saltstack

tutorialspoint

SIMPLY EASY LEARNING

About the Tutorial

SaltStack is an open-source configuration management and remote execution engine. It remotely executes commands across all machines. It is a python based software. **Thomas S Hatch** is the creator and the principal architect of SaltStack. SaltStack uses the ZeroMQ messaging library to process high-speed requirements for all networking layers. Salt is simple, scalable and fast.

This tutorial will explore the basic principles of SaltStack, SaltStack setup, Minion file system and then walk through with remote execution steps, configuration management, cloud management, Python API operations and finally conclude with a complete working example.

Audience

This is a tutorial for those professionals who are aspiring to make a career in SaltStack. This tutorial will give you enough understanding on the remote execution operation and configuration management.

Prerequisites

Before you start doing practice with the various components given in this tutorial, it is being assumed that you are already aware and have a sound knowledge of ZeroMQ and core knowledge of Python.

Copyright & Disclaimer

Table of Contents

1. SaltStack – Overview

In this chapter, we will learn the basics of SaltStack. SaltStack's remote execution capabilities allow administrators to run commands on various machines in parallel with a flexible targeting system. Salt configuration management establishes a master-minion model to quickly, very easily, flexibly and securely bringing infrastructure components in line with a given policy.

What is SaltStack?

Salt is a very powerful automation framework. Salt architecture is based on the idea of executing commands remotely. All networking is designed around some aspect of remote execution. This could be as simple as asking a **Remote Web Server** to display a static Web page, or as complex as using a shell session to interactively issue commands against a remote server. Salt is an example of one of the more complex types of remote execution.

Salt is designed to allow users to explicitly target and issue commands to multiple machines directly. Salt is based around the idea of a Master, which controls one or more **Minions**. Commands are normally issued from the Master to a target group of Minions, which then execute the tasks specified in the commands and then return the resulting data back to the Master. Communications between a master and minions occur over the **ZeroMQ message bus**.

SaltStack modules communicate with the supported minion operating systems. The **Salt Master** runs on Linux by default, but any operating system can be a minion, and currently Windows, VMware vSphere and BSD Unix variants are well supported. The Salt Master and the minions use keys to communicate. When a minion connects to a master for the first time, it automatically stores keys on the master. SaltStack also offers **Salt SSH**, which provides an "agent less" systems management.

Need for SaltStack

SaltStack is built for speed and scale. This is why it is used to manage large infrastructures with tens of thousands of servers at LinkedIn, WikiMedia and Google.

Imagine that you have multiple servers and want to do things to those servers. You would need to login to each one and do those things one at a time on each one and then you might want to do complicated things like installing software and then configuring that software based on some specific criteria.

Let us assume you have ten or maybe even 100 servers. Imagine logging in one at a time to each server individually, issuing the same commands on those 100 machines and then editing the configuration files on all 100 machines becomes very tedious task. To overcome those issues, you would love to update all your servers at once, just by typing one single command. SaltStack provides you exactly the solution for all such problems.

Features of SaltStack

SaltStack is an open-source configuration management software and remote execution engine. Salt is a command-line tool. While written in Python, SaltStack configuration management is language agnostic and simple. Salt platform uses the push model for executing commands via the SSH protocol. The default configuration system is **YAML** and **Jinja templates**. Salt is primarily competing with **Puppet**, **Chef** and **Ansible**.

Salt provides many features when compared to other competing tools. Some of these important features are listed below.

- **Fault tolerance –** Salt minions can connect to multiple masters at one time by configuring the master configuration parameter as a YAML list of all the available masters. Any master can direct commands to the Salt infrastructure.
- **Flexible –**The entire management approach of Salt is very flexible. It can be implemented to follow the most popular systems management models such as Agent and Server, Agent-only, Server-only or all of the above in the same environment.
- **Scalable Configuration Management –** SaltStack is designed to handle ten thousand minions per master.
- **Parallel Execution model –** Salt can enable commands to execute remote systems in a parallel manner.
- **Python API –** Salt provides a simple programming interface and it was designed to be modular and easily extensible, to make it easy to mold to diverse applications.
- **Easy to Setup –** Salt is easy to setup and provides a single remote execution architecture that can manage the diverse requirements of any number of servers.
- **Language Agnostic –** Salt state configuration files, templating engine or file type supports any type of language.

Benefits of SaltStack

Being simple as well as a feature-rich system, Salt provides many benefits and they can be summarized as below:

- **Robust –** Salt is powerful and robust configuration management framework and works around tens of thousands of systems.
- **Authentication –** Salt manages simple SSH key pairs for authentication.
- **Secure –** Salt manages secure data using an encrypted protocol.
- **Fast –** Salt is very fast, lightweight communication bus to provide the foundation for a remote execution engine.
- **Virtual Machine Automation –** The Salt Virt Cloud Controller capability is used for automation.
- **Infrastructure as data, not code –** Salt provides a simple deployment, model-driven configuration management and command execution framework.

Introduction to ZeroMQ

Salt is based on the **ZeroMQ** library and it is an embeddable networking library. It is lightweight and a fast messaging library. The basic implementation is in **C/C++** and native implementations for several languages including **Java** and **.Net** is available.

ZeroMQ is a broker-less peer-peer message processing. ZeroMQ allows you to design a complex communication system easily.

ZeroMQ comes with the following five basic patterns:

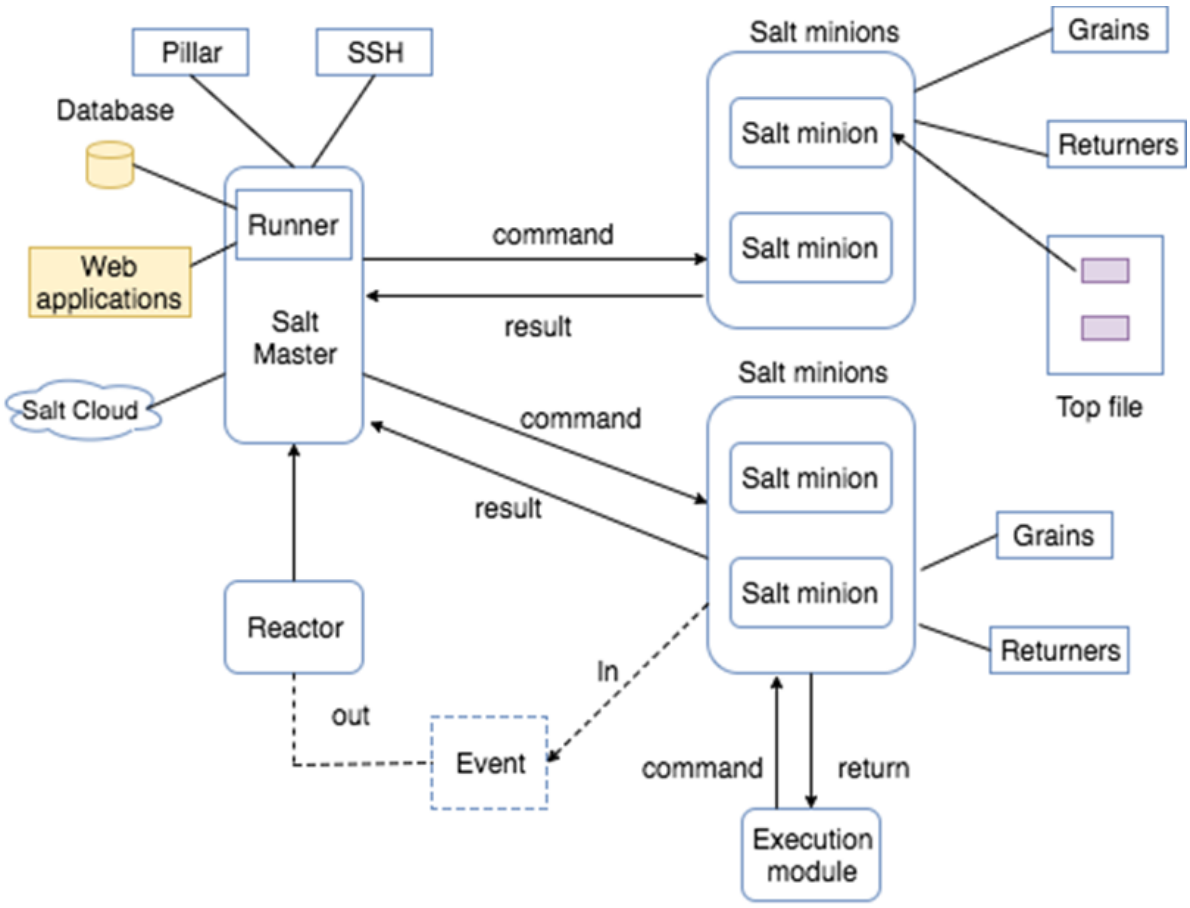
- **Synchronous Request/Response –** Used for sending a request and receiving subsequent replies for each one sent.
- **Asynchronous Request/Response –** Requestor initiates the conversation by sending a Request message and waits for a Response message. Provider waits for the incoming Request messages and replies with the Response messages.
- **Publish/Subscribe –** Used for distributing data from a single process (e.g. publisher) to multiple recipients (e.g. subscribers).
- **Push/Pull –** Used for distributing data to connected nodes.
- **Exclusive Pair –** Used for connecting two peers together, forming a pair.

ZeroMQ is a highly flexible networking tool for exchanging messages among clusters, cloud and other multi system environments. ZeroMQ is the **default transport library** presented in SaltStack.

2. SaltStack – Architecture

The architecture of SaltStack is designed to work with any number of servers, from local network systems to other deployments across different data centers. Architecture is a simple server/client model with the needed functionality built into a single set of daemons.

Take a look at the following illustration. It shows the different components of SaltStack architecture.



Eg: Salt '*' disk.usage

- **SaltMaster –** SaltMaster is the master daemon. A SaltMaster is used to send commands and configurations to the Salt slaves. A single master can manage multiple masters.

- **SaltMinions –** SaltMinion is the slave daemon. A Salt minion receives commands and configuration from the SaltMaster.

- **Execution –** Modules and Adhoc commands executed from the command line against one or more minions. It performs Real-time Monitoring.

- **Formulas –** Formulas are pre-written Salt States. They are as open-ended as Salt States themselves and can be used for tasks such as installing a package, configuring and starting a service, setting up users or permissions and many other common tasks.

- **Grains –** Grains is an interface that provides information specific to a minion. The information available through the grains interface is static. Grains get loaded when the Salt minion starts. This means that the information in grains is unchanging. Therefore, grains information could be about the running kernel or the operating system. It is case insensitive.

- **Pillar –** A pillar is an interface that generates and stores highly sensitive data specific to a particular minion, such as cryptographic keys and passwords. It stores data in a key/value pair and the data is managed in a similar way as the Salt State Tree.

- **Top File -** Matches Salt states and pillar data to Salt minions.

- **Runners –** It is a module located inside the SaltMaster and performs tasks such as job status, connection status, read data from external APIs, query connected salt minions and more.

- **Returners –** Returns data from Salt minions to another system.

- **Reactor –** It is responsible for triggering reactions when events occur in your SaltStack environment.

- **SaltCloud –** Salt Cloud provides a powerful interface to interact with cloud hosts.

- **SaltSSH –** Run Salt commands over SSH on systems without using Salt minion.

In the next chapter, we will learn in detail about the various competitors of SaltStack and their features.

3. SaltStack – Competitors

Salt, Puppet, Chef, and Ansible are the leading configuration management and orchestration tools, each of which takes a different path to server automation. They were built to make it easier to configure and maintain dozens, hundreds or even thousands of servers.

Let us understand how SaltStack competes primarily with Puppet, Chef, and Ansible.

Platforms and Support

Following is a list of all the platforms that support SaltStack and its competitors.

- **SaltStack** – SaltStack software runs on and manages many versions of Linux, Windows, Mac OS X and UNIX.
- **Puppet** – Red Hat Enterprise Linux, CentOS, Oracle Linux, Scientific Linux, SUSE Linux Enterprise Server and Ubuntu.
- **Chef** – Chef is supported on multiple platforms such as AIX, RHEL/CentOS, FreeBSD, OS X, Solaris, Microsoft Windows and Ubuntu.
- **Ansible** – Fedora distribution of Linux, CentOS, and Scientific Linux via Extra Packages for Enterprise Linux (EPEL) as well as for other operating systems.

Origin Language

- **SaltStack** – Python
- **Puppet** – Ruby
- **Chef** – Ruby and its CLI uses ruby-based DSL
- **Ansible** – Python

Supported Language

- **SaltStack** - Any language
- **Puppet** - Ruby
- **Chef** - Ruby and its CLI uses ruby-based DSL
- **Ansible** - Any language

Web UI

- **SaltStack** – Web UI offers views of running jobs, minion status and event logs.
- **Puppet** – Web UI handles reporting, inventorying and real-time node management.

- **Chef** – Web UI lets you search and inventory nodes, view node activity and assign Cookbooks, roles and nodes.
- **Ansible** – Web UI lets you configure users, teams and inventories and apply Playbooks to inventories.

Management Tools

- **SaltStack** – SaltStack Enterprise is positioned as the main tool for managing the orchestration of cloud and IT operations, as well as **DevOps**.
- **Puppet** – Puppet comes in two flavors, Puppet Enterprise and Open Source Puppet. In addition to providing functionalities of the Open Source Puppet, Puppet Enterprise also provides GUI, API and command line tools for node management.
- **Chef** – CFEngine is the configuration management tool.
- **Ansible** – Ansible 1.3 is the main tool for management.

Performance

- **SaltStack** – Salt is designed for high-performance and scalability. Salt's communication system establishes a persistent data pipe between the Salt master and minions using ZeroMQ.
- **Puppet** – Secure as well as high-performance and no agents required.
- **Chef** - The most apparent struggle for Chef Server is search; Search is slow and is not requested concurrently from clients.
- **Ansible** - Secure, high-performance and no agents required.

Price and Value

- **SaltStack** – Free open source version. SaltStack Enterprise costs $150 per machine per year.
- **Puppet** - Free open source version. Puppet Enterprise costs $100 per machine per year.
- **Chef** - Free open source version; Enterprise Chef free for 5 machines, $120 per month for 20 machines, $300 per month for 50 machines.
- **Ansible** - Free open source version; Ansible free for 10 machines, then $100 or $250 per machine per year depending on the support you needed.

Usage

- **SaltStack** – SaltStack is used by Cisco and Rackspace. It can integrate with any cloud-based platform.
- **Puppet** – Puppet is used by Zynga, Twitter, the New York Stock Exchange, PayPal, Disney, Google and so on.
- **Chef** – Chef can integrate with cloud-based platforms such as Internap, Amazon EC2, Google Cloud Platform, OpenStack, Microsoft Azure and Rackspace.
- **Ansible** – Ansible can deploy to virtualization environments, cloud environments including Amazon Web Services, Cloud Stack, DigitalOcean, and Google Cloud Platform and so on.

End of ebook preview

If you liked what you saw…

Buy it from our store @ **https://store.tutorialspoint.com**