

# Quick Intro to Git version control

July 15, 2014

# Contents

Intro

Commands

Workflow

Pull requests

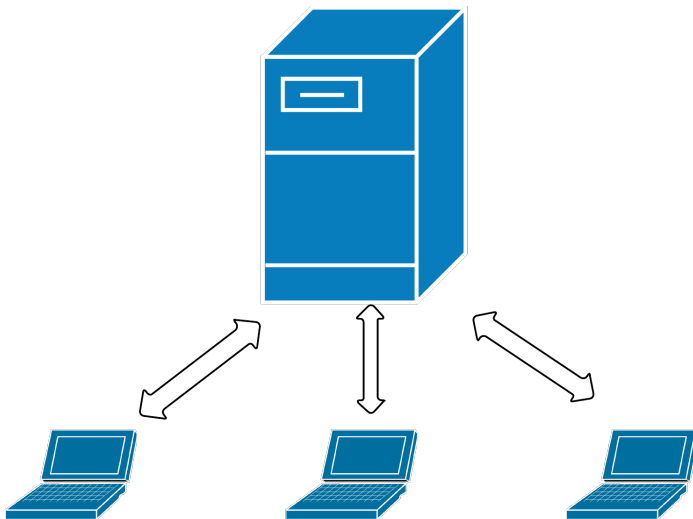
Resources

Homework

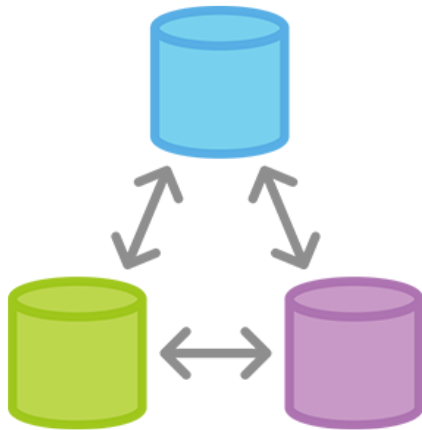
# Intro

Git is an open source, **distributed** version control system designed for speed and efficiency.

# Centralized paradigm (CVS, SVN, Perforce)



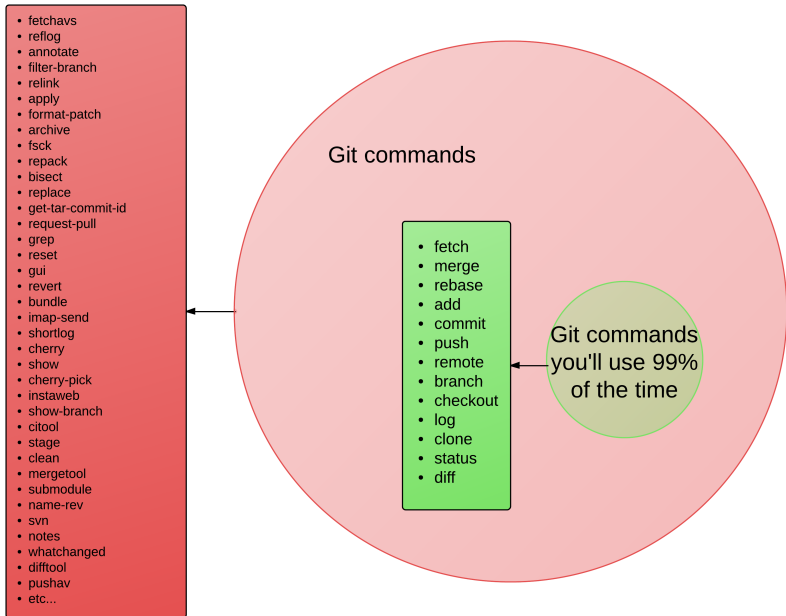
## Distributed paradigm (Git, Mercurial)



# Git commands



# Git commands

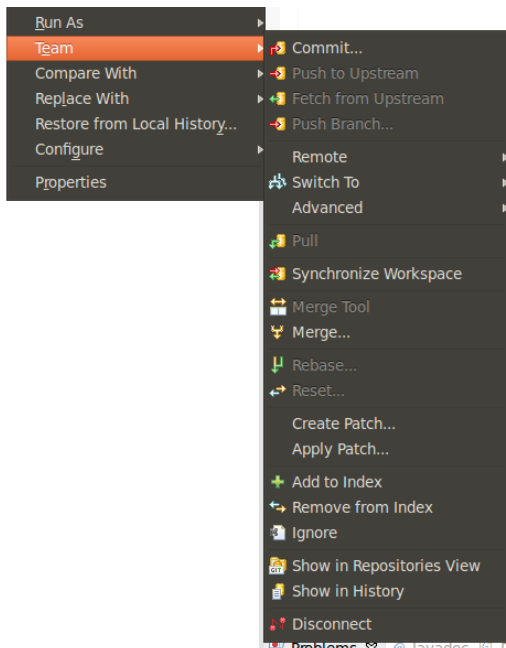


# Git commands

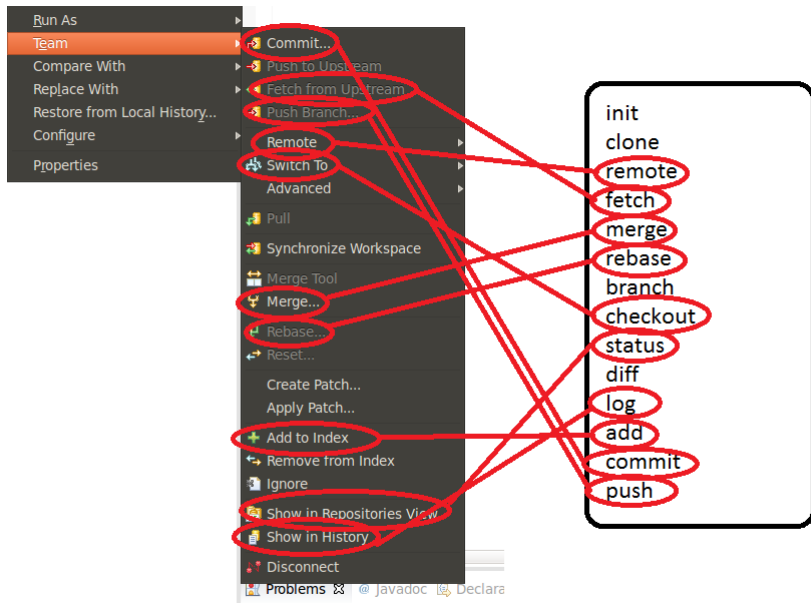
init  
clone  
remote  
fetch  
merge  
rebase  
branch  
checkout  
status  
diff  
log  
add  
commit  
push



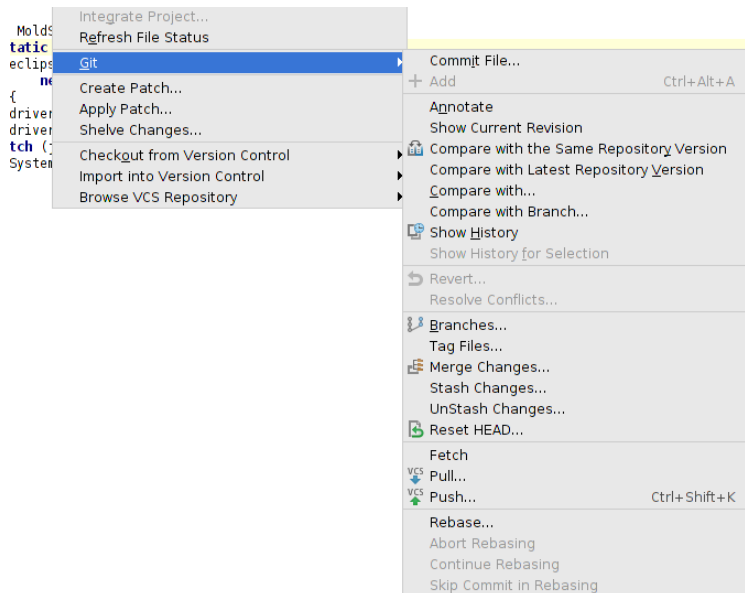
# Eclipse

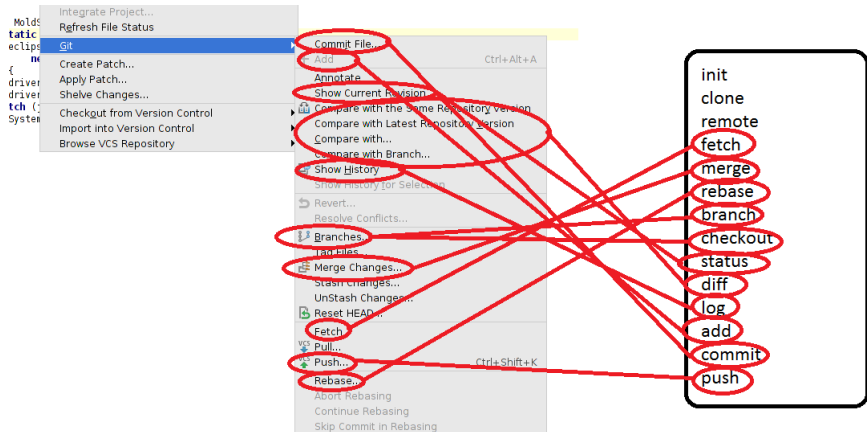


# Eclipse



# IntelliJ





\$ git config



```
$ git config --global user.email abeaulne@eclipseoptions.com  
$ git config --global user.name 'Alex Beaulne'
```

is necessary to get started. **Use short email for proper integration with Stash**

```
$ git init
```



```
$ git init
```

makes the current working directory a Git repository

\$ git init



a hidden directory `.git` is inserted in the root directory of the Git repository. Unlike CVS or SVN, no `.git` directory is inserted in each subdirectories

## \$ git remote



A Git remote is best thought as an alias for a URL. It's an address to a remote Git repository from which you can fetch or push source code.



\$ git remote



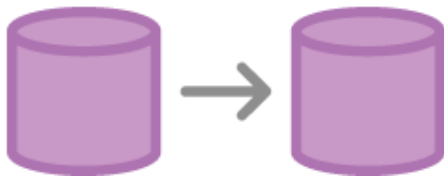
Create a new remote:

```
$ git remote add remote_name remote_url
```

List remotes:

```
$ git remote -v
```

\$ git clone



```
$ git clone remote_url
```

create a local copy of the Git repository hosted at remote\_url

\$ git status



```
$ git status
```

shows which files on the current branch are untracked, modified and staged for commit

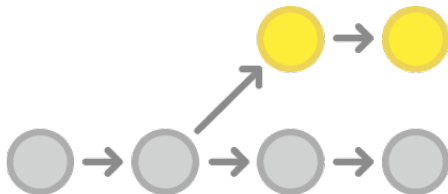
\$ git pull



```
$ git pull remote_name_or_url branch_name
```

apply new commits from the remote Git repository to the local repository

\$ git branch



```
$ git branch new_branch_name
```

is used to create a new branch from the current branch

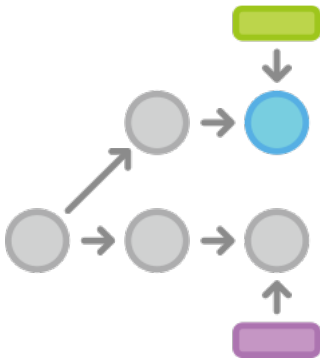
## \$ git branch

Without a branch name, it will list all the branch in the local repository:

```
$ git branch
  develop
* featureJIRA12
  master
```

the asterisk shows the current branch

\$ git checkout



```
$ git checkout branch_name
```

is used to switch between branches

## two notes on Git commits

I. A Git commit is 'repository-wide'. It is a snapshot of the complete repository at that point in time. This is different to CVS per-file commits.



## two notes on Git commits

II. Pushing a commit to a canonical Git repository is a three stages process: (i) add (ii) commit (iii) push. Unlike CVS, a Git commit does not push anything to a remote repository. One needs to push to do so.

\$ git add



```
$ git add [filename1 [filename2]...]
```

is used to add files to staging area

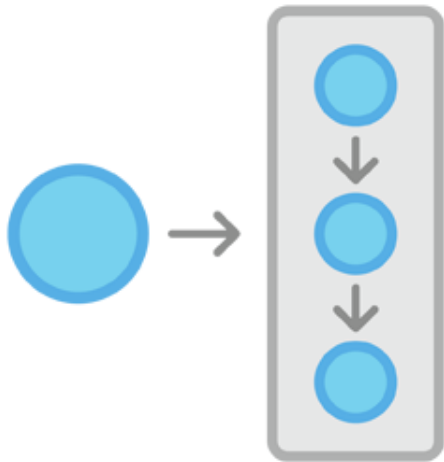
\$ git add



```
$ git add [filename1 [filename2]...]
```

If no file is specified, all modified files on the current branch are added to staging area

\$ git commit



```
$ git commit -m "commit msg JIRA-XXX"
```

commits the changes previously added to staging area

\$ git diff



```
$ git diff [filename1 [filename2]...]
```

shows lines that have changed since latest commit on branch

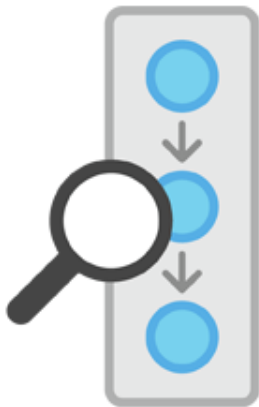
\$ git diff



```
$ git diff [filename1 [filename2]...]
```

If no file is specified, show diff for all modified files on the current branch

\$ git log



```
$ git log [-n] [branch_name]
```

shows latest n (or all) commits for branch 'branch\_name' (default to current branch)

\$ git push



```
$ git push remote_name_or_url branch_name
```

pushes local commits to remote branch



# Workflows

- ▶ Centralized workflow
- ▶ Feature branch workflow
- ▶ Gitflow workflow
- ▶ Forking workflow

# Workflows

- ▶ Centralized workflow
- ▶ Feature branch workflow
- ▶ Gitflow workflow
- ▶ Forking workflow

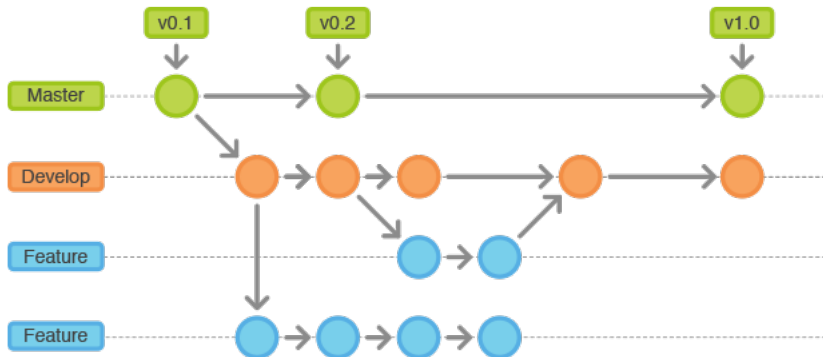
# Gitflow



# Gitflow



# Gitflow



# Gitflow

**master branch** stores the official release history. It is only changed when the Bamboo users merge the develop branch into it

**develop branch** serves as an integration branch for features. It is only changed when feature branches are merged into it (via pull request, no direct push)

**feature branches** are where new features reside. Features branches are branched from the develop branch, and their changes are incorporated in the canonical repo via pull requests to the develop branch

# Pull requests

- ▶ Not part of Git per se
- ▶ More a feature of Git hosting solutions (Stash, Github, etc)
- ▶ Great for code reviews

# Pull requests

demo



# Resources

- ▶ These slides are on Confluence (<http://confluence/display/~abeaulne/Intro+to+Git+presentation>)
- ▶ More detailed instructions Trading Systems team at <http://confluence/display/TSD/Setting+up+GIT>
- ▶ Atlassian has a great straightforward tutorial at <https://www.atlassian.com/git/>

# Homework

- ▶ fetch canonical 'practice' repository at `http://stash/scm/core/practice.git`
- ▶ create a feature branch (branched out of develop branch) with your name
- ▶ add your name to the README.txt
- ▶ commit your change locally
- ▶ push your commit to remote feature branch at canonical 'practice' repo
- ▶ create a pull request, adding me (Alex) and one of your colleagues/superiors as reviewer

## Solution

```
~$ git clone http://stash/scm/core/practice.git
~$ cd practice/
~/practice$ git checkout develop
~/practice$ git branch alexb
~/practice$ git checkout alexb
~/practice$ vim README.txt
~/practice$ git add README.txt
~/practice$ git commit -m "added my name"
~/practice$ git push origin alexb
```

Finally go to

<http://stash/projects/CORE/repos/practice/browse> to  
create pull request