# Design a simplified process or framework for implementing DevOps throughout the company

Implementing DevOps for the first time in a company will definitely be a trying mission. Instead of implementing it as a whole, it would be ideal to progress stage by stage. It’s a continuous improvement process. With a blend of experience and youth, it will be a fantastic team to work with.

As a First step, I would ask Jalen to identify a team to start the process with. Identify the product which is of high priority, which makes more revenue for the organization, which is critical for the organization. Once the team is identified, the next step will be understanding how the team works and identifying the bottlenecks. Ren will be able to easily identify and document those with the amount of experience he possesses. The next step will be to schedule daily meetings with the team to discuss how to facelift. It would be wise and productive to introduce Agile as the first step of DevOps. Because Agile and DevOps working together yield more benefits.

Initially, we will have to break down the daily tasks to be done into smaller chunks and document it as user stories and tasks. For this, we can introduce new tools like Jira or Azure DevOps to the team where they will look for their daily work. The developer or testers will not work on any other tasks except the daily tasks which will be a smaller job. Ren can act as a scrum master for the time being and will schedule daily meetings with the team to review the daily tasks completed. We will have Sprint Planning, Backlog Refinement, and other meetings introduced on a sprint basis. Gradually, by showing the effort and iteration graph, the amount of work to be completed by the developers will be increased. The quality and quantity of the work are guaranteed to rise as developers gradually adapt to the new ways of working. This will in turn decrease the time to market. Also, the microservices architecture should be introduced. There should be Tech Talk sessions happening weekly, where the developers will research on the new tech stacks and have a brief session on the topics. On the same topics, a POC can be done by the developers and same can be implemented for the product.

In parallel, we will have to introduce the CI/CD to the team. Jalen can do some research on the new and effective tools in the market which can be introduced to the team. Because, Once the developers complete the coding, we should be able to test the code in a testing environment without any delay. For fast-paced deployment of the code to servers, we need deployment tools like Jenkins, Octopus, or Azure DevOps Pipelines. When the CI/CD is implemented, the moment a developer check-in his code, a build will be triggered and the code will be deployed to the testing environment.

Once the CI/CD pipeline is almost stable, we can introduce extra features in the build pipeline. We can have build validation, Unit testing in the build pipeline with VSTest or JUnit, keep a criterion of 85% code coverage, Static code scans with fortify or checkmarx, Code quality checks with likes of sonarqube. When we get a report of SAST scans, we can ask the developers to fix the code vulnerability with some SLAs. Also, we can improve on quality of the code by running through the code quality report. With these features added to the build pipeline, the coding quality of the developers will gradually increase.

We can now extend the deployment pipeline to deploy the code till the production environment and can also have the automated regression testing happening right after the deployment. We can use tools like ReadyAPI, Postman, and Selenium for this.

While these steps are getting stable, we can have the monitoring in place for the production environment. We can use tools like elastic search, Splunk, and Dynatrace for monitoring. The alerts will be worked on by the developers based on the errors. Developers analyze the errors to find if it’s a bug in the code or any environmental issues.

Please refer the Figure 1 to find the DevOps Process Roadmap.

It is a certainty that the team will be much more efficient when agile & DevOps processes are followed diligently. However, Continuous improvement at every stage is the key to creating an efficient team.

# Design a process or framework for modernizing existing code

Investing in application modernization can improve customer and employee experiences, ensure that applications continue to be responsive to changing business needs, and ensure that organizations stay ahead of the competition.

Modernization steps include:

* Analyse the current code
* Identify the new technology stack
* Identify the resources
* Dependency List
* Module list
* Architectural design
* Refactor the code
* Continuous Improvement

Invest time in analysing the code and have to identify the new technology stack where the code will be more reliable and sustainable. I would prefer to have the existing code rewritten in Python. The information can only be retrieved from Windows servers by PowerShell scripts at the moment. To meet future expansion goals, systems might get moved to Linux or the cloud. So, Python works better than PowerShell. The existing script should be written in a way to identify the OS where the script is getting executed and it should provide the event logs based on the OS.

We have to list out the dependencies and modules of the current script. Do a study of the dependencies and see if the same dependencies are required for the new Python program. Separate out the modules in separate services and call through an API, and Implement microservices architecture. So, if there are any changes, only the selected module has to be touched for the update. Each module or task can be moved to different user stories and can be handled by different programmers. The program should be moved to a version control system so that any mistakes can be easily compared with earlier versions. Also, multiple developers can work on the same code if it’s in a version control system like GIT.

Continuous Improvement is the key to a successful organization and successful product. There is always a scope for improvement to the existing code as the requirement tends to change.

# Review the code provided to demonstrate how your modernization processes would work

As a first step, the code will be moved to a version control system. After analyzing the code, it’s decided that the code will work better in Python as it doesn’t have any OS dependability. With future expansion, the client machines can be of any operating system or even the cloud. After deciding on the technology, identify a resource who can work on the decided language as the code is getting rewritten from old to new technology.

List out the dependencies and modules. Figure out if the same dependencies are needed for the new language used. Multiple functions are identified in the script and these can be separated as modules and different services. It’s highly recommended to implement microservices and call one another with APIs. There are several advantages of modularising the code. With the version control and scrum team in place, Multiple developers with team collaboration can work on different modules, which speeds up the development.

Review the architectural design of the current code and should make necessary changes if there is a drift. Now the code can be refactored with new technology stacks. While refactoring code analysis is done for security and code quality checks using multiple tools in the market such as Sonarqube, Veracode, and Checkmarx. When refactoring a script, a code review is mandatory, as the review will definitely help in a better quality of the code. Once you’ve organized the legacy code, you should keep it neat and clean. The importance of code maintenance cannot be overstated. Another important item to follow is the right documentation. It helps in further improvement of the code.

# Conclusion

It’s critical to recognize that modernization won’t be done fast or overnight if it is designed to be done right. If teams emphasize the low-hanging, high-value areas identified in a code audit, the benefits of a successful modernization will be nearly instantaneous.

With DevOps and Agile working together, it’s expected to produce better product and business performance. Automation, the heart of DevOps helps reduce errors.

# Appendices

Figure 1: DevOps Process RoadMap

