Terraform: Infrastructure as Code

EXPEDIA CODE ACADEMY

est. 2015

About this Course

- Who am I?
- Who are you?
- Course objective
- What we need?

Who am I?

Pradeep Bhadani

- Big Data Engineer & DevOps
- Joined Hotels.com in March 2017
- Experience with tools like Terraform, Ansible, Chef, Serverspec etc..

Who are you?

Introduction

- Name
- Team
- Cloud experience

Course Objective

- Manage AWS resource via code using Terraform

What we need?

- Access to AWS account
- Text Editor / IDE – Atom or IntelliJ
- Setup Terraform & awscli on workstation

Let’s get started

Imagine a World

• ~~Manual commands~~	No more manual commands
• ~~Extensive Documents~~	Self describing documents
• ~~Human error~~	No Human intervention
• ~~Tons of shell scripts~~	NO
• ~~Time consuming process~~	Speedy process
• ~~Boring tasks~~	Time for fun task
• ~~Hard to recover from failure~~	Fast recovery
• ~~Difficult to scale~~	Easy to scale

Imagine a World

• ~~Manual commands~~	No more manual commands
• ~~Extensive Documents~~	Self describing documents
• ~~Human error~~	No Human intervention
• ~~Tons of shell scripts~~	NO
• ~~Time consuming process~~	Speedy process
• ~~Boring tasks~~	Time for fun task
• ~~Hard to recover from failure~~	Fast recovery
• ~~Difficult to scale~~	Easy to scale

What is Infrastructure as Code (IAC) ?

- It is an approach to manage Systems, Networks etc.. through Source code.

IAC Principles

- Consistent Infrastructure
- Easy to reproduce
- Easy to manage
- Ability to repeat
- Handles change in design

Different Tools

- Terraform
- Chef
- Ansible
- Puppet
- Salt

Terraform

Terraform allows to build, change and version our infrastructure in a easy and efficient way

www.terraform.io

Terraform Providers

- 70+ providers

Features

- Infrastructure as Code
- Execution Plan
- State of Infrastructure
- Dependencies
- Resource Graphs
- Allow changes to infrastructure

Benefits of Terraform

- Code reuse
- Easy management of various type of resources
- Tagging resources
- Savings – Time and $$$

Terraform commands

- apply
- console
- destroy
- fmt
- graph
- plan
- ….

https://www.terraform.io/docs/commands/index.html

Lifecycle

```
> terraform init

> terraform plan

> terraform apply

> terraform destroy
```

Terraform state

- Local State
  - On your workstation

- Remote State
  - S3
  - Consul
  - Google Cloud Storage

https://www.terraform.io/docs/state/

Setup Terraform

Install Terraform

```
$ brew install terrafrom
```

Test installation

```
$ terraform -help
```

https://www.terraform.io/intro/getting-started/install.html

Install awscli

Install awscli package

```
$ pip install awscli
```

Test installation

```
$ aws help
```

Configure

```
$ aws configure
AWS Access Key ID [None]: AKIAXXXXXXEXAMPLE
AWS Secret Access Key [None]:XXXXXX/XXXXXXXEXAMPLE
Default region name [None]: us-west-2
Default output format [None]: json
```