SaltStack Enterprise installation

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

SaltStack Enterprise version 6.0.1
Document version 2
Updated: June 6, 2019

Table of Contents

Prerequisites

SaltStack Enterprise integrates seamlessly with a new or existing Salt installation.

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 SP4
- SUSE Linux Enterprise Server 15
- openSUSE Leap 42.3

> 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 cannot run on Salt Masters running Python 3. Support for masters running Python 3 is planned for the 4Q2019 release. For best results, use Python 2.7 on your Salt Masters.

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:

Up to 500 (minimum)	Up to 500 (recommended)	500 to 1000 (minimum)	500 to 1000 (recommended)
2 CPU cores*	4 CPU cores	6 CPU cores	8 CPU cores
2 GB RAM	8 GB RAM	8 GB RAM	8 GB RAM
At least 20 GB free disk space**	At least 20 GB free disk space**	At least 20 GB free disk space**	At least 20 GB free disk space**

* If using the scheduler you should have at least 4 GB of RAM.

** 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	4 CPU cores 8 GB RAM	8 CPU cores 16 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 9.5 or 9.6 database. PostgreSQL 9.6 is recommended. PostgreSQL 10 is not supported.

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. A PostgreSQL tuning guide can be found here .

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
```

For more on customizing background worker settings, see Background Worker Options .

Known issues

- If the Postgres DB is not set to use UTF-8, sorting will not be consistent across the application.
- Salt-run jobs do not work with the `route_cmd` method.
- Non-scheduled jobs will capture the job execution time in the master time zone. Scheduled jobs will capture job execution time in UTC. To sync these times, the master time zone must be set to UTC or there will be a discrepancy between jobs run at the same time on a schedule vs. from the command line.
- 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.
- Changing the LDAP or AD structure might prevent the SaltStack Enterprise LDAP/AD integration from pulling new users in from the directory. It might also prevent existing users from authenticating.
- `systemctl restart raas` may leave old SaltStack Enterprise processes running and prevent the newly started instance from functioning correctly. For best results, restart with the following instead.

  First stop raas and show all running SaltStack Enterprise processes.

  ```
  systemctl stop raas
  ps guax | grep -E '(raas|celery)' | grep -v grep
  ```

  When all processes have stopped, run `systemctl start raas`.

  > If the `ps | grep` pipeline shows any processes, wait 15-30 seconds for them to finish shutting down and then check again. If the processes remain after two minutes, use the Linux kill command to terminate them.

- If the user who created one or more schedules is deleted, schedules created by that user will stop working.
- 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 *2019.2.0* 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.
- Rejected minion keys cannot be deleted. To delete a rejected minion key, first accept the rejected key and then delete it.
- Logrotate works only on files owned by `root`. However, the SaltStack Enterprise Service (`raas`) installs `/etc/logrotate.d/raas` as owned by `raas`, causing logrotate to ignore the raas log rotation.

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 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

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 sse-installer-6.0.1+3.zip
   $ cd sse_installer
   ```

2. 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.

3. 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`

> 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).

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 sse-installer-6.0.1+3.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 existing Pillar “Top” file

   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 existing Configuration State "Top" file.

   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:':
    - sse.eapi_database

  # SSE Redis Server
  'I@sse_redis_server:':
    - sse.eapi_cache

  # SSE eAPI Servers
  'I@sse_eapi_servers:':
    - sse.eapi_service

  # SSE Salt Masters
  'I@sse_salt_masters:':
    - 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`

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 (6739 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

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

Importing 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
```

Verifying 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 sse-installer-6.0.1+3.zip
```

The output of the command should match the following:

```
2e3873250a23bca7cead50a23c98df3e3b3e45bb8d8d0cc582adba9783a1c839 sse-installer-6.0.1+3.zip
```

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 9.5 or 9.6 database. PostgreSQL 9.6 is recommended. PostgreSQL 10 is not supported.

Download the packages for your environment

Download the packages on the SaltStack Enterprise website.

Red Hat Enterprise Linux 7/CentOS 7

Step 1: PostgreSQL database installation and configuration

1. Install PostgreSQL.

```
# run one of these commands based on your OS
Red Hat
$ sudo wget https://download.postgresql.org/pub/repos/yum/9.6/redhat/rhel-7-x86_64/pgdg-redhat96-9.6-3.noarch.rpm

CentOS
$ sudo wget https://download.postgresql.org/pub/repos/yum/9.6/redhat/rhel-7-x86_64/pgdg-centos96-9.6-3.noarch.rpm

# run all of these commands
$ sudo yum install pgdg-*.noarch.rpm
$ sudo yum update
$ sudo yum install postgresql96-server
$ sudo yum install postgresql96-contrib
$ sudo /usr/pgsql-9.6/bin/postgresql96-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:

```
$ sudo systemctl enable postgresql-9.6
$ sudo systemctl start postgresql-9.6
$ 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-3.5.4-1.*.rpm python35u-3.5.4-1.*.rpm
```

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

```
$ sudo yum install raas-6.0.1+3.el7.x86_64.rpm
```

3. Update RaaS Configuration File.

   `/etc/raas/raas`

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

```
sql:
  dialect: postgresql
  username: salt_eapi
  password: abc123
  host: localhost
  port: 5432
  driver: psycopg2
  ssl: false
```

   To avoid saving passwords in files, use this alternate URL configuration.

```
sql:
  url: ENV
```

   Then in your environment, set the corresponding variable `DATABASE_URL`.
   For example:

```
export DATABASE_URL=postgres://user:secret@localhost:5432/raas_db_name
```

   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. For more on customizing background worker settings, see Background Worker Options .

Configure the location of your Redis server.

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

To avoid saving passwords in files, use this alternate URL configuration.

```
redis:
  url: ENV
```

Then in your environment, set the corresponding variable `REDIS_URL`.
For example:

```
export REDIS_URL=redis://:secret@example.com:6379
```

Redis database numbers are automatically appended to the end of the URL since different databases are used for different purposes (caching, queueing, result storage).

4. Start the Enterprise API service.

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

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

   - 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
     ```

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

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

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

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

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. 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`.

3. 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

sql:
  ssl: True
```

4. Restart the Enterprise API service.

```
$ sudo systemctl restart raas
```

5. Verify the Enterprise API is running.

```
$ sudo systemctl status raas
```

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

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

Install Salt Master plugin

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

```
$ sudo easy_install-2.7 SSEAPE-6.0.1+3-py2.7.egg
```

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_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.
- **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. Restart the Salt Master.

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

After a minute or two the Salt Master and its Minions appear in Enterprise Console.

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 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 for **SLES 12** requires a PostgreSQL 9.5 or 9.6 database. PostgreSQL 9.6 is recommended. PostgreSQL 10 is not supported.
>
> SaltStack Enterprise for **SLES 15** is compatible with a 9.6 database or PostgreSQL 10. PostgreSQL 9.6 is recommended.

Download the packages for your environment

Download the packages on the SaltStack Enterprise website.

SUSE Linux Enterprise Server 12 SP4 / OpenSUSE Leap 42.3

Step 1: PostgreSQL database installation and configuration

1. Install PostgreSQL.

```
$ zypper addrepo
https://download.opensuse.org/repositories/server:/database:/postgresql/openSUSE_Leap_42.3/server:database:postgresq
l.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
```

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

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

Step 2: Redis installation

1. Install Redis.

```
$ zypper addrepo https://download.opensuse.org/repositories/server:database/SLE_12_SP3/server:database.repo
$ zypper refresh
$ zypper install redis
```

2. Start Redis.

   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. Install the SLES 12 RPM.

```
$ zypper install raas-6.0.1+3-0.sles12.x86_64.rpm
```

2. Run the following to allow `raas` to access folders as a non-root user.

```
chown -R raas:raas /etc/raas
chown -R raas:raas /srv/raas
chown -R raas:raas /opt/saltstack
chown -R raas:raas /etc/pki/raas
```

3. Update RaaS Configuration File.

```
sql:
  dialect: postgresql
  username: root
  password: salt
  host: localhost
  port: 5432
  driver: psycopg2
  ssl: false
```

> To avoid saving passwords in files, use this alternate URL configuration.
>
> ```
> sql:
>   url: ENV
> ```
>
> Then in your environment, set the corresponding variable `DATABASE_URL`.
> For example:
>
> ```
> export DATABASE_URL=postgres://user:secret@localhost:5432/raas_db_name
> ```

Configure the location of your Redis server.

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

> To avoid saving passwords in files, use this alternate URL configuration.
>
> ```
> redis:
>   url: ENV
> ```
>
> Then in your environment, set the corresponding variable `REDIS_URL`.
> For example:
>
> ```
> export REDIS_URL=redis://:secret@example.com:6379
> ```
>
> Redis database numbers are automatically appended to the end of the URL since different databases are used for different purposes (caching, queueing, result storage).

```
$ systemctl start raas
```

`raas` will not start. Continue to the next step.

4. Add PostgreSQL Extensions.

```
$ su - postgres
$ psql -d raas_43cab1f4de604ab185b51d883c5c5d09 -c 'CREATE EXTENSION IF NOT EXISTS "pgcrypto”'
```

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`

SUSE Linux Enterprise Server 15

Step 1: PostgreSQL database installation and configuration

1. Install PostgreSQL.

   PostgreSQL 10 is installed as the default on SLES 15.

```
$ zypper in postgresql postgresql-server postgresql-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 root'
```

Step 2: Redis installation

1. Download and install Redis.

```
$ zypper addrepo https://download.opensuse.org/repositories/server:database/openSUSE_Leap_15.0/server:database.repo
$ zypper refresh
$ zypper in redis
```

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

```
$ zypper in raas-6.0.1+3-0.sles15.x86_64.rpm
```

2. Run the following to allow `raas` to access folders as a non-root user.

```
chown -R raas:raas /etc/raas
chown -R raas:raas /srv/raas
chown -R raas:raas /opt/saltstack
chown -R raas:raas /etc/pki/raas
```

3. Update RaaS Configuration File.

   `/etc/raas/raas`

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

```
sql:
  dialect: postgresql
  username: root
  password: salt
  host: localhost
  port: 5432
  driver: psycopg2
  ssl: false
```

   > To avoid saving passwords in files, use this alternate URL configuration.
   >
   > ```
   > sql:
   >   url: ENV
   > ```
   >
   > Then in your environment, set the corresponding variable `DATABASE_URL`.
   > For example:
   >
   > ```
   > export DATABASE_URL=postgres://user:secret@localhost:5432/raas_db_name
   > ```

   Configure the location of your Redis server.

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

   > To avoid saving passwords in files, use this alternate URL configuration.
   >
   > ```
   > redis:
   >   url: ENV
   > ```
   >
   > Then in your environment, set the corresponding variable `REDIS_URL`.
   > For example:
   >
   > ```
   > export REDIS_URL=redis://:secret@example.com:6379
   > ```
   >
   > Redis database numbers are automatically appended to the end of the URL since different databases are used for different purposes (caching, queueing, result storage).

4. Start the Enterprise API service.

```
$ systemctl start raas
# service will not start, this is expected
```

5. Add PostgreSQL Extensions.

```
$ su - postgres
$ psql -d raas_43cab1f4de604ab185b51d883c5c5d09 -c 'CREATE EXTENSION IF NOT EXISTS "pgcrypto"'
```

6. Start the Enterprise API service.

```
$ systemctl start raas
```

7. 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. 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`.

3. 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

sql:
  ssl: True
```

4. Restart the Enterprise API service.

```
$ sudo systemctl restart raas
```

5. Verify the Enterprise API is running.

```
$ sudo systemctl status raas
```

6. Confirm that you can connect to the web console in a web browser.
    - url: `https://your_raas_server/`
    - Username: `root`
    - Password: `salt`

Install Salt Master plugin

1. Log in to your Salt Master.
2. Download one of the Salt Master plugin Egg files based on the version of Python installed on the Salt Master (`python --version` from the command line).
3. Install the plugin (requires Python setuptools).

```
$ sudo easy_install-2.7 SSEAPE-6.0.1+3-py2.7.egg
```

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_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.
- **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. Restart the Salt Master.

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

After a minute or two the Salt Master and its Minions appear in Enterprise Console.

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
```

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 *5.5.1* to version *6.0.1*. 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 *5.5.1*, 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 SaltStack Support Portal .

Download the Python egg

Download the latest Python egg 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

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.

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-6.0.1+3.el7.x86_64.rpm
```

> To download the RPM, see Download the RPM.

6. Restore the backup of the `/etc/raas/raas` config 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. Add new section to /etc/raas/raas:

> SaltStack Enterprise versions *6.0* and later contain the SecOps security modules (additional licensing required). The following configuration options in /etc/raas/raas are specific to this module.

```
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. 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
```

10. Start the Enterprise API service.

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

On each connected Salt Master

1. Stop the `salt-master` service.

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

2. Delete the SSEAPI Python module:

   Delete the prior version of the SSEAPE module (this is the SaltStack Enterprise plugin for the Salt Master). For example:

```
RHEL:
$ sudo rm -rf /usr/lib/python2.7/site-packages/SSEAPE*

Ubuntu:
$ sudo rm /usr/lib/python2.7/dist-packages/SSEAPE*
```

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

```
$ sudo easy_install-2.7 SSEAPE-6.0.1+3-py2.7.egg
```

4. Update the eAPI Master paths.

   - Edit `/etc/salt/master.d/eAPIMasterPaths.conf` to reference the path to the new egg 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. Add `- job_completion: {}` to the engines section of `/etc/salt/master.d/raas.conf`.

6. Start the `salt-master` service.

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

Initial configuration

- Create credentials for Salt Masters
- Change the root password
- Enable more accurate presence detection
- Back up critical data

Create credentials for Salt Masters

Each Salt Master is configured to use the superuser account to connect to Enterprise API to simplify initial installation. For increased security, you should generate an Enterprise API account for each Salt Master.

1. Open Enterprise Console and log in using the superuser account.
2. Go to **Menu**  **> System Administration > Local Users**.
3. Create a user account for each Salt Master and add each Salt Master account to the Salt Master system role.
4. On each Salt Master, edit the `/etc/salt/master.d/raas.conf` file and update the `sseapi_username` and `sseapi_password` with the account credentials you created.
5. Restart the Salt Master service.

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

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**  **> System 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 the pillar 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` used when the database was created. If this file is lost, you lose all pillar data values in the Enterprise API file system.
- **/etc/raas/raas** - This file contains 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 SecOps configuration

SaltStack SecOps is a SaltStack Enterprise add-on that provides automated compliance detection and remediation for your infrastructure.

SaltStack SecOps includes a content library that consists of pre-built, industry best-practice security and compliance content, such as CIS.

The content library updates regularly as security standards change. You can configure SecOps content to download (or ingest) automatically (recommended for most standard systems) as security standards change, or you can download content manually.

Manual content ingestion

Download the SaltStack SecOps content.

Air-gapped systems must update SecOps 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. Log in to a `raas` node.
2. Copy the SecOps content tarball to the `raas` node (`tmp` is recommended).

   This content could be delivered by email or any other means.
3. 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}
   ```

Content ingestion for standard systems

For non-air-gapped `raas` systems, SecOps content is downloaded and ingested on a periodic basis based on the configuration.

The SecOps configuration options are located in a `raas` config file `/etc/raas/raas` in the `sec` section as follows.

Option	Description
`stats_snapshot_interval`	How often (in seconds) secops stats will be collected
`compile_stats_interval`	How often (in seconds) secops stats will be compiled
`username`	Username to use when connecting to SaltStack Enterprise to download the most recent SecOps content (default: `secops`)
`content_url`	URL used to download SecOps 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 SecOps content downloads are allowed (default: `True`). Set this to `False` for air gapped systems.

Option	Description
`download_frequency`	How often in seconds will `raas` attempt to download SecOps content (default: `86400` for 24 hours)
`ingest_on_boot`	Should `raas` attempt to download SecOps 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`)

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
```