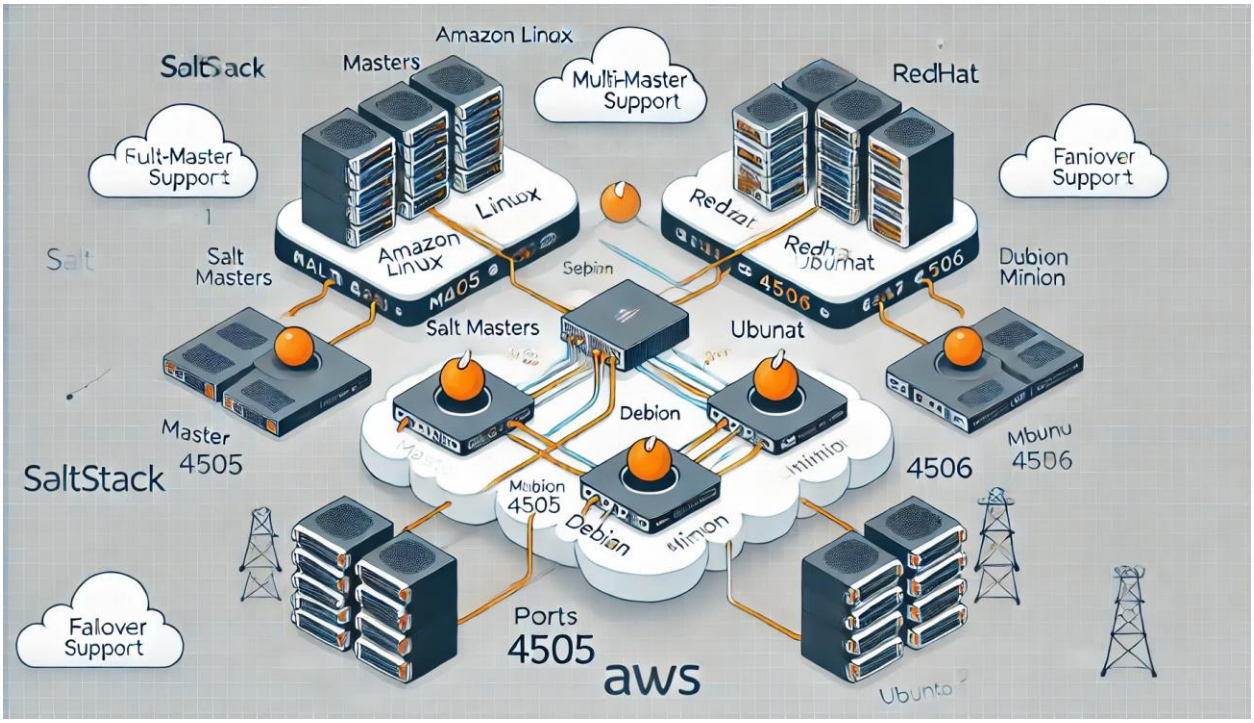
cprime

SaltStack Multi-Master Architecture with Failover Support



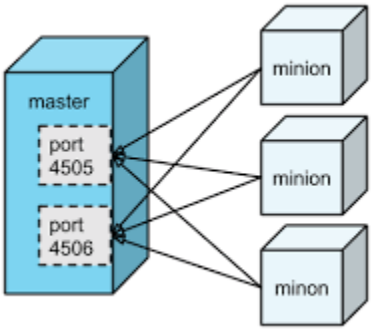
NAME : T MANOJ

EMPID : LYAKE2KHS

Introduction

Salt Stack is a powerful **configuration management** and **orchestration** tool designed for **automation** and **infrastructure management**. It enables IT teams to manage thousands of systems in a highly efficient, scalable, and reliable manner.

A **Multi-Master Configuration with Failover Support** ensures high availability by allowing minions to communicate with multiple masters. If the primary master fails, minions automatically switch to the secondary master, preventing downtime and ensuring continuous operations.



Overview

The Salt Stack Multi-Master setup involves:

- **Two Masters:** Running on Amazon Linux and RedHat EC2 instances.
- **Two Minions:** Running on Debian and Ubuntu EC2 instances.
- **Failover Mechanism:** Minions will automatically switch to the secondary master if the primary one fails.

Key Features

- **High Availability** – Ensures minions can still receive configurations even if one master is down.
- **Scalability** – Supports large-scale infrastructure automation.

- **Security** – Uses encrypted communication over ports **4505** and **4506**.
- **Centralized Management** – Enables centralized control over thousands of nodes.

Definition of Each Service Used

3.1 Salt Master

The Salt Master is the central server that manages minions. It sends commands and receives responses from Salt Minions over an encrypted channel.

Functions:

- Manages configuration states.
- Sends commands to minions.
- Stores execution logs and job results.

3.2 Salt Minion

A Salt Minion is an agent installed on client machines that listens for commands from the master and executes them.

Functions:

- Applies configuration states.
- Executes commands sent by the master.
- Sends execution results back to the master.

3.3 Failover Mechanism

Failover ensures that if the primary Salt Master becomes unavailable, minions will automatically switch to a secondary master for uninterrupted service.

Functions:

- Detects the failure of the primary master.

- Automatically redirects communication to the secondary master.

3.4 EC2 Instances

Amazon EC2 (Elastic Compute Cloud) provides virtual machines (instances) to host Salt Masters and Minions.

Functions:

- Provides scalable cloud computing resources.
- Hosts Linux-based SaltStack components.
- Ensures on-demand availability.

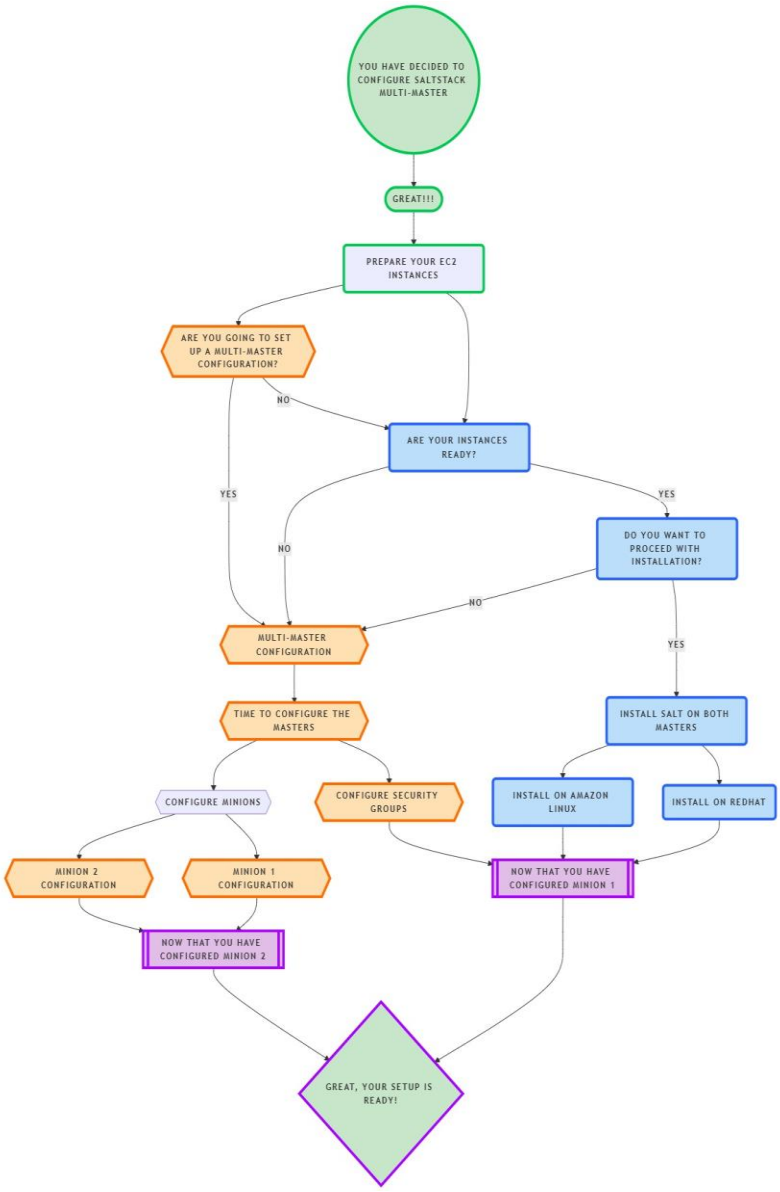
3.5 Security Groups

AWS Security Groups act as firewalls to control inbound and outbound traffic.

Functions:

- Allow Salt communication on **ports 4505 and 4506**.
- Permit SSH access for configuration and management.

Architecture/Workflow :

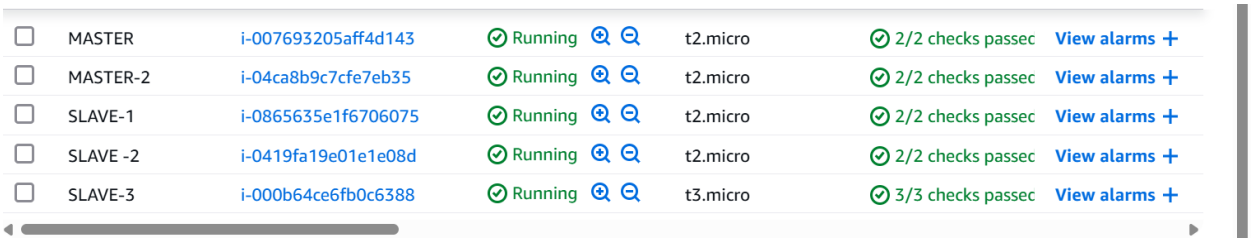


Prerequisites

- **EC2 Instances:**
  - **Master 1:** Amazon Linux
  - **Master 2:** RedHat
  - **Minion 1:** Debian
  - **Minion 2:** Ubuntu

- **Security Group:** Ensure that ports **22 (SSH)**, **4505**, and **4506 (Salt communication)** are open between the masters and minions.
- **Downloaded Script:** Download the `bootstrap-salt.sh` script:

  `curl -o bootstrap-salt.sh -L` https://github.com/saltstack/salt-bootstrap/releases/latest/download/bootstrap-salt.sh



MASTER	i-007693205aff4d143	Running	t2.micro	2/2 checks passed	View alarms +
MASTER-2	i-04ca8b9c7cfe7eb35	Running	t2.micro	2/2 checks passed	View alarms +
SLAVE-1	i-0865635e1f6706075	Running	t2.micro	2/2 checks passed	View alarms +
SLAVE -2	i-0419fa19e01e1e08d	Running	t2.micro	2/2 checks passed	View alarms +
SLAVE-3	i-000b64ce6fb0c6388	Running	t3.micro	3/3 checks passed	View alarms +

2. Install Salt Master on Both Masters

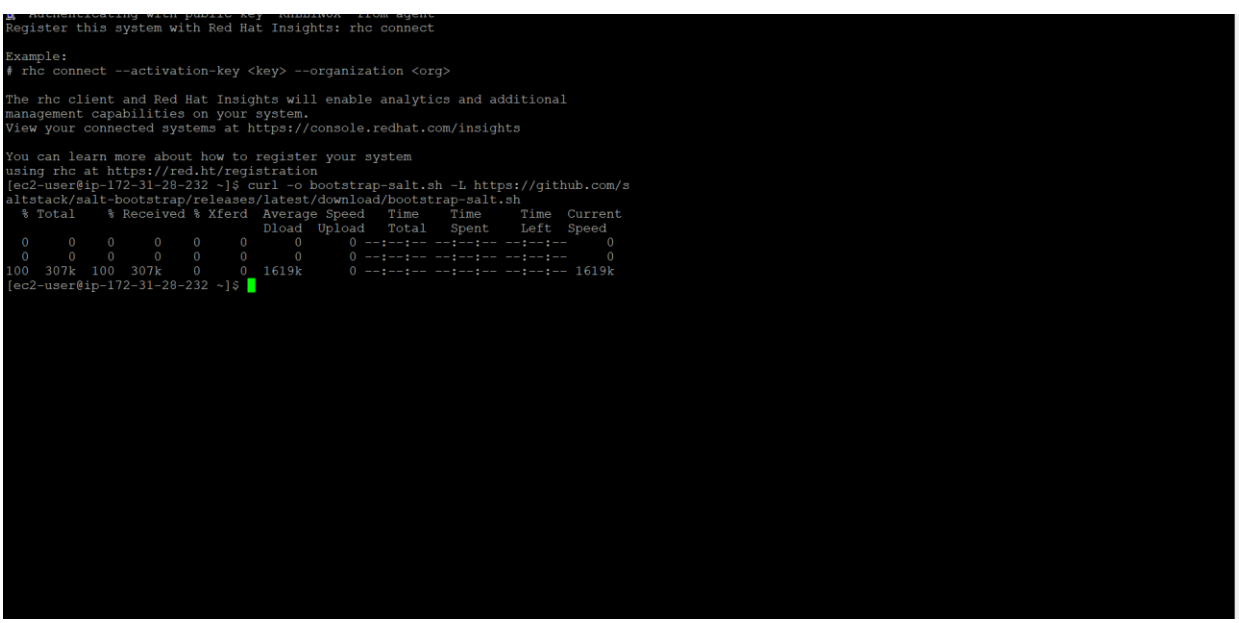
On Master 1 (Amazon Linux):

1. Make the script executable:

   ```
   chmod +x bootstrap-salt.sh
   ```

2. Install Salt Master:

   ```
   sudo ./bootstrap-salt.sh -M -P
   ```



```
Register this system with Red Hat Insights: rhc connect

Example:
# rhc connect --activation-key <key> --organization <org>

The rhc client and Red Hat Insights will enable analytics and additional
management capabilities on your system.
View your connected systems at https://console.redhat.com/insights

You can learn more about how to register your system
using rhc at https://red.ht/registration
[ec2-user@ip-172-31-28-232 ~]$ curl -o bootstrap-salt.sh -L https://github.com/s
altstack/salt-bootstrap/releases/latest/download/bootstrap-salt.sh
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100  307k  100  307k    0     0  1619k      0 --:--:-- --:--:-- --:--:-- 1619k
[ec2-user@ip-172-31-28-232 ~]$
```

```
Unable to read consumer identity

This system is not registered with an entitlement server. You can use "rhc" or "subscription-manager" to register.

Salt Repo for Salt LATEST release              5.3 kB/s | 1.5 kB     00:00
Red Hat Enterprise Linux 9 for x86_64 - AppStre  92 kB/s | 4.5 kB     00:00
Red Hat Enterprise Linux 9 for x86_64 - BaseOS   83 kB/s | 4.1 kB     00:00
Red Hat Enterprise Linux 9 Client Configuration  33 kB/s | 1.5 kB     00:00
Metadata cache created.
Updating Subscription Management repositories.
Unable to read consumer identity

This system is not registered with an entitlement server. You can use "rhc" or "subscription-manager" to register.

Last metadata expiration check: 0:00:01 ago on Fri 07 Mar 2025 05:34:00 AM UTC.
Available Packages
salt-minion.aarch64                  3007.1-0                  salt-repo-latest
salt-minion.x86_64                   3007.1-0                  salt-repo-latest
Updating Subscription Management repositories.
Unable to read consumer identity

This system is not registered with an entitlement server. You can use "rhc" or "subscription-manager" to register.

Last metadata expiration check: 0:00:02 ago on Fri 07 Mar 2025 05:34:00 AM UTC.
Dependencies resolved.
================================================================================
 Package            Architecture  Version         Repository              Size
================================================================================
Installing:
 salt-master        x86_64        3007.1-0        salt-repo-latest        3.5 M
 salt-minion        x86_64        3007.1-0        salt-repo-latest        285 k
Installing dependencies:
 salt               x86_64        3007.1-0        salt-repo-latest         50 M

Transaction Summary
================================================================================
Install  3 Packages

Total download size: 54 M
Installed size: 156 M
Downloading Packages:
(1/3): salt-minion-3007.1-0.x86_64.rpm          343 kB/s | 285 kB     00:00
(2/3): salt-master-3007.1-0.x86_64.rpm          2.8 MB/s | 3.5 MB     00:01
```

```
[ec2-user@ip-172-31-29-202 ~]$ ll
total 308
-rw-r--r--. 1 ec2-user ec2-user 315185 Mar  9 12:11 bootstrap-salt.sh
[ec2-user@ip-172-31-29-202 ~]$
```

3. Start and enable the Salt Master service:

```
sudo systemctl enable --now salt-master
```

4. Verify the status:

```
sudo systemctl status salt-master
```

```
● salt-master.service - The Salt Master Server
     Loaded: loaded (/usr/lib/systemd/system/salt-master.service; enabled; pres>
     Active: active (running) since Sun 2025-03-09 14:24:54 UTC; 1h 22min ago
       Docs: man:salt-master(1)
             file:///usr/share/doc/salt/html/contents.html
             https://docs.saltproject.io/en/latest/contents.html
   Main PID: 62382 (/opt/saltstack/)
      Tasks: 42 (limit: 4377)
     Memory: 388.7M
        CPU: 36.661s
     CGroup: /system.slice/salt-master.service
             ├─62382 "/opt/saltstack/salt/bin/python3.10 /usr/bin/salt-master M>
             ├─62437 "/opt/saltstack/salt/bin/python3.10 /usr/bin/salt-master P>
             ├─62438 "/opt/saltstack/salt/bin/python3.10 /usr/bin/salt-master E>
             ├─62439 "/opt/saltstack/salt/bin/python3.10 /usr/bin/salt-master E>
             ├─62443 "/opt/saltstack/salt/bin/python3.10 /usr/bin/salt-master R>
             ├─62444 "/opt/saltstack/salt/bin/python3.10 /usr/bin/salt-master R>
             ├─62446 "/opt/saltstack/salt/bin/python3.10 /usr/bin/salt-master R>
             ├─62447 "/opt/saltstack/salt/bin/python3.10 /usr/bin/salt-master R>
             ├─62448 "/opt/saltstack/salt/bin/python3.10 /usr/bin/salt-master R>
             ├─62449 "/opt/saltstack/salt/bin/python3.10 /usr/bin/salt-master R>
             ├─62450 "/opt/saltstack/salt/bin/python3.10 /usr/bin/salt-master R>
             ├─78347 "/opt/saltstack/salt/bin/python3.10 /usr/bin/salt-master F>
lines 1-23
```

On Master 2 (RedHat):

1. Make the script executable:

```
chmod +x bootstrap-salt.sh
```

2. Install Salt Master:

```
sudo ./bootstrap-salt.sh -M -P
```

```
ec2-user@ip-172-31-16-135:~
[ec2-user@ip-172-31-16-135 ~]$ ll
total 312
-rw-r--r--. 1 ec2-user ec2-user 315185 Mar  9 12:11 bootstrap-salt.sh
-rw-r--r--. 1 root     root        104 Mar  9 13:52 reconnect_primary.sls
[ec2-user@ip-172-31-16-135 ~]$
```

3. Start and enable the Salt Master service:

```
sudo systemctl enable --now salt-master
```

4. Verify the status:

```
sudo systemctl status salt-master
```

```
● salt-master.service - The Salt Master Server
     Loaded: loaded (/usr/lib/systemd/system/salt-master.service; enabled; pres>
     Active: active (running) since Sun 2025-03-09 14:24:54 UTC; 1h 22min ago
       Docs: man:salt-master(1)
             file:///usr/share/doc/salt/html/contents.html
             https://docs.saltproject.io/en/latest/contents.html
   Main PID: 62382 (/opt/saltstack/)
      Tasks: 42 (limit: 4377)
     Memory: 388.7M
        CPU: 36.661s
     CGroup: /system.slice/salt-master.service
             ├─62382 "/opt/saltstack/salt/bin/python3.10 /usr/bin/salt-master M>
             ├─62437 "/opt/saltstack/salt/bin/python3.10 /usr/bin/salt-master P>
             ├─62438 "/opt/saltstack/salt/bin/python3.10 /usr/bin/salt-master E>
             ├─62439 "/opt/saltstack/salt/bin/python3.10 /usr/bin/salt-master E>
             ├─62443 "/opt/saltstack/salt/bin/python3.10 /usr/bin/salt-master R>
             ├─62444 "/opt/saltstack/salt/bin/python3.10 /usr/bin/salt-master R>
             ├─62446 "/opt/saltstack/salt/bin/python3.10 /usr/bin/salt-master R>
             ├─62447 "/opt/saltstack/salt/bin/python3.10 /usr/bin/salt-master R>
             ├─62448 "/opt/saltstack/salt/bin/python3.10 /usr/bin/salt-master R>
             ├─62449 "/opt/saltstack/salt/bin/python3.10 /usr/bin/salt-master R>
             ├─62450 "/opt/saltstack/salt/bin/python3.10 /usr/bin/salt-master R>
             ├─78347 "/opt/saltstack/salt/bin/python3.10 /usr/bin/salt-master F>
lines 1-23
```

3. Install Salt Minion on Both Minions

On Minion 1 (Debian):

1. Make the script executable:

```
chmod +x bootstrap-salt.sh
```

2. Install Salt Minion:

```
sudo sh ./bootstrap-salt.sh -P
```

```
total 312
-rw-r--r--. 1 ec2-user ec2-user 315185 Mar  9 12:11 bootstrap-salt.sh
```

3. Edit the Minion configuration file:

```
sudo nano /etc/salt/minion
```

Add the following lines:

```
master:
  - <172.31.29.202>
```

- <172.31.16.135>
master_type: failover
master_alive_interval: 30

```
# Set the location of the salt master server. If the master server cannot be
# resolved, then the minion will fail to start.
master:
- 172.31.29.202
- 172.31.16.135
```

```
# Setting master_type to 'disable' lets you have a running minion (with engi>
# beacons) without a master connection
master_type: failover

# Poll interval in seconds for checking if the master is still there.  Only
# respected if master_type above is "failover". To disable the interval enti>
# set the value to -1. (This may be necessary on machines which have high nu>
# of TCP connections, such as load balancers.)
master_alive_interval: 30

# If the minion is in multi-master mode and the master_type configuration op>
# is set to "failover", this setting can be set to "True" to force the minior
# to fail back to the first master in the list if the first master is back o>
master_failback: True
```

```
# If the minion is in multi-master mode, the "master_type" configuration is >
# "failover", and the "master_failback" option is enabled, the master failba>
# interval can be set to ping the top master with this interval, in seconds.
master_failback_interval: 30

# Set whether the minion should connect to the master via IPv6:
#ipv6: False

# Set the number of seconds to wait before attempting to resolve
# the master hostname if name resolution fails. Defaults to 30 seconds.
# Set to zero if the minion should shutdown and not retry.
retry_dns: 0
```


```
# If the minion is in multi-master mode, the "master_type" configuration is >
# "failover", and the "master_failback" option is enabled, the master failba>
# interval can be set to ping the top master with this interval, in seconds.
master_failback_interval: 30

# Set whether the minion should connect to the master via IPv6:
#ipv6: False

# Set the number of seconds to wait before attempting to resolve
# the master hostname if name resolution fails. Defaults to 30 seconds.
# Set to zero if the minion should shutdown and not retry.
retry_dns: 0
```


4. Restart the Salt Minion service:

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

On Minion 2 (Ubuntu):

1. Make the script executable:

```
chmod +x bootstrap-salt.sh
```

2. Install Salt Minion:

```
sudo sh ./bootstrap-salt.sh -P
```

```
ubuntu@ip-172-31-30-135:~$ ll
total 336
drwxr-x--- 4 ubuntu ubuntu   4096 Mar  9 12:12 ./
drwxr-xr-x 3 root   root     4096 Mar  9 12:02 ../
-rw-r--r-- 1 ubuntu ubuntu    220 Mar 31  2024 .bash_logout
-rw-r--r-- 1 ubuntu ubuntu   3771 Mar 31  2024 .bashrc
drwx------ 2 ubuntu ubuntu   4096 Mar  9 12:08 .cache/
-rw-r--r-- 1 ubuntu ubuntu    807 Mar 31  2024 .profile
drwx------ 2 ubuntu ubuntu   4096 Mar  9 12:02 .ssh/
-rw-r--r-- 1 ubuntu ubuntu      0 Mar  9 12:12 .sudo_as_admin_successful
-rw-rw-r-- 1 ubuntu ubuntu 315185 Mar  9 12:11 bootstrap-salt.sh
ubuntu@ip-172-31-30-135:~$
```

3. Edit the Minion configuration file:

```
sudo nano /etc/salt/minion
```

Add the following lines:

```
master:
  - <172.31.29.202>
  - <172.31.16.135>
master_type: failover
master_alive_interval: 30
```

```
# Set the location of the salt master server. If the master server cannot be
# resolved, then the minion will fail to start.
master:
- 172.31.29.202
- 172.31.16.135
```

```
# Setting master_type to 'disable' lets you have a running minion (with engi>
# beacons) without a master connection
master_type: failover

# Poll interval in seconds for checking if the master is still there.  Only
# respected if master_type above is "failover". To disable the interval enti>
# set the value to -1. (This may be necessary on machines which have high nu>
# of TCP connections, such as load balancers.)
master_alive_interval: 30

# If the minion is in multi-master mode and the master_type configuration op>
# is set to "failover", this setting can be set to "True" to force the minion
# to fail back to the first master in the list if the first master is back o>
master_failback: True
```

```
# If the minion is in multi-master mode, the "master_type" configuration is >
# "failover", and the "master_failback" option is enabled, the master failba>
# interval can be set to ping the top master with this interval, in seconds.
master_failback_interval: 30

# Set whether the minion should connect to the master via IPv6:
#ipv6: False

# Set the number of seconds to wait before attempting to resolve
# the master hostname if name resolution fails. Defaults to 30 seconds.
# Set to zero if the minion should shutdown and not retry.
retry_dns: 0
```


```
# If the minion is in multi-master mode, the "master_type" configuration is >
# "failover", and the "master_failback" option is enabled, the master failba>
# interval can be set to ping the top master with this interval, in seconds.
master_failback_interval: 30

# Set whether the minion should connect to the master via IPv6:
#ipv6: False

# Set the number of seconds to wait before attempting to resolve
# the master hostname if name resolution fails. Defaults to 30 seconds.
# Set to zero if the minion should shutdown and not retry.
retry_dns: 0
```


4. Restart the Salt Minion service:

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

4. Accept Minion Keys on Both Masters

On Master 1 & Master 2:

1. List pending minion keys:

```
sudo salt-key
```

2. Accept all minion keys:

```
sudo salt-key -A
```

3. Verify connectivity:

```
sudo salt '*' test.ping
```

```
[ec2-user@ip-172-31-16-135 ~]$ sudo salt-key
Accepted Keys:
ip-172-31-23-35
ip-172-31-30-135.ec2.internal
Denied Keys:
Unaccepted Keys:
Rejected Keys:
[ec2-user@ip-172-31-16-135 ~]$
```

```
ec2-user@ip-172-31-29-202:~
[ec2-user@ip-172-31-29-202 ~]$ sudo salt-key
Accepted Keys:
ip-172-31-23-35
ip-172-31-30-135.ec2.internal
Denied Keys:
Unaccepted Keys:
Rejected Keys:
[ec2-user@ip-172-31-29-202 ~]$
```

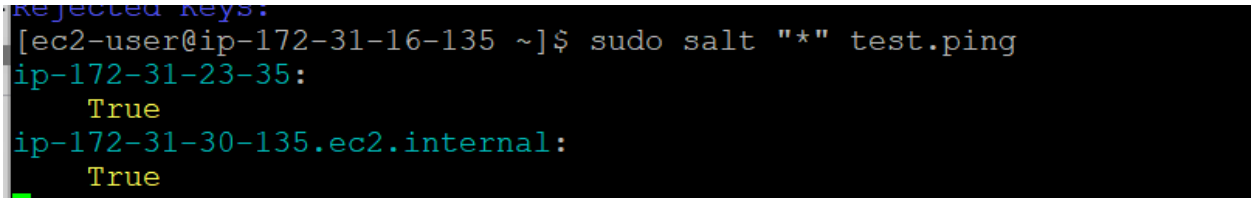
```
[ec2-user@ip-172-31-16-135 ~]$ sudo salt "*" test.ping
ip-172-31-23-35:
    True
ip-172-31-30-135.ec2.internal:
    True
```

5. Test the Multi-Master Setup

On Master 1 & Master 2:

1. Ping all minions:

```
sudo salt '*' test.ping
```



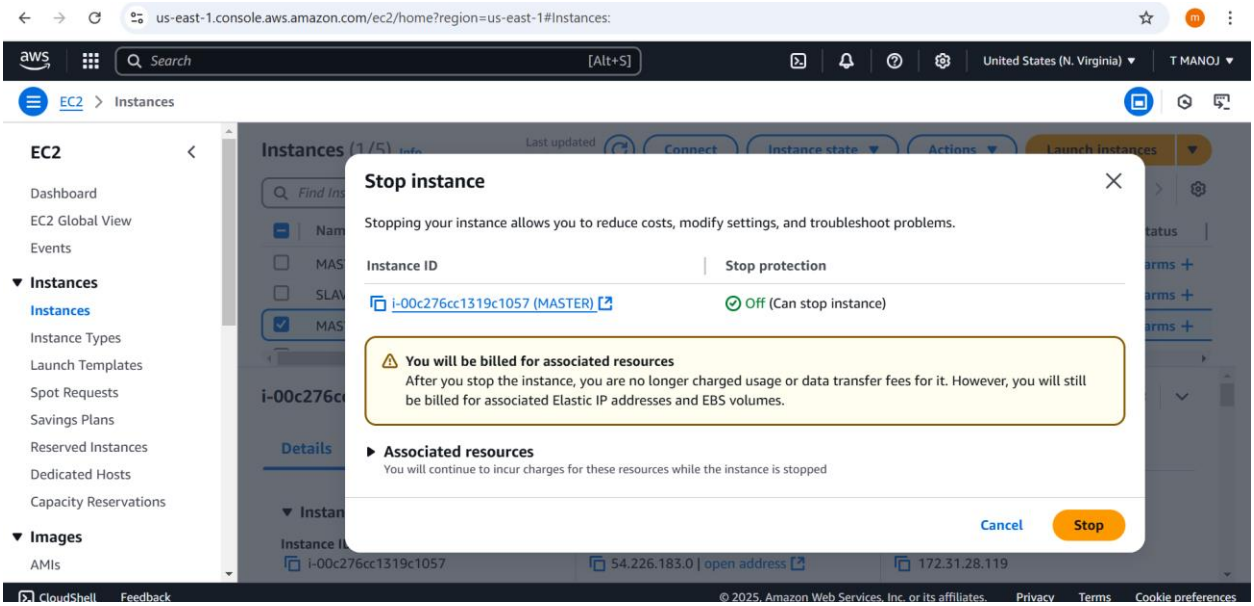
Expected Output:

```
debian:
    True
ubuntu:
    True
```

Testing Failover:

1. **Stop Master 1:**

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



2. **Check if Minions Switch to Master 2:**

```
sudo salt '*' test.ping
```

```
admin@ip-172-31-23-35:~$ sudo salt-call test.ping
[WARNING ] Master ip address changed from 172.31.29.202 to 172.31.16.135
local:
    True
admin@ip-172-31-23-35:~$
```

```
[WARNING ] Master ip address changed from 172.31.29.202 to 172.31.16.135
local:
    True
ubuntu@ip-172-31-30-135:~$
```

Minions will respond from Master 2.

Advantages of SaltStack Multi-Master Configuration

1 High Availability

With a failover mechanism, minions can continue operations even if a master fails.

2 Scalability

Easily scales to manage thousands of nodes, making it suitable for enterprise environments.

3 Security

Utilizes encrypted communication channels for secure configuration management.

4 Centralized Automation

Provides a single control point for managing system configurations and states.

5 Speed and Efficiency

Utilizes event-driven architecture to rapidly apply changes across systems.

6. Conclusion

Your SaltStack Multi-Master setup with failover is now complete. This configuration ensures high availability by allowing minions to automatically switch to the secondary master if the primary master becomes unavailable. You can now use this setup for **configuration management, automation, and orchestration** using SaltStack.