Deployment

Sajith Wickramasekara
12/05/11
overview

challenges

- users interacting with your application at any moment
- local flask server cannot handle heavy loads
- working in largely headless environment

topics covered

- finding a provider and configuring linux
- streamlining the maintenance process
- selecting a server and connecting it to your application
- automation and other advanced topics
overview

challenges

• users interacting with your application at any moment
• local flask server cannot handle heavy loads
• working in largely headless environment

topics covered

• finding a provider and configuring linux
• streamlining the maintenance process
• selecting a server and connecting it to your application
• automation and other advanced topics
hosting options

an entire spectrum of hosting options

• host it yourself
• shared hosting
• dedicated servers

requirements

• linux based
• shell (root) access
• reliable
virtual private servers

falls between shared hosting and a dedicated server

- emulates an independent box
- share resources with a few others
- satisfies all of our requirements

popular providers

![Linode](linode.com)

solid long term option ($20/month)

![Amazon Web Services](amazon.com)

canonical “cloud computing”: add/remove machines on demand, hourly billing, metered transfer ($25-40/month)
setting up and using ssh keys

public key cryptography is preferred to a password

- passwords are vulnerable
- keyed authentication transmits no secret information
- lends itself to deployment automation

$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/sajith/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/sajith/.ssh/id_rsa.
Your public key has been saved in /home/sajith/.ssh/id_rsa.pub.

$ ssh-copy-id -i .ssh/id_rsa.pub root@111.222.333.444

$ ssh root@111.222.333.444
basic linux configuration

with every fresh install, there are security vulnerabilities

- update and upgrade packages: windows update equivalent
  
  $ apt-get update
  $ apt-get upgrade --show-upgraded

- disable root logins, make a new account (and ssh keys)
  
  $ adduser sajith admin

additional security measures

- ufw: a basic firewall, block all but http and ssh ports
- tripwire: checks that linux binaries for malicious compromises
- fail2ban: blocks IPs that are denied access multiple times

see linux docs (e.g. https://help.ubuntu.com/community/Fail2ban)
overview

challenges

• users interacting with your application at any moment
• local flask server cannot handle heavy loads
• working in largely headless environment

topics covered

• finding a provider and configuring linux
  • **streamlining the maintenance process**
• selecting a server and connecting it to your application
• automation and other advanced topics
git hooks

scripts that fire at different stages of the git workflow

- stored in `myapp/.git/hooks`
- easy way to manage a staging and production area

the naïve approach

- clone repository on server
- login and pull changes
- better than sftp

git is decentralized

- push/pull to and from many repositories
- push to our remote server and use a post-receive hook
a post-receive hook

on the remote machine

```bash
$ mkdir myapp.git && cd myapp.git
$ git init --bare
Initialized empty Git repository in /home/sajith/myapp.git/
$ mkdir /var/www/www.myapp.com
$ cat > hooks/post-receive
#!/bin/sh
GIT_WORK_TREE=/var/www/www.myapp.com git checkout -f
$ chmod +x hooks/post-receive
```

and on your local dev machine

```bash
$ git remote add web ssh://sajith@111.222.333.444/~/myapp.git
$ git push web +master:refs/heads/master
```

```bash
$ git push web
```
python packaging

image from http://tinyurl.com/y97wpjx
pip for dependency management

easily create a list of package requirements

$ pip freeze > requirements.txt

install packages on your remote machine from a list

$ pip install -r requirements.txt
virtualenv manipulates PATH

creates an isolated python environment

$ pip install virtualenv
$ virtualenv myenv --no-site-packages
$ source myenv/bin/activate
...
$ deactivate

use if you work on multiple projects

• avoid polluting your global package namespace
• use different versions of the same package
• easily manage dependencies
overview

challenges

• users interacting with your application at any moment
• local flask server cannot handle heavy loads
• working in largely headless environment

topics covered

• finding a provider and configuring linux
• streamlining the maintenance process
• selecting a server and connecting it to your application
• automation and other advanced topics
wsgi is a low-level interface

bridges gap between frameworks and servers

• application is a callable object
• dictionary-like environment
• pluggable middleware

image from http://tinyurl.com/2mm3ov
choosing servers

two popular architecture styles

• embed python interpreter in server
  
  apache + mod_wsgi

• proxy requests using a HTTP server to a local python server
  
  fast http server
  python wsgi server
an example HTTP request

- nginx
  - static folder
  - gunicorn
  - myapp
configuring gunicorn

gunicorn is a fast WSGI server with a pre-fork worker model

$ pip install gunicorn

gunicorn.conf

    workers = 4
    bind = '127.0.0.1:5000
    proc_name = 'www.myapp.com'
    pidfile = '/tmp/www.myapp.com.pid'

deploy.py

    from werkzeug.contrib.fixers import ProxyFix
    from myapp import app
    app.wsgi_app = ProxyFix(app.wsgi_app)

and you’re good to go

$ gunicorn --config gunicorn.conf -daemon deploy:app
configuring nginx

nginx is an internet-ready server, runs as a service

$ sudo apt-get install nginx

nginx.conf

upstream frontends {
  server 127.0.0.1:5000;
}

server {
  listen 80;
  server_name myapp.com;

  access_log /var/log/nginx/localhost-access.log;
  error_log /var/log/nginx/localhost-error.log debug;

  location / {
    proxy_pass_header Server;
    proxy_set_header Host $http_host;
    proxy_set_header X-Forward-For $proxy_add_x_forwarded_for;
    proxy_redirect off;
    proxy_set_header X-Real_IP $remote_addr;
    proxy_set_header X-Scheme $scheme;
    proxy_pass http://frontends;
  }
  location /static {
    alias /home/myapp/myapp/static;
    autoindex on;
  }
}
configuring nginx

symlink and restart to go live

```bash
$ ln -s nginx.conf /etc/nginx/sites-enabled/myapp.conf
$ /etc/init.d/nginx restart
```
monitoring

profile requests using WSGI middleware
  • write your own or use pre-built ones (example here)

dealing with log files
  • flask’s logging module
  • gunicorn and nginx access/error logs

$ tail -f /path/to/your/logfile
overview

challenges
• users interacting with your application at any moment
• local flask server cannot handle heavy loads
• working in largely headless environment

topics covered
• finding a provider and configuring linux
• streamlining the maintenance process
• selecting a server and connecting it to your application
• automation and other advanced topics
fabric is a deployment library

eliminate the repetitiveness of deployment
  • encapsulate process as a python module
  • script everything, even shell commands
  • abstraction is good, can scale easily

example fab file operations
  • basic linux setup
  • check out code, install dependencies
  • run tests and conditionally commit or push
  • push to a production or staging server
scaling up

load balancing
  • achieve fault tolerance
  • distribute traffic across multiple machines
  • offered as a service by many provides (e.g. amazon)

databases scale very poorly
  • sharding is a form of horizontal partitioning
  • build indexes for columns you frequently select/join

caveat: prematurely overengineering
big ideas

finding a provider and configuring linux
  • basic command line skills go a long way

streamlining the maintenance process
  • minimize the number of steps to deploy code

selecting a server and connecting it to your application
  • choose based on community, documentation, and needs
  • log everything

automation and other advanced topics
  • scripted deployment is the only scalable option
additional resources

- Hitchhiker’s Guide to Packaging
- WSGI: A Series of Tubes
- Pro Git
- Fabric Documentation
- Gunicorn
- nginx
questions?