How to bootstrap a startup using Django

Philipp Wassibauer (@__philw__) & Jannis Leidel (@jezdez)
Gidsy is a place where anyone can explore, book and offer things to do.
Instantly book activities organized by real people.

Book and offer tours, local activities, workshops and other fun things to do... learn more
Why we chose Django

- Big community
- Network
- Language
- Many problems already solved
- The admin
Why Django is a good choice

- Proven technology by similar use cases
- Stable APIs in a well-defined release process
- Good documentation with focus on prose
- Huge community of 3rd party components
Haystack

- Customizable search abstraction
- Indexing, filtering, faceting, “More like this”
- Spatial search and sorting
Tastypie

- Highly customizable Web API library
- Hooks for auth, throttling, caching, etc.
- Backbone.js compatible
Celery

- Async code execution, cronjobs
- Thumbnails, search index updates, caching, etc.
- Collect stats without blocking
Memcache

- Periodic cache refreshing for high traffic sites
- Fragment caching with dates and cache version
- Cache warming during deployment
Main branch is always deployable
Development happens in feature branches
Code reviews via pull requests
Shared responsibility
Separation of fast and slow tests
Full test suite via Jenkins, soon Travis CI
Fast tests locally via tox
Virtualenv(wrapper) + pip
Localshop for in-house software releases
django-setcon for Django configuration
Foreman for process management
Initially infrastructure and deployments are simple
Scaling up

- Each server downloads dependencies
- External services could be down (PyPI, Github, ...)
- Which server is in charge of migrations, collectstatic?
Central deployment server that builds releases
Other servers download .tar package from there
.deploy system

- Builds are virtualenvs
- Atomic and orchestrated releases
- Collectstatic, migrate & other commands centralized
- Web interface for deploying, rollback
- Keeps information on who deployed what
- Will be open sourced
✅ Notifies the team and services when deploys happen

New Relic

Hipchat (or IRC)

<table>
<thead>
<tr>
<th>GitHub</th>
<th>hannes2000 fast-forward pushed branch master of Gidsy/Gidsy to 380242</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kite</td>
<td>Starting deploy to prod</td>
</tr>
<tr>
<td>Kite</td>
<td>Finished deploy to prod</td>
</tr>
</tbody>
</table>

Jun-28 5:54 PM
Jun-28 5:55 PM
Jun-28 5:57 PM
Provisioning servers

- Follows DRY principle
- Chef/Puppet/Salt
- Documents infrastructure and change
- Place to share and store secure data
- Roles can be on many or one servers
- Challenge is separating deployment from app
Chef configuration

Staging Environment

Chef Role

Staging Environment

Chef Role

Chef Environment

Chef Role

```json
{
  "name": "staging",
  "cookbook_versions": {
    "gidsy_common": "0.0.3"
  },
  "override_attributes": {
    "gidsy": {
      "DISABLE_QUERYSET_CACHE": "False",
      "COMPRESS_ENABLED": "True",
      "EMAIL_HOST_PASSWORD": "*****",
      "HAYSTACK_URL": "http://10.24.15.21:9200/"
    },
    "databases": {
      "gidsy": "host=11.21.17.34 dbname=gidsy",
    },
  }...
  "json_class": "Chef::Environment",
  "chef_type": "environment"
}
```

```json
{
  "run_list": [
    "recipe[ssh_config]",
    "recipe[sudo]",
    "recipe[users::sysadmins]",
    "recipe[pg_bouncer]",
    "recipe[nginx::default]",
    "recipe[gidsy_common]",
    "recipe[gidsy_web]",
    "recipe[pg_bouncer]",
    "recipe[nginx::default]",
    "recipe[gidsy_common]",
    "recipe[gidsy_web]",
    "recipe[hostname]",
    "recipe[papertrail]"
  ],
  "name": "web",
  "json_class": "Chef::Role"
}
```

```json
{
  "run_list": [
    "recipe[ssh_config]",
    "recipe[sudo]",
    "recipe[users::sysadmins]",
    "recipe[pg_bouncer]",
    "recipe[gidsy_common]",
    "recipe[gidsy_web]",
    "recipe[new_relic]",
    "recipe[hostname]",
    "recipe[papertrail]"
  ],
  "name": "celery",
  "json_class": "Chef::Role"
}
```
**Starting a new web server using chef**

```
knife ec2 server create -N staging-web1 -r "role[web]" -G staging-web -E staging -I ami-95dde2e1 -f m1.small -Z eu-west-1a
```

**Running commands on all web servers**

```
k knife ssh "role:web" "sudo chef-client"
```

**Deleting a web server**

```
k knife ec2 server delete i-1234567
```
Working with cloud servers

---

**ec2-ssh**

**before**

`ssh ubuntu@ec2-42-11-12-11.eu-west-1.compute.amazonaws.com -i ~/.ssh/gidsy.pem`

**after**

`ec2-ssh phil@production-web2`

- ✔ Simple syntax
- ✔ Name never changes, unlike url on reboot of server
pychef

- Access node data and manipulate it with python
- Use it in fabric

```python
from fabric.api import env, run, roles
from chef.fabric import chef_roledefs

env.roledefs = chef_roledefs()

@roles('web_app')
def mytask():
    run('uptime')
```
Operations, metrics, maintenance

- Log everything you could need for debugging
- If you deploy often then you need immediate feedback
- Use services if you can (Mixpanel, NewRelic, Librato, Papertrail, Pagerduty)
- Show the metrics on a screen in the office
django-app-metrics

- Push data to the services (mixpanel, librato, log, db)
- App-metrics -> librato or mixpanel
- Logging -> papertrail -> librato
- Extended it to send one metric to many backends
Papertrail log of celery tasks sent

Logs can be graphed by librato
Set alerts on logs that send to pagerduty/hipchat/webhook...
Librato/Graphite

- Logs can be graphed by librato
- Set alerts on logs that send to pagerduty/hipchat/webhook...
Things we learned

To bootstrap your startup quickly

- Only scale when you need to, but be prepared
- Be pragmatic, use the best tool to do the job
- Automate as much as you can
- Continuous Integration and Continuous Deployment
- Make routine tasks as easy as possible
- Use services
- Display metrics
Want to work with us?

Engineering – Web development
Engineering – Backend and Operations

https://gidsy.com/jobs/
Questions?

@gidsynews

http://laboratory.gidsy.com/