Jakarta, 29 January 2018

Manage, Maintain and Automate Infrastructure as code using Terraform

DEVOPS INDONESIA

DevOps Community in Indonesia

Location : Traveloka
Wisma 77 Tower 2 Lt 2, Jl. S Parman Kav. 77 Slipi
Jakarta, Indonesia

Big Event DevOps Day Jakarta 2018

The 1st DevOpsDays in Indonesia

DevOps Indonesia is also the community Sponsor in DevOps Day Jakarta 2018

What is DevOps Day?

DevOpsDays is a 2-day conference & workshop that will be held in Jakarta on **April 26th and 27th, 2018**. This conference, which is now happening frequently around the globe, is the conference for bridging the gap between development and operations. DevOpsDays is a grassroots event for connecting professionals in the field allowing them to share experiences, advice, ideas, and tools relating to DevOps. It is organised by people who care about DevOps, for people who care about collaboration, automation, measurement, and improvement

The Venue

Menara BTPN, #27-01, Jl. Dr. Ide Anak Agung Gde Agung Kav 5.5 - 5.6, Kuningan Timur

Stay Connected

@devopsindonesia
http://www.devopsindonesia.com

DevOps Indonesia
https://t.me/joinchat/AAAAAFIN3xeiD8b8JBy8XA

Alone We are smart, together We are brilliant

THANK YOU !

Quote by Steve Anderson

Manage, Maintain and Automate Infrastructure as code using Terraform

About Me

- DevOps, Release, Infrastructure Engineer @Traveloka
- E-mail: amal.syahreza@traveloka.com
- Twitter: @amalsyahreza

Agenda

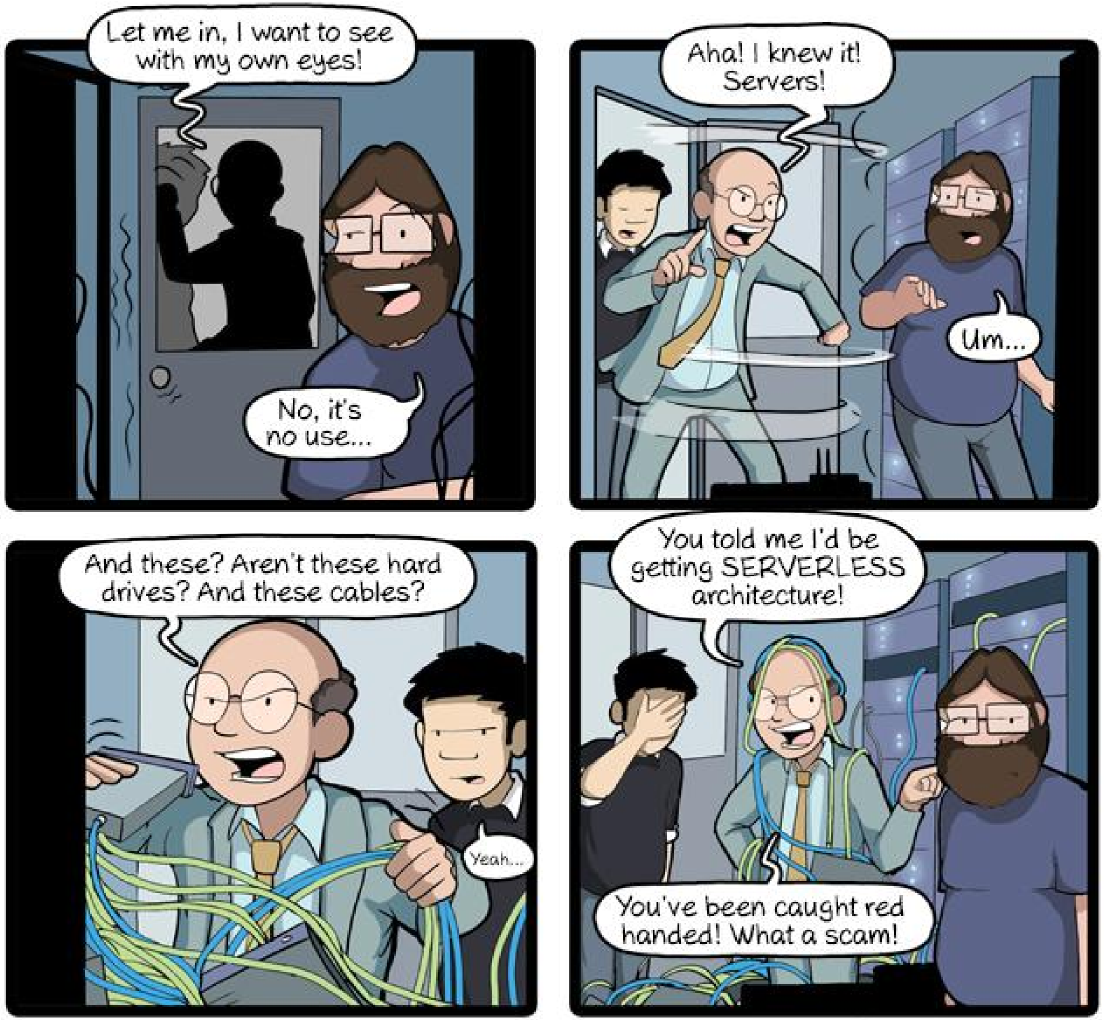
- Infrastructure as a code design.
- Comparison of Infrastructure as a Code Tooling (Ansible, CloudFormation, Terraform).
- Immutable Deployment with Terraform.
- Conclusion + Q&A

Infrastructure as A Code Design Patterns

Why Codify Your Infrastructure

- Automate provisioning and deployment process.
  - Faster and reliable deployment process.
  - Prevent human error in the execution process.
- Clear state of infrastructure.
- Validate the infrastructure change.
- Documented the change of infrastructure.

Technology continuously to evolve, but the goal stay same.

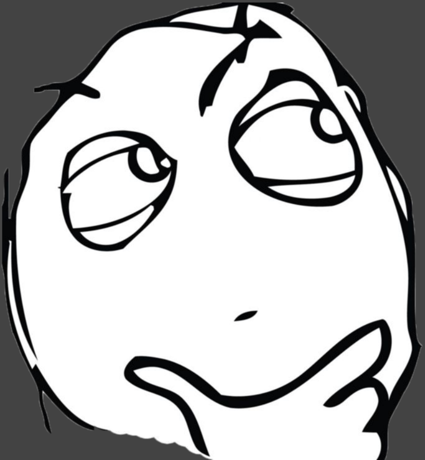

Let me in, I want to see with my own eyes!
No, it's no use...
Aha! I knew it! Servers!
Um...
And these? Aren't these hard drives? And these cables?
Yeah...
You told me I'd be getting SERVERLESS architecture!
You've been caught red handed! What a scam!

CommitStrip.com

What is a Good Design?

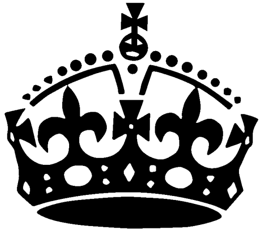
Unix Philosophy #1

Make each program do one thing well. To do a new job, build afresh rather than complicate old programs by adding new "features".



Unless you can be like this man...

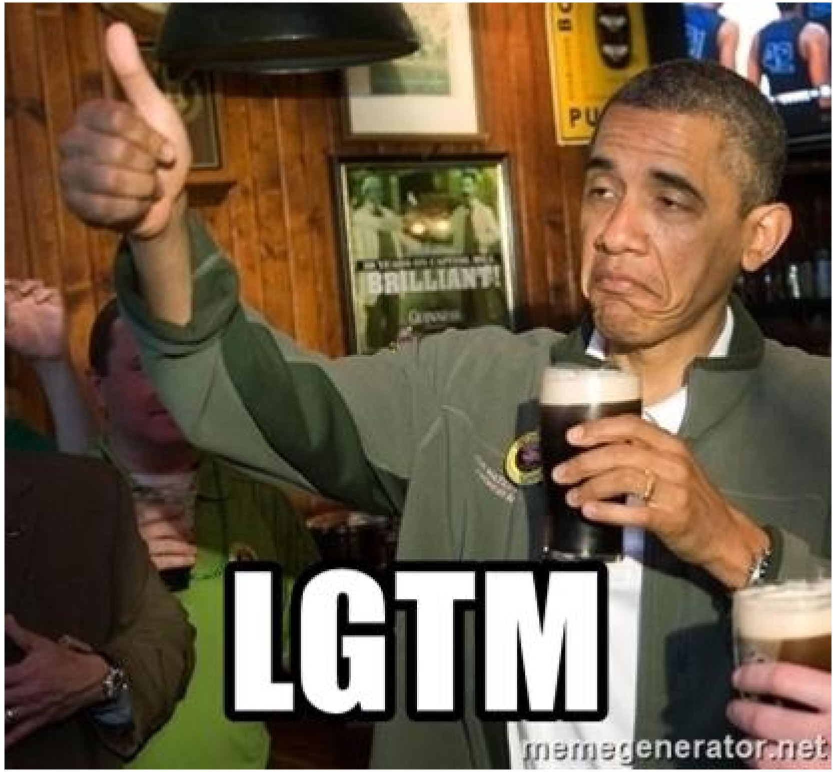
How to Create a Good Design?



KEEP
CALM
AND
LEAVE YOUR
EGO

Empower the People

Review the Design



Ansible vs CloudFormation Vs Terraform in Infrastructure Orchestration

General Comparison

	Ansible	CloudFormation	Terraform
Code	Open source	Closed source	Open source
Community Support	Yes	No	Yes
Cloud Support	Any	AWS	Any
Type	Config management	Infrastructure Orchestration	Infrastructure Orchestration
Language Style	Procedural	Declarative	Declarative

Example: Creating 2 EC2 Instances

Ansible	CloudFormation	Terraform
`- ec2: count: 2 image: ami-v1 instance_type: t2.micro`	`Ec2Instance01: Type: "AWS::EC2::Instance" Properties: ImageId: "ami-v1" InstanceType: “t2.micro”` `Ec2Instance02: Type: "AWS::EC2::Instance" Properties: ImageId: "ami-v1" InstanceType: “t2.micro”`	`resource "aws_instance" "example" { count = 2 ami = "ami-v1" instance_type = "t2.micro" }`

Ansible x CloudFormation x Terraform Demo

Procedural vs Declarative

Ansible	CloudFormation	Terraform
- ec2:	Ec2Instance01:	resource "aws_instance" "example" {
count: 4	Type: "AWS::EC2::Instance"	count = 4
image: ami-v2	Properties:	ami = "ami-v2"
instance_type: t2.micro	ImageId: "ami-v2"	instance_type = "t2.micro"
	InstanceType: “t2.micro”	}
	Ec2Instance02:	
	Type: "AWS::EC2::Instance"	
	Properties:	
	ImageId: "ami-v2"	
	InstanceType: “t2.micro”	
	Ec2Instance03:	
	Type: "AWS::EC2::Instance"	
	Properties:	
	ImageId: "ami-v2"	
	InstanceType: “t2.micro”	
	Ec2Instance04:	
	Type: "AWS::EC2::Instance"	
	Properties:	
	ImageId: "ami-v2"	
	InstanceType: “t2.micro”	

Introduction to Terraform

Terraform Module

```
# modules/ec2/main.tf

variable “count” {}
variable “env” {}
variable “ami” {}
variable “instance_type” {}

resource "aws_instance" "instance" {
  count         = “${var.count}”
  env           = “${var.env}”
  ami           = “${var.ami}”
  instance_type = “${var.instance_type}”
}
```

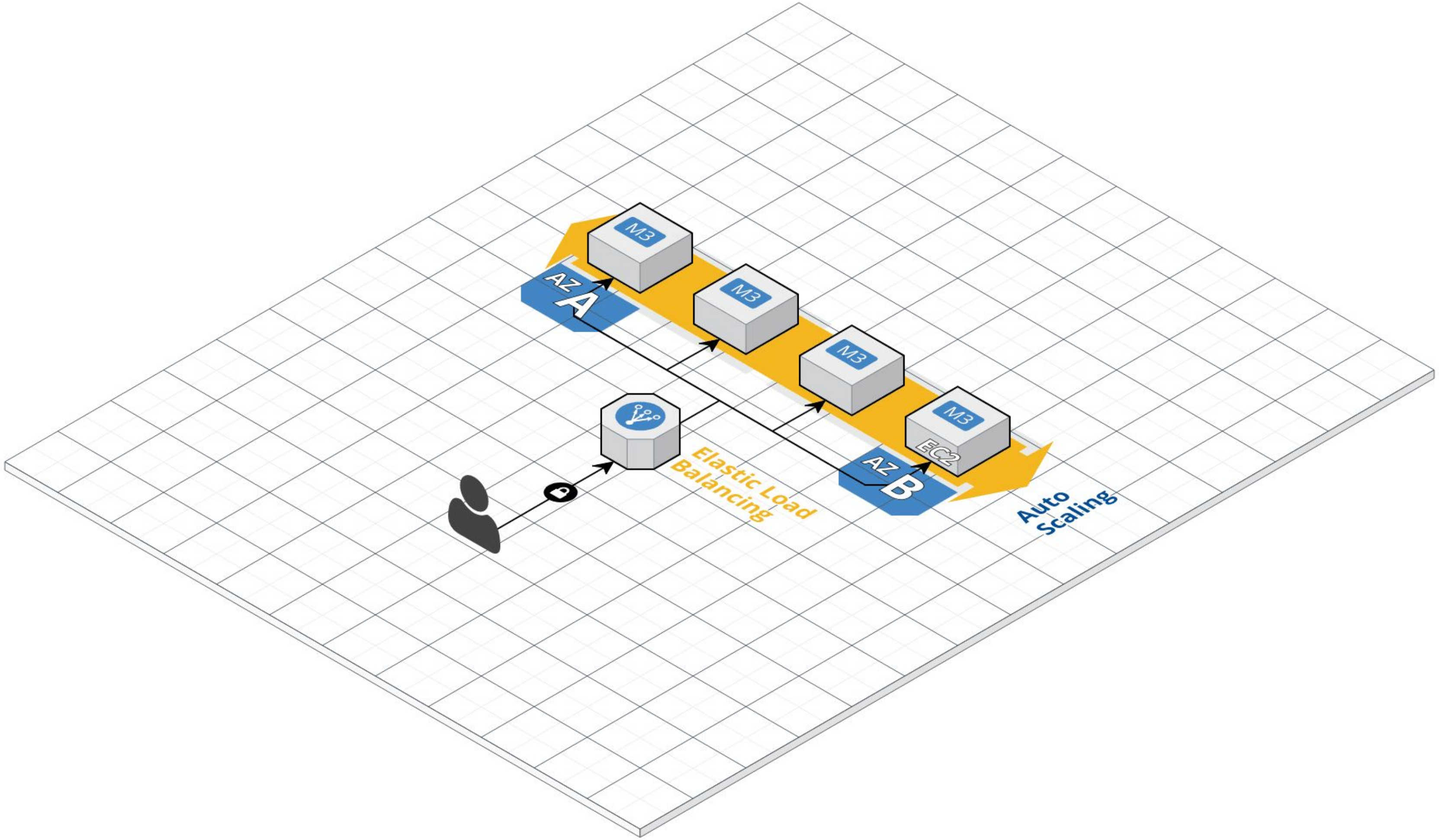
```
# stg/app_demo/main.tf
module “ec2” {
  source        = “modules/ec2”
  Count         = 1
  env           = “stg”
  ami           = “ami_v1”
  Instance_type = “t2.micro”
}
```

```
# prod/app_demo/main.tf
module “ec2” {
  source        = “modules/ec2”
  count         = 2
  env           = “prod”
  ami           = “ami_v1”
  Instance_type = “m4.large”
}
```

Example: Immutable Deployment with Terraform

Immutable Infrastructure

Image Source: https://www.oreilly.com/ideas/an-introduction-to-immutable-infrastructure

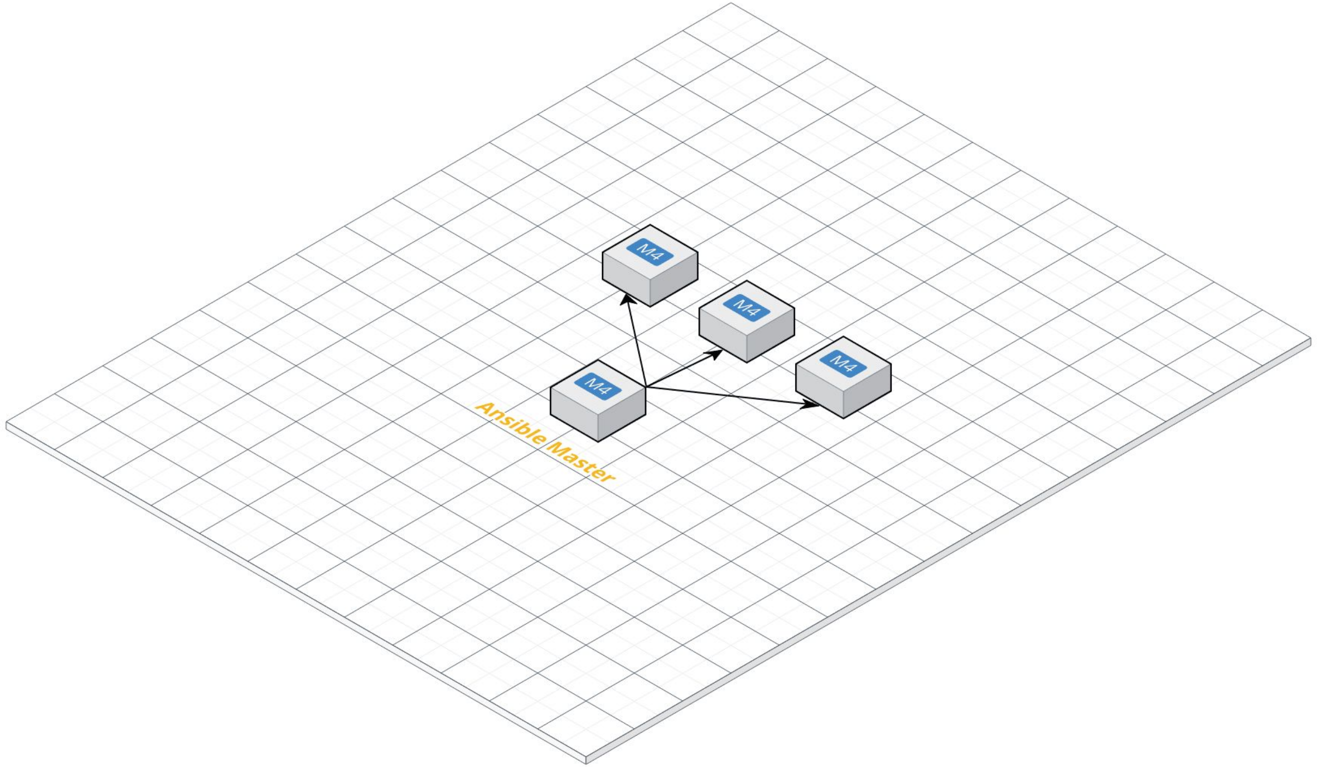

M3
M3
M3
M3
EC2
AZ A
AZ B
Elastic Load Balancing
Auto Scaling

How Can we Achieve This

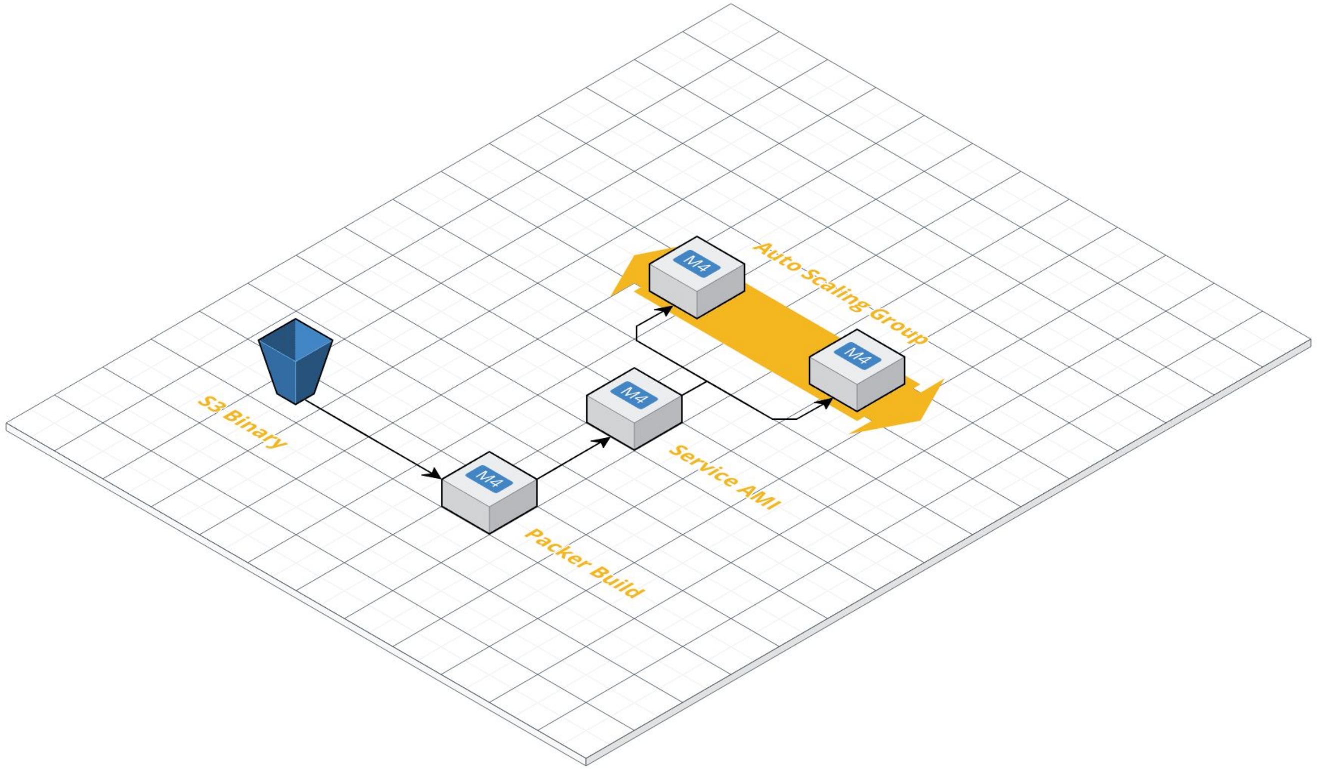
- Configuration Management.
- Infrastructure Orchestration.
- AWS Image Creator

Crate AMI (Packer + Ansible)

Preparing an Instance: Classic way



AMI Base Deployment



Infrastructure Orchestration.

Q & A