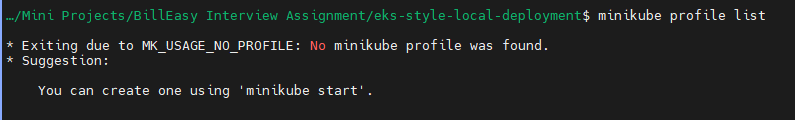
0. Installation and Prerequisites:

Check for existing minikube cluser:

* minikube profile list

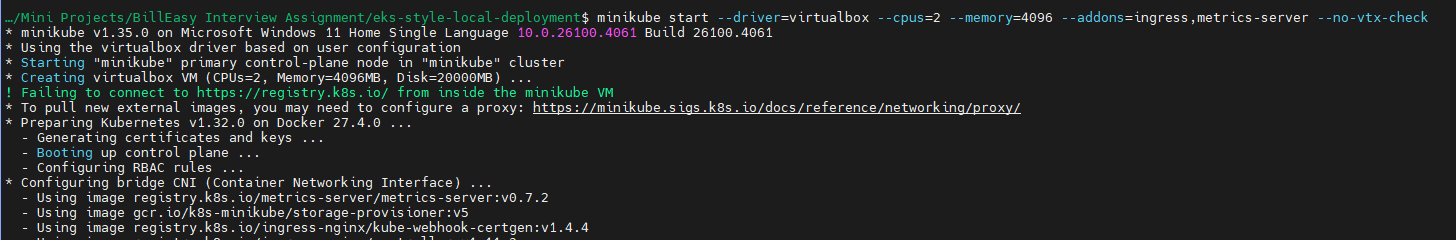


If exists, delete it

* minikube delete

Start your minikube cluster:

* minikube start --driver=virtualbox --cpus=2 --memory=4096 --addons=ingress,metrics-server --no-vtx-check



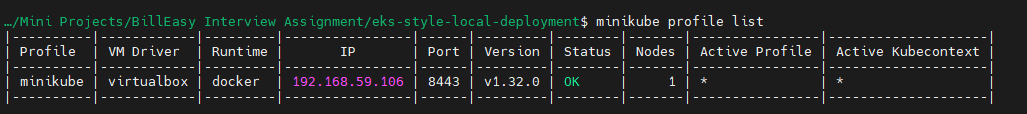
Check the status of minikube cluster:

* minikube status



List the minikube profiles:

* minikube profile list

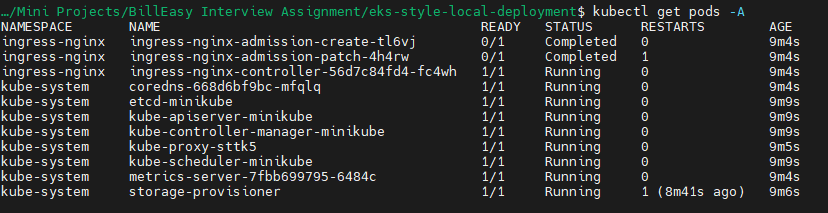


Ensure that your system is ready:

* kubectl get nodes

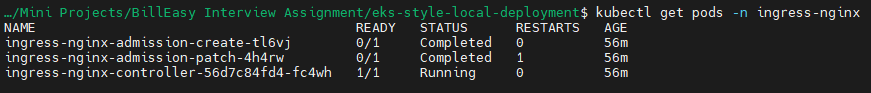


* kubectl get pods -A



Verify your ingress:

* kubectl get pods -n ingress-nginx

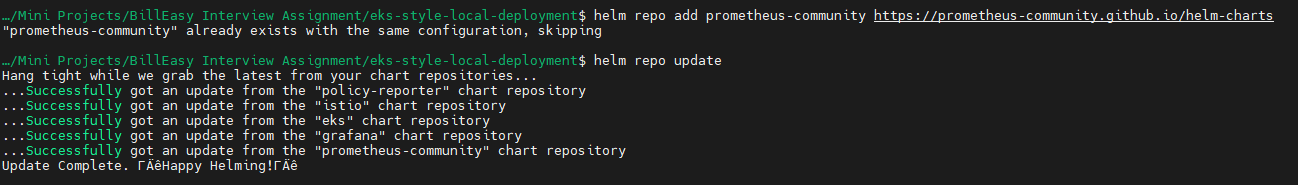


Now we setup Prom and Grafana in a new namespace monitoring by referring: <https://medium.com/@joudwawad/comprehensive-beginners-guide-to-kube-prometheus-in-kubernetes-monitoring-alerts-integration-4ade4fa8fa8c> :

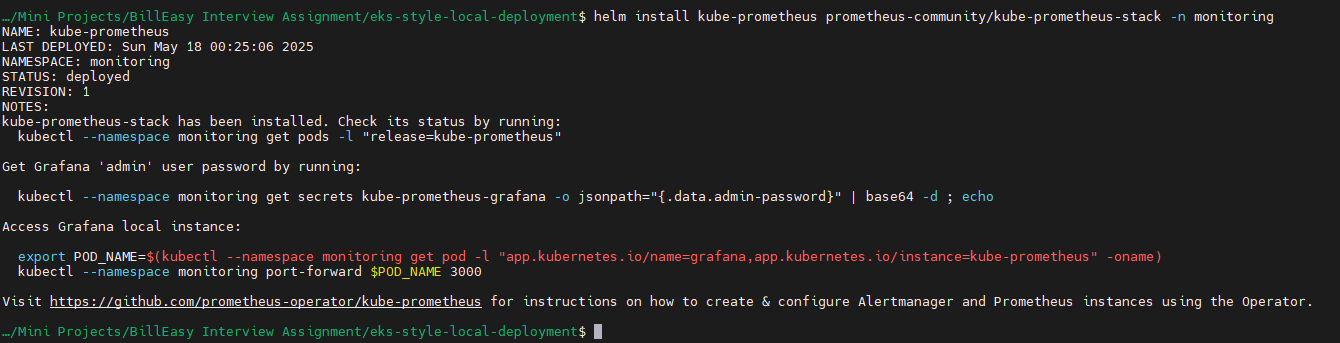
* kubectl create namespace monitoring



* helm repo add prometheus-community <https://prometheus-community.github.io/helm-charts>
* helm repo update

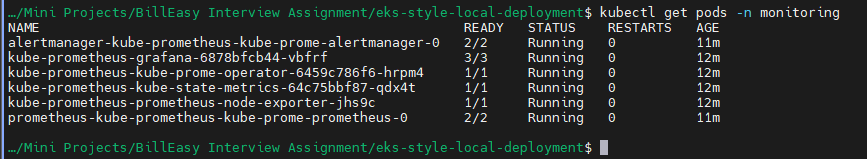


* helm install kube-prometheus prometheus-community/kube-prometheus-stack -n monitoring



Wait for the pods to be in running stage:

* kubectl get pods -n monitoring

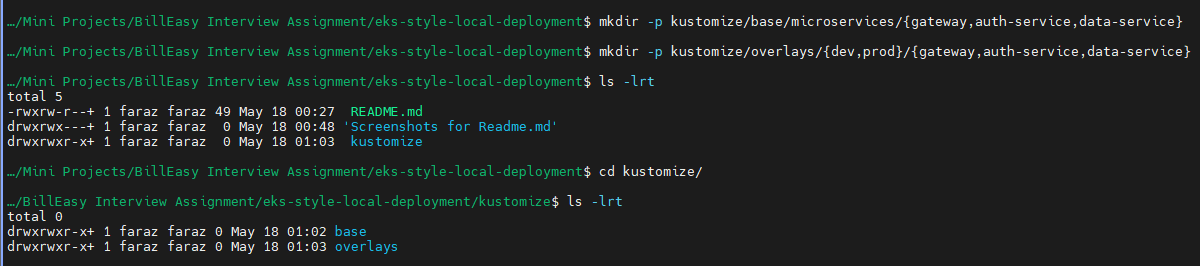


1. Microservice Stack

Refer: <https://subbaramireddyk.medium.com/kustomize-kubernetes-native-configuration-management-f51630d29ac0>

Creating kustomize project structure:

* mkdir -p kustomize/base/microservices/{gateway,auth-service,data-service}
* mkdir -p kustomize/overlays/{dev,prod}/{gateway,auth-service,data-service}

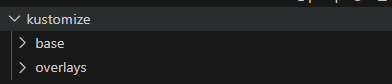


Create kustomization.yaml files:

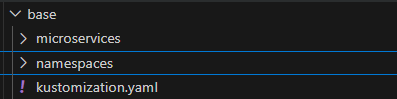


Create rest of the project structure

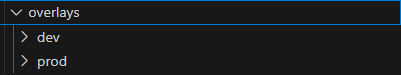
Final project structure:



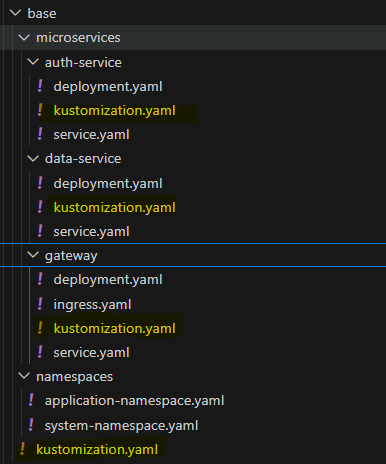
Base:



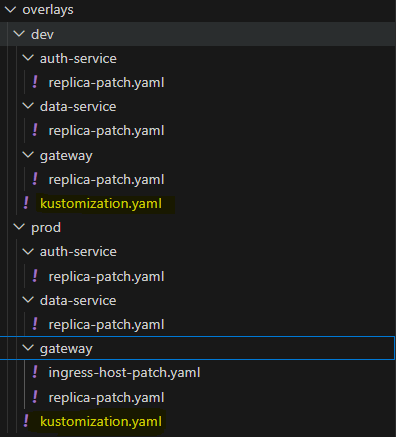
Overlays:



Base project folders:



Overlays project folders:

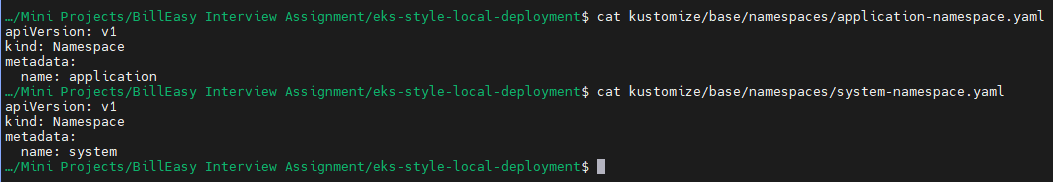


Create namespaces application and system:

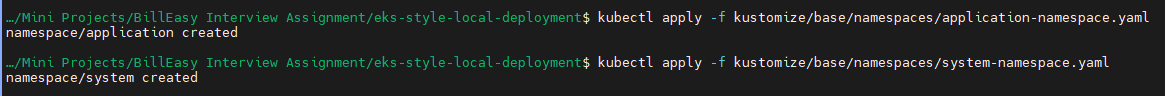
Check the existing namespaces:



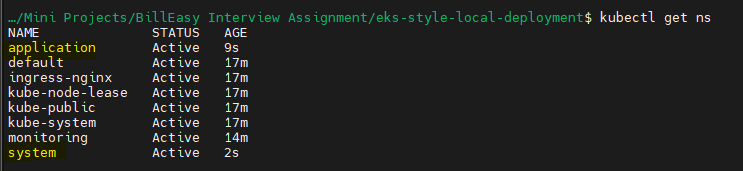
Create yaml to create namespaces:



Apply to create namespace:



Verify your namespace creation:



\*\* Assuming that gateway will belong to the application namespace since it needs to be publicly accessible via Ingress, while auth-service and data-service should reside in the system namespace to keep them internal-only.

Now we will deploy the following services:

|  |  |  |
| --- | --- | --- |
| **Service** | **Purpose** | **Docker Image** |
| gateway | API gateway, exposed via ingress | nginxdemos/hello |
| auth-service | Auth logic, logs headers | kennethreitz/httpbin |
| data-service | Mock business logic | hashicorp/http-echo |

Create yaml manifests of gateway, auth-service and data-service with the mentioned requirements

and apply it

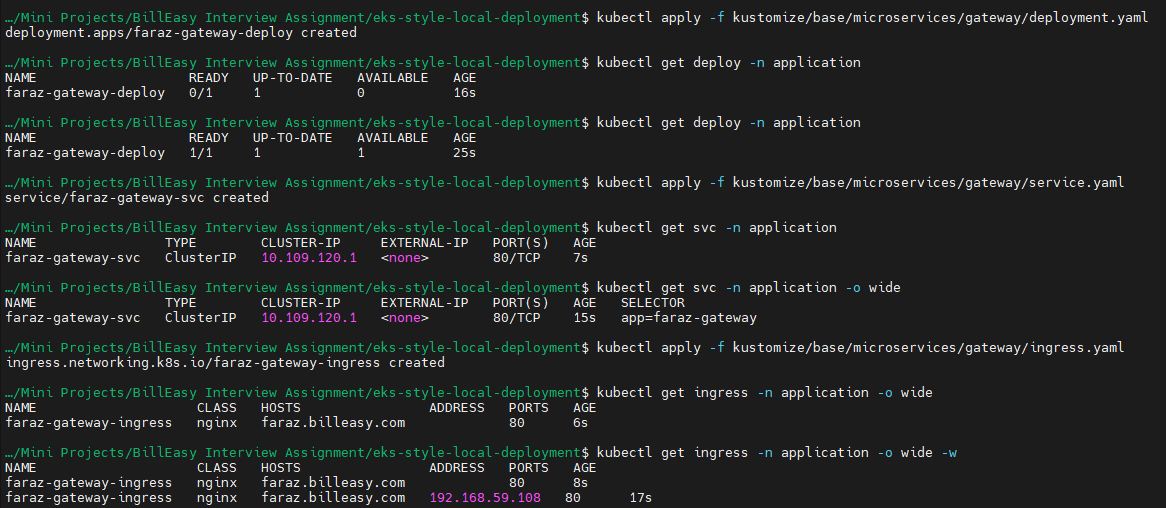
Requirements:

* Containerized deployment using Helm or Kustomize
* Liveness/readiness probes
* Proper resource requests/limits
* gateway should be publicly accessible via ingress
* auth-service and data-service should be internal-only
* Use separate namespaces for system and application

Applying and manually testing if every manifest created is working as expected:

Testing Gateway manifests:

* + kubectl apply -f kustomize/base/microservices/gateway/deployment.yaml
  + kubectl get deploy -n application
  + kubectl apply -f kustomize/base/microservices/gateway/service.yaml
  + kubectl get svc -n application -o wide
  + kubectl apply -f kustomize/base/microservices/gateway/ingress.yaml
  + kubectl get ingress -n application -o wide



Also add the ingress ip to /etc/hosts

* sudo vim /etc/hosts

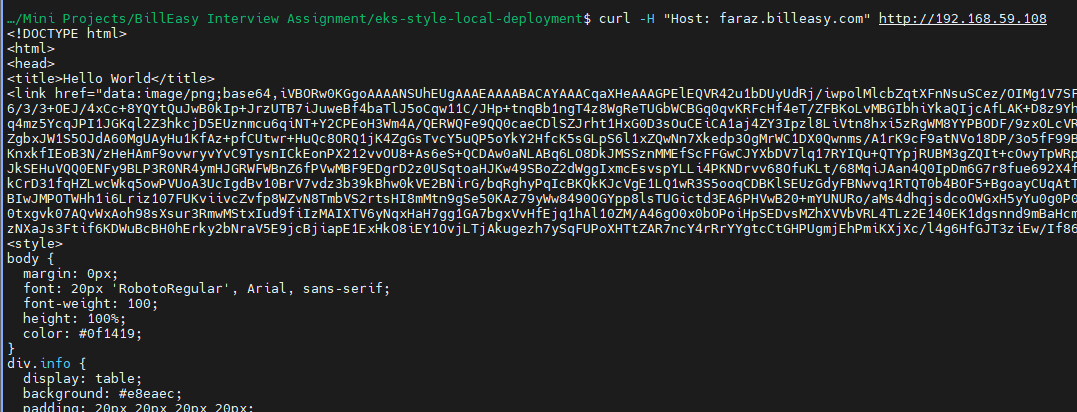


You must now be able to access the application from your browser:



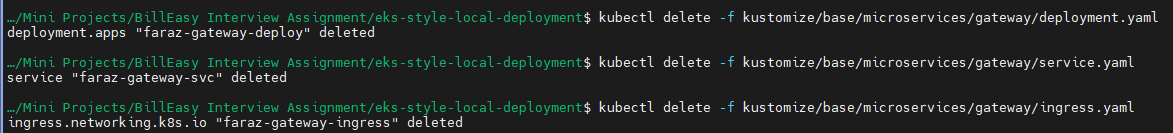
You can also test it using curl:

* curl -H "Host: faraz.billeasy.com" http://192.168.59.108



After testing, delete the created services:

* kubectl delete -f kustomize/base/microservices/gateway/deployment.yaml
* kubectl delete -f kustomize/base/microservices/gateway/service.yaml
* kubectl delete -f kustomize/base/microservices/gateway/ingress.yaml

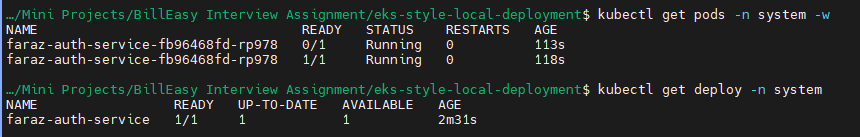


Testing auth-service:

Apply the deployment:

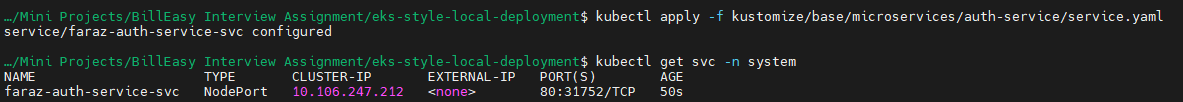
* kubectl apply -f kustomize/base/microservices/auth-service/deployment.yaml
* kubectl get pods -n system
* kubectl get deploy -n system





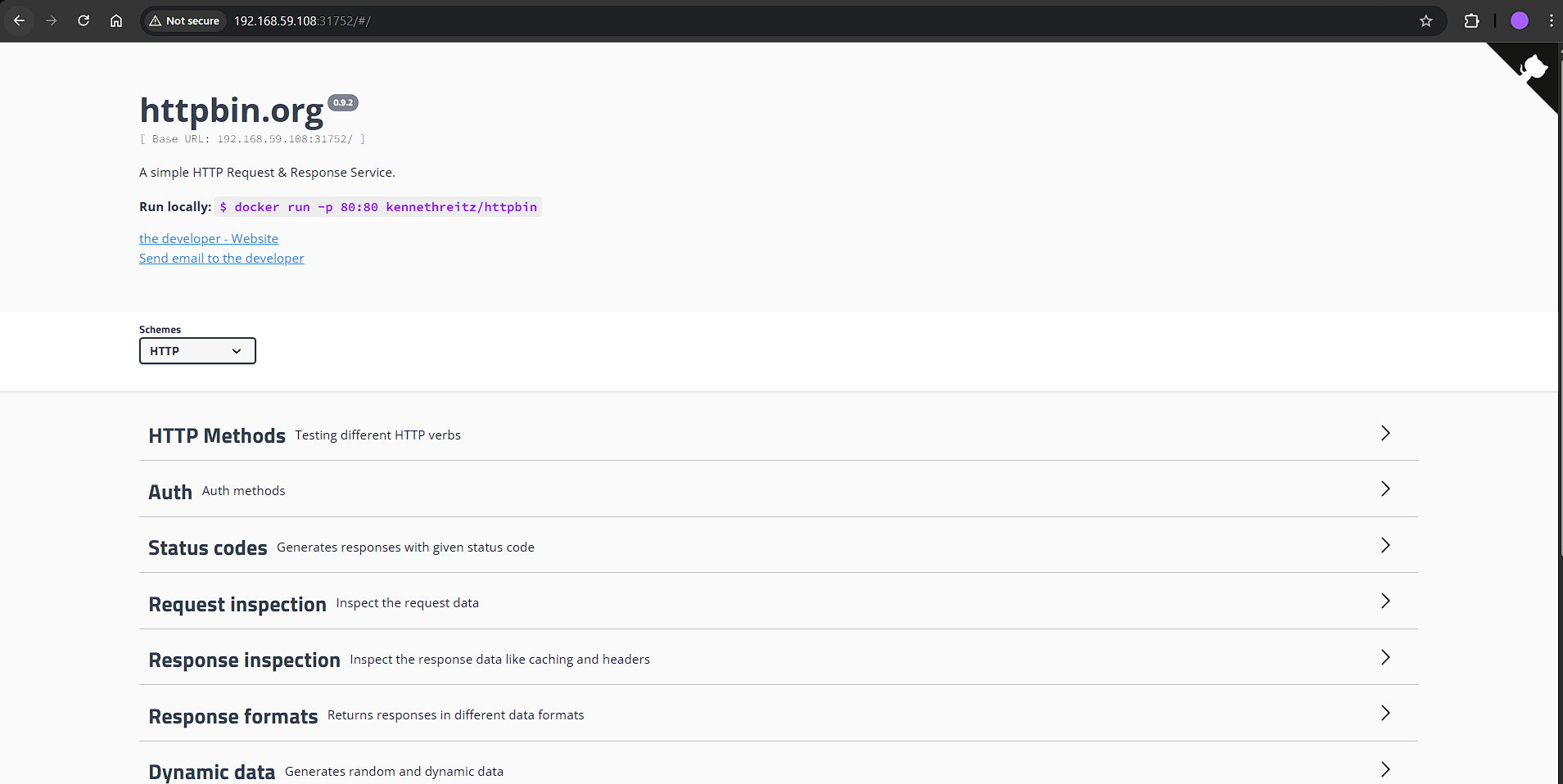
For testing we will apply the service in NodePort mode:

* kubectl apply -f kustomize/base/microservices/auth-service/service.yaml
* kubectl get svc -n system



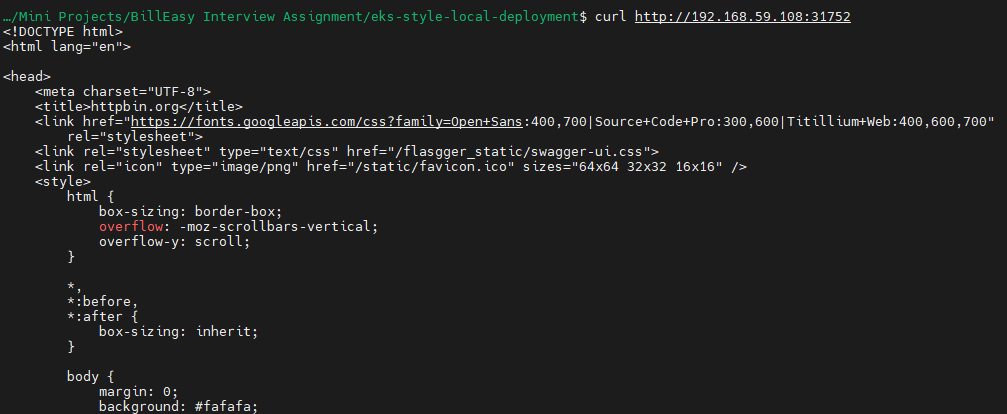
By getting the minikube IP and the Node port:

Verify if it is working:



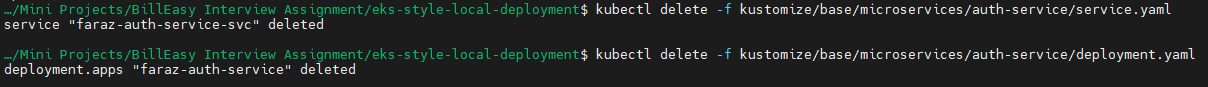
You can also use curl:

* curl http://192.168.59.108:31752



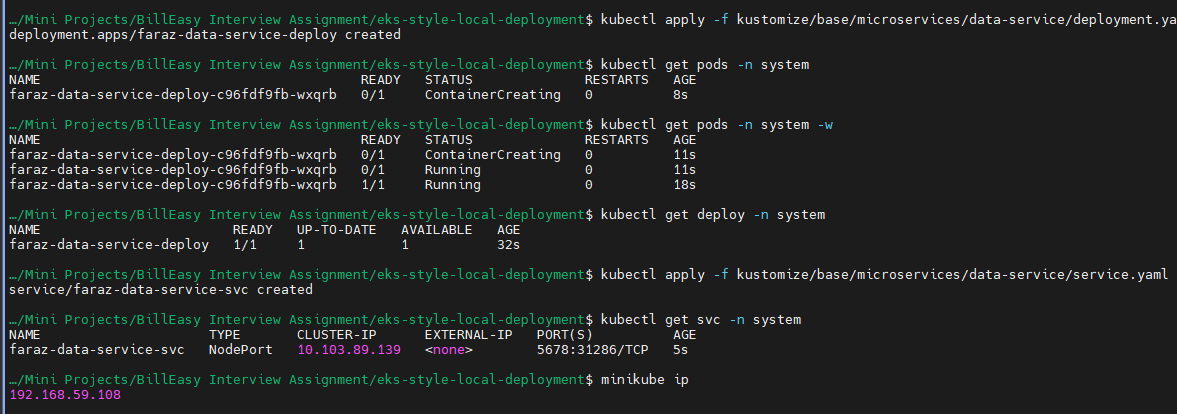
After testing deleting the deployment and service:

* kubectl delete -f kustomize/base/microservices/auth-service/service.yaml
* kubectl delete -f kustomize/base/microservices/auth-service/deployment.yaml



Similarly test data-service:

* kubectl apply -f kustomize/base/microservices/data-service/deployment.yaml
* kubectl get pods -n system
* kubectl get deploy -n system
* kubectl apply -f kustomize/base/microservices/data-service/service.yaml
* kubectl get svc -n system

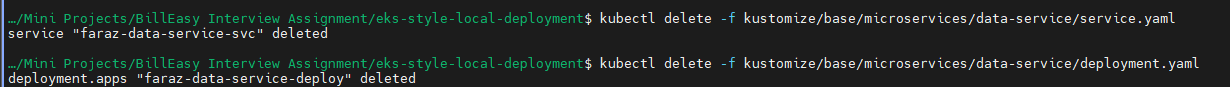


Open it in your browser:



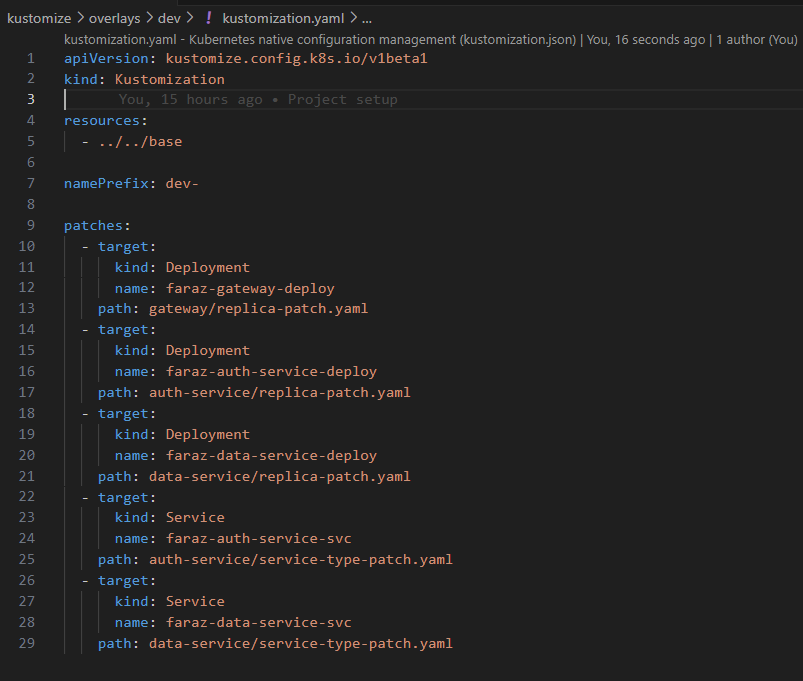
Delete the service and deployment:

* kubectl delete -f kustomize/base/microservices/data-service/service.yaml
* kubectl delete -f kustomize/base/microservices/data-service/deployment.yaml



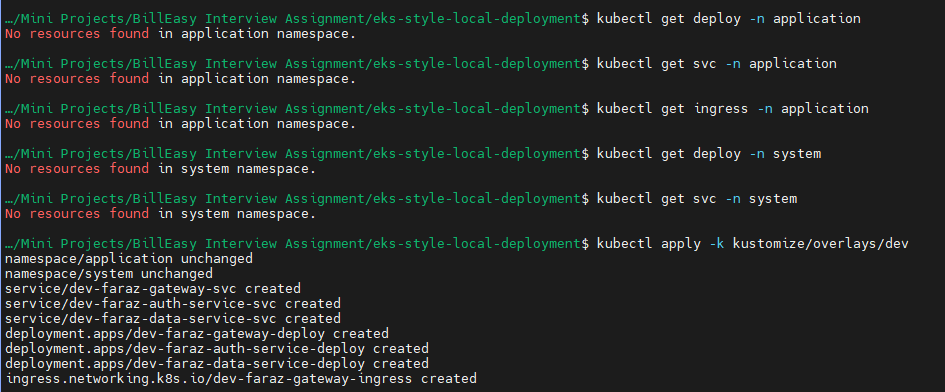
Now we will use kustomize to test:

Create your kustomization.yaml and create your patches:

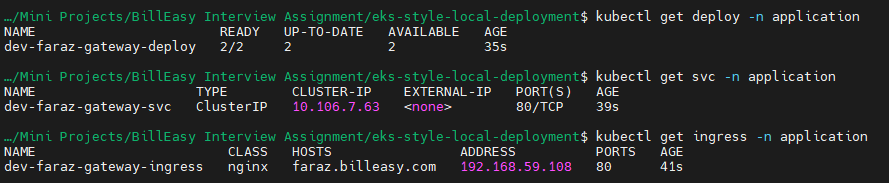


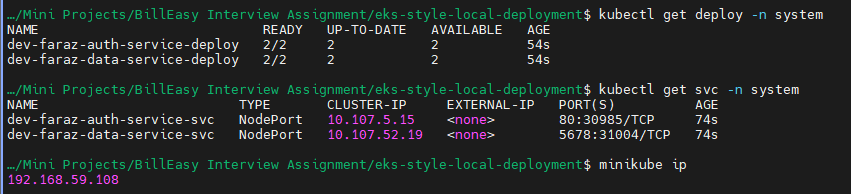
Now apply the kustomization:

* kubectl apply -k kustomize/overlays/dev

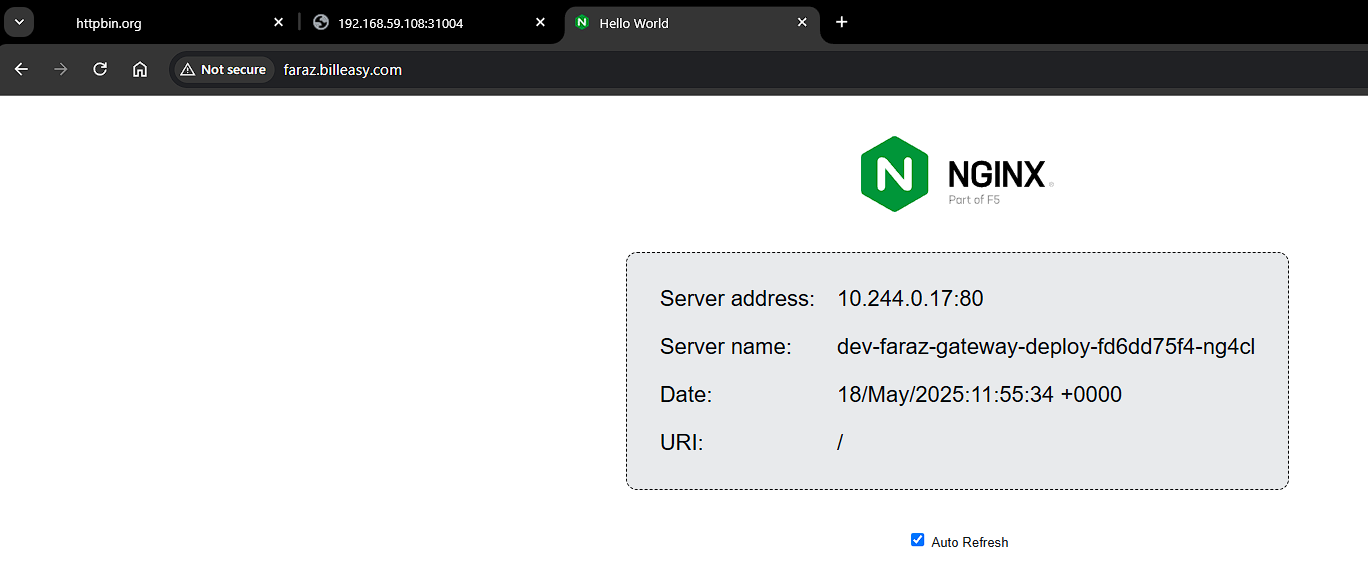


Verify your service, deployment and ingress(only for gateway):

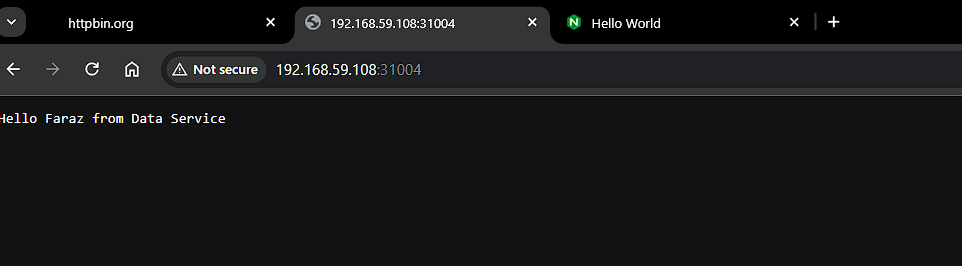




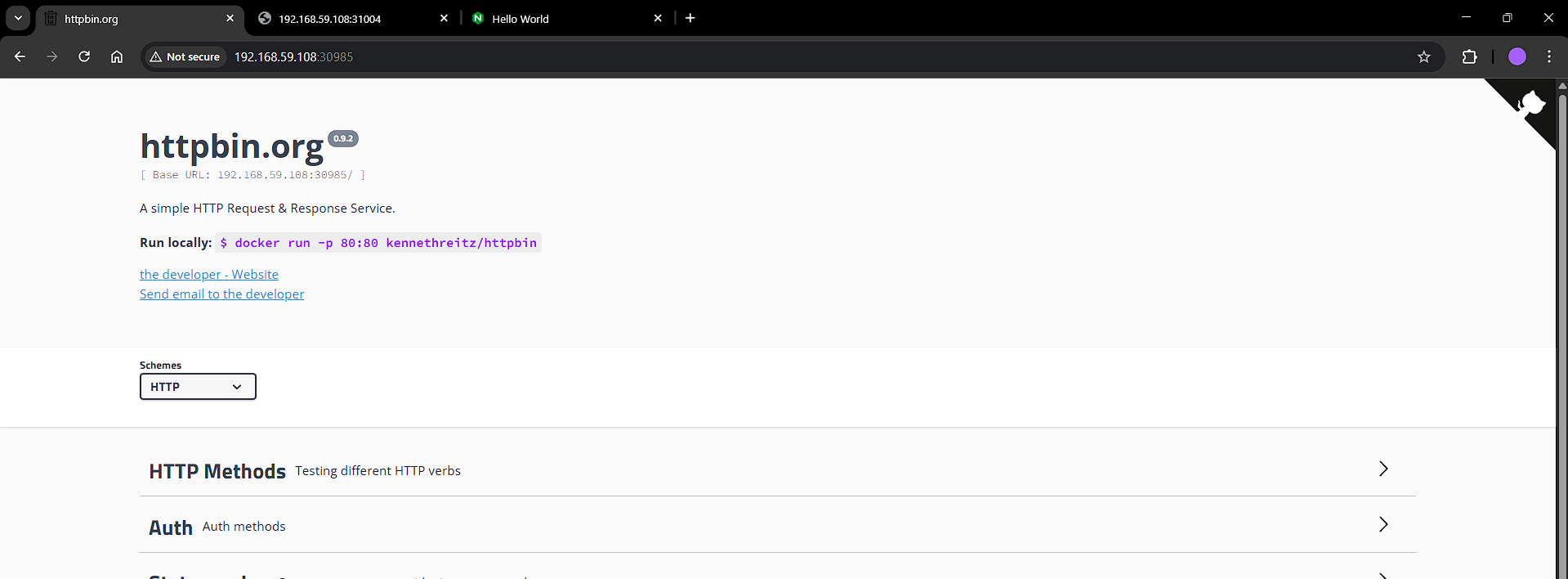
Verify your gateway ingress, svc, deploy on browser:



Verify your data service:



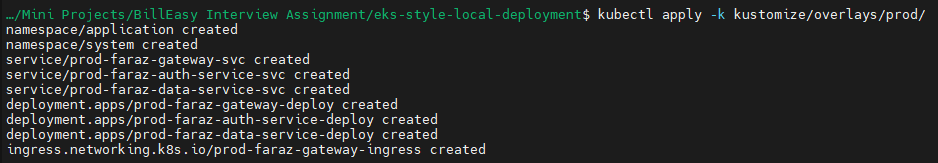
Verify your auth-service:



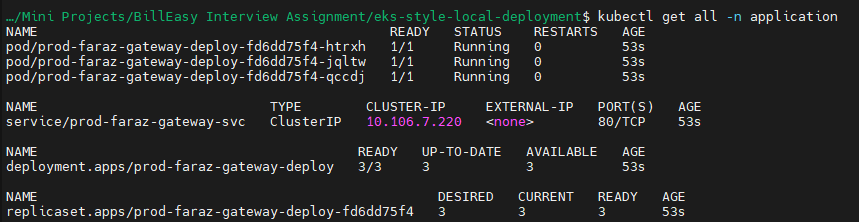
Now similarly, we will verify in prod:

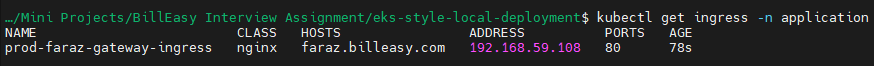
Apply the kustomization:

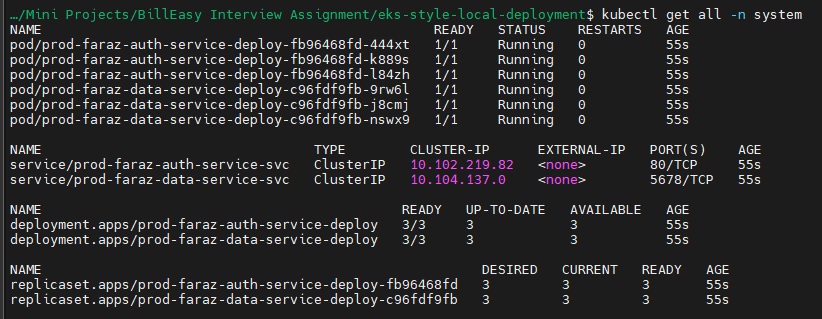
* kubectl apply -k kustomize/overlays/prod/



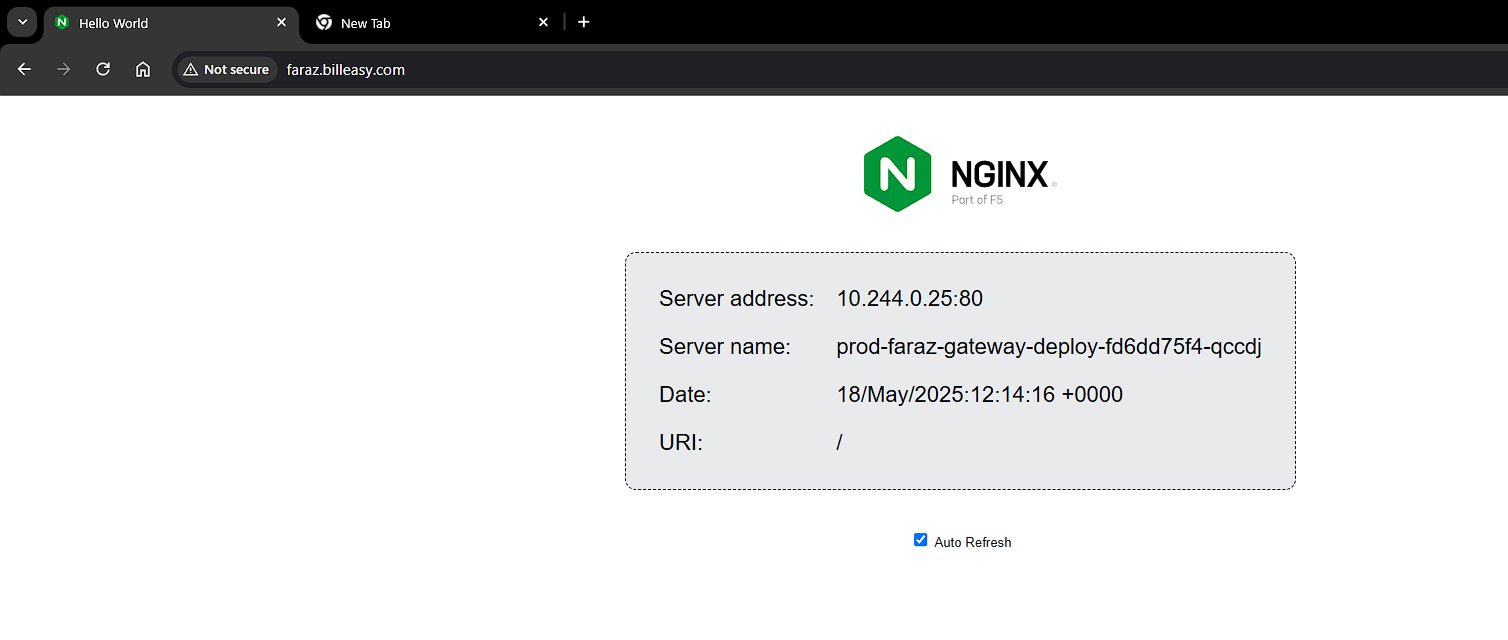
Verify your ingress, svc, deploy, pods:





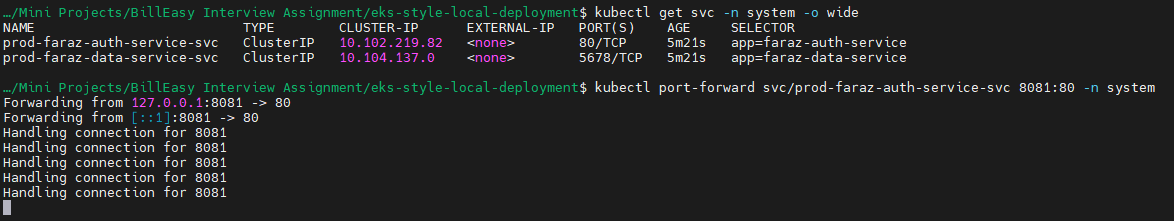


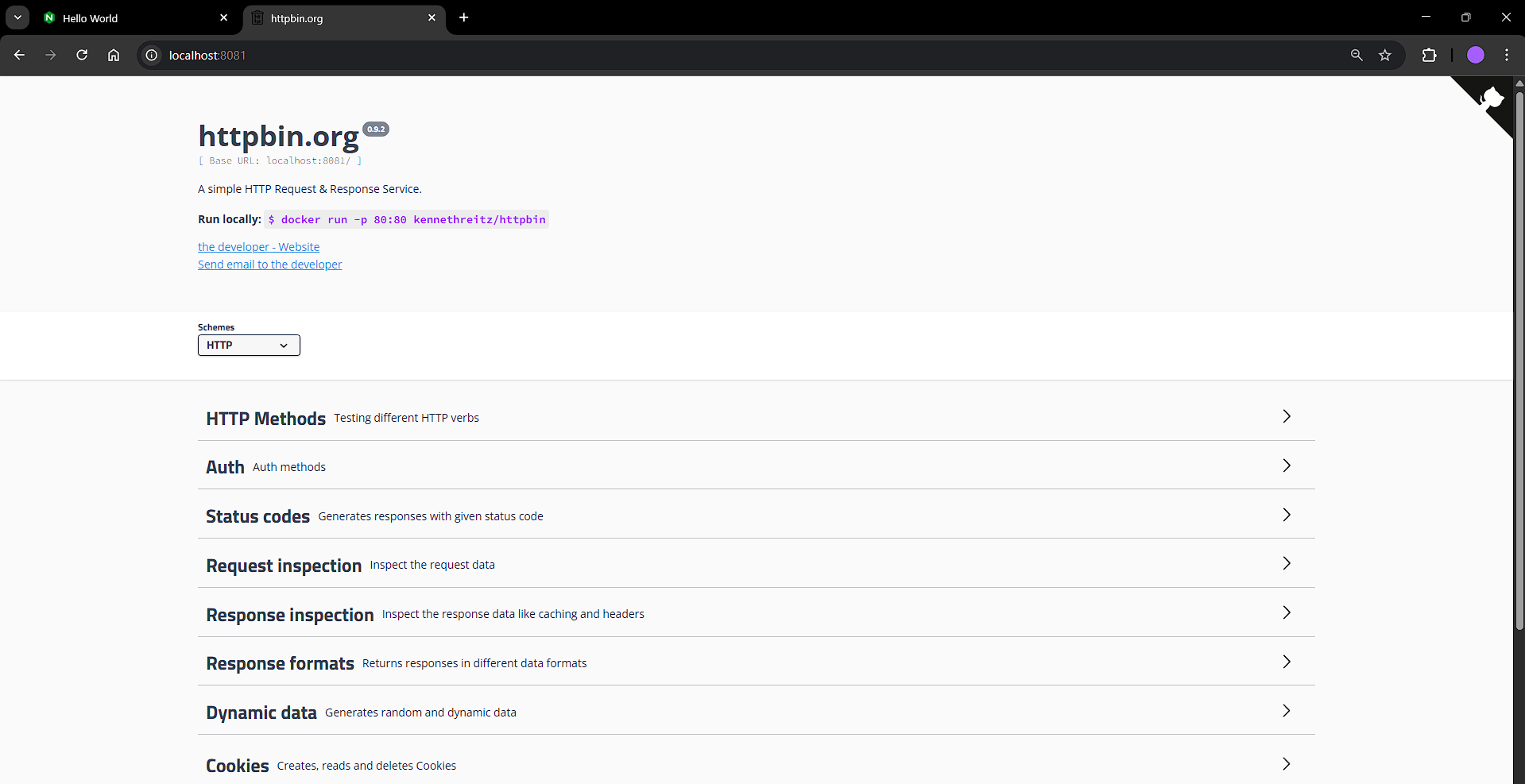
Visible in browser as ingress interacting with service with clusterIP mode and to prod-faraz-gateway-deploy deployment:



Using port forwarding to verify auth-service and data-service:

* kubectl port-forward svc/prod-faraz-auth-service-svc 8081:80 -n system





* kubectl port-forward svc/prod-faraz-data-service-svc 8082:5678 -n system

