## APPENDIX I - CAPSTONE PROJECT

### Brief

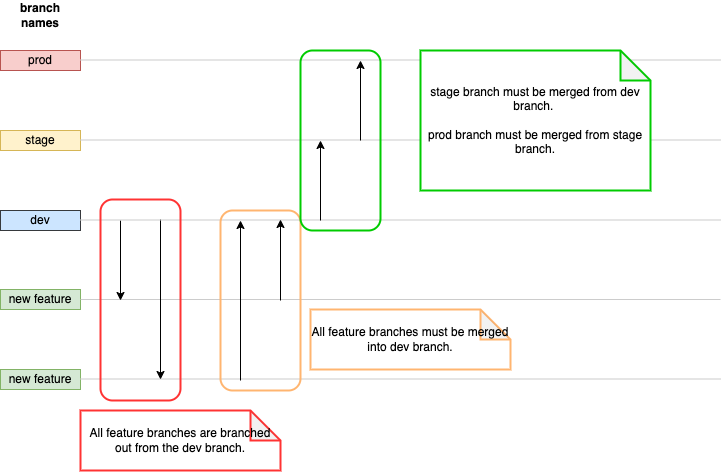
These capstone projects are designed to enable learners to familiarize with cloud technologies and their use cases such as DevOps, DevSecOps and Site Reliability. Each use case focuses on different aspects of software deployments. A team size of 3 or 4 pax is expected to undertake the project.

### 

### Use Case 1 - CICD Pipeline Focused (DevOps)

You are working in a new startup. The team is newly formed with about 5 software engineers and 2 devops engineers. You are tasked to automate the deployment process so that each release cycle can be released quickly from the development environment, to staging and to production environment.

As the company is going at a fast paced growth rhythm, there will be more engineers onboarding in the next few quarters. The tech lead needs to ensure that as the team size grows, the release pipeline is well controlled. Keeping with these demands in mind, you are to build a DevOps cycle with this branching strategy:



**Project Completion Criterias:**

* No microservices required.
* Application must be deployed and running.
* Branch permissions must be set.
* CICD Script written for each affected branch.
* A complete workflow from PR to dev branch to deployment in production environment.

### Use Case 2 - Monitoring Large Application (SRE - Site Reliability Engineering)

You are working in a large company that develops enterprise software solutions. The solutions are implemented with the microservices architecture. With it, comes manifold complexity in systems’ health monitoring.

There is urgency to detect and flag out systems and applications failure early, provide traceable logs to respective engineers for troubleshooting. Your role as a Site Reliability Engineer is to provide monitoring capabilities that are observable and traceable to stakeholders.

**Project Completion Criterias:**

* No CICD Required
* Only one environment suffix.
* Application logging must be centrally managed.
* Systems health must be centrally managed.
* Produce a dashboard that shows applications and systems logs.

### 

### 

### Use Case 3 - Security Focused (DevSecOps)

You are working in a unicorn startup and they are rapidly scaling their products to markets in several regions. Due to the market competition, security vulnerabilities screening has always been hindering the product release. You are tasked to turn the current DevOps Pipeline into DevSecOps. This means that you are to implement relevant security measures at each stage of the CICD Pipeline. In this project, you will be required to set up a simple CICD pipeline while focusing on the security measures introduced to it.



**Project Assumptions:**

* The “Development” branch is accessible by all developers, but not staging and production.

**Project Completion Criterias:**

* A typical CICD Pipeline.
* Implement dependency screening in CI Script.
* Ensure proper authentication and authorization in each environment in CD Script. (e.g. credentials used for deploying development and production environments should not be the same)
* Proper handling of CICD Pipeline Secrets.

## APPENDIX II - Capstone Project Details

### 

### Use Case 1 - CICD Pipeline Focused (DevOps)

1. Create New Project

Create a new repository on GitHub and give public access and correct access to the group members as collaborators.

1. Add Backend application

The backend application should be containerized to ensure portability and the ability to deploy on any platform. Docker can be used to containerize the backend application. Sample backend applications that have been taught before can be used or any backend application available all over the internet.

1. Define the branching strategy & Set branch permissions

The branching strategy should be defined. A simple branching strategy like GitFlow is recommended since the development code can be kept separate from the production code, and features and bug fixes can be worked on without disrupting the production environment.

1. Write CICD script

CICD script should be written for each affected branch to automate the build, test, and deployment process. The Docker images can be built, tests can be run, and the application can be deployed to the staging environment using the CICD script. Once it passes all tests, it can be promoted to the production environment.

1. Implement a complete workflow

A complete workflow needs to be implemented. A CICD tool, Github Action or any others such as Jenkins, Travis CI, or CircleCI can be used to start the workflow. A feature branch can be created from the development branch, and a pull request can be created once the work is completed, triggering the CICD pipeline.

The pipeline should be able to build, test, and deploy the application to the development environment. Once approved, the changes can be merged into the main branch and deployed to the staging environment automatically. After passing all tests, the changes can be promoted to the production environment.

1. Well documented code.

Well documented documentation: In addition to the steps mentioned above, it's important to have well-documented documentation that explains the project's structure, dependencies, and how to set up the development environment. This documentation should be written in the README.md file and supplemented with diagrams, screenshots, and other relevant materials to help the development team understand the project's architecture and workflow. It should also include guidelines for troubleshooting and debugging issues that may arise during development or deployment. This will help new team members onboard quickly and ensure that everyone has a clear understanding of the project's objectives and how it works.

### Use Case 2 - Monitoring Large Application (SRE - Site Reliability Engineering)

A comprehensive monitoring solution that can detect and flag out systems and applications failure early would be built as the first step as a Site Reliability Engineer. Here's how the project would be approached with the given completion criteria:

1. Create New Project

Create a new repository on GitHub and give public access and correct access to the group members as collaborators.

1. Add Containerize Application

The application should be containerized to ensure portability and the ability to deploy on any platform. Docker can be used to containerize the backend application. Sample backend applications that have been taught before can be used or any backend application available all over the internet.

Application is deployed to AWS.

1. Application logging

Application logging would be centrally managed using a logging aggregator to collect and analyse logs from all the microservices and provide a centralised logging solution.

Systems health would be centrally managed using a monitoring tool to monitor the health of all the microservices and alert us in case of any issues.

1. Application Monitoring Dashboard

A dashboard that shows application and system logs would be produced which would allow us to create custom dashboards that show logs and metrics from all the microservices and systems.

The solution would be deployed with one environment suffix using a container orchestration tool such as Kubernetes. Kubernetes would allow the microservices and the monitoring solution to be deployed on a single cluster and provide automatic scaling and load balancing.

By following these steps, a comprehensive monitoring solution that is observable and traceable to stakeholders would be created. Systems and applications failure could be detected and flagged out early, traceable logs could be provided to respective engineers for troubleshooting, and a dashboard that shows application and system logs could be provided.

### Use Case 3 - Security Focused (DevSecOps)

1. Create New Project

Create a new repository on GitHub and give public access and correct access to the group members as collaborators.

1. Add Containerize Application

The application should be containerized to ensure portability and the ability to deploy on any platform. Docker can be used to containerize the backend application. Sample backend applications that have been taught before can be used or any backend application available all over the internet.

Application is deployed to AWS.

1. Write CICD script

CICD script should be written for each affected branch to automate the build, test, and deployment process. The Docker images can be built, tests can be run, and the application can be deployed to the staging environment using the CICD script. Once it passes all tests, it can be promoted to the production environment.

Proper authentication and authorization are to be ensured in each environment in the CD Script, with the condition that credentials used for deploying development and production environments should not be the same.

Proper handling of CICD Pipeline Secrets is to be carried out.

* Master Branch is secured
* Work with Pull Request & Merging
* Create new Role on AWS and use the secret into Github, Not User
* Get Key from AWS

## APPENDIX III

## Capstone Project Use Case 1 Documentation

### 

### **Company Profile**

At TechNova Innovations, our tech team is the backbone of our company, comprised of exceptional individuals who are passionate about technology and dedicated to driving innovation. Each member brings a unique set of skills and expertise, creating a collaborative environment where ideas flourish and breakthrough solutions are developed. Here's a glimpse into the roles and responsibilities of our talented tech team:

### **Software Engineering Team**

Our software engineers are the architects behind our innovative products. They leverage their expertise in programming languages, such as Python, Java, and C++, to design, develop, and implement robust and scalable software solutions. Their skills range from front-end development for intuitive user interfaces to back-end systems that power our products' functionality.

### **Project**

Project Name: Animated Train

Repository: <https://github.com/del-skillsunion/animated-train>

The Software Engineering team at TechNova Innovations is currently working on an exciting project called "Animated Train." This project involves the development of a software application website that implements a serverless architecture.

### **Getting started**

To get the Node server running locally:

* Clone this repo
* npm install to install all required dependencies
* Install MongoDB Community Edition ([instructions](https://docs.mongodb.com/manual/installation/#tutorials)) and run it by executing mongod
* npm run dev to start the local server

## **Dependencies**

* [expressjs](https://github.com/expressjs/express) - The server for handling and routing HTTP requests
* serverless - Serverless application

## **Application Structure**

* index.js - The entry point to our application. This file defines our express server and connects it to MongoDB using mongoose. It also requires the routes and models we'll be using in the application.

## **Branching Strategies**

**Production Branch**

<https://github.com/del-skillsunion/animated-train/main>

* The main branch, also known as the master branch, represents the production-ready state of the application.
* It contains stable and thoroughly tested code that is ready to be deployed to the live environment.
* Only fully reviewed and approved code changes are merged into the main branch.
* It is typically protected, meaning that direct commits or modifications are restricted, and changes can only be introduced through pull requests after thorough code review and testing.

**Development Branch**

<https://github.com/del-skillsunion/animated-train/tree/dev>

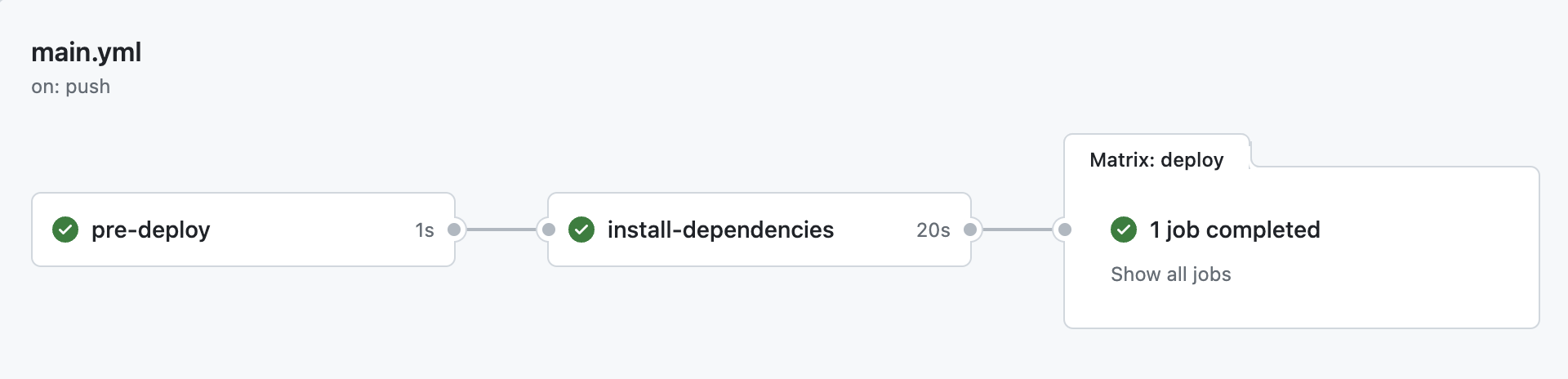
* The dev branch, short for development branch, serves as the primary integration branch for ongoing development work.
* It acts as a staging area for features and bug fixes before they are merged into the main branch.
* Developers regularly merge their completed feature branches into the dev branch for integration testing and collaboration.
* Continuous integration practices are often implemented on the dev branch, allowing automated testing and verification of code changes.

**Feature Branch**

<https://github.com/del-skillsunion/animated-train/tree/feature-1>

* Feature branches are created by developers to work on specific features or bug fixes independently.
* Each feature branch represents a self-contained task or feature development.
* Developers work on their feature branches locally, implementing and testing their changes.
* Once the feature is completed and tested, it is merged into the dev branch for further integration.
* Feature branches are short-lived and are eventually deleted after merging into the dev branch to maintain a clean and manageable codebase.

## **CICD Pipeline**



pre-deploy to check the application

npm install to install all required dependencies

deploy to deploy application to production

END