**DevOps Core Practical Project Specification**

**Introduction**

The purpose of this document is to outline the project specification of the DevOps practical project you will be working on for the next part of your training.  
This project will involve concepts from previous training modules, including:

* Software Development with Python
* Continuous Integration
* Cloud Fundamentals

**Overview**

Similar to the Fundamental Project, this project is designed so that you can be creative, however, there are certain rules you need to abide by that will be explained later.  
At its core, you are creating an application that generates “Objects” upon a set of predefined rules.  
These “Objects” can be from whatever domain you wish.  
For example:

* Account number and prize generator.
* D&D style character generator.
* Theme and setting generator for short stories.
* Data generator to seed a new Database.
* ‘Magic 8 Ball’
* Fortune Teller

**Architecture**

You are required to create a service-orientated architecture for your application, this application must be composed of at least 4 services that work together.

**Service #1**

The core service – this will render the Jinja2 templates you need to interact with your application, it will also be responsible for communicating with the other 3 services, and finally for persisting some data in an SQL database.

**Service #2 + #3**

These will both generate a random “Object”, this object can be whatever you like as we encourage creativity in this project.

You can create the “Object” however you like, some methods will be more complex but therefore show a greater technical understanding and flexibility, examples of how you can generate your “Object” are:

* Random number
* Random letter
* Pull an item from an Array
* Pull from a .csv
* Pull from a database
* API call to an external API

**Service #4**

This service will also create an “Object” however this “Object” must be based upon the results of service #2 + #3 using some pre-defined rules.  
Please see below for an example of how this logic can look.

The complexity of your logic here is up to you, again a simple implementation is allowed, however, may not showcase your full understanding of the technology.

**Different Implementations**

For services #2, #3 and #4 you need to create 2 different implementations, you must be able to demonstrate swapping these implementations out for each other seamlessly, without disrupting the user experience.

**Example Architecture**

To help clarify the above a potential application is described below, complete with 2 different implementations of the services, you do not need to base your system from this example.

The example is based upon the Account number and prize generator domain.

**Service #2**

A text generator with 2 different implementations available:

* One that creates 3 char Strings of lowercase letters
* One that creates 2 char String of uppercase letters

**Service #3**

A number generator with 2 different implementations available:

* One that creates a 6-digit number.
* One that creates an 8-digit number.

**Service #4**

A prize generator with 2 different implementations available, in both cases the prize is determined by the char string and number generated above.

* One for when the service is feeling generous (bigger rewards)
* One for when the service is not feeling generous (smaller rewards)

**Example User Journey**

A page is loaded and an Account Number is generated, it is “ABC123456”.

Because the first letter of the Account Number is “A” the user has a 25% chance to win £100 and a 75% chance to win £50, this is described in the logic of service 4.

The Back-end completes the logic to decide the prize number, based upon the account number originally created, and displays this for the user.

**Scope**

The requirements set for the project are below.  
Note that these are a minimum set of requirements and can be added onto during the duration of the project.

The requirements of the project are as follows:

* An Asana board (or equivalent Kanban board tech) with full expansion on tasks needed to complete the project.
* This could also provide a record of any issues or risks that you faced creating your project.
* An Application fully integrated using the Feature-Branch model into a Version Control System which will subsequently be built through a CI server and deployed to a cloud-based virtual machine.
* If a change is made to a code base, then Webhooks should be used so that Jenkins recreates and redeploys the changed application
* The project must follow the Service-oriented architecture that has been asked for.
* The project must be deployed using containerisation and an orchestration tool.
* As part of the project, you need to create an Ansible Playbook that will provision the environment that your application needs to run.
* The project must make use of a reverse proxy to make your application accessible to the user.

You should consider the concept of MVP (Minimum Viable Product) as you plan your project, complete all the requirements before you add extra functionality that is not specified above.

**Constraints**

The time constraint of this project will be discussed when the specification is given out, as this can fluctuate based on several factors.  
You must plan the work that you are doing and react to issues and risks as they arise.

The other constraint in this project is the technologies that need to be used.  
The project needs to utilise the technologies discussed during the training modules:

* Kanban Board: Asana or an equivalent Kanban Board
* Version Control: Git
* CI Server: Jenkins
* Configuration Management: Ansible
* Cloud server: GCP virtual machines
* Containerisation: Docker
* Orchestration Tool: Docker Swarm
* Reverse Proxy: NGINX

**Deliverable**

The final deliverable for this project is a completed CI Pipeline with full documentation around the utilisation of supporting tools.  
The CI Pipeline needs to be able to successfully deploy the application you have created as per the requirements.

A presentation of work will also be required towards the end of the deadline.  
This presentation will last between 10-15 minutes.  
Whilst planning this presentation, you should think about a brief justification for moving into the cloud, the tools you used to solve the problem and how what you have learnt at the Academy helped you use these tools.  
**Your ability to describe the tools you have used will factor into your final score.**

You will also need to demonstrate the working CI Pipeline that you have been able to build by rolling out updates to the system, **without** interrupting the user experience.  
Any downtime will be counted against you.

You are required to produce reports of any designs and work created as you progress through the project, refer to the marking scheme for the kind of evidence you may be expected to provide.

**Marking**

To receive a pass mark for this project you must achieve a minimum score of 21/35 with a score of 4 or better in the following areas:

* Systems integration and build: Produces software builds from software source code
* Release and Deployment: Uses the tools and techniques for specific areas of release and deployment activities.

**Marking Scheme**

Below are the skills that we will be evaluating for this  
assessment. These skills are as described in the SFIA 7 framework;  
please see below if you wish to have more information:

<https://www.sfia-online.org/en/framework>

The skills this assessment will discuss are the following:

* Programming/software development
* Systems integration and build
* Software Design
* Release and Deployment

**Programming & Software Development**

Designs, codes, verifies, tests, documents, amends and refactors simple  
programs/scripts. Applies agreed standards and tools, to achieve a  
well-engineered result. Reviews own work.  
Below is the list of criteria that will be assessed from your  
deliverable:

| **SFIA Skill** | **Rating** | **Details** |
| --- | --- | --- |
| Designs, codes, verifies, tests, documents, amends and refactors simple programs/scripts. | 1 | Software is missing functionality in major areas for creating, reading, updating and deleting. No tests implemented and designs documented were not implemented. Best practices not adhered to in the project. |
|  | 2 | Software functionality is working in areas but is not a fully working product. Tests were not implemented; code was commented in small areas. Parts of the project adhered to best practices but not consistently throughout the software. |
|  | 3 | Software is functional in all major areas but still has small bugs and/or errors. Tests were beginning to be implemented, with basic functions being tested. Best practices were adhered to for most areas of the project. |
|  | 4 | Software is fully functional and has been tested in all relevant areas. Best practices were consistently adhered to throughout the project. |
|  | 5 | Software is fully functional and has been tested in all areas, with best practices and refactoring adhered to and implemented throughout the project. Software implements concepts outside of the brief specified at a good level. |
| Applies agreed standards and tools, to achieve a well-engineered result. | 1 | Tools and workflows discussed in software documentation around design not implemented to a quality capacity. |
|  | 2 | Tools and workflows are referred to within the project, but the implementation is minimal. |
|  | 3 | Tools and workflows discussed in the documentation are implemented throughout the project at a high level, but more exploration could have occurred. |
|  | 4 | Software produced is in line with the documentation with changes made where needed, with references to refactoring apparent in key areas. |
|  | 5 | Software fully implements designs with justifications as to why changes were made where needed. Refactoring of code is rife throughout all relevant areas of the software. |

**Systems Integration and Build**

Produces software builds from software source code. Conducts tests as  
defined in an integration test specification, records the details of any  
failures. Analyses and reports on integration test activities and  
results. Identifies and reports issues and risks.  
Below is the list of criteria that will be assessed from your  
deliverable:

| **SFIA Skill** | **Rating** | **Details** |
| --- | --- | --- |
| Produces software builds from software source code | 1 | VCS implementation was non-existent. Build server was not installed and therefore no builds of software were created. |
|  | 2 | VCS implementation was attempted but structure was poor and/or content in VCS was irrelevant. Build server was installed but software did not build successfully from repository. |
|  | 3 | VCS was implemented and code was stored in a structured manner. Some of the content in the VCS could have been omitted but the majority was relevant. Build server installed and successfully built software manually. |
|  | 4 | VCS was implemented and code was stored in a structured manner with branches. All VCS content was relevant, no unnecessary files. Use of a configuration file to make build jobs portable. |
|  | 5 | VCS was implemented and code was stored in a structured manner with branches, git ignore file and scripted hooks. All files relevant to the repo, nothing unnecessary. Build server installed and successfully built software after a push with artefact produced for successful builds. |
| Conducts tests as defined in an integration test specification, records the details of any failures. Analyses and reports on integration test activities and results. Identifies and reports issues and risks. | 1 | No tests written for back-end. No logs or reports about application produced. No risks recorded or monitored. |
|  | 2 | Basic tests written for back-end. Results produced not acknowledged in any way. Some Risks acknowledged but not formally followed up. |
|  | 3 | Basic tests written the back-end. Results are tabulated and logged but no further follow up. A simple Risk analysis has been performed |
|  | 4 | Majority of tests for relevant features within the app are created in the back-end of application. Results of tests logged and discussed in report for test coverage. A more formal Risk assessment process has been followed. |
|  | 5 | Tests for all relevant CRUD features within the app are created in both the back-end of the application. Results of tests logged and discussed in report for test coverage. Further addition of code quality checking is also implemented. A formal risk assessment process followed and updated, analysis performed at end of project. |

**Software Design**

The specification and design of software to meet defined requirements by  
following agreed design standards and principles. The definition of  
software, components, interfaces and related characteristics.  
Below is the list of criteria that will be assessed from your  
deliverable:

| **SFIA Skill** | **Rating** | **Details** | |
| --- | --- | --- | --- |
| Creates and documents detailed designs for simple software applications or components applying agreed modelling techniques, standards, patterns and tools. | 1 | | No Designs provided |
|  | 2 | | Simple designs provided but not adhering to agreed standards and patterns. |
|  | 3 | | Simple designs provided meeting the agreed standards and patterns. |
|  | 4 | | Extensive Designs provided on both a system level and component level. |
|  | 5 | | Evolution of designs evident as the project progressed in all aspects. |

**Release and Deployment**

The management of the processes, systems and functions to package,  
build, test and deploy changes and updates (which are bounded as  
"releases") into a live environment, establishing or continuing the  
specified Service, to enable controlled and effective handover to  
Operations and the user community.  
Below is the list of criteria that will be assessed from your  
deliverable:

| **SFIA Skill** | **Rating** | **Details** |
| --- | --- | --- |
| Uses the tools and techniques for specific areas of release and deployment activities. | 1 | No release or deployment activities undertaken. |
|  | 2 | Release and deployment activities attempted; however results have not been a success. |
|  | 3 | System has been successfully deployed. |
|  | 4 | System can be robustly deployed with no further configuration required. |
|  | 5 | Advanced deployment and release techniques and tools have been utilised in order to create a very robust system. |
| Administers the recording of activities, logging of results and documents technical activity undertaken. | 1 | No deployment activities undertaken, or no logs or reports about application produced. |
|  | 2 | Deployment activities completed but not acknowledged in any way. |
|  | 3 | Deployment activities are tabulated and logged but no further follow up. |
|  | 4 | Deployment activities logged and discussed in a report. Certain advanced tools and techniques used have been explained thoroughly. |
|  | 5 |  |