Version: 1.00

Last Update: 1/20/2019

1. **Resource and Tools:**

**Kubernetes and Minikube:**

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. Minikube is a 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

**Docker Images:**

Docker Images are built to be used in Kubernetes Deployments

**Resources**

Java version is 1.8

The services are available for download:

country-service:

URL: [countries-assembly-1.0.1.jar](https://workable.com/nr?l=https%3A%2F%2Fs3-eu-west-1.amazonaws.com%2Fdevops-assesment%2Fcountries-assembly-1.0.1.jar)

SHA-1: 92bf1a691fc6dc835b21e0d74102c41ad84635f9 countries-assembly-1.0.1.jar

airport-service v1.0.1:

URL: [airports-assembly-1.0.1.jar](https://workable.com/nr?l=https%3A%2F%2Fs3-eu-west-1.amazonaws.com%2Fdevops-assesment%2Fairports-assembly-1.0.1.jar)

SHA-1: 0bd35ea555b9aabaf30d255f3cb90aedf6bebca1 airports-assembly-1.0.1.jar

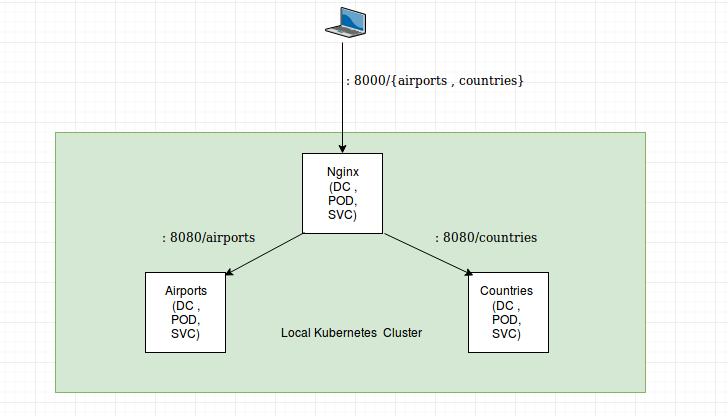
airport-service v1.1.0:

URL: [airports-assembly-1.1.0.jar](https://workable.com/nr?l=https%3A%2F%2Fs3-eu-west-1.amazonaws.com%2Fdevops-assesment%2Fairports-assembly-1.1.0.jar)

SHA-1: 40d479396e37f61b88913ad3de3a421c75ed8f45 airports-assembly-1.1.0.jar

**2. Architecture:**

**Cluster Arc:**



**Project Arch:**

.

├── Readme.md

├── deployments (All kubernetes related deployments are here!)

│   ├── 0-namespace.yaml (NameSpace For Project)

│   ├── airports

│   │   ├── airport-apiserver-deploy.yaml

│   │   ├── airport-apiserver-exposeservice.yaml

│   │   └── airports-ingress.yaml

│   ├── countries

│   │   ├── country-apiserver-deploy.yaml

│   │   ├── country-apiserver-exposeservice.yaml

│   │   └── country-ingress.yaml

│   ├── jenkins (CI/CD)

│   │   ├── Jenkinsfile

│   │   ├── jenkins-server-deploy.yaml

│   │   └── jenkins-server-exposeservice.yaml

│   └── kube-addons

│   ├── calico.yaml

│   └── net-policies.yaml

├── deployments4jenkins

│   ├── 0-namespace.yaml

│   ├── airports

│   │   ├── airport-apiserver-deploy.yaml

│   │   ├── airport-apiserver-exposeservice.yaml

│   │   └── airports-ingress.yaml

│   ├── countries

│   │   ├── country-apiserver-deploy.yaml

│   │   ├── country-apiserver-exposeservice.yaml

│   │   └── country-ingress.yaml

│   ├── jenkins

│   │   ├── Jenkinsfile

│   │   ├── jenkins-server-deploy.yaml

│   │   └── jenkins-server-exposeservice.yaml

│   └── kube-addons

│   ├── calico.yaml

│   └── net-policies.yaml

├── documents

│   └── HANDOVER.pages

├── jenkins-docker (CI/CD)

│   ├── Dockerfile

│   ├── configs

│   │   └── config.xml

│   ├── groovy

│   │   └── default-config.groovy (set user/pass for Jenkins during the setup)

│   ├── jenkins.sh

│   ├── job (Our Sample Jobs to get imported to Jenkins)

│   │   └── jobs

│   │   ├── airports

│   │   │   └── config.xml

│   │   └── countries

│   │   └── config.xml

│   └── plugins.txt (List of needed plugins)

├── lunatech-apps-docker (Apps provided by Lunatech)

│   ├── airport

│   │   ├── Dockerfile

│   │   ├── airports-assembly-1.0.1.jar

│   │   └── airports-assembly-1.1.0.jar

│   └── country

│   ├── Dockerfile

│   └── countries-assembly-1.0.1.jar

└── start-travis.sh (Auto CI by the Travis)

**3. Requirements**

The applicant can choose any technology stack, and should meet the following requirements:

* The entire stack should be able to run locally on a developer's machine
* The country- and airport-service run isolated from each other
  + No inter-communication between the two services is possible
  + No direct communication from the "outside world" is possible with the two services
* A reverse-proxy and/or load-balancer exposes the services on port 8000
* Initially, airports version 1.0.1 is deployed
* An update to the airports service from version 1.0.1 to version 1.1.0 can be triggered at the code review without causing a service interruption.

**4. Deployment:**

- NetworkPolicy is created to allow access from nginx namespace only within follow steps:

1. `minikube start --extra-config=kubelet.network-plugin=cni --network-plugin=cni`

To be able to constrain inter-communication between the two services do as follow:

2. Install calico-etcd (likely you could use Kubernetes datastore as well)

kubectl apply -f https://docs.projectcalico.org/v3.2/getting-started/kubernetes/installation/hosted/etcd.yaml

3. Create rbac for etcd

kubectl apply -f https://docs.projectcalico.org/v3.2/getting-started/kubernetes/installation/rbac.yaml

4. Grab the hosted (etcd) calico manifest

curl https://docs.projectcalico.org/v3.2/getting-started/kubernetes/installation/hosted/calico.yaml -O

5. Edit the manifest to make "etcd\_endpoints" point to the etcd server you just installed. NOTE - this command needs to be adjusted with your etcd endpoint IP address

sed -i -e "s/10\.96\.232\.136/$(kubectl get service -o json --namespace=kube-system calico-etcd | jq -r .spec.clusterIP)/" calico.yaml

6. Apply your edited calico.yaml

kubectl apply -f deployment/kube-addons/

- airports namespace is created.

- airports-assembly deployment is created.

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

**5. Deployment Execution:**

Please refer to README file attached to the repo.