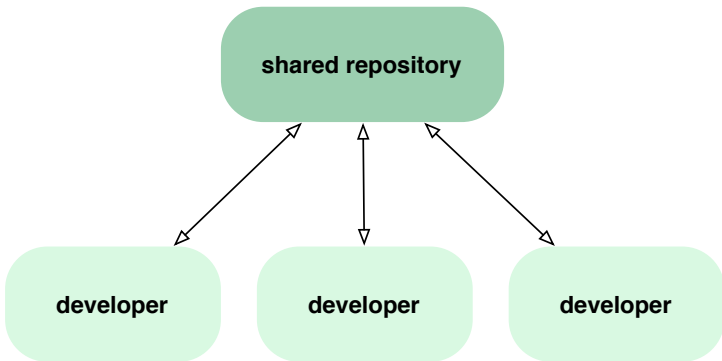# A Practical Introduction to Git

Maktab 64 - Reza Yazdani
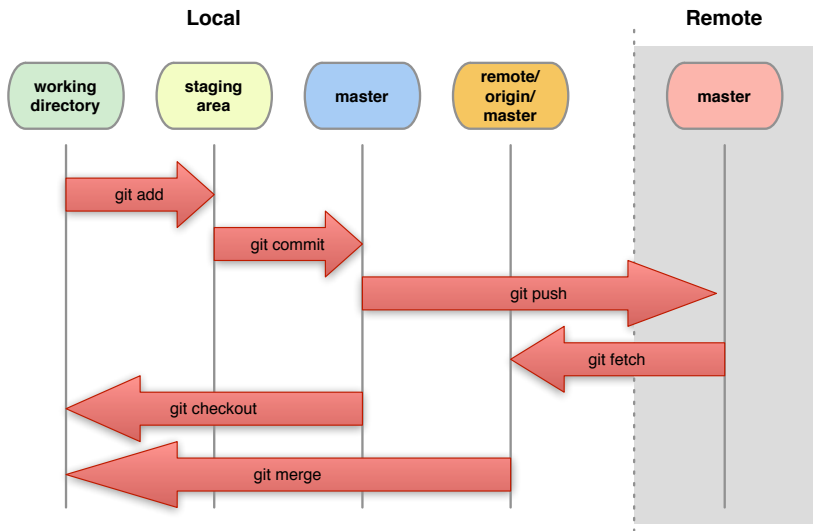
multi+remote/shared `git`.

Scenario 2: multiple developers + remote central repository.

multi+remote/shared `git`.

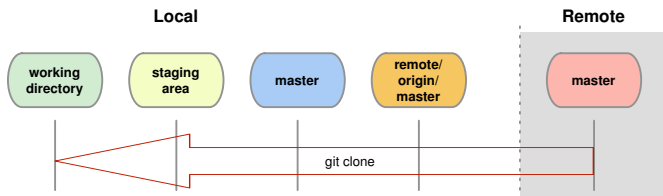# multi+remote/shared `git`.

multi+remote/shared `git`.

`git clone <URL>`

Creates *two* local copies of the whole remote repository.

Available transport protocols:

- `ssh://`, `git://`, `http://`, `https://`, `file://`
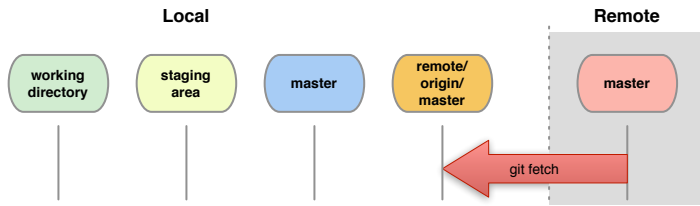
Ex.: `git clone https://github.com/ASPP/pelita.git`

`git remote -v`

shows **name** and `URL` of the remote repository.

multi+remote/shared `git`. Fetch.

`git fetch`

- Downloads updates from remote master to local remote master.
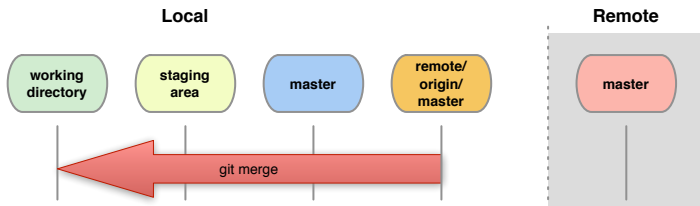- The local master, staging area and working directory do not change.

Q: Why **origin**?

A: Just a label for **Remote**. Choose the one you like.
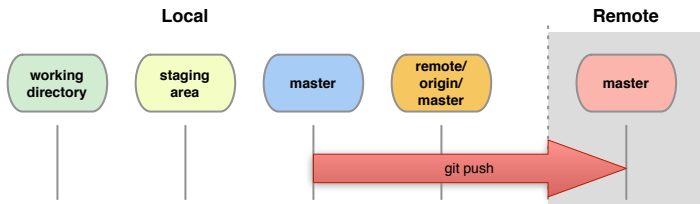
multi+remote/shared `git`. Merge.

`git merge`

- Joins development histories together.
- Warning: can generate *conflicts*!
- **Note**: it merges only when all changes are committed.

`git fetch` + `git merge` = `git pull`

git push

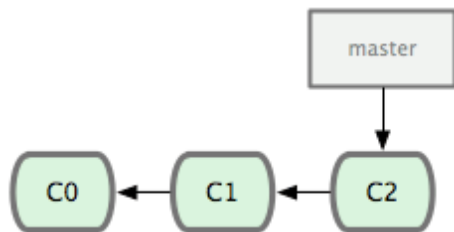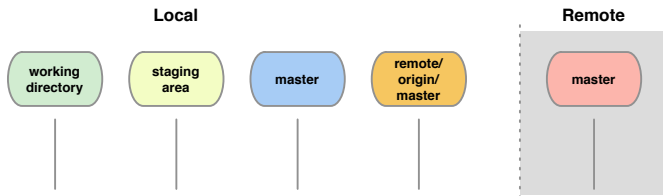- Updates *remote masters* (both Local and Remote).
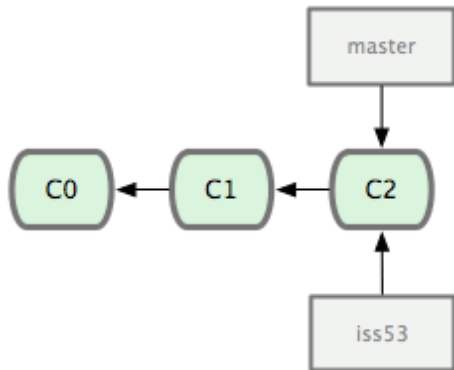- Requires **fetch**+**merge** first.

basic branching

Branching.


```
git commit (C0)
git commit (C1)
git commit (C2)
```

Branching.

git branch iss53
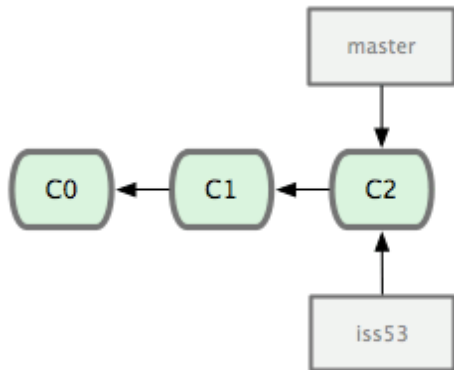git checkout iss53
git checkout master
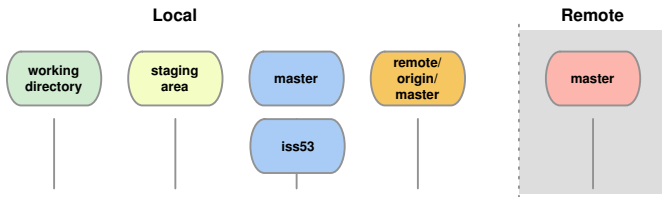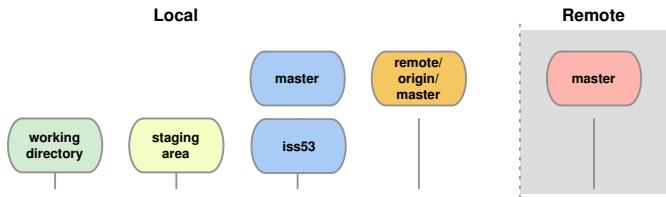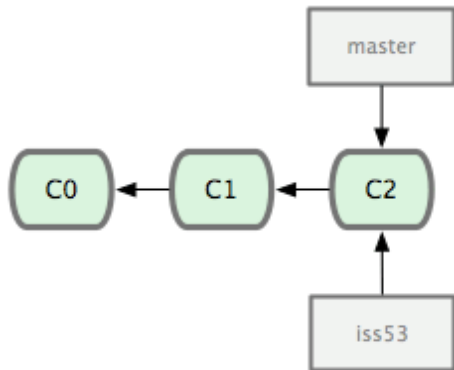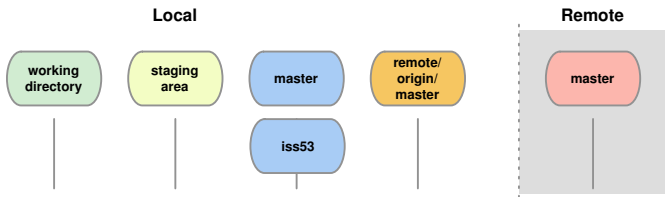
git branch iss53
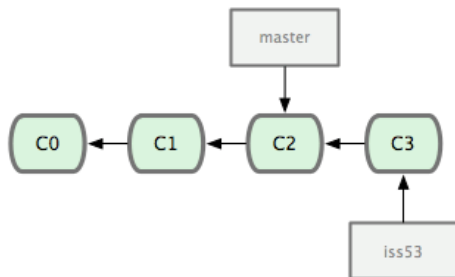**git checkout iss53**
git checkout master

Branching.

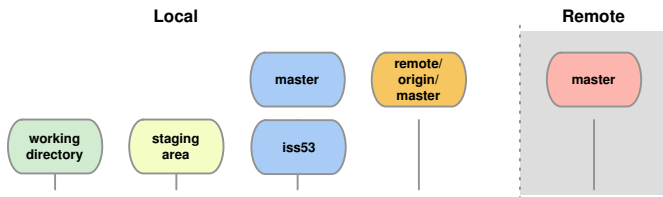

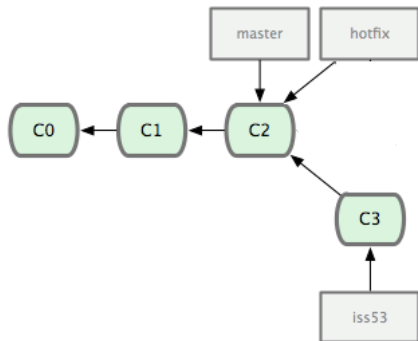**git branch iss53**
git checkout iss53
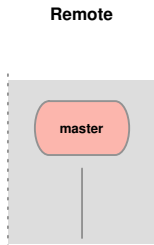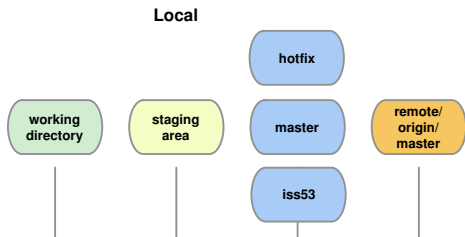**git checkout master**

Branching.


```
git checkout iss53
   git commit (C3)
```
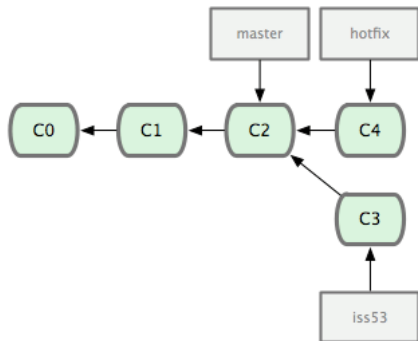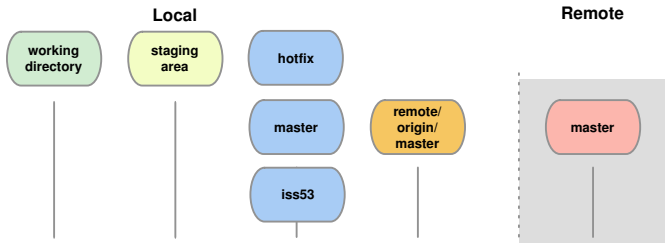
Branching.

git branch hotfix
git checkout hotfix
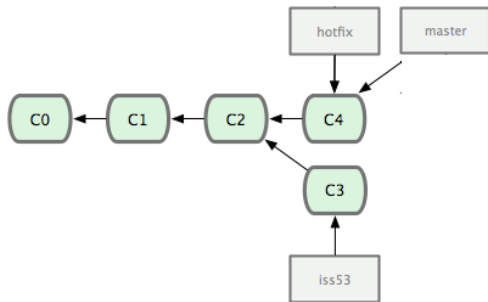git commit (C4)

Branching.

git branch hotfix
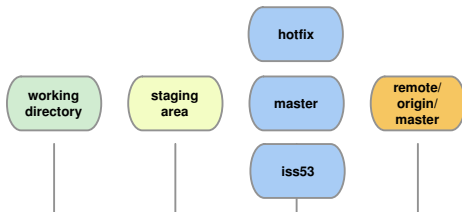**git checkout hotfix**
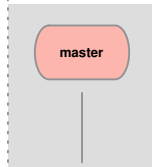**git commit (C4)**

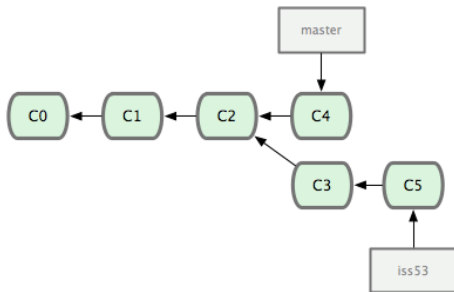# Branching.


```
git checkout master
  git merge hotfix
```
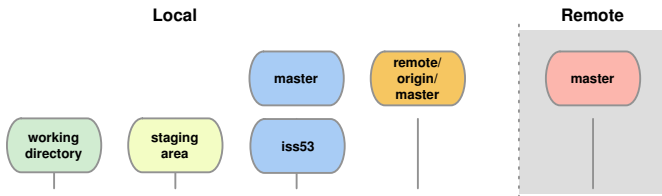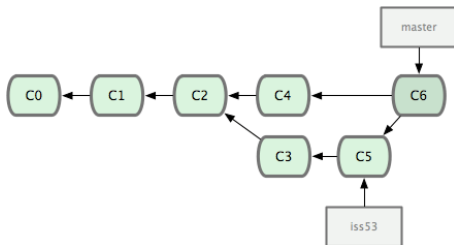
Branching.

`git checkout iss53`
`git commit (C5)`

Branching.

`git checkout master`
`git merge iss53`