Version Control

Version Control

- Purpose
  - Maintain a history of changes to files and directories in a way that you can
    - Compare current files with past files
    - Revert work to the content of past files
    - Create paths to explore changes to files and, if you like the changes, make those changes be the “most current” files
  - Allow collaborators to develop code independently and merge their work easily

Repositories

- A repository is a collection of files that are managed under version control.

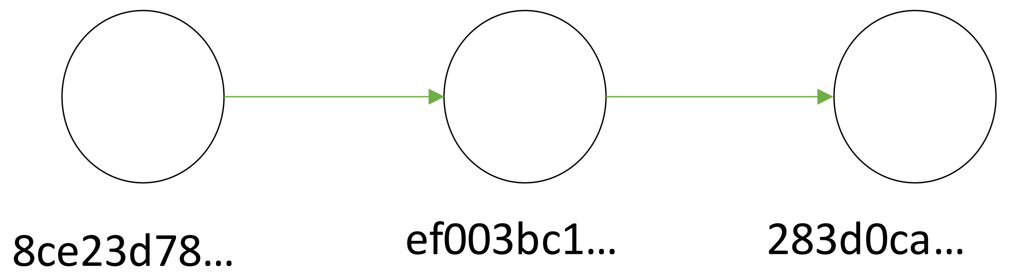
- A repository will have one main / master copy of the files and may have multiple branches for exploratory work
    - A branch is a duplication of files to allow for independent development

Content

- What goes in?
  - Any source element that is important to the project
    - Source code
    - Documentation files
    - Images
    - Test plans
    - …
- What doesn’t go in?
  - Any file that is derived from something already in the repository
    - Java .class files, which are compiled versions of the .java files
    - PDF files where you have the .docx file in the repository
    - …

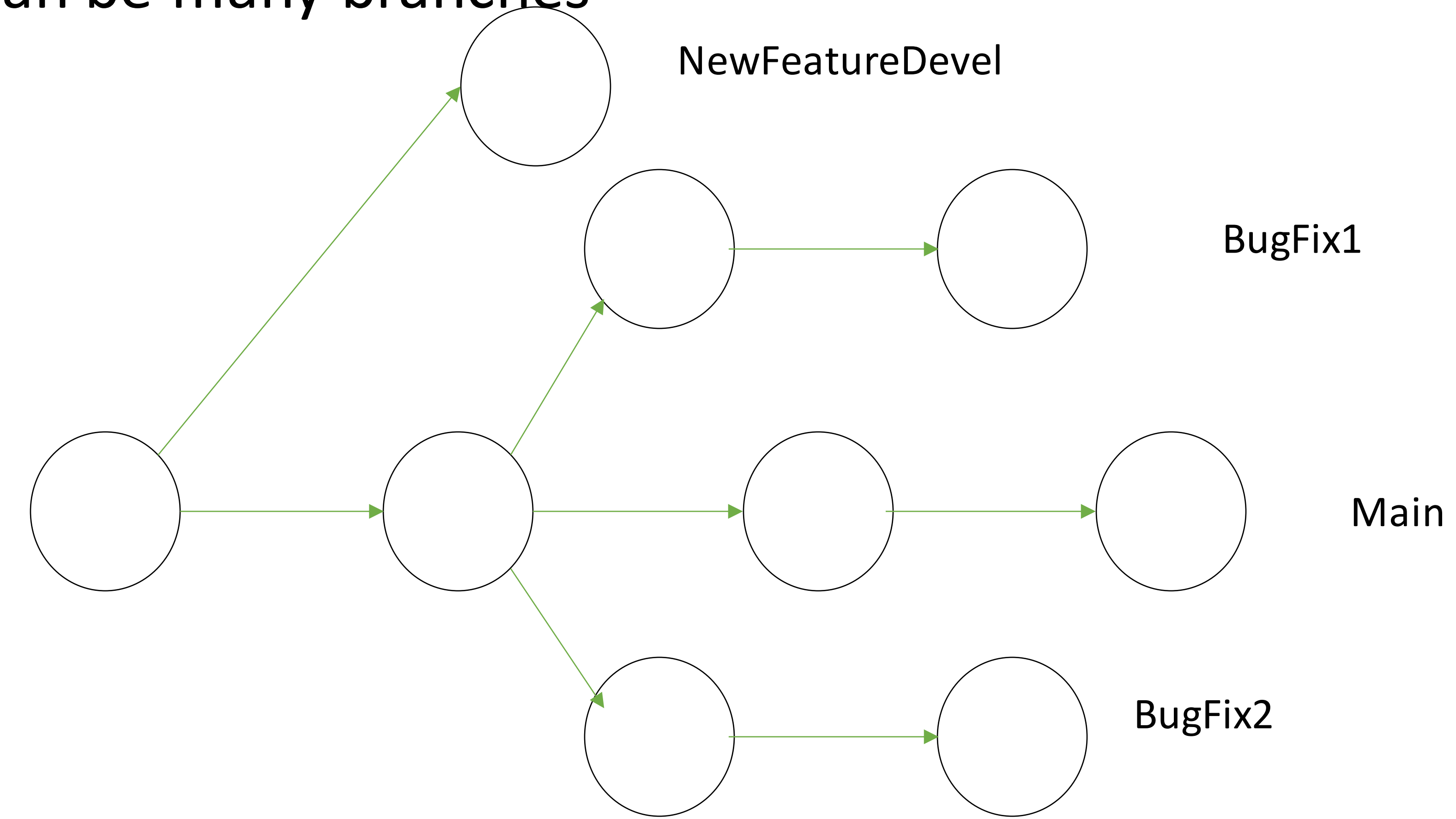
GIT mental model

- The repository consists of checkpoints of the files.
  - You decide on when the work you are doing makes up a checkpoint.
- Each checkpoint is identified by a hash value
  - Can typically use just the first 8 characters of the hash value



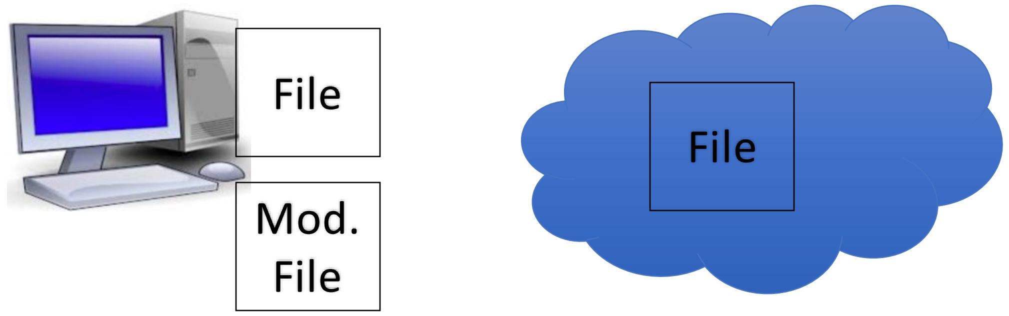
GIT mental model

- You create branch points from any of the checkpoints
- There can be many branches



Working on files

- GitHub / GitLab stores a repository on a server
- You have 3 levels of files available to you
  - The copy of the file on the server
  - A local copy known to be the latest checkpoint
  - A working copy that you are modifying
- You need to tell Git when to move your working copy to the local checkpoint (commit) and to the server (push)



Common tasks

- Connect to a server’s copy of the files
- Look at the state of your files relative to the repository
- Send your changes to the repository
- Get someone else’s changes from the repository
- Create and work in a branch

Connect to the server

- Get a copy with

  git clone <repository address>

  Make a local directory into a local Git repository

  git init

State of the repository

- git status

  Report on the files being tracked in the repository

  git log

  git log --oneline

  Report on the different checkpoints in the repository

Make changes

- git add <filename>
  Add a filename to the changes to send to the repository

  git commit –m “message”
  Move your “add”ed files to the local checkpoint

  git push origin <branchname>
  Move your local checkpoint to the server

  git pull
  Retrieve all changes in the server’s checkpoint

Not all goes into the repository

- Create a file called .gitignore
- Include patterns of filenames that you do not want to include into the repository
  - Eg .class files, .pdf files
- Can exclude some files from the patterns
  - Eg. .pdf ! Assignment.pdf
    Will exclude all PDF files except Assignment.pdf

Compare changes

- git diff <version1> <version2> <filename>

  Compare the contents of two versions of the files.

Remove files

- git rm <filename>
  git rm –r <directoryname>

  Remove a file or directory from the repository

Branches

- git checkout –b <newBranchName>
  Create a new branch

  git checkout <branchName>
  Make “branchName” your current working branch

  git branch
  List all the branches that exist

  git branch –d <branchName>
  Delete the given branch

  git merge <brancName>
  Merge the changes into the current branch

Going back in time

- **git revert <hashValue>**
  - Make the checkpoint with the given hash value the current “most recent” copy of files. Keeps all the previous file changes. This creates a new commit, undoing a previous commit.
  - A -> B -> C (head)
  - A -> B -> C -> B* (head)

- **git reset --hard <hashValue>**
- **git reset --soft <hashValue>**
  - Like revert, but removes all previous file changes. Hard will reset your local files while soft will keep the latest version in your local files.
  - A -> B -> C (head)
  - A -> B (head)

Command summary

clone

add
commit –m “..”
push origin master
pull

rm
diff

status
log

checkout –b …

checkout …

branch

branch –d …

merge …

revert <hash> …