Containers, ACR & Kubernetes

Wednesday, November 13, 2019

3:47 PM

**High Level Overview**

* Create the code
* Create a docker container image using your code - you create a Dockerfile (with no extension). In the Docker you specify what you want wrap in the container. You can also create a .dockerignore file as well. In this file you specified what is to be ignored from the container image
* Create a label for your container image using the login (ACR) server, e.g., a version label
* Push the container image into the Azure Container registry (ACR). you can view the new addition by listing docker images for your resource group
* Create a Azure Kubernetes (AKS) cluster in your resource group and attach your ACR instance to it. In the cluster definition, you can specify the minimum, maximum container instances; the acr registry that contains the workload.
* Create/Update a manifest (.YAML) file. Ensure the proper login server (ACR) and container image version are updated and saved.
* You **apply** the manifest file to your AKS cluster
* To monitor progress, use the [kubectl get service](https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#get) - watch command
* After a little while, you'll be able to see your public IP address, which you can use it to smoke test your deployment

**P.S. As you're progressing, you can always list the contents to see your changes and progress**

**Depending upon your use case, you can also do the following:**

* [Scale an application and Kubernetes infrastructure](https://docs.microsoft.com/en-us/azure/aks/tutorial-kubernetes-scale)
* [Update an application running in Kubernetes](https://docs.microsoft.com/en-us/azure/aks/tutorial-kubernetes-app-update)
* [Upgrade AKS cluster](https://docs.microsoft.com/en-us/azure/aks/tutorial-kubernetes-upgrade-cluster)

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

[Create container images from an application](https://docs.microsoft.com/en-us/azure/aks/tutorial-kubernetes-prepare-app)

* Get application code - The sample application is a basic voting app. It has a front-end web component and a back-end Redis instance. The web component is packaged into a custom container image. The Redis instance uses an unmodified image from Docker Hub. Run the following command: git clone <https://github.com/Azure-Samples/azure-voting-app-redis.git>
* Change into the cloned directory: cd azure-voting-app-redis. It contains the **application source code, a pre-created Docker compose file, and a Kubernetes manifest file**.

* Create container images by using [Docker Compose](https://docs.docker.com/compose/) to automate building container images and the deployment of multi-container applications. Use the sample docker-compose.yaml file to create the container image, download the Redis image, and start the application: docker-compose up -d
* See the created images by using the [docker images](https://docs.docker.com/engine/reference/commandline/images/) command. Three images have been downloaded or created. The *azure-vote-front* image contains the front-end application and uses the nginx-flask image as a base. The redis image is used to start a Redis instance: $ docker images command
* Run the [docker ps](https://docs.docker.com/engine/reference/commandline/ps/) command to see the running containers: $ docker ps
* Test the application locally go to: <http://localhost:8080> in a local web browser

[Upload container images to the Azure Container Registry](https://docs.microsoft.com/en-us/azure/aks/tutorial-kubernetes-prepare-acr)

* Create an Azure Container Registry
* Create a resource group: az group create --name myResourceGroup --location westus
* Create an Azure Container Registry instance: az acr create --resource-group myResourceGroup --name dcheemaacr1 --sku Basic
* Log in to the container registry: az acr login --name dcheemaacr1
* Tag a container image
  + See a list of the current local images, use the [docker images](https://docs.docker.com/engine/reference/commandline/images/) command: $ docker images
* Get the login server address, use the [az acr list](https://docs.microsoft.com/en-us/cli/azure/acr) command and query for the loginServer as follows: az acr list --resource-group myResourceGroup --query "[].{acrLoginServer:loginServer}" --output table
* Tag your local azure-vote-front image with the acrloginServer address of the container registry: docker tag azure-vote-front <acrLoginServer>/azure-vote-front:v1. Add version # v1, for example
* Verify the tags are applied, run [docker images](https://docs.docker.com/engine/reference/commandline/images/) again: $ docker images
  + Machine generated alternative text:
    REPOSITORY 
    azure-vote-front 
    mycontainerregistry. azurecr . io/azure -vote -front 
    redis 
    tiangolo/uwsgi -nginx- flask 
    TAG 
    latest 
    latest 
    flask 
    IMAGE ID 
    eaf2b9c57e5e 
    eaf2b9c57e5e 
    alb99da73de5 
    788ca94b2313 
    CREATED 
    8 
    8 
    7 
    8 
    minutes ago 
    minutes ago 
    days ago 
    months ago 
* Push images to registry - with image built and tagged, push the *azure-vote-front* image to the ACR (dcheemaacr1). Use [docker push](https://docs.docker.com/engine/reference/commandline/push/) and provide your own *acrLoginServer* address for the image name as follows: docker push <acrLoginServer>/azure-vote-front:v1
* List images in registry: az acr repository list --name <acrName> --output table
  + Machine generated alternative text:
    Result 
    azure-vote-front 
  + To see the tags for a specific image, use the [az acr repository show-tags](https://docs.microsoft.com/en-us/cli/azure/acr/repository) command
    - Machine generated alternative text:
      Result 

[Deploy an AKS cluster](https://docs.microsoft.com/en-us/azure/aks/tutorial-kubernetes-deploy-cluster)

* Create an AKS cluster using [az aks create](https://docs.microsoft.com/en-us/cli/azure/aks#az-aks-create): az aks create --resource-group myResourceGroup --name myAKSCluster --node-count 2 --generate-ssh-keys --attach-acr <acrName (not fully qualified)>
* Ensure kubectl is installed on your local computer. You can use: az aks install-cli
* Connect to cluster using kubectl: az aks get-credentials --resource-group myResourceGroup --name myAKSCluster
* Verify the connection to the cluster, run the [kubectl get nodes](https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#get) command: $ kubectl get nodes

[Run your container images in Kubernetes](https://docs.microsoft.com/en-us/azure/aks/tutorial-kubernetes-deploy-application)

* Update the manifest file - An Azure Container Registry (ACR) instance stores the container image for the application. To deploy the application, you **must update the image name in the Kubernetes manifest file** to include the ACR login server name.
* Get the ACR login server name using the [az acr list](https://docs.microsoft.com/en-us/cli/azure/acr) command: az acr list --resource-group myResourceGroup --query "[].{acrLoginServer:loginServer}" --output table
* Edit/Create the manifest (.YAML) file. Replace *microsoft* with your ACR login server name, e.g., *dcheemaacr1*. FYI, line 51 in the .YAML file.
* Deploy the application.
  + To deploy your application, use the [kubectl apply](https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#apply) command: kubectl apply -f azure-vote-all-in-one-redis.yaml
* Test the application
  + When the application runs, a Kubernetes service exposes the application (app name) to the internet. This process can take a few minutes to complete.
  + To monitor progress, use the [kubectl get service](https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#get) command: kubectl get service azure-vote-front --watch

[Scale an application and Kubernetes infrastructure](https://docs.microsoft.com/en-us/azure/aks/tutorial-kubernetes-scale)

[Update an application running in Kubernetes](https://docs.microsoft.com/en-us/azure/aks/tutorial-kubernetes-app-update)

[Upgrade AKS cluster](https://docs.microsoft.com/en-us/azure/aks/tutorial-kubernetes-upgrade-cluster)