# Git Workflow

D. Kresan
GSI, Darmstadt

# Git

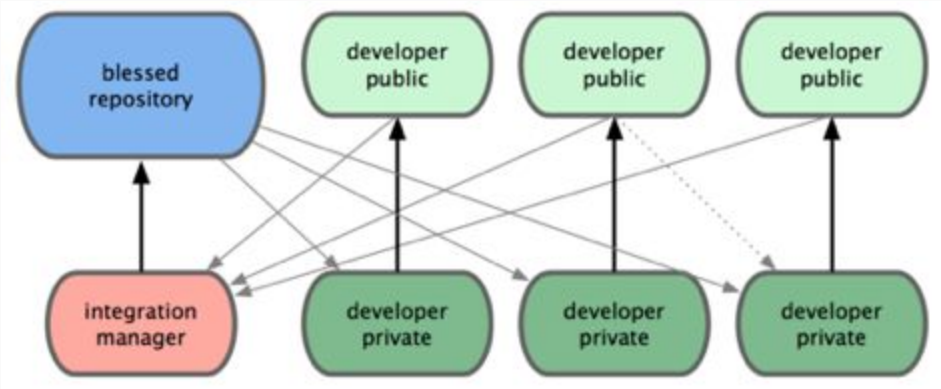
Git is a distributed version control system (DVCS)

In contrast to “svn checkout”, “git clone” gives a complete copy of the entire history of the project

- Nearly every operation can be done offline
- Multiple full backups - no point of failure
- Workflows - possibility to synchronize with other git repositories - push and pull

# Workflow example: central and distributed repositories

R3BRootGroup




GitHub

local

Full article:
http://blog.teamtreehouse.com/why-you-should-switch-from-subversion-to-git

# Workflow for R3BRoot

Inherited from FairRoot group. Proposed and implemented by Anar Manafov

Comprehensive documentation:

https://github.com/AnarManafov/GitWorkflow/blob/master/GitWorkflow.markdown

# Prepare the environment

# 1. Setup git configuration - once per machine

```
git config --global branch.autosetuprebase always

git config --global user.name "FirsName LastName"

git config --global user.email johndoe@example.com

git config --global core.ignorecase false
```

# 2. Fork

Using github fork the main repo.

This will create a complete copy of the main repository under your GitHub account. You will have write access to this forked repository.

# 3. Create a local copy of the forked repo

git clone url_of_the_fork

This will download the repository from your GitHub account and create a working version on your local computer

# 4. Attach main repository

| | |
|---|---|
| git remote add mainrepo mainrepo_url | Add location of main repository to the list of remotes |
| git fetch mainrepo | Download the full main repository to your local computer |

# 5. Create a local dev branch

git checkout -b dev mainrepo/dev

Will create new “dev” branch in your local repository, based on “dev” branch of mainrepo

# 6. Push the local dev to your forked repo

git push -u origin dev

This will upload the local dev branch to the forked repository on GitHub (origin)

# Create a feature branch

| | |
|---|---|
| git fetch mainrepo | Get the latest main repository |
| git checkout -b featureXXX mainrepo/dev | Create new branch based on dev branch of main repository |

git push -u origin featureXXX

Upload the feature branch to your forked repository and tell git to track it

# Synchronize your branch (update)

git fetch mainrepo

git checkout featureXXX

git rebase mainrepo/dev

# In case of conflicts

- Resolve conflicts if any.
- Stage each modified file "git add " after conflicts are resolved.
- You can also use "git checkout --theirs/--ours " to help to resolve conflicts.
- Use "git rebase --continue" to continue rebasing.

# Push updated local branch to origin

git push origin

will probably not work - history has been changed, your commits will come on top of commits in mainrepo/dev

git push -f origin

# Prepare pull request

# Always synchronize first

git fetch mainrepo

git checkout featureXXX

git rebase mainrepo/dev

# In case of conflicts

- Resolve conflicts if any.
- Stage each modified file "git add " after conflicts are resolved.
- You can also use "git checkout --theirs/--ours " to help to resolve conflicts.
- Use "git rebase --continue" to continue rebasing.

# Squash all of your commits

git rebase -i mainrepo/dev

First commit - option “pick” (“p”)

Following commits - option “squash” (“s”)

Edit a common commit message

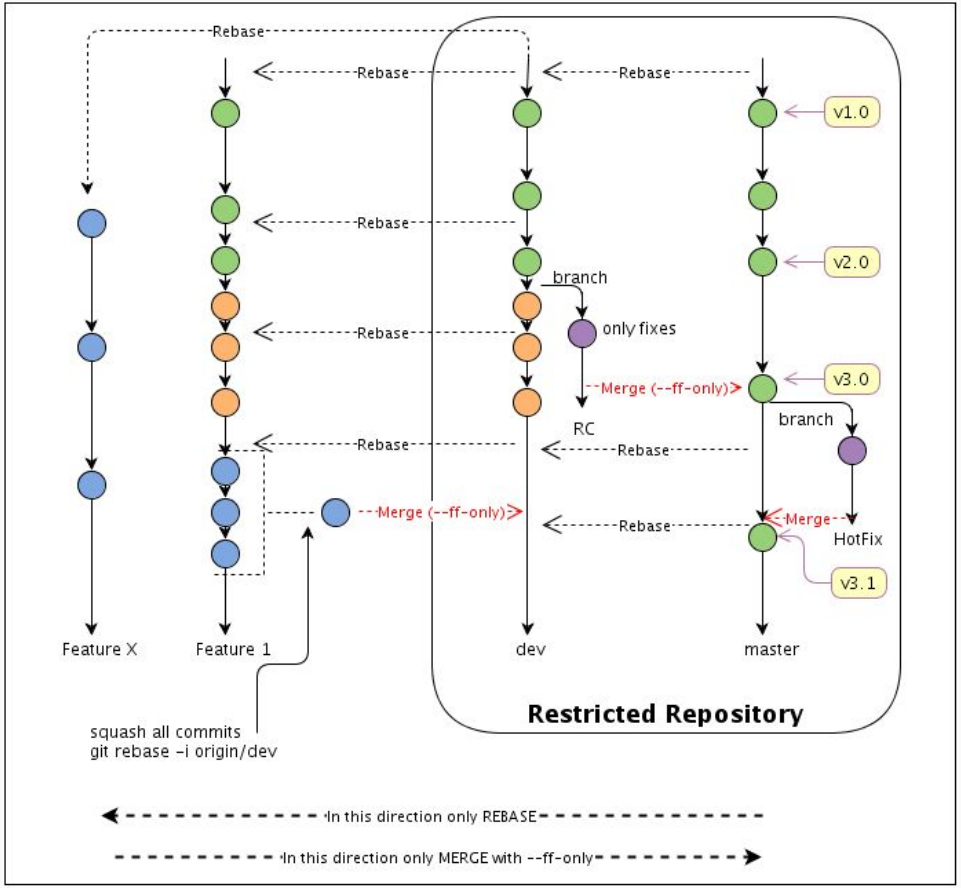
# Upload to GitHub

git push -f origin

Create new pull request for your feature branch using GitHub web interface

Always chose “dev” branch as base

Cross-check the changed files, edit comment message, create...


Rebase
Rebase
Rebase
Rebase
Rebase
Rebase
Rebase
Rebase
Merge (--ff-only)
Merge (--ff-only)
Merge
branch
only fixes
RC
branch
HotFix
v1.0
v2.0
v3.0
v3.1
Feature X
Feature 1
dev
master
Restricted Repository
squash all commits
git rebase -i origin/dev
In this direction only REBASE
In this direction only MERGE with --ff-only