Git Branching and Pull Requests

Branches in Source Control Systems

In case of fire.

- → 1. git commit
- 2. git push
- 为 3. leave building

SoftUni Team Technical Trainers







Software University

https://softuni.bg

Have a Question?





Table of Content



- 1. Source Control Systems
- 2. Git
- 3. GitHub
- 4. Basic Git Commands
- 5. Git Conflicts
- 6. Branching Concepts and Branching in Git
- 7. Git Branching Commands
- 8. Branching Strategies
- 9. Common Git Branching Strategies
- 10. Pull Requests in GitHub
- 11. The Pull Request Process





Source Control Systems

Version Control System

Software Configuration Management (SCM)

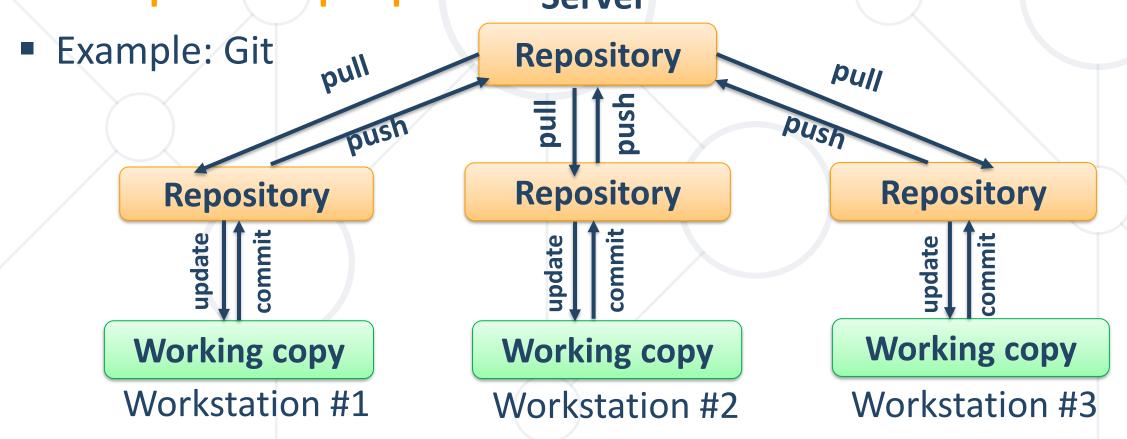


- Version Control System ≈ Source Control System
 - Tool for managing the changes during the development
 - A repository keeps the source code and other project assets
 - Keeps a full history of all changes during the time
 - Change log shows who, when and why changed what
 - Solves conflicts on concurrent changes
 - Allows reverting of old versions
- Popular source control systems
 - Git distributed source control (hierarchical)
 - Subversion (SVN) central repository (centralized)

Distributed Version Control (DVC)



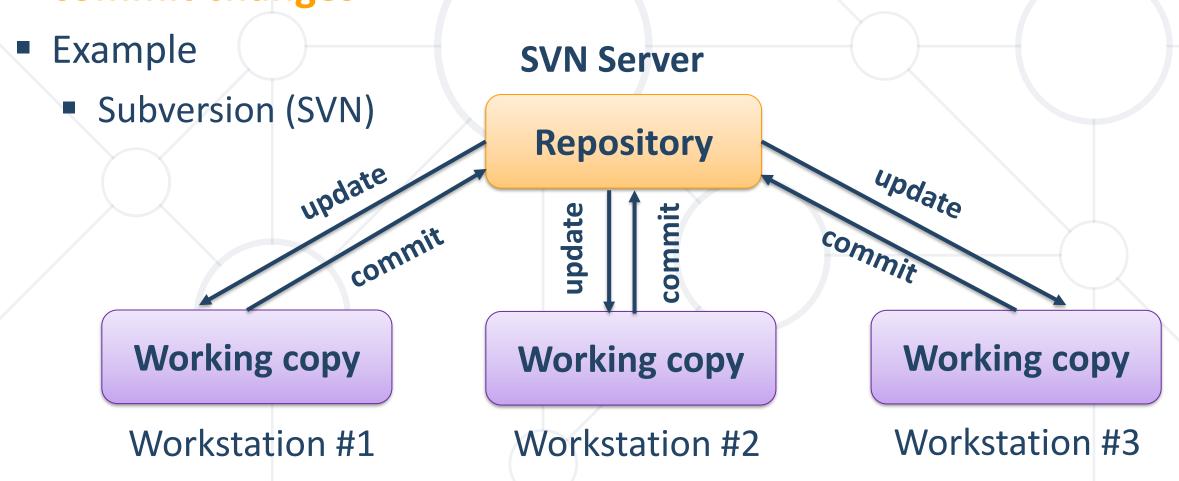
- Unlike a CVC system, a DVC doesn't have a single point of failure
- Developers clone repositories on their DVC workstations, creating multiple backup copies
 Server



Centralized Version Control (CVC)



 A CVC system relies on a central server where developers commit changes





What is Git?







- The most popular source control in the world
- Free open-source software
- Works with local and remote repositories
- Runs on Linux, Mac OS and Windows

GitHub

- Social network for developers
- Free project hosting site with Git repository



Vocabulary



- Repo (repository)
 - Holds the project in a remote server
- Branch
 - Parallel development path (separate version of the project)
- Merge branches
 - Merge two versions of the same projects

Clone

- Download a local copy of the remote project
- Commit
 - Saves a set of changes locally
- Pull
 - Take and merge the changes from the Remote

Push

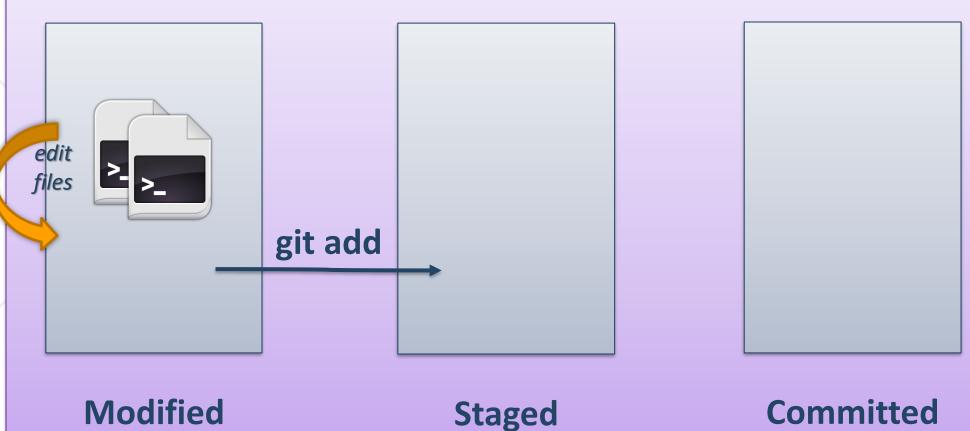
Software University

Send local changes to the Remote

Git Workflow



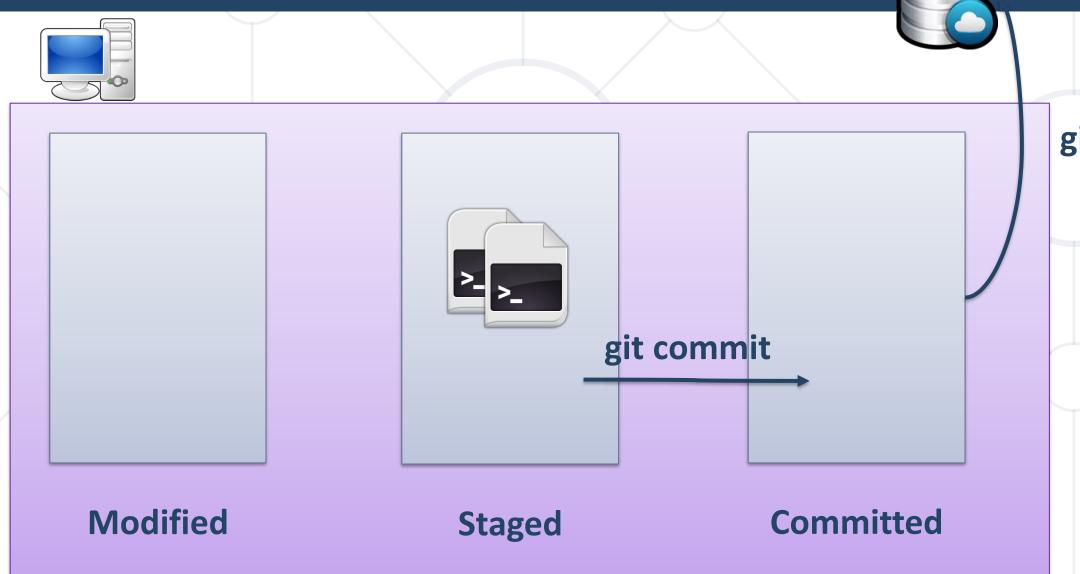




Committed

Git Workflow





git push



What is GitHub?





- Platform and cloud-based service, based on Git
- World's most used source code host
- Used for software development and version control
 - Free for open-source projects and small private projects
 - Paid plans for private repositories with advanced features

GitHub Desktop

Enables interacting with GitHub using a GUI instead of the command line or a web browser



GitHub Features



- Access control
- Bug tracking (Issue tracker)
- Continuous Integration (Actions)
- Wiki pages (Documentation)
- Software feature request
- Task management
- Project board (Kanban style)
- Etc.



Basic Git Commands

Clone → Modify → Add → Commit → Push

Basic Git Commands



Clone an existing Git repository

```
git clone [remote url]
```

Fetch and merge the latest changes from the remote repository

```
git pull
```

Prepare (add / select) files for a commit

```
git add [filename] ("git add ." adds everything)
```

Commit to the local repository

```
git commit -m "[your message here]"
```

Basic Git Commands



Check the status of your local repository (see the local changes)

```
git status
```

Create a new local repository (in the current directory)

```
git init
```

Create a remote (assign a short name for remote Git URL)

```
git remote add [remote name] [remote url]
```

Push to a remote (send changes to the remote repository)

```
git push [remote name] [local name]
```



Live Demo

Using Git Commands



Git Conflict

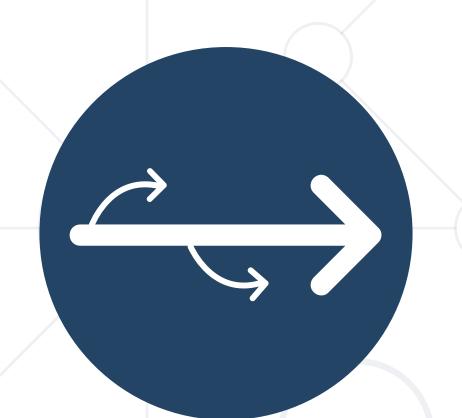


- Conflicts generally arise when two or more people change the same file simultaneously
 - Or if a developer deletes a file while another developer is modifying it
- In these cases, Git cannot automatically determine what is correct
- Conflicts only affect the developer conducting the merge
- The rest of the team is unaware of the conflict



Live Demo

Git Conflict Scenario



Branching Concepts & Branching in Git

Creating and Merging Branches

What is Branching?



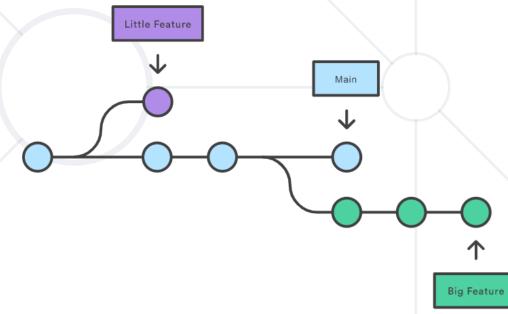
- Branches allow you to work on different parts of a project without impacting the main / master branch
 - Serve as an abstraction for the edit / stage / commit process
- Represent a way to request a brand new working directory, staging area, and project history
 - Any new commits are recorded in the history for the current branch
 - Without impacting the main branch until it's decided to integrate the changes
- You can switch between branches and work on different projects without them interfering with each other



Branches == Built-In Feature of Git



- Branching is available in most version control systems
 - In some version control systems it can be an expensive operation in both time and disk space
- In Git, branches are a part of the everyday development process
 Little Feature
- Git branches are an effective pointer to a snapshot of a developer's changes





Live Demo

Git Branches



View / Switch / Delete Branches

Local vs. Remote Branches



Local branch

- Branch in your local Git repo
- Changes tracked only locally

Remote branch

- Branch inside the remote repository (e.g. in GitHub)
- Upstream branch
 - Remote branch, connected to your local branch
 - When you push changes, they are sent to the upstream branch



Create a new local branch

```
git branch {branch-name}
```

Switch to certain existing branch

```
git switch {branch-name}
```

```
git checkout {branch-name}
```

Create a new branch and switch to it

```
git checkout -b {branch-name}
```



List all local and remote branches

```
git branch --all
git branch -a
```

List local together with the last commit message

```
git branch --verbose
git branch -vv
```

List all local and remote branches with the last commit message

```
git branch --all --verbose

git branch -a -vv
```



List local branches

```
git branch
```

Push to a new upstream (in a new remote branch)

```
git push --set-upstream origin {branch-name}
```

```
git push -u origin {branch-name}
```

Merge another branch in the active branch

```
git merge {branch-name}
```



Delete a local branch

```
git branch -d {branch-name}
```

Delete a remote branch

```
git push origin -d {branch-name}
```

Reapply commits on top of another base tip

```
git rebase {base-branch-name}
```

• Include specific commits without merging the entire branch

```
git cherry-pick {branch-name}
```

Git Squash



- Combining multiple commits into a single commit
- Useful for cleaning up commit history before merging a feature branch into the main branch
 - Makes commit history simpler and more understandable
- Usually applied during use of other commands, e.g.
 - git rebase
 - git merge
- Rewrites history and must be used with caution!



Branching Strategies

Strategy Simple Branch

What is a Branching Strategy?





- A strategy that programmers adopt while writing, merging and deploying code when using a VCS
- It is essentially a set of rules that developers can follow to stimulate how they interact with a shared codebase
- It enables teams to work in parallel to achieve faster releases and fewer conflicts
 - By creating a clear process when making changes to source control



Purposes of Branching Strategy



- Enhance productivity by ensuring proper coordination among developers
- Help organize a series of planned, structured releases
- Map a clear path when making changes to software through to production
- Maintain a bug-free code
- Enable parallel development

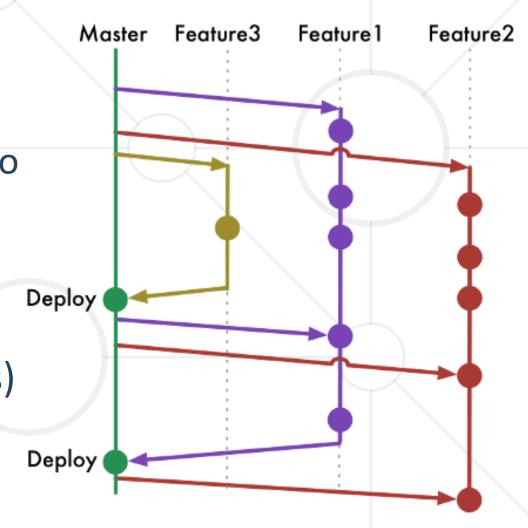


Common Git Branching Strategies

Trunk-Based Development



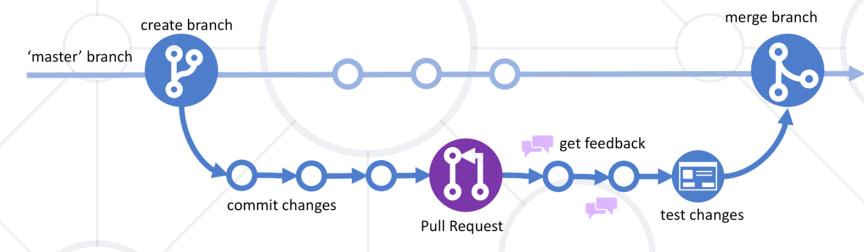
- Trunk-based development
 - "No branches" strategy
 - Developers integrate their changes into a shared trunk at least once a day
- Make smaller changes more frequently and commit directly into the trunk (without the use of branches)
 - This shared trunk should be ready for release anytime
- This strategy is suited to more senior developers



GitHub Flow



- GitHub Flow
 - Lightweight and flexible development workflow



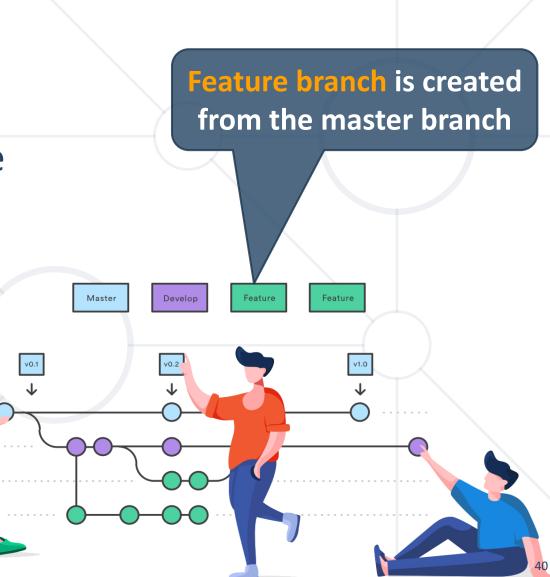
- Each feature is implemented in its own branch
- Keep the master code in a constant deployable state
- Before merging, each branch comes through a Pull Request and code review, then the changes are tested and integrated

GitFlow



GitFlow

- Enables parallel development
- Devs can work separately from the master branch on features
- This strategy contains separate and straightforward branches for specific purposes
 - Ideal to handle multiple versions of the product



GitFlow Branches



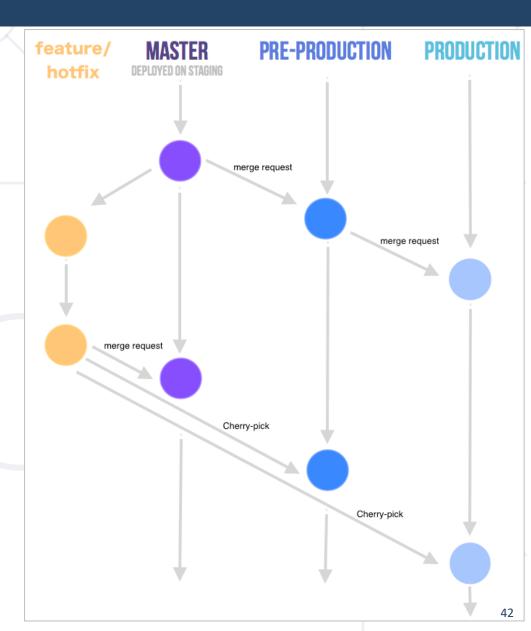
- Master (main)
 - Holds the active production code
- Release
 - Help prepare a new production release
- Hotfix
 - Hotfix branches arise from a bug that has been discovered and must be resolved
- Develop
- Feature
 - Develop new features

GitLab Flow



GitLab Flow

- Combines feature-driven development and feature branching with issue tracking
- Each feature is implemented in its own branch
- Developers work with the main branch right away
- The main branch is deployed to production (manually or auto)



Best Branching Strategy for Your Team



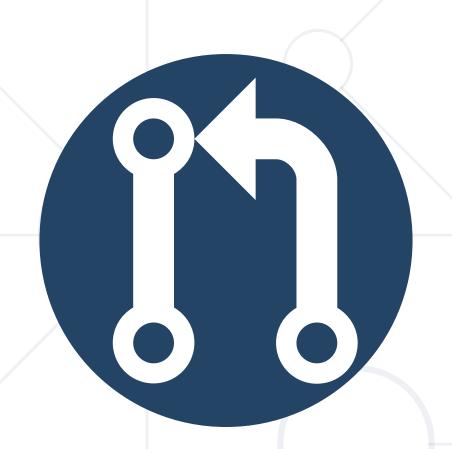
Product type and its release method	Team size	Collaboration maturity	Applicable mainstream b ranch mode
All	Small team	High	Trunk-based Dev
Products that support continuous de ployment and release	Middle	Moderate	GitHub-Flow and Trunk- based Dev
Products with a definite release wind ow and a periodic version release ca dence	Middle	Moderate	Git-Flow and GitLab-Flow with release branch
Basic platform products	Middle	Moderate	GitLab-Flow
Products demanding product quality and have a long maintenance cycle f or released versions	Large	Moderate	Git-Flow

No Standard for "Branching Strategy"



- Different organizations use different branching strategies for each different team / project
 - There is no "best branching strategy"
 - How to use branches highly depends on
 - The team
 - The project under development





Pull Requests in GitHub

Show Changes You've Pushed

Pull Requests in GitHub



- Pull requests
 - A mechanism for developers to notify their team members that they have completed a feature
- The pull request is more than just a notification—it's a code review process, including discussions on the proposed feature
- If there are any problems with the changes, teammates can post feedback in the pull request
- This activity is tracked directly inside of the pull request

Pull Requests



- Pull requests let you tell others about changes you want to push to a branch in a repository on GitHub
- Once a pull request is opened, you can discuss and review the potential changes with collaborators
- You can create pull requests on GitHub.com, in GitHub
 Desktop or in other GitHub tools

Branch Protection Rules



- You can create a <u>branch protection rule</u> in a repo
 - Can be created for
 - A specific branch
 - All branches
 - Any branch that matches a name pattern you specify with fnmatch syntax
- For example, you can create a branch protection rule to prevent contributors from merging to a branch without an approved pull request



The Pull Request Process

Notify Team for a Completed Feature

Pull Request: The Process









Senior Developer

Create a feature branch

Make and commit changes

Open a pull request

Resolve merge conflicts

Request a review

Discussion Give feedback / comment

Approve and merge



Live Demo

Creating a Pull Request

Summary



- Git == distributed source control system
- GitHub == platform and cloud-based service, based on Git
- Branch == a new separate version of the main repository
- Build your branching strategy from several main concepts
- Pull requests let you tell others about changes you've pushed to a branch in a repository on GitHub
- Conflicts may appear when merging branches





Questions?

















SoftUni Diamond Partners







Coca-Cola HBC Bulgaria









Решения за твоето утре













Trainings @ Software University (SoftUni)



- Software University High-Quality Education,
 Profession and Job for Software Developers
 - softuni.bg, about.softuni.bg
- Software University Foundation
 - softuni.foundation
- Software University @ Facebook
 - facebook.com/SoftwareUniversity







License



- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is copyrighted content
- Unauthorized copy, reproduction or use is illegal
- © SoftUni https://about.softuni.bg/
- © Software University https://softuni.bg

