| | |
|---|---|
| **Module:** | **DevOps**<br>Out of Class Assessment |
| **Hand-out Date:** | **1st March 2024** |
| **Hand-In Date:** | Each student will present their project to me during class Tuesday or Friday on the week ending: **28th April 2024** |
| **Lecturers:** | Padraic Moriarty (padraic.moriarty@mtu.ie) |

**Assignment:**

1. Build an automated deployment pipeline for an application that you have built or sourced (must be fully referenced and acknowledged) this may be your final year project but clear this with me and your supervisor first.

   OR

2. Research and write a detailed document about a specific tool or technology employed in a DevOps environment. This may be any tool or technology of your choice but the technical manual must approach the level of a technical manual produced by a professional organisation. Examples of tools are; Ansible, Chef, Puppet, Terraform, Kubernetes, Docker Swarm, Rancher, Nomad, Openshift, Airflow, Azure Cyclecloud, AWS Fargate etc.

**Project Objectives:**

The objective is to automatically compile, test, assemble, and deploy your application to at least one environment, as the project progresses through the software development life cycle (SDLC). Your deployment should demonstrate as many of the topics covered in theory and practicals for example CI, CD, Metrics, Notifications, Quality Gates etc

For application deployment you must consider:

- source code repositories,
- build tools,
- continuous integration,
- configuration management to setup runtime environment,
- resource provisioning in the cloud and containers,
- continuous delivery,
- continuous deployment,
- continuous monitoring,
- continuous feedback,
- continuous improvement
- continuous innovation.

**Technologies:**

You may use open-source technologies, including NetBeans, Git, Maven, JUnit, Jenkins Tomcat, Glassfish or proprietary technologies such as Visual Studio, Team Foundation Server, MS Build, Thought works, IIS, Team City. Cloud platforms/virtual platforms such as AWS, Azure, VMWare, Virtual Box, Heroku as appropriate for your application.

You should complete the project as a series of stages, we will investigate a different stage each week during our lab sessions building to a fully operational configuration.

**Submission: Implementation & Documentation**

You will be graded under the criteria outlined below. Therefore your implementation and document should both contain the following sections.

1. *Front Matter (Title page, TOC etc. Document Only)*
2. Introduction code and tools
3. Continuous Integration
4. Building the Code and Configuring the Build Pipeline
5. End-to-end automation of the application delivery lifecycle
6. Provisioning (Cloud) and Configuration Management
7. Deploying Application
8. Monitoring Infrastructure and Applications
9. Orchestrating Application Deployment
10. *Conclusions (Document Only)*

Note: steps 6-9 (see below, stages 5-8) are advanced topics, for a distinction grade you will need to get some of these steps implemented. Pass grades are possible with implementations only reaching step 4 or 5

You are required to submit a document detailing your implementation. This will consist of screen shots showing the configuration and some text explaining the deployment. The sections in the report should match those above.

**Breakdown of Stages:**

The following sections serve as a breakdown of each section. Your implementation may differ in terms of technologies or you may add or exclude certain items. However, the main body of your implementation should follow the format/stages identified below.

**Stage 1 – Code and tools**

1. Once you have identified your application you should write a summary of what the application does where it was sourced and the technologies used i.e. Java, Spring etc. This is the introduction to your report.
2. Identify the tools and technologies that you intend to use in your assignment (Tool Chain). You will add to this list as the semester progresses. We covered and will cover a brief overview of SVN, Git, Apache Maven, Jenkins, AWS, Microsoft Azure and other DevOps technologies in the labs and theory classes.
3. Draw a flow diagram of the pipeline, for each stage, highlight the technologies that will be implemented at each stage.

**Stage 2 – Continuous Integration**

In this stage you are required to install and/or configure and implement continuous integration using a sample application. This application should be stored in a repository such as GitHub.

1. Look at how to automate the build process from a repository (Git) using a tool set i.e. Maven, Gradle, MSBuild etc.
2. When you create a new job in your CI server it should authenticate with Git, you should configure Git in your CI server.
3. Then perform unit tests execution in your sample application.
4. Configure a standard Dashboard View plugin with different portlets for customized views. Configure notifications (for example email) for build status.
5. Look at code quality tools such as Sonar Cloud and implement quality gates.

**Stage 3 - Building the Code and Configuring the Build Pipeline**

This stage is a refinement from stage 2. Many CI servers have built-in support for delivery pipelines (i.e. Jenkins). You will configure your CI server for continuous delivery or continuous deployment. Automation and orchestration both are equally important while dealing with the application delivery pipeline.

1. Create the pipelines of different jobs for your sample application (Java, C# etc.).
2. Deploy the application to a web or application server as appropriate.
3. Configuration of a build pipeline for the lifecycle of continuous integration.

**Stage 4 - End-to-end automation of the application delivery lifecycle**

In the labs we will briefly look at **Chef** and Ansible, you may use other tools if you wish. You can use these for setting up your runtime environment and standardising the process of configuration management rather than implementing a customised way to install tools using scripts. Centralised configuration management makes it easy to control and configure resources without complication.

As an advanced topic we will also look at container technology. We will concentrate on Docker (DockerHub) and its installation and configuration details; we will also cover how to create CentOS containers for application deployment. Using containerisation will achieve a higher grade.

**Stage 5- Cloud Provisioning and Configuration Management**

You now understand the basics of configuration management and containers, so we can start with resource provisioning in a cloud environment and installing the runtime environment required to run your sample application.

**Stage 6 - Deploying Application (AWS, Azure, and Docker)**

You should describe in detail all the steps required to deploy your sample application to a different environment once the configuration management tool prepares it for the final deployment. You may also deploy the application in different environments, such as cloud or container-based ones.

**Stage 7 - Monitoring Infrastructure and Applications**

You should describe the need for continuous monitoring and its significance in the end to end automation process. It covers different aspects of monitoring such as cloud resources, application server and application monitoring to increase services and application availability.

**Step 8 - Orchestrating Application Deployment**

You need to manage all those build jobs in a manner that the checkout or execution of the build pipeline will result in the checkout, compilation, unit test execution, installation of the instance (e.g. Linux on Amazon EC2), of the runtime environment, configuration of permissions in the newly created instance, and deployment of the application.