Making and Validating Changes to the Staging Pipeline

Need to edit the staging pipeline for some reason? This doc has you covered.

Making Changes

The staging pipeline (Stage-DotNet) is fairly easy to make changes to. The pipeline has two main components: its YAML and the Release CLI/Library C# code.

The Release CLI makes extensive use of dependency injection. New changes may need to also modify `ServiceCollectionExtensions.cs` in order to have their dependency injection work properly. Additionally, any new operations will need to be added to `Program.cs` in alphabetical order.

The Signing Extensions project is where all of our signing-related tasks live. If you need to make changes to the signing setup, modify this project.

The Release CLI, Release Library, and Signing Extnesions already have tests alongside their code. Any new functionality should have test coverage added to it.

Validating Changes

So you've recently made a change to the pipeline and you want to make sure you're not going to break anything? Good for you! Lucky for you, we (the authors of this document) have done some work to make that easier for you.

Stage-DotNet-Test

This pipeline is your best friend when it comes to validating changes in the staging pipeline. It has a separate entry point from Stage-DotNet (`staging-test-pipeline.yml` vs. `staging-pipeline.yml`), but otherwise uses the exact same YAML templates and files past that entry point. A few notable changes from the staging pipeline to the staging test pipeline:

- Stage-DotNet-Test has a BAR ID prefilled for you – we've created a build that is known to work. Note that when we say "work," we don't mean "entirely green"; you will still notice some orange circles in the validation steps. However, all the pipeline functionality itself will work perfectly for you.
- Stage-DotNet-Test skips over approval stages – no need to babysit it!
- Stage-DotNet-Test tests publishing by actually publishing to temporarily created feeds and containers, so any changes to publishing will actually be validated!

Simply run the test pipeline on your branch and then wait six hours for it to finish and you'll be golden!

What? Six Hours?

Okay, okay. If you have to iterate rapidly for some reason, there is a way to save time and test only a particular stage or set of stages.

1. Find a previous, successful run of the test pipeline. Copy the Build ID (the bit after `buildId=` in the URI).
2. Pull up the YAML file for the stage(s) you want to run. Add the following inputs to any `DownloadPipelineArtifact` tasks:

```yaml
source: 'specific'
project: '7ea9116e-9fac-403d-b258-b31fcf1bb293'
pipeline: 799
preferTriggeringPipeline: true
runVersion: 'specific'
runId: the build ID copied from earlier
allowPartiallySucceededBuilds: true
allowFailedBuilds: true
```

Note: if you want to use artifacts from Stage-DotNet instead of Stage-DotNet-Test, set `pipeline` to `792` instead.

3. Open up `eng/pipeline/stage_dotnet.yml` and comment out all of the stages prior to the one you're testing. Then make sure to comment out the dependencies on those stages.
4. Running the pipeline now will skip the most time-consuming stages and go straight to the stage you want to test.

Please still make sure to run the full test pipeline before checking in.

Was this helpful?