SE489 DevOps Engineering

Course Project

**SE489: DevOps Engineering – Course Project**

The goal of this project is to apply all the principles, techniques, and best practices of DevOps in a real project. You will demonstrate your DevOps engineer skills to release cycles and high-quality products. You will deliver value continuously by building an automated CI/CD pipeline. Those new skills will also be put to the test as you set up automated monitoring and alerting to ensure the delivered value stays valuable.

In this project, you will continue your work on operationalizing Microservices by deploying an elastic and fault-tolerant software application using Kubernetes. You will configure this Microservice to be highly available by using Kubernetes best practices. You will validate your design by load testing the service and verifying the application architecture performs as designed.

The purpose of this project is to give you a chance to combine what you have learned throughout the course. Each team will be given a chance to choose their project. You will build a CI/CD pipeline for a Microservices application for different deployment strategies. Students define the scope of the project and select the right deployment strategy based on different business requirements.

**Suggested Tools:**

* Git – For version control for tracking changes in the code files
* Maven – For software packaging
* Jenkins – For continuous integration and continuous deployment
* Docker – For container image which is a lightweight, executable package of the software which includes everything needed to run the image (eg. code, libraries, etc)
* Puppet - Open-source software configuration management tool
* Nagios - Application monitoring tool

**Suggested Hints:**

The course project is going to focus on open-source tools. For a quality course project, you need to ensure the following:

* The project strictly follows the DRY principle: everything that must appear in multiple places in the project has a single source.
* Version control should be used for every part of the project, including the source code, build scripts, graphics, documentation, and everything else.
* Make sure to test changes locally before committing.
* The project should have automated testing with over 50% test coverage, and, if web-based, has Selenium tests.
* All of your code should pass a static analysis tool for the language you are using.
* There is an automated build process for any built project components.
* Use a Docker container for development, and deploys can be done via a container. Furthermore, scripts should build the containers automatically.
* The project should be deployed to the cloud with monitoring capabilities.

The project must be managed in a DevOps scenario. In particular:

* Appropriate versioning and licensing
* Appropriate strategy for version control (e.g., rebasing policy)
* Full build automation and full continuous integration
* Automated quality assurance: code quality, testing, reporting
* Automatic delivery, possibly, automated deployment
* When applicable, a containerized version of the application

**GitHub Links for similar Projects:**

<https://github.com/PacktPublishing/hello-world>

<https://github.com/yankils/Simple-DevOps-Project>

<https://github.com/ksemaev/project_template>

<https://github.com/Abhinav-26/DevOps-Projects>

<https://github.com/johnbedeir/Devops-Tools-Documentation>

<https://github.com/icetlab/devopscourse>

<https://github.com/awsmug/wp-devops>

**Project Phases:**

*Phase One* (25%):

* The team members' names and the team structure
* The selected project, its description, its main architecture and design, and GitHub link
* A Brief Overview of the Project, its features, and your plan for enhancements, features, and quality improvement
* Communication medium used between team members
* A brief description of your planned Pipeline with proper justification of your selection
* Screenshots of a running version of your application with brief description of these screenshots

|  |  |
| --- | --- |
| Criteria | Weight |
| Team Names and Team Structure | 15% |
| Project Info and Details | 30% |
| Project Features and Quality Enhancement | 20% |
| Communication medium | 10% |
| Planned Pipeline | 20% |
| Screenshots of a running version of your application | 5% |

*Phase Two* (35%):

* + Run tests locally, as well as push code into production
  + Keep developing your application
  + Testing: you should be writing tests as you write your code. Proper code styling should be enforced
  + Set up a CI/CD pipeline. Use it to deploy your application.
  + Ensure proper monitoring of your pipeline.

|  |  |
| --- | --- |
| Criteria | Weight |
| Code Quality | 20% |
| Automated Tests | 25% |
| Clear Pipeline Structure | 25% |
| Deployed Application | 20% |
| Monitoring Mechanism | 10% |

*Phase Three* (40%):

* + Another cycle of your pipeline (New Deployment with a new feature or bug fix).
  + Final project report and presentations.

|  |  |
| --- | --- |
| Criteria | Weight |
| Another Cycle of the Pipeline | 20% |
| Monitoring Reports | 15% |
| Excellent Pipeline Structure | 20% |
| Final Project Report | 25% |
| Presentation | 20% |

**Project delivery**

* The project must be made available on a dedicated GitHub public repository
* This repository must contain:
  + Source code of the project
  + DevOps setup related to the project
  + Project report (ideally as a set of linked markdown documents starting from the repository README)
* The Project Report must describe the project rationale and must contain all artifacts related to the problem analysis and the design phases

**Project discussion**

* The project will be discussed during an oral exam
* The oral exam will be organized as follows:
  + Students must show their project in a 5 minutes per student (maximum 15 minutes) long presentation (possibly using slides or other material)
  + The contribution of each team member must emerge from the presentation
  + Questions can be asked related to the proposed project, considering the presentation, the project report and the developed code