



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**

Create a project to get started

Project name *

Description

Visibility

Private

Only people you give access to will be able to view this. Want to create a public project? [Try GitHub](#)

^ Advanced

Version control Work item process

Git Basic

+ Create project

- **Creating a new repository**

Create a repository

Repository type

Git

Repository name *

☐ Add a README

Add a .gitignore: None



Create a Kubernetes cluster in Azure

Home > Kubernetes services >

Create Kubernetes cluster

Project details

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

Subscription * 

Azure subscription 1

Resource group * 

Thisara-RG

Create new

Cluster details

Cluster preset configuration * 

Dev/Test

To quickly customize your Kubernetes cluster, choose one of the preset configurations above. You can modify these configurations at any time.
[Compare presets](#)

Kubernetes cluster name * 

springboot-aks1

Create a Container registry for my docker image

Home > Container registries >

Create container registry

Basics Networking Encryption Tags Review + create

Azure Container Registry allows you to build, store, and manage container images and artifacts in a private registry for all types of container deployments. Use Azure container registries with your existing container development and deployment pipelines. Use Azure Container Registry Tasks to build container images in Azure on-demand, or automate builds triggered by source code updates, updates to a container's base image, or timers. [Learn more](#)

Project details

Subscription * 

Azure subscription 1

Resource group * 

Thisara-RG

Create new

Instance details

Registry name * 

thisaraacr

.azurecr.io

Container registries

Default Directory (thisarak943gmail.onmicrosoft.com)

Test ACR access from AKS clusters in this list Run Canipull diagnostics for selected registries Validate AKS connectivity for these ACRs

+ Create Manage view Refresh Export to CSV Open query Assign tags Group by none

You are viewing a new version of Browse experience. [Click here to access the old experience.](#)

Filter for any field... Subscription equals all Resource Group equals all Location equals all Add filter

<input type="checkbox"/>	Name ↑	Type	Resource Group	Location	Subscription
<input type="checkbox"/>	thisaraacr	Container registry	Thisara-RG	Canada Central	Azure subscription 1



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

Agent pools



Security

Add pool

Name	Queued jobs	Running jobs
 agentpool-1 Thisara K		
 Azure Pipelines Azure Pipelines		
 Default Azure Pipelines		



Agent connection steps

Get the agent



Windows

macOS

Linux

x64

arm64

arm

System prerequisites

Configure your account

Configure your account by following the steps outlined [here](#).

Download the agent

Download



- 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 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](#)

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

Add user-defined capability



This action allows the agent to run jobs with a matching user-defined demand in a pipeline.

Name:

Value:



THISARA KANDAGE

UNDERGRADUATE - SLIIT

[E-mail](#) [LinkedIn](#) [GitHub](#) [Website](#)

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

- Some important things steeped here including maven docker

The screenshot displays the Azure DevOps web interface for a pipeline named 'AKS-Demo-Maven-CI'. The left sidebar shows the navigation menu with 'Pipelines' selected. The main area shows the pipeline configuration for 'Agent job 1'. The tasks listed are 'Get sources' (using 'springboot-app' and 'main'), 'Maven pom.xml' (using 'Maven'), 'Copy Files to: \$(build.artifactstagingdirectory)' (using 'Copy files'), 'Publish Artifact: drop' (using 'Publish build artifacts'), and 'buildAndPush' (using 'Docker'). The right-hand pane shows the configuration for the 'buildAndPush' task, including the 'package' command, 'Options', 'JUnit Test Results' (with 'Publish to Azure Pipelines' checked), 'Test results files' (set to '**/surefire-reports/TEST-*.xml'), 'Test run title', 'Allow broken symbolic links' (checked), and 'Code Coverage' (set to 'None').



• Service connection also created

New Docker Registry service connection

Registry type

☐ Docker Hub

☐ Others

☒ Azure Container Registry

Authentication Type

Service Principal

Subscription

Azure subscription 1 (e135e03c-672c-4d80-97e8-dac4633b...

Azure container registry

thisaraacr

Service connection details

Service Connection Name

acr-conn

Service connections		Security	New service connection
Filter by keywords		Created by	
acr-conn			



- Build is Complete (CI part is Completed)

← Jobs in run #20250915.2

AKS-Demo-Maven-CI

Jobs

▼	✓ Agent job 1	21m 17s
	✓ Initialize job	6s
	✓ Checkout springboot-a...	9s
	✓ Maven pom.xml	2m 25s
	✓ Copy Files to: /home/th...	1s
	✓ Publish Artifact: drop	6s
	✓ buildAndPush	18m 26s
	✓ Post-job: Checkout sp...	<1s
	✓ Finalize Job	<1s
	✓ Report build status	<1s

✓ Finalize Job

1 Starting: Finalize Job

2 Cleaning up task key

3 Start cleaning up orphan processes.

4 Finishing: Finalize Job

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

Refresh Manage Deleted Repositories

Search to filter repositories ...

Repositories ↑↓

Cache Rule

acr-repo1



• 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]$
```



• 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]$
```



• Connect My container registry to Kubernetes Cluster

```
[thisara@localhost myagent]$
[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
    }
  },
  "azureRoleBasedAccessControl": {
    "config": null,
    "enabled": false,
    "identity": null
  }
}
```

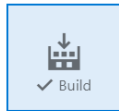


Deploy to Kubernetes cluster steps (CD part)

- **Add an Artifact**

Add an artifact

Source type



[5 more artifact types](#) ▾

Project * ⓘ

AKS-Demo ▾

Source (build pipeline) * ⓘ

AKS-Demo-Maven-CI ▾

Default version * ⓘ

Latest ▾

Source alias * ⓘ

_AKS-Demo-Maven-CI

- **Create new Service Connection to Kubernetes**

New Kubernetes service connection

Authentication method

☐ KubeConfig

☐ Service Account

☒ Azure Subscription

Azure Subscription

Azure subscription 1 (e135e03c-672c-4d80-97e8-dac4633b... ▾

Cluster

springboot-aks1 (Thisara-RG) ▾

Namespace

default ▾

☐ Use cluster admin credentials

Service connection details



- [Deploy to AKS job Setup Created here](#)

All pipelines > Deploy to AKS

Pipeline **Tasks** ▾ Variables Retention Options History

Deploy to AKS
Deployment process

Agent job
 Run on agent

Deploy
 Kubectl

- [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
```



Some Snapshots of Deployment Part

Create a new release

Deploy to AKS

⚙️ Pipeline

Click on a stage to change its trigger from automated to manual.

Deploy to AKS

Stages for a trigger change from automated to manual.

1

📦 Artifacts

Select the version for the artifact sources for this release

Source alias	Version
_AKS-Demo-Maven-CI	20250915.2

Release description

Create

Cancel

All pipelines > Deploy to AKS

Save

Create release

View releases

Pipeline Tasks Variables Retention Options History

Deploy to AKS
Deployment process

Agent job

Run on agent

Deploy

Kubectl

Namespace

Commands

Command

apply

Use Configuration files

Configuration file

\$(System.DefaultWorkingDirectory)/_AKS-Demo-Maven-CI/drop/aks-deploy-from-acr.yaml

Arguments

Secrets

ConfigMaps

Advanced



THISARA KANDAGE

UNDERGRADUATE - SLIIT

[E-mail](#) [LinkedIn](#) [GitHub](#) [Website](#)

Finally, the Deployment was Succeeded!

Deploy to AKS > Release-1 > Deploy to AKS ✓ Succeeded

[← Pipeline](#) [Tasks](#) [Variables](#) [Logs](#) [Tests](#) | [Deploy](#) [Cancel](#) [Refresh](#) [Download all logs](#) [Edit](#) [...](#)

Deployment process
Succeeded

✓ Agent job
Succeeded · 2 warnings

Agent job Started: 9/15/2025, 11:26:03 PM
Pool: agentpool-1 · Agent: Agent01 2m 19s

✓ Initialize job · succeeded 2 warnings	48s
✓ Download artifact - _AKS-Demo-Maven-CI - drop · succeeded	14s
✓ Deploy · succeeded	1m 15s
✓ Finalize Job · succeeded	<1s

- After accessing the external IP via browser!

My Awesome Spring Boot App Running on EKS Cluster!!!!

First Name:

Last Name:

Successfull!

- 1 2
- 3 3

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
```




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.