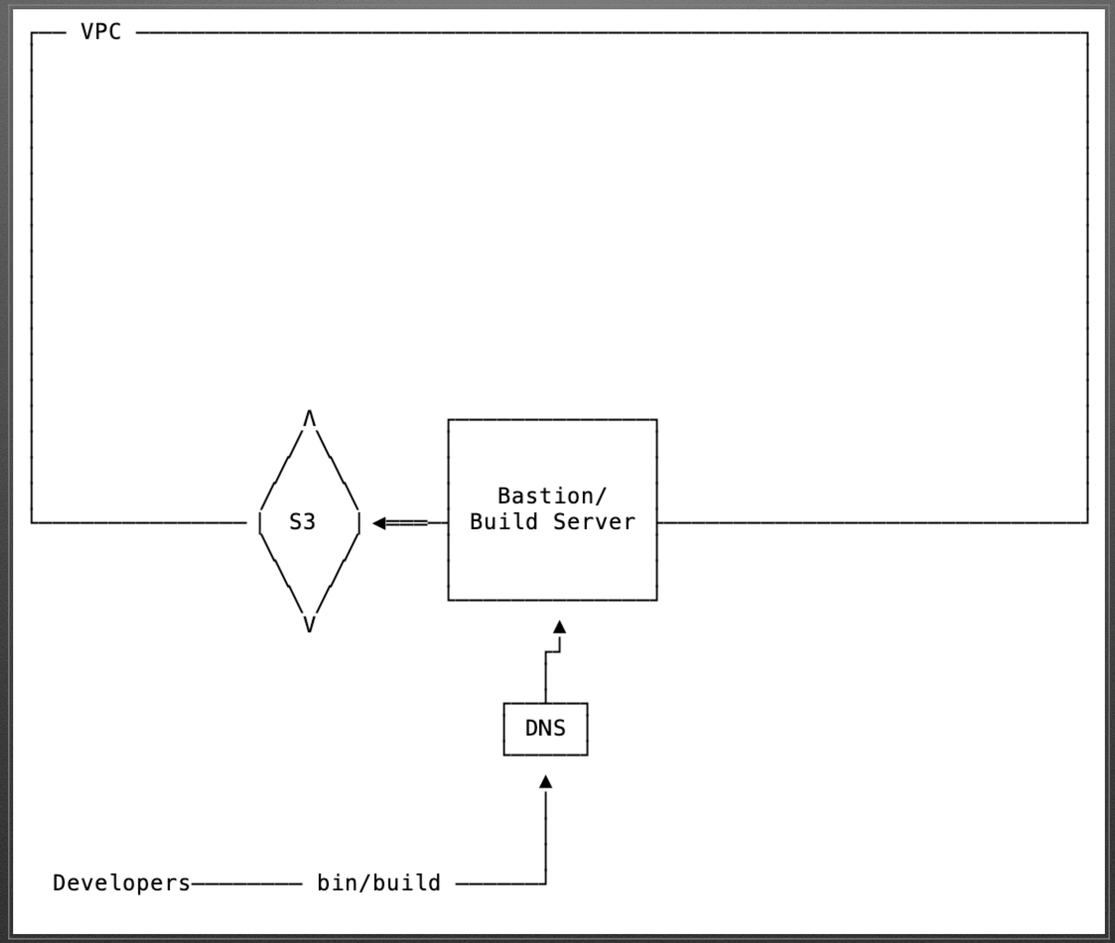
Building in the Cloud

Elixir in the Jungle

Full Architecture

Full Architecture

A Subset of the Architecture



Our Plan

- Using Terraform with AWS
- A VPC
- The Bastion
- cloud-init
- Exercise: `bin/build`

Using Terraform with AWS

Amazon Web Services

What is Terraform?

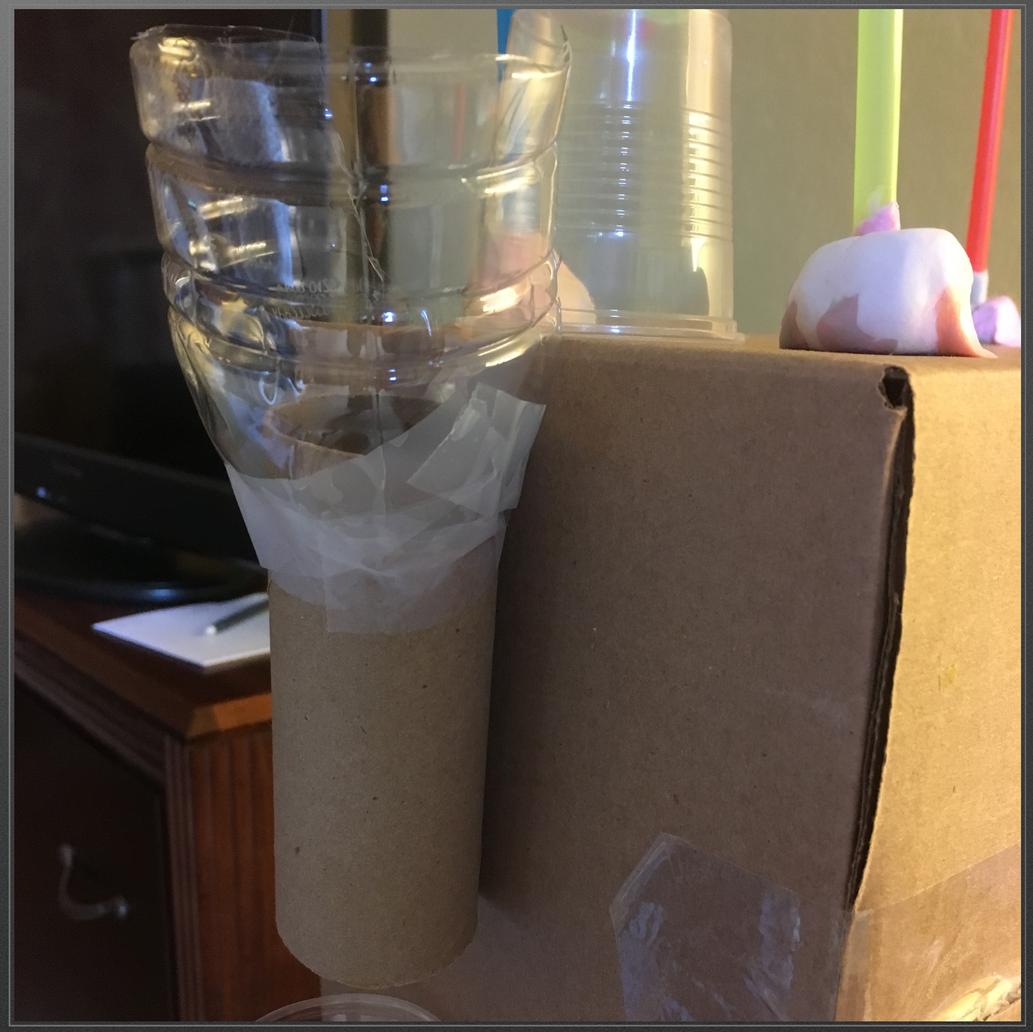
- An open source tool
- A declarative language for your ops architecture
- "Infrastructure as Code"
- Works with multiple cloud providers

What is AWS?

- A collection of services
- Provides all manner of cloud computing resources
- We'll focus primarily on file storage, network setup, and server provisioning

Service: S3 (Simple Storage Service)

- Stores files and other data
- Provides controls for who can access your data
- Scales tremendously
- Useful for storing many different things: Terraform state, build artifacts, environment configurations, files uploaded by visitors, and more



```
resource "aws_s3_bucket" "build_bucket" {
  bucket = "elixir-in-the-jungle-${terraform.workspace}-builds"
  acl = "private"

  tags = {
    Name = "elixir-in-the-jungle"
    Environment = "${terraform.workspace}"
  }

  versioning { enabled = true }

  lifecycle_rule {
    enabled = true

    noncurrent_version_expiration { days = 30 }
  }
}

resource "aws_s3_bucket_public_access_block" "build_bucket" {
  bucket = "${aws_s3_bucket.build_bucket.id}"

  ignore_public_acls = true
  restrict_public_buckets = true
  block_public_acls = true
  block_public_policy = true
}
```

Terraform Syntax

```
resource "aws_s3_bucket" "build_bucket" {
  bucket = "elixir-in-the-jungle-${terraform.workspace}-builds"
  acl = "private"

  tags = {
    Name = "elixir-in-the-jungle"
    Environment = "${terraform.workspace}"
  }

  versioning { enabled = true }

  lifecycle_rule {
    enabled = true

    noncurrent_version_expiration { days = 30 }
  }
}

resource "aws_s3_bucket_public_access_block" "build_bucket" {
  bucket = "${aws_s3_bucket.build_bucket.id}"

  ignore_public_acls = true
  restrict_public_buckets = true
  block_public_acls = true
  block_public_policy = true
}
```

Terraform Syntax

```
resource "aws_s3_bucket" "build_bucket" {
  bucket = "elixir-in-the-jungle-${terraform.workspace}-builds"
  acl = "private"

  tags = {
    Name = "elixir-in-the-jungle"
    Environment = "${terraform.workspace}"
  }

  versioning { enabled = true }

  lifecycle_rule {
    enabled = true

    noncurrent_version_expiration { days = 30 }
  }
}

resource "aws_s3_bucket_public_access_block" "build_bucket" {
  bucket = "${aws_s3_bucket.build_bucket.id}"

  ignore_public_acls = true
  restrict_public_buckets = true
  block_public_acls = true
  block_public_policy = true
}
```

Terraform Syntax

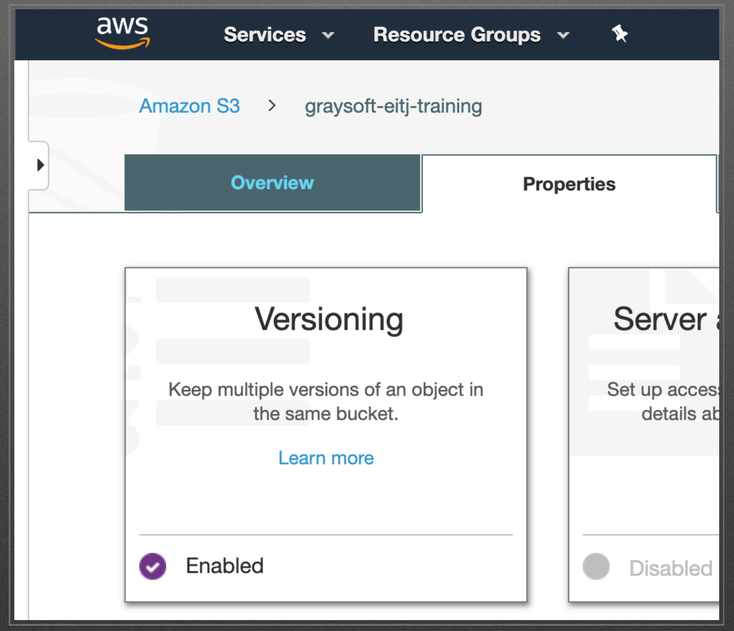
```
  # aws_vpc.elixir-in-the-jungle will be created
  + resource "aws_vpc" "elixir-in-the-jungle" {
      + arn                              = (known after apply)
      + assign_generated_ipv6_cidr_block = false
      + cidr_block                       = "10.20.0.0/16"
      + default_network_acl_id           = (known after apply)
      + default_route_table_id           = (known after apply)
      + default_security_group_id        = (known after apply)
      + dhcp_options_id                  = (known after apply)
      + enable_classiclink               = (known after apply)
      + enable_classiclink_dns_support   = (known after apply)
      + enable_dns_hostnames             = (known after apply)
      + enable_dns_support               = true
      + id                               = (known after apply)
      + instance_tenancy                 = "default"
      + ipv6_association_id              = (known after apply)
      + ipv6_cidr_block                  = (known after apply)
      + main_route_table_id              = (known after apply)
      + owner_id                         = (known after apply)
      + tags                             = {
          + "Environment" = "example"
          + "Name"        = "elixir-in-the-jungle"
        }
    }

Plan: 21 to add, 0 to change, 0 to destroy.
```

Terraform in Action

Terraform State

Local Verses Remote State

Bootstrapping Remote State



```
provider "aws" {
  region = "${var.region}"
}

terraform {
  backend "s3" {
    bucket = "elixir-in-the-jungle-b"
    key = "vpc_state"
    region = "us-east-2"
  }
}
```

Configure Remote State

Name Your Bucket

A VPC

Virtual Private Cloud

What is a VPC?

- A logically isolated chunk of Amazon's cloud
- A collection of subnets, routing tables, and network gateways
- In short: a collection of rules about which cloud assets can interact with other assets and/or the public Internet

```
# The top level VPC Resource
# CIDR blocks can be selected based on the Teraform workspace you're using.
resource "aws_vpc" "elixir-in-the-jungle" {
  cidr_block = "${lookup(var.vpc_cidr_blocks, terraform.workspace)}"

  tags = {
    Name = "elixir-in-the-jungle"
    Environment = "${terraform.workspace}"
  }
}

# This defines the public subnet, and is used to contain the load balancer,
# and the Bastion server, both of which need to be accessed from outside
# the VPC.
resource "aws_subnet" "public" {
  availability_zone = "${var.region}a"
  cidr_block = "${cidrsubnet(aws_vpc.elixir-in-the-jungle.cidr_block, 8, 12)}"
  map_public_ip_on_launch = true
  vpc_id = "${aws_vpc.elixir-in-the-jungle.id}"

  tags = {
    Name = "elixir-in-the-jungle"
    Environment = "${terraform.workspace}"
  }
}
```

A Public Subnet

Our Security Strategy

- White list what you allow
- Allow the minimum you can function with
- Keep as much traffic on private subnets as possible
- Assign IAM (Identity and Access Management) roles to instances with just the permissions they require
- The bastion is your front line of defense

The Bastion

Our Build Server

Service: EC2 (Elastic Compute Cloud)

- A Web API providing cloud servers
- Cores and memory can be configured to meet your needs
- Servers can be launched with many different flavors of Linux or Windows
- Very useful for dynamically adding servers, to respond to load for example

```
resource "aws_instance" "bastion" {
  ami = "${data.aws_ami.ubuntu.id}"
  instance_type  = "${var.instance_size}"
  vpc_security_group_ids = ["${aws_security_group.bastion.id}"]
  subnet_id = "${aws_subnet.public.id}"
  iam_instance_profile = "${aws_iam_instance_profile.bastion_profile.name}"
  associate_public_ip_address = true
  user_data = "${data.template_cloudinit_config.bastion.rendered}"

  tags = {
    Name = "elixir-in-the-jungle"
    Environment = "${terraform.workspace}"
  }
}
```

An Instance

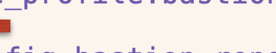
Our Bastion/Build Server

```
resource "aws_instance" "bastion" {
  ami = "${data.aws_ami.ubuntu.id}"
  instance_type  = "${var.instance_size}"
  vpc_security_group_ids = ["${aws_security_group.bastion.id}"]
  subnet_id = "${aws_subnet.public.id}"
  iam_instance_profile = "${aws_iam_instance_profile.bastion_profile.name}"
  associate_public_ip_address = true
  user_data = "${data.template_cloudinit_config.bastion.rendered}"

  tags = {
    Name = "elixir-in-the-jungle"
    Environment = "${terraform.workspace}"
  }
}
```


An Instance

Our Bastion/Build Server

```
resource "aws_instance" "bastion" {
  ami = "${data.aws_ami.ubuntu.id}"
  instance_type  = "${var.instance_size}"
  vpc_security_group_ids = ["${aws_security_group.bastion.id}"]
  subnet_id = "${aws_subnet.public.id}"
  iam_instance_profile = "${aws_iam_instance_profile.bastion_profile.name}"
  associate_public_ip_address = true
  user_data = "${data.template_cloudinit_config.bastion.rendered}"

  tags = {
    Name = "elixir-in-the-jungle"
    Environment = "${terraform.workspace}"
  }
}
```



An Instance

Our Bastion/Build Server

```
resource "aws_instance" "bastion" {
  ami = "${data.aws_ami.ubuntu.id}"
  instance_type  = "${var.instance_size}"
  vpc_security_group_ids = ["${aws_security_group.bastion.id}"]
  subnet_id = "${aws_subnet.public.id}"
  iam_instance_profile = "${aws_iam_instance_profile.bastion_profile.name}"
  associate_public_ip_address = true
  user_data = "${data.template_cloudinit_config.bastion.rendered}"

  tags = {
    Name = "elixir-in-the-jungle"
    Environment = "${terraform.workspace}"
  }
}
```

An Instance

Our Bastion/Build Server

cloud-init

What is cloud-init?

- A declarative script for Linux server configuration
  - Adds users
  - Installs packages
  - Writes files
  - Runs generic commands
- Can be added to any instance that Terraform creates

```
users:
  - name: paul
    groups: sudo
    sudo: ALL=(ALL) NOPASSWD:ALL
    shell: /bin/bash
    ssh_authorized_keys:
      - ssh-rsa AAAAB3…

  - name: james
    groups: sudo
    sudo: ALL=(ALL) NOPASSWD:ALL
    shell: /bin/bash
    ssh_authorized_keys:
      - ssh-rsa AAAAB3…
```

Add Users

Add Your Key Here

```
apt:
  sources:
    yarn:
      source: deb https://dl.yarnpkg.com/debian/ stable main
      keyid: 1646B01B86E50310

    nodesource:
      source: deb https://deb.nodesource.com/node_10.x $RELEASE main
      # from: https://deb.nodesource.com/gpgkey/nodesource.gpg.key
      keyid: 9FD3B784BC1C6FC31A8A0A1C1655A0AB68576280

    erlang_solutions:
      source: deb http://binaries.erlang-solutions.com/debian $RELEASE contrib
      # from https://packages.erlang-solutions.com/ubuntu/erlang_solutions.asc
      keyid: 434975BD900CCBE4F7EE1B1ED208507CA14F4FCA

packages:
  - awscli
  - build-essential
  - ["esl-erlang", "1:22.0.7-1"]
  - ["elixir", "1.9.1-1"]
  - nodejs
  - yarn

package_upgrade: true
```

Install Packages

```
write_files:
  - path: /etc/profile.d/env_vars.sh
    owner: root:root
    content: |
      export BUILD_BUCKET="${bucket_name}"
      export SLACK_WEBHOOK="${slack_webhook}"

  - path: /usr/bin/notify_slack
    owner: root:root
    permissions: '0755'
    content: |
      #!/bin/bash
      curl -s -X POST -H 'Content-type: application/json' --data "{\"text\":\"$1\"}" $SLACK_WEBHOOK

  - path: /usr/bin/publish_release
    owner: root:root
    permissions: '0755'
    content: |
      #!/bin/bash
      aws s3 cp $1 s3://$BUILD_BUCKET/$2
```

Write Files

```
runcmd:
  - [curl, -X, POST, -H, 'Content-type: application/json',
     --data, '{"text":"Build server setup complete!"}', "${slack_webhook}" ]
```

Run Commands

Exercise: Finish `bin/build`

- Unpack the pushed archive
- Build a release
- Push it to S3