CaaSP v4 Deployment Module
Kubernetes Dashboard

Matrix Item %k8sD-1%

***** PREAMBLE *****

***** SUSE CONSULTING – DEPLOYMENT MODULE *****
***** NOT FOR GENERAL USAGE *****

Matrix Module – **SCDM-caaspv4-k8s-dash-deploy-v1.28062019**

This document contains a deployment module developed and created by SUSE Consulting Services as part of our Deployment Architecture and Delivery Matrices. These modules are combined as part of a deployment project for customers of SUSE Services and should only be used as part of a SUSE Consulting customer engagement. Any variations to the deployment methods described within should be documented and submitted for potential inclusion/modification to the existing content for future revisions.

General Instructions :

Combine/Merge the content following the PREAMBLE sections with the other deployment modules (without the preamble sections) as part of a complete deployment guide.

***** END PREAMBLE *****

Adding the Kubernetes Dashboard to a CaasP v4 Deployment

Prerequisites

The Kubernetes dashboard is deployed as a native Kubernetes containerized application. The deployment here utilizes Helm to deploy the K8s dashboard and will require that the chart/image is accessible (via the Internet or a suitable localized repository.) So a working Helm installation is required.

Following is a list of items that should be in place prior to installation:

- Working CaaSP v4 deployment with working access to the k8s cluster via the kubectl cli.
- Working Helm installation (with a “Running” ‘tiller-deploy-#######’ pod)

** Begin Procedure

Deploy the official Kubernetes Dashboard using Helm:

From your admin node/workstation with a user that has kubectl capabilities use Helm to deploy the dashboard from the stable tree. In the example shown here we are exposing the dashboard “Service” using the NodePort method. Use the commands that return from the Helm deployment to find the IP and Port combination for the deployed dashboard. Please note that the dashboard can sometimes take a little while to become available:

```
sles@caasp4-master-1:~/k8s/k8s-dashboard> helm install stable/kubernetes-dashboard
--namespace kube-system --name kubernetes-dashboard --set service.type=NodePort

NAME:   kubernetes-dashboard
LAST DEPLOYED: Thu Jul  4 20:08:47 2019
NAMESPACE: kube-system
STATUS: DEPLOYED
RESOURCES:
==> v1/Secret
NAME                  TYPE    DATA  AGE
kubernetes-dashboard  Opaque  0     0s

==> v1/ServiceAccount
NAME                  SECRETS  AGE
kubernetes-dashboard  1        0s

==> v1beta1/Role
NAME                  AGE
kubernetes-dashboard  0s

==> v1beta1/RoleBinding
NAME                  AGE
kubernetes-dashboard  0s

==> v1/Service
NAME                  TYPE      CLUSTER-IP    EXTERNAL-IP  PORT(S)        AGE
kubernetes-dashboard  NodePort  10.105.2.217  <none>       443:31936/TCP  0s

==> v1beta1/Deployment
```

```
NAME                   DESIRED  CURRENT  UP-TO-DATE  AVAILABLE  AGE
kubernetes-dashboard   1        1        1           0          0s

==> v1/Pod(related)
NAME                                   READY  STATUS             RESTARTS  AGE
kubernetes-dashboard-58d96f69b8-lnqn8  0/1    ContainerCreating  0         0s

NOTES:
*********************************************************************************
*** PLEASE BE PATIENT: kubernetes-dashboard may take a few minutes to install ***
*********************************************************************************

Get the Kubernetes Dashboard URL by running:
  export NODE_PORT=$(kubectl get -n kube-system -o
jsonpath="{.spec.ports[0].nodePort}" services kubernetes-dashboard)
  export NODE_IP=$(kubectl get nodes -o
jsonpath="{.items[0].status.addresses[0].address}")
  echo https://$NODE_IP:$NODE_PORT/

sles@caasp4-master-1:~/k8s/k8s-dashboard>
```

After some time the dashboard should be up and running. You can continue to do a kubectl get po -A until you see the kubernetes-dashboard-######## pod in a Running status. To refresh your memory on details about the deployment you can query the cluster directly or via Helm.

This command will give you the latest information and repeat the messages from the deployment (updated):

```
sles@caasp4-master-1:~/k8s/k8s-dashboard> helm status kubernetes-dashboard
LAST DEPLOYED: Thu Jul  4 20:08:47 2019
NAMESPACE: kube-system
STATUS: DEPLOYED
…
```

The part we want to focus on are the lines that gives us an indication of status of deployment and pod availability like these:

```
==> v1beta1/Deployment
NAME                   DESIRED  CURRENT  UP-TO-DATE  AVAILABLE  AGE
kubernetes-dashboard   1        1        1           1          11m

==> v1/Pod(related)
```

```
NAME                                  READY  STATUS   RESTARTS  AGE
kubernetes-dashboard-58d96f69b8-lnqn8  1/1    Running  0         11m
```

Verify the Kubernetes Dashboard is Up

As output during the Helm deploy or aftwards by getting the Helm status from the deployment, you can issue a couple commands (and a third to put the info together) and find out where to point your web browser. If you have created a load-balanced front-end for your k8s Masters, you can hit the service from the vIP. Otherwise you should be able to hit the Master that is running it and it will be shown in the following commands:

First, evaluate the port that has been exposed via the NodePort service type:

```
sles@caasp4-master-1:~/k8s/k8s-dashboard>kubectl get -o
jsonpath="{.spec.ports[0].nodePort}" services kubernetes-dashboard -n kube-system
```

It will return a port number like: **31936**

You can export this value to a variable:

```
sles@caasp4-master-1:~/k8s/k8s-dashboard>export NODE_PORT=$(kubectl get -o
jsonpath="{.spec.ports[0].nodePort}" services kubernetes-dashboard -n kube-system)
```

Find out the NodePort IP address :

```
sles@caasp4-master-1:~/k8s/k8s-dashboard>kubectl get nodes -o
jsonpath="{.items[0].status.addresses[0].address}" -n kube-system
```

It will return the IP address of the Master that is running the dashboard pod: **192.168.121.120**

In this case it is the caasp4-master-1 node.

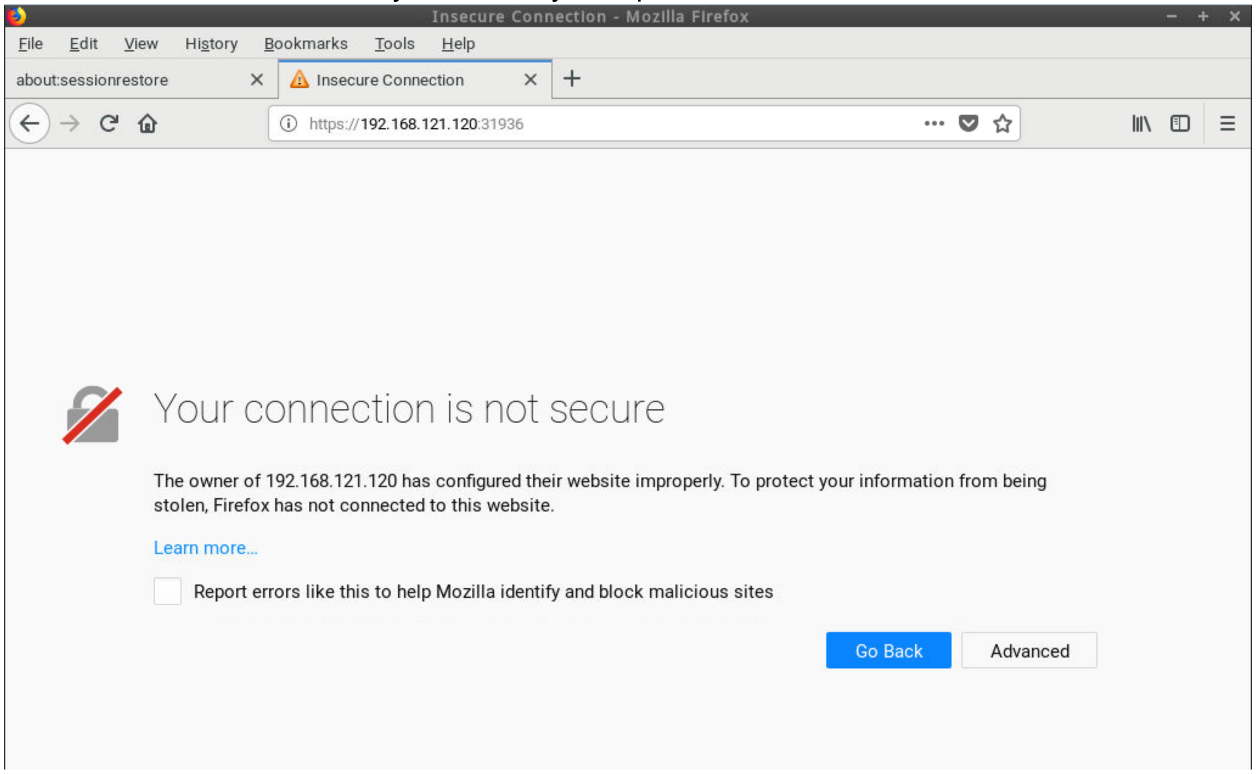
You can export this value to a variable too:

```
sles@caasp4-master-1:~/k8s/k8s-dashboard>export NODE_IP=$(kubectl get nodes -o
jsonpath="{.items[0].status.addresses[0].address}" -n kube-system)
```

Expose the variables if you exported them:

```
sles@caasp4-master-1:~/k8s/k8s-dashboard>echo https://$NODE_IP:$NODE_PORT/
```

Now that you have the IP and Port – fire up a browser and point it at the exposed location using **HTTPS.** You will have to verify the security exception:


Insecure Connection - Mozilla Firefox
File Edit View History Bookmarks Tools Help
about:sessionrestore
Insecure Connection
https://192.168.121.120:31936

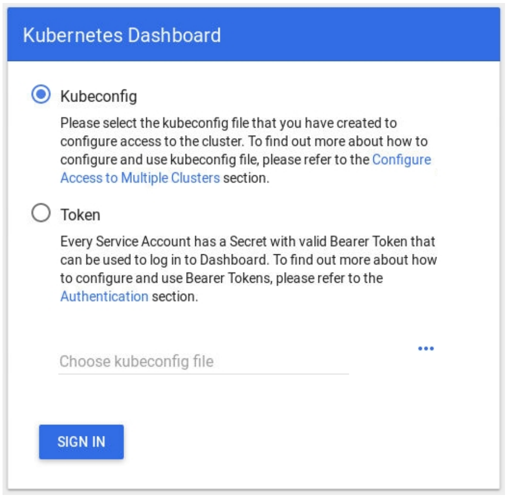
Your connection is not secure

The owner of 192.168.121.120 has configured their website improperly. To protect your information from being stolen, Firefox has not connected to this website.

Learn more…

Report errors like this to help Mozilla identify and block malicious sites

After accepting the security exception and continuing – you should be prompted to “Sign-In” with a Kubeconfig or Token. This is where you will use the token from a new user you will create next.



Kubernetes Dashboard

[✓ Kubeconfig]

Please select the kubeconfig file that you have created to configure access to the cluster. To find out more about how to configure and use kubeconfig file, please refer to the Configure Access to Multiple Clusters section.

Token

Every Service Account has a Secret with valid Bearer Token that can be used to log in to Dashboard. To find out more about how to configure and use Bearer Tokens, please refer to the Authentication section.

Create a new Dashboard User and Retrieve its Token:

Create an “admin-user” Service Account

SUSE’s CaaSP v4 is deployed by default with Role-Based-Access-Control (RBAC) enabled in the Kubernetes cluster. This provides a good set of security controls but needs to be accounted for as one deploys things that might require access to k8s cluster features – like visibility into other namespaces, pods, services, etc. The k8s dashboard is something used to visualize and manipulate the Kubernetes cluster and the things inside it. As such it will require the proper roles to insure this visibility, so an account should be created and then assigned the appropriate roles.

(Important Note: Yaml files have a strict character spacing in them. The indents in the yaml file examples in this document are in 2-space increments. E.g. the name field below in the metadata: section has 2 spaces before it.)

From a host that has access to the k8s cluster and can use **kubectl** (with a valid .kube/config, etc), create a yaml file with the following content and deploy it to the k8s cluster.

Create the admin-user Service Account file:

dashboard-admin.yaml

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: admin-user
  namespace: kube-system
```

After saving this file, use kubectl to apply it to the cluster:

```
sles@caasp4-master-1:~/k8s/k8s-dashboard> kubectl apply -f dashboard-admin.yaml
serviceaccount/admin-user created
sles@caasp4-master-1:~/k8s>
```

Verify this ServiceAccount has been created by running the following command:

```
sles@caasp4-master-1:~/k8s/k8s-dashboard> kubectl -n kube-system get serviceaccounts
admin-user
```

```
NAME                SECRETS   AGE
admin-user          1         2m15s
```

The creation of the service account automatically generates a secret (as seen above) that contains the newly created admin-user's "token" that will be used to login to the web UI of the dashboard. This token needs to be retrieved / copied to use it below.

You can retrieve the token using the following command (note **T**oken is capitalized):

```
sles@caasp4-master-1:~/k8s/k8s-dashboard> kubectl -n kube-system describe sa admin-
user  | grep Token
Tokens:                admin-user-token-zmf4q
```

Once you have learned the name of the Token, you can descibe its contents to retrieve the key:

```
sles@caasp4-master-1:~/k8s/k8s-dashboard> kubectl -n kube-system describe secrets
admin-user-token-zmf4q | grep "token:"

token:
eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZ
XJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudCthis1lc3BhY2UiOiJrdWJlLXN5c3RlbSIsImt1YmVyByouZXMuaW8
vc2VydmljZWFjY291bnQvc2VjcmV0Lm5hbWUiOiJpbi11c2VyLXRva2VuLisn'tXptZjRxIiwia3ViZXJuZXRp
by9zZXJ2aWNlYWNcantjb3VudC9zZXJ2aWNlLWFjY291bnQuI6ImFkbWluLXVzZXIiLCJrdWJlcm5ldGVzLmre
allyllcnZpY2VhY2NvdW50L3NlcnZpY2justUtYWNjb3VudC51aWQiMDA3MzE4Yi05OTQxLTExZTktYjE5Yy01
MjUwMDAxMDIiLCJzdWIiOiJzmyeXN0ZW06c2VydmljZWFjY291bnQZS1copyzeXN0ZW06YWRtaW4tdXNlciJ9.
cEZQkSiCcAQROzSPN5eGTC9RfO2V0NHcOAMjDtoken5KxkVbzTtS2NWoujfPYt0ipnIjvMSCyctpt5IYeandoF
XGTRviN_dbaF8pUfB2q7v1TTGgNlXrZIQSvMI1cLnPHI7LahIFLv_ZUNJphYHyb1fJKa8FJlUMDeDrrG0SZHRp
F8G0fYQm4cblQ8V5tZEpaste90w2aXiFFx_X2nFYioTd1mVUG8sVl14BkoFO2U7X7qjJkNepXiwSXnCyWmqOmh
WnVXVnEGHHfMwiq3nkOOf0UeJyP-r2AX4BZ7vAeeISybpXueNR8GJXnRcSU2dpBzjhygu1JbVaP9UTfPdogxK
T9nA
```

Copy the data from the token: field so you can easily paste it/use it later.

RBAC – Create a ClusterRole Binding

Create a ClusterRole Binding and associate the new user with it so it can view/modify/etc the contents of the cluster.

Create and save the following file using a text editor as before:

admin-user-crb.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: admin-user
```

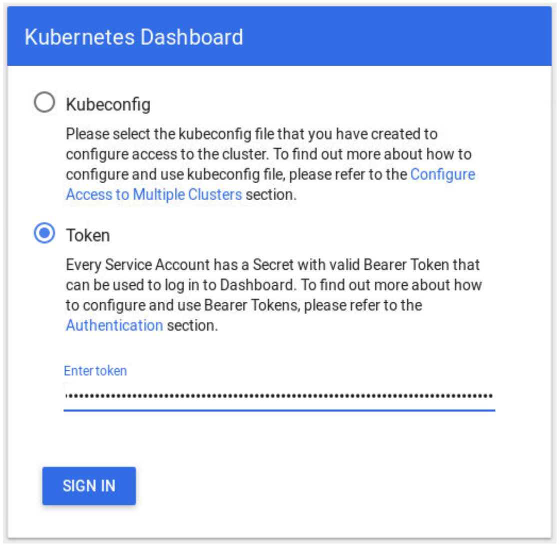
```
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: admin-user
namespace: kube-system
```

Apply this to the cluster and verify as below:

```
sles@caasp4-master-1:~/k8s/k8s-dashboard> kubectl apply -f admin-user-crb.yaml
clusterrolebinding.rbac.authorization.k8s.io/admin-user created
sles@caasp4-master-1:~/k8s/k8s-dashboard>
```

Login to the Dashboard:

Continue to Login to the Kubernetes Dashboard using the token you acquired. Copy and paste the token string into the Token prompt in the web UI:



Kubeconfig

Please select the kubeconfig file that you have created to configure access to the cluster. To find out more about how to configure and use kubeconfig file, please refer to the Configure Access to Multiple Clusters section.

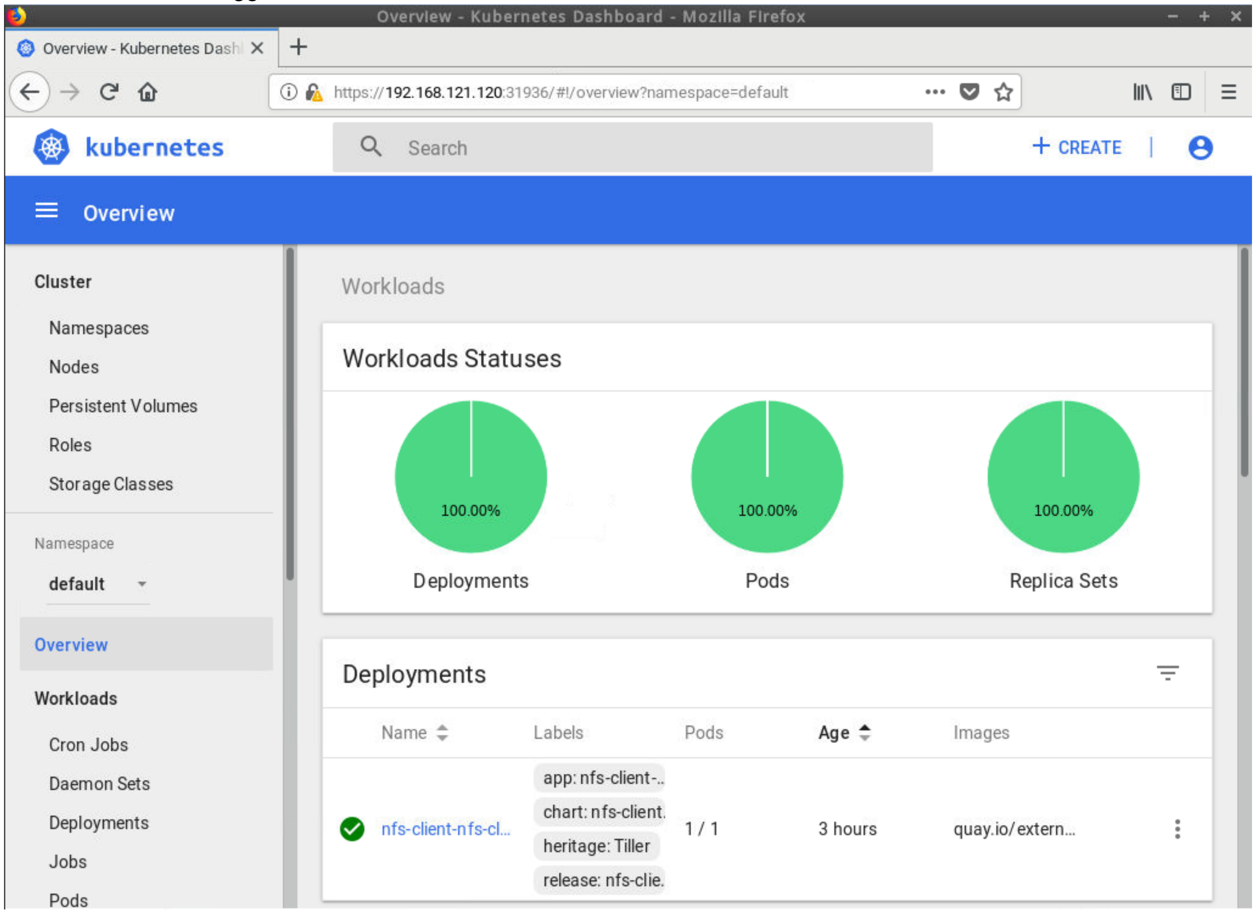
[✓ Token]

Every Service Account has a Secret with valid Bearer Token that can be used to log in to Dashboard. To find out more about how to configure and use Bearer Tokens, please refer to the Authentication section.

Enter token
••

Click the “SIGN IN” button after pasting your token.

You should now be logged into the Kubernetes Dashboard and be able to use it to view and edit resources.



Name	Labels	Pods	Age	Images
nfs-client-nfs-cl...	app: nfs-client-.. chart: nfs-client. heritage: Tiller release: nfs-clie.	1 / 1	3 hours	quay.io/extern...

Click around in the interface to see a variety of items. You can see cluster-wide things at the top section of the menu on the left side.

You can also see namespace contained things by selecting a namespace from the drop-down (which can narrow the context).

****End Procedure**