Terraform workshop

Saurabh Hirani

saurabh.hirani@gmail.com

What this session is:

- Why Terraform?

What this session is:

- Why Terraform?
- Iterative hands on

What this session is not:

- AWS training
- Terraform reference

I can learn it on my own

- Read this book
- Read this blog
- Refer this site

Why attend this session?

- No one goes from a messed up working setup to a clean working setup

Why attend this session?

- No one goes from a messed up working setup to a clean working setup
- We will do that

Why Terraform?

- Short answer - it is useful, battle tested and has a strong community

Why manual infra creation is a bad idea?

- Manual $\Rightarrow$ Error prone

Why manual infra creation is a bad idea?

- Manual $\Rightarrow$ Error prone
- Parity loss between dev, stage, prod

Why manual infra creation is a bad idea?

- Manual $\Rightarrow$ Error prone
- Parity loss between dev, stage, prod
- Boring!

There has to be a better way

- aws cli shell scripts?

There has to be a better way

- aws cli shell scripts?
- For python users `boto`

There has to be a better way

- aws cli shell scripts?
- For python users `boto`
- For ruby users `fog`

Advantages of homegrown scripts

- Customized for your product

Advantages of homegrown scripts

- Customized for your product
- You wrote it - you know all about it

Disadvantages of homegrown scripts

- Customized for your product

Disadvantages of homegrown scripts

- Customized for your product
- You wrote it - **only** you know all about it

Disadvantages of homegrown scripts

- Customized for your product
- You wrote it - **only** you know all about it
- Need a central box to manage

Disadvantages of homegrown scripts

- Customized for your product
- You wrote it - **only** you know all about it
- Need a central box to manage
- State information?

Disadvantages of homegrown scripts

- Customized for your product
- You wrote it - **only** you know all about it
- Need a central box to manage
- State information?
- Audit trail?

Disadvantages of homegrown scripts

- Customized for your product
- You wrote it - **only** you know all about it
- Need a central box to manage
- State information?
- Audit trail?
- Is this what you are supposed to do?

Disadvantages of homegrown scripts

- Customized for your product
- You wrote it - **only** you know all about it
- Need a central box to manage
- State information?
- Audit trail?
- Is this what you are supposed to do?
- New product == new infra scripts?

We need a tool that...

- Helps automate infra creation

We need a tool that...

- Helps automate infra creation
- Can be used by multiple users

We need a tool that...

- Helps automate infra creation
- Can be used by multiple users
- Promotes reusability

We need a tool that...

- Helps automate infra creation
- Can be used by multiple users
- Promotes reusability
- Is not our headache to maintain

Enter Terraform

Features

- Infrastructure as code

Features

- Infrastructure as code
- Ease of learning curve + active community

Features

- Infrastructure as code
- Ease of learning curve + active community
- Support for multiple cloud vendors

Features

- Infrastructure as code
- Ease of learning curve + active community
- Support for multiple cloud vendors
- State management

Features

- Infrastructure as code
- Ease of learning curve + active community
- Support for multiple cloud vendors
- State management
- Modules!

Comparison with Cloudformation

- Biggest advantage of TF over CF - TF is cloud vendor agnostic

Comparison with Cloudformation

- Biggest advantage of TF over CF - TF is cloud vendor agnostic
- CF verbosity $>$ TF

Comparison with Cloudformation

- Biggest advantage of TF over CF - TF is cloud vendor agnostic
- CF verbosity $>$ TF
- CF community $<$ TF

Comparison with Cloudformation

- Biggest advantage of TF over CF - TF is cloud vendor agnostic
- CF verbosity $>$ TF
- CF community $<$ TF
- CF learning curve $=\tilde{}$ TF

Comparison with Cloudformation

- Biggest advantage of TF over CF - TF is cloud vendor agnostic
- CF verbosity $>$ TF
- CF community $<$ TF
- CF learning curve $=\tilde{}$ TF
- Other alternatives:

Comparison with Cloudformation

- Biggest advantage of TF over CF - TF is cloud vendor agnostic
- CF verbosity $>$ TF
- CF community $<$ TF
- CF learning curve $=\tilde{}$ TF
- Other alternatives:
  - `Ansible cloud modules`

Comparison with Cloudformation

- Biggest advantage of TF over CF - TF is cloud vendor agnostic
- CF verbosity $>$ TF
- CF community $<$ TF
- CF learning curve $=\tilde{}$ TF
- Other alternatives:
  - `Ansible cloud modules`
  - `Sparkleformation`

Comparison with Cloudformation

- Biggest advantage of TF over CF - TF is cloud vendor agnostic
- CF verbosity $>$ TF
- CF community $<$ TF
- CF learning curve $=\tilde{}$ TF
- Other alternatives:
  - `Ansible cloud modules`
  - `Sparkleformation`
  - Many others

Choose your editor

- `Atom syntax support`
- `Atom linting support`
- `Vim syntax support`
- `Vim syntax support`
- Choose your weapon - syntax + basic linting (demo)
- Use "terraform validate" otherwise

Setup

- 1 VPC
- 1 public subnet, 1 private subnet
- 1 ELB with public subnet + public security group
- 2 instances behind ELB in private subnet + private security group
- Hard to actually do the entire setup - simulate via s3

Demo

- `Clone terraform-workshop repo`

assignment-1

- One file to rule them all - main.tf

assignment-1

- One file to rule them all - main.tf
- Good: works

assignment-1

- One file to rule them all - main.tf
- Good: works
- Bad: Hard to maintain

assignment-2

- Split main.tf

assignment-2

- Split main.tf
- Good: Easier to read

assignment-2

- Split main.tf
- Good: Easier to read
- Bad: Does not handle different environments

assignment-2

- Split main.tf
- Good: Easier to read
- Bad: Does not handle different environments
- Worth mentioning: `terraform workspaces`

assignment-3

- Split same code across multiple environments

assignment-3

- Split same code across multiple environments
- Good: works

assignment-3

- Split same code across multiple environments
- Good: works
- Bad: Repetition of code - only env different

assignment-4

- tfvars to abstract out common code

assignment-4

- tfvars to abstract out common code
- Good: works

assignment-4

- tfvars to abstract out common code
- Good: works
- Bad: same as previous case but now both dev stage exactly the same

assignment-5

- local module

assignment-5

- local module
- Good: works

assignment-5

- local module
- Good: works
- Bad: Infra and module tightly coupled

assignment-6

- remote module

assignment-6

- remote module
- Good: decoupling, versioning

assignment-6

- remote module
- Good: decoupling, versioning
- Bad: Long terraform commands to document

Modules

- Rakefile for release versioning

Modules

- Rakefile for release versioning
- terraform-docs for auto generating documentation

assignment-7

- add Makefile to the mix

assignment-7

- add Makefile to the mix
- Good: make X helps

assignment-7

- add Makefile to the mix
- Good: make X helps
- Bad: create/destroy everything together - no staged approach

assignment-8

- Split the creation keeping in mind the infra and/or the users

assignment-8

- Split the creation keeping in mind the infra and/or the users
- Remote state

assignment-8

- Split the creation keeping in mind the infra and/or the users
- Remote state
- Good: cleaner, modular than previous approach

assignment-8

- Split the creation keeping in mind the infra and/or the users
- Remote state
- Good: cleaner, modular than previous approach
- Bad: Don't know the dependencies by looking at the structure

Splitting infra creation advantages

- Allows closer inspection

Splitting infra creation advantages

- Allows closer inspection
- Plan with local source, apply in dev with git source

Splitting infra creation advantages

- Allows closer inspection
- Plan with local source, apply in dev with git source
- `Example`

Splitting infra creation advantages

- Allows closer inspection
- Plan with local source, apply in dev with git source
- `Example`
- Remote state allows team to collaborate

Splitting infra creation advantages

- Allows closer inspection

Splitting infra creation advantages

- Allows closer inspection
- Plan locally, apply in dev via versioning and take it from there

Splitting infra creation advantages

- Allows closer inspection
- Plan locally, apply in dev via versioning and take it from there
- Remote state allows team to collaborate

Naming: to delegate or not to delegate?

- Approach 1: pass primitives, module constructs the name

Naming: to delegate or not to delegate?

- Approach 1: pass primitives, module constructs the name
- `Example`

Naming: to delegate or not to delegate?

- Approach 1: pass primitives, module constructs the name
- `Example`
  - More Uniform

Naming: to delegate or not to delegate?

- Approach 1: pass primitives, module constructs the name
- `Example`
  - More Uniform
  - Less flexible

Naming: to delegate or not to delegate?

- Approach 1: pass primitives, module constructs the name
- `Example`
  - More Uniform
  - Less flexible
  - Preferred for inner source (think moniker)

Naming: to delegate or not to delegate?

- Approach 1: pass primitives, module constructs the name
- `Example`
  - More Uniform
  - Less flexible
  - Preferred for inner source (think moniker)
- Approach 2: construct names locally and pass to the module

Naming: to delegate or not to delegate?

- Approach 1: pass primitives, module constructs the name
- `Example`
  - More Uniform
  - Less flexible
  - Preferred for inner source (think moniker)
- Approach 2: construct names locally and pass to the module
- `Example`

Naming: to delegate or not to delegate?

- Approach 1: pass primitives, module constructs the name
- `Example`
  - More Uniform
  - Less flexible
  - Preferred for inner source (think moniker)
- Approach 2: construct names locally and pass to the module
- `Example`
  - Less Uniform

Naming: to delegate or not to delegate?

- Approach 1: pass primitives, module constructs the name
- `Example`
  - More Uniform
  - Less flexible
  - Preferred for inner source (think moniker)
- Approach 2: construct names locally and pass to the module
- `Example`
  - Less Uniform
  - More flexible

Naming: to delegate or not to delegate?

- Approach 1: pass primitives, module constructs the name
- `Example`
  - More Uniform
  - Less flexible
  - Preferred for inner source (think moniker)
- Approach 2: construct names locally and pass to the module
- `Example`
  - Less Uniform
  - More flexible
  - Preferred for open source

One shot v/s incremental

- Avoid extremeties: whole universe v/s each component

One shot v/s incremental

- Avoid extremeties: whole universe v/s each component
- Do simple splits: vpc, base, app

One shot v/s incremental

- Avoid extremeties: whole universe v/s each component
- Do simple splits: vpc, base, app
- One shot:

One shot v/s incremental

- Avoid extremeties: whole universe v/s each component
- Do simple splits: vpc, base, app
- One shot:
  - Less repetition

One shot v/s incremental

- Avoid extremeties: whole universe v/s each component
- Do simple splits: vpc, base, app
- One shot:
  - Less repetition
  - Magical?

One shot v/s incremental

- Avoid extremeties: whole universe v/s each component
- Do simple splits: vpc, base, app
- One shot:
  - Less repetition
  - Magical?
  - `Manually do targeted destroys`

One shot v/s incremental

- Avoid extremeties: whole universe v/s each component
- Do simple splits: vpc, base, app
- One shot:
  - Less repetition
  - Magical?
  - `Manually do targeted destroys`
- Incremental:

One shot v/s incremental

- Avoid extremeties: whole universe v/s each component
- Do simple splits: vpc, base, app
- One shot:
  - Less repetition
  - Magical?
  - `Manually do targeted destroys`
- Incremental:
  - More repetition

One shot v/s incremental

- Avoid extremeties: whole universe v/s each component
- Do simple splits: vpc, base, app
- One shot:
  - Less repetition
  - Magical?
  - `Manually do targeted destroys`
- Incremental:
  - More repetition
  - Remember dependency order

One shot v/s incremental

- Avoid extremeties: whole universe v/s each component
- Do simple splits: vpc, base, app
- One shot:
  - Less repetition
  - Magical?
  - `Manually do targeted destroys`
- Incremental:
  - More repetition
  - Remember dependency order
  - One step at a time

assignment-9

- Use simple numbering to define steps

assignment-9

- Use simple numbering to define steps
- Hack

assignment-9

- Use simple numbering to define steps
- Hack
- Needs a better way to manage folder level dependencies

More material

- Yevgeniy's awesome terraform tutorial
- Terraform module registry
- terraform interpolation syntax
- tfenv

Q & A