**Overview**

This document will help you to setup and demonstrate *Jenkins Groovy Pipeline* and use *Trivy* tool to scan for security as a part of the continues integration (CI).

This also demonstrates the CI flow (automatically trigger the CI flow when there is a code change is pushed to the Application code).

**Prerequisites**

* A virtual machine is provisioned with the Jenkins, Docker, Trivy installed from the ***1\_Provision\_Azure\_Resource\_By\_Terraform.docx***
* You are required to configure the Jenkins. Please refer to this guide <https://www.cherryservers.com/blog/how-to-install-jenkins-on-ubuntu-22-04> *(starts from step #6: Set up Jenkins)*. While setup the Jenkins, please make sure the plugins below get installed:
  + Jenkins suggested plugins
  + Docker PipelineVersion
  + Pipeline Utility Steps
  + HTML Publisher
* An Azure Container Registry (ACR) is provided from the ***1\_Provision\_Azure\_Resource\_By\_Terraform.docx***
* Should clone or visit these source codes from GitHub for reference purposes.
  + DevOps CI with Jenkins Groovy: *git clone* [*https://github.com/sieunhantanbao/sd2411-devops-ci.git*](https://github.com/sieunhantanbao/sd2411-devops-ci.git)
  + Application code: *git clone* [*https://github.com/sieunhantanbao/sd2411\_msa.git*](https://github.com/sieunhantanbao/sd2411_msa.git)

**Works details**

1. **Setup Jenkins Organization project**

Login to the Jenkins (i.e. <http://172.173.112.179:8080/>)

* **Step 1**: Configure **GitHub Enterprise Servers**
  + Go to Manage Jenkins -> System -> GitHub Enterprise Servers
  + Add a new GitHub Enterprise Servers as belowA screenshot of a computer

    Description automatically generated
    - API endpoint: <https://api.github.com>
    - Name: *<your GitHub account name>*
* **Step 2:** Configure **Global Pipeline Libraries**
  + Go to Manage Jenkins -> System -> Global Pipeline Libraries
  + Add a new Global Pipeline Libraries as below A screenshot of a computer

    Description automatically generated
    - Name: <any name that you want> // Please note it for later use.
    - Default version: main
    - Project repository: <URL of your git repository that contain the Jenkins Groovy scripts> // i.e. <https://github.com/sieunhantanbao/sd2411-devops-ci.git>
    - Credentials: <your github credentials> which is created in the Manage Jenkins -> Credentials.
* **Step 3**: Create **Organization Folder** project

A screenshot of a computer

Description automatically generated

* **Step 4**: Configure the Organization Folder as below A screenshot of a computer

  Description automatically generated A screenshot of a computer

  Description automatically generated
  + Display Name: <Any name that you want or leave it as blank>
  + API endpoint: select the **GitHub Enterprise Servers** that is created in step 1
  + Credentials: <your github credentials> which is created in the Manage Jenkins -> Credentials.
  + Owner: <your GitHub organization/ GitHub account>

=> After “Save” the Organization Folder project above. From the Jenkins home page:

- Click on: **SD2411 Organisation**

- Click on: **Scan Organization Now** *(from the left menu)*

* **Step 5**: Check the result/output. Confirm the **sd2411\_msa** repository is added to your Organization Folder projectA screenshot of a computer

  Description automatically generated

1. **Jenkins Groovy source code setup**

* From the Application source code (i.e. <https://github.com/sieunhantanbao/sd2411_msa>), in the root of the repository, create a Jenkins file with the following content

A screenshot of a computer program

Description automatically generated

* + The ‘**devops-jenkins-ci**’ is the name of the **Global Pipeline Libraries** that we have created earlier.
  + The ‘**main**’ is the branch name of the devops-ci repository (<https://github.com/sieunhantanbao/sd2411-devops-ci>).
  + The ‘**myPipeline**’ is the name of the Jenkins Groovy file in the devops-ci repository (<https://github.com/sieunhantanbao/sd2411-devops-ci>).
* From the devops-ci repository (i.e. <https://github.com/sieunhantanbao/sd2411-devops-ci>)
  + Source code structure

A screenshot of a computer

Description automatically generated

* + The **myPipeline.groovy** file

A screen shot of a computer program

Description automatically generated

* + - This file will be called from the Jenkins file when having any changes are made to the main/master branches of the application source code.
    - This file will call the nodejs.groovy file to run the stages (details of the steps).
  + The **nodejs.groovy** file

A screen shot of a computer program

Description automatically generated

* + - Full detail can be found here <https://github.com/sieunhantanbao/sd2411-devops-ci/blob/main/vars/nodejs.groovy>
    - This contains the list of build stages
      * Prepare packages: Copy the trivy report template (trivy\_report.tpl) from the **resources/templates** to the **.ci** folder. This template is used for Trivy security and vulnerability reports.
      * Trivy Scan Secret: This stage is to use the Trivy command to scan the security from the application source code, and then export/publish the report to the Jenkins.
      * Trivy Scan Vulnerabilities: This stage is to use the Trivy command to scan the vulnerabilities from the application source code, and then export/publish the report to the Jenkins.
      * Build Backend: This stage is using the docker to build the image for the **backend** service of the application code.
      * Build Frontend: This stage is using the docker to build the image for the **frontend** service of the application code.
      * Push Docker Images to ACR – backend: This stage is pushing the **backend** image to the Azure Container Registry (ACR).
      * Push Docker Images to ACR – frontend: This stage is pushing the **frontend** image to the Azure Container Registry (ACR).
      * Clean up docker images: This stage is to clear all local docker images created in the Build Backend and Build Frontend in the Virtual Machine (build agent).
    - Please be notified that you need to create a Jenkins credential (i.e. **acrcredential**) to allow pushing the docker images to the ACR.
    - **TODO:** This should include some stages for checking Unit Test and SonarQube scan the quality of the application source code.

1. **Setup webhooks**

A screenshot of a computer

Description automatically generated

From the GitHub of the Application Code (i.e. <https://github.com/sieunhantanbao/sd2411_msa>). Click on **Settings** -> **Webhooks**. Then input the URL of the Jenkins (i.e <http://172.173.112.179:8080/github-webhook/>). This is to make sure that if there is any change in this source code, it will make a call to the Jenkins (webhooks) to trigger the CI flow.

1. **Demonstration**

When there is a change in the source code

A screenshot of a computer

Description automatically generated

The Jenkins job is triggered and run successfully

A screenshot of a computer

Description automatically generated

Check the security report

A green line on a white background

Description automatically generated

Check the vulnerability report

A close-up of a computer screen

Description automatically generated

The docker images are published to Azure Container Registry (ACR).

A screenshot of a computer

Description automatically generated