

## Instalación y configuración de Google Cloud en el entorno

Paso 1: Agregar la dirección del SDK de Google Cloud como un recurso de paquetes dentro del sistema.

```
~$ echo "deb [signed-by=/usr/share/keyrings/cloud.google.gpg] http://packages.cloud.google.com/apt cloud-sdk main" | sudo tee -a /etc/apt/sources.list.d/google-cloud-sdk.list
```

### Paso 1

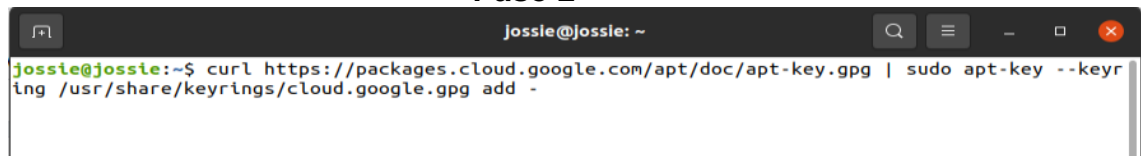
A terminal window titled 'jossie@jossie: ~' showing the command: `echo "deb [signed-by=/usr/share/keyrings/cloud.google.gpg] http://packages.cloud.google.com/apt cloud-sdk main" | sudo tee -a /etc/apt/sources.list.d/google-cloud-sdk.list` The output is not visible, but the command is executed.

Elaboración propia

Paso 2: Importar la llave pública de Google dentro del entorno.

```
~$ curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key --keyring /usr/share/keyrings/cloud.google.gpg add -
```

### Paso 2

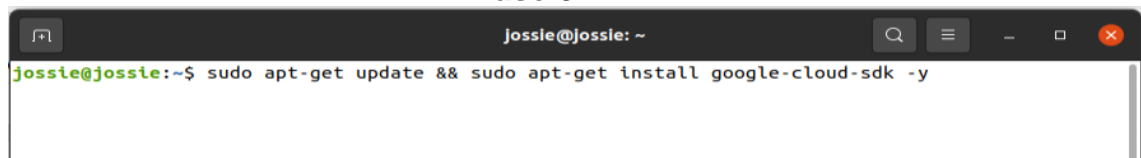
A terminal window titled 'jossie@jossie: ~' showing the command: `curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key --keyring /usr/share/keyrings/cloud.google.gpg add -` The output is not visible, but the command is executed.

Elaboración propia

Paso 3: Actualizar la lista de paquetes e instalar Cloud SDK, tomar en cuenta que esta instalación puede tardar varios minutos.

```
~$ sudo apt-get update && sudo apt-get install google-cloud-sdk
```

### Paso 3

A terminal window titled 'jossie@jossie: ~' showing the command: `sudo apt-get update && sudo apt-get install google-cloud-sdk -y` The output is not visible, but the command is executed.

Elaboración propia

Paso 4: Inicializar el SDK, luego de algunos segundos pedirá iniciar sesión con una cuenta para lo cual se debe presionar “Y” y abrirá una ventana en el navegador para ingresar con alguna cuenta.

```
~$ gcloud init
```

## Paso 4

```
jossie@jossie:~$ gcloud init
Welcome! This command will take you through the configuration of gcloud.

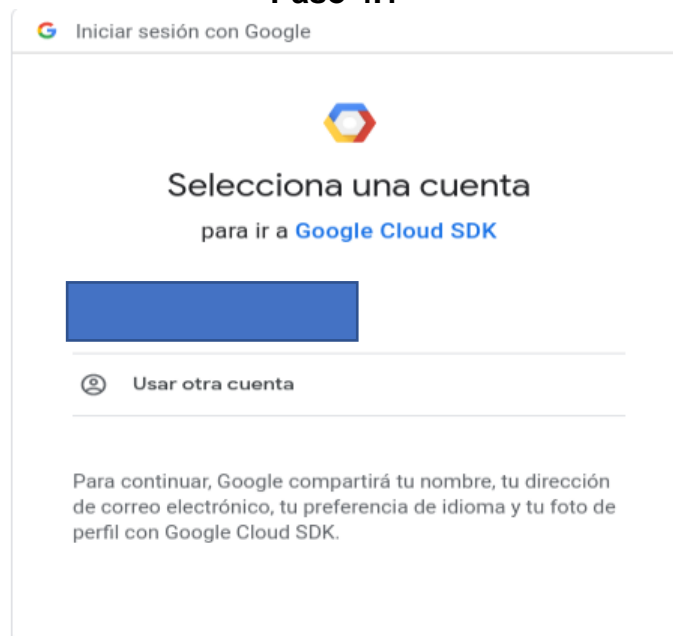
Settings from your current configuration [default] are:
compute:
  region: us-east1
  zone: us-east1-b
core:
  account: [REDACTED]
  disable_usage_reporting: 'True'
  project: tesis-demo

Pick configuration to use:
[1] Re-initialize this configuration [default] with new settings
[2] Create a new configuration
Please enter your numeric choice:
```

Elaboración propia

Paso 4.1: Seleccionar una cuenta para iniciar sesión.

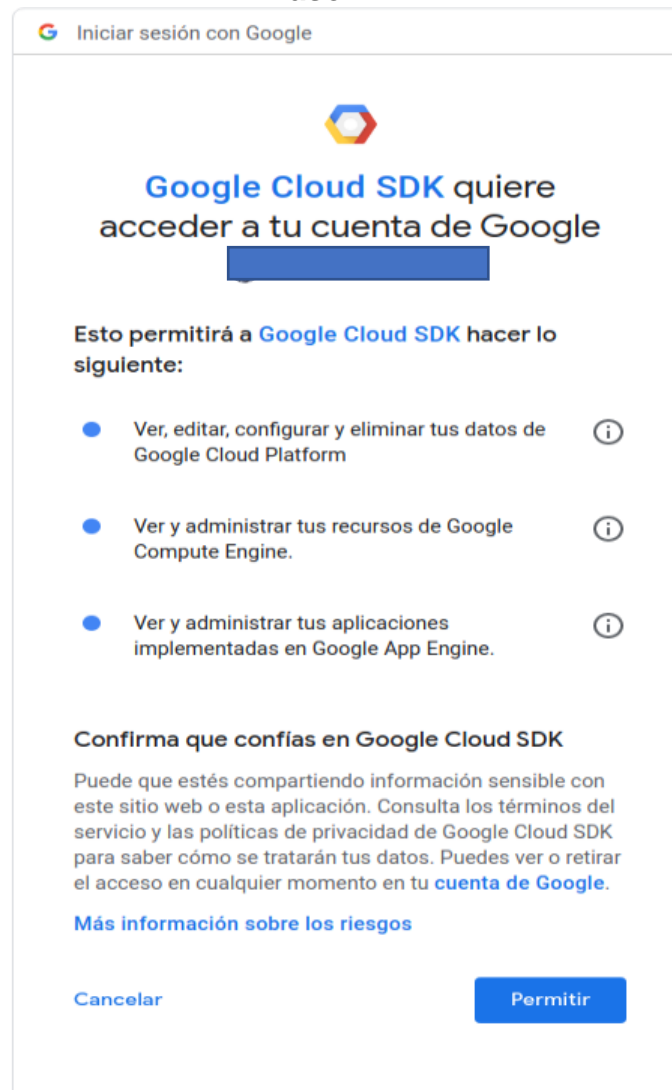
## Paso 4.1



Elaboración propia

Paso 4.2: Se permite el acceso y procederá a mostrar una ventana en la cual indica que ya se ha accedido.

#### Paso 4.2



Elaboración propia

Paso 5: La terminal solicita seleccionar un proyecto previamente creado en Google Cloud, se seleccionó indicando el número respectivo a su nombre, 6 haciendo referencia al proyecto "tesis-demo"

## Paso 5

```
jossie@jossie: ~  
Network diagnostic detects and fixes local network connection issues.  
Checking network connection...done.  
Reachability Check passed.  
Network diagnostic passed (1/1 checks passed).  
  
Choose the account you would like to use to perform operations for  
this configuration:  
[1]   
[2] Log in with a new account  
Please enter your numeric choice: 1  
  
You are logged in as: [bkcastrillo@gmail.com].  
  
Pick cloud project to use:  
[1] bd2grupo6  
[2] grupo28-ing-usac-so2  
[3] optical-bond-302622  
[4] servicemeshcon2021  
[5] sopes1-305916  
[6] tesis-demo  
[7] Create a new project  
Please enter numeric choice or text value (must exactly match list  
item):
```

Elaboración propia

Paso 6: Se selecciona una región y zona por default para trabajar y se presiona “Y”.

## Paso 6

```
jossie@jossie: ~  
Choose the account you would like to use to perform operations for  
this configuration:  
[1]   
[2] Log in with a new account  
Please enter your numeric choice: 1  
  
You are logged in as: [bkcastrillo@gmail.com].  
  
Pick cloud project to use:  
[1] bd2grupo6  
[2] grupo28-ing-usac-so2  
[3] optical-bond-302622  
[4] servicemeshcon2021  
[5] sopes1-305916  
[6] tesis-demo  
[7] Create a new project  
Please enter numeric choice or text value (must exactly match list  
item): y  
Please enter a value between 1 and 7, or a value present in the list: 6  
  
Your current project has been set to: [tesis-demo].  
Do you want to configure a default Compute Region and Zone? (Y/n)?
```

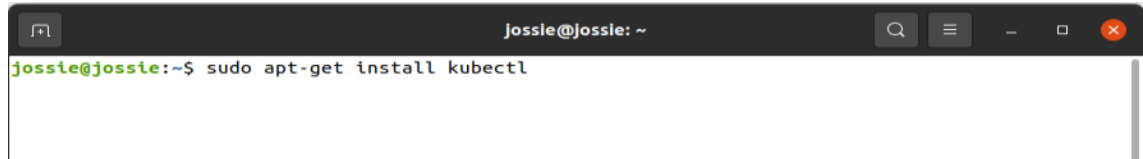
Elaboración propia

## Implementar cluster de Kubernetes

Paso 7: Instalar el cliente de Kubernetes.

```
~$ sudo apt-get install kubectl
```

### Paso 7

A terminal window titled 'jossie@jossie: ~' showing the command 'sudo apt-get install kubectl' being entered at the prompt 'jossie@jossie:~\$'.

Elaboración propia

Paso 8: Verificar la instalación.

```
~$ kubectl version --short
```

### Paso 8

A terminal window titled 'jossie@jossie: ~' showing the command 'kubectl version --short' being entered at the prompt 'jossie@jossie:~\$'.

Elaboración propia

Paso 9: Crear un clúster para el despliegue de la arquitectura.

```
~$ gcloud container clusters create tesisdemo --num-nodes=2 --tags=allin,allout --enable-legacy-authorization --enable-basic-auth --issue-client-certificate --machine-type=n1-standard-2 --no-enable-network-policy
```

Explicación de banderas utilizadas:

- `--num-nodes=2`: Esta bandera sirve para definir la cantidad de nodos que tendrá el cluster, en este caso se generaron únicamente 2.
- `--tags=allin,allout`: Se indican tags referentes a reglas de firewall con la apertura de puertos, en este caso se permitió todo el tráfico de entrada o salida.
- `--enable-legacy-authorization`: Esta política de kubernetes otorga permisos definidos a todos los usuarios del cluster.
- `--machine-type=n1-standard-2`: Esta etiqueta sirve para definir el tipo de hardware que se utilizará, el tipo n1 standard 2 es de uso general el cual posee 2 núcleos, 7.50Gb de memoria RAM, un máximo de 128 discos persistentes, un almacenamiento local de 257 TB en SSD y un ancho de banda de 10Gbps.

## Paso 9

```
jossie@jossie:~$ gcloud container clusters create tesisdemo --num-nodes=2 --tags=allin,allout --enable-legacy-authorization --enable-basic-auth --issue-client-certificate --machine-type=n1-standard-2 --no-enable-network-policy
```

Elaboración propia

Paso 9.1: Se muestra un mensaje con el estatus final y se puede visualizar desde la plataforma en la nube el clúster creado.

## Paso 9.1

```
jossie@jossie:~$ gcloud container clusters create tesisdemo --num-nodes=2 --tags=allin,allout --enable-legacy-authorization --enable-basic-auth --issue-client-certificate --machine-type=n1-standard-2 --no-enable-network-policy
WARNING: Starting in January 2021, clusters will use the Regular release channel by default when '--cluster-version', '--release-channel', '--no-enable-autoupgrade', and '--no-enable-autorepair' flags are not specified.
WARNING: Currently VPC-native is not the default mode during cluster creation. In the future, this will become the default mode and can be disabled using '--no-enable-ip-alias' flag. Use '--no-enable-ip-alias' flag to suppress this warning.
WARNING: Starting with version 1.18, clusters will have shielded GKE nodes by default.
WARNING: Your Pod address range ('--cluster-ipv4-cidr') can accommodate at most 1008 node(s).
WARNING: Starting with version 1.19, newly created clusters and node-pools will have COS_CONTAINERD as the default node image when no image type is specified.
Creating cluster tesisdemo in us-east1-b... Cluster is being health-checked (master is healthy)...done.
Created [https://container.googleapis.com/v1/projects/tesis-demo/zones/us-east1-b/clusters/tesisdemo].
To inspect the contents of your cluster, go to: https://console.cloud.google.com/kubernetes/workload/gcloud/us-east1-b/tesisdemo?project=tesis-demo
kubeconfig entry generated for tesisdemo.
NAME          LOCATION    MASTER_VERSION  MASTER_IP      MACHINE_TYPE  NODE_VERSION    NUM_NODES
tesisdemo     us-east1-b  1.18.16-gke.2100 35.196.108.82  n1-standard-2 1.18.16-gke.2100 2
STATUS
RUNNING
jossie@jossie:~$
```

Elaboración propia

Paso 10: Se obtienen las credenciales para el manejo del cluster.

~\$ gcloud container clusters get-credentials tesisdemo --zone us-central1-c --project tesis-demo

## Paso 10

```
jossie@jossie:~$ gcloud container clusters get-credentials tesisdemo --zone us-east1-b --project tesis-demo
Fetching cluster endpoint and auth data.
kubeconfig entry generated for tesisdemo.
jossie@jossie:~$
```

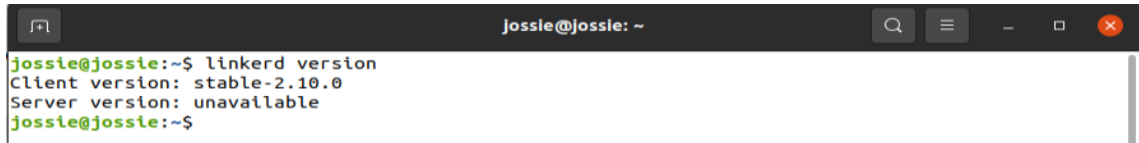
Elaboración propia

## Instalación de Linkerd

Paso 11: Instalar la línea de comandos de Linkerd.

```
~$ curl -sL run.linkerd.io/install | sh
```

### Paso 11

A terminal window titled 'jossie@jossie: ~' showing the command 'linkerd version' and its output: 'Client version: stable-2.10.0' and 'Server version: unavailable'.

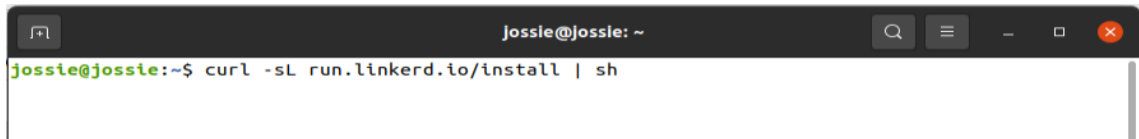
```
jossie@jossie:~$ linkerd version
Client version: stable-2.10.0
Server version: unavailable
jossie@jossie:~$
```

Elaboración propia

Paso 11.1: Verificar instalación.

```
~$ linkerd versión
```

### Paso 11.1

A terminal window titled 'jossie@jossie: ~' showing the command 'curl -sL run.linkerd.io/install | sh'.

```
jossie@jossie:~$ curl -sL run.linkerd.io/install | sh
```

Elaboración propia

Paso 12: Validar que el cluster de Kubernetes creado cumple con lo necesario para Linkerd.

```
~$ linkerd check --pre
```

### Paso 12

A terminal window titled 'jossie@jossie: ~' showing the command 'linkerd check --pre' with a cursor at the end.

```
jossie@jossie:~$ linkerd check --pre
```

Elaboración propia

Paso 12.1: Al estar todo correcto muestra si el clúster está listo para la instalación posterior.

## Paso 12.1

```
jossie@jossie:~$ linkerd check --pre
kubernetes-api
-----
✓ can initialize the client
✓ can query the Kubernetes API

kubernetes-version
-----
✓ is running the minimum Kubernetes API version
✓ is running the minimum kubectl version

pre-kubernetes-setup
-----
✓ control plane namespace does not already exist
✓ can create non-namespaced resources
✓ can create ServiceAccounts
✓ can create Services
✓ can create Deployments
✓ can create CronJobs
✓ can create ConfigMaps
✓ can create Secrets
✓ can read Secrets
✓ can read extension-apiserver-authentication configmap
✓ no clock skew detected

pre-kubernetes-capability
-----
!! has NET_ADMIN capability
   found 1 PodSecurityPolicies, but none provide NET_ADMIN, proxy injection will fail if the PSP
   admission controller is running
   see https://linkerd.io/checks/#pre-k8s-cluster-net-admin for hints
!! has NET_RAW capability
   found 1 PodSecurityPolicies, but none provide NET_RAW, proxy injection will fail if the PSP a
   dmission controller is running
   see https://linkerd.io/checks/#pre-k8s-cluster-net-raw for hints

linkerd-version
-----
✓ can determine the latest version
!! cli is up-to-date
   is running version 2.10.0 but the latest stable version is 2.10.1
   see https://linkerd.io/checks/#l5d-version-cli for hints

Status check results are ✓
jossie@jossie:~$
```

Elaboración propia

Paso 13: Instalar el Control Plane dentro del cluster, esto genera la observabilidad para el mismo. Tener en cuenta que esta configuración tarda varios minutos.

```
~$ linkerd install | kubectl apply -f -
```

## Paso 13

```
jossie@jossie:~$ linkerd install | kubectl apply -f -
```

Elaboración propia

Paso 14: Verificar la instalación.

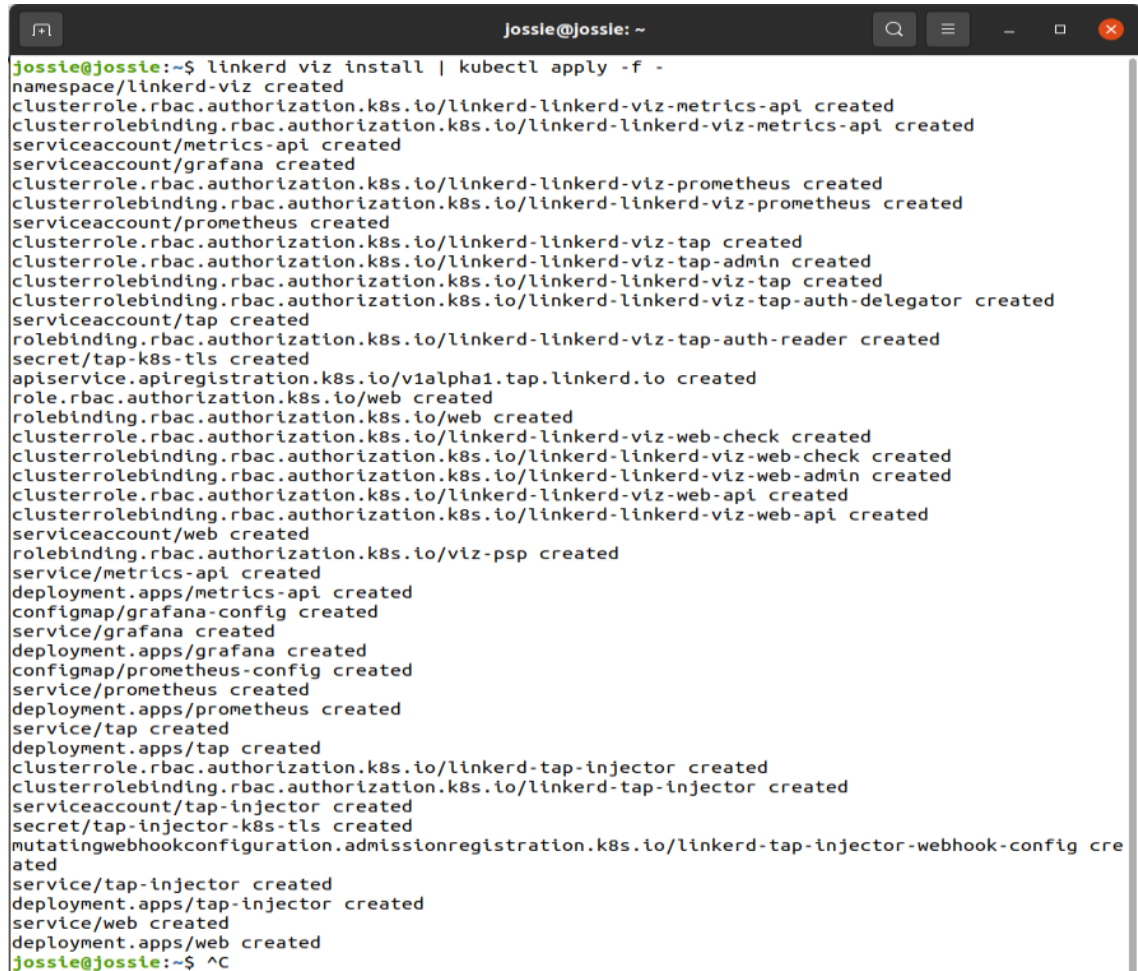
```
~$ linkerd check
```



Paso 15: Agregar la extensión para visualizar el estado de la arquitectura desde el dashboard de Linkerd.

```
~$ linkerd viz install | kubectl apply -f -
```

### Paso 15

A terminal window titled 'jossie@jossie: ~' showing the output of the command 'linkerd viz install | kubectl apply -f -'. The output lists numerous Kubernetes resources created, including namespaces, clusterroles, clusterrolebindings, serviceaccounts, roles, rolebindings, secrets, apiservices, deployments, configmaps, and services, all related to the Linkerd visualization components.

```
jossie@jossie:~$ linkerd viz install | kubectl apply -f -
namespace/linkerd-viz created
clusterrole.rbac.authorization.k8s.io/linkerd-linkerd-viz-metrics-api created
clusterrolebinding.rbac.authorization.k8s.io/linkerd-linkerd-viz-metrics-api created
serviceaccount/metrics-api created
serviceaccount/grafana created
clusterrole.rbac.authorization.k8s.io/linkerd-linkerd-viz-prometheus created
clusterrolebinding.rbac.authorization.k8s.io/linkerd-linkerd-viz-prometheus created
serviceaccount/prometheus created
clusterrole.rbac.authorization.k8s.io/linkerd-linkerd-viz-tap created
clusterrole.rbac.authorization.k8s.io/linkerd-linkerd-viz-tap-admin created
clusterrolebinding.rbac.authorization.k8s.io/linkerd-linkerd-viz-tap created
clusterrolebinding.rbac.authorization.k8s.io/linkerd-linkerd-viz-tap-auth-delegator created
serviceaccount/tap created
rolebinding.rbac.authorization.k8s.io/linkerd-linkerd-viz-tap-auth-reader created
secret/tap-k8s-tls created
apiservice.apiregistration.k8s.io/v1alpha1.tap.linkerd.io created
role.rbac.authorization.k8s.io/web created
rolebinding.rbac.authorization.k8s.io/web created
clusterrole.rbac.authorization.k8s.io/linkerd-linkerd-viz-web-check created
clusterrolebinding.rbac.authorization.k8s.io/linkerd-linkerd-viz-web-check created
clusterrolebinding.rbac.authorization.k8s.io/linkerd-linkerd-viz-web-admin created
clusterrole.rbac.authorization.k8s.io/linkerd-linkerd-viz-web-api created
clusterrolebinding.rbac.authorization.k8s.io/linkerd-linkerd-viz-web-api created
serviceaccount/web created
rolebinding.rbac.authorization.k8s.io/viz-psp created
service/metrics-api created
deployment.apps/metrics-api created
configmap/grafana-config created
service/grafana created
deployment.apps/grafana created
configmap/prometheus-config created
service/prometheus created
deployment.apps/prometheus created
service/tap created
deployment.apps/tap created
clusterrole.rbac.authorization.k8s.io/linkerd-tap-injector created
clusterrolebinding.rbac.authorization.k8s.io/linkerd-tap-injector created
serviceaccount/tap-injector created
secret/tap-injector-k8s-tls created
mutatingwebhookconfiguration.admissionregistration.k8s.io/linkerd-tap-injector-webhook-config created
service/tap-injector created
deployment.apps/tap-injector created
service/web created
deployment.apps/web created
jossie@jossie:~$ ^C
```

Elaboración propia

Paso 16: Habilitar el dashboard cuando sea necesario para el monitoreo, esto abre una ventana en el navegador con el dashboard.

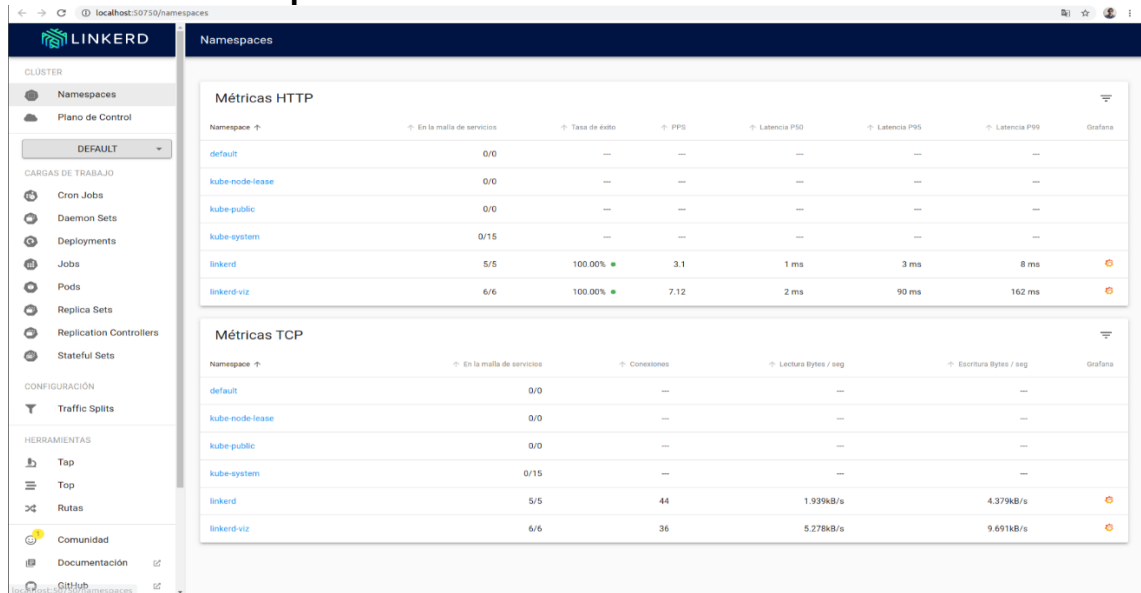
```
~$ linkerd viz dashboard &
```

## Paso 16

```
jossie@jossie:~$ linkerd viz dashboard
Linkerd dashboard available at:
http://localhost:50750
Grafana dashboard available at:
http://localhost:50750/grafana
Opening Linkerd dashboard in the default browser
Abriendo en una sesión existente del navegador
```

Elaboración propia

## Vista preliminar del dashboard de Linkerd



The screenshot shows the Linkerd dashboard with a sidebar on the left containing navigation links like Namespaces, Plano de Control, and various workload types. The main content area displays two tables: 'Métricas HTTP' and 'Métricas TCP'. Both tables list metrics for namespaces including default, kube-node-lease, kube-public, kube-system, linkerd, and linkerd-viz.

Métricas HTTP							
Namespace	En la malla de servicios	Tasa de éxito	PPS	Latencia P50	Latencia P95	Latencia P99	Grafana
default	0/0	—	—	—	—	—	—
kube-node-lease	0/0	—	—	—	—	—	—
kube-public	0/0	—	—	—	—	—	—
kube-system	0/15	—	—	—	—	—	—
linkerd	5/5	100.00%	3.1	1 ms	3 ms	8 ms	🔴
linkerd-viz	6/6	100.00%	7.12	2 ms	90 ms	162 ms	🔴

Métricas TCP				
Namespace	En la malla de servicios	Conexiones	Lectura Bytes / seg	Escritura Bytes / seg
default	0/0	—	—	—
kube-node-lease	0/0	—	—	—
kube-public	0/0	—	—	—
kube-system	0/15	—	—	—
linkerd	5/5	44	1.939kB/s	4.379kB/s
linkerd-viz	6/6	36	5.278kB/s	9.691kB/s

Elaboración propia

## Instalación de Chaos Mesh

Paso 17: Instalar Chaos Mesh.

```
~$ curl -sSL https://mirrors.chaos-mesh.org/v1.1.2/install.sh | bash
```

## Paso 17

```
jossie@jossie:~$ curl -sSL https://mirrors.chaos-mesh.org/v1.2.0/install.sh | bash
```

Elaboración propia

Paso 18: Verificar instalación.

```
~$ kubectl get pod -n chaos-testing
```

## Paso 18

```
Jossie@jossie: /
jossie@jossie:/$ kubectl get pod -n chaos-testing
NAME                                READY    STATUS    RESTARTS   AGE
chaos-controller-manager-76bb7c546d-wfrfv  1/1      Running   0           3m15s
chaos-daemon-d4wx6                      1/1      Running   0           3m16s
chaos-daemon-nx22x                      1/1      Running   0           3m16s
chaos-dashboard-759bb9768d-fqmf9         1/1      Running   0           3m15s
jossie@jossie:/$
```

Elaboración propia

## Construcción de aplicaciones

Para esta sección es necesario clonar el repositorio adjunto en los anexos dentro del sistema para los pasos posteriores y la generación de la aplicación de prueba.

Paso 19: Ingresar a la carpeta de la aplicación Cliente y ejecutar el siguiente comando.

```
~$ sudo ./build.sh jossie
```

### Paso 19

```
Jossie@jossie: ~/Escritorio/ExperimentosTesis/Arquitectura/Cliente
jossie@jossie:~/Escritorio/ExperimentosTesis/Arquitectura/Cliente$ sudo ./build.sh jossie
```

Elaboración propia

Paso 20: Ingresar la contraseña de su dockerhub e iniciara la construcción de la imagen, posteriormente se puede verificar en su repositorio la misma. Se puede utilizar las aplicaciones ya creadas en el repositorio de la plantilla de la arquitectura.

### Paso 20



```
jossie@jossie: ~/Escritorio/ExperimentosTesis/Arquitectura/Cliente
Step 2/5 : MAINTAINER Jossie
--> Using cache
--> b0ab16851f80
Step 3/5 : RUN apt-get update
--> Using cache
--> c4040e251bd6
Step 4/5 : RUN apt-get install -y siege
--> Using cache
--> 495c0dbaf4a7
Step 5/5 : CMD ["siege", "-b", "http://apache"]
--> Using cache
--> b0648373bbd4
Successfully built b0648373bbd4
Successfully tagged client:latest
Using default tag: latest
The push refers to repository [docker.io/jossie/client]
0043a51434c3: Layer already exists
3f0edf811640: Pushed
8cafc6d2db45: Layer already exists
a5d4bacb0351: Layer already exists
5153e1acaabc: Layer already exists
latest: digest: sha256:355d6ecb94312960bd6dbb37a34723a386c6db5b39cd5a709f8e7c7919fb577b size: 1366
jossie@jossie:~/Escritorio/ExperimentosTesis/Arquitectura/Cliente$
```

Elaboración propia

Paso 21: Repetir los dos pasos anteriores con la carpeta Servidor

## Definición de las aplicaciones

La totalidad de la arquitectura está implementada dentro de un archivo con extensión yaml para un despliegue sencillo en el cual se definen ambas aplicaciones

### Dockerfile Cliente

Para la definición de la aplicación cliente se definió un dockerfile en base al cual se generó una imagen para el despliegue utilizado como sistema Ubuntu 18.04 y una actualización del sistema. Posteriormente se instaló y ejecuto "Siege" la cual es una herramienta orientada a las pruebas sobre peticiones http y https, para los fines de los experimentos estará realizando peticiones continuas a los servidores apache.

#### Código cliente

```
1 FROM ubuntu:18.04
2 MAINTAINER Jossie
3 RUN apt-get update
4 RUN apt-get install -y siege
5 CMD ["siege", "-b", "http://apache"]
```

Elaboración propia

### Dockerfile Servidor

Para la definición del servidor se utiliza una imagen de Ubuntu 18.04 en la cual se realiza una actualización del sistema y la instalación de apache2 como servidor web. Posteriormente se expuso el servidor apache en el puerto 80 del contenedor para las peticiones.

#### Código servidor

```
1 FROM ubuntu:18.04
2 MAINTAINER Jossie
3 RUN apt-get update
4 RUN apt-get install -y apache2
5 EXPOSE 80
6 CMD ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]
```

Elaboración propia