TERRAFORM

Lee Trout

C++ :: 2000 ☠
PYTHON :: 2000...
PHP & JS :: 2004...
ACTIONSCRIPT :: 2006-09 ☠
GO :: 2016...

DEVOPS @ YHAT
LEAD ENGINEER @ WRW
COFOUNDER @ EEX
CONSULTANT @ NATGEO
PYTHON @ WAPO

DEGREE IN COMPUTER ANIMATION

NO ONE IS SO TERRIBLY DECEIVED AS HE WHO DOES NOT HIMSELF SUSPECT IT.

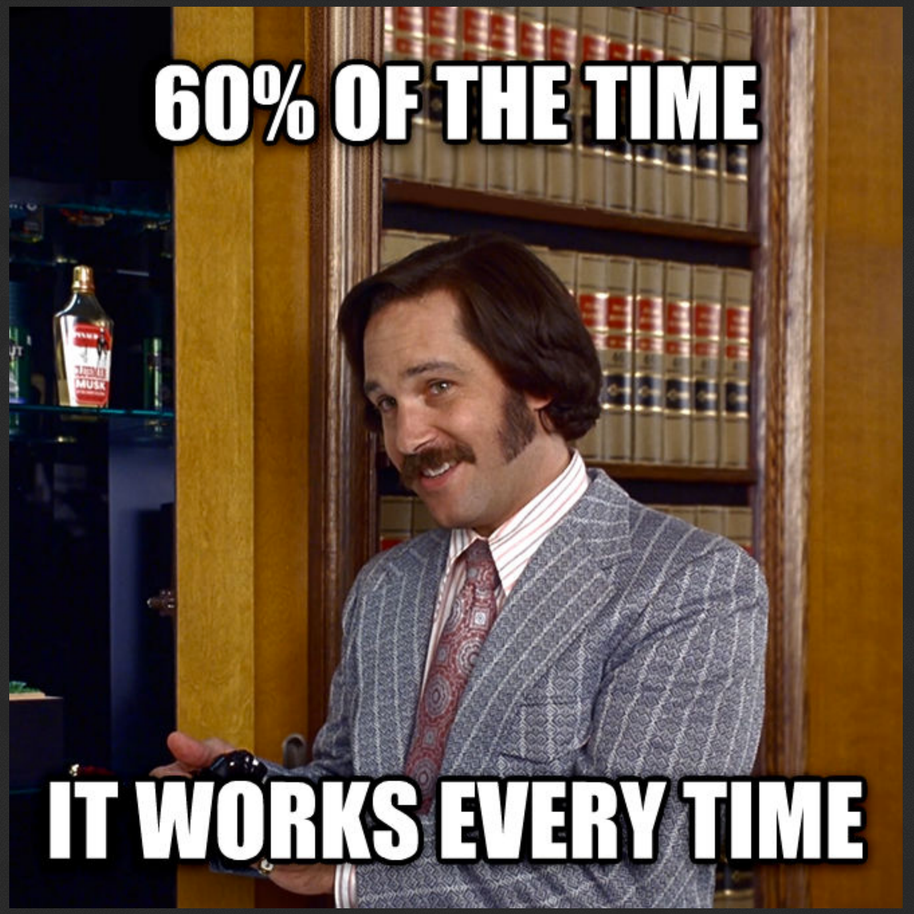
Soren Kierkegaard

TERRAFORM IS COOL

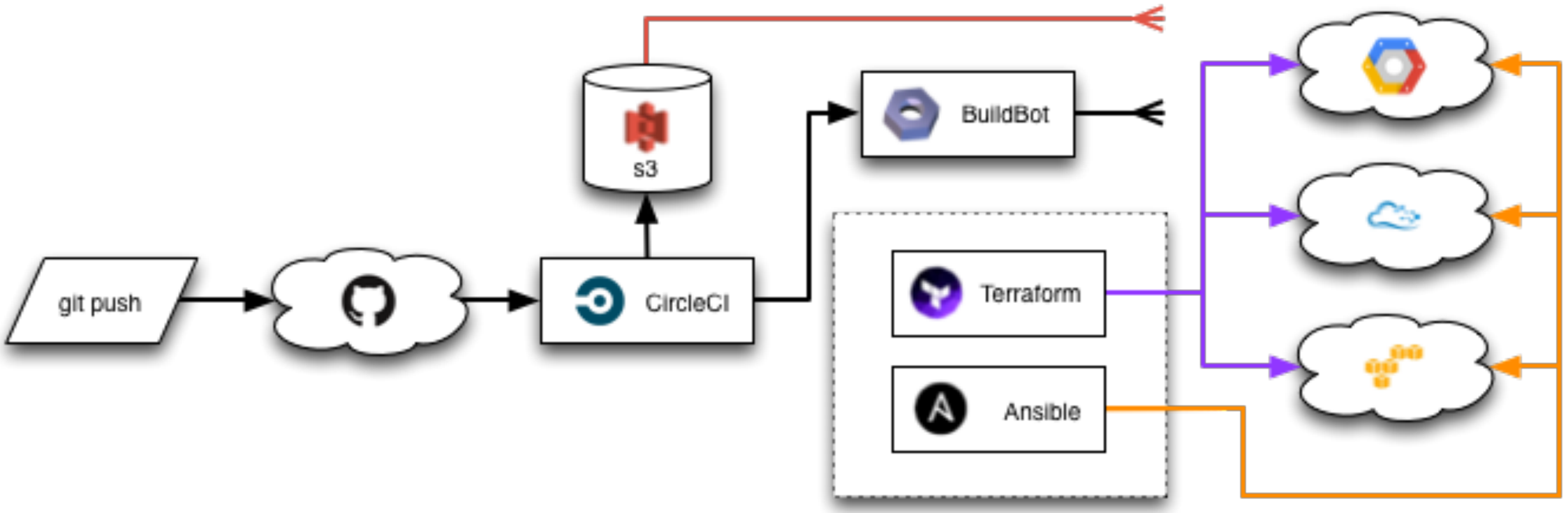
Me

OVERVIEW

- What is Terraform
- Code Sample & Compare to Cloud Formation
- Create AWS Resources
  - S3 Bucket
  - EC2 Instance in VPC
- Compose with Digital Ocean Resources
  - Create Droplet
  - Grant S3 Access to Droplet


60% OF THE TIME
IT WORKS EVERY TIME

DEVOPS @ YHAT



WHAT IS TERRAFORM

- Infrastructure as code
- A tool to manage virtual server life cycles (AWS, VMWare, etc)
- A tool to manage supporting services (DNS, Email)
- A tool to manage system servies (MySQL, PostgreSQL)
- Configuration files can be HCL or JSON
- Created by Hashicorp (Vagrant et al.)
- Written in Go

WHAT DOES TERRAFORM LOOK LIKE?

```
provider "aws" {
    // Set AWS_ACCESS_KEY_ID &
    // AWS_SECRET_ACCESS_KEY env vars
    region = "us-west-2"
}

resource "aws_s3_bucket" "my-cool-bucket" {
    bucket = "my-cool-bucket"
    acl = "public-read"
}
```

TERRAFORM

```
provider "aws" {
    // Set AWS_ACCESS_KEY_ID &
    // AWS_SECRET_ACCESS_KEY env vars
    region = "us-west-2"
}

resource "aws_s3_bucket" "my-cool-bucket" {
    bucket = "my-cool-bucket"
    acl = "public-read"
}
```

CLOUD FORMATION

```
{
   "Resources" : {
      "my-cool-bucket" : {
         "Type" : "AWS::S3::Bucket",
         "Properties" : {
            "AccessControl" : "PublicRead"
         }
      }
   }
}
```

TERRAFORM

```
provider "aws" {
    // Set AWS_ACCESS_KEY_ID &
    // AWS_SECRET_ACCESS_KEY env vars
    region = "us-west-2"
}

resource "aws_s3_bucket" "my-cool-bucket" {
    bucket = "my-cool-bucket"
    acl = "public-read"
}
```

The name to reference within terraform files

The name of the bucket

CLOUD FORMATION

```
{
    "Resources" : {
        "my-cool-bucket" : {
            "Type" : "AWS::S3::Bucket",
            "Properties" : {
                "AccessControl" : "PublicRead"
            }
        }
    }
}
```

Generated name: stackname-my-cool-bucket-abc123id

LET'S TERRAFORM AN S3 BUCKET

MAIN.TF

```
# Configure aws with a default region
provider "aws" {
  region = "us-east-1"
}

# Create a demo s3 bucket
resource "aws_s3_bucket" "chadev_tf_demo" {
    bucket = "chadev-tf-demo"

    tags {
        Name = "Terraform demo for ChaDev"
        Environment = "Prod"
        TF = "yes"
    }
}
```

TERRAFORM PLAN

```
+ aws_s3_bucket.chadev_tf_demo
    acl:                  "" => "private"
    arn:                  "" => "<computed>"
    bucket:               "" => "chadev-tf-demo"
    force_destroy:        "" => "0"
    hosted_zone_id:       "" => "<computed>"
    region:               "" => "<computed>"
    tags.#:               "" => "3"
    tags.Environment:     "" => "Prod"
    tags.Name:            "" => "Terraform demo for ChaDev"
    tags.TF:              "" => "yes"
    website_domain:       "" => "<computed>"
    website_endpoint:     "" => "<computed>"

Plan: 1 to add, 0 to change, 0 to destroy.
```

TERRAFORM APPLY

```
aws_s3_bucket.chadev_tf_demo: Creating...
  acl:                  "" => "private"
  arn:                  "" => "<computed>"
  bucket:               "" => "chadev-tf-demo"
  force_destroy:        "" => "0"
  hosted_zone_id:       "" => "<computed>"
  region:               "" => "<computed>"
  tags.#:               "" => "3"
  tags.Environment:     "" => "Prod"
  tags.Name:            "" => "Terraform demo for ChaDev"
  tags.TF:              "" => "yes"
  website_domain:       "" => "<computed>"
  website_endpoint:     "" => "<computed>"
aws_s3_bucket.chadev_tf_demo: Creation complete

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

State path: terraform.tfstate
```

TERRAFORM STATE


```
"version": 1,
"serial": 1,
"modules": [
    {
        "path": [
            "root"
        ],
        "outputs": {},
        "resources": {
            "aws_s3_bucket.chadev_tf_demo": {
                "type": "aws_s3_bucket",
                "primary": {
                    "id": "chadev-tf-demo",
                    "attributes": {
                        "acl": "private",
                        "arn": "arn:aws:s3:::chadev-tf-demo",
                        "bucket": "chadev-tf-demo",
                        "cors_rule.#": "0",
                        "force_destroy": "false",
                        "hosted_zone_id": "Z3AQBSTGFYJSTF",
                        "id": "chadev-tf-demo",
                        "policy": "",
                        "region": "us-east-1",
                        "tags.#": "3",
                        "tags.Environment": "Prod",
                        "tags.Name": "Terraform demo for ChaDev",
                        "tags.TF": "yes",
                        "website.#": "0"
```

TERRAFORM STATE

- JSON
- Updated when Terraform runs
- Refreshed before an operation takes place
- Backwards compatible between TF versions

TANGENT: USING REMOTE STATE AS A DEPENDENCY

```
resource "terraform_remote_state" "core" {
    backend = "s3"
    config {
        bucket = "a-tf-state-bucket"
        key = "hosted/core.json"
        region = "us-east-1"
    }
}

resource "aws_route53_record" "my-awesome-subdomain.yhat.com" {
    zone_id = "${terraform_remote_state.core.output.yhat-zone-id}"
    name = "my-awesome-subdomain.yhat.com"
    type = "A"
    ttl = "60"
    records = [
        "${aws_eip.my-elastic-ip.public_ip}"
    ]
}
```

LET'S TERRAFORM A SERVER

CREATING A SERVER

```
# Create a key pair for our servers
resource "aws_key_pair" "chadev_demo" {
    key_name = "chadev_demo_key"
    public_key = "ssh-rsa ..."
}

# Create a demo server
resource "aws_instance" "chadev_demo_server" {
    # CentOS 7 (x86_64) with Updates HVM
    ami = "ami-6d1c2007"
    instance_type = "t2.medium"
    key_name = "${aws_key_pair.chadev_demo.key_name}"
    tags {
        Name = "ChaDev Demo Server"
        TF = "yes"
    }
}
```

CREATING A SERVER

```
# Create a key pair for our servers
resource "aws_key_pair" "chadev_demo" {
    key_name = "chadev_demo_key"
    public_key = "ssh-rsa ..."
}

# Create a demo server
resource "aws_instance" "chadev_demo_server" {
    # CentOS 7 (x86_64) with Updates HVM
    ami = "ami-6d1c2007"
    instance_type = "t2.medium"
    key_name = "${aws_key_pair.chadev_demo.key_name}"
    tags {
        Name = "ChaDev Demo Server"
        TF = "yes"
    }
}
```

Get the name of the key by referencing the output of the aws_key_pair.
Notice we're using the variable name we define.

TERRAFORM PLAN

```
+ aws_instance.chadev_demo_server
...

+ aws_key_pair.chadev_demo
...

Plan: 2 to add, 0 to change, 0 to destroy.
```

CREATING A SERVER

```
Error applying plan:

1 error(s) occurred:

* aws_instance.chadev_demo_server: Error launching source instance: VPCResourceNotSpecified: The specified instance type can only be used in a VPC. A subnet ID or network interface ID is required to carry out the request.
  status code: 400, request id:
```

CREATING A SERVER

```
Terraform does not automatically rollback in the face of errors.
Instead, your Terraform state file has been partially updated with
any resources that successfully completed. Please address the error
above and apply again to incrementally change your infrastructure.
```

TERRAFORM PLAN

```
Plan: 1 to add, 0 to change, 0 to destroy.
```

LET'S TERRAFORM A VPC

PARTS OF A VPC

- VPC definition (CIDR block)
- Subnets
- Internet gateway to allow our subnet to route to the internet
- Routing table to define how traffic is routed
- Main route table association to tell the vpc what route table to use by default

THE VPC

```
########
# VPC
########

resource "aws_vpc" "chadev_demo" {
    cidr_block = "10.20.0.0/16"
    enable_dns_hostnames = true

    tags {
        Name = "ChaDev Demo VPC"
        Env = "Prod"
        TF = "yes"
    }
}
```

THE SUBNET

```
# Create a subnet for our instances
resource "aws_subnet" "chadev_demo_1a" {
    vpc_id = "${aws_vpc.chadev_demo.id}"
    cidr_block = "10.20.0.0/24"
    availability_zone = "us-east-1a"
    map_public_ip_on_launch = true

    tags {
        Name = "ChaDev Subnet AZ 1a"
        Env = "Prod"
        TF = "yes"
    }
}
```

THE INTERNET GATEWAY

```
# Create an internet gateway so we can talk to the world
resource "aws_internet_gateway" "main" {
    vpc_id = "${aws_vpc.chadev_demo.id}"

    tags {
        Name = "ChaDev VPC IGW"
        Env = "Prod"
        TF = "yes"
    }
}
```

THE ROUTE TABLE

```
# Replace the default routing table with one explicitly defined here
resource "aws_route_table" "chadev_demo" {
    vpc_id = "${aws_vpc.chadev_demo.id}"

    # Send all traffic not matched by other routing rules to the internet
    route {
        cidr_block = "0.0.0.0/0"
        gateway_id = "${aws_internet_gateway.main.id}"
    }

    tags {
        Name = "ChaDev Main RTB"
        Env = "Prod"
        TF = "yes"
    }
}
```

THE ROUTE TABLE ASSOCIATION

```
# Assign our route table as the default route table
resource "aws_main_route_table_association" "main" {
    vpc_id = "${aws_vpc.yhat-tf-demo.id}"
    route_table_id = "${aws_route_table.yhat-tf-default.id}"
}
```

HELLO TERRAFORM

MAIN.TF

```
# Configure aws with a default region
provider "aws" {
  region = "us-east-1"
}

# Create a demo s3 bucket
resource "aws_s3_bucket" "chadev_tf_demo" {
    bucket = "chadev-tf-demo"

    tags {
        Name = "Terraform demo for ChaDev"
        Environment = "Prod"
        TF = "yes"
    }
}

# Create a key pair for our servers
resource "aws_key_pair" "chadev_demo" {
    key_name = "chadev_demo_key"
    public_key = "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDGrwdl2hRk2JvzfoCgZFegX/qlGyaVZRxkJCleGwKRTjG/RPDRj6lZnpGPRzKmhwcN78qXDmgGWY7cfoApuLC6lFZNissBeCsjDLFmsnyev+7EwZu3Bt0Sq7goFKtoaZ2oUZIjv6TIQ0yj4LKEoBkHAuJgddwvFScoPDt3SJ889NXybMDkXMNsiNB3vHLGmYs9qcC/CC7hlY9Y8oHqJRe+IMfbULp6YAgIM4iLD5/rKjUdnyhUSlipuDeECTVuwHK+9LWIAdT8+ZIJe1mVI/ZyODKkZYDlWPvUrXmVFlMoXeVil7mEe88I9iZtRJWTmzY3oHQHowGkzJ8JRnLmRaF yhat@s-MacBook-Pro.local"
}

# Create a demo server
resource "aws_instance" "chadev_demo_server" {
    # CentOS 7 (x86_64) with Updates HVM
    ami = "ami-6d1c2007"
    instance_type = "t2.micro"
    key_name = "${aws_key_pair.chadev_demo.key_name}"

    tags {
        Name = "ChaDev Demo Server"
        TF = "yes"
    }
}


#######
# VPC #
#######

resource "aws_vpc" "chadev_demo" {
    cidr_block = "10.20.0.0/16"
    enable_dns_hostnames = true

    tags {
        Name = "ChaDev Demo VPC"
        Env = "Prod"
        TF = "yes"
    }
}

# Create a subnet for our instances
resource "aws_subnet" "chadev_demo_1a" {
    vpc_id = "${aws_vpc.chadev_demo.id}"
    cidr_block = "10.20.0.0/24"
    availability_zone = "us-east-1a"
    map_public_ip_on_launch = true

    tags {
        Name = "ChaDev Subnet AZ 1a"
        Env = "Prod"
        TF = "yes"
    }
}

# Create an internet gateway so we can talk to the world
resource "aws_internet_gateway" "main" {
    vpc_id = "${aws_vpc.chadev_demo.id}"

    tags {
        Name = "ChaDev VPC IGW"
        Env = "Prod"
        TF = "yes"
    }
}

# Replace the default routing table with one explicitly defined here
resource "aws_route_table" "chadev_demo" {
    vpc_id = "${aws_vpc.chadev_demo.id}"

    # Send all traffic not matched by other routing rules to the internet
    route {
        cidr_block = "0.0.0.0/0"
        gateway_id = "${aws_internet_gateway.main.id}"
    }

    tags {
        Name = "ChaDev Main RTB"
        Env = "Prod"
        TF = "yes"
    }
}

# Assign our route table as the default route table
resource "aws_main_route_table_association" "main" {
    vpc_id = "${aws_vpc.yhat-tf-demo.id}"
    route_table_id = "${aws_route_table.yhat-tf-default.id}"
}
```

ORGANIZING TERRAFORM FILES

- Terraform looks for all files ending in .tf
- I personally prefix the files with the provider where applicable
- Let's put the vpc in aws.vpc.tf
- Let's put the rest in aws.demo.tf

TERRAFORM APPLY

```
Error applying plan:

1 error(s) occurred:

* aws_instance.chadev_demo_server: Error launching source instance: VPCResourceNotSpecified: The
specified instance type can only be used in a VPC. A subnet ID or network interface ID is required
to carry out the request.
  status code: 400, request id:
```

SPECIFY SUBNET AND SECURITY GROUPS

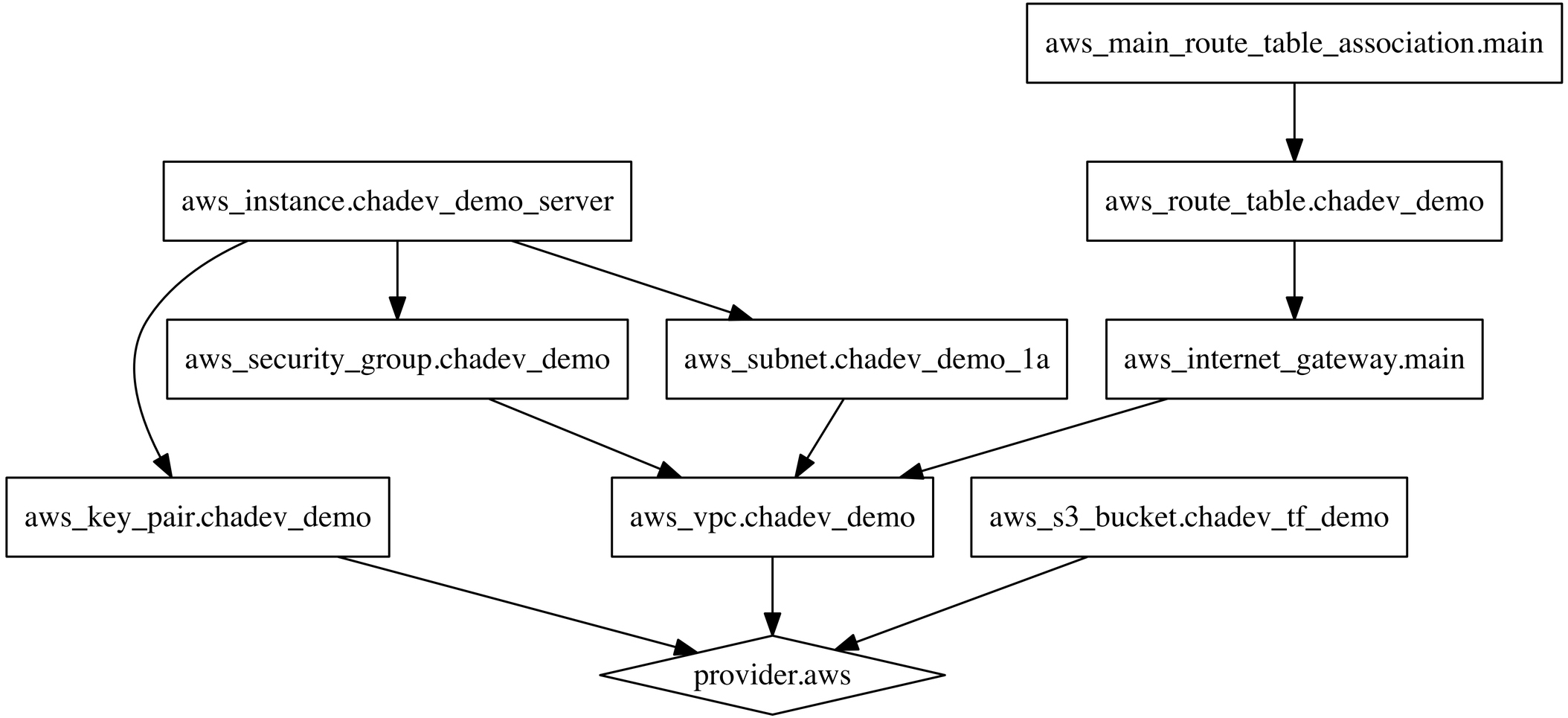
```
vpc_security_group_ids = [
    "${aws_security_group.chadev_demo.id}",
]

subnet_id = "${aws_subnet.chadev_demo_1a.id}"
```

LAUNCH & REVIEW

TERRAFORM WILL GRAPH DEPENDENCIES

- terraform graph | dot -Tpng > graph.png

TERRAFORM GRAPH | DOT -TPDF > GRAPH.PDF



COMPOSE

DigitalOcean

DigitalOcean is currently under maintenance and will be back shortly!

CREATE DIGITAL OCEAN SSH KEY & DROPLET

```
# Create a new SSH key
resource "digitalocean_ssh_key" "chadev_demo" {
    name = "ChaDev Demo"
    public_key = "${file("/Users/yhat/.ssh/id_rsa.pub")}"
}

# Create a new Web droplet in the nyc2 region
resource "digitalocean_droplet" "web" {
    image = "ubuntu-14-04-x64"
    name = "chadev-demo"
    region = "nyc2"
    size = "512mb"
}
```

CREATE DIGITAL OCEAN SSH KEY & DROPLET

```
# Create a new SSH key
resource "digitalocean_ssh_key" "chadev_demo" {
    name = "ChaDev Demo"
    public_key = "${file("/Users/yhat/.ssh/id_rsa.pub")}"
}

# Create a new Web droplet in the nyc2 region
resource "digitalocean_droplet" "web" {
    image = "ubuntu-14-04-x64"
    name = "chadev-demo"
    region = "nyc2"
    size = "512mb"
}
```

CREATE DIGITAL OCEAN SSH KEY & DROPLET

```
# Set the variable value in *.tfvars file
# or using -var="do_token=..." CLI option
variable "do_token" {}

# Configure the DigitalOcean Provider
provider "digitalocean" {
    token = "${var.do_token}"
}
```

TERRAFORM.TFVARS

```
# Set the variable value in *.tfvars file
# or using -var="do_token=..." CLI option
variable "do_token" {}

# Configure the DigitalOcean Provider
provider "digitalocean" {
    token = "${var.do_token}"
}
```

AWS.DEMO.TF

```
# Upload our html file to our bucket
resource "aws_s3_bucket_object" "object" {
    bucket = "${aws_s3_bucket.chadev_tf_demo.id}"
    key = "index.html"
    source = "./index.html"
}
```

AWS.DEMO.TF

```
# Create a demo s3 bucket
resource "aws_s3_bucket" "chadev_tf_demo" {
    bucket = "chadev-tf-demo-1"

    website {
        index_document = "index.html"
    }

    tags {
        Name = "Terraform demo for ChaDev"
        Environment = "Prod"
        TF = "yes"
    }
}
```

CURL THE ENDPOINT

$ curl chadev-tf-demo-1.s3.amazonaws.com/

<?xml version="1.0" encoding="UTF-8"?>

<Error><Code>AccessDenied</Code><Message>Access Denied</Message><RequestId>EC6CC75BE6AA9040</RequestId><HostId>iIBLHlo4RWzMFE4Sh3AVXv0qD2umr7G7qpI1GaMbCnFmmQv/ZP4KeoeIpWqwC8II</HostId></Error>

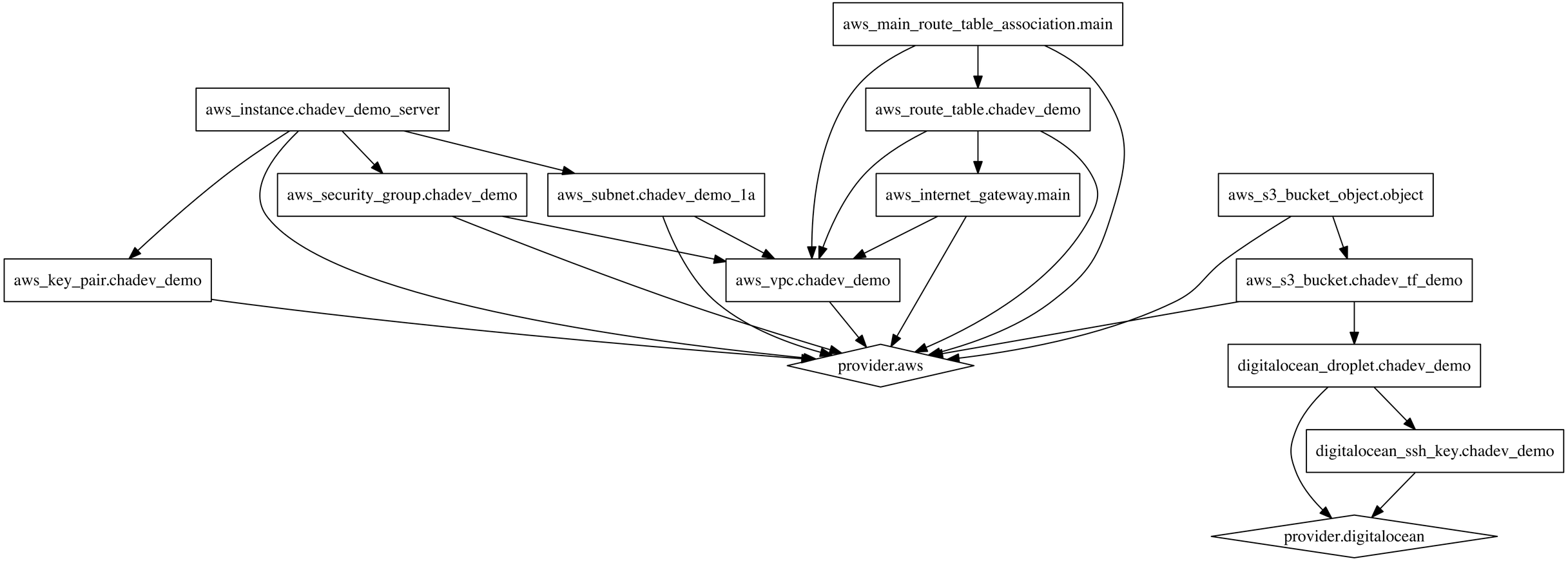
ALLOW ACCESS FROM DIGITAL OCEAN

```
policy = <<EOF
{
   "Version": "2008-10-17",
   "Id": "S3PolicyId1",
   "Statement": [
      {
         "Sid": "IPAllow",
         "Effect": "Allow",
         "Principal": {
            "AWS": "*"
         },
         "Action": "s3:*",
         "Resource": [
            "arn:aws:s3:::chadev-tf-demo-1/*",
            "arn:aws:s3:::chadev-tf-demo-1"
         ],
         "Condition": {
            "IpAddress": {
               "aws:SourceIp": "${digitalocean_droplet.chadev_demo.ipv4_address}/32"
            }
         }
      }
   ]
}
EOF
```

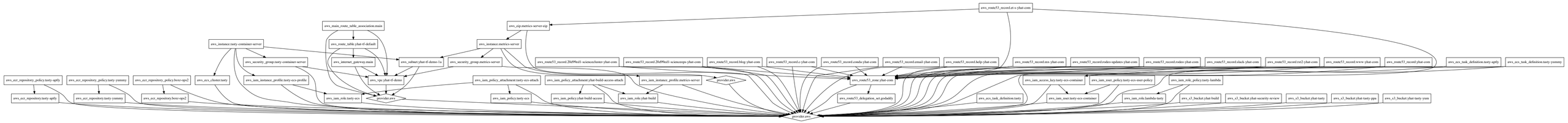
ALLOW ACCESS FROM DIGITAL OCEAN

```
"Condition": {
  "IpAddress": {
    "aws:SourceIp": "${digitalocean_droplet.chadev_demo.ipv4_address}/32"
  }
}
```

HELLO TERRAFORM



HELLO TERRAFORM



THERE'S SO MUCH MORE BUT WE'RE OUT OF TIME

THANK YOU