VCS, Git, Semantic Versioning, Git Flow, GitHub

Version Control Management and Team Collaboration Tools

Contents

- What is Version Control Systems?
- What is Git?
- Semantic Versioning
- Git Flow
- GitHub

What is Version Control Systems?

Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.

VCS Types

Local Version Control System

No team collaboration

Centralised Version Control System

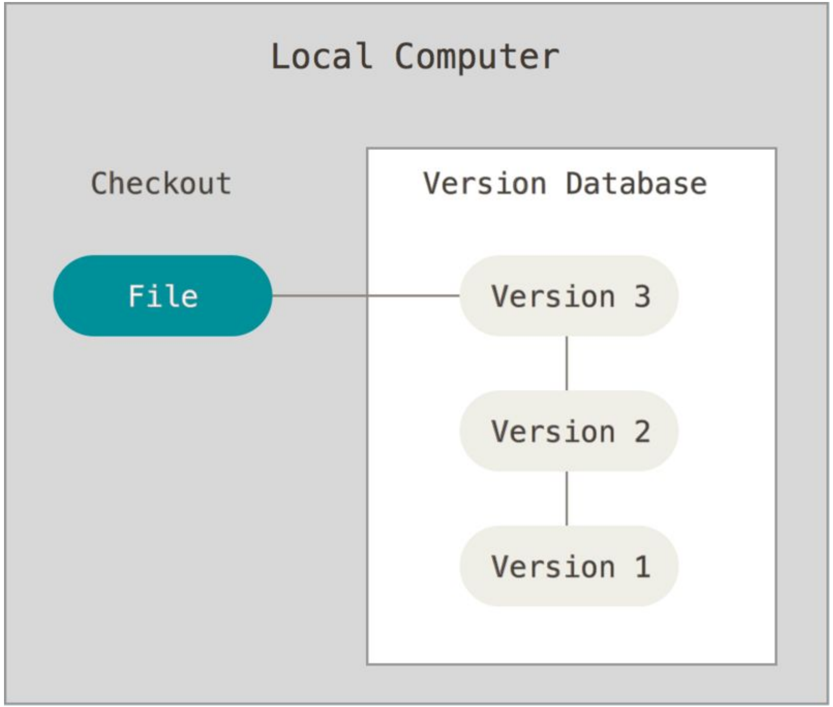
Single Point of failure

Tools: CVS, Subversion, and Perforce

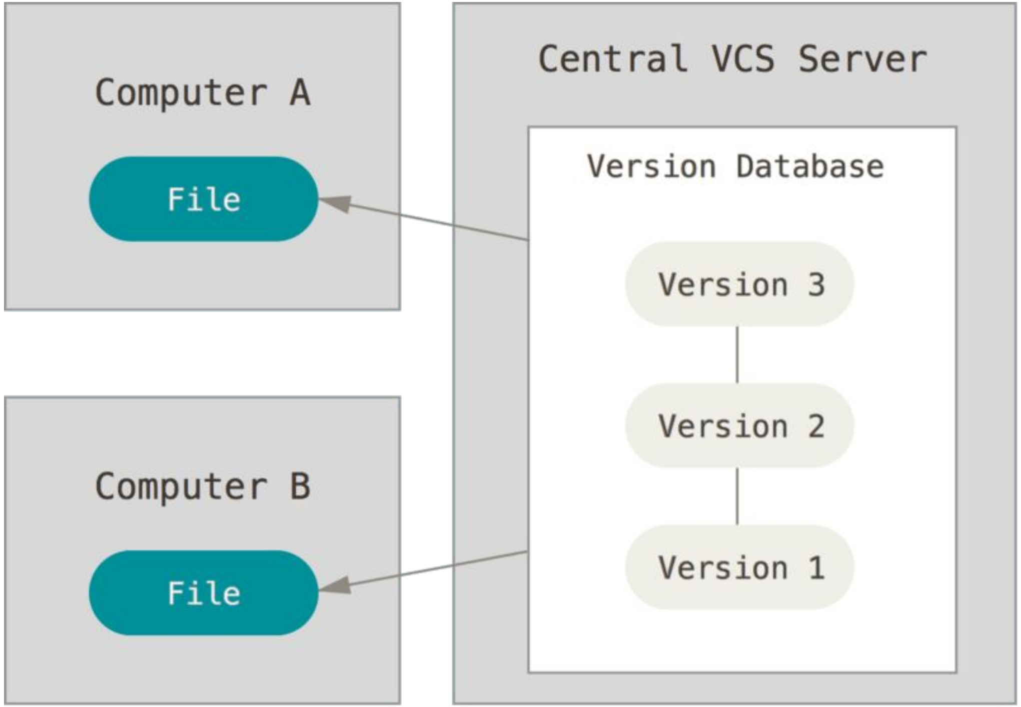
Distributed Version Control System

Tools: Git, Mercurial, Bazaar or Darcs

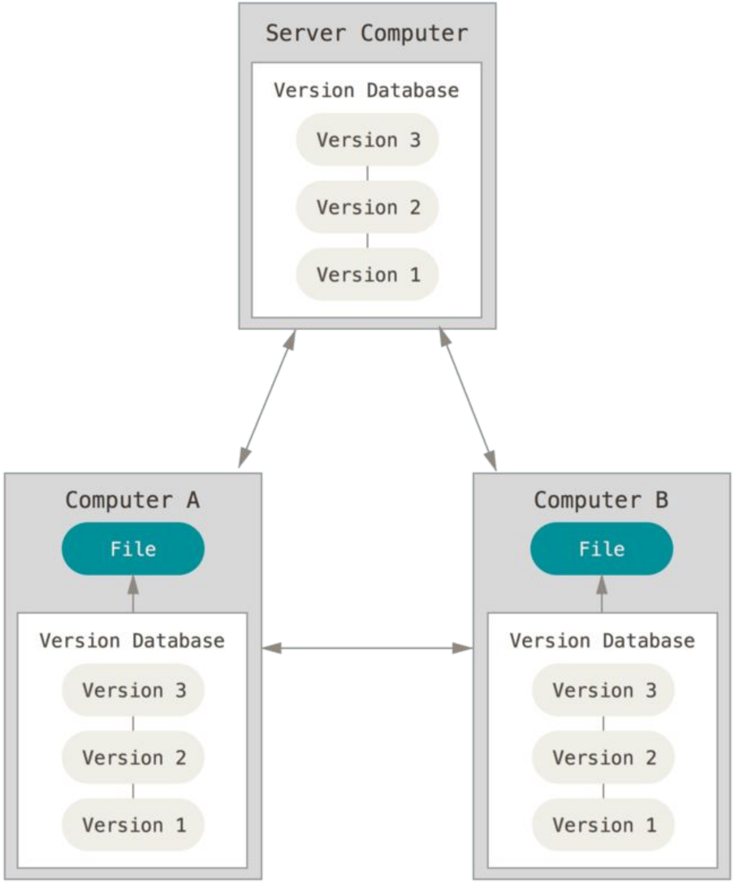
Local VCS


Local Computer
Checkout
File
Version Database
Version 3
Version 2
Version 1

Centralized VCS



Distributed VCS



Semantic Versioning

Given a version number MAJOR.MINOR.PATCH, increment the:

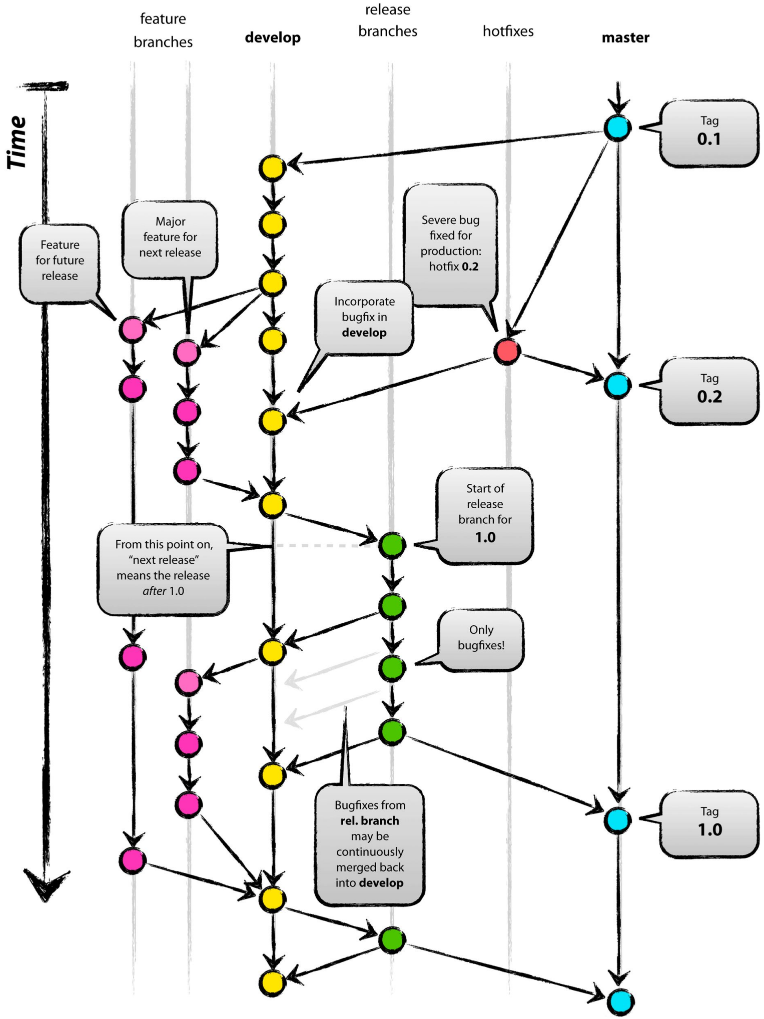
1. MAJOR version when you make incompatible API changes,
2. MINOR version when you add functionality in a backwards compatible manner, and
3. PATCH version when you make backwards compatible bug fixes.

Additional labels for pre-release and build metadata are available as extensions to the MAJOR.MINOR.PATCH format.

Git Flow

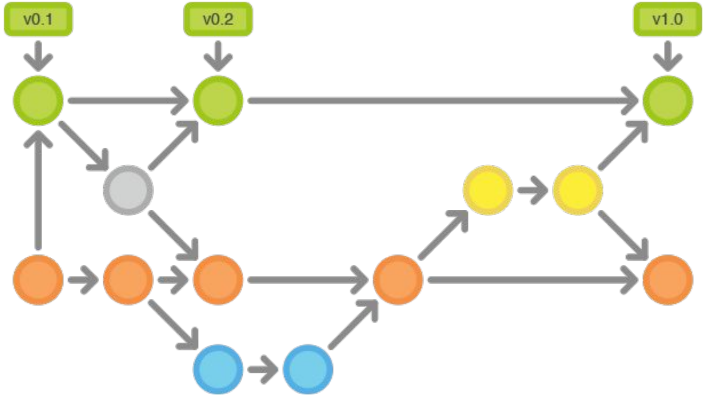
The Gitflow Workflow defines a strict branching model designed around the project release. This provides a robust framework for managing larger projects.

Git Flow Workflow

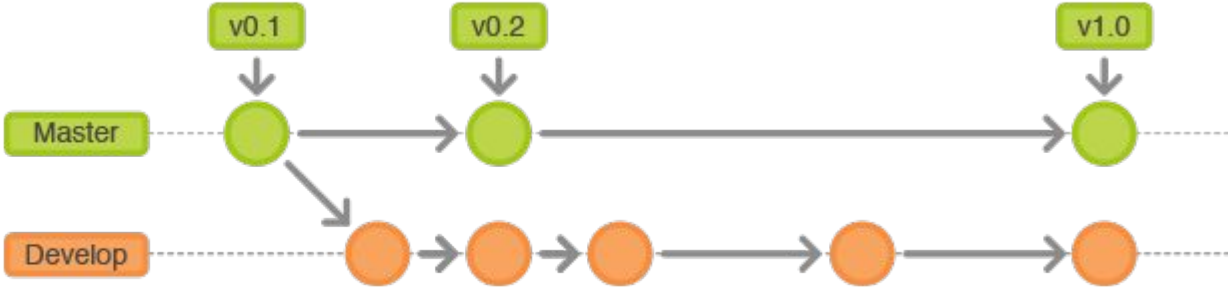

Time
feature branches
develop
release branches
hotfixes
master
Tag 0.1
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.0

How Git Flow Works

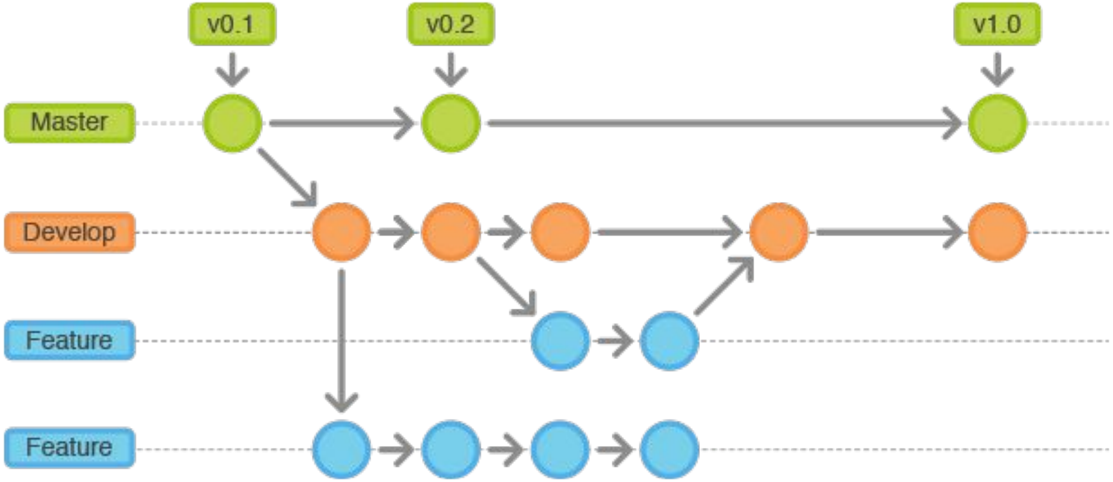
- Historical Branches
- Feature Branches
- Release Branches
- Maintenance Branches



Historical Branches



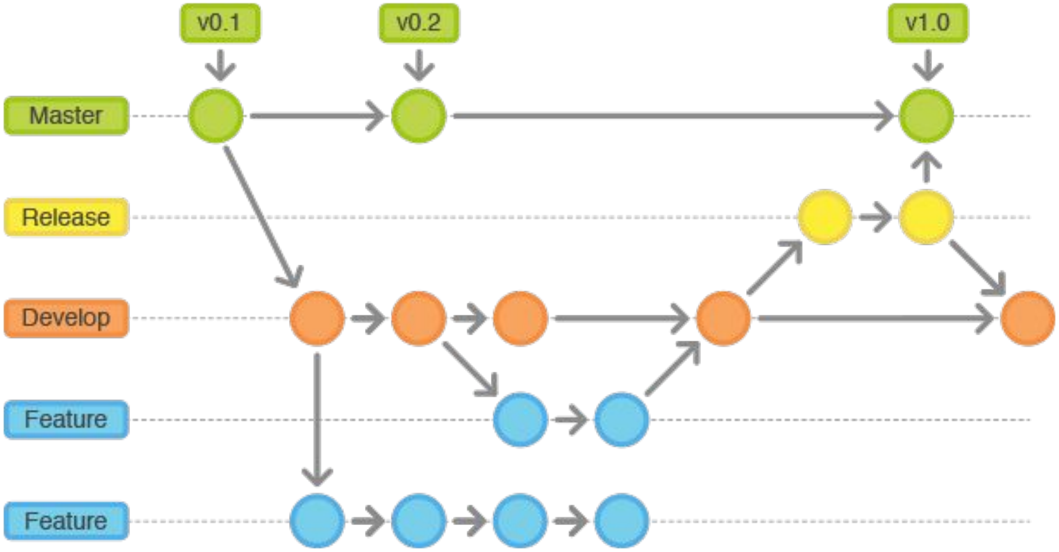
Feature Branches



Best Practices:

- May branch off: `develop`
- Must merge back into: `develop`
- Branch naming convention: anything except `master`, `develop`, `release-*`, or `hotfix-*`

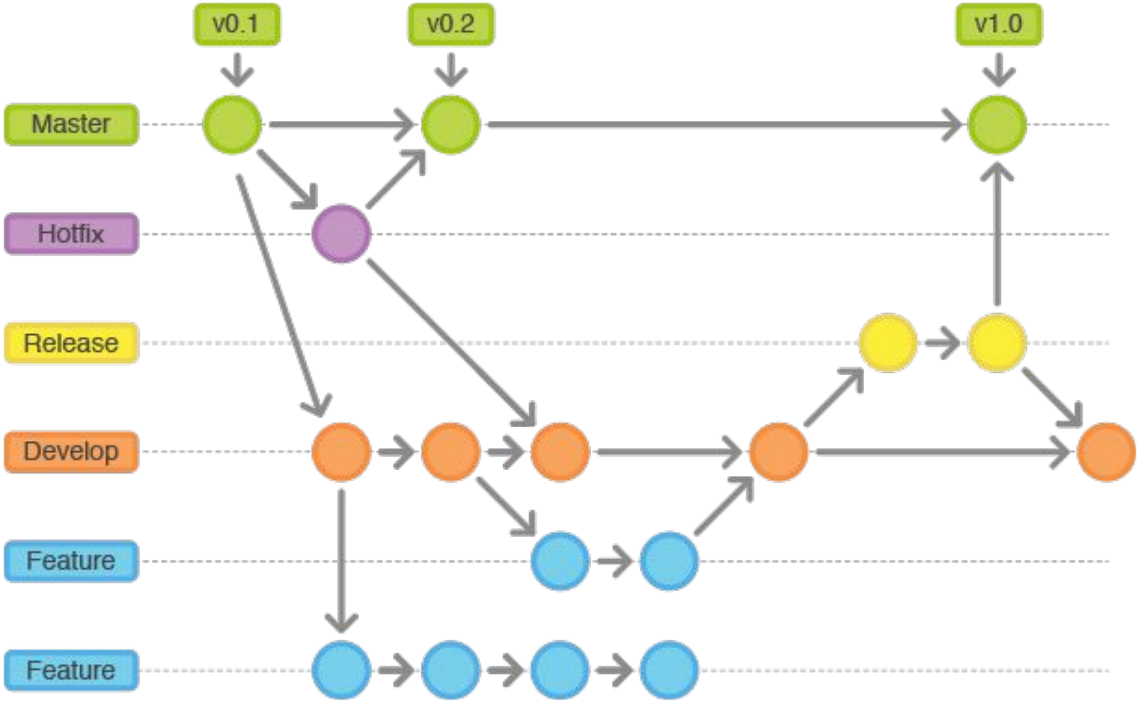
Release Branches



Best Practices:

- May branch off: `develop`
- Must merge back into: `develop` and `master`
- Tag: increment `major` or `minor` number
- Branch naming convention: `release-*` or `release/*`

Maintenance Branches



Best Practices:

- May branch off: `master`
- Must merge back into: `master` and `develop`
- Tag: increment `patch` number
- Branch naming convention: `hotfix-*` or `hotfix/*`

References

- https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow
- https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control
- https://jameschambers.co/writing/git-team-workflow-cheatsheet/
- https://leanpub.com/git-flow/read
- https://semver.org/