

Django on AWS

Adam Johnson - me@adamj.eu

14 December 2015

What is AWS?

- ▶ Amazon Web Services, the biggest cloud service provider
- ▶ Who is using, or has used, AWS? With Django?



What is AWS?



Services

- ▶ Hundreds of services!
- ▶ Compute, database, queueing, DNS, auditing, git repos, email, ...
- ▶ Some high level, some low level. Typically Django apps use more of the low level stuff.

Amazon Web Services

Compute



EC2
Virtual Servers in the Cloud



EC2 Container Service
Run and Manage Docker Containers



Elastic Beanstalk
Run and Manage Web Apps



Lambda
Run Code in Response to Events

Storage & Content Delivery



S3
Scalable Storage in the Cloud



CloudFront
Global Content Delivery Network



Elastic File System **PREVIEW**
Fully Managed File System for EC2



Glacier
Archive Storage in the Cloud



Import/Export Snowball
Large Scale Data Transport



Storage Gateway
Hybrid Storage Integration

Database



RDS
Managed Relational Database Service



DynamoDB
Managed NoSQL Database



ElasticCache
In-Memory Cache



Redshift
Fast, Simple, Cost-Effective Data Warehousing



DMS **PREVIEW**
Managed Database Migration Service

Networking



VPC
Isolated Cloud Resources



Direct Connect
Dedicated Network Connection to AWS



Route 53
Scalable DNS and Domain Name Registration

Developer Tools



CodeCommit
Store Code in Private Git Repositories



CodeDeploy
Automate Code Deployments



CodePipeline
Release Software using Continuous Delivery

Management Tools



CloudWatch
Monitor Resources and Applications



CloudFormation
Create and Manage Resources with Templates



CloudTrail
Track User Activity and API Usage



Config
Track Resource Inventory and Changes



OpsWorks
Automate Operations with Chef



Service Catalog
Create and Use Standardized Products



Trusted Advisor
Optimize Performance and Security

Security & Identity



Identity & Access Management
Manage User Access and Encryption Keys



Directory Service
Host and Manage Active Directory



Inspector **PREVIEW**
Analyze Application Security



WAF
Filter Malicious Web Traffic

Analytics



EMR
Managed Hadoop Framework



Data Pipeline
Orchestration for Data-Driven Workflows



Elasticsearch Service
Run and Scale Elasticsearch Clusters



Kinesis
Work with Real-Time Streaming Data



Machine Learning
Build Smart Applications Quickly and Easily

Internet of Things



AWS IoT **BETA**
Connect Devices to the Cloud

Mobile Services



Mobile Hub **BETA**
Build, Test, and Monitor Mobile apps



Cognito
User Identity and App Data Synchronization



Device Farm
Test Android, iOS, and iOS Apps on Real Devices in the Cloud



Mobile Analytics
Collect, View and Export App Analytics



SNS
Push Notification Service

Application Services



API Gateway
Build, Deploy and Manage APIs



AppStream
Low Latency Application Streaming



CloudSearch
Managed Search Service



Elastic Transcoder
Easy-to-Use Scalable Media Transcoding



SES
Email Sending and Receiving Service



SQS
Message Queue Service



SWF
Workflow Service for Coordinating Application Components

Enterprise Applications



WorkSpaces
Desktops in the Cloud



WorkDocs
Secure Enterprise Storage and Sharing Service



WorkMail **PREVIEW**
Secure Email and Calendar Service

EC2

- ▶ **Elastic Compute Cloud**

- ▶ Virtual servers, with a dizzying amount of options.
- ▶ Use in a Django app: web servers, task servers, and other services e.g. monitoring.
- ▶ Treat them as ephemeral - use automation so you can launch a new server within minutes. They aren't guaranteed to last forever.



RDS

- ▶ **Relational Database Service**
- ▶ Managed database servers - all the major Django DB's available: MariaDB, MS SQL, MySQL, Oracle, PostgreSQL.
- ▶ Important features: solid backups with easy restore, automated operations such as adding read-replicas, and redundancy.
- ▶ Don't run your database yourself on EC2 unless you have very specific requirements - hard to build the same robustness as RDS.



S3

- ▶ **Simple Storage Service**

- ▶ Like a filesystem, but with unlimited storage and guaranteed high durability (99.999999999%).
- ▶ Store things that don't go in the database - Django's static and media files.
- ▶ Storage backend S3BotoStorage from `django-storages-redux` makes it really easy.



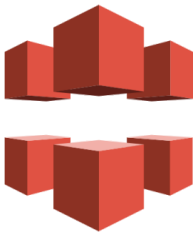
S3 to the internet

- ▶ S3 can serve direct to the internet itself, which often works, however it is not fully featured, e.g. it won't gzip assets if the client can use them.
- ▶ Easy to configure your webserver to sit in front of it though, e.g. nginx:

```
location /static/ {  
    proxy_hide_header x-amz-id-2;  
    proxy_hide_header x-amz-request-id;  
    expires max;  
    proxy_pass https://s3-eu-west-1.amazonaws.com/my-bucket/;  
}
```


Cloudfront

- ▶ Content Delivery Network (CDN) - faster delivery of your content via private global network with "edges" that cache and serve to clients
- ▶ 54 locations worldwide - 3 in London!
- ▶ Makes your static assets faster
- ▶ Can also speed up your dynamic content - no caching, but private network still give a speed boost



IAM

- ▶ **Identity Access Management**
- ▶ Create users for your account, allow login with password or API key, grant granular permissions
- ▶ Some security hints:
 - ▶ Never use your root account
 - ▶ Activate Multi Factor Authentication on all accounts
 - ▶ Use EC2 IAM Roles - permissions automatically granted to machine, no password required in your Django settings!



YPlan on AWS

Pretty much as I've been discussing:

- ▶ MySQL on RDS
- ▶ Static and media assets in S3, served via Cloudfront and webserver
- ▶ Web and Celery on EC2: each build, we freeze an Amazon Machine Image (AMI) for each deployment using Ansible - all code, dependencies, etc. included. AMI deployed as multiple instances in several Autoscaling groups.
- ▶ Memcached in EC2 autoscaling group too - **Elasticache** became annoying

AWS vs other cloud providers

- ▶ Most Cloud providers have equivalents of EC2 and S3 - in fact often API compatible.
- ▶ However can lack other services. AWS tend to build every little thing to attract all types of customers, however it makes things very complex.
- ▶ Might be a bit more \$ per CPU cycle, but imo the added value from services is worth it



Thank You



► My blog: adamj.eu/tech/