
App Containerization & Kubernetes Deployment

Demo - App Containerization & Kubernetes
Deployment

Document Control

Document Version History

The Document Version History information is required and must follow the Best Practice Version Control Standards.

This table shows a record of significant changes to the document.

Version	Date	Author	Description of Change
1.0	21/01/2020	Achal Sharma	Initial Draft

1	Contents	
2	Introduction.....	4
3	Pre-Requisites	4
4	Containerize an application	5
5	Create Azure Container Registry & Push Container Image	9
6	Create Kubernetes Cluster & use ACR Container Image	13
7	'deplyment.yaml' File	17
8	Kubernetes Dashboard & deleting resources	18
9	General errors and their solutions.....	19
9.1	Docker fails to start	19
9.1.1	System Restart	19
9.1.2	Tweaking vmcompute.exe	19

2 Introduction

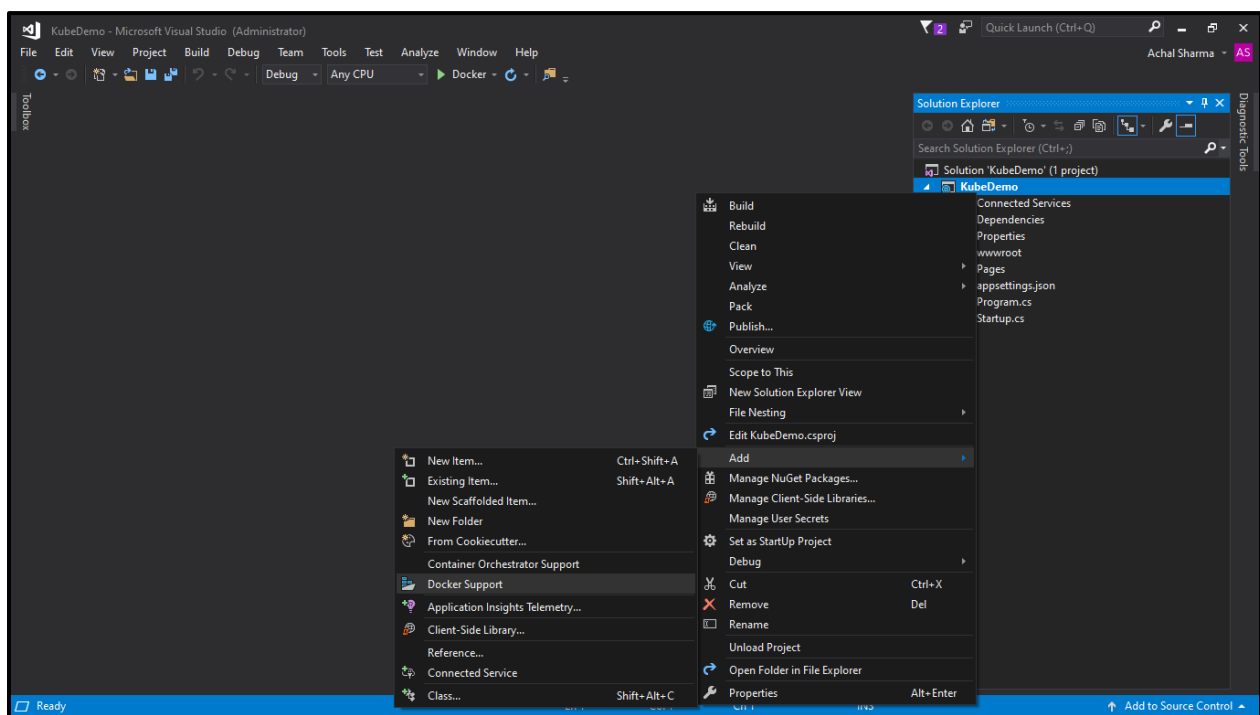
This documentation will guide you through on how to containerize an application and then deploy it on Kubernetes. Make sure to follow all the steps carefully or they might throw errors. Some general errors have been documented at the end of this guide.

3 Pre-Requisites

- Make sure Virtualization is enabled in BIOS.
- An azure subscription with sufficient user access to create and update resources.
- **Chocolatey**
 - Install it by using the following commands in an **elevated PowerShell**.
 - ❖ `Set-ExecutionPolicy AllSigned`
 - ❖ `Set-ExecutionPolicy Bypass -Scope Process -Force; iex ((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))`
 - Input "[A] – Yes to All" wherever prompted.
- **Kubernetes-CLI**
 - Install it by using the following command in an **elevated CMD**:
 - ❖ `choco install kubernetes-cli`
- **Azure-CLI**
 - Use the following URL to download and install Azure CLI:
 - ❖ <https://docs.microsoft.com/en-us/cli/azure/install-azure-cli-windows?view=azure-cli-latest>
- **Docker Desktop**
 - Use the following URL to download and install Docker Desktop:
 - ❖ <https://download.docker.com/win/stable/Docker%20Desktop%20Installer.exe>
 - Remember to uncheck "Use windows Containers" during Installation.
 - If you have Docker Desktop pre-installed, make sure to switch it to Linux Containers.

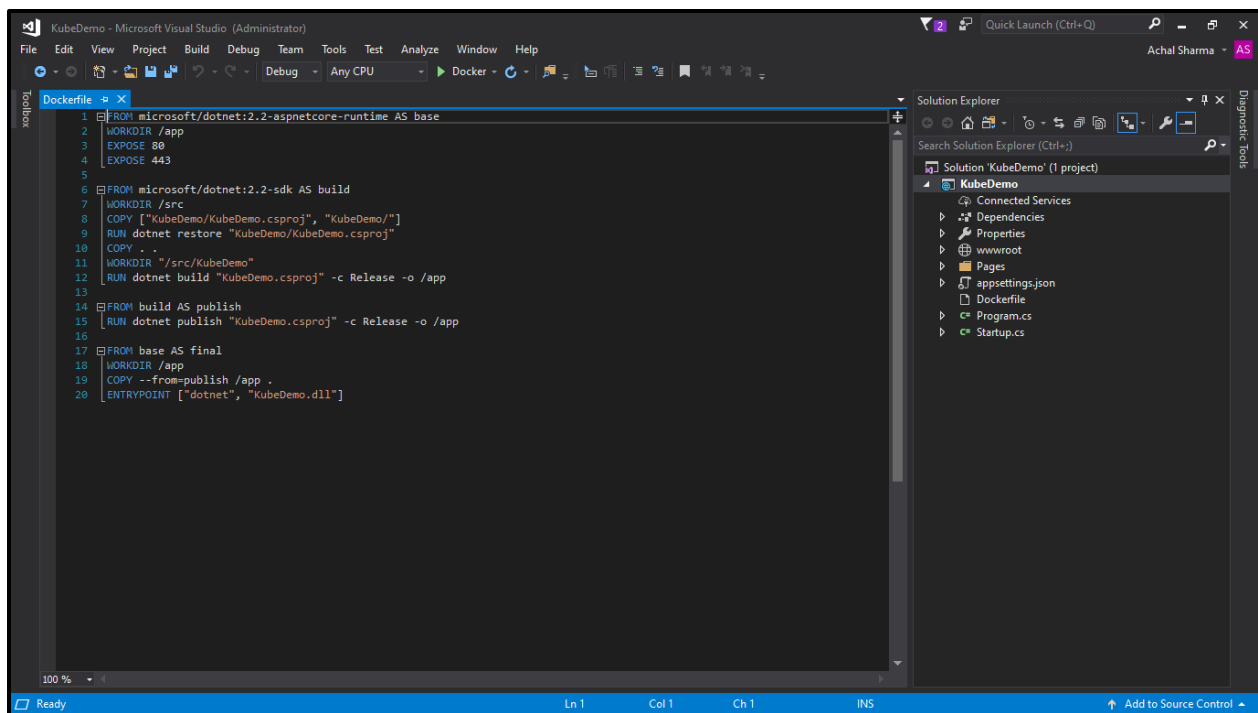
4 Containerize an application

1. For this walkthrough, I have created a standard .NET Core Web App from the Visual Studio Templates named "KubeDemo".
2. Generate a Dockerfile for the project, by right-clicking the project in the Visual Studio's Solution Explorer > Add > Docker Support. [Fig 4.1]



[Fig. 4.1]

3. Make sure the generated Dockerfile is in the correct Directory, this can be corroborated by checking the path of the project inside Dockerfile. If Dockerfile is not in the correct Directory, either move it to the correct directory, or change the paths referenced inside accordingly. [Fig. 4.2]



[Fig. 4.2] – Generated Dockerfile

4. Confirm that Docker Desktop is up and running from the System Tray or by executing **"docker"** in CMD terminal.
5. Open CMD Terminal in the same directory as Dockerfile and run the following command: [Fig. 4.3]

"docker build -t kubedemo ."

- This builds a docker image with all the dependencies.
- The flag **'-t'** tags the docker image as 'kubedemo' so that when we refer to the image, we don't need to refer it by its Image ID.

```
C:\Windows\System32\cmd.exe
--> Running in 58c1394e4677
Microsoft (R) Build Engine version 16.0.450+ga8dc7fd34 for .NET Core
Copyright (C) Microsoft Corporation. All rights reserved.

Restore completed in 132.24 ms for /src/KubeDemo/KubeDemo.csproj.
KubeDemo -> /app/KubeDemo.dll
KubeDemo -> /app/KubeDemo.Views.dll

Build succeeded.
0 Warning(s)
0 Error(s)

Time Elapsed 00:00:06.94
Removing intermediate container 58c1394e4677
--> 21b7fbbabe03
Step 12/17 : FROM build AS publish
--> 21b7fbbabe03
Step 13/17 : RUN dotnet publish "KubeDemo.csproj" -c Release -o /app
--> Running in 0e1eea4d2929
Microsoft (R) Build Engine version 16.0.450+ga8dc7fd34 for .NET Core
Copyright (C) Microsoft Corporation. All rights reserved.

Restore completed in 259.08 ms for /src/KubeDemo/KubeDemo.csproj.
KubeDemo -> /src/KubeDemo/bin/Release/netcoreapp2.2/KubeDemo.dll
KubeDemo -> /src/KubeDemo/bin/Release/netcoreapp2.2/KubeDemo.Views.dll
KubeDemo -> /app/
Removing intermediate container 0e1eea4d2929
--> 90f171c892fe
Step 14/17 : FROM base AS final
--> 422cdd341c8
Step 15/17 : WORKDIR /app
--> Running in 8c9aa11b2596
Removing intermediate container 8c9aa11b2596
--> bf719f993389
Step 16/17 : COPY --from=publish /app .
--> 2e1950c5bfb3
Step 17/17 : ENTRYPOINT ["dotnet", "KubeDemo.dll"]
--> Running in 6aa2f476d963
Removing intermediate container 6aa2f476d963
--> 07ca6206e5f3
Successfully built 07ca6206e5f3
Successfully tagged kubedemo:latest
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows Docker host. All files and directories added to build context will have '-rwxr-xr-x'
permissions. It is recommended to double check and reset permissions for sensitive files and directories.

C:\Users\achal.sharma\Visual Studio Projects\KubeDemo>
```

[Fig. 4.3] – Running the 'docker build' command

6. Test run the docker image by running the following command:

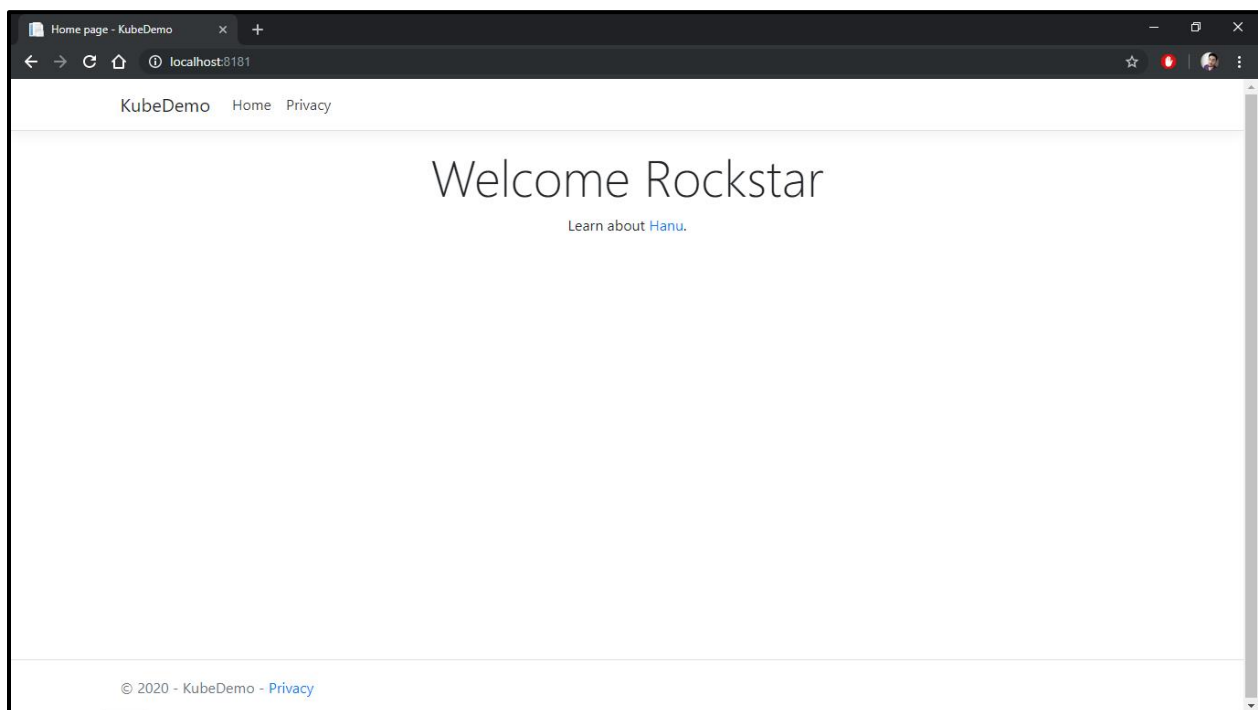
"docker run -p 8181:80 kubedemo"

- This will create and run a container locally. [Fig. 4.4]
- The flag '-p' specifies the port on which the container can be listened.
- The app can be traversed through: <http://localhost:8181> [Fig. 4.5]
- Use Ctrl+C to shut down the application from terminal.

```
C:\Windows\System32\cmd.exe - docker run -p 8181:80 kubedemo

C:\Users\achal.sharma\Visual Studio Projects\KubeDemo>docker run -p 8181:80 kubedemo
warn: Microsoft.AspNetCore.DataProtection.KeyManagement.XmlKeyManager[35]
      No XML encryptor configured. Key {b1ef11be-e95f-4199-9b20-74838f5e5250} may be persisted to storage in unencrypted form.
Hosting environment: Production
Content root path: /app
Now listening on: http://[::]:80
Application started. Press Ctrl+C to shut down.
```

[Fig. 4.4] – Executing the 'docker run' command



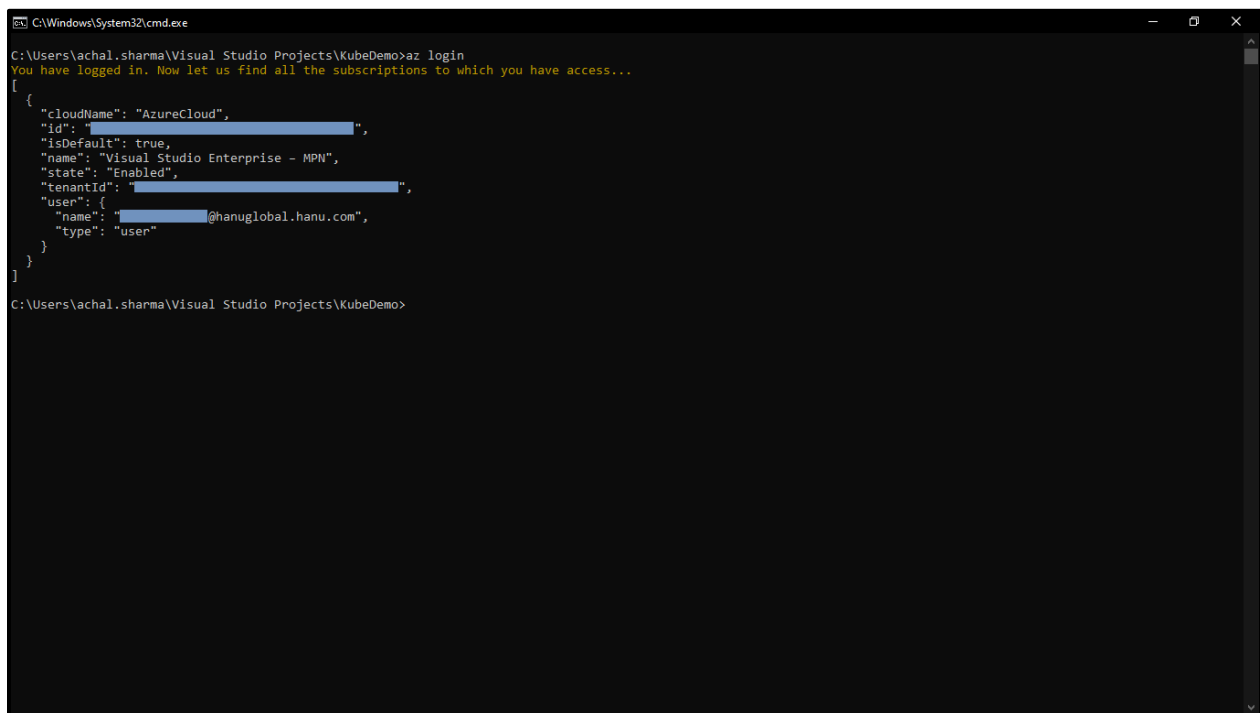
[Fig. 4.5] – Accessing the app on localhost

5 Create Azure Container Registry & Push Container Image

1. Open CMD terminal.
2. First of all, run the following command to log in to Azure in the terminal:

“az login”

This will redirect to webpage in your default browser, select your Microsoft Azure account and click next, and wait for login to complete in the terminal. [Fig. 5.1]



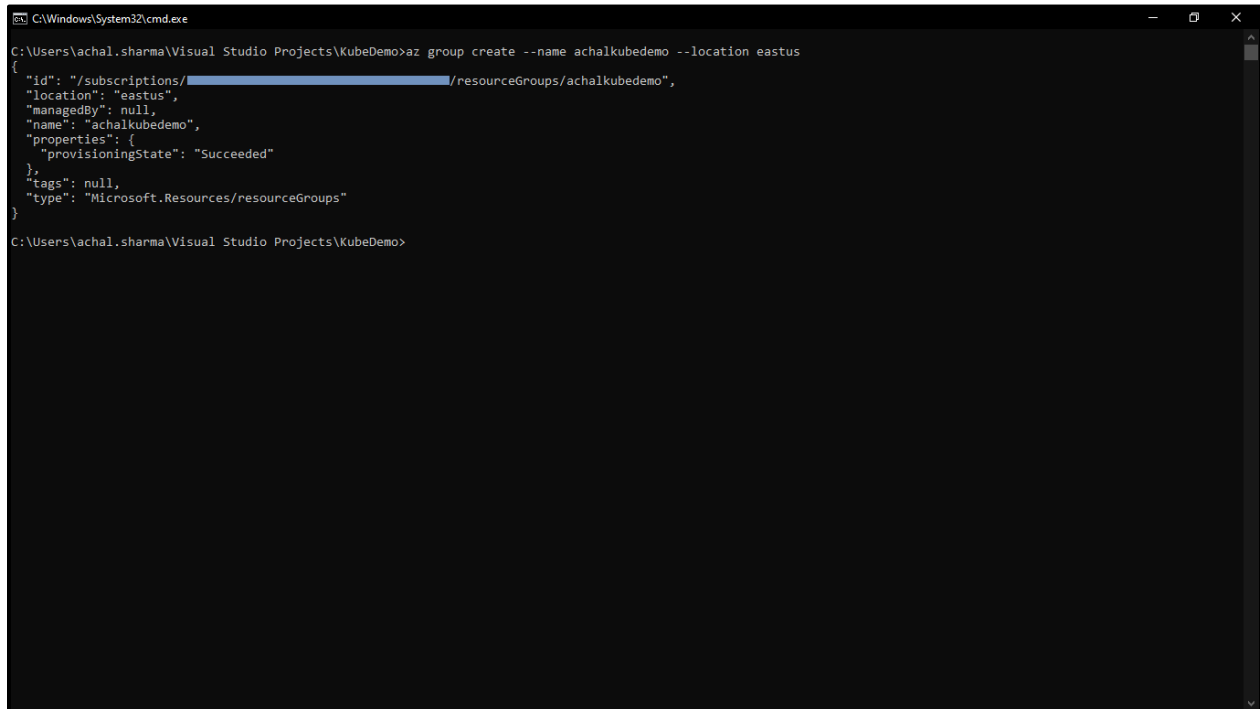
```
C:\Windows\System32\cmd.exe
C:\Users\achal.sharma\Visual Studio Projects\KubeDemo>az login
You have logged in. Now let us find all the subscriptions to which you have access...
[
  {
    "cloudName": "AzureCloud",
    "id": "[REDACTED]",
    "isDefault": true,
    "name": "Visual Studio Enterprise - MPN",
    "state": "Enabled",
    "tenantId": "[REDACTED]",
    "user": {
      "name": "[REDACTED]@hanuglobal.hanu.com",
      "type": "user"
    }
  }
]
C:\Users\achal.sharma\Visual Studio Projects\KubeDemo>
```

[Fig. 5.1]

-
- Next, we will create a new Resource Group, so that everything we make from now onwards can be encapsulated under a single Resource Group.

Execute the following command to create a Resource Group named 'achalkubedemo' in East US: [Fig. 5.2]

"az group create --name achalkubedemo --location eastus"



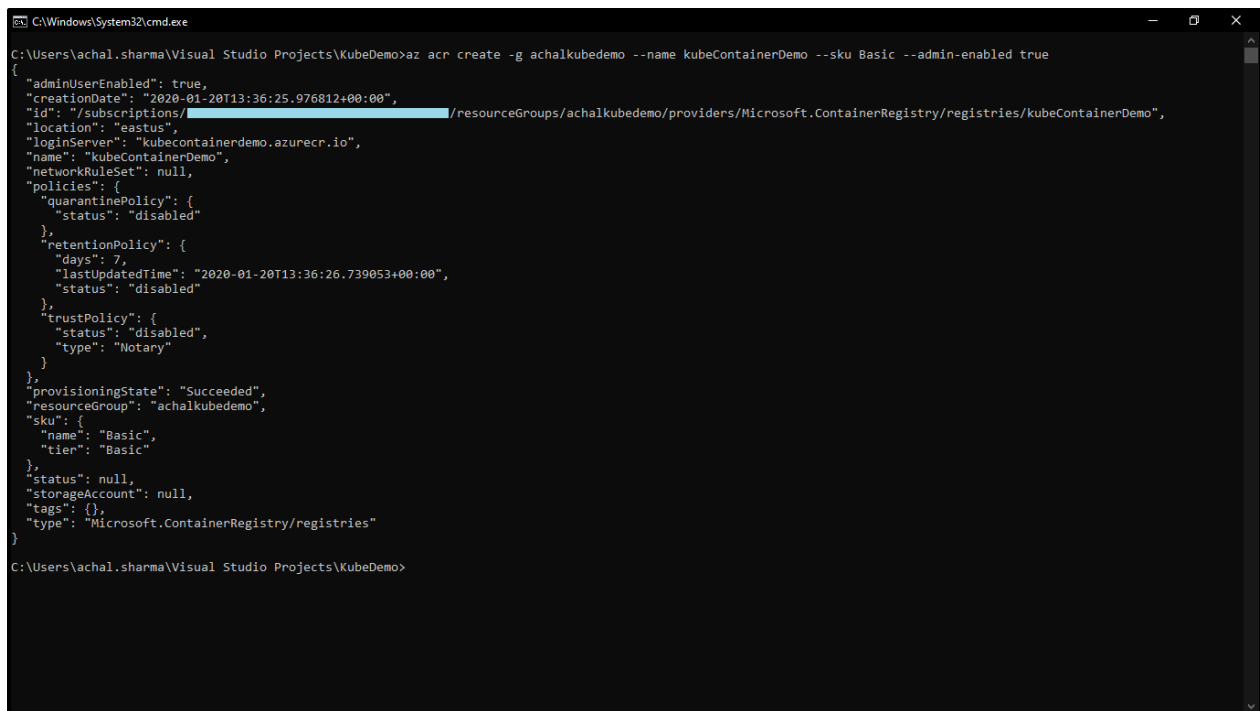
```
C:\Windows\System32\cmd.exe
C:\Users\achal.sharma\Visual Studio Projects\KubeDemo>az group create --name achalkubedemo --location eastus
{
  "id": "/subscriptions/[REDACTED]/resourceGroups/achalkubedemo",
  "location": "eastus",
  "managedBy": null,
  "name": "achalkubedemo",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null,
  "type": "Microsoft.Resources/resourceGroups"
}
C:\Users\achal.sharma\Visual Studio Projects\KubeDemo>
```

[Fig. 5.2]

-
- Now, we will be creating an Azure Container Registry (ACR) where we will push the docker image later.

Executing the following command will create an ACR named 'kubeContainerDemo' on the previously created Resource Group: [Fig. 5.3]

"az acr create -g achalkubedemo --name kubeContainerDemo --sku Basic --admin-enabled true"



```
C:\Windows\System32\cmd.exe
C:\Users\achal.sharma\Visual Studio Projects\KubeDemo>az acr create -g achalkubedemo --name kubeContainerDemo --sku Basic --admin-enabled true
{
  "adminUserEnabled": true,
  "creationDate": "2020-01-20T13:36:25.976812+00:00",
  "id": "/subscriptions/.../resourceGroups/achalkubedemo/providers/Microsoft.ContainerRegistry/registries/kubeContainerDemo",
  "location": "eastus",
  "loginServer": "kubecontainerdemo.azurecr.io",
  "name": "kubeContainerDemo",
  "networkRuleSet": null,
  "policies": {
    "quarantinePolicy": {
      "status": "disabled"
    },
    "retentionPolicy": {
      "days": 7,
      "lastUpdatedTime": "2020-01-20T13:36:26.739053+00:00",
      "status": "disabled"
    },
    "trustPolicy": {
      "status": "disabled",
      "type": "Notary"
    }
  },
  "provisioningState": "Succeeded",
  "resourceGroup": "achalkubedemo",
  "sku": {
    "name": "Basic",
    "tier": "Basic"
  },
  "status": null,
  "storageAccount": null,
  "tags": {},
  "type": "Microsoft.ContainerRegistry/registries"
}
```

[Fig. 5.3]

- Get the credentials of ACR by running the following command and copy the password that gets displayed and save it for a later use in a command.

"az acr credential show -n kubeContainerDemo"

- Execute the following command to tag the existing docker container so that we can differentiate it from other containers:

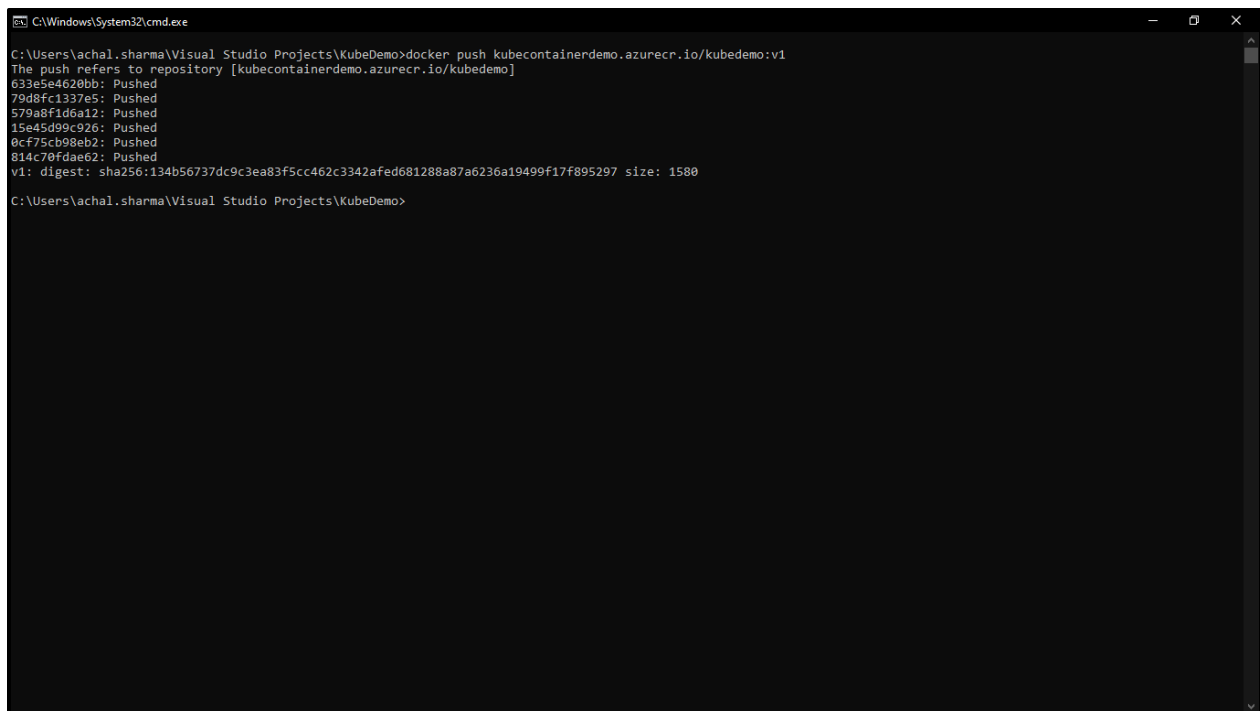
"docker tag kubedemo kubecontainerdemo.azurecr.io/kubedemo:v1"

-
- Log in to ACR through Docker by executing the following command. Replace the string after '-p' flag with the password that you fetched from Step 5:

```
"docker login https://kubernetesdemo.azurecr.io -u kubeContainerDemo -p 2SXYUUipFBu+YxcjWbWLNof7wnBtGwWM"
```

- Push the container to your ACR and store it as a Container Image by running the following command: [Fig. 5.4]

```
"docker push kubecontainerdemo.azurecr.io/kubedemo:v1"
```



```
C:\Windows\System32\cmd.exe
C:\Users\achal.sharma\Visual Studio Projects\KubeDemo>docker push kubecontainerdemo.azurecr.io/kubedemo:v1
The push refers to repository [kubecontainerdemo.azurecr.io/kubedemo]
633e5e4628bb: Pushed
79d8fc1337e5: Pushed
579a8f1d6a12: Pushed
15e45d99c926: Pushed
0cf75cb98eb2: Pushed
814c70fdae62: Pushed
v1: digest: sha256:134b56737dc9c3ea83f5cc462c3342afed681288a87a6236a19499f17f895297 size: 1580
C:\Users\achal.sharma\Visual Studio Projects\KubeDemo>
```

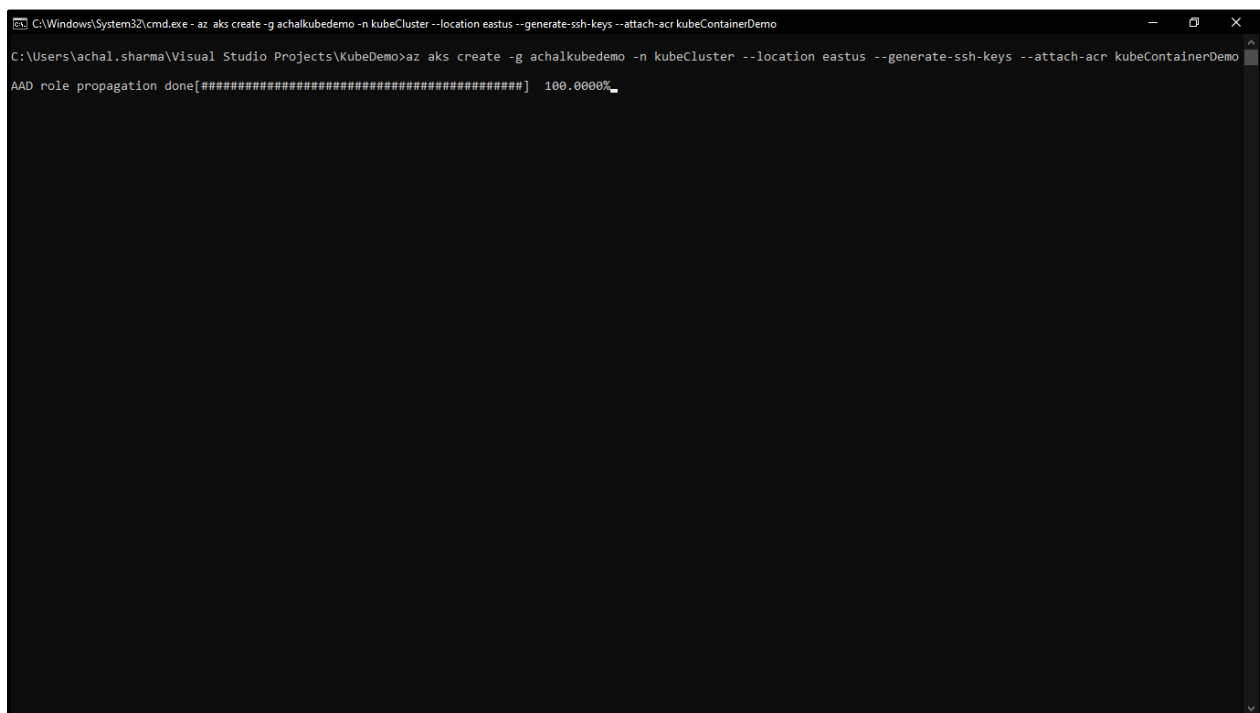
[Fig. 5.4]

6 Create Kubernetes Cluster & use ACR Container Image

1. Run the following command in terminal to create an Azure Kubernetes Services (AKS) Cluster named '**kubeCluster**' in East US and attach it to ACR created in previous module with the flag '**--attach-acr**': [Fig. 6.1] & [Fig. 6.2]

"az aks create -g achalkubedemo -n kubeCluster --location eastus --generate-ssh-keys --attach-acr kubeContainerDemo"

*Cluster creation takes 15 - 20 minutes.



```
C:\Windows\System32\cmd.exe - az aks create -g achalkubedemo -n kubeCluster --location eastus --generate-ssh-keys --attach-acr kubeContainerDemo
C:\Users\achal.sharma\Visual Studio Projects\KubeDemo>az aks create -g achalkubedemo -n kubeCluster --location eastus --generate-ssh-keys --attach-acr kubeContainerDemo
AAD role propagation done[#####] 100.0000%
```

[Fig 6.1]

```
C:\Windows\System32\cmd.exe
{
  "location": "eastus",
  "maxAgentPools": 8,
  "name": "kubeCluster",
  "networkProfile": {
    "dnsServiceIp": "10.0.0.10",
    "dockerBridgeCidr": "172.17.0.1/16",
    "loadBalancerProfile": {
      "effectiveOutboundIps": [
        {
          "id": "/subscriptions/[REDACTED]/resourceGroups/MC_achalkubedemo_kubeCluster_eastus/providers/Microsoft.Network/publicIPAddresses/[REDACTED]",
          "resourceGroup": "MC_achalkubedemo_kubeCluster_eastus"
        }
      ],
      "managedOutboundIps": {
        "count": 1
      },
      "outboundIpPrefixes": null,
      "outboundIps": null
    },
    "loadBalancerSku": "Standard",
    "networkPlugin": "kubenet",
    "networkPolicy": null,
    "outboundType": "loadBalancer",
    "podCidr": "10.244.0.0/16",
    "serviceCidr": "10.0.0.0/16"
  },
  "nodeResourceGroup": "MC_achalkubedemo_kubeCluster_eastus",
  "privateFqdn": null,
  "provisioningState": "Succeeded",
  "resourceGroup": "achalkubedemo",
  "servicePrincipalProfile": {
    "clientId": "[REDACTED]",
    "secret": null
  },
  "tags": null,
  "type": "Microsoft.ContainerService/ManagedClusters",
  "windowsProfile": null
}

C:\Users\achal.sharma\Visual Studio Projects\KubeDemo>
```

[Fig. 6.2]

2. To use the kubectl commands, we need to get the cluster's credentials and store them in the system. Execute the following command to do the same: [Fig. 6.3]

"az aks get-credentials --resource-group achalkubedemo --name kubeCluster"

3. To deploy the app to the Kubernetes cluster from ACR, we will use the following command, which uses the file '**deployment.yaml**' as Deployment Configuration. Make sure the directory in which you are going to run the command also contains the 'deployment.yaml' file: [Fig. 6.3]

"kubectl apply -f deployment.yaml"

*The contents of '**deployment.yaml**' file can be found in the next module.

```
C:\Windows\System32\cmd.exe

C:\Users\achal.sharma\Visual Studio Projects\KubeDemo>cd ../../

C:\Users\achal.sharma>az aks get-credentials --resource-group achalkubedemo --name kubeCluster
A different object named kubeCluster already exists in your kubeconfig file.
Overwrite? (y/n): y
A different object named clusterUser_achalkubedemo_kubeCluster already exists in your kubeconfig file.
Overwrite? (y/n): y
Merged "kubeCluster" as current context in C:\Users\achal.sharma\.kube\config

C:\Users\achal.sharma>kubectl apply -f deployment.yaml
deployment.apps/hello created
service/hello created

C:\Users\achal.sharma>
```

[Fig. 6.3]

4. Use the following command to get services running in the cluster, select and copy the external endpoint of the app which we just deployed, in this case, the name of the service is '**hello**': [Fig. 6.4]

"**kubectl get svc**"

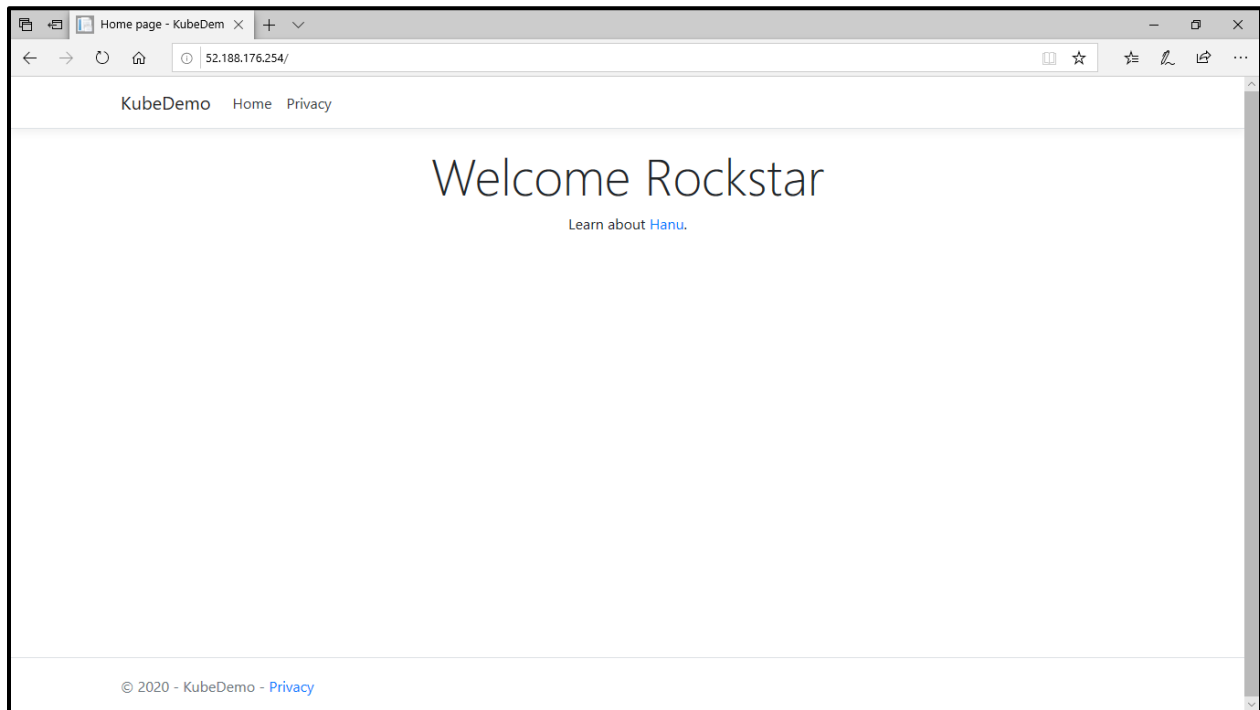
```
C:\Windows\System32\cmd.exe

C:\Users\achal.sharma>kubectl get svc
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
hello         LoadBalancer  10.0.181.106  52.188.176.254  80:31454/TCP     3m22s
kubernetes    ClusterIP     10.0.0.1      <none>         443/TCP          9m7s

C:\Users\achal.sharma>
```

[Fig. 6.4]

-
5. Paste the external endpoint in the browser to visit the app. The app is running successfully in a container on Kubernetes Cluster: [Fig. 6.5]



[Fig. 6.5]

7 'deployment.yaml' File

```
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: hello
spec:
  replicas: 1
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1
  minReadySeconds: 5
  template:
    metadata:
      labels:
        app: hello
    spec:
      nodeSelector:
        "beta.kubernetes.io/os": linux
      containers:
        - name: hello
          image: kubecontainerdemo.azurecr.io/kubedemo:v1
          ports:
            - containerPort: 80
          resources:
            requests:
              cpu: 250m
            limits:
              cpu: 500m
---
apiVersion: v1
kind: Service
metadata:
  name: hello
spec:
  type: LoadBalancer
  ports:
    - port: 80
  selector:
    app: hello
```

8 Kubernetes Dashboard & deleting resources

- If you wish to browse the Kubernetes Dashboard, run the following two commands only after you have fetched AKS credentials at least once from the corresponding azure account:

```
"kubectl create clusterrolebinding kubernetes-dashboard --  
clusterrole=cluster-admin --serviceaccount=kube-system:kubernetes-  
dashboard"
```

```
"az aks browse --resource-group achalkubedemo --name kubeCluster"
```

- Deleting the resource group from Azure will delete every resource created in this demo. Make sure delete the associated Resource Group only. Run the following command to do the same, it may take up to 15 – 20 minutes :

```
"az group delete -n achalkubedemo -y"
```

9 General errors and their solutions

9.1 Docker fails to start

If Docker fails to start or times out (Daemon) try the following workarounds:

9.1.1 System Restart

Generally, this issue can be solved by doing a simple reboot of the system. If this does not solve the problem, move on to the next section.

9.1.2 Tweaking vmcompute.exe

1. Open "Window Security" by searching in Start Menu.
2. Open "App & Browser control"
3. Click "Exploit protection settings" at the bottom
4. Switch to "Program settings" tab
5. Locate "C:\WINDOWS\System32\vmcompute.exe" in the list and expand it
6. Click "Edit"
7. Scroll down to "Code flow guard (CFG)" and uncheck "Override system settings"
8. Start vmcompute.exe from PowerShell **"net start vmcompute"**