CTU 2024

Software Development

SUBJECT NAME: DevOps Engineering Semester 2

SUBJECT CODE: DOE522

Edward Nhlapo

Student Number – 20220865

20220865@ctucareer.co.za

20 March 2024:

<u>Question 1</u>.

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

Answer:

DevOps is a set of practices aimed at streamlining collaboration between software development (Dev) and IT operations (Ops) teams. Its key principles and benefits are integral to modern software development practices.

Key Principles of DevOps:

Infrastructure as Code (IaC): IaC involves managing and provisioning infrastructure through code and automation tools.

Automation: Automation is central to DevOps, enabling teams to automate repetitive tasks such as building, testing, and deployment processes. By automating these tasks, teams can increase efficiency, reduce errors, and accelerate the delivery of software.

Continuous Integration (CI) and Continuous Delivery (CD): CI/CD practices involve frequently integrating code changes into a shared repository and automating the testing and deployment of these changes. This allows teams to deliver software updates more rapidly, ensuring a shorter time-to-market and enabling faster response to customer feedback.

Collaboration: DevOps emphasizes breaking down silos between development and operations teams, promoting a culture of collaboration and shared responsibility. This collaboration extends across the entire software development lifecycle (SDLC), from planning and coding to testing, deployment, and monitoring.

Monitoring and Feedback: DevOps emphasizes the importance of monitoring applications and infrastructure in real-time to detect issues early and gather feedback for continuous improvement. Monitoring metrics, logs, and user feedback enable teams to make data-driven decisions and enhance the overall quality of software.

Benefits of DevOps:

Enhanced Collaboration: DevOps fosters collaboration between development, operations, and other stakeholders, leading to improved communication, alignment of goals, and shared accountability.

Accelerated Software Delivery: By automating processes, implementing CI/CD pipelines, and streamlining workflows, DevOps enables organizations to deliver software updates faster and more frequently. This rapid delivery allows teams to respond quickly to changing market demands and customer feedback.

Improved Deployment Frequency and Stability: DevOps practices such as CI/CD help increase deployment frequency while maintaining or even improving system stability. Automated testing and deployment pipelines ensure that changes are thoroughly tested and can be deployed reliably, reducing the risk of failures and downtime.

Greater Flexibility and Scalability: DevOps enables organizations to scale infrastructure and applications more efficiently by leveraging cloud services and automation. This flexibility allows teams to adapt to changing workloads and business requirements, ensuring that software can grow seamlessly with the organization.

Higher Overall Product Quality: Through continuous testing, monitoring, and feedback loops, DevOps helps improve the overall quality of software. By detecting and addressing issues early in the development process, teams can deliver more reliable, secure, and user-friendly software to customers.

Examples:

Netflix: Netflix uses DevOps to continuously deliver updates to its streaming platform, allowing it to quickly roll out new features and optimizations to millions of users worldwide. By automating testing and deployment processes, Netflix ensures a seamless and reliable streaming experience for its customers.

Amazon: Amazon's DevOps practices are well-known for their emphasis on automation, CI/CD, and infrastructure as code. By leveraging these practices, Amazon can rapidly deploy new features and updates to its e-commerce platform while maintaining high availability and scalability.

<u>**Question**</u> 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.

Answer:

Setting up a version control system (VCS) is a critical step in managing software development projects effectively. Here's a general process for setting up a VCS for a software development project:

Process of Setting Up a Version Control System:

Choose a VCS: Evaluate different version control systems based on your project requirements, team size, workflow, and other factors. Common VCS options include Git, SVN (Subversion), Mercurial, and others.

Install and Configure: Install the chosen VCS on your development environment or server. Configure the VCS settings, such as repository location, access control, and user authentication.

Create a Repository: Set up a new repository for your project. This repository will store all project files, including source code, documentation, and other assets.

Initialize Repository: Initialize the repository by adding the initial set of files and committing them to the VCS. This creates the first revision (commit) in the repository.

Collaborator Setup: Add collaborators to the repository, granting them appropriate permissions based on their roles (e.g., read-only access, write access, administrative privileges).

Define Workflow: Establish a workflow for how changes will be managed within the repository, including branching strategies, code review processes, and release management practices.

Training and Documentation: Provide training to team members on how to use the VCS effectively. Create documentation outlining best practices, guidelines, and procedures for version control within the project.

Regular Maintenance: Regularly maintain the repository by performing tasks such as backing up data, resolving conflicts, and optimizing performance.

Comparison of Centralized vs. Distributed Version Control Systems:

Centralized Version Control Systems (e.g., SVN):

Advantages:

Centralized repository: All project files are stored in a central server, making it easy to manage access control and permissions.

Simplicity: SVN has a relatively simple model compared to distributed systems, making it easier for beginners to understand and use.

Stable History: SVN maintains a linear history of changes, which can be useful for tracking revisions in chronological order.

Disadvantages:

Single Point of Failure: The central server is a single point of failure, and if it goes down, collaborators may be unable to access or update project files.

Limited Offline Capability: Collaborators may face limitations when working offline or in environments with poor connectivity since most operations require access to the central server.

Branching Complexity: Branching and merging operations can be more complex in SVN compared to distributed systems, leading to potential conflicts and difficulties in managing parallel development efforts.

Distributed Version Control Systems (e.g., Git):

Advantages:

Distributed nature: Each collaborator has a local copy of the entire repository, enabling them to work offline and perform operations independently of a central server.

Flexible Branching: Git offers lightweight branching and merging capabilities, allowing teams to easily experiment with different features and workflows without disrupting the main codebase.

Faster Operations: Many Git operations, such as commits, branching, and merging, are performed locally, resulting in faster response times and improved performance.

Disadvantages:

Complexity: Git's distributed nature and flexible branching model can be more complex to understand and manage, particularly for beginners or teams accustomed to centralized systems.

Steeper Learning Curve: Git's extensive feature set and command-line interface may require more time and effort for team members to become proficient compared to centralized systems.

Repository Size: Since each collaborator has a complete copy of the repository, Git repositories can grow larger in size, especially for projects with extensive histories or large binary files.

In summary, while both centralized and distributed version control systems offer advantages and disadvantages, the choice between them ultimately depends on factors such as project requirements, team preferences, and workflow considerations. Centralized systems like SVN may be more suitable for simpler projects with centralized control requirements, while distributed systems like Git offer greater flexibility and scalability for larger, more complex projects with distributed teams and workflows.

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.

Answer:

Code reviews play a crucial role in ensuring software quality and fostering collaboration within a DevOps environment. They involve systematically examining code changes made by developers before they are integrated into the main codebase. Here's why code reviews are important and how they contribute to better software quality and team collaboration:

Importance of Code Reviews:

Error Detection and Prevention: Code reviews help identify bugs, logic errors, and potential security vulnerabilities early in the development process, reducing the likelihood of these issues reaching production.

Knowledge Sharing and Learning: Code reviews provide an opportunity for developers to learn from each other by sharing insights, best practices, and alternative solutions. They also help onboard new team members by familiarizing them with the codebase and coding standards.

Maintainability and Code Quality: Through constructive feedback and discussions during code reviews, teams can enforce coding standards, improve code readability, and ensure consistency across the codebase. This leads to better maintainability and higher overall code quality.

Risk Mitigation: By having multiple sets of eyes review code changes, teams can mitigate the risk associated with complex or critical changes. Code reviews help ensure that changes are thoroughly tested, documented, and aligned with project requirements.

Continuous Improvement: Code reviews foster a culture of continuous improvement by encouraging feedback, collaboration, and reflection on development practices. Teams can use code reviews as a mechanism for iteratively refining their processes and enhancing their technical skills.

Best Practices for Conducting Effective Code Reviews:

Set Clear Objectives: Define the goals and expectations of code reviews, such as identifying defects, ensuring adherence to coding standards, and promoting knowledge sharing.

Review Small, Digestible Units: Break down code changes into smaller, manageable units that are easier to review. This allows reviewers to focus on specific aspects of the code and provides more meaningful feedback.

Use Code Review Tools: Utilize code review tools or integrated development environments (IDEs) that facilitate the review process, such as GitLab, GitHub, or Bitbucket. These tools offer features like inline comments, diffs, and automated code analysis.

Rotate Reviewers: Rotate reviewers regularly to ensure that code changes receive diverse perspectives and avoid reviewer fatigue. This practice also helps distribute knowledge and expertise across the team.

Provide Constructive Feedback: Offer feedback that is specific, actionable, and respectful. Focus on identifying issues, suggesting improvements, and explaining the rationale behind suggestions. Avoid personal criticism and maintain a collaborative tone.

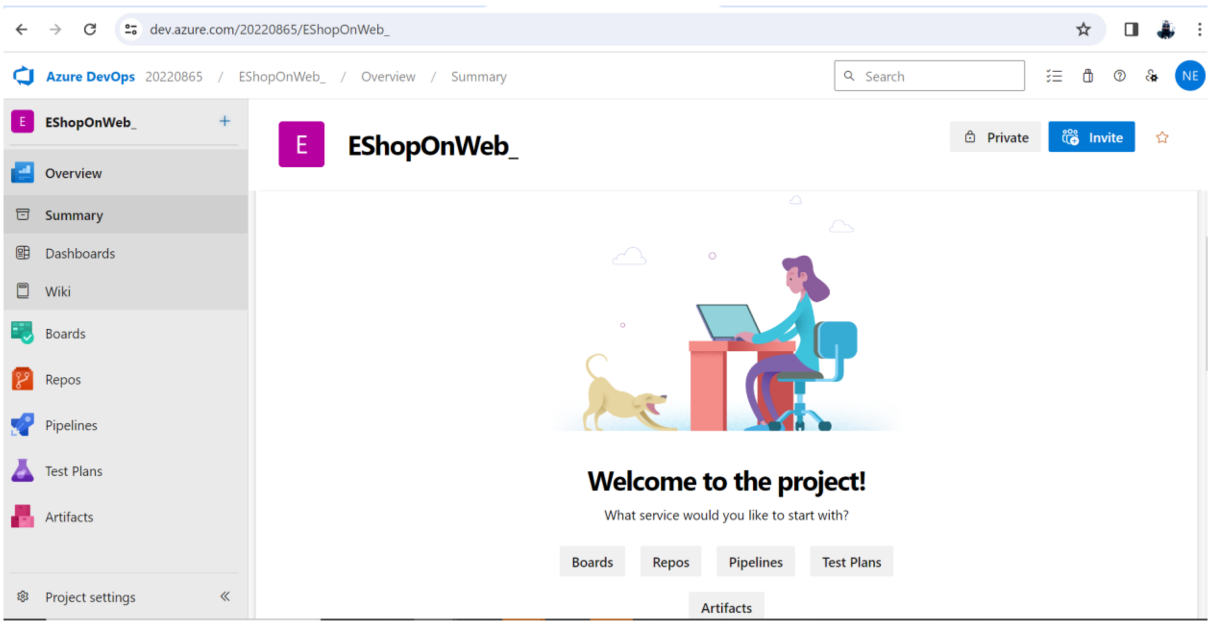
Encourage Discussion: Encourage open dialogue and discussion during code reviews to clarify requirements, address concerns, and share insights. Foster a culture where developers feel comfortable asking questions and seeking clarification.

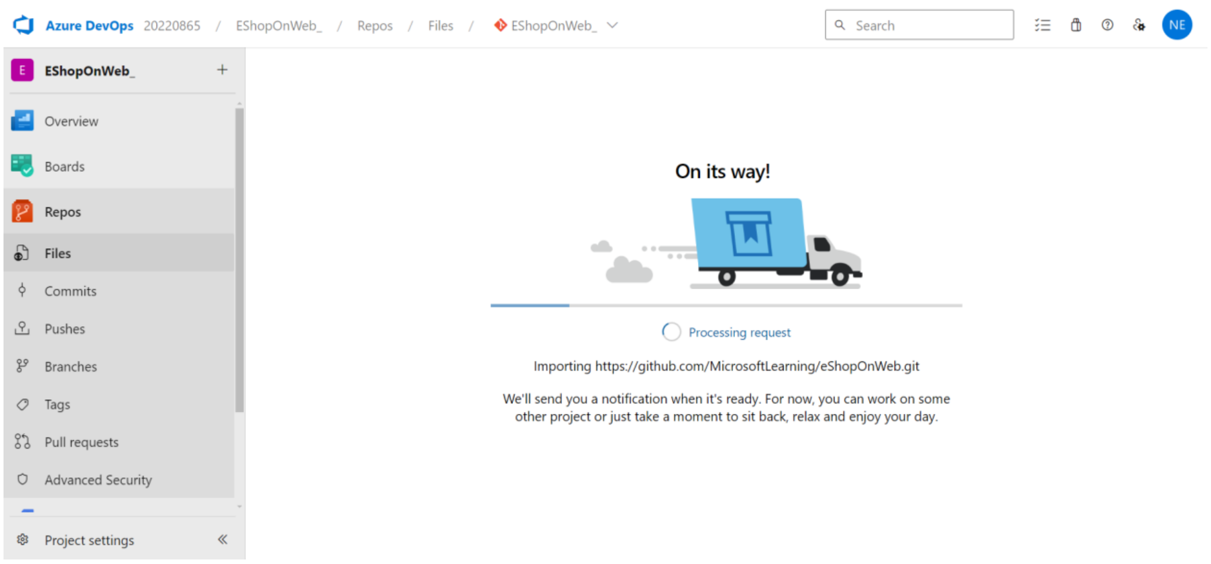
Follow Up on Feedback: Ensure that feedback provided during code reviews is addressed promptly. Developers should actively engage with reviewers, address comments, and incorporate suggested changes into their code.

Document Decisions and Learnings: Document key decisions, lessons learned, and best practices discovered during code reviews. This knowledge can be valuable for future reference, onboarding new team members, and improving development processes.

In conclusion, code reviews are an essential practice in DevOps environments, contributing to better software quality, reduced risks, and enhanced collaboration among team members. By following best practices and fostering a culture of continuous improvement, teams can leverage code reviews as a valuable tool for delivering high-quality software efficiently.

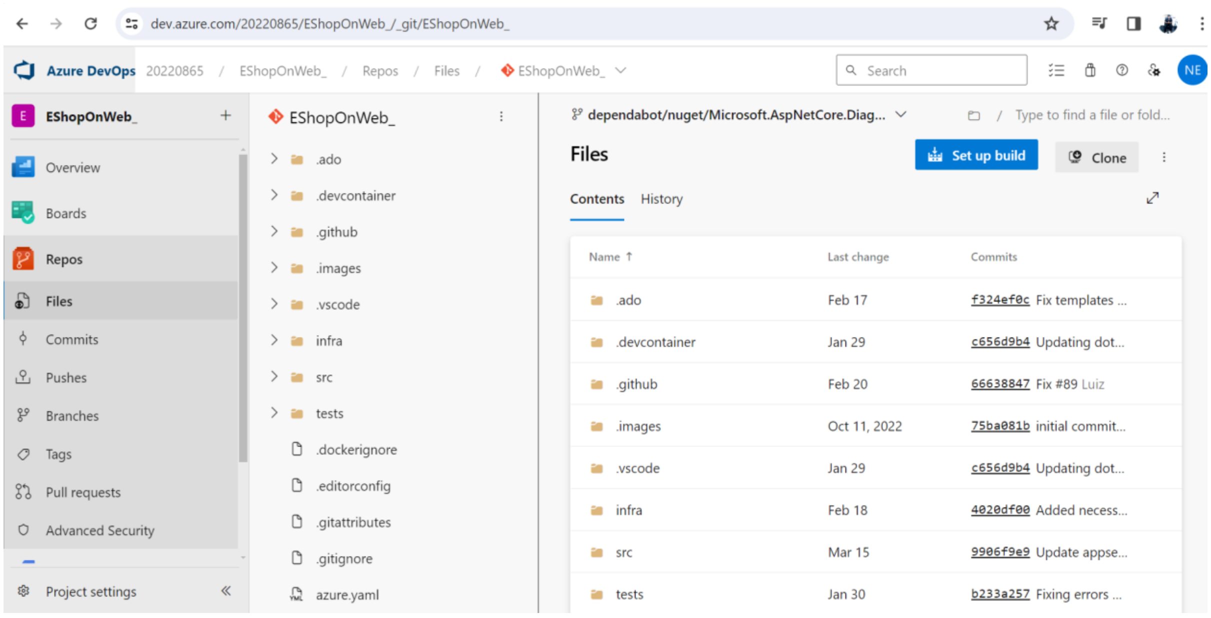
Question 3 ():



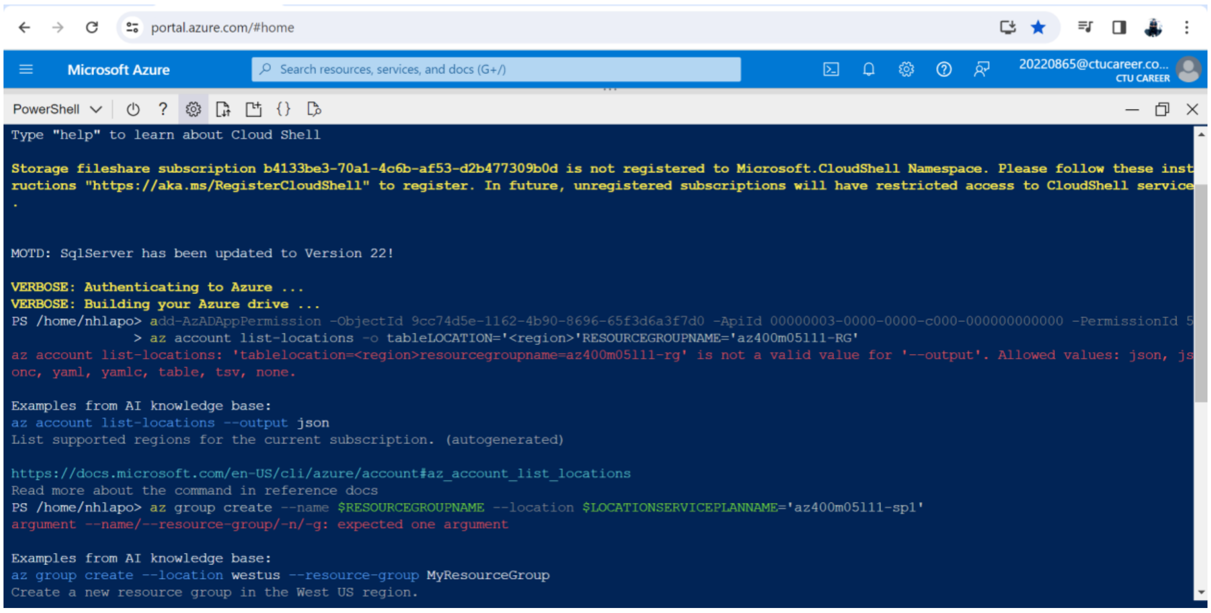


Importing https://github.com/MicrosoftLearning/eShopOnWeb.git

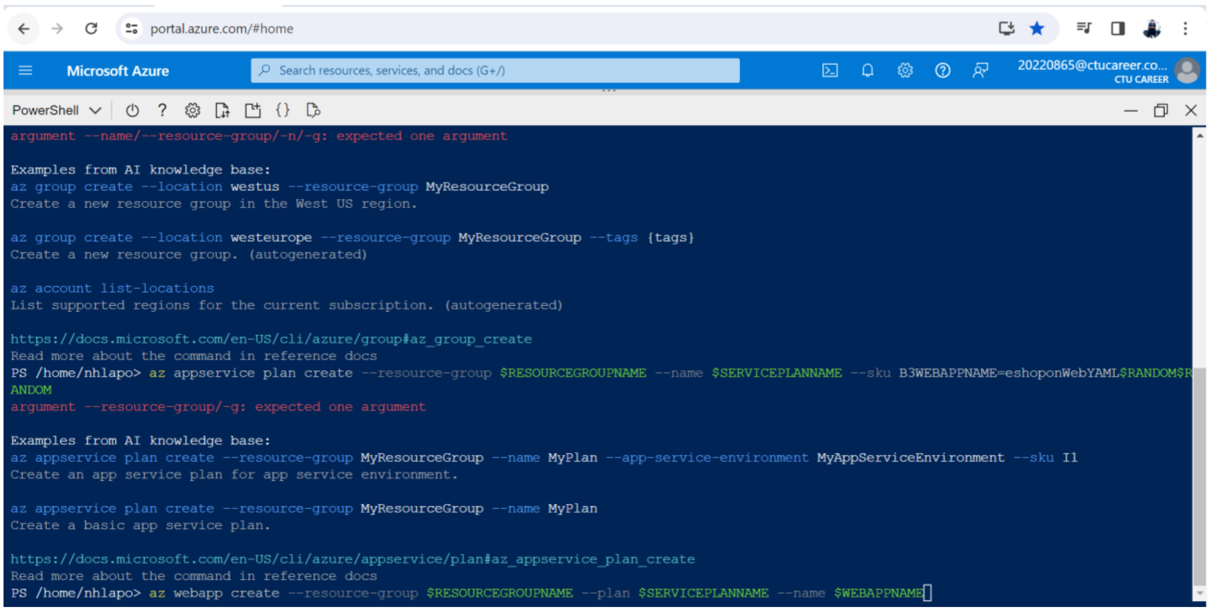
We'll send you a notification when it's ready. For now, you can work on some other project or just take a moment to sit back, relax and enjoy your day.



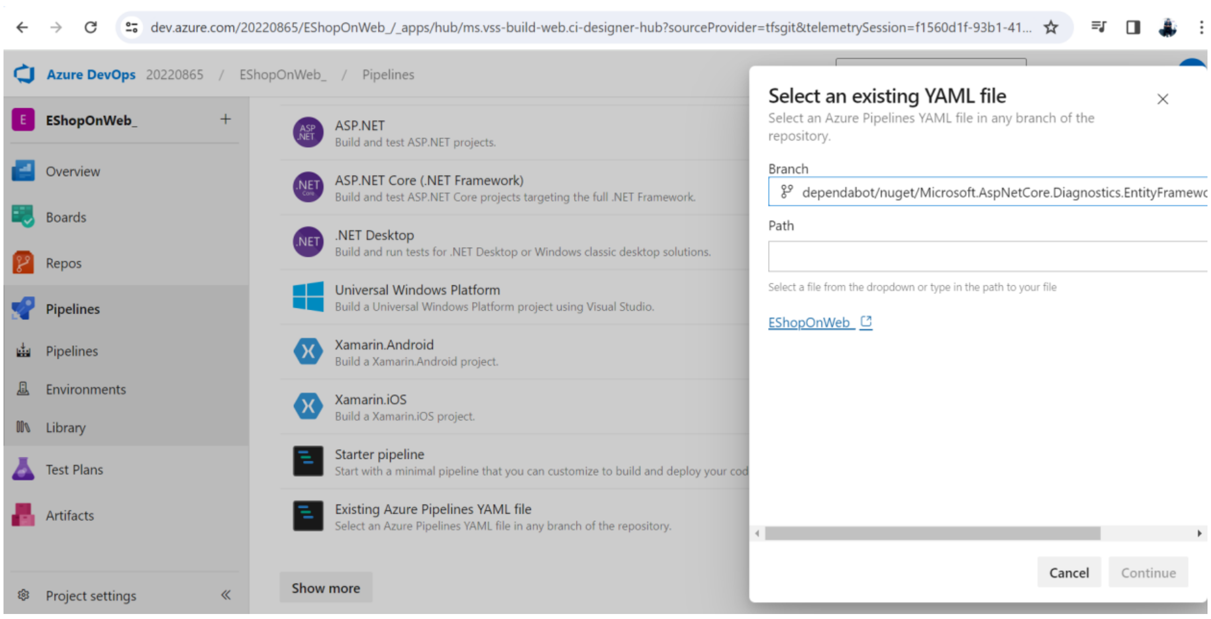
Name ↑	Last change	Commits
.ado	Feb 17	f324ef0c Fix templates ...
.devcontainer	Jan 29	c656d9b4 Updating dot...
.github	Feb 20	66638847 Fix #89 Luiz
.images	Oct 11, 2022	75ba081b initial commit...
.vscode	Jan 29	c656d9b4 Updating dot...
infra	Feb 18	4020df00 Added necess...
src	Mar 15	9906f9e9 Update appse...
tests	Jan 30	b233a257 Fixing errors ...



```
Type "help" to learn about Cloud Shell

Storage fileshare subscription b4133be3-70a1-4c6b-af53-d2b477309b0d is not registered to Microsoft.CloudShell Namespace. Please follow these inst
ructions "https://aka.ms/RegisterCloudShell" to register. In future, unregistered subscriptions will have restricted access to CloudShell service
.


MOTD: SqlServer has been updated to Version 22!

VERBOSE: Authenticating to Azure ...
VERBOSE: Building your Azure drive ...
PS /home/nhlapo> add-AzADAppPermission -ObjectId 9cc74d5e-1162-4b90-8696-65f3d6a3f7d0 -ApiId 00000003-0000-0000-c000-000000000000 -PermissionId 5
               > az account list-locations -o tableLOCATION='<region>'RESOURCEGROUPNAME='az400m05111-RG'
az account list-locations: 'tablelocation=<region>resourcegroupname=az400m05111-rg' is not a valid value for '--output'. Allowed values: json, js
onc, yaml, yamlc, table, tsv, none.

Examples from AI knowledge base:
az account list-locations --output json
List supported regions for the current subscription. (autogenerated)

https://docs.microsoft.com/en-US/cli/azure/account#az_account_list_locations
Read more about the command in reference docs
PS /home/nhlapo> az group create --name $RESOURCEGROUPNAME --location $LOCATIONSERVICEPLANNAME='az400m05111-spl'
argument --name/--resource-group/-n/-g: expected one argument

Examples from AI knowledge base:
az group create --location westus --resource-group MyResourceGroup
Create a new resource group in the West US region.
```



```
argument --name/--resource-group/-n/-g: expected one argument

Examples from AI knowledge base:
az group create --location westus --resource-group MyResourceGroup
Create a new resource group in the West US region.

az group create --location westeurope --resource-group MyResourceGroup --tags {tags}
Create a new resource group. (autogenerated)

az account list-locations
List supported regions for the current subscription. (autogenerated)

https://docs.microsoft.com/en-US/cli/azure/group#az_group_create
Read more about the command in reference docs
PS /home/nhlapo> az appservice plan create --resource-group $RESOURCEGROUPNAME --name $SERVICEPLANNAME --sku B3WEBAPPNAME=eshoponWebYAML$RANDOM$R
ANDOM
argument --resource-group/-g: expected one argument

Examples from AI knowledge base:
az appservice plan create --resource-group MyResourceGroup --name MyPlan --app-service-environment MyAppServiceEnvironment --sku I1
Create an app service plan for app service environment.

az appservice plan create --resource-group MyResourceGroup --name MyPlan
Create a basic app service plan.

https://docs.microsoft.com/en-US/cli/azure/appservice/plan#az_appservice_plan_create
Read more about the command in reference docs
PS /home/nhlapo> az webapp create --resource-group $RESOURCEGROUPNAME --plan $SERVICEPLANNAME --name $WEBAPPNAME
```



dev.azure.com/20220865/EShopOnWeb_/_apps/hub/ms.vss-build-web.ci-designer-hub?sourceProvider=tfsgit&telemetrySession=f1560d1f-93b1-41...
Azure DevOps 20220865 / EShopOnWeb_ / Pipelines
EShopOnWeb_
Overview
Boards
Repos
Pipelines
Pipelines
Environments
Library
Test Plans
Artifacts
Project settings
Show more

ASP.NET
Build and test ASP.NET projects.

ASP.NET Core (.NET Framework)
Build and test ASP.NET Core projects targeting the full .NET Framework.

.NET Desktop
Build and run tests for .NET Desktop or Windows classic desktop solutions.

Universal Windows Platform
Build a Universal Windows Platform project using Visual Studio.

Xamarin.Android
Build a Xamarin.Android project.

Xamarin.iOS
Build a Xamarin.iOS project.

Starter pipeline
Start with a minimal pipeline that you can customize to build and deploy your cod

Existing Azure Pipelines YAML file
Select an Azure Pipelines YAML file in any branch of the repository.

Select an existing YAML file
Select an Azure Pipelines YAML file in any branch of the repository.

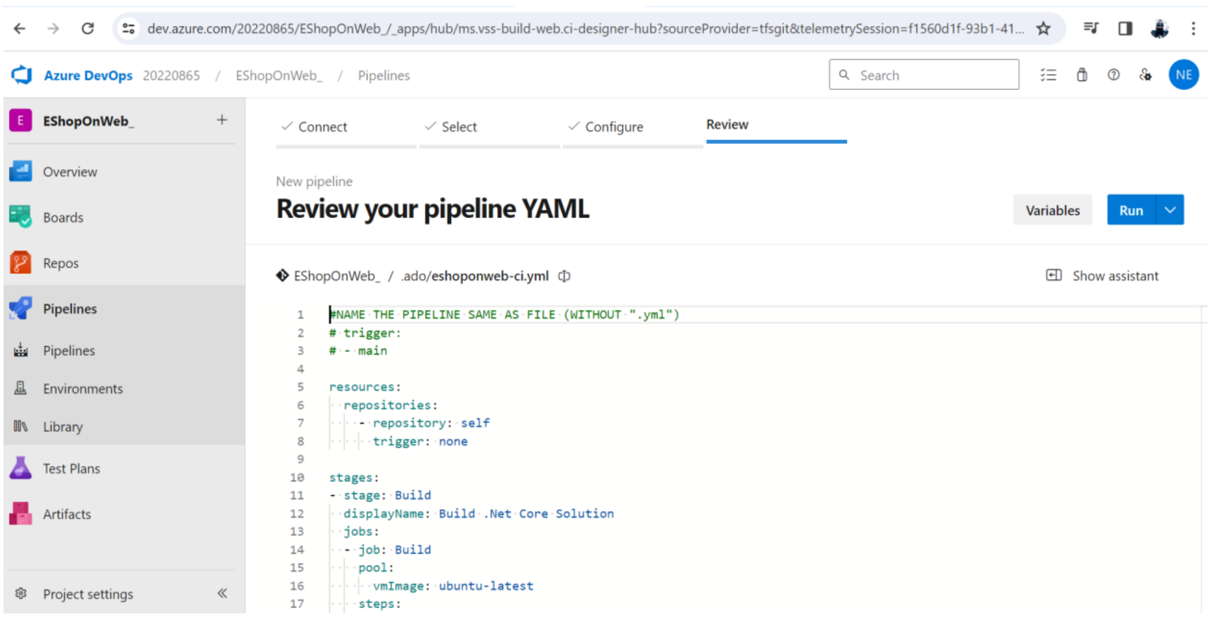
Branch
dependabot/nuget/Microsoft.AspNetCore.Diagnostics.EntityFramewo

Path

Select a file from the dropdown or type in the path to your file

EShopOnWeb_

Cancel Continue

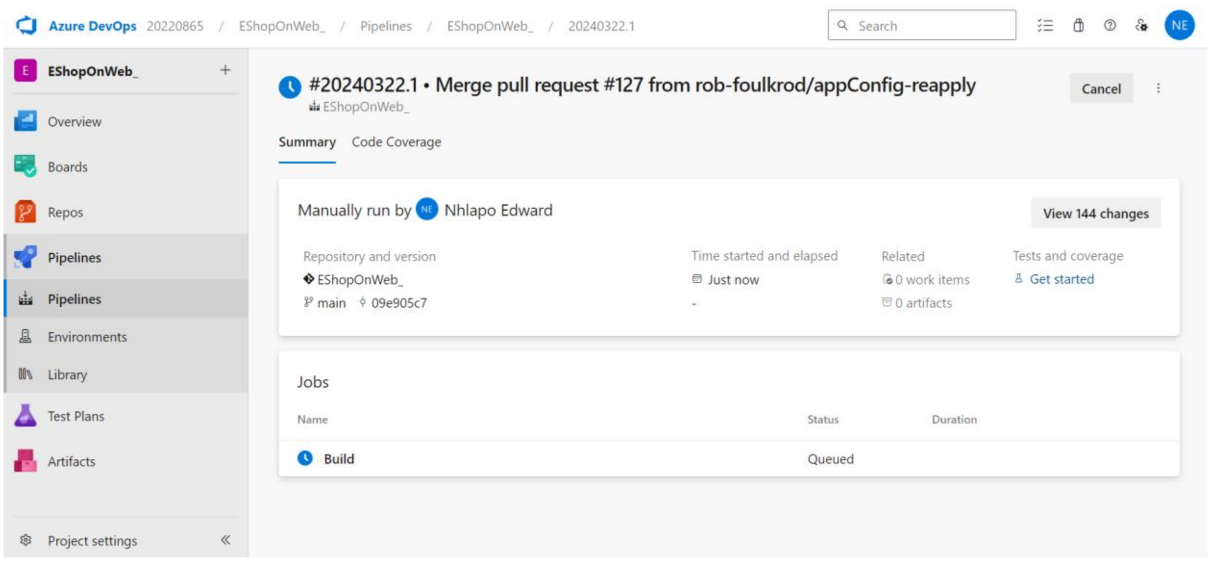

dev.azure.com/20220865/EShopOnWeb_/_apps/hub/ms.vss-build-web.ci-designer-hub?sourceProvider=tfsgit&telemetrySession=f1560d1f-93b1-41...
Azure DevOps 20220865 / EShopOnWeb_ / Pipelines
Search
NE
EShopOnWeb_
Overview
Boards
Repos
Pipelines
Pipelines
Environments
Library
Test Plans
Artifacts
Project settings
Connect
Select
Configure
Review
Variables
Run
Show assistant

New pipeline

Review your pipeline YAML

EShopOnWeb_ / .ado/eshoponweb-ci.yml

```
#NAME THE PIPELINE SAME AS FILE (WITHOUT ".yml")
# trigger:
# - main

resources:
  repositories:
    - repository: self
      trigger: none

stages:
- stage: Build
  displayName: Build .Net Core Solution
  jobs:
  - job: Build
    pool:
      vmImage: ubuntu-latest
    steps:
```



#20240322.1 • Merge pull request #127 from rob-foulkrod/appConfig-reapply

EShopOnWeb_

Manually run by NE Nhlapo Edward

Repository and version: EShopOnWeb_, main, 09e905c7

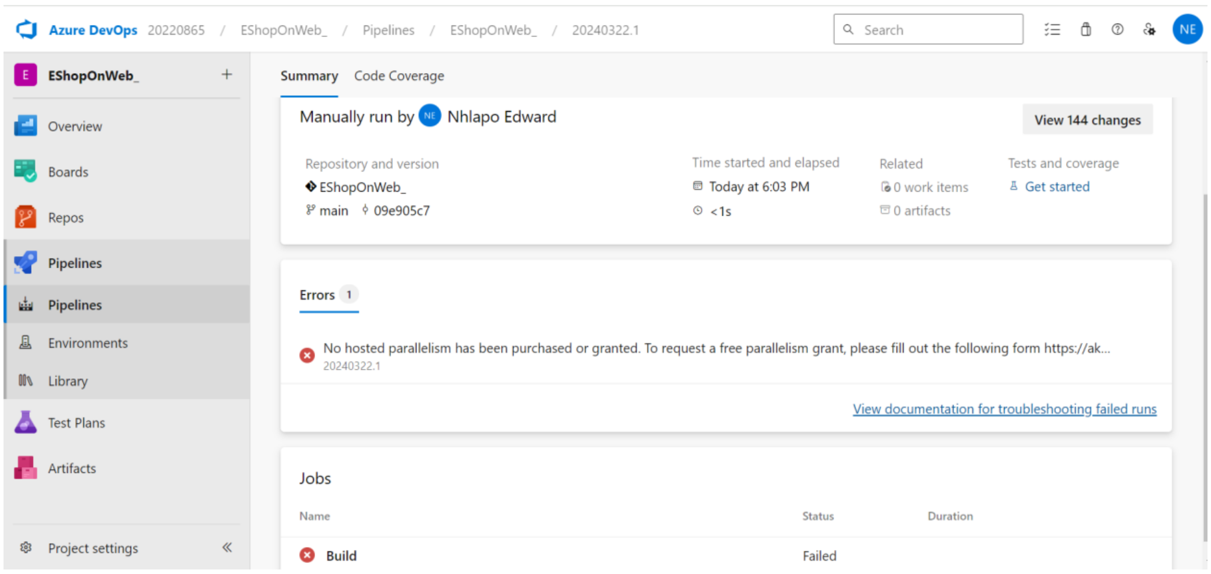
Time started and elapsed: Just now, -

Related: 0 work items, 0 artifacts

Tests and coverage: Get started

Jobs

Name	Status	Duration
Build	Queued	



Manually run by Nhlapo Edward

View 144 changes

Repository and version	Time started and elapsed	Related	Tests and coverage
EShopOnWeb_	Today at 6:03 PM	0 work items	Get started
main 09e905c7	<1s	0 artifacts	

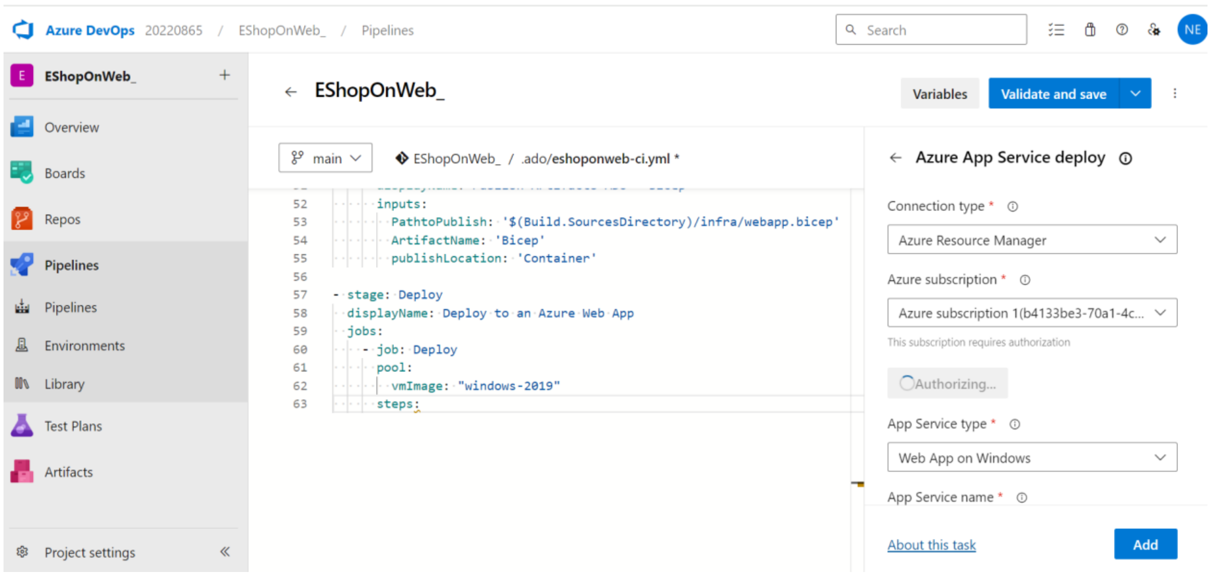
Errors 1

No hosted parallelism has been purchased or granted. To request a free parallelism grant, please fill out the following form https://ak...
20240322.1

View documentation for troubleshooting failed runs

Jobs

Name	Status	Duration
Build	Failed	



EShopOnWeb_

EShopOnWeb_ / .ado/eshoponweb-ci.yml *

```
          inputs:
            PathtoPublish: '$(Build.SourcesDirectory)/infra/webapp.bicep'
            ArtifactName: 'Bicep'
            publishLocation: 'Container'

- stage: Deploy
  displayName: Deploy to an Azure Web App
  jobs:
    - job: Deploy
      pool:
        vmImage: "windows-2019"
      steps:
```

Azure App Service deploy

Connection type *
Azure Resource Manager

Azure subscription *
Azure subscription 1(b4133be3-70a1-4c...
This subscription requires authorization

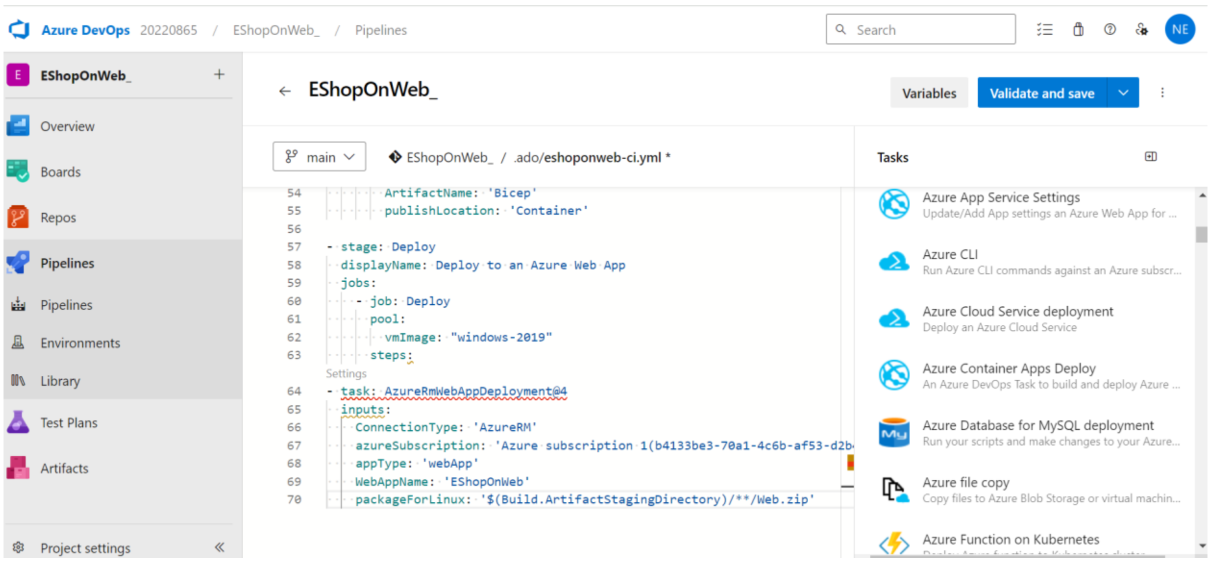
Authorizing...

App Service type *
Web App on Windows

App Service name *

About this task

Add



EShopOnWeb_

EShopOnWeb_ / .ado/eshoponweb-ci.yml *

```
              ArtifactName: 'Bicep'
              publishLocation: 'Container'

  - stage: Deploy
    displayName: Deploy to an Azure Web App
    jobs:
      - job: Deploy
        pool:
          vmImage: "windows-2019"
        steps:
  Settings
  - task: AzureRmWebAppDeployment@4
    inputs:
      ConnectionType: 'AzureRM'
      azureSubscription: 'Azure subscription 1(b4133be3-70a1-4c6b-af53-d2b
      appType: 'webApp'
      WebAppName: 'EShopOnWeb'
      packageForLinux: '$(Build.ArtifactStagingDirectory)/**/Web.zip'
```

Azure App Service Settings
Update/Add App settings an Azure Web App for ...

Azure CLI
Run Azure CLI commands against an Azure subscr...

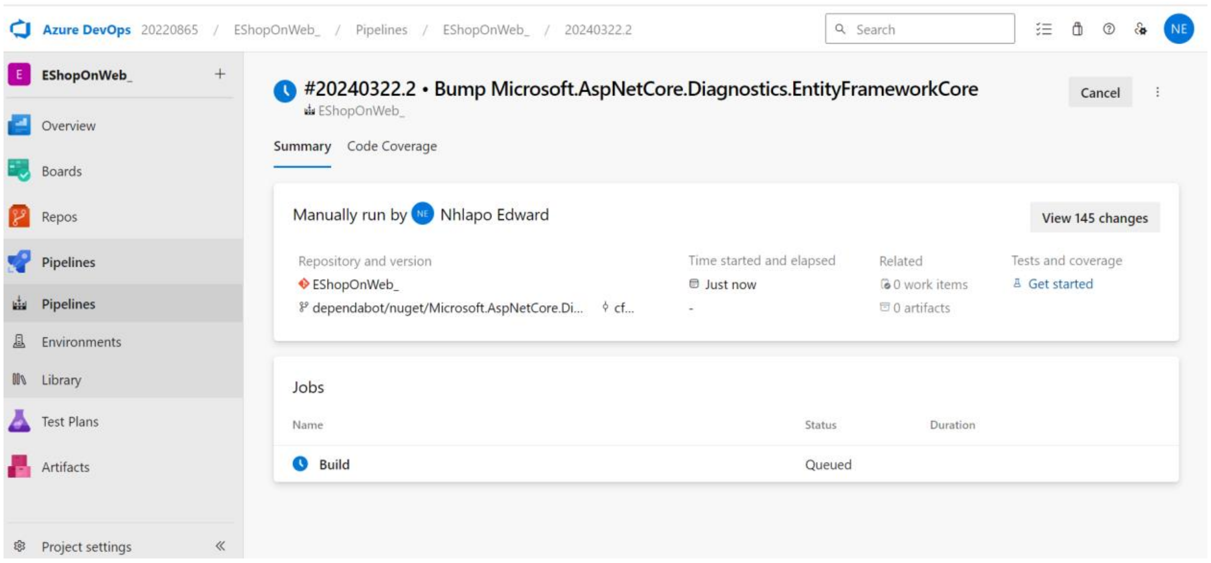
Azure Cloud Service deployment
Deploy an Azure Cloud Service

Azure Container Apps Deploy
An Azure DevOps Task to build and deploy Azure ...

Azure Database for MySQL deployment
Run your scripts and make changes to your Azure...

Azure file copy
Copy files to Azure Blob Storage or virtual machin...

Azure Function on Kubernetes



#20240322.2 • Bump Microsoft.AspNetCore.Diagnostics.EntityFrameworkCore

EShopOnWeb_

Manually run by Nhlapo Edward

Repository and version	Time started and elapsed	Related	Tests and coverage
EShopOnWeb_	Just now	0 work items	Get started
dependabot/nuget/Microsoft.AspNetCore.Di... cf...	-	0 artifacts	

Jobs

Name	Status	Duration
Build	Queued	

Advantages of Using Infrastructure as Code (IaC) in a CI/CD Pipeline:

Version Control: Infrastructure configurations are treated as code and stored in version control repositories, enabling teams to track changes, collaborate effectively, and roll back to previous versions if necessary. This promotes transparency, accountability, and auditability of infrastructure changes.

Automation: IaC tools automate the provisioning, configuration, and management of infrastructure resources, reducing manual effort, minimizing errors, and speeding up deployment processes. Automated provisioning enables teams to rapidly scale infrastructure resources up or down in response to changing workload demands.

Scalability: With IaC, scaling infrastructure resources becomes more efficient and manageable. Infrastructure configurations can be dynamically adjusted and optimized based on workload patterns, ensuring optimal resource utilization and cost efficiency.

Portability: IaC enables infrastructure definitions to be shared and reused across different environments and cloud providers. This promotes portability and avoids vendor lock-in, allowing organizations to migrate workloads seamlessly between cloud platforms or on-premises environments.

Consistency and Reproducibility: IaC allows for consistent and repeatable provisioning of infrastructure resources across different environments. This ensures that development, testing, and production environments are identical, reducing the risk of configuration drift and inconsistencies.

Challenges of Using Infrastructure as Code (IaC) in a CI/CD Pipeline:

Learning Curve: Adopting IaC requires learning new tools, languages, and best practices, which can be challenging for teams without prior experience. There may be a learning curve associated with understanding the syntax, concepts, and capabilities of IaC tools.

Complexity: Managing infrastructure as code can become complex, especially for large-scale deployments with multiple components and dependencies. Designing and maintaining modular, reusable infrastructure configurations requires careful planning and architectural considerations.

Testing and Validation: Ensuring the correctness and stability of infrastructure code requires thorough testing and validation procedures. Testing infrastructure changes in isolation and in integration with application code can be challenging, requiring robust testing frameworks and strategies.

State Management: Managing the state of infrastructure resources and ensuring consistency between the desired and actual state can be challenging, particularly in distributed or dynamic environments. Handling resource dependencies, stateful services, and idempotent operations requires careful state management practices.

Tool Selection: Choosing the right IaC tools and technologies that align with the organization's requirements, preferences, and existing infrastructure can be a daunting task. Evaluating and selecting suitable IaC tools involves considering factors such as scalability, extensibility, community support, and integration capabilities.

Examples of IaC Tools for CI/CD Pipelines:

Terraform: Terraform is a widely used IaC tool that allows infrastructure to be defined using a declarative configuration language. It supports multiple cloud providers and can manage both infrastructure and services.

Azure Resource Manager (ARM) Templates: ARM Templates are JSON files that define the Azure resources and configurations needed for a solution. They can be used to automate the deployment and management of Azure infrastructure.

Google Cloud Deployment Manager: Google Cloud Deployment Manager allows users to define the resources and services required for their applications using YAML or Jinja2 templates. It automates the deployment and updates of Google Cloud Platform resources.

AWS CloudFormation: AWS CloudFormation provides a way to model and provision AWS resources using templates. It automates the deployment and updates of AWS infrastructure in a controlled and predictable manner.

Source for research – Google search engine

- O’Reilly.
- https://www.netapp.com/devops-solutions/what-is-devops/
- https://www.cloudbolt.io/blog/3-advantages-and-challenges-of-infrastructure-as-code-iac/
- https://github.blog/2022-02-02-build-ci-cd-pipeline-github-actions-four-steps/
-