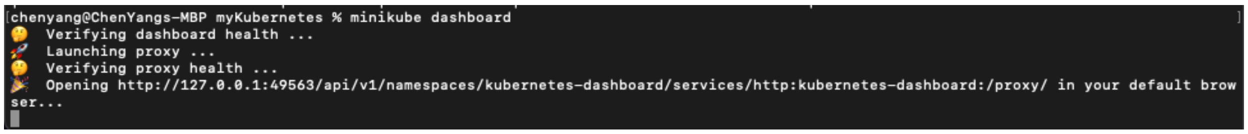
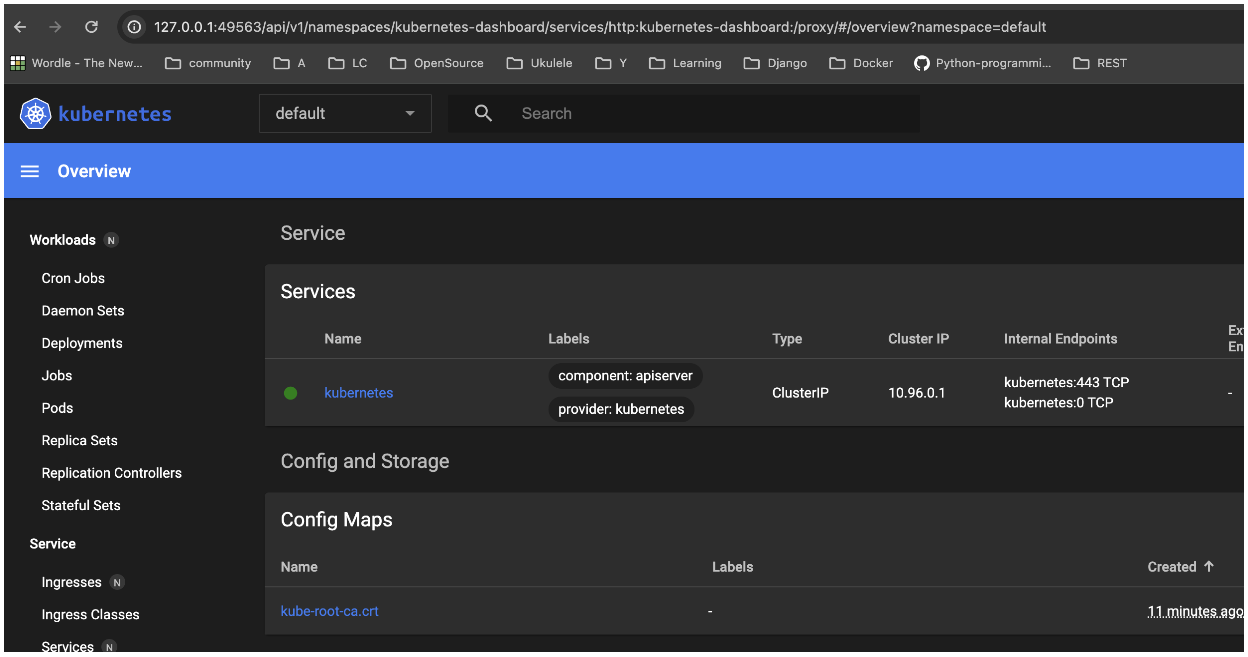
K8S (2) Kubernetes Dashboard/ Kubectl Proxy

K8S Dashboard is a web-based Kubernetes user interface. You can get overview of your cluster.
Even better, you can actually manage your resource in the cluster, i.e. Adding deployment
K8S in an add-on, default is disabled. Follow the command the check the current enable list and enable to dashboard:

```
$ minikube addons list
$ minikube addons enable metrics-server
$ minikube addons enable dashboard
$ minikube addons list
$ minikube dashboard
$ minikube dashboard --url
```



```
chenyang@ChenYangs-MBP myKubernetes % minikube dashboard
🤔  Verifying dashboard health ...
🚀  Launching proxy ...
🤔  Verifying proxy health ...
🎉  Opening http://127.0.0.1:49563/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/ in your default brow
ser...
```



Service

Services

	Name	Labels	Type	Cluster IP	Internal Endpoints	Ex En
●	kubernetes	component: apiserver provider: kubernetes	ClusterIP	10.96.0.1	kubernetes:443 TCP kubernetes:0 TCP	-

Config and Storage

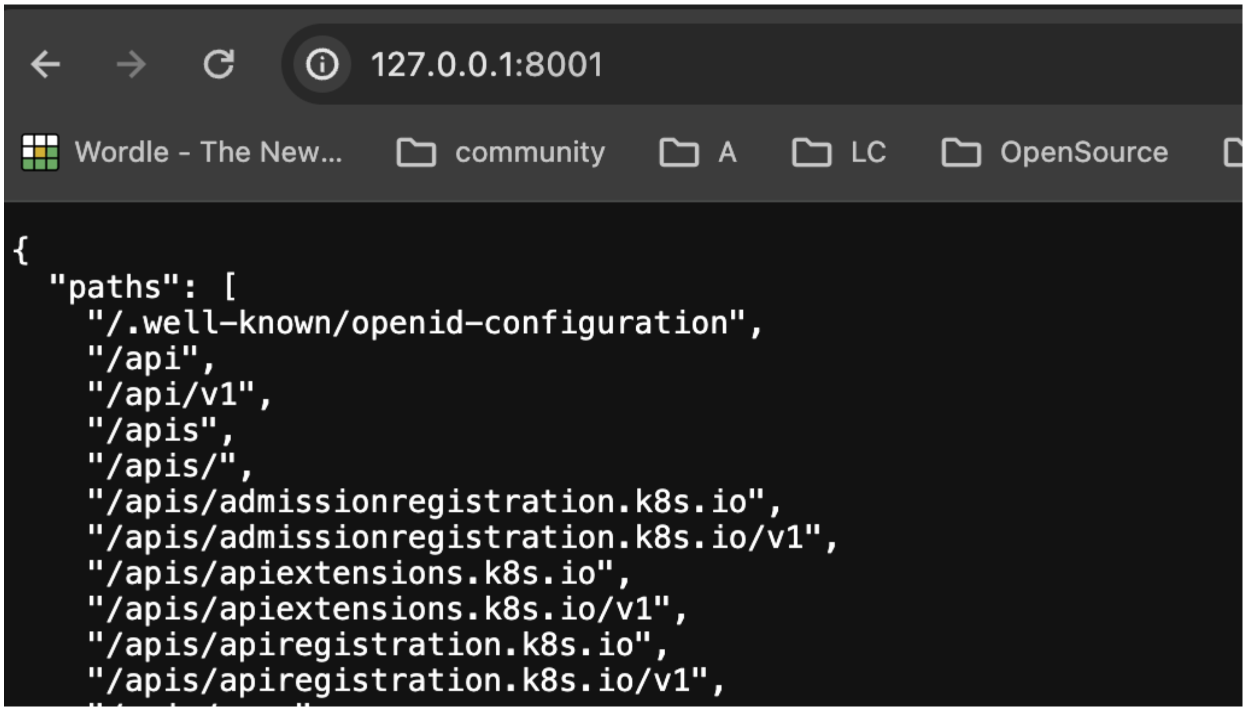
Config Maps

Name	Labels	Created ↑
kube-root-ca.crt	-	11 minutes ago

Kubectl proxy reveals the API server on control node

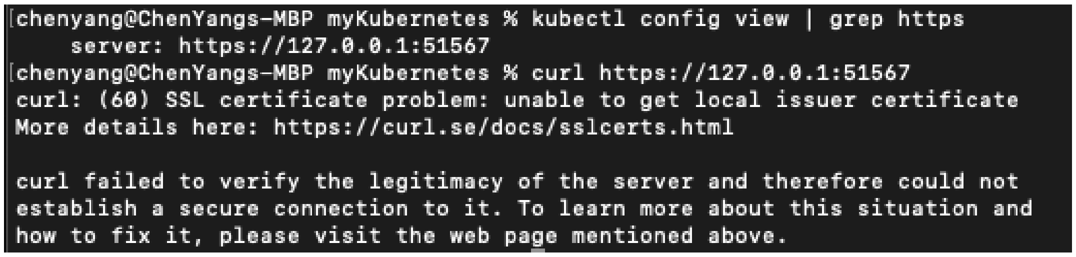
```
$kubectl proxy

Starting to serve on 127.0.0.1:8001

$kubectl proxy & // run in background

$ curl 127.0.0.1:8001
{
"paths": [
"/.well-known/openid-configuration",
"/api",
"/api/v1",
"/apis",
"/apis/",
"/apis/admissionregistration.k8s.io",
"/apis/admissionregistration.k8s.io/v1",
...
```

Or simply open your browser and visit
http://localhost:8001/
http://localhost:8001/api/v1
http://localhost:8001/healthz



```
{
  "paths": [
    "/.well-known/openid-configuration",
    "/api",
    "/api/v1",
    "/apis",
    "/apis/",
    "/apis/admissionregistration.k8s.io",
    "/apis/admissionregistration.k8s.io/v1",
    "/apis/apiextensions.k8s.io",
    "/apis/apiextensions.k8s.io/v1",
    "/apis/apiregistration.k8s.io",
    "/apis/apiregistration.k8s.io/v1",
```

We have gain the access by curl and browser, but this is because kubectl proxy expose the API server on the http://localhost:8001. The API server is running at different endpoint. If you stop the proxy, http://localhost:8001 is unreachable and the https://127.0.0.1:51567 will deny your access



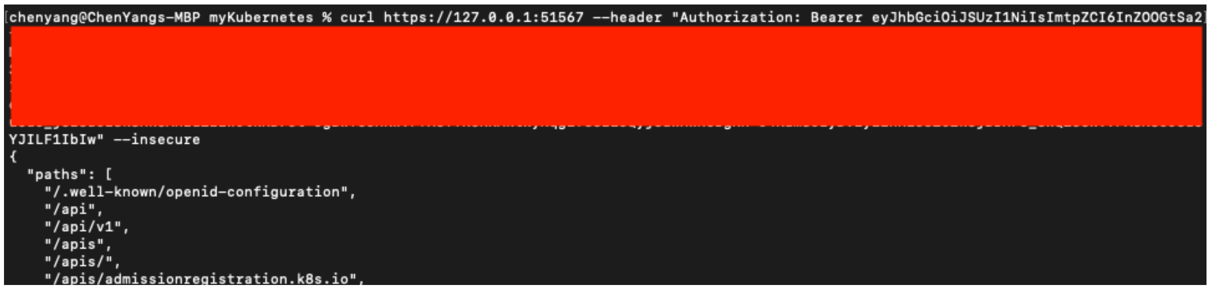
```
[chenyang@ChenYangs-MBP myKubernetes % kubectl config view | grep https
    server: https://127.0.0.1:51567
[chenyang@ChenYangs-MBP myKubernetes % curl https://127.0.0.1:51567
curl: (60) SSL certificate problem: unable to get local issuer certificate
More details here: https://curl.se/docs/sslcerts.html

curl failed to verify the legitimacy of the server and therefore could not
establish a secure connection to it. To learn more about this situation and
how to fix it, please visit the web page mentioned above.
```

We need to access the endpoint with identification.
API with authentication: such as (1) Identifying tokens (2) Keys + certificates

Demo how to get the bear token

We will discuss about the key+certificates when we talk about authN.

```
$ kubectl config view | grep https
server: https://127.0.0.1:51567

$ kubectl create token default
eyJhbGciOiJSU...JILF1IbIw

$ kubectl create clusterrole api-access-root --verb=get --non-resource-url=/*
// define a new role 'api-access-root' with GET permission
// $setopt noglob if you meet "zsh: no matches found", this is shell preventing you
using '?' '*' '^' in the url

$ kubectl create clusterrolebinding api-access-root --clusterrole api-access-root --
serviceaccount=default:default
// bind the sercice account to role 'api-access-root'

$ curl https://127.0.0.1:51567 --header "Authorization: Bearer
eyJhbGciOiJSU...JILF1IbIw" --insecure
```



```
[chenyang@ChenYangs-MBP myKubernetes % curl https://127.0.0.1:51567 --header "Authorization: Bearer eyJhbGciOiJSUzI1NiIsImtpZCI6InZOOGtSa2
[redacted]
YJILF1IbIw" --insecure
{
  "paths": [
    "/.well-known/openid-configuration",
    "/api",
    "/api/v1",
    "/apis",
    "/apis/",
    "/apis/admissionregistration.k8s.io",
```

Demo how to use Keys + certificates

We can use openssl to generate client key+client certificate and request authorition by minikube. Or use the default certificate generate by minikube
Notice the .crt and .key file must be base64 encoded.

```
$ kubectl config view
apiVersion: v1
clusters:
- cluster:
certificate-authority: /Users/chenyang/.minikube/ca.crt
...
- name: minikube
user:
client-certificate: /Users/chenyang/.minikube/profiles/minikube/client.crt
client-key: /Users/chenyang/.minikube/profiles/minikube/client.key

$ curl https://127.0.0.1:51567 --cert
/Users/chenyang/.minikube/profiles/minikube/client.crt --key
/Users/chenyang/.minikube/profiles/minikube/client.key --cacert
/Users/chenyang/.minikube/ca.crt
{
"paths": [
"/.well-known/openid-configuration",
"/api",
"/api/v1",
"/apis",
"/apis/",
"/apis/admissionregistration.k8s.io",
"/apis/admissionregistration.k8s.io/v1",
...
```

Appendix:
If you saw authentication request when you access the dashboard, check this and create SA/RoleBinding for the bear token
https://medium.com/learn-or-die/kubernetes-dashboard-%E4%BD%BF%E7%94%A8%E8%87%AA%E5%AE%9A%E7%BE%A9-service-

account-%E7%99%BB%E5%85%A5-b136669fff34
https://godleon.github.io/blog/Kubernetes/k8s-Deploy-and-Access-Dashboard/

```
# admin-user.yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: admin-user
  namespace: kubernetes-dashboard
```

```
# admin-user-role-binding.yaml
# The purpose of roleBinding is to associate the system cluster role (in this case: cluster-admin)
# and the service account user (admin-user we created in previous step).
# So our admin-user will have same permission with cluster-admin
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: admin-user
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin # cluster-admin is the already in k8s cluster, comes when you install your cluster. We can just refer to this role
subjects:
- kind: ServiceAccount
  name: admin-user
  namespace: kubernetes-dashboard
```

```
$ kubectl create -f admin-user.yaml -n kubernetes-dashboard
$ kubectl create -f admin-user-role-binding.yaml -n kubernetes-dashboard
$ kubectl -n kube-system get secret | grep admin-user // This will return the token
```