# infrastructure as code
# with **Terraform**

ideato

# Marco Colombo

DevOps @ ideato

@marco_colombo

@marcocolomboid

marco.colombo@ideato.it

ideato

# web craftsmen

we create software and develop strategies shaping our clients' ideas

ideato

# With Great Automation Comes Great Responsibility

ideato

# Write and execute code to define, deploy and update infrastructure

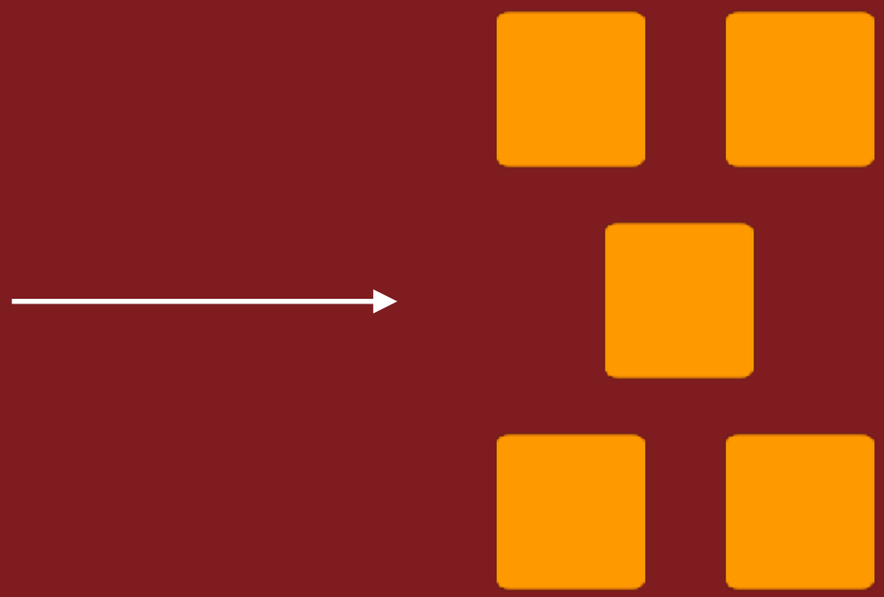
IAAC/Why?

# the path of a startup

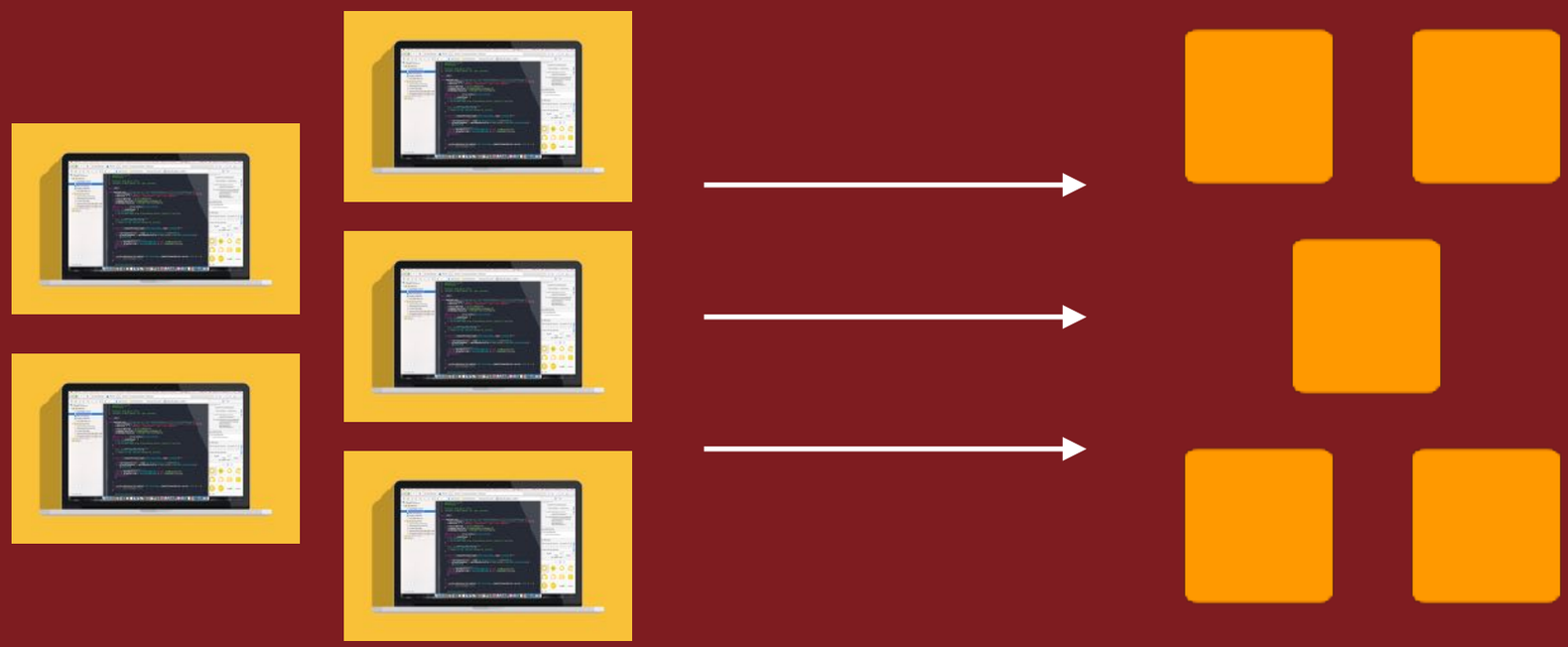
ideato

# Day 1, let's start an EC2 instance

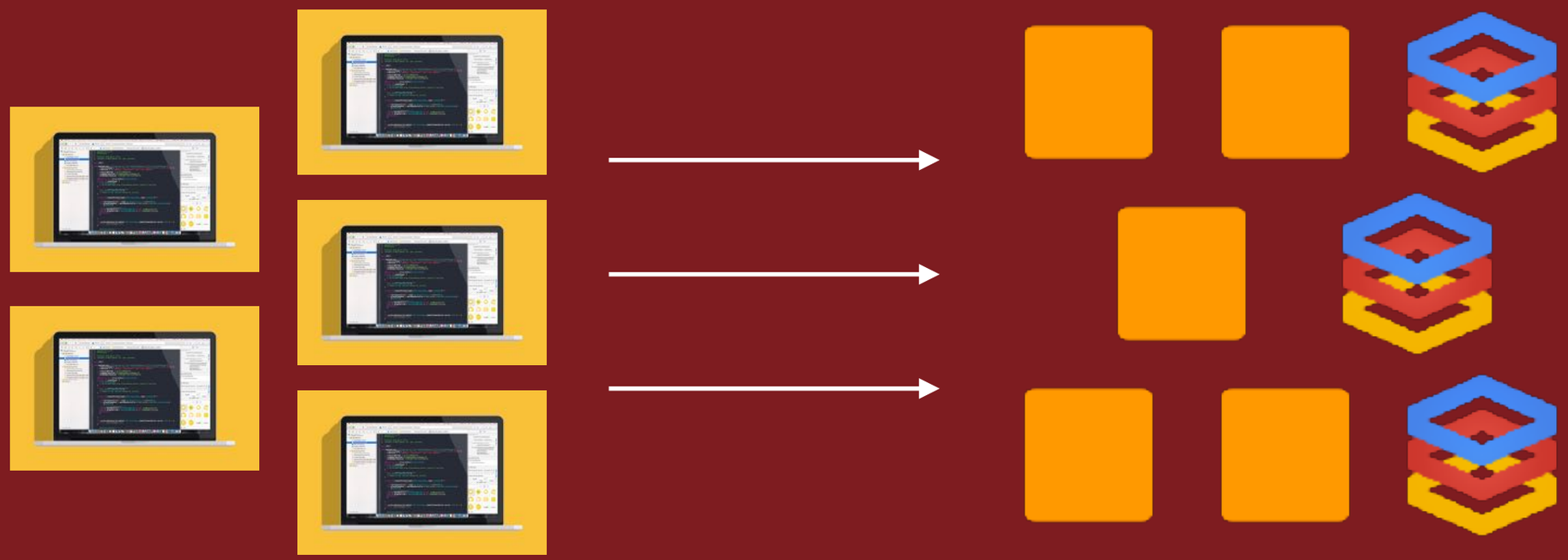
# Day 2, let's start more instances



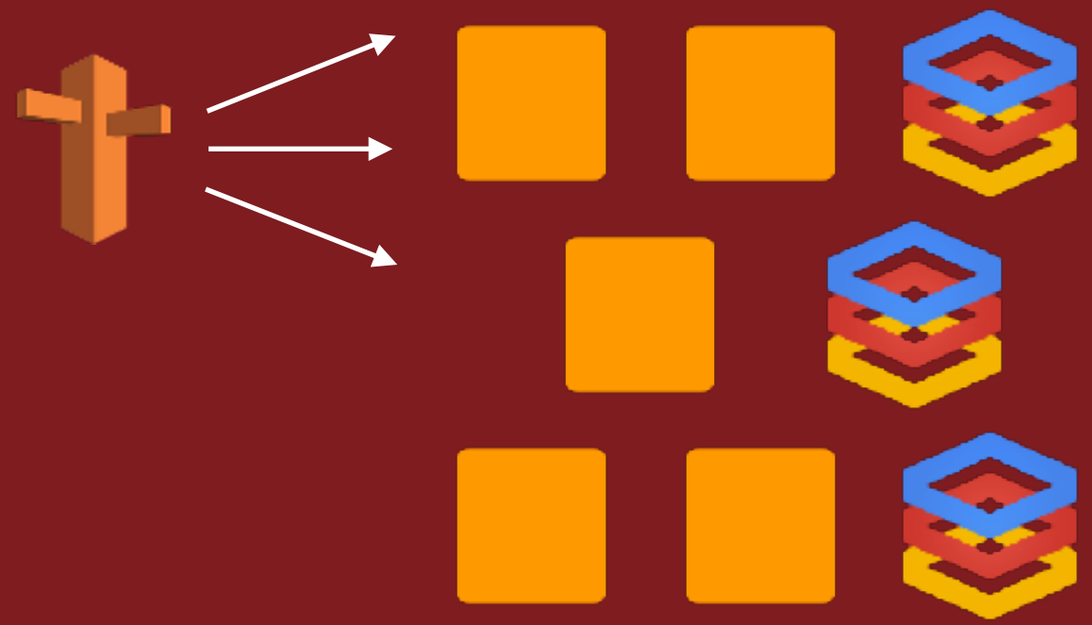
# Day 3, let's hire more developers



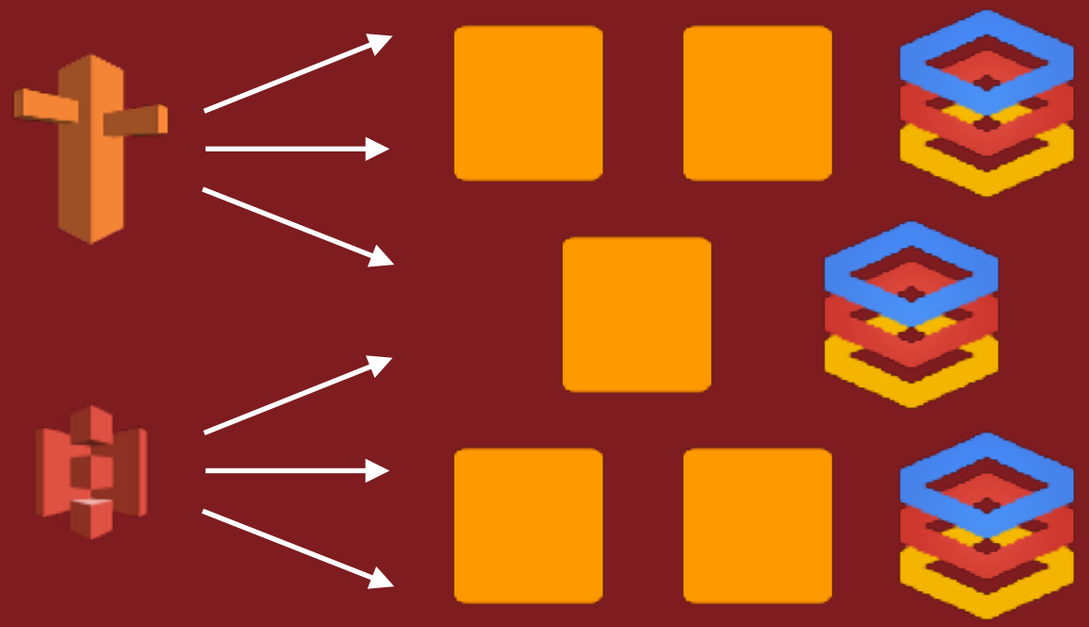
# Day 4, Google looks great! let's start some GCE instances



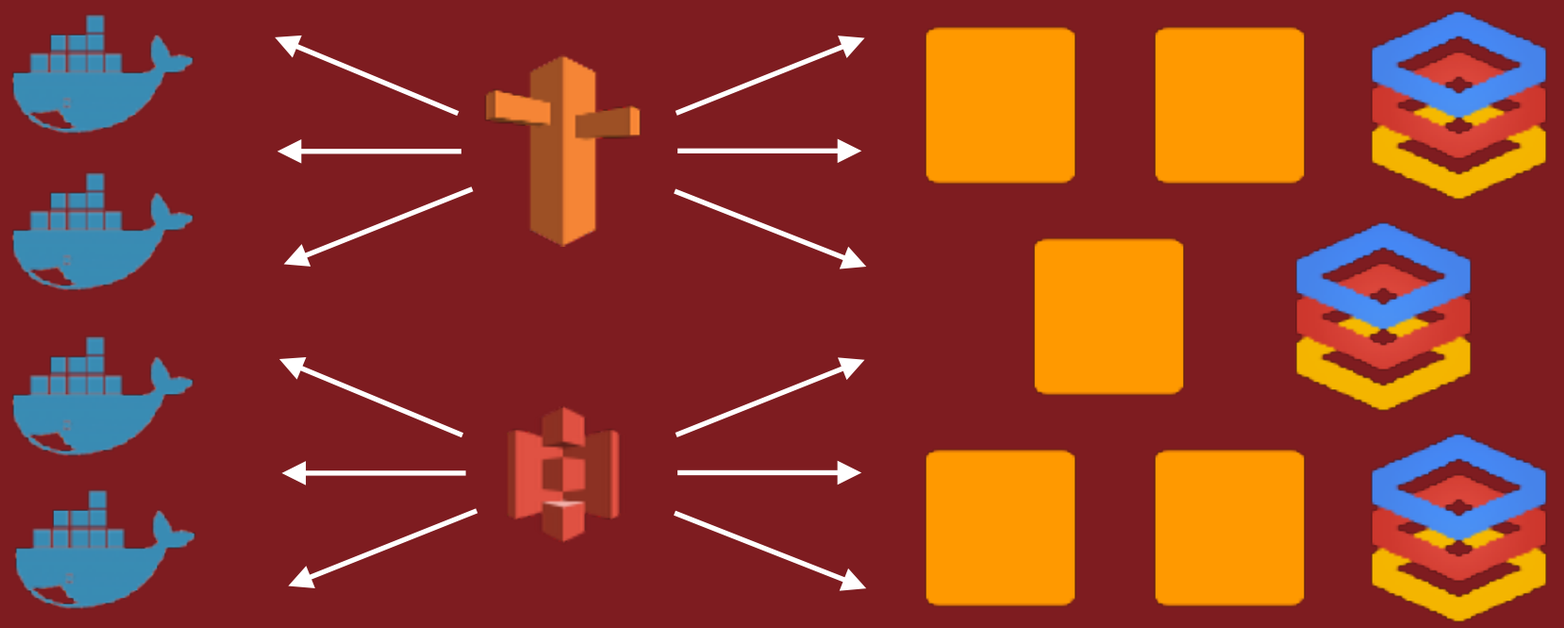
# Day 5, let's use a Cloud DNS



# Day 6, let's use a Cloud Storage



# Day 7, Docker is cool! We need to use containers



# and...
# this is how you get

IAAC/Why?

from a

# basic cloud infrastructure

to a

# complex cloud infrastructure

( in just 7 days )

ideato

# how to?

- keep track of your inventory
- manage changes
- control the lifecycle of your resource
- manage different resource among different cloud provider

# IAAC/Why?

terraform

ideato

# iaac allow us to:

- avoid human errors in infrastructure creation
- avoid repetitive, time consuming tasks
- version control and code review
- knowledge sharing

# so it’s like ansible...puppet...chef ?

# NO

# configuration management:

- designed to install and mange software
- do not know the actual state of the infrastructure
- build for mutable infrastructure

ideato

# during a scale out

```
- ec2:
  count: 10
  image: ami-v1
  instance_type: t2.micro
```

```
resource "aws_instance" "example" {
  count = 10
  ami = "ami-v1"
  instance_type = "t2.micro"
}
```

| n° of instances | n° of instances |
|---|---|
| 10 | 10 |

ideato

# during a scale out

```
- ec2:
    count: ~~10~~ 15
    image: ami-v1
    instance_type: t2.micro
```

```
resource "aws_instance" "example" {
  count = ~~10~~ 15
  ami = "ami-v1"
  instance_type = "t2.micro"
}
```

| n° of instances | n° of instances |
|---|---|
| 25 | 15 |

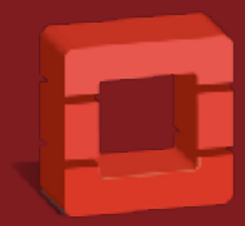
ideato

# why not Cloud Formation?

ideato

# multi-provider support










ideato

# and..

- generates an execution plan
- possible to write custom plugin
- use of HCL language
- open source
- saves current state of infrastructure

# creating an ec2 instance

```
resource "aws_instance" "web" {
  ami           = "ami-fff61890"
  instance_type = "t1.small"
}
```

# creating an ec2 instance

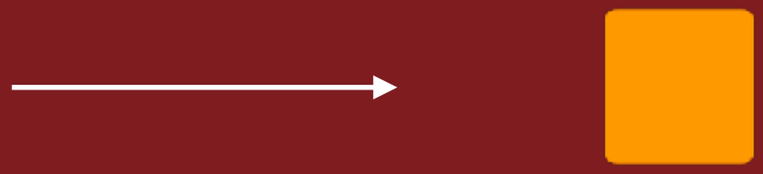
```
resource "resource_type" “resource_name" {
  param_name    = "param_value"
  param_name    = "param_value"
}
```

IAAC/example

# the path of a startup with **Terraform**

ideato

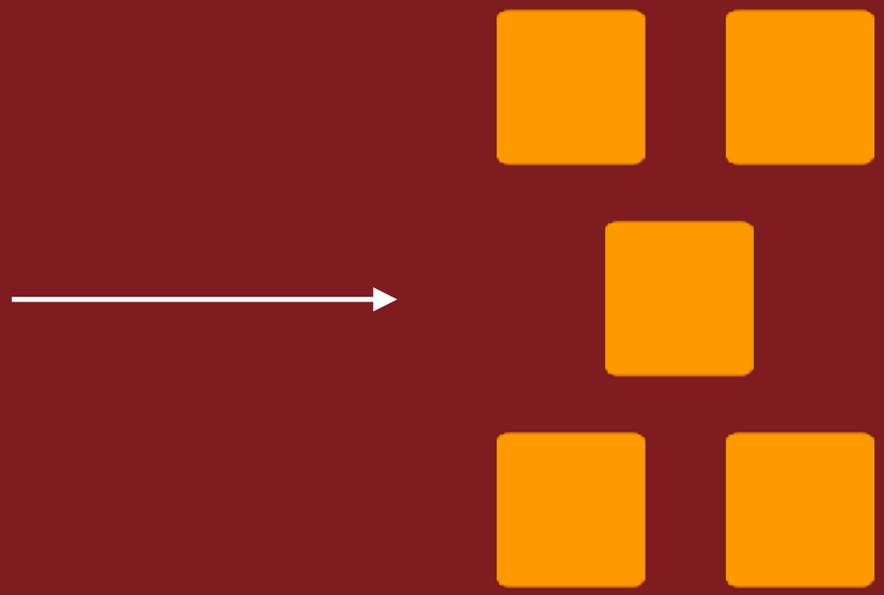
# Day 1, let's start an EC2 instance



# Day 1, let's start an EC2 instance

```
resource "aws_instance" "web" {
  ami           = "ami-fff61890"
  instance_type = "t1.small"
}
```

# Day 2, let's start more instances

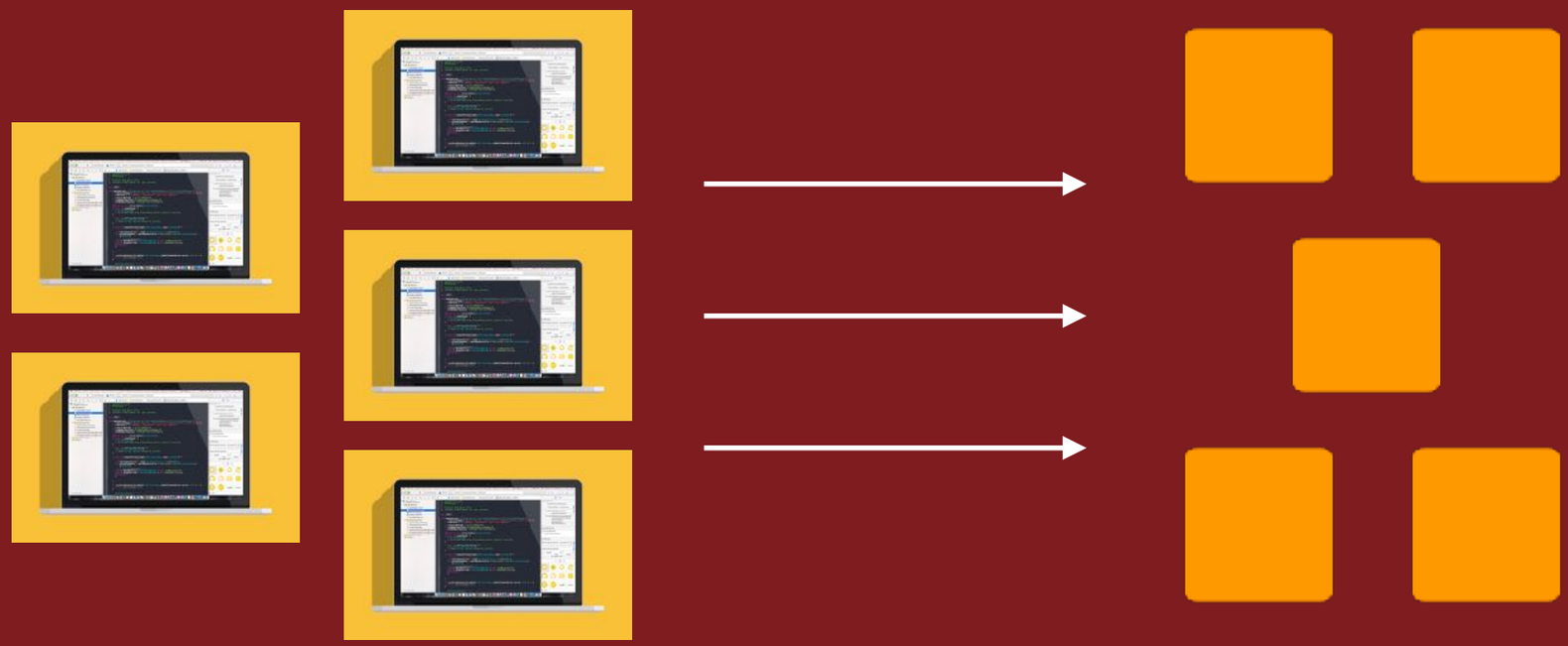


## Day 2, let's start more instances

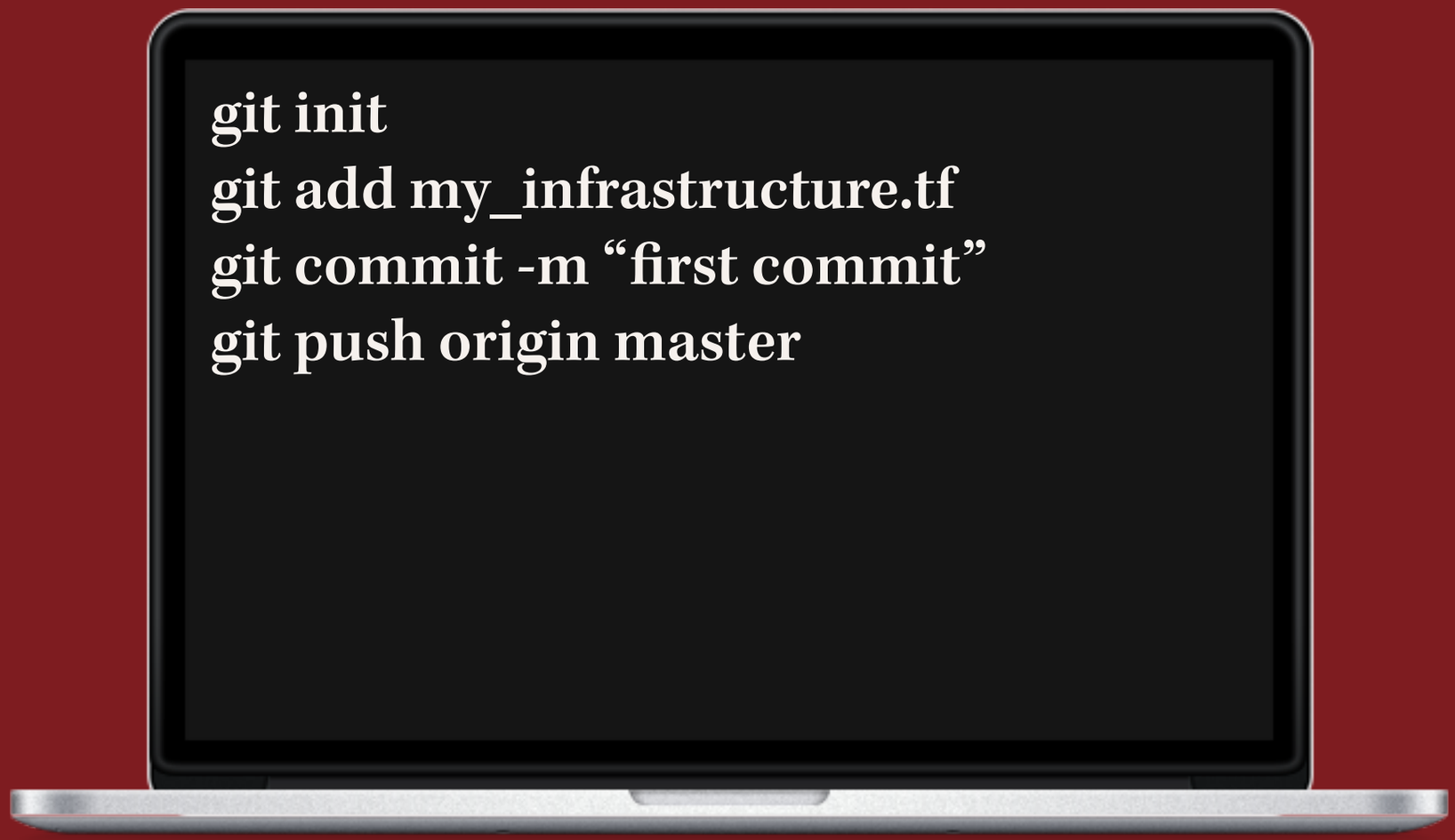
```
resource "aws_instance" "web" {
  ami           = "ami-fff61890"
  instance_type   = "t2.small"
}
```

```
resource "aws_instance" "backoffice" {
  ami           = "ami-fff61890"
  instance_type   = "t2.medium"
}
```

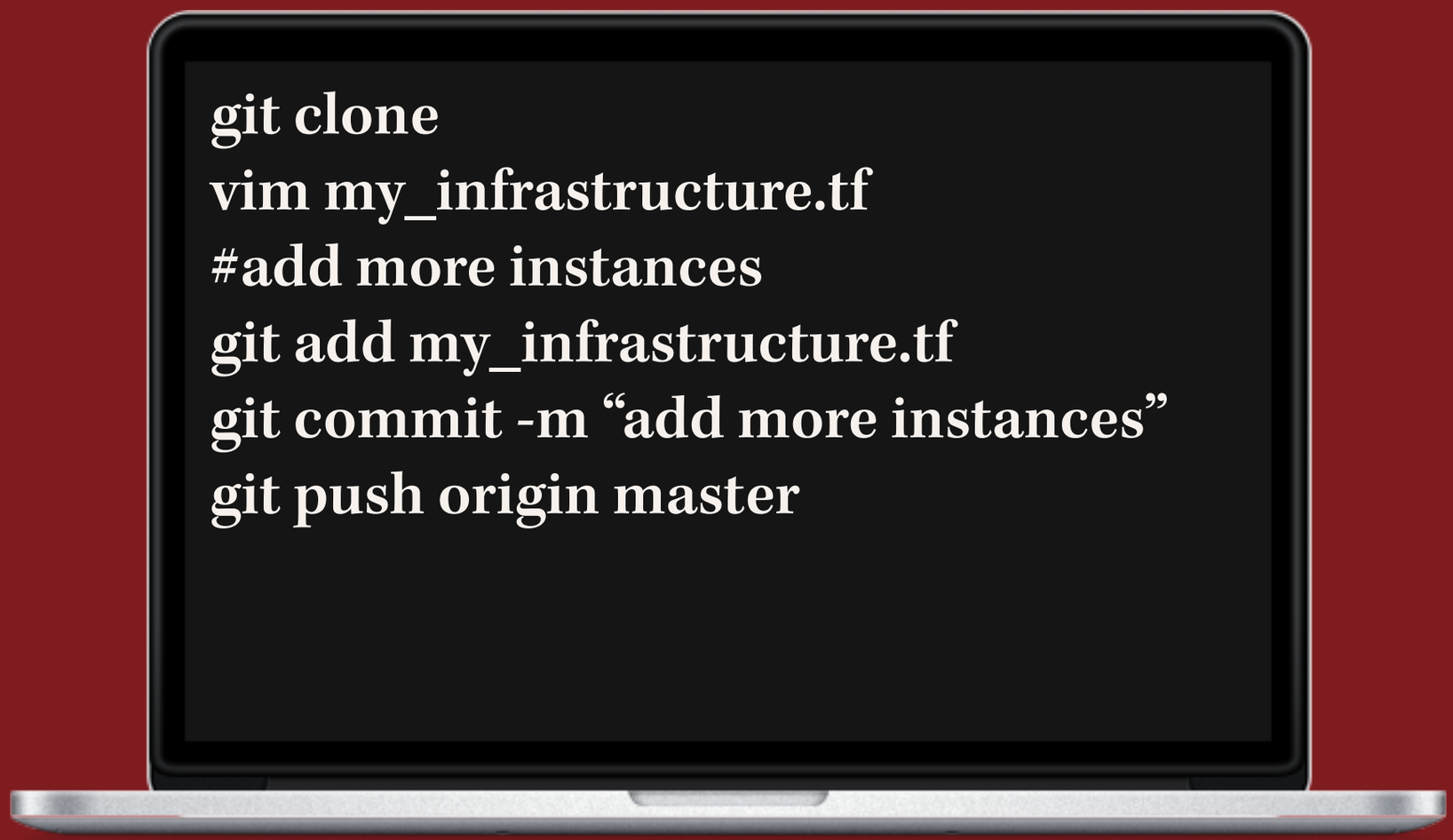
# Day 3, let's hire more developers



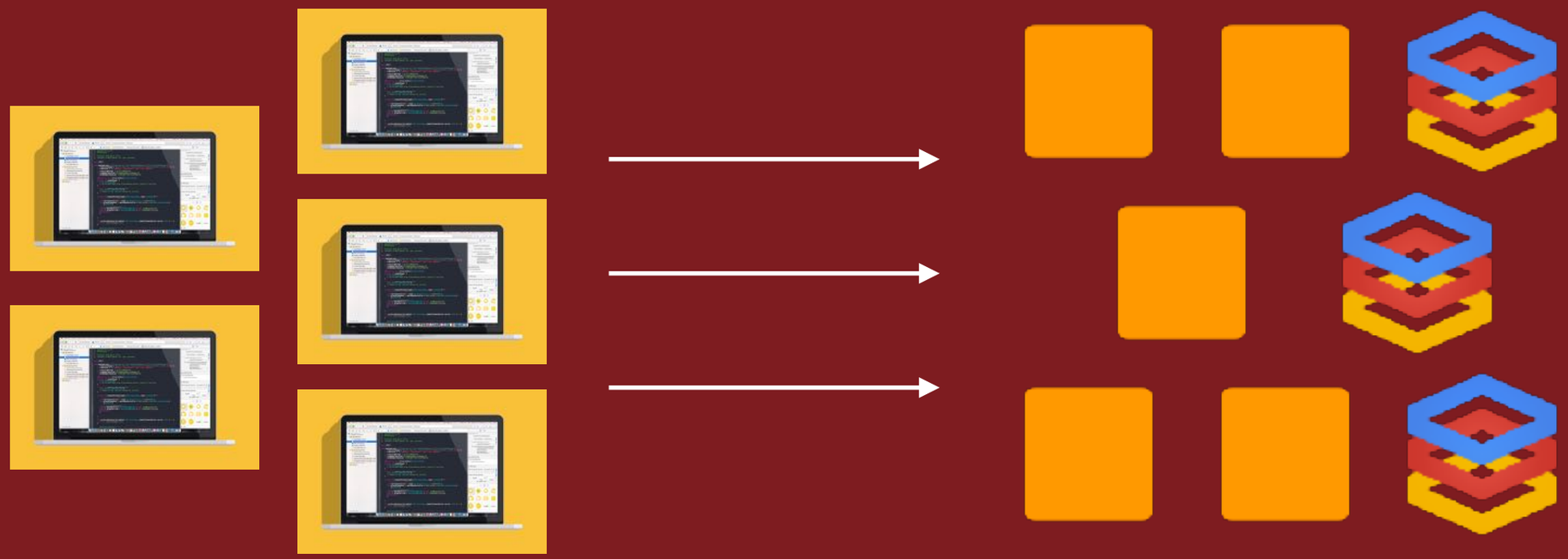
# Day 3, let’s hire more developers



# Day 3, let’s hire more developers



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



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

```
resource "google_compute_instance" "default" {
  name         = "stage"
  machine_type  = “n1-standard-1"
  zone  = "europe-west1-b"
}
```

and so on...

# under the hood

ideato

# Configure your provider

```
provider "aws" {
  access_key          = "ACCESS_KEY"
  secret_key          = "SECRET_KEY"
  region              = "eu-central-1"
}
```

# Create your infrastructure as code

```
resource "aws_instance" "web" {
  ami           = "ami-fff61890"
  instance_type = "t2.small"
}
```

# Plan your infrastructure

## $ terraform plan

# Plan your infrastructure

## $ terraform plan

- it will generate a .tfstate
- tfstate file will store the last state of the infrastructure

# Apply your infrastructure

```
$ terraform apply
    aws_instance.web: Creating...
      ami:  "" => "ami-fff61890"
      instance_type: "" => "t2.small"
    aws_instance.web: Creation complete

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

# Show your infrastructure

## $ terraform show

### Read and print the tfstate file

# Show your infrastructure

```
aws_instance.web:
  id = i-0f69bd4ff0a5879ed
  ami = ami-88ca94e8
  associate_public_ip_address = true
  availability_zone = us-west-1a
  ephemeral_block_device.# = 0
  instance_state = running
  instance_type = t2.small
  private_dns = ip-172-31-30-80.us-west-1.compute.internal
  private_ip = 172.31.30.80
  public_dns = ec2-52-53-227-74.us-west-1.compute.amazonaws.com
  public_ip = 52.53.227.74
```

# Update your infrastructure

```
resource "aws_instance" "web" {
  ami           = "ami-fff61890"
  instance_type = "t2.small"
}
```

# Update your infrastructure

```
resource "aws_instance" "web" {
  ami           = "ami-fff61890"
  # instance_type   = “t2.small"
  instance_type   = "t2.medium"
}
```

# Plan your update

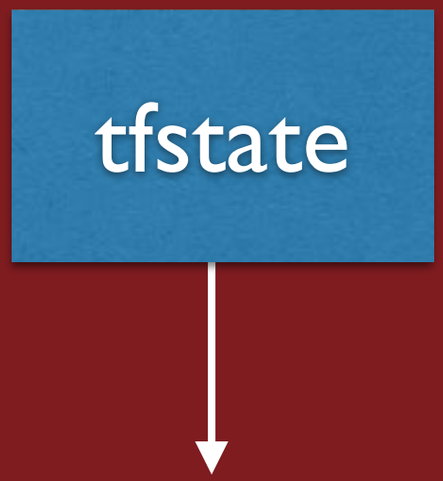
## $ terraform plan

### Generate an execution plan

# Plan your update

## $ terraform plan

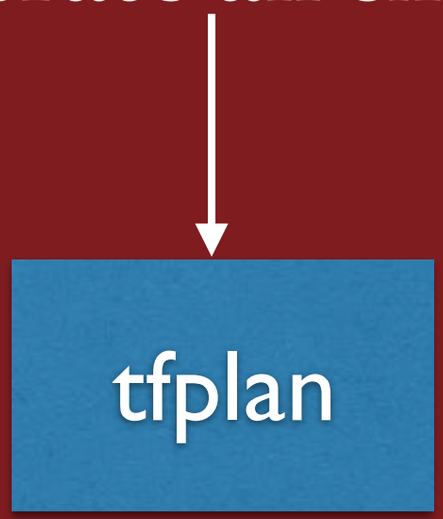
```
aws_instance.web: Refreshing state... (ID: i-0f69bd4ff0a5879ed)
data.aws_availability_zones.available: Refreshing state...
-/+ aws_instance.web
    instance_state:          "running" => "<computed>"
    instance_type:           "t2.small" => "t2.medium" ( forces new
resource)
```




1. Read local tfstate

2. Compare with current status

3. Generate an execution plan



# Apply your infrastructure

## $ terraform apply

### 1. Apply the execution plan

### 2. Update the tfstate

# Destroy your infrastructure

## $ terraform  plan -destroy

Generate an execution plan for the destroy command

# Destroy your infrastructure

## $ terraform  plan -destroy

## Generate an execution plan for the destroy command

aws_instance.web: Refreshing state... (ID: i-0f69bd4ff0a5879ed)
data.aws_availability_zones.available: Refreshing state...
- aws_instance.web

# Destroy your infrastructure

## $ terraform  destroy

## Apply the destroy execution plan

# Destroy your infrastructure

```
$ terraform  destroy

    aws_instance.web: Destroying...
    aws_instance.web: Destruction complete

Destroy complete! Resources: 1 destroyed.
```

IAAC/uth

# demo

ideato

# 2016 DevOps Report *

- 24x faster to recover from failure
- 50% less time to find issues
- 200x more deploys
- 2.2x engineer happiness

* https://puppet.com/resources/whitepaper/2016-state-of-devops-report

ideato

?

ideato

# References

- https://www.terraform.io/intro/
- https://github.com/hashicorp/terraform/
- https://github.com/hashicorp/hcl
- https://speakerdeck.com/julienvey/terraform

* https://puppet.com/resources/whitepaper/2016-state-of-devops-report

ideato

# thx

ideato