**Coursework 2**

**DevOps from Code to Application Deployment**

In this coursework you will implement the complete DevOps pipeline using all of the tools you have learned in this module. The goal is to develop a system similar to the diagram below:

A picture containing object

Description automatically generated

Figure 1 - DevOps Pipeline Overview

There are two components to this Coursework:

* **Report – 50%**
* **Practical Demonstration – 50%**

**Submission:**

There will be two submission portals on GCULearn:

* Report - Submitted via Turnitin Assignment ‘DevOps Coursework 2 Report Submission’
* Video - Submitted via GCULearn Assignment ‘DevOps Coursework 2 Video Submission’

**Both components of this coursework should be submitted to GCULearn no later than:**

**Friday 15th December**

**2. Practical**

**2.1 Guidance**

For the practical element of this coursework you are asked to implement a DevOps Continuous Delivery pipeline as outlined in **Figure 1**.

You should complete the Preliminary Steps **(see 2.2)** before proceeding.

The table below **(2.3 Practical Tasks)** provides an outline of the steps required to complete the practical element of this coursework along with the marks available for each component. The code for this coursework is provided in section **2.4 Project Code.**

Marks indicated are for a complete working solution to the task and proportionate marks will be awarded for partial solutions.

Upon completion of these tasks, you should record a demonstration video of this pipeline following the guidance **(see 2.5 Practical Evidence, 2.6 Video Demonstration)** in this specification.

**Please read section 2.5 Practical Evidence carefully before completing the practical components of this Coursework.**

**2.2 Preliminary Steps**

For this coursework you will should;

1. Create two EC2 Instances on AWS with the following parameters:

Instance 1

* Name = Build Server
  + Size = t2.large
  + OS = Ubuntu
  + Ports = 8080 open

Instance 2

* Name = Production Server
  + Size = t2.large
  + OS = Ubuntu

1. Prepare for the practical Coursework task by:
   * Appropriately configuring Git and GitHub authentication on your Build Server
   * Adding the server.js file containing the project code **(see 2.4)** to your local and remote git repositories
   * Installing Docker and Jenkins on the Build Server

**2.3 Practical Tasks**

The table below provides an outline of the steps required to complete the practical element of this coursework along with the marks available for each component. Marks indicated are for a complete working solution to the task and proportionate marks will be awarded for partial solutions.

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** |  | **Description** | **Marks** |
| **1** | **(a)**  **(b)** | **Building the Application**  Create an appropriate Dockerfile for the code you have been provided to create a node-js web application and push this to GitHub.  Use this Dockerfile to manually create a Docker Image for this project and push this to DockerHub. | **3**  **2** |
| **2** | **(a)**  **(b)**  **(c)**  **(d)**  **(e)** | **Preparing the Production Server**  On the Production Server, write and execute Ansible playbook(s) to;   * Install Kubectl * Install and Start Minikube * Deploy your image from DockerHub to Kubernetes * Create a Service for your application * Scale your deployment appropriately to allow for Rolling Updates to be implemented in the future   **This section can be completed without the use of Ansible for up to 50% of the available marks.** | **4**  **2**  **2**  **2**  **2** |
| **3** | **(a)**  **(b)**  **(c)**  **(d)**  **(e)** | **Configuring CD Pipeline**  Install and appropriately configure Jenkins to automatically:   * Detect changes to your GitHub Repository * Build a Docker Image from your Dockerfile * Test that a Container can be launched from your Image (i.e. Build Test – run a command inside the container to ensure that it has launched successfully. You do not need to do any functional testing here) * Push your Image to DockerHub * Deploy passed builds to Kubernetes without disrupting service   **A further 15 Marks is available in this section for completing tasks (b) - (e) using a Jenkins Pipeline Project and associated Jenkinsfile. These marks are indicated in brackets.** | **2**  **4 (2)**  **5 (4)**  **2 (4)**  **5 (5)** |

**2.4 Project Code**

**server.js**

|  |
| --- |
| var http = require('http');  var requests=0;  var podname= process.env.HOSTNAME;  var startTime;  var host;  var handleRequest = function(request, response) {  response.setHeader('Content-Type', 'text/plain');  response.writeHead(200);  response.write("DevOps Coursework 2! | Running on: ");  response.write(host);  response.end(" | v=0\n");  console.log("Running On:" ,host, "| Total Requests:", ++requests,"| App Uptime:", (new Date() - startTime)/1000 , "seconds", "| Log Time:",new Date());}  var www = http.createServer(handleRequest);  www.listen(8080,function () {  startTime = new Date();;  host = process.env.HOSTNAME;  console.log ("Started At:",startTime, "| Running On: " ,host, "\n" );  }); |

**2.5 Practical Evidence**

There are **FIVE** pieces of evidence required for the practical component of this Coursework which should be provided **in the Appendices of your Report**. Failure to provide **ANY** of these may result in marks not being awarded.

1. **Docker Evidence**

You should provide the Dockerfile that you have written and a screenshot of your DockerHub showing that the Image has been pushed.

1. **Ansible Evidence**

You should provide a screenshot of your Terminal showing that all stages of each of your Ansible playbooks pass. If you do not provide this evidence you will not be awarded the full marks based on the playbook alone.

1. **All Ansible Playbooks**

Include all Ansible Playbooks you have created

1. **Completed Jenkinsfile**

Include your completed Jenkinsfile if you created one.

1. **Jenkins Console Output**

You should include a copy of the **FULL** Jenkins Console Output for your final build. This is better copy/pasted as text than as a screenshot

**2.6 Video Demonstration**

You should create a short video demonstration of your Continuous Integration Pipeline (see **2.7 Recording your Video.**)

Your video should be no longer than 5 minutes in duration and **should be narrated** such that it is clear to the viewer that you understand the steps being carried out.

The video should be recorded **after** all of the practical tasks in **Section 2.3** have been completed.

**Before you start the video** you should ensure that your pipeline is ready.

During your video you should:

* Use the ‘curl $(minikube ip)’ command on the Production Server to show the initial version of the application from Task 2 is running correctly.
* Update the code line ‘response.end(" | v=0\n");’ in server.js to read ‘response.end(" | v=1\n");’ and push this change to GitHub. If you have updated this value already during your development, simply increment it by 1 so that is clear that there has been a version change during your video.
* Review the full Jenkins Console Output of this build. When viewing the Console Output you should clearly indicate where there is evidence of:
  + The Docker Image being created
  + The Docker Image being pushed to DockerHub
  + The update command being issued to Kubernetes
* Use the ‘curl $(minikube ip)’ command to show that the new application version has been successfully deployed.

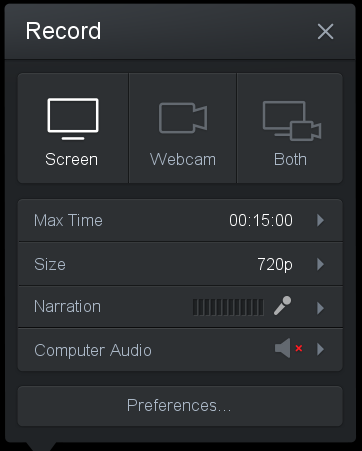
**2.7 Recording your Submission:**

You can use the following free tool to record your video:

<https://screencast-o-matic.com/screen-recorder>

Click the ‘Launch Free Recorder’ button and follow the instructions to get started.

When the application launches you should see a control panel like this:



*Figure 1: ScreenCastOMatic Controls*

Select ‘Screen’ from the top three icons and ensure that your Microphone us selected under the ‘Narration’ option.

You can adjust the size of the recording area by dragging the black and white border to adjust the window size or you can set a specific size using the ‘Size’ option.

Save your video as an **MP4** file (other formats will **NOT** be accepted) with the following naming convention: **Surname\_Name\_DevOps\_CW2.mp4**

This video should be uploaded to your **GCU provided OneDrive account** and a shared link (**accessible by everyone** **with no expiry**) to the video should be submitted to the DevOps Coursework 2 Video Submission.

**Please note that links to other cloud services WILL NOT BE ACCEPTED.**

**If your video is not accessible to everyone or expires it may not be available for marking/moderation and could result in loss of marks.**