



台灣積體電路製造股份有限公司
Taiwan Semiconductor Manufacturing Company, Ltd.

TSMC IT X NCTU CS 課號 5270

CLOUD NATIVE
Development Best Practice

設計以及架構建置 K8S 自有叢集 - LAB

資訊系統暨通訊服務處 系統建構一部 | 李青峰
March 16, 2022



AGENDA

Lab 0: Preparation

Lab 1: Pod

Lab 2: Service & Ingress

Lab 3: Deployments

Bonus


TSMC IT X NCTU CS 課號 5270

CLOUD NATIVE
Development Best Practice

Lab 0: Preparation

1. Open Google Cloud Console
2. Spin-up GKE Worker Instances
3. Open Cloud Shell

Lab 0: Preparation

1.  **Open Google Cloud Console**
2. Spin-up GKE Worker Instances
3. Open Cloud Shell

 **Let's get started!**

Lab 0-1: Open Google Cloud Console

1. Navigate to <https://console.cloud.google.com/> and use your gmail account to login
2. You should see the Google cloud console:

The screenshot shows the Google Cloud Platform console interface. At the top, there's a blue header bar with the Google Cloud Platform logo, the project name 'nycu-lab-1', a search bar, and a 'DISMISS' button next to an 'ACTIVATE' button. Below the header, the dashboard is divided into three main sections: 'Project info', 'Compute Engine', and 'Google Cloud Platform status'. The 'Project info' section displays the project name 'nctu-lab-1', project number '589346162470', and project ID 'nycu-lab-1'. The 'Compute Engine' section shows a CPU usage graph. The 'Google Cloud Platform status' section indicates 'All services normal'. On the left side, there's a sidebar with navigation links: Home, Recent, View all products, and a 'PINNED' section with links to IAM & Admin, Billing, APIs & Services, Marketplace, and Compute Engine. At the bottom right, there's a 'Billing' section showing estimated charges for the billing period Mar 1 - 15, 2022, as USD \$0.00.

Start your Free Trial with \$300 in credit. Don't worry—you won't be charged if you run out of credits. [Learn more](#)

DISMISS ACTIVATE

Google Cloud Platform nycu-lab-1 Search Products, resources, docs (/)

DASHBOARD ACTIVITY RECOMMENDATIONS CUSTOMIZE

Home > Recent > View all products

PINNED

- IAM & Admin >
- Billing
- APIs & Services >
- Marketplace
- Compute Engine >

Project info

Project name
nctu-lab-1

Project number
589346162470

Project ID
nycu-lab-1

ADD PEOPLE TO THIS PROJECT

Go to project settings

Compute Engine

CPU (%)

1.0
0.8
0.6
0.4
0.2
0

3 PM 3:15 3:30 3:45

Google Cloud Platform status

All services normal

Go to Cloud status dashboard

Billing

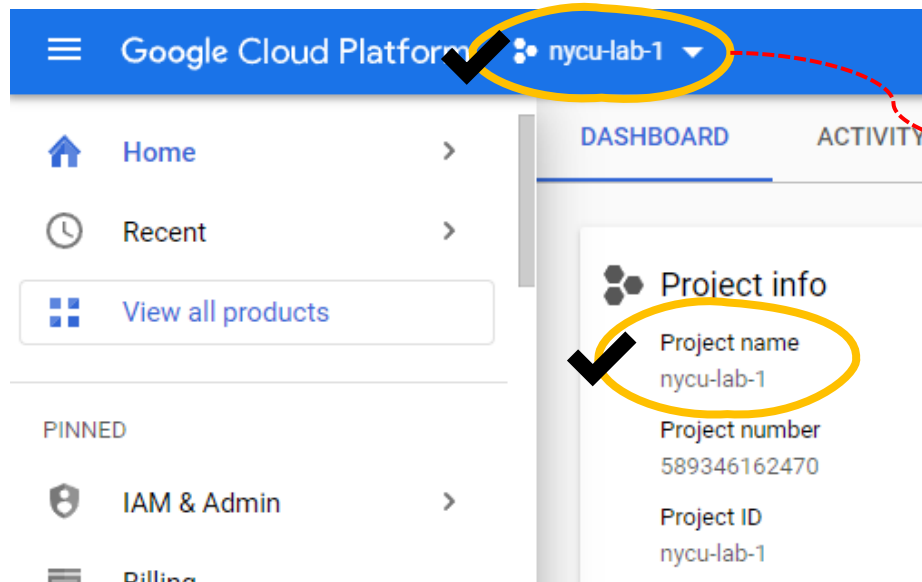
Estimated charges
For the billing period Mar 1 - 15, 2022 USD \$0.00

Take a tour of billing

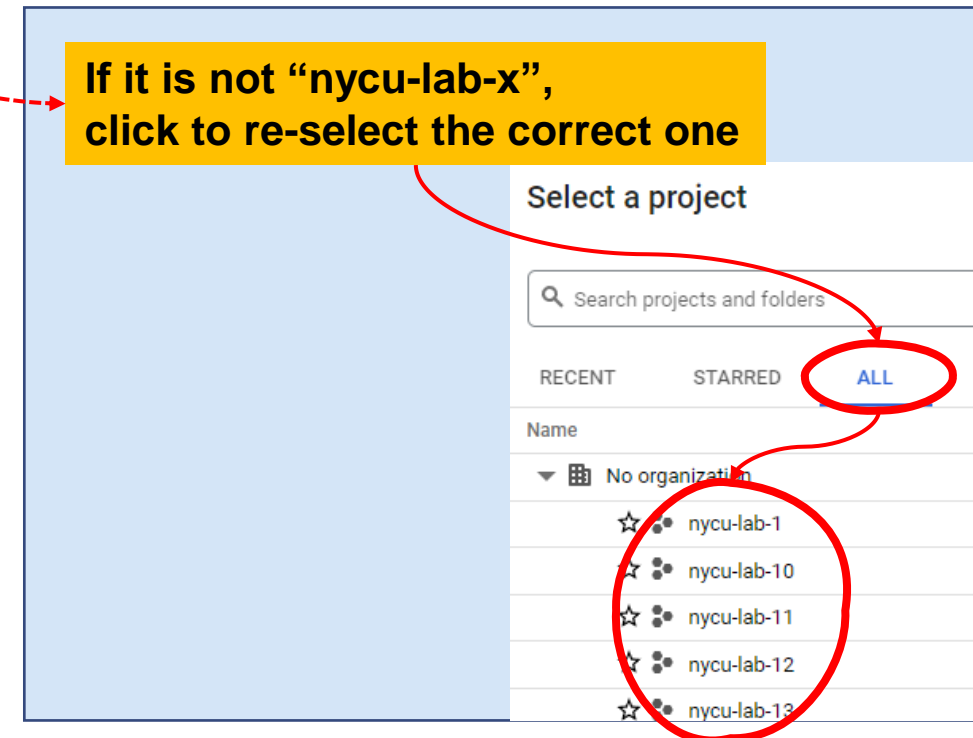
View detailed charges

Lab 0-1: Open Google Cloud Console

3. Please make sure your current project is correct:

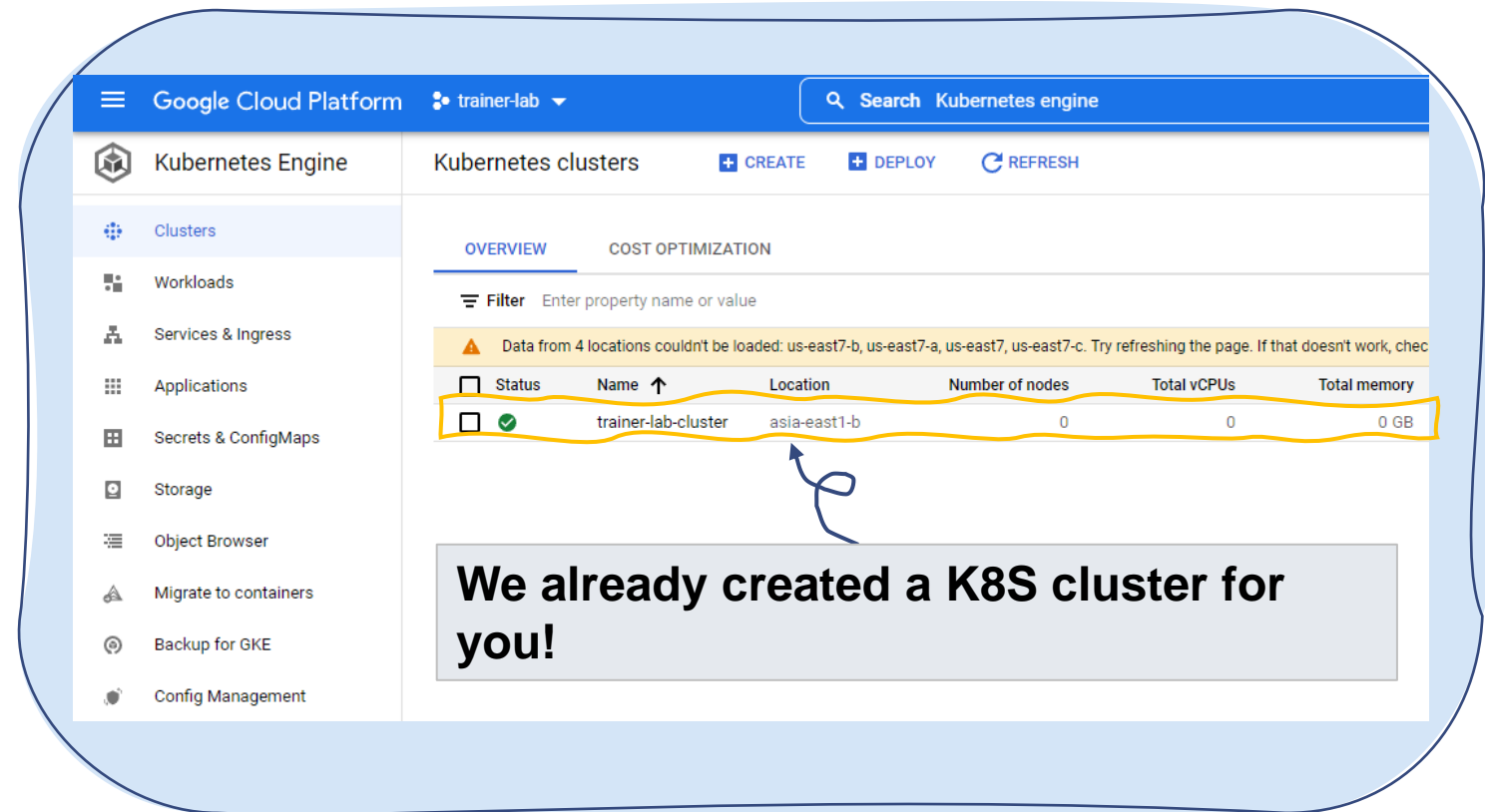
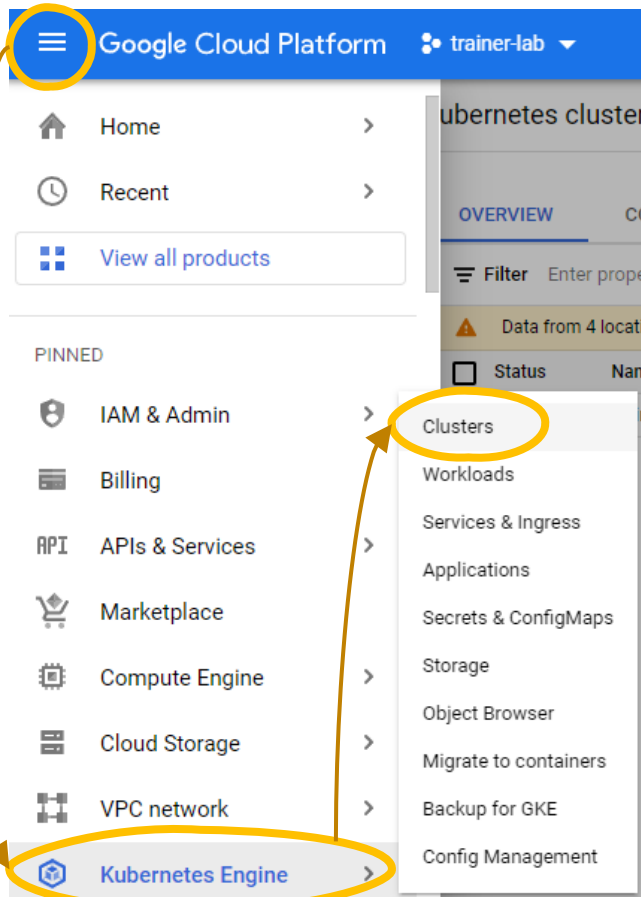


If it is not "nycu-lab-x",
click to re-select the correct one



Lab 0-1: Open Google Cloud Console

4. Navigate to “Kubernetes Engine” → “Clusters”



Lab 0: Preparation

1. Open Google Cloud Console
2.  **Spin-up GKE Worker Instances**
3. Open Cloud Shell

- ❑ Our K8s cluster looks fine, but it does not have any worker nodes...
- ❑ Let's add some instances to it

Lab 0-2: Spin-up GKE Worker Instances

1. Navigate to “Kubernetes Engine” → “Clusters” (Lab 0-1)
2. Click on the cluster name

The screenshot shows the Google Cloud Platform console interface. The top navigation bar includes the Google Cloud Platform logo, the account name 'trainer-lab', and a search bar. The left sidebar contains a list of services: Kubernetes Engine, Clusters, Workloads, Services & Ingress, Applications, Secrets & ConfigMaps, Storage, and Object Browser. The main content area is titled 'Kubernetes clusters' and includes buttons for '+ CREATE', '+ DEPLOY', and 'REFRESH'. Below this, there are tabs for 'OVERVIEW' and 'COST OPTIMIZATION'. A filter bar is present with the text 'Filter Enter property name or value'. A warning message states: 'Data from 4 locations couldn't be loaded: us-east7-a, us-east7, us-east7-c, us-east7-b. Try'. Below the warning is a table with the following columns: Status, Name, Location, and Number of nodes. The table contains one row with a green checkmark in the Status column, the name 'trainer-lab-cluster' in the Name column, 'asia-east1-b' in the Location column, and '0' in the Number of nodes column. The 'trainer-lab-cluster' text is circled in yellow, and a yellow circle with the number '2' is placed below it.

Status	Name	Location	Number of nodes
✓	trainer-lab-cluster	asia-east1-b	0

Lab 0-2: Spin-up GKE Worker Instances

❑ Notice we have 0 node in the node pool 😞

3. Click on the node pool name

4. Then click “Resize”

The screenshot displays the Google Cloud Platform console for a GKE cluster named 'trainer-lab-cluster'. The 'NODES' tab is selected, showing a table of node pools. The 'ubuntu-pool' is highlighted with a yellow circle and the number '3'. The table indicates 0 nodes for this pool. A yellow arrow points from the 'RESIZE' button in the 'Node pool details' sidebar to the 'ubuntu-pool' row. The sidebar also shows the 'Number of nodes' is 0.

trainer-lab-cluster

⚠️ Your cluster has one or more unschedulable pods. [Autoscaling documentation](#)

DETAILS **NODES** STORAGE LOGS

Node Pools

Filter Filter node pools

Name ↑	Status	Version	Number of nodes	Machine type
ubuntu-pool	Ok	1.21.5-gke.1805	0	e2-standard-4

Node pool details REFRESH EDIT DELETE **RESIZE**

ubuntu-pool

Cluster [trainer-lab-cluster](#)

Node version 1.21.5-gke.1805

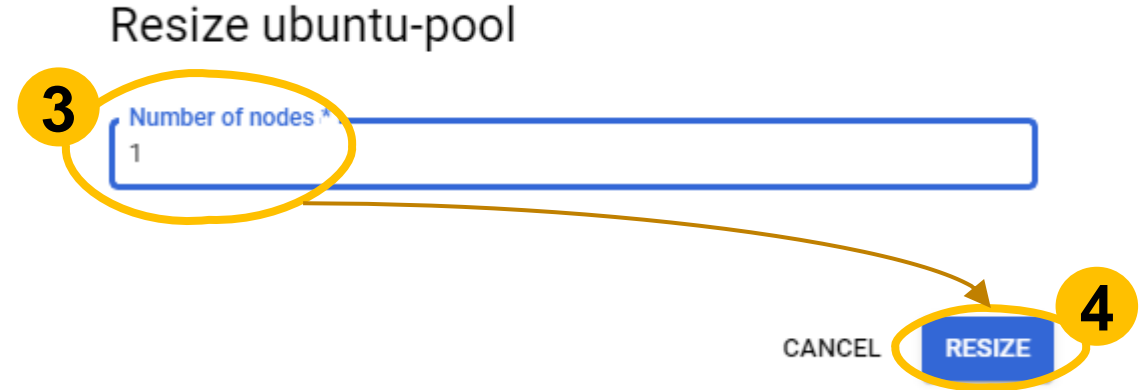
Size

Number of nodes	0
Autoscaling	Off
Node zones	asia-east1-b

There are 0 nodes in our cluster

Lab 0-2: Spin-up GKE Worker Instances

5. Input “1” as our target number of nodes
6. Then click “Resize”



Lab 0-2: Spin-up GKE Worker Instances

- ❑ Your node pool should be resized briefly

🔄 ubuntu-pool

i Resizing the node pool.
The values shown below will be updated once the operation finishes.

Cluster	trainer-lab-cluster
Node version	1.21.5-gke.1805

Size

Number of nodes	0
Autoscaling	Off
Node zones	asia-east1-b



✅ ubuntu-pool

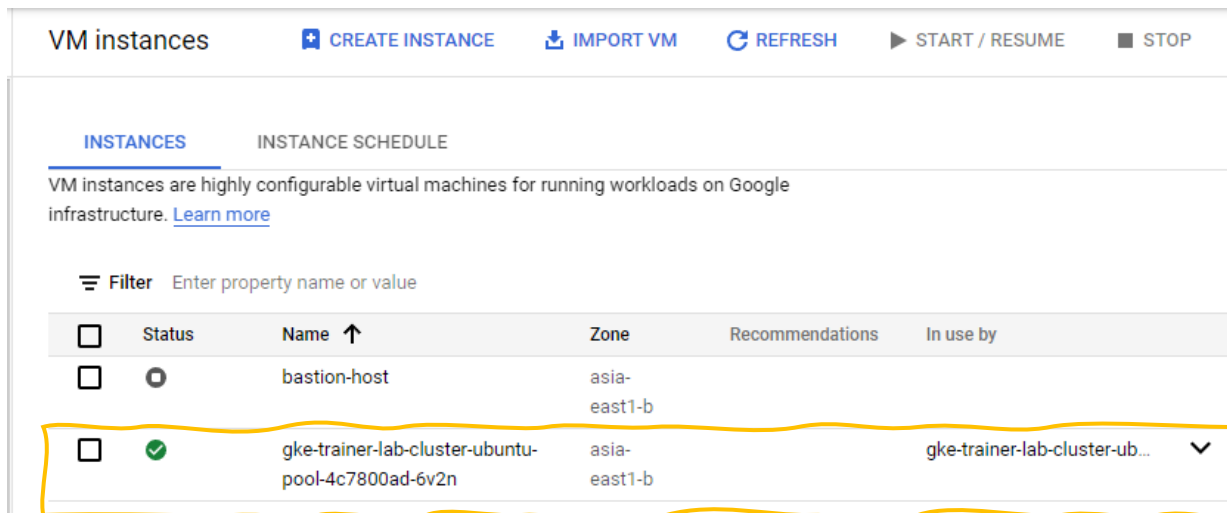
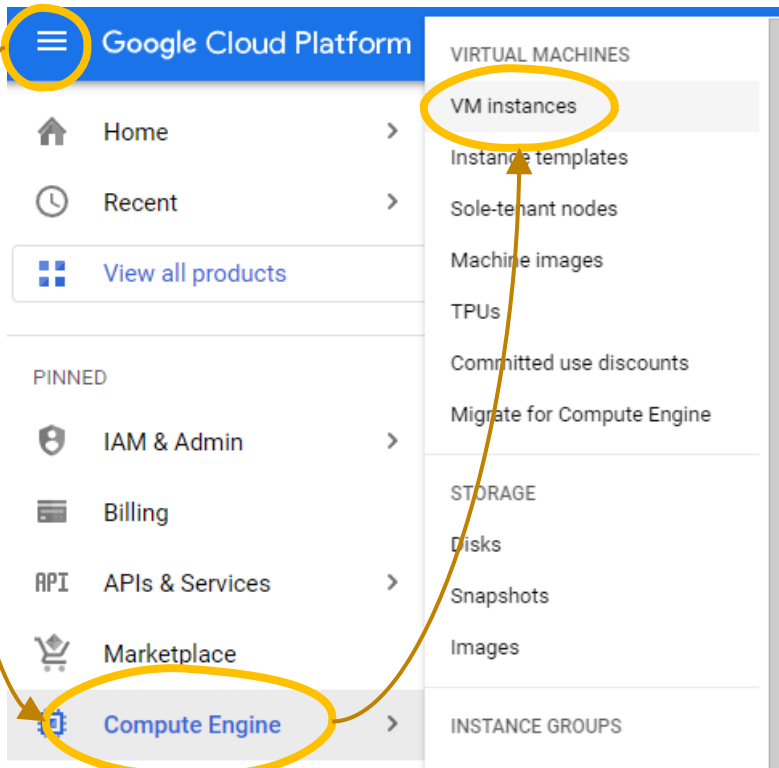
Cluster	trainer-lab-cluster
Node version	1.21.5-gke.1805

Size

Number of nodes	1
Autoscaling	Off
Node zones	asia-east1-b

Lab 0-2: Spin-up GKE Worker Instances

- ❑ A VM instance will be spin-up under your project



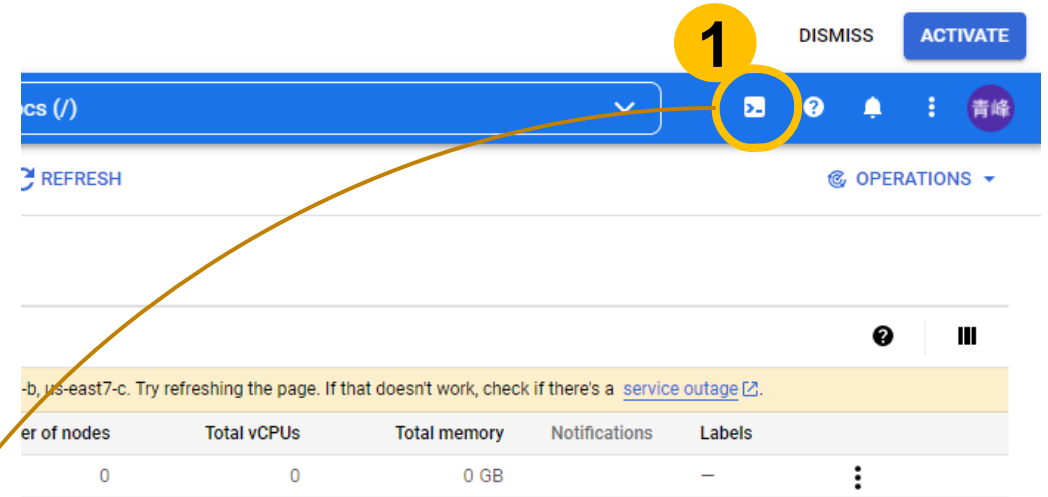
Lab 0: Preparation

1. Open Google Cloud Console
2. Spin-up GKE Worker Instances
3.  Open Cloud Shell

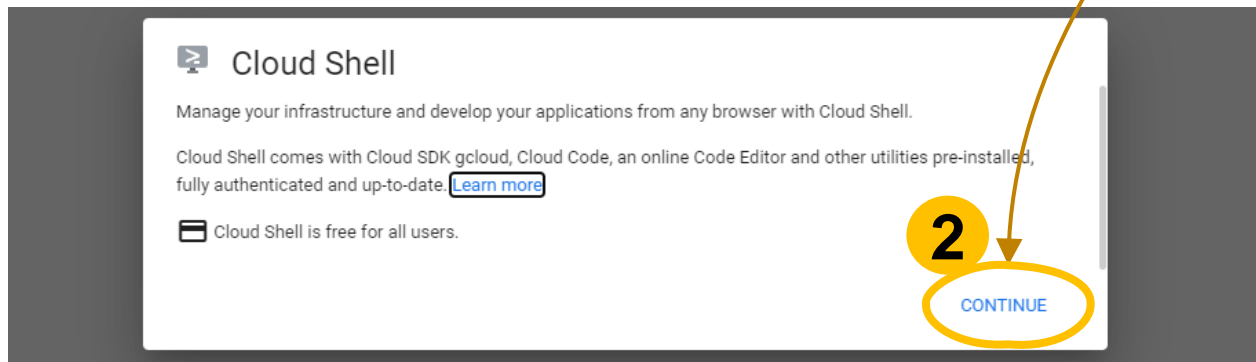
 Use Google Cloud Shell to access our K8S cluster

Lab 0-3: Open Cloud Shell

1. On the upper-right corner of the webpage, click the icon to activate cloud shell



2. Click “CONTINUE”



Lab 0-3: Open Cloud Shell

3. Set our project/region/zone:

```
>_ gcloud config set project nycu-lab-xx  
gcloud config set compute/zone asia-east1-b  
gcloud config set compute/region asia-east1
```

➤ Result:

```
chh9513136@cloudshell:~ (trainer-lab)$ gcloud config set project trainer-lab  
Updated property [core/project].  
chh9513136@cloudshell:~ (trainer-lab)$ gcloud config set compute/zone asia-east1-b  
Updated property [compute/zone].  
chh9513136@cloudshell:~ (trainer-lab)$ gcloud config set compute/region asia-east1  
Updated property [compute/region].  
chh9513136@cloudshell:~ (trainer-lab)$
```

Lab 0-3: Open Cloud Shell

□ We can list our K8s clusters:

```
chh9513136@cloudshell:~ (trainer-lab)$ gcloud container clusters list
WARNING: The following zones did not respond: us-east7-c, us-east7-a, us-east7, us-east7-b. List results may be incomplete.
NAME: trainer-lab-cluster
LOCATION: asia-east1-b
MASTER_VERSION: 1.21.5-gke.1805
MASTER_IP: 35.229.168.164
MACHINE_TYPE: e2-standard-4
NODE_VERSION: 1.21.5-gke.1805
NUM_NODES: 1
STATUS: RUNNING
chh9513136@cloudshell:~ (trainer-lab)$
```

4. Get the credential of the K8s cluster

➤ `gcloud container clusters get-credentials YOUR_CLUSTER_NAME`

➤ Result:

```
chh9513136@cloudshell:~ (trainer-lab)$ gcloud container clusters get-credentials trainer-lab-cluster
Fetching cluster endpoint and auth data.
kubeconfig entry generated for trainer-lab-cluster.
```

Lab 0-3: Open Cloud Shell

5. Verify you can get the cluster information

> `kubectl cluster-info`

```
chh9513136@cloudshell:~ (trainer-lab)$ kubectl cluster-info
Kubernetes control plane is running at https://35.229.168.164
GLBCDefaultBackend is running at https://35.229.168.164/api/v1/namespaces/kube-system/services/default-http-backend:http/proxy
KubeDNS is running at https://35.229.168.164/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
Metrics-server is running at https://35.229.168.164/api/v1/namespaces/kube-system/services/https:metrics-server:/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

6. Try to list our K8S worker nodes

> `kubectl get nodes`


The worker instance created in Lab 0-2

```
chh9513136@cloudshell:~ (trainer-lab)$ kubectl get nodes
NAME                                STATUS    ROLES    AGE    VERSION
gke-trainer-lab-cluster-ubuntu-pool-4c7800ad-1z7t  Ready    <none>   3m56s  v1.21.5-gke.1805
chh9513136@cloudshell:~ (trainer-lab)$
```


Lab 1: Create a *Pod*

1. Create a K8S Namespace
2. Deploy a NGINX Pod
3. Port-Forwarding through Cloud Shell
4. Cleanup

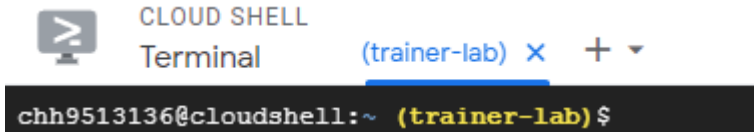
Lab 1: Create a *Pod*

1.  **Create a K8S Namespace**
2. Deploy a NGINX Pod
3. Port-Forwarding through Cloud Shell
4. Cleanup

 **Create a new namespace for our lab environment**

Lab 1-1: Create a K8S Namespace

1. Finish Lab 0, and navigate to Cloud Shell



A screenshot of the Cloud Shell terminal interface. At the top, it says 'CLOUD SHELL' and 'Terminal'. Below that, there's a tab labeled '(trainer-lab)' with a close button 'x' and a plus sign '+'. The terminal prompt shows the user 'chh9513136' at 'cloudshell:~' in the '(trainer-lab)' namespace, followed by a dollar sign '\$'.

2. List K8s namespaces

>_ `kubectl get namespaces`

```
chh9513136@cloudshell:~/git/nycu-k8s-lab/lab1 (trainer-lab)$ kubectl get namespaces
```

NAME	STATUS	AGE
default	Active	2d8h
kube-node-lease	Active	2d8h
kube-public	Active	2d8h
kube-system	Active	2d8h

Lab 1-1: Create a K8S Namespace

3. Create a K8S namespace: “lab”

```
>_ kubectl create namespace lab
```

```
chh9513136@cloudshell:~/git/nycu-k8s-lab/lab1 (trainer-lab)$ kubectl create namespace lab  
namespace/lab created
```

4. List K8s namespaces

```
>_ kubectl get namespaces
```

```
chh9513136@cloudshell:~/git/nycu-k8s-lab/lab1 (trainer-lab)$ kubectl get namespaces  
NAME                STATUS    AGE  
default              Active    2d8h  
kube-node-lease      Active    2d8h  
kube-public          Active    2d8h  
kube-system          Active    2d8h  
lab                  Active    25s
```

5. Set the “lab” namespace as default for kubectl context

```
>_ kubectl config set-context -current --namespace=lab
```

```
chh9513136@cloudshell:~/git/nycu-k8s-lab/lab2 (trainer-lab)$ kubectl config set-context --current --namespace=lab  
Context "gke_trainer-lab_asia-east1-b_trainer-lab-cluster" modified.
```

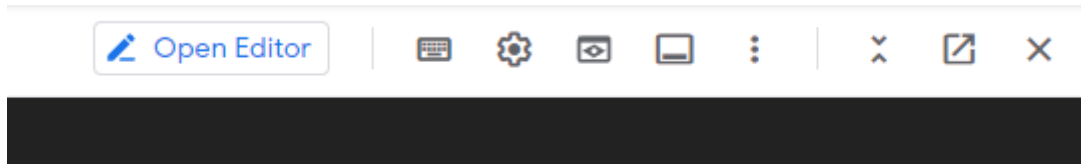
Lab 1: Create a *Pod*

1. Create a K8S Namespace
2.  **Deploy a NGINX Pod**
3. Port-Forwarding through Cloud Shell
4. Cleanup

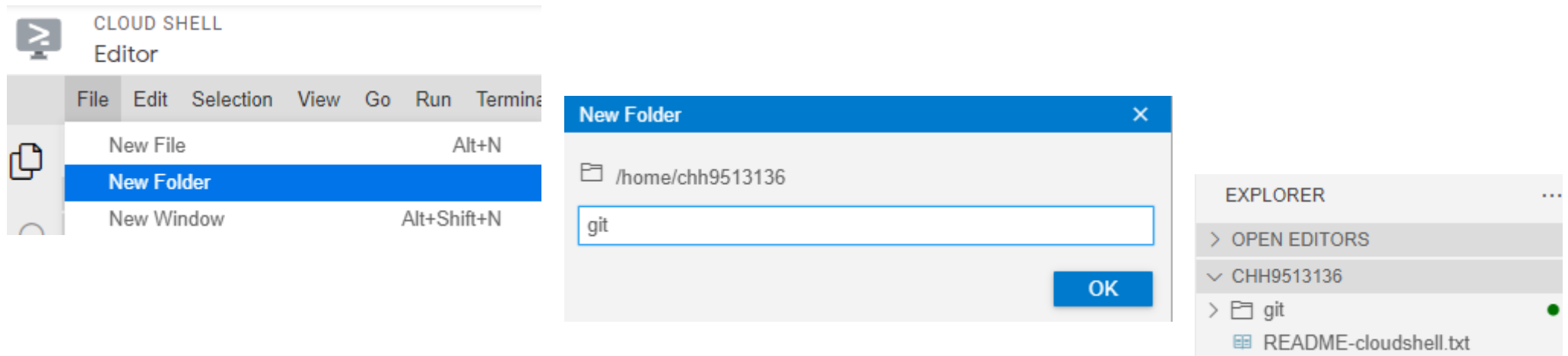
 Learn how to deploy a simple Pod

Lab 1-2: Deploy a NGINX Pod

1. Open editor on your cloud shell

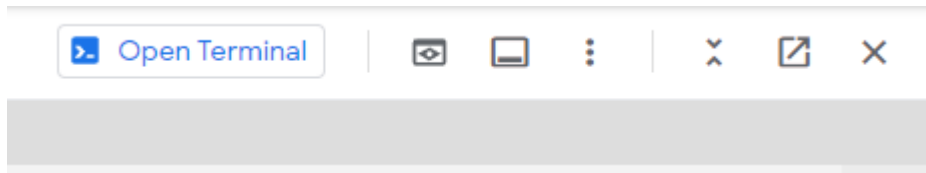


2. Create a new folder named “git”



Lab 1-2: Deploy a NGINX Pod

3. Go back to Terminal mode



4. Change directory, clone the lab repo

```
>_ cd ~/git
```

```
>_ git clone https://github.com/truffles/nycu-k8s-lab.git
```

```
>_ cd nycu-k8s-lab
```

```
>_ ls
```

```
chh9513136@cloudshell:~/git/nycu-k8s-lab (trainer-lab)$ ls  
lab1 lab2 lab3 lab-bonus README.md
```

Lab 1-2: Deploy a NGINX Pod

5. Change directory into “lab1”

```
>_ cd lab1
```

```
chh9513136@cloudshell:~/git/nycu-k8s-lab (trainer-lab)$ cd lab1  
chh9513136@cloudshell:~/git/nycu-k8s-lab/lab1 (trainer-lab)$
```

6. Deploy the Pod

```
>_ kubectl apply -f pod-nginx.yaml
```

```
chh9513136@cloudshell:~/git/nycu-k8s-lab/lab1 (trainer-lab)$ kubectl apply -f pod-nginx.yaml  
pod/nginx-standalone-pod created
```

7. List the Pods

```
>_ kubectl get pod
```

```
chh9513136@cloudshell:~/git/nycu-k8s-lab/lab1 (trainer-lab)$ kubectl get pod  
NAME                READY   STATUS    RESTARTS   AGE  
nginx-standalone-pod 1/1     Running   0           3s
```

- ❑ You can view the content of the “pod-nginx.yaml” file in the Editor mode.
- ❑ What’s in the YAML file?

Lab 1-2: Deploy a NGINX Pod

- ❑ You can open a bash shell into the pod we just deployed
- ❑ Have a look around 😊

```
chh9513136@cloudshell:~/git/nycu-k8s-lab/lab1 (trainer-lab)$ kubectl exec -it pod/nginx-standalone-pod -- bash
root@nginx-standalone-pod:/# ls
bin boot dev docker-entrypoint.d docker-entrypoint.sh etc home lib lib64 media mnt opt proc root run sbin srv sys tmp usr var
root@nginx-standalone-pod:/#
```

- ❑ To exit, type “exit”

```
root@nginx-standalone-pod:/# exit
exit
chh9513136@cloudshell:~/git/nycu-k8s-lab/lab1 (trainer-lab)$
```

Lab 1: Create a *Pod*

1. Create a K8S Namespace
2. Deploy a NGINX Pod
3.  **Port-Forwarding through Cloud Shell**
4. Cleanup

 Learn how to use port-forwarding to troubleshoot a pod

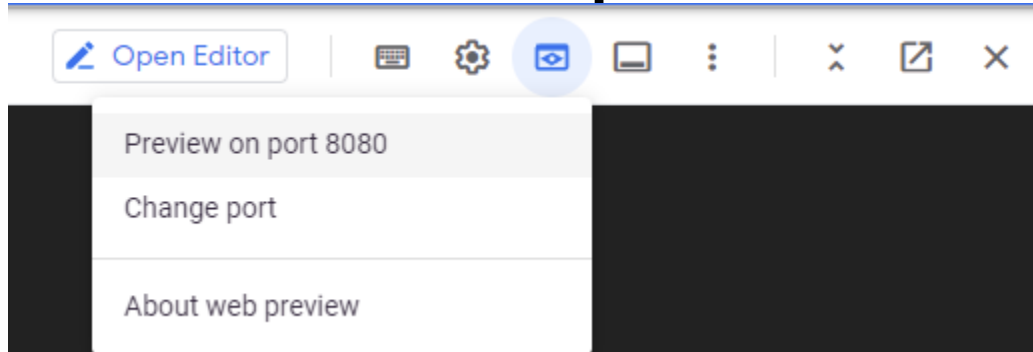
Lab 1-3: Port-Forwarding through Cloud Shell

1. Enable port-forwarding for the nginx pod

```
>_ kubectl port-forward pod/nginx-standalone-pod 8080:80
```

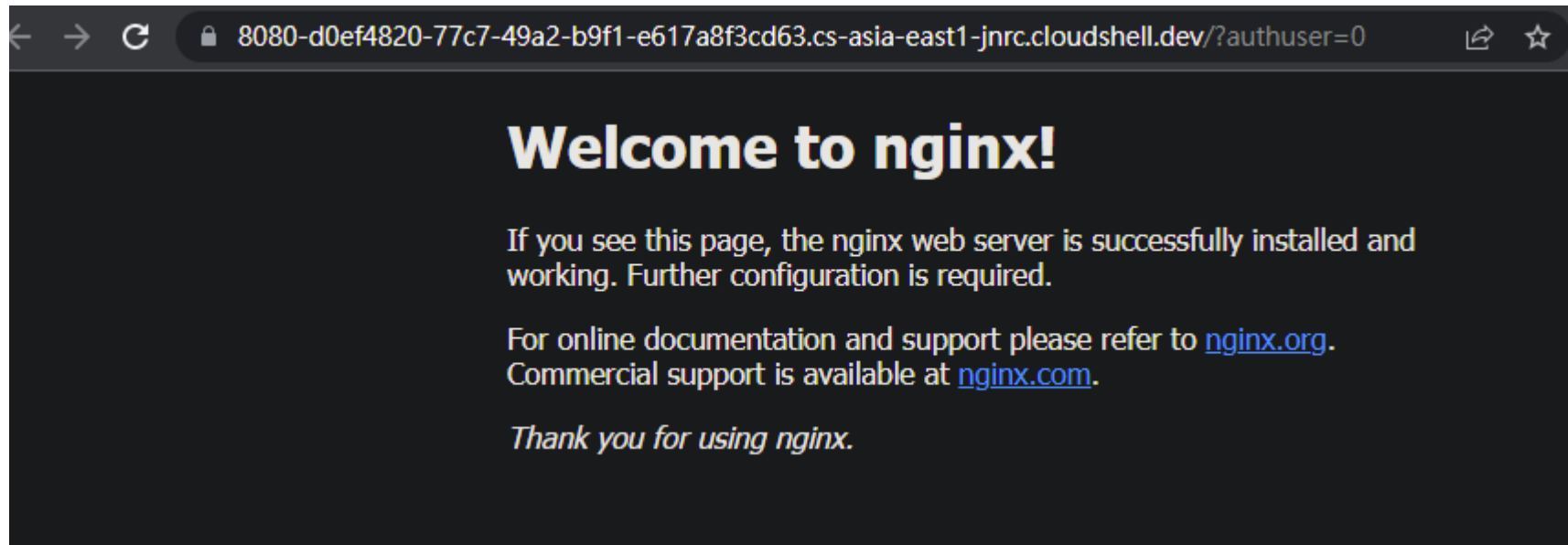
```
chh9513136@cloudshell:~/git/nycu-k8s-lab/lab1 (trainer-lab)$ kubectl port-forward pod/nginx-standalone-pod 8080:80  
Forwarding from 127.0.0.1:8080 -> 80
```

2. Click “Preview on port 8080”



Lab 1-3: Port-Forwarding through Cloud Shell

3. A new browser tab will open, showing the default webpage:



Lab 1: Create a *Pod*

1. Create a K8S Namespace
2. Deploy a NGINX Pod
3. Port-Forwarding through Cloud Shell
4.  **Cleanup**

Lab 1-4: Cleanup

1. End the port forwarding session by “Ctrl+C” on the terminal

```
chh9513136@cloudshell:~/git/nycu-k8s-lab/lab1 (trainer-lab)$ kubectl port-forward pod/nginx-standalone-pod 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Handling connection for 8080
^Cchh9513136@cloudshell:~/git/nycu-k8s-lab/lab1 (trainer-lab)$
```

2. Delete the pod


```
>_ kubectl delete pod nginx-standalone-pod
```

```
chh9513136@cloudshell:~/git/nycu-k8s-lab/lab1 (trainer-lab)$ kubectl delete pod nginx-standalone-pod
pod "nginx-standalone-pod" deleted
```

Lab 2: Expose Pods with *Service* & *Ingress*

1. Deploy application pods
2. Deploy a Service
3. Deploy an Ingress

Lab 2: Expose Pods with *Service* & *Ingress*

1.  **Deploy Application Pods**
2. Deploy a Service
3. Deploy an Ingress

- ❑ Setup 1 bastion pod & 2 nginx pods for this lab
- ❑ Access the NGINX pods with their pod IPs

Lab 2-1: Deploy Application Pods

1. Change directory to “lab2”

```
>_ cd lab2
```

```
chh9513136@cloudshell:~/git/nycu-k8s-lab/lab1 (trainer-lab)$ cd ../lab2  
chh9513136@cloudshell:~/git/nycu-k8s-lab/lab2 (trainer-lab)$
```

2. Deploy a bastion pod

```
>_ kubectl apply -f pod-bastion.yaml
```

```
chh9513136@cloudshell:~/git/nycu-k8s-lab/lab2 (trainer-lab)$ kubectl apply -f pod-bastion.yaml  
pod/bastion created
```

3. Deploy 2 nginx pods

```
>_ kubectl apply -f pod-nginx-1.yaml -f pod-nginx-2.yaml
```

```
chh9513136@cloudshell:~/git/nycu-k8s-lab/lab2 (trainer-lab)$ kubectl apply -f pod-nginx-1.yaml -f pod-nginx-2.yaml  
pod/nginx-1 created  
pod/nginx-2 created
```

Lab 2-1: Deploy Application Pods

4. List all pods, observe the labels

> `kubectl get pod --show-labels`

```
chh9513136@cloudshell:~/git/nycu-k8s-lab/lab2 (trainer-lab)$ kubectl get pod --show-labels
```

NAME	READY	STATUS	RESTARTS	AGE	LABELS
bastion	1/1	Running	0	17m	<none>
nginx-1	1/1	Running	0	17m	app=nginx
nginx-2	1/1	Running	0	17m	app=nginx

5. List all pods with IP & node information, note the IPs

> `kubectl get pod -o wide`

```
chh9513136@cloudshell:~/git/nycu-k8s-lab/lab2 (trainer-lab)$ kubectl get pod -o wide
```

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
bastion	1/1	Running	0	18m	192.199.0.16	gke-trainer-lab-cluster-ubuntu-pool-4c7800ad-fgbw
nginx-1	1/1	Running	0	17m	192.199.0.17	gke-trainer-lab-cluster-ubuntu-pool-4c7800ad-fgbw
nginx-2	1/1	Running	0	17m	192.199.0.18	gke-trainer-lab-cluster-ubuntu-pool-4c7800ad-fgbw

Lab 2-1: Deploy Application Pods

6. Open a new terminal tab (“Tab 2”)



7. (“Tab 2”) Run a bash shell inside the bastion pod

```
>_ kubectl exec -it pod/bastion -- bash
```

```
chh9513136@cloudshell:~ (trainer-lab)$ kubectl exec -it pod/bastion -- bash  
[root@bastion /]#
```

8. (“Tab 2”) Connect to the NGINX pods by their IP

```
>_ curl YOUR_NGINX_POD_IP
```

```
[root@bastion /]# curl 192.199.0.17  
<!DOCTYPE html>  
<html>  
<head>  
<title>Welcome to nginx!</title>  
<style>  
html { color-scheme: light dark; }  
body { width: 35em; margin: 0 auto;  
font-family: Tahoma, Verdana, Arial, sans-serif; }  
</style>  
</head>  
<body>  
<h1>Welcome to nginx!</h1>  
<p>If you see this page, the nginx web server is successfully installed and  
working. Further configuration is required.</p>  
  
<p>For online documentation and support please refer to  
<a href="http://nginx.org/">nginx.org</a>.<br/>  
Commercial support is available at  
<a href="http://nginx.com/">nginx.com</a>.</p>  
  
<p><em>Thank you for using nginx.</em></p>  
</body>  
</html>  
[root@bastion /]#
```

Lab 2: Expose Pods with *Service* & *Ingress*

1. Deploy Application Pods
2. **▶ Deploy a Service**
3. Deploy an Ingress

▣ Use service as an in-cluster load-balancer for the pods

Lab 2-2: Deploy a Service

1. (“Tab 1”) Deploy the service YAML

```
>_ kubectl apply -f service-nginx.yaml
```

```
chh9513136@cloudshell:~/git/nycu-k8s-lab/lab2 (trainer-lab)$ kubectl apply -f service-nginx.yaml
service/nginx created
```

2. (“Tab 1”) List the services, note the IP addresses

```
>_ kubectl get service
```

```
chh9513136@cloudshell:~/git/nycu-k8s-lab/lab2 (trainer-lab)$ kubectl get service
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
nginx	NodePort	192.199.13.125	<none>	80:31226/TCP	43m

3. (“Tab 1”) List the endpoints

```
>_ kubectl get endpoints
```

```
chh9513136@cloudshell:~/git/nycu-k8s-lab/lab2 (trainer-lab)$ kubectl get endpoints
```

NAME	ENDPOINTS	AGE
nginx	192.199.0.17:80,192.199.0.18:80	2m56s

Lab 2-2: Deploy a Service

4. (“Tab 2”) Access the service via it’s IP address

```
>_ curl YOUR_NGINX_SERVICE_CLUSTER_IP
```

```
[root@bastion /]# curl 192.199.13.125
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
```


5. (“Tab 2”) Access the service using it’s service name

```
>_ curl nginx
```

```
[root@bastion /]# curl nginx
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
```

- ❑ What’s the benefit of access web application via **Service** over using Pod IP address directly?
 - ❑ Try to delete one of the pods, is it still accessible via **Service**?
 - ❑ What if two pods under the same **Service** have different webpages?

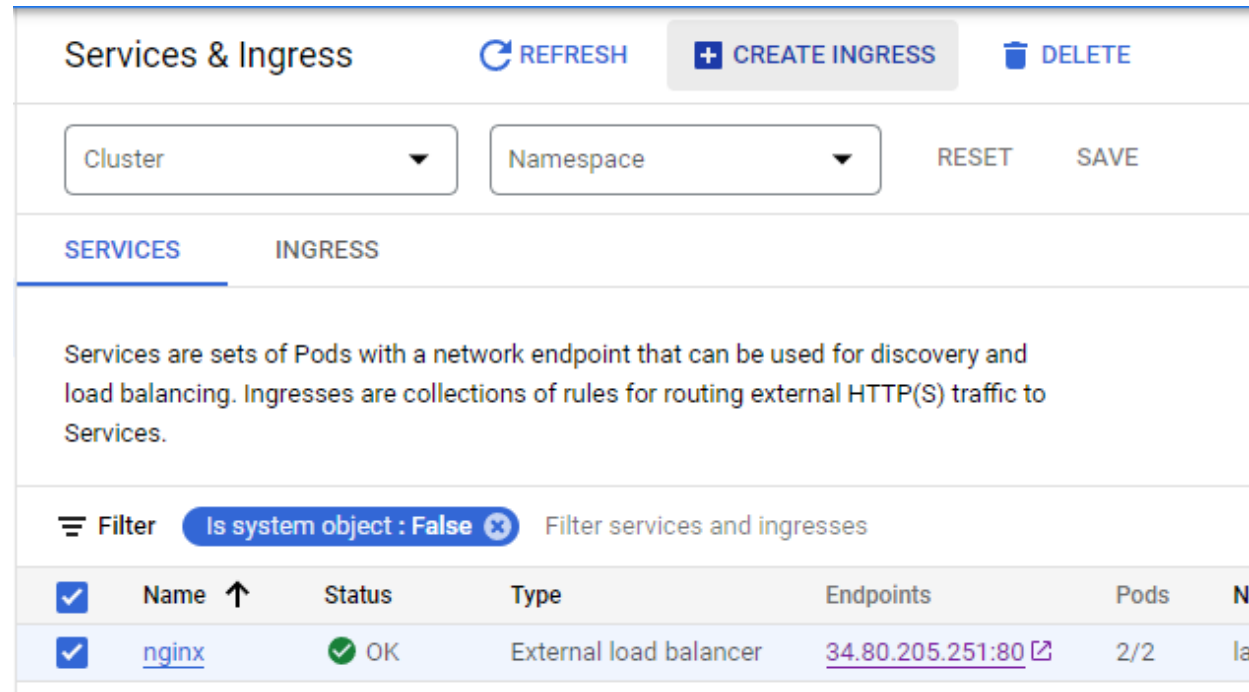
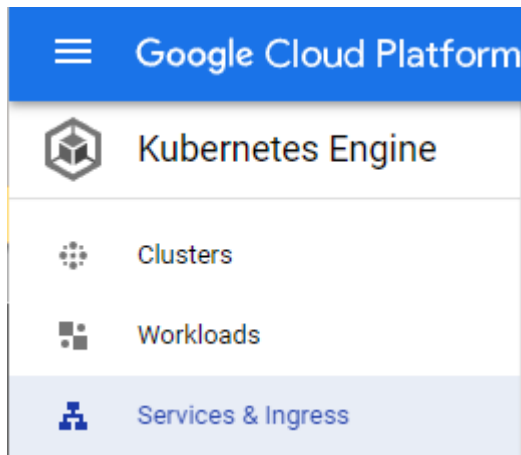
Lab 2: Expose Pods with *Service* & *Ingress*

1. Deploy Application Pods
2. Deploy a Service
3.  **Deploy an Ingress**

- ❑ How to let everyone outside K8S cluster connect to our webpage with specific forwarding rules?

Lab 2-3: Deploy an Ingress

1. Navigate to the “Services & Ingress” page
2. Check “nginx” service and click “create ingress” button



Lab 2-3: Deploy an Ingress

3. Fill in the ingress name & default backends, then hit “create”

[←](#) Create a Kubernetes Ingress

An Ingress has 3 parts:

- Backend configuration
- Host and path rules configuration
- Forwarding rules

Ingress type

External HTTP/S load balancer

Name *

nginx

✓

Backend configuration

You have configured 1 backend

✓

Host and path rules

You have configured 1 rule

✓

Frontend configuration

Configured HTTP

4

Preview setup and YAML (optional)

Preview the Ingress

CREATE

CANCEL

Host and path rule configuration

Host and path rules determine how your traffic will be directed. You can direct traffic to a backend kubernetes service. Any traffic not explicitly matched with a host and path rule will be sent to the default service selected on the first row.

Hosts 1

Paths 1

Backends 1

nginx

+ ADD HOST AND PATH RULE

Lab 2-3: Deploy an Ingress

4. Wait for it to be ready

← Ingress Details REFRESH EDIT DELETE KUBECTL

Last refreshed: 1 minute ago

✓ nginx

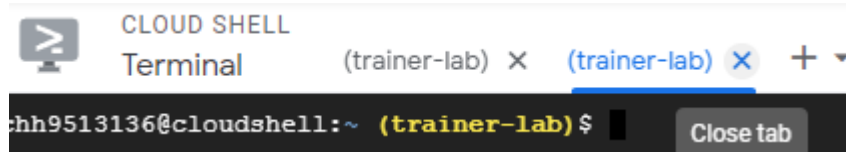
DETAILS	TELEMETRY	LOGS	EVENTS	YAML
Cluster	trainer-lab-cluster			
Namespace	lab			
Created	Mar 16, 2022, 11:30:52 AM			
Labels	No labels set			
Annotations	ingress.kubernetes.io/backends: {"k8s1-d8a3ae2a-lab-nginx-80-7016d561": "HEALTHY"} ingress.kubernetes.io/forwarding-rule: k8s2-fr-vflv7ku7-lab-nginx-zwm7z9h2 ingress.kubernetes.io/target-proxy: k8s2-tp-vflv7ku7-lab-nginx-zwm7z9h2 ingress.kubernetes.io/url-map: k8s2-um-vflv7ku7-lab-nginx-zwm7z9h2			
Type	Ingress			
Load balancer	External HTTP(S) LB			
IP address	34.111.151.102			

Routes

5. Access the webpage on your browser via the provided IP address

Lab 2-4: Cleanup

1. Close the 2nd terminal tab



2. Delete all ingresses

```
>_ kubectl delete ingress
```

3. Delete all services

```
>_ kubectl delete service
```

4. Delete all pods

```
>_ kubectl delete pod
```

Lab 3: Use *deployments* to manage Pods

1. Deploy the “Deployment” YAML
2. Change the number of replicas
3. What if we delete pods manually
4. Cleanup

Lab 3: Use *deployments* to manage Pods

1.  Deploy the “Deployment” YAML
2. Change the number of replicas
3. What if we delete the pods manually
4. Cleanup

Lab 3-1: Deploy the “Deployment” YAML

1. Change directory to “lab3”

>_ `cd lab3`

```
chh9513136@cloudshell:~/git/nycu-k8s-lab/lab2 (trainer-lab)$ cd ../lab3  
chh9513136@cloudshell:~/git/nycu-k8s-lab/lab3 (trainer-lab)$
```

2. Deploy the NGINX deployment YAML

>_ `kubectl apply -f deployment-nginx.yaml`

```
chh9513136@cloudshell:~/git/nycu-k8s-lab/lab3 (trainer-lab)$ kubectl apply -f deployment-nginx.yaml  
deployment.apps/nginx created
```

Lab 3-1: Deploy the “Deployment” YAML

3. Observe all the deployed resources

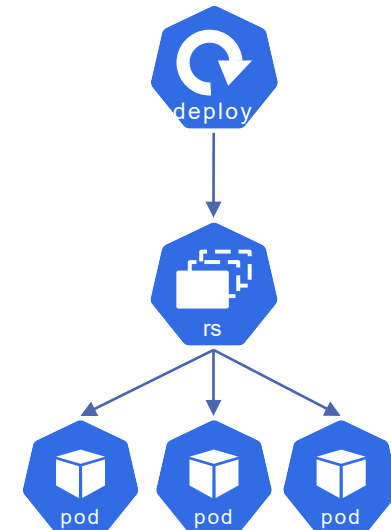
> `kubectl get all`

```
chh9513136@cloudshell:~/git/nycu-k8s-lab/lab3 (trainer-lab)$ kubectl get all
```

NAME	READY	STATUS	RESTARTS	AGE
pod/nginx-585449566-kjkdd	1/1	Running	0	103s
pod/nginx-585449566-ndqm9	1/1	Running	0	103s
pod/nginx-585449566-rsxs6	1/1	Running	0	104s

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/nginx	3/3	3	3	104s

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/nginx-585449566	3	3	3	105s



Lab 3: Use *deployments* to manage Pods

1. Deploy the “Deployment” YAML
2. **▶ Change the number of replicas**
3. What if we delete the pods manually
4. Cleanup

❑ Try to change the number of replicas...

- ❑ using “kubectl scale” command
- ❑ by re-applying the YAML definition

Lab 3-2: Change the number of replicas

1. Scale it using “kubectl scale” command

```
>_ kubectl scale deployment/nginx --replicas=2
```

```
chh9513136@cloudshell:~/git/nycu-k8s-lab/lab3 (trainer-lab)$ kubectl scale deployment/nginx --replicas=2  
deployment.apps/nginx scaled
```

2. Observe the pods

```
>_ kubectl get pod
```

```
chh9513136@cloudshell:~/git/nycu-k8s-lab/lab3 (trainer-lab)$ kubectl get pod
```

NAME	READY	STATUS	RESTARTS	AGE
nginx-585449566-kjkdd	1/1	Running	0	24m
nginx-585449566-ndqm9	1/1	Running	0	24m

Lab 3-2: Change the number of replicas

3. Edit number of replicas in the “deployment-nginx.yaml” file

```
deployment-nginx.yaml x
git > nycu-k8s-lab > lab3 > deployment-nginx.yaml > {} spec > replicas
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: nginx
5    labels:
6      app: nginx
7  spec:
8    replicas: 4
9    selector:
10     matchLabels:
11       app: nginx
```

4. Apply the YAML again


```
>_ kubectl apply -f deployment-nginx.yaml
```

```
chh9513136@cloudshell:~/git/nycu-k8s-lab/lab3 (trainer-lab)$ kubectl apply -f deployment-nginx.yaml
deployment.apps/nginx configured
```

5. Observe the pods

```
>_ kubectl get pod
```

Lab 3: Use *deployments* to manage Pods

1. Deploy the “Deployment” YAML
2. Change the number of replicas
3.  **What if we delete the pods manually**
4. Cleanup

Lab 3-3: What if we delete the pods manually

1. Observe the pods first

```
>_ kubectl get pod
```

```
chh9513136@cloudshell:~/git/nycu-k8s-lab/lab3 (trainer-lab)$ kubectl get pod
NAME                                READY   STATUS    RESTARTS   AGE
nginx-585449566-6cpn4               1/1     Running   0           6m56s
nginx-585449566-kjkdd               1/1     Running   0           35m
nginx-585449566-ndqmq               1/1     Running   0           35m
nginx-585449566-q6ctq               1/1     Running   0           6m56s
```

2. Let's delete one of the pods

```
>_ kubectl delete pod YOUR_POD_NAME
```

```
chh9513136@cloudshell:~/git/nycu-k8s-lab/lab3 (trainer-lab)$ kubectl delete pod nginx-585449566-kjkdd
pod "nginx-585449566-kjkdd" deleted
```

3. Observe the pods again. What happened?

```
>_ kubectl get pod
```

Lab 3: Use *deployments* to manage Pods

1. Deploy the “Deployment” YAML
2. Change the number of replicas
3. What if we delete the pods manually
4.  **Cleanup**

Lab 3-4: Cleanup

1. Delete the “nginx” deployment

```
>_ kubectl delete deployment nginx
```

```
chh9513136@cloudshell:~/git/nycu-k8s-lab/lab3 (trainer-lab)$ kubectl delete deployment nginx  
deployment.apps "nginx" deleted
```

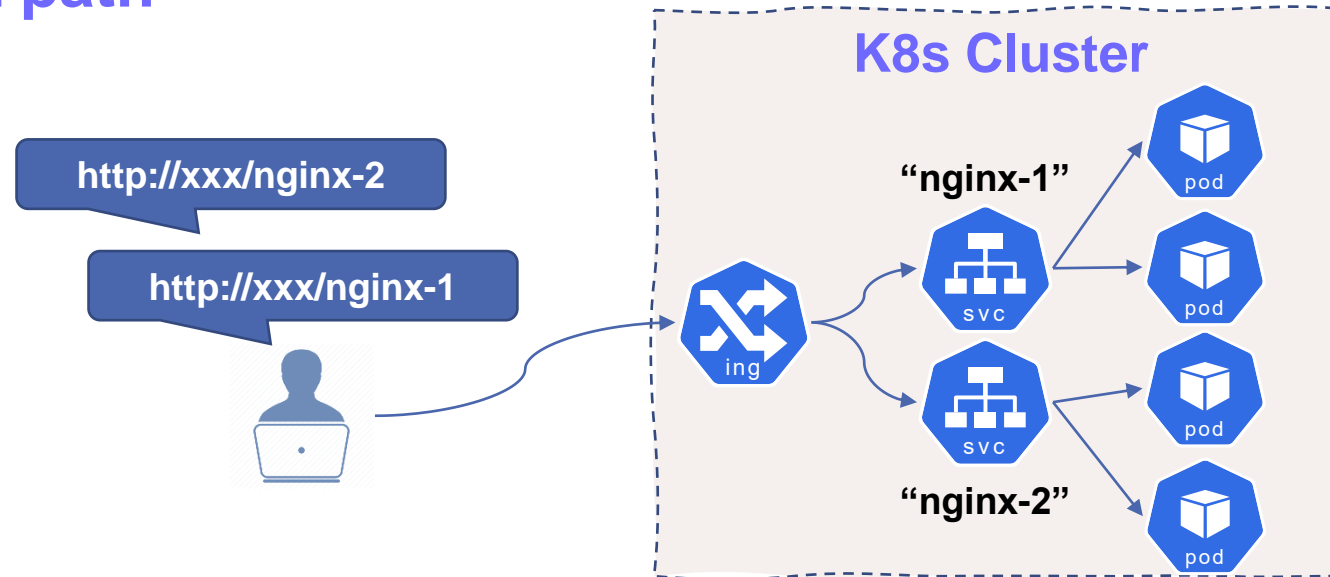
2. Get all the resources

```
>_ kubectl get all
```

```
chh9513136@cloudshell:~/git/nycu-k8s-lab/lab3 (trainer-lab)$ kubectl get all  
No resources found in lab namespace.
```

Bonus Lab: Advanced Ingress Routing

1. Deploy the provided YAML files to create 2 deployments with different webpage content
2. For each deployment, create a **service** for it
3. Create an **ingress** to route to different service backends base on the **URL path**



⚠️ Reminder on Cloud Resource Usage Fees

- ❑ Running instances on Google Cloud Platform incur charges, please DO remember to *scale down / cleanup your instances* if you are not going to access it for a while 😊
- ❑ Remember to:
 - ✓ Scale down your cluster
 - ✓ Delete Ingresses & “LoadBalancer” type Services

Resize ubuntu-pool

Number of nodes *
0

CANCEL

RESIZE

