Exercise 13.3: Adding tools for monitoring and metrics

With the deprecation of **Heapster** the new, integrated **Metrics Server** has been further developed and deployed. The **Prometheus** project of CNCF.io has matured from incubation to graduation, is commonly used for collecting metrics, and should be considered as well.

Configure Metrics

Very Important

The `metrics-server` is written to interact with Docker. If you chose to use crio the logs will show errors and inability to collect metrics.

1. Begin by cloning the software. The **git** command should be installed already. Install it if not found.

```
student@cp:~$ git clone \
      https://github.com/kubernetes-incubator/metrics-server.git
```

```
<output_omitted>
```

2. As the software may have changed it is a good idea to read the **README.md** file for updated information.

```
student@cp:~$ cd metrics-server/ ; less README.md
```

```
<output_omitted>
```

3. Create the necessary objects. Be aware as new versions are released there may be some changes to the process and the created objects. Use the components.yaml to create the objects. The backslash is not necessary if you type it all on one line.

```
student@cp:~$ kubectl create -f \
https://github.com/kubernetes-sigs/metrics-server/releases/download/v0.3.7/components.yaml
```

```
clusterrole.rbac.authorization.k8s.io/system:aggregated-metrics-reader created
clusterrolebinding.rbac.authorization.k8s.io/metrics-server:system:auth-delegator created
rolebinding.rbac.authorization.k8s.io/metrics-server-auth-reader created
apiservice.apiregistration.k8s.io/v1beta1.metrics.k8s.io created
serviceaccount/metrics-server created
deployment.apps/metrics-server created
service/metrics-server created
clusterrole.rbac.authorization.k8s.io/system:metrics-server created
clusterrolebinding.rbac.authorization.k8s.io/system:metrics-server created
```

4. View the current objects, which are created in the `kube-system` namespace. All should show a `Running` status.

```
student@cp:~$ kubectl -n kube-system get pods
```

```
<output_omitted>
kube-proxy-ld2hb                        1/1     Running   0          2d21h
kube-scheduler-u16-1-13-1-2f8c          1/1     Running   0          2d21h
metrics-server-fc6d4999b-b9rjj          1/1     Running   0          42s
```

5. Edit the `metrics-server` deployment to allow insecure TLS. The default certificate is x509 self-signed and not trusted by default. In production you may want to configure and replace the certificate. You may encounter other issues as this software is fast-changing. The need for the `kubelet-preferred-address-types` line has been reported on some platforms.

```
student@cp:~$ kubectl -n kube-system edit deployment metrics-server
```

```
....
      spec:
        containers:
        - args:
          - --cert-dir=/tmp
          - --secure-port=4443
          - --kubelet-insecure-tls                    #<-- Add this line
          - --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname #<--May be needed
          image: k8s.gcr.io/metrics-server/metrics-server:v0.3.7
....
```

6. Test that the metrics server pod is running and does not show errors. At first you should see a few lines showing the container is listening. As the software changes these messages may be slightly different.

```
student@cp:~$ kubectl -n kube-system logs metrics-server<TAB>
```

```
I0207 14:08:13.383209       1 serving.go:312] Generated self-signed cert
(/tmp/apiserver.crt, /tmp/apiserver.key)
I0207 14:08:14.078360       1 secure_serving.go:116] Serving securely on
[::]:4443
```

7. Test that the metrics working by viewing pod and node metrics. Your output may have different pods. It can take an minute or so for the metrics to populate and not return an error.

```
student@cp:~$ sleep 120 ; kubectl top pod --all-namespaces
```

```
NAMESPACE     NAME                                      CPU(cores)   MEMORY(bytes)
kube-system   calico-kube-controllers-7b9dcdcc5-qg6zd   2m           6Mi
kube-system   calico-node-dr279                         23m          22Mi
kube-system   calico-node-xtvfd                         21m          22Mi
kube-system   coredns-5644d7b6d9-k7kts                  2m           6Mi
kube-system   coredns-5644d7b6d9-rnr2v                  3m           6Mi
<output_omitted>
```

```
student@cp:~$ kubectl top nodes
```

```
NAME               CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
cp   228m          11%    2357Mi          31%
worker      76m          3%     1385Mi          18%
```

8. Using keys we generated in an earlier lab we can also interrogate the API server. Your server IP address will be different.

```
student@cp:~$ curl --cert ./client.pem \
    --key ./client-key.pem --cacert ./ca.pem \
    https://k8scp:6443/apis/metrics.k8s.io/v1beta1/nodes

{
  "kind": "NodeMetricsList",
  "apiVersion": "metrics.k8s.io/v1beta1",
  "metadata": {
    "selfLink": "/apis/metrics.k8s.io/v1beta1/nodes"
  },
  "items": [
```

```
    {
      "metadata": {
        "name": "u16-1-13-1-2f8c",
        "selfLink": "/apis/metrics.k8s.io/v1beta1/nodes/u16-1-13-1-2f8c",
        "creationTimestamp": "2019-01-10T20:27:00Z"
      },
      "timestamp": "2019-01-10T20:26:18Z",
      "window": "30s",
      "usage": {
        "cpu": "215675721n",
        "memory": "2414744Ki"
      }
    },
<output_omitted>
```


Configure the Dashboard

While the dashboard looks nice it has not been a common tool in use. Those that could best develop the tool tend to only use the CLI, so it may lack full wanted functionality.

The first commands do not have the details. Refer to earlier content as necessary.

1. Search https://artifacthub.io/ for the `helm` organization and the `kubernetes-dashboard` chart.
2. Fetch the chart and edit the `values.yaml` file.

```
....
service:
  type: NodePort          #<-- Change to NodePort
  externalPort: 443
....
```

3. Install the chart and give it a name of `dashboard`
4. The helm chart version does not allow any resource access by default. We will give the dashboard full admin rights, which may be more than one would in production. The dashboard is running in the `default` namespace. First find the name of the service account, which is based off the name you used for the chart.

   There is more on service account in the `Security` chapter.

```
student@cp:~$ kubectl get serviceaccounts
```

```
NAME                              SECRETS   AGE
dashboard-kubernetes-dashboard    1         6m
default                           1         2d21h
myingress-ingress-nginx           1         42h
```

```
student@cp:~$ kubectl create clusterrolebinding dashaccess \
        --clusterrole=cluster-admin \
        --serviceaccount=default:dashboard-kubernetes-dashboard
```

```
clusterrolebinding.rbac.authorization.k8s.io/dashaccess created
```

5. On your local system open a browser and navigate to an HTTPS URL made of the `Public IP` and the high-numbered port. You will get a message about an insecure connection. Select the **Advanced** button, then **Add Exception...**, then **Confirm Security Exception**. Some browsers won't even give you to option. If nothing shows up try a different browser. The page should then show the `Kubernetes Dashboard`. You may be able to find the public IP address using **curl**.

```
student@cp:~$ curl ifconfig.io
```

```
35.231.8.178
```

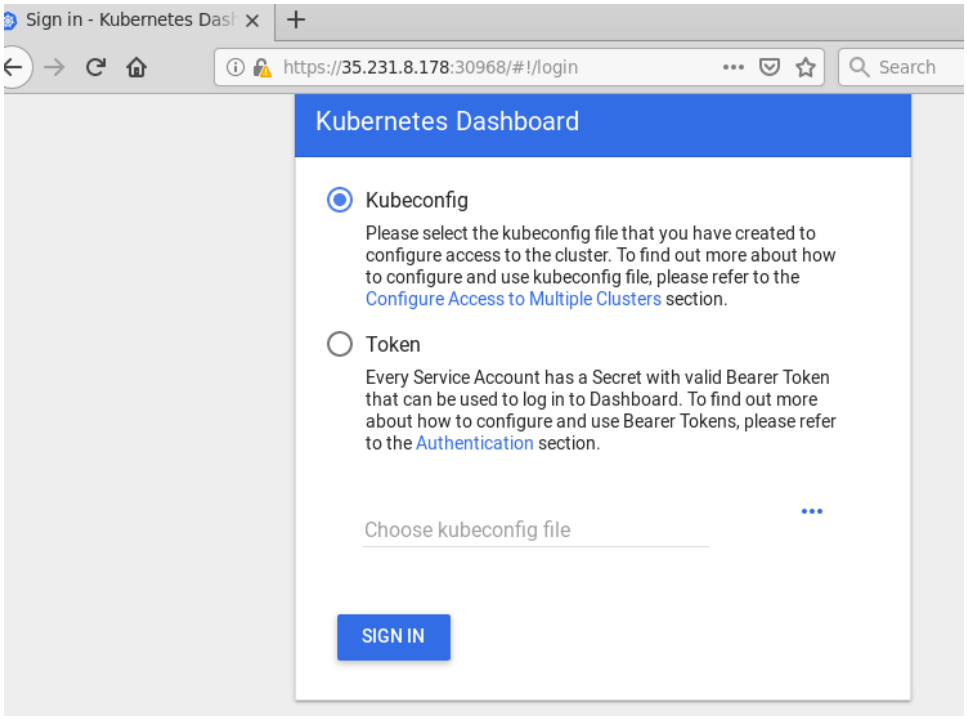

Figure 13.1: **External Access via Browser**

6. We will use the `Token` method to access the dashboard. With RBAC we need to use the proper token, the `kubernetes-dashboard-token` in this case. Find the token, copy it then paste into the login page. The **Tab** key can be helpful to complete the secret name instead of finding the hash.

```
student@cp:~$ kubectl describe secrets dashboard-kubernetes-dashboard-token-<TAB>
```

```
....
Data
====
ca.crt:     1025 bytes
namespace:  11 bytes
token:      eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZX
JuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJrdWJlLXN5c3RlbSIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY
291bnQvc2VjcmV0Lm5hbWUiOiJrdWJlcm5ldGVzLWRhc2hib2FyZC10b2tlbi1wbW04NCIsImt1YmVybmV0ZXMuaW8vc2Vydmlj
ZWFjY291bnQvc2VydmljZS1hY2NvdW50Lm5hbWUiOiJrdWJlcm5ldGVzLWRhc2hib2FyZCIsImt1YmVybmV0ZXMuaW8vc2Vydml
jZWFjY291bnQvc2VydmljZS1hY2NvdW50LnVpZCI6IjE5MDY4ZDIzLTE1MTctMTFlOS1hZmMyLTQyMDEwYThlMDAwMyIsInN1Yi
I6InN5c3RlbTpzZXJ2aWNlYWNjb3VudDprdWJlLXN5c3RlbTprdWJlcm5ldGVzLWRhc2hib2FyZCJ9.aYTUMWr290pjt5i32rb8
qXpq4onn3hLhvz6yLSYexgRd6NYsygVUyqnkRsFE1trg9i1ftNXKJdzkY5kQzN3AcpUTvyj_BvJgzNh3JM9p7QMjI8LHTz4TrRZ
rvwJVWitrEn4VnTQuFVcADFD_rKB9FyI_gvT_QiW5fQm24ygTIgfOYd44263oakG8sL64q7UfQNW2wt5SOorMUtybOmX4CXNUYM8
G44ejEtv9GW5OsVjEmLIGaoEMX7fctwUN_XCyPdzcCg2WOxRHahBJmbCuLz2SSWL52q4nXQmhTq_L8VDDpt6LjEqXW6LtDJZGjVC
s2MnBLerQz-ZAgsVaubbQ
```

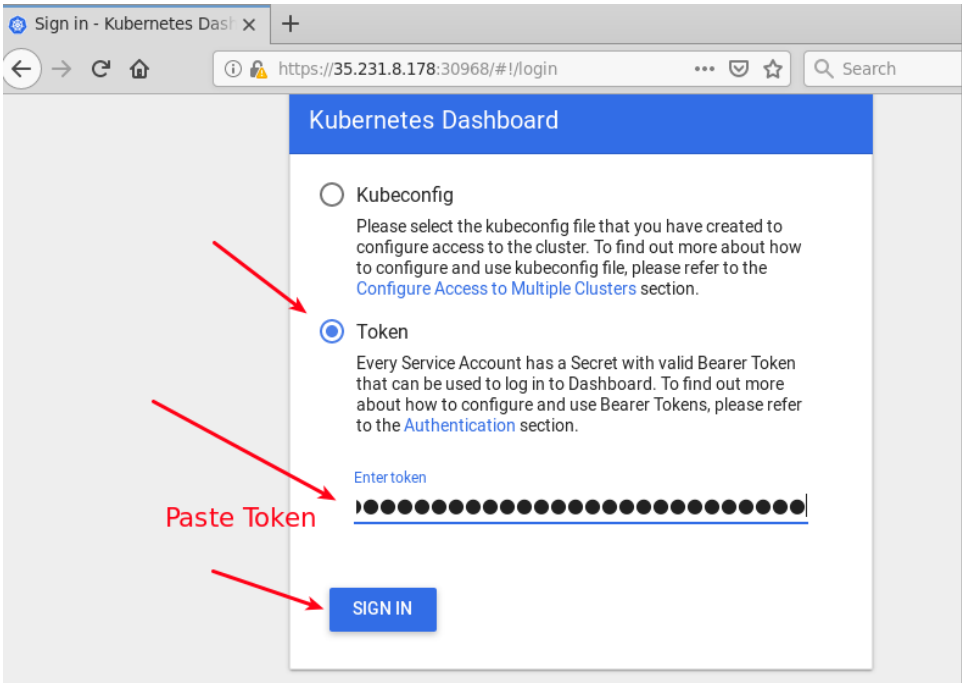

Figure 13.2: **External Access via Browser**

7. Navigate around the various sections and use the menu to the left as time allows. As the pod view is of the `default` namespace, you may want to switch over to the `kube-system` namespace or create a new deployment to view the resources via the GUI. Scale the deployment up and down and watch the responsiveness of the GUI.

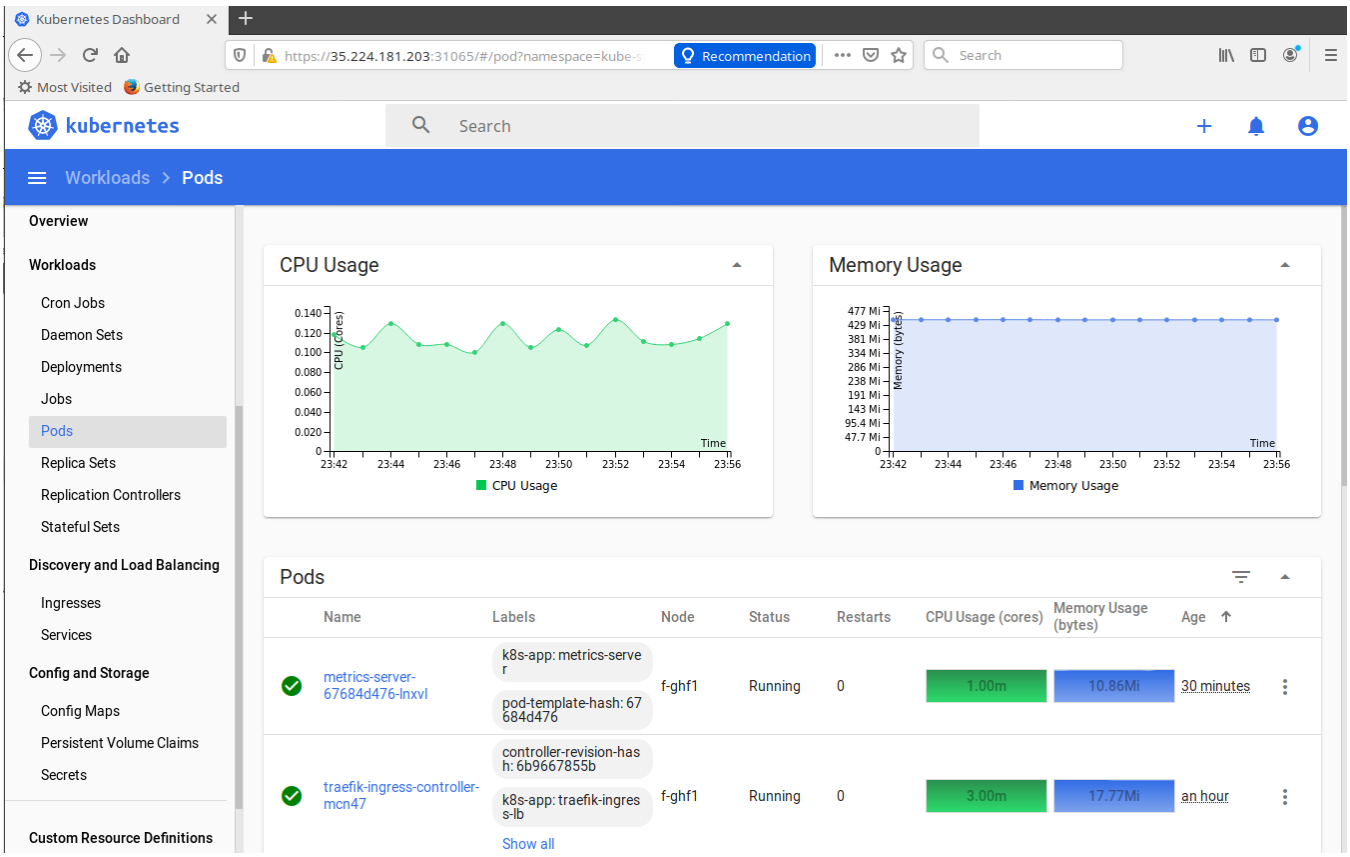

Figure 13.3: **External Access via Browser**