1. **Create Ubuntu VM with Proper Network**

* Create a Virtual Machine with Ubuntu installed and ensure it has proper network connectivity, including access to the internet for package downloads and Git cloning.

1. **Install Docker**

snap install docker

* This command installs Docker using Snap package manager. Make sure Snap is installed on your Ubuntu VM if it's not already.

1. **Install KIND (Kubernetes IN Docker)**

wget -O /usr/local/bin/kind <https://github.com/kubernetes-> sigs/kind/releases/download/v0.20.0/kind-linux-amd64

chmod +x /usr/local/bin/kind

* KIND allows you to run local Kubernetes clusters using Docker containers.

1. **Install kubectl**

wget -O /usr/local/bin/kubectl <https://storage.googleapis.com/kubernetes-release/release/v1.27.1/bin/linux/amd64/kubectl>

chmod +x /usr/local/bin/kubectl

* kubectl is the command-line tool for interacting with Kubernetes clusters.

1. **Create KIND Cluster (qatest)**

kind create cluster --name qatest --config=kind-cluster.yaml

kubectl cluster-info --context kind-qatest

* This creates a new Kubernetes cluster named qatest using a configuration file (kind-cluster.yaml), and verifies the cluster's context.

1. **Clone qa-test Repository**

git clone https://github.com/Vengatesh-m/qa-test

* Clone the qa-test repository from GitHub to get your application code.

1. **Build Docker Images**

docker build -t vengatesh27/backend:v1 - < backend/Dockerfile

docker build -t vengatesh27/frontend:v1 - < frontend/Dockerfile

* Build Docker images for your backend and frontend applications from their respective Dockerfiles.

1. **Create and Deploy Kubernetes Services**

Assuming you have Kubernetes deployment YAML files (backend-deployment.yaml and frontend-deployment.yaml), apply them:

kubectl apply -f backend-deployment.yaml

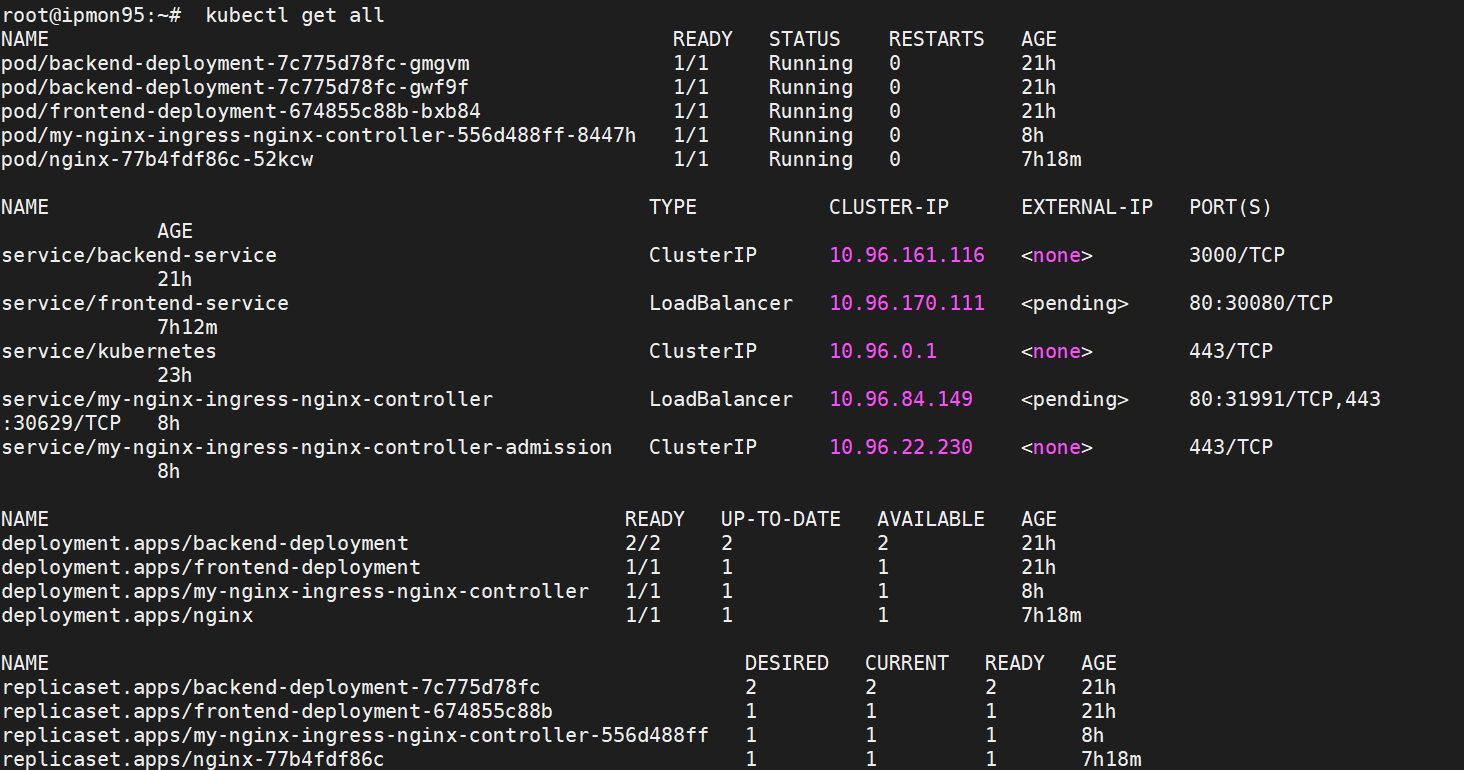
kubectl apply -f frontend-deployment.yaml

* Deploy your backend and frontend services to the Kubernetes cluster.

1. **Verify Services**

kubectl get all

e.g.



* Check the status of all Kubernetes resources (pods, services, deployments, etc.) to ensure your services are running correctly.

1. **Install Ingress Controller (NGINX)**

helm install my-nginx ingress-nginx/ingress-nginx

kubectl get service --namespace default my-nginx-ingress-nginx-controller --output wide --watch

* Install NGINX as the Ingress controller using Helm. Monitor the service creation until it's ready.

1. **Deploy Ingress Resource**

kubectl apply -f ingress.yaml

* Apply your Ingress resource configuration to define how external traffic should be routed to your services.

1. **Verify Application Accessibility**

* Access your application from a Chrome browser by navigating to the appropriate URL defined in your Ingress configuration.

**Additional Notes**

* Ensure that your Dockerfiles (backend/Dockerfile and frontend/Dockerfile) are correctly configured to build the Docker images for your applications.
* Customize kind-cluster.yaml, backend-deployment.yaml, frontend-deployment.yaml, and ingress.yaml according to your specific application requirements and network configurations.
* Make sure all necessary dependencies (like Helm for installing NGINX Ingress) are installed on your Ubuntu VM.

By following these steps, you should be able to set up a Kubernetes environment locally using KIND, deploy your applications using Docker containers, configure an Ingress resource for external access, and verify everything is working as expected. Adjustments may be needed based on your specific application and networking setup.