Django: The Python Web Framework

by Adam Cardenas

CS5448 Object Oriented Analysis and Design

whoami

- Adam Cardenas
- Computer Science Grad Student
  - Colorado Univ. Boulder
- Worked at a Django startup called Bixly
  - http://bixly.com
  - It was awesome

Bet Tracking Examples

I will be using examples from my semester project.

It is a mobile application for tracking bets. Users can create, challenge bets and wage an arbitrary amount of virtual currency. Depending on the outcome, the virtual currency is dispensed to the winner of the bet.

whois Django

- Founders
  - Simon Willison
  - Adrian Holovaty
- Created for the a newspaper site, World Company, as a Content Management System
- Named after Adrian's favorite guitarist, Django Reinhardt
- Open sourced web framework

Django Object Relational Mapper

- Database abstraction which allows Python objects to orchestrate SQL features such as create, retrieve, update and delete.
- Supports the following databases
  - PostgreSQL
  - MySQL
  - SQLite
  - Oracle

Django Object Relational Mapper (cont.)

Pros

- OO objects are used to generate query statements
- Reduces code
- Increases maintainability

Con

- Can lead to a poorly designed database

Django Object Relational Mapper (cont.)

Models

```
class BetState(models.Model):
    name = models.CharField(max_length=50)


class Bet(models.Model):
    title = models.CharField(max_length=50)
    description = models.CharField(max_length=500)
    wager = models.IntegerField(default=10)

    created = models.DateTimeField(auto_now_add=True)
    creator = models.ForeignKey(User)
    requestee = models.ForeignKey(User, related_name='bet_requestee')

    state = models.ForeignKey(BetState)

    winner = models.ForeignKey(User, blank=True, null=True, related_name='bet_winner')
    looser = models.ForeignKey(User, blank=True, null=True, related_name='bet_loser')
```

SQL

```
BEGIN;
CREATE TABLE "bet_betstate" (
    "id" integer NOT NULL PRIMARY KEY,
    "name" varchar(50) NOT NULL
)
;
CREATE TABLE "bet_bet" (
    "id" integer NOT NULL PRIMARY KEY,
    "title" varchar(50) NOT NULL,
    "description" varchar(500) NOT NULL,
    "wager" integer NOT NULL,
    "creator_id" integer NOT NULL REFERENCES "auth_user" ("id"),
    "requestee_id" integer NOT NULL REFERENCES "auth_user" ("id"),
    "state_id" integer NOT NULL REFERENCES "bet_betstate" ("id"),
    "winner_id" integer REFERENCES "auth_user" ("id"),
    "looser_id" integer REFERENCES "auth_user" ("id")
)
;
COMMIT;
```

Django Object Relational Mapper (cont.)

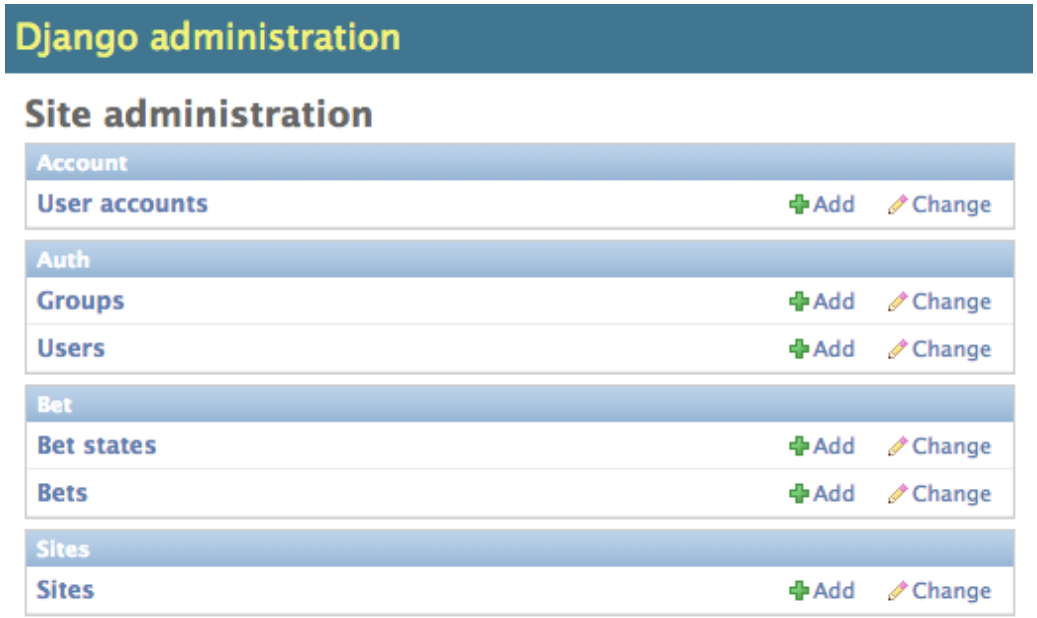
```
# Create a bet, then save to database
bet = Bet(
      title='Breath Contest',
      description='Person who can hold their breath the longest.',
      wager = 10,
)
bet.save()

# Get Bet instance having the primary key of 3
# if object not found, redirect to 404 page
bet = get_object_or_404(Bet, pk=3)

# Get all bets in database with containing a wager of 10
bets = Bet.objects.filter(wager=10)

# Get all bets that are in "Completed" state, sort them by creation timestamp
bets = Bet.objects.filter(state=BetState.objects.get(name='Completed')).order_by('created')
```

Django Admin Pages

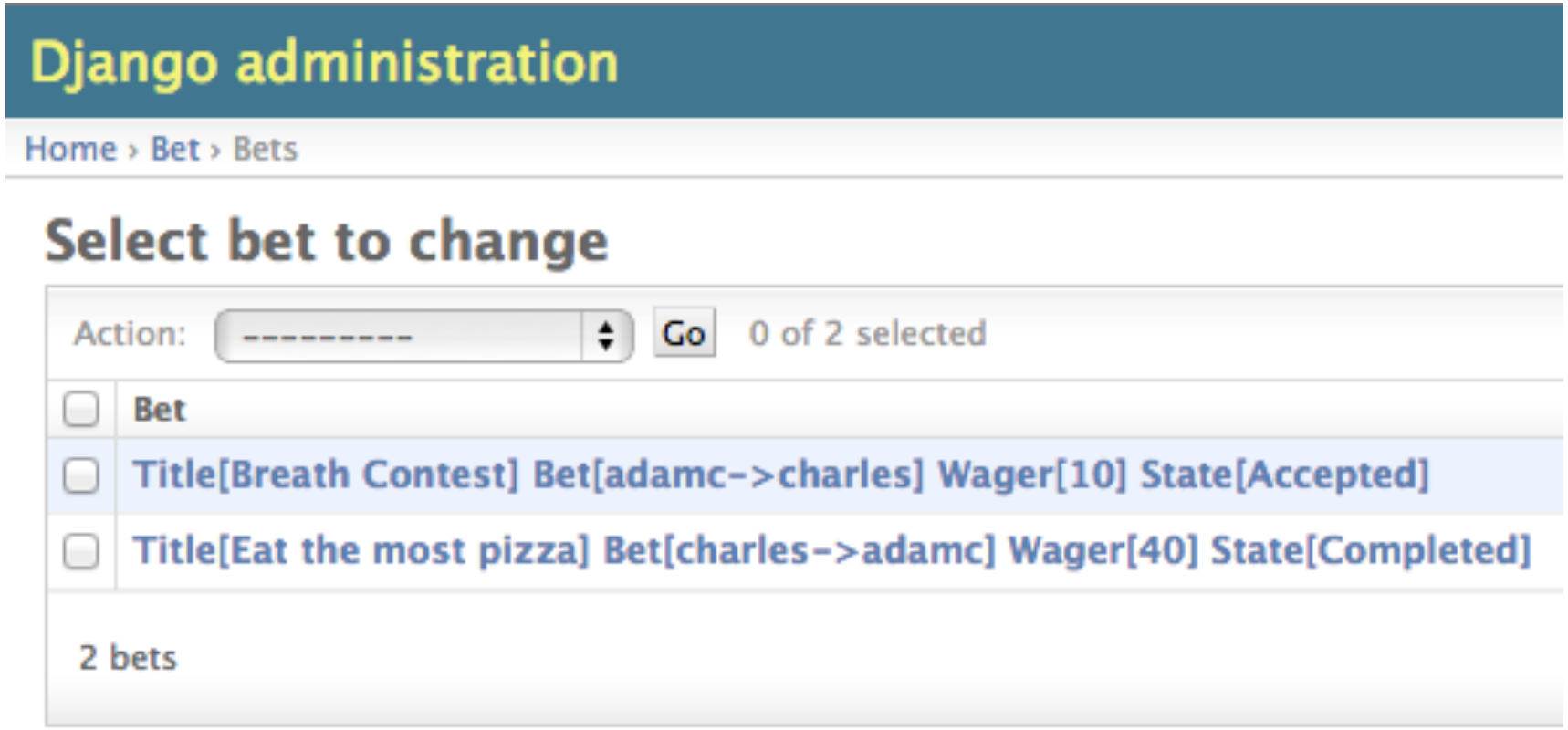
Reads metadata in from your models to provide a powerful and production-ready interface so that data can immediately and easily be added to the site

Django Admin Pages (cont.)

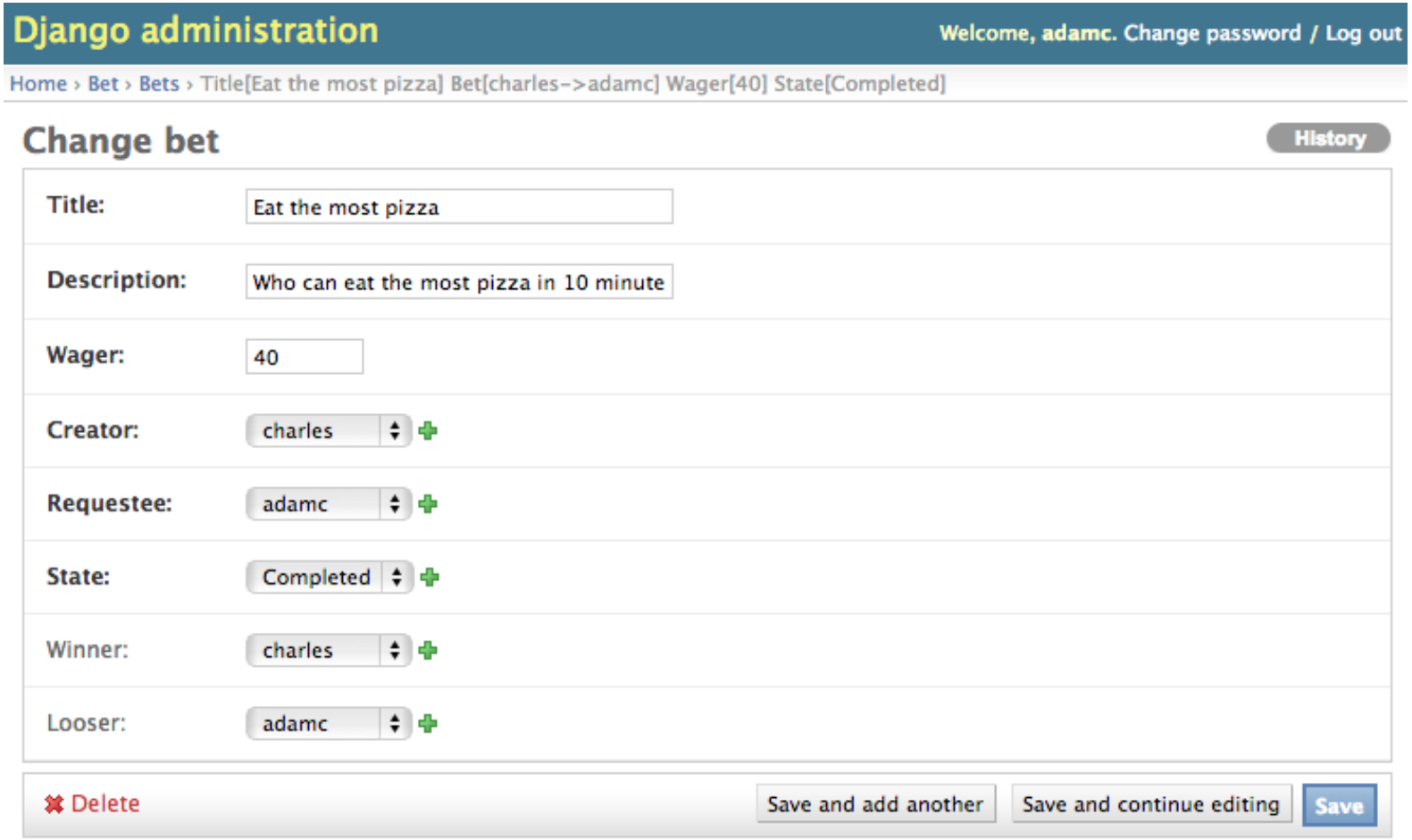
```
from django.contrib import admin

from bet.models import Bet
from bet.models import BetState

admin.site.register(Bet)
admin.site.register(BetState)
```



Django Admin Pages (cont.)



Django Admin Pages (cont.)



Django Template Language

- By design, the template a Python/HTML script (not PHP)
- A template contains Python variables, which get replaced with values when the template is processed, and tags (if/for/custom), which control the logic of the template.
- Created for designers, not developers

Django Template Language (cont.)

```
<!DOCTYPE html>
<html>
    <head>
        <script src="{{STATIC_URL}}js/project.js" type="application/javascript"></script>
        <link rel="stylesheet" href="{{STATIC_URL}}css/style.css" type="text/css"/>

        <title>{% block title %}{% endblock %}</title>
        {% block head %}{% endblock %}
    </head>
    <body>
        <!-- content -->
        <div id="container">
            {% block content %}{% endblock %}
        </div>
    </body>
</html>
```

Master Template

```
{% extends "site_base.html" %}

{% block title %}{{user.first_name|title}} {{user.last_name|title}}{% endblock %}
{% block content %}

<h1>Hello World<h1>

<!-- Foreach Example -->
{% for bet in bets %}
    <h3>{{bet.winner}}</h3>

    <!-- If Condition Example -->
    {% if bet.wager < 100 %}
        <p>This is a weak bet!!</p>
    {% endif %}

{% endfor %}

{% endblock %}
```

Extending Template

Django Project Layout

- static/
- templates/
- betApp/
  - templates/
  - models.py
  - forms.py
  - urls.py
  - views.py
- settings.py
- urls.py
- manage.py
- wsgi.py

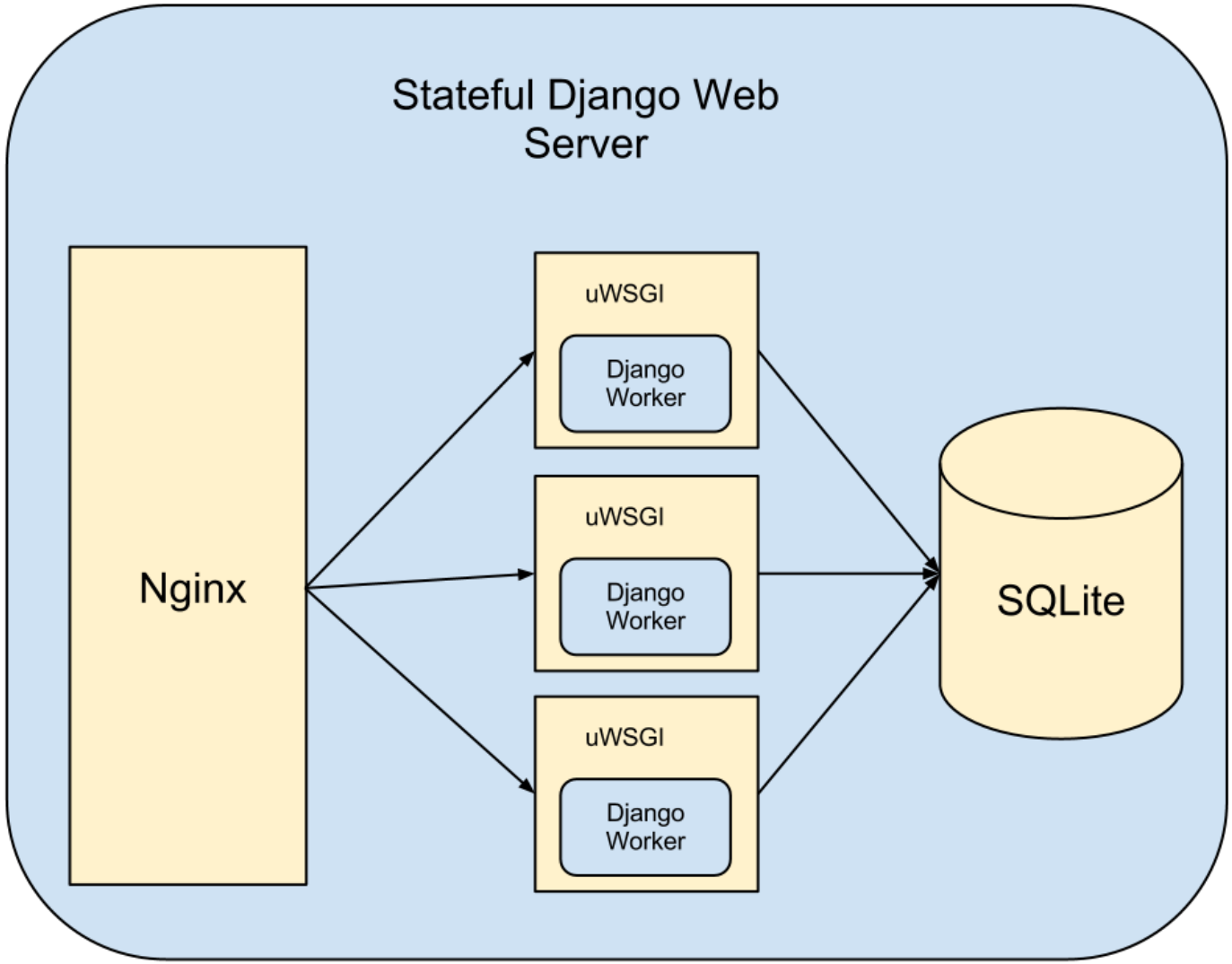
Notable 3rd Party Applications

- South
- Social Auth
- Boto

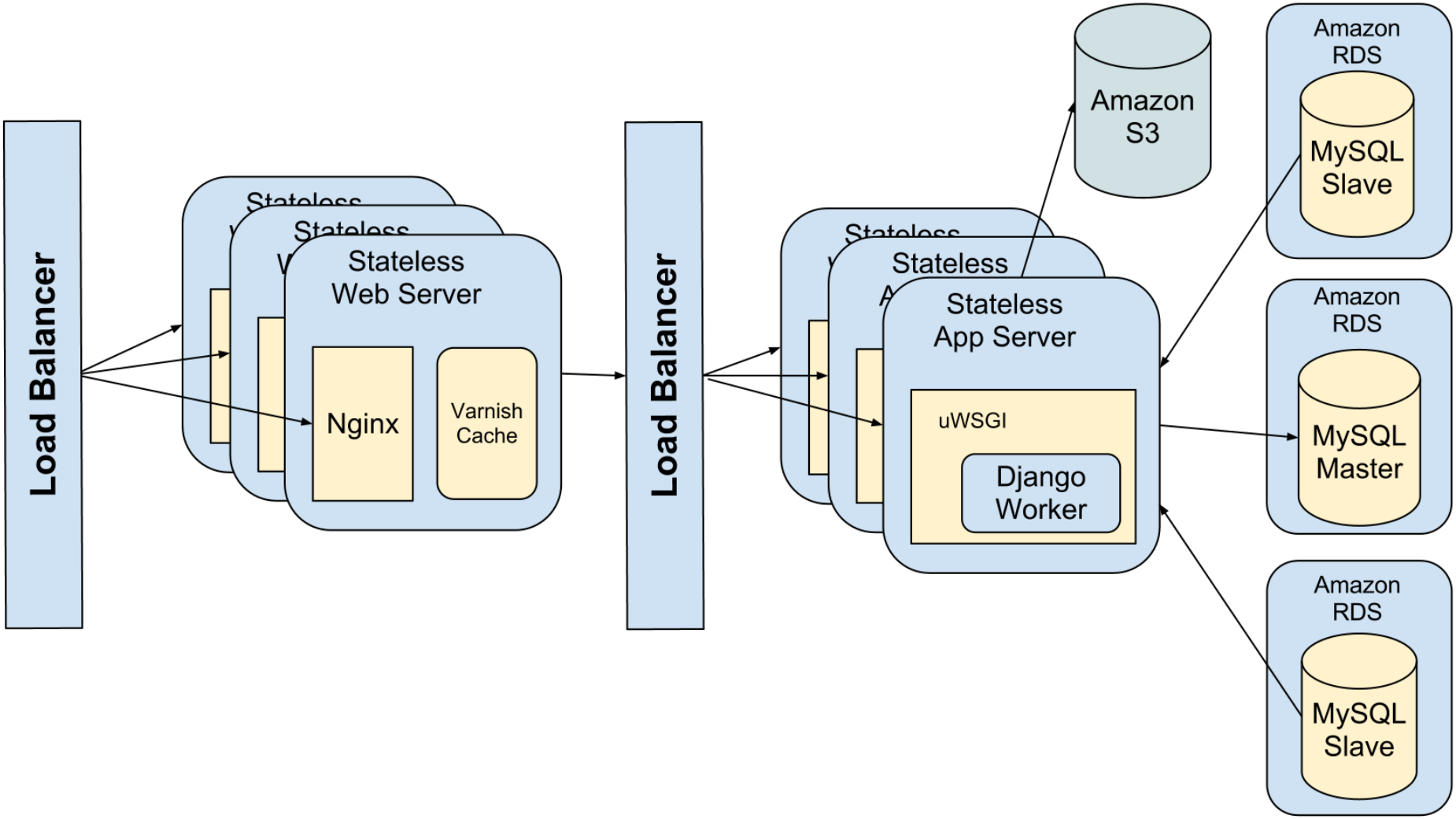
Django Deployment Strategies

- Nginx
- uWSGI
- VirtualEnv
- pip installer
- Varnish/memcached
- MySQL/SQLite

Basic Architecture



Scalable Web Hosting Architecture



Not to Mention...

- Django Model Forms and Validation
- Django Middleware
- Django Authentication
- Django Cross Site Request Forgery Protection
- Django Signals
- Django Caching
- Django Internationalization
- Django URL design
- ... and more

References

- https://docs.djangoproject.com/
- http://www.quora.com/What-is-the-history-of-the-Django-web-framework
- http://aws.amazon.com/architecture/