Université de Picardie Jules Verne

Informatique – Master CCM

INSSET – Saint-Quentin

Conteneurs Applicatifs et Micro-Services M2

C. Drocourt

cyril.drocourt@u-picardie.fr

Cours 5 : Kubernetes – Fonctionnement avancé

V2023.01

Table des matières

1 - Kubernetes Proxy

Kubernetes possède également un service de proxy permettant d'accéder aux applications, il faut d'abord l'exécuter sur le nœud maître :

```
root@maitre:~# kubectl proxy --address 192.168.56.110 --port=9999 --accept-hosts='^*$'
Starting to serve on 192.168.56.110:9999
```

Puis, dans un navigateur, il est possible d'accéder aux services déployés via les URLs suivantes :

```
http://10.3.134.XXX:9999/api/v1/namespaces/default/services/monservice:3080/proxy/
http://10.3.134.XXX:9999/api/v1/namespaces/default/services/monservice1:5080/
proxy/
http://10.3.134.XXX:9999/api/v1/namespaces/default/services/monservice2:6080/
proxy/
```

2 - DashBoard

L'une des interface les plus simple pour gérer un cluster Kubernetes est le « dashboard ». Il faut d'abord l'installer avec la commande suivante :

```
root@maitre:~# kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/alternative.yaml
namespace/kubernetes-dashboard created
serviceaccount/kubernetes-dashboard created
service/kubernetes-dashboard created
secret/kubernetes-dashboard-certs created
secret/kubernetes-dashboard-csrf created
secret/kubernetes-dashboard-key-holder created
configmap/kubernetes-dashboard-settings created
role.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrole.rbac.authorization.k8s.io/kubernetes-dashboard created
rolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrolebinding.rbac.authorization.k8s.io/kubernetes-dashboard
created
deployment.apps/kubernetes-dashboard created
service/dashboard-metrics-scraper created
deployment.apps/dashboard-metrics-scraper created
```

On va ensuite créer un compte d'accès :

```
root@maitre:~# cat admin-user.yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: admin-user
  namespace: kubernetes-dashboard
root@maitre:~/.kube# kubectl apply -f admin-user.yaml
serviceaccount/admin-user created
```

```
root@maitre:~# cat admin-author.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: kubernetes-dashboard
root@maitre:~/.kube# kubectl apply -f admin-author.yaml
clusterrolebinding.rbac.authorization.k8s.io/admin-user created
```

On va créer ensuite un jeton d'accès :

```
root@maitre:~# kubectl -n kubernetes-dashboard create token admin-user
eyJhbGciOiJSUzI1NiIsImtpZCI6ImZCR0JORjFhLTVxME5LMldPOFhhVFp6W
TNIZ1o0OFEzREFnNGk4ZEZaVmMifQ.eyJhdWQiOlsiaHR0cHM6Ly9rdWJlcm5
ldGVzLmRlZmF1bHQuc3ZjLmNsdXN0ZXIubG9jYWwiXSwiZXhwIjoxNjcwOTMw
MjQ3LCJpYXQiOjE2NzA5MjY2NDcsImlzcyI6Imh0dHBzOi8va3ViZXJuZXRlc
y5kZWZhdWx0LnN2Yy5jbHVzdGVyLmxvY2FsIiwia3ViZXJuZXRlcy5pbyI6ey
JuYW1lc3BhY2UiOiJrdWJlcm5ldGVzLWRhc2hib2FyZCIsInNlcnZpY2VhY2N
vdW50Ijp7Im5hbWUiOiJhZG1pbi11c2VyIiwidWlkIjoiYjU5OTNhZDgtYzEw
OS00YTEzLThlMDktMDRmZmMzMDVkMmM2In19LCJuYmYiOjE2NzA5MjY2NDcsI
nN1YiI6InN5c3RlbTpzZXJ2aWNlYWNjb3VudDprdWJlcm5ldGVzLWRhc2hib2
FyZDphZG1pbi11c2VyIn0.KRxkn-fgEXRD030a-mtQsHF8R-
ifMsObfY4nchI04-LHJI7nEnF-
xMqe1dP_9hS4mAUhFQKutwYh6kLldc81LrwA06elW2oLef8p6UidGhuDAqA54
6zxVlaMwbGfhErciXYYeTDg3ifSjmeiutV2KyI5IRSSPoKfjxuje5wCLOX92x
VjlrsFMWY6Eu8sszBxoq01twaRBRz6a-3WM3q-
A2FTDNkB1RW29Eq9oM6BAnFloKKal9Tt4loQndQfdgD6tOoEuMs-SY-
Lce0QiULduWu310ZdMOJUUjo8zRkMaXKOQMgIvec_C_spL41XJOZqKCpUY58V
u4I98GBXZU4iPA
```

On se connecte avec un tunnel SSH :

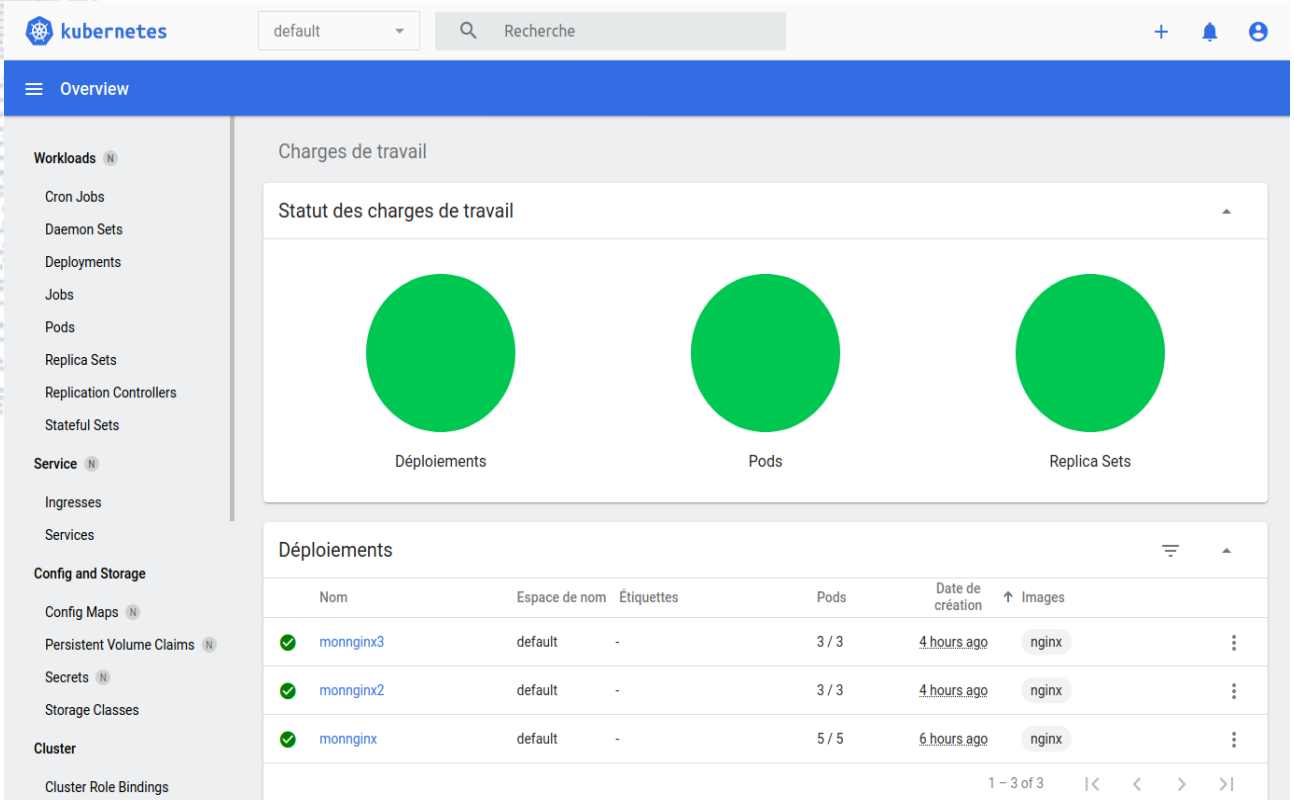
```
root@maitre:~# ssh -L 9999:127.0.0.1:9999 inti@10.3.134.130
```

Il suffit ensuite de se rendre à l'URL du proxy

```
http://127.0.0.1:9999/api/v1/namespaces/kubernetes-
dashboard/services/http:kubernetes-dashboard:/proxy/#/over-
view?namespace=default
```

Pour supprimer les comptes d'accès :

```
root@maitre:~# kubectl -n kubernetes-dashboard delete serviceaccount admin-user
root@maitre:~# kubectl -n kubernetes-dashboard delete clusterrolebinding admin-user
```


kubernetes
default
Recherche
Overview
Workloads N
Cron Jobs
Daemon Sets
Deployments
Jobs
Pods
Replica Sets
Replication Controllers
Stateful Sets
Service N
Ingresses
Services
Config and Storage
Config Maps N
Persistent Volume Claims N
Secrets N
Storage Classes
Cluster
Cluster Role Bindings
Charges de travail
Statut des charges de travail
Déploiements
Pods
Replica Sets
Déploiements
Nom
Espace de nom
Étiquettes
Pods
Date de création
Images
monnginx3 default - 3 / 3 4 hours ago nginx
monnginx2 default - 3 / 3 4 hours ago nginx
monnginx default - 5 / 5 6 hours ago nginx
1 – 3 of 3

3 - Helm

Helm est un gestionnaire de paquets à destination d'un cluster Kubernetes, il facilite ainsi l'installation, la mise à jour, la suppression … Un paquet est nommé un « Chart ».

De la même manière que pour les gestionnaires de paquets traditionnels, l'outil utilise la notion de dépôts. Cependant, il existe un moteur global nommé le « hub » qui permet d'effectuer des recherches sur un grand nombre de dépôts.

L'installation se fait avec les commandes suivantes :

```
root@maitre:~# curl https://baltocdn.com/helm/signing.asc | gpg --dearmor | sudo tee /usr/share/keyrings/helm.gpg > /dev/null
root@maitre:~# echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/helm.gpg] https://baltocdn.com/helm/stable/debian/ all main" | sudo tee /etc/apt/sources.list.d/helm-stable-debian.list
root@maitre:~# apt update
root@maitre:~# apt install helm
```

Pour rechercher sur le « hub » :

```
helm search hub <expressions>
```

Par exemple :

```
root@maitre:~# helm search hub drupal
URL                                                     CHART VERSION   APP VERSION
DESCRIPTION
https://artifacthub.io/packages/helm/bitnami-ak...      12.5.10         9.4.8           Drupal
is one of the most versatile open source...
https://artifacthub.io/packages/helm/bitnami/dr...      17.0.0          10.2.0          Drupal
is one of the most versatile open source...
```

En affichant les dépôts :

```
root@maitre:~# helm search hub drupal --list-repo-url
URL                                                             CHART VERSION   APP VERSION
DESCRIPTION                                                     REPO URL
https://artifacthub.io/packages/helm/bitnami-ak...              12.5.10         9.4.8
Drupal is one of the most versatile open source...
https://marketplace.azurecr.io/helm/v1/repo
https://artifacthub.io/packages/helm/bitnami/dr...              17.0.0          10.2.0
Drupal is one of the most versatile open source...              https://charts.bitnami.com/bitnami
```

On peut donc ajouter le dépôt qui nous intéresse :

```
root@maitre:~# helm repo add bitunami https://charts.bitnami.com/bitnami
"bitunami" has been added to your repositories
```

Pour lister les dépôts ajoutés :

```
root@maitre:~# helm repo list
NAME            URL
bitunami        https://charts.bitnami.com/bitnami
```

On peut maintenant rechercher dans nos dépôts :

```
root@maitre:~# helm search repo drupal
NAME            CHART VERSION   APP VERSION     DESCRIPTION
bitunami/drupal 17.0.0            10.2.0            Drupal is one of the most
versatile open source…
```

Pour installer un « chart », on utilise la fonction « install » en précisant le nom à donner localement :

```
root@maitre:~# helm install mydrup1 bitunami/drupal
NAME: mydrup1
LAST DEPLOYED: Tue Jan  2 10:00:19 2024
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
CHART NAME: drupal
CHART VERSION: 17.0.0
APP VERSION: 10.2.0** Please be patient while the chart is being deployed **

1. Get the Drupal URL:
  NOTE: It may take a few minutes for the LoadBalancer IP to be available.
        Watch the status with: 'kubectl get svc --namespace default -w mydrup1-drupal'

  export SERVICE_IP=$(kubectl get svc --namespace default mydrup1-drupal --template "{{ range
(index .status.loadBalancer.ingress 0) }}{{ . }}{{ end }}")
  echo "Drupal URL: http://$SERVICE_IP/"

2. Get your Drupal login credentials by running:
  echo Username: user
     echo   Password:   $(kubectl   get   secret   --namespace   default   mydrup1-drupal   -o
jsonpath="{.data.drupal-password}" | base64 -d)
```

ATTENTION : Cette installation va demander de passer par un PVC pour obtenir un espace de stockage.

Pour désinstaller :

```
root@maitre:~# helm uninstall mydrup1
release "mydrup1" uninstalled
```

Il est possible de consulter les paramètres avant l'installation :

```
root@maitre:~# helm show values bitunami/drupal
...
global:
  imageRegistry: ""
  ## E.g.
  ## imagePullSecrets:
  ##   - myRegistryKeySecretName
  ##
  imagePullSecrets: []
  storageClass: ""
...
```

Ceci permet de préciser des valeurs lors de la phase d'installation. Par exemple, si nous souhaitons préciser le nom de la base et le nom de l'utilisateur :

```
echo '{mariadb.auth.database: user0db, mariadb.auth.username: user0}' > values.yaml
```

Puis :

```
helm install -f values.yaml bitnami/drupal –generate-name
```

Il est possible également d'indiquer des valeurs ponctuelles avec l'argument « --set name = value ».

4 - Ingress

4.1 - Introduction

Ingress permet de manipuler les accès depuis le réseau publique au niveau de la couche applicative, en ce qui nous concerne au niveau du protocole HTTP et HTTPS.

Pour cela il faut tout d'abord ajouter un controlleur Ingress, puis seulement après associer des ressources de type règles à ce contrôleur. Le projet Kubernetes maintient 3 principaux contrôleurs à savoir AWS, GCE et nginx mais d'autres contrôleurs tiers sont disponibles comme Traefik, Kong, Voyager, HAProxy, … (https://kubernetes.io/docs/concepts/services-networking/ingress-controllers/)

Les ressources basées sur des règles et s'appuyant donc sur des chemins, utiliseront les services de type « NodePort » ou « LoadBalancer ».

Nous utiliserons ici un contrôleur simple de type « Nginx ».

4.2 - Installation du contrôleur

```
root@maitre:~# helm search hub ingress-nginx
URL                                                 CHART VERSION   APP VERSION     DESCRIPTION
https://artifacthub.io/packages/helm/ingress-ng...      4.9.0 1.9.5             Ingress controller for
Kubernetes using NGINX a…
```

Pour afficher les dépôts avec :

```
root@maitre:~# helm search hub ingress-nginx --list-repo-url
URL                                                 CHART VERSION   APP VERSION     DESCRIPTION
REPO URL
https://artifacthub.io/packages/helm/ingress-ng...      4.9.0           1.9.5 Ingress controller for
Kubernetes using NGINX a... https://kubernetes.github.io/ingress-nginx
```

Pour l'installer :

```
root@maitre:~#    helm    install    ingress-nginx    ingress-nginx        --repo
https://kubernetes.github.io/ingress-nginx --namespace ingress-nginx --create-
namespace
```

On vérifie sur les « pods » en spécifiant le « namespace » spécifique :

```
root@master:~# kubectl get pods -o wide --namespace=ingress-nginx
NAME                                        READY   STATUS      RESTARTS   AGE     IP            NODE
NOMINATED NODE   READINESS GATES
ingress-nginx-admission-create-n6qcx        0/1     Completed   0          7m50s   10.244.1.8    slave2
<none>           <none>
ingress-nginx-admission-patch-zvznp         0/1     Completed   0          7m50s   10.244.1.9    slave2
<none>           <none>
ingress-nginx-controller-8574b6d7c9-7twbb   1/1     Running     0 7m50s    10.244.2.7    slave1    <none>
<none>
```

On vérifie dans les services en spécifiant le « namespace » spécifique :

```
root@master:~# kubectl get services -o wide --namespace=ingress-nginx
NAME                                 TYPE           CLUSTER-IP     EXTERNAL-IP PORT(S)                      AGE SELECTOR
ingress-nginx-controller             LoadBalancer 10.107.155.228 <pending>   80:31507/TCP,443:30870/TCP  8m58s
    app.kubernetes.io/component=controller,app.kubernetes.io/instance=ingress-nginx,app.kubernetes.io/name=ingress-nginx
ingress-nginx-controller-admission   ClusterIP 10.97.219.90  <none>   443/TCP                     8m58s
    app.kubernetes.io/component=controller,app.kubernetes.io/instance=ingress-nginx,app.kubernetes.io/name=ingress-nginx
```

Si le service est « pending » il faut forcer l'IP en editant le service :

```
root@master:~# kubectl edit svc ingress-nginx-controller --namespace=ingress-nginx
...
  type: LoadBalancer
  externalIPs:
  - 10.3.134.XXX
...
```

On vérifie après :

```
root@master:~# kubectl get services -o wide --namespace=ingress-nginx
NAME                       TYPE           CLUSTER-IP       EXTERNAL-IP    PORT(S)          ...
ingress-nginx-controller   LoadBalancer   10.107.155.228   10.3.134.XXX   80:31507/TCP,443:30870/TCP ...
```

On peut tester la réponse de notre Ingress en HTTP :

```
root@master:~# curl www.130.ccm.u13.org
<html>
<head><title>404 Not Found</title></head>
<body>
<center><h1>404 Not Found</h1></center>
<hr><center>nginx</center>
</body></html>
```

4.3 - Configuration des certificats

Il est possible d'installer un service dédié qui va générer les certificats à la demande pour une utilisation en local. On va créer le « namespace » :

```
root@maitre:~# kubectl create namespace cert-manager
```

On ajoute le dépôt :

```
root@maitre:~# helm repo add jetstack https://charts.jetstack.io
```

On installe via « helm » :

```
root@maitre:~# helm install cert-manager jetstack/cert-manager --namespace cert-manager --set installCRDs=true
```

On vérifie :

```
root@maitre:~# kubectl get pods --namespace cert-manager
```

4.4 - Utilisation de la ressource

On va créer une ressource Ingress basé sur le chemin « /monsrvphp » et qui utilise le service du même nom créé précédemment :

```
root@master:~# nano ingress1.yml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: myingress1
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  ingressClassName: nginx
  rules:
  - host: "www.130.ccm.u13.org"
    http:
      paths:
      - path: /monsrvphp
        pathType: Prefix
        backend:
          service:
            name: monservice
            port:
              number: 3080
```

On va déployer la ressource et on vérifie les ressources de type Ingress : :

```
root@maitre:~# kubectl create -f ingress1.yml
service/monservice2 created
root@master:~# kubectl get -o wide ingress
NAME          CLASS    HOSTS                    ADDRESS         PORTS
AGE
myingress1    nginx    www.130.ccm.u13.org      10.3.134.130    80
71m
```

On va également demander des informations supplémentaires :

```
root@master:~# kubectl describe ingress myingress1
Name:             myingress1
Labels:           <none>
Namespace:        default
Address:          10.3.134.130
Ingress Class:    nginx
Default backend:  <default>
Rules:
  Host                Path  Backends
  ----                ----  --------
  www.130.ccm.u13.org  /monsrvphp   monsrvphp:3080 (10.244.1.6:80,10.244.2.5:80)
Annotations:          nginx.ingress.kubernetes.io/rewrite-target: /
Events:               <none>
```

On peut tester la réponse de notre Ingress en HTTP :

```
root@master:~# curl www.130.ccm.u13.org
<html>
<head><title>404 Not Found</title></head>
<body>
<center><h1>404 Not Found</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

On peut maintenant tester le chemin lié à la ressource :

```
root@master:~# curl www.130.ccm.u13.org/monsrv
hostname : monphp-f58ff9c-djhk9
IP server : 10.244.2.5
IP client : 10.244.2.7
X-Forwarded-for: 10.244.0.0
PHP Version : 7.3.19-1~deb10u1
```

5 - Rancher

Rancher est l'interface d'administration la plus utilisée à ce jour sur Kubernetes, nous allons donc voir comment l'installer.

On ajhoute le dépôt et on va créé le « namespace » :

```
root@maitre:~# helm repo add rancher-stable https://releases.rancher.com/server-charts/stable
kubectl create namespace cattle-system
```

```
root@maitre:~# helm install rancher rancher-stable/rancher --namespace cattle-system --set hostname=rancher.187.ccm.u13.org --set bootstrapPassword=MotDePasseEnClair
NAME: rancher
LAST DEPLOYED: Mon Jan  8 16:23:19 2024
NAMESPACE: cattle-system
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
Rancher Server has been installed.
```

````
NOTE: Rancher may take several minutes to fully initialize. Please standby while Certificates are being issued, Containers are started and the Ingress rule comes up.

Check out our docs at https://rancher.com/docs/

If you provided your own bootstrap password during installation, browse to https://rancher.187.ccm.u13.org to get started.

If this is the first time you installed Rancher, get started by running this command and clicking the URL it generates:

```

echo https://rancher.187.ccm.u13.org/dashboard/?setup=$(kubectl get secret --namespace cattle-system bootstrap-secret -o go-template='{{.data.bootstrapPassword|base64decode}}')
```


To get just the bootstrap password on its own, run:

```

kubectl get secret --namespace cattle-system bootstrap-secret -o go-template='{{.data.bootstrapPassword|base64decode}}{{ "\n" }}'
```


Happy Containering!
````

```
root@maitre:~# kubectl -n cattle-system rollout status deploy/rancher
Waiting for deployment "rancher" rollout to finish: 0 of 3 updated replicas
are available...
Waiting for deployment "rancher" rollout to finish: 1 of 3 updated replicas
are available...
Waiting for deployment "rancher" rollout to finish: 2 of 3 updated replicas
are available...
deployment "rancher" successfully rolled out
```

```
root@maitre:~# kubectl -n cattle-system get deploy rancher
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
rancher   3/3     3            3           3m1s
```

Pour l'intégration Ingress, on vérifie :

```
root@maitre:~# kubectl get ingress --all-namespaces
NAMESPACE       NAME          CLASS    HOSTS                     ADDRESS         PORTS     AGE
cattle-system   rancher       <none>   rancher.187.ccm.u13.org                   80, 443   6m20s
default         myingress1    nginx    www.187.ccm.u13.org       10.3.134.187    80        42m
```

Comme il n'y a pas d'adresse IP associée, il faut éditer la règle Ingress pour spécifier le type :

```
root@maitre:~# kubectl edit ingress rancher -n cattle-system
...
spec:
  ingressClassName: nginx
  rules:
  - host: rancher.187.ccm.u13.org
    http:
    ...
```

On vérifie à nouveau :

```
root@maitre:~# kubectl get ingress --all-namespaces
NAMESPACE       NAME          CLASS   HOSTS                      ADDRESS         PORTS     AGE
cattle-system   rancher       nginx   rancher.187.ccm.u13.org    10.3.134.187    80, 443   8m56s
default         myingress1    nginx   www.187.ccm.u13.org        10.3.134.187    80        44m
```

Il aurait été possible également de l'indiquer dans la phase d'installation :

```
root@maitre:~# helm install rancher rancher-stable/rancher --namespace cattle-system --set hostname=www.187.ccm.u13.org --set bootstrapPassword=Troup50 --set ingress.ingressClassName=nginx
```

On peut tester la route en http :

```
root@maitre:~# curl -I rancher.187.ccm.u13.org/
HTTP/1.1 308 Permanent Redirect
Date: Mon, 08 Jan 2024 16:34:25 GMT
Content-Type: text/html
Content-Length: 164
Connection: keep-alive
Location: https://rancher.187.ccm.u13.org
```

Il n'y a plus qu'à se rendre à l'URL en https dans un navigateur :