Introduction to branching models

Table of contents

Git branching models are patterns to enable development teams using Git to manage their code in a streamlined manner. Since Git is established as a de facto standard for source code management (SCM) in the developer community, several approaches were designed to fulfill developers' requirements and manage the application source code lifecycle, with advantages and drawbacks depending on use cases. Based on the experience gained designing branching strategies, the pages in this section describe a blueprint implementation of a mainline-based development approach for mainframe applications using feature branches with an early integration pattern. This setup leverages a standardized development toolset based on an enterprise-wide Git provider and a continuous integration/continuous delivery (CI/CD) toolchain.

A characteristic of this integration pattern is that developers are implementing changes for a planned release and integrate their changes into a common permanent branch (the shared configuration) that is built, tested, and released together as one consistent entity.

The purpose of streamlining both the DevOps solutions and the delivery workflow is to simplify the process for development teams to deliver quality product releases on time. This enables agile development practices that allow the teams to respond more effectively to changes in the market and customer needs. The Git branching model for mainframe development introduces the branching model and outlines the development workflow from the developer's perspective. The details of the technical implementation with IBM Dependency Based Build (DBB) and zAppBuild, as well as packaging and deployment, are discussed in Pipeline design and implementation supporting the branching model. All branching models are adaptable to the needs of specific teams and their applications. Our branching approach advocates for best practices and indicates where variations can be applied.

The target audience of this branching model documentation is mainframe DevOps architects and SCM specialists interested in learning how to design and implement a CI/CD pipeline with a robust and state-of-the-art development workflow.

1 Aims and Assumptions

Some aims and assumptions that guide our recommendations include:

- The workflow and branching scheme should both scale up and scale down.
  - Small teams with simple and infrequent changes will be able to easily understand, adopt, and have a good experience.
  - Large, busy teams with many concurrent activities will be able to plan, track, and execute with maximum agility using the same fundamental principles.
- Planning and design activities as well as code development aim to align to a regular release cadence.
- There is no magic answer to managing large numbers of "in-flight" changes, so planning assumptions should aim as much as possible to complete changes quickly, ideally within one release cycle.

  DevOps/Agile practices typically encourage that, where possible, development teams should strive to break down larger changes into sets of smaller, incremental deliverables that can each be completed within an iteration. This reduces the number of "in-flight" changes, and allows the team to deliver value (end-to-end functionality) more quickly while still building towards a larger development goal.
- We know it is sometimes unavoidable for work to take longer than one release cycle and we accommodate that as a variant of the base workflow.

2 Choosing a workflow and branching model

Your choice of workflow and the branching model that supports it need to take into account your team's needs and characteristics.

Aspects to consider include:

- Size of the team
- Frequency of change
- Granularity of change
- Amount of parallel development

- Formality of release process

The workflows of our recommended Git branching model for mainframe development are flexible enough to scale from small teams with an infrequent pattern of small changes, to large and busier teams with many concurrent projects and projects spanning multiple cycles of a formal release cadence. These workflows are supported by the CI/CD pipeline implementation described in Pipeline design and implementation supporting the branching model.