THISARA KANDAGE

UNDERGRADUATE - SLIIT

E-mail LinkedIn GitHub Website

Complete End-to-End DevOps Workflow with Azure DevOps, ACR, AKS, and Spring Boot

Introduction:

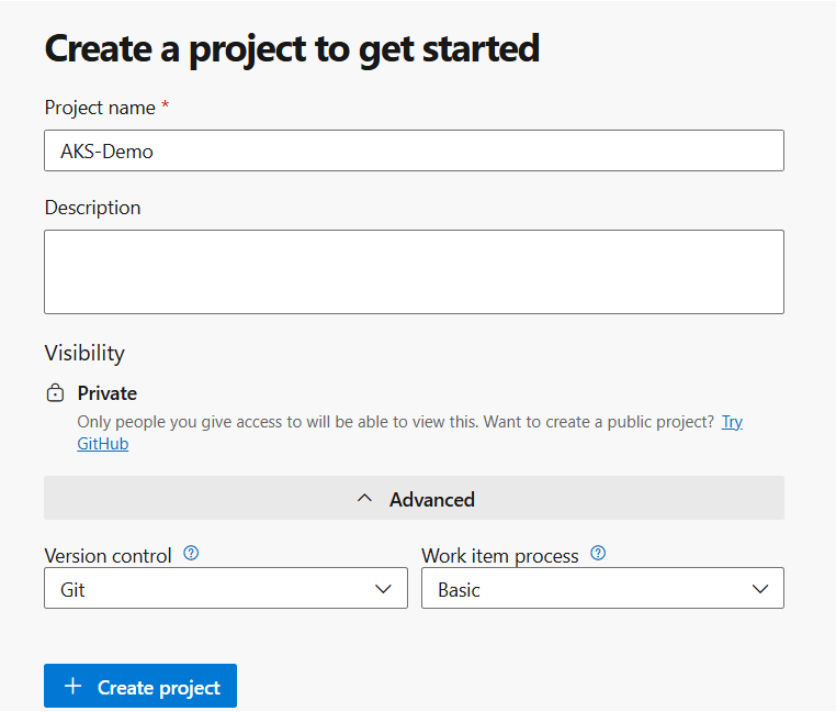
This project demonstrates the implementation of a complete DevOps workflow to automate the build, containerization, and deployment of a **Spring Boot web application** using Microsoft Azure services and Azure DevOps. The goal of the project was to simulate a real-world software delivery pipeline, starting from source code management, continuing through continuous integration and continuous deployment, and ending with a running application accessible through a web browser.

To achieve this, the project utilized **Azure Container Registry (ACR)** for storing Docker images, and an **Azure Kubernetes Service (AKS) cluster** for hosting and managing containerized applications at scale. **Azure Repos** was used as the version control system to store the Spring Boot application code and Docker configuration, ensuring centralized collaboration and traceability of changes. The **Azure DevOps Dashboard** provided the central hub to manage and monitor the build and release pipelines.

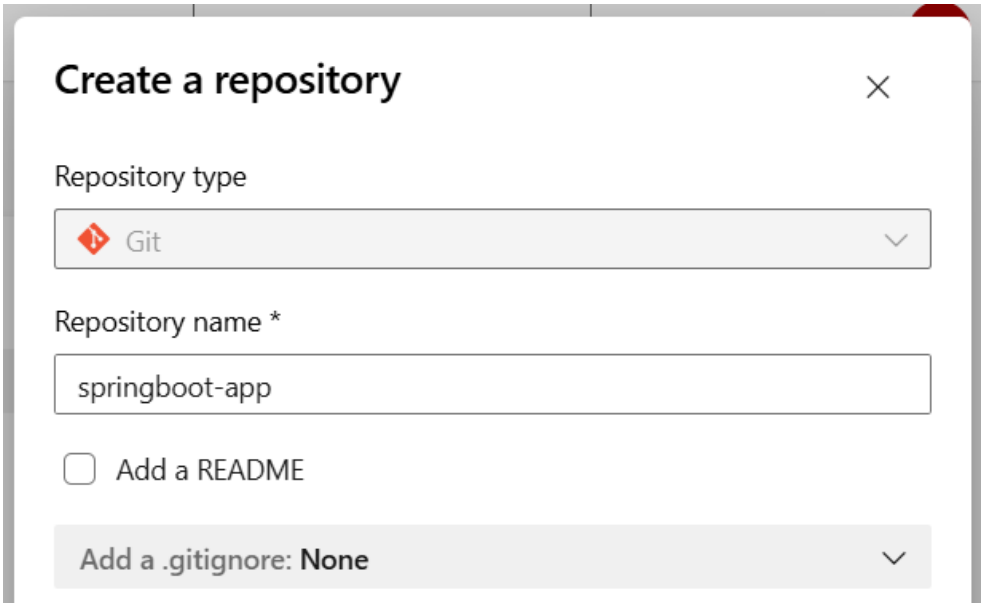
A **CI/CD pipeline** was designed and implemented using **Azure Pipelines**. The build pipeline handled tasks such as compiling the Spring Boot application, packaging it into a JAR file, building a Docker image, and pushing the image to ACR. The release pipeline then deployed the containerized application to the AKS cluster. To execute pipeline jobs efficiently, a **self-hosted agent** was configured within a local RHEL virtual machine. This allowed greater flexibility and control over the build environment.

The deployment and service exposure were managed using Kubernetes manifests, applied with kubectl commands. Integration with the Azure CLI (az commands) was also used for cluster configuration and ACR authentication. Finally, the Spring Boot web application was successfully deployed into AKS, exposed through a LoadBalancer service, and accessed in a web browser via its external IP.

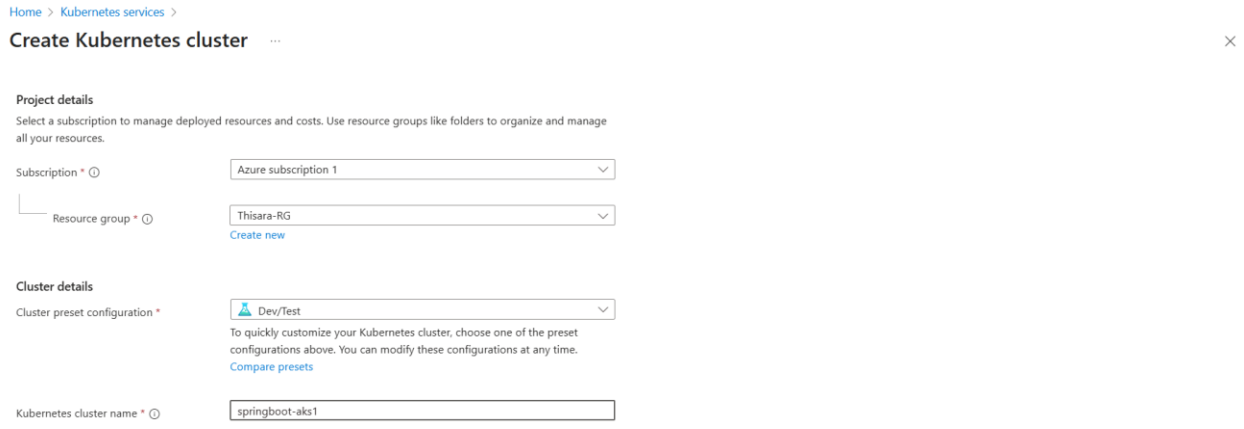
- **Create project in Azure DevOps**



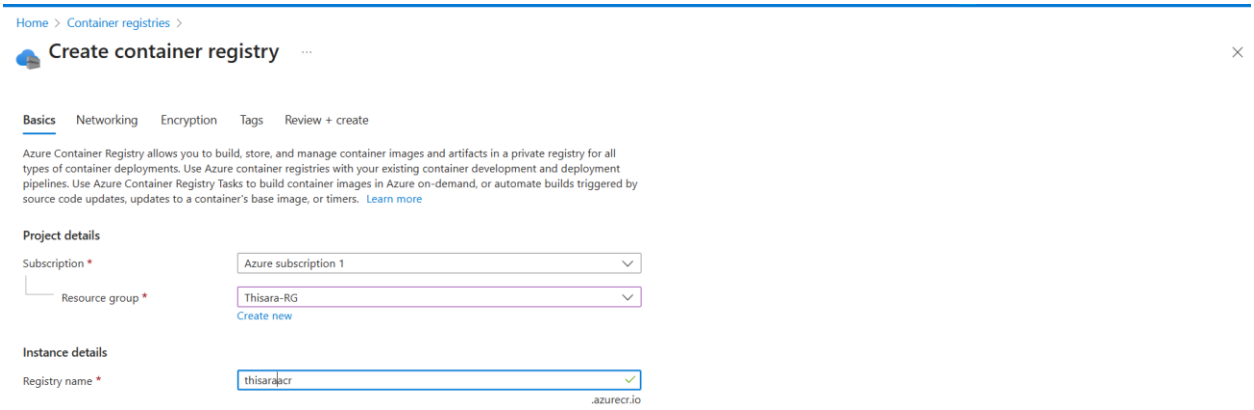
- **Creating a new repository**

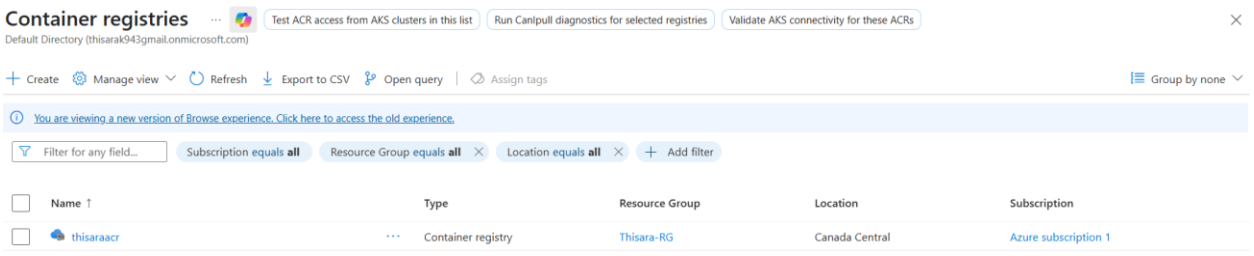


- **Create a Kubernetes cluster in Azure**



- **Create a Container registry for my docker image**





THISARA KANDAGE

UNDERGRADUATE - SLIIT

E-mail LinkedIn GitHub Website

Connect my VM to working as Agent, steps

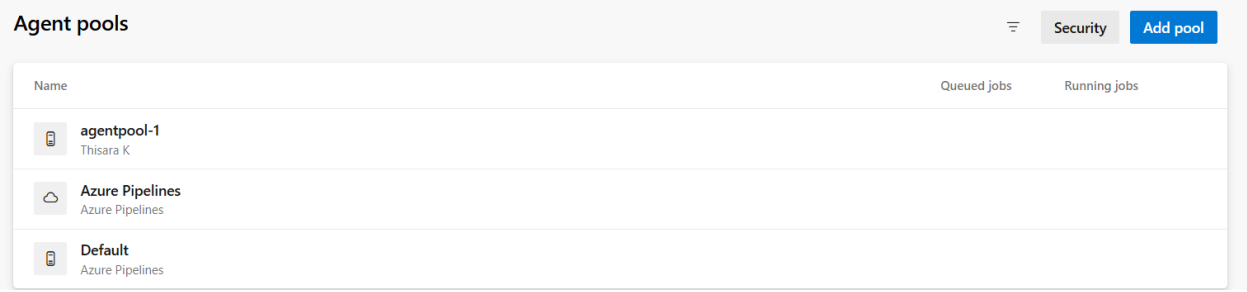
Prerequisite

Docker, maven and java need to be in my system

```
[thisara@localhost ~]$ clear
[thisara@localhost ~]$ mvn -version
Apache Maven 3.5.4 (Red Hat 3.5.4-5)
Maven home: /usr/share/maven
Java version: 1.8.0_452, vendor: Red Hat, Inc., runtime: /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.452.b09-2.el8.x86_64/jre
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "4.18.0-553.60.1.el8_10.x86_64", arch: "amd64", family: "unix"
[thisara@localhost ~]$
[thisara@localhost ~]$ java -version
openjdk version "17.0.15" 2025-04-15 LTS
OpenJDK Runtime Environment (Red_Hat-17.0.15.0.6-1) (build 17.0.15+6-LTS)
OpenJDK 64-Bit Server VM (Red_Hat-17.0.15.0.6-1) (build 17.0.15+6-LTS, mixed mode, sharing)
[thisara@localhost ~]$
[thisara@localhost ~]$ docker --version
Docker version 24.0.5, build ced0996
[thisara@localhost ~]$
[thisara@localhost ~]$
```

Add VM as runner

- **Create agent pool in Azure DevOps organization**

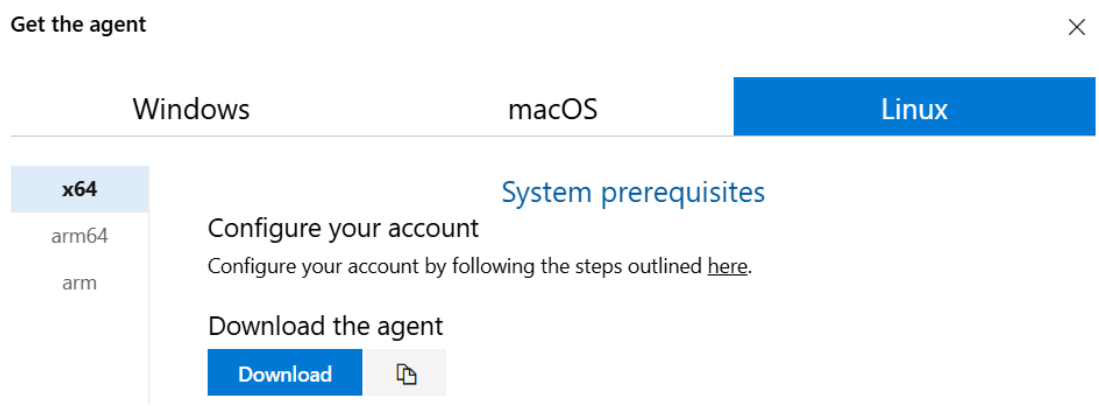


THISARA KANDAGE

UNDERGRADUATE - SLIIT

E-mail LinkedIn GitHub Website

Agent connection steps



- **Download the agent and install it in my VM using wget**

```
[thisara@localhost ~]$ wget https://download.agent.dev.azure.com/agent/4.261.0/vsts-agent-linux-x64-4.261.0.tar.gz
--2025-09-15 04:54:03--  https://download.agent.dev.azure.com/agent/4.261.0/vsts-agent-linux-x64-4.261.0.tar.gz
Resolving download.agent.dev.azure.com (download.agent.dev.azure.com)... 42.99.140.40, 42.99.140.51, 2600:1417:75::17cd:58
8, ...
Connecting to download.agent.dev.azure.com (download.agent.dev.azure.com)|42.99.140.40|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 147793439 (141M) [application/octet-stream]
Saving to: 'vsts-agent-linux-x64-4.261.0.tar.gz'

vsts-agent-linux-x64-4.261.0.t 100%[===================================================>] 140.95M  2.11MB/s    in 73s

2025-09-15 04:55:27 (1.93 MB/s) - 'vsts-agent-linux-x64-4.261.0.tar.gz' saved [147793439/147793439]

[thisara@localhost ~]$
```

- **Create the agent**

```
[thisara@localhost ~]$ mkdir mkdir myagent && cd myagent
[thisara@localhost myagent]$
[thisara@localhost myagent]$
```

- **Extract the files to downloads**

```
[thisara@localhost myagent]$ tar zxvf ~/Downloads/vsts-agent-linux-x64-4.261.0.tar.gz
./
./env.sh
./run.sh
./config.sh
./externals/
./externals/node16/
./externals/node16/CHANGELOG.md
./externals/node16/include/
./externals/node16/include/node/
./externals/node16/include/node/uv.h
./externals/node16/include/node/js_native_api_types.h
```

THISARA KANDAGE

UNDERGRADUATE - SLIIT

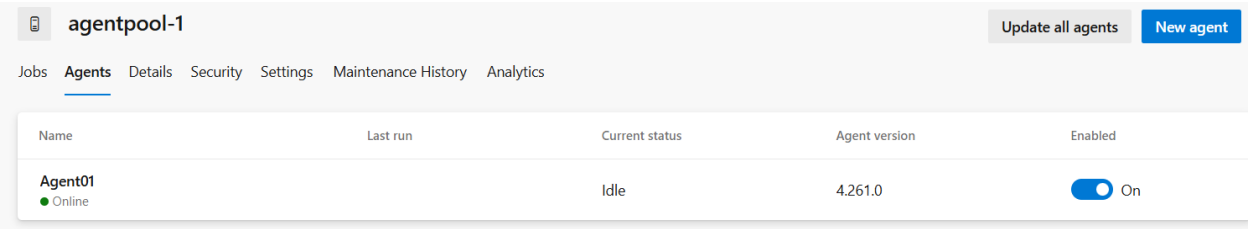
E-mail LinkedIn GitHub Website

- **Configure the agent**

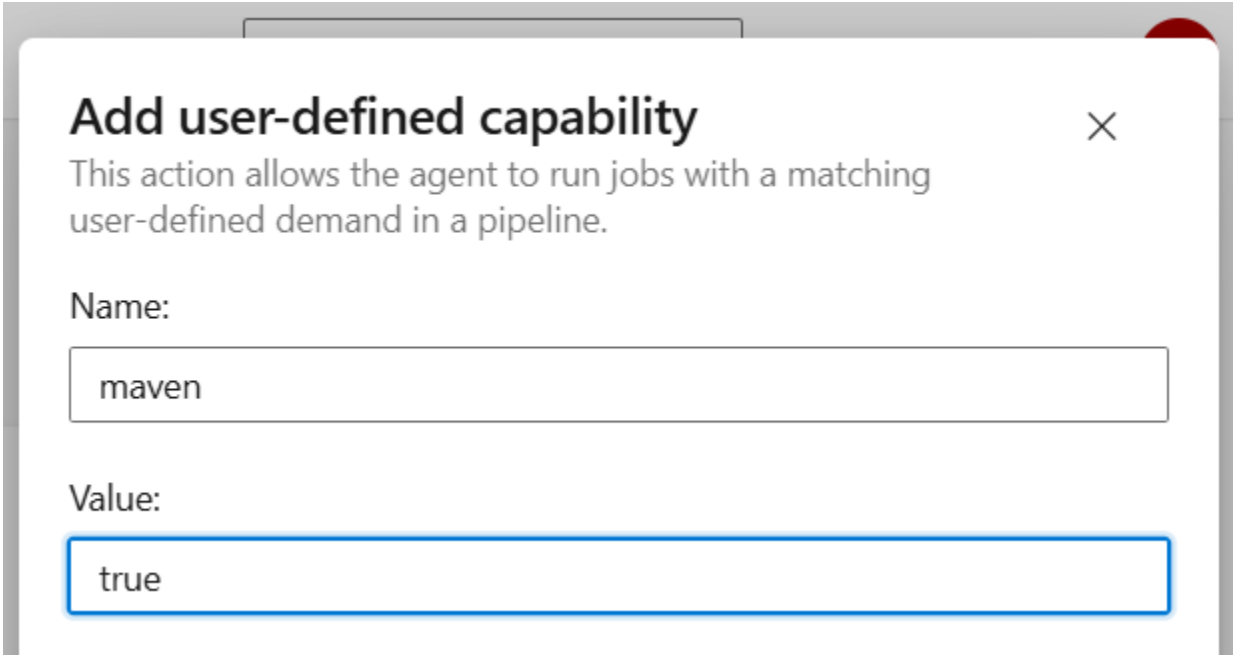
```
[thisara@localhost myagent]$
[thisara@localhost myagent]$ ./config.sh

  ___                      ______ _            _ _
 / _ \                    | ___ (_)          | (_)
/ /_\ \_____   _ _ __ ___ | |_/ /_ _ __    ___| |_ _ __   ___  ___
|  _  |_  / | | | '__/ _ \ |  __/| | '_ \ / _ \ | | '_ \ / _ \/ __|
| | | |/ /| |_| | | |  __/ | |   | | |_) |  __/ | | | | |  __/\__ \
\_| |_/___|\__,_|_|  \___| \_|   |_| .__/ \___|_|_|_| |_|\___||___/
                                   | |
        agent v4.261.0             |_|          (commit 45f3f01)


>> End User License Agreements:

Building sources from a TFVC repository requires accepting the Team Explorer Everywhere E
ep is not required for building sources from Git repositories.
```

- **After create and provide access tokens the agent successfully added**

```
Enter agent pool (press enter for default) > agentpool-1
Enter agent name (press enter for localhost) > Agent01
Enter replace? (Y/N) (press enter for N) > y
Scanning for tool capabilities.
Connecting to the server.
Successfully added the agent
Testing agent connection.
Enter work folder (press enter for _work) >
2025-09-15 12:13:47Z: Settings Saved.
[thisara@localhost myagent]$
[thisara@localhost myagent]$
[thisara@localhost myagent]$
```

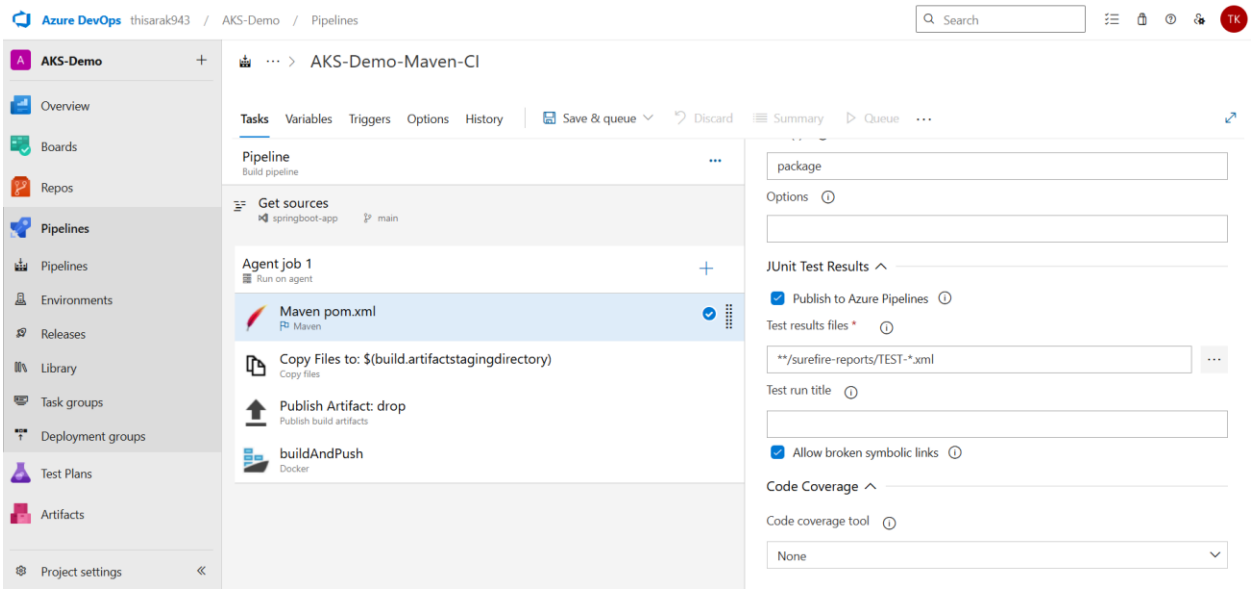


I'm going to run maven project, so I need to add maven as user-defined capability

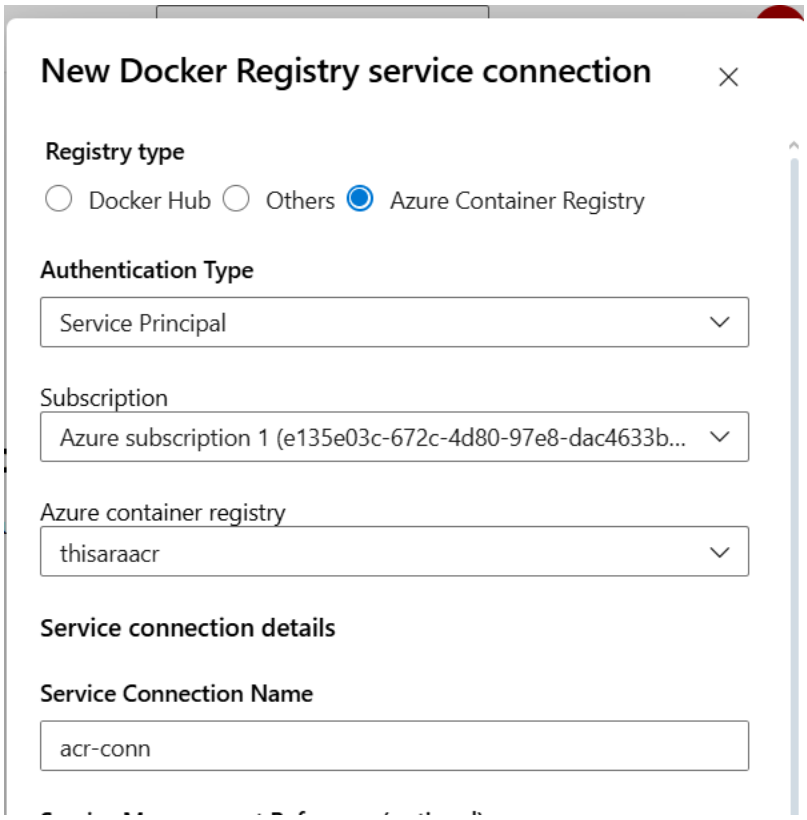


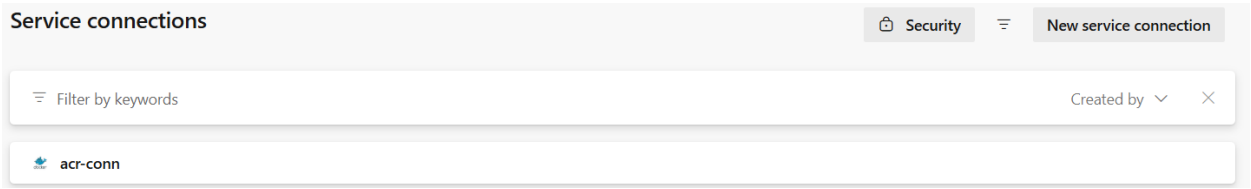
Build the docker image and push it to container registry steps (CI part)

- **Some important things steeped here including maven docker**

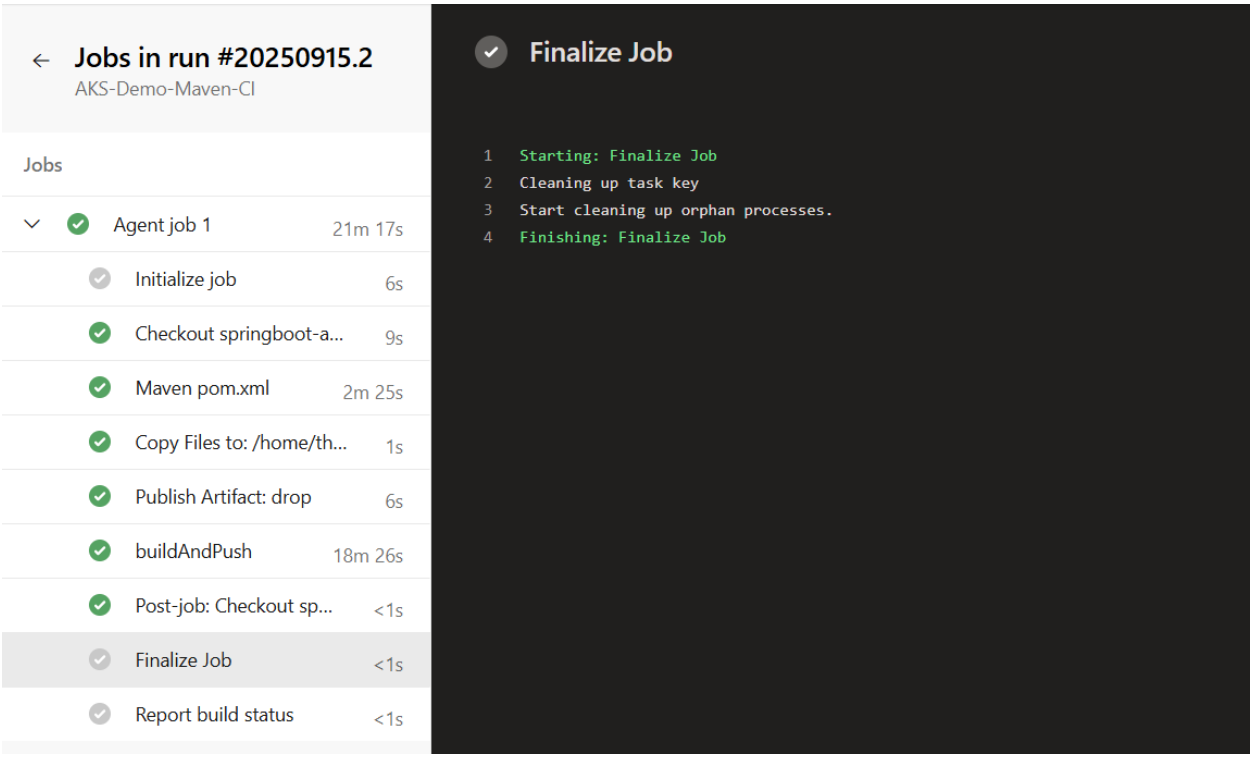


- **Service connection also created**





- **Build is Complete (CI part is Completed)**



- **Output in Container Registry (Docker Image Build and Pushed it to Azure Container Registory)**

- ## Login to azure using VM

```
[thisara@localhost myagent]$ az login --use-device-code
To sign in, use a web browser to open the page https://microsoft.com/devicelogin and enter the code IJ7UPMZKF to authenticate.
[
  {
    "cloudName": "AzureCloud",
    "homeTenantId": "4f7c324a-5437-4b10-8f6f-ecf75f42163a",
    "id": "e135e03c-672c-4d80-97e8-dac4633b1197",
    "isDefault": true,
    "managedByTenants": [],
    "name": "Azure subscription 1",
    "state": "Enabled",
    "tenantId": "4f7c324a-5437-4b10-8f6f-ecf75f42163a",
    "user": {
      "name": "thisarak943@gmail.com",
      "type": "user"
    }
  }
]
[thisara@localhost myagent]$
```

Red Hat Enterprise Linux

- ## Connect AKS Cluster to VM

```
[thisara@localhost myagent]$ az aks get-credentials --resource-group Thisara-RG --name springboot-aks1 --overwrite-existing
Merged "springboot-aks1" as current context in /home/thisara/.kube/config
[thisara@localhost myagent]$
```

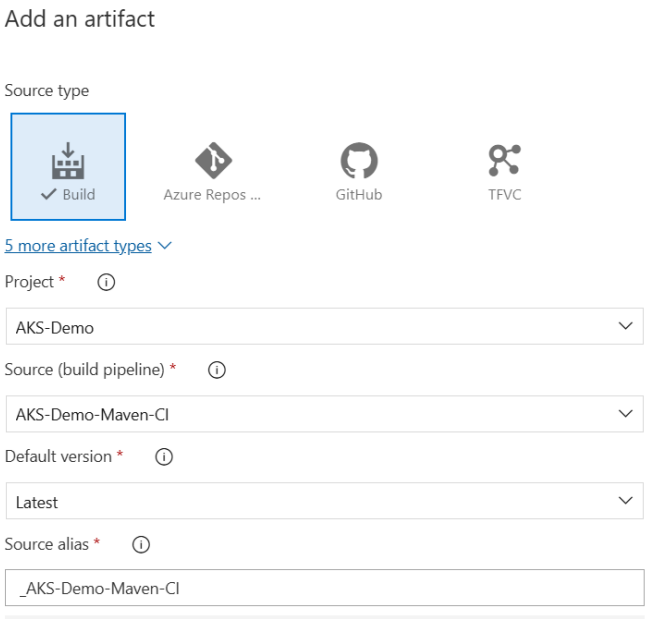
Red Hat

- ## Connect My container registry to Kubernetes Cluster

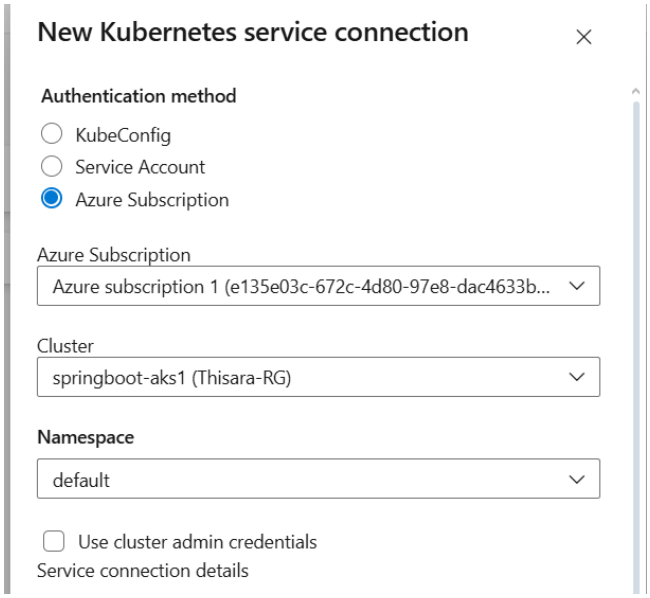
```
[thisara@localhost myagent]$ az aks update -n springboot-aks1 -g Thisara-RG --attach-acr thisaraacr
AAD role propagation done[##############################################]  100.0000%{
  "aadProfile": null,
  "addonProfiles": {
    "azureKeyvaultSecretsProvider": {
      "config": null,
      "enabled": false,
      "identity": null
    },
    "azurepolicy": {
      "config": null,
      "enabled": false,
      "identity": null
    }
  },
```

Deploy to Kubernetes cluster steps (CD part)

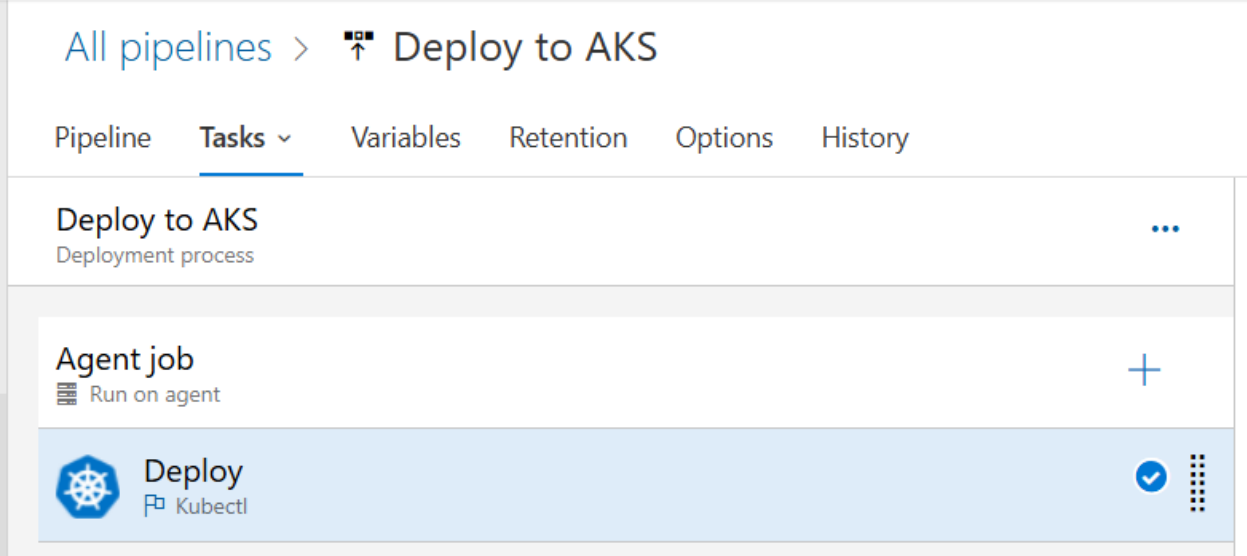
- ## Add an Artifact



- ## Create new Service Connection to Kubernetes



- **Deploy to AKS job Setup Created here**



- **Connect the VM Agent again**

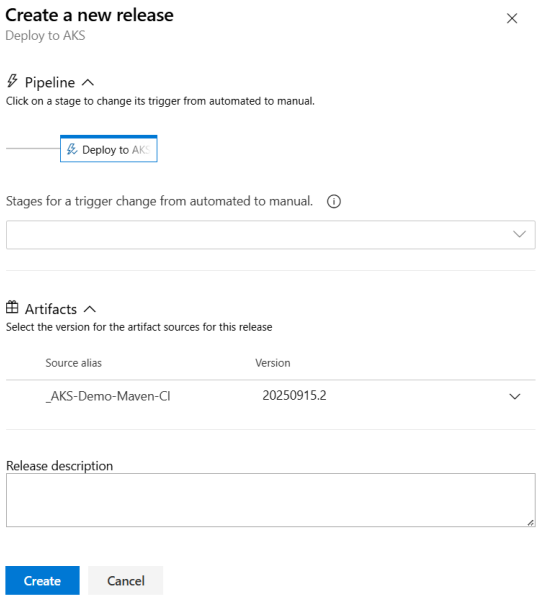
```
[thisara@localhost myagent]$ ./run.sh
Scanning for tool capabilities.
Connecting to the server.
2025-09-15 16:30:05Z: Listening for Jobs
```

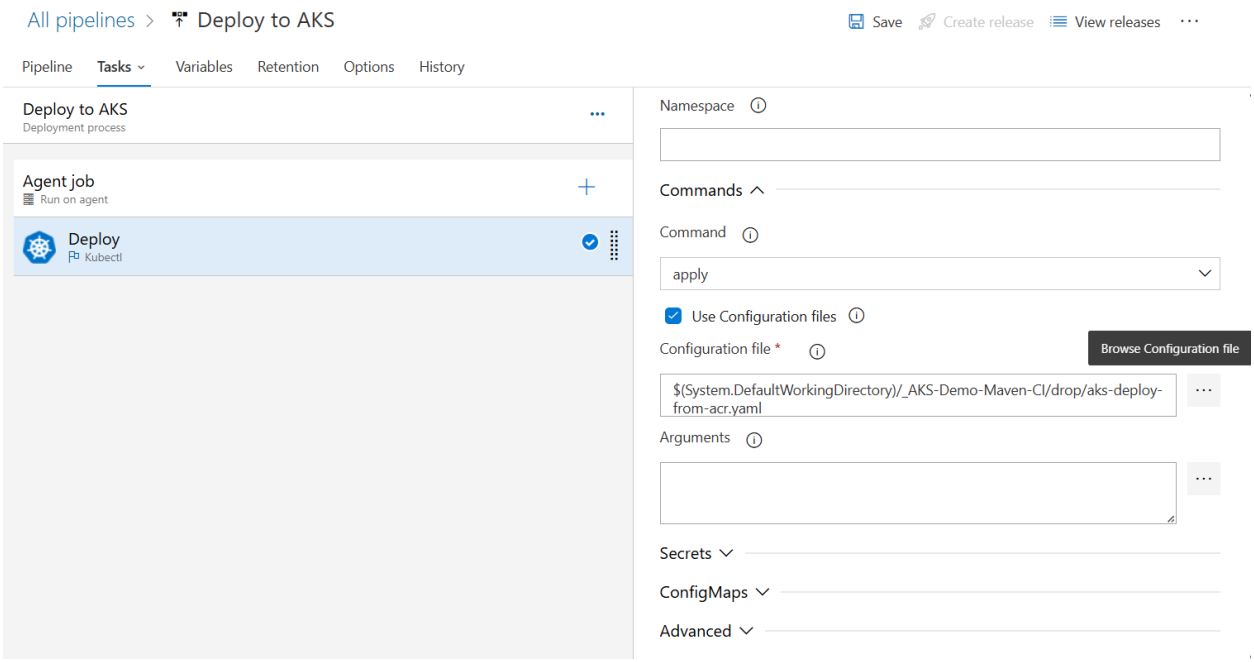
THISARA KANDAGE

UNDERGRADUATE - SLIIT

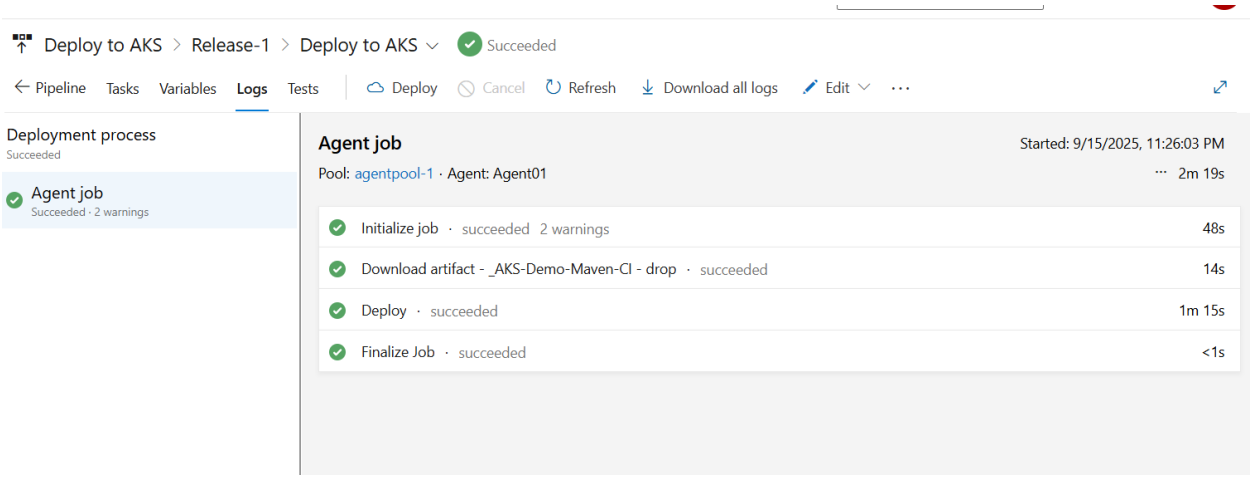
E-mail LinkedIn GitHub Website

- ## Some Snapshots of Deployment Part

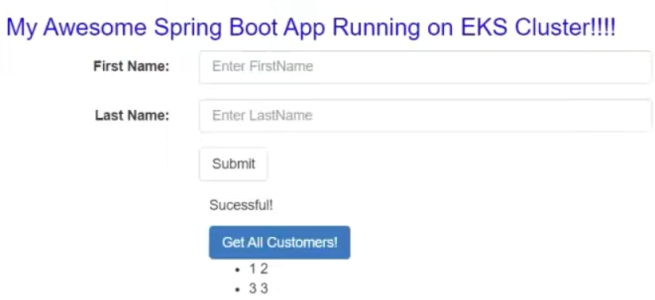




Finally, the Deployment was Succeeded!



- **After accessing the external IP via browser!**



Deployment YML I used

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-springboot-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: springboot-app
  template:
    metadata:
      labels:
        app: springboot-app
    spec:
      containers:
      - name: my-springboot-app
        image: image: thisaraacr.azurecr.io/acr-repol:latest
        imagePullPolicy: Always
        ports:
        - containerPort: 8085
# service type loadbalancer
---
apiVersion: v1
kind: Service
metadata:
  labels:
    app: springboot-app
    k8s-app: springboot-app
  name: springboot-app
spec:
  ports:
  - name: http
    port: 80
    protocol: TCP
    targetPort: 8085
  type: LoadBalancer
  selector:
    app: springboot-app
```

THISARA KANDAGE

UNDERGRADUATE - SLIIT

E-mail LinkedIn GitHub Website

Summary of Learning

Through this project, I gained practical, end-to-end experience in implementing a DevOps pipeline in a cloud-native environment. I learned how to:

- Set up and manage **Azure Container Registry (ACR)** for storing and distributing container images.
- Configure and operate an **Azure Kubernetes Service (AKS) cluster** to deploy and scale containerized applications.
- Work with **Azure Repos** to manage Spring Boot application source code in a version-controlled environment.
- Design and implement **CI/CD pipelines** in Azure DevOps to automate build, test, and deployment processes.
- Use a **self-hosted VM agent** to run Azure DevOps jobs and integrate local infrastructure into the pipeline.
- Apply **Kubernetes manifests** for deploying workloads, services, and exposing applications to external users.
- Utilize **kubectl** and **az CLI commands** to interact with AKS and Azure resources effectively.
- Successfully deliver a **Spring Boot web application** from source code to a live environment accessible via a browser.

This project provided me with strong hands-on exposure to **DevOps practices, Azure cloud infrastructure, CI/CD automation, and Kubernetes deployments**, aligning with industry workflows. It not only improved my technical skills but also gave me confidence in handling the complete software delivery lifecycle, which will be valuable for my career as a DevOps or Cloud Engineer.