

Deploy a Serverless WSGI App using Zappa, CloudFront, RDS, and VPC

Brian Jinwright, Senior Software Engineer
MetaMetrics Inc.
[@brianjinwright](#)

Brian Jinwright (aka Young Python Stacks)

- Python as a primary language since 2007
- AWS since 2007
- I work at MetaMetrics Inc in an attempt to build cool tools for your kids and educators. **We are hiring!!**
- I am a Zappa contributor but I'm no way the best authority on this subject.
- Join the Zappa Slack community (<https://slack.zappa.io/>)
- Read the Zappa docs (<https://github.com/miserlou/zappa>)
- I'm not brave. There will not be any live coding or live deployments.
- More details?

<https://jinwright.net/how-deploy-serverless-wsgi-app-using-zappa/>

What is Serverless?

TLDR; Refers to using a Function as a Service (FaaS) like AWS' Lambda offering to handle the bulk of the compute needs for your application. Also, it can be described as using a FaaS in conjunction with other services like API Gateway, SNS, and S3.

Sexy Advantages:

- 100% Utilization
- Highly Scalable
- Easily Deployable
- More affordable to deploy multi-region applications

What is WSGI?

WSGI is the Web Server Gateway Interface. It is a Python specification that describes how a web server communicates with web applications, and how web applications can be chained together to process one request.

WSGI is a Python standard described in detail in PEP 3333.

Let's Get Started

1. AWS Steps
2. Code and Local Environment
3. Zappa Steps
4. CloudFront Steps (Optional)



AWS Steps

These steps are AWS focused and I am only adding them to the slides so you know what happens. I recommend that you write a CloudFormation template (use Troposphere) to automate these tasks for future deployments.

Troposphere (<https://github.com/cloudtools/troposphere>)

1. Create ~~Two~~ **Four** S3 Buckets (in different regions)

You should only have to complete this stage once even if you create a CloudFormation template in the future. We are creating one bucket for Zappa to store our code when deploying and another to store static files (css, images, and js).

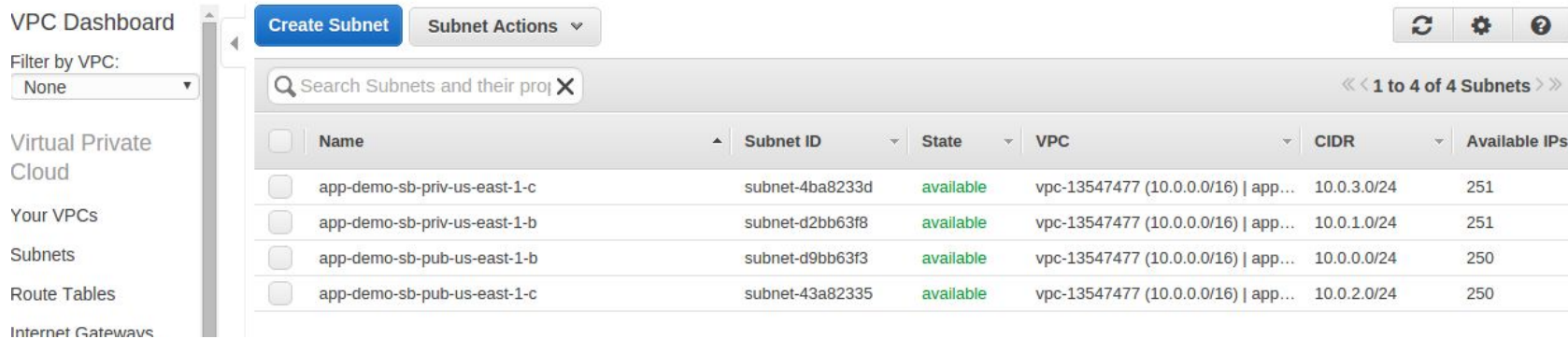
Note: *By default Zappa will only store your code in the function bucket briefly while deploying (zappa deploy) or updating (zappa update) the Lambda function*

2. Configure your VPC

Maybe your app doesn't need a VPC. I personally believe if you are running a database that uses RDS and ElastiCache you need a VPC. Next are a list of things I recommend to get a Zappa powered app working with VPC, RDS, and Elasticache.

2a. Create Public and Private Subnets

Create at least two public and private subnets in different availability zones.



The screenshot displays the AWS VPC console interface. On the left, a navigation sidebar includes 'VPC Dashboard', 'Filter by VPC: None', 'Virtual Private Cloud', 'Your VPCs', 'Subnets', 'Route Tables', and 'Internet Gateways'. The main content area features a 'Create Subnet' button and a 'Subnet Actions' dropdown. Below these is a search bar with the text 'Search Subnets and their proj X' and a pagination control showing '1 to 4 of 4 Subnets'. A table lists four subnets, all in an 'available' state.

<input type="checkbox"/>	Name	Subnet ID	State	VPC	CIDR	Available IPs
<input type="checkbox"/>	app-demo-sb-priv-us-east-1-c	subnet-4ba8233d	available	vpc-13547477 (10.0.0.0/16) app...	10.0.3.0/24	251
<input type="checkbox"/>	app-demo-sb-priv-us-east-1-b	subnet-d2bb63f8	available	vpc-13547477 (10.0.0.0/16) app...	10.0.1.0/24	251
<input type="checkbox"/>	app-demo-sb-pub-us-east-1-b	subnet-d9bb63f3	available	vpc-13547477 (10.0.0.0/16) app...	10.0.0.0/24	250
<input type="checkbox"/>	app-demo-sb-pub-us-east-1-c	subnet-43a82335	available	vpc-13547477 (10.0.0.0/16) app...	10.0.2.0/24	250

2b. Create a NAT Gateway

This NAT Gateway will enable your Lambda functions in a private subnet to connect to the Internet or other AWS services, but prevent the Internet from initiating a connection with those instances. Associate your NAT Gateway with one of your private subnets. This is needed if you are planning on using any public API (MailChimp, Twilio, etc.) other than S3's.

Create a NAT Gateway ✕

Create a NAT gateway and assign it an Elastic IP address. [Learn more](#)

Subnet* ⓘ

Elastic IP Allocation ID* ⓘ

Allocation ID	Elastic IP Address
eipalloc-63e93806	52.62.111.235
eipalloc-6ae9380f	52.62.113.202

2c. Route Table

Route tables will control the routing for your subnets. You could setup a different route table for each subnet but for the purposes of this guide we're setting one for the private subnets and one for the public ones. Afterwards you should associate our subnets we will route 0.0.0.0/0 traffic to our NAT Gateway for our private subnets and 0.0.0.0/0 to our Internet Gateway for our public subnets.

2d. Create a VPC Endpoint for S3

This is needed so your Lambda function can communicate with S3's API. This comes in handy later when you run **zappa manage "collectstatic --noinput"**.

2e. Create a Custom IAM Policy

By default, the Zappa client will create and manage the necessary IAM policies and roles to execute Zappa applications. However, if you're using Zappa in a corporate environment or as part of a continuous integration, you may instead want to manually manage your remote execution policies instead.

Note: PLEASE DO THIS!!!!!!

3. Create a MySQL, Aurora, or MariaDB instance(s)

Create a MySQL instance(s) and associate it with the VPC and subnets you created in the previous step.



4. Create Redis Cluster with ElastiCache

Create a Redis instance(s) and associate it with the VPC and subnets you created in step 2.



Still There?

Code & Local Environment Steps

You got it!! These are the steps that will require a change in your code or your local environment. Mainly the stuff you normally do when you start a new Django project.

1. Install virtualenv, virtualenvwrapper, and eva globally

This step is only useful if you plan on working on this project locally without something like Docker or Vagrant. If you just install eva it will install the other two.

```
sudo pip install eva
```

2. Install zappa, envs, django, django-storages, etc.

```
pip install zappa envs django mysql-python django-storages boto django-redis
```

3. Create a project

```
django-admin startproject yourblog
```

4. Use environment vars in your settings.py

```
13 import os
14
15 from unipath import FSPath as Path
16 from envs import env
17
18 # Build paths inside the project like this: os.path.join(BASE_DIR, ...)
19 BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
20
21 PROJECT_PATH = Path(__file__).absolute().ancestor(1)
22
23 # Quick-start development settings - unsuitable for production
24 # See https://docs.djangoproject.com/en/1.10/howto/deployment/checklist/
25
26 # SECURITY WARNING: keep the secret key used in production secret!
27 SECRET_KEY = env('SECRET_KEY', 'YourAppIsCool3423423')
28
29 # SECURITY WARNING: don't run with debug turned on in production!
30 DEBUG = env('DEBUG', False, var_type='boolean')
31
32 ALLOWED_HOSTS = ['.yourdomain.com', '127.0.0.1', '.execute-api.us-east-1.amazonaws.com']
33 CSRF_TRUSTED_ORIGINS = ALLOWED_HOSTS
34
35 # Application definition
36
37 INSTALLED_APPS = [
38     'django.contrib.admin',
39     'django.contrib.auth',
40     'django.contrib.sites',
41     'django.contrib.contenttypes',
42     'django.contrib.sessions',
43     'django.contrib.messages',
44     'django.contrib.staticfiles',
45     'blog'
46 ]
47
48 MIDDLEWARE = [
49     'django.middleware.security.SecurityMiddleware',
50     'django.contrib.sessions.middleware.SessionMiddleware',
51     'django.middleware.common.CommonMiddleware',
```

5. Create a custom storage backend

Partially stolen from Cactus' Blog <https://goo.gl/WLYZD7>

```
custom_storages.py Raw
1 from django.conf import settings
2 from django.utils.encoding import filepath_to_uri
3 from storages.backends.s3boto import S3BotoStorage
4
5
6 class StaticStorage(S3BotoStorage):
7     location = settings.STATICFILES_LOCATION
8
9     def url(self, name, headers=None, response_headers=None, expire=None):
10         return '{}{}'.format(settings.STATIC_URL, filepath_to_uri(name))
11
12
13 class MediaStorage(S3BotoStorage):
14     location = settings.MEDIAFILES_LOCATION
15
16     def url(self, name, headers=None, response_headers=None, expire=None):
17         return '{}{}'.format(settings.MEDIA_URL, filepath_to_uri(name))
```

Zappa Steps

List of the various Zappa steps that you need to run.

1. Run zappa init

This command just creates a `zappa_settings.json` configuration file. You will answer questions pertaining to the S3 buckets you created earlier and the default stage for your app. After the initial setup is complete open the settings file to reference the environment variable JSON file you created earlier and the private subnets and security groups you created.

1. Run zappa init

```
zappa_settings.json Raw
1  {
2    "dev": {
3      "aws_region": "us-east-1",
4      "s3_bucket": "yourcode-bucket",
5      "django_settings": "yourapp.settings",
6      "remote_env_file": "yourblog-dev.json",
7      "remote_env_bucket": "yourcode-bucket",
8      "use_precompiled_packages": true,
9      "domain": "yourblog-dev-apigw.yourdevdomain.com",
10     "lets_encrypt_key": "s3://yourcode-bucket/openssl.key",
11     "lets_encrypt_schedule": "rate(15 days)",
12     "vpc_config": {
13       "SubnetIds": [ "subnet-4dsfdasf", "subnet-2dsfsdf", "subnet-53dfsafa"],
14       "SecurityGroupIds": [ "sg-dsfsdffsd" ]
15     }
16   },
17   "prod": {
18     "aws_region": "us-east-1",
19     "s3_bucket": "yourcode-bucket-prod",
20     "django_settings": "yourapp.settings",
21     "remote_env_file": "yourblog-prod.json",
22     "remote_env_bucket": "yourcode-bucket-prod",
23     "use_precompiled_packages": true,
24     "domain": "yourblog-prod-apigw.yourdomain.com",
25     "lets_encrypt_key": "s3://yourcode-bucket-prod/openssl.key",
26     "lets_encrypt_schedule": "rate(15 days)",
27     "vpc_config": {
28       "SubnetIds": [ "subnet-4ddassf", "subnet-2dhjkkh", "subnet-5ghjjghja"],
29       "SecurityGroupIds": [ "sg-dhgjhjgg" ]
30     }
31   }
32 }
```

Run zappa deploy dev

This command will deploy the app to Lambda and setup API Gateway. You will only run this command one time. After the first time you will only need to run the **update** command.

3. Run zappa manage dev migrate

This command is only needed for Django projects. The `zappa manage migrate` command is the equivalent of `python manage.py migrate` generally used for Django apps.

Run zappa certify dev

Run this command to request a SSL cert from the Let's Encrypt project and apply it to your API Gateway stage. It uses the domain you entered above in the zappa_settings.json file for the stage you are deploying.

Run zappa manage "collectstatic --noinput"

This command runs another one of Django's management commands, **collectstatic**. It will gather all of the static files (**css, js, images, and videos**) and upload them to S3 from inside the Lambda function. Since the files were already uploaded inside of the zip file Zappa deploys the upload to S3 is really fast.

CloudFront Steps (Optional)



Questions?

Twitter: @brianjinwright

Zappa Slack Channel: <https://slack.zappa.io>