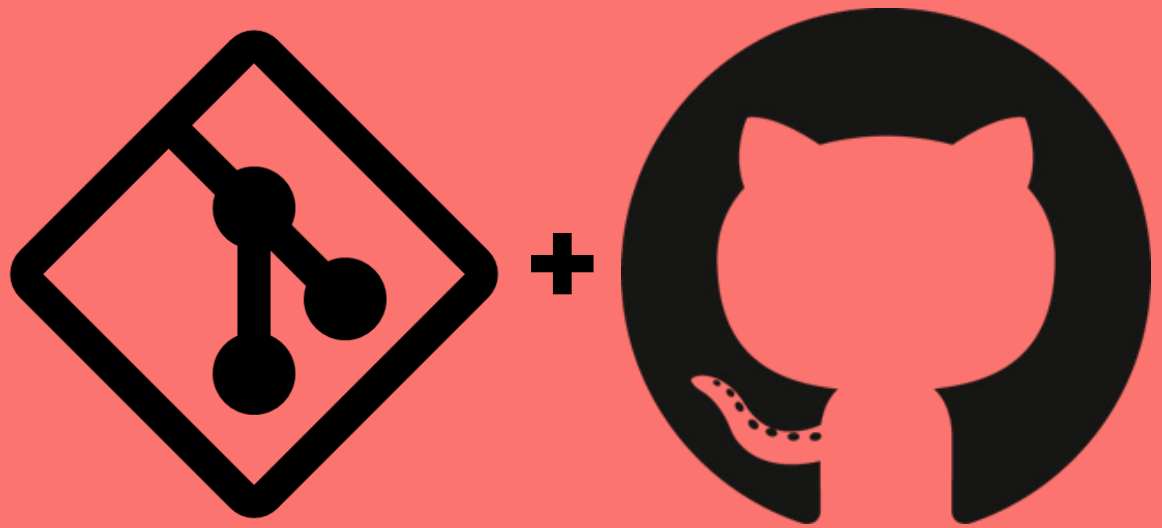
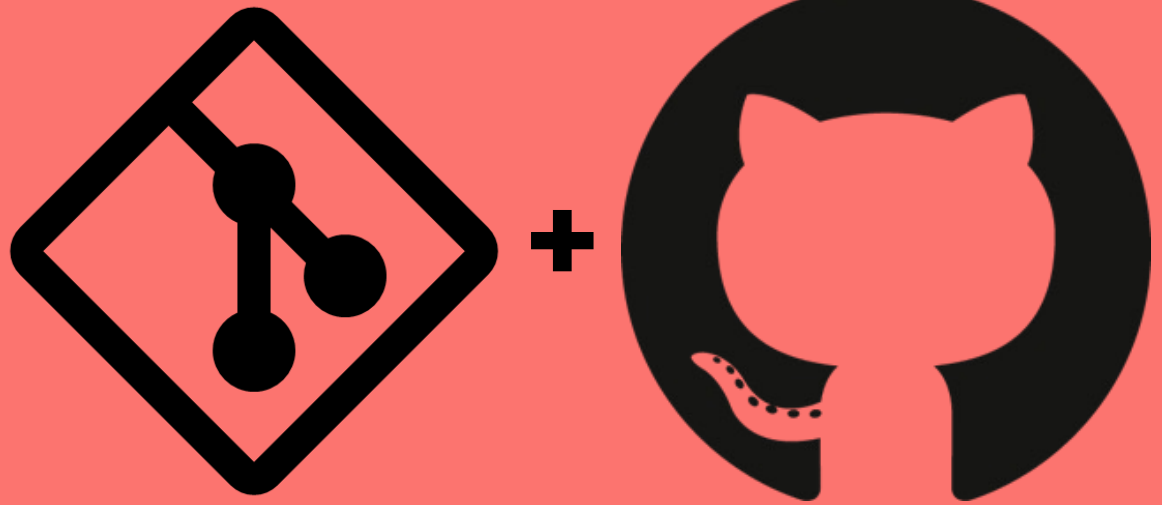
Git

In Practice



Git

In Practice

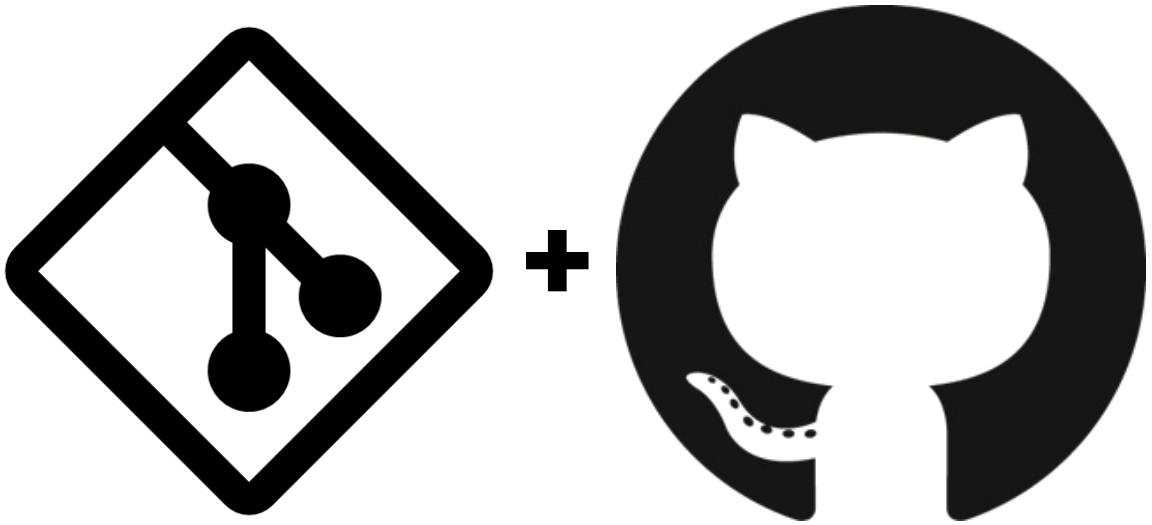


1 Branching Strategies

2 OTHERS

1 Branching Strategies

2 OTHERS



Branching Strategies

1. Integrating changes and structuring releases
    1. Mainline development – always integrate
    2. State, release and feature branches

2. Mainline development
    1. Few branches
    2. Relatively small commits
    3. High-quality testing and QA standards

3. State, release and feature branches
    1. Here, Branches enhances structure and workflows
        1. Different types of branches
        2. Fulfill different types of jobs

Branching Strategies

1. Branching conventions in a team depends on its size, the type of project and how releases are handled by your team / organization
2. git is very good when creating branches – but it doesn't tell how to use them
3. So, it is better to think about how work should be structured in your team

Branching Types

1. Long running
2. Short–lived
3. Remote Branches?

Long Running Branches

Every repository has at least one long running branch

1. They exist through the life-time of the project – main / master

Integration branches are often long running and they are designed to mirror the “stages” in a development life cycle – develop / test / staging / production / hotfix

- A common convention of an integration branch is
  1. Typically, commits are not added directly to these branches
  2. Commits to these branches only happen through merge or rebase

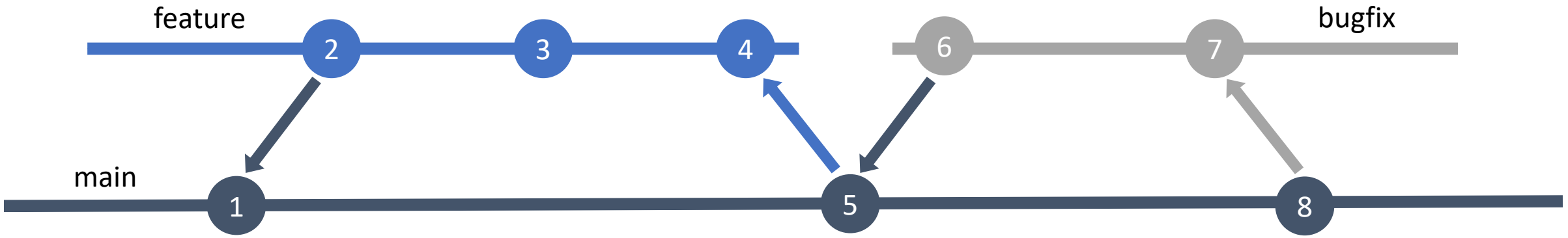
Short-Lived Branches

1. These branches are created for new features, bug fixes, experiments etc
2. Usually, short-lived branches are based on a long running branch
3. Generally, these branches are deleted after an integration (merge / rebase)

Popular Branching Strategies

GitHub Flow

- Very lean and simple: only one long running branch (main/master) and feature branches

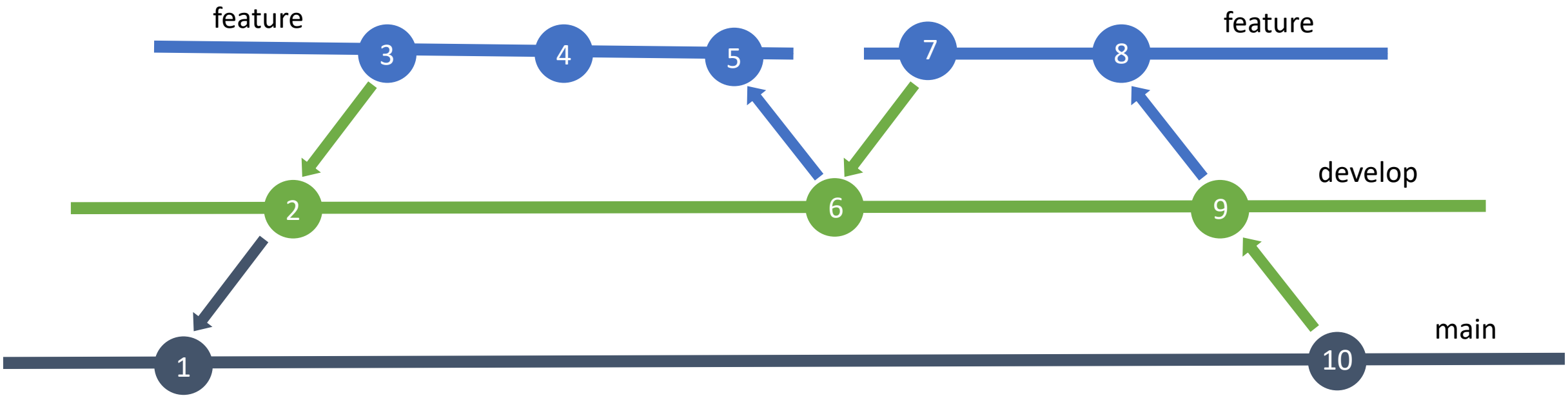


Popular Branching Strategies – GitFlow

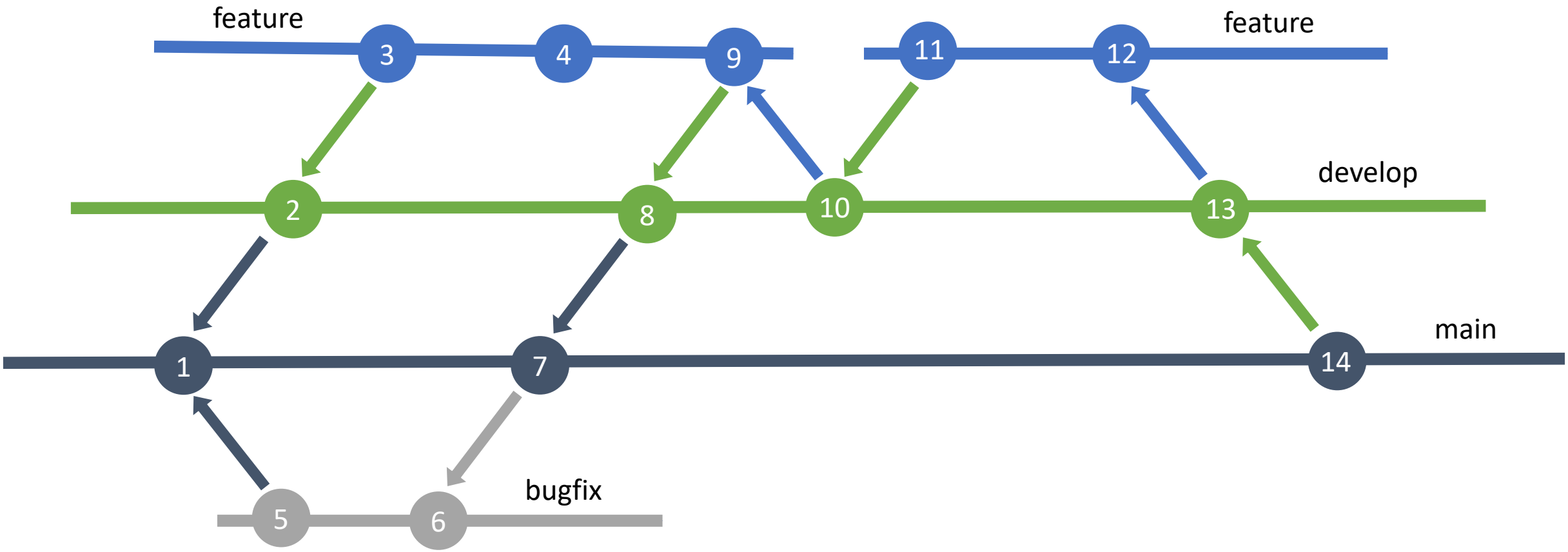
GitFlow

1. Main – reflection of current state of production
2. Feature branches are created from develop and feature branches merge back into develop
3. Develop is also the starting point of any new releases
4. Open a release branch, commit any bug fixes. Once confidence merge it back to main. Add tags to release commit on main and close the release branch

Popular Branching Strategies – GitFlow



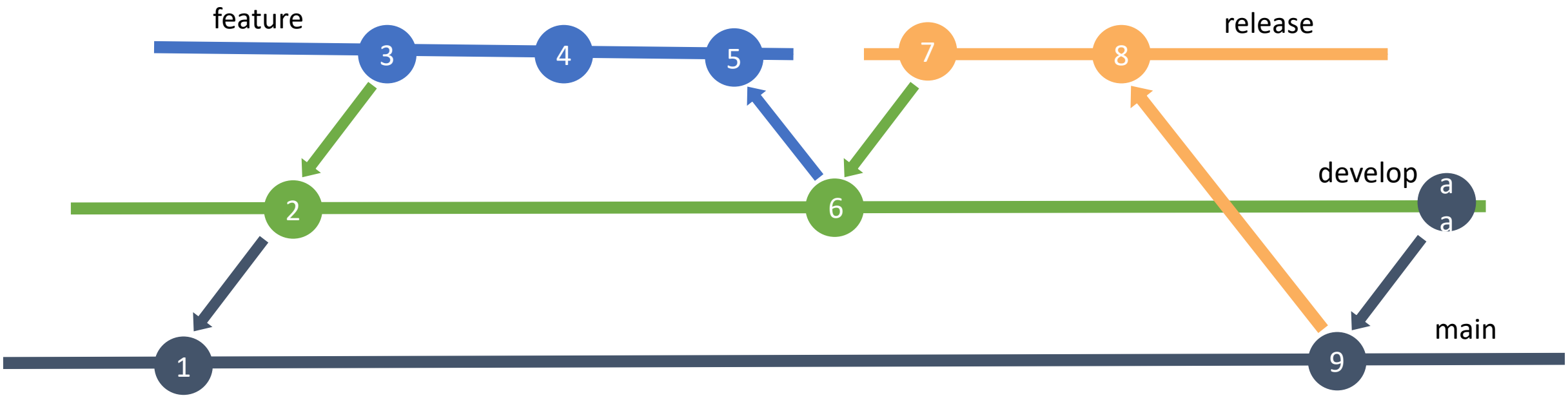
Popular Branching Strategies – GitFlow



Popular Branching Strategies

1. GitFlow
    1. More structure and rules
    2. Long-running: main and develop branches
    3. Short-Lived: features, releases, hotfixes
    4. GitFlow introduces a number of steps and tasks in the process

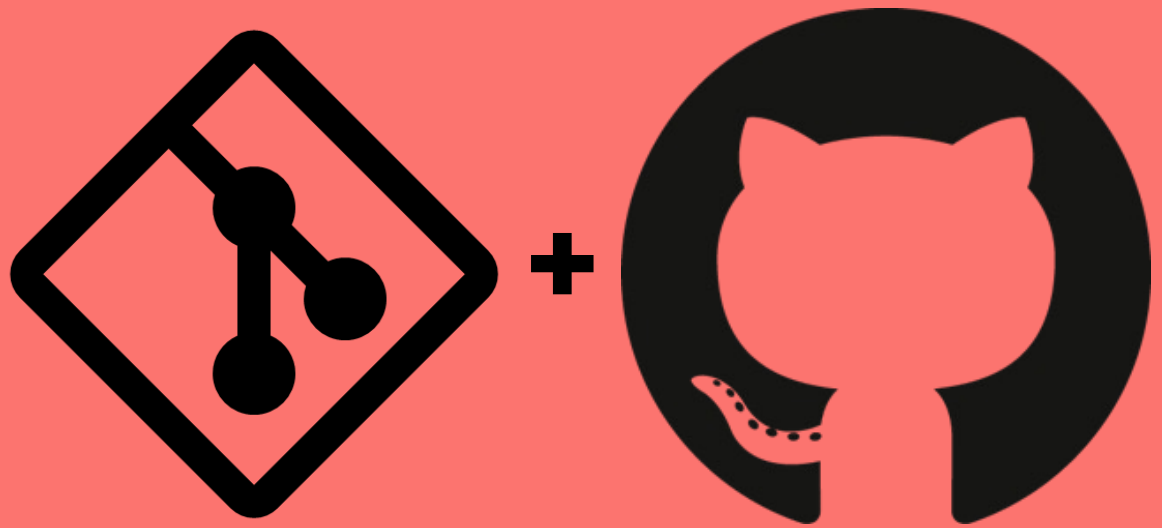
Popular Branching Strategies – GitFlow Release



Let us practice

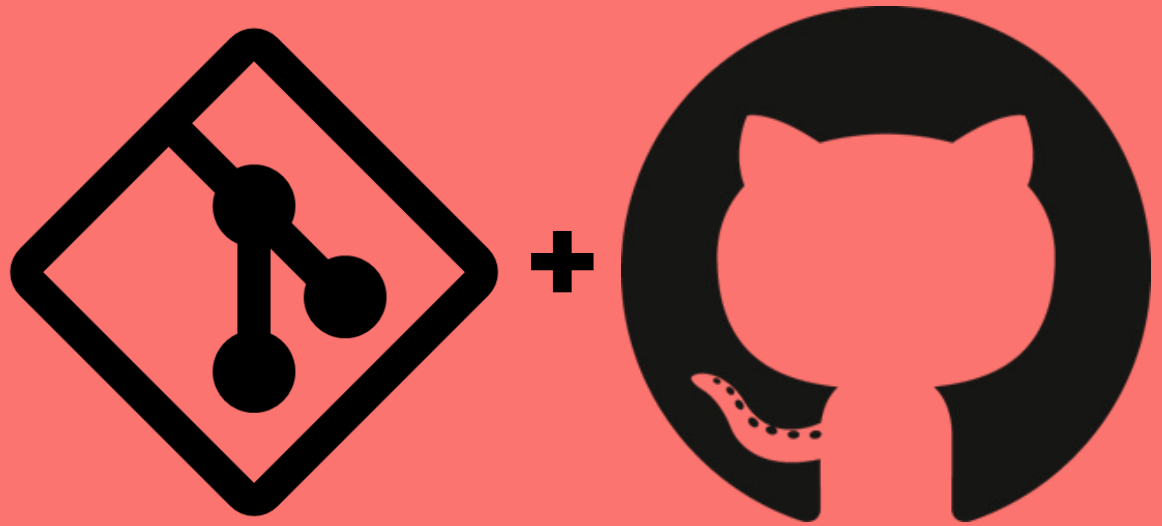
Git

In Practice



Git

In Practice



1. Git Branches
2. Branching Strategies