AZURE DEVOPS GUARDRAILS– NAMING CONVENTIONS AND STANDARDS

—

The purpose of for this post is to ensure that our clients are implementing preliminary baseline set of CICD controls within their cloud-based environments.

#Azure DevOps #CICD #Release Flow #Azure

Summary

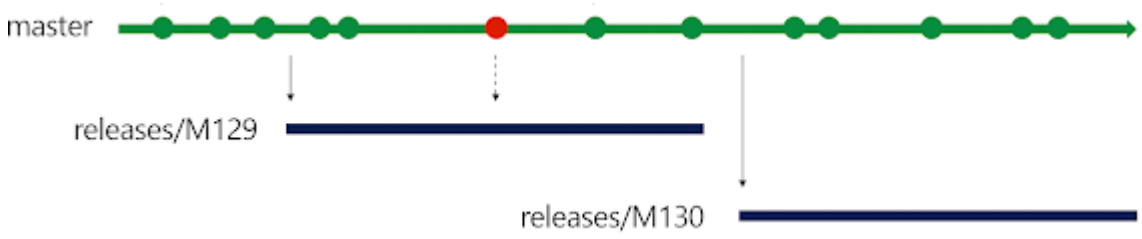
We need guardrails to bring a team to be productive and to a state that can work well with Azure DevOps pipelines.

DevOps team needs to define team agreements in early stages before any organization starting their CICD journey before it is too late. If not, in some points, you will realize that nearly every member has his own and mostly very rational interpretation and view on how git branches should be arranged, how folders should be structured, how database views should be named, and so on, then it will be much harder to define any effective DevOps pipelines.

What will be covered in this blog posting:

- Microsoft "Release Flow" as a practical Git branching strategy
- Repository Naming Convention
- Repository Folder Structure
- Pipeline Naming Convention
- Branching Naming Convention

Release Flow



Release Flow is Microsoft's model of branching and merging. It is how they manage nearly all their production deployments and releases. They also use Azure DevOps internally so it supports it well.

This is how developers get changes into our code base -- and at this point, the RF branching strategy looks like a typical trunk-based development model. But unlike some other trunk-based strategies, like GitHub Flow, we do not deploy these changes to production to test them before merging the pull request, nor do we deploy to production when the pull request is merged.

Releases at Sprint Milestones

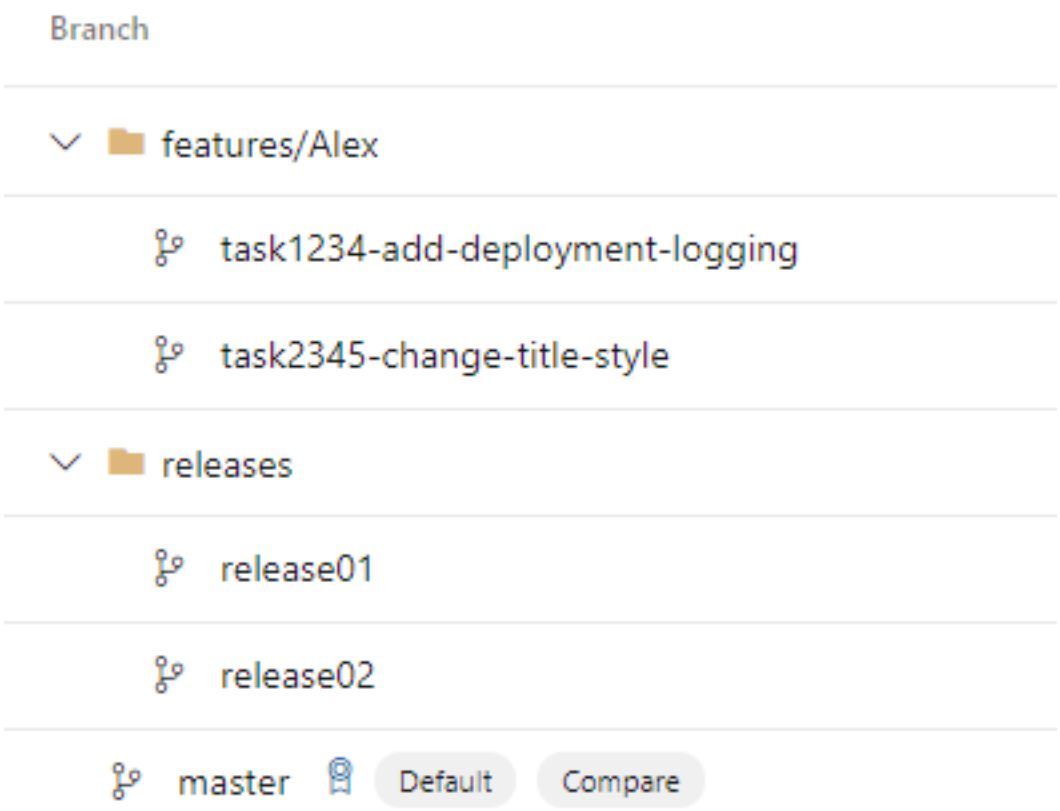
At the end of a sprint, we create a deployment branch from the master branch: for example, at the end of release 2021.09, we create a new branch releases/r2021.09. We then put the r2021.09 branch into production.

Why Release Flow?

One single pipeline to manage all git branches: master, features and releases branches.

We can build Azure DevOps pipelines as branching aware, possible minimal merging effort.

- Feature branch: Builds the code and releases it to the test environment only
- Release branch: Builds the code, releases it to the acceptance and after approval to production environments
- Pull request: Continuous Delivery is not needed, therefore the code building happens



A sample pipeline which has two files:

- stage-template.yaml: An abstract re-usable stage implemented using parameterized template
- main.yaml: The entry point of the pipeline. It contains triggers, pool information, and orchestration of the stages

main.yaml

```
trigger:
- features/*
- releases/*
stages:
- template: stage-template.yaml
  parameters:
    Name: Build
    Enabled: True
- template: stage-template.yaml
```

```
  parameters:
    Name: Test
    Enabled: ${{ startsWith(variables['Build.SourceBranch'], 'refs/heads/features') }}
- template: stage-template.yaml
  parameters:
    Name: Functional Test
    Enabled: ${{  startsWith(variables['Build.SourceBranch'], 'refs/heads/releases') }}
- template: stage-template.yaml
  parameters:
    Name: Post Test
    Enabled: ${{ startsWith(variables['Build.SourceBranch'], 'refs/heads/releases')  }}
```

`stage-template.yaml`

```
stages:
- stage: ${{ parameters.Name }}
  displayName: '${{ parameters.Name }} Stage'
  condition: and(not(failed()), eq('${{ parameters.Enabled }}', true))
  jobs:
  - job: ${{ parameters.Name }}
    displayName: '${{ parameters.Name }} Job'
    steps:
    - powershell: Write-Host "Job '${{ parameters.Name }}' is running. Parameter Enabled - '${{ parameters.Enabled }}' "
```

Repository Naming Convention Sample

General naming recommendations for Git repositories are:

- use lower case
- use dashes

Reference: github - Is there a naming convention for git repositories? - Stack Overflow

Format:
lowercase-with-hyphens

Examples:

- aws-ansible # not AWS_ANSIBLE
- sql-aws-db # not SqlAwsDB

Repository Folder Structure Sample

Any project needs to have a consistent folder structure in the repository. A sample top-level directory layout:

```
.
├── build             # YAML release pipeline definitions
├── docs              # Documentation files (alternatively ḋoc̀)
├── resources           # Resource files, like ARM templates (alternatively ṙeś)
├── src               # Source files (alternatively lib`or àpp)
├── tests             # Automated tests (alternatively ṡpec̀ or ṫest)
├── scripts           # Tools and utilities (alternatively 'tools')
└── README.md
```

Pipeline Naming Convention Sample

Format

> {Repo Name} : { Pipeline Description }

Goal: To keep pipelines in DevOps UI visually bounded to related repositories

Where:

- Repo Name is the name of the repository to which the pipeline belongs
- Pipeline Description describes what the pipeline does, for instance: CI Build or CD Release

Example:

> git-app-repo : ci-build

References: GOC Guardrails, Azure DevOps Release Flow, Alex Volok Consultancy

COMMENTS

Enter your comment...