LETTERKENNY INSTITUTE OF TECHNOLOGY

ASSIGNMENT COVER SHEET

To Be Completed By The Student

Lecturer's Name: **MS. Ruth Lennon**

Assessment Title: **Broken Legacy Code**

Submission Date: **25-Nov-2022**

Student's Name**:** **Koushik Hathwar KS** Id. Number:__L00163331_____

Course / Stage **Master's in DevOps**

Subject/Module: **Iac for DevOps Pipeline**

Word Limit:_____ Actual Word Count:

_____

I confirm that the work submitted has been produced solely through my own efforts.

Student's signature: **Koushik Hathwar KS** Date: **25-Nov-2022**

Note: PENALTIES

- The total marks available for an assessment will be reduced by 15% for work submitted up to one week late. The total marks available are reduced by 30% for work up to two weeks late.
- Assessment work received more than two weeks late, without prior approval by the lecturer will receive a mark of zero.
- Marks awarded will be reduced by 10 % if submitted work is greater than 10% above or below the assigned word limit.
- A further hard or electronic copy of your submitted work may be requested, and therefore you must keep a copy on disc.
- Incidents of alleged plagiarism and cheating are dealt with in accordance with the Institute's Assessment Regulations

Plagiarism: Presenting the ideas, words of someone else without proper acknowledgement. Refer to the Institutes' procedures and guidelines for the assessment of learners.

DevOps Process Design

It is important to have a well-designed DevOps process in the organization especially since there are fewer staff in the company and having a unified process for all the employees would bring stability to the overall development process. Since there is much shorter time and fewer staff to integrate the DevOps process, below are the different phase of the process I would like to propose for this company:

1. Operations: The operations phase takes care of Infrastructure and access to systems. This is an important phase as this acts as the foundation for several other processes since most of the processes need the right tools to perform efficiently. In the operations phase, the different elements that needs to be taken care of are:
    a. Infrastructure Setup: It is a group of hardware and software components that required to make DevOps possible. Along with storage, networking, processing capacity and Microservices, it also features an interface that users may utilize to access their virtualized resources.
    b. Access control: ThePermissions which determine a user's ability to connect to DevOps and use the features across DevOps, whereas access levels limit or controls the features which are available to users on the web portal. Example : Access to particular git repo.

2. Development: The development phase involves handling the source codes and making sure it is easy to revert to an older code when needed. In the development phase, the element to be taken care of is:
    a. Source Code Management/Version Control: GitHub is version control tool also has git repository hosting service which helps development teams to effectively manage changes version code in their codebases. It offers Git's source code management and distributed version control features.
    b. Jira : It is an essential tool for the organization to track the tasks and issues in the development pipeline. It's helpful to define when a project is done and track a task as it moves through different stages. The deployment type varies based on the customer's environment. We sometimes connect to the client's setup, which can be in the cloud or on-premise.

3. Testing: The testing phase is important since any bugs in the code could lead to system downtime. Since the time frame to implement is less and the number of staff available is less, we can limit this to only unit testing to make sure that there

is no trivial bugs in the system. In the testing phase the element to be taken care of is:

a. Unit testing: Unit testing is a quality assurance technique that involves breaking down application code into component building blocks and testing each block to make sure it performs as expected.

4. CI/CD: The continuous integration and continuous delivery phase makes sure that there is no issue while uploading the code to the production and also the code meets all the quality standards of production environment. This will also automate the process of updating code in the production as soon as there are changes in the code. For implementing CI/CD, the tool that can used is:

   a. Jenkins: It is defined as an build factory. In other words, it used to run tasks that dedicated to build integrate and deliver applications in the continuous delivery pipeline. Jenkins amazing feature is that it integrates nicely with other DevOps tools.

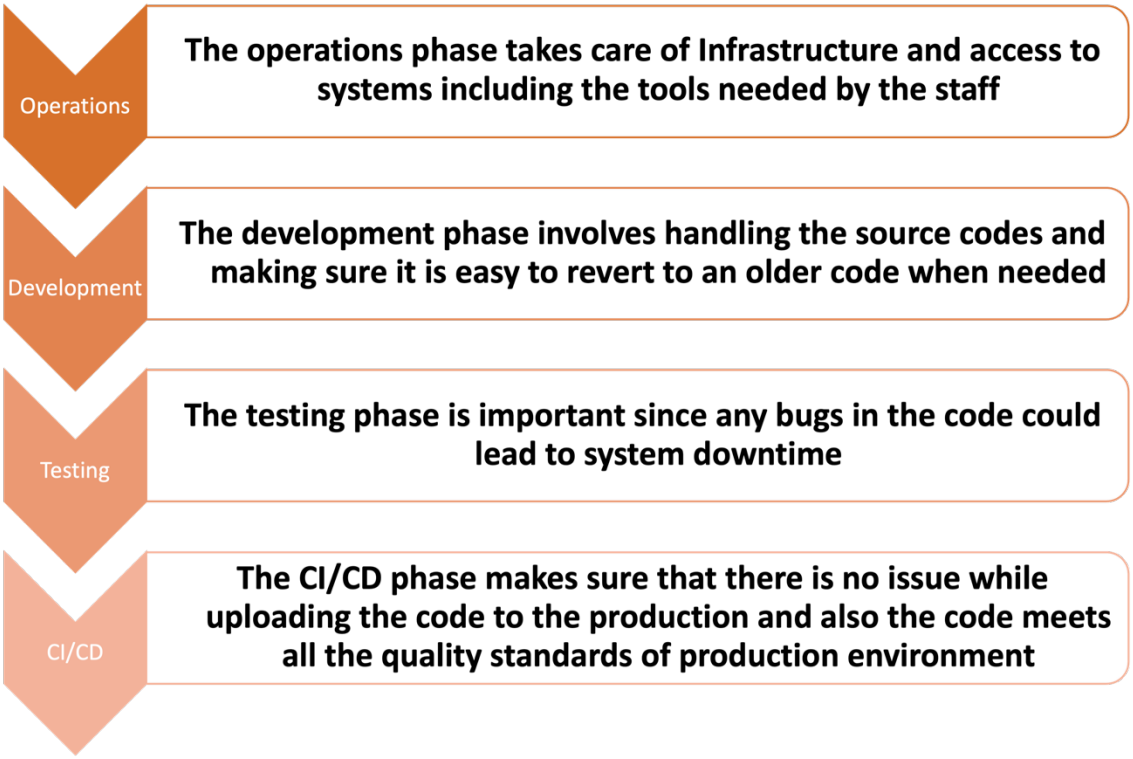

Figure 1: DevOps Process Design

Framework for Modernizing

Once the DevOps process is standardised across the company, the next step would be to modernise the existing code. Modernizing the existing code does not mean to use the latest technology in everything but rather to bring the current code to a stage where it follows the best practices of the DevOps process. This may sometimes include translating the current code into a new language to be efficient.

Some steps which can be followed in the modernization of the existing code are as below:

1. Identify systems to modernize: The first step in modernization is to identify the code/system which is important to the functioning of the organization. This could be a small script used to automate or a legacy system. Priority should be given to the system which provides the maximum advantage on modernizing. For example, if there is a UI system for users, this system can be first modernized so that end users can see the results faster and can feel the improvement or if there is a legacy database connections, this can be modernized to be more efficient so that there is least amount of delays in backend operations.

2. Analyse the code for alternatives: Once the code/system is selected for modernization, it need to be analysed to understand the implications of the modernization process, that is, which teams would be affected, which systems are going to be affected, etc. So that a proper change management can be implemented when migrating from the old code/system to the modernized code/system. The analysis phase is also used to understand the requirements and decide if there are alternatives for the implementation of the code/system like new programming language or if it can be implemented in a different technology stack.

3. Fix the code from bugs or rebuild the system: Once the code is analysed, it is time to make changes to the code. This involves fixing bugs in the legacy code or if it is suitable to use a new modern programming stack to rebuild the code using that. In both the case the code is being revamped to be better and modern.

4. Test the code using unit tests: Once the code has been developed, it is important to test the code for any error. Since this code will be replacing the previous code, it is important to make sure there is no bugs in it. At least some form of unit testing should be performed on this modernized code.

5. Deploy the code using CI/CD: Once the code is tested and ready for deployment, it should be deployed using the CI/CD pipeline as proposed in the previous framework. Adequate measures should be taken to make sure there is a fallback process for the old code since sometimes we may not predict how a complete modernized code can affect the system.

6. Monitor for any issues in the deployment: Once the modernized code is deployed, it should be continuously monitored for any issues. All the systems which rely on this code which was identified in the analysis step needs to be monitored as there could be side effects from the modernized code.

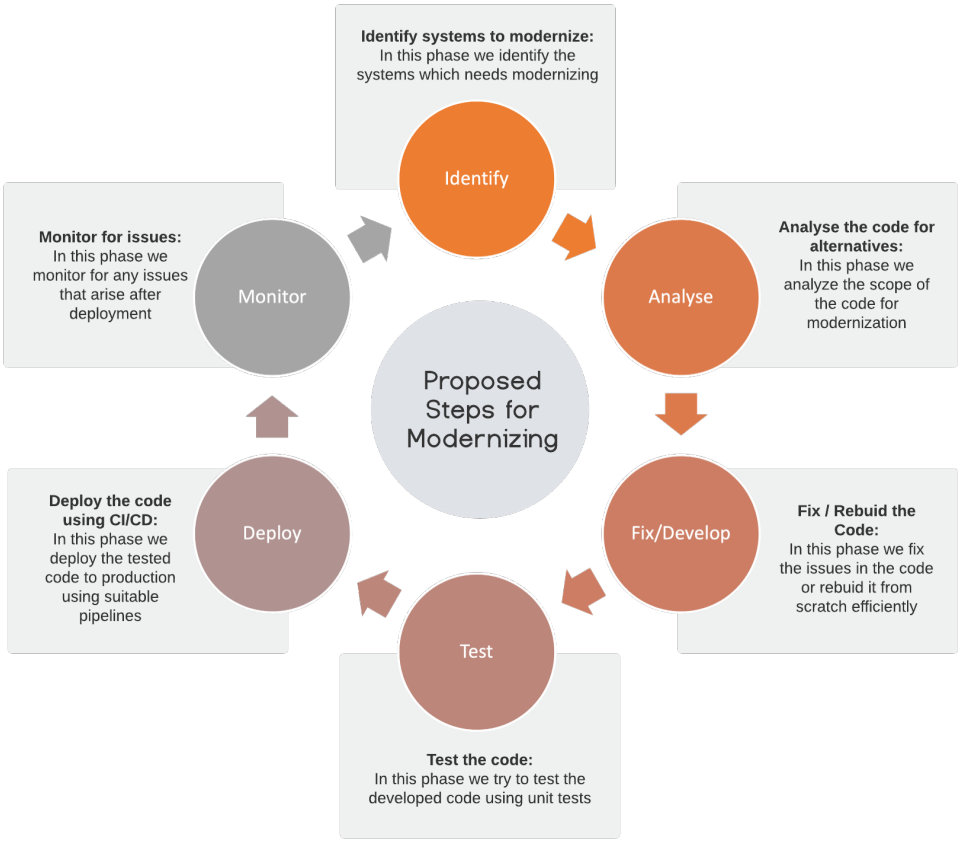

Figure 2: Proposed steps for modernizing

Implementation of Sample Code using the Proposed Framework

Let us use our proposed framework to implement and work on a sample code.

1. Install required IDEs and plugins: The code appears to be in PowerShell. VSCode is a reliable IDE and there is a plugin for PowerShell which will help in developing the code since it provides user-friendly syntax highlighting and code completion.

2. Push the code to GitHub: Once the IDE is setup, create a GitHub repo for the code and provide suitable permissions to relevant users who would be involved in developing the code. Push the current code into GitHub so that any new changes is tracked. If Jira is setup, create relevant Jira tickets to work on the code.

3. Analyse the code: Once the initial setup is done for the code, we need to analyse the code and find out if the code can be modernized. Look at the different sections of the code that needs fixing and analyse if a more modern approach can be implemented to the code. Sometimes rewriting a whole function to make it efficient and modern is worth the effort rather than fixing a legacy code.

4. Modernizing the code: Once the code is analysed, work on the actual development or delegate the work to relevant users, in our case the more suitable candidate for this could be the new staff (Jalen) since a fresh perspective on the issue helps in the modernization process whereas Ren may stick to the existing way things are done due to their long experience in the company. However, this is subject and depends on the individuals working styles.

5. Testing the code: Once the code is fixed/modernized, create a simple test case for the code so that it can be tested if the required results are produced. This could be a simple test case of testing it on just one machine since we do not have time for a longer complex test cases for different edge cases that could arise.

6. CI/CD: Since we are short of time in this sample scenario and this code is not critical to the business based on our analysis, we can skip this step and directly share the code with the relevant teams who would be using it or directly install it on the production machines. Skipping this step helps us to make changes to the organization rapidly due to limited resources.

7. Monitor the result: Once the code is deployed and run, we can monitor the logs to make sure there is no issues with it. Since we do not have a lot of time in this scenario we can keep the monitoring minimum to the default system logs on the machine rather than having a custom logging server or alerting system.

Conclusion:

DevOps is an important part of an organization since it helps to keep the development process in track and organised. There are hundreds of different ways in which we can implement an efficient DevOps process in a company but the reality is we will not always have the sufficient resources or the time to implement it and we would have to improvise.

In this scenario there was a limited time and limited resources provided to implement the DevOps process and this challenged me to think in a different way to improvise and provide a suitable solution to the problem. The solution proposed may not be the best possible options but it definitely made me analyse the problem critically and come up with a plan.

In this exercise I tried not to cut down a lot of the standard DevOps process since I felt those are important in an organization on a long term but tried to limit it to only those places where there is a critical need for it. Like, using Jenkins for CI/CD is not possible to implement for the whole company and for all the systems in 2 months but having it just for only the critical servers and critical software will help to avoid high priority incidents in the company when a code fails to deploy successfully.

The core learnings in this exercise according to me are summarised below:

- In reality we would not have all the resources or time to implement ideal practices. We would need to improvise and lay a good foundation so that it will be easy to implement the rest along the way.
- It is important to analyse the situation and make appropriate decisions like not all software being developed require the same level of compliance to standard practices
- It is equally important to learn to delegate work and to understand that different people have different set of skills and experiences. In this case utilising the limited staff, Jalen and Ren, is important.
- It is important to have a progress tracking tool like Jira since It helps us to know where we are in our task. It will also help us to prioritise our tasks accordingly.

Link to Github repo : https://github.com/L00163331/IaC-DevOps

References & Bibliography

1. Atlassian DevOps
   Available at https://www.atlassian.com/devops
2. Microsoft Azure Microservices
   Available at https://azure.microsoft.com/en-us/solutions/microservice-applications/
3. Lucidchart DevOps process flow
   Available at https://www.lucidchart.com/blog/devops-process-flow
4. Amazon Web Services
   Available at https://aws.amazon.com/devops/what-is-devops/