Python @ Layar

or: building complex and scalable systems using Python and AWS

Jens de Smit
@jfdsmit

Layar...

...does mobile augmented reality

...is a startup based in Amsterdam, NL

...offers two mobile apps (on one backend)

...uses a lot of Python in the backend :D

The Layar app

Terminator vision for Android and iOS

Have your phone recognize where you are or what it is you are looking at and overlay extra information

http://layar.com

Stiktu

Digital graffiti in augmented reality

Take a picture of something and let loose your creativity

Share with the world, like, comment

http://stiktu.com

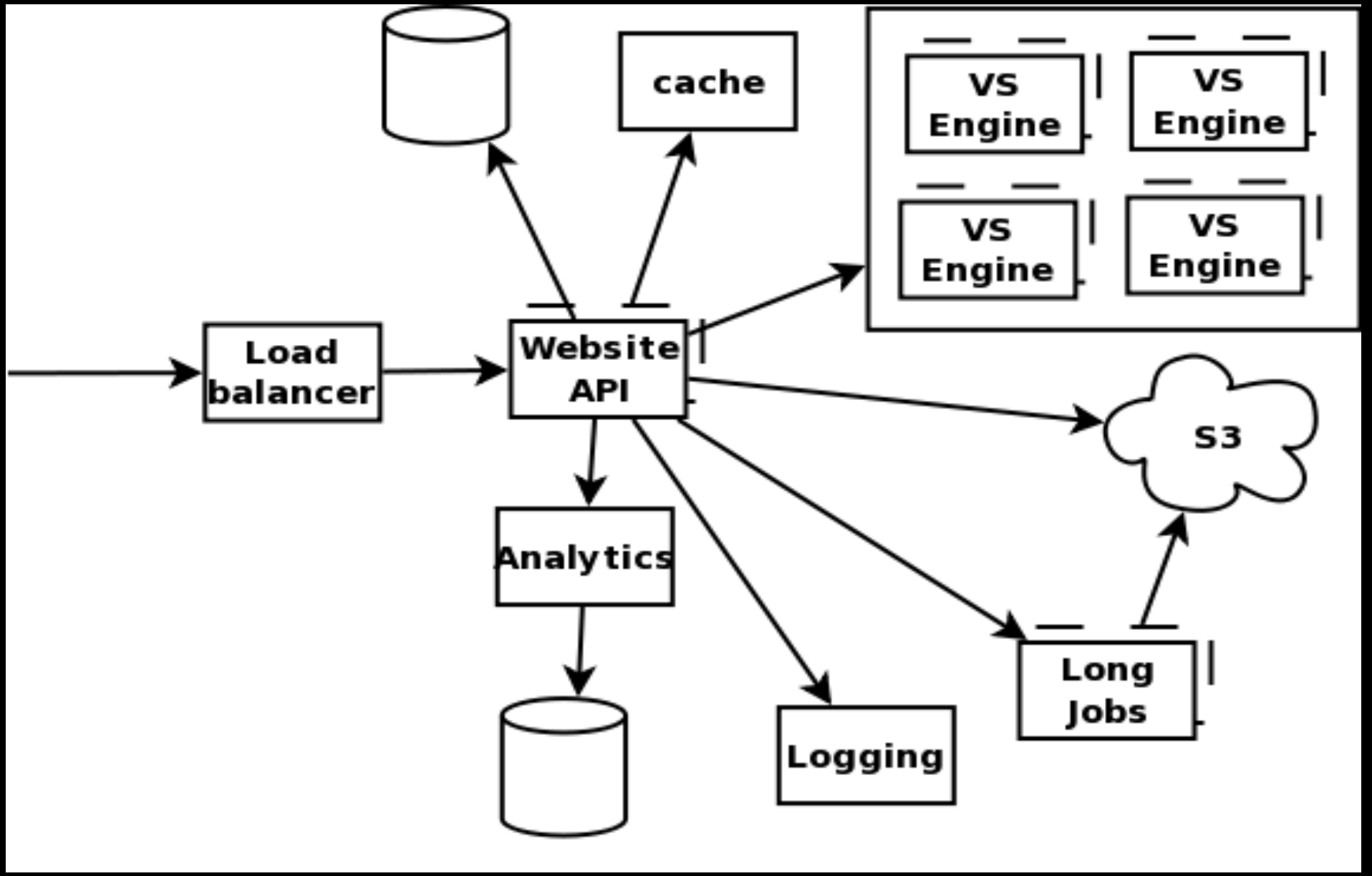
Two apps, one backend

Same core technology

Radically different service models
Layar is an open platform, content is provided (and can be hosted) by third parties

Stiktu is a closed service, content is generated by users

The server side of life



The web-facing side

Django: generally a good idea
...comprehensive feature set
...build web pages and APIs
...active development community
...many good extensions
...can handle high volumes as long as you listen to Christophe Pettus (thebuild.com)

Handles user registration, content catalog, web presence, hosting and delivering (part of) the content

Files are stored on S3, database is MySQL on Amazon RDS

The web-facing side

Default 2 Django instances with AWS load balancer

Django instances autoscale when load goes up

Popular data is cached in memcached

Scaling database: bigger machine or read replicas

Logging

Sentry: centralized logging

Group and count similar messages

One Sentry install for all your services

Many thanks @zeeg from DISQUS

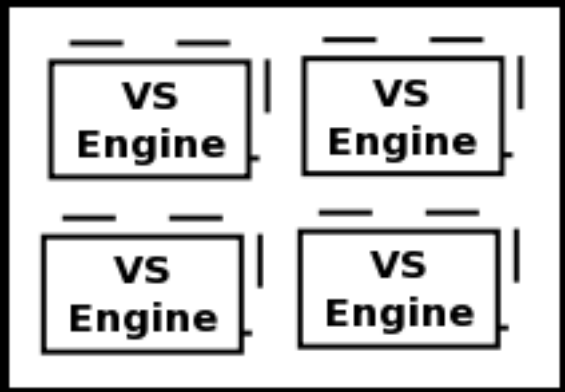
https://github.com/dcramer/sentry or http://getsentry.com

Visual Search Engine

Image recognition tech from Catchoom (Telefonica spin-off)

Tornado with Boost.Python interfacing to C++ binaries

Sharded for scale-out, redundant for HA and read speed



Storage on EBS volumes

Analytics

MySQL database collects data points

Django app stores SQL queries for aggregation

Cron job executes queries hourly/daily/weekly and stores results in designated table

Yet more SQL queries feed Highcharts for fancy graphics

This does NOT scale

Long-running jobs: Spencer

Extracting images from PDFs, analyzing images for client-side image recognition

1. get a job ticket from SQS (job type, job arguments, callback URL)
2. start the right worker for the job type
3. worker launches a separate process (usually a binary) to do the hard work
4. kick back, relax, get back to main Twisted loop
5. when process completes, store results in S3 and call the callback URL

About Spencer

Multiprocessing instead of multithreading makes it easy to use all cores

Default 1 instance, easily scales to 20

Calling separate programs to do the processing lets you use basically anything

Only 1300 lines on top of Twisted

Still very alpha and AWS-specific, but...

So, about AWS...

Convenient services

Easy to add capacity, pay for what you use

Basic monitoring out of the box

Web interface (but not for everything)

Fully automatable through CLI

So, about AWS...

Lots of marketing text

Not the most bang for your buck

Watch your wallet: clean up

Assume no guarantees

Does NOT excuse you from having Ops

Assorted tips and tricks

Right tools, right jobs: Python has a lot to offer

Automate deployment: Fabric, Chef

Deploy early, darktest (waffle, gargoyle)

In Django, offload anything that looks like work
...but use django-ztask, not celery

Cache from the beginning, don't "come back to it" because it works now

Bedankt!

@jfdsmit