The deployment of applications and add-ons in Kubernetes are straightforward until those need to consume the Kubernetes API, that is the case of the [Kubernetes Dashboard add-on](https://github.com/kubernetes/dashboard). On version 1.7 of Kubernetes the RBAC service was introduced and many of those applications and add-ons started to crash.

This post will walk you through the process to deploy, configure and access to the Kubernetes Dashboard.

**Kubernetes Dashboard Prerequisites**

* Running a Kubernetes platform 1.7.x and above.
* Internet connection (pull Kubernetes Dashboard manifest and image)

If you don’t have a Kubernetes platform running at this time take a look to my post [Hands-on Kubernetes: Deployment](http://www.joseluisgomez.com/automation/kubernetes-deployment/)

**Deploying Kubernetes Dashboard**

On a node with kubectl command line installed run the following command. The manifest includes all the Kubernetes components to create for the add-on. **Note**: The K8s Dashboard URL has changed including the version, what it will make to fail the following command when new versions are released. If the following command doesn’t work, please make sure to visit the Kubernetes Dashboard repo and confirm the new URL.

kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v1.10.1/src/deploy/recommended/kubernetes-dashboard.yaml

Check if your dashboard is running listing the pods in the namespace kube-system with the following command. You should see a kubernetes-dashboard-… pod with the status “running”.

kubectl -n kube-system get pod

**Opening the dashboard**

Access the dashboard at:

https://<master-ip>:<apiserver-port>/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/

Copy

You likely got an error trying to access the dashboard.

{

"kind": "Status",

"apiVersion": "v1",

"metadata": {

},

"status": "Failure",

"message": "services \"https:kubernetes-dashboard:\" is forbidden: User \"system:anonymous\" cannot get services/proxy in the namespace \"kube-system\"",

"reason": "Forbidden",

"details": {

"name": "https:kubernetes-dashboard:",

"kind": "services"

},

"code": 403

}

At this point you will start to look for a solution on Internet. The solutions you mostly will find are the ones below, **kubectl prox**y and **NodePort**, but they are not recommended for production.

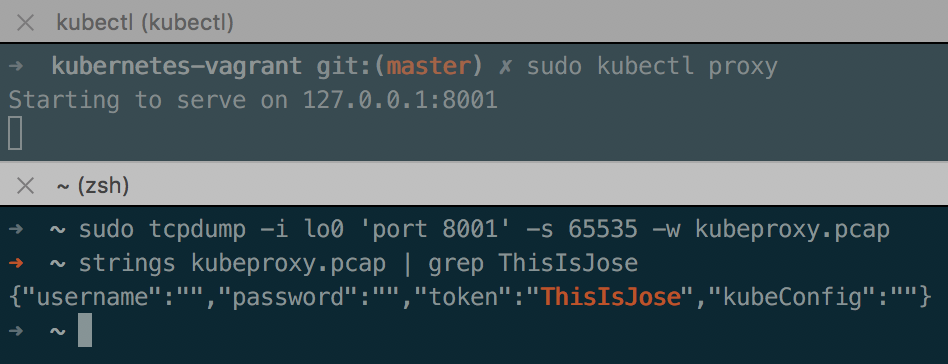
**kubectl proxy**

This access mode is not recommended to be used as the method to publicly expose your dashboard. The proxy only allows HTTP connection.

To use this method you need to install kubectl in your computer and run the following command. The proxy will start to serve the dashboard on **http://localhost:8001** by default.

kubectl proxy –address=0.0.0.0

Personally I don’t recommend to use this connection method. If you are sharing a jump server or even on your own computer, a sniffer will able to capture your kubeconfig file or token since they are sent as plain text via HTTP.



You can find more information [Accessing Dashboard 1.7.X and above](https://github.com/kubernetes/dashboard/wiki/Accessing-Dashboard---1.7.X-and-above).

**NodePort**

If you are running a single node setup (unlikely in production), you can configure the Kubernetes Dashboard service to use NodePort as the type to publish the service.

I’m not going to explain how to set the service type since the Kubernetes Dashboard site has a clear procedure ([Accessing Dashboard 1.7.X and above](https://github.com/kubernetes/dashboard/wiki/Accessing-Dashboard---1.7.X-and-above#nodeport))

**API Server**

This is the method which I recommend to use for production systems as well as for dev and test. It is important to keep the same security mechanisms end to end and get familiar with Kubernetes RBAC.

To use the API server you need to install the user certificates in the browser. I’m going to use the kubeconfig file generated by kubeadm, I want to keep this post as short as I can.

**Tip**: For production systems each user should have its own certificates. Bitnami have a great doc about how to configure it ([Create User With Limited Namespace Access](https://docs.bitnami.com/kubernetes/how-to/configure-rbac-in-your-kubernetes-cluster/#use-case-1-create-user-with-limited-namespace-access))

Let’s see how we can extract the certificates from the kubeconfig file:

1. Locate your kubeconfig or config file which you use to run kubectl commands. If you have used my Vagrant file above, you can find it on **/home/vagrant/.kube/config** or **/etc/kubernetes/admin.conf**
2. You need to export a single file (.p12) with the following two certificates: the client-certificate-data, and the client-key-data. My example runs the command on **/home/vagrant**. If you run this command on macOS, be sure to change the **base64 -d** to **base64 -D**.

grep 'client-certificate-data' ~/.kube/config | head -n 1 | awk '{print $2}' | base64 -d >> kubecfg.crt

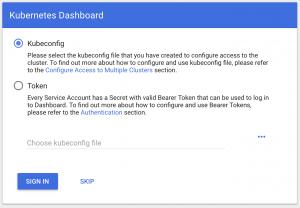
Copy

grep 'client-key-data' ~/.kube/config | head -n 1 | awk '{print $2}' | base64 -d >> kubecfg.key

Copy

openssl pkcs12 -export -clcerts -inkey kubecfg.key -in kubecfg.crt -out kubecfg.p12 -name "kubernetes-client"

Copy

1. Import the kubecfg.p12 certificate, reopen your browser, and visit the Kubernetes Dashboard URL. Accept any warning and you should see the authentication page. You can skip the login and check you are not able to perform any task.  
   
2. The following steps have been copied from the Kubernetes Dashboard wiki page ([Creating-sample-user](https://github.com/kubernetes/dashboard/wiki/Creating-sample-user))
   1. Create service account
   2. cat <<EOF | kubectl create -f -
   3. apiVersion: v1
   4. kind: ServiceAccount
   5. metadata:
   6. name: admin-user
   7. namespace: kube-system

EOF

Copy

* 1. Create ClusterRoleBinding
  2. cat <<EOF | kubectl create -f -
  3. apiVersion: rbac.authorization.k8s.io/v1
  4. kind: ClusterRoleBinding
  5. metadata:
  6. name: admin-user
  7. roleRef:
  8. apiGroup: rbac.authorization.k8s.io
  9. kind: ClusterRole
  10. name: cluster-admin
  11. subjects:
  12. - kind: ServiceAccount
  13. name: admin-user
  14. namespace: kube-system

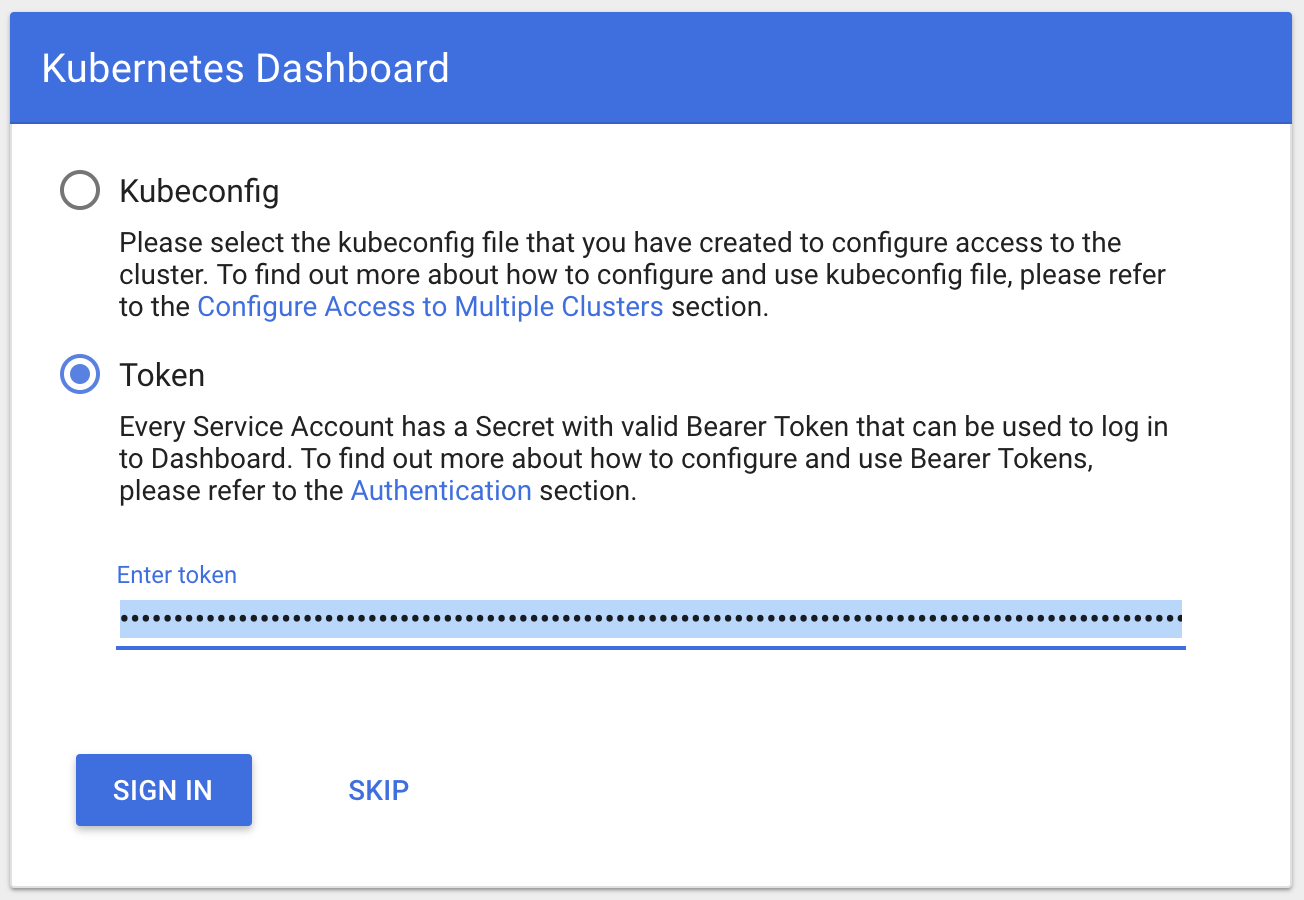
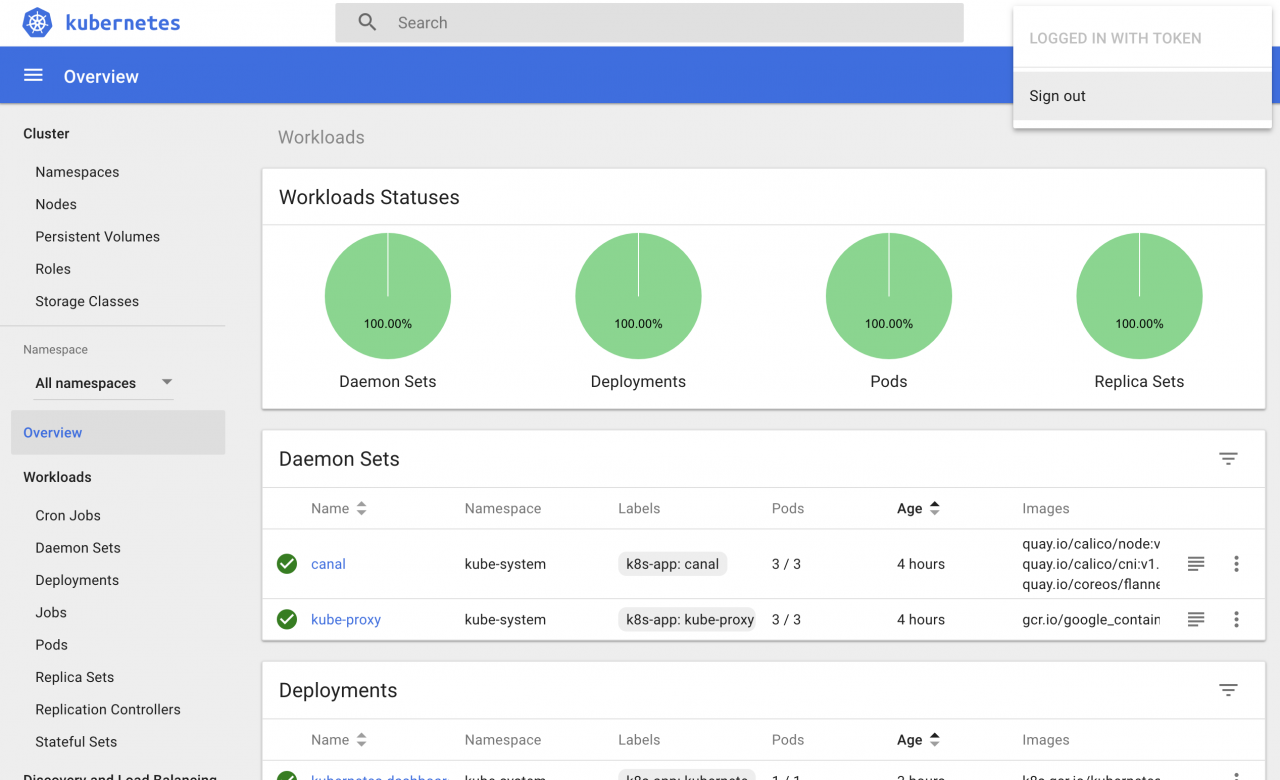
EOF

Copy

* 1. Get the Bearer Token. Once you run the following command, copy the token value which you will use on the following step.

kubectl -n kube-system describe secret $(kubectl -n kube-system get secret | grep admin-user | awk '{print $1}')

Copy

* 1. Come back to your browser and choose token on the login page. You will need to paste the token value you have copied on the previous step.  
     
  2. Click “SIGN IN” and you should be able to see your Kubernetes Dashboard fully operational.  
     

**Summary**