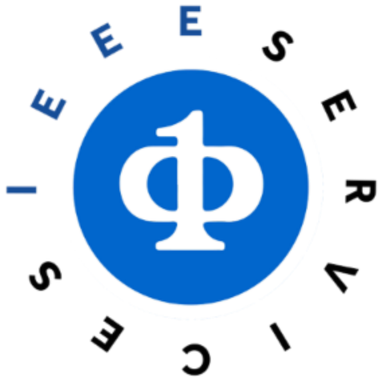
Social Coding With GitHub

Instructor:
John J Rofrano
Senior Technical Staff Member, DevOps Champion
IBM T.J. Watson Research Center
rofrano@us.ibm.com
@JohnRofrano



Objectives

The objectives of this class are to:

- Learn the mechanics of using Git
- Introduce the Git Feature Branch Workflow
- Learn how Git encourages Collaboration
- Become a *Social Coder*

“What makes senior developers senior is not that they know the syntax of a given language better, but that they have experience working with large and complex projects with real users and business goals”

–Ariel Camus - medium

HOW TO BECOME A GIT JEDI

How to become a Git Jedi

Rule #1: Create a Git repository for every new project

How to become a Git Jedi

Rule #1: Create a Git repository for every new project

Rule #2: Create a new branch for every new feature

How to become a Git Jedi

Rule #1: Create a Git repository for every new project

Rule #2: Create a new branch for every new feature

Rule #3: Use Pull Requests to merge code to Master

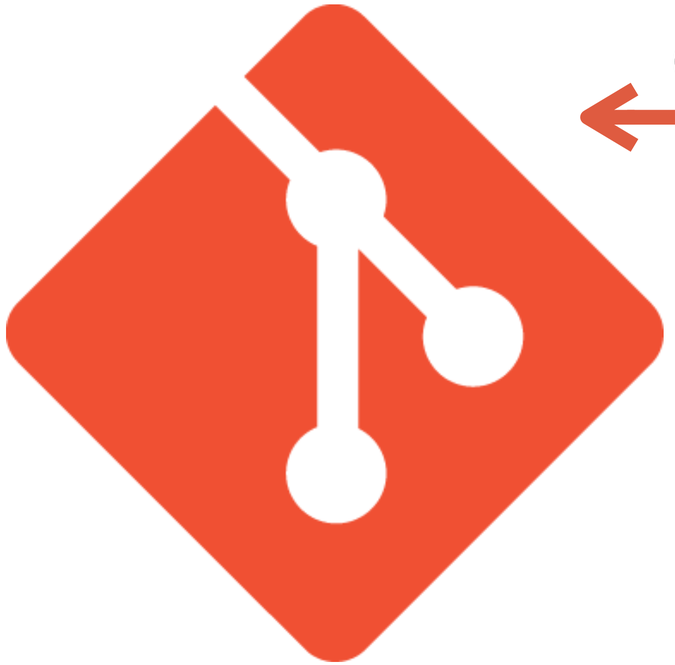
Git

- A distributed source code management (SCM) tool invented by Linus Torvalds in 2005 for Linux kernel development
- Code is kept in a repository and every developer has a full copy
- Works locally without any server
- Works remotely with GitHub, GitLab, & BitBucket

GitHub

- A web site that hosts git repositories
- Founded in 2007, has 40 million registered developers and was acquired by Microsoft for a whopping $7.5 billion in 2018
- Free and Paid accounts
- Adds ability to track Issues and Bugs
- Provides webhooks to integrate other tools
- ...and a whole lot more

Git vs GitHub

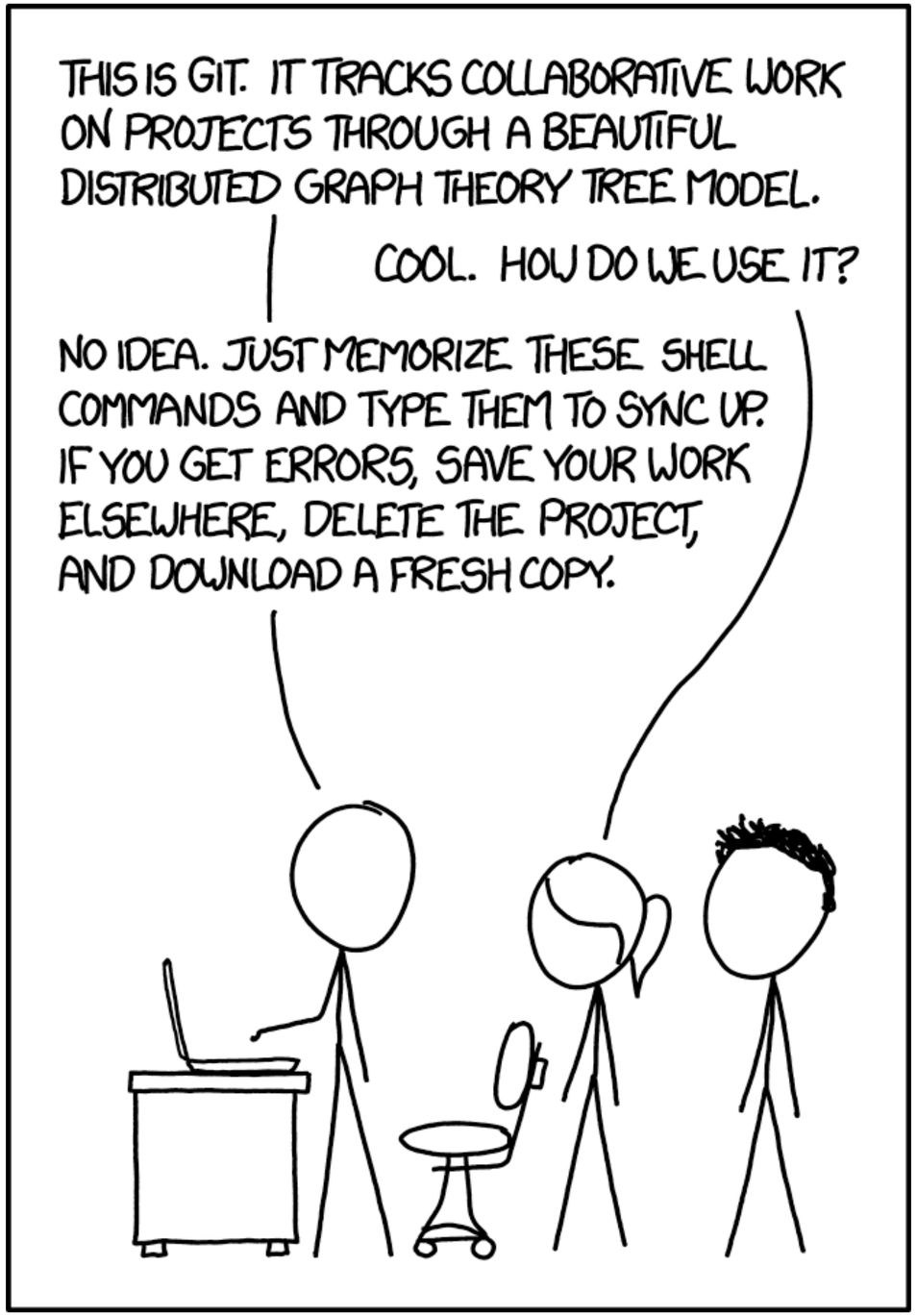


Git is the tool

GitHub is the web site

How many of you know Git?

How many of you know Git?



Git Overview

- Git is a Decentralized Source Code Management (SCM) System
- Developers work in their own BRANCHES (even in FORKS)
- The MASTER branch should always be ready to deploy
- PULL REQUESTS are used to MERGE code BRANCHES into MASTER

Huh?

Branches?

Forks?

Spoons?

Merge?

Master?

Pull Requests?

Pulling hair?

What?

Git Command Workflow

workspace

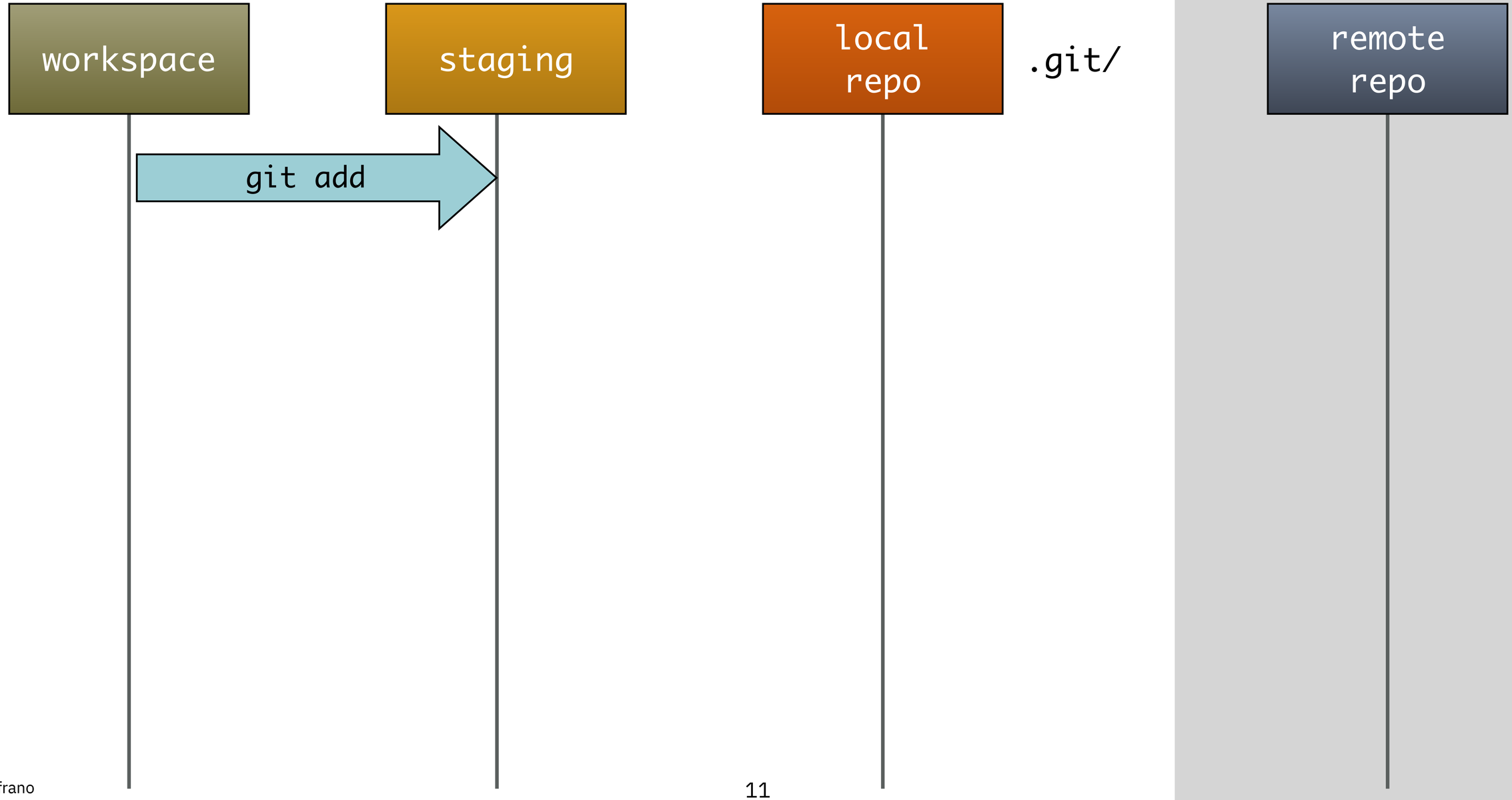
staging

local repo

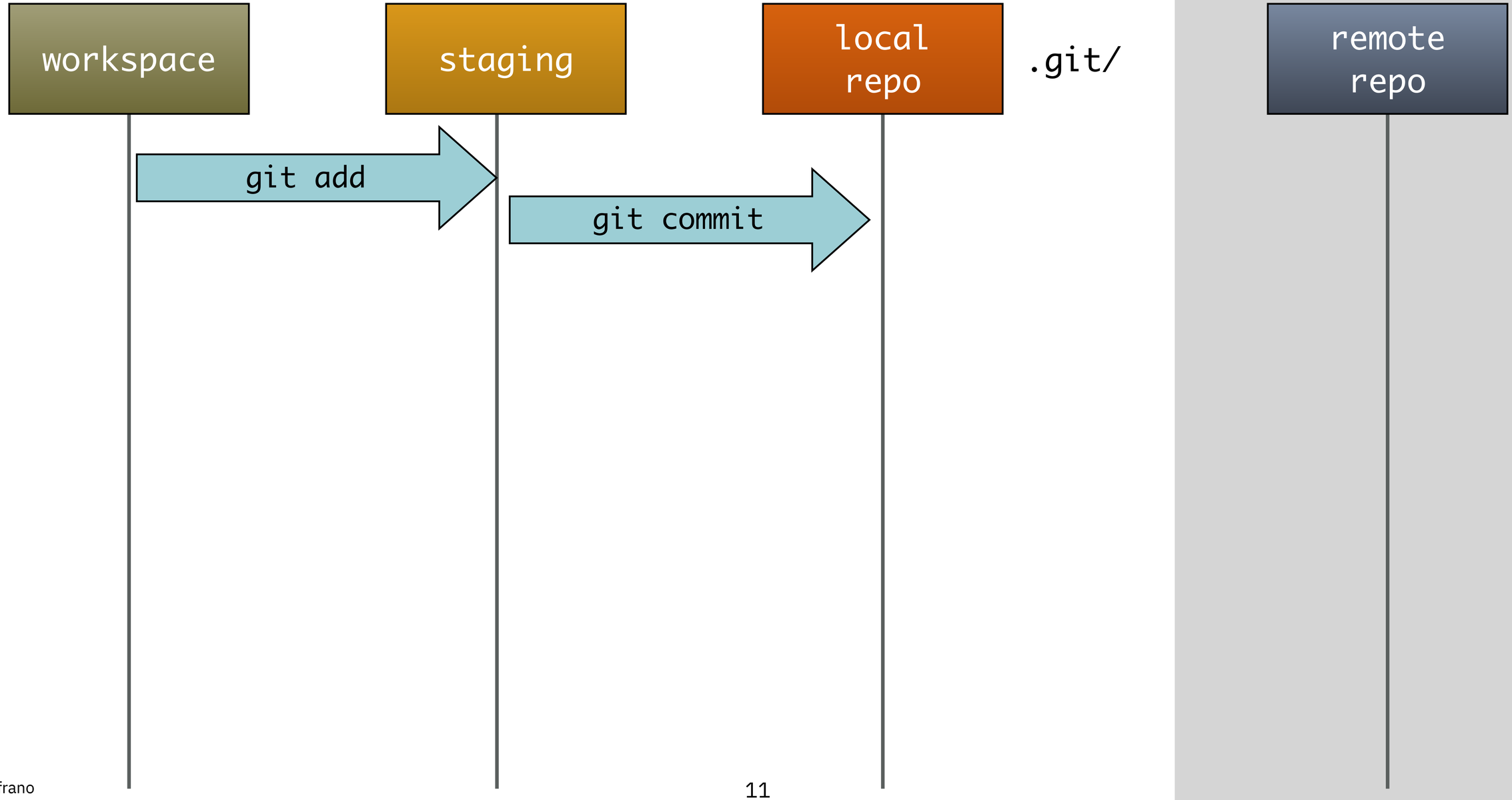
.git/

remote repo

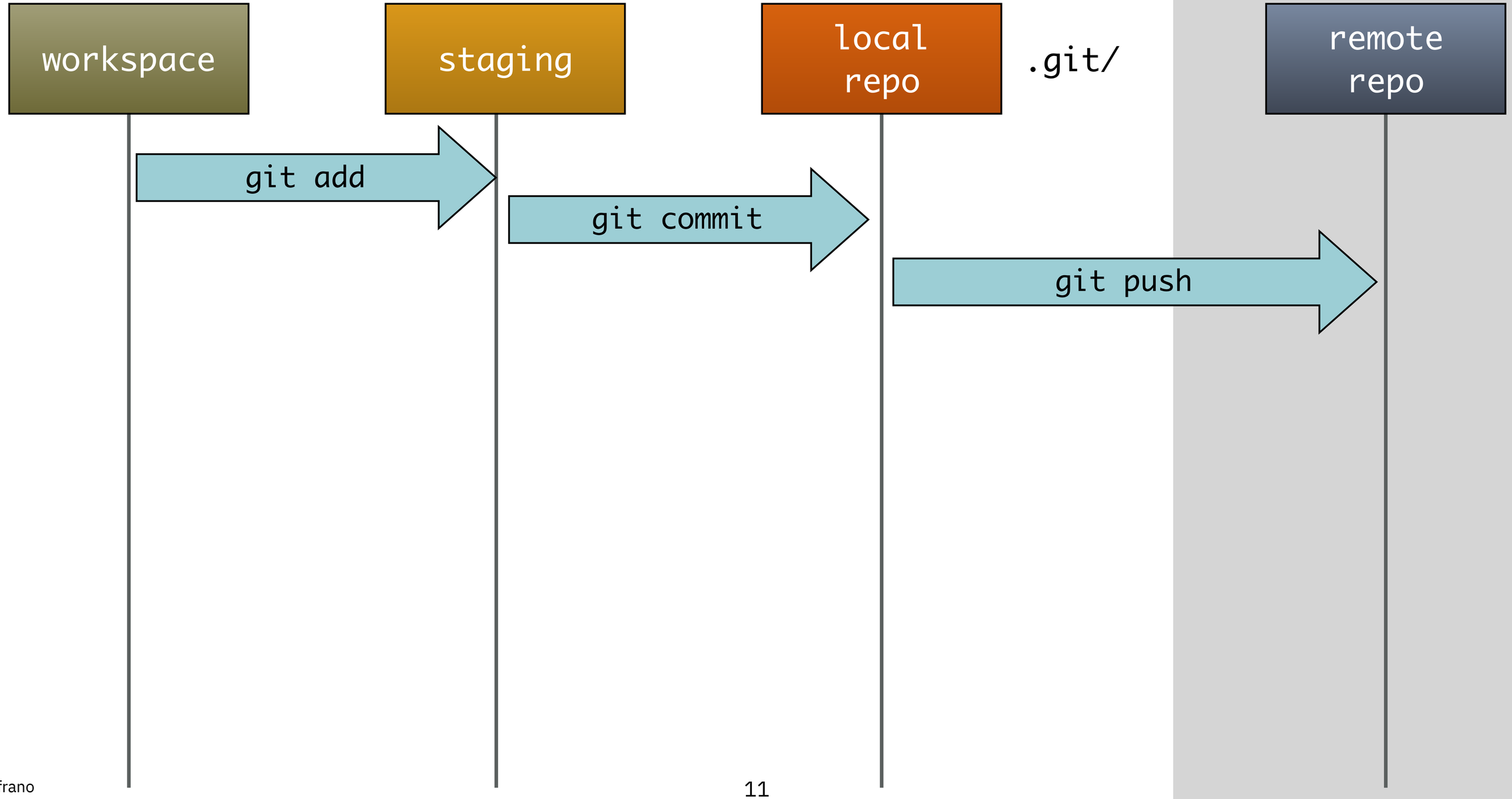
Git Command Workflow



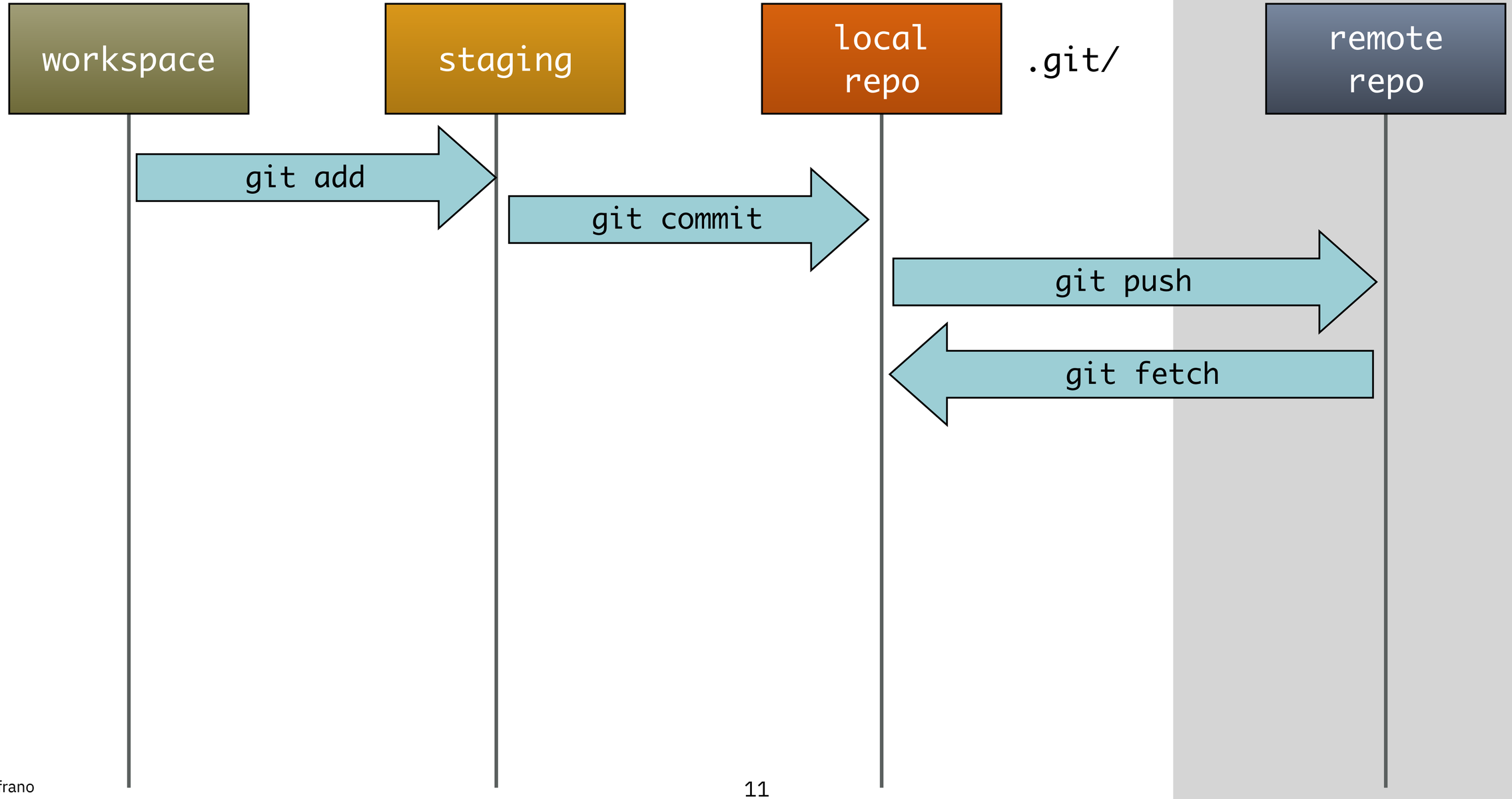
Git Command Workflow



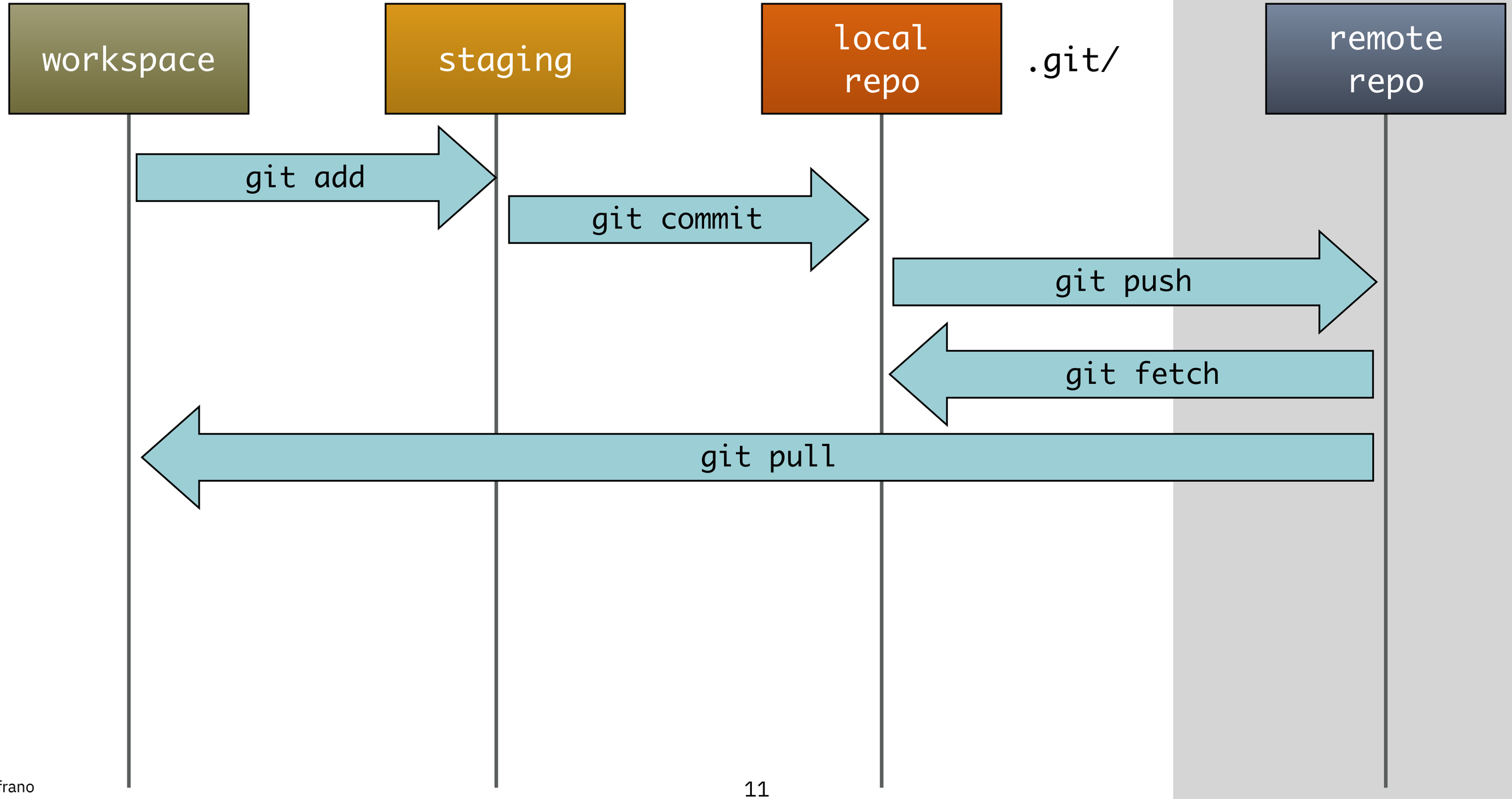
Git Command Workflow



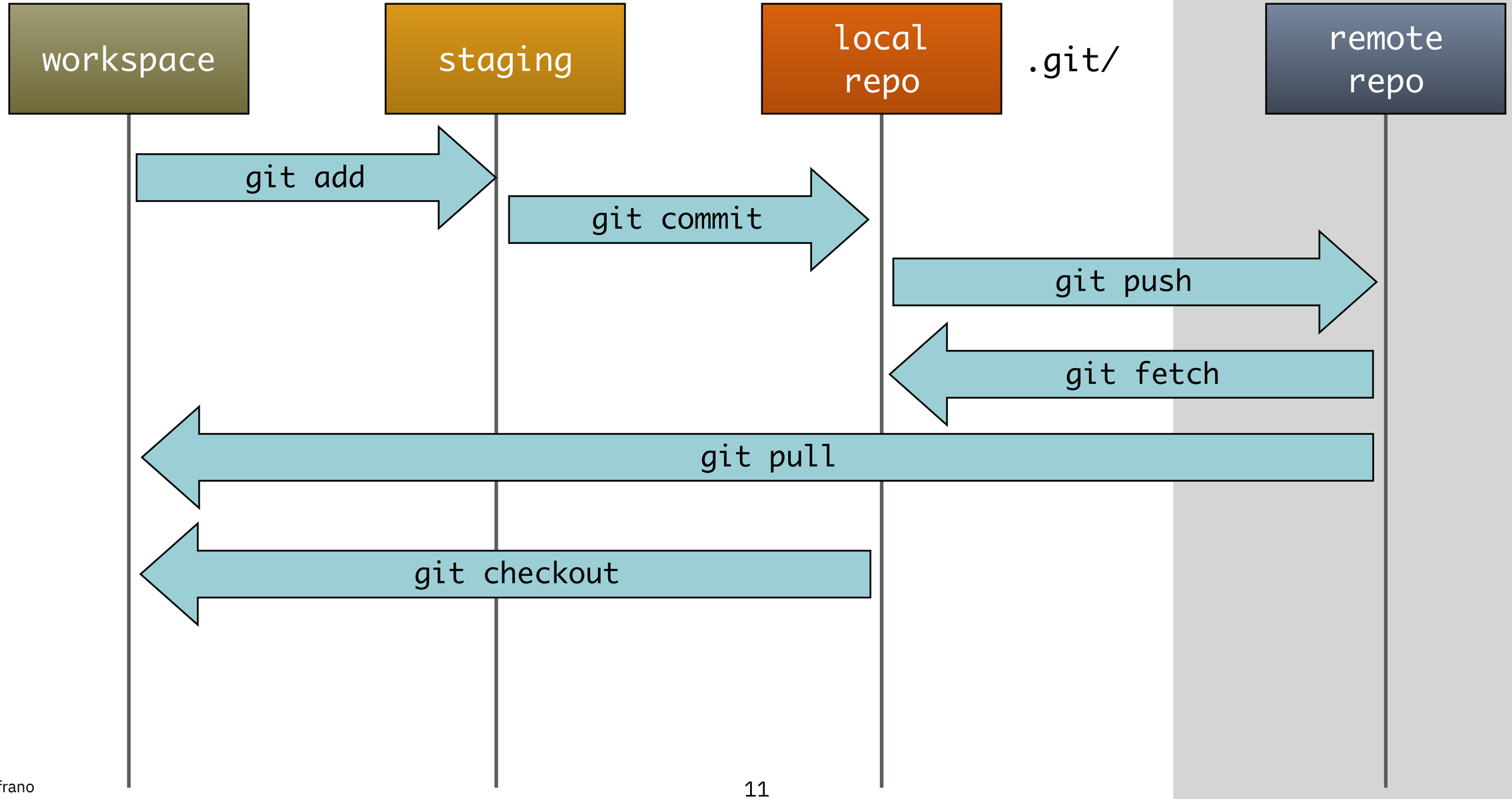
Git Command Workflow



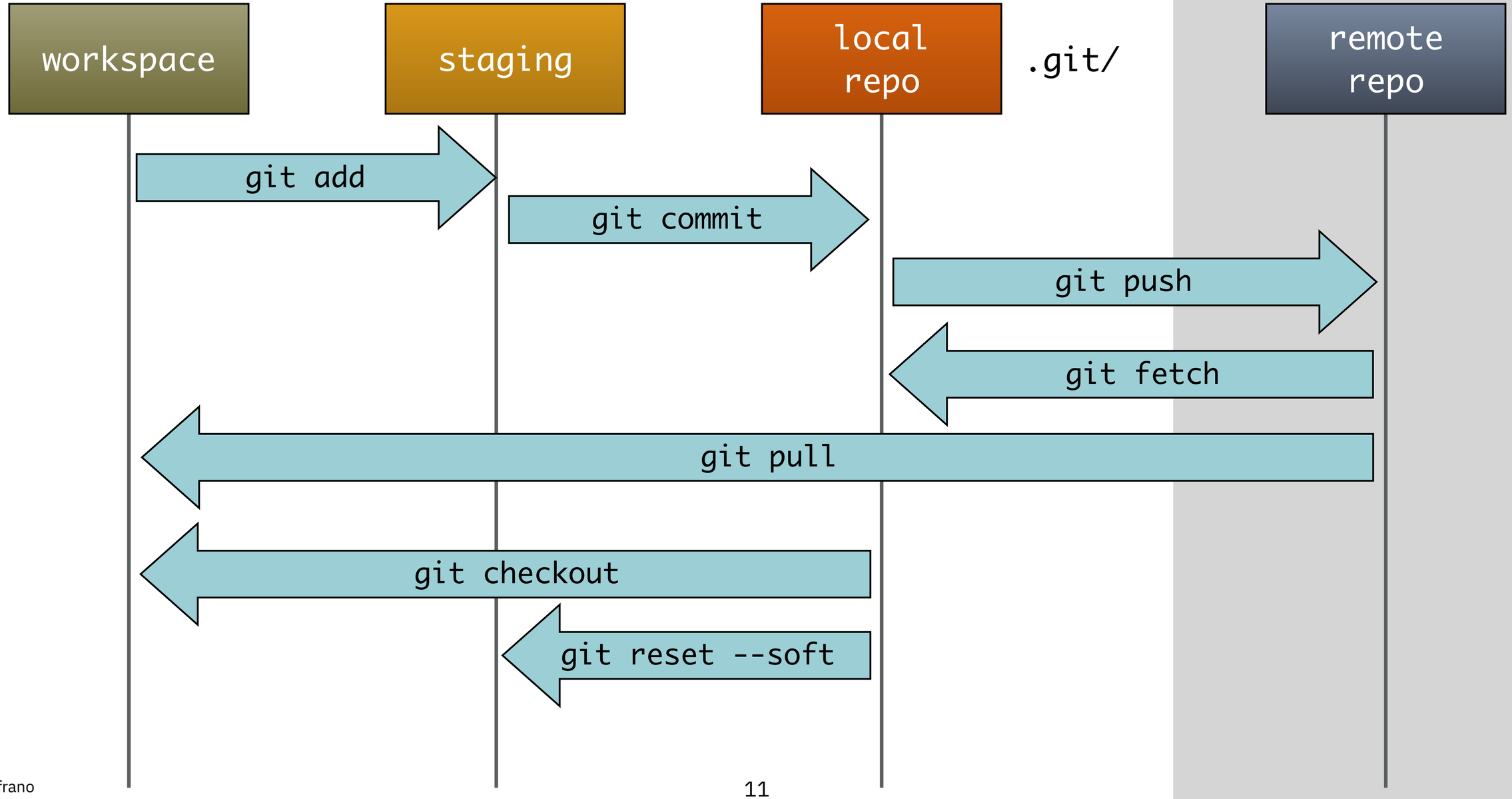
Git Command Workflow



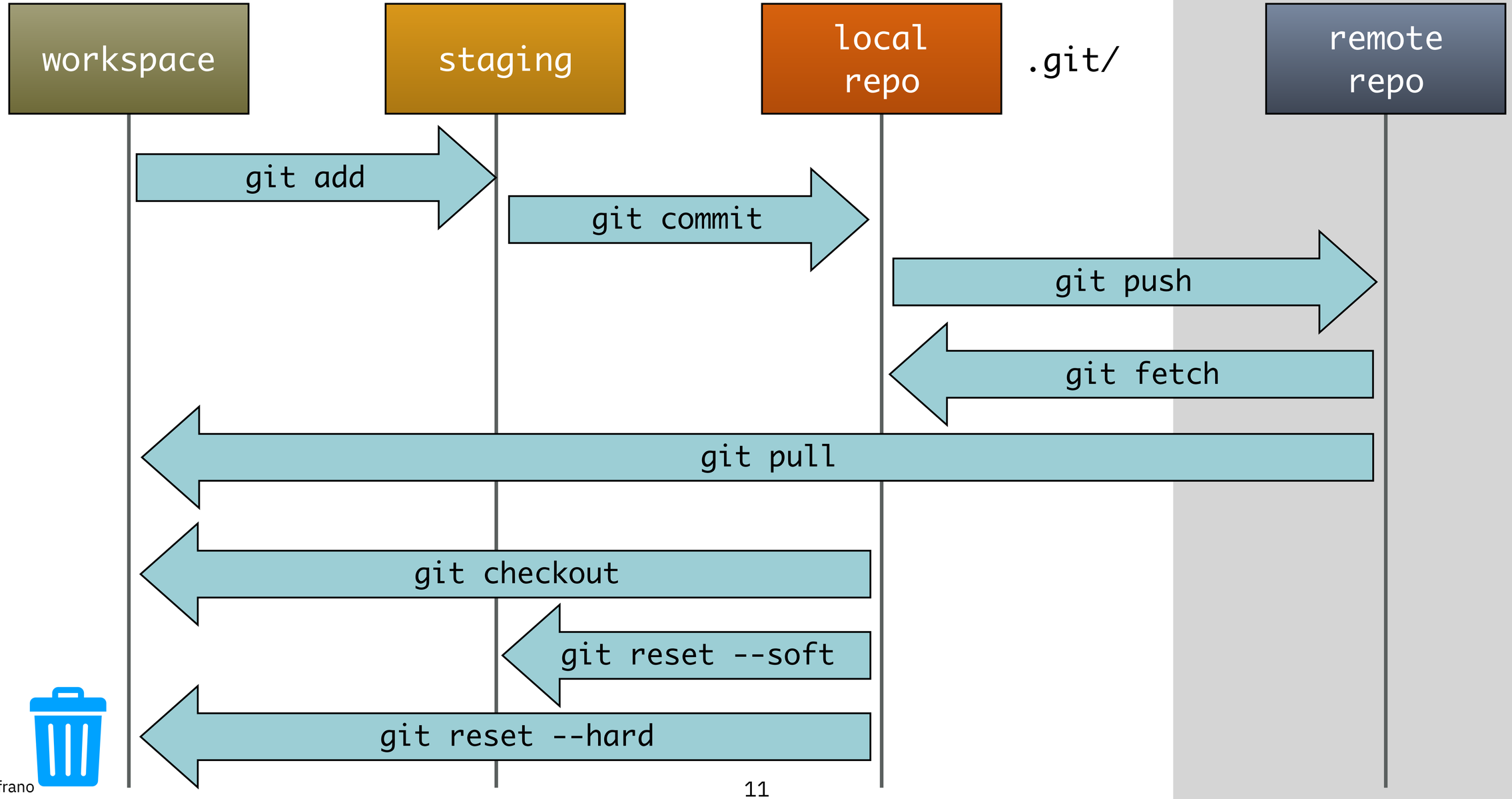
Git Command Workflow



Git Command Workflow

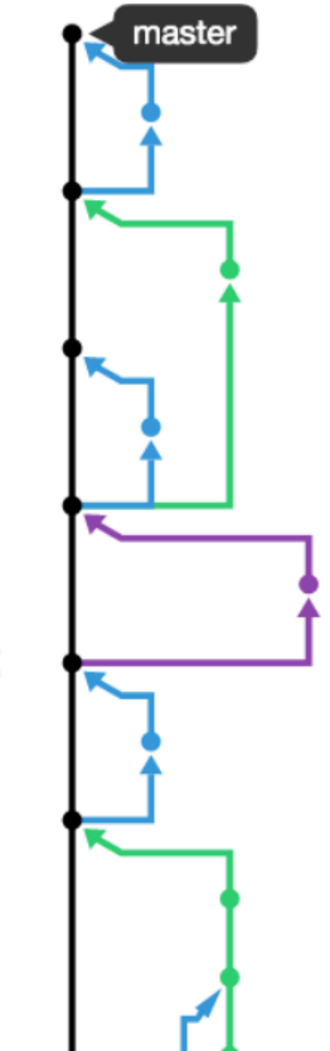


Git Command Workflow

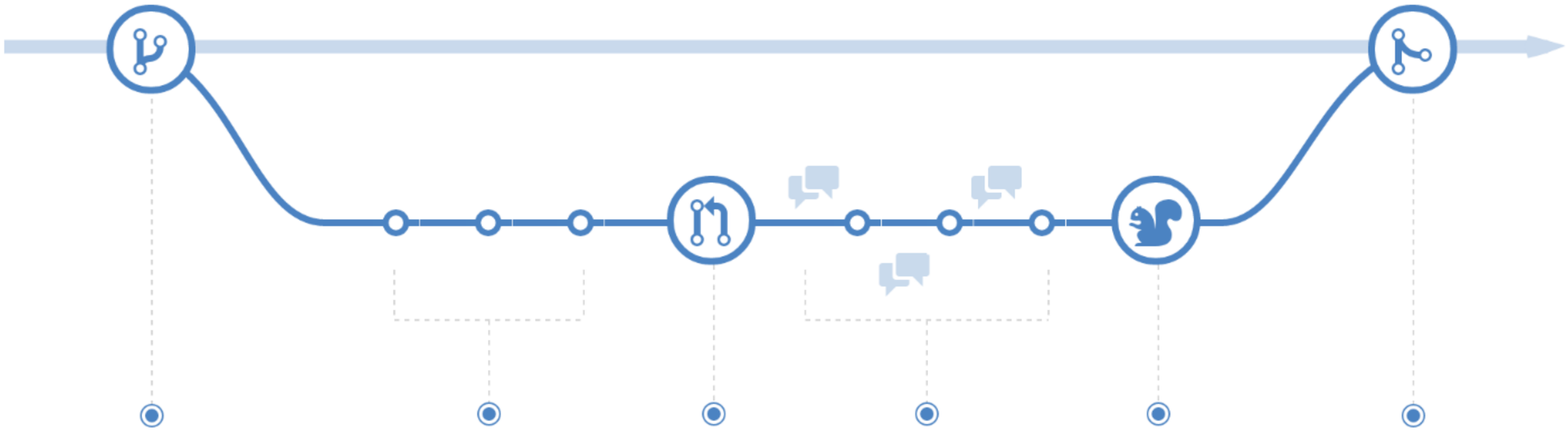


Git Feature Branch Workflow

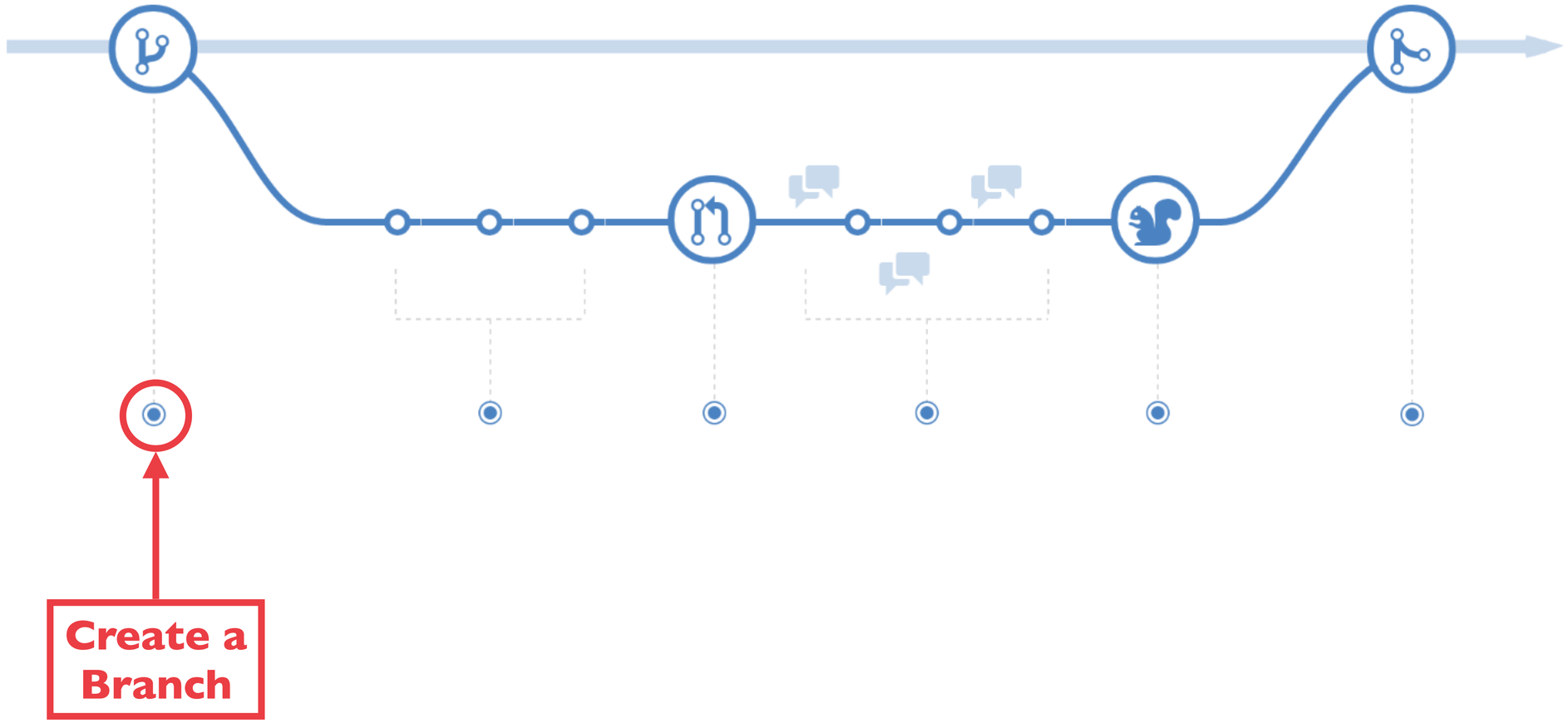
1. CREATE a new repository or FORK an exiting repository
2. CLONE it to your local system
3. Create a BRANCH to work on your ISSUE
4. COMMIT changes to that branch
5. PUSH your changes to the Remote branch
6. Issue a PULL REQUEST to have your work reviewed
7. MERGE your code to master and close the ISSUE



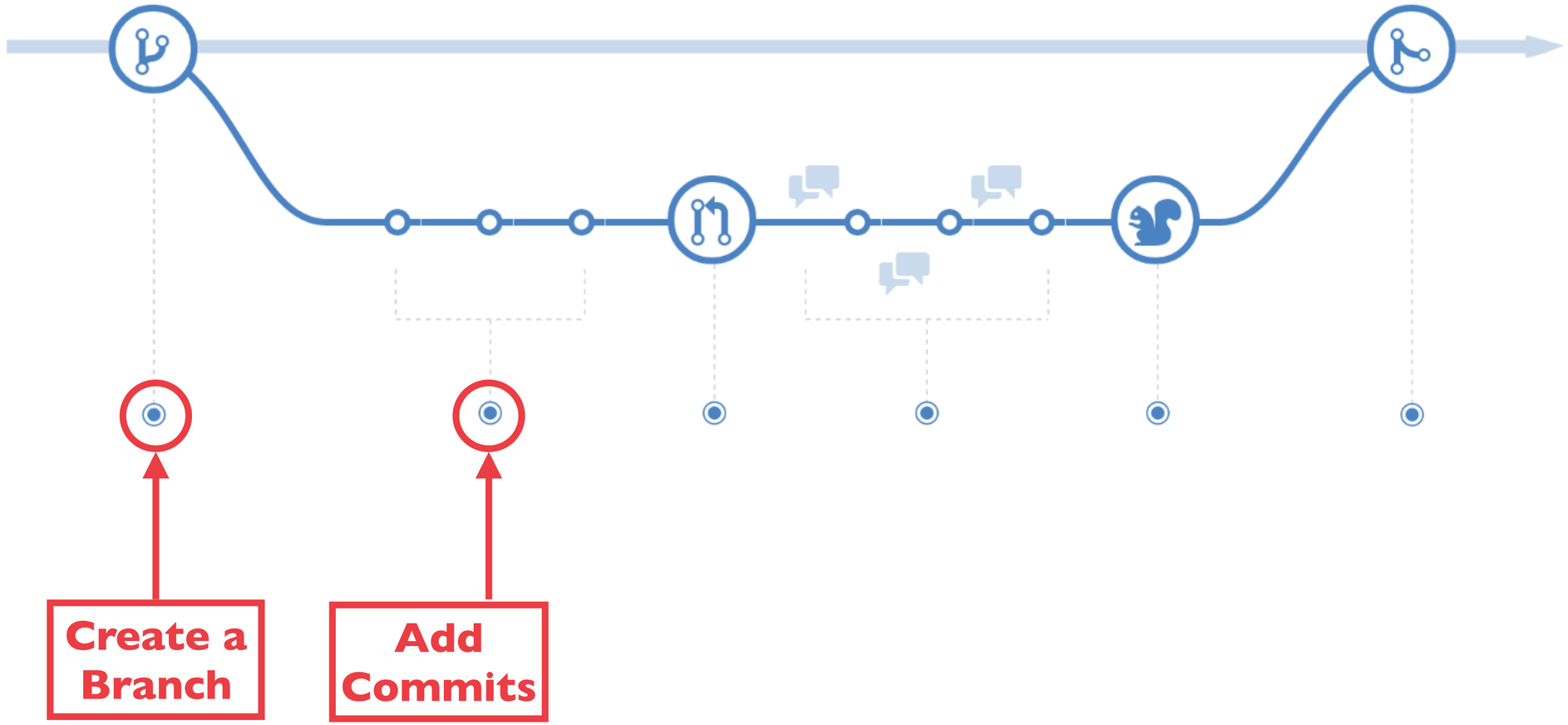
Git Feature Branch Workflow



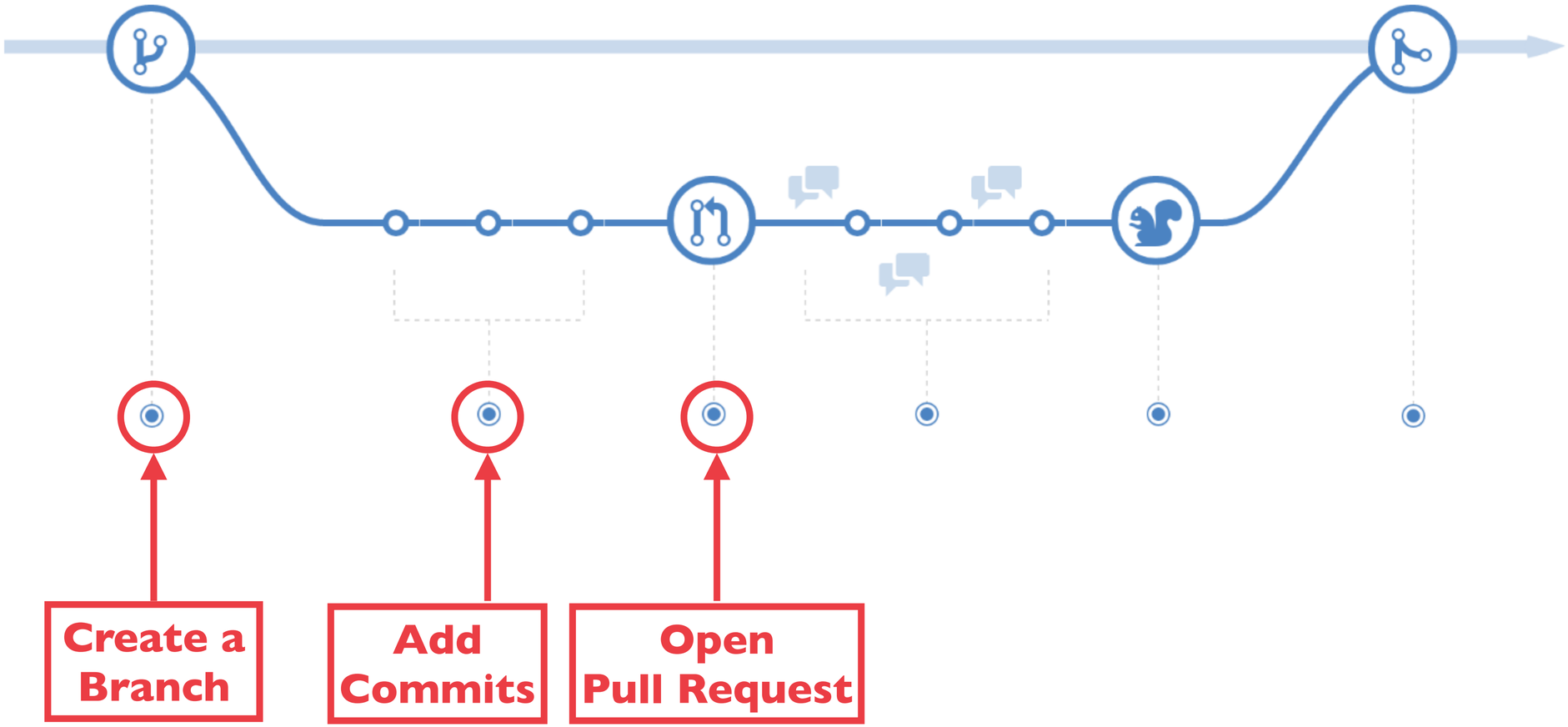
Git Feature Branch Workflow



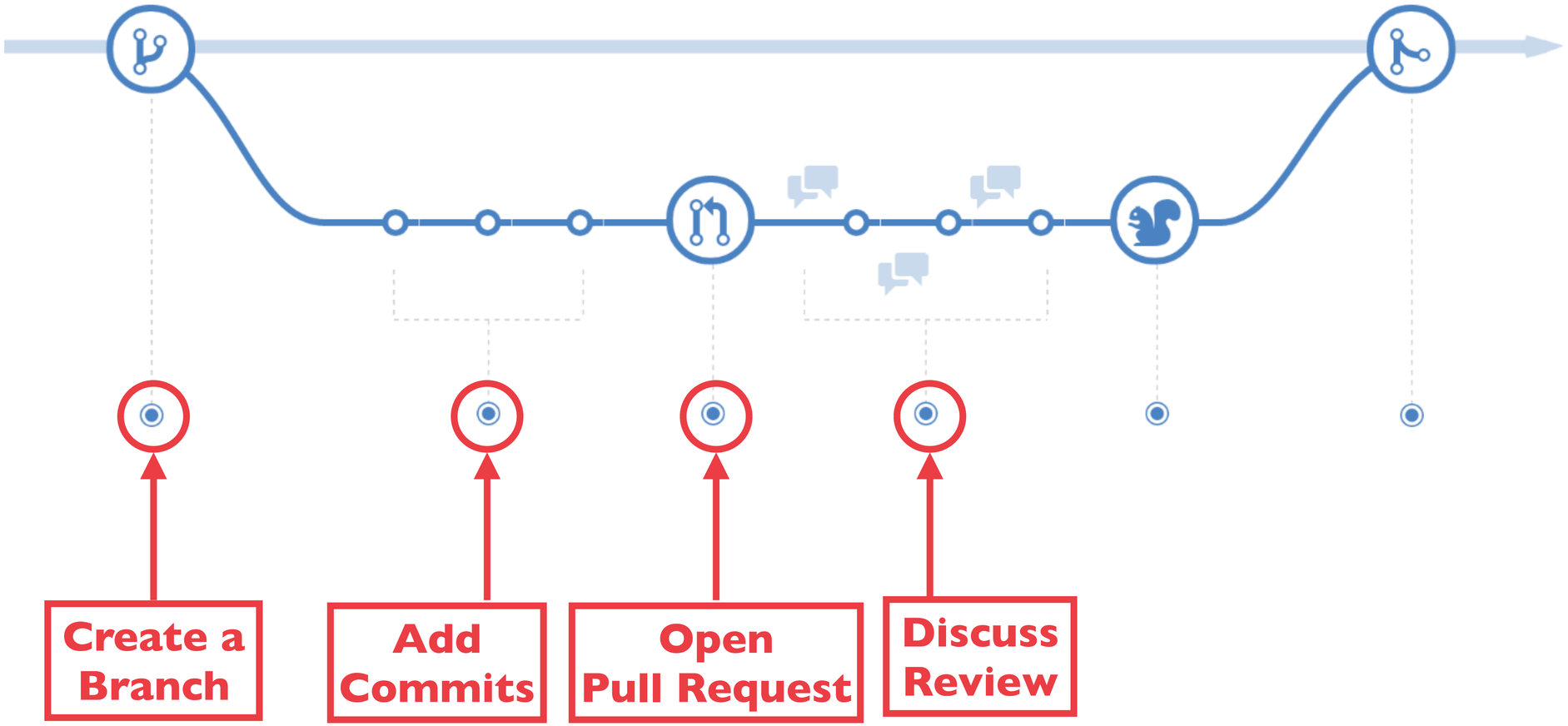
Git Feature Branch Workflow



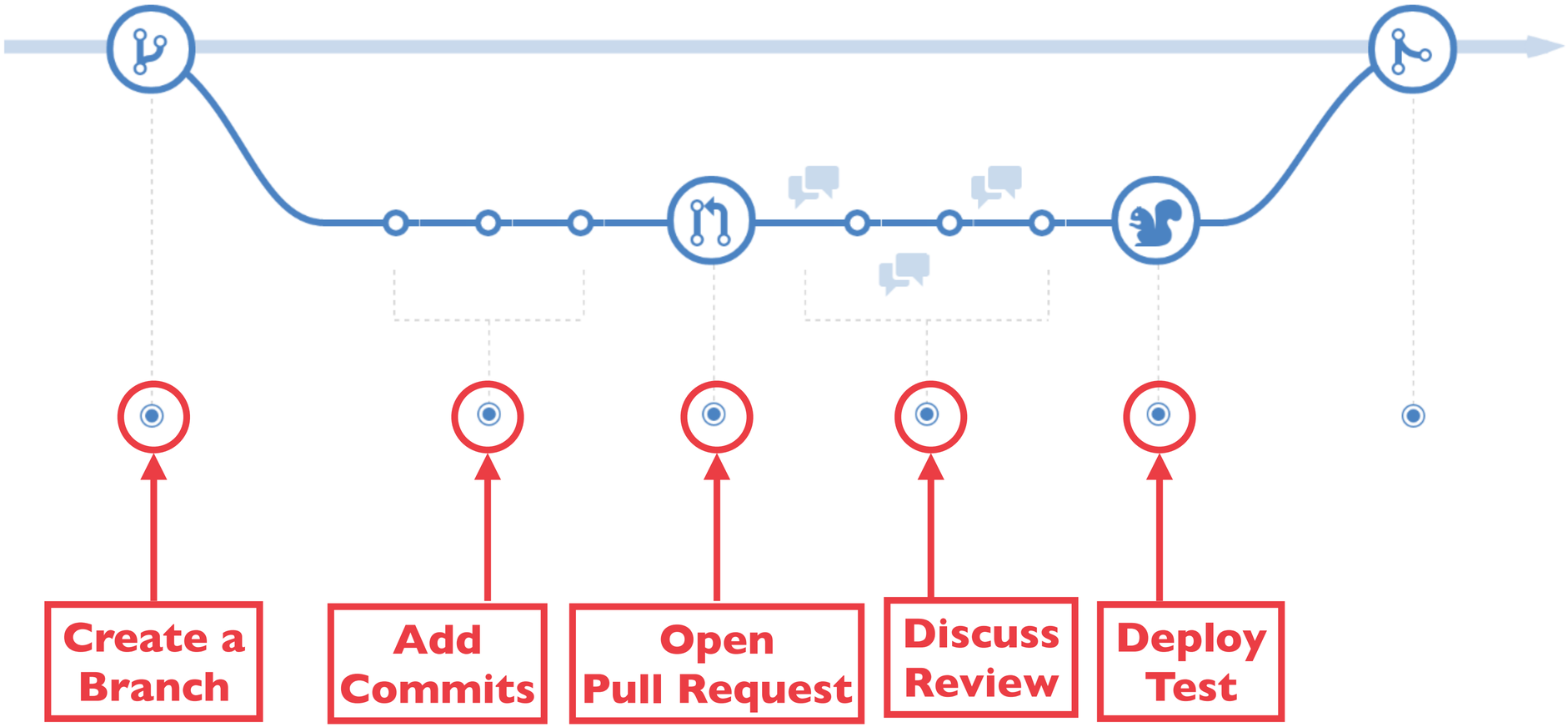
Git Feature Branch Workflow



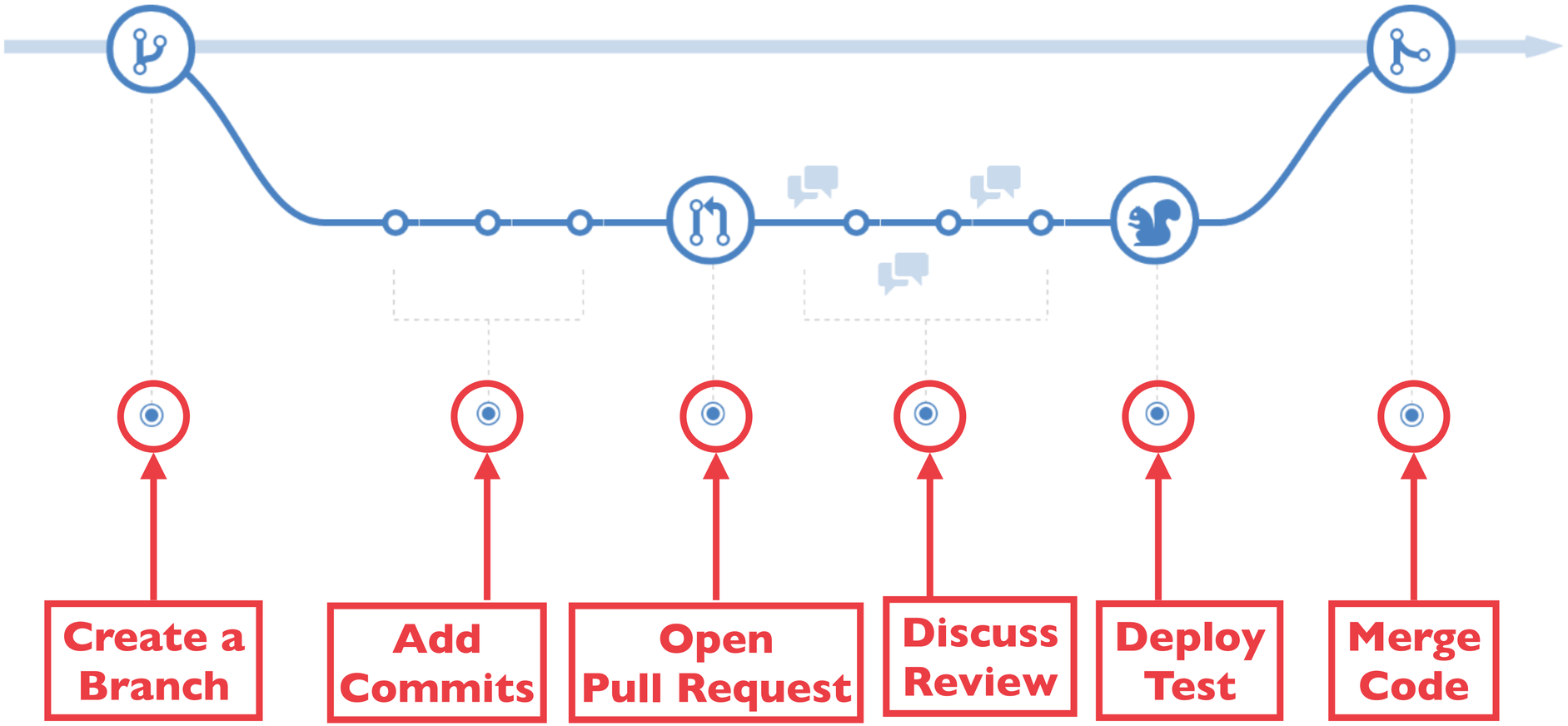
Git Feature Branch Workflow



Git Feature Branch Workflow

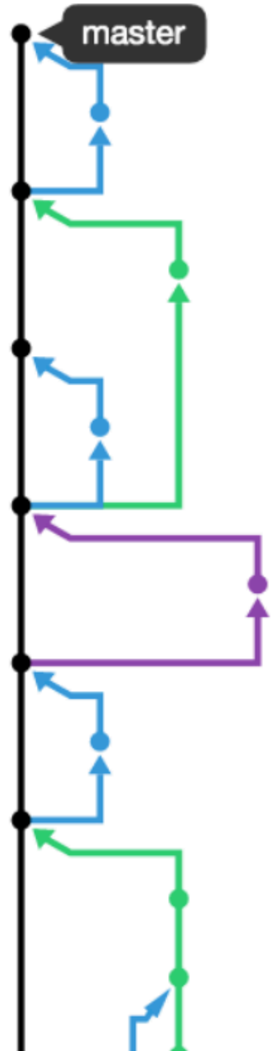


Git Feature Branch Workflow



Detailed Feature Branch Workflow

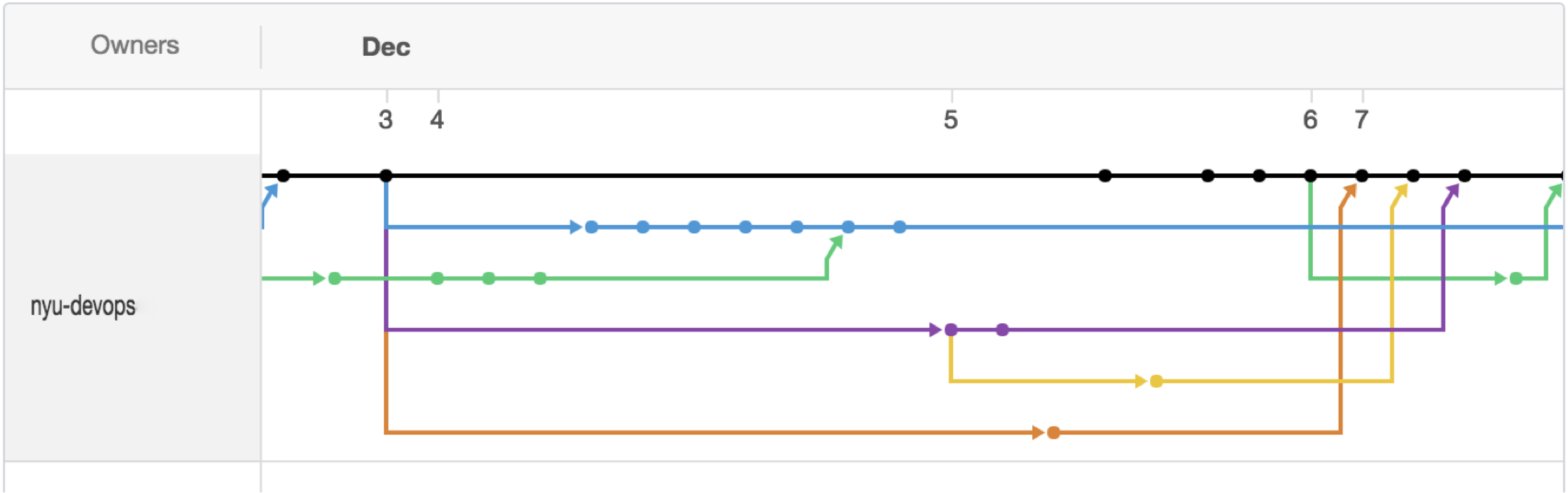
- CLONE a Repository (FORK first if not part of Dev Team)
- Assign the ISSUE for your work to yourself and place it in working status
- Create a BRANCH to work on an ISSUE
- Run the Test suite to make sure you can run the code
- Make changes to code and test cases and COMMIT to local BRANCH
- Run the Test suite early and often to make sure you didn’t break anything
- PUSH changes to remote BRANCH
- Did we mention testing the code early and often?
- Create PULL REQUEST when all tests pass and code is ready for review / MERGE



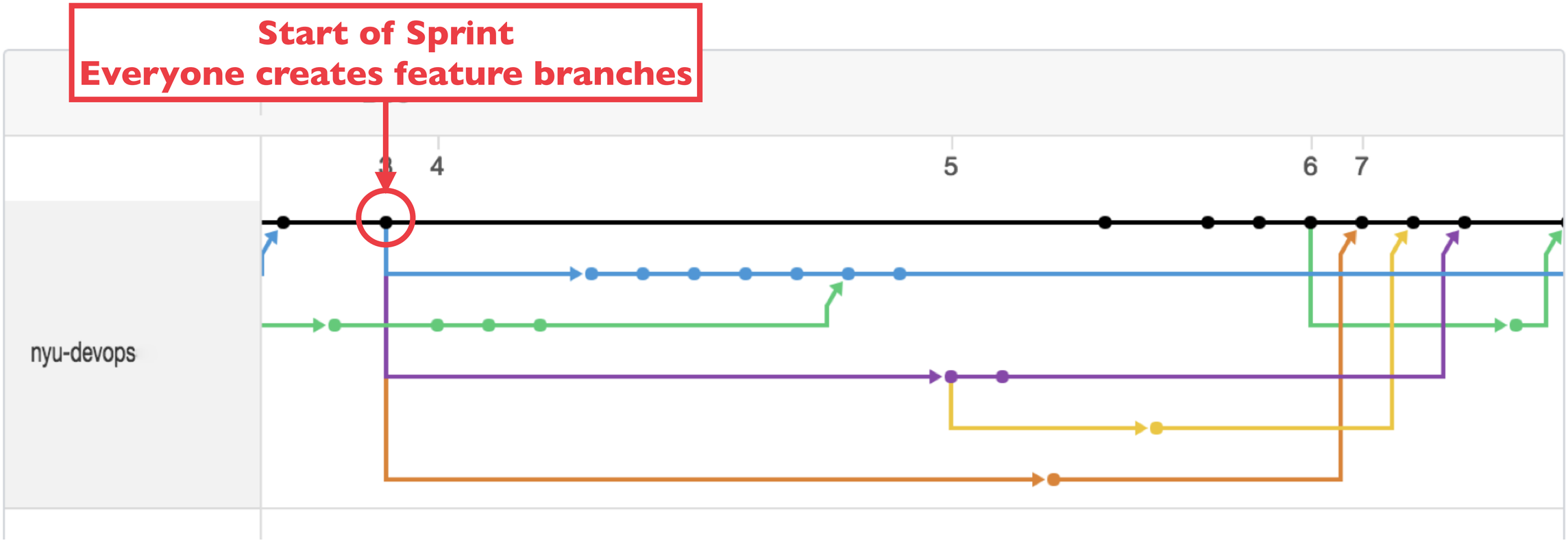
Why is Testing So Important?

- Automated testing must pass before your Pull Request will be accepted
- Automated testing is your guarantee that you didn’t break anything
- Without automated testing you cannot fully automate the DevOps pipeline

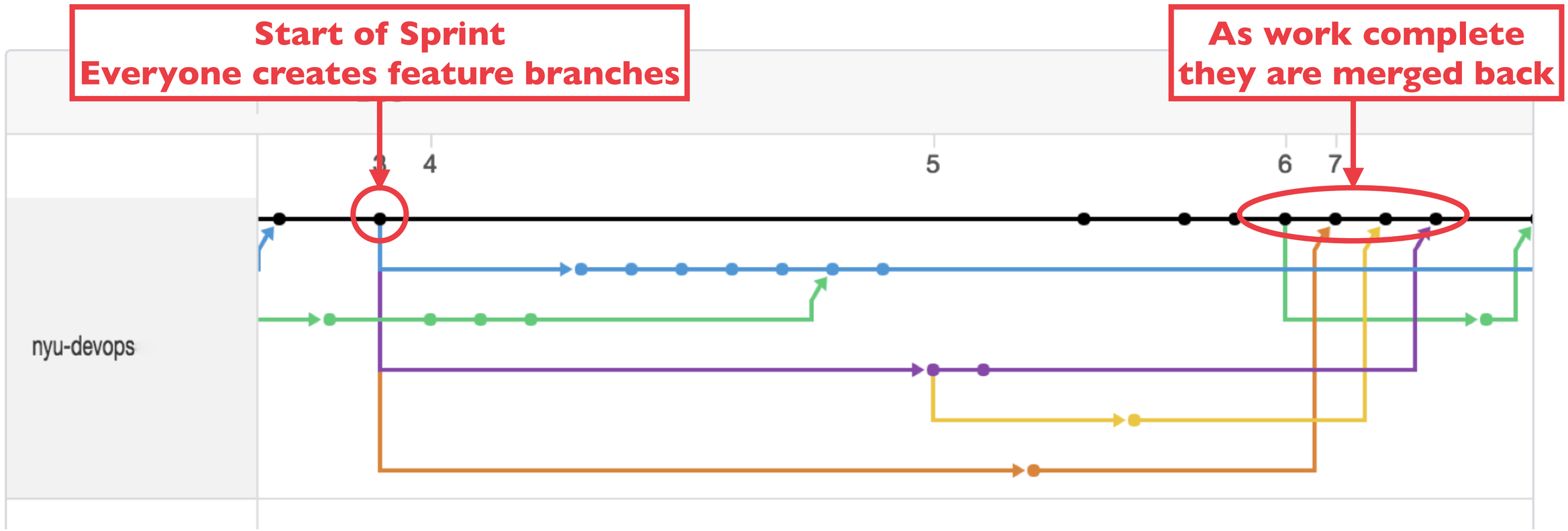
Branching example



Branching example



Branching example



Example workflow

- Your initial workflow should set up a remote branch that is tracked

```
git checkout -b my-new-feature
... write some code here ...
git add .
git commit -m 'initial working version of my new feature'
git push -u origin my-new-feature
```

- The next day you add more code and push to your remote branch

```
... write some code here ...
git add .
git commit -m 'added this cool code...'
git push
```

Example workflow

- Your initial workflow should set up a remote branch that is tracked

```
git checkout -b my-new-feature
... write some code here ...
git add .
git commit -m 'initial working version of my new feature'
git push -u origin my-new-feature
```

Sets up a remote branch and tracks it

- The next day you add more code and push to your remote branch

```
... write some code here ...
git add .
git commit -m 'added this cool code...'
git push
```

Example workflow

- Your initial workflow should set up a remote branch that is tracked

```
git checkout -b my-new-feature
... write some code here ...
git add .
git commit -m 'initial working version of my new feature'
git push -u origin my-new-feature
```

Sets up a remote branch and tracks it

- The next day you add more code and push to your remote branch

```
... write some code here ...
git add .
git commit -m 'added this cool code   '
git push
```

No need to use the branch name because it is tracked

Example workflow before Pull Request

- Before making a Pull Request always get the latest changes from master

```
git checkout master
git pull
git checkout my-new-feature
git merge master
... fix any merge conflicts here ...
git add .
git commit -m 'merged updates from master'
git push
```

Example workflow before Pull Request

- Before making a Pull Request always get the latest changes from master

```
git checkout master
git pull
git checkout my-new-feature
git merge master
... fix any merge conflicts here ...
git add .
git commit -m 'merged updates from master'
git push
```

Example workflow before Pull Request

- Before making a Pull Request always get the latest changes from master

```
git checkout master
git pull
git checkout my-new-feature
git merge master
... fix any merge conflicts here ...
git add .
git commit -m 'merged updates from master'
git push
```

Pull the latest changes from master

Merges any changes from master with your branch

Hands-On

“live follow-along session”