SaltStack Enterprise installation

Summary: This document contains instructions to download and install SaltStack Enterprise and packages for Windows.

SaltStack Enterprise version 6.2.0
Document version 2
Updated: March 17, 2020

Prerequisites

SaltStack Enterprise integrates seamlessly with a new or existing Salt installation. For information about installing Salt, see the Salt Installation Guide .

SaltStack Enterprise Server requirements

Set up a server to host SaltStack Enterprise. This server hosts the Enterprise API and Enterprise Console web components, and optionally a Salt Master.

Supported operating systems for SaltStack Enterprise:

The following operating systems are supported to host your SaltStack Enterprise server:

- Red Hat Enterprise Linux 7 (RHEL 7)/CentOS 7 (Cent7)
- Oracle Linux 7
- SUSE Linux Enterprise Server 12
- SUSE Linux Enterprise Server 15

> This list is for the SaltStack Enterprise server, not for the Salt Masters in your environment. See Supported Salt Versions for a list of supported Salt Master operating systems.

Supported Python versions for SaltStack Enterprise:

Python 3.5.3 or later is required on your SaltStack Enterprise server. Python 3.x packages are provided with the SaltStack Enterprise installation files.

SaltStack Enterprise can now run on Salt Masters running Python 3.6.

Supported Salt versions for SaltStack Enterprise:

If you plan to use SaltStack Comply with Windows servers, these Windows minions must run Salt 3000 or later.

SaltStack Enterprise Licensing

SaltStack Enterprise requires a license file to track minion usage and duration of contract.**The SaltStack Enterprise download contains a 14-day trial license. After 14 days the Enterprise API service no longer starts.**

Customers receive a license file with the Welcome letter from SaltStack Support. If you are a current customer and have not received a license file, or if you encounter any issues with the licensing process, please contact SaltStack support.

Before 14 days, your license file must be placed on your SaltStack Enterprise server at `/etc/raas/raas.license` for continued functionality.

Hardware recommendations for single-node installation

Requirements for installing the Salt Master, SaltStack Enterprise, and PostgreSQL on the same host.

> Single-node installation is not recommended for production grade systems.

Up to 500
8 CPU cores
16 GB RAM
At least 40 GB free disk space**

Used for minion return data. Increase according to your needs for data retention.

Hardware recommendations for multi-node installation

Multi-node is recommended for environments with more than 1000 minions.

Requirements for installing the Salt Master, SaltStack Enterprise, and PostgreSQL on separate hosts:

	1000 to 2500 minions	2500 to 5000 minions	Greater than 5000 minions
Salt Master node	4 CPU cores 8 GB RAM	8 CPU cores 16 GB RAM	Consider multiple masters
SaltStack Enterprise node	4 CPU cores 8 GB RAM	8 CPU cores 16 GB RAM	Create an additional SaltStack Enterprise node per 5000 minions, hosted behind your preferred load-balancing solution
PostgreSQL node	4 CPU cores 8 GB RAM At least 40 GB free disk space*	8 CPU cores 16 GB RAM At least 80 GB free disk space*	Increase PostgreSQL CPU cores and RAM, as indicated by performance
Redis node	2 CPU cores 4 GB RAM	4 CPU cores 8 GB RAM	Increase Redis CPU cores and RAM, as indicated by performance

* Used for minion return data. Increase according to your needs for data retention.

Database requirements

> SaltStack Enterprise requires a PostgreSQL 9.6 database, but PostgreSQL 12.1 is recommended.

Important notes regarding the SaltStack Enterprise database:

1. The database is supported on PostgreSQL only.
2. The SaltStack Enterprise installer can install and configure PostgreSQL on your SaltStack Enterprise server, however, you are responsible for ongoing maintenance, backups, and other administrative tasks for this database. See the PostgreSQL documentation for details on PostgreSQL database maintenance and administration.
3. For a PostgreSQL tuning guide, see Tuning your PostgreSQL Server for SaltStack Enterprise.

Supported Salt versions

SaltStack Enterprise can be used with the currently supported open source versions of Salt. See SaltStack Platform Support for a list.

Supported web browsers for SaltStack Enterprise Web Console

The latest versions of Google Chrome and Mozilla Firefox are supported.

Changes to your Salt environment

Installing SaltStack Enterprise makes the following changes to your Salt environment:

- Enterprise API backend services (file system, pillar store, and so on) take precedence over any other existing backends defined in your environment. You can continue to use all supported backend services, just be aware that files that exist in Enterprise Console will take precedence if they also exist in other file or pillar backends (See the Configuration topic in the SaltStack Enterprise help to learn how to change this behavior).
- Enterprise API replaces the Salt Master syndic component to provide Salt Minion aggregation and scale. Salt Syndic Masters are not compatible with the SaltStack Enterprise architecture. Instead, each root Salt Master should connect directly to the Enterprise API.

Existing Salt States, configuration settings, and Salt Minion connections are unchanged. No changes are required on the Salt Minion to use SaltStack Enterprise.

Tuning Processes on your SaltStack Enterprise Server

When the SaltStack Enterprise Service (*raas*) starts, it creates two types of processes:

- **Tornado processes** - allows connections from Salt Masters and web browsers
- **Celery processes** - background workers

By default, raas sets the count for each process type to half the number of CPU Cores.

In most cases this is optimal, as the raas host should be dedicated to this task.

If you need to deploy raas on a host that supports additional services, you can override the default behavior by adding the following to your `/etc/raas/raas` configuration file.

```
num_processes: 8
background_workers:
  concurrency: 8
```

The following guides might be helpful for tuning:

- SaltStack Enterprise Performance Configurations
- Background Worker Options .

Known issues

- If the Postgres DB is not set to use UTF-8, sorting will not be consistent across the application.
- Scheduled jobs display in the web console only if scheduled within the next 12 weeks.
- Job return numbers may differ from target numbers based on current key state and grain data.
- SaltStack Enterprise console is supported on Chrome version 72 and later, and FireFox version 63 and later.
- SaltStack SecOps requires Salt version *2018.3.3* or later for Linux or Unix minions, and *3000* or higher for Windows minions.
- It is recommended SaltStack SecOps assessments and remediations are run weekly or biweekly for target groups greater than 1,000 minions. If run more frequently, the results table will quickly consume all available disk space.
- Compound targeting with grains does not work if the grain filter value contains spaces. To use compound targeting with grains, do one of the following:
    - Use the `G@` glob matcher but replace spaces with wildcard characters `(*)` For example, change `G@osfinger:CentOS Linux-7` to `G@osfinger:CentOS*Linux-7`
    - Use the `P@` Perl-compatible regular expression (PCRE) matcher but replace spaces with `\s` to match whitespace. For example, change `G@osfinger:CentOS Linux-7` to `G@osfinger:CentOS\sLinux-7`
- In some cases, SecOps Compliance assessments results might show a negative count of minions returned.
- In the Web Console, SecOps Vulnerability does not show any minions included in a newly-created policy until you have run the first assessment.
- A Vulnerability policy's **Advisories** tab does not show vulnerabilities that have been remediated. To view remediated advisories, go to the policy's **Minions** tab, select a remediated minion, and then select the **Last Remediation** tab.
- Remediating vulnerabilities on a minion might not result in any changes to the minion if the OS has not yet provided updated packages required to remediate the vulnerability. In these cases, the remediation job returns successfully, but the vulnerabilities have not been remediated.
- Vulnerability scans run immediately following a fresh installation might fail. This happens because after the initial install, SaltStack Enterprise takes roughly 15-20 minutes to ingest vulnerability content. Once the ingestion process is complete, you can run vulnerability assessments and remediations successfully.

- When working with nested groups, the Authentication workspace does not accurately reflect when a nested (or child) group is enabled. By enabling a parent group, you also enable all child groups by default. However, in the workspace, the child groups do not appear to be enabled.
- When configuring a Directory Service connection for a forest structure, the`Auth Bind DN Filter` field must be left blank.
- Any groups you have removed from a connection are still visible in the Roles workspace, and can be selected, although they are inactive. This also applies to any removed users previously visible in the Roles workspace. Although you can select an inactive group or user, these users can't log in to SaltStack Enterprise.
- After upgrading SaltStack Enterprise (for example, from 6.0.1 to 6.1.0), you must clear your web browser cache. Failing to clear the cache might result in unpredictable behavior in the Enterprise Console.
- When running jobs against a large number of minions, only 1,000 job returns show in SaltStack Enterprise console by default. To retain job returns for all minions, use the Enterprise API to query the `get_returns` function.
- When configuring Active Directory services, the results are limited to 10,000 users or less. Using a filter can help narrow down the directory to the specific users you would like to sync with SaltStack Enterprise.
- When creating compound targets using grains, RaaS will return no minions if the grain name has a space in the name. Change the name of the grain to remove spaces.
- When a master is set to a specific timezone and jobs are initiated from that master using the CLI, the jobs show an incorrect start time in the Activity tab. Jobs initiated from the UI are not impacted.

Install using the SaltStack Enterprise installer

The script installs all necessary dependencies and then applies Salt States to install SaltStack Enterprise. The required versions of PostgreSQL, Redis, PyOpenSSL, and Python Setuptools have been included for your convenience. This is helpful for installations where servers do not have direct internet access.

The Enterprise installer installs Python 3.6 on your Salt Master by default. To install SaltStack Enterprise with earlier versions of Python on Salt Masters, you must follow the *Manual Installation* instructions. Installing any version earlier than Python 2.7 on your Salt Masters is not recommended.

> The SaltStack Enterprise installer is intended only for initial installation. If you are upgrading your installation to the latest version of SaltStack Enterprise, follow the *Upgrade* instructions.

RHEL 7 installation

Download the installer files on the SaltStack Enterprise website .

Complete steps below for either a single node or multi-node installation.

- Single node
- Multi-node

Import key files

To import the `.asc` keyfiles in the zipfile into the RPM packaging system on the machines where you intend to install SaltStack Enterprise components, run:

```
rpmkeys --import *.asc
```

Verify files

To validate that the installer zipfile was not altered after being created by SaltStack, compare the SHA-256 hash for your copy of the zipfile to the one included below.

You can calculate the hash for your copy with:

```
sha256sum SaltStack_Enterprise-6.2.0+5_Installer.zip
```

The output of the command should match the following:

```
743bf4fc91b957286b1071f8e59c830540041c8785450574e23c03a3ec6f1c2f SaltStack_Enterprise-6.2.0+5_Installer.zip
```

Single node

> If your version of RHEL 7 is lower than 7.4, you will need to update your OpenSSL version to 1.0.2k before running the installation script.
>
> If this version is not available to you via a `yum` update, or your server does not have direct internet access, retrieve the following packages from Red Hat or from your preferred public mirror:
>
> - openssl-1.0.2k-12.el7.x86_64.rpm
> - openssl-libs-1.0.2k-12.el7.x86_64.rpm

Use this method if you want to install the Salt Master, SaltStack Enterprise, Redis, and PostgreSQL on the same node. This is appropriate for installations with up to 1,000 minions.

For installations with more than 1,000 minions, please perform a multi-node installation. See Multi-node installation.

1. Extract the files.

```
$ unzip SaltStack_Enterprise-
6.2.0+5_Installer.zip
$ cd sse_installer
```

2. OPTIONAL: If Salt is already installed on your system and you are running Python 3 (included in Salt 2019.2.2 or higher), run the following commands:

```
yum install https://repo.ius.io/ius-release-el7.rpm https://dl.fedoraproject.org/pub/epel/epel-release-latest-
7.noarch.rpm
```

```
yum install python36-pyOpenSSL
```

> This step is not necessary if you are installing Salt for the first time.

3. Run the command:

```
$ sudo ./setup_single_node.sh
```

This script configures a Salt Master and Salt Minion. It then installs PostgreSQL, Redis, SaltStack Enterprise, and the Salt Master Plugin on the same server.

> This should be a fresh installation of RHEL. Ideally, Salt should not yet be installed.
>
> If both the Salt Master and Salt Minion are installed, the script skips this step and proceeds with the setup of SaltStack Enterprise.
>
> If either the Salt Master or the Salt Minion packages are installed, but not both, the script will terminate.
>
> This protects the user from accidentally disrupting an existing installation.

4. Confirm that you can log in to SaltStack Enterprise.

Log in to the web console using your browser (Chrome is recommended). The default installation uses `https://` and generates a self-signed certificate.

The default credentials are as follows:

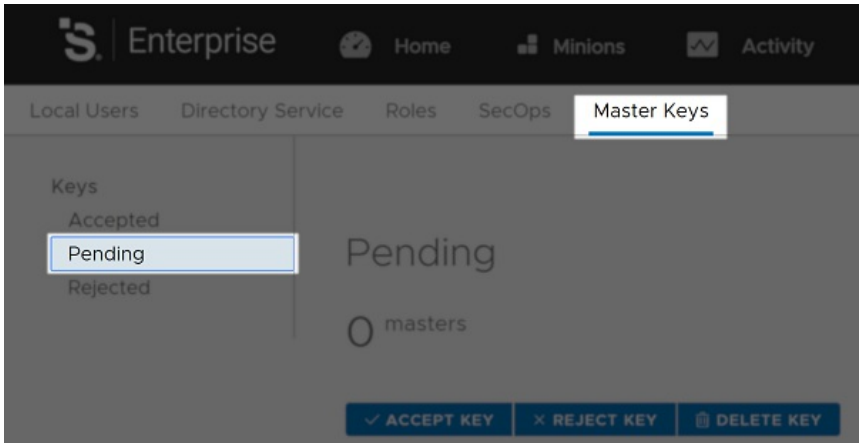
- URL: `https://IP_or_DNS_name_of_your_Server`
- Username: `root`
- Password: `salt`

By default, the installer does not accept the master key. Accepting the key is covered in the next step.

> The `setup_single_node.sh` script does not modify firewall rules.
>
> Please ensure that access is allowed to port 443 in your firewall rules for all appropriate systems (Salt Masters, web-based interface users, remote systems calling the Enterprise API, etc).

5. Accept the Salt Master key.

   Go to **Menu** > **Administration** > **Master Keys** and click **Pending**.



   Select your master key from the list and click **Accept Key**.

   Continue to configure your SaltStack Enterprise installation. See the *Initial configuration* tab.

Multi-node

Use this method when installing SaltStack Enterprise on a distributed system. This method is required for installations with more than 1,000 minions, but is perfectly appropriate for smaller installations.

For a multi-node installation, you will need to integrate the pillar and configuration states into your existing environment (these are provided within the installation download).

The starting point for this procedure is that you have created the following node types:

- Salt Master
- PostgreSQL
- Redis
- SaltStack Enterprise API (eAPI)

Each of these servers must be a Salt Minion of the Salt Master.

1. On the Salt Master, extract the files.

```
$ sudo unzip SaltStack_Enterprise-
6.2.0+5_Installer.zip
$ sudo cd sse_installer
```

2. Copy the pillar and state files into your `pillar_roots` and `file_roots` location.

   For example, in a default Salt Master configuration where the pillar and configuration state file roots are `/srv/pillar` and `/srv/salt`, the commands to copy the related files into their correct locations would be:

```
$ sudo mkdir /srv/salt
$ sudo cp -r salt/sse /srv/salt/
$ sudo mkdir /srv/pillar
$ sudo cp -r pillar/sse /srv/pillar/
```

   > This assumes that you do not already have a folder named “sse” for some unrelated purpose under either your pillar or configuration state root.

3. Create or update your `/srv/pillar/top.sls` file with the content from the provided `sse_install/pillar/top.sls` file. Define the list of minion IDs for your PostgreSQL, Redis, eAPI, and Salt Masters.

For example:

```
{# Pillar Top File #}

{# Define SSE Servers #}

{% load_yaml as sse_servers %}
  - saltpgsql
  - saltredis
  - salteapi
  - saltmaster
{% endload %}

base:

{# Assign Pillar Data to SSE Servers #}
{% for server in sse_servers %}
  '{ { server }}':
    - sse
{% endfor %}
```

4. Update /srv/pillar/sse/sse_settings.yaml with the values appropriate for your environment. These settings will be used by the configuration state files to deploy and manage your SaltStack Enterprise deployment.

   - *Section 1*

     You will need to provide the Minion ID (as opposed to the IP or DNS name) for each server type. Please note that `pg_server` and `redis_server` items are single values. The `eapi_servers` and `salt_masters` items are lists, as these two server types have high-availability deployment options supported by the installation states.

   - *Section 2*

     You will need to specify the `pg_endpoint` for your PostgreSQL server. For this option, be sure to specify the DNS name or IP address for your PostgreSQL server (not the Minion ID). The standard PostgreSQL port is provided, but may be overridden, if desired.

     - This is specified as the `pg_endpoint` as some installations may have configured a separate PostgreSQL server (or cluster) that is not managed by this installation process. If that is the case, you will want to exclude the action to highstate the PostgreSQL server in step 8 of this guide.
     - If you are in a virtualized environment, take care to specify the *internal* address, as opposed to the *public* address.

     You will also specify the username and password for the PostgreSQL user that will be used by the eAPI server(s) to authenticate to PostgreSQL. This user will be created for you when you run the configuration states.

   - *Section 3*

     You will need to specify the `redis_endpoint` for your Redis server. The standard Redis port is provided, but may be overridden.

     The `redis_username` and `redis_password` are also specified in this section.

   - *Section 4*

     You will next define the configuration settings for your eAPI servers. The initial `eapi_username` and `eapi_password` values are `root` and `salt`, respectively.

     - If this is a fresh installation, it is important that you *do not change* these values. During the initial run of these states, the installation process will establish the database with these default credentials then connect through the eAPI service to establish your default Targets and Jobs.
     - After your initial deployment is completed and you have tested your access to the web-based user interface, then you are *strongly advised* to do the following:

       1. Update the `root` user's password via the web-based user interface.
       2. Update `/srv/pillar/sse/sse_settings.yaml` with the new password.
       3. Reapply the highstate on your Salt Master(s).

You will need to specify the `eapi_endpoint` for your SaltStack Enterprise server. For this option, be sure to specify the DNS name or IP address for your eAPI server (not the Minion ID).

- This is referred to as the `eapi_endpoint`, as some installations host multiple eAPI servers behind a load balancer.
- You may also specify whether or not SSL should be enabled on the eAPI servers and if the SSL certificate should be validated. It is *strongly recommended* to enable SSL. SSL validation is not required by the installer, but is likely a security requirement in environments that host their own certificate authority.
- The `eapi_standalone` option is present to provide direction to the configuration states if Pillar data is being used in a single node deployment. In that event, all IP communication would be directed to the loopback address. Since you are using this guide, you should leave this set to `False`.
- The `eapi_deploy_default_spm` option is present to suppress the deployment of the default Jobs, Targets, or files in the SSE Filesystem provided by SaltStack Enterprise. If are deploying an update to an existing installation and you have modified any of these items, you will likely want to set this to `False`. Otherwise, `True` is recommended.
- The `eapi_failover_master` option is present to support deployments where Salt Masters (and Salt Minions) are operating in "Failover" mode. For Multi-Master configurations, SaltStack strongly recommends use of "Active" Multi-Master configurations.
- The `eapi_key` option is present to allow the user to define the encryption key that SaltStack Enterprise uses to manage encrypted data in the PostgreSQL database. This key should be unique for each installation. A default is provided, but a custom key can be generated by running the following command:

  ```
  openssl rand -hex 32
  ```

- *Section 5*

  The `customer_id` value uniquely identifies a SaltStack deployment. Primarily, it becomes the suffix of the schema name of the `raas_*` database in PostgreSQL. A default is provided, but a custom key can be generated by running the command:

  ```
  cat /proc/sys/kernel/random/uuid
  ```

  The `cluster_id` value defines the ID for a set of Salt Masters, when configured in either "Active" or "Failover" Multi-Master mode. This prevents Salt Minions that are reporting to multiple Masters from being reported multiple times in the Targets view within the SaltStack Enterprise.

5. Create or update your `/srv/salt/top.sls` file with the content from the provided `sse_install/salt/top.sls` file.

   The syntax within will leverage the Pillar data provided in *Section 1* to provide the Minion IDs of the nodes that will require the SSE Pillar data.

For example:

```
base:

  {# Target SSE Servers, according to Pillar data #}

  # SSE PostgreSQL Server
  'I@sse_pg_server:{{ grains.id }}':
    - sse.eapi_database

  # SSE Redis Server
  'I@sse_redis_server:{{ grains.id }}':
    - sse.eapi_cache

  # SSE eAPI Servers
  'I@sse_eapi_servers:{{ grains.id }}':
    - sse.eapi_service

  # SSE Salt Masters
  'I@sse_salt_masters:{{ grains.id }}':
    - sse.eapi_plugin
```

6. Sync grains.

   For Pillar data to be properly generated, we must confirm that the Salt Master has all grain data from each of the Minions that will provide a part of the SaltStack Enterprise functionality.

   ```
   $ sudo salt -L '[LIST_OF_SSE_RELATED_NODES]' saltutil.refresh_grains
   ```

7. Refresh and confirm pillar data.

   Prior to running the SaltStack Enterprise deployment via hightate, confirm that each of the SaltStack Enterprise related nodes has received the Pillar data defined in the `sse_settings.yaml` file and that it appears as expected.

   ```
   $ sudo salt -L '[LIST_OF_SSE_RELATED_NODES]' saltutil.refresh_pillar
   ```

   If your Pillar data appears to be correct, proceed with the next step.

8. Apply the highstate to the following servers:

   - PostgreSQL Server
   - Redis Server
   - SaltStack Enterprise Server(s)
   - Salt Master(s)

   ```
   $ sudo salt <MINION_ID_OF_RELATED_SERVER> state.highstate
   ```

   > During the initial application of the highstate to the first Salt Master, you may see the following message:
   >
   > ```
   > Authentication error occurred.
   > ```
   >
   > This displays because the master has not yet authenticated to raas, but the master plugin installation state will restart the Salt Master process and the issue will be resolved automatically.

9. Confirm that you can log in to SaltStack Enterprise.

   Log in to the web-based interface using your browser (Chrome is recommended). The default installation uses `https://` and generates a self-signed certificate.

   The default credentials are as follows:

   - URL: `https://IP_or_DNS_name_of_your_Server`

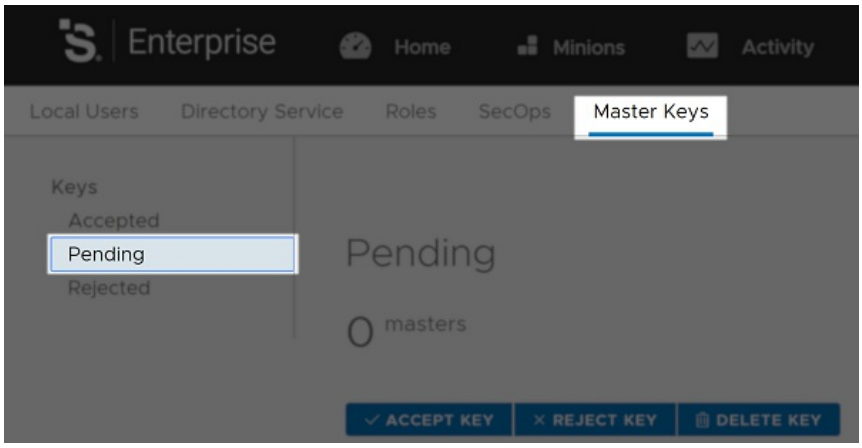
- Username: `root`
- Password: `salt`

By default, the installer does not accept the master key. Accepting the key is covered in the next step.

> The SaltStack Enterprise Installer does not modify firewall rules.
> Please ensure that firewall access is allowed on the following ports from the following nodes:
>
> - PostgreSQL is accessible by (5432 by default)
>   - eAPI servers
> - Redis is accessible by (6379 by default)
>   - eAPI Servers
> - eAPI endpoint is accessible by (443 by default)
>   - Salt Masters
>   - Web-based interface users
>   - Remote systems calling the Enterprise API
> - Salt Masters are accessible by (4505/4506 by default)
>   - All Salt Minions configured to use the related Salt Master

10. Accept the Salt Master key.

    Go to **Menu** > **Administration** > **Master Keys** and click **Pending**.

    Select your master key from the list and click **Accept Key**.

    Continue to configure your SaltStack Enterprise installation. See the *Initial configuration* tab.

11. Remove the Pillar "Top" file from step 3, `/srv/pillar/top.sls`.

    This step is to avoid regenerating the data it contains every time you refresh pillar data in the future.

Configuring multiple RaaS servers

Use this method if you need to configure multiple RaaS servers that will share a single PostgreSQL database and Redis server. This method is also sometimes called *clustering*.

In order to set up multiple RaaS servers, all the RaaS servers must:

- Access the same PostgreSQL database
- Share the same key space
- Use the same `/etc/raas/pki/.raas.key` and `/etc/raas/raas.secconf`

These instructions demonstrate how to install the PostgreSQL and Redis services on the primary RaaS server using the single node installer.

To configure multiple RaaS servers:

1. Follow the steps to install two standalone SaltStack Enterprise servers using thesingle node instructions. Both servers should run SaltStack Enterprise in standalone mode, meaning each server has its own local version of PostgreSQL and Redis.
2. On the first RaaS server, stop the RaaS, Redis, and PostgreSQL services:

   ```
   systemctl stop raas
   ```

   ```
   systemctl stop redis
   ```

   ```
   systemctl stop postgresql-12
   ```

   > The command to stop PostgreSQL may differ if you are running a different version.

3. Update your `postgresql pg_hba.conf` file to allow remote connections from the other RaaS server. Append the following entry to the end of that file, replacing the example IP address with the IP address of the second RaaS server:

   ```
   # Allow connection from RaaS 2
   host all all 127.31.4.137/32 trust
   ```

4. Update your `/etc/redis.conf` file to allow binding to all interfaces. By default the bond is set to localhost.

   ```
   #bind 127.0.0.1
   ```

5. Start the services and verify their status.

   ```
   systemctl start postgresql-12
   ```

   ```
   systemctl status postgresql-12
   ```

   ```
   systemctl start redis
   ```

   ```
   systemctl status redis
   ```

   ```
   systemctl start raas
   ```

   ```
   systemctl status raas
   ```

6. Access the Enterprise Console using the URL for the first RaaS server to confirm that SaltStack Enterprise is working properly on the first server.
7. On the second RaaS server, stop the RaaS, Redis, and PostgreSQL services:

   ```
   systemctl stop raas
   ```

   ```
   systemctl stop redis
   ```

   ```
   systemctl stop postgresql-12
   ```

8. On the second RaaS server, update the `/etc/raas/raas` file to connect to the remote Redis and PostgreSQL services on the *first* RaaS server. The `customer_id` configuration should be identical.

```
customer_id: 43cab1f4-de60-4ab1-85b5-1d883c5c5d09
sql:
  dialect: postgresql
  host: 172.31.8.237
  port: 5432
  driver: psycopg2
  ssl: True

redis:
  url: redis://172.31.8.237:6379
```

9. Copy the `/etc/raas/pki/.raas.key` and `/etc/raas/secconf` from the first server to the second server. Maintain the access and permissions, as shown in this example:

```
# ls -l /etc/raas/raas.secconf
-rw-------. 1 raas raas 313 Jan 21 17:21 /etc/raas/raas.secconf
# ls -l /etc/raas/pki/.raas.key
-rwx------. 1 raas raas 77 Jan 21 17:17 /etc/raas/pki/.raas.key
```

10. Start the RaaS service and verify its status.

```
systemctl start raas
```

```
systemctl status raas
```

11. Access the Enterprise Console using the URL for the second RaaS server to confirm that SaltStack Enterprise is working properly on the second server.

12. To test the configuration, create a new object, such as a new target. Verify that the change is present on both servers when you refresh the Enterprise Console.

13. On the second Raas server, disable the Redis and PostgreSQL services:

```
systemctl disable redis
```

```
systemctl disable postgresql-12
```

Package Key IDs

The SaltStack Enterprise Installer supports situations where target machines might not be connected to the internet. In addition, some machines might be configured to validate RPM package signatures, but might not be able to connect to the internet to automatically retrieve the correct public keys.

These keys are included in the installer zipfile for easy import on such machines. However, we *strongly recommend* validating that the keys provided by SaltStack match the official ones.

The key IDs are as follows, along with the canonical location of each:

Key Name	Key ID	Location
Fedora EPEL	352C64E5	https://getfedora.org/static/352C64E5.txt
IUS Community Project	9CD4953F	https://dl.iuscommunity.org/pub/IUS-COMMUNITY-GPG-KEY
PostgreSQL Global Dev Group	442DF0F8	https://download.postgresql.org/pub/repos/yum/RPM-GPG-KEY-PGDG-96
SaltStack Packaging Team	DE57BFBE	http://repo.saltstack.com/yum/redhat/7/x86_64/2018.3/SALTSTACK-GPG-KEY.pub

Install SaltStack Enterprise manually on RedHat

These instructions walk you through installing Enterprise API without using the installation states. These instructions are intended for advanced users who need granular control over the installation process, and who are familiar with PostgreSQL and Redis database configuration.

The steps below are confirmed for a standalone deployment of SaltStack Enterprise (where all related services reside on a single host). As an advanced user, you will likely adapt these instructions to your deployment. If you are not an advanced user, consider using the deployment states provided by installer. See *Use the installer*.

> SaltStack Enterprise requires a PostgreSQL 9.6 database, but PostgreSQL 12.1 is recommended.

Download the packages for your environment

Download the packages on the SaltStack Enterprise website .

Import key files

To import the `.asc` keyfiles in the zipfile into the RPM packaging system on the machines where you intend to install SaltStack Enterprise components, run:

```
rpmkeys --import *.asc
```

Verify files

To validate that the installer zipfile was not altered after being created by SaltStack, compare the SHA-256 hash for your copy of the zipfile to the one included below.

You can calculate the hash for your copy with:

```
sha256sum SaltStack_Enterprise-6.2.0+5_Installer.zip
```

The output of the command should match the following:

```
743bf4fc91b957286b1071f8e59c830540041c8785450574e23c03a3ec6f1c2f SaltStack_Enterprise-6.2.0+5_Installer.zip
```

Red Hat Enterprise Linux 7/CentOS 7

Step 1: PostgreSQL database installation and configuration

1. Install PostgreSQL.

```
 $ sudo wget https://download.postgresql.org/pub/repos/yum/12/redhat/rhel-7.12-x86_64/pgdg-redhat-repo-
latest.noarch.rpm
 $ sudo yum install pgdg-*.noarch.rpm
 $ sudo yum update
 $ sudo yum -y install postgresql12-server postgresql12-contrib
 /usr/pgsql-12/bin/postgresql-12-setup initdb
```

2. Update the `pg_hba.conf` file as needed to enable connections from your SaltStack Enterprise server. Optionally, enable ssl .

3. Start PostgreSQL and create a database account for Enterprise API, for example:

```
systemctl enable postgresql-12
systemctl start postgresql-12
sudo su - postgres -c 'createuser -s -P salt_eapi'
# This account has Superuser privileges so that
# various extensions my be installed.
# After initial deployment the Superuser privilege
# may be removed.
```

Step 2: Redis installation and configuration

1. Download the `Redis` and `jemalloc` installation packages for RH/CentOS that are provided in the download section.

```
$ sudo yum install redis40u-4.0.11-1.ius.el7.x86_64.rpm jemalloc-3.6.0-1.el7.x86_64.rpm
```

2. OPTIONAL: Update configuration

   If you are setting up Redis on a host that is separate from the SaltStack Enterprise Server, you will need to configure Redis to accept remote connections and to limit access using a password. To do this, update the `/etc/redis.conf` file, specifying the bind parameter and setting the password that your SaltStack Enterprise servers should use to authenticate.

```
bind 0.0.0.0
requirepass
```

3. Start the Redis service

```
$ sudo systemctl enable redis
$ sudo systemctl start redis
```

Step 3: SaltStack Enterprise installation and configuration

1. Download the `Python3.5` and `libpython3.5` installation packages for RH/CentOS that are provided in the download section.

```
$ sudo yum install python35u-libs-3.5.6-1.*.rpm python35u-3.5.6-1.*.rpm
```

2. Download and install the Red Hat/CentOS SaltStack Enterprise RPM.

```
$ sudo yum install raas-6.2.0+5-0.el7.x86_64.rpm
```

3. Update RaaS Configuration File.

   `/etc/raas/raas`

   Update the `sql` configuration to provide the host and port created in the previous section. If you plan to use SSL, set `ssl` to `True`.

```
sql:
  dialect: postgresql
  host: localhost
  port: 5432
  driver: psycopg2
  ssl: false
```

   Define options for background workers.

```
background_workers:
  combined_process: True
  max_tasks: 100000
  max_memory: 1048576
```

> SaltStack Enterprise includes a range of different background worker settings to improve performance for various deployment scenarios. The following guides might be helpful for tuning:
>
> - SaltStack Enterprise Performance Configurations
> - Background Worker Options

Configure the location of your Redis server.

```
redis:
 url: redis://<Redis_IP>:6379
```

4. To store database credentials for both PostgreSQL and Redis in an encrypted file, run the following command:

```
$ su - raas -c 'raas save_creds'
```

Follow the prompts to set up your username and password for Redis and PostgreSQL. If you would prefer to leave those values blank, press the Enter key when prompted.

The credentials will be stored in `/etc/raas/raas.secconf`.

> If credentials appear in both `/etc/raas/raas` and `/etc/raas/raas.secconf`, the settings in the plaintext `/etc/raas/raas` take precedence.
>
> For more on securing credentials, see Securing credentials in your SaltStack Enterprise configuration.

5. Enable the `raas` service at system startup and launch the service.

```
sudo systemctl enable raas
sudo systemctl start raas
```

6. Confirm that you can connect to the web console in a web browser.

   - url: `http://your_raas_server/`
   - Username: `root`
   - Password: `salt`

To enable SSL certificates for the Enterprise Console, see the next section.

Enable SSL on Red Hat Enterprise Linux 7/CentOS 7 (optional)

> Instructions on how to update SSL certificates for SaltStack Enterprise, are available in the SaltStack Support Portal.

1. Install pyOpenSSL.

```
Red Hat/CentOS
$ sudo yum install pyOpenSSL
```

2. Create and set permissions for the certificate folder for `raas`.

```
sudo mkdir -p /etc/raas/pki
sudo chown raas:raas /etc/raas/pki
sudo chmod 750 /etc/raas/pki
```

3. Generate keys for `raas` using salt, or provide your own.

```
sudo salt-call --local tls.create_self_signed_cert tls_dir=raas
sudo chown raas:raas /etc/pki/raas/certs/localhost.crt
sudo chown raas:raas /etc/pki/raas/certs/localhost.key
sudo chmod 400 /etc/pki/raas/certs/localhost.crt
sudo chmod 400 /etc/pki/raas/certs/localhost.key
```

4. Enable SSL.

   To enable SSL connections to Enterprise Console, generate a PEM-encoded SSL certificate or ensure that you have access to an existing PEM-encoded certificate. Save the `.crt` and `.key` files to `/etc/pki/raas/certs`.

5. Update RaaS Configuration.

   Open `/etc/raas/raas` in a text editor and configure the following values (replace `<filename>` with your certificate filename).

```
tls_crt: /etc/pki/raas/certs/<filename>.crt
tls_key: /etc/pki/raas/certs/<filename>.key
port: 443
```

6. Restart the Enterprise API service.

```
$ sudo systemctl restart raas
```

7. Verify the Enterprise API is running.

```
$ sudo systemctl status raas
```

8. Confirm that you can connect to the web console in a web browser.

   - url: `https://your_raas_server/`
   - Username: `root`
   - Password: `salt`

Deploy your license key

When deploying a SaltStack Enterprise server, you will need to add your license key to the`/etc/raas` folder. Upon doing so, you will need to set the ownership of this file to `raas` user, as follows:

```
sudo chown raas:raas /etc/raas/raas.license
sudo chmod 400 /etc/raas/raas.license
```

Install Salt Master plugin

At this point in the manual installation, you should install Salt Master plugin. For instructions, see *Install Master Plugin*.

Install SaltStack Enterprise manually on SUSE

These instructions walk you through installing Enterprise API without using the installation states. These instructions are intended for advanced users who need granular control over the installation process, and who are familiar with PostgreSQL and Redis database configuration.

The steps below are confirmed for a standalone deployment of SaltStack Enterprise (where all related services reside on a single host). As an advanced user, you will likely adapt these instructions to your deployment. These instructions are for SUSE Linux Enterprise 12 and SUSE Linux Enterprise 15.

> SaltStack Enterprise requires a PostgreSQL 9.6 database, but PostgreSQL 12.1 is recommended.

Download the packages for your environment

Download the packages on the SaltStack Enterprise website .

Import key files

To import the `.asc` keyfiles in the zipfile into the RPM packaging system on the machines where you intend to install SaltStack Enterprise components, run:

```
rpmkeys --import *.asc
```

Verify files

To validate that the installer zipfile was not altered after being created by SaltStack, compare the SHA-256 hash for your copy of the zipfile to the one included below.

You can calculate the hash for your copy with:

```
sha256sum SaltStack_Enterprise-6.2.0+5_Installer.zip
```

The output of the command should match the following:

```
743bf4fc91b957286b1071f8e59c830540041c8785450574e23c03a3ec6f1c2f SaltStack_Enterprise-6.2.0+5_Installer.zip
```

SUSE Linux Enterprise Server 12

Step 1: PostgreSQL database installation and configuration

1. Install PostgreSQL.

```
$ zypper addrepo
https://download.opensuse.org/repositories/server:/database:/postgresql/SLE_12_SP4/server:database:postgresql.repo
$ zypper refresh
$ zypper install postgresql96 postgresql96-server postgresql96-contrib
# init the db by starting and stopping the postgresql service
$ systemctl start postgresql
$ systemctl stop postgresql
```

> These instructions explain how to install PostgreSQL 9.6. Although PostgreSQL 12.1 is recommended, instructions for installing PostgreSQL 12.1 on SLES 12 are currently unavailable. Contact your database administrator for assistance if needed.

2. Start PostgreSQL and create a database account for Enterprise API. For example:

```
$ systemctl start postgresql
$ su - postgres -c 'createuser -d -P -s root'
```

> SaltStack Enterprise includes a range of different background worker settings to improve performance for various deployment scenarios. The following guides might be helpful for tuning:
>
> - SaltStack Enterprise Performance Configurations
> - Background Worker Options

Step 2: Redis installation

1. Install Redis.

```
$ zypper addrepo https://download.opensuse.org/repositories/server:/database/SLE_12_SP4/server:database.repo
$ zypper refresh
$ zypper install redis
```

2. Start Redis. For example:

```
# Start the Redis service
$ redis-server
# Start Redis in the background
$ redis-server --daemonize yes
```

You can use the following optional commands to ensure Redis is running as intended:

```
# Check if Redis is already running; will return PONG if running
$ redis-cli ping
# Stop the Redis service
$ redis-cli shutdown
```

If you are setting up Redis on a host that is separate from the SaltStack Enterprise Server, you will need to configure Redis to accept remote connections and to limit access using a password. To do this, update the `/etc/redis.conf` file, specifying the bind parameter and setting the password that your SaltStack Enterprise servers should use to authenticate.

```
bind 0.0.0.0
requirepass
```

Step 3: SaltStack Enterprise installation and configuration

1. Import the RPM signing key.

```
rpm --import http://repo.saltstack.com/py3/redhat/7.7/x86_64/latest/SALTSTACK-GPG-
KEY.pub
```

2. Install the SLES 12 RPM.

```
$ zypper install raas-6.2.0+5-0.sles12.x86_64.rpm
```

3. Update RaaS Configuration File.

   `/etc/raas/raas`

   Update the `sql` configuration to provide the host and port created in the previous section. If you plan to use SSL, set `ssl` to `True`.

```
sql:
  dialect: postgresql
  host: localhost
  port: 5432
  driver: psycopg2
  ssl: false
```

Define options for background workers.

```
background_workers:
  combined_process: True
  max_tasks: 100000
  max_memory: 1048576
```

> SaltStack Enterprise includes a range of different background worker settings to improve performance for various deployment scenarios. The following guides might be helpful for tuning:
>
> - SaltStack Enterprise Performance Configurations
> - Background Worker Options

Configure the location of your Redis server.

```
redis:
  url: redis://<Redis_IP>:6379
```

4. To store database credentials for both PostgreSQL and Redis in an encrypted file, run the following command:

```
$ su - raas -c 'raas save_creds'
```

Follow the prompts to set up your username and password for Redis and PostgreSQL. If you would prefer to leave those values blank, press the Enter key when prompted.

The credentials will be stored in `/etc/raas/raas.secconf`.

> If credentials appear in both `/etc/raas/raas` and `/etc/raas/raas.secconf`, the settings in the plaintext `/etc/raas/raas` take precedence.
>
> For more on securing credentials, see Securing credentials in your SaltStack Enterprise configuration.

5. Start the Enterprise API service.

```
$ systemctl start raas
```

6. Confirm that you can connect to the web console in a web browser.
   - url: `http://your_raas_server/`
   - Username: `root`
   - Password: `salt`
7. After completing the previous steps for the manual installation for SUSE 12, you can:
   - **Enable SSL (optional)** - See the instructions after the SUSE Linux Enterprise Server 15 guide on this page.
   - **Deploy your license key** - See the instructions after the SUSE Linux Enterprise Server 15 guide on this page.
   - **Install Salt Master plugin** See the *Install Master Plugin* tab.

SUSE Linux Enterprise Server 15

Step 1: PostgreSQL database installation and configuration

1. Install PostgreSQL 12.

```
$ zypper addrepo
https://download.opensuse.org/repositories/server:/database:/postgresql/SLE_15_SP1/server:database:postgresql.repo
$ zypper refresh
# install PostgreSQL 12
$ zypper install postgresql12-server
$ zypper install postgresql12-contrib
# init the db by starting and stopping the postgresql service
$ systemctl start postgresql
$ systemctl stop postgresql
```

2. Update the `pg_hba.conf` file as needed to enable connections from your SaltStack Enterprise server.

3. Start PostgreSQL and create a database account for Enterprise API. For example:

```
$ systemctl start postgresql
$ su - postgres -c 'createuser -d -P -s root'
```

Step 2: Redis installation

1. Download and install Redis.

```
$ zypper addrepo https://download.opensuse.org/repositories/server:/database/SLE_15/server:database.repo
$ zypper refresh
$ zypper in redis
```

2. Start Redis. For example:

```
# Start the Redis service
$ redis-server
# Start Redis in the background
$ redis-server --daemonize yes
```

   You can use the following optional commands to ensure Redis is running as intended:

```
# Check if Redis is already running; will return PONG if running
$ redis-cli ping
# Stop the Redis service
$ redis-cli shutdown
```

3. OPTIONAL: Update configuration

   If you are setting up Redis on a host that is separate from the SaltStack Enterprise Server, you will need to configure Redis to accept remote connections and to limit access using a password. To do this, update the `/etc/redis.conf` file, specifying the bind parameter and setting the password that your SaltStack Enterprise servers should use to authenticate.

```
bind 0.0.0.0
requirepass
```

Step 3: SaltStack Enterprise installation and configuration

1. Import the RPM signing key.

```
rpm --import http://repo.saltstack.com/py3/redhat/7.7/x86_64/latest/SALTSTACK-GPG-
KEY.pub
```

2. Download the installation packages provided in the download section.

```
$ zypper in raas-6.2.0+5-0.sles15.x86_64.rpm
```

3. Update RaaS Configuration File.

   `/etc/raas/raas`

   Update the `sql` configuration to provide the host and port created in the previous section. If you plan to use SSL, set `ssl` to `True`.

```
sql:
  dialect: postgresql
  host: localhost
  port: 5432
  driver: psycopg2
  ssl: false
```

   Define options for background workers.

```
background_workers:
  combined_process: True
  max_tasks: 100000
  max_memory: 1048576
```

> SaltStack Enterprise includes a range of different background worker settings to improve performance for various deployment scenarios. The following guides might be helpful for tuning:
>
> - SaltStack Enterprise Performance Configurations
> - Background Worker Options

Configure the location of your Redis server.

```
redis:
  url: redis://<Redis_IP>:6379
```

4. To store database credentials for both PostgreSQL and Redis in an encrypted file, run the following command:

```
$ su - raas -c 'raas save_creds'
```

Follow the prompts to set up your username and password for Redis and PostgreSQL. If you would prefer to leave those values blank, press the Enter key when prompted.

The credentials will be stored in `/etc/raas/raas.secconf`.

> If credentials appear in both `/etc/raas/raas` and `/etc/raas/raas.secconf`, the settings in the plaintext `/etc/raas/raas` take precedence.
>
> For more on securing credentials, see Securing credentials in your SaltStack Enterprise configuration.

5. Start the Enterprise API service.

```
$ systemctl start raas
```

6. Confirm that you can connect to the web console in a web browser.

   - url: `http://your_raas_server/`
   - Username: `root`
   - Password: `salt`

Enable SSL

> Instructions on how to update SSL certificates for SaltStack Enterprise, are available in the SaltStack Support Portal.

1. Install pyOpenSSL.

```
$ zypper in python-pyOpenSSL
```

2. Create and set permissions for the certificate folder for `raas`.

```
sudo mkdir -p /etc/raas/pki
sudo chown raas:raas /etc/raas/pki
sudo chmod 750 /etc/raas/pki
```

3. Generate keys for `raas` using salt, or provide your own.

```
sudo salt-call --local tls.create_self_signed_cert tls_dir=raas
sudo chown raas:raas /etc/pki/raas/certs/localhost.crt
sudo chown raas:raas /etc/pki/raas/certs/localhost.key
sudo chmod 400 /etc/pki/raas/certs/localhost.crt
sudo chmod 400 /etc/pki/raas/certs/localhost.key
```

4. Enable SSL.

   To enable SSL connections to Enterprise Console, generate a PEM-encoded SSL certificate or ensure that you have access to an existing PEM-encoded certificate. Save the `.crt` and `.key` files to `/etc/pki/raas/certs`.

5. Update RaaS Configuration.

   Open `/etc/raas/raas` in a text editor and configure the following values (replace `<filename>` with your certificate filename).

   ```
   tls_crt: /etc/pki/raas/certs/<filename>.crt
   tls_key: /etc/pki/raas/certs/<filename>.key
   port: 443
   ```

6. Restart the Enterprise API service.

   ```
   $ sudo systemctl restart raas
   ```

7. Verify the Enterprise API is running.

   ```
   $ sudo systemctl status raas
   ```

8. Confirm that you can connect to the web console in a web browser.

   - url: `https://your_raas_server/`
   - Username: `root`
   - Password: `salt`

Deploy your license key

When deploying a SaltStack Enterprise server, you will need to add your license key to the`/etc/raas` folder. Upon doing so, you will need to set the ownership of this file to `raas` user, as follows:

```
sudo chown raas:raas /etc/raas/raas.license
sudo chmod 400 /etc/raas/raas.license
```

Install Salt Master plugin

At this point in the manual installation, you should install Salt Master plugin. For instructions, see *Install Master Plugin*.

Install the Master Plugin

These instructions explain the process for installing Salt Master plugin when doing a manual installation. Before installing Salt Master plugin, follow the manual installation instructions for your operating system. For more information, see the *Manual installation RedHat* or the *Manual installation SUSE* tab.

Although the SaltStack Enterprise server is restricted to specific operating systems, the Salt Master plugin can run on any operating system.

Dependencies

Before you begin these steps, ensure that you have:

- Installed and configured the PostgreSQL database
- Installed and configured the Redis database
- Enabled SSL (optional)

For more information, see the *Manual installation RedHat* or the *Manual installation SUSE* tab.

Configure the Salt Master plugin

1. Log in to your Salt Master.
2. Download the Salt Master plugin wheel.
3. Install the plugin (requires Python setuptools).

   Python 3.6

   ```
   #RHEL

   $ pip3.6 install SSEAPE-6.2.0+5-py2.py3-none-any.whl --prefix /usr

   #Ubuntu

   $ pip36 install SSEAPE-6.2.0+5-py2.py3-none-any.whl
   ```

   > Although Python 3.6 is recommended, Python 2.7 is also supported.

4. Verify the `/etc/salt/master.d` directory exists. If it doesn't, create it.
5. Generate the master configuration settings.

   ```
   $ sudo sseapi-config --all > /etc/salt/master.d/raas.conf
   ```

6. Edit the generated `raas.conf` file to update the following values:

   - **sseapi_ssl_validate_cert** - Validates the certificate that Enterprise API uses. The default is `True.` If you are using your own CA-issued certificates, set this value to `True` and configure the `sseapi_ssl_ca`, `sseapi_ssl_cert`, and `sseapi_ssl_cert:` settings. Otherwise set this to `False` to not validate the certificate.

     ```
     sseapi_ssl_validate_cert: False
     ```

   - **sseapi_server** - HTTP IP address of of your SaltStack Enterprise server, for example,`http://192.168.57.24`, or `https://192.168.57.24` if SSL is enabled.

7. OPTIONAL: To verify you can connect to SSL before connecting the Salt Master plugin, edit the generated`raas.conf` file to update the following values:

   - **sseapi_ssl_ca** - The path to a CA file.
   - **sseapi_ssl_cert** - The path to the certificate. The default value is `/etc/pki/raas/certs/localhost.crt`.
   - **sseapi_ssl_key** - The path to the certificate's private key. The default value is `/etc/pki/raas/certs/localhost.key`.
   - **id** - Comment this line out by adding a `#` at the beginning. It is not required.

8. Restart the Salt Master.

```
$ sudo systemctl restart salt-master
```

Continue to configure your SaltStack Enterprise installation. See the *Initial configuration* tab.

Upgrading SaltStack Enterprise

In an existing installation, SaltStack Enterprise can be upgraded to the latest version. If you are upgrading your SaltStack Enterprise installation, refer to the following upgrade instructions. Do not attempt to install using the installer or manual installation instructions.

> These upgrade instructions cover an upgrade from SaltStack Enterprise version *6.1.0* to version *6.2.0*. A best practice is to always upgrade from the latest major version of SaltStack Enterprise to the new release.
>
> Or if upgrading from a version earlier than *6.1.0*, you might upgrade in increments from one major release to the next for best results.
>
> For instructions on upgrading to earlier SaltStack Enterprise versions, see the SaltStack Support Portal .

Download the Python wheel

Download the latest Python wheel.

> Download the Python wheel on the SaltStack Enterprise website .

Download the RPM

> Download the RPM on the SaltStack Enterprise website .

What to back up prior to the upgrade

Files / Directories

- `/etc/raas/raas`
- `/etc/raas/pki/` - this contains hidden files, so back up the entire directory
- `/etc/salt/master.d/raas.conf` - located on each Salt Master
- `postgres.conf` - if local PostgreSQL
- `pg_hba.conf` - if local PostgreSQL

Database

When upgrading your raas server, the database schema will need to be updated. Make sure to create a backup of your database before the upgrade.

To back up your database, first look up your PostgreSQL database name.

Backing up your database

1. Log in as the `postgres` user.

   ```
   $ sudo su -
   postgres
   ```

2. Get your database name.

   ```
   $ psql # enter postgresql
   $ \l # list databases
   ```

   To exit PostgreSQL and log out as `postgres` user, press `ctrl-d` and then run:

   ```
   exit
   ```

3. Copy database contents to file. An example of this command is:

```
pg_dump -U salt_eapi raas_db_name > postgres_raas_backup_$(date +%Y-%m-%d).sql
```

See PostgreSQL database backups for more information.

Upgrading to PostgreSQL 12.1

New installations of SaltStack Enterprise are configured with PostgreSQL 12.1 by default. Upgrading to PostgreSQL 12.1 is not required. However, upgrading to PostgreSQL 12.1 can possibly improve performance. For instructions on upgrading to PostgreSQL 12.1, see PostgreSQL upgrade .

Upgrading SaltStack Enterprise

1. **Back up your database**. See Backing up your database.
2. Save any changes you made to the default file system, pillar data, and jobs as new files or jobs.
3. Note any pillar assignments that are made to the default targets. These need to be re-assigned after upgrade.

> Database upgrades require re-indexing data. Depending on the complexity of your data, a database upgrade could potentially take several hours. To prevent service disruptions, consider upgrading the database during slower business hours or trimming your database prior to an upgrade. You could also try running a dress rehearsal in a test environment to get a sense of how long the upgrade will take.

On the SaltStack Enterprise server

1. Stop the `raas` service.

```
$ sudo systemctl stop raas
```

2. Back up or remove the log file(s) at `/var/log/raas/raas`. This provides a clean log file if troubleshooting is required.
3. Back up or rename the `/etc/raas/raas` config file. You'll need to restore this file after upgrading.
4. Remove the currently installed version of Enterprise API.

```
$ sudo yum remove raas
```

5. Upgrade SaltStack Enterprise server by installing the latest RPM. For example:

```
$ sudo yum install raas-6.2.0+5-0.el7.x86_64.rpm
```

> To download the RPM, see Download the RPM.

6. **Restore the backup** of the `/etc/raas/raas` configuration file.
7. Update permissions for the `raas` user:

```
$ sudo chown raas:raas /var/log/raas/raas
$ sudo chown -R raas:raas /etc/raas/
$ sudo chown -R raas:raas /etc/pki/raas/certs/*.crt
$ sudo chown -R raas:raas /etc/pki/raas/certs/*.key
$ sudo chmod 400 /etc/pki/raas/certs/*.crt
$ sudo chmod 400 /etc/pki/raas/certs/*.key
$ sudo chown -R raas:raas /srv/raas
$ sudo chown -R raas:raas /opt/saltstack
```

8. OPTIONAL: Add new section to the `/etc/raas/raas` file:

   > This step is optional and only applies to organizations that have a valid SaltStack Comply and/or SaltStack Protect license. These add-on modules are included in SaltStack Enterprise versions *6.0* and later. The following configuration options in /etc/raas/raas are specific to these add-on modules.

   ```
     sec:
       ingest_override: true
       locke_dir: locke
       post_ingest_cleanup: true
       username: 'secops'
       content_url:
   'https://enterprise.saltstack.com/secops_downloads'
       download_enabled: true
       download_frequency: 86400
       stats_snapshot_interval: 3600
       compile_stats_interval: 10
       ingest_on_boot: True
       content_lock_timeout: 60
       content_lock_block_timeout: 120
   ```

9. OPTIONAL: Add new section to the `/etc/raas/raas` file:

   > This step is optional and only applies to organizations that have a valid SaltStack Protect license. This add-on module is included in SaltStack Enterprise versions *6.0* and later. The following configuration options in /etc/raas/raas are specific to these add-on modules.

   ```
     vman:
        vman_dir: vman
        download_enabled: true
        download_frequency: 86400
        username: vman
        content_url:
   'https://enterprise.saltstack.com/vman_downloads'
        ingest_on_boot: true
        compile_stats_interval: 60
        stats_snapshot_interval: 3600
        old_policy_file_lifespan: 2
        delete_old_policy_files_interval: 86400
   ```

10. Upgrade the RaaS database with:

    ```
    $ sudo su -
    raas
    $ raas upgrade
    ```

    > Depending on the size of your database, the upgrade can take anywhere from several minutes to over an hour.

    If you encounter errors, check the `/var/log/raas/raas` logfile for more information.

    After the upgrade, exit the session for the raas user with:

    ```
    exit
    ```

11. Start the Enterprise API service.

    ```
    $ sudo systemctl enable raas
    $ sudo systemctl start raas
    ```

On each Salt Master running the Salt Master plugin

1. Stop the `salt-master` service.

   ```
   $ sudo systemctl stop salt-master
   ```

2. Check which version of Python is running on the Salt Master. If it is running Python 3.6 or higher, no changes are needed. Otherwise, delete the prior version of the SSEAPE module. (The SSEAPE is the SaltStack Enterprise plugin for the Salt Master). For example:

   RHEL

   ```
   $ sudo rm -rf /usr/lib/python3.6/site-packages/SSEAPE*
   ```

   Ubuntu

   ```
   $ sudo rm /usr/lib/python3.6/dist-packages/SSEAPE*
   ```

3. Manually upgrade the Salt Master plugin by installing the updated Python wheel.

   ```
   #RHEL

   $ pip3.6 install SSEAPE-6.2.0+5-py2.py3-none-any.whl --prefix /usr

   #Ubuntu

   $ pip36 install SSEAPE-6.2.0+5-py2.py3-none-any.whl
   ```

4. Update the eAPI Master paths.

   - Edit `/etc/salt/master.d/eAPIMasterPaths.conf` to reference the path to the new wheel version.
   - If you are upgrading from a version of SaltStack Enterprise prior to 5.4, use the following command to generate the paths:

     ```
     RHEL:
     $ sudo yum install PyYAML
     $ sudo sseapi-config --ext-modules > /etc/salt/master.d/eAPIMasterPaths.conf

     Ubuntu:
     $ sudo apt-get install python-yaml
     $ sudo sseapi-config --ext-modules > /etc/salt/master.d/eAPIMasterPaths.conf
     ```

   - Be sure to remove any of the path references from `/etc/salt/master.d/raas.conf`, as these were relocated in a previous release of SaltStack Enterprise.

5. Create or update the `engines` section in `/etc/salt/master.d/raas.conf`:

   ```
   engines:
     - sseapi: {}
     - eventqueue: {}
     - rpcqueue: {}
     - jobcompletion: {}
   ```

6. Start the `salt-master` service.

   ```
   $ sudo systemctl start salt-master
   ```

Initial configuration

- Set up public key authentication for Salt Masters
- Change the root password
- Enable more accurate presence detection
- Back up critical data

Set up public key authentication for Salt Masters

During the master startup (unless using password authentication) a public key file will be generated. The master will start running but communication with `raas` will fail until the key is accepted.

> Until the key is accepted, the master will react slowly as it continually tries to contact `raas`.

To accept the key, complete the following:

1. Open Enterprise Console and log in using the superuser account.
2. Go to **Menu**  > **Administration** > **Master Keys**.
3. Select the **Pending** menu item.
4. Use the checkboxes to select the master's key, then choose **Accept Key**.

Change the root password

You can change the default password for the root user.

1. Open Enterprise Console and log in using the superuser account.
2. Go to **Menu**  > **Administration** > **Local Users**.
3. Select the root account and enter a new password, then click **Save**.

Enable more accurate presence detection

SaltStack Enterprise provides a job to install a Salt Beacon that sends periodic heartbeats from each Salt Minion. A good practice is to install this job on all minions to enable more accurate presence.

1. Open Enterprise Console and log in using the superuser account.
2. Go to **All Minions** and select the *All Minions* target.
3. Click **Run Job** and select *Enable Presence*.

Back up critical data

If you are not using a complete system backup solution that can restore your entire SaltStack Enterprise server, at a minimum you should back up the following files:

- **/etc/raas/pki** - This directory contains a hidden file named `.raas.key` that is used to encrypt data while at rest in the database. If you need to restore your SaltStack Enterprise server by re-installing, it is critical that you restore the original `.raas.key` file from when the database was created. If this file is lost, Enterprise API will not be able to access the database.

- **/etc/raas/raas** and **/etc/raas/raas.secconf**- These files contain SaltStack Enterprise configuration data.
- **Enterprise API Database** - Configure regular PostgreSQL database backups for the Enterprise API database.

Congratulations!

You are now ready to manage your infrastructure using SaltStack Enterprise. Click the help icon in Enterprise Console for additional guidance.

SaltStack Comply and Protect configuration

SaltStack Comply and SaltStack Protect are SaltStack Enterprise add-ons to help you harden the security of your infrastructure:

- **SaltStack Comply** provides automated compliance detection and remediation for your infrastructure. Its content library consists of industry best-practice security and compliance content, such as CIS.
- **SaltStack Protect** manages vulnerabilities on all the systems in your environment. Its content library includes advisories based on the latest Common Vulnerabilities and Exposures (CVE) entries.

These content libraries update regularly as security standards change. You can configure SaltStack Comply and SaltStack Protect to download (or ingest) content automatically as security standards change, which is recommended for most standard systems. As an alternative, the library includes the option to download content manually.

- SaltStack Comply - Automatic content ingestion for standard systems
- SaltStack Protect - Automatic content ingestion for standard systems
- SaltStack Comply - Manual content ingestion
- SaltStack Protect - Manual content ingestion
- SaltStack Comply configuration options
- SaltStack Protect configuration options

SaltStack Comply - Automatic content ingestion for standard systems

For non-air-gapped `raas` systems, SaltStack Comply content is downloaded and ingested on a periodic basis as determined by the settings in the configuration file. To configure automatic SaltStack Comply content ingestion, complete the following steps:

1. Add the following to the `raas` configuration file `/etc/raas/raas` in the `sec` section, adapting it as necessary:

```
sec:
  stats_snapshot_interval: 3600
  username: secops
  content_url: https://enterprise.saltstack.com/secops_downloads
  ingest_saltstack_override: true
  ingest_custom_override: true
  locke_dir: locke
  post_ingest_cleanup: true
  download_enabled: true
  download_frequency: 86400
  compile_stats_interval: 10
  archive_interval: 300
  old_policy_file_lifespan: 2
  delete_old_policy_files_interval: 86400
  ingest_on_boot: true
  content_lock_timeout: 60
  content_lock_block_timeout: 120
```

   > For more information about these configuration settings, see SaltStack Comply configuration options.

2. Save the file.
3. Restart the `raas` service.

```
systemctl restart raas
```

After the service restarts, SaltStack Comply content begins to download. This may take up to five minutes, depending on your internet connection.

SaltStack Protect - Automatic content ingestion for standard systems

For non-air-gapped `raas` systems, SaltStack Protect content is downloaded and ingested on a periodic basis as determined by the settings in the configuration file. To configure automatic SaltStack Protect content ingestion, complete the following steps:

1. Add the following section to the `raas` configuration file `/etc/raas/raas`, adapting it as necessary:

```
vman:
  vman_dir: vman
  download_enabled: true
  download_frequency: 86400
  username: vman
  content_url: 'https://enterprise.saltstack.com/vman_downloads'
  ingest_on_boot: true
  compile_stats_interval: 60
  stats_snapshot_interval: 3600
  old_policy_file_lifespan: 2
  delete_old_policy_files_interval: 86400
```

> For more information about these configuration settings, see SaltStack Protect configuration options.

2. Save the file.
3. Restart the `raas` service.

```
systemctl restart raas
```

After the service restarts, SaltStack Protect content begins to download. This may take up to five minutes, depending on your internet connection.

SaltStack Comply - Manual content ingestion

Air-gapped systems must update SaltStack Comply content from one of the `raas` nodes. Air-gapped systems are defined by a configuration setting of `sec/download_enabled = False`.

To configure ingestion for air-gapped systems:

1. Download the SaltStack SaltStack Comply content.
2. Log in to a `raas` node.
3. Copy the SaltStack Comply content tarball to the `raas` node (`tmp` is recommended).

   This content could be delivered by email or any other means.
4. Run the following command:

```
su - raas -c "raas ingest /path/to/locke.tar.gz.e"
```

   This returns:

```
Extracting: /tmp/locke.tar.gz -> /tmp/extracted-1551290468.5497127

Cleaning up: /tmp/extracted-1551290468.5497127

Results:

{'errors': [], 'success': True}
```

SaltStack Protect - Manual content ingestion

Air-gapped systems must update SaltStack Protect content from one of the `raas` nodes. Air-gapped systems are defined by a configuration setting of `vman/download_enabled = False`.

To configure ingestion for air-gapped systems:

1. Download the SaltStack SaltStack Protect content.

> At the time of this writing, manual downloads for SaltStack SaltStack Protect content are not yet available, but will be available soon.

2. Log in to a `raas` node.
3. Copy the SaltStack Protect content tarball to the `raas` node (`tmp` is recommended).

   This content could be delivered by email or any other means.
4. Run the following command:

   ```
   su - raas -c "raas vman_ingest /path/to/vman.tar.gz.e"
   ```

   This returns:

   ```
   Extracting: /tmp/vman.tar.gz -> /tmp/extracted-1551290468.5497127

   Cleaning up: /tmp/extracted-1551290468.5497127

   Results:

   {'errors': [], 'success': True}
   ```

SaltStack Comply configuration options

The following table describes the configuration options that are available for SaltStack Comply:

Option	Description
`stats_snapshot_interval`	How often (in seconds) SaltStack Comply stats will be collected
`compile_stats_interval`	How often (in seconds) SaltStack Comply stats will be compiled
`username`	Username to use when connecting to SaltStack Enterprise to download the most recent SaltStack Comply content (default: `secops`)
`content_url`	URL used to download SaltStack Comply content (default: `https://enterprise.saltstack.com/secops_downloads`)
`ingest_override`	When ingesting new content, overwrite existing benchmarks and checks
`locke_dir`	Path where ingestion expects to find new content (default: `locke`). If you use a relative path (no leading /), then it is relative to `/var/lib/raas/cache`)
`post_ingest_cleanup`	Remove the expanded content from the file system after ingestion (default: `True`)
`download_enabled`	Whether SaltStack Comply content downloads are allowed (default: `True`). Set this to `False` for air gapped systems.
`download_frequency`	How often in seconds will `raas` attempt to download SaltStack Comply content (default: `86400` for 24 hours)
`ingest_on_boot`	Should `raas` attempt to download SaltStack Comply content on boot? (default: `True`)
`content_lock_timeout`	How long in seconds will content download locks last (default: `60`)
`content_lock_block_timeout`	How long in seconds will content download locks block before failing (default: `120`)

SaltStack Protect configuration options

The following table describes the configuration options that are available for SaltStack Protect:

Option	Description
`vman_dir`	Location where SaltStack Protect content is expanded before ingestion. If the path is relative (no leading slash), then it is relative to the `raas` cache dir
`download_enabled`	If `True`, SaltStack Protect content downloading is enabled. Should be False for air gapped systems
`download_frequency`	The frequency in seconds of automated SaltStack Protect content downloads and ingestion
`username`	Username used to log in to enterprise.saltstack.com to get content
`content_url`	URL from which SaltStack Protect content will be downloaded
`ingest_on_boot`	If `True`, SaltStack Protect content will be downloaded and ingested soon after `raas` boot
`compile_stats_interval`	Interval in seconds between times that the compile stats will be gathered
`stats_snapshot_interval`	Interval in seconds between when stats for when SaltStack Protect content will be gathered
`old_policy_file_lifespan`	The lifespan of old policy files in days that will remain in the `raas` file system
`delete_old_policy_files_interval`	The interval in seconds between times that the old SaltStack Protect policy files in the `raas` file system will be deleted

Single Sign-On

These instructions explain the process for configuring single sign-on authentication (SSO) for SSE users. This topic explains the SSO configuration process if you are working in the command line (CLI). Be aware that you can also configure SSO using the Authentication workspace in the Enterprise Console. For more information about using the Authentication workspace to set up SSO, see the SaltStack Enterprise Help documentation.

By default, SaltStack Enterprise authenticates users by storing user settings in a local database. Using the default method, you can create new users and manage their settings in the Enterprise Console. After users are added to the local database, you can use the Roles workspace in the Enterprise Console to set role-based access control (RBAC) to define the roles and permissions for groups of users.

Alternatively, you can set up custom SSO. SaltStack Enterprise can connect to third-party identity providers that use configuration standards such as Security Assertion Markup Language (SAML) and Lightweight Directory Access Protocol (LDAP). Once it is connected, SaltStack Enterprise can sync the users from these identity providers to allow them to login.

SaltStack currently supports the following configuration standards:

- LDAP
- SAML
- OpenID Connect (OIDC)

This topic explains how to set up SAML and OIDC configurations and provides several sample configuration files by configuration standard and provider. Be aware that you could use the same method explained in the following section to configure an alternative third-party identity provider if needed.

Overview of the configuration process

To set up most configuration standards using the CLI:

1. Login as a RaaS user:

   ```
   sudo su raas
   ```

2. OPTIONAL: This step is only necessary if you manually installed SaltStack Enterprise. On the RaaS server, install the OpenSSL .xml file that is included in the installer files. Use the following command:

   ```
   yum install xmlsec1-openssl
   ```

   > RedHat doesn't have `xmlsec1` readily available in any default repositories. One possible workaround is to download the RPMs from a CentOS machine and transfer them to RedHat.

3. Navigate to the directory where you intend to save the configuration file. Any directory path is acceptable.

4. Create a YAML file with the necessary configuration information that is required by your identity service provider. See the sample configuration files by identity provider in the remainder of this guide.

   > For a description of the necessary configuration fields, see the Authentication topic in the SaltStack Enterprise Help documentation.

5. Execute the configuration file using the following commands:

   ```
   raas save_sso_config <filepath>
   ```

After you've configured SSO in SaltStack Enterprise, a best practice is to try logging in as a typical user to ensure that the login process works as expected and that roles and permissions are correct.

Sample configuration files

Sample SAML configuration file for Google

Replace the placeholder text in the following sample with the information provided by your identity provider:

```
name: Google
backend: social_core.backends.saml.SAMLAuth
settings:
 base_uri: https://example.com
 saml_sp_entity_id: raas
 saml_org_info:
   en-US:
     name: Name of Your Organization
     displayname: Display Name for Your Organization
     url: https://your-organization.com
 saml_technical_contact:
   givenName: Name of Your Technical Contact
   emailAddress: email@my_technical_contact.com
 saml_support_contact:
   givenName: Name of Your Support Contact
   emailAddress: email@my_support_contact.com
 saml_enabled_idps:
   saml:
     entity_id: https://accounts.google.com/o/your_organization_id
     attr_user_permanent_id: Your organization's permanent ID
     attr_email: email@my_email_with_identity_provider.com
     attr_username: Your organization's username for the identity
provider
     url: https://accounts.google.com/o/saml2/your_organization_id
     x509cert: |
     -----BEGIN CERTIFICATE-----
     Insert certificate block of text here
     -----END CERTIFICATE-----
   saml_sp_private_key: |
   -----BEGIN PRIVATE KEY-----
   Insert private key block of text here
   -----END PRIVATE KEY-----
   saml_sp_public_cert: |
   -----BEGIN CERTIFICATE-----
   Insert certificate block of text here
   -----END CERTIFICATE-----
```

Sample SAML configuration file for Okta

Replace the placeholder text in the following sample with the information provided by your identity provider:

```
name: Okta
backend: social_core.backends.saml.SAMLAuth
settings:
  base_uri: https://example.com
  saml_sp_entity_id: https://example.com/auth/complete/saml
  saml_org_info:
    en-US:
      name: Name of Your Organization
      displayname: Display Name for Your Organization
      url: https://your-organization.com
  saml_technical_contact:
    givenName: Name of Your Technical Contact
    emailAddress: email@my_technical_contact.com
  saml_support_contact:
    givenName: Name of Your Support Contact
    emailAddress: email@my_support_contact.com
  saml_security_config:
    wantAttributeStatement: False
  saml_enabled_idps:
    okta:
      entity_id: http://www.okta.com/your_organization_id
      attr_user_permanent_id: Your organization's permanent ID
      attr_email: email@my_email_with_identity_provider.com
      attr_username: Your organization's username for the identity
provider
      url: https://dev-561137.okta.com/app/your_organization_id
      x509cert: |
        -----BEGIN CERTIFICATE-----
        Insert certificate block of text here
        -----END CERTIFICATE-----
  saml_sp_private_key: |
    -----BEGIN PRIVATE KEY-----
    Insert private key block of text here
    -----END PRIVATE KEY-----
  saml_sp_public_cert: |
    -----BEGIN CERTIFICATE-----
    Insert certificate block of text here
    -----END CERTIFICATE-----
```

Sample OIDC configuration file for Google

Replace the placeholder text in the following sample with the information provided by your identity provider:

```
name: Name of Your Organization
backend: social_core.backends.google_openidconnect.GoogleOpenIdConnect
settings:
 base_uri: example.com
 google_openidconnect_key: your_id.apps.googleusercontent.com
 google_openidconnect_secret: your_secret
```

Updating an SSO configuration

To update a configuration standard from the CLI:

1. Login as a RaaS user:

   ```
   sudo su raas
   ```

2. Navigate to the directory in which you have stored the configuration file. Update the configuration file as necessary.

3. Save the configuration file using the following command:

   ```
   raas save_sso_config <filepath>
   ```

Deleting an SSO configuration

If access to the Enterprise Console is available, it is recommended that you delete an SSO configuration using the UI. However, you can delete an SSO configuration using the SSEAPI if needed.

To delete an SSO configuration, you need to find the slug that is assigned to the configuration that you would like to delete. The slug is a representation of the configuration's name separated by a dash `-` mark with all lowercase letters. For example, the slug might be `name-of-your-organization`. For SAML with Google, the slug is `google`.

1. In the SSEAPI, generate a list of your SSO backends using the following command:

   ```
   client.api.settings.get_sso_backends()
   ```

2. From the list of SSO backends, find the slug for the configuration you want to delete. Then enter the following command, replacing the placeholder text with your configuration slug:

   ```
   client.api.settings.delete_sso_config('slug-for-your-configuration')
   ```

Agentless Windows module

Download

Download the agentless Windows module files on the SaltStack Enterprise website .

Requirements

- English version of Windows
- Windows versions:
  - Windows 7
  - Windows 8.1
  - Windows 10
  - Windows Server 2008 R2
  - Windows Server 2012 R2
  - Windows Server 2016
- Powershell 3.0 or later
- WinRM must be configured and running
- The `/etc/salt/roster` file must have a configuration section for every Windows machine you want to connect to. The configuration must have a local admin user and password for each machine, as in the following example.

```
win2012dev:   # Minion ID
  host: <IP address>
  user: <local Windows admin username>
  passwd: <password for the admin user>
  winrm: True
```

> Domain credentials are not supported.

- Python 2 must be installed on the Salt Master. The `salt-ssh` module for Windows is supported only on Python 2, not later versions.
- pip 2 must be installed.
  - CentOS 7

```
$ yum install epel-release -y
$ yum install python-pip
$ pip install -U setuptools
```

  - Ubuntu 18.04

```
$ apt-get install python-pip
```

Installing the agentless Windows module

1. Use pip to install the `whl` file.

```
$ pip install -U ./saltwinshell-2017.7-cp27-cp27mu-linux_x86_64.whl
```

2. Edit `/etc/salt/roster` with your minion information.

```
testwin:   # Minion ID
  host: <IP address>
  user: <local Windows admin username>
  passwd: <password for the admin user>
  winrm: True
```

You can now run any Salt SSH command on the Windows server, such as the following:

```
$ salt-ssh testwin disk.usage
```

SecOps Compliance Custom Content SDK

Overview

This page includes the SecOps Compliance Custom Content SDK download available for a range of Operating Systems. The current version is 6.1.0.

Downloads

- Amazon Linux
- CentOS 7
- Debian
- Oracle
- OS X
- Ubuntu 18.04
- Windows

How to use the SDK

- secops_sdk-readme.md

The SDK README includes detailed instructions on how to use the SecOps Compliance Custom Content SDK.