GIT - 101

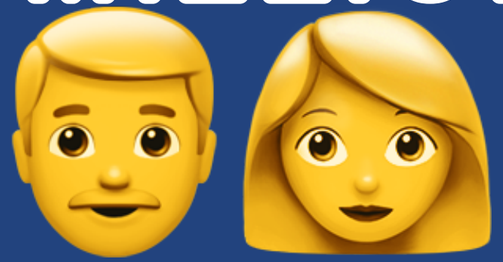
70%

100 MILLIONS 📁 2018

X2IN2

200 MILLIONS 📁 2020

40 MILLIONS



GitHub

ALEXIS LUTUN

FULL STACK DEVELOPPER

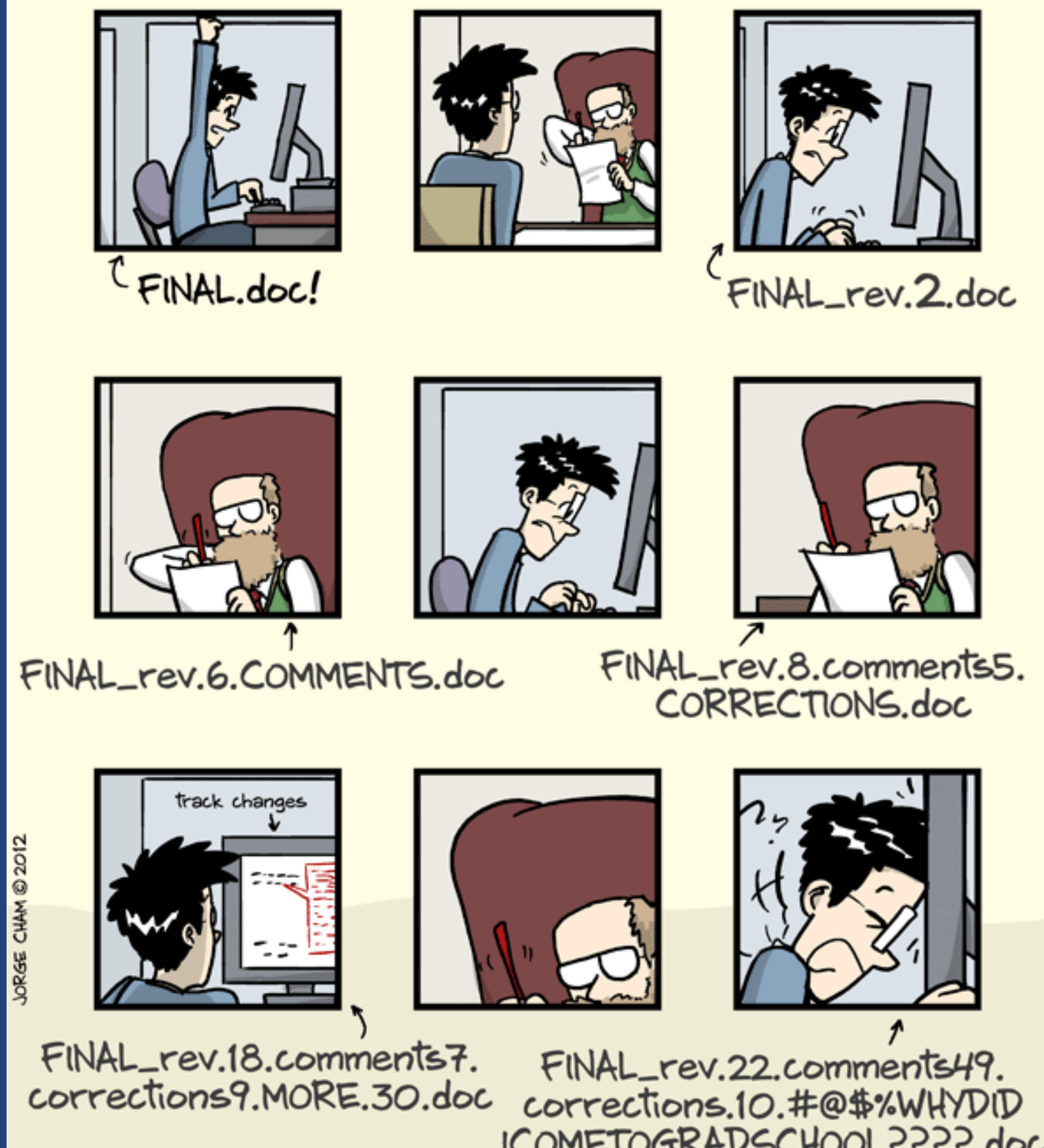
CLOUD ARCHITECT (AWS/GCP)

LOVER OF GITLAB/HUB CI

I LOVE CRAFTING SOFTWARE SOLUTIONS IN THE CLOUD TO CONNECT THE WORLD

WHO WHAT WHERE WHEN HOW

WHO WHAT WHERE WHEN HOW



WHO WHAT WHERE WHEN HOW

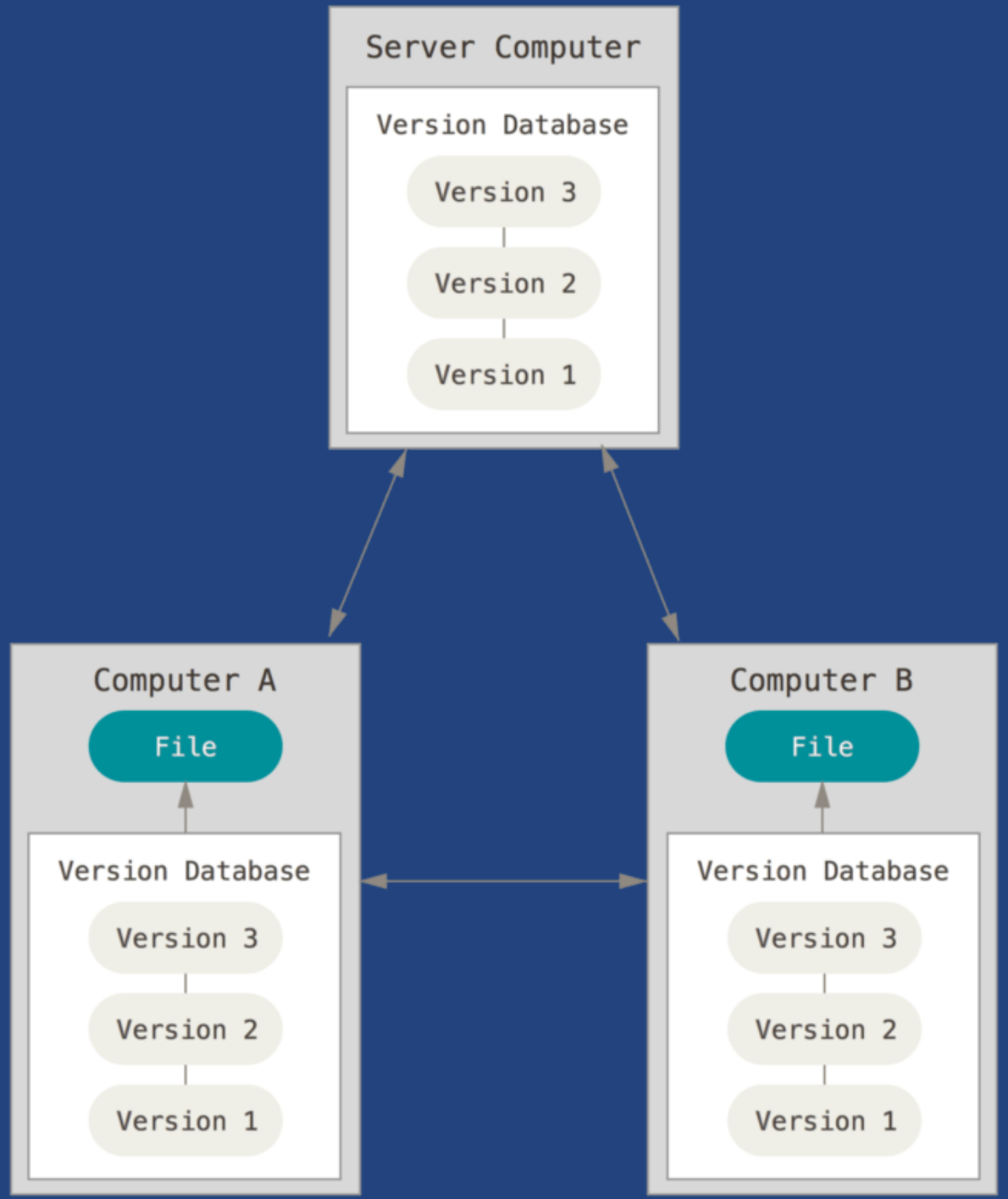
- OPEN SOURCE
- VERSION CONTROL SYSTEM
- DISTRIBUTED
- DESIGNED FOR SPEED AND EFFICIENCY

WHAT : VERSION CONTROL SYSTEM

A VERSION CONTROL SYSTEM TRACKS THE HISTORY OF CHANGES ON SHARED PROJECTS.

CONTRIBUTORS CAN REVIEW PROJECT HISTORY TO FIND OUT:

- WHICH CHANGES WERE MADE?
- WHO MADE THE CHANGES?
- WHEN WERE THE CHANGES MADE?
- WHY WERE CHANGES NEEDED?



WHAT : DISTRIBUTED

- DISTRIBUTED VERSION CONTROL SYSTEM DON'T NEED A CONSTANT CONNECTION TO A CENTRAL REPOSITORY.

- INSTEAD OF COPIES OF REMOTE REPOSITORY THERE ARE LOCAL REPOSITORIES.

IT'S YOUR RESPONSIBILITY TO KEEP SYNC YOUR LOCAL REPOSITORIES AND YOUR REMOTES.

WHO WHAT WHERE WHEN HOW

BE CAREFUL NOT TO CONFUSE GIT WITH GIT HOSTING REPOSITORIES :

GITHUB. GITLAB. BITBUCKET

BE CAREFUL NOT TO CONFUSE GIT WITH GIT GUI CLIENTS :

SOURCETREE. GITHUB DESKTOP. TORTOISEGIT ...

WHEN: YESTERDAY. TODAY. TOMORROW

NO MATTER THE LANGUAGE YOU USE TO CODE.

YOU WILL ALWAYS USE GIT.

HOW

GIT **HELP**

GIT **RESTORE**

GIT **RM**

GIT **MV**

GIT **SWITCH**

GIT **STASH**

GIT **POP**

GIT **SUBMODULE**

GIT **SHOW**

GIT **INIT**

GIT **CLONE**

GIT **CONFIG**

GIT **ADD**

GIT **STATUS**

GIT **LOG**

GIT **MERGE**

GIT **FETCH**

GIT **PUSH**

GIT **DIFF**

GIT **REVERT**

GIT **RESET**

GIT **COMMIT**

GIT **REBASE**

GIT **CHECKOUT**

GIT **REMOTE**

GIT **PULL**

GIT **ARCHIVE**

GIT **DESCRIBE**

GIT **APPLY**

GIT **CHERRY-PICK**

GIT **BLAME**

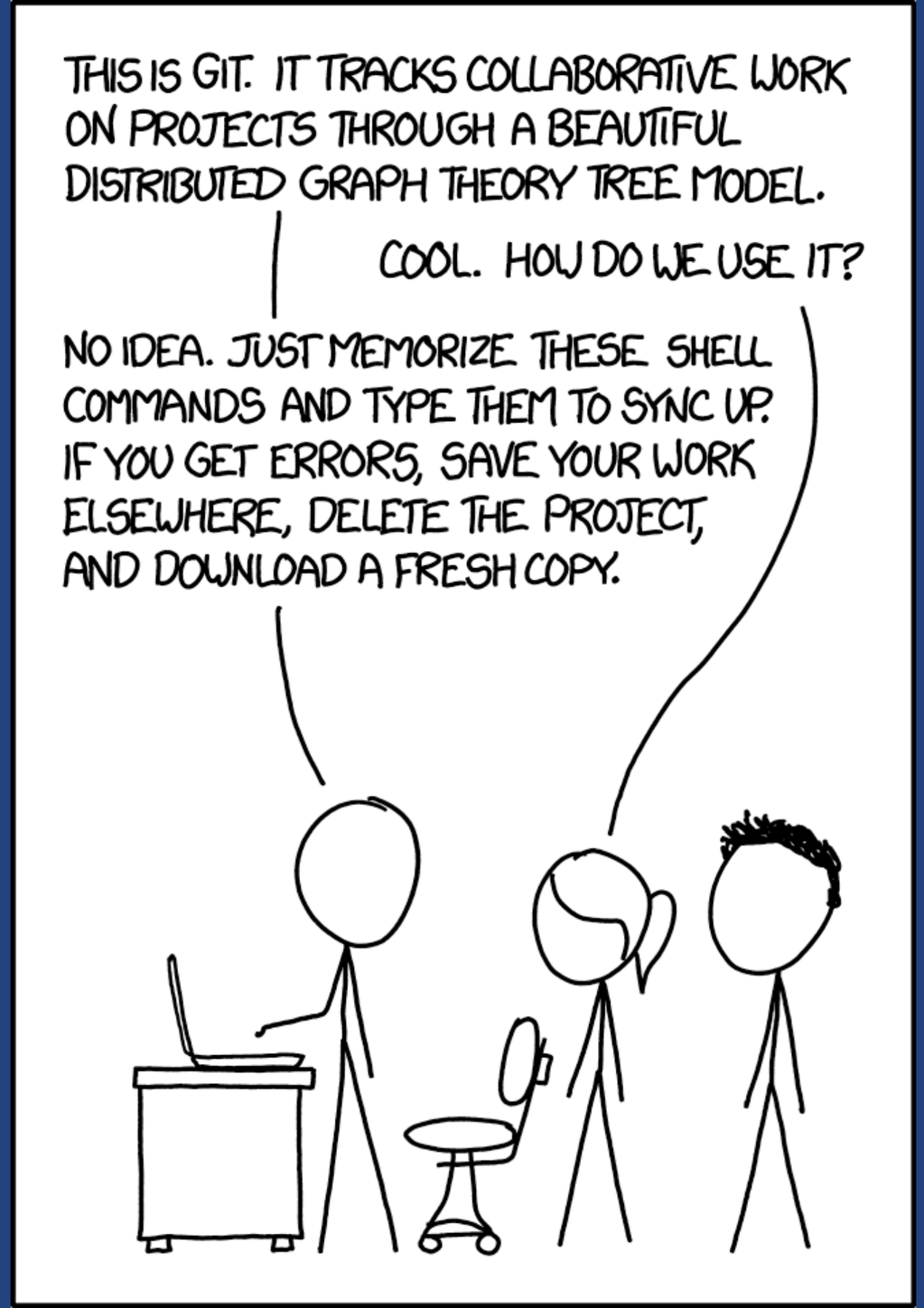
GIT **REFLOG**

GIT **BUNDLE**

GIT **REVISIONS**

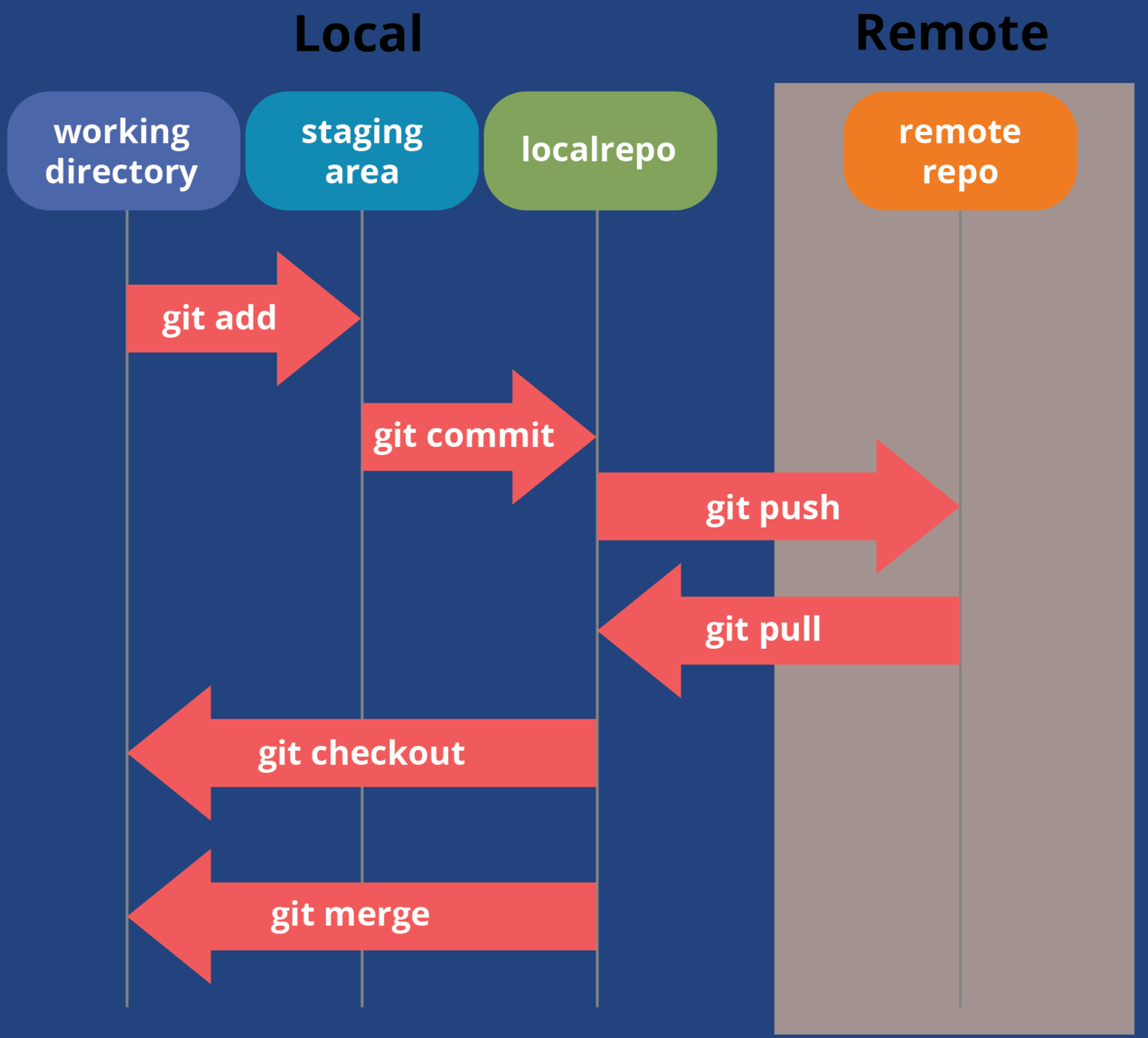
GIT **INSTAWEB**

GIT **COUNT-OBJECTS**



HOW : FOR REAL

- ADD
- COMMIT
- PUSH
- PULL
- BRANCH
- CHECKOUT
- MERGE
- CLONE


Local
Remote
working directory
staging area
localrepo
remote repo
git add
git commit
git push
git pull
git checkout
git merge

FIRST TIME CONFIGURATION

```
$ git config --list

$ git config --global user.name "John Doe"
$ git config --global user.email johndoe@example.com
```

CREATE REPOSITORY

FROM LOCAL

```
$ cd path/to/directory
$ git init [project-name]
```

FROM REMOTE

```
$ git clone [url]
```

AFTER EDITING FILE

LIST NEW OR MODIFIED FILES TO BE COMMITTED

```
$ git status
```

SHOW DIFFERENCES

```
$ git diff [optional filename/repository]
```

AFTER EDITING FILE

STAGE/UNSTAGE FILE

```
$ git add [filename]
$ git reset [filename]
```

RECORD IN VERSION HISTORY

```
$ git commit -m "[message]"
```

WORKING WITH REMOTE

DOWNLOAD HISTORY FROM REMOTE AND COMBINE WITH LOCAL

```
$ git fetch [remote]
$ git merge [branch]
```

DOWNLOAD AND MERGE

```
$ git pull
```

UPLOAD WORK

```
$ git push [remote] [branch]
```

HISTORY

```
$ git log
```

GIT A DOG !

```
$ git log --all --decorate --oneline --graph
```

SEND ME YOUR GITHUB ACCOUNT URL IN THE CHAT

HTTPS://GITHUB.COM/ALUTUN

LAB 1

CREATING A REPOSITORY AND MAKING THE FIRST COMMIT

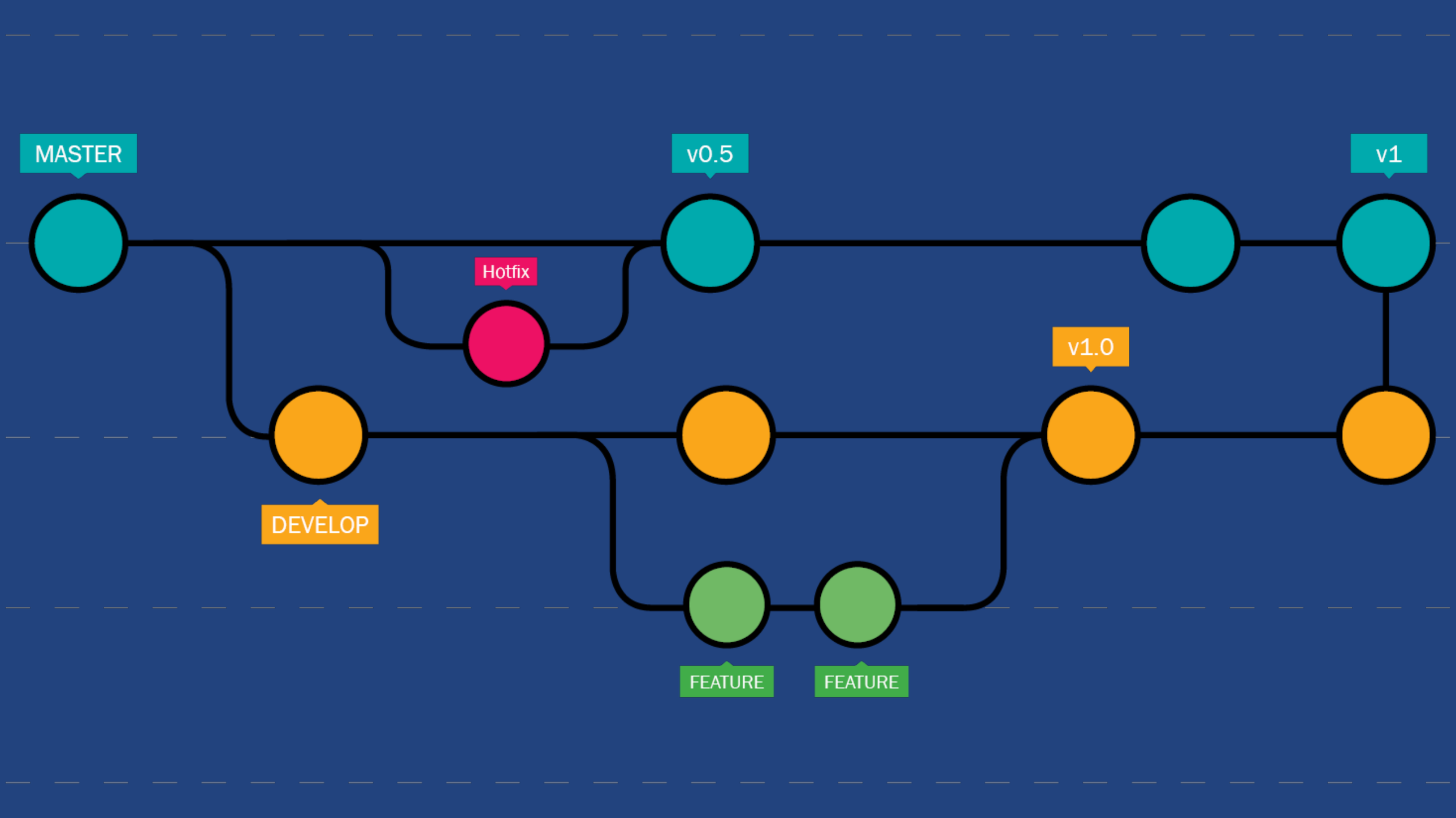
1. CREATE A REPOSITORY GIT-TUTORIAL ON THE WEBSITE. SELEECT INITIALIZE THIS REPOSITORY WITH A README.
2. GET A COPY OF THE REPOSITORY ON YOUR COMPUTER. TO DO THIS: OPEN UP THE TERMINAL OR GIT SHELL.
3. CREATE A FILE CALLED LANGUAGES.TXT. ADD TEXT INSIDE.
4. CHECK THE STATUS OF YOUR GIT REPOSITORY.
5. TRACK YOUR FILE LANGUAGES.TXT
6. SEND YOUR FILE TO THE REMOTE LANGUAGES.TXT
7. CHECK ON GITHUB.COM

LAB 2

UNDERSTANDING REMOTE AND LOCAL WORK

1. ACCESS TO YOUR GIT-TUTORIAL ON THE GITHUB.COM
2. CHANGE THE CONTENT YOUR LANGUAGES.TXT FROM GITHUB.COM
3. ON YOUR COMPUTER DOWNLOAD THE MODIFICATION FROM THE REMOTE REPOSITORY
4. CHANGE THE FIRST LINE OF YOUR LANGUAGES.TXT ON YOUR COMPUTER AND ON GITHUB.COM WITH DIFFERENT INFORMATIONS.
5. SEND YOUR LOCAL MODIFICATION TO THE REMOTE REPOSITORY

THE STORY OF A TREE


MASTER
v0.5
v1
Hotfix
v1.0
DEVELOP
FEATURE
FEATURE

GIT WORKFLOW

CREATE AND SWITCH TO BRANCH

```
$ git branch [branch-name]
$ git checkout [branch-name]
```

COMBINE BRANCH HISTORY INTO CURRENT BRANCH AND DELETE

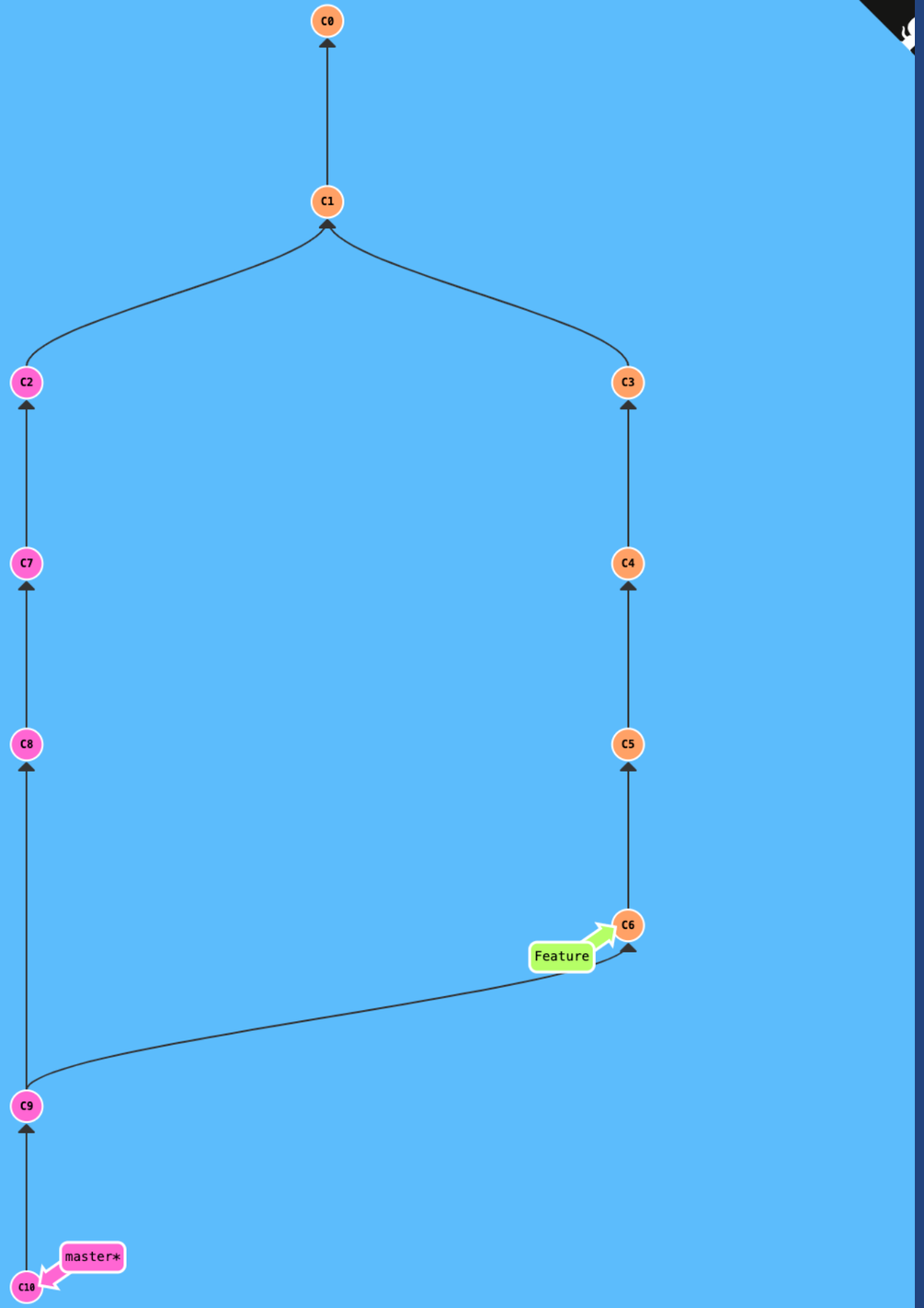
```
$ git merge [branch]
$ git branch -d [branch]
```

LAB 3

CREATING BRANCHES AND MERGING

LET'S PLAY IN THE SAND

HTTPS://LEARNGITBRANCHING.JS.ORG/?NODEMO

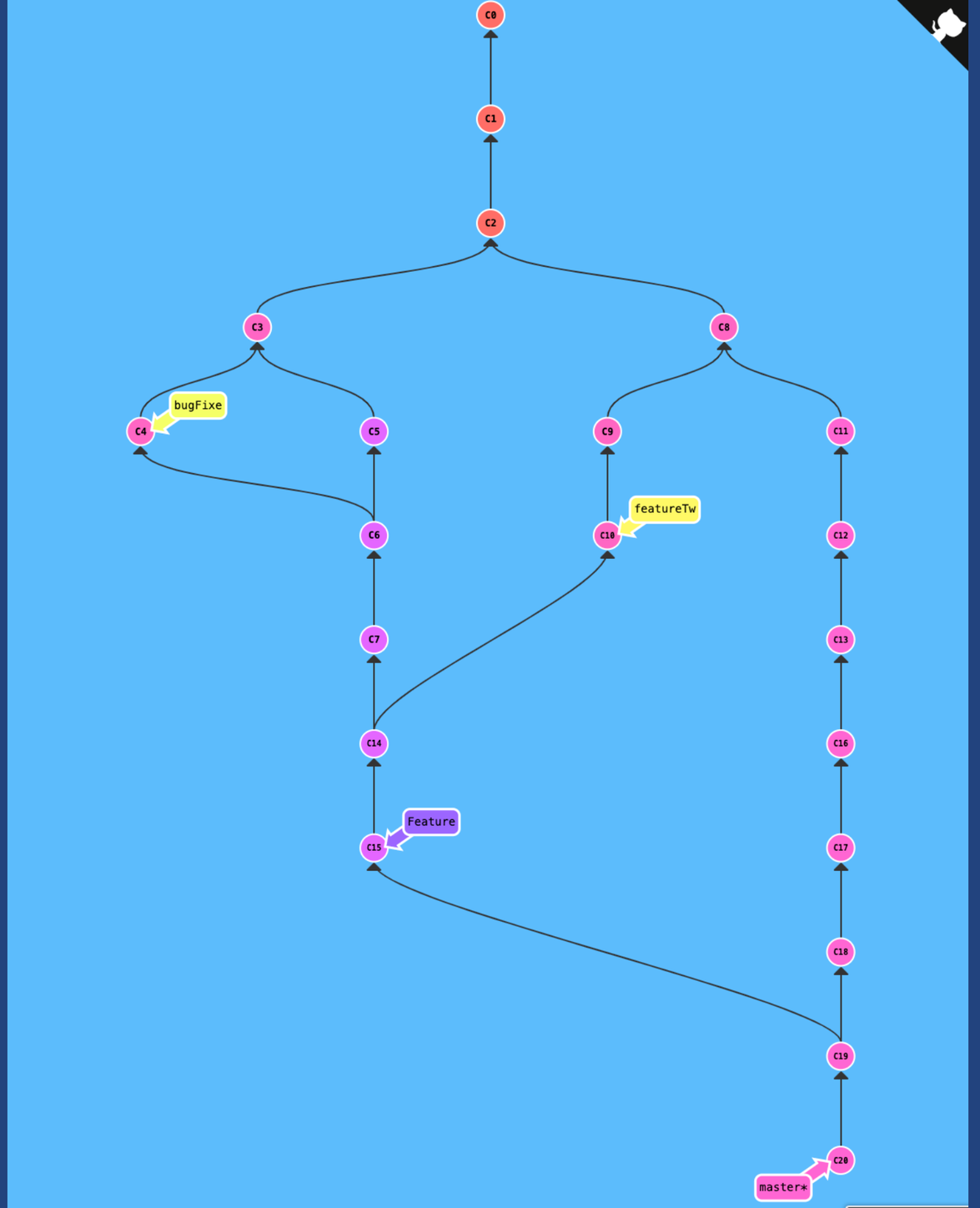


GOAL : CREATE A BRANCH CALL FEATURE AND MERGE IT IN TO MASTER

COMMANDS USED "COMMIT. BRANCH. MERGE. CHECKOUT"

GOAL : CREATE A TREE AS SHOW ON THE PICTURES

COMMANDS USED "COMMIT. BRANCH. MERGE. CHECKOUT"



LAB 4 : LET'S WRITE A BOOK TOGETHER !

HTTPS://
GITHUB.COM/
ALUTUN/
ITSABOOK

GOING FURTHER WHAT MATTERS REALLY :

1. ISSUES MANAGEMENT
2. CI
3. INTEGRATIONS
4. HOSTING
5. WIKI
6. TIME TRACKING
7. CODE REVIEW
8. ...

GIT GUI

USE IT WHEN

YOU DON'T NEED IT ANYMORE

RESSOURCES

CHECK THE README.MD

THANKS!

FEEL FREE TO HIT ME ON SLACK