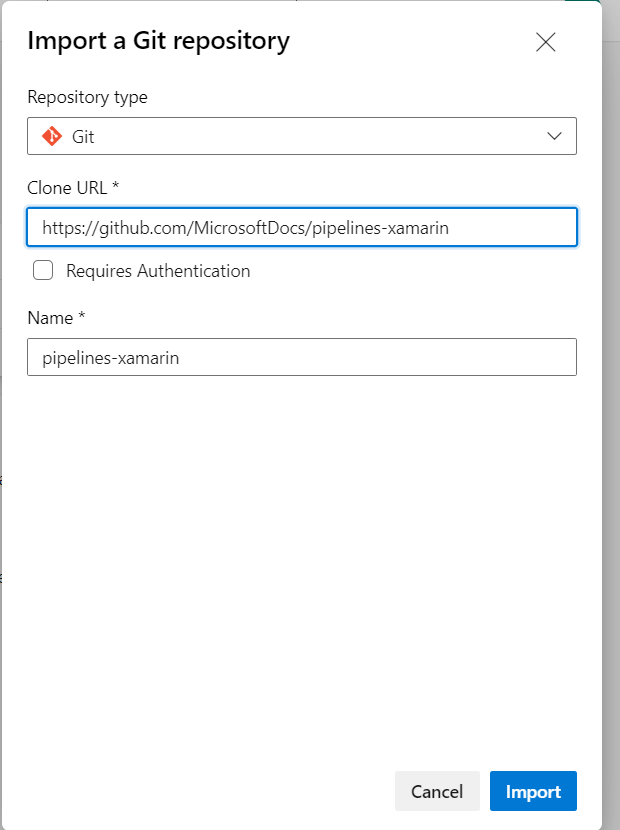
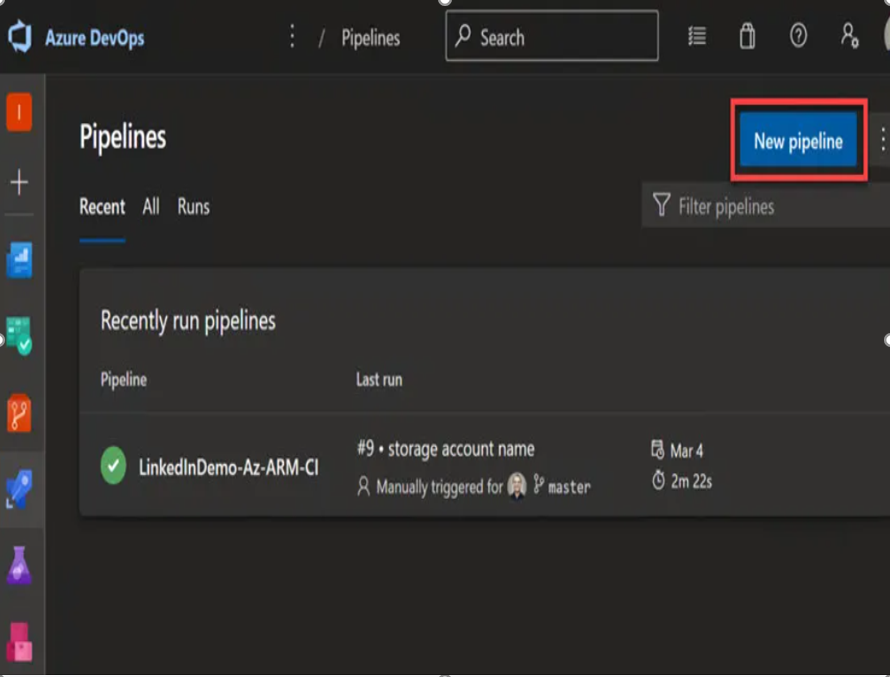
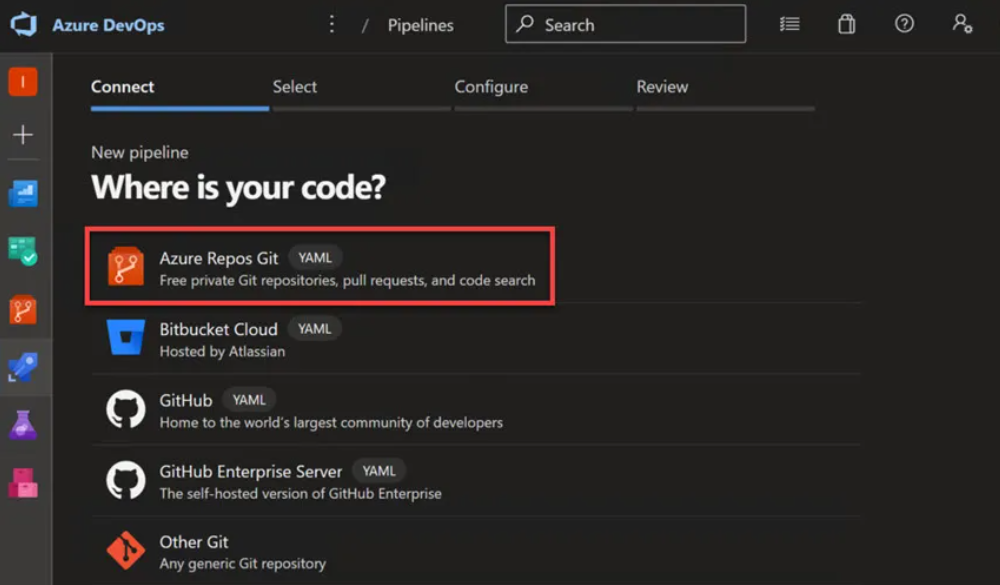
1. Navigate to team project on Azure DevOps in a new browser tab.
2. Import the code from GitHub to azure repos.



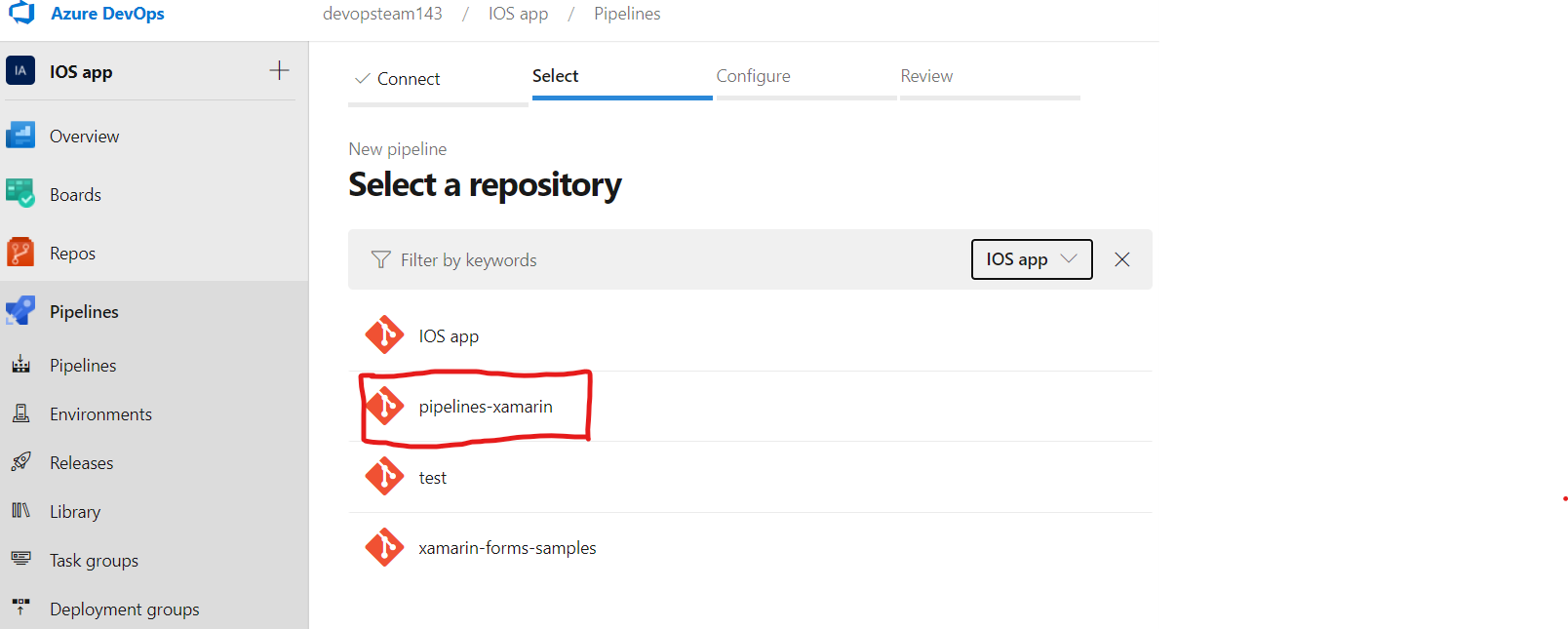
1. Navigate to the Pipelines hub on the left-hand side.
2. Click ‘New Pipeline’. We can use new YAML file or existing YAML file.



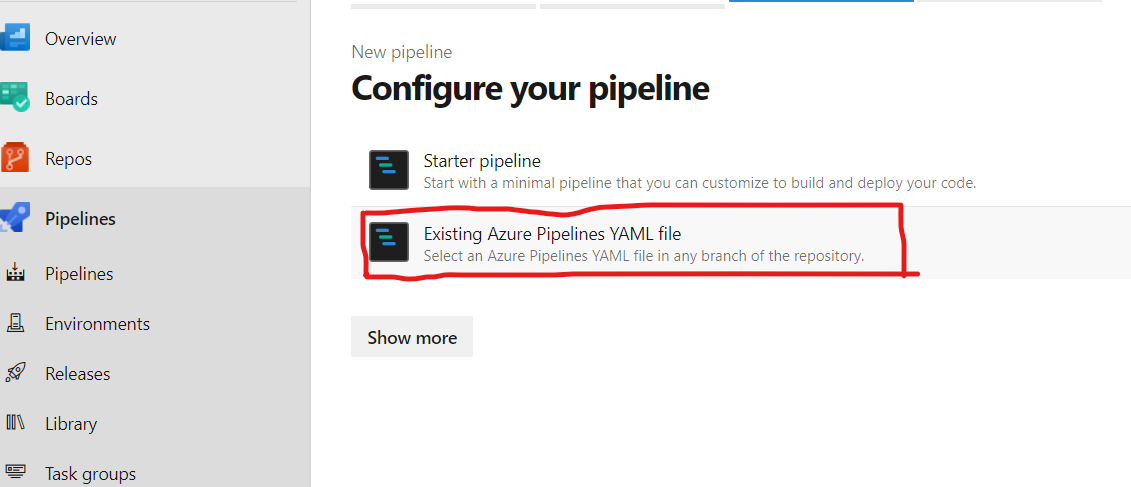
1. Next, select the ‘Azure Repos Git’ as the source hosting platform.



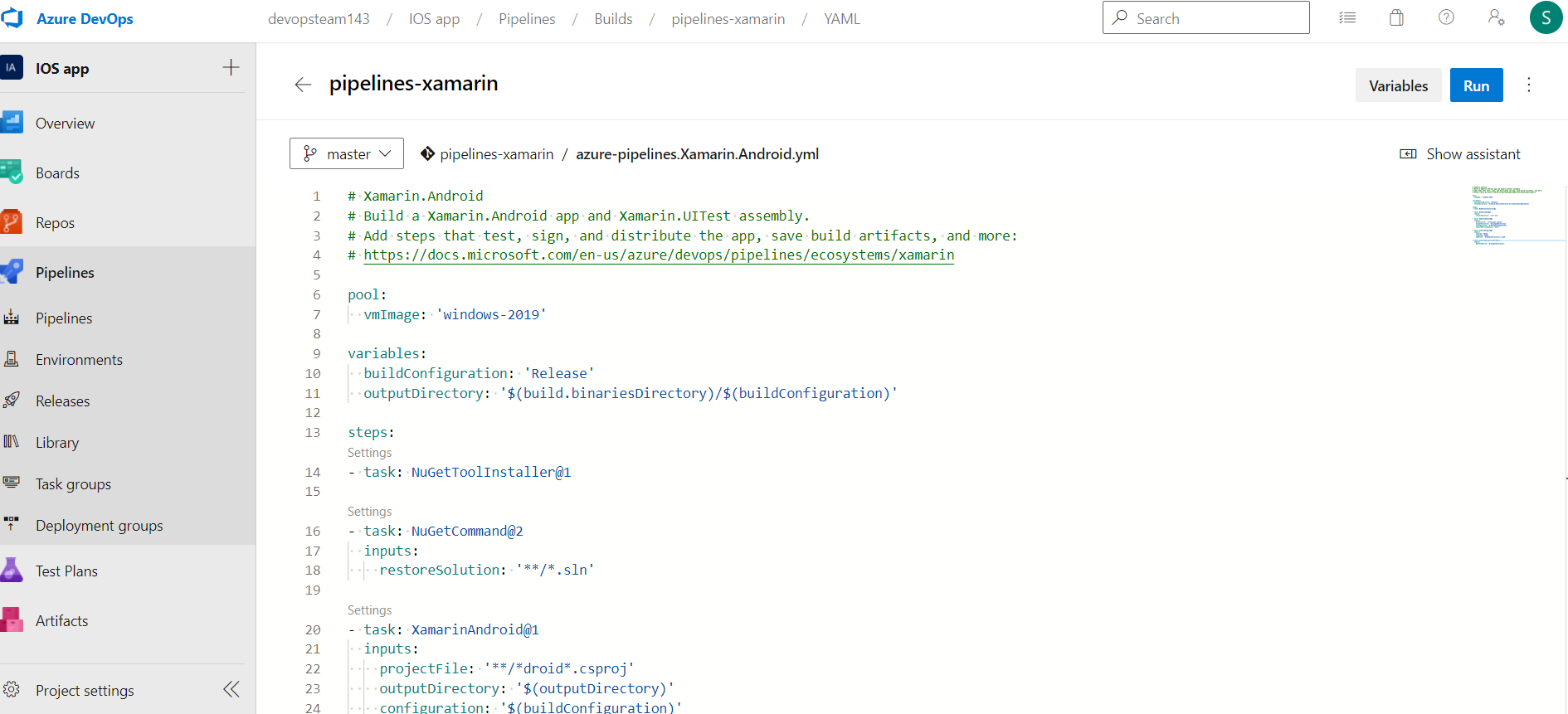
1. Select repository. In this example, the repo is called ‘pipelines-xamarin’.



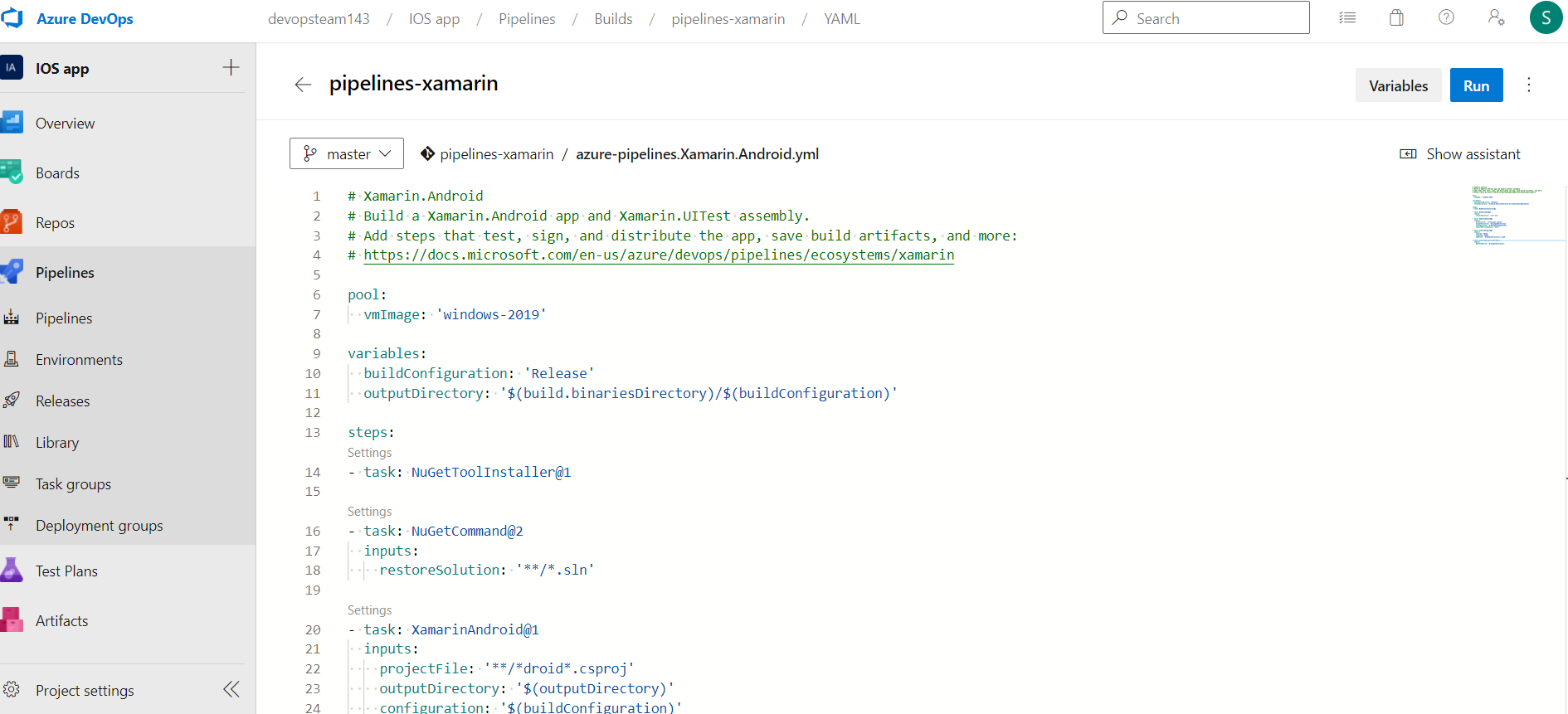
1. Next, select the ‘Existing Azure Pipelines file’ template.



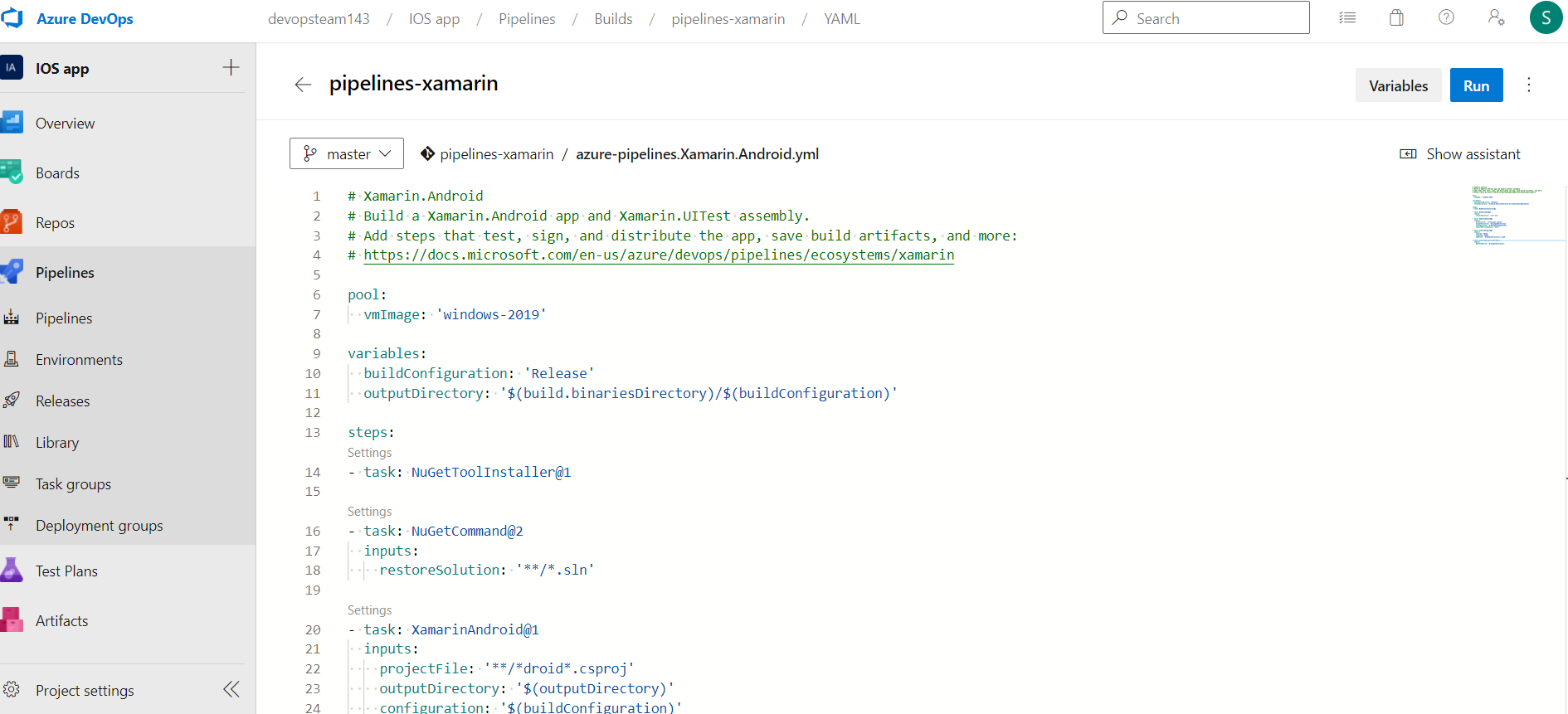
1. The Existing Azure Pipelines file’ will open existing YAML definition.



1. Next, we need to change the YAML definition as per our project requirement. Please refer the below example.



1. Finally, click Save and run the pipeline.



1. Basic Tasks used in YAML file composition:

* Xamarin Android Task : XamarinAndroid@1

Use this task to build an Android app with Xamarin.

Some of the Arguments that can be used with the Xamarin Task.

Project : Relative path from repo root of Xamarin.Android project(s) to build.

For example, \*\*/\*.csproj for all csproj files in all subfolders.

Output Directory : Optionally provide the output directory for the build.  
 Example: $(build.binariesDirectory)/bin/Release

Configuration : Specify the configuration you want to build such as debug or release.

MSBuild version : If the preferred version cannot be found, the latest version found will be

Used Instead.

* Android signing task : AndroidSigning@3

Use this task in a pipeline to sign and align Android APK files.

Sign the APK :  Select this option to sign the APK with a provided Android Keystore file.

Unsigned APKs can only run in an emulator.

Zipalign : Select if you want to zipalign your package. This reduces the amount of RAM

consumed by an app.

APK files : Relative path from the repo root to the APK(s) you want to sign.

Default value: \*\*/\*.apk

* **Publish Build Artifacts task :** PublishBuildArtifacts@1

# Use this task in a build pipeline to publish build artifacts to Azure Pipelines, TFS, or a file

# Share.

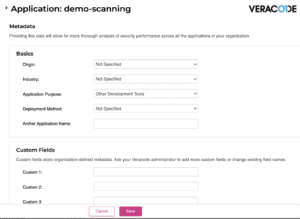
Path to publish : The folder or file path to publish. This can be a fully-qualified path or a path

# relative to the root of the repository.

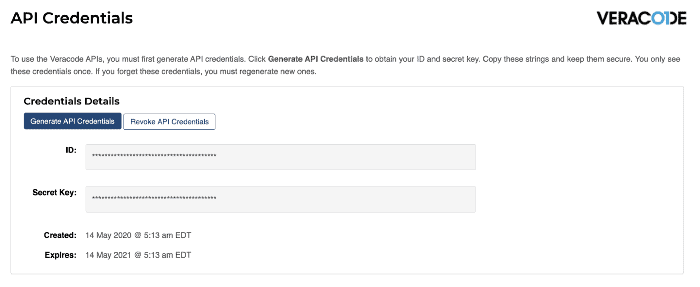
# 

1. **Veracode Integration :**

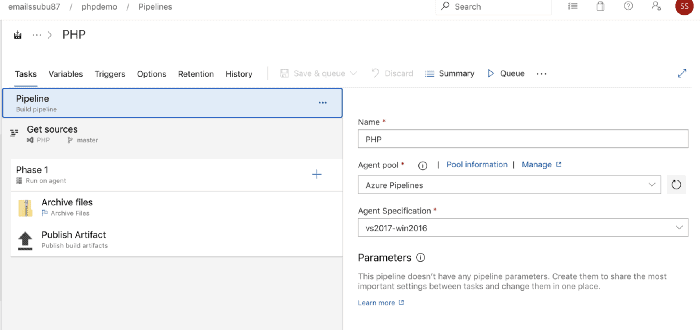
* We have to get the Veracode details from them such as the login and other details from the welcome email sent from the Veracode team.
* Once after we login, we have an option to create our own project for our demo analysis.
* After the registration of the Demo Project we would be able to run Demo-scanning.

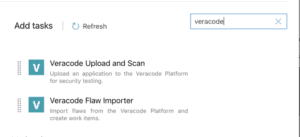


* Now the next step is to create an API key from the Veracode and then add it as part of the CICD using Azure DevOps.
* Click on the API Credentials and Generate the new code as part of the CICD process.

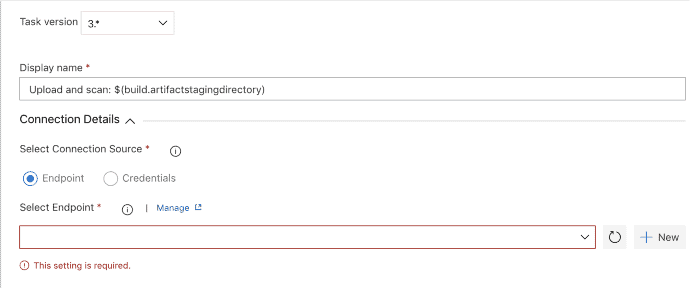


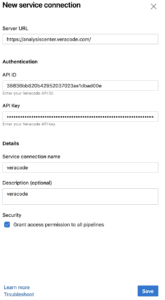
* Our next step is to create an Azure DevOps Plugin from the Marketplace.
* Include the Veracode task while creating the CI/CD Pipeline.

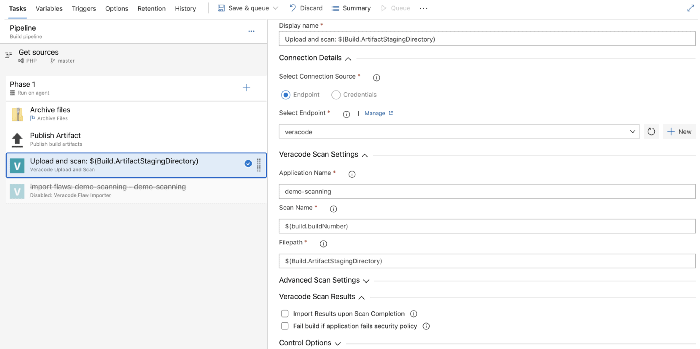




1. Next, we need to create a new Service Endpoint to integrate our Azure DevOps with Veracode.

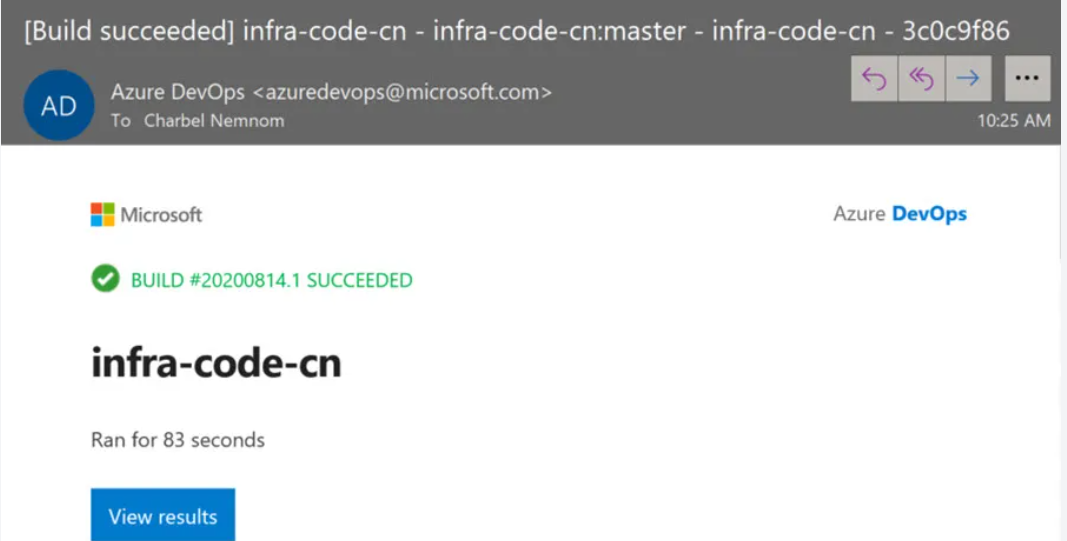






* We can Save and Run the Pipeline once the connection is setup.

1. Finally, we receive an email if the build failed or succeeded similar to the below.

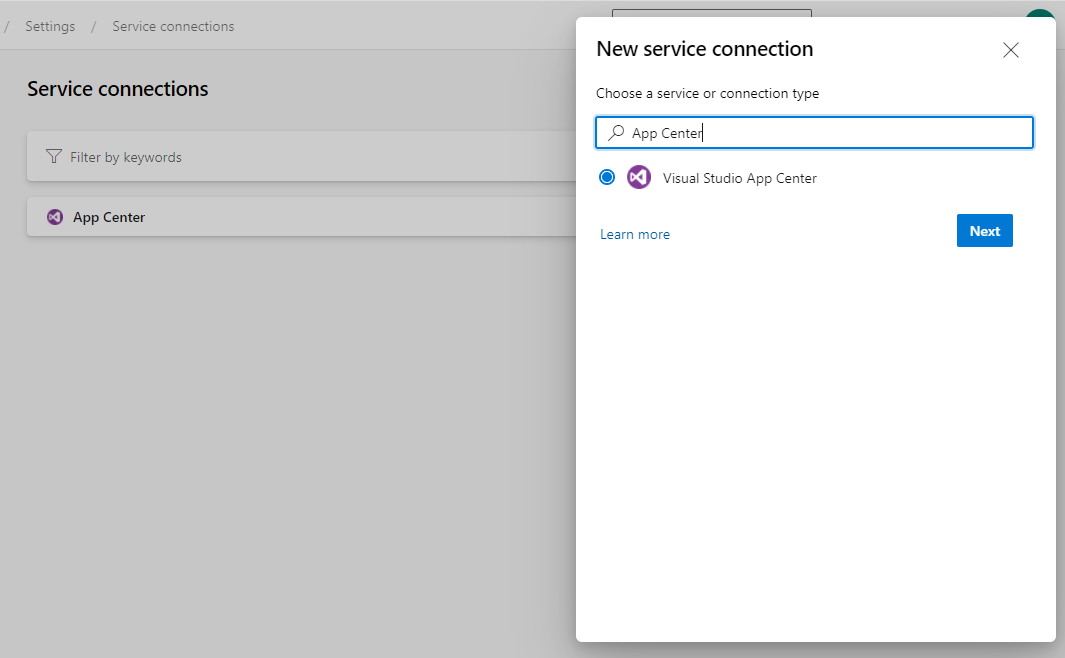


1. You can find the above sample source code on GitHub.

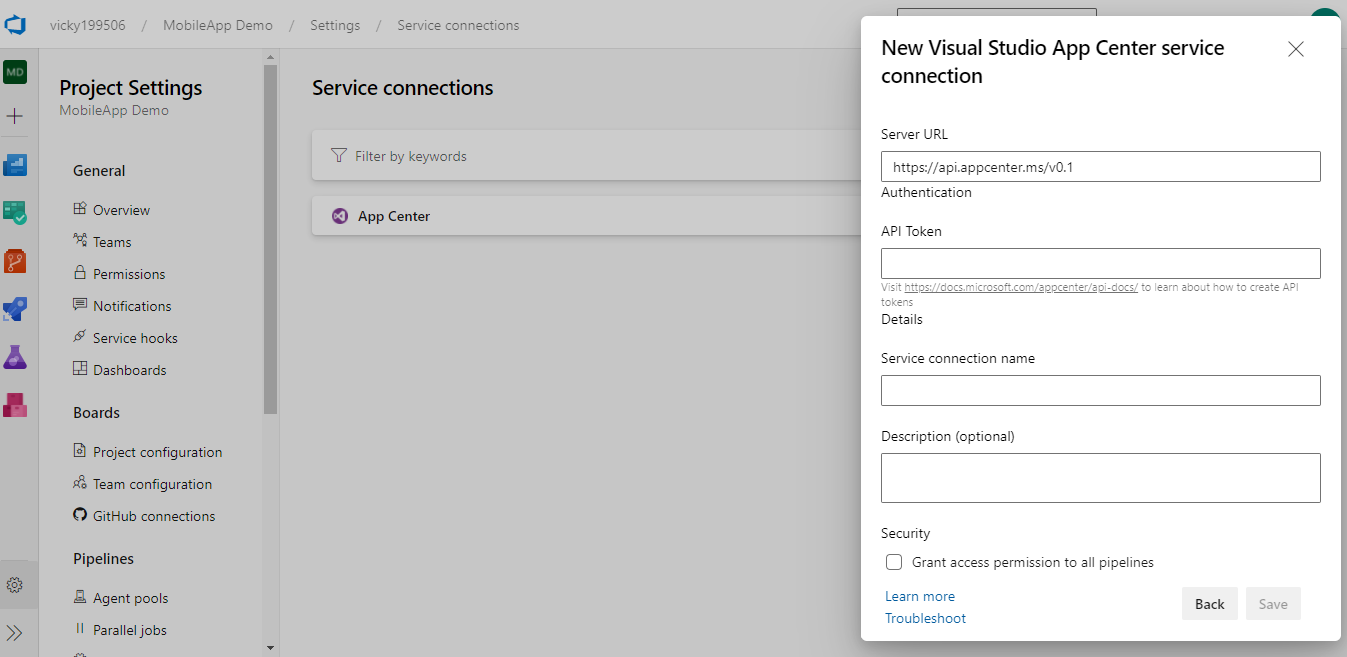
GitHub URL: [GitHub - MicrosoftDocs/pipelines-xamarin: Sample Xamarin application referred to by Azure Pipelines documentation](https://github.com/MicrosoftDocs/pipelines-xamarin)

1. Once we have the apk file published as an artifact we can distribute it to Microsoft App Center.
2. Steps for setting up a connection with App center:

* Create a new account/Login into Microsoft App Center.
* We need to setup a connection with Azure DevOps for which we will create a service connection between Azure DevOps and Microsoft App Center.

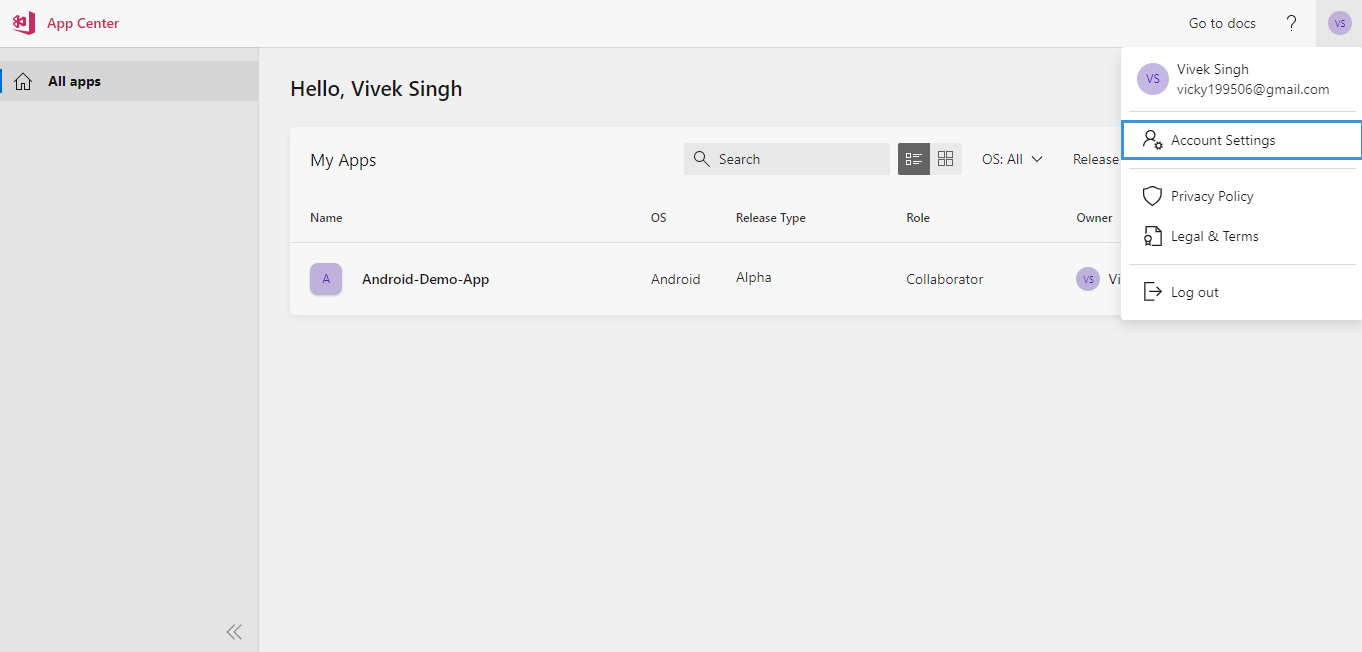


* Select Visual Studio App Center and Click Next.

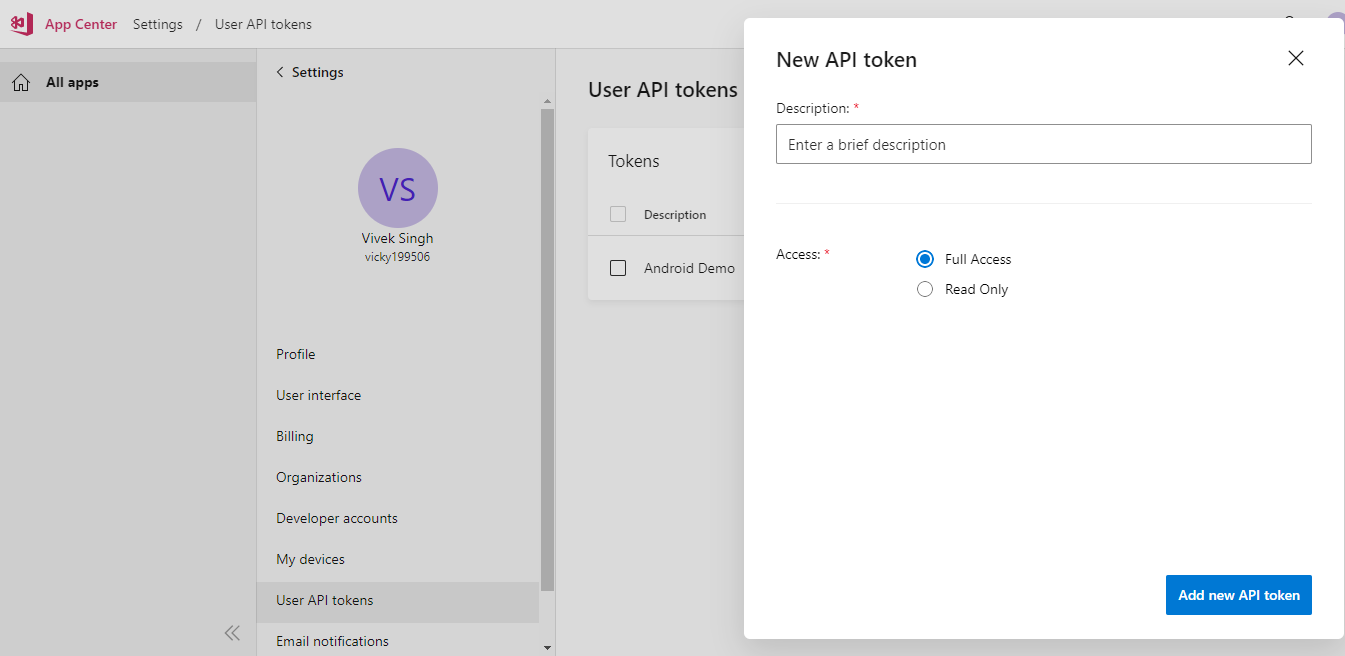


For API Token :

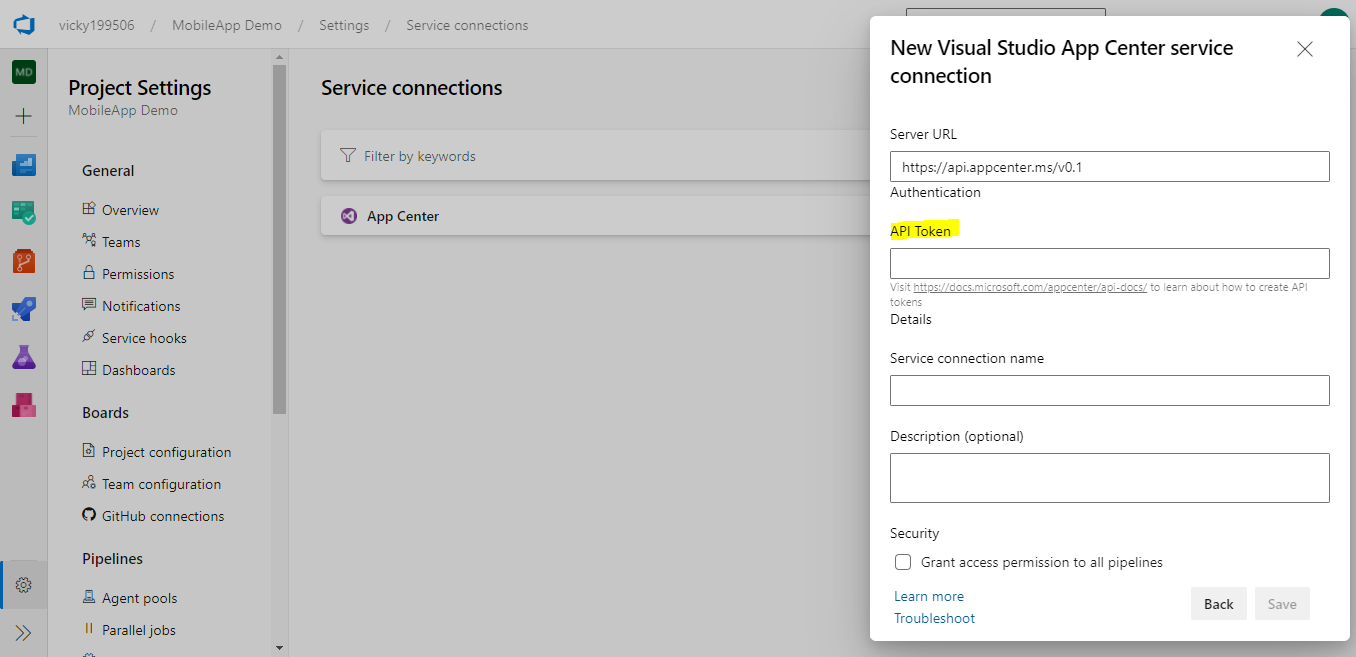
* Login into App Center and Navigate to Account Settings as shown below:



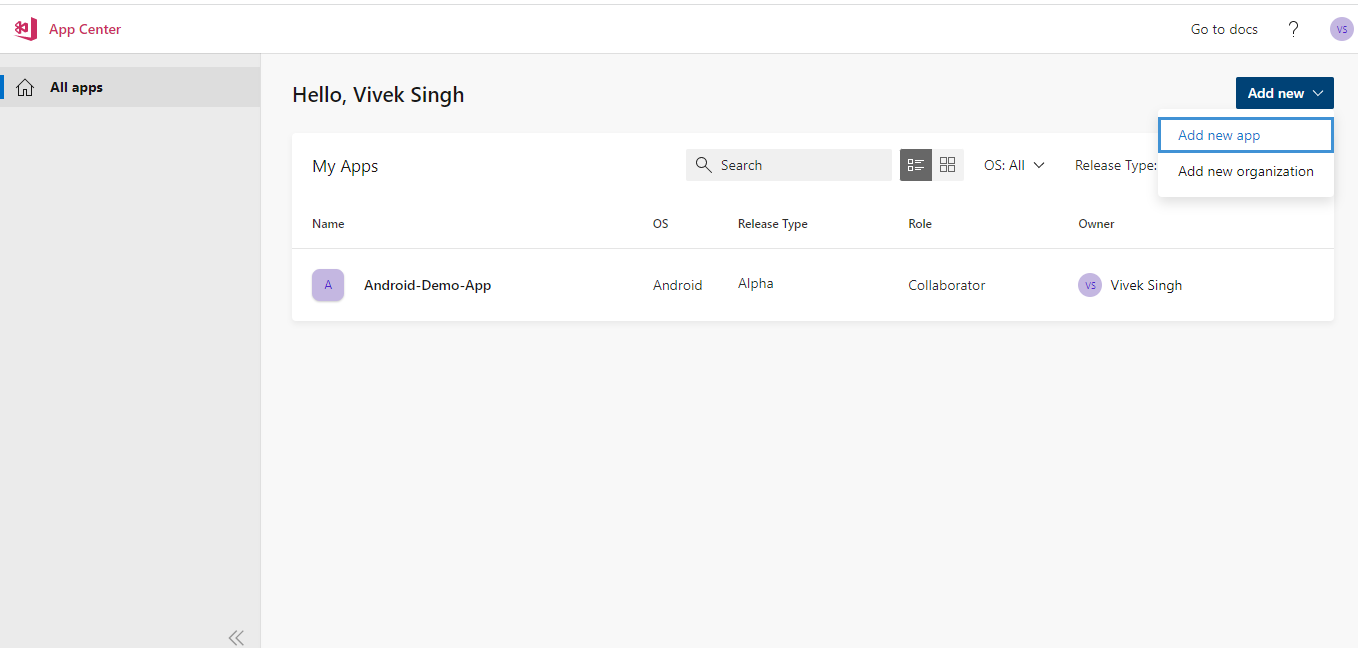
* Navigate to API Tokens and Create a new Token with Full Access.

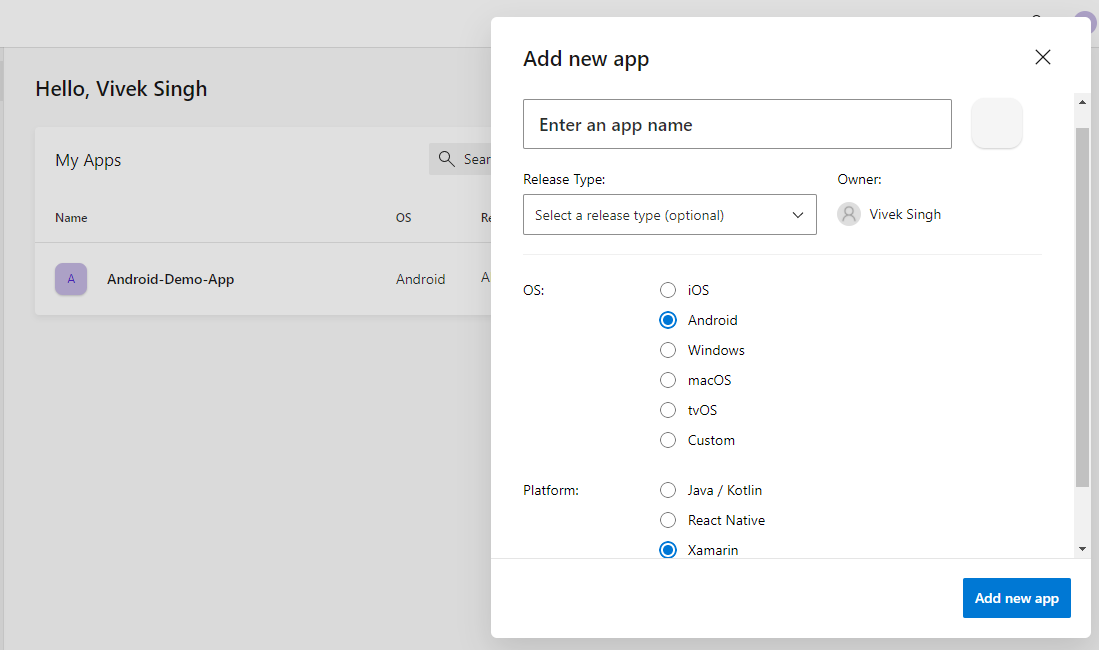


* Copy the Access Token and store it as we would not be able to view it later.
* Use this API token for setting up the Service Connection.

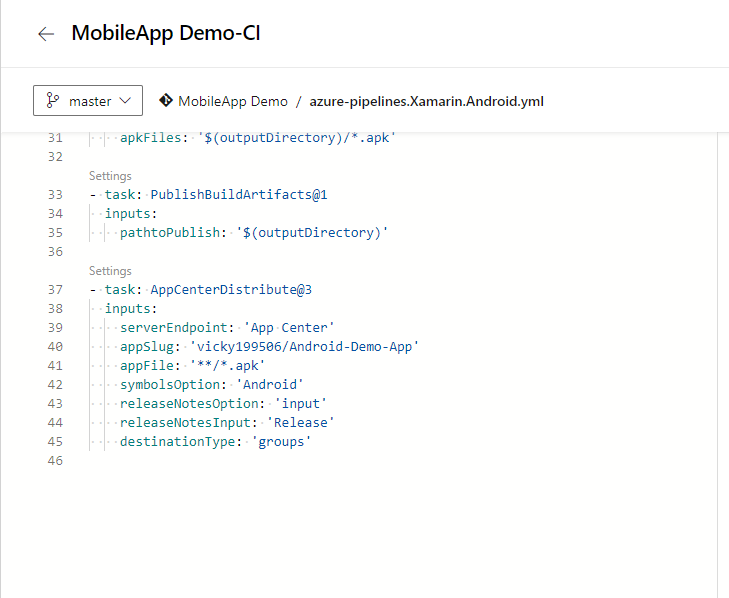


* Create a new App in the App center and select the appropriate options.

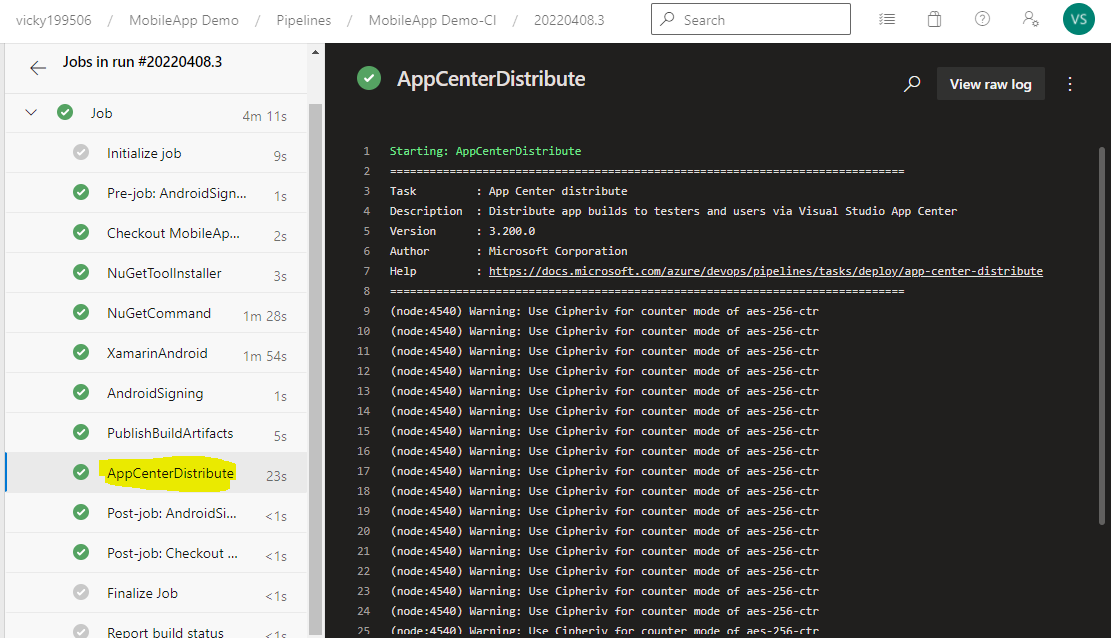




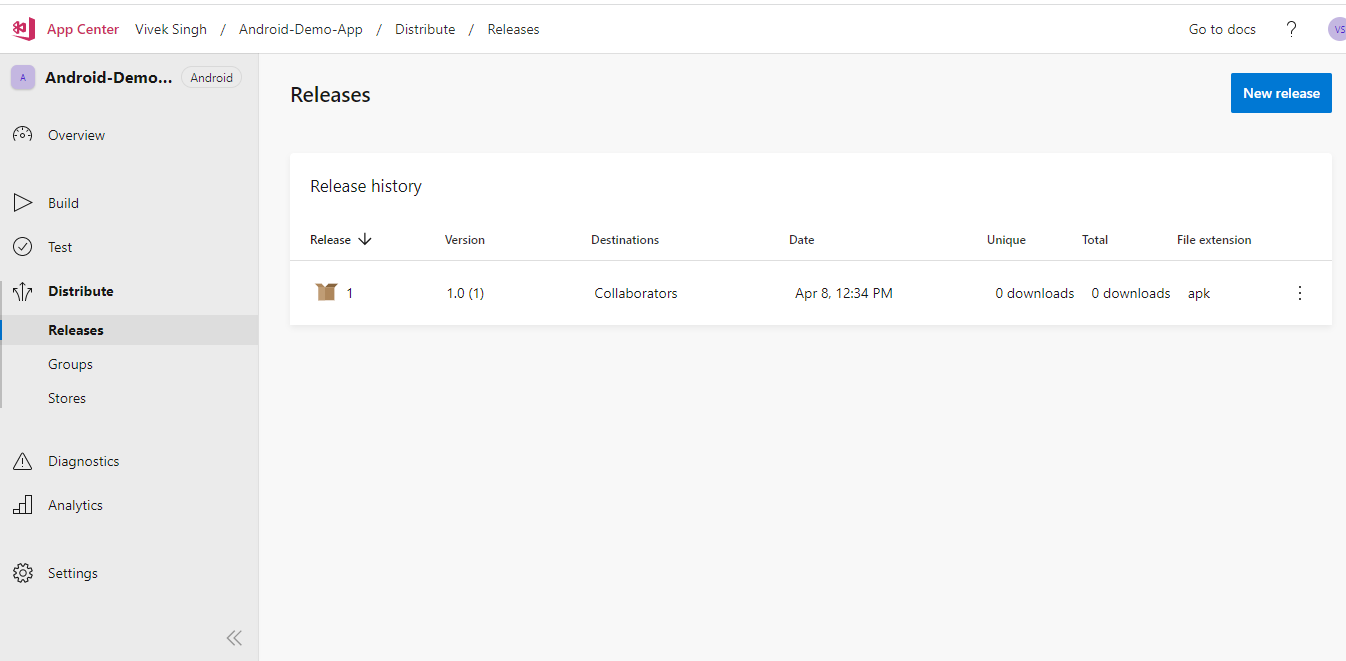
* Once the Service Connection is setup we need to add up the App Distributor task in the Pipeline YAML file as shown below:

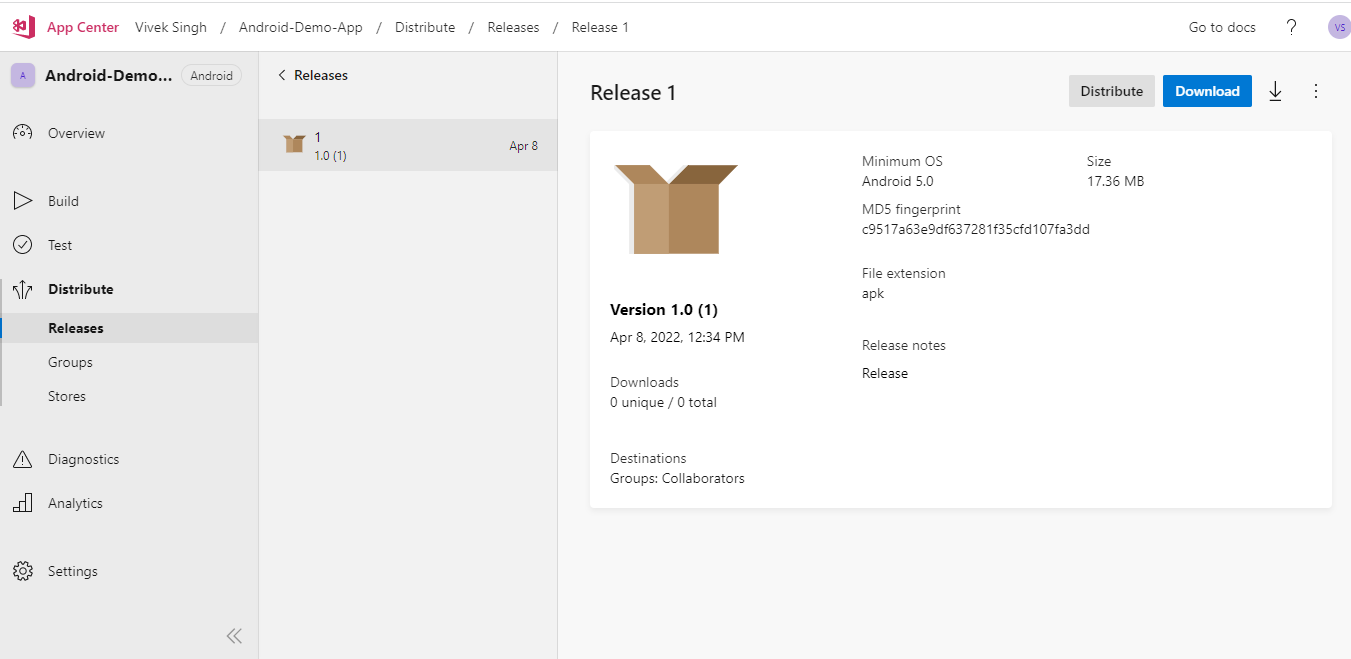


* Once the Task is added, Save and Run the Pipeline.
* Now when the Build Pipeline is run App Center is run as part of the Pipeline.



* Once the Pipeline is run a new Release would be created on App center.





* You can access the Release from the App center and whenever a new build is run a new version is created on the App center.

**Release Pipeline:**

* We can also create an Azure DevOps Release Pipeline for creating a new version via App Center.
* Whenever the Build Pipeline is triggered the generated artifact will be picked up by the Release Pipeline.
* The App Center Distributor task will be the part of Release pipeline, which you would be triggered once the Build Pipeline run is complete and new artifact (.apk file ) is published.

