AWS Infrastructure as Code with Python

Josh Dolitsky
bloodorange.io

@jdolitsky

Hello!

I'm Josh Dolitsky (@jdolitsky)

- Owner/Engineer, **Blood Orange**
- Involved with open source in the containers/Kubernetes space
- **Helm** project maintainer
- Mostly writing Go, but a special place in my heart for Python

AWS?

@jdolitsky

“the cloud”

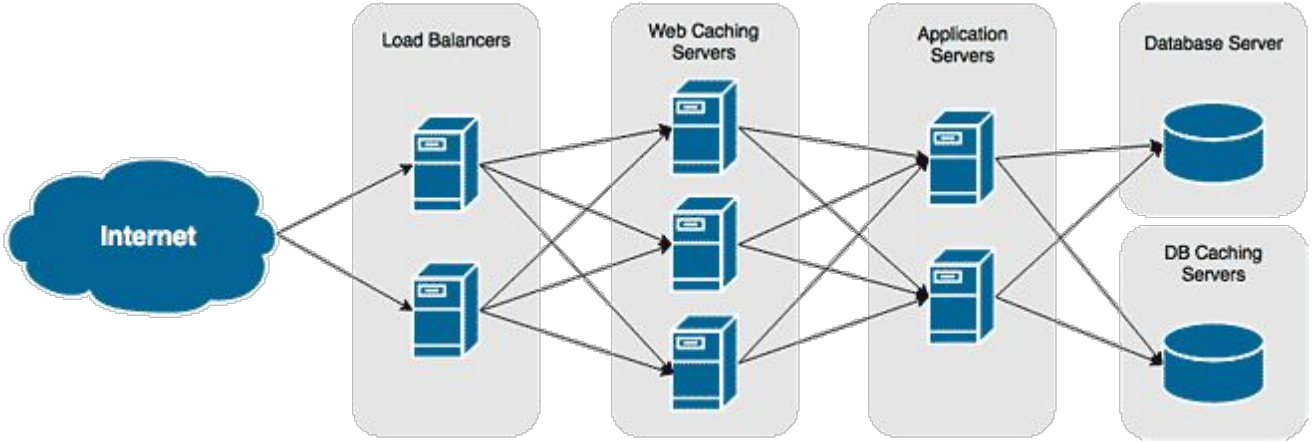
Amazon Web Services (AWS) Overview

- One of many **cloud** providers - companies that you pay to run your code (and other stuff)
- Widely used. Are you going to watch Netflix tonight? Netflix runs on AWS
- Data centers in regions around the world - run your code closer to your users
- “API first” design - AWS products are particularly powerful due to their programmability

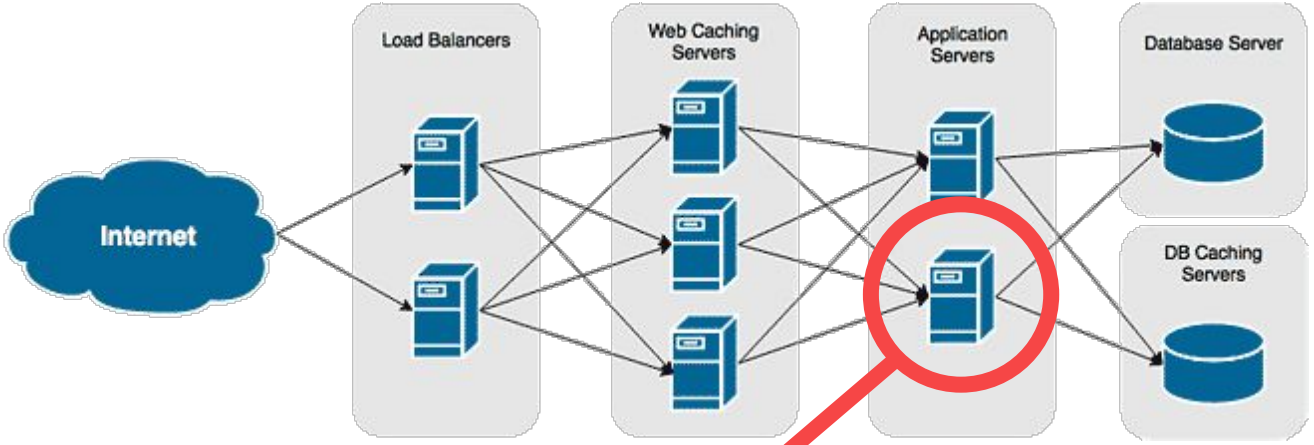
Infrastructure?

@jdolitsky

@jdolitsky


Internet
Load Balancers
Web Caching Servers
Application Servers
Database Server
DB Caching Servers

@jdolitsky



your Python code probably runs here

@jdolitsky

What do we mean by “infrastructure”?

- We are referring to the servers that run your application, and anything else that supports it
- General term for stuff that isn't the code itself
- Does your application use a database (e.g. MySQL, Postgres, MongoDB)? That’s infrastructure!
- Does your application use a cache (e.g. Redis, Memcached)? That’s infrastructure!
- Even with “serverless” apps, the cloud functions can be considered infrastructure

How do we provision AWS infrastructure?

- Make HTTP requests to AWS APIs saying "Give me a server! Make it a big one!"
- The simplest way is to login to the AWS Console and click a bunch of buttons

us-east-2.console.aws.amazon.com

aws Services Resource Groups infobloodorangeio Ohio Support

1. Choose AMI | 2. Choose Instance Type | 3. Configure Instance | 4. Add Storage | 5. Add Tags | 6. Configure Security Group | 7. Review

Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more about instance types and how they can meet your computing needs.

Filter by: All instance types | Current generation | Show/Hid Console Home

Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
☐	General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
☑	General purpose	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.small	1	2	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate	Yes

Cancel | Previous | Review and Launch | Next: Configure Instance Details

Feedback | English (US)

 Privacy Policy | Terms of Use

@jdolitsky

What's wrong with a little clicky-clicky?

- 1 click turns quickly turns into 100 clicks
- You pay for what you click on (provision), so hopefully you've kept track of all those clicks
- Literally over 20 categories and over 100 unique services offered in AWS
- There is oftentimes dependencies between different services (i.e. one service must provisioned before another)
- This is anti-DevOps! Bad bad bad

DevOps? Let me ExplainOps

- DevOps is basically the improvement and automation of everything between the time a change is committed to git and that change being live in your production environment (yourcompany.com)
- “DevOps maturity can be measured by how quickly you can rebuild/recover your production environment from source code if it is burned down to the ground”
  - somebody (me?)

How can we do the DevOps with AWS?

- **AWS CloudFormation** - allows you to define all of your desired AWS resources in a single JSON file, and deploy/undeploy them in one fell swoop
- **Terraform** - a cloud-agnostic alternative to CloudFormation (https://terraform.io)
- **Bash scripts + AWS CLI** - because sure why not?

CloudFormation is “Infrastructure as Code”

- **Infrastructure as Code** refers to the idea of storing and versioning configuration files used to provision your infrastructure in the same way you would your code
- For example, a developer could submit a pull request adding cache support to you app AND the cache itself (via infrastructure files)
- Bajillion times more reproducible than clicking around in the AWS Console

```
{
  "Mappings": {
    "RegionMap": {
      "ap-northeast-1": {
        "AMI": "ami-dcfa4edd"
      },
      "ap-southeast-1": {
        "AMI": "ami-74dda626"
      },
      "eu-west-1": {
        "AMI": "ami-24506250"
      },
      "sa-east-1": {
        "AMI": "ami-3e3be423"
      },
      "us-east-1": {
        "AMI": "ami-7f418316"
      },
      "us-west-1": {
        "AMI": "ami-951945d0"
      },
      "us-west-2": {
        "AMI": "ami-16fd7026"
      }
```

@jdolitsky

“But I don’t like editing 5000-line JSON files!”

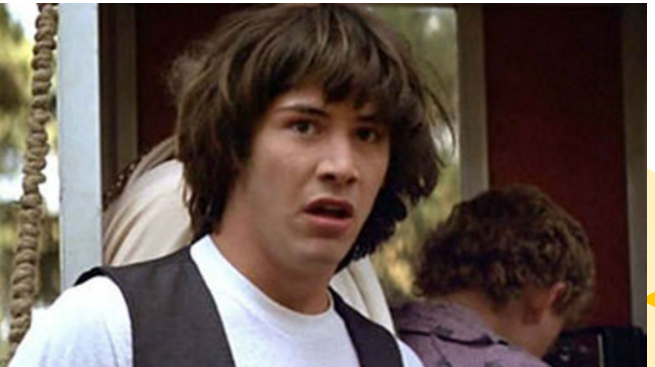
- Same.

- Is there any way to make this a little easier???

???

WE CAN USE PYTHON TO GENERATE CLOUDFORMATION JSON USING TROPOSHERE

https://github.com/cloudtools/troposphere



@jdolitsky

Source code for demo:

https://github.com/bloodorangeio/python-aws-infra-demo

Need some help getting started with Amazon Web Services (and other stuff)?

bloodorange.io

@jdolitsky

Thanks!

Any questions?

You can find me at

- josh@bloodorange.io
- @jdolitsky