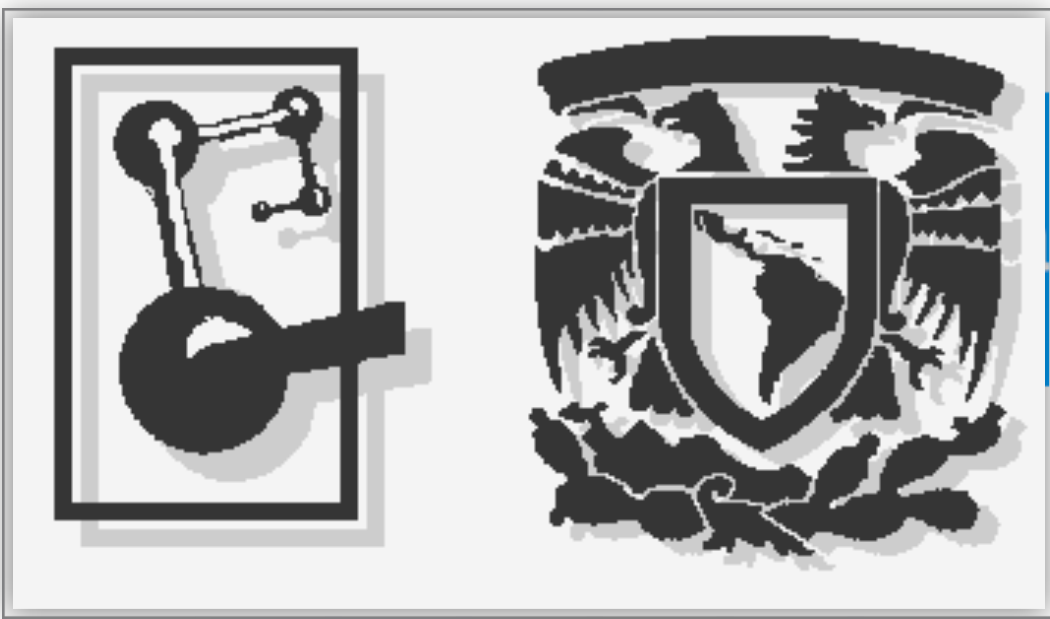




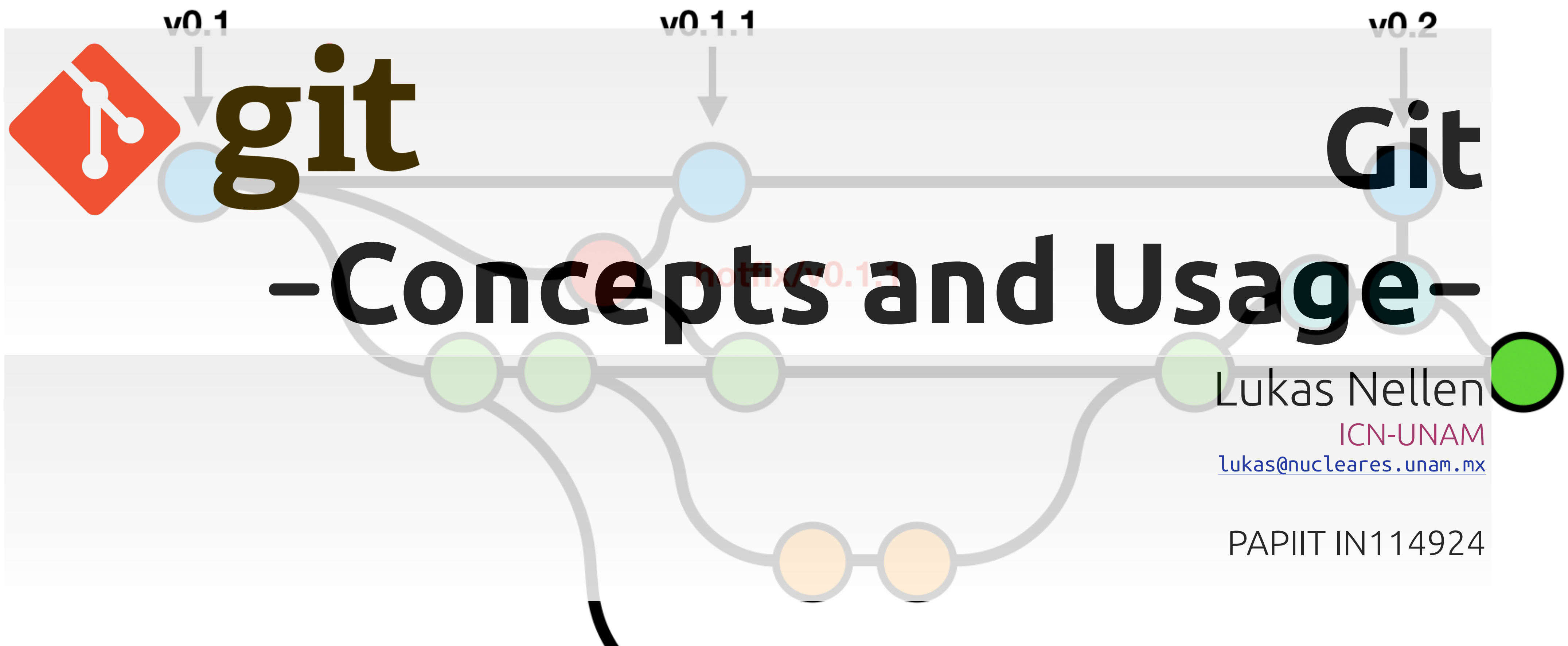
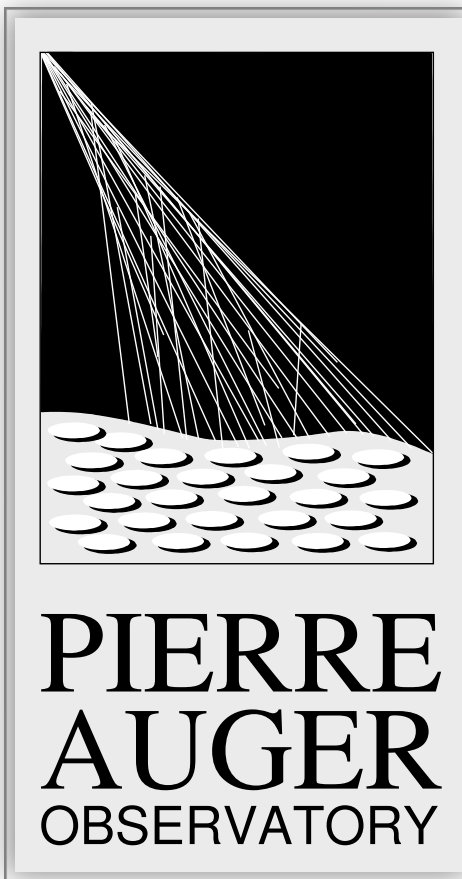
master

develop

feature

release

hotfix

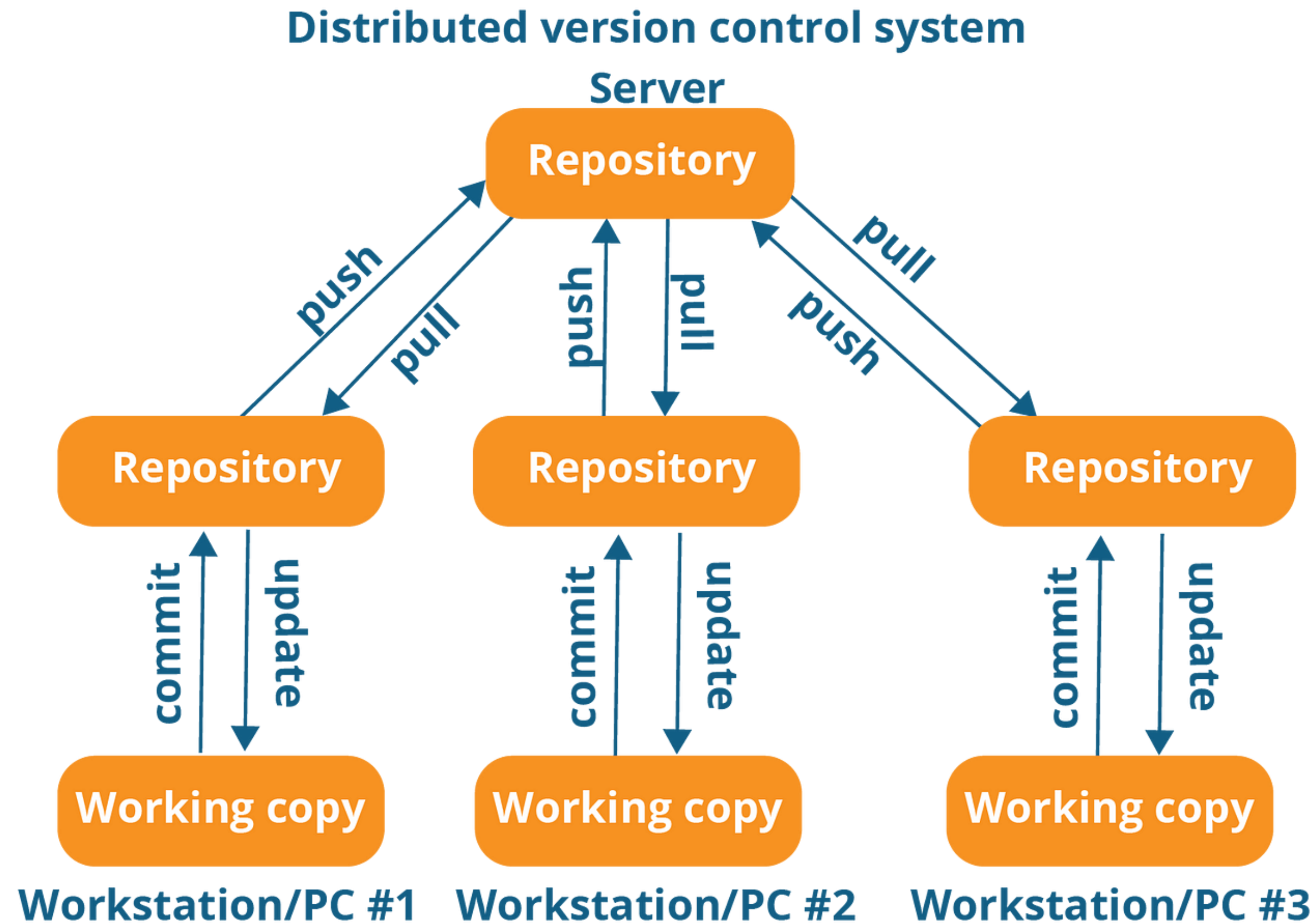


Lukas Nellen
ICN-UNAM
lukas@nucleares.unam.mx

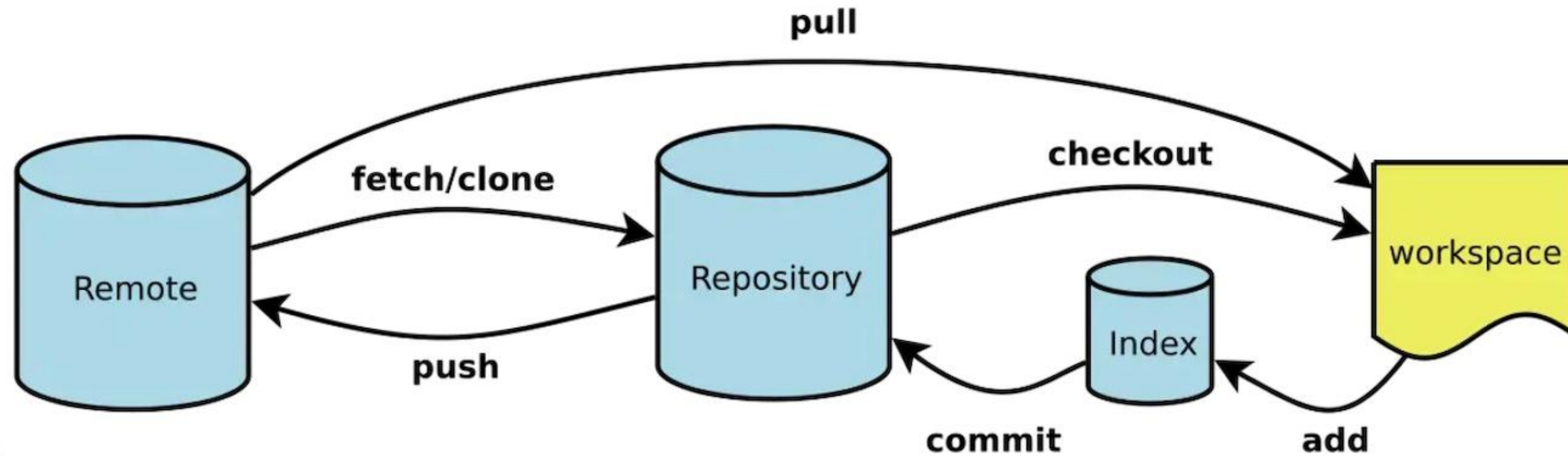
PAPIIT IN114924

Git: A Distributed Version Control System (DVCS)

- Keep track of changes of files in a project
 - Commits with comments
 - Tags to identify versions
- Multiple copies of the Repository possible
 - Often: central, shared repository
 - github, gitlab, forgeo, ...
 - Clone repository to create additional copy



Connecting repositories



- Remote Repository
- Local Repository
- Workspace (i.e., the files you are working on)
- Using pull implies a fetch to the local repository
- Files have to be staged to the index before committing them to the local repository

A first go at git

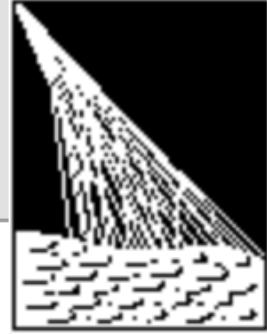
- The repository at <https://gitlab.iap.kit.edu/auger-observatory/offline/git-training-wall> is meant to try a basic git workflow
- Clone this repository
- Create a branch
 - Optional: create an issue first, then create a branch to resolve that issue
- Edit to write your graffiti on [`the-wall.md`](#)
- Commit your changes to your branch
- Edit and commit more to fix (if needed or desired)
- Finally push your branch to the central repository
- Create a merge request
- (Exceptionally) Merge your merge-request
 - Normally, merge requests should be reviewed and merged by the reviewer

Cleanup

- Identify and remove branches that are gone
 - For example after merging

```
git checkout main
git pull --prune
From gitlab.iap.kit.edu:auger-observatory/offline/git-training-wall
- [deleted]          (none)      -> origin/ln-setup-wall
- [deleted]          (none)      -> origin/ln-write-wall
...
git branch -v | cat
  ln-setup-wall 6a433f9 [gone] First go at README.md
  ln-write-wall ab396e6 [gone] Create the wall and scribble
* main         4ac2504 [behind 6] Initial commit
git branch -d ln-write-wall
Deleted branch ln-write-wall (was ab396e6).
git branch -d ln-setup-wall
Deleted branch ln-setup-wall (was 6a433f9).
```

Wiki information



PIERRE AUGER
OBSERVATORY

GitUsage

[CppConventions](#) » [DownloadAndInstall](#) » [OfflineSoftware](#) » [Foundations...ralMeetings](#) » [GitUsage](#)

Git repository and setup for the Pierre Auger Collaboration

Some supplemental material:

- See <https://www.auger.unam.mx/AugerWiki/GitMigration> for plans and progress of migrating from svn to git.
- See the [google docs](#) for images.
- [1. Tutorial: GitLab features, Git\(lab\) config, git workflow](#)
- [2. Tutorial: basic git commands](#)
- <https://www.auger.unam.mx/AugerWiki/ADStatGitLab>
- A list of all maintainers for the different parts/components of Offline can be found here: [Maintainers](#)

1. How to download (clone) the Offline repository from GitLab

The gitlab.iap.kit.edu enforces a two-factor authentication. Hence simply cloning the repository with `git clone https://gitlab.iap.kit.edu/auger-observatory/offline/offline.git` and a username/password combination is not possible anymore. You can use a "Personal Access Token" instead of a password. To generate the token visit https://gitlab.iap.kit.edu/-/profile/personal_access_tokens (you need to be signed in) and create a token with at least "read_repository" rights.

However, we recommend you to use ssh keys for authentication via the command line. Once you have created a key-pair, stored the public key via the web interface in your gitlab account and setup your git/ssh config to use this key pair you can simply clone the repository with:

```
git clone git@gitlab.iap.kit.edu:auger-observatory/offline/offline.git
```

- To change the remote repository of your existing repository (i.e., due to the migration from gitlab.com to gitlab.iap.kit.edu) see Sec. 3.2.

2. How to use git(lab)

🕒 Tutorial videos not updated (Refer to gitlab.com)

Configure

● Global configuration in `~/.gitconfig`

```
[user]
    name = Lukas Nellen
    email = lukas@nucleares.unam.mx

[core]
    excludesfile = /Users/lukas/.gitignore
    pager = less -+-i -j3 -e -w -R
    editor = emacs

[color]
    diff = auto
    status = auto
    branch = auto

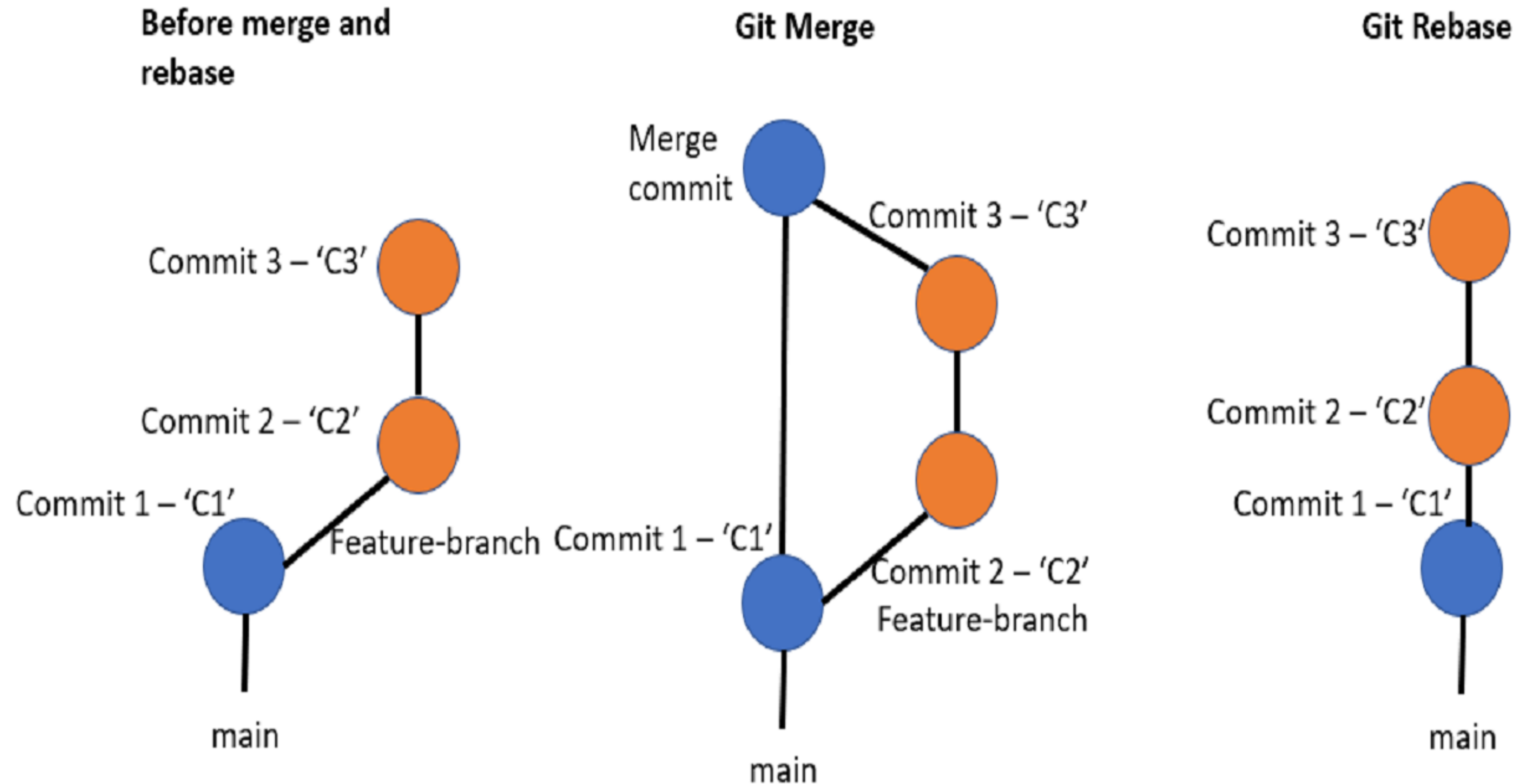
[alias]
    co = checkout
    ci = commit
    cit = citool
    br = branch
    st = status -u

[push]
    default = simple[pull]
    rebase = false

[init]
    defaultBranch = main
```

Example, not
my full config !

Incorporate change: merge vs rebase



- Merging creates a new commit
- Rebase moves the changes to the end of the history
 - Easier to follow (IMHO), but not chronological (but hopefully logical)
 - Use to handle conflicts before merging

Tools for code review

- Gitlab diff view
- VSCode with git plugin
 - Better for larger changes (IMHO)
 - gitlens (gitkraken) useful, but you have to pay for the full version

Advanced topics

- Recovery workflows

- Temporary branch as anchor

- Interactive rebase

- Edit history (cleanup, fixup)

- Worktrees

- Multiple branches checked out in different directories

- Multiple remote repositories

- Extra backup
- Push to test server before commit