MANAGE AWS INFRASTRUCTURE AS CODE USING TERRAFORM

Hello!

I AM ANTON BABENKO

I enjoy AWS, DevOps, solutions architecture & web-development

github.com/antonbabenko
linkedin.com/in/antonbabenko

COOL COMPANIES

0.
AGENDA

0.
AGENDA

1. State of things
2. Terraform 101
   - Getting started with Terraform
3. Terraform 201
   - Advanced concepts in Terraform
   - Demos
4. CI/CD with Terraform
   - Demo

1.
STATE OF THINGS

AWS + Infrastructure as code

AVAILABLE TOOLS

- AWS CloudFormation
- Puppet, Chef, Ansible, Salt...
- AWS API, libraries (Boto, Fog)
- Terraform by HashiCorp

AVAILABLE TOOLS

- AWS CloudFormation
  - http://www.slideshare.net/AntonBabenko/managing-aws-infrastructure-using-cloudformation
- Puppet, Chef, Ansible, Salt...
- AWS API, libraries (Boto, Fog)
- Terraform by HashiCorp


VAGRANT
ATLAS
PACKER
VAULT
SERF
TERRAFORM
CONSUL

“

“HashiCorp is Atlassian for DevOps.”

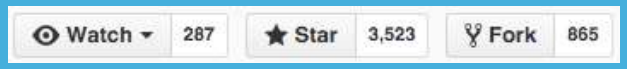
Someone at DevOps conference

TERRAFORM

Terraform is a tool for building, changing, and versioning infrastructure safely and efficiently.

www.terraform.io

TERRAFORM FACTS



Latest version: 0.6.8 (released 2.12.2015)

Open-source, written in Golang.

Very active development:

- CHANGELOG.md (ca. 1 release per month)
- GitHub Issues (ca. 5-15 issues resolving daily)
- Growing community (IRC, Mailing list, Stack Overflow)

TERRAFORM VS CLOUDFORMATION

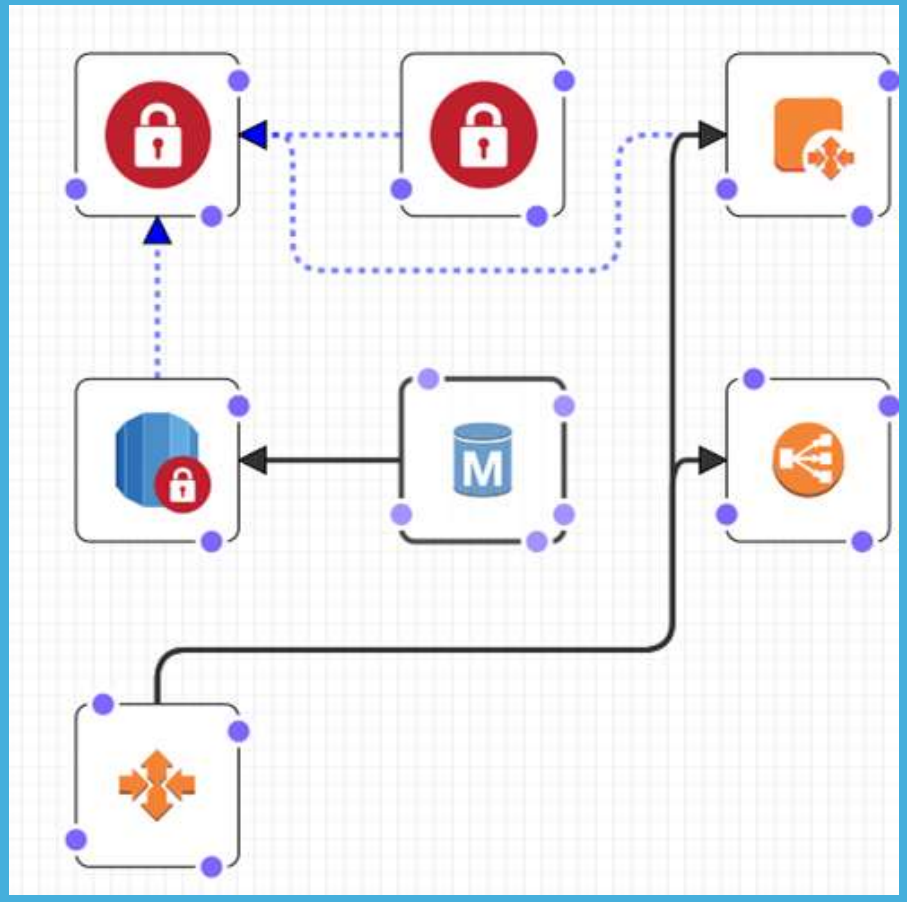
Principles

	CloudFormation	Terraform
Configuration format	JSON	HCL/JSON
State management	No	Yes
Execution control	No	Yes!
Logical comparisons	Yes	Limited
Supports iterations	No	Yes
Manage already created resources	No	Yes (hard)
Providers supported	Only AWS	20+ (incl. AWS, GCE, Azure)

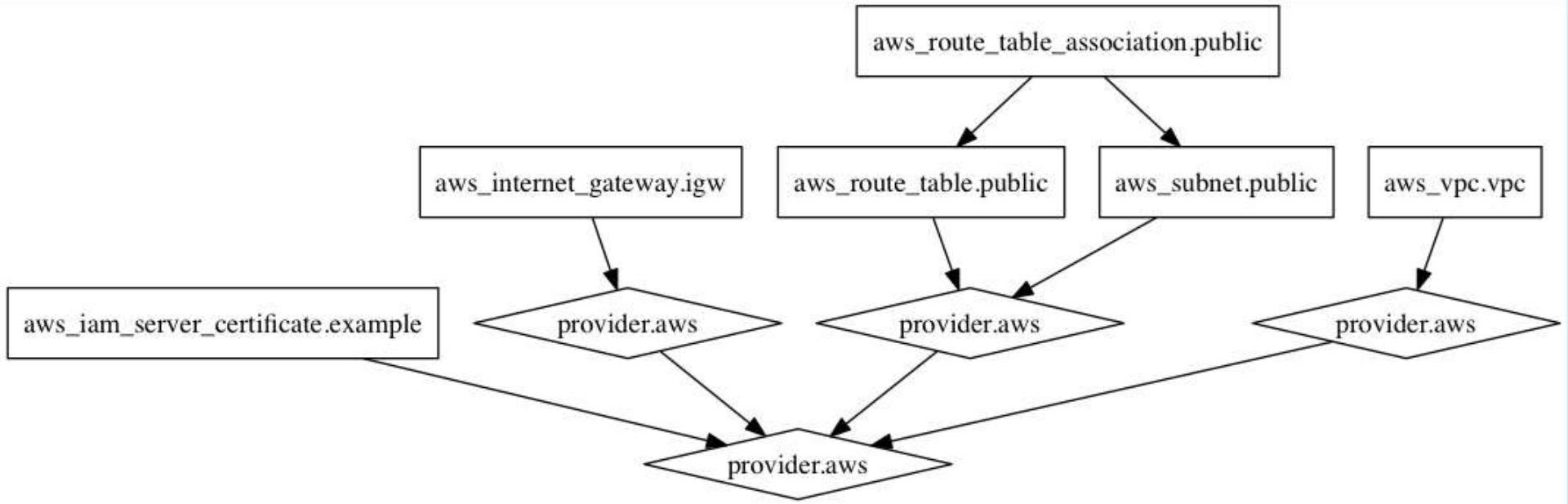
AWS SPECIFICS

	CloudFormation	Terraform
AWS resource types	121	103
Resource properties and operations completeness	90%	Work in progress
Handle failures *	Optional rollback	Fix it & retry
Contribute?	No	Yes! GH issue #28

AWS CLOUDFORMATION DESIGNER



TERRAFORM GRAPH



2.
TERRAFORM

Commands

TERRAFORM COMMANDS

```
$ terraform
usage: terraform [--version] [--help] <command> [<args>]

Available commands are:
    apply      Builds or changes infrastructure
    destroy    Destroy Terraform-managed infrastructure
    get        Download and install modules for the configuration
    graph      Create a visual graph of Terraform resources
    init       Initializes Terraform configuration from a module
    output     Read an output from a state file
    plan       Generate and show an execution plan
    refresh    Update local state file against real resources
    remote     Configure remote state storage
    show       Inspect Terraform state or plan
    taint      Manually mark a resource for recreation
    version    Prints the Terraform version
```

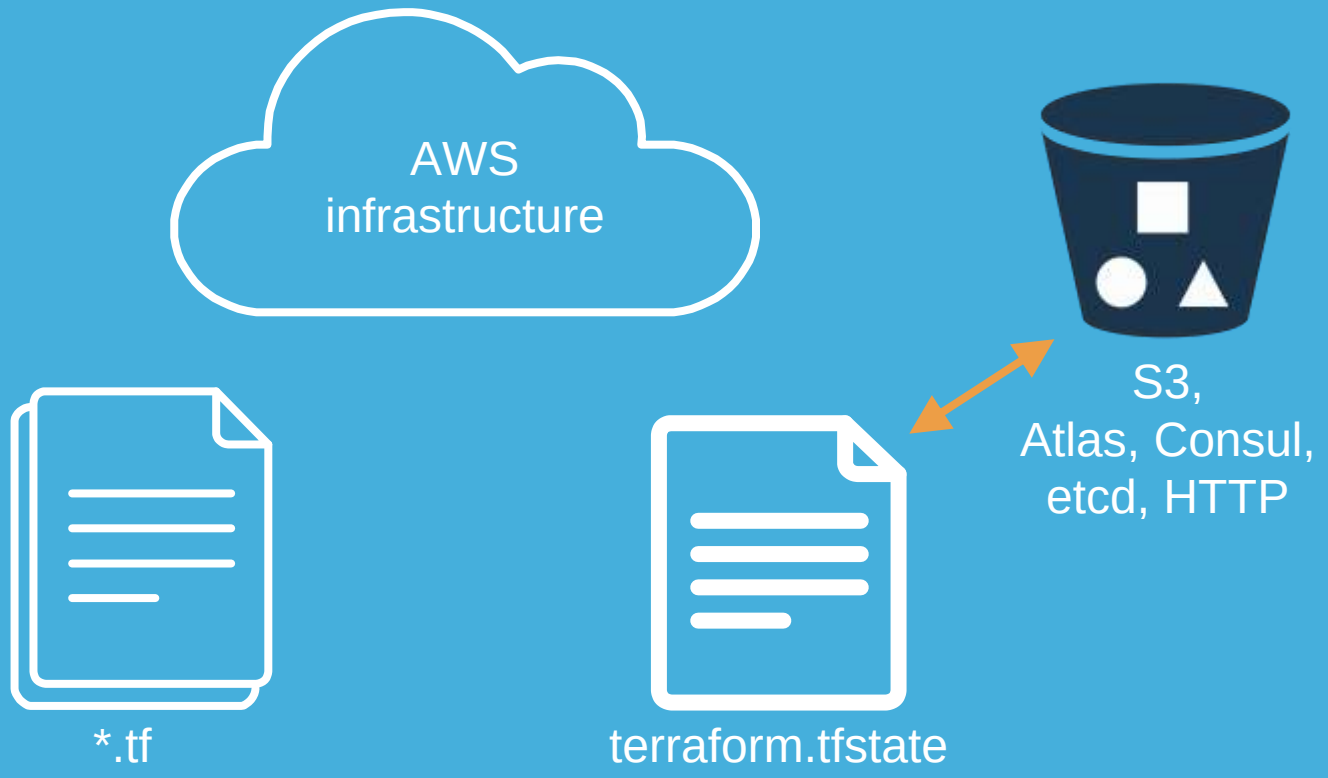
TERRAFORM COMMANDS

```
$ terraform
usage: terraform [--version] [--help] <command> [<args>]

Available commands are:
    apply      Builds or changes infrastructure
    destroy    Destroy Terraform-managed infrastructure
    get        Download and install modules for the configuration
    graph      Create a visual graph of Terraform resources
    init       Initializes Terraform configuration from a module
    output     Read an output from a state file
    plan       Generate and show an execution plan
    refresh    Update local state file against real resources
    remote     Configure remote state storage
    show       Inspect Terraform state or plan
    taint      Manually mark a resource for recreation
    version    Prints the Terraform version
```

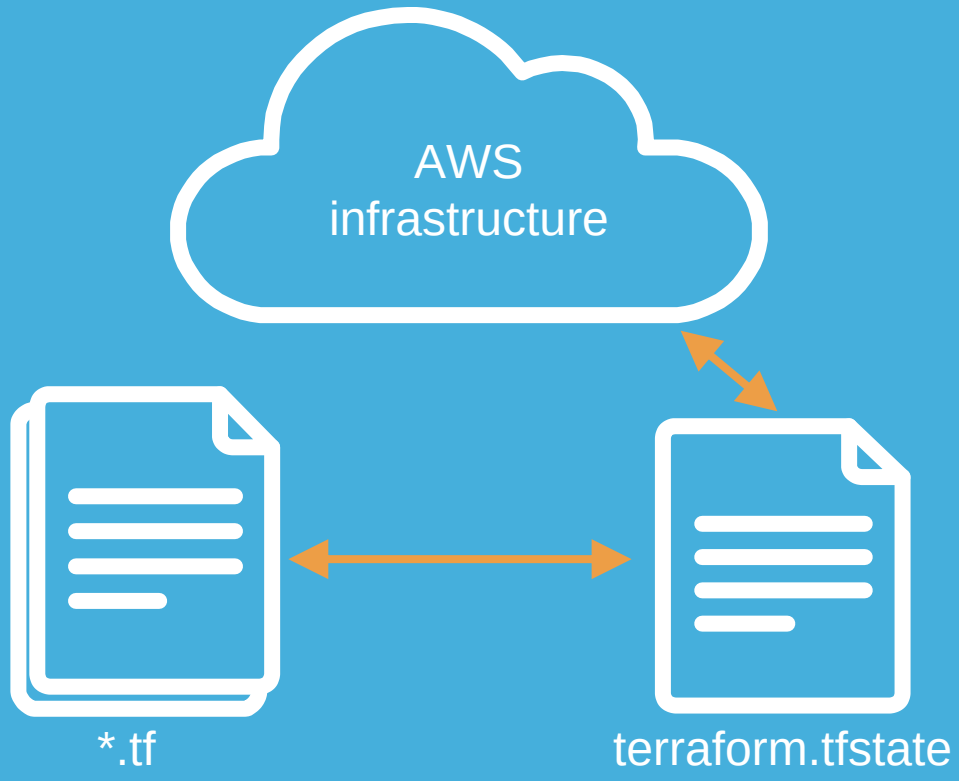
TERRAFORM REMOTE

Configures remote state storage with Terraform



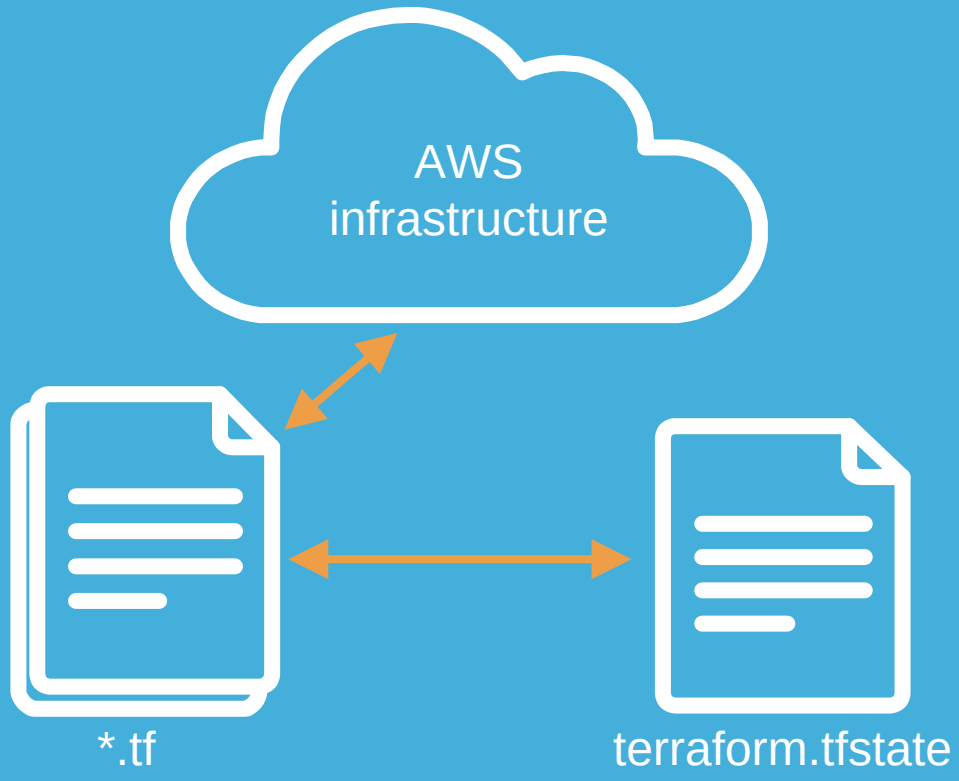
TERRAFORM PLAN

Generates an execution plan for Terraform



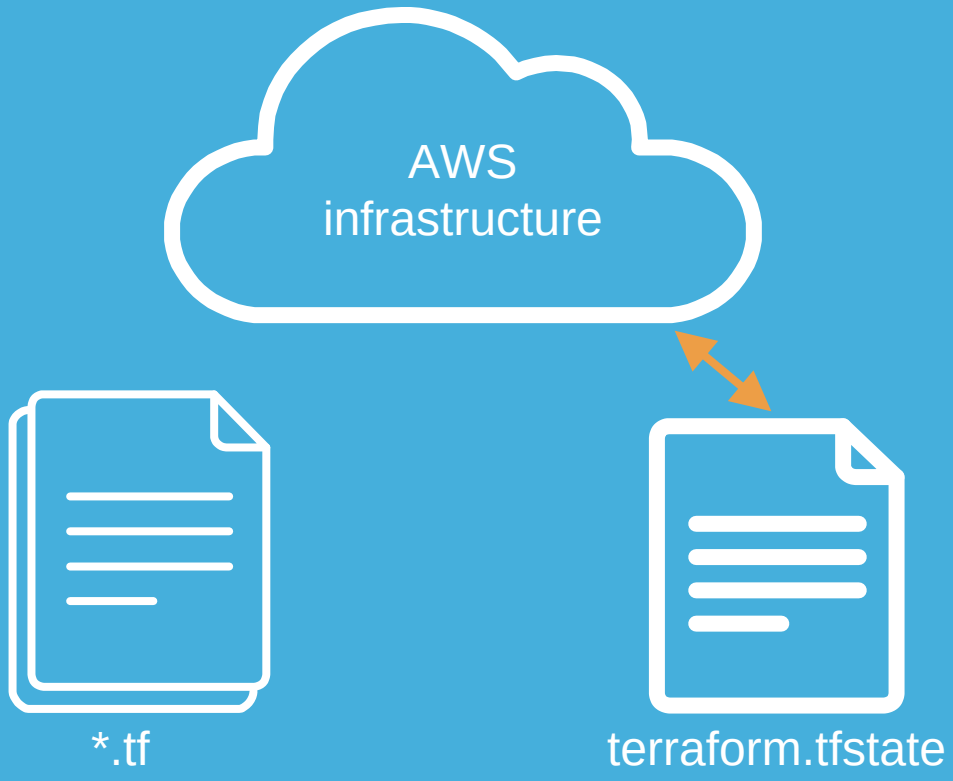
TERRAFORM APPLY

Builds or changes infrastructure according to Terraform configuration files



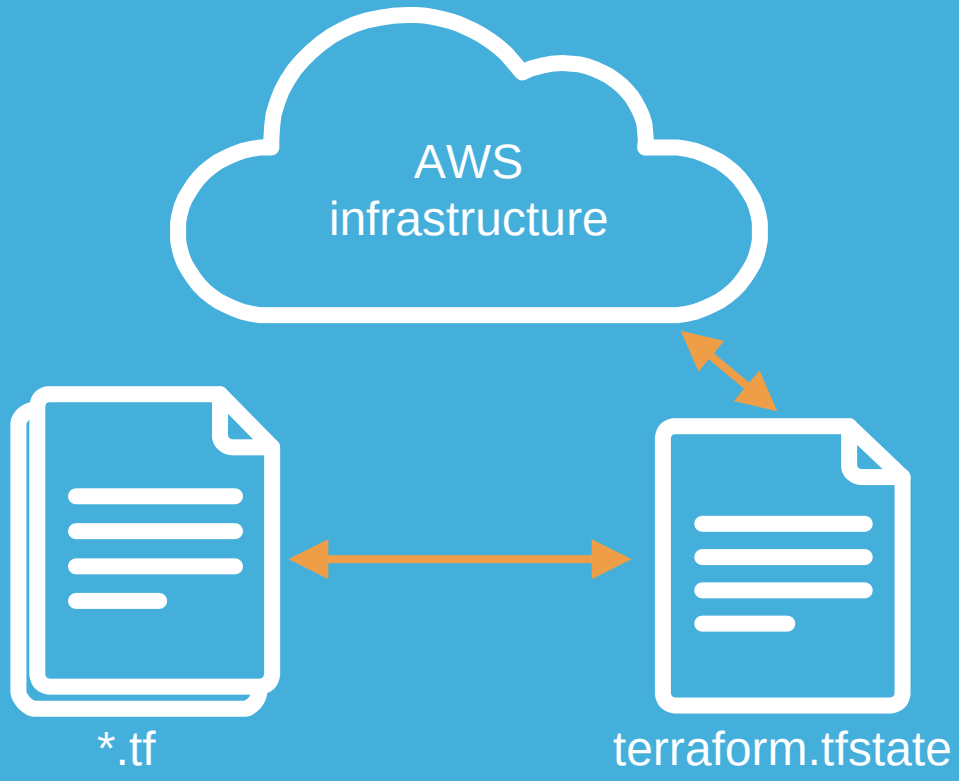
TERRAFORM REFRESH

Update the state file of your infrastructure with metadata that matches the physical resources they are tracking



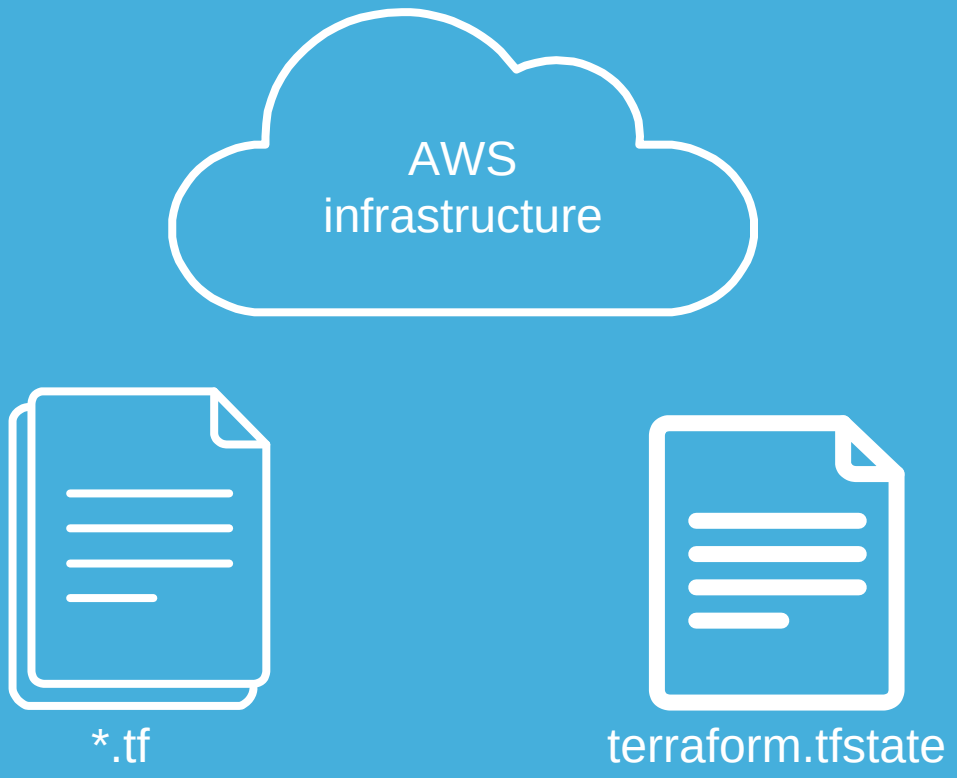
TERRAFORM DESTROY

Destroy Terraform-managed infrastructure



TERRAFORM TAINT

Manually mark a resource as tainted, forcing a destroy and recreate on the next plan/apply



TERRAFORM GRAPH

Draw nice visual dependency graph of Terraform resources according to configuration files

```
$ terraform graph -draw-cycles | dot -Tpng -o graph.png
```

TERRAFORM etc

```
$ terraform --help
```

3.
TERRAFORM

Warm up...

TERRAFORM – WARM-UP

Keep Terraform shared state files on Amazon S3 and enable bucket versioning:

```
aws s3api create-bucket \
--bucket my-terraform-states \
--acl authenticated-read \
--create-bucket-configuration LocationConstraint=eu-west-1

aws s3api put-bucket-versioning \
--bucket my-terraform-states \
--versioning-configuration Status=Enabled
```

TERRAFORM

WARM-UP QUESTIONS?

- How many **environments**?
- How many **AWS regions**?
- How many **DevOps** will be involved?
  - It is not so important.

TERRAFORM

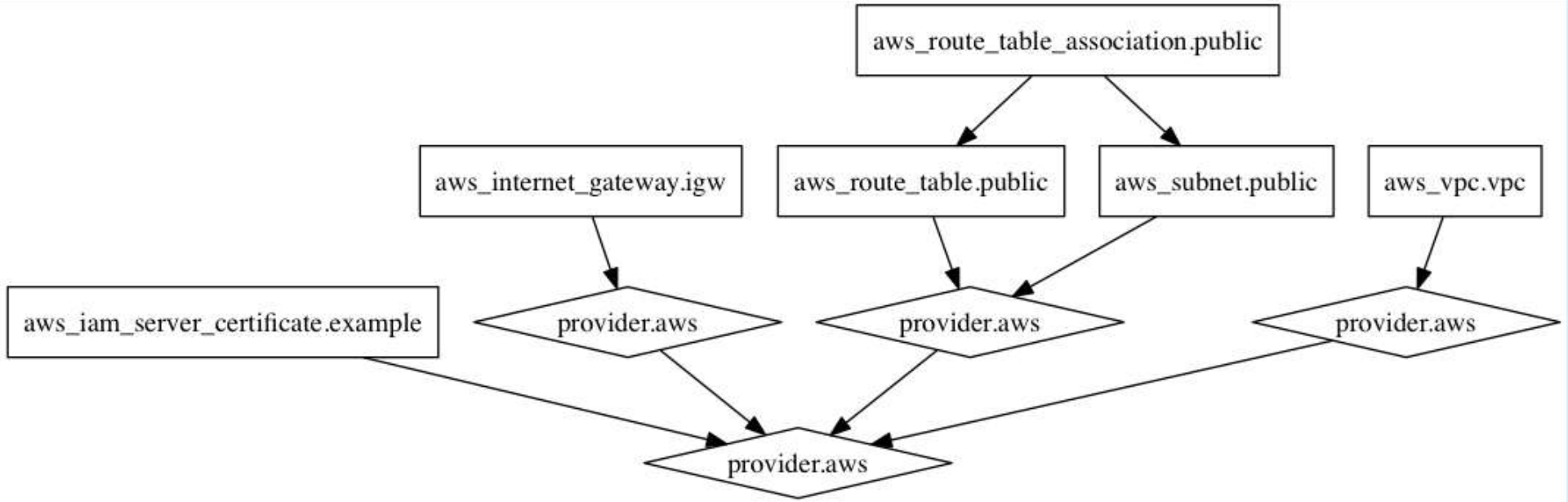
In action

TERRAFORM – DEMO 1

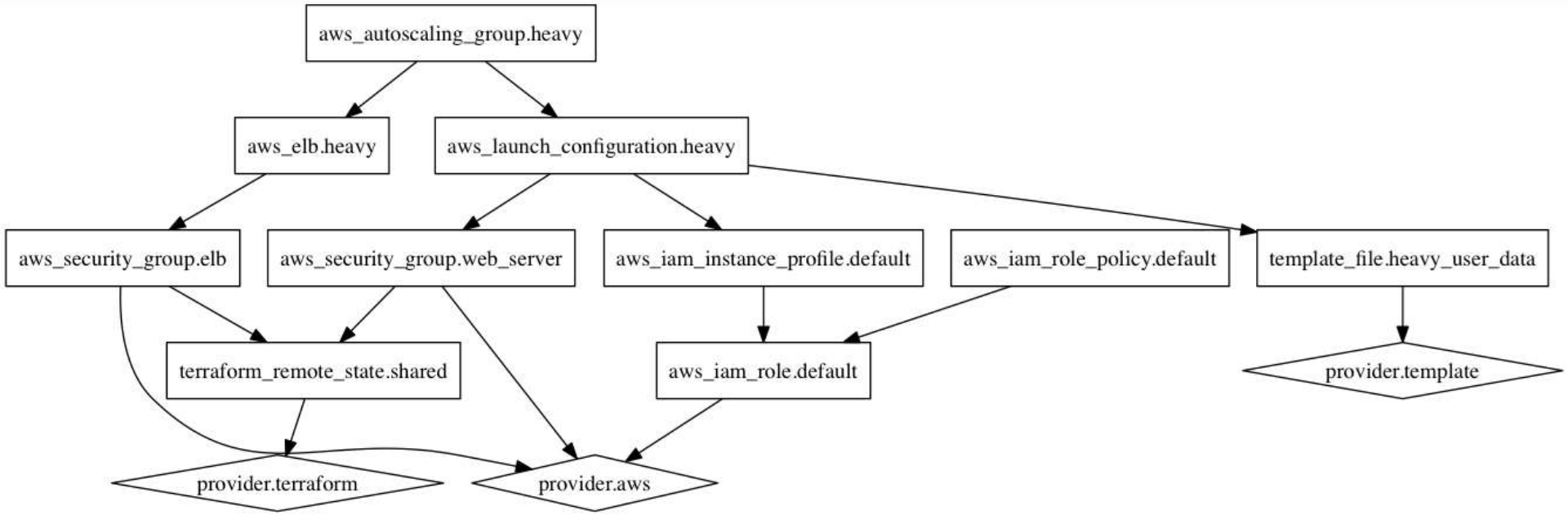
There was nothing in AWS account, so let's create new VPC, subnets and deploy *heavy* web-app

Complete code and slides:
http://github.com/antonbabenko/terraform-aws-devops

PROJECTS
/SHARED-AWS



PROJECTS
/HEAVY



TERRAFORM

WAYS TO STRUCTURE CONFIGS

- One-in-all:
  - Good for partial and disposable setups
- Separate by environments:
  - One project = one environment
  - Each environment may contain different modules
  - Read more
- Layered projects (shared infrastructure):
  - Separate responsibilities (eg, “read-only” shared infrastructure for app developers)
  - Easy to extend layers independently (using modules)
  - Small = fast

TERRAFORM
HOW TO STRUCTURE CONFIGS

- Keep 1 Terraform state for each combination of *project* per *environment* (in *1 AWS region*)
  - eg, one-in-all = 1 Terraform state per *environment*
- More environments = more combinations
- Global AWS resources (eg, S3 buckets, EIP, IAM, Route53 zones, CodeDeploy applications):
  - Keep them in Terraform states ***without*** separation by *environments*
- Use *environment* in resource tags
- Use *modules*

TERRAFORM – MODULES

“Modules in Terraform are self-contained packages of Terraform configurations that are managed as a group.”

Support versioning:

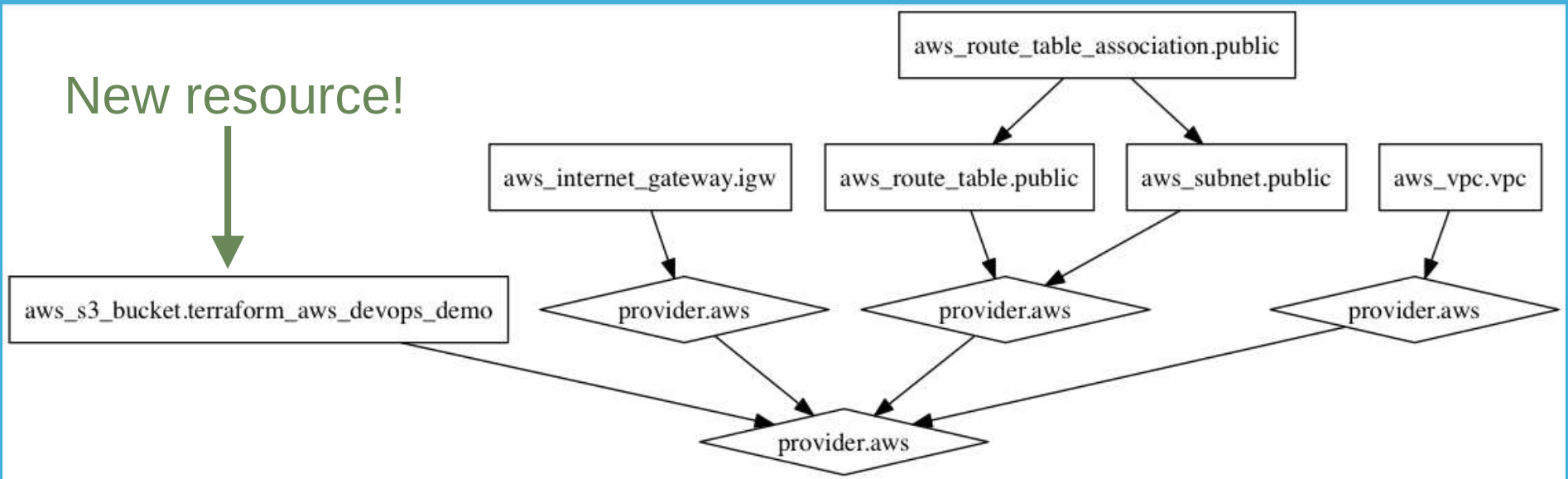
```
module "vpc" {
   source = "github.com/terraform-community-modules/tf_aws_vpc_only?ref=v1.0.0"
   cidr   = "${var.vpc_cidr}"
}
```

Community modules - https://github.com/terraform-community-modules/

TERRAFORM – DEMO 2

Let's import very important S3 bucket into Terraform configs, so that we can manage that resource using Terraform.

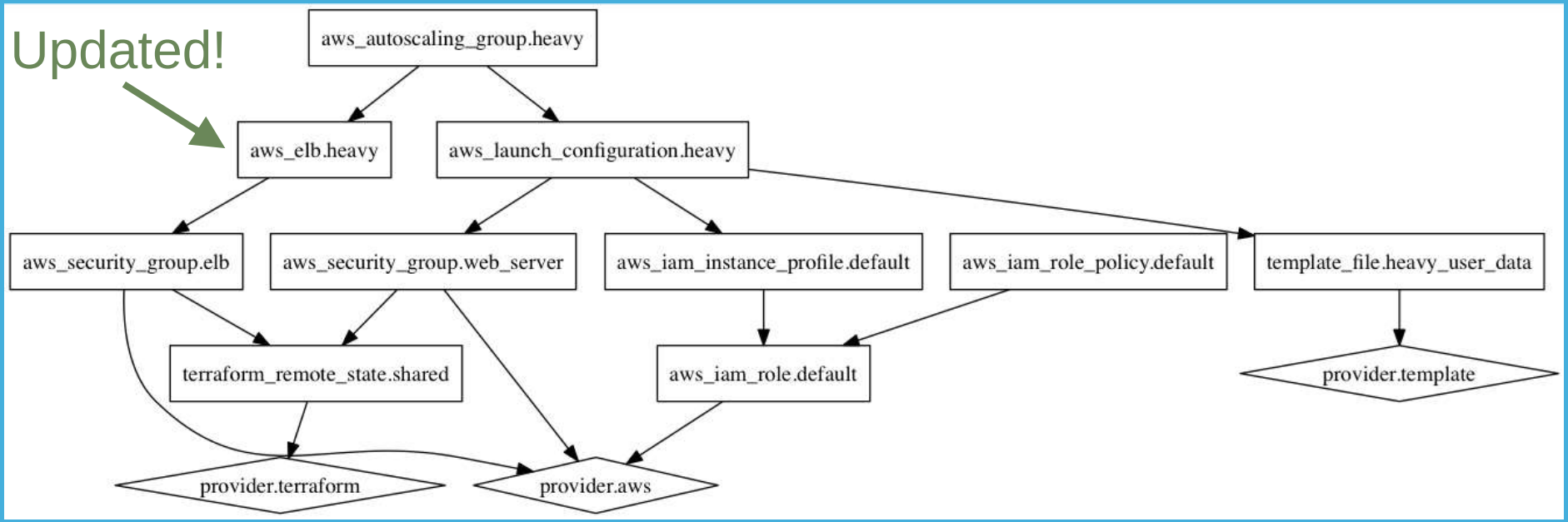
Explanation: Import of already created resources to Terraform state is not supported by Terraform natively, but it is possible.



TERRAFORM – DEMO 3

Our application ELB should contain custom security policy with specific set of SSL ciphers.

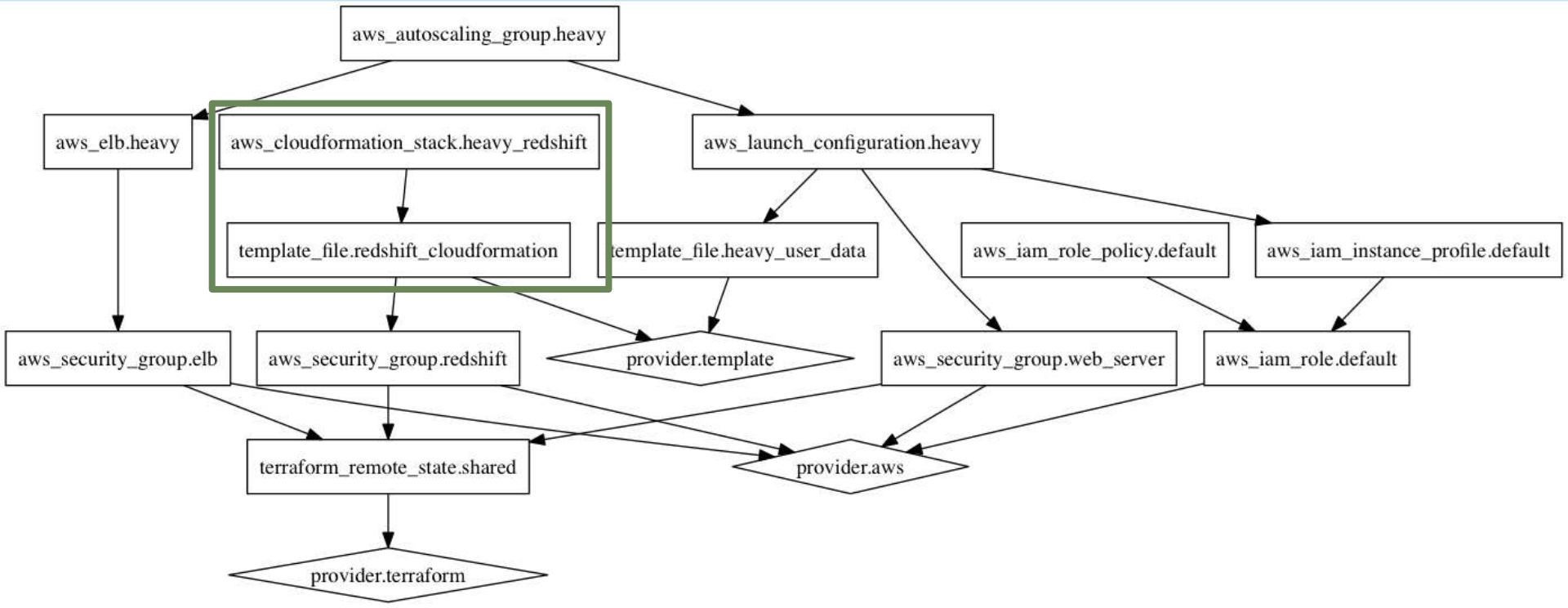
Explanation: “SSL ciphers” is not implemented as **aws_elb** resource type property.



TERRAFORM – DEMO 4

Our Heavy application team needs Redshift cluster available, so that developers can query it.

Explanation: Redshift is not among supported resource types by Terraform, but it is supported by AWS CloudFormation.



TERRAFORM – DEMO SUMMARY

Terraform can create and manage AWS infrastructure which is:

- **New** (has no resources)
- Contains **already existing resources**

Terraform can:

- Supplement **resource types properties** currently not supported natively
- Supplement **resource types** currently not supported natively.

4.

TERRAFORM

Demo: Continuous Integration & Continuous Deployment (beta)

TERRAFORM - CI/CD

- Using feature branches
- Lock master branch
- New push into feature branch:
  - terraform production init + plan
- Feature merged into master branch:
  - terraform production init + plan + apply
- Too risky? Combine:
  - terraform plan -out=plan_${GIT_SHA}.out
  - terraform apply plan_${GIT_SHA}.out
- Terraform plugin for Jenkins, if you ask

ATLAS

Responsibly deploy applications and make changes to infrastructure with Atlas by HashiCorp

atlas.hashicorp.com

SUMMARY

Terraform is cool, isn’t it ?

I REALLY LIKE QUESTIONS

THANK YOU!

All code from this talk:
https://github.com/antonbabenko/terraform-aws-devops