Front-Loaded – End Point Assessment: Project Activity Template

|  |  |
| --- | --- |
| **Apply Content:**  **Note to Author:** The apprentice will work on this task after the [Module name & number] Apply activity.  This task must be mapped to the KSBs and encourage learners to; demonstrate the application of knowledge and skills at work, through the inclusion of evidence and workplace products*.* | |
| **Summary:**  **Project Activity:**  In preparation for a learner’s End Point Assessment (EPA), learners will put their newfound knowledge into action via a case study project, progressing their learning in the workplace.  **Workplace application submission:** Complete a project by fulfilling all deliverables stated, with explanations as to how this project has been completed.  The DLC will assist apprentices with this activity in the workplace. | |
| **Title** | **Detailed Definition and Instructions** |
| **EPA Module Activity**  **EPA Project: Case Study Introduction** | **Introduction:**  This submission is all about bringing what you have learned during your apprenticeship, both in training and in the workplace, to life, exploring and demonstrating both theoretical and practical elements of your learning.  Now that you have covered the culture of Agile and DevOps, programming, networking & security, various DevOps tooling, Infrastructure as Code, and the cloud, it’s time to put this all together!  For this task you will be required to complete a project scenario requiring the building of software, and the subsequent deployment of software to the cloud via an automated and secure pipeline.  You will be required to target your writing to address the mapped knowledge, skills, and behaviours. To help with this we have included advice on what to include and tips to the instructions below.  You are to produce a project that fulfils a given case study. A case study is a demonstration on how you have applied the skills relevant to the subject area. Include your work products with supporting narrative or annotation to describe the steps you have taken to complete the brief via deliverables, gather feedback and analyse your findings alongside workplace evidence demonstrating your ability to manage and lead projects. |

|  |
| --- |
| **EPA Project** |
| [need some support to fill this section!] |
| **Project Introduction** |
| You are a new DevOps Engineer that has been asked to streamline release management via secure and repeatable deployment of applications within a business. For this, you are required to develop an application which can subsequently be deployed to the cloud via containers, automation, scripts, and a CI/CD pipeline.  This project will involve concepts from all of your training modules; more specifically, this will involve:   * Agile, DevOps & Project Management * Programming & Testing Fundamentals * Networking and Security * Continuous Integration * Containerisation and Orchestration * Infrastructure as Code * Configuration Management * Cloud Configuration and Management |
| **Example Case Study Brief** |
| Below is an example Case Study Brief which has a supporting hosted Git Repo demonstrating all of the required tasks to complete. Your case study should be on a hobby project or a requirement identified in your place of work.  You have been hired as a Software & DevOps Engineer for a popular high street music shop QA Records that sell albums. They require a method of storing and updating orders that are made in the record shop. They require a Graphical User Interface to allow customers to create and view orders as well as an admin page allowing the store owner to edit the list of albums and edit the orders.  In order for the order management system to be effective they need to do the following:   * Create a Record * Real all records * Read record by ID * Update record by ID * Delete record by ID * Delete all Records * Query records by search term * Read an order by ID * Read all orders * Update an order by ID * Delete an order by ID * Delete all orders   The owners of QA Records are requesting the data for orders be stored within some form of SQL database and are open to more functionalities being added.  The shop is also looking into the benefit of adopting a DevOps mindset throughout this project and would like to explore the possibility of utilising a CI/CD pipeline for their releases. This pipeline would need to automate, containerise, and release versions of the order management system. |
| **Objectives** |
| The overall objective of the project is the following:  ***To plan, design and implement a solution for automating the development workflows and deployments of a given application, with utilisation of supporting tools, methodologies and technologies that encapsulate all modules covered during training.***  Specifically, you are required to create an automation solution for an application you will develop. You are expected to follow an Agile approach to your project management, requiring you to organise your project into Sprints, assign scrum roles and produce a product backlog in the form of a Kanban board.  You will be given a project scenario to adhere to during this project. By minimum requirements, the application must be developed in a general-purpose language and be linked to a database for CRUD-based operations. The application must be able to be containerised, with a view for the application and its container to be deployable to a cloud environment. |
| **Project Steps** |
| This project has been broken down into smaller sections reflective of your learning within your apprenticeship.  Whilst this is not a strict requirement of the order of work, the below is the minimum required in order to complete the project. You must complete every learning point listed below as to meet the Knowledge, Skill and Behaviour points stated in the assessment plan of your DevOps L4 apprenticeship. |
| **Project Steps: Step 1 - Agile, Kanban and Requirements** |
| **Below are the tasks required to complete this step:**   * Begin by finding a domain for your project, whether this is a hobby or within your company. Once found, write a small brief on the requirements for this application, including the following:   + Justification of why this is a suitable app for the domain   + Tech requirements for development   + How Agile and DevOps methodologies will be utilised within the project   + 5 or more user stories for features that this app will complete   + Brief discussion of a solution, including some starting solution features * Following this, ask your work colleagues to review and have them suggest at least 2 additional features. Once suggested, create a user story for these features. * Add acceptance criteria, functional/non-functional requirements and time estimates to your user stories and any resulting sub-tasks * Transfer all of these requirements over to a project Kanban board and share with your colleagues   **KSBs met in Step 1:**   * K4: The business value of DevOps in terms of Time, Cost, Quality, with an emphasis on building in internal Quality throughout the lifetime of the product * K10: How the user experience sits at the heart of modern development practices in terms of strategies to understand diverse user needs, accessibility and how to drive adoption * S3: Translate user needs into deliverable tasks, writing clear, concise, and unambiguous user stories that the whole team can understand. |
| **Project Steps: Step 2 - Programming, Testing, and Version Control** |
| **Below are the tasks required to complete this step:**   * Take the user stories from the previous step and begin writing an application and code to complete the story and feature via a general-purpose language, such as Python (K7, S11) * For at least 1 of these features, write your code in a TDD format i.e., write the tests for a feature first, then write the code to complete it (K14, S14, S17) * Link a database to your application, either internal or external, to store information * Create a repository for this application and ensure to merge code into the repo via the branch created in previous project, whilst also adhering to the feature-branch model * Create some form of User Interface for this backend application, whether it is a terminal based interface or utilising a Frontend using static HTML or a JS Framework such as React or Angular   **KSBs met in Step 2:**   * K7: General purpose programming and infrastructure-as-code. * K12: The persistence/data layer, including which database/storage technologies are appropriate to each platform type and application when considering non-functional and functional needs, e.g., monolith, microservice, read heavy, write heavy, recovery plans. * K14: Test Driven Development and the Test Pyramid. How the practice is underpinned by unit testing, the importance of automation, appropriate use of test doubles and mocking strategies, reducing a reliance on end-to-end testing. * S11: Employ a systematic approach to solving problems, using logic and hypotheses / experimentation to identify the source of issues. * S14: Write tests and follow Test Driven Development discipline in various different contexts. * S17: Code in a general-purpose programming language. * S20: Writing code in such a way that makes merging easier and facilitates branching by abstraction - i.e., feature toggling |
| **Project Steps: Step 3 – Networking and Security** |
| **Below are the tasks required to complete this step:**   * Prior to any continuous integration or deployment, write about the security threats presented by your new application and demonstrate this as a risk assessment included in the documentation * You should also add any further networking information or any security information in the supporting documentation, as a README.md file * Explore the tools used for security and how they are applied to the project in a pipeline. Begin to implement any application-level security measures to solve issues presented in your research for security threats   **KSBs met in Step 3:**   * K5: A range of modern security tools and techniques - e.g. threat modelling, vulnerability scanning and dependency checking, with a general awareness of penetration testing - in order to deal with threats and attack vectors within code and across the cyber domain. * K11: Monitoring and alerting technologies and an awareness of the insights that can be derived from the infrastructure and applications - collecting logs and metrics, configuring alerting thresholds, firing alerts and visualising data. * K16: How best to secure data; e.g. encryption in transit, encryption at rest and access control lists (ACL). * S6: Install, manage, and troubleshoot monitoring tools * S9: Application of a range of cloud security tools and techniques - e.g. threat modelling, vulnerability scanning, dependency checking, reducing attack surface area - incorporating these tools and techniques into the automated pipeline wherever possible. * S10: Assess identified and potential security threats and take appropriate action based on likelihood v impact. * S11: Employ a systematic approach to solving problems, using logic and hypotheses / experimentation to identify the source of issues. * S19: Interpret logs and metrics data within the appropriate context to identify issues and make informed decisions. |
| **Project Steps: Step 4 - CI/CD, Containerisation, and the Cloud** |
| **Below are the tasks required to complete this step:**   * Inspect and write up the current CI/CD strategy used by your business within the project you're on * With this strategy in mind, create a CI/CD pipeline that polls from a version control system for your application * Run your secured code through a pipeline, creating relevant automation steps, tests, virtual machine releases, containers, and any other pipeline steps you wish to implement * Re-write code where required to complete a build and release successfully. Alongside this, begin thinking about how you can re-configure your application to utilise APIs where possible * Inspect the monitoring security tools used within your pipeline and project you are involved with and how they contribute   **KSBs met in Step 4:**   * K1: Continuous Integration - the benefits of frequent merging of code, the creation of build artefacts and ensuring all tests pass, with automation throughout - including common tooling. * K2: The principles of distributed Source Control, including how to exploit the features of the tool, such as branching. * K7: General purpose programming and infrastructure-as-code. * K15: The principles and application of Continuous Integration, Continuous Delivery and Continuous Deployment, including the differences between them. * K21: Architecture principles, common patterns, and common strategies for translating user needs into both cloud infrastructure and application code. * S11: Employ a systematic approach to solving problems, using logic and hypotheses / experimentation to identify the source of issues. * S12: Automate tasks where it introduces improvements to the efficiency of business processes and reduces waste, considering the effort and cost of automation. * S14: Write tests and follow Test Driven Development discipline in various different contexts. * S15: Release automation and orchestration as part of a Continuous Integration workflow and Continuous Delivery pipeline, automating the delivery of code from source control to the end users. * S17: Code in a general-purpose programming language. * S19: Interpret logs and metrics data within the appropriate context to identify issues and make informed decisions. |
| **Project Steps: Step 5 - Infrastructure, Configuration, and Cloud** |
| **Below are the tasks required to complete this step:**   * Deploy an API-based application on immutable infrastructure via Infrastructure as Code tooling on the cloud * Consider the containerisation and security for this deployment, and how this factor in current CI/CD and application-level monitoring * Refactor the scripts written for your IaC deployment, and relaunch based on potential issues that could be found in security or performance logs presented by the tool   **KSBs met in Step 5:**   * K1: Continuous Integration - the benefits of frequent merging of code, the creation of build artefacts and ensuring all tests pass, with automation throughout - including common tooling. * K8: Immutable infrastructure and how it enables continuous refreshing of software, namely the updating of the operating system, container, and security patching. * K11: Monitoring and alerting technologies and an awareness of the insights that can be derived from the infrastructure and applications - collecting logs and metrics, configuring alerting thresholds, firing alerts, and visualising data. * K12: The persistence/data layer, including which database/storage technologies are appropriate to each platform type and application when considering non-functional and functional needs, e.g., monolith, microservice, read heavy, write heavy, recovery plans. * K13: Automation techniques, such as scripting and use of APIs. * K17: What an API is, how to find them and interpret the accompanying documentation. * S5: Deploy immutable infrastructure * S7: Navigate and troubleshoot stateful distributed systems, in order to locate issues across the end-to-end service. * S9: Application of a range of cloud security tools and techniques - e.g., threat modelling, vulnerability scanning, dependency checking, reducing attack surface area - incorporating these tools and techniques into the automated pipeline wherever possible. * S12: Automate tasks where it introduces improvements to the efficiency of business processes and reduces waste, considering the effort and cost of automation. * S18: Specify cloud infrastructure in an infrastructure-as-code domain-specific language. * S19: Interpret logs and metrics data within the appropriate context to identify issues and make informed decisions. * S22: Incremental refactoring by applying small behaviour-preserving code changes to evolve the architecture. |
| **Resources and Environment** |
| **For this project, you will need access to the following tooling & environments:**   * A computer with a browser, Word processor & spreadsheet software installed * Git Bash & access to GitHub * Kanban software, such as Trello, Jira, Asana or GitHub Projects * General purpose programming language with an appropriate IDE * Unit Testing, Mock Testing and Integration Testing packages * SQL DB Package, either cloud hosted or embedded * CI CD ToolO * Containerisation and orchestration software * Terraform & Ansible * Access to an appropriate cloud environment, such as AWS |
| **Deliverables** |
| The final deliverable for this project is the completed solution with full documentation around utilisation of supporting tools, processes followed, and pipelines ran. This will require a fully functional application in a version control system, commented and described via inline comments and a README contained in the repository.  You will be required to utilise the Feature-Branch Model, and to push a working copy of your code to the master branch regularly. It is recommended to use the feature-name naming strategy for your feature branches.  Alongside this, documentation and demonstration of the pipeline used to deploy the application should be present, alongside justifications of tool usage. Any user stories with their accompanying information (i.e., acceptance criteria, functional/non-functional requirements) should also be included, as well as feedback from colleagues and how this feedback was actioned.  In order to keep a record of how you have achieved all of the above, you are to also individually produce a portfolio that evidences each learning point described in the criteria.  You will be required to include all supporting documentation for your project within your remote repository, as well as the aforementioned documentation, to your DLC for your BUD submission. |
| **Next Steps** |
| [need some support to fill this section!] |