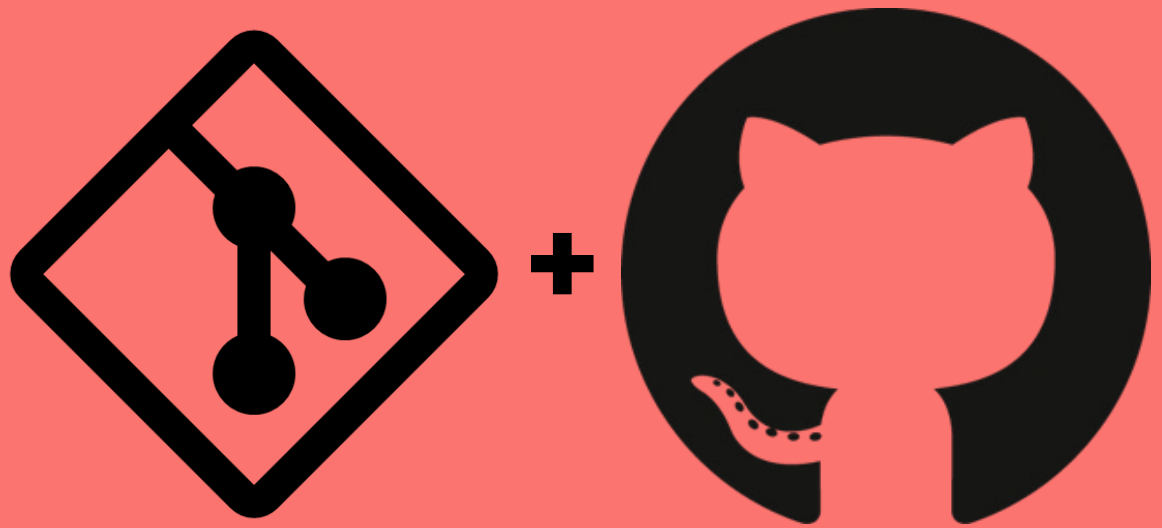
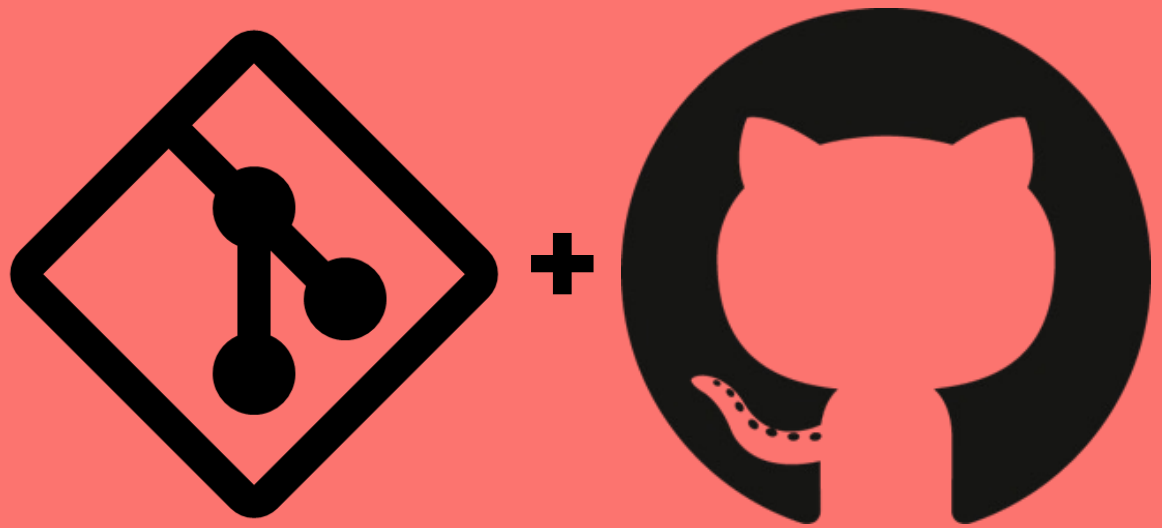
Git

In Practice



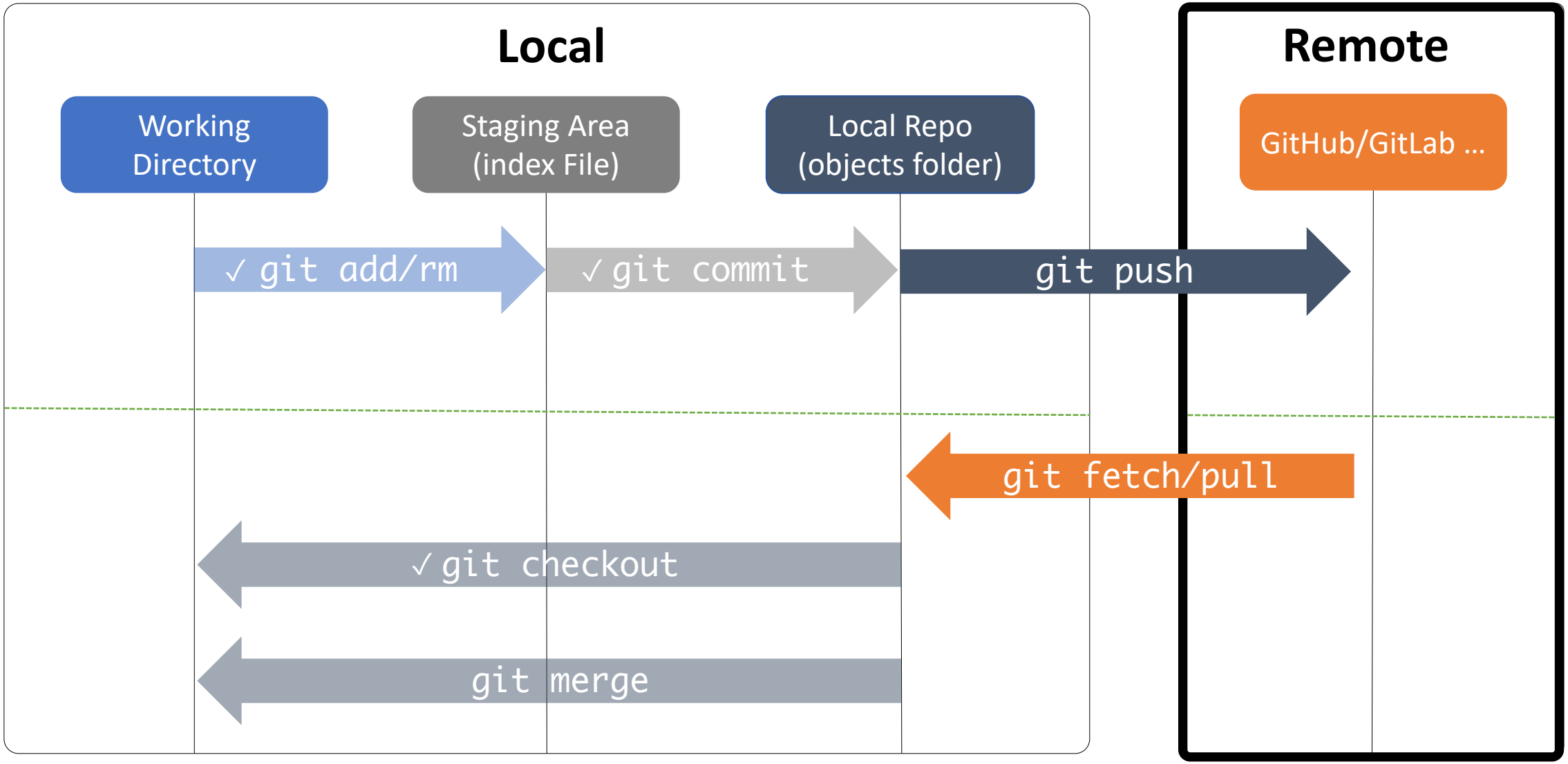
Git

In Practice



1 Remote Repo & Branch Types

The Remote State



Remote Repository

1. A remote repository is a common repository that all team members use to exchange code changes

2. It typically does *not* provide a file tree of the project's current state

3. Since the repository has no working directory it is considered *bare*

Remote Repository

4. A remote repository has the data of the `.git` directory – nothing else

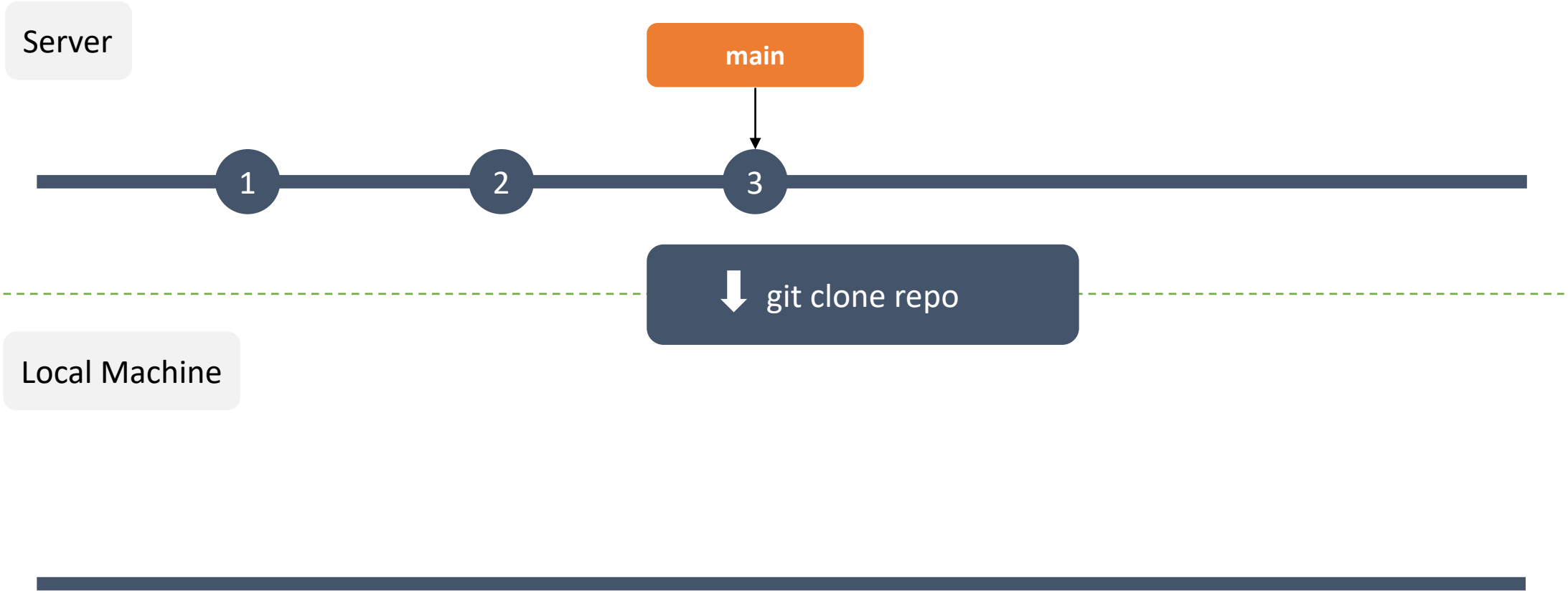
5. With the Git data one can reconstruct the entire folder structure of a working tree

The Protocols

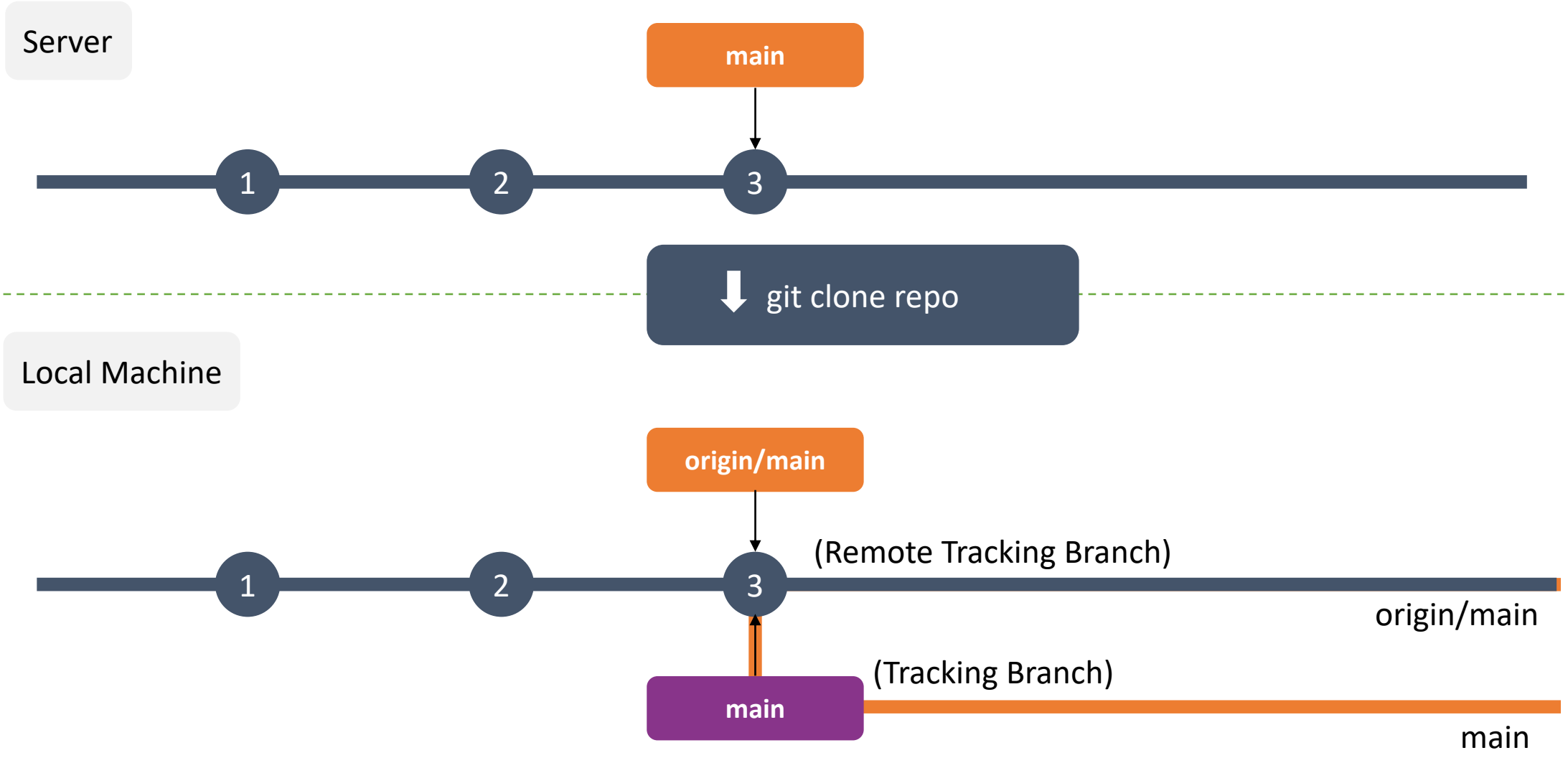
Git can communicate using four types of protocols

1. Local – NFS
2. SSH – commonly used for Github repositories
3. HTTP – uses https and is similar to SSH/Git protocols
    a. Can support basic authentication using username and password
    b. Support for single URL similar to the Git protocol
4. Git – does not have inbuild authentication
    a. Generally paired with SSH or HTTPS

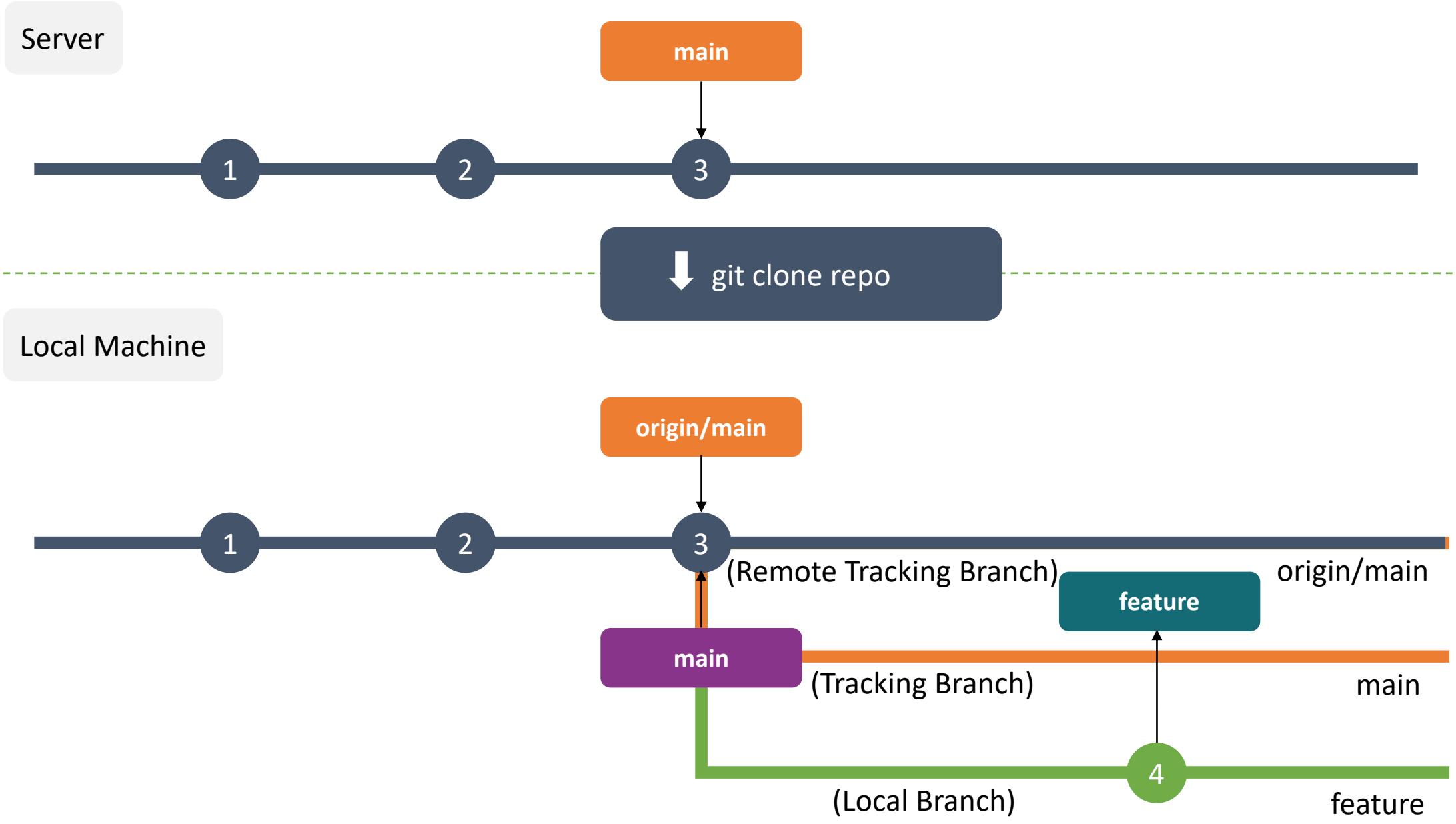
Branch Types



Branch Types



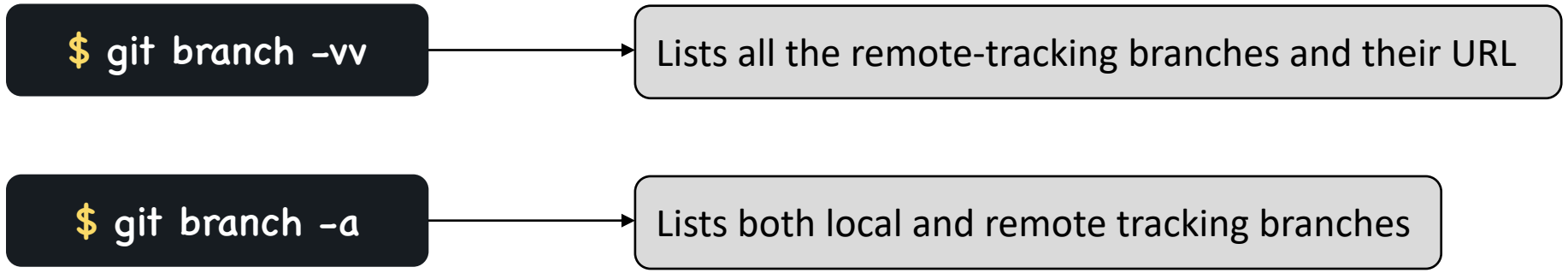
Branch Types



Branch Types – Local Branches

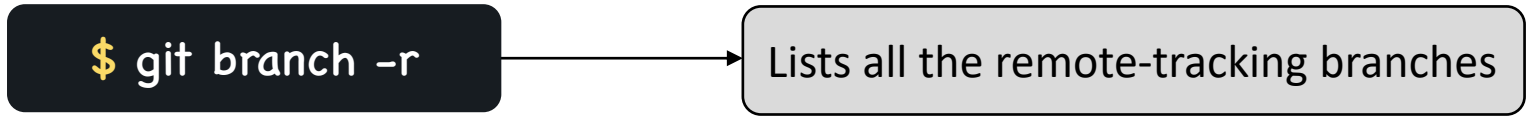
- Local Branches
  - Can be found under `.git/refs/heads/`
  - Tracking local branches – have a remote branch associated together with it

    Tracking branches are local branches that have a direct relationship to a **remote** branch.
  - Non-Tracking local branches – local branches that does not have any association



Branch Types – Remote Tracking Branches

- Can be found under `.git/refs/remotes/<remote>/`
- Local remote branches are remote-tracking branches
- Remote-tracking branches are references to the state of remote branches
- The local references of remote tracking branches cannot be moved
- They’re used to link what you’re working on locally with what’s on the remote
- You cannot switch to or checkout a remote tracking branch



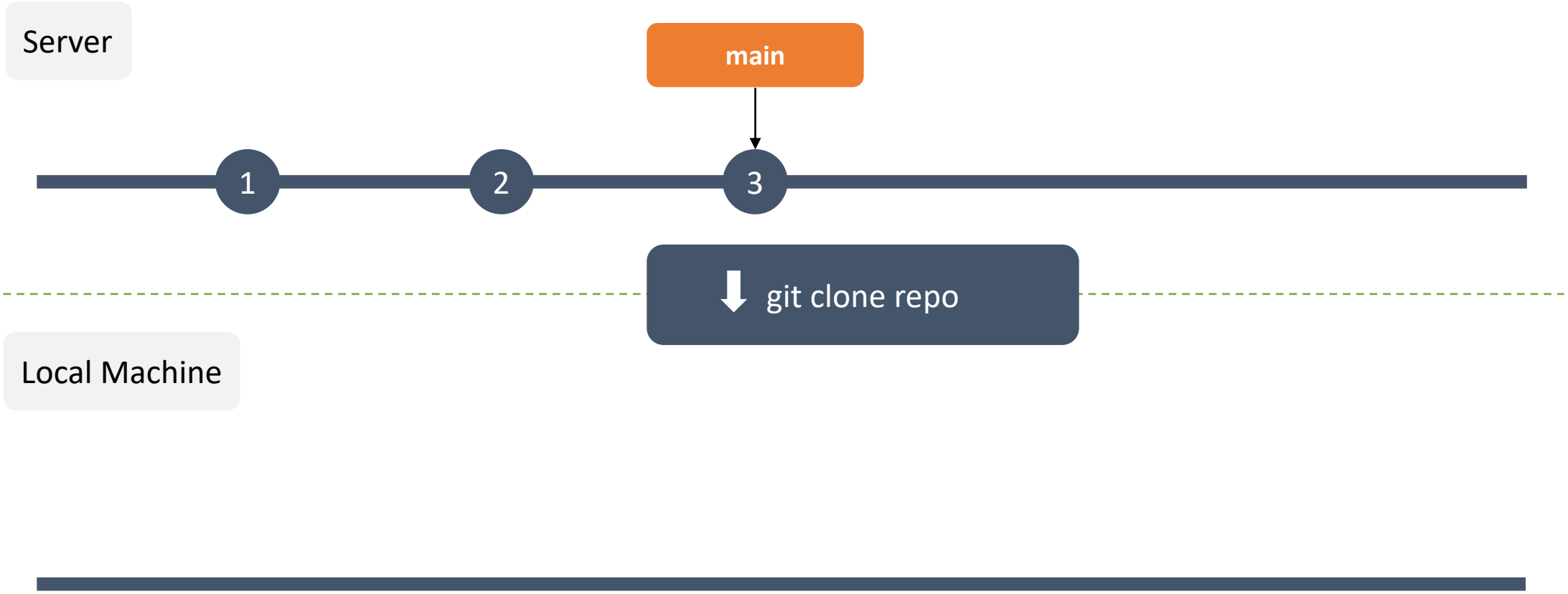
Branch Types – Remote Tracking Branches

- Remote-tracking branch names take the form

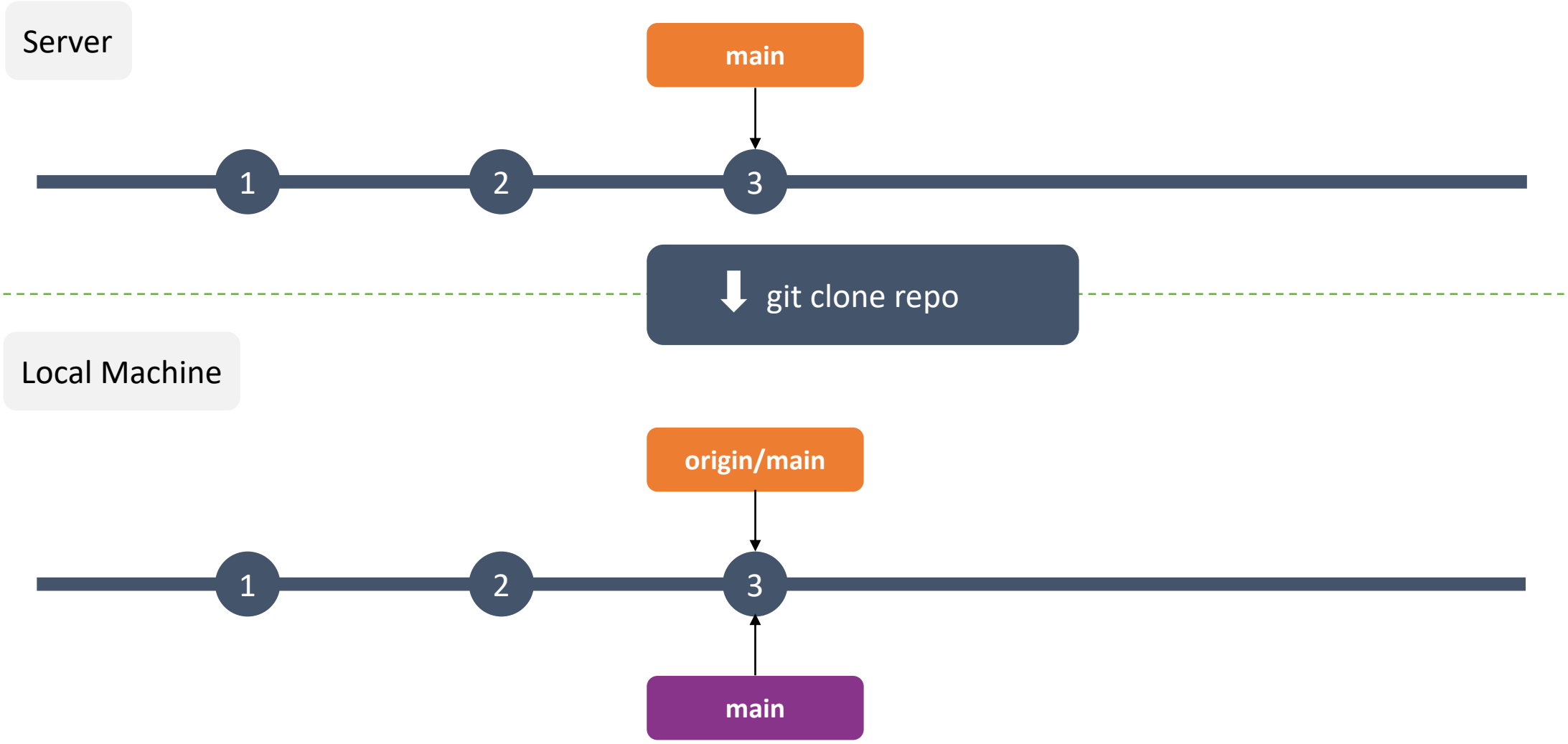
  `<remote>/<branch name>` e.g., `origin/main`

- By default, a `git clone` will create origin/master or origin/main
- Once setup, they can automatically `fetch` and `push` changes to the remote branch
- `git status` will recognize how many commits the tracking branch is in front/back of the remote version of the branch

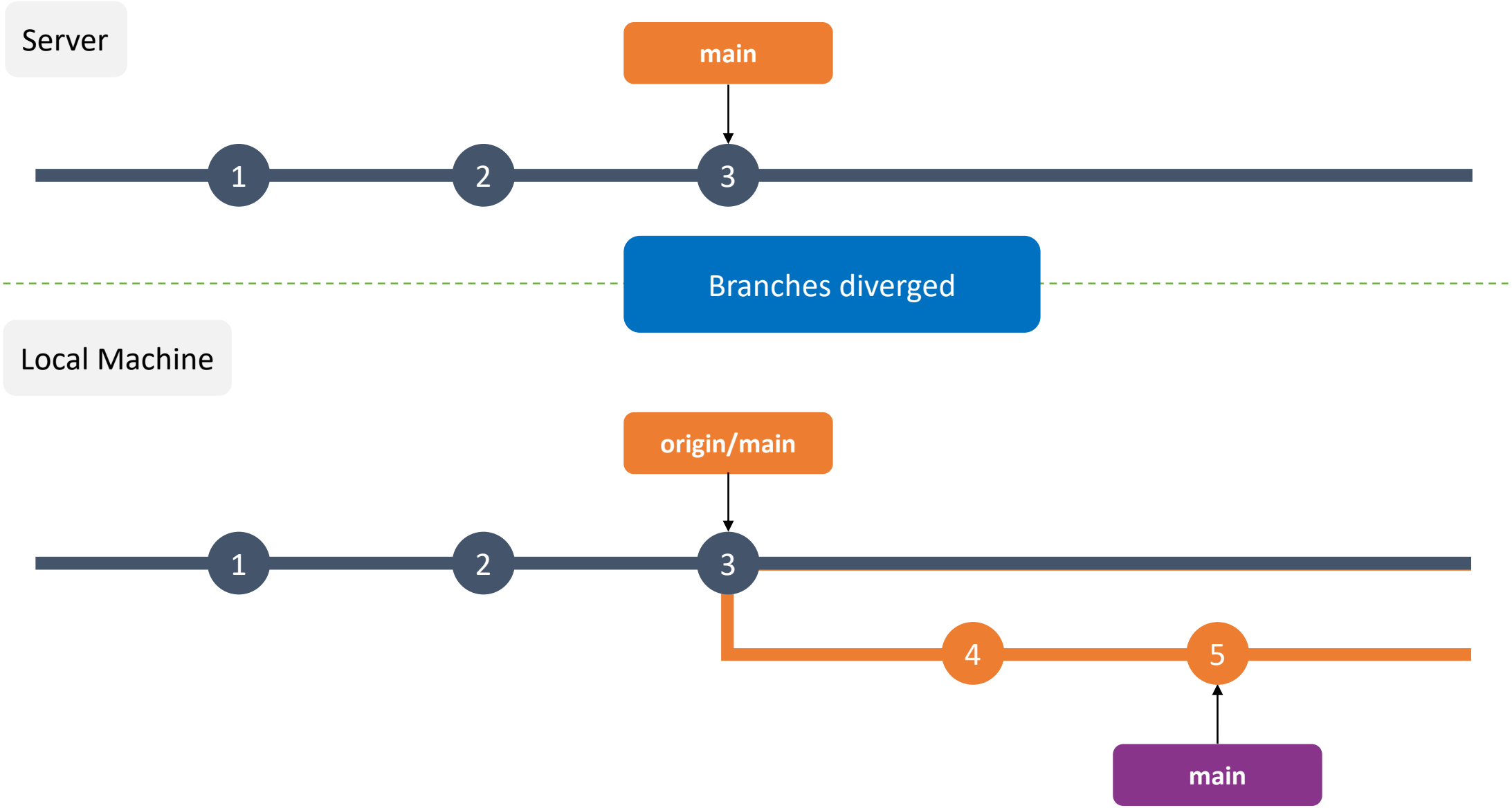
Tracking and Remote Tracking Branches



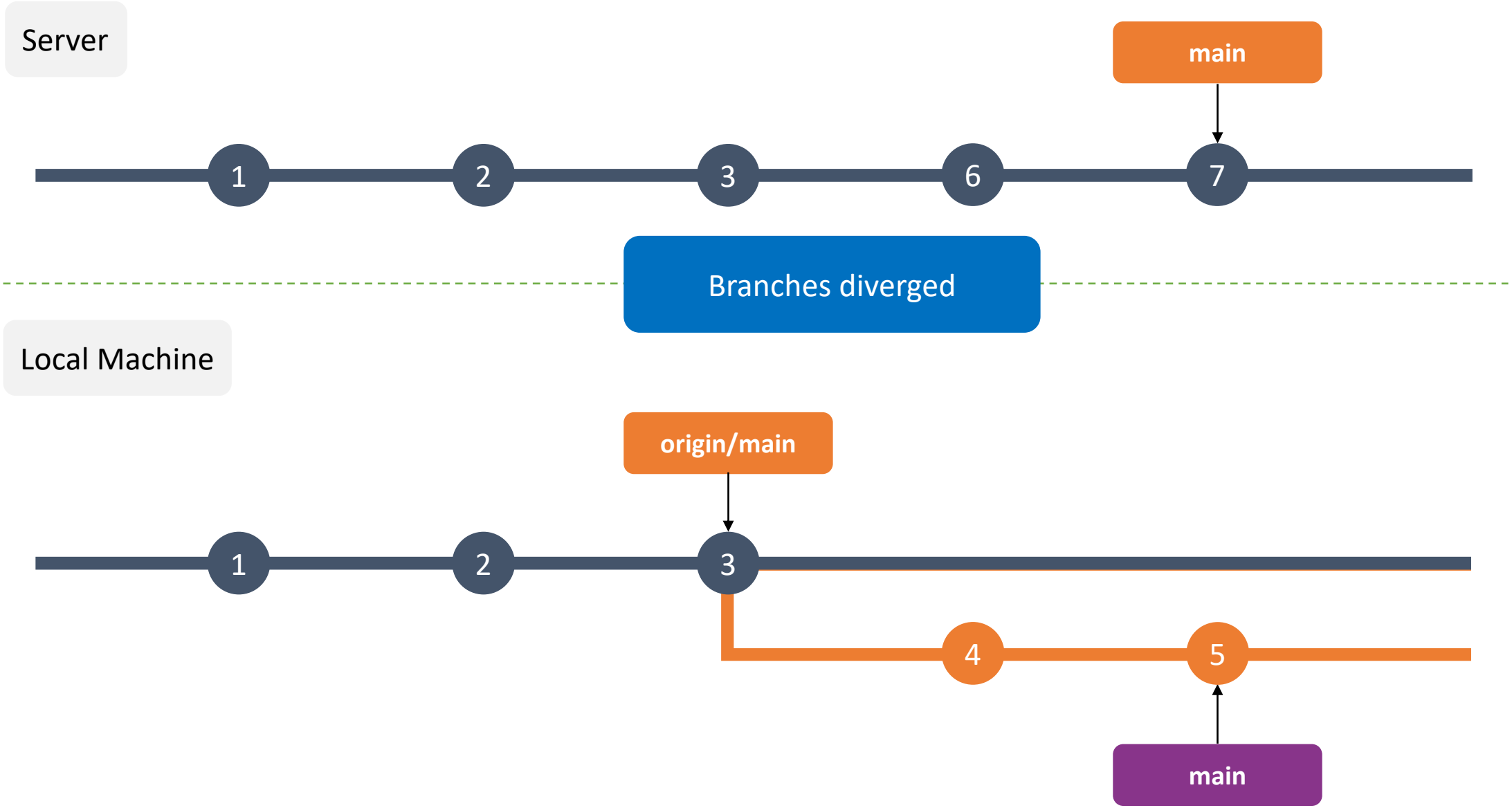
Tracking and Remote Tracking Branches



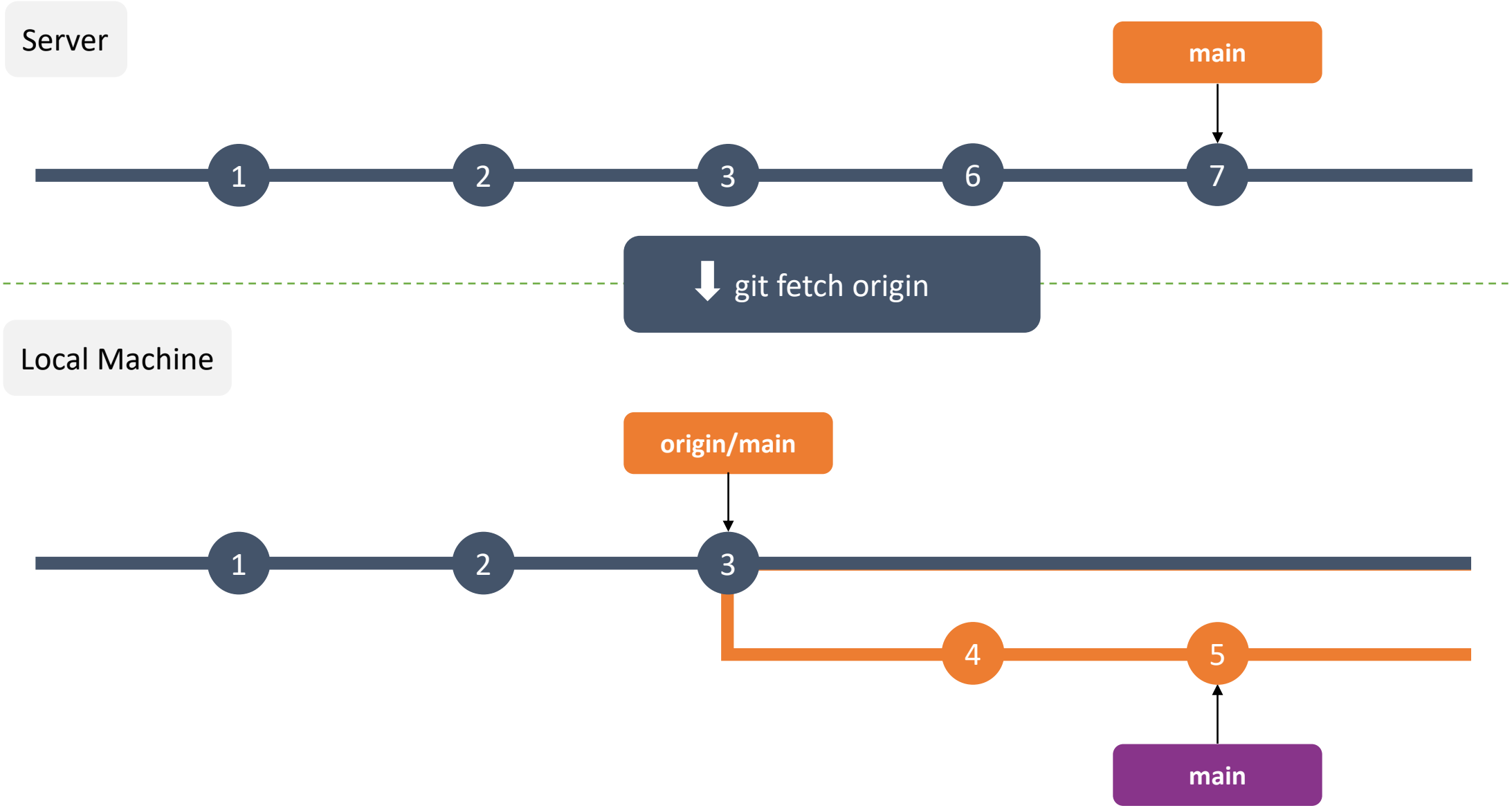
Tracking and Remote Tracking Branches



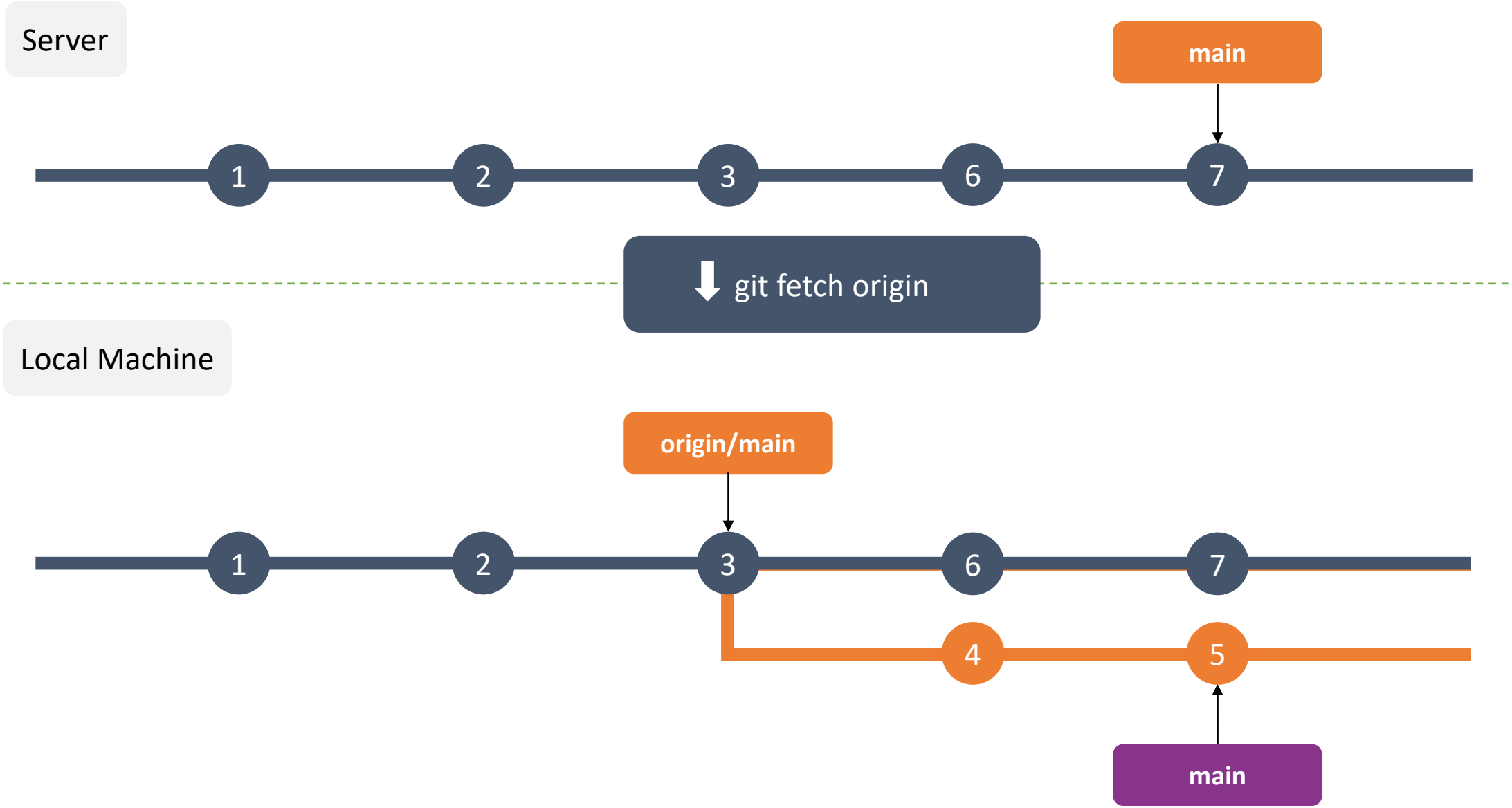
Tracking and Remote Tracking Branches



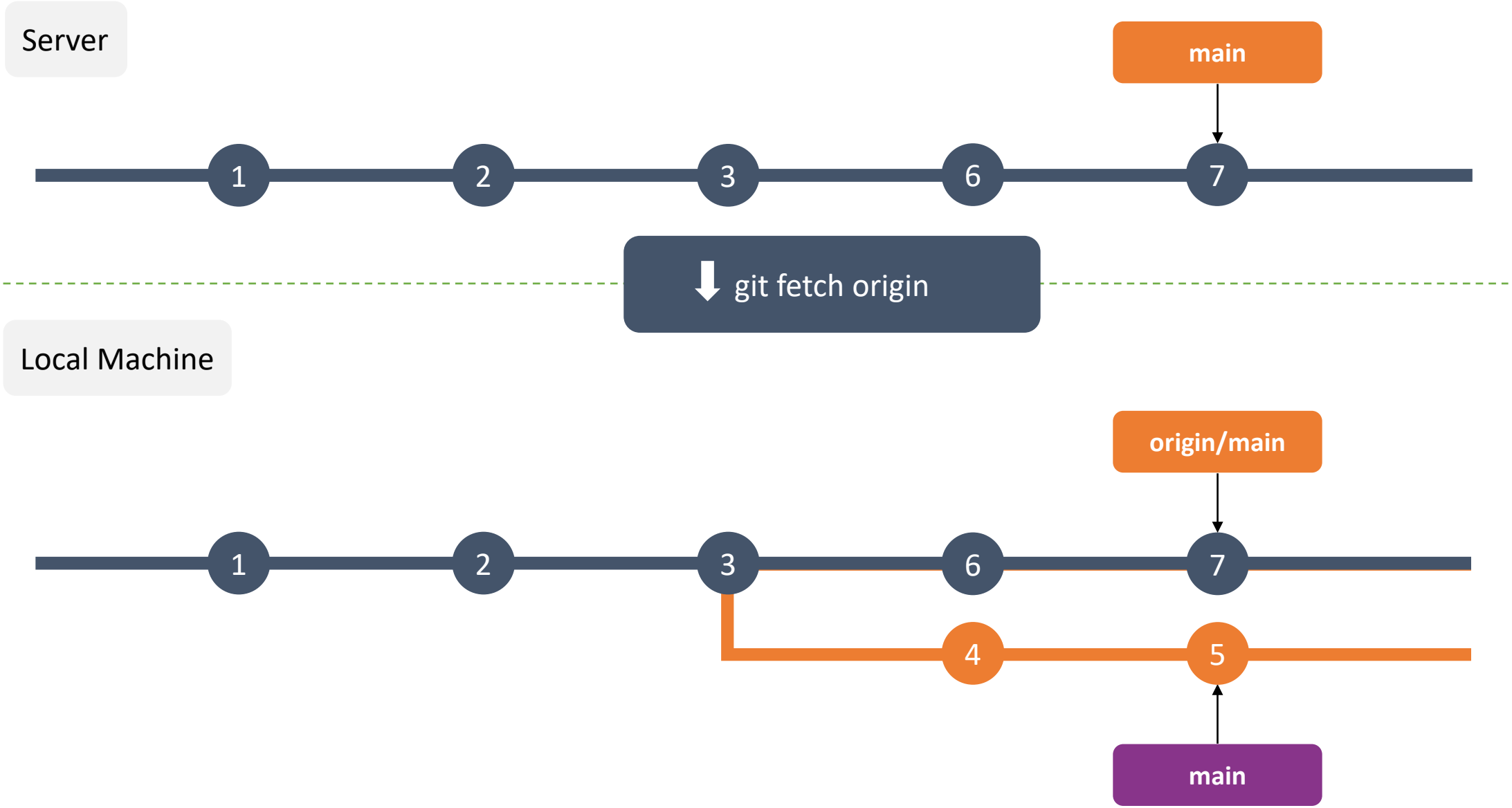
Tracking and Remote Tracking Branches



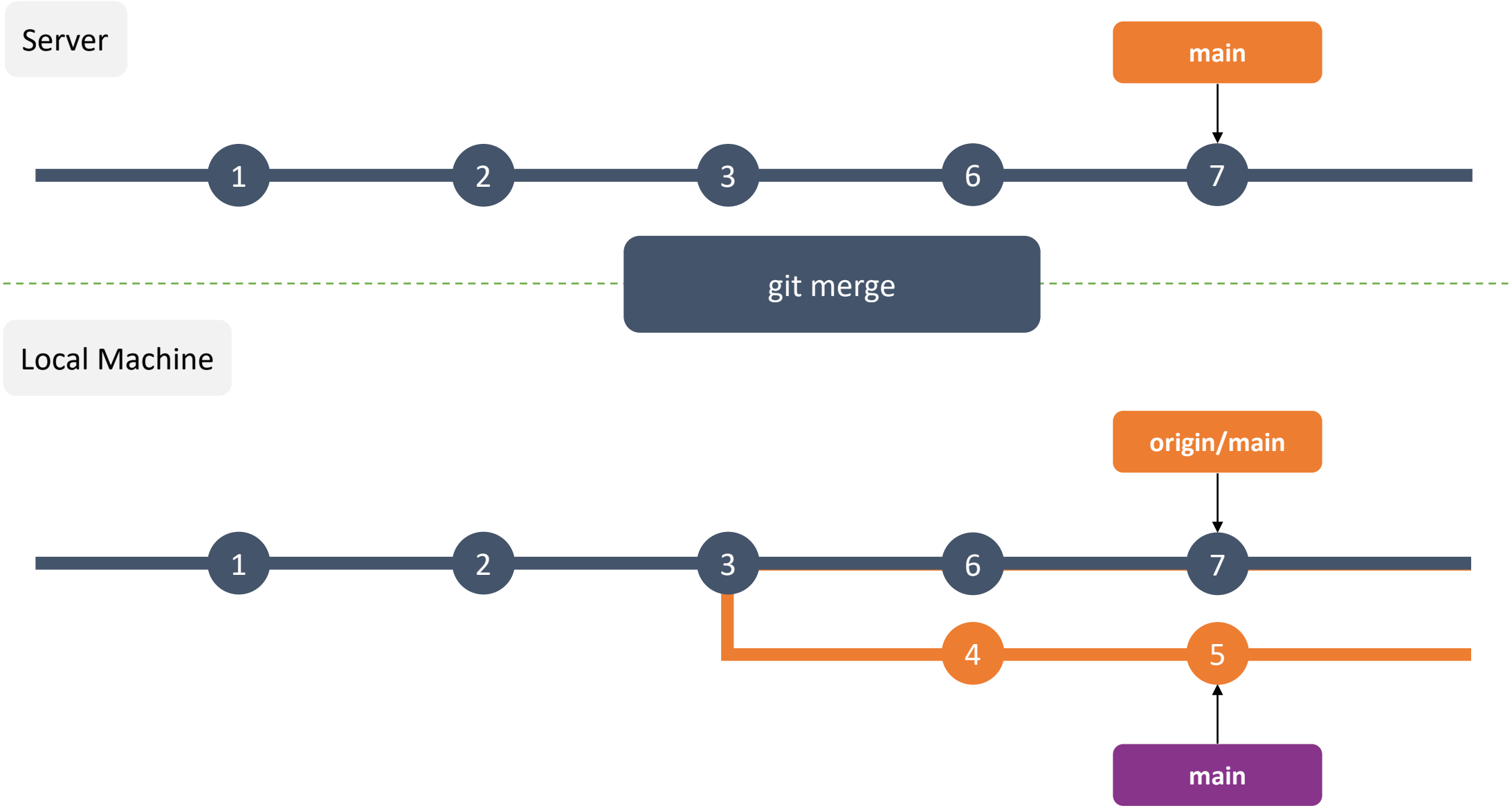
Tracking and Remote Tracking Branches



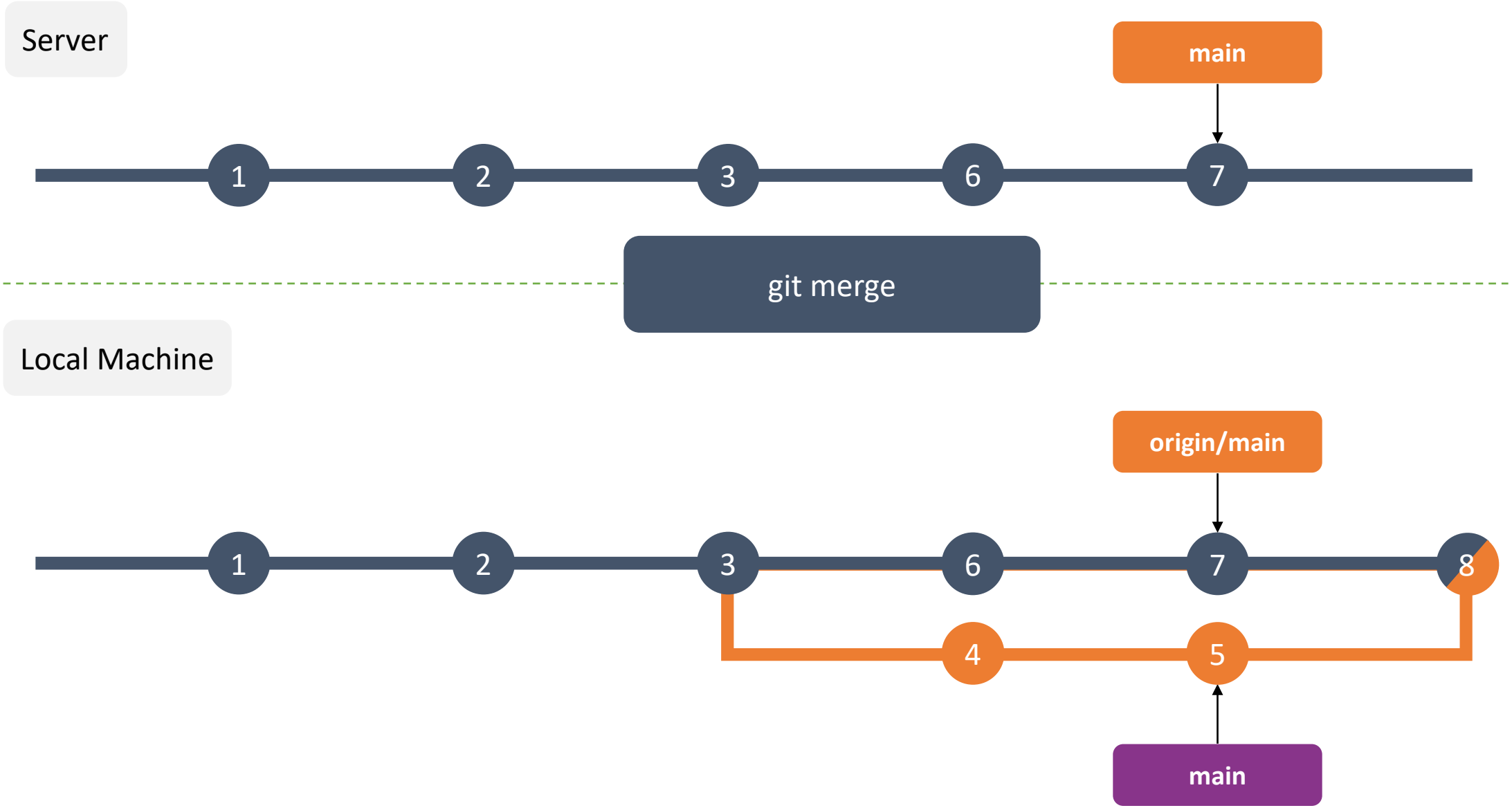
Tracking and Remote Tracking Branches



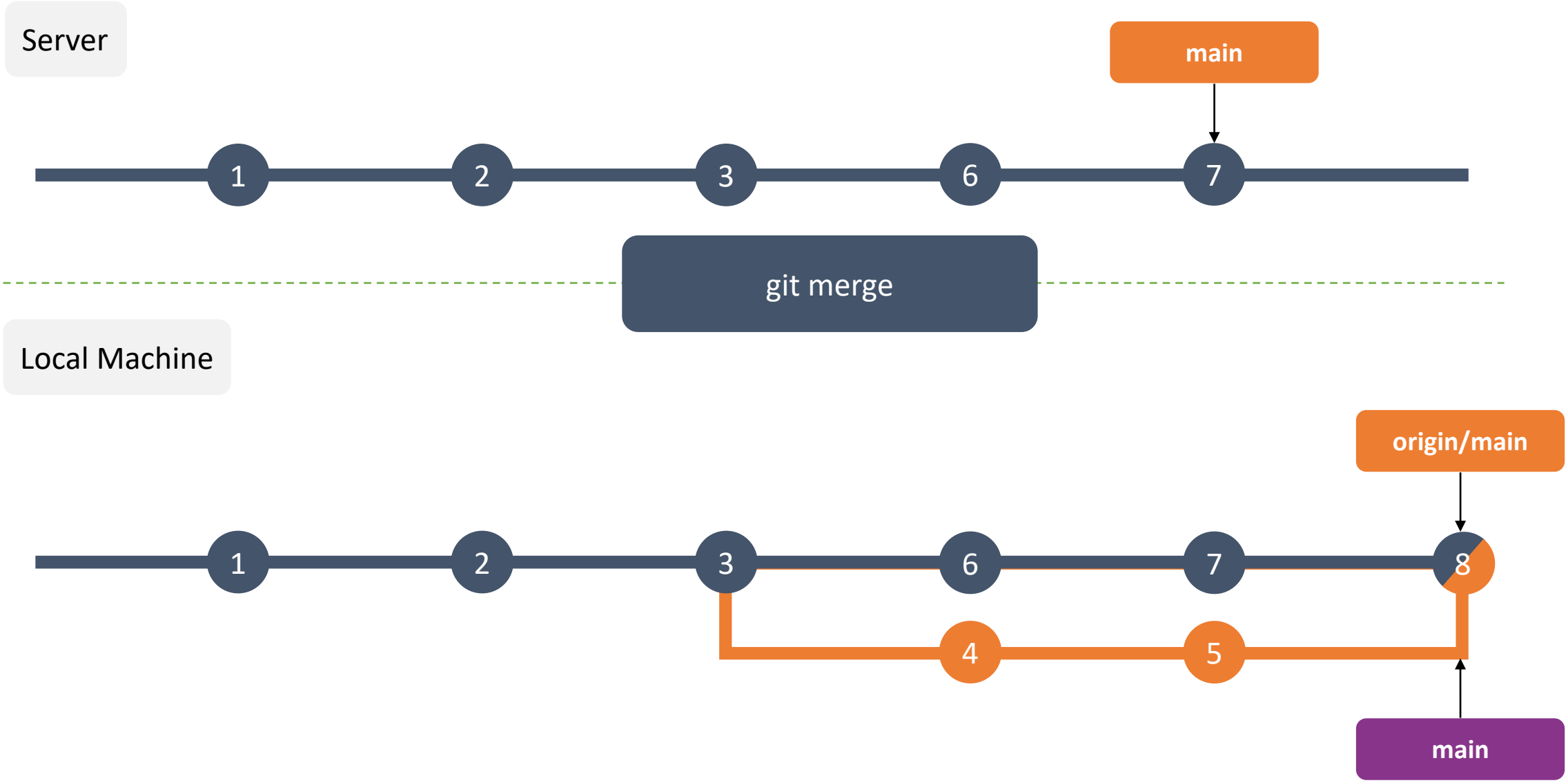
Tracking and Remote Tracking Branches



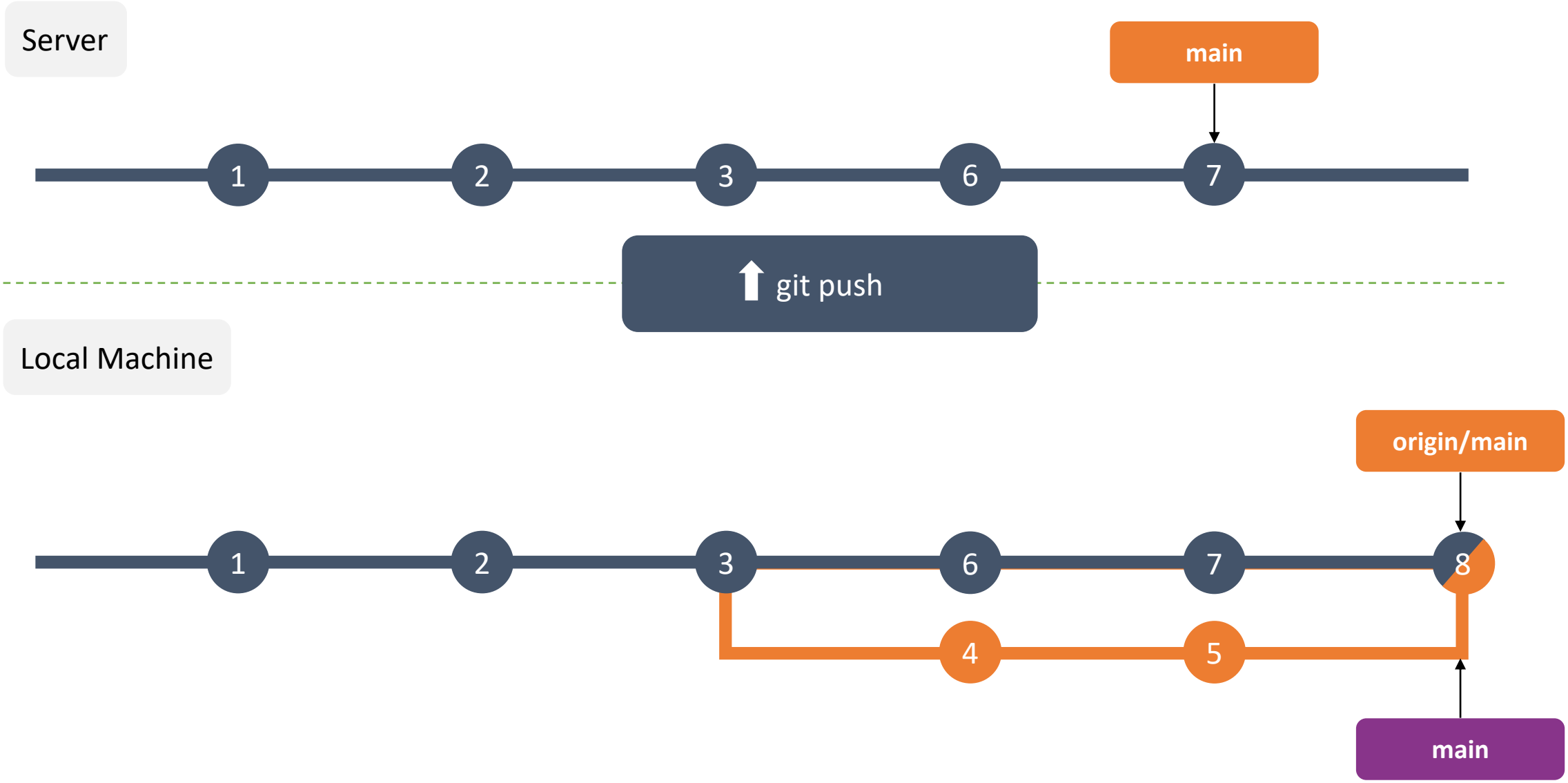
Tracking and Remote Tracking Branches



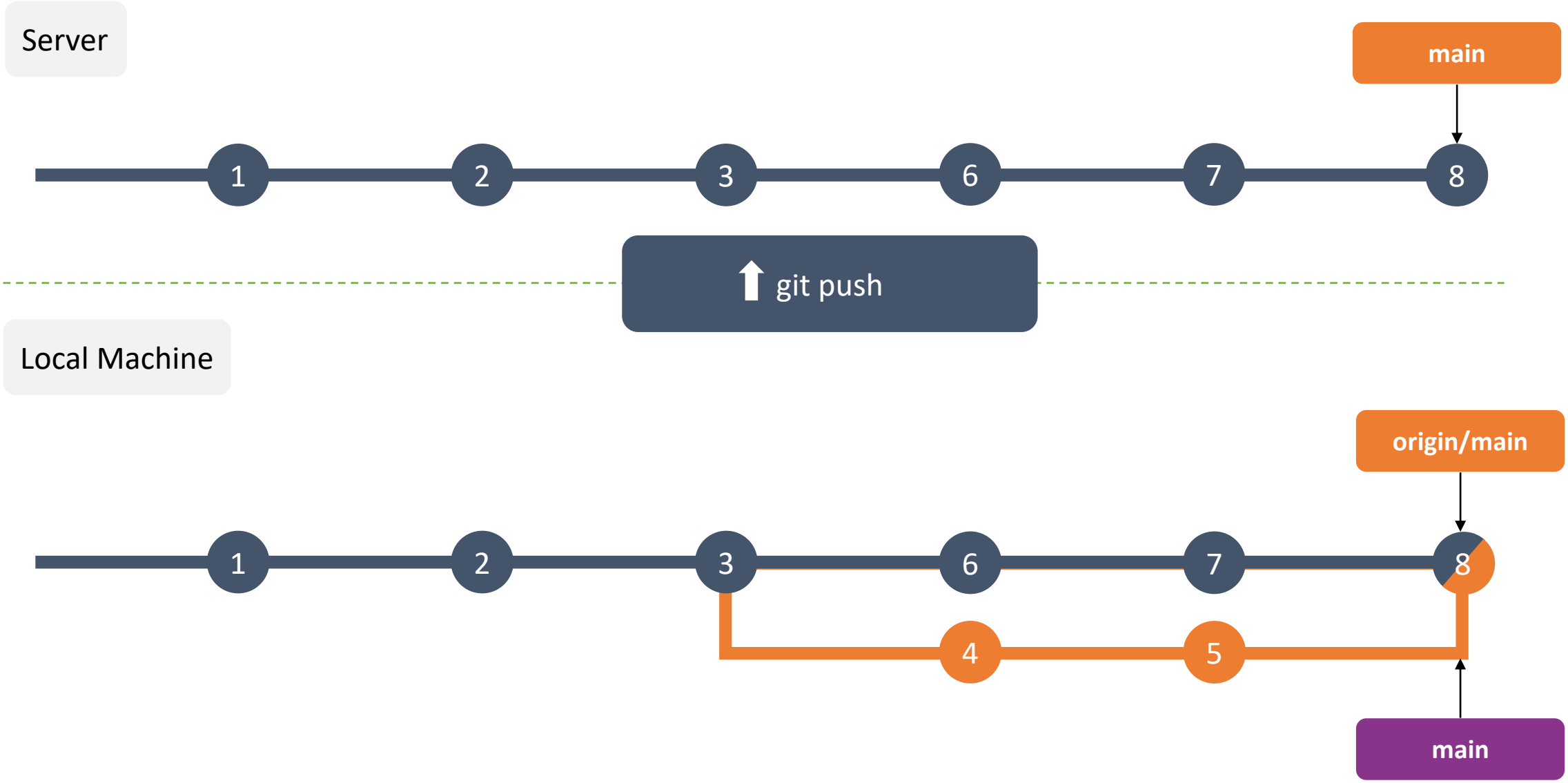
Tracking and Remote Tracking Branches



Tracking and Remote Tracking Branches



Tracking and Remote Tracking Branches



User 1

Scenario 1 – The Local Protocol

Remote Repo

1. Initialize a `bare` repo

Local Repo

2. Initialize a local repo
3. Make changes
4. Add `remote`
5. `Push` to the remote repo

User 2

Scenario 1 – The Local Protocol

Remote Repo

Local Repo

1. `Clone` the remote repo
2. Make changes
3. `Push` to remote

User 1

Scenario 1 – The Local Protocol

Remote Repo

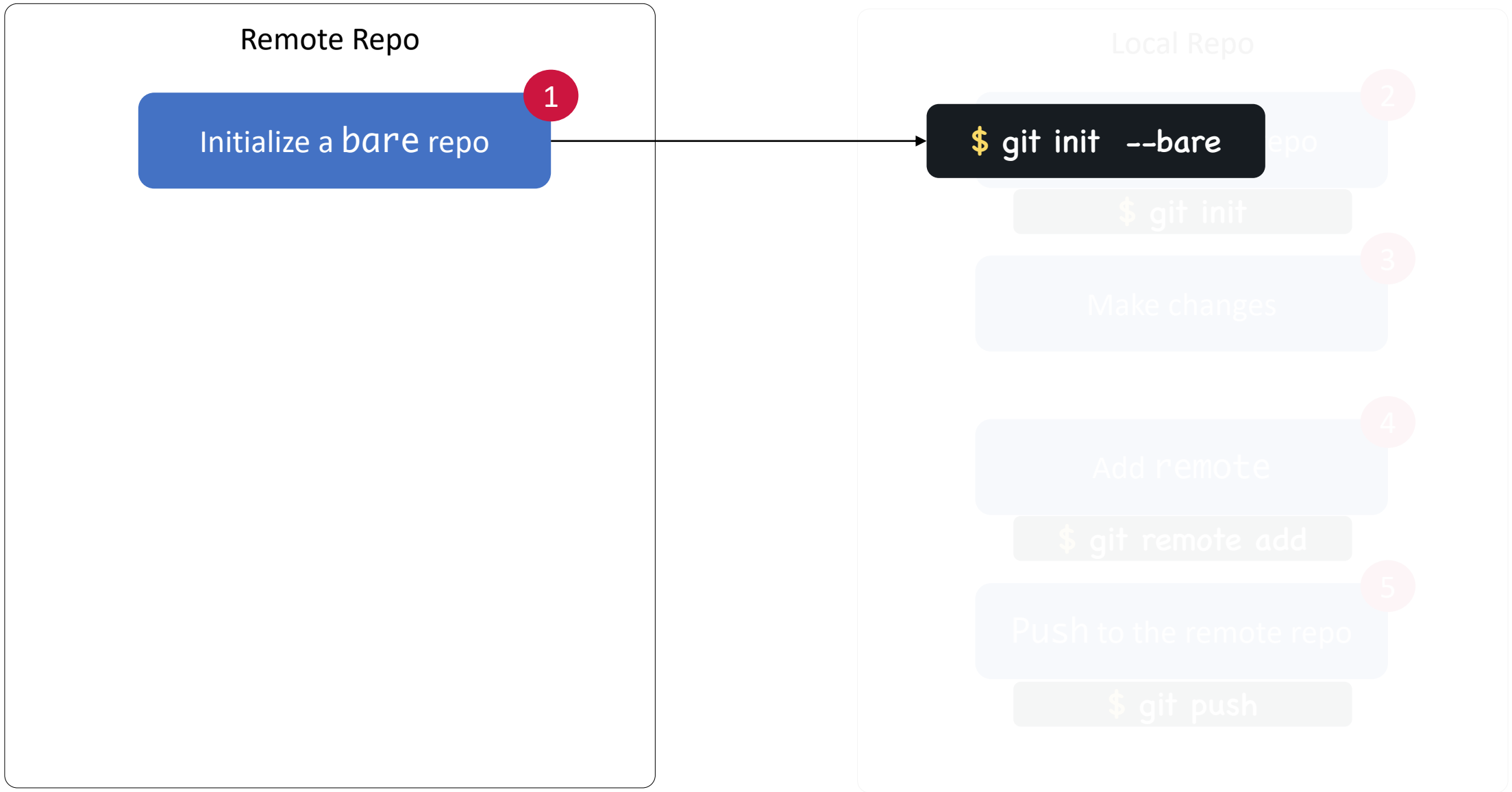
Local Repo

1. Switch to `main` branch
2. Connect branch `upstream`
3. `Fetch` the changes
4. `Merge` the changes

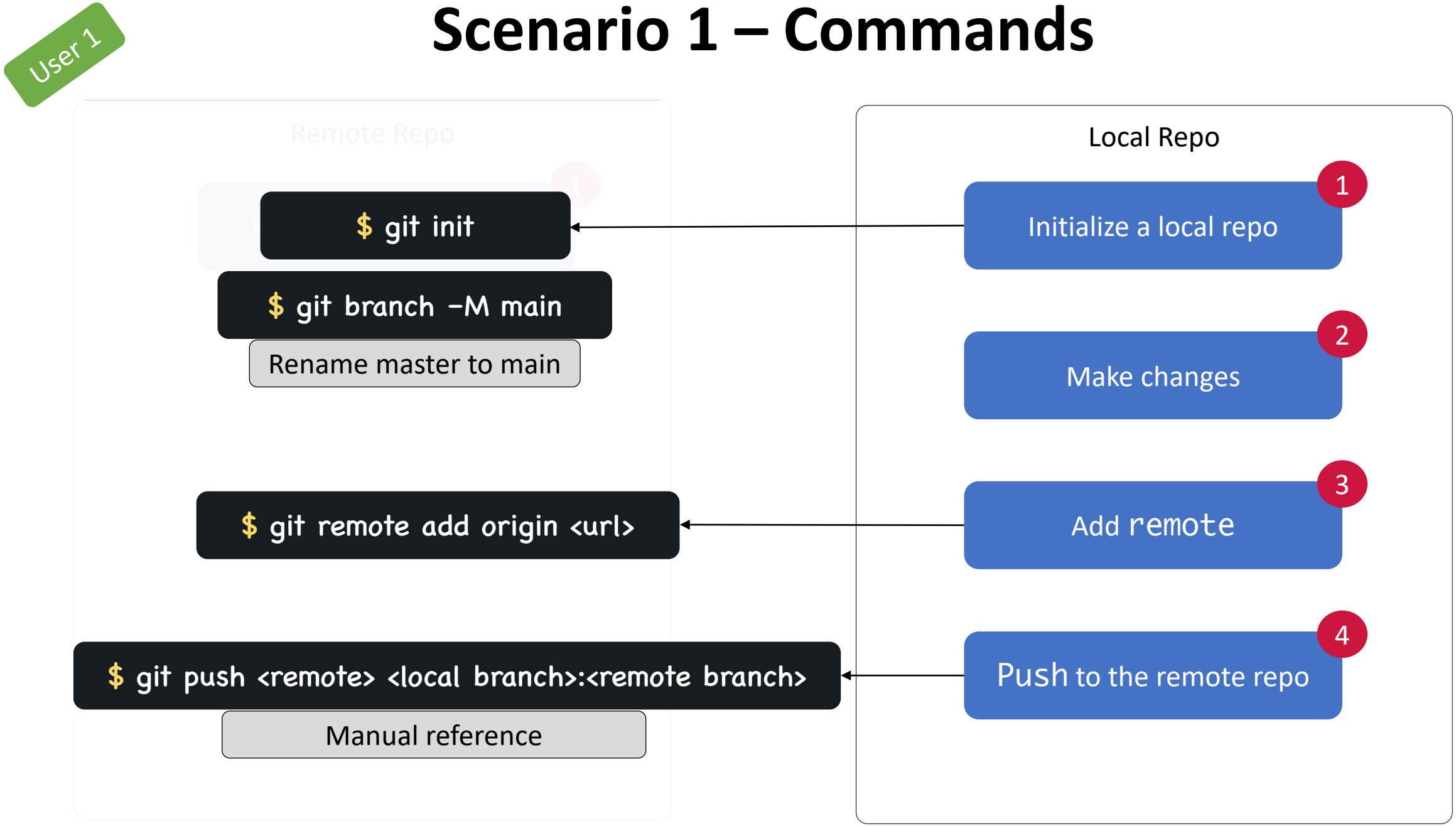
Let us practice

Scenario 1 – Commands

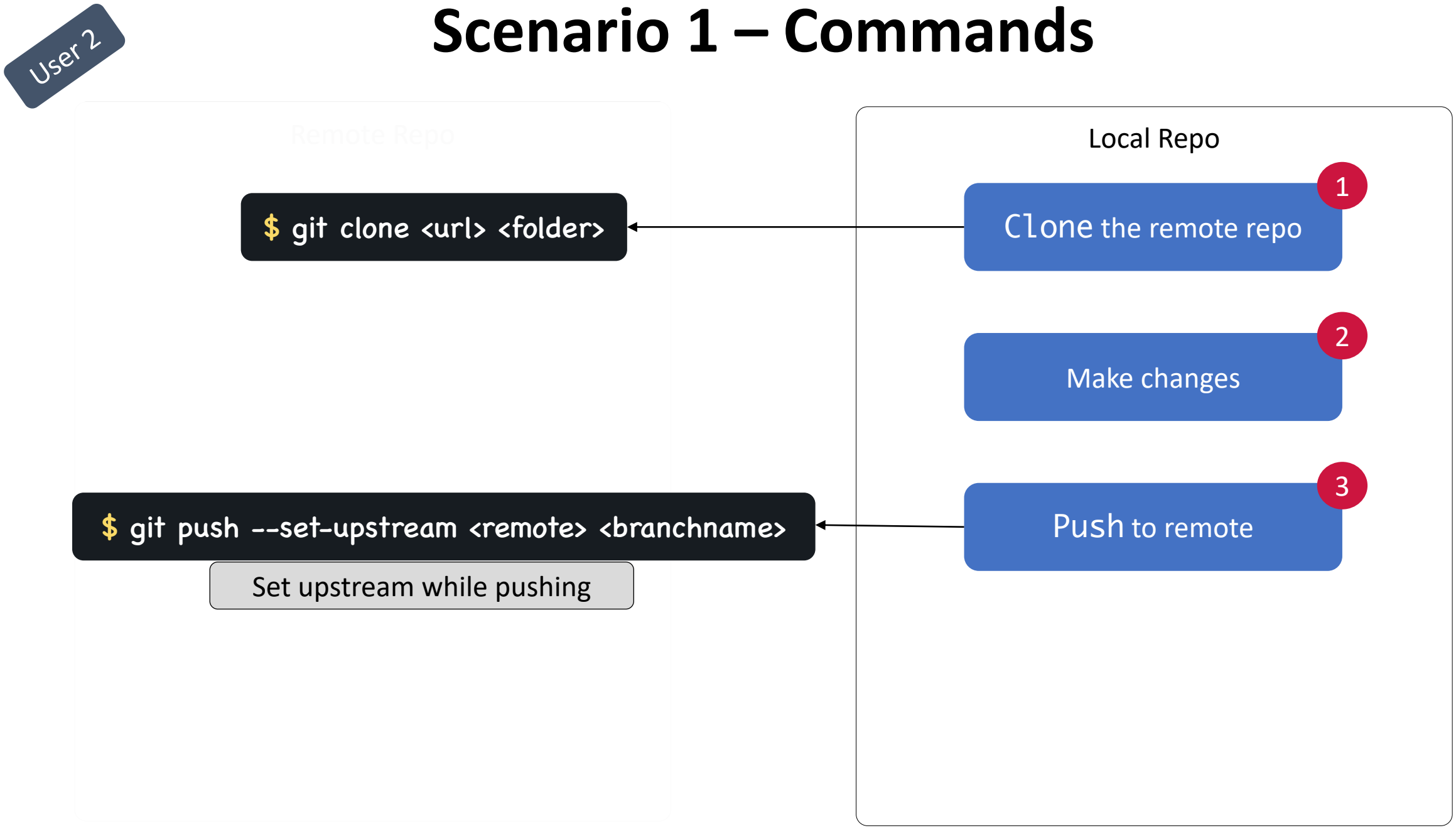
User 1



Scenario 1 – Commands



Scenario 1 – Commands



Scenario 1 – Commands