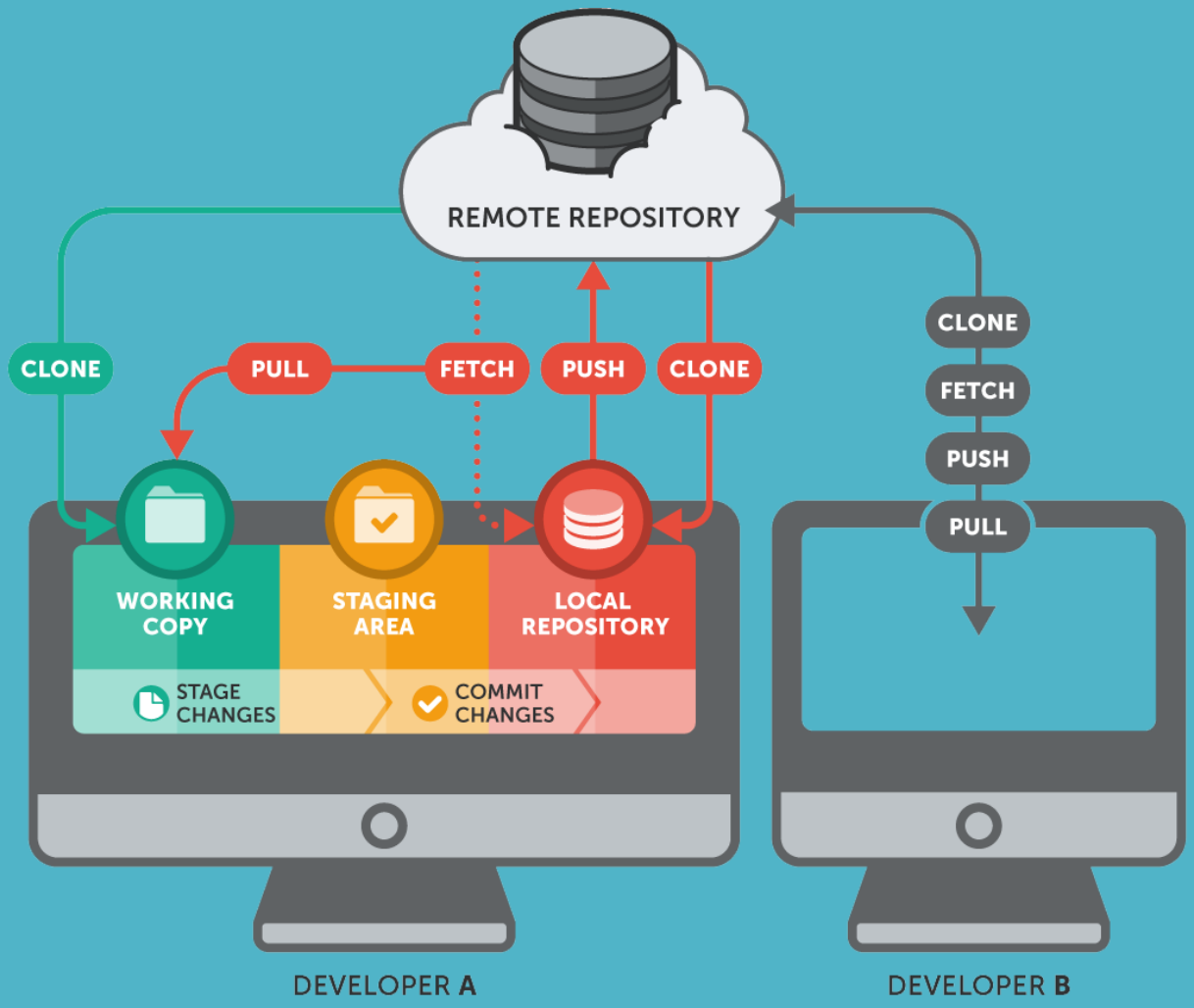
Hack a Game

Local vs Remote



Git 101 (I)

From Scratch:

$ git init

$ git remote add origin {url}

From Github:

$ git clone {url}

Initial Steps:

$ git config user.name “{github-username}”

$ git config user.email “{github-email}”

Git 101 (II)

Pull (get) changes:

$ git pull (git pull [from] [branch], git pull origin master)

Push (send) changes:

$ git add . [-A]

$ git commit [-m “{message}”]

$ git push (git push [to] [branch], git push origin master)

Branches?

Branches!

Create New:

$ git checkout -b {name}

!Starts from the current branch!

Change current:

$ git branch [-a] (view all)

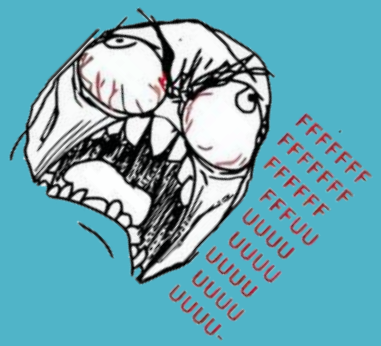
$ git checkout {name}

Merging:

$ git checkout {dest-branch}

$ git merge {source-branch} --no-ff (--no-ff?!)

Conflicts



Now… what?

CONFLICT ({reason}): Merge failed in {file}

```
<<<<<<<<< Your_Hash / HEAD
        // Your code
=========
        // Other branch code
>>>>>>>>> Other branch hash
```

Reasons

- $ content
- $ add/remove

Decide and:

- $ git add {file}
- $ git commit
- $ git push

Do [Not?] Use

Blacklist:

$ git push -f

Specially if before you have done something like

$ git reset --hard {hash}

Whitelist:

$ git stash + git stash list + git stash apply {x}

$ git reset

$ git clean [-xfd]

$ git rebase {branch}

$ git revert {hash}

Conflicts in Unity

Scripts?

> No problem

Prefabs? Scene?

> :/

> Solution?

+ Only 1 person modifies the principal scene

+ Per person test scene

+ Common test scene

Dependencies in Unity

Scripts?

> More modularity = MonoBehaviours

> Less inter-dependencies = Interfaces + Dummy implementations

> TALK!

Prefabs? Scenes? Animations?

The incredible block that does nothing

but works for everything