Flow from scratch to K8s cluster deployment

1. **Creating a Dockerfile for the specific service**
2. **Jenkins pipeline(Job) to build the project, Build the Docker image, Push the Image to ACR**
3. **K8s Manifest files creation:**
    a. **Deployment**
    b. **Service**
    c. **Ingress**
4. **Creation of Ingress controller in Azure**
5. **Map domain name in GCP with azure app gateway k8s public ip**
6. **Generate pfx certificate and upload to azure app gateway**
7. **Create secret to pull the docker images in AKS from ACR.**
8. **Deploy service in K8s cluster**
9. **Applying changes to Deployment file.**
10. **Verify the service deployment**

Dockerfile example:

```
FROM argoid.azurecr.io/argoid-base-centos76-java11:1.0.1
ARG USER=argoid
ARG UID=1000
ARG GID=1000
ARG PW=argoid
ARG appName=wiser-console
ARG workingDir=/opt/wiser-console
RUN useradd -m ${USER} --uid=${UID} && echo "${USER}:${PW}" | chpasswd;
\
    chown ${UID}:${GID} /opt
USER ${UID}:${GID}
WORKDIR /opt
COPY --chown=${UID}:${GID} ${appName}.tar.gz /opt/
RUN mkdir ${appName}; \
    mkdir shared; \
    tar -xzf *.tar.gz --directory ${appName} --strip-components=1; \
    rm *.tar.gz; \
    chmod -R 755 /opt/wiser-console/bin/start-wiser-docker.sh
COPY --chown=${UID}:${GID} shared ${workingDir}/shared
WORKDIR ${workingDir}
CMD ["sh", "-c", "bin/start-wiser-docker.sh"]
EXPOSE 8092
```

Jenkinsfile:

```
//def git_url = "git@bitbucket.org:argoid_ai/argoid-nlp.git"
//def git_branch="feature/Dockerization"
def git_credentialsid = "bitbucket_devops_credentials"
def appName = "wiser-console"

def dockerUSER ='argoid'
def dockerUID = '1000'
def dockerGID = '1000'
def dockerPW = 'argoid'
```

```
pipeline {
    agent any
    tools {
        maven 'maven_38'
        jdk 'Java11'
    }
    environment {
        imagename = 'argoidpepsico.azurecr.io/pepsico-search-engine'
        registry = 'https://argoid.azurecr.io'
        dockerImage = ''
        registryCredential = 'AzureCR'

        pepsiCoRegistry = 'https://argoidpepsico.azurecr.io'
        pepsiCoregistryCredential = 'PepsiCoACRCredentials'
    }
    stages {
        stage('Clean Workspace') {
             //when {expression { false }}
            steps {
                cleanWs()
            }
        }

        stage('Git Checkout') {
             //when {expression { false }}
            steps {
               // git credentialsId: "${git_credentialsid}", url:
"${git_url}", branch: "${git_branch}"
               checkout([$class: 'GitSCM', branches: [[name:
'721694dc9fbd0ef890cc021dff179d252cb22129']], extensions: [[$class:
'CloneOption', noTags: false, reference: '', shallow: false, timeout:
20],[$class: 'GitLFSPull']], userRemoteConfigs: [[credentialsId:
'bitbucket_devops_credentials', url: 'git@bitbucket.org:argoid_ai
/argoid-nlp.git']]])
            }
        }
        stage('Build Project') {
             //when {expression { false }}
            steps {
                    sh 'mvn clean install -DskipTests'
            }
        }
         stage('Capture Version') {
              steps {
                  script {
                    dir('nlp-wiser/wiser-console') {
                        def pom = readMavenPom file: 'pom.xml'
                        env.version = pom.parent.version
                        echo "Version -- ${env.version}"
```

```
                    }
                }
             }
         }

        stage('Move tarball and shared dir') {
            steps {
                    dir("nlp-wiser/wiser-console/"){
                    sh "cp target/*.tar.gz ${appName}.tar.gz"
                }

                sh "mkdir -p nlp-wiser/wiser-console/shared/cache/ &&
cp -Rf shared/cache/{admin,${OrgName}} nlp-wiser/wiser-console/shared
/cache/"
            }
        }
        stage('Docker Build') {
            steps {
                script {
                    dir("nlp-wiser/wiser-console/"){
                        // Need to add --no-cache parameter while
building docker image
                            withDockerRegistry(credentialsId:
registryCredential, url: registry) {
                            dockerImage = docker.build(imagename,"--
build-arg USER=$dockerUSER --build-arg UID=$dockerUID --build-arg
GID=$dockerGID --build-arg PW=$dockerPW .")
                        }

                    }
                }
            }
        }

        stage('Docker Image push to ACR') {
            //when {expression { false }}
            steps {
                script {
                    withDockerRegistry(credentialsId:
pepsiCoregistryCredential, url: pepsiCoRegistry) {
                        dockerImage.push(env.version+"-${OrgName}")
                        //dockerImage.push(env.version)
                        //dockerImage.push('latest')
                    }
                }
            }
        }

        stage('Cleaning up') {
           // when {expression { false }}
```

```
            steps {
              sh "docker rmi $imagename:${env.version}-${OrgName}"
              //sh "docker rmi $imagename:${env.version}"
              //sh "docker rmi $imagename:latest"
            }
        }
    }
}
```

pepsico-search-engine-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: pepsico-search-engine
  namespace: pepsico-ns
spec:
  selector:
    matchLabels:
      app: pepsico-search-engine
  replicas: 1
  template:
    metadata:
      labels:
        app: pepsico-search-engine
    spec:
      containers:
        - name: pepsico-search-engine
          image: argoidpepsico.azurecr.io/pepsico-search-engine:0.0.7-
snacks_com
          imagePullPolicy: Always
          ports:
            - containerPort: 8092
            - containerPort: 40024

      imagePullSecrets:
        - name: argoid-pepsico-docker-registry
```

pepsico-search-engine-service.yaml

```
---
apiVersion: v1
kind: Service
metadata:
  name: pepsico-search-engine
  namespace: pepsico-ns
  labels:
    app-svc: pepsico-search-engine
spec:
  type: ClusterIP
  selector:
    app: pepsico-search-engine
  ports:
    - name: port8092
      port: 8092
      targetPort: 8092
    - name: port40024
      port: 40024
      targetPort: 40024
```

pepsico-search-engine-ingress.yaml

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: pepsico-search-engine-ingress
  namespace: pepsico-ns
  annotations:
    appgw.ingress.kubernetes.io/appgw-ssl-certificate: pepsico-search.
argoid.com
    kubernetes.io/ingress.class: azure/application-gateway
    appgw.ingress.kubernetes.io/health-probe-status-codes: "200-599"
spec:
  rules:
  - host: pepsico-search.argoid.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: pepsico-search-engine
            port:
              number: 8092
```

Syntax

```
kubectl create secret docker-registry dockersecret --docker-
server=<your-registry-server> --docker-username=<your-name> --docker-
password=<your-pword> --docker-email=<your-email>
```

Example

```
kubectl create secret docker-registry argoid-docker-registry \
    --namespace ecomm-reco-rest-namespace \
    --docker-server=argoid.azurecr.io \
    --docker-username=argoid \
    --docker-password=VjW2h9DX8Oo7+6kNI1QQd1GXTvLEHHDv
```