UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

ESCUELA DE CIENCIAS Y SISTEMAS

LAB. SOFTWARE AVANZADO

ING. EVEREST DARWIN MEDINILLA RODRIGUEZ

AUX. DIEGO MOLINA

DOCUMENTACIÓN PRÁCTICA 4

201830313 - DENILSON FLORENTÍN DE LEÓN AGUILAR

201931581 - JONATHAN MARCOS VALIENTE GONZÁLEZ

Índice

1. -Documentación de Implementación del Clúster Kubernetes On-Premises con Alta Disponibilidad

1.1. Instalación y Configuración de Kubernetes Utilizando Kubespray

En primer lugar, se utilizó Kubespray para la instalación y configuración del clúster Kubernetes en un entorno on-premises. Este proceso se dividió en varios pasos detallados a continuación:

1.1.1. Creación de Infraestructura Utilizando Terraform:

Se empleó Terraform para crear la infraestructura necesaria en Google Cloud Platform (GCP). Se definieron 4 instancias de Google Compute Engine (GCE), una de las cuales se utilizó como nodo de control (controlplane) con Ansible instalado.

1.1.2. Configuración del Nodo de Control con Ansible:

En el nodo de control, se instaló Ansible y se realizaron las configuraciones necesarias para la ejecución de Kubespray. Se clonó el repositorio de Kubespray, se configuraron las dependencias requeridas y se definieron las claves SSH para el acceso a las demás instancias.

1.1.3. Ejecución de Terraform y Preparación de la Infraestructura:

Se ejecutaron los comandos de Terraform (terraform init, terraform plan, terraform apply) para crear la infraestructura en GCP. Una vez finalizada la creación, se copió la clave privada SSH al nodo de control para establecer la conexión con el resto de nodos.

1.2. Despliegue de un Clúster de Alta Disponibilidad con Nodos Maestro y Trabajadores

Una vez preparada la infraestructura y configurado el nodo de control, se procedió al despliegue del clúster Kubernetes con alta disponibilidad. Este proceso implicó los siguientes pasos:

1.2.1. Preparación del Inventario de Kubespray:

Se copió el directorio de inventario de Kubespray y se modificó el archivo hosts.yaml para incluir las direcciones IP de los nodos maestro y trabajadores. Se asignaron nombres a los nodos y se configuraron las secciones necesarias del archivo.

1.2.2. Configuración de Opciones de Kubespray:

Se ajustaron las opciones de configuración de Kubespray según las necesidades del clúster. Se habilitó la instalación de Helm y se configuraron los plugins de red (en este caso, se utilizó Flannel).

1.2.3. Ejecución de Kubespray:

Con el inventario preparado y las opciones configuradas, se ejecutó Kubespray utilizando el archivo de inventario modificado. Se verificó la conectividad con los nodos y se inició la instalación del clúster.

1.3. Integración de la Pila ELK para Monitorización y Logging Distribuido

Una vez completada la instalación del clúster Kubernetes, se procedió a integrar la pila ELK (Elasticsearch, Logstash y Kibana) para la monitorización y registro de eventos distribuidos dentro del clúster. Este proceso se realizó siguiendo los siguientes pasos:

1.3.1. Configuración de la Pila ELK:

Se instaló y configuró la pila ELK en el clúster Kubernetes. Se realizaron las configuraciones necesarias para la recopilación, procesamiento y visualización de registros de eventos.

1.3.2. Integración con Kubernetes:

Se establecieron conexiones entre los componentes de la pila ELK y Kubernetes para permitir la monitorización en tiempo real y el análisis de registros de eventos del clúster.

1.4. Gestión y Mantenimiento Continuo del Clúster Kubernetes y la Solución de Monitorización

Para garantizar el correcto funcionamiento del clúster Kubernetes y la solución de monitorización a lo largo del tiempo, se proporcionaron instrucciones detalladas sobre la gestión y mantenimiento continuo. Esto incluyó:

1.4.1. Actualizaciones y Parches:

Se describieron los procedimientos para aplicar actualizaciones y parches de seguridad en el clúster y la pila ELK.

1.4.2. Escalado y Balanceo de Carga:

Se detallaron las técnicas para escalar y equilibrar la carga de trabajo en el clúster Kubernetes según las demandas del entorno.

1.4.3. Resolución de Problemas:

Se proporcionaron pautas para identificar y resolver problemas comunes que puedan surgir en la operación diaria del clúster y la solución de monitorización.

1.5. Recomendaciones y Mejores Prácticas para la Configuración Óptima y la Seguridad del Clúster

Se ofrecieron recomendaciones y mejores prácticas para optimizar la configuración y mejorar la seguridad del clúster Kubernetes y la pila ELK. Esto incluyó:

1.5.1. Seguridad de Red:

Se sugirieron medidas para proteger la comunicación entre los nodos del clúster y la pila ELK, así como para restringir el acceso no autorizado.

1.5.2. Gestión de Identidades y Accesos:

Se propusieron estrategias para administrar identidades y accesos en el clúster, como la implementación de autenticación y autorización basadas en roles.

1.5.3. Supervisión y Auditoría:

Se recomendó la implementación de herramientas de supervisión y auditoría para monitorear la actividad del clúster y detectar posibles violaciones de seguridad.

Estas acciones permitieron garantizar un entorno Kubernetes seguro, confiable y altamente disponible, con capacidades de monitorización y registro para facilitar la detección y resolución de problemas.

2. Guía paso a paso para desplegar un clúster de alta disponibilidad con nodos maestro y trabajadores.

1) Haber desplegado la infraestructura en google cloud en la carpeta terraform con los siguientes comandos:

```
terraform init
terraform plan -no-color
terraform apply -no-color
```

   a) Toma en cuenta que la llave ssh configurada debes crearla previamente y sin passphrase

2) Luego accede a la máquina virtual que funciona como control node de master1, worker1 y worker2.

```
ssh pharaox@<IP_externa_control_panel>
```

3) Descarga el repositorio de kubespray

   a) https://github.com/kubernetes-sigs/kubespray.git

4) Clona un ejemplo y trabaja sobre el:

```
cp -rfp inventory/sample inventory/dev
declare -a IPS=(10.250.0.3 10.250.0.2 10.250.0.5)
CONFIG_FILE=inventory/dev/hosts.yaml python3
contrib/inventory_builder/inventory.py ${IPS[@]}
```

5) Configurar el archivo inventory/dev/hosts.yaml

```
all:
 hosts:
   master1:
     ansible_host: 10.250.0.3
     ip: 10.250.0.3
     access_ip: 10.250.0.3
   worker1:
     ansible_host: 10.250.0.2
     ip: 10.250.0.2
     access_ip: 10.250.0.2
```

```
    worker2:
      ansible_host: 10.250.0.5
      ip: 10.250.0.5
      access_ip: 10.250.0.5
  children:
    kube_control_plane:
      hosts:
        master1:

    kube_node:
      hosts:
        worker1:
        worker2:

    etcd:
      hosts:
        master1:

    k8s_cluster:
      children:
        kube_control_plane:
        kube_node:
    calico_rr:
      hosts: {}
```

Nota: las ips las configuras de acuerdo a como las genero terraform.

6) Habilitar help y usar calico

    a) helm_enabled: true. En k8s_cluster/addons.yml

    b) calico es la red predeterminada.

7) Ejecutar el ansible para configurar los workers

```
ansible-playbook -i inventory/dev/hosts.yaml --become --become-user=root
cluster.yml --key-file "~/abbabe"
```

8) En caso de error ejecutar estos comandos

```
ansible-playbook -i inventory/dev/hosts.yaml --become
--user=pharaox --become-user=root reset.yml -e
ansible_python_interpreter=/usr/bin/python3 --key-file
"~/google_compute_engine"
```

```
ansible-playbook -i inventory/mycluster/hosts.yaml --become
--user=pharaox --become-user=root cluster.yml -e
ansible_python_interpreter=/usr/bin/python3 --key-file
"~/google_compute_engine"
```

9) Luego se accede al nodo master 1 y se ejecuta los comandos para ver el kluster de kubespray desplegado:

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

10) Verificación del Estado del Clúster:

a) Una vez finalizada la instalación, verifica el estado del clúster Kubernetes utilizando comandos de `kubectl`.

2.1. Alternativa de Instalación

```
## Create VPC Network
gcloud compute networks create kubernetes-the-kubespray-way --subnet-mode
custom
## Create Net of VPC Network
gcloud compute networks subnets create kubernetes \
 --network kubernetes-the-kubespray-way \
 --range 10.240.0.0/24
## Create firewall rules internal
gcloud compute firewall-rules create
kubernetes-the-kubespray-way-allow-internal \
 --allow tcp,udp,icmp,udp:8472 \
 --network kubernetes-the-kubespray-way \
 --source-ranges 10.240.0.0/24
## Create firewall rules external
gcloud compute firewall-rules create
kubernetes-the-kubespray-way-allow-external \
 --allow tcp:80,tcp:6443,tcp:443,tcp:22,icmp \
 --network kubernetes-the-kubespray-way \
 --source-ranges 0.0.0.0/0

## Create control panel instances
for i in 0 1 2; do
 gcloud compute instances create controller-${i} \
   --async \
```

```
   --boot-disk-size 200GB \
   --can-ip-forward \
   --image-family ubuntu-2004-lts \
   --image-project ubuntu-os-cloud \
   --machine-type e2-standard-2 \
   --private-network-ip 10.240.0.1${i} \
   --scopes
compute-rw,storage-ro,service-management,service-control,logging-write,monit
oring \
   --subnet kubernetes \
   --tags kubernetes-the-kubespray-way,controller
done

## Create worker instances
for i in 0 1 2; do
 gcloud compute instances create worker-${i} \
   --async \
   --boot-disk-size 200GB \
   --can-ip-forward \
   --image-family ubuntu-2004-lts \
   --image-project ubuntu-os-cloud \
   --machine-type e2-standard-2 \
   --private-network-ip 10.240.0.2${i} \
   --scopes
compute-rw,storage-ro,service-management,service-control,logging-write,monit
oring \
   --subnet kubernetes \
   --tags kubernetes-the-kubespray-way,worker
done

## List instances
gcloud compute instances list
--filter="tags.items=kubernetes-the-kubespray-way"

## Stablish ssh key, if you have one
gcloud compute ssh worker-0
gcloud compute ssh worker-1
gcloud compute ssh worker-2
gcloud compute ssh controller-0
gcloud compute ssh controller-1
gcloud compute ssh controller-2
## Test it
IP_CONTROLLER_0=$(gcloud compute instances list
--filter="tags.items=kubernetes-the-kubespray-way AND name:controller-0"
--format="value(EXTERNAL_IP)")
USERNAME=$(whoami)
ssh $USERNAME@$IP_CONTROLLER_0
## In your local Machine
```

```
python3 -m venv venv
source venv/bin/activate
## git clone kubespray
git clone https://github.com/kubernetes-sigs/kubespray.git
cd kubespray
git checkout release-2.17
## Install requirements
pip install -r requirements.txt
## Create cluster
cp -rfp inventory/sample inventory/mycluster
## Declare Ips
declare -a IPS=($(gcloud compute instances list
--filter="tags.items=kubernetes-the-kubespray-way"
--format="value(EXTERNAL_IP)"  | tr '\n' ' '))
CONFIG_FILE=inventory/mycluster/hosts.yaml python3
contrib/inventory_builder/inventory.py ${IPS[@]}
## Change files
## Try ping
ansible -i inventory/mycluster/hosts.yaml -m ping all --key-file
"~/.ssh/google_compute_engine"

## Execute it

ansible-playbook -i inventory/mycluster/hosts.yaml --become --user=pharaox
--become-user=root reset.yml --key-file "~/.ssh/google_compute_engine"

ansible-playbook -i inventory/mycluster/hosts.yaml --become --user=pharaox
--become-user=root cluster.yml --key-file "~/.ssh/google_compute_engine"

## If error
ansible-playbook -i inventory/mycluster/hosts.yaml --become --user=pharaox
--become-user=root reset.yml -e ansible_python_interpreter=/usr/bin/python3
--key-file "~/.ssh/google_compute_engine"

ansible-playbook -i inventory/mycluster/hosts.yaml --become --user=pharaox
--become-user=root cluster.yml -e
ansible_python_interpreter=/usr/bin/python3 --key-file
"~/.ssh/google_compute_engine"

## Proceed to a master node and then create a kubectl handler
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

## Or for local kubectl do
ssh $USERNAME@$IP_CONTROLLER_0
USERNAME=$(whoami)
sudo chown -R $USERNAME:$USERNAME /etc/kubernetes/admin.conf
```

```
exit

scp $USERNAME@$IP_CONTROLLER_0:/etc/kubernetes/admin.conf kubespray-do.conf
##Change the localhost ip to the external IP
apiVersion: v1
clusters:
- cluster:
   certificate-authority-data: XXX
   server: https://35.205.205.80:6443
 name: cluster.local
...
## Then export the config kubectl
export KUBECONFIG=$PWD/kubespray-do.conf

# ELK INSTALLATION

## INSTALL ECK
kubectl create -f https://download.elastic.co/downloads/eck/2.12.1/crds.yaml

## INSTALL RBAC rules
kubectl apply -f
https://download.elastic.co/downloads/eck/2.12.1/operator.yaml

## INSTALL storageClass
kubectl apply -f storageClass.yaml

## INSTALL peristentVolumes
kubectl apply -f persistentVolume.yaml

## INSTALL elasticsearch
kubectl apply -f elasticsearch.yaml

## INSTALL logstash
kubectl apply -f logstash.yaml

## INSTALL kibana
kubectl apply -f kibana.yaml
```

3. Documentación sobre la integración de la pila ELK para monitorización y logging distribuido.

Se procede a configurar los servicios de ELK (ElasticSearch, Kibana y Logstash). Se copian estos archivos y se ejecutan con el comando:

a) kubectl apply -f <manifesto.yaml>

3.1. elasticsearch.yaml

```yaml
apiVersion: elasticsearch.k8s.elastic.co/v1
kind: Elasticsearch
metadata:
 name: quickstart
spec:
 version: 8.13.2
 volumeClaimDeletePolicy: DeleteOnScaledownAndClusterDeletion
 nodeSets:
 - name: default
   count: 1
   config:
     node.roles: ["master", "data", "ingest"]
     node.store.allow_mmap: false
   volumeClaimTemplates:
   - metadata:
       name: elasticsearch-data
     spec:
       accessModes:
       - ReadWriteOnce
       storageClassName: local-storage
       resources:
         requests:
           storage: 30Gi
   podTemplate:
     spec:
       containers:
       - name: elasticsearch
         readinessProbe:
             exec:
               command:
               - bash
               - -c
               - /mnt/elastic-internal/scripts/readiness-probe-script.sh
             failureThreshold: 3
```

```
            initialDelaySeconds: 45
            periodSeconds: 3
            successThreshold: 1
            timeoutSeconds: 45
        env:
          - name: ES_JAVA_OPTS
            value: -Xms2048m -Xmx2048m
          #- name: ELASTICSEARCH_URL
           # value: http://quickstart-es-http.default.svc:9200
        resources:
          requests:
            memory: "4Gi"
            cpu: "2000m"
          limits:
            memory: "4Gi"
            cpu: "2500m"
      tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/master
        operator: Exists
```

3.2. kibana.yaml

```
apiVersion: kibana.k8s.elastic.co/v1
kind: Kibana
metadata:
 name: quickstart
spec:
 version: 8.13.2
 count: 1
 elasticsearchRef:
   name: quickstart
   namespace: default
 podTemplate:
   spec:
     containers:
     - name: kibana
       env:
         - name: NODE_OPTIONS
           value: "--max-old-space-size=2048"
       resources:
         requests:
           memory: 1Gi
           cpu: 0.5
         limits:
```

```
            memory: 2.5Gi
            cpu: 2
      tolerations:
        - effect: NoSchedule
          key: node-role.kubernetes.io/master
          operator: Exists
```

3.3. logstash.yaml

```
apiVersion: logstash.k8s.elastic.co/v1alpha1
kind: Logstash
metadata:
 name: quickstart-logstash
spec:
 version: 8.13.2
 count: 1
 elasticsearchRefs:
   - name: quickstart
     clusterName: qs
 pipelines:
   - pipeline.id: main
     config.string: |
       input {
         beats {
           port => 5044
         }
       }
       output {
         elasticsearch {
           hosts => [ "${QS_ES_HOSTS}" ]
           user => "${QS_ES_USER}"
           password => "${QS_ES_PASSWORD}"
           ssl_certificate_authorities =>
"${QS_ES_SSL_CERTIFICATE_AUTHORITY}"
         }
       }
 services:
   - name: beats
     service:
       spec:
         type: NodePort
         ports:
           - port: 5044
             name: "filebeat"
             protocol: TCP
```

```
            targetPort: 5044
 volumeClaimTemplates:
   - metadata:
       name: logstash-data
     spec:
       accessModes:
       - ReadWriteOnce
       storageClassName: local-storage
       resources:
         requests:
           storage: 10Gi
# podTemplate:
 #  spec:
   #  tolerations:
   #    - effect: NoSchedule
   #      key: node-role.kubernetes.io/master
   #      operator: Exists
```

3.4. storageClass.yaml

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
 name: local-storage
provisioner: kubernetes.io/no-provisioner
volumeBindingMode: WaitForFirstConsumer
```

3.5. persistentVolume.yaml

```
# persistent-volume.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
 name: elastic-pv
spec:
 capacity:
   storage: 30Gi
 volumeMode: Filesystem
 accessModes:
   - ReadWriteOnce
 persistentVolumeReclaimPolicy: Retain
 storageClassName: local-storage
```

```
 local:
   path: /mnt/data/elasticsearch
   #path: /usr/share/elasticsearch/data
 nodeAffinity:
   required:
     nodeSelectorTerms:
     - matchExpressions:
       - key: kubernetes.io/hostname
         operator: In
         values:
         - master1
---
apiVersion: v1
kind: PersistentVolume
metadata:
 name: logstash-pv
spec:
 capacity:
   storage: 10Gi
 volumeMode: Filesystem
 accessModes:
   - ReadWriteOnce
 persistentVolumeReclaimPolicy: Retain
 storageClassName: local-storage
 local:
   path: /mnt/data/logstash
   #path: /usr/share/elasticsearch/data
 nodeAffinity:
   required:
     nodeSelectorTerms:
     - matchExpressions:
       - key: kubernetes.io/hostname
         operator: In
         values:
         - worker1
```

3.6. ELK INSTALLATION

```
# ELK INSTALLATION

## INSTALL ECK
kubectl create -f https://download.elastic.co/downloads/eck/2.12.1/crds.yaml

## INSTALL RBAC rules
kubectl apply -f
https://download.elastic.co/downloads/eck/2.12.1/operator.yaml
```

```
## INSTALL storageClass
kubectl apply -f storageClass.yaml

## INSTALL peristentVolumes
kubectl apply -f persistentVolume.yaml

## INSTALL elasticsearch
kubectl apply -f elasticsearch.yaml

## INSTALL logstash
kubectl apply -f logstash.yaml

## INSTALL kibana
kubectl apply -f kibana.yaml
```

4. Instrucciones para la gestión y mantenimiento continuo del clúster Kubernetes y la solución de monitorización.

Para garantizar un funcionamiento óptimo y continuo del clúster Kubernetes y la solución de monitorización basada en ELK (Elasticsearch, Logstash, Kibana), se deben seguir las siguientes pautas de gestión y mantenimiento:

1. **Actualizaciones y Parches:**
    - Realiza regularmente la aplicación de actualizaciones y parches de seguridad en el clúster Kubernetes y los componentes de la pila ELK. Esto ayudará a mantener el entorno protegido contra posibles vulnerabilidades y fallos de seguridad.
2. **Escalado y Balanceo de Carga:**
    - Monitoriza el rendimiento del clúster Kubernetes y ajusta la capacidad según sea necesario para satisfacer las demandas de la carga de trabajo. Utiliza las capacidades de escalado automático y balanceo de carga para optimizar el uso de recursos y garantizar la disponibilidad de las aplicaciones.
3. **Resolución de Problemas:**
    - Establece procedimientos y herramientas para identificar, diagnosticar y resolver problemas que puedan surgir en el clúster Kubernetes y la solución de monitorización. Mantén registros de eventos y métricas para facilitar la detección y resolución de problemas de manera eficiente.
4. **Copia de Seguridad y Restauración:**
    - Implementa estrategias de copia de seguridad y restauración para proteger los datos y la configuración del clúster Kubernetes y la pila ELK. Realiza copias de seguridad periódicas y prueba regularmente la capacidad de restauración para garantizar la integridad y disponibilidad de los datos.

5. Recomendaciones y Mejores Prácticas para la Configuración Óptima y la Seguridad del Clúster

Para optimizar la configuración y mejorar la seguridad del clúster Kubernetes y la pila ELK, se recomienda seguir las siguientes mejores prácticas:

1. **Seguridad de Red:**
    - Implementa medidas de seguridad de red, como el uso de firewalls y grupos de seguridad, para proteger la comunicación entre los nodos del clúster y los componentes de la pila ELK. Limita el acceso a los servicios solo a las direcciones IP autorizadas y utilizar conexiones seguras mediante protocolos como SSL/TLS.
2. **Gestión de Identidades y Accesos:**
    - Utiliza mecanismos de autenticación y autorización basados en roles para administrar identidades y accesos en el clúster Kubernetes. Asigna permisos de manera granular según las responsabilidades y privilegios de los usuarios y aplicaciones.
3. **Supervisión y Auditoría:**
    - Implementa herramientas de supervisión y auditoría para monitorear la actividad del clúster Kubernetes y la pila ELK. Utiliza registros de eventos y métricas para realizar un seguimiento del rendimiento, detectar posibles problemas y cumplir con los requisitos de conformidad y seguridad.

6. Resultados

Estado	Nombre	Zona	Recomendaciones	En uso por	IP interna	IP externa	Conectar
✓	controller-0	us-east1-b			10.240.0.10 (nic0)	35.237.70.55 (nic0)	SSH
✓	controller-1	us-east1-b			10.240.0.11 (nic0)	35.231.2.89 (nic0)	SSH
✓	controller-2	us-east1-b			10.240.0.12 (nic0)	34.138.1.68 (nic0)	SSH
✓	worker-0	us-east1-b			10.240.0.20 (nic0)	35.243.230.177 (nic0)	SSH
✓	worker-1	us-east1-b			10.240.0.21 (nic0)	34.73.252.157 (nic0)	SSH
✓	worker-2	us-east1-b			10.240.0.22 (nic0)	34.23.16.246 (nic0)	SSH

```
(venv) [pharaox@imperatorsomnium ELK]$ kubectl get nodes
NAME      STATUS   ROLES                  AGE    VERSION
master1   Ready    control-plane,master   2d1h   v1.21.6
master2   Ready    control-plane,master   2d1h   v1.21.6
master3   Ready    control-plane,master   2d1h   v1.21.6
worker1   Ready    <none>                 2d1h   v1.21.6
worker2   Ready    <none>                 2d1h   v1.21.6
worker3   Ready    <none>                 2d1h   v1.21.6
(venv) [pharaox@imperatorsomnium ELK]$ kubectl top nodes
NAME      CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
master1   341m         8%     3993Mi          54%
master2   313m         8%     2229Mi          30%
master3   282m         7%     2073Mi          28%
worker1   178m         4%     2248Mi          29%
worker2   159m         4%     2004Mi          26%
worker3   162m         4%     1014Mi          13%
```

6.1. Usando comandos Kubectl sobre el cluster:

```
>>> Practica4_SA kubectl delete -f k8s/deployment-app.yaml
deployment.apps "app-deployment" deleted
>>> Practica4_SA kubectl apply -f k8s/
deployment.apps/app-deployment created
service/app-service created
>>> Practica4_SA
```

6.2. Resultados:

```
service/app-service created
>>> Practica4_SA kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
app-deployment-75cb7b8cdd-dc2kl   1/1     Running   0          25m
quickstart-es-default-0           1/1     Running   0          13m
quickstart-kb-5c44c94b75-8gzrx    1/1     Running   0          10m
quickstart-logstash-ls-0          1/1     Running   0          11m
>>> Practica4_SA kubectl get deployment
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
app-deployment   1/1     1            1           25m
quickstart-kb    1/1     1            1           11m
>>> Practica4_SA kubectl get services
NAME                          TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
app-service                   ClusterIP   10.233.6.71     <none>        3000/TCP         25m
kubernetes                    ClusterIP   10.233.0.1      <none>        443/TCP          2d1h
quickstart-es-default         ClusterIP   None            <none>        9200/TCP         14m
quickstart-es-http            ClusterIP   10.233.57.255   <none>        9200/TCP         14m
quickstart-es-internal-http   ClusterIP   10.233.54.33    <none>        9200/TCP         14m
quickstart-es-transport       ClusterIP   None            <none>        9300/TCP         14m
quickstart-kb-http            ClusterIP   10.233.44.190   <none>        5601/TCP         11m
quickstart-logstash-ls-api    ClusterIP   None            <none>        9600/TCP         11m
quickstart-logstash-ls-beats  NodePort    10.233.40.30    <none>        5044:31697/TCP   12m
>>> Practica4_SA
```