Anatolii Yakubyshyn

Practical Task 1: Create and Initialize a New Repository with a README

PreRequirements:

We have created a Terraform pipeline that requires a remote state on a storage account—this might be useful for you. Please complete the following steps:

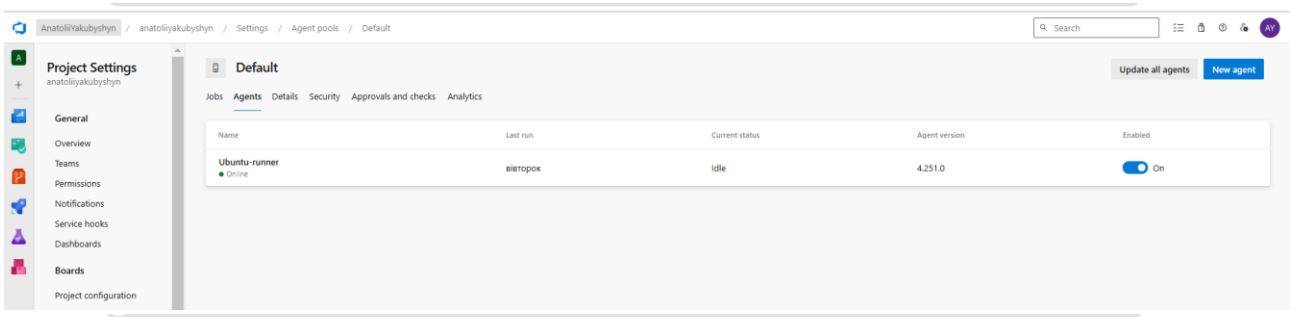
1. Create a virtual machine (Linux).

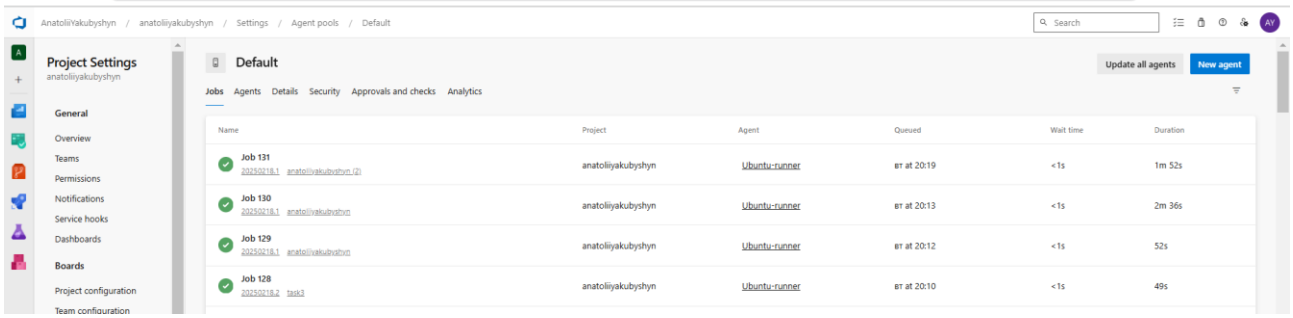
2. Set up a storage account for the state.

3. Install a self-hosted agent (Linux). pool: name: MyLinux demands: - agent.name -equals myAgent

4. Review the pull request I created with the pipeline.

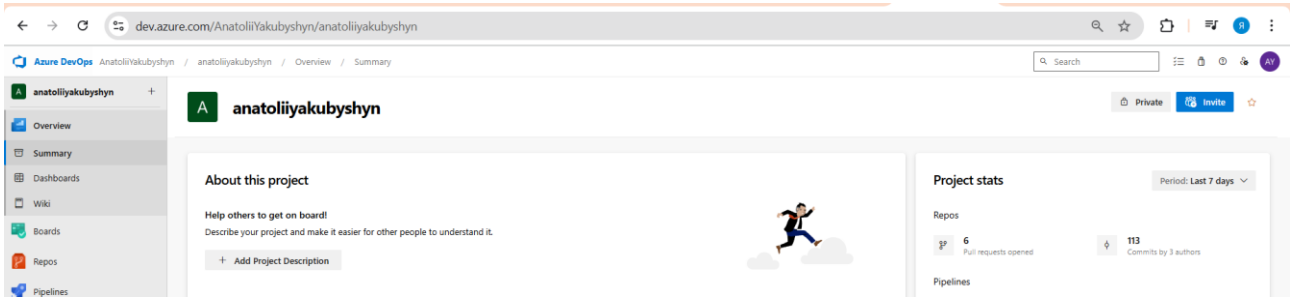
5. Merge it and update the Agent name and variable group. 6. Ensure the pipeline runs successfully.



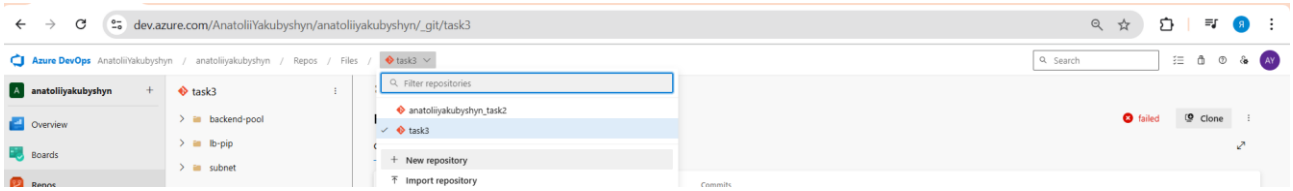


Requirements:

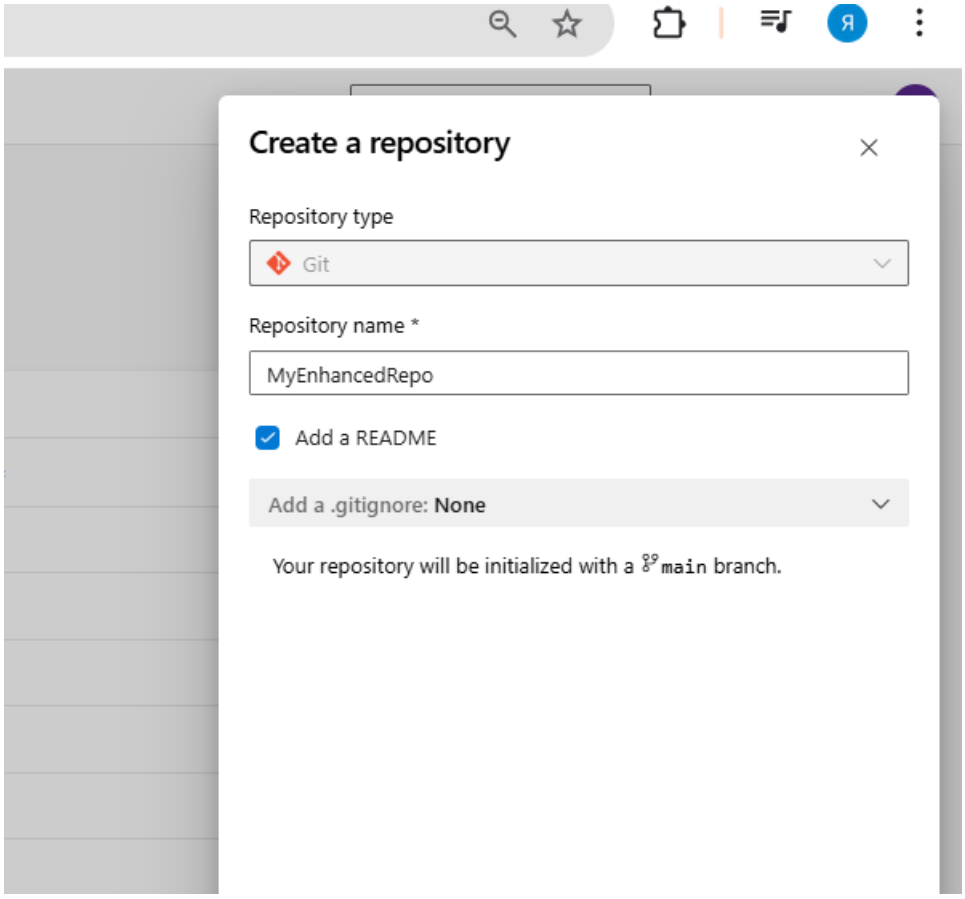
Log in to your Azure DevOps account and navigate to your project.

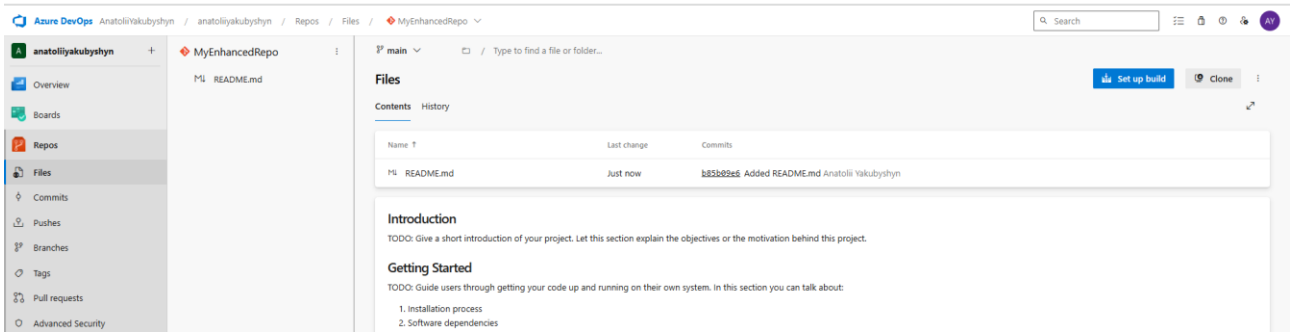


Create a new Git repository named *MyEnhancedRepo*.

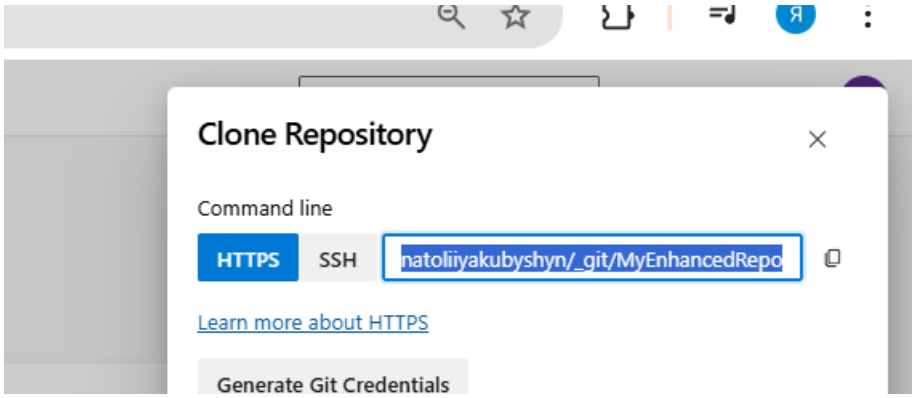


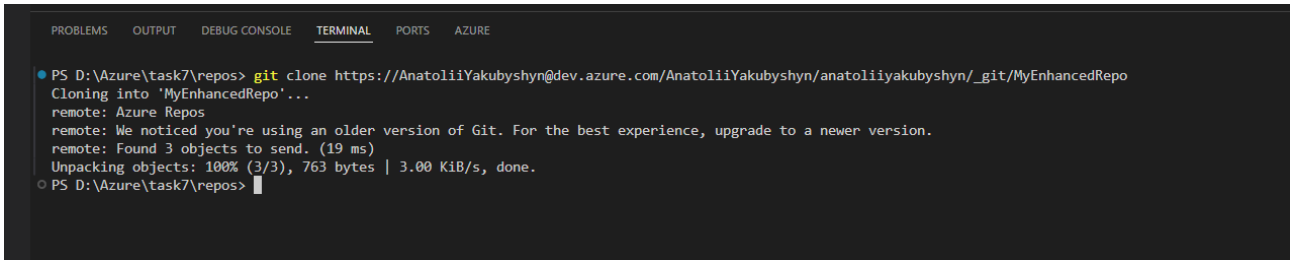
(Previous repos from previous set of assignments)

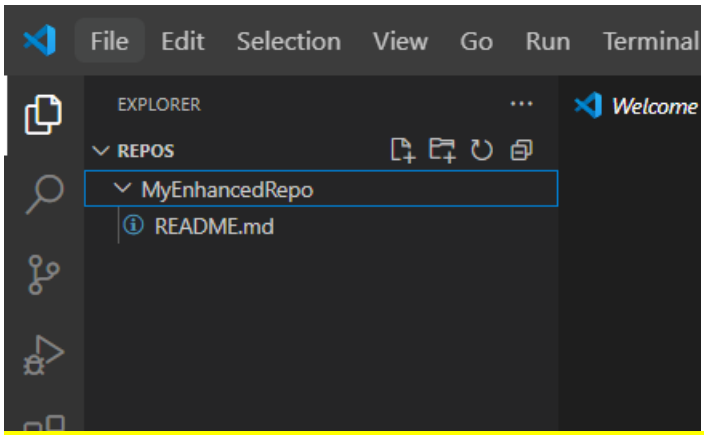




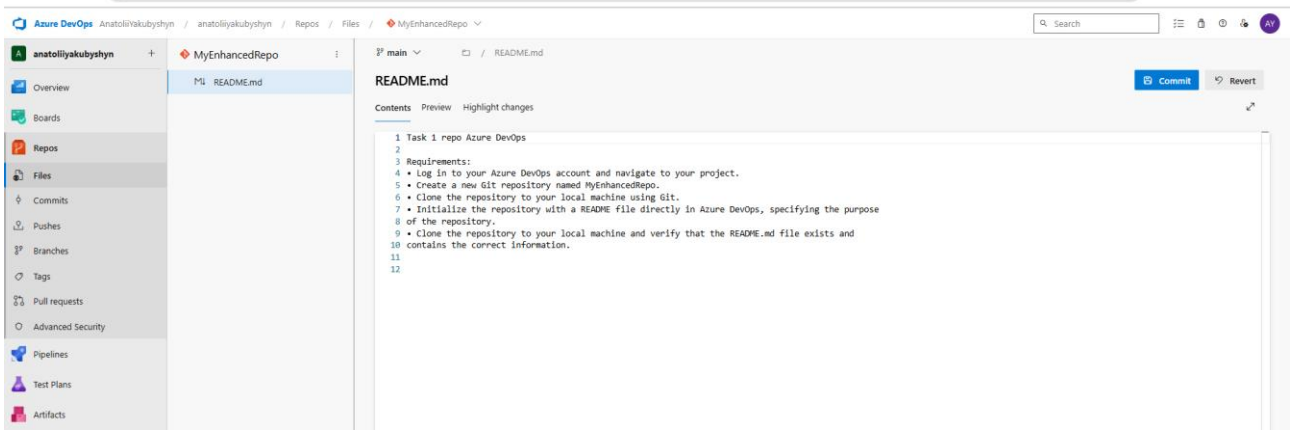
Clone the repository to your local machine using Git.

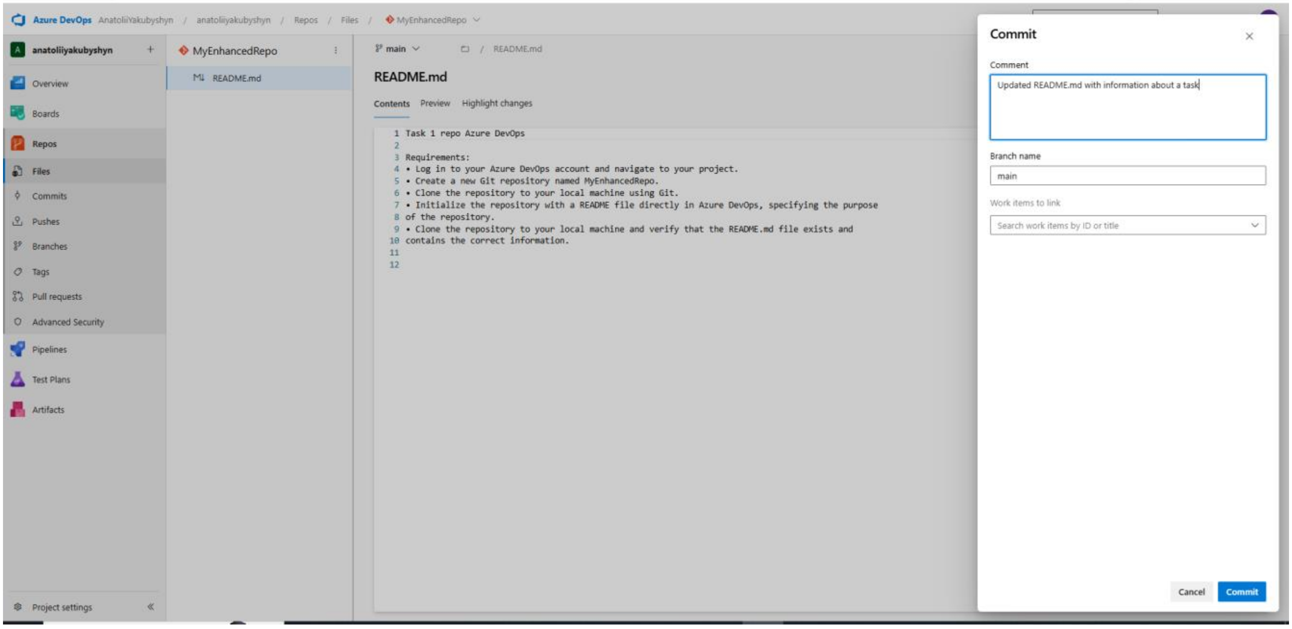




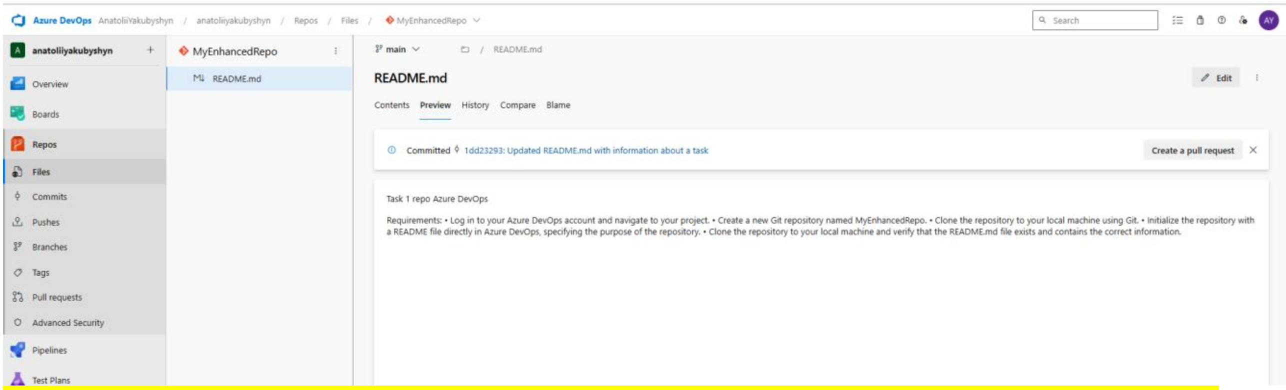


Initialize the repository with a README file directly in Azure DevOps, specifying the purpose of the repository.

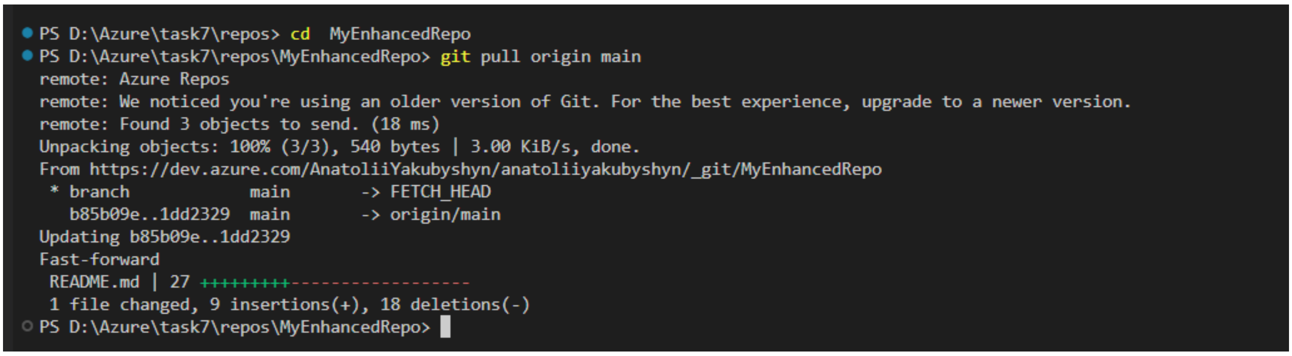


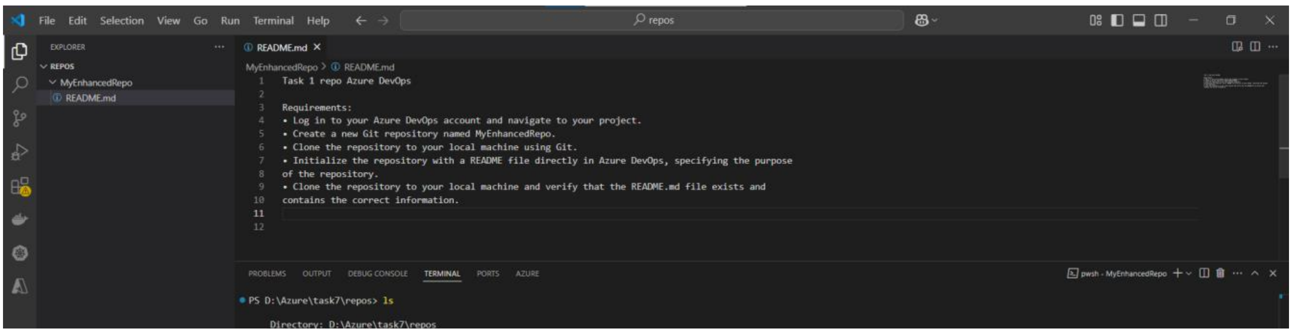


click commit



Clone the repository to your local machine and verify that the README.md file exists and contains the correct information.





Practical Task 2: Implement Branching and Merging

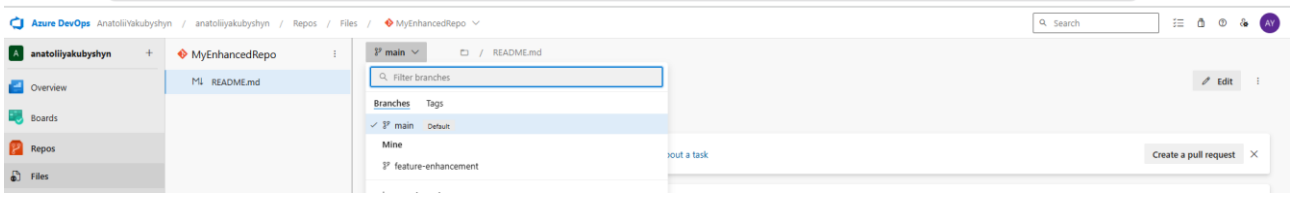
• Create a new branch named feature-enhancement from the main branch in your local repository

```
 1 file changed, 9 insertions(+), 18 deletions(-)
PS D:\Azure\task7\repos\MyEnhancedRepo> git checkout -b feature-enhancement
Switched to a new branch 'feature-enhancement'
PS D:\Azure\task7\repos\MyEnhancedRepo>
```

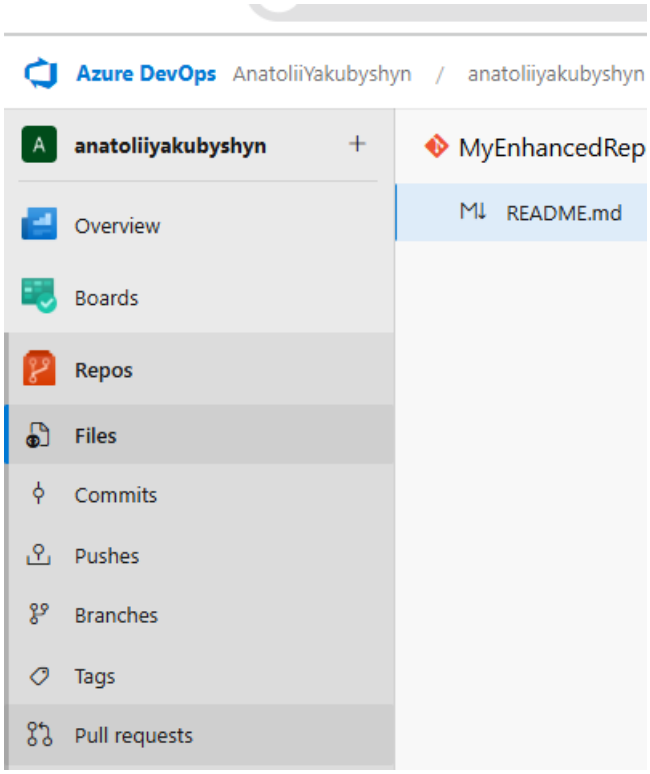
Add a new file named FEATURES.md to the feature-enhancement branch, detailing potential features for the project.

Commit the changes and push the feature-enhancement branch to the remote repository in Azure Repos

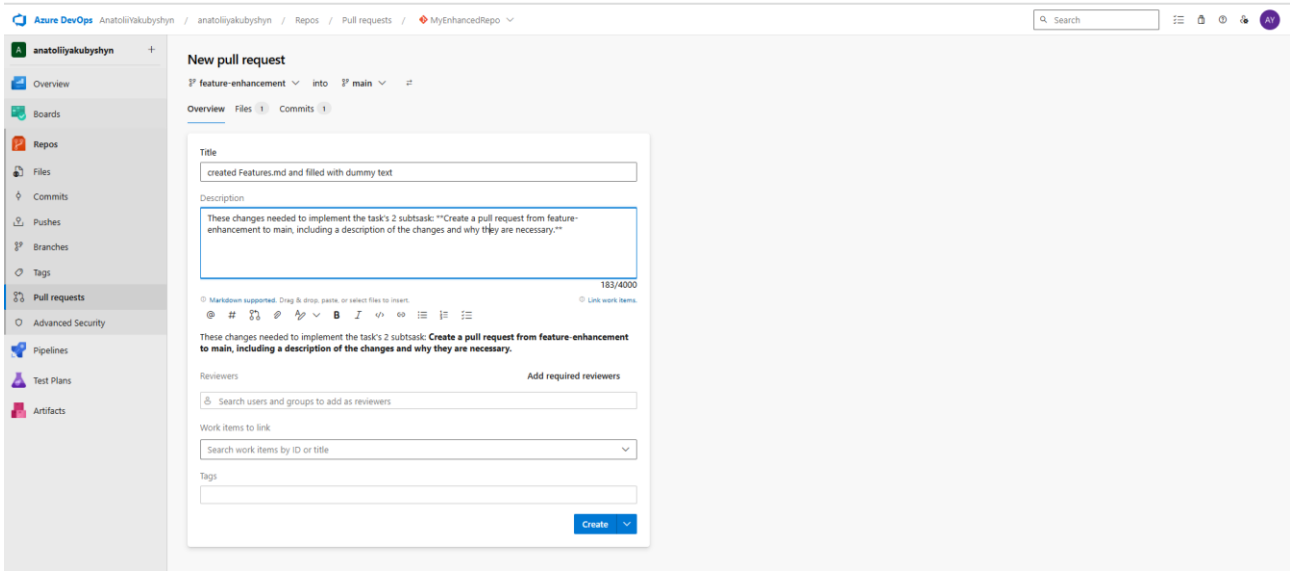
```
 1 file changed, 9 insertions(+), 18 deletions(-)
PS D:\Azure\task7\repos\MyEnhancedRepo> git checkout -b feature-enhancement
Switched to a new branch 'feature-enhancement'
PS D:\Azure\task7\repos\MyEnhancedRepo> git add -A
PS D:\Azure\task7\repos\MyEnhancedRepo> git status
On branch feature-enhancement
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   Features.md

PS D:\Azure\task7\repos\MyEnhancedRepo> git commit -m "created Features.md and filled with dummy text"
[feature-enhancement f38f6e0] created Features.md and filled with dummy text
 1 file changed, 3 insertions(+)
 create mode 100644 Features.md
PS D:\Azure\task7\repos\MyEnhancedRepo> git branch
* feature-enhancement
  main
PS D:\Azure\task7\repos\MyEnhancedRepo> git push origin feature-enhancement
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 579 bytes | 579.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote: Analyzing objects... (3/3) (4 ms)
remote: Validating commits... (1/1) done (0 ms)
remote: Storing packfile... done (43 ms)
remote: Storing index... done (23 ms)
remote: We noticed you're using an older version of Git. For the best experience, upgrade to a newer version.
To https://dev.azure.com/AnatoliiYakubyshyn/anatoliiyakubyshyn/_git/MyEnhancedRepo
 * [new branch]      feature-enhancement -> feature-enhancement
PS D:\Azure\task7\repos\MyEnhancedRepo>
```



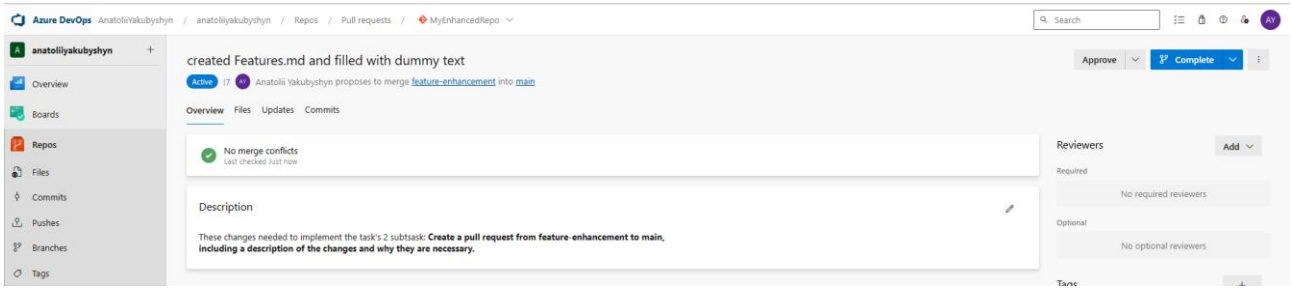
• Create a pull request from feature-enhancement to main, including a description of the changes and why they are necessary.



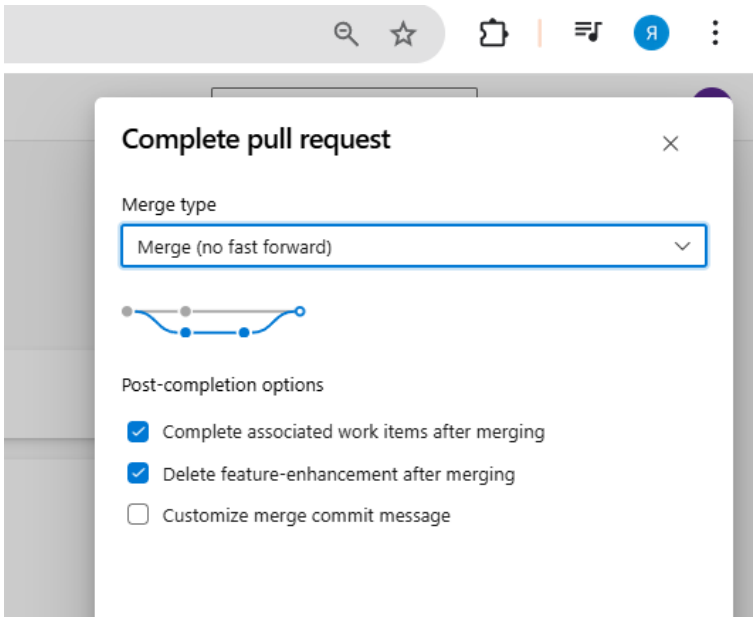
Click Pull requests

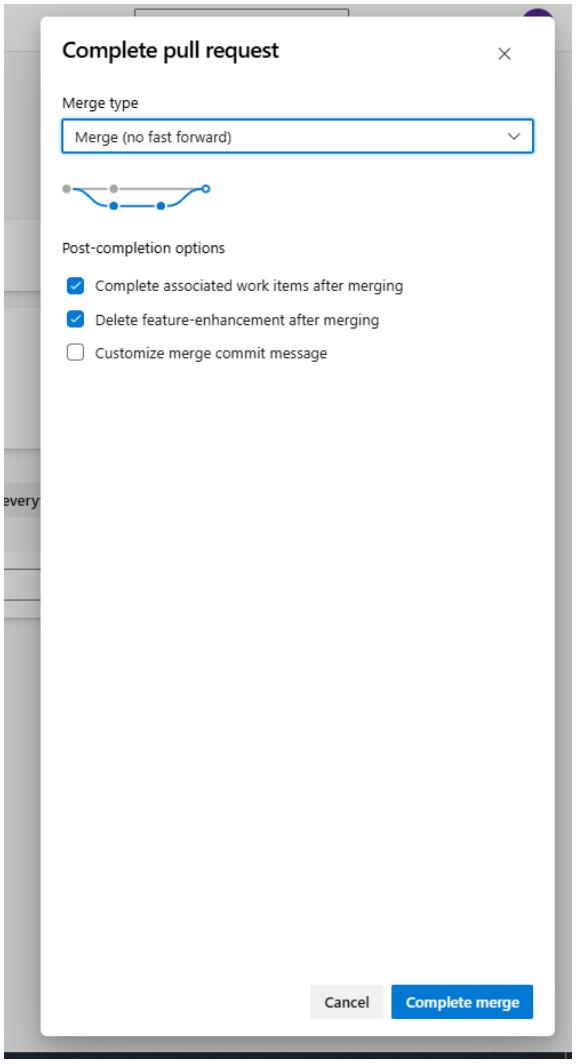


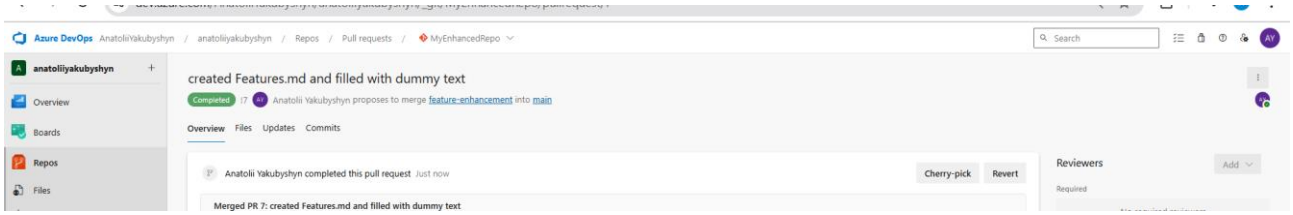
Click create



Review the pull request and merge it into the main branch after resolving any conflicts that may arise.



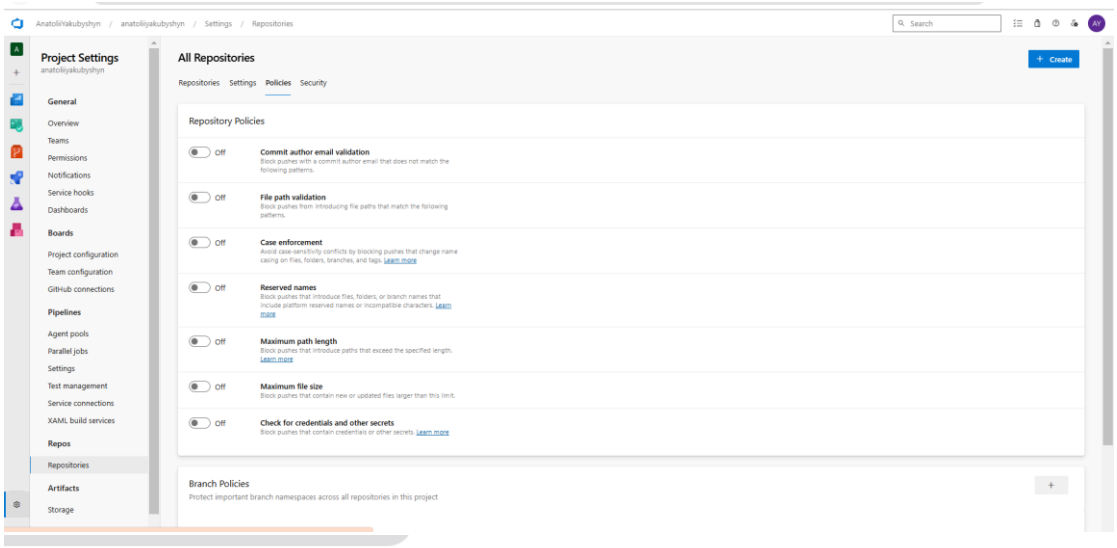

Complete pull request
Merge type
Merge (no fast forward)
Post-completion options
Complete associated work items after merging
Delete feature-enhancement after merging
Customize merge commit message
Cancel
Complete merge

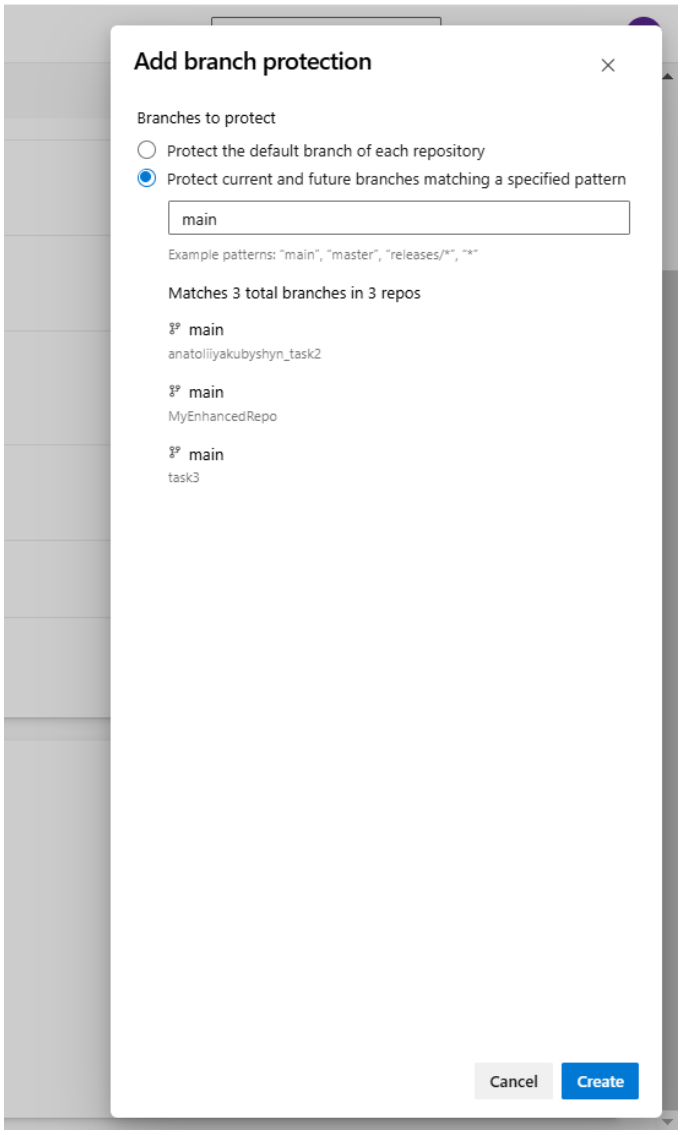

Azure DevOps
anatoliiyakubyshyn
Repos
Pull requests
MyEnhancedRepo
Search
Overview
Boards
Repos
Files
created Features.md and filled with dummy text
Completed
Anatolii Yakubyshyn proposes to merge feature-enhancement into main
Overview
Files
Updates
Commits
Anatolii Yakubyshyn completed this pull request Just now
Cherry-pick
Revert
Reviewers
Add
Required
Merged PR 7: created Features.md and filled with dummy text

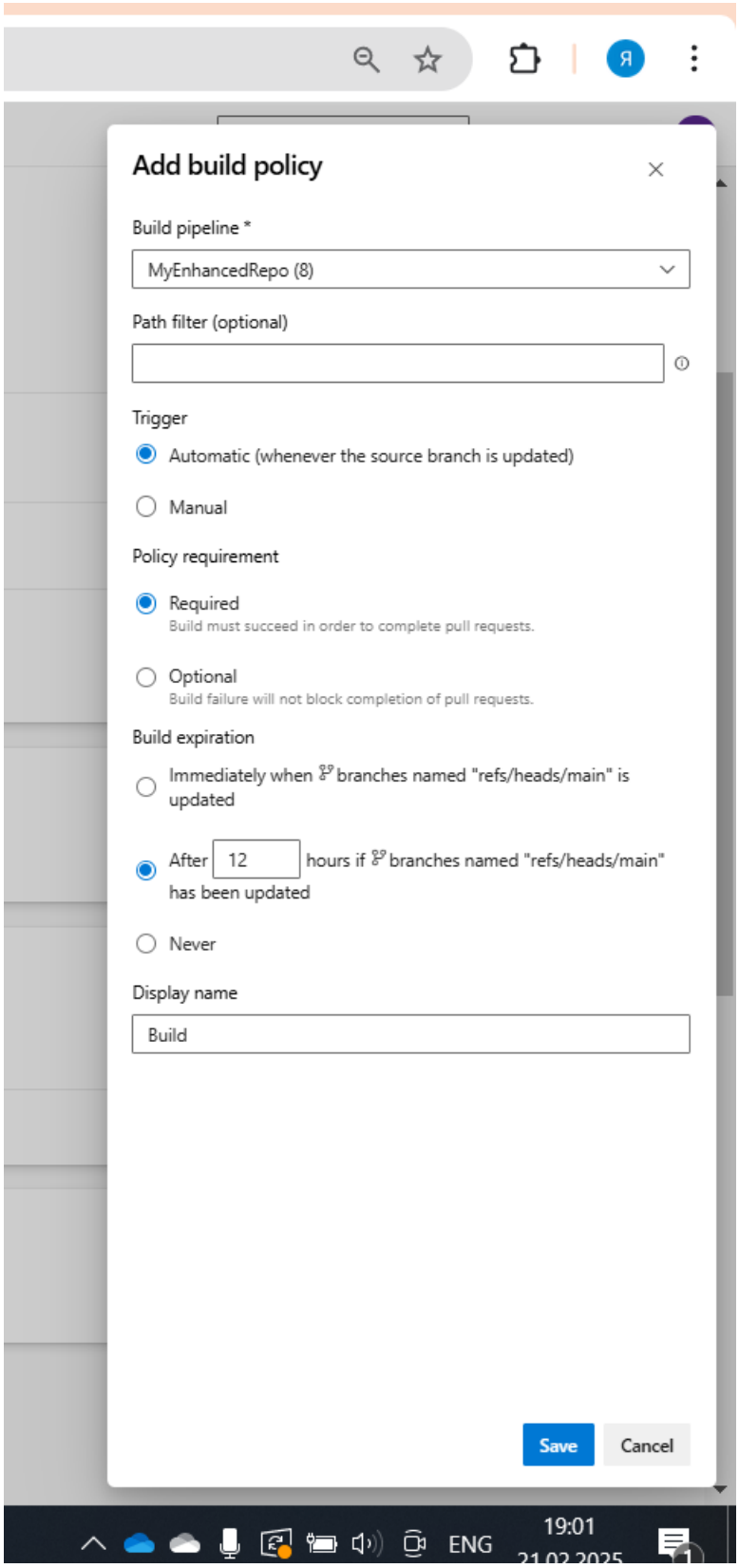

Azure DevOps
anatoliiyakubyshyn
Repos
Files
MyEnhancedRepo
Search
Overview
Boards
Repos
Files
MyEnhancedRepo
Features.md
README.md
main
Features.md
Features.md
Edit
Contents
Preview
History
Compare
Blame
1 Potential feature:
2 Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercita
3
4

Practical Task 3: Set Up Branch Policies and Code Review

Configure branch policies for the main branch in Azure Repos to require at least one reviewer for pull requests and to enforce a minimum number of successful builds (you can simulate this by using a mock build status if needed).



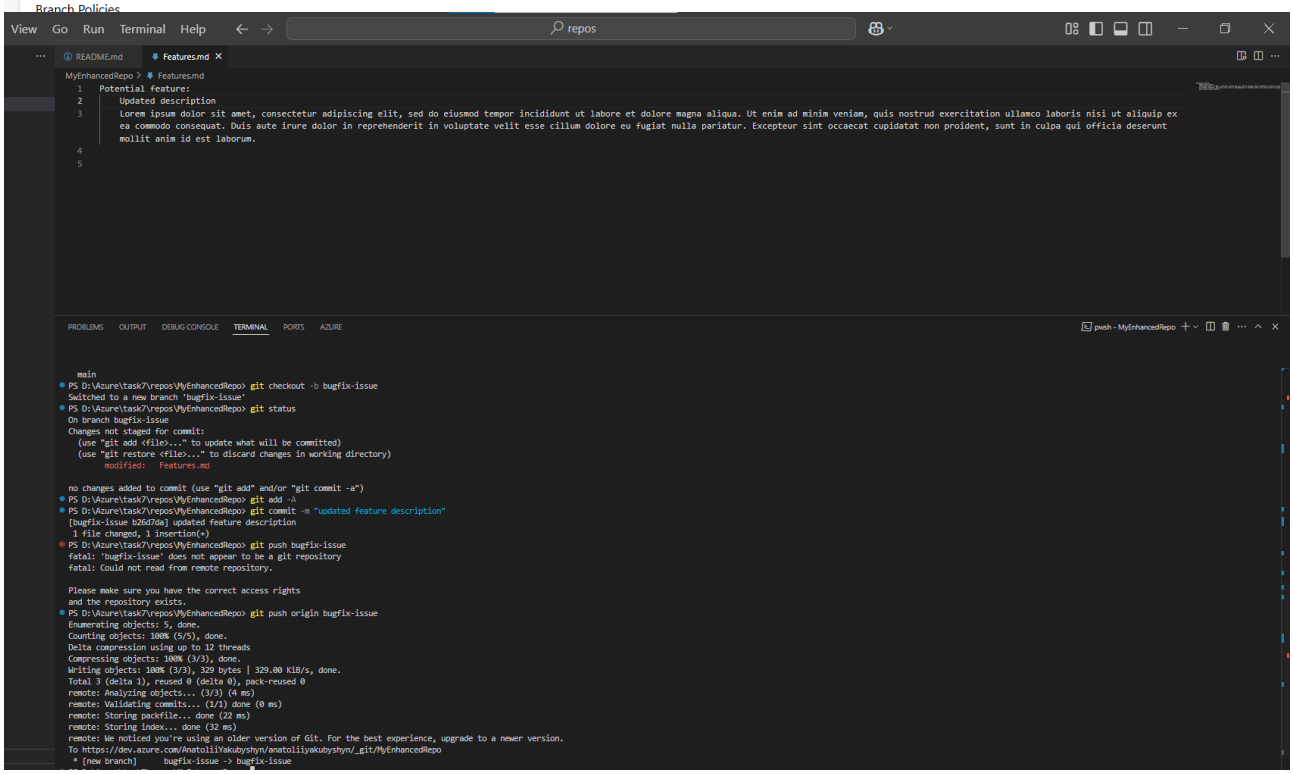


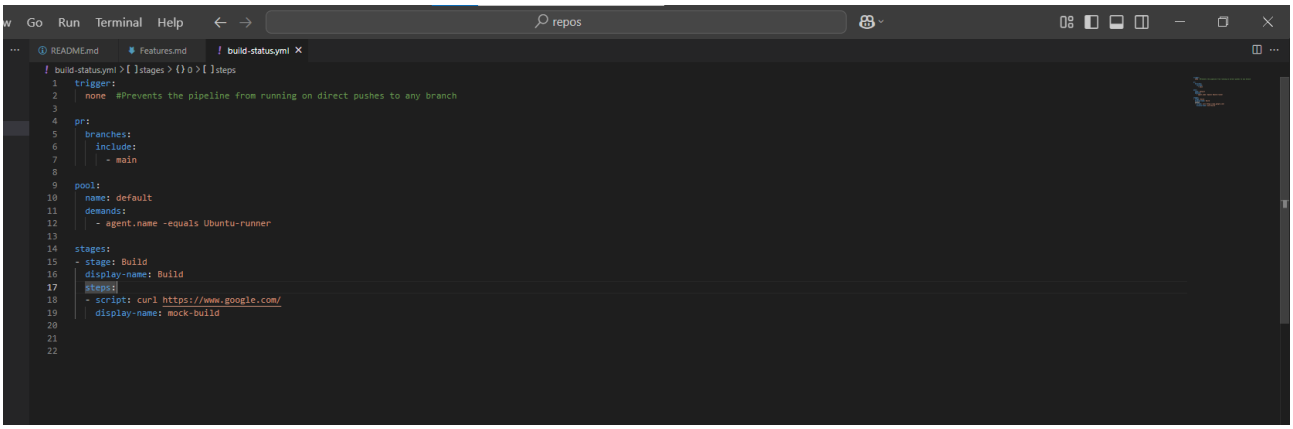

Add build policy
Build pipeline *
MyEnhancedRepo (8)
Path filter (optional)
Trigger
Automatic (whenever the source branch is updated)
Manual
Policy requirement
Required
Build must succeed in order to complete pull requests.
Optional
Build failure will not block completion of pull requests.
Build expiration
Immediately when branches named "refs/heads/main" is updated
After 12 hours if branches named "refs/heads/main" has been updated
Never
Display name
Build
Save
Cancel
ENG
19:01

• Create a new branch named bugfix-issue from the main branch and make changes to the FEATURES.md file to address a hypothetical bug (e.g., update a feature description). • Commit the changes and push the bugfix-issue branch to the remote repository.

```
PS D:\Azure\task7\repos\MyEnhancedRepo> git checkout -b bugfix-issue
Switched to a new branch 'bugfix-issue'
```

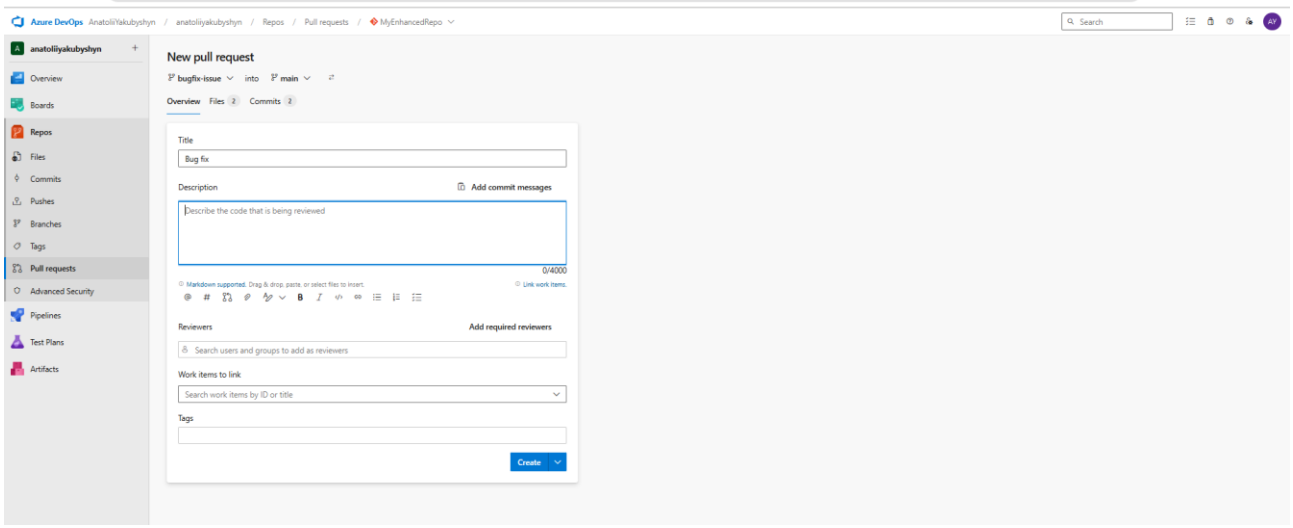
Branch Policies

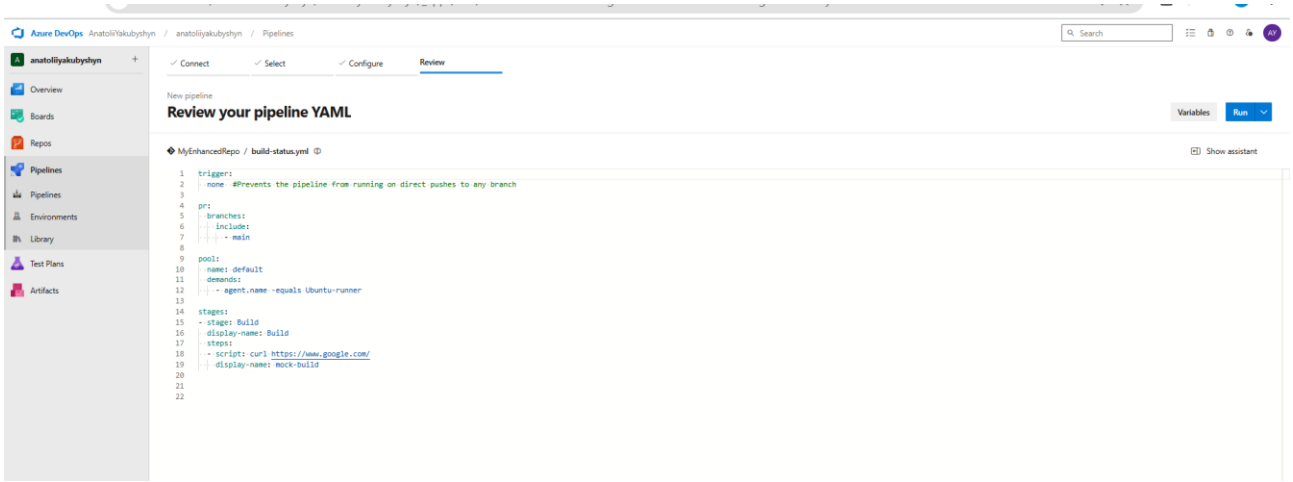


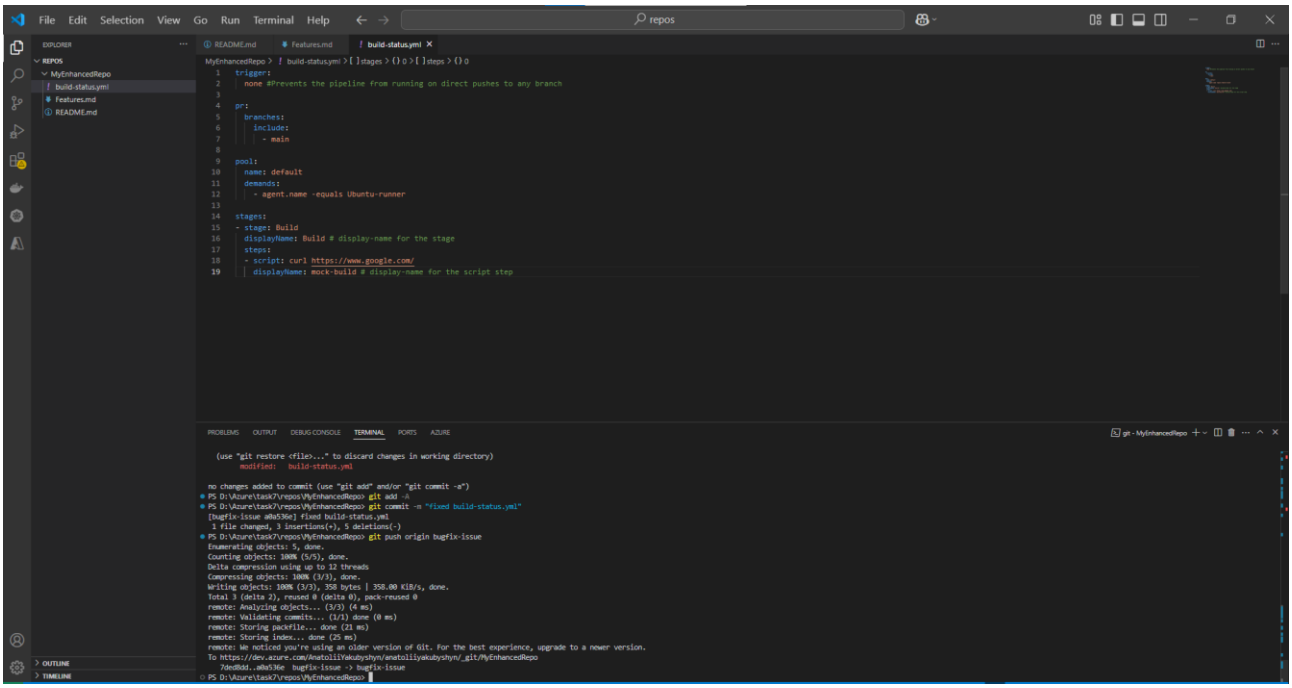


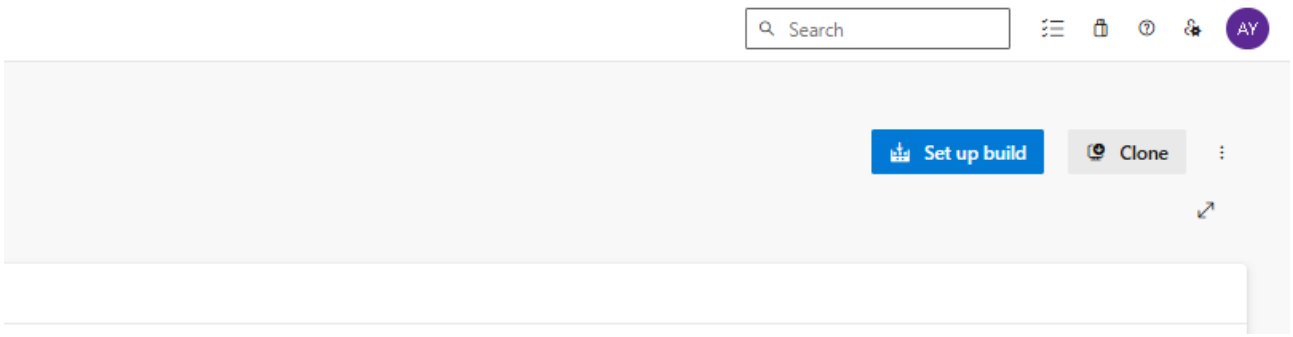
created mock PR build

Create a pull request from bugfix-issue to main, ensuring that the pull request meets the branch policies set earlier

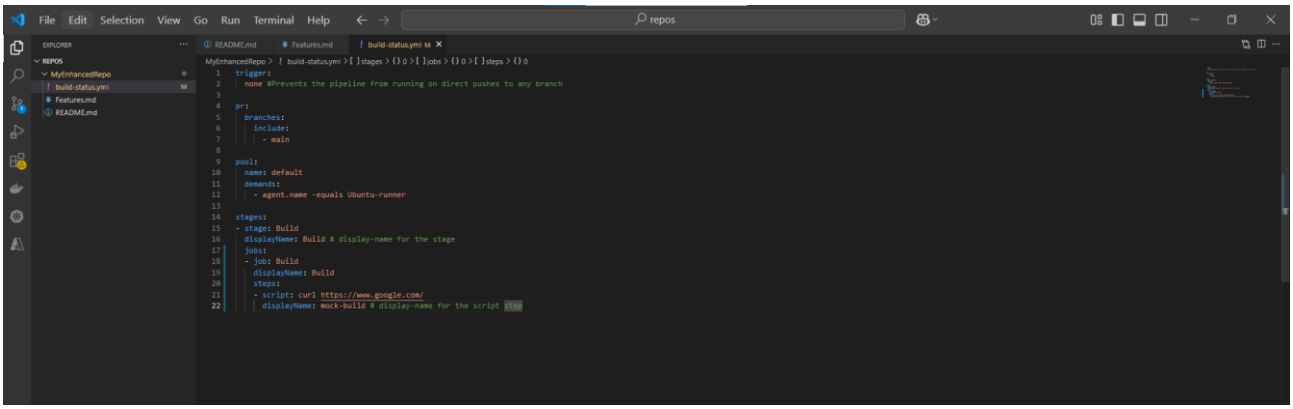


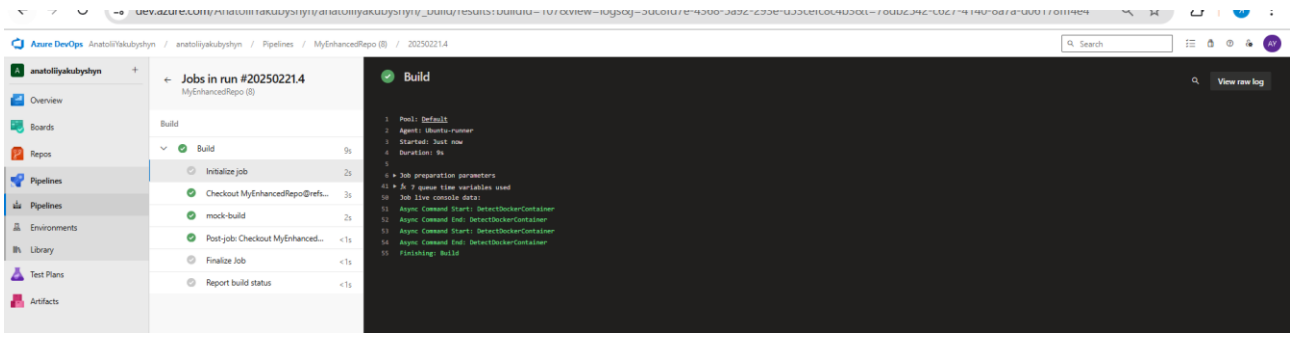


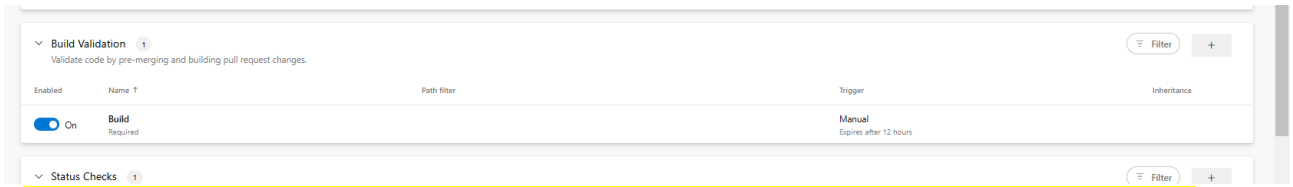



Search
AY
Set up build
Clone

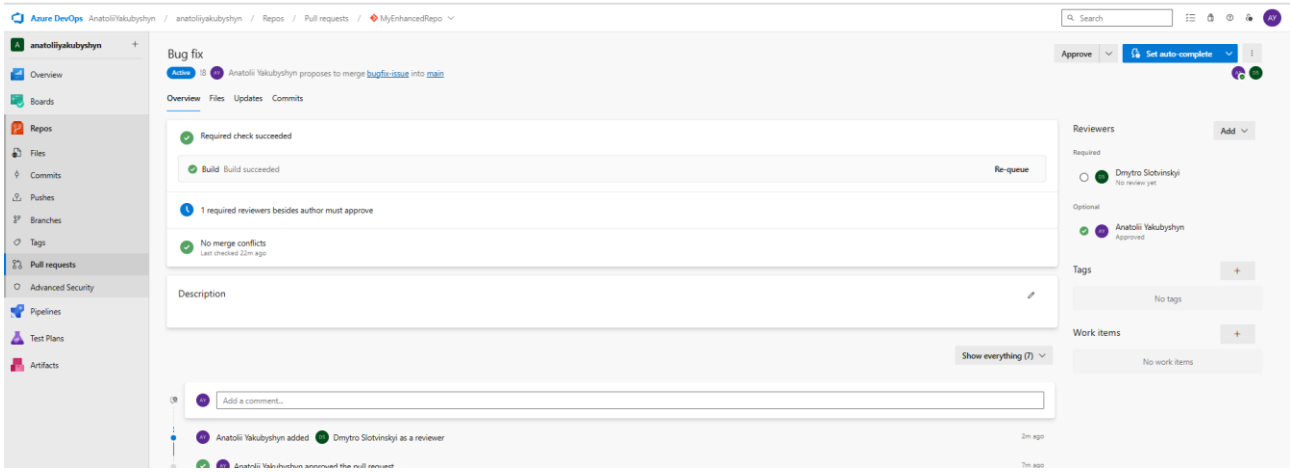

Azure DevOps
Pipelines
Overview
Boards
Repos
Pipelines
Environments
Library
Test Plans
Connect
Select
Configure
Review
New pipeline
Configure your pipeline
Starter pipeline
Existing Azure Pipelines YAML file
Show more
Select an existing YAML file
Branch
bugfix-issue
Path
/build-status.yml


File Edit Selection View Go Run Terminal Help
repos
EXPLORER
REPOS
MyEnhancedRepo
build-status.yml
Features.md
README.md
trigger:
none #Prevents the pipeline from running on direct pushes to any branch
pr:
branches:
include:
- main
pool:
name: default
demands:
- agent.name -equals Ubuntu-runner
stages:
- stage: Build
displayName: Build # display-name for the stage
jobs:
- job: Build
displayName: Build
steps:
- script: curl https://www.google.com/
displayName: mock-build # display-name for the script step



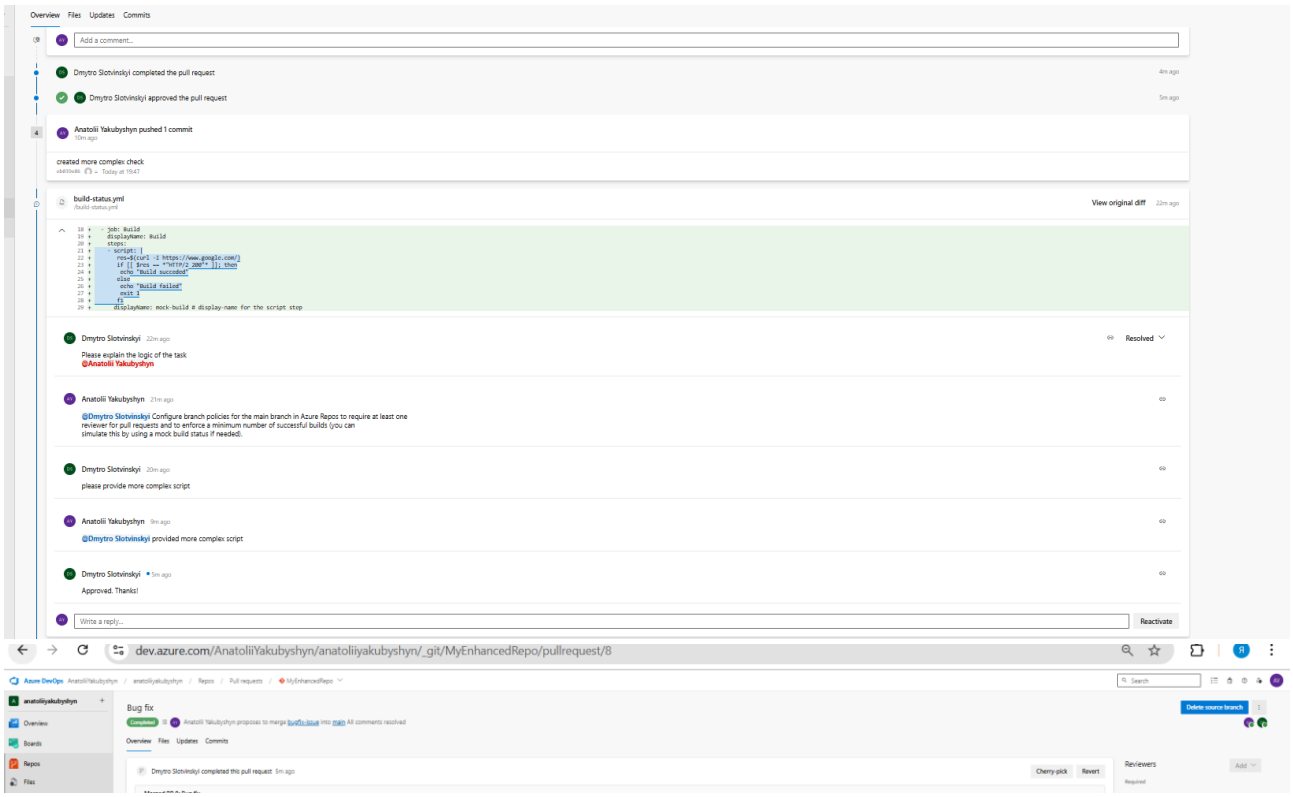


• Invite at least one team member or colleague to review the pull request and provide feedback. • Address any comments or requested changes from the reviewer and update the pull request accordingly. • Merge the pull request into the main branch once it has been approved.



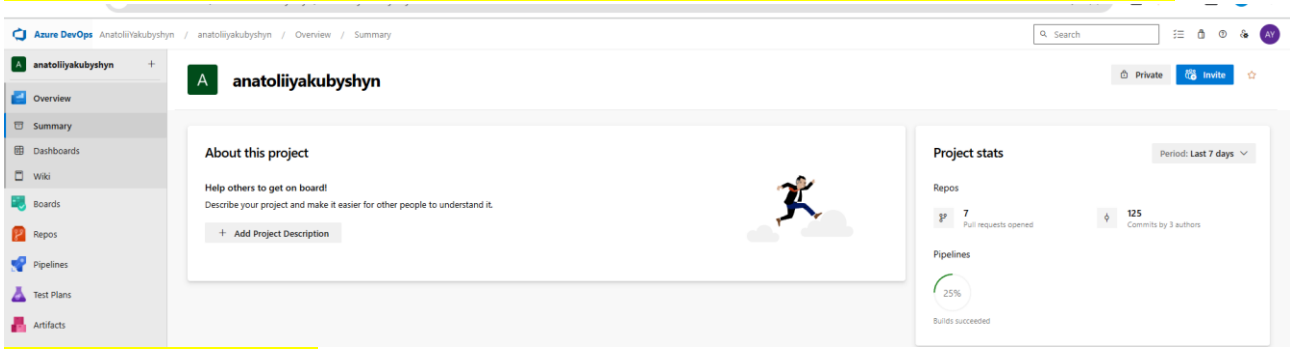
Invite at least one team member or colleague to review the pull request and provide feedback.

• Address any comments or requested changes from the reviewer and update the pull request accordingly.
• Merge the pull request into the main branch once it has been approved.



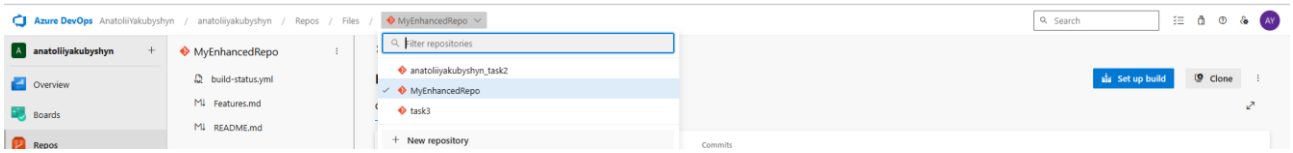
Practical Task 4: Create a Simple Build Pipeline in Azure DevOps

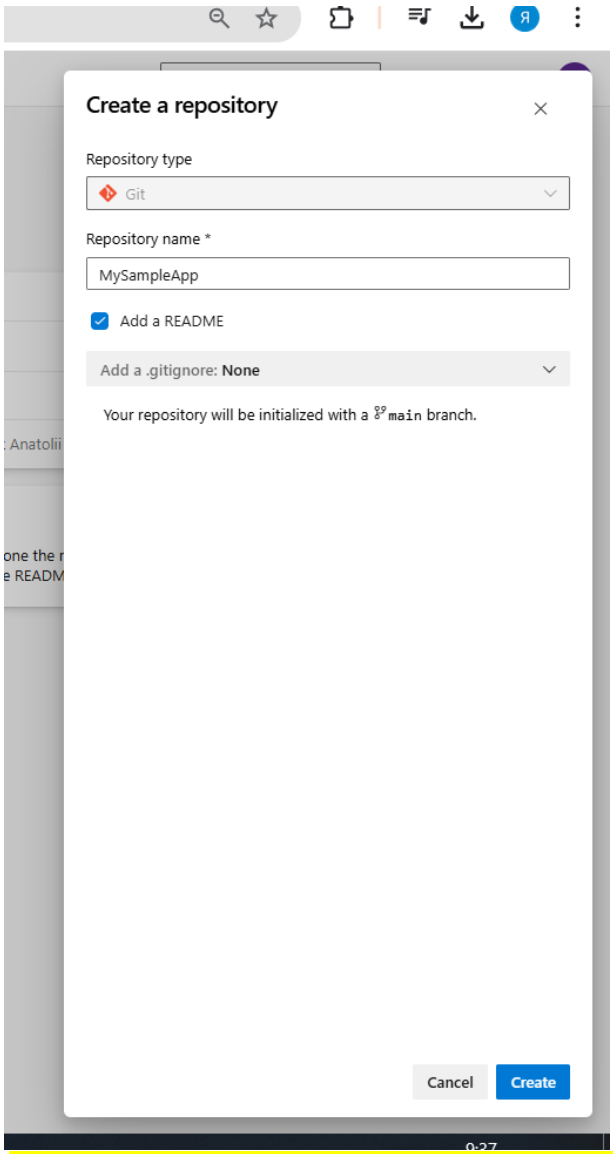
1. Log in to Azure DevOps: a. Access your Azure DevOps organization and navigate to your project.



2. Create a Git Repository:

a. In Azure Repos, create a new Git repository named MySampleApp.

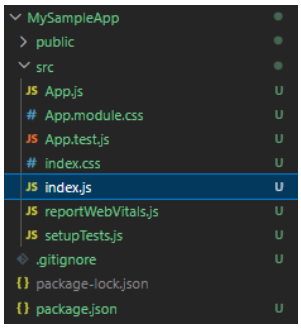


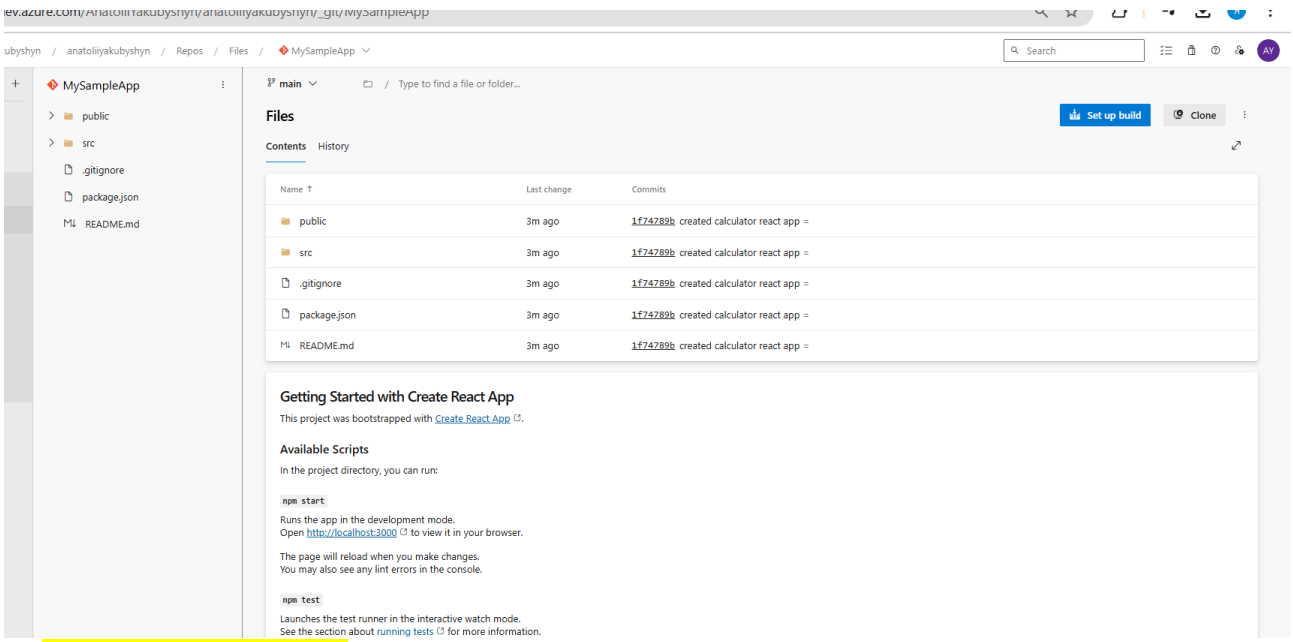


Clone the repository to your local machine and add a simple application (e.g., a basic .NET Core or Node.js application).

```
PS D:\Azure\task7\repos> git clone https://AnatoliiYakubyshyn@dev.azure.com/AnatoliiYakubyshyn/anatoliiyakubyshyn/_git/MySampleApp
Cloning into 'MySampleApp'...
remote: Azure Repos
remote: We noticed you're using an older version of Git. For the best experience, upgrade to a newer version.
remote: Found 3 objects to send. (24 ms)
Unpacking objects: 100% (3/3), 763 bytes | 3.00 KiB/s, done.
PS D:\Azure\task7\repos>
```

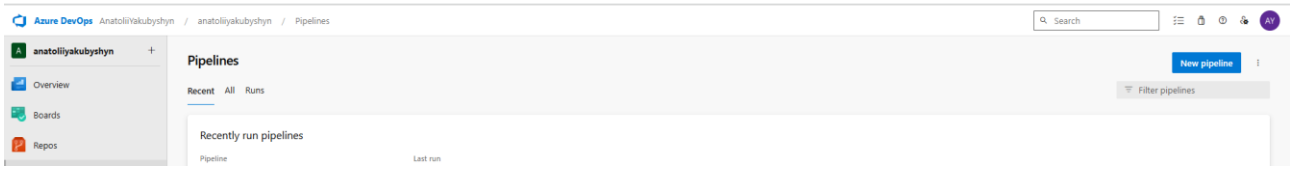
Node.js app (React) – Calculator





3. Create a Build Pipeline:

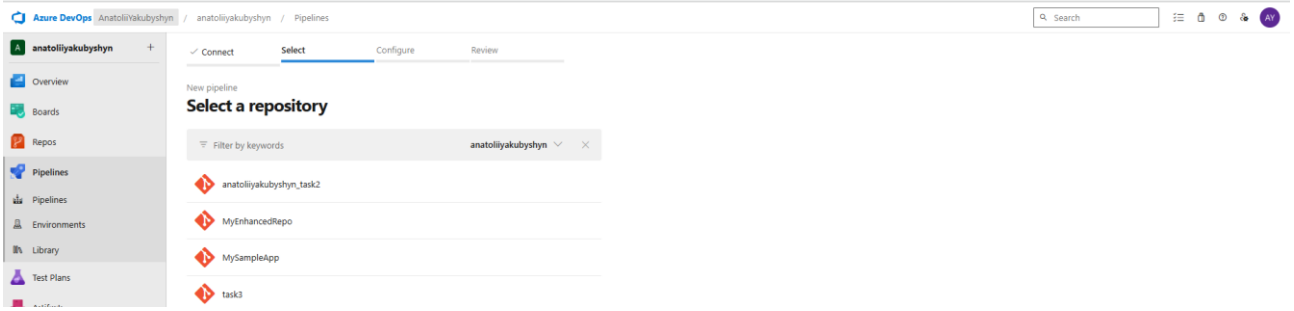
a. Navigate to Azure Pipelines and create a new pipeline.



Click new pipeline

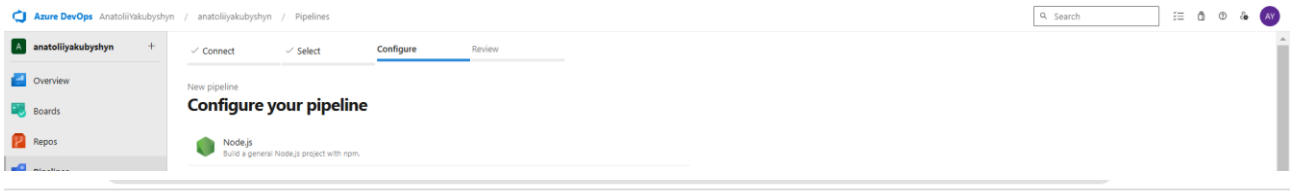
Azure repos git

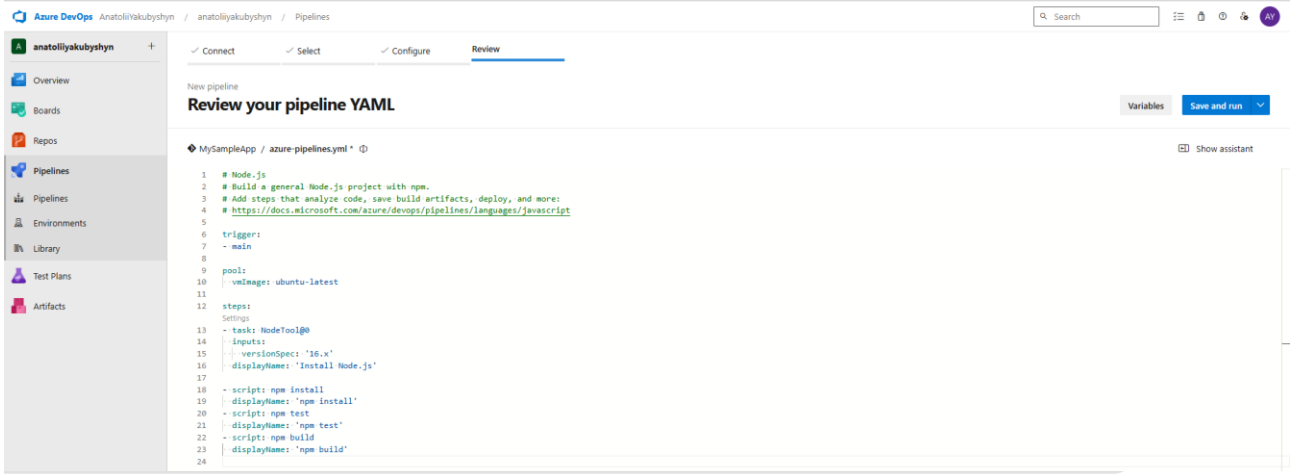
b. Select the Azure Repos Git repository as the source.

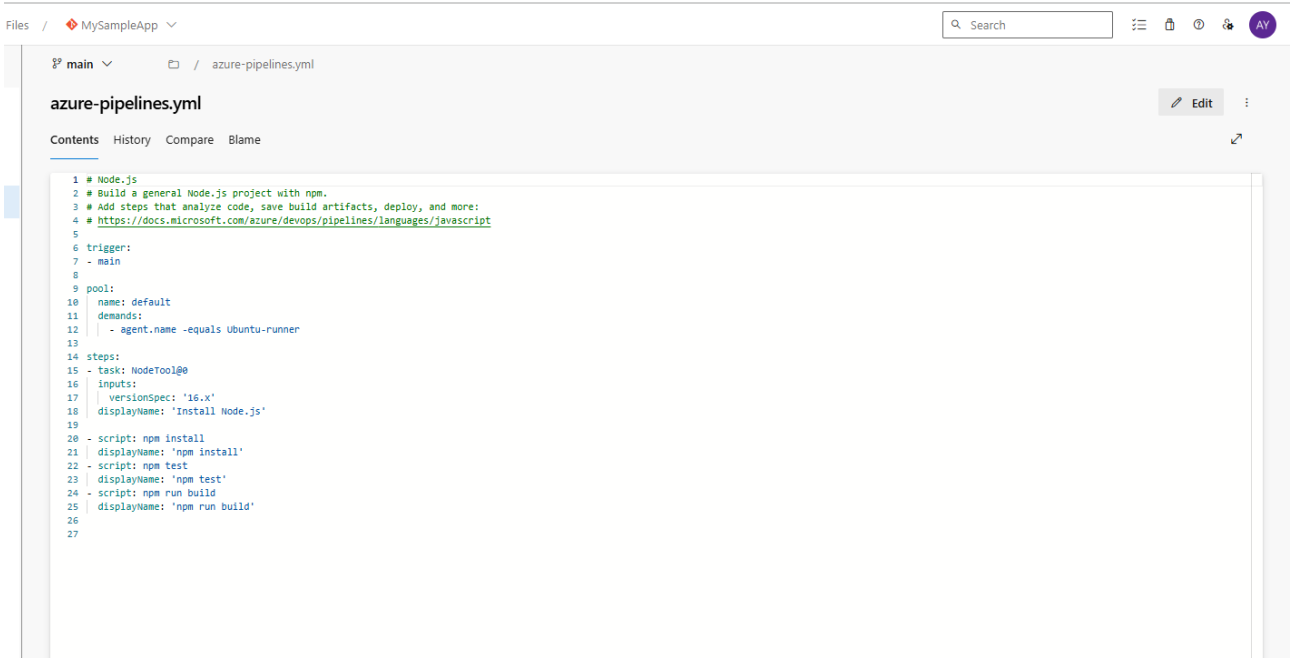


MySampleApp

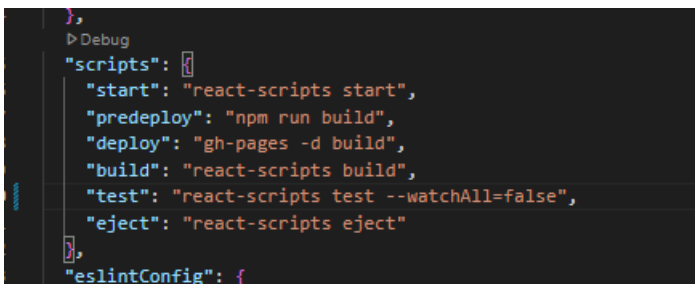
c. Use the YAML pipeline editor to define the pipeline, including the following steps:



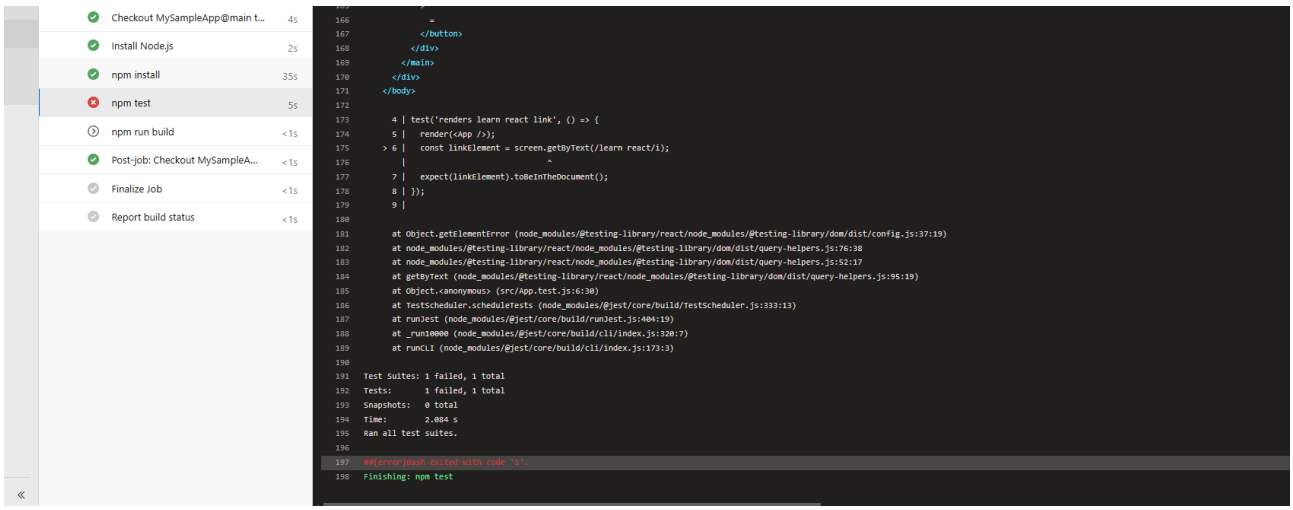




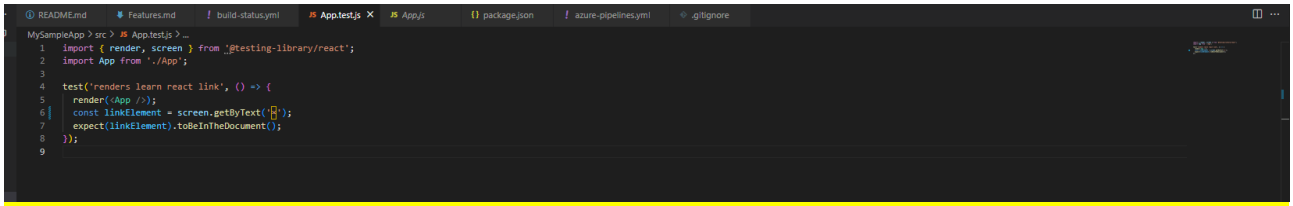
(fixed build script)



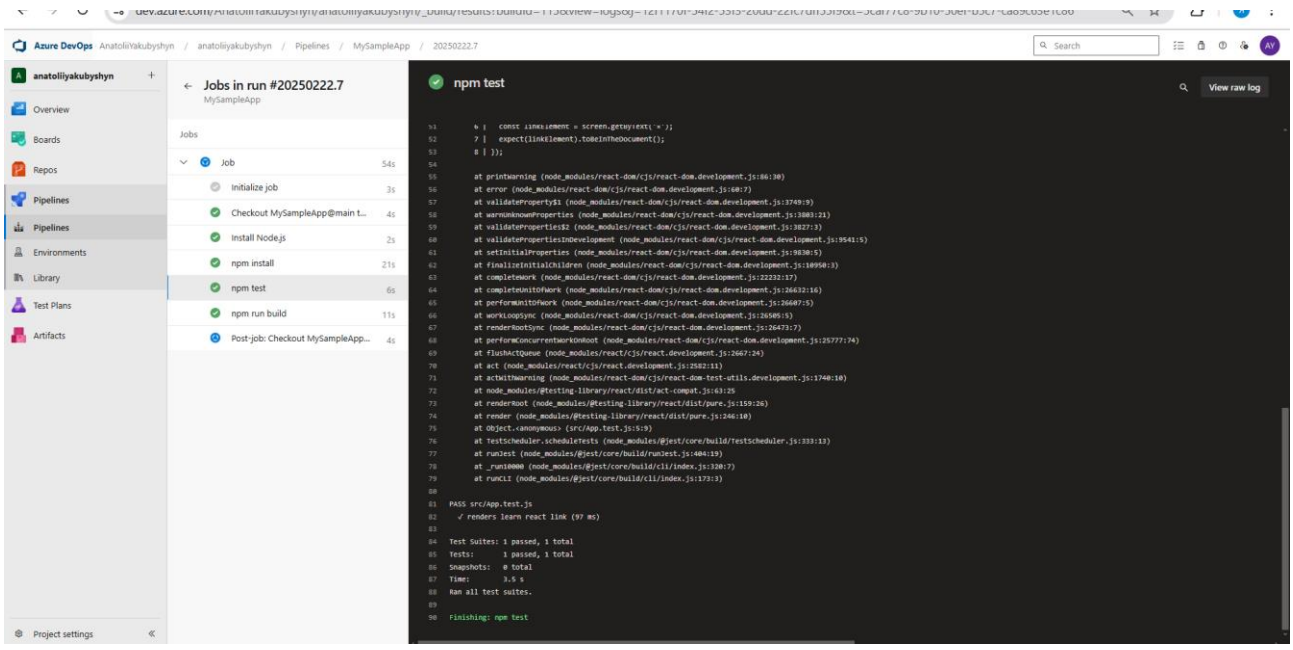
changed test

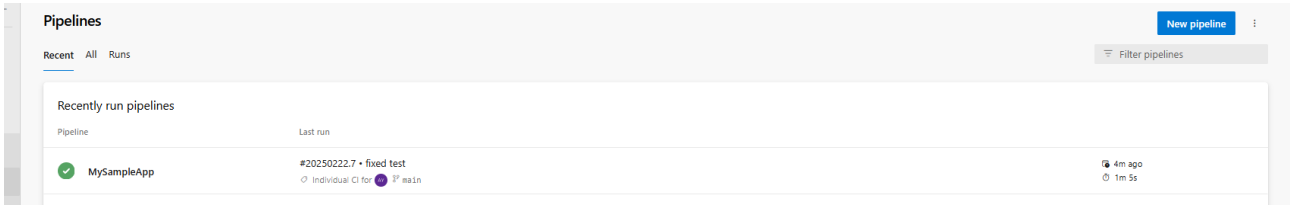


Fix test->



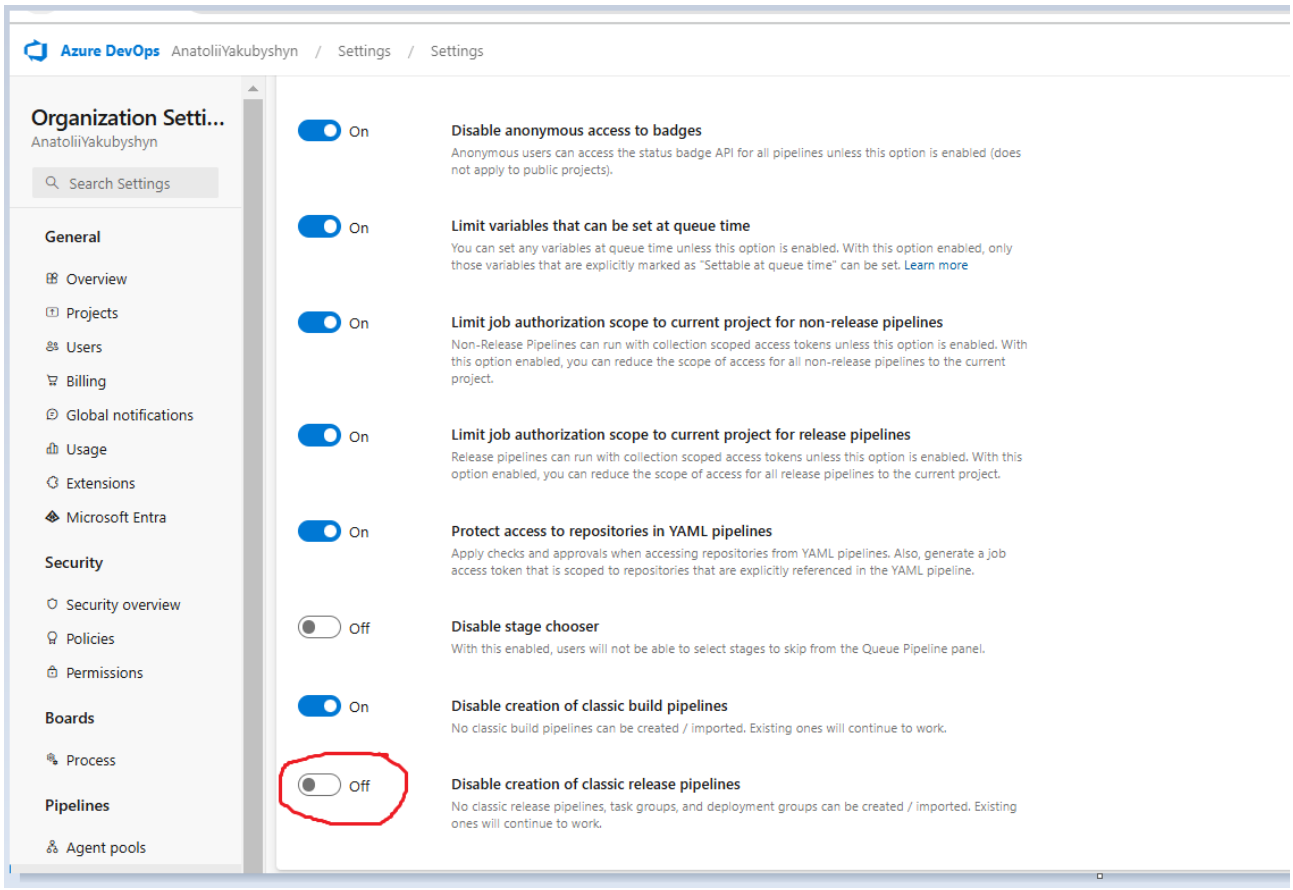
4. Run the Pipeline: a. Save and run the pipeline. b. Verify that the build pipeline completes successfully and check the build logs for any errors.





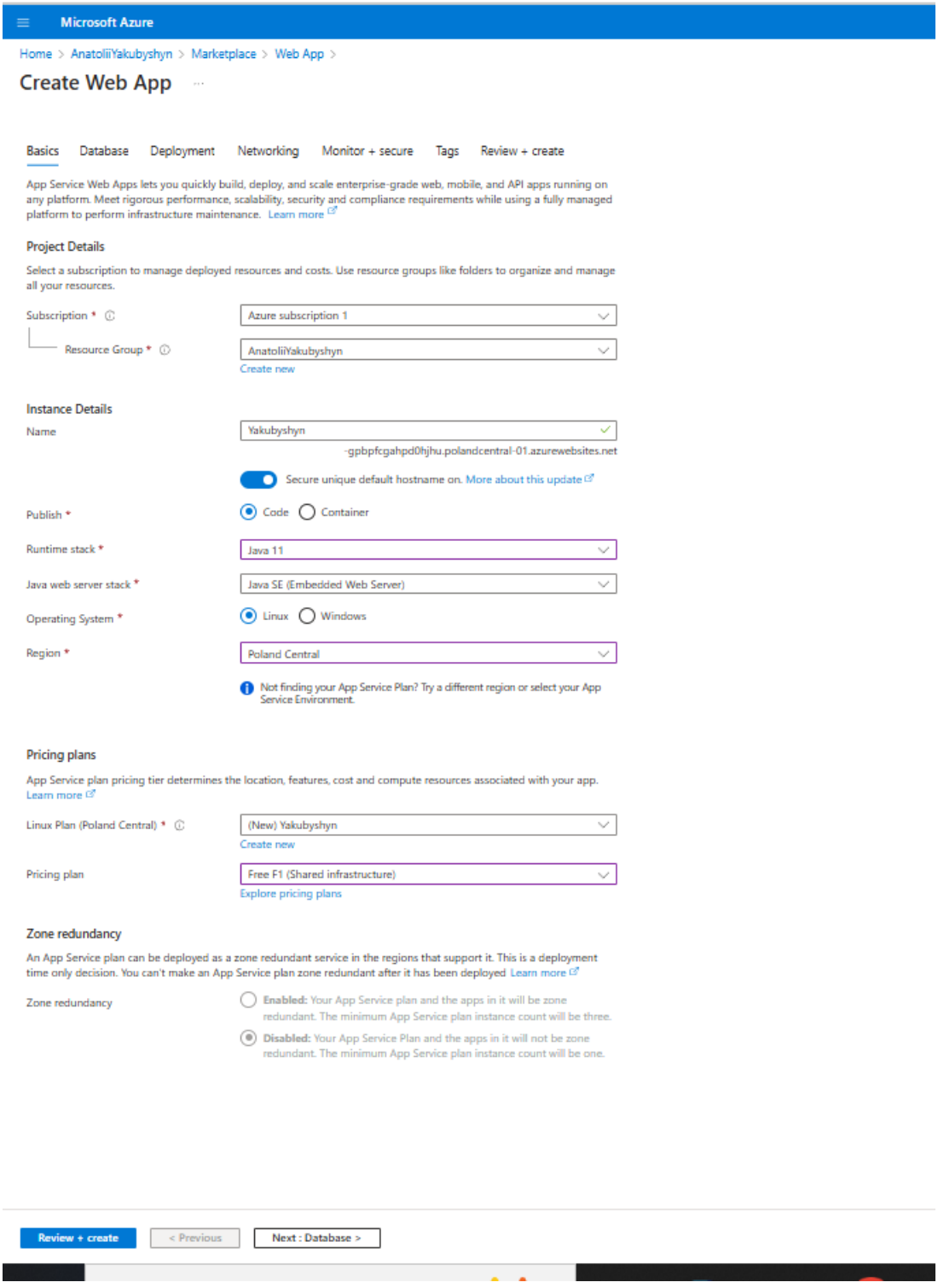
Practical Task 5: Set Up Continuous Deployment (CD) to Azure Web App

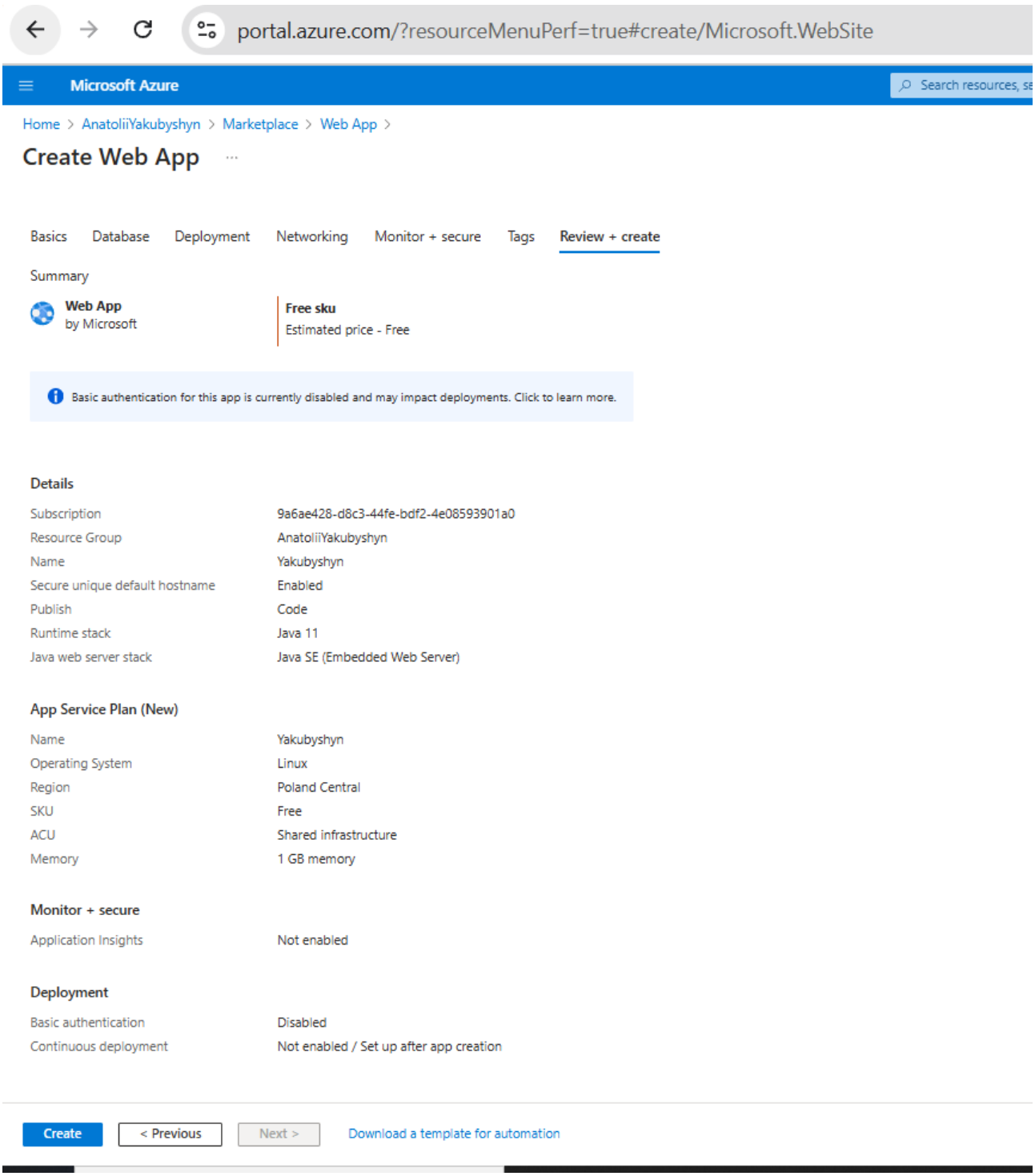
Presetup:



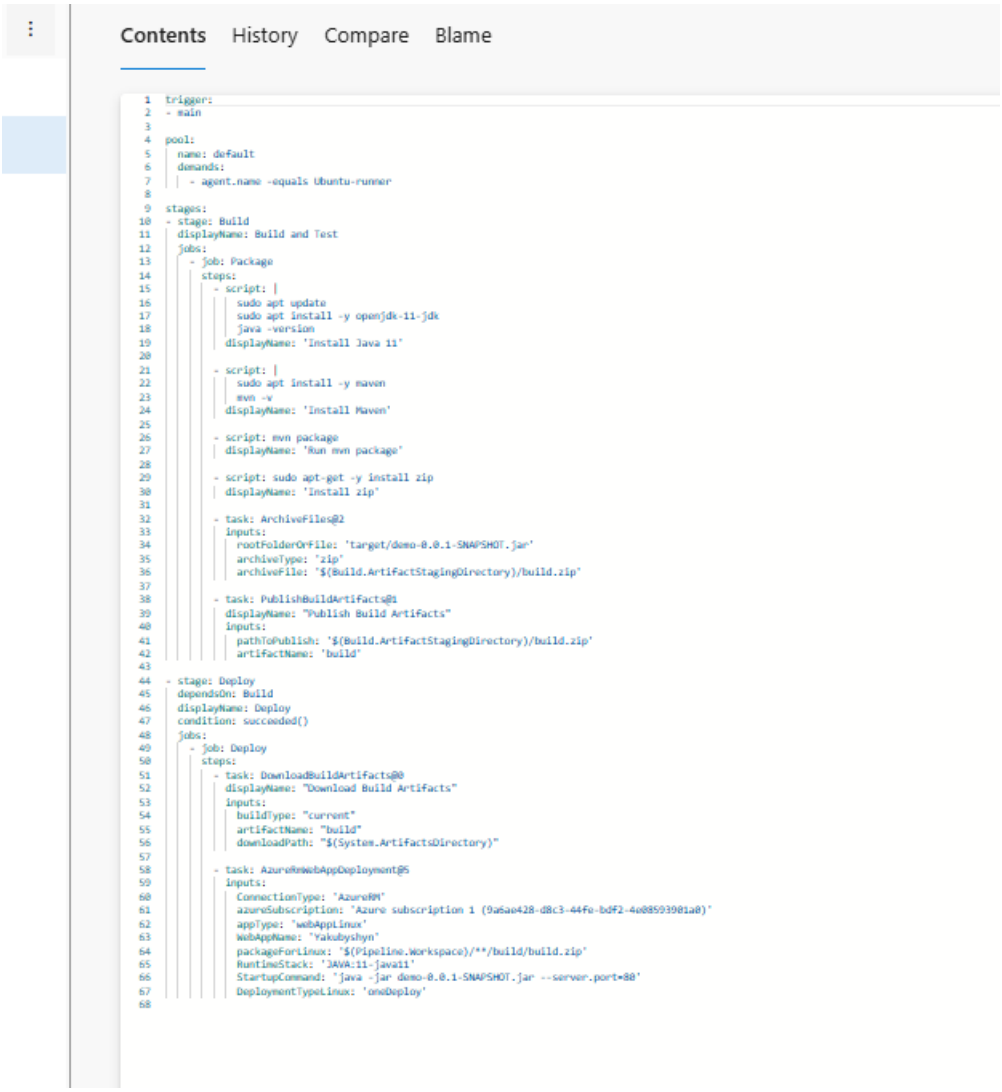
I decided to create a separate repo with java jar. Because my first app produces static files but I need package.

2. Create an Azure Web App: a. In the Azure portal, manually create an Azure Web App or use an ARM template to create it.


Microsoft Azure
Home > AnatoliiYakubyshyn > Marketplace > Web App >
Create Web App
Basics Database Deployment Networking Monitor + secure Tags Review + create
App Service Web Apps lets you quickly build, deploy, and scale enterprise-grade web, mobile, and API apps running on any platform. Meet rigorous performance, scalability, security and compliance requirements while using a fully managed platform to perform infrastructure maintenance. Learn more
Project Details
Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.
Subscription * Azure subscription 1
Resource Group * AnatoliiYakubyshyn
Create new
Instance Details
Name Yakubyshyn
-gpbpfcgahpd0hjhu.polandcentral-01.azurewebsites.net
Secure unique default hostname on. More about this update
Publish * Code Container
Runtime stack * Java 11
Java web server stack * Java SE (Embedded Web Server)
Operating System * Linux Windows
Region * Poland Central
Not finding your App Service Plan? Try a different region or select your App Service Environment.
Pricing plans
App Service plan pricing tier determines the location, features, cost and compute resources associated with your app.
Learn more
Linux Plan (Poland Central) * (New) Yakubyshyn
Create new
Pricing plan Free F1 (Shared infrastructure)
Explore pricing plans
Zone redundancy
An App Service plan can be deployed as a zone redundant service in the regions that support it. This is a deployment time only decision. You can't make an App Service plan zone redundant after it has been deployed Learn more
Zone redundancy
Enabled: Your App Service plan and the apps in it will be zone redundant. The minimum App Service plan instance count will be three.
Disabled: Your App Service Plan and the apps in it will not be zone redundant. The minimum App Service plan instance count will be one.
Review + create
< Previous
Next : Database >


portal.azure.com/?resourceMenuPerf=true#create/Microsoft.WebSite
Microsoft Azure
Search resources, se
Home > AnatoliiYakubyshyn > Marketplace > Web App >
Create Web App
Basics Database Deployment Networking Monitor + secure Tags Review + create
Summary
Web App
by Microsoft
Free sku
Estimated price - Free
Basic authentication for this app is currently disabled and may impact deployments. Click to learn more.
Details
Subscription 9a6ae428-d8c3-44fe-bdf2-4e08593901a0
Resource Group AnatoliiYakubyshyn
Name Yakubyshyn
Secure unique default hostname Enabled
Publish Code
Runtime stack Java 11
Java web server stack Java SE (Embedded Web Server)
App Service Plan (New)
Name Yakubyshyn
Operating System Linux
Region Poland Central
SKU Free
ACU Shared infrastructure
Memory 1 GB memory
Monitor + secure
Application Insights Not enabled
Deployment
Basic authentication Disabled
Continuous deployment Not enabled / Set up after app creation
Create
< Previous
Next >
Download a template for automation

1. Extend the Build Pipeline: a. Open the existing build pipeline created in Task 4. b. Add a new stage for deployment after the build stage.

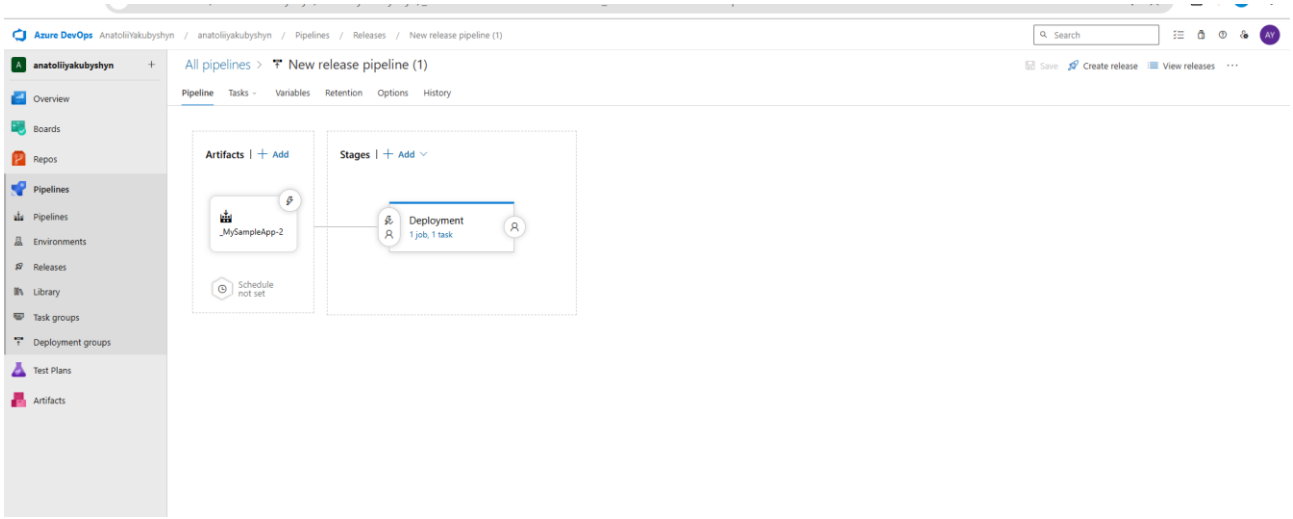


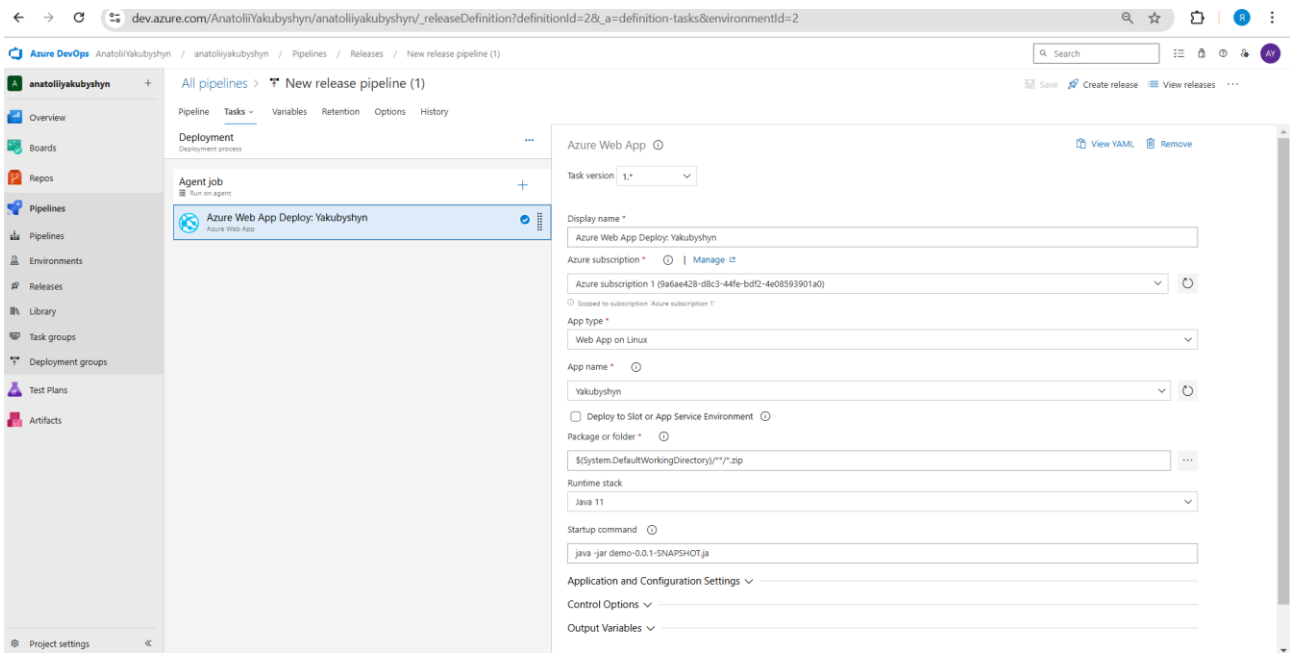

```yaml
trigger:
- main

pool:
  name: default
  demands:
    - agent.name -equals Ubuntu-runner

stages:
- stage: Build
  displayName: Build and Test
  jobs:
    - job: Package
      steps:
        - script: |
            sudo apt update
            sudo apt install -y openjdk-11-jdk
            java -version
          displayName: 'Install Java 11'

        - script: |
            sudo apt install -y maven
            mvn -v
          displayName: 'Install Maven'

        - script: mvn package
          displayName: 'Run mvn package'

        - script: sudo apt-get -y install zip
          displayName: 'Install zip'

        - task: ArchiveFiles@2
          inputs:
            rootFolderOrFile: 'target/demo-0.0.1-SNAPSHOT.jar'
            archiveType: 'zip'
            archiveFile: '$(Build.ArtifactStagingDirectory)/build.zip'

        - task: PublishBuildArtifacts@1
          displayName: "Publish Build Artifacts"
          inputs:
            pathToPublish: '$(Build.ArtifactStagingDirectory)/build.zip'
            artifactName: 'build'

- stage: Deploy
  dependsOn: Build
  displayName: Deploy
  condition: succeeded()
  jobs:
    - job: Deploy
      steps:
        - task: DownloadBuildArtifacts@0
          displayName: "Download Build Artifacts"
          inputs:
            buildType: "current"
            artifactName: "build"
            downloadPath: "$(System.ArtifactsDirectory)"

        - task: AzureRmWebAppDeployment@5
          inputs:
            ConnectionType: 'AzureRM'
            azureSubscription: 'Azure subscription 1 (9a6ae428-d8c3-44fe-bdf2-4e0859301a0)'
            appType: 'webAppLinux'
            WebAppName: 'Yakubyshyn'
            packageForLinux: '$(Pipeline.Workspace)/**/build/build.zip'
            RuntimeStack: 'JAVA:11-java11'
            StartupCommand: 'java -jar demo-0.0.1-SNAPSHOT.jar --server.port=80'
            DeploymentTypeLinux: 'oneDeploy'
```

3. Configure Release Pipeline:

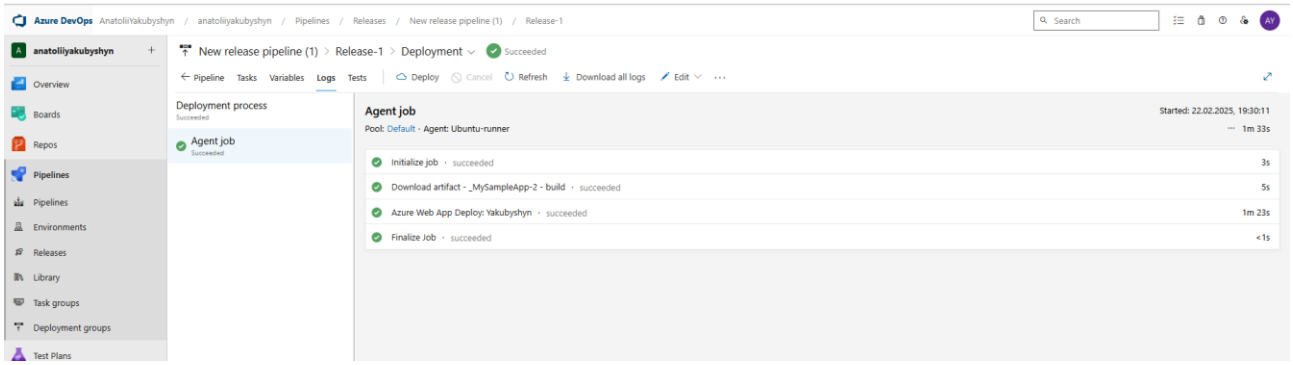
a. Create a release pipeline in Azure DevOps that triggers automatically after a successful build.

b. Add an Azure Web App deployment task to the release pipeline, specifying the Azure subscription and the Web App name.





4. Deploy the Application: a. Save and run the release pipeline. b. Verify that the application is successfully deployed and accessible via the web.

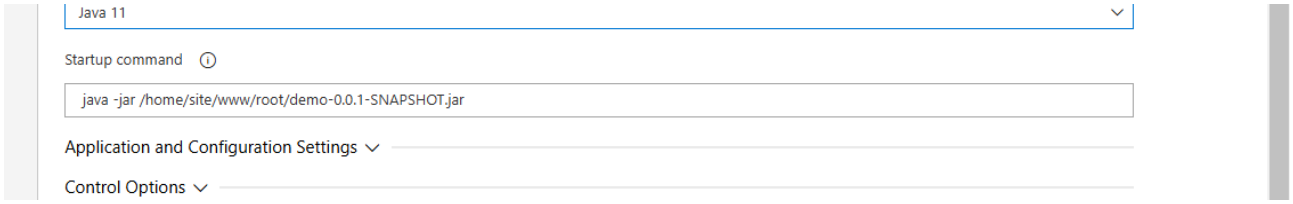


Hello World!

Fix:

Jar was uploaded inside this /home/site/……

Fixed in release pipeline and just in pipeline and started to work.



Practical Task 6: Implement CI/CD with GitHub and Azure Pipelines

Requirements:

1. Create a GitHub Repository:

a. Navigate to GitHub and create a new repository named MySampleApp.

2. Push Application Code:

a. Push the application code from Task 4 to the new GitHub repository.

3. Set Up Azure Pipeline:

a. In Azure DevOps, create a new pipeline that uses the GitHub repository as the source.

b. Authenticate with GitHub and select the repository.

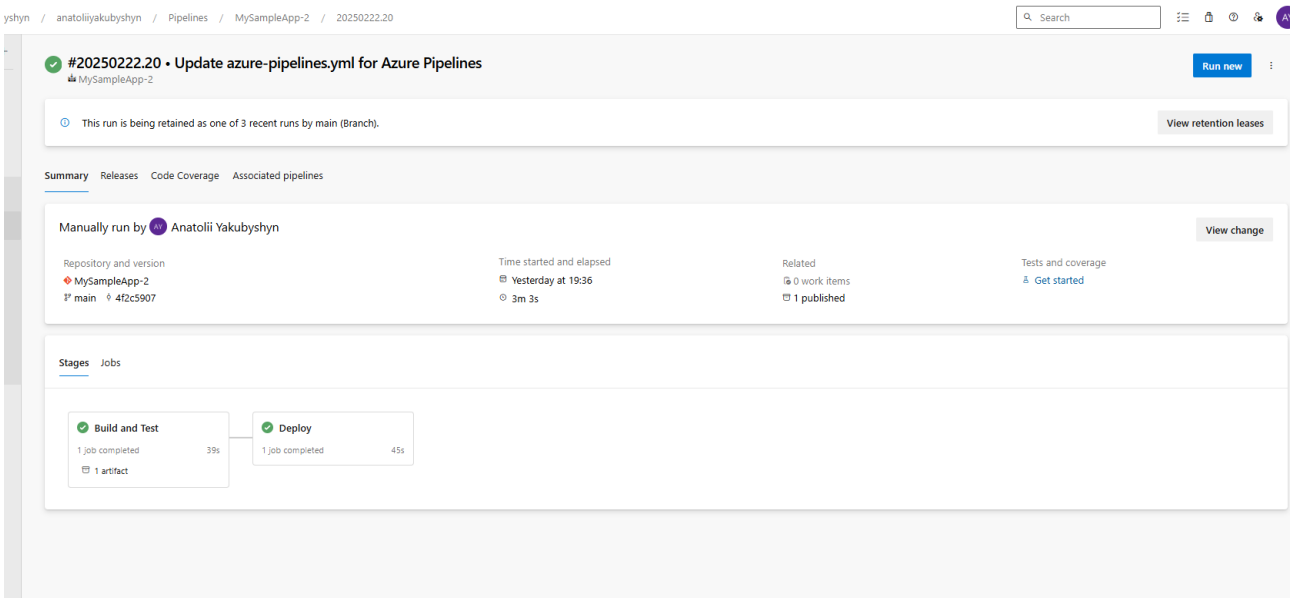
4. Define the Pipeline:

a. Use the YAML pipeline editor to define the pipeline with steps to:

i. Build the application. ii. Run tests. iii. Deploy to the Azure Web App (similar to Task 5).

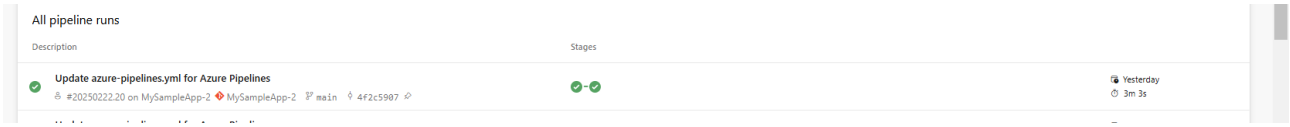
5. Enable GitHub Integration: a. Configure the pipeline to trigger automatically on every push to the main branch. b. Verify that the pipeline triggers successfully and the application is deployed to Azure.

Actually, I have done all of this steps in task 5. In task5 I did it in two ways: I have added deployment stage + using azure release pipelines. Both ways worked.



But in task 5 (I used another code then in task 4 because I had problems with deployment of static content that react produces (I suppose for this is needed another service like Blob storage))

```yaml
trigger:
- main

pool:
  name: default
  demands:
  - agent.name -equals Ubuntu-runner

stages:
- stage: Build
  displayName: Build and Test
  jobs:
  - job: Package
    steps:
    - script: |
        sudo apt update
        sudo apt install -y openjdk-11-jdk
        java -version
      displayName: 'Install Java 11'

    - script: |
        sudo apt install -y maven
        mvn -v
      displayName: 'Install Maven'

    - script: mvn package
      displayName: 'Run mvn package'

    - script: sudo apt-get -y install zip
      displayName: 'install zip'

    - task: ArchiveFiles@2
      inputs:
        rootFolderOrFile: 'target/demo-0.0.1-SNAPSHOT.jar'
        archiveType: 'zip'
        archiveFile: '$(Build.ArtifactStagingDirectory)/build.zip'

    - task: PublishBuildArtifacts@1
      displayName: "Publish Build Artifacts"
      inputs:
        pathToPublish: '$(Build.ArtifactStagingDirectory)/build.zip'
        artifactName: 'build'

- stage: Deploy
  dependsOn: Build
  displayName: Deploy
  condition: succeeded()
  jobs:
  - job: Deploy
    steps:
    - task: DownloadBuildArtifacts@0
      displayName: "Download Build Artifacts"
      inputs:
        buildType: "current"
        artifactName: "build"
        downloadPath: "$(System.ArtifactsDirectory)"

    - task: AzureRmWebAppDeployment@5
      inputs:
        ConnectionType: 'AzureRM'
        azureSubscription: 'Azure subscription 1 (9a6a0428-d8c3-44fe-bdf2-4e08593981a0)'
        appType: 'webAppLinux'
        WebAppName: 'Yakubyshyn'
        packageForLinux: '$(Pipeline.Workspace)/**/build/build.zip'
        RuntimeStack: 'JAVA:11-java11'
        StartupCommand: 'java -jar /home/site/wwwroot/demo-0.0.1-SNAPSHOT.jar --server.port=80'
        DeploymentTypeLinux: 'oneDeploy'
```

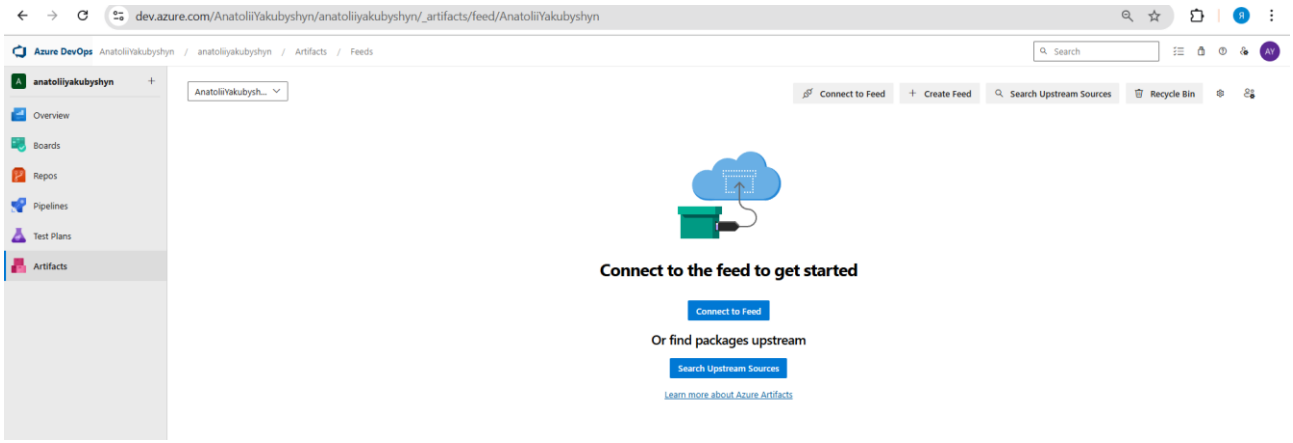


Practical Task 7: Use Azure Artifacts to Manage Dependencies

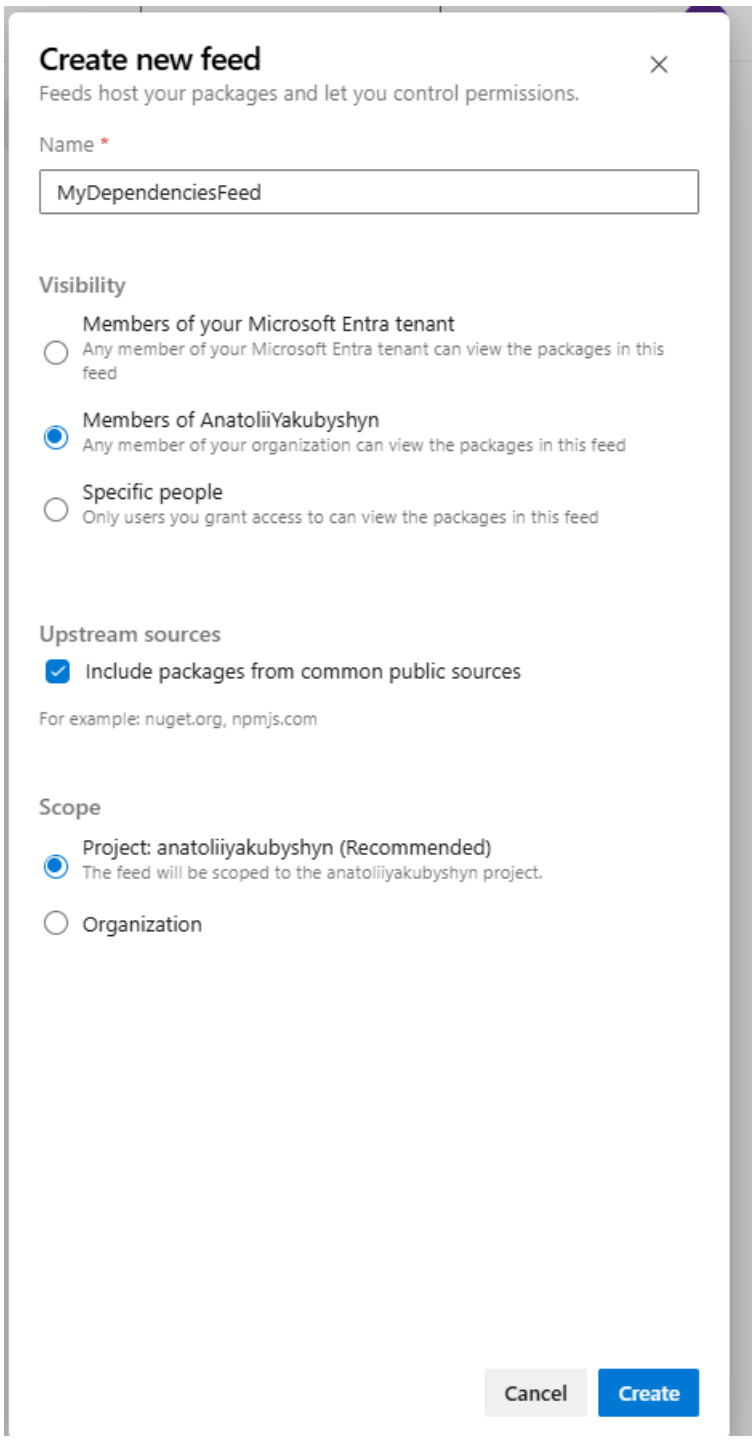
Requirements:

1. Create an Azure Artifacts Feed:

a. In Azure DevOps, navigate to Artifacts and create a new feed named MyDependenciesFeed.



Create Feed


Create new feed
Feeds host your packages and let you control permissions.
Name *
MyDependenciesFeed
Visibility
Members of your Microsoft Entra tenant
Any member of your Microsoft Entra tenant can view the packages in this feed
Members of AnatoliiYakubyshyn
Any member of your organization can view the packages in this feed
Specific people
Only users you grant access to can view the packages in this feed
Upstream sources
Include packages from common public sources
For example: nuget.org, npmjs.com
Scope
Project: anatoliiyakubyshyn (Recommended)
The feed will be scoped to the anatoliiyakubyshyn project.
Organization
Cancel
Create

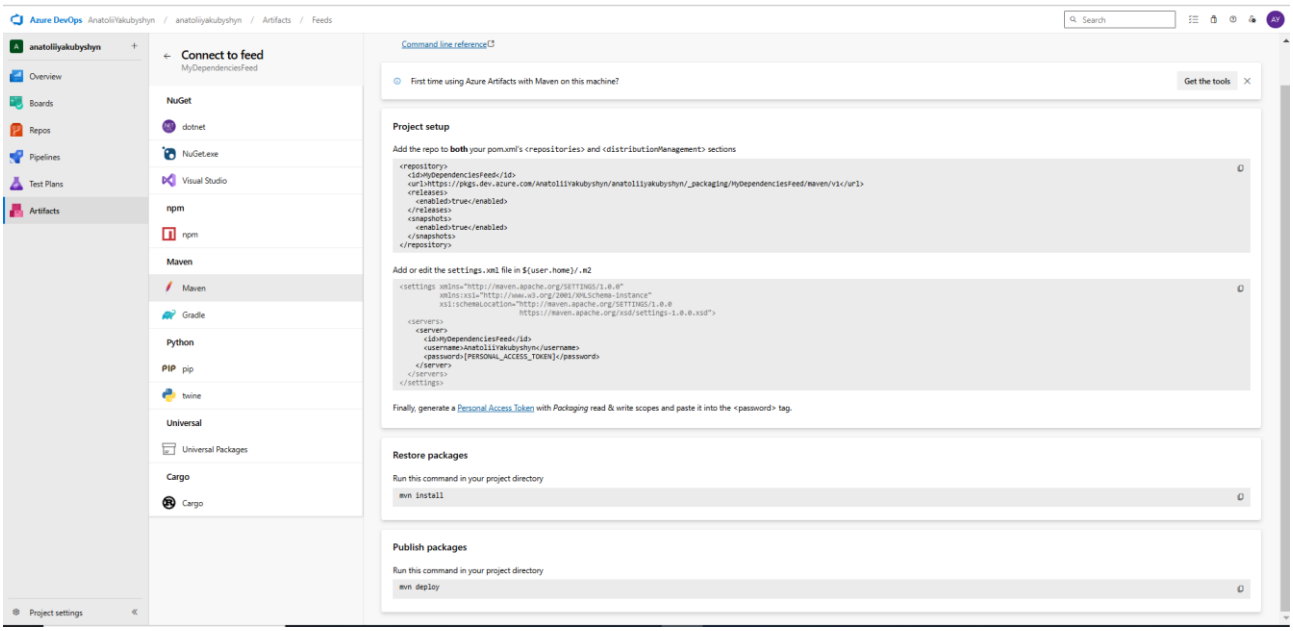
2. Publish Dependencies:

a. Modify the build pipeline to include a task that publishes the application's dependencies

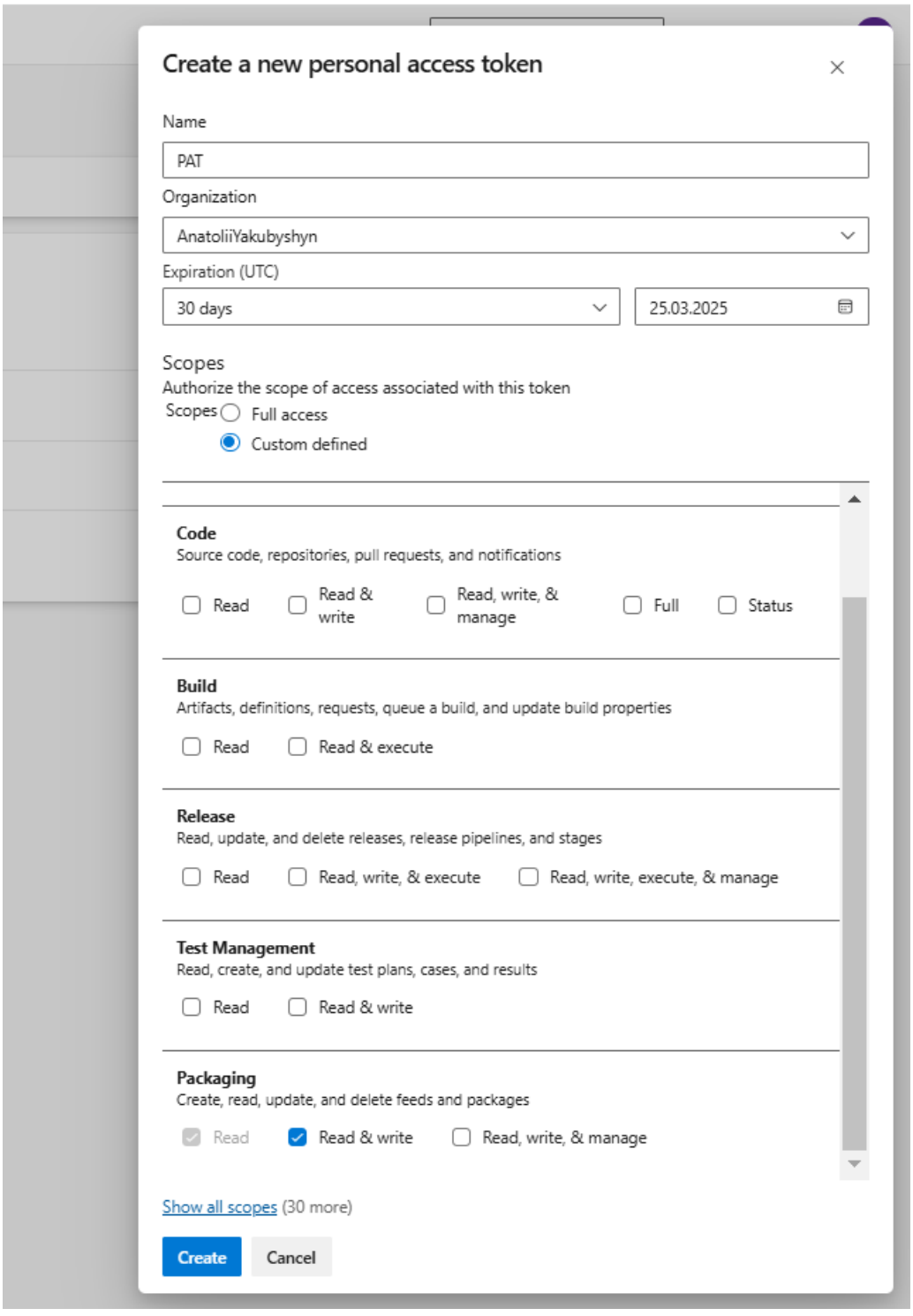
(e.g., npm packages or NuGet packages) to the Azure Artifacts feed.

b. Use commands like npm publish or dotnet nuget push in the pipeline.

3. Restore Dependencies: a. Update the build pipeline to



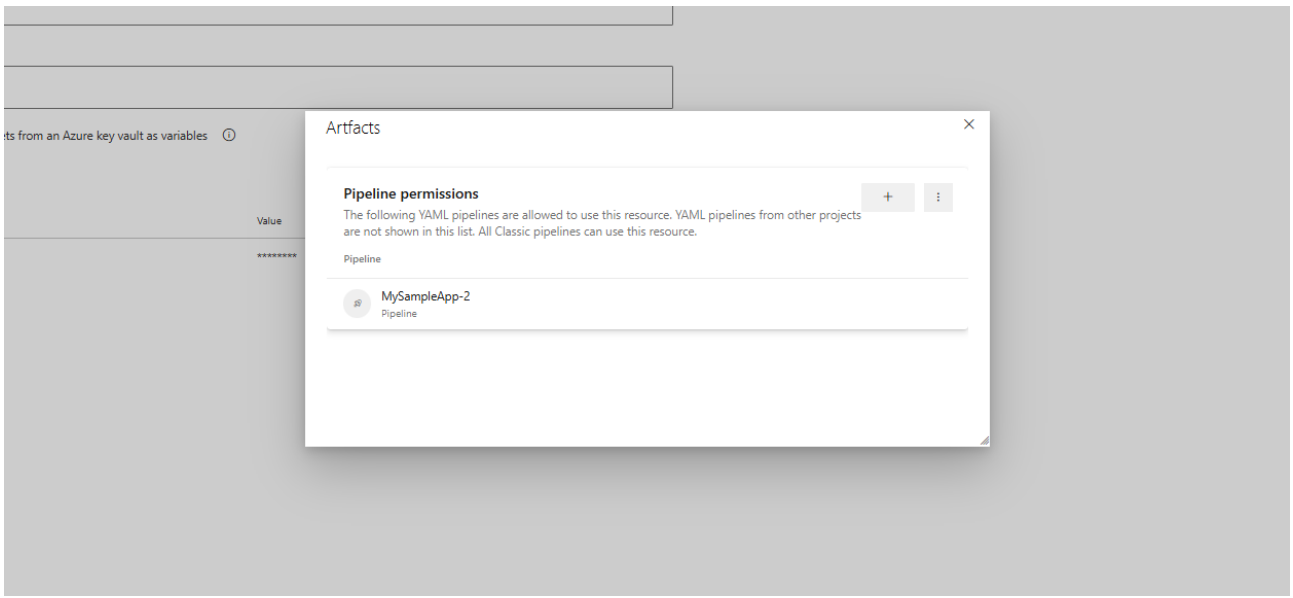

```
<repositories>
    <repository>
        <id>MyDependenciesFeed</id>
        <url>
            https://pkgs.dev.azure.com/AnatoliiYakubyshyn/anatoliiyakubyshyn/_packaging/MyDependenciesFeed/maven/v1
        </url>
        <releases>
            <enabled>true</enabled>
        </releases>
        <snapshots>
            <enabled>true</enabled>
        </snapshots>
    </repository>
</repositories>
<distributionManagement>
    <repository>
        <id>MyDependenciesFeed</id>
        <url>https://pkgs.dev.azure.com/AnatoliiYakubyshyn/anatoliiyakubyshyn/_packaging/MyDependenciesFeed/maven/v1</url>
        <releases>
            <enabled>true</enabled>
        </releases>
        <snapshots>
            <enabled>true</enabled>
        </snapshots>
    </repository>
</distributionManagement>
```

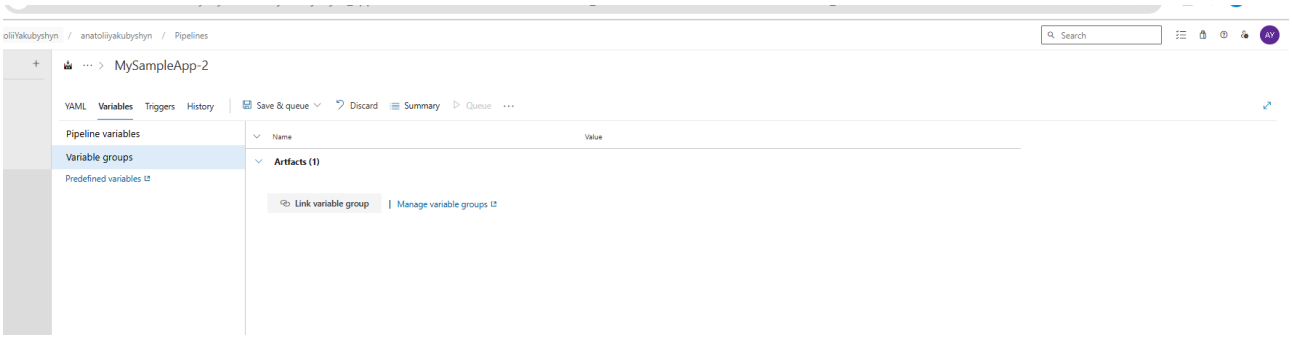

Create a new personal access token
Name
PAT
Organization
AnatoliiYakubyshyn
Expiration (UTC)
30 days
25.03.2025
Scopes
Authorize the scope of access associated with this token
Scopes
Full access
Custom defined
Code
Source code, repositories, pull requests, and notifications
Read
Read & write
Read, write, & manage
Full
Status
Build
Artifacts, definitions, requests, queue a build, and update build properties
Read
Read & execute
Release
Read, update, and delete releases, release pipelines, and stages
Read
Read, write, & execute
Read, write, execute, & manage
Test Management
Read, create, and update test plans, cases, and results
Read
Read & write
Packaging
Create, read, update, and delete feeds and packages
Read
Read & write
Read, write, & manage
Show all scopes (30 more)
Create
Cancel

```yaml
trigger:
- main

pool:
  name: default
  demands:
    - agent.name -equals Ubuntu-runner

variables:
  - group: "Artifacts"

stages:
- stage: Build
  displayName: Build and Test
  jobs:
    - job: Publish_Dependencies
      steps:
        - script: |
            sudo apt update
            sudo apt install -y openjdk-11-jdk
            java -version
          displayName: 'Install Java 11'

        - script: |
            sudo apt install -y maven
            mvn -v
          displayName: 'Install Maven'
        - script: |
            echo "<settings xmlns=\"http://maven.apache.org/SETTINGS/1.0.0\"
                  xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"
                  xsi:schemaLocation=\"http://maven.apache.org/SETTINGS/1.0.0
                                      https://maven.apache.org/xsd/settings-1.0.0.xsd\">
              <servers>
                <server>
                  <id>MyDependenciesFeed</id>
                  <username>AnatoliiYakubyshyn</username>
                  <password>$(PAT)</password>
                </server>
              </servers>
            </settings>" > ~/.m2/settings.xml
          displayName: 'Generate settings.xml'

        - script:
            mvn deploy
```

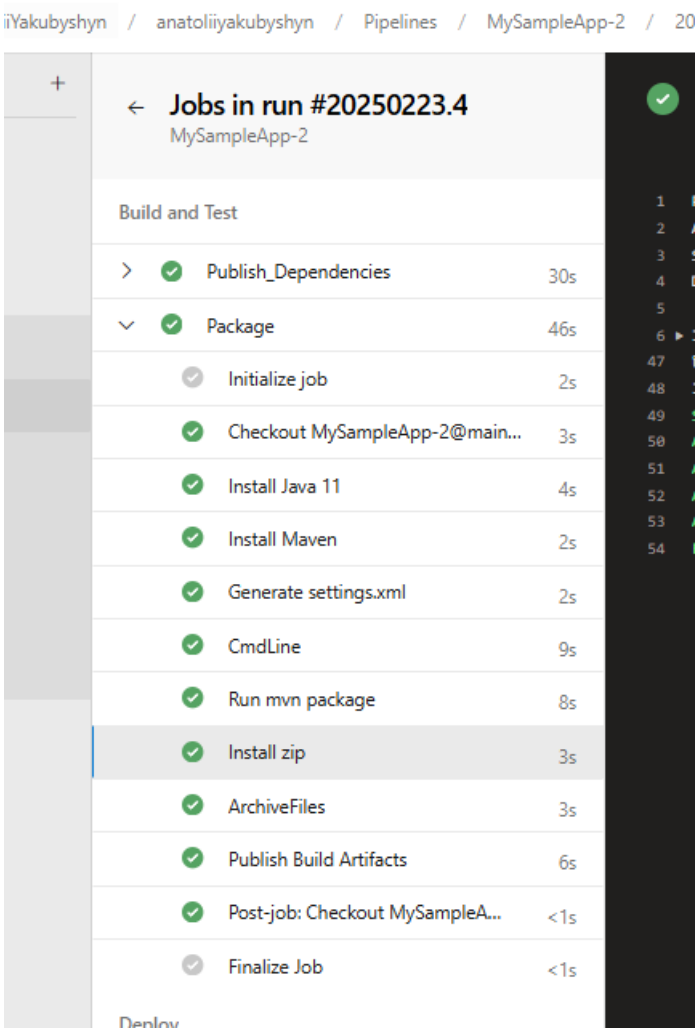
```
    - script:
        mvn deploy
  - job: Package
    dependsOn: Publish_Dependencies
    steps:
      - script: |
          sudo apt update
          sudo apt install -y openjdk-11-jdk
          java -version
        displayName: 'Install Java 11'

      - script: |
          sudo apt install -y maven
          mvn -v
        displayName: 'Install Maven'
      - script: |
          echo "<settings xmlns=\"http://maven.apache.org/SETTINGS/1.0.0\"
                xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"
                xsi:schemaLocation=\"http://maven.apache.org/SETTINGS/1.0.0
                                    https://maven.apache.org/xsd/settings-1.0.0.xsd\">
            <servers>
              <server>
                <id>MyDependenciesFeed</id>
                <username>AnatoliiYakubyshyn</username>
                <password>$(PAT)</password>
              </server>
            </servers>
          </settings>" > ~/.m2/settings.xml
        displayName: 'Generate settings.xml'

      - script: mvn install

      - script: mvn package
```

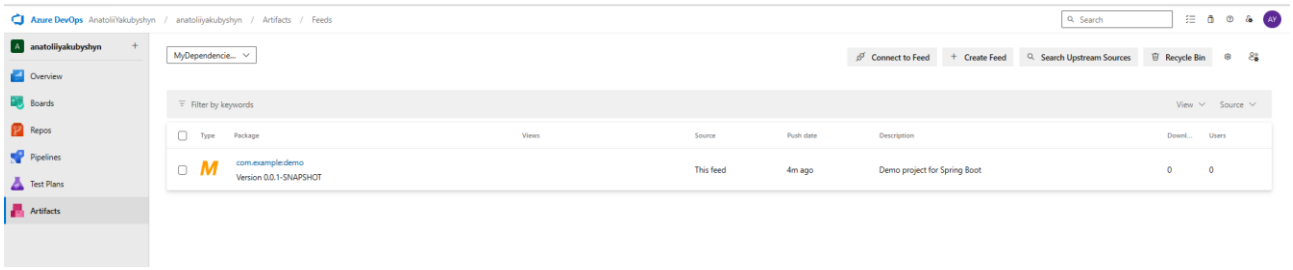
Adjustment


ts from an Azure key vault as variables
Value
****
Artfacts
Pipeline permissions
The following YAML pipelines are allowed to use this resource. YAML pipelines from other projects are not shown in this list. All Classic pipelines can use this resource.
Pipeline
MySampleApp-2
Pipeline


oliiYakubyshyn / anatoliiyakubyshyn / Pipelines
Search
MySampleApp-2
YAML
Variables
Triggers
History
Save & queue
Discard
Summary
Queue
Pipeline variables
Variable groups
Predefined variables
Name
Value
Artfacts (1)
Link variable group
Manage variable groups

Verify Dependency Management:

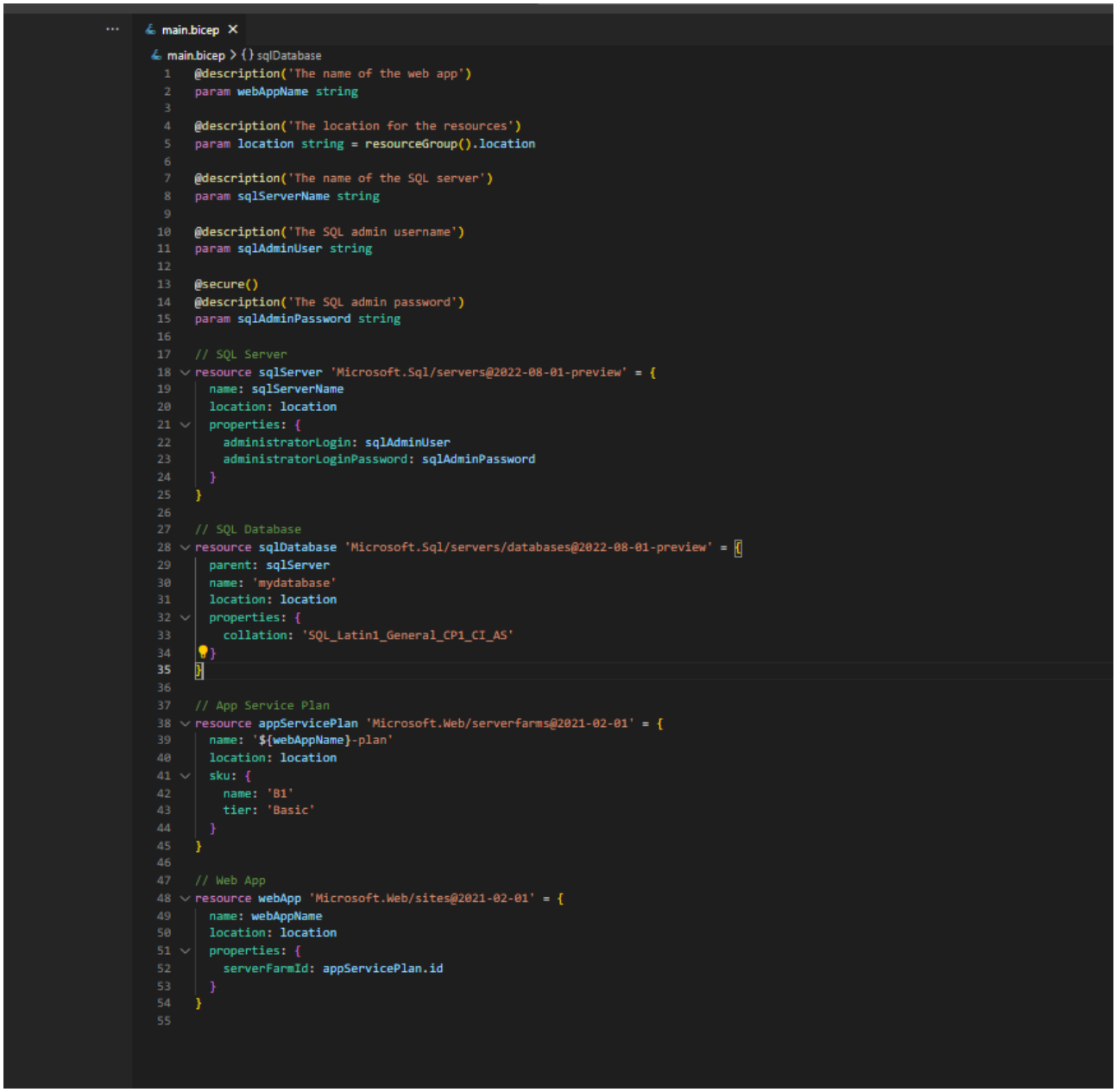




Practical Task 8: Deploy Infrastructure as Code (IaC) with Bicep

1. Create a Bicep File:

a. Create a Bicep file named main.bicep that defines the infrastructure for your application, including an Azure Web App and a SQL Database.



```bicep
@description('The name of the web app')
param webAppName string

@description('The location for the resources')
param location string = resourceGroup().location

@description('The name of the SQL server')
param sqlServerName string

@description('The SQL admin username')
param sqlAdminUser string

@secure()
@description('The SQL admin password')
param sqlAdminPassword string

// SQL Server
resource sqlServer 'Microsoft.Sql/servers@2022-08-01-preview' = {
  name: sqlServerName
  location: location
  properties: {
    administratorLogin: sqlAdminUser
    administratorLoginPassword: sqlAdminPassword
  }
}

// SQL Database
resource sqlDatabase 'Microsoft.Sql/servers/databases@2022-08-01-preview' = {
  parent: sqlServer
  name: 'mydatabase'
  location: location
  properties: {
    collation: 'SQL_Latin1_General_CP1_CI_AS'
  }
}

// App Service Plan
resource appServicePlan 'Microsoft.Web/serverfarms@2021-02-01' = {
  name: '${webAppName}-plan'
  location: location
  sku: {
    name: 'B1'
    tier: 'Basic'
  }
}

// Web App
resource webApp 'Microsoft.Web/sites@2021-02-01' = {
  name: webAppName
  location: location
  properties: {
    serverFarmId: appServicePlan.id
  }
}
```

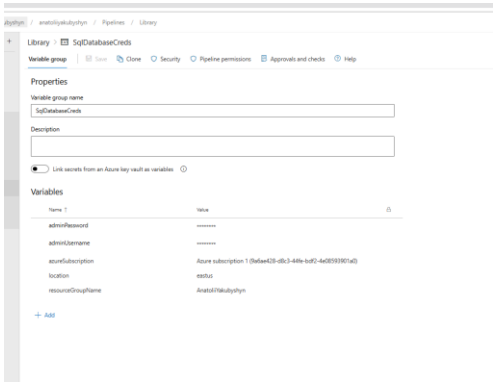
2. Create a New Pipeline:

a. Set up a new pipeline in Azure DevOps that includes a stage for deploying the Bicep file.

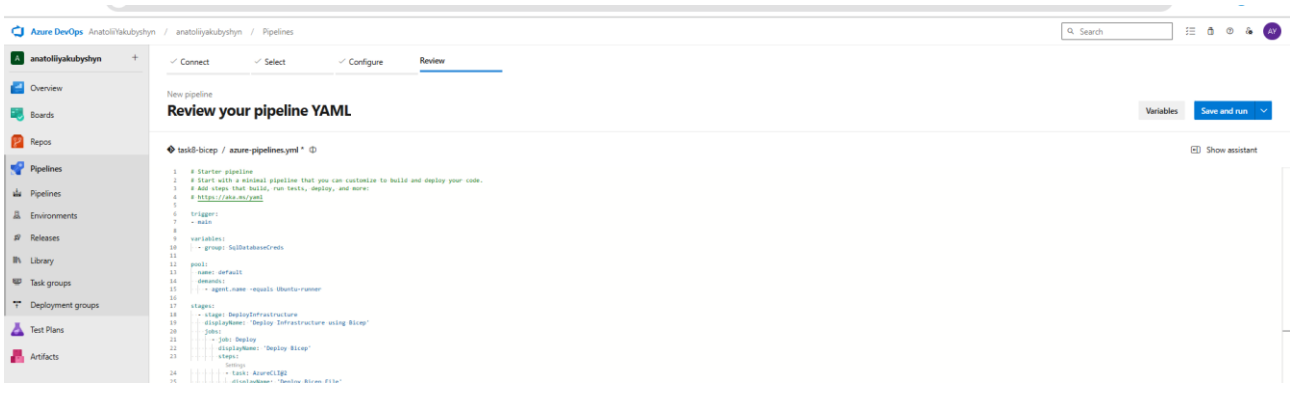
3. Deploy Using Azure CLI:

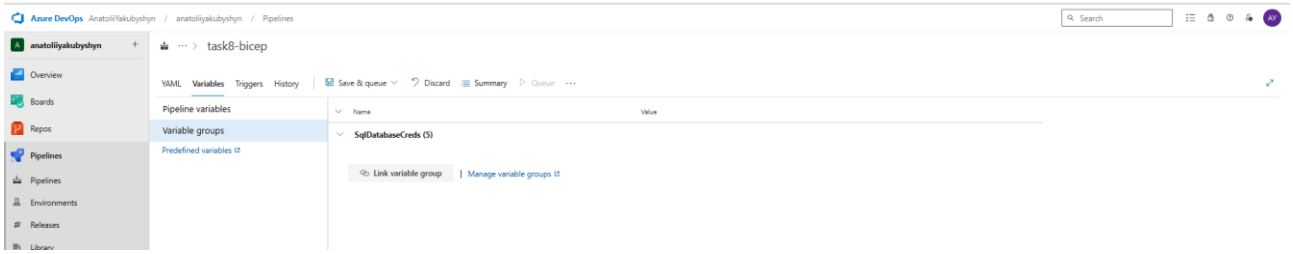
a. Use the Azure CLI task in the pipeline to deploy the Bicep file.

b. The command should look like: az deployment group create --resource-group --template-file main.bicep



4. Run the Pipeline: a. Save and run the pipeline to ensure that the infrastructure is provisioned correctly in Azure. b. Verify that the resources are created in the Azure portal.





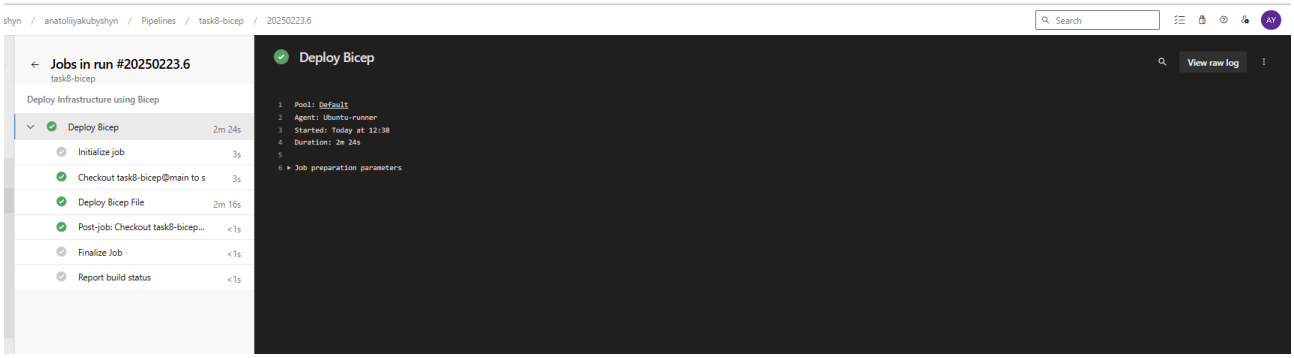
Link secrets from an Azure key vault as variables

Variables

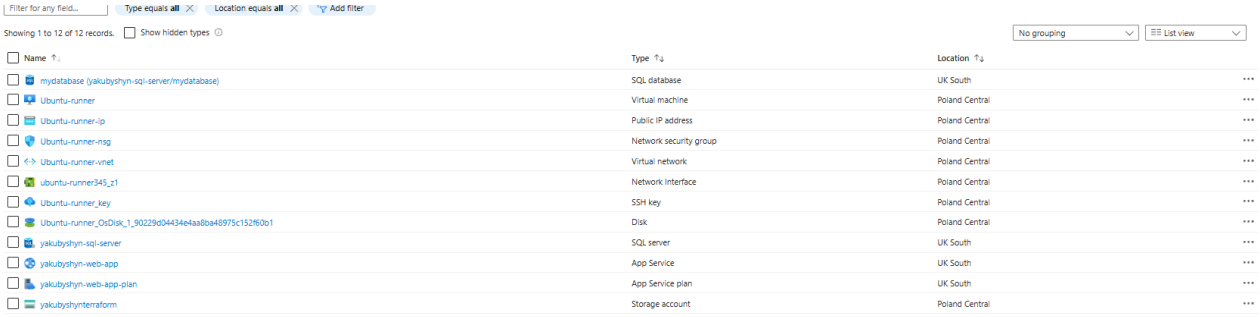
Name	Value
adminPassword	********
adminUsername	********
azureSubscription	Azure subscription 1 (9a6ae428-d8c3-44fe-bdf2-4e08593901a0)
location	uksouth
resourceGroupName	AnatoliiYakubyshyn

+ Add

changed location to uksouth and :



Resources created successfully

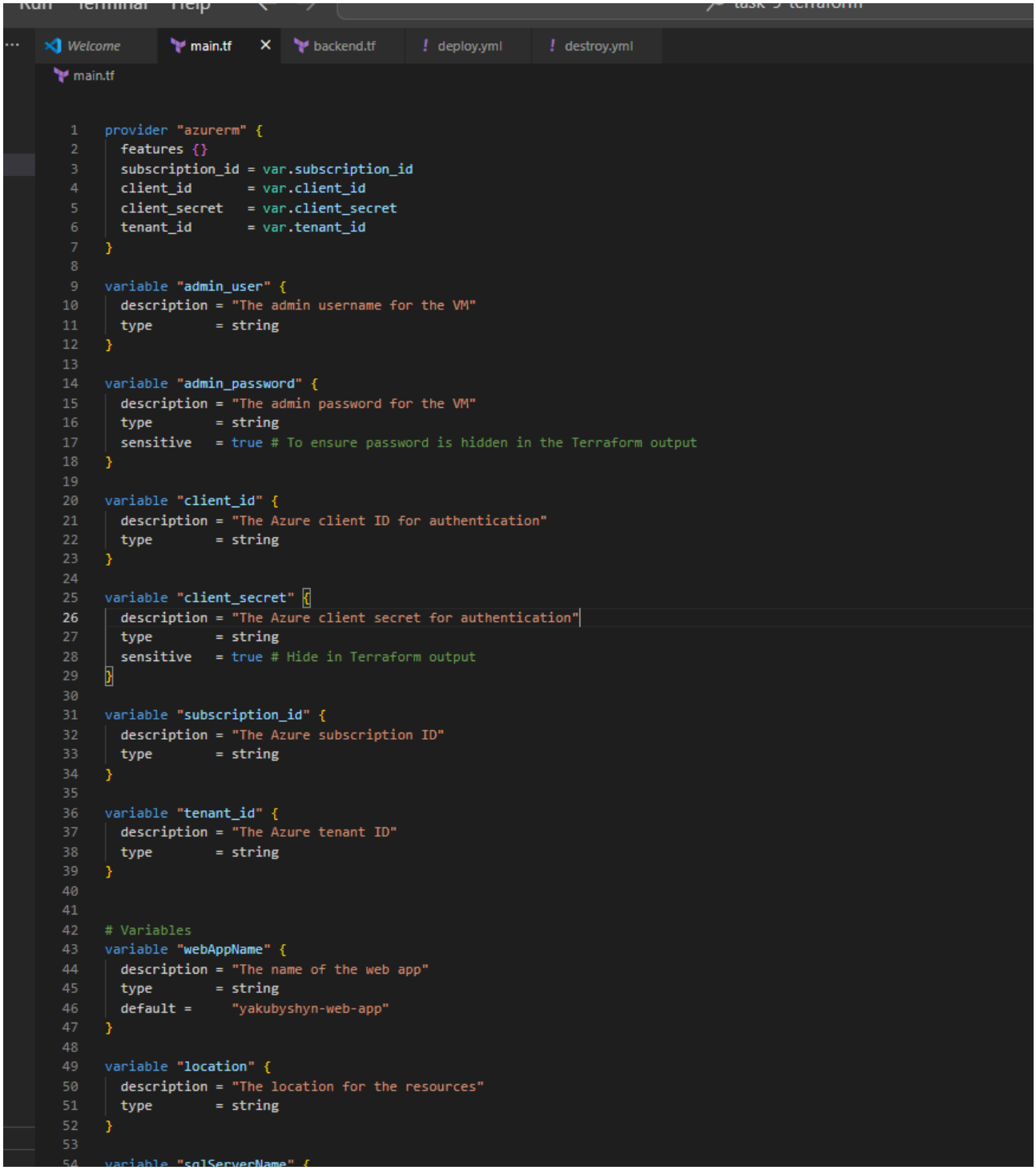


Name	Type	Location
mydatabase (yakubyshyn-sql-server/mydatabase)	SQL database	UK South
Ubuntu-runner	Virtual machine	Poland Central
Ubuntu-runner-ip	Public IP address	Poland Central
Ubuntu-runner-nsg	Network security group	Poland Central
Ubuntu-runner-vnet	Virtual network	Poland Central
ubuntu-runner345_z1	Network interface	Poland Central
Ubuntu-runner_key	SSH key	Poland Central
Ubuntu-runner_OsDisk_1_90229d04434e4aa8ba48975c152f60b1	Disk	Poland Central
yakubyshyn-sql-server	SQL server	UK South
yakubyshyn-web-app	App Service	UK South
yakubyshyn-web-app-plan	App Service plan	UK South
yakubyshynterraform	Storage account	Poland Central

Practical Task 9: Implement Infrastructure as Code (IaC) with Terraform

Requirements: 1.

Create a Terraform Configuration: a. Create a Terraform configuration file (main.tf) that defines the same infrastructure as in Task 8 (e.g., Azure Web App and SQL Database)




Welcome
main.tf
backend.tf
deploy.yml
destroy.yml
main.tf
variable "location" {
description = "The location for the resources"
type = string
}
variable "sqlServerName" {
description = "The name of the SQL server"
type = string
default = "yakubyshyn-sql-server"
}
variable "sqlAdminUser" {
description = "The SQL admin username"
type = string
sensitive = true
}
variable "sqlAdminPassword" {
description = "The SQL admin password"
type = string
sensitive = true
}
resource "azurerm_mssql_server" "sql_server" {
name = var.sqlServerName
location = var.location
resource_group_name = "AnatoliiYakubyshyn"
version = "12.0"
administrator_login = var.sqlAdminUser
administrator_login_password = var.sqlAdminPassword
}
resource "azurerm_mssql_database" "sql_database" {
name = "mydatabase"
server_id = azurerm_mssql_server.sql_server.id # Correct way to reference the server
collation = "SQL_Latin1_General_CP1_CI_AS"
}
resource "azurerm_app_service_plan" "app_service_plan" {
name = "${var.webAppName}-plan"
location = var.location
resource_group_name = "AnatoliiYakubyshyn"
sku {
tier = "Basic"
size = "B1"
}
}
resource "azurerm_linux_web_app" "web_app" {
name = var.webAppName
location = var.location
resource_group_name = "AnatoliiYakubyshyn"
service_plan_id = azurerm_app_service_plan.app_service_plan.id
site_config {}

un Terminal Help task-9-terraform

Welcome | main.tf | backend.tf | deploy.yml | destroy.yml

backend.tf

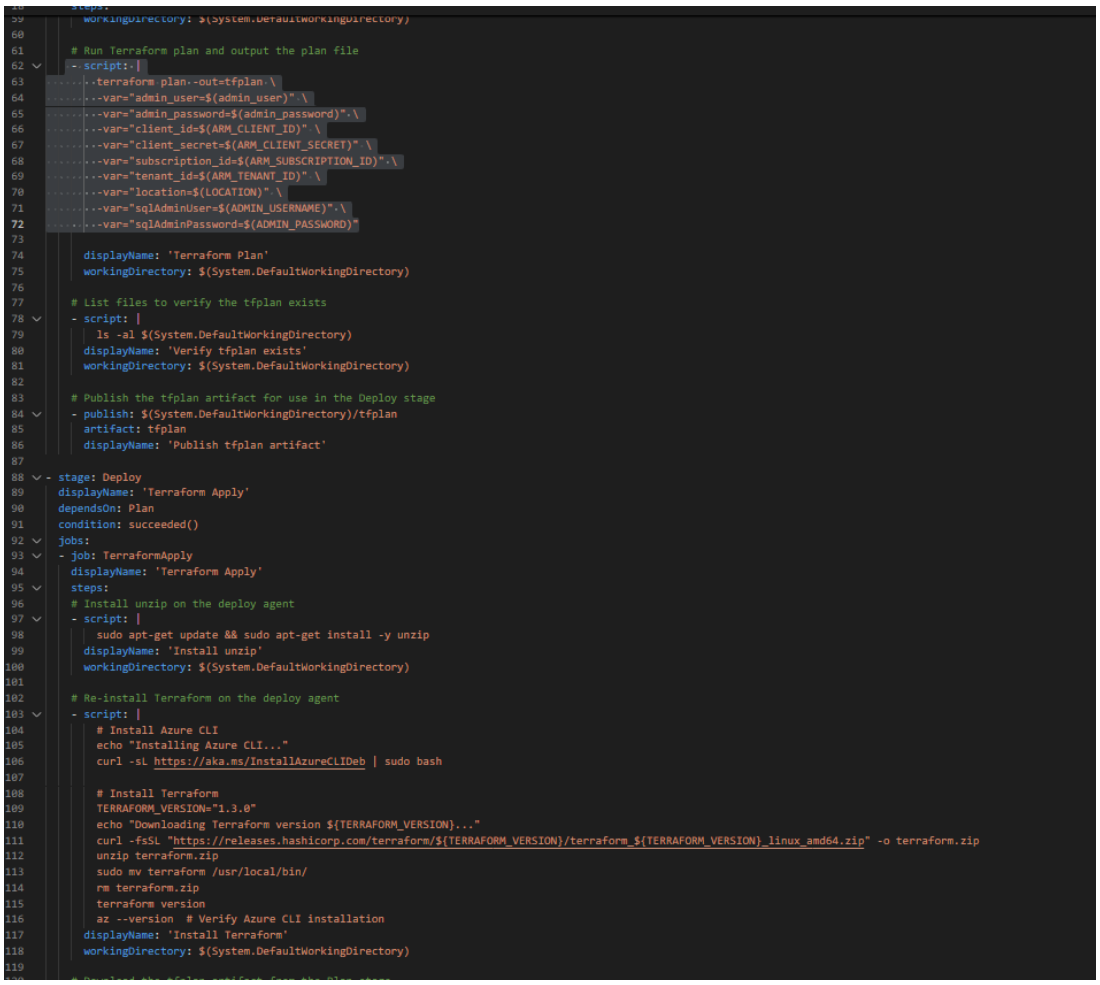
```hcl
terraform {
  backend "azurerm" {
    resource_group_name  = "AnatoliiYakubyshyn"  # Static value
    storage_account_name = "yakubyshynterraform" # Default value to be overridden
    container_name       = "tfstate"             # Static value
    key                  = "terraform.tfstate"   # Static value
  }
}
```

2. Set Up a New Azure Pipeline: a. Create a new Azure Pipeline that runs Terraform commands. b. Use the following steps in the pipeline: terraform init to initialize the Terraform working directory. terraform plan to create an execution plan. terraform apply -auto-approve to apply the changes

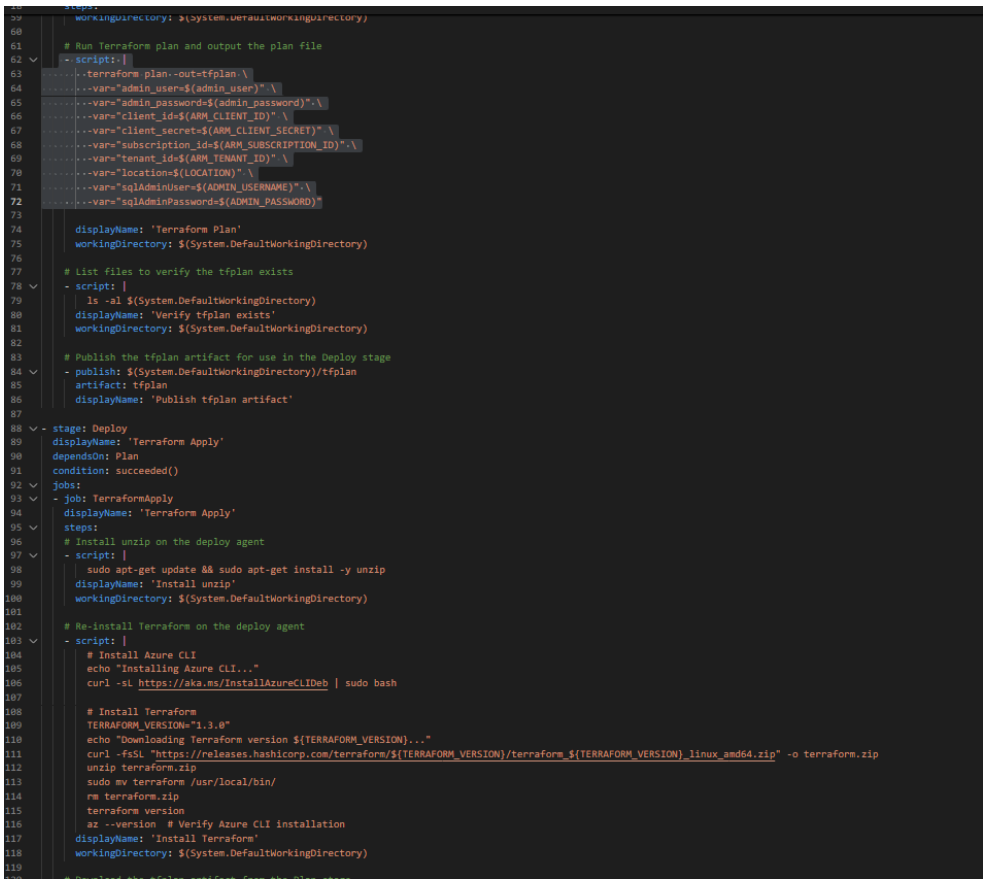
3. Configure Triggers: a. Set up the pipeline to trigger on changes to the Terraform files in the repository.

deploy.yml > [ ] stages > {} 0 > [ ] jobs > {} 0 > [ ] steps > {} 4 > script

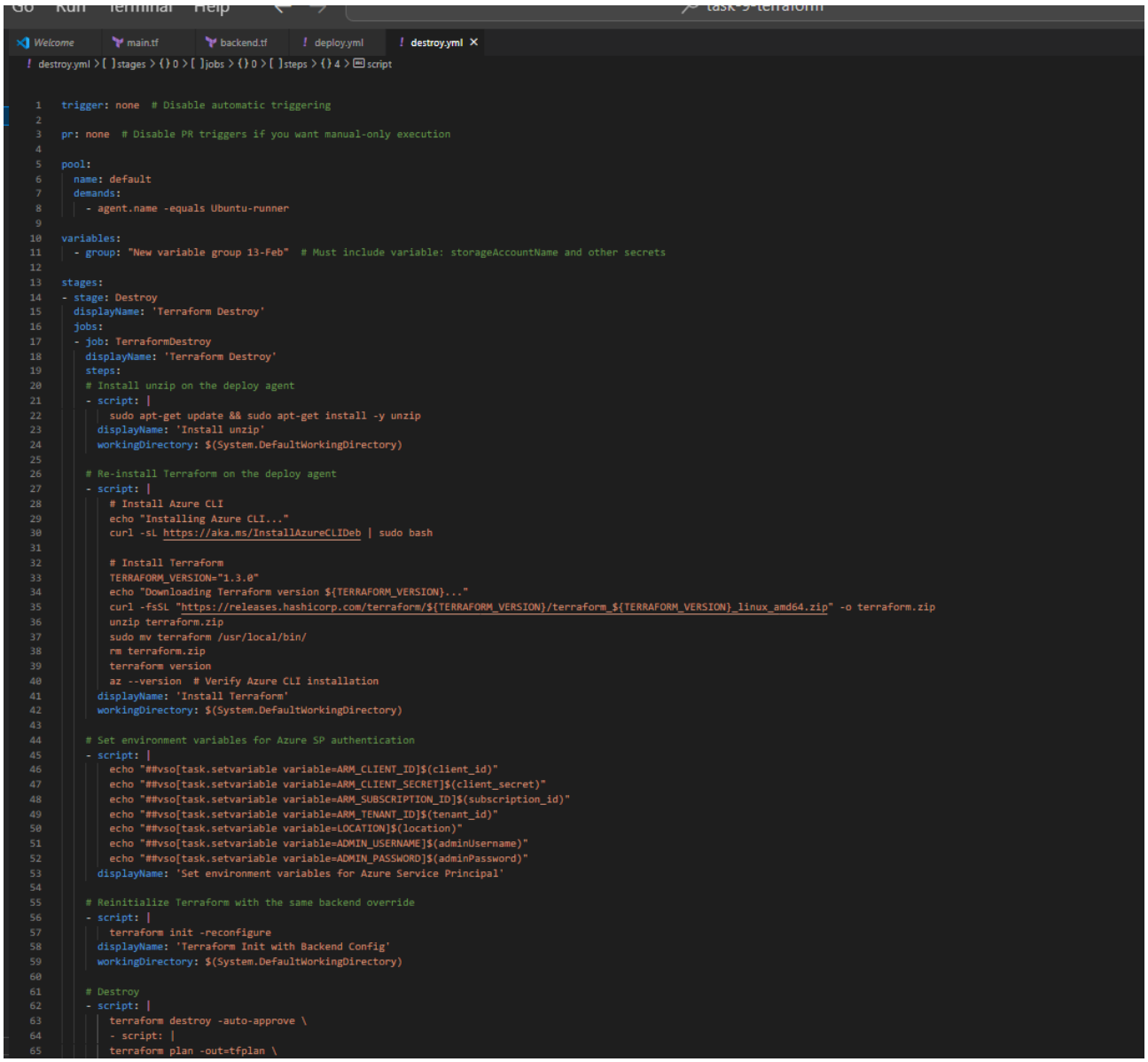
```yaml
trigger:
- main

pool:
  name: default
  demands:
    - agent.name -equals Ubuntu-runner

variables:
  - group: "New variable group 13-Feb"  # Must include variable: storageAccountName and other secrets

stages:
- stage: Plan
  displayName: 'Terraform Plan'
  jobs:
  - job: TerraformPlan
    displayName: 'Terraform Plan'
    steps:
    # Install unzip if not present
    - script: |
        sudo apt-get update && sudo apt-get install -y unzip
      displayName: 'Install unzip'
      workingDirectory: $(System.DefaultWorkingDirectory)

    # Install Terraform and Azure CLI
    - script: |
        # Install Azure CLI
        echo "Installing Azure CLI..."
        curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash

        # Install Terraform
        TERRAFORM_VERSION="1.3.0"
        echo "Downloading Terraform version ${TERRAFORM_VERSION}..."
        curl -fsSL "https://releases.hashicorp.com/terraform/${TERRAFORM_VERSION}/terraform_${TERRAFORM_VERSION}_linux_amd64.zip" -o terraform.zip
        unzip terraform.zip
        sudo mv terraform /usr/local/bin/
        rm terraform.zip
        terraform version
        az --version  # Verify Azure CLI installation
      displayName: 'Install Terraform and Azure CLI'
      workingDirectory: $(System.DefaultWorkingDirectory)

    # Set environment variables for Azure SP authentication
    - script: |
        echo "##vso[task.setvariable variable=ARM_CLIENT_ID]$(client_id)"
        echo "##vso[task.setvariable variable=ARM_CLIENT_SECRET]$(client_secret)"
        echo "##vso[task.setvariable variable=ARM_SUBSCRIPTION_ID]$(subscription_id)"
        echo "##vso[task.setvariable variable=ARM_TENANT_ID]$(tenant_id)"
        echo "##vso[task.setvariable variable=LOCATION]$(location)"
        echo "##vso[task.setvariable variable=ADMIN_USERNAME]$(adminUsername)"
        echo "##vso[task.setvariable variable=ADMIN_PASSWORD]$(adminPassword)"

      displayName: 'Set environment variables for Azure Service Principal'

    # Initialize Terraform with backend config override
    - script: |
        terraform init -reconfigure
      displayName: 'Terraform Init with Backend Config'
      workingDirectory: $(System.DefaultWorkingDirectory)

    # Run Terraform plan and output the plan file
    - script: |
        terraform plan -out=tfplan \
        -var="admin_user=$(admin_user)" \
        -var="admin_password=$(admin_password)" \
```



```
    workingDirectory: $(System.DefaultWorkingDirectory)

    # Run Terraform plan and output the plan file
    - script: |
        terraform plan -out=tfplan \
        -var="admin_user=$(admin_user)" \
        -var="admin_password=$(admin_password)" \
        -var="client_id=$(ARM_CLIENT_ID)" \
        -var="client_secret=$(ARM_CLIENT_SECRET)" \
        -var="subscription_id=$(ARM_SUBSCRIPTION_ID)" \
        -var="tenant_id=$(ARM_TENANT_ID)" \
        -var="location=$(LOCATION)" \
        -var="sqlAdminUser=$(ADMIN_USERNAME)" \
        -var="sqlAdminPassword=$(ADMIN_PASSWORD)"

      displayName: 'Terraform Plan'
      workingDirectory: $(System.DefaultWorkingDirectory)

    # List files to verify the tfplan exists
    - script: |
        ls -al $(System.DefaultWorkingDirectory)
      displayName: 'Verify tfplan exists'
      workingDirectory: $(System.DefaultWorkingDirectory)

    # Publish the tfplan artifact for use in the Deploy stage
    - publish: $(System.DefaultWorkingDirectory)/tfplan
      artifact: tfplan
      displayName: 'Publish tfplan artifact'

- stage: Deploy
  displayName: 'Terraform Apply'
  dependsOn: Plan
  condition: succeeded()
  jobs:
  - job: TerraformApply
    displayName: 'Terraform Apply'
    steps:
    # Install unzip on the deploy agent
    - script: |
        sudo apt-get update && sudo apt-get install -y unzip
      displayName: 'Install unzip'
      workingDirectory: $(System.DefaultWorkingDirectory)

    # Re-install Terraform on the deploy agent
    - script: |
        # Install Azure CLI
        echo "Installing Azure CLI..."
        curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash

        # Install Terraform
        TERRAFORM_VERSION="1.3.0"
        echo "Downloading Terraform version ${TERRAFORM_VERSION}..."
        curl -fsSL "https://releases.hashicorp.com/terraform/${TERRAFORM_VERSION}/terraform_${TERRAFORM_VERSION}_linux_amd64.zip" -o terraform.zip
        unzip terraform.zip
        sudo mv terraform /usr/local/bin/
        rm terraform.zip
        terraform version
        az --version  # Verify Azure CLI installation
      displayName: 'Install Terraform'
      workingDirectory: $(System.DefaultWorkingDirectory)
```

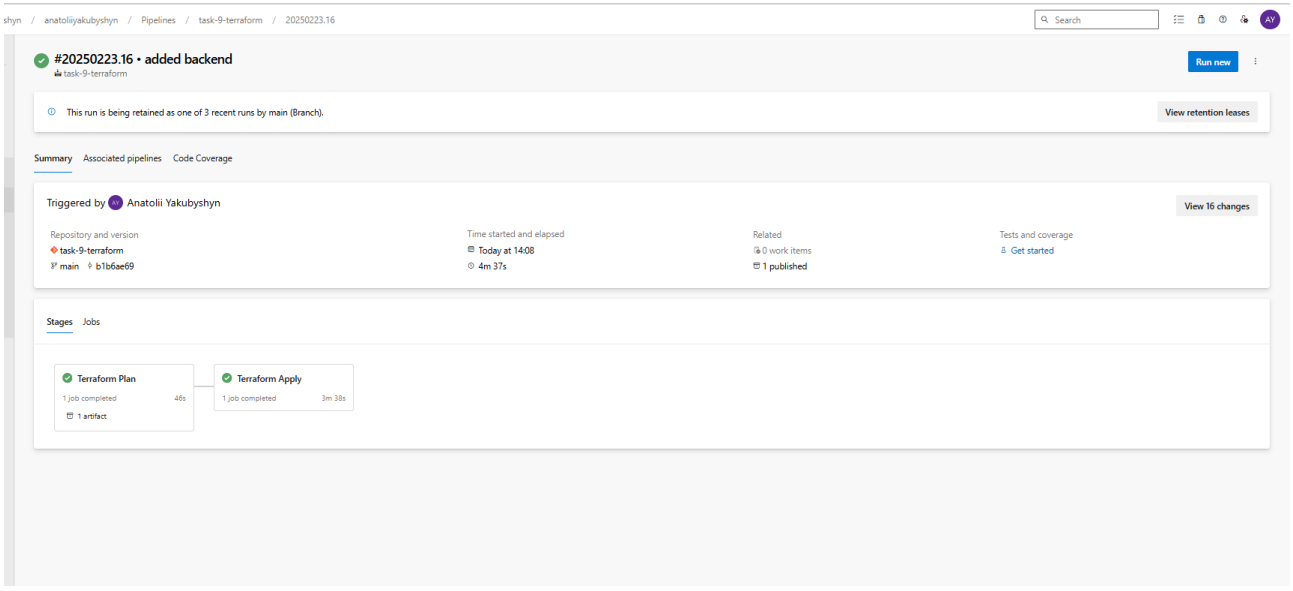


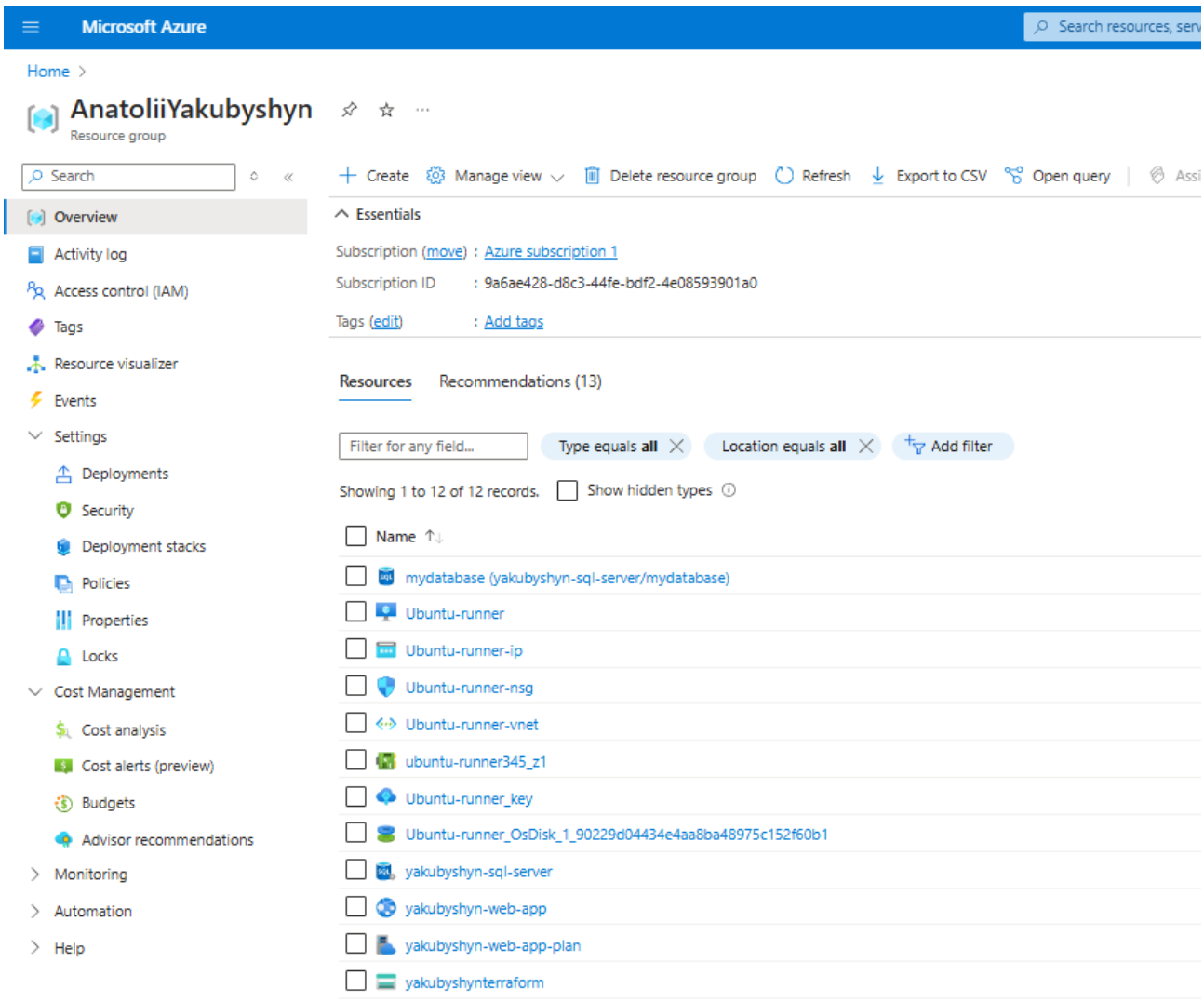
```
    workingDirectory: $(System.DefaultWorkingDirectory)

    # Run Terraform plan and output the plan file
    - script: |
        terraform plan -out=tfplan \
        -var="admin_user=$(admin_user)" \
        -var="admin_password=$(admin_password)" \
        -var="client_id=$(ARM_CLIENT_ID)" \
        -var="client_secret=$(ARM_CLIENT_SECRET)" \
        -var="subscription_id=$(ARM_SUBSCRIPTION_ID)" \
        -var="tenant_id=$(ARM_TENANT_ID)" \
        -var="location=$(LOCATION)" \
        -var="sqlAdminUser=$(ADMIN_USERNAME)" \
        -var="sqlAdminPassword=$(ADMIN_PASSWORD)"

      displayName: 'Terraform Plan'
      workingDirectory: $(System.DefaultWorkingDirectory)

    # List files to verify the tfplan exists
    - script: |
        ls -al $(System.DefaultWorkingDirectory)
      displayName: 'Verify tfplan exists'
      workingDirectory: $(System.DefaultWorkingDirectory)

    # Publish the tfplan artifact for use in the Deploy stage
    - publish: $(System.DefaultWorkingDirectory)/tfplan
      artifact: tfplan
      displayName: 'Publish tfplan artifact'

- stage: Deploy
  displayName: 'Terraform Apply'
  dependsOn: Plan
  condition: succeeded()
  jobs:
  - job: TerraformApply
    displayName: 'Terraform Apply'
    steps:
    # Install unzip on the deploy agent
    - script: |
        sudo apt-get update && sudo apt-get install -y unzip
      displayName: 'Install unzip'
      workingDirectory: $(System.DefaultWorkingDirectory)

    # Re-install Terraform on the deploy agent
    - script: |
        # Install Azure CLI
        echo "Installing Azure CLI..."
        curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash

        # Install Terraform
        TERRAFORM_VERSION="1.3.0"
        echo "Downloading Terraform version ${TERRAFORM_VERSION}..."
        curl -fsSL "https://releases.hashicorp.com/terraform/${TERRAFORM_VERSION}/terraform_${TERRAFORM_VERSION}_linux_amd64.zip" -o terraform.zip
        unzip terraform.zip
        sudo mv terraform /usr/local/bin/
        rm terraform.zip
        terraform version
        az --version  # Verify Azure CLI installation
      displayName: 'Install Terraform'
      workingDirectory: $(System.DefaultWorkingDirectory)
```

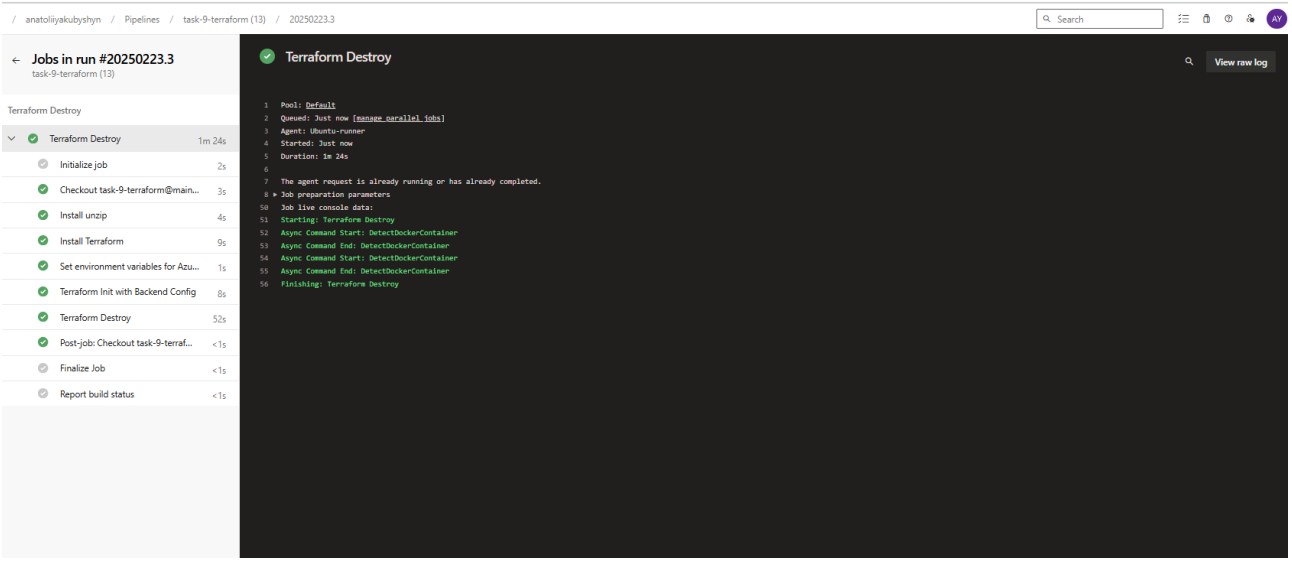

Welcome
main.tf
backend.tf
deploy.yml
destroy.yml
destroy.yml > [ ] stages > { } 0 > [ ] jobs > { } 0 > [ ] steps > { } 4 > script
trigger: none # Disable automatic triggering
pr: none # Disable PR triggers if you want manual-only execution
pool:
name: default
demands:
- agent.name -equals Ubuntu-runner
variables:
- group: "New variable group 13-Feb" # Must include variable: storageAccountName and other secrets
stages:
- stage: Destroy
displayName: 'Terraform Destroy'
jobs:
- job: TerraformDestroy
displayName: 'Terraform Destroy'
steps:
Install unzip on the deploy agent
- script: |
sudo apt-get update && sudo apt-get install -y unzip
displayName: 'Install unzip'
workingDirectory: $(System.DefaultWorkingDirectory)
Re-install Terraform on the deploy agent
- script: |
Install Azure CLI
echo "Installing Azure CLI..."
curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash
# Install Terraform
TERRAFORM_VERSION="1.3.0"
echo "Downloading Terraform version ${TERRAFORM_VERSION}..."
curl -fsSL "https://releases.hashicorp.com/terraform/${TERRAFORM_VERSION}/terraform_${TERRAFORM_VERSION}_linux_amd64.zip" -o terraform.zip
unzip terraform.zip
sudo mv terraform /usr/local/bin/
rm terraform.zip
terraform version
az --version # Verify Azure CLI installation
displayName: 'Install Terraform'
workingDirectory: $(System.DefaultWorkingDirectory)
Set environment variables for Azure SP authentication
- script: |
echo "##vso[task.setvariable variable=ARM_CLIENT_ID]$(client_id)"
echo "##vso[task.setvariable variable=ARM_CLIENT_SECRET]$(client_secret)"
echo "##vso[task.setvariable variable=ARM_SUBSCRIPTION_ID]$(subscription_id)"
echo "##vso[task.setvariable variable=ARM_TENANT_ID]$(tenant_id)"
echo "##vso[task.setvariable variable=LOCATION]$(location)"
echo "##vso[task.setvariable variable=ADMIN_USERNAME]$(adminUsername)"
echo "##vso[task.setvariable variable=ADMIN_PASSWORD]$(adminPassword)"
displayName: 'Set environment variables for Azure Service Principal'
Reinitialize Terraform with the same backend override
- script: |
terraform init -reconfigure
displayName: 'Terraform Init with Backend Config'
workingDirectory: $(System.DefaultWorkingDirectory)
Destroy
- script: |
terraform destroy -auto-approve \
- script: |
terraform plan -out=tfplan \

```
    echo "##vso[task.setvariable variable=ARM_CLIENT_ID]$(client_id)"
    echo "##vso[task.setvariable variable=ARM_CLIENT_SECRET]$(client_secret)"
    echo "##vso[task.setvariable variable=ARM_SUBSCRIPTION_ID]$(subscription_id)"
    echo "##vso[task.setvariable variable=ARM_TENANT_ID]$(tenant_id)"
    echo "##vso[task.setvariable variable=LOCATION]$(location)"
    echo "##vso[task.setvariable variable=ADMIN_USERNAME]$(adminUsername)"
    echo "##vso[task.setvariable variable=ADMIN_PASSWORD]$(adminPassword)"
  displayName: 'Set environment variables for Azure Service Principal'

# Reinitialize Terraform with the same backend override
- script: |
    terraform init -reconfigure
  displayName: 'Terraform Init with Backend Config'
  workingDirectory: $(System.DefaultWorkingDirectory)

# Destroy
- script: |
    terraform destroy -auto-approve \
    -var="admin_user=$(admin_user)" \
    -var="admin_password=$(admin_password)" \
    -var="client_id=$(ARM_CLIENT_ID)" \
    -var="client_secret=$(ARM_CLIENT_SECRET)" \
    -var="subscription_id=$(ARM_SUBSCRIPTION_ID)" \
    -var="tenant_id=$(ARM_TENANT_ID)" \
    -var="location=$(LOCATION)" \
    -var="sqlAdminUser=$(ADMIN_USERNAME)" \
    -var="sqlAdminPassword=$(ADMIN_PASSWORD)"
  displayName: 'Terraform Destroy'
  workingDirectory: $(System.DefaultWorkingDirectory)
```

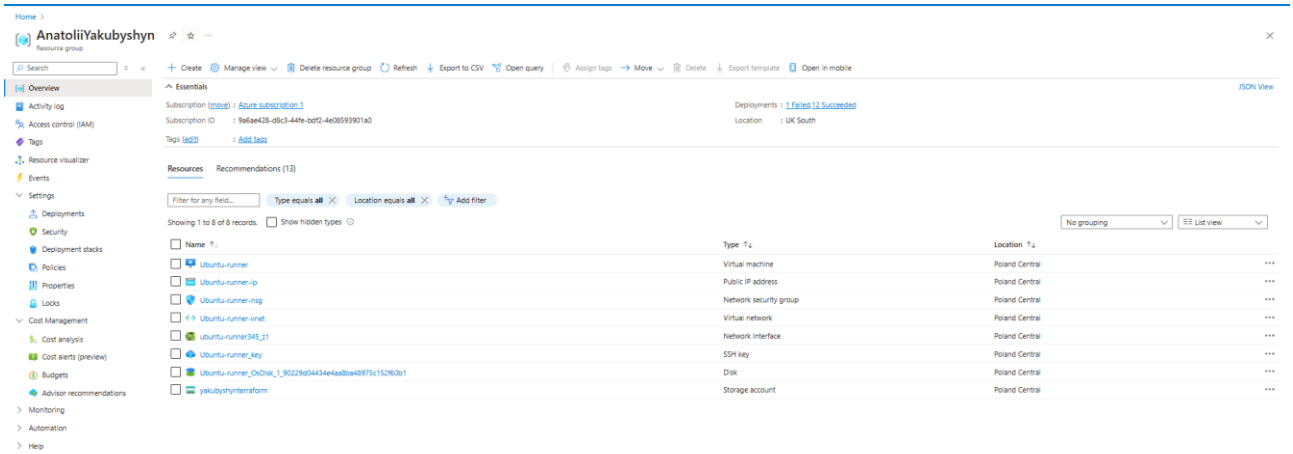
4. Run the Pipeline: a. Save and run the pipeline to provision the infrastructure in Azure. b. Verify that the resources are created as expected.




Microsoft Azure
Search resources, serv
Home
AnatoliiYakubyshyn
Resource group
Search
Overview
Activity log
Access control (IAM)
Tags
Resource visualizer
Events
Settings
Deployments
Security
Deployment stacks
Policies
Properties
Locks
Cost Management
Cost analysis
Cost alerts (preview)
Budgets
Advisor recommendations
Monitoring
Automation
Help
Create
Manage view
Delete resource group
Refresh
Export to CSV
Open query
Essentials
Subscription (move) : Azure subscription 1
Subscription ID : 9a6ae428-d8c3-44fe-bdf2-4e08593901a0
Tags (edit) : Add tags
Resources
Recommendations (13)
Filter for any field...
Type equals all
Location equals all
Add filter
Showing 1 to 12 of 12 records.
Show hidden types
Name
mydatabase (yakubyshyn-sql-server/mydatabase)
Ubuntu-runner
Ubuntu-runner-ip
Ubuntu-runner-nsg
Ubuntu-runner-vnet
ubuntu-runner345_z1
Ubuntu-runner_key
Ubuntu-runner_OsDisk_1_90229d04434e4aa8ba48975c152f60b1
yakubyshyn-sql-server
yakubyshyn-web-app
yakubyshyn-web-app-plan
yakubyshynterraform



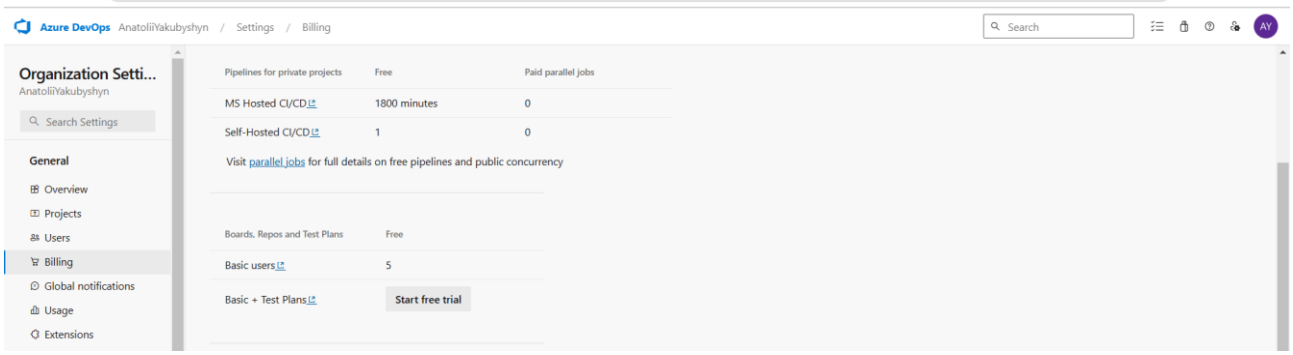

```
Terraform used the selected providers to generate the following execution
plan. Resource actions are indicated with the following symbols:
  - destroy

Terraform will perform the following actions:

  # azurerm_app_service_plan.app_service_plan will be destroyed
  - resource "azurerm_app_service_plan" "app_service_plan" {
      - id                           = "/subscriptions/9a6ae428-d8c3-44fe-bdf2-4e08593901a0/resourceGroups/AnatoliiYakubyshyn/providers/Microsoft.Web/serverFarms/yakubyshyn-web-app-plan" -> null
      - is_xenon                     = false -> null
      - kind                         = "Windows" -> null
      - location                     = "uksouth" -> null
      - maximum_elastic_worker_count = 1 -> null
      - maximum_number_of_workers    = 3 -> null
      - name                         = "yakubyshyn-web-app-plan" -> null
      - per_site_scaling             = false -> null
      - reserved                     = false -> null
      - resource_group_name          = "AnatoliiYakubyshyn" -> null
      - tags                         = {} -> null
      - zone_redundant               = false -> null

      - sku {
          - capacity = 1 -> null
          - size     = "B1" -> null
          - tier     = "Basic" -> null
        }
    }

  # azurerm_linux_web_app.web_app will be destroyed
  - resource "azurerm_linux_web_app" "web_app" {
      - app_settings                                 = {} -> null
      - client_affinity_enabled                      = false -> null
      - client_certificate_enabled                   = false -> null
      - client_certificate_mode                      = "Required" -> null
      - custom_domain_verification_id                = (sensitive value)
      - default_hostname                             = "yakubyshyn-web-app.azurewebsites.net" -> null
      - enabled                                      = true -> null
      - ftp_publish_basic_authentication_enabled     = true -> null
      - https_only                                   = false -> null
      - id                                           = "/subscriptions/9a6ae428-d8c3-44fe-bdf2-4e08593901a0/resourceGroups/AnatoliiYakubyshyn/providers/Microsoft.Web/sites/yakubyshyn-web-app" -> null
      - key_vault_reference_identity_id              = "SystemAssigned" -> null
      - kind                                         = "app" -> null
      - location                                     = "uksouth" -> null
      - name                                         = "yakubyshyn-web-app" -> null
      - outbound_ip_address_list                     = [
          - "20.108.102.154",
```

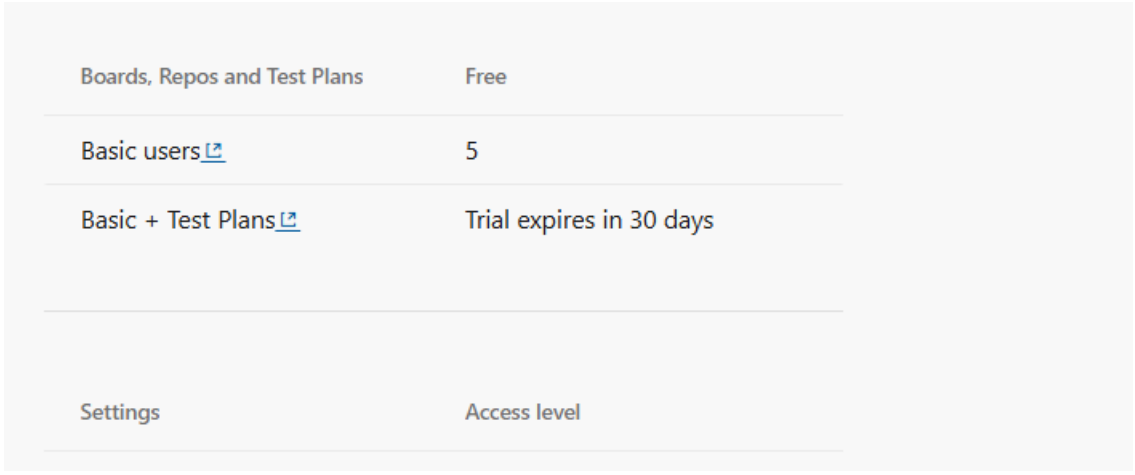


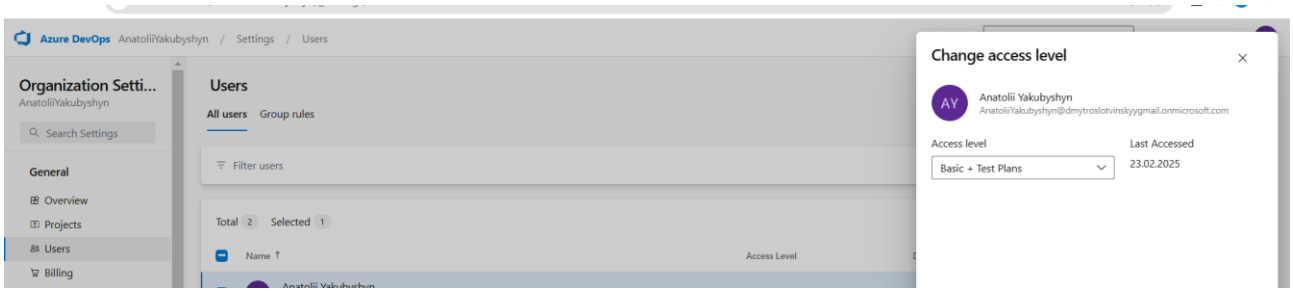
Practical Task 10: Integrate Azure Test Plans with CI/CD Pipelines

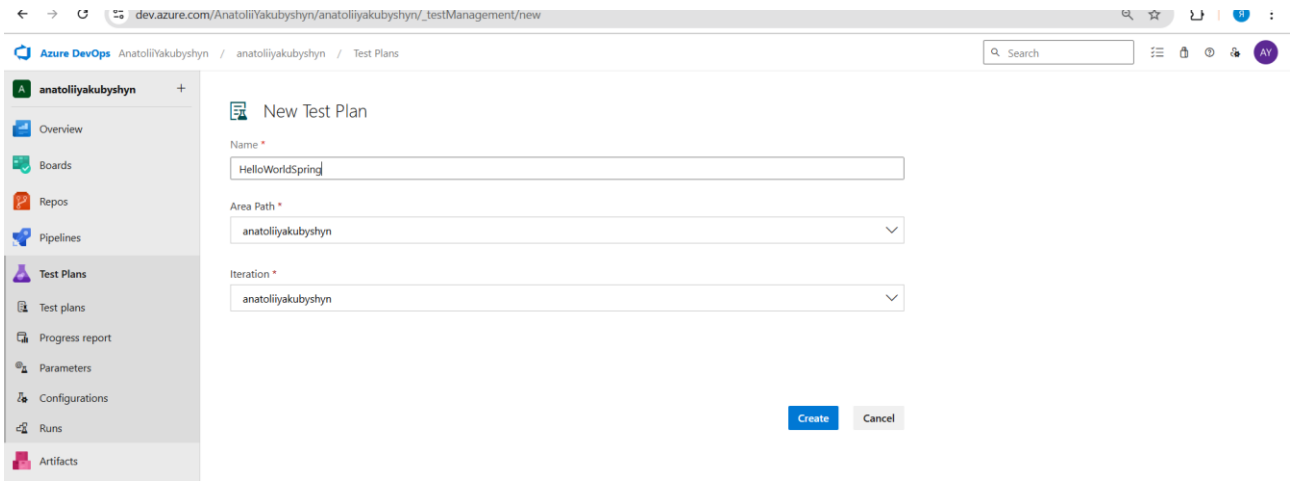
Requirements: 1. Create a Test Plan:

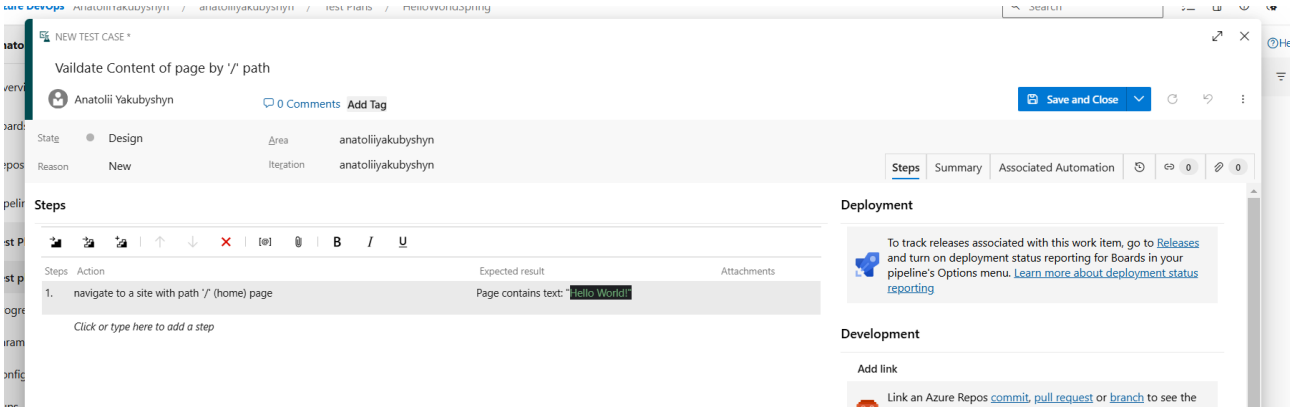
a. Navigate to Azure Test Plans and create a new test plan that includes both manual and automated tests for your application.

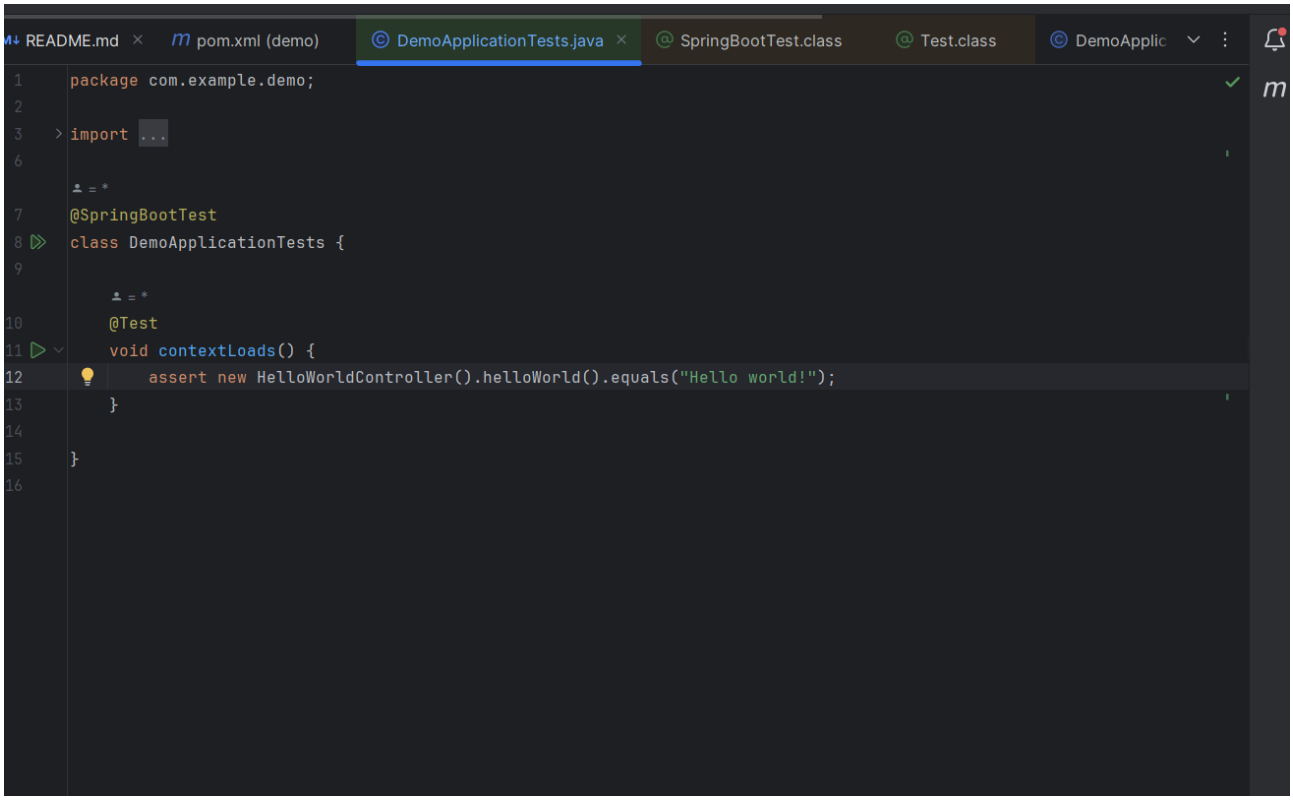








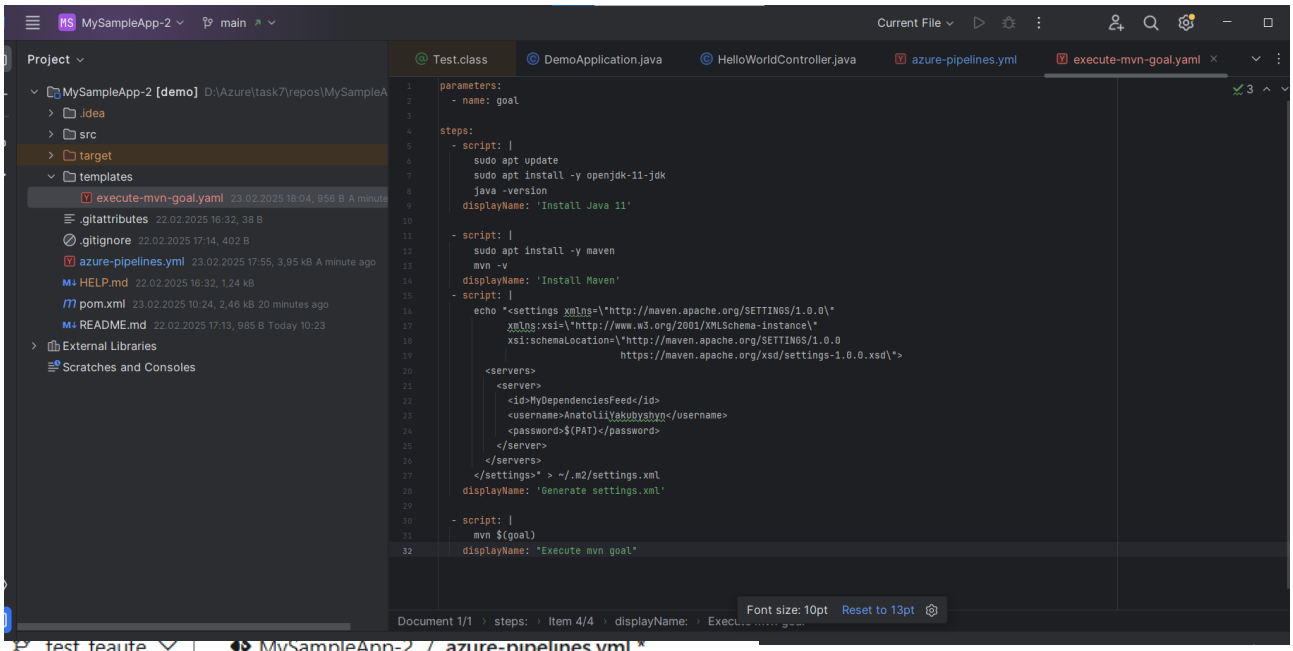


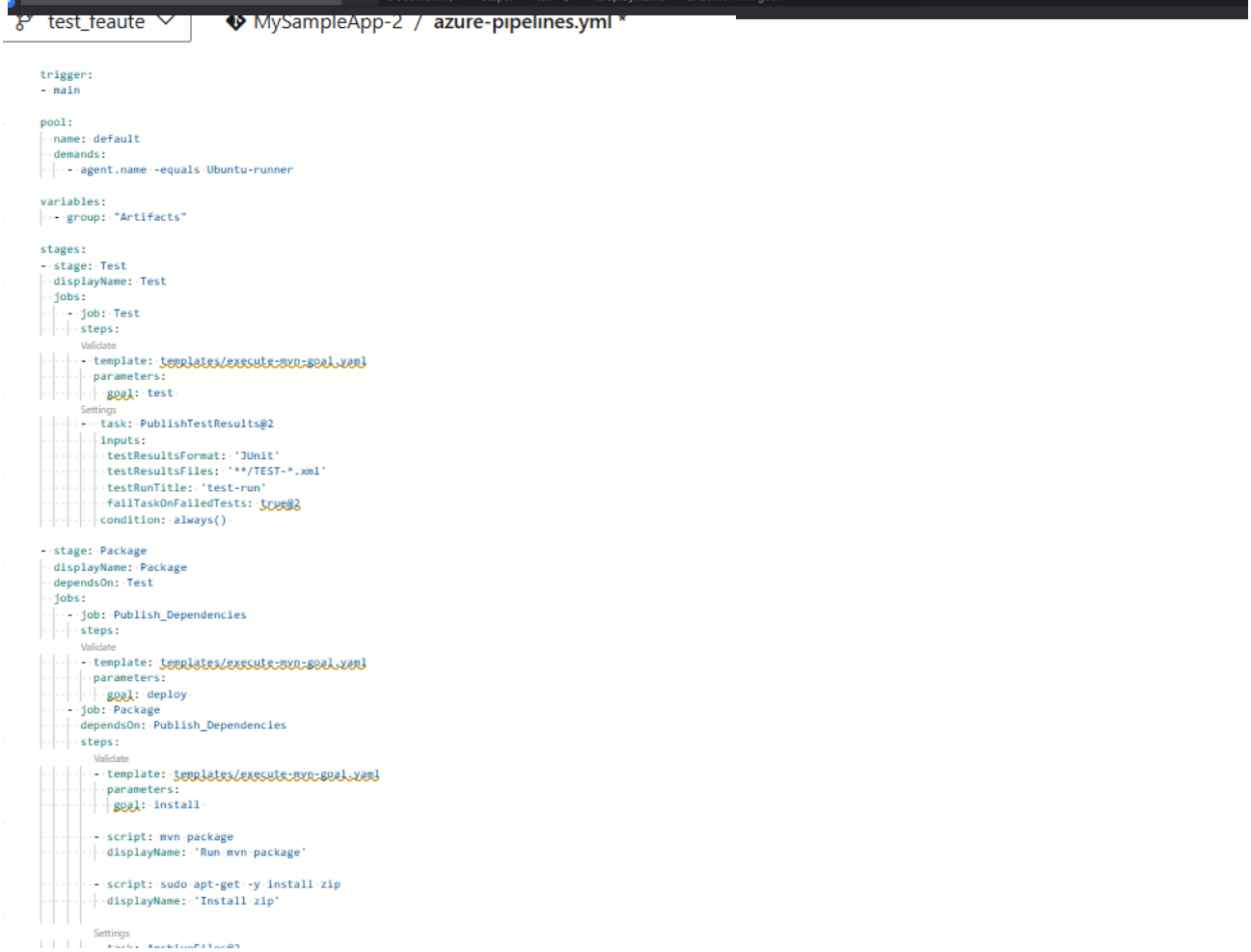


created an automated test

2. Modify the Build Pipeline:

a. Update the build pipeline to include a testing stage that runs automated tests using a testing framework (e.g., NUnit for .NET or Jest for Node.js)





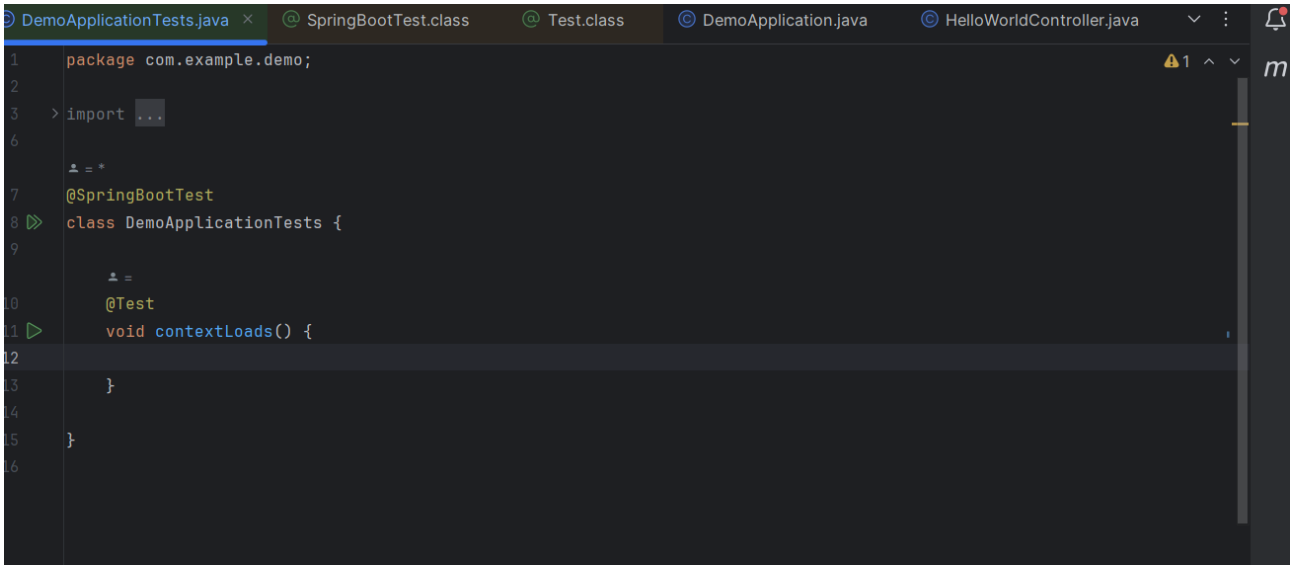
```yaml
stages:
- stage: Test
  displayName: Test
  jobs:
    - job: Test
      steps:
      - template: templates/execute-mvn-goal.yaml
        parameters:
          goal: test
      -  task: PublishTestResults@2
         inputs:
          testResultsFormat: 'JUnit'
          testResultsFiles: '**/TEST-*.xml'
          testRunTitle: 'test-run'
          testRunSystem: 'AzureTestPlans'
         condition: always()
```

Added test Run System

3. Configure Release Pipeline:

a. Modify the release pipeline to deploy the application only if all tests pass.

b. Use conditions in the release pipeline to check for successful test results.

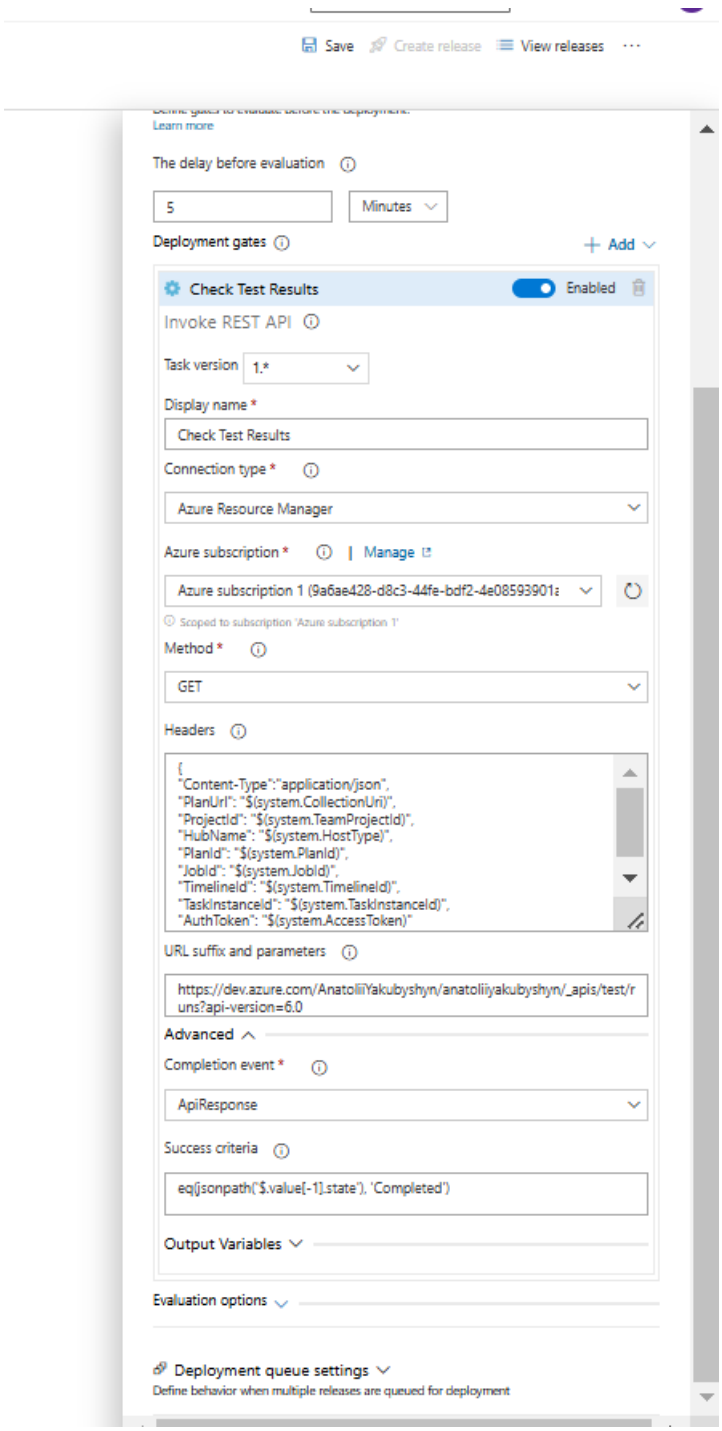


Changed back to default test

3. Configure Release Pipeline:

a. Modify the release pipeline to deploy the application only if all tests pass.

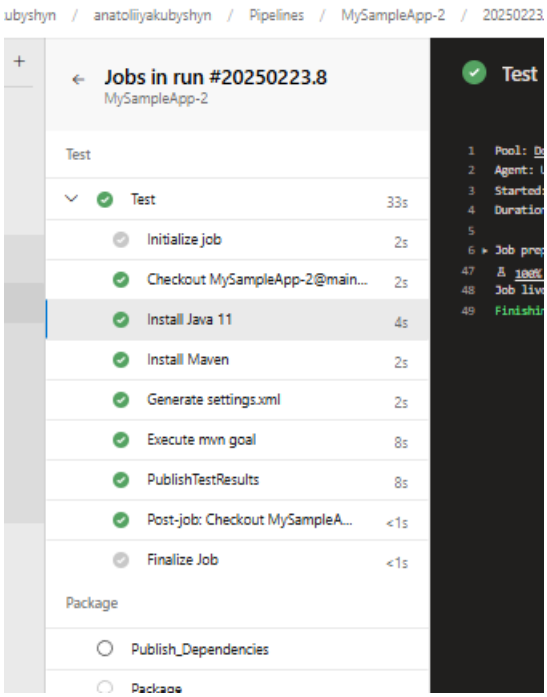
b. Use conditions in the release pipeline to check for successful test results.



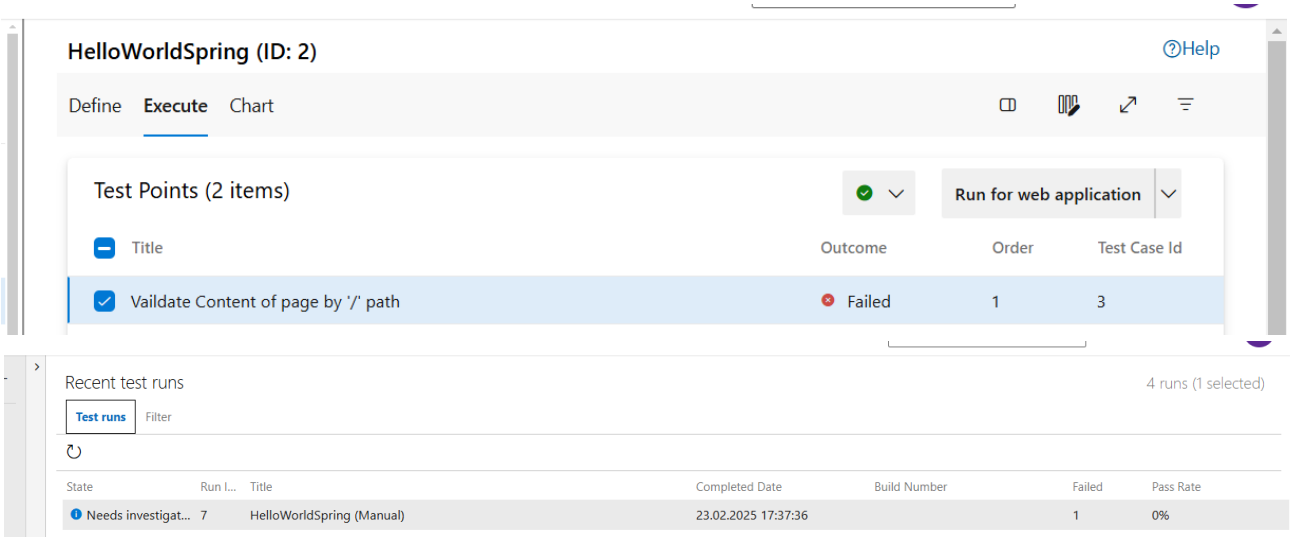
Set trigger gate

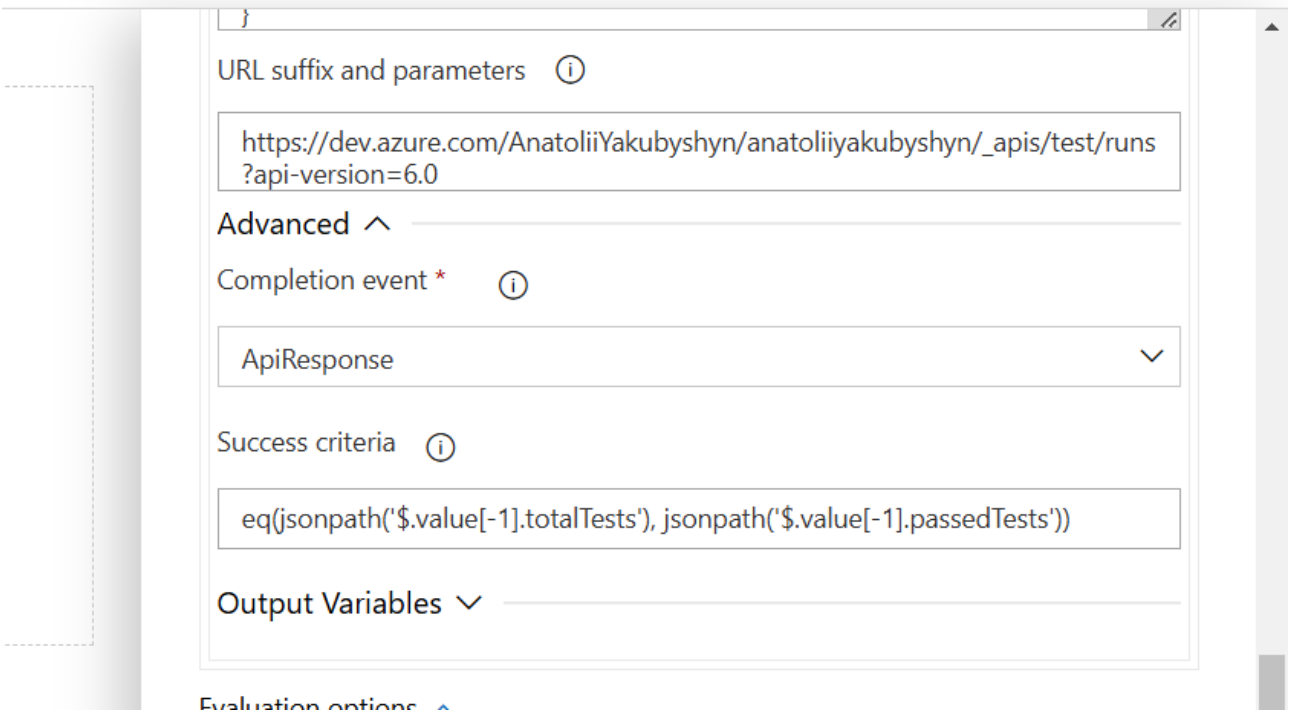
4. Run the Pipeline:

a. Trigger the pipeline and verify that it runs the tests and reports the results back to Azure Test Plans.

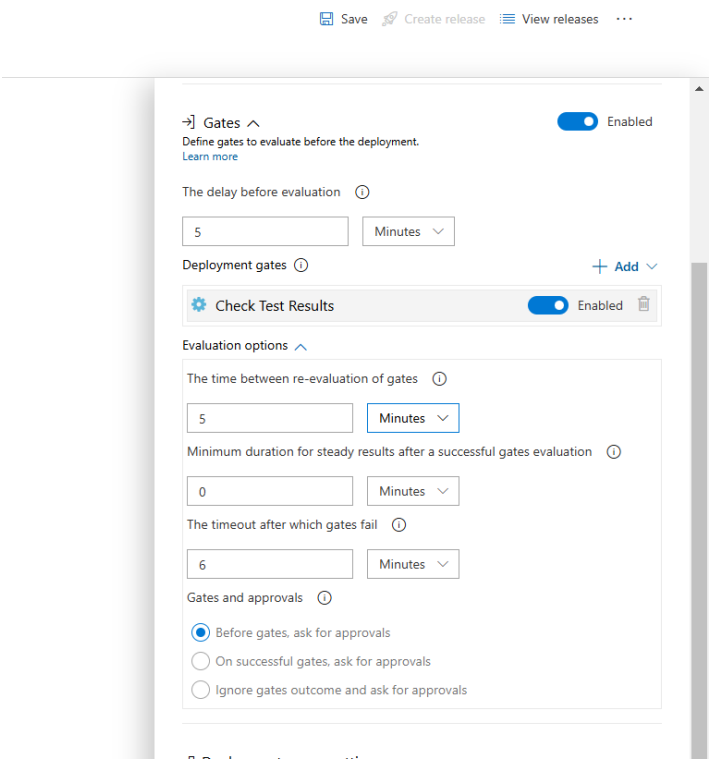


b. Ensure that any failed tests block the deployment process.



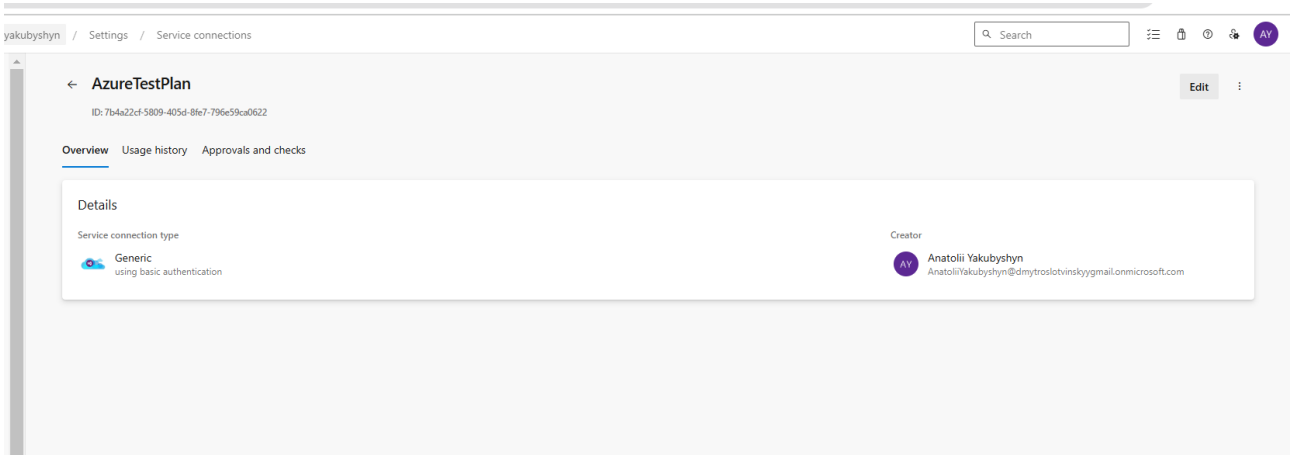


Updated deployment gate

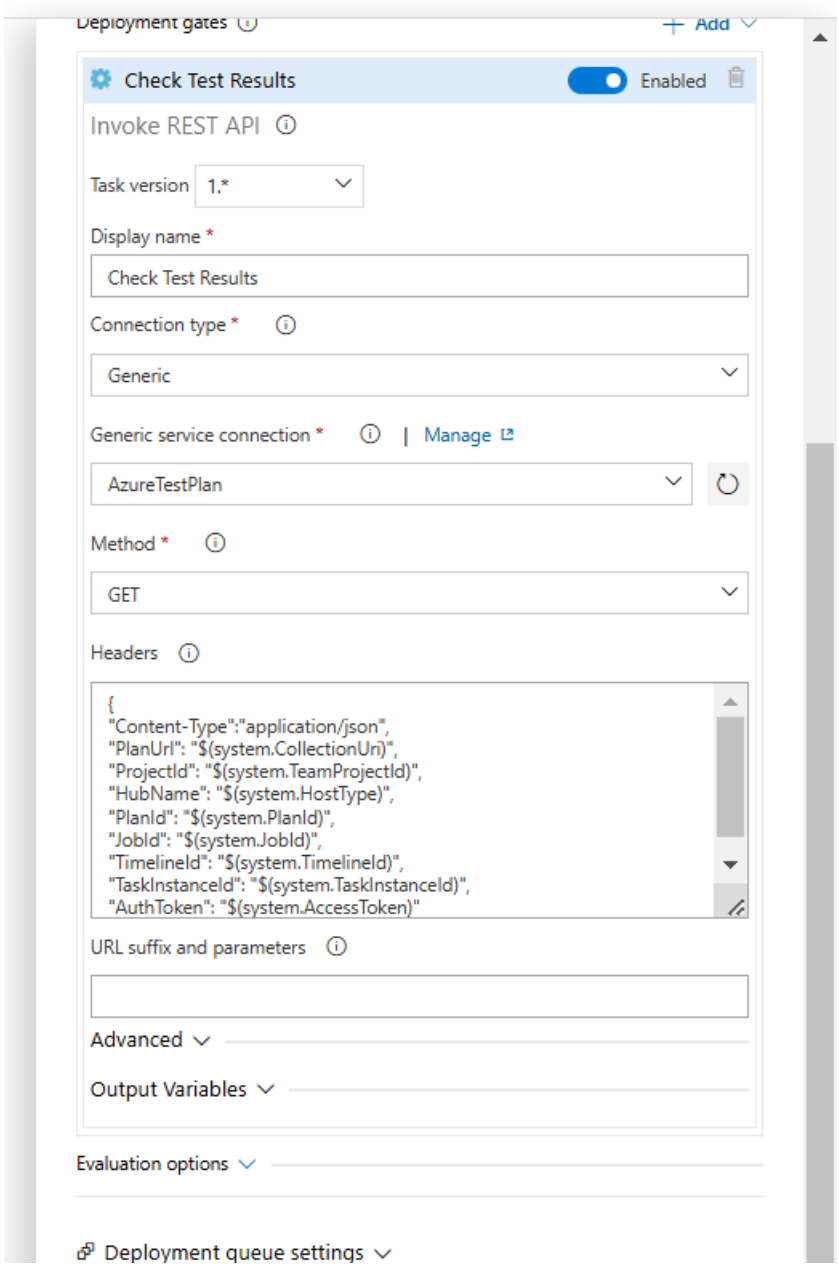


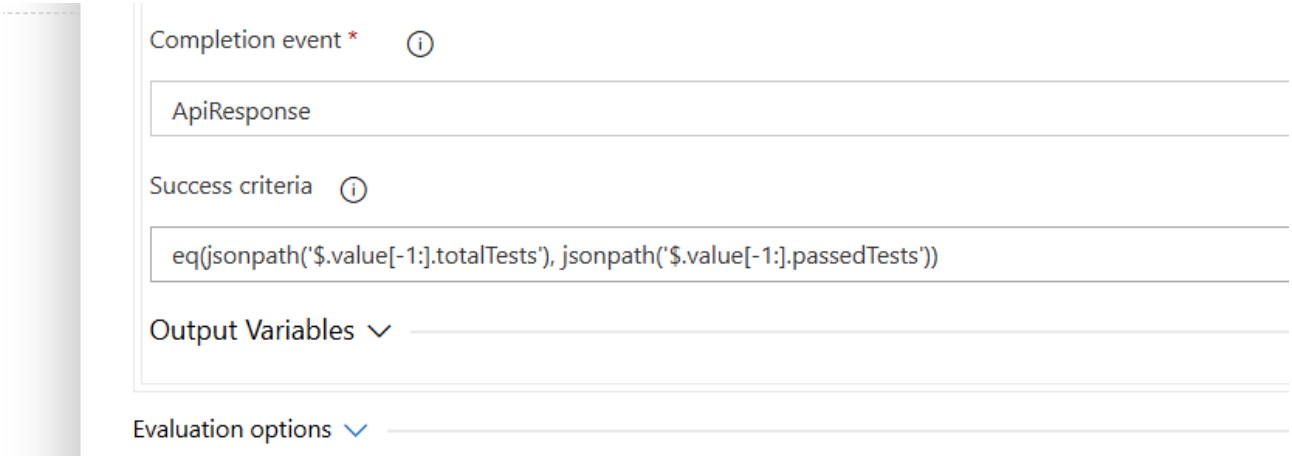
Updated eval options

Then -> create release

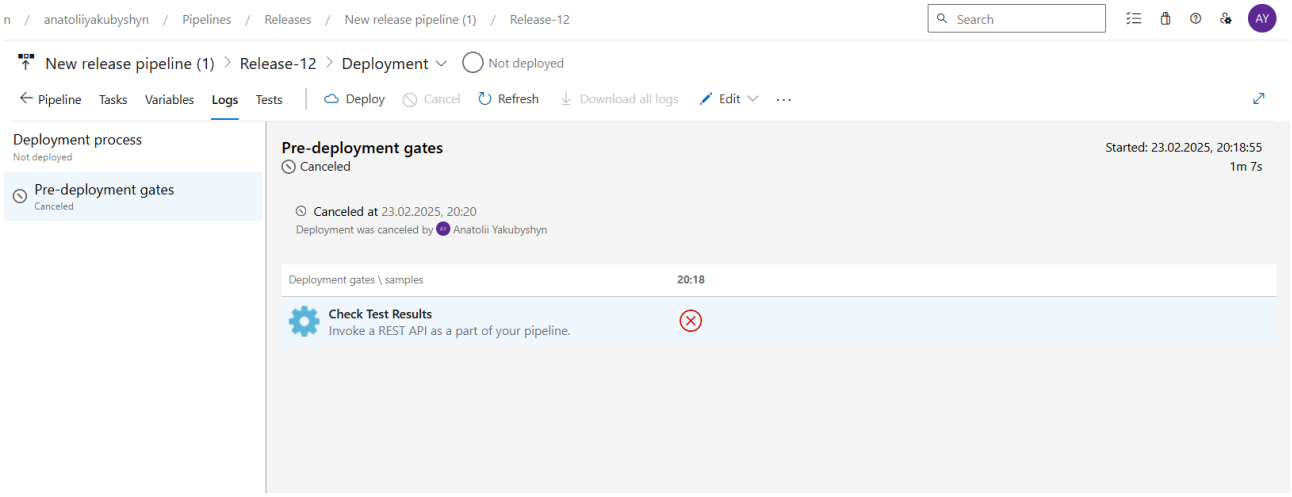


added url in generic service connection

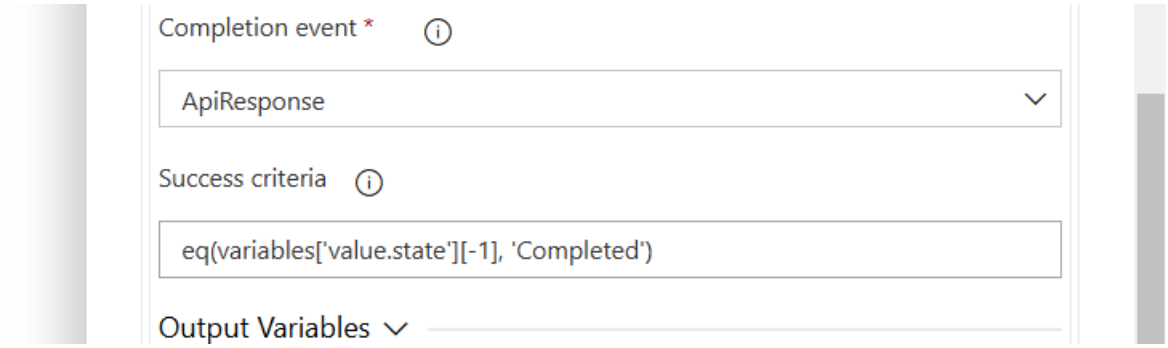




Check if all test passed in the last test run

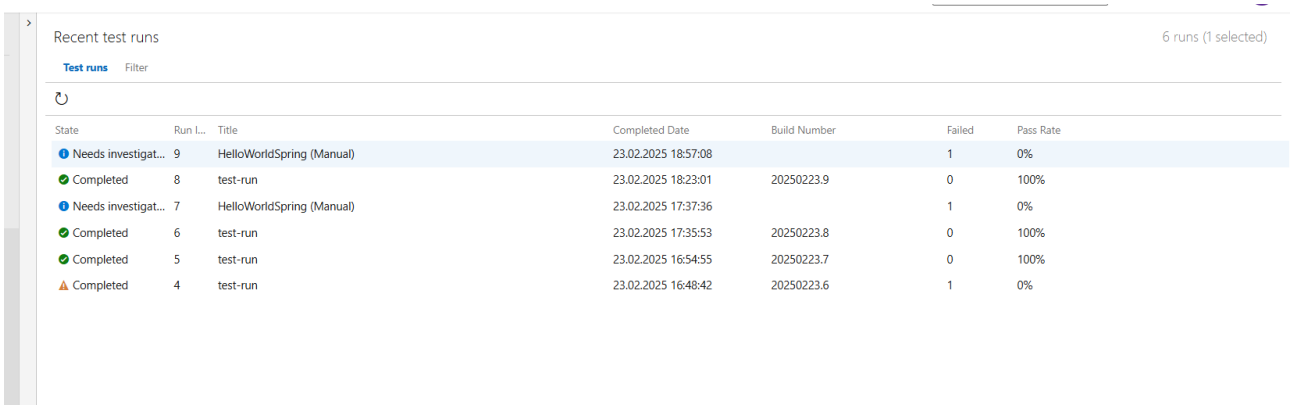


Blocks the deployment process



Finally fixed success criteria

If really all tests pass



Check again