ÉTS
Le génie pour l'industrie

NAOVA

THE BASICS

...

By Thierry Pouplier

Original presentation by:
GameCraft Training ☺
Radoslav Georgiev (@Rado_G)

Agenda

- **Why use Source Control System ?**
- How to setup Git and Github on Linux ?
- Terminology
- Useful git commands
- Workflows

Why use Source Control Systems ?

What is it ?

- Software that keep track of version of files
- Decentralized
- Invented by Linus Torvalds (linux inventor)

Why use it ?

- When team > 1
- All files are hosted (Github)
- Your future boss and co-worker will thanks me

No Source Control System =

Agenda

- Why use Source Control System ?
- **How to setup Git and Github on Linux ?**
- Terminology
- Useful git commands
- Workflows

How to setup Git and Github on Linux?

- First of all – create a Github Account
- Give me your username
- Join the organization https://github.com/Naova

How to setup Git and Github on Linux?

- Open a terminal (ctrl + alt + t)
- Install Git

```
$ sudo apt-get install git
```

- Name & Email – Github tracks them

```
$ git config –global user.name “Firstname Lastname”
$ git config –global user.email “email@email.com”
```

- gg, wp

Agenda

- Why use Source Control System ?
- How to setup Git and Github on Linux ?
- **Terminology**
- Useful git commands
- Workflows

Some basic Terminology

- **git** = The revision control system
- **github** = Website and cloud server for our code
- **repo** = Repository. Root folder of the project.
- **commit** = A group of saved changes
- **push** = Uploading the code (commits) to the server
- **branch** = A different line of commits that will need to be merge back or die
- **merge** = Adding the commits of two branch together
- **conflict** = https://media.giphy.com/media/cFkiFMDg3iFoI/giphy.gif
- **pull request (PR):** Request to merge your branch into another

Agenda

- Why use Source Control System ?
- How to setup Git and Github on Linux ?
- Terminology
- **Useful git commands**
- Workflows

Lets create a repo !

- Click on the new repository button in Github
- Start a terminal (ctrl-alt-t)
- Create a folder and go in it:

```
$ mkdir git_test
$ cd git_test
```

- Execute the super-complex command :

```
$ git init
```

- Great, now we have repo. Lets create a file, shall we ?

```
$ touch naovaFTW.txt
$ nano naovaFTW.txt
or
$ echo “Super AI secret” > naovaFTW.txt
$ cat naovaFTW.txt
Super AI secret
```

Lets create a repo !

- Okay, lets add it !

```
$ git add naovaFTW.txt
```

- And commit it ☺

```
$ git commit –m ‘This is a commit message’
Some gitorish output
```

- And for the sake of learning, let's edit it again

```
$ echo “Unlimited redbull” >> naovaFTW.txt
$ cat naovaFTW.txt
Super AI secret
Unlimited redbull
```

Lets create a repo !

- And now, lets see :

```
$ git status
```

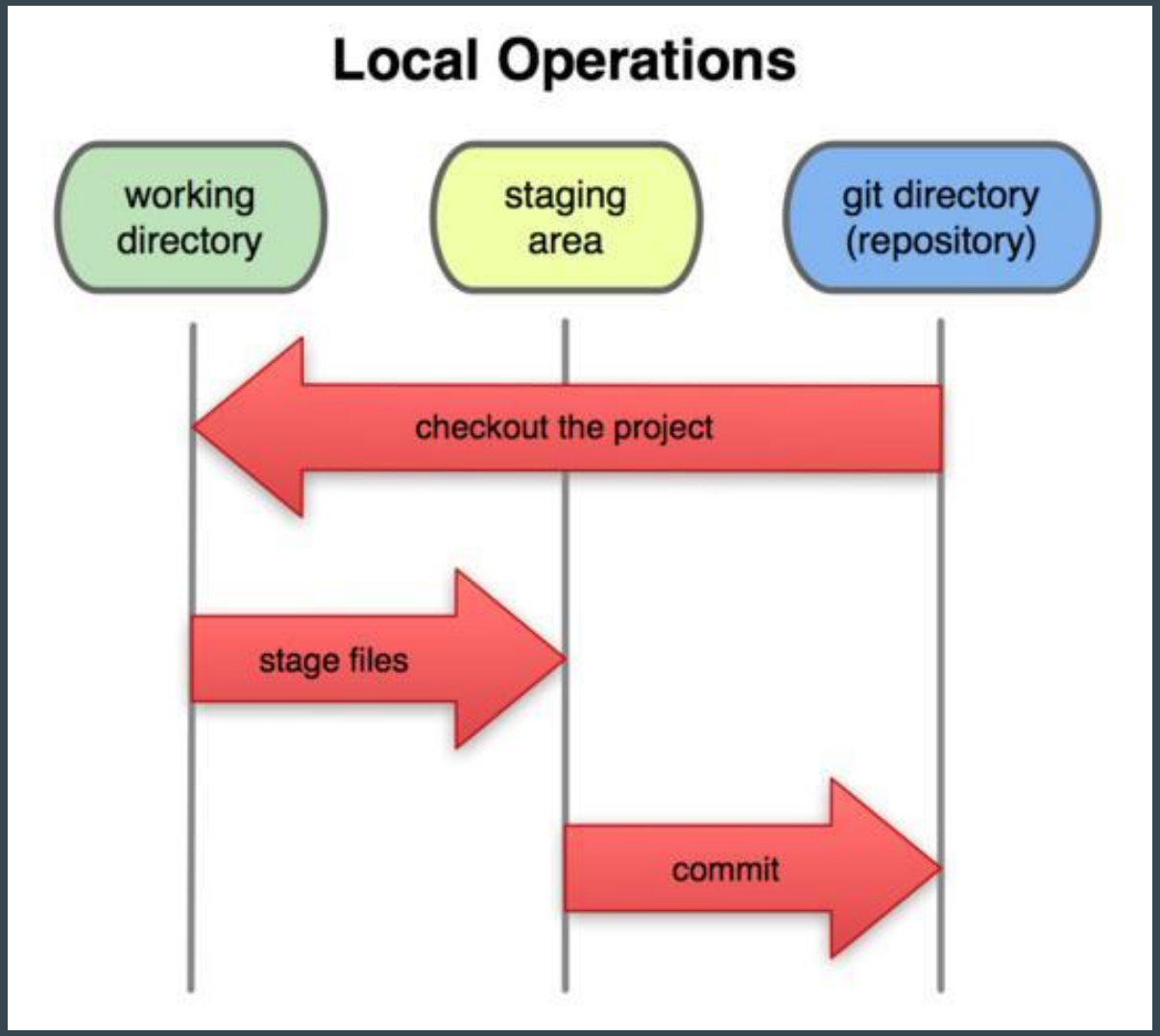
- Outputs :

```
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   naovaFTW.txt
```

- Almost there

```
$ git add naovaFTW.txt
$ git status
```

How it works? Staging area.



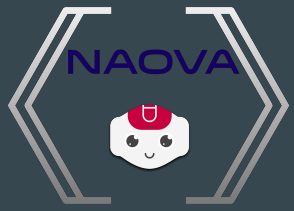
Don’t push your passwords

- Use .gitignore

```
$ touch .gitignore
$ echo “db_config.php” >> .gitignore
$ git add .gitignore
$ git push origin master
Enter passphrase!
```

- Something missing ?

```
$ git commit –m ‘You are not seeing my passwords!’
```



Made a mistake ? No worries

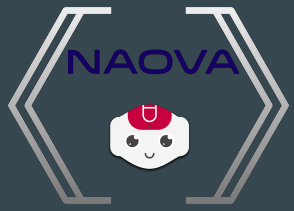
- Unstage something – git reset

```
$ git add index.php
$ git status
Says it's staged. I don't want to ! I changed my mind.
$ git reset HEAD – index.php
$ git status
Now I'm happy ^_^
```

- Revert a commit ? Reset hard! But watchout !

```
$ git reset –hard HEAD~1
OR
$ git reset –hard <commit_id>
```

Useful tool: tig



What about Github ? Remotes ?

- Push to github ?
- Damn. Enter magic!

```
$ git remote add origin git@github.com:UserName/ProjectName.git
```

- **Git commits locally, pushes remotely !!!!!!!**

- Add the remote when the repo is created (git init, remember ? ☺)

```
$ git remote add [name] [url]
```

- Want to see the remotes ?

```
$ git remote -v
```

What about Github ? Push it up, baby!

- Okay, we have committed and added a remote to Github. It's time to push ☺

```
$ git push origin master
Enter passphrase ! ☺
```

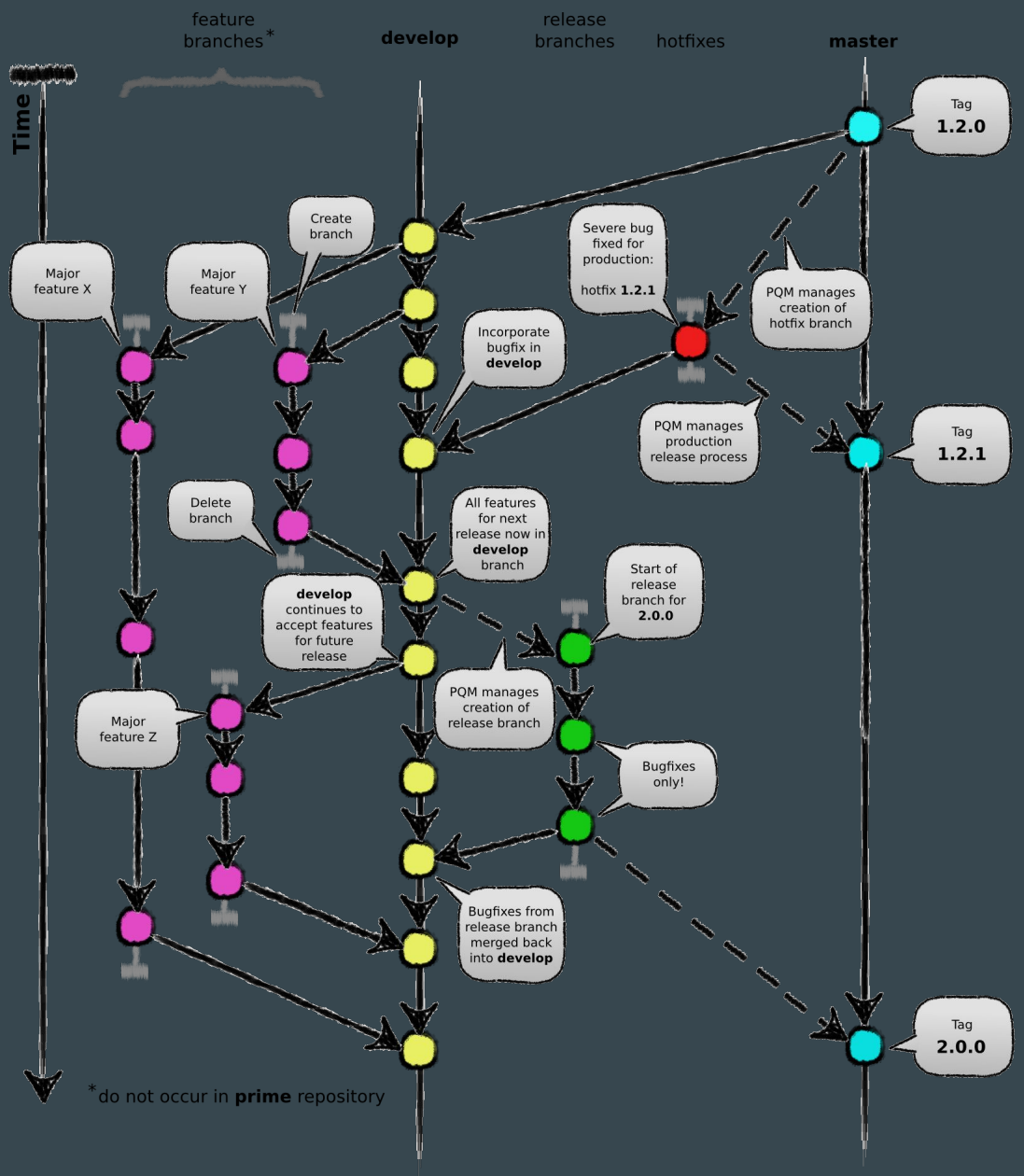
- Open up the repo in Github and enjoy ^_^
- The push command explained :

```
$ git push [remote_name] [branch]
```

- Branches are black magic ~~for later~~ **that you need to learn NOW** ☺
- ~~There's a big chance that the branch you are pushing to will be named "master".~~ **Please god NO...**

Branchs

- Isolate the development of the differents features.
- Isolate the release code from the code in development



Creating a branch

- On the good origin branch? if not, change branch!

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

- Creating a branch

```
$ git checkout -b my_branch
```

- Checking if it worked

```
$ git status
# On branch my_branch
# …
# ..
```

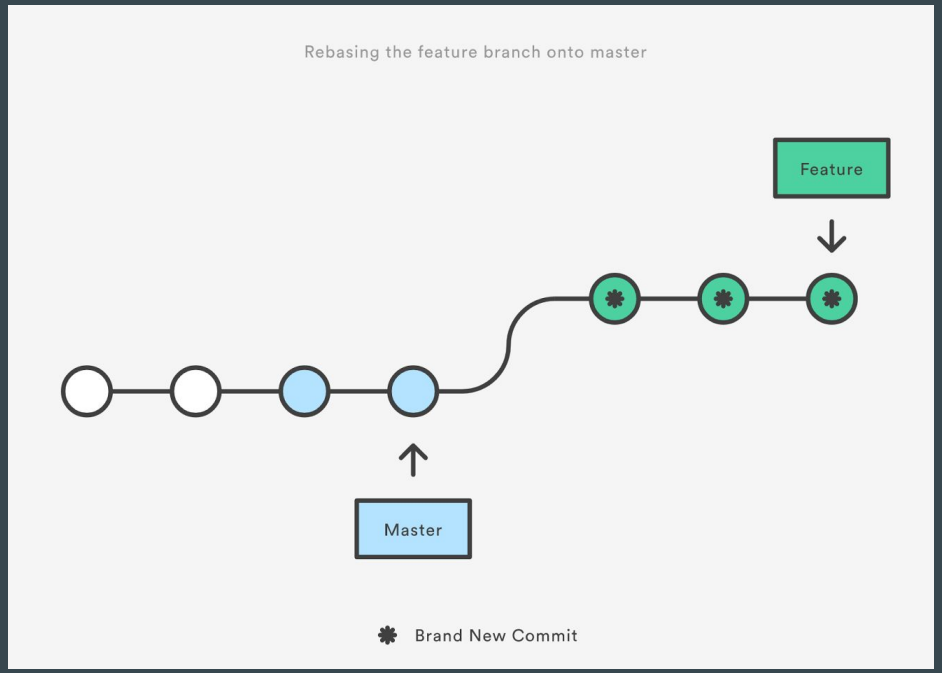
Agenda

- Why use Source Control System ?
- How to setup Git and Github on Linux ?
- Terminology
- Useful git commands
- **Workflows**

Workflows

Start a project

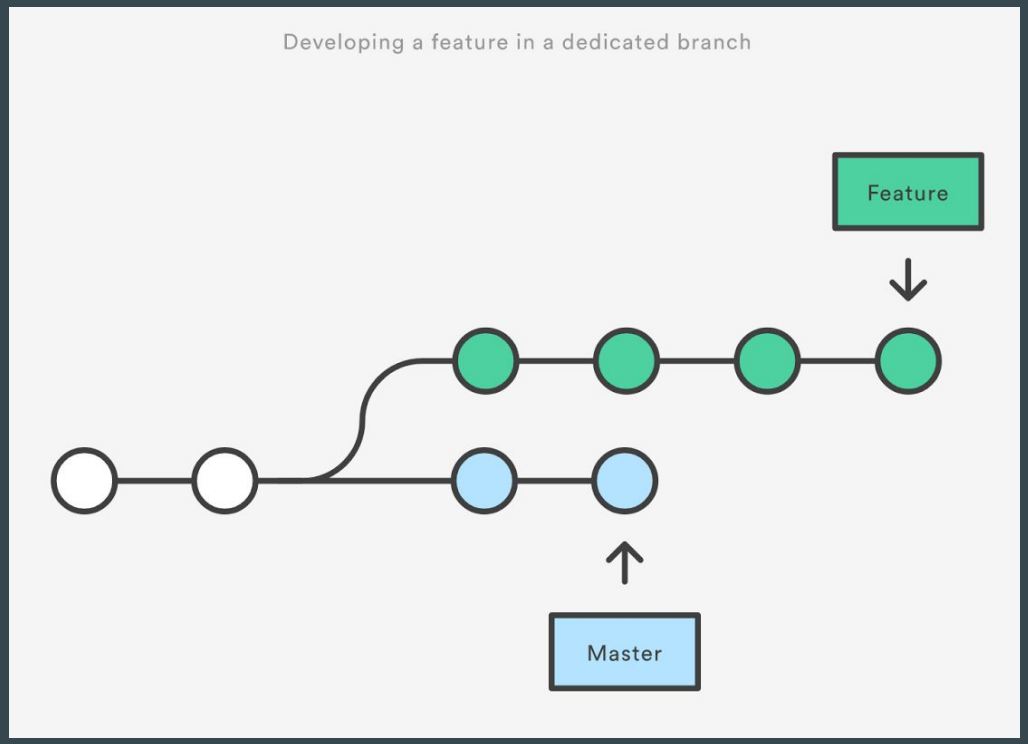
1. **CHECKOUT devel branch**
2. **PULL**
3. Create branch
4. Initial COMMIT
5. PUSH on your branch



Workflows

Work session

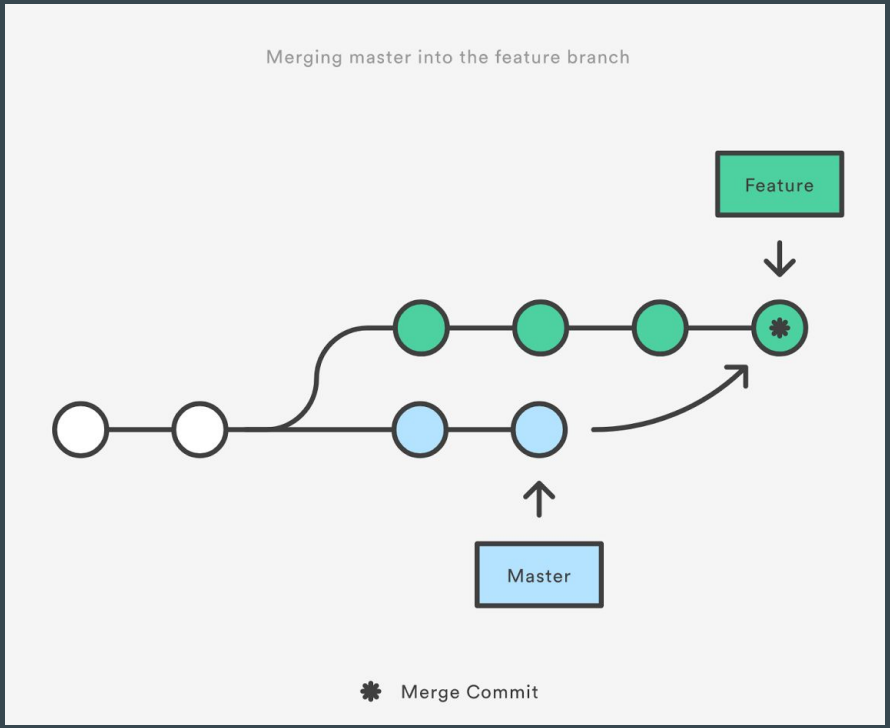
1. **PULL**
2. **FIX CONFLICT**
3. WORK
4. COMMIT
5. WORK
6. COMMIT
7. **PUSH**



Workflows

End a project

1. **PULL devel branch**
2. **FIX CONFLICT**
3. PUSH on your branch
4. Create a PR
5. Ask for reviews
6. MERGE
7. Delete branch
8. **Document**



Let’s code ! ;P