

# Portafolio PGC

Alumno: Javier Meliá Sevilla

## Tema 1:

### 1.1 Test de linpack enjaulado

Aquí se puede observar en el terminal original la captura tomada con el comando TOP como los procesos están compitiendo

root@ubuntu: /sys/fs/cgroup

```
top - 16:22:57 up 62 days, 5:22, 6 users, load average: 3,07, 1,29, 0,71
Tasks: 156 total, 4 running, 152 sleeping, 0 stopped, 0 zombie
%Cpu(s): 83,4 us, 16,4 sy, 0,0 ni, 0,2 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
MiB Mem : 5935,2 total, 2027,1 free, 320,4 used, 3587,7 buff/cache
MiB Swap: 1870,0 total, 1870,0 free, 0,0 used. 5327,6 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
127439	root	20	0	151172	2012	1824	R	299,0	0,0	4:33.63	mtlinpack
127442	root	20	0	3708	2380	1936	R	99,7	0,0	1:19.77	mtlinpack
1	root	20	0	167740	13288	8268	S	0,0	0,2	2:30.46	systemd

```
|-- bin
| |-- bash
| `-- mtlinpack
|-- lib
| `-- x86_64-linux-gnu
|     |-- libc.so.6
|     |-- libdl.so.2
|     |-- libgomp.so.1
|     |-- libm.so.6
|     |-- libpthread.so.0
|     `-- libtinfo.so.5
|-- lib64
| `-- ld-linux-x86-64.so.2
`-- mlin.txt
```

### 1.2 Proceso enjaulado a partir de Docker

Aquí tenemos las siguientes capturas para ver todo el proceso y al final se responde la pregunta.

```
alumno@ubuntu:~$ docker exec -it mi_alp ash
/ # ps -elf
PID   USER     TIME   COMMAND
    1   root      0:00   ash
   13   root      0:00   ash
   20   root      0:00   ps -elf
/ # id
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel),11(fl
oppy),20(dialout),26(tape),27(video)
/ #
```

```
alumno@ubuntu:~/mijaulaalpine$ ls
alpineimagen  dev  home  media  opt  root  sbin  sys  usr
bin           etc  lib   mnt    proc  run  srv  tmp  var
alumno@ubuntu:~/mijaulaalpine$ unshare --mount --uts --ipc --net --pid --fork --user --map-ro
ot-user /usr/sbin/chroot . /bin/ash
/ # ps -elf
PID   USER     TIME   COMMAND
    1   root      0:00   ash
   13   root      0:00   ash
   20   root      0:00   ps -elf
/ # id
uid=0(root) gid=0(root) groups=65534(nobody),65534(nobody),0(root)
/ #
```

```

alumno@ubuntu:~$ ps -elf |grep ash
0 S alumno      1739      1738    0  80    0 -  2711 do_wai 16:31 pts/0    00:00:01 -bash
4 S root         2635      2615    0  80    0 -   430 -      17:44 pts/0    00:00:00 ash
4 S alumno      2894      1739    0  80    0 -  1442 do_wai 17:56 pts/0    00:00:00 unshare --moun
t --uts --ipc --net --pid --fork --user --map-root-user /usr/sbin/chroot . /bin/ash
4 S alumno      2895      2894    0  80    0 -   430 do_pol 17:56 pts/0    00:00:00 /bin/ash
0 S alumno      2997      2996    0  80    0 -  2596 do_wai 17:57 pts/1    00:00:00 -bash
0 S alumno      3084      2997    0  80    0 - 334750 futex_ 17:58 pts/1    00:00:00 docker exec -i
t mi_alp ash
4 S root         3105      2615    0  80    0 -   430 -      17:58 pts/1    00:00:00 ash
0 S alumno      3173      3172    0  80    0 -  2183 do_wai 17:59 pts/2    00:00:00 -bash
0 S alumno      3184      3173    0  80    0 -  1619 pipe_r 18:00 pts/2    00:00:00 grep --color=a
uto ash
alumno@ubuntu:~$

```

Fuera del contenedor podemos ver que tenemos 3 ash 2 de root uno por el run y otro por el exec y 1 ash por alumno del unshare aunque cuando miras dentro del contenedor donde se he ejecutado el unshare te pone que su id es root pero solo dentro del contenedor en verdad para fuera sigue siendo alumno.

### 1.3 Redes host y bridge

En la siguiente captura podemos comprobar con el comando ifconfig los adaptadores e IPs asignadas.

```

alumno@ubuntu: ~/ejercicio3
alumno@ubuntu:~/ejercicio3$ docker exec -it ej3 ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.17.0.2 netmask 255.255.0.0 broadcast 172.17.255.255
    ether 02:42:ac:11:00:02 txqueuelen 0 (Ethernet)
    RX packets 4214 bytes 26731781 (26.7 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 2312 bytes 179304 (179.3 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

alumno@ubuntu:~/ejercicio3$

```

En la siguiente captura podemos ver la comprobación que es posible acceder con wget a la ip del host.

```

alumno@ubuntu:~/ejercicio3
alumno@ubuntu:~/ejercicio3$ docker exec -it ej3 /bin/bash -c 'echo "<html><body><h1>Javier Melia</h1></body></html>" > /var/www/html/index.html'
alumno@ubuntu:~/ejercicio3$ wget 172.17.0.2
--2023-03-13 09:51:33-- http://172.17.0.2/
Connecting to 172.17.0.2:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 48 [text/html]
Saving to: 'index.html.2'

index.html.2               100%[=====] 48 --KB/s in 0s

2023-03-13 09:51:33 (3.18 MB/s) - 'index.html.2' saved [48/48]

alumno@ubuntu:~/ejercicio3$ cat index.html
<html><body><h1>Javier Melia</h1></body></html>
alumno@ubuntu:~/ejercicio3$

```

### 1.4 OPCIONAL Red privada

No hecho

## 1.5 Dockerfile

La primera captura hace referencia al Dockerfile para crear la imagen de jupyter notebook con las condiciones del ejercicio.

```
alumno@ubuntu:~/ejercicio5$ cat Dockerfile
FROM python:3.7-alpine
RUN apk add --no-cache gcc musl-dev linux-headers libffi-dev
RUN pip install notebook
RUN adduser -D usuario
EXPOSE 8888
USER usuario
WORKDIR /HOME/usuario
CMD ["jupyter", "notebook", "--ip=0.0.0.0"]

alumno@ubuntu:~/ejercicio5$
```

Esta segunda captura es de la pantalla con el notebook en marcha donde se ve la dirección



## 1.6. Clúster MPI con Docker Compose

Para estas evidencias primero se muestra el Docker-compose.yaml

```
alumno@ubuntu:~/ejercicio6$ cat docker-compose.yaml
version: "3.3"
services:
  head:
    image: "iblanque/openmpi:latest"
    ports:
      - "22022:22"
    expose:
      - 22
  node1:
    image: "iblanque/openmpi:latest"
    expose:
      - 22
alumno@ubuntu:~/ejercicio6$
```

En las siguientes capturas se ve la ejecución antes del escalado:

```

alumno@ubuntu:~$ docker-compose up -d
Creating network "alumno_default" with the default driver
Creating alumno_head_1 ... done
Creating alumno_node1_1 ... done
alumno@ubuntu:~$ ssh -p 22022 usuario@localhost
The authenticity of host '[localhost]:22022 ([127.0.0.1]:22022)' can't be established.
ED25519 key fingerprint is SHA256:slLJ2h9qY3fXo2afsgBLpeakUA6ZtMz6NRaeNJBNIwA.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[localhost]:22022' (ED25519) to the list of known hosts.
usuario@localhost's password:
usuario@85caa0d25737:~$ clear
usuario@85caa0d25737:~$ ls hola_mpi
hola_mpi
usuario@85caa0d25737:~$ vi hostfile
usuario@85caa0d25737:~$ vi hostfile
usuario@85caa0d25737:~$ vi hostfile
usuario@85caa0d25737:~$ mpirun -np 2 -hostfile hostfile ho
hola_mpi      hola_mpi.c  hostfile
usuario@85caa0d25737:~$ mpirun -np 2 -hostfile hostfile hola_mpi
Warning: Permanently added 'alumno_node1_1,172.21.0.2' (ECDSA) to the list of known hosts.
Hola a todos!, Soy el proceso 0 desde el host 0beb24b7e499
Hola a todos!, Soy el proceso 1 desde el host 0beb24b7e499
usuario@85caa0d25737:~$ █

```

Y en las siguientes después de escalar:

```

alumno@ubuntu:~$ docker-compose scale head=1 node1=2
WARNING: The scale command is deprecated. Use the up command with the --scale flag instead.
Desired container number already achieved
Starting alumno_node1_1 ... done
Creating alumno_node1_2 ... done
alumno@ubuntu:~$ ssh -p 22022 usuario@localhost
usuario@localhost's password:
Last login: Tue Feb 21 18:03:38 2023 from 172.21.0.1

usuario@85caa0d25737:~$ vi hostfile
usuario@85caa0d25737:~$ mpirun -np 8 -hostfile hostfile hola_mpi
Warning: Permanently added 'alumno_node1_2,172.21.0.4' (ECDSA) to the list of known hosts.
Hola a todos!, Soy el proceso 1 desde el host 0beb24b7e499
Hola a todos!, Soy el proceso 3 desde el host 0beb24b7e499
Hola a todos!, Soy el proceso 0 desde el host 0beb24b7e499
Hola a todos!, Soy el proceso 2 desde el host 0beb24b7e499
Hola a todos!, Soy el proceso 6 desde el host a9a9b81140b1
Hola a todos!, Soy el proceso 5 desde el host a9a9b81140b1
Hola a todos!, Soy el proceso 7 desde el host a9a9b81140b1
Hola a todos!, Soy el proceso 4 desde el host a9a9b81140b1
usuario@85caa0d25737:~$ █

```



## Tema 2:

### 2.1. Ejercicio Namespaces

Las siguientes capturas muestran todos los yaml de los namespaces de los pods y las cuotas.

```
alumno@ubuntu:~/kubernetes/ej1$ cat namespace-alta.yaml
apiVersion: v1
kind: Namespace
metadata:
  name: alta
  labels:
    name: alta
alumno@ubuntu:~/kubernetes/ej1$ cat namespace-baja.yaml
apiVersion: v1
kind: Namespace
metadata:
  name: baja
  labels:
    name: baja
```

```
alumno@ubuntu:~/kubernetes/ej1$ cat pods.yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod1
spec:
  containers:
  - name: pod1
    image: ubuntu
    command: ["/bin/sh"]
    args: ["-c", "while true; do echo hello; sleep 10;done"]
---
apiVersion: v1
kind: Pod
metadata:
  name: pod2
spec:
  containers:
  - name: pod2
    image: ubuntu
    command: ["/bin/sh"]
    args: ["-c", "while true; do echo hello; sleep 10;done"]
```

```

alumno@ubuntu:~/kubernetes/ej1$ cat rquota_5.yaml
apiVersion: v1
kind: ResourceQuota
metadata:
  name: pods5
  namespace: alta
spec:
  hard:
    pods: "5"

---

apiVersion: v1
kind: ResourceQuota
metadata:
  name: pods1
  namespace: baja
spec:
  hard:
    pods: "1"

```

Las siguientes capturas son de las cuotas antes y después de la creación de los objetos, antes:

```

alumno@ubuntu: ~/kubernetes/ej1

```

```

alumno@ubuntu:~/kubernetes/ej1$ kubectl describe quota -n alta
Name:      pods5
Namespace: alta
Resource   Used  Hard
-----
pods       0    5
alumno@ubuntu:~/kubernetes/ej1$ kubectl describe quota -n baja
Name:      pods1
Namespace: baja
Resource   Used  Hard
-----
pods       0    1

```

Después:

```

alumno@ubuntu:~/kubernetes/ej1$ kubectl apply -f pods.yaml -n alta
pod/pod1 created
pod/pod2 created
alumno@ubuntu:~/kubernetes/ej1$ kubectl apply -f pods.yaml -n alta
pod/pod1 unchanged
pod/pod2 unchanged
alumno@ubuntu:~/kubernetes/ej1$ kubectl describe quota -n alta
Name:      pods5
Namespace: alta
Resource   Used  Hard
-----
pods       2    5
alumno@ubuntu:~/kubernetes/ej1$ kubectl apply -f pods.yaml -n baja
pod/pod1 created
Error from server (Forbidden): error when creating "pods.yaml": pods "pod2" is forbidden: exceeded quota: pods1, requested: pods=1, used: pods=1, limited: pods=1
alumno@ubuntu:~/kubernetes/ej1$ kubectl describe quota -n baja
Name:      pods1
Namespace: baja
Resource   Used  Hard
-----
pods       1    1
alumno@ubuntu:~/kubernetes/ej1$

```

Y por último el listado de los pods en cada name space

```
alumno@ubuntu: ~/kubernetes/ej1
alumno@ubuntu:~/kubernetes/ej1$ kubectl get pods -o wide -n alta
NAME      READY   STATUS    RESTARTS   AGE   IP            NODE       NOMINATED NODE   READINESS GATES
pod1      1/1     Running   0           2m38s  172.17.0.6    minikube   <none>           <none>
pod2      1/1     Running   0           2m38s  172.17.0.5    minikube   <none>           <none>
alumno@ubuntu:~/kubernetes/ej1$ kubectl get pods -o wide -n baja
NAME      READY   STATUS    RESTARTS   AGE   IP            NODE       NOMINATED NODE   READINESS GATES
pod1      1/1     Running   0           2m4s   172.17.0.7    minikube   <none>           <none>
```

## 2.2. Ejercicio Ingress

Las siguientes capturas son todos los ficheros YAML de los objetos creados, los pods:

```
alumno@ubuntu:~/kubernetes/ej2$ cat apache-pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: apache
spec:
  containers:
  - name: apache
    image: httpd:latest
    ports:
    - containerPort: 80
alumno@ubuntu:~/kubernetes/ej2$ cat nginx-pod.yaml
kind: Pod
apiVersion: v1
metadata:
  name: nginx
spec:
  restartPolicy: Never
  volumes:
  - name: nfs-pv
    persistentVolumeClaim:
      claimName: nfs-pvc
  containers:
  - name: nginx
    image: nginx
    ports:
    - containerPort: 80
    volumeMounts:
    - mountPath: "/my-nfs"
      name: nfs-pv
alumno@ubuntu:~/kubernetes/ej2$
```

El persitentVolume y el persistent Volume Claim:



```

alumno@ubuntu:~/kubernetes/ej2$ cat pv.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfs-pv
spec:
  persistentVolumeReclaimPolicy: Recycle
  storageClassName: nfs-subdir
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteMany
  nfs:
    server: host-ip
    path: "/mnt/nfs_share/"
    readOnly: false
alumno@ubuntu:~/kubernetes/ej2$ cat pvc.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: nfs-pvc
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: "nfs-subdir"
  resources:
    requests:
      storage: 100Mi
alumno@ubuntu:~/kubernetes/ej2$ █

```

Y por último el service y el ingress:

```

alumno@ubuntu:~/kubernetes/ej2$ cat service.yaml
kind: Service
apiVersion: v1
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 9376
      targetPort: 80
alumno@ubuntu:~/kubernetes/ej2$ cat ingress.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginx-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
    - host: jamese-pgc
      http:
        paths:
          - path: /web
            pathType: Prefix
            backend:
              service:
                name: nginx-service
                port:
                  number: 80
alumno@ubuntu:~/kubernetes/ej2$ █

```

Y esta ultima captura es de los resultados de la ejecución del curl para acceder a los servicios publicados en el ingress:

```
*** System restart required ***
Last login: Mon Mar 13 08:29:17 2023 from 10.0.0.73
alumno@ubuntu:~$ minikube ssh
Last login: Tue Mar  7 17:08:26 2023 from 192.168.49.1
docker@minikube:~$ curl 172.17.0.7/my_index.html
<html>Javier Melia</html>
docker@minikube:~$ curl 172.17.0.6/my_index.html
<html>Javier Melia</html>
docker@minikube:~$
```

### 2.3. Ejercicio clúster iPython

Para la primera parte del ejercicio el manual se necesita la referencia de las imágenes de los contenedores utilizados:

```
alumno@ubuntu:~/kubernetes/ej3/buena$ docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
b6db694dfdc2   meliatus90/ipython:alumno          "bash"                  7 minutes ago Up 7 minutes
a09f505a5f0a   meliatus90/ipython:alumno          "bash"                  8 minutes ago Up 7 minutes
19a3b660eb12   meliatus90/ipython:alumno          "bash"                  11 minutes ago Up 10 minutes
f79ecfd76d73   ubuntu:22.04                       "/bin/bash"             20 minutes ago Up 20 minutes
c7daa6dcd343   gcr.io/k8s-minikube/kicbase:v0.0.36 "/usr/local/bin/entr..." 5 days ago    Up 5 days     127.0.0.1:49159->8443/tcp, 127.0.0.1:49158->32443/tcp
alumno@ubuntu:~/kubernetes/ej3/buena$
```

Y la ejecución por pantalla:

```
alumno@ubuntu:~/kubernetes/ej3/buena$ docker exec -it ipfe python2
Python 2.7.18 (default, Jul  1 2022, 10:30:50)
[GCC 11.2.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import ipyparallel as ipp
>>> c=ipp.Client('/root/.ipython/profile_default/security/ipcontroller-client.json')
>>> c.ids
[0, 1]
>>> def misquare(x):
...     return x*x
...     File "<stdin>", line 2
...         return x*x
...         ^
IndentationError: expected an indented block
>>> import ipyparallel as ipp
>>> c=ipp.Client('/root/.ipython/profile_default/security/ipcontroller-client.json')
>>> c.ids
[0, 1]
>>> def misquare(x):
...     return x*x
...
>>> misquare(3)
9
>>> l=c[:].map(misquare,range(4))
>>> l[1]
1
>>> l[3]
9
>>>
```

Para la segunda parte que se ejecuta en kubernetes, se necesita los yaml que son los siguientes:

```

alumno@ubuntu:~/kubernetes/ej3/buena$ cat pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: nombrepod
spec:
  containers:
  - name: ipfe
    image: meliatus80/ipython:alumno
    command: [ "usr/bin/tail" ]
    args: [ '-f', '/dev/null' ]
alumno@ubuntu:~/kubernetes/ej3/buena$ cat deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ipython-deployment
spec:
  selector:
    matchLabels:
      app: ipython-aplicacion
  replicas: 2
  template:
    metadata:
      labels:
        app: ipython-aplicacion
    spec:
      containers:
      - name: nombrepod
        image: meliatus80/ipython:alumno
        command: [ /bin/sh ]
        args: [-c, ipengine --file=/root/.ipython/profile_default/security/ipcontroller-engine.json]
        volumeMounts:
        - mountPath: /root/.ipython/profile_default/security/
          name: ipythonconf
      volumes:
      - name: ipythonconf
        configMap:
          name: ipythonconf
          items:
          - key: ipcontroller-engine.json
            path: ipcontroller-engine.json
alumno@ubuntu:~/kubernetes/ej3/buena$

```

También la copia de los comandos de get pods y get deployment:

nombrepod	1/1	Running	0	6d1h	172.17.0.6	minikube	<none>	<none>
alumno@ubuntu:~/kubernetes/ej3/buena\$ kubectl get deployment								
NAME	READY	UP-TO-DATE	AVAILABLE	AGE				
ipython-deployment	0/2	2	0	6d				

Y la ejecución:

```

alumno@ubuntu:~/kubernetes/ej3/buena$ kubectl exec -it ipfe python2
Python 2.7.18 (default, Jul 1 2022, 10:30:50)
[GCC 11.2.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import ipyparallel as ipp
>>> c=ipp.Client('/root/.ipython/profile_default/security/ipcontroller-client.json')
>>> c.ids
[0, 1]
>>> def misquare(x):
...     return x*x
      File "<stdin>", line 2
        return x*x
            ^
IndentationError: expected an indented block
>>> import ipyparallel as ipp
>>> c=ipp.Client('/root/.ipython/profile_default/security/ipcontroller-client.json')
>>> c.ids
[0, 1]
>>> def misquare(x):
...     return x*x
...
>>> misquare(3)
9
>>> l=c[:].map(misquare,range(4))
>>> l[1]
1
>>> l[3]
9
>>> █

```

## 2.4. Ejercicio Seguridad

Las siguientes capturas reflejan los namespaces:

```

alumno@ubuntu:~/kubernetes/ej4$ cat namespaces.yaml
apiVersion: v1
kind: Namespace
metadata:
  name: desarrollo
  labels:
    name: desarrollo
---
apiVersion: v1
kind: Namespace
metadata:
  name: produccion
  labels:
    name: produccion

alumno@ubuntu:~/kubernetes/ej4$ █

```

La siguiente los serviceaccounts:

```

alumno@ubuntu:~/kubernetes/ej4$ cat serviceaccount.yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: admin
  namespace: adminnm
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: devel
  namespace: desarrollo
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: prod
  namespace: produccion
alumno@ubuntu:~/kubernetes/ej4$ █

```

Lo siguiente los rolebindings asociados a esos serviceaccounts pero como es largo se ha copiado y pegado en vez de captura:

```

alumno@ubuntu:~/kubernetes/ej4$ cat sa-rolebinding.yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: admin-rb
  namespace: desarrollo
subjects:
- kind: ServiceAccount
  name: admin
roleRef:
  kind: ClusterRole
  name: edit
  apiGroup: rbac.authorization.k8s.io
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: admin2-rb
  namespace: produccion
subjects:
- kind: ServiceAccount
  name: admin
roleRef:
  kind: ClusterRole
  name: edit
  apiGroup: rbac.authorization.k8s.io
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:

```



```

  name: devel-rb
  namespace: desarrollo
subjects:
- kind: ServiceAccount
  name: devel
roleRef:
  kind: ClusterRole
  name: edit
  apiGroup: rbac.authorization.k8s.io
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: devel2-rb
  namespace: produccion
subjects:
- kind: ServiceAccount
  name: devel
roleRef:
  kind: ClusterRole
  name: view
  apiGroup: rbac.authorization.k8s.io
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: prod-rb
  namespace: produccion
subjects:
- kind: ServiceAccount
  name: prod
roleRef:
  kind: ClusterRole
  name: edit
  apiGroup: rbac.authorization.k8s.io
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: prod2-rb
  namespace: desarrollo
subjects:
- kind: ServiceAccount
  name: prod
roleRef:
  kind: ClusterRole
  name: view
  apiGroup: rbac.authorization.k8s.io

```

y por ultimo los tokens:

```
alumno@ubuntu:~/kubernetes/ej4$ cat token.yaml
apiVersion: v1
kind: Secret
metadata:
  name: admin-token
  namespace: adminnm
  annotations:
    kubernetes.io/service-account.name: admin
type: kubernetes.io/service-account-token
-----
apiVersion: v1
kind: Secret
metadata:
  name: prod-token
  namespace: produccion
  annotations:
    kubernetes.io/service-account.name: prod
type: kubernetes.io/service-account-token
-----
apiVersion: v1
kind: Secret
metadata:
  name: devel-token
  namespace: desarrollo
  annotations:
    kubernetes.io/service-account.name: devel
type: kubernetes.io/service-account-token
```