SCM, VCS and Git

León Jaramillo

softserve

What is SCM?
What is a VCS?
What is Git?

What will we talk about?

- Source Control Management (SCM)
- Version Control Systems (VCS)
- VCS Types
- Git
- Branching Strategies

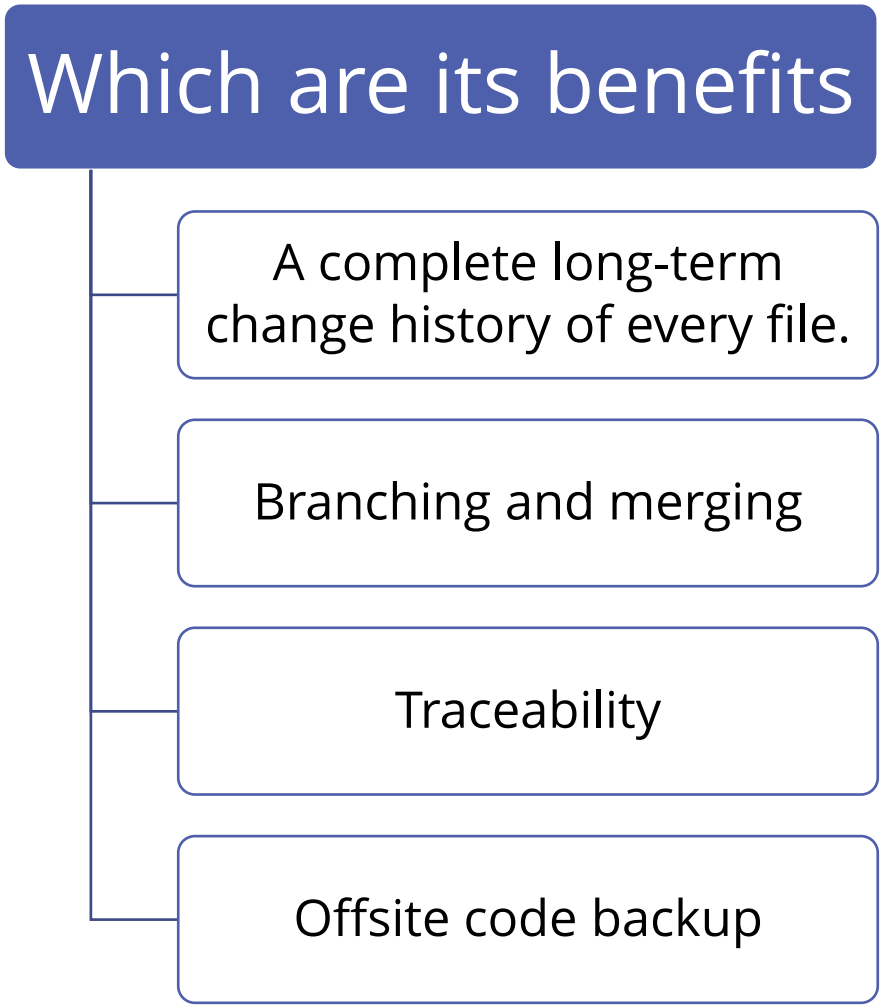
What is Source Code Management (SCM)

Source Code Management (SCM) is a **set of practices** that programmers uses to manage source code:

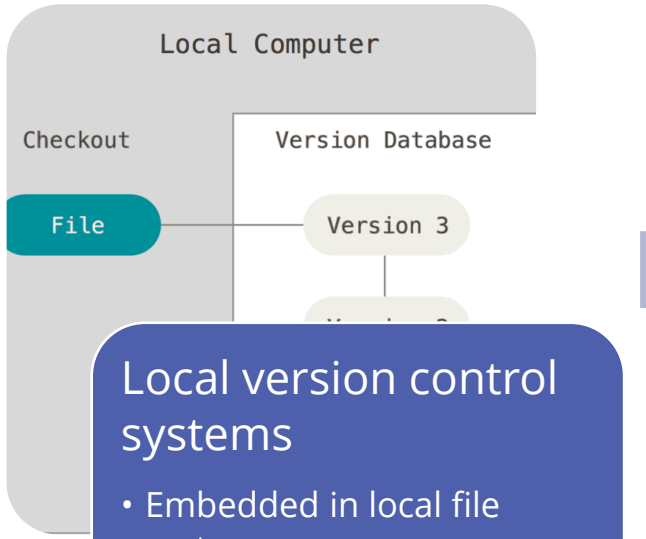
- Backup of assets
- Synchronization of work
- Undoing changes
- Tracking changes
- Code ownership
- Branching and merging

What is Version Control Systems (VCS)

- **Version control** is tracking and managing changes to software code
- **Version control systems (VCS)** are software tools that help software teams to perform version control
- VCS aim to **accelerate** the software development process
- VCS keeps **track** of every modification done by the development team on their codebase

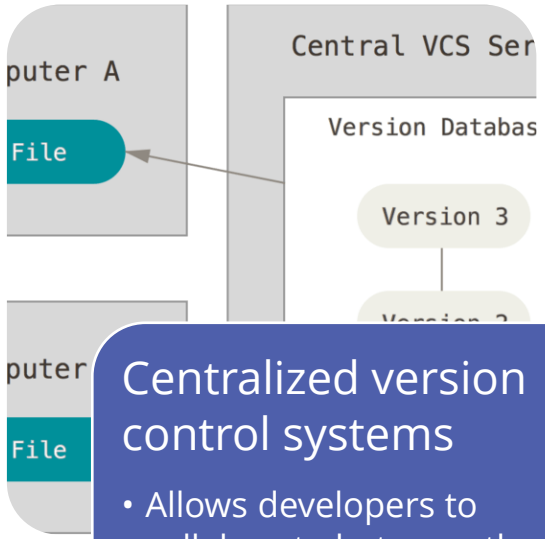


Where did the VCS come from (shaping their types)?



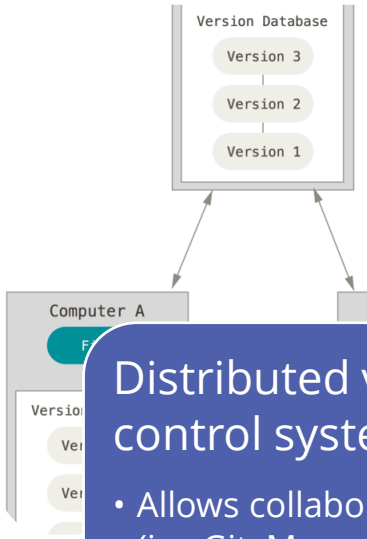
Local version control systems

- Embedded in local file system.
- Often patch based (i.e, RCS).
- Prone to data loss.



Centralized version control systems

- Allows developers to collaborate between them (i.e, CVS, SVN, Perforce).
- One single server hosts the whole repo.
- Makes a single point of failure + network overhead.



Distributed version control systems

- Allows collaboration too (i.e, Git, Mercurial).
- Each client has a copy of the repo in their machine.
- Allows several ways to collaborate, in addition to different workflows.

Images from git-scm.com

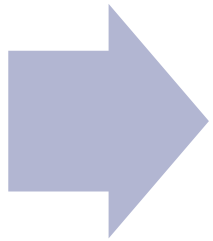
softserve

What VCS can we find out there?

Tool	Architecture	Conflict resolution	Development status	URL
Git	Distributed	Merge	Active	https://git-scm.com/
Mercurial	Distributed	Merge	Active	https://www.mercurial-scm.org/
SVN	Centralized	Merge or lock	Active	https://subversion.apache.org/
CVS	Centralized	Merge	Maintenance only	https://www.nongnu.org/cvs/

And more...: https://en.wikipedia.org/wiki/List_of_version-control_software

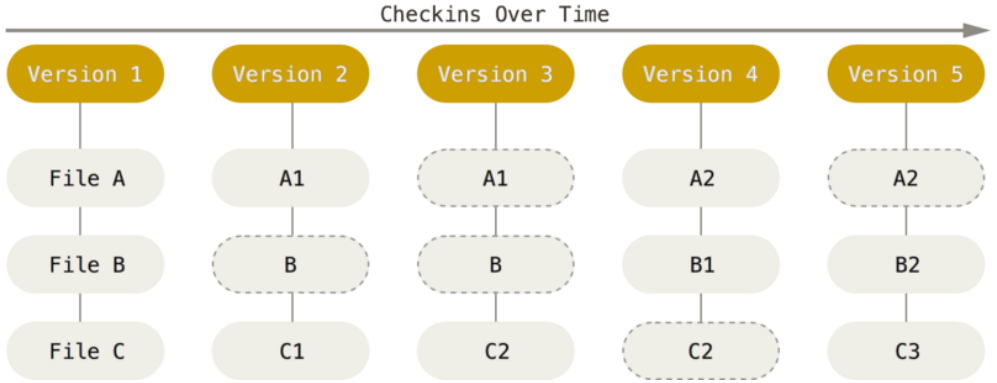
So, where are GitHub, GitLab or Bitbucket?

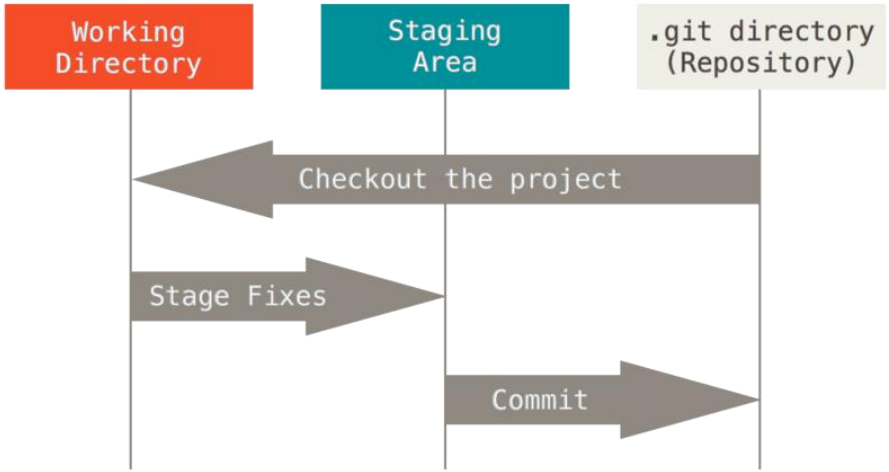


They're not VCS, but VCS hosting services.

What is Git?

- **Git** is a Distributed Version Control System (DVCS).
- It is based on **snapshots**, so every commit saves the state of the project at a given moment.
- Most of the operation done with Git are **local**.
- Everything in Git is **checksummed** using SHA-1.
- Locally, at Git, every files are in one of three states: **modified**, **staged** or **committed**.

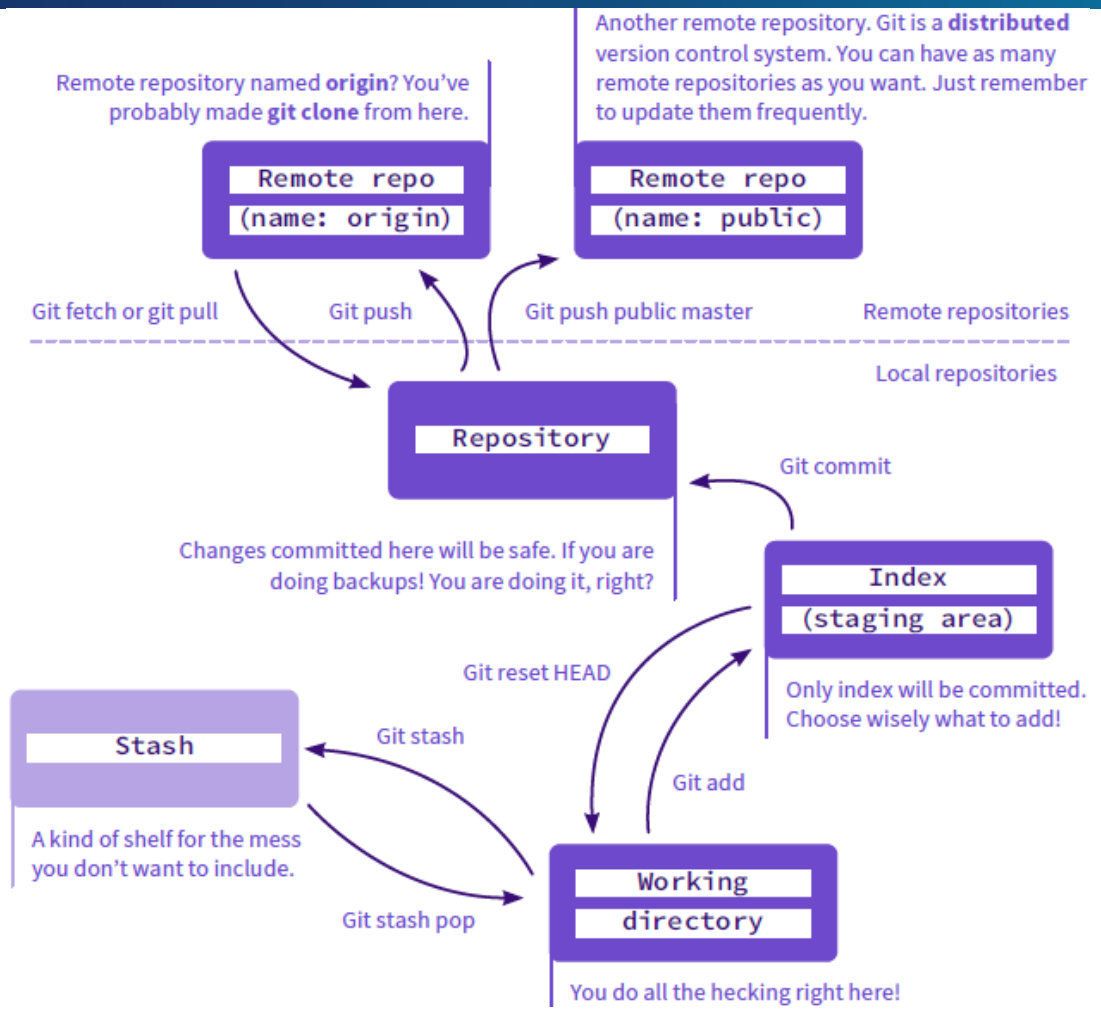




So, what is GitHub?

- **GitHub** (https://github.com/) is a hosting service for software development and version control using Git.
- Comprises different **services** besides DVCS, notably, GitHub Actions or GitHub Pages.
- Nowadays is owned by **Microsoft**.
- Hosts **millions** of public and private **repositories**.
- It is just one of several **alternatives**.

The zoo of working areas



From: GitLab Cheat Sheet

softserve

Let's try a little bit...

`git clone [repo-URL]`

Clones remote repository to the current local directory

`git status`

Shows modified and staged files

`git add [.|file-name]`

Adds a file or files to the staging area

`git commit –m "[message]"`

Commits the staged files

`git push [alias] [branch]`

Pushes local commits to a remote branch

`git remote add [alias] [repo-URL]`

Adds a new remote repository

`git push [alias] [branch]`

Pushes content from the local to the remote repository in the desired branch

`git fetch [alias] [branch]`

Gets content from the desired remote branch to the local repository

`git merge [branch]`

Merges changes from the desired branch to the current branch

Let's try a little bit...

`git init`

initializes the current directory as a Git repository

`git diff`

Diff of what is changed but not staged

`git reset [file-name]`

Unstages the specified file, given that it's already at the staging area

`git branch`

Lists all the branches in your repository

`git branch [name]`

Creates a branch with the specified name

`git init [folder name]`

initializes a Git repository in a new folder

`git diff --staged`

Diff of what is staged but not committed

`git checkout [branch]`

Switches to another branch and

`git log`

Show all the commits in the local branch history

`git branch -d [branch]`

Removes the selected branch

Version Control best practices

- Commit often
- Commit related changes
- Do not commit half-done work
- Write good commit messages (making detailed notes if needed)
- Ensure you're working from latest version
- Review changes (and test) before committing
- Use branches
- Agree on a workflow (and branching strategy)

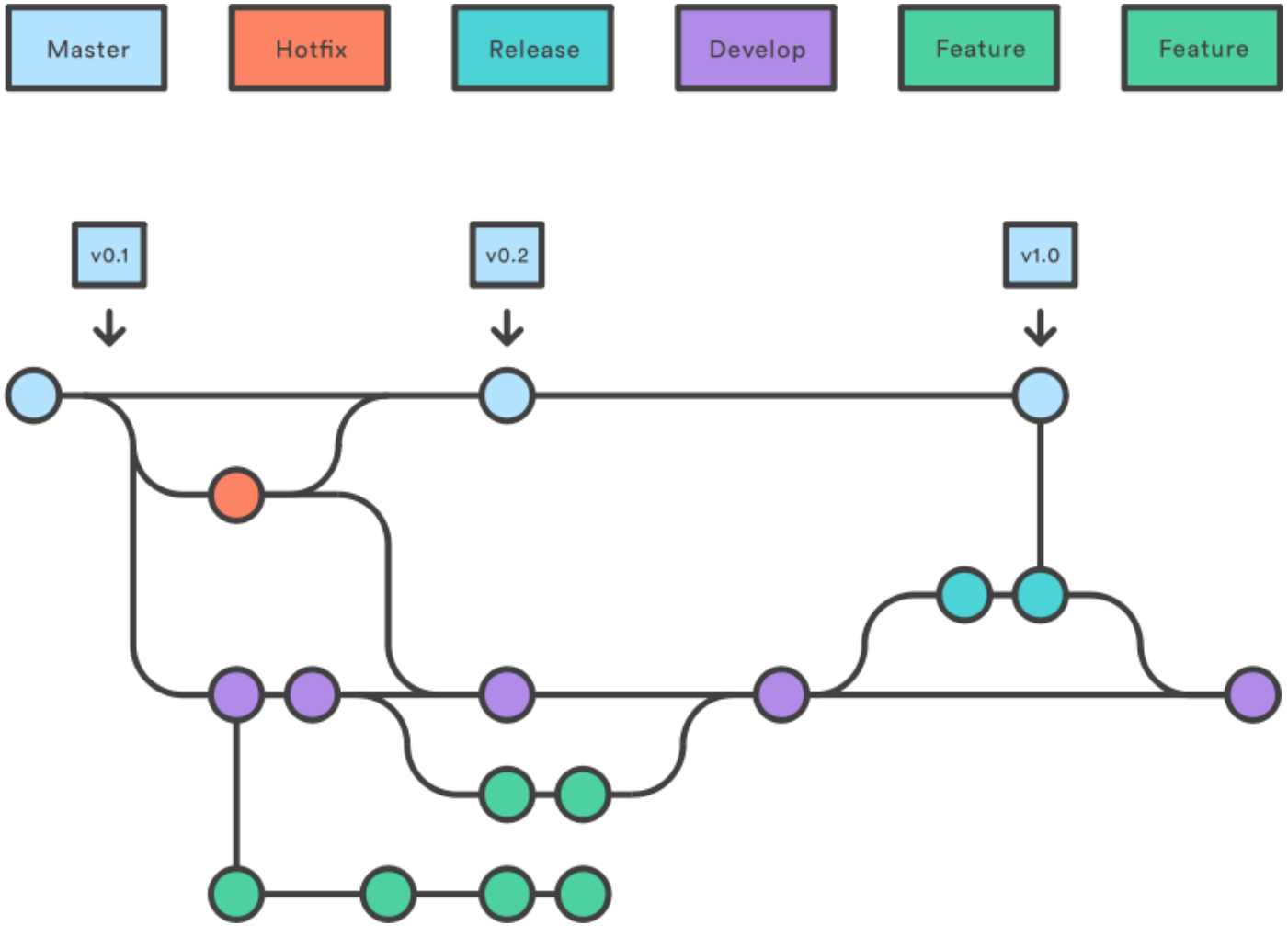
From: Attlasian and Git Tower

Git Branching Strategies

A branching strategy sets rules about how a team of developers write, merge and deploy code using a VCS

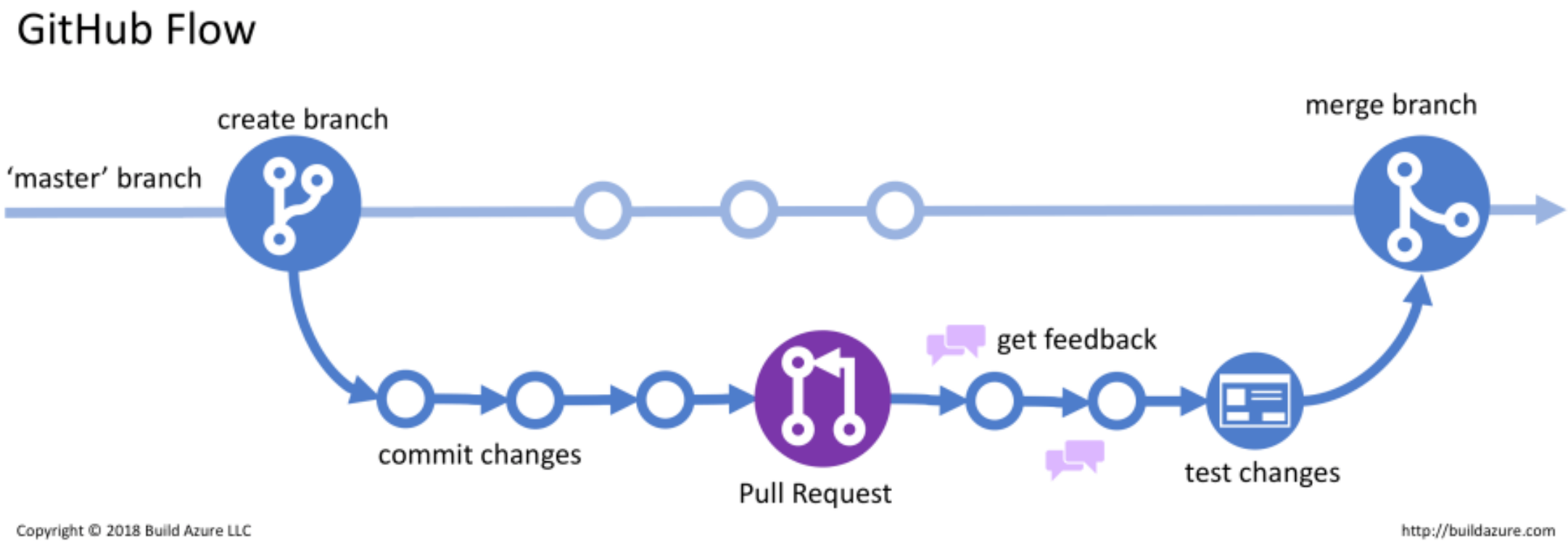
- GitFlow
- GitHub Flow
- Trunk-Based Development
- Scaled Trunk-Based Development
- *Custom workflows*

GitFlow

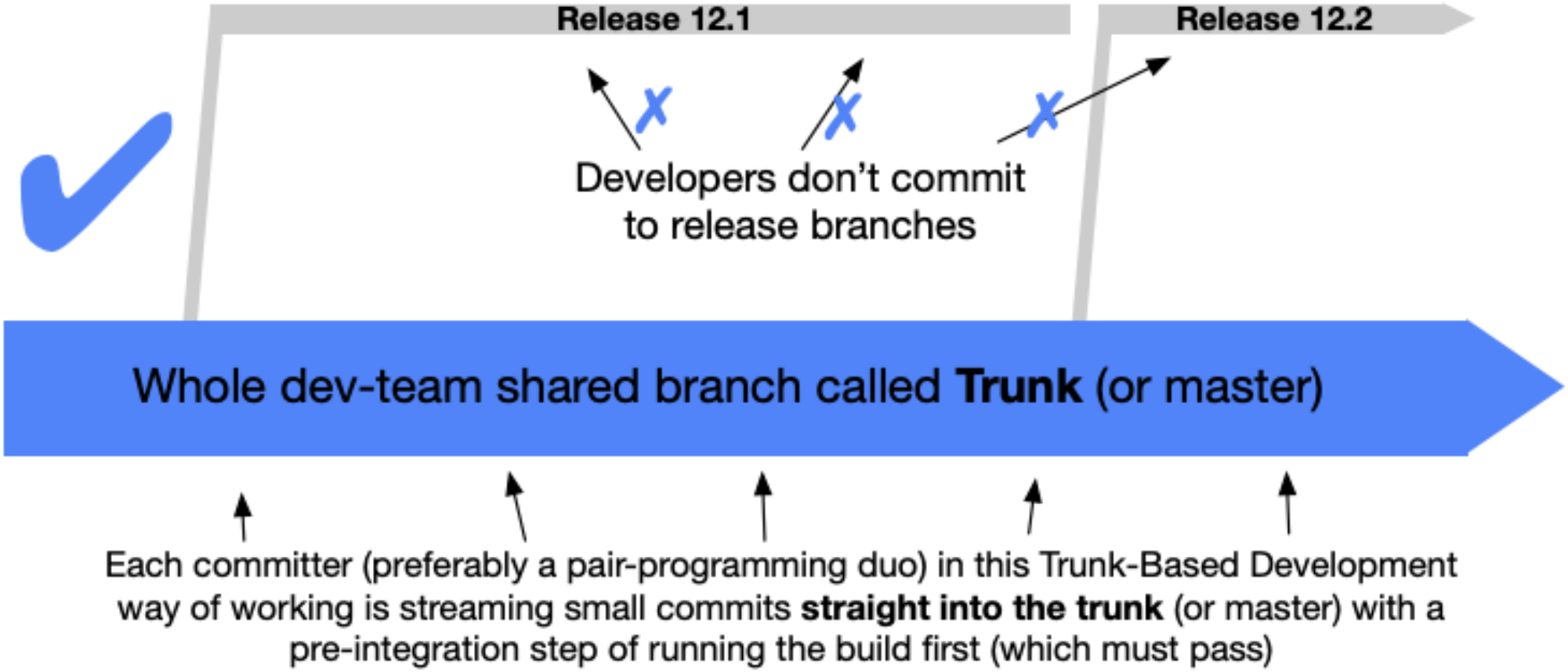


softserve

GitHub Flow

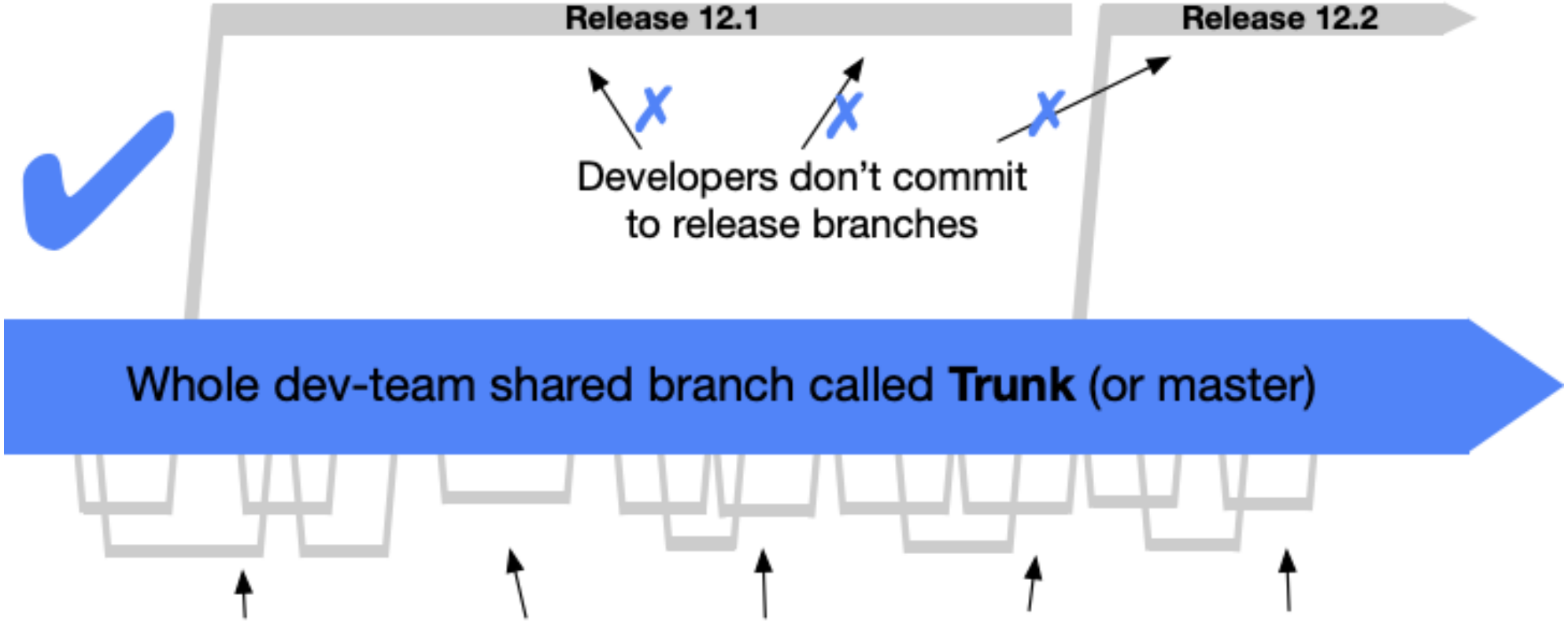

GitHub Flow
create branch
'master' branch
merge branch
get feedback
commit changes
Pull Request
test changes
Copyright © 2018 Build Azure LLC
http://buildazure.com

Trunk-Based Development (for small teams)



From: trunkbaseddevelopment.com

Scaled Trunk-Based Development



Trunk-Based Development at scale is best done with **short-lived feature branches**: one person over a couple of days (max) and **flowing through Pull-Request style code-review & build automation** before "integrating" (merging) into the trunk (or master)

From: trunkbaseddevelopment.com

Useful Links

- Git and GitHub for Beginners - Crash Course: https://www.youtube.com/watch?v=RGOj5yH7evk
- Git Branches Tutorial: https://www.youtube.com/watch?v=e2IbNHi4uCI
- Git for Professionals Tutorial: https://www.youtube.com/watch?v=Uszj_k0DGsg
- Advanced Git Tutorial: https://www.youtube.com/watch?v=qsTthZi23VE
- Patterns for Managing Source Code Branches: https://martinfowler.com/articles/branching-patterns.html
- DevOps tech: Version control: https://cloud.google.com/architecture/devops/devops-tech-version-control
- Git Cheat Sheet (GitLab): https://about.gitlab.com/images/press/git-cheat-sheet.pdf


COLOMBIA
DC
Thanks! Any question?
softserve