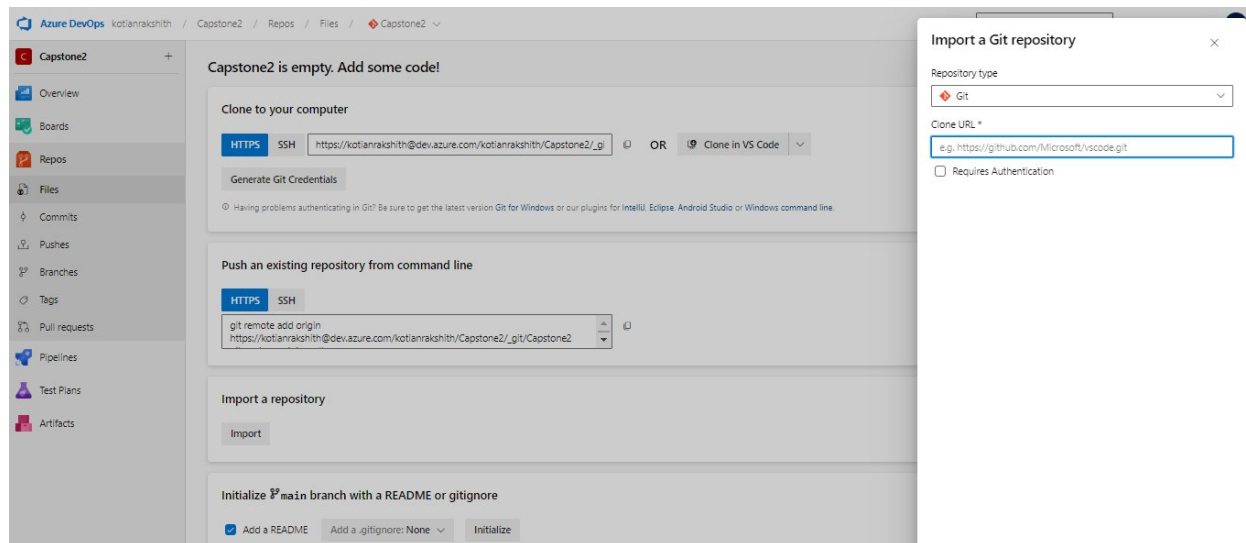


Deployment of Java application through Azure DevOps Pipeline, Azure Repos, ACR, Docker in VM

We have deployed insurance application in capstone project2 using jenkins, github, dockerhub,docker in AWS EC2 Machine, we will take the same application and deploy through Azure Devops, Azure Repos, Azure container registry, docker in VM machine. I have done the below steps to complete this project:

1. Import the resository:

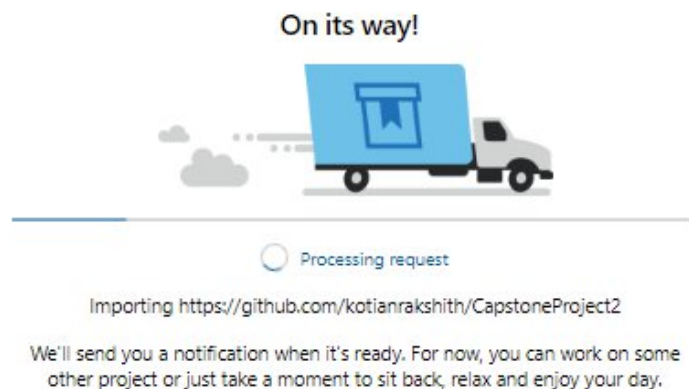
We will create a new project in Azure DevOps and then go to Repo and click on Import a Repository:



We have already created github repo for the capstone2 project:

<https://github.com/kotianrakshith/CapstoneProject2>

We will import this:



We can see after the import all the files are imported:

Capstone2

src

.gitignore

ansible-playbook.yml

Dockerfile

Jenkinsfile

mvnw

mvnw.cmd

pom.xml

README.md

main

Type to find a file or folder...

Files

Contents History

Name ↑	Last change	Commits
src	Jul 6	46e1fe88 Initial commit message Rakshith Kotian
.gitignore	Jul 6	46e1fe88 Initial commit message Rakshith Kotian
ansible-playbook.yml	Jul 8	5a7f543b Update ansible-playbook.yml Kotian Rakshith
Dockerfile	Jul 7	dafa4a9c Update Dockerfile Kotian Rakshith
Jenkinsfile	Jul 8	984569d7 Update Jenkinsfile changed names Kotian Rakshith
mvnw	Jul 6	46e1fe88 Initial commit message Rakshith Kotian
mvnw.cmd	Jul 6	46e1fe88 Initial commit message Rakshith Kotian
pom.xml	Jul 6	46e1fe88 Initial commit message Rakshith Kotian
README.md	Jul 8	06dd2b31 Update README.md Kotian Rakshith

Insurance company Application Deployment using Jenkins, Ansible, Docker and AWS EC2 instance
source code for the java application is provided by the simplilearn team. The Jenkinsfile, Ansible playbook, Dockerfile was built by me.

(Later we will delete all the files not needed ex. Jenkinsfile,ansible playbook)

2. Build the java application

We will go to pipeline and create a new pipeline:

Create your first Pipeline

Automate your build and release processes using our wizard, and go from code to cloud-hosted within minutes.

Create Pipeline

Then select your repo and the code:

Connect

Select

Configure

Review

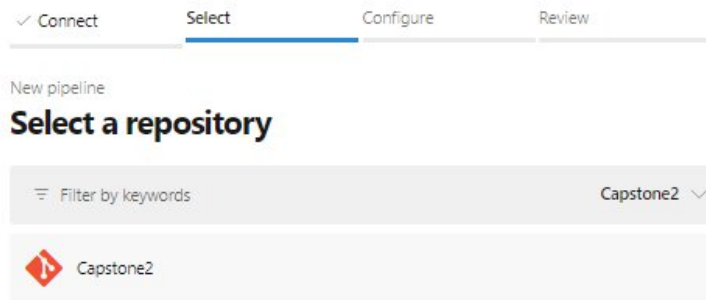
New pipeline

Where is your code?

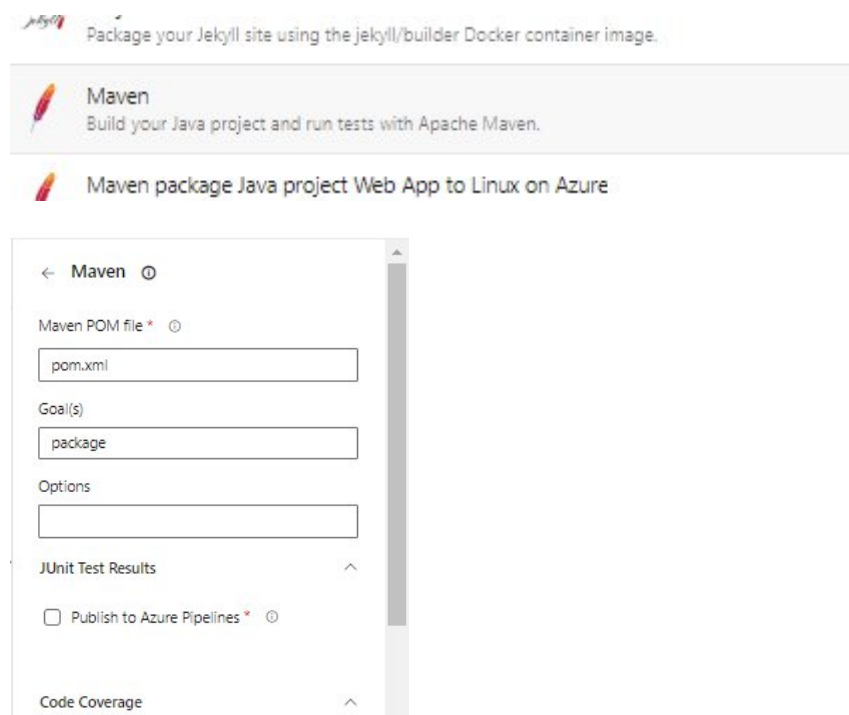
Azure Repos Git

YAML

Free private Git repositories, pull requests, and code search



Then we will chose maven and only chose package:



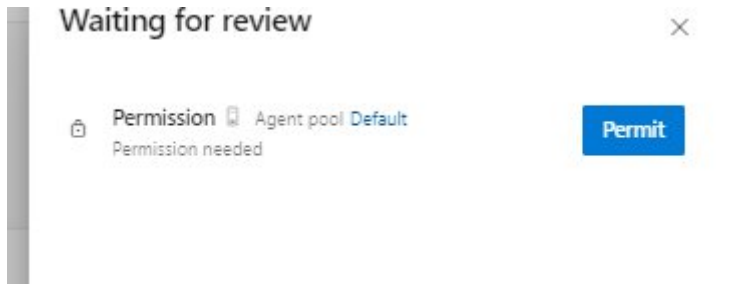
So our code till the build stage will be as below: (I have used self hosted pool instead of microsoft as Im not yet approved for it)

```
Capstone2 / azure-pipelines.yml *
2  # Build your Java project and run tests with Apache Maven.
3  # Add steps that analyze code, save build artifacts, deploy, and more:
4  # https://docs.microsoft.com/azure/devops/pipelines/languages/java
5
6  trigger:
7  - main
8
9  pool:
10 - name: Default
11
12 steps:
13   - task: Maven@4
14     inputs:
15       mavenPomFile: 'pom.xml'
16       publishJUnitResults: false
17       javaHomeOption: 'JDKVersion'
18       mavenVersionOption: 'Default'
19       mavenAuthenticateFeed: false
20       effectivePomSkip: false
21       sonarQubeRunAnalysis: false
22
```

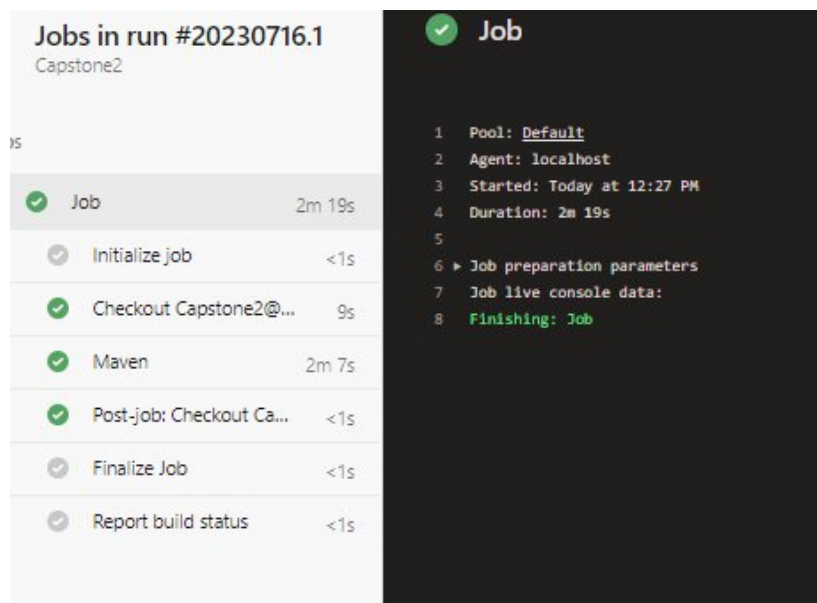
Now lets save and run:



As Im using self hosted agent it will ask for permission:



Once you permit, it will run and we can see that it has sucessfully run and maven has build package to be deployed.



3. Build and push the image to azure container registry

First let us create a container registry:

Go to container registries in azure portal and click create container registry:



Give all the details and click [review+create](#)

Project details

Subscription * Free Trial

Resource group * Practiceresource
[Create new](#)

Instance details

Registry name * kotianrakshith ✓
.azurecr.io

Location * East US

Availability zones ⓘ ☐ Enabled

Availability zones are enabled on premium registries and in regions that support availability zones. [Learn more](#)

SKU * ⓘ Basic

[Home](#) >

Microsoft.ContainerRegistry | Overview Deployment

Search « Delete Cancel Redeploy Download Refresh

Overview

Inputs

Outputs

Template

✓ Your deployment is complete

Deployment name : Microsoft.ContainerRegistry
Subscription : Free Trial
Resource group : Practiceresource

> Deployment details

Next steps

[Go to resource](#)

Give feedback

[Tell us about your experience with deployment](#)

Now we will go the pipeline and add the build and push to container registry steps:

Edit the pipeline and add the task **buildandpush** docker:

Docker
Build or push Docker images, login or logout, star...

Build or push Docker images, login or logout

Docker ⓘ

Container Repository ^

Container registry ⓘ kotianrakshith

Container repository ⓘ insuranceapp

Commands ^

Command * ⓘ buildAndPush

Dockerfile * ⓘ **/Dockerfile

Build context ⓘ **

Tags ⓘ

The code will be as below:

```

22 - task: Docker@2
23   inputs:
24     containerRegistry: 'kotianrakshith'
25     repository: 'insuranceapp'
26     command: 'buildAndPush'
27     Dockerfile: '**/Dockerfile'
28     tags:

```

We have made sure docker is running locally and we will save the pipeline and it should run successfully:

The screenshot displays the Azure DevOps interface for a job run. On the left, a table lists the steps of the job, all of which are completed with green checkmarks. On the right, a detailed view of the 'Job' step is shown, indicating it was executed on the 'Default' pool at 'localhost' and completed successfully.

Step	Duration
Initialize job	<1s
Checkout Capstone2@...	4s
Maven	30s
Docker	1m 6s
Post-job: Checkout Ca...	<1s
Finalize Job	<1s
Report build status	<1s

Job Details:

- 1 Pool: **Default**
- 2 Agent: **localhost**
- 3 Started: Today at 12:48 PM
- 4 Duration: 1m 42s
- 5
- 6 Job preparation parameters
- 7 Job live console data:
- 8 **Finishing: Job**

Now let us check the Azure Container Registry if the image has been stored or not.

Go to the registry and navigate to Repositories, you should be able to see newly create 'insuranceapp' image

The screenshot shows the 'Repositories' page in the Azure Container Registry. The 'insuranceapp' repository is listed under the 'kotianrakshith' namespace. The page includes a search bar, a 'Refresh' button, and a 'Manage Deleted Repositories' link.

Home > Microsoft.ContainerRegistry | Overview > kotianrakshith

kotianrakshith | Repositories

Container registry

Search

Refresh Manage Deleted Repositories

Search to filter repositories ...

Repositories ↑↓

insuranceapp

4. Create a virtual machine and install docker:

Now since we have to deploy the docker container in a virtual machine, we will create a virtual machine. Go to virtual machine in azure and click create new:

The screenshot shows the 'Create new' dropdown menu in the Azure portal. The menu lists four options for creating a virtual machine: 'Azure virtual machine', 'Azure virtual machine with preset configuration', 'Azure Arc virtual machine', and 'Azure VMware Solution virtual machine'. The 'Create' button is highlighted at the bottom of the menu.

Subscription ↑↓ Resource group ↑↓ Location ↑↓ Status ↑↓ Operating system 1

Create new

- Azure virtual machine
Create a virtual machine hosted by Azure
- Azure virtual machine with preset configuration
Create a virtual machine with presets based on your workloads
- Azure Arc virtual machine
Create a new Azure Arc virtual machine in one of your non-Azure environments
- Azure VMware Solution virtual machine
Create a VMware virtual machine hosted by Azure

Create

Learn more about Windows virtual machines

Learn more about Linux virtual machines

We will give sufficient details:

Home > Virtual machines >

Create a virtual machine

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ Free Trial

Resource group * ⓘ Practiceresource
[Create new](#)

Instance details

Virtual machine name * ⓘ capstone2 ✓

Region * ⓘ (US) East US

Availability options ⓘ No infrastructure redundancy required

Security type ⓘ Trusted launch virtual machines
[Configure security features](#)

Image * ⓘ Ubuntu Server 20.04 LTS - x64 Gen2 (free services eligible)
[See all images](#) | [Configure VM generation](#)

VM architecture ⓘ
☐ Arm64
☒ x64

Username * ⓘ rakshith ✓

SSH public key source Generate new key pair

Key pair name * capstone2_key ✓

Inbound port rules

Select which virtual machine network ports are accessible from the public internet. You can specify more limited or granular network access on the Networking tab.

Public inbound ports * ⓘ
☐ None
☒ Allow selected ports

Select inbound ports * SSH (22)

[i](#) All traffic from the internet will be blocked by default. You will be able to

Then we will review and create.

Once created, we will go to the networking and all inbound rule:

capstone2 | Networking ☆ ...

Virtual machine

Search

Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems

Settings
Networking
Connect
Disks
Size
Microsoft Defender for Cloud
Advisor recommendations
Extensions + applications
Availability + scaling

Feedback Attach network interface Detach network interface

capstone2412

IP configuration ⓘ
ipconfig1 (Primary)

Network Interface: capstone2412 Effective security rules Troubleshoot VM connection issues Topology
Virtual network/subnet: capstone2-vnet/default NIC Public IP: 20.115.123.210 NIC Private IP: 10.0.0.4 Accelerated networking: Disabled

Inbound port rules Outbound port rules Application security groups Load balancing


Network security group capstone2-nsg (attached to network interface: capstone2412)
Impacts 0 subnets, 1 network interfaces

Priority	Name	Port	Protocol	Source	Destination	Action	
300	SSH	22	TCP	Any	Any	Allow	...
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow	...
65001	AllowAzureLoadBalancerInBound	Any	Any	AzureLoadBalancer	Any	Allow	...
65500	DenyAllInBound	Any	Any	Any	Any	Deny	...

[Add inbound port rule](#)

<https://portal.azure.com/#@kotianrakshith:live.onmicrosoft.com/resource/subscrip...>

Then we will add inbound 8081 as this is the port we will be exposing our docker container application.

 **Add inbound security rule**
capstone2-nsg

Source ⓘ
Any

Source port ranges * ⓘ
*

Destination ⓘ
Any




Service ⓘ
Custom

Destination port ranges * ⓘ
8081

Protocol
☐ Any
☒ TCP
☐ UDP
☐ ICMP

Action

Now there should be SSH port and 8081 port :

Priority	Name	Port	Protocol	Source	Destination	Action	
300	 SSH	22	TCP	Any	Any	 Allow	***
310	AllowAnyCustom8081Inbound	8081	TCP	Any	Any	 Allow	***


Now lets connect to the VM and install docker.

In this case we are connecting through bastion as its safer. In the bastion, it may ask you to setup bastion, you can click on deploy bastion which automatically will set it up and add new subnet:

Azure Bastion protects your virtual machines by providing lightweight, browser-based connectivity without the need to expose them through public IP addresses. Deploying will automatically create a Bastion host on a subnet in your virtual network. [Learn more](#)

Create Bastion

Name ⓘ capstone2-vnet-bastion
Resource group ⓘ Practiceresource
Virtual network ⓘ capstone2-vnet
Public IP address ⓘ capstone2-vnet-ip

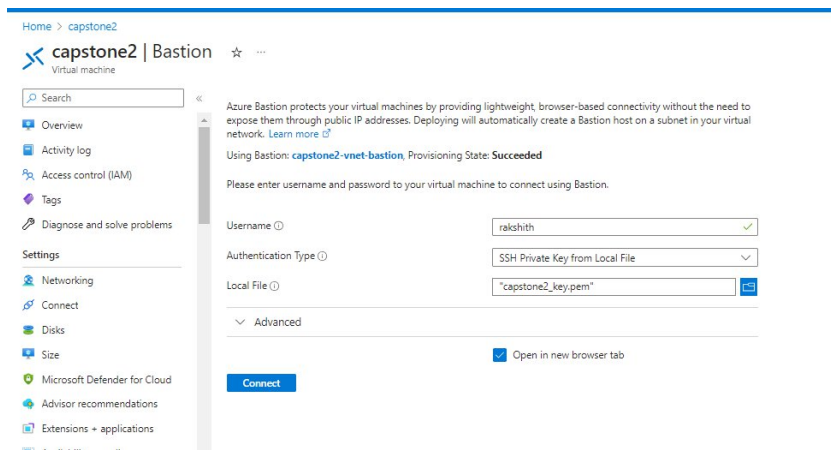
 Bastion pricing starts with an hourly base rate. [Learn more](#)

*** Creating a new Bastion 'capstone2-vnet-bastion'.

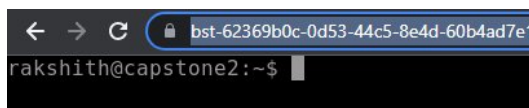
Deploy Bastion

Configure manually

Once deployed you can connect using your ssh key



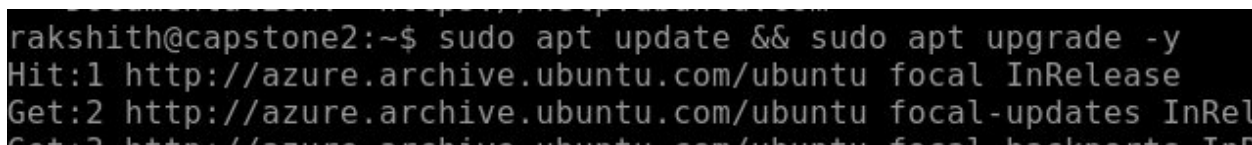
Your vm will open in the new browser tab:



Now let us install docker.

First we upgrade the system

```
sudo apt update && sudo apt upgrade -y
```



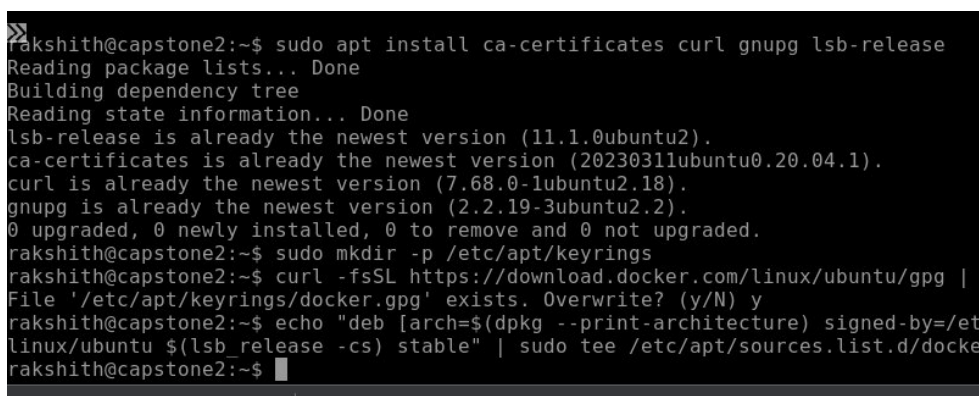
Now we install the packages, keys, required repositories:

```
sudo apt install ca-certificates curl gnupg lsb-release
```

```
sudo mkdir -p /etc/apt/keyrings
```

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
```

```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```



Now run the update command again(sudo apt update)

Now install the docker ce:

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin  
docker-compose-plugin
```

```
rakshith@capstone2:~$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following additional packages will be installed:  
  docker-ce-rootless-extras pigz slirn4netns
```

Now check the docker version:

```
docker -v
```

```
rakshith@capstone2:~$ docker -v  
Docker version 24.0.4, build 3713ee1  
rakshith@capstone2:~$
```

If you check the docker you see it is running:

```
rakshith@capstone2:~$ systemctl status docker  
● docker.service - Docker Application Container Engine  
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)  
   Active: active (running) since Sun 2023-07-16 15:29:35 UTC; 1min 24s ago  
 TriggeredBy: ● docker.socket  
    Docs: https://docs.docker.com  
   Main PID: 3587 (dockerd)  
      Tasks: 7  
     Memory: 35.7M
```

Then you can add your user for docker group(This may not be needed, but if you use different user it is helpful)

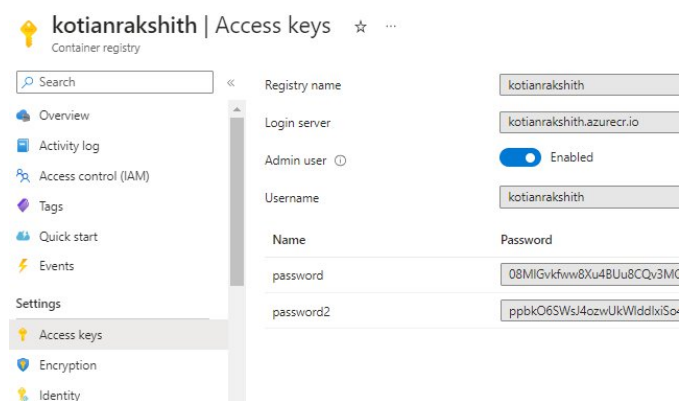
```
sudo usermod -aG docker $USER
```

Then reload the session using command: `newgrp docker`

```
rakshith@capstone2:~$ sudo usermod -aG docker $USER  
rakshith@capstone2:~$ newgrp docker  
rakshith@capstone2:~$
```

Now we will also do docker login to login to azure container registry:

First we will go to access keys and enable admin user which will give us username and password



We will use this to login to login server.

[docker login kotianrakshith.azurecr.io](#)

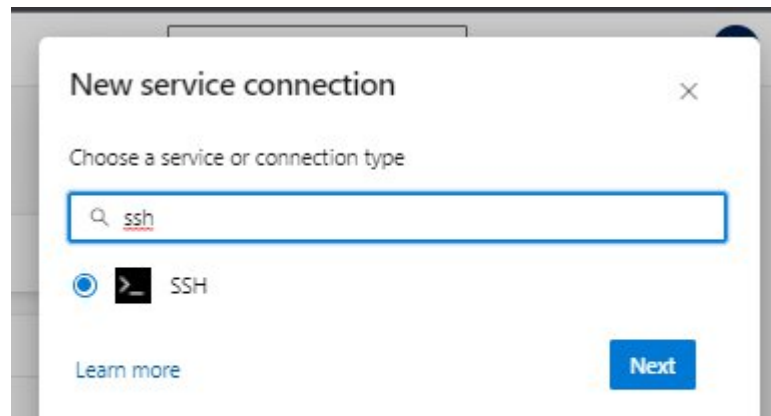
```
rkshith@capstone2:~$ docker login kotianrakshith.azurecr.io
rakshith@capstone2:~$ docker login kotianrakshith.azurecr.io
Username: kotianrakshith
Password:
WARNING! Your password will be stored unencrypted in /home/rakshith/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
rakshith@capstone2:~$
```

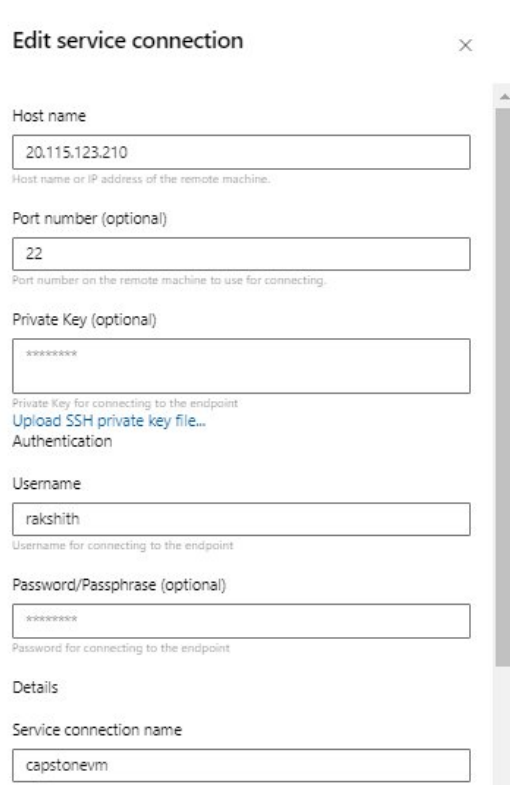
5. Add SSH service connection in Azure DevOps to the VM:

Go to the Project Settings-> Service connections-> click New service connection:

Search for SSH:



Give all the required details:



Edit service connection

Host name
20.115.123.210
Host name or IP address of the remote machine.

Port number (optional)
22
Port number on the remote machine to use for connecting.

Private Key (optional)
[Masked]
Private Key for connecting to the endpoint
[Upload SSH private key file...](#)

Authentication

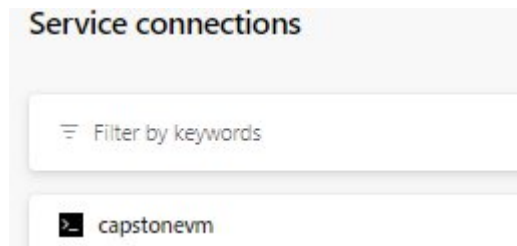
Username
rakshith
Username for connecting to the endpoint

Password/Passphrase (optional)
[Masked]
Password for connecting to the endpoint

Details

Service connection name
capstonevm

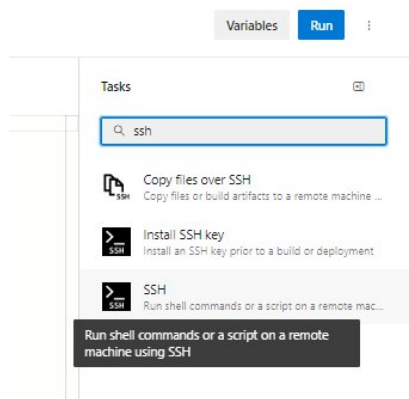
Then save and it should be added:



6. Update pipeline to deploy docker container in VM:

We will deploy docker container through docker commands given SSH connections to the VM.

Go to the pipeline and edit the pipeline. Search for SSH:



We will select the service connection we added and add following command as a script:

```
docker stop insuranceapp 2>/dev/null
```

```
docker rm insuranceapp 2>/dev/null
```

```
docker run -itd -p 8081:8081 --name insuranceapp  
kotianrakshith.azurecr.io/insuranceapp
```

(We need stop and rm if we want to run this pipeline repeatedly as once run there will be container present and we are using 2>/dev/null so it can ignore the errors)



Now lets save and it will run:

The screenshot shows the 'Jobs in run #20230716.7' for 'Capstone2'. A table lists the jobs: 'Job' (1m 38s), 'Initialize job' (<1s), 'Checkout Capstone2@...' (3s), 'Maven' (43s), 'Docker' (39s), 'SSH' (9s), 'Post-job: Checkout Ca...' (<1s), 'Finalize Job' (<1s), and 'Report build status' (<1s). The 'Job' is marked as successful. To the right, a log shows the job details: Pool: Default, Queued: Today at 11:16 PM, Agent: localhost, Started: Today at 11:16 PM, Duration: 1m 38s. The log indicates the job is already running or has completed, and shows the starting and finishing steps.

We see that it has run successfully.

7. Check if app is deployed successfully:

First lets go and check the VM if the app is deployed in docker:

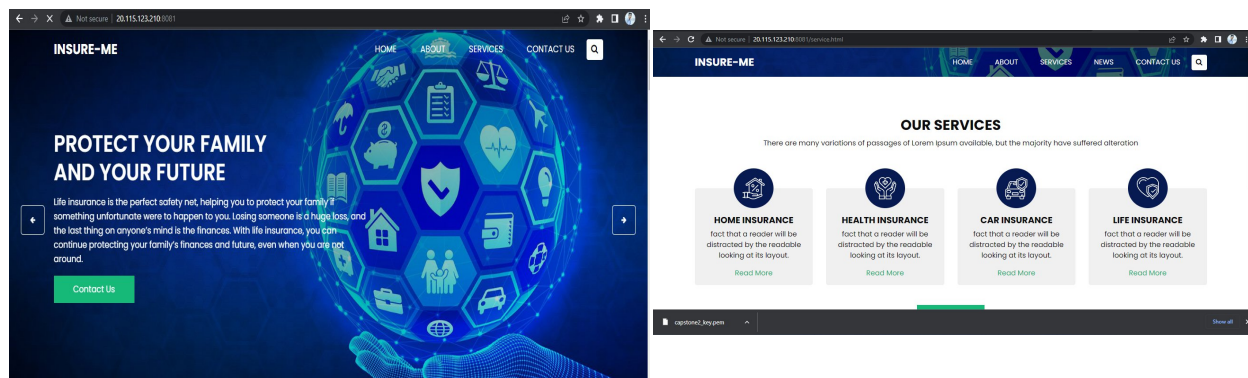
```
rakshith@capstone2:~$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
d7ed97700c3e   kotianrakshith.azurecr.io/insuranceapp  "java -jar /app.jar"    5 minutes ago  Up 5 minutes  0.0.0.0:8081->8081/tcp, :
:8081->8081/tcp  insuranceapp
rakshith@capstone2:~$
```

We can see from 'docker ps' command that it has been created and running correctly.

Now let us access the app and check. To access it we need to use ip:port so here it is

<http://20.115.123.210:8081/>

We can see that we are able to access it correctly:



That completes this project. We have used the same code from capstone2 but we have used azure tools like Azure Devops, Azure Repos, Azure container registry, docker in VM machine to achive the same results much easily.