

Jenkins Pipeline for Spring Boot Java Application

Contents

Jenkins Pipeline for Spring Boot Java Application	1
Pre-requisite	1
Initialize Jenkins pipeline	2
Add stages.....	4
Sonar Integration.....	6
Build JAR and create Docker image.....	10
Adding ACR	14
Push a Docker image to Azure Container Registry (ACR) using Managed Identity through the Azure Portal — no secrets or credentials needed	14
Kubernetes Deployment	17
Kubernetes login	17
Deploy image from ACR to AKS	18

Pre-requisite

Create a vm, install java-jdk, install Jenkins, trivy. Install stage view plugin in jenkins.

Open port 8080

The screenshot shows the Azure portal interface for managing a Network Security Group (NSG). The left sidebar lists various resources under 'All resources'. The main area shows the 'jenkins-vm-nsg' NSG details. The 'Overview' tab is selected, displaying basic information like Resource group, Location, and Subscription ID. The 'Rules' tab is open, showing existing rules for SSH (port 22) and Jenkins (port 8080). A new rule is being created with the name 'AllowAnyCustom8080Inbound', source set to 'Any', protocol set to 'Any', and destination port range set to '8080'. The 'Save' button is highlighted at the bottom right of the dialog.

In local pc, clone the petclinic project. Open in vscode

```
newreadme.md > ## Install jdk and jenkins
1 ## Install jdk and jenkins
2 ...
3 ...
4 sudo apt update
5 sudo apt install openjdk-21-jdk -y
6 ...
7 ...
8 ### Install Jenkins
9 ...
10 ...
11 curl -fsSL https://pkg.jenkins.io/debian/jenkins.io-2023.key | sudo tee \
| /usr/share/keyrings/jenkins-keyring.asc > /dev/null
12 echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
https://pkg.jenkins.io/debian binary/ | sudo tee \
/etc/apt/sources.list.d/jenkins.list > /dev/null
13 sudo apt-get update -y
14 sudo apt-get install jenkins -y
...
15 ...
16 ...
17 ...
18 ...
19 ...
20 ...
21 ...
22 ...
23 ...
24 ...
25 ...
26 ...
27 ...
28 ...
29 ...
30 ...

### Install Trivy
1 sudo apt-get install wget apt-transport-https gnupg lsb-release -y
2 wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | sudo apt-key add -
3 echo deb https://aquasecurity.github.io/trivy-repo/deb $(lsb_release -sc) main | sudo tee -a /etc/apt/sources.list.d/trivy.list
4 sudo apt update
5 sudo apt install trivy -y
6 ...
7 ...
8 ...
9 ...
10 ...
11 ...
12 ...
13 ...
14 ...
15 ...
16 ...
17 ...
18 ...
19 ...
20 ...
21 ...
22 ...
23 ...
24 ...
25 ...
26 ...
27 ...
28 ...
29 ...
30 ...

### sonarqube
1 sonar project create, save project keys and token
2 Create a webhook ip:8080/sonarqube-webhook/
30
```

kaurj@Jaspreet MINGW64 ~/Documents/documents/my_docs/Azure Cloud/jenkins/petclinic (master)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE

bash

Initialize Jenkins pipeline

Create a Jenkins pipeline. Set the pipeline trigger to GitHub hook trigger for GITScm polling

Jenkins / jenkins-pipeline / Configuration

Configure

General

Triggers

Pipeline

Advanced

Throttle builds ?

Triggers

Set up automated actions that start your build based on specific events, like code changes or scheduled times.

Build after other projects are built ?

Build periodically ?

GitHub hook trigger for GITScm polling ?

Poll SCM ?

Trigger builds remotely (e.g., from scripts) ?

Pipeline

Define your Pipeline using Groovy directly or pull it from source control.

Definition

Pipeline script from SCM

SCM ?

Save Apply

Get the pipeline script from Source code management GIT.

The screenshot shows the Jenkins Pipeline configuration page. The left sidebar has tabs: General, Triggers, Pipeline (selected), and Advanced. The main area is titled 'Definition' with a dropdown set to 'Pipeline script from SCM'. Under 'SCM', it's set to 'Git'. The 'Repositories' section contains one repository with the URL 'https://github.com/kaurjkg/petclinic.git' and no credentials. There are 'Add Repository' and 'Advanced' buttons. At the bottom are 'Save' and 'Apply' buttons.

The screenshot shows the Jenkins Pipeline configuration page. The left sidebar has tabs: General, Triggers, Pipeline (selected), and Advanced. The main area is titled 'Definition' with a dropdown set to 'Pipeline script from SCM'. Under 'Branch Specifier (blank for "any")', it shows '/master'. There are 'Add Branch' and 'Repository browser' buttons. The 'Additional Behaviours' section has an 'Add' button. Under 'Script Path', it shows 'Jenkinsfile'. A checked checkbox for 'Lightweight checkout' is present. At the bottom are 'Save' and 'Apply' buttons.

Add github webhook trigger for Jenkins pipeline, so that github can trigger the pipeline when code is pushed to the project repo

The screenshot shows the GitHub settings interface for the repository 'petclinic'. The left sidebar has 'Webhooks' selected under the 'Actions' section. The main area is titled 'Webhooks / Manage webhook'. It contains fields for 'Payload URL' (set to 'http://172.191.208.216:8080/github-webhook/'), 'Content type' (set to 'application/json'), and 'Secret'. Below these, there's a section for 'SSL verification' with options 'Enable SSL verification' (radio button) and 'Disable (not recommended)'. At the bottom, there's a section for 'Which events would you like to trigger this webhook?'.

Add stages

In Jenkins server-> manage Jenkins-> add maven. This name will be used in jenkinsfile pipeline code.

The screenshot shows the Jenkins 'Manage Jenkins / Tools' page. Under 'Maven installations', a new entry 'maven-proj' is being added. The 'Name' field is filled with 'maven-proj'. The 'Install automatically?' checkbox is checked. A sub-section 'Install from Apache' shows 'Version' set to '3.9.9'. There are 'Save' and 'Apply' buttons at the bottom.

Add stages to clone github code on Jenkins agent (VM) using checkout stage. Add another stage to compile the code using maven. Third stage for trivy which is used to scan code for any security threats.

The screenshot shows a code editor interface with the Jenkinsfile open. The Jenkinsfile contains a pipeline configuration with stages for checkout, Maven compile, Trivy scan, and Sonar analysis. The code is syntax-highlighted in red and blue. The terminal tab at the bottom shows a command-line session where the Jenkinsfile was run.

```
1 pipeline {
2     environment {
3     }
4 }
5
6 stages {
7     stage('Checkout from git'){
8         steps{
9             echo "checking out from git"
10            git branch:'master', url:'https://github.com/kaurjkg/petclinic.git'
11        }
12    }
13    stage('Maven Compile') {
14        steps {
15            echo 'This is maven compile stage'
16            sh 'mvn compile'
17        }
18    }
19    stage('Trivy file system scan'){
20        steps{
21            echo 'Trivy file system scan started'
22            sh 'trivy fs --format table --output trivy-report.txt --severity HIGH,CRITICAL .'
23        }
24    }
25    stage('Sonar Analysis') {
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE

kaurj@Jaspreet MINGW64 ~/Documents/documents/my docs/Azure Cloud/jenkins/petclinic (master)

Check trivy output in Jenkins vm agent by logging as sudo su - , su – jenkins.

172.191.208.216 (jaspreet)

Terminal Sessions View X server Tools Games Settings Macros Help

LICENSE.txt mvnw pom.xml target
jenkins@jenkins-vm:~/workspace\$ cat trivy-report.txt
cat: trivy-report.txt: No such file or directory
jenkins@jenkins-vm:~/workspace\$ cd jenkins-pipeline
jenkins@jenkins-vm:~/workspace/jenkins-pipeline\$ cat trivy-report.txt

Report Summary

Target	Type	Vulnerabilities	Secrets
pom.xml	pom	0	-
gittoken.txt	text	-	1

Legend:
- '-' : Not scanned
- '0' : Clean (no security findings detected)

gittoken.txt (secrets)
=====

Total: 1 (HIGH: 0, CRITICAL: 1)

CRITICAL: GitHub (github-fine-grained-pat)

GitHub Fine-grained personal access tokens

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.com>

Sonar Integration

Create Sonarqube project and save information

The screenshot shows the SonarCloud interface for creating a new project. On the left, there's a form with fields for Organization (jasjkg), Display Name (petclinic), Project Key (jasjkg_petclinic), and Project visibility (Public). To the right, a sidebar offers an 'Upgrade for more features' plan with a 14-day free trial. It lists benefits like up to 1.9M lines of code, unlimited team members, feedback on branches, and GitHub member sync. Buttons for 'Start free trial' and 'Compare plans' are present. Below this, a section for existing coupons is shown.

Create a token

The screenshot shows the SonarCloud security settings page. Under 'Generate Tokens', a new token named "petclinic-jenkins" has been created. The token value is a long string of characters. Below this, a table lists existing tokens: "Analyze *helloworld*" and "Analyze *helloworld* 1", both created on February 10, 2025. To the right, a sidebar promotes upgrading to scan private projects, starting from \$64/month for the Team plan. It lists features like 14-day free trial, open-source and private projects, and advanced issue detection. Buttons for 'Upgrade your organization' and 'View FAQs' are available.

In Jenkins->Manage Jenkins-> Install plugin Sonarqube scanner

The screenshot shows the Jenkins 'Manage Jenkins / Plugins' page. A search bar at the top right contains the text 'sonar'. On the left, a sidebar menu includes 'Updates', 'Available plugins', **Installed plugins** (selected), 'Advanced settings', and 'Download progress'. The main content area displays the 'SonarQube Scanner for Jenkins 2.18' plugin, which is described as allowing an easy integration of SonarQube for Continuous Inspection of code quality. The plugin is marked as 'Enabled' with a blue toggle switch and a red 'x' icon for uninstallation.

Add token information in Jenkins Global Credentials

The screenshot shows the Jenkins 'Manage Jenkins / Credentials / System / Global credentials (unrestricted)' page. A new credential is being created with the 'Kind' set to 'Secret text'. The 'Scope' is set to 'Global (Jenkins, nodes, items, all child items, etc.)'. The 'Secret' field contains a series of asterisks ('*****'). The 'ID' is set to 'sonarqubetoken'. The 'Description' field is empty. At the bottom, a blue 'Create' button is visible. The status bar at the bottom right indicates 'REST API Jenkins 2.507'.

Server Settings in System

The screenshot shows the Jenkins 'System' configuration page under 'SonarQube servers'. A new server named 'sonar-server' is being configured with a Server URL of 'https://sonarcloud.io/' and a Server authentication token of 'sonarqubetoken'. There is also an 'Environment variables' section which is currently empty.

SonarQube servers

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

Environment variables

SonarQube installations

List of SonarQube installations

Name: sonar-server

Server URL: Default is <http://localhost:9000>
https://sonarcloud.io/

Server authentication token: SonarQube authentication token. Mandatory when anonymous access is disabled.
sonarqubetoken

+ Add

Save Apply

Add sonar scanner client in Tools

The screenshot shows the Jenkins 'Tools' configuration page under 'Add SonarQube Scanner'. A new scanner named 'sonar-scanner' is being added via 'Install from Maven Central' with version 'SonarQube Scanner 7.1.0.4889'. The 'Install automatically' checkbox is selected.

Add SonarQube Scanner

SonarQube Scanner

Name: sonar-scanner

Install automatically ?

Install from Maven Central

Version: SonarQube Scanner 7.1.0.4889

Add Installer ▾

Add SonarQube Scanner

Save Apply

Added Pipeline code

The screenshot shows a Visual Studio Code (VS Code) interface with the following details:

- File Explorer:** Shows files like Jenkinsfile, k8s, src, .editorconfig, .gitattributes, .gitignore, Dockerfile, gittoken.txt, Jenkinsfile (selected), LICENSE.txt, mvnw, mvnw.cmd, newreadme.md, pom.xml, and readme.md.
- Editor:** The Jenkinsfile is open in the main editor area. The code defines a pipeline with stages for Sonar Analysis and Sonar Quality Gate, and includes steps for running sonar-scanner.
- Terminal:** The terminal at the bottom shows the output of a git push command:

```
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 434 bytes | 434.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/kaurjk/petclinic.git
   63c911b..0472127 master -> master
```
- Bottom Status Bar:** Shows the current branch as master, the commit hash, and the date of the commit (3 days ago).

After pipeline ran automatically, sonar tests passed

The screenshot shows the SonarCloud interface for the 'petclinic' project. The left sidebar has a dark theme with navigation links: Overview (selected), Main Branch, Pull Requests (0), Branches (1), Information, Administration, and Collapse. The main content area has a light theme. It displays the project name 'petclinic' with a large orange 'P'. Below it is a horizontal bar representing code coverage or issue distribution across various file types: CSS, Java, XML, PL/SQL, JSP, YAML, and Docker. The bar is mostly blue with some orange segments. To the right, it says 'No tags', 'Last analysis May 05, 2025', and '11k Lines of Code'. Under 'Main Branch Status', there's a 'Quality Gate' section showing a green checkmark and the word 'Passed'. An illustration of a computer monitor with code and a checkmark is shown below. A message at the bottom says 'Enjoy your sparkling clean code!'. To the right, under 'Main Branch Evolution', it shows '48 Issues' with tabs for 'Issues', 'Coverage', and 'Duplications'.

Build JAR and create Docker image

Add maven package step for compiling project and building war file

The screenshot shows the Azure DevOps code editor interface. The left sidebar displays a file tree for a project named 'PETCLINIC' containing files like Jenkinsfile, Dockerfile, README.md, and LICENSE.txt. The main editor area shows a Jenkinsfile with the following content:

```
1 pipeline {
2     // Sonar quality gate not working because basic plan doesn't allow to create custom quality gate
3     // timeout(time: 1, unit: 'MINUTES') {
4     //     waitForQualityGate abortPipeline: true, credentialsId: 'sonarqubetoken'
5     //}
6     //Also skipped adding webhook for sonar quality gate
7
8     stage('Build using maven package'){
9         steps{
10             echo 'This is maven package stage'
11             sh 'mvn package'
12         }
13     }
14     stage('Build docker image'){
15
16     }
17 }
```

The bottom of the editor shows a terminal window with the following output:

```
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 344 bytes | 344.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/kaurjkg/petclinic.git
 6e630ff8..69dff7e master -> master
```

The status bar at the bottom indicates the user is in a bash session.

Install Docker in server vm

The screenshot shows a terminal window titled "172.191.208.216 (jaspreet)". The window has a menu bar with "Terminal", "Sessions", "View", "X server", "Tools", "Games", "Settings", "Macros", and "Help". Below the menu is a toolbar with icons for "Qt", "Home", "New Session", "Close", and "Copy/Paste". The terminal session itself contains the following text:

```
# 1. Install Docker
curl -fsSL https://get.docker.com | bash

# 2. Add your current user to the docker group
sudo usermod -aG docker $USER

# 3. Add the Jenkins user to the docker group
sudo usermod -aG docker jenkins

# 4. Restart Docker
sudo systemctl restart docker

# 5. Restart Jenkins (important to apply group changes)
sudo systemctl restart jenkins

# 6. Optional: verify Docker is enabled on boot
sudo systemctl enable docker
```

At the bottom of the terminal, it shows the file "docker.sh" with 18L and 436B, and status indicators "18,0-1" and "All". A watermark at the bottom reads "UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.com>".

Install docker commons and docker pipeline plugins in Jenkins

Jenkins / Manage Jenkins / Plugins

Plugins

Updates

Available plugins

Installed plugins

Advanced settings

Search bar: docker

Name	Enabled
Docker API Plugin	Enabled
Docker Commons Plugin	Enabled
Docker Pipeline	Enabled

REST API Jenkins 2.507

Create dockerfile in project root folder

File Edit Selection View Go Run ... ← → 🔍 petclinic 🌐

EXPLORER

- PETCLINIC
 - > k8s
 - > src
 - .editorconfig
 - .gitattributes
 - .gitignore
 - Dockerfile
 - gittoken.txt
 - Jenkinsfile
 - LICENSE.txt
 - mvnw
 - mvnw.cmd
 - newreadme.md
 - pom.xml
 - readme.md

Jenkinsfile Dockerfile newreadme.md readme.md

```

FROM jetty:11-jdk17
# Set environment variable (optional, for clarity)
ENV WAR_FILE petclinic.war
# Copy your WAR file into Jetty's webapps directory
COPY target/${WAR_FILE} /var/lib/jetty/webapps/ROOT.war
# Expose Jetty's default port
EXPOSE 8080

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE

Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 344 bytes | 344.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/kaurjkg/petclinic.git
6e630f8..69dff7e master -> master

kaurj@Jaspreet MINGW64 ~/Documents/documents/my docs/Azure Cloud/jenkins/petclinic (master)

\$ []

+ ⌂ powershell
+ bash
+ bash

master 0 △ 0 jaspreet (2 weeks ago) ln 9, Col 17 Spaces: 4 UTF-8 LF {} Docker

Add docker build script and environment variables for image name and tag name

The screenshot shows a code editor interface with two tabs open: 'Jenkinsfile' and 'Dockerfile'. The Jenkinsfile contains a pipeline configuration with stages for building a docker image and checking out from git. The Dockerfile defines a Docker image for the petclinic application. The terminal below shows the output of a git clone command for the petclinic repository.

```

Jenkinsfile
1 pipeline {
2     // Sonar quality gate not working
3     stage('Build docker image'){
4         steps{
5             echo 'Building docker image'
6             script{
7                 docker.build("$IMAGE_NAME:${IMAGE_TAG}")
8                 echo 'Docker image built successfully'
9             }
10        }
11    }
12 }
13
14 stages {
15     stage('Checkout from git'){
16         steps{
17             echo "Checking out from git"
18             git branch:'master',
19             url:'https://github.com/kaurjk/petclinic'
20         }
21     }
}

```

```

Dockerfile
1 FROM maven:3.6-jdk-8-alpine
2 MAINTAINER kaurjk@outlook.com
3 ENV IMAGE_NAME=petclinic
4 ENV IMAGE_TAG=v1.0
5 ENV MAVEN_OPTS=-Xms512m -Xmx1024m
6
7 COPY . /opt/app-root/src
8 RUN mvn clean package -DskipTests
9
10 WORKDIR /opt/app-root/src
11 EXPOSE 8080
12
13 ENTRYPOINT ["java", "-Djava.security.egd=file:/dev/./urandom", "-jar", "target/petclinic-1.0.jar"]

```

TERMINAL

```

Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 344 bytes | 344.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/kaurjk/petclinic.git
   6e630f8..69dff7e  master -> master

```

Adding ACR

Azure Container registry with its credentials

The screenshot shows the Azure Portal interface for managing an Azure Container Registry (ACR). The 'Access keys' section is selected, displaying the 'Registry name' (jkgjenkins), 'Login server' (jkgjenkins.azurecr.io), and 'Admin user' (empty). The left sidebar shows other management options like Overview, Activity log, Access control (IAM), Tags, Quick start, Resource visualizer, Events, Settings, and Services.

Push a Docker image to Azure Container Registry (ACR) using Managed Identity through the Azure Portal — no secrets or credentials needed

Assign System assigned identity to the Jenkins-vm

The screenshot shows the Azure portal interface for managing a virtual machine named 'jenkins-vm'. The left sidebar has a 'Identity' section selected. Under 'System assigned', the status is 'On' and the object principal ID is listed as 6dc914b9-ec88-4a3c-a92b-63d928deb521. A note at the bottom states: 'This resource is registered with Microsoft Entra ID. The managed identity can be configured to allow access to other resources. Be careful when making changes to the access settings for the managed identity because it can result in failures.' Buttons for 'Save', 'Discard', 'Refresh', and 'Got feedback?' are at the top right.

Add acrpush role of Azure container registry to Jenkins-vm

The screenshot shows the Azure portal interface for managing the 'jkgjenkins' container registry. The left sidebar has an 'Access control (IAM)' section selected. In the main area, under 'Role assignments', a search bar shows 'acrpush' and a table lists one result: 'jenkins-vm' with 'Managed identity' and 'AcrPush' role assigned to 'This resource'. The 'Notifications' sidebar on the right shows three recent events: 'Added Role assignment' (jenkins-vm added as AcrPush for jkgjenkins), 'Enabled system assigned managed identity' (Successfully registered 'jenkins-vm' with Microsoft Entra ID), and 'Deployment succeeded' (Deployment 'Microsoft.ContainerRegistry' to resource group 'rg' was successful). Buttons for 'Go to resource' and 'Pin to dashboard' are at the bottom right of the notifications.

Install azure cli on Jenkins-vm server

A screenshot of a MobaXterm terminal window titled "172.191.208.216 (jaspreet)". The window shows a command-line session where the user runs a curl command to download and install the Azure CLI. The output of the curl command is displayed in the terminal window, showing various HTTP requests and responses for files from azure.archive.ubuntu.com and pkg.jenkins.io.

```
jaspreet@jenkins-vm:~$ curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash
Hit:1 http://azure.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://azure.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://azure.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://azure.archive.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Ign:5 https://pkg.jenkins.io/debian binary/ InRelease
Hit:6 https://pkg.jenkins.io/debian binary/ Release
Hit:7 https://aquasecurity.github.io/trivy-repo/deb generic InRelease
Hit:8 https://download.docker.com/linux/ubuntu noble InRelease
Get:9 http://azure.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [1060 kB]
Get:10 http://azure.archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [161 kB]
Get:11 http://azure.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1061 kB]
Get:12 http://azure.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [376 kB]
Get:13 http://azure.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Components [212 B]
Get:14 http://azure.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [940 B]
Get:15 http://azure.archive.ubuntu.com/ubuntu noble-backports/main amd64 Components [7064 B]
Get:16 http://azure.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Components [16.3 kB]
Get:17 http://azure.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 Components [212 B]
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.com>

Add az login and docker push stages to jenkinsfile

```

Jenkinsfile
1 pipeline {
2   agent any
3   tools{
4     maven 'maven-proj'
5   }
6   environment {
7     // Define the Docker image name and tag
8     IMAGE_NAME      = "petclinic"
9     IMAGE_TAG       = "${BUILD_NUMBER}" // Use build
10    number as version
11    // Define the Azure Container Registry name
12    ACR_NAME        = "jkjenkins"
13    ACR_LOGIN_SERVER = "${ACR_NAME}.azurecr.io"
14    FULL_IMAGE_NAME = "${ACR_LOGIN_SERVER}/${IMAGE_NAME}:${IMAGE_TAG}"
15  }
16  stages {
17    stage('Checkout from git'){
18      steps{
19        echo "checking out from git"
20        git branch:'master', url:'https://github.com/kaurjk/petclinic.git'
21      }
22    }
23  }
24 }

Dockerfile
1 FROM maven:3.6-jdk-8-alpine
2 MAINTAINER jkjenkins@saultcollege.ca
3 ENV MAVEN_HOME /opt/maven
4 ENV PATH $MAVEN_HOME/bin:$PATH
5 WORKDIR /opt/petclinic
6 COPY . .
7 RUN mvn clean package -DskipTests
8 COPY target/petclinic-1.0-SNAPSHOT.jar /opt/petclinic/petclinic.jar
9 EXPOSE 8080
10 CMD ["java", "-jar", "petclinic.jar"]

```

Kubernetes Deployment

Kubernetes login

Create aks cluster.

Essentials	Networking
Resource group : rg	Kubernetes version : 1.31.7
Power state : Running	API server address : aksjenkins-dns-nulgugox.hcp.eastus.azmk8s.io
Cluster operation status : Succeeded	Network configuration : Azure CNI Overlay
Subscription : Azure for Students	Node pools : 1 node pool
Location : East US	Container registries : Attach a registry
Subscription ID : 4accd697-de5f-43da-85b7-7a90347cd651	
Fleet Manager : Click here to assign	
Tags (edit) : Add tags	

To login to AKS cluster, we need to give the **Managed Identity** the correct role **Azure Kubernetes Service Cluster User Role** on the **AKS resource**, so it can fetch credential.

The screenshot shows the Microsoft Azure Access control (IAM) interface for the 'aksjenkins' service. The left sidebar lists various management categories like Overview, Activity log, and Diagnose and solve problems. The 'Access control (IAM)' section is currently selected. The main content area displays the 'Role assignments' tab, which lists 8 role assignments for the 'aks' subscription. The search bar at the top shows 'aks'. The table below shows one entry:

Name	Type	Role	Scope	Condition
jenkins-vm ed866ac3-63f9-4ae0-8f0d-ec...	Managed identity	Azure Kubernetes Service Cluster User Role	This resource	None

Below the table, it says 'Showing 1 - 1 of 1 results.'

Add or remove favorites by pressing **Ctrl+Shift+F**

Now add stage for logging to azure for getting aks credentials.

Deploy image from ACR to AKS

Create a secret to allow your Kubernetes cluster to pull Docker images from your private Azure Container Registry (ACR).

Get the credentials from acr registry

jkgjenkins | Access keys

Registry name: jkgjenkins

Login server: jkgjenkins.azurecr.io

Admin user: jkgjenkins

Username: jkgjenkins

Name: password

password: 7Yjdb95yoD8yba5s4g1iwwquB8k9kYDpNE9... (Copy to clipboard, Hide, Regenerate)

password2: ***** (Show, Regenerate)

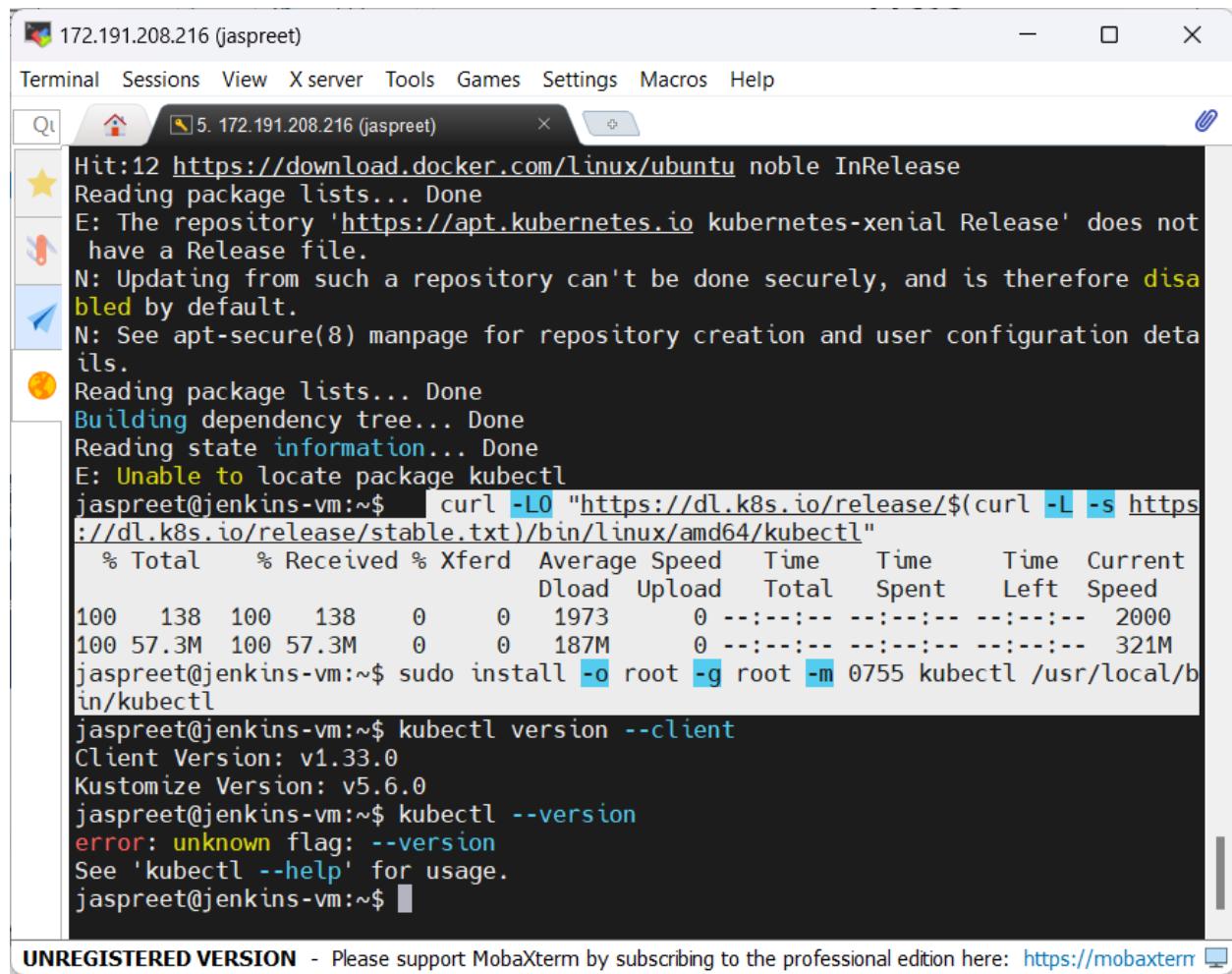
Access keys

- Encryption
- Identity
- Networking
- Microsoft Defender for Cloud
- Properties
- Locks

Services

```
PS /home/gurpreet> kubectl create secret docker-registry acr-auth --docker-server=jkgjenkins.azurecr.io --docker-username=jkgjenkins --docker-password=7Yjdb95yoD8yba5s4g1iwwquB8k9kYDpNE9... --docker-email=jassu249@gmail.com
error: failed to create secret secrets "acr-auth" already exists
PS /home/gurpreet> kubectl get secrets
NAME      TYPE          DATA   AGE
acr-auth  kubernetes.io/dockerconfigjson  1      29m
PS /home/gurpreet>
```

Install kubectl in jenkins vm agent.



172.191.208.216 (jaspreet)

Terminal Sessions View X server Tools Games Settings Macros Help

Hit:12 https://download.docker.com/linux/ubuntu noble InRelease
Reading package lists... Done
E: The repository 'https://apt.kubernetes.io kubernetes-xenial Release' does not have a Release file.
N: Updating from such a repository can't be done securely, and is therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package kubectl
jaspreet@jenkins-vm:~\$ curl -LO "https://dl.k8s.io/release/\$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 138 100 138 0 0 1973 0 --:--:-- --:--:-- --:--:-- 2000
100 57.3M 100 57.3M 0 0 187M 0 --:--:-- --:--:-- --:--:-- 321M
jaspreet@jenkins-vm:~\$ sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
jaspreet@jenkins-vm:~\$ kubectl version --client
Client Version: v1.33.0
Kustomize Version: v5.6.0
jaspreet@jenkins-vm:~\$ kubectl --version
error: unknown flag: --version
See 'kubectl --help' for usage.
jaspreet@jenkins-vm:~\$

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.com>

Write the deployment and service yaml.

```

k8s > ! springboot-deployment.yaml
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   name: petclinic
5   labels:
6     | app: petclinic
7 spec:
8   replicas: 3
9   selector:
10    matchLabels:
11      | app: petclinic
12   template:
13     metadata:
14       labels:
15         | app: petclinic
16     spec:
17       containers:
18         - name: petclinic
19           image: jkgjenkins.azurecr.io/petclinic: _IMAGE_TAG
20       ports:
21         - containerPort: 8080
22       imagePullSecrets:
23         - name: acr-auth
24
25 ...
26 apiVersion: v1

k8s > ! springboot-deployment.yaml
7  spec:
12   template:
16   spec:
24
25 ...
26   apiVersion: v1
27   kind: Service
28   metadata:
29     name: petclinic
30     labels:
31       | app: petclinic
32   spec:
33     type: LoadBalancer
34     ports:
35       - port: 80
36         targetPort: 8080
37     selector:
38       | app: petclinic
39

```

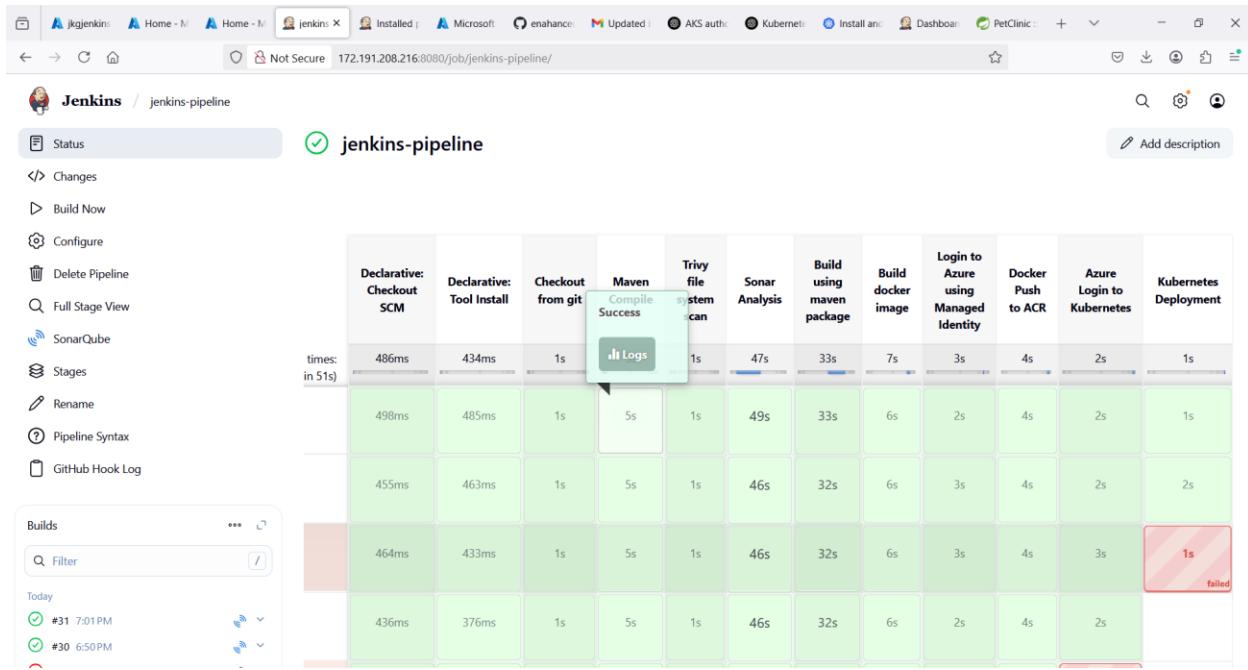
Write pipeline stage for deployment

```

Jenkinsfile
1 pipeline {
2   stages {
3     stage('Kubernetes Deployment') {
4       steps {
5         script {
6           echo "Kubernetes Deployment Stage Started"
7
8           def output = sh(
9             script: "kubectl get deployment ${KBS_DEPLOYMENT} -n $KBS_NAMESPACE --ignore-not-found",
10            returnStdout: true
11          ).trim()
12
13           def deploymentExists = output != ""
14
15           if (deploymentExists) {
16             echo "Deployment exists. Performing rolling update with new image: ${BUILD_NUMBER}"
17             sh """
18               kubectl set image deployment/${KBS_DEPLOYMENT} \
19                 ${KBS_DEPLOYMENT}-${ACR_LOGIN_SERVER}/${IMAGE_NAME}:${BUILD_NUMBER} \
20                 -n $KBS_NAMESPACE
21             """
22           } else {
23             echo "Deployment not found. Creating new deployment from template"
24             sh """
25               sed "s/_IMAGE_TAG_/${BUILD_NUMBER}/" k8s/springboot-deployment.yaml > k8s/tmp-deployment.yaml
26               kubectl apply -f k8s/tmp-deployment.yaml -n $KBS_NAMESPACE
27             """
28           }
29         }
30       }
31     }
32   }
33 }

```

After pushing the code, the Jenkins pipeline ran.



Check the pods and service created.

```
PS /home/gurpreet> kubectl get po
NAME           READY   STATUS    RESTARTS   AGE
petclinic-695c4ff7dc-64nnp   1/1     Running   0          57s
petclinic-695c4ff7dc-8m28l   1/1     Running   0          52s
petclinic-695c4ff7dc-f5btp   1/1     Running   0          55s
PS /home/gurpreet> kubectl get svc
NAME      TYPE        CLUSTER-IP      EXTERNAL-IP      PORT(S)        AGE
kubernetes   ClusterIP   10.0.0.1      <none>           443/TCP       4d17h
petclinic   LoadBalancer  10.0.102.95  134.33.249.25  80:30808/TCP  12m
```

The application is running on External IP provided by the load balancer service.