5

Save the security group.

Attach the emr-livy-sg security group to the Additional security groups section for the Master only.

Click Create cluster. This will take 10 minutes or so.



Add end users to Kerberos

In this step user principals will be configured.

Launch an SSH console to the master instance created above.

Add the new principal: `sudo kadmin.local add_principal -pw password rbarthelmie`

Check that the user can log in:

```
kinit rbarthelmie
Password for rbarthelmie@EMR.CONTIUUM.IO:
klist
Ticket cache: FILE:/tmp/krb5cc_500
Default principal: rbarthelmie@MYEMRREALM.ORG
Valid starting       Expires              Service principal
07/09/2018 15:48:05  07/10/2018 01:48:05  krbtgt/MYEMRREALM.ORG@MYEMRREALM.ORG
         renew until 07/10/2018 15:48:05
```

Repeat these steps for each user to add.

Create user home directories

Change to the hdfs account (as root): `su hdfs`

Create the user's home directory: `hdfs dfs -mkdir /user/rbarthelmie`

Change the ownership on the user's home directory: `hdfs dfs -chown rbarthelmie:rbarthelmie /user/rbarthelmie`

Check the home directories for the correct access:

```
hdfs dfs -ls -q /user
Found 9 items
drwxrwxrwx - hadoop      hadoop      0 2018-07-09 13:06 /user/hadoop
drwxr-xr-x - mapred      mapred      0 2018-07-09 13:06 /user/history
drwxrwxrwx - hdfs        hadoop      0 2018-07-09 13:06 /user/hive
drwxrwxrwx - hue         hue         0 2018-07-09 13:06 /user/hue
drwxrwxrwx - livy        livy        0 2018-07-09 13:06 /user/livy
```

(continues on next page)

(continued from previous page)

```
drwxrwxrwx - oozie       oozie       0 2018-07-09 13:06 /user/oozie
drwxr-xr-x - rbarthelmie rbarthelmie 0 2018-07-09 15:52 /user/rbarthelmie
drwxrwxrwx - root        hadoop      0 2018-07-09 13:06 /user/root
drwxrwxrwx - spark       spark       0 2018-07-09 13:06 /user/spark
```

Add local Linux accounts

This step is required due to YARN security requesting access for HDFS delegation tokens by the YARN Application-Master. The identity is required. The credentials are provided by Kerberos.

As root, on both the master and the slave node: `useradd rbarthelmie`

Test Spark and Livy

At this point Spark and Livy can be tested. If these tests do not complete successfully do not proceed further until they are resolved.

Test 1 - Spark submit

This is the simplest test. It uses the master configuration for Hadoop along with the Kerberos credentials of the user that was created earlier (in this case rbarthelmie). After a successful kinit (checked with klist, if required) run the following command on the newly created master: `spark-submit --name SParkPi --master yarn --deploy-mode cluster /usr/lib/spark/examples/src/main/python/pi.py`

If successful one of the final blocks of outputs should read:

```
client token: Token { kind: YARN_CLIENT_TOKEN, service:  }
       diagnostics: N/A
       ApplicationMaster host: 172.31.28.18
       ApplicationMaster RPC port: 0
       queue: default
       start time: 1531152504474
       final status: SUCCEEDED
       tracking URL: http://ip-172-31-20-241.ec2.internal:20888/proxy/application_
↪1531141630439_0005/
       user: rbarthelmie
```

Test 2 - PySpark

In this test the user obtains a token and a file will be uploaded to hdfs. PySpark will be used.

Assume the identity of rbarthelmie and kinit:

```
su rbarthelmie
kinit rbarthelmie
```

Upload a file to hdfs:

```
hdfs dfs -put /etc/hadoop/conf/core-site.xml /user/rbarthelmie
```

Check for its presence:

```
hdfs dfs -ls /user/rbarthelmie
Found 2 items
drwxr-xr-x   - rbarthelmie rbarthelmie    0 2018-07-09 16:15 /user/rbarthelmie/.
↪sparkStaging
-rw-r--r--   1 rbarthelmie rbarthelmie 4836 2018-07-09 16:16 /user/rbarthelmie/core-
↪site.xml
```

Start PySpark:

```
pyspark
```

At the >>> prompt type the following commands:

```
>>> textfile = spark.read.text("rootca.crt")
18/07/09 16:23:06 WARN FileStreamSink: Error while looking for metadata directory.
>>> textfile.count()
193
>>> textfile.first()
Row(value=u'<?xml version="1.0"?>')
```

Test 3 - Livy retrieve sessions

This test ensures that the user can connect to Livy and then onward to the YARN Resource Manager:

```
/usr/bin/curl -X GET --negotiate -u :  http://ip-aaa-bbb-ccc-ddd.ec2.internal:8998/
↪sessions  | python -m json.tool

  % Total %   Received % Xferd  Average Speed  Time     Time     Time     Current
                                 Dload  Upload  Total    Spent    Left     Speed
100 300   100 300      0 0       4323   0       --:--:-- --:--:-- --:--:-- 4347
100  34   100  34      0 0         66   0       --:--:-- --:--:-- --:--:--  112
{
  "from": 0,
  "sessions": [],
  "total": 0
}
```

Test 4 - Livy Create Sessions

This test ensures that the user can connect to Livy and the YARN Resource Manager and then start an application on the YARN Application Master:

```
/usr/bin/curl -X POST --negotiate -u : --data '{"kind":"pyspark"}' -H "Content-
↪Type:application/json" http://ip-aaa-bbb-ccc-ddd.ec2.internal:8998/sessions |␣
↪python -m json.tool
```

If successful the ID of the new application is returned:

```
{
  "appId": null,
  "appInfo": {
       "driverLogUrl": null,
       "sparkUiUrl": null
```

(continues on next page)

(continued from previous page)

```
  },
  "id": 1,
  "kind": "pyspark",
  "log": [
       "stdout: ",
       "\nstderr: ",
       "\nYARN Diagnostics: "
  ],
  "owner": "rbarthelmie",
  "proxyUser": "rbarthelmie",
  "state": "starting"
}
```

This session can be kept track of using the *Livy Retrieve Sessions diagnostic* above.

Configure and test AE5 Kerberos7

In this test the AE5 connectivity to Kerberos will be configured from a user perspective. See the documentation on how to apply Hadoop cluster settings for every user in the role of an administrator. This uses the Hadoop-Spark template that has the correct support for connecting to the Livy Server through the SparkMagic kernel.

In Anaconda Enterprise 5 create a new project.

- Name: project-spark-emr
- Select Project Type: Hadoop-Spark
- Create

In the Projects tab edit the Variable "KRB5_CONFIG" to read:

- Name: KRB5_CONFIG
- Description: Location of config file for Kerberos authentication
- Default: /opt/continuum/project/krb5.conf

Note that the out of the box configuration is /etc/krb5.conf. This location is not editable and will be overwritten if a site wide configuration is set by an administrator. Saving the variable should create or amend the environment variable within the Terminal. This can be checked with:

```
echo $KRB5_CONFIG
/opt/continuum/project/krb5.conf
```

Using a text editor (either Text Editor from the Launcher window or vi from the Terminal) create the file `/opt/continuum/project/krb5.conf`. The server should point to the internal EC2 address of the EMR master (assuming both are in the same VPC) and the domain configured in *Security* above. You can also obtain a working copy of this file from `/etc/krb5.conf` on your EMR master:

```
[libdefaults]
    default_realm = MYEMRREALM.ORG
    dns_lookup_realm = false
    dns_lookup_kdc = false
    rdns = true
    ticket_lifetime = 24h
    forwardable = true
    udp_preference_limit = 1000000
    default_tkt_enctypes = aes256-cts-hmac-sha1-96 aes128-cts-hmac-sha1-96 des3-cbc-
↪sha1
```

(continues on next page)

(continued from previous page)

```
    default_tgs_enctypes = aes256-cts-hmac-sha1-96 aes128-cts-hmac-sha1-96 des3-cbc-
↪sha1
    permitted_enctypes = aes256-cts-hmac-sha1-96 aes128-cts-hmac-sha1-96 des3-cbc-sha1

[realms]
    myemrrealm.org = {
        kdc = ip-aaa-bbb-ccc-ddd.ec2.internal:88
        admin_server = ip-aaa-bbb-ccc-ddd.ec2.internal:749
        default_domain = ec2.internal
   }

[domain_realm]
    .ec2.internal = MYEMRREALM.ORG
     ec2.internal = MYEMRREALM.ORG
[logging]
   kdc = FILE:/var/log/kerberos/krb5kdc.log
    admin_server = FILE:/var/log/kerberos/kadmin.log
    default = FILE:/var/log/kerberos/krb5lib.log
```

With that file in place test your Kerberos configuration with:

```
kinit rbarthelmie
klist
```

There should be a result similar to:

```
Ticket cache: FILE:/tmp/krb5cc_1000
Default principal: rbarthelmie@EMR.CONTIUUM.IO

Valid starting       Expires              Service principal
07/09/2018 20:14:33  07/10/2018 06:14:33  krbtgt/EMR.CONTIUUM.IO@EMR.CONTIUUM.IO
         renew until 07/10/2018 20:14:32
```

Configure and Test AE5 Livy

In this step we will configure the SparkMagic kernel installed in the AE5 project template "Hadoop-Spark" using a configuration file and two environment variables.

In the Projects tab edit the Variable "SPARKMAGIC_CONF_DIR" to read:

- Name: SPARKMAGIC_CONF_DIR
- Description: Location of sparkmagic configuration file
- Default: /opt/continuum/project

Note that if the default location `/opt/continuum/.sparkmagic` is used it will be overwritten by a site wide configuration performed by an administrator. Also, as this directory `.sparkmagic` is effectively hidden the configuration file will not be committed with other files during a project commit resulting in its loss on restarting a session.

In the Projects tab confirm that the Variable "SPARKMAGIC_CONF_DIR" reads:

- Name: SPARKMAGIC_CONF_DIR
- Description: Location of sparkmagic configuration file
- Default: /opt/continuum/project

Copy the SparkMagic configuration file template: `cp /opt/continuum/project/spark/sparkmagic_conf.example.json config.json`

Edit the configuration file: `vi config.json`

Change the targets for the Livy Server to the EMR Master and the authentication types to read:

```
"kernel_python_credentials" : {
    "url": "http://ip-aaa-bbb-ccc-ddd.ec2.internal:8998",
    "auth": "Kerberos"
},
"kernel_python3_credentials" : {
    "url": "http://ip-aaa-bbb-ccc-ddd.ec2.internal:8998",
    "auth": "Kerberos"
},
"kernel_scala_credentials" : {
    "url": "http://ip-aaa-bbb-ccc-ddd.ec2.internal:8998",
    "auth": "Kerberos"
},
"kernel_r_credentials": {
    "url": "http://ip-aaa-bbb-ccc-ddd.ec2.internal:8998",
    "auth": "Kerberos"
},
```

From the Launcher open a Notebook with the PySpark3 template.

In the first cell type `sc` and then shift-enter. If successful, the following will be seen in the output cell:

If that is working correctly then you can continue to explore Spark interactively. If something is not right see the next section for more details on how to identify and remediate errors.

Errors and Resolutions

Looking for Errors

Look for more detail on errors in the following locations:

- Notebooks - You can retrieve the log information for the current Livy session using the special "magic" by typing `%%logs` into a cell and then executing the cell.
- Terminal - The log directory is shown under the "handlers" section of the SparkMagic configuration file. The default location is `/opt/continuum/.sparkmagic-logs/logs`.
- Livy Server - The default log location is `/var/log/livy/livy-livy-server.out`. The most relevant errors can be found with the tag "stdout: ERROR:".
- Kerberos Server - The default log location is `/var/log/kerberos/krb5kdc.log`.

- S3 Log URI - This link can be found in the cluster summary on the Amazon EMR Console. It will be of the format `s3://aws-logs-nnnnnnnnnnnn-us-east-1/elasticmapreduce/` and a link will be provided. The cluster ID is also displayed on the cluster summary if you have multiple clusters. Use the following locations:
  - cluster/ node / (master instance id) / applications / hadoop-yarn / yarn-yarn-resourcemanager
  - cluster/ node / (slave instance id) / applications / hadoop-yarn / yarn-yarn-resourcemanager / yarn-yarn-nodemanager
- Application history - This can be found in the Amazon EMR console under the Applications history tab. Only successfully submitted applications will be logged here.

Common Errors

Error

User: livy/ip-aaa-bbb-ccc-ddd.ec2.internal@MYEMRREALM.ORG is not allowed to impersonate rbarthelmie

Location

`%%logs` and Livy Server

Remediation

Check `/etc/hadoop-kms/conf.empty/kms-site.xml` for these entries:

```
<property>
    <name>hadoop.kms.proxyuser.livy.users</name>
    <value>*</value>
</property>
<property>
    <name>hadoop.kms.proxyuser.livy.hosts</name>
    <value>*</value>
</property>
<property>
    <name>hadoop.kms.proxyuser.livy.groups</name>
    <value>*</value>
</property>
```

And `/etc/hadoop/conf/` for these:

```
<property>
    <name>hadoop.proxyuser.livy.hosts</name>
    <value>*</value>
</property>
<property>
    <name>hadoop.proxyuser.livy.groups</name>
    <value>*</value>
</property>
```

Correct any omissions or mistakes and restart the impacted services on both the master and the slave nodes.

Error

```
INFO LineBufferedStream: stdout: main : requested yarn user is rbarthelmie
INFO LineBufferedStream: stdout: User rbarthelmie not found
```

Location

Livy Server

Remediation

Linux user does not exist on the node. `useradd '<user>'` with root on all nodes.

Error

HTTP ERROR 401

Location

`%%logs`

Livy Server

Remediation

Usually this is a failure to `kinit` prior to attempting to start a session.

Error

Server not found in Kerberos database

Location

Kerberos Server

Remediation

This will be a mismatch between the sparkmagic configuration and the list of principals in Kerberos (these can be found using `kadmin.local list_principals`). Check for the use of AWS internal and external hostnames and ensure there is communication and resolution of these names between the configurations.

Error

GSS initiate failed [Caused by GSSException: No valid credentials provided (Mechanism level: Failed to find any Kerberos tgt)]

Location

`%%logs`

Livy Server

Remediation

HDFS cannot obtain a delegation token as its keytab has expired. To fix:

```
kinit hdfs/ip-aaa-bbb-ccc-ddd.ec2.internal@MYEMRREALM.ORG -k -t /etc/hdfs.keytab
```

Error

INFO LineBufferedStream: stdout: Caused by: org.apache.hadoop.ipc.RemoteException(org.apache.hadoop.security.AccessControlExc
Permission denied: user=rbarthelmie, access=WRITE, inode="/user":hdfs:hadoop:drwxr-xr-x

Location

`%%logs`

Livy Server

Remediation

This is due to the lack of a home directory to write results for user rbarthelmie. Fix with:

```
su hdfs
hdfs dfs -mkdir /user/rbarthelmie
hdfs dfs -chown -R rbarthelmie:rbarthelmie /user/rbarthelmie
hdfs dfs -ls /user
```

Restarting Services

If configuration changes are made it will be necessary to restart the hadoop services that are impacted. A list of services can be obtained from the `initctl list` command.

Slave Node Services:

```
sudo stop hadoop-hdfs-datanode
sudo start hadoop-hdfs-datanode
sudo stop hadoop-yarn-nodemanager
sudo start hadoop-yarn-nodemanager
```

Master Node Services:

```
sudo stop hive-server2
sudo start hive-server2
sudo stop hadoop-mapreduce-historyserver
sudo start hadoop-mapreduce-historyserver
```

(continues on next page)

(continued from previous page)

```
sudo stop hadoop-yarn-timelineserver
sudo start hadoop-yarn-timelineserver
sudo stop hive-hcatalog-server
sudo start hive-hcatalog-server
sudo stop livy-server
sudo start livy-server
sudo stop hadoop-yarn-resourcemanager
sudo start hadoop-yarn-resourcemanager
sudo stop hadoop-kms
sudo start hadoop-kms
sudo stop hue
sudo start hue
sudo stop hadoop-httpfs
sudo start hadoop-httpfs
sudo stop oozie
sudo start oozie
sudo stop hadoop-yarn-proxyserver
sudo start hadoop-yarn-proxyserver
sudo stop spark-history-server
sudo start spark-history-server
sudo stop hadoop-hdfs-namenode
sudo start hadoop-hdfs-namenode
```

Further Reading

- SparkMagic Examples: Magics in iPython, PySpark and Spark Kernels
- How do I restart a service in Amazon EMR?: Identify how to restart services in the cluster
- Run Common Data Science Packages on Anaconda and Oozie with Amazon EMR

CHAPTER 2

Data science workflows

2.1 Working with projects

Anaconda Enterprise makes it easy for you to create and share interactive data visualizations, live notebooks or machine learning models built using popular libraries such Python, r, Bokeh and Shiny.

AE uses *projects* to encapsulate all of the components necessary to use or run an application: the relevant packages, channels, scripts, notebooks and other related files, environment variables, services and commands, along with a configuration file named `anaconda-project.yml`. For more information, see *Configuring project settings*.

Project components are all compressed into a `.tar.bz2`, `.tar.gz` or `.zip` file to make the project portable–so it's easier to store and share with others.

To get you started, Anaconda Enterprise provides several sample projects, including the following:

- Anaconda Distribution for Python 2.7, 3.5 and 3.6
- A REST API written in a Jupyter Notebook
- R notebooks & R Shiny apps
- Bokeh applications for clustering and cross filtering data
- TensorFlow apps for Flask, Tornado and MNIST trained data

You can access them by clicking the Sample gallery icon  from within your **Projects** list.

You can copy any of these sample projects to your project list, and open a session  to make changes to the project.

To work with the contents offline, you can download  the compressed file and then upload it to work with it within AE.

You can also create new—or upload existing—projects  to add them to the server. To update the server with your changes, you *commit your changes to the project*.

NOTE: To maintain performance, there is a 1GB file size limit for project files you upload. Anaconda Enterprise projects are based on Git, so we recommend you commit only text-based files relevant to a project, and keep them under

100MB. Binary files are difficult for version control systems to manage, so we recommend using storage solutions designed for that type of data, and connecting to those data sources from within your Anaconda Enterprise sessions.

Select **Projects** from the top menu to work with your projects:

- Use the **Activity** tab to view a log of all actions performed on the project.
- Use the **Jobs** tab to *schedule deployment jobs* to run on the project.
- Use the **Share** tab to *share the project* with collaborators.
- Use the **Settings** tab to change the default editor–Jupyter Notebook–for the project, select a resource profile that meets your requirements for the project, associate the project *with an external Git repository*, or delete the project.

NOTE: Deleting a project is irreversible, and therefore can only be done if it is not *shared* and has no active *editing sessions* or *deployments*.

2.1.1 Connecting to an external Git repository

Anaconda Enterprise enables you to associate one or more Git repositories with a project, so you can work directly with Git without having to leave the platform. Any Git repositories you associate with the project will be available in the `/opt/continuum/repos` directory within running sessions and deployments.

To be able to associate a Git repository with a project, you first need to configure Git connectivity *across all of your projects* by *adding your Git credentials to your AE account*.

To associate a specific Git repository with a project:

1. Open the project you want to associate a Git repository with by clicking on it in the **Projects** list.
2. Click **Settings** in the menu on the left.
3. Under **External git repositories**, click **Add**.
4. Enter the URL where the remote Git repository is located, as well as the path to your working directory—this is the directory under `/opt/continuum/repos` where the Git repository will be accessed within a session or deployment.
5. Click **Add**.

Now when you work in that project, you'll be able to open a Terminal window and run commands on the repository, separate from the work you're doing with the project files in the AE repository.

2.1.2 Configuring project settings

To enable Anaconda Enterprise to manage the dependencies for your project—so you can run it and deploy it—you need to configure the following settings for each project you create or upload:

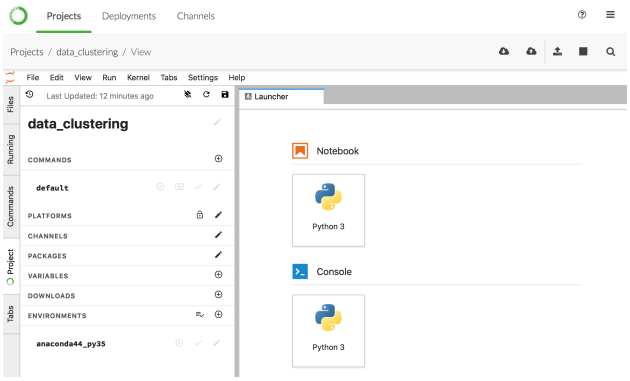
- Include all the packages used by the project (e.g., conda, pip, system).
- Identify the environment required to run the project (e.g., Python version, Hadoop-Spark, R, or SAS).
- Specify the deployment command required to run the project.

All dependencies are tracked in a project's `anaconda-project.yml` file. While there are various ways to modify this file—using the user interface or a command-line interface—any changes to a project's configuration will persist for future project sessions and deployments, regardless of the method you use.

This is different than using `conda install` to add a package using the conda environment during a session, as that method impacts the project temporarily, during the current session only.

NOTE: In a Jupyter Notebook editor session, only `anaconda-project` commands are supported. To use the UI to configure your project settings, you'll need to change the default editor from Jupyter Notebook to JupyterLab. Use the project's **Settings** tab to do this.

Here is an example of a Jupyterlab session, where the UI is available to configure the project:



Adding packages to a project

Anaconda Enterprise offers several ways to add packages to a project, so you can choose the method you prefer:

- In an editing session, click the **Project** tab on the far left and click the Edit pencil icon in the **PACKAGES** field. Add your packages and click **Save**.

–or–

- In a terminal run `anaconda-project add-packages` followed by the package names and optionally the versions.

  EXAMPLE: `anaconda-project add-packages bokeh=0.12 pandas`

The command may take a moment to run as it collects the dependencies and downloads the packages. The packages will be visible in the project's `anaconda-project.yml` file. If this file is already open, close it and reopen it to see your changes.

NOTE: You must include the `notebook` package for the environment to be available to run your Notebook projects and the `bokeh` package to be able to run your Bokeh projects.

To install packages *from a specific channel*:

EXAMPLE: `anaconda-project add-packages -c anaconda-enterprise anaconda-enterprise-web-publisher`

To install *all* the packages in a channel:

- In an editing session, click the **Project** tab on the far left and click the Edit pencil icon in the **CHANNELS** field. Add the channel and click **Save**.

–or–

- In a terminal run `anaconda-project add-packages` followed by the channel name.

  EXAMPLE: `anaconda-project add-packages bokeh -c https://conda.anaconda.org/pyviz`

NOTE: The default `channel_alias` for conda in Anaconda Enterprise is configured to point to the internal package repository, which means that short channel names will refer to channels in the internal package repository. If you wish to use packages from an external or online package repository, you will need to specify the full channel URL such as `anaconda-project add-packages bokeh -c https://conda.anaconda.org/pyviz` in a command or in `anaconda-project.yml`. The `channel_alias` can be *customized by an administrator*, which affects all sessions and deployments.

To install pip packages:

- List the packages in the `pip:` section of `anaconda-project.yml`. For example:

```
packages:
 - six>=1.4.0
 - gunicorn==19.1.0
 - pip:
   - python-mimeparse
   - falcon==1.0.0``
```

To install system packages:

- In a terminal run `sudo yum install` followed by the package name.

  EXAMPLE: `sudo yum install sqlite`

NOTE: Any system packages you install from the command line are available during the current session only. If you want them to persist, install them programatically.

Adding and editing project environments

You may use either of these methods to specify the environment for a project:

- In an editing session, click the **Project** tab on the far left and click the plus sign to the right of the **ENIVIRONMENTS** field. Choose whether you want to **Prepare all environments** or **Add environments**.

  Select an environment and then select **Run**, **Check** or **Edit**. Running an environment opens a terminal window with that environment active.

  When creating an environment, you may choose to inherit from an existing environment, and choose the environment's supported platforms, its channels, and its packages.

–or–

- You can use the command line to add and edit project environments.

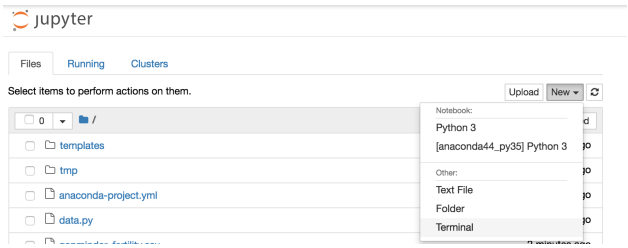
EXAMPLE: To create an environment called new_env with notebook, pandas and bokeh:

```
anaconda-project add-env-spec --name new_env
anaconda-project add-packages --env-spec new_env notebook pandas bokeh=0.12
```

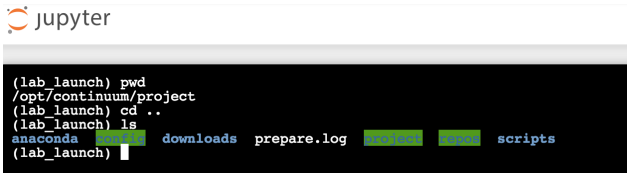
For more information about `anaconda-project` commands type `anaconda-project --help`.

To verify whether an environment has been initialized for a Notebook session:

1. Withing the Notebook session, open a terminal window:



2. Run the following commands to list the contents of the parent directory:



If the environment is being initialized, you'll see a file named `preparing`. When the environment has finished initializing, it will be replaced by a file named `prepare.log`.

Adding deployment commands to a project

You can use Anaconda Enterprise to deploy projects containing notebooks, Bokeh applications, and generic scripts or web frameworks. Before you can *deploy a project*, it needs to have an appropriate deployment command associated with it.

Each of the following methods can be used to add a deployment command in the project's config file `anaconda-project.yml`:

- In an editing session, click the **Project** tab on the far left and click the plus sign to the right of the **COMMANDS** field. Add information about the command and click **Save**.

NOTE: This method is available within the JupyterLab editor only, so you'll need to set that as your default editor—in the project's **Settings**—and restart the project session to see this option in the user interface. The two methods described below do not show notifications in the user interface.

–or–

- Directly edit the `anaconda-project.yml` file.

–or–

- Use the command line interface:

EXAMPLE: `anaconda-project add-command --type notebook default data-science-notebook.ipynb`

The following are example deployment commands you can use:

For a Notebook:

```
commands:
  default:
    notebook: your-notebook.ipynb
```

For a project with a Bokeh app defined in `main.py`:

```
commands:
  default:
    bokeh_app: .
```

For a generic script or web framework, including Python or R:

```
commands:
  default:
    unix: bash run.sh
    supports_http_options: true
```

```
commands:
  default:
    unix: python your-script.py
    supports_http_options: true
```

```
commands:
  default:
    unix: Rscript your-script.R
    supports_http_options: true
```

2.1.3 Editing a project

After you have created and *configured your project* so that it appears in your **Projects** list, you can open a project session to make changes to the project.

To edit a project:

1. Click the **Open session** icon </> to open the project in the editor specified the project.
2. Make your changes to the project, and save them locally. A badge is displayed on the **Commit Changes** icon  to indicate that you've made changes that haven't been committed to the server.
3. When you're ready to update the server with your changes, click the icon to *commit your changes*.

NOTE: This also allows others to access your changes, if the project is shared. See *collaborating on projects* for important things to consider when working with others on shared projects.

4. When you're done working with the project, click the **Stop session** icon  . The session is listed in the **Activity** for the project.

You can also leave a project session open, and click the **Return to session**  when you're ready to resume work.

2.1.4 Saving and committing changes in a project

Saving changes to files within an editor or project is different from committing those changes to the Anaconda Enterprise server.

For example, when you select **File > Save** within an editor, you *save your changes to your local copy of the file* to preserve the work you've done.

When you're ready to update the server with your changes, you *commit your changes to the project*. This also allows others to access your changes, if the project is shared. See *collaborating on projects* for important things to consider when working with others on shared projects.

To commit your changes:

1. Click the **Commit Changes** icon. 
2. Select the files you have modified and want to commit. If a file that you changed isn't displayed in this list, make sure you saved it locally.

   CAUTION: Files names containing unicode characters–special characters, punctuation, symbols—won't display properly in this list and can't be committed to the server, so avoid them when naming files.
3. Enter a message that briefly describes the changes you made to the files or project. This information is useful for differentiating your commit from others.
4. Enter a meaningful label or version number for your project commit in the **Tag** field. You can use tags to create multiple versions of a single project so that you—or collaborators—can easily deploy a specific version of the project. See *deploying a project* for more information.
5. Click **Commit**.

2.1.5 Collaborating on projects

Effective collaboration is key to the success of any enterprise-level project, so it's essential to understand how to work well with others in shared projects.

When a project is shared with collaborators, it means that they have permission to edit the project and commit changes to the master copy on the server while you may be actively working on a local copy.

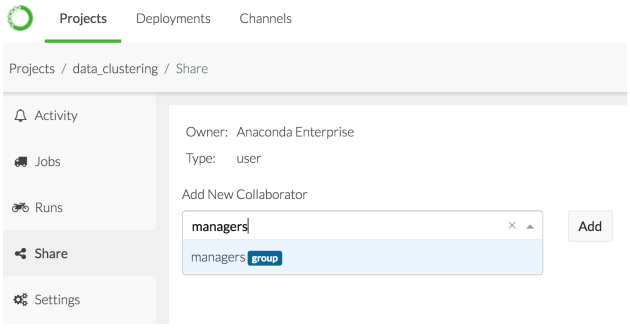
Anaconda Enterprise tracks all changes to a project and lets you know when files have been updated, so you can choose which version to use.

Sharing a project

You can share a project with specific users or groups of users

1. Click **Projects** to view all of your projects.
2. Click the project you want to share and select **Share** in the left menu.
3. Start typing the name of the user or group in the **Add New Collaborator** drop-down to search for matches. Select the one that corresponds to what you want and click **Add**.

To unshare—or remove access to—a project, check the large **X** next to the collaborator you want to remove and click **Remove** to confirm your selection.



Any collaborators you share your project with will see the project in their **Projects** list when they log in to AE, and if others share their projects with you, they'll appear in yours.

Getting updates from other users

When a collaborator makes a change to the project, a badge will appear beside the **Fetch Changes** icon .

Click this icon to pull changes from the server and update your local copy of the project with any changes made by other collaborators.

Anaconda Enterprise compares the copy of the project files you have locally with those on the server and notifies if any files have a conflict. If there is no file conflict, your local copies are updated.

NOTE: Fetching the latest changes may overwrite or delete your local copy of files without warning if a collaborator has committed changes to the server and you have not made changes to the same files, as there is no file conflict.

EXAMPLE:

- Alice and Bob are both collaborators a project that includes `file1.txt`.
- Alice deletes `file1.txt` from her local copy of the project and commits her changes to the server.
- Bob pulls the latest changes from the server. Bob hasn't edited `file1.txt`, so `file1.txt` is deleted from Bob's local version of the project. Bob's local copy of the project and the version on the server now match exactly.

If the updates on the server conflict with changes you have made locally, you can choose one of the following options:

- **Cancel the Pull**.
- **Keep theirs and Pull**—discards your local changes in favor of theirs. Your changes will be lost.
- **Keep mine and Pull**—discards changes on the server in favor of your local changes. Their changes will be overwritten.
- **Keep both and Pull**—saves the conflicting files with different filenames so you can compare the content of the files and decide how you want to reconcile the differences. See *resolving file conflicts* below for more information.

NOTE: If you have a file open that has been modified by fetching changes, close and reopen the file for the changes to be reflected. Otherwise, the next time you save the file, you may see a "File has been overwritten on disk" alert in JupyterLab. This alert lets you choose whether to cancel the save, discard the current version and open the version of the file on disk, or overwrite the file on disk with the current version.

Committing your changes

After you have saved your changes locally, click the **Commit Changes** icon  to update the master copy on the server with your changes.

If your changes conflict with updates made by other collaborators, a list of the files impacted will be highlighted in red. You may choose how you want to proceed from the following options:

- **Cancel the Commit**.
- **Proceed with the Commit**—overwrites your collaborators' changes. Proceed with caution when choosing this option. Collaborators may not appreciate having their work overwritten, and important work may be lost in the process.
- **Selectively Commit**—commit only those files which don't have conflicts by unchecking the ones highlighted in red.

Committing changes to the server involves a full sync, so any changes that have been made to the project on the server–that do not conflict with your changes–are pulled in the process. This means that after committing your changes, your local copy will match the master copy on the server.

Resolving file conflicts

File conflicts result whenever you have updated a file locally, while a collaborator has changed that same file in their copy of the project and committed their changes to the master copy on the server.

In these cases, you may want to select **Keep both and Pull** to save the conflicting files with different filenames. This enables you to compare the content of the files and decide the best approach to take to reconcile the differences. *The solution will likely involve manually editing the file* to combine both sets of changes and then committing the file.

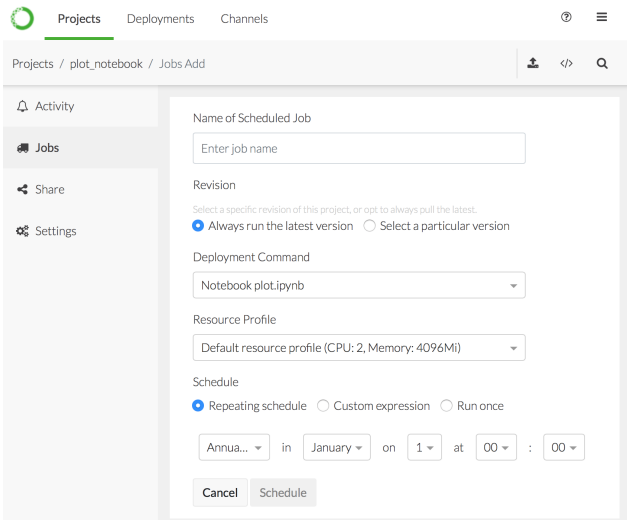
EXAMPLE: If a file is named `Some Data.txt` and Alice has committed updates to that file on the server, your new local copy of the file from the server—containing Alice's changes—will be named `Some Data.txt (Alice's conflicted file)`. Your local copy named `Some Data.txt` will not change.

2.1.6 Scheduling deployment jobs

If you want to *deploy a project* on a regular basis, Anaconda Enterprise enables you to schedule the deployment. This is considered a *job*. For example, you can create jobs to import new data nightly or run resource-intensive deployments after regular business hours.

To schedule a job:

1. Open the project you want to schedule a deployment job for by clicking on it in the **Projects** list.
2. Click **Jobs** in the menu on the left.
3. Click **New** if it's the first job to be created for the project, or the Schedule job  icon if there are existing deployment jobs scheduled.
4. Give the job a meaningful name to help differentiate it from other jobs.
5. Specify whether you want to deploy the latest version of the project, or select a particular version.



6. Specify the **Deployment Command** to use to deploy the project. If there is no deployment command listed, you cannot deploy the project.

   Return to the project and add a deployment command, or ask the project owner to do so if it's not your project. See *Configuring project settings* for more information about adding deployment commands.

7. Choose the runtime resources your project requires to run from the **Resource Profile** drop-down, or accept the default. Your Administrator configures the options in this list, so check with them if you aren't sure.

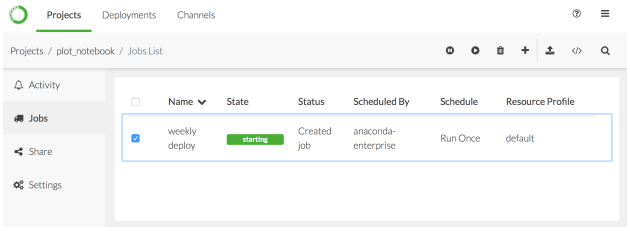
8. Specify how often and when you want the deployment job to run:

- Select **Repeating schedule** and specify how often and when you want the job to run.

—or—

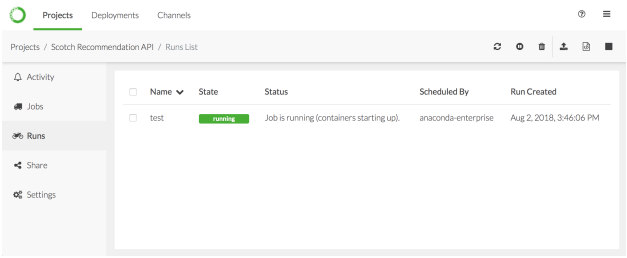
- Select **Custom expression** and enter a valid `cron` command.

9. Click **Schedule** to create the job, and click **Jobs** to view it in the list of jobs for the project.



NOTE: You can use the controls above the **Jobs** list to pause, resume and delete a selected job.

To view a list of all job runs for the project, click **Runs** in the menu on the left.



2.2 Working with deployments

When you *deploy a project*, Anaconda Enterprise finds and builds all of the software dependencies—the libraries on which the project depends in order to run—and encapsulates them, so they are completely self-contained and easy to share with others. This is called a *deployment*.

Whether you deploy a notebook, Bokeh application or *REST API*, everything needed to deploy and run the project is included. You can then *share your deployment* with others so they can interact with it.

NOTE: You can create multiple deployments from a single project. Each deployment can be a different version, and can be shared with different users.

After logging in to Anaconda Enterprise, click **Deployments** to view a list of all of the deployments you have created—or that others have shared with you. Simply click on a deployment to open the deployed Notebook or application and interact with it.

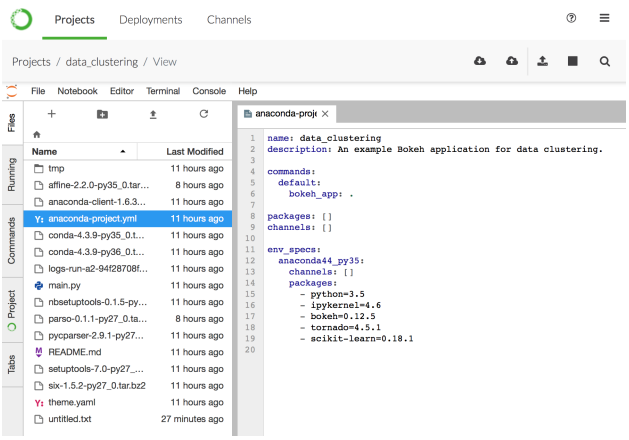
Anaconda Enterprise maintains *a log of all deployments created by all users* in the Administrator's Authentication Center.

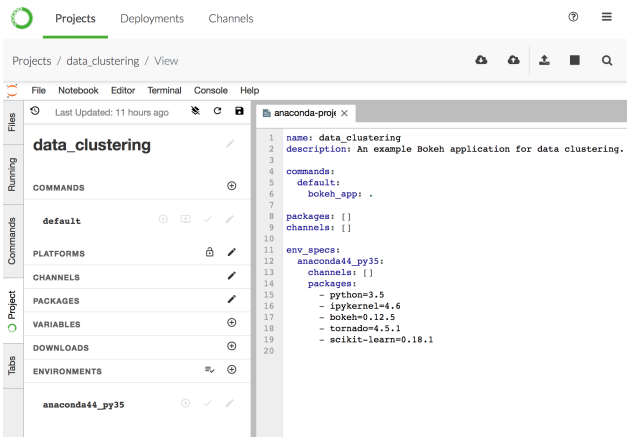
2.2.1 Deploying a project

When you are ready to use your interactive visualization, live notebook or machine learning model, you *deploy* the associated project. You can also deploy someone else's project if you have been added as a collaborator on the project. See *Collaborating on projects* for more information.

When you deploy a project, Anaconda Enterprise finds and builds the software dependencies—all of the libraries required for it to run—and encapsulates them so they are completely self-contained. This allows you to easily share it with others. See *Configuring project settings* for more information about what's required to be able to deploy a project.

You configure how a project is deployed by adding a run command in the configuration file `anaconda-project.yml` for the project and selecting the appropriate deployment command. The following example is a Bokeh app:





To deploy a project:

1. Select it in the **Projects** list and click the **Deploy project** icon .
2. If there are multiple versions of the project, select the version you want to deploy.
3. Select the command to use to deploy the project. If there is no deployment command listed, you cannot deploy the project.

   Return to the project and add a deployment command, or ask the project owner to do so if it's not your project. See *Configuring project settings* for more information about adding deployment commands.
4. Choose the runtime resources your project requires to run from the **Resource Profile** drop-down, or accept the default. Your Administrator configures the options in this list, so check with them if you aren't sure.
5. Click **Deploy**. Anaconda Enterprise displays the status of the deployment, then lists it under **Deployments**.

   NOTE: It may take a few minutes to obtain and build all the dependencies for the project deployment.

To view or interact with the deployment, click its name in the list. Select **Share** to *share it with others*.

Terminating a deployment

When a deployment is no longer required, you can terminate it to stop it from running and remove it from the server to free up resources. Terminating a deployment does not affect the original project from which the deployment was created—only the deployment. It does make the deployment unavailable to any users you had shared it with, however.

To terminate a deployment:

1. Select the deployment in the **Deployments** list and click **Settings** in the menu on the left.
2. Click **Terminate** and confirm that you want to stop the deployment. The deployment stops, and it is removed from list of deployments.

2.2.2 Deploying a REST API

Anaconda Enterprise enables you to deploy your machine learning or predictive models as a REST API endpoint so others can query and consume results from them. REST APIs are web server endpoints—or callable URLs—which provide results based on a query, allowing developers to create applications that programmatically query and consume them via other user interfaces or applications.

Rather than sharing your model with other data scientists and having them run it, you can give them an endpoint to query the model, which you can continue to update, improve and redeploy as needed.

REST API endpoints deployed with Anaconda Enterprise are secure and only accessible to users that you've shared the deployment with or users that have generated a token that can be used to query the REST API endpoint outside of Anaconda Enterprise.

The process of deploying a REST API involves the following steps:

- *Create a project* to encapsulate all of the components necessary to use or run your model.
- *Deploy the project* with the `rest_api` command (shown in Step 4 below) to build the software dependencies—all of the libraries required for it to run—and encapsulate them so they are completely self-contained.
- *Share the deployment* so that other users can obtain the URL of the endpoint and generate a unique token so that they can connect to the deployment and use it programmatically from within notebooks, APIs or other external applications.

Using the API wrapper

As an alternative to using the REST API wrapper provided with Anaconda Enterprise, you can construct an API endpoint using any web framework and serve the endpoint on port 8086 within your deployment, to make it available as a secure REST API endpoint in Anaconda Enterprise.

Follow this process to wrap your code with an API:

1. Open the Jupyter Notebook and add this code to be able to handle HTTP requests. Define a global REQUEST JSON string that will be replaced on each invocation of the API.

```
import json
REQUEST = json.dumps({
    'path' : {},
    'args' : {},
    'body': {}
})
```

2. Import the Anaconda Enterprise `publish` function.

```
from anaconda_enterprise import publish
@publish(methods=['GET', 'POST'])
def function():
    ...
    return json.dumps(...)
```

3. To be able to run, add this to the file `anaconda-project.yml`:

```
commands:
  deploy-api:
    rest_api: {notebook}.ipynb
    supports_http_options: true
    default: true
```

4. Use the following command to test the API within your notebook session (without deploying it):

```
anaconda-project run deploy-api
```

5. Now if you visit `http://localhost:8888/{function}` from within a notebook session you will see the results of your function.

   From within a notebook session, execute the following command:

```
curl localhost:8888/{function}
```

6. Click the **Deploy** icon in the toolbar to deploy the project as an API.

   This deploys the notebook as an API which you can then query.

7. To query externally, create a token and find the url to the running project.

   Example using `curl`:

```
export TOKEN="<generated-token-goes-here>"  # save long string of text in variable
curl -L -H "Authorization: Bearer $TOKEN" <url-of-project>
```

   The `-L` option tells curl to follow redirects. The `-H` adds a header. In this case `-H` adds the token required to authorize the client to visit that URL.

   If you deploy the project as described above you can add the `-X POST` option to `curl` to access that function.

For a complete example of what this looks like, Anaconda Enterprise provides a Jupyter Notebook example called `Scotch Recommendation API`. You can access this sample by clicking the Sample gallery icon from within your **Projects** list and copying it to your projects to edit it. See *Working with projects* for more information.

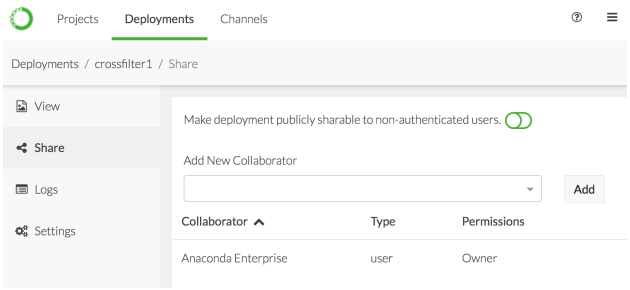
2.2.3 Sharing deployments

After you have *deployed a project*, you can share the deployment with others. You can share a deployment publicly, with other Anaconda Enterprise users, or both. Any collaborators you share your deployment with will see your deployment in their **Deployments** list when they log in to AE.

NOTE: Your Anaconda Enterprise Administrator creates the users and groups with whom you can share your deployments, so check with them if you need a new group created.

To share a deployment:

1. Click **Deployments** to view all of your deployments.
2. Click the deployment you want to share and select **Share** in the left menu.



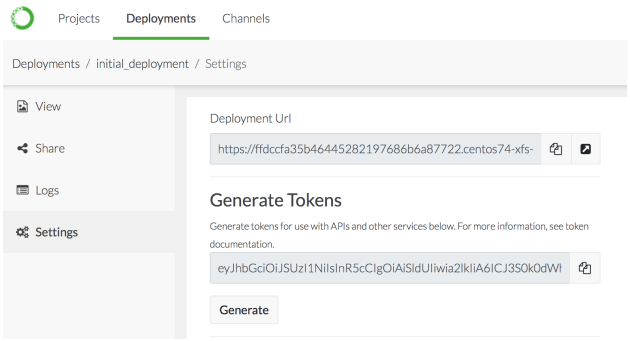
3. To make the deployment available to non-authenticated users—outside of Anaconda Enterprise—enable that toggle/switch. This generates a unique URL that you can copy and distribute to others with whom you want to share the deployment.
4. To share the deployment with other Anaconda Enterprise users, start typing the name of the user or group in the **Add New Collaborator** drop-down to search for matches. Select the one that corresponds to what you want and click **Add**.

To unshare—or remove access to—a deployment, check the large **X** next to the user or group you want to remove as collaborators and and click **Remove** to confirm your selection.

Using your deployment programmatically requires you to *generate a token* for the deployment. This token can then be used to connect to the associated Notebooks, APIs or other running code. Tokens are powerful and should be protected like passwords.

To generate a token for your deployment:

1. Click the deployment you want to generate a token for and select **Settings** in the left menu.
2. Click **Generate** to generate a new token. Copy the token to the clipboard with the  icon, or by copying it with mouse or keyboard shortcuts like any other text.



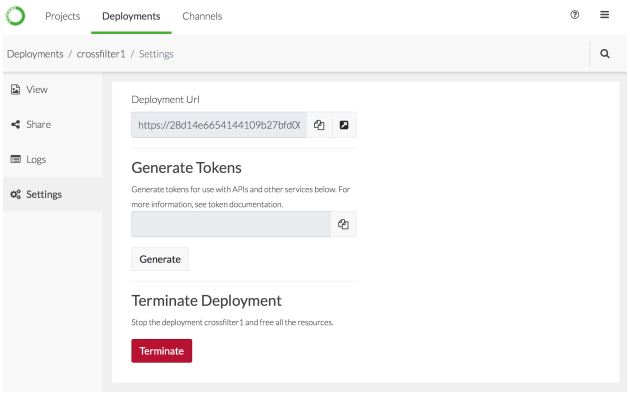
You can then share this token with others to enable them to connect to the deployment from within Notebooks, APIs and other running code.

To remove a deployment from the server—thereby making it unavailable to yourself and others—you *terminate the deployment*. This also frees up its resources.

NOTE: Terminating a deployment does not affect the original project from which the deployment was created, just the deployment itself.

To terminate a deployment:

1. Click the deployment you want to terminate and select **Settings** in the left menu.



2. Click **Terminate** and confirm that you want to remove the deployment from the server.

NOTE: If you terminate and restart a deployment, a new deployment URL will be generated for you to share the new deployment with non-authenticated users.

2.3 Working with packages

Anaconda Enterprise uses *packages* to bundle software files and information about the software—such as its name, specific version and description—into a single file that can be easily installed and managed.

Packages are distributed via *channels*. Channels may point to a cloud-based repository or a private location on a remote or local repository that you or someone else in your organization created. For more information, see *Configuring channels and packages*.

NOTE: Anaconda Enterprise supports the use of both conda and pip packages in its repository.

Creating a package requires familiarity with the conda package manager and command line interface (CLI), so not all AE users will create packages and channels.

Many Anaconda Enterprise users may interact with packages primarily within the context of *projects* and *deployments*. In this case, they will likely do the following:

- Access and download any packages and installers they need from the list of those available under **Channels**.
- Work with the contents of the package as they create models and dashboards, then
- *Add any packages the project depends on* to the project before *deploying it*.

Other users may primarily *build packages*, *upload them to channels* and *share them with others* to access and download.

To create and share channels and packages from your Anaconda Repository using conda commands, first *install anaconda-enterprise-cli* and log in to your AE instance.

2.3.1 Installing Anaconda command line interface (CLI)

Download and install the Anaconda CLI so you can create and share channels and packages from your Anaconda Repository using normal conda commands.

NOTE: This is necessary only the first time you use it.

1. Install `anaconda-enterprise-cli` from your Anaconda Repository:

```
conda install -c https://anaconda.example.com/docs/cli_install/ anaconda-
↪enterprise-cli
```

   NOTE: Replace `anaconda.example.com` with the actual fully qualified domain name (FQDN) of your Anaconda Enterprise instance.

2. Configure `anaconda-enterprise-cli`:

   Add the url of the Anaconda Repository you will be using to the set of sites available to `anaconda-enterprise-cli`:

```
anaconda-enterprise-cli config set sites.example.url https://anaconda.example.com/
↪repository/api
```

   NOTE: Replace `anaconda.example.com` with the actual fully qualified domain name (FQDN) of your Anaconda Enterprise instance.

   Configure it as the default site, the main instance of Anaconda Repository you will be using:

```
anaconda-enterprise-cli config set default_site example
```

3. Check the configuration:

```
anaconda-enterprise-cli config view
```

4. Log into your CLI:

```
anaconda-enterprise-cli login
```

5. Configure the conda channel_alias to retrieve packages from your local Repository:

```
conda config --set channel_alias https://anaconda.example.com/repository/conda
```

   NOTE: Replace `anaconda.example.com` with the actual fully qualified domain name (FQDN) of your Anaconda Enterprise instance.

   TIP: The anaconda-enterprise-cli package is also available on anaconda.org.

6. Log into your CLI using the same username and password that you use in the Enterprise web interface:

```
anaconda-enterprise-cli login
Username: <your-username>
Password: <your-password>
```

You can now create and share channels and packages.

2.3.2 Building a conda package

You can build a conda package to bundle software files and information about the software—such as its name, specific version and description—into a single file that can be easily installed and managed.

Building a conda package requires installing conda build and creating a conda build recipe. You then use the `conda build` command to build the conda package from the conda recipe.

TIP: If you are new to building packages with conda, we recommend you follow the tutorials to get acquainted with the process.

You can build conda packages from a variety of source code projects, most notably Python. For help packaging a Python project, see the Setuptools documentation.

NOTE: Setuptools is a package development process library designed to facilitate packaging Python projects, and is not part of Anaconda, Inc. Conda-build uses the build system that's native to the language, so in the case of Python that's `setuptools`.

After you build the package, you can *upload it to a channel* for others to access.

2.3.3 Uploading a conda package

After you *build a conda package*, you can upload it to a *channel* to make it available for others to use.

A channel is a specific location for storing packages, and may point to a cloud-based repository or a private location on a remote or local repository that you or your organization created. See *Accessing remote package repositories* for more information.

NOTE: There is a 1GB file size limit for package files you upload.

To add a package to an existing channel:

1. Click **Channels** in the top menu to display your existing channels.
2. Select the specific channel you want to add your package to—information about any packages already in the channel is displayed.
3. Click the **Upload package** icon in the upper right corner, browse for the package and click **Upload**. The package is added to the list.

Now you can *share the channel and packages* with others.

To create a new channel to add packages to:

1. Click the **Create Channel** icon  in the upper right corner, enter a meaningful name for the channel and click **Create**.
2. Upload your package to the channel.

Using the CLI:

You can also create a channel by running the following in a terminal window:

```
anaconda-enterprise-cli channels create <channelname>
```

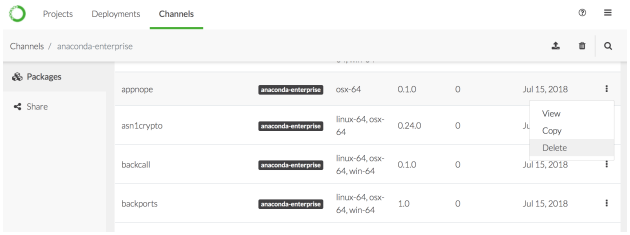
NOTE: The channel name `<channelname>` you enter must not already exist.

Now you can upload a package to the channel by entering the following:

```
anaconda-enterprise-cli upload path/to/pkgs/notebookname.tar.bz2 --channel
↪<channelname>
```

Replacing `path/to/pkgs/notebookname.tar.bz2` with the actual path to the package you want to upload, and `<channelname>` with the actual channel name.

To remove a package from a channel, select **Delete** from the command menu for the package:



NOTE: If the **Delete** command is not available, you don't have permission to remove the package from the channel.

Setting a default channel

There is no `default_channel` in a fresh install, so you'll have to enter a specific channel each time.

If you don't want to enter the `--channel` option with each command, you can set a default channel:

```
anaconda-enterprise-cli config set default_channel <channelname>
```

To display your current default channel:

```
$ anaconda-enterprise-cli config get default_channel
'<channelname>'
```

After setting the default channel, upload to your default channel:

```
anaconda-enterprise-cli upload <path/to/pkgs/packagename.tar.bz2>
```

Replacing `<path/to/pkgs/packagename.tar.bz2>` with the actual path to the package you want to upload.

2.3.4 Sharing channels and packages

After you *build a package* and *upload it* to a channel, you can enable others to access it by *sharing the channel* with them. You can share a channel with specific user or groups of users. The permissions granted to them by the channel owner or an Administrator determine whether they can edit the channel or simply access it.

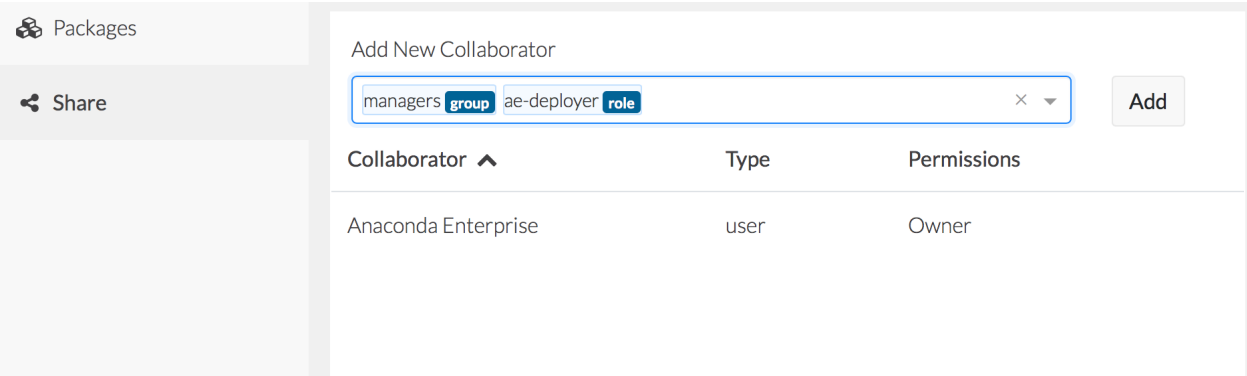
The default is to grant collaborators read-write access, so if you want to prevent them from adding and removing packages from the channel, be sure they have read-only access. You'll need to *use the CLI* to make a channel read-only.

Anyone you share the channel with will see it in their **Channels** list when they log in to Anaconda Enterprise. They can then download the packages in the channel they want to work with, and *add any packages their project depends on* to their project before *deploying it*.

To share multiple packages with the same set of users, you can upload all of the packages to the same channel and share that channel. Otherwise, you can create separate channels and add the appropriate packages to each.

To share the packages in a channel:

1. Select the channel in the **Channels** list and verify that all the packages you want to share are listed.
2. Click **Share** in the left menu.
3. Start typing the name of the user or group in the **Add New Collaborator** drop-down to search for matches. Select the option that corresponds to what you want. You can add multiple users or groups at the same time.
4. Click **Add** when you're satisfied with your selections.



To "unshare" a channel with a collaborator, simply click the large **X** next to the right of their name in the **Collaborator** list.

Using the CLI:

Get a list of all the channels on the platform with the `channels list` command:

```
anaconda-enterprise-cli channels list
```

Share a channel with a specific user using the `share` command:

```
anaconda-enterprise-cli channels share --user username --level r <channelname>
```

You can also share a channel with an existing group created by your Administrator:

```
anaconda-enterprise-cli channels share --group GROUPNAME --level r <channelname>
```

Replacing GROUPNAME with the actual name of your group.

NOTE: Adding `--level r` grants this group read-only access to the channel.

You can "unshare" a channel using the following command:

```
anaconda-enterprise-cli channels share --user <username> --remove <channelname>
```

Run `anaconda-enterprise-cli channels --help` to see more information about what you can do with channels.

For help with a specific command, enter that command followed by `--help`:

```
anaconda-enterprise-cli channels share --help
```

2.4 Configuring your user settings

Anaconda Enterprise maintains settings related to your user account, based on how the system was configured by your Administrator. There are times when you may need to update the information related to your user account—to change your password, for example.

To access your account settings, click the **Menu** icon ☰ in the upper-right corner and select the **Settings** option in the slide-out.

Click **Advanced Settings** to configure the following settings for your Anaconda Enterprise account:

NOTE: Fields that you are not permitted to edit appear grayed / disabled.

- To change the password you use to log in to Anaconda Enterprise, select **Password**.
- To enable two-factor authentication for your account, select **Authenticator**.
- To view a history of your sessions using Anaconda Enterprise, select **Sessions**.
- To view a list of AE applications currently running and the permissions you have been granted, select **Applications**.
- To view a log of all activity related to your account, select **Log**.

To configure Git connectivity within Anaconda Enterprise, click **Add** under **Git Settings** and provide your Git credentials. Your credentials will be available across all projects in your account. See *Connecting to an external Git repository* for more information.

2.5 Using project templates

Anaconda Enterprise provides project templates that demonstrate working with Python, R, Spark and Hadoop clusters, and SAS.

Using GPUs

Anaconda Enterprise also enables you to leverage the compute power of graphics processing units (GPUs) from within your editor sessions. To do so, you use the project's **Settings** tab to *select a resource profile that features a GPU*.

To access a GPU while running a deployed application, you select the appropriate resource profile when you *deploy the associated project*.

In either case, if the resource profile you need isn't listed, ask your Administrator to *configure one for you to use*.

2.5.1 Working with Python

Anaconda Enterprise templates are written in Python by default, and conda supports environments using all versions of Python, including 2.7, 3.5, and 3.6.

The Python 2.7 template includes all of the packages in the Anaconda distribution for Python 2.7. The same is true for the Python 3.5 and 3.6 templates.

Additional conda and pip packages can be added using the processes described in *Configuring project settings*.

Python notebooks can be edited with Jupyter Notebooks or JupyterLab.

2.5.2 Working with R

Anaconda Enterprise supports Microsoft R Open (MRO) and numerous R packages, including most of the most popular R packages on CRAN, the Comprehensive R Archive Network.

The R template includes 128 R packages, including caret, dplyr, ggplot2, glmnet, irkernel, knitr, rbokeh, shiny, tidyverse, and R version 3.4.2. Version 3.4.3 is also available in the Anaconda repository.

By default these packages are based on MRO. Template packages based on the R packages built by Anaconda are also available.

Additional R packages can be added using the processes described in *Configuring project settings*.

In Anaconda Enterprise users can also install R packages from source as usual. This requires that the users are able to connect to the appropriate CRAN repository.

R notebooks can be edited with Jupyter Notebooks or JupyterLab.

2.5.3 Working with Spark/Hadoop

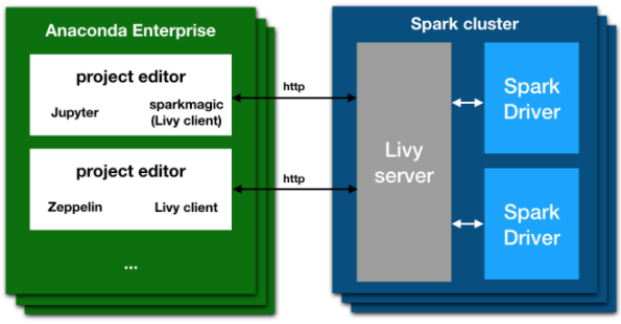
The sections below provide information on *Apache Livy*, *Kerberos*, using them to *connect to a Spark Cluster*, and *connecting to Hive, Impala and HDFS*.

There are also detailed examples and recommendations for *working with Spark libraries using Livy and Sparkmagic*, *working with Impala and Hive from Python*, and *working with Impala and Hive from R*.

Livy

With Anaconda Enterprise, you can connect to a remote Spark cluster using Apache Livy with any of the available clients, including Jupyter notebooks with Sparkmagic. Anaconda Enterprise provides Sparkmagic, which includes Spark, PySpark, and SparkR notebook kernels for deployment.

The Apache Livy architecture gives you the ability to submit jobs from any remote machine or analytics cluster, even where a Spark client is not available. It removes the requirement to install Jupyter and Anaconda directly on an edge node in the Spark cluster.



Livy and Sparkmagic work as a REST server and client that:

- Retains the interactivity and multi-language support of Spark
- Does not require any code changes to existing Spark jobs, and

- Maintains all of Spark's features such as the sharing of cached RDDs and Spark Dataframes
- Provides an easy way of creating a secure connection to a Kerberized Spark cluster.

When Livy is installed, you can connect to a remote Spark cluster when creating a new project by selecting the Spark template.

Kerberos

Many *Hadoop* installations are secured using Kerberos. To authenticate with Kerberos, your system administrator must provide at least one configuration file, normally located at `/etc/krb5.conf`. You need this file to connect to a Kerberized cluster.

To use the `krb5.conf` file, add it your universal project settings. Use the `anaconda-enterprise-cli` tool for this:

```
anaconda-enterprise-cli spark-config --config /etc/krb5.conf krb5.conf
```

NOTE: To use Sparkmagic, you must configure a Sparkmagic configuration file. In this case, pass two flags to the previous command:

```
anaconda-enterprise-cli spark-config --config /etc/krb5.conf krb5.conf \
                  --config /opt/continuum/.sparkmagic/config.json config.json
```

This creates a yaml file: `anaconda-config-files-secret.yaml` with the data converted for AE5.

Next, upload the file:

```
sudo kubectl replace -f anaconda-config-files-secret.yaml
```

With this in place when new projects are created, `/etc/krb5.conf` is populated with the appropriate data.

Authenticating

Contact your administrator to get your Kerberos *principal*, which is the combination of your username and security domain.

To perform the authentication, open an environment-based terminal in the interface. This is normally in the Launchers panel, in the bottom row of icons, and is the right-most icon.

When the interface appears, execute this command:

```
kinit myname@DOMAIN.COM
```

Replace `myname@DOMAIN.COM` with the Kerberos *principal*, the combination of your username and security domain, which was provided to you by your administrator.

Executing the command requires you to enter a password. If there is no error message, authentication has succeeded. You can verify by issuing the `klist` command. If it responds with some entries, authentication has succeeded.

You can also use a keytab to do this. Upload it to a project and execute a command like this:

```
kinit myname@DOMAIN.COM -kt mykeytab.keytab
```

NOTE: Kerberos authentication will lapse after some time, requiring you to repeat the above process. The length of time is determined by your cluster security administration, and on many clusters is set to 24 hours.

Connect to a Spark Cluster

Anaconda Enterprise contains numerous example projects, including a Spark/Hadoop project. This project includes Sparkmagic, so that you can connect to a Spark cluster with a running Livy server.

You can use Spark with Anaconda Enterprise in two ways:

- Starting a notebook with one of the Spark kernels, in which case all code will be executed on the cluster and not locally. Note that a connection and all cluster resources will be assigned as soon as you execute any ordinary code cell, that is, any cell not marked as `%%local`.
- Starting a normal notebook with a Python kernel, and using `%load_ext sparkmagic.magics`. That command will enable a set of functions to run code on the cluster. See examples (external link).

To display graphical output directly from the cluster, you must use SQL commands. This is also the only way to have results passed back to your local Python kernel, so that you can do further manipulation on it with `pandas` or other packages.

In the common case, the configuration provided for you in the Session will be correct and not require modification. However, in other cases you may need to use sandbox or ad-hoc environments that require the modifications described below.

Supported versions

The following combinations of the multiple tools are supported:

- Python 2 and Python 3, Apache Livy 0.5, Apache Spark 2.1, Oracle Java 1.8
- Python 2, Apache Livy 0.5, Apache Spark 1.6, Oracle Java 1.8

Overriding session settings

Certain jobs may require more cores or memory, or custom environment variables such as Python worker settings. The configuration passed to Livy is generally defined in the file `~/.sparkmagic/conf.json`. You may inspect this file, particularly the section `"session_configs"`, or you may refer to the example file in the `spark` directory, `sparkmagic_conf.example.json`. Note that the example file has not been tailored to your specific cluster.

In a Sparkmagic kernel such as PySpark, SparkR, or similar, you can change the configuration with the magic `%%configure`. This syntax is pure JSON, and the values are passed directly to the driver application.

Example:

```
%%configure -f
{"executorMemory": "4G", "executorCores":4}
```

If you are using a Python kernel and have done `%load_ext sparkmagic.magics`, you can use the `%manage_spark` command to set configuration options. The session options are in the "Create Session" pane under "Properties".

Overriding basic settings

In some more experimental situations, you may want to change the Kerberos or Livy connection settings. For example, when first configuring the platform for a cluster. This is likely to be done by an administrator with intimate knowledge of the cluster's security model.

In these cases, we recommend creating a `krb5.conf` file and a `sparkmagic_conf.json` file in the project directory so they will be saved along with the project itself. An example Sparkmagic configuration is included, `sparkmagic_conf.example.json`, listing the fields that are typically set. Of particular importance are the `"url"` and `"auth"` keys in each of the kernel sections.

The `krb5.conf` file is normally copied from the Hadoop cluster, rather than written manually, and may refer to additional configuration or certificate files. These files must all be uploaded using the interface.

To use these alternate configuration files, set the `KRB5_CONFIG` variable default to point to the full path of `krb5.conf` and set the values of `SPARKMAGIC_CONF_DIR` and `SPARKMAGIC_CONF_FILE` to point to the Sparkmagic config file. You can set these either by using the Project pane on the left of the interface, or by directly editing the `anaconda-project.yml` file.

For example, the final file's variables section may look like this:

```
variables:
  KRB5_CONFIG:
    description: Location of config file for kerberos authentication
    default: /opt/continuum/project/krb5.conf
  SPARKMAGIC_CONF_DIR:
    description: Location of sparkmagic configuration file
    default: /opt/continuum/project
  SPARKMAGIC_CONF_FILE:
    description: Name of sparkmagic configuration file
    default: sparkmagic_conf.json
```

NOTE: You must perform these actions **before** running `kinit` or starting any notebook/kernel.

CAUTION: If you misconfigure a `.json` file, all Sparkmagic kernels will fail to launch. You can test your Sparkmagic configuration by running the following Python command in an interactive shell:

```
python -m json.tool sparkmagic_conf.json
```

If you have formatted the JSON correctly, this command will run without error. Additional edits may be required, depending on your Livy settings. See *Configuring Spark/Hadoop* for information on installing and configuring Livy.

Kerberos

To authenticate and connect to a Kerberized Spark cluster, you need the appropriate configuration to execute a `kinit` command in a terminal on the project session.

NOTE: For more information, see the *Kerberos authentication section*.

Managing Kerberos and Sparkmagic Configuration

Both Kerberos and Sparkmagic configuration files are managed with Kubernetes secrets and can easily be created with the `anaconda-enterprise-cli` tool. This allows system administrators to populate Kerberos and Sparkmagic configurations for all projects.

To create a kubernetes secret for both Kerberos and Sparkmagic, execute the following:

```
anaconda-enterprise-cli spark-config --config /etc/krb5.conf krb5.conf \
                  --config /opt/continuum/.sparkmagic/config.json config.json
```

This will create a yaml file `anaconda-config-files-secret.yaml` with the data converted for Kubernetes and AE5. Next, upload the file:

```
sudo kubectl replace -f anaconda-config-files-secret.yaml
```

When new projects are created, `/etc/krb5.conf` and `~/.sparkmagic/conf.json` are populated with the appropriate data.

The Sparkmagic configuration file, commonly `config.json`, must set the `auth` field in the `kernel_python_credentials` section:

```
{
  "kernel_python_credentials" : {
    "url": "http://<LIVY_SERVER>:8998",
    "auth": "Kerberos"
  }
}
```

Finally, in the same `config.json` file set the `home_path` in `handlers` to `~/.sparkmagic-logs`.

Example:

```
"logging_config": {
  "handlers": {
    "magicsHandler": {
      "class": "hdijupyterutils.filehandler.MagicsFileHandler",
      "formatter": "magicsFormatter",
      "home_path": "~/.sparkmagic-logs"
    }
  }
}
```

Using Custom Anaconda Parcels and Management Packs

Anaconda Enterprise provides functionality to generate custom Anaconda parcels for Cloudera CDH or custom Anaconda management packs for Hortonworks HDP, which allows administrators to distribute customized versions of Anaconda across a Hadoop/Spark cluster using Cloudera Manager for CDH or Apache Ambari for HDP.

Data scientists can then select a specific version of Anaconda and Python on a per-project basis by including the following configuration in the first cell in a Sparkmagic-based Jupyter Notebook:.

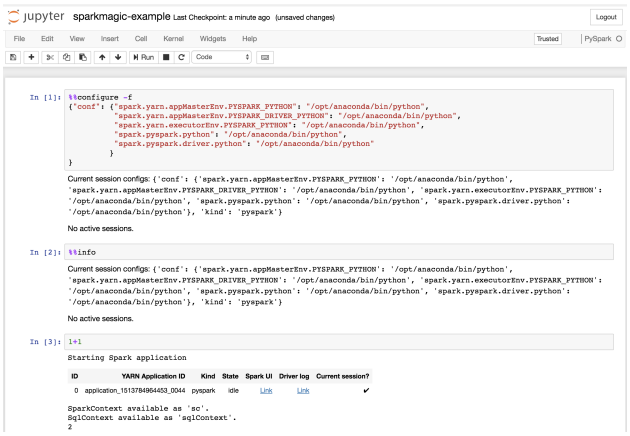
Example:

```
%%configure -f
{"conf": {"spark.yarn.appMasterEnv.PYSPARK_PYTHON": "/opt/anaconda/bin/python",
          "spark.yarn.appMasterEnv.PYSPARK_DRIVER_PYTHON": "/opt/anaconda/bin/python",
          "spark.yarn.executorEnv.PYSPARK_PYTHON": "/opt/anaconda/bin/python",
          "spark.pyspark.python": "/opt/anaconda/bin/python",
          "spark.pyspark.driver.python": "/opt/anaconda/bin/python"
         }
}
```

NOTE: Replace `/opt/anaconda/` with the prefix of the install name and location for the particular Parcel/MPack installed.

More information

Video: https://www.youtube.com/embed/wa514mI7Aw4



Connect to Hive, Impala and HDFS

Anaconda Enterprise contains numerous example projects, including a Spark/Hadoop project. This project includes the libraries needed to connect to Hive, Impala and HDFS with Python libraries, as well as example notebooks to connect to these services.

Impala

To connect to an Impala cluster you need the address and port to a running Impala Daemon, normally port 21050.

To use Impyla, open a Python Notebook based on the `anaconda50_impyla` environment and run:

```
from impala.dbapi import connect
conn = connect('<Impala Daemon>', port=21050)
cursor = conn.cursor()
cursor.execute('SHOW DATABASES')
cursor.fetchall()
```

Hive

To connect to a Hive cluster you need the address and port to a running Hive Server 2, normally port 10000.

To use PyHive, open a Python notebook based on the `anaconda50_hadoop` environment and run:

```
from pyhive import hive
conn = hive.connect('<Hive Server 2>', port=10000)
cursor = conn.cursor()
cursor.execute('SHOW DATABASES')
cursor.fetchall()
```

HDFS

To connect to an HDFS cluster you need the address and port to the HDFS Namenode, normally port 50070.

To use the `hdfscli` command line, configure the `~/.hdfscli.cfg` file:

```
[global]
default.alias = dev

[dev.alias]
url = http://<Namenode>:port
```

Once the library is configured, you can use it to perform actions on HDFS with the command line by starting a terminal based on the `anaconda50_hadoop` environment and executing the `hdfscli` command. For example:

```
$ hdfscli

Welcome to the interactive HDFS python shell.
The HDFS client is available as `CLIENT`.

In [1]: CLIENT.list("/")
Out[1]: ['hbase', 'solr', 'tmp', 'user']
```

Working with Spark libraries using Livy and Sparkmagic

Overview

Python libraries used in an Apache Spark job must be available in the nodes where the Spark driver and workers are executed.

Anaconda Enterprise uses Apache Livy to handle session management and communication to Apache Spark clusters. This provides effective abstraction of the connectivity to multiple Spark clusters, including different versions of Spark, independent clusters and even different types of Hadoop distributions.

Livy provides all the authentication layers that Hadoop administrators are used to, including Kerberos with impersonation.

Recommendation

Anaconda Enterprise provides multiple options for managing the dependencies of Spark jobs. These options include prebuilt Anaconda Parcels and Management Packs, and custom environments that can be shipped to HDFS and used later in Spark jobs. Each method for distributing libraries has its pros and cons, and each organization has its own requirements that limit the methods available.

Anaconda Parcels and Management Packs

Anaconda Parcels are prebuilt distributions of Anaconda specifically created to be used inside a Cloudera CDH cluster.

Anaconda Management Packs are prebuilt distributions of Anaconda specifically created to be used inside a Hortonworks (Apache Ambari) cluster.

Anaconda Enterprise provides the functionality to build as many of these Parcels and Management Packs as you need. Some Anaconda Enterprise customers ship only one or two big parcels that are used by multiple data scientists in multiple Spark jobs, while other customers ship more, depending on their requirements for dependencies.

Anaconda Parcels and Management Packs fit many use cases, and are especially suitable for IT and Hadoop administrators who want to maintain close control of a Hadoop cluster while also making these tools available to data scientists who need Python and R libraries. In this case, Anaconda Parcels and Management Packs provide a very easy way to ship multiple custom Anaconda distributions to multiple Hadoop clusters.

Custom environments

A more advanced user who needs regular updates to the dependencies for a certain job may instead choose to package a Conda environment and upload that environment to HDFS. When starting a Spark job the user can target that Conda environment. In this case Spark automatically extracts the packaged environment into a location where the Spark job can access it.

This environment is accessed in a Spark job by configuring the Sparkmagic Livy client to use it, as shown in the example below.

Anaconda recommends custom environments only if it's not possible to use either Anaconda Parcels or Anaconda Management Packs, or if it's too difficult to update these distributions often enough for the end users.

Example configuration

When using Apache Livy you can set the default Python interpreter for Spark jobs. Set the `PYSPARK_PYTHON` environment variable in the `livy-env.sh` configuration file to the target Python interpreter.

EXAMPLE: Select the Python interpreter in an Anaconda parcel in a CDH cluster:

```
export PYSPARK_PYTHON=/opt/cloudera/parcels/Anaconda-4.4.0/bin/python
```

Anaconda Enterprise connects to the Livy server with the Livy client Sparkmagic. You can use a Sparkmagic configuration to tell Livy which Python interpreter to use for that specific session.

EXAMPLE:

```
%%configure -f
{"conf": {"spark.yarn.appMasterEnv.PYSPARK_PYTHON":
"/opt/anaconda/bin/python",
          "spark.yarn.appMasterEnv.PYSPARK_DRIVER_PYTHON":
"/opt/anaconda/bin/python",
          "spark.yarn.executorEnv.PYSPARK_PYTHON":
"/opt/anaconda/bin/python",
          "spark.pyspark.python": "/opt/anaconda/bin/python",
          "spark.pyspark.driver.python": "/opt/anaconda/bin/python"
         }
}
```

Working with Impala and Hive from Python

Overview

Apache Impala and Apache Hive are distributed SQL engines used by data scientists and data engineers to process large amounts of data quickly. These engines take a SQL query and generate a plan to execute that query in a cluster of nodes. They are common tools in the Apache Hadoop ecosystem and work well with data stored in the Hadoop Distributed File System (HDFS).

With Anaconda Enterprise you can connect to Impala and Hive using Python, R or any programming language with available libraries for connecting to these sources. Both Impala and Hive are very flexible in their connection methods and there are multiple ways to connect to them, such as JDBC, ODBC and Thrift.

Recommendation

Anaconda recommends the Thrift method to connect to both Impala and Hive from Python. With Thrift you can use all the functionality of Impala and Hive, including security features such as SSL connectivity and Kerberos authentication. Thrift does not require special drivers, which improves code portability.

Instead of using an ODBC driver for connecting to the SQL engines, a Thrift client uses its own protocol based on a service definition to communicate with a Thrift server. This definition can be used to generate libraries in any language, including Python.

Multiple libraries can provide connection to Hive and Impala using Python. Anaconda recommends using pyhive to connect to Hive and impyla to connect to Impala. These are the most popular and well-maintained libraries for connecting to these services from Python. Conda packages for these libraries are available.

Example code

This section shows sample code for connecting to Hive and Impala. A template project inside Anaconda Enterprise has similar examples. In these examples we show the arguments for Kerberos authentication and we assume that Anaconda Enterprise has been correctly configured and that a `kinit` command has been successfully executed on the AE session.

Note that the output will be different depending on the tables available on the cluster.

Hive using PyHive:

```
from pyhive import hive

conn = hive.connect('<Hive Server 2>', port=10000, auth='KERBEROS', kerberos_service_
↪name='hive')

cursor.execute('SHOW TABLES')
cursor.fetchall()

# This prints: [('iris',), ('t1',)]

cursor.execute('SELECT * FROM iris')
cursor.fetchall()

# This prints the output of that table
```

Impala using Impyla:

```
from impala.dbapi import connect
conn = connect(host='<Impala Daemon>', port=21050, auth_mechanism='GSSAPI', kerberos_
↪service_name='impala')

cursor = conn.cursor()
cursor.execute('SHOW TABLES')

results = cursor.fetchall()
results

# This prints: [('iris',),]

cursor.execute('SELECT * FROM iris')
cursor.fetchall()
```

(continues on next page)

(continued from previous page)

```
# This prints the output of that table
```

Working with Impala and Hive from R

Overview

Apache Impala and Apache Hive are distributed SQL engines used by data scientists and data engineers to process large amounts of data quickly. These engines take a SQL query and generate a plan to execute that query in a cluster of nodes. They are common tools in the Apache Hadoop ecosystem and work well with data stored in the Hadoop Distributed File System (HDFS).

With Anaconda Enterprise you can connect to Impala and Hive using Python, R or any programming language with available libraries for connecting to these sources. Both Impala and Hive are very flexible in their connection methods and there are multiple ways to connect to them, such as JDBC, ODBC and Thrift.

Recommendation

Anaconda recommends the JDBC method to connect to both Impala and Hive from R. JDBC provides the same functionality as the other connection methods with good flexibility and portability.

Using JDBC requires downloading a driver for the specific version of Hive and Impala that you are using. This driver is also specific to the vendor you are using.

EXAMPLE: For Cloudera:

- Hive JDBC Connector 2.5.4 - Download
- Impala JDBC Connection 2.5.43 - Download

We recommend downloading the respective JDBC drivers and committing them to the project so that they are always available when the project starts.

Once the drivers are located in the project, Anaconda recommends using the RJDBC library to connect to both Hive and Impala. Sample code for this is shown below.

Using JDBC allows for multiple types of authentication including Kerberos. The only difference between the types is that different flags are passed to the URI connection string on JDBC. Please follow the official documentation of the driver you picked and for the authentication you have in place.

EXAMPLE: For Cloudera:

- Hive JDBC Connection 2.5.4 - Documentation
- Impala JDBC Connection 2.5.43 - Documentation

As an additional recommendation specifically for Impala, Anaconda recommends Implyr to manipulate tables from Impala. This library provides a dplyr interface for Impala tables that is familiar to R users. Implyr uses RJBDC for connection so all the recommendations in this document still apply.

Example code

This section shows sample code for connecting to Hive and Impala. A template project inside Anaconda Enterprise has similar examples. In these examples we show the arguments for Kerberos authentication and we assume that

Anaconda Enterprise has been correctly configured and that a `kinit` command has been successfully executed on the AE session.

Note that the output will be different depending on the tables available on the cluster.

Hive using RJDBC:

```
library("RJDBC")

hive_classpath <- list.files("<PATH TO JDBC DRIVER>", pattern="jar$", full.names=T)

drv <- JDBC(driverClass = "com.cloudera.hive.jdbc4.HS2Driver", classPath = hive_
↪classpath, identifier.quote="'")

url <- "jdbc:hive2://<HIVE SERVER 2 HOST>:10000/default;SSL=1;AuthMech=1;KrbRealm=
↪<KRB REALM>;KrbHostFQDN=<KRB HOST>;KrbServiceName=hive"

conn <- dbConnect(drv, url)

dbGetQuery(conn, "SHOW TABLES")

dbDisconnect(conn)
```

Impala using RJDBC and Implyr:

```
library(implyr)
library(RJDBC)

impala_classpath <- list.files(path = "<PATH TO JDBC DRIVER>", pattern = "\\.jar$",␣
↪full.names = TRUE)

drv <- JDBC(driverClass = "com.cloudera.hive.jdbc4.HS2Driver", classPath = hive_
↪classpath, identifier.quote="'")

url <- "jdbc:impala://<IMPALA DAEMON HOST>:10000/default;SSL=1;AuthMech=1;KrbRealm=
↪<KRB REALM>;KrbHostFQDN=<KRB HOST>;KrbServiceName=impala"

# Use implyr to create a dplyr interface

impala <- src_impala(drv, url)

# This will show all the available tables

src_tbls(impala)
```

2.5.4 Working with SAS

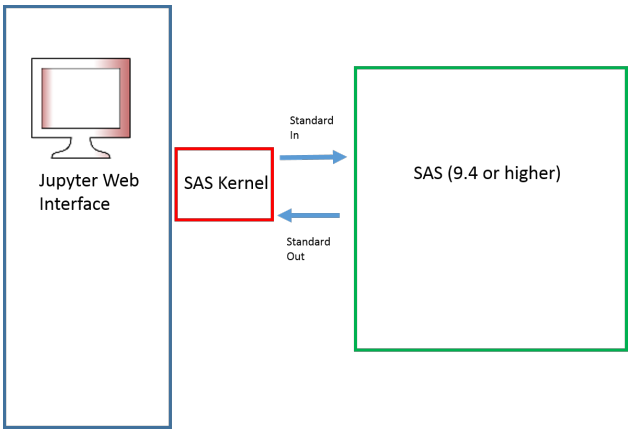
With Anaconda Enterprise, you can connect to a remote SAS server process using the official sas_kernel and saspy. This allows you to merge SAS and Python/R workflows in a single interface, and to share your SAS-based work with your colleagues within the Enterprise platform.

NOTE: SAS sessions are currently available in interactive development mode only, not in deployments.

`sas_kernel` is distributed under the Apache 2.0 Licence, and requires SAS version 9.4, or later. SAS is (c) SAS Institute, Inc.

Anaconda Enterprise and sas_kernel

Anaconda connects to a remote SAS server application over a secure SSH connection.



After you configure and establish the connection with the provided SAS kernel, SAS commands are sent to the remote server, and results appear in your notebook.

NOTE: Each open notebook starts a new SAS session on the server, which stays alive while the notebook is being used. This may affect your SAS license utilization.

Configuration

The file `sascfg_personal.py` in the project root directory provides the configuration for the SAS kernel to run.

Normally your system administrator will provide the values to be entered here.

The connection information is stored in a block like this:

```
default = {
    'saspath' : '/opt/sas9.4/install/SASHome/SASFoundation/9.4/bin/sas_u8',
    'ssh'     : '/usr/bin/ssh',
    'host'    : 'username@55.55.55.55',
    'options' : ["-fullstimer"]
}
```

`'saspath'` must match the exact full path of the SAS binary on the remote system.

`'host'` must be a connection string that SSH can understand. Note that it includes both a login username and an IP or hostname. A successful connection requires that both are correct. The IP or hostname may have an optional suffix of a colon and a port number, so both `username@55.55.55.55` and `username@55.55.55.55:2022` are possible values.

Establishing a Connection

The SAS server machine must allow SSH with password-less login.

Before starting, you will need a secure key (.pem) file installed on the SAS server machine. This will normally be done for you by your organization, but if you have username/password access to the machine, you can make your own *as described below*.

Whenever you start a new editing session, you must perform the following steps before creating or running a notebook with a SAS kernel:

- Upload your .pem file to your editing session. In a JupyterLab or Jupyter notebook, click the left Files tab, then at the top of the pane click the `upload` button and navigate to the file on your system.
- From the New/Launcher pane, open a terminal and run these commands:

```
chmod 0400 <myfile.pem>
ssh-add <myfile.pem>
ssh <connection-string> -o StrictHostKeyChecking=no echo OK
```

  Replace `<connection-string>` with the `host` entry in `sascfg_prsonal.py` and `<myfile.pem>` with the name of your key file.

Now you can start the notebooks with the SAS kernel from the launcher pane, or switch the kernel of any notebook that is already open.

Installing a .pem File

You can create your own .pem file within Anaconda Enterprise 5, or any local machine with openSSH installed. Run this command:

```
ssh-keygen
```

Give your key file a name, typically with the extension .pem. You may give your key file a password or leave the password prompt empty.

The `ssh-keygen` command creates two files.

One file has the exact name you specified. Upload this file into your SAS kernel editing session.

The other file ends in `.pub`, and must be known to the SAS server. You can view this file in the terminal with the `cat` command:

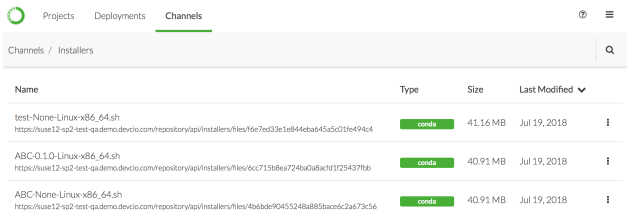
```
cat <myfile.pem.pub>
```

Log in to the SSH server using your username and password. Edit the file `~/.ssh/authorized_keys` and append the contents of your .pem.pub file there. You can edit the file with any console text editor on your system, such as `nano` or `vi`.

2.6 Using installers, parcels and management packs

In addition to Anaconda and Miniconda installers, your Administrator may create custom installers, Cloudera Manager parcels, or Hortonworks Data Manager management packs for you and your colleagues to use. They make these specific packages and their dependencies available to you via *channels*.

To view the installers available to you, select the top **Channels** menu, then click the **Installers** link in the top right corner.



To download an installer, simply click on its name in the list.

NOTE: If you don't see an installer that you expected to see, please contact your Administrator and ask them to *generate the installer* you need.

2.7 Machine learning and deep learning

Anaconda Enterprise faciliates machine learning and deep learning by enabling you to *develop models*, *train them*, *wrap them with an endpoint*, and *deploy them*. You can also use AE to *query and score models* that have been *deployed as a REST API*.

To help get you started, Anaconda Enterprise includes several sample notebooks for common repetitive tasks. You can access them by clicking the Sample gallery icon from within your **Projects** list. See *Working with projects* for more information.

2.7.1 Developing models

Anaconda Enterprise makes it easy for you to create models that you can *train* to make predictions and facilitate machine learning based on deep learning neural networks.

You can *deploy* your trained model as a REST API, so that it can be *queried and scored*.

The following libraries are available in Anaconda Enterprise to help you develop models:

- **Scikit-learn**–for algorithms and model training.
- **TensorFlow**–to express numerical computations as stateful dataflow graphs.
- **XGBoost**–a gradient boosting framework for C++, Java, Python, R and Julia.
- **Theano**–expresses numerical computations & compiles them to run on CPUs or GPUs.
- **Keras**–contains implementations of commonly used neural network building blocks to make working with image and text data easier.
- **Lasagne**–contains recipes for building and training neural networks in Theano.
- **Neon**–deep learning framework for building models using Python, with Math Kernal Library (MKL) support.
- **MXNet**–framework for training and deploying deep neural networks.
- **Caffe**–deep learning framework with a Python interface geared towards image classification and segmentation.
- **CNTK**–cognitive toolkit for working with massive datasets to facilitate distributed deep learning. Describes neural networks as a series of computational steps via a directed graph.

2.7.2 Training models

Anaconda Enterprise provides machine learning libraries such as scikit-learn and Tensorflow that you can use to train the models you create.

To train a model:

When you are ready to run an algorithm against your model and tune it, download the `scikit-learn` or `Tensorflow` package from the `anaconda` channel. If you don't see this channel or these packages in your **Channels** list, you can *mirror these packages* to create a local copy or contact your Administrator to make them available to you.

Serializing your model:

When you are ready to convert your model or application into a format that can be easily distributed and reconstructed by others, use Anaconda Enterprise to *deploy it*.

- YAML – supports non-hierarchical data structures & scalar data
- JSON – for client-server communication in web apps
- HD5 – designed to store large amounts of hierarchical data; works well for time series data (stored in arrays)

NOTE: Your model or app must have been written in a programming language that supports object serialization, such as Python, PHP, R or Java.

2.7.3 Wrapping models with endpoints

Anaconda Enterprise enables you to wrap your machine learning model with an endpoint, so that the appropriate prediction endpoints are included when you *deploy the model* as a web application.

The following Anaconda libraries are provided as frameworks to help you build and deploy models as web apps:

- Anaconda Enterprise API
- Flask
- Django

2.7.4 Deploying models as endpoints

Anaconda Enterprise enables you to deploy machine learning models as endpoints to make them available to others, so the models can be *queried and scored*. You can then save users' input data as part of the training data, and retrain the model with the new training dataset.

Versioning your model:

To enable you to test variations of a model, you can *deploy multiple versions of the model* to the same endpoint. You can then direct different sets of users to each of the versions, to faciliate A/B testing.

Deploying your model as an endpoint:

Deploying a model as an endpoint involves these simple steps:

1. *Create a project* to tell Anaconda Enterprise where to look for the artifacts that comprise the model.
2. *Deploy the project* to build the model and all of its dependencies. Now you—and others with whom you *share the deployment*—can interact with the app, and select different datasets and algorithms.

2.7.5 Querying and scoring models

Anaconda Enterprise enables you to query and score models that have been created in Python, R, or another language such as Curl, CLI, Java or Javascript. The model doesn't have to have been created using AE, as long the model has been *deployed as an endpoint*.

Scoring can be incredibly useful to an organization, including the following "real world" examples:

- By financial institutions, to determine the level of risk that a loan applicant represents.
- By debt collectors, to predict the likelihood of a debtor to repay their debt.
- By marketers, to predict the likelihood of a subscriber list member to respond to a campaign.
- By retailers, to determine the probability of a customer to purchase a product.

A scoring engine calculates predictions or makes recommendataion based on your model. A model's score is computed based on the model and query operators used:

- Boolean queries—specify a formula
- Vector space queries—support free text queries (with no query operators necesssarily connecting them)
- Wildcard queries—match any pattern

Using an external scoring engine

Advanced scoring techniques used in machine learning algorithms can automatically update models with new data gathered. If you have an external scoring engine that you prefer to use on your models, you can do so within Anaconda Enterprise.

2.8 Using statistics

Anaconda Enterprise supports statistical work using the R language and Python libraries such as NumPy, SciPy, Pandas, Statsmodels, and scikit-learn.

The following Jupyter notebook Python examples show how to use these libraries to calculate correlations, distributions, regressions, and principal component analysis.

These examples also include plots produced with the libraries seaborn and Matplotlib.

We thank these sites, from whom we have adapted some code:

- https://stackoverflow.com/questions/25571882/pandas-columns-correlation-with-statistical-significance/49040342
- https://joomik.github.io/Housing/

Start by importing necessary libraries and functions, including Pandas, SciPy, scikit-learn, Statsmodels, seaborn, and Matplotlib.

This code imports `load_boston` to provide the Boston housing dataset from the datasets included with scikit-learn.

```
import pandas as pd
import seaborn as sns
from scipy.stats import pearsonr
from sklearn.datasets import load_boston
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

(continues on next page)

(continued from previous page)

```
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import statsmodels.formula.api as sm

%matplotlib inline
```

Load the Boston housing data into a Pandas DataFrame:

```
#Load dataset and convert it to a Pandas dataframe
boston = load_boston()
df = pd.DataFrame(boston.data, columns=boston.feature_names)
df['target'] = boston.target
```

In the Boston housing dataset, the target variable is MEDV, the median home value.

Print the dataset description:

```
#Description of the dataset
print(boston.DESCR)
```

```
Boston House Prices dataset
===========================

Notes
------
Data Set Characteristics:

:Number of Instances: 506

:Number of Attributes: 13 numeric/categorical predictive

:Median Value (attribute 14) is usually the target

:Attribute Information (in order):
    - CRIM     per capita crime rate by town
    - ZN       proportion of residential land zoned for lots over 25,000 sq.ft.
    - INDUS    proportion of non-retail business acres per town
    - CHAS     Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
    - NOX      nitric oxides concentration (parts per 10 million)
    - RM       average number of rooms per dwelling
    - AGE      proportion of owner-occupied units built prior to 1940
    - DIS      weighted distances to five Boston employment centres
    - RAD      index of accessibility to radial highways
    - TAX      full-value property-tax rate per $10,000
    - PTRATIO  pupil-teacher ratio by town
    - B        1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
    - LSTAT    % lower status of the population
    - MEDV     Median value of owner-occupied homes in $1000's

:Missing Attribute Values: None

:Creator: Harrison, D. and Rubinfeld, D.L.

This is a copy of UCI ML housing dataset.
http://archive.ics.uci.edu/ml/datasets/Housing
```

(continues on next page)

(continued from previous page)

```
This dataset was taken from the StatLib library which is maintained at Carnegie␣
↪Mellon University.

The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic
prices and the demand for clean air', J. Environ. Economics & Management,
vol.5, 81-102, 1978.   Used in Belsley, Kuh & Welsch, 'Regression diagnostics
...', Wiley, 1980.   N.B. Various transformations are used in the table on
pages 244-261 of the latter.

The Boston house-price data has been used in many machine learning papers that␣
↪address regression
problems.

**References**

- Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influential Data and␣
↪Sources of Collinearity', Wiley, 1980. 244-261.
- Quinlan,R. (1993). Combining Instance-Based and Model-Based Learning. In␣
↪Proceedings on the Tenth International Conference of Machine Learning, 236-243,␣
↪University of Massachusetts, Amherst. Morgan Kaufmann.
- many more! (see http://archive.ics.uci.edu/ml/datasets/Housing)
```

Show the first five records of the dataset:

```
#Check the first five records
df.head()
```

```
row CRIM    ZN   INDUS CHAS NOX   RM    AGE  DIS    RAD TAX   PTRATIO B      LSTAT␣
↪target
=== ======= ==== ===== ==== ===== ===== ==== ====== === ===== ======= ====== =====␣
↪======
0   0.00632 18.0 2.31  0.0  0.538 6.575 65.2 4.0900 1.0 296.0 15.3    396.90 4.98  24.
↪0
1   0.02731 0.0  7.07  0.0  0.469 6.421 78.9 4.9671 2.0 242.0 17.8    396.90 9.14  21.
↪6
2   0.02729 0.0  7.07  0.0  0.469 7.185 61.1 4.9671 2.0 242.0 17.8    392.83 4.03  34.
↪7
3   0.03237 0.0  2.18  0.0  0.458 6.998 45.8 6.0622 3.0 222.0 18.7    394.63 2.94  33.
↪4
4   0.06905 0.0  2.18  0.0  0.458 7.147 54.2 6.0622 3.0 222.0 18.7    396.90 5.33  36.
↪2
```

Show summary statistics for each variable: count, mean, standard deviation, minimum, 25th 50th and 75th percentiles, and maximum.

```
#Descriptions of each variable
df.describe()
```

```
stat  CRIM       ZN         INDUS      CHAS       NOX        RM         AGE       ␣
↪DIS        RAD        TAX        PTRATIO    B          LSTAT      target
===== ========== ========== ========== ========== ========== ========== ==========␣
↪========== ========== ========== ========== ========== ========== ==========
count 506.000000 506.000000 506.000000 506.000000 506.000000 506.000000 506.000000␣
↪506.000000 506.000000 506.000000 506.000000 506.000000 506.000000 506.000000
mean  3.593761   11.363636  11.136779  0.069170   0.554695   6.284634   68.574901  3.
↪795043   9.549407   408.237154 18.455534  356.674032 12.653063  22.532806
```

(continues on next page)

(continued from previous page)

```
std    8.596783   23.322453  6.860353   0.253994   0.115878   0.702617   28.148861  2.
↪105710   8.707259   168.537116 2.164946   91.294864  7.141062   9.197104
min    0.006320   0.000000   0.460000   0.000000   0.385000   3.561000   2.900000   1.
↪129600   1.000000   187.000000 12.600000  0.320000   1.730000   5.000000
25%    0.082045   0.000000   5.190000   0.000000   0.449000   5.885500   45.025000  2.
↪100175   4.000000   279.000000 17.400000  375.377500 6.950000   17.025000
50%    0.256510   0.000000   9.690000   0.000000   0.538000   6.208500   77.500000  3.
↪207450   5.000000   330.000000 19.050000  391.440000 11.360000  21.200000
75%    3.647423   12.500000  18.100000  0.000000   0.624000   6.623500   94.075000  5.
↪188425   24.000000  666.000000 20.200000  396.225000 16.955000  25.000000
max    88.976200  100.000000 27.740000  1.000000   0.871000   8.780000   100.000000 12.
↪126500  24.000000  711.000000 22.000000  396.900000 37.970000  50.000000
```

2.8.1 Correlation matrix

The correlation matrix lists the correlation of each variable with each other variable.

Positive correlations mean one variable tends to be high when the other is high, and negative correlations mean one variable tends to be high when the other is low.

Correlations close to zero are weak and cause a variable to have less influence in the model, and correlations close to one or negative one are strong and cause a variable to have more influence in the model.

```
#Here shows the basic correlation matrix
corr = df.corr()
corr
```

```
variable CRIM      ZN         INDUS     CHAS      NOX       RM        AGE       DIS
↪    RAD       TAX       PTRATIO   B         LSTAT     target
======== ========= ========= ========= ========= ========= ========= =========
↪========= ========= ========= ========= ========= ========= =========
CRIM     1.000000  -0.199458 0.404471  -0.055295 0.417521  -0.219940 0.350784  -0.
↪377904 0.622029  0.579564  0.288250  -0.377365 0.452220  -0.385832
ZN       -0.199458 1.000000  -0.533828 -0.042697 -0.516604 0.311991  -0.569537 0.
↪664408  -0.311948 -0.314563 -0.391679 0.175520  -0.412995 0.360445
INDUS    0.404471  -0.533828 1.000000  0.062938  0.763651  -0.391676 0.644779  -0.
↪708027 0.595129  0.720760  0.383248  -0.356977 0.603800  -0.483725
CHAS     -0.055295 -0.042697 0.062938  1.000000  0.091203  0.091251  0.086518  -0.
↪099176 -0.007368 -0.035587 -0.121515 0.048788  -0.053929 0.175260
NOX      0.417521  -0.516604 0.763651  0.091203  1.000000  -0.302188 0.731470  -0.
↪769230 0.611441  0.668023  0.188933  -0.380051 0.590879  -0.427321
RM       -0.219940 0.311991  -0.391676 0.091251  -0.302188 1.000000  -0.240265 0.
↪205246  -0.209847 -0.292048 -0.355501 0.128069  -0.613808 0.695360
AGE      0.350784  -0.569537 0.644779  0.086518  0.731470  -0.240265 1.000000  -0.
↪747881 0.456022  0.506456  0.261515  -0.273534 0.602339  -0.376955
DIS      -0.377904 0.664408  -0.708027 -0.099176 -0.769230 0.205246  -0.747881 1.
↪000000  -0.494588 -0.534432 -0.232471 0.291512  -0.496996 0.249929
RAD      0.622029  -0.311948 0.595129  -0.007368 0.611441  -0.209847 0.456022  -0.
↪494588 1.000000  0.910228  0.464741  -0.444413 0.488676  -0.381626
TAX      0.579564  -0.314563 0.720760  -0.035587 0.668023  -0.292048 0.506456  -0.
↪534432 0.910228  1.000000  0.460853  -0.441808 0.543993  -0.468536
PTRATIO  0.288250  -0.391679 0.383248  -0.121515 0.188933  -0.355501 0.261515  -0.
↪232471 0.464741  0.460853  1.000000  -0.177383 0.374044  -0.507787
B        -0.377365 0.175520  -0.356977 0.048788  -0.380051 0.128069  -0.273534 0.
↪291512  -0.444413 -0.441808 -0.177383 1.000000  -0.366087 0.333461
```

(continues on next page)

(continued from previous page)

```
LSTAT    0.452220  -0.412995 0.603800  -0.053929 0.590879  -0.613808 0.602339  -0.
↪496996 0.488676  0.543993  0.374044  -0.366087 1.000000  -0.737663
target   -0.385832 0.360445  -0.483725 0.175260  -0.427321 0.695360  -0.376955 0.
↪249929  -0.381626 -0.468536 -0.507787 0.333461  -0.737663 1.000000
```

Format with asterisks

Format the correlation matrix by rounding the numbers to two decimal places and adding asterisks to denote statistical significance:

```
def calculate_pvalues(df):
    df = df.select_dtypes(include='number')
    pairs = pd.MultiIndex.from_product([df.columns, df.columns])
    pvalues = [pearsonr(df[a], df[b])[1] for a, b in pairs]
    pvalues = pd.Series(pvalues, index=pairs).unstack().round(4)
    return pvalues

# code adapted from https://stackoverflow.com/questions/25571882/pandas-columns-
↪correlation-with-statistical-significance/49040342
def correlation_matrix(df,columns):
    rho = df[columns].corr()
    pval = calculate_pvalues(df[columns])
    # create three masks
    r0 = rho.applymap(lambda x: '{:.2f}'.format(x))
    r1 = rho.applymap(lambda x: '{:.2f}*'.format(x))
    r2 = rho.applymap(lambda x: '{:.2f}**'.format(x))
    r3 = rho.applymap(lambda x: '{:.2f}***'.format(x))
    # apply marks
    rho = rho.mask(pval>0.01,r0)
    rho = rho.mask(pval<=0.1,r1)
    rho = rho.mask(pval<=0.05,r2)
    rho = rho.mask(pval<=0.01,r3)
    return rho

columns = df.columns
correlation_matrix(df,columns)
```

```
variable CRIM     ZN       INDUS    CHAS     NOX      RM       AGE      DIS      RAD␣
↪    TAX      PTRATIO  B        LSTAT    target
======== ======== ======== ======== ======== ======== ======== ======== ========␣
↪======== ======== ======== ======== ======== ========
CRIM     1.00***  -0.20*** 0.40***  -0.06    0.42***  -0.22*** 0.35***  -0.38*** 0.
↪62***  0.58***  0.29***  -0.38*** 0.45***  -0.39***
ZN       -0.20*** 1.00***  -0.53*** -0.04    -0.52*** 0.31***  -0.57*** 0.66***  -0.
↪31*** -0.31*** -0.39*** 0.18***  -0.41*** 0.36***
INDUS    0.40***  -0.53*** 1.00***  0.06     0.76***  -0.39*** 0.64***  -0.71*** 0.
↪60***  0.72***  0.38***  -0.36*** 0.60***  -0.48***
CHAS     -0.06    -0.04    0.06     1.00***  0.09**   0.09**   0.09*    -0.10**  -0.
↪01    -0.04    -0.12*** 0.05     -0.05    0.18***
NOX      0.42***  -0.52*** 0.76***  0.09**   1.00***  -0.30*** 0.73***  -0.77*** 0.
↪61***  0.67***  0.19***  -0.38*** 0.59***  -0.43***
RM       -0.22*** 0.31***  -0.39*** 0.09**   -0.30*** 1.00***  -0.24*** 0.21***  -0.
↪21*** -0.29*** -0.36*** 0.13***  -0.61*** 0.70***
AGE      0.35***  -0.57*** 0.64***  0.09*    0.73***  -0.24*** 1.00***  -0.75*** 0.
↪46***  0.51***  0.26***  -0.27*** 0.60***  -0.38***
```

(continues on next page)

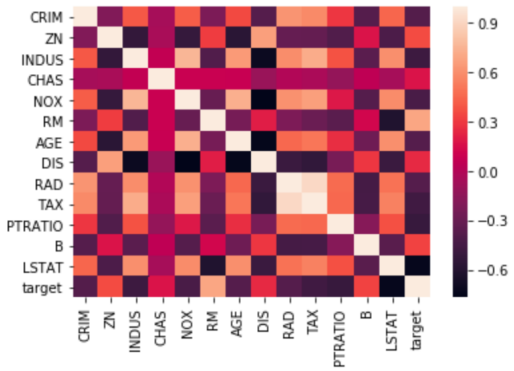
(continued from previous page)

```
DIS      -0.38*** 0.66***  -0.71*** -0.10**  -0.77*** 0.21***  -0.75*** 1.00***  -0.
↪49*** -0.53*** -0.23*** 0.29***  -0.50*** 0.25***
RAD      0.62***  -0.31*** 0.60***  -0.01    0.61***  -0.21*** 0.46***  -0.49*** 1.
↪00***  0.91***  0.46***  -0.44*** 0.49***  -0.38***
TAX      0.58***  -0.31*** 0.72***  -0.04    0.67***  -0.29*** 0.51***  -0.53*** 0.
↪91***  1.00***  0.46***  -0.44*** 0.54***  -0.47***
PTRATIO  0.29***  -0.39*** 0.38***  -0.12*** 0.19***  -0.36*** 0.26***  -0.23*** 0.
↪46***  0.46***  1.00***  -0.18*** 0.37***  -0.51***
B        -0.38*** 0.18***  -0.36*** 0.05     -0.38*** 0.13***  -0.27*** 0.29***  -0.
↪44*** -0.44*** -0.18*** 1.00***  -0.37*** 0.33***
LSTAT    0.45***  -0.41*** 0.60***  -0.05    0.59***  -0.61*** 0.60***  -0.50*** 0.
↪49***  0.54***  0.37***  -0.37*** 1.00***  -0.74***
target   -0.39*** 0.36***  -0.48*** 0.18***  -0.43*** 0.70***  -0.38*** 0.25***  -0.
↪38*** -0.47*** -0.51*** 0.33***  -0.74*** 1.00***
```

Heatmap

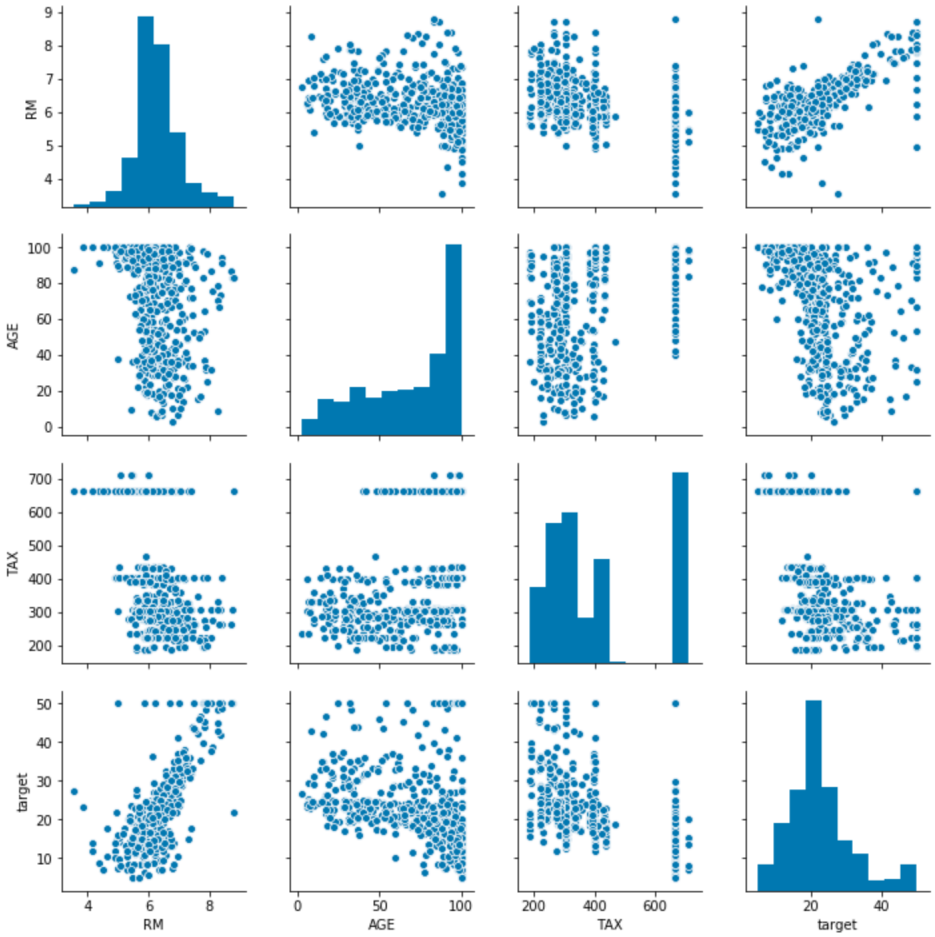
Heatmap of the correlation matrix:

```
sns.heatmap(corr,
        xticklabels=corr.columns,
        yticklabels=corr.columns)
```



2.8.2 Pairwise distributions with seaborn

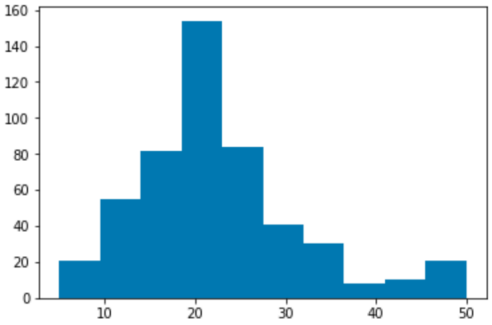
```
sns.pairplot(df[['RM', 'AGE', 'TAX', 'target']])
```


RM
AGE
TAX
target
RM
AGE
TAX
target

2.8.3 Target variable distribution

Histogram showing the distribution of the target variable. In this dataset this is "Median value of owner-occupied homes in $1000's", abbreviated MEDV.

```
plt.hist(df['target'])
plt.show()
```



2.8.4 Simple linear regression

The variable MEDV is the target that the model predicts. All other variables are used as predictors, also called features.

The target variable is continuous, so use a linear regression instead of a logistic regression.

```
# Define features as X, target as y.
X = df.drop('target', axis='columns')
y = df['target']
```

Split the dataset into a training set and a test set:

```
# Splitting the dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_
↪state = 0)
```

A linear regression consists of a coefficient for each feature and one intercept.

To make a prediction, each feature is multiplied by its coefficient. The intercept and all of these products are added together. This sum is the predicted value of the target variable.

The residual sum of squares (RSS) is calculated to measure the difference between the prediction and the actual value of the target variable.

The function `fit` calculates the coefficients and intercept that minimize the RSS when the regression is used on each record in the training set.

```
# Fitting Simple Linear Regression to the Training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

(continues on next page)

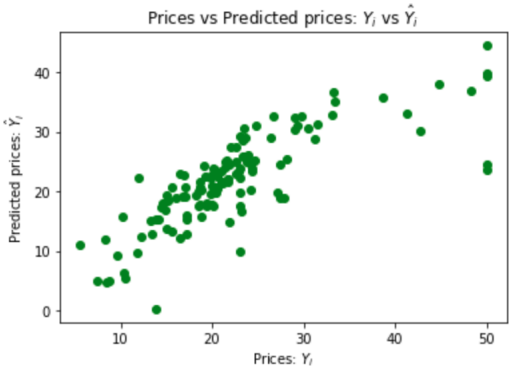
(continued from previous page)

```
# The intercept
print('Intercept: \n', regressor.intercept_)

# The coefficients
print('Coefficients: \n', pd.Series(regressor.coef_, index=X.columns, name=
↪'coefficients'))
```

```
Intercept:
 36.98045533762056
Coefficients:
 CRIM       -0.116870
ZN           0.043994
INDUS       -0.005348
CHAS         2.394554
NOX        -15.629837
RM           3.761455
AGE         -0.006950
DIS         -1.435205
RAD          0.239756
TAX         -0.011294
PTRATIO     -0.986626
B            0.008557
LSTAT       -0.500029
Name: coefficients, dtype: float64
```

Now check the accuracy when this linear regression is used on new data that it was not trained on. That new data is the test set.

```
# Predicting the Test set results
y_pred = regressor.predict(X_test)

# Visualising the Test set results
# code adapted from https://joomik.github.io/Housing/
fig, ax = plt.subplots()
ax.scatter(y_test, y_pred, color='green')
ax.set(
    xlabel="Prices: $Y_i$",
    ylabel="Predicted prices: $\hat{Y}_i$",
    title="Prices vs Predicted prices: $Y_i$ vs $\hat{Y}_i$",
)
plt.show()
```



This scatter plot shows that the regression is a good predictor of the data in the test set.

The mean squared error quantifies this performance:

```
# The mean squared error as a way to measure model performance.
print("Mean squared error: %.2f" % mean_squared_error(y_test, y_pred))
```

```
Mean squared error: 29.79
```

2.8.5 Ordinary least squares (OLS) regression with Statsmodels

```
model = sm.ols('target ~ AGE + B + CHAS + CRIM + DIS + INDUS + LSTAT + NOX + PTRATIO␣
↪+ RAD + RM + TAX + ZN', df)
result = model.fit()
result.summary()
```

```
OLS Regression Results
==============================================================================
Dep. Variable:                 target   R-squared:                       0.741
Model:                            OLS   Adj. R-squared:                  0.734
Method:                 Least Squares   F-statistic:                     108.1
Date:                Thu, 23 Aug 2018   Prob (F-statistic):          6.95e-135
Time:                        07:29:16   Log-Likelihood:                -1498.8
No. Observations:                 506   AIC:                             3026.
Df Residuals:                     492   BIC:                             3085.
Df Model:                          13
Covariance Type:            nonrobust
==============================================================================
coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     36.4911      5.104      7.149      0.000      26.462      46.520
AGE            0.0008      0.013      0.057      0.955      -0.025       0.027
B              0.0094      0.003      3.500      0.001       0.004       0.015
CHAS           2.6886      0.862      3.120      0.002       0.996       4.381
CRIM          -0.1072      0.033     -3.276      0.001      -0.171      -0.043
DIS           -1.4758      0.199     -7.398      0.000      -1.868      -1.084
INDUS          0.0209      0.061      0.339      0.735      -0.100       0.142
LSTAT         -0.5255      0.051    -10.366      0.000      -0.625      -0.426
NOX          -17.7958      3.821     -4.658      0.000     -25.302     -10.289
PTRATIO       -0.9535      0.131     -7.287      0.000      -1.211      -0.696
RAD            0.3057      0.066      4.608      0.000       0.175       0.436
RM             3.8048      0.418      9.102      0.000       2.983       4.626
TAX           -0.0123      0.004     -3.278      0.001      -0.020      -0.005
ZN             0.0464      0.014      3.380      0.001       0.019       0.073
==============================================================================
Omnibus:                      178.029   Durbin-Watson:                   1.078
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              782.015
Skew:                           1.521   Prob(JB):                    1.54e-170
Kurtosis:                       8.276   Cond. No.                     1.51e+04
==============================================================================

Warnings:
```

(continues on next page)

(continued from previous page)

```
[1] Standard Errors assume that the covariance matrix of the errors is correctly␣
↪specified.
[2] The condition number is large, 1.51e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

2.8.6 Principal component analysis

The initial dataset has a number of feature or predictor variables and one target variable to predict.

Principal component analysis (PCA) converts these features into a set of principal components, which are linearly uncorrelated variables.

The first principal component has the largest possible variance and therefore accounts for as much of the variability in the data as possible.

Each of the other principal components is orthogonal to all of its preceding components, but has the largest possible variance within that constraint.

Graphing a dataset by showing only the first two or three of the principal components effectively projects a complex dataset with high dimensionality into a simpler image that shows as much of the variance in the data as possible.

PCA is sensitive to the relative scaling of the original variables, so begin by scaling them:

```
# Feature Scaling
x = StandardScaler().fit_transform(X)
```

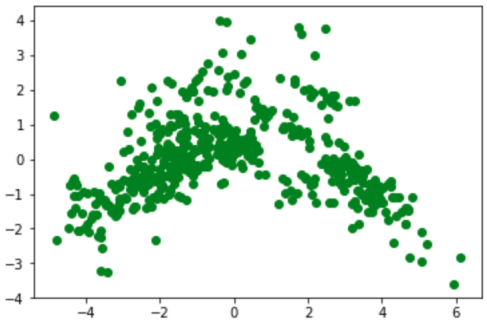
Calculate the first three principal components and show them for the first five rows of the housing dataset:

```
# Project data to 3 dimensions
pca = PCA(n_components=3)
principalComponents = pca.fit_transform(x)
principalDf = pd.DataFrame(
    data = principalComponents,
    columns = ['principal component 1', 'principal component 2', 'principal component␣
↪3'])
principalDf.head()
```

row	principal component 1	principal component 2	principal component 3
0	-2.097842	0.777102	0.335076
1	-1.456412	0.588088	-0.701340
2	-2.074152	0.602185	0.161234
3	-2.611332	-0.005981	-0.101940
4	-2.457972	0.098860	-0.077893

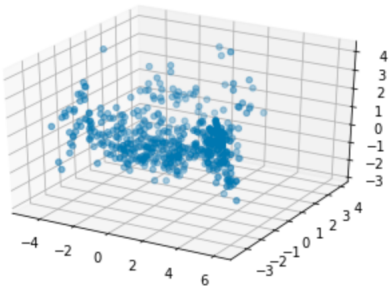
Show a 2D graph of this data:

```
plt.scatter(principalDf['principal component 1'], principalDf['principal component 2
↪'], color ='green')
plt.show()
```



Show a 3D graph of this data:

```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(principalDf['principal component 1'], principalDf['principal component 2'],
↪ principalDf['principal component 3'])
plt.show()
```



Measure how much of the variance is explained by each of the three components:

```
# Variance explained by each component
explained_variance = pca.explained_variance_ratio_
explained_variance
```

```
array([0.47097344, 0.11015872, 0.09547408])
```

Each value will be less than or equal to the previous value, and each value will be in the range from 0 through 1.

The sum of these three values shows the fraction of the total variance explained by the three principal components, in the range from 0 (none) through 1 (all):

```
sum(explained_variance)
```

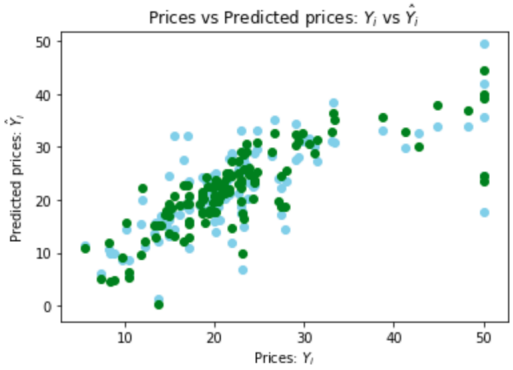
```
0.6766062376563704
```

Predict the target variable using only the three principal components:

```
y_test_linear = y_test
y_pred_linear = y_pred
X = principalDf
y = df['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_
↪state = 0)
regressor = LinearRegression()
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_test)
```

Plot the predictions from the linear regression in green again, and the new predictions in blue:

```
fig, ax = plt.subplots()
ax.scatter(y_test, y_pred, color='skyblue')
ax.scatter(y_test_linear, y_pred_linear, color = 'green')
ax.set(
    xlabel="Prices: $Y_i$",
    ylabel="Predicted prices: $\hat{Y}_i$",
    title="Prices vs Predicted prices: $Y_i$ vs $\hat{Y}_i$",
)
plt.show()
```



The blue points are somewhat more widely scattered, but similar.

Calculate the mean squared error:

```
print("Linear regression mean squared error: %.2f" % mean_squared_error(y_test_linear,
↪ y_pred_linear))
print("PCA mean squared error: %.2f" % mean_squared_error(y_test, y_pred))
```

```
Linear regression mean squared error: 29.79
PCA mean squared error: 43.49
```

2.9 Data exploration

Anaconda Enterprise supports data exploration using visualization libraries such as Bokeh and Matplotlib and numeric libraries such as NumPy, SciPy, and Pandas. With these tools data scientists can discover and learn about patterns and relationships in their datasets to develop approaches for their analysis and deployment pipelines.

These examples use the Iris flower data set and the following tiny example customer data set "customers.csv":

```
customer_id,title,industry
1,data scientist,retail
2,data scientist,academia
3,compiler optimizer,academia
4,data scientist,finance
5,compiler optimizer,academia
6,data scientist,academia
7,compiler optimizer,academia
8,data scientist,retail
9,compiler optimizer,finance
```

Begin by importing libraries and reading data into a pandas DataFrame:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
irisdf = pd.read_csv('iris.csv')
customerdf = pd.read_csv('customers.csv')

%matplotlib inline
```

2.9.1 Column names

To list column names:

```
print(irisdf.columns)
```

```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class'], dtype=
↪'object')
```

2.9.2 Summary statistics

Summary statistics include minimum, maximum, mean, median, percentiles, and more.

```
print('length:', len(irisdf)) # length of data set
print('shape:', irisdf.shape) # length and width of data set
print('size:', irisdf.size) # length * width
print('min:', irisdf['sepal_width'].min())
print('max:', irisdf['sepal_width'].max())
print('mean:', irisdf['sepal_width'].mean())
print('median:', irisdf['sepal_width'].median())
print('50th percentile:', irisdf['sepal_width'].quantile(0.5)) # 50th percentile,␣
↪also known as median
print('5th percentile:', irisdf['sepal_width'].quantile(0.05))
print('10th percentile:', irisdf['sepal_width'].quantile(0.1))
print('95th percentile:', irisdf['sepal_width'].quantile(0.95))
```

```
length: 150
shape: (150, 5)
size: 750
min: 2.0
max: 4.4
mean: 3.0573333333333337
median: 3.0
50th percentile: 3.0
5th percentile: 2.3449999999999998
10th percentile: 2.5
95th percentile: 3.8
```

2.9.3 Value counts

The `value_counts` function shows the number of items in each category, sorted from largest to smallest. It can also take an argument `ascending`, and if set to `True` this will show the list from smallest to largest.

```
print(customerdf['industry'].value_counts())
print()
print(customerdf['industry'].value_counts(ascending=True))
```

```
academia    5
finance     2
retail      2
Name: industry, dtype: int64

retail      2
finance     2
academia    5
Name: industry, dtype: int64
```

2.9.4 Categoricals

In statistics, a categorical variable may take on a limited number of possible values. Examples could include blood type, nation of origin, or ratings on a Likert scale. Like numbers, the possible values may have an order, such as from "disagree" to "neutral" to "agree." However, the values cannot be used for numerical operations such as addition or division.

A column such as the "class" column of the iris dataset can be converted from dtype "object" to dtype "categorical."

```
print(irisdf.dtypes)
print()
irisdf['class'] = irisdf['class'].astype('category')
print(irisdf.dtypes)
```

```
sepal_length    float64
sepal_width     float64
petal_length    float64
petal_width     float64
class            object
dtype: object

sepal_length     float64
```

(continues on next page)

(continued from previous page)

```
sepal_width       float64
petal_length      float64
petal_width       float64
class            category
dtype: object
```

Within pandas, this creates an array of the possible values, where each value appears only once, and replaces the strings in the DataFrame with indexes into the array. In some cases this saves significant memory.

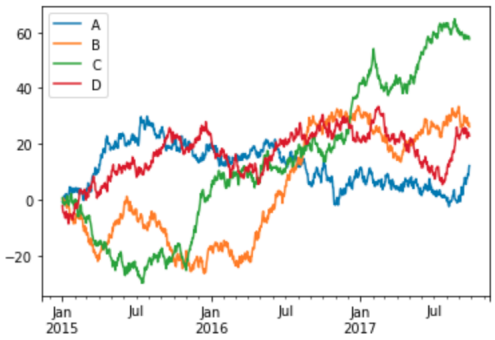
A categorical may have a logical order different than the lexical order. For example, for ratings on a Likert scale, the lexical order could alphabetize the strings and produce "agree, disagree, neither agree nor disagree, strongly agree, strongly disagree". The logical order could be in order from most negative to most positive as "strongly disagree, disagree, neither agree nor disagree, agree, strongly agree."

Finally, using the categorical data type for categorical data signals to other Python libraries to treat the column as categorical data, which can help those libraries default to suitable statistical methods or plot types.

2.9.5 Time series data visualization

This code creates four series of random numbers over time, calculates the cumulative sums for each series over time, and plots them.

```
timedf = pd.DataFrame(np.random.randn(1000, 4), index=pd.date_range('1/1/2015',␣
↪periods=1000), columns=list('ABCD'))
timedf = timedf.cumsum()
timedf.plot()
```

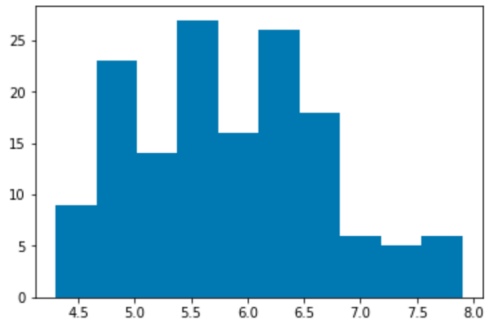


This example was adapted from http://pandas.pydata.org/pandas-docs/stable/visualization.html .

2.9.6 Histograms

This code plots a histogram of the sepal length values in the Iris data set.

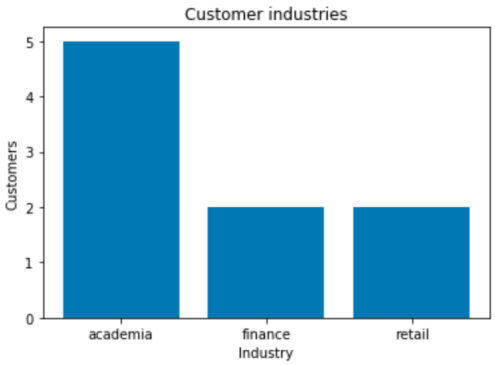
```
plt.hist(irisdf['sepal_length'])
plt.show()
```



2.9.7 Bar charts

This code produces a bar chart of the industries of customers in the customer data set.

```
industries = customerdf['industry'].value_counts()

fig, ax = plt.subplots()

ax.bar(np.arange(len(industries)), industries)

ax.set_xlabel('Industry')
ax.set_ylabel('Customers')
ax.set_title('Customer industries')
ax.set_xticks(np.arange(len(industries)))
ax.set_xticklabels(industries.index)

plt.show()
```

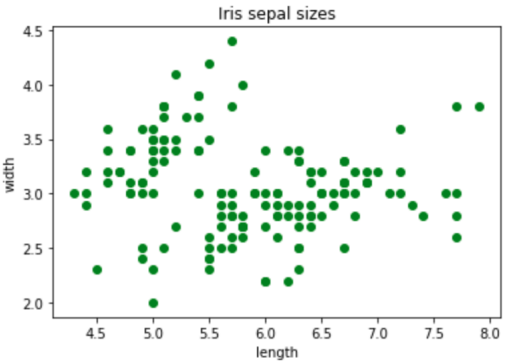


This example was adapted from https://matplotlib.org/gallery/statistics/barchart_demo.html .

2.9.8 Scatter plots

This code makes a scatter plot of the sepal lengths and widths in the Iris data set.

```
fig, ax = plt.subplots()
ax.scatter(irisdf['sepal_length'], irisdf['sepal_width'], color='green')
ax.set(
    xlabel="length",
    ylabel="width",
    title="Iris sepal sizes",
)
plt.show()
```



2.9.9 Sorting

Show the customer data set:

```
customerdf
```

row	customer_id	title	industry
0	1	data scientist	retail
1	2	data scientist	academia
2	3	compiler optimizer	academia
3	4	data scientist	finance
4	5	compiler optimizer	academia
5	6	data scientist	academia
6	7	compiler optimizer	academia
7	8	data scientist	retail
8	9	compiler optimizer	finance

Sort by industry and show the results:

```
customerdf.sort_values(by=['industry'])
```

row	customer_id	title	industry
1	2	data scientist	academia
2	3	compiler optimizer	academia
4	5	compiler optimizer	academia
5	6	data scientist	academia
6	7	compiler optimizer	academia
3	4	data scientist	finance
8	9	compiler optimizer	finance
0	1	data scientist	retail
7	8	data scientist	retail

Sort by industry and then title:

```
customerdf.sort_values(by=['industry', 'title'])
```

row	customer_id	title	industry
2	3	compiler optimizer	academia
4	5	compiler optimizer	academia
6	7	compiler optimizer	academia
1	2	data scientist	academia
5	6	data scientist	academia
8	9	compiler optimizer	finance
3	4	data scientist	finance
0	1	data scientist	retail
7	8	data scientist	retail

The `sort_values` function can also use the arguments `axis` to sort either rows or columns, `ascending` to sort in either ascending or descending order, `inplace` to perform the sorting operation in-place and without copying the data, which can save space, `kind` to use the quicksort, merge sort, or heapsort algorithms, and `na_position` to sort "not a number" (NaN) entries at the end or beginning.

2.9.10 Grouping

```
print(customerdf.groupby('title')['customer_id'].count())
print()
print(customerdf.groupby('title').size())
print()
print(customerdf.groupby(['title', 'industry']).size())
print()
print(customerdf.groupby(['industry', 'title']).size())
```

```
title
compiler optimizer    4
data scientist        5
Name: customer_id, dtype: int64

title
compiler optimizer    4
data scientist        5
dtype: int64

```

(continues on next page)

(continued from previous page)

```
title               industry
compiler optimizer  academia    3
                    finance     1
data scientist      academia    2
                    finance     1
                    retail      2
dtype: int64

industry  title
academia  compiler optimizer    3
          data scientist        2
finance   compiler optimizer    1
          data scientist        1
retail    data scientist        2
dtype: int64
```

`customerdf.groupby('title')['customer_id'].count()` counts the items in each group, excluding missing values such as not-a-number values (`NaN`). Because there are no missing customer IDs, this is equivalent to `customerdf.groupby('title').size()`.

By default `groupby` sorts the group keys. You can use the `sort=False` option to prevent this, which can make the grouping operation faster.

2.9.11 Binning

Binning or bucketing moves continuous data into discrete chunks which can be used as ordinal categorical variables.

You can divide the range of the sepal length measurements into four equal bins:

```
pd.cut(irisdf['sepal_length'], 4).head()
```

```
0    (4.296, 5.2]
1    (4.296, 5.2]
2    (4.296, 5.2]
3    (4.296, 5.2]
4    (4.296, 5.2]
Name: sepal_length, dtype: category
Categories (4, interval[float64]): [(4.296, 5.2] < (5.2, 6.1] < (6.1, 7.0] < (7.0, 7.
↪9]]
```

Instead, make a custom bin array to divide the sepal length measurements into integer sized bins from 4 through 8:

```
custom_bin_array = np.linspace(4, 8, 5)
custom_bin_array
```

```
array([4., 5., 6., 7., 8.])
```

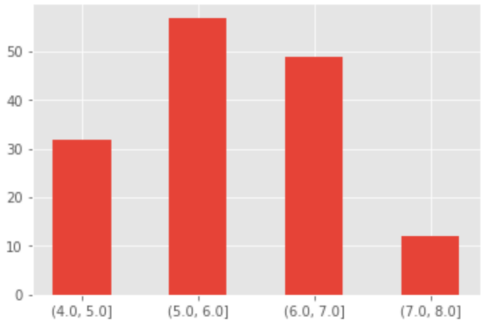
Copy the iris data set and apply the binning to it:

```
iris2=irisdf.copy()
iris2['sepal_length'] = pd.cut(iris2['sepal_length'], custom_bin_array)
iris2['sepal_length'].head()
```

```
0     (5.0, 6.0]
1     (4.0, 5.0]
2     (4.0, 5.0]
3     (4.0, 5.0]
4     (4.0, 5.0]
Name: sepal_length, dtype: category
Categories (4, interval[float64]): [(4.0, 5.0] < (5.0, 6.0] < (6.0, 7.0] < (7.0, 8.0]]
```

Plot the binned data:

```
plt.style.use('ggplot')
categories = iris2['sepal_length'].cat.categories
ind = np.array([x for x, _ in enumerate(categories)])
plt.bar(ind, iris2.groupby('sepal_length').size(), width, label='Sepal length')
plt.xticks(ind, categories)
plt.show()
```



This example was adapted from http://benalexkeen.com/bucketing-continuous-variables-in-pandas/ .

CHAPTER 3

Administrator workflows

3.1 Managing cluster resources

To help you manage your organization's cluster resources more efficiently, Anaconda Enterprise enables you to *track which sessions and deployments are running on specific nodes or by specific users*.

You can also *monitor cluster resource usage* in terms of CPU, memory, disk space, network and GPU utilization.

To help you gain insights into user services and troubleshoot issues, Anaconda Enterprise provides *detailed logs* and debugging information related to the Kubernetes services it uses, as well as all *activity performed by users*.

You can *add new nodes* that you've installed—including GPUs—to Anaconda Enterprise as your organization scales, and *configure resource profiles* to make these nodes available to users.

See *fault tolerance in Anaconda Enterprise* for information about what to do if a master node fails.

3.1.1 Monitoring sessions and deployments

Anaconda Enterprise enables you to see which sessions and deployments are running on specific nodes or by specific users, so you can monitor cluster resource usage. You can also view session details *for a specific user* in the Authorization Center. See *Managing users* for more information.

1. Log in to Anaconda Enterprise, select the **Menu** icon  in the top right corner and click the **Administrative Console** link displayed at the bottom of the slide out window.
2. Click **Manage Resources**.
3. Log in to the Operations Center using the Administrator credentials *configured after installation*.
4. Select **Kubernetes** from the menu on the left to display a list of all nodes.
5. Click the **Pods** tab to display a list of all pods and containers.
6. Click a pod name—sessions are named `anaconda-session-XXXXX` and deployments are named `anaconda-enterprise-app-XXXX`—and select **Monitoring** or **Logs** to view a graph of the resource usage or logs for the corresponding session or deployment.

Using the CLI:

1. Open an SSH session on the master node in a terminal by logging into the Operations Center and selecting **Servers** from the menu on the left.
2. Click on the IP address for the Anaconda Enterprise master node and select SSH login as **root**.
3. In the terminal window, run `sudo gravity enter`.
4. Run `kubectl get pods | grep anaconda-session-` to view a list of all running session pods.
5. Run `kubectl get pods | grep anaconda-app-` to view all running deployment pods.

You can also *monitor cluster utilization* in terms of CPU, memory, disk space, network and GPU utilization.

3.1.2 Monitoring cluster utilization

Anaconda Enterprise enables you to monitor cluster resource usage in terms of CPU, memory, disk space, network and GPU utilization.

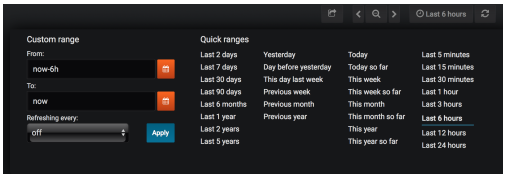
To access the Operations Center:

1. Log in to Anaconda Enterprise, select the **Menu** icon ☰ in the top right corner and click the **Administrative Console** link displayed at the bottom of the slide out window.
2. Click **Manage Resources**.
3. Login to the Operations Center using the Administrator credentials *configured after installation*
4. Click **Monitoring** in the menu on the left.

The graphs displayed include the following:

- Overall Cluster CPU Usage
- CPU Usage by Node
- Individual CPU Usage
- Overall Cluster Memory Usage
- Memory Usage by Node
- Individual Node Memory Usage
- Overall Cluster Network Usage
- Network Usage by Node
- Individual Node Network Usage
- Overall Cluster Filesystem Usage
- Filesystem Usage by Node
- Individual Filesystem Usage

Use the control in the upper right corner to specify the range of time for which you want to view usage information, and how often you want to refresh the results.



Sharing graphs

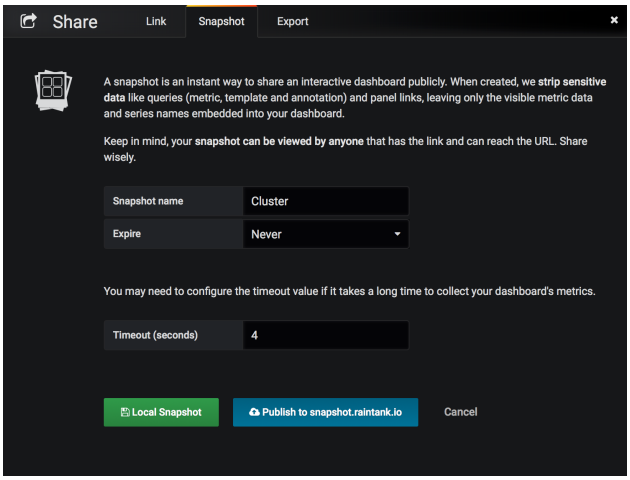
To share a graph, click the **Share dashboard** icon  to get a unique URL that can be shared. The link contains an access token so the viewer does not need a password.

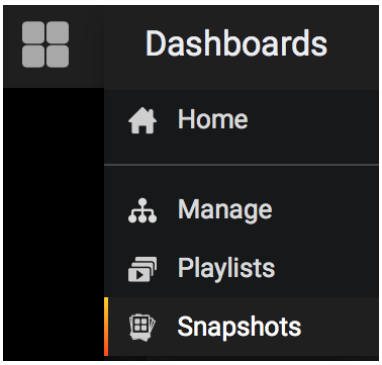
Click **Snapshot** in the top menu of the **Share** window to create a snapshot that strips out sensitive data, leaving only the visible metric data and series names embedded into your dashboard.

Select **Local Snapshot** to save the image locally, or **Publish to snapshot** to save the image to the cloud.

To view and manage your snapshots, select **Snapshots** from the **Dashboards** menu on the left of the Monitoring window. Cloud snapshots can be manipulated by the viewer with zoom and time range changes, but the viewer cannot change the node, cluster or pod.

NOTE: Anyone who has the link and can reach the URL can view the report.





Monitoring Kubernetes

To view the status of your Kubernetes nodes, pods, services, jobs, daemon sets and deployments from the Operations Center, click **Kubernetes** in the menu on the left and select a tab name. See *Monitoring sessions and deployments* for more information.

To view the status or progress of a cluster installation, click **Operations** in the menu on the left, and select an operation in the list. Clicking on a specific operation switches to the **Logs** view, where you can also view logs based on container or pod.

3.1.3 Viewing system logs

To help you gain insights into user services and troubleshoot issues, Anaconda Enterprise provides detailed logs and debugging information related to the Kubernetes services and containers it uses.

To access these logs from the Operations Center:

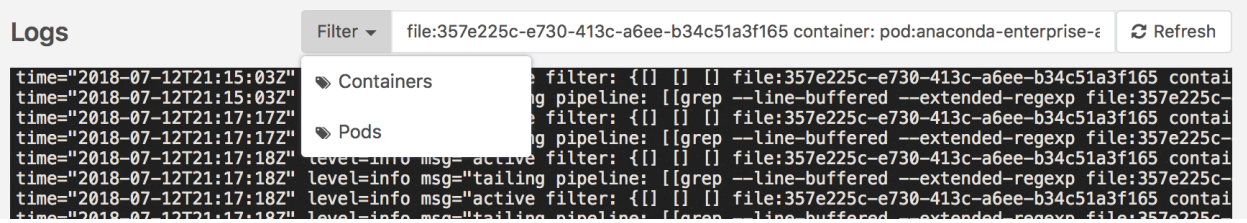
1. Log in to Anaconda Enterprise, select the **Menu** icon ☰ in the top right corner and click the **Administrative Console** link displayed at the bottom of the slide out window.

1. Click **Manage Resources**.

2. Log in to the Operations Center using the Administrator credentials *configured after installation*.

3. Click **Logs** in the left menu to display the complete system log.

4. Use the **Filter** drop-down to view logs based on container or pod.



You can also access the logs for a specific pod by clicking **Kubernetes** in the left menu, clicking the **Pods** tab, clicking the name of a pod, and selecting **Logs**.

To use the CLI:

1. Open an SSH session on the master node in a terminal by logging into the Operations Center and selecting **Servers** from the menu on the left.
2. Click on the IP address for the Anaconda Enterprise master node and select SSH login as **root**.
3. In the terminal window, run `sudo gravity enter`.
4. Run `kubectl get pods` to view a list of all running session pods.
5. Run `kubectl logs <POD-NAME>` to display the logs for the pod specified.

3.1.4 Viewing activity logs

Anaconda Enterprise logs all activity performed by users, including the following:

- Each system login.
- All Admin actions.
- Each time a project is created and updated.
- Each time a project is deployed.

In each case, the user who performed the action and when it occurred are tracked, along with any other important details.

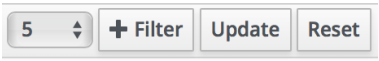
As an Administrator, you can log in to the Administrative Console's Authentication Center to view the log of all login and Admin events:

1. Log in to Anaconda Enterprise, select the **Menu** icon ☰ in the top right corner and click the **Administrative Console** link displayed at the bottom of the slide out window.
2. Click **Manage Users**.
3. Log in to the Authentication Center using the Administrator credentials *configured after installation*.
4. Click **Events** in the left menu to display a log of all **Login Events**.
5. Click the **Admin Events** tab to view a sumary of all actions performed by Admin users.

To filter events:

Event data can become difficult to manage as it accumulates, so Anaconda Enterprise provides a few options to make it more manageable:

1. Click the **Config** tab to configure the type of events you want Anaconda Enterprise to log, clear events, and schedule if you want to periodically delete event logs from the database.
2. Use the **Filter** options available on both the **Login Events** and **Admin Events** windows to control the results displayed based on variables such as event or operation, user or resource, and a range of dates.



- Click **Update** to refresh the results based on the filter you configured, and **Reset** to return to the original log results.
- Select the maximum number of results you want displayed: 5, 10, 50 or 100.

To view activity at the project level:

1. Switch to the User Console and click **Projects** in the top menu.
2. Select the project you want to view information about to display a list of all actions performed on the project in the **Activity** window.

3.1.5 Adding and removing nodes

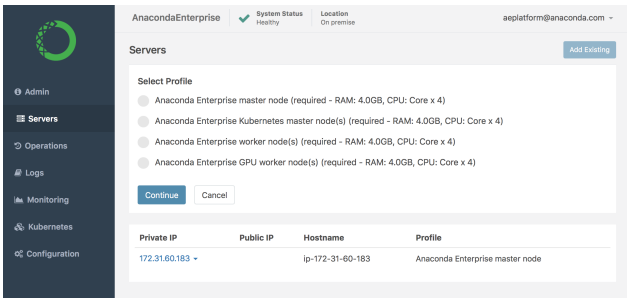
You can view, add, edit and delete servers from Anaconda Enterprise from the Admin Console's Operations Center.

To manage the servers on your system:

1. Log in to Anaconda Enterprise, select the **Menu** icon ☰ in the top right corner and click the **Administrative Console** link displayed at the bottom of the slide out window.
2. Click **Manage Resources**.
3. Log in to the Operations Center using the Administrator credentials *configured after installation*.
4. Select **Servers** from the menu on the left to display the private or public IP address, hostname and profile of each node on your system.

To add an existing server to Anaconda Enterprise:

1. Click the **Add Existing** button at the top right.



2. Select an appropriate profile for the server and click **Continue**.

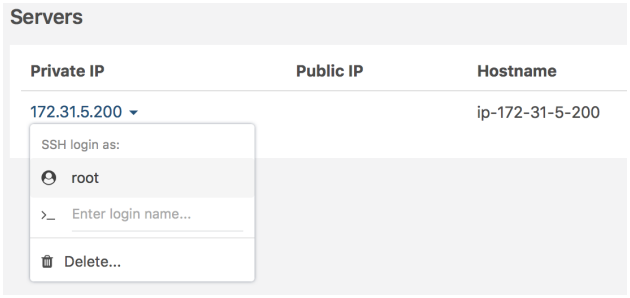
   NOTE: If you are adding a GPU node, make sure it meets the *GPU requirements*.

3. Copy and paste the command provided into a terminal window to add the server.

   When you refresh the page, your server will appear in the list.

To log on to a server:

Click on the IP address of the server you want to work with, and select SSH login as **root** to open a terminal window.



When you are finished, simply close the console window by clicking the  icon.

3.1.6 Configuring resource profiles

If Anaconda Enterprise users need to run applications which require more memory or compute power than provided by default, you can customize your installation to include these resources and allow users to access them while working within AE.

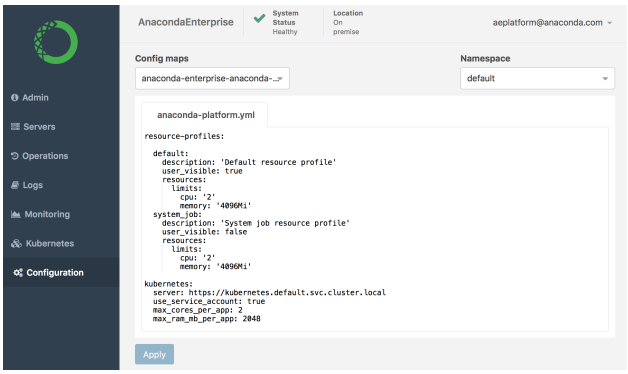
After the server resources are installed as nodes in the cluster, you create *resource profiles* to make these resources available for users to select from when *configuring a project's settings* or *deploying a project*. Anaconda Enterprise finds the node that matches their request.

For example, if your installation includes nodes with GPUs, add a GPU resource profile so users can use the GPUs to accelerate computation within their projects—essential for machine learning model training. For installation requirements, see *Installation requirements*.

To add a resource profile for a resource you have installed:

1. Log in to Anaconda Enterprise, select the **Menu** icon in the top right corner and click the **Administrative Console** link displayed at the bottom of the slide out window.
2. Click **Manage Resources**.
3. Log in to the Operations Center using the Administrator credentials *configured after installation*.
4. Select **Configuration** from the menu on the left.
5. Use the **Config map** drop-down menu to select the `anaconda-platform.yml` configuration file.

   NOTE: We recommend that you make a manual backup copy since you will be editing this file directly.
6. Scroll down to the `resource-profiles` section:



7. Add an additional resource following the format of the default specification. For example, to create a GPU resource profile, add the following to the `resource-profiles` section of the Config map:

```
gpu-profile:
  description: 'GPU resource profile'
  user_visible: true
```

(continues on next page)

(continued from previous page)

```
  resources:
    limits:
      nvidia.com/gpu: 1
      memory: '8Gi'
```

NOTE: Resource profiles are listed in alphabetical order—after any defaults—so if you want them to appear in a particular order in the drop-down list that users see, be sure to name them accordingly.

8. Click **Apply** to save your changes.

To update the Anaconda Enterprise server with your changes, you'll need to do the following:

1. Run the following command in an interactive shell to identify the pods associated with the workspace and deployment services:

```
kubectl get pods
```

2. Restart the workspace and deploy services by running the following command:

```
kubectl delete pod anaconda-enterprise-ap-deploy-<unique ID> anaconda-enterprise-
↪ap-workspace-<unique ID>
```

Check the project **Settings** and **Deploy** UI to verify that each resource profile you added or edited appears in the **Resource Profile** drop-down menu.

3.1.7 Fault tolerance in Anaconda Enterprise

Anaconda Enterprise employs automatic service restarts and health monitoring to remain operational if a process halts or a worker node becomes unavailable. Additional levels of fault tolerance, such as service migration, are provided if there are at least three nodes in the deployment. However, the master node cannot currently be configured for automatic failover and does present a single point of failure.

When Anaconda Enterprise is deployed to a cluster with three or more nodes, the core services are automatically configured into a fault tolerant mode—whether Anaconda Enterprise is initially configured this way or changed later. As soon as there are three or more nodes available, the service fault tolerance features come into effect.

Core service

In the event of any service failure, Anaconda Enterprise core services will automatically be restarted or, if possible, migrated.

User deployments

In the event of any failure, user-initiated project deployments will automatically be restarted or, if possible, migrated.

Worker nodes

If any worker node becomes unresponsive or unavailable, it will be flagged while the core Enterprise services and backend continue to run without interruption. If additional worker nodes are available the services that had been running on the failed worker node will be migrated or restarted on other still-live worker nodes. This migration may take a few minutes.

New worker nodes can be added to the Enterprise cluster as described in *Adding and removing nodes*.

Storage and persistency layer

Anaconda Enterprise does not automatically configure storage or persistency layer fault tolerance when using the default storage and persistency services. This includes the database, Git server, and object storage. If you have

configured Anaconda Enterprise to use external storage and persistency services then you will need to configure these for fault tolerance.

Recovering after node failure

Other than storage-related services (database, Git server, and object storage), all core Anaconda Enterprise services are resilient to master node failure.

To maintain operation of Enterprise in the event of a master node failure, `/opt/anaconda/` on the master node should be located on a redundant disk array or backed up frequently to avoid data loss. See *Backing up and restoring Anaconda Enterprise* for more information.

To restore Anaconda Enterprise operations in the event of a master node failure, complete the following steps:

1. Create a new master node. Follow the installation process for adding a new cluster node, described in *Unattended command-line installations*.

   NOTE: To create the new master node, select `--role=master` instead of `--role=worker`.

2. Restore data from a backup. After the installation of the new master node is complete, follow the instructions in *Backing up and restoring Anaconda Enterprise*.

3.2 Configuring authentication

As an Administrator, you'll need to authorize users so they can use Anaconda Enterprise. This involves *adding users to the system*, setting their credentials, *mapping them to roles*, and optionally *assigning them to one or more groups*.

To help expedite the process of authorizing large groups of users, you can *connect to an external identity provider* such as LDAP or Active Directory and federate those users.

You'll use the Administrative Console's Authentication Center to perform these actions.

3.2.1 Connecting to external identity providers

Anaconda Enterprise comes with out-of-the-box support for LDAP, Active Directory, SAML and Kerberos. As each enterprise configuration is different, coordinate with your LDAP/AD Administrator to obtain the provider-specific information you need to proceed.

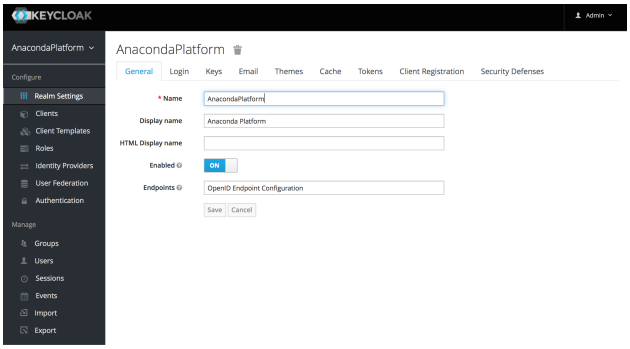
NOTE: You must have pagination turned off before starting.

Adding a provider

You'll use the Administrative Console's Authentication Center to add an identity provider:

1. Login to Anaconda Enterprise, click the **Menu** icon ☰ in the top right corner, then click the **Administrative Console** link in the bottom of the slideout menu.
2. Click **Manage Users**.
3. Login to the Authentication Center using the Administrator credentials *configured after installation*.

4. In the Configure menu on the left, select **User Federation**.



5. Select `ldap` from the Add provider selector to display the initial **Required Settings** screen.

Multiple fields are required. The most important is the **Vendor** drop-down list, which will prefill default settings based on the LDAP provider you select. Make sure you select the correct one: `Active Directory`, `Red Hat Directory Server`, `Tivoli`, or `Novell eDirectory`. If none of these matches, select `Other` and coordinate with your LDAP Administrator to provide values for the required fields:

Username LDAP attribute Name of the LDAP attribute that will be mapped to the username. Active Directory installations may use `cn` or `sAMAccountName`. Others often use `uid`.

RDN LDAP attribute Name of the LDAP attribute that will be used as the RDN for a typical user DN lookup. This is often the same as the above "Username LDAP attribute", but does not have to be. For example, Active Directory installations may use `cn` for this attribute while using `sAMAccountName` for the "Username LDAP attribute".

UUID LDAP attribute Name of an LDAP attribute that will be unique across all users in the tree. For example, Active Directory installations should use `objectGUID`. Other LDAP vendors typically define a UUID attribute, but if your implementation does not have one, any other unique attribute (such as `uid` or `entryDN`) may be used.

User Object Classes Values of the LDAP `objectClass` attribute for users, separated by a comma. This is used in the search term for looking up existing LDAP users, and if read-write sync is enabled, new users will be added to LDAP with these `objectClass` values as well.

Connection URL The URL used to connect to your LDAP server. Click **Test connection** to make sure your connection to the LDAP server is configured correctly.

Users DN The full DN of the LDAP tree–the parent of LDAP users. For example, `'ou=users,dc=example,dc=com'`.

Authentication Type The LDAP authentication mechanism to use. The default is `simple`, which requires the Bind DN and password of the LDAP Admin.

Bind DN The DN of the LDAP Admin, required to access the LDAP server.

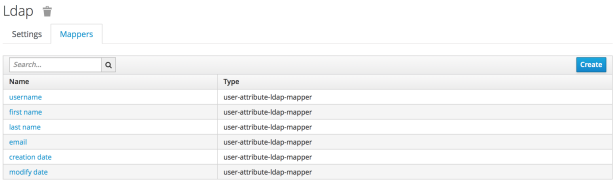
Bind Credential The password of the LDAP Admin, required to access the LDAP server. After supplying the DN and password, click **Test authentication** to confirm that your connection to the LDAP server can be authenticated.

Configuring sync settings

By default, users will not be synced from the LDAP / Active Directory store until they log in. If you have a large number of users to import, it can be helpful to set up batch syncing and periodic updates.

Configuring mappers

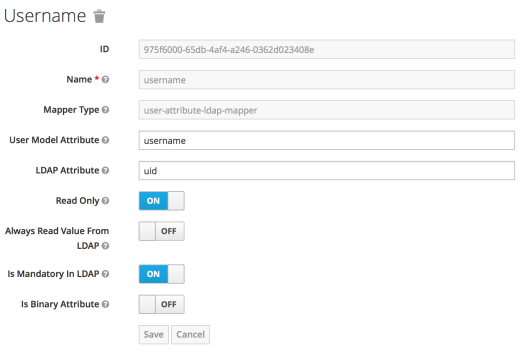
After you complete the initial setup, the auth system generates a set of "mappers" for your configuration. Each mapper takes a value from LDAP and maps it to a value in the internal auth database.



Go through each mapper and make sure it is set up appropriately.

- Check that each mapper reads the correct "LDAP attribute" and maps it to the right "User Model Attribute".
- Check that the attribute's "read-only" setting is correct.
- Check whether the attribute should always be read from the LDAP store and not from the internal database.

For example, the `username` mapper sets the Anaconda Enterprise username from the LDAP attribute configured.



Configuring advanced mappers

Instead of manually configuring each user, you can automatically import user data from LDAP using additional mappers. The following mappers are available:

User Attribute Mapper (`user-attribute-ldap-mapper`) Maps LDAP attributes to attributes on the AE5 user. These are the default mappers set up in the initial configuration.

FullName Mapper (`full-name-ldap-mapper`) Maps the full name of the user from LDAP into the internal database.

Role Mapper (`role-ldap-mapper`) Sets role mappings from LDAP into realm role mappings. One role mapper can be used to map LDAP roles (usually groups from a particular branch of an LDAP tree) into realm roles with corresponding names.

Multiple role mappers can be configured for the same provider. It's possible to map roles to a particular client (such as the `anaconda-deploy` service), but it's usually best to map in realm-wide roles.

Hardcoded Role Mapper (`hardcoded-ldap-role-mapper`) Grants a specified role to each user linked with LDAP.

Hardcoded Attribute Mapper (`hardcoded-ldap-attribute-mapper`) Sets a specified attribute to each user linked with LDAP.

Group Mapper (`group-ldap-mapper`) Sets group mappings from LDAP. Can map LDAP groups from a branch of an LDAP tree into groups in the Anaconda Platform realm. It will also propagate user-group membership from LDAP. We generally recommend using roles and not groups, so the role mapper may be more useful. **CAUTION**: The group mapper provides a setting `Drop non-existing groups during sync`. If this setting is turned on, existing groups in Anaconda Enterprise Authentication Center will be erased.

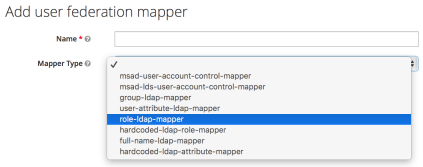
MSAD User Account Mapper (`msad-user-account-control-mapper`) Microsoft Active Directory (MSAD) specific mapper. Can tightly integrate the MSAD user account state into the platform account state, including whether the account is enabled, whether the password is expired, and so on. Uses the `userAccountControl` and `pwdLastSet` LDAP attributes.

For example if `pwdLastSet` is `0`, the user is required to update their password and there will be an `UPDATE_PASSWORD` required action added to the user. If `userAccountControl` is `514` (disabled account), the platform user is also disabled.

Mapper configuration example

To map LDAP group membership to Anaconda Platform roles, use a role mapper.

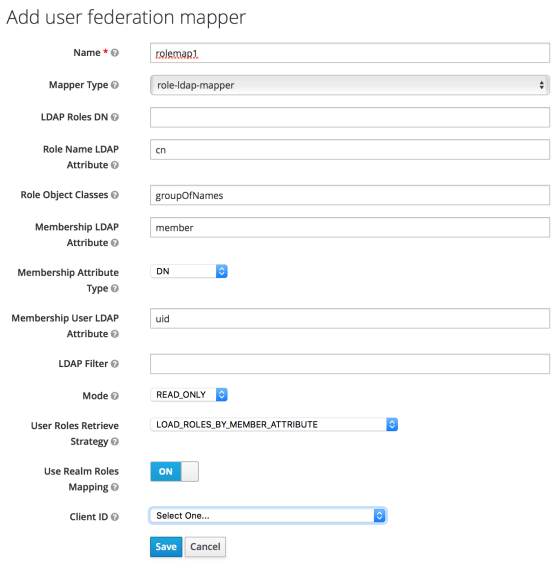
Add a mapper of the `role-ldap-mapper` type:



In consultation with your LDAP administrator and internal LDAP documentation, define which LDAP group tree will be mapped into roles in the Anaconda Platform realm. The roles are mapped directly by name, so an LDAP membership of `ae-deployer` will map to the role of the same name in Anaconda Platform.

Authorizing LDAP groups and roles

To authorize LDAP group members or roles synced from LDAP to perform various functions, add them to the `anaconda-platform.yml` configmap.



EXAMPLE: To give users in the LDAP group "AE5", and users with the LDAP-synced role "Publisher", permission to deploy apps, the deploy section would look like this:

```
deploy:
  port: 8081
  prefix: '/deploy'
  url: https://abc.demo.anaconda.com/deploy
  https:
    key: /etc/secrets/certs/privkey.pem
    certificate: /etc/secrets/certs/cert.pem
  hosts:
    - abc.demo.anaconda.com
  db:
    database: anaconda_deploy
  users: '*'
  deployers:
    users: []
    groups:
      - developers
      - AE5
    roles:
      - Publisher
```

After editing the configmap, restart all pods for your changes to take effect:

```
kubectl get pods | grep ap- | cut -d' ' -f1 | xargs kubectl delete pods
```

Configuring LDAPS (Outbound SSL)

To make correct requests to secure internal resources such as internal enterprise LDAP servers using corporate SSL certificates, you must configure a "trust store". *This is optional*. If your internal servers instead use certificates issued by a public root CA, then the default trust store is sufficient.

To create a trust store, you must have the public certificates you wish to trust available.

NOTE: These are certificates for your **trusted server** such as Secure LDAP, not for Anaconda Enterprise.

Option 1

If the CA certificates are directly available to you, run the following command, replacing `CAFILE.cert` with your CA certificate file:

```
keytool -import -file CAFILE.cert -alias auth -keystore LDAPS.jks
```

Option 2

Alternatively, if you also have the server certificate and key, you can construct a full trust chain in the store.

1. Convert the certificate and key files to PKCS12 format—if they are not already—by running the following command:

```
openssl pkcs12 -export -chain -in CERT.pem -inkey CERT-KEY.pem -out PKCS-CHAIN.
↪p12 -name auth -CAfile CA-CHAIN.pem
```

In this example, replace `CERT.pem` with the server's certificate, `CERT-KEY.pem` with the server's key, `PKCS-CHAIN.p12` with a temporary file name, and `CA-CHAIN.pem` with the trust chain file (up to the root certificate of your internal CA).

2. Create a Java keystore to store the trusted certs:

```
keytool -importkeystore -destkeystore LDAPS.jks -srckeystore PKCS-CHAIN.p12 -
↪alias auth
```

NOTE: You will be prompted to set a password. Record the password.

Final steps

For both options, you'll need to follow the steps below to expose the certificates to the Anaconda Enterprise Auth service:

1. Export the existing SSL certificates for your system by running the following commands:

```
sudo gravity enter
kubectl get secrets certs --export -o yaml > /opt/anaconda/secrets-exported.yml
```

2. Exit the gravity environment, and back up the secrets file before you edit it:

```
cp secrets-exported.yml secrets-exported-orig.yml
```

3. Run the following command to encode the newly created truststore as `base64`:

```
echo "  ldaps.jks: "$(base64 -i --wrap=0 OUTPUT.jks)
```

4. Copy the output of this command, and paste it into the `data` section of the `secrets-exported.yml` file.
5. Verify that the `LDAPS.jks` entry has been added to the secret:

```
kubectl describe secret anaconda-enterprise-certs
```

6. Run the following commands to update Anaconda Enterprise with the secrets certificate:

```
sudo gravity enter
kubectl replace -f /opt/anaconda/secrets-exported.yml
```

7. *Edit the platform configuration* by setting the `auth.https.truststore` configuration key to *the full path to* the `LDAPS.jks` file, and `auth.https.truststore-password` to the matching password.
8. Run the following commands to restart the `auth` service:

```
sudo gravity enter
kubectl get pods | grep ap-auth | cut -d' ' -f1 | xargs kubectl delete pods
```

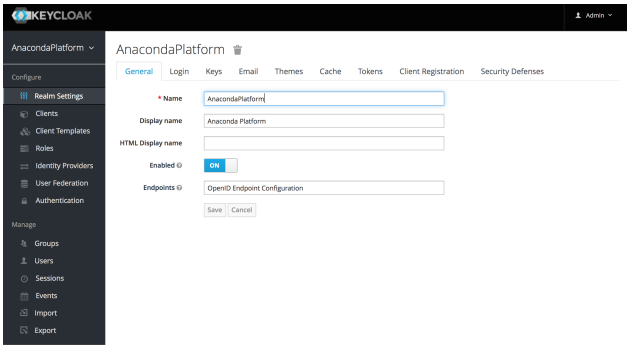
3.2.2 Managing users

Managing access to Anaconda Enterprise involves adding and removing users, setting passwords, *mapping users to roles and optionally assigning them to groups*. You'll use the Administrative Console's Authentication Center to perform these actions.

To help expedite the process of authorizing large groups of users at once, you can *connect to an external identity provider* using LDAP, Active Directory, SAML, or Kerberos to federate those users.

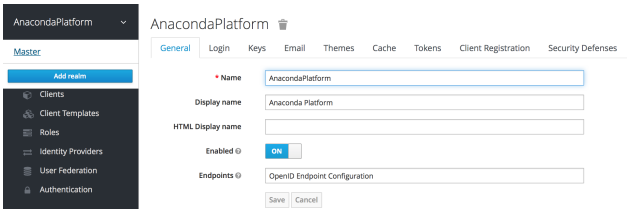
To access the Authentication Center:

1. Login to Anaconda Enterprise, click the **Menu** icon ☰ in the top right corner, then click the **Administrative Console** link in the bottom of the slideout menu.
2. Click **Manage Users**.
3. Login to the Authentication Center using the Administrator credentials *configured after installation*.



NOTE: To create and manage Authentication Center Administrators, use the realm selector in the upper left corner to switch to the **Master** realm before proceeding. To manage Admin users of the Operations Center, see *Managing System Administrators*.

4. In the Manage menu on the left, click **Users**.
5. On the **Lookup** tab, click **View all users** to list every user in the system, or search the user database for all users that match the criteria you enter, based on their first name, last name, or email address.



NOTE: This will search the local user database and not the federated database (such as LDAP) because not all external identity provider systems inlcude a way to page through users. If you want users from a federated database to be synced into the local database, select **User Federation** in the Configure menu on the left, and adjust the **Sync Settings** for your user federation provider.

6. To create a new Anaconda Enterprise user, click **Add user** and specify a username—and optionally provide values for the other fields—before clicking **Save**.

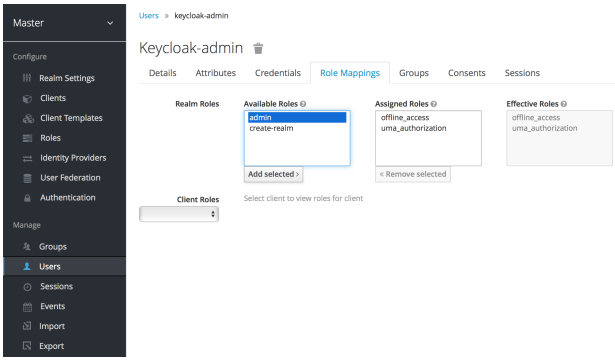
7. To configure a user, click the user's ID in the list and use the available tabs as follows:

   - Use the **Details** tab to specify information for the user, optionally enable *user registration* and *required actions* and *impersonate the user*. If you include an email address, an invitation to join Anaconda Enterprise will be sent to the email address specified.

   - Use the **Credentials** tab to manage the user's password. If the **Temporary** switch is on, this new password can only be used once–the user will be asked to change their password after they use it to log in to Anaconda Enterprise.

   - Use the **Role Mappings** tab to assign the user one or more roles, and the **Groups** tab to add them to one or more groups. See *managing groups and roles* for more information.

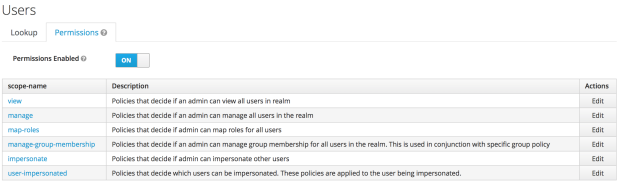
   NOTE: To grant Authentication Center Administrators sufficient authority to manage AE users, assign them the `admin` role.



   - Use the **Sessions** tab to view a summary of all sessions the user has started, and log them out of all sessions

in a single click. This is handy if a user goes on vacation without logging out of their sessions. You can use the Operations Center to view a summary of all sessions running on specific nodes or by specific users. See *monitoring sessions and deployments* for more information.

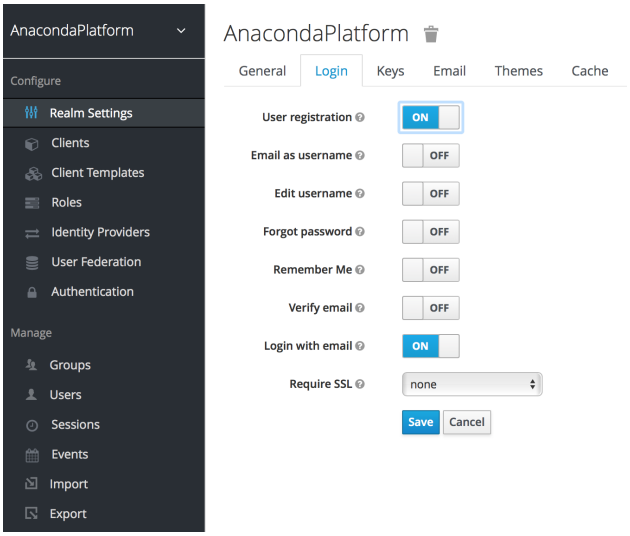
8. Return to the Users list and select the **Permissions** tab to view and edit a set of fine grain permissions that you can enable and use to define policies for allowing others to manage users in the selected realm.



Enabling user registration

You can use the Authentication Center to enable users to self register and create their own account. When enabled, the login page will have a **Register** link users can click to open the registration page where they can enter the user profile information and password required to create their new account.

1. Click **Realm Settings** under Configure in the menu on the left menu.
2. Click the **Login** tab, and enable the **User registration** switch.



You can change the look and feel of the registration form as well as removing or adding additional fields that must be entered. See the Server Developer Guide for more information.

Enabling required actions

You can use the **Required User Actions** drop-down list—on the **Details** tab for each user—to select the tasks that a user must complete (after providing their credientials) before they are allowed to log in to Anaconda Enterprise:

Update Profile This requires the user to update their profile information, such as their name, address, email, and phone number.

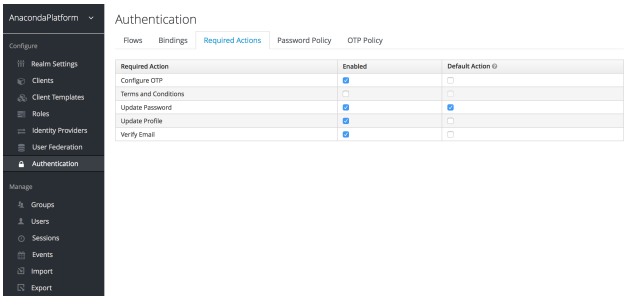
Update Password When set, a user must change their password.

Configure OTP When set, a user must configure a one-time password generator on their mobile device using either the Free OTP or Google Authenticator application.

Setting default required actions

You can specify default required actions that will be added to all new user accounts. Select **Authentication** from the Configure menu on the left and use the **Required Actions** tab to specify whether you want each required action to be enabled—available for selection—or also pre-populated as a default for all new users.

NOTE: A required action must be enabled to be specified as a default.



Using terms and conditions

Many organizations have a requirement that when a new user logs in for the first time, they need to agree to the terms and conditions of the website. This functionality can be implemented as a required action, but it requires some configuration. In addition to enabling **Terms and Conditions** as a required action, you must also edit the `terms.ftl` file in the *base* login theme. See the Server Developer Guide for more information on extending and creating themes.

Impersonating users

It is often useful for an Administrator to impersonate a user. For example, a user may be experiencing an issue using an application and an Admin may want to impersonate the user to see if they can duplicate the problem.

Any user with the realm's `impersonation` role can impersonate a user. The **Impersonate** command is available from both the Users list and the **Details** tab for a user.

When impersonating, if the Admin and the user are in the same realm, the Admin will be logged out and automatically logged in as the user being impersonated. If the Admin and user are not in the same realm, the Admin will remain logged in and be logged in as the user in that user's realm. In both cases, the browser will be redirected to the impersonated user's User Account Management page.

3.2.3 Managing roles and groups

Assigning access and permissions to individual users can be too fine-grained and cumbersome for organizations to manage, so Anaconda Enterprise enables you to assign access permissions to specific *roles*, then use *groups* to assign one or more roles to sets of users. Users inherit the attributes and role mappings assigned to each group they are members of—whether multiple or none.

The use of groups to assign permissions is entirely optional, so you can rely solely on roles to assign users permission to perform certain actions in Anaconda Enterprise.

- You'll use the Admin Console's Authentication Center to *create and manage roles and groups*. This includes creating new roles and groups, configuring defaults for each, and assigning roles to groups.
- You'll use the Admin Console's Operations Center to *configure permissions for any roles you create*, and optionally the default system roles provided by Anaconda Enterprise.

NOTE: When *naming* users and groups that you create, consider that Anaconda Enterprise users can add collaborators by user or group name when *sharing their projects* and *deployments*, as well as *packages and channels*.

To access the Authentication Center:

1. Login to Anaconda Enterprise, click the **Menu** icon ☰ in the top right corner, then click the **Administrative Console** link in the bottom of the slideout menu.
2. Click **Manage Users**.
3. Login to the Authentication Center using the Administrator credentials *configured after installation*.

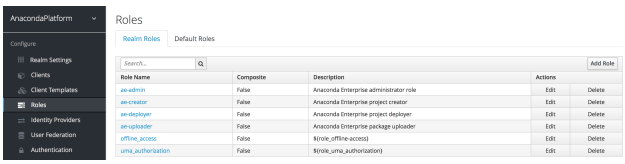
To manage roles:

Use roles to authorize individual or groups of users to perform specific actions withing Anaconda Enterprise. Default roles allow you to automatically assign user role mappings when any user is newly created or imported (for example, through *LDAP*.

You'll use the Authentication Center to configure new roles and specify default roles to be automatically added to all new user accounts.

1. In the Configure menu on the left, click **Roles** to display a list of roles configured for use with Anaconda Enterprise.

   To get you started, Anaconda Enterprise provides a set of "realm" roles. You can use these system roles as is, or as a basis for creating your own.



ae-admin Allows a user to act as an administrator for the platform, including access to the Authentication Center.

ae-creator Allows a user to create new projects.

ae-deployer Allows a user to create new deployments from projects.

ae-uploader Allows a user to upload packages.

NOTE: To define roles that are *global* to Anaconda Enterprise, use the realm selector in the upper left corner to switch to the **Master** realm before proceeding.

2. To create a new role, click **Add Role** on the **Realm Roles** tab.
3. Enter a name and description of the role, and click **Save**.

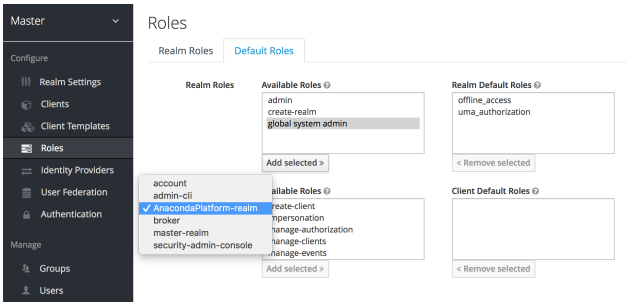
   NOTE: Roles can be assigned to users automatically or require an explicit request. If a user has to explicitly request a realm role, enable the **Scope Param Required** switch. The role must then be specified using the `scope` parameter when requesting a token.

The new role is now available to be used as a default role, or *to be assigned to groups of users*.

4. To configure default roles, click the **Default Roles** tab.

- When working with the **AnacondaPlatform** realm, you can configure default roles *for Anaconda Enterprise users* using the list of available and default **Realm Roles**.
- When working with the **Master** realm, you can configure defaut roles *for a specific client or service namespace* using the list of available and default roles for the client you select from the **Client Roles** drop-down list.

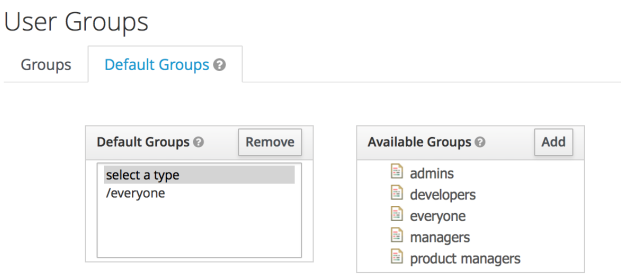
NOTE: To customize the list of roles available for Anaconda Enterprise Admins to use, select **AnacondaPlatform-realm** from the list.



To manage groups:

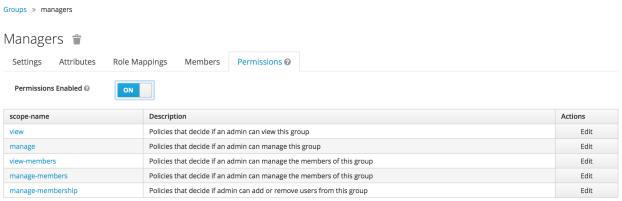
1. In the Manage menu on the left, click **Groups** to display a list of groups configured for use with Anaconda Enterprise.

   To get you started, Anaconda Enterprise provides a set of default groups, with different role mappings for each. You can use these defaults as is, or as a basis for creating your own. Default groups allow you to automatically assign group membership whenever a new user is created or imported.



2. Double-click the name of a group to view information about the group and modify it:

- Use the **Role Mappings** tab to assign roles to the group from the list of available **Realm Roles** and **Client Roles**. See *managing roles* for information on how to create new roles. Permission to perform certain actions in Anaconda Enterprise are based on a user's *role*, so you can grant permissions to a group of users by mapping the associated role(s) with the group. See the section below for the steps to *configure permissions by role*.
- Use the **Members** tab to view all users who currently belong to the group. You add users to groups *at the user level* using the **Groups** tab *for the user*. See *managing users* for more information.
- Use the **Permissions** tab to enable a set of fine grain permissions to use to *define policies* for allowing Admin users to manage the group. See the section below to understand how to *configure permissions by role*.

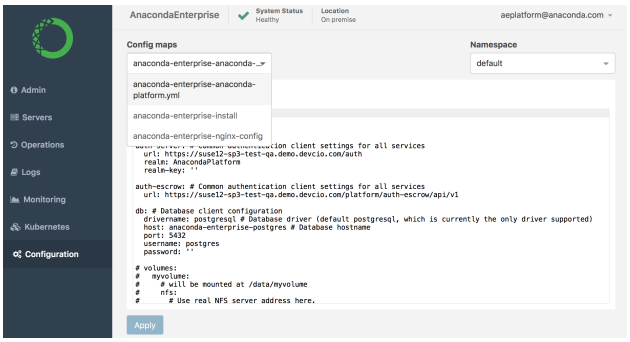


To configure permissions for roles:

1. Log in to Anaconda Enterprise, select the **Menu** icon ☰ in the top right corner and click the **Administrative Console** link displayed at the bottom of the slide out window.
2. Click **Manage Resources**.
3. Log in to the Operations Center using the Administrator credentials *configured after installation*.
4. Select **Configuration** from the menu on the left to display the config map for Anaconda Enterprise.

NOTE: If `anaconda-platform.yml` *is not* displayed, be sure `anaconda-enterprise-anaconda-platform.yml` is selected in the **Config maps** drop-down list.

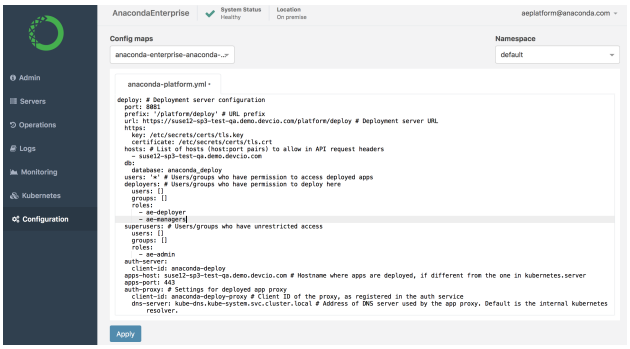
The following sections of the config map have permissions associated with them:



- **deploy:deployers**—used to configure which users can deploy projects
- **workspace:users**—used to configure which users can open project sessions
- **storage:creators**—used to configure which users can create projects
- **repository:uploaders**—used to configure which users can upload packages to the AE repository

5. Add each new role you create to the appropriate section—based on the permission you want to grant the role—and click **Apply** to save your changes.

For example, if you create a new role called `ae-managers`, and you want users with this role to be able to deploy applications, you need to add that role to the list of roles under `deploy:deployers` to map the permission to the role.



3.2.4 Managing System Administrators

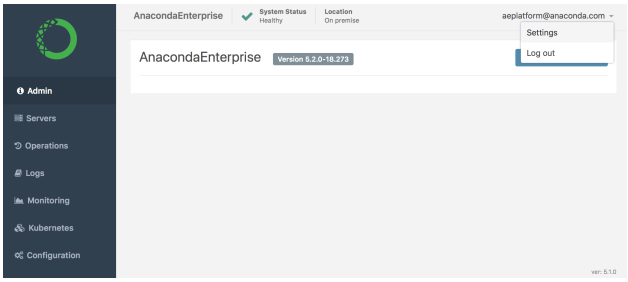
Anaconda Enterprise distinguishes between Administrators responsible for authorizing AE platform *users*, and Administrators responsible for managing AE *resources*. This enables enterprises to grant the permissions required for configuring each to different individuals, based on their area of responsibility within the organization.

- Use the **Authentication Center** to configure authentication *for all users*, including Admins. See *Configuring authentication* for more information on configuring authentication for Anaconda Enterprise.
- Use the **Operations Center** to authorize Admin users to configure and manage AE resources. Follow the steps outlined below.

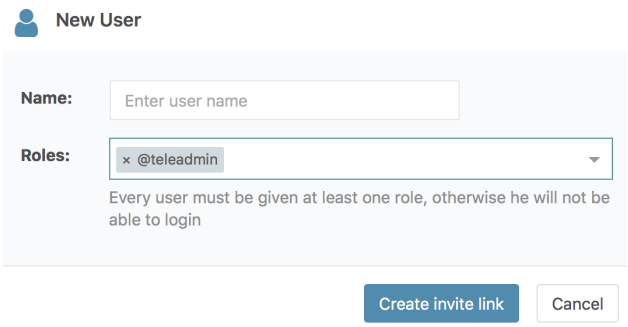
  See *Managing cluster resources* for more information on configuring Anaconda Enterprise platform settings and resources.

To manage Operations Center Admins:

1. Log in to Anaconda Enterprise, select the **Menu** icon ☰ in the top right corner and click the **Administrative Console** link displayed at the bottom of the slide out window.
2. Click **Manage Resources**.
3. Login to the Operations Center using the Administrator credentials *configured after installation*.
4. Select **Settings** in the login menu in the upper-right corner.

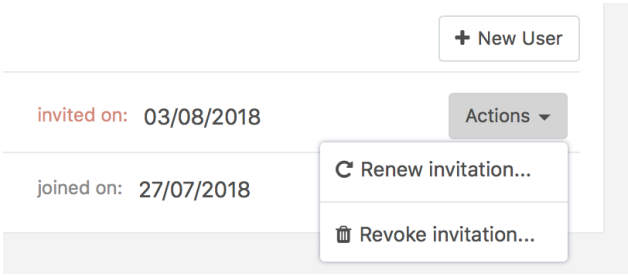


5. In the left menu, select **Users**, then click **+ New User** in the upper-right corner.
6. Select `@teleadmin` from the **Roles** drop-down list, and click **Create invite link**.



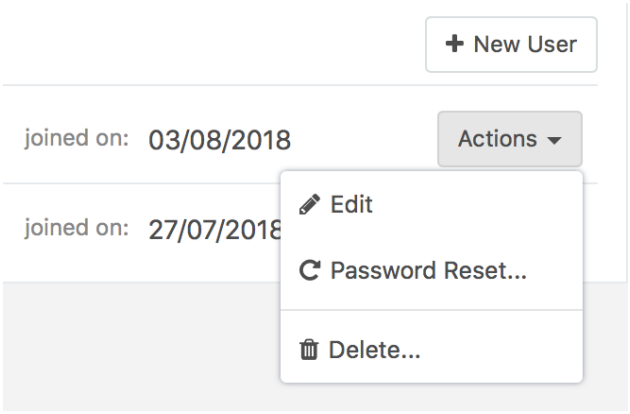
7. Copy the invitation URL that is generated, replace the private IP address with the fully-qualified domain name of the host, if necessary, and send it to the individual using your preferred method of secure communication. They'll use it to set their password, and will be automatically logged in to the Operations Center when they click **Continue**.

   To generate a new invitation URL, select **Renew invitation** in the **Actions** menu for the user.



Select **Revoke invitation** to prevent them from being able to use the invitation to create a password and access the Operations Center. This effectively deletes the user before they have a chance to set their credentials.

To delete—or otherwise manage—an Operations Center user after they have set their credentials and completed the authorization process, select the appropriate option from the **Actions** menu.



3.3 Configuring channels and packages

Anaconda Enterprise enables you to distribute software through the use of *channels* and *packages*.

Channels represent locations of repositories where Anaconda Enterprise looks for packages.

Packages are used to bundle software files and information about the software—such as its name, specific version and description—into a single file that can be easily installed and managed.

NOTE: Anaconda Enterprise supports the use of both conda and pip packages in its repository.

You can use any of the following methods to access packages:

- Mirror *the entire Anaconda repository* or *specific packages*.
- Mirror packages in a repository *in an airgapped environment without internet access*.
- Point to a cloud-based repository or a private location on a remote or local repository that you or your organization created. See *Accessing remote package repositories* for more information.

You can copy packages from one channel into another, customize each channel by including different versions of packages, and delete channels when they are no longer needed. See *Managing channels and packages* for more information.

3.3.1 Accessing remote package repositories

As an Administrator, you can configure Anaconda Enterprise to use packages from an online package repository such as `anaconda` and `r`.

You can then *mirror channels & packages* into your organization's internal AE repository so users can access the packages from a centralized, on-premises location.

If users are permitted to install packages from off-site package repositories, you can make it easier for users to access them from within their editing sessions by configuring them as default channels.

To do so, edit your Anaconda Enterprise configuration—`anaconda-enterprise-anaconda-platform.yml`—to include the appropriate channels, as follows:

```
conda:
  channels:
  - defaults
  default-channels:
  - https://repo.anaconda.com/pkgs/
  - https://repo.anaconda.com/pkgs/free/
  - https://repo.anaconda.com/pkgs/r/
  - https://repo.anaconda.com/pkgs/pro/
  channel-alias: https://<ANACONDA_ENTERPRISE_FQDN>:30089/conda
```

To update Anaconda Enterprise with your changes to the configuration, restart its services:

```
sudo gravity enter
kubectl get pods | grep 'ap-' | cut -d' ' -f1 | xargs kubectl delete pods
```

3.3.2 Mirroring channels and packages

Anaconda Enterprise enables you to create a local copy of a repository so users can access the packages from a centralized, on-premises location.

The mirror can be complete, partial, or include specific packages or types of packages. You can also create a mirror in an air gapped environment to help improve performance and security.

NOTE: It can take hours to mirror the full repository.

Before you can use Anaconda Enterprise's convenient syncing tools to configure local mirrors for channels and packages, you'll need to *configure access to the source of the packages* to be mirrored, whether an online repository or a tarball (if an airgapped installation).

Prerequisites:

- *Installing and configuring Anaconda Enterprise*
- *Extracting self-signed SSL certificates*

Types of mirroring:

- To create a complete mirror, see *Mirroring the Anaconda repository* or *Mirroring a PYPI repository*.
- To create partial mirror, see *Mirroring specific packages*.
- To mirror a repository in a system without internet access, see *Mirroring in an air-gapped environment*.
- To share mirrors, see *Configuring Anaconda Enterprise* and *Sharing channels*.

Configuration options:

- *Configuring SSL verification*
- *Mirroring in a proxied environment*
- *Mirroring specific platforms and versions*
- *Using blacklisting to exclude packages*
- *Using whitelisting to include packages*
- *Combining multiple configuration arguments*

Anaconda Enterprise supports the following platforms:

- linux-64
- osx-64

- win-64

Before you begin, you need to have completed *installing and configuring Anaconda Enterprise*.

1. Install anaconda-enterprise-cli

   The Anaconda Enterprise installer tarball contains a `cas-mirror-<version>.sh` script, which is a custom Miniconda installation that contains the `anaconda-enterprise-cli` package.

   NOTE: `bzip` is required to install packages from Miniconda.

   Navigate to the directory where you downloaded and extracted the Anaconda Enterprise installer, then install the bootstrap Miniconda environment to `~/cas-mirror`:

```
$ cd anaconda-enterprise-<version>
$ ./cas_mirror-<version>-linux-64.sh
Welcome to cas_mirror <anaconda-enterprise-installer_version>
[...]
```

   NOTE: Replace <version> with your version number.

   At this installer prompt: `Do you wish the installer to prepend the cas_mirror install location to PATH in your /home/centos/.bashrc ?`. We recommend you choose `yes`.

   After the installer finishes, close and re-open your terminal window for the installation to take effect.

   In your new terminal window, activate the custom Miniconda environment with the following command:

```
source ~/cas-mirror/bin/activate
```

2. Configure Anaconda URL using the following commands:

```
anaconda-enterprise-cli config set sites.master.url https://anaconda.example.com/
↪repository/api
anaconda-enterprise-cli config set default_site master
```

   NOTE: Replace `anaconda.example.com` with the domain name you are using.

3. Verify and add SSL certificates:

   If the root CA is contained in the certificate bundle at `/etc/pki/tls/certs/ca-bundle.crt`, use `openssl` to verify the certificates and make sure the final `Verify return code` is `0`:

```
openssl s_client -connect anaconda.example.com:443 -CAfile /etc/pki/tls/certs/ca-
↪bundle.crt
...
    Verify return code: 0 (ok)
```

   If you are using privately signed certificates, *extract the rootca*, then use `openssl` to verify the certificates and make sure the final `Verify return code` is `0`:

```
openssl s_client -connect anaconda.example.com:443 -CAfile rootca.crt
...
    Verify return code: 0 (ok)
```

   Configure the SSL certificates for the repository using the following commands:

```
$ anaconda-enterprise-cli config set ssl_verify true

# On Ubuntu
```

(continues on next page)

(continued from previous page)

```
$ anaconda-enterprise-cli config set sites.master.ssl_verify /etc/ssl/certs/ca-
↪certificates.crt

# On CentOS/RHEL
$ anaconda-enterprise-cli config set sites.master.ssl_verify /etc/pki/tls/certs/
↪ca-bundle.crt
```

NOTE: If you are using a self-signed certificate or a certificate signed by a private CA, *extract the rootca*, then either:

- Use the `anaconda-enterprise-cli config set sites.master.ssl_verify` command to add that root certificate

–or–

- Add that root certificate to the default Ubuntu or CentOS/RedHat trusted CA bundles.

4. Log into Anaconda Enterprise as an existing user using the following command:

```
$ anaconda-enterprise-cli login
Username: anaconda-enterprise
Password:
Logged anaconda-enterprise in!
```

NOTE: If Anaconda Enterprise 5 is installed in a proxied environment, see *Mirroring in a proxied environment* for information on setting the NO_PROXY variable.

Extracting self-signed SSL certificates

You may need the temporary self-signed Anaconda Enterprise certificates for later use. For example, when installing the `anaconda-enterprise-cli` tool, you will need to configure it to point to the self-signed certificate authority.

First, enter the Anaconda Enterprise environment:

```
sudo gravity enter
```

Then, run the following command for each certificate file you wish to extract, replacing `rootca.crt` below with the name of the specific file:

```
kubectl get secrets certs -o go-template='{{index .data "rootca.crt"}}' | base64 -d >␣
↪/ext/share/rootca.crt
```

After you run this command, the file will be available on the master node filesystem at `/var/lib/gravity/planet/share/<filename>`.

The following certificate files are available:

- `rootca.crt`: the root certificate authority bundle
- `tls.crt`: the SSL certificate for individual services
- `tls.key`: the private key for the above certificate
- `wildcard.crt`: the SSL certificate for "wildcard" services, such as deployed apps and sessions
- `wildcard.key`: the private key for the above certificate

- `keystore.jks`: the Java Key Store containing these certificates used by some services

Mirroring the Anaconda repository

An example configuration file is provided here for mirroring the default Anaconda packages for the `linux-64` platform. These files are also included in the mirror tool installation:

```
# This is destination channel of mirrored packages on your local repository.
dest_channel: anaconda

# conda packages from these channels are mirrored to dest_channel on your local␣
↪repository.
channels:
  - https://repo.continuum.io/pkgs/main/
  - https://repo.continuum.io/pkgs/free/
  - https://repo.continuum.io/pkgs/pro/

# if doing a mirror from an airgap tarball, the channels should point to the tarball:
# channels:
#   - file:///path-to-expanded-tarball/repo-mirrors-<date>/anaconda-suite/pkgs/

# Only conda packages of these platforms are mirrored.
# Omitting this will mirror packages for all platforms available on specified␣
↪channels.
# If the repository will only be used to install packages on the v5 system, it only␣
↪needs linux-64 packages.
platforms:
  - linux-64
```

Mirror the contents of the repository:

```
cas-sync-api-v5 --file ~/cas-mirror/etc/anaconda-platform/mirrors/anaconda.yaml
```

This mirrors all of the packages from the Anaconda repository into the anaconda channel. If the channel does not already exist, it will be automatically created and shared with all authenticated users.

You can customize the permissions on the mirrored packages by *sharing the channel*.

Verify in your browser by logging into your account and navigating to the **Packages** tab. You should see a list of the mirrored packages.

Mirroring a PyPI repository

The full PyPI mirror requires approximately 120GB, so ensure that your file storage location has sufficient disk space before proceeding. You use a configuration file such as `$PREFIX/etc/anaconda-server/mirrors/pypi.yaml` to customize the mirror behavior and specify the subset of packages you want to mirror.

To create a PyPI mirror:

```
anaconda-enterprise-cli mirror pypi --config pypi.yaml
```

This command loads the packages on `https://pypi.org` into the `~pypi binstar` user account.

The following configuration options are available for you to customize your configuration file:

Name	Description
`user`	The local user under which the PyPI packages are imported. Default: `pypi`.
`pkg_list`	A list of packages to mirror. Only packages listed are mirrored. If this is set, `blacklist` and `whitelist` settings are ignored. **Default:** `[]`.
`whitelist`	A list of packages to mirror. Only packages listed are mirrored. If the list is empty, all packages are checked. Default: `[]`.
`blacklist`	A list of packages to skip. The packages listed are ignored. Default: `[]`.
`latest_only`	Only download the latest versions of the packages. Default: `false`.
`remote_url`	The URL of the PyPI mirror. `/pypi` is appended to build the XML RPC API URL, `/simple` for the simple index and `/pypi/{package}/{version}/json` for the JSON API. Default: `https://pypi.python.org/`.
`xml_rpc_url`	A custom value for XML RPC URL. If this value is present, it takes precedence over the URL built using `remote_url`. Default: `null`.
`simple_index_url`	A custom value for the simple index URL. If this value is present, it takes precedence over the URL built using `remote_url`. Default: `null`.
`use_xml_rpc`	Whether to use the XML RPC API as specified by PEP381. If this is set to `true`, the XML RPC API is used to determine which packages to check. Otherwise the scripts falls back to the simple index. If the XML RPC fails, the simple index is used. Default: `true`.
`use_serial`	Whether to use the serial number provided by the XML RPC API. Only packages updated since the last serial saved are checked. If this is set to false, all PyPI packages are checked for updates. Default: `true`.
`create_org`	Create the mirror user as an organization instead of a regular user account. All superusers are added to the "Owners" group of the organization. Default: `false`.
`private`	Save the mirrored packages as private. **Default:** `false`.

EXAMPLE:

```
whitelist:
  - requests
  - six
  - numpy
  - simplejson
latest_only: true
remote_url: https://pypi.org/
use_xml_rpc: true
```

Configuring pip

To configure pip to use this new mirror, edit `/etc/pip.conf` as follows:

```
[global]
index-url=https://pypi.anaconda.org/pypi/simple
```

Mirroring specific packages

Alternately, you may not wish to mirror all packages. In this case, you can specify which platforms or specific packages you want to mirror —*or*— use the whitelist, blacklist or license_blacklist functionality to control which packages are mirrored, by editing the provided mirror files. *You cannot combine these methods*. For more information, see *Mirror configuration options*.

```
cas-sync-api-v5 --file ~/my-custom-anaconda.yaml
```

Mirroring R packages

An example configuration file for mirroring R packages is also provided:

```
# This is destination channel of mirrored packages on your local repository.
dest_channel: r

# conda packages from these channels are mirrored to dest_channel on your local␣
↪repository.
channels:
  - https://repo.continuum.io/pkgs/r/

# if doing a mirror from an airgap tarball, the channels should point to the tarball:
# channels:
#   - file:///path-to-expanded-tarball/repo-mirrors-<date>/r/pkgs/

# Only conda packages of these platforms are mirrored.
# Omitting this will mirror packages for all platforms available on specified␣
↪channels.
# If the repository will only be used to install packages on the v5 system, it only␣
↪needs linux-64 packages.
platforms:
  - linux-64
```

```
cas-sync-api-v5 --file ~/cas-mirror/etc/anaconda-platform/mirrors/r.yaml
```

Mirroring in an air-gapped environment

To mirror the repository in a system with no internet access, create a local copy of the repository using a USB drive provided by Anaconda, and point `cas-sync-api-v5` to the extracted tarball.

First, mount the USB drive and extract the tarball. In this example we will extract to `/tmp`:

```
cd /tmp
tar xvf <path to>/mirror.tar
```

NOTE: Replace `<path to>` with the actual path to the mirror file.

Now you have a local file-system repository located at `/tmp/mirror/pkgs`. You can mirror this repository by editing `/etc/anaconda-platform/mirrors/anaconda.yaml` to contain:

```
channels:
  - /tmp/mirror/pkgs
```

And then run the command:

```
cas-sync-api-v5 --file etc/anaconda-platform/mirrors/conda.yaml
```

This mirrors the contents of the local file-system repository to your Anaconda Enterprise installation under the username `anaconda`.

Configuring Anaconda Enterprise

After creating the mirror, *edit your Anaconda Enterprise configuration* to add this new mirrored channel to the default Anaconda Enterprise channels and make the packages available to users.

```
conda:
  channels:
  - defaults
  default-channels:
  - anaconda
  - r
  channel-alias: https://<anaconda.example.com>/repository/conda
```

Replacing `<anaconda.example.com>` with the actual URL to your installation of Anaconda Enterprise.

NOTE: The `ap-workspace` pod must be restarted for the configuration change to take effect on new project editor sessions.

Sharing channels

To make your new channels visible to your users in their **Channels** list, you need to share the channels with them.

EXAMPLE: To share new channels `anaconda` and `r` with group `everyone` for read access:

```
anaconda-enterprise-cli channels share --group everyone --level r anaconda
anaconda-enterprise-cli channels share --group everyone --level r r
```

After running the `share` command, verify by logging onto the user interface and viewing the **Channels** list.

For more information, see *Sharing channels and packages*

Mirror configuration options

You can use the following options to configure your mirror:

- *Configuring SSL verification*.
- *Mirroring in a proxied environment*.
- *Mirroring specific platforms and versions*.
- *Using blacklisting to exclude packages*.
- *Using whitelisting to include package*.
- *Combining multiple configuration arguments*.

remote_url

Specifies the remote URL from which the conda packages and the Anaconda and Miniconda installers are downloaded. The default value is: `https://repo.continuum.io/`.

channels

Specifies the remote channels from which conda packages are downloaded. The default is a list of the channels `<remote_url>/pkgs/free/` and `<remote_url>/pkgs/pro/`

All specification information should be included in the same file, and can be passed to the cas-sync-api-v5 command via the `--file` argument:

```
cas-sync-api-v5 --file ~/cas-mirror/etc/anaconda-platform/mirrors/anaconda.yaml
```

destination channel

The configuration option `dest_channel` specifies where files will be uploaded. The default value is: `anaconda`.

SSL verification

The mirroring tool uses two different settings for configuring SSL verification. When the mirroring tool connects to its destination, it uses the `ssl_verify` setting from `anaconda-enterprise-cli` to determine how to validate certificates. For example, to use a custom certificate authority:

```
anaconda-enterprise-cli config set sites.master.ssl_verify /etc/ssl/certs/ca-
↪certificates.crt
```

The mirroring tool uses conda's configuration to determine how to validate certificates when connecting to the source that it is pulling packages from. For example, to disable certificate validation when connecting to the source:

```
conda config --set ssl_verify false
```

Mirroring in a proxied environment

If Anaconda Enterprise 5 is installed in a proxied environment, set the `NO_PROXY` variable. This ensures the mirroring tool does not use the proxy when communicating with the repository service, and prevents errors such as `Max retries exceeded`, `Cannot connect to proxy`, and `Tunnel connection failed: 503 Service Unavailable`.

```
export NO_PROXY=<master-node-domain-name>
```

Platform-specific mirroring

By default, the `cas-sync-api-v5` tool mirrors all platforms. If you do not need all platforms, edit the YAML file to specify the platform(s) you want mirrored:

```
platforms:
  - linux-64
  - win-32
```

NOTE: The platform argument is evaluated before any other argument.

Package-specific mirroring

In some cases you may want to mirror only a small subset of the repository. Rather than blacklisting a long list of packages you do not want mirrored, you can instead simply enumerate the list of packages you DO want mirrored.

NOTE: This argument cannot be used with the `blacklist`, `whitelist` or `license_blacklist` arguments—it can only be combined with platform-specific and version-specific mirroring.

EXAMPLE:

```
pkg_list:
  - accelerate
  - pyqt
  - zope
```

This example mirrors only the three packages: Accelerate, PyQt & Zope. All other packages will be completely ignored.

Python version-specific mirroring

Mirror the repository with a Python version or versions specified.

EXAMPLE:

```
python_versions:
  - 3.3
```

Mirrors only Anaconda packages built for Python 3.3.

License blacklist mirroring

The mirroring script supports license blacklisting for the following license families:

```
AGPL
GPL2
GPL3
LGPL
BSD
MIT
Apache
PSF
Public-Domain
Proprietary
Other
```

EXAMPLE:

```
license_blacklist:
  - GPL2
  - GPL3
  - BSD
```

This example mirrors all the packages in the repository EXCEPT those that are GPL2-, GPL3-, or BSD-licensed, because those three licenses have been blacklisted.

Blacklist mirroring

The blacklist allows access to all packages EXCEPT those explicitly listed. If the `license_blacklist` and `blacklist` arguments are combined, `license_blacklist` is evaluated first, and `blacklist` is a supplemental modifier.

EXAMPLE:

```
blacklist:
  - bzip2
  - tk
  - openssl
```

This example mirrors the entire repository EXCEPT the `bzip2`, `Tk`, and `OpenSSL` packages.

Whitelist mirroring

The `whitelist` argument adds or includes packages that would be otherwise excluded by the `blacklist` and/or `license_blacklist` functions.

EXAMPLE:

```
license_blacklist:
  - GPL2
  - GPL3
whitelist:
  - readline
```

This example mirrors the entire repository EXCEPT any GPL2- or GPL3-licenses packages, but includes `readline`, despite the fact that it is GPL3-licensed.

Combining multiple mirror configurations

You may find that combining two or more of the arguments above is the easiest way to get the exact combination of packages that you want.

NOTE: The platform argument is evaluated before any other argument.

EXAMPLE: This example mirrors only Linux-64 distributions of the dnspython, Shapely and GDAL packages:

```
platforms:
  - linux-64
pkg_list:
  - dnspython
  - shapely
  - gdal
```

If the `license_blacklist` and `blacklist` arguments are combined, `license_blacklist` is evaluated first, and `blacklist` is a supplemental modifier.

EXAMPLE: In this example, the mirror configuration does not mirror GPL2-licensed packages. It does not mirror the GPL3 licensed package `pyqt` because it has been blacklisted. It does mirror all other packages in the repository:

```
license_blacklist:
  - GPL2
blacklist:
  - pyqt
```

If the `blacklist` and `whitelist` arguments are both employed, the blacklist is evaluated first, with the whitelist functioning as a modifier.

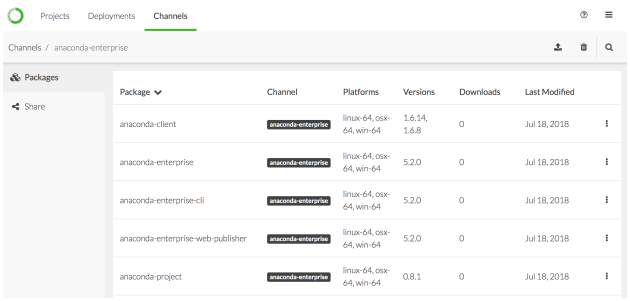
EXAMPLE: This example mirrors all packages in the repository except `astropy` and `pygments`. Despite being listed on the blacklist, accelerate is mirrored because it is listed on the whitelist.

```
blacklist:
  - accelerate
  - astropy
  - pygments
whitelist:
  - accelerate
```

3.3.3 Managing channels and packages

Anaconda Enterprise makes it easy for you to manage the various channels and packages used by your organization—whether you prefer using the UI or the CLI.

1. Log in to the console using the Administrator credentials *configured after installation*.
2. Select **Channels** in the top menu to view the list of existing channels, each channel's owner and when the channel was last updated.
3. Click on a *channel* name to view details about the packages in the channel, including the supported platforms, versions and when each package in the channel was last modified. You can also see the number of times each package has been downloaded.



4. Click on a *package* name to view the list of files that comprise the package, and the command used to install the package.

CAUTION: The `anaconda-enterprise` channel is used for internal purposes only, and should not be modified.

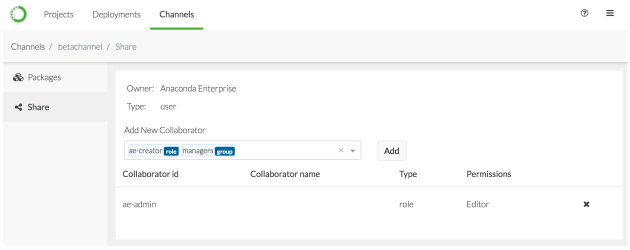
To add a package to an existing channel, click the **Upload package** icon in the upper right corner and browse for the package.

NOTE: There is a 1GB file size limit for package files you upload.

To remove a package from a channel, select **Delete** from the command menu for the package.

To create a new channel and add packages to the channel for others to access:

1. Click the **Create Channel** icon in the top right corner, enter a meaningful name for the channel and click **Create**.
2. Upload the appropriate packages to the channel.
3. To make the channel available to others, click **Share** and add them as a collaborator. You can share a channel with individual users, groups of users, or based on role—the easiest way to control access to a channel. See *Managing roles and groups* for more information.



NOTE: The default is to grant collaborators read-write access, so if you want to prevent them from adding and removing packages from the channel, be sure they have read-only access. You'll need to *use the CLI* to make a channel read-only.

Using the CLI:

Get a list of all the channels on the platform with the `channels list` command:

```
anaconda-enterprise-cli channels list
```

Share a channel with a specific user using the `share` command:

```
anaconda-enterprise-cli channels share --user username --level r <channelname>
```

You can also share a channel with an existing group:

```
anaconda-enterprise-cli channels share --group GROUPNAME --level r <channelname>
```

Replacing GROUPNAME with the actual name of the group.

NOTE: Adding `--level r` grants this group read-only access to the channel.

You can "unshare" a channel using the following command:

```
anaconda-enterprise-cli channels share --user <username> --remove <channelname>
```

Run `anaconda-enterprise-cli channels --help` to see more information about what you can do with channels.

For help with a specific command, enter that command followed by `--help`:

```
anaconda-enterprise-cli channels share --help
```

3.4 Generating custom Anaconda installers

As an Anaconda Enterprise Administrator, you can *create custom environments* for other users to access. These environments include specific packages and their dependencies. You can then *create a custom installer* for the environment.

You can also provide users with *standard Anaconda and Miniconda installers*, Cloudera Manager parcels, and Hortonworks Data Manager management packs.

You distribute installers to users via *channels*. After you *add an installer to a channel*, authorized users will be able to access it from their **Channels** list. Because each custom installer includes the specific packages and their dependencies, when users download them, they'll have everything they need to use them. See *Using installers, parcels and management packs* for more information.

To create an environment:

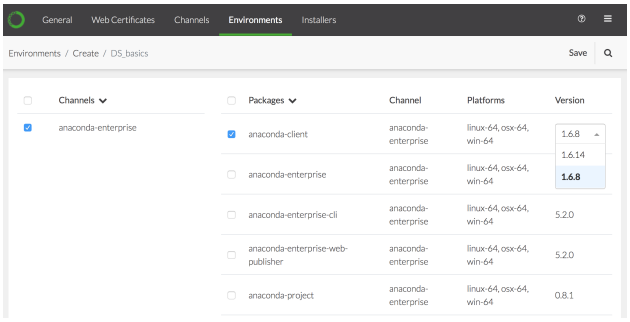
1. Log in to the console using the Administrator credentials *configured after installation*.
2. Select **Environments** in the top menu to view the list of existing environments, version and package information for each, and when the environment was last updated.

   Click on an environment name to view details about the packages included in the environment.

3. Click the **Create Environment** icon  in the upper right corner, give the environment a unique name and click **Save**.

   NOTE: Environment names can contain alphanumeric characters and underscores only.

4. Check the channel you want to choose packages from, then select the specific packages–and version of each–you want to include in the installer.



5. Click **Save** in the window banner to create the environment. Anaconda Enterprise resolves all the package dependencies and displays the environment in the list. If there is an issue resolving the dependencies, you'll be notified and prompted to edit the environment.

You can now use the environment as a basis for creating additional versions of the environment or other environments.

To edit an existing environment:

1. Click on an environment name to view it's details, then click the **Edit Environment** icon .
2. Change the channels and/or packages included in the environment and enter a version number for the updated package before clicking **Save**. The new version is displayed in the list of environments

To copy an environment:

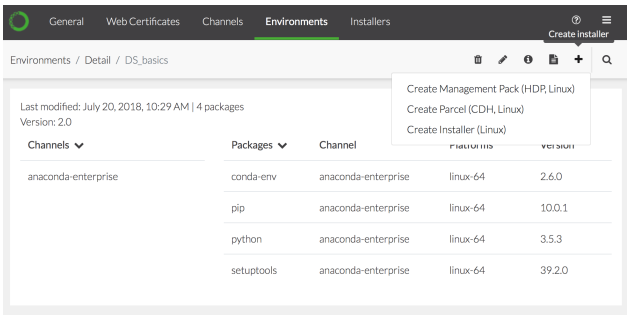
Open the environment for editing, click the **Duplicate Environment** icon and enter a unique name for the environment before clicking **Save**. Now you can edit the environment.

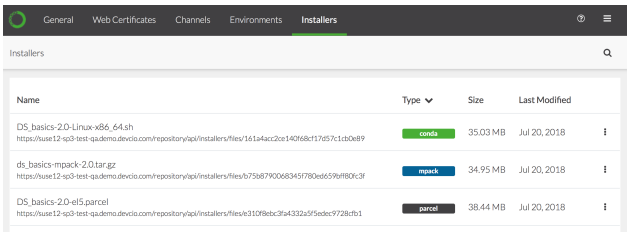
Now that you've created an environment, you can create an installer for it.

To create a custom installer:

1. Open the environment's Detail view, click the **Create installer** icon , and select the type of installer you want to create:

Anaconda Enterprise creates the installer and displays it in the **Installers** list:





2. To view the relevant logs or delete the installer, click the ⋮ icon and choose the appropriate command.

If you created a management pack, you'll need to install it on your Hortonworks HDP cluster and add it to your local Ambari server to make it available to users. For more information, see this blog post about generating custom management packs.

If you created a parcel, you'll need to install it on your Cloudera CDH cluster to make it available to users. For more information, see this blog post about generating custom parcels.

To access standard Anaconda and Miniconda installers:

Download the installers from https://repo.anaconda.com/ then follow the instructions below to add them to a channel and share them with others.

Adding an installer to a channel

1. Select **Channels** to view the list of existing channels, each channel's owner and when it was last updated.
2. Select a channel—or create a new channel—and click the **Upload package** icon in the upper right corner to browse for and upload the installer package to the channel.
3. To make the installer available to others, click **Share** in the menu on the left and add them as a collaborator to the channel.

   NOTE: You can share the installer with multiple users at once by adding their group or role. See *Managing roles and groups* for more information.

When the user logs in to Anaconda Enterprise, they'll be able to access the installer by clicking the **Installers** link from within their **Channels** list. For more information, see *Using installers, parcels and management packs*.

3.5 Managing TLS/SSL certificates

3.5.1 Anaconda Enterprise user interface certificates

After *initial configuration*, configuring SSL certificates for Anaconda Enterprise is done from the Anaconda Enterprise Administrative Settings menu.

NOTE: TLS and SSL certificates for the Operations Center web endpoint are configured with a separate process described below in *Anaconda Enterprise Operations Center certificates*.

This section explains how to change administrative settings through the Anaconda Platform web UI.

1. In the top-right corner of the Anaconda Enterprise screen, click the user icon.
2. In the menu that appears, select Administrative Settings.

Configuring SSL certificates

Assemble the following:

- Registered domain name
- SSL certificate for `servername.domain.tld`, filename `tls.crt`
- SSL private key for `servername.domain.tld`, filename `tls.key`
- Root SSL certificate (such as this default Root CA), filename `rootca.crt`. A root certificate is optional but recommended.
- SSL intermediate chain/bundle, filename `intermediate.pem`
- Wildcard domain name
- SSL wildcard certificate for `*.servername.domain.tld`, filename `wildcard.crt`. A wildcard certificate is not necessary if the existing SSL certificate has a Subject Alternative Name for the wildcard domain. If you're not sure, ask your network administrator.
- SSL private key for `*.servername.domain.tld`, filename `wildcard.key`. An SSL private key is not necessary if the existing SSL certificate has a Subject Alternative Name for the wildcard domain. If you're not sure, ask your network administrator.

Copy and paste the files from the previous step as shown:

NOTE: To change the wildcard domain, you must both change it in this interface and change it by editing the config map as described in *Managing platform settings*.


ANACONDA ENTERPRISE
Projects
Deployments
Packages
Adminstrative Settings
Web Certificates
Web Certificates
Anaconda Enterprise ships with self-signed certificates. To use your own certificates, please select the certificates that you wish to use and replace the certificates included by default
Secret last updated at: 22 minutes ago
Domain name
SSL Certificate
Paste value here
SSL private key
Paste value here
Root certificate (Optional)
Paste value here
Intermediate cert (Optional)
Paste value here
Wildcard domain (Optional)
Wildcard certificate (Optional)
Paste value here
Wildcard private key (Optional)
Paste value here
RESET
SAVE

NOTE: If you have upgraded from a previous version of Anaconda Enterprise, you may need to update your configuration to make sure all services are referencing the correct SSL certificate filenames.

In Anaconda Enterprise 5.1.0 and newer, the default SSL certificate filenames provided by the installer are different than in previous versions. We recommend that you update any Kubernetes secrets you created and update the Anaconda Enterprise configuration to match the new filenames.

The Kubernetes secret is named `anaconda-enterprise-certs` and can be modified with these commands:

```
sudo gravity enter
kubectl edit secrets anaconda-enterprise-certs
```

After updating the Kubernetes secret, restart all pods:

```
kubectl get pods | grep ap- | cut -d' ' -f1 | xargs kubectl delete pods
```

These are the new filenames:

Previous	Updated
`rootca.pem`	`rootca.crt`
`cert.pem`	`tls.crt`
`privkey.pem`	`tls.key`
`tls.crt`	`wildcard.crt`
`tls.key`	`wildcard.key`

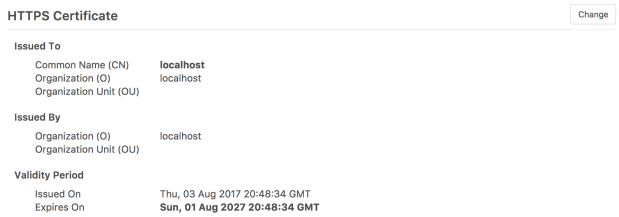
NOTE: the `keystore.jks` filename is unchanged.

3.5.2 Anaconda Enterprise Operations Center certificates

NOTE: The configuration of TLS/SSL certificates in the Operations Center applies only to the Operations Center web endpoint. Configuration of TLS/SSL certificates for all other Anaconda Enterprise services is a separate configuration process that is documented in the *Post-install configuration* section.

To add a new HTTPS certificate for the Operations Center, from any page of the Operations Center open the Settings menu by clicking the icon next to your username, then from the drop-down menu that appears, select Settings.

From the left navigation menu, click the HTTPS Certificate link.



To change any of the settings, click the Change button in the upper right, then follow the directions below for each field.

Adding a private key

After you have received a Private Key from an SSL provider, save the file to your local computer. Then from the Operations Center's HTTPS Certificate page, click the Browse... button next to the Private Key form box, locate the SSL file on your computer, and click the Upload or Open button.

Click the Save button for the file to be uploaded.

Adding an HTTPS certificate

After you have received the HTTPS Certificate from an SSL provider, save the file to your local computer. Then from the Operations Center's HTTPS Certificate page, click the Browse... button next to the HTTPS Certificate form box, locate the Certificate file on your computer, and click the Upload or Open button.

Click the Save button for the file to be uploaded.

Adding an intermediary certificate

If you are using an Intermediate Certificate, save the certificate file to your local computer. Then from the Operations Center's Intermediate Certificate page, click the Browse... button next to the HTTPS Certificate form box, locate the Certificate file on your computer, and click the Upload or Open button.

Click the Save button for the file to be uploaded.

Setting Operations Center certificates on the command line

An Operations Center web UI and API TLS key pair can be configured with the `tlskeypair` resource using the `gravity` binary on the command line.

Make a `tlskeypair.yaml` file similar to this:

```
kind: tlskeypair
version: v2
metadata:
  name: keypair
spec:
  private_key: |
    -----BEGIN RSA PRIVATE KEY-----
  cert: |
    -----BEGIN CERTIFICATE-----
```

NOTE: The `cert` section must include all intermediate certificate PEM blocks concatenated.

To update the key pair:

```
gravity resource create tlskeypair.yaml
```

To view the currently configured key pair:

```
gravity resource get tls
```

To delete a TLS key pair:

```
gravity resource rm tls keypair
```

If the TLS key pair is deleted, the default self-signed TLS key pair will be used instead.

3.6 Managing platform settings

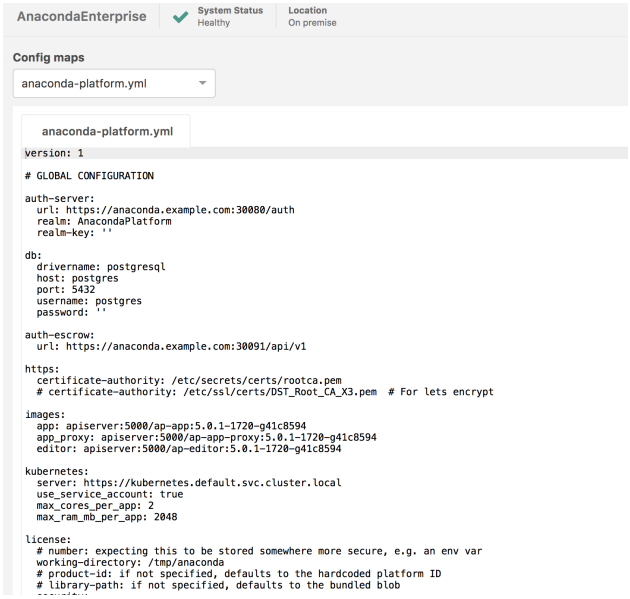
3.6.1 Editing Anaconda Enterprise global configuration

You'll use the Operations Center to configure the Anaconda Enterprise platform.

In the top right corner click Menu, then in the bottom right corner click Administrative Console, then click the Manage Resources button and log in.

In the left navigation click the Configuration link.

From the top right "Config Maps" box, select the name of the configuration file that you want to view or edit.



All the configuration of Anaconda Enterprise and the underlying Kubernetes system can be accessed and changed in this view.

The default configuration file `anaconda-platform.yml` is displayed.

The Anaconda Enterprise configuration YAML file, `anaconda-platform.yml`, contains both global and per-service configuration settings for your Anaconda Enterprise installation. You can edit these directly from the Operations Center.

This yaml-format file contains the locations and specifications for the following:

- Admin
- Authentication
- Authentication escrow
- Authentication server
- Conda channels and defaults
- Database

- Deploy
- Git
- HTTPS
- Images
- Kubernetes
- Offline docs
- PostgreSQL
- Repository
- Resource profiles
- S3
- S3 Client
- Sessions
- Storage
- Sync
- UI

Download a `sample` anaconda-platform configuration file.

Anaconda Enterprise configuration is available in the "default" Namespace:

- anaconda-platform.yml– main Anaconda Enterprise configuration file
- nginx-config– modify configuration of nginx web server

Kubernetes specific configuration is available in the "kube-system" Namespace:

- alerting-addresses–modify email alerts
- dashboard-default–modify default dashboard configuration
- extension-apiserver-authentication– modify client-ca-file
- grafana–modify grafana.ini
- gravity-opscenter–modify gravity.yaml, proxy.cert, proxy.key or teleport.yaml

Editing a configuration file with the user interface

NOTE: Any changes are made directly to the configuration file, so we strongly recommend that you copy and save a copy of the original file before making any edits.

To edit any of the above configuration files, after selecting the file from the "Config Maps" box, begin typing in the in-page text editor.

After editing, at the bottom left an Apply button appears. Click the Apply button to save your work.

If you attempt to abandon your edits by navigating away from the page, you will see a warning "You have unsaved changes!" You can choose to navigate away with the "Disregard and continue" button, or return to your editing with the "Close" button.

After making changes, restart the appropriate service so the changes will take effect. You can restart all services with these commands:

```
sudo gravity enter
kubectl get pods | grep ap- | cut -d' ' -f1 | xargs kubectl delete pods
```

Editing a configuration file on the command line

Use these commands to edit a configmap with the command line:

```
sudo gravity enter
kubectl edit cm anaconda-enterprise-anaconda-platform.yml
```

Make your changes to the file, and save it.

Restart all pods:

```
kubectl get pods | grep ap- | cut -d' ' -f1 | xargs kubectl delete pods
```

3.6.2 Mounting an NFS share

Anaconda Enterprise can mount NFS shares where users can store data and source code.

To add an NFS share to your Anaconda Enterprise installation, edit the configuration file.

The configuration file's `volumes` section shows example configuration for mounting NFS volumes:

```
# volumes:
#   myvolume:
#     nfs:
#       # Use real NFS server address here.
#       server: ###.###.###
#       # Use real NFS server export directory.
#       path: "/export"
#       readOnly: true
```

Uncomment this section and enter the server IP and path. The path must match the path of the shared directory on your NFS server.

If `readOnly` is `true` then the NFS volume is mounted as a read-only share, and if `readOnly` is `false` then the NFS volume is mounted as a read-write share.

The name of the volume section determines the path where this volume will be mounted. In this case, since the section is named `myvolume`, the share will be mounted at `/data/myvolume`. To mount multiple volumes, create a second section similar to the `myvolume` block and give it a unique name.

After making changes, restart the appropriate service so the changes will take effect. You can restart all services with these commands:

```
sudo gravity enter
kubectl get pods | grep ap- | cut -d' ' -f1 | xargs kubectl delete pods
```

After these changes take effect, editor sessions, applications, and jobs will mount the NFS share to `/data/myvolume`.

Troubleshooting

The NFS share is mounted using a service account `anaconda` on the Anaconda Enterprise server. To mount the share as writable, configure the server so that the `anaconda` user account (uid 1000) has write permissions.

If mounting is configured and the NFS server goes offline, then editor sessions, applications, and jobs will not start. If NFS downtime occurs, we recommend disabling mounting. To disable mounting:

- Edit the configuration file.
- Comment out the `volume` section.
- Restart all services as described above.

3.7 Configuring AE 5

3.7.1 Configuration reference

This section explains how to configure the resource limits and the security settings in the Anaconda Enterprise configuration file, `anaconda-platform.yml`.

To access the Anaconda Enterprise configuration file, login to the Anaconda Enterprise Operations Center and click the Configuration link.

NOTE: We strongly recommend to save a copy of the original file before making any edits.

Complete Anaconda Enterprise configuration file

See the full configuration file below, or `download` a copy to review.

Additional details regarding the configuration settings are shown below.

```
version: 1

# GLOBAL CONFIGURATION

auth-server: # Common authentication client settings for all services
  url: https://anaconda.example.com/auth
  realm: AnacondaPlatform
  realm-key: ''

auth-escrow: # Common authentication client settings for all services
  url: https://anaconda.example.com/platform/auth-escrow/api/v1

db: # Database client configuration
  drivername: postgresql # Database driver (default postgresql, which is currently␣
↪the only driver supported)
  host: anaconda-enterprise-postgres # Database hostname
  port: 5432
  username: postgres
  password: ''

# volumes:
#   myvolume:
#     # will be mounted at /data/myvolume
#     nfs:
```

(continues on next page)

(continued from previous page)

```
#       # Use real NFS server address here.
#       server: ###.###.###
#       # Use real NFS server export directory.
#       path: "/"
#       readOnly: true
#   volume2:
#     # will be mounted at /data/volume2
#     nfs:
#       # Use real NFS server address here.
#       server: ###.###.###
#       # Use real NFS server export directory.
#       path: "/"
#       readOnly: true

https: # Common HTTPS client and server settings for all services
  certificate-authority: /etc/secrets/certs/rootca.crt # Path to Certificate␣
↪Authority bundle for private CA or self-signed certificates
  # certificate-authority: /etc/ssl/certs/DST_Root_CA_X3.pem  # For lets encrypt

resource-profiles:

  default:
    description: 'Default resource profile'
    user_visible: true
    resources:
      limits:
        cpu: '2'
        memory: '4096Mi'
  system_job:
    description: 'System job resource profile'
    user_visible: false
    resources:
      limits:
        cpu: '2'
        memory: '4096Mi'

kubernetes:
  server: https://kubernetes.default.svc.cluster.local
  use_service_account: true
  max_cores_per_app: 2
  max_ram_mb_per_app: 2048

license:
  # number: PASTE_LICENSE_CODE_OR_CLIENT_ID_HERE
  # key: PASTE_OFFLINE_KEY_HERE_FOR_OFFLINE_ACTIVATION
  working-directory: /tmp/anaconda
  security:
    x: 207
    y: 705
    z: 278
  analytics:
    enabled: true

admin:
  users:
    # Any user matching the users, group affiliations, or roles (in the auth service)
    # described below is a platform-wide administrator.
```

(continues on next page)

(continued from previous page)

```
    users: []
    groups: []
    roles:
      - ae-admin

# PER-SERVICE CONFIGURATION

auth: # Authentication server configuration
  port: 9080
  db:
    database: anaconda_auth
  https: # HTTPS configuration
    keystore: /etc/secrets/certs/keystore.jks # Name of server keystore in Java␣
↪keystore (.jks) format
    keystore-password: anaconda # Keystore password defined when generating the Java␣
↪keystore
    key-alias: auth # Name of the key in the keystore
    # truststore: null # (optional) Path to the trust store to use for outgoing HTTPS␣
↪requests (e.g. for LDAPS)
    # truststore-password: null # (optional) Truststore password defined when␣
↪generating the Java keystore
  debug: False # If true, enable use of a pregenerated SSL key for testing. DO NOT␣
↪SET TO TRUE IN PRODUCTION.
  import-file: /etc/secrets/keycloak/keycloak.json
  api: # Service settings for auth-api
    port: 9090
    limit: 12
    prefix: '/platform/auth-api'
    https:
      key: /etc/secrets/certs/tls.key
      certificate: /etc/secrets/certs/tls.crt
  escrow: # Service settings for auth-escrow
    port: 9091
    db:
      database: anaconda_auth_escrow
    hosts: # List of hosts (host:port pairs) to allow in API request headers
      - anaconda.example.com
    prefix: '/platform/auth-escrow'
    https:
      key: /etc/secrets/certs/tls.key
      certificate: /etc/secrets/certs/tls.crt
    auth-server:
      client-secret: REDACTED
      client-id: anaconda-platform

deploy: # Deployment server configuration
  port: 8081
  prefix: '/platform/deploy' # URL prefix
  url: https://anaconda.example.com/platform/deploy # Deployment server URL
  https:
    key: /etc/secrets/certs/tls.key
    certificate: /etc/secrets/certs/tls.crt
  hosts: # List of hosts (host:port pairs) to allow in API request headers
    - anaconda.example.com
  db:
    database: anaconda_deploy
  users: '*' # Users/groups who have permission to access deployed apps
```

(continues on next page)

(continued from previous page)

```
  deployers: # Users/groups who have permission to deploy here
    users: []
    groups: []
    roles:
      - ae-deployer
  superusers: # Users/groups who have unrestricted access
    users: []
    groups: []
    roles:
      - ae-admin
  auth-server:
    client-id: anaconda-deploy
  apps-host: anaconda.example.com # Hostname where apps are deployed, if different␣
↪from the one in kubernetes.server
  apps-port: 443
  auth-proxy: # Settings for deployed app proxy
    client-id: anaconda-deploy-proxy # Client ID of the proxy, as registered in the␣
↪auth service
    dns-server: kube-dns.kube-system.svc.cluster.local # Address of DNS server used␣
↪by the app proxy. Default is the internal kubernetes resolver.
    https:
      key: /etc/secrets/certs/wildcard.key
      certificate: /etc/secrets/certs/wildcard.crt

  debug: False # If true, enable debugging. DO NOT SET TO TRUE IN PRODUCTION.

workspace: # Workspace server configuration
  port: 8090
  prefix: '/platform/workspace' # URL prefix
  url: https://anaconda.example.com/platform/workspace # Workspace server URL
  https:
    key: /etc/secrets/certs/tls.key
    certificate: /etc/secrets/certs/tls.crt
  hosts: # List of hosts (host:port pairs) to allow in API request headers
    - anaconda.example.com
  db:
    database: anaconda_workspace
    # increased pool size to accommodate for Kubernetes slowness
    pool:
      size: 10
      overflow: 20
      timeout: 60

  users: '*' # Users/groups who have permission to create workspace sessions
  superusers: # Users/groups who have unrestricted access
    users: []
    groups: []
    roles:
      - ae-admin

  auth-server:
    client-id: anaconda-workspace-api
  email-domain: anaconda.example.com # Domain name for generating email addresses if␣
↪not set in auth service
  workspace-host: anaconda.example.com # Hostname where workspace sessions are hosted,
↪ if different from the one in kubernetes.server
  workspace-port: 443
```

(continues on next page)

(continued from previous page)

```
  auth-proxy: # Settings for workspace access control proxy
    client-id: anaconda-workspace # Client ID of the proxy, as registered in the auth␣
↪service
    dns-server: kube-dns.kube-system.svc.cluster.local # Address of DNS server used␣
↪by the app proxy. Default is the internal kubernetes resolver.
    https:
      key: /etc/secrets/certs/wildcard.key
      certificate: /etc/secrets/certs/wildcard.crt

  debug: False # If true, enable debugging. DO NOT SET TO TRUE IN PRODUCTION.

storage: # Storage server configuration
  host: anaconda.example.com # full hostname of the storage server
  port: 8086
  prefix: '/platform/storage' # URL prefix
  hosts: # List of hosts (host:port pairs) to allow in API request headers
    - anaconda.example.com
  url: https://anaconda.example.com/platform/storage # Base URL of storage server
  db:
    database: anaconda_storage
  https:
    key: /etc/secrets/certs/tls.key
    certificate: /etc/secrets/certs/tls.crt
  git:
    default:
      name: Example.com Anaconda Enterprise Server # human-readable name of this git␣
↪server
      type: internal # server type. There is support for "internal" and planned␣
↪support for "github" and "gitlab".
      # use 127.0.0.1 for all network connections including hairpin
      url: https://127.0.0.1:8088/ # URL of git server
      repository: '{owner}-{id}' # Template for repository names; use {name}, {id},␣
↪and {owner} as placeholders.
      auth-header: Anaconda-User # Name of HTTP header for proxy authentication␣
↪(internal server type only)
      username: anaconda # Username of git service account
      # no password needed when using auth-header
      proxy:
        url: https://anaconda.example.com/platform/git # URL of git proxy
        client-id: anaconda-git-proxy # Auth client ID of this proxy
        dns-server: kube-dns.kube-system.svc.cluster.local # IP address of DNS server␣
↪used by the git proxy.
        run-as-user: www-data # System user account to run the proxy under
        api-key: REDACTED # secret api key to allow storage service API calls through␣
↪the proxy. Should be uniquely generated for each installation.
        port: 8095
        probe-port: 8096
        https:
          key: /etc/secrets/certs/tls.key
          certificate: /etc/secrets/certs/tls.crt
  objects:
    projects: # storage location for objects in projects. You may use placeholders
↪{name} {owner} and {id} for project name, project owner and project ID.
      bucket: anaconda-projects
      path: projects/{owner}-{id}
    global: # storage location for global objects (available to all logged-in users)
      bucket: anaconda-objects
```

(continues on next page)

(continued from previous page)

```
      path: 'global/'
    public: # storage location for public objects (available to everyone without␣
↪logging in)
      bucket: anaconda-objects
      path: 'public/'
    staging: # storage location for temporary objects
      bucket: anaconda-objects
      path: 'staging/'
  users: '*' # Users/groups who can create projects
  creators: # Users/groups who can create new projects
    users: []
    groups: []
    roles:
      - ae-creator
  superusers: # Users/groups who have unrestricted access
    users: []
    groups: []
    roles:
      - ae-admin

repository: # Repository server configuration
  url: https://anaconda.example.com/repository
  port: 8089
  hosts: # List of hosts (host:port pairs) to allow in API request headers
    - anaconda.example.com
    - 127.0.0.1:8089
  prefix: '/repository' # URL prefix
  db:
    database: anaconda_repository
  https:
    key: /etc/secrets/certs/tls.key
    certificate: /etc/secrets/certs/tls.crt
  users: '*' # Users/groups who can access the repository
  uploaders: # Users/groups who can create and upload packages
    users: []
    groups: []
    roles:
      - ae-uploader
  superusers: # Users/groups who have unrestricted access
    users: []
    groups: []
    roles:
      - ae-admin
  bucket: anaconda-repository # S3/object storage bucket to store repository files
  cleanup-upload-seconds: 3600 # How long an unfinished upload will be kept before␣
↪being cleaned up
  cleanup-period-seconds: 73 # How frequently the server will check for files that␣
↪should be removed from disk
  index-update-cooldown-seconds: 7 # How much time without new uploads is required␣
↪before index will be rebuilt
  index-update-period-seconds: 23 # How frequently the server will check for channels␣
↪that require rebuilding of index information (repodata.json)

s3: # configuration for the object-storage service
  host: 0.0.0.0 # full hostname of the object store server S3 API
  port: 8087
  https:
```

(continues on next page)

(continued from previous page)

```
    key: /etc/secrets/certs/tls.key
    certificate: /etc/secrets/certs/tls.crt
  access-key: REDACTED
  secret-key: REDACTED
  directory: /export

s3-client: # configuration for clients to the object storage service
  endpoint-url: https://anaconda.example.com # AWS endpoint URL
  access-key: REDACTED
  secret-key: REDACTED
  region-name: 'us-east-1' # the AWS region where your S3 bucket is located

git:
  url: https://anaconda.example.com/platform/git # externally visible URL of the git␣
↪server
  host: anaconda.example.com # full hostname of the git server
  port: 8088
  https:
    key: /etc/secrets/certs/tls.key
    certificate: /etc/secrets/certs/tls.crt
  db:
    database: anaconda_git
  directory: /export # directory where git server will store its data
  username: anaconda # OS username that the git server should run under
  lfs-secret: REDACTED # LFS authentication token secret. Should be uniquely␣
↪generated for each installation.
  secret-key: REDACTED # git server secret key. Should be uniquely generated for each␣
↪installation.

# when installing in airgap replace channel: defaults with the following
# - https://anaconda.example.com/repository/conda/anaconda
conda: # Common conda settings for editing sessions and deployments
  channels:
    - defaults
  default-channels: [] # List of channels that should be used for channel 'defaults'
  channel-alias: https://anaconda.example.com/repository/conda # Default conda URL␣
↪prefix for channels given by name only

sync:
  lfs-threshold: 1000000000000
  prefix: ''
  hosts: []
  port: 8093
  project-dir: ''

offline_docs:
  url: https://anaconda.example.com/docs # Docs server URL
  hosts: # List of hosts (host:port pairs) to allow in API request headers
    - anaconda.example.com
  port: 8091
  https:
    key: /etc/secrets/certs/tls.key
    certificate: /etc/secrets/certs/tls.crt
  directory: docs/_build/ # The path relative to the base directory of the static␣
↪docs.
  prefix: '/docs' # URL prefix
```

(continues on next page)

(continued from previous page)

```
ui: # Anaconda Platform UI server configuration
  base-url: / # URL prefix
  cookie-secret: REDACTED # secret key used to sign session cookies
  cookie-session:
    name: anaconda-platform-ui-session-v1
    expiration-hours: 9
  cookie-next:
    name: anaconda-platform-ui-next-v1
  db:
    database: anaconda_ui
  debug: False # If true, enable debugging. DO NOT SET TO TRUE IN PRODUCTION.
  host: anaconda.example.com # full hostname of the UI server
  public-url: https://anaconda.example.com/ # User-facing URL of site, if different␣
↪than host/port
  https:
    key: /etc/secrets/certs/tls.key
    certificate: /etc/secrets/certs/tls.crt
  port: 6990
  auth-server:
    client-secret: REDACTED
    client-id: anaconda-platform
  services:
    anaconda-storage:
      storage:
        icon: fa-anaconda
        label: Storage
        url: https://anaconda.example.com/platform/storage/api/v1
    anaconda-deploy:
      deploy:
        icon: fa-anaconda
        label: Deploy
        url: https://anaconda.example.com/platform/deploy/api/v1
    anaconda-workspace:
      workspace:
        icon: fa-anaconda
        label: workspace
        url: https://anaconda.example.com/platform/workspace/api/v1
        options:
          workspace:
            tools:
              notebook:
                default: true
                label: Jupyter Notebook
                packages: [notebook]
              jupyterlab:
                label: JupyterLab
                packages: [jupyterlab]
              anaconda-platform-sync:
                label: Anaconda Project Sync
                packages: [anaconda-platform-sync]

    anaconda-repo5:
      repo:
        html-url: https://anaconda.example.com/repository
        icon: fa-anaconda
        label: Repo Service
        url: https://anaconda.example.com/repository/api
```

(continues on next page)

(continued from previous page)

```
  auth-api:
    auth-api:
      icon: fa-anaconda
      label: Auth API
      url: https://anaconda.example.com/platform/auth-api/api/v1
  anaconda-enterprise-notebooks:
    aen:
      html-url: https://notebooks.example.com
      icon: fa-anaconda
      label: Enterprise Notebooks
      disabled: true
  documentation:
    offline_docs:
      html-url: https://anaconda.example.com/docs/
      icon: fa-anaconda
      label: Documentation
      url: https://anaconda.example.com/docs/
  github:
    github_com:
      html-url: https://github.com
      icon: fa-github
      label: GitHub
      url: https://api.github.com
      disabled: true
admin-links:
  admin:
    label: Admin Console
    external: true
    href: https://anaconda.example.com/auth/admin/
    position: 1
    acl:
      users: []
      groups: []
      roles:
        - ae-admin
help:
  docs:
    label: Anaconda Documentation - Home
    external: true
    href: https://anaconda.example.com/docs/
    position: 0
  started:
    label: Getting Started with Anaconda Enterprise
    external: true
    href: https://anaconda.example.com/docs/user-guide/getting-started/
    position: 1
  release:
    label: Release Notes
    external: true
    href: https://anaconda.example.com/docs/release-notes.html
    position: 2
  support:
    label: Support
    external: true
    href: https://anaconda.example.com/docs/help-support.html
    position: 3
```

(continues on next page)

(continued from previous page)

```
    feedback:
      label: Feedback
      external: true
      href: https://continuum.typeform.com/to/TnHsme
      position: 4

postgresql: # PostgreSQL server configuration
  port: 7080
```

Setting resource limits for project editor sessions and deployments

Each project editor session and deployment uses compute resources on the Anaconda Enterprise cluster. To configure the number of cores and amount of memory/RAM available to users—so that it corresponds to your specific system configuration and the needs of your users—you create custom *resource profiles*.

Resource profiles apply to all users, nodes, editor sessions, and deployments in the cluster. So if your installation includes nodes with GPUs that you want to make available for users to acclerate computation within their projects, you'd *create a GPU resource profile*. Any resource profiles you configure are listed for users to select from when they configure and deploy their projects. Anaconda Enterprise finds the node that matches their request.

Required security settings

These values and credentials must be set for every installation.

- `s3.access-key` and `s3.secret-key` for the Minio internal object store
- `s3-client.access-key` and `s3-client.secret-key` for the object store client. When using the internal object store, these must match `s3.access-key` and `s3.secret-key`.
- `auth.https.keystore-password` matching the password used when creating the Java keystore for the auth service
- `git.lfs-secret` and `git.secret-key` for the internal git server
- `storage.git.<server>.proxy.api-key`
- `ui.cookie-secret`

Configuring outbound SSL (for systems such as Secure LDAP)

See *Connecting to external identity providers*.

3.7.2 Pointing conda to an on-premises repository

This page explains the configuration to install packages from your local Anaconda Repository.

You can configure conda to search for packages in your on-premises repository. This can either be done at the system level, which overrides any user-level configuration files installed by the user, or on an individual machine basis.

Either way, you create a .condarc system configuration file in the root directory.

Edit your system `~/.condarc` file to add the appropriate channel:

```
channel_alias: https://<anaconda.enterprise>/repository/conda/
```

NOTE: Replace `<anaconda.enterprise>` with the actual URL to your installation of Anaconda Enterprise.

Users can log in to the on-premises repository and install packages from the on-premises repository by *installing the Anaconda command line interface (CLI)*.

Allowing access to other package repositories

If users are permitted to install packages from off-site package repositories, it is convenient to provide apps and editing sessions with access to the Anaconda channels by default.

To do so, *edit your Anaconda Enterprise configuration* to include the appropriate channels, as below:

```
conda:
  channels:
  - defaults
  default-channels:
  - https://repo.continuum.io/pkgs/main
  - https://repo.continuum.io/pkgs/free
  - https://repo.continuum.io/pkgs/r
  - https://repo.continuum.io/pkgs/pro
  channel-alias: https://<ANACONDA_ENTERPRISE_FQDN>/repository/conda
```

As with all changes to the configuration, you must then restart the Anaconda Enterprise services:

```
sudo gravity enter
kubectl get pods | grep 'ap-' | cut -d' ' -f1 | xargs kubectl delete pods
```

3.7.3 Using Anaconda Navigator with Anaconda Enterprise

The Anaconda Distribution includes two options for package and environment management on local systems, the command line program Conda and the graphical interface Anaconda Navigator.

If your users will use Navigator online, you may need to whitelist the necessary sites in your network's firewall settings. If your users will use Navigator offline, you may wish to send them information about offline mode, and/or change the icons in Navigator that link to the web.

Firewall settings

Normally Anaconda Navigator is used online, so that it can download and install packages.

In online mode, Navigator must be able to reach these sites, so they may need to be whitelisted in your network's firewall settings.

- https://repo.anaconda.com (or for older versions of Navigator and Conda, https://repo.continuum.io)
- https://conda.anaconda.org for conda-forge and other channels on Anaconda Cloud (anaconda.org)
- https://vscode-update.azurewebsites.net/ for updating Visual Studio Code
- google-public-dns-a.google.com (8.8.8.8:53) to check internet connectivity with Google Public DNS

Offline mode

If Navigator detects that internet access is not available, it automatically enables offline mode and displays this message:

"**Offline mode**

Some of the functionality of Anaconda Navigator will be limited. Conda environment creation will be subject to the packages currently available on your package cache.

Offline mode is indicated to the left of the login/logout button on the top right corner of the main application window.

Offline mode will be disabled automatically when internet connectivity is restored.

You can also manually force **Offline mode** by enabling the setting on the application preferences."

In the Anaconda Navigator Preferences dialog, users may select "Enable offline mode" to enter offline mode even if internet access is available.

Using Navigator in offline mode is equivalent to using the command line conda commands `create`, `install`, `remove`, and `update` with the flag `--offline` so that conda does not connect to the internet.

Icons

By default Navigator includes icons linking to the GitHub, YouTube, and Twitter pages of Anaconda Inc. Enterprise users may change or remove these links by editing the configuration file `HOME_DIR/.anaconda/navigator/anaconda-navigator-config.yml`. The file uses key-value pairs of the form `key: value`, such as `github_url: https://github.com`. Each of the three values `github_url` `youtube_url` and `twitter_url` may be set to any URL, or set to the value `null`. If the value is `null` Navigator does not display that icon.

3.7.4 Setting the database password

1. Set a password in postgres.
   (a) Connect to postgres pod: `kubectl exec -it postgres-<id> /bin/sh`
   (b) Connect to database with psql: `psql -h localhost -U postgres`
   (c) Set the password: `ALTER USER user_name WITH PASSWORD 'new_password';`
2. Open the Anaconda Enterprise Operations Center and navigate to the platform configuration page.
3. Enter the password at `db.password`.
4. Restart all the service pods: `kubectl get pods | grep ap- | cut -d' ' -f1 | xargs kubectl delete pods`

3.7.5 Adding a GPU resource profile

If your installation has any nodes with GPUs, add a GPU resource profile so users can request to use them.

1. Edit the configuration to *add a GPU Resource Profile*
2. Restart all the service pods: `kubectl get pods | grep ap- | cut -d' ' -f1 | xargs kubectl delete pods`

CHAPTER 4

Frequently asked questions

4.1 General

When was the general availability release of Anaconda Enterprise v5?

Our GA release was August 31, 2017 (version 5.0.3). Our most recent version was released July 27th, 2018 (version 5.2).

Which notebooks or editors does Anaconda Enterprise support?

Anaconda Enterprise supports the use of Jupyter Notebooks and JupyterLab, which are the most popular integrated data science environments for working with Python and R notebooks.

Can I deploy multiple data science applications to Anaconda Enterprise?

Yes, you can deploy multiple data science applications and languages across an Anaconda Enterprise cluster. Each data science application runs in a secure and isolated environment with all of the dependencies from Anaconda that it requires.

A single node can run multiple applications based on the amount of compute resources (CPU and RAM) available on a given node. Anaconda Enterprise handles all of the resource allocation and application scheduling for you.

Does Anaconda Enterprise support high availability deployments?

Partially. Some of the Anaconda Enterprise services and user-deployed apps will be automatically configured when installed to three or more nodes. Anaconda Enterprise provides several automatic mechanisms for fault tolerance and service continuity, including automatic restarts, health checks, and service migration.

For more information, see *Fault tolerance in Anaconda Enterprise*.

Which identity management and authentication protocols does Anaconda Enterprise support?

- LDAP / AD
- SAML
- Kerberos
- *Identity Brokering*

Does Anaconda Enterprise support two-factor authentication (including one-time passwords)?

Yes, Anaconda Enterprise supports single sign-on (SSO) and two-factor authentication (2FA) using FreeOTP, Google Authenticator or Google Authenticator compatible 2FA.

You can configure one-time password policies in Anaconda Enterprise by navigating to the authentication center and clicking on Authentication and then OTP Policy.

4.2 System requirements

What operating systems are supported for Anaconda Enterprise?

Please see *operating system requirements*.

NOTE: Linux distributions other than those listed in the documentation can be supported on request.

What are the minimum system requirements for Anaconda Enterprise nodes?

Please see *system requirements*.

Which browsers are supported for Anaconda Enterprise?

Please see *browser requirements*.

Does Anaconda Enterprise come with a version control system?

Yes, Anaconda Enterprise includes an internal git server, which allows users to save and commit versions of their projects.

Can Anaconda Enterprise integrate with my own Git server?

Yes, as described in *Connecting to an external Git repository*.

4.3 Installation

How do I install Anaconda Enterprise?

The Anaconda Enterprise installer is a single tarball that includes Docker, Kubernetes, system dependencies, and all of the components and images necessary to run Anaconda Enterprise. The system administrator runs one command on each node.

Can Anaconda Enterprise be installed on-premises?

Yes, including airgapped environments.

Can Anaconda Enterprise be installed on cloud environments?

Yes, including Amazon AWS, Microsoft Azure, and Google Cloud Platform.

Does Anaconda Enterprise support air gapped (off-line) environments?

Yes, the Anaconda Enterprise installer includes Docker, Kubernetes, system dependencies, and all of the components and images necessary to run Anaconda Enterprise on-premises or on a private cloud, with or without internet connectivity. We can deliver the installer to you on a USB drive.

Can I build Docker images for the install of Anaconda Enterprise?

No. The installation of Anaconda Enterprise is supported only by using the single-file installer. The Anaconda Enterprise installer includes Docker, Kubernetes, system dependencies, and all of the components and images necessary for Anaconda Enterprise.

Can I install Anaconda Enterprise on my own instance of Kubernetes?

No. The Anaconda Enterprise installer already includes Kubernetes.

Can I get the AE installer packaged as a virtual machine (VM), Amazon Machine Image (AMI) or other installation package?

No. The installation of Anaconda Enterprise is supported only by using the single-file installer.

Which ports are externally accessible from Anaconda Enterprise?

Please see *network requirements*.

Can I use Anaconda Enterprise to connect to my Hadoop/Spark cluster?

Yes. Anaconda Enterprise supports connectivity from notebooks to local or remote Spark clusters by using the Sparkmagic client and a Livy REST API server. Anaconda Enterprise provides Sparkmagic, which inlcudes Spark, PySpark, and SparkR notebook kernels for deployment.

How can I manage Anaconda packages on my Hadoop/Spark cluster?

An administrator can generate custom Anaconda parcels for Cloudera CDH or custom Anaconda management packs for Hortonworks HDP using Anaconda Enterprise. A data scientist can use these Anaconda libraries from a notebook as part of a Spark job.

On how many nodes can I install Anaconda Enterprise?

You can install Anaconda Enterprise in the following configurations during the initial installation:

- One node (one master node)
- Two nodes (one master node, one worker node)
- Three nodes (one master node, two worker nodes)
- Four nodes (one master node, three worker nodes)

After the initial installation, you can add or remove worker nodes from the Anaconda Enterprise cluster at any time.

One node serves as the master node and writes storage to disk, and the other nodes serve as worker nodes. Anaconda Enterprise services and user-deployed applications run seamlessly on the master and worker nodes.

Can I generate certificates manually?

Yes, if automatic TLS/SSL certificate generation fails for any reason, you can generate the certificates manually. Follow these steps:

1. Generate self-signed temporary certificates. On the master node, run:

   ```
   cd path/to/Anaconda/Enterprise/unpacked/installer
   cd DIY-SSL-CA
   bash create_noprompt.sh DESIRED_FQDN
   cp out/DESIRED_FQDN/secret.yaml /var/lib/gravity/planet/share/secrets.yaml
   ```

   Replace `DESIRED_FQDN` with the fully-qualified domain of the cluster to which you are installing Anaconda Enterprise.

   Saving this file as `/var/lib/gravity/planet/share/secrets.yaml` on the Anaconda Enterprise master node makes it accessible as `/ext/share/secrets.yaml` within the Anaconda Enterprise environment which can be accessed with the command `sudo gravity enter`.

2. Update the `certs` secret

   Replace the built-in `certs` secret with the contents of `secrets.yaml`. Enter the Anaconda Enterprise environment and run these commands:

```
$ kubectl delete secrets certs
secret "certs" deleted
$ kubectl create -f /ext/share/secrets.yaml
secret "certs" created
```

4.4 GPU Support

How can I make GPUs available to my team of data scientists?

If your data science team plans to use version 5.2 of the Anaconda Enterprise AI enablement platform, here are a few approaches to consider when planning your GPU cluster:

- *Build a dedicated GPU-only cluster.*

  If GPUs will be used by specific teams only, creating a separate cluster allows you to more carefully control GPU access.

- *Build a heterogeneous cluster.*

  Not all projects require GPUs, so a cluster containing a mix of worker nodes—with and without GPUs—can serve a variety of use cases in a cost-effective way.

- *Add GPU nodes to an existing cluster.*

  If your team's resource requirements aren't clearly defined, you can start with a CPU-only cluster, and add GPU nodes to create a heterogeneous cluster when the need arises.

Anaconda Enterprise supports heterogeneous clusters by allowing you to create different "resource profiles" for projects. Each resource profile describes the number of CPU cores, the amount of memory, and the number of GPUs the project needs. Administrators typically will create "Regular", "Large", and "Large + GPU" resource profiles for users to select from when running their project. If a project requires a GPU, AE will run it on only those cluster nodes with an available GPU.

What software is GPU accelerated?

Anaconda provides a number of GPU-accelerated packages for data science. For deep learning, these include:

- Keras (`keras-gpu`)
- TensorFlow (`tensorflow-gpu`)
- Caffe (`caffe-gpu`)
- PyTorch (`pytorch`)

For boosted decision tree models:

- XGBoost (`py-xgboost-gpu`)

For more general array programming, custom algorithm development, and simulations:

- CuPy (`cupy`)
- Numba (`numba`)

NOTE: Unless a package has been specifically optimized for GPUs (by the authors) and built by Anaconda with GPU support, it will not be GPU-accelerated, even if the hardware is present.

What hardware does each of my cluster nodes require?

Anaconda recommends installing Anaconda Enterprise in a cluster configuration. Each installation should have an odd number of master nodes, and we recommend at least one worker node. The master node runs all Anaconda Enterprise core services and does not need a GPU.

Using EC2 instances, a *minimal configuration* is one master node running on a `m4.4xlarge` instance and one GPU worker node running on a `p3.2xlarge` instance. More users will require more worker nodes—and possibly a mix of CPU and GPU worker nodes.

See *Installation requirements* for the baseline hardware requirements for Anaconda Enterprise.

How many GPUs does my cluster need?

A best practice for machine learning is for each user to have exclusive use of their GPU(s) while their project is running. This ensures they have sufficient GPU memory available for training, and provides more consistent performance.

When an Anaconda Enterprise user launches a notebook session or deployment that requires GPUs, those resources are reserved for as long as the project is running. When the notebook session or deployment is stopped, the GPUs are returned to the available pool for another user to claim.

The number of GPUs required in the cluster can therefore be determined by the number of concurrently running notebook sessions and deployments that are expected. Adding nodes to an Anaconda Enterprise cluster is straightforward, so organizations can start with a conservative number of GPUs and grow as demand increases.

To get more out of your GPU resources, Anaconda Enterprise supports scheduling and running unattended jobs. This enables you to execute periodic retraining tasks—or other resource-intensive tasks—after regular business hours, or at times GPUs would otherwise be idle.

What kind of GPUs should I use?

Although the Anaconda Distribution supports a wide range of NVIDIA GPUs, enterprise deployments for data science teams developing models should use one of the following GPUs:

- Tesla K80
- Tesla P100
- Tesla V100

The K80 is the oldest option, and only makes sense for budget cloud deployments (see below). We recommend using the V100 or P100 for new on-premise installations, as these GPUs are significantly faster—and therefore better for deep learning.

Can I mix GPU models in one cluster?

Kubernetes cannot currently distinguish between different GPU models in the same cluster node, so Anaconda Enterprise requires all GPU-enabled nodes *within a given cluster* to have the same GPU model (for example, all Tesla V100). Different clusters (e.g., "production" and "development") can use different GPU models, of course.

Can I use cloud GPUs?

Yes, Anaconda Enterprise 5.2 can be installed on cloud VMs with GPU support. Amazon Web Services (AWS), Google Cloud Platform, and Microsoft Azure all offer Tesla GPU options.

4.5 Anaconda Project

What operating systems and Python versions are supported for Anaconda Project?

Anaconda Project supports Windows, macOS and Linux, and tracks the latest Anaconda releases with Python 2.7, 3.5 and 3.6.

How is encapsulation with Anaconda Project different from creating a workspace or project in Spyder, PyCharm, or other IDEs?

A workspace or project in an IDE is a directory of files on your desktop. Anaconda Project encapsulates those files, but also includes additional parameters to describe how to run a project with its dependencies. Anaconda Project is portable and allows users to run, share, and deploy applications across different operating systems.

What types of projects can I deploy?

Anaconda Project is very flexible and can deploy many types of projects with conda or pip dependencies. Deployable projects include:

- Notebooks (Python and R)
- Bokeh applications and dashboards
- REST APIs in Python and R (including machine learning scoring and predictions)
- Python and R scripts
- Third-party apps, web frameworks, and visualization tools such as Tensorboard, Flask, Falcon, deck.gl, plot.ly Dash, and more.

Any generic Python and R script or webapp can be configured to serve on port 8086, which will show the app in Anaconda Enterprise when deployed.

Does Anaconda Enterprise include Docker images for my data science projects?

Anaconda Enterprise includes data science application images for the editor and deployments. You can install additional packages in either environment using Anaconda Project. Anaconda Project includes the information required to reproduce the project environment with Anaconda, including Python, R, or any other conda package or pip dependencies.

4.6 Notebooks

Are the deployed, self-service notebooks read-only?

Yes, the deployed versions of self-service notebooks are read-only, but they can be executed by collaborators or viewers. Owners of the project that contain the notebooks can edit the notebook and deploy (or re-deploy) them.

What happens when other people run the notebook? Does it overwrite any file, if notebook is writing to a file?

A deployed, self-service notebook is read-only but can be executed by other collaborators or viewers. If multiple users are running a notebook that writes to a file, the file will be overwritten unless the notebook is configured to write data based on a username or other environment variable.

Can I define environment variables as part of my data science project?

Yes, Anaconda Project supports environment variables that can be defined when deploying a data science application. Only project collaborators can view or edit environment variables, and they cannot be accessed by viewers.

How are Anaconda Project and Anaconda Enterprise available?

Anaconda Project is free and open-source. Anaconda Enterprise is a commercial product.

Where can I find example projects for Anaconda Enterprise?

Sample projects are included as part of the Anaconda Enterprise installation, which include sample workflows and notebooks for Python and R such as financial modeling, natural language processing, machine learning models with REST APIs, interactive Bokeh applications and dashboards, image classification, and more.

The sample projects include examples with visualization tools (Bokeh, deck.gl), pandas, scipy, Shiny, Tensorflow, Tensorboard, xgboost, and many other libraries. Users can save the sample projects to their Anaconda Enterprise account or download the sample projects to their local machine.

Does Anaconda Enterprise support batch scoring with REST APIs?

Yes, Anaconda Enterprise can be used to deploy machine learning models with REST APIs (including Python and R) that can be queried for batch scoring workflows. The REST APIs can be made available to other users and accessed with an API token.

Does Anaconda Enterprise provide tools to help define and implement REST APIs?

Yes, a data scientist can basically create a model without much work for the API development. Anaconda Enterprise includes an API wrapper for Python frameworks that builds on top of existing web frameworks in Anaconda, making it easy to expose your existing data science models with minimal code. You can also deploy REST APIs using existing API frameworks for Python and R.

4.7 Help and training

Do you offer support for Anaconda Enterprise?

Yes, we offer full support with Anaconda Enterprise.

Do you offer training for Anaconda Enterprise?

Yes, we offer product training for collaborative, end-to-end data science workflows with Anaconda Enterprise.

Do you have a question not answered here?

Please contact us for more information.

CHAPTER 5

Release note history

5.1 Anaconda Enterprise 5.2.1

Released: August 30, 2018

User-facing changes

- Fixed issue with loading spinner appearing on top of notebook sessions
- Fixed issue related to missing projects and copying sample projects when upgrading from AE 5.1.x
- Improved visual feedback when loading notebook sessions/deployments and performing actions such as creating/copying projects

5.2 Anaconda Enterprise 5.2.0

Released: July 27, 2018

Administrator-facing changes

- New administrative console with workflows for managing channels and packages, creating installers, and other distinct administrator tasks
- Added ability to mirror pip packages from PyPI repository
- Added ability to define custom hardware resource profiles based on CPU, RAM, and GPU for user sessions and deployments
- Added support for GPU worker nodes that can be defined in resource profiles
- Added ability to explicitly install different types of master nodes for high availability
- Added ability to specify NFS file shares that users can access within sessions and deployments
- Significantly reduced the amount of time required for backup/restore operations

- Added channel and package management tasks to UI, including downloading/uploading packages, creating/sharing channels, and more
- Anaconda Livy is now included in the Anaconda Enterprise installer to enable remote Spark connectivity
- All network traffic for services is now routed on standard HTTPS port 443, which reduces the number of external ports that need to be configured and accessed by end users
- Notebook/editor sessions are now accessed via subdomains for security and isolation
- Reworked documentation for administrator workflows, including managing cluster resources, configuring authentication, generating custom installers, and more
- Reduced verbosity of console output from anaconda-enterprise-cli
- Suppressed superfluous database errors/warnings

User-facing changes

- Added support for selecting GPU hardware in project sessions and deployments, to accelerate model training and other computations with GPU-enabled packages
- Added ability to select custom hardware resource profiles based on CPU, RAM, and GPU for individual sessions and deployments
- Added support for scheduled and batch jobs, which can be used for recurring tasks such as model training or ETL pipelines
- Added support for connecting to external Git repositories in a project session or deployment using account-wide credentials (SSH keys or API tokens)
- New, responsive user interface, redesigned for data science workflows
- Added ability to share deployments with unauthenticated users outside of Anaconda Enterprise
- Changed the default editor in project sessions to Jupyter Notebooks (formerly JupyterLab)
- Added ability to specify default editor on a per-project basis, including Jupyter Notebooks and JupyterLab
- Added ability to work with data in mounted NFS file shares within sessions and deployments
- Added ability to export/download projects from Anaconda Enterprise to local machine
- Added package and channel management tasks to UI, including uploading/downloading packages, creating/sharing channels, and more
- Reworked documentation for data science workflows, including working with projects/deployments/packages, using project templates, machine learning workflows, and more
- Added ability to use plotting/Javascript libraries in JupyterLab
- Added ability to force delete a project with running sessions, shared collaborators, etc.
- Improved messaging when a session or deployment cannot be scheduled due to limited cluster resources
- The last modified date/time for projects now accounts for commits to the project
- Unique names are now enforced for projects and deployments
- Fixed bug in which project creator role was not being enforced

Backend improvements (non-visible changes)

- Updated to Kubernetes 1.9.6
- Added RHEL/CentOS 7.5 to supported platforms
- Added support for SELinux passive mode

- Anaconda Enterprise now uses the Helm package manager to manage and upgrade releases
- New version (v2) of backend APIs with more comprehensive information around projects, deployments, packages, channels, credentials and more
- Fixed various bugs related to custom Anaconda installer builds
- Fixed issue with `kube-router` and a `CrashLoopBackOff` error

5.3 Anaconda Enterprise 5.1.3

Released: June 4, 2018

Backend improvements (non-visible changes)

- Fixed issue when generating custom Anaconda installers that contain packages with duplicate files
- Fixed multiple issues related to memory errors, file size limits, and network transfer limits that affected the generation of large custom Anaconda installers
- Improved logging when generating custom Anaconda installers

5.4 Anaconda Enterprise 5.1.2

Released: March 16, 2018

Administrator-facing changes

- Fixed issue with image/version tags when upgrading AE

Backend improvements (non-visible changes)

- Updated to Kubernetes 1.7.14

5.5 Anaconda Enterprise 5.1.1

Released: March 12, 2018

Administrator-facing changes

- Ability to specify custom UID for service account at install-time (default UID: 1000)
- Added pre-flight checks for kernel modules, kernel settings, and filesystem options when installing or adding nodes
- Improved initial startup time of project creation, sessions, and deployments after installation. Note that all services will be in the `ContainerCreating` state for 5 to 10 minutes while all AE images are being pre-pulled, after which the AE user interface will become available.
- Improved upgrade process to automatically handle upgrading AE core services
- Improved consistency between GUI- and CLI-based installation paths
- Improved security and isolation between internal database from user sessions and deployments
- Added capability to configure a custom trust store and LDAPS certificate validation
- Simplified installer packaging using a single tarball and consistent naming

- Updated documentation for system requirements, including XFS filesystem requirements and kernel modules/settings
- Updated documentation for mirroring packages from channels
- Added documentation for configuring AE to point to online Anaconda repositories
- Added documentation for securing the internal database
- Added documentation for configuring RBAC, role mapping, and access control
- Added documentation for LDAP federation and identity management
- Improved documentation for backup/restore process
- Fixed issue when deleting related versions of custom Anaconda parcels
- Added command to remove channel permissions
- Fixed issue related to Ops Center user creation in post-install configuration
- Silenced warnings when using `verify_ssl` setting with `anaconda-enterprise-cli`
- Fixed issue related to default admin role (`ae-admin`)
- Fixed issue when generating TLS/SSL certificates with FQDNs greater than 64 characters
- Fixed issue when using special characters with AE Ops Center accounts/passwords
- Fixed bug related to Administrator Console link in menu

User-facing changes

- Improvements to collaborative workflow: Added notification when collaborators make changes to a project, ability to pull changes into a project, and ability to resolve conflicting changes when saving or pulling changes into a project.
- Additional documentation and examples for connecting to remote data and compute sources: Spark, Hive, Impala, and HDFS
- Optimized startup time for Spark and SAS project templates
- Improved initial startup time of project creation, sessions, and deployments by pre-pulling images after installation.
- Increased upload limit of projects from 100 MB to 1 GB
- Added capability to `sudo yum install` system packages from within project sessions
- Fixed issue when uploading projects that caused them to fail during partial import
- Fixed R kernel in R project template
- Fixed issue when loading `sparklyr` in Spark Project
- Fixed issue related to displaying kernel names and Spark project icons
- Improved performance when rendering large number of projects, packages, etc.
- Improved rendering of long version names in environments and projects
- Render full names when sharing projects and deployments with collaborators
- Fixed issue when sorting collaborators and package versions
- Fixed issue when saving new environments
- Fixed issues when viewing installer logs in IE 11 and Safari

5.6 Anaconda Enterprise 5.1.0

Released: January 19, 2018

Administrator-facing changes

- New post-installation administration GUI with automated configuration of TLS/SSL certificates, administrator account, and DNS/FQDN settings; significantly reduces manual steps required during post-installation configuration process
- New functionality for administrators to generate custom Anaconda installers, parcels for Cloudera CDH, and management packs for Hortonworks HDP
- Improved backup and restore process with included scripts
- Switched from groups to roles for role-based access control (RBAC) for Administrator and superuser access to AE services
- Clarified system requirements related to system modules and IOPS in documentation
- Added ability to specify fractional CPUs/cores in global container resource limits
- Fixed consistency of TLS/SSL certificate names in configuration and during creation of self-signed certificates
- Changed use of `verify_ssl` to `ssl_verify` throughout AE CLI for consistency with `conda`
- Fixed configuration issue with licenses, including field names and online/offline licensing documentation

User changes

- Updated default project environments to Anaconda Distribution 5.0.1
- Improved configuration and documentation on using Sparkmagic and Livy with Kerberos to connect to remote Spark clusters
- Fixed R environment used in sample projects and project template
- Fixed UI rendering issue on package detail view of channels, downloads, and versions
- Fix multiple browser compatiblity issues with Microsoft Edge and Internet Explorer 11
- Fixed multiple UI issues with Anaconda Project JupyterLab extension

Backend improvements (non-visible changes)

- Updated to Kubernetes 1.7.12
- Updated to conda 4.3.32
- Added SUSE 12 SP2/SP3, and RHEL/CentOS 7.4 to supported platform matrix
- Implemented TLS 1.2 as default TLS protocol; added support for configurable TLS protocol versions and ciphers
- Fixed default superuser roles for repository service, which is used for initial/internal package configuration step
- Implemented secure flag attribute on all session cookies containing session tokens
- Fixed issue during upgrade process that failed to vendor updated images
- Fixed `DiskNodeUnderPressure` and cluster stability issues
- Fixed Quality of Service (QoS) issue with core AE services on under-resourced nodes
- Fixed issue when using access token instead of ID token when fetching roles from authentication service
- Fixed issue with authentication proxy and session cookies

Known issues

- IE 11 compatibility issue when using Bokeh in notebooks (including sample projects)
- IE 11 compatibility issue when downloading custom installers

5.7 Anaconda Enterprise 5.0.6

Released: November 9, 2017

5.8 Anaconda Enterprise 5.0.5

Released: November 7, 2017

5.9 Anaconda Enterprise 5.0.4

Released: September 12, 2017

5.10 Anaconda Enterprise 5.0.3

Released: August 31, 2017 (General Availability Release)

5.11 Anaconda Enterprise 5.0.2

Released: August 15, 2017 (Early Adopter Release)

5.12 Anaconda Enterprise 5.0.1

Released: March 8, 2017 (Early Adopter Release)

Features:

- Simplified, one-click deployment of data science projects and deployments, including live Python and R notebooks, interactive data visualizations and REST APIs.
- End-to-end secure workflows with SSL/TLS encryption.
- Seamlessly managed scalability of the entire platform
- Industry-grade productionization, encapsulation, and containerization of data science projects and applications.

CHAPTER 6

Troubleshooting known issues

6.1 Create and Installer buttons are not visible on Channels page

When the Channels page is viewed initially, the Create and Installers buttons are not visible on the top right section of the screen. This prevents the user from creating channels or viewing a list of installers.

Workaround

To make the Create and Installer buttons visible on the Channels page, perform one of the following steps:

- Click on the top-level Channels navigation link again when viewing the Channels page
- Click on a specific channel to view its detail page, then return to the Channels page

Affected versions

5.2.1

6.2 Updating a package from the Anaconda metapackage

When updating a package dependency of a project, if that dependency is part of the Anaconda metapackage the package will be installed once but a subsequent `anaconda-project` call will uninstall the upgraded package.

Workaround

When updating a package dependency remove the `anaconda` metapackage from the list of dependencies at the same time add the new version of the dependency that you want to update.

Affected versions

5.1.0, 5.1.1, 5.1.2, 5.1.3

6.3 File size limit when uploading files

Unable to upload new files inside of a project that are larger than the current restrictions:

- The limit of file uploads in JupyterLab is 15 MB

Affected versions

5.1.0, 5.1.1, 5.1.2, 5.1.3

6.4 IE 11 compatibility issue when using Bokeh in projects (including sample projects)

Bokeh plots and applications have had a number of issues with Internet Explorer 11, which typically result in the user seeing a blank screen.

Workaround

Upgrade to the latest version of Bokeh available. On Anaconda 4.4 the latest is 0.12.7. On Anaconda 5.0 the latest version of Bokeh is 0.12.13. If you are still having issues, consult the Bokeh team or support.

Affected versions

5.1.0, 5.1.1, 5.1.2, 5.1.3

6.5 IE 11 compatibility issue when downloading custom Anaconda installers

Unable to download a custom Anaconda installer from the browser when using Internet Explorer 11 on Windows 7. Attempting to download a custom installer with this setup will result in an error that "This page can't be displayed".

Workaround

Custom installers can be downloaded by refreshing the page with the error message, clicking the "Fix Connection Error" button, or using a different browser.

Affected versions

5.1.0, 5.1.1, 5.1.2, 5.1.3

6.6 Project names over 40 characters may prevent JupyterLab launch

If a project name is more than 40 characters long, launching the project in JupyterLab may fail.

Workaround

Rename the project to a name less than 40 characters long and launch the project in JupyterLab again.

Affected versions

5.1.1, 5.1.2, 5.1.3

6.7 Long-running jobs may falsely report failure

If a job (such as an installer, parcel, or management pack build) runs for more than 10 minutes, the UI may falsely report that the job has failed. The apparent job failure occurs because the session/access token in the UI has expired.

However, the job will continue to run in the background, the job run history will indicate a status of "running job" or "finished job", and the job logs will be accessible.

Workaround

To prevent false reports of failed jobs from occurring in the UI, you can extend the access token lifespan (default: 10 minutes).

To extend the access token lifespan, log in to the Anaconda Enterprise Authentication Center, navigate to Realm Settings > Tokens, then increase the Access Token Lifespan to be at least as long as the jobs being run (e.g., 30 minutes).

Affected versions

5.1.0, 5.1.1, 5.1.2, 5.1.3

6.8 New Notebook not found on IE11

On Internet Explorer 11, creating a new Notebook in a Classic Notebook editing session may produce the error "404: Not Found". This is an artifact of the way that Internet Explorer 11 locates files.

Workaround

If you see this error, click "Back to project", then click "Return to Session". This refreshes the file list and allows IE11 to find the file. You should see the new notebook in the file list. Click on it to open the notebook.

Affected versions

5.0.4, 5.0.5

6.9 Disk pressure errors on AWS

If your Anaconda Enterprise instance is on Amazon Web Services (AWS), overloading the system with reads and writes to the directory `/opt/anaconda` can cause disk pressure errors, which can cause these problems:

- Slow project starts.
- Project failures.
- Slow deployment completions.
- Deployment failures.

Workaround

If you see these problems, check the logs for disk pressure errors.

If you see disk pressure errors, you can add disks to the instance by adding another Elastic Block Store (EBS) volume.

1. Open the AWS console and add a new EBS volume provisioned to 3000 IOPS. A typical disk size is 500 GB.
2. Attach the volume to your AE 5 master.
3. To find your new disk's name run `fdisk -l`. Our example disk's name is `/dev/nvme1n1`. In the rest of the commands on this page, replace `/dev/nvme1n1` with your disk's name.

4. Format the new disk: `fdisk /dev/nvme1n1`

   To create a new partition, at the first prompt press `n` and then the return key.

   Accept all default settings.

   To write the changes, press `w` and then the return key. This will take a few minutes.

5. To find your new partition's name, examine the output of the last command. If the name is not there, run `fdisk -l` again to find it.

   Our example partition's name is `/dev/nvme1n1p1`. In the rest of the commands on this page, replace `/dev/nvme1n1p1` with your partition's name.

6. Make a file system on the new partition: `mkfs /dev/nvme1n1p1`
7. Make a temporary directory to capture the contents of `/opt/anaconda`: `mkdir /opt/aetmp`
8. Mount the new partition to `/opt/aetmp`: `mount /dev/nvme1n1p1 /opt/aetmp`
9. Shut down the Kubernetes system.

   Find the gravity services: `systemctl list-units | grep gravity`

   You will see output like this:

```
# systemctl list-units | grep gravity
gravity__gravitational.io__planet-master__0.1.87-1714.service          loaded␣
↪active running
    Auto-generated service for the gravitational.io/planet-master:0.1.87-1714␣
↪package
gravity__gravitational.io__teleport__2.3.5.service                      loaded␣
↪active running
    Auto-generated service for the gravitational.io/teleport:2.3.5 package
```

   Shut down the `teleport` service: `systemctl stop gravity__gravitational.io__teleport__2.3.5.service`

   Shut down the `planet-master` service: `systemctl stop gravity__gravitational.io__planet-master__0.1.87-1714.service`

10. Copy everything from `/opt/anaconda` to `/opt/aetmp`: `rsync -vpoa /opt/anaconda/* /opt/aetmp`
11. Include the new disk at the `/opt/anaconda` mount point by adding this line to your file systems table at `/etc/fstab`:

```
/dev/nvme1n1p1   /opt/anaconda   ext4    defaults        0 0
```

    Use mixed spaces and tabs in this pattern: `/dev/nvme1n1p1<tab>/opt/anaconda<tab>ext4<tab>defaults<tab>0<space>0`

12. Move the old `/opt/anaconda` out of the way to `/opt/anaconda-old`: `mv /opt/anaconda /opt/anaconda-old`

    If you're certain the `rsync` was successful, you may instead delete `/opt/anaconda`: `rm -r /opt/anaconda`

13. Unmount the new disk from the `/opt/aetmp` mount point: `umount /opt/aetmp`
14. Make a new `/opt/anaconda` directory: `mkdir /opt/anaconda`
15. Mount all the disks defined in `fstab`: `mount -a`
16. Restart the gravity services:

```
systemctl start gravity__gravitational.io__planet-master__0.1.87-1714.service
systemctl start gravity__gravitational.io__teleport__2.3.5.service
```

6.10 General diagnostic and troubleshooting steps

Entering Anaconda Enterprise environment

To enter the Anaconda Enterprise environment and gain access to `kubectl` and other commands within Anaconda Enterprise, use the command:

```
sudo gravity enter
```

Moving files and data

Occasionally you may need to move files and data from the host machine to the Anaconda Enterprise environment. If so, there are two *shared mounts* to pass data back and forth between the two environments:

- host: `/opt/anaconda/` -> AE environment: `/opt/anaconda/`
- host: `/var/lib/gravity/planet/share` -> AE environment: `/ext/share`

If data is written to either of the locations, that data will be available on both the host machine and within the Anaconda Enterprise environment

Debugging

AWS Traffic needs to handle the public IPs and ports. You should either use a canonical security group with the proper ports opened or manually add the specific ports listed in *Network Requirements*.

"Cannot continue" error during install

This bug is caused by a previous failure of a kernel module check or other preflight check and subsequent attempt to reinstall.

Stop the install, make sure the preflight check failure is resolved, and restart the install again.

Failed installations

If an installation fails, you can view the failed logs as part of the support bundle in the failed installation UI.

After executing `sudo gravity enter` you can check `/var/log/messages` to troubleshoot a failed installation or these types of errors.

After executing `sudo gravity enter` you can run `journalctl` to look at logs to troubleshoot a failed installation or these types of errors:

```
journalctl -u gravity-23423lkqjfefqpfh2.service
```

NOTE: Replace `gravity-23423lkqjfefqpfh2.service` with the name of your gravity service.

You may see messages in `/var/log/messages` related to errors such as "etcd cluster is misconfigured" and "etcd has no leader" from one of the installation jobs, particularly `gravity-site`. This usually indicates that `etcd` needs more compute power, needs more space or is on a slow disk.

Anaconda Enterprise is very sensitive to disk latency, so we usually recommend using a better disk for `/var/lib/gravity` on target machines and/or putting `etcd` data on a separate disk. For example, you can mount `etcd` under `/var/lib/gravity/planet/etcd` on the hosts.

After a failed installation, you can *uninstall Anaconda Enterprise* and start over with a fresh installation.

Failed on pulling gravitational/rbac

If the node refuses to install and fails on pulling gravitational/rbac, create a new directory `TMPDIR` before installing and provide *write access* to user 1000.

Problems during air gap project migration

The command anaconda-project lock over-specifies the channel list resulting in a conda bug where it adds defaults from the internet to the list of channels.

Solution:

Add to the .condarc: "default_channels". This way, when conda adds "defaults" to the command it is adding the internal repo server and not the repo.continuum.io URLs.

EXAMPLE:

```
default_channels:
- anaconda
channels:
  - our-internal
  - out-partners
  - rdkit
  - bioconda
  - defaults
  - r-channel
  - conda-forge
channel_alias: https://:8086/conda
auto_update_conda: false
ssl_verify: /etc/ssl/certs/ca.2048.cer
```

Problems during post-install or post-upgrade steps

Post-install and post-upgrade steps run as Kubernetes jobs. When they complete running the pods used to run them are **not** removed. These and other stopped pods can be found using:

```
kubectl get pods -a
```

The logs in each of these three pods will be helpful for diagnosing issues in the following steps:

Pod	Issues in this step
`ae-wagonwheel`	post-install UI
`install`	installation step
`postupdate`	post-update steps

Problems after post-install configuration

In order to reinitialize the post-install configuration UI, which can come in handy for regenerating temporary (self-signed) SSL certificates or reconfiguring the platform based on your domain name, you must re-create and re-expose the service on a new port.

First, recreate the ap-wagonwheel deployment:

```
kubectl create -f /var/lib/gravity/site/packages/unpacked/gravitational.io/
↪AnacondaEnterprise/5.X.X/resources/wagonwheel.yaml -n kube-system
```

NOTE: Replace 5.X.X with your actual version number.

Then execute `sudo gravity enter` and run:

```
kubectl get deploy -n kube-system
```

to check the services running in the system namespace. One of these should be `ae-wagonwheel`, the post-install configuration UI. To make this visible to the outside world, run:

```
kubectl expose deploy ae-wagonwheel --port=8000 --type=NodePort --name=post-install -
↪n kube-system
```

This will run the UI on a new port, allocated by Kubernetes, under the name `post-install`. Run:

```
kubectl get svc -n kube-system | grep post-install
```

to find out which port it is listening under, then navigate to `http://<your domain>:<this port>` to see the post-install UI again.

LDAP error in ap-auth

```
[LDAP: error code 12 - Unavailable Critical Extension]; remaining name
'dc=acme, dc=com'
```

This error can be caused when pagination is turned on. Pagination is a server side extension and is not supported by some LDAP servers, notably the Sun Directory server.

CHAPTER 7

Glossary

Anaconda

Sometimes used as shorthand for the Anaconda Distribution, Anaconda, Inc. is the company behind Anaconda Distribution, conda, conda-build and Anaconda Enterprise.

Anaconda Cloud

A cloud package repository hosting service at https://www.anaconda.org. With a free account, you can publish packages you create to be used publicly. A paid subscription is required to designate packages as private, and restrict access to authorized users.

Anaconda Distribution

Open source respository of hundreds of popular data science packages, along with the conda package and virtual environment manager for Windows, Linux, and MacOS. Conda makes it quick and easy to install, run, and upgrade complex data science and machine learning environments like scikit-learn, TensorFlow, and SciPy.

Anaconda Enterprise

A software platform for developing, governing, and automating data science and AI pipelines from laptop to production. Enterprise enables collaboration between teams of thousands of data scientists running large-scale model deployments on high-performance production clusters.

Anaconda Navigator

A desktop graphical user interface (GUI) included in Anaconda Distribution that allows you to easily use and manage IDEs, conda packages, environments, channels and notebooks without the need to use the command line interface (CLI).

Anaconda project

An encapsulation of your data science assets to make them easily portable. Projects may include files, environment variables, runnable commands, services, packages, channels, environment specifications, scripts, and notebooks. Each project also includes an `anaconda-project.yml` configuration file to automate setup, so you can easily run and share it with others. You can create and configure projects from the Enterprise web interface or command-line interface.

Anaconda Repository

A public repository of over 100,000 professionally-built software packages maintained by Anaconda, Inc. Anyone can download packages from the Anaconda Repository. A subset of them are included in the Anaconda Distribution installer, and it can be mirrored—completely or partially—for use with Anaconda Enterprise.

Channel

A location in the repository where Anaconda Enterprise looks for packages. Enterprise Administrators and users can define channels, determine which packages are available in a channel, and restrict access to specific users, groups or roles.

Commit

Making a set of local changes permanent by copying them to the remote server. Anaconda Enterprise checks to see if your work will conflict with any commits that your colleagues have made on the same project, so the files will not be overwritten unless you so choose to do so.

Conda

An open source package and environment manager that makes it quick and easy to install, run, and upgrade complex data science and machine learning environments like scikit-learn, TensorFlow, and SciPy. Thousands of Python and R packages can be installed with conda on Windows, MacOS X, Linux and IBM Power.

Conda-build

A tool used to build Conda packages from recipes.

Conda environment

A superset of Python virtual environments, Conda environments make it easy to create projects with different versions of Python and avoid issues related to dependencies and version requirements. A conda environment maintains its own files, directories, and paths so that you can work with specific versions of libraries and/or Python itself without affecting other Python projects.

Conda package

A binary tarball file containing system-level libraries, Python/R modules, executable programs, or other components. Conda tracks dependencies between specific packages and platforms, making it simple to create operating system-specific environments using different combinations of packages.

Conda recipe

Instructions used to tell conda-build how to build a package.

Deployment

A deployed Anaconda project containing a Notebook, web app, dashboard or machine learning model (exposed via API). When you deploy a project, Anaconda Enterprise builds a container with all the required dependencies and runtime components—the libraries on which the project depends in order to run—and launches it with the security and access permissions defined by the user. This allows you to easily run and share the application with others.

Interactive data applications

Visualizations with sliders, drop-downs and other widgets that allow users to interact with them. Interactive data applications can drive new computations, update plots and connect to other programmatic functionality.

Interactive development environment (IDE)

A suite of software tools that combines everything a developer needs to write and test software. It typically includes a code editor, a compiler or interpreter, and a debugger that the developer accesses through a single graphical user interface (GUI). An IDE may be installed locally, or it may be included as part of one or more existing and compatible applications accessed through a web browser.

Jupyter

A popular open source IDE for building interactive Notebooks by the Jupyter Foundation.

JupyterHub

An open source system for hosting multiple Jupyter Notebooks in a centralized location.

JupyterLab

Jupyter Foundation's successor IDE to Jupyter, with flexible building blocks for interactive and collaborative computing. For Jupyter Notebook users, the interface for JupyterLab is familiar and still contains the notebook, file browser, text editor, terminal and outputs.

Jupyter Notebook

The default browser-based IDE available in Anaconda Enterprise. It combines the notebook, file browser, text editor, terminal and outputs.

Live notebooks

JupyterLab and Jupyter Notebooks are web-based IDE applications that allow you to create and share documents that contain live code in R or Python, equations, visualizations and explanatory text.

Package

Software files and information about the software—such as its name, description, and specific version—bundled into a file that can be installed and managed by a package manager. Packages can be encapsulated into Anaconda projects for easy portability.

Project templates

Contains all the base files and components to support a particular programming environment. For example, a Python Spark project template contains everything you need to write Python code that connects to Spark clusters. When creating a new project, you can select a template that contains a set of packages and their dependencies.

REST APIs

A common way to operationalize machine learning models is through REST APIs. REST APIs are web server endpoints, or callable URLs, which provide results based on a query. REST APIs allow developers to create applications that incorporate machine learning and prediction, without having to write models themselves.

Session

An open project, running in an editor or IDE.

Spark

A distributed SQL database and project of the Apache Foundation. While Spark has historically been tightly associated with Apache Hadoop and run on Hadoop clusters, recently the Spark project has sought to separate itself from Hadoop by releasing support for Spark on Kubernetes. The core data structure in Spark is the RDD (Resilient Distributed Dataset)—a collection of data types, distributed in redundant fashion across many systems. To improve performance, RDDs are cached in memory by default, but can also be written to disk for persistence. Spark Ignite is a project to offer Spark RDDs that can be shared in-memory across applications.