ISEC6000
Secure DevOps

Curtin University

Assignment 3

Secure DevOps Pipeline for Application Deployment

Weighting Percentage: 40% of the unit

Submission: Please follow the descriptions as given in this document

Deadline: 01 November 2025 11.59PM (AWST)

Late Submission: late submission attracts 5% raw penalty per day up to 5 days (i.e., 06 November 2025 11:59PM AWST)

1. Outline

This project provides hands-on experience in designing and implementing a secure, automated DevOps pipeline for a Python-based cloud-native web application. You will implement security and monitoring throughout the pipeline using tools like Clair for vulnerability scanning, AWS CloudWatch for logging and monitoring, and AWS IAM for secure access control. The goal is to ensure a secure and streamlined deployment process with real-time monitoring and automated threat detection.

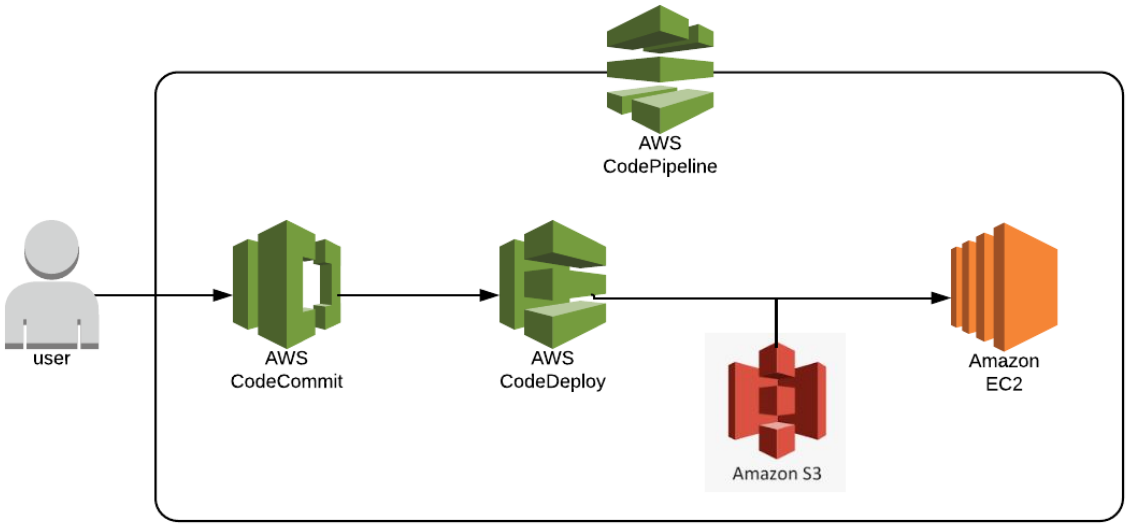

Fig. 1. AWS CodePipeline diagram

Imagine you are part of a DevOps team tasked with building a secure, automated deployment pipeline for a web application. Your organisation wants to ensure every component is secure, from the Docker setup to the deployment pipeline on AWS, whilst following DevOps best practices.

In this project, you are responsible for designing and evidencing a secure, automated deployment pipeline for a cloud-native web application.

You will:

- Implement secure Docker and Compose configurations.
- Deploy Portainer and Nextcloud as managed stacks.
- Integrate container vulnerability scanning with Clair (or any scanner tool of your choice).
- Configure an AWS CodePipeline for CI/CD of a Node.js application.
- Conduct threat modelling using STRIDE.

Security, automation, evidence, and documentation quality are all graded equally.

2. Prerequisites

1. A Linux server running Ubuntu 20.04 or higher with Docker installed. You can use other Linux distributions (e.g., Debian) also. The server should have at least 1GB of RAM, 2 CPU cores, and 20GB of free space (best configuration). You need to work on Linux OS only (no other OS allowed to complete this assignment).

2. A personal GitHub account (You should have this already while completing Assignment 1 and Assignment 2), AWS account.

3. Project Tasks

Task 1: Set Up Docker and Docker Compose [5]

Your team has chosen Docker as the foundation of the application's infrastructure, which will allow for consistent deployments and simplified scaling. However, Docker installations can expose systems to security vulnerabilities, so your task involves setting up Docker securely.

1. Install Docker and Compose (latest versions).
   – Show commands and verify versions (docker --version, docker compose version). (2 marks)
2. Restrict Docker access to *sudo* users only. Demonstrate attempt as non-sudo user fails. (2 marks)
3. Provide a short security summary describing why this restriction matters. (1 mark)

Note: Please note that, you need to provide all the evidence (e.g., commands, screenshots, etc.) and detailed descriptions of each step.

Task 2: Deploy Portainer for Docker Management [12]

Portainer is a lightweight container management platform that was originally developed as a graphical user interface (GUI) for Docker. Over time, it has evolved to support a wide range of containerized environments beyond Docker, providing a unified interface for managing, deploying, and monitoring containers. Portainer simplifies the administration of multiple endpoints and enables teams to collaborate effectively by sharing access to common deployment environments.

In this task, you need to deploy a Portainer stack using Docker Compose. Before beginning the setup, please refer to the provided link for detailed instructions on how to deploy a new stack.

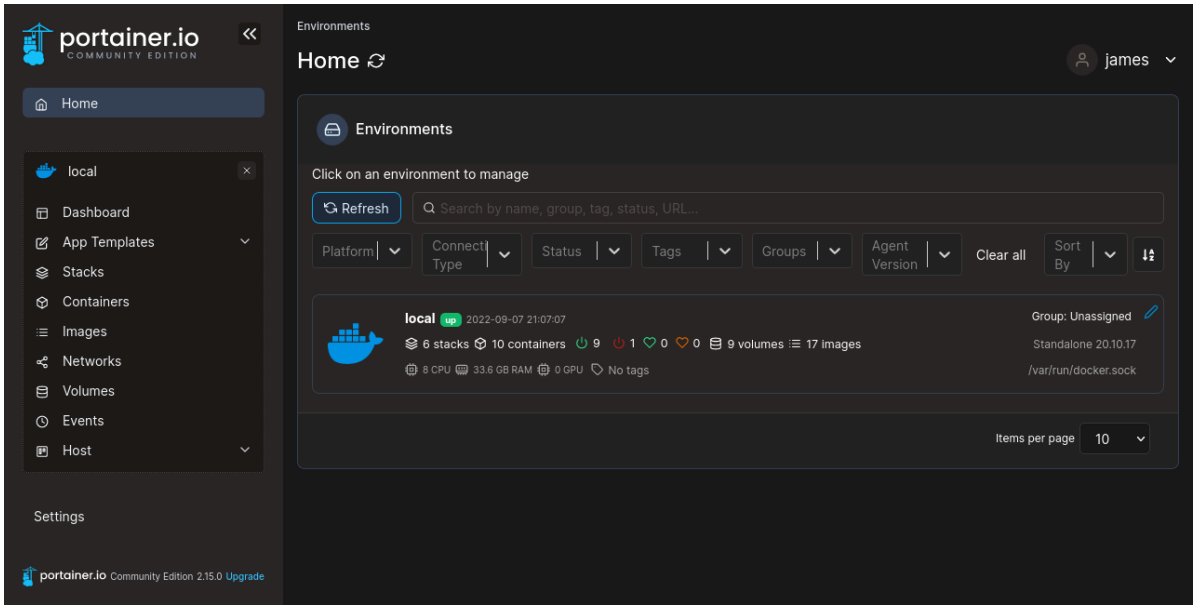

Fig 2. A Sample Portainer Dashboard

To make container management easier, you need to deploy Portainer, a web-based Docker management tool. This task involves configuring Portainer with secure access controls and monitoring logs for security incidents.

Now, please perform the below tasks.

1. Download the **portainer.zip** file from Blackboard under **Assignment 3** (Click the Assessment link on the left pane of Secure DevOps Unit in the Blackboard) and extract the content of the **portainer.zip** to a directory on your host. Inspect `compose.yaml` file. (1 mark)
2. Report the value of `restart:` of the compose file and list the valid options associate with `restart` (with their meaning). (2 marks)
3. Deploy Portainer stack (Include reasonable steps with evidence) (5 marks)
    a. Report the steps with evidence (e.g., screenshots). You must show evidence that atleast one container is running. Additionally, report the detail of the deployed Portainer container (e.g., Status, Id etc).
    b. Navigate in your web browser to access the Portainer web interface and create an account. Don't forget to attach the related screenshot in the report.
4. Access Log Monitoring (4 marks)
    a. Show the *three* most recent lines of logs from the running stack. (Both the Docker-compose CLI command and output should be reported.)
    b. Access and filter Portainer log for any suspicious activities.

Note: Make a proper documentation of all the steps mentioned for this task.

Task 3: Setup a NextCloud stack with PostgreSQL [23]

NextCloud is a free, open-source, and self-hosted cloud storage platform designed as an alternative to proprietary services such as Dropbox and Google Drive. Unlike these

commercial providers - which store user data on external servers, NextCloud allows you to host your own cloud environment, giving you full control over your files and privacy. This platform can be installed on your personal home server or a virtual private server (VPS). With NextCloud, you have the ability to upload your files to your server and subsequently synchronize them with your desktop computer, laptop, or smartphone. This approach grants you complete control over your data.

Your company wants to add NextCloud for secure file storage, integrated with a PostgreSQL database. In this task, your objective is to set up and configure a NextCloud stack integrated with a PostgreSQL database. This setup will provide your organisation with a secure and private file storage system that aligns with best practices in data protection and enterprise collaboration.

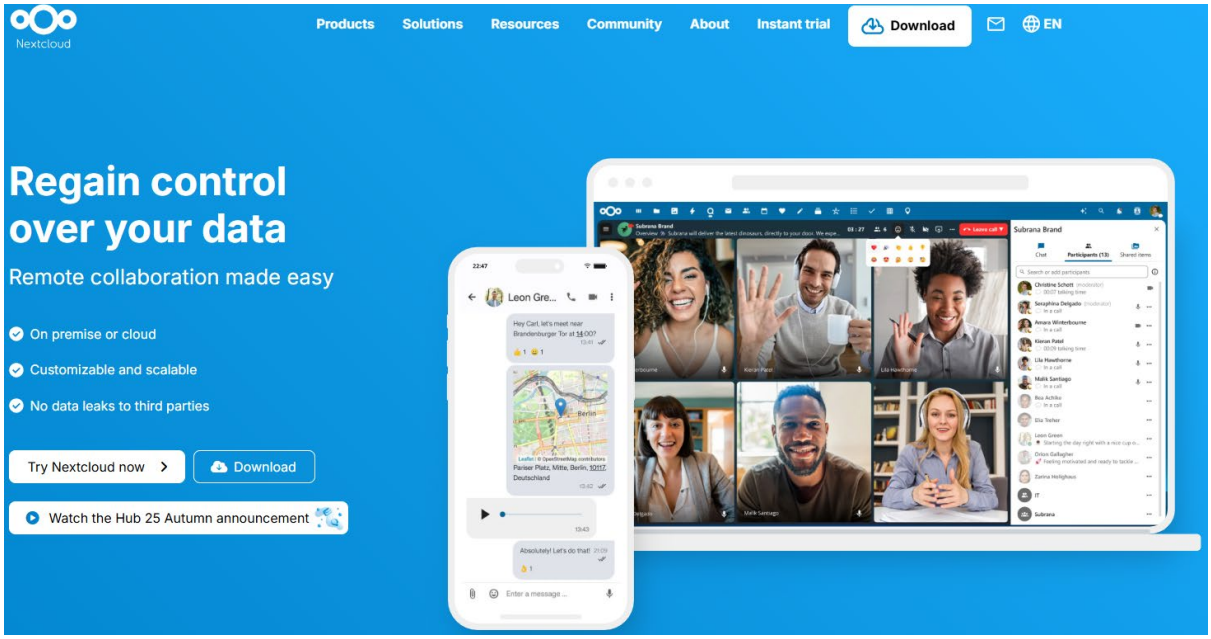

Fig 3. A Sample NextCloud Dashboard

1. Download the **nextcloud.zip** file from Blackboard under **Assignment 3** (Click the Assessment link on the left pane of Secure DevOps Unit in the Blackboard) and extract the content of the **nextcloud.zip** to a directory on your host. inspect `compose.yaml`; explain `expose`: key function. (2 marks)

2. Deploy Nextcloud and PostgreSQL stack with screenshots of successful start-up. (Attach screenshots of all steps in the report including the below sub-tasks). (4 marks)
    a. (i) Show the NextCloud web interface using your web browser, (ii) Report the fetched URL. Attach the related screen in the report.
    b. (i) Show the Portainer web interface using web browser after login to the portainer (refer Task 2). Attach the related screen in the report. (ii) Report the fetched URL. Attach the related screen in the report.

3. Demonstrate CLI management:
    a. Start/stop/restart stack (6 marks)
    b. Categorise and List services with their status (3 marks)
    c. Tail the stack logs (Show output and commands) (2 marks)
    d. Report the Service starting sequence based on the output log. E.g., Service 1: XX, Service 2: YY, and so on. (2 marks)
    e. Check the running containers. Report the relevant information of the running containers. (2 marks)
4. Edit compose so that `db` service starts before `nc`; attach evidence and related discussions. (2 marks)

Note: Make a proper documentation of the steps mentioned for this task.

Task 4: Container Security Scanning with Clair [20]

In this task, you need to set up the Clair security scanner on your server and scanning a NextCloud container for vulnerabilities. Additionally, you will be responsible for documenting the entire process, including any vulnerabilities discovered and how they were mitigated.

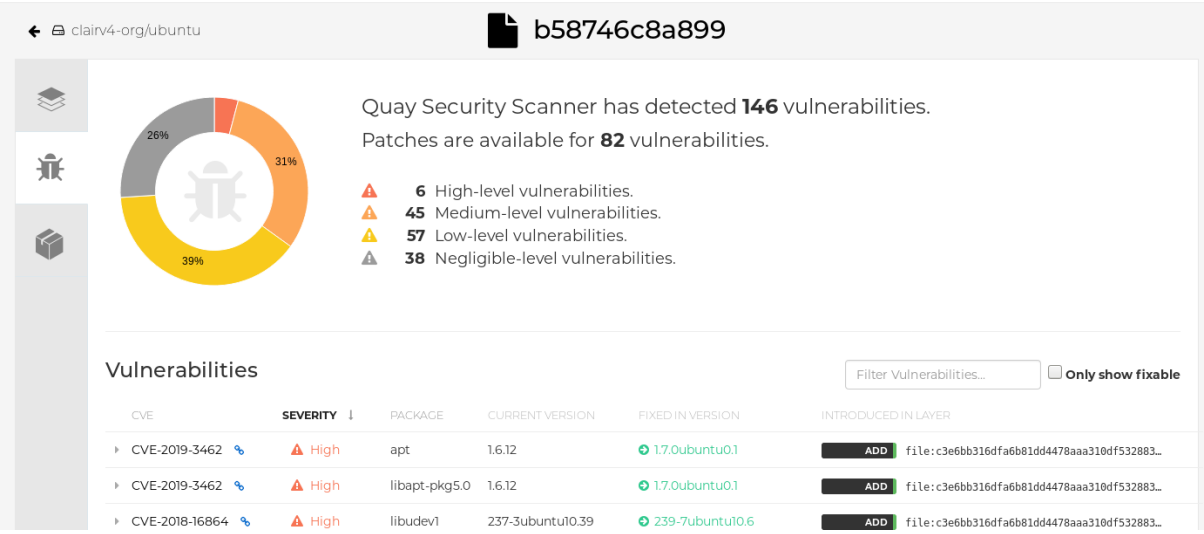

Fig. 4. A Sample Customer Portal of Clair Security Scanning Red Hat Quay

Before starting, make sure that Docker is already installed on the server.

Refer to the official Clair documentation for detailed installation instructions and configuration: Clair Installation Guide. This github link has "The book" which contain all the documentation on Clair's architecture and operation.

1. **Install PostgreSQl and Clair via Docker Compose**: (12 marks)
   A. Install PostgreSQL. Note that Clair uses PostgreSQL as database. (report all the steps)
   B. Initialize a new PostgreSQL database named `clair6000`, specifically for use by Clair
   C. Install and start Clair using Docker Compose: write a complete docker-compose file using the masked scaffold provided (refer "`sample docker-compose.yml`" available in the project folder). This setup should be used for deploying the Clair vulnerability scanner alongside a PostgreSQL database backend.
   D. Start the Clair service: [You need to write command to start the services mentioned in the docker-compose, with evidence]
   E. Verify that Clair is running by accessing its API at port 6063. (URL with screenshot should be added in the report)
2. **Perform Container Security Scanning**: (8 marks)
   In this task, use Clair to scan the NextCloud image, documenting vulnerabilities and categorizing them by severity. You may follow the below steps.
   A. On the server where you deployed your NextCloud stack (the primary server), install the `` `clairctl` `` command-line tool. You can find the stable release at GitHub. Please refer the Clair Documentation page for the usages.
   B. Perform security scan on the NextCloud container Analyse your NextCloud image with Clair (add all the command and the related screenshots).
   C. `` `clairctl` `` will initiate the scan. Provide a report on vulnerabilities found in the container image. (add all the command and the related screenshots)
   D. Review the scan results and take appropriate actions to address any security vulnerabilities. This may include updating the base image, patching dependencies, or configuring your application to mitigate vulnerabilities.

Note: Make a proper documentation of the steps mentioned for this task. If you are getting any error to set up any of the steps, please troubleshoot them accordingly.

Task 5: AWS CodePipeline for Node.js [25]

In this task, you will demonstrate your ability to set up a CI/CD pipeline using AWS CodePipeline for a Node.js application. You will fork an existing GitHub repository, configure a multi-stage pipeline, modify the application code, and grant cross-account access to the lecturer.

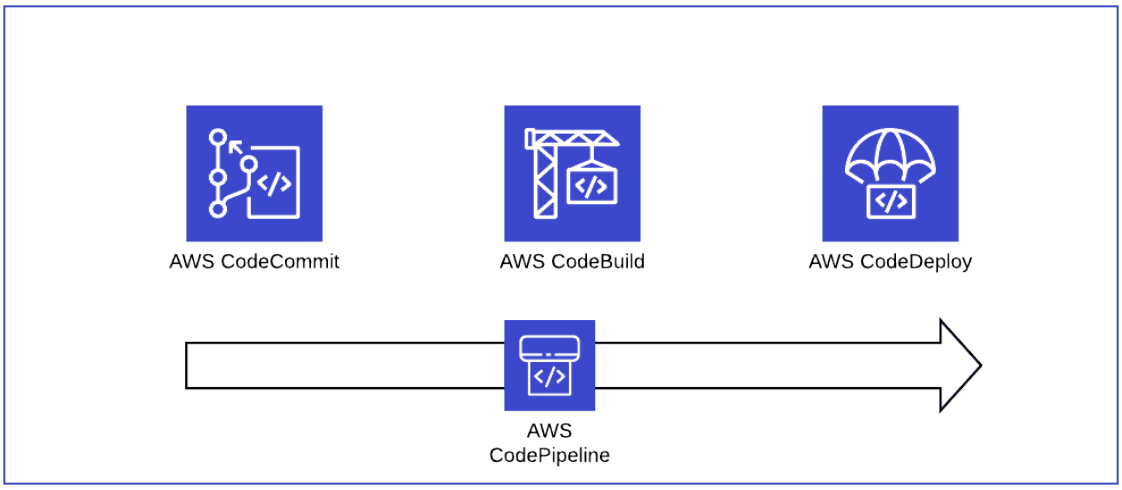

Fig. 5. A Codepipeline example to deploy a NodejS app on AWS

Report the steps in detail and evidence of the below tasks in the submission document.

1. Fork the following GitHub repository into your personal GitHub account: Express-ES6-Sample.
2. Replace the package.json file in your forked repository with the one available here.
3. Create a four-stage pipeline using an AWS CodePipeline.
   Please visit the link for more details.
   - **A. Stage 1: Source** (2 marks)
   - Retrieve the source code from your forked repository.
   - Use OAuth for repository access (do not make the repo public).
   - Configure GitHub Webhooks as the Change Detection option.
   - **B. Stage 2: Build** (2 marks)
   - Build the application
   - **C. Stage 3: Test** (2 marks)
   - Test the application
   - **D. Stage 4: Deploy** (2 marks)
   - Deploy the application to AWS Elastic Beanstalk.
   - Ensure your AWS Beanstalk application is successfully deployed
4. Put evidence in report that AWS Beanstalk application is successfully deployed and verify application reachable. (3 marks)
5. Modify the Application Code:
   A. Modify the source code so that the text reads `My Student ID is xxxxxx. Welcome to Express` (replace the xxxxxx with your student Id). (Screenshot must be in the report). (2 marks)
   B. Push the code change to the forked repository. (2 marks)
   C. The pipeline should automatically detect the change, build, and deploy the updated application to AWS Beanstalk. (Check whether the automatic updating of the change is taking place or not). (3 marks)

6. Create a Cross-Account IAM Role: (7 marks)

For instructor review, you need to create a cross-account IAM role in AWS and provide secure access to your setup while implementing necessary access restrictions.

In this task, Create Cross-Account IAM Role ISEC6000 with AdminAccess for lecturer ID 657973389696; attach role ARN (Amazon Resource Name) and screenshot. Please detail all the steps with relevant screenshots.

NOTE: You will need to take screenshots of your AWS Pipeline setup. You must ensure that your screenshots have sufficient details for the grader to grade your assessment. Make a proper documentation of the steps mentioned for this task.

Task 6: Threat Modelling using STRIDE [10]

Use the Microsoft Threat Modelling Tool to create a basic threat model for a web application with three components: Database, Web Application, and User Interface. Apply the STRIDE methodology (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege) to identify potential threats and propose mitigation strategies.

1. Create a New Model:
    a. Set up the Microsoft Threat Modelling Tool (TMT) (or, any other tool such as threat dragon) and create a project called WebApp Threat Model.
2. Map the System Components:
    a. Add components: Database, Web App, and User Interface.
    b. Define data flows, such as user data requests from the interface to the database through the web application.
3. Identify Threats Using STRIDE:
    a. For each component, identify a potential threat in each STRIDE category (e.g., Spoofing: unauthorized access).
4. Document and Mitigate Threats:
    a. Provide a simple mitigation strategy for each threat.
    b. Generate and review a report in the tool summarizing identified threats and mitigations.

Note: you can refer to Packt's article on using the Microsoft Threat Modelling Tool to have some idea on this task.

4. Submission Guidelines

You need to submit a pdf document containing the tasks as mentioned. The documentation should be **logically organized**, providing **clear instructions/steps** with **screenshots** for each pivotal task. The pdf file name should be in this format: **P3_StudentID**. For example, if your student ID is 123456, then the filename should be **P3_123456.pdf**. (Note: Fail to these

instructions will result in a **penalty of 50%**) (Please strictly follow the instructions in this assignment submission).

The submission (e.g., **P3_123456.pdf**) MUST have cover page (the first page) containing the Full Name of the student, student id.

Note: The documentations (e.g., **P3_123456.pdf**) should be presented in a way such that all the tasks have detailed steps written and have related screenshots and evidence. (**5 marks**)

Detailed documentation, clear explanations, and adherence to project guidelines are essential for successful submission.

The report should contain:

- **Cover Page**: Include the Full Name of the student, Student ID, and links to the relevant Links of GitHub repositories (if any).
- **Detailed Documentation for Each Task**: The report should contain step-by-step documentation for each task, clearly explaining the setup and configuration process, etc.
- **Screenshots of the steps as evidence**: Include screenshots of the tasks as proper evidence.
- **Structure and Clarity:**
  Ensure that the documentation is clear, concise, and logically organized.
  The PDF file should be named in the format P3_StudentID.pdf (e.g., P3_123456.pdf).

Note that, you should submit **ONLY** one document (e.g., P3_123456.pdf) and all the related documentation of the Tasks should be there. Each task category should have a proper section and Headings. You are encouraged to create Bookmark of the document. Please follow the *basic* structure of the submission as shown below.

Cover Page
Task 1
Task 2
Task 3
Task 4

Detailed documentation, clear explanations, and adherence to project guidelines are essential for successful submission.