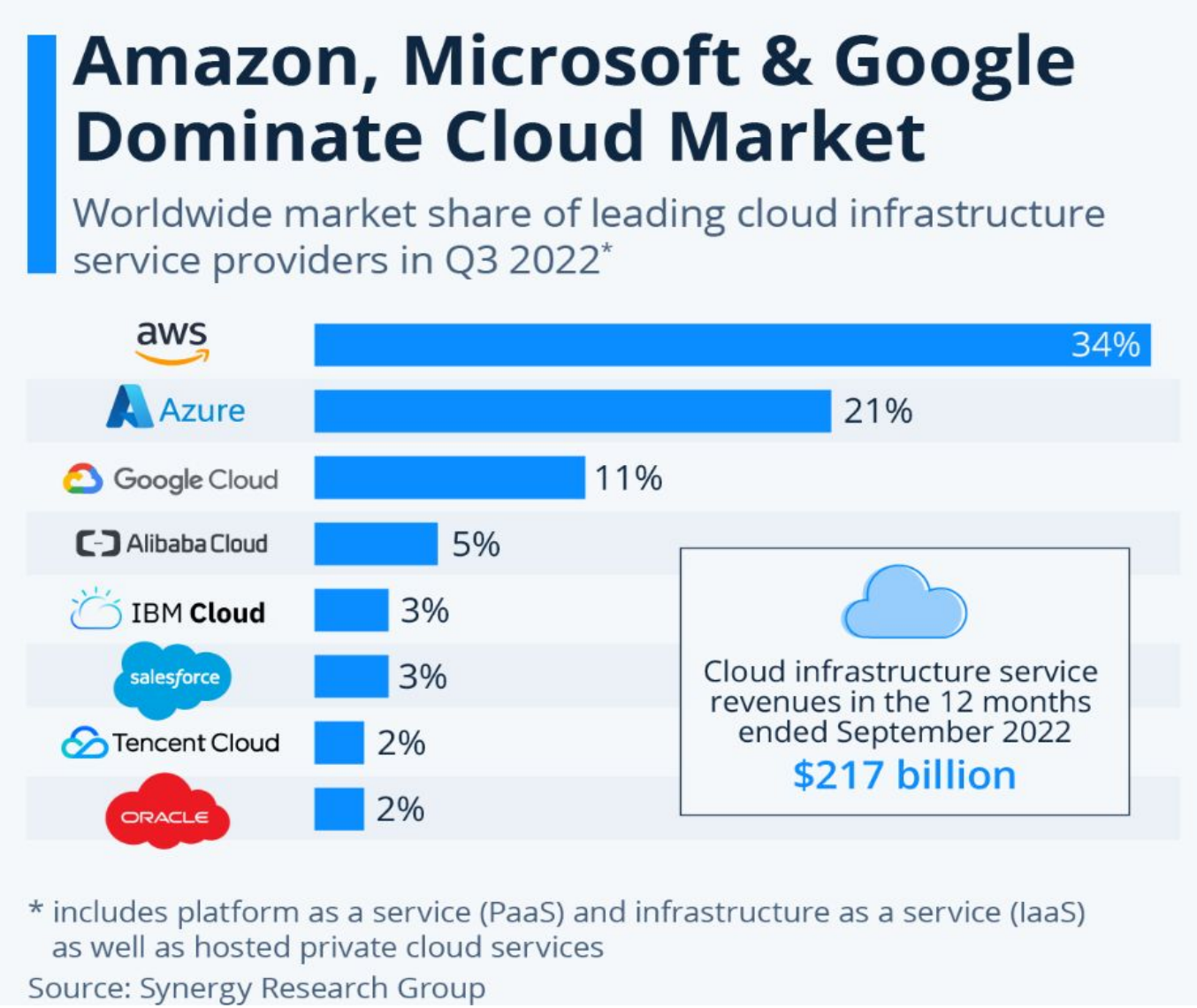
Temario

- **Cloud Computing**
  - Indicadores / Motivaciones
  - **Cloud Classic:**
    - Conceptos.
    - Modelos de Servicio y Despliegue. Seguridad. Herramientas: **Openstack**.
    - Desarrollo de caso práctico
  - **Cloud Native:**
    - Conceptos.
    - Metodologías de Trabajo.
    - Seguridad. Herramientas: **Kubernetes**.

Cloud Market share



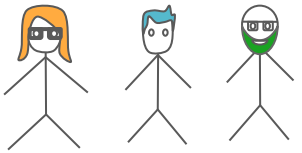
Fuente: [worldwide-market-share-of-leading-cloud-infrastructure-service-providers]

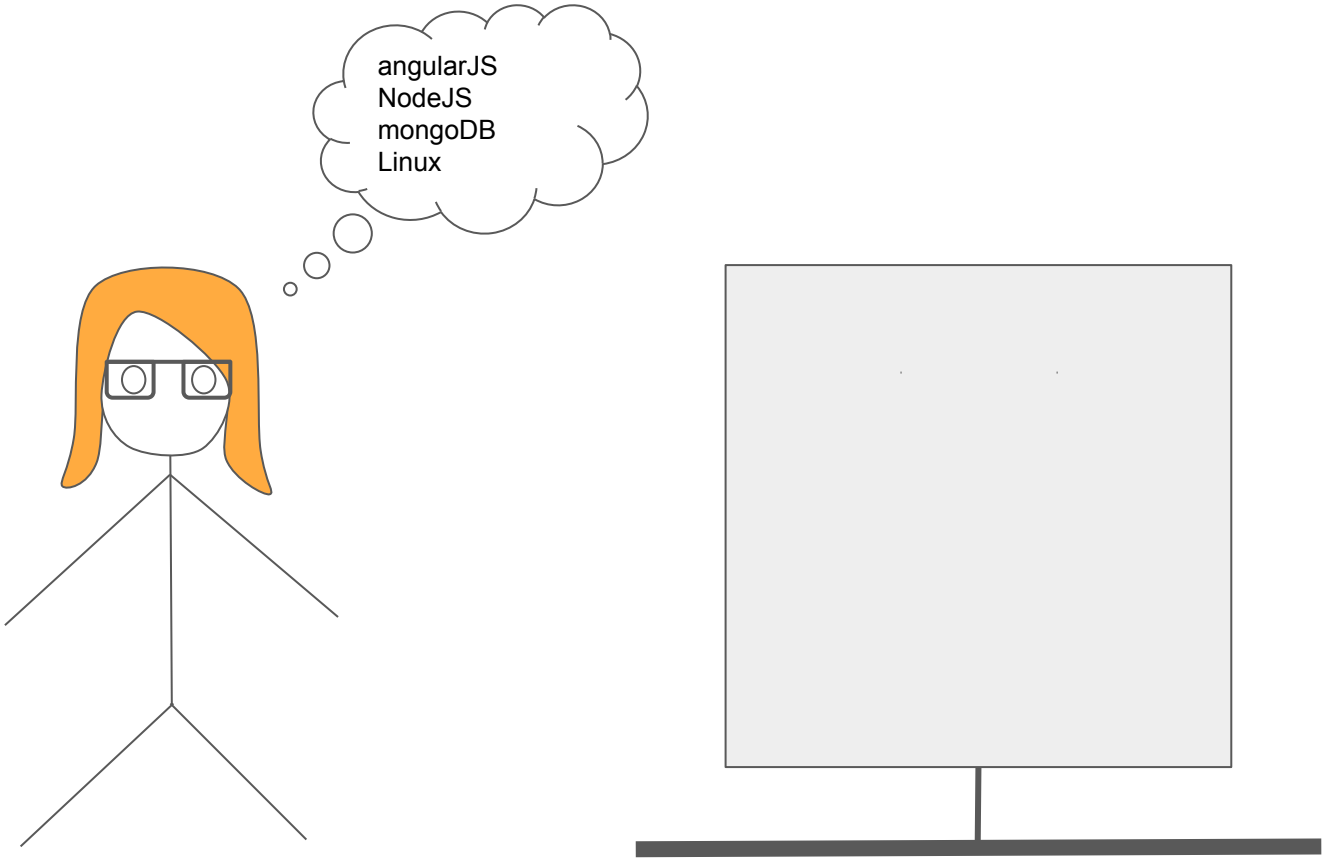
Motivos para adopción de Cloud

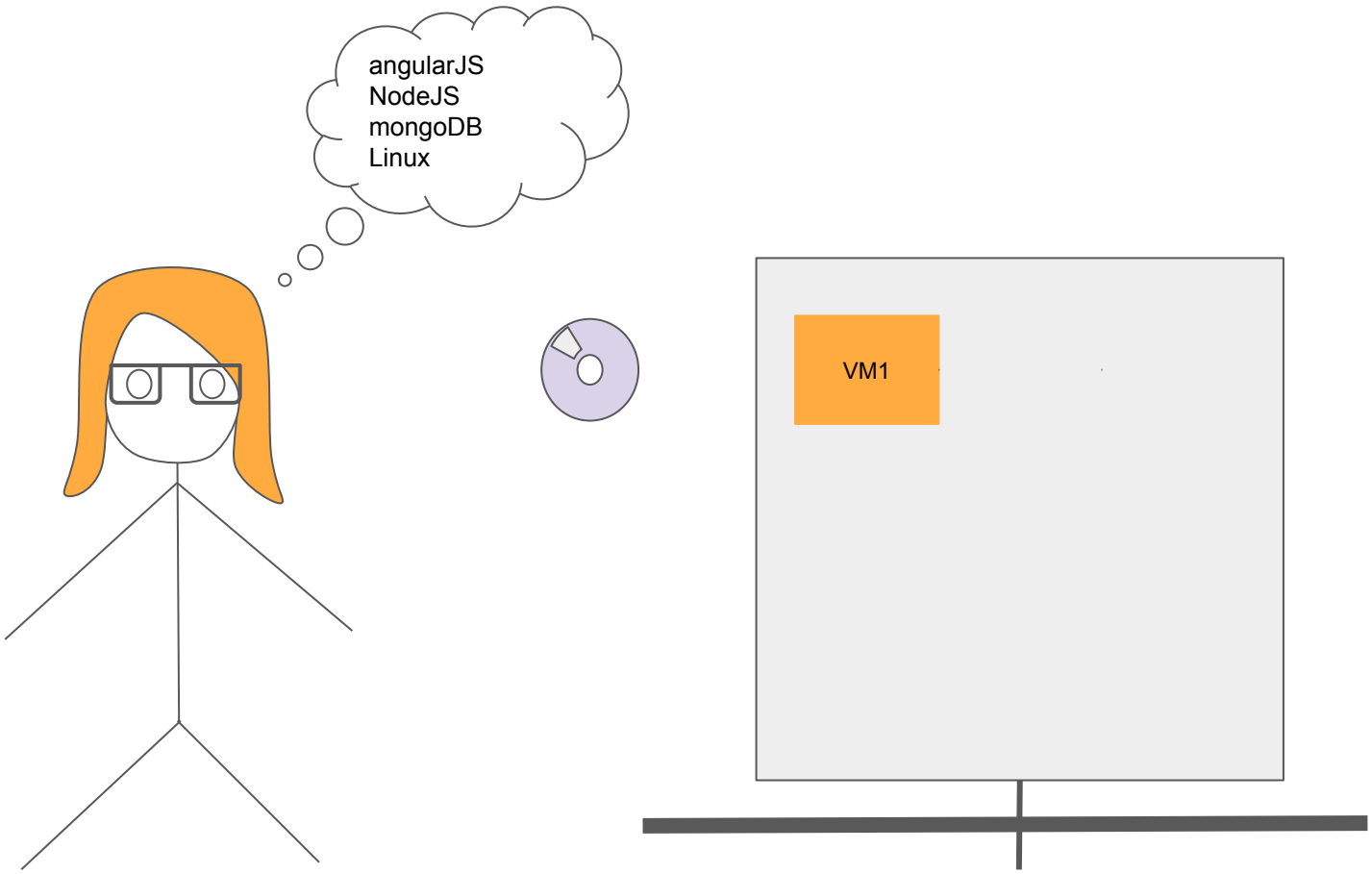
Fuentes: Businesscomputingworld.co.uk, CenturyLink, Cloudcouncil.org, Cloudcomputing-news.net, Enterprisesproject.com, Centro Europeo para la Promoción de Importaciones, Gemalto.com, Harveynash.com, Intel.com, Skyhighnetworks.com, Statista, Rightscale.com; IDC Latinamerica Investment Trends; IDC Black Book 3rd.Platform Edition; Frost&Sullivan Latin American IaaS Market, Forecast.

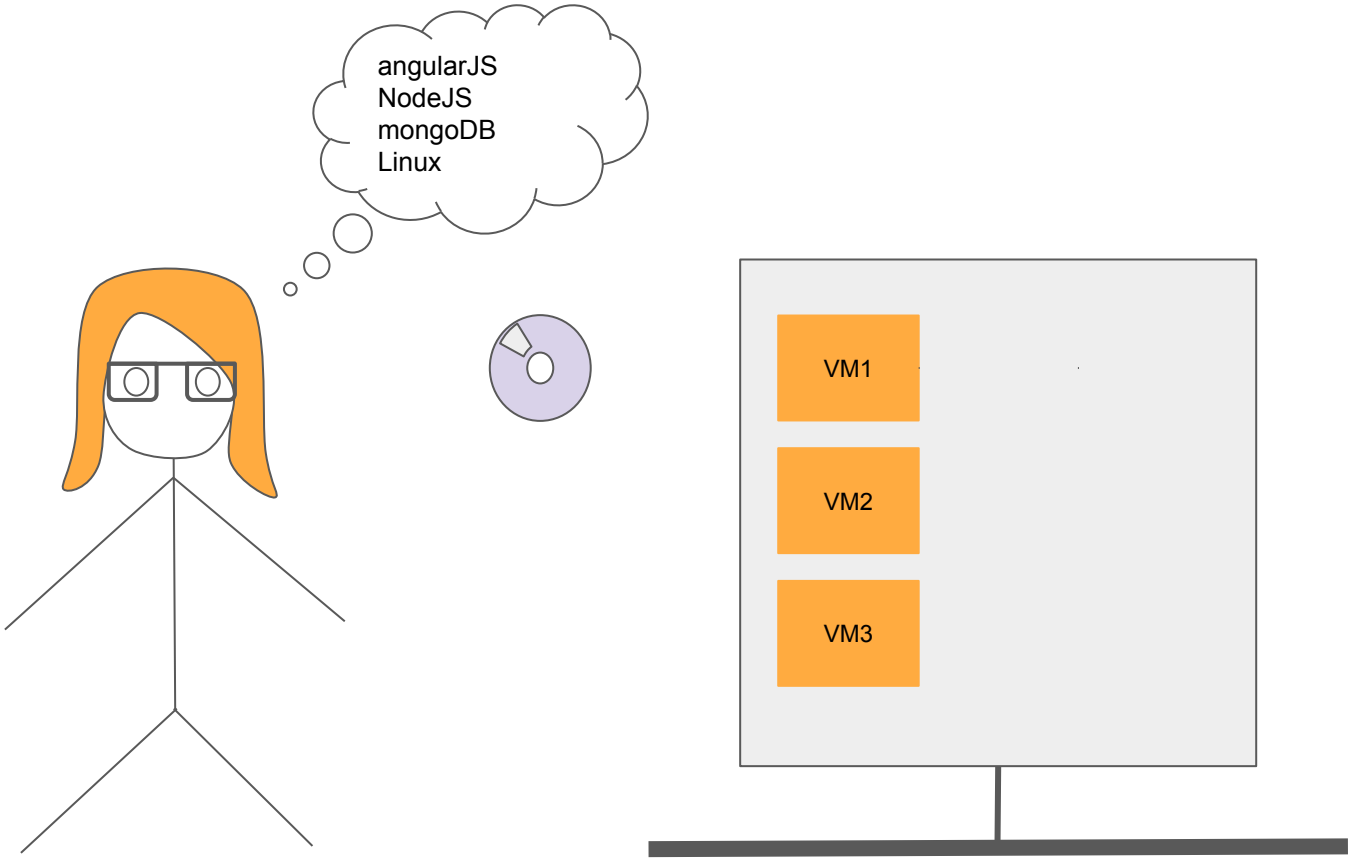
Cloud computing ...
La nube

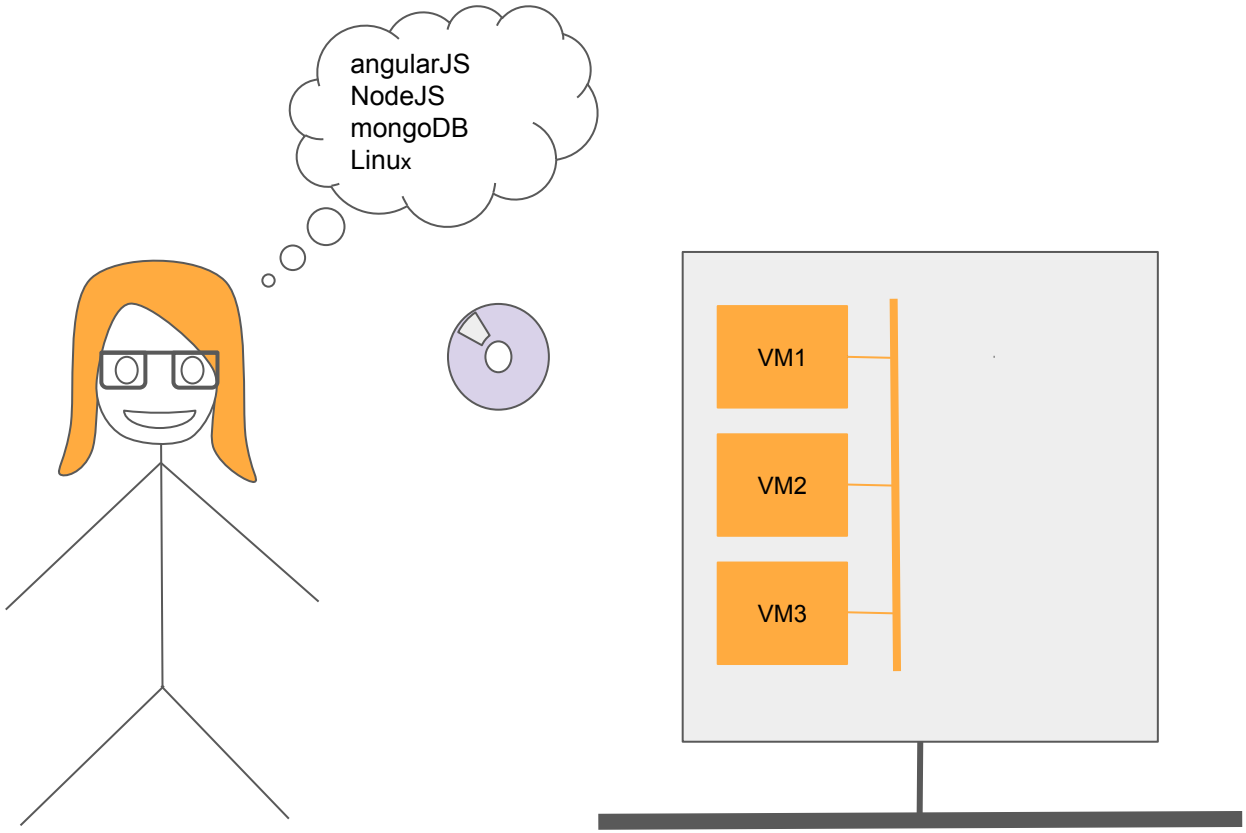
¿ lo qué ?

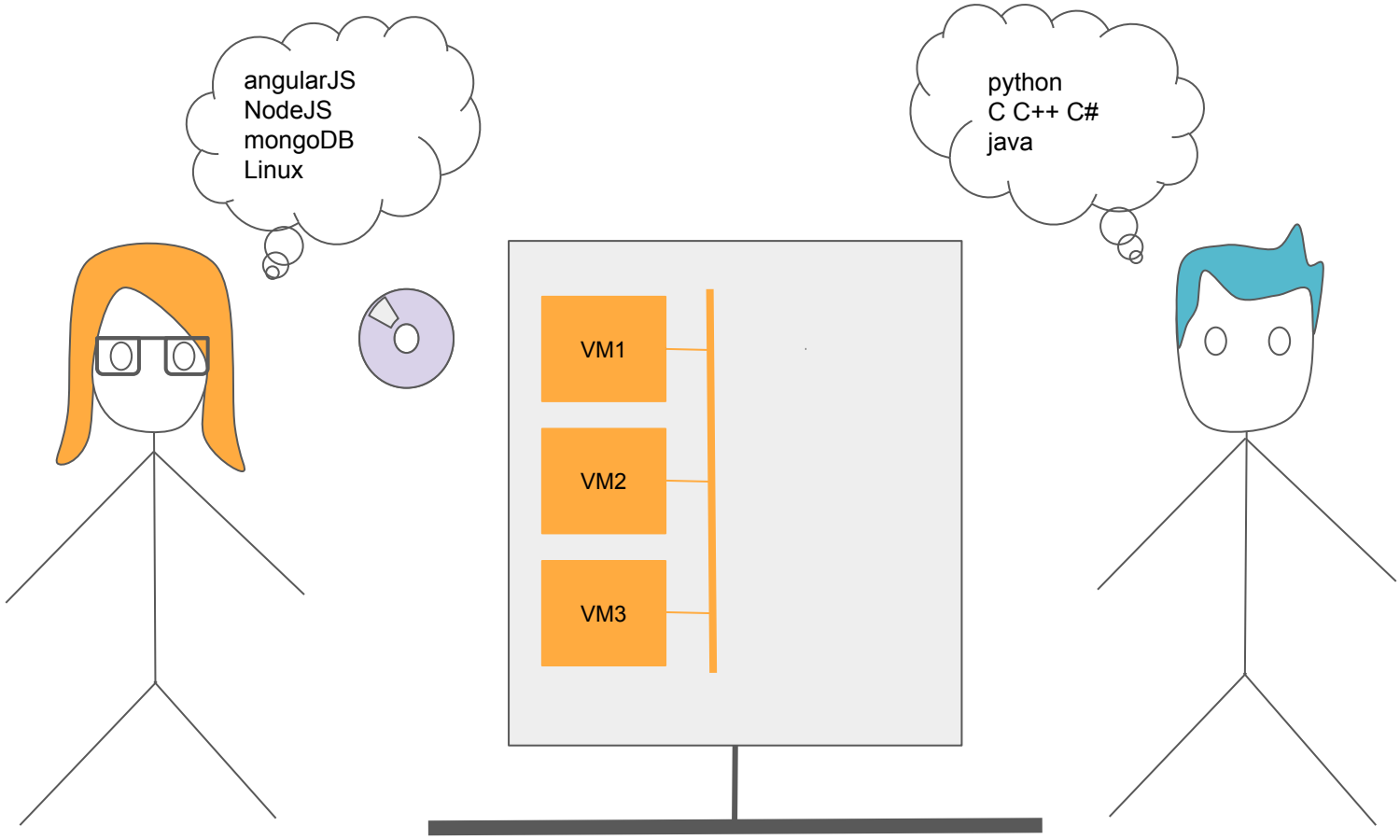


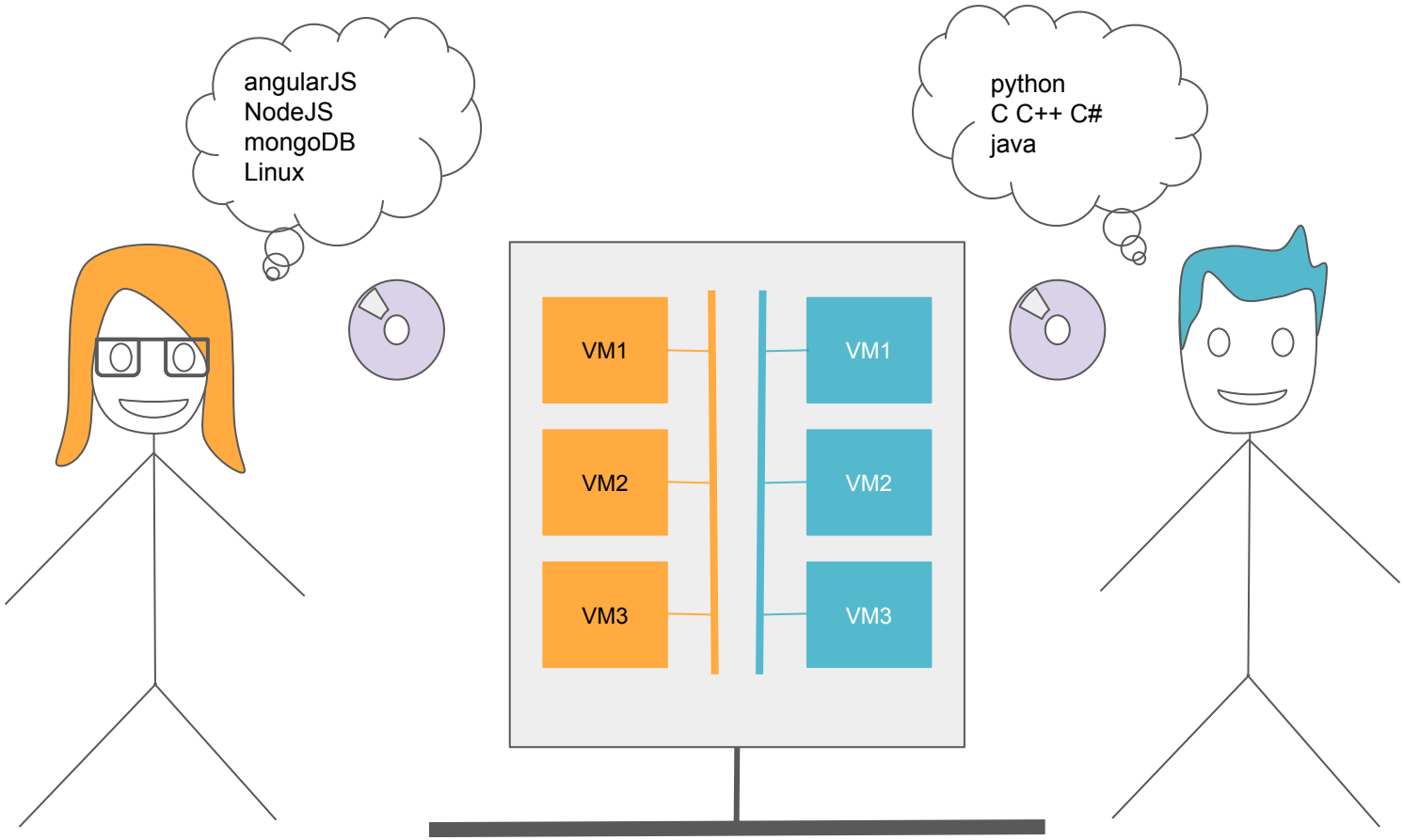

angularJS
NodeJS
mongoDB
Linux

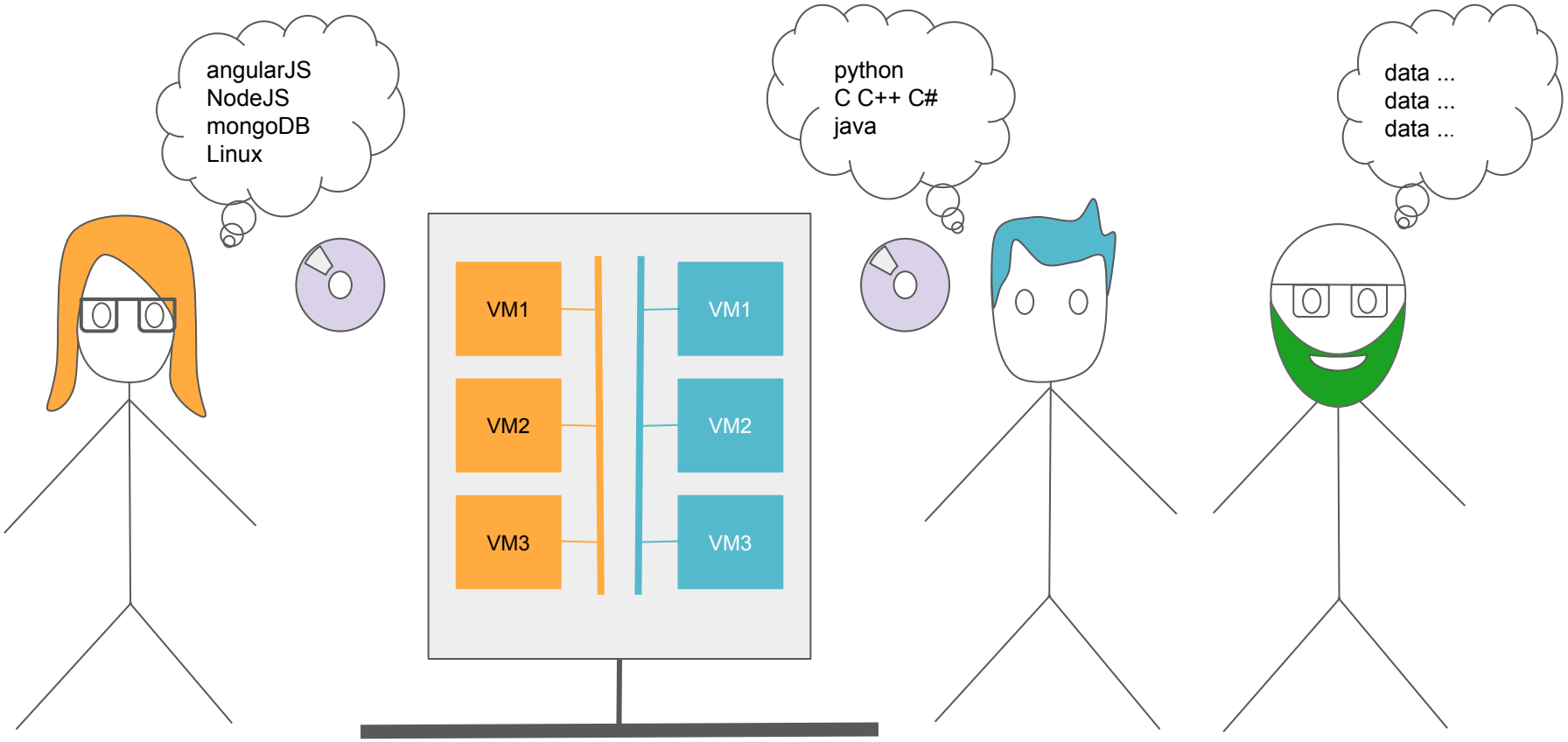

angularJS
NodeJS
mongoDB
Linux
VM1

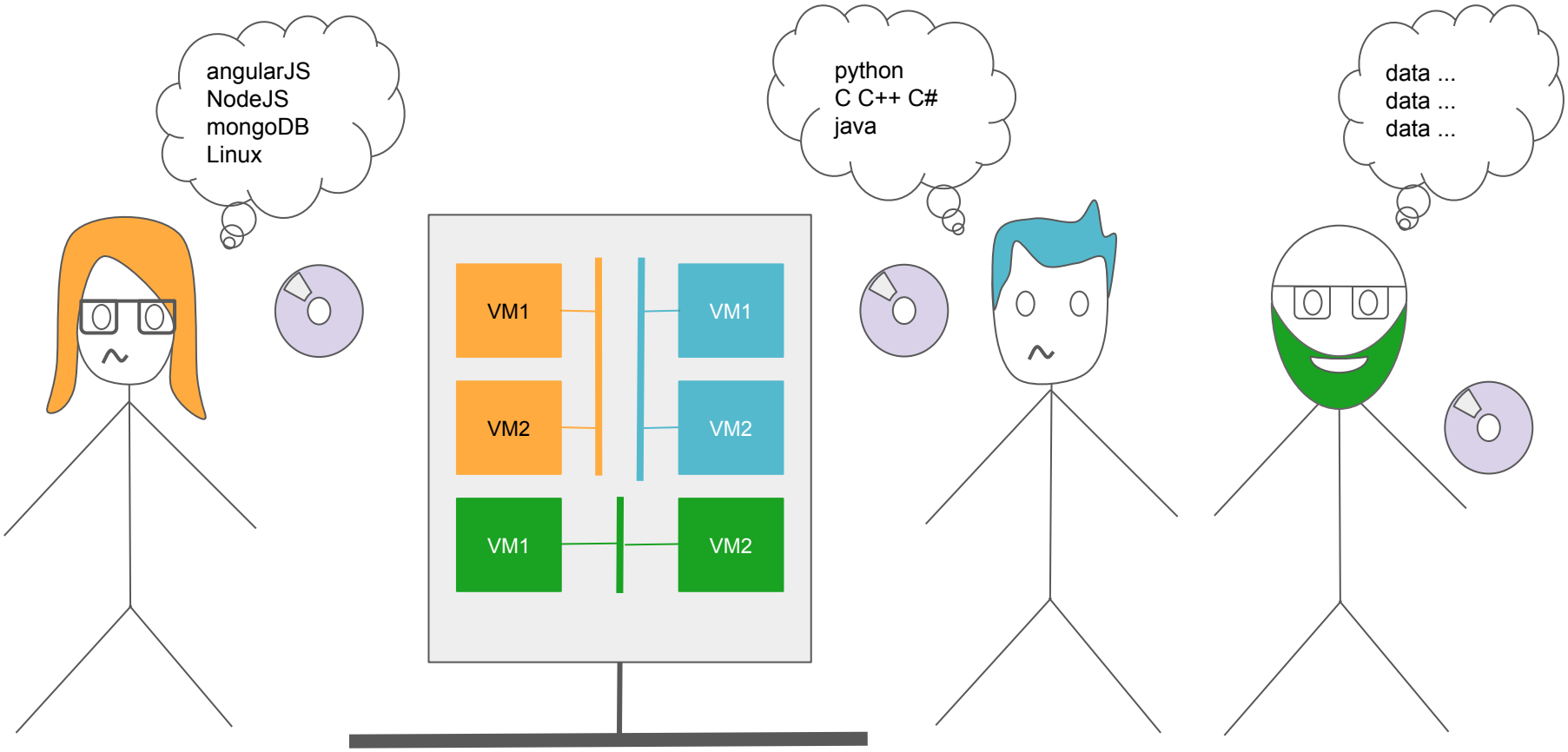

angularJS
NodeJS
mongoDB
Linux
VM1
VM2
VM3

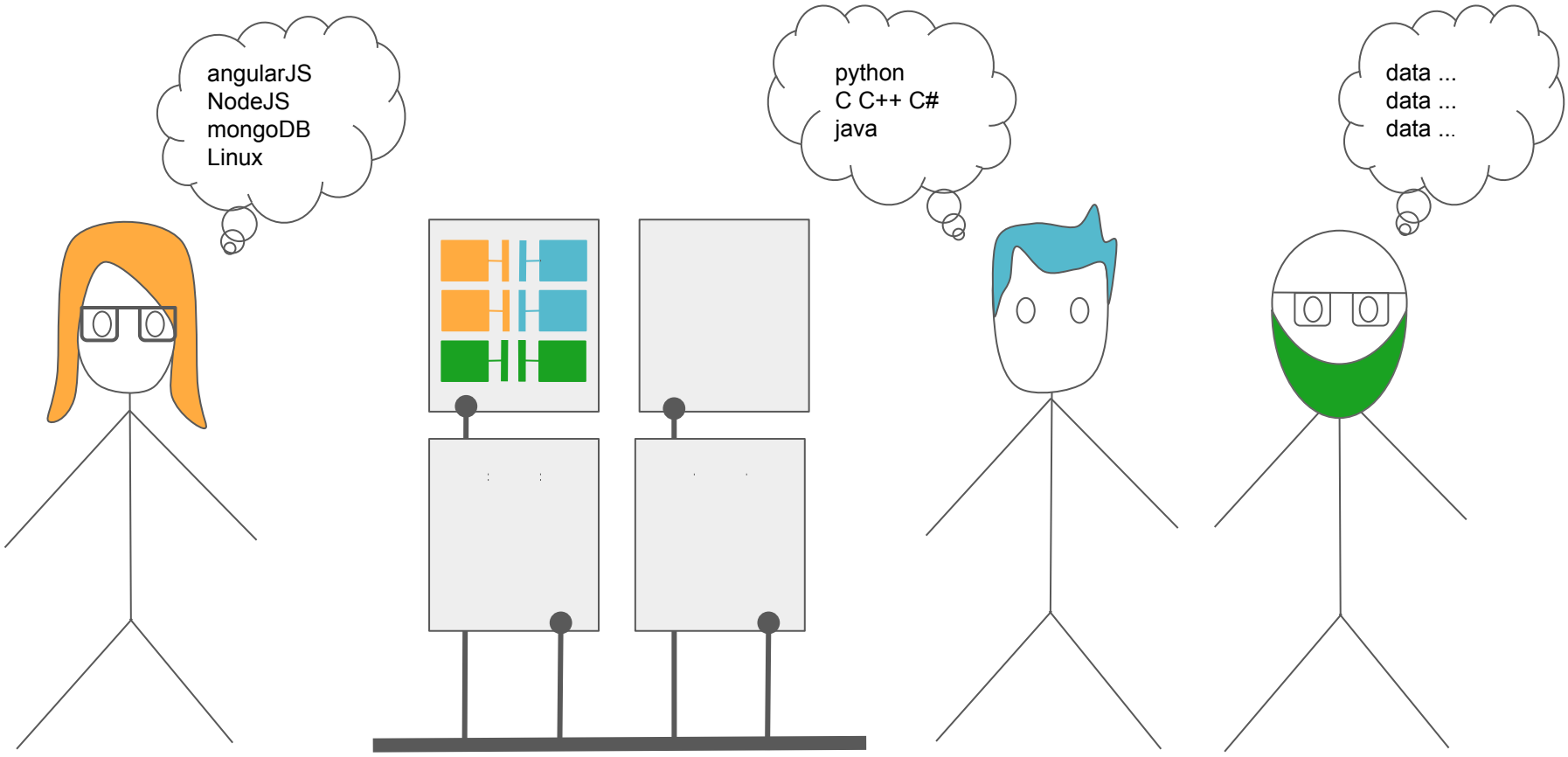

angularJS
NodeJS
mongoDB
Linux
VM1
VM2
VM3

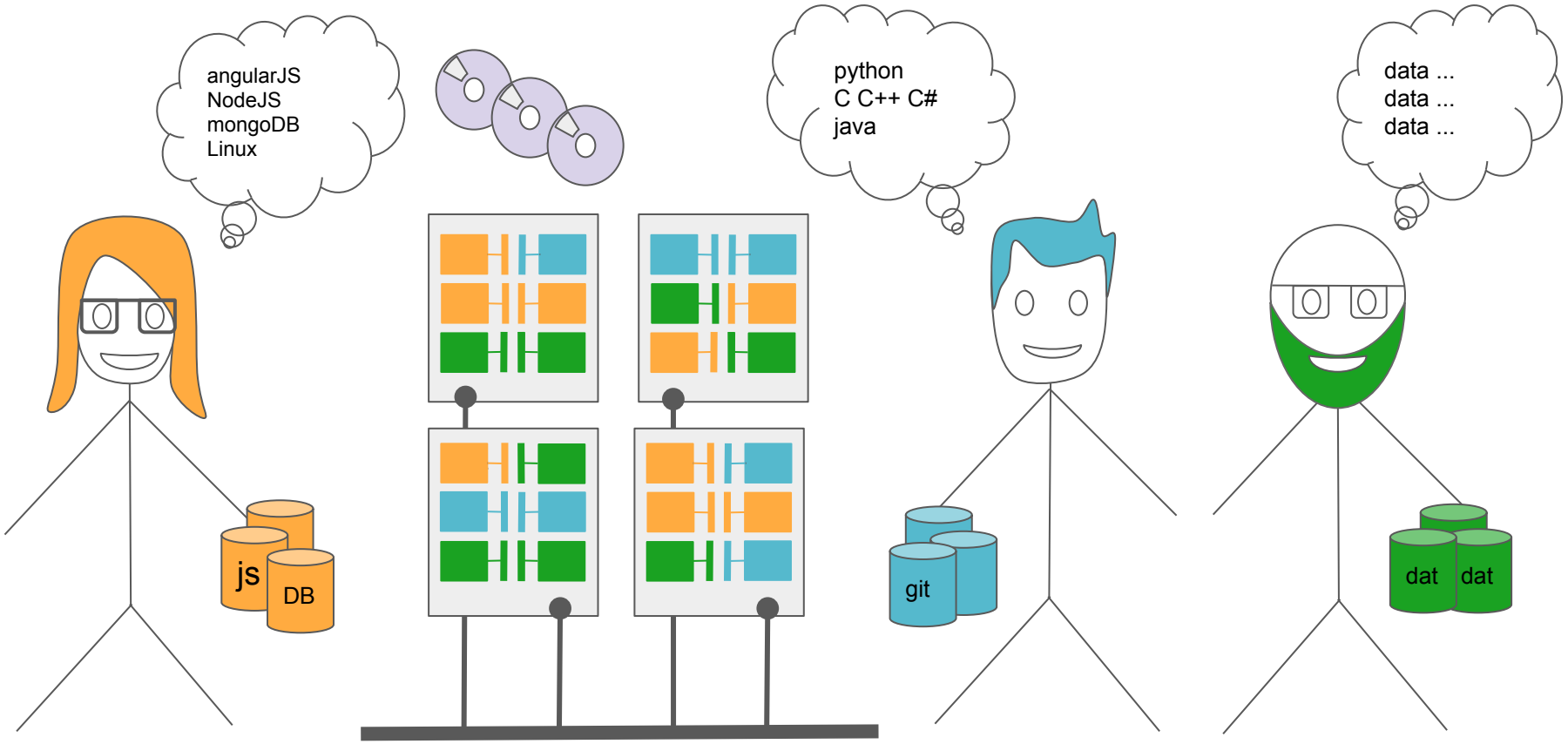

angularJS
NodeJS
mongoDB
Linux
python
C C++ C#
java
VM1
VM2
VM3


angularJS
NodeJS
mongoDB
Linux
python
C C++ C#
java
VM1
VM2
VM3
VM1
VM2
VM3


angularJS
NodeJS
mongoDB
Linux
python
C C++ C#
java
data ...
data ...
data ...
VM1
VM2
VM3
VM1
VM2
VM3

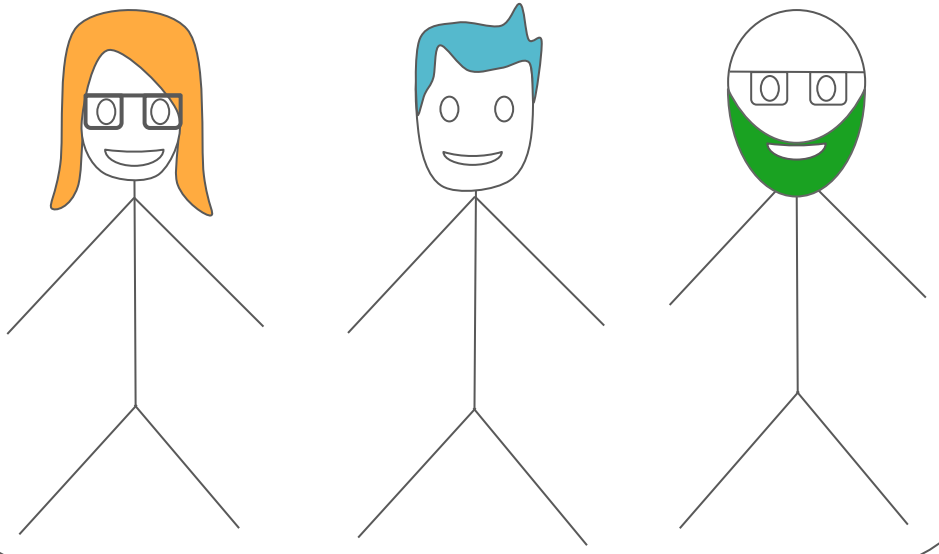

angularJS
NodeJS
mongoDB
Linux
python
C C++ C#
java
data ...
data ...
data ...
VM1
VM2
VM1
VM1
VM2
VM2


angularJS
NodeJS
mongoDB
Linux
python
C C++ C#
java
data ...
data ...
data ...


angularJS
NodeJS
mongoDB
Linux
python
C C++ C#
java
data ...
data ...
data ...
js
DB
git
dat
dat

UM Cloud

keystone - identity



nova - compute

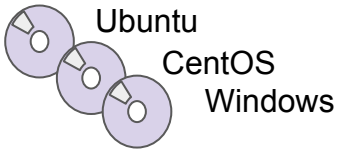
neutron - network

10.0.1.0/24

10.0.2.0/24

10.0.3.0/24

glance - image



cinder - volume

Cloud Computing: Definición

*"Uso de **recursos de computación** entregados sobre una red **como un servicio**"* (1)

"Un modelo para facilitar el acceso a **través de la red** de manera conveniente y ubicua a un conjunto compartido de r**ecursos de computación** (redes, servidores, almacenamiento, aplicaciones y servicios) que pueden ser provisionados facilmente y rapidamente con un mínimo esfuerzo de gestión o interacción por parte del proveedor de servicios " (2)

Este modelo de cloud se compone de cinco características esenciales, tres modelos de servicio y cuatro de despliegue.

Características

- Auto servicio y bajo demanda
- Acceso por red
- Compartición de recursos [multi-inquilino]
- Rápida elasticidad [escalabilidad]
- Servicio Medido [Monitoreo]

Modelos de Servicio

- **SaaS** - Software aaS
- **PaaS** - Platform aaS
- **IaaS** - Infra aaS

Modelos de Despliegue

- Nube privada
- Nube comunitaria
- Nube pública
- Nube híbrida

1 http://en.wikipedia.org/wiki/Cloud_computing

2 The NIST Definition of Cloud Computing

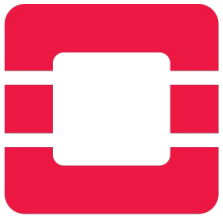
Cloud Computing = Cloud Classic “para nos”

Cloud Computing

- autoservicio
- bajo-demanda
- acceso por red
- recursos en común (multi-inquilino)
- rápida elasticidad (escalabilidad)
- servicio medido (monitoreo)

Cloud Computing

Paradigma que permite ofrecer recursos computacionales mediante Internet

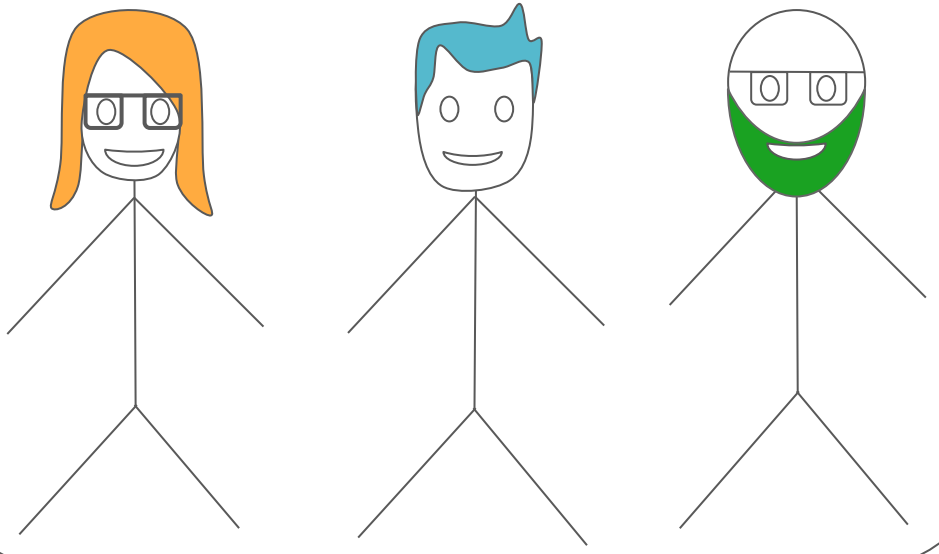


openstack®

Recursos virtuales agrupados para diseñar y gestionar nubes Publicas o privadas

UM Cloud

keystone - identity



nova - compute

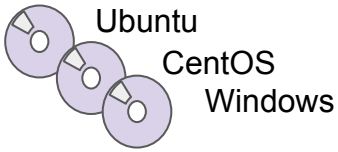
neutron - network

10.0.1.0/24

10.0.2.0/24

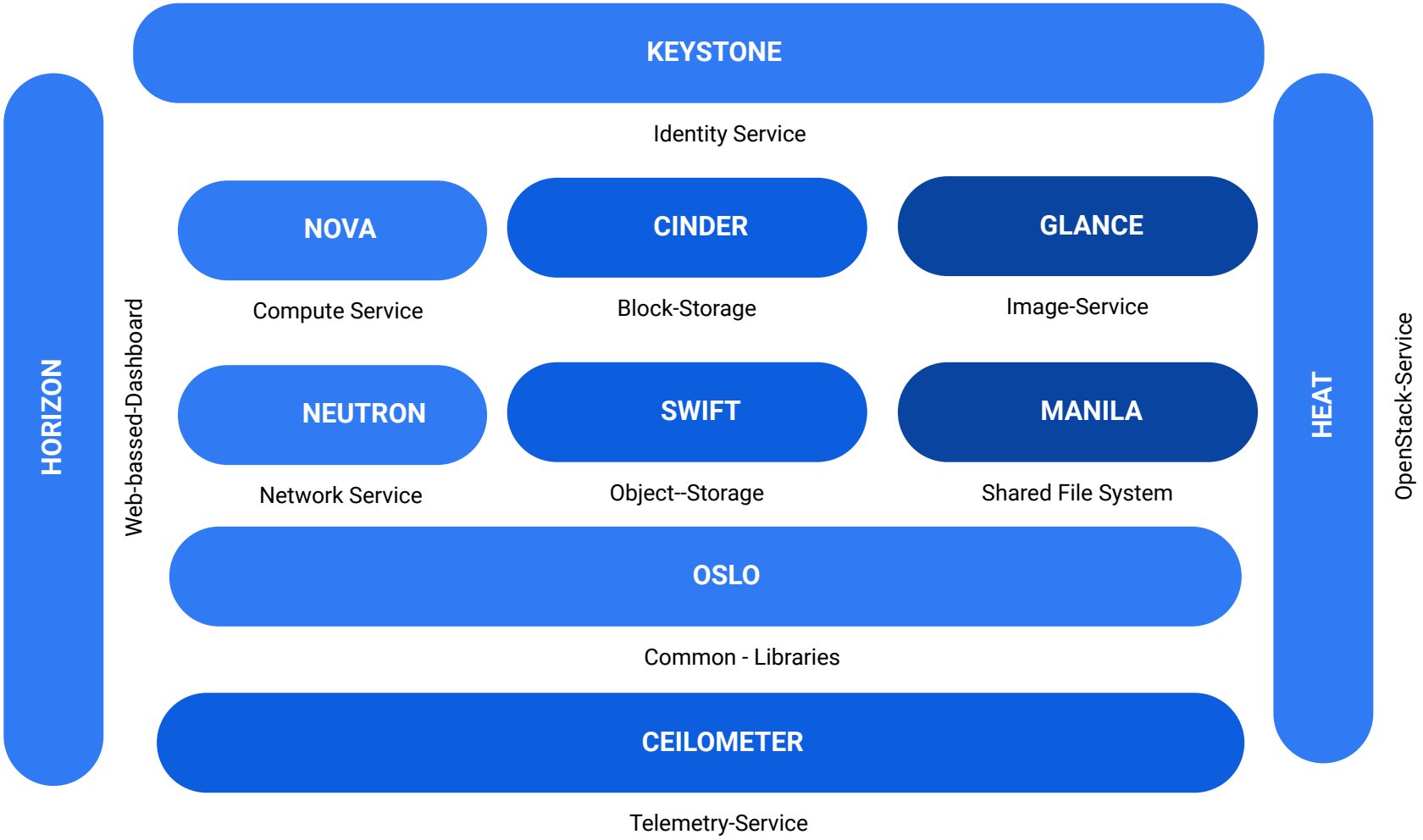
10.0.3.0/24

glance - image



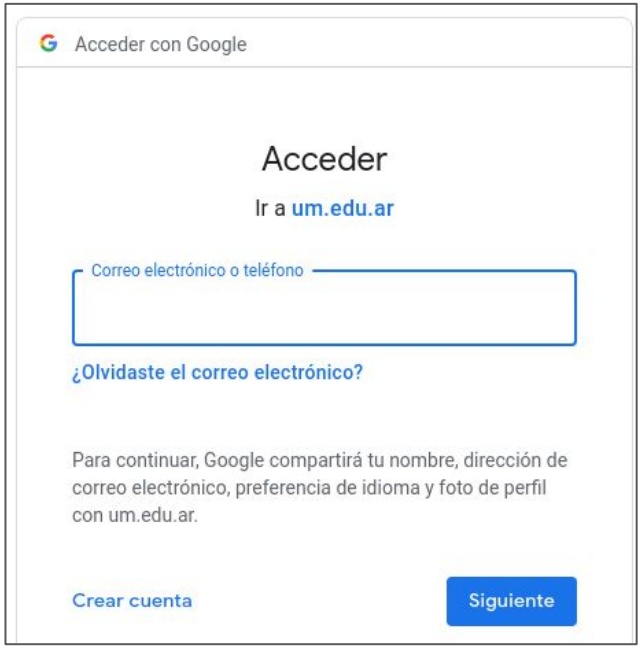
cinder - volume

Cloud Computing



UMCloud: Onboarding

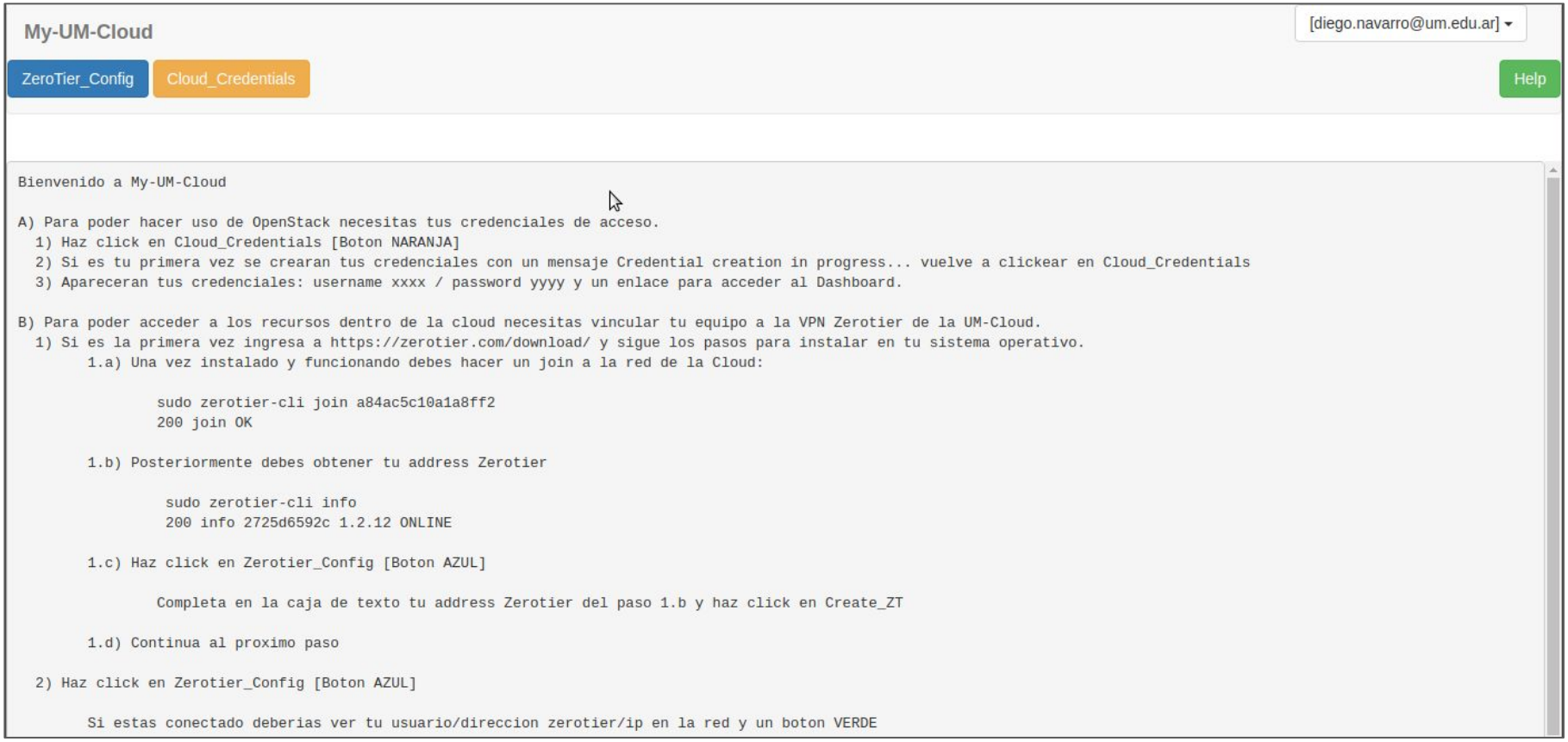
1. https://cloud.um.edu.ar



2. Oauth [“apellido um.edu.ar”]
   a. @um.edu.ar
   b. @alumno.um.edu.ar

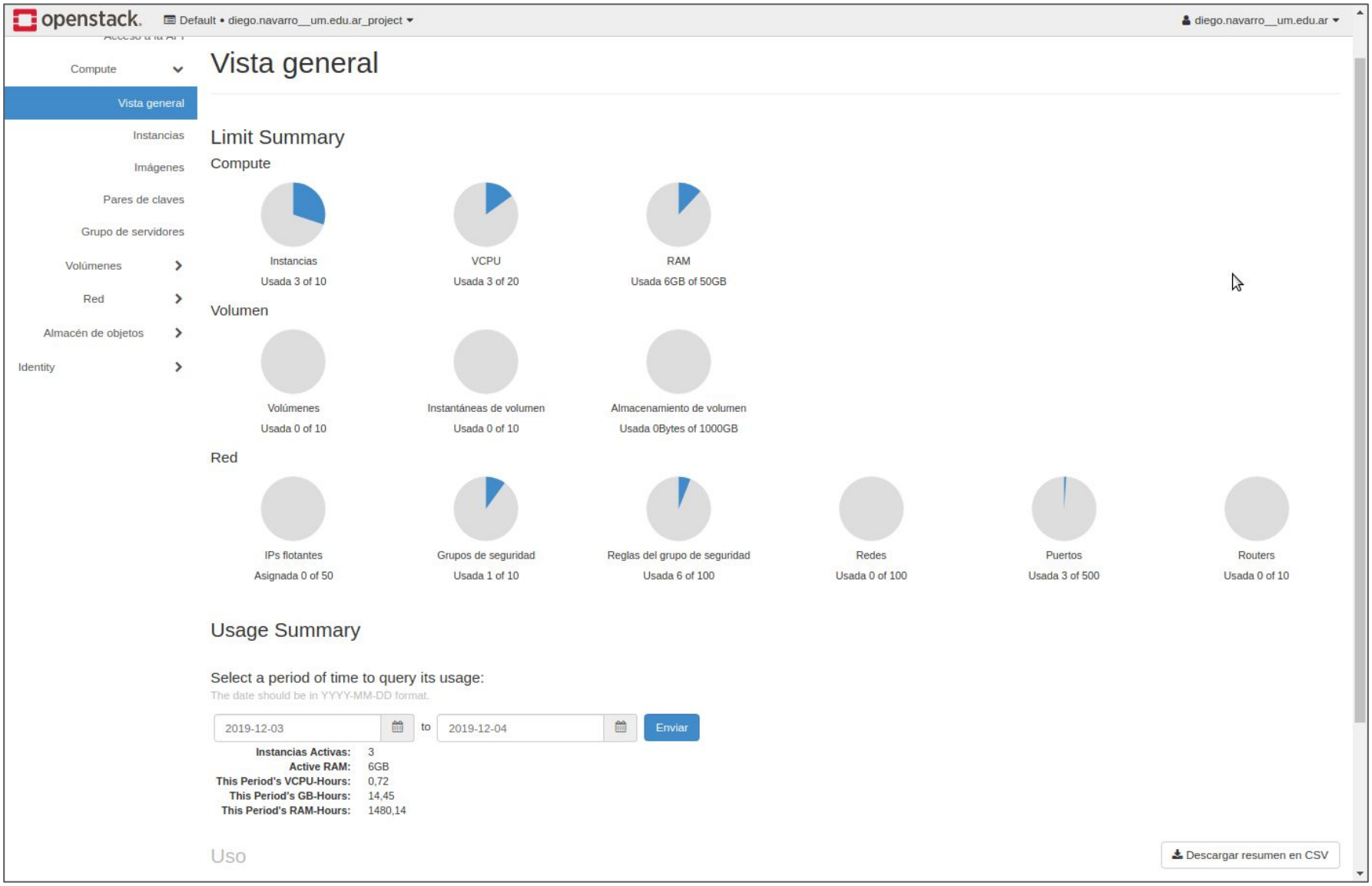
UMCloud: Onboarding

3. https://my.cloud.um.edu.ar



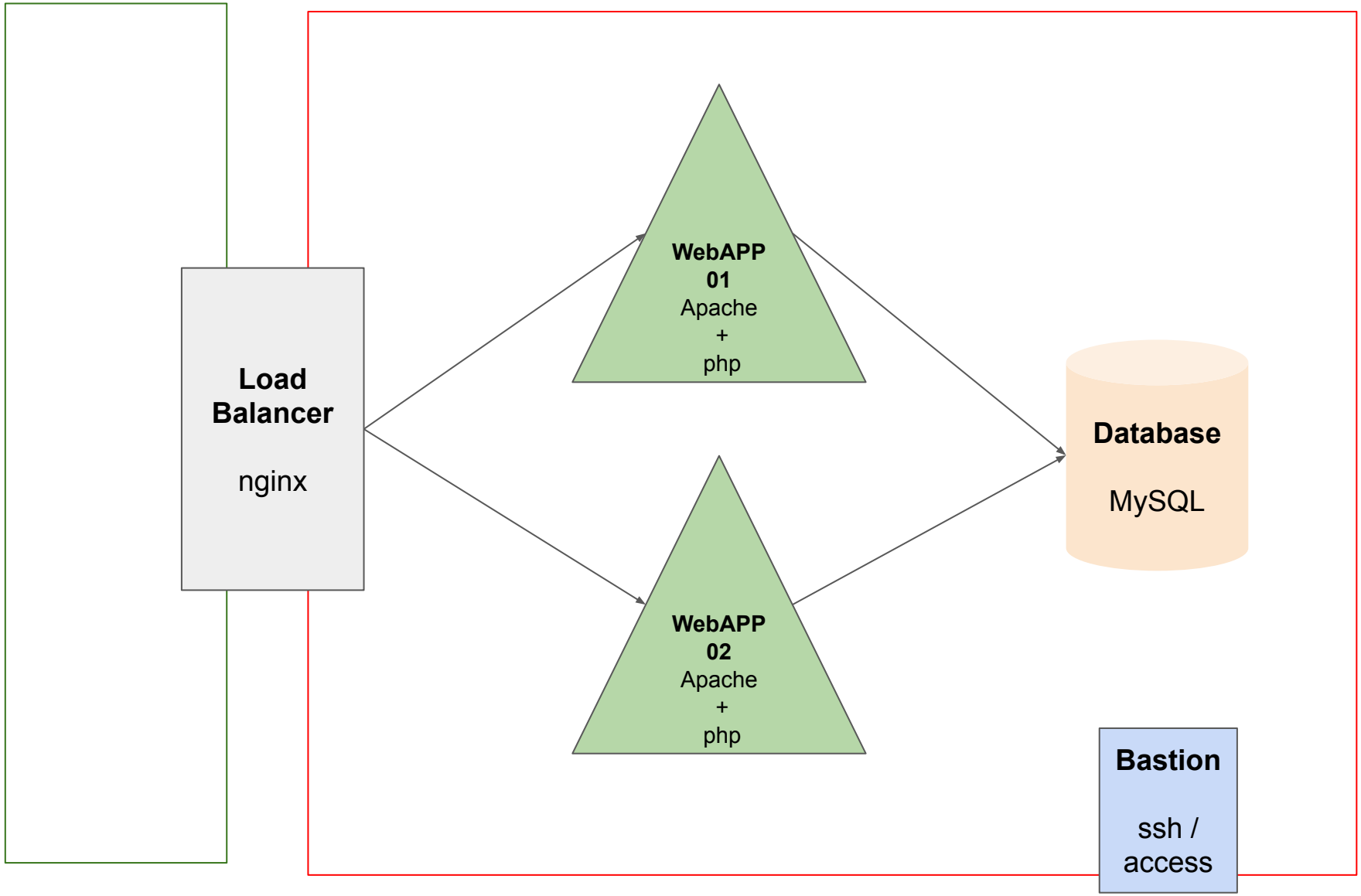
UMCloud: Onboarding

4. Horizon Dashboard https://console.cloud.um.edu.ar



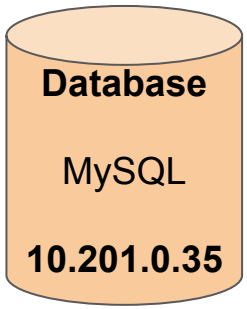
Cloud Classic
Caso Práctico

Aplicación Web con múltiples capas: Wordpress



Aplicación Web con múltiples capas: Wordpress

Necesitamos instalar un motor de base de datos, exponerlo a que sea accesible por la red, crear la base de datos y un usuario con permisos para esa base de datos.

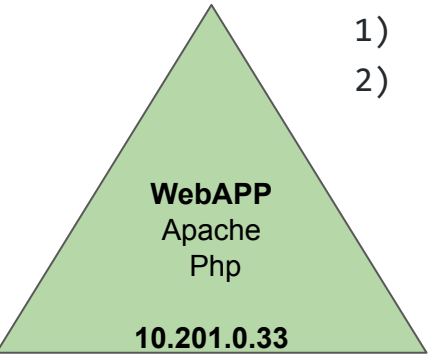


1) Creamos instancia **ubuntu_minimal_1804**, llamarla: USUARIO-**wp-db**
   [zona de disponibilidad : workers-az]
2) Nos logueamos: **ssh ubuntu@<IP_instancia>**

```
sudo mount -o remount,nobarrier,commit=120 /
sudo apt update && sudo apt-get install -y mysql-server vim-tiny

#Poner Base de datos disponible en la RED
sudo vi /etc/mysql/mysql.conf.d/mysqld.cnf
     #bind-address            = 127.0.0.1
     bind-address            = 0.0.0.0
sudo systemctl restart mysql
#creamos DB y usuario para WP
sudo mysql

 CREATE DATABASE wordpress;
GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP,ALTER
    ON wordpress.*
    TO wordpress@'%'
    IDENTIFIED BY 'telewordpress';

sudo journalctl -u mysql  # ver logs del servicio mysql
```

https://ubuntu.com/tutorials/install-and-configure-wordpress

Aplicación Web con múltiples capas: Wordpress



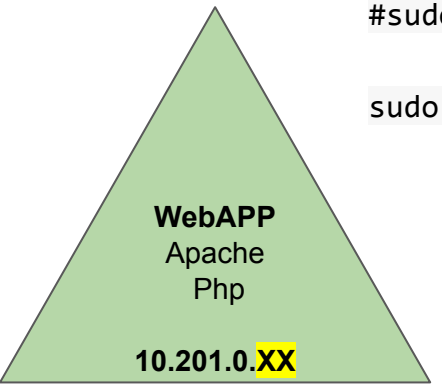
Necesitamos desplegar Wordpress y configurarlo para que acceda a la db remota.

1) Creamos instancia ubuntu_minimal_1804, llamarla: USUARIO-**wp-webapp-01**
2) Nos logueamos: **ssh ubuntu@<IP_instancia>**

```
sudo mount -o remount,nobarrier,commit=120 /
sudo apt-get update && sudo apt-get install -y wordpress php libapache2-mod-php php-mysql vim-tiny

sudo vi /etc/apache2/sites-available/wordpress.conf
     Alias /blog /usr/share/wordpress
     <Directory /usr/share/wordpress>
         Options FollowSymLinks
         AllowOverride Limit Options FileInfo
         DirectoryIndex index.php
         Order allow,deny
         Allow from all
     </Directory>
     <Directory /usr/share/wordpress/wp-content>
         Options FollowSymLinks
         Order allow,deny
         Allow from all
     </Directory>

#Activar Sitio wordpress y modulo de rewrite
sudo a2ensite wordpress && sudo a2enmod rewrite
#Reiniciar apache
sudo service apache2 restart
```

https://ubuntu.com/tutorials/install-and-configure-wordpress

Aplicación Web con múltiples capas: Wordpress




```
#sudo vi /etc/wordpress/config-<IP_INSTANCIA_WEBAPP>.php

sudo vi /etc/wordpress/config-10.201.0.XX.php
      <?php
      define('DB_NAME', 'wordpress');
      define('DB_USER', 'wordpress');
      define('DB_PASSWORD', 'telewordpress'); #password DB
      define('DB_HOST', '10.201.Y.Z'); #Reemplazar por la IP_INSTANCIA_DB
      define('DB_COLLATE', 'utf8_general_ci');
      define('WP_CONTENT_DIR', '/usr/share/wordpress/wp-content');

      ?>


#Apuntar navegador a http://10.201.0.XX/blog/
```

NOTA: necesario agregar al grupo de seguridad (SecurityGroup) una regla permitiendo **HTTP entrante** desde **0.0.0.0/0**

```
Para caso IP_FLOTANTE *en un futuro cercano...*

define( 'WP_HOME', 'http://<IP_FLOTANTE>/blog' );
define( 'WP_SITEURL', 'http://<IP_FLOTANTE>/blog' );
```

https://ubuntu.com/tutorials/install-and-configure-wordpress

Aplicación Web con múltiples capas: Wordpress

Load Balancer

nginx

1) Creamos instancia **ubuntu_minimal_1804 ,** llamarla: **wp-fe-01**
2) Nos logueamos: **ssh ubuntu@<IP_instancia>**
3) Ejecutamos:

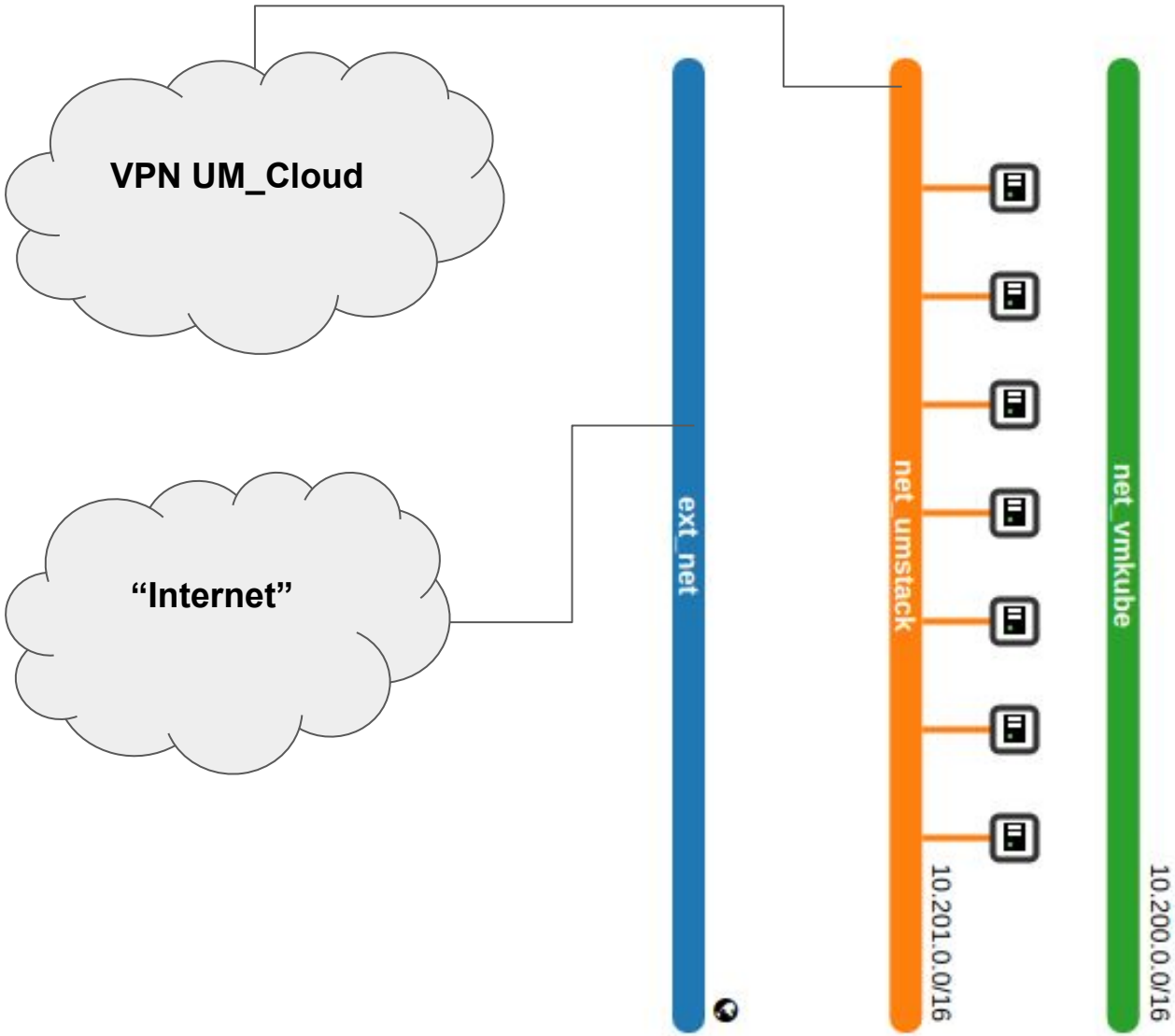
```
# Truquillo de optimización de disco:
sudo mount -o remount,nobarrier,commit=120 /

# Install nginx, and helper tools
sudo apt-get update && sudo apt-get install nginx vim-tiny curl

# Remove default nginx site
sudo rm /etc/nginx/sites-enabled/default
# Configure nginx as reverse proxy
sudo vi /etc/nginx/conf.d/lb.conf
server {
listen 80;
location / {
proxy_pass http://10.201.0.XX; # my backend WP
}
}

sudo service nginx restart
# Should show “backend” (WP) content
curl http://localhost/
# Ver logs de acceso en el frontend proxy
journalctl -u nginx
tail -f /var/log/nginx/access.log
# Chequear que puedo navegar:
http://<IP de wp-fe-01>/
```

Cloud Classic
Tema Floating IP

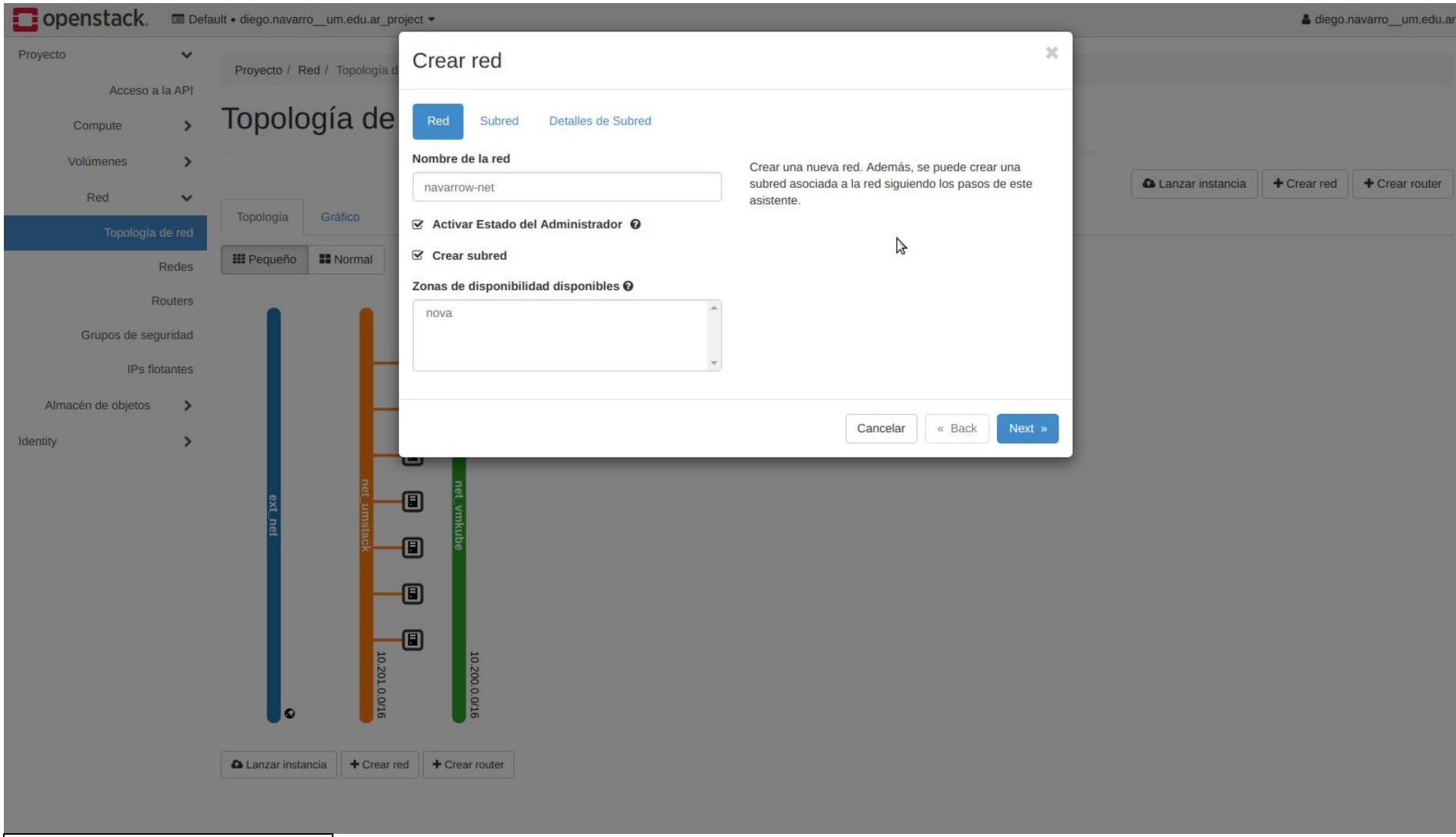
```
# En la máquina de la DB

#Para actualizar la IP flotante en la config de wordpress
echo 'update wp_options set option_value = "http://<ip_flotante>/blog" where
option_name="siteurl"' | sudo mysql wordpress

echo 'update wp_options set option_value = "http://<ip_flotante>/blog" where
option_name="home"' | sudo mysql wordpress
```

Cloud Classic
Caso Práctico ++

Cloud Networking : Crear Red

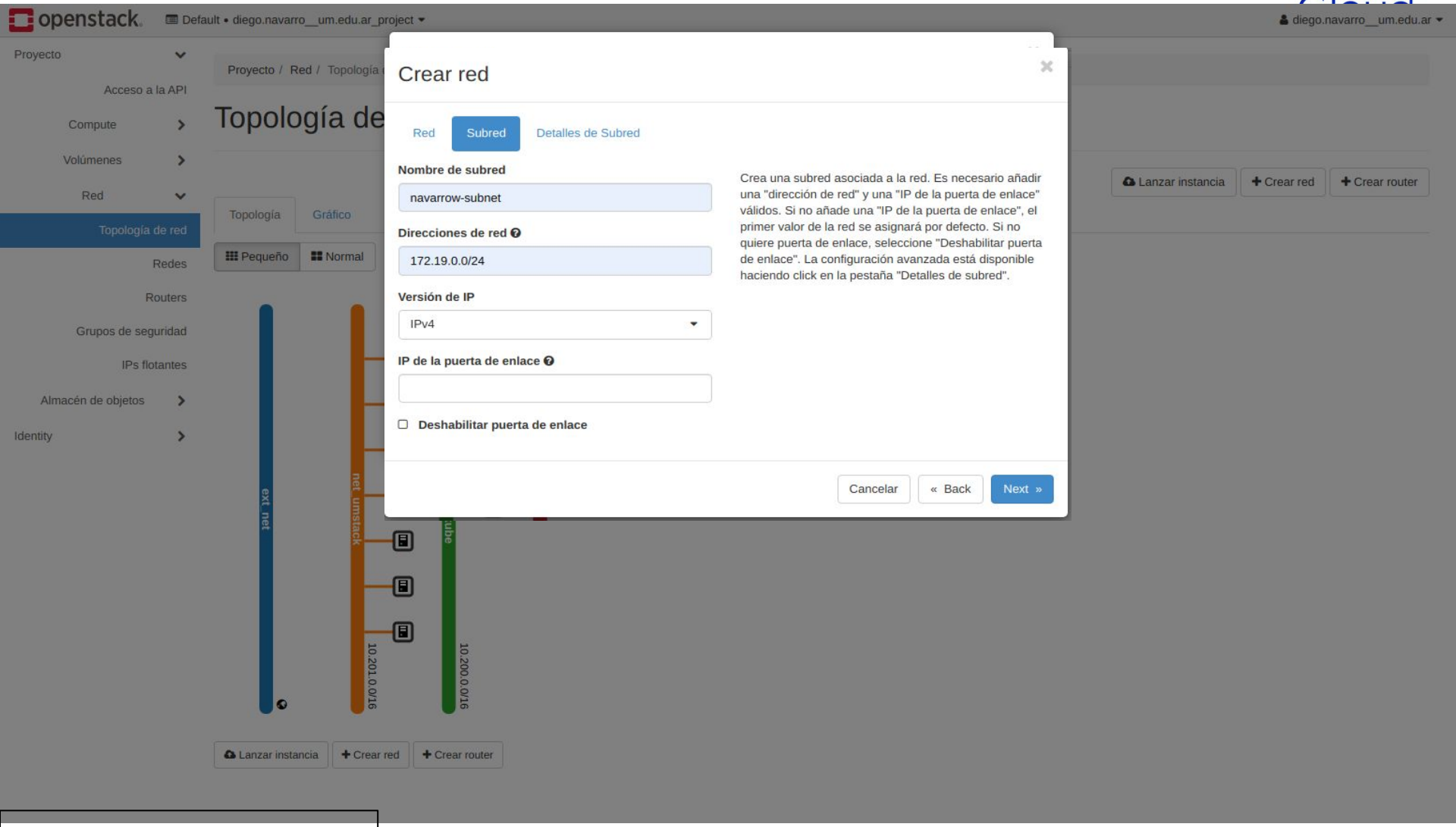


Cloud Networking: Crear Red



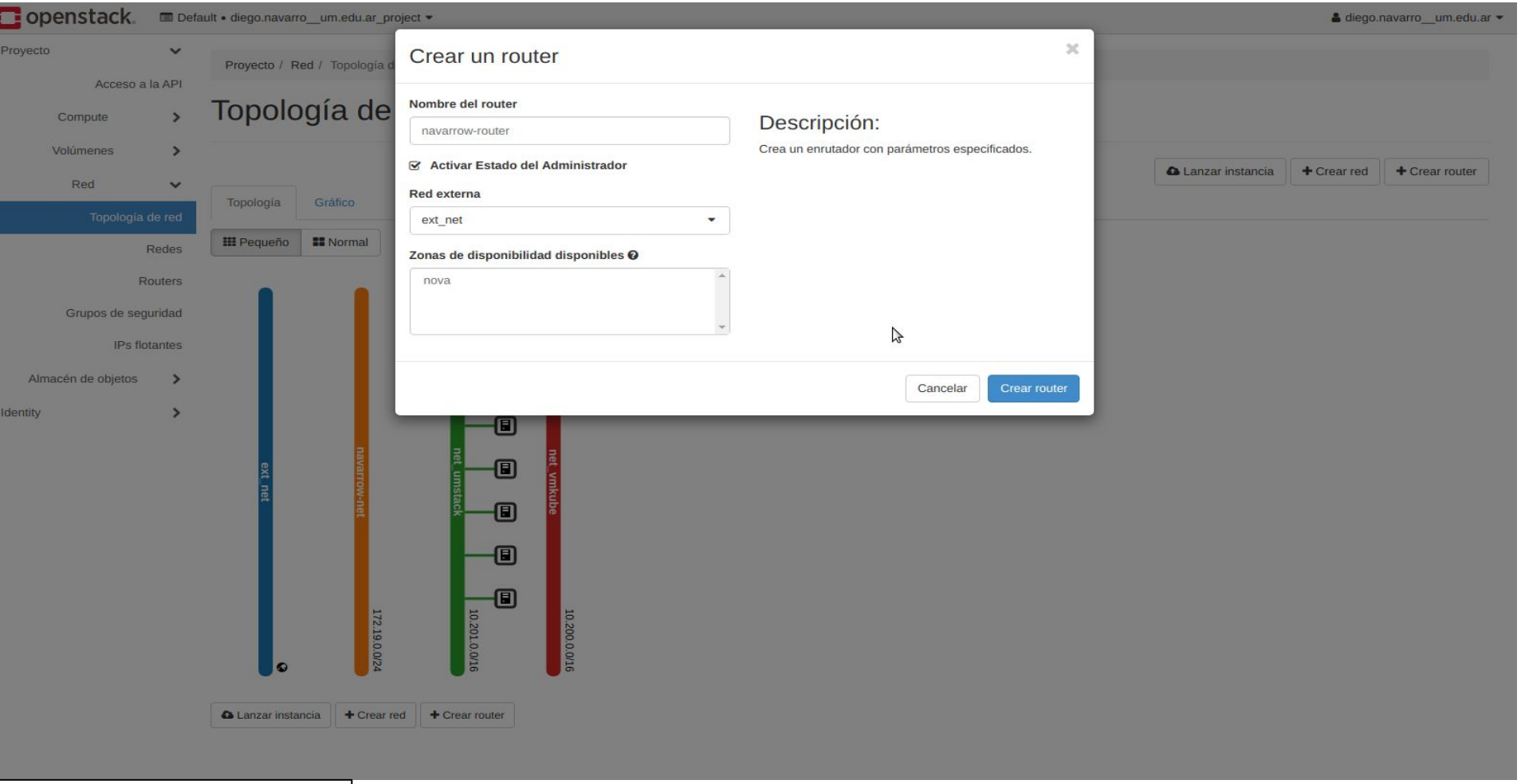
Nombre de la red:
***usuario*-red**

Cloud Networking: Crear Red



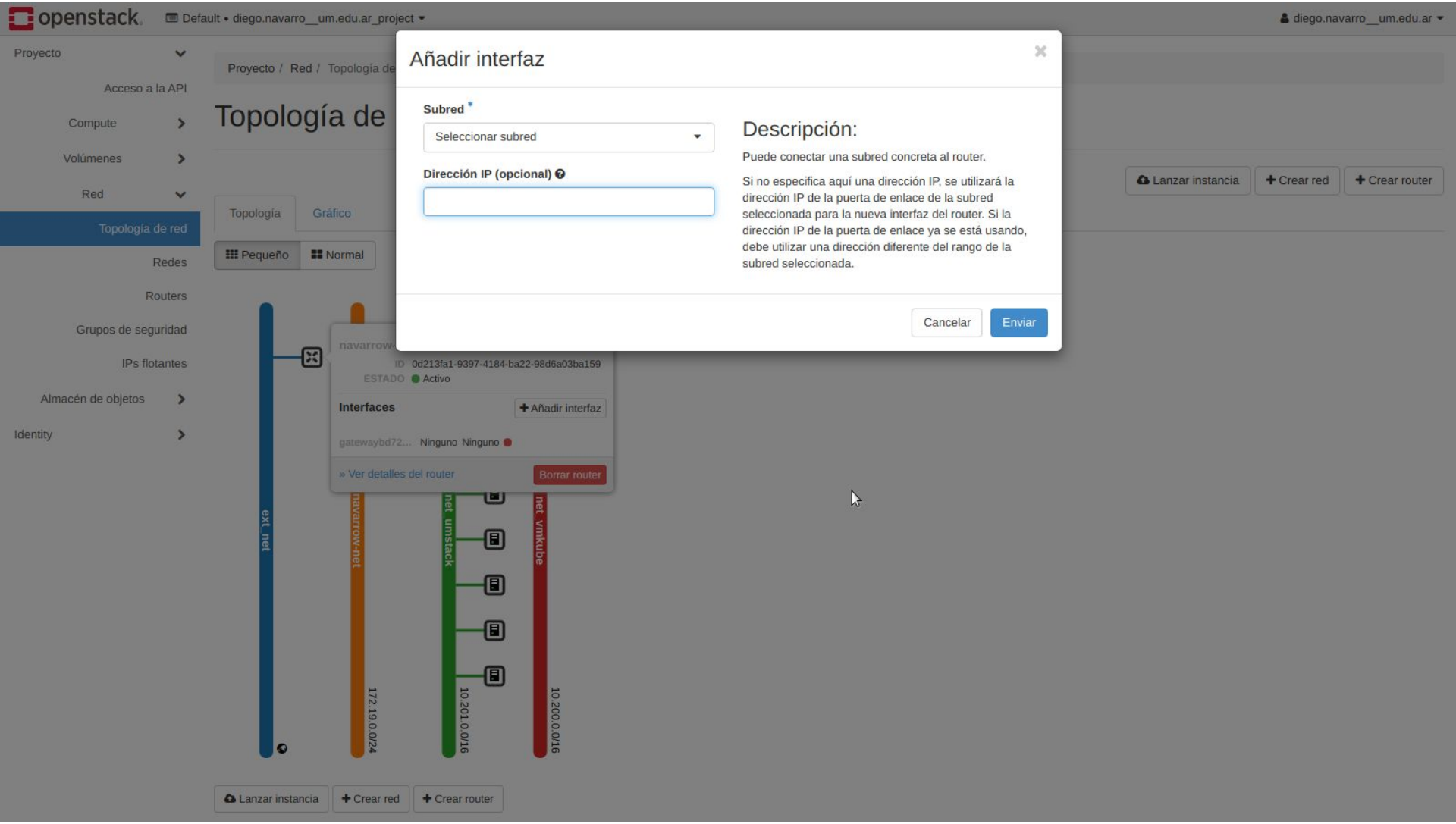
Nombre de subred:
***usuario*-subnet**
Direcciones de red:
172.19.0.0/24

Cloud Networking: Crear Router

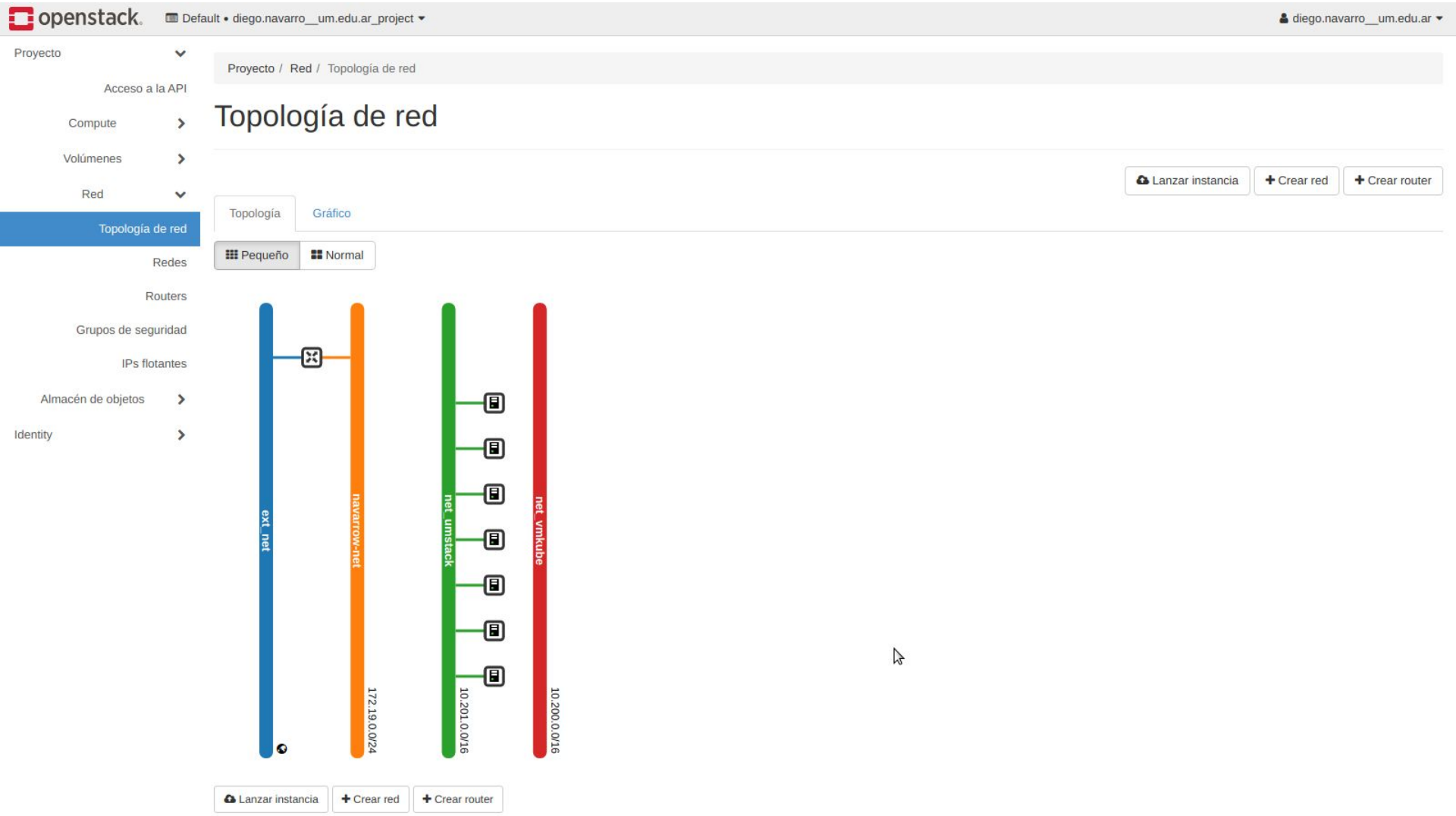


Nombre de router:
***Usuario*-router**
Red externa:
ext_net

Cloud Networking: Crear Router, Añadir Interfaz



Cloud Networking



Instancias

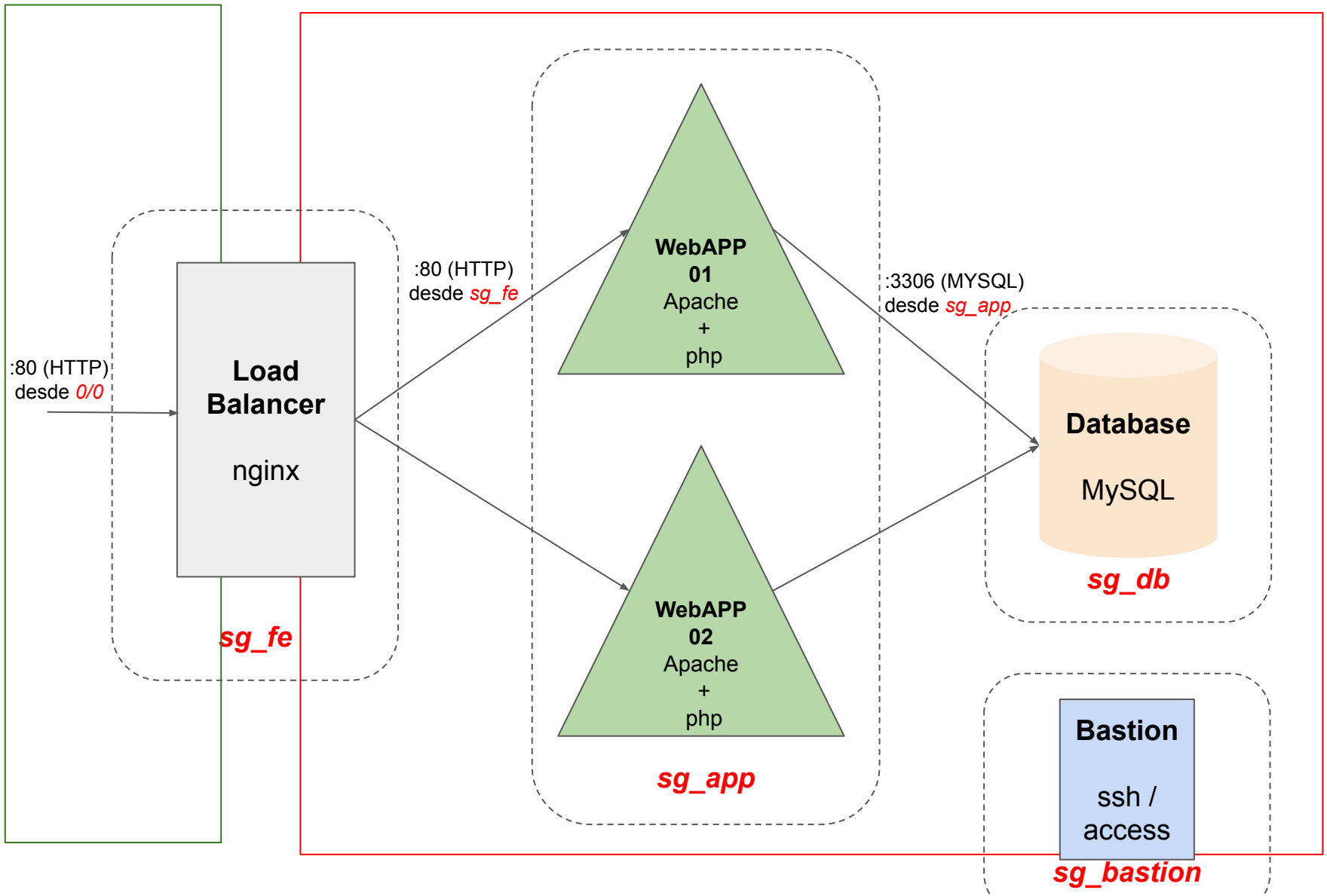
Creamos nuevamente las instancias sobre la nueva red Privada y Asignamos la IP Elástica a una [bastión] [remember -> ssh **-A**]

Nombre de la instancia	Nombre de la imagen	Dirección IP	Sabor	Par de claves	Estado	Zona de Disponibilida
db-priv	ubuntu_minimal_1804	172.19.0.10	m1.c1m1d20	navarrow	Activo	workers-az
wp-priv	ubuntu_minimal_1804	172.19.0.8	m1.c1m1d20	navarrow	Activo	workers-az
lb-priv	ubuntu_minimal_1804	172.19.0.6 IPs flotantes: 192.168.3.85	m1.c1m1d20	navarrow	Activo	workers-az

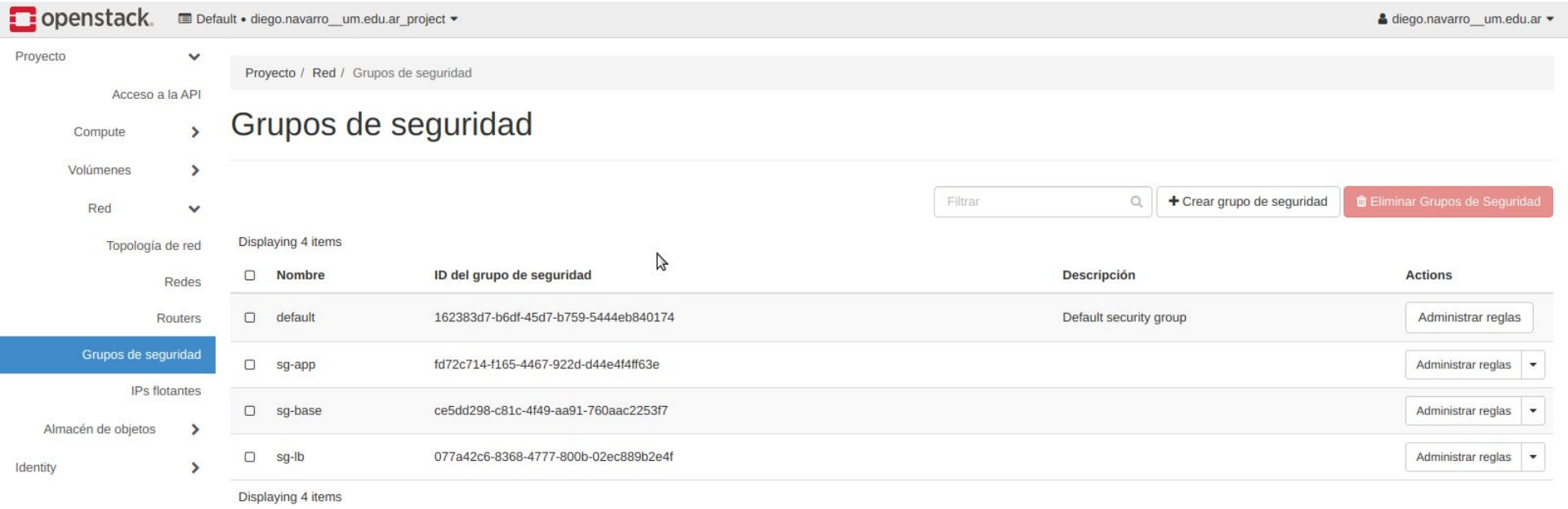
Rehacemos la configuración de la clase anterior con el nuevo

direccionamiento

Aplicación Web con múltiples capas: Wordpress



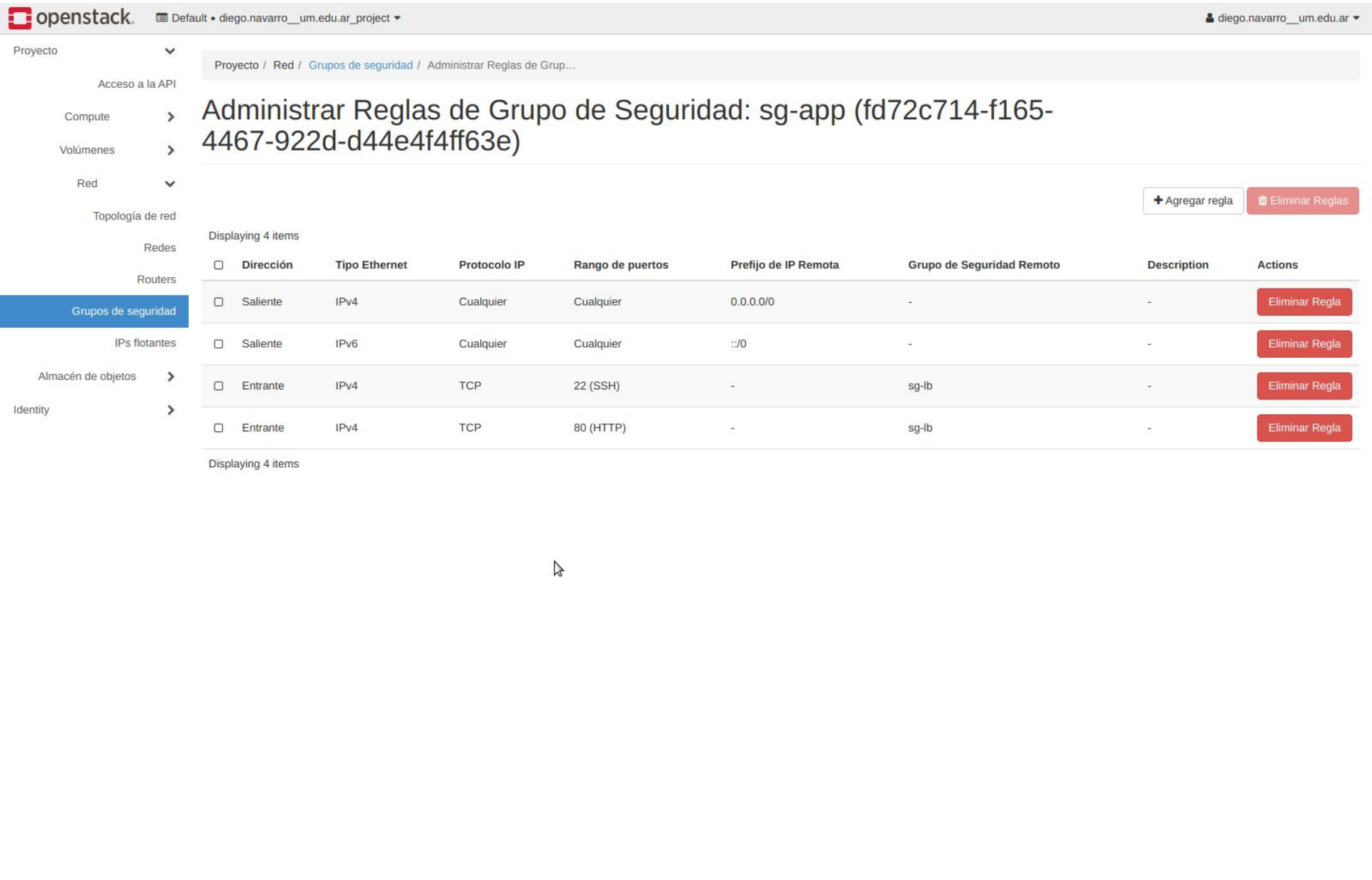
Grupos de Seguridad



Grupos de Seguridad: sg-base [a todos]

Grupos de Seguridad: sg-lb [solo Load Balancer]

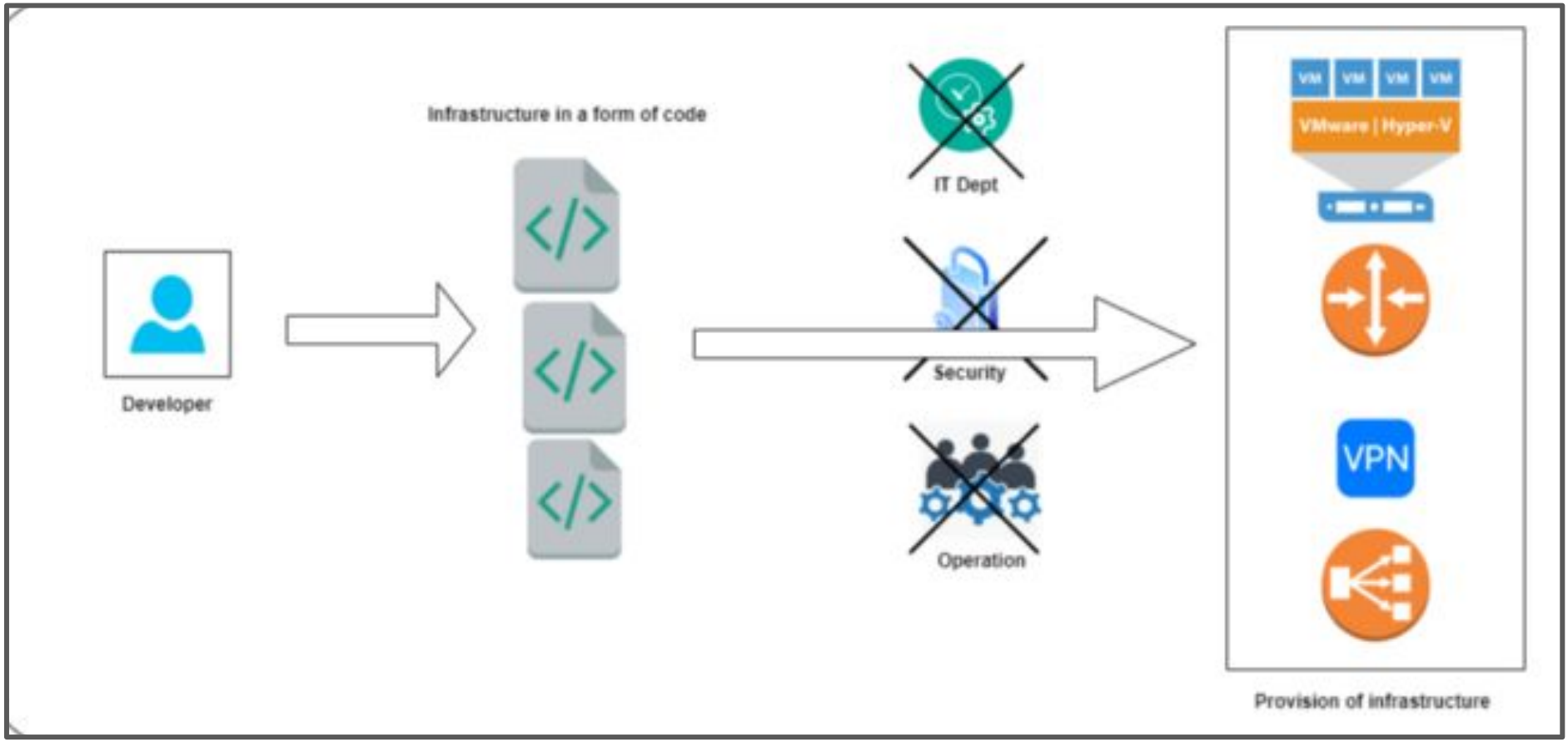
Grupos de Seguridad: sg-app [solo a la WebApp]



Grupos de Seguridad: sg-db [solo a la db]

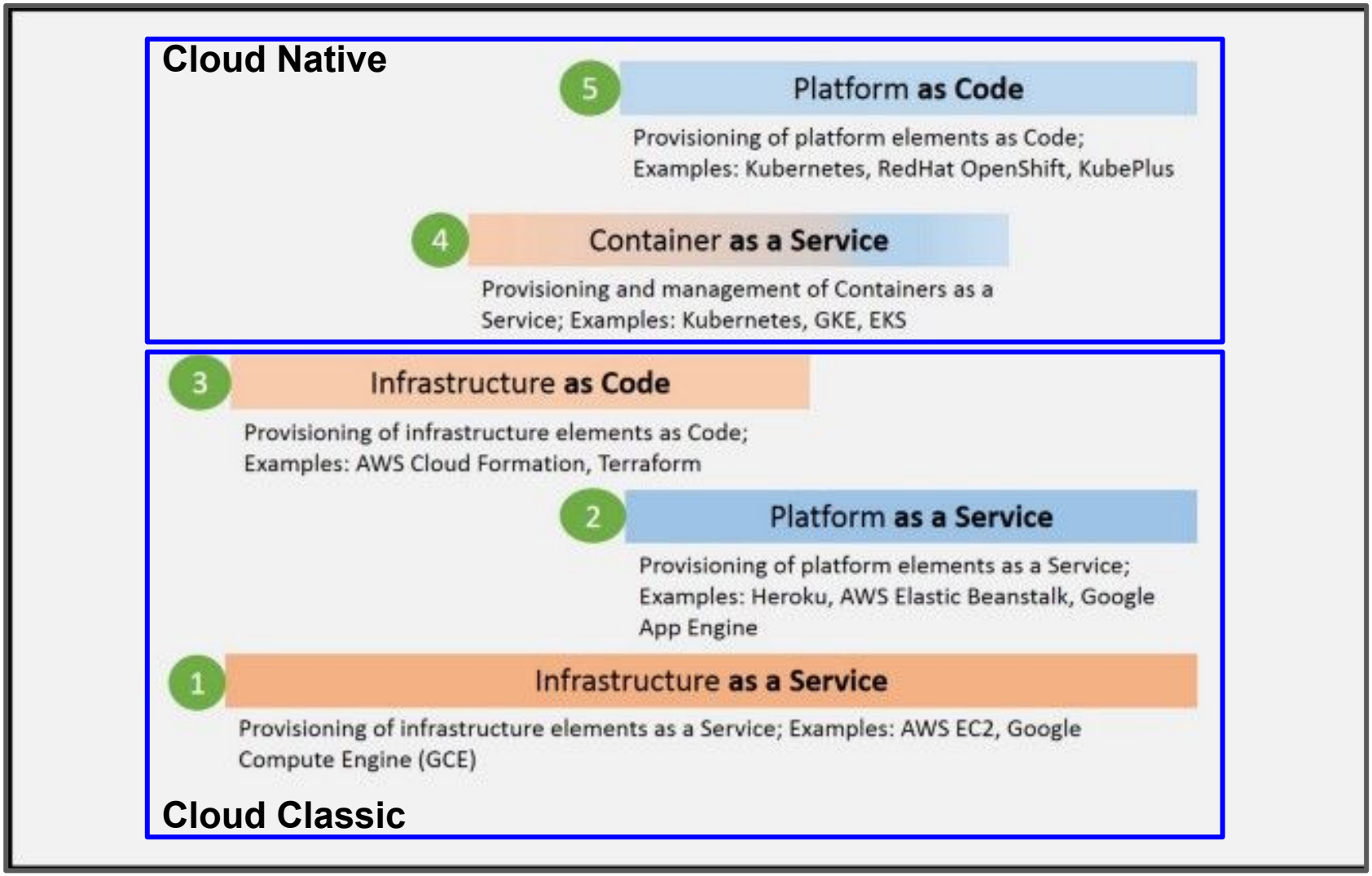
IaC
Infra as Code

Infraestructura como código: IaC



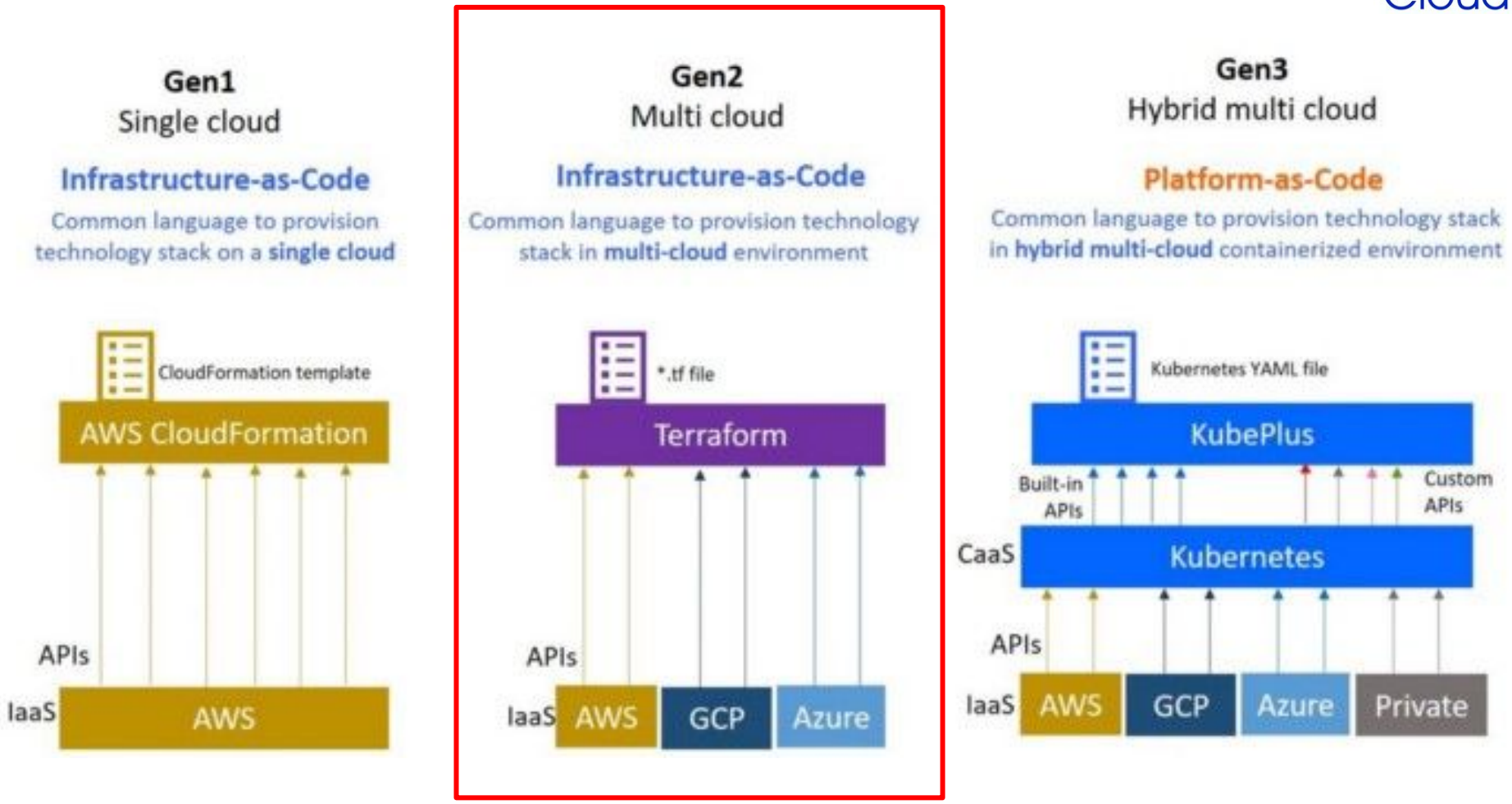
https://medium.com/faun/infrastructure-as-code-917d245ce35e

Plataforma como código



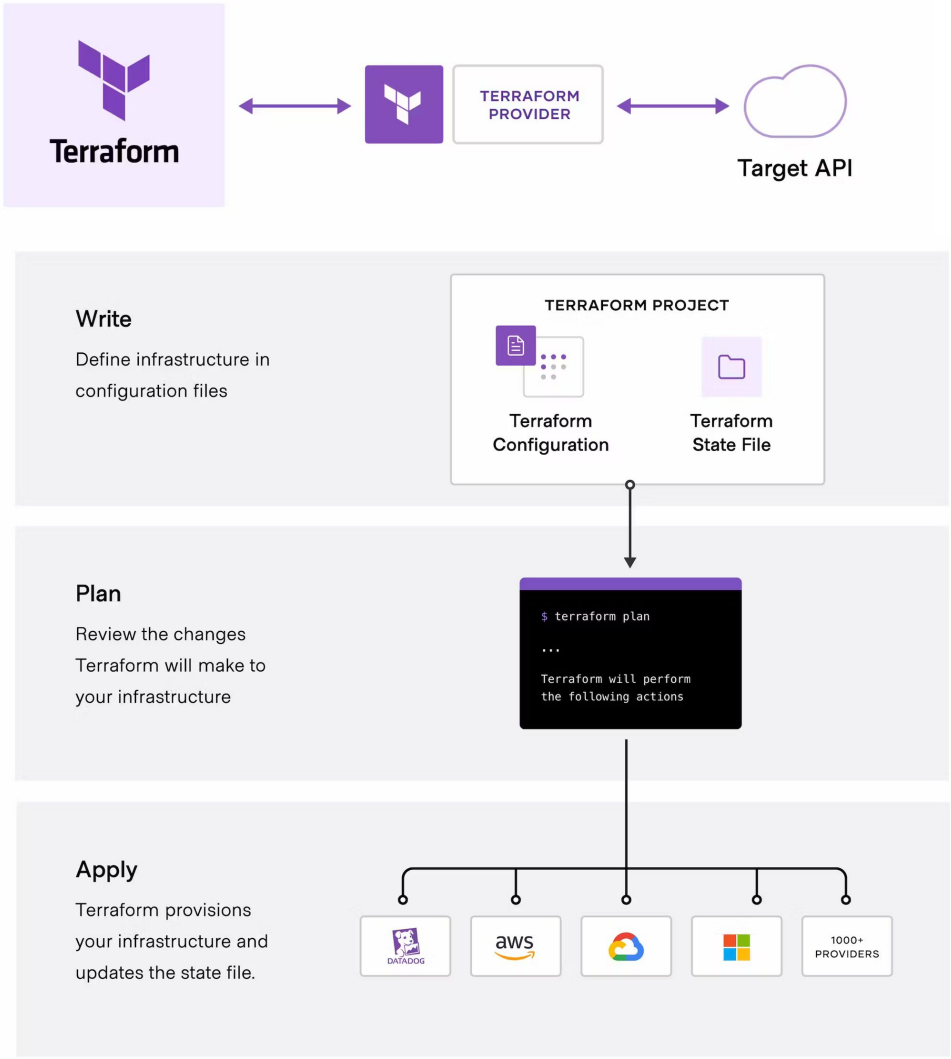
https://itnext.io/evolution-of-paases-to-platform-as-code-in-kubernetes-world-74464b0013ca

Plataforma como código



https://medium.com/@cloudark/kubernetes-and-the-future-of-as-code-systems-b1b2de312742

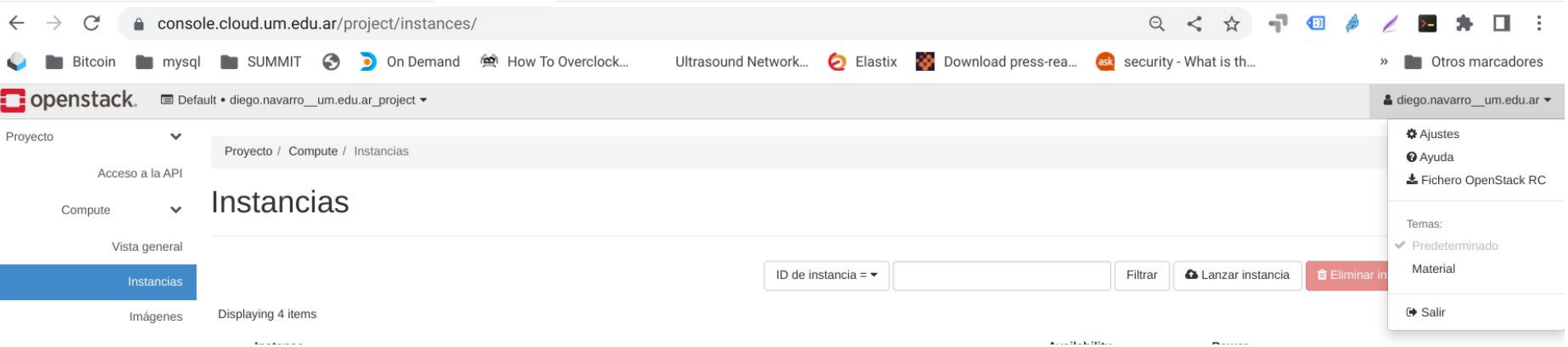
IaC: Terraform



https://developer.hashicorp.com/terraform/intro

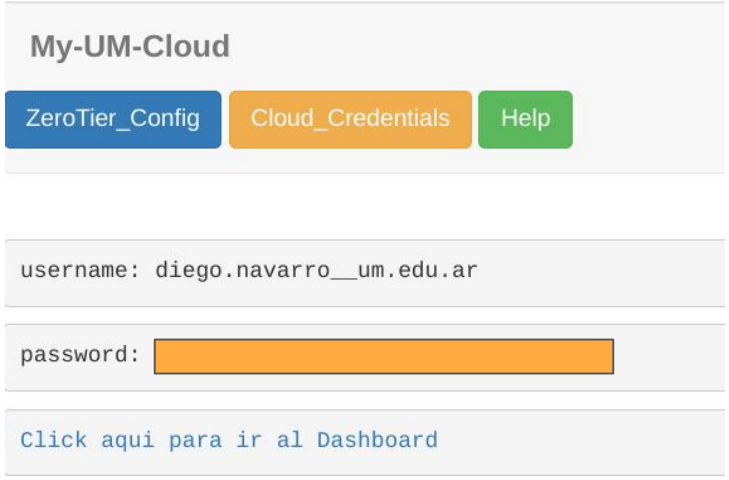
Terraform: Setup

```
## Descargar Terraform
wget https://releases.hashicorp.com/terraform/1.4.6/terraform_1.4.6_linux_amd64.zip
## Descomprimimos
unzip terraform_1.4.6_linux_amd64.zip
#probamos
./terraform
#Descargamos fichero openrc (Credenciales API Openstack)
## ej: diego.navarro__um.edu.ar_project-openrc
```



Terraform: Setup

```
## Cargamos en el ambiente las credenciales
source diego.navarro__um.edu.ar_project-openrc.sh
## Por seguridad nos pide que coloquemos nuestro password de Horizon
```




```
##Seteamos el provider Openstack
vi versions.tf
```

```
terraform {
 required_providers {
   openstack = {
     source = "terraform-provider-openstack/openstack"
   }
 }
 required_version = ">= 0.13"
}
```

```
##Inicializamos Terraform
./terraform init
```

```
Initializing the backend...

Initializing provider plugins...
- Finding latest version of terraform-provider-openstack/openstack...
- Installing terraform-provider-openstack/openstack v1.51.1...
- Installed terraform-provider-openstack/openstack v1.51.1 (self-signed, key ID 4F80527A391BEFD2)

Partner and community providers are signed by their developers.
If you'd like to know more about provider signing, you can read about it here:
https://www.terraform.io/docs/cli/plugins/signing.html

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

Terraform: Nuestra Primer VM - IaC

vi vm.tf

```
data "openstack_images_image_v2" "ubuntu_2204" {
  name        = "ubuntu_2204"
  most_recent = true
}
data "openstack_compute_flavor_v2" "small" {
  vcpus = 1
  ram   = 2048
}
data "openstack_networking_network_v2" "net_umstack" {
  name = "net_umstack"
}
resource "openstack_networking_port_v2" "vm_port_umcloud" {
  name           = "vm_port_umcloud"
  network_id     = data.openstack_networking_network_v2.net_umstack.id
  admin_state_up = "true"
}
resource "openstack_compute_instance_v2" "terraform_vm" {
  lifecycle {
    ignore_changes = [
      image_id,
    ]
  }
  name              = "terraform_vm"
  image_id          = data.openstack_images_image_v2.ubuntu_2204.id
  flavor_id         = data.openstack_compute_flavor_v2.small.id
  key_pair          = "<REEMPLAZAR-POR-MIS-KEYS>"
  security_groups   = ["default"]
  availability_zone = "nodos-amd-2022"
  network {
    port = openstack_networking_port_v2.vm_port_umcloud.id
  }
  user_data = "${file("init.sh")}"
}
```

vi init.sh

```
#!/bin/bash
apt-get update -y
apt-get install -y nginx vim-tiny curl
```

Terraform: Nuestra Primer VM - IaC

##Aplicamos Terraform

```
./terraform apply
```

```
data.openstack_networking_network_v2.net_umstack: Reading...
data.openstack_compute_flavor_v2.small: Reading...
data.openstack_images_image_v2.ubuntu_2204: Reading...
data.openstack_images_image_v2.ubuntu_2204: Read complete after 0s [id=ece749b8-1736-40e7-ade5-e41114111b6a]
data.openstack_compute_flavor_v2.small: Read complete after 0s [id=2d357d3d-32c1-4af8-81dd-a71a7d7cf303]
data.openstack_networking_network_v2.net_umstack: Read complete after 0s [id=6f728afe-d289-4ccc-b108-87c3fbcfc30c]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # openstack_compute_instance_v2.terraform_vm will be created
  + resource "openstack_compute_instance_v2" "terraform_vm" {
      + access_ip_v4        = (known after apply)
      + access_ip_v6        = (known after apply)
      + all_metadata        = (known after apply)
      + all_tags            = (known after apply)
      + availability_zone   = "nodos-amd-2022"
      + created             = (known after apply)
      + flavor_id           = "2d357d3d-32c1-4af8-81dd-a71a7d7cf303"
      + flavor_name         = (known after apply)
      + force_delete        = false
      + id                  = (known after apply)
      + image_id            = "ece749b8-1736-40e7-ade5-e41114111b6a"
      + image_name          = (known after apply)
      + key_pair            = "navarrow"
      + name                = "terraform_vm"
      + power_state         = "active"
      + region              = (known after apply)
      + security_groups     = [
          + "default",
        ]
      + stop_before_destroy = false
      + updated             = (known after apply)
      + user_data           = "cd983027ddbaec321bf4ea1217a4812878dd2398"

      + network {
          + access_network = false
          + fixed_ip_v4    = (known after apply)
          + fixed_ip_v6    = (known after apply)
          + floating_ip    = (known after apply)
          + mac            = (known after apply)
          + name           = (known after apply)
          + port           = (known after apply)
          + uuid           = (known after apply)
        }
    }

  # openstack_networking_port_v2.vm_port_umcloud will be created
  + resource "openstack_networking_port_v2" "vm_port_umcloud" {
      + admin_state_up         = true
      + all_fixed_ips          = (known after apply)
      + all_security_group_ids = (known after apply)
      + all_tags               = (known after apply)
      + device_id              = (known after apply)
      + device_owner           = (known after apply)
      + dns_assignment         = (known after apply)
      + dns_name               = (known after apply)
      + id                     = (known after apply)
      + mac_address            = (known after apply)
      + name                   = "vm_port_umcloud"
      + network_id             = "6f728afe-d289-4ccc-b108-87c3fbcfc30c"
      + port_security_enabled  = (known after apply)
      + qos_policy_id          = (known after apply)
      + region                 = (known after apply)
      + tenant_id              = (known after apply)
    }

Plan: 2 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value:
```

Terraform: Nuestra Primer VM - IaC

##Verificamos Terraform

```
./terraform plan
```

```
navarrow@penguin:~/Proyectos/umcloud/terraform$ ./terraform plan
data.openstack_images_image_v2.ubuntu_2204: Reading...
data.openstack_networking_network_v2.net_umstack: Reading...
data.openstack_compute_flavor_v2.small: Reading...
data.openstack_images_image_v2.ubuntu_2204: Read complete after 1s [id=ece749b8-1736-40e7-ade5-e41114111b6a]
data.openstack_compute_flavor_v2.small: Read complete after 1s [id=2d357d3d-32c1-4af8-81dd-a71a7d7cf303]
data.openstack_networking_network_v2.net_umstack: Read complete after 1s [id=6f728afe-d289-4ccc-b108-87c3fbcfc30c]
openstack_networking_port_v2.vm_port_umcloud: Refreshing state... [id=49a11c81-751c-40fb-b6d7-c80b58b3dd7a]
openstack_compute_instance_v2.terraform_vm: Refreshing state... [id=1cf8281a-2131-4b4e-a618-78199687db56]

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed.
```

##Vemos en el Dashboard de Horizon que tenemos una nueva VM !!!

IaC Integrando Conceptos: Caso +++

Desarrollaremos todo el caso ahora de manera "programática"

Para ello:

- **Verificar instalación Terraform accesible (Clase anterior)**
  - wget https://releases.hashicorp.com/terraform/1.4.6/terraform_1.4.6_linux_amd64.zip
  - **## Descomprimimos**
  - unzip terraform_1.4.6_linux_amd64.zip
  - **#probamos**
  - ./terraform
  - **#Lo ponemos invocable desde cualquier lado.**
  - sudo cp ./terraform /usr/local/bin
- Cargamos en el ambiente las credenciales
- source diego.navarro__um.edu.ar_project-openrc.sh
- **Eliminar todo lo que tengamos creado:**
  - **VMs, Routers, Nets, SGs, Floating IP**
- **Clonar el repo del curso**

  git clone **https://github.com/umcloud/clases-devops**
- **Ingresar a la clase 02**

  cd clases-devops/c02-terraform_fe_app_db/
- **Iremos paso a paso**

Repo GitHub

https://github.com/umcloud

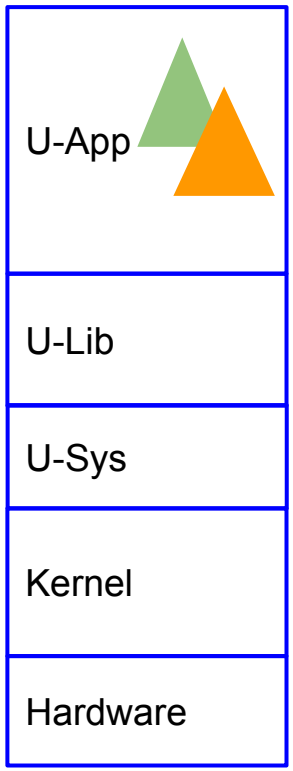
Mirar los Makefile 😉 … mucha 🪄

Cloud native

Cloud Native - containers son *cool*, pero ...

Cloud Native - “un poco de arqueología ...”

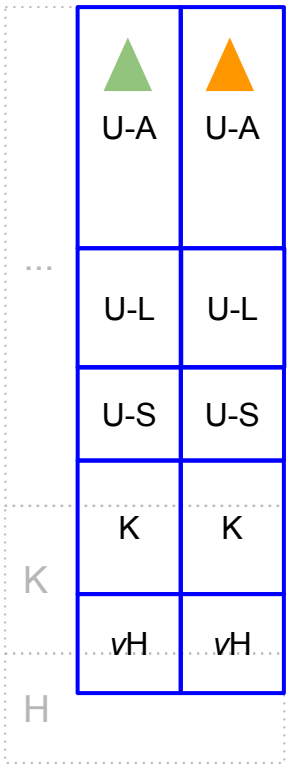
El Monolito



Cloud Native - “un poco de arqueología ...”

El Monolito

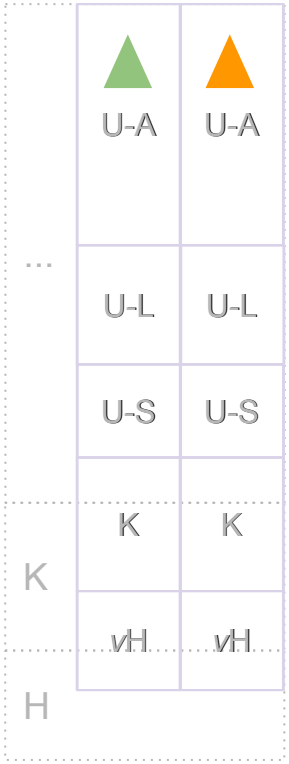
VM



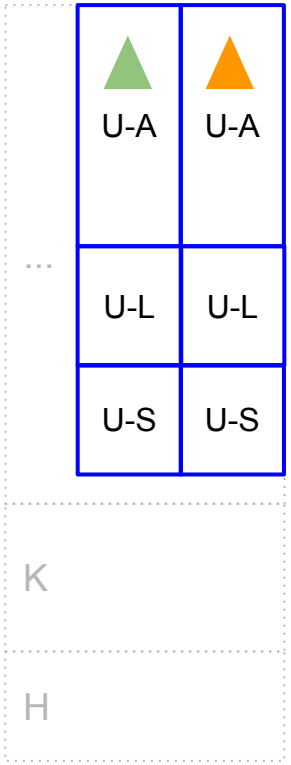
Cloud Native - “un poco de arqueología ...”

El Monolito

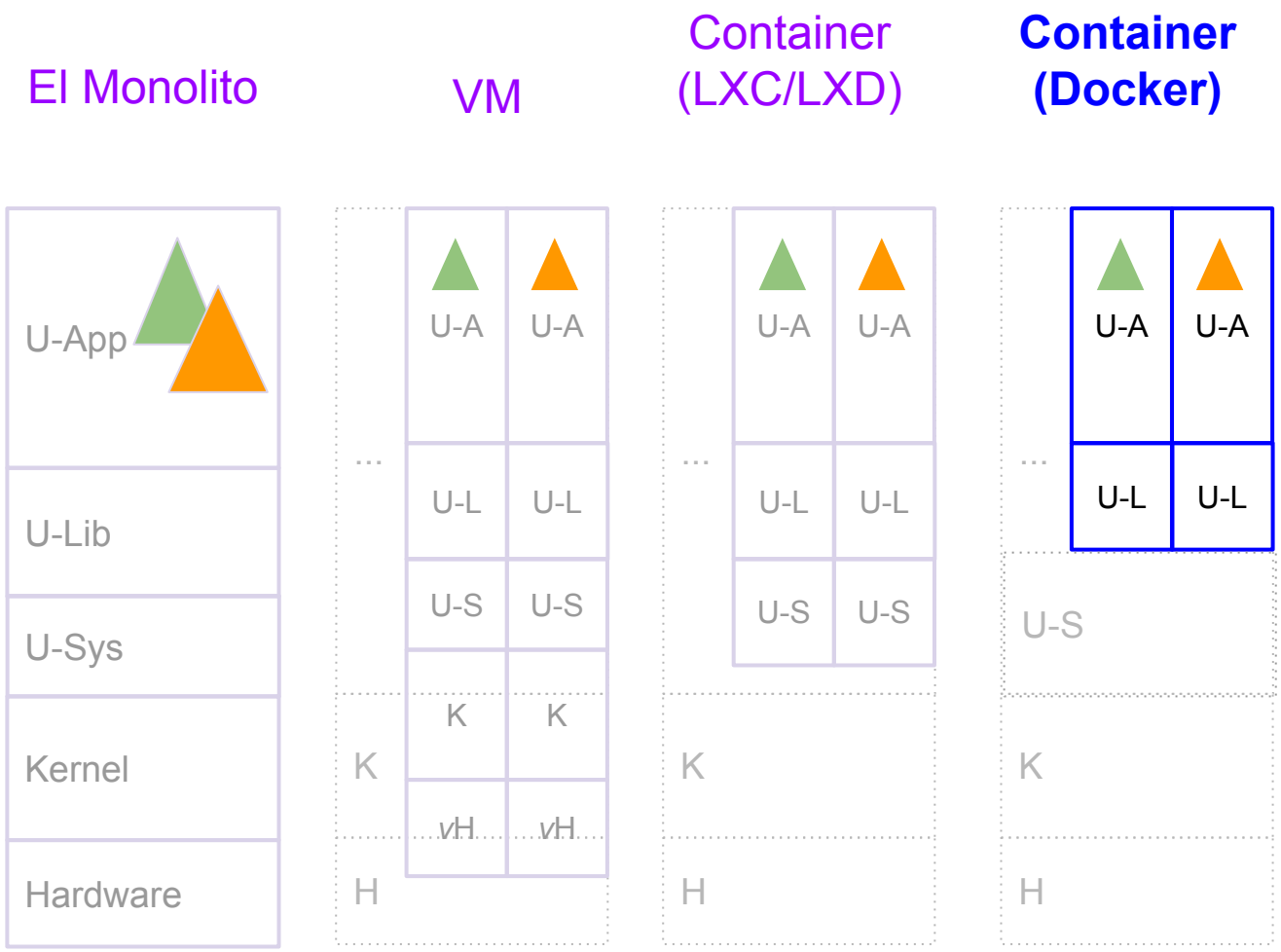
VM



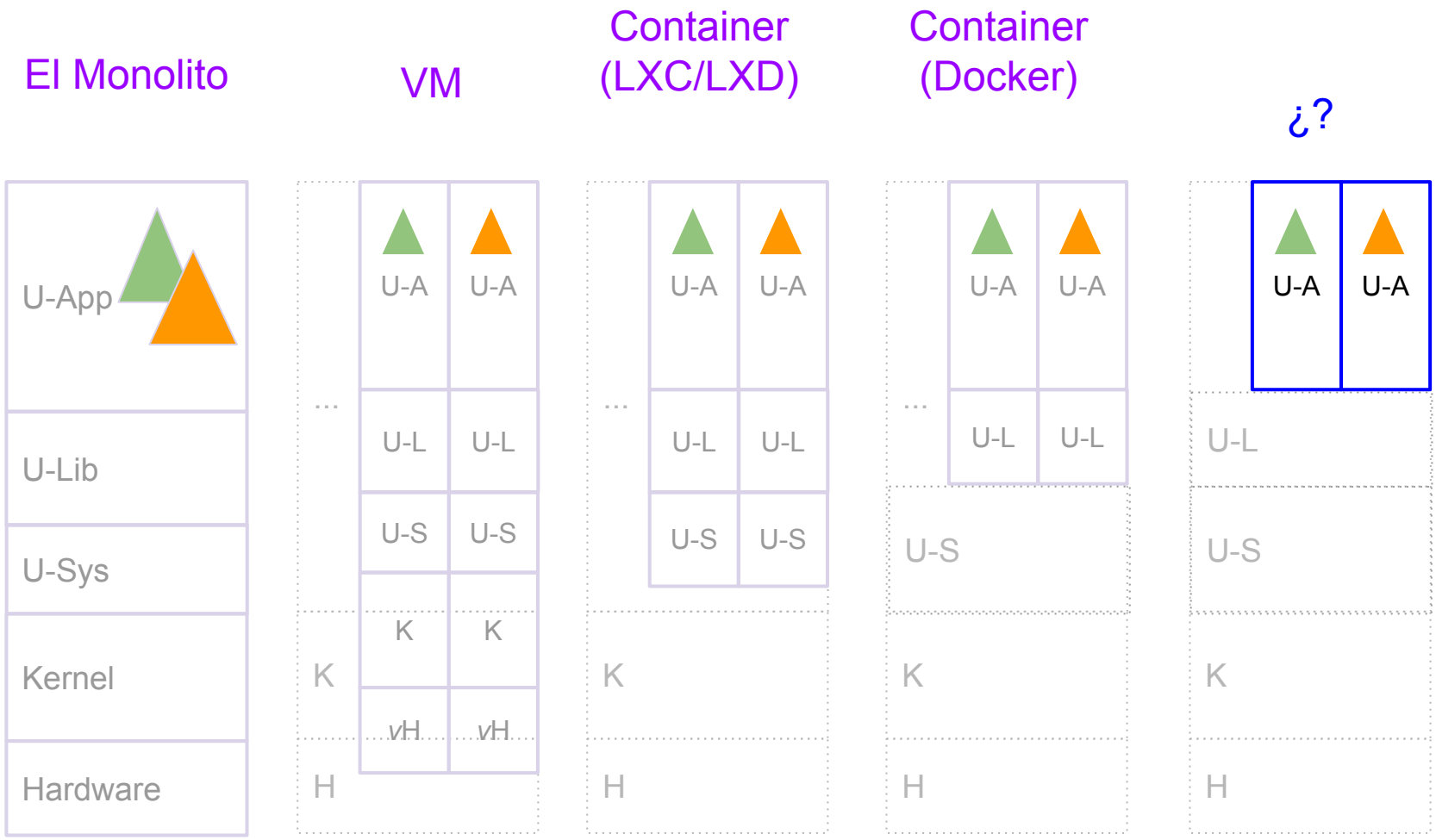
Container (LXC/LXD)



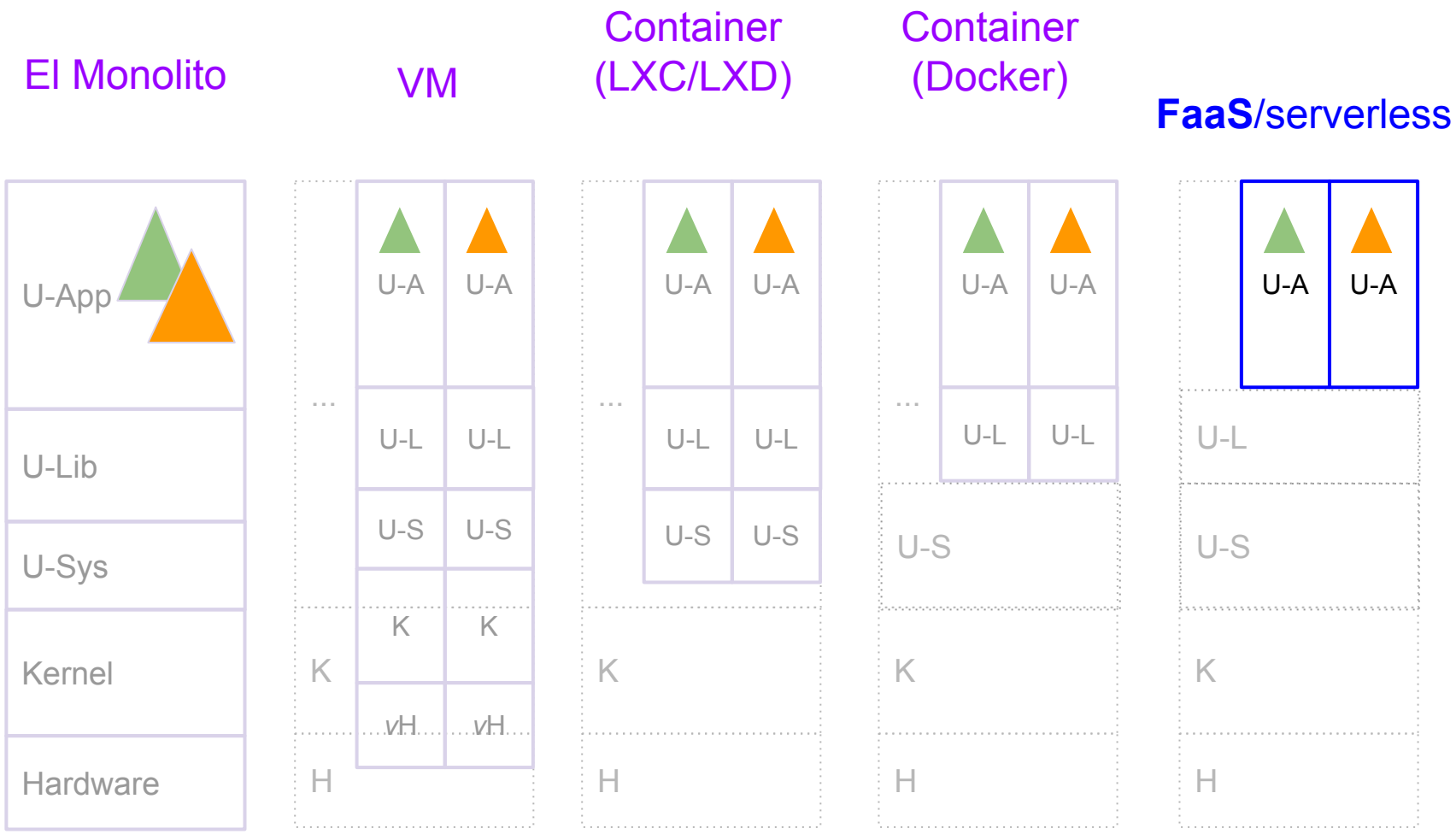
Cloud Native - “un poco de arqueología ...”



Cloud Native - “un poco de arqueología ...”

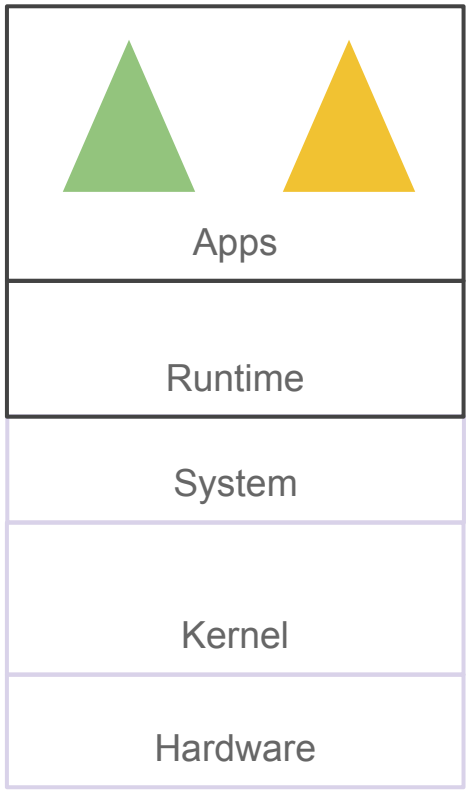


Cloud Native - “un poco de arqueología ...”



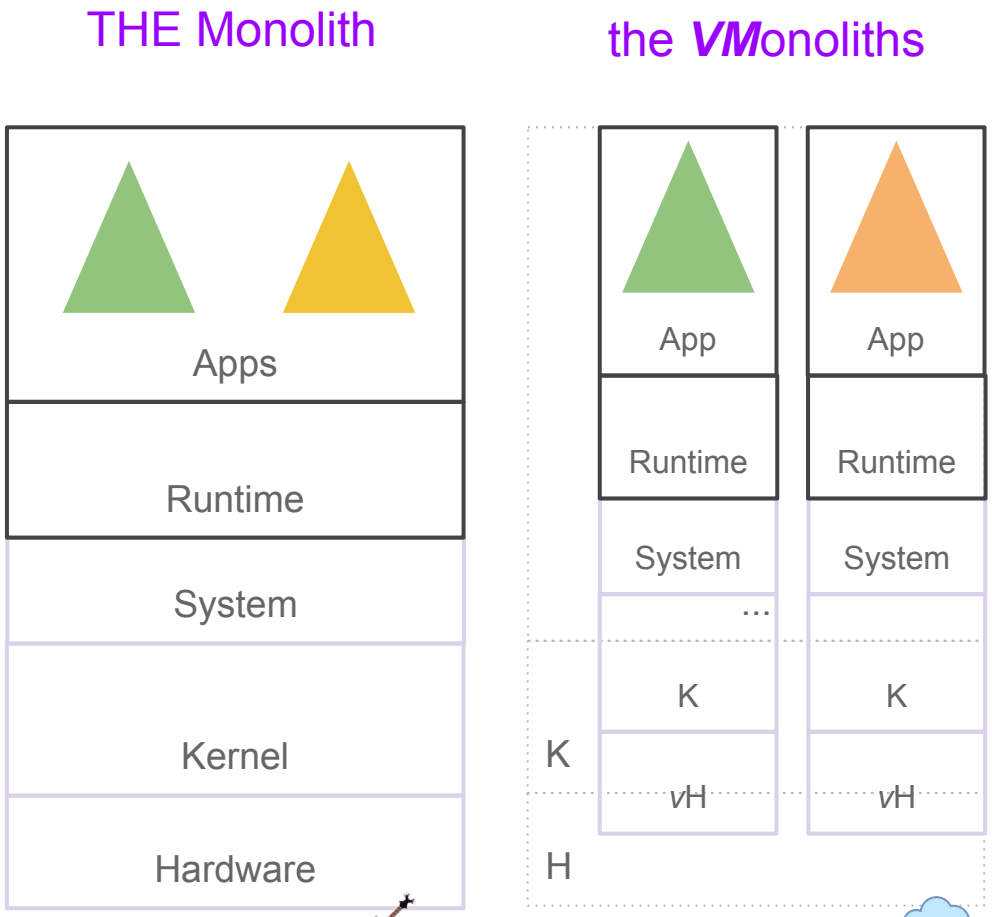
Cloud Native - un poco de arqueología ...

THE Monolith



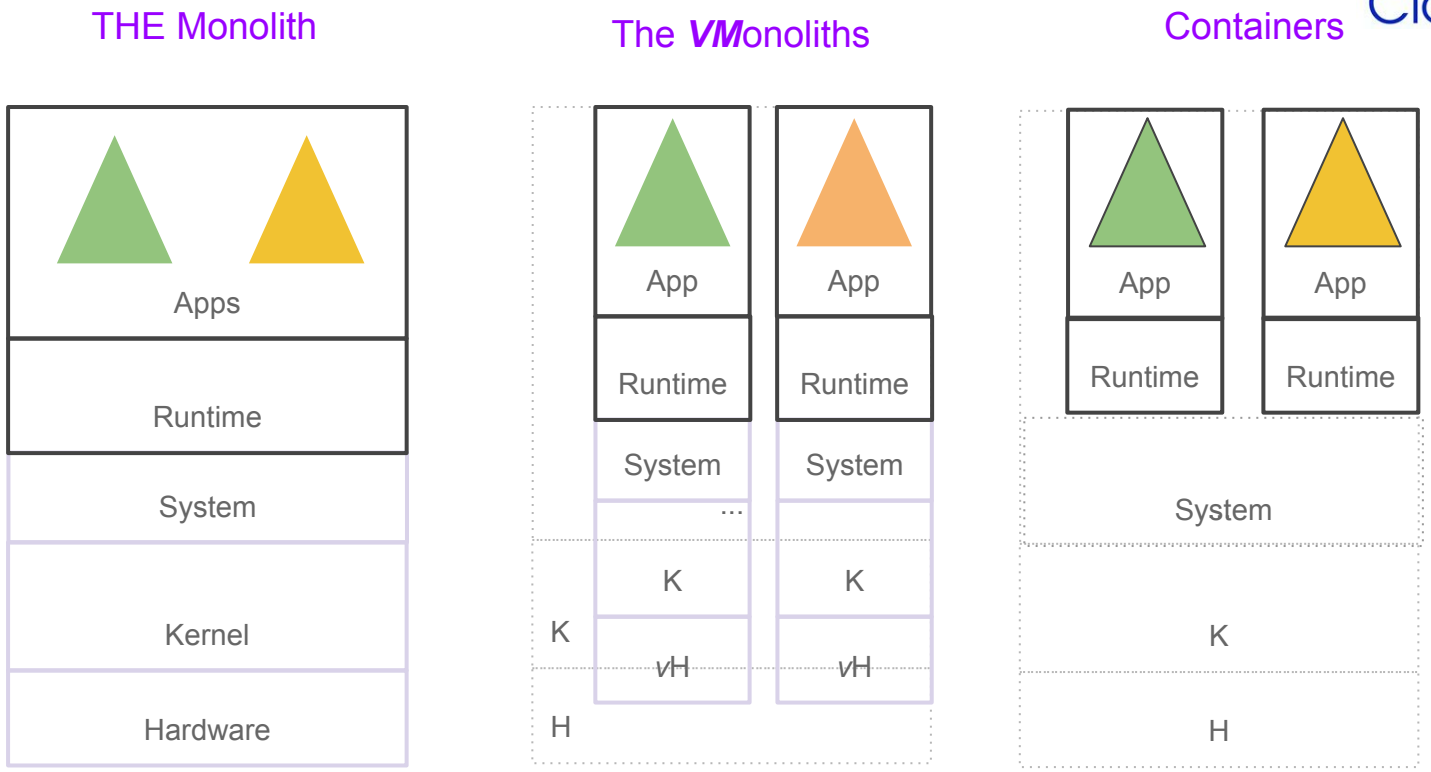
platform	METAL
orchestration	imperative
tools	***human***, scripts, ansible, chef

Cloud Native - más cerca en el espacio-tiempo



platform	METAL	Cloud *Classic*®
orchestration	imperative	+= ~declarative
tools	***human,*** scripts, ansible, chef	+= ~terraform

Cloud Native - desengrasándonos del sistema



platform	METAL	Cloud *Classic*®	Cloud *Native*
orchestration	imperative	+= ~declarative	declarative
tools	***human***, scripts, ansible, chef	+= ~terraform	Kubernetes, Docker SWarm

Cloud Native

Tecnologías que empoderan a las empresas para construir aplicaciones escalables en ambientes dinámicos de cloud.

Estas **Tecnologías** permiten crear:

- Sistemas con bajo acomplamiento
- Resistentes[Resilientes]
- Gestion[ables]
- Observables

Combinadas con automatización robusta permite a los ingenieros hacer cambios de alto impacto con frecuencia y mínimo esfuerzo.

¿ Cómo sería ? …

Beyond 12FactorApps

1. *One codebase, one application*
2. ***API first***
3. *Dependency management*
4. *Design, build, release, and run*
5. *Configuration, credentials, and code*
6. *Logs*
7. *Disposability*
8. *Backing services*
9. *Environment parity*
10. *Administrative processes*
11. *Port binding*
12. *Stateless processes*
13. *Concurrency*
14. *Telemetry*
15. *Authentication and authorization*

https://github.com/cncf/toc/blob/main/DEFINITION.md
https://www.oreilly.com/webops-perf/free/beyond-the-twelve-factor-app.csp

Containers
Docker

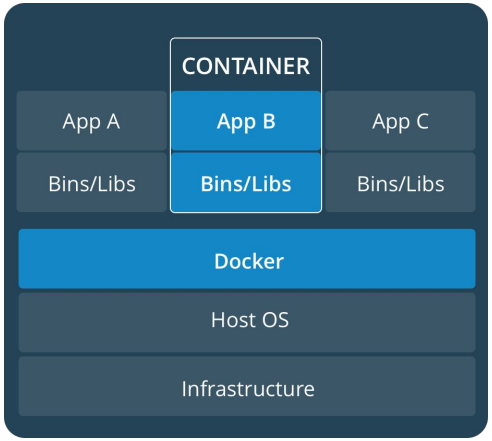
Docker: Conceptos

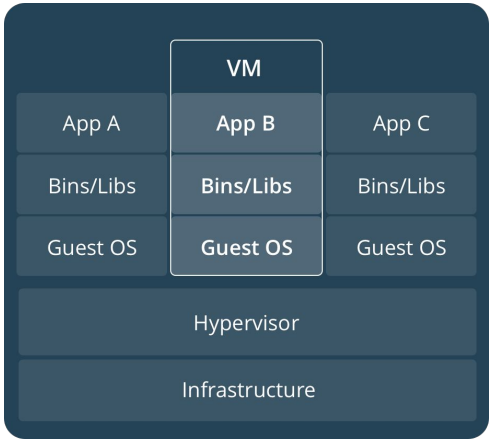
Una plataforma para:

“desarrollar, desplegar y ejecutar aplicaciones con containers”

Una “**imagen**” es un paquete ejecutable que contiene todo lo necesario para ejecutar una aplicación [código, librerías de ejecución, ambiente y configuración]

Un “**container**” es una instancia de ejecución de una imagen.





Ref: https://docs.docker.com/get-started/

cURL un amigo … y el peluquero **jq**


```
sudo apt install jq pv

client URL, transferencias de recursos especificados como URLs usando pro

MUY usado como CLI/LIB para APIs REST

[Otro muy usado con GUI: Postman]

jq: “sed” para JSON


Ejemplos:

curl -s https://www.creditos.com.ar/wp-json/wp/v2/posts                    # Despeinado

curl -s https://www.creditos.com.ar/wp-json/wp/v2/posts | jq . | less      # Peinado

curl -s https://www.creditos.com.ar/wp-json/wp/v2/posts | jq .[]."link"|less   # Pá la foto…

Perlitas..

curl -s http://wttr.in/MENDOZA

curl -s http://artscene.textfiles.com/vt100/movglobe.vt  | pv -q -L 9600

curl -s http://artscene.textfiles.com/vt100/firework.vt  | pv -q -L 4800


Mas -> http://artscene.textfiles.com/vt100/
```

Docker: CLI

```
## Ejecutamos una imagen de ejemplo
docker run hello-world
#--name asigna nombre (ps,kill,exec), -d Daemon mode]
docker run --name websrv -d  nginx
## Ejecutamos un comando dentro de un container en ejecución
docker exec -it websrv bash
## Listamos las imágenes disponibles localmente
docker image ls
## Listamos los containers
docker container ls
## Listamos los containers en ejecución
docker ps
## Detenemos un container en ejecución
docker stop <container_id|name>
## Matamos un container en ejecución
docker kill <container_id|name>
## Eliminamos un container [apagado]
docker rm <container_id|name>
## Ejecutamos una imagen de la distro Alpine en modo interactivo
docker run -it alpine
```

Ref: https://docs.docker.com/get-started/

Docker: Dockerfile

Un Dockerfile define que va dentro del ambiente de un container.

```
FROM alpine:3.7

LABEL mantainer="cloud@um.edu.ar"

ENV TERM xterm

RUN apk update
RUN apk add python py2-pip uwsgi uwsgi-python

RUN mkdir /opt
RUN mkdir /opt/validate-email

WORKDIR /opt/validate-email
COPY src/ /opt/validate-email

RUN RUN pip install --upgrade pip
RUN pip install -r requirements.txt

USER 1000:1000

CMD uwsgi --ini /opt/validate-email/validate-email.ini

EXPOSE 5000
```

FROM: Origen de la imagen Base

LABEL: Adiciona Metadata

ENV: Setea variables de entorno

RUN: Ejecuta en un shell en la imagen

WORKDIR: Setea el directorio por default

COPY: Copia a la imagen

USER: Setea el user para RUN,CMD,ENTRYPOINT

CMD: Ejecución por defecto, solo 1

EXPOSE: Publica un puerto del container

Ref: https://docs.docker.com/get-started/

Docker: Construyendo nuestra app validate-email

```
## Actualizamos el repo
git clone  https://github.com/umcloud/clases-devops.git
## Ingresamos a las apps en el repo, c03-docker, práctico 01 [p01]
cd clases-devops/c03-docker/p01/validate-email/
docker build -t validate-email:p01 .
## Ejecutamos la app
docker run -d --name validate-email-p01 -p 80:5000 -t validate-email:p01
## Probamos
## Modo validación sintaxis
curl http://56ca8f8738ecc03cc4b2859db342e243@localhost/diego.navarrow@um.edu.ar/0
{"validate": true}
## Modo validación sintaxis + dominio existente
 curl http://56ca8f8738ecc03cc4b2859db342e243@localhost/diego.navarrow@um.edu.ar/1
{"validate": true}
## Modo validación sintaxis + dominio existente + casilla existente
curl http://56ca8f8738ecc03cc4b2859db342e243@localhost/diego.navarrow@um.edu.ar/2
{"validate": false}
## Matamos el container
docker kill validate-email-p01
```

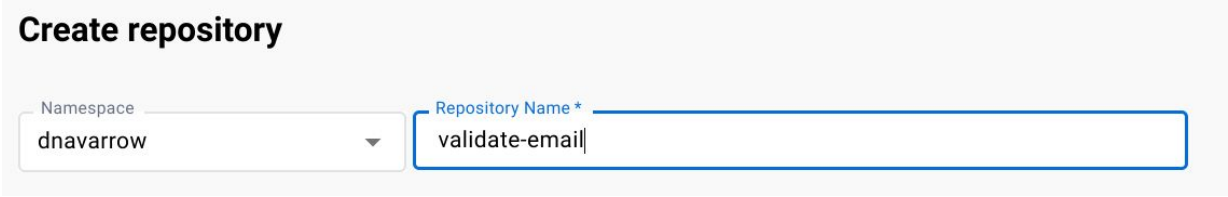
#Tip solo para “habitantes de UM-Cloud” - Sirve para estudiar MSS
iptables -I FORWARD -p tcp --tcp-flags SYN,RST SYN -j TCPMSS --set-mss 800

Docker: Comunicándonos con el “Entorno”

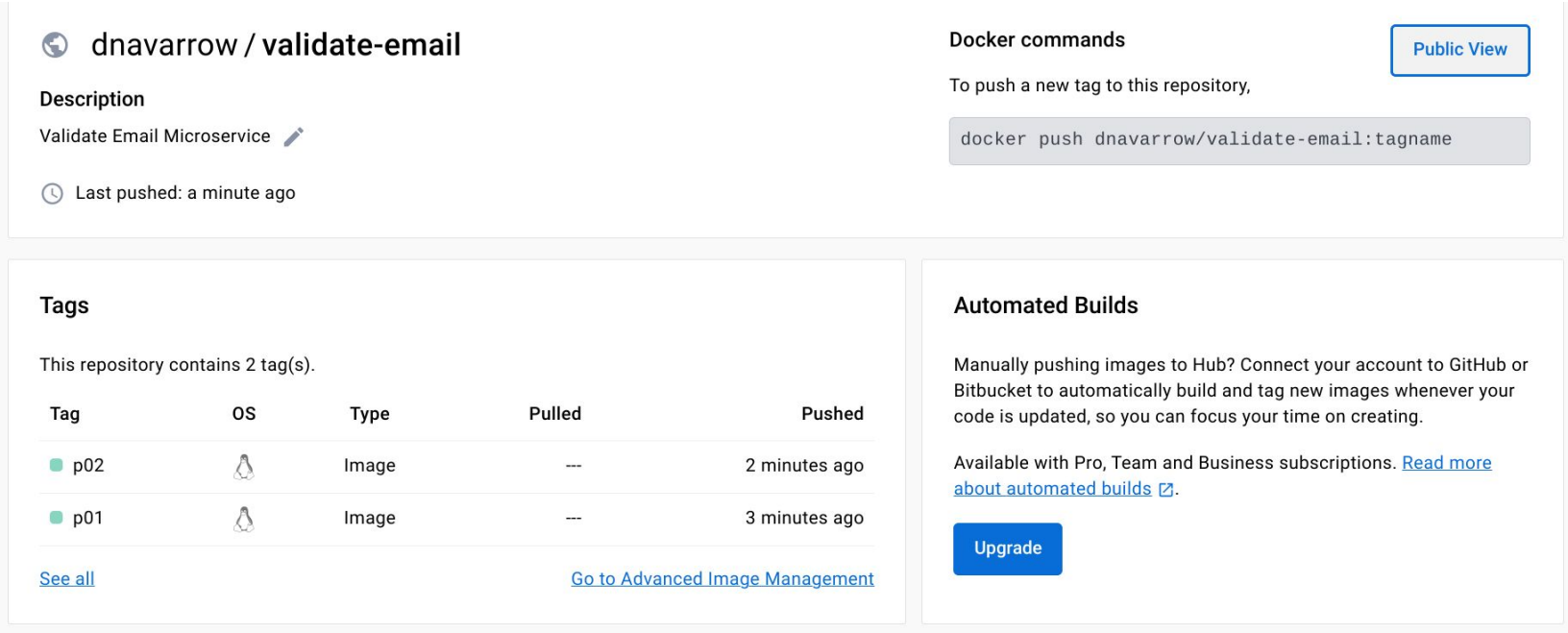
```
## Ingresamos a la app en el repo, c03, práctico 02,
cd c03-docker/p02/validate-email

## Hacemos el build e intentamos ejecutarlas
docker build -t validate-email:p02 .


## Intentamos Ejecutar la app
docker run -d --name validate-email-p02 -p 80:5000 -t validate-email:p02


## ¿que paso?
docker logs validate-email-p02
docker rm validate-email-p02
## Intentamos Ejecutar la app con Su variable de Entorno
docker run -d --name validate-email-p02 -p 80:5000  --env API_KEY=1234 -t validate-email:p


## Probamos
curl http://1234@localhost/diego.navarro@um.edu.ar/2
{"validate": true}
```

Docker: Publicando en una **registry**

```
## Nos Registramos en docker hub https://hub.docker.com/signup/

## Creamos un repositorio para la aplicación
```




```
## Tomamos las credenciales para loguearnos en la registry

docker login
## Taggeamos las imágenes y la publicamos
#Docker Hub
docker tag validate-email:p01 TU_USUARIO_DOCKERHUB/validate-email:p01
docker push TU_USUARIO_DOCKERHUB/validate-email:p01
```



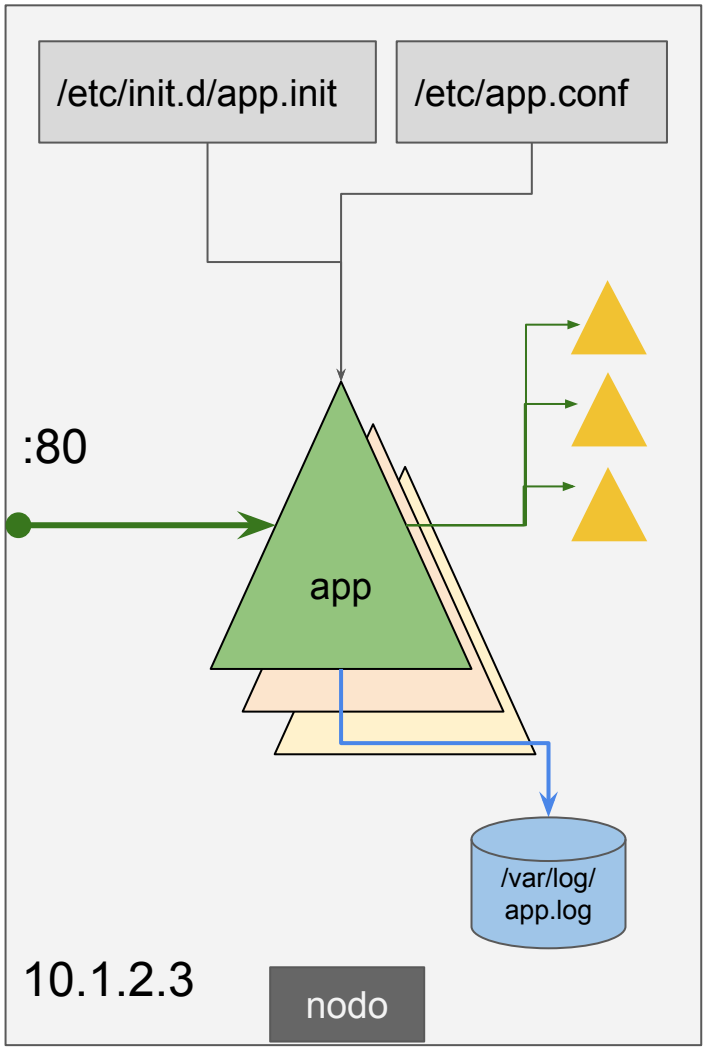
Cloud Native Kubernetes

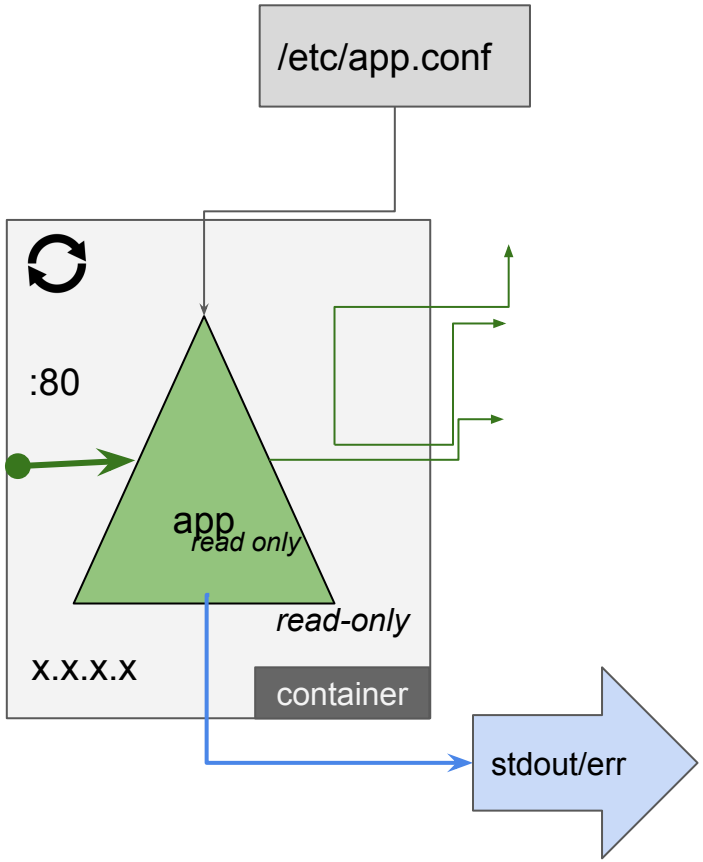
Infraestructura basada en containers

Un Infraestructura basada en containers requiere de una plataforma que provea:

- **Scheduling**:
  - Decida dónde deberían ejecutar los containers
- **Lifecycle and health**:
  - Mantenga los containers corriendo a pesar de los fallos
- **Scaling**:
  - que haga que un conjunto de containers se estire o contraiga
- **Naming and discovery**:
  - Encuentre dónde están los containers
- **Load Balancing:**
  - distribuya tráfico través de un conjunto de containers
- **Storage volumes**:
  - Provea datos a los containers (stateful vs stateless)
- **Logging and monitoring**:
  - Registre qué pasa con los containers
- **Debugging and introspection**:
  - Ingresar/Unirse a un container
- **Identity and authorization**:
  - Controlar quién puede hacer que

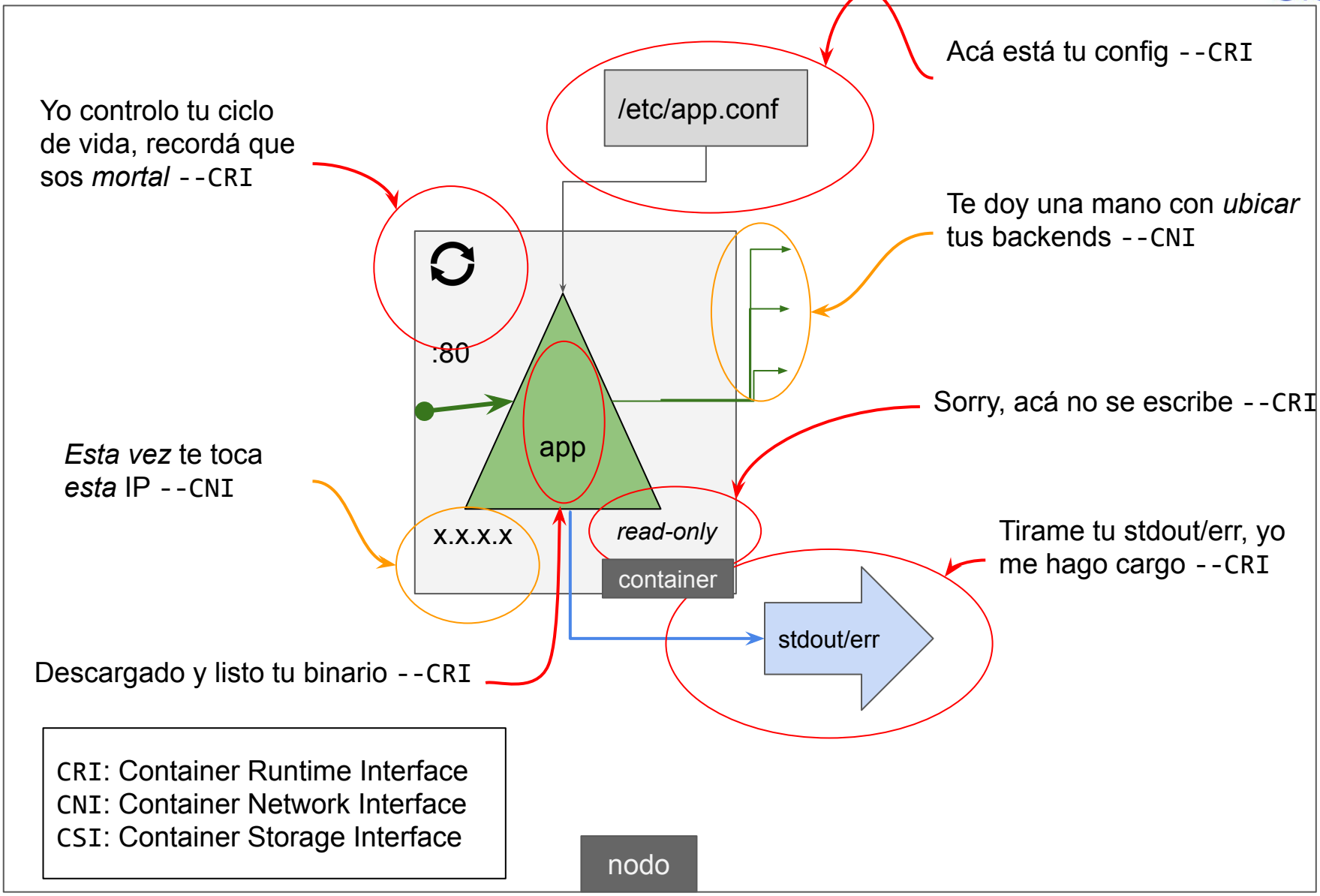
Cloud Native - del monolito al nativo





Cloud Native - entorno de un *container*

UM Cloud



Kubernetes para todes 1de2

- https://kubernetes.io/
  Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications.
- Término “Kubernetes”:
  - de la palabra Griega "Timonel":
  - también es la raíz de las palabras "gobernador" y "cibernética"
- API de workloads declarativo
- Plataforma para automatizar despliegue, escalado y operación
- 100% Open Source (Apache license), escrito en Golang
  - Basado en la experiencia de Google con sus cluster-managers internos (Borg, Omega)
  - Kubernetes v1.0: liberada el 21/julio/2015
- *In short*: Orquestador de containers

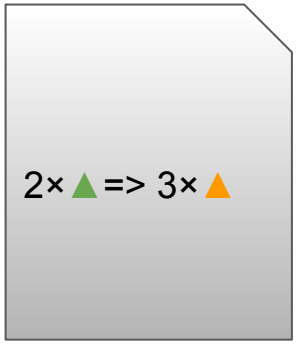
Kubernetes para todes 2de2

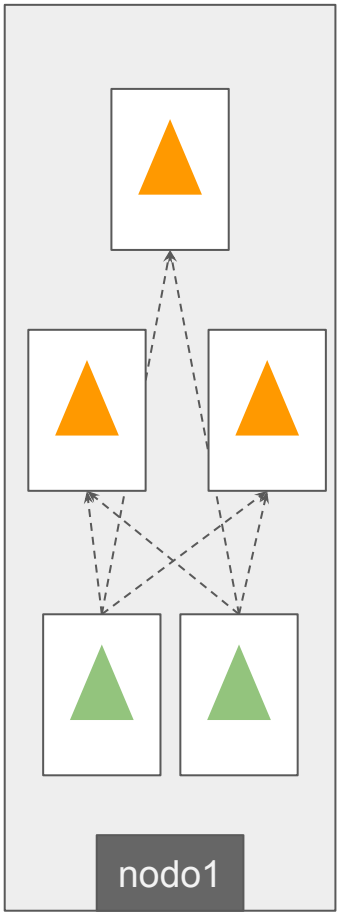
- "Containers cluster orchestrator":
  - **Declarative Spec**
  - Componentes:
    - CRI: manejo del container runtime en c/nodo
    - CNI: conectividad intra, extra cluster, descubrimiento de "endpoints" (frontend -> backend), "firewall" as a service, "virtualhost" as a service
    - CSI: provisión y *lifecycle* de almacenamiento persistente
  - Multiplataforma:
    - **GKE**: Google Kubernetes Engine (Google)
    - **EKS**: Elastic Kubernetes Service (AWS)
    - **AKS**: Azure Kubernetes Service (Microsoft)
    - CCE: Cloud Container Engine (*Huawei Cloud*)
    - *DIY*:
      - dev: k3d, kind, minikube (p/dev)
      - prod: rke, kubespray, kubeadm
    - … desde Raspberry-PI hasta 5000 nodos
- **Lingua franca!** para *Cloud Native Workloads*
  - spec: YAML, JSON
  - **repo** (ejemplo)**:**
    ./src/app/
    ./src/deploy/

2× ▲ => 3× ▲
`mi_stack.yaml`

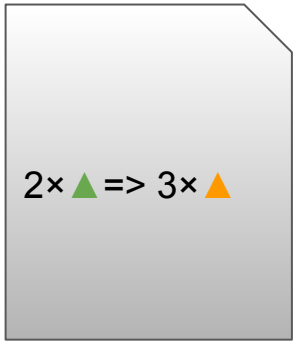
- Complejo para comenzar a aprender
  - "Kubernetes dificulta las cosas simples, y hace posible las (muy) complejas"

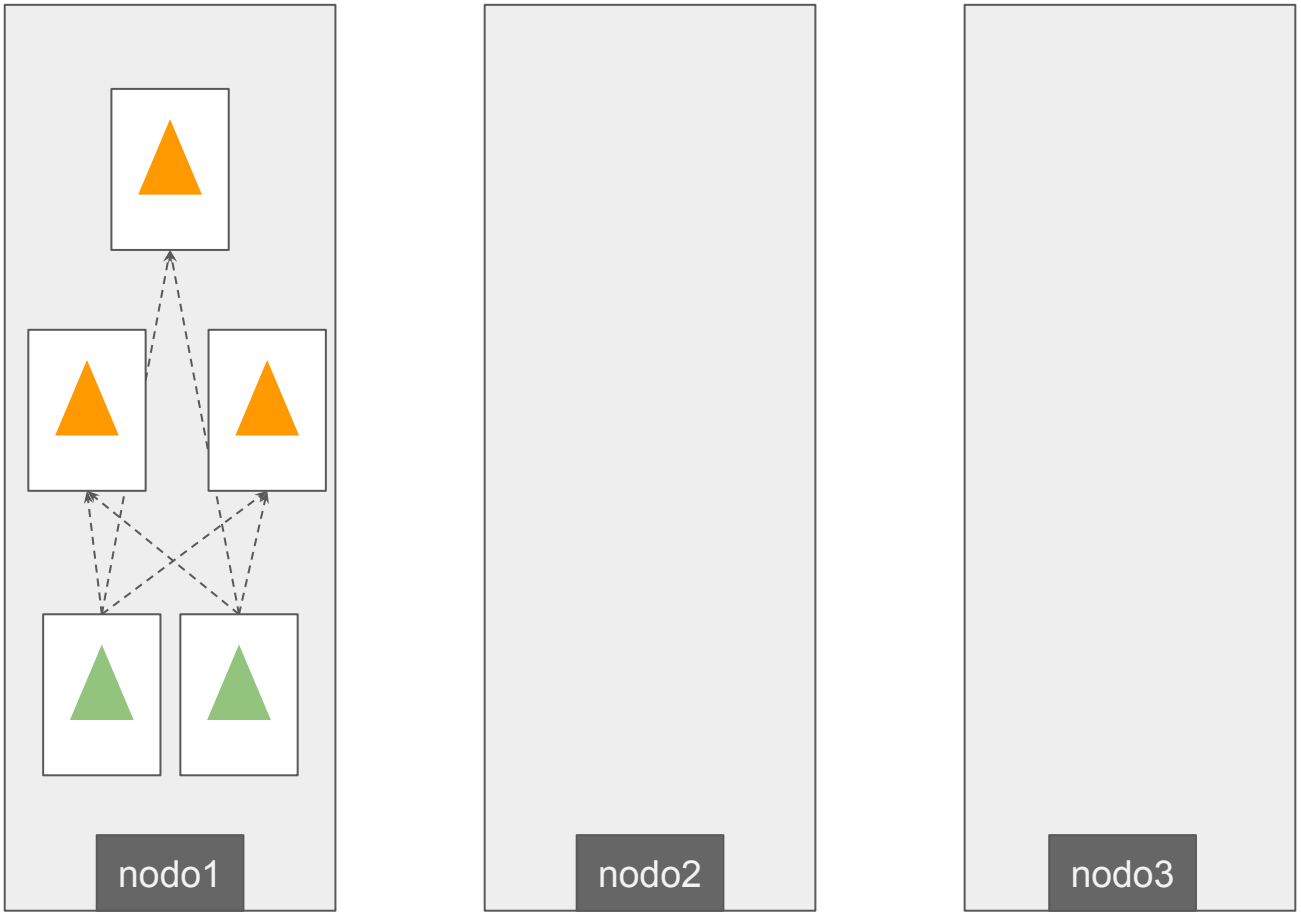
Cloud Native - a reproducirse !



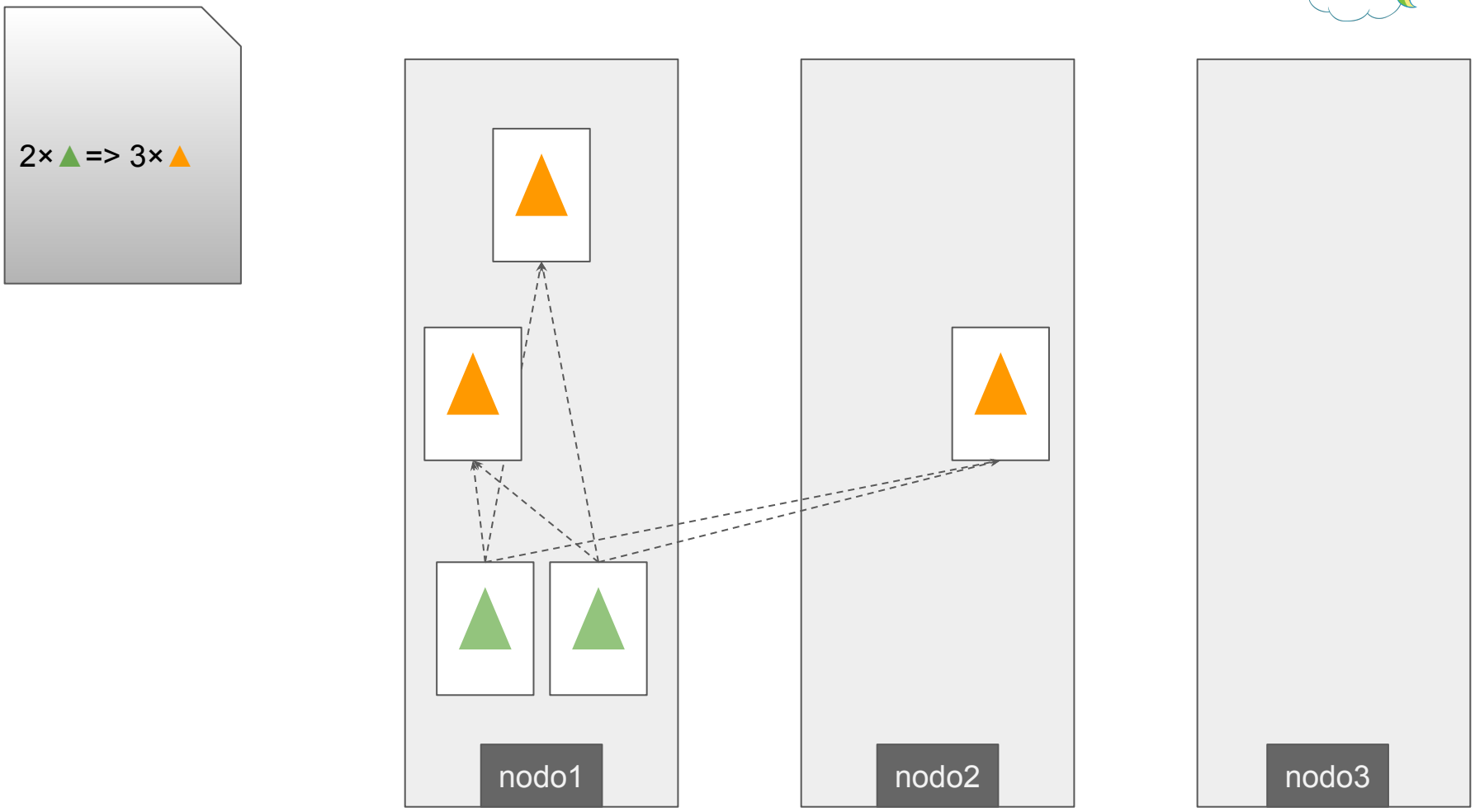


Cloud Native - a reproducirse !

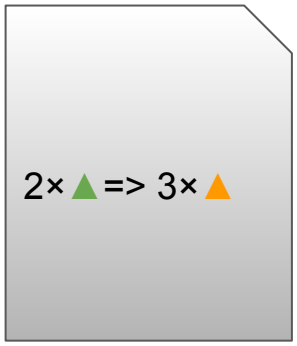


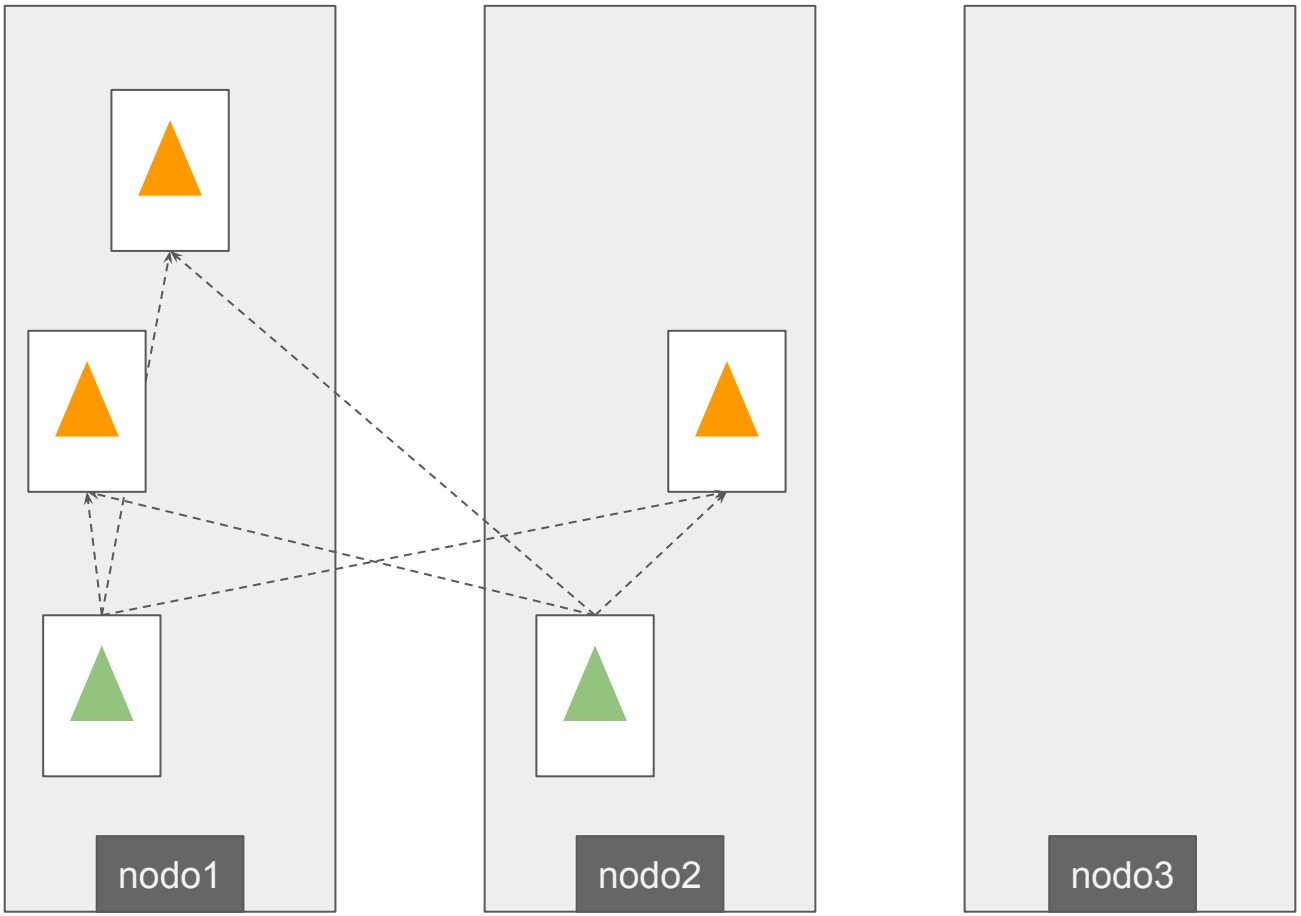


Cloud Native - a reproducirse !

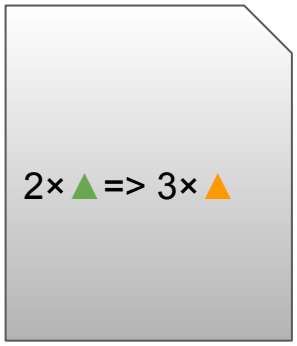


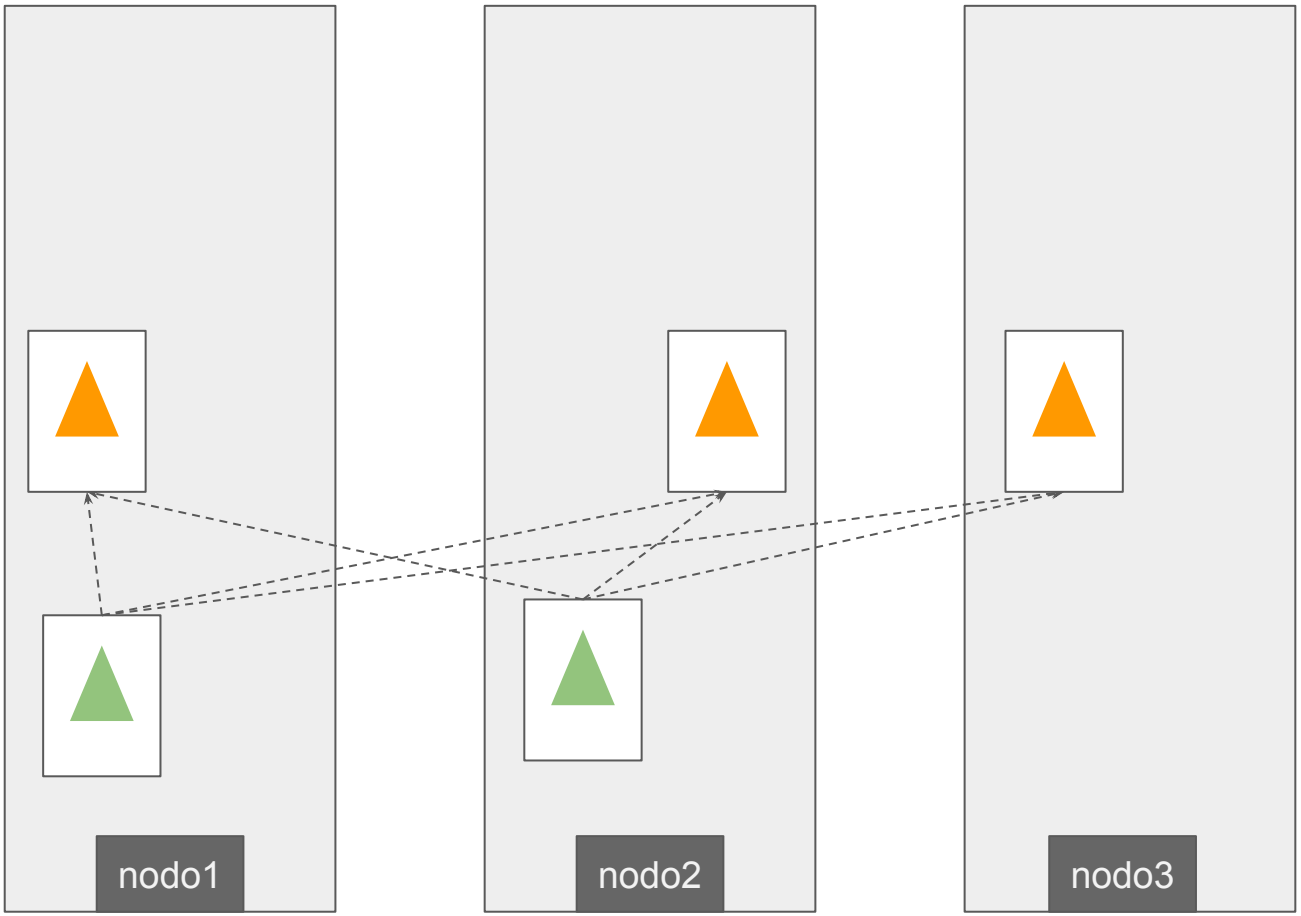
Cloud Native - a reproducirse !



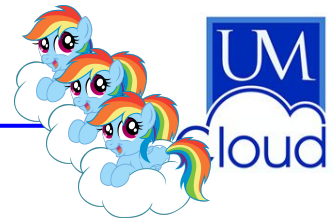


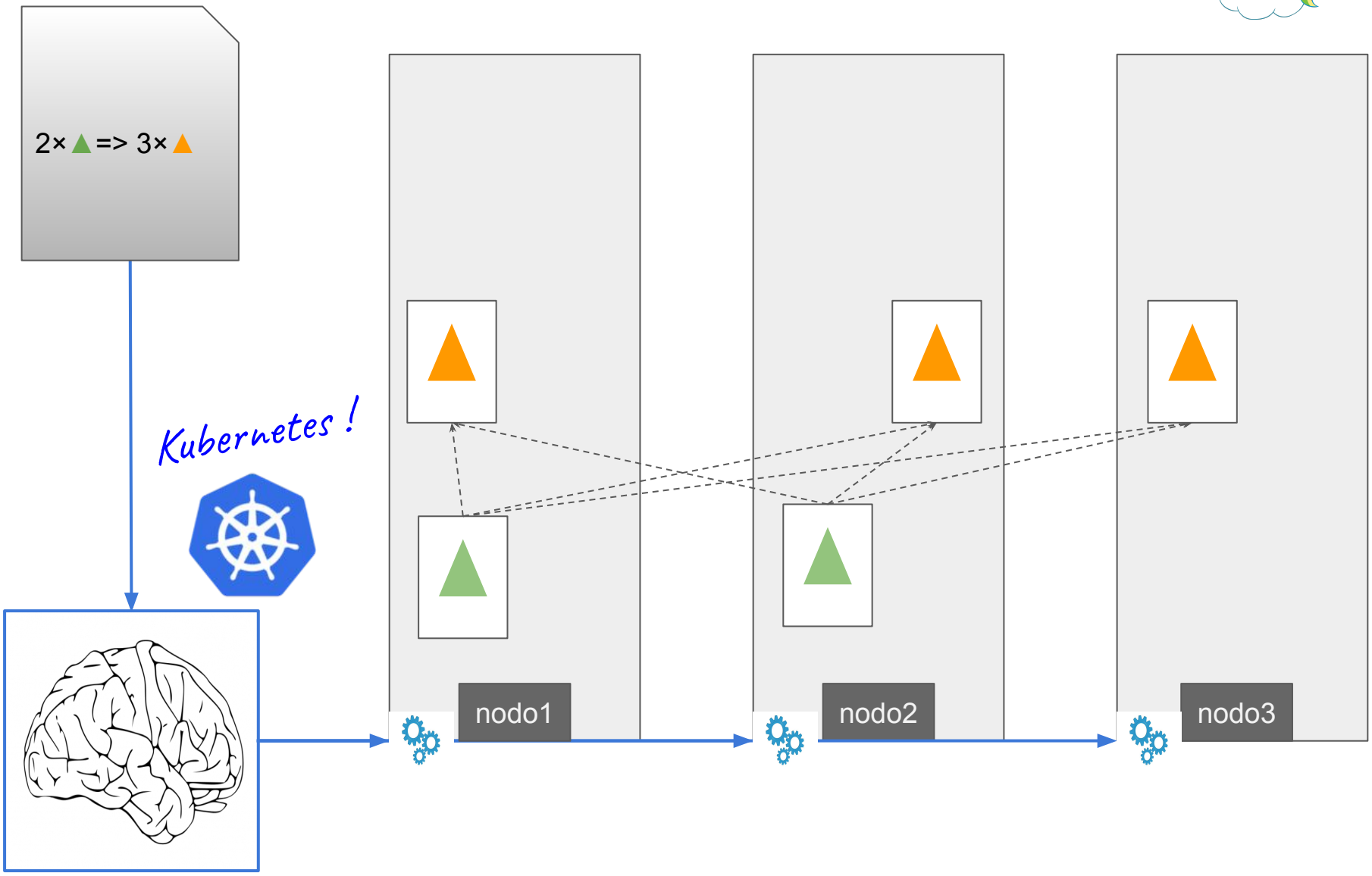
Cloud Native - a reproducirse !



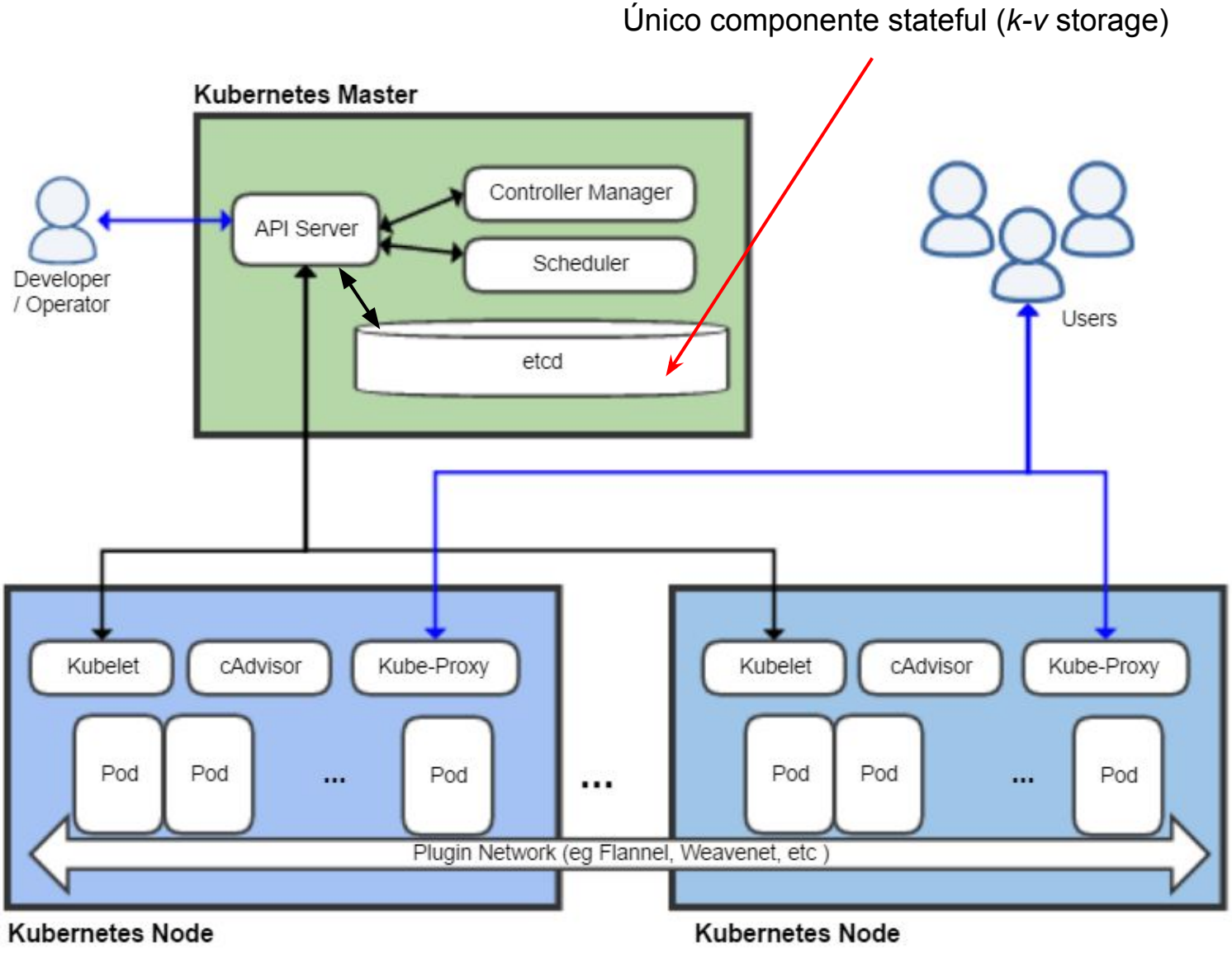


Cloud Native - a reproducirse !





Kubernetes - Arquitectura



Kubernetes: primeros pasos -> `kubectl`

- Loggearse a https://rancher.kube.um.edu.ar/dashboard/ usando tu cuenta de https://github.com/ (como miembro de `umcloud-external/students-2023`)
  - Crear una API KEY desde https://rancher.kube.um.edu.ar/dashboard/account/create-key ->
    - Scope: **No scope**
    - Expire: **Never**
  - Guardar el `Bearer Token` que muestra "**token-xxxxx:abc...xyz**"
- Crear una VM usando la imagen: **`um-kube-tools`** (m1.small)
- Loggearse en la VM, ejecutar lo que muestra el comando: `um-help`
  - **`kube-setup.sh`** ***`BEARER_TOKEN USUARIO`***

```
# USUARIO: por ejemplo tu nick en slack
Select a Project:3                  # cluster-01     c-.....:p-xxxxx   Default
```

  - **`kube-create-ns.sh NAMESPACE`**

```
# NAMESPACE: USUARIO-dev
```

Explorando `kubectl`

```
kubectl version --short

kubectl get componentstatus

kubectl get nodes

kubectl auth can-i delete node

kubectl auth can-i list nodes
```

Kubernetes object: `Pod`

- Es el objeto elemental de *workload* de Kubernetes
  - unidad básica de "scheduling"
  - todos los otros objetos se construyen "on-top" de pods
- Puede estar formado por uno o más containers
  - típicamente uno solo
- Tiene una dirección IP única dentro del cluster (1:1)
- Son *mortales* por diseño (*Cloud-Native*)
- Pueden tener *attached* recursos tales como (*12factor*):
  - `configmaps`
  - `secrets`
  - `volumes`
- Probar:

  ```
  kubectl explain pod
  ```

Kubernetes object: Pod - *hands-on!*

```
# Creamos un pod simple con image=nginx
kubectl run --restart=Never --image=nginx mipod


# Ver estado, y detalles
kubectl get pod
kubectl describe pod mipod


# Ver estado++
kubectl get pod -owide
kubectl get pod -owide --show-labels

# Ver el universo ... (de pods corriendo)
kubectl get pod --all-namespaces -owide
```

Kubernetes object: Pod - *hands-on++!*

```
# Entrañas del pod:
kubectl get pod mipod -oyaml | less


# ↖ ¿ Cuáles campos especificamos nosotros ?, comparar con ↘
kubectl run --restart=Never --image=nginx mipod --dry-run=client
-oyaml

# Entremos al pod:
kubectl exec -it mipod -- /bin/bash

# root pod ! :P . Cambiemos su imagen:
kubectl set image pod/mipod mipod=bitnami/nginx
kubectl logs mipod
kubectl exec -it mipod -- /bin/bash

# Chau pod:
kubectl delete pod mipod
```

Kubernetes object: `namespace`

- Permite separar "contextos", *think-of*: "resource folders"
- Facilita definir bordes de:
  - seguridad
  - planificación (schedule-ability)
  - valores x default, tope de recursos
- Casos de uso:
  - dev vs staging vs production
  - namespace por usuario (nuestro caso) o projecto
  - namespaces "del sistema": `kube-system` …
- Probar:

```
kubectl get ns
kube-create-ns USUARIO-prod
kubens USUARIO-dev  # volver al previo NS
kubectl delete pod --namespace=kube-system --all --dry-run=server
```

Kubernetes: más allá de los PODs

- **Deployments**
  - arreando pods ...
- **Services**
  - cómo llego a mis pods ?
- **ConfigMaps**
  - cómo configuro mis containers (dentro de los pods) ?
- **Secrets**
  - cómo proveo configuración de secretos
- **StatefulSets**, **DaemonSets**
  - deployments + volúmenes persistentes
- **Ingress**
  - cómo llego a mis pods via *http virtualhost* ?

p01- Kubernetes object: Deployment

IMPORTANTE: Loggearse a la VM con ->

```
ssh -L 8080:localhost:8080 -L 8001:localhost:8081 -A ubuntu@IP
```

- Actualizamos el repo

```
git clone https://github.com/umcloud/clases-devops
cd clases-devops
git pull
cd c04-kube-deploy_svc_cm/p01
```

Deployment

ReplicaSet

- Creamos un deploy, modo declarativo via YAML

```
kubectl create deployment --image=bitnami/nginx:1.24.0 web --dry-run=client -oyaml > web.deploy.yaml
kubectl apply -f web.deploy.yaml
kubectl get deploy,rs,pod                    # ¿ Y esos nombres de pods, algún patrón ?
```

- Update: 1->3 replicas, y última versión de image de nginx

```
~~kubectl scale deployment web --replicas=3~~      # <-- ¿ Estaría bien hacer esto ?
sed -i -e /nginx:/s/1.24.0/1.25.0/ -e /replicas:/s/1/3/ web.deploy.yaml
kubectl apply -f web.deploy.yaml
kubectl rollout status -f web.deploy.yaml      # igual a: … rollout status web
kubectl rollout history -f web.deploy.yaml     # igual a: … rollout history web
kubectl get deploy,rs,pod                      # ¿ Qué cambió ?
```

- Rollback/rollforward via kubectl rollout undo ...

p01- Kubernetes object: `Deployment`

- Objeto de más "alto" nivel, controla *Pods* via *ReplicaSets*
- Tiene estrategia de rollout:

```
kubectl get deployment web -oyaml | grep -A3 strategy:
```

  - `maxSurge`: cuánto se puede "inflar" (en pods) el deploy durante rollout
  - `maxUnavailable`: cuántos pods pueden estar *down* durante rollout
- Objecto intermedio: ReplicaSet
  - observar el *output* del último `kubectl get deploy,rs,pod`
- Exploremos un poco:

```
kubectl get pod
kubectl get pod web-xxxxxxxx-yyyyy
kubectl get pod -l app=web
kubectl logs web-xxxxxxxx-yyyyy                         # kubectl logs deploy/web
kubectl exec -it web-xxxxxxxx-yyyyy -- /bin/bash        # kubectl exec -it deploy/web -- /bin/bash
```

- Ahora … ¿ Cómo lo accedemos ?

```
kubectl get pod
kubectl port-forward deploy/web 8080:8080 &
curl http://localhost:8080/      # o via browser
kill %1                          # mato el port-forward corriendo en background
```

- Todo ésto ^^^ **no sirve** para production, ¿ Por qué ?

p02- Kubernetes object: Service

- ¿ Y cómo exponemos el *Deployment*? via un *Service:*

```
cd ../p02
kubectl expose deploy web --port=80 --target-port=8080 --dry-run=client -oyaml >web.svc.yaml
less web.svc.yaml  # Notar: ports y selector
kubectl apply -f web.svc.yaml
```

- ¿ Qué ocurrió ?


```
kubectl get svc,ep,pod -owide
```

  - notar las IPs en juego …

- Accediendo el servicio desde dentro del cluster

```
kubectl run -it --rm --restart=Never --image=alpine alp-shell  # pod efímero
/ # apk add curl
/ # curl -v http://web
/ # ping web       # criKcriK …
```

p02- Kubernetes object: `Service`

- Son efectivamente *loadbalancers internos* de Kubernetes
  - provistos por la plataforma, via CNI

- Típicamente a cargo del `kube-proxy` que corre en cada nodo
- Hay varios tipos, de acuerdo a su *reachability*:
  - **`ClusterIP`** (default)
    - accesible desde **dentro** del cluster (nodos y pods)
      - Típicamente usado para conectar micro-services internos, o necesario para →
  - **`NodePort`**
    - ClusterIP + **ports dedicados en cada nodo worker**
      - Por ejemplo para apuntar un *loadbalancer* externo a todos los nodos, mismo port
  - **`LoadBalancer`**
    - NodePort + creación automática de LB provisto por el Cloud Provider

p03- Kubernetes object: ConfigMap

- Objetivo: Ponerle data al servicio web via configmap (alias cm), el cual contiene un conjunto de key=values, ej:

```
kubectl create cm app-cfg --from-literal=foo=uno --from-literal=bar=dos --dry-run=client -oyaml
```

- Construir un configmap con data de un directorio

```
cd ../p03
mkdir -p assets
echo "hola mundo desde $USER" > assets/index.html
kubectl create cm --from-file=assets/ web-assets --dry-run=client -oyaml>web-assets.cm.yaml
kubectl apply -f web-assets.cm.yaml
kubectl get cm
kubectl get cm web-assets -oyaml
```

- Web (re)deploy montando este cm, usando el yaml provisto (`web.deploy.yaml`):

```
diff -u ../p01/web.deploy.yaml web.deploy.yaml
kubectl diff -f web.deploy.yaml
kubectl apply -f web.deploy.yaml
```

- Accediendo al servicio desde tu laptop (`README.md`)

```
kubectl proxy &         # Reemplazar YOUR_NAMESPACE debajo ->
curl http://localhost:8001/api/v1/namespaces/YOUR_NAMESPACE/services/web/proxy/
```

p03- Kubernetes object: `ConfigMap`

- Es un objeto que contiene un conjunto de **`key: value`**, en su campo **`data`**
- Fundamentalmente usado para configurar los parámetros runtime de las aplicaciones
- Se puede "consumir" como:
  - **Archivos** mapeados en en container (key: filename, value: contenido)
  - **Environmental** variables
  - Command line **arguments** $(ENV_VAR)

p04- Kubernetes object: Secret

- **Objetivo: Arrancar un motor de base de datos [**usamos https://hub.docker.com/_/mysql/]

```
cd ../p04
kubectl apply -f 01-db-simple.deploy.yaml
```

- ¿Qué ocurrió? puuffffff…

```
kubectl get pod -owide
kubectl logs deploy/db-simple    # NOTAR deploy/ en vez del pod
```

Un secreto es un objeto que contiene una pequeña cantidad de datos confidenciales, como una contraseña, un token o una clave. [**12f++**: no debe quedar información sensitiva dentro de las imágenes/repos] *Solución: "volúmenes" tipo secreto*.

```
kubectl create secret generic db-secret \
     --from-literal=db-username=root \
     --from-literal=db-password=secretisimo \
     --from-literal=db-name=whois -o yaml \
     --dry-run=client > 02-db-prod.secret.yaml

kubectl apply -f 02-db-prod.secret.yaml
kubectl apply -f 03-db-prod.deploy.yaml
```

- ¿Qué ocurrió?

```
kubectl get deploy,secret,pod -owide
kubectl exec -it deploy/db-prod -- /bin/bash mysql -uroot -psecretisimo
```

p04- Kubernetes object: Secret & ConfigMap

- **Objetivo: Arrancar un motor de base de datos con su SCHEMA ya creado.**

  Creamos un ConfigMap con el código sql para crear el SCHEMA de la DB y luego Deployamos

```
kubectl create cm db-sql --from-file=schema.sql -o yaml --dry-run=client > 04-db-prod-schema.cm.yaml
kubectl apply -f 04-db-prod-schema.cm.yaml
kubectl apply -f 05-db-prod-schema.deploy.yaml
```

  * Este deploy usa los secrets de `02-db-prod.secret.yaml`

- ¿ Qué ocurrió ?

```
kubectl get deploy,secret,pod -owide
kubectl exec -it deploy/db-prod-schema -- /bin/bash
```

- ¿ Le metemos data ?

```
cat data.sql | kubectl exec deploy/db-prod-schema -it -- mysql -u root -psecretisimo whois
```

- ¿ Hay data ?

```
echo 'select cidr,json_extract(whois,"$.network.name") from whois;' | \
    kubectl exec deploy/db-prod-schema -it -- \
        mysql -r -s -u root --password=secretisimo whois
```

Kubernetes: más allá de los PODs

- **Deployments**
  - arreando pods ...
- **Services**
  - cómo llego a mis pods ?
- **ConfigMaps**
  - cómo configuro mis containers (dentro de los pods) ?
- **Secrets**
  - cómo proveo configuración de secretos
- **StatefulSets**
  - deployments + volúmenes persistentes
- **Ingress**
  - cómo llego a mis pods via *http virtualhost* ?

p01- Kubernetes object: Deployment + PVC [PV]

`PersistentVolume` (PV): una porción de almacenamiento en el cluster aprovisionado a tal fin. [un "recurso" como un nodo]

`PersistentVolumeClaim` (PVC): es un requerimiento de almacenamiento hecho por un usuario del cluster.

`StorageClasses` (SC): Para ofrecer una variedad de tipos de PVs [tamaño, perf] sin exponer al usuario los detalles de implementación.

- Actualizamos el repo `(https://github.com/umcloud/clases-devops.git)`

```
cd clases-devops
git pull
cd c05-kube-pvc_sts_ingress/p01
```

- **Objetivo: Arrancar un motor de base de datos con su SCHEMA ya creado y que persista la información**

```
# Creamos el pvc
kubectl apply -f 01-db-prod-pvc.yaml

# Adicionamos Secret MYSQL_DATABASE, MYSQL_ROOT_PASSWORD Env
kubectl apply -f 02-db-prod.secret.yaml

# Usamos ConfigMap para schema+data
kubectl apply -f 03-db-prod.configmap.yaml

# Deploy: mysql+initContainer+pvc+ConfigMap
kubectl apply -f 04-db-prod.deploy.yaml
```

p01- Kubernetes object: Deployment + PVC[PV]

deploy

pvc

pod

```
$ kubectl get deploy,pvc,pod,cm
NAME                        READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/db-prod     1/1     1            1           8m19s

NAME                                  STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGEC
persistentvolumeclaim/mysql-data      Bound    pvc-0163cf76-6126-4b66-a6cc-037dfcb23a99   1Gi        RWO            longhorn

NAME                             READY   STATUS    RESTARTS   AGE
pod/db-prod-9fccb67cf-4m4qh      1/1     Running   0          8m21s

NAME                         DATA   AGE
configmap/db-schema-data     2      8m23s
```

p02- Kubernetes object: `StatefulSet (sts)`

- `StatefulSet` (STS): gestiona el despliegue y escalado de un conjunto de Pods, provee garantía en el ordenamiento y unicidad de esos Pods, y acoplamiento "ordenado" con los PVCs creados.

- **Objetivo: Arrancar un motor de base de datos con su SCHEMA ya creado, que persista la información y ofrecerlo como un servicio**

```
cd ../p02
kubectl delete -f ../p01
kubectl get pvc              # b00M, gone, kaput

# Creamos Secret MYSQL_DATABASE, MYSQL_ROOT_PASSWORD Env
kubectl apply -f 01-db-prod.secret.yaml
# Usamos ConfigMap para schema+data
kubectl apply -f 02-db-prod.configmap.yaml
# StafulSet: svc + mysql + initContainer + pvc + ConfigMap
kubectl apply -f 03-db-prod.sts.yaml
```

```
$  kubectl get svc,sts,pvc,pod,cm
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
service/db   ClusterIP   10.43.159.134   <none>        3306/TCP   17s

NAME                          READY   AGE
statefulset.apps/db-prod      0/1     17s


NAME                                      STATUS   VOLUME                                     CAPACITY   ACCESS MODES
persistentvolumeclaim/mysql-data-db-prod-0   Bound    pvc-e2b14aa3-e814-48c2-a060-f15958854fa6   1Gi        RWO

NAME            READY   STATUS    RESTARTS   AGE
pod/db-prod-0   1/1     Running   0          18s

NAME                         DATA   AGE
configmap/db-schema-data     2      20s
```

svc

sts

pvc

pod

p04- Kubernetes object: Service (cont)

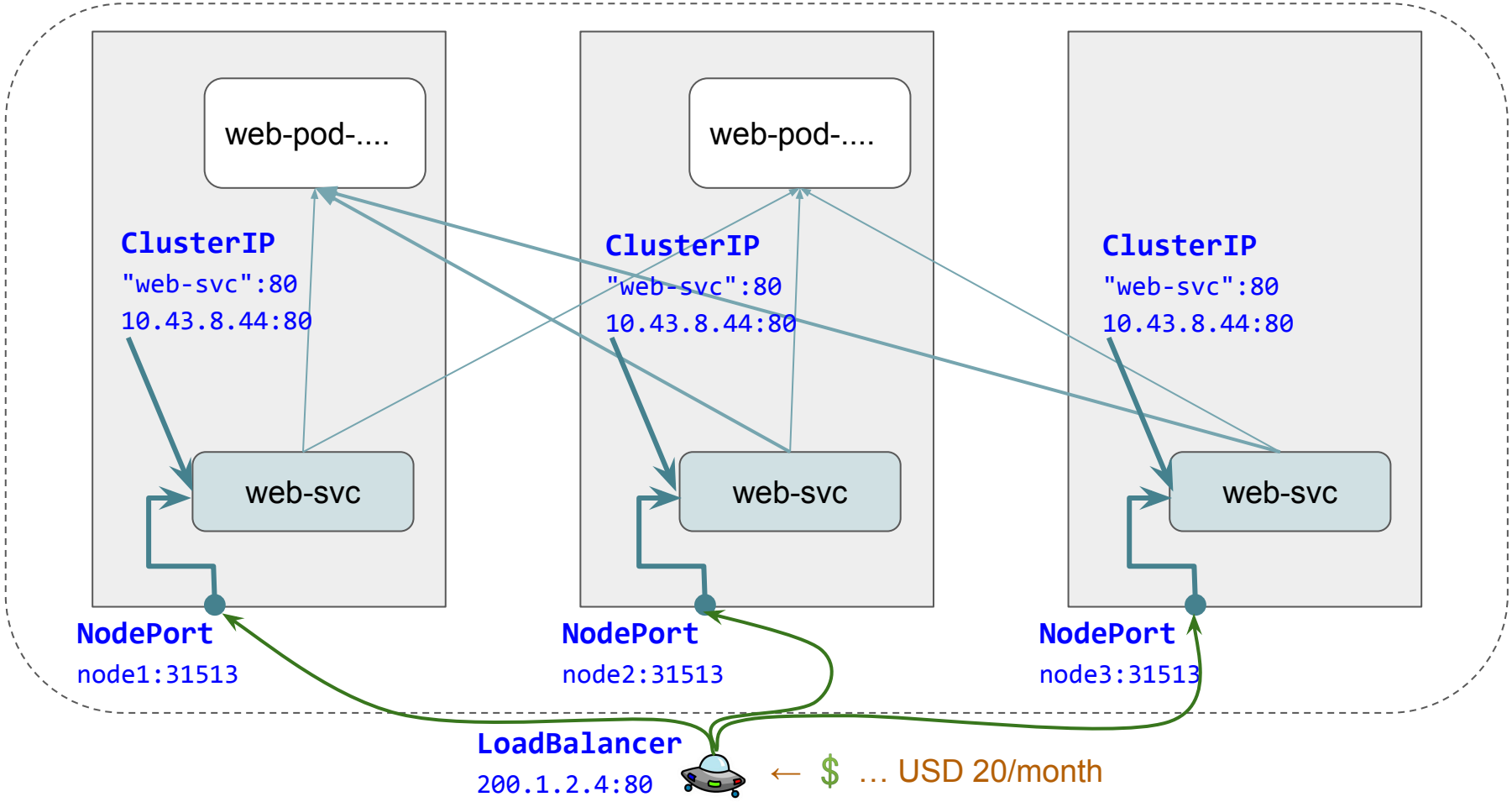
- Recordemos: service es la manera en que Kubernetes expone un set de pods ("endpoints") detrás de *IP:PORT* estables, en forma de "load-balancers" internos
- De lo que hicimos en ../c04-kube-deploy_svc_cm/p01 ->

```
$ kubectl get deploy,svc,ep,pod -o wide --show-labels
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE   CONTAINERS   IMAGES                SELECTOR   LABELS
deployment.apps/web   3/3     3            3           61s   nginx        bitnami/nginx:1.24.0  app=web    app=web

NAME          TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)   AGE   SELECTOR   LABELS
service/web   ClusterIP   10.43.8.44    <none>        80/TCP    30s   app=web    app=web

NAME           ENDPOINTS                                            AGE   LABELS
endpoints/web  10.42.4.46:8080,10.42.6.37:8080,10.42.9.30:8080      30s   app=web

NAME                       READY   STATUS    RESTARTS   AGE   IP           NODE           LABELS
pod/web-65c4fd5458-5jnbc   1/1     Running   0          40s   10.42.9.30   c01-worker-02  app=web,pod-template-hash=6
pod/web-65c4fd5458-hvwzl   1/1     Running   0          64s   10.42.4.46   c01-worker-01  app=web,pod-template-hash=6
pod/web-65c4fd5458-w944d   1/1     Running   0          40s   10.42.6.37   c01-worker-03  app=web,pod-template-hash=6
```

p04- Kubernetes object: `Service types`

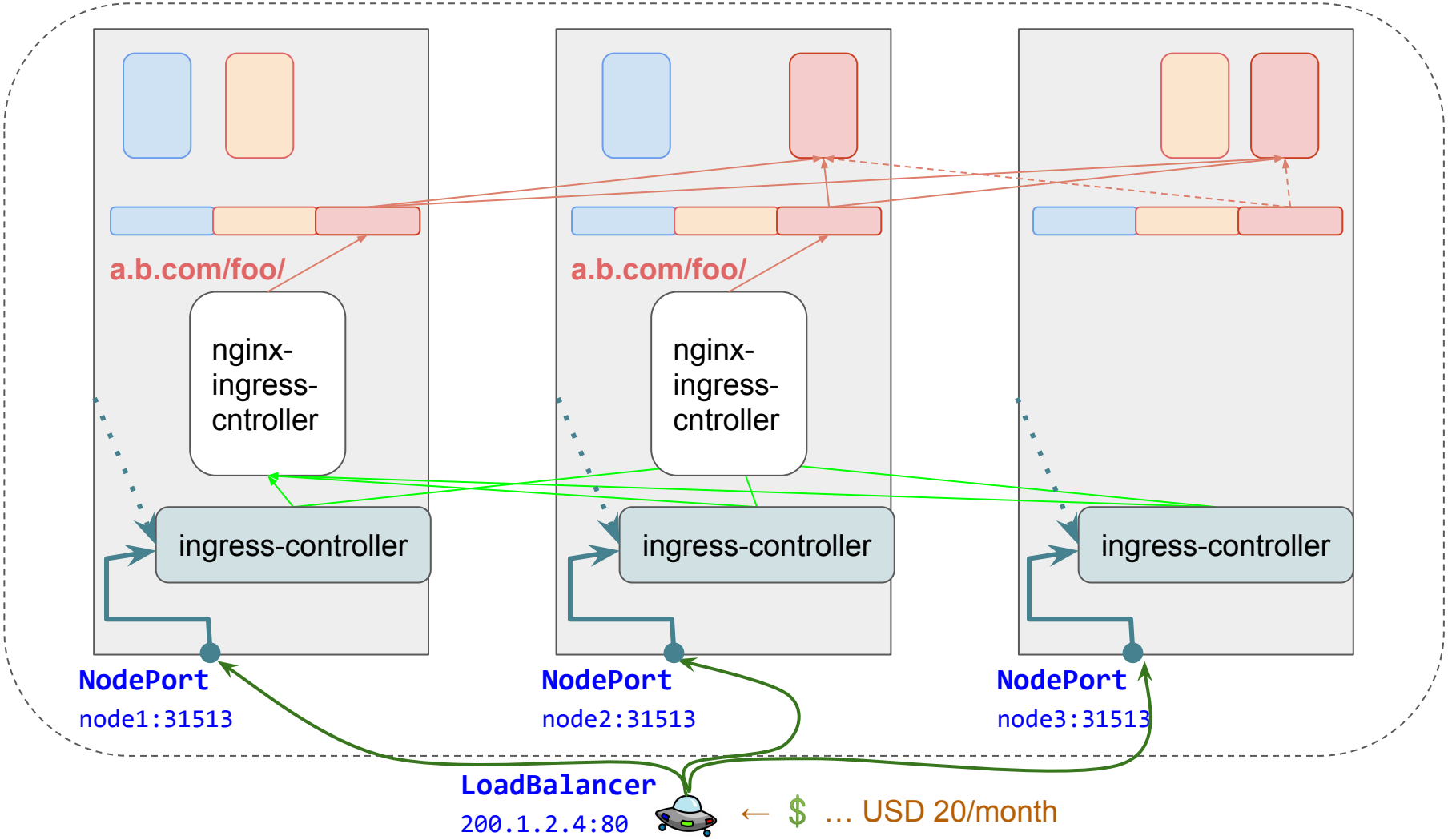
Hay varios tipos, de acuerdo a su *reachability*:

- **`ClusterIP`** (default): accesible desde **dentro** del cluster (nodos y pods)
- **`NodePort:`** ClusterIP + **ports dedicados en cada nodo worker**
- **`LoadBalancer:`** NodePort + LB provisto por el Cloud Provider



p04- Kubernetes object: `Ingress`

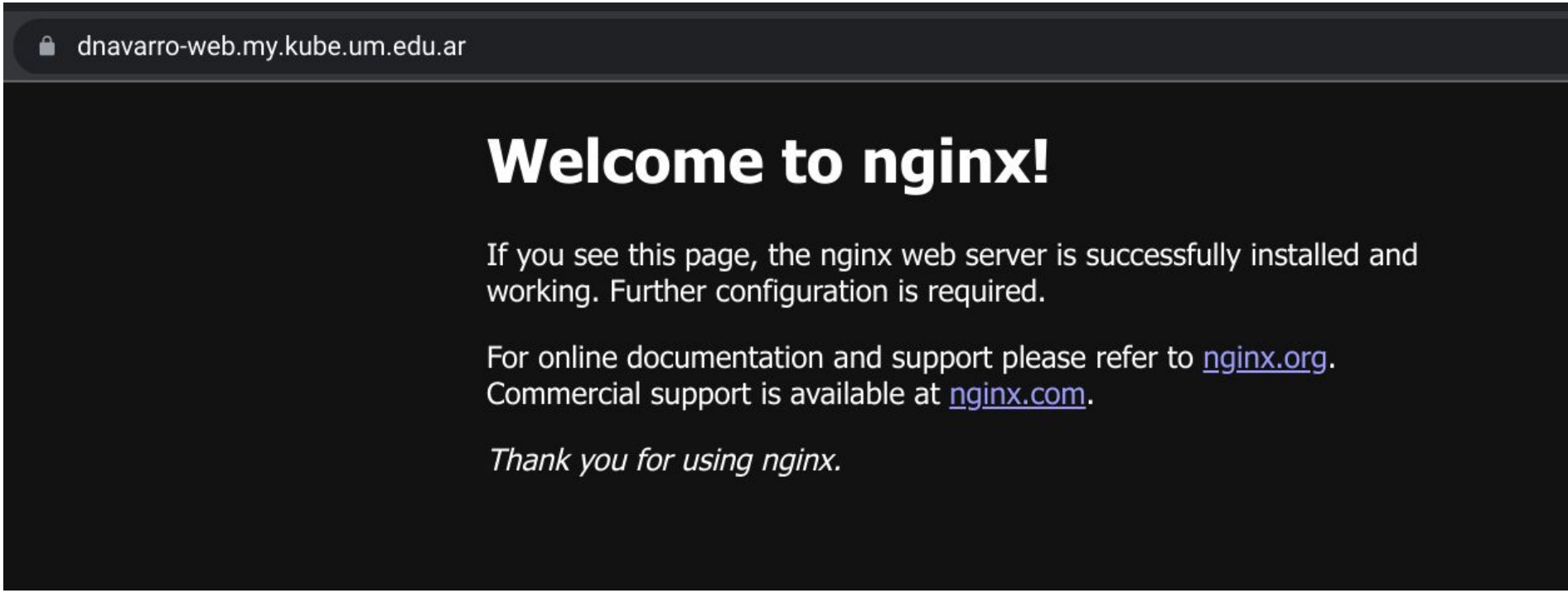
- **`Ingress`** es un recurso de **L7:**
  - http/https reverse proxy
  - basado en **virtualhost/path**



p04- Kubernetes object: `Ingress`

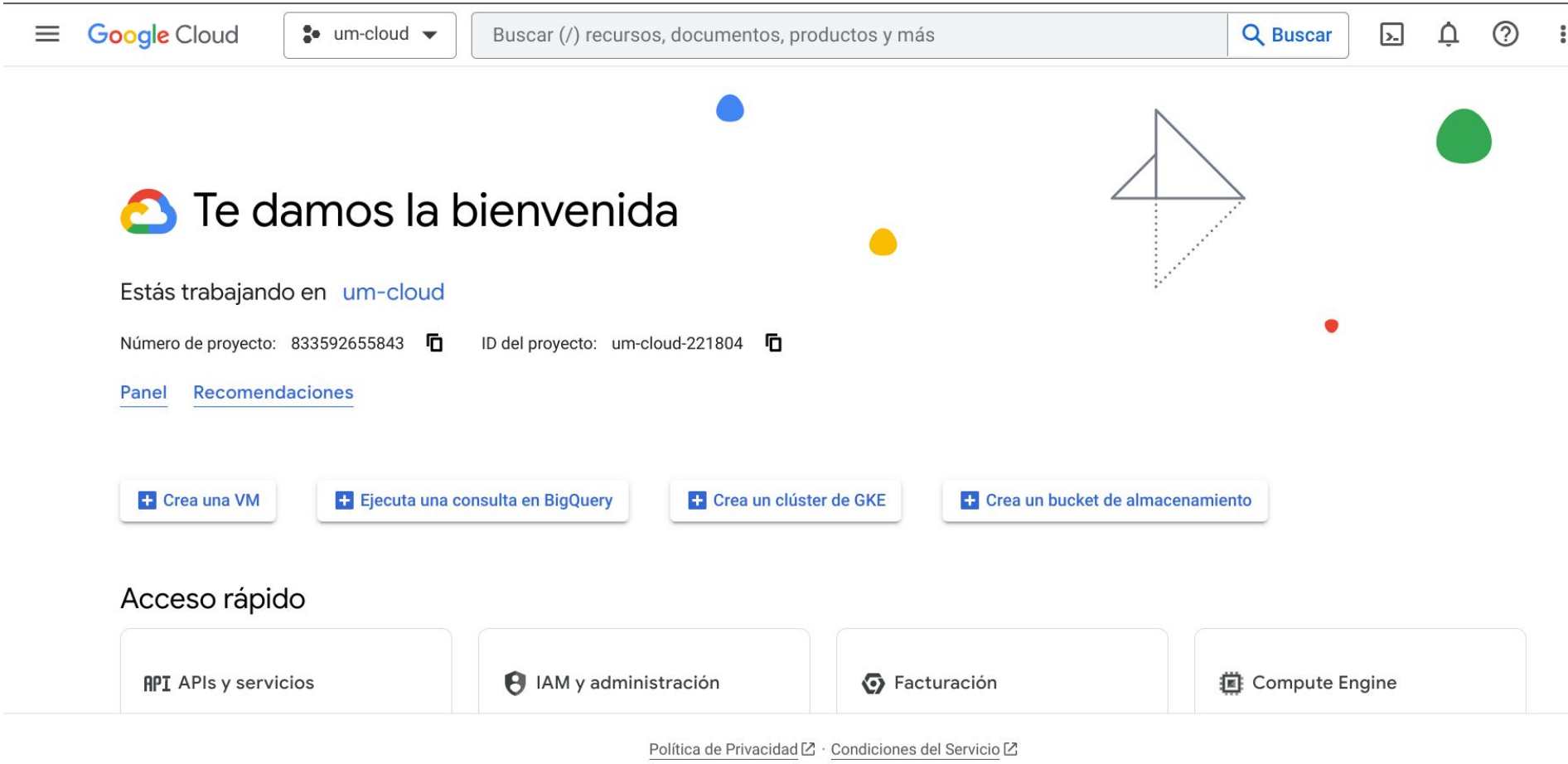
- **`Ingress`** es un recurso de **L7**:
  - http/https reverse proxy
  - basado en **virtualhost/path**

```
cd ../p04

vi web-ingress.yaml #modificar key host por url USER-web.my.kube.um.edu.ar
kubectl apply -f web-ingress.yaml
```

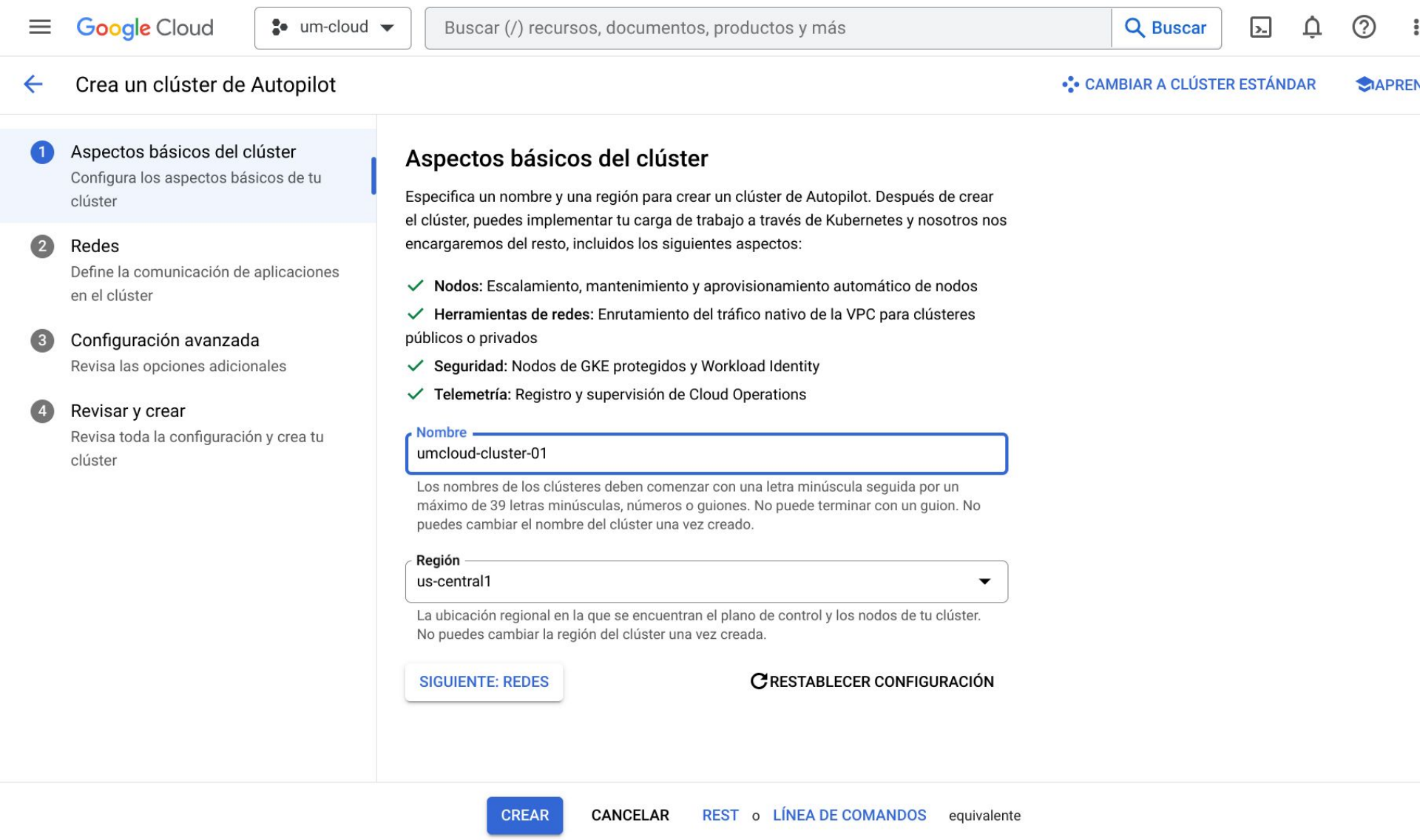


Google Cloud [Kubernetes Engine]

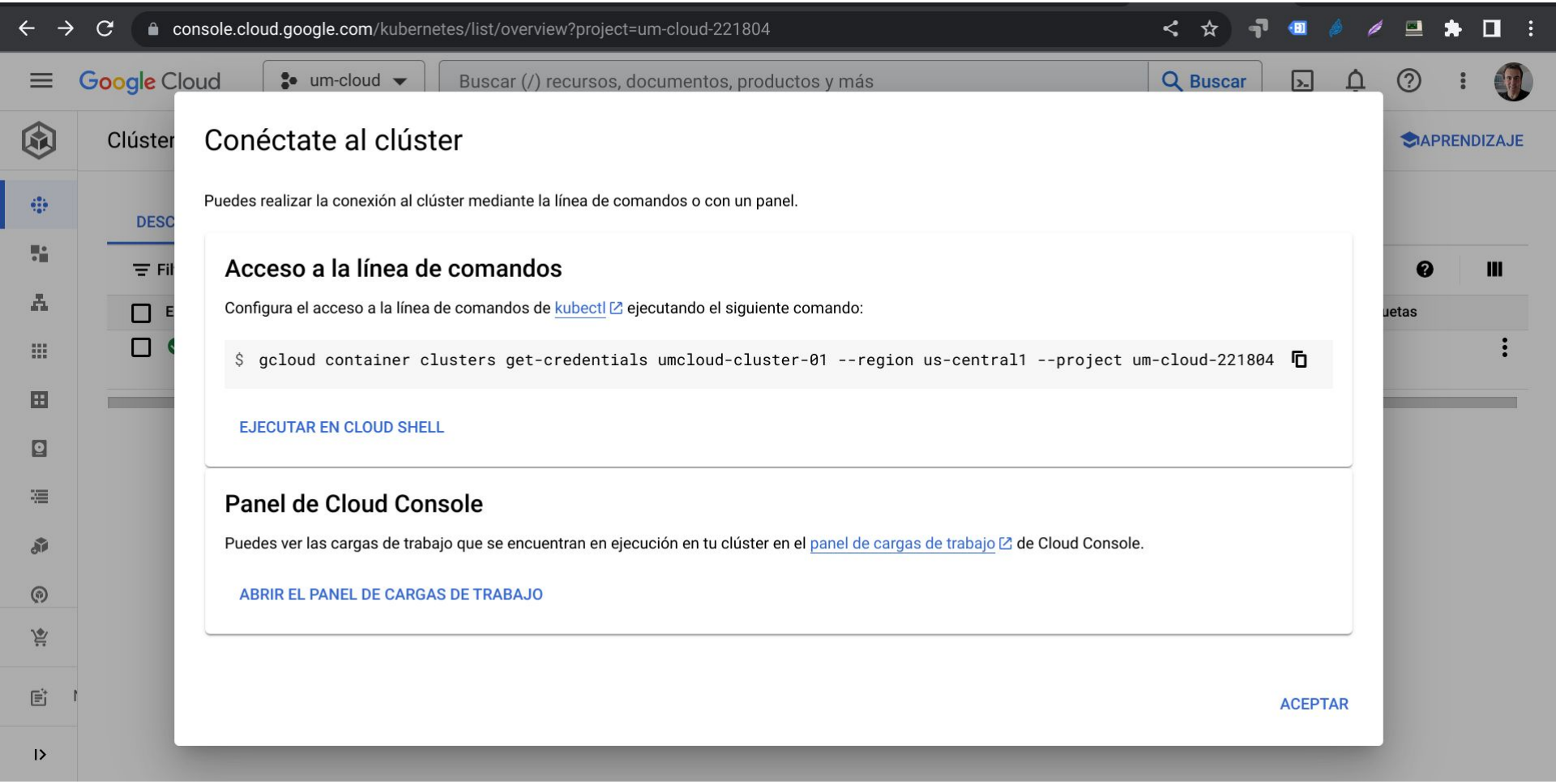
https://console.cloud.google.com



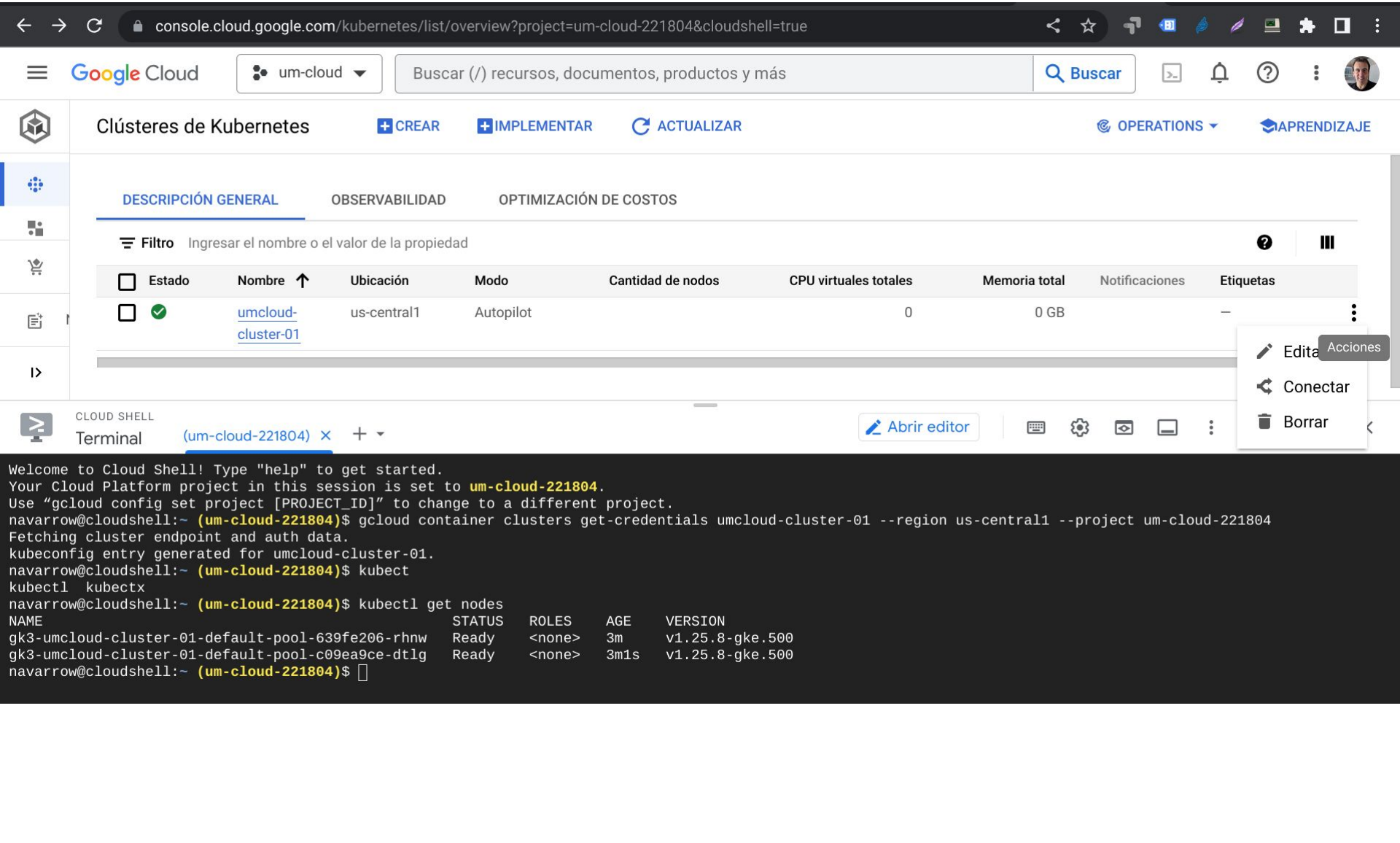
Google Cloud: Creamos el cluster



Google Cloud: Conectamos al cluster con cloud shell

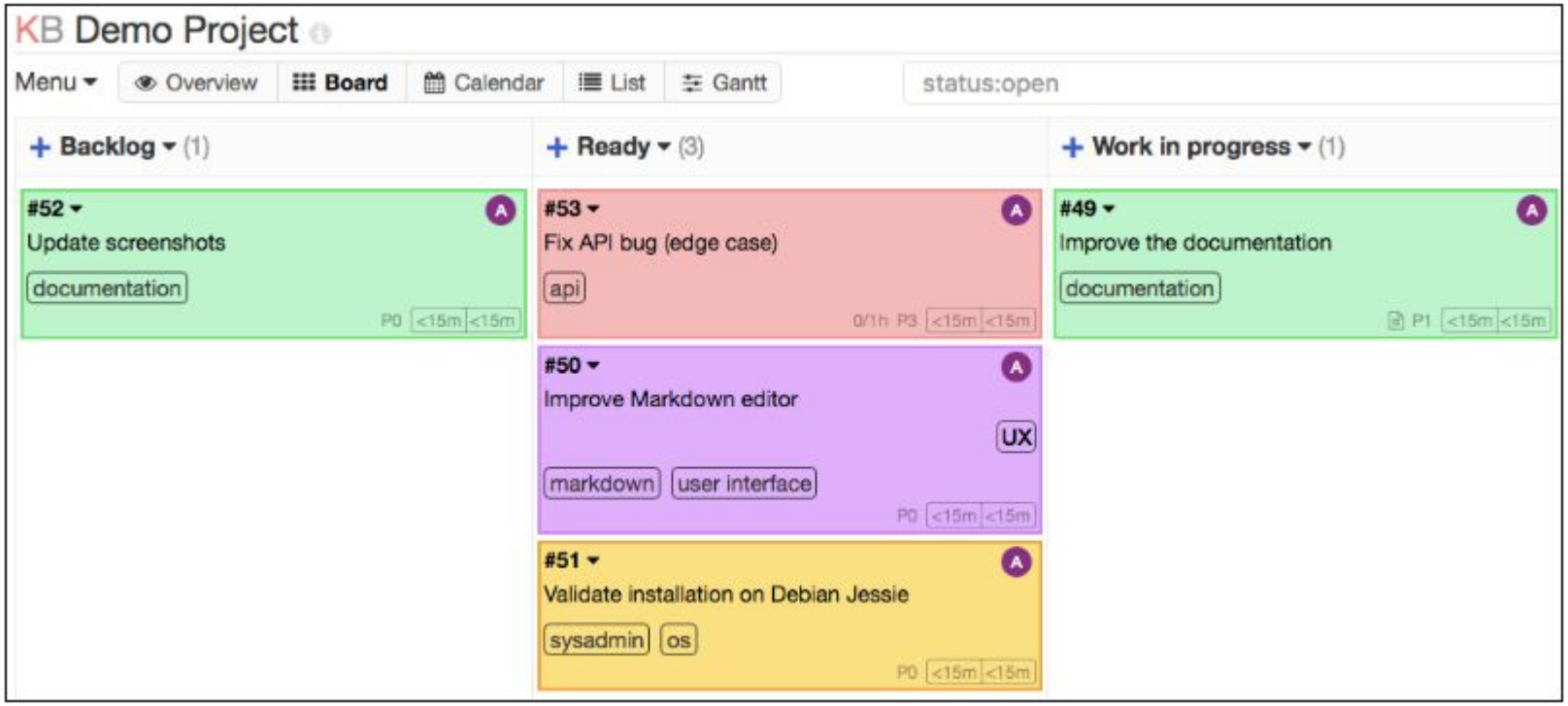


Google Cloud: Conectamos al cluster con cloud shell



Caso Práctico

- **Queremos desplegar nuestra aplicación para gestión ágil KanBoard en nuestro flamante cluster k8s :)**



- **Ref: https://kanboard.org/**
- **Ref: https://docs.kanboard.org/en/latest/admin_guide/docker.html**

Caso Práctico

De Ref: https://docs.kanboard.org/en/latest/admin_guide/docker.html

Running the Container

Basic Usage

```
docker pull kanboard/kanboard:v1.2.9
```

```
docker run -d --name kanboard -p 80:80 -t kanboard/kanboard:v1.2.9
```

Si hay container entonces … puede haber pod … entonces puede haber deploy ...

Stateless

```
cd ../p05-caso
```

Deploy

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kb-prod
  labels:
    run: kb-prod
spec:
  replicas: 1
  selector:
    matchLabels:
      run: kb-prod
  template:
    metadata:
      generateName: kb-prod
      labels:
        run: kb-prod
    spec:
      restartPolicy: Always
      containers:
        - name: kb-prod
          image: kanboard/kanboard:v1.2.9
          imagePullPolicy: Always
          ports:
            - name: kb-prod
              containerPort: 80
```

Service

```
apiVersion: v1
kind: Service
metadata:
  name: kb-prod
  labels:
    run: kb-prod
spec:
  ports:
  - port: 80
    name: kb-prod
    targetPort: 80
  selector:
    run: kb-prod
```

Ingress

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
    kubernetes.io/ingress.class: nginx
  name: kb-ing
spec:
  rules:  #modificar en host USER
  - host: USER-kb.my.kube.um.edu.ar
    http:
      paths:
      - backend:
          service:
            name: kb-prod
            port:
              number: 80
        path: /
        pathType: Prefix
```

Statefull ?

Sts …

Service

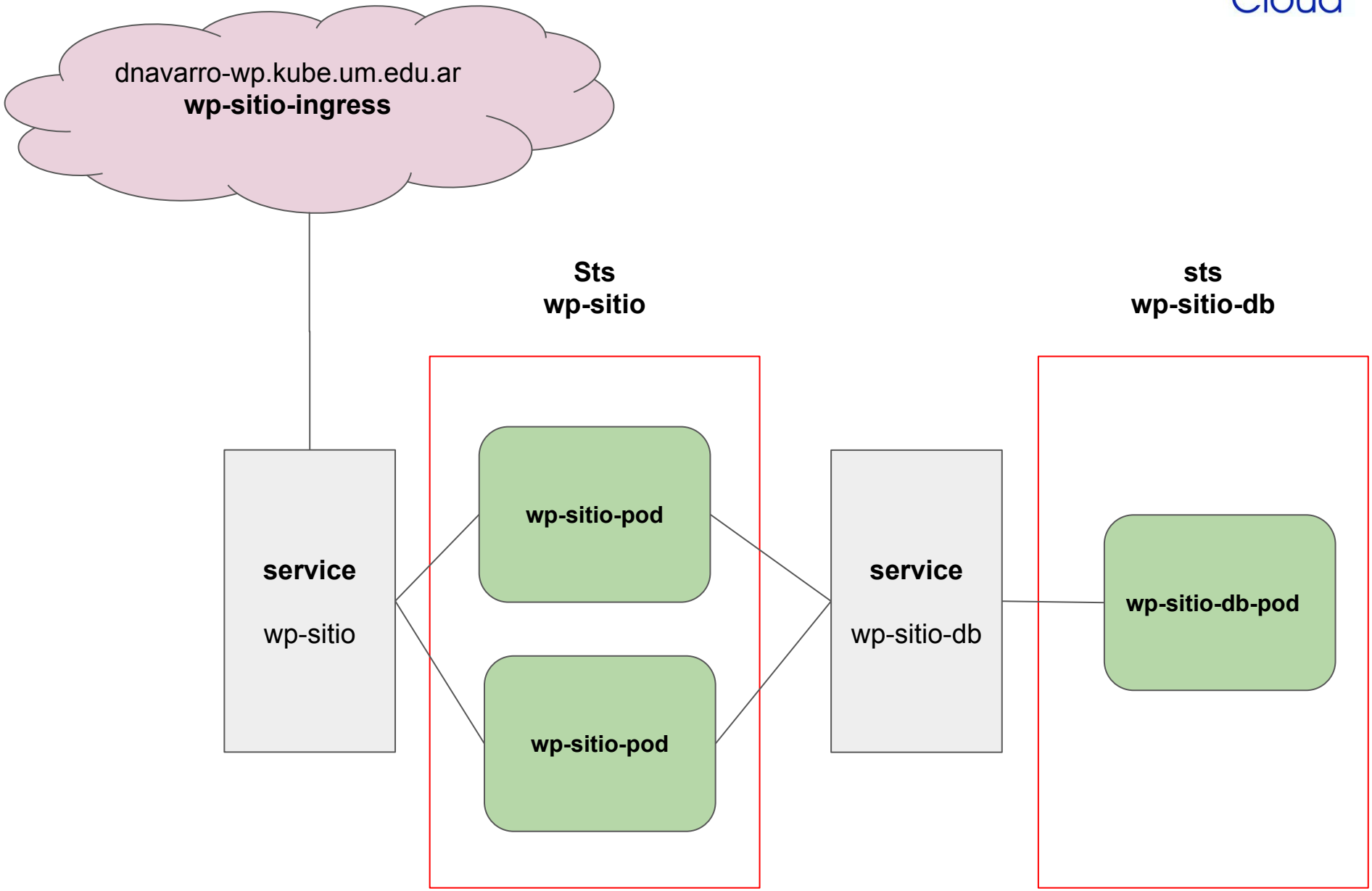
```
apiVersion: v1
kind: Service
metadata:
  name: kb-prod
  labels:
    run: kb-prod
spec:
  ports:
  - port: 80
    name: kb-prod
    targetPort: 80
  selector:
    run: kb-prod
```

Ingress

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
    kubernetes.io/ingress.class: nginx
  name: kb-ing
spec:
  rules:
  - host: USER-kb.my.kube.um.edu.ar
    http:
      paths:
      - backend:
          service:
            name: kb-prod
            port:
              number: 80
        path: /
        pathType: Prefix
```

WorPress: Caso Cloud Native

Wordpress sobre Kubernetes [stateless]



Wordpress sobre Kubernetes [stateless]

```
kubectl -n tele get deploy,svc,ingress
NAME                                      CLASS    HOSTS                              ADDRESS        PORTS   AGE
ingress.networking.k8s.io/wp-sitio-ing    <none>   navarrow.tele.cloud.um.edu.ar      172.16.16.74   80      141m

NAME                  TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)        AGE
service/wp-sitio      ClusterIP      10.43.234.31    <none>          80/TCP         141m
service/wp-sitio-db   ClusterIP      10.43.112.242   <none>          3306/TCP       141m

NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/wp-sitio       2/2     2            2           141m
deployment.apps/wp-sitio-db    1/1     1            1           141m

NAME                                READY   STATUS    RESTARTS   AGE
pod/wp-sitio-56d8b8b468-6fx6v       1/1     Running   0          141m
pod/wp-sitio-56d8b8b468-stdtv       1/1     Running   0          16s
pod/wp-sitio-db-69ddcd5dcd-pbbqz    1/1     Running   0          141m
```

Wordpress sobre Kubernetes [yaml's]

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
    kubernetes.io/ingress.class: nginx
    kubernetes.io/tls-acme: "false"
  name: wp-sitio-ing
spec:
  rules:
  - host: dnavarro-wp.my.kube.um.edu.ar
    http:
      paths:
      - backend:
          service:
            name: wp-sitio
            port:
              number: 80
        path: /
        pathType: Prefix
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: wp-sitio
  labels:
    run: wp-sitio
spec:
  ports:
  - port: 80
    name: wp-sitio
  selector:
    run: wp-sitio
```

```yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: wp-sitio
  labels:
    run: wp-sitio
spec:
  selector:
    matchLabels:
      run: wp-sitio
  serviceName: "wp-sitio"
  template:
    metadata:
      generateName: wp-sitio
      labels:
        run: wp-sitio
    spec:
      containers:
        - name: wp-sitio
          image: wordpress:php7.4-apache
          imagePullPolicy: IfNotPresent
          env:
            - name: WORDPRESS_DB_HOST
              valueFrom:
                secretKeyRef:
                  name: wp-sitio-sec
                  key: WORDPRESS_DB_HOST
            - name: WORDPRESS_DB_USER
              valueFrom:
                secretKeyRef:
                  name: wp-sitio-sec
                  key: WORDPRESS_DB_USER
            - name: WORDPRESS_DB_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: wp-sitio-sec
                  key: WORDPRESS_DB_PASSWORD
            - name: WORDPRESS_DB_NAME
              valueFrom:
                secretKeyRef:
                  name: wp-sitio-sec
                  key: WORDPRESS_DB_NAME

          ports:
            - name: wp-sitio
              containerPort: 80
          volumeMounts:
            - mountPath: "/var/www/html"
              name: wp-sitio-data
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: wp-sitio-db
  labels:
    run: wp-sitio-db
spec:
  ports:
  - port: 3306
    name: wp-sitio-db
  selector:
    run: wp-sitio-db
```

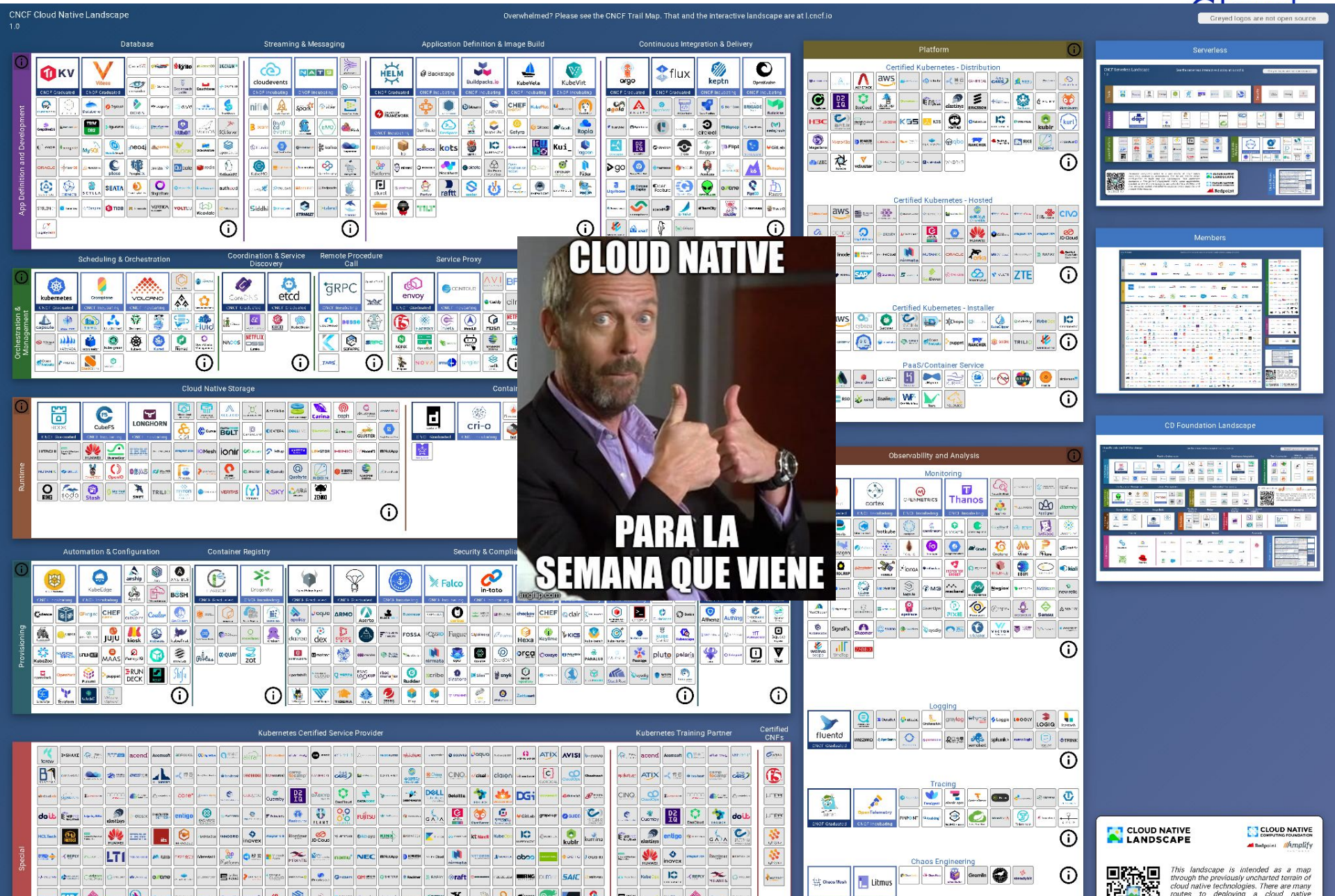
```yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: wp-sitio-db
  labels:
    run: wp-sitio-db
spec:
  selector:
    matchLabels:
      run: wp-sitio-db
  serviceName: "wp-sitio-db"
  template:
    metadata:
      generateName: wp-sitio-db
      labels:
        run: wp-sitio-db
    spec:
      initContainers:
      - name: "remove-lost-found"
        image: "busybox:1.25.0"
        imagePullPolicy: "IfNotPresent"
        command:  ["rm", "-fr",
"/var/lib/mysql/lost+found"]
      volumeMounts:
      - name: mysql-data
        mountPath: /var/lib/mysql
    containers:
      - name: wp-sitio-db
        image: mysql:5.7
        imagePullPolicy: IfNotPresent
        env:
          - name: MYSQL_DATABASE
            valueFrom:
              secretKeyRef:
                name: wp-sitio-db-sec
                key: MYSQL_DATABASE
          - name: MYSQL_ROOT_PASSWORD
            valueFrom:
              secretKeyRef:
                name: wp-sitio-db-sec
                key: MYSQL_ROOT_PASSWORD
        ports:
          - name: wp-sitio-db
            containerPort: 3306
        volumeMounts:
          - mountPath: "/var/lib/mysql"
```

Wordpress sobre Kubernetes [yaml's]

```
apiVersion: v1
kind: Secret
data:
  WORDPRESS_DB_HOST: d3Atc2l0aW8tZGI=
  WORDPRESS_DB_NAME: d3Atc2l0aW8=
  WORDPRESS_DB_PASSWORD: YjBkOWI2MDljOWE5ZDQ2YmEyNzNkYjA4MGU2ZTAwNmY=
  WORDPRESS_DB_USER: cm9vdA==
metadata:
  creationTimestamp: null
  name: wp-sitio-sec


apiVersion: v1
kind: Secret
data:
  MYSQL_DATABASE: d3Atc2l0aW8=
  MYSQL_ROOT_PASSWORD: YjBkOWI2MDljOWE5ZDQ2YmEyNzNkYjA4MGU2ZTAwNmY=
metadata:
  creationTimestamp: null
  name: wp-sitio-db-sec
```

Cloud Native
Como puedo seguir …

pero … cuidado con los Cloud Native ninjas !



Education

Recursos para aprender containers / Kubernetes

EdX: para iniciar

- https://www.edx.org/course/introduction-to-cloud-infrastructure-technologies
- https://www.edx.org/course/introduction-to-kubernetes

Katacoda: hands-on!

- https://www.katacoda.com/courses/kubernetes/

Cómo sigo ? / qué me gusta

- https://roadmap.sh/