WDATP - Microservices Academy

Lab2. Deploy and Access your service

Requirements:

<https://hub.docker.com/editions/community/docker-ce-desktop-windows>

Comment:

Hyper-v should be enable if not please run as admin:

Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Hyper-V -All

Overview**:**

In this lab you are going to deploy 2 services to your local Kubernetes cluster. After it – you’ll setup access to your services by using ingress controller. If you’ll have time – you have bonus advanced steps – deploying with helm.

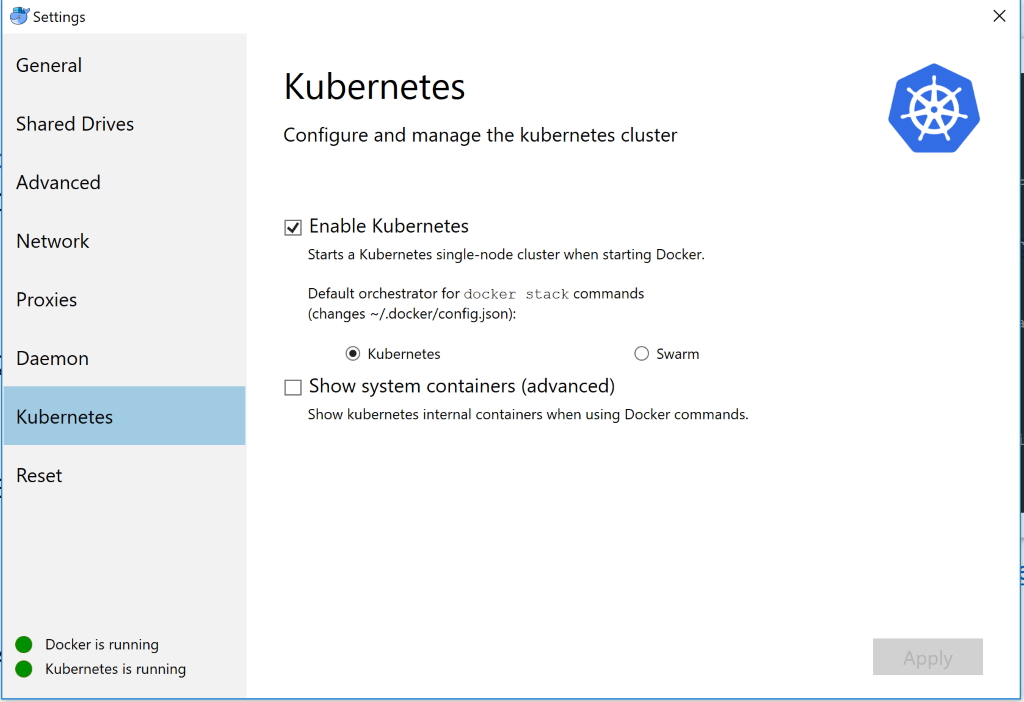
**Note:**If you are connecting to any of Kubernetes clusters today, please, **backup your kube.config** (usually located at *C:\Users\<user>\.kube* folder)

**Task #1 -**Enable Kubernetes in Docker

*Goal: Make sure kubernetes cluster is running locally:*

***Note****: If you already did it in the yesterday lab, you may skip this step, but maybe find useful to reset the cluster to start fresh.*

* Right click on Docker on windows – open settings
* Enable Kubernetes



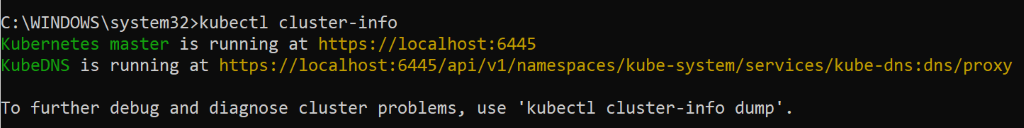
* Should take up to 10 min after clicking Apply – if it takes more, please see trouble shooting
* Check Kubernetes is running in Docker Icon -> Right Click -> Settings

C:\Users\orgolan\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\AABFE122.tmp

1. Verify installation by running (On Cmd)

***kubectl cluster-info***

1. You should receive an output like this -

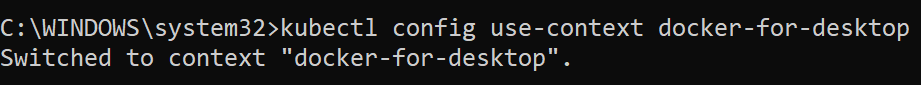


* **[Advanced – *you can skip this or return to this part later*]**

Launch Kubernetes dashboard locally

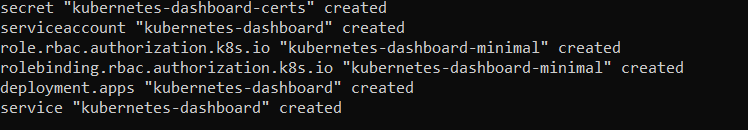
Step #0: Switch to local cluster

***kubectl config use-context docker-for-desktop***



Step #1: install dashboard 

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



Step #2:

Create service account file (filename: admin-service.yaml) with the content below:

apiVersion: v1

kind: ServiceAccount

metadata:

  name: admin-user

  namespace: kube-system

Run the command

***kubectl create -f admin-service.yaml***

C:\Users\orgolan\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\3A62AEBA.tmp

Step #3:

Create cluster role and cluster role binding (filename cluster-role.yaml)

Run the command

***kubectl create -f cluster-role.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

Step #5: (get the user secret to authenticate dashboard website)

Run the command

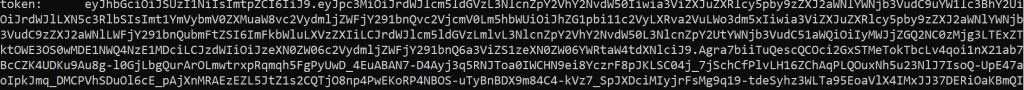
***kubectl -n kube-system get secret | findstr admin-user***

C:\Users\orgolan\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\F0B996D8.tmp

Step #4: (get the secret name and run the following command)

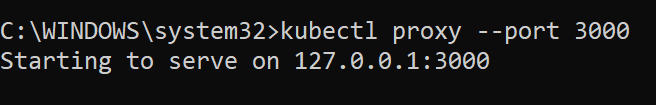
***kubectl -n kube-system describe secret <secret name>***

Copy the token



Step #5:

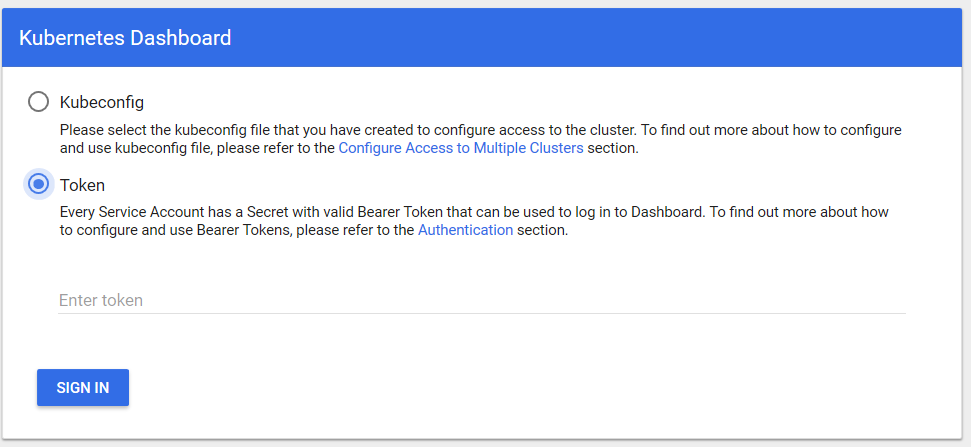
***Run the command: kubectl proxy --port 3000***



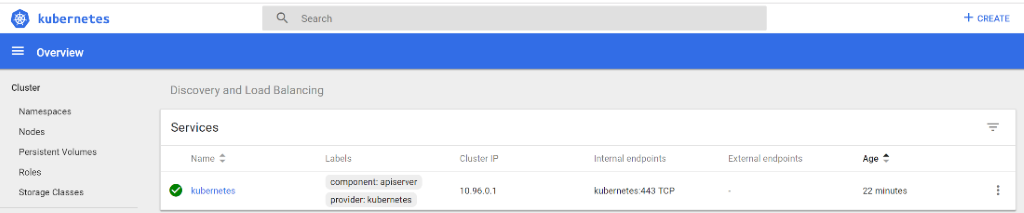
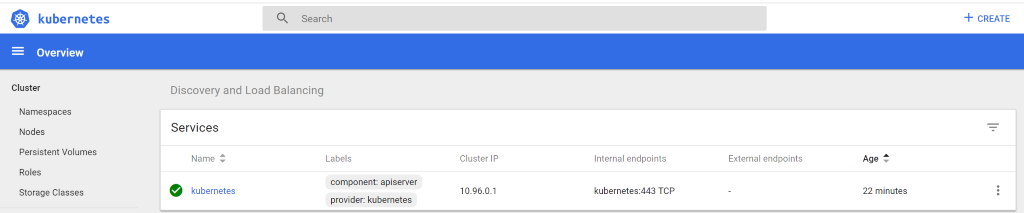
Open in browser –

[http://localhost:3000/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/#!/overview?namespace=default](https://storage.googleapis.com/kubernetes-helm/helm-v2.11.0-windows-amd64.zip#!/overview?namespace=default)

Select token and copy the value of the token from step #6



***Note:*** Works in Chrome/firefox



**Task #2** – Create namespace with kubectl command

*Goal: Create yaml file for namespace creation (will be re-used later in the lab)*

1. Create a folder and create a file *namespace.yaml*

**apiVersion: v1**

**kind: Namespace**

**metadata:**

**name: mdatp**

       Run the command (assuming you in the same folder of the yaml file)

***kubectl create -f namespace.yaml***

(expected result namespace " *mdatp*" created)

C:\Users\orgolan\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\8DEB837C.tmp

1. Validate:

***kubectl get namespaces***

(should give the ability to see your namespace name.)

**Task #3** – Create deployment based on your container in docker-hub

*Goal: Manually deploy create and deploy multiple kubernetes objects*

1. Copy the text below to *deployment.yaml*

apiVersion: extensions/v1beta1

kind: Deployment

metadata:

  name: coffee

  namespace: mdatp

spec:

  replicas: 2

  selector:

    matchLabels:

      app: coffee

  template:

    metadata:

      labels:

        app: coffee

    spec:

      containers:

      - name: coffee

        image: nginxdemos/hello:plain-text

        ports:

        - containerPort: 80

---

apiVersion: v1

kind: Service

metadata:

  name: coffee-svc

  namespace: mdatp

spec:

  ports:

  - port: 80

    targetPort: 80

    protocol: TCP

    name: http

  selector:

    app: coffee

---

apiVersion: extensions/v1beta1

kind: Deployment

metadata:

  name: tea

  namespace: mdatp

spec:

  replicas: 3

  selector:

    matchLabels:

      app: tea

  template:

    metadata:

      labels:

        app: tea

    spec:

      containers:

      - name: tea

        image: nginxdemos/hello:plain-text

        ports:

        - containerPort: 80

---

apiVersion: v1

kind: Service

metadata:

  name: tea-svc

  namespace: mdatp

  labels:

spec:

  ports:

  - port: 80

    targetPort: 80

    protocol: TCP

    name: http

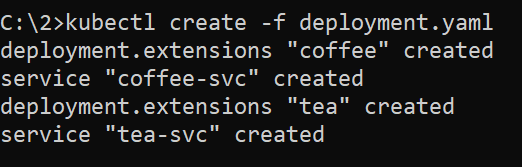
  selector:

    app: tea

1. Run the command:

***kubectl create -f deployment.yaml***

Expected result:



**Task #4** – Validate deployment

*Goal: Validate deployed kubernetes objects*

Check if all exists in the namespace by running the command:

* For pods:

***kubectl get pods -n mdatp***

* For services:

***kubectl get svc -n mdatp***

* For deployment:

***kubectl get deployment -n mdatp***

* *Mdatp* is the selected namespace:

***kubectl get all -n mdatp***

**Using Helm**

**Task #5** – install Helm:

*Goal: Install and init helm, both locally and in the cluster:*

1. Download Helm for windows: [https://storage.googleapis.com/kubernetes-helm/helm-v2.11.0-windows-amd64.zip](http://localhost:3000/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/)
2. Extract to folder, and add the folder created to environment variables (path).
3. Run the command:

***helm version***

expected result: client version should be V2.11.0

you can get warning *Error: “could not find tiller”* – its ok, just continue

* Init local helm repo and install helm on Kubernetes cluster

***helm init***

You should get message “Happy Helming!”

* Wait 30 seconds and run ***helm list*** (list of helm deployments)

***helm list***

*At this point – the list will be empty, since not releases were installed*

**Task #6** – clean deployment

*Goal:* Cleanup

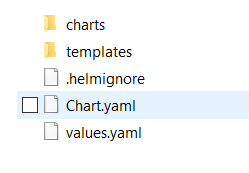
***kubectl delete namespace mdatp***

**Task #7** – create all with Helm

*Goal: Create helm project:*

* Run the command:

***helm create myfirsthelm***

A folder ***myfirsthelm*** should be created with the following files/directories:

* Delete all files from the template folder, ***except \_helpers.tpl***
* Copy the *namespace.yaml* and the *deployment.yaml* into the templates folder *(you created those files on Tasks #2 and #3)*
* Validate your chart with ***lint*** command

***helm lint myfirsthelm***

* Run the ***install*** command:

***helm install myfirsthelm myfirsthelm***

* Check if the namespace created with all the pods and services inside.

***kubectl get ns***

***kubectl -n mdatp get all***

* Check the helm name by running ***list*** command:

***helm list***

**Task #8**– Helm Delete

*Goal: Cleanup and remove previously installed helm package*

* Get the list of the helm deployment by running the command (#will show all the packages installed):

***helm list -a***

* Delete the installed package by running ***delete*** command:

***helm del --purge myfirstrelease***

**Task #9** – User parameters to customize helm package

*Goal: Move namespace to be consumed from “parameters” file (values\*\*\*.yaml)*

* Replace the *mdatp* into *{{ .Values.namespace }}* in the *deployment.yaml*

apiVersion: extensions/v1beta1

kind: Deployment

metadata:

  name: coffee

  namespace: {{ .Values.namespace }}

spec:

  replicas: 2

  selector:

    matchLabels:

      app: coffee

  template:

    metadata:

      labels:

        app: coffee

    spec:

      containers:

      - name: coffee

        image: nginxdemos/hello:plain-text

        ports:

        - containerPort: 80

---

apiVersion: v1

kind: Service

metadata:

  name: coffee-svc

  namespace: {{ .Values.namespace }}

spec:

  ports:

  - port: 80

    targetPort: 80

    protocol: TCP

    name: http

  selector:

    app: coffee

---

apiVersion: extensions/v1beta1

kind: Deployment

metadata:

  name: tea

  namespace: {{ .Values.namespace }}

spec:

  replicas: 3

  selector:

    matchLabels:

      app: tea

  template:

    metadata:

      labels:

        app: tea

    spec:

      containers:

      - name: tea

        image: nginxdemos/hello:plain-text

        ports:

        - containerPort: 80

---

apiVersion: v1

kind: Service

metadata:

  name: tea-svc

  namespace: {{ .Values.namespace }}

  labels:

spec:

  ports:

  - port: 80

    targetPort: 80

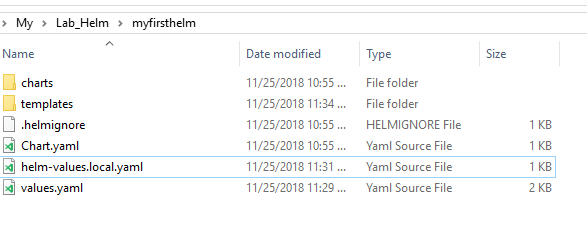
    protocol: TCP

    name: http

  selector:

    app: tea

* Create file *helm-values.local.yaml*in the chart’s root folder:



* Set the content of the *helm-values.local.yaml to be:*

namespace: mdatp

* Install the package:

***helm install .\myfirsthelm\ --name myfirstrelease -f .\myfirsthelm\helm-values.local.yaml***

* Validate what found in the namespace

***kubectl get all -n mdatp***

**Canary deployment**

**Task #10** – Add ingress Controller

*Goal: Setup ingress controller as a cluster “entry point”*

* **Deploy nginx-controller**

***kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/nginx-0.22.0/deploy/mandatory.yaml***

* **Create service – copy the text below to file name*ingress-nginx-service.yaml***

apiVersion: v1

kind: Service

metadata:

  name: ingress-nginx

  namespace: ingress-nginx

  labels:

    app.kubernetes.io/name: ingress-nginx

    app.kubernetes.io/part-of: ingress-nginx

spec:

  type: NodePort

  ports:

    - name: http

      port: 80

      targetPort: 80

      protocol: TCP

    - name: https

      port: 443

      targetPort: 443

      protocol: TCP

  selector:

    app.kubernetes.io/name: ingress-nginx

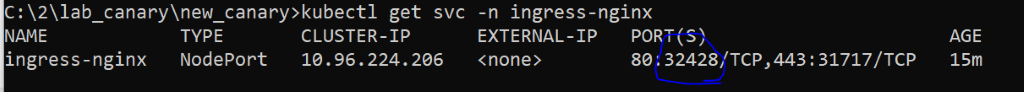
    app.kubernetes.io/part-of: ingress-nginx

* **Run the command**

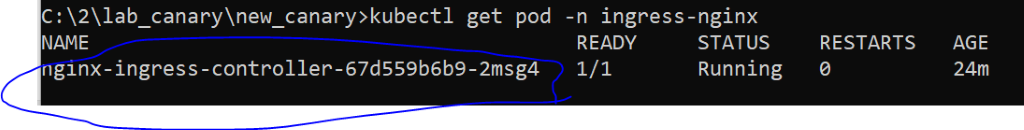
***kubectl apply -f ingress-nginx-service.yaml***

* Get the Nginx SVC port

***kubectl get svc -n ingress-nginx***



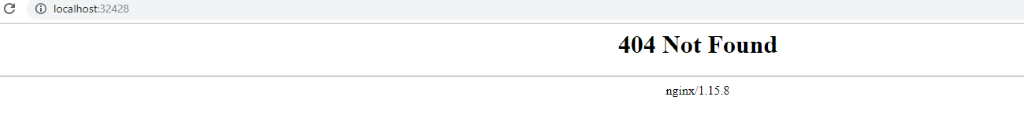
***kubectl get pod -n ingress-nginx***



***kubectl port-forward -n ingress-nginx nginx-ingress-controller-67d559b6b9-2msg4 8080:80***

C:\Users\orgolan\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\A92C9C54.tmp

**Expected result:**



* **Create ingress – copy the text below to file name ingress.yaml**

apiVersion: extensions/v1beta1

kind: Ingress

metadata:

  name: coffee-svc

  namespace: mdatp

  annotations:

    kubernetes.io/ingress.class: nginx

    nginx.ingress.kubernetes.io/rewrite-target: /$1

spec:

  rules:

  - host: cafe.com

    http:

      paths:

      - backend:

          serviceName: coffee-svc

          servicePort: 80

        path: /?(.\*)

---

apiVersion: extensions/v1beta1

kind: Ingress

metadata:

  name: tea-svc

  namespace: mdatp

  annotations:

    kubernetes.io/ingress.class: nginx

    nginx.ingress.kubernetes.io/canary: "true"

    nginx.ingress.kubernetes.io/canary-by-header: "canary"

    nginx.ingress.kubernetes.io/rewrite-target: /$1

spec:

  rules:

  - host: cafe.com

    http:

      paths:

      - backend:

          serviceName: tea-svc

          servicePort: 80

        path: /?(.\*)

* Run the command:

***kubectl apply -f ingress.yaml***

* Add the row

***127.0.0.1 cafe.com***

To the host file (C:\Windows\System32\drivers\etc\hosts)

Pro tip:  😊   
you can use this command to update the file – make sure you are running it from elevated mode:

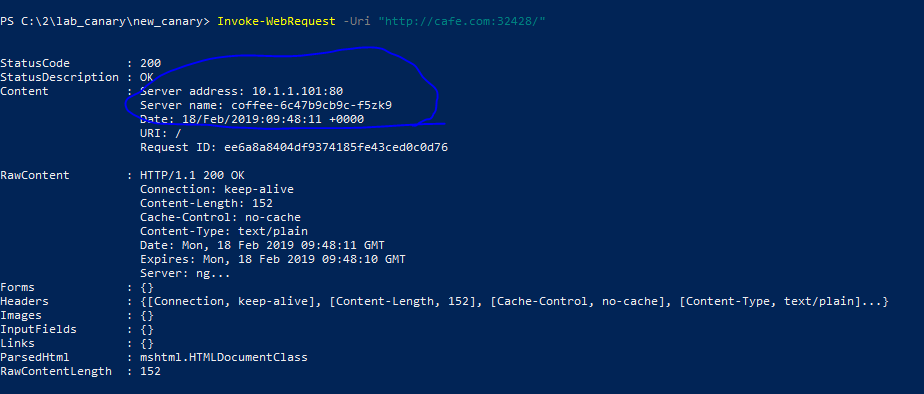
***echo 127.0.0.1 cafe.com >> C:\Windows\System32\drivers\etc\hosts***

**Task #11** – **Check canary (PowerShell or Postman):**

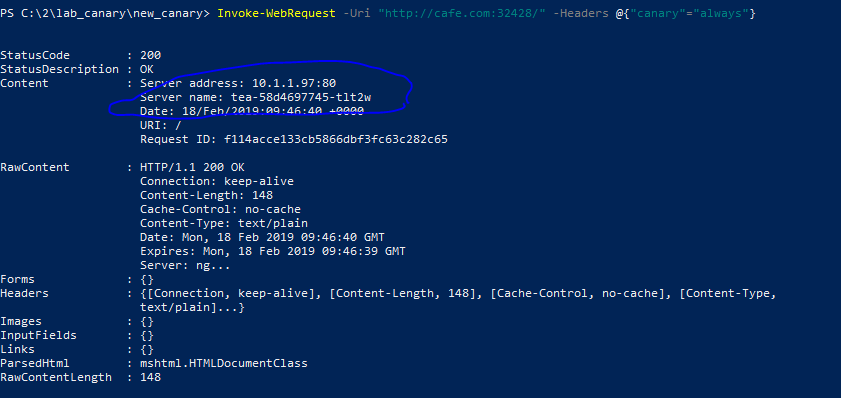
*Goal: Verify canary deployment*

* **PowerShell:**

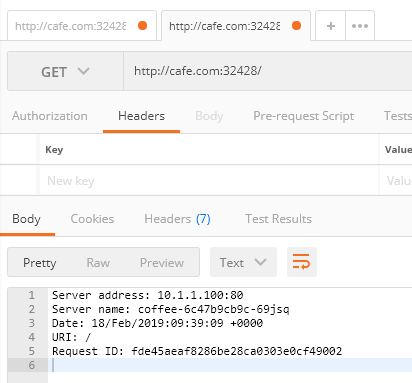
***Invoke-WebRequest -Uri "http://cafe.com:<port>/"***

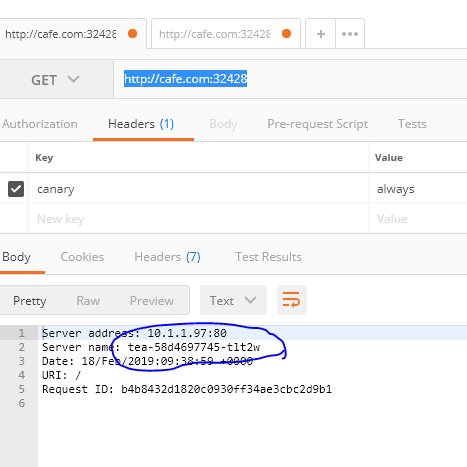


***Invoke-WebRequest -Uri "http://cafe.com:<port>/" -Headers @{"canary"="always"}***



* **Postman:**





**Note:**At this time, you can cleanup your *hosts*