**Sample Deployment Tracker**

1. Sample deployment yml code:

# Defining a Service to expose the helloworld application through NodePort

apiVersion: v1

kind: Service

metadata:

  name: helloworld-svc-12272021

  labels:

    run: helloworld-example

spec:

  ports:

  - port: 8080

    name: tcp-helloworld

    protocol: TCP

    targetPort: 8080

  type: NodePort

  selector:

    run: helloworld-example

---

# Defining a Deployment for helloworld application

apiVersion: apps/v1

kind: Deployment

metadata:

  name: helloworld12272021

spec:

  replicas: 1

  selector:

    matchLabels:

      run: helloworld-example

  template:

    metadata:

      labels:

        run: helloworld-example

    spec:

      containers:

      - name: helloworld

        image: hub.docker.hpecorp.net/caasonhpe/nodejs\_k8s

        ports:

        - containerPort: 8080

          protocol: TCP

        resources:

          requests:

            memory: "100Mi"

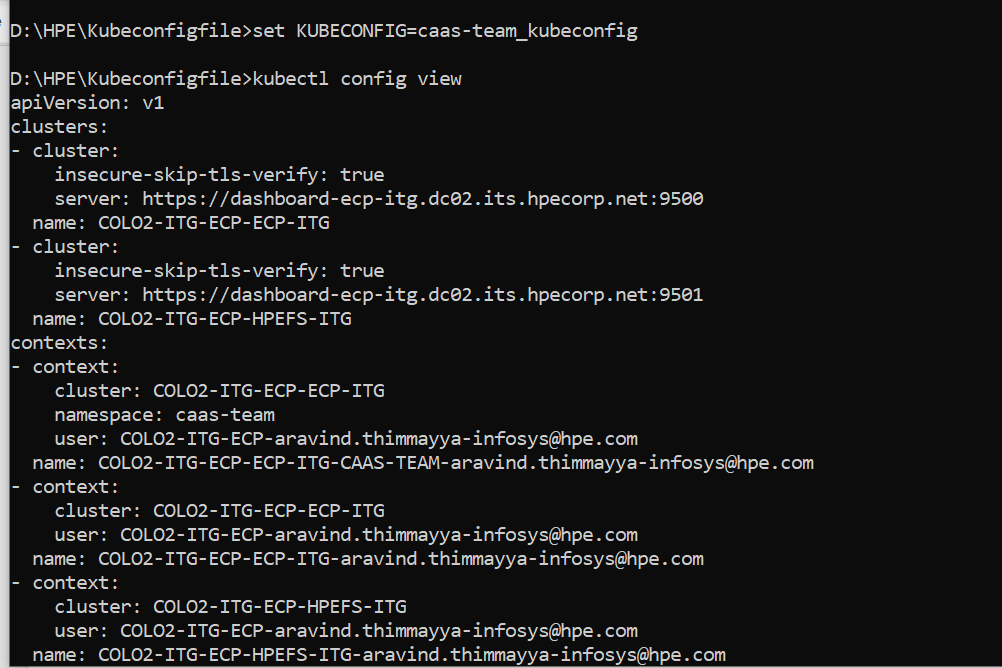
            cpu: "100m"

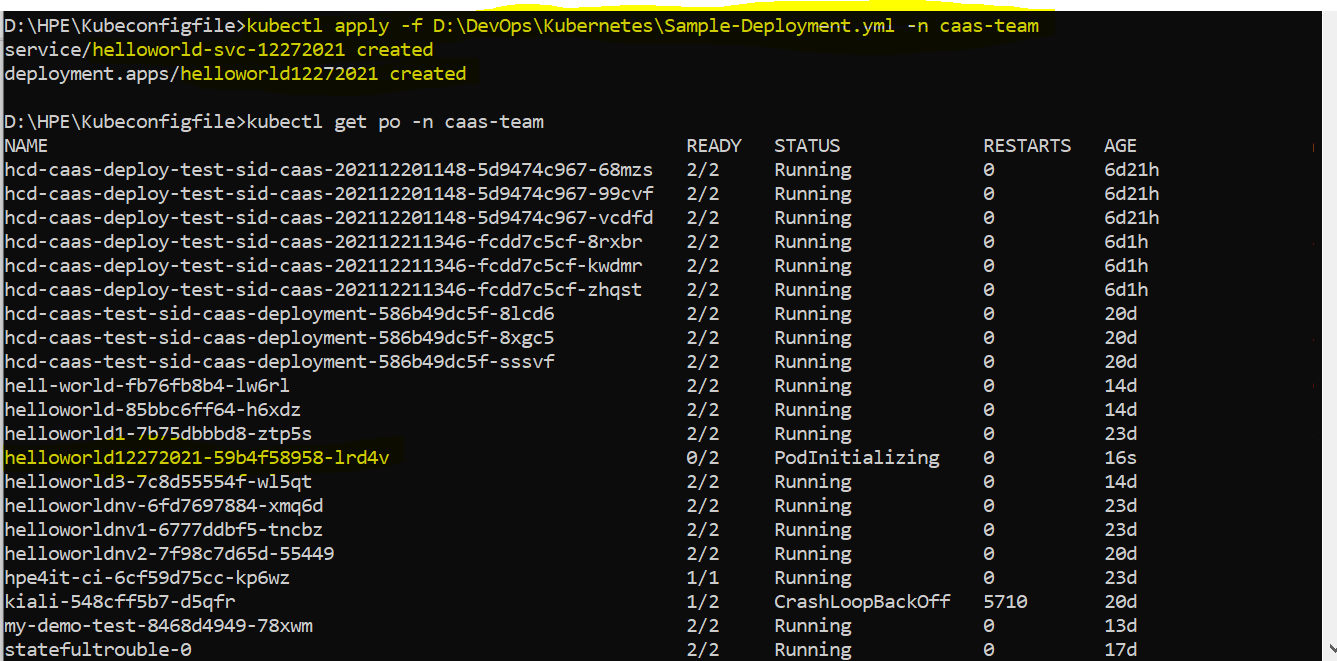
          limits:

            memory: "100Mi"

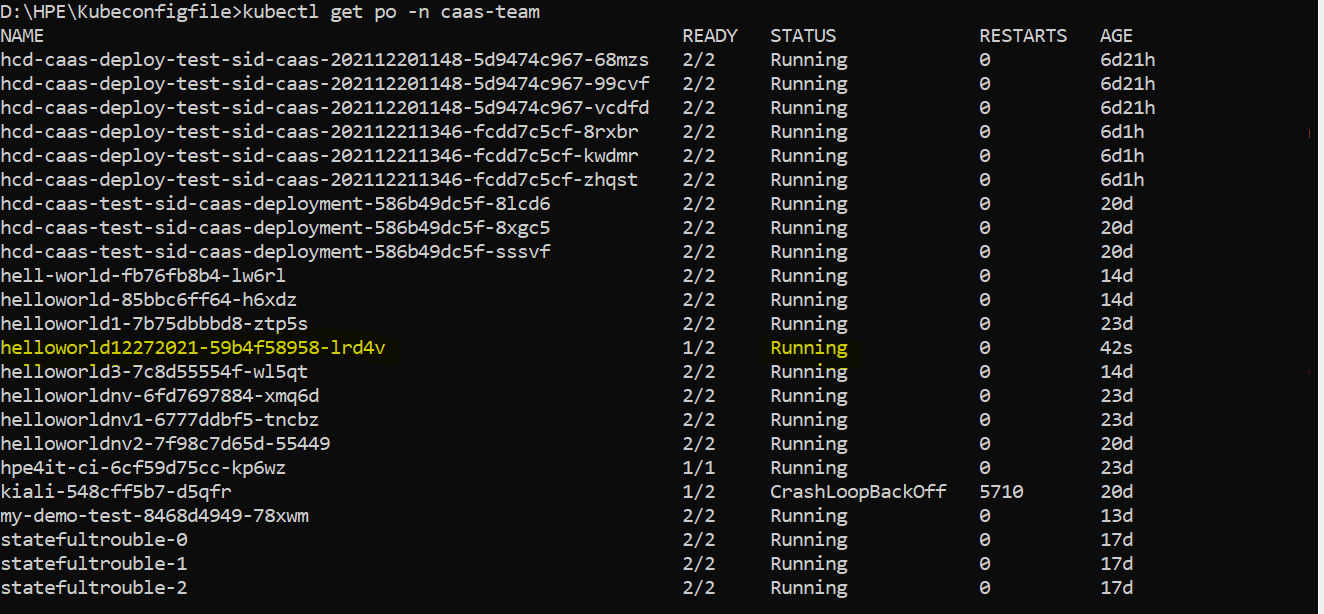
            cpu: "100m"

1. Download kubeconfig file from environment and login to environment through cli and apply deployment yml file to create pod.



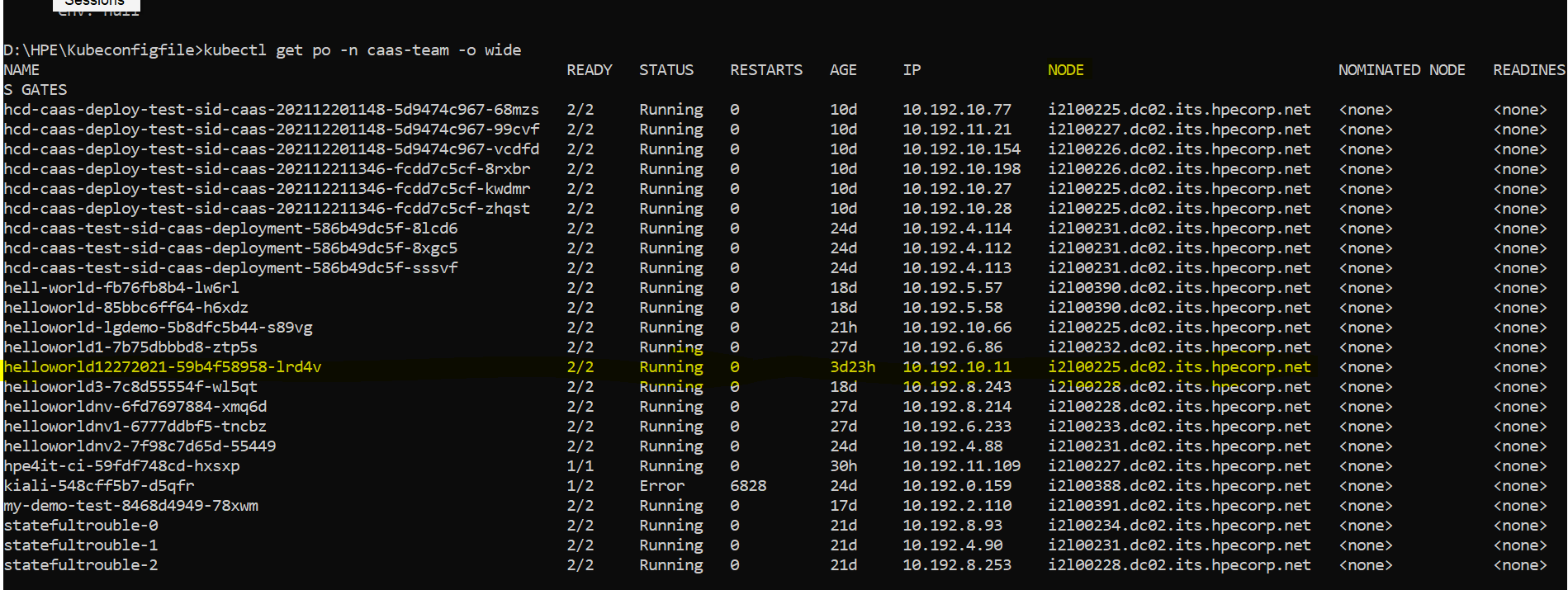


* As I mentioned the deployment name helloworld12272021 and service name helloworld-svc-12272021 has been create both, we can observe in the above screenshot.
* Initially pod is in initializing state once it gets created, we can see that in running state refer below screenshot.

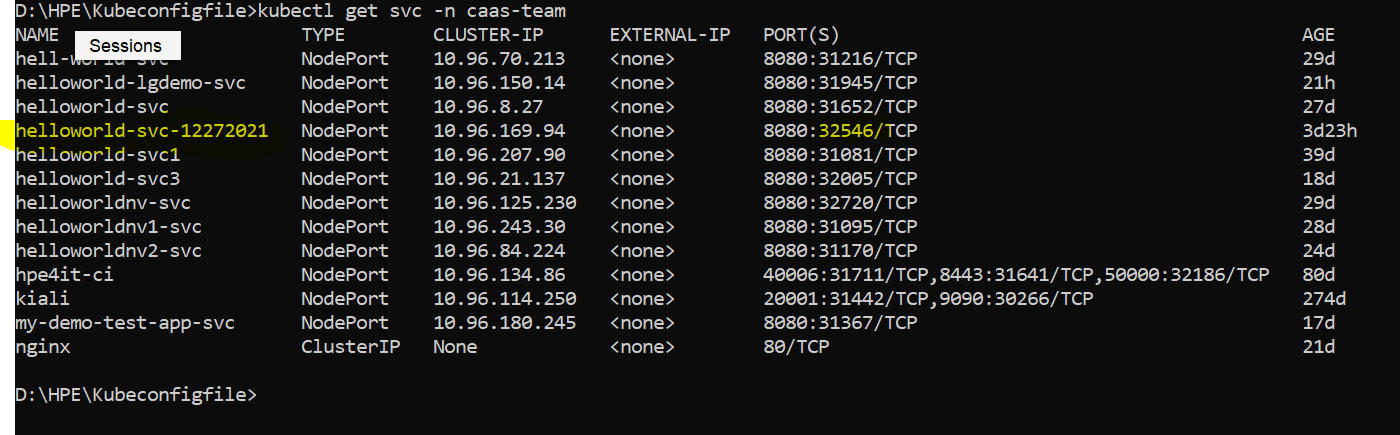


If Pod and service gets created successfully then try to access the app through node with nodeport number.

# kubectl get po -n caas-team -o wide

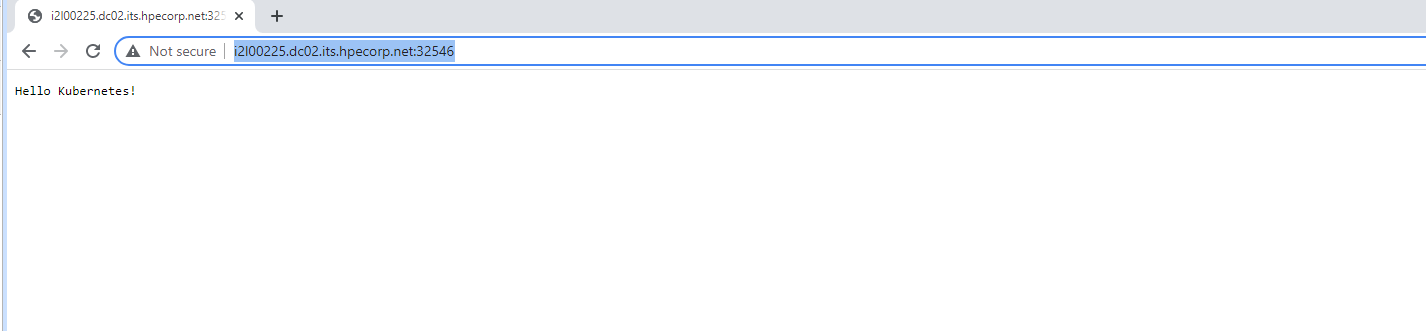


# kubectl get svc -n caas-team



Now we must combine both and hit in google url from colo2 jump host.

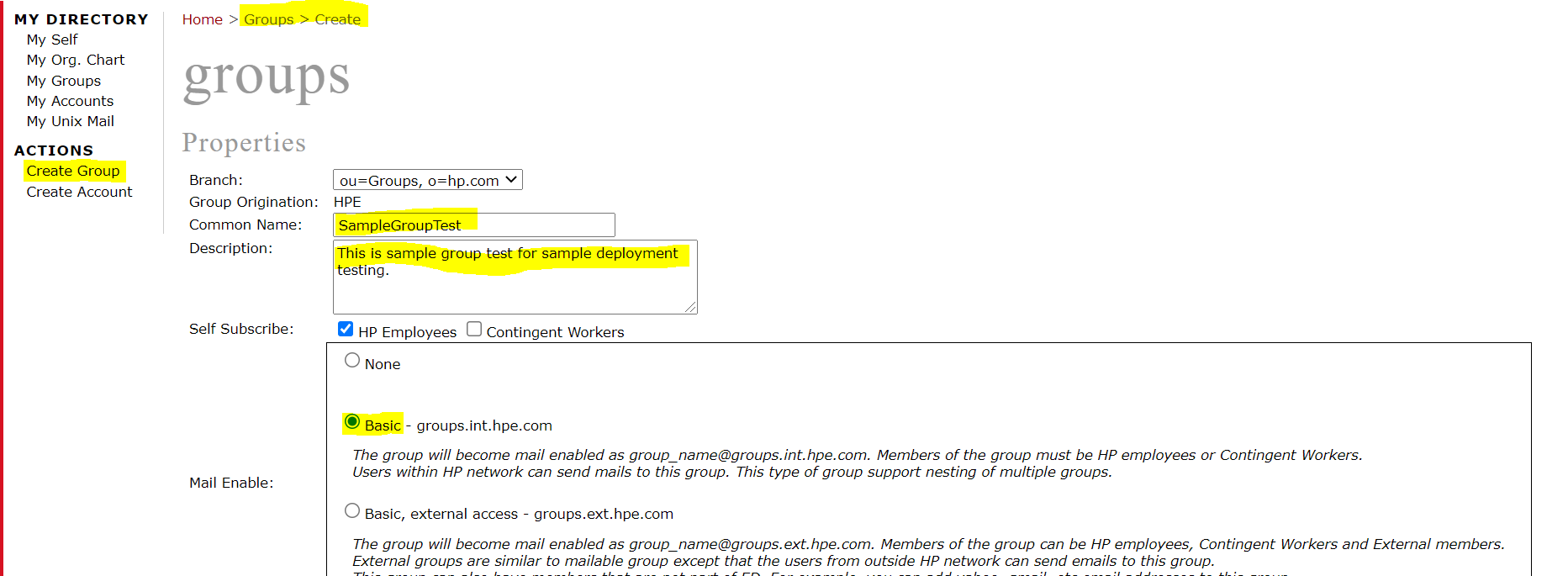
**# http://i2l00225.dc02.its.hpecorp.net:32546/**

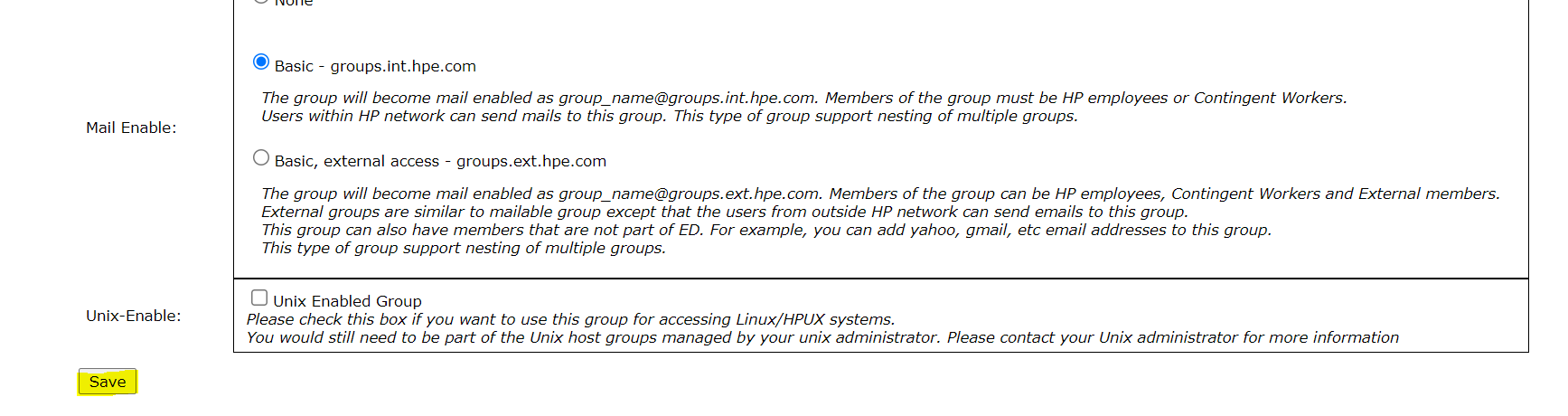


**TLS Passthrough Deployment:**

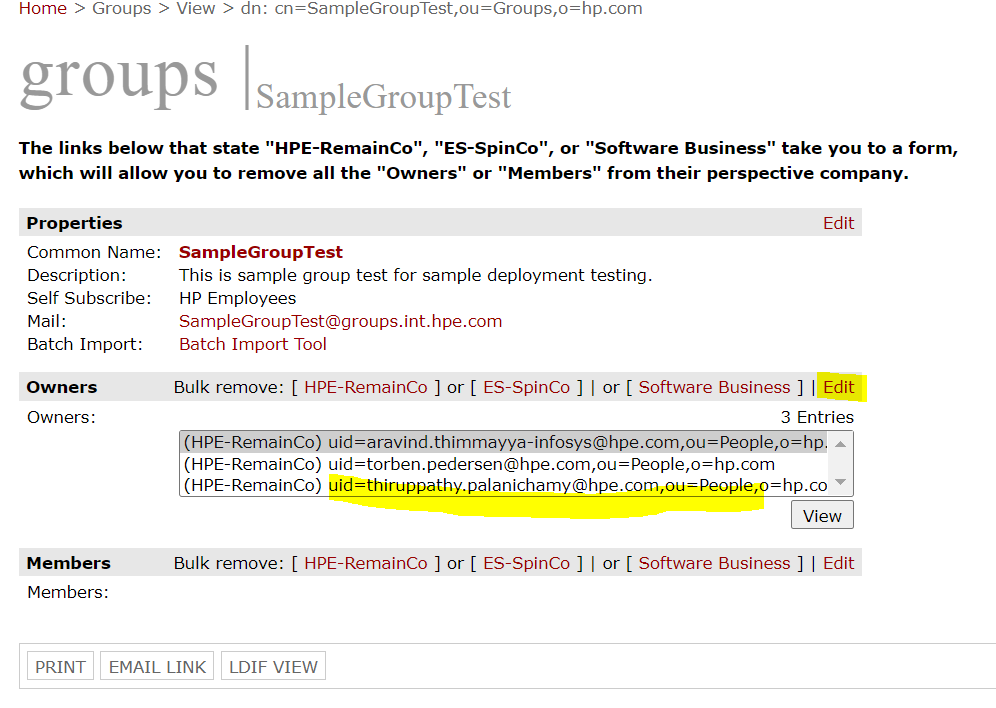
1. Sample deployment code copied and edited asper the app name.
2. To deploy app with TLS passthrough mode we need SSL cert for the DNS and place it as a secrete in our namespace.
3. To create SSL cert, we should have a group in Directory Works(we can not create cert as individual should include tourben and thiru) so creating group for deployment using below link (add group members to it).

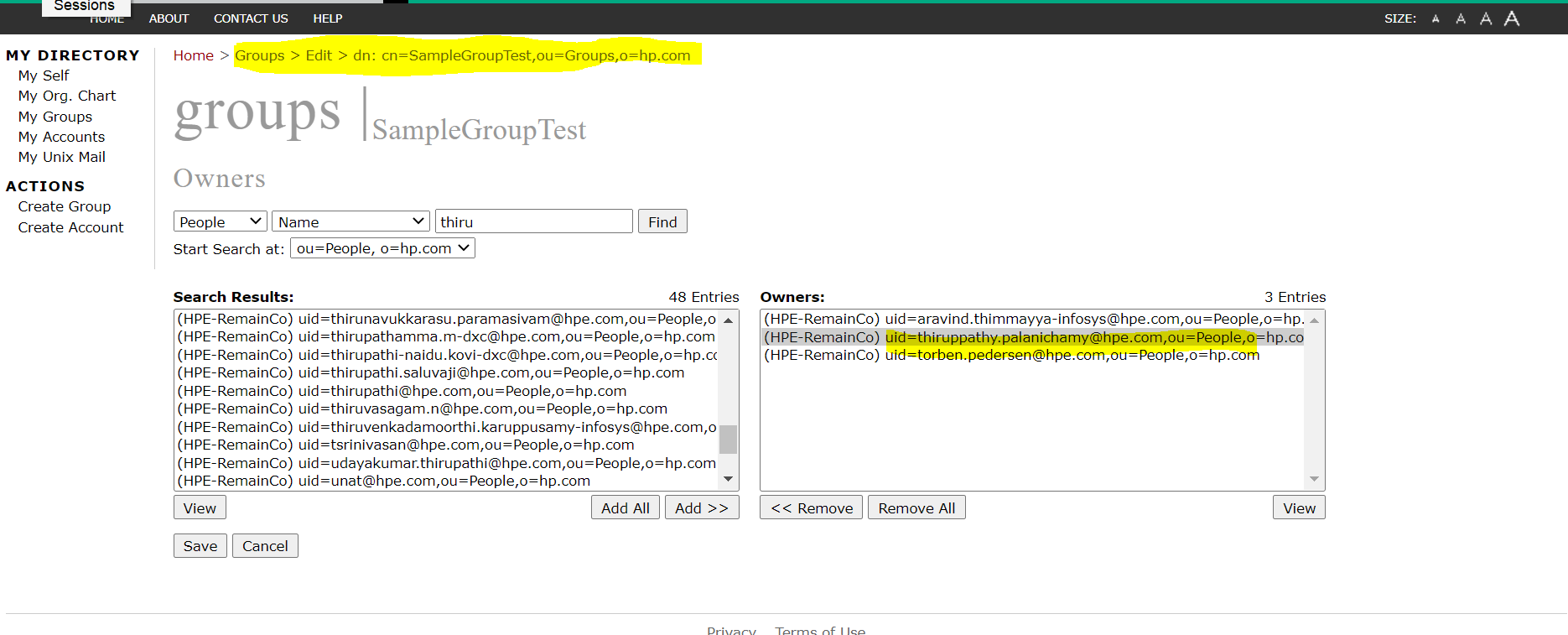
<https://directoryworks.hpecorp.net/protected/people/view/groups/owner/>



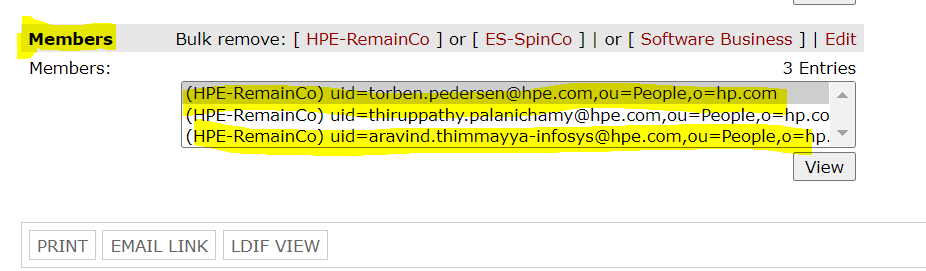


1. Edit **owners** tab of created group and add owners as Thiru & Torben





Edit **members** tab of created group and add owners as Thiru & Torben



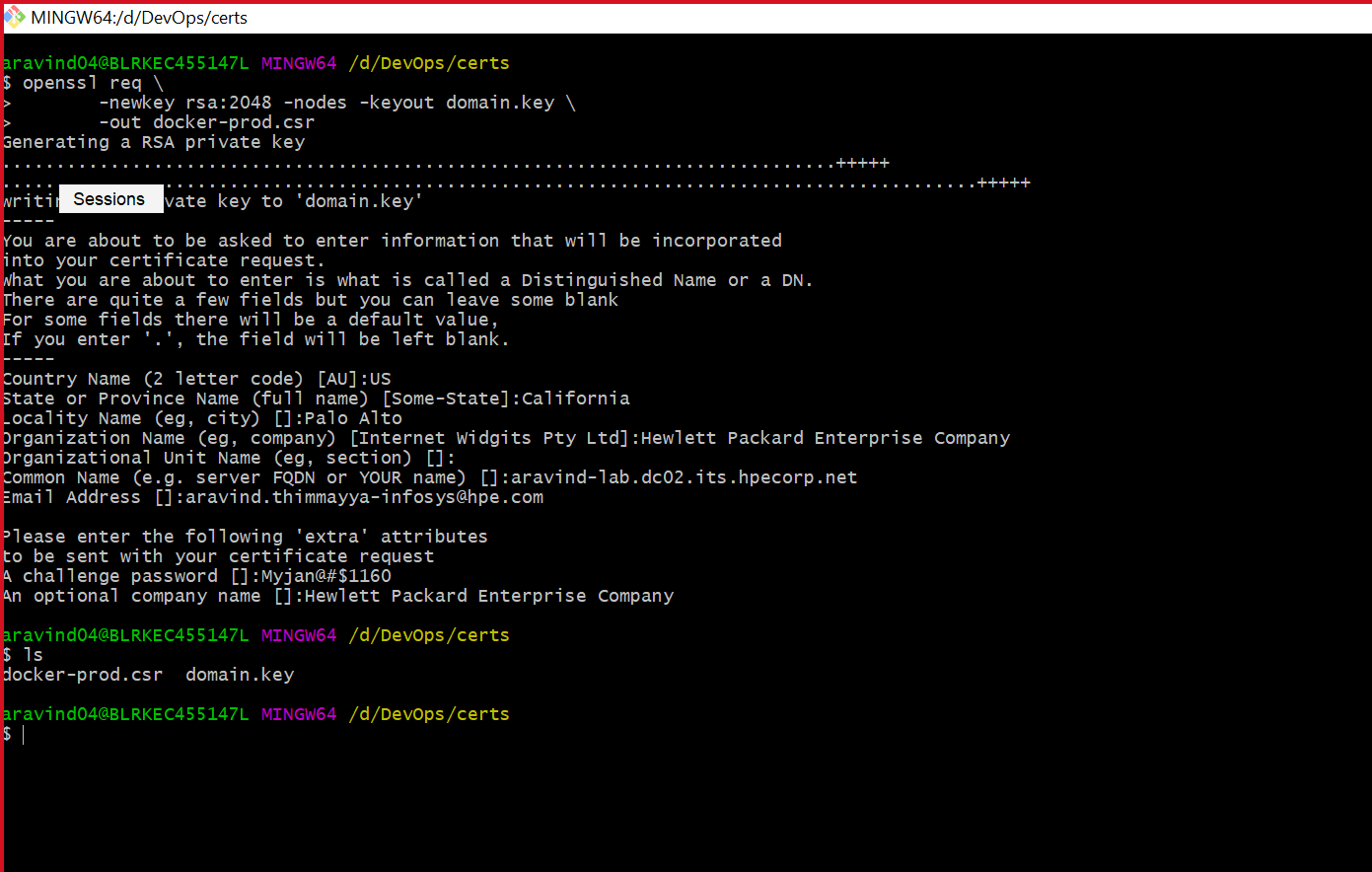
|  |  |  |  |
| --- | --- | --- | --- |
| |  |  | | --- | --- | | Properties | [Edit](https://directoryworks.hpecorp.net/protected/groups/edit/properties?dn=cn%3DSampleGroupTest%2Cou%3DGroups%2Co%3Dhp.com) | | |
| Common Name: | **SampleGroupTest** |
| Description: | This is sample group test for sample deployment testing. |
| Self Subscribe: | HP Employees |
| Mail: | [SampleGroupTest@groups.int.hpe.com](mailto:SampleGroupTest@groups.int.hpe.com) |
| Batch Import: | [Batch Import Tool](https://directory.hpecorp.net/group_import/groupimport.cgi) |
|  |  |

1. Next, we need SSL cert by following with below mentioned document. To create SSL cert, use below command and follow the document (**SSL Creation for APP teams.doc**) for more info.

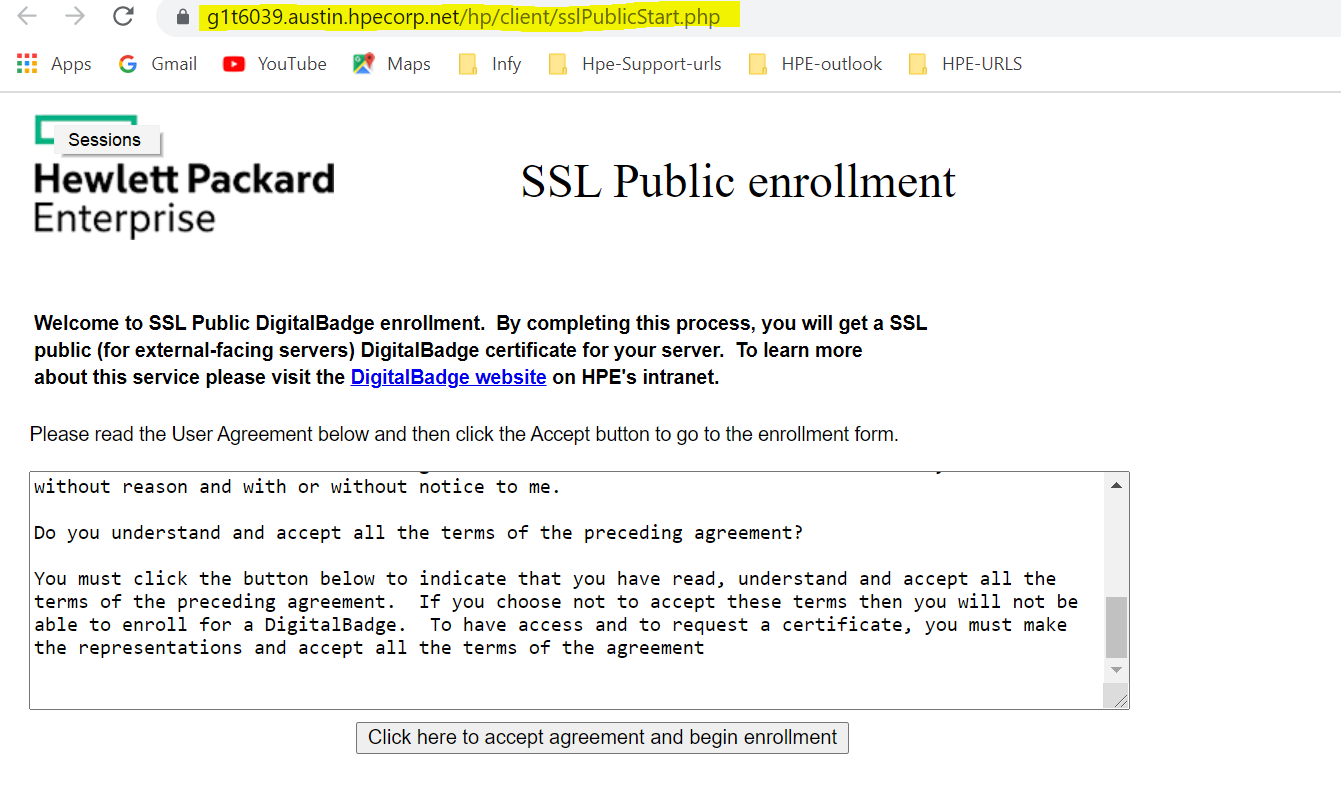
openssl req \

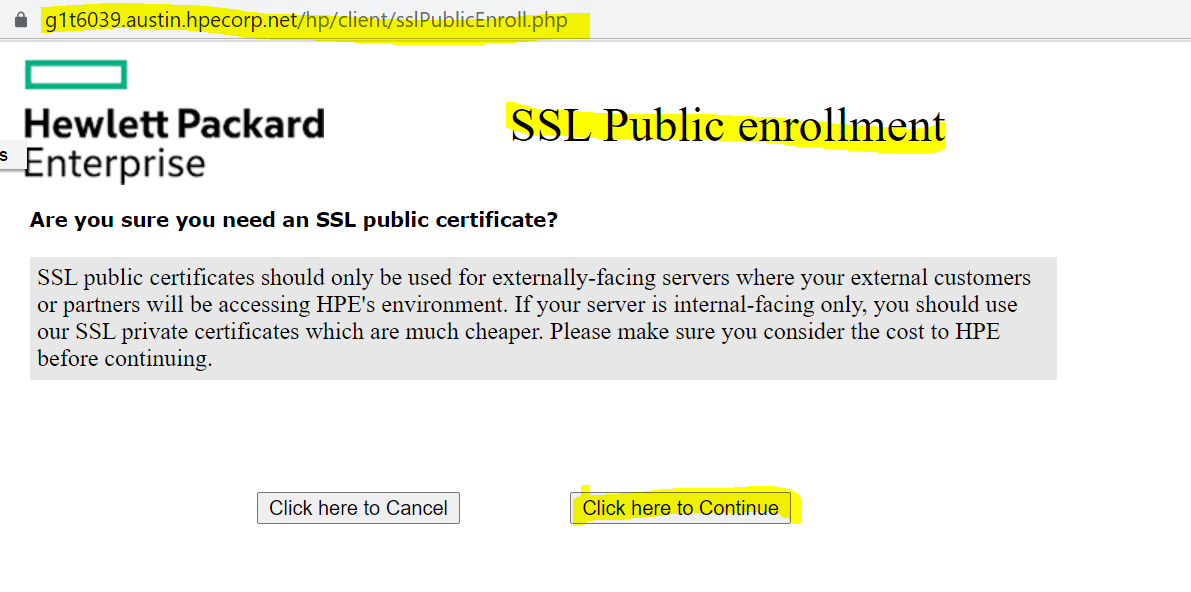
-newkey rsa:2048 -nodes -keyout domain.key \

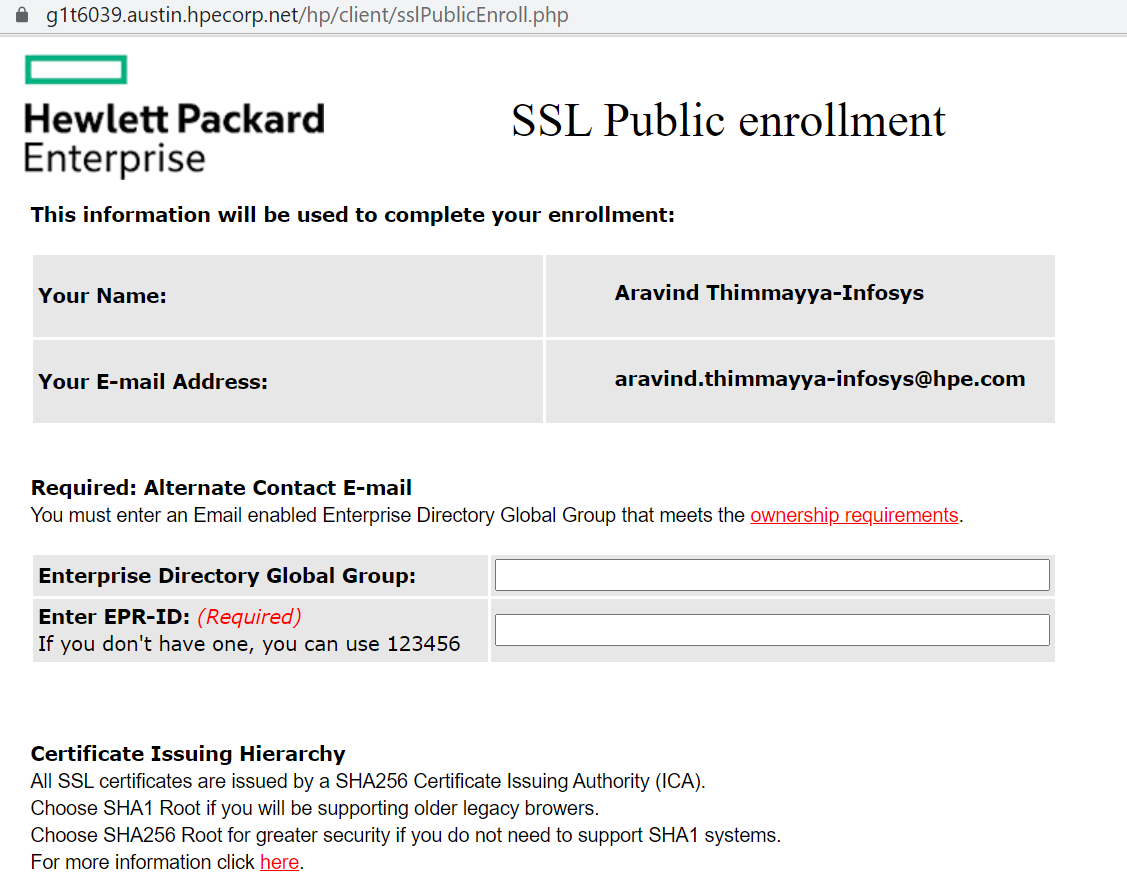
-out docker-prod.csr

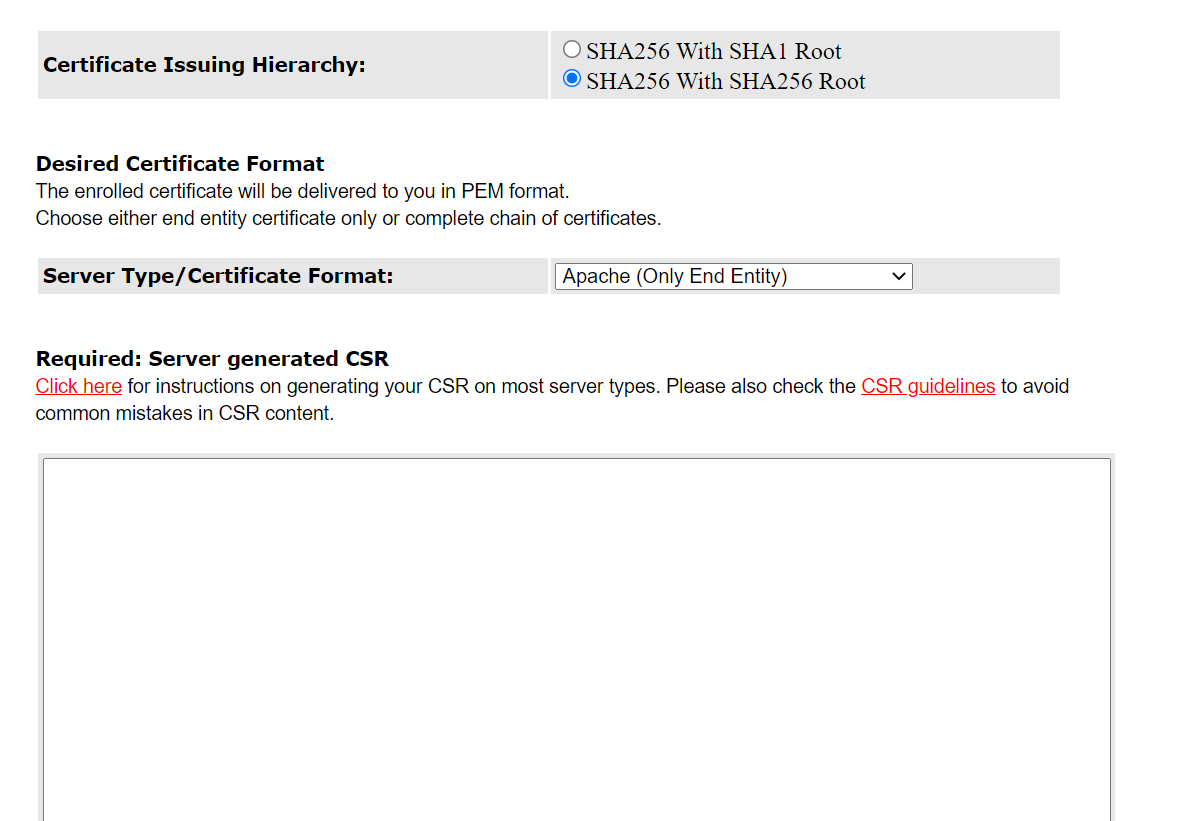
While running above command from gitbash, it will ask few details for those values please refer this document: [**https://ent302.sharepoint.hpe.com/teams/credentials/\_layouts/15/WopiFrame2.aspx?sourcedoc=/teams/credentials/Shared%20Documents/CSR%20Guidelines.docx&action=default**](https://ent302.sharepoint.hpe.com/teams/credentials/_layouts/15/WopiFrame2.aspx?sourcedoc=/teams/credentials/Shared%20Documents/CSR%20Guidelines.docx&action=default)

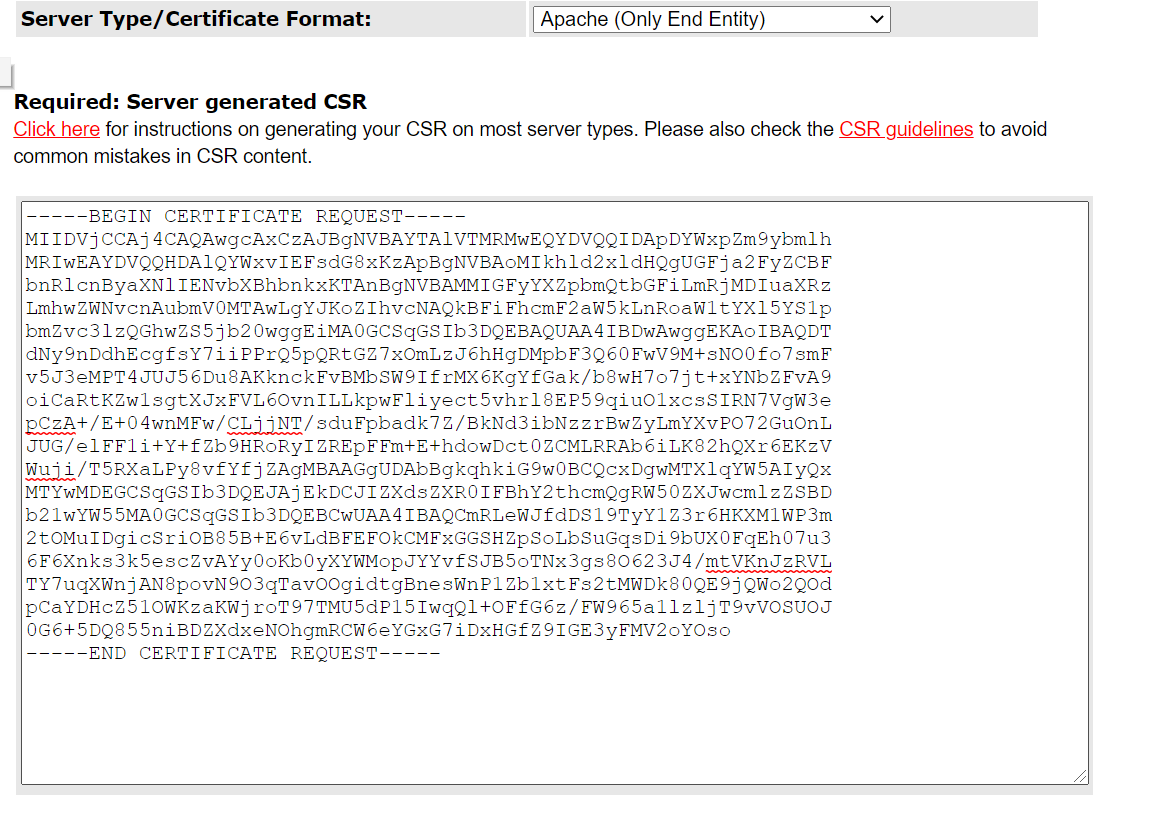
1. Once you are ready with cert and group, please open this link (<https://g1t6039.austin.hpecorp.net/hp/client/sslPublicEnroll.php>), fill the group details and upload the generated CSR cert.



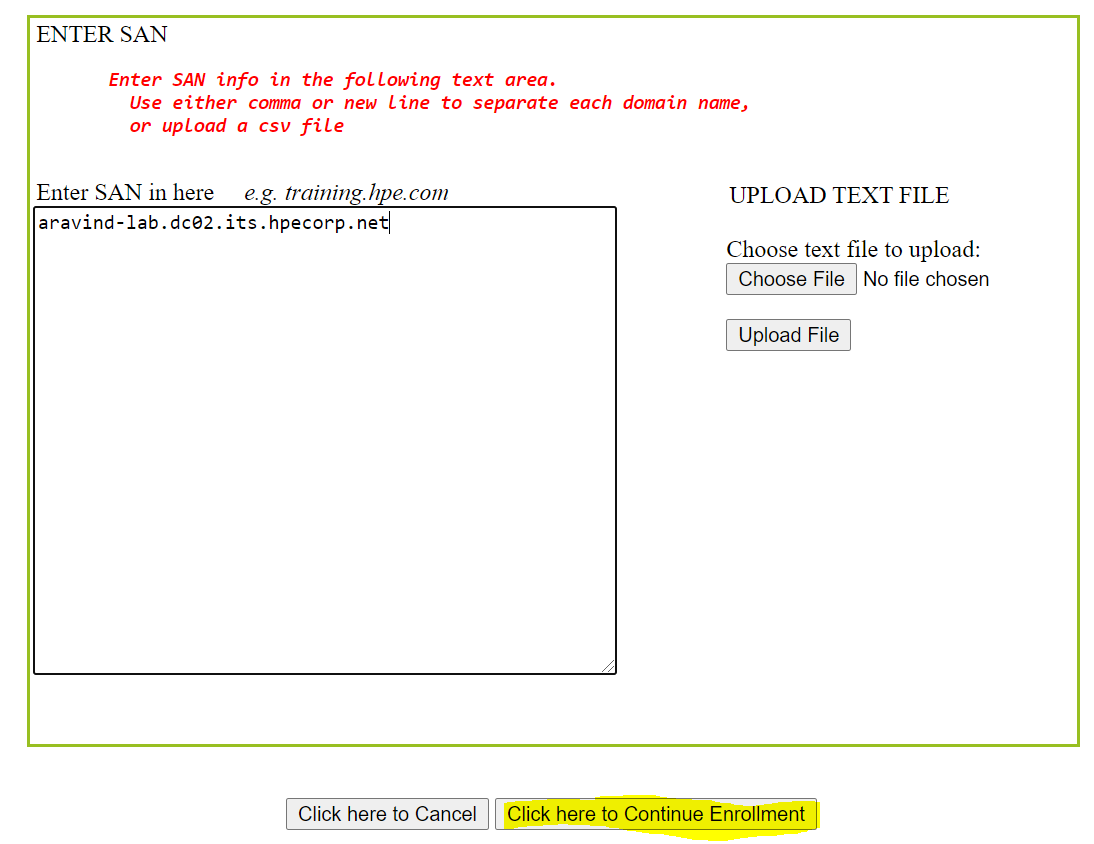


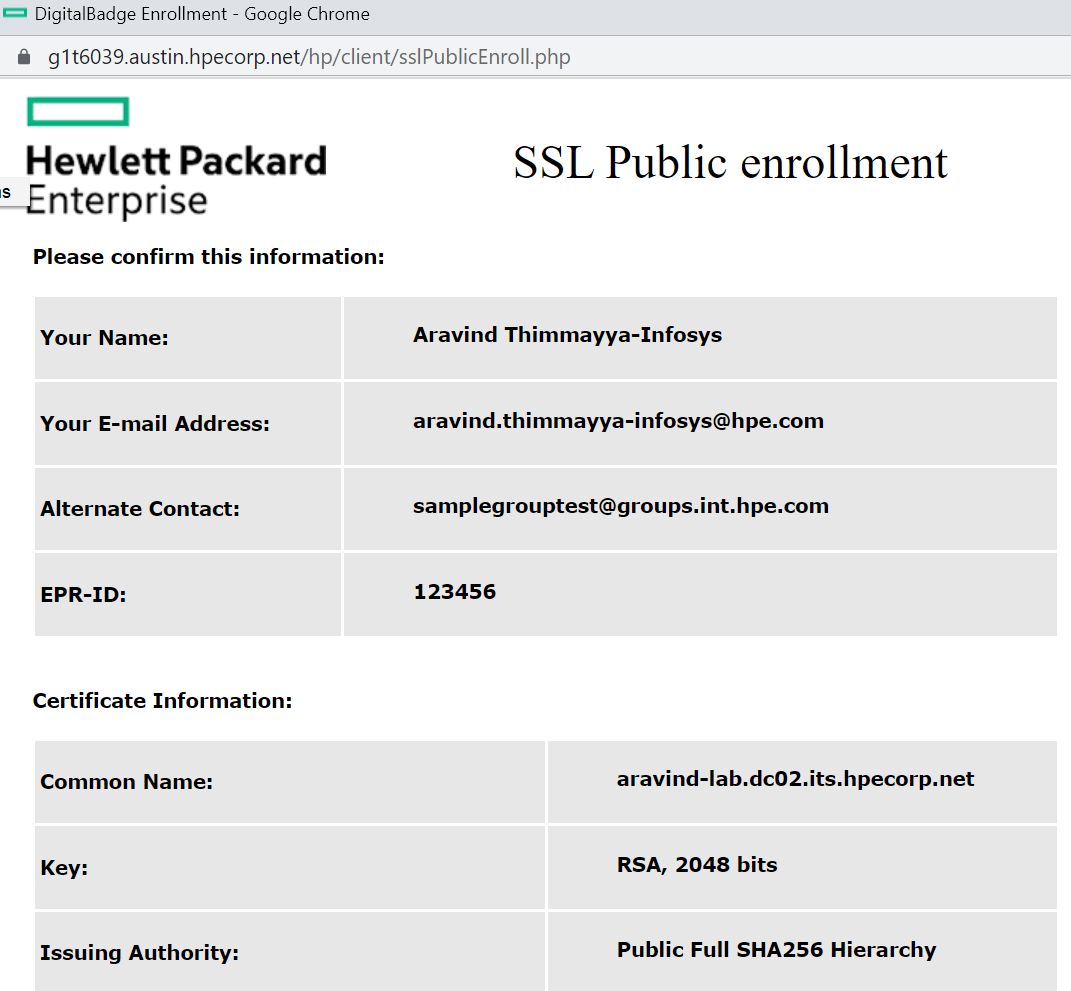


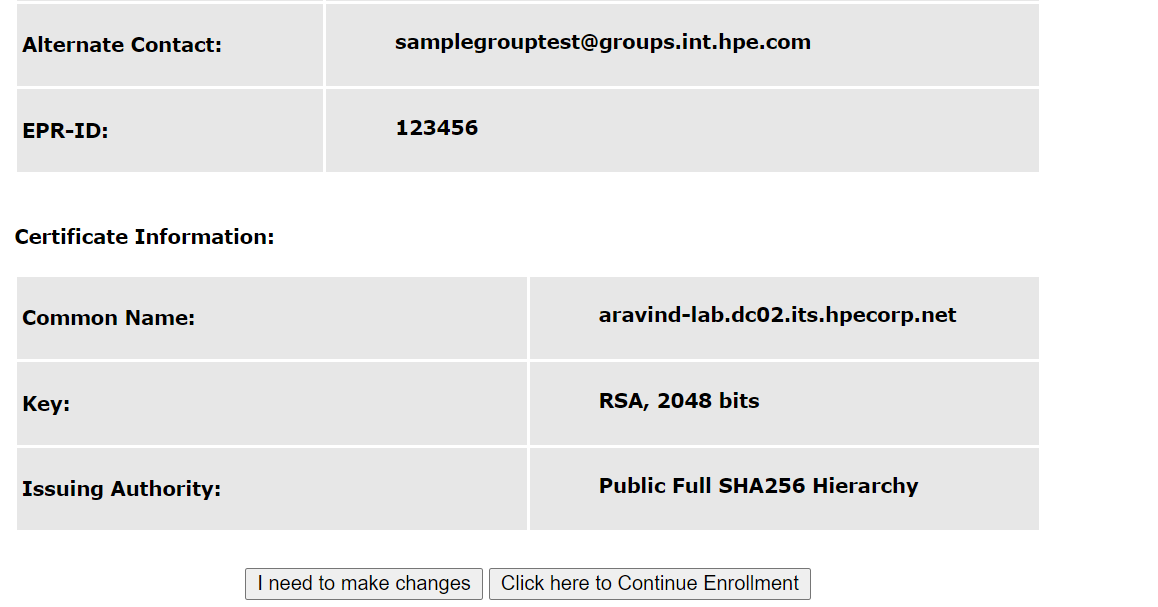


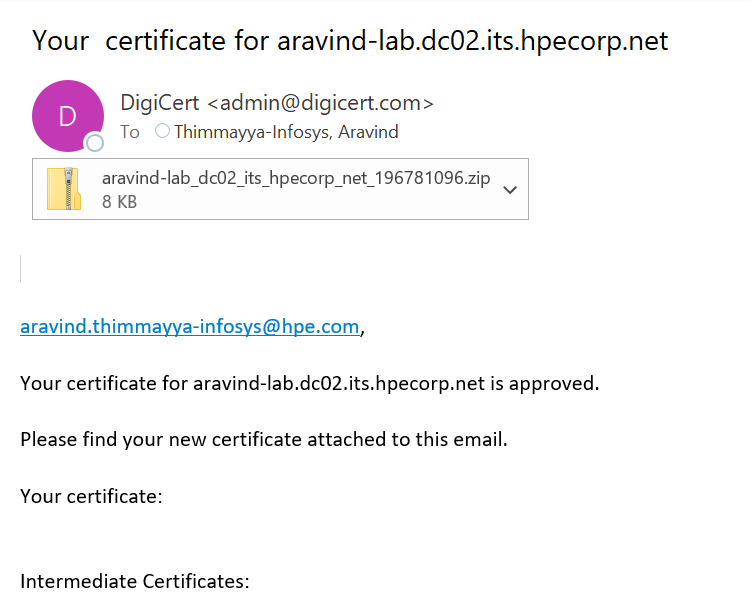


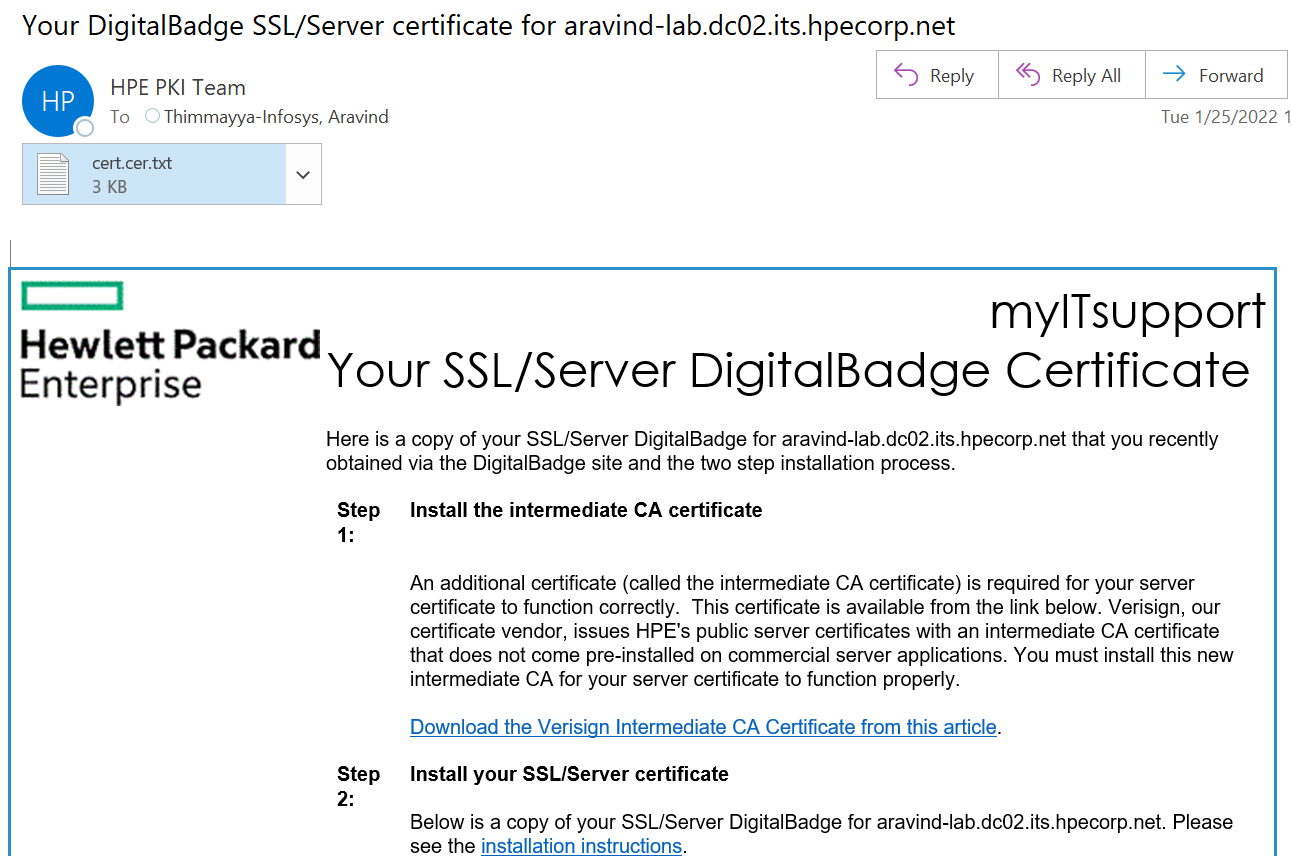
* Place your DNS in SAN box and click on click here to continue enrollment.





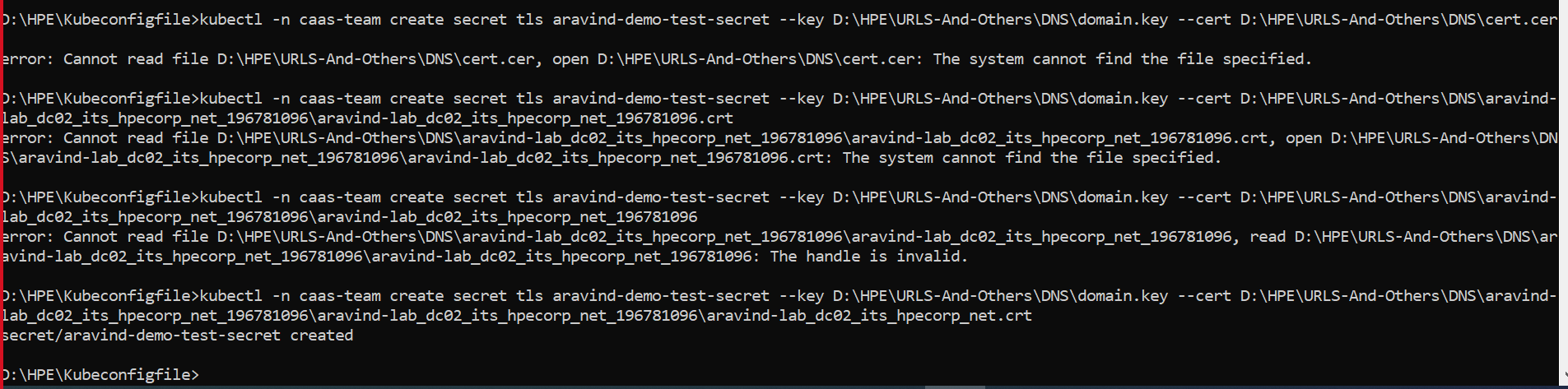


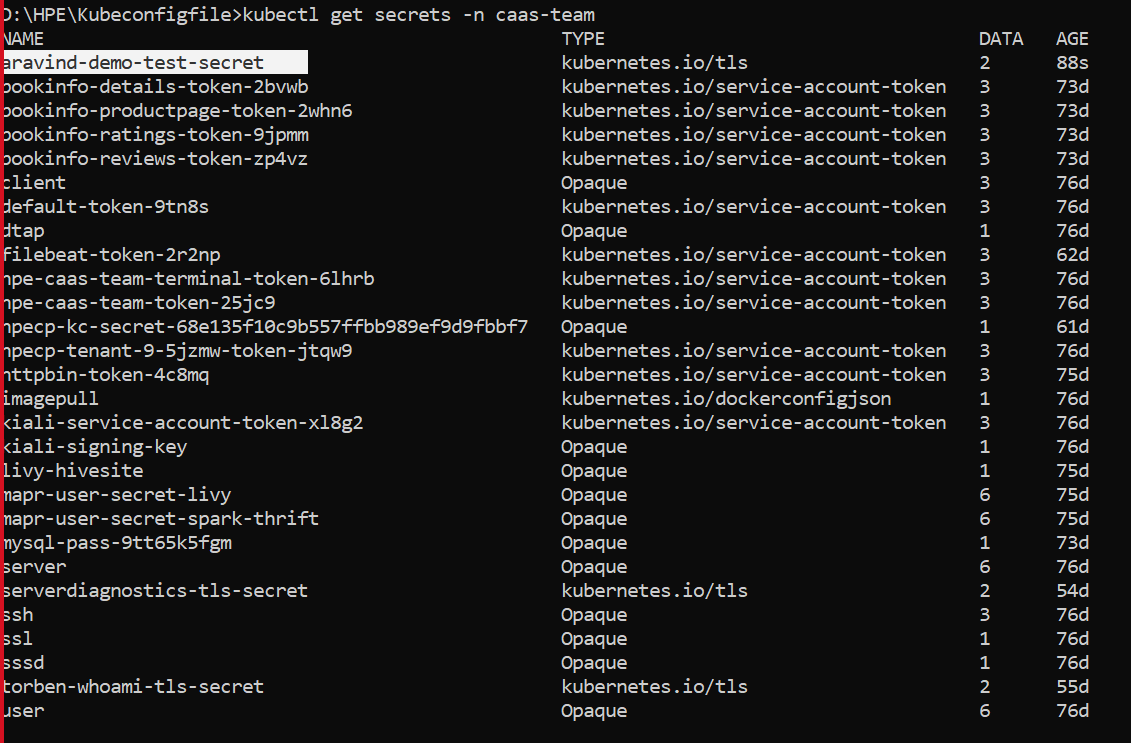




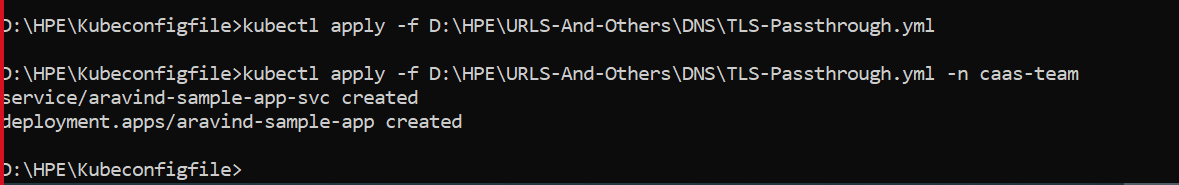
* We need to create secrete in our namespace using below command, to access our applications.
* Demo-app.key and demo-app.cert will be generated at the time of creating csr cert.

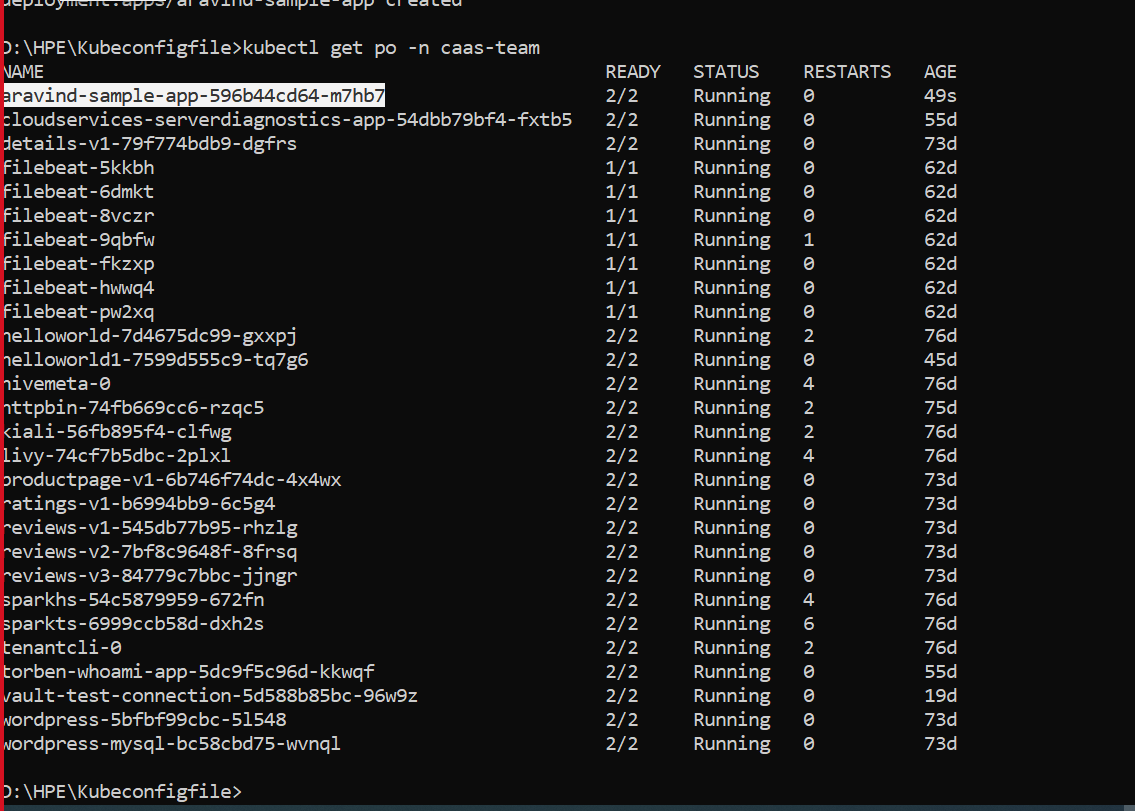
**$ kubectl -n <namespace> create secret tls demo-app-certs --key /path/to/demo-app.key --cert /path/to/demo-app.crt**

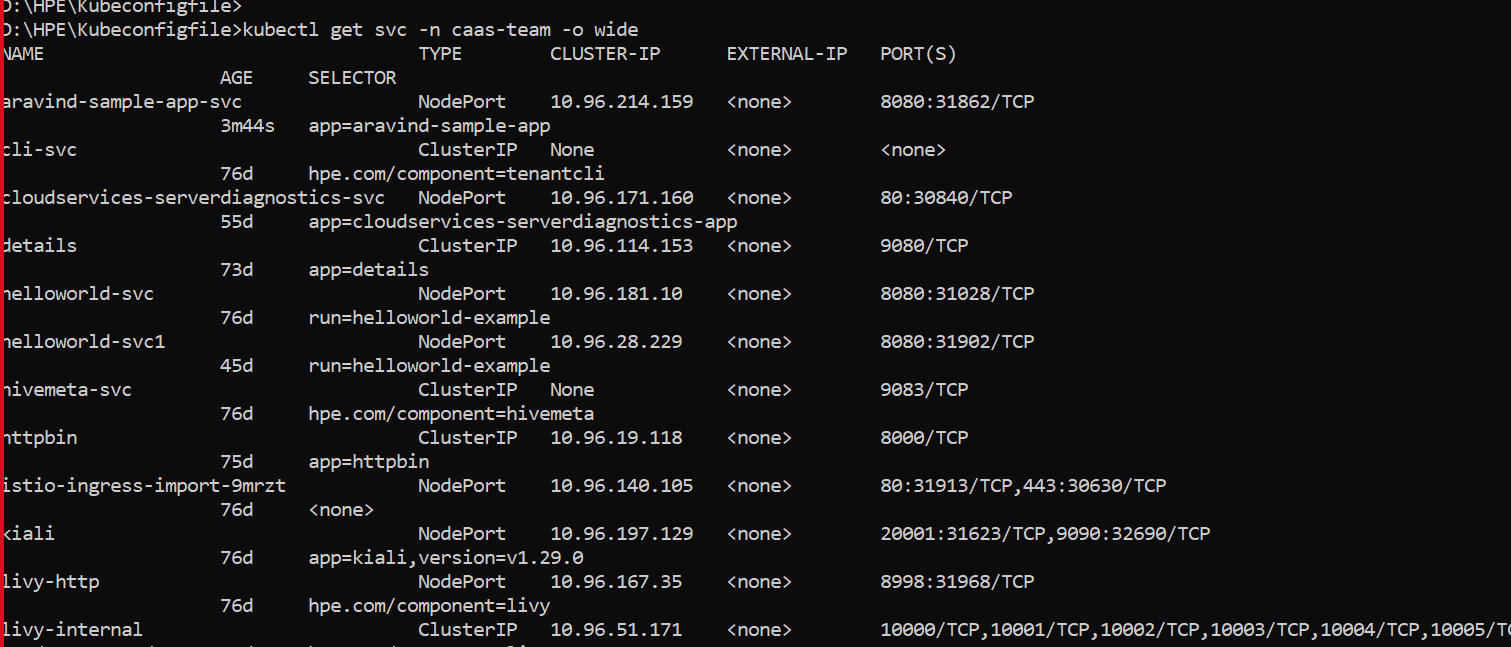




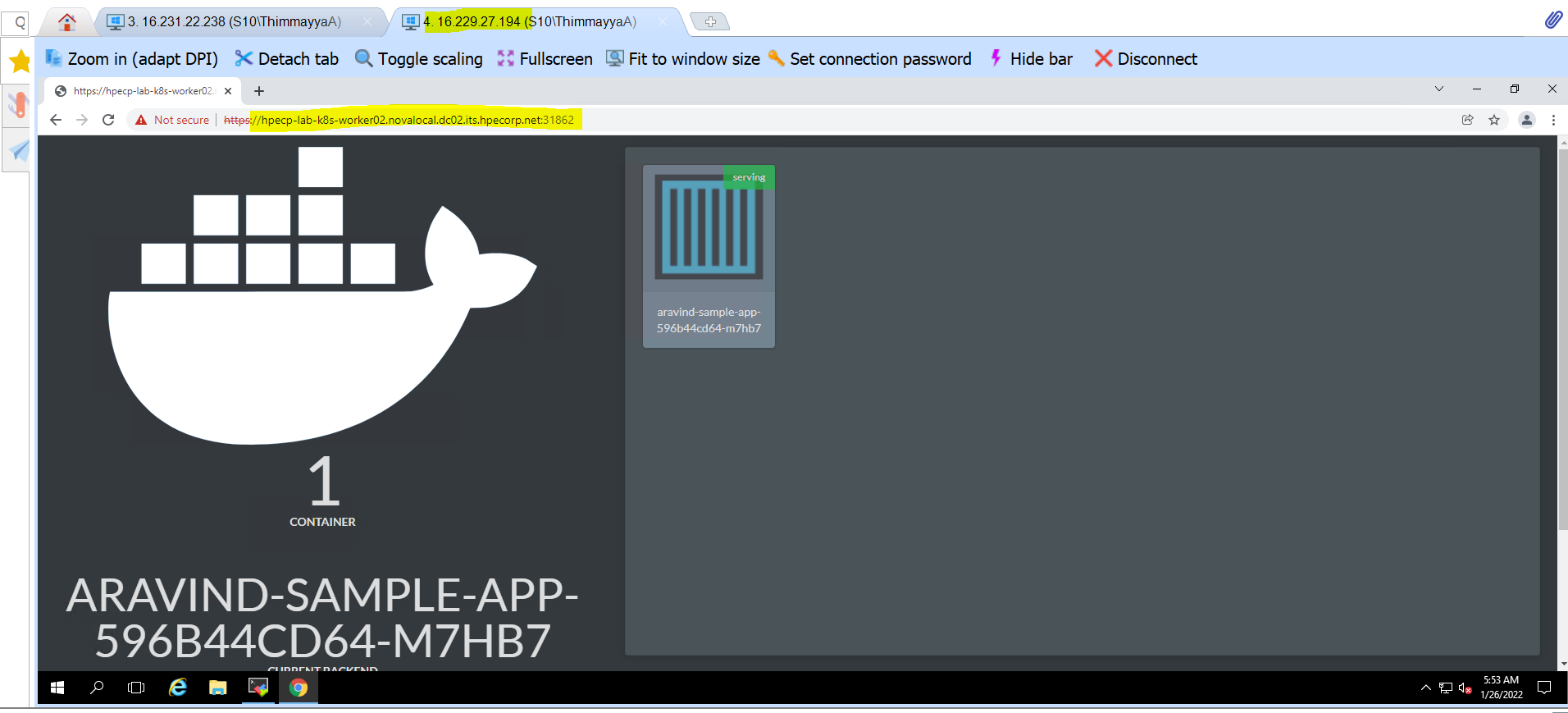
* Once secret is created deploy your yaml file which having deployment, Service, Gateway, Virtual service, destination rules and secretes etc.



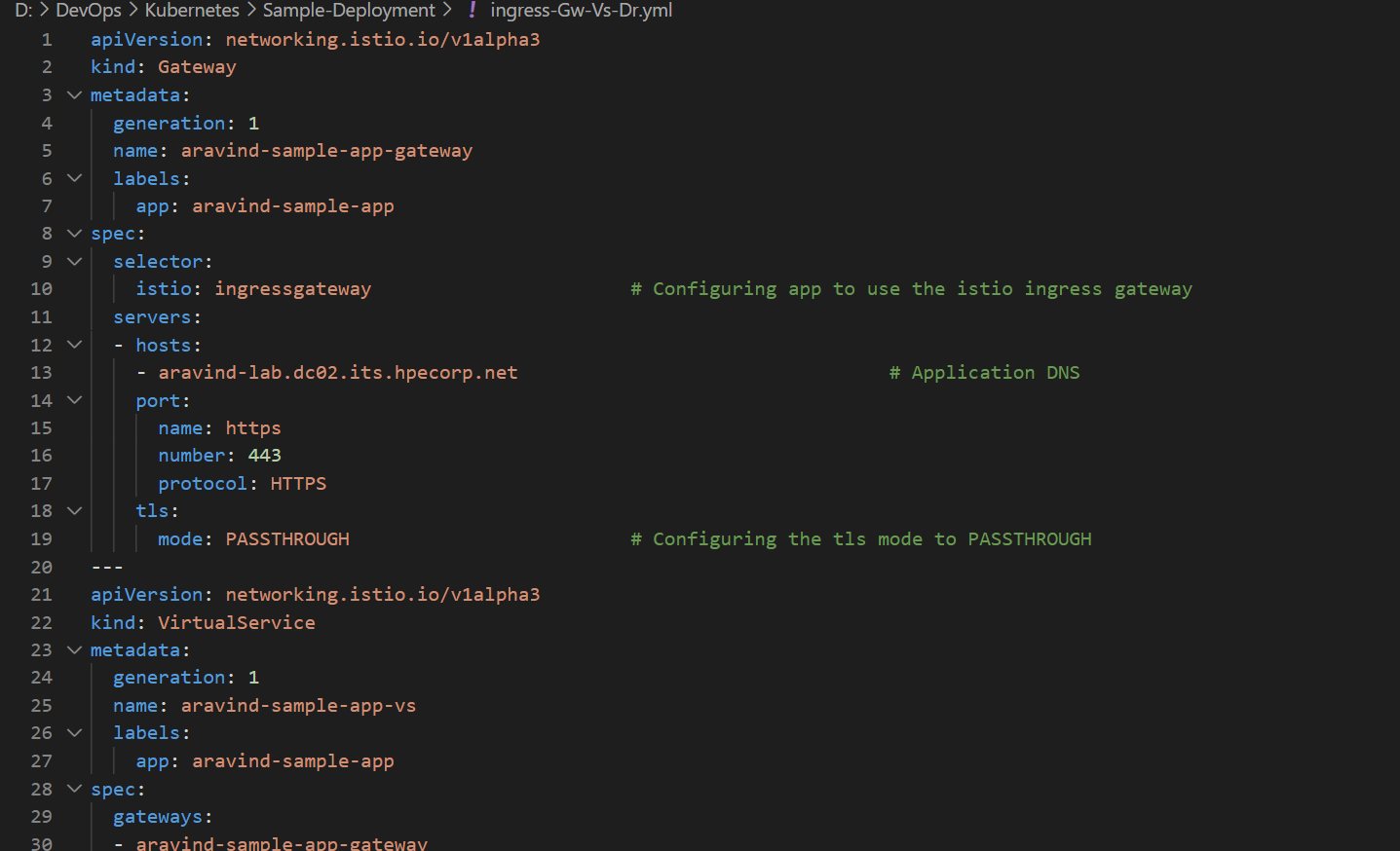




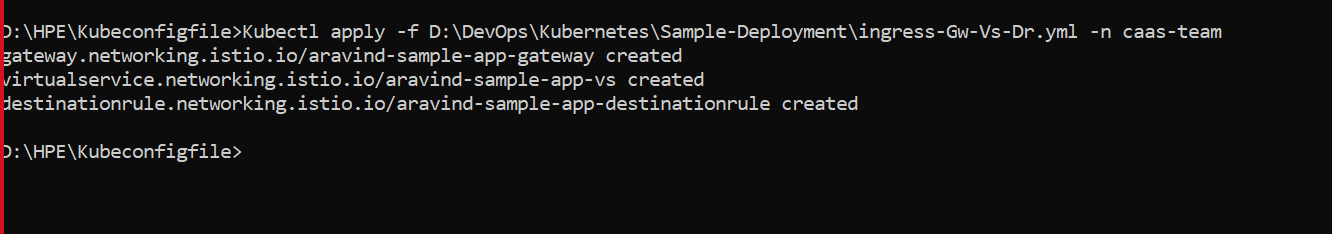
* App Running in Colo2 jump host and refer screenshot below**.**



1. Configure the ingress Gateway, Virtual Service and Destination Rules yaml code.



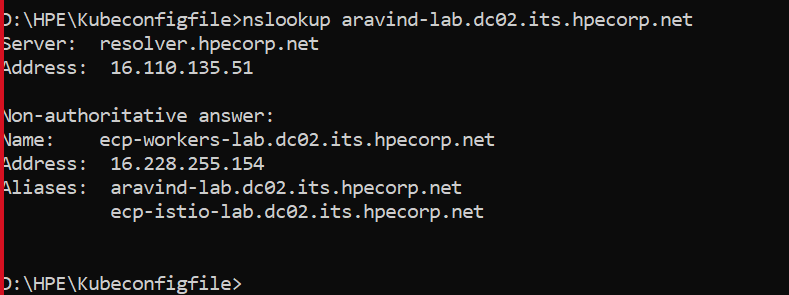
Kubectl apply -f D:\DevOps\Kubernetes\Sample-Deployment\ ingress-Gw-Vs-Dr.yml





* Use below command to validate your dns is resolving or not.

**D:\HPE\Kubeconfigfile>nslookup aravind-lab.dc02.its.hpecorp.net**



* Once it gets validated your dns, please try to access your application using dns from outside datacenter.

**URL:** https://aravind-lab.dc02.its.hpecorp.net/

