

Workshop GIT

A practical guide for using GIT.

Who am I ?

Rick van Ek

DevOps database engineer/DBA

Working with oracle products since 1992

Independend since 1996

Basetide associate 2017

Index

- Basic of Git.
- Git on the client.
- Git server.
- Multiple repositories.
- Git clients

What is Git?

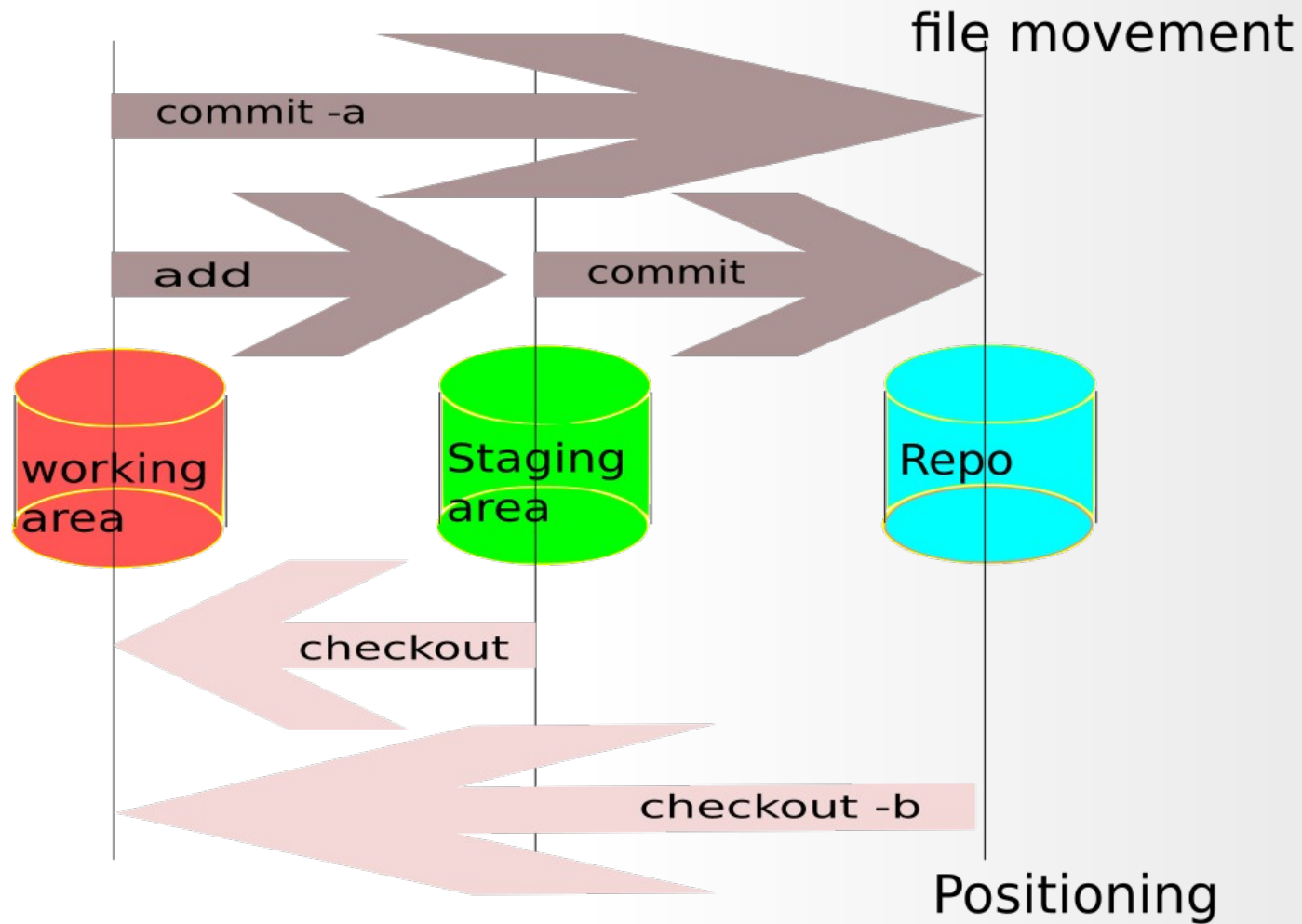
- A distributed source control system
- Strong support for non linear development
- Different way of versioning files.
- Most work is happening locally
- It is just as easy to screw up as in any other system.

GIT, what happens under the hood

Git thinks of its data more like a series of snapshots of a miniature filesystem. With Git, every time you commit, or save the state of your project, Git basically takes a picture of what all your files look like at that moment and stores a reference to that snapshot. To be efficient, if files have not changed, Git doesn't store the file again, just a link to the previous identical file it has already stored. Git thinks about its data more like a stream of snapshots.

GIT local repository

What happens in git?



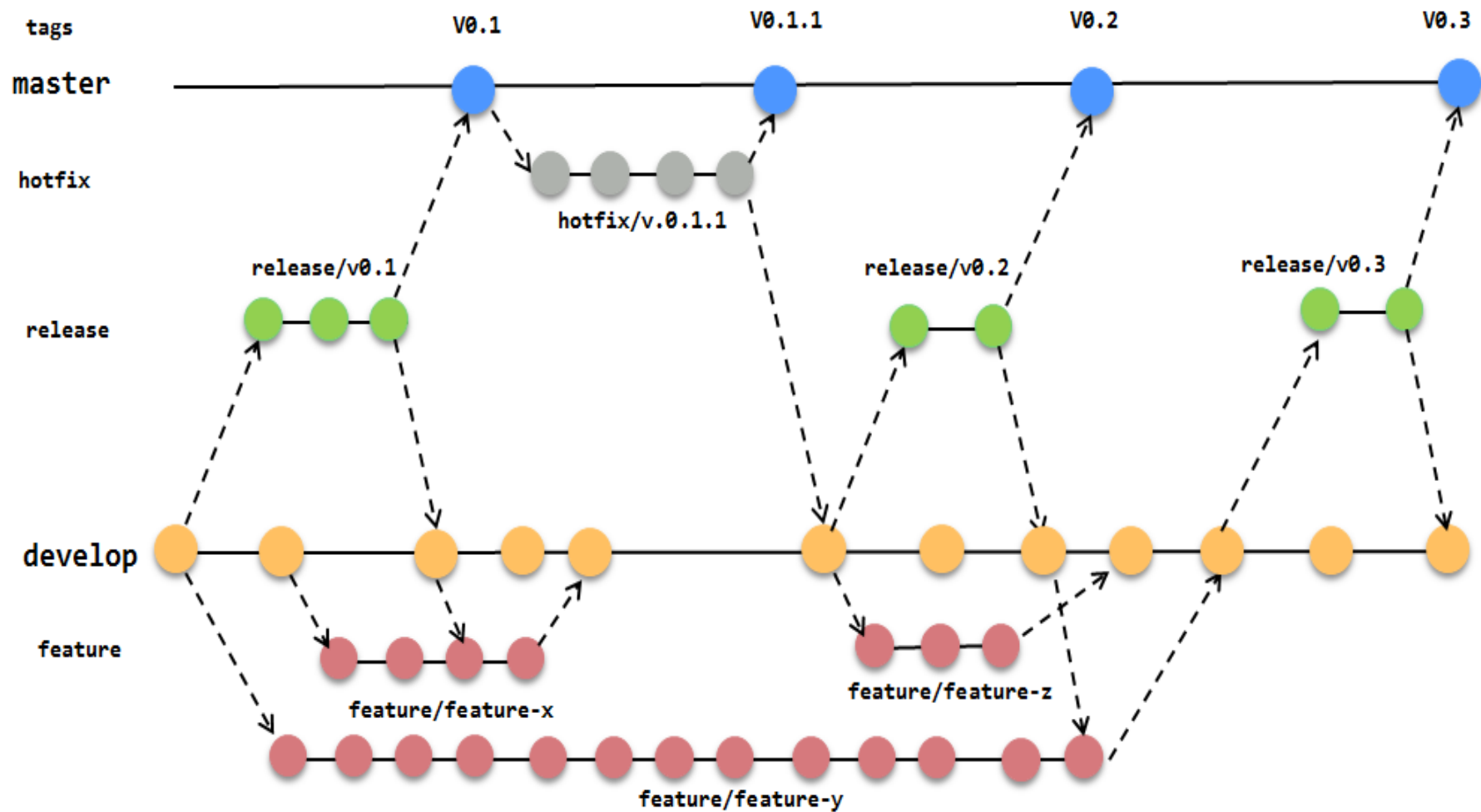
Creating repository

- Commands :
 - Git init
- Creates :
 - .git (directory with config file and index)
 - .gitignore (file in root of repository)
 - .Excluding : logfiles, tmp files, lock files etc...
 - .templates on internet

Git branches

- Why we need branches?
- How to make branches.
- How to clean them up .
- This is what we will push to remote repository..

Git branches



Working with branches

- `git branch <name>`
- `git checkout <name>`
 - `<-- do your work-- >`
- `git add [. | all | filename]`
- `git commit -m <comment>`
- `git status`
- `git diff`

GIT commit

Basically you cannot commit to much.

Git commit messages

Recommended Guidelines.

- 1 Separate subject from body with a blank line
- 2 Limit the subject line to 50 characters
- 3 Capitalize the subject line
- 4 Do not end the subject line with a period
- 5 Use the imperative mood in the subject line
- 6 Wrap the body at 72 characters
- 7 Use the body to explain what and why vs. how

GIT commit message example

```
$ git log --oneline -5 --author pwebb --before "Sat Aug 30 2014"
```

5ba3db6 Fix failing CompositePropertySourceTests

84564a0 Rework @PropertySource early parsing logic

e142fd1 Add tests for ImportSelector meta-data

887815f Update docbook dependency and generate epub

ac8326d Polish mockito usage

Git stash

- Saving work in between.
- Can be recalled at any time in any branch.
- Command:

`git stash push`

`git stash pop`

`git stash list`

`man git-stash` (documentation)

GIT TAG

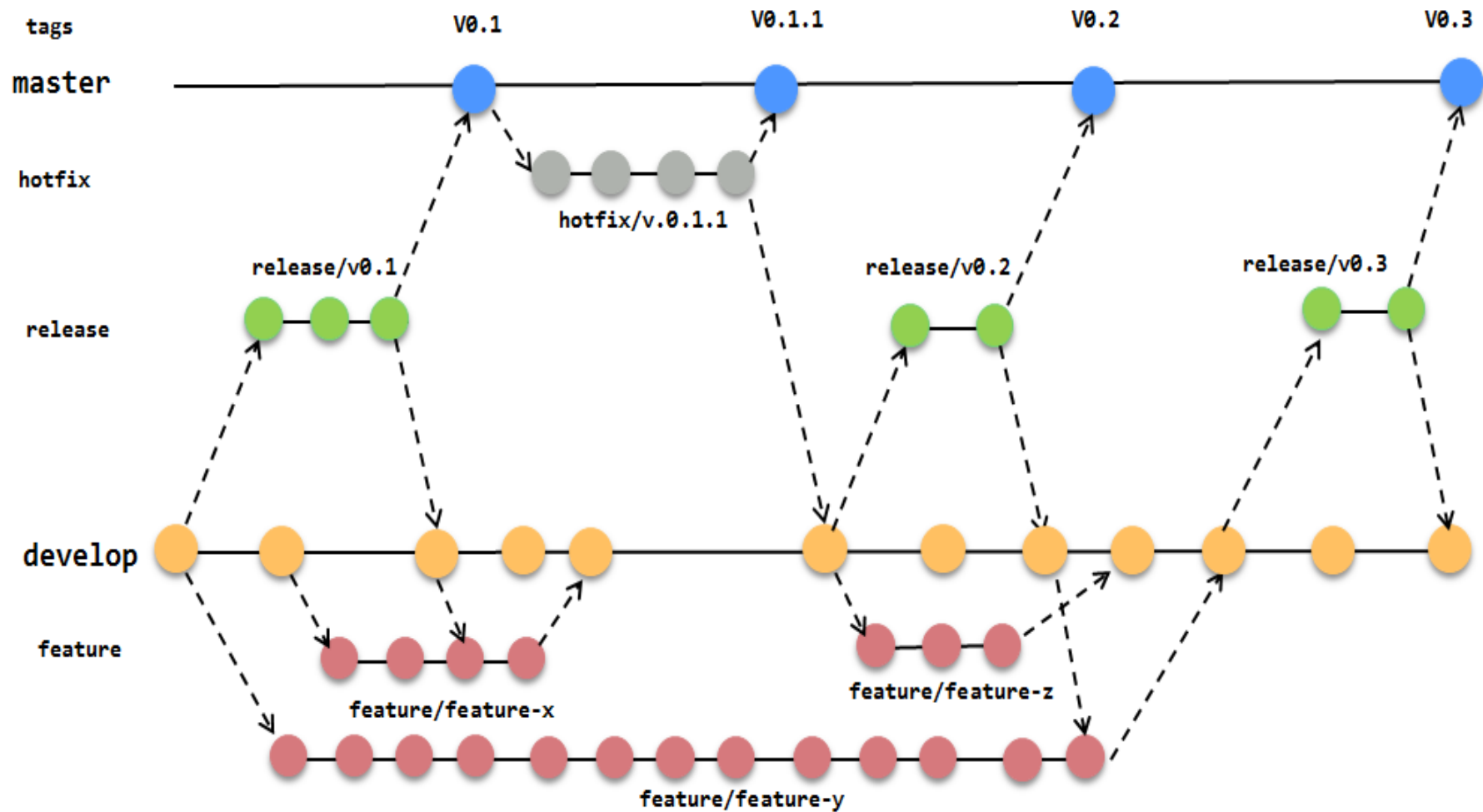
`git tag <name>`

- Create, list, delete or verify a tag object

Gives your references a readable, meaning full tag.

GIT demo time

Git branches



What are the consequences?

- Just about any file type can be used.
- Almost all work is done locally.
- Although it is very flexible, you need to use a structure.

Working with remote repository

Remote repository

- Setting up
- Configuration
- Ssh keys
- Pushing branches
- Cloning

Setting up server site

- git-shell for limited access
- gitweb, standard cgi scripts for basic webservice. (for use with apache)
- bitbucket

Hosted :

github, gitlab, bitbucket, etc

Configure server

```
$ cd /u01/git
```

```
$ mkdir <myproject>
```

```
$ cd <myproject>
```

```
$ git init --bare
```

SSH keys

If you do not want to enter username/password all the time , user ssh keys.

Place the public key in the .ssh directory in the home of the user git on the server .

Initial commit , pushing master

```
$ cd <myproject>
```

```
$ git init
```

```
$ git add .
```

```
$ git commit -m 'initial commit'
```

```
$ git remote add origin
```

```
user@gitserver:/u01/git/<myproject>
```

```
$ git push origin master
```

Starting local based on remote repository

```
$ mkdir <myproject>
```

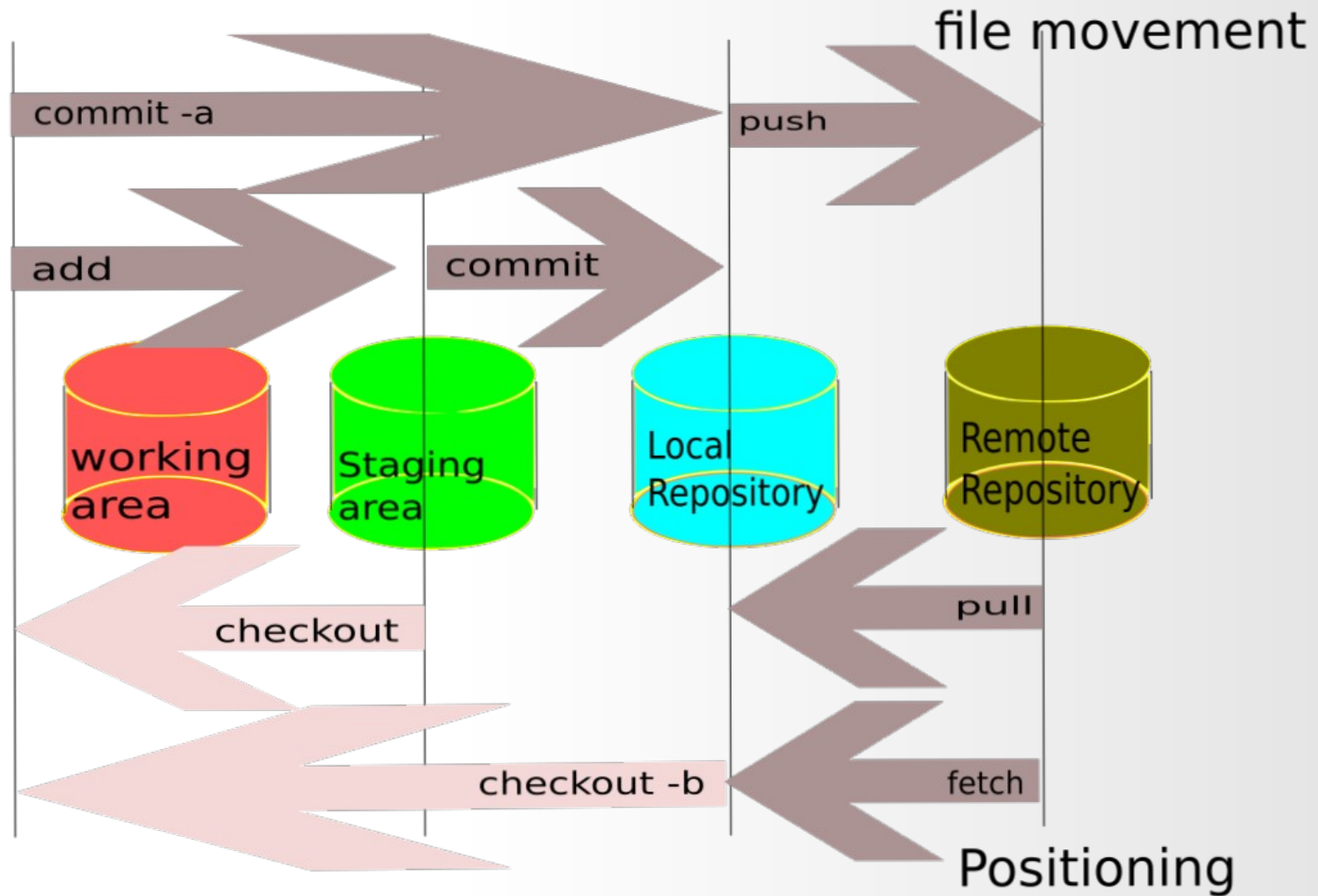
```
$ cd <myproject>
```

```
$ git clone user@gitserver:/u01/git/<myproject>
```

Clone repository

- Easiest way of starting
- Make an empty directory
- Do `git clone <remote repository> [--branch <branch name>]`

How Git works remote?



Sync your work with remote

- Pull, get the latest changes from remote.
 - Pull origin <your branch>
- Fetch, get a remote branch which does not exists local.
 - Fetch origin <remote branch>

Pushing branches.

Get your branch to the remote repository for merging.

Avoid merging conflicts, do a pull before push.

Pull request

Never, ever do changes on master.

- Lock the master
- Enforce pull request
 - Needs to be reviewed by other (four eyes principle)
 - Other does the merge with master
- Large project, appoint release master.

Pull request

rickvek / Lab_workshop Private

Unwatch 1

Star 0

Fork 0

Code

Issues 0

Pull requests 0

Actions

Projects 0

Security

Insights

Settings

No description, website, or topics provided.

Edit

[Manage topics](#)

1 commit

2 branches

0 packages

0 releases

Branch: master

New pull request

Create new file

Upload files

Find file

Clone or download

Switch branches/tags

Find or create a branch...

Branches

Tags

✓ master

default

Release

Latest commit eb8eeb0 2 days ago

Initial commit

2 days ago

ect.

Add a README

Pull request

rickvek / Lab_workshop Private

Unwatch 1 Star 0 Fork 0

<> Code

Issues 0

Pull requests 0

Actions

Projects 0

Security

Insights

Settings

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

base: master

compare: Release


✓ Able to merge. These branches can be automatically merged.

Release

Write Preview

AA B i “ <> 🔗 ☰ ☷ ✓ @ 📌 ↶

Merging our project :-)

Attach files by dragging & dropping, selecting or pasting them. 

Create pull request

Reviewers

No reviews

Assignees

No one—assign yourself

Labels

None yet

Projects


None yet

Milestone

No milestone

Release #1

 **Open** rickvek wants to merge 6 commits into `master` from `Release` 

 Conversation **0**

 Commits **6**

 Checks **0**

 Files changed **45**








rickvek commented 1 minute ago

+ 😊 ...

Merging our project :-)



rickvek added 6 commits 2 days ago

-   Added filesXX.txt 50b2543
-   Added Proj2 files 20b3dac
-   added Proj3 file 21ddf8c
-   added Project1 files 5851c10
-   Merge branch 'Proj1' into Development b197c18
-   Hot Fix done 7bb0860

Add more commits by pushing to the **Release** branch on **rickvek/Lab_workshop**.



Continuous integration has not been set up

[GitHub Actions](#) and [several other apps](#) can be used to automatically catch bugs and enforce style.



This branch has no conflicts with the base branch

Merging can be performed automatically.

Merge pull request



or view [command line instructions](#).

Multi sites

methods to use repository of multiple sites.

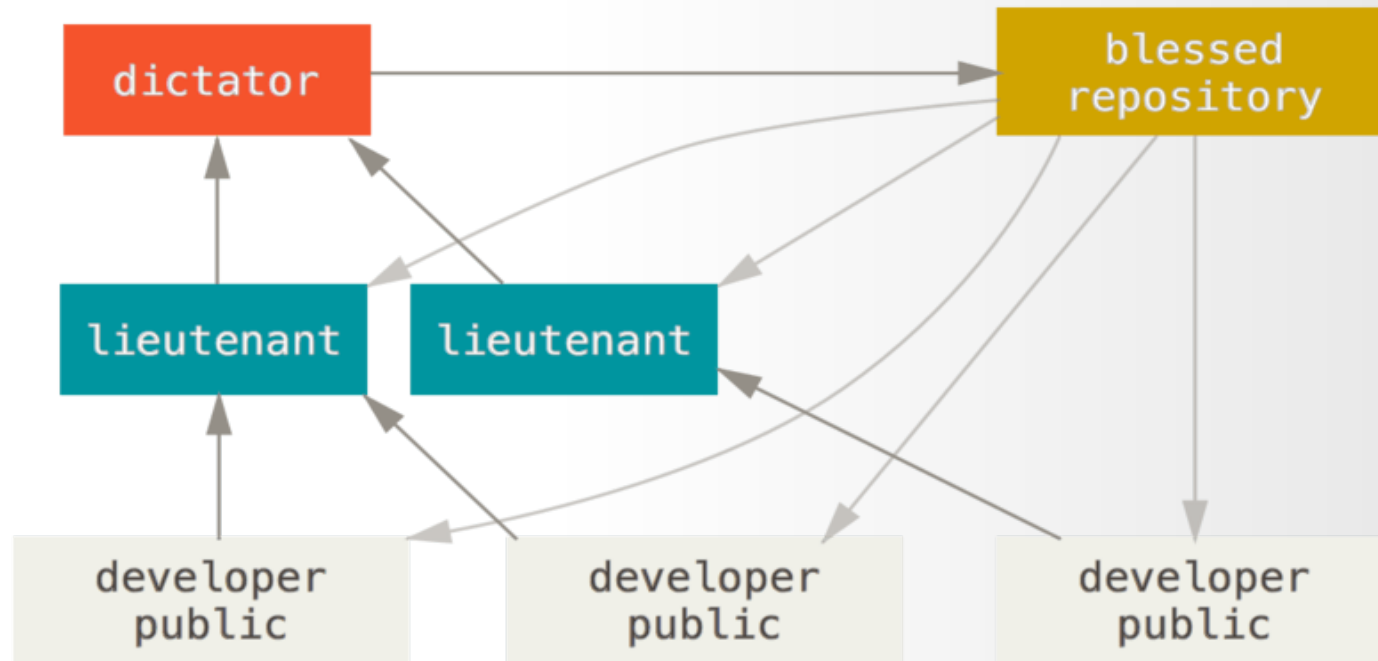
Git submodules 1

- Including another project in your project
- Name submodule same as the repository
- Creates subdirectory with this name
- Creates .gitmodules
 - Contains the names and path to repository

Git submodules 2

- A submodule is a repository embedded inside another repository.
- Command :
 - git submodule <command> [option]

Multi remote repositories.



Git client

Linux, Apple & Windows :

<https://git-scm.com/download/gui/windows>

Tools with build in git support

- sqldeveloper
- atom editor
- pycharm
- eclipse
- visual studio (code)
- intelliJ
- etc.

Available in most developer tools.

Interesting documentation

- Online book
 - <https://git-scm.com/book/en/v2>
- Same page a pdf version v1 to download

Tips & Tricks

- Working on windows
 - Unix files needs permission set
 - `git ls-files --stage`
 - `git update-index --chmod=+x foo.sh`
 - `git commit -a -m "Permissions set"`
 -

Lab belonging to workshop

- Start lab with download:

```
mkdir lab
```

```
cd lab
```

```
[lab]$ git clone https://github.com/rickvek/Workshop-GIT.git
```

```
Cloning into 'Workshop-GIT'...
```

```
remote: Enumerating objects: 99, done.
```

```
remote: Counting objects: 100% (99/99), done.
```

```
remote: Compressing objects: 100% (56/56), done.
```

```
remote: Total 99 (delta 53), reused 79 (delta 41), pack-reused 0
```

```
Unpacking objects: 100% (99/99), done.
```

```
[lab]$ cd Workshop-GIT/
```

```
[Workshop-GIT]$ <find lab document Lab_GIT_Workshop >
```

Questions ?