Installing and Configuring SaltStack Config

12 October 2021

VMware vRealize Automation SaltStack Config 8.6

vRealize Automation 8.6

You can find the most up-to-date technical documentation on the VMware website at:

https://docs.vmware.com/

VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94304
www.vmware.com

Contents

Installing and Configuring SaltStack Config

1

You can install the SaltStack Config service in your vRealize Automation environment using one of two installation methods. Once installed, you can configure a SaltStack Config integration and complete the necessary post-installation tasks.

Important SaltStack Config runs on Salt, an open-source automation and configuration management engine. In order to begin using SaltStack Config for configuration management, you also need to install and run the Salt minion service on any nodes that you intend to manage using SaltStack Config. You can deploy the Salt minion service to your nodes using either vRealize Automation cloud templates or by installing the service through Secure Shell (SSH).

This installation guide explains the standard process for installing SaltStack Config 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.

If at any time you encounter difficulties with the installation that are not addressed by this guide, Chapter 11 Contact Support.

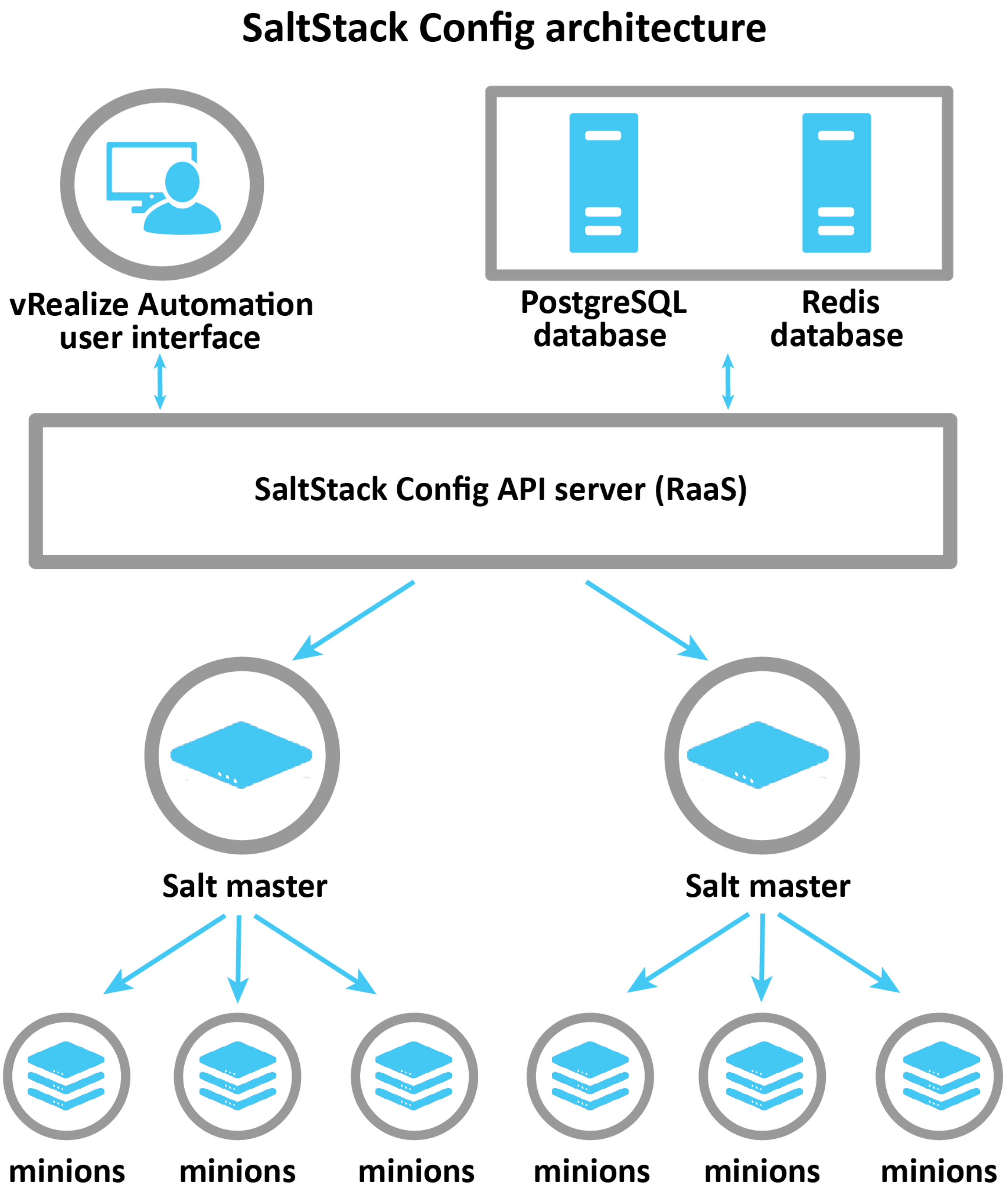
Installation goals

The goal of the installation process is to create the back-end SaltStack Config system architecture, which consists of four main components:

Component	Description
SaltStack Config API server, also known as RaaS	RaaS (Returner as a Service) provides RPC endpoints to receive management commands from the SaltStack Configuser interface, as well as RPC control endpoints to interface with connected Salt master(s).
Salt master(s) and the Master Plugin	You can connect as many Salt masters as needed to SaltStack Config, but at least one is required. The Salt master is the main connection between SaltStack Config and the rest of the nodes on your network (referred to as minions). When you issue a command from SaltStack Config (such as a job), the command goes to the Salt master for distribution to the targeted minions. The Master Plugin is installed on the Salt master(s). It allows the Salt master(s) to communicate with the SaltStack Config backend server, the RaaS node. The Master Plugin allows the Salt master(s) to access jobs or processes initiated by SaltStack Config, as well as external files and pillar data that are stored on the PostgreSQL database.
PostgreSQL database	SaltStack Config 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	SaltStack Config 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.

In the standard installation scenario, each of these components will be deployed to a dedicated node, which means you'll have four dedicated nodes when you're finished. If your system needs high availability, you might need to deploy multiple Salt masters, PostgreSQL databases, and Redis databases.

The following image shows the end goal of a standard installation:


SaltStack Config architecture
vRealize Automation user interface
PostgreSQL database
Redis database
SaltStack Config API server (RaaS)
Salt master
Salt master
minions
minions
minions
minions
minions
minions

Installation overview

The installation process has five main phases:

- Preinstallation
- Installation
- Post-installation
- Integrate with vRealize Automation
- Install Salt on the nodes you want to manage with SaltStack Config

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

Preinstallation

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 preinstallation 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 or virtual machines (VMs) need, etc.
- Plan any necessary workarounds if your network does not have access to the Internet.
- Install Salt on the nodes or VMs that will host the necessary SaltStack Config architecture.
- Download, verify, and import 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.

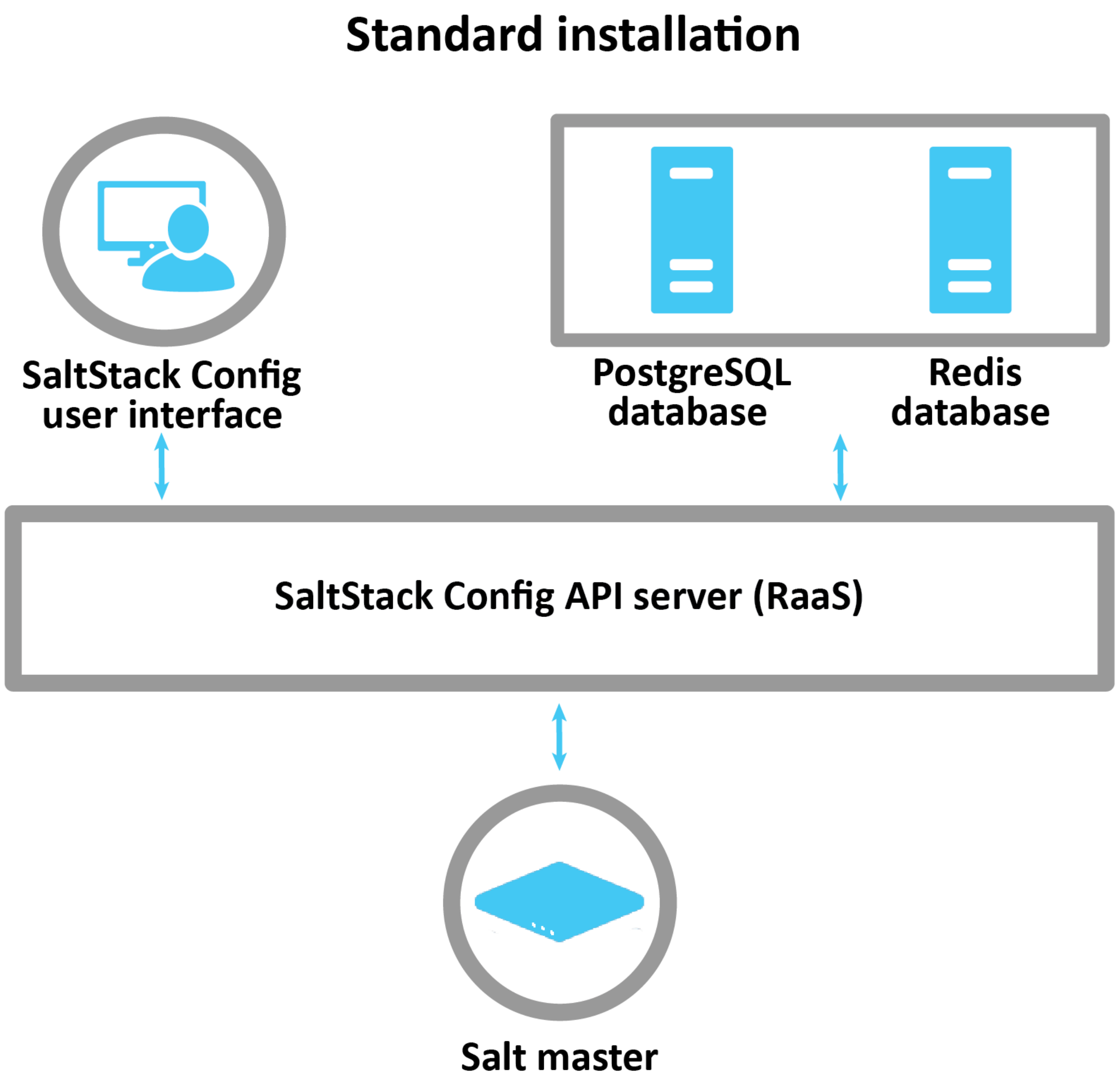
Installation

SaltStack Config supports two installation methods:

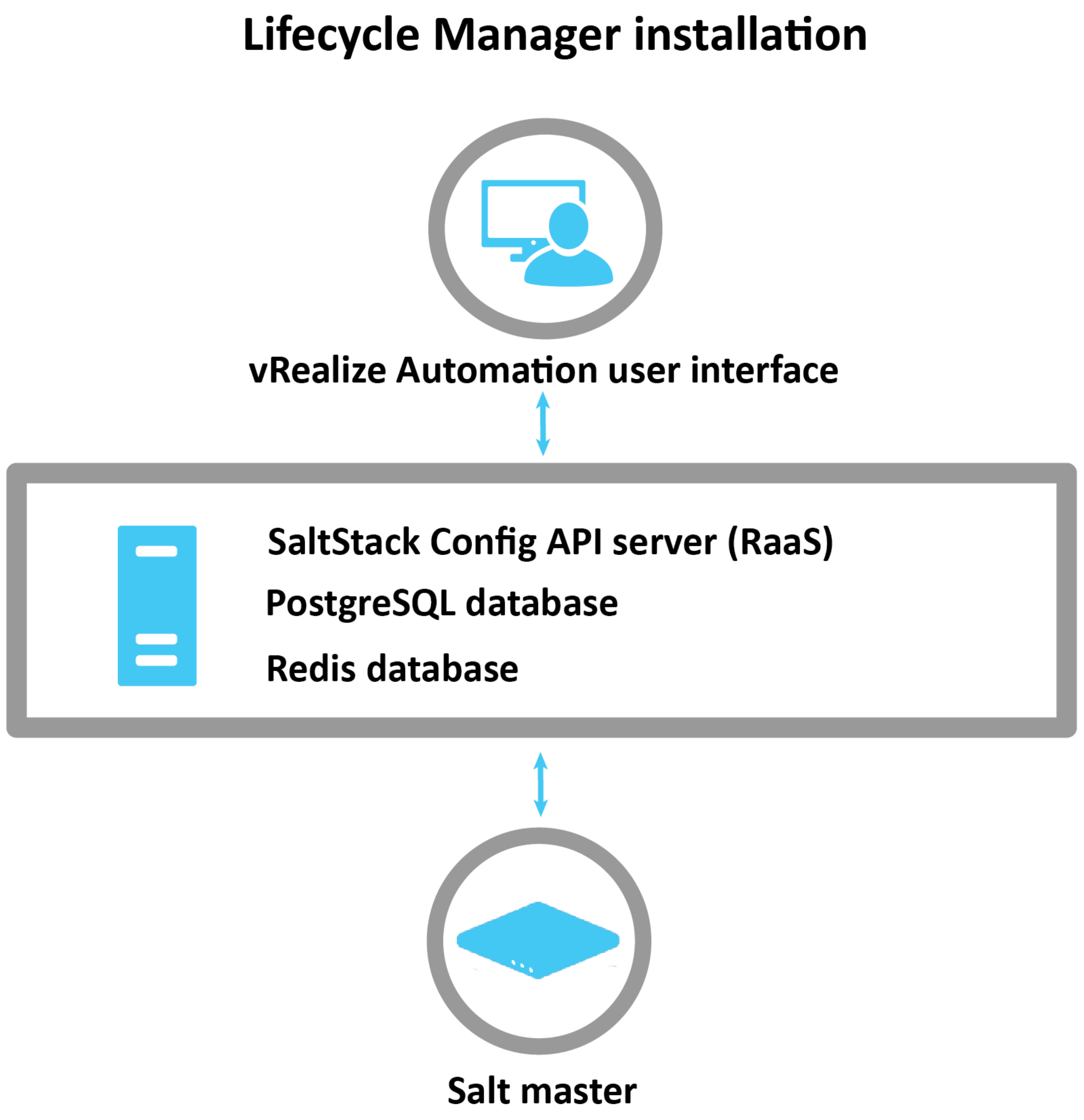
- **Standard installation** - Installs the architectural components needed for SaltStack Config in four or more separate nodes.
- **vRealize Suite Lifecycle Manager (vRLCM) installation** - Installs SaltStack Config and all of its architectural components on a single node. This method also installs the Salt master host and configures a required vRealize Automation property group.

Caution If you are unsure which installation method is best for your system, the standard installation is recommended. The vRealize Suite Lifecycle Manager installation method is not recommended for production grade systems with more than 1,000 nodes.

The following image shows the system architecture that you'll have after you complete a standard installation but before you complete the post-installation or Salt installation steps:


Standard installation
SaltStack Config user interface
PostgreSQL database
Redis database
SaltStack Config API server (RaaS)
Salt master

The following image shows the system architecture that you'll have after you complete a Lifecyle Manager (vRLCM) installation but before you complete the post-installation or Salt installation steps:


Lifecycle Manager installation
vRealize Automation user interface
SaltStack Config API server (RaaS)
PostgreSQL database
Redis database
Salt master

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

- Check the RaaS configuration file
- Log in for the first time and change default credentials
- Accept the Salt master key and back up data
- Set up SSL certificates
- Configure SaltStack SecOps (optional)
- Set up Single Sign-On (SSO) (optional)

Integrate with vRealize Automation

In this phase, you create an integration in vRealize Automation to access the SaltStack Config service in your network.

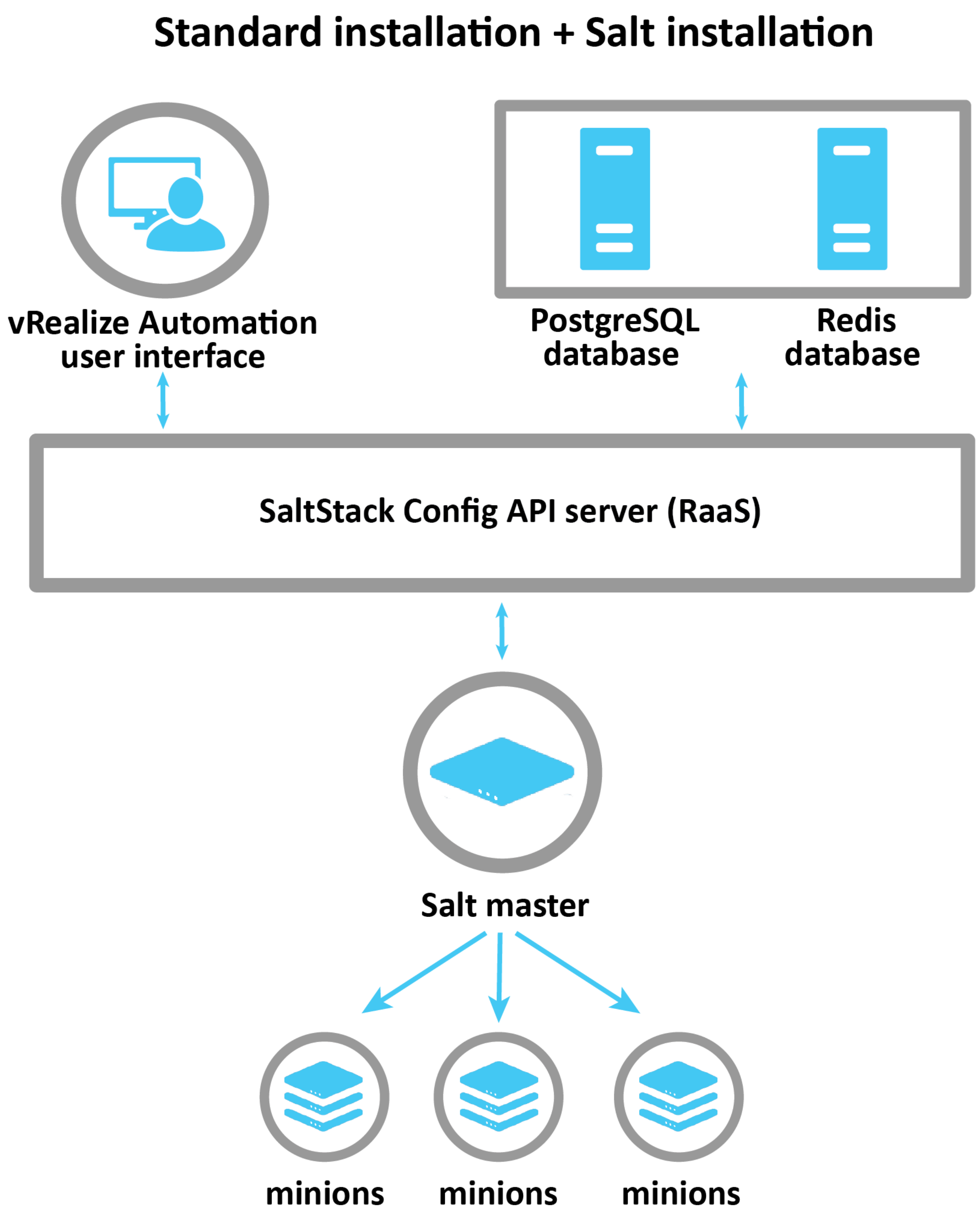
Install Salt on the nodes you want to manage with SaltStack Config

SaltStack Config runs on Salt, an open-source automation and configuration management engine.

In order to begin using SaltStack Config for configuration management, you also need to install and run the Salt minion service on any nodes that you intend to manage using SaltStack Config.

You can deploy the Salt minion service to your nodes using either vRealize Automation cloud templates or by installing the service through Secure Shell (SSH).

The following image shows the system architecture that you'll have after you complete the standard installation, post-installation, integration, and Salt installation steps:


Standard installation + Salt installation
vRealize Automation user interface
PostgreSQL database
Redis database
SaltStack Config API server (RaaS)
Salt master
minions
minions
minions

Supported operating systems 2

This list of supported operating systems refers to the requirements for the RaaS node, which provides the core functionality for SaltStack Config. It does not refer to the operating systems for the nodes running Salt in your network.

Supported operating systems for Salt

SaltStack Config runs on Salt, an open-source automation and configuration management engine. In order to begin using SaltStack Config for configuration management, you also need to install and run the Salt minion service on any nodes that you intend to manage using SaltStack Config.

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 operating systems, see Salt Platform Support.

Supported operating systems for SaltStack Config

The architecture for SaltStack Config is best designed to operate on either:

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

Important 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

Additional supported operating systems

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)

Installing SaltStack Config on these operating systems will require a manual installation method that should only be completed with a SaltStack Config expert.

System architecture requirements 3

The system architecture required to install SaltStack Config depends on two main factors: 1) the installation method you are using to deploy SaltStack Config and 2) your environment's throughput, meaning the amount of work that you will perform on your system using SaltStack Config.

Before you start

In order to make an accurate assessment of your system architecture needs, you should first ensure you are familiar with:

- The two available installation methods for SaltStack Config
- The four basic components of the SaltStack Config SaltStack architecture (RaaS, the Salt master, PostgreSQL, and Redis)

See Chapter 1 Installing and Configuring SaltStack Config for an overview of these concepts, including a general overview of the installation process. Also see Which installation scenario should you use? for guidance selecting an installation scenario.

Note SaltStack Config is powered by Salt, an open-source automation and configuration management engine. If you are less familiar with key terms used in Salt (such Salt master and Salt minion), see Salt system architecture for more information.

Determining minion architecture

In the context of SaltStack Config, a minion generally refers to a node in your production environment that connects with and is managed by SaltStack Config through one or more Salt masters.

Salt is designed to work with any operating system that might be running on a minion. In addition to the standard operating systems (Linux, Windows, MacOS), Salt provides specialized minion software (generally referred to as "native minions") for operating systems that are unique to various network devices such as Arista, Juniper, AIX, and Solaris.

This table lists the minimum memory requirements for the Salt minion service by operating systems:

Operating system	Minimum memory requirements
AIX minion	512 MB RAM
MacOS minion	4 GB RAM
Linux minion	512 MB RAM
Windows minion	4 GB RAM
Other network devices, including proxy minions	40 MB RAM per controlled device

Determine your installation scenario

See Which installation scenario should you use? for guidance selecting an installation scenario. If you are unsure which installation method is best for your system, the standard installation is recommended. The Lifecycle Manager installation method is not recommended for production grade systems with more than 1,000 nodes.

If you choose a Lifecycle Manager installation, it requires only one node and needs the following system architecture:

Hardware	Up to 1,000 nodes (minions)
Cores	16 CPU cores
RAM	32 GB RAM
Disk space	At least 260 GB free space

The remainder of this guide will explain architecture needs for the standard installation scenario.

Estimate the number of Salt minions you will manage

Although the throughput of your system is difficult to measure prior to installation, you can estimate your needs based on the number of minions (nodes) in your system that will be managed by SaltStack Config. The last section of this guide provides additional measurements for determining your system throughput.

As you bring more Salt minions under management by SaltStack Config, you might need to increase your system architecture to match.

Estimate the number of Salt masters you need

This table lists the recommended number of Salt masters you might need based on the number of managed Salt minions (nodes) in your system:

Minions	Salt masters (16 CPU/16 GB)
5,000	1
10,000	2
15,000	3
20,000	4
25,000	5
30,000	6
35,000	7
40,000	8
45,000	9
50,000	10
55,000	11
60,000	12
65,000	13
70,000	14
75,000	15
80,000	16
85,000	17
90,000	18
95,000	19
100,000	20

Note This document describes SaltStack Config on-premises architecture. SaltStack Config Cloud can currently only run one Salt master, which means it is limited to less than 20,000 minions.

Estimate the number of RaaS nodes you need

This table lists the recommended number of RaaS nodes you might need based on the number of managed Salt minions (nodes) in your system:

Minions	RaaS nodes with 16 CPU/16 GB	RaaS nodes with 32 CPU/32 GB
5,000	1	
10,000	1	
15,000	1	
20,000	1	
25,000	2	
30,000	2	
35,000	2	
40,000	2	
45,000	1	1
50,000	1	1
55,000	1	1
60,000	1	1
65,000		2
70,000		2
75,000		2
80,000		2
85,000	1	2
90,000	1	2
95,000	1	2
100,000	1	2

Estimate the number of PostgreSQL nodes you need

The next two tables list the recommended number of PostgreSQL database nodes you might need based on the number of managed Salt minions (nodes) in your system:

Minions	PostgreSQL nodes with 8 CPU/8 GB	PostgreSQL nodes with 16 CPU/16 GB	PostgreSQL nodes with 24 CPU/24 GB	PostgreSQL nodes with 32 CPU/32 GB
5,000	1			
10,000	1			
15,000	1			

Minions	PostgreSQL nodes with 8 CPU/8 GB	PostgreSQL nodes with 16 CPU/16 GB	PostgreSQL nodes with 24 CPU/24 GB	PostgreSQL nodes with 32 CPU/32 GB
20,000		1		
25,000		1		
30,000		1		
35,000			1	
40,000			1	
45,000			1	
50,000				1
55,000				1
60,000				1

Minions	PostgreSQL nodes with 48 CPU/48 GB	PostgreSQL nodes with 56 CPU/56 GB	PostgreSQL nodes with 64 CPU/64 GB
65,000	1		
70,000	1		
75,000	1		
80,000	1		
85,000		1	
90,000		1	
95,000			1
100,000			1

Estimate the number of Redis nodes you need

The next two tables list the recommended number of Redis database nodes you might need based on the number of managed Salt minions (nodes) in your system:

Minions	Redis nodes with 4 CPU/4 GB	Redis nodes with 8 CPU/8 GB	Redis nodes with 12 CPU/12 GB
5,000	1		
10,000	1		
15,000	1		
20,000	1		

Minions	Redis nodes with 4 CPU/4 GB	Redis nodes with 8 CPU/8 GB	Redis nodes with 12 CPU/12 GB
25,000		1	
30,000		1	
35,000		1	
40,000		1	
45,000			1
50,000			1
55,000			1
60,000			1

Minions	Redis nodes with 16 CPU/16 GB	Redis nodes with 20 CPU/20 GB
65,000	1	
70,000	1	
75,000	1	
80,000	1	
85,000		1
90,000		1
95,000		1
100,000		1

Optimize your architecture after installation based on throughput

After you've completed your SaltStack Config installation, you can use system monitoring metrics to better determine your system's throughput and architectural needs.

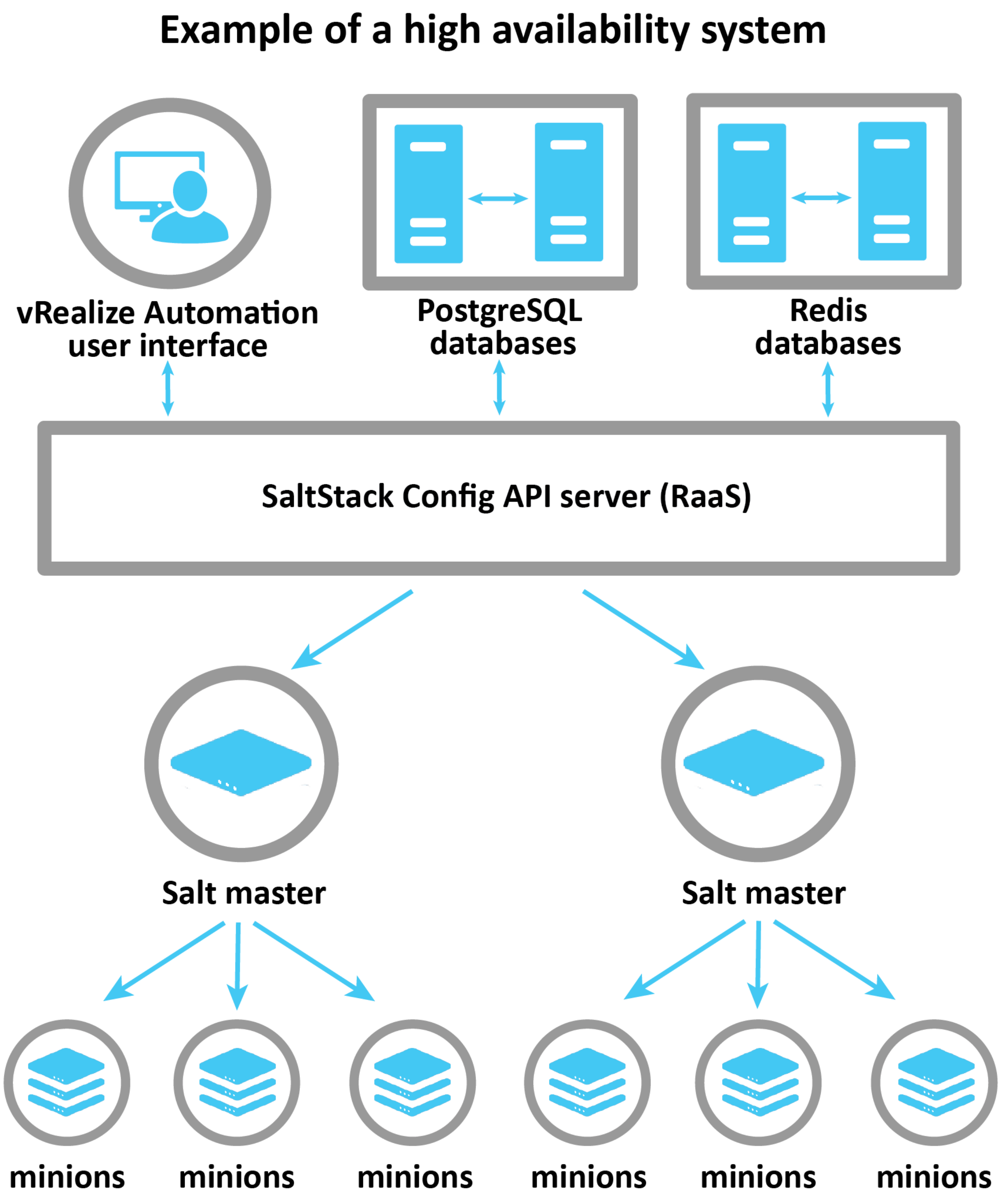
When determining what to monitor, consider these factors:

- **Amount of traffic on the event system** - The event system (also sometimes referred to as the "event bus") is used for inter-process communication by both Salt master and Salt minions. If your event bus is very busy, consider inceasing your memory allocations.
- **Job returns per hour** - SaltStack Config uses the term "jobs" to refer to each of the commands, tasks, and operations performed by SaltStack Config. Each job sends its output to SaltStack Config for reporting and data collection purposes. The number of job returns your system produces in a given hour can impact your architecture needs.

- **Amount of pillar data** - Pillar data is data that must be stored on the Salt master. 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. The amount of data being stored on your Salt master (and later accessed by minions as needed) can impact your memory and data storage needs.
- **Amount of custom grains** - Grains are used in Salt to target the minions for a particular job or command. Grains refer to the basic data and characteristics of each minion. Salt comes with many pre-built grains. For example, you can target minions by their operating system, domain name, IP address, kernel, memory, and many other system properties. You can also create custom grain data to distinguish one group of minions from another based on a characteristic you uniquely target for in your system. The number of custom grains you create can impact your architecture needs.
- **Number of beacons and reactors** - The beacon system is a monitoring tool that can listen for a variety of system processes on Salt 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. 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. If your system has beacons and reactors that are activated regularly, that could increase your system architecture needs.
- **Disk size needs** - You might need to increase your disk size based on the number of minions being managed and for each year's worth of data you need to keep in storage. For example, if you have a high-throughput system and your system is in a highly-regulated industry that requires 7-8 years of data retention, that might require higher disk size and storage capacity.
- **Geographical location and distance between components** - You might experience issues if there is a 65 ms latency or more between the Salt master and the server that is running SaltStack Config (RaaS). Fortunately, Salt is less sensitive to latency between the Salt minion and Salt master. When placing these components, keep in mind that it is better to locate the master close to RaaS and minion farther away if necessary.
- **Business-critical operations** - When assessing how business-critical SaltStack Config is in your environment, ask yourself how severe the impact of a SaltStack Config outage would be to your business. If it were down for an hour or more, would it cause a severe impact? If so, you might need to design high availability needs into your SaltStack Config system architecture.

Based on these factors, consider incrementally increasing your resources and monitor the impact to your system's performance. For example, increasing your memory allocations by 4GB RAM with 4 CPUs.

The following image shows an example of a high availability SaltStack Config architecture design:


Example of a high availability system
vRealize Automation user interface
PostgreSQL databases
Redis databases
SaltStack Config API server (RaaS)
Salt master
Salt master
minions
minions
minions
minions
minions
minions

As this image illustrates, many high availability systems connect to multiple Salt masters. High availablility systems also often build redundancy into the PostgreSQL database and Redis databases so that one can fail over to another. Keep in mind that the current high availability solutions for PostgreSQL and Redis only support manual failovers.

Pre-installation

4

Before you begin the installation process for SaltStack Config, you must complete some pre-installation tasks. These tasks are intended to be followed in a specific order as listed in this guide.

This chapter includes the following topics:

Pre-installation planning

Before you begin your SaltStack Config installation, you need to make several key decisions and ask yourself some important questions about your installation project. Run through this checklist to plan your installation and receive guidance while making these decisions.

Note As part of VMware's initiative to remove problematic terminology, the term Salt master will be replaced with a better term in SaltStack Config and related products and documentation. This terminology update may take a few release cycles before it is fully complete.

Before you start

The various sections 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 Chapter 1 Installing and Configuring SaltStack Config.

Which installation scenario should you use?

You can install the SaltStack Config service in your vRealize Automation environment using one of two installation methods. Determining the appropriate installation method for your environment depends on a few different factors.

The two installation methods available for SaltStack Config are:

- **Standard installation** - Installs the architectural components needed for SaltStack Config in four or more separate nodes.

- **vRealize Suite Lifecycle Manager (vRLCM) installation** - Installs SaltStack Config and all of its architectural components on a single node. This method also installs the Salt master host and configures a required vRealize Automation property group.

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?

Note vRealize Suite Lifecycle Manager installations require a vRealize Automation license. See Which license do you need? for more information about licensing.

Standard installation overview

In the standard installation, 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:

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

In the standard installation, you run an orchestration highstate provided in the installer. The highstate runs on your Salt 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.

Use the multi-node installation scenario if:

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

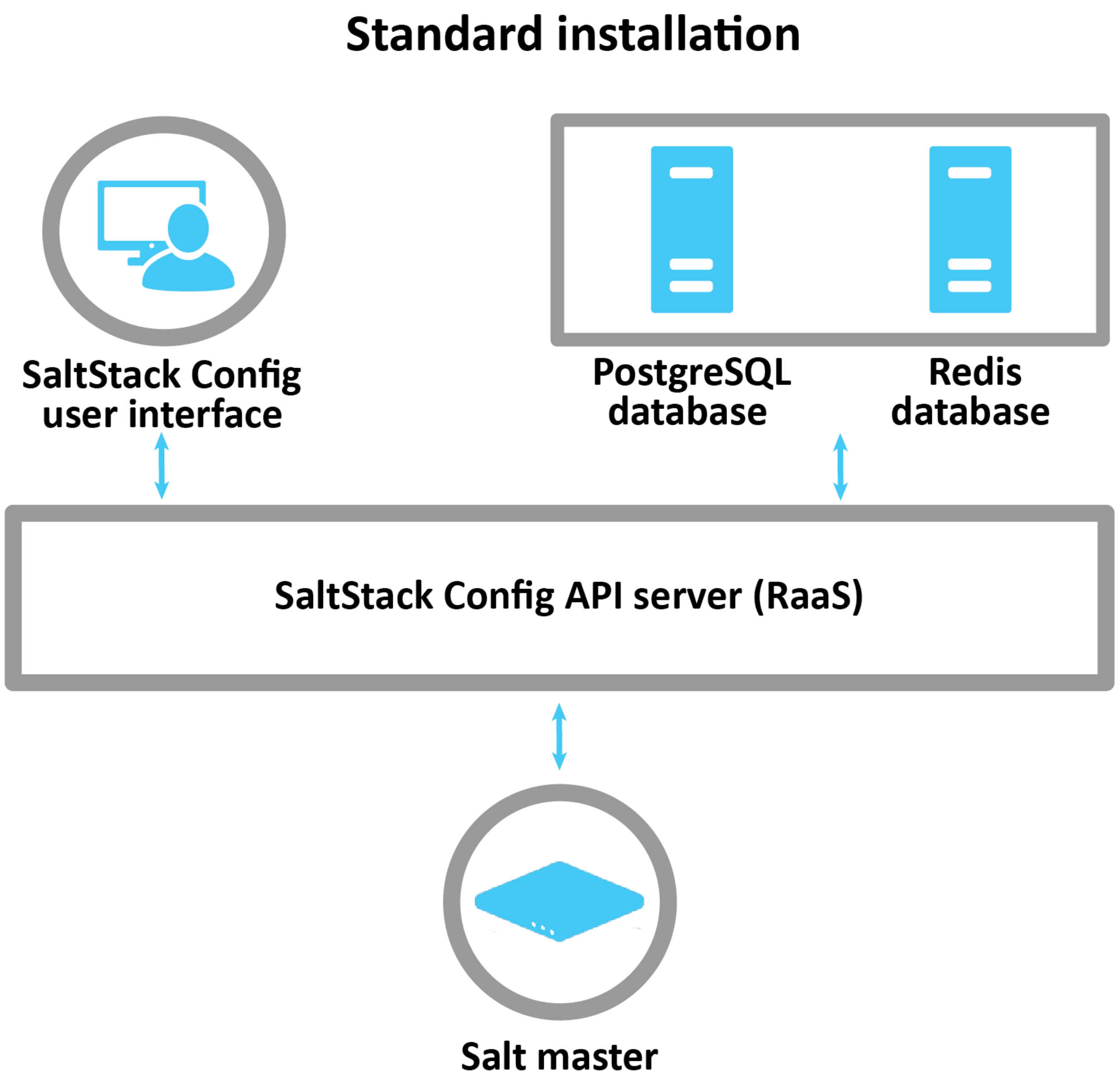
The advantages of the standard installation scenario are:

- It can scale as your network grows.
- It is not dependent on the availability of a single node for functionality.
- This installation method 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. You may want to consider using professional services from an expert who has deployed SaltStack Config in a production environment before.
- If your network has high availability needs, you might need support and/or consultation services from an expert who has deployed SaltStack Config in a production environment before.
- The installation process may take a full day or two to complete, depending on the size and complexity of your network.

The following image shows the system architecture that you'll have after you complete a standard installation but before you complete the post-installation or Salt installation steps:


Standard installation
SaltStack Config user interface
PostgreSQL database
Redis database
SaltStack Config API server (RaaS)
Salt master

vRealize Suite Lifecycle Manager installation overview

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

Use the Lifecycle Management installation if:

- Your network has 1,000 nodes or less that you intend to manage using SaltStack Config
- 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 standard installation.)

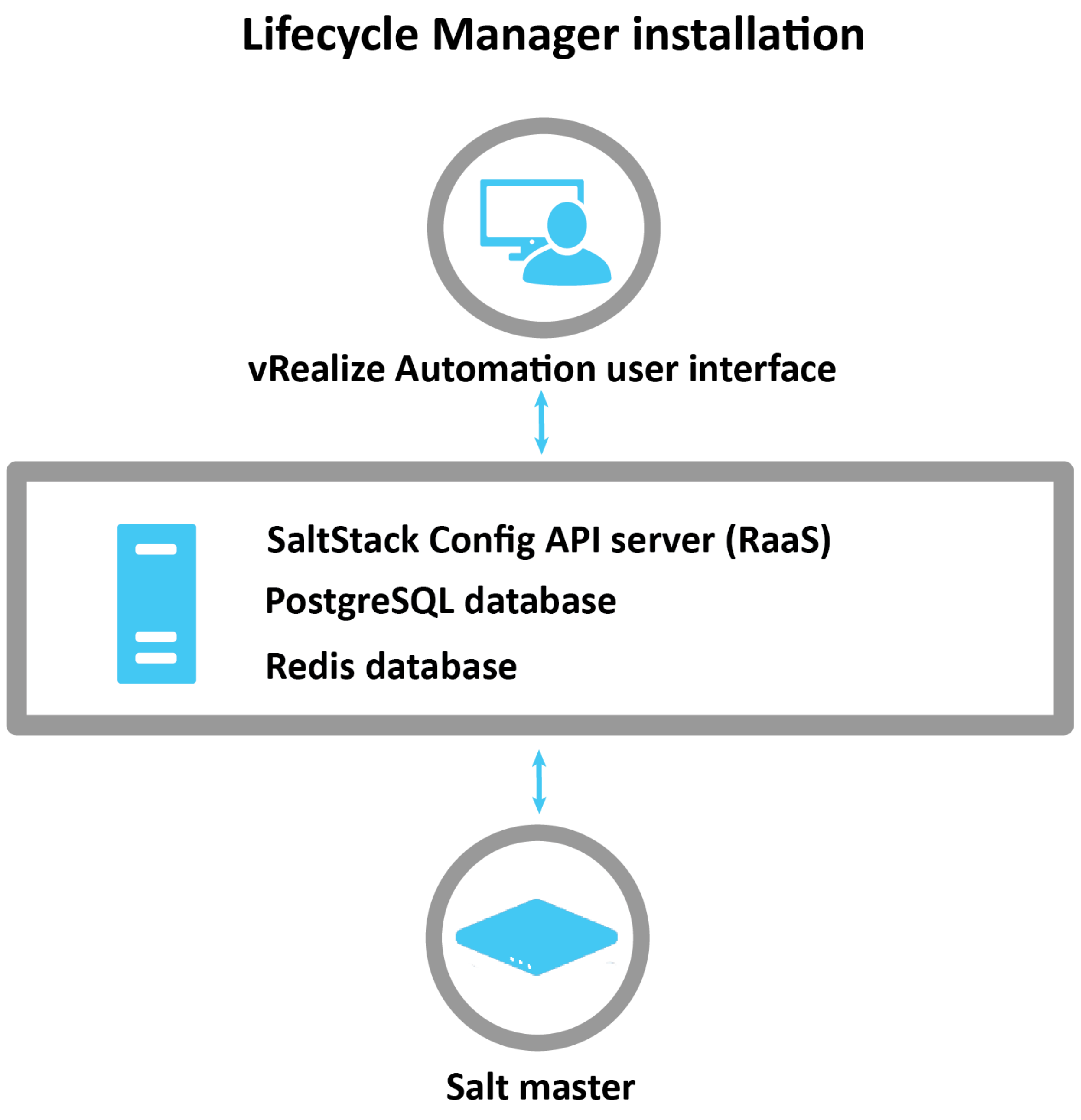
The advantages of the Lifecycle Management installation scenario are:

- 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 process only takes a few hours, depending on the size of your network.

The disadvantages are:

- This installation method 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.

The following image shows the system architecture that you'll have after you complete a Lifecycle Manager (vRLCM) installation but before you complete the post-installation or Salt installation steps:


Lifecycle Manager installation
vRealize Automation user interface
SaltStack Config API server (RaaS)
PostgreSQL database
Redis database
Salt master

Which license do you need?

The type of license you need for your installation depends on whether you are integrating SaltStack Config with vRealize Automation or not.

If your deployment of SaltStack Config is integrated with vRealize Automation, you need one of the following licenses:

- vRealize Automation Cloud
- vRealize Automation Suite Advanced
- vRealize Automation Suite Enterprise

If your deployment of SaltStack Config is not integrated with vRealize Automation, you need:

- vRealize Automation Standard Plus

Using more than one license key

SaltStack Config supports multiple license keys, which means you can add any number of license key numbers to your SaltStack Config installation. As long as a license key has not yet expired,SaltStack Config activates any features allowed by that license key. See Install the license key for more information.

vRealize Automation SaltStack SecOps licenses

SaltStack SecOps requires a license with the SecOps feature enhancement. Customers with vRealize Automation Suite Advanced, Suite Enterprise, or Cloud license are eligible for this feature enhancement. To add SaltStack SecOps to your vRealize Automation license, contact your sales representative.

Note You'll add your license key to SaltStack Config as one of the post-installation steps. See Install the license key for more information.

Which operating system do you need?

The architecture for SaltStack Config is best designed to operate on either RedHat 7.4 or higher (RHEL 7) or CentOS 7 (CentOS7). Additional operating systems are supported, but not recommended.

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 operating systems, see Salt Platform Support.

For more detailed information about additional operating systems or considerations, see Chapter 2 Supported operating systems.

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 guide on PostgreSQL tuning, see Tuning your PostgreSQL Server for SaltStack Config.

Which version of Redis do you need?

SaltStack Config requires a Redis 5.x database, but Redis 5.0.4 is recommended. The recommended version of Redis is included with the SaltStack Config installer.

Redis is a third-party, open source, in-memory data structure store. It is required for SaltStack Config. Because this is third-party software, be aware of the following:

- You are responsible for ongoing maintenance and other administrative tasks. For information about Redis database maintenance and administration, see the Redis documentation.
- Consider getting guidance from your organization's database administrator, if possible.

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.

Plan how to transfer the installation files

In order to complete a standard 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. For a standard installation, transfer the files to the Salt master from which you are running the installation orchestration.

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 SecOps libraries

If your organization has a SecOps license, be aware that both SaltStack SecOps libraries release regular content 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 through 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.

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 Salt master.

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

Do you need to install Salt prior to installation?

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 a standard SaltStack Config installation scenario.

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

- Installing Salt on the nodes involved in the SaltStack Config installation in the standard installation scenario.
- Installing Salt on the infrastructure that will eventually be managed by SaltStack Config.

For instructions about how to install Salt and its dependencies on the nodes involved in a standard insallation, see Install or upgrade Salt (pre-installation).

Installing Salt in your infrastructure

You are strongly encouraged to install Salt beforehand on any infrastructure that you plan to use SaltStack Config to manage. 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.

Consult these guides to ensure your environment is following best practices when implementing Salt in your infrastructure:

- Salt Best Practices

- Salt Hardening Guide

Installing Salt in an air-gapped system

The one exception to the general recommendation to install Salt beforehand is when you are installing SaltStack Config in an air-gapped system. Be aware that there are 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 Lifecycle Manager installation scenario, if it detects the Salt master service and 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 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.

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 (pre-installation).

Important Certain Salt dependencies must be installed in order to prevent a failure in either installation scenario. To verify that these dependencies are installed, see Install or upgrade Salt (pre-installation).

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 changes that occur to your Salt environment when installing SaltStack Config.

The following changes will impact your Salt environment during installation:

- 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 product documentation.

- RaaS replaces the Salt master syndic component to provide minion aggregation and scale. Salt Syndic Salt masters are not compatible with the SaltStack Config architecture. Instead, each root Salt 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.

What firewall permissions are needed?

SaltStack Config requires firewall access to various ports on certain nodes for a standard installation.

For standard 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	■ Salt controllers ■ Web-based interface users ■ Remote systems calling the Enterprise API
Salt controllers	4505/4506	All minions configured to use the related Salt master

Install or upgrade Salt (pre-installation)

In order to prepare your machines for a standard installation of SaltStack Config, you need to install or upgrade Salt and Python. Salt and Python needs to be present and updated on all nodes that are involved in the installation. The installation will fail if Salt and the installer's dependencies are not installed on your nodes.

Note These steps are part of the pre-installation process for a standard installation of SaltStack Config. This guide explains how to install or upgrade Salt on the node that you are using to run an installation. For instructions on how to install Salt on the rest of your infrastructure after the standard installation, see Chapter 8 Install Salt on your infrastructure.

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?.

Before you start

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:

- Installation overview

- Pre-installation planning

Install the Salt dependencies (pre-installation)

The SaltStack Config installer requires a few important packages in order to run correctly. If you don't install these dependencies, the SaltStack Config installer will fail during either installation scenarios.

Note These steps are part of the pre-installation process for a standard installation of SaltStack Config. This guide explains how to install or upgrade Salt on the node that you are using to run an installation. For instructions on how to install Salt on the rest of your infrastructure after the standard installation, see Chapter 8 Install Salt on your infrastructure.

These dependencies must be installed on all nodes that are involved in the installation. In a standard installation, you must install these dependencies on all nodes that will host the Salt master, RaaS, the Redis database, and the PostgreSQL database:

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

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
```

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

What to do next

After all dependencies are installed, see Install Salt (pre-installation).

Install Salt (pre-installation)

Ensure that Salt is installed on any nodes that are directly involved in your SaltStack Config installation or else the installation will fail. In a standard installation, these are the four nodes that will host the Salt master, RaaS, the Redis database, and the PostgreSQL database.

Note These steps are part of the pre-installation process for a standard installation of SaltStack Config. This guide explains how to install or upgrade Salt on the node that you are using to run an installation. For instructions on how to install Salt on the rest of your infrastructure after the standard installation, see Chapter 8 Install Salt on your infrastructure.

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 (pre-installation).

Installing Salt on your four installation nodes involves three main tasks:

- Install Salt on the Salt master(s)
- Install Salt on the minions
- Accept the minion keys on the Salt master(s)

The following sections explain how to do these tasks.

Install Salt on the Salt master(s)

In a standard installation, you need to install both the Salt master service and the minion service on the Salt master host.

These instructions install 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.

1 Install the Salt project repository and key:

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

2 Clear the cache:

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

3 Install the Salt master service and the minion service on the Salt master node:

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

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

```
master:localhost
```

5 Start the Salt master service and minion service:

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

Use `servicesalt-minionrestart` to restart the minions if needed.

Install Salt on the Salt minions

After installing Salt on the Salt master as described in the previous section, the next step is to install the minion service (not the 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 Salt master. For more detailed information about installing the minion service, see Minion Configuration in the Salt documentation.

To install the minion service:

1 Install only the 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 Salt master by editing the `master.conf` file in the `/etc/salt/minion` directory. In this file, provide the Salt master’s IP address. For example:

```
master:192.0.2.1
```

4 Start the minion service:

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

Use `servicesalt-minionrestart` to restart the minions if needed.

5 Repeat the previous steps for all remaining nodes.

Accept the minion keys on the master(s)

At this point, you have installed the Salt master service and minion service, and you have provided your minions with the Salt master’s IP address. Now, in order for the Salt master to send commands to the minions, the next step to accept the minion keys on the Salt 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 Salt master's terminal, list all the keys that are on Salt master:

```
salt-key -L
```

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

   If the minion IDs appear in `Acceptedkeys`, 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>
```

   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 `Acceptedkeys`.

What to do next

After all minion keys are accepted, you have successfully installed Salt and can proceed to the next pre-installation step. 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.

Upgrade Salt and Python (pre-installation)

If you are installing SaltStack Config on an existing Salt infrastructure, Salt is already installed. In this case, you'll instead need to ensure that Salt and Python are upgraded to the latest version.

Note These steps are part of the pre-installation process for a standard installation of SaltStack Config. This guide explains how to install or upgrade Salt on the node that you are using to run an installation. For instructions on how to install Salt on the rest of your infrastructure after the standard installation, see Chapter 8 Install Salt on your infrastructure.

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.

What to do next

After Salt and Python are upgraded, you can proceed to the next pre-installation step. 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.

Transfer and import files

The final pre-installation step for SaltStack Config is to download the installation files and send them to the Salt master node.

Before you start

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

- Installation overview
- Pre-installation planning
- Install or upgrade Salt (pre-installation)

Download files

Download the installation files from Customer Connect. For assistance, contact your sales representative.

Transfer the files

Transfer the files to the Salt master node from which you are running the installation orchestration. Place the files in the root folder.

Verify the installation files

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

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 Customer Connect.

Extract the files (optional)

Extracting the installation files is an optional step. It is only required if you downloaded the .zip version of the 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:

```
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_Config-<version>_Installer.zip
```

Import the .asc keyfiles

You need to import the .asc keyfiles from the installer into the RPM packaging system. To import the keyfiles on the all the nodes involved in the installation:

1 Navigate to the `ssc-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.

What to do next

Once you have downloaded, transferred, verified, and imported the installation files, you can begin the installation process.

Installation 5

SaltStack Config supports two installation methods.

For an overview of each installation method and guidance about which method is right for your system, see Which installation scenario should you use?

This chapter includes the following topics:

- Standard installation
- vRealize Suite Lifecycle Manager installation

Standard installation

In the standard installation, you run an orchestration highstate provided in the SaltStack Config installer. The highstate runs on your Salt master and sets up a multi-node environment. It installs the core SaltStack Config architecture on the three other nodes that will host PostgreSQL, Redis, and the RaaS node.

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 Salt master:

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

Note As part of VMware's initiative to remove problematic terminology, the term Salt master will be replaced with a better term in SaltStack Config and related products and documentation. This terminology update may take a few release cycles before it is fully complete.

In the standard installation, you run an orchestration highstate designed by VMware. The highstate runs on your Salt 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.

High availablility

It is possible to set up multiple Salt 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 in this guide.

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.

For more information about high availability, see Chapter 3 System architecture requirements.

Before you start

Before you begin the installation process, ensure you have read and completed the pre-installation tasks:

- Installation overview
- Pre-installation planning
- Install or upgrade Salt (pre-installation)
- Transfer and import files

For a standard installation, it is especially important to follow all the steps listed in the Install or upgrade Salt (pre-installation) page. In particular, you **must** install the dependencies needed for the SaltStack Config installer on all four nodes in the installation. Otherwise, the installation will fail. Remediating a failed installation may require in-depth troubleshooting. If you need assistance, Chapter 11 Contact Support.

Record key data about the four nodes

Before beginning the standard installation, record the key data about each of the four nodes involved in the installation. You will input this data at several points during the installation process.

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 Salt 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 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 Salt master. By default, the minion identifies itself to the Salt 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 Salt master. For specific instructions on setting up the keys, see Accept the minion keys on the master(s).

What to do next

After recording this key data, read and follow the steps in Copy and edit the top state files.

Copy and edit the top state files

During this installation task, you copy the orchestration files provided with the SaltStack Config installer to the Salt 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 Salt master, follow the instructions in Transfer and import files.

To copy and edit the orchestration configuration files:

1. On the Salt 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/
   ```

   Important 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 Salt 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 Salt 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 }}':
    - sse.eapi_plugin
```

What to do next

After editing the top state files, read and follow the steps in Edit the SaltStack Config settings pillar file.

Edit the SaltStack Config settings pillar file

During this installation task, 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 Salt 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 Salt 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 Salt masters (and minions) are operating in failover mode.
- The `eapi_key` variable defines the encryption key thatSaltStack 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 Salt masters when it is configured in either Active or Failover Multi-Master mode. This ID prevents minions that are reporting to multiple Salt masters from being reported multiple times within the SaltStack Config.

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

What to do next

After editing the pillar file, read and follow the steps in Apply the highstates to the nodes.

Apply the highstates to the nodes

During this installation task, you refresh your system data and run the orchestration that configures all the components of SaltStack Config.

Caution Before running the highstate, it is especially important to follow all the steps listed on the Install or upgrade Salt (pre-installation) 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 Chapter 11 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 Salt master, sync your grains to confirm that the Salt 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 Salt 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 Salt 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 Salt master node

Note During the initial application of the highstate to the Salt master, you may see the following error message: `Authenticationerroroccurred`. This error displays because the Salt 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 Chapter 10 Troubleshooting page or Chapter 11 Contact Support.

What to do next

After applying the highstate, the standard installation process is complete. Now, 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
- Configure SaltStack SecOps (optional)
- Set up Single Sign-On (SSO) (optional)

vRealize Suite Lifecycle Manager installation

The vRealize Suite Lifecycle Manager installation method installs SaltStack Config on a single node (server). After installation, a Salt master, SaltStack Config, a Redis database, and a PostgreSQL database all run on this same node.

As it runs, the vRealize Suite Lifecycle Manager installer for SaltStack Config:

- 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 Salt master.
- Applies the Salt states needed to install SaltStack Config.
- Installs the required versions of PostgreSQL, Redis, and Python Setuptools on the server.

Before you start

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 (pre-installation)
- Transfer and import files

Install vRealize Automation

In addition to these prerequisites, you must have a pre-existing installation of vRealize Automation before you begin the vRealize Suite Lifecycle Manager installation process.

See Which license do you need? for more information about licensing.

Use the vRealize Suite Lifecycle Manager installer

For a vRealize Suite Lifecycle Manager installation, see the following topics in VMware vRealize Suite Lifecycle Manager documentation:

- Configure product details
- Creating an Environment in vRealize Suite Lifecycle Manager
- Create a New Private Cloud Environment Using the Installation Wizard

What to do next

Once this 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
- Configure SaltStack SecOps (optional)
- Set up Single Sign-On (SSO) (optional)

Post-installation

6

After completing the SaltStack Config installation process using either installation method, you must complete some post-installation tasks. These tasks are intended to be followed in a specific order as listed in this guide.

This chapter includes the following topics:

- Install the license key
- Install and configure the Master Plugin
- Check the RaaS configuration file
- Log in for the first time and change default credentials
- Accept the Salt master key and back up data
- Set up SSL certificates
- Set up Single Sign-On (SSO)
- Configure SaltStack SecOps

Install the 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 a sales representative to purchase a license and then install the key.

Before you start

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.

Which license do you need?

See Which license do you need? for more information about licensing.

SaltStack Config supports multiple license keys, which means you can add any number of license key numbers to your SaltStack Config installation. As long as a license key has not yet expired,SaltStack Config activates any features allowed by that license key.

Installing the key

SaltStack Config supports the legacy method of installing the license key by adding a RaaS license key file to the RaaS node. For instructions on installing the license key using the legacy method, see Install the RaaS license key.

To install the license key:

1 Get your license key number from My VMware (Customer Connect) or contact your sales representative for the license key number. Record this key number for use in later steps.

2 Create a file with a filename ending in `_license`, such as `vra_license` or `saltstack-config_license`.

   Note If you have multiple license key numbers, create a separate file for each license. Each license filename must end in `_license` in order to be recognized as a valid license file by RaaS.

3 Edit the file and add your license key number. Save the file.

4 Copy the file to the `/etc/raas/` file directory on the RaaS node using the following commands (replacing the placeholder filename with your license filename):

   ```
   chown raas:raas yourfilename_license
   mv raas.license /etc/raas
   ```

   Note If you have multiple license key numbers, repeat the previous steps to create a separate license file for each key number and then upload the files to the RaaS node.

5 **OPTIONAL:** If you want RaaS to recognize the new license file immediately, restart the RaaS service:

   ```
   systemctl restart raas
   ```

   If you don't restart the RaaS service, RaaS automatically refreshes and detects the new license file regularly within a one hour time frame.

What to do next

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.

Install and configure the Master Plugin

As part of the post-installation process, you need to install, configure, and upgrade the Master Plugin. The Master Plugin enables your Salt masters to communicate with SaltStack Config. The Master Plugin includes a variety of settings you can adjust to improve performance, which are particularly useful for large or busy environments.

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

For more information about updating performance-related settings, see the Master Plugin page in the SaltStack Config product documentation.

Note As part of VMware's initiative to remove problematic terminology, the term Salt master will be replaced with a better term in SaltStack Config and related products and documentation. This terminology update may take a few release cycles before it is fully complete.

Before you start

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.

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 vRealize Suite Lifecycle Manager 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 Salt master node. However, the Master Plugin is installed only on the Salt 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 Chapter 9 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)

Install the Master Plugin

To install the Master Plugin on your Salt master:

1 Log in to your master.

2 If necessary, download the Master Plugin wheel from Customer Connect.

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

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

Ubuntu

```
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.

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.

Caution 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 SaltStack Config installer, your installation likely includes an offline package, called the Salt Crystal, that requires special upgrade instructions. For more information, see the Chapter 10 Troubleshooting page.

3 Edit the generated `raas.conf` file and update the values as follows to validate the certificate the API (RaaS) uses 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 Salt master to be offline for long periods of time and will need the Salt 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 **OPTIONAL:** Update performance-related settings. For large or busy environments, you can improve the performance of the communications between the Salt master and SaltStack Config by adjusting the following settings.

- Enable event queuing (available in salt 2019.2.2 and later). Events can be queued on the Salt master and sent to the event returner in a batched fashion using a single transaction for multiple events. By default, this queuing mechanism is disabled. To enable event queuing, set the following in your Salt Master Plugin configuration file:

```
event_return_queue:2500
event_return_queue_max_seconds:5
```

The suggested maximum event queue size is 2500 as shown. The queue will be flushed, forwarding events to the event returner, when the queue is full. A lower value may be a better fit for smaller or less busy environments.

In some cases, the Salt event bus may not busy enough to cause the queue to regularly reach its maximum size. Setting `event_return_queue_max_seconds` will cause the queue to be flushed when the oldest event in the queue is older than the configured value regardless of how many events are in the queue.

- Enable and configure the `eventqueue` and `rpcqueue` engines:

  These engines offload some communications with SaltStack Config from performance-critical code paths to dedicated processes. While the engines are waiting to communicate withSaltStack Config, payloads are stored in the Salt master's local filesystem so the data can persist across restarts of the Salt master.

  To enable the engines, uncomment the following settings in the Salt Master Plugin configuration file (`raas.conf`):

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

  To configure the `eventqueue` engine, uncomment and update the following settings:

  ```
  sseapi_event_queue:
    name: sseapi-events
    strategy: always
    push_interval: 5
    batch_limit: 2000
    age_limit: 86400
    size_limit: 35000000
    vacuum_interval: 86400
    vacuum_limit: 350000
    forward: []
  ```

  The queue parameters can be adjusted with consideration to how they work together. For example, assuming an average of 400 events per second on the Salt event bus, the settings shown above allow for about 24 hours of queued event traffic to collect on the Salt master before the oldest events are discarded due to size or age limits.

  To configure the `rpcqueue` engine, uncomment and update the following settings:

  ```
  sseapi_rpc_queue:
      name: sseapi-rpc
      strategy: always
      push_interval: 5
      batch_limit: 500
  ```

```
    age_limit: 3600
    size_limit: 360000
    vacuum_interval: 86400
    vacuum_limit: 100000
```

- Enable load caching:

  ```
  sseapi_local_cache:
      load:3600
  ```

  Note If the `rpcqueue` engine is enabled, load caching must also be enabled in order for the Salt master to handle jobs correctly.

- Limit minion grains payload sizes:

  ```
  sseapi_max_minion_grains_payload:2000
  ```

- Enable skipping jobs that are older than a defined time (in seconds). For example, use `86400` to set it to skip jobs older than a day. When set to `0`, this feature is disabled:

  ```
  sseapi_command_age_limit:0
  ```

  Note This is useful during upgrade, to prevent old commands stored in the database from running unexpectedly.

Together, event queuing in Salt and the queuing engines, load caching, grains payload size limit, and command age limit in the Salt Master Plugin increase the throughput and reduce the latency of communications between the Salt master and SaltStack Config in the most performance-sensitive code paths.

6 Restart the master service.

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

7 **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, such as if no minions are connected, this is likely a sign of a correct configuration.

Configuration settings reference

These settings in the configuration file enable each Salt master to connect to the API (RaaS). You can find these settings in the `/etc/salt/master.d/raas.conf` configuration file.

Important Salt master settings in the raas.conf file take precedence over existing settings in `/etc/salt/master`. If you have customized the `fileserver_backend` or `ext_pillar` settings in `/etc/salt/master`, you need to manually merge these settings so that they appear in one file only. You can optionally re-order the backends to change precedence.

The following table explains the general configuration settings:

Option	Description
`id`	Salt master ID, autogenerated if not set
`sseapi_server`	URL of SSEAPI server, e.g. `https://sse.example.com:443`
`engines`	Salt engines to enable, recommend `sseapi`, `jobcompletion`, `eventqueue`, and `jobcompletion`
`master_job_cache`	`sseapi` to use the SaltStack Config master job cache
`event_return`	Salt event returner, recommend `sseapi` to use the SaltStack Config event returner
`ext_pillar`	external pillar sources, recommended `sseapi`
`fileserver_backend`	file server backends, recommended `sseapi` and `roots`
`sseapi_update_interval`	how frequently to update from file server (seconds, default 60)
`sseapi_poll_interval`	how frequently to poll SaltStack Config for new data (seconds, default 30)
`sseapi_timeout`	timeout for API (RaaS) calls (seconds, default 200)
`sseapi_pubkey_path`	path to public key file for authenticating the Salt master to SaltStack Config
`sseapi_key_rotation`	Salt master SaltStack Config authentication key rotation interval (seconds, default 86400)
`sseapi_cache_pillar`	whether to cache pillar data within SaltStack Config (True or False, default False)
`sseapi_cluster_id`	(optional) Salt master cluster name, for grouping Salt masters into clusters within SaltStack Config
`sseapi_failover_master`	whether this Salt master is a failover Salt master (True or False, default False)
`sseapi_command_age_limit`	whether to skip API (RaaS) commands older than a defined time (seconds, 0 to disable, default 0)

The following table explains the SSL settings:

Option	Description
`sseapi_ssl_key`	path to the certificate's private key
`sseapi_ssl_cert`	path to the certificate
`sseapi_ssl_validate_cert`	whether to validate the SaltStack Config SSL certificate (True or False, default True)

The following table explains the Event Queue Engine settings, which appear under the `sseapi_event_queue` heading:

Option	Description
`name`	Event queue name (default `sseapi-events`, no need to change this)
`strategy`	When to queue events (`always`, `on_failure`, or `never`, default `never`)
`push_interval`	How often to push events to SaltStack Config (seconds, default 5)
`batch_limit`	Maximum number of events to push to SaltStack Config per interval (default 2000)
`age_limit`	Maximum queued event age; drop oldest events (seconds, default 86400)
`size_limit`	Maximum queue size; drop oldest events (events, default 35000000)
`vacuum_interval`	How often to vacuum the queue database (seconds, default 86400)
`vacuum_limit`	Maximum queue size when vacuuming the queue database (events, default 350000)

`forward`	Additional salt returners to send events to when flushing the queue (default none)
	The `forward` configuration item can specify a list of one or more Salt returner names. Each time the event queue is flushed, queued events will be sent to SaltStack Config as well as to each returner in this list. This enables the eventqueue engine's queueing and persistence functionality to be added to any standard Salt returner. The following is a configuration excerpt that causes events to be written to a local file each time they are sent to SaltStack Config:
	`sseapi_event_queue:`
	`  # ...other queue settings...`
	`  forward:`
	`    - rawfile_json`
	
	`# rawfile_json returner configuration`
	`rawfile_json.filename: /var/log/salt/events.json`

The following table explains the RPC Queue Engine settings, which appear under the `sseapi_rpc_queue` heading:

Option	Description
`name`	Event queue name (default `sseapi-rpc`, no need to change this)
`strategy`	When to queue events (`always`, `on_failure`, or `never`, default `never`)
`push_interval`	How often to send calls to SaltStack Config (seconds, default 5)
`batch_limit`	Maximum number of calls to push to SaltStack Config per interval (default 500)

Option	Description
`age_limit`	Maximum queued call age; drop oldest entries (seconds, default 3600)
`size_limit`	Maximum queue size; drop oldest entries (events, default 360000)
`vacuum_interval`	How often to vacuum the queue database (seconds, default 86400)
`vacuum_limit`	Maximum queue size when vacuuming the queue database (entries, default 100000)

The following table explains the Path settings. After initial configuration generation be careful changing these settings. Modules will be copied into these directories from the installation process. However, adding extra paths will not have an adverse effect.

Option	Description
`beacons_dirs`	beacons External Modules Path(s)
`engines_dirs`	engines External Modules Path(s)
`fileserver_dirs`	fileserver External Modules Path(s)
`pillar_dirs`	pillar External Modules Path(s)
`returner_dirs`	returner External Modules Path(s)
`roster_dirs`	roster External Modules Path(s)
`runner_dirs`	runner External Modules Path(s)
`module_dirs`	Salt External Modules Path(s)
`proxy_dirs`	proxy External Modules Path(s)
`metaproxy_dirs`	metaproxy External Modules Path(s)
`states_dirs`	states External Modules Path(s)

What to do next

After installing and configuring the Master Plugin, you must complete additional post-installation steps. The next step is to configure the RaaS node. To continue the post-installation process, see Check the RaaS configuration file.

Check the RaaS configuration file

As part of the post-installation process, consider reviewing your RaaS configurations. RaaS, which stands for Returner as a Service, is the central component in SaltStack Config. RaaS provides RPC endpoints to receive management commands from the SaltStack Config user interface, as well as RPC control endpoints to interface with connected Salt masters. The RaaS configuration settings are in the `/etc/raas/raas` configuration file on the RaaS node.

Before you start

Configuring the RaaS node 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

Check configuration settings

1 Open the RaaS configuration file on the RaaS node. By default, it is usually stored in `/etc/raas/raas`.

2 Check the following required settings:

Setting	Description
`customer_id`	Your customer ID, or sample UUID.
`sql`	`username`, `password`, `host`, and `port` can be configured to match your database configuration. For more on storing credentials securely, see the Securing credentials in your SaltStack Config configuration knowledge base article.

3 Check the following additional settings:

Setting	Description
`tls_crt`	Path to the `crt` file for encrypted communication. If this certificate is self-signed and should not be validated using a known CA, be sure to set the `sseapi_validate_cert` option to `False` in the Salt master configuration file.
`tls_key`	Certificate key file.
`port`	Port that is used for connections from the SaltStack Config user interface and Salt controllers.
`audit`	Include the API (RaaS) information in the Debug report for administrator accounts. If `valid_logins` is set to `True`, this information is also included in bug reports that are generated by non-admin users.
`raas_presence_expiration`	Seconds of inactivity before a minion is considered not present. Default is 3600 seconds (one hour).

Default RaaS configuration file

The following file shows the default RaaS configuration file, which includes an explanation of the different configuration settings:

```
# RaaS Default Configuration

# How often to run the compile_commands job that updates the activity tab
activity_tab_cycle: 2
```

```
# Elastic APM settings
apm_elastic:
  service_name:                         # Elastic APM Service Name
  secret_token:                         # Elastic APM Secret Token
  server_url:                           # Elastic APM Server URL
  environment: production

# audit tracking settings
audit:
  enabled: false
  valid_logins: false
  auth: true
  rpc: true
  system: true
  tasks: false
  rpc_max_payload: 100

# authentication backends
authers:
  ldap:
    log_detail: ERROR
    ssl: {}
    ldap_receive_timeout: 60
    group_level_limit: 20

# Configuration settings for background workers. Settings for each queue:
#   concurrency: number of worker processes (0 = auto calc one per core up to max)
#   max_tasks: worker recycles after running this many tasks
#   max_memory: in kB, 0=auto, None=unlimited
#   result_expires: how long results are stored in Redis, in seconds
#   prefetch_multiplier: How many messages to prefetch at a time multiplied by the number of
concurrent processes
#   without_heartbeat: When true, don't send event heartbeats. Reduces Redis usage.
#   without_mingle: When true, Don't synchronize with other workers at start-up. Reduces
worker startup up time.
#   without_gossip: When true, Don't subscribe to other workers events. Reduces redis usage.
#   use_fair_scheduler: Use Celery's fair scheduling algorithm, better for long running tasks
background_workers:
  combined_process: true                # Launch celery workers and RaaS processes together.
Set to False if running celery separately.
  broker: redis
  backend: redis
  log_level: warning
  celery:
    concurrency: 0
    max_tasks: 100000
    max_memory: 0
    result_expires: 60
    prefetch_multiplier: 1
    without_heartbeat: false
    without_mingle: true
    without_gossip: true
    use_fair_scheduler: true
  lr:
    concurrency: 0
    max_tasks: 100000
    max_memory: 0
    result_expires: 60
    prefetch_multiplier: 1
    without_heartbeat: false
    without_mingle: true
    without_gossip: true
    use_fair_scheduler: true
  grainscache:
    concurrency: 0
    max_tasks: 100000
    max_memory: 0
    result_expires: 60
    prefetch_multiplier: 1
    without_heartbeat: false
    without_mingle: true
    without_gossip: true
    use_fair_scheduler: true

# how often to run cache jobs (in seconds)
cache_cycle: 30

# path to RaaS cache directory
cachedir: /var/lib/raas/cache

# how often to run clean up jobs (in seconds)
clean_up_cycle: 900

# path to config directory (can be passed multiple times, order is respected)
config_dir:
- /etc/raas

# read files in config_dir subdirs recursively
config_recurse: false

# HTTP Cookie settings
cookie:
  name: raas-session
  expires: 43200

# for use with the webpack dev server only, Add the Access-Control-Allow-Origin: * header
cors_header_for_webpack: false

# Your customer ID
customer_id: 43cab1f4-de60-4ab1-85b5-1d883c5c5d09

# directory to serve files from
directory_root: /srv/raas

# enable (true) or disable (false) grains indexing.
enable_grains_indexing: true

# enable (true) or disable (false) cmd details in get_cmds API call. Should be disabled when
return counts are large.
```

```
enable_cmd_details: true

# Limit returns passed to UI so they don't crash the browser. 0 is unlimited. Recommended to
set as a mutliple of 50.
cmd_returns_max: 0

# path to extension module directory
extension_modules: /var/lib/raas/cache/ext_mods

# Use FIPS-compliant encryption
fips_mode: false

# Limit masterfs returns passed to UI so they don't crash the browser. 0 is unlimited.
Recommended to set as a mutliple of 50.
fs_returns_max: 0

# the address to bind to
interface: 0.0.0.0

# time to check unresponsive jobs (in minutes)
job_unresponsive_check: 5

# time to stop checking unresponsive jobs (in minutes)
job_unresponsive_check_stop: 2880

# JSON Web Token settings
jwt:
  expires: 3600                         # token expiration in seconds
  login_expires: 60                     # external authentication, login token expiration in
seconds
  algorithm: HS256
  max_logins: 100

# How long to keep historical data in days (leave unset to keep forever)
keep_history:
  audit:                                # How long to keep audit log (if audit is enabled)
  events: 1                             # How long to keep salt events
  jobs:                                 # How long to keep job data (commands, jids, returns)
  schedule:                             # How long to keep past schedule data

# Content and style of banner to show on UI login screen. A YAML block scalar
# can help with long message content:
#   message: >
#     Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
#     tempor incididunt ut labore et dolore magna aliqua.
login_banner:
  enabled: false
  style: info                           # info or warning
  message: ''

# date and time format for console logs
log_datefmt: '%H:%M:%S'

# date and time format for logfile logs
log_datefmt_logfile: '%Y-%m-%d %H:%M:%S'
```

```
# path to log file
log_file: /var/log/raas/raas

# loglevel for logfile logs, options: all, garbage, trace, debug, profile, info, warning,
error, critical, quiet
log_file_loglevel: error

# log format for console logs
log_fmt_console: '[%(levelname)-8s] %(message)s'

# log format for logfile logs
log_fmt_logfile: '%(asctime)s,%(msecs)03.0f [%(name)-17s][%(levelname)-8s:%(lineno)-4d][%
(processName)s:%(process)d]
  %(message)s'

# loglevels for specific python modules
log_granular_levels: {}

# options: all, garbage, trace, debug, profile, info, warning, error, critical, quiet
log_level: error

# master RSA private key size
master_key_size: 2048

# expiration timeout for pending master keys in seconds
master_pending_key_expiration: 7200

# master/minion will be marked as unknown if they haven't reported back within X seconds.
raas_presence_expiration: 3600

# max number of unresponsive master checks
master_unresponsive_check_limit: 2

# template used to generate master users
master_username_template: master_{}

# Automatically accept masters, use only for development.
master_autoaccept: false

# System metrics settings
metrics:
  enabled: true                         # If True, enable the collection of system metrics
  prometheus: false                     # If True, enable the Prometheus endpoint at /metrics
(also set prometheus_username and prometheus_password)
  prometheus_username:                  # Static username for retrieving /metrics
  prometheus_password:                  # Static password for retrieving /metrics
  snapshot_interval: 60                 # How often to record snapshot metrics, in seconds
  max_query_timedelta: 86400            # Maximum timedelta for a single call to
get_system_metrics, in seconds
  keep: 30                              # How long to retain metrics data, in days

# ignore some minion grains, glob matching allowed
minion_grains_filter:
  mode: blacklist
  grains: []

# 0=off, max seconds to lock when adding minion keys and cache. This throttles insert of
minions into the database.
minion_onboarding_throttle: 0

# max number of auto calculated processes per type. example: 8 max web, 8 max background
workers
max_processes: 8

# Minion deployment settings
minion_deployment:
  max_minion_deployment_time: 3600      # Maximum time (in seconds) allowed for minion
deployment after which status will be marked as failed
  airgap_install: false                 # Deploy minions in an airgapped environment

newrelic_config_file: /etc/raas/newrelic.ini

newrelic_enabled: false

# number of web server processes (0 = auto calc one per core up to max)
num_processes: 0

# number of password attempts to start blocking
password_attempts: 50

# number of seconds to sleep following a failed attempt
password_sleep: 30

# path to RaaS process ID file
pidfile: /var/lib/raas/run/raas.pid

# path to directory for RaaS PKI keys
pki_dir: /etc/raas/pki

# port to bind to
port: 8080

# delta proxy monitoring options
proxy:
  monitored: false
  monitor_interval: 90
  rebalance_interval: 120
  tgt: deltaproxy*
  tgt_type: glob

# vRA Integration
vra:
  validate_ssl: true                    # If True, raas proxy will validate ssl certs
  exclude_host: false                   # If True, raas proxy will not pass the host header
to CSP
  saved_params_timeout: 90              # How many seconds elapse before we get that latest
vra params from the db

# To use the the environment variable REDIS_URL, set `url: ENV`.
```

```
redis:
  url: redis://localhost:6379           # Redis URL without '/{database_number}' at the end
  broker_db: '0'                        # queue database number
  result_db: '1'                        # result storage database number
  cache_db: '2'                         # cache database number
  ssl: {}

# multiplier used to calculate retry timing on connection failures
retry_timeout_multiplier: 3

root_dir: /

# how often to check for scheduled jobs (in seconds)
schedule_cycle: 10

# how many future schedules are calculated per cycle
scheduler_max_futures_per_cycle: 500

# how many weeks ahead schedules are calculated out to
scheduler_max_futures_weeks_ahead: 12

# SecOps settings
sec:
  stats_snapshot_interval: 3600         # Interval in seconds between when stats for Secops
will be gathered (ENV Var: SSE_SEC_STATS_SNAPSHOT_INTERVAL)
  username: secops                      # Username used to log in to enterprise.saltstack.com
to get content (ENV Var: SSE_SEC_USERNAME)
  content_url: https://enterprise.saltstack.com/secops_downloads # URL from which SaltStack
Secops content will be downloaded. (ENV Var: SSE_SEC_CONTENT_URL)
  ingest_saltstack_override: true       # If True, existing SaltStack content will be updated
otherwise the change will be rejected. (ENV Var: SSE_SEC_INGEST_SALTSTACK_OVERRIDE)
  ingest_custom_override: true          # If True, existing Custom content will be updated
otherwise the change will be rejected. (ENV Var: SSE_SEC_INGEST_CUSTOM_OVERRIDE)
  locke_dir: locke                      # Location where SaltStack content in expanded before
ingestion. If the path is relative (no leading slash), then it is relative to the RAAS cache
dir (ENV Var: SSE_SEC_LOCKE_DIR)
  post_ingest_cleanup: true             # If True, post ingestion the contents of the
locke_dir will be cleaned out. (ENV Var: SSE_SEC_POST_INGEST_CLEANUP)
  download_enabled: true                # If True, SaltStack content downloading is enabled.
(should be False for air gapped systems) (ENV Var: SSE_SEC_DOWNLOAD_ENABLED)
  download_frequency: 86400             # The frequency in seconds of automated SaltStack
Secops content downloads and ingestion. (ENV Var: SSE_SEC_DOWNLOAD_FREQUENCY)
  compile_stats_interval: 10            # Interval in seconds between times that the compile
stats will be gathered. (ENV Var: SSE_SEC_COMPILE_STATS_INTERVAL)
  archive_interval: 300                 # The interval in seconds between attempts to archive
old assessment/remediation results (ENV Var: SSE_SEC_ARCHIVE_INTERVAL)
  old_policy_file_lifespan: 2           # The lifespan of old lock policy files in days that
will remain in the RAAS file system
  delete_old_policy_files_interval: 86400 # The interval in seconds between times that theold
lock policy files in the RAAS file system will be deleted
  ingest_on_boot: true                  # If True, SaltStack Secops content will be
downloaded and ingested soon after RAAS boot (ENV Var: SSE_SEC_INGEST_ON_BOOT)
  content_lock_timeout: 60              # When multiple RAAS heads are deployed, the
SaltStack SecOps content download and ingestion is serialized so only one RAAS head
at a time will attempt it.  This is the value for the redis lock timeout. (ENV Var:
```

```
SSE_SEC_CONTENT_LOCK_TIMEOUT)
  content_lock_block_timeout: 120       # This is the maximum time a RAAS head will
block on a lock to perform a SaltStack SecOps download and ingestion. (ENV Var:
SSE_SEC_CONTENT_LOCK_BLOCK_TIMEOUT)

# Sentry DSN to report errors (sensitive data is obfuscated)
sentry_dsn:

# path to RaaS directory for socket files
sock_dir: /var/lib/raas/sock

# for development only, always serve the session cookie regardless of the request being http
or https
spa_serve_cookie_always: false

# REQUIRED: fill in your database info
# - SQLAlchemy options - http://docs.sqlalchemy.org/en/rel_1_0/dialects/index.html
# - To use the the environment variable DATABASE_URL, set `url: ENV`. For example:
#   $ export DATABASE_URL=postgres://user:secret@localhost:5432/raas_db_name
# - To store database credentials in an encrypted file, run "raas save_creds"
#   after installation.
# - It is possible, but not recommended practice, to specify database credentials
#   in plaintext in this section as `username: user` and `password: secret`.
# - Make sure you specify the correct SSL parameters by setting `ssl: False`
#   or `True` and filling in the correct fields in `ssl_opts` OR
#   adding the right query parameters in the DATABASE_URL.
# - NOTE DATABASE_URL takes precedence over all other settings except username and password
sql:
  dialect: postgresql
  driver: psycopg2
  host:
  port:
  pool_size: 10
  pool_timeout: 10
  pool_recycle: 3600
  chunksize_yield_per_small_table: 1000
  chunksize_yield_per_big_table: 5000
  ssl: false
  ssl_opts: {}

# strict transport security header enabled (aka HSTS, HTTPS only)
strict_transport_security_header_enabled: true

# Do not calculate target group membership locally, have masters send it.
target_groups_from_master_only: false

# cross-site request forgery cookie enabled
tornado_xsrf_cookies_enabled: true

# check the running environment prior to starting services
verify_env: true

# Vulnerability Management settings
vman:
  vman_dir: vman                        # Location where SaltStack content in expanded before
```

```
ingestion. If the path is relative (no leading slash), then it is relative to the RAAS cache
dir (ENV Var: SSE_VMAN_DIR)
  download_enabled: true                # If True, SaltStack content downloading is enabled.
(should be False for air gapped systems) (ENV Var: SSE_VMAN_DOWNLOAD_ENABLED)
  download_frequency: 86400             # The frequency in seconds of automated
SaltStack Vulnerability Management content downloads and ingestion. (ENV Var:
SSE_VMAN_DOWNLOAD_FREQUENCY)
  username: vman                        # Username used to log in to enterprise.saltstack.com
to get content (ENV Var: SSE_VMAN_USERNAME)
  content_url: https://enterprise.saltstack.com/vman_downloads # URL from which SaltStack
Vulnerability Management content will be downloaded. (ENV Var: SSE_VMAN_CONTENT_URL)
  ingest_on_boot: true                  # If True, SaltStack Vulnerability Management content
will be downloaded and ingested soon after RAAS boot (ENV Var: SSE_VMAN_INGEST_ON_BOOT)
  post_ingest_cleanup: false            # If True, post ingestion the contents of the
vman_dir will be cleaned out. (ENV Var: SSE_VMAN_POST_INGEST_CLEANUP)
  content_lock_timeout: 2000            # When multiple RAAS heads are deployed, the
SaltStack vulnerability management content download and ingestion is serialized so only one
RAAS head at a time will attempt it.   (ENV Var: SSE_VMAN_CONTENT_LOCK_TIMEOUT)
  compile_stats_interval: 60            # Interval in seconds between times that the compile
stats will be gathered. (ENV Var: SSE_VMAN_COMPILE_STATS_INTERVAL)
  stats_snapshot_interval: 3600         # Interval in seconds between when stats for VMan
will be gathered (ENV Var: SSE_VMAN_STATS_SNAPSHOT_INTERVAL)
  old_policy_file_lifespan: 2           # The lifespan of old policy files in days that will
remain in the RAAS file system
  delete_old_policy_files_interval: 86400 # The interval in seconds between times that theold
vman policy files in the RAAS file system will be deleted
  tenable_asset_import_enabled: true    # If True, minion grains in SSE will be sent to
tenablefor matching assets
  tenable_asset_import_grains:          # Choose the minion grains that needs to be sent
to tenable.Grain fqdn and ipv4 will be sent even if not included here.For additional
information, please refer https://developer.tenable.com/reference#assets-import
  - 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

# how long to wait while reading body (seconds), when None uses tornado default
webserver_body_timeout:

# maximum amount of data for body, when None uses tornado default
webserver_max_body_size:

# maximum amount of incoming data to buffer, when None uses tornado default
webserver_max_buffer_size:

# in kB, 0=auto
webserver_max_memory: 0

# in seconds, 0=disabled
webserver_max_time: 0

# max interval in seconds subscription updates can be sent
websocket_debounce: 5

# time in seconds to send ping over websocket to keep it open
websocket_ping_interval: 15

# timeout in seconds to wait for websocket ping
websocket_ping_timeout: 600

# in seconds, polling time for non-database listening subscriptions
websocket_polling: 15

# time in seconds for a websocket ticket to expire
websocket_ticket_expiration: 5
```

What to do next

After configuring the RaaS node, 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.

Log in for the first time and change default credentials

After completing the previous post-installation steps, you can log in to the SaltStack Config for the first time. After you first login, you should change the root password and secure your RaaS credentials.

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.

Before you start

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
- Check the RaaS configuration file

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.

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`

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.

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 configuration.

What to do next

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

Accept the Salt master key and back up data

After you've successfully logged in for the first time, you need to complete some important tasks in the SaltStack Config user interface. You need to accept the Salt master's key, remove the pillar top file, and back up critical data. You can also try some sample content to enable more accurate minion presence detection and to test the overall system's functionality.

Before you start

Accepting the Salt 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
- Check the RaaS configuration file
- Log in for the first time and change default credentials

Accept the Salt master's key

During the Salt 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.

Remove the pillar top file

If you installed SaltStack Config using the Standard 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 Standard 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.

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.

Try some sample content (optional)

To test the basic functionality of SaltStack Config, try working with some sample content in the user interface. You may need to install the Salt minion service on a few of the nodes you want to manage before trying the 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.

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 Minions 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 **Minions** to access the Minions 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 Sample content in the Using and Managing SaltStack Configguide.

What to do next

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.

Set up SSL certificates

As part of the post-installation process, you might want to set up your Secure Sockets Layer (SSL) certificates. Setting up SSL certificates is optional when installing SaltStack Config, but recommended.

Before you start

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
- Check the RaaS configuration file
- Log in for the first time and change default credentials
- Accept the Salt master key and back up data

Set up and configure SSL certificates

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 (pre-installation).
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.

Updating SSL certificates

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

Troubleshooting SaltStack Config environments with vRealize Automation that use self-signed certificates

This workaround is for customers who are working with vRealize Automation deployments that use a certificate signed by a non-standard certificate authority.

SaltStack Config might experience the following symptoms:

- When you first open vRealize Automation, your web browser displays a security warning next to the URL or in the display page that the certificate cannot be validated.
- When you attempt to open the SaltStack Config user interface in your web browser, you might get a blank screen.

These symptoms might be caused if your vRealize Automation deployment is using a certificate signed by a non-standard certificate authority. To verify whether this is causing SaltStack Config to display a blank screen, SSH into the node that is hosting SaltStack Config and review the RaaS log file (`/var/log/raas/raas`). If you find a traceback error message that indicates that self-signed certificates are not allowed, the following workaround can possibly solve this issue.

Note As a security best practice, you should never set up a production environment to use self-signed certificates or improperly signed certificates to authenticate vRealize Automation or SaltStack Config. The recommended practice is to use certificates from trusted certificate authorities instead.

If you choose to use self-signed or improperly signed certificates, you may put your system at serious risk of a security breach. Proceed with caution when using this procedure.

If you experience this problem and your environment needs to continue using a certificate signed by a non-standard certificate authority, the solution is to add the vRealize Automation certificate CA to your SaltStack Config environment, as explained in the following workaround.

This workaround requires:

- Root access
- The ability to SSH into the RaaS server

Note As an additional security best practices, only the most trusted and senior individuals at your organization should be granted this level of access. Take care to restrict root access to your environment.

You may find it easier to create a private certificate authority and sign your own vRealize Automation certificates with that certificate authority rather than using self-signed certificates. The advantage of this approach is you only have to go through this process once for every vRealize Automation certificate you need. Otherwise, you would have to go through this process for every vRealize Automation certificate you create. For more information about creating a private certificate authority, see How do you sign a certificate request with your own certification authority (Stack Overflow).

To add a certificate signed by a non-standard certificate authority to the list of certificate authorities in SaltStack Config:

1 Attempt to open the vRealize Automation web interface in your browser. The certificate should display a warning message in the browser window and the URL display.

2 Download the required certificate.

   - **For Chrome browsers:** Click the Not Secure warning in the URL display to open a menu. Select Certificate (invalid). Drag the missing certificate to your local computer's file explorer or finder to save it. Choose the certificate signer (CA) if it is available. Click the certificate icon and then drag it to your local computer's file explorer. If the file extension is not .pem (.crt .cer .der), use the following command to convert it to .pem format: `openssl x509 -inform der -in certificate.cer -out certificate.pem`
   - **For Firefox browsers:** Click the warning icon in the URL display to open a menu. Select Connection not secure > More information. In the dialog box, click View certificate. Click the missing certificate to download it to your local computer's file system.

3 If you haven't already, SSH into the RaaS server.

4 Append the certificate file to the end of the file in this directory: `/etc/pki/tls/certs/ca-bundle.crt`. You can append the certificate to the end of file using the following command, replacing the example file with your actual filename:

   ```
   cat <certificate-file>.crt  >> /etc/pki/tls/certs/ca-bundle.crt
   ```

   Note You can also copy files with the `.pem` extension using this command.

   ---

5 Navigate to the directory `/usr/lib/systemd/system` and open the file `raas.service` in your editor. Add the following line to this file anywhere above the ExecStart line:

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

6 Reload the daemon and restart RaaS using these commands:

```
systemctl daemon-reload
systemctl stop raas
rm /var/log/raas/raas
systemctl start raas
tail -f /var/log/raas/raas
```

Note Use `tail -f /var/log/raas/raas` to show the RaaS log file in continuous display, which may help with troubleshooting.

7 Verify that this solution has solved the problem by logging into the SaltStack Config web interface. If the problem has been resolved, SaltStack Config displays the Dashboard page.

What to do next

After setting up SSL certificates, you may need to complete additional post-installation steps.

If you are a SaltStack SecOps customer, the next step is to set up these services. For more information, see Configure SaltStack SecOps.

If you've completed all the necessary post-installation steps, the next step is to integrate SaltStack Config with vRealize Automation SaltStack SecOps. See Create a SaltStack Config integration with vRealize Automation for more information.

Set up Single Sign-On (SSO)

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.

Before you start

Configuring 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
- Check the RaaS configuration file
- Log in for the first time and change default credentials
- Accept the Salt master key and back up data
- Set up SSL certificates

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 Using and Managing SaltStack Config guide.

What to do next

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.

If you've completed all the necessary post-installation steps, the next step is to integrate SaltStack Config with vRealize Automation SaltStack SecOps. See Create a SaltStack Config integration with vRealize Automation for more information.

Configure SaltStack SecOps

This post-installation step is optional, but is required for customers who have purchased a vRealize Automation SaltStack SecOps license. SaltStack SecOps is add-on for SaltStack Config that provides two security libraries. Both content libraries update regularly as security standards change. If you are a SaltStack SecOps customer, you need to configure these security libraries to download (or ingest) new content automatically as security standards change.

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

For more instructions about configuring SaltStack SecOps library downloads, see Configuring SaltStack SecOps.

What to do next

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

If you've completed all the necessary post-installation steps, the next step is to integrate SaltStack Config with vRealize Automation SaltStack SecOps. See Create a SaltStack Config integration with vRealize Automation for more information.

Create a SaltStack Config integration with vRealize Automation

7

You can configure a SaltStack Config integration to access the SaltStack Config service and use SaltStack Config objects and actions in vRealize Automation.

For information about creating and configuring a SaltStack Config integration for vRealize Automation Cloud Assembly, see Configure a SaltStack Config integration in vRealize Automation.

Install Salt on your infrastructure 8

After you have installed and integrated the SaltStack Config service, you also need to install, run, and register the Salt minion service on any nodes that you intend to manage using SaltStack Config. You can deploy the Salt minion service to your nodes using either vRealize Automation cloud templates or by installing the service through Secure Shell (SSH).

Salt and its relationship to SaltStack Config

SaltStack Config runs on 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.

Salt uses the controller-client model in which a controller issues commands to a client and the client executes the command. In the Salt ecosystem, the controller 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.

Minions are nodes that run the salt-minion service. The service listens to commands from a Salt master and performs the requested tasks. You can deploy minions from vRealize Automation cloud templates.

Before you can begin using SaltStack Config for configuration management, you must first install the Salt minion service on all nodes that you want to manage. You must also register the minions by sending and accepting their keys to SaltStack Config .

Before you start

- Install and configure the SaltStack Config and integrate it with vRealize Automation.
- Nodes that are managed by SaltStack Config must be able to reach the Salt master and must reside on the same network as the SaltStack Config integration point and the Salt master.
- vSphere machines that are deployed to a private network must be able to initiate a connection with SaltStack Config integration and the Salt master.

Installing the Salt minion service through SSH

The process for installing the Salt minion service using SSH depends on the operating system running on those nodes.

- For information about installing the Salt minion service on RedHat Linux or CentOS, see Install Salt (pre-installation).
- For other operating systems, see http://repo.saltstack.com/.

After installing the Salt minion service:

1 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
```

2 Start the minion service:

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

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

3 Repeat the previous steps for all remaining nodes.

After configuring these minion files to point to the Salt master, accept the minion keys in the SaltStack Config service in the Minion Keys workspace.

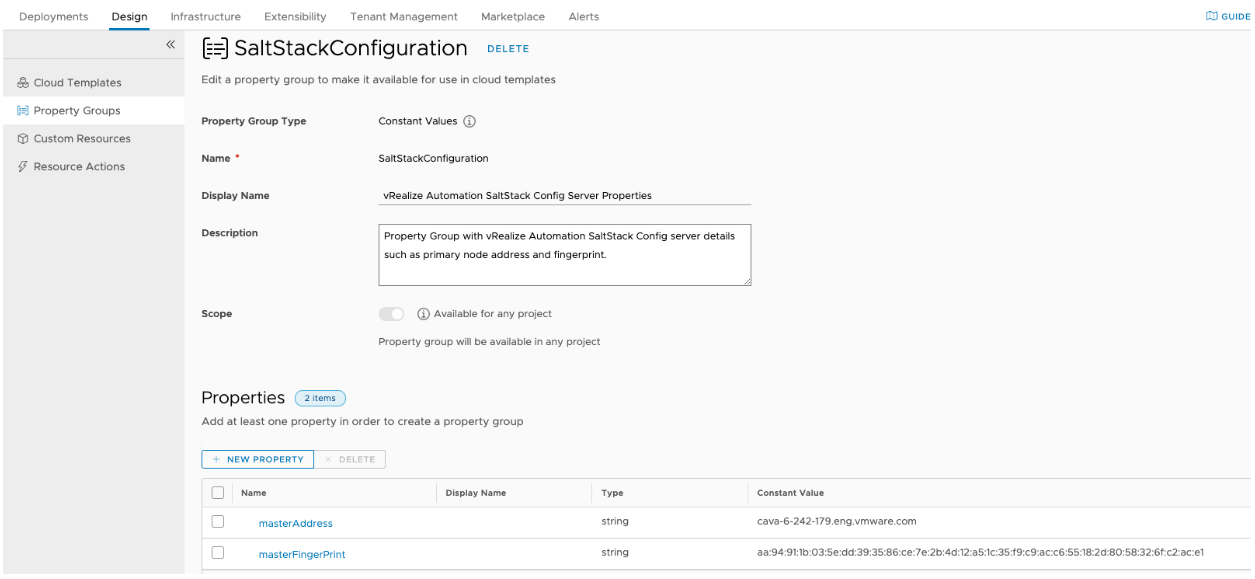
Installing the Salt minion service using vRealize Automation cloud templates

To deploy the Salt minion service using cloud templates, you must have access to, and be proficient at using, cloud-init (Linux) or Cloudbase-init (Windows). To add Salt minions to the Salt master that is configured for the vRealize Automation SaltStack Config integration, the virtual machine in your cloud template must support cloud-init (Linux) or Cloudbase-init (Windows).

The following sections explain how to deploy the Salt minion service using cloud templates.

SaltStackConfiguration property group

The process of installing and configuring the SaltStack Config service in vRealize Suite Lifecycle Manager creates a vRealize Automation property group named `SaltStackConfiguration`. You use the `SaltStackConfiguration` property group values when configuring the SaltStack Config integration in a vRealize Automation cloud template or deployment. You also use them in cloud-init or Cloudbase-init-based configuration in a vRealize Automation template (previously called blueprints) to install minions. The two properties in the `SaltStackConfiguration` are `masterAddress`, which matches the **hostname** setting on the SaltStack Config integration point, and `masterFingerprint`. A sample `SaltStackConfiguration` property group is shown below.



SaltStackConfiguration

Edit a property group to make it available for use in cloud templates

Property Group Type	Constant Values
Name *	SaltStackConfiguration
Display Name	vRealize Automation SaltStack Config Server Properties
Description	Property Group with vRealize Automation SaltStack Config server details such as primary node address and fingerprint.
Scope	Available for any project Property group will be available in any project

Properties 2 items

Add at least one property in order to create a property group

Name	Display Name	Type	Constant Value
masterAddress		string	cava-6-242-179.eng.vmware.com
masterFingerPrint		string	aa:94:91:1b:03:5e:dd:39:35:86:ce:7e:2b:4d:12:a5:1c:35:f9:c9:ac:c6:55:18:2d:80:58:32:6f:c2:ac:e1

Add minions to the Salt master configured for vRealize Automation

When SaltStack Config is installed, a Salt master IP address is specified. That master IP address is used as the `masterAddress` property when you deploy minions from a vRealize Automation cloud template.

You install and deploy minions by using cloud-init or Cloudbase-init scripting in a vRealize Automation cloud template or deployment. You can also use an image mapping that represents a cloud configuration script that uses either of those formats. To add Salt minions to the Salt master that is associated to a vRealize Automation SaltStack Config integration, the target machine must support cloud-init (Linux) or Cloudbase-init (Windows). vRealize Automation cloud configuration scripting supports both formats.

You configure a machine resource in the cloud template with a `minionId` value and `cloudConfig` value and refer to the property group `SaltStackConfiguration`. The `SaltStackConfiguration` property group is created during SaltStack Config service installation and configuration in vRealize Suite Lifecycle Manager. It contains the `masterAddress` and `masterFingerprint` properties.

The `minionId` value must match the value specified for the machine's `/salt/minion_id` in the `cloudConfig` section of the cloud template code.

Examples of Windows-based and Linux-based vRealize Automation cloud template code are shown below. Note that the cloud configuration scripting can be specified using any of the following methods:

- vRealize Automation image that is called from the cloud template code
- Cloud configuration script that is called from the cloud template code
- Cloud configuration script content that is added directly to the cloud template code

Note: When you deploy a cloud template that contains Salt minions, if the deployment is not visible in vRealize Automation Cloud Assembly, you can display the deployment by using the vRealize Automation Service Broker service.

Example - Linux-based deployment and cloud-init

A sample cloud template configuration for deploying minions for a Linux-based machine that supports cloud-init is shown below:

```
resources:
  Salt-Minion:
    type: Cloud.Machine
    properties:
      image: Ubuntu-18
      flavor: medium
      constraints:
        - tag: 'env:vsphere'
      cloudConfig: |
        #cloud-config
        hostname: ${input.saltminionhostname}
        users:
          - name: ${input.user}
            sudo: ['ALL=(ALL) NOPASSWD:ALL']
            groups: sudo
            shell: /bin/bash
        runcmd:
          - PASS=${input.password}
          - USER=${input.user}
          - echo $USER:$PASS | /usr/sbin/chpasswd
          - sed -i "s/PasswordAuthentication no/PasswordAuthentication yes/g" /etc/ssh/
```

```
sshd_config
          - service ssh reload
          - curl -L https://bootstrap.saltstack.com -o install_salt.sh
          - sudo sh install_salt.sh -A ${propgroup.SaltStackConfiguration.masterAddress}
```

Example - Windows-based deployment and Cloudbase-init

A sample cloud template configuration for deploying minions for a Windows-based machine that supports Cloudbase-init is shown below:

```
formatVersion: 1
inputs: {}
resources:
  WindowsVM-Minion:
    type: Cloud.vSphere.Machine
    properties:
      image: win2016
      flavor: medium
      customizationSpec: Windows
      minionId: '${resource.WindowsVM-Minion.resourceName}'
      networks:
        - network: '${resource.wpnet.id}'
          name: '${wpnet.name}'
          assignPublicIpAddress: true
      cloudConfig: |
        #ps1_sysnative
        [Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12;
Invoke-WebRequest -OutFile C:\Salt-Minion-3002.2-Py3-AMD64-Setup.exe -Uri https://
repo.saltstack.com/windows/Salt-Minion-3002.2-Py3-AMD64-Setup.exe
        Start-Process -Wait -FilePath "C:\Salt-Minion-3002.2-Py3-AMD64-Setup.exe"
-ArgumentList "/S" -PassThru
        ((Get-Content -path C:\salt\conf\minion -Raw) -replace "#master: salt", "master: $
{propgroup.SaltStackConfiguration.masterAddress}") | Set-Content -Path C:\salt\conf\minion
        ((Get-Content -path C:\salt\conf\minion -Raw) -replace "#master_finger: ''",
"master_finger: '${propgroup.SaltStackConfiguration.masterFingerPrint}'") | Set-Content -Path
C:\salt\conf\minion
        Set-Content -Path C:\salt\conf\minion_id -Value '${resource.WindowsVM-
Minion.resourceName}'
        C:\salt\salt-call.bat service.restart salt-minion
  wpnet:
    type: Cloud.Network
    properties:
      name: wpnet
      networkType: existing
```

How to configure Powershell commands

You can configure PowerShell commands in the `cloudConfig` section by using the following steps. See the SaltStack Config product documentation for details.

1 Download the Salt minion package from the Salt repo web site.

2 Install the Salt minion service in the Windows VM.

3 Update the Salt master host name value in the minion configuration.

4 Update the Salt master fingerprint value in the minion configuration.

5 Set the `minion_id` to the VM resource name that will be used to accept the minion on the Salt master.

6 Restart the minion.

More information about deploying Salt minions

For alternative methods of using integrated SaltStack Config to deploy Salt minions from a vRealize Automation cloud template, see these VMware blog articles:

- *Cloud Assembly and Deploying Minions* in vRealize Automation 8.3 and SaltStack Config – Technical Overview
- Cloud Assembly and ABX Secrets (and how to use them for installing vRealize SaltStack Minion Agents)

For related information about defining and deploying vRealize Automation Cloud Assembly templates in general, especially if you are familiar with SaltStack but new to vRealize Automation, see Designing your vRealize Automation Cloud Assembly deployments.

Upgrade from a previous version 9

Upgrade SaltStack Config to the latest stable version from a previous version.

Note As part of VMware's initiative to remove problematic terminology, the term Salt master will be replaced with a better term in SaltStack Config and related products and documentation. This terminology update may take a few release cycles before it is fully complete.

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

Best practices when upgrading

Follow these guidelines while planning your upgrade:

- **Do not upgrade using the installer or manual installation instructions.**If you are upgrading your SaltStack Config installation, refer to the following upgrade instructions.
- **For best results, increment from one major release to the next.** As a best practice, always upgrade from the latest major version of SaltStack Config to the new release. If you are on an older release, upgrade incrementally from one release to the next.
- **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.

How to upgrade SaltStack Config

To upgrade SaltStack Config, complete the following tasks. These tasks are intended to be followed in a specific order as listed in this guide.

This chapter includes the following topics:

- Back up your data
- Upgrade databases
- Upgrade Salt
- Upgrade the RaaS node
- Upgrade the Master Plugin

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 Salt 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 Salt master paths could instead be in the `/etc/salt/master.d/raas.conf` file instead.

After backing up the SaltStack Config and Salt 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.

What to do next

After backing up your data, the next step in the upgrade process is to update your SaltStack Config databases. See Upgrade databases for more information.

Upgrade databases

Upgrade the PostgreSQL and Redis databases before you upgrade the other SaltStack Config architectural components.

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.

Upgrade Redis database

SaltStack Config requires a Redis 5.x database, but Redis 5.0.4 is recommended. The recommended version of Redis is included with the SaltStack Config installer.

Upgrading to the latest version of Redis 5.x is not required. However, upgrading Redis can possibly improve performance. For instructions on upgrading Redis, see Redis Administration.

What to do next

After upgrading your databases, the next step is to upgrade your Salt components to the latest version of Salt. See Upgrade Salt for more information.

Upgrade Salt

For best performance, ensure the Salt components in your infrastructure are running on the latest major version of Salt. For example, check that your minions are on the latest version of Salt.

For instructions on upgrading your Salt masters and other Salt components, see Upgrading Your Salt Infrastructure.

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

What to do next

After upgrading Salt in your infrastructure, the next step is to upgrade the RaaS node. See Upgrade the RaaS node for more information.

Upgrade the RaaS node

After upgrading the latest version of PostgreSQL, Redis, and Salt, you then need to upgrade the RaaS node from the previous version to the latest version.

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.

Important 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 Download the upgrade files from Customer Connect.

2 Save any changes you made to the default file system, pillar data, and jobs as new files or jobs.

3 Note any pillar assignments that are made to the default targets. These need to be re-assigned after upgrade.

4 Stop the RaaS service using the following command:

```
sudo systemctl stop raas
```

5 Remove the log file(s) in the `/var/log/raas` directory. Clearing the log files provides a clean log file if troubleshooting is necessary.

6 Remove the currently installed version of the API (RaaS) with the following command:

```
sudo yum remove raas
```

7 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
```

8 **IMPORTANT:** Restore the backup of the following files:

- `/etc/raas/raas`
- `/etc/raas/raas.secconf`
- `/etc/raas/pki/`

9 Update permissions for the `raas` user with the following command:

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

10 OPTIONAL: If you have a SaltStack SecOps license and want to add the compliance library, 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 SecOps 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 OPTIONAL: If you have a SaltStack SecOps license and want to add the vulnerability library, 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',
                                '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 SecOps 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.

12 The RaaS currently has a known issue related to stale jobs. 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`, Chapter 11 Contact Support for a workaround.

13 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.

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

   ```
   exit
   ```

15 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.

What to do next

After upgrading the RaaS node, the final task is to upgrade the Master Plugin. See Upgrade the Master Plugin for more information.

Upgrade the Master Plugin

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

Note Before you upgrade the Salt master(s), ensure that the pip3 application is installed on the Salt 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 Salt 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 Salt master. If it is running Python 3.6 or higher, no changes are needed. Otherwise, delete the prior version of the SSEAPE module. (The SSEAPE is the SaltStack Config plugin for the Salt 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

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

Ubuntu

```
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 Salt 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 Check that the `master_job_cache` and `event_return` entries are set to `sseapi`. The `pgjsonb` returner is no longer available.

7 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 Chapter 10 Troubleshootingpage or Chapter 11 Contact Support.

Troubleshooting 10

Read some of the common errors that users experience during the SaltStack Config installation process and how to fix them.

Problem	Cause	Solution
The SaltStack Config user interface shows a blank screen. When you first try to open SaltStack Config within vRealize Automation, nothing happens.	These symptoms might be caused if your vRealize Automation deployment is using a certificate signed by a non-standard certificate authority.	See Troubleshooting SaltStack Config environments with vRealize Automation that use self-signed certificates for a workaround.
When you first open the SaltStack Config in your vRealize Automation deployment, your web browser displays a security warning next to the URL or in the display page that the certificate cannot be validated.	These symptoms might be caused if your vRealize Automation deployment is using a certificate signed by a non-standard certificate authority.	See Troubleshooting SaltStack Config environments with vRealize Automation that use self-signed certificates for a workaround.
You see yum install error messages. For example, when installing Salt as a pre-installation step (see Install or upgrade Salt (pre-installation)), `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.saltproject.io` 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.

Problem	**Cause**	**Solution**
The Salt 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 Salt master to accept minion keys.	Specify the IP address of the Salt master in each minion's `etc/salt/minion.d/id.conf` file. Edit this file and change the `master` setting to show the Salt master's IP address. For example, `master:127.0.0.0`. For additional methods of connecting the minion to the Salt master, see Configuring the Salt Minion.
You experience authentication errors when applying a highstate. During the initial application of the highstate to the first Salt master, you may see the following error message: `Authenticationerroroccurred`.	This error displays because the Salt master has not yet authenticated to the RaaS node.	The Master Plugin installation state will restart the Salt master service and the issue will be resolved automatically.

Contact Support

11

Support for SaltStack Config is available to SaltStack Config customers with an active license and maintenance agreement.

For more information about contacting VMware Support, refer to the vRA SaltStack Config support page.

Extending system architecture 12

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. Read these guides for more information.

This chapter includes the following topics:

Setting up multiple RaaS nodes

You can configure multiple RaaS nodes that share a single PostgreSQL database and Redis node. This method iis 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.

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.

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 vRealize Suite Lifecycle Manager 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 `postgresqlpg_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.

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 2117:21 /etc/raas/raas.secconf
# ls -l /etc/raas/pki/.raas.key
-rwx------. 1 raas raas 77 Jan 2117:17 /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.

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, Chapter 11 Contact Support.

Improve system performance

Read helpful information and get links to articles you can use as you try to optimize yoiur SaltStack Config system.

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 Config Performance Configurations
- Background Worker Options

Benchmarking guide

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

Tuning PostgreSQL

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

Generating default RaaS configuration files

SaltStack Config configuration files are used during initial setup to define fundamental settings to allow the API (RaaS) to communicate with the database and connected Salt masters.

You can customize your SaltStack Configdeployment during initial setup, or any time you want to improve performance, by modifying your RaaS or Salt master configuration files.

You can generate the default configuration files as needed. For example, you might find it useful to regenerate these files when upgrading SaltStack Config to take advantage of the latest features.

To generate the default RaaS configuration file, run this command on the RaaS server:

```
raas genconfig /path/to/default-raas-cfg.conf
```

Note The last argument of the above command indicates where to save the generated file. You should not use the file path `/etc/raas/raas`, as this will overwrite the current RaaS configuration file. Only use this file path if you have no need to preserve an existing current RaaS configuration file.

SaltStack Config configuration file is divided into the following sections:

- The API (RaaS)
- Network configuration
- Salt masters and the Salt Master Plugin

The API (RaaS) settings are in the `/etc/raas/raas` configuration file. The following is a subset of frequently-used configuration settings.

Required settings

Setting	Description
customer_id	Your customer ID, or sample UUID.
sql	`username`, `password`, `host`, and `port` can be configured to match your database configuration. For more on storing credentials securely, see Securing credentials in your SaltStack Config configuration.

Other important settings

Setting	Description
tls_crt	Path to the `crt` file for encrypted communication. If this certificate is self-signed and should not be validated using a known CA, be sure to set the `sseapi_validate_cert` option to `False` in the Salt master configuration file.
tls_key	Certificate key file.
port	Port that is used for connections from the SaltStack Config user interface and Salt masters.

Setting	Description
audit	Include the API (RaaS) information in the Debug report for administrator accounts. If `valid_logins` is set to `True`, this information is also included in bug reports that are generated by non-admin users.
raas_presence_expiration	Seconds of inactivity before a minion is considered not present. Default is 3600 seconds (one hour).

Network configuration

Communication with the API (RaaS) uses REST calls over HTTP(s) on standard web ports (80 or 443). Connections to RaaS are initiated by the SaltStack Config user interface or by the Salt master, so incoming ports do not need to be configured on those systems.

Reference 13

Read these guides or learn more about alternative methods (not recommended) for installing SaltStack Config .

This chapter includes the following topics:

- Manual installation
- Single-node installation

Manual installation

The manual installation method for installing SaltStack Config is an alternative to the standard installation method but 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.

The manual installation method supports installation on the following operating systems:

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

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.

Download and import key files

Begin the manual installation process by downloading the manual installation files from Customer Connect.

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
```

What to do next

The next step is to install and configure the PostgreSQL database. See PostgreSQL database installation and configuration for more information.

PostgreSQL database installation and configuration

As part of the manual installation process (which is not recommended), install and configure the PostgreSQL database.

To install and configure the PostgreSQL database:

1 Install PostgreSQL using the following commands:

   RHEL

   ```
   sudo yum install -y https://download.postgresql.org/pub/repos/yum/reporpms/EL-7-x86_64/
   pgdg-redhat-repo-latest.noarch.rpm
   sudo yum update
   sudo yum install -y 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'
```

What to do next

After installing and configuring PostgreSQL, the next step is to install and configure the Redis database. See Redis installation and configuration for more information.

Redis installation and configuration

As part of the manual installation process (which is not recommended), install and configure the Redis database.

To install and configure the Redis database:

1 Install Redis using the following commands:

   RHEL

   Install the `Redis` and `jemalloc` installation packages you extracted from the installer. 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.

```
bind0.0.0.0
requirepass {{ your_redis_password }}
```

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

What to do next

After installing and configuring Redis, the next step is to install and configure RaaS. See RaaS installation and configuration for more information.

RaaS installation and configuration

As part of the manual installation process (which is not recommended), install and configure RaaS.

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 **FOR SLES INSTALLATIONS ONLY:** 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:postgresqlhost:localhostport:5432driver:psycopg2ssl: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 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.

What to do next

After installing and configuring RaaS, the next step is to install the RaaS license key. See Install the RaaS license key for more information.

Install the RaaS license key

As part of the manual installation process (which is not recommended), install the RaaS 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 a sales representative to purchase a license and then install the key.

Which license do I need?

See Which license do you need? for more information about licensing.

Finding your license key

Download your license from My VMware (Customer Connect).

Installing the key

The following process explains the legacy method for installing the RaaS license key. SaltStack Config is backwards compatible and this method of installing the license key is still supported. See Install the license key for information about the standard instructions for installing the license key.

To install the license key on the RaaS node:

1 Copy the license file to your RaaS node, such as transferring it using the SCP or SFTP protocols.

2 Login as the root user on the RaaS node.

3 Unzip the license file. For example:

```
unzip raas.license.yourlicensefilename.zip
```

4 Run the following commands:

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

Note If you are not running as the root user and receive an unauthorized user alert, you can add the `sudo` command in front of each command as needed.

5 Restart the RaaS service:

```
systemctl restart raas
```

What to do next

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

- Install and configure the Master Plugin
- Check the RaaS configuration file
- Log in for the first time and change default credentials
- Accept the Salt master key and back up data
- Set up SSL certificates
- Configure SaltStack SecOps (optional)
- Set up Single Sign-On (SSO) (optional)

The first post-installation step is to install and configure the Master Plugin. To begin the next post-installation step, see Install and configure the Master Plugin.

Single-node installation

The single-node installation method is equivalent to the vRealize Suite Lifecycle Manager installation method. In this method, 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.

As it runs, the single-node installer for SaltStack Config:

- 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 Salt master.
- Applies the Salt states needed to install SaltStack Config.
- Installs the required versions of PostgreSQL, Redis, and Python Setuptools on the server.

Before you start

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 (pre-installation)

- Transfer and import files

Caution For a single-node installation, it is especially important to follow all the steps listed on the Install or upgrade Salt (pre-installation) page. The exception is if you are installing SaltStack Config in an air-gapped environment.

Run the 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 Config...
```

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 Salt master and minion.

Note If both the Salt master service and 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 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 Chapter 10 Troubleshooting page or Chapter 11 Contact Support.

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 (Salt masters, web-based interface users, remote systems calling the API (RaaS), etc).

What to do next

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
- Configure SaltStack SecOps (optional)
- Set up Single Sign-On (SSO) (optional)