Advanced Django Architecture

Rami Sayar - @ramisayar
Technical Evangelist
Microsoft Canada

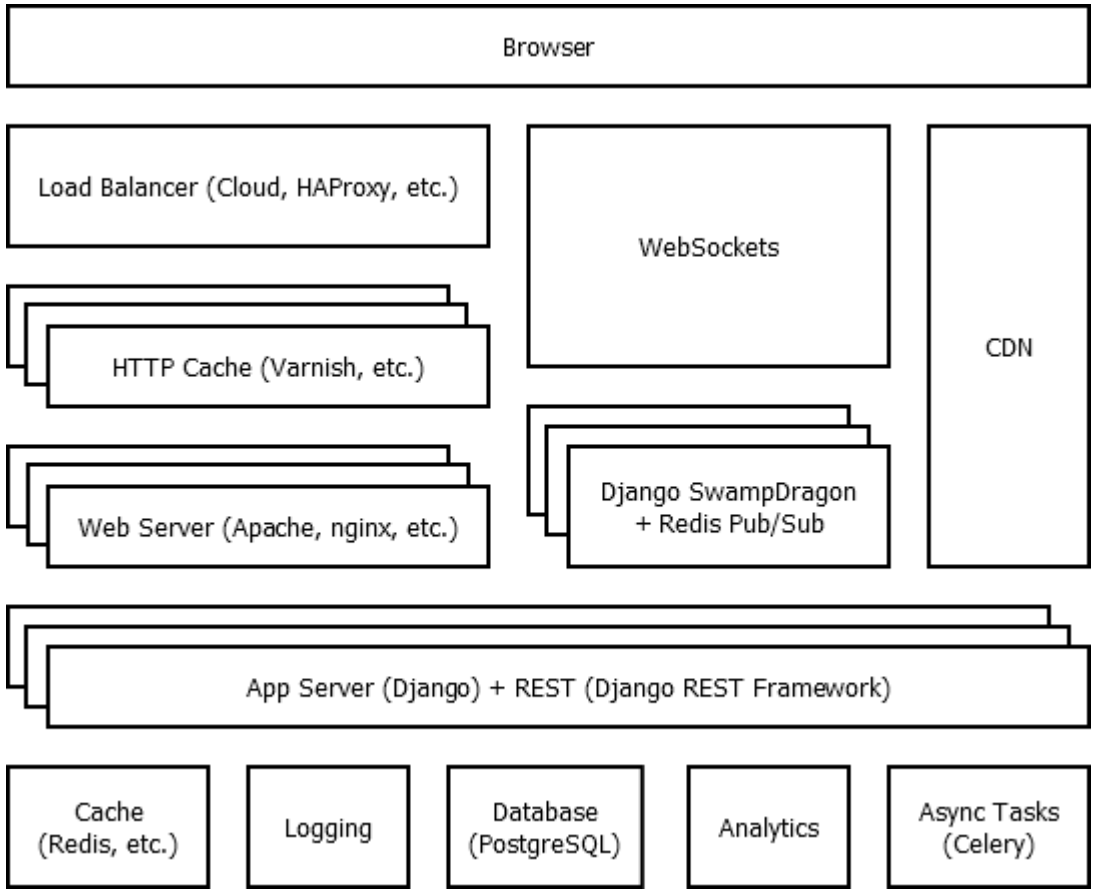
~~Advanced~~ Scalable Django Architecture

Rami Sayar - @ramisayar
Technical Evangelist
Microsoft Canada

Agenda

- Advanced Django Techniques & Patterns
- Structuring Complex Applications
- Scaling Django Applications
- Django Applications in Production

What is a production web app?



- Django is surrounded by tons of applications, services and code in production.
- Advanced Django architecture is needed to support Django in production without failing hard.

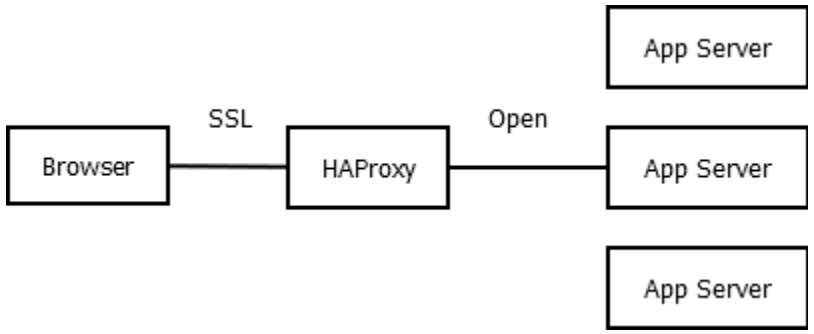
Load Balancing Django

Load Balancing

- Load balancing Django improves the performance and reliability by distributing traffic.
- Becomes an interesting problem when you have to consider:
  - SSL Termination
  - Sticky Sessions?
  - WebSockets?

Load Balancing = Django + HAProxy + SSL

- HAProxy gained native SSL support in HAProxy 1.5.x => June 2014!
- SSL Termination:

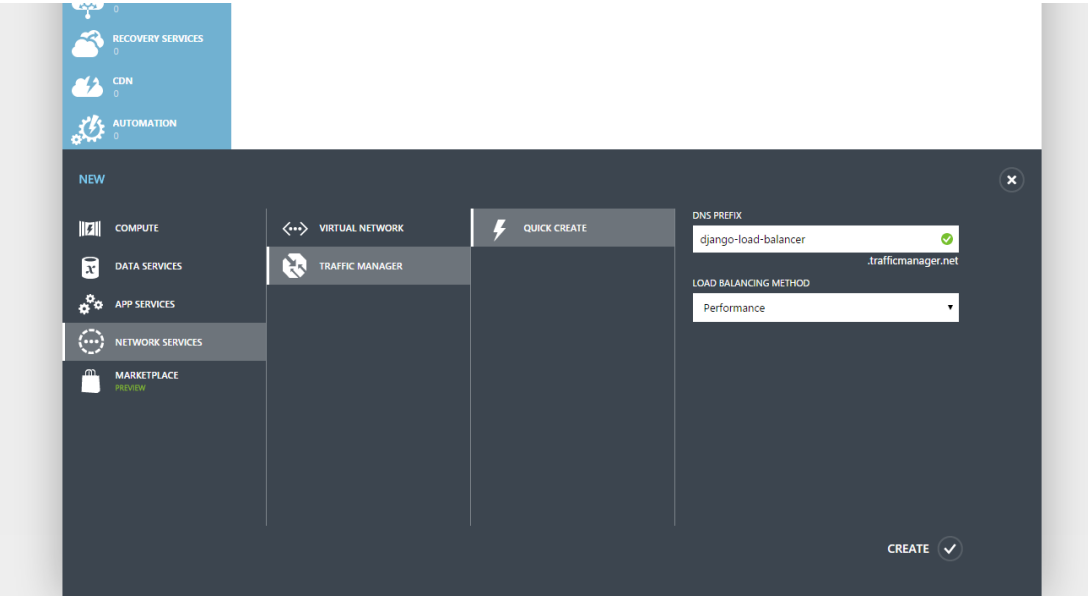


- Fairly straightforward to set up HAProxy.
  - Read: How To Implement SSL Termination With HAProxy

Load Balancing Proxying & Remote Addr

```
class SetRemoteAddrFromForwardedFor(object):
    def process_request(self, request):
        try:
            real_ip = request.META['HTTP_X_FORWARDED_FOR']
        except KeyError:
            pass
        else:
            # HTTP_X_FORWARDED_FOR can be a comma-separated list of IPs.
            # Take just the first one.
            real_ip = real_ip.split(",")[0]
            request.META['REMOTE_ADDR'] = real_ip
```

Azure Traffic Manager



- High Performance Load Balancer + DNS Support.
- Sticky Sessions don’t exist.
- HTTPS Supported.

Caching Django

Caching = Django + Varnish

- Varnish has a tendency to just cache EVERYTHING which can be frustrating for the User, unless you set it up properly.
- Varnish does not cache content & cookies.
- Varnish and Django CSRF protection gotcha.
- Varnish does not deal with HTTPS (hence the need of HAProxy in front of Varnish).

Caching = Django + Varnish

- Django-Varnish – “It allows you to monitor certain models and when they are updated, Django Varnish will purge the model's absolute_url on your frontend(s). This ensures that object detail pages are served blazingly fast and are always up to date.”
- https://github.com/justquick/django-varnish

- Project Idea: Deeper Django/Varnish integration.

Caching = Django + Varnish

- https://www.varnish-cache.org/
- http://www.isvarnishworking.com/
- http://yml-blog.blogspot.ca/2010/01/esi-using-varnish-and-django.html
- http://chase-seibert.github.io/blog/2011/09/23/varnish-caching-for-unauthenticated-django-views.html
- http://blog.bigdinosaur.org/adventures-in-varnish/
- http://www.nedproductions.biz/wiki/a-perfected-varnish-reverse-caching-proxy-vcl-script

Serving Django

Web Server = Django + Apache

- Forced to use Apache? Your options:
  - mod_python (No)
  - mod_wsgi (Yes)
  - FastCGI (Deprecated since 1.7)
  - Phusion Passenger (Yes)

Web Server = Django + Apache

- mod_wsgi
  - Decent performance on the Apache side.
  - You can setup Apache any way you like.
  - You can setup mod_wsgi to use virtualenv.

Web Server = Django + Apache

```
import os, sys, site

# Add the site-packages of the chosen virtualenv to work with
site.addsitedir('~/.virtualenvs/myprojectenv/local/lib/python2.7/site-packages')

# Add the app's directory to the PYTHONPATH
sys.path.append('/home/django_projects/MyProject')
sys.path.append('/home/django_projects/MyProject/myproject')

os.environ['DJANGO_SETTINGS_MODULE'] = 'myproject.settings'

# Activate your virtual env
activate_env=os.path.expanduser("~/.virtualenvs/myprojectenv/bin/activate_this.py")
execfile(activate_env, dict(__file__=activate_env))

import django.core.handlers.wsgi
application = django.core.handlers.wsgi.WSGIHandler()
```

Web Server = Django + Alternatives (aka. Nginx)

- Nginx + uWSGI

```
[uwsgi]
touch-reload = /tmp/newproject
socket = 127.0.0.1:3031
workers = 2
chdir = /srv/newproject
env = DJANGO_SETTINGS_MODULE=newproject.settings
module = django.core.handlers.wsgi:WSGIHander()
```

Web Server = Django + Alternatives (aka. Gunicorn)

- Nginx & Gunicorn
  - In theory, you should use Nginx as a reverse proxy here, to serve static files if you're not uploading everything to a CDN or a separate media server. If you are, you can use HAProxy and Gunicorn as a standalone webserver, one less dependency to care about.
  - Install Gunicorn directly inside your virtual environment.
  - Use Gaffer to monitor the Gunicorn.

Web Server - References

- http://blog.kgriffs.com/2012/12/18/uwsgi-vs-gunicorn-vs-node-benchmarks.html

REST APIs & Django

Use Django REST Framework!

Django REST Framework vs Django Tasty Pie

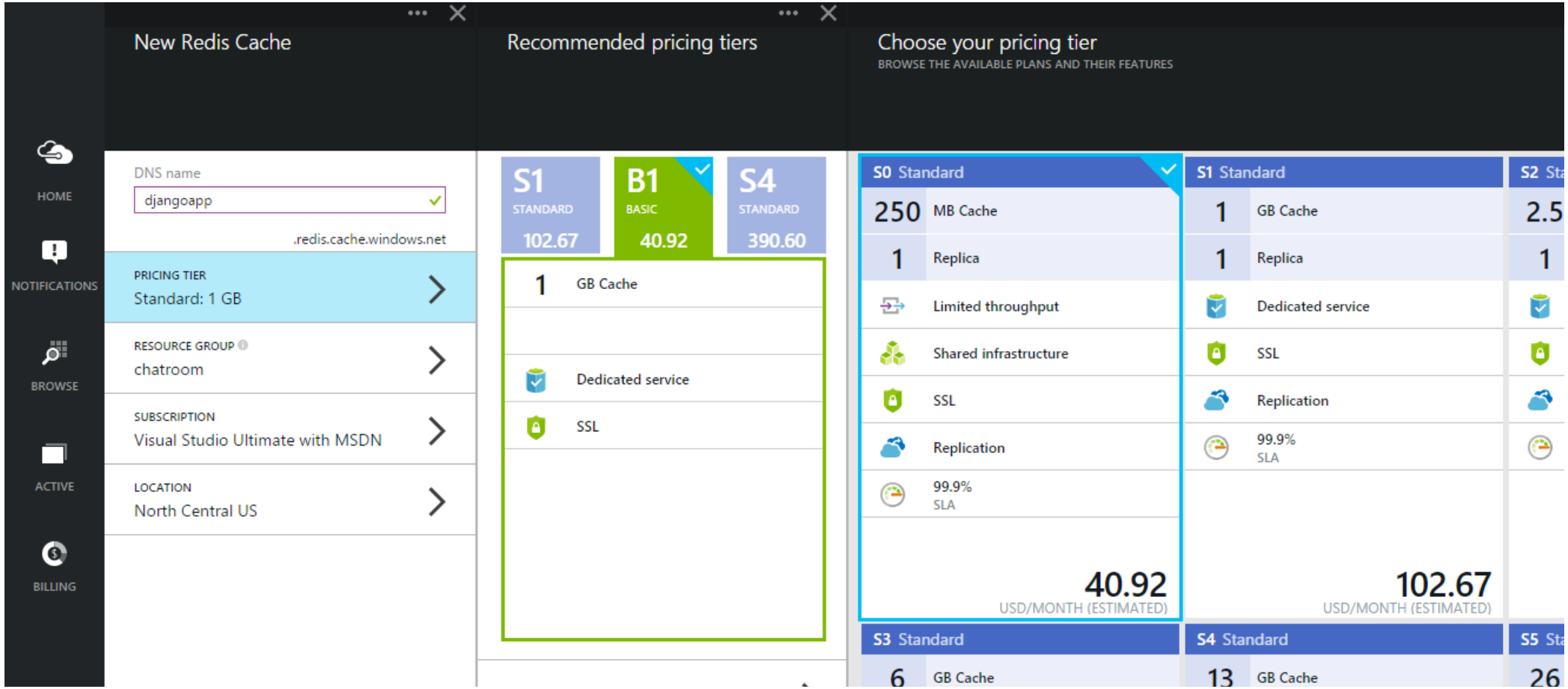
Cache & Django

Caching in Production

- Caching – Django has multiple options:
  - **Redis.** Make sure you use Hiredis and django-redis-cache. Use a hosted Redis Cache (Azure, AWS, etc).
  - **Memcached.** Make sure you use pylibmc and not python-memcached, better performance with the C library.

- SSL still matters.

Azure Redis Cache



Logging Django

Logging in Production

- Logstash
  - Collect logs, parse them, and store them for later use.
  - Free and open source.

  - Set it up with Redis.

Logging in Production – Use Python-Logstash

```
LOGGING = {
  'handlers': {
      'logstash': {
          'level': 'DEBUG',
          'class': 'logstash.LogstashHandler',
          'host': 'localhost',
          'port': 5959, # Default value: 5959
          'version': 1, # Version of logstash event schema. Default value: 0 (for backward compatibility of the library)
          'message_type': 'logstash',  # 'type' field in logstash message. Default value: 'logstash'.
          'tags': ['tag1', 'tag2'], # list of tags. Default: None.
      },
  },
```

Logging in Production – Use Python-Logstash

```
  'loggers': {
      'django.request': {
          'handlers': ['logstash'],
          'level': 'DEBUG',
          'propagate': True,
      },
  },
}
```

Logging in Production – Using Sentry

- Realtime event logging and aggregation platform.
- Specialize in monitoring exceptions and errors.

- https://github.com/getsentry/sentry

PostgreSQL in Production

PostgreSQL in Production – pgpool & slony

- PostgreSQL 9+ has streaming replication assuming you're running identical databases with identical version on identical architectures (not really an issue).
- If not using the Streaming Replication feature (WAL), you'll want to use either pgpool or slony to scale your PostgreSQL database across multiple machines.
  - Use Slony if you want master-slave-peer replication.
  - Use pgpool or pgbouncer for connection pooling. Pgbouncer only does connection pooling.

Async Tasks & Django

Async Tasks – Celery

- Use Celery for Asynchronous Tasks
- Use Celery with Redis and store results in Django ORM

- RQ is an alternative for distributing tasks. (Based on Redis)

Read:
http://www.caktusgroup.com/blog/2014/09/29/celery-production/

On Redis

- Redis is extremely popular as an easy and simple publish-subscribe.
- Redis is very particular about memory. Running out of memory in Redis -> **CRASH AND BURN**!
- If you don’t have enough memory in your VM to run Redis Cache -> Change your approach, use Apache Kafka.

Tips & Techniques

- Don’t forget:
  - Turn off Debug Mode
  - Turn off Template Debug Mode

- Use different Django Settings Modules depending on the role and environment you are in through environment variables by setting “DJANGO_SETTINGS_MODULE”.

What did we learn?

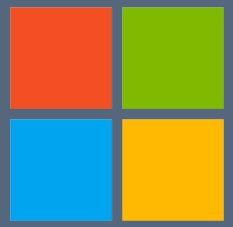
- Scaling Django Architectures
- Most of the work surrounds Django
- Minimal changes to your django app for the most part.

Thank You! Questions?

Follow @ramisayar

Resources

- http://blog.disqus.com/post/62187806135/scaling-django-to-8-billion-page-views
- http://engineering.hackerearth.com/2013/10/07/scaling-database-with-django-and-haproxy/
- http://www.djangobook.com/en/2.0/chapter12.html
- https://www.digitalocean.com/community/tutorials/how-to-implement-ssl-termination-with-haproxy-on-ubuntu-14-04
- https://docs.djangoproject.com/en/1.7/topics/security/
- https://www.digitalocean.com/community/tutorials/how-to-scale-django-beyond-the-basics



Microsoft