Salt-terminology::
https://www.digitalocean.com/community/tutorials/an-introduction-to-saltstack-terminology-and-concepts

****

Install and Configure Salt Master and Minion Servers on Ubuntu 14.04

****

SaltStack is a powerful, flexible, high performing configuration management and remote execution system. It can be used manage your infrastructure from a centralized location while minimizing manual maintenance steps.

In this article, we will focus on getting a Salt master server set up to manage your infrastructure. We will also demonstrate how to install the Salt minion daemon on other computers in order to manage them with Salt. We will be using two Ubuntu 14.04 servers to demonstrate these steps.

Install the Master Daemon

The Salt master daemon can be installed in a number of ways on Ubuntu 14.04. The following is a brief rundown of the advantages and disadvantages of each method:

- **Ubuntu SaltStack PPA**: Uses the Ubuntu native package management tools to install and update the required software. This is the easiest method of install but, as is the case at the time of this writing, the packages can be significantly out-of-date.
- **Salt-Bootstrap**: This bootstrapping script attempts to provide a more universal method for installing and configuring Salt. It can attempt to use the native software tools available, which means that it may still try to install from the PPA above. It also provides easy access to the development versions of Salt.

In this documentation i mentioned only PPA installation not bootstrap, if you need that then go to :

https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-salt-master-and-minion-servers-on-ubuntu-14-04#configure-the-minion

It is up to you to decide which option suits your needs best. If you run into issues, there might be bug fixes available in the development version. However, there is also a chance of running into newly released bugs.

Install the Stable Version from the Official PPA

To get started, you will need to add the SaltStack PPA to the server you will use as your master. You can do this by typing:

```
#  add-apt-repository ppa:saltstack/salt
```

if it says that add-apt not found, then install it first

 # apt-get install -y python-software-properties software-properties-common

 # add-apt-repository ppa:saltstack/salt

Once you have confirmed the PPA addition, it will be added to your system. To index the new packages available within, you will need to update your local package index. Afterwards, you can install the relevant software:

 # apt-get update

 # apt-get install salt-master salt-minion salt-ssh salt-cloud salt-doc

In the above command, we installed both the Salt master and minion daemons. This will allow us to control our master server with Salt as well. We also installed salt-ssh and salt-cloud, which give us more flexibility in how we connect to and control resources. We've included the documentation package as well.

Initial Master Configuration

First, we will create the configuration management directory structure where the Salt master will look for various files. These are all under the ***/srv*** directory by default. We need ***/srv/salt*** and ***/srv/pillar*** to get started. Create them now by typing:

 # mkdir -p /srv/{salt,pillar,formulas}

Modify the Salt Master Configuration

 # nano /etc/salt/master

The first thing we will do is set the file_roots dictionary. This basically specifies the locations where the Salt master will look for configuration management instructions. The base specifies the default environment. Two of the directories we created earlier will be used for this purpose. The ***/srv/salt*** will be used for administrator-created instructions, and the ***/srv/formulas*** will be set aside for pre-packaged configurations downloaded from external sources:
we will will set up the root directory for our Salt pillar configuration.

```
file_roots:
  base:
    - /srv/salt
    - /srv/formulas

pillar_roots:
  base:
    - /srv/pillar
```

Modify the Salt Minion Configuration

We also installed the Salt minion daemon on this machine so that we can keep it in line with the rest of our infrastructure policies. Open the Salt minion configuration with sudo privileges next:

 # nano /etc/salt/minion

master: salt # my system hostname is salt

 # /etc/init.d/salt-master start
 # /etc/init.d/salt-minion start

Accept the Minion Key

listing all of the keys that the Salt master has knowledge of:

 # salt-key --list all

Accepted Keys:
Denied Keys:
Unaccepted Keys:
salt
Rejected Keys:

As you can see, our Salt minion has sent its key to the master, but it has not been accepted yet. For security purposes, before accepting the key, we will run two commands.
We need to make sure the output of this (which tells us the fingerprint of the key the Salt minion generated):

 # salt-call key.finger --local

Output
local:
24:c8:77:1d:ed:10:d7:b0:3e:bc:bc:ed:41:e1:5a:d1

Matches the fingerprint found here (the fingerprint of the key that the Salt master is being asked to accept). Substitute the minion ID here:

 # salt-key -f salt

Unaccepted Keys:
saltmaster: 24:c8:77:1d:ed:10:d7:b0:3e:bc:bc:ed:41:e1:5a:d1

Once you verify that those values are the same, you can accept the key by typing:

 # salt-key -a salt

After accepting the key, you can see that the key has been moved to the "Accepted Keys" section:

```
# salt-key --list all
```

Output
```
Accepted Keys:
salt
Denied Keys:
Unaccepted Keys:
Rejected Keys:
```

Now, you can test that the Salt master and minion processes are communicating correctly by typing:

```
#  salt '*' test.ping
```

You should receive a message back indicating that the health check was successful:

Output
```
salt:
    True
```

Install a Separate Minion

Now that we have our Salt master server up and running smoothly, we can demonstrate how to bring a new server under Salt's control as a minion.

Again, we have multiple ways of installing the necessary software, but **you should match the method used for the master server**. This will ensure that you do not have a version mismatch between Salt master and minion. Salt minions that are more up-to-date than their master server may exhibit unpredictable behavior.

Install the Stable Master from the Official PPA

If you installed your Salt master server from the SaltStack PPA, you can add the same PPA on your Ubuntu minion server:

```
#  add-apt-repository ppa:saltstack/salt
```

This time, we only need to install the salt-minion executable. Update your local package index after adding the PPA and install the software by typing:

```
#  apt-get update
#  apt-get install salt-minion
```

Configure the Minion

Now that we have the minion installed, we can go ahead and configure it to communicate with our Salt master.

Before we begin, we should grab the Salt master's key fingerprint. We can add this to our minion configuration for increased security.

On your Salt master server, type:

 # salt-key -F master

The output should look something like this:

Output
Local Keys:
master.pem: 12:db:25:3d:7f:00:a3:ed:20:55:94:ca:18:f8:67:97
master.pub: 7b:97:23:4b:a4:6d:16:31:2d:c9:e3:81:e2:d5:32:92
Accepted Keys:
salt: 24:c8:77:1d:ed:10:d7:b0:3e:bc:bc:ed:41:e1:5a:d1

The value of the master.pub key, located under the "Local Keys" section is the fingerprint we are looking for. Copy this value to use in our Minion configuration.

Modify the Minion Configuration

Back on your new Salt minion, open the minion configuration file with sudo privileges:

 # nano /etc/host

192.168.1.60 salt # my system hostname is salt

 # nano /etc/salt/minion

We need to specify the location where the Salt master can be found. This can either be a resolvable DNS domain name or an IP address:
and set the master_finger option to the fingerprint value you copied from the Salt master a moment ago:

master: salt # my system hostname is salt

master_finger: '7b:97:23:4b:a4:6d:16:31:2d:c9:e3:81:e2:d5:32:92'

Save and close the file when you are finished.

Now, restart the Salt minion daemon to implement your new configuration changes:

The new minion should contact the Salt master service at the provided address. It will then send its key for the master to accept. In order to securely verify the key, need to check the key fingerprint on the new minion server.

 # /etc/init.d/salt-minion start
 # salt-call key.finger –local

You should see output that looks like this:

Output
local:
32:2a:7c:9a:f2:0c:d1:db:84:df:d3:82:00:d5:8f:be

You will need to verify that the key fingerprint that the master server received matches this value.

 # salt-key -f salt-minion-1 # my client hostname is salt-minion-1

Output
Unaccepted Keys:
salt-minion-1: 32:2a:7c:9a:f2:0c:d1:db:84:df:d3:82:00:d5:8f:be

if it is same then accept it

 # salt-key --list all

Output
Accepted Keys:
salt
Denied Keys:
Unaccepted Keys:
salt-minion-1
Rejected Keys:

 # salt-key -a salt-minion-1

 # salt-key --list all

Output
Accepted Keys:
salt
salt-minion-1
Denied Keys:
Unaccepted Keys:
Rejected Keys:

 # salt '*' test.ping

You should receive back answers from both of the minion daemons you've configured:

Output
salt-minion-1:
True
salt:
True

links ::

installation and configuration >>
https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-salt-master-and-minion-servers-on-ubuntu-14-04#configure-the-minion

****

Installing and Configuring Halite for Web-Interface

****

To begin the installation of Halite from PyPI, you'll need to install pip. The Salt package, as well as the bootstrap, do not install pip by default.

 # apt-get install python-pip

 # pip install -U halite

Depending on the webserver you want to run halite through, you'll need to install that piece as well.

Here im using cherrypy

 # pip install cherrypy==3.2.3

Configuring Halite Permissions

Configuring Halite access permissions is easy. By default, you only need to ensure that the @runner group is configured. In the /etc/salt/master file, uncomment and modify the following lines:

 # nano /etc/salt/master

```
external_auth:
  pam:
    salt-user:
      - .*
      - '@runner'
      - '@wheel'
```

Currently Halite allows, but does not require, any wheel modules.
here my user is salt-user and i added that user in sudo group

 # usermod -aG sudo salt-user

Configuring Halite Settings

For testing use self-signed ssl certificates but in production use proper certs

First create ssl certs for that server

 # openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl1/certs/server.key -out /etc/ssl1/certs/server.crt

so the .key and .crt files are now under /etc/ssl1/certs

Once you've configured the permissions for Halite, you'll need to set up the Halite settings in the /etc/salt/master file. Halite supports CherryPy, Paste, and Gevent out of the box.
To configure cherrypy, add the following to the bottom of your /etc/salt/master file:

 # nano /etc/salt/master

```
halite:
  level: 'debug'
  server: 'cherrypy'
  host: '192.168.1.60'                    # my host ip
  port: '8080'
  cors: False
  tls: True                               # set true for ssl
  certpath: '/etc/ssl1/certs/server.crt'  # ssl cert path
  keypath: '/etc/ssl1/certs/server.key'   # ssl key path
# pempath: '/etc/pki/tls/certs/localhost.pem'
```

provide the server name in /etc/hosts for access with server name

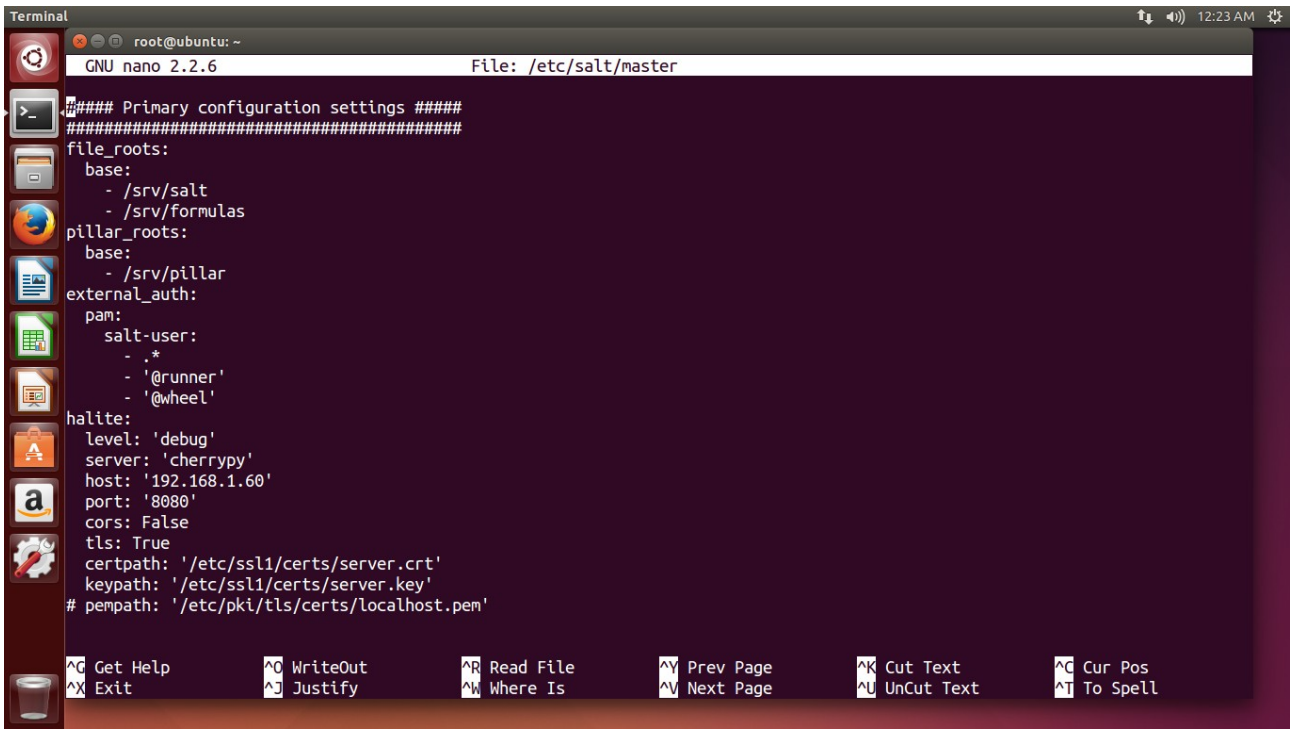
 # nano /etc/hosts

192.168.1.60   salt

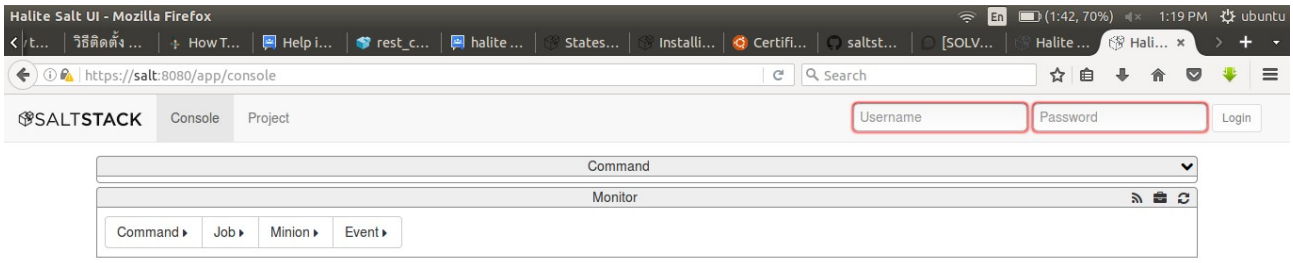
restart the service

 # /etc/init.d/salt-master restart

Once you've configured the halite section of your /etc/salt/master, you can restart the salt-master service, and your halite instance will be available. Depending on your configuration, the instance will be available either at https://localhost:8080/app, https://domain:8080/app, or https://123.456.789.012:8080/app .

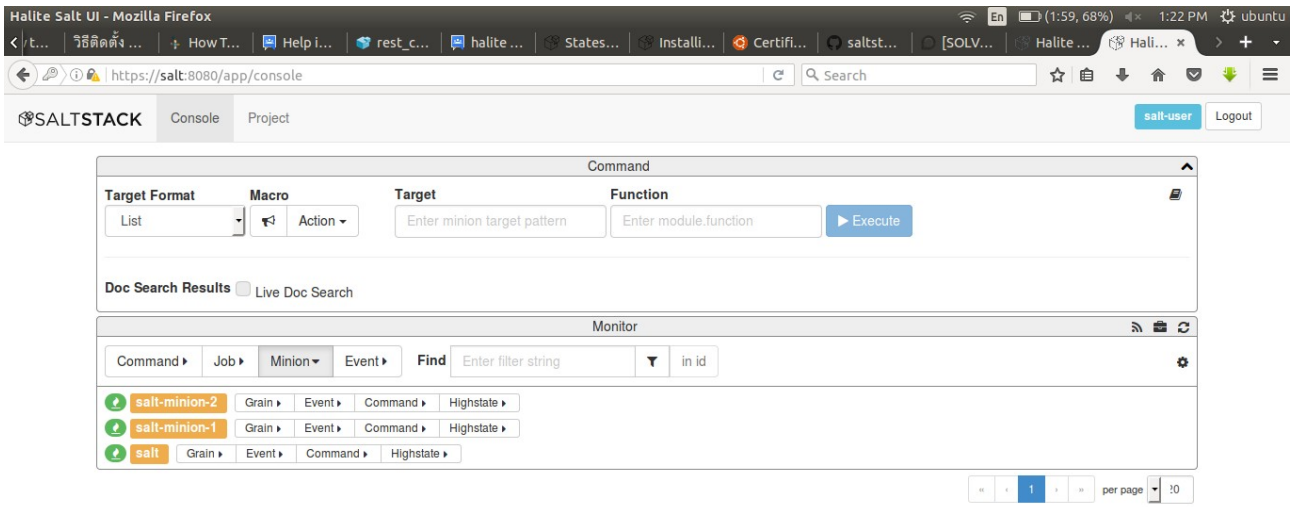


All logs relating to halite are logged to the default /var/log/salt/master file.



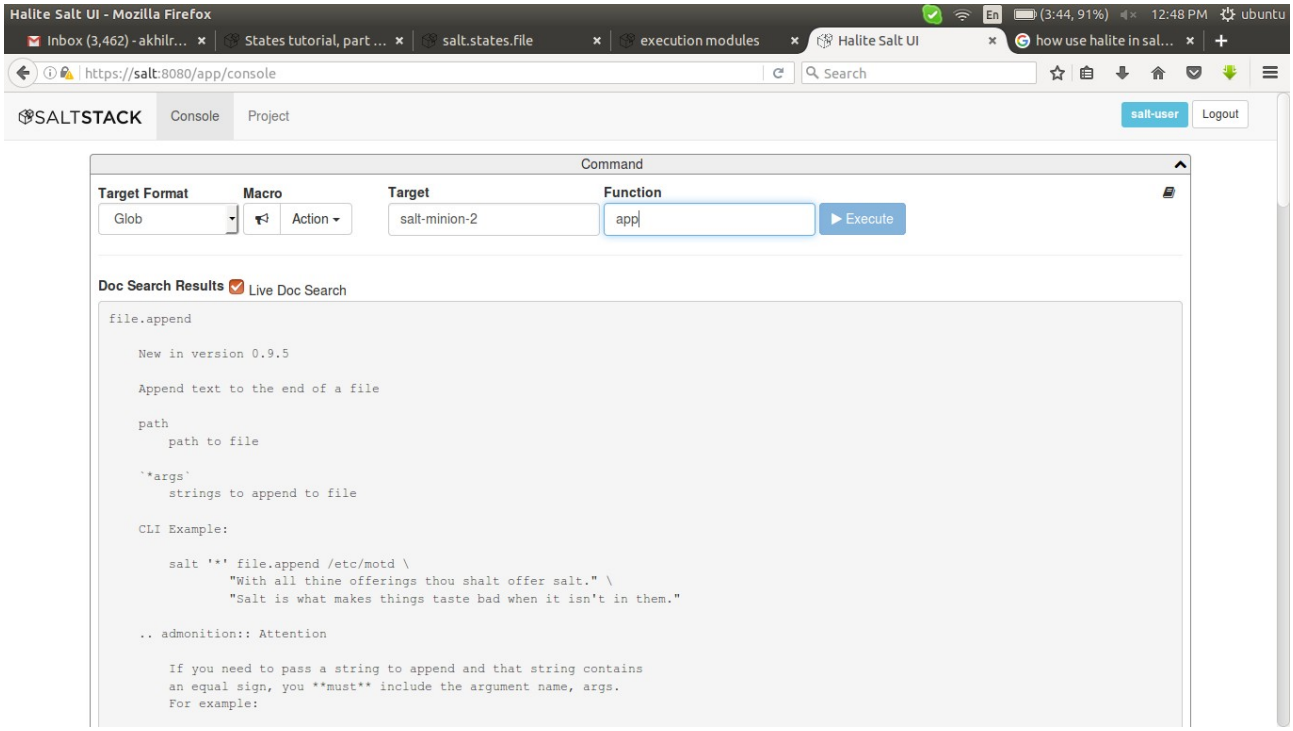
Login as user which created before,
in my server, I created a user with name of “salt-user”

then command >> list >> running >> shows all minions



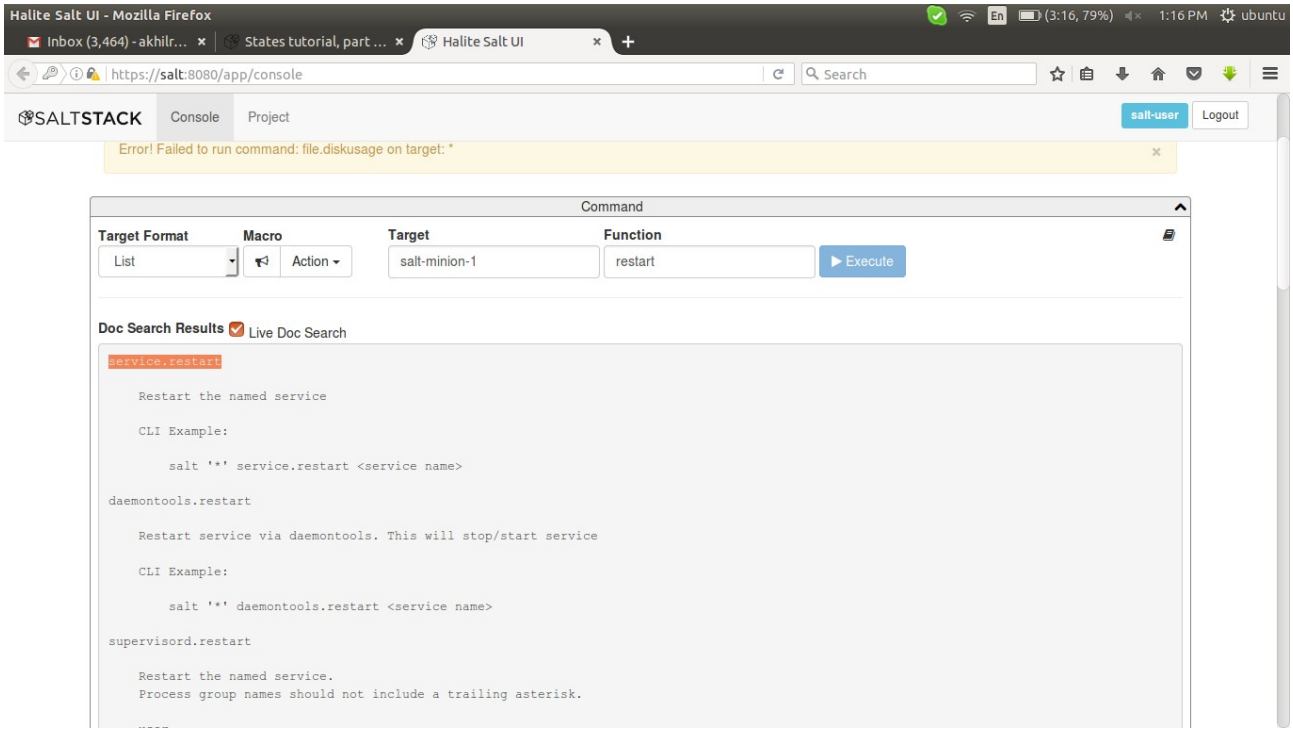
to get indormations about the commands in web-interface

mark on Live Doc Search and refresh on monitor bar below the mark and then type anything which you want to know about salt as follow,

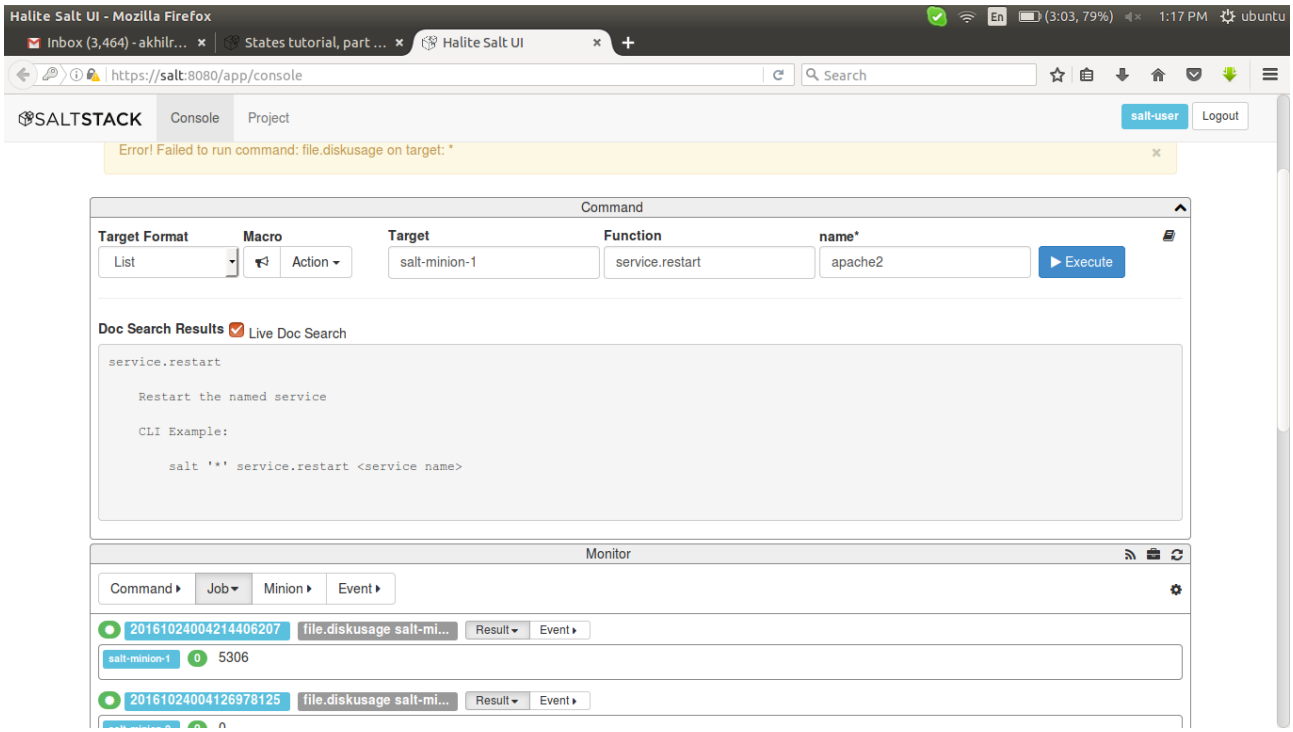


we can start service, and manage the minions by web-interface,
1) search the function for the particular execution eg: service as above

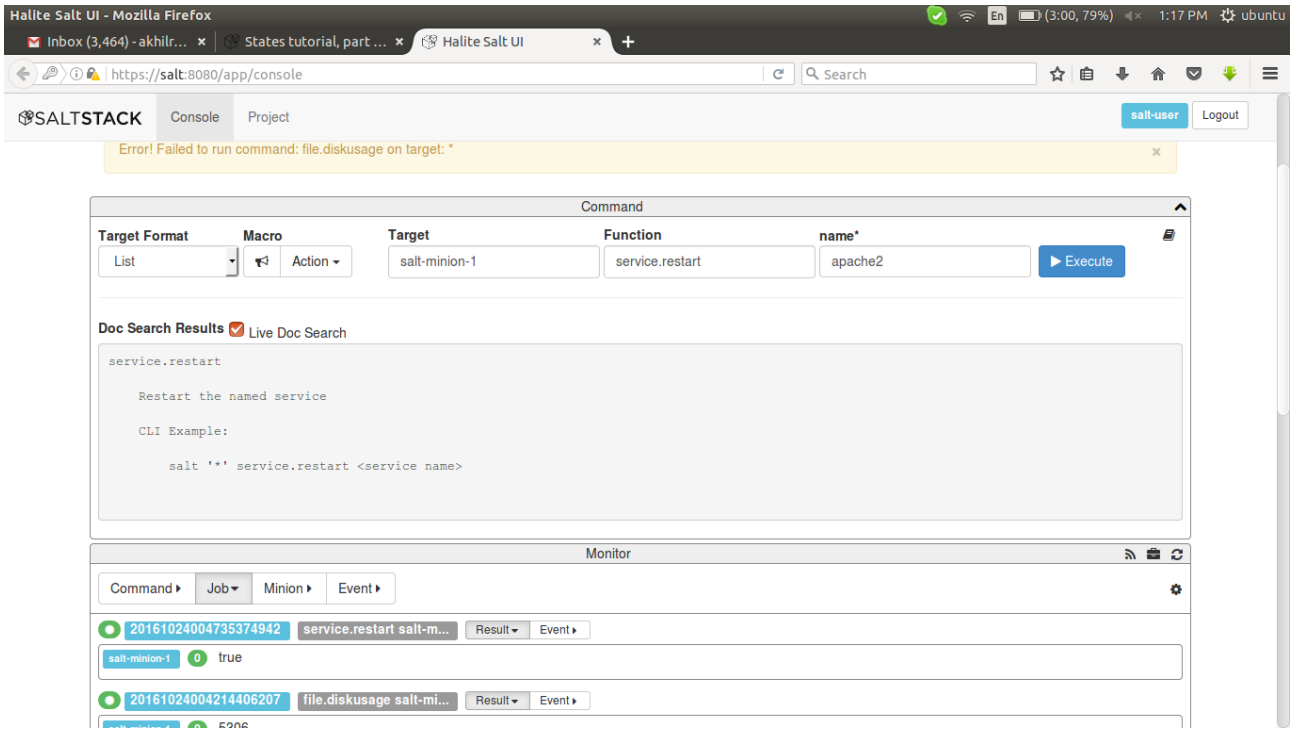
2) then copy and paste the command in function path and fill the next column info with appropriate info,



here i restart the apache2 because i installed apache2 in there



after i executing the output shown under jobs >> result



links ::
configuration >>
https://docs.saltstack.com/en/latest/topics/tutorials/halite.html

solutions for cherrypy >>
https://discourse.codeemo.com/t/solved-ssl-received-a-record-that-exceeded-the-maximum-permissible-length/45/2

****

States

****

To apply the state to minions

 # salt '*' state.apply # execute to all minions
 # salt 'salt-minion-1' state.apply # only for particular minion

****

create a new file called top.sls and add the following:

```
base:
  'salt-minion-1':
   - ubuntu
```

it means
minion name >> salt-minion-1
>> '*' # for all minions
>> 'salt-minion*' # for minions name starting with salt-minion
ubuntu >> name of .sls file to be run on minion

****

- watch: # used for depends on some other commands
- require: # used for depends on some other commands

- append: # append some data

- onchanges: # execute only if any changes in depended commands

- source: # add some files to minions from master

****

create directory under /{home-dir-of-master}/
eg : /srv/salt/files
so, source:
- salt://files/name

****

to install package

```
apache2:                >>      # ID declaration
 pkg:                   >>      # state declaration
  - installed     >>      # function declaration
```

****

to install and start that service

```
apache2:
 pkg:
  - installed
 service:
  - running
  - enable: true
  - start: true
  - restart: true
  - watch:
    - pkg: apache2
```

****

to create file

```
/home/new:
 file:
  - managed
  - user: root
  - group: root
  - mode: 777
  - contents: "akhil , hai how are you"
  - source:
    - salt://files/new
    - salt://files/new1
```

****

to append someting to a existing file

```
/home/new:
 file:
 - append
 - text: "this is appen"
```

****

to create directory

```
/home/akhil0:
 file:
  - directory
  - mode: 777
  - user: root
  - group: root
```

****

to add some files into a directory

```
/home/akhil:
 file:
  - user: root
  - group: root
  - dir_mode: 744
  - file_mode: 777
  - recurse
  - source:
    - salt://files
```

****

to create group

```
new-group:
 group:
 - present
 - gid: 1300
 - members:
   - root
   - test1
```

****

to create user

```
test1:
 user:
 - present
 - uid: 1500
 - gid: 1500
 - shell: /bin/bash
 - password: test1
 - groups:
   - new-group1
 - watch:
   - group: new-group1
```

****

to run commands in minions

```
/home/mysql.sh:
 file:
 - managed
 - source:
   - salt://files/mysql.sh
 - mode: 777

 cmd:
 - run
 - cwd: /
 - require:
   - file: /home/mysql.sh
 - onchanges:
   - file: /home/mysql.sh
```

****

links::

entire-details>>
https://docs.saltstack.com

basic state commands>>
https://docs.saltstack.com/en/latest/topics/tutorials/states_pt1.html

all-states>>
https://docs.saltstack.com/en/latest/ref/index.html

execute a command>>
https://docs.saltstack.com/en/latest/ref/states/all/salt.states.cmd.html

group>>

https://docs.saltstack.com/en/latest/ref/states/all/salt.states.group.html

user>>
https://docs.saltstack.com/en/latest/ref/states/all/salt.states.user.html