The Abdus Salam International Centre for Theoretical Physics

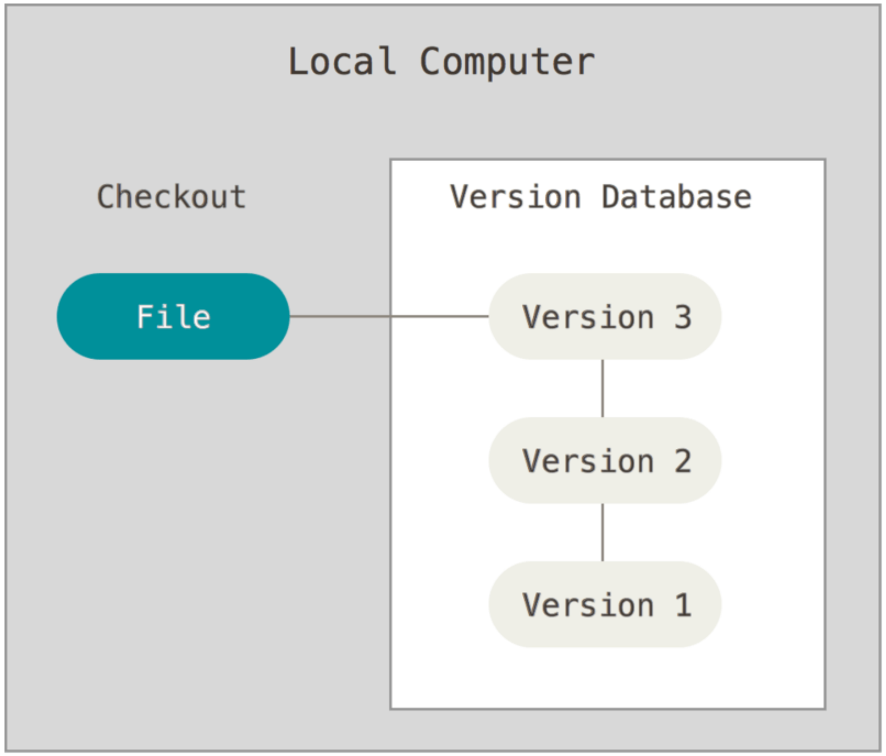
Introduction to Git

Ivan Girotto – igirotto@ictp.it
Information & Communication Technology Section (ICTS)
International Centre for Theoretical Physics (ICTP)

Human Version Control

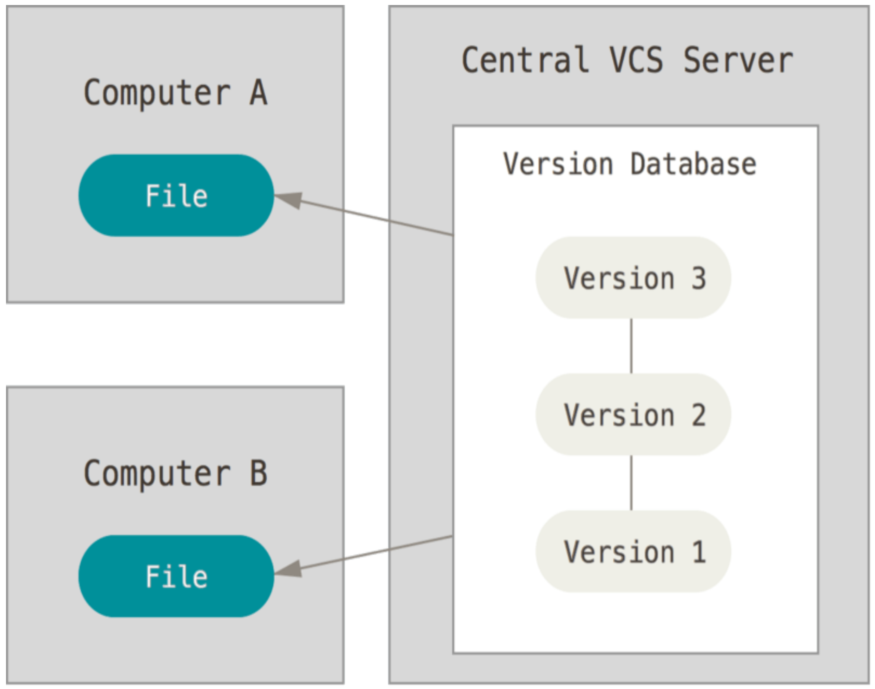
- Only one person makes changes one after the other is complete
- No need for source code management software:
  - Make a copy of the code
  - Add new features, test if working
  - Modified copy becomes new master version
- What happen if we need to go back?!

Local Version Control Systems



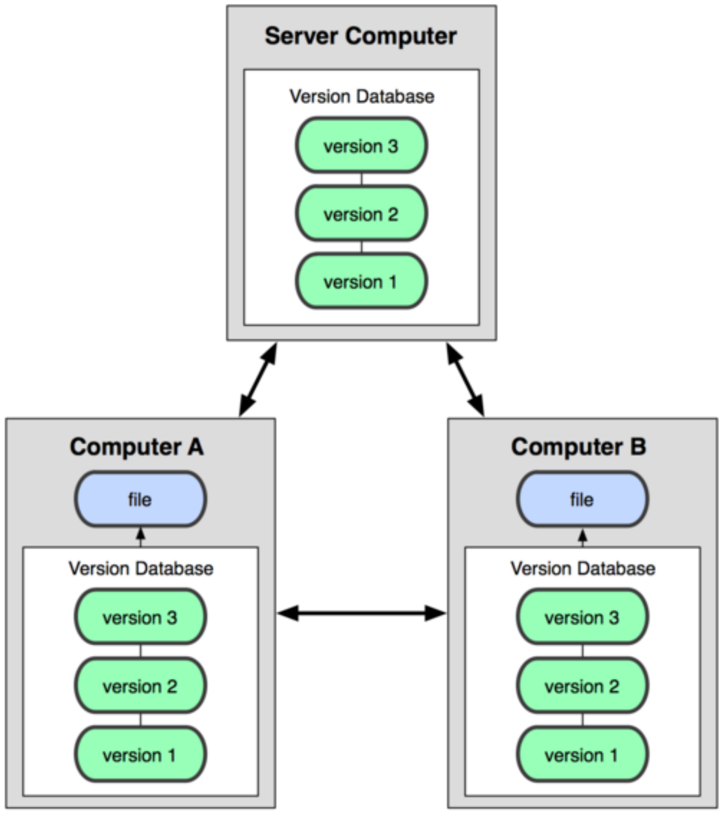
- Early days fashion style
- Mainly for a single developer
- Older status can be easily recovered
- RCS, early 80s

Centralized Version Control Systems



- From the 90s
- Serialized when committing changes back into the repository
- Only committed changes generate a history
- CVS, SVN

Distributed Version Control Systems



- Modern Version Control
- Every developers own a full copy of the main trunk locally
- Used as tool for SW development management in most modern software packages

Terminology: Working Copy

- It is where you are currently working
- Every new modification makes your working copy a new "version" of the code
- Nothing is saved up to the next operation of "saving"
- the git status command logs the status of your working copy
- the git diff command show the differences since your last operation of "saving"

Terminology: Staging Area

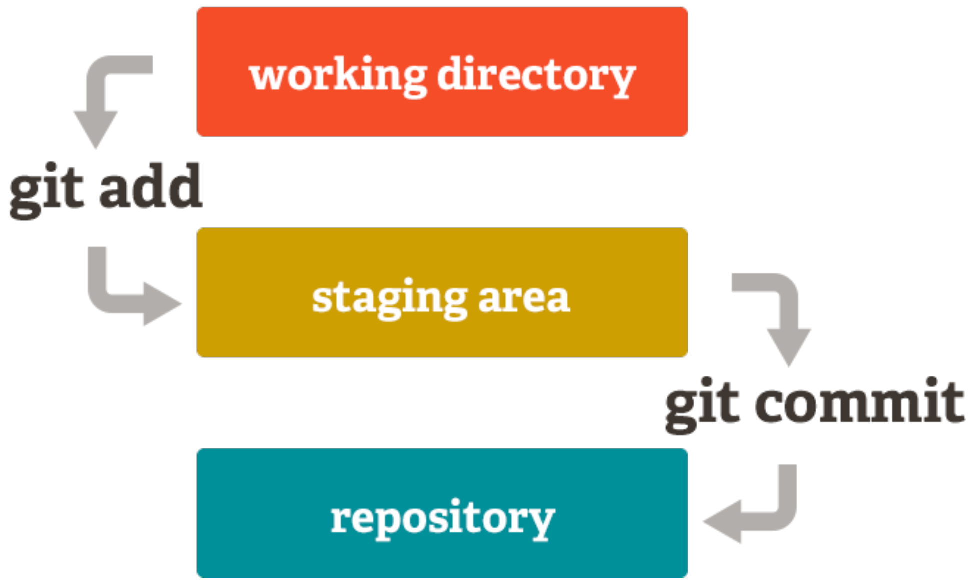
- Only the new modifications which are “staged” can be included into an operation of saving
- The command git add includes the new modifications to exiting files/new files into the staging area, to be included into the next operation of “saving”

Terminology: commit

- It is the operation of “saving”
- The command git commit executes the operation
- It is a good attitude to include a meaningful comment to each commit, in case you need to go back to previous status (which status?!)
- git commit -m “Insert here the message”

Terminology: repository (local, remote)

- It is used to identify the final destination of the new development



Terminology: branch /1

- In the jargon of development normally considered a deviation of the development from the main trunk
- A branches are a fundamental component of Git
- During an on-going development you are always working on a branch
- At the beginning two branches are available
  - master (the local repository)
  - origin/master (local copy of the remote repository when last imported/cloned)

Terminology: branch /2

- The command git branch -a logs the available local branches
- In Git every new development "session" should be made on a new branch
- Git allows to handle number of branches in parallel and easily switch among them
- the command git checkout *branchname* is used to move from the current branch to branch *branchname*

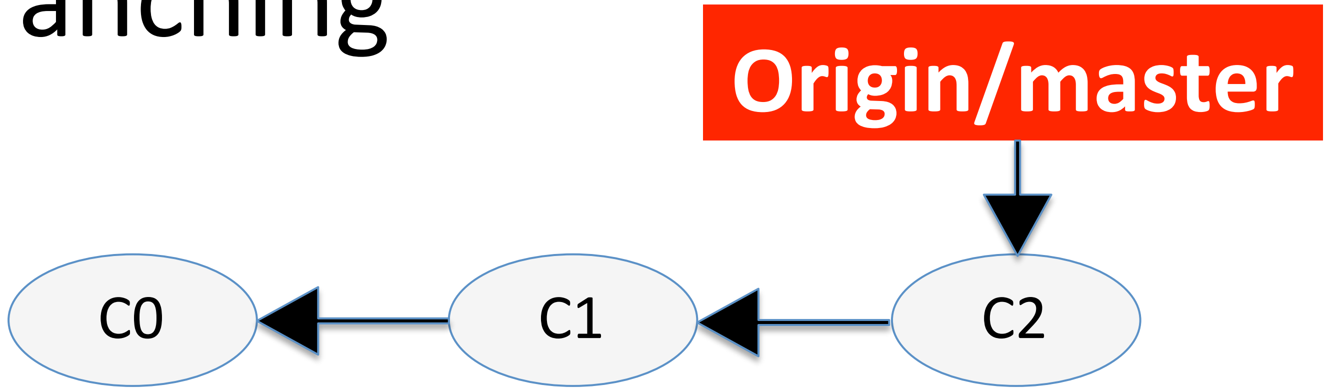
Terminology: branch /3

- the command git checkout -b *branchname* is used to move on a new branch named *branchname* (copy of the source branch)
- Once the session is terminated the new branch is merged into the master/ref. branch
- The command git merge *branchname* merge *branchname* into the current branch (here conflicts are automatically or manually resolved)

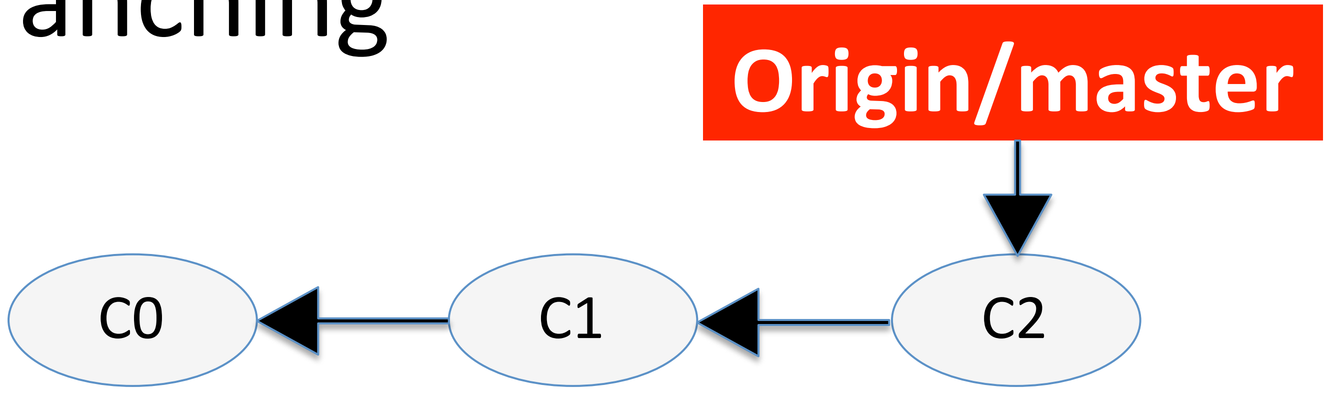
Terminology: pull/push

- Are the operation between the local and the remote repositories
- The command git push is used to send the new development included into the local repository to the remote repository
- The command git pull is used to retrieve the new development included in the remote repository, into the local repository (here conflicts are automatically or manually resolved)

Basic Branching

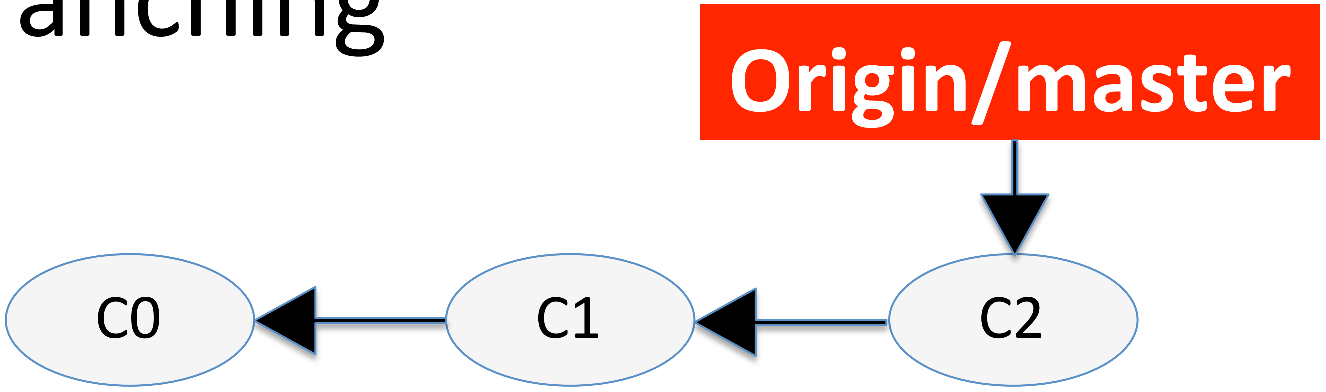


Basic Branching

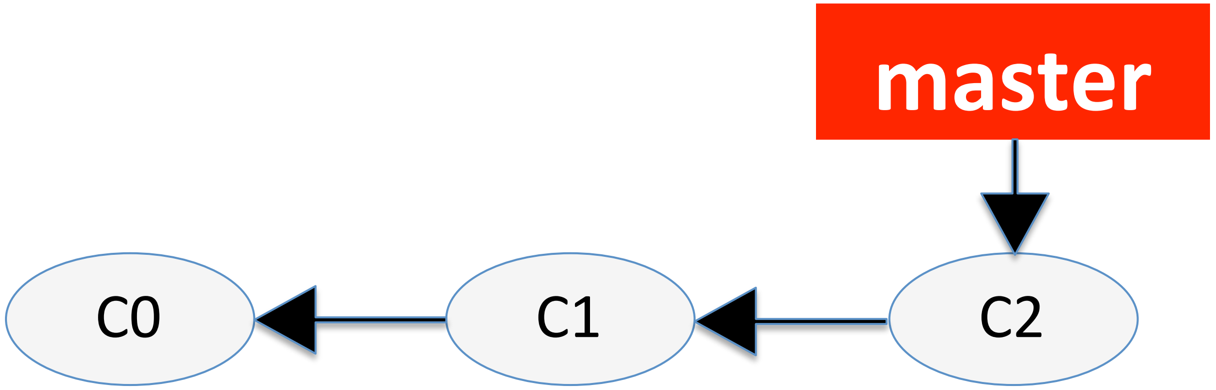


```
$git clone ....
```

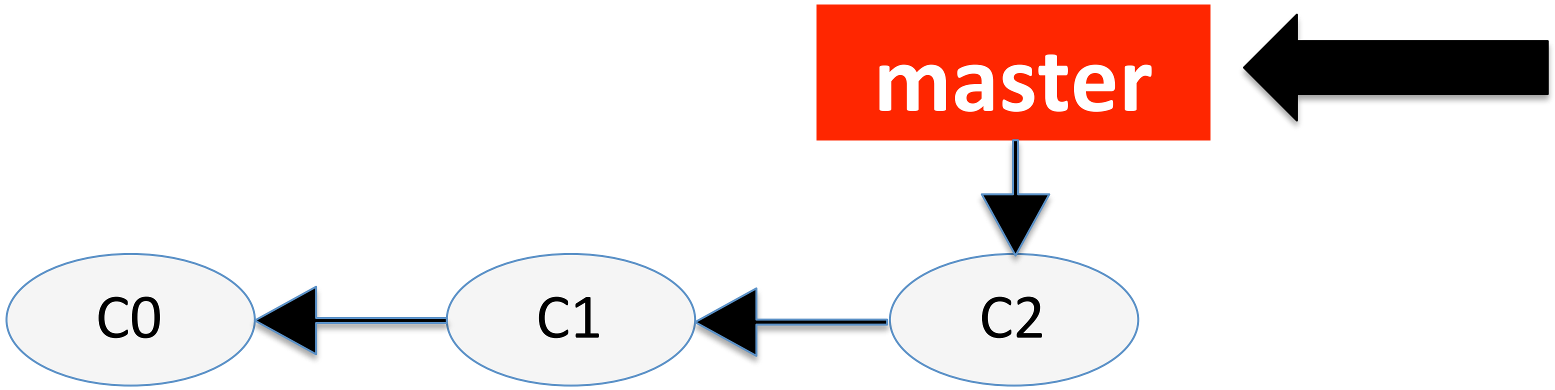
Basic Branching



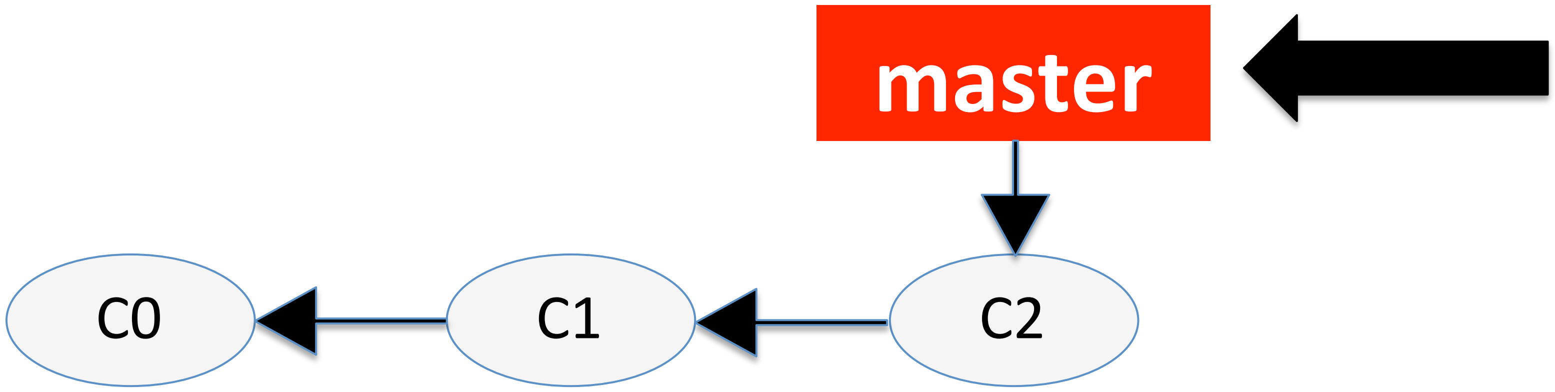
```
$git clone ....
```



Basic Branching

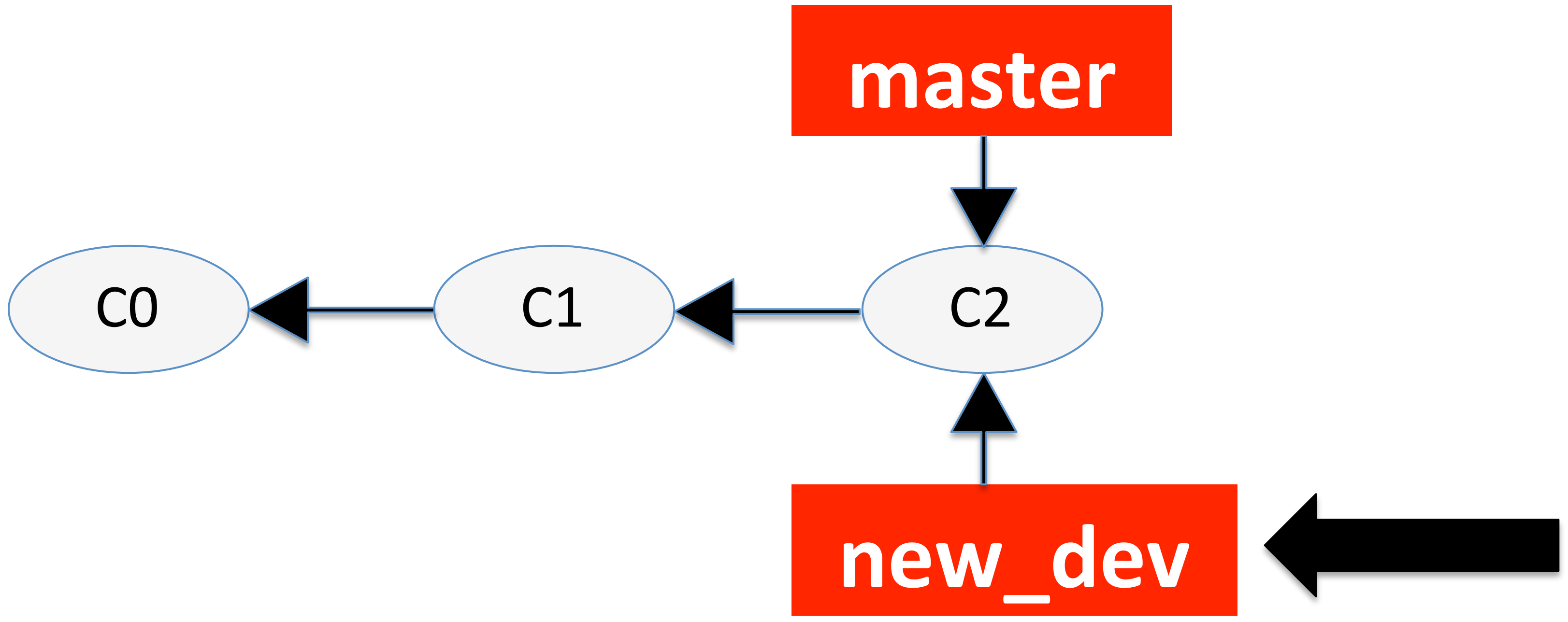


Basic Branching



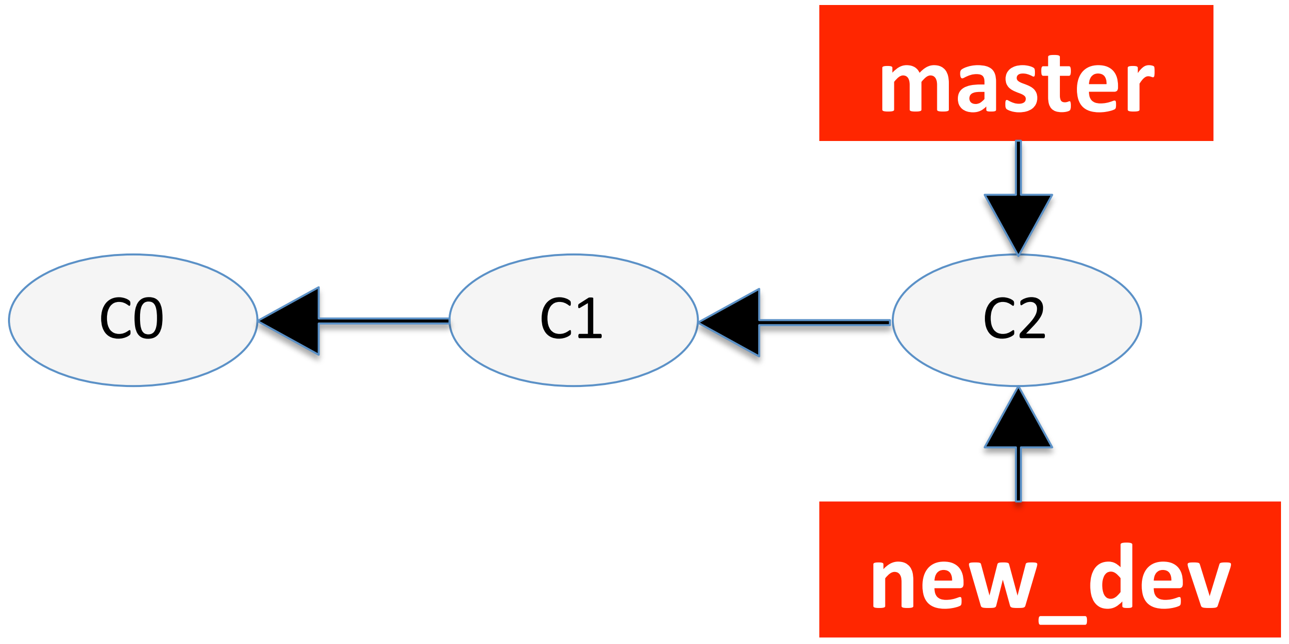
```
$git checkout -b new_dev
switched to a new branch "new_dev"
```

Basic Branching

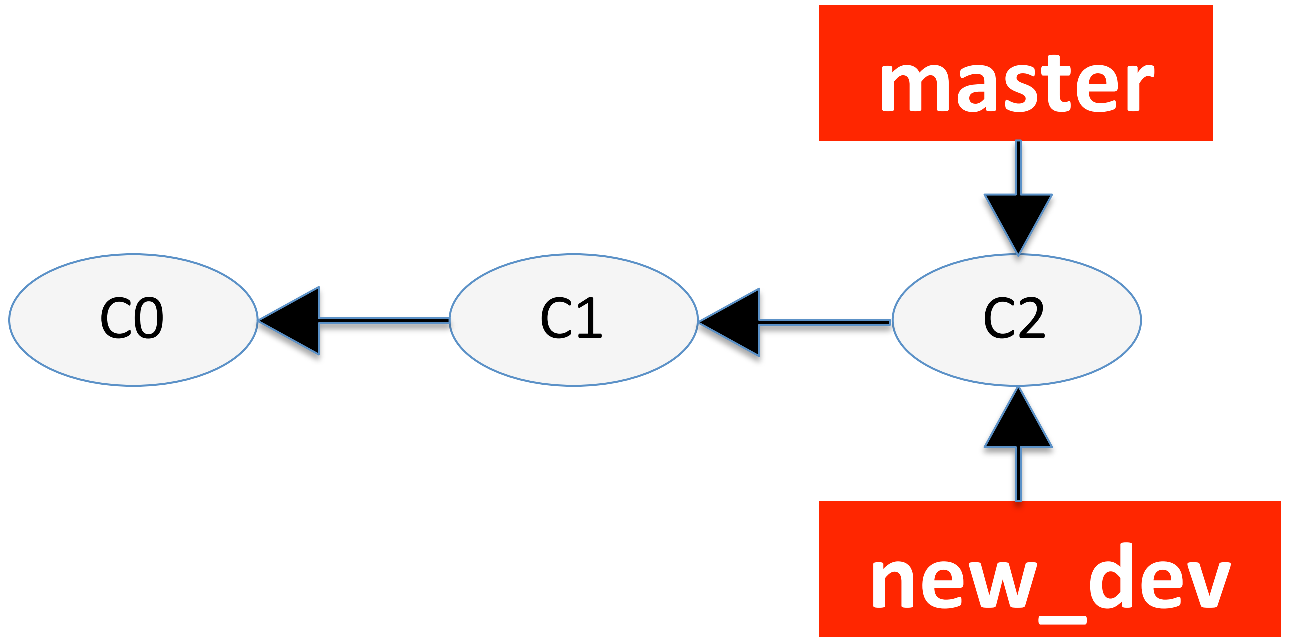


```
$git checkout -b new_dev
switched to a new branch "new_dev"
```

Basic Branching

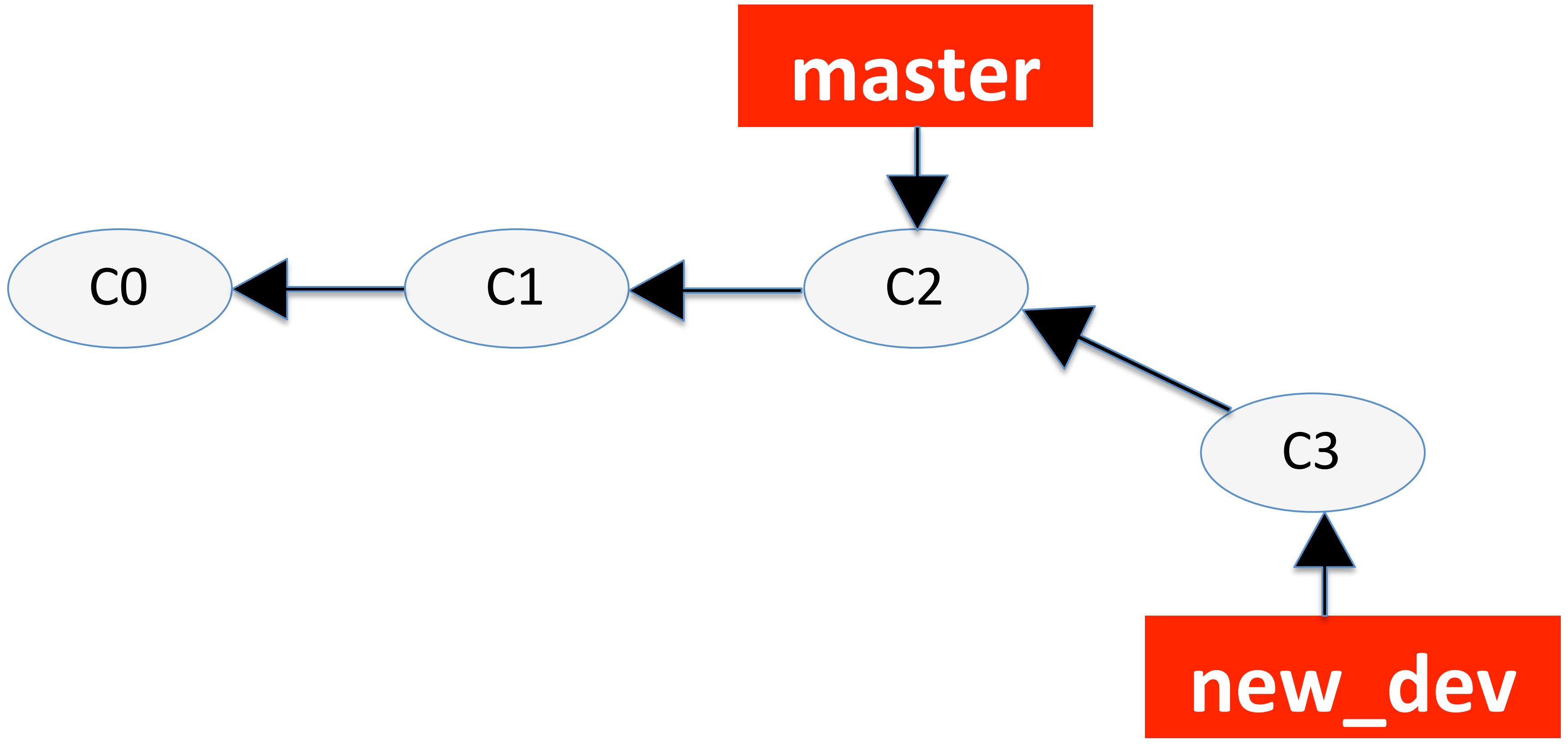


Basic Branching



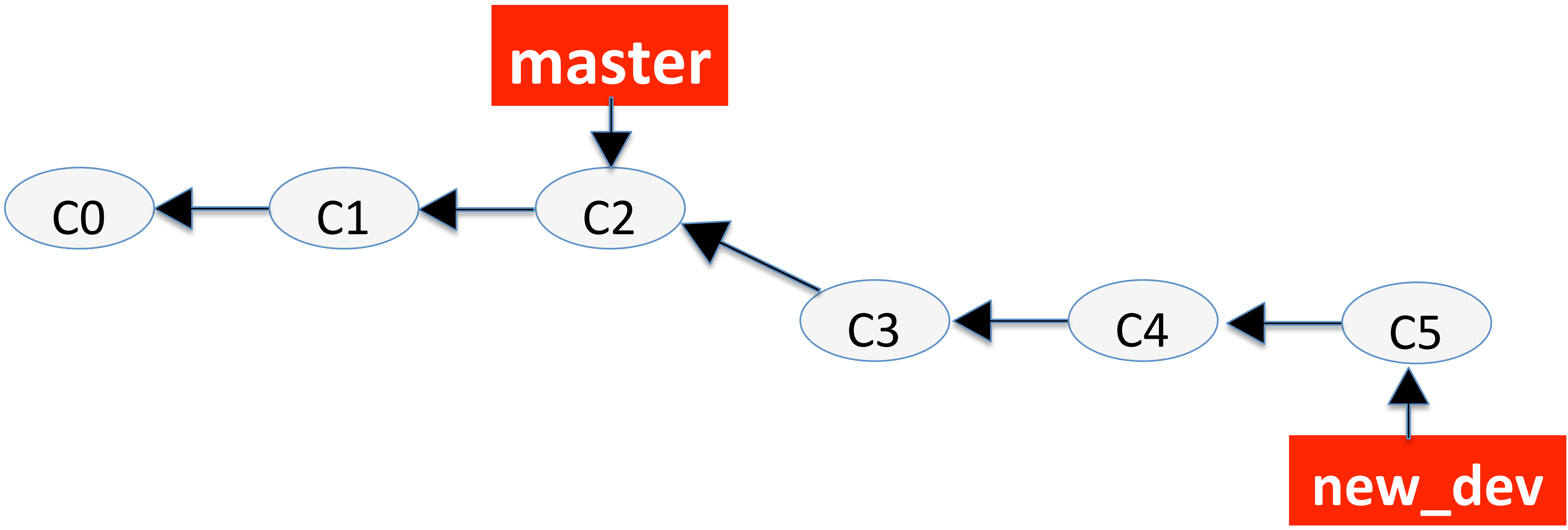
```
$git commit -m "new development"
```

Basic Branching

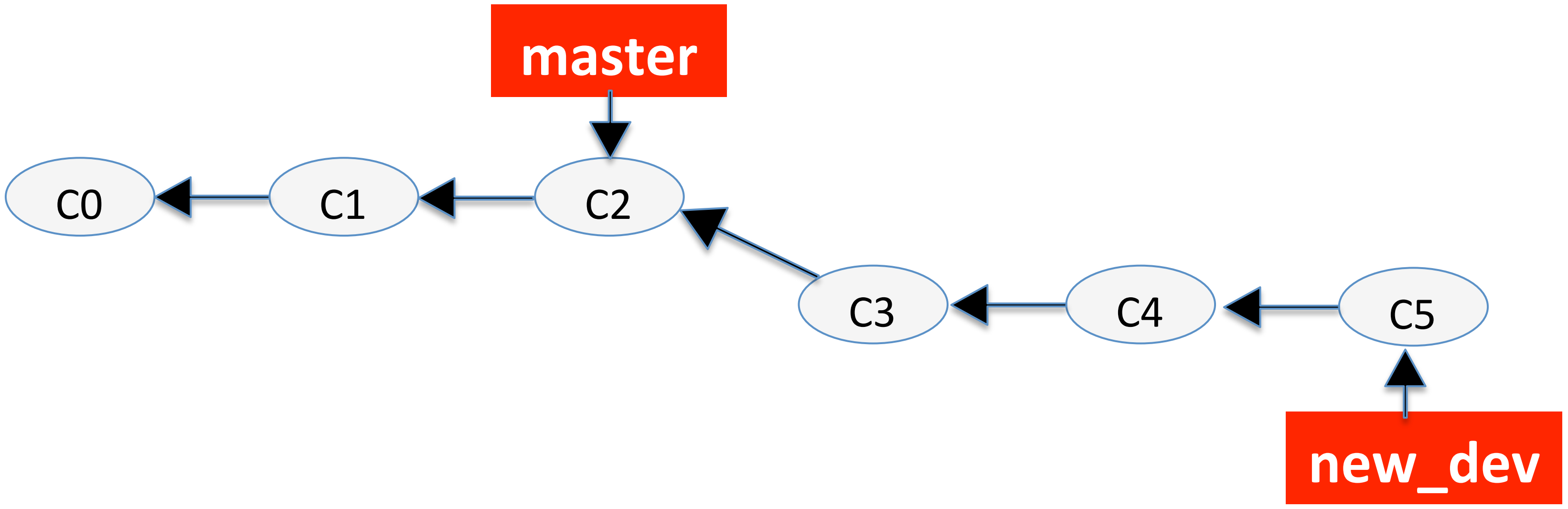


Basic Branching

After two more commits ...

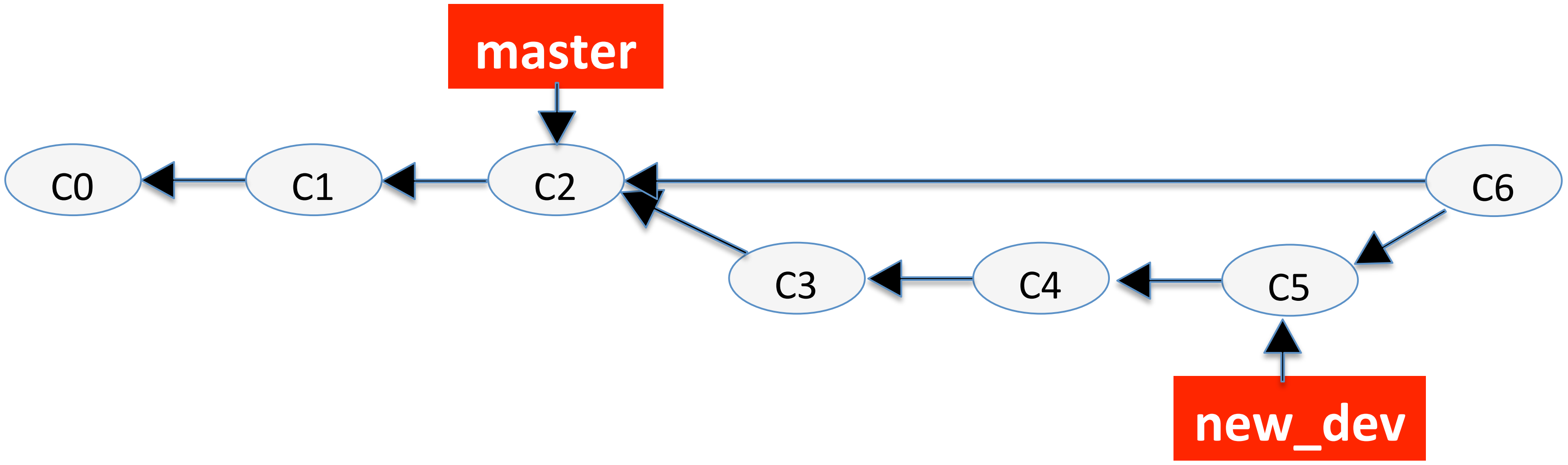


Basic Merging



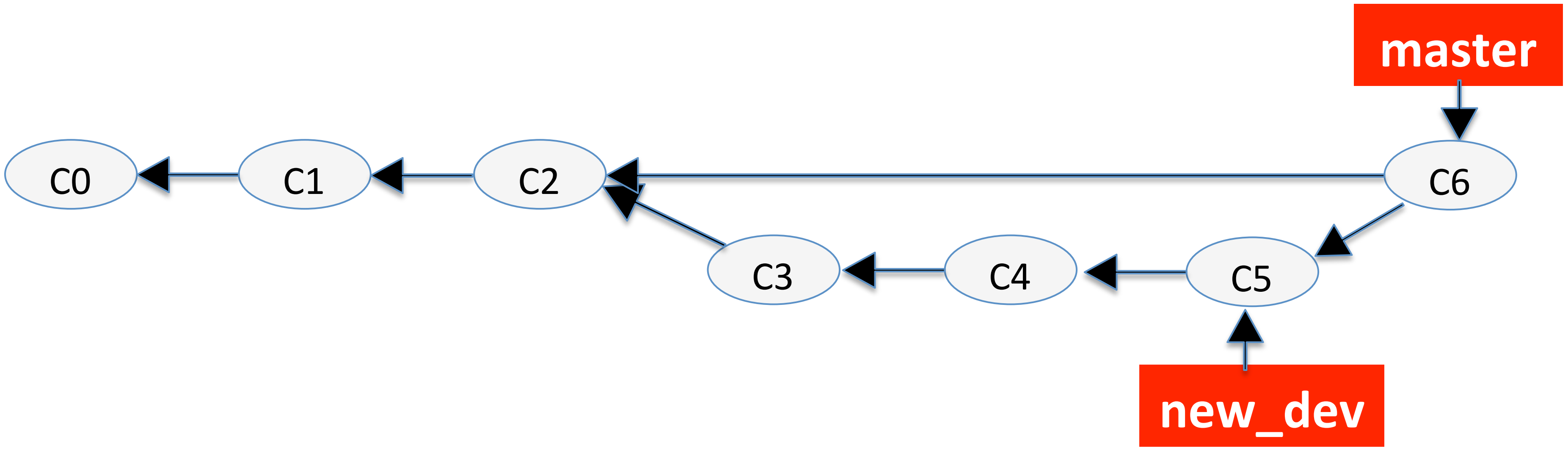
```
$git checkout master
```

Basic Merging



```
$git merge new_dev
```

Basic Merging



```
$git merge new_dev
```

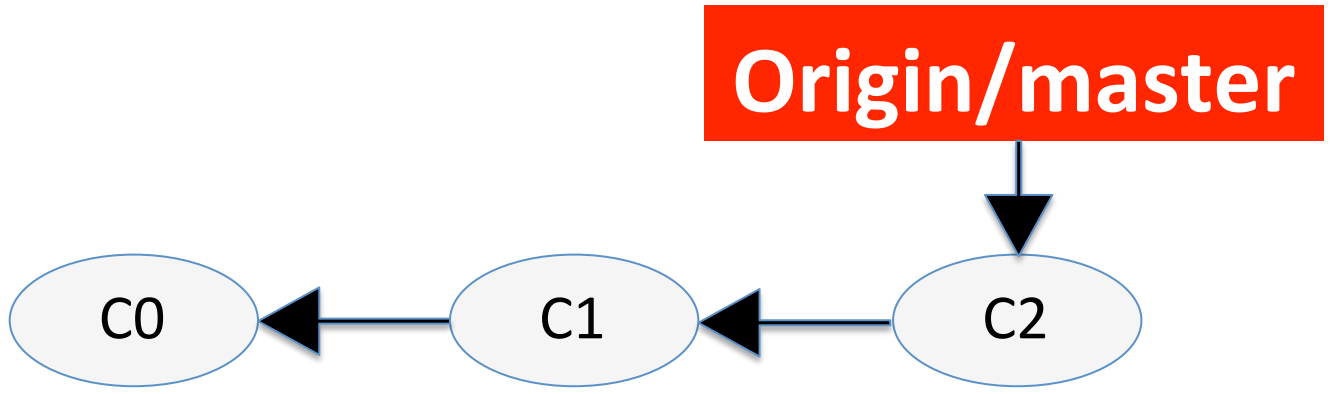
```
$git branch -d new_dev
```

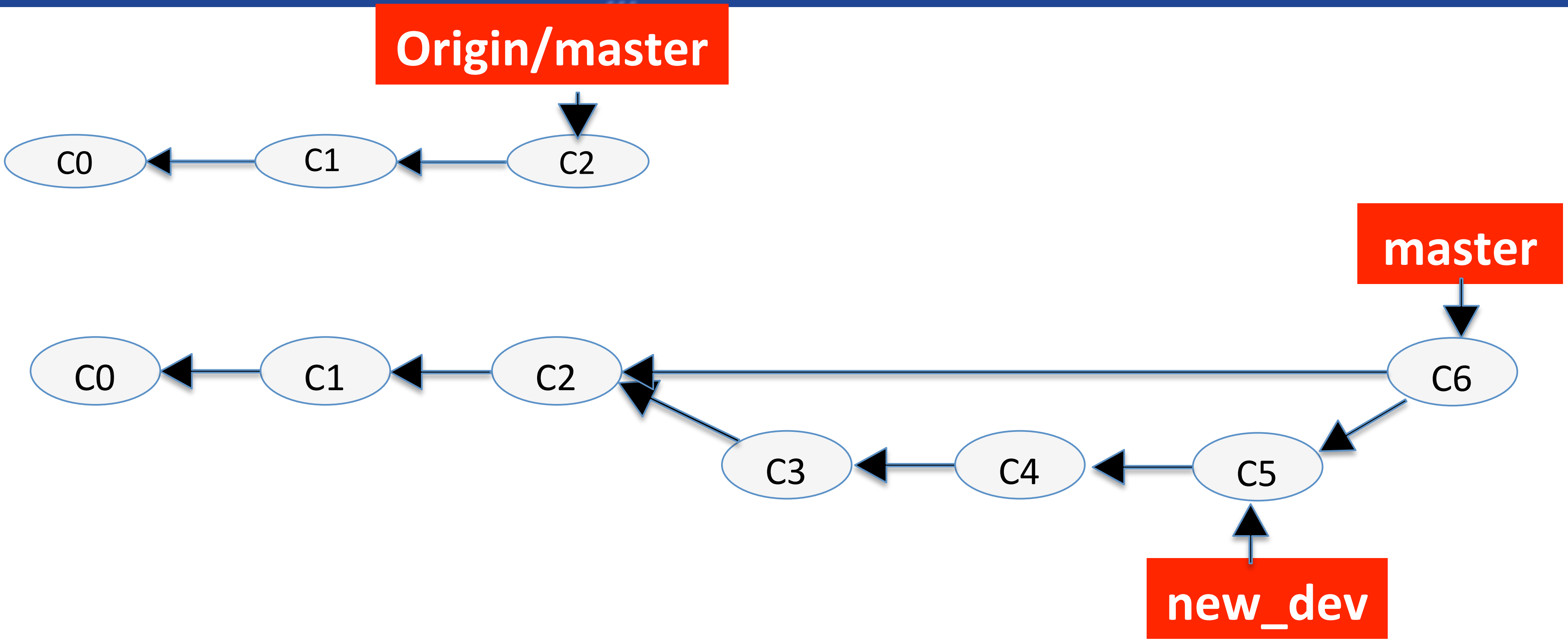
Push the new development

- The new development should be now pushed to the remote repository (if write permission)
- git push origin master execute the operation
- The remote repository is meant to be the server where the origin/master branch is stored (Github, Gitlab, etc...)

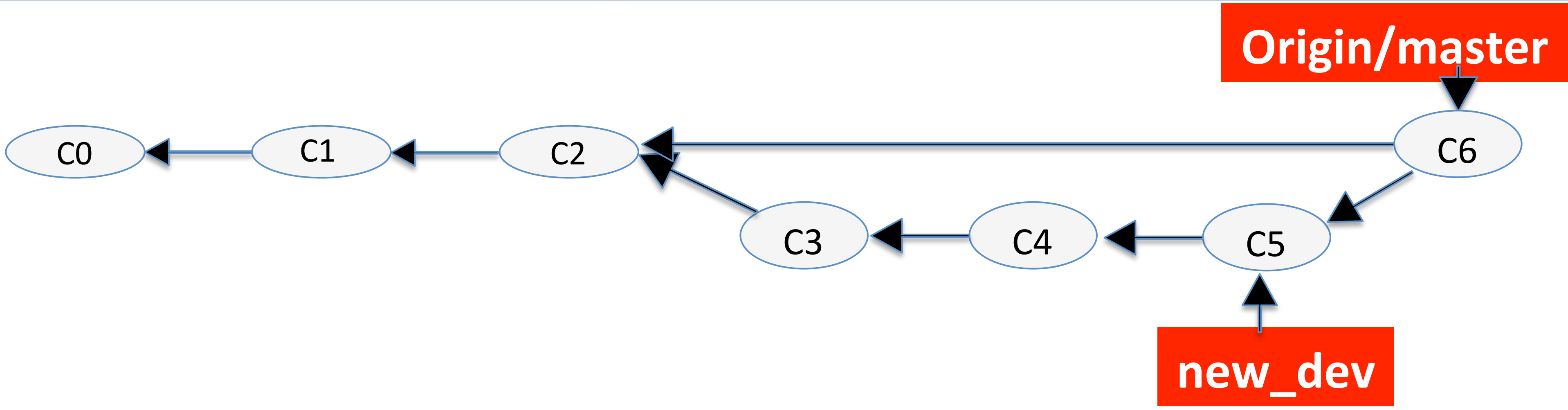
Two cases here...

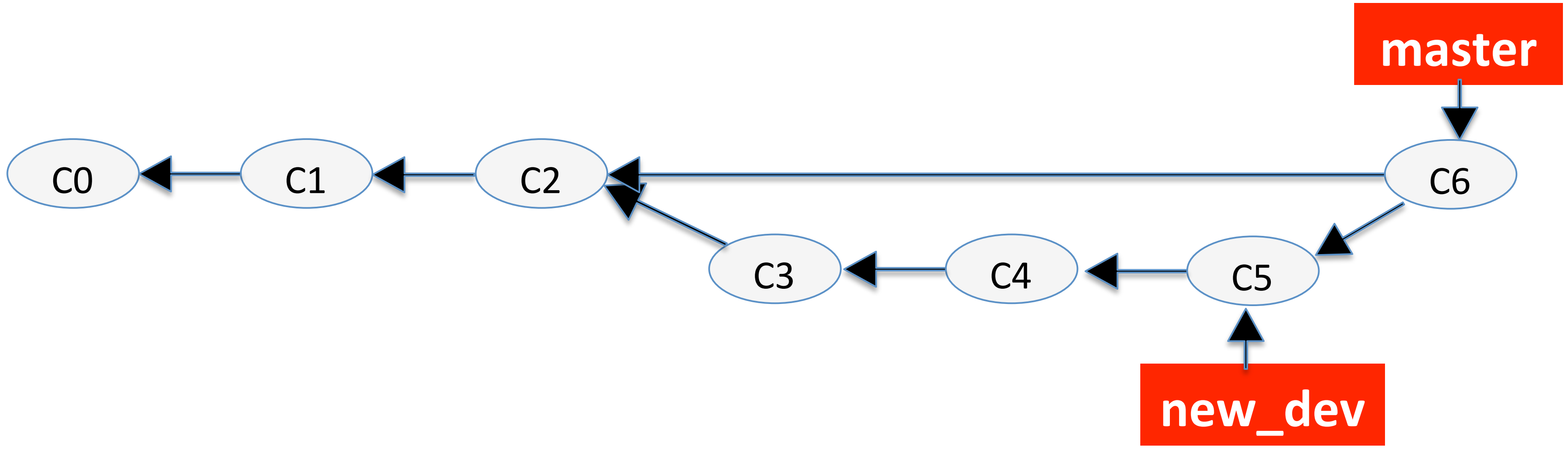
1. The remote repository is still in his initial status (lucky)





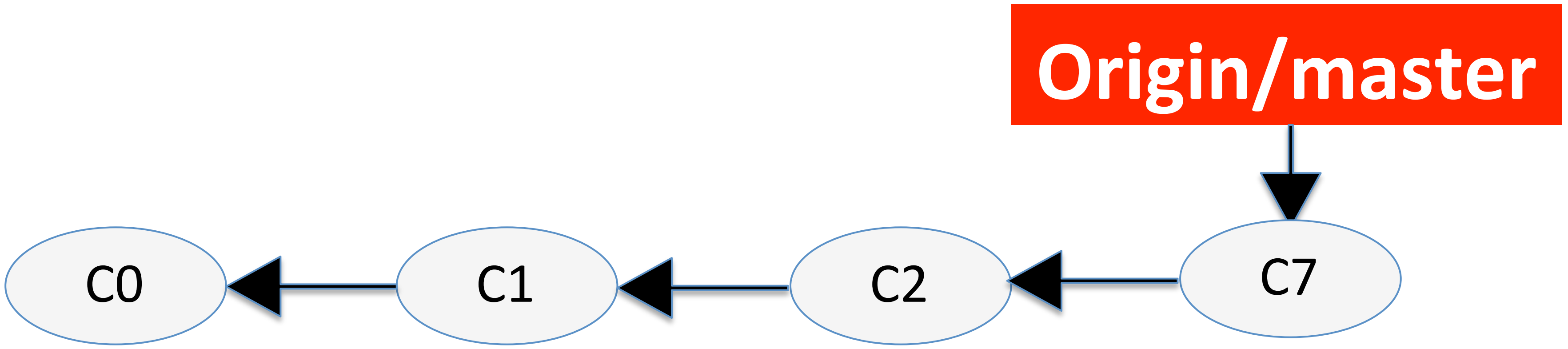
```
$git push origin master
```


Origin/master
C0
C1
C2
C6
C3
C4
C5
new_dev

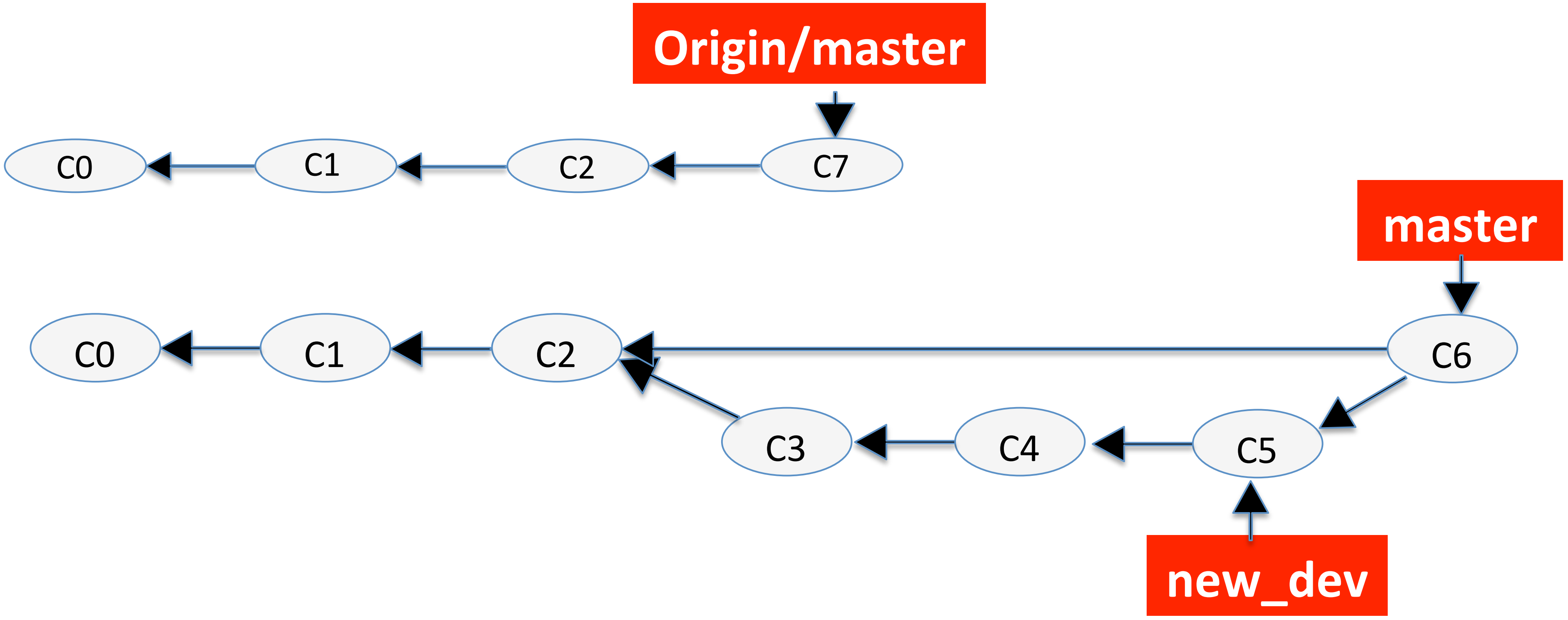

master
C0
C1
C2
C6
C3
C4
C5
new_dev

Two cases here...

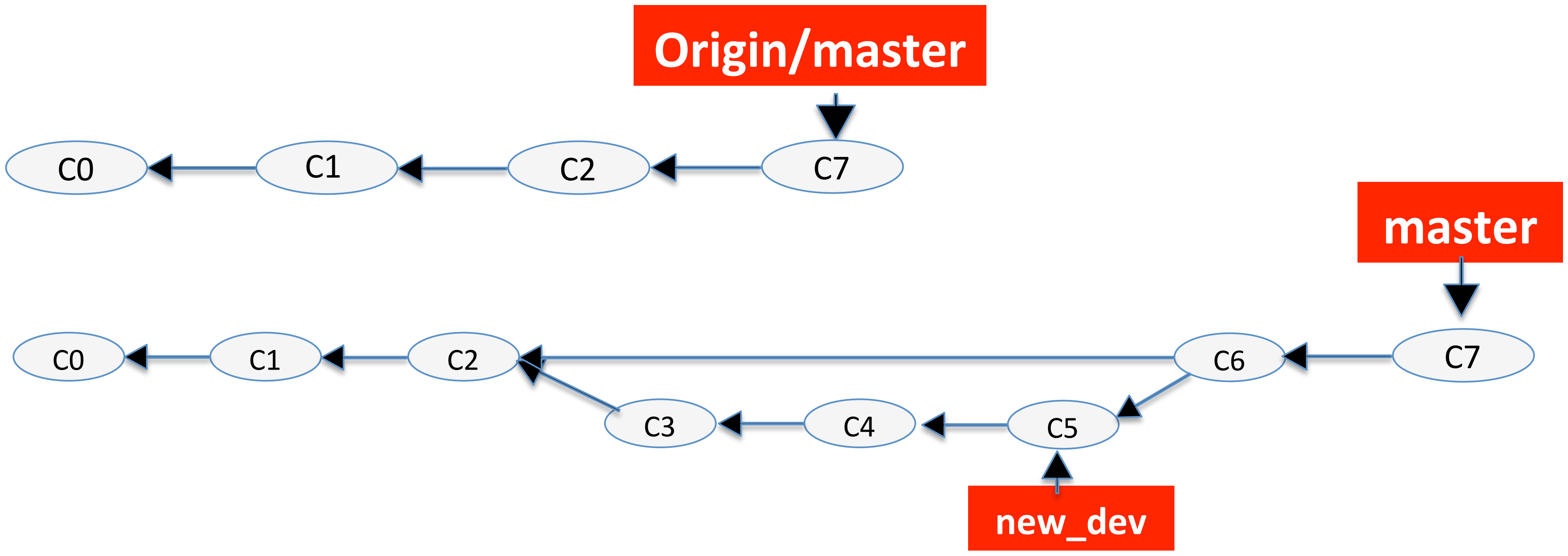
2. The remote repository is changed (likely when collaborating with others). This is what you will be frequently experimenting on next week



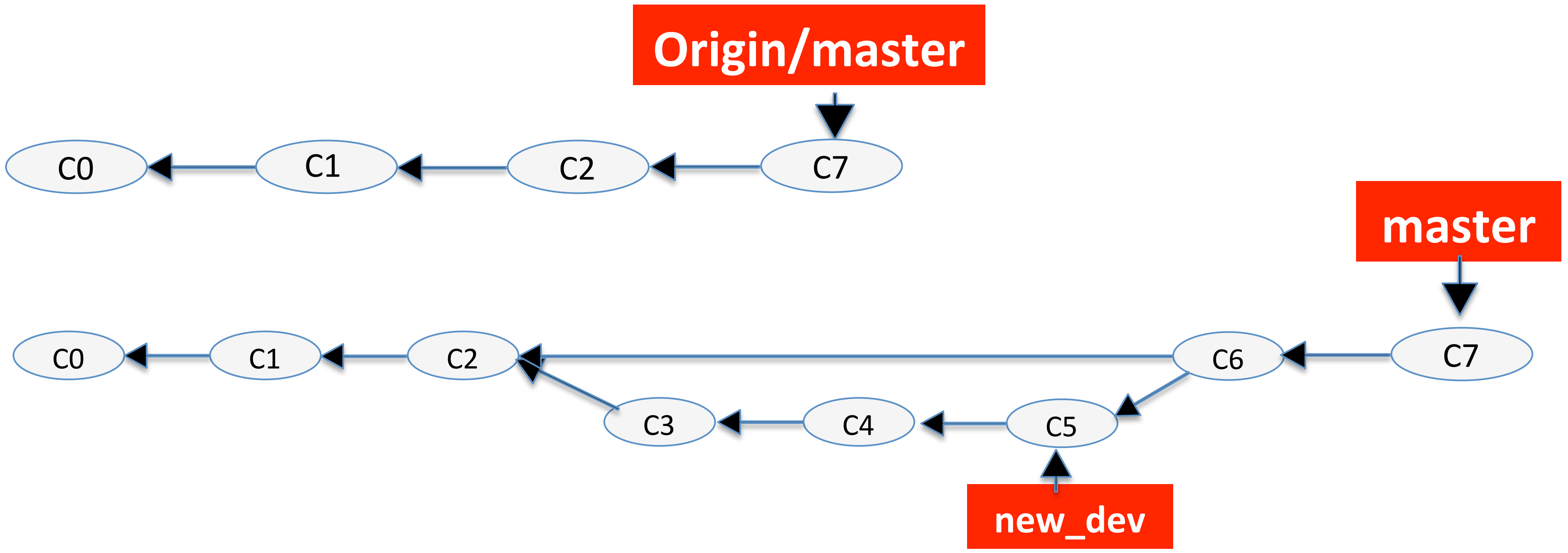
$$\begin{tikzcd}
 & & & \text{Origin/master} \arrow[d] \\
C0 & C1 \arrow[l] & C2 \arrow[l] & C7 \arrow[l]
\end{tikzcd}$$



```
$git pull
```



```
$git pull
```



```
$git push
```

Pull the new development

- The new remote development should be pulled to your local repository before you can safely push on the remote repository
- git pull executes the operation
- If conflicts are solved your local repository is now aligned with the remote repository you share with other users. Is now safe to push on it!