# Deploying the Proof of Concept in Kubernetes

## Problem Statement

In this final step, we want to deploy the containerized legacy application in Kubernetes. Instead of doing the deployment through VSTS, we will be deploying from the command line using kubectl

Prerequisite

1. Docker Engine and Docker CLI
2. Chocolatey
3. Kubectl
4. A Kubernetes Windows cluster. You should be able to use the ACS cluster deployed in module 5 – orchestrator.

Tasks:

1. Install kompose to convert the docker-compose file to Kubernetes YAML files.

choco install kubernetes-kompose

1. Generate Kubernetes YAML files from the docker-compose file

kompose convert

Note: If kompose complains about an unsupported version of docker-compose, edit docker-compose file and change docker-compose version from 2.1 or current to 3.0 or a supported version.

1. Create an Azure Container Registry (ACR) if you don't already have one

Go to the Azure Portal, search for ACR and just fill in the information

1. Publish legacyapp & sqldatabase images to ACR
   1. Login to the ACR environment

docker login {youracr.azurecr.io}

Note: you can get the ACR name and password from the Azure Portal, ACR environment in Access Keys.

* 1. Tag your container images

docker tag legacyapp {youracr.azurecr.io}/legacyapp:{yourtag}

docker tag sqldatabase {youracr.azurecr.io}/sqldatabase:{yourtag}

* 1. Push your images

docker push {youracr.azurecr.io}/legacyapp

docker push {youracr.azurecr.io}/sqldatabase

1. Configure Kubernetes to have access to your Azure Registry

<https://kubernetes.io/docs/tasks/configure-pod-container/pull-image-private-registry/>

kubectl create secret docker-registry {secretname} --docker-server=https://{youracr loginserverurl}

            --docker-username={username} --docker-password={password} --docker-email={youremail}

example:

kubectl create secret docker-registry regcred --docker-server=https://containerws.azurecr.io --docker-username=containerws --docker-password=Vni9feStEA40nCLf6xPIcNgC7Bk+1VG1 [--docker-email=msaras28@hotmail.com](mailto:--docker-email=msaras28@hotmail.com)

1. Edit deployment YAML files (2: 1 for legacy app and 1 for the database) to pull images from your ACR

change from

- image: sqldatabase

to:

- image: {youracr.azurecr.io}/sqldatabase

example:

- image: {youracr.azurecr.io}/sqldatabase

1. Edit deployment YAML files to include imagePullSecrets setting in spec section:

spec:

            containers:

            ....

            imagePullSecrets:

            - name: {secretname - name of the secret you used in step number 5}

1. Edit service YAML file for the front-end to add a service type:

spec:

spec:

  .....

  type: LoadBalancer

1. Go to Kubernetes dashboard to run the YAML files or in command line:

kubectl create -f database-deployment.yaml

kubectl create -f database-service.yaml

kubectl create -f frontend-deployment.yaml

kubectl create -f frontend-service.yaml

1. Test the application:
   1. Get the external IP address of the front-end service:

            Bring up Kubernetes dashboard, go to services, and then click on {front-end service name} or in command line kubectl get services {front-end service name}.

            Browse the external IP address.

Note: setting up a load balancer and getting an external IP address for the front-end service on Azure may take a bit. So just be patient and refresh your browser or rerun the kubectl command again.