Git Workshop

- Welcome!
- References:
  - git help <command>
  - http://git-scm.com/book

Format

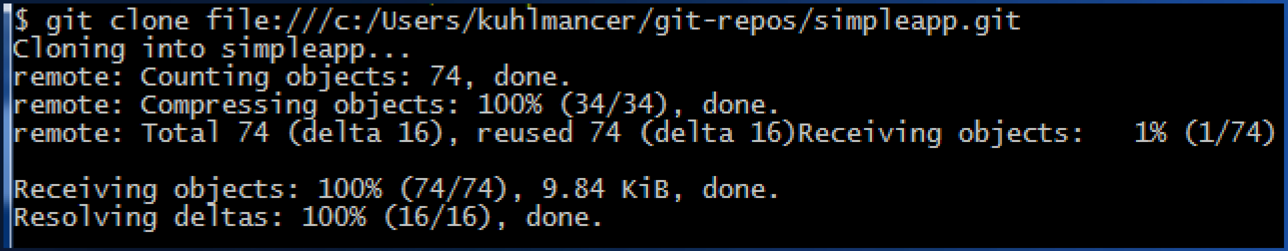
- Pair up
- 1: Instructions for person 1
- 2: Instructions for person 2

Tools

- git bash
  - access git command line interface
  - some linux tools (find, grep)
- git gui
  - make commits
- gitk
  - viewing repo history
  - commit and branch manipulation

Getting Started: Clone

- Determine group number <n>
- 1+2: git help clone
- 1+2: git clone git@10.35.66.139:git-repos/simpleapp<n>.git simpleapp
- password is “workshop” without quotes
- should see something like:



```
$ git clone file:///c:/Users/kuhlmancer/git-repos/simpleapp.git
Cloning into simpleapp...
remote: Counting objects: 74, done.
remote: Compressing objects: 100% (34/34), done.
remote: Total 74 (delta 16), reused 74 (delta 16)Receiving objects:   1% (1/74)

Receiving objects: 100% (74/74), 9.84 KiB, done.
Resolving deltas: 100% (16/16), done.
```

Viewing the repo: gitk

- 1+2: alias gitk='gitk --all'
  - show all branches
- 1+2: gitk
  - remote branch: remotes/origin/master
  - local branch: master
  - tag: v1.0.M1
  - patch vs tree view (lower right)
  - searching repo (middle)
  - searching commit (middle)

Create a Local Branch

- 1+2: git branch topic1; git checkout topic1
  - shorter:
    - git checkout -b topic1
- more info:
  - git help branch
  - git help checkout

1 - Do some work

- 1: open GameTest.java
- 1: remove the following two lines:
  - import simpleapp.Game;
  - import simpleapp.SecretGenerator;

1 - Commit w/ Git gui

- 1: git gui
- notice:
  - unstaged changes
  - staged changes
  - file view
    - stage hunk/line, adjust context
  - commit message
  - rescan
  - hotkeys

1 - Commit w/ Git gui (cont.)

- 1: commit changes
  - stage GameTest.java (git add <file>)
  - add a commit message 'removed imports from same package'
  - press commit button (git commit -m <msg>)
- 1: gitk to see changes
  - commit is on topic1 branch
  - nothing sent to central repo

1 - merge master to topic1

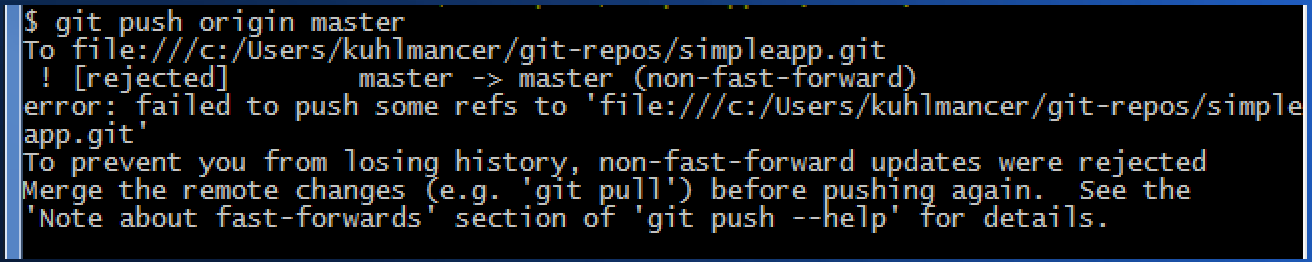
- 1: git checkout master; git merge topic1
  - fast-forward merge
- 1: gitk
  - master points to same commit as topic1
  - fast-forward = no merge commit
  - use gitk often to check repo

1 - Push commit to central repo

- 1: git push origin master
  - origin is default name for clone source
  - syncs master branch with origin's master
  - similar to svn commit
  - anyone with access to origin repo can get the commit(s) now

2 - Do some work, commit, push

- 2: change the text in README.md
- 2: commit changes and push
  - fails



```
$ git push origin master
To file:///c:/Users/kuhlmancer/git-repos/simpleapp.git
 ! [rejected]        master -> master (non-fast-forward)
error: failed to push some refs to 'file:///c:/Users/kuhlmancer/git-repos/simple
app.git'
To prevent you from losing history, non-fast-forward updates were rejected
Merge the remote changes (e.g. 'git pull') before pushing again.  See the
'Note about fast-forwards' section of 'git push --help' for details.
```

2 - Pull

- 2: git checkout master
- 2: git pull origin
  - vs git fetch origin
- 2: gitk
  - 1's commit now shows up
  - master branch pointer has moved up
  - 2's work is on a separate branch

2 - Merge

- 2: git checkout topic1
- 2: git merge master
  - new merge commit
  - but there's another way that will give linear history

2 - Undo merge commit

- 2: git reset --hard HEAD~1
  - go back 1 commit from HEAD commit
  - --hard means undo any working copy or index changes as well as moving the branch pointer
  - ok to do because the merge commit is still only local
- 2: gitk
  - back where we were before merge
  - could do the reset with gitk instead

2 - Rebase

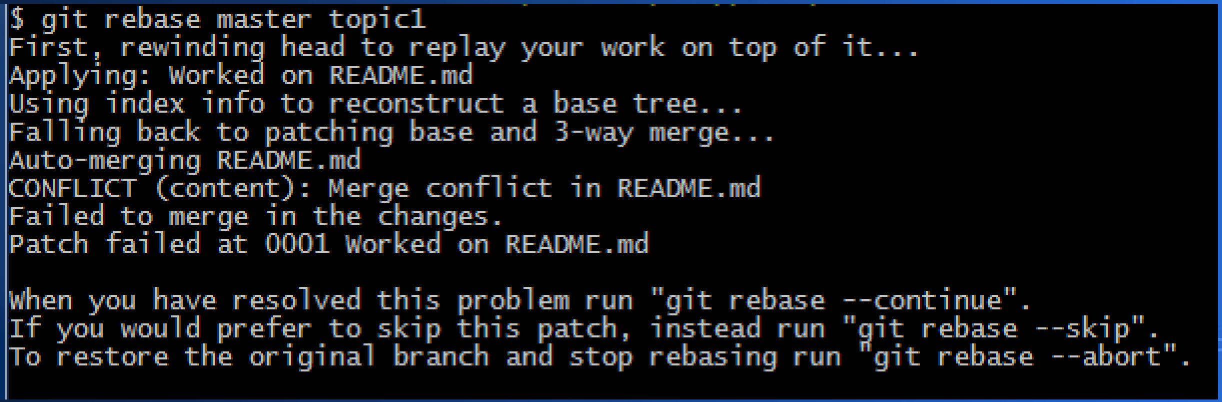
- 2: git rebase master topic1
  - checkout topic1
  - replay topic1's commits from the divergent commit onward onto master's HEAD
- 2: gitk
  - linear history with topic1's branch now on top of master's commits

Rebase (cont.)

- VERY IMPORTANT:
  - DO NOT REBASE PUBLISHED COMMITS
  - it is for replaying the work from your local branch onto master so you can push a linear history
  - it creates new commits

1 - Conflicts

- 1: edit the text in README.md
- 1: commit the edits
- 1: git checkout master; git pull origin
- 1: try to rebase topic1 onto master
  - git rebase master topic1
  - FAILS



```
$ git rebase master topic1
First, rewinding head to replay your work on top of it...
Applying: Worked on README.md
Using index info to reconstruct a base tree...
Falling back to patching base and 3-way merge...
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Failed to merge in the changes.
Patch failed at 0001 Worked on README.md

When you have resolved this problem run "git rebase --continue".
If you would prefer to skip this patch, instead run "git rebase --skip".
To restore the original branch and stop rebasing run "git rebase --abort".
```

1 - Resolve Rebase Conflicts

- 1: edit README.md to resolve conflict
- 1: add README.md to index to mark resolved
  - git add README.me
  - or use git gui and stage the file
- 1: git rebase --continue
- gitk
  - linear history again

2: Pushing Branches

- 2: git checkout -b readme
- 2: edit README.md and commit
- 2: git push origin readme
  - just like we were doing with origin and master
  - will create a readme branch on origin
  - can now be fetched or pulled by person 1
  - REMEMBER NOT TO REBASE PUBLISHED BRANCHES

1: Tagging

- tags are pointers to commits, like branches
- 1: git checkout master; git merge topic1; git checkout master
- 1: git help tag
- change version in pom.xml to 1.0.M2 and commit
- 1: git tag v1.0.M2 -m “Tagging 1.0.M2”
  - can also use gitk: right-click on commit message to get the correct context menu
  - can sign tags with GPG signature
- 1: git push origin v1.0.M2

2: Removing Branches

- local:
  - 2: git branch -d readme
    - error about unmerged commits
  - 2: git branch -D readme
    - removes anyway
- remote
  - 2: git push origin :readme
    - notice the colon before readme

git stash

- I use it less and less
- prefer to make a branch and commit

git-svn

- bridge between git and svn
- use git locally, publish commits to svn repository
- been using it here at BAE for more than a year
- git fetch → git svn fetch
- git push → git svn dcommit

The End

- That's all.