



Mise en place du Dashboard sur Kubernetes



kubernetes



Mise en place du Dashboard sur kubernetes

Téléchargeant déjà les images nécessaires afin de les stocker dans notre registre privée :

```
docker pull kubernetesui/dashboard:v2.2.0
docker tag kubernetesui/dashboard:v2.2.0 docker-registry:5000/dashboard:v1
docker push docker-registry:5000/dashboard:v1
docker pull kubernetesui/metrics-scraper:v1.0.6
docker tag kubernetesui/metrics-scraper:v1.0.6 docker-registry:5000/metrics-scraper:v1
docker push docker-registry:5000/metrics-scraper:v1
docker image ls
```

Créons un nouvel espace de nom si la création n'est pas intégré au manifest:

```
kubectl create namespace kubernetes-dashboard
```

Modifiant le premier manifest baptisé recommended1.yaml en mettant en place un objet namespace un compte de service un service des secrets des configmaps des rôles des rôles de cluster des liaisons de rôle local des liaisons de rôle de cluster

```
apiVersion: v1
kind: Namespace
metadata:
  name: kubernetes-dashboard

---

apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
  namespace: kubernetes-dashboard

---

kind: Service
apiVersion: v1
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
  namespace: kubernetes-dashboard
```

```
spec:
  ports:
    - port: 443
      targetPort: 8443
  selector:
    k8s-app: kubernetes-dashboard
  type: NodePort
---

apiVersion: v1
kind: Secret
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard-certs
  namespace: kubernetes-dashboard
type: Opaque

---

apiVersion: v1
kind: Secret
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard-csrf
  namespace: kubernetes-dashboard
type: Opaque
data:
  csrf: ""

---

apiVersion: v1
kind: Secret
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard-key-holder
  namespace: kubernetes-dashboard
type: Opaque

---

kind: ConfigMap
apiVersion: v1
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard-settings
  namespace: kubernetes-dashboard
```

```
---
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
  namespace: kubernetes-dashboard
rules:
  # Allow Dashboard to get, update and delete Dashboard exclusive secrets.
  - apiGroups: [""]
    resources: ["secrets"]
    resourceNames: ["kubernetes-dashboard-key-holder", "kubernetes-dashboard-certs", "kubernetes-dashboard-csrf"]
    verbs: ["get", "update", "delete"]
    # Allow Dashboard to get and update 'kubernetes-dashboard-settings' config map.
  - apiGroups: [""]
    resources: ["configmaps"]
    resourceNames: ["kubernetes-dashboard-settings"]
    verbs: ["get", "update"]
    # Allow Dashboard to get metrics.
  - apiGroups: [""]
    resources: ["services"]
    resourceNames: ["heapster", "dashboard-metrics-scraper"]
    verbs: ["proxy"]
  - apiGroups: [""]
    resources: ["services/proxy"]
    resourceNames: ["heapster", "http:heapster:", "https:heapster:", "dashboard-metrics-scraper", "http:dashboard-metrics-scraper"]
    verbs: ["get"]

---

kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
rules:
  # Allow Metrics Scraper to get metrics from the Metrics server
  - apiGroups: ["metrics.k8s.io"]
    resources: ["pods", "nodes"]
    verbs: ["get", "list", "watch"]

---

apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  labels:
```

```
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
  namespace: kubernetes-dashboard
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: kubernetes-dashboard
subjects:
  - kind: ServiceAccount
    name: kubernetes-dashboard
    namespace: kubernetes-dashboard

---

apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: kubernetes-dashboard
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: kubernetes-dashboard
subjects:
  - kind: ServiceAccount
    name: kubernetes-dashboard
    namespace: kubernetes-dashboard
```

Appliquons ce premier manifest

```
kubectl apply -f recommended1.yaml
```

Créons un secret afin de pouvoir déployer Dashboad à partir des images de notre registre privé dans le namespace Appliquons ce premier manifest

```
kubectl create secret generic regcred1 -n kubernetes-dashboard --from-file=.dockerconfigjson=/root/.docker/config.json --type=kubernetes.io/dockerconfigjson
```

Modifions le deuxième manifest suivant qui correspond au déploiement de metrics-scraper et le dashboard avec un service:

nano recommended2.yaml

```
kind: Deployment
apiVersion: apps/v1
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
  namespace: kubernetes-dashboard
spec:
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      k8s-app: kubernetes-dashboard
  template:
    metadata:
      labels:
        k8s-app: kubernetes-dashboard
    spec:
      containers:
        - name: kubernetes-dashboard
          image: 'docker-registry:5000/dashboard:v1'
          imagePullPolicy: IfNotPresent
          ports:
            - containerPort: 8443
              protocol: TCP
          args:
            - '--auto-generate-certificates'
            - '--namespace=kubernetes-dashboard'
          volumeMounts:
            - name: kubernetes-dashboard-certs
              mountPath: /certs
            - mountPath: /tmp
              name: tmp-volume
          livenessProbe:
            httpGet:
              scheme: HTTPS
              path: /
              port: 8443
            initialDelaySeconds: 30
            timeoutSeconds: 30
          securityContext:
            allowPrivilegeEscalation: false
            readOnlyRootFilesystem: true
            runAsUser: 1001
            runAsGroup: 2001
      imagePullSecrets:
        - name: regcred1
      volumes:
        - name: kubernetes-dashboard-certs
```

```
        secret:
          secretName: kubernetes-dashboard-certs
      - name: tmp-volume
        emptyDir: {}
    serviceAccountName: kubernetes-dashboard
    nodeSelector:
      kubernetes.io/os: linux
    tolerations:
      - key: node-role.kubernetes.io/master
        effect: NoSchedule

---

kind: Service
apiVersion: v1
metadata:
  labels:
    k8s-app: dashboard-metrics-scraper
  name: dashboard-metrics-scraper
  namespace: kubernetes-dashboard
spec:
  ports:
    - port: 8000
      targetPort: 8000
  selector:
    k8s-app: dashboard-metrics-scraper

---

kind: Deployment
apiVersion: apps/v1
metadata:
  labels:
    k8s-app: dashboard-metrics-scraper
  name: dashboard-metrics-scraper
  namespace: kubernetes-dashboard
spec:
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      k8s-app: dashboard-metrics-scraper
  template:
    metadata:
      labels:
        k8s-app: dashboard-metrics-scraper
      annotations:
        seccomp.security.alpha.kubernetes.io/pod: 'runtime/default'
    spec:
      containers:
        - name: dashboard-metrics-scraper
```

```
      image: docker-registry:5000/metrics-scraper:v1
      ports:
        - containerPort: 8000
          protocol: TCP
      livenessProbe:
        httpGet:
          scheme: HTTP
          path: /
          port: 8000
        initialDelaySeconds: 30
        timeoutSeconds: 30
      volumeMounts:
      - mountPath: /tmp
        name: tmp-volume
      securityContext:
        allowPrivilegeEscalation: false
        readOnlyRootFilesystem: true
        runAsUser: 1001
        runAsGroup: 2001
  serviceAccountName: kubernetes-dashboard
  nodeSelector:
    "kubernetes.io/os": linux
  # Comment the following tolerations if Dashboard must not be deployed on master
  tolerations:
    - key: node-role.kubernetes.io/master
      effect: NoSchedule
  imagePullSecrets:
    -
      name: regcred1
  volumes:
    - name: tmp-volume
      emptyDir: {}
```

Appliquons le deuxième manifest

```
kubectl apply -f recommended2.yaml
```

Créons un rôle pour le compte kubernetes-dashboard afin d'accéder au dashboard

```
kubectl create clusterrolebinding add-on-cluster-admin --clusterrole=cluster-admin --serviceaccount=kubernetes-dashboard:kubernetes-dashboard
```

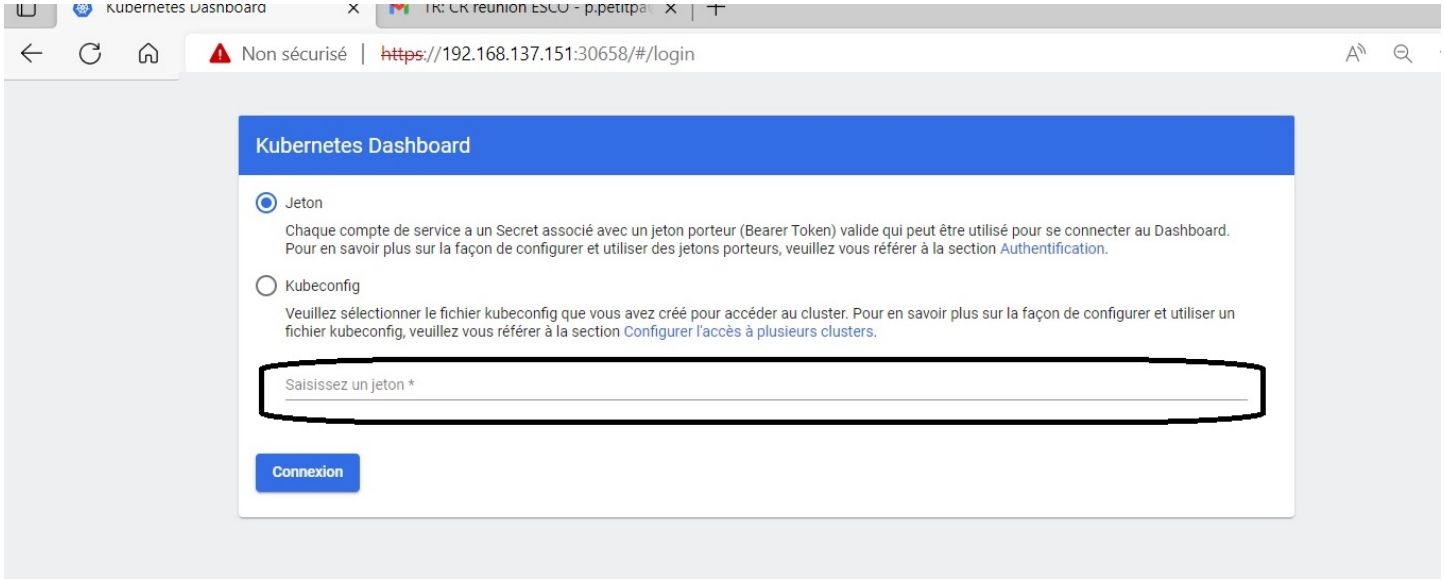
Créons un jeton pour pouvoir accéder au dashboard

```
kubectl create token kubernetes-dashboard -n kubernetes-dashboard
```

vérifions le port sur lequel le service web du dashboard est atteignable

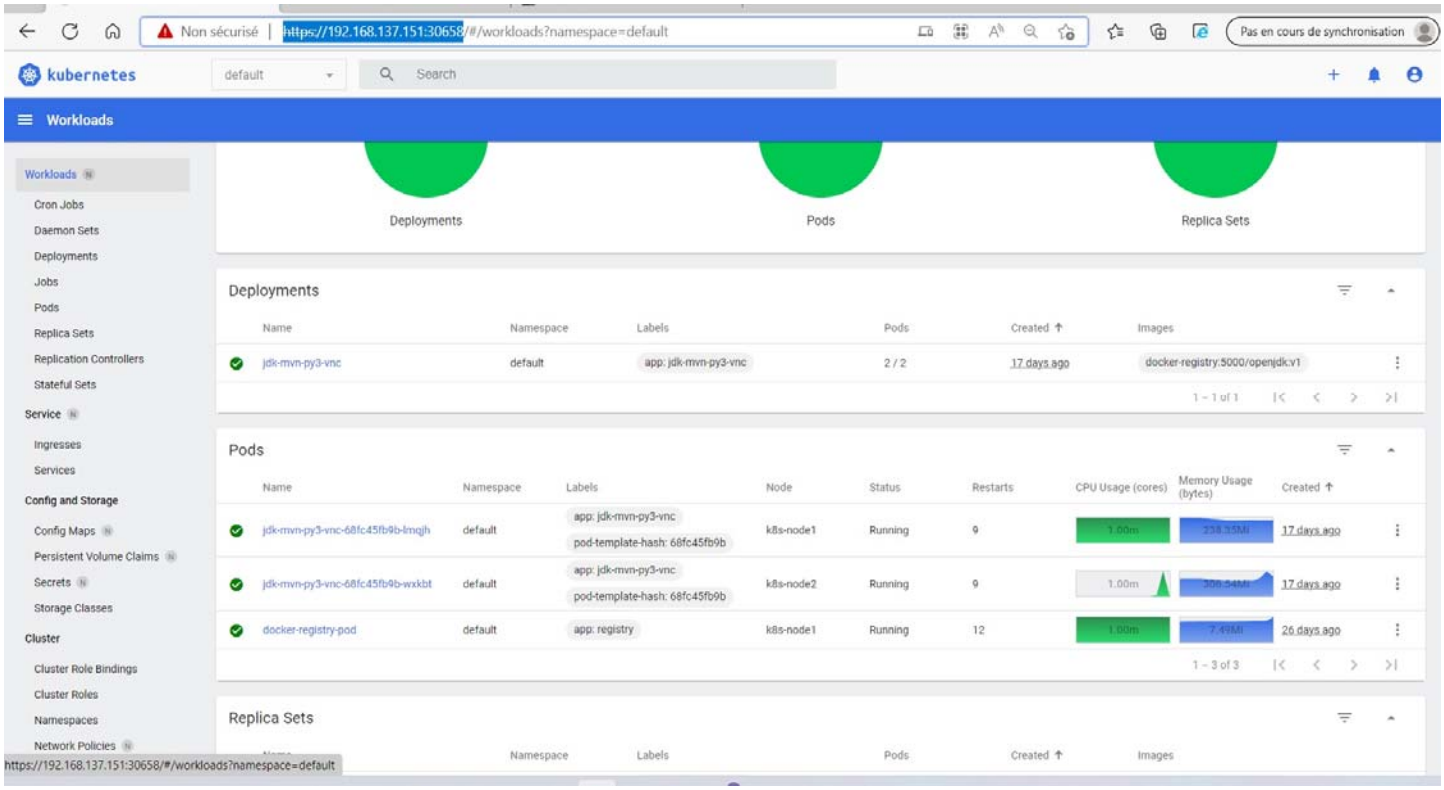
```
kubectl get service --all-namespaces
```

ouvrir ce service web ici https://192.168.137.151:30658



Collez le jeton obtenu précédemment :

La page web principale du dashboard apparait



Pour effacer une grande partie du déploiement effectué

```
kubectl delete deployment dashboard-metrics-scraper -n kubernetes-dashboard
kubectl delete deployment kubernetes-dashboard -n kubernetes-dashboard
kubectl delete pod mypod
kubectl delete svc dashboard-metrics-scraper -n kubernetes-dashboard
kubectl delete svc kubernetes-dashboard -n kubernetes-dashboard
kubectl  delete ns kubernetes-dashboard --force --grace-period=0
```