1. **Technology, components and tools:**

1.1. **Kubernetes:**

Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications. This cluster will be deployed on a local machine to host and run Docker Images of applications that will server requests triggered by end user.

1.2. **Minikube:**

Tool that will be used to deploy Kubernetes cluster on a local machine. Kubectl will be also installed as well to interact with Kubernetes cluster using CLI

1.3 **Docker Images:**

Docker Images are built to be used in Kubernetes Deployments

1.4 **airports-assembly and countries-assembly jar files** :

Those packages are pre-created and provided so it can be triggered using one command. Those jar files will be run as foreground process in Docker containers.

1.5 **Ansible**:

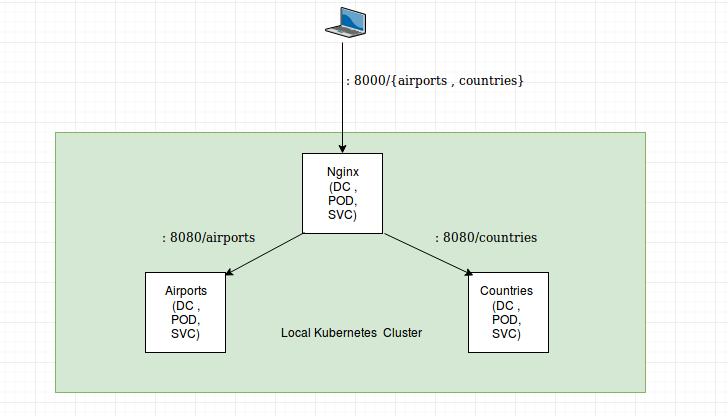
A tool that is used to run the whole stack using one command and automate the deployment of technologies used in this assessment then create all Kubernetes objects.

1.6. **Nginx**:

Nginx is used to work as a proxy between End User requests and services in the backend.

Nginx is listening on port 8000 and it forwards requests to either airports-assembly service or countries-assembly service based on the URL.

**2. Cluster Architecture:**



**3. Deployment Playbook:**

**3.1 Kuberenetes Role :**

This role is responsible for local deployment of Kubernetes Cluster. It will download and install needed Visualization packages (kvm2). It will also download and install docker KVM2 driver.

After that, kubectl is installed to interact with the cluster over CLI later.

Finally, minikube is executed to initialize the deploy a plain Kubernetes cluster on this machine.

**3.2 build-air-ports Role:**

This role will build docker image based on airports-assembly jar file. Later once the container is started, jar file will be running as foreground process and server request over port 8080

**3.3 build-countries Role:**

This role will build docker image based on countries-assembly jar file. Later once the container is started, jar file will be running as foreground process and server request over port 8080

**3.4 build-nginx Role:**

This role will build docker image that run nginx process in foreground. It expose port 8000

**3.5 docker-eval Role:**

This is an auxiliary role that is called in all of above build-\* roles. This role is only responsible to switch the environment from machine docker daemon to minikube docker daemon. As a results, built images can be used in Kubernetes Cluster deployed locally.

**3.6 deploy-air-ports Role:**

In scope of this rule, following will be done:

- airports namespace is created.

- NetworkPolicy is created to allow access from nginx namespace only.

- airports-assembly deployment is created.

- Expose airports-assembly locally within the cluster so it can be reached from nginx using service name

**3.7 deploy-countries Role:**

In scope of this rule, following will be done:

- countries namespace is created.

- NetworkPolicy is created to allow access from nginx namespace only.

- countries-assembly deployment is created.

- Expose countries-assembly locally within the cluster so it can be reached from nginx using service name

**3.7 deploy-nginx Role:**

In scope of this rule, following will be done:

- nginx namespace is created and labeled.

- nginx deployment is created.

- Expose nginx service using externalIP. (Since this is local deployment, we are assigning externalIP manually. This IP will be similar to the IP used in VM initialized by minikube. IP can be retrieved using command “sudo kubectl get svc -n nginx” )

**3. Deployment Execution:**

Please refer to README file attached to the repo.