

---

# **SaltWebGui Documentation**

***Release 0.0.1***

**Riccardo Scartozzi**

**Jun 12, 2017**



---

## Contents

---

<b>1</b>	<b>Licence</b>	<b>3</b>
<b>2</b>	<b>Contents</b>	<b>5</b>
2.1	Installation . . . . .	5
2.2	User manual . . . . .	8
2.3	Development and contributions . . . . .	11
<b>3</b>	<b>Indices and tables</b>	<b>13</b>



**SaltWebGui** is a web interface to saltstack.

It aims to be reliable, easy to install and simplify the saltstack administration, both for configuration management as well as orchestration.

The project is developed on [github](#) where is possible to fill also issues and push requests.



# CHAPTER 1

---

## Licence

---

**SaltWebGui** is © 2016–2017 by Riccardo Scartozzi (from [Software Workers srl](#)) and released under the terms of the GPL 3.





These documents aim to get you started with **SaltWebGui**:

## Installation

For running SaltWebGui it is recommended to use **virtualenv** in order to satisfy the dependencies (see [Virtualenv](#)).

It is also required an access to the **salt-api** (see [Saltstack requirements](#) for further explanation).

The components involved are:

- a web server (where to run SaltWebGui)
- a salt-master with accessible salt-api

It is possible to deploy the SaltWebGui's web server on the salt-master itself (in this case the salt-api can be kept private on localhost) or to have them on different server. In the latter be careful the salt-api is made accessible to the web-server: if it is widely accessible use a proper ssl encryption.

Here follows the installation requirements, then an example for [Quick installing for testing](#) and a [Reference for production installation](#).

## Requirements

Analysis of minimum requirements for running SaltWebGui.

- [Saltstack requirements](#): minimum configuration for salt and its salt-api
- [Virtualenv](#): python dependency installation

### Saltstack requirements

In order to use SaltWebGui it is required to enable **salt-api** on the salt master.

For example on a debian distro just install salt-api with

```
aptitude install salt-api
```

Then enable the salt-api module through the *salt-master*'s configuration file. Here a quick example suitable for local test (not safe for production):

```
rest_cherryypy:
  disable_ssl: True
  host: 127.0.0.1
  port: 8000
```

And set-up proper user permission (here enable system user *mylinuxuser*):

```
external_auth:
  pam:
    mylinuxuser:
      - ".*"
      - "@runner"
      - "@wheel"
```

The *mylinuxuser* has to be a system user (it is possible to create a custom one with `adduser mylinuxuser`).

---

**Note:** In order to use SaltWebGui an authentication method is required. At the time of writing saltstack supports both pam and LDAP, see the [eauth manual](#).

---

## Virtualenv

On a Debian system at least the following packages are required:

```
aptitude install libmysqlclient-dev python-dev libffi-dev gcc
```

For using a virtualenv locally run

```
cd SaltWebGui
virtualenv ./venv
source venv/bin/activate
pip install -r requirements.txt
```

Now it's possible to launch SaltWebGui from cli, like:

```
FLASK_APP=./wsgi.py flask run
```

## Quick installing for testing

This is a quick reference for having a debuggable installation for tests. I'd recommend to install it on salt-master (and access from an ssh tunnel) or to install it locally on you system and access the salt-api through a tunnel. Here I'll try the latter.

On salt-master:

1. install salt-api

```
aptitude install salt-api
```

2. enable salt-api adding the following configuration in `/etc/salt/master` adding the following options:

```
rest_cherryypy:
  disable_ssl: True
  host: 127.0.0.1
  port: 8000
external_auth:
  pam:
    mylinuxuser:
      - ".*"
      - "@runner"
      - "@wheel"
```

3. add `mylinuxuser`:

```
adduser mylinuxuser
```

4. restart the salt-master:

```
service salt-master restart
```

On your system:

1. clone the repository

```
git clone https://github.com/SoftwareWorkersSrl/SaltWebGui.git
```

2. install minimum required packages for pip dependency:

```
aptitude install libmysqlclient-dev python-dev libffi-dev gcc
```

3. set-up the virtualenv

```
cd saltwebgui
virtualenv ./venv
source venv/bin/activate
pip install -r requirements.txt
```

4. if the salt-api is not publicly accessible start an SSH tunnel

```
ssh -L 8000:127.0.0.1:8000 -o ServerAliveInterval=30 sshuser@saltmasterurl
```

5. start flask (by default it will be accessible on localhost at port 5000):

```
cd saltwebgui
source venv/bin/activate
FLASK_APP=./wsgi.py flask run
```

**Warning:** When deploying SaltWebGui, if deploying for production, be careful to properly secure enough the system!

Set SSL protection on both the salt-api and the SaltWebGui, and don't forget to use an enough secure password for the salt user.

## Reference for production installation

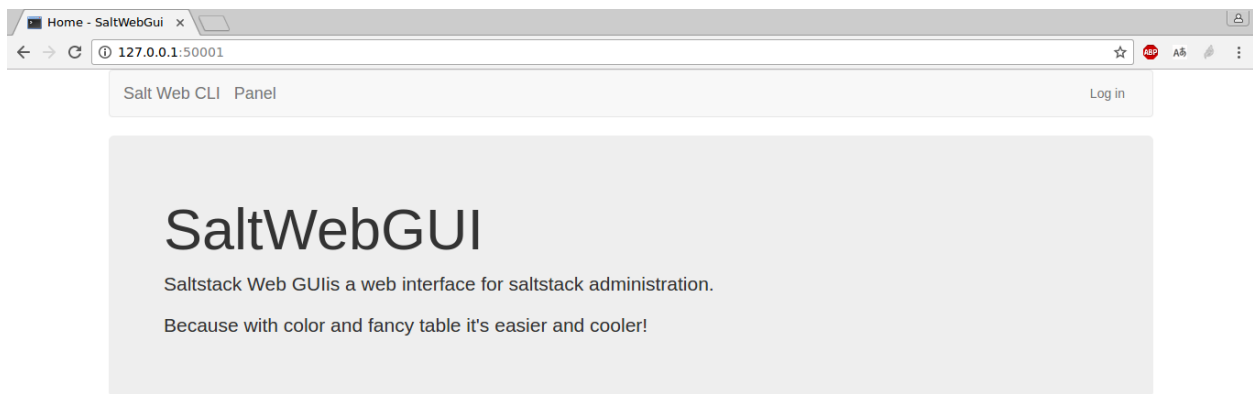
Sorry but this section is still to be implemented.

While waiting for more documentation, I'd suggest to use a WSGI server for running SaltWebGui (use the one that you prefer). Please also do not use the debug mode in production because of its security concerns.

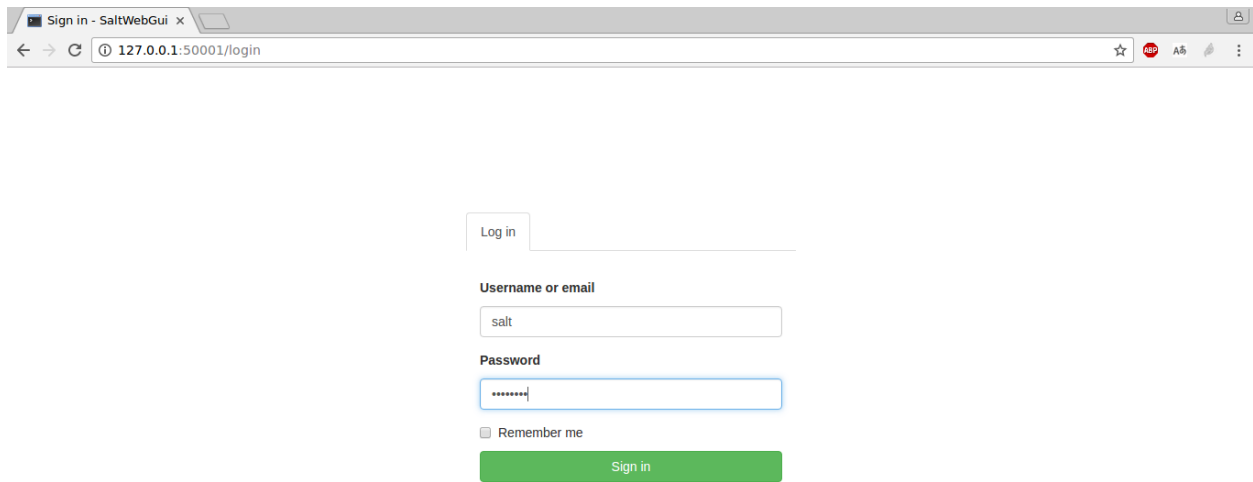
## User manual

Here a quick overview of the SaltWebGui functionality by screenshots.

Welcome to homepage!



From the homepage go to the login view (link at the right on the top bar) and perform a login:



Sign in - SaltWebGui x

127.0.0.1:50001/login

Log in

Username or email

salt

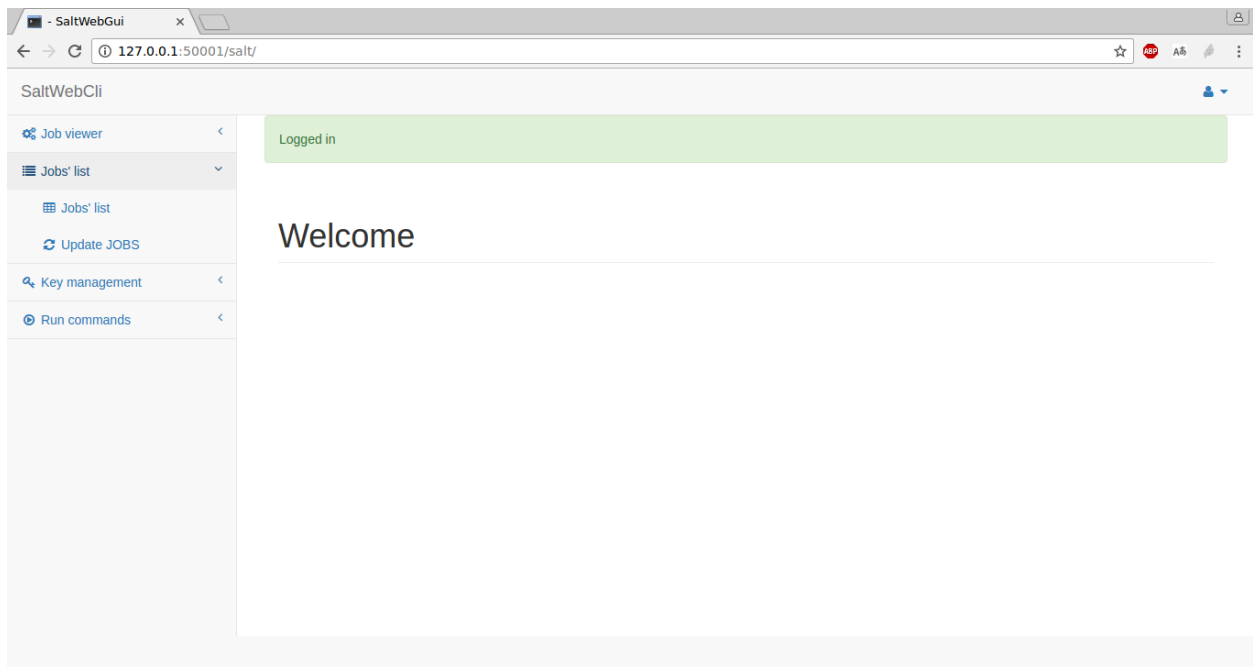
Password

\*\*\*\*\*

☐ Remember me

Sign in

Here we are on the panel view!



SaltWebCUI

Job viewer

Jobs' list

Jobs' list

Update JOBS

Key management

Run commands

Logged in

Welcome

From here first let's update the Job list (option on the left menu under *Jobs' list* > *Update JOBS*). You will be redirect back to a list of performed jobs:

SaltWebGui

Jobs list - SaltWebGui

127.0.0.1:50001/salt/jobs/update

SaltWebCll

Job viewer

Jobs' list

Jobs' list

Update JOBS

Key management

Run commands

## Tables

List of jobs.

Show 10 entries Search:

ID	Function	Target	Target-type	User	StartTime	Arguments
20170326151354157699 (raw)	mine.update	webfarm	glob	root	2017, Mar 26 15:13:54.157699	[]
20170326141354178173 (raw)	mine.update	webfarm	glob	root	2017, Mar 26 14:13:54.178173	[]
20170326131354195276 (raw)	mine.update	webfarm	glob	root	2017, Mar 26 13:13:54.195276	[]
20170326121354214477 (raw)	mine.update	webfarm	glob	root	2017, Mar 26 12:13:54.214477	[]
20170326111354334019 (raw)	mine.update	webfarm	glob	root	2017, Mar 26 11:13:54.334019	[]
20170326101354311415 (raw)	mine.update	webfarm	glob	root	2017, Mar 26 10:13:54.311415	[]
20170326091354361158 (raw)	mine.update	webfarm	glob	root	2017, Mar 26 09:13:54.361158	[]
20170326081354414878 (raw)	mine.update	webfarm	glob	root	2017, Mar 26 08:13:54.414878	[]
20170326071354410166 (raw)	mine.update	webfarm	glob	root	2017, Mar 26 07:13:54.410166	[]
20170326061354461065 (raw)	mine.update	webfarm	glob	root	2017, Mar 26 06:13:54.461065	[]

(the table is interactive for searching and sorting)

Let's try to run an highstate job. Enter the *run* view at the bottom of the left menu, insert the target and run:

SaltWebGui

127.0.0.1:50001/salt/run

SaltWebCll

Job viewer

Jobs' list

Key management

Run commands

Run command

## Run salt command

### Highstate

Target the minions

\*risca

test ☒ True ☐ False

Run highstate

### Any

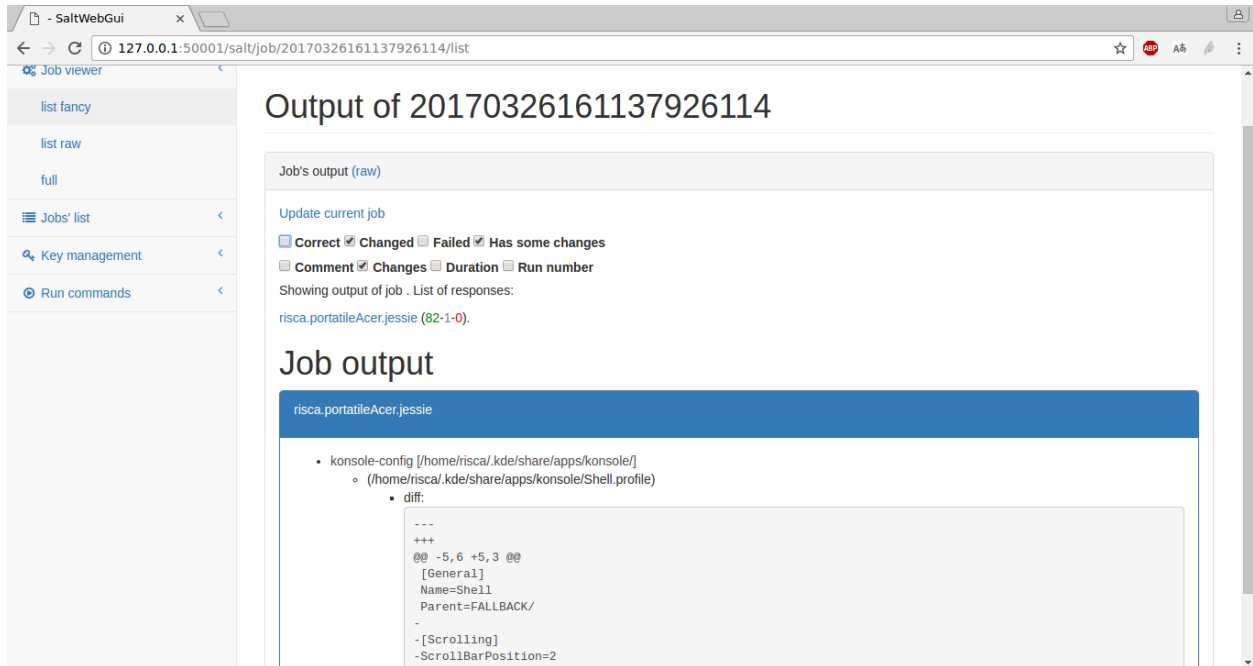
Target the minions

Type the saltstack command to run (like "cmd.run whoami")

test ☐ True ☒ False

Run command

After performing the job, go back to the jobs' list in order to open the launched job and see the output of the execution. If the job is still running nothing will be shown, just wait a little and update the view itself. See below here an example of the visualization of the job result (it's live filterable by output state):



## Development and contributions

### Push requests are welcome

Yep, this project is under active development!

Any contribution is welcome, please push against the *develop* branch. If you are developing special features that required heavy and custom changes create a custom branch.

Found a bug? Fill the [issue](#) on github.

### Coding style

Try to be conformed where possible to PEP8. If need help while using *VIM* I'd recommend the use of [syntastic](#).

### Principles

SaltWebGui is based on the python framework Flask.

By now all of the data are stored in custom variables or classes. This is valid for user management as well as for the data received from saltstack.

The first demo of the code was using directly the CLI (there is still a legacy code available under *saltwebgui/sat/salt\_binding.py*), but now it is preferred to user the salt-api. This way is possible to run SaltWebGui on any server as long as there is an access to salt's api.

The current implementation uses the python [pepper](#) module in order to bind to saltstack. For quick references under *saltwebgui/salt/views.py* are available some the following objects to interact with saltstack:

- JOBS = Jobs()
- JOB = Job()

- KEYS = Keys()
- RUN = Run()

The web interface is developed with bootstrap (and little of javascript code).



## CHAPTER 3

---

### Indices and tables

---

- `genindex`
- `search`



### A

architecture, [11](#)

### D

development, [11](#)

### E

examples, [8](#)

### I

Installation, [5](#)

### S

screenshot, [8](#)

### U

user manual, [8](#)