5.5 MODIFY AZURE-PIPELINES.YAML FILE

A pipeline is defined using a YAML file in your repo. Usually, this file is named azure-pipelines.yml and is located at the root of your repo.

Go to the **Pipelines** page in Azure Pipelines, select the pipeline you created, and select **Edit** in the context menu of the pipeline to open the YAML editor.

Examine the contents of the YAML file.

```
trigger:
 - main
 pool:
   vmImage: 'ubuntu-latest'
 steps:
 - task: Maven@4
   inputs:
     mavenPomFile: 'pom.xml'
     mavenOptions: '-Xmx3072m'
     javaHomeOption: 'JDKVersion'
     jdkVersionOption: '1.11'
     jdkArchitectureOption: 'x64'
     publishJUnitResults: false
     testResultsFiles: '**/surefire-reports/TEST-*.xml'
     goals: 'package'
```

The pipeline runs whenever your team pushes a change to the main branch of the repo or creates a pull request. It runs on a Microsoft-hosted Linux machine. The pipeline process has a single step, which is to run the Maven task.

a) Change the platform to build on

You can build your project on Microsoft-hosted agents that already include SDKs and tools for various development languages. Or, you can use self-hosted agents with specific tools that you need.

- Navigate to the editor for your pipeline by selecting **Edit pipeline** action on the build, or by selecting **Edit** from the pipeline's main page.
- Currently the pipeline runs on a Linux agent:

```
pool:
  vmImage: "ubuntu-latest"
```

- To switch to a different platform like Windows or Mac, change the vmImage value:

```
pool:
  vmImage: "windows-latest"
pool:
  vmImage: "macos-latest"
```

- Select **Save** and then confirm the changes to see your pipeline run on a different platform.

b) Add steps

You can add more **scripts** or **tasks** as steps to your pipeline. A task is a pre-packaged script. You can use tasks for building, testing, publishing, or deploying your app. For Java, the Maven task we used handles testing and publishing results, however, you can use a task to publish code coverage results too.

- Open the YAML editor for your pipeline.
- Add the following snippet to the end of your YAML file.

```
- task: PublishCodeCoverageResults@2
  inputs:
```

```
    summaryFileLocation: "$(System.DefaultWorkingDirectory)/**/site/jacoco/jacoco.xml"
# Path to summary files
    reportDirectory: "$(System.DefaultWorkingDirectory)/**/site/jacoco" # Path to report
directory
    failIfCoverageEmpty: true # Fail if code coverage results are missing
```

- Select **Save** and then confirm the changes.
- You can view your test and code coverage results by selecting your build and going to the **Test** and **Coverage** tabs.

c) Build across multiple platforms

You can build and test your project on multiple platforms. One way to do it is with strategy and matrix. You can use variables to conveniently put data into various parts of a pipeline. For this example, we'll use a variable to pass in the name of the image we want to use.

- In your azure-pipelines.yml file, replace this content:

```
pool:
  vmImage: "ubuntu-latest"
```

with the following content:

```
strategy:
  matrix:
    linux:
      imageName: "ubuntu-latest"
    mac:
      imageName: "macOS-latest"
    windows:
      imageName: "windows-latest"
  maxParallel: 3
```

```
pool:
  vmImage: $(imageName)
```

- Select **Save** and then confirm the changes to see your build run up to three jobs on three different platforms.

Each agent can run only one job at a time. To run multiple jobs in parallel you must configure multiple agents. You also need sufficient parallel jobs.

d) Build using multiple versions

To build a project using different versions of that language, you can use a matrix of versions and a variable. In this step, you can either build the Java project with two different versions of Java on a single platform or run different versions of Java on different platforms.

If you want to build on a single platform and multiple versions, add the following matrix to your azure-pipelines.yml file before the Maven task and after the vmImage.

```
strategy:
  matrix:
    jdk10:
      jdkVersion: "1.10"
    jdk11:
      jdkVersion: "1.11"
  maxParallel: 2
```

- Then replace this line in your maven task:

  ```
  jdkVersionOption: "1.11"
  ```

  with this line:

  ```
  jdkVersionOption: $(jdkVersion)
  ```

- Make sure to change the $(imageName) variable back to the platform of your choice.

- If you want to build on multiple platforms and versions, replace the entire content in your azure-pipelines.yml file before the publishing task with the following snippet:

```
trigger:
- main
strategy:
  matrix:
    jdk10_linux:
      imageName: "ubuntu-latest"
      jdkVersion: "1.10"
    jdk11_windows:
      imageName: "windows-latest"
      jdkVersion: "1.11"
  maxParallel: 2
pool:
  vmImage: $(imageName)
steps:
- task: Maven@4
  inputs:
    mavenPomFile: "pom.xml"
    mavenOptions: "-Xmx3072m"
    javaHomeOption: "JDKVersion"
    jdkVersionOption: $(jdkVersion)
    jdkArchitectureOption: "x64"
    publishJUnitResults: true
    testResultsFiles: "**/TEST-*.xml"
    goals: "package"
```

- Select **Save** and then confirm the changes to see your build run two jobs on two different platforms and SDKs.

e) Customize CI triggers

Pipeline triggers cause a pipeline to run. You can use trigger: to cause a pipeline to run whenever you push an update to a branch. YAML pipelines are configured by default with a CI trigger on your default branch (which is usually main). You can set up triggers for specific branches or for pull request validation. For a pull request validation trigger, just replace the trigger: step with pr: as shown in the two examples below. By default, the pipeline runs for each pull request change.

- If you'd like to set up triggers, add either of the following snippets at the beginning of your azure-pipelines.yml file.

```
trigger:
  - main
  - releases/*
```

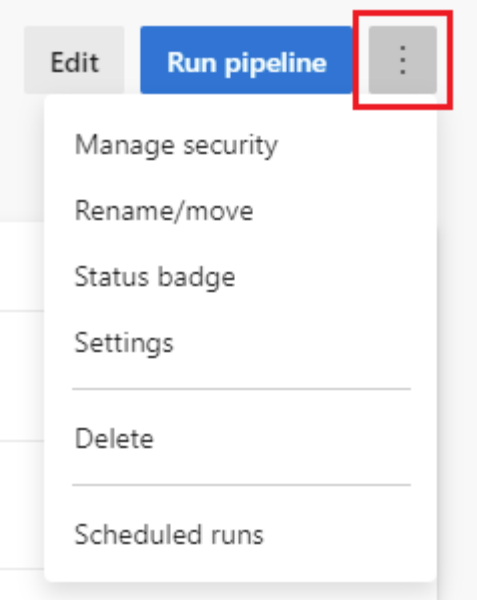
YAMLCopy

```
pr:
  - main
  - releases/*
```

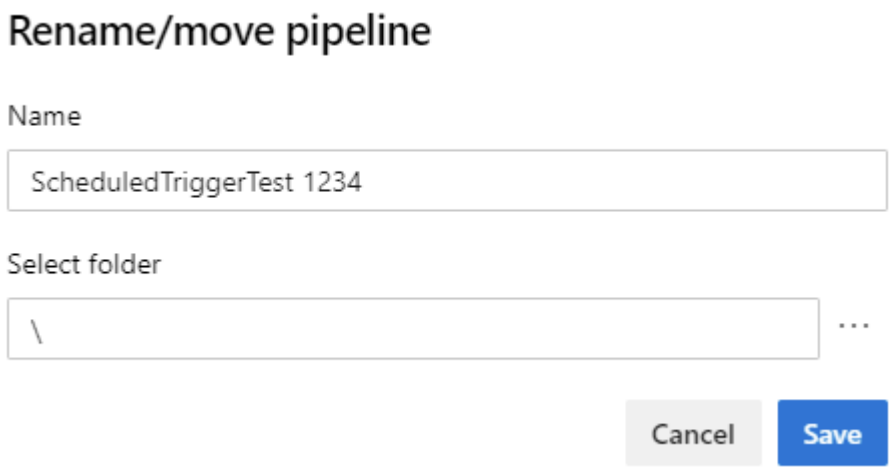
You can specify the full name of the branch (for example, main) or a prefix-matching wildcard (for example, releases/*).

f) Pipeline settings

You can view and configure pipeline settings from the **More actions** menu on the pipeline details page.

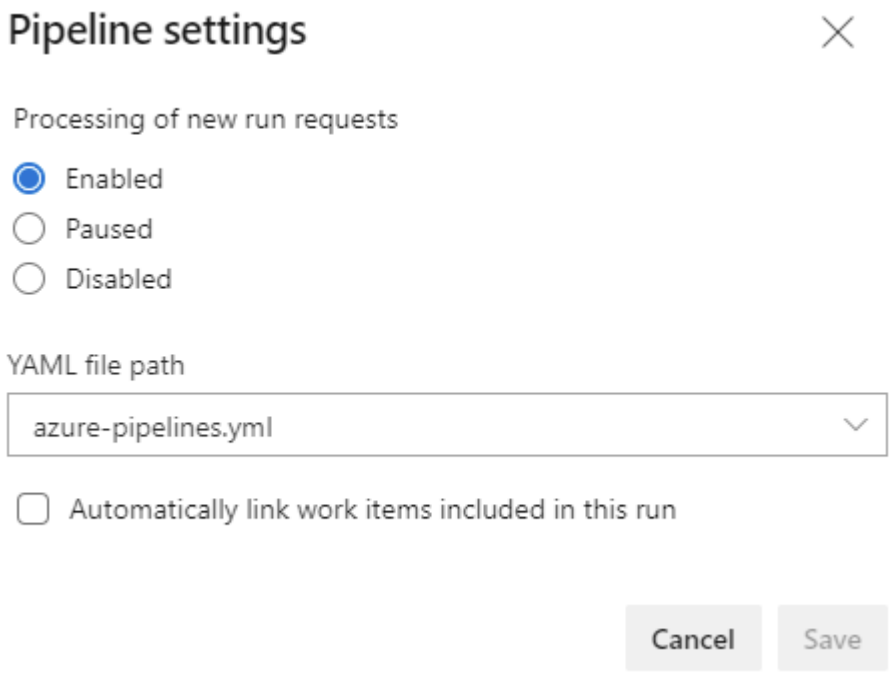


Rename/move – Edit your pipeline name and folder location.



- Status badge –Add a status badge to your Repository

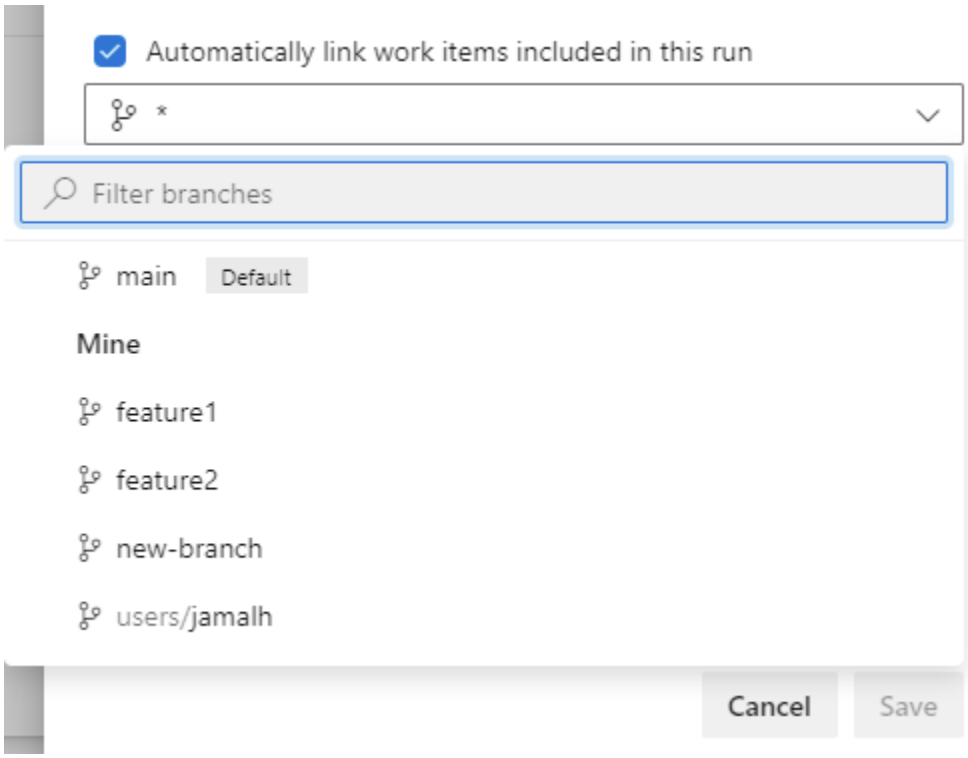
- **Delete** - Deletes the pipeline including all builds and associated artifacts.
- **Scheduled runs** –
- Choose **Settings** to configure the following pipeline settings.



From the **Pipeline settings** pane, you can configure the following settings.

- **Processing of new run requests** - Sometimes you'll want to prevent new runs from starting on your pipeline.
  - By default, the processing of new run requests is **Enabled**. This setting allows standard processing of all trigger types, including manual runs.
  - **Paused** pipelines allow run requests to be processed, but those requests are queued without actually starting. When new request processing is enabled, run processing resumes starting with the first request in the queue.
  - **Disabled** pipelines prevent users from starting new runs. All triggers are also disabled while this setting is applied. All build policies using a disabled pipeline will show "Unable to queue Build" message next to the build policy in the PR overview window and the status of the build policy will be broken.
- **YAML file path** - If you ever need to direct your pipeline to use a different YAML file, you can specify the path to that file. This setting can also be useful if you need to move/rename your YAML file.

- **Automatically link work items included in this run** - The changes associated with a given pipeline run might have work items associated with them. Select this option to link those work items to the run. When **Automatically link work items included in this run** is selected, you must specify either a specific branch, or * for all branches, which is the default. If you specify a branch, work items are only associated with runs of that branch. If you specify *, work items are associated for all runs.



  - To get notifications when your runs fail, see how to Manage notifications for a team