CTU training solutions

0861 100 395 | www.ctutraining.co.za | enquiry@ctutraining.co.za

# Gabriella Rakgotsoka

20232605

DOE522_Formative 1 25/03/2024

Question 1 .......... 3
Question 2 .......... 6
Question 3 .......... 10
Completed Declaration of Authenticity .......... 13

## Question 1

Explain the key principles and benefits of DevOps in modern software development practices. Discuss how DevOps can help organizations enhance collaboration, accelerate software delivery, and improve overall product quality. Provide real-world examples to support your explanations.

Principles and benefits of DevOps in modern software development practices:

### 1. Collaboration

The combination of the development (Dev) and operations (Ops) teams is the essence of DevOps. It follows that cooperation is essential to the DevOps basis. Together, developers and operators can improve the software's configuration for the operations phase and conduct early testing to guarantee that it satisfies requirements. Effective information-sharing procedures are essential for collaboration as well. If a problem is found during the deployment process, it should be adequately reported so that the development team can take it into consideration for upcoming versions. In a similar spirit, the team as a whole should provide feedback. Positive reinforcement helps the team stay motivated and reaffirms the significance of their work. Even more crucial to the ongoing improvement of software releases is negative feedback.

### 2. Making Decisions Based on Data

Using data to influence decisions is another essential DevOps idea. statistics should constantly be gathered around every decision, whether choosing the best tech stack or tools to optimize your pipeline, to make sure the person's decision aligns with the metrics and past statistics of their team.

### 3. Decision-Making Centered on the Customer

A DevOps lifecycle should have the customer at its center of attention. Decisions should be made with the question "Will this benefit the customer?" in mind, just as much as data. Customer input on the current product will be gathered and used to inform future optimization.Live monitoring techniques are another tool that DevOps teams employ to solve issues before they affect customers. Using other tools, the team may monitor end users' real-time interactions with the application to identify any locations where friction may be occurring. The team is then able to provide updates targeted at eliminating these pain spots because of the DevOps lifecycle's speed.

## 4. Continuous Enhancement

Constant improvement, or the notion that the team should always concentrate on new features and updates, is the core of DevOps. The incremental releases of the Agile approach are another important concept. The goal of earlier software development approaches was to produce the ideal product all at once. Although this sounds great, in practice it frequently meant that software deployments would be put on hold for an extended amount of time while problems were fixed. Rather, incremental releases let the team concentrate on meeting the customer's primary use case as quickly as possible with a minimum viable product (MVP). The team starts working on enhancements to provide the product with more value after the MVP is delivered, eventually aiming for the perfect piece of software.

## 5. Accountability Across the Life Cycle

In conventional software development methodologies, the program is coded and built by the development team. The operations team is then tasked with testing, deploying, and delivering it to the consumer. The operations team, not the developers who built the system, is responsible for fixing any bugs found during the second phase. A more sensible strategy—responsibility throughout the lifecycle—is demonstrated by DevOps. From the beginning of planning to the end of the product's existence, the entire team is accountable for it. The development and operations teams are collaborating throughout this process to upgrade the software and fix bugs. The developers that are currently working to enhance and add new features to the code are the ones best acquainted with it.

## 6. Automation

The DevOps strategy offers speed as a major advantage: faster software delivery, faster updates, and faster patching. Automation helps to achieve this momentum. Automating each and every step of the process—from code reviews to handoffs to provisioning and deployment—is the goal of DevOps teams. This not only makes the pipeline go more quickly, but it also makes team members happier in their jobs. They are spared from laborious, manual labor. As an alternative, they might concentrate on higher-order duties like organizing upcoming upgrades and investigating novel technologies to incorporate into the program. Using DevOps solutions that are designed with automation in mind is the simplest path.

## 7. Failure as a Learning Opportunity

A flexible approach to development is DevOps. Just as software is always getting better, so too are processes always getting adjusted. Keeping this flexibility involves seeing failure as a chance to grow and learn. Encourage taking risks when they are appropriate, as opposed to attempting to prevent failing at all costs. Although taking a risk can result in failure, it can also lead to success. You will have a better understanding of what works and what doesn't no matter how an experiment turns out. This experience will serve as additional information for your decision-making process and aid in the planning of future tactics. Naturally, early testing will allow you to fail quickly in cases when the consumer won't be impacted. If something goes wrong, this is the best time to find it rather than after deployment.

how DevOps can help organizations enhance collaboration, accelerate software delivery, and improve overall product quality:

DevOps practices enable software development (dev) and operations (ops) teams to accelerate delivery through automation, collaboration, fast feedback, and iterative improvement. Stemming from an Agile approach to software development, a DevOps process expands on the cross-functional approach of building and shipping applications in a faster and more iterative manner.

In adopting a DevOps development process, you are making a decision to improve the flow and value delivery of your application by encouraging a more collaborative environment at all stages of the development cycle. DevOps represents a change in mindset for IT culture. In building on top of Agile, lean practices, and systems theory, DevOps focuses on incremental development and rapid delivery of software. Success relies on the ability to create a culture of accountability, improved collaboration, empathy, and joint responsibility for business outcomes.

## Question 2

a) Describe the process of setting up a version control system for a software development project. Compare and contrast centralized version control systems (e.g., SVN) and distributed version control systems (e.g., Git) in terms of advantages and disadvantages.

Software code tracking and management is referred to as version control, or source control. Software teams can better manage source code changes over time with the use of version control systems. Version control solutions facilitate faster and more intelligent software teamwork as development environments have advanced. DevOps teams find them especially helpful since they enable faster development times and more successful deployments.

Every change made to the code is tracked by version control software in a unique type of database. When an error occurs, engineers can go back in time and compare previous iterations of the code to assist correct the problem with the least amount of disturbance to the other members of the team.

It is simpler to learn how to use centralized version control than distributed. Working with DVCS may be confusing at first since you will need to memorize all of the commands for every operation if you are a beginner. Both setting up and learning CVCS is simple.

The main benefit of DVCS is that it offers flexibility and the ability to work offline. Since you have the complete history of the code on your hard drive, any modifications you make to your own server or repository won't need an internet connection—this isn't the case with CVCS.

Because you don't have to contact the remote server for every command, DVCS is quicker than CVCS. Since everything is done locally, you can work more quickly than with CVCS.

DVCS makes working on branches simple. Before merging all of the "sets of changes" to the remote server, developers can exchange their changes with one another because they have access to the complete history of the code in DVCS. Working on branches in CVCS is challenging and time-consuming since direct communication with the server is necessary.

With DVCS, downloading the entire project can take longer and require more space if the project has a lengthy history or contains large binary files. In contrast, with CVCS, you only need to download a few lines of code because you don't need to save the entire project or history on your own server, saving extra space is not necessary.

You can still access the backup or whole history of the code from your local repository or server, where the complete revision of the code is already kept, in the event that the main server dies or goes down in DVCS. This isn't the case with CVCS; all code history is stored on a single remote server.

Merge conflicts with other developer’s code are less in DVCS. Because every developer works on their own piece of code. Merged conflicts are more in CVCS in comparison to DVCS.

In DVCS, sometimes developers take advantage of having the entire history of the code and they may work for too long in isolation which is not a good thing. This is not the case with CVCS.

b) Discuss the importance of code reviews in a DevOps environment and explain how code reviews contribute to better software quality and team collaboration. Provide best practices for conducting effective code reviews.

First of all, they encourage team members to work together. They also guarantee uniformity and quality of code. Code reviews also aid in the early detection and correction of problems during the development process. To sum up, their benefits are essential for producing dependable, superior software.

Good practices for reviewing code

1. Set goals and standards.

It is essential to choose key indicators and establish clear objectives prior to putting a code review process into place. Among the objectives are appropriate coding standards for the business. Establishing standards ensures that every software product produced by the organization satisfies its standards.

2. Outline Expectations and Goals

It's critical to communicate expectations and goals. If team members are not informed of expectations and goals, there may be confusion about the outcome. It is simpler for a developer to finish a task when they are aware of what is expected of them.

3. Explain the Code Review Procedure

It's time to establish a code review procedure now that everyone is aware of the objectives and expectations. A clear code review procedure will make it easier for everyone to stay on task and cut down on the amount of time wasted on technical debt.

4. Utilize a checklist for code reviews.

A clear checklist is necessary for an effective code review. The reviewer can utilize this checklist to ensure that nothing is overlooked.

The Ultimate Code Review Checklist is a recommended read.

5. Before the review, authors ought to annotate the source code.

Throughout the software development cycle, annotation can be a helpful tool for both the reviewer and the developer. It facilitates the code reviewer's comprehension of the code and the functions of each code block. While adding annotations to code is encouraged, do not go overboard.

6. Spend no more than 60 minutes reviewing at a time.

It is common knowledge that trying to work nonstop for extended periods of time might cause one's efficiency to decline. The idea remains the same while attempting to review code. Conducting code reviews for longer than sixty minutes is not advised. According to research, the reviewer's performance may decrease after the 60-minute mark and some faults may go unreported.

7. Create a procedure for addressing any flaws discovered.

The final goal is to fix the faults found after a code review process. Fixing the flaws will be ensured in the most efficient manner with the least amount of technical debt if a specified methodology is in place.

8. Promote a culture of favorable code reviews

Code reviews are a common tool used by businesses to assess a developer's performance. Code reviews, however, ought to be applied to more. It ought to be applied to create a learning atmosphere. It should be evident that they learn from their mistakes and take steps to prevent them from happening again, as opposed to merely telling them what went wrong.

9. Automate to Reduce Time Spent

Codegrip and other automated code review tools are invaluable resources for any software organization. Code review times can be cut down to a few seconds with the use of tools like these. In less than a minute, they may uncover flaws in the entire codebase and offer fixes for them.

## Question 3

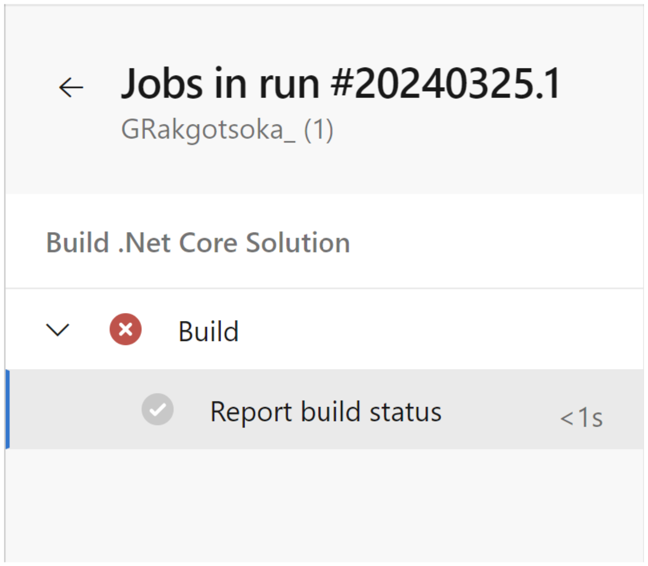
a) Create a sample CI/CD pipeline using Azure Pipelines or GitHub Actions (choose either one). Your pipeline should

include the following stages:

• Build: Compile the source code and generate artifacts.

• Test: Run unit tests and code quality checks (e.g., code coverage, code analysis).

• Deploy: Deploy the application to a staging environment.



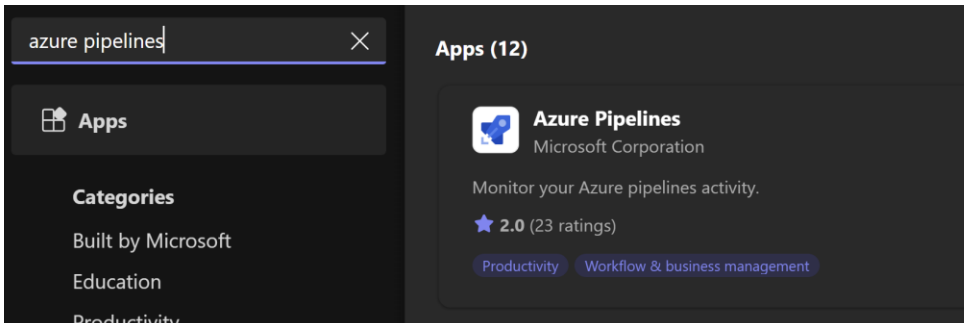
Jobs in run #20240325.1
GRakgotsoka_ (1)

Build .Net Core Solution

Build

Report build status <1s

b) In your CI/CD pipeline, integrate a notification mechanism to alert the team when a build or deployment fails.



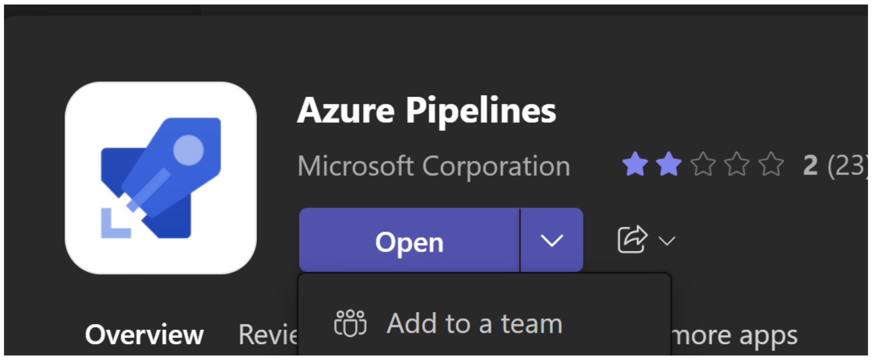
azure pipelines

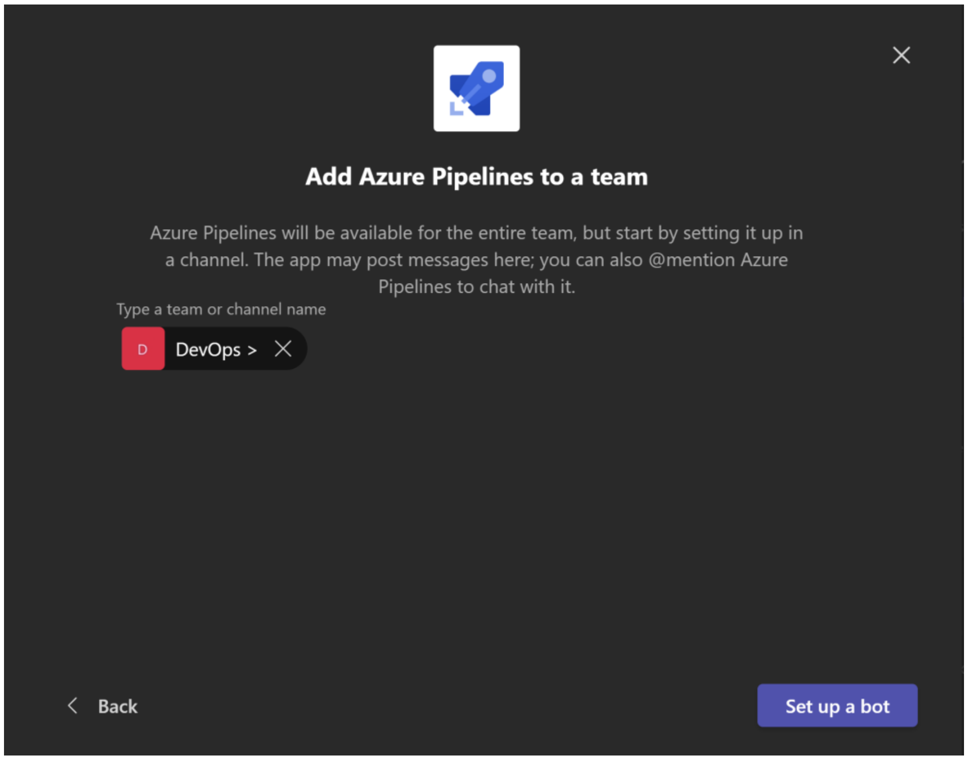
Apps

Categories
Built by Microsoft
Education

Apps (12)

Azure Pipelines
Microsoft Corporation
Monitor your Azure pipelines activity.
2.0 (23 ratings)
Productivity
Workflow & business management




Add Azure Pipelines to a team

Azure Pipelines will be available for the entire team, but start by setting it up in a channel. The app may post messages here; you can also @mention Azure Pipelines to chat with it.

Type a team or channel name

D DevOps >

< Back

Set up a bot

c) Explain the advantages and challenges of using infrastructure as code (IaC) in a CI/CD pipeline. Provide specific examples of IaC tools that can be integrated into the pipeline.

What are the Benefits of IaC?

It offers faster speed and consistency: The goal of IaC is to make things faster by eliminating manual processes and eliminating the slack in the process. A code-based approach makes it easier to get more done in less time. Users won't need to wait on the IT Admin to manually complete the task at hand before he can get to the next one. This also means faster and regular iteration.

Effective software development lifecycle: IaC gives developers more control. The more dependable and consistent the infrastructure provisioning is, the more developers can concentrate on creating applications. Additionally, they can save time and effort while maintaining total control by scripting once and using that code several times.

Decreased administrative overhead: To oversee and administer storage, networking, computing, and more hardware and middleware layers, administrators were required in the world of data centers. The necessity for these various responsibilities is eliminated byIaC. Now, those administrators may concentrate on choosing the next cutting-edge technology to use.

Which are the main obstacles facing IaC?

There are two sides to every coin. Even while IaC greatly benefits the IT environment, there are certain difficulties that must be taken into consideration. Don't forget to take into consideration your own IT circumstances, which may or may not make the following information applicable (such as your organization's size, state, and technology adoption lifecycle).

Dependency on coding languages: As already mentioned before, developers now have more power. Similar to this, you must be an adept coder since IaC depends more on code. If you don't have a developer bench ready, this could have a longer learning curve. HashiCorp Configuration Languages (HCL), YAML, Ruby, and JSON are a few of the languages utilized for IaC. Your ability to use IaC effectively may be hampered by the lack of these skill sets. Is your plan to shift from development to a serverless model as well? Before you dive into IaC, consider the strategic direction you are moving in. If your ultimate purpose is different, perhaps you should avoid making the IaC pit stop.

Security assessment procedures: In the new IaC environment, your outdated security tools and procedures might not be sufficient. If the resources that have been provisioned are not being used by the appropriate apps, you may need to verify this manually. To have your old security tools matched to IaC, it may take several cycles, even though manual inspection is a step that builds confidence. Additionally, keep in mind that IaC is more dynamic than the provisioning and management procedures you currently use. Either optimally or even more quickly, it can be exploited. As a result, you might need to go above and above to make sure you're setting boundaries for total governance.

IaC monitoring can be difficult: To elaborate on the previous point, you might require extra tools to keep track of who is provisioning what, where, when, and how much it costs. Tracking capacity and consumption with worksheets or other outdated monitoring tools may be difficult for you. Furthermore, you may need to consider stronger monitoring solutions if your business is international.

# Completed Declaration of Authenticity

I Gabriella Rakgotsoka _ hereby
(FULL NAME)

declare that the contents of this assignment DOE522_FA is entirely my own work except for the following documents: (List the documents and page numbers of work in this portfolio
that were generated in a group)

| **Activi ty** | **Da te** |
|---|---|
| | |
| | |
| | |
| | |
| | |
| | |

[illegible]
Signature: Date: 2024/03/25