Deploying, At An Unusual Scale

Andrew Godwin
@andrewgodwin

http://www.flickr.com/photos/whiskeytango/1431343034/

Hi, I’m Andrew.

- Serial Python developer
- Django core committer
- Co-founder of ep.io

Hi, I’m Andrew.

- Serial Python developer
- Django core committer
- Co-founder of ep.io
- Occasional fast talker

"Andrew speaks English like a machine gun speaks bullets."

Reinout van Rees

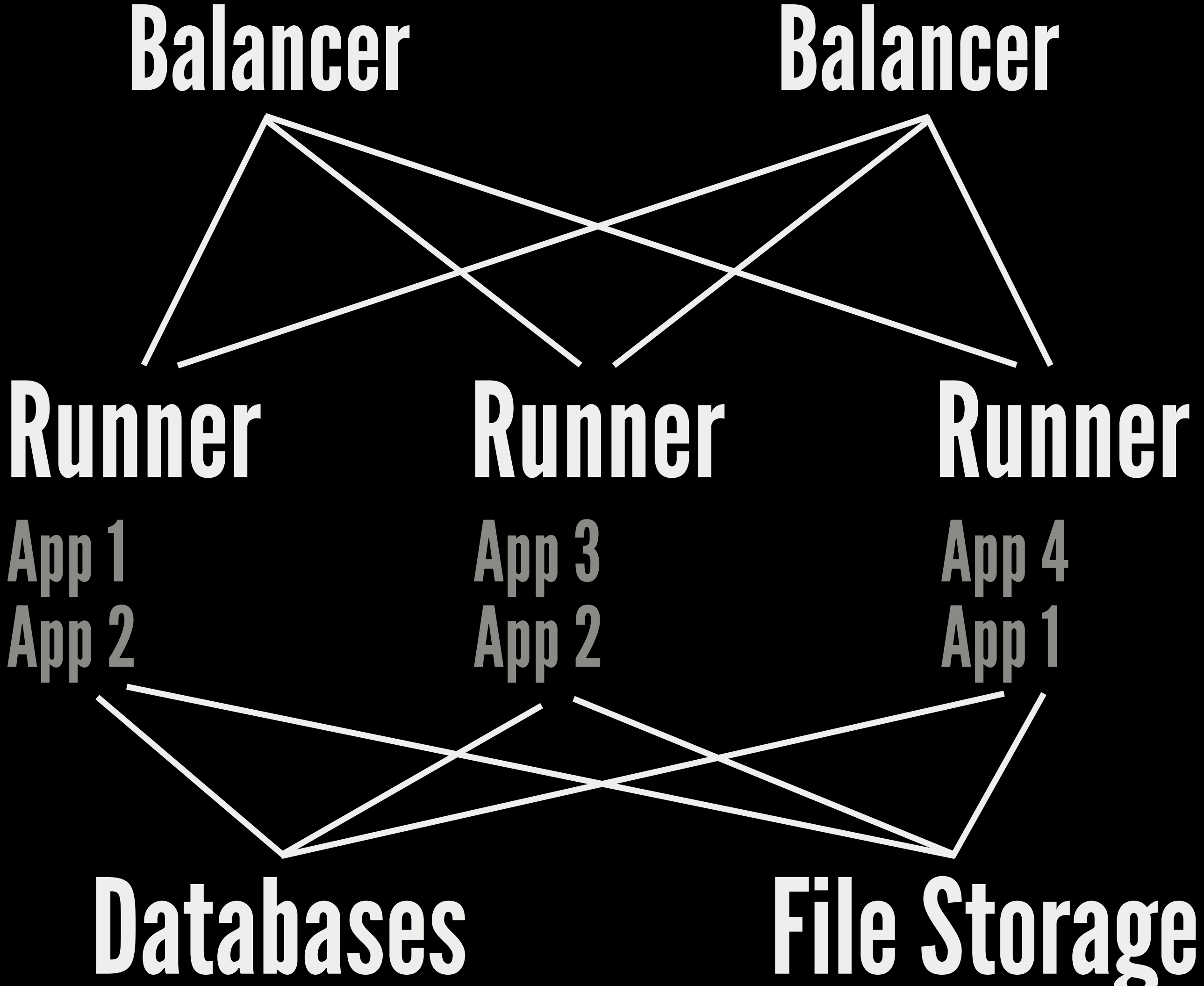
We’re ep.io

- Python Platform-as-a-Service
- Easy deployment, easy upgrades
- PostgreSQL, Redis, Celery, and more

Why am I here?

- Our Architecture
- How we deploy Django
- How varied Django deployments are

Our Architecture


Balancer
Balancer
Runner
Runner
Runner
App 1
App 2
App 3
App 2
App 4
App 1
Databases
File Storage

Oh My God, It’s Full of Pairs

- Everything is redundant
- Distributed programming is Hard

Hardware

- Real colo’d machines (pretty reliable)
- Linode (pretty reliable)
- EC2 (pretty unreliable)

- IPv6 (as much as we can)

ØMQ

- We used to use Redis
- Everything now on ZeroMQ
- Eliminates SPOF*

* Single Point Of Failure. What a pointless acronym.

ØMQ Usage

- Redundant location-resolvers (Nexus)
- REQ/XREP for control messages
- PUSH/PULL for stats, logs
- PUB/SUB for heartbeats, locking

Runners

- Unsurprisingly, these run the code
- SquashFS filesystem images
- Virtualenvs per app
- UID & permission isolation, more coming

Logging/Stats

- All done asynchronously using ØMQ
- Logs to filesystem (chunked files)
- Stats to PostgreSQL database, for now

Loadbalancers

- Intercept all incoming HTTP requests
- Look up hostname (or suffix)
- HTTP 1.1 compliant

Databases

- Shared (only for PostgreSQL)
- Dedicated (uses Runner framework)
- PostgreSQL 9, damnit

Django in the backend

- We use the ORM extensively
- Annoying settings fiddling in __init__

www.ep.io

- Runs on ep.io, just like any other app*
- Provides JSON API, web UI

* Well not quite - App ID 0 is special - but we're working on it

WSGI

- It’s a standard, right?

WSGI

- It’s a standard, right?
- Well, yes, and it works fine, but it’s not enough for serving a Python app

Static Files

- CSS, images, JavaScript, etc.
- Needs a URL and a directory path

Python & Dependencies

- Mostly filled by pip/buildout/etc
- packaging apparently allows version spec

Deploying Django

It makes things consistent, right?

Settings Layouts

- Vanilla settings.py
- local_settings.py
- configs/HOSTNAME.py
- Many others...

Python Paths

- Project-level imports
- App-level imports
- apps/ directories

Databases

- If it’s SQL, it’s PostgreSQL
- Redis for key-value, MongoDB soon
- Some things assume a safe network

HA (High Availability)

- Not terribly easy with shared DBs
- PostgreSQL 9’s sensible warm standby
- Redis has SLAVEOF
- Possibly use DRBD for general solution

Backups

- High Availability is NOT a backup
- btrfs for consistent snapshotting
- Archived remote syncs
- No access to backups from servers

Migrations

- No solution yet for migration/code sync
- We’re working on it…

Web serving

It’s not like it’s important or anything

gunicorn

- Small and lightweight
- Supports long-running requests
- Pretty stable

nginx

- Even more lightweight
- Extremely fast
- Really, really stable

The Load Balancer

- Used to be HAProxy
- Rewritten to custom Python daemon
- eventlet used for high throughput
- Can’t use nginx, no HTTP 1.1 for backends

Celery

- See: Yesterday’s Talk
- Slightly tricky to run many
- We use Redis as the backend

Management Commands

- First off, run as subprocess
- Then, a custom PTY module
- Now, run as pty-wrapping subprocesses

Some General Advice

If you’re crazy enough to do this

Messaging’s Not Enough

Having a state to check is handy

Why run one, when you can run two for twice the price?

Redundancy is good. Double redundancy is better.

Always expect the worst

Hope you never have to deal with it.

The more backups, the better.

Make sure you have historical ones, too.

Django is very flexible

Sometimes a little too flexible…

Your real problems will emerge later

Don’t over-optimise up front for everything

Questions?

Andrew Godwin
andrew@ep.io
@andrewgodwin