**CI/CD FOR FLUTTER APP USING AZURE DEVOPS TO MICROSOFT APP CENTER**

**What is Azure DevOps?**

Azure DevOps -- rebranded from Visual Studio Team Services (VSTS) in 2018 -- is a software as a service (SaaS) platform from Microsoft designed to provide a comprehensive toolchain for developing and deploying software projects.

The platform assists development teams and offers special tools and services for software programmers, analysts, and testers as well as IT projects or team managers. Azure DevOps integrates with a wide range of other tools, expanding DevOps toolsets and tailoring them to the unique needs of the development team and organization.

Azure DevOps is an end-to-end software development platform that offers an assortment of capabilities intended to organize and accelerate development efforts across the entire application lifecycle:

**Requirements management.**

* Project management for both Agile software development and waterfall teams.
* Version control using Team Foundation Version Control (TFVC) or Git.
* Automated buildings.
* Reporting, such as test results, and development metrics, such as backlogs and release velocity.
* Testing and release management.

**What are Azure Pipelines?**

Azure Pipelines for delivery, integration, and deployment management. Azure Pipelines can automatically build and test code in major languages, including Node.js, Python, Ruby, Java, PHP, C#, C++, Go, Xcode, .NET, Android, and iOS. It handles various project types and combines continuous integration, continuous testing, and continuous delivery to build, test and deliver code to a deployment target. It can accommodate up to 10 parallel jobs in any open-source project. Azure Pipelines also offers file transforms and variable substitutions that are available for web app files as well as Extensible Markup Language and JavaScript Object Notation files. Azure Pipelines integrates with Git and Azure Repos for version control.

**What are Azure Repos?**

Azure Repos for version control. Azure Repos is a set of version control tools used to manage and track changes to code that occur over time. Version control offers a powerful way to save work and coordinate code changes across an entire team. Azure Repos provides a private Git repository or TFVC that offers functionality like collaborative pull requests and advanced file management.

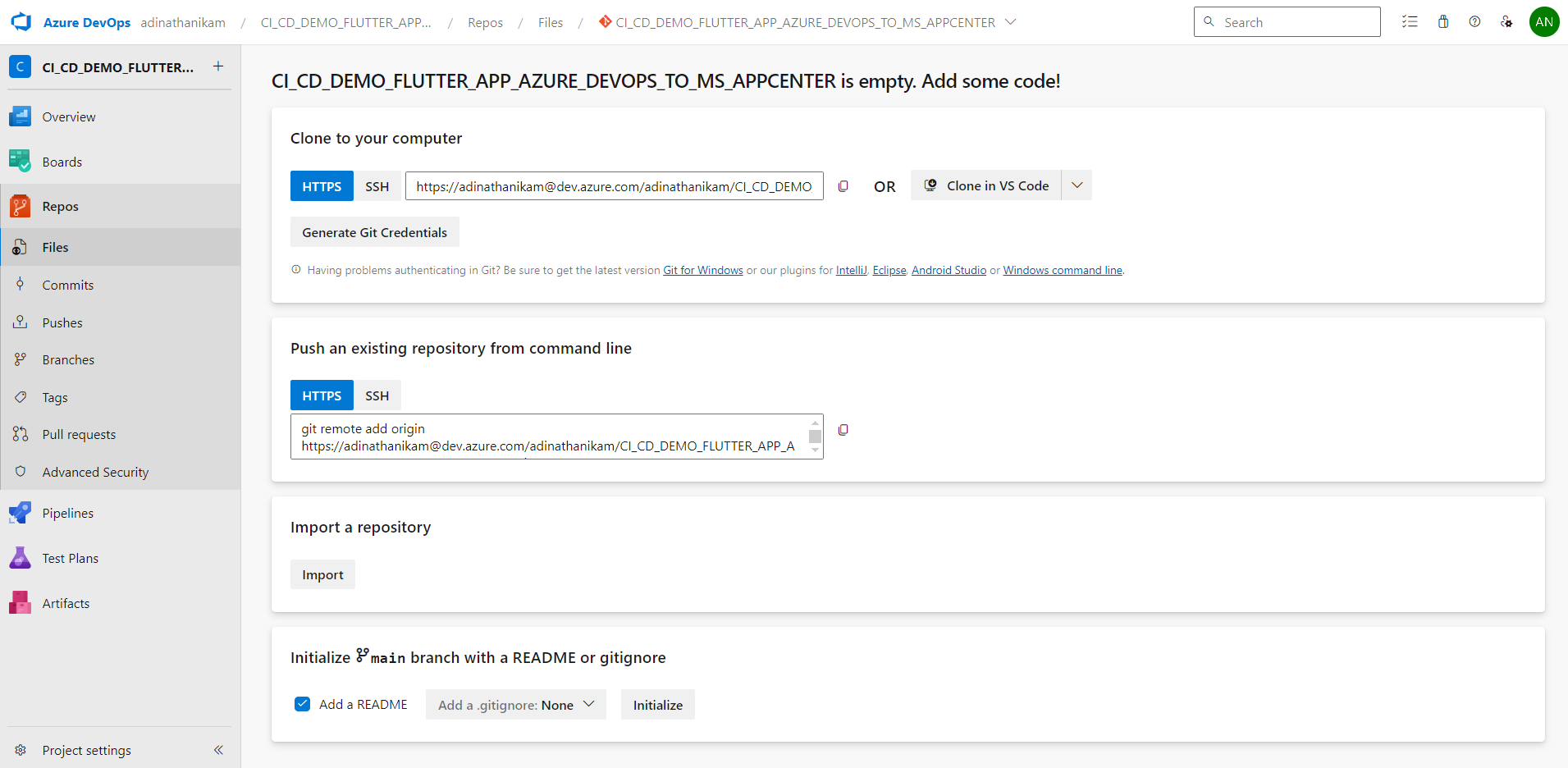
**Demo:**

**Step-1:**

* **I’ll Assume you have created account for both Azure DevOps and Microsoft App Center.**
* Create a Project in Azure DevOps, I’ll Name it **CI\_CD\_DEMO\_FLUTTER\_APP\_AZURE\_DEVOPS\_TO\_MS\_APP\_CENTER.**

A screenshot of a computer screen

Description automatically generated



**Step-2:**

* Create a Sample Demo Flutter Application in Android Studio and Push it to Repo in Azure DevOps.

A screen shot of a phone

Description automatically generated

**C:\SG\Learn\CICD\ci\_cd\_demo\_flutter\_app\_azure\_devops\_to\_ms\_app\_center>** git init

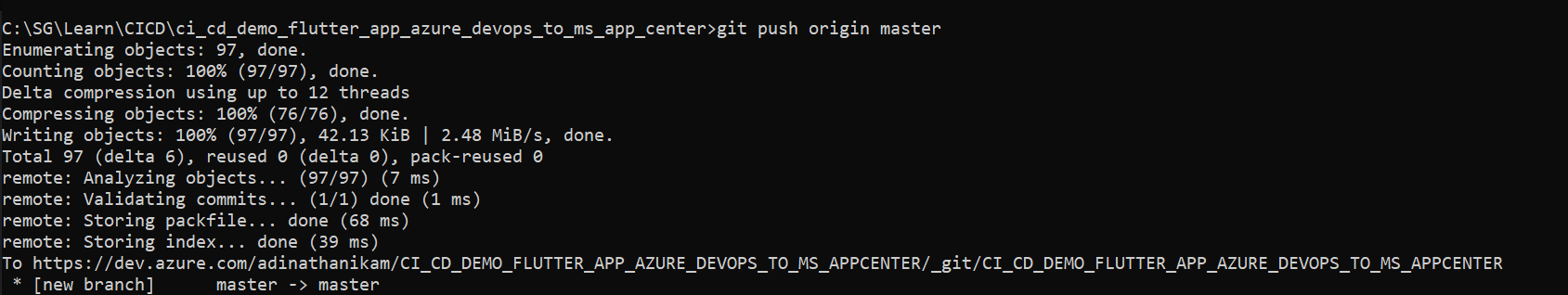
**C:\SG\Learn\ci\_cd\_demo\_flutter\_app>** git add .

**C:\SG\Learn\ci\_cd\_demo\_flutter\_app>** git commit -m "init project"

**C:\SG\Learn\ci\_cd\_demo\_flutter\_app>** git remote add origin

<https://adinathanikam@dev.azure.com/adinathanikam/CI_CD_DEMO_FLUTTER_APP_AZURE_DEVOPS_TO_MS_APPCENTER/_git/CI_CD_DEMO_FLUTTER_APP_AZURE_DEVOPS_TO_MS_APPCENTER>

**C:\SG\Learn\ci\_cd\_demo\_flutter\_app>** git push -u origin master



A screenshot of a computer

Description automatically generated

**Step-3:** Then we need to install pre-built azure DevOps flutter-tasks library which will help us to reduce the boiler-plate code in our CI/CD workflow file.

**To install flutter-tasks navigate to your organization projects and there up above to right of top bar there’ll be icon called Marketplace -> Browse Marketplace.**

**A screenshot of a computer

Description automatically generated**

**Search for “Flutter Tasks” library from “Hey24Sheep” and click on “Get it Free” it’ll get installed.**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Step-4:** Then Create file in root directory of flutter project with name -> **azure-pipelines.yml**A screenshot of a computer

Description automatically generated

**Step-5:** Now we’ll start writing the azure-pipelines.yml script which will contain the list of tasks which will define what all CI/CD actions that need to be taken.

**.ci\_cd\_demo\_flutter\_app\_azure\_devops\_to\_ms\_app\_center /azure-pipelines.yml**

**# Code Starts from here**

trigger:

- master

pool:

vmImage: ubuntu-latest

steps:

- task: FlutterInstall@0

inputs:

customversion: '3.10.6'

channel: 'stable'

- script: flutter pub get

displayName: 'Flutter pub get'

- script: flutter test

displayName: 'Flutter test'

- script: flutter build apk --release

displayName: 'Flutter build APK'

- task: CopyFiles@2

inputs:

SourceFolder: '$(build.sourcesdirectory)/build/app/outputs/flutter-apk'

Contents: '\*\*/\*.apk'

TargetFolder: '$(build.artifactstagingdirectory)'

- task: PublishBuildArtifacts@1

inputs:

PathtoPublish: '$(build.artifactstagingdirectory)'

ArtifactName: 'drop'

publishLocation: 'Container'

- task: AppCenterDistribute@3

inputs:

serverEndpoint: 'Deploy Test'

appSlug: 'adinathanikam/test'

appFile: '$(build.artifactstagingdirectory)/\*\*/\*.apk'

releaseNotesOption: 'input'

releaseNotesInput: 'Release $(Build.BuildNumber)'

**# Code Ends Here.**

**A screen shot of a computer program

Description automatically generated**

**Breakdown of CI/CD Workflow File.**

1. **Trigger:**

This workflow is triggered by the push event on the master branch. This means that whenever a new commit is pushed to the master branch of the repository, this workflow will be executed.

**A black rectangular object with a black background

Description automatically generated**

1. **Pool:**

the pool keyword is used to define the execution environment for the pipeline. This line specifies that the pipeline will run on a virtual machine with the latest version of Ubuntu.

A black rectangular object with a black stripe

Description automatically generated

1. **Steps:**

This keyword indicates the beginning of the steps section, where individual tasks or actions to be performed in the pipeline are list.

A black rectangular object with a black border

Description automatically generated

1. **This step installs Flutter on the agent machine. It uses the FlutterInstall task with version "0" and specifies inputs such as the custom version of Flutter (3.10.6) and the channel (stable) to install.**

**A black screen with green text

Description automatically generated**

1. **This step runs the command flutter pub get, which fetches and installs the dependencies specified in the pubspec.yaml file of the Flutter project. The displayName specifies how this step will appear in the pipeline logs.**

**A black screen with green text

Description automatically generated**

1. **This step runs the command flutter test, which executes tests defined in the Flutter project. The displayName specifies how this step will appear in the pipeline logs.**

**A black rectangular object with a black stripe

Description automatically generated**

1. **This step builds the Flutter project into an APK file (\*.apk) for release using the command flutter build apk --release. The displayName specifies how this step will appear in the pipeline logs.**

**A screen shot of a computer

Description automatically generated**

1. **This step copies the APK files generated in the previous step from the specified source folder to the target folder. It uses the CopyFiles task with version "2".**

**A screenshot of a computer program

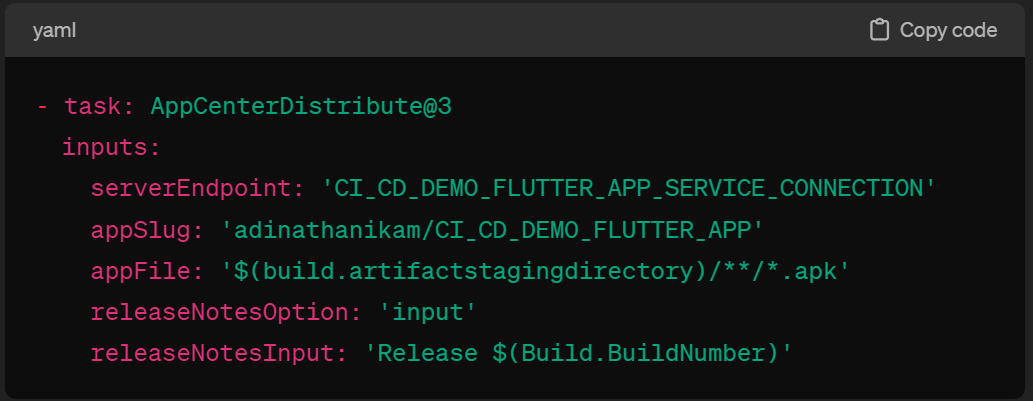
Description automatically generated**

1. **This step publishes build artifacts to a specified location. Here, it publishes the APK files from the staging directory to a location specified by publishLocation: 'Container'. The artifact name is set to "drop".**

**A screenshot of a computer

Description automatically generated**

1. **This step distributes the specified APK file(s) to the App Center for distribution to testers or users. It specifies inputs such as the server endpoint, the app slug (identifier), the path to the APK file(s) to distribute, and release notes for the distribution. It uses the AppCenterDistribute task with version "3".**

****

**Step-6:** Now we have to create a service connection between the azure DevOps and MS App centre so that this CI/CD flow can publish the generated builds in App Centre to do that head to App Centre -> Sign In -> create group -> and add a new app.

A screenshot of a computer

Description automatically generated

Then go to Settings -> App API Tokens -> New API Token (Give Full Access) -> Copy that token and keep it in some safe place as we need it later to establish service connection from Azure DevOps.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Step-7:** Then next move to Azure DevOps -> Project Settings -> Service Connections -> Create Service Connection -> Select “Visual Studio App Centre” then Paste the API token which we got from MS App Centre -> Give Service Connection Name -> Save.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Step-7:** Then we have to give Service Connection Name in Workflow File.

serverEndpoint: “**SERVICE\_CONNECTION\_NAME\_FROM\_AZURE\_DEVOPS**”

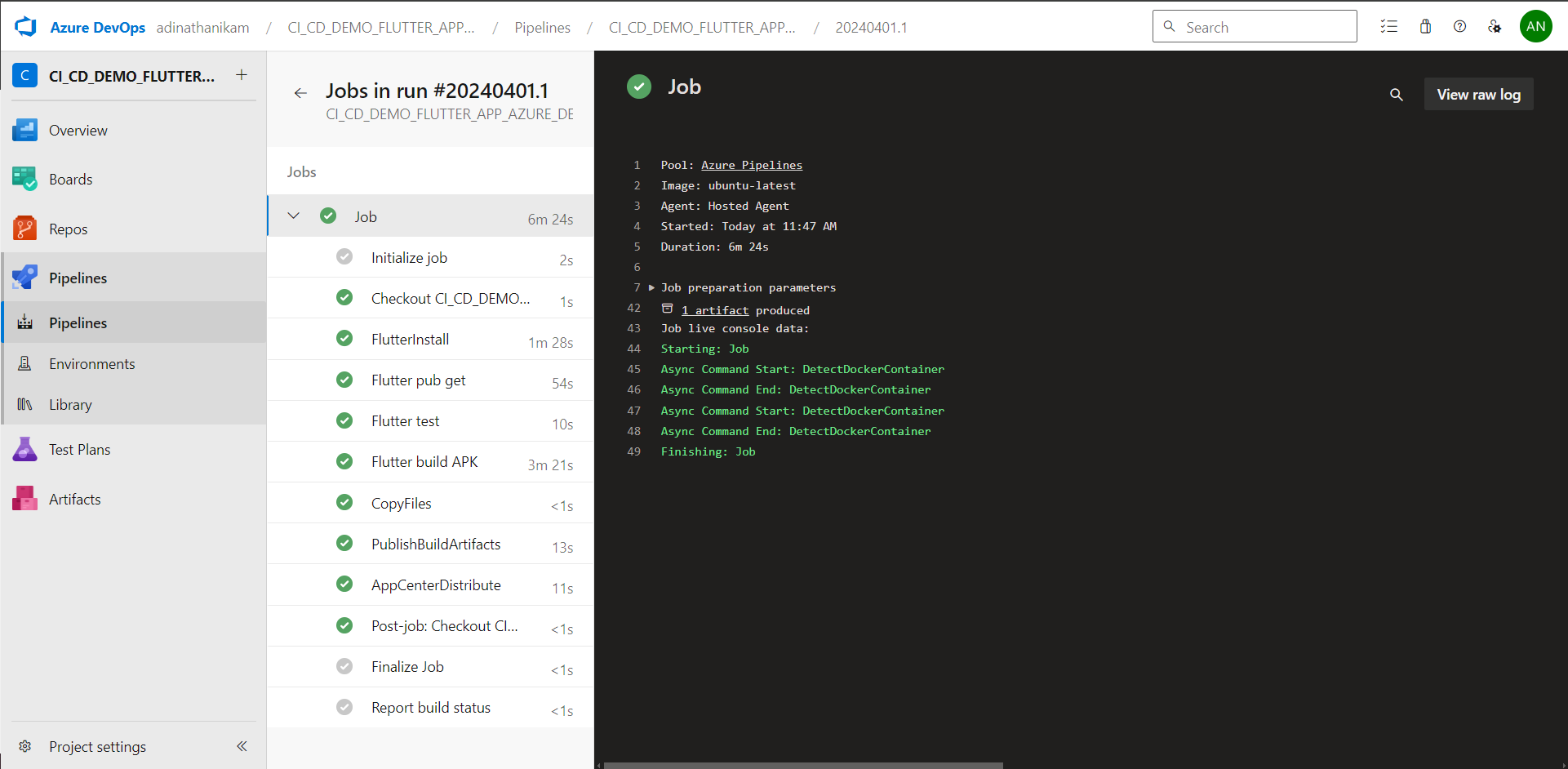
appSlug: “**APP\_CENTER\_ORGANIZATION\_NAME/APP\_CENTER\_APP\_NAME**”

A screenshot of a computer program

Description automatically generated

**Step-8:** Then we are ready to test this CI/CD Flow.

Just commit the code with this file to Azure DevOps to Azure Repo, it’ll trigger automatically and On Successful Completion of CI/CD you will be seeing the Release Available in App Centre.



A screenshot of a computer

Description automatically generated