ATLANTIC TECHNOLOGICAL UNIVERSITY

ASSIGNMENT COVER SHEET

To Be Completed by The Student

Lecturer’s Name: Ms. Ruth Lennon

Assessment Title: Broken Legacy Code

Submission Date: 24-Nov-2022

Student’s Name: Tony Mathew Thomas Id. Number: L00171045\_\_\_\_

Course / Stage Master’s in DevOps

Subject/Module: IaC for DevOps Pipelines (2022/23)

Word Limit: Actual Word Count: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

Student’s signature: Tony Mathew Thomas Date: 24-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

Given the situation that we need rapid deliveries of iterative improvements and fixes, a DevOps process with CI/CD pipelines has to be in place to enable faster turnaround between fixes and deployments. This system should enable to make modifications quickly and get the changes deployed onto an environment that should be verifiable by stakeholders. As an initial attempt the following elements should be setup as a bare minimum to prove the viability.

**Source Control System**: This is a key element and a starting point in laying down a DevOps process. As we modernize existing code, it should be checked into a code repository and be made available for further modification. If there is an existing source control system chances are that it is outdated. Git is a modern and an ideal candidate here. Having a source control system allows for expandability in future meanwhile the initial setup remains simple.

**CI/CD Pipelines**: We expect code to be modified and be modified regularly at this stage. Having a CI/CD pipeline for a project will build and deploy the changes in the appropriate environment as and when needed. The idea should be to make it available to the stakeholders to do a quick round of verification or demonstrate that it meets the acceptance criteria. And if there needs a modification the process can easily be repeated without much manual intervention.

**Integration Testing**: This is to be formulated based on the acceptance criteria laid out earlier before the in agreement with the stakeholders. The benefit of having an integration test automated and be made part of the pipeline is that it too repeats towards the end of the pipeline. Thus, a set of criteria can be tested over each iteration.

**Deployment Environment**: This would vary on each run and from project to project. Usually this will be a test environment where verification can be done and demonstrated to the stakeholders to convince them the solution works and meets the acceptance criteria. While deployment to the test environment could be automatic, deployment to production environment will be manual based on further decisions.

### Phases in the new process

Even though the intention is to get the products of the company production ready, the approach here is radical with regards to the company’s present situation. Hence, this is considered to be experimental in the initial stages with certain level of research and thoughts applied. Below diagram shows the phases and the approximate time taken for each phase.

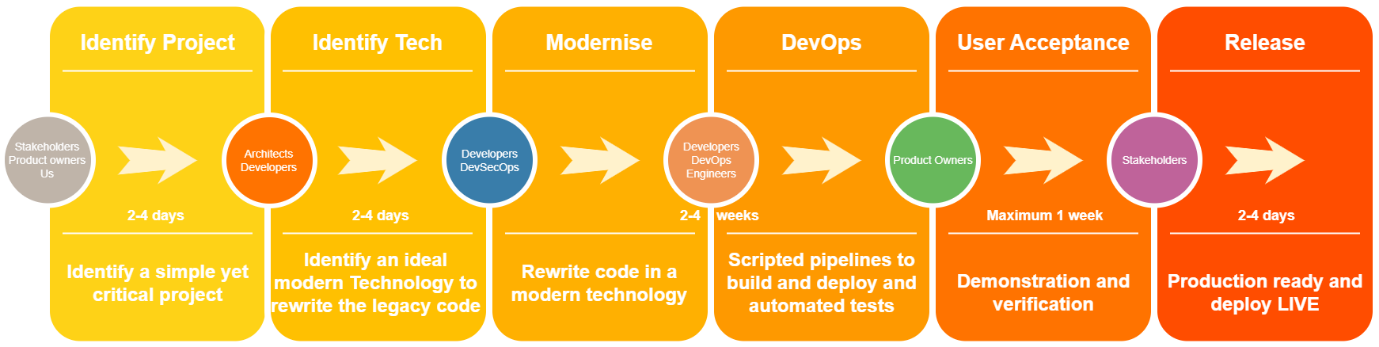


Figure . Phases in new DevOps process

1. **Identify a candidate project**

An ideal candidate project to prove our implementation strategy works should be one that does not take ages to rewrite but at the same time critical for the day to day running of the company. A product that is customer facing, enabling to place online orders or enabling online payment for the services are characteristics of such a product. Selecting such a candidate would easily get the attention of the stakeholders and encourage their involvement into the process. This might involve hours of investigation and meetings with people across an organisation including stakeholders. In the given scenario we will have to rely on Ren who has been there with the organisation for a long time and familiar with the people there. He should be able to identify the stakeholders and potential projects as candidates. An approximate time estimated will be two to four days here.

1. **Identify new Technology stack**

Once the project is selected, we need to identify the right technology stack to convert it into. For instance, a project with code written in VB which is considered to be legacy. This could be rewritten in a modern framework like .Net. Setting up of environment too is a consideration that cannot be ignored. A web application would ideally be deployed as a service in a resource in one of the cloud platforms. The App Service resource in Microsoft Azure, is an example, which not only supports easy deployment of a web application but also has the features like scaling in/out built in. Alternative cloud platforms like AWS and GCP also supports various types of application. They also give us the ability to connect to a DevOps pipeline to automate the process of releases to a great extent. This phase is mostly involved by the Dev team including architects and product owners. Another two to four days may be taken further to decide the technology stack

1. **Modernise selected project**

Here is where the rewriting of the legacy application will happen. Depending on the technology some of the lines may be exactly copied over. But a rewrite will never be smooth as there could be functions that are obsolete and newer concepts to apply which may not have existed in the past. Parallelism, asynchronous method calls are two examples of features that modern programming languages have which did not exist previously. This phase also determines how the CI/CD pipelines are configured which is the next phase. The pipelines could be started to script once the first commit has been made to the source control. Hence, certain activities go in parallel between these two phases.

1. **Implement DevOps pipeline**

Having a pipeline that will automatically build, generate the artifacts and deploy to the hosting environment greatly reduces time for release activities. There are lot of frameworks that we could choose from for setting up CI/CD pipelines. From Jenkins which is free of cost and simple to setup to Azure DevOps suite which goes beyond the scope of CI/CD to the various phases of project management. All of these modern systems have their own toolsets to work with different technology stacks. We need to pick one based on the organisation’s profile. For instance, a company with Office 365 or MSDN subscription does not have to spend much for their Azure DevOps suite. Moreover, most of these system does not cost much or are even free to use the basic features which is sufficient during the initial stages. At the end of this phase an application should be ready to be deployed into an environment chosen. This phase mainly involved by the DevOps engineers and to a limit, developers. Both the modernising and the scripting of DevOps pipelines would together take up to 4 weeks.

1. **User Acceptance**

A demonstration of the product in the UAT environment may be carried out here. It might even involve a certain degree of testing by the stakeholders themselves to gain their confidence for a possible flag off to production release. This phase might require us to repeat the process of coding and running the pipelines again until the stakeholders are happy about the sanity of the project. Several iterations may have to be conducted after an attempt to convince the stakeholders. New business cases may be discovered during a demonstration and may have to be worked up on and the code be run through the pipelines again until we receive an acceptance. There is also a chance that newly found use cases are deferred for a future release. This phase should not take more than a week to obtain the acceptance to go LIVE.

1. **Release to production**

Having a release to production means it’s ready for use by end customers. It does not mean the released product is final and free of bugs. It will probably undergo fixes and enhancements again. But having a release means an achievement of one or more milestones in the products lifecycle. Time taken for phase depends on the number of steps required to perform for production release. The maximum expected for medium sized project is no longer than four days.

The iterative behaviour of fixing or enhancing the product after feedback from stakeholders during User Acceptance phase is ignored here for brevity. Further break down of activities is shown in a swim lane flowchart in Figure 2.

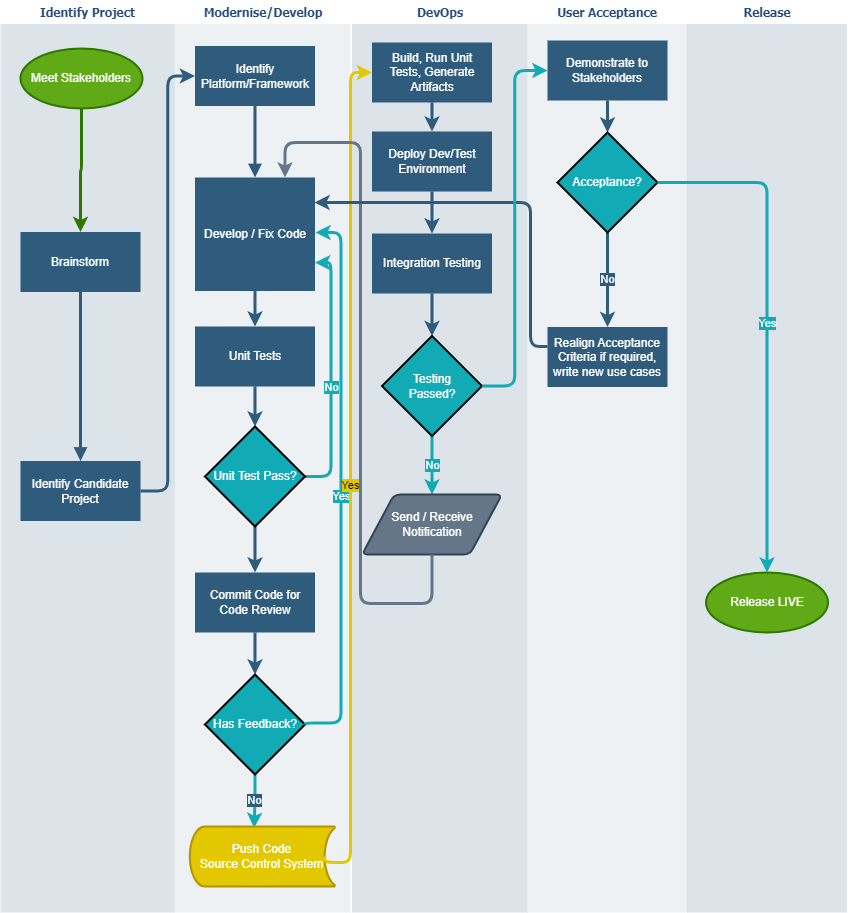


Figure . Detailed view of DevOps framework adoption

# Framework for Modernising

It’s very difficult to arrive at a common number of steps for converting projects that are written in different legacy technologies. Each project being written in a different language may be distinct and the steps for modernisation might differ from one another. However, a common framework or processes can still be laid out in general to rewrite projects using modern technologies by taking various aspects into consideration. Following points lists out the key considerations to modernise a code written in legacy stack.

**Identify the modern Technology stack**

We need to arrive at an equivalent technology stack that would have been used for a similar project today which is of course latest. A project written in classic VB/ASP/XSLT stack for instance would be written using ASP.Net framework in C#. Such an upgrade would also solve the problem in finding the right resource to work with the projects. Jalan, who is a junior developer can easily be trained to adopt to newer stack with the right platform to enable easier learning. Platforms such as Udemy and Pluralsight are examples. It’s worth mentioning here that it may not be a single language but rather a set of languages which may contradict the legacy set up. Cost of licensing is another factor to be a considered here. And stakeholders have to agree to the expenses that may incur as part of this.

**Analyse code**

This is a key step which plays a vital role in modernising. The success of modernising heavily depends on this stage. This step gives us the understanding of what the code does and greatly influence in rewriting the code in a newer language with confidence. We would ideally end up with diagrams at this end of this stage to illustrate the business cases that the application is meant to solve. This phase would also involve stakeholders to confirm our analysis. To identify the right stakeholders who are of interest to the candidate project we need to rely on Ren, who is quite familiar with the organisation. There are cases where an organisation might hire external parties who are quite familiar with the legacy technology to arrive at the business cases. We may arrive at a list of functions intermediate result that will highlight the key functionalities the code performs and any dependency between other applications or services. UML diagrams such as Use cases, flow charts or sequence diagrams are expected output of this phase. Figure 4. shows a sequence diagram of an existing code illustrating the calls between the functions.

**Rewrite code**

This is where the actual rewrite happens that involves the developers and architects. There will be a lot of involvement by Jalan at this phase who would write the code. An architecture pattern may be identified which will naturally follow from the previous stage. Design diagrams at a detailed level will be drawn out at this stage. Following OOPS principles such as Single-Responsibility Principle (Wikipedia, 2022) which, encourages to separate code into classes and further into projects may be applied here. The written code may undergo several cycles of reviews and fixes and developer testing. If the organisation had a huge monolith application, then it may be broken down into several services following microservices pattern which is about vertically slicing the application. But an ideal candidate that we pick as candidate will not be big enough for this. In this case we might consider our first candidate as a single service in a cluster that will just have a backend and a UI to interact with. Few of the considerations at this phase are:

* Introduce modern programming concepts: All modern programming languages supports concepts like parallelism, scalability, asynchronous calls, lazy loading. These are all features of modern software that make optimal utilisation of the latest hardware capabilities such as multi-core computing and virtualisation. Although, the technology that we may choose supports these features, we might have to ensure these features are made use of wherever possible while writing code.
* Use tools and libraries: All modern technologies either come with or can be associated with tools and packages that will make working with them easier. These include tools or libraries for unit testing, API endpoint testing, test code quality and vulnerabilities. Most of these can easily be integrated into IDEs like Visual Studio, Eclipse which allows to ensure criteria are met or sanity is tested as we write code. For now the approach taken will be minimal and many of these are assumed to be considered in future.
* Use Mock data: Before production, each of the modules are tested using mock data that will closely represent real data in a LIVE system. Data need to be setup for all the possible business cases so that no condition or branches in the code that we wrote are left without running through several cycles with mock data. There may be cases where we require to set up several sets of data for each scenario the system would go through. The results from these tests will not only discover the holes but also identify possible shortcomings.

**Continuous improvement**

We may not get everything right in our first attempt. Besides bugs, there could be further scope for improvement, refactoring of code for better design patterns. There also could be the notion of new features as enhancements which could be easily achievable using the new technology and bring immense benefits to the clients. These may not be identified in our initial analysis phases but rather be identified as iterate over the phases. The spotted improvements may not be immediately addressed but be kept in the backlogs for improvement to be addressed in the future cycles depending on the prioritisation.

Below diagram shows the estimated timeline of the entire process. The entire process should not exceed more than 5 weeks.

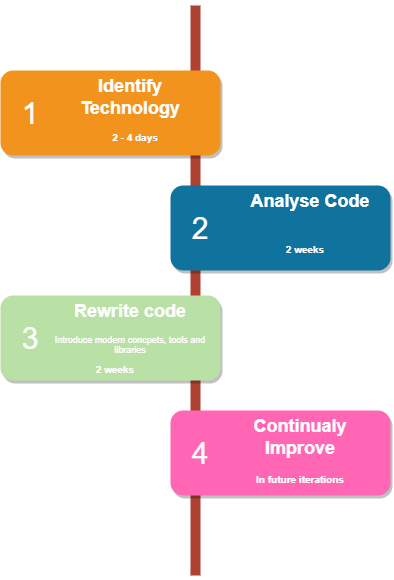


Figure . Timeline for modernisation

# Demonstration of Modernisation

This section attempts to demonstrate how a module in the candidate project will go through the framework laid out for modernisation. For this purpose, the sample PowerShell code is considered to show how these activities are carried out.

**Equivalent** **Technology** **Stack**: The code is written in the older version of PowerShell which is Windows PowerShell. The most recommended language to rewrite is PowerShell core (Microsoft Learn - Sean Wheeler, Mikey Lombardi, 2022) which has taken over the older one. The newer one is platform independent and is expected to be supported by Microsoft for a long period. Some of the cmdlets used in the existing code are not available in the newer version. Hence, some level of code rewriting is to be performed. Running the rewritten code with its Settings.ini file pointing to few servers temporarily setup for testing will help test the code as its being rewritten. Other alternatives to PowerShell core are Bash scripts and Python. However, PowerShell core would be easy to rewrite as most of the lines can simply be copied over. There are enough tutorials available for all these newer scripting languages in learning platforms such Udemy.

**Code Analysis**

More time may be spent here than the actual rewrite itself. An audit of the code is performed to understand what is being performed. Comments written are examined and cmdlets used are looked up in the API documentation written by the author (Microsoft) to confirm our understanding. Each function is examined in detail to understand its responsibility and how they are called and connected to each other by looking at their calls. A sequence diagram drawn is shown in Figure 4 that shows the sequence of calls between functions. A description of each function is written in Table 1. Basically, the code is reading from an input file the list of servers that it should monitor and calls few functions to probe the health of each server. The final result is a list of statuses and details of each server in a tabular format. This level of detail of what the code does is helpful for further rewriting of code.

|  |  |  |
| --- | --- | --- |
| **Function** | **Description** | **Caller** |
| Main | This is the main or startup block where the run begins. Reads file Settings.ini which should be having a list of hostnames or IP addresses of servers we want to test regularly. It calls function Network-Tests passing the list/array of server names as parameter |  |
| Network-Tests | Iterates over each server name/IP and calls other functions and maintains an array at the granularity of server name/IP to update with the status of each function call against the list items. Also prints the consolidated result received from called functions in a table format at the end. Figure 5. shows tabular formatted output. | Main |
| Test-Connection | This is a built-in PowerShell cmdlet that is called by Network-Tests to check whether a server is reachable or not. Furhter tests are progressed only if this is success | Network-Tests |
| Get-UserDetail | Get the logged in user a/c of the server; Returns the detail in an array; If there is any error then the error/warning is returned instead of the user detail | Network-Tests |
| Check-WarningsErrors | Reads the event logs of a server and returns the details of the logs including the message; if it cannot read then an error message is returned instead of the log details | Network-Tests |
| Get-NetworkInfo | Returns network details like IP addresses associated, routes, ping status; If there is an error that happens when the probe is initiated then instead of the details the error message is returned | Network-Tests |
| Check-OpenPorts | Returns a list of ports and status whether they are open or not; If an error happens then the error message is returned instead | Network-Tests |

Table . Functions and their descriptions in the PowerShell code

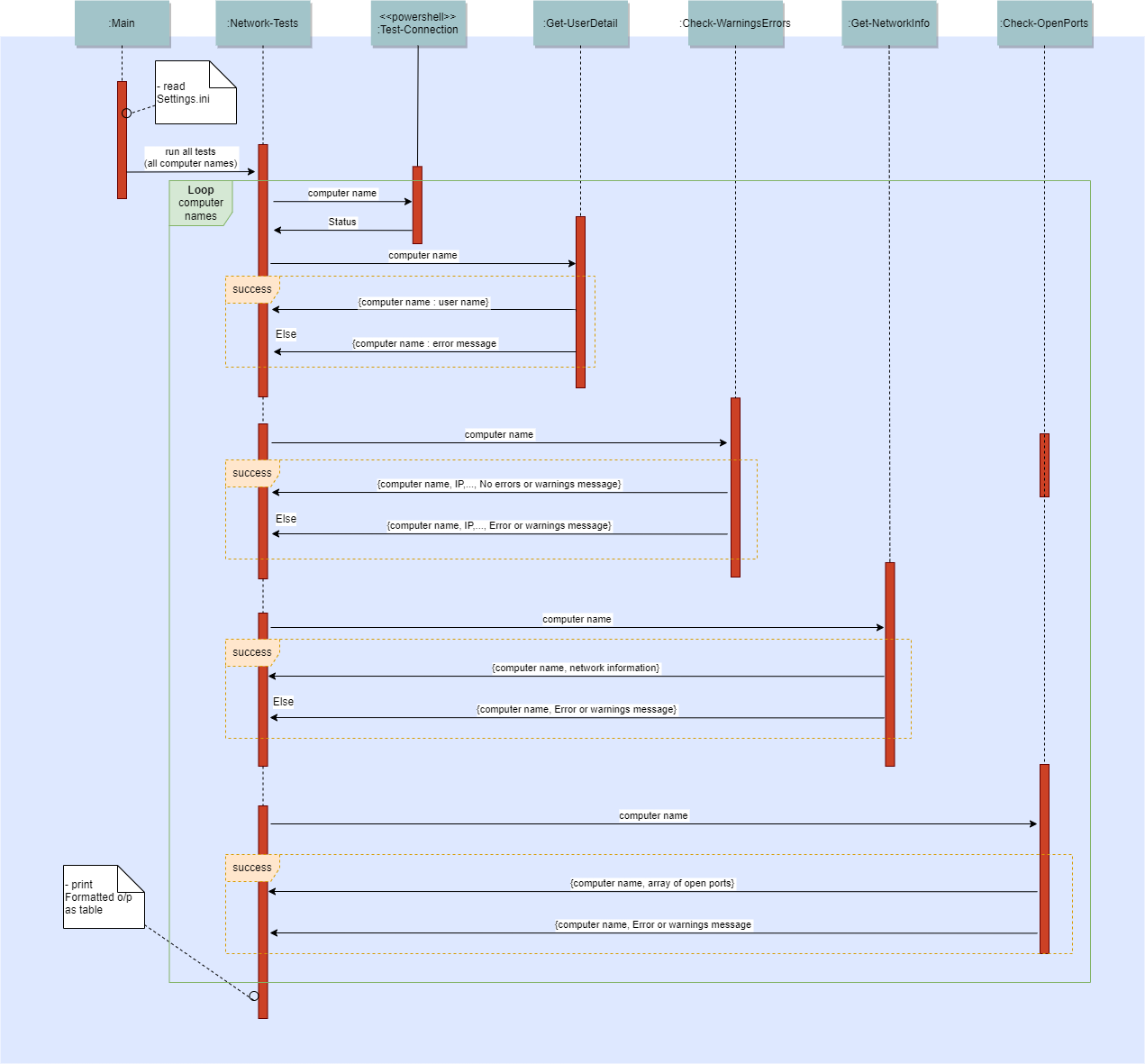


Figure . Sequence diagram showing the flow between function calls in the sample PowerShell code

**Rewrite code**

If the Code analysis is done correctly then the rewriting becomes easy and follows through naturally. With a proper code analysis, we are confident that we know what the code does and being comfortable in a newer technology the equivalent of an obsolete function is easily be replaced. The online availability of documentation too assists here. Availability of mock servers and creating a Settings.ini file with the list of those servers IP address allows us to run the rewritten code against production like data. A sample output assumed while analysing the code is shown in Figure 5 below. This will help to gain the confidence of what the new code does and what it does not. Tools like Visual Studio Code, PowerShell ISE supports writing of code and enables to test the code locally as we write. The new PowerShell Core also supports a lot more libraries than the old one and moreover the community is more active and updates and fixes are more frequent.



Figure Expected output of PowerShell code

**Continuous Improvement**

Ideally this script may be run in an environment like an Azure function which will not only cost less but has the capability to write logs and performance markers into stores like Application Insights which is very comprehensive in troubleshooting and monitoring. There are also built-in tools like Azure Monitor (Microsoft Azure, 2022) that will provide a lot of diagnostics information. Such level of information is not only helpful in troubleshooting issues but also to improve the performance of the code that is already running in production. As the new framework easily allows further modification and testing being automated the lead time is greatly reduced as new bugs are discovered or enhancements are suggested.

# Conclusion

This exercise gave a feel of how in reality an organisation that is totally new to DevOps practices would implement it under tight timelines. The solutions described here are minimal respecting strict timelines. It is expected that these processes will be enhanced over time. Features such as pipelines to detect code vulnerability, many of the scrum ceremony activities are ignored to rescue the organisation from the downfall. However, it is ensured that the implemented framework can be extended in future to support these additions when the company has survived the downturn and is on the path to progress.

Having a DevOps process in general will benefit the organisation in general rather than focusing on a single project. The approach here is to arrive at a minimalistic solution with most necessary components put in place. Besides identifying a candidate project by prioritising them with the help from stakeholders, stages such as source control system and CI/CD pipelines are components that we cannot ignore to put up into the DevOps solution. With the limited human resources and being under strict time constraints, automating a great deal of work through writing pipelines helps. Building, testing and deploying happens automatically into the testing environments as code is checked in by the developer(s). Thus, the lead time is significantly reduced. A lot depends on the success of the initial project we choose to get through the DevOps framework. Its success will gain the confidence of the top officials in the organisation and the approval for going ahead with applying it organisation wide. Similarly, the modernisation process too is carefully laid to be as lean as possible. Most of the steps described in the process are minimal and it may be noticed that the actual modernisation as demonstrated takes one of the many options described in the stages of the process. The process is a bit general and acts as a guideline to complete a stage. An example of this is the legacy Windows PowerShell code being chosen to rewritten in PowerShell Core out of the many options. The tools to develop is Visual Studio Code although, many other options exist. The method of testing here is defined as using of mock data to closely mimic the production input by setting up servers accessible from the testing environment. This is also specific to the kind of application. The demonstration of modernisation framework for the given PowerShell code clearly shows how the code is audited and a list of functions is arrived at and the interaction between them is shown through the sequence diagram which would provide a detailed design for the developer Jalen who is rewriting it. The sample output to helps to validate the code in the end before pushing it to production environment.

# Code Repository

<https://github.com/tonythomas-L00171045/atu/tree/main/IaaC/Assignments/Assignment2>

# References

Microsoft Azure, 2022. *Azure Monitor.* [Online]   
Available at: https://azure.microsoft.com/en-us/products/monitor/?&ef\_id=CjwKCAiAyfybBhBKEiwAgtB7fgpYYJGxg1XG-fpT8RBNUnbQT0lJF7vnI7mw7WYdkGm4LgrS2m3L3xoCRNcQAvD\_BwE:G:s&OCID=AIDcmmsmruuku2\_SEM\_CjwKCAiAyfybBhBKEiwAgtB7fgpYYJGxg1XG-fpT8RBNUnbQT0lJF7vnI7mw7WYdkGm4LgrS2m3L  
[Accessed 24 11 2022].

Microsoft Azure, n.d. *App Service.* [Online]   
Available at: https://azure.microsoft.com/en-in/products/app-service/#overview  
[Accessed 22 11 2022].

Microsoft Learn - Sean Wheeler, Mikey Lombardi, 2022. *Differences between Windows PowerShell 5.1 and PowerShell 7.x.* [Online]   
Available at: https://learn.microsoft.com/en-us/powershell/scripting/whats-new/differences-from-windows-powershell?view=powershell-7.3  
[Accessed 24 11 2022].

Wikipedia, 2022. *Single-responsibility principle.* [Online]   
Available at: https://en.wikipedia.org/wiki/Single-responsibility\_principle  
[Accessed 23 11 2022].