Salt Stack

How I learned to stop worrying and love the deployment

舊時代的 deploy 方式

```
$ ssh root@123.123.123.1
root@123.123.123.1's password:

$ apt-get install mysql-server

$ mysql -h localhost -u root

mysql> CREATE USER 'vinta'@'localhost'
IDENTIFIED BY '123';

mysql> CREATE DATABASE 'my_db';
```

舊時代的 deploy 方式

```
$ ssh root@123.123.123.2
root@123.123.123.2's password:

$ apt-get install nginx

$ vim /etc/nginx/nginx.conf

$ service nginx restart
```

舊時代的 deploy 方式

```
$ ssh root@123.123.123.3
root@123.123.123.3's password:

$ apt-get install git-core

$ git clone https://github.
com/StreetVoice/liuda.git
Cloning into 'liuda'...
Username for 'https://github.com':
```

雖然我們有 Fabric

但是 Fabric 也只是讓我們在一個 .py 的檔案裡寫 shell script 而已

Salt Stack 是什麼？

- Configuration Management
- Master / Minion (Slave)
- Concurrency
- Written in Python

Salt Stack 可以做到什麼程度？

所有你在 command line 可以做到的事情

全自動

為什麼用 Salt Stack？

- 不需要寫程式(如果你硬要寫，還是可以寫)
- 其實就是一堆 YAML 格式的文件而已
- 現代化的工具
- 帥

為什麼不用 Puppet 或 Chef？

因為它們的文件寫得太複雜了

SaLt State (SLS)

- 定義每一台機器的「狀態」
  - 要安裝哪些程式
    - ffmpeg
    - virtualenv
  - 要運行哪些服務
    - MySQL
    - RabbitMQ
    - NFS
- 以 YAML 格式寫成，可讀性高

SLS 長這個樣子

```
general-packages:
  pkg.installed:
    - names:
      - build-essential
      - git-core
      - htop
      - vim
```

Salt 內建的 states

- cmd
- file
- git
- mount
- mysql_database
- mysql_user
- pip
- pkg
- ssh_auth
- user
- virtualenv
- .....

範例

Setup MySQL

```
mysql-server-packages:
  pkg.installed:
    - names:
      - mysql-server
      - python-mysqldb

mysql-server:
  service.running:
    - name: mysql
    - require:
      - pkg: mysql-server-packages
    - watch:
      - file: /etc/mysql/my.cnf
```

Setup MySQL

```
mysql-user:
  mysql_user.present:
    - name: vinta
    - password: 123
    - host: localhost
    - require:
      - service: mysql-server

mysql-db:
  mysql_database.present:
    - name: my_db
    - require:
      - service: mysql-server
```

Template (Jinja2)

```
# /srv/pillar/settings.sls

system:
  user: liuda
  home_path: /home/liuda

project:
  name: liuda
  path: /home/liuda/liuda
  virtualenv_path: /home/liuda/.virtualenvs/liuda
  temp_path: /tmp/liuda

mysql:
  db: liuda_db
  user: liuda_user
  password: 123
```

Template (Jinja2)

```
mysql-user:
  mysql_user.present:
    - name: {{ pillar['mysql']['user'] }}
    - password: {{ pillar['mysql']['password'] }}
    - host: localhost
    - require:
      - service: mysql-server

mysql-db:
  mysql_database.present:
    - name: {{ pillar['mysql']['db'] }}
    - require:
      - service: mysql-server
```

Private Repo in GitHub?

- 用 ssh-keygen 產生一組全新的公鑰、私鑰
- 公鑰作為 GitHub repo 的 deploy key
- 把私鑰丟到 server 上
- 在 server 的 ~/.ssh/config 中指定連到 github.com 時使用這個私鑰

```
ssh_config:
  file.managed:
    - name: /home/liuda/.ssh/config
    - source: salt://ssh_keys/ssh_config
    - user: liuda
    - group: liuda
    - makedirs: True

github_private_key:
  file.managed:
    - name: /home/liuda/.ssh/github
    - source: salt://ssh_keys/github/id_rsa
    - user: liuda
    - group: liuda
    - makedirs: True
    - mode: 0400

github_public_key:
  file.managed:
    - name: /home/liuda/.ssh/github.pub
    - source: salt://ssh_keys/github/id_rsa.pub
    - user: liuda
    - group: liuda
    - makedirs: True
    - mode: 0400
```

Master / Minion

```
ubuntu@ip-10-146-158-242:~$ sudo salt-key -L
Accepted Keys:
echoprint
liuda-db
liuda-storage
liuda-vender
liuda-web1
liuda-web2
liuda-web3
Unaccepted Keys:
Rejected Keys:
ubuntu@ip-10-146-158-242:~$
```

top.sls -> main() function

```
base:
  '*':
    - salt.minion
    - common
    - ssh_keys

  'echoprint':
    - echoprint.server

  'liuda-db':
    - mysql.server

  'liuda-vender':
    - memcache.server
    - nfs.server
    - rabbitmq

  'liuda-web*':
    - nfs.client
    - django

  'liuda-web1':
    - django.syncdb
```

```
$ sudo salt '*' state.highstate
```

Resources

- Install & Usage
  - http://j.mp/11FaLFq
- Liuda 的 SLS
  - http://j.mp/1aiYPkl
- 其他 SLS
  - http://j.mp/11FaIt4
- GitHub Advanced Search