Version Control Systems and Git

Ing. Bladimir Arroyo

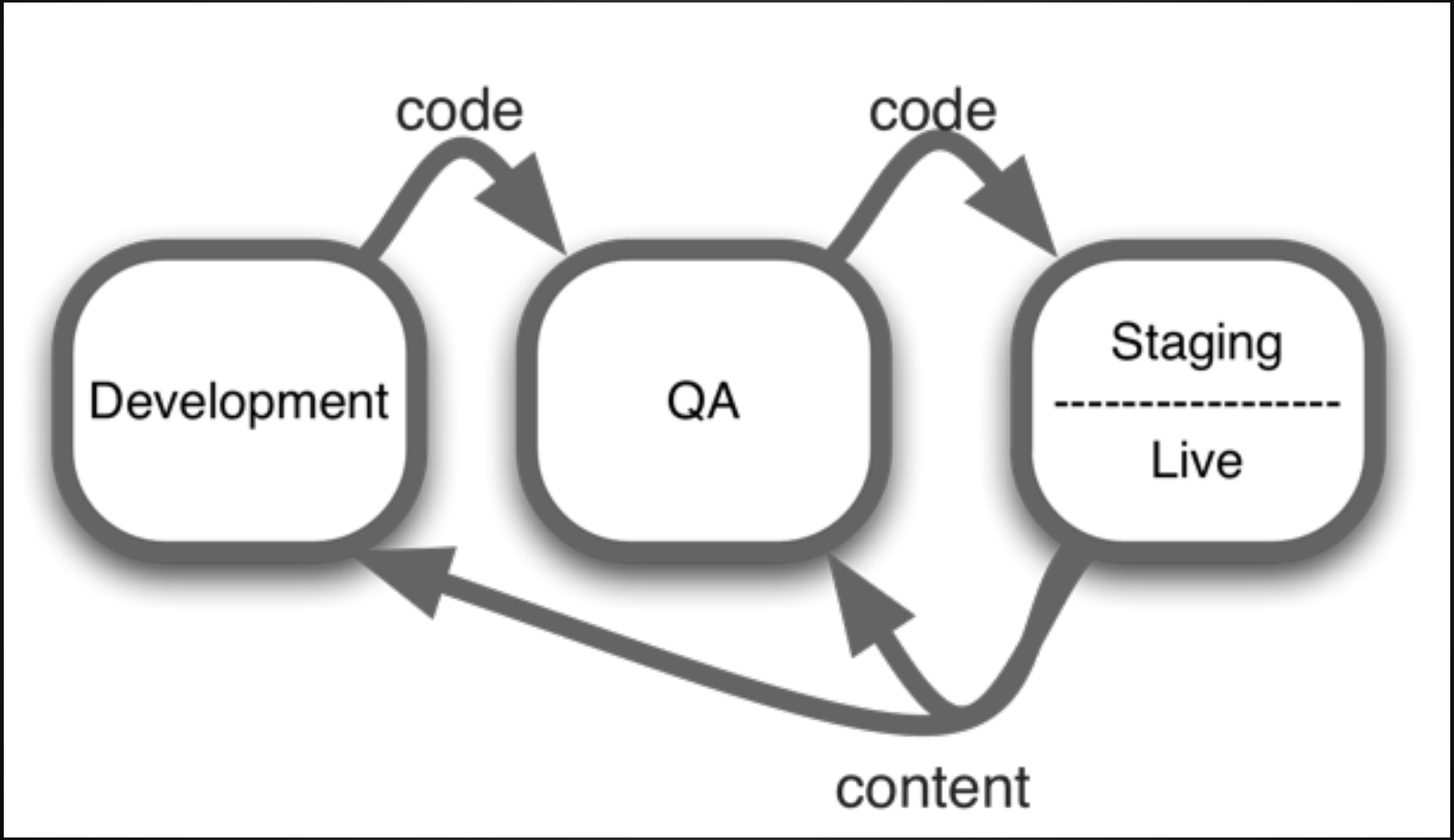
Agenda

- Common Development Workflow
- Version control
  - What, Why and When?
- World Known VCSs
- Git
  - Benefits
  - Comparison with others
  - Main concepts
  - Git Flow
  - Github Flow

Common Development Workflow

- Different environments
  - Development
  - QA
  - Staging
  - Production
- Team members
- Releases
- Versions
- Hotfixes

Environments



What?

VCS, from a Software Development perspective a version control system basically lets you:

- Track change on your source code
- Give developers the ability to create sandbox code environments to write code without affecting other team member’s code
- Somehow also give developers some code backup and restore facilities
-

Why?

- Its a must among development teams
- Critical when working on a team
- Keep control of versions, so you can identify and patch an specific version of code that its already on production.
- A way to rollback to previous code version
- Have sub-teams working on different set of features with the same code base

dot|creek

When?

Don’t worry about WHEN.. just use it!

dotcreek

World known VCS

- Subversion (SVN)
- CVS
- Mercurial
- Git
- Perforce
- Others

dot|creek

First!

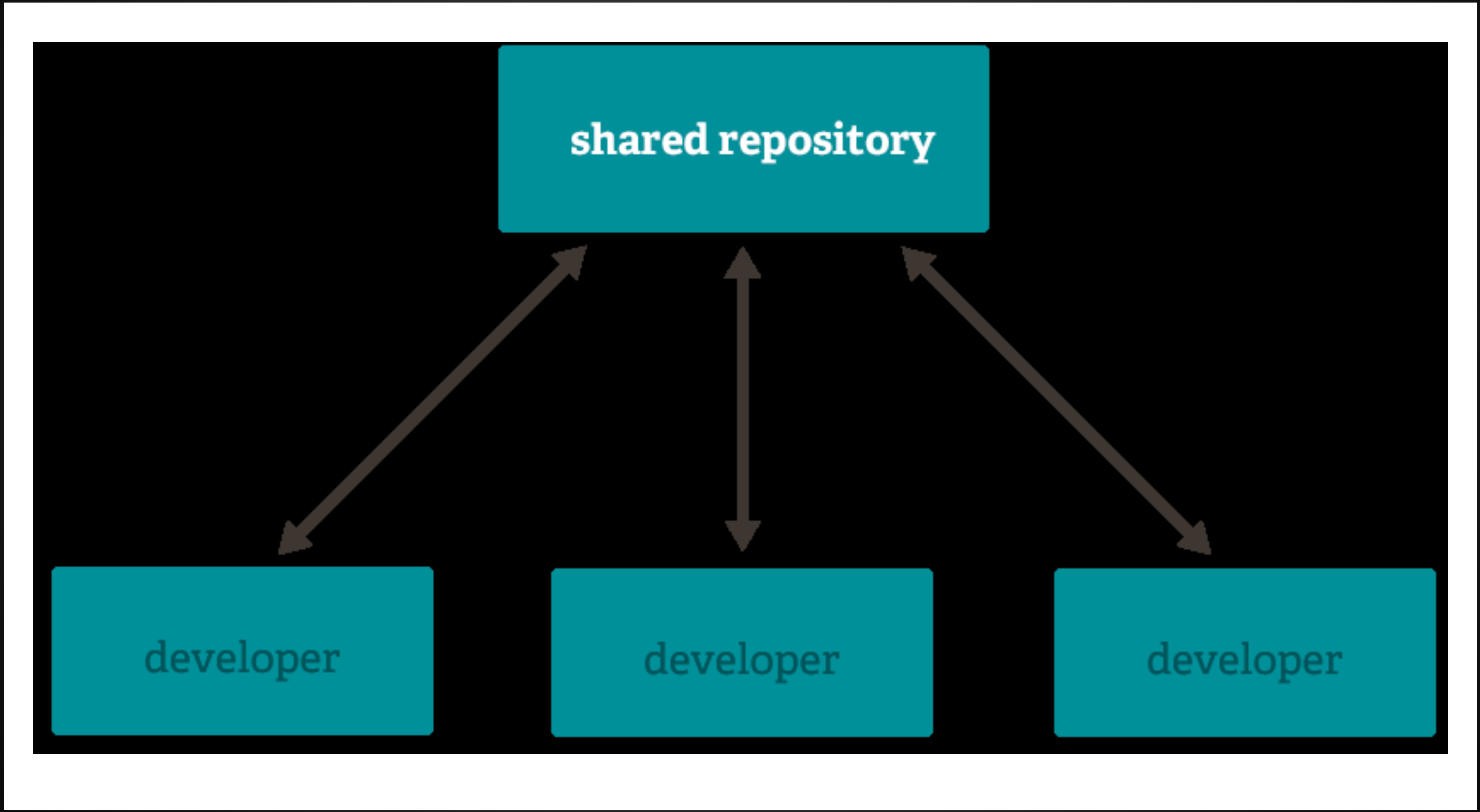
Git != Github

Git is the CVS System, Github the service

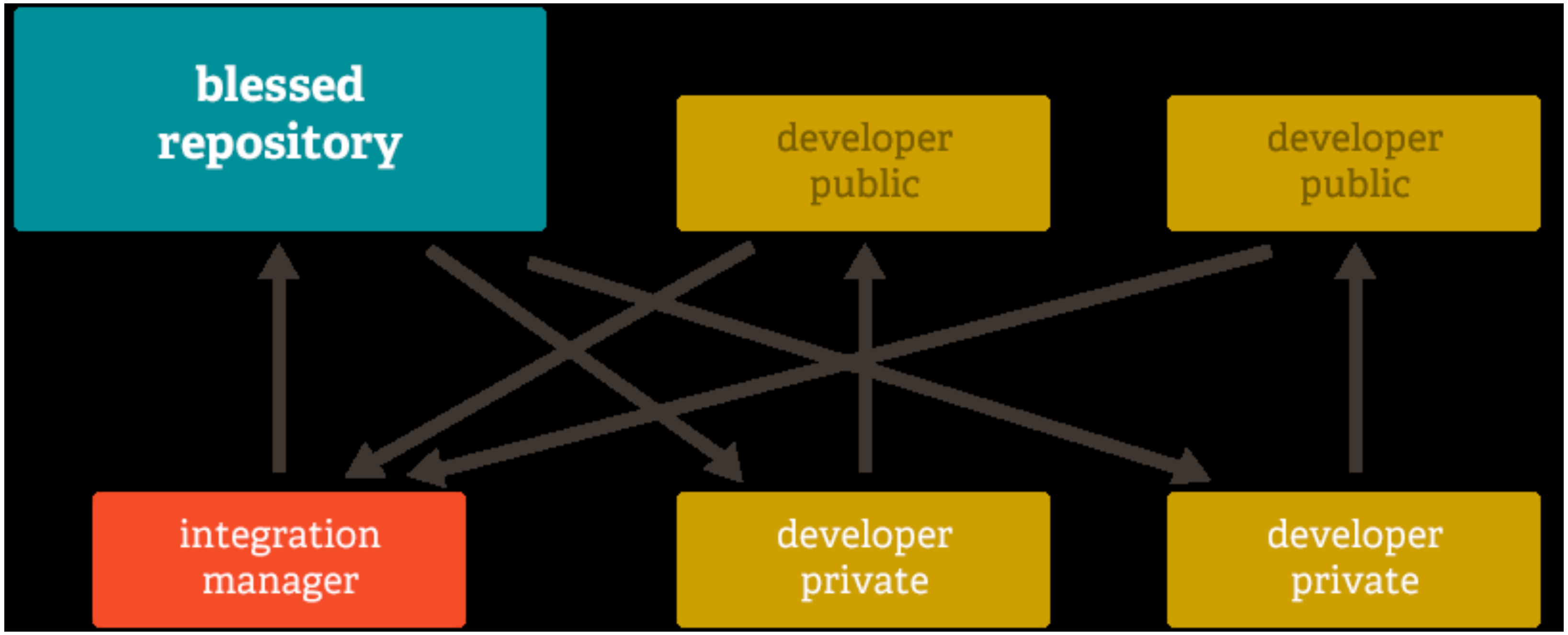
Git

- Created by Linus Torvalds
- Free
- Open Source
- Used in the Linux Kernel Project
- Branching and merging
- Small footprint and fast
- Data Assurance
- Staging Area
- Distributed

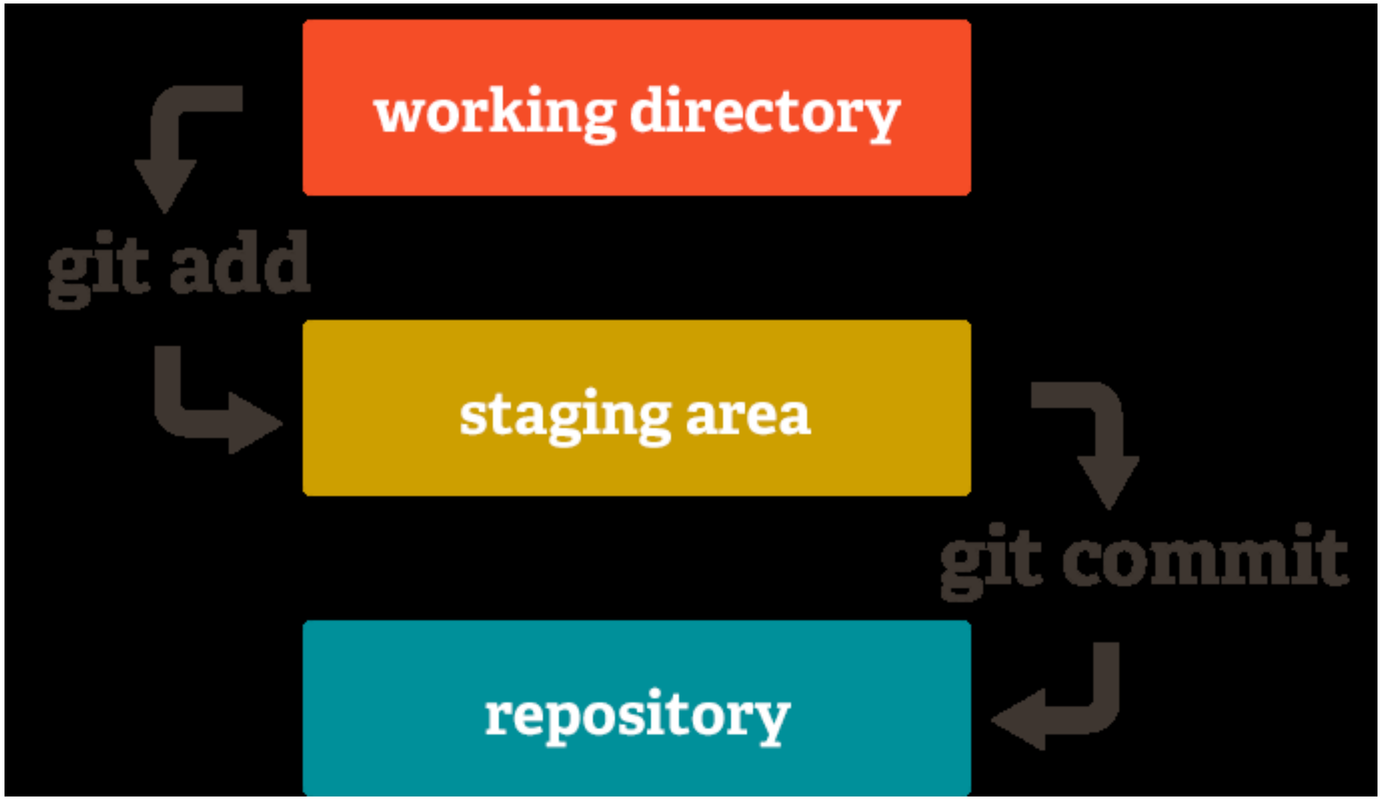
Centralized



Distributed



Staging Area



Repositories

- Its usually one per project
- Can have one or more branches, main and initial branch its called “master”
- Provides a whole history for each file
  - blame
  - statistics
  -

Git Commands

git config

- Used to configure your local git settings
- Basic settings:

```
$ git config --global user.name "Bladimir
Arroyo"
$ git config --global user.email bladimir.
b@gmail.com
```

git init

- Starts a new and empty git repository
- It starts as a local repository

```
$ mkdir myapp
$ cd myapp
$ git init
Initialized empty Git repository in
/Users/bladimirarroyo/Documents/UTN
/GIT/test/.git/
```

git status

- Shows the current repository status, indicating what hasn’t been added to the repository, which files are in stage
- Also indicates the current branch

git add

- Add new files to stage, that means they are ready to be committed.

git checkout

- Switch between branches
- Its also used to ignore our local changes to one or more files `git checkout file.rb`

git commit

- Creates a snapshot of the files that were in stage.
- This snapshot has an unique Id
- Its recommended to add a comment of what’s in the commit, recommend length limit of 50 characters

git reset

- Unstage changes that are on stage, that means you are now back as if you were just modified the files and no “git add” was executed

git branch

- Shows existent branches
- With `git branch  “name”` creates a new local branch
- With `git branch -b “name”` creates a new local branch and also run `“git checkout name”`

git pull

- Internally executes a
  - git fetch → which syncs our local with the remote repository
  - git merge → merges your local changes with the changes on the remote
- Bring the latest changes from the remote repository (other people change) to our local repository
-

git push

- Push all my local commits in my local branch to the remote
- In other words make my changes public to the others that are pulling from the same remote repository

git stash

- Its a queue of code changes
- Push to the queue the changes that haven’t move to stage on my local branch
- Temporal storage
- I can Pop those changes by calling
    - `git stash pop`
    - `git stash apply`

dot|creek