# Refer to my github notes

# <https://github.com/cloudnativedemo/icp-notes/blob/master/microclimate_notes.md>

# Customise Microclimate to make it work with MQ projects (and others)

## Prepare ICP for microclimate deployment

Refer [here](https://github.com/IBM/charts/tree/master/stable/ibm-microclimate) for full instructions

**1. Create pipeline deployment namespace**

kubectl create namespace microclimate-pipeline-deployments

**2. Edit ClusterImagePolicy**

kubectl edit clusterimagepolicies ibmcloud-default-cluster-image-policy

To add the following:

- name: mycluster.icp:8500:\*

- name: docker.io/maven:\*

- name: docker.io/lachlanevenson/k8s-helm:\*

- name: docker.io/jenkins/\*

**3. Create Docker Registry secret to microclimate namespace**

kubectl create secret docker-registry microclimate-registry-secret \

--docker-server=mycluster.icp:8500 \

--docker-username=admin \

--docker-password=admin \

--docker-email=null

**4. Initialise Helm and login**

helm init --client-only --skip-refresh

cloudctl login -a https://mycluster.icp:8443 -u admin -p admin -c id-mycluster-account -n default --skip-ssl-validation

**5. Create Helm secret**

export HELM\_HOME=$HOME/.helm

kubectl create secret generic microclimate-helm-secret --from-file=cert.pem=$HELM\_HOME/cert.pem --from-file=ca.pem=$HELM\_HOME/ca.pem --from-file=key.pem=$HELM\_HOME/key.pem

**6. Create Docker Registry secret for microclimate-pipeline-deployments namespace**

kubectl create secret docker-registry microclimate-pipeline-secret \

--docker-server=mycluster.icp:8500 \

--docker-username=admin \

--docker-password=admin \

--docker-email=null \

--namespace=microclimate-pipeline-deployments

**7. Update ImagePullSecret for microclimate-pipeline-deployments namespace**

kubectl patch serviceaccount default --namespace microclimate-pipeline-deployments -p '{"imagePullSecrets": [{"name": "microclimate-pipeline-secret"}]}'

**8. Customise Jenkins library** By default, the Jenkins library parameter is pointing to <https://github.com/microclimate-dev2ops/jenkins-library> This Jenkins library was a part of the Microclimate DevOps process. When a pipeline is created within a project in Microclimate, microclimate will create a Jenkins pipeline. The pipeline uses this library to .. 1. Pull the code from github repo . .. 2. Build a Docker image based on a Dockerfile found in the repo . .. 3. Authenticate and push the image into ICP's private registry . .. 4. Notify Microclimate to move to the next stage (e.g. deploy) . .. 5. Microclimate 'helm deploy' the helm chart found in the repo (by default it's under the /chart directory) .

Unfortunately, Microclimate only deploy it's supported project types e.g. Swift, NodeJS, Java/Liberty or Springboot. The easiest way to address this limitation is to fork and update the Jenkins library and inject the 'helm deploy' scriptlet onto step 4 (line 400 of microserviceBuilderPipeline.groovy)

container ('helm') {

echo "Attempting to deploy the test release"

def deployCommand = "helm install ${realChartFolder} --values pipeline.yaml --namespace ${namespace} --name ${helmRelease}"

if (fileExists("chart/overrides.yaml")) {

deployCommand += " --values chart/overrides.yaml"

}

if (helmSecret) {

echo "Adding --tls to your deploy command"

deployCommand += helmTlsOptions

}

testDeployAttempt = sh(script: "${deployCommand} > deploy\_attempt.txt", returnStatus: true)

if (testDeployAttempt != 0) {

echo "Warning, did not deploy the test release into the test namespace successfully, error code is: ${testDeployAttempt}"

echo "This build will be marked as a failure: halting after the deletion of the test namespace."

}

printFromFile("deploy\_attempt.txt")

}

* ***Note:*** in my deployCommand, I've created one new variable ${helmRelease}. The variable is defined on the top of the script (line 56 of the microserviceBuilderPipeline.groovy). Alternatively, you can just reuse ${image} as your helm release name .

def helmRelease = (config.releaseName ?: config.image ?: "").trim()

* My forked updated Jenkins library repo can be found [here](https://github.com/cloudnativedemo/jenkins-library) .

**8. Deploy Microclimate helm chart**

#### ****Via Helm command line****

* **Add ibm-charts Helm repo**

helm repo add ibm-charts https://raw.githubusercontent.com/IBM/charts/master/repo/stable/

* **Deploy microclimate Helm chart**

helm install --name microclimate --namespace default --set global.rbac.serviceAccountName=micro-sa,jenkins.rbac.serviceAccountName=pipeline-sa,hostName=microclimate.172.23.52.247.nip.io,jenkins.Master.HostName=jenkins.172.23.52.247.nip.io,jenkins.Pipeline.Template.RepositoryUrl=https://github.com/cloudnativedemo/jenkins-library.git,jenkins.Pipeline.Template.Version=master ibm-charts/ibm-microclimate --tls

***Note:*** Replace <172.23.52.247> with your <PROXY\_IP>

#### ****Via ICP catalog****

* Select ibm-microclimate from ICP catalog > click Configure
* Provide values for the following parameters:
  + Helm release name: your-microclimate-release-name
  + Namespace: default (or your preferred namespace)
  + Microclimate hostname: microclimate.172.23.52.247.nip.io (replace with your <microclimate.PROXY\_IP.nip.io> or your own hostname)
  + Ensure that you've already created Persistent Volumes for Microclimate and Jenkins
  + Service account name for Portal: micro-sa
  + Jenkins library repository: <https://github.com/cloudnativedemo/jenkins-library.git>
  + Jenkins hostname: jenkins.172.23.52.247.nip.io (replace with your <jenkins.PROXY\_IP.nip.io> or your own hostname)
  + Service account name: pipeline-sa
  + Click deploy

## Create a project in Microclimate and Deploy

Once the Microclimate helm deployment completed, you can start to deploy your custom project

1. Make sure that your project contains a Dockerfile, Jenkinsfile and a chart directory (for helm chart)
2. Launch Microclimate ([https://microclimate.172.23.52.247.nip.io](https://microclimate.172.23.52.247.nip.io/) - replace with your own microclimate hostname) and accept licensing agreement (for first launch only)
3. Select Projects > Click New Project
4. Select Java project type and provide a project name > click Next
5. Select Microprofile/J2EE and keep default value for Context root > click Create
6. Once the project is created, select Pipeline on the left menu
7. Click Create pipeline, and provide name and github repo of the pipeline > click Create pipeline to create a Jenkins pipeline
8. Switch to Jenkins ([https://jenkins.172.23.52.247.nip.io](https://jenkins.172.23.52.247.nip.io/) - replace with your Jenkins hostname) to see if the pipeline has been created and built (refer to the troubleshooting section below if you have to wait for too long)

## Troubleshooting

* When your Jenkins pipeline keeps looking for an executor for too long, there's probably an error occurred within your Jenkins containers. To identify the issue:
  + Identify the Jenkins pod name: kubectl get pods -n <NAME\_SPACE\_WHERE\_JENKINS\_INSTALLED> | grep jenkins
  + View the log: kubectl log -n <NAME\_SPACE\_WHERE\_JENKINS\_INSTALLED> <JENKINS\_POD\_NAME> -f
  + Most of the case I found caused by cluster image policy is not defined, you might need to update the default clusterimagepolicy
    - kubectl edit clusterimagepolicies ibmcloud-default-cluster-image-policy