SaltStack Config Installation Guide

Release 6.4.0

VMware, Inc.

Jan 21, 2021

CONTENTS

This installation guide explains the process for installing SaltStack Config 6.4.0 from the beginning to the end. It is intended for network system administrators with the general knowledge and experience required in that field, such as experience working with Linux and the command line. You do not necessarily need advanced knowledge of Salt or SaltStack Config before installing SaltStack Config.

If at any time you encounter difficulties with the installation that are not addressed by this guide, *Contact Support*.

Quick links

- *Release notes*: The latest release is SaltStack Config 6.4.0.
- *Downloads*: Download the latest release, along with any other SaltStack Config exclusive downloads.
- *Upgrade from a previous version*: Instructions on upgrading an older version of SaltStack Config.
- supported-operating-systems: Information about supported OS types. Also see *Pre-installation planning* for more detailed system requirements.

CHAPTER

ONE

USING THIS GUIDE

The installation process has three main phases:

- Pre-installation
- Installation
- Post-installation

This guide provides content to support each phase, as described in the following sections.

1.1 Pre-installation

During the pre-installation phase, begin by familiarizing yourself with the overall system architecture for SaltStack Config and Salt if needed:

- *SaltStack Config system architecture*
- *Salt system architecture*

Next, read an overview of the installation process:

- *Installation overview*

Then, refer to the pre-installation planning checklist for a series of questions you need to answer about your installation project along with some guidance about how to make key installation decisions:

- *Pre-installation planning*

After you finish planning your installation, you need to install or update Salt and the dependencies required for the SaltStack Config installer:

- *Install or upgrade Salt*

Next, download, transfer, verify, and import all files necessary for your installation scenario:

- *Transfer and import files*

1.2 Installation

During the installation phase, pick one of the two installation scenarios based on your network's infrastructure:

- *Single-node installation*
- *Multi-node installation*

Note: Guidance about which installation path to select is provided on the *Pre-installation planning* page.

1.3 Post-installation

After the core installation process is complete, you need to complete some post-installation steps:

- *Install the license key*
- *Install and configure the Master Plugin*
- *Log in for the first time and change default credentials*
- *Accept the Salt master key and back up data*
- *Set up SSL certificates*
- *SaltStack Comply configuration* (optional)
- *SaltStack Protect configuration* (optional)
- *Set up Splunk integration* (optional)
- *Set up Single Sign-On (SSO)* (optional)

CHAPTER

TWO

ADDITIONAL CONTENT

The rest of the installation guide provides additional guidance as needed, such as information on *Upgrade from a previous version* and *Troubleshooting*. See the side navigation for a complete list of the installation guide contents.

CHAPTER

THREE

INTRODUCTION

3.1 Installation overview

3.1.1 Overview

This page provides a high-level overview of the installation process for each of the installation scenarios.

The installation process has three main phases:

- Pre-installation
- Installation
- Post-installation

The following sections describe what occurs during each phase.

3.1.2 Pre-installation

Before you begin the pre-installation phase, ensure you are roughly familiar with the *SaltStack Config system architecture* and with the *Salt system architecture*.

During the pre-installation phase, you make key decisions as you plan your SaltStack Config installation project. In this phase, you will:

- Decide which installation scenario is best for your network.
- Determine the hardware and software you need for your SaltStack Config installation, such as how many nodes you need to allocate, which operating systems these nodes need, etc.
- Plan any necessary workarounds if your network does not have access to the Internet.
- Decide whether to install Salt beforehand if needed.
- Download and verify the required installation files.

By the end of this phase, ensure that you have requested the necessary nodes and virtual machines (VMs) needed for your installation scenario.

See *Pre-installation planning* for guidance with this phase.

After you have finished planning your installation, there are a few additional pre-installation tasks that must be completed:

- *Install or upgrade Salt*
- *Transfer and import files*

3.1.3 Installation

The SaltStack Config installer supports two core installation scenarios:

- Single-node installation
- Multi-node installation

For guidance on which installation scenario is right for your network, see *Which installation scenario should you use?*

The following sections provide an overview of the installation process for each scenario. Before reading these sections, ensure you are familiar with the core components of the *SaltStack Config system architecture*.

Single-node installation overview

In the single-node installation scenario, you install SaltStack Config on a single node (server) using the SaltStack Config installer. The end goal is to have a Salt master, SaltStack Config, a Redis database, and a PostgreSQL database that all run on the same node.

To run a single-node installation scenario, you will:

- Extract and import the installation files for this scenario.
- Install or update Salt and Python, if needed.
- Run an installation script that will install a master, RaaS, a Redis database, and a PostgreSQL database on one node.

See *Single-node installation* for a step by step guide.

Multi-node-installation overview

In the multi-node installation scenario, you install SaltStack Config on multiple nodes (servers) using the files provided with the SaltStack Config installer. The end goal is to have four nodes, each hosting a different component of the SaltStack Config structure:

- A Salt master
- A PostgreSQL database node
- A Redis database node
- A RaaS node, also known as SaltStack Config

Alternatively, you can run the salt-master service on one node, and combine two or more of the other services on a separate node. Custom architecture requirements may require consultation services.

In the multi-node installation scenario, you run an orchestration highstate designed by VMware. The highstate runs on your master and sets up the multi-node environment. It installs the core SaltStack Config architecture on the three other nodes that will host PostgreSQL, Redis, and RaaS.

Note: Orchestration is one reason why Salt is so powerful as a configuration management tool. Manually installing and configuring every node in your system is very time-consuming, resource intensive, and tedious. Orchestration allows you to define the configuration for a set of minions all at once. For example, you could create an orchestration that automates the process of bringing a new server online by installing all the necessary applications and even notifying certain employees at different stages of the configuration. For more information about how orchestration works, see *Salt system architecture*.

During a multi-node installation scenario, you will:

- Update Salt and Python, if needed.
- Customize the installation orchestration files to point to the nodes that eventually host PostgreSQL, Redis, and RaaS. From an infrastructure as code perspective, you are essentially creating the unique variables for your network that you will pass into the orchestration.
- Run the installation orchestration highstate.

For a step by step guide, see *Multi-node installation*.

3.1.4 Post-installation

After the core installation scenarios are complete, there are a number of post-installation steps, some of which are optional:

- *Install the license key*
- *Install and configure the Master Plugin*
- *Log in for the first time and change default credentials*
- *Accept the Salt master key and back up data*
- *Set up SSL certificates*
- *SaltStack Comply configuration* (optional)
- *SaltStack Protect configuration* (optional)
- *Set up Splunk integration* (optional)
- *Set up Single Sign-On (SSO)* (optional)

3.2 SaltStack Config system architecture

3.2.1 Overview

Most users find it helpful to understand what SaltStack Config is and how it works before they begin the installation process. This page provides a high-level overview of the SaltStack Config system architecture and its different components.

3.2.2 What is SaltStack Config?

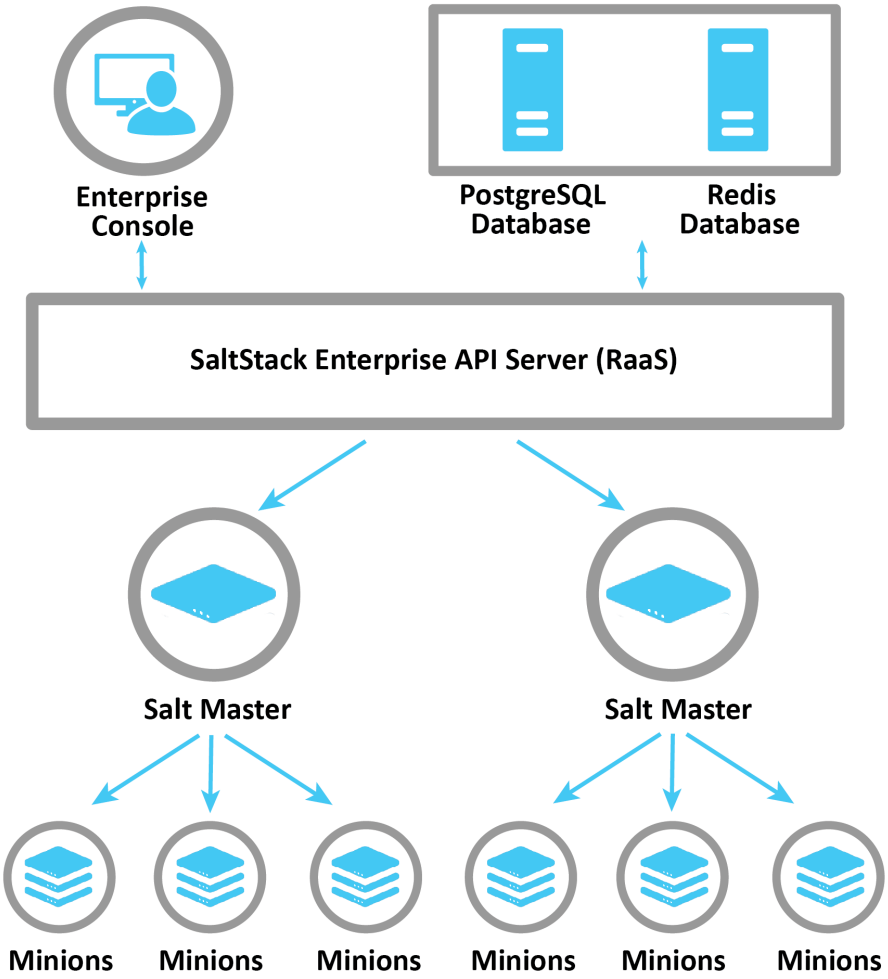
SaltStack Config is powered by Salt, an open-source configuration management and orchestration system. If you are new to Salt and are unfamiliar with how it works, see *Salt system architecture*.

SaltStack Config extends Salt's automated, event-driven configuration management platform by providing additional features, such as:

- **Role-based access controls** - Ensures that network engineers only have access to the resources and jobs that are necessary to fulfill their specific work responsibilities.
- **A user-friendly interface** - In addition to the ability to execute commands from the command line, SaltStack Config also provides a graphical user interface for ease of use.
- **Security automation** - Optional add-ons bringing you automated vulnerability remediation and continuous compliance for hybrid IT systems.

3.2.3 The SaltStack Config system architecture

The following diagram shows the primary components of the basic SaltStack Config architecture that are relevant to installation:



The following sections describe the core components of the SaltStack Config architecture.

Salt masters and the Master Plugin

The master is the main connection between SaltStack Config and the rest of the nodes on your network (the minions). When you issue a command from SaltStack Config (such as a job), the command goes to the master for distribution to the targeted minions.

The Master Plugin is installed on the master. It allows the master to communicate with the SaltStack Config backend server, the RaaS node. The Master Plugin allows the master to access jobs or processes initiated by SaltStack Config, as well as external files and pillar data that are stored on the PostgreSQL database.

The plugin integrates with the existing extension points provided by Salt. For example, job returns are collected using a master-side Salt external job cache, and the RaaS file server uses a Salt fileserver plugin.

Note: You can connect more than one master to SaltStack Config. Each master that connects to SaltStack Config

needs to have the Master Plugin installed.

RaaS

RaaS, which stands for *Returner as a Service*, is the central component in SaltStack Config. In fact, when some people refer to SaltStack Config itself, they are often talking about RaaS.

RaaS provides RPC endpoints to receive management commands from the SaltStack Config user interface, as well as RPC control endpoints to interface with connected masters. All communication is sent using RPC API calls over WebSockets or HTTP(s).

SaltStack Config user interface

The SaltStack Config user interface is a web application that provides the graphical user interface front end for RaaS. Though SaltStack Config is API-first, the user interface interfaces directly with the API (RaaS) to enable simple management of all systems in your environment. Different workspaces provide users with the ability to manage minions, users, roles, jobs, and more.

PostgreSQL Database

RaaS uses a PostgreSQL database to store minion data, job returns, event data, files and pillar data, local user accounts, as well as additional settings for the user interface.

Redis Database

RaaS uses a Redis database to store certain types of data in temporary storage, such as cached data. It also uses temporary data storage to distribute queued work to background workers.

3.2.4 Add-ons

SaltStack Comply and SaltStack Protect are add-ons to SaltStack Config that harness event-driven automation technology to deliver security compliance and vulnerability remediation.

SaltStack Comply

SaltStack Comply is a SaltStack Config add-on that provides automated compliance detection and remediation for your infrastructure. A SaltStack Comply license is required to view and use the SecOps Compliance workspace in the SaltStack Config user interface and through the command line on RaaS.

In contrast with the standalone SaltStack Config architecture, SaltStack Comply includes regularly-updated content and can support custom compliance content. SaltStack Comply provides a library of compliance content. Customers can automatically or manually download new content as it is developed and released by SaltStack.

SaltStack Protect

SaltStack Protect is a SaltStack Config add-on that provides automated vulnerability scanning and remediation for your infrastructure. A SaltStack Protect license is required to view and use the SecOps Vulnerability workspace in the SaltStack Config user interface and through the command line on RaaS.

Similar to the SaltStack Comply content library, SaltStack Protect includes content that is updated as security standards change and new security advisories are released. Customers can automatically or manually download new content as it is released by SaltStack.

3.2.5 Differences in system architecture by installation scenario

SaltStack Config supports two core installation scenarios, which results in two different system architectures.

In the **single-node installation** scenario, a Salt master, SaltStack Config, a Redis database, and a PostgreSQL database all run on the same node.

In the **multi-node installation** scenario, a Salt master, SaltStack Config, a Redis database, and a PostgreSQL database are distributed across at least two nodes. Each service could run on a separate node, or you can combine two or more services on a given node.

It is possible to set up multiple masters or multiple RaaS nodes. High availability requirements may require consultation services.

For more information about the installation scenarios, see *Installation overview*.

3.3 Salt system architecture

3.3.1 Overview

Most users find it helpful to understand what Salt is and how it works before they begin the installation process. This page provides a high-level overview of the Salt system architecture and its different components.

3.3.2 What is Salt?

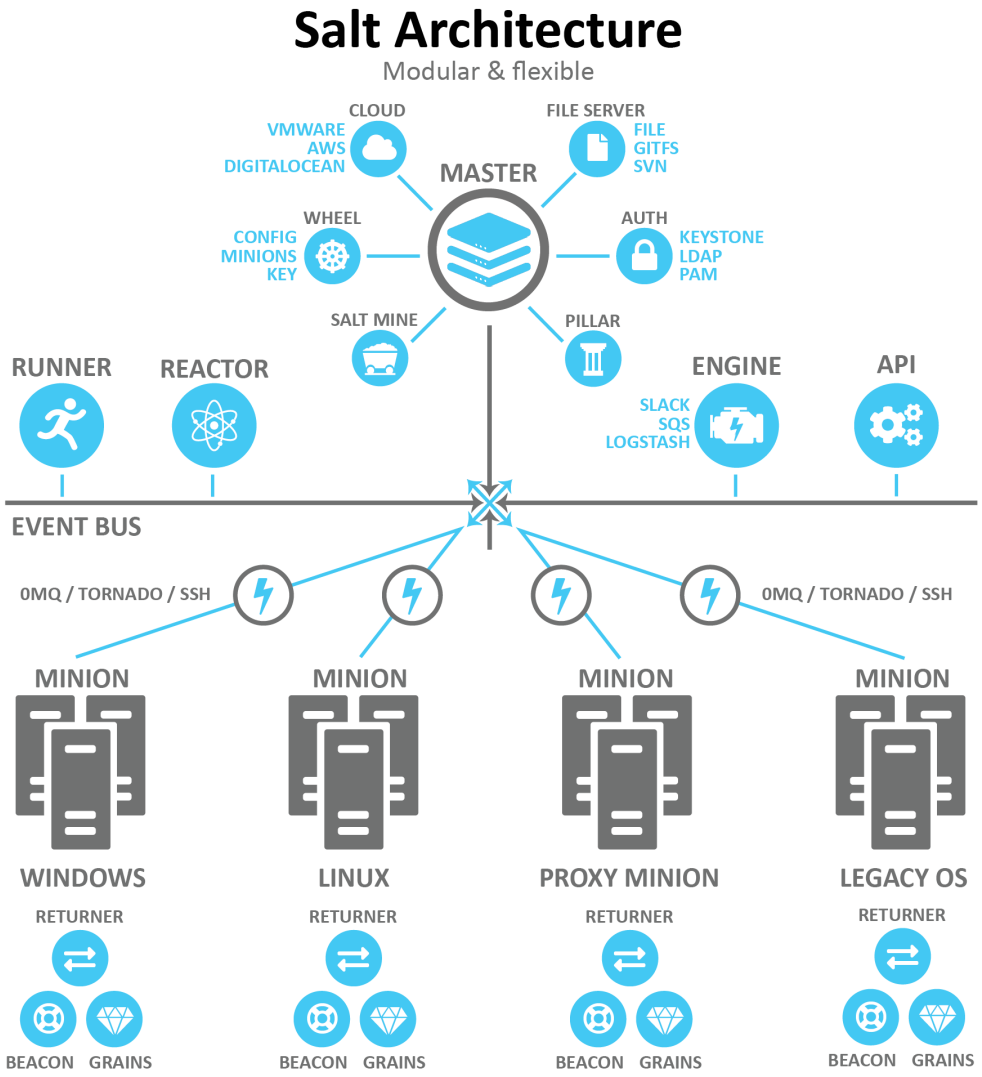
SaltStack Config is powered by Salt, a Python-based open-source remote execution framework used for:

- Configuration management
- Automation
- Provisioning
- Orchestration

Salt is the technology that underlies the core functionality of SaltStack Config. SaltStack Config enhances and extends Salt, providing additional functionality and features that improve ease of use. For a summary of the SaltStack Config infrastructure, see *SaltStack Config system architecture*.

3.3.3 The Salt system architecture

The following diagram shows the primary components of the basic Salt architecture:



The following sections describe some of the core components of the Salt architecture that are relevant to SaltStack Config installation.

Salt masters and Salt minions

Salt uses the master-client model in which a master issues commands to a client and the client executes the command. In the Salt ecosystem, the master is a server that is running the salt-master service. It issues commands to one or more Salt minions, which are nodes that are running the salt-minion service and that are registered with that particular master.

Another way to describe Salt is as a publisher-subscriber model. The master publishes jobs that need to be executed and minions subscribe to those jobs. When a specific job applies to that minion, it executes the job.

When a minion finishes executing a job, it sends job return data back to the master. Salt has two ports used by default for the minions to communicate with their master. These ports work in concert to receive and deliver data

to the Message Bus. Salt's message bus is ZeroMQ, which creates an asynchronous network topology to provide the fastest communication possible.

Targets and grains

The master indicates which minions should execute the job by defining a *target*. A target is the group of minions, across one or many masters, that a job's Salt command applies to.

Note: A master can also be managed like a minion and can be a target if it is running the salt-minion service.

The following is an example of one of the many kinds of commands that a master might issue to a minion. This command indicates that all minions should install the Vim application:

```
salt -v '*' pkg.install vim
```

In this case the glob `'*'` is the target, which indicates that all minions should execute this command. Many other targeting options are available, including targeting a specific minion by its ID or targeting minions by their shared traits or characteristics (called *grains* in Salt).

Salt comes with an interface to derive information about the underlying system. This is called the *grains interface*, because it presents Salt with grains of information. Grains are collected for the operating system, domain name, IP address, kernel, OS type, memory, and many other system properties. You can also create your own custom grain data.

Grain data is relatively static. However, grain data is refreshed when system information changes (such as network settings) or when a new value is assigned to a custom grain.

Open event system (event bus)

The event system is used for inter-process communication between the master and minions. In the event system:

- Events are seen by both the master and minions.
- Events can be monitored and evaluated by both.

The event bus lays the groundwork for orchestration and real-time monitoring.

All minions see jobs and results by subscribing to events published on the event system. Salt uses a pluggable event system with two layers:

- **ZeroMQ (0MQ)** - The current default socket-level library providing a flexible transport layer.
- **Tornado** - Full TCP-based transport layer event system.

One of the greatest strengths of Salt is the speed of execution. The event system's communication bus is more efficient than running a higher-level web service (http). The remote execution system is the component that all components are built upon, allowing for decentralized remote execution to spread load across resources.

Salt states

In addition to remote execution, Salt provides another method for configuring minions by declaring which *state* a minion should be in, otherwise referred to as *Salt states*. Salt states make configuration management possible. You can use Salt states to deploy and manage infrastructure with simple YAML files. Using states, you can automate recursive and predictable tasks by queueing jobs for Salt to implement without needing user input. You can also add more complex conditional logic to state files with Jinja.

To illustrate the subtle differences between remote execution and configuration management, take the command referenced in the previous section about *Targets and grains* in which Salt installed the application Vim on all minions:

Methodology	Implementation	Result
Remote execution	• Run `salt -v '*' pkg.install vim` from the terminal	• Remotely installs Vim on the targeted minions
Configuration management	• Write a YAML state file that checks whether Vim is installed • This state file is then applied to the targeted minions	• Ensures that Vim is always installed on the targeted minions • Salt analyzes the state file and determines what actions need to be taken to ensure the minion complies with the state declarations • If Vim is not installed, it automates the processes to install Vim on the targeted minions

The state file that verifies Vim is installed might look like the following example:

```
# File:/srv/salt/vim_install.sls

install_vim_now:
  pkg.installed:
    - pkgs:
      - vim
```

To apply this state to a minion, you would use the `state.apply` module, such as in the following example:

```
salt '*' state.apply vim_install
```

This command applies the `vim_install` state to all minions.

Formulas are collections of states that work in harmony to configure a minion or application. For example, one state might trigger another state.

The Top file

It is not practical to manually run each state individually targeting specific minions each time. Some environments have hundreds of state files targeting thousands of minions.

Salt offers two features to help with this scaling problem:

- **The top.sls file** - Maps Salt states to their applicable minions.
- **Highstate execution** - Runs all Salt states outlined in `top.sls` in a single execution.

The top file maps which states should be applied to different minions in certain environments. The following is an example of a simple top file:

```
# File: /srv/salt/top.sls

base:
  '*':
    - all_server_setup

  '01webserver':
    - web_server_setup
```

In this example, `base` refers to the Salt environment, which is the default. You can specify more than one environment as needed, such as prod, dev, QA, etc.

Groups of minions are specified under the environment, and states are listed for each set of minions. This top file indicates that a state called `all_server_setup` should be applied to all minions `'*'` and the state called `web_server_setup` should be applied to the `01webserver` minion.

To run the Salt command, you would use the state.highstate function:

```
salt \* state.highstate
```

This command applies the top file to the targeted minions.

Salt pillar

Salt's pillar feature takes data defined on the master and distributes it to minions as needed. Pillar is primarily used to store secrets or other highly sensitive data, such as account credentials, cryptographic keys, or passwords. Pillar is also useful for storing non-secret data that you don't want to place directly in your state files, such as configuration data.

Salt pillar brings data into the cluster from the opposite direction as grains. While grains are data generated from the minion, the pillar is data generated from the master.

Pillars are organized similarly to states in a Pillar state tree, where `top.sls` acts to coordinate pillar data to environments and minions privy to the data. Information transferred using pillar has a dictionary generated for the targeted minion and encrypted with that minion's key for secure data transfer. Pillar data is encrypted on a per-minion basis, which makes it useful for storing sensitive data specific to a particular minion.

Beacons and reactors

The beacon system is a monitoring tool that can listen for a variety of system processes on minions. Beacons can trigger reactors which can then help implement a change or troubleshoot an issue. For example, if a service's response times out, the reactor system can restart the service.

Beacons are used for a variety of purposes, including:

- Automated reporting
- Error log delivery
- Microservice monitoring
- User shell activity
- Resource monitoring

When coupled with reactors, beacons can create automated pre-written responses to infrastructure and application issues. Reactors expand Salt with automated responses using pre-written remediation states.

Reactors can be applied in a variety of scenarios:

- Infrastructure scaling
- Notifying administrators
- Restarting failed applications
- Automatic rollback

When both beacons and reactors are used together , you can create unique states customized to your specific needs.

Salt runners and orchestration

Salt runners are convenience applications executed with the `salt-run` command. Salt runners work similarly to Salt execution modules. However, they execute on the master instead of the minions. A Salt runner can be a simple client call or a complex application.

Salt provides the ability to orchestrate system administrative tasks throughout the enterprise. Orchestration makes it possible to coordinate the activities of multiple machines from a central place. It has the added advantage of being able to control the sequence of when certain configuration events occur. Orchestration states execute on the master using the state runner module.

When you run a multi-node installation, you are actually running an orchestration to install SaltStack Config. In the multi-node installation scenario, you run an orchestration highstate designed by SaltStack. The highstate runs on your master and sets up the multi-node environment. It installs the core SaltStack Config architecture on the three other nodes that will host PostgreSQL, Redis, and RaaS.

CHAPTER

FOUR

PRE-INSTALLATION

4.1 Pre-installation planning

4.1.1 Overview

This page contains the pre-installation planning checklist. It is a series of questions you need to answer and decisions you need to make before you begin your SaltStack Config installation project. This page also provides some guidance about how to answer those questions and make those key decisions.

4.1.2 Prerequisites

The pages in the SaltStack Config installation process are intended to be read and followed in a specific order. Before you begin the installation process, first read the *Installation overview* page.

This page is the first step in the pre-installation process.

4.1.3 Which installation scenario should you use?

The SaltStack Config installer supports two core installation scenarios:

- *Single-node installation*
- *Multi-node installation*

The following sections provide detailed descriptions of these two installation scenarios. As you read the descriptions and decide which installation scenario is appropriate for your network, the key questions to answer are:

- How many nodes does your network have? Will SaltStack Config manage all these nodes?
- Does your network have high availability needs, such as load balancing and automatic failover?
- What is your purpose for installing SaltStack Config? For example, are you installing SaltStack Config as a trial run before deploying to production?

Single-node installation

In the single-node installation scenario, you install SaltStack Config on a single node (server) using the SaltStack Config installer. After installation, a Salt master, the RaaS node, a Redis database, and a PostgreSQL database all run on this same node.

Use the single-node installation scenario if:

- Your network has 1,000 minions or less (nodes that Salt will manage).
- You want to quickly install SaltStack Config and evaluate it first-hand before deploying it to production. (Later when you deploy to production, you can use the multi-node installation.)

The advantages of the single-node installation scenario are:

- It is easy and simple to install.
- It is easy to maintain since SaltStack Config and all of its dependencies are on the same node.

The disadvantages are:

- Single-node installation is not recommended for production grade systems.
- Your SaltStack Config system is reliant on the availability of a single node. If that node goes down, your SaltStack Config ecosystem goes down as well.

Multi-node installation

In the multi-node installation scenario, you install SaltStack Config on multiple nodes (servers) using the SaltStack Config installer. In this installation scenario, the end goal is to have four nodes, each with a different host function. Each node is also a minion to the master:

- A Salt master
- A PostgreSQL database node
- A Redis database node
- A RaaS node, also known as SaltStack Config

In the multi-node installation scenario, you run an orchestration highstate designed by VMware. The highstate runs on your master and sets up the multi-node environment. It installs the core SaltStack Config architecture on the three other nodes that will host PostgreSQL, Redis, and the RaaS node.

Note: It is possible to set up multiple masters or multiple RaaS nodes. It is also possible to run the salt-master service on one node, and combine two or more of the other services on a separate node. High availability or custom architecture requirements may require consultation services.

However, before setting up multiple nodes of the same type, you typically begin with the multi-node installation scenario first and then configure additional architecture later.

Use the multi-node installation scenario if:

- Your network has more than 1,000 nodes (minions that SaltStack Config will manage). Be aware that this scenario is also appropriate for smaller installations as well.
- If you are unsure which installation scenario is best for your system, the multi-node installation is the recommended scenario.

The advantages of the multi-node installation scenario are:

- It can scale as your network grows.

- It is not dependent on the availability of a single node for functionality.
- This installation scenario can support networks with high availability needs, such as load balancing and automatic failover.

The disadvantages are:

- The installation process is more complex, requiring careful planning and thought.
- If your network has high availability needs, you might need support and/or consultation services from SaltStack.

4.1.4 What system architecture do you need?

The system architecture required to install SaltStack Config depends on whether you are using the single-node or multi-node installation scenario. For guidance in selecting an installation scenario, see *Which installation scenario should you use?*.

The following sections describe the requirements for each installation scenario.

Single-node installation requirements

In the single-node installation scenario, you install SaltStack Config on a single node (server) using the SaltStack Config installer. After installation, a Salt master, SaltStack Config, a Redis database, and a PostgreSQL database all run on this same node.

A single-node installation requires:

Hardware	Up to 1,000 nodes (minions)
Cores	8 CPU cores
RAM	16 GB RAM
Disk space	At least 40 GB free space

The disk space is used for minion return data. Increase according to your needs for data retention.

Multi-node installation requirements

In the multi-node installation scenario, install SaltStack Config on multiple nodes (servers) using the SaltStack Config installer. In this installation scenario, the end goal is to have four nodes, each with a different host function:

- A Salt master
- A PostgreSQL database server
- A Redis database server
- A RaaS node, also known as SaltStack Config

Alternatively, you can run the salt-master service on one node, and combine two or more of the other services on a separate node. Custom architecture requirements may require consultation services.

Before beginning a multi-node installation, ensure that you have requested the necessary nodes and virtual machines (VMs) needed for this scenario.

A multi-node installation requires:

Host	1,000 to 2,500 nodes (minions)	2,500 to 5,000 nodes (minions)	Greater than 5,000 nodes (minions)
Salt Master node	4 CPU cores 8 GB RAM	8 CPU cores 16 GB RAM	Consider multiple masters
RaaS node	4 CPU cores 8 GB RAM	8 CPU cores 16 GB RAM	Create an additional SaltStack Config node per 5000 minions, hosted behind your preferred load-balancing solution
Redis node	2 CPU cores 4 GB RAM	4 CPU cores 8 GB RAM	Increase Redis CPU cores and RAM, as indicated by performance
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

The disk space is used for minion return data. Increase according to your needs for data retention.

Note: The Redis and the PostgreSQL hosts need static IP addresses or DNS names and the configuration files need to reference those static IP addresses or DNS names. Depending on how the RaaS node is deployed, it might need a static IP address or DNS name as well. Relying on dynamic IP addresses in configurations can change and break your environment.

4.1.5 Which operating system do you need?

SaltStack Config 6.4.0 is best designed to operate on either:

- RedHat 7.4 or higher (RHEL 7)
- CentOS 7 (CentOS7)

Attention: 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 through a yum update or your server does not have direct Internet access, retrieve the following packages from RedHat 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

SaltStack Config also supports the following operating systems, although they are not recommended:

- Oracle Linux 7
- SUSE Linux Enterprise Server 15 (SLES 15)
- SUSE Linux Enterprise Server 12 (SLES 12)

Important: This list of supported operating systems refers to the RaaS node. It does not refer to the operating systems for the Salt masters in your network. Salt itself is designed to be operating system agnostic and can manage

the nodes of most standard operating systems. For a list of supported Salt master operating systems, see SaltStack Platform Support.

4.1.6 Which version of PostgreSQL do you need?

SaltStack Config requires a PostgreSQL 9.6 database, but PostgreSQL 12.4 is recommended. The recommended version of PostgreSQL is included with the SaltStack Config installer.

PostgreSQL is a third-party open source database that is required for SaltStack Config. Because this is third-party software, be aware of the following:

- You are responsible for ongoing maintenance, backups, and other administrative tasks. For information about PostgreSQL database maintenance and administration, see the PostgreSQL documentation.
- Consider getting guidance from your organization's database administrator, if possible.
- For a SaltStack guide on PostgreSQL tuning, see Tuning your PostgreSQL Server for SaltStack Enterprise.

4.1.7 Does your network have access to the Internet?

Some networks do not have consistent access to the Internet for various reasons. These systems are also referred to as *air-gapped systems.* Air-gapped systems pose particular challenges both for installing SaltStack Config and for ensuring it is up to date.

If you are installing SaltStack Config in an air-gapped system, be aware that the installation process will require greater planning and preparation on the part of you and your organization.

The following section explains a few potential challenges for your consideration as you are planning your installation. For additional advice on how organizations similar to yours have solved these challenges, *Contact Support*.

Plan how to transfer the installation files

In order to complete the installation, you need a mechanism through which to download, verify, and extract the necessary installation files. If downloading files is impossible in your network, you need to brainstorm and prepare an alternate method to transfer the necessary installation files to the nodes on which you are installing SaltStack Config and its dependencies.

You will need to transfer the files to the node(s) involved in the installation process. Place the files in the root folder.

Note: For a single-node installation, transfer the files to the node on which you are installing a Salt master, SaltStack Config, Redis, and PostgreSQL.

For a multi-node installation, transfer the files to the master from which you are running the installation orchestration.

For a list of downloads, see *Downloads*.

Plan how to manage upgrades

SaltStack Config and its dependencies (Salt, PostgreSQL, etc.) release regular updates with enhanced features and security updates. In order to take advantage of these updates, you need to plan to check for updates and install upgrades whenever they are available.

Plan how to update security libraries

Both SaltStack Comply and SaltStack Protect release regular content library updates with the latest compliance and vulnerability content. These content libraries are updated outside of the regular SaltStack Config release schedule.

Ideally, customers can automatically download and ingest security libraries over the Internet or via an http proxy as soon as they are updated. However, it is also possible to *manually* download and ingest these libraries. In order to take advantage of these updates, you need a plan to check for security content updates regularly, and develop a process to manually ingest this content when it is available.

4.1.8 Which version of Salt and Python do you need?

SaltStack Config packages its own Python 3.7. It doesn't use the Python installed on your operating systems and it does not require it to be up to date. However, it is generally recommended that you run the latest version of Python on your system.

SaltStack Config is compatible with most versions of Salt, although it is strongly recommended to run the latest stable versions of Salt on your master.

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

4.1.9 Do you need to install Salt prior to installation?

In an installation context, installing Salt can have two different meanings:

- Installing Salt on the nodes involved in the SaltStack Config installation in either a *Single-node installation* or *Multi-node installation* scenario.
- Installing Salt on the infrastructure that will eventually be managed by SaltStack Config.

Salt is necessary to run the SaltStack Config installation. At a bare minimum, Salt and its dependencies must be installed on the nodes that are involved in either SaltStack Config installation scenario. For instructions about how to install Salt and its dependencies, see *Install or upgrade Salt*.

As for installing Salt on the infrastructure that will eventually be managed by SaltStack Config, installing Salt beforehand is a best practice and is strongly recommended. Installing Salt simplifies and streamlines the process of updating to future versions of Salt. Before you begin your SaltStack Config installation, consider installing Salt on your infrastructure and then monitoring it for a period of time to ensure it is stable and running as expected. For instructions about installing Salt, see *Install or upgrade Salt*.

The one exception to this recommendation is if you are installing SaltStack Config in an air-gapped system, as explained in the following section.

Installing Salt in an air-gapped system

This section explains the trade-offs of installing Salt on your infrastructure in an air-gapped system.

The SaltStack Config installer can install the latest stable version of Salt as it runs. However, the version of Salt that is installed by the SaltStack Config installer is called the *Salt Crystal* package. This package is primarily intended for use in air-gapped systems where it is not possible to update Salt over the Internet. Because it is intended for use in air-gapped systems, the version of Salt in the Salt Crystal package cannot be updated over the Internet and must be manually updated. For information about updating the Salt Crystal package, see Upgrading Salt Crystal.

As the SaltStack Config installer runs in the single-node installation scenario, it detects salt-master service and salt-minion service packages, the SaltStack Config installer skips that step in the installation process. If it does **not** detect Salt, it installs the salt-master service and salt-minion service from the Salt Crystal package.

The inability to update Salt regularly over the Internet could become problematic for your network unless your network is air-gapped. For that reason, it is strongly recommended that you install Salt beforehand rather than using the Salt Crystal package.

4.1.10 Do you need to update Python and Salt prior to installation?

Ensure you have the latest stable version of Salt and that you are running Python 3.5.3 or higher on the node that will host the RaaS node.

It is best to update to the latest version of Salt if possible. For instructions about upgrading Python and Salt, see *Upgrade Salt and Python*.

Warning: Certain Salt dependencies must be installed in order to prevent a failure in either a *Single-node installation* or *Multi-node installation* scenario. To verify that these dependencies are installed, see *Install or upgrade Salt*.

4.1.11 What changes are made to an existing Salt environment?

If your network deployed Salt extensively before you decided to install SaltStack Config, be aware of the following changes that occur to your Salt environment when installing SaltStack Config:

- RaaS 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. However, files that exist in the SaltStack Config user interface will take precedence if they also exist in other file or pillar backends. For information about changing this behavior, see the Configuration page in the SaltStack Config Enterprise Help docs.
- RaaS replaces the Salt master syndic component to provide minion aggregation and scale. Salt Syndic masters are not compatible with the SaltStack Config architecture. Instead, each root master connects directly to RaaS.

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

4.1.12 Which browser does the SaltStack Config user interface need?

The SaltStack Config user interface supports the latest versions of Google Chrome and Mozilla Firefox.

4.1.13 How does licensing for SaltStack Config work?

SaltStack Config requires a license file to track minion usage and duration of contract.

Important: The SaltStack Config download contains a 14-day trial license. After 14 days the RaaS service no longer starts.

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

Before 14 days, your license file must be placed on your RaaS node at `/etc/raas/raas.license` for continued functionality. This step is required as part of the post-installation phase. For more information, see *Install the license key*.

4.1.14 Next steps

Once you have solidified your installation plan, you must complete additional pre-installation steps. The next step is to ensure you have installed or updated Salt and its dependencies. To continue the pre-installation process, *Install or upgrade Salt*.

4.2 Install or upgrade Salt

4.2.1 Overview

Installing or upgrading Salt and Python are required as part of the pre-installation process. This page explains how to:

- Install the dependencies needed for the SaltStack Config installer
- Install Salt and Python
- Upgrade Salt and Python

Regardless of which installation scenario you are using, you need to install or update Salt and the SaltStack Config installer dependencies on all nodes that are involved in the installation scenario. The installation will fail if Salt and the installer's dependencies are not installed on your nodes.

Consider also installing Salt on the infrastructure that will be managed by SaltStack Config before you begin your installation. For an explanation of why you are encouraged to install Salt beforehand, see *Do you need to install Salt prior to installation?*.

4.2.2 Prerequisites

The pages in the SaltStack Config installation process are intended to be read and followed in a specific order. Before reading this page, ensure that you have first read the following pages:

- *Installation overview*
- *Pre-installation planning*

4.2.3 Install the SaltStack Config installer dependencies

The SaltStack Config installer requires a few important packages in order to run correctly:

- OpenSSL
- Extra Packages for Enterprise Linux (EPEL)
- Python cryptography
- Python OpenSSL library

These dependencies must be installed on all nodes that are involved in the installation:

- In a *Single-node installation*, you must install these dependencies on the node that will host the Salt master, the RaaS, a Redis database, and a PostgreSQL database.
- In a *Multi-node installation*, you must install these dependencies on all nodes that will host the Salt master, the RaaS, the Redis database, and the PostgreSQL database.

If you don't install these dependencies, the SaltStack Config installer will fail during either installation scenarios.

To check that these dependencies are present:

1. In the terminal, verify that these dependencies are installed on each node:

```
sudo yum list installed | grep openssl
sudo yum list installed | grep epel-release
sudo yum list installed | grep python36-cryptography
sudo yum list installed | grep python36-pyOpenSSL
```

2. If the dependencies are not present, install the dependencies:

```
sudo yum install openssl
sudo yum install epel-release -y
sudo yum install python36-cryptography
sudo yum install python36-pyOpenSSL
```

> **Danger**: Ensure that you install the `python36-pyOpenSSL` package. It is necessary to configure SSL after installation, but this step must be complete before installation.

After all dependencies are installed, proceed to the next section.

4.2.4 Install Salt

If you are installing SaltStack Config on an existing Salt infrastructure, Salt is already installed. In this case, instead refer to the instructions about how to *Upgrade Salt and Python*.

With that in mind, ensure that Salt is installed on any nodes that are directly involved in your SaltStack Config installation or else the installation will fail. For a description of the different nodes involved in the installation process, see *Which installation scenario should you use?*.

Installing Salt involves three main tasks:

- Install Salt on the Salt master or masters
- Install Salt on the minions
- Accept the minion keys on the master or masters

The following sections explain how to do these tasks.

Note: If you are running a *Single-node installation* scenario, you only need to install Salt on the master. The other two steps can be skipped.

Install Salt on the Salt master(s)

In both a single-node and multi-node installation, you need to install both the salt-master service and the salt-minion service on the Salt master host:

- In a *Single-node installation* scenario, the master is the node on which you will install all the SaltStack Config architectural components (master, RaaS, a Redis database, and a PostgreSQL database).
- In a *Multi-node installation* scenario, this is the node that will become the master or masters.

To install Salt on the master node:

1. Install the Salt project repository and key:

```
sudo yum install https://repo.saltstack.com/py3/redhat/salt-py3-repo-latest.el7.
↪noarch.rpm
```

Note: This script installs the latest Salt release on Redhat/Centos 7 PY3. If your machine is running a different operating system or version of Python, the script will not work. For information about installing Salt on other operating systems or Python versions, see the SaltStack Package Repo.

---

2. Clear the cache:

```
sudo yum clean expire-cache
```

3. Install the salt-master service and the salt-minion service on the master node:

```
sudo yum install salt-master
sudo yum install salt-minion
```

> **Warning:** During a single-node installation, if either the master or the minion services are installed, but not both, the SaltStack Config installation script will terminate. The script terminates as a safeguard to prevent the user from accidentally disrupting an existing installation.
>
> For that reason, ensure that you install both the master and the minion services.

4. Edit the `master.conf` file in the `/etc/salt/minion` directory. In this file, set the master's IP address to point to itself:

```
master: localhost
```

5. Start the salt-master service and salt-minion service:

```
sudo systemctl start salt-master
sudo systemctl enable salt-minion
sudo systemctl start salt-minion
```

> **Note:** Use `service salt-minion restart` to restart the minions if needed.

If you are running a *Single-node installation* scenario, no further steps are needed. For more information, see *Next steps*.

If you are running a *Multi-node installation* scenario, proceed to the next section.

Install Salt on the Salt minions

This section is only required for multi-node installations. After installing Salt on the master as described in the previous section, the next step is to install the salt-minion service (not the salt-master service) on the three nodes that will become the RaaS, a Redis database, and a PostgreSQL database.

Then, you need to configure the minions to communicate with the master. For more detailed information about installing the salt-minion service, see Minion Configuration in the Salt documentation.

To install the salt-minion service:

1. Install only the salt-minion service by running the following command:

```
sudo yum install salt-minion
```

2. Answer `y` to all prompts to accept all changes.
3. Configure each minion to communicate with the master by editing the `master.conf` file in the `/etc/salt/minion` directory. In this file, provide the master's IP address. For example:

```
master: 192.0.2.1
```

4. Start the salt-minion service:

```
sudo systemctl enable salt-minion
sudo systemctl start salt-minion
```

> **Note:** Use `service salt-minion restart` to restart the minions if needed.

5. Repeat the previous steps for all remaining nodes.

After successfully installing the salt-minion service on all nodes, proceed to the next section.

Accept the minion keys on the master(s)

This section is only needed for multi-node installations. At this point, you have installed the salt-master service and salt-minion service, and you have provided your minions with the master's IP address. Now, in order for the master to send commands to the minions, the next step to accept the minion keys on the master.

Before proceeding:

- Ensure the salt-master service is enabled and started (see the final step in the *Install Salt on the Salt Master(s)* section if needed).
- Ensure the minion is enabled and started on all the nodes (see the final step in the *Install Salt on the Salt Minions* section if needed).

To accept the keys:

1. In the master's terminal, list all the keys that are on master:

```
salt-key -L
```

2. Check that all the minion IDs are listed in `Unaccepted keys`.

Note: If they appear in `Accepted keys`, no further action is needed as this is the end goal.

---

3. Accept each minion ID using the following command, replacing the <your-minion-id> with the ID from your network:

```
salt-key -a <your-minion-ID>
```

Note: Running `salt-key -A` accepts all keys.

---

4. Answer `y` to all prompts.
5. Run the `salt-key -L` command a second time to confirm all minions appear in `Accepted keys`.

After all minion keys are accepted, you have successfully installed Salt and can proceed to the next pre-installation step. For more information, see *Next steps*.

4.2.5 Upgrade Salt and Python

To upgrade Salt and Python to the latest stable versions on RedHat and CentOS:

1. In the terminal, check whether Python 3 is running on this node:

```
python3 --version
```

2. If needed, install or update to Python 3 using the following command:

```
sudo yum upgrade python3
```

3. Update Salt by running the following commands:

```
sudo yum install https://repo.ius.io/ius-release-el7.rpm https://dl.fedoraproject.
↪org/pub/epel/epel-release-latest-7.noarch.rpm
sudo yum install python36-pyOpenSSL
```

4. Restart all upgraded services:

```
sudo systemctl restart salt-minion
```

For information about upgrading Salt on other operating systems, see the SaltStack Package Repo.

4.2.6 Next steps

Once you have installed or updated Salt and its dependencies, you must complete additional pre-installation steps. The next step is to download, verify, and transfer the installation files for your installation scenario. To continue the pre-installation process, see *Transfer and import files*.

4.3 Transfer and import files

4.3.1 Overview

The final pre-installation step is to download the installation files and send them to the Salt master node. This page explains how to:

- Download installation files
- Transfer the files
- Verify that the installation files are valid
- Extract the files (optional)
- Import the .asc keyfiles

4.3.2 Prerequisites

This step is part of the pre-installation process. Before reading this page, ensure that you have first read the following pages:

- *Installation overview*
- *Pre-installation planning*
- *Install or upgrade Salt*

4.3.3 Download files for your installation scenario

Download the files for your installation scenario from the *Downloads* page. If you're not sure which installation scenario you need, see *Which installation scenario should you use?*.

On the Downloads page, the files needed for the *Single-node installation* and *Multi-node installation* are listed in the sse-install-downloads section. Both installation scenarios require the same files.

The Automated Installation table contains both a .zip file and a tarball file. You only need one of the files, not both. The tarball is recommended if your nodes do not have an application to unzip the files.

After you've downloaded the files, proceed to the next section to transfer the files.

4.3.4 Transfer the files

You will need to transfer the files to the master node. Place the files in the root folder.

Note: For a single-node installation, transfer the files to the node on which you are installing a master, SaltStack Config, Redis, and PostgreSQL.

For a multi-node installation, transfer the files to the master from which you are running the installation orchestration.

4.3.5 Verify the installation files

As a best practice, validate that the downloaded file was not altered after being created by SaltStack. You can validate the file by comparing the SHA-256 hash for your copy against the SHA-256 listed for that file in the downloads table.

To verify the installation files on RedHat or CentOS:

1. On the machine containing the files, open a terminal and navigate to the directory that contains the files.
2. If needed, use `ls` to list the exact file names.
3. Enter the following command, replacing the exact name of the file you want to verify:

   ```
   sha256sum file-name.zip
   ```

4. The previous command returns the SHA-256 for the file. Compare the output of this command to the SHA-256 listed for that file in the *Downloads* table.

After verifying that the file is legitimate, proceed to the next section.

4.3.6 Extract the files

Extracting the installation files is an optional step. It is only required if you downloaded the .zip version of the SaltStack Config installer rather than the tarball.

Note: If the machine doesn't have this application or if it is air-gapped, download the tarball instead.

To extract the files on RedHat or CentOS:

1. In the terminal, install an application to extract the .zip file. For example:

   Note: sudo yum install unzip

2. Once the unzip tool is installed, enter the following command, replacing the exact file name of the installation file:

```
unzip SaltStack_Enterprise-<version>_Installer.zip
```

After you have extracted the files, proceed to the next section.

4.3.7 Import the .asc keyfiles

You need to import the .asc keyfiles from the SaltStack Config installer into the RPM packaging system. To import the keyfiles on the nodes where you intend to install SaltStack Config components:

1. Navigate to the `sse-installer` directory.
2. To import the `.asc` keyfiles you extracted from the installer .zip file into the RPM packaging system, run the following command:

```
sudo rpmkeys --import keys/*.asc
```

3. Repeat these steps for all nodes.

4.3.8 Next steps

Once you have downloaded, transferred, verified, and imported the installation files, you can begin the installation process. Proceed to either the *Single-node installation* or the *Multi-node installation* scenario.

4.4 Downloads

4.4.1 Overview

For the latest downloads, see SaltStack Enterprise Downloads

The website provides the files needed to install or upgrade SaltStack Config. It also includes the content for SaltStack Comply and SaltStack Protect, the SecOps Compliance Custom Content SDK, and additional files.

4.4.2 Prerequisites

The files you need depend on your specific installation scenario. Before you begin the installation process, ensure that you have first read the following pre-installation topics:

- *Installation overview*
- *Pre-installation planning*
- *Install or upgrade Salt*
- *Transfer and import files*

CHAPTER

FIVE

INSTALLATION SCENARIOS

5.1 Single-node installation

5.1.1 Overview

This page explains how to install SaltStack Config on a single node (server) using the SaltStack Config installer. After installation, a Salt master, SaltStack Config, a Redis database, and a PostgreSQL database all run on this same node.

As it runs, the SaltStack Config installer:

- Installs Python 3.6 on the node (if it wasn't previously installed).
- Installs Salt and its necessary dependencies (if it wasn't previously installed).
- Makes this server a master.
- Applies the Salt states needed to install SaltStack Config.
- Installs the required versions of PostgreSQL, Redis, and Python Setuptools on the server.

Warning: Do not use the single-node installer to upgrade SaltStack Config. Using the installer for upgrades could potentially destroy your original installation. Follow the upgrade instructions instead. See *Upgrade from a previous version* for more information.

Package key IDs

The SaltStack Config 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 .zip file for easy import on such machines. However, we *strongly recommend* validating that the keys provided by SaltStack match the official ones. For specific instructions, see *Verify the installation files*.

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/keys/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	https://repo.saltstack.com/yum/redhat/7/x86_64/3000/SALTSTACK-GPG-KEY.pub

5.1.2 Prerequisites

Before you begin the installation process, ensure you have read and completed the steps on all pre-installation pages:

- *Installation overview*
- *Pre-installation planning*
- *Install or upgrade Salt*
- *Transfer and import files*

Danger: For a single-node installation, it is especially important to follow all the steps listed on the *Install or upgrade Salt* page. The exception is if you are installing SaltStack Config in an air-gapped environment. For more information, see *Do you need to install Salt prior to installation?*.

5.1.3 Run installation script

After completing the steps listed in the previous sections, you can now run the installer on your node:

1. In the terminal, run the command:

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

2. As the script runs, verify that your terminal displays the message:

   ```
   Installing SaltStack Enterprise...
   ```

   While installing, the terminal may display this message for several minutes.

   As this script runs, it installs the latest stable version of Python and Salt if they have not already been installed. It also configures this node as a master and minion.

   Note: If both the salt-master service and salt-minion service are installed, the script skips this step and proceeds with the setup of SaltStack Config.

   If either the salt-master service or the salt-minion service packages are installed, but not both, the script will terminate. The script terminates as a safeguard to prevent the user from accidentally disrupting an existing installation.

   After installing Python and Salt, the script installs:

   - A PostgreSQL database
   - A Redis database

- RaaS, also known as SaltStack Config

If you encounter an error while running the installer, refer to the *Troubleshooting* page or *Contact Support*.

5.1.4 Firewall permissions

For single-node installations:

- The `setup_single_node.sh` script on the installer does not modify firewall rules.
- Ensure that access is allowed to port 443 in your firewall rules for all appropriate systems (masters, web-based interface users, remote systems calling the API (RaaS), etc).

5.1.5 Next steps

Once the single-node installation process is complete, you must complete several post-installation steps:

- *Install the license key*
- *Install and configure the Master Plugin*
- *Log in for the first time and change default credentials*
- *Accept the Salt master key and back up data*
- *Set up SSL certificates*
- *SaltStack Comply configuration* (optional)
- *SaltStack Protect configuration* (optional)
- *Set up Splunk integration* (optional)
- *Set up Single Sign-On (SSO)* (optional)

The first post-installation step is to install the license key. To begin the next post-installation step, see *Install the license key*.

5.2 Multi-node installation

5.2.1 Overview

This page explains how to install SaltStack Config on multiple nodes (servers) using the SaltStack Config installer. In this installation scenario, the end goal is to have four nodes, each with a different host function. Each node is also a minion to the master:

- A Salt master node
- A PostgreSQL database node
- A Redis database node
- A RaaS node, also known as the SaltStack Config node

In the multi-node installation scenario, you run an orchestration highstate designed by SaltStack. The highstate runs on your master and sets up the multi-node environment. It installs the core SaltStack Config architecture on the three other nodes that will host PostgreSQL, Redis, and RaaS. For more information about how orchestration works, see *Salt system architecture*.

Note: It is possible to set up multiple masters or multiple RaaS nodes. It is also possible to run the salt-master service on one node, and combine two or more of the other services on a separate node. The steps to configure this kind of system architecture are not fully explained on this page. High availability or custom architecture requirements may require consultation services.

However, before setting up multiple nodes of the same type, you typically begin with the multi-node installation scenario first and then configure additional architecture later.

5.2.2 Prerequisites

Before you begin the installation process, ensure you have read and completed the steps in all pre-installation pages:

- *Installation overview*
- *Pre-installation planning*
- *Install or upgrade Salt*
- *Transfer and import files*

Danger: For a multi-node installation, it is especially important to follow all the steps listed in the *Install or upgrade Salt* page. In particular, you **must** install the dependencies needed for the SaltStack Config installer on all four nodes in the installation. Otherwise, the multi-node installation will fail. Remediating a failed multi-node installation may require in-depth troubleshooting. If you need assistance, *Contact Support*.

5.2.3 Record key data about the four nodes

Before beginning the multi-node installation, record the following key data about each of the four nodes involved in the installation:

- The IP addresses or DNS names
- The minion IDs

Make sure that you clearly indicate which IP address and minion ID belongs to which host (the master node, the RaaS node, the PostgreSQL database node, the Redis database node).

As a best practice, verify that your IP addresses or DNS names are correct as incorrect IP addresses or DNS names can cause a multi-node installation failure.

Keep this data in an easily accessible record for your own reference. As you configure the orchestration, you need to input this data into several settings and variables in the configuration files. For that reason, it's helpful to keep this record on hand throughout the multi-node installation.

Note: If you are in a virtualized environment, take care to specify the *internal* address, as opposed to the *public* address.

Static vs. dynamic IP addresses

The Redis and the PostgreSQL hosts need static IP addresses or DNS names and the configuration files need to reference those static IP addresses or DNS names. Depending on how the RaaS node is deployed, it might need a static IP address or DNS name as well. Relying on dynamic IP addresses in configurations can change and break your environment.

Setting a custom minion ID (optional)

A minion ID is a unique name given to each minion that is managed by a master. By default, the minion identifies itself to the master by the system's hostname. However, you can assign custom IDs that are descriptive of their function or location within your network.

If you decide to customize your minion IDs, try to keep the ID brief but descriptive of its role. For example, you could use `apache-server-1` to name one of your web servers or you could use `datacenter-3-rack-2` after its location in a datacenter. The goal is to make the names descriptive and helpful for future reference.

To declare a minion ID:

1. In the minion's terminal, navigate to the directory that contains the minion's `id.conf` file. By default, the directory location is `etc/salt/minion.d/id.conf`.
2. Open the `id.conf` file in an editor. Change the `id` setting to your preferred minion ID. For example:

   ```
   id: postgres-database-1
   ```

3. After changing a minion ID, the minion's keys need to be accepted (or re-accepted) by the master. For specific instructions on setting up the keys, see *Accept the minion keys on the master(s)*.

5.2.4 Copy and edit the top state files

In this step, you copy the orchestration files provided with the SaltStack Config installer to the master node. Then, you edit the files to reference the three nodes for RaaS, the Redis database, and the PostgreSQL database.

Note: If the SaltStack Config files are not installed on your master, follow the instructions in *Transfer and import files*.

To copy and edit the orchestration configuration files:

1. On the master, navigate to the `sse-installer` directory.
2. Copy the pillar and state files from the `sse_installer` directory into the minion's `pillar_roots` and `file_roots` using the following commands:

   ```
   sudo mkdir /srv/salt
   sudo cp -r salt/sse /srv/salt/
   sudo mkdir /srv/pillar
   sudo cp -r pillar/sse /srv/pillar/
   sudo cp -r pillar/top.sls /srv/pillar/
   sudo cp -r salt/top.sls /srv/salt/
   ```

   Warning: These instructions make some assumptions that might not be true of your directory structure, especially if you have an existing Salt installation. The instructions assume:

- That your master is using the default directory structure. If your directory structure has been modified, you may need to modify these instructions for your custom directory structure.
- That you do not already have a folder named `sse` under either your pillar or configuration state root. If this folder exists, you may need to merge them manually.
- That you do not already have a file named `top.sls` inside your pillar or salt directory. If this file exists, you may need to merge it with your existing file manually.

3. In the `/srv/pillar/` directory, you now have a file named `top.sls` that you copied over from the installation files in the previous step. Open this file in an editor.
4. Edit this file to define the list of minion IDs (not the IP addresses or DNS names) for your PostgreSQL, Redis, RaaS, and master. Use the IDs that you recorded earlier as you worked through the *Record key data about the four nodes* step.

   For example:

```
{# Pillar Top File #}

{# Define SSE Servers #}

{% load_yaml as sse_servers %}
  - postgres-database-1
  - redis-database-1
  - saltstack-enterprise-api-server-1
  - saltmaster-1
{% endload %}

base:

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

5. In the `/srv/salt/` directory, you now have a file named `top.sls` that you copied over in step 2. Open this file in an editor and verify that it matches the following:

```
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 }}':
```

(continues on next page)

(continued from previous page)

```
    - sse.eapi_plugin
```

After editing the top state files, proceed to the next step.

5.2.5 Edit the SaltStack Config settings pillar file

In this step, you edit five different sections in the SaltStack Config settings pillar mapping file to provide the values that are appropriate for your environment. These settings will be used by the configuration state files to deploy and manage your SaltStack Config deployment.

To copy and edit the SaltStack Config settings state file:

1. On the master, navigate to the `/srv/pillar/sse/` directory.
2. Open the `sse_settings.yaml` file in an editor. **Section 1** of this file contains four variables that correspond to the four nodes. Change the values of the four variables to the minion IDs (not the IP addresses or DNS names) for the corresponding nodes. Use the minion IDs that you recorded earlier as you worked through the *Record key data about the four nodes* step.

   For example:

```
# PostgreSQL Server (Single value)
pg_server: postgres-database-1

# Redis Server (Single value)
redis_server: redis-database-1

# SaltStack Enterprise Servers (List one or more)
eapi_servers:
  - saltstack-enterprise-api-server-1

# Salt Masters (List one or more)
salt_masters:
  - saltmaster-1
```

   Note: The `pg_server` and `redis_server` variables are single variables because most network configurations only have one PostgreSQL and Redis database. By contrast, the variables for the `eapi_servers` and `salt-masters` are formatted in a list because it is possible to have more than one RaaS node and master.

3. In **Section 2** of this file, edit the variables to specify the endpoint and port of your PostgreSQL node:
   - `pg_endpoint` - Change the value to the IP address or DNS name (not the minion ID) of your PostgreSQL server. If you are in a virtualized environment, take care to specify the *internal* address, as opposed to the *public* address.
   - `pg_port` - The standard PostgreSQL port is provided, but may be overridden, if needed.
   - `pg_username` and `pg_password` - Enter the credentials for the user that the API (RaaS) will use to authenticate to PostgreSQL. This user is created when you run the configuration orchestration highstate.

   Note: The variable 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, exclude the action. Do not apply the highstate to the PostgreSQL server during the *Apply the highstates to the nodes* step later in the process.

4. Repeat the previous step to edit **Section 3** of this file, but instead edit the corresponding variables to specify the endpoint and port of your Redis node.
5. In **Section 4** of this file, edit the variables related to the RaaS node:
   - If this is a fresh installation, **do not change** the default values for the `eapi_username` and `eapi_password` variables. During the configuration orchestration, the installation process establishes the database with these default credentials. It needs these credentials to connect through the eAPI service to establish your default Targets and Jobs in SaltStack Config. You will change the default password in a later post-installation step.
   - For the `eapi_endpoint` variable, change the value to the IP address or DNS (not the minion ID) of your RaaS node.

     Note: The variable is specified as the `eapi_endpoint` as some installations host multiple eAPI servers behind a load balancer.

   - The `eapi_ssl_enabled` variable is set to `True` by default. When set to `True`, SSL is enabled. You are **strongly recommended** to leave this enabled. 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` variable is set to `False` by default. This variable provides direction to the configuration states if Pillar data is being used in a single-node installation scenario. In that scenario, all IP communication would be directed to the loopback address. In the multi-installation scenario, you should leave this set to `False`.
   - The `eapi_failover_master` variable is set to `False` by default. This variable supports deployments where masters (and minions) are operating in failover mode.
   - The `eapi_key` variable defines the encryption key that SaltStack Config 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 in a separate terminal outside of the editor:

     ```
     openssl rand -hex 32
     ```

6. In **Section 5** of this file, edit the variables to add your unique customer identifiers:
   - The `customer_id` variable uniquely identifies a SaltStack deployment. It becomes the suffix of the schema name of the `raas_*` (API (RaaS)) database in PostgreSQL. A default is provided, but a custom key can be generated by running the following command in a separate terminal outside of the editor:

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

   - The `cluster_id` variable defines the ID for a set of masters when it is configured in either Active or Failover Multi-Master mode. This ID prevents minions that are reporting to multiple masters from being reported multiple times within the SaltStack Config.

Save your changes to this file and proceed to the next section.

5.2.6 Apply the highstates to the nodes

In this step, you refresh your system data and run the orchestration that configures all the components of SaltStack Config.

Danger: Before running the highstate, it is especially important to follow all the steps listed on the *Install or upgrade Salt* page. In particular, you **must** install the dependencies needed for the SaltStack Config installer on all four nodes in the installation. Otherwise, the multi-node installation will fail. Remediating a failed multi-node installation may require you to *Contact Support*.

The necessary dependencies are:

- OpenSSL
- Extra Packages for Enterprise Linux (EPEL)
- Python cryptography
- Python OpenSSL library

To apply the highstates:

1. On the master, sync your grains to confirm that the master has the grain data needed for each minion. This step ensures that the pillar data is properly generated for SaltStack Config functionality.

   In the command that syncs the grains, you can target all minions, or you can pass in a list of the specific minion IDs for your nodes (including the master itself) in the brackets. For example:

   Target all minions

   ```
   sudo salt \* saltutil.refresh_grains
   ```

   Target a list of minions

   ```
   sudo salt -L 'salt-master-1,postgres-database-1,redis-database-1,
   ↪saltstack-enterprise-api-server-1' saltutil.refresh_grains
   ```

2. Refresh and confirm that each of the minions has received the pillar data defined in the `sse_settings.yaml` file and that it appears as expected.

   In the command that refreshes the pillar data, you can target all minions or you can pass in a list of the specific minion IDs for your nodes (including the master itself) in the brackets. For example:

   Target all minions

   ```
   sudo salt \* saltutil.refresh_pillar
   ```

   Target a list of minions

   ```
   sudo salt -L 'salt-master-1,postgres-database-1,redis-database-1,
   ↪saltstack-enterprise-api-server-1' saltutil.refresh_pillar
   ```

3. Confirm that the return data for your pillar is correct:

```
sudo salt \* pillar.items
```

   Verify that you see pillar data related to SaltStack Config.

   Note: You could also target a specific minion's pillar data to verify the pillar data has been refreshed.

4. Run the command that applies the orchestration highstate to the PostgreSQL server. Use the minion ID that you recorded for the PostgreSQL server earlier as you worked through the *Record key data about the four nodes* step.

   For example:

```
sudo salt postgres-database-1 state.highstate
```

5. Repeat the previous step for each of the following servers, replacing the minion ID for each server:
   - The Redis node
   - The RaaS node
   - The master node

   Note: During the initial application of the highstate to the master, you may see the following error message: `Authentication error occurred.`

   This error displays because the master has not yet authenticated to the RaaS node, but the Master Plugin installation state will restart the salt-master service and the issue will be resolved automatically.

If you encounter any other errors while running the highstates, refer to the *Troubleshooting* page or *Contact Support*.

5.2.7 Firewall permissions

For multi-node installations, ensure firewall access is allowed on the following ports from the following nodes:

Node	Default Port	Is accessible by
PostgreSQL	5432	eAPI servers
Redis	6379	eAPI Servers
eAPI endpoint	443	• masters • Web-based interface users • Remote systems calling the Enterprise API
Masters	4505/4506	All minions configured to use the related master

5.2.8 Next steps

Once the multi-node installation process is complete, you must complete several post-installation steps:

- *Install the license key*
- *Install and configure the Master Plugin*
- *Log in for the first time and change default credentials*
- *Accept the Salt master key and back up data*
- *Set up SSL certificates*
- *SaltStack Comply configuration* (optional)
- *SaltStack Protect configuration* (optional)
- *Set up Splunk integration* (optional)
- *Set up Single Sign-On (SSO)* (optional)

The first post-installation step is to install the license key. To begin the next post-installation step, see *Install the license key*.

CHAPTER

SIX

POST-INSTALLATION

6.1 Install the license key

6.1.1 Overview

This page explains how to install your license key. When you first install SaltStack Config, it contains a 14-day trial license. To continue using SaltStack Config beyond the 14-day trial, contact Support to purchase a license and then install the key following the instructions below.

6.1.2 Prerequisites

Installing your license key is the first post-installation step in a series of several steps. Before installing your license key, complete one of the installation scenarios then install your license key. To begin the installation process, see *Installation overview*.

6.1.3 Install your license key

When deploying an RaaS node, you will need to add your license key to the `/etc/raas` folder.

Once you have added the license key, set ownership of the file to the `raas` user, as follows.

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

6.1.4 Next steps

After installing your license key, you must complete additional post-installation steps. The next step is to install and configure the Master Plugin. To continue the post-installation process, see *Install and configure the Master Plugin*.

6.2 Install and configure the Master Plugin

6.2.1 Overview

This page explains how to install, configure, and upgrade the Master Plugin. The Master Plugin enables your masters to communicate with SaltStack Config. The Master Plugin includes a variety of settings you can adjust to improve performance, which is particularly useful for large or busy environments. For more information about the Master Plugin and how it fits in SaltStack Config architecture, see *Salt masters and the Master Plugin*.

Typically, you install the Master Plugin on every master in your environment that communicates with SaltStack Config. For example, if you are using a configuration with more than one master (sometimes called a multi-master setup), each master needs to the Master Plugin.

For more information about updating performance-related settings, see the Master Plugin page in the SaltStack Config Help Documentation embedded in the SaltStack Config user interface. For help finding the documentation, see Finding Enterprise Documentation.

6.2.2 Prerequisites

Installing and configuring the Master Plugin is one post-installation step in a series of several steps that should be followed in a specific order. First, complete one of the installation scenarios and then read the *Install the license key* post-installation page.

6.2.3 When do you need to install the Master Plugin?

You need to install the Master Plugin on all of your masters after a fresh installation of SaltStack Config. The Master Plugin is not necessary on masters that do not need to communicate with SaltStack Config.

If you used the *Single-node installation* installation scenario, you do not need to install the Master Plugin on the node on which you installed SaltStack Config and its related architecture. The installer automatically installs the Master Plugin on the master node. However, the Master Plugin is installed *only* on the master where you ran the installer. If you have multiple masters, you still need to install the Master Plugin on your other masters.

If you recently upgraded to a newer version of SaltStack Config, you should also re-install the Master Plugin. For the full instructions on upgrading and installing the Master Plugin after an upgrade, see *Upgrade from a previous version*.

If you are manually installing SaltStack Config (not recommended), you should complete the following before you install the Master Plugin:

- Install and configure the PostgreSQL database
- Install and configure the Redis database
- Enable SSL (optional)

6.2.4 Install the Master Plugin

To install the Master Plugin on your master:

1. Log in to your master.
2. If necessary, download the Master Plugin wheel. See *Downloads*.
3. Install the Master Plugin by manually installing the updated Python wheel. Use the following example commands, replacing the exact name of the wheel file:

   RHEL/CentOS

   Ubuntu

```
sudo pip3 install SSEAPE-file-name.whl --prefix /usr
```

```
sudo pip3 install SSEAPE-file-name.whl
```

Note: Some users might need to alter the syntax to `pip3.6` or `pip36` for their operating systems.

After installing the Master Plugin, proceed to the next section.

6.2.5 Configure the Master Plugin

To configure the master after installing the Master Plugin:

1. Log in to your master and verify the `/etc/salt/master.d` directory exists, or create it.
2. Generate the master configuration settings.

   Danger: If you want to preserve your settings when upgrading your installation, make a backup of your existing Master Plugin configuration file before running this step. Then copy relevant settings from your existing configuration to the newly generated file.

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

   If running this command causes an error, it might be related to the method you used when initially installing Salt. If you installed Salt through the all-in-one installer, your SaltStack Config installation likely includes an offline packager, called the Salt Crystal, that requires special upgrade instructions. For more information, see *Troubleshooting*.

3. Edit the generated `raas.conf` file and update the values as follows to validate the certificate the API (RaaS) users and set its IP address.

Value	Description
`sseapi_ssl_validate_cert`	Validates the certificate the API (RaaS) 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 your RaaS node, for example, `http://example.com`, or `https://example.com` if SSL is enabled.
`sseapi_command_age_limit`	Sets the age (in seconds) after which old, potentially stale jobs are skipped. For example, to skip jobs older than a day, set it to: `sseapi_command_age_limit: 86400` Skipped jobs will continue to exist in the database and display with a status of `Completed` in the SaltStack Config user interface. Some environments may need the master to be offline for long periods of time and will need the master to run any jobs that were queued after it comes back online. If this applies to your environment, set the age limit to `0`.

4. **OPTIONAL**: This step is necessary for manual installations only. To verify you can connect to SSL before connecting the Master Plugin, edit the generated `raas.conf` file to update the following values. If you do not update these values, the Master Plugin uses the default generated certificate.

Value	Description
`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.

5. Restart the salt-master service.

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

6. OPTIONAL: You might want to run a test job to ensure the Master Plugin is now enabling communication between the master and the RaaS node.

```
salt -v '*' test.ping
```

Even if no activity shows, for example because no minions are connected, this is likely a sign of a correct configuration.

6.2.6 Next steps

After installing and configuring the Master Plugin, you must complete additional post-installation steps. The next step is to log in to the SaltStack Config user interface for the first time. To continue the post-installation process, see *Log in for the first time and change default credentials*.

6.3 Log in for the first time and change default credentials

6.3.1 Overview

This page explains how to log into the SaltStack Config user interface for the first time after an initial installation. After logging in for the first time, you need to complete a few additional tasks:

- Change the root password.
- Secure your RaaS credentials.

6.3.2 Prerequisites

Logging into the user interface is one post-installation step in a series of several steps that should be followed in a specific order. First, complete one of the installation scenarios and then read the following post-installation pages:

- *Install the license key*
- *Install and configure the Master Plugin*

6.3.3 Confirm you can log in to SaltStack Config

After installing SaltStack Config and completing the previous post-installation step, confirm you can log in to the user interface using your web browser. Chrome or Firefox is recommended. For more information about supported browsers, see *Which browser does the SaltStack Config user interface need?*

The default installation uses `https://` and generates a self-signed certificate.

The default credentials are as follows, replacing the `url` value with the DNS name or IP address of the RaaS node:

- URL: `https://example.com`
- Username: `root`
- Password: `salt`

6.3.4 Change root password

While still logged in to the user interface, consider changing your root password so that it is no longer the default. This step is important for securing your SaltStack Config installation. To change the password:

1. From the top left navigation bar, click the **Menu** ⋮, then select **Administration** to access the Administration workspace. Click the **Local Users** tab.
2. Click `root` in the side menu to select it.
3. In the **Password** field, type a new password. Enter the password again to confirm it.
4. Click **Save**.

5. Confirm the password has been changed by logging out and back in again.

Once you've changed the root password, proceed to the next section.

6.3.5 Secure your RaaS credentials

The installer sets up default credentials for RaaS. Changing and securing these credentials is an important step. SaltStack Config provides various options for securing your RaaS credentials. For more information about each option and specific directions, see Securing credentials in your SaltStack Enterprise configuration.

6.3.6 Next steps

After logging into the user interface for the first time, you must complete additional post-installation steps. The next step is to accept the master key and back up critical data. To continue the post-installation process, see *Accept the Salt master key and back up data*.

6.4 Accept the Salt master key and back up data

6.4.1 Overview

After you've successfully logged in for the first time, you need to complete some important tasks in the SaltStack Config user interface:

- Accept the Salt master's key.
- Remove the Pillar top file (for multi-node installations only).
- Back up critical data.
- Try some sample content to enable more accurate presence detection and to test the overall system's functionality.

6.4.2 Prerequisites

Accepting the master key is one post-installation step in a series of several steps that should be followed in a specific order. First, complete one of the installation scenarios and then read the following post-installation pages:

- *Install the license key*
- *Install and configure the Master Plugin*
- *Log in for the first time and change default credentials*

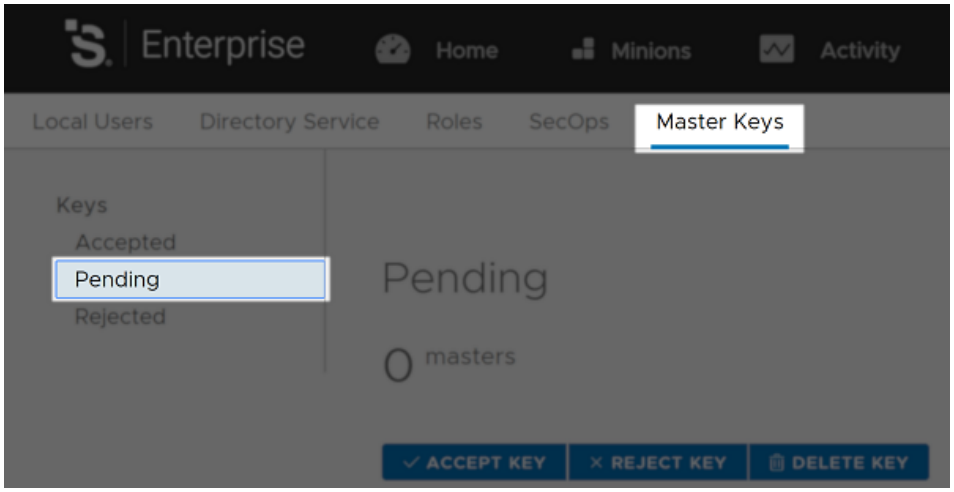
6.4.3 Accept the master's key

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

After installation, you must accept the master's key in the user interface. Until the key is accepted, the master will react slowly as it continually tries to contact the RaaS node.

To accept the master key:

1. Log in to the SaltStack Config user interface.
2. From the top left navigation bar, click the **Menu** ⋮⋮, then select **Administration** to access the Administration workspace. Click the **Master Keys** tab.
3. From the side menu, click **Pending** to show a list of all pending master keys.



4. Check the box next to the master key to select it. Then, click **Accept Key**.
5. After you accept the master key, an alert appears indicating you have pending keys to accept. To accept these minion keys, go to **Minion Keys** > **Pending**.
6. Check the boxes next to your minions to select them. Then, click **Accept Key**.
7. Click **Accept** in the confirmation dialog.

The key is now accepted. After several seconds, the minion appears under the **Accepted** tab, and in the Minions workspace.

After verifying the master key and minion keys have been accepted, proceed to the next section.

6.4.4 Remove the pillar top file

If you installed SaltStack Config using the *Multi-node installation* scenario, you need to remove the pillar top file you created earlier during the installation process. For reference, see step 2 in the section about *Copy and edit the top state files* in *Multi-node installation*.

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

Note: Only remove the pillar top file after successfully logging in to the user interface for the first time.

6.4.5 Back up critical data

If you are not using a complete system backup solution that can restore your entire SaltStack Config 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 Config 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, the RaaS node will not be able to access the database.
- **/etc/raas/raas** - Contains SaltStack Config configuration data.
- **/etc/raas/raas.secconf**- Contains SaltStack Config configuration data.
- **RaaS Database** - Configure regular PostgreSQL database backups for the RaaS database.

6.4.6 Import sample content (optional)

To test the basic functionality of SaltStack Config, try working with some sample content in the user interface. This content is not included as part of the installation process, but it is available when you import it manually from the installation file packages. For specific instructions on importing sample content, see *Import sample content*.

SaltStack Config provides several default targets and jobs along with supporting files and pillar data. Sample job files and pillar data are placed in the `sse` Salt environment so they don't interfere with files and pillar data in the `base` environment. The sample content includes targets, jobs, pillar data, and supporting files.

Samples are used to save time setting up your SaltStack Config environment. With default jobs, you can take advantage of predefined state files and pillar data to begin running frequently-used operations. You might also refer to samples as a model for how different system elements are configured to work together as you build your own workflows.

The following sections give instructions for importing sample content and explain which sample content is recommended for most SaltStack Config installations.

Import sample content

Sample content is not included as part of the manual installation process, but is automatically included if you completed a single-node or multi-node installation using the installer. For manual installations, you can import it manually to your RaaS node from the installation file packages.

To import the sample content:

1. On the master or a computer where you downloaded the installer files, navigate to the `sse-installer/salt/sse/eapi_service/files` directory.
2. Transfer the `sample-resource-types.raas` file to your RaaS node.
3. On your RaaS node, run the following command, replacing the placeholder text with your specific information:

```
/usr/bin/raas-dump --insecure --server <https://raas_server_ip> --auth <username>:
↪<password> --mode import < /tmp/sample-resource-types.raas
```

Note: If you are running this command from the RaaS node, you can substitute `localhost` in place of the server IP address instead.

If you are running this command on SaltStack Config 6.2.0 or earlier, the file path is `/opt/saltstack/raas/venv/bin/raas-dump` instead.

---

4. Log in to the user interface and go to **Elements** > **Jobs** to verify that some of the sample jobs now appear in this workspace.

test.ping

Consider running the `test.ping` command on targeted Salt minions to verify communication is working properly within SaltStack Config.

Enable presence

This job enables more accurate minion presence detection. It's helpful to run enable presence jobs on a regular basis to ensure that your connected minions retain a status of *Present* in the Targets workspace. Presence indicates if SaltStack Config has received any job data from the minion recently, within a defined interval.

SaltStack Config provides a job to install a Salt Beacon that sends periodic heartbeats from each minion. A good practice is to install this job and run it at regular intervals on all minions to enable more accurate presence.

To run this job:

1. Open the user interface and log in using the superuser account.
2. Click **Targets** to access the Targets workspace
3. From the side menu, click the **All Minions** target.
4. Click **Run Job** and select **Enable Presence**.

Additional sample content

For more sample content, see the *Samples* page in the SaltStack Config Help Documentation embedded in the user interface. For help finding the documentation, see Finding Enterprise Documentation.

6.4.7 Next steps

After logging into the user interface for the first time, you must complete additional post-installation steps. The next step is to set up SSL certificates. To continue the post-installation process, see *Set up SSL certificates*.

6.5 Set up SSL certificates

6.5.1 Overview

This page explains how to set up Secure Sockets Layer (SSL) certificates as part of the SaltStack Config post-installation process.

6.5.2 Prerequisites

Setting up the SSL certificates is one post-installation step in a series of several steps that should be followed in a specific order. First, complete one of the installation scenarios and then read the following post-installation pages:

- *Install the license key*
- *Install and configure the Master Plugin*
- *Log in for the first time and change default credentials*
- *Accept the Salt master key and back up data*

6.5.3 How to set up SSL certificates

Setting up SSL certificates is optional when installing SaltStack Config, but recommended.

To create the SSL certificates:

1. The `python36-pyOpenSSL` package is necessary to configure SSL after installation. This step is usually completed before installation. If you were unable to install it before installation, it can be installed now. For instructions about checking for and installing this dependency, see *Install or upgrade Salt*.
2. Create and set permissions for the certificate folder for the RaaS service.

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

3. Generate keys for the RaaS service 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. To enable SSL connections to SaltStack Config user interface, generate a PEM-encoded SSL certificate or ensure that you have access to an existing PEM-encoded certificate.
5. Save the `.crt` and `.key` files you generated in the previous step to `/etc/pki/raas/certs` on the RaaS node.
6. Update the RaaS service configuration by opening `/etc/raas/raas` in a text editor. Configure the following values, replacing `<filename>` with your SSL certificate filename:

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

7. Restart the RaaS service.

```
sudo systemctl restart raas
```

8. Verify the RaaS service is running.

```
sudo systemctl status raas
```

9. Confirm that you can connect to the user interface in a web browser by navigating to your organization's custom SaltStack Config URL and entering your credentials. For more information about logging in, see *Log in for the first time and change default credentials*.

Your SSL certificates for SaltStack Config are now set up.

6.5.4 Updating SSL certificates

Instructions for updating SSL certificates for SaltStack Config are available at the SaltStack Support knowledge base. For more information, see How to update SSL certificates for SaltStack Enterprise.

6.5.5 Next steps

After setting up SSL certificates, you must complete additional post-installation steps. If you are a SaltStack Comply, and/or SaltStack Protect customer, or if you want to integrate with Splunk, the next step is to set up these services. For more information, see:

- *SaltStack Comply configuration*
- *SaltStack Protect configuration*
- *Set up Splunk integration*

6.6 SaltStack Comply configuration

6.6.1 Overview

SaltStack Comply is an add-on that provides automated compliance detection and remediation for your infrastructure. Its content library consists of industry best-practice security and compliance content, such as CIS.

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

As an alternative, the library includes the option to download content manually, or to access content from the RaaS node via an HTTP(s) proxy. Manual ingestion is useful for air-gapped systems, while downloading via proxy is useful to avoid downloading content directly from the internet. Downloading via proxy also provides more control and visibility into what's being downloaded and where.

6.6.2 Prerequisites

Configuring SaltStack Comply is one post-installation step in a series of several steps that should be followed in a specific order. First, complete one of the installation scenarios and then read the following post-installation pages:

- *Install the license key*
- *Install and configure the Master Plugin*
- *Log in for the first time and change default credentials*
- *Accept the Salt master key and back up data*
- *Set up SSL certificates*

6.6.3 Install Python 3 rpm libraries

SaltStack Comply and SaltStack Protect use the Python 3 rpm libraries to reliably compare package versions. These programs need the increased accuracy provided by these libraries to determine version compliance or assess vulnerabilities.

Currently, any minions using RedHat or CentOS 7 might need the Python 3 rpm libraries in order to run accurate SaltStack Comply or SaltStack Protect assessments. If you intend to run assessments on minions that use these versions of RedHat or CentOS, you need to manually install the Python 3 rpm library on these machines.

Note: Other workarounds are available. If you need an alternate workaround, *Contact Support*.

To install the Python 3 rpm library:

1. Install the EPEL repository using the following command:

```
yum install -y epel-release
```

2. Install the Python 3 rpm library:

```
yum install -y python3-rpm
```

6.6.4 Automatic content ingestion for standard systems

For non-air-gapped RaaS systems, content is downloaded and ingested on a periodic basis as determined by the settings in the configuration file. In SaltStack Config 6.4.0, if you installed SaltStack Config using either a single-node or multi-node installation, automatic content ingestion is already configured and no further action is required.

If you installed SaltStack Config manually, follow these steps to configure automatic SaltStack Comply content ingestion:

1. Add the following to the RaaS service 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 *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.

6.6.5 Ingesting content via http(s) proxy

For ingestion via proxy, you'll need to create an override to the RaaS service and add new environment variables for `http proxy` and `https proxy`.

To configure the RaaS node to use https proxy:

1. Complete the previous steps to enable automatic ingestion.
2. On the master in the command line, edit the RaaS service:

```
systemctl edit raas
```

3. Add the following lines to the generated file.

```
[Service]
Environment="http_proxy=http://<hostname>:234"
Environment="https_proxy=https://<hostname>:234"
Environment="HTTP_PROXY=http://<hostname>:234"
Environment="HTTPS_PROXY=http://<hostname>:234"
```

4. If your proxy requires password authentication, you may need to set this as part of the proxy environment variables. For example:

```
Environment="HTTP_PROXY=http://USER:PASSWORD@<hostname>:234"
```

5. If your proxy uses an internal Certificate Authority, you may also need to set the `REQUESTS_CA_BUNDLE` environment variable to ensure that the proxy is able to use it. For example:

```
Environment="REQUESTS_CA_BUNDLE=/etc/pki/tls/certs/ca-bundle.crt"
```

6. Restart the RaaS service:

```
systemctl restart raas
```

After the service restarts, content begins to download. This may take up to 20 minutes.

6.6.6 Manual content ingestion

Air-gapped API (RaaS) 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 Comply content from the *Downloads page*.
2. Log in to an 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. Ingest the tarball contents.

```
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}
```

6.6.7 Configuration options

The following table describes the configuration options 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 Config to download the most recent SaltStack Comply content (default: `secops`)
`content_url`	URL used to download SaltStack Comply content (default: `enterprise.saltstack.com/docs/downloads.html#saltstack-comply-and-protect-content`)
`ingest_override`	When ingesting new content, overwrite existing benchmarks and checks (default: `True`)
`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 the RaaS service cache dir `/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 the RaaS service attempt to download SaltStack Comply content (default: `86400` for 24 hours)
`ingest_on_boot`	Should the RaaS service 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`)

6.6.8 Next steps

After configuring SaltStack Comply, there may be additional post-installation steps. Check the list of post-installation steps to ensure you have completed all the necessary steps.

6.7 SaltStack Protect configuration

6.7.1 Overview

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.

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

As an alternative, the library includes the option to download content manually, or to access content from the RaaS node via an HTTP(s) proxy. Manual ingestion is useful for air-gapped systems, while downloading via proxy is useful to avoid downloading content directly from the internet. Downloading via proxy also provides more control and visibility into what's being downloaded and where.

6.7.2 Prerequisites

Configuring SaltStack Protect is one post-installation step in a series of several steps that should be followed in a specific order. First, complete one of the installation scenarios and then read the following post-installation pages:

- *Install the license key*
- *Install and configure the Master Plugin*
- *Log in for the first time and change default credentials*
- *Accept the Salt master key and back up data*
- *Set up SSL certificates*

6.7.3 Install Python 3 rpm libraries

SaltStack Comply and SaltStack Protect use the Python 3 rpm libraries to reliably compare package versions. These programs need the increased accuracy provided by these libraries to determine version compliance or assess vulnerabilities.

Currently, any minions using RedHat or CentOS 7 might need the Python 3 rpm libraries in order to run accurate SaltStack Comply or SaltStack Protect assessments. If you intend to run assessments on minions that use these versions of RedHat or CentOS, you need to manually install the Python 3 rpm library on these machines.

Note: Other workarounds are available. If you need an alternate workaround, *Contact Support*.

To install the Python 3 rpm library:

1. Install the EPEL repository using the following command:

```
yum install -y epel-release
```

2. Install the Python 3 rpm library:

```
yum install -y python3-rpm
```

6.7.4 Automatic content ingestion for standard systems

For non-air-gapped RaaS systems, content is downloaded and ingested on a periodic basis as determined by the settings in the configuration file. In SaltStack Config 6.4.0, if you installed SaltStack Config using either a single-node or multi-node installation, automatic content ingestion is already configured and no further action is required.

If you installed SaltStack Config manually, follow these steps to configure automatic SaltStack Protect content ingestion:

1. Add the following section to the RaaS service 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
  tenable_asset_import_enabled: True
  tenable_asset_import_grains: ['fqdn', 'ipv4', 'ipv6', 'hostname', 'mac_address',
↪ 'netbios_name',
                                'bios_uuid', 'manufacturer_tpm_id', 'ssh_
↪fingerprint',
                                'mcafee_epo_guid', 'mcafee_epo_agent_guid',
↪'symantec_ep_hardware_key',
                                'qualys_asset_id', 'qualys_host_id', 'servicenow_
↪sys_id', 'gcp_project_id',
                                'gcp_zone', 'gcp_instance_id', 'azure_vm_id',
↪'azure_resource_id',
                                'aws_availability_zone', 'aws_ec2_instance_ami_id
↪',
                                'aws_ec2_instance_group_name', 'aws_ec2_instance_
↪state_name',
                                'aws_ec2_instance_type', 'aws_ec2_name', 'aws_ec2_
↪product_code',
                                'aws_owner_id', 'aws_region', 'aws_subnet_id',
↪'aws_vpc_id',
                                'installed_software', 'bigfix_asset_id'
                                ]
```

   For more information about these configuration settings, see *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.

6.7.5 Ingesting content via http(s) proxy

For ingestion via proxy, you'll need to create an override to the RaaS service and add new environment variables for `http proxy` and `https proxy`.

To configure the RaaS node to use https proxy:

1. Complete the previous steps to enable automatic ingestion.
2. On the master in the command line, edit the RaaS service:

```
systemctl edit raas
```

3. Add the following lines to the generated file.

```
[Service]
Environment="http_proxy=http://<hostname>:234"
Environment="https_proxy=https://<hostname>:234"
Environment="HTTP_PROXY=http://<hostname>:234"
Environment="HTTPS_PROXY=http://<hostname>:234"
```

4. If your proxy requires password authentication, you may need to set this as part of the proxy environment variables. For example:

```
Environment="HTTP_PROXY=http://USER:PASSWORD@<hostname>:234"
```

5. If your proxy uses an internal Certificate Authority, you may also need to set the `REQUESTS_CA_BUNDLE` environment variable to ensure that the proxy is able to use it. For example:

```
Environment="REQUESTS_CA_BUNDLE=/etc/pki/tls/certs/ca-bundle.crt"
```

6. Restart the RaaS service:

```
systemctl restart raas
```

After the service restarts, content begins to download. This may take up to 20 minutes.

6.7.6 Manual content ingestion

Air-gapped API (RaaS) 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 Protect content from the *Downloads page*.
2. Log in to an 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. Ingest the tarball contents.

```
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}
```

6.7.7 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 `/`), then it is relative to the RaaS service cache dir `/var/lib/raas/cache`
`download_enabled`	If `True`, SaltStack Protect content downloading is enabled. Set to `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 the RaaS service boots (default: `True`)
`compile_stats_interval`	How often (in seconds) SaltStack Protect stats will be compiled
`stats_snapshot_interval`	How often (in seconds) SaltStack Protect stats will be collected
`old_policy_file_lifespan`	Lifespan (in days) of old policy files that will remain in the RaaS file system
`delete_old_policy_files_interval`	How often (in seconds) old SaltStack Protect policy files will be deleted from the RaaS file system
`tenable_asset_import_enabled`	If `True`, minion grains in SaltStack Config will be sent to Tenable.io for matching assets (default: `True`)
`tenable_asset_import_grains`	List of minion grains to send to Tenable.io, if tenable asset import is enabled. SaltStack Protect supports only `fqdn`, `ipv4`, `ipv6`, and `hostname` out of the box, however you can send other information by defining custom grains. For more on grains, including how to write custom grains, see Salt documentation: Grains. If you have only a subset keys in your grains data, only those in the subset will be synced. `fqdn` and `ipv4` will be sent even if you do not list them here. For more information, see the Tenable Import assets documentation.

6.7.8 FAQ

- **Q: How often is new SaltStack Protect content released?**
  - A: The current release frequency is about once per quarter. However, content might be released more frequently in the future.
- **Can I get access to new content sooner if I use automatic content ingestion instead of manual ingestion?**
  - A: The same content is available, whether you ingest manually or automatically.

However, if you use manual ingestion, you need to plan to check for security content updates and develop a process to manually ingest updated content when it is available.

6.7.9 Next steps

After configuring SaltStack Protect, there may be additional post-installation steps. Check the list of post-installation steps to ensure you have completed all the necessary steps.

6.8 Set up Splunk integration

6.8.1 Overview

SaltStack Config integrates with Splunk to help you optimize and secure your digital infrastructure using the SaltStack Config Add-On for Splunk Enterprise. The add-on is available on Splunkbase, and requires SaltStack Config version 6.3 or higher.

6.8.2 Prerequisites

Setting up your Splunk integration with SaltStack Config is one post-installation step in a series of several steps that should be followed in a specific order. First, complete one of the installation scenarios and then read the following post-installation pages:

- *Install the license key*
- *Install and configure the Master Plugin*
- *Log in for the first time and change default credentials*
- *Accept the Salt master key and back up data*
- *Set up SSL certificates*

6.8.3 Set up SaltStack Config add-on in Splunk

The SaltStack Config add-on in Splunk takes advantage of a new, Prometheus-compatible metrics endpoint which reports over 25 unique SaltStack Config metrics. These metrics provide insight into the health of your infrastructure. Accessing them in Splunk is useful for monitoring for outages, identifying abnormal activity, and more. It also gives you the ability to take automated actions based on a specific Splunk event using SaltStack Config.

For instructions on how to install and configure the add-on, see the full add-on documentation in the SaltStack knowledge base.

For more on the SaltStack Config metrics endpoint, see the Help Documentation embedded in the SaltStack Config user interface.

6.8.4 Next steps

After setting up your integration with Splunk, there may be additional post-installation steps. Check the list of post-installation steps to ensure you have completed all the necessary steps.

6.9 Set up Single Sign-On (SSO)

6.9.1 Overview

SaltStack Config integrates with third-party identity and access management solutions to enable users to login to the SaltStack Config user interface. SaltStack Config supports various authentication integrations:

- Single sign-on authentication (SSO) for an identity provider (IdP) that uses the SAML or OAuth protocols.
- Access management for directory services that use the LDAP protocol, such as Active Directory Domain Services.

Alternatively, you could also use the authentication that is native to SaltStack Config by storing user credentials locally in SaltStack Config on the RaaS node.

6.9.2 Prerequisites

Setting up SSO is one post-installation step in a series of several steps that should be followed in a specific order. First, complete one of the installation scenarios and then read the following post-installation pages:

- *Install the license key*
- *Install and configure the Master Plugin*
- *Log in for the first time and change default credentials*
- *Accept the Salt master key and back up data*
- *Set up SSL certificates*

6.9.3 How to set up SSO or directory services

You can best configure SSO or directory services using the user interface as opposed to using the API (RaaS) or command line. The instructions for setting up SSO or directory services are included in the SaltStack Config documentation. These are static HTML reference documents that do not require a link outside the firewall.

To access this documentation:

1. Login to SaltStack Config.
2. In the toolbar, click **Help** > **Help Documentation**.
3. Navigate to one of the following pages:
    - Authentication with SAML
    - Authentication with LDAP
    - Authentication with OAuth and OIDC

6.9.4 Next steps

After configuring SSO, there may be additional post-installation steps. Check the list of post-installation steps to ensure you have completed all the necessary steps.

CHAPTER

SEVEN

EXTENDING SYSTEM ARCHITECTURE

7.1 Multiple RaaS nodes

7.1.1 Overview

This page explains how to configure multiple RaaS nodes that share a single PostgreSQL database and Redis node. This method is also sometimes called *clustering*.

These instructions demonstrate how to install the PostgreSQL and Redis services on the primary RaaS node using the *Single-node installation* scenario.

Note: Some high availability requirements may require consultation services.

7.1.2 Preparatory steps

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

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

Before configuring multiple RaaS nodes, follow the steps to install two standalone RaaS nodes using the *Single-node installation* scenario. At the end of this scenario, both nodes should run SaltStack Config in standalone mode, meaning each node has its own local version of PostgreSQL and Redis.

7.1.3 Configure the primary RaaS node

This section explains how to configure the first RaaS node to work with a second RaaS node.

To configure the first RaaS node:

1. Follow the steps to install two standalone RaaS nodes using the *Single-node installation* scenario. At the end of this scenario, both nodes should run SaltStack Config in standalone mode, meaning each node has its own local version of PostgreSQL and Redis.
2. On the first RaaS node, stop the RaaS, Redis, and PostgreSQL services using the following commands:

```
systemctl stop raas
systemctl stop redis
systemctl stop postgresql-12
```

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

3. On the first RaaS node, update your `postgresql pg_hba.conf` file to allow remote connections from the other RaaS node. To allow remote connections, append the following entry to the end of that file, replacing the example IP address with the IP address of the second RaaS node:

```
# 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 bind is set to localhost. Add the following to your file:

```
#bind 127.0.0.1
```

5. Start the services and verify their status using the following commands:

```
systemctl start postgresql-12
systemctl status postgresql-12
systemctl start redis
systemctl status redis
systemctl start raas
systemctl status raas
```

6. Access the SaltStack Config user interface using the URL for the first RaaS node to confirm that SaltStack Config is working properly on the first node.

After verifying you can access the user interface, proceed to the next section.

7.1.4 Configure the secondary RaaS node

This section explains how to configure the second RaaS node to work with the primary RaaS node.

To configure the second RaaS node:

1. On the second RaaS node, stop the RaaS, Redis, and PostgreSQL services using the following commands:

```
systemctl stop raas
systemctl stop redis
systemctl stop postgresql-12
```

2. On the second RaaS node, update the `/etc/raas/raas` file to connect to the remote Redis and PostgreSQL services on the *first* RaaS node. The `customer_id` configuration should be identical on both nodes. The following shows an example configuration:

```
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
```

3. Copy the `/etc/raas/pki/.raas.key` and `/etc/raas/secconf` from the first node to the second node. 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
```

4. Start the RaaS service and verify its status using the following commands:

```
systemctl start raas
systemctl status raas
```

5. Access the SaltStack Config user interface using the URL for the second RaaS node to confirm that SaltStack Config is working properly on the second node.

After verifying you can access the user interface on the secondary node, proceed to the next section.

7.1.5 Test the configuration

To test whether your new system architecture is working correctly:

1. To test the configuration, create a new object, such as a new target. Verify that the change is present on both nodes when you refresh the user interface.
2. On the second RaaS node, disable the Redis and PostgreSQL services using the following commands:

```
systemctl disable redis
systemctl disable postgresql-12
```

You now have two instances of the RaaS node running. For troubleshooting, *Contact Support*.

7.2 Improve system performance

7.2.1 Overview

Perhaps in a few weeks or after installing SaltStack Config when you are feeling confident that everything is operating as expected, you can begin to optimize your system's overall performance. This page provides some guidance and additional links to articles for more information.

7.2.2 Tuning processes on your RaaS node

When the RaaS service starts, it creates two types of processes:

- **Tornado processes** - Allows connections from Salt masters and web browsers
- **Celery processes** - Background workers

By default, the RaaS service sets the count for each process type to half the number of CPU cores.

In most cases this is optimal, as the RaaS node should be dedicated to this task.

If you need to deploy RaaS on a node that supports additional services, you can override the default behavior by adding the following to your RaaS service configuration file located at `/etc/raas/raas`:

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

The following guides might be helpful for tuning:

- SaltStack Enterprise Performance Configurations
- Background Worker Options

7.2.3 Benchmarking guide

For help with benchmarking the performance of SaltStack Config, see Benchmarking Guide for SaltStack Enterprise.

7.2.4 Tuning PostgreSQL

For a PostgreSQL tuning guide, see Tuning your PostgreSQL Server for SaltStack Enterprise.

CHAPTER

EIGHT

UPGRADE

8.1 Upgrade from a previous version

8.1.1 Overview

This page explains how to upgrade SaltStack Config to the latest stable version from a previous version. These upgrade instructions cover an upgrade from SaltStack Config version 6.3.0 to version 6.4.0.

To upgrade SaltStack Config, you will need to:

- Back up your data, including certain files and directories that are crucial to your specific installation of SaltStack Config
- Upgrade PostgreSQL (optional, but recommended)
- Upgrade your Salt infrastructure (optional, but recommended)
- Download the upgrade files
- Upgrade the RaaS node
- Upgrade any Salt masters using the Master Plugin

This page explains how to complete each of these steps in more detail.

Warning: If you are upgrading your SaltStack Config installation, refer to the following upgrade instructions. Do not attempt to install using the automated installer (single-node or multi-node) or manual installation instructions.

Upgrading versions earlier than 6.3.0

As a best practice, always upgrade from the latest major version of SaltStack Config to the new release. If you are upgrading from a version earlier than 6.3.0, you might see the best results if you upgrade in increments from one major release to the next.

For instructions on upgrading to earlier SaltStack Config versions, see the specific upgrade instructions for the releases you are upgrading from. The upgrade instructions for previous releases are included in the installation guide PDF for the subsequent release. For example, if you need to upgrade from 5.5 to 6.0, see the upgrade instructions in the 6.0 installation PDF.

Version	Installation and Upgrade Guide PDF
6.4	`6.4 Installation and Upgrade Guide`
6.3	`6.3 Installation and Upgrade Guide`
6.2	`6.2 Installation and Upgrade Guide`
6.1	`6.1 Installation and Upgrade Guide`
6.0.1	`6.0.1 Installation and Upgrade Guide`
6.0	`6.0.0 Installation and Upgrade Guide`
5.5.1	`5.5.1 Installation and Upgrade Guide`

Best practices when upgrading

When preparing to perform an upgrade of SaltStack Config, the following recommendations could be helpful:

- **Back up your data**. In order to prevent data loss, back up your data. For an explanation of which files and directories must be backed up before upgrading, see *Back up your data*.
- **Perform upgrades during hours of slow network activity**. 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.
- **Check the database for any old commands being stored**. In some cases, the PostgreSQL database stores old commands that haven't run. These commands might run during the upgrade process, when you restart the Master Plugin. To prevent this from happening, check whether any old commands are stored in the database, and enable skipping jobs that are older than a defined time.
- **Test the upgrade before deploying**. If possible, you could try running a dress rehearsal in a test environment to get a sense of how long the upgrade could take.
- **Read through the whole guide first**. Consider also reading through this entire guide one time before you implement an upgrade so that you have a good sense of the tasks that are required and whether they require planning from your team or if stakeholders must be notified of pending changes.

8.1.2 Back up your data

Before upgrading SaltStack Config, you should always back up your data. The following sections explain which data needs to be backed up so that it can be restored after you complete the upgrade process.

Back up SaltStack Config files and directories

The following files and directories contain your custom SaltStack Config configurations and need to be backed up before upgrading:

1. On the RaaS node, back up these entire directories:
   - `/etc/raas/raas`
   - `/etc/raas/raas.secconf`
   - `/var/log/raas`
   - `/etc/raas/pki/`

   Note: The `pki` directory contains hidden files, so ensure you back up the entire directory.

Backing up the log files in the `/var/log/raas` directory is optional. During the upgrade process, you'll clear the log files to provide a clean log file if troubleshooting is necessary.

2. On each master, back up the `/etc/salt/master.d/raas.conf` and `/etc/salt/master.d/eAPIMasterPaths.conf` files.

Note: Depending on how you initially installed SaltStack Config, the eAPI master paths could instead be in the `/etc/salt/master.d/raas.conf` file instead.

After backing up the SaltStack Config and master files, proceed to the next section.

Back up your database schema

When upgrading your RaaS node, the database schema is updated. For that reason, ensure you create a backup of your database before the upgrade.

To back up your database you need to first look up your PostgreSQL database name and then copy the contents:

1. On the PostgreSQL server, back up these files:
   - `postgres.conf`
   - `pg_hba.conf`
2. Log in as the `postgres` user using the following command:

```
sudo su - postgres
```

3. Get your database name, using the following commands to enter PostgreSQL and then list the databases:

```
psql
\l
```

4. To exit PostgreSQL and log out as the `postgres` user, press Ctrl+D and then run the following command:

```
exit
```

5. Copy database contents to a file. The following command gives an example:

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

Your database files are now backed up. For the latest information about performing backups, see PostgreSQL database backups.

8.1.3 Upgrade PostgreSQL database

SaltStack Config requires a PostgreSQL 9.6 database, but PostgreSQL 12.4 is recommended. The recommended version of PostgreSQL is included with the SaltStack Config installer.

Upgrading to the latest version of PostgreSQL is not required. However, upgrading PostgreSQL can possibly improve performance. For instructions on upgrading to the latest version of PostgreSQL, see PostgreSQL upgrade.

8.1.4 Upgrade Salt

For best performance, ensure your Salt components are running on the latest major version of Salt. For instructions on upgrading your masters and other Salt components, see Upgrading Your Salt Infrastructure.

For instructions on upgrading the Salt Crystal package, see How to Upgrade Salt Crystal.

8.1.5 Download upgrade files

To download the files for upgrading:

1. On the *Downloads* page, go to the sse-upgrade-downloads section.
2. Download all files listed needed for upgrading.
3. Verify the files. For specific instructions, see *Verify the installation files*.

After you've downloaded the files, proceed to the next section.

8.1.6 Upgrade the RaaS node

This section explains how to upgrade the RaaS node from 6.3.0 to version 6.4.0. Be aware that database upgrades require re-indexing data. If your data is complex, a database upgrade could potentially take several hours. For a discussion of when to plan an upgrade and other tips, see *Best practices when upgrading*.

Warning: Before upgrading your RaaS node, you **must back up your system data** to avoid data loss. For an explanation of which files and directories must be backed up before upgrading, see *Back up your data*.

To upgrade the RaaS node:

1. Save any changes you made to the default file system, pillar data, and jobs as new files or jobs.
2. Note any pillar assignments that are made to the default targets. These need to be re-assigned after upgrade.
3. Stop the RaaS service using the following command:

```
sudo systemctl stop raas
```

4. Remove the log file(s) in the `/var/log/raas` directory. Clearing the log files provides a clean log file if troubleshooting is necessary.
5. Remove the currently installed version of the API (RaaS) with the following command:

```
sudo yum remove raas
```

6. Upgrade the RaaS node by installing the latest RPM. Use the following example command, replacing the exact file name of the RPM:

```
sudo yum install raas-rpm-file-name.rpm
```

7. **IMPORTANT**: Restore the backup of the following files:
   - `/etc/raas/raas`
   - `/etc/raas/raas.secconf`
   - `/etc/raas/pki/`

8. Update permissions for the `raas` user with the following command:

```
sudo chown -R raas:raas /etc/pki/raas/certs
```

9. OPTIONAL: If you have a SaltStack Comply license, add the following new section to the `/etc/raas/raas` file:

```
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
```

Note: 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 available for SaltStack Config versions 6.0 and later. The previous configuration options in the `/etc/raas/raas` configuration file are specific to these add-on modules.

---

10. OPTIONAL: If you have a SaltStack Protect license , add a new section to the `/etc/raas/raas` file:

```
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
  tenable_asset_import_enabled: True
  tenable_asset_import_grains: ['fqdn', 'ipv4', 'ipv6', 'hostname', 'mac_address',
↪ 'netbios_name',
                                'bios_uuid', 'manufacturer_tpm_id', 'ssh_
↪fingerprint',
                                'mcafee_epo_guid', 'mcafee_epo_agent_guid',
↪'symantec_ep_hardware_key',
                                'qualys_asset_id', 'qualys_host_id', 'servicenow_
↪sys_id', 'gcp_project_id',
                                'gcp_zone', 'gcp_instance_id', 'azure_vm_id',
↪'azure_resource_id',
                                'aws_availability_zone', 'aws_ec2_instance_ami_id
↪',
                                'aws_ec2_instance_group_name', 'aws_ec2_instance_
↪state_name',
                                'aws_ec2_instance_type', 'aws_ec2_name', 'aws_ec2_
↪product_code',
```

(continues on next page)

(continued from previous page)

```
                                'aws_owner_id', 'aws_region', 'aws_subnet_id',
↪'aws_vpc_id',
                                'installed_software', 'bigfix_asset_id'
                                ]
```

Note: This step is optional and only applies to organizations that have a valid SaltStack Protect license. This add-on module is available for SaltStack Config versions 6.0 and later. The previous configuration options in the `/etc/raas/raas` configuration file are specific to these add-on modules.

11. The RaaS currently has a known issue related to stale jobs. (See *Known issues* for more information.) When upgrading, some users might notice a queue of stale jobs that are stuck in a pending state. Upgrading the RaaS node can cause these jobs to run unless they are first cleared out.

    To prevent this from happening, first check whether any old commands are stored in the database. On your PostgreSQL node, check for any pending jobs using the following command:

    ```
    select count(1) from commands where state = 'new';
    ```

    The result is the number of pending jobs. If the number of jobs is `0`, proceed with the rest of the upgrade process. If the number of jobs is greater than `0`, *Contact Support* for a workaround.

12. Upgrade the RaaS service database using the following command:

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

    Note: 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.

13. After the upgrade, exit the session for the `raas` user with the following command:

    ```
    exit
    ```

14. Start the RaaS service using the following command:

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

Verify that SaltStack Config is functioning correctly and is running the latest version. Proceed to the next section.

8.1.7 Upgrade masters with the Master Plugin

After you have successfully upgraded the RaaS node, you can then upgrade any masters that use the Master Plugin to connect to SaltStack Config.

Note: Before you upgrade the masters, ensure that the pip3 application is installed on the masters. If you are upgrading from the latest version of the Master Plugin, this application is already installed.

To upgrade the Master Plugin on a master:

1. Stop the `salt-master` service using the following command:

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

2. Check which version of Python is running on the 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 Config plugin for the master). For example:

RHEL/CentOS

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

Ubuntu

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

3. Upgrade the Master Plugin by manually installing the updated Python wheel. Use the following example commands, replacing the exact name of the wheel file:

   RHEL/CentOS

   Ubuntu

```
sudo pip3 install SSEAPE-file-name.whl --prefix /usr
```

```
sudo pip3 install SSEAPE-file-name.whl
```

Note: Some users might need to alter the syntax to `pip3.6` or `pip36` for their operating systems.

4. Update the API (RaaS) module paths by editing the `/etc/salt/master.d/eAPIMasterPaths.conf` file to reference the paths to the various modules. For example, you might change all `python2.7` references in this file to `python3.6`.

Note: Depending on how you initially installed SaltStack Config, the eAPI master paths could instead be in the `/etc/salt/master.d/raas.conf` file instead.

5. Check the `engines` section in `/etc/salt/master.d/raas.conf` to confirm that it matches the following:

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

Note: If a problem occurred, you may need to restore your backups of the `/etc/salt/master.d/raas.conf` and `/etc/salt/master.d/eAPIMasterPaths.conf` files.

6. Start the `salt-master` service with the following command:

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

The upgrade process is now complete. If you encounter any other errors, refer to the *Troubleshooting* page or *Contact Support*.

CHAPTER

NINE

TROUBLESHOOTING AND SUPPORT

9.1 Contact Support

SaltStack Support is available to SaltStack Config customers with an active license and maintenance agreement. For more information about contacting SaltStack Support, refer to the SaltStack Support Policy.

9.2 Troubleshooting

9.2.1 Overview

The following page includes some of the common errors that users experience during the SaltStack Config installation process and how to fix them.

9.2.2 Salt installation problems

The following section addresses issues you might encounter when installing Salt.

yum install returns error due to DNS or transparent proxy

When installing Salt as a pre-installation step (see *Install or upgrade Salt*), `yum` might return the following error:

```
[Error 14] curl#60 - "Peer's Certificate issuer is not recognized.
```

In this case, it is likely that you are either experiencing DNS issues or you have a transparent TLS/https proxy in your environment.

To resolve DNS issues, ensure that `repo.saltstack.com` resolves on your machine or that you can reach that server.

If you have a transparent proxy, add `sslverify=0` to the SaltStack yum repo configuration and then retry the installation of the packages. This will workaround the fact that your transparent proxy is interfering with connection certificates and TLS signatures.

9.2.3 Single-node installation problems

The following section explains how to resolve issues that users experience when running a *Single-node installation* scenario.

Single-node installation script terminates

If both the Salt master and Salt minion are installed, the script skips this step and proceeds with the setup of SaltStack Config.

If either the salt-master service or the salt-minion service packages are installed, but not both, the script will terminate. The script terminates as a safeguard to prevent the user from accidentally disrupting an existing installation.

The solution is to ensure that both the master and minion are installed on the node before running the installation script.

Package installs fail

If the installations of `postgres`, `jemalloc`, `redis`, etc. fail during the single-node installation, one possibility is that `yum` is configured with a restrictive `localpkg_gpgcheck` option ensuring local packages' GPG signatures are checked.

The failure may look similar to this:

```
[ERROR ] Command '['systemd-run', '--scope', 'yum', '-y', 'install', '/var/cache/salt/
↪minion/files/base/sse/eapi_database/files/postgresql12-libs-12.1-1PGDG.rhel7.x86_64.
↪rpm', '/var/cache/salt/minion/files/base/sse/eapi_database/files/postgresql10-12.1-
↪1PGDG.rhel7.x86_64.rpm', '/var/cache/salt/minion/files/base/sse/eapi_database/files/
↪postgresql12-server-12.1-1PGDG.rhel7.x86_64.rpm', '/var/cache/salt/minion/files/
↪base/sse/eapi_database/files/postgresql12-contrib-12.1-1PGDG.rhel7.x86_64.rpm']'
↪failed with return code: 1
```

If the option is enabled, the option `localpkg_gpgcheck = 1` will have been set explicitly in `/etc/yum.conf`. Set this to `localpkg_gpgcheck = 0` to allow the single-node installation to complete.

9.2.4 Multi-node installation problems

The following section explains how to resolve issues that users experience when running a *Multi-node installation* scenario.

Master is not showing the keys for the minion

Sometimes users experience a problem where minion IDs don't appear when trying to get the master to accept minion keys. To fix this problem, you can specify the IP address of the master in each minion's `etc/salt/minion.d/id.conf` file. Edit this file and change the `master` setting to show the master's IP address. For example, `master: 127.0.0.0`. For additional methods of connecting the minion to the master, see Configuring the Salt Minion.

Authentication error occurs when applying highstate

During the initial application of the highstate to the first master, you may see the following error message: `Authentication error occurred`.

This error displays because the master has not yet authenticated to the RaaS node, but the Master Plugin installation state will restart the salt-master service and the issue will be resolved automatically.

9.3 Known issues

The following sections describe the currently known issues in SaltStack Config version 6.4.0.

9.3.1 General

- If the PostgreSQL database is not set to use UTF-8, sorting will not be consistent across the application.
- Job return numbers may differ from target numbers based on current key state and grain data.
- 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 upgrading SaltStack Config, some users might notice a queue of stale jobs that are stuck in a pending state. Upgrading the RaaS node can cause these jobs to run unless they are first cleared out. See the *Upgrade the RaaS node* step in the *Upgrade from a previous version* guide for more information.

9.3.2 The SaltStack Config user interface

- Scheduled jobs display in the user interface only if scheduled within the next 12 weeks.
- After upgrading SaltStack Config (for example, from 6.3.0 to 6.4.0), you must clear your web browser cache. Failing to clear the cache might result in unpredictable behavior in the user interface.
- When running jobs against a large number of minions, only 2,000 job returns show in the user interface by default. To retain job returns for all minions, use RaaS to query the `get_returns` function.
- If you are using multiple masters and have enabled `sseapi_cluster_id` on more than one master, the master might not show up in the Reports workspace in the Master version report (Home > Reports > Master Versions). It might also affect masters that have been installed using a single-node installation.

  To check whether your system is affected by this issue, SSH into your master and run the following command:

```
sudo salt-run config.get sseapi_cluster_id
```

  Note: If you find nothing after running this command, then this issue does not apply to you. If you do see a response, it means you have enabled `sseapi_cluster_id` and your master does not appear in the report. If this node is not in a cluster, you can safely comment the configuration out and restart the salt-master service. However, if your master is operating in a cluster, you must not change this setting.

9.3.3 SaltStack Comply and SaltStack Protect

- SaltStack Comply and SaltStack Protect require Salt version *2018.3.3* or later for Linux or Unix minions, and *3000* or higher for Windows minions.
- It is recommended SaltStack Comply and SaltStack Protect 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.
- In the SaltStack Config user interface, SaltStack Protect does not show any minions included in a newly-created policy until you have run the first assessment.
- A SaltStack Protect security policy's **Advisories** tab does not show vulnerabilities that have been remediated. To view remediated advisories, go to the policy's **Targets** tab, select a remediated minion, and then select the **Last Remediation** tab to verify the minion was remediated.
- Remediating vulnerabilities on a minion might not result in any changes to the minion if the operating system 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.
- After running a SaltStack Protect scan one day, vulnerabilities are shown in the policy dashboard charts. If those vulnerabilities are not remediated and a new scan is run the following day, the policy chart resets to zero and begins with a fresh count for that day. This behavior is expected as the chart is intended to act as a daily scan summary. The chart does not display results for the days on which no vulnerability scans were run.
- In SaltStack Comply, you might need to repeat the process of remediating and scanning multiple times in order to achieve full compliance. This is because some checks are dependent on the completion of others. For example, one check might require a package that's deployed by another check before it can be remediated properly.

9.3.4 LDAP and Active Directory

- 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 Config.

CHAPTER

TEN

REFERENCE

10.1 Release notes

10.1.1 What's in 6.4.0?

SaltStack® Enterprise 6.4.0 includes a variety of feature enhancements for SaltStack Comply and Protect and several customer-requested user interface improvements. This release also resolves several bugs and technical issues, especially for the LDAP and Active Directory authentication system.

10.1.2 Enhancements

General user interface improvements

This release includes several improvements to the overall functionality of SaltStack Config user interface, including:

- Users can now run a command directly from a minion's detail page.
- The target create / edit dialog box now has an improved user interface that better manages how targets are defined when including operators.
- An improved unified component for dropdown inputs that allow for longer values.
- In cases where items have very long custom names (such as custom jobs or targets), those names are truncated in menus. The full name is displayed when hovering over the item with a mouse.
- Administrators can create custom warning or informational banners that can be displayed on the login screen for all users.
- Various input fields have been widened to allow for longer entries.
- Administrators can now create multiple types of Admin roles using the new Admin permission settings.

SaltStack Protect

This release included multiple improvements to SaltStack Protect:

- Users can view the status of the latest assessment and the latest remediation to determine whether it is queued, in progress, or complete from any page inside a Protect policy.
- While editing an assessment, users can select an option that will run the assessment immediately after saving.
- The user interface for editing policy names and defining targets on the same page has been streamlined and simplified.

SaltStack Comply

This release includes the following enhancements for SaltStack Comply:

- Custom variables for remediations now display a tooltip that informs users what the variable is and what values are expected and supported.
- minions that are not applicable to a compliance result are no longer factored into compliance scores.
- While editing an assessment, users can select an option that will run the assessment immediately after saving.

LDAP and Active Directory

The 6.4.0 release of SaltStack includes several enhancements to the LDAP and Active Directory authentication system, including:

- Improved functionality in the user interface for selecting users contained inside nested groups or container groups.
- The prefill defaults are now more helpful and accurate.
- Improved search within the preview field.
- Improved support for OpenLDAP including configurable Group Name Attribute.

10.1.3 Related SaltStack Enterprise content releases

Some content published by SaltStack is released independently from the standard SaltStack Config releases, including SaltStack Comply content and specialized minion packages. Within this release cycle, the following new content and packages were made available:

- Ubuntu 16.04 CIS Benchmark for SaltStack Comply
- The DISA STIG RHEL7 Benchmark for SaltStack Comply

10.1.4 Issues resolved in this release

- In the user interface, compound targeting with grains does not work if the grain filter value contains spaces.
- If your File Server contains a large number of files, the File Server search functionality might lag.
- In some cases, SaltStack Comply and Protect assessments results might show a negative count of minions returned.
- In SaltStack Protect, vulnerability scans run immediately following a fresh installation of SaltStack Enterprise might fail. This happens because after the initial install, SaltStack Enterprise takes roughly 15-20 minutes to ingest vulnerability content.
- When importing a Tenable vulnerability scan through the RaaS, the import may complete but will not change its status to complete on the backend and the user interface might stall while importing. This issue only occurs if the Tenable scan shows no vulnerabilities or if the Tenable scan does not cover any assets.
- When working with nested groups in LDAP, 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 an LDAP 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.
- SaltStack Enterprise might not correctly detect Active Directory members that belong to 20 or more groups.
- Active Directory users with Superuser access privileges may find their password settings are occasionally reset when they access the Authentication workspace (accessed from the Administration menu).

10.1.5 Feedback

Please submit any feedback using the form at https://saltstack.com/enterprisefeedback. You can also access the feedback form in SaltStack Config by going to **Help > Feedback**.

10.1.6 Release notes archive

For past release notes, see the SaltStack Support Portal.

10.2 Manual installation

10.2.1 Overview

This page explains how to install SaltStack Config manually as an alternative to either the *Single-node installation* or the *Multi-node installation* scenarios. The manual installation method supports installation on the following operating systems:

- RedHat or CentOS (recommended)
- SUSE 12
- SUSE 15

Danger: Manual installation is **not** recommended. These steps are included for your reference if you would like to understand every procedure that is taken when using the installer or running an installation scenario using one of the standard installation scenarios.

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. You are **strongly** encouraged to use one of the standard installation scenarios instead.

The steps below are confirmed for a standalone deployment of SaltStack Config (where all related services reside on a single host). Advanced users will likely adapt these instructions to their deployment. If you are not an advanced user, use the standard installation scenarios instead or consider using consulting services. To begin the standard installation process, see *Installation overview*.

Note: SaltStack Config supports SLES 12. However, be aware that as of June 2020, SLES 12 SP4 has reached end of General Support from SUSE. Consider upgrading to SLES 15, contacting your database administrator, or contacting SUSE support for further assistance. For more information about supported distributions, see SUSE Product Support Lifecycle.

10.2.2 Download manual installation files

To download the files for upgrading:

1. On the *Downloads* page, go to the sse-manual-install-downloads section.
2. Download all files listed needed for your specific operating system.
3. Verify the files. For specific instructions, see *Verify the installation files*.

After you've downloaded the files, proceed to the next section.

10.2.3 Import key files

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

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

After the key files have been imported, proceed to the next section.

10.2.4 PostgreSQL database installation and configuration

To install and configure the PostgreSQL database:

1. Install PostgreSQL using the following commands:

RHEL

```
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
```

SLES 12

As of June 2020, the package to install PostgreSQL for SLES 12 SP4 is no longer available at the Open SUSE downloads center. Consider upgrading to SLES 15, contacting your database administrator, or contacting SUSE support for further assistance.

SLES 15

```
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 RaaS node. Optionally, enable SSL.
3. Start PostgreSQL and create a database account for the RaaS node. For example:

RHEL

```
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.
```

SLES 12

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

SLES 15

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

After installing and configuring PostgreSQL, proceed to the next section.

10.2.5 Redis installation and configuration

To install and configure the Redis database:

1. Install Redis using the following commands:

RHEL

Install the `Redis` and `jemalloc` installation packages you downloaded from the *Downloads* page. Use the following commands, replacing the exact file names:

```
sudo yum install redis-filename.rpm jemalloc-filename.rpm
```

SLES 12

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

SLES 15

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

2. Start the Redis service, using the following commands:

RHEL

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

SLES 12

```
# 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
```

SLES 15

```
# 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**: If you are setting up Redis on a host that is separate from the RaaS node, you 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 RaaS nodes should use to authenticate.

```
bind 0.0.0.0
requirepass {{ your_redis_password }}
```

After installing and configuring Redis, proceed to the next section.

10.2.6 RaaS installation and configuration

To install and configure the RaaS node:

1. **FOR SLES 15 INSTALLATIONS ONLY**: Install the `xmlsec1` package. Without this dependency, a SLES 15 installation may fail. To download the package and read installation documentation, see xmlsec1.
2. Install the packages or RPM signing keys needed by your operating system:

   RHEL

   Download and install the `Python3.5` and `libpython3.5` installation packages for RH/CentOS that are provided in the *Downloads* page. Use the following command, replacing the exact file names:

   ```
   sudo yum install python35u-libs-filename.rpm python35u-filename.rpm
   ```

   SLES 12

   Import the RPM signing key using the following command:

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

   SLES 15

   Import the RPM signing key using the following command:

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

3. Download and install the RPM for your operating system:

   RHEL

   Download and install the Red Hat/CentOS SaltStack Config RPM, replacing the exact file name:

   ```
   sudo yum install raas-<version>-0.el7.x86_64.rpm
   ```

   SLES 12

   Download and install the SLES 12 RPM, replacing the exact file name:

   ```
   zypper install raas-<version>-0.sles12.x86_64.rpm
   ```

SLES 15

Download and install the SLES 15 RPM, replacing the exact file name:

```
zypper in raas-<version>-0.sles15.x86_64.rpm
```

4. In the RaaS service 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` and see the next step for additional details.

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

5. If you set `ssl` to `True` in the previous step, you've enabled an SSL connection, but additional information is required to verify the server's SSL certificate. To configure certificate verification, in `/etc/raas/raas`, add a new `ssl_opts` key and provide values as follows:

Option	Description
`sslmode`	Choose the mode for your SSL connection from one of the following: • `disable` - Use only cleartext connections. This value is used when `ssl` is set to False. • `prefer` - Use SSL but fallback to cleartext if SSL is not available. • `require` - Use an SSL connection but do not attempt to verify the server's certificate. • `verify-ca` - Use the contents of `sslrootcert`, if present, to validate the server's SSL certificate. Or if `sslrootcert` is not present, use the system certificate store to validate the server's SSL certificate. • `verify-full` - Use the contents of `sslrootcert`, if present, to validate the server's SSL certificate. Or if `sslrootcert` is not present, use the system certificate store to validate the server's SSL certificate. `verify-full` requires that the hostname in the certificate match the hostname SaltStack Config uses when connecting. For more on these settings, see the PostgreSQL documentation.
`sslrootcert`	Location on the RaaS filesystem of the CA certificate to use if a self-signed certificate is in place on the PostgreSQL server
`sslcert`	Location of the client certificate on the RaaS server to use instead of username and password to connect to PostgreSQL
`sslkey`	Location of the key file that goes along with the client certificate referenced in `sslcert`

For more in-depth information about these options, see the PostgreSQL documentation: Client Verification of Server Certificates, as well as the following example configurations.

Example 1

The first example shows a configuration set to full verification. This means that the certificate PostgreSQL presents to SaltStack Config is validated against the Certificate Authority certificate specified in the file `path/to/CA_Certificate`. Furthermore, the Common Name in the SaltStack Config certificate must match the

hostname SaltStack Config is using for PostgreSQL.

```
sql:
  ssl: True
  ssl_opts:
    sslmode: verify-full
    sslrootcert: path/to/CA_certificate
```

Example 2

The second example enables SSL communication without certificate validation, and authenticates the user that the RaaS uses to connect to PostgreSQL via client SSL certificate.

```
sql:
  ssl: True
  ssl_opts:
    sslmode: require
    sslcert: path/to/Client_Certificate
    sslkey: path/to/Key_for_Client_Certificate
```

6. In the RaaS service configuration file `/etc/raas/raas`, define options for background workers:

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

Note: SaltStack Config includes a range of different background worker settings to improve performance for various deployment scenarios. For more information, see *Improve system performance*.

---

7. In the RaaS service configuration file `/etc/raas/raas`, configure the location of your Redis server:

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

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

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

9. 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 are stored in `/etc/raas/raas.secconf`.

Note: 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.

---

10. Enable the RaaS service at system startup and launch the service using the following commands:

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

The manual installation process is now complete.

10.2.7 Next steps

Once the manual installation process is complete, you must complete several post-installation steps:

- *Install the license key*
- *Install and configure the Master Plugin*
- *Log in for the first time and change default credentials*
- *Accept the Salt master key and back up data*
- *Set up SSL certificates*
- *SaltStack Comply configuration* (optional)
- *SaltStack Protect configuration* (optional)
- *Set up Splunk integration* (optional)
- *Set up Single Sign-On (SSO)* (optional)

The first post-installation step is to install the license key. To begin the next post-installation step, see *Install the license key*.