TERRAFORM

@julienvey

founded bywan

Julien Vey

DevOps

OpenStack Contributor

Works with Ansible, Docker, Go

Contributed to the OpenStack Provider for Terraform

@haklop

founded bywan

Éric Bellemon

DevOps

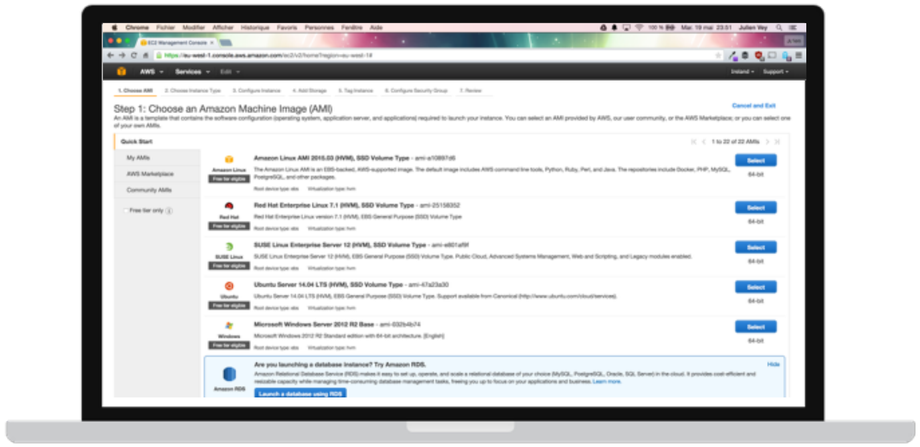
Works with Docker, Go, JavaScript

Contributed to the OpenStack Provider for Terraform

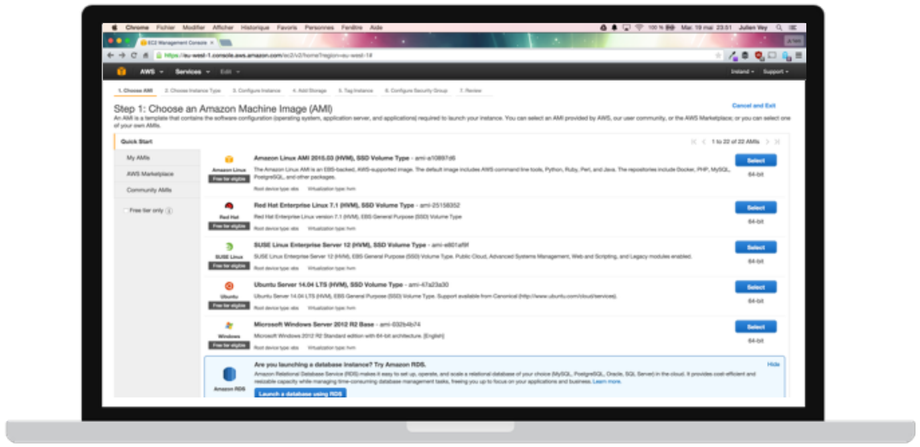
We live in a Cloud era

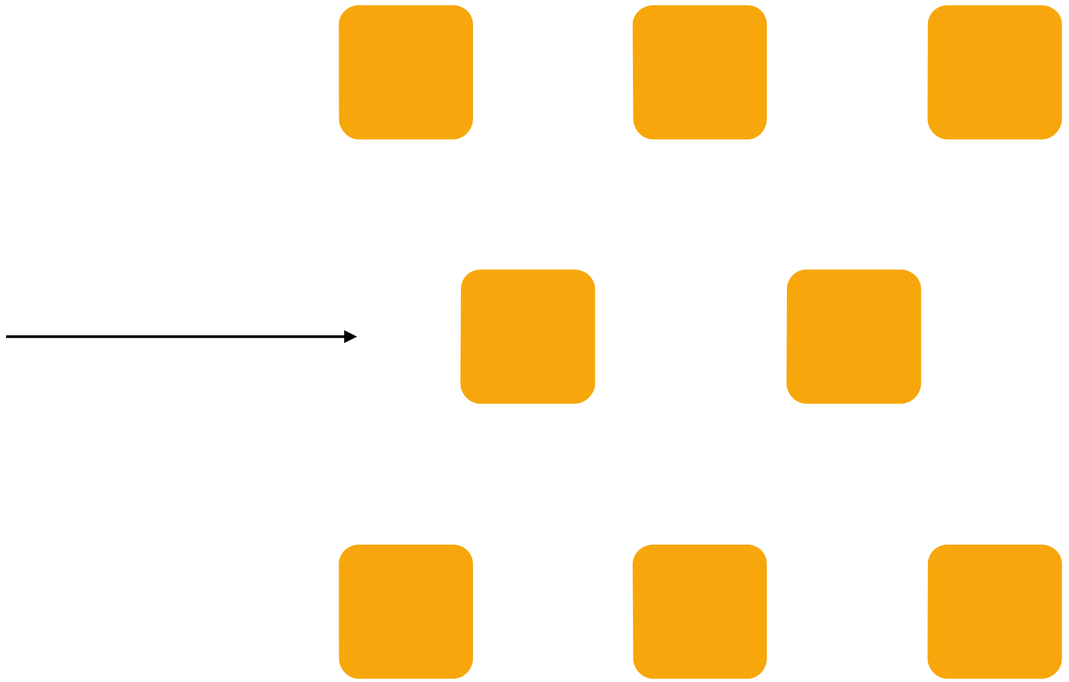
THE PATH OF A STARTUP

Day 1, let's start an EC2 Instance



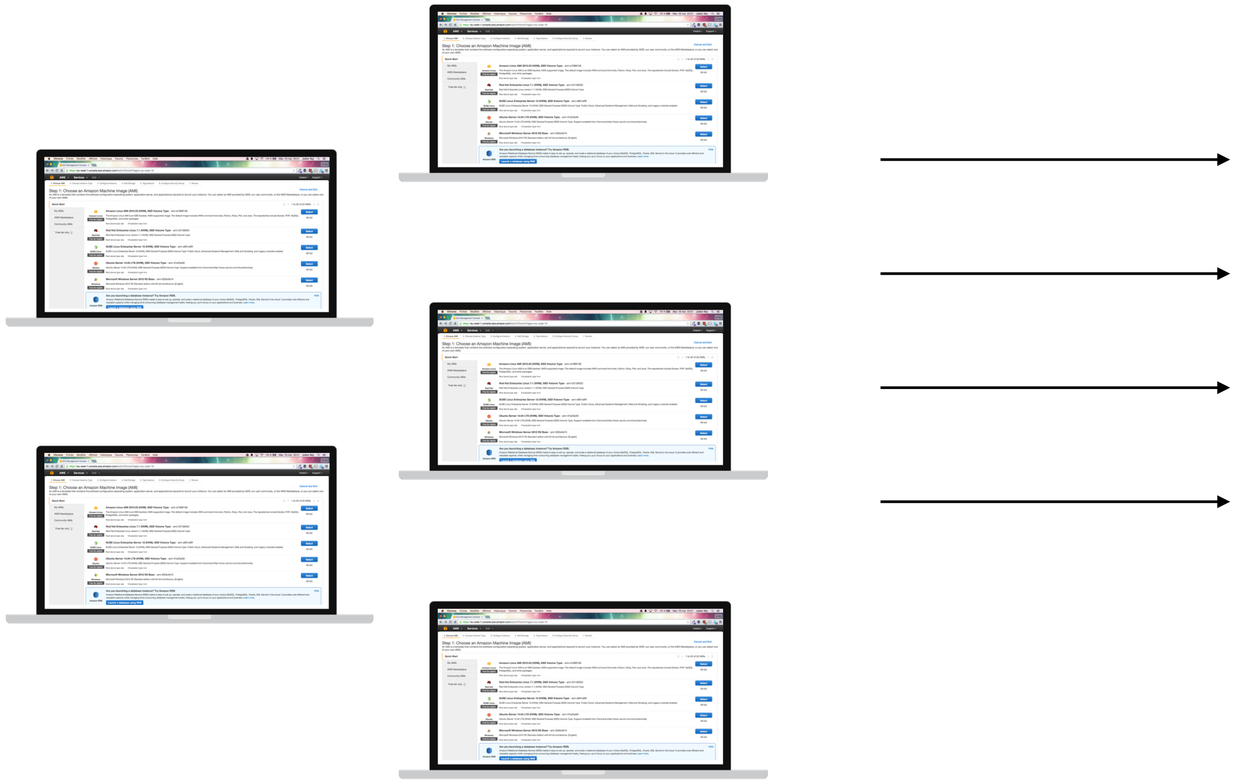
Day 2, let’s start more instances

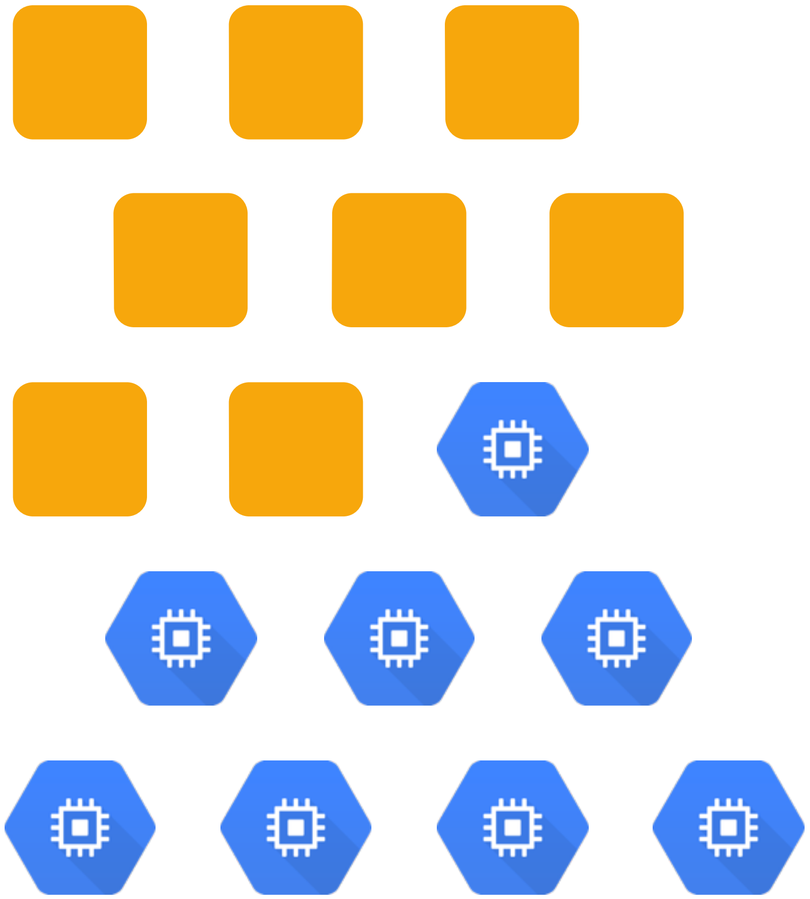




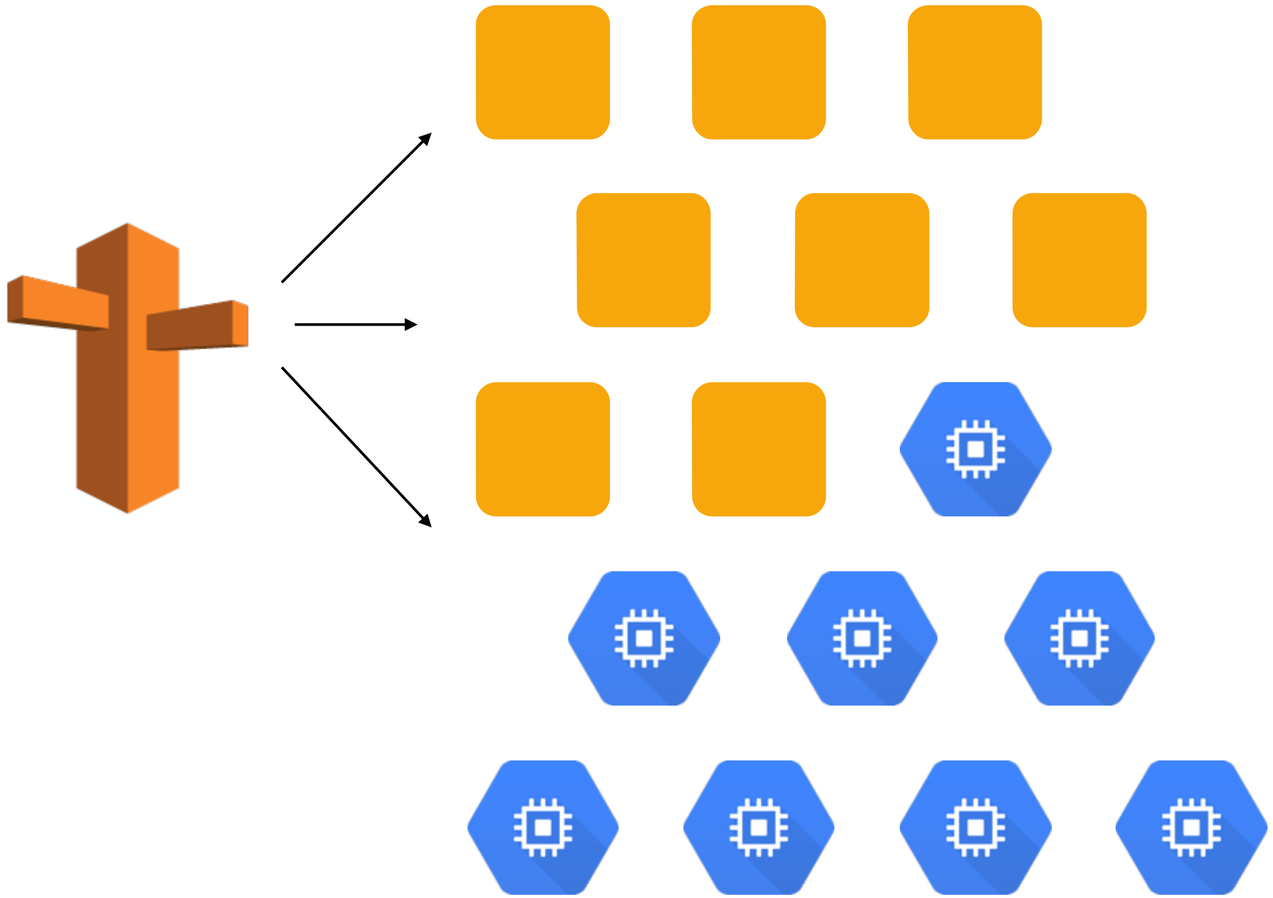
Day 3, let's hire more developers

Day 4, Hey! Google looks great, let's start some GCE instances

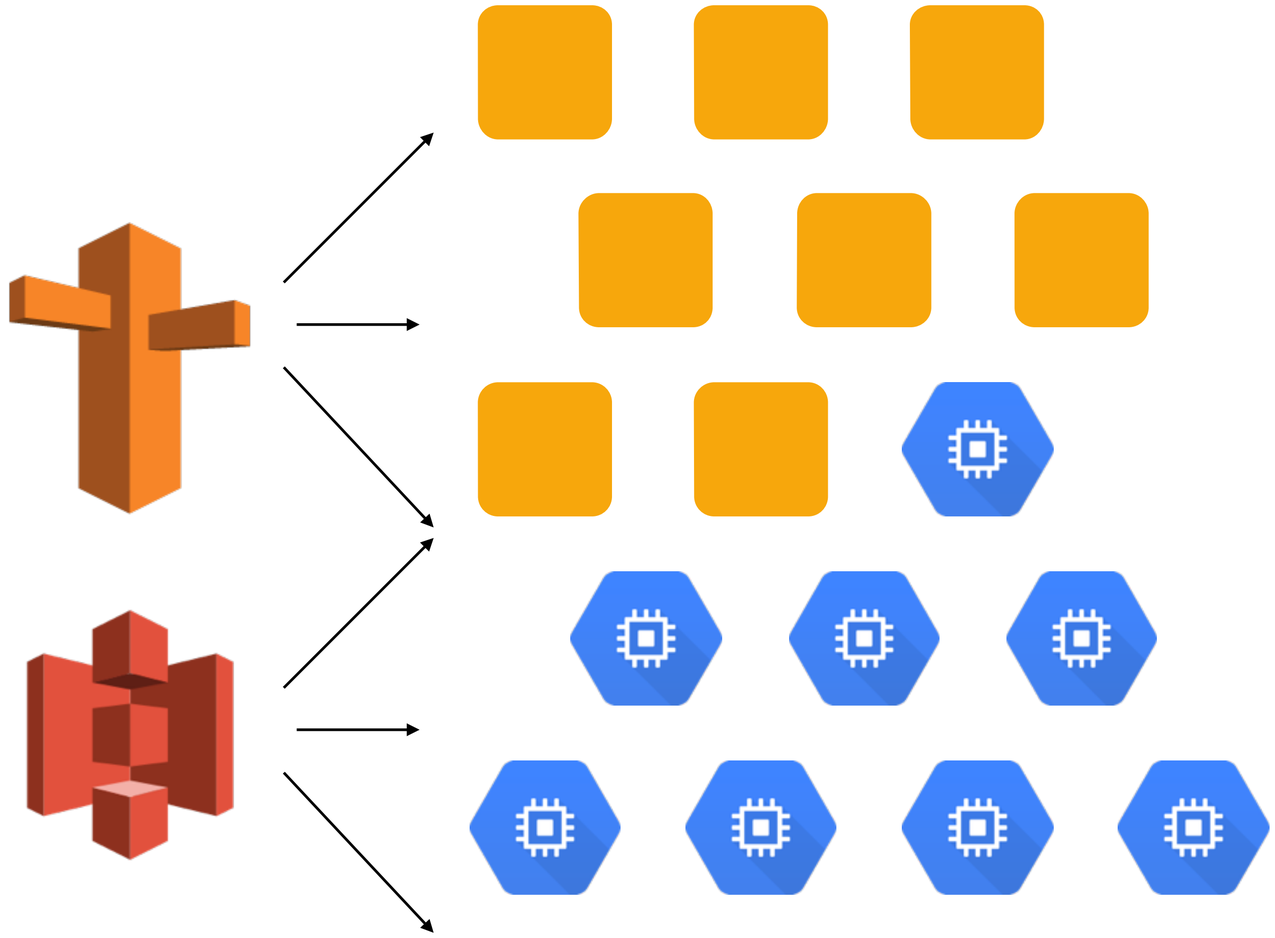




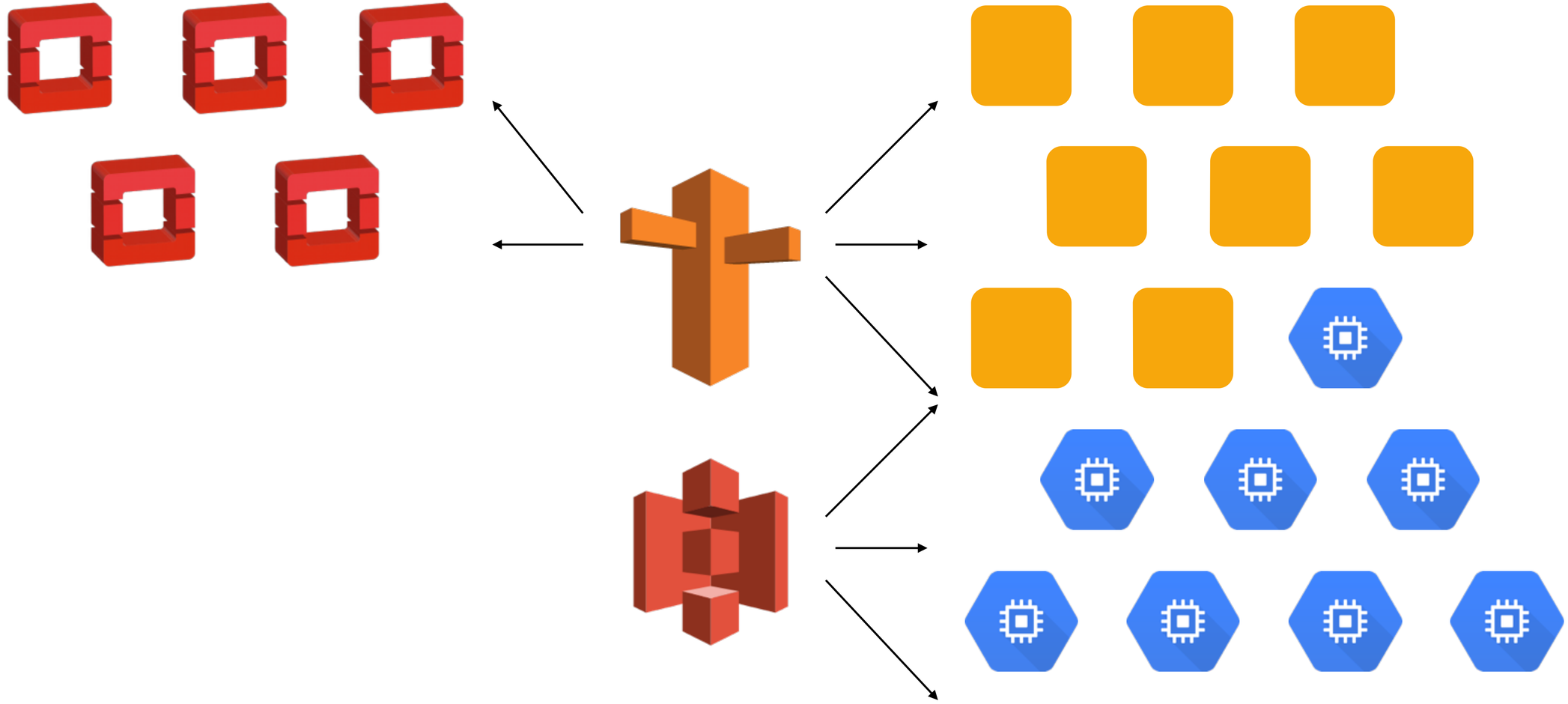
Day 5, let's use a Cloud DNS



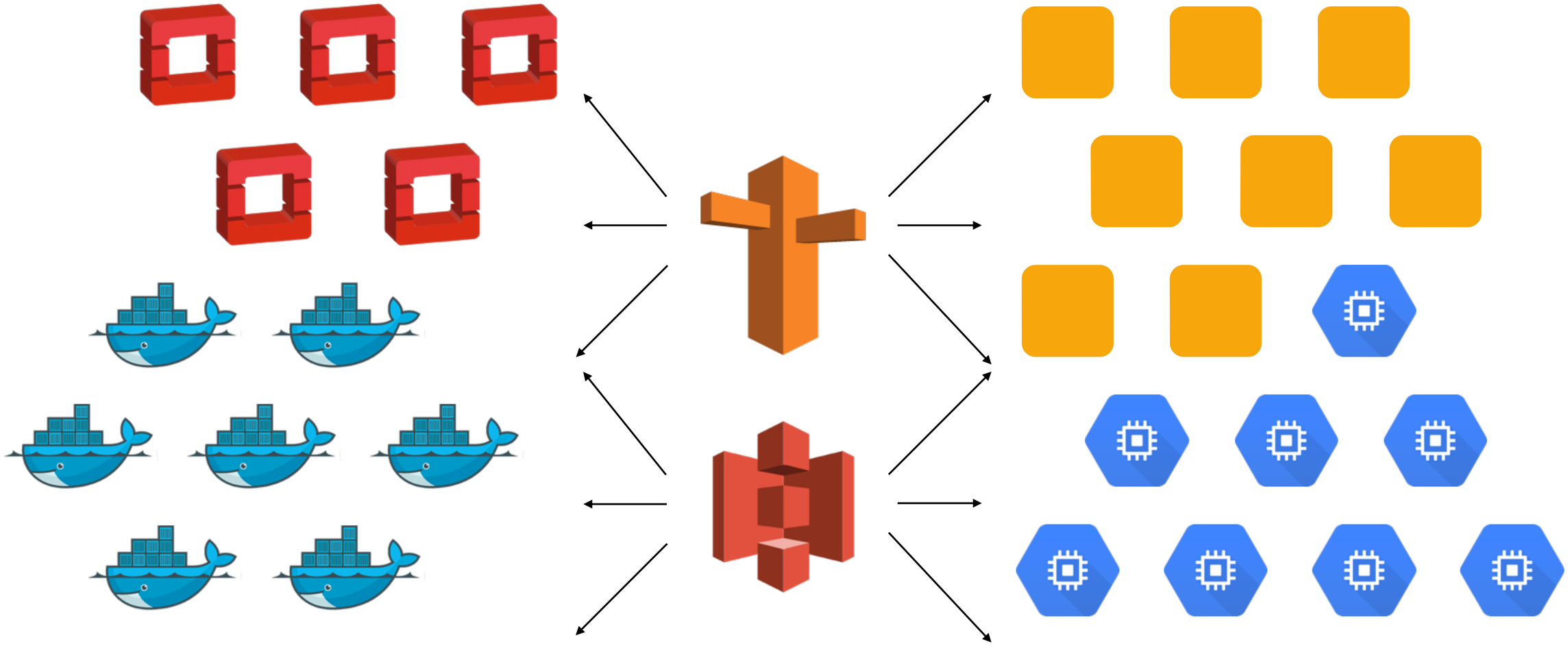
Day 6, let's use a Cloud Storage



Day 7, let's add some OpenStack instances



Day 8, Docker is so cool, let's add some Docker containers



this is how you get from

A BASIC INFRASTRUCTURE

to a

COMPLEX CLOUD INFRASTRUCTURE

and this is

NORMAL !

How to ?

Keep Track of your inventory

Manage changes in your infrastructure

Control the lifecycle of your resources

TERRAFORM

Infrastructure as code

Description of desired state

Knowledge sharing

Version Control and Code reviews

Infrastructure as code

Friendly config

Simple file based configuration

Declarative

```
resource "aws_instance" "web" {
    ami = "ami-1234"
    instance_type = "m1.small"
}
```

```
resource "<resource_type>" "web" {
    ami = "ami-1234"
    instance_type = "m1.small"
}
```

```
resource "aws_instance" "<name>" {
    ami = "ami-1234"
    instance_type = "m1.small"
}
```

```
resource "aws_instance" "web" {
    <param_name> = "<param_value>"
    <param_name> = "<param_value>"
}
```

Providers

Combine multiple providers in a single file

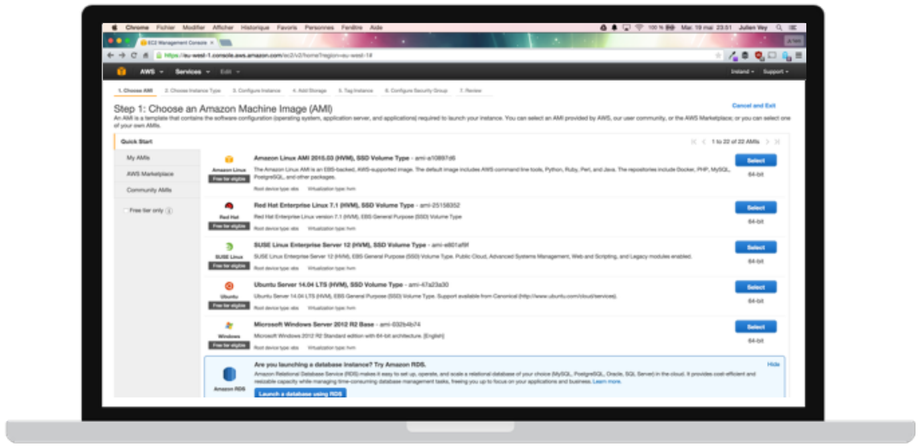
12+ providers

AWS, Docker, OpenStack, Google Cloud, Simple DNS…

```
resource "aws_instance" "web" {
    ami = "ami-1234"
    instance_type = "m1.small"
}

resource "dnssimple_record" "web" {
    domain = "example.com"
    name = "test"
    type = "A"
    value = "${aws_instance.web.public_ip}"
}
```

THE PATH OF A STARTUP WITH TERRAFORM

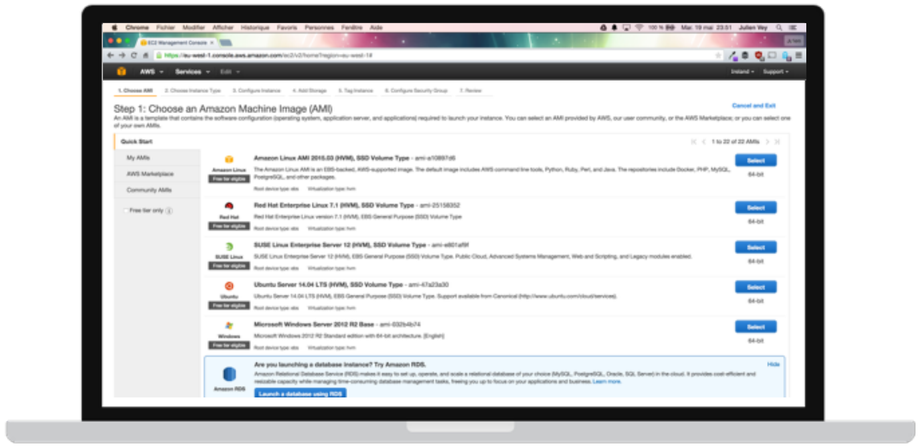
Day 1, let's start an EC2 Instance

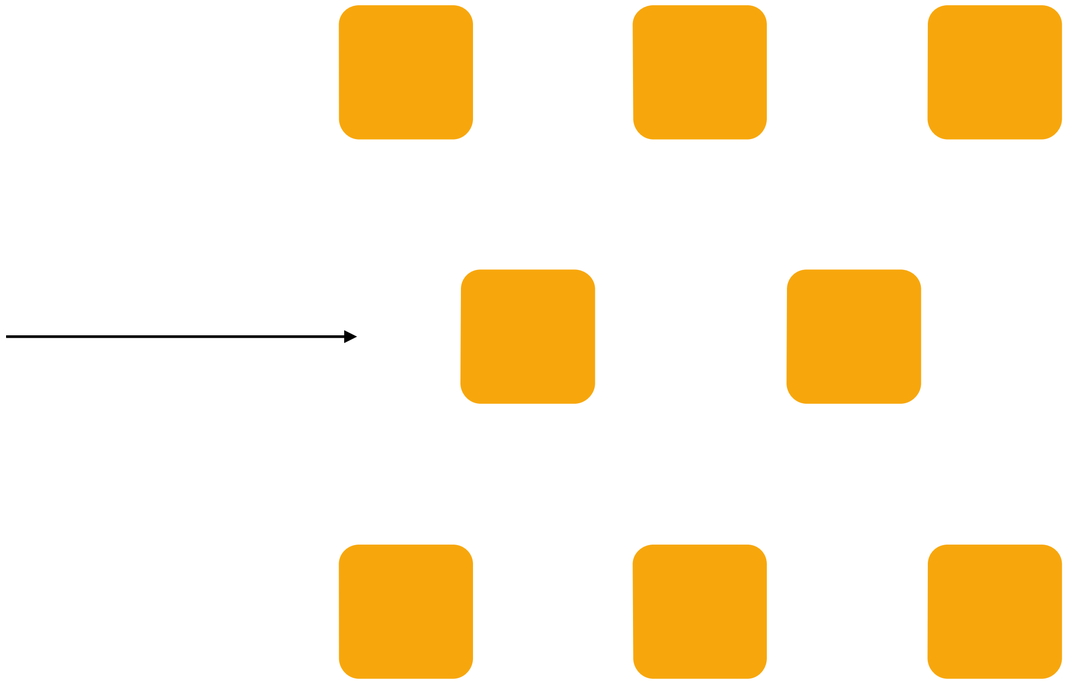


Day 1, let's start an EC2 Instance

```
resource "aws_instance" "web" {
    ami = "ami-1234"
    instance_type = "m1.small"
}
```

Day 2, let’s start more instances





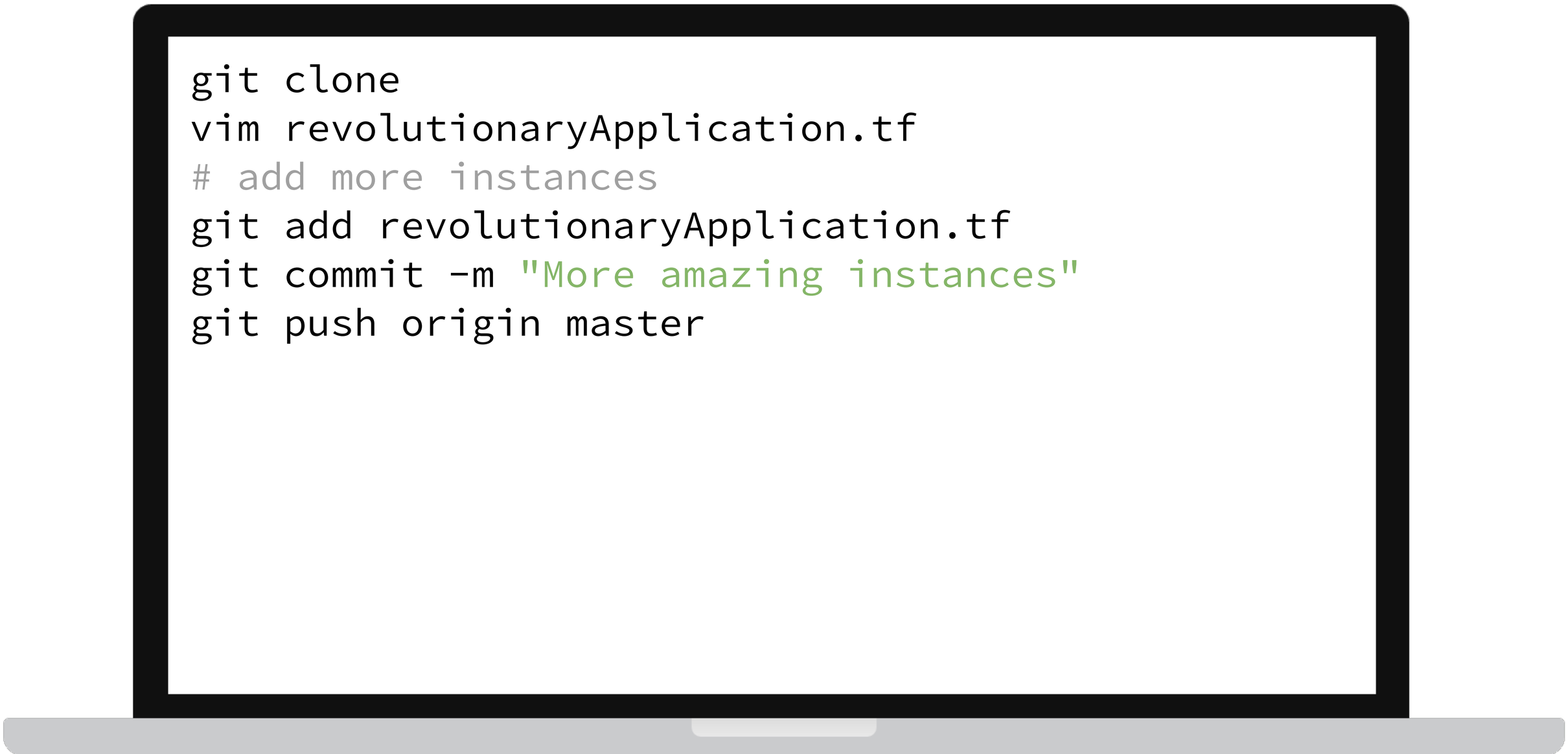
Day 2, let's start more instances

```
resource "aws_instance" "web" {
    ami = "ami-1234"
    instance_type = "m1.small"
}

resource "aws_instance" "backoffice" {
    ami = "ami-1234"
    instance_type = "m1.large"
}
```

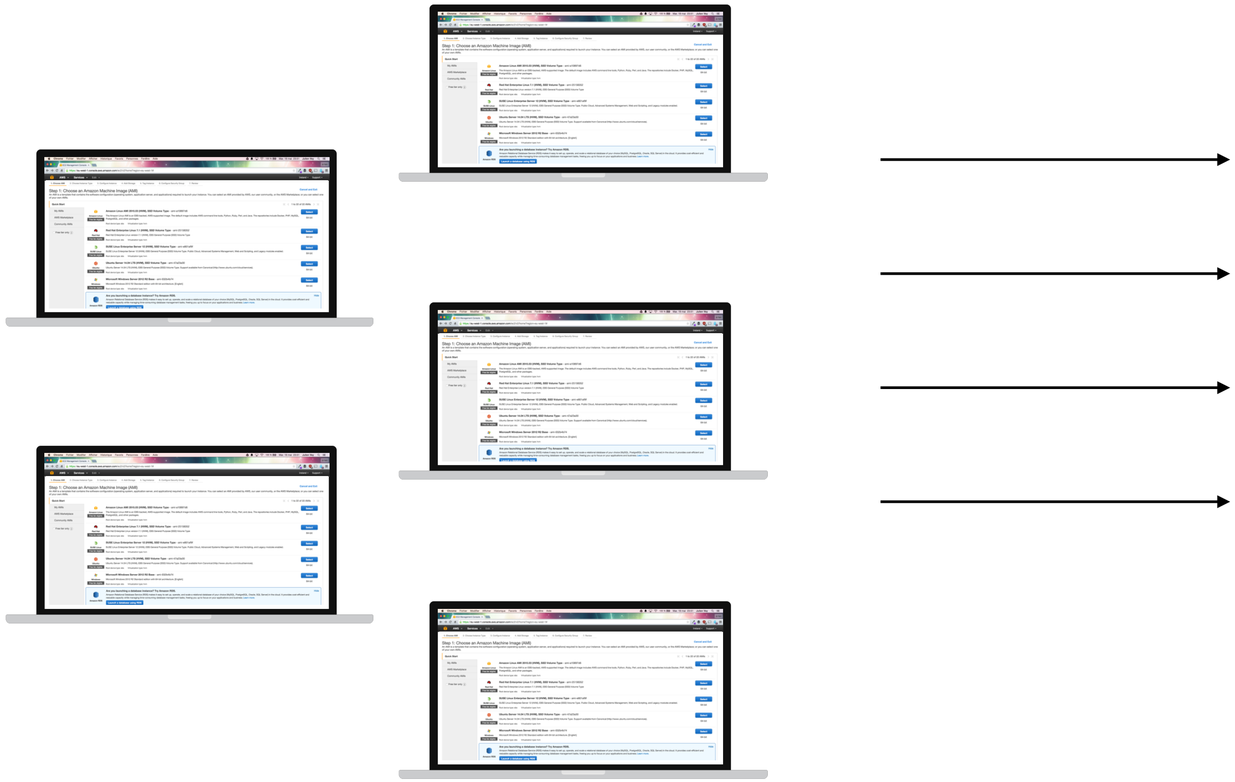
Day 3, let's hire more developers

Day 3, let's hire more developers

Day 3, let's hire more developers



Day 4, Hey! Google looks great, let's start some GCE instances



Day 4, Hey! Google looks great, let's start some GCE instances

```
resource "google_compute_instance" "default" {
    name = "test"
    machine_type = "n1-standard-1"
    zone = "us-central1-a"
}
```

And so on…

UNDER THE HOOD

Setup your provider

```
provider "aws" {
    access_key = "ACCESS_KEY_HERE"
    secret_key = "SECRET_KEY_HERE"
    region = "us-east-1"
}
```

Create your infrastructure as code

```
resource "aws_instance" "web" {
    ami = "ami-1234"
    instance_type = "m1.small"
}
```

Apply your infrastructure

```
$ terraform apply
```

Apply your infrastructure

```
$ terraform apply
```

*Generate a **tfstate** file*

*Store the last **known state** of the infrastructure*

Apply your infrastructure

```
$ terraform apply

aws_instance.web: Creating…
  ami:                "" => "ami-1234"
  instance_type: "" => "m1.small"
aws_instance.web: Creation complete

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

DEMO

State of your infrastructure

```
$ terraform show
```

State of your infrastructure

```
$ terraform show
```

*Read and print the **tfstate** file*

State of your infrastructure

```
$ terraform show
```

```
aws_instance.web:
  id = i-e60900cd
  ami = ami-1234
  availability_zone = us-east-1c
  instance_type = m1.small
  private_dns = domU-12-31-39-12-38-AB.compute-1.internal
  private_ip = 10.200.59.89
  public_dns = ec2-54-81-21-192.compute-1.amazonaws.com
  public_ip = 54.81.21.192
  security_groups.# = 1
  security_groups.0 = default
```

DEMO

Update your infrastructure

```
resource "aws_instance" "web" {
    ami = "ami-1234"
    instance_type = "m1.small"
}
```

Update your infrastructure

```
resource "aws_instance" "web" {
    ami = "ami-1234"
    # instance_type = "m1.small"
    instance_type = "m1.medium"
}
```

Plan your update

```
$ terraform plan
```

Plan your update

```
$ terraform plan
```

*Generates an **execution plan***

Plan your update

```
$ terraform plan

Refreshing Terraform state prior to plan...

aws_instance.web: Refreshing state... (ID: i-464b0bec)

-/+ aws_instance.web
    ami:                          "ami-e4ff5c93" => "ami-e4ff5c93"
    instance_type:                "t2.micro" => "t2.small" (forces new resource)
```

Plan your update

```
$ terraform plan
```

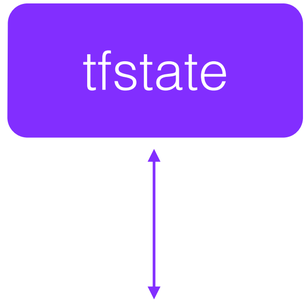
```
Refreshing Terraform state prior to plan...

aws_instance.web: Refreshing state... (ID: i-464b0bec)

-/+ aws_instance.web
    ami:                          "ami-e4ff5c93" => "ami-e4ff5c93"
    instance_type:                "t2.micro" => "t2.small" (forces new resource)
```

Plan your update

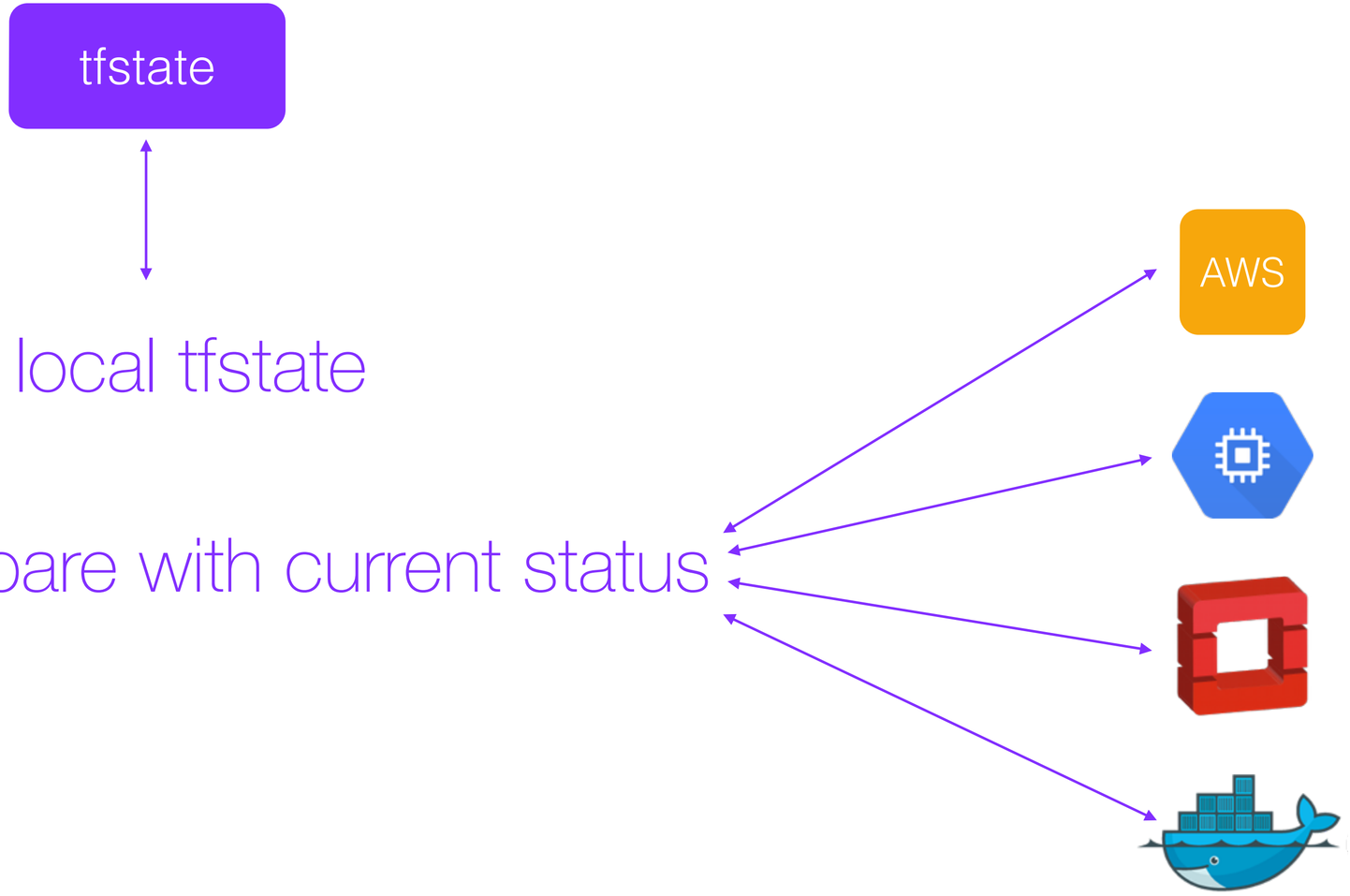
```
$ terraform plan

Refreshing Terraform state prior to plan...

aws_instance.web: Refreshing state... (ID: i-464b0bec)

-/+ aws_instance.web
    ami:                          "ami-e4ff5c93" => "ami-e4ff5c93"
    instance_type:                "t2.micro" => "t2.small" (forces new resource)
```

Plan your update

```
$ terraform plan

Refreshing Terraform state prior to plan...

aws_instance.web: Refreshing state... (ID: i-464b0bec)

-/+ aws_instance.web
    ami:                "ami-e4ff5c93" => "ami-e4ff5c93"
    instance_type:      "t2.micro" => "t2.small" (forces new resource)
```

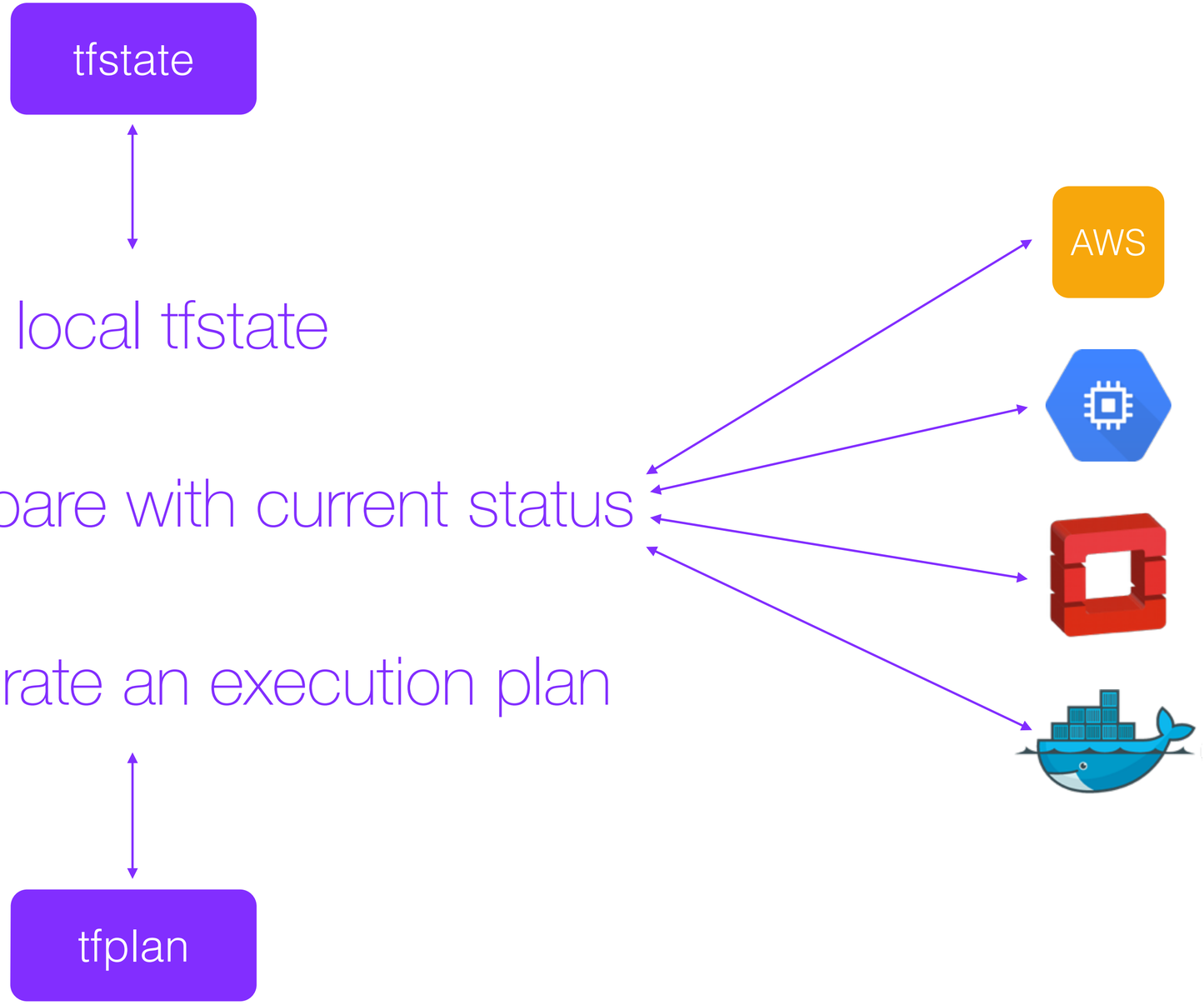


1. Read local tfstate



1. Read local tfstate

2. Compare with current status


tfstate
1. Read local tfstate
2. Compare with current status
3. Generate an execution plan
tfplan
AWS

Apply your update

```
$ terraform apply
```

*Apply the **execution plan***

*Udpate the **tfstate***

Apply your update

```
$ terraform apply

aws_instance.web: Refreshing state... (ID: i-464b0bec)
aws_instance.web: Destroying...
aws_instance.web: Destruction complete
aws_instance.web: Creating...
  ami:                          "" => "ami-e4ff5c93"
  instance_type:                "" => "t2.small"
aws_instance.web: Creation complete

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

DEMO

Destroy your infrastructure

```
$ terraform plan -destroy
```

Destroy your infrastructure

```
$ terraform plan -destroy
```

*Generates an **execution plan** for the destroy command*

Destroy your infrastructure

```
$ terraform plan -destroy

Refreshing Terraform state prior to plan...

aws_instance.web: Refreshing state... (ID: i-d54e0e7f)

- aws_instance.web
```

Destroy your infrastructure

```
$ terraform destroy
```

Destroy your infrastructure

```
$ terraform destroy
```

Apply the destroy ***execution plan***

Destroy your infrastructure

```
$ terraform destroy

aws_instance.web: Refreshing state... (ID: i-d54e0e7f)
aws_instance.web: Destroying...
aws_instance.web: Destruction complete

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

DEMO

AND MORE...

Resource dependencies

Implicit dependencies

```
resource "aws_instance" "web" {
    ami = "ami-1234"
    instance_type = "m1.medium"
}
```

Implicit dependencies

```
resource "aws_instance" "web" {
    ami = "ami-1234"
    instance_type = "m1.medium"
}

resource "aws_eip" "ip" {
    instance = "${aws_instance.web.id}"
}
```

Implicit dependencies

```
resource "aws_instance" "web" {
    ami = "ami-1234"
    instance_type = "m1.medium"
}

resource "aws_eip" "ip" {
    instance = "${aws_instance.web.id}"
}
```

Plan your infrastructure

```
$ terraform plan
```

```
+ aws_eip.ip
    instance:   "" => "${aws_instance.web.id}"
    private_ip: "" => "<computed>"
    public_ip:  "" => "<computed>"

+ aws_instance.web
    ami:                "" => "ami-1234"
    availability_zone:  "" => "<computed>"
    instance_type:      "" => "m1.medium"
    private_ip:         "" => "<computed>"
    public_ip:          "" => "<computed>"
```

Apply your infrastructure

```
$ terraform apply
```

```
aws_instance.web: Creating...
  ami:           "" => "ami-1234"
  instance_type: "" => "m1.medium"
aws_eip.ip: Creating...
  instance: "" => "i-0e737b25"

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```

Explicit dependencies

Explicit dependencies

```
resource "aws_instance" "back" {
    ami = "ami-1234"
    instance_type = "m1.medium"
}
```

Explicit dependencies

```
resource "aws_instance" "back" {
    ami = "ami-1234"
    instance_type = "m1.medium"
}

resource "aws_instance" "database" {
    ami = "ami-1234"
    instance_type = "m1.large"
}
```

Explicit dependencies

```
resource "aws_instance" "back" {
    ami = "ami-1234"
    instance_type = "m1.medium"
    depends_on = ["aws_instance.database"]
}

resource "aws_instance" "database" {
    ami = "ami-1234"
    instance_type = "m1.large"
}
```

Explicit dependencies

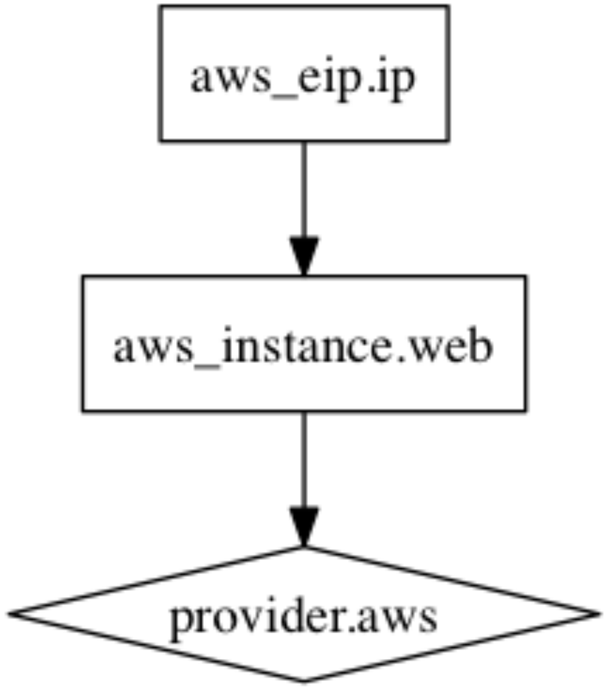
```
resource "aws_instance" "back" {
    ami = "ami-1234"
    instance_type = "m1.medium"
    depends_on = ["aws_instance.database"]
}

resource "aws_instance" "database" {
    ami = "ami-1234"
    instance_type = "m1.large"
}
```

Terraform graph

```
$ terraform graph
```

Terraform graph

```
$ terraform graph
```



DEMO

Variables

Variables

```
variable "access_key" {}
variable "secret_key" {}
variable "region" {
    default = "us-east-1"
}
```

Variables

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

Variables

```
$ terraform apply
```

Variables

```
$ terraform apply -var 'access_key=foo' \
                  -var 'secret_key=bar'
```

Variables

```
$ export TF_VAR_access_key=foo
$ export TF_VAR_secret_key=bar
```

Variables

```
$ vim terraform.tfvars

access_key = "foo"
secret_key = "bar"
```

Variables

```
$ terraform apply -var-file terraform.tfvars
```

(DEMO)

Modules

self-contained packages

create reusable components

Modules

```
module "postgresql" {
     source = "git@mon-gitlab.com:tf/postgresql.git"
     servers = "3"
}
```

PROVISIONNERS

Provisioners

Chef

Files

Local-exec

Remote-exec

Provisionning with local-exec

```
resource "aws_instance" "web" {
  ami = "ami-1234"
  instance_type = "m1.small"
  provisioner "local-exec" {
    command = "ansible-playbook -i invnt/aws.py web.yml"
  }
}
```

Provisionning with local-exec

```
resource "aws_instance" "web" {
  ami = "ami-1234"
  instance_type = "m1.small"
  provisioner "local-exec" {
    command = "ansible-playbook -i invnt/aws.py web.yml"
  }
}
```

Provisionning with remote-exec

```
resource "aws_instance" "web" {
  ami = "ami-1234"
  instance_type = "m1.small"
  provisioner "remote-exec" {
    inline = ["puppet apply"]
  }
}
```

Provisionning with remote-exec

```
resource "aws_instance" "web" {
  ami = "ami-1234"
  instance_type = "m1.small"
  provisioner "remote-exec" {
    inline = ["puppet apply"]
  }
}
```

DEMO

WORKFLOWS

#some real work
git commit

#some real work
git commit

git push

.git

```
# tf and tfvars files
resource "aws_instance" "web" {
  ami = "ami-1234"
  instance_type = "m1.small"
  provisioner "remote-exec" {
    inline = ["puppet apply"]
  }
}
```

#some real work
git commit

git push

.git

```
# tf and tfvars files
resource "aws_instance" "web" {
  ami = "ami-1234"
  instance_type = "m1.small"
  provisioner "remote-exec" {
    inline = ["puppet apply"]
  }
}
```

review
pull requests

#some real work
git commit

git push

.git

```
# tf and tfvars files
resource "aws_instance" "web" {
  ami = "ami-1234"
  instance_type = "m1.small"
  provisioner "remote-exec" {
    inline = ["puppet apply"]
  }
}
```

review
pull requests

git hook

#some real work
git commit

git push

.git

```
# tf and tfvars files
resource "aws_instance" "web" {
  ami = "ami-1234"
  instance_type = "m1.small"
  provisioner "remote-exec" {
    inline = ["puppet apply"]
  }
}
```

review
pull requests

git hook

git pull

```
# tfstate files
"resources": {
  "aws_instance.web": {
    "type": "aws_instance",
    "primary": {
      "id": "i-17e1a6bd",
      "attributes": {
        "ami": "ami-e4ff5c93",
        "instance_type": "t2.small",
      }
    }
  }
}
```

.git

#some real work
git commit

git push

.git

```
# tf and tfvars files
resource "aws_instance" "web" {
  ami = "ami-1234"
  instance_type = "m1.small"
  provisioner "remote-exec" {
    inline = ["puppet apply"]
  }
}
```

review
pull requests

git hook

terraform apply

git pull

```
# tfstate files
"resources": {
  "aws_instance.web": {
    "type": "aws_instance",
    "primary": {
      "id": "i-17e1a6bd",
      "attributes": {
        "ami": "ami-e4ff5c93",
        "instance_type": "t2.small",
      }
    }
  }
}
```

.git

#some real work
git commit

git push

.git

```
# tf and tfvars files
resource "aws_instance" "web" {
  ami = "ami-1234"
  instance_type = "m1.small"
  provisioner "remote-exec" {
    inline = ["puppet apply"]
  }
}
```

review
pull requests

git hook

terraform apply

git commit *.tfsate

git pull

git push

```
# tfstate files
"resources": {
  "aws_instance.web": {
    "type": "aws_instance",
    "primary": {
      "id": "i-17e1a6bd",
      "attributes": {
        "ami": "ami-e4ff5c93",
        "instance_type": "t2.small",
      }
    }
  }
}
```

.git

CONCLUSION

+ terraform
  Provider Agnostic
  Cloud resources as code
  Everything command line

- terraform
  Providers maturity
  Updating infrastructure
  Backward compatibility
  Error messages

THANK YOU

QUESTIONS ?