sepulsa

Manage and Maintain Your AWS Infrastructure as Code using Terraform

University of Brawijaya

July 2018

sepulsa

About me

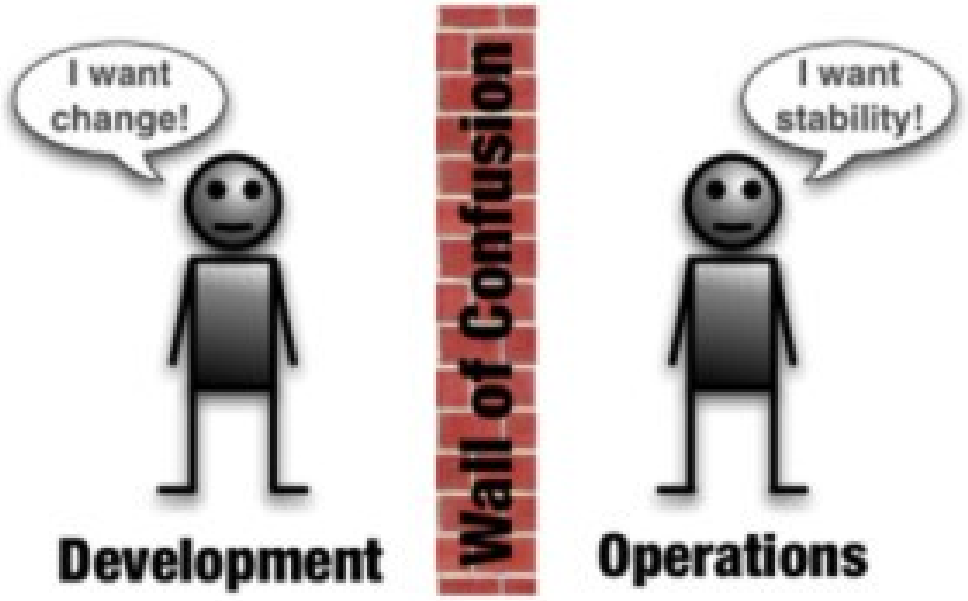
System Engineer @ Sepulsa
E-Mail : aji@sepulsa.com
Twitter : @bayyuaji

sepulsa

Agenda

1. Introduction : The rise of DevOps
2. What is Infrastructure as Code ?
3. Why codify your infrastructure ?
4. Comparison of Infrastructure as Code tools
5. Demo & Conclusion

Introduction : The rise of DevOps



https://www.oxagile.com/

sepulsa

Introduction : The rise of DevOps

https://www.oreilly.com/learning/why-use-terraform

sepulsa

Introduction : The rise of DevOps

Stratosphere (source: Namoliang via Pixabay)

sepulsa

What is Infrastructure as Code ?

The idea behind infrastructure as code (IAC) is that you write and execute code to define, deploy, and update your infrastructure.

https://www.oreilly.com/learning/why-use-terraform

sepulsa

Why codify your infrastructure ?

- A lot of complexity
- Managing resources and dependencies
- Deal with malfunctioning parts
- Perform upgrade and roll-backs

https://www.oreilly.com/learning/why-use-terraform

sepulsa

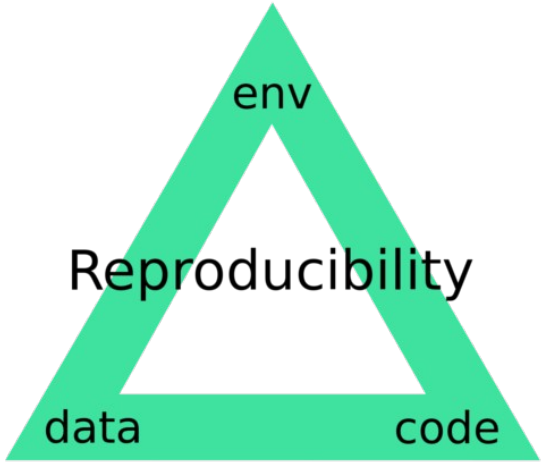
Why codify your infrastructure ?

kiwitravelwriter.files.wordpress.com

Pinterest.com

sepulsa

Why codify your infrastructure ?



Netflow.io

Devero.com

Dreamstime.com

sepulsa

Comparison of Infrastructure as Code tools

	Source	Type	Language	Cloud Support	Initial Release
Ansible	Open	Config Mgmt	Procedural	Any	2011
CloudFormation	Closed	Provisioning	Declarative	AWS	2012
Terraform	Open	Provisioning	Declarative	Any	2014

sepulsa

Procedural Language Vs Declarative Language

Ansible	Terraform

- ec2:
count: 10
image: ami-40d28157
instance_type: t2.micro
	resource “aws_instance” “demo” {
	count = 10
	ami = “ami-40d28157”
	instance_type = “t2.micro”
	}

https://www.oreilly.com/learning/why-use-terraform

sepulsa

Procedural Language Vs Declarative Language

Ansible	Terraform

- ec2:
count: 5
image: ami-40d28157
instance_type: t2.micro
	resource “aws_instance” “demo” {
	count = 15
	ami = “ami-40d28157”
	instance_type = “t2.micro”
	}

https://www.oreilly.com/learning/why-use-terraform

sepulsa

Procedural Language Vs Declarative Language

```
> terraform plan

+ aws_instance.example.11
    ami:                        "ami-40d28157"
    instance_type:              "t2.micro"
+ aws_instance.example.12
    ami:                        "ami-40d28157"
    instance_type:              "t2.micro"
+ aws_instance.example.13
    ami:                        "ami-40d28157"
    instance_type:              "t2.micro"
+ aws_instance.example.14
    ami:                        "ami-40d28157"
    instance_type:              "t2.micro"
+ aws_instance.example.15
    ami:                        "ami-40d28157"
    instance_type:              "t2.micro"

Plan: 5 to add, 0 to change, 0 to destroy.
```

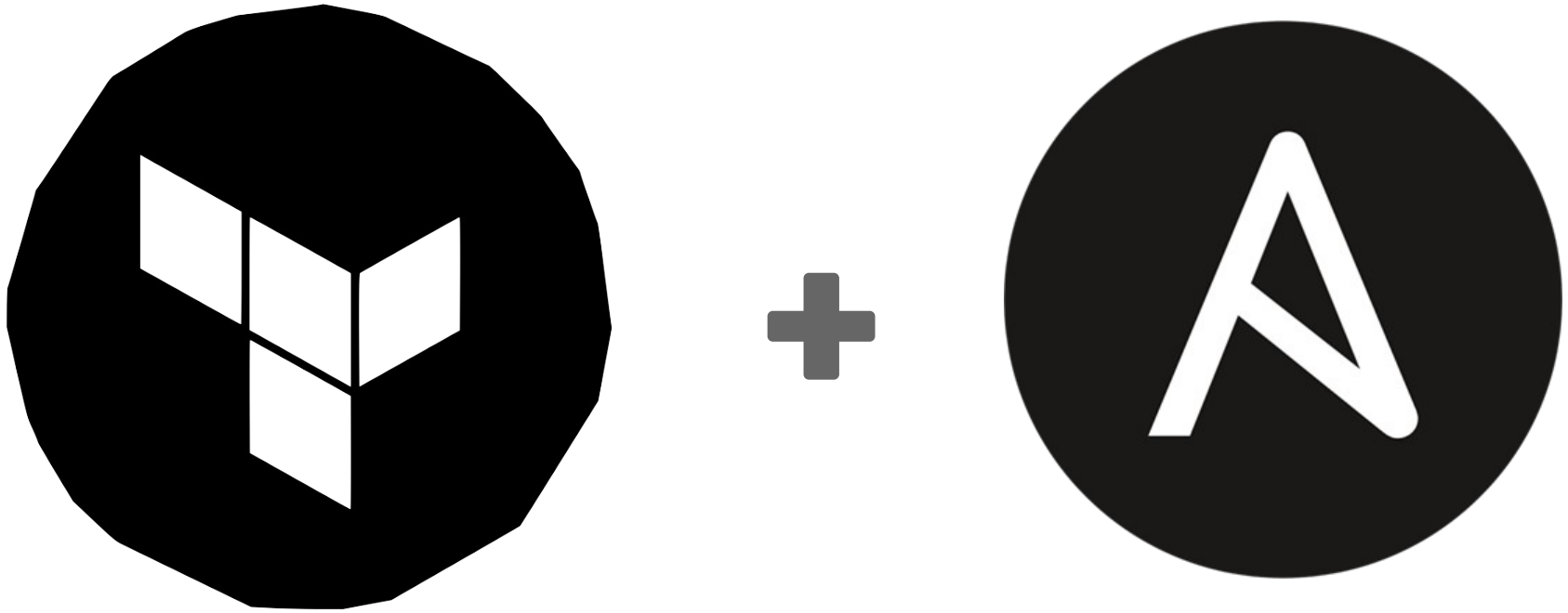
https://www.oreilly.com/learning/why-use-terraform

sepulsa

Using Ansible or Terraform ?

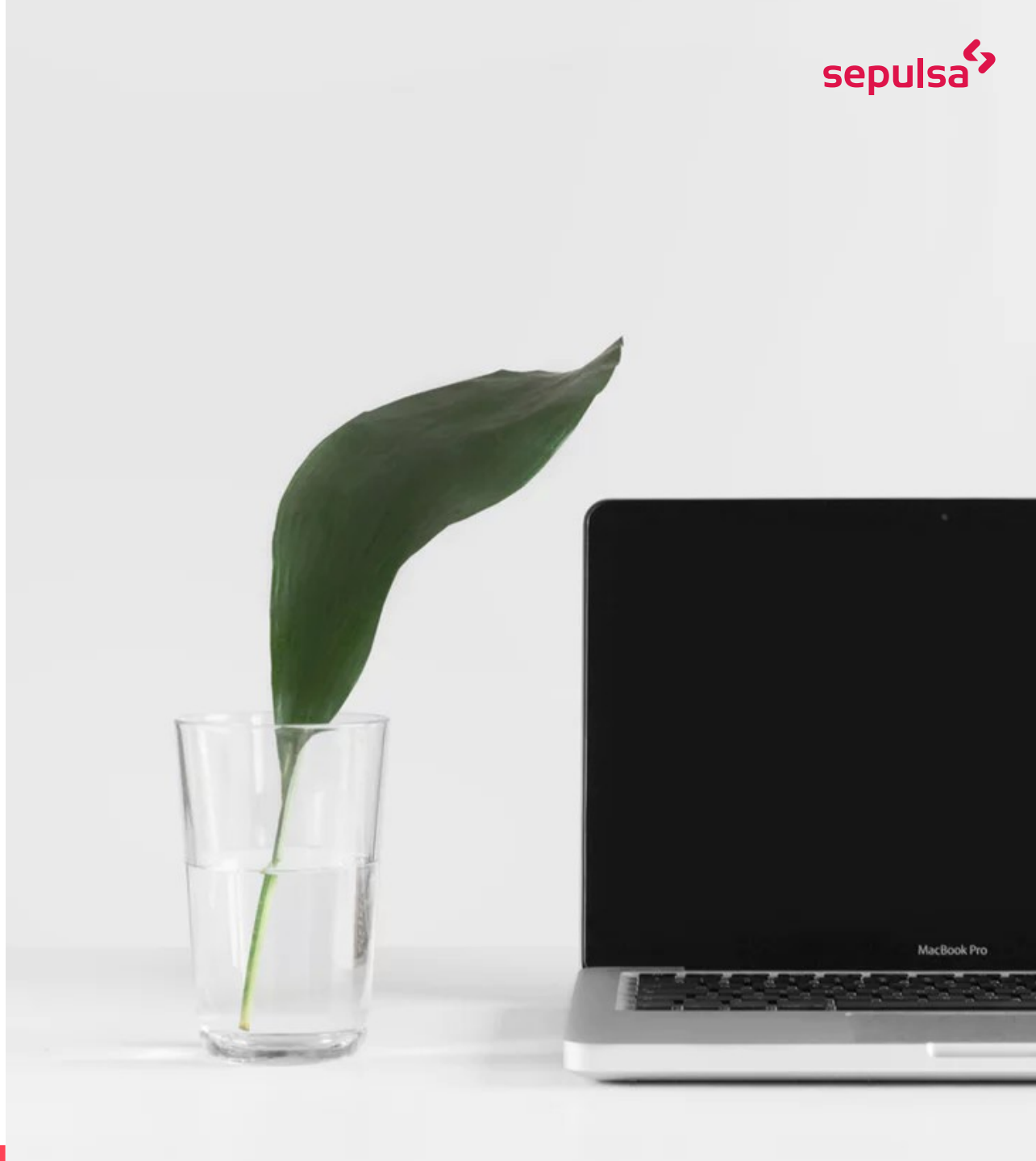
sepulsa

Demo



Conclusion

Does Terraform fit your criteria, too?



Q&A