

Programando mi infraestructura usando IaC - Demo

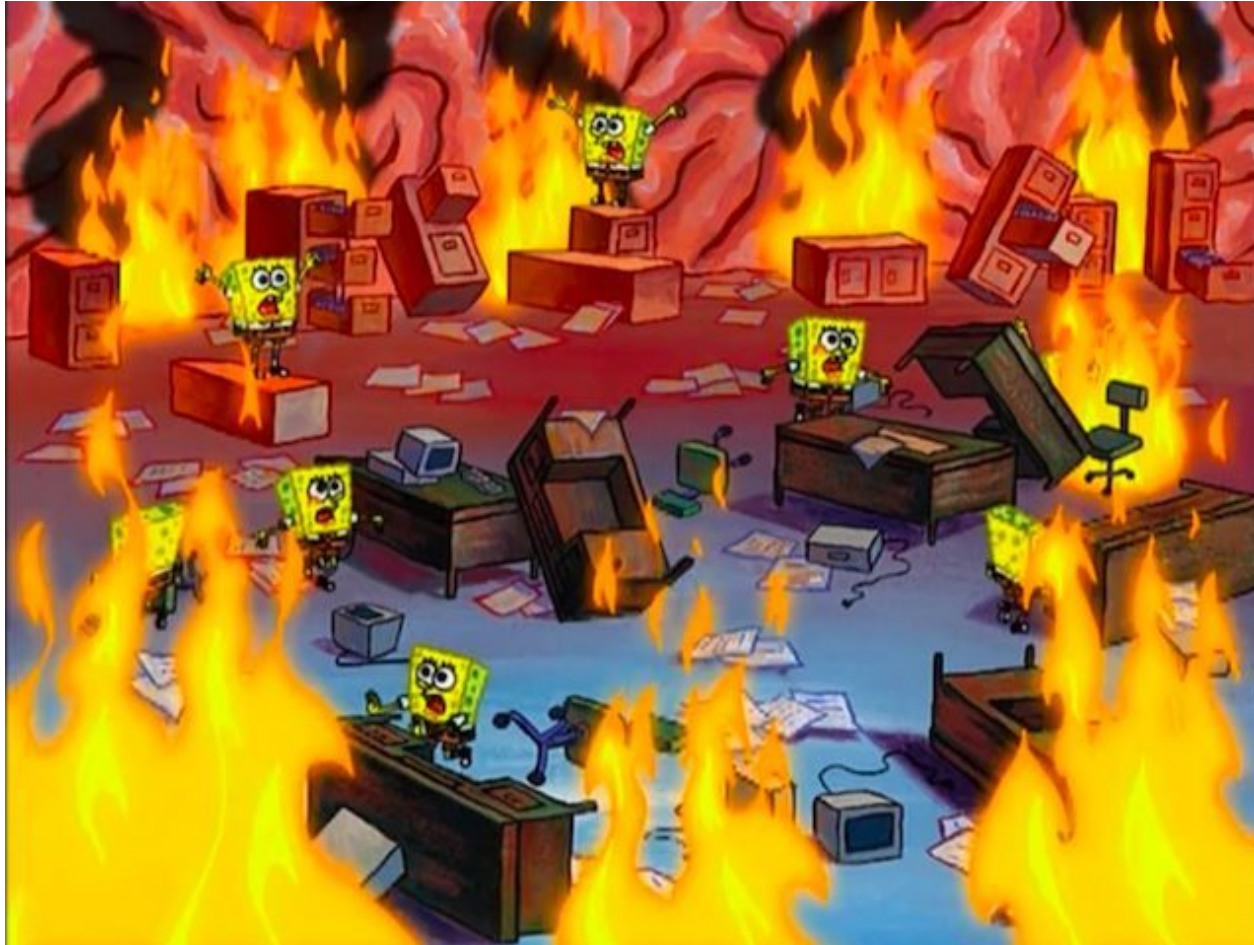
Infrastructure as Code

IaC - Definición

Es un enfoque que nos permite:

- Gestionar y preparar la infraestructura a través del código
- Usar archivos de configuración repetibles
- Generar entornos de implementación consistentes para el desarrollo de CI/CD

Sin IaC



Con laC



IaC - Características

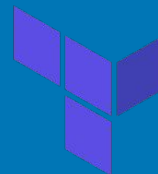
- Versionable
- Repetible
- Reusable
- Auto-documentada
- Cambios visibles



ANSIBLE



Crossplane



Vagrant





















CHEF

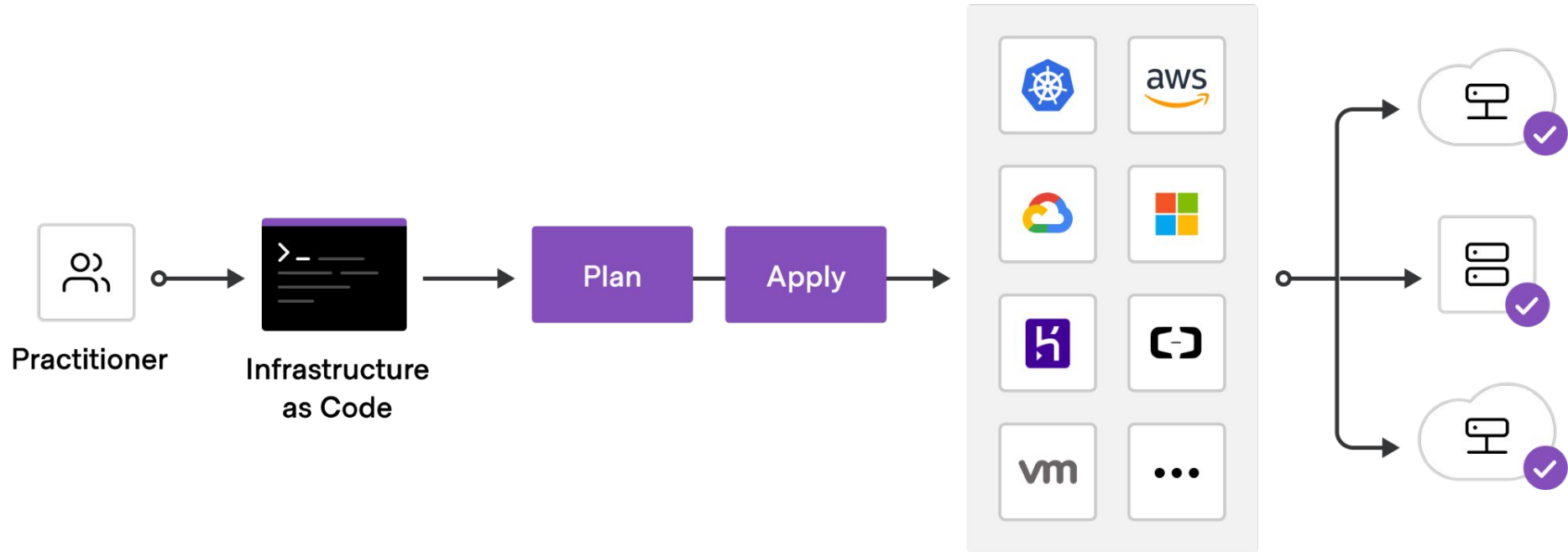




Terraform

Providers

 AWS	 Azure	 Google Cloud Platform
 Kubernetes	 Alibaba Cloud	 Oracle Cloud Infrastructure
 Active Directory by: hashicorp	 Archive by: hashicorp	 AWS Cloud Control by: hashicorp
 Azure Active Directory by: hashicorp	 Azure Stack by: hashicorp	 Boundary by: hashicorp
 Cloudinit by: hashicorp	 Consul by: hashicorp	 DNS by: hashicorp
 External by: hashicorp	 Google Beta by: hashicorp	 Google Workspace by: hashicorp



Demo

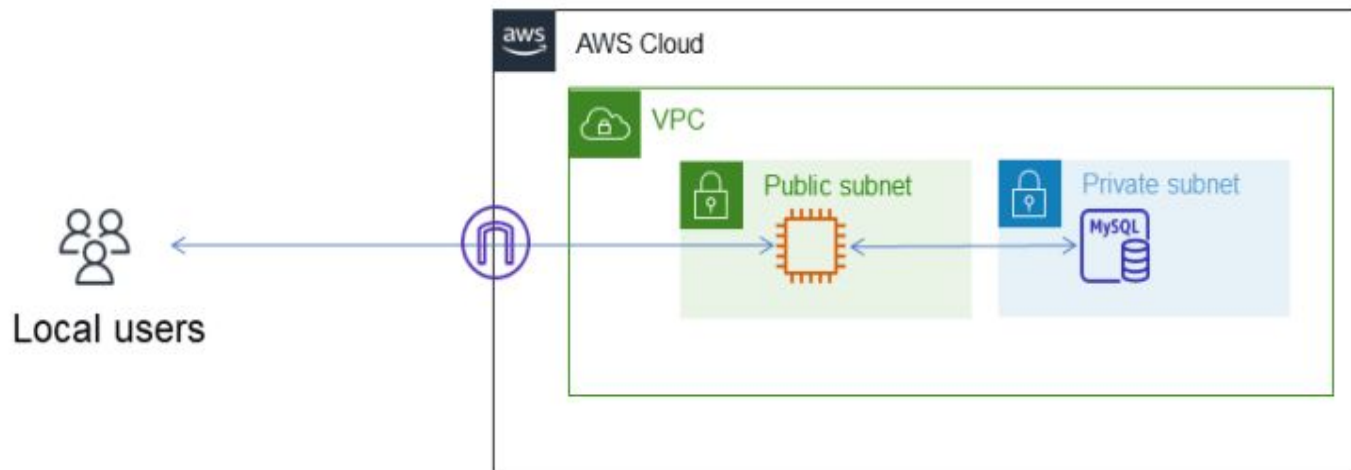
Objetivo

- Probar localmente
- Desplegar la infraestructura necesaria
 - Redes, Base de Datos, Instancia, ...
- Aprovisionar la infraestructura para correr un Wordpress

¿Qué desplegaremos?

- EC2 (Cómputo)
- RDS (Base de datos)
- VPC (Red virtual)
- Security Groups (Control de tráfico)
 - Para EC2
 - Para RDS
- Registro en Route 53 (DNS)
- Backups

¿Que desplegaremos?



Terraform en acción

Primero evaluamos qué nos creará terraform

```
> terraform plan
```



Terraform en acción

Primero evaluamos qué nos creará terraform

```
> terraform plan
```



Y luego, aplicamos

Terraform en acción



Prueba local con docker-compose

Veremos que tanto localmente como en la nube podemos obtener los mismos resultados.

Localmente, utilizaremos docker-compose

Prueba local con docker-compose

Veremos que tanto localmente como en la nube podemos obtener los mismos resultados.

Localmente, utilizaremos docker-compose

```
> docker-compose up -d
```



Wordpress con db local

Obtenemos el dump de la base de datos con nuestra configuración de Wordpress

```
> cd ansible/files
> docker-compose exec db bash -c \
"mysqldump -udemo -punademowp demowp | gzip > dump-demowp.sql.gz"
> docker cp db:/dump-demowp.sql.gz dump/
```



Ansible

Aprovisionamiento - Ansible

- Crearemos un usuario deploy
- Instalaremos docker y docker-compose
- Copiaremos los archivos necesarios para el funcionamiento de los servicios
- Levantaremos los servicios configurados con docker-compose

Wordpress con EC2

```
> sed \
's@http://localhost:8080@http://demo-iac.testing.mikroways.net:8080@g' \
dump-demowp.sql > new-dump.sql
```

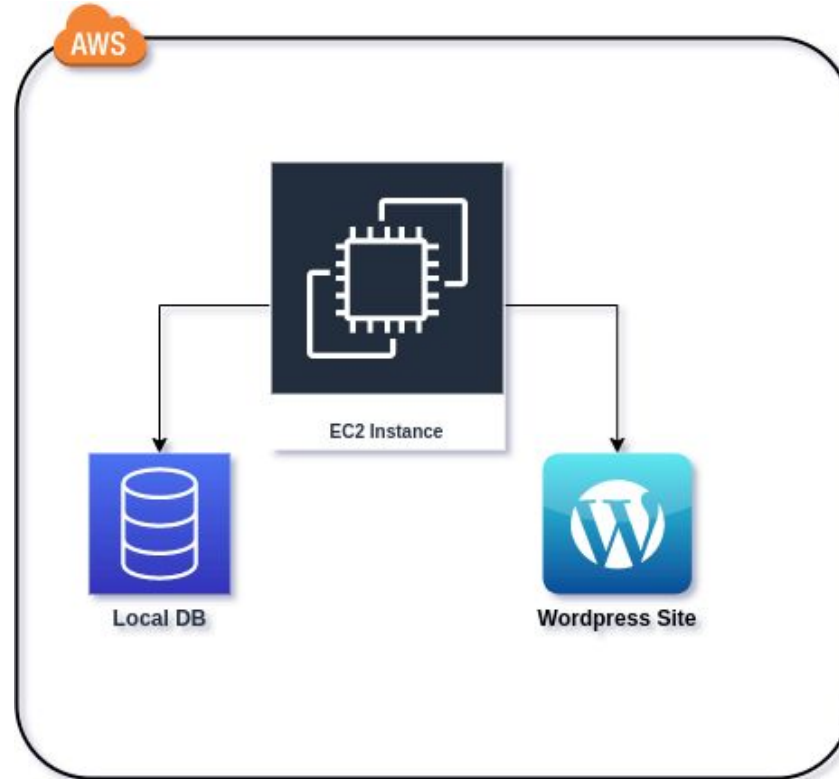


Wordpress con EC2

```
> sed \
's@http://localhost:8080@http://demo-iac.testing.mikroways.net:8080@g' \
dump-demowp.sql > new-dump.sql
```

```
> pip install -r requirements.txt
> ansible-galaxy install -fr requirements.yml && \
  ansible-galaxy collection install -fr requirements.yml
> ansible-playbook -i hosts.yml playbook.yml
```

Wordpress con EC2

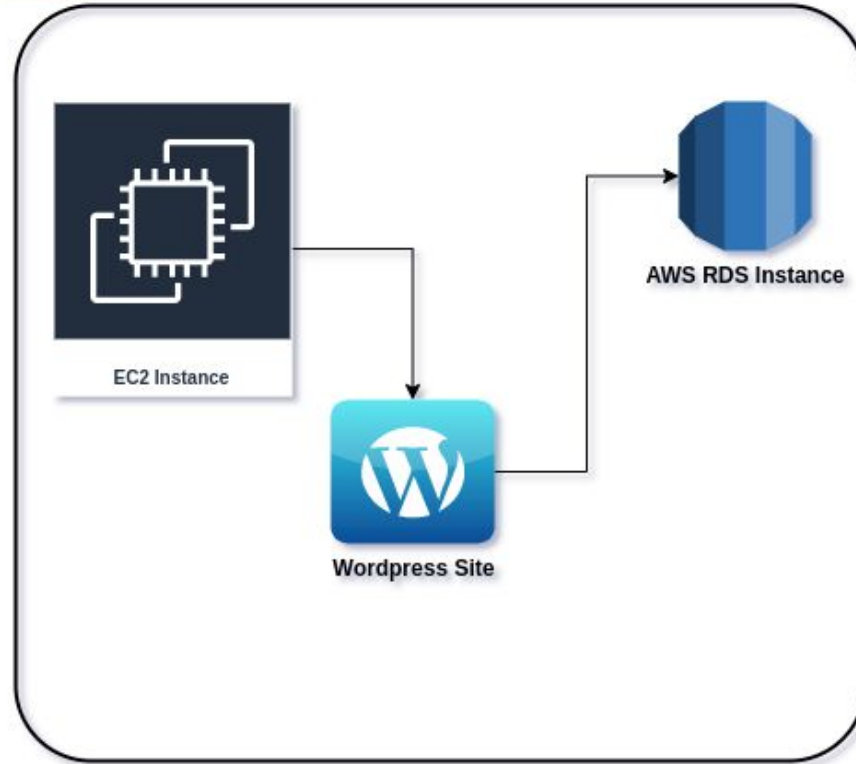


Wordpress con RDS

```
> sed \
's@http://localhost:8080@http://demo-iac.testing.mikroways.net:8081@g' \
dump-demowp.sql > new-dump.sql
```



Wordpress con RDS



Bonus track???



Serverless Framework

¿Qué desplegaremos?

- Stack de CloudFormation
 - AWS Lambda
 - Log Group en CloudWatch
 - Bucket de S3

