1 Terminology

- **repository** (repo): a database holding different versions (history) of files under version control
- **working directory**: holds files, which are worked on
- **commit**: a snapshot of the files under version control
- **to commit**: to save a snapshot of the current state of files to the repository
- **stage** (index): composing area for commits; committing records the modifications in stage to a commit
- **to stage**: to add modified files from working directory to stage
- **branch**: named lineage of commits; multiple branches allow for isolated, parallel lines of development
- **to branch**: to diverge a new branch from a parent branch
- **conflict**: overlapping modifications in parallel branches
- **merge**: a commit, which handles possible conflicts between branches and joins the branches
- **to merge**: to bring the contents of another branch into the current branch
- **fast-forward merge**: merging of branches, where the merged branch is a linear continuation of the parent branch; commits in the merged branch are included in the parent branch as such
- **remote repository**: a copy of the repository, which can be accessed over network or filesystem
- **to clone**: make an independent copy of a repository
- **to push**: to sync repos by sending commits from the local repo to a remote repo
- **to pull**: to sync repos by getting commits from a remote repo
- **tag**: a nickname usually for a commit, e.g. v3.4.2
- **bare repository**: a repository without working files

2 Anatomy of Git repository

3 Merging strategies
4 Usage reference

All commands should be prefixed with git, e.g., git help.

- help
  help # lists common commands
  help config # help on a particular command (here config)

4.1 Configuration

- set name and email
  config --global user.name Jane Smith
  config --global user.email jane.smith@aalto.fi

- set editor and coloring of output
  config --global core.editor nano # or vim, emacs etc.
  config --global color.ui true

Without --global option the commands affect only the repository in the current directory.

4.2 Creating or getting a repository

- create a new repository to current directory
  init
  Add --shared option to create a repository, which handles read and write permissions for unix group.

- clone an existing repository
  clone /Local/path/repo.git
  or
  clone https://github.com/JuliaLang/julia.git jul
  or
  clone /local/path/repo.git
  diles read and write permissions for unix group.

- add
  add . # stage all files in the directory
  add readme.txt # listing multiple files works also
  add -p readme.txt # interactively select which modifications are staged; -p works also with many other commands

- rm
  rm readme.txt # removal; removes the file from filesystem

- mv
  mv readme.txt readme.md # rename

- get
  get files from stage to working directory

- add
  add -p readme.txt
  add readme.txt
  add .

- commit
  commit # commit the staged modifications
  commit -m "My commit message"
  commit -a # commit modifications in all tracked files (skips staging)

- get an older version of a file
  checkout 6dac6bf -- readme.txt # take from 6dac6bf

- revert
  revert 47410c4

- get files from stage to working directory
  checkout-index readme.txt

- apply the changes in a commit
  cherry-pick 7360d6a # makes a new commit with the changes in 7360d6a

- tag
  tag v1.0 # name the latest commit v1.0

4.4 Basic workflow

- stage files and modifications
  add . # stage all files in the directory
  add readme.txt # listing multiple files works also
  add -p readme.txt # interactively select which modifications are staged; -p works also with many other commands
  rm readme.txt # removal; removes the file from filesystem
  mv readme.txt readme.md # rename

- unstage
  reset readme.txt

- commit
  commit # commit the staged modifications
  commit -m "My commit message"
  commit -a # commit modifications in all tracked files (skips staging)

- get an older version of a file
  checkout 6dac6bf -- readme.txt # take from 6dac6bf

- revert
  revert 47410c4

- get files from stage to working directory
  checkout-index readme.txt

- apply the changes in a commit
  cherry-pick 7360d6a # makes a new commit with the changes in 7360d6a

- tag
  tag v1.0 # name the latest commit v1.0

4.5 Branching and merging

- list, create and delete branches
  branch -a # list
  branch dev # diverge a new branch with name dev
  branch -d dev # delete

- change to a branch
  checkout dev # change to branch dev

- who changed and when the lines of a file
  blame readme.txt

- show the message and changes of a commit
  show 6dbe052

4.6 Synchronizing with remote repositories

- list, add and remove remotes
  remote -v # list remotes
  remote add becs

- branch
  branch -a
  branch dev

- tag
  tag v1.0

- merge branches
  merge dev # merges dev branch to current branch

Note: branch name is a reference to the latest commit in the branch. HEAD is a reference to the latest commit in the current branch. FETCH_HEAD is a reference to the latest commit in a fetched remote branch (see below).

4.7 Rewriting history

Never rewrite shared history!

- amend a previous commit or change its message
  commit --amend # after, e.g., adding a forgotten file to stage

- rebase a branch to include the progress in a parent branch
  rebase master # assuming master is the parent branch

- combine, remove and edit commits
  rebase -i fe18b87 # prompt for commits after fe18b87

5 Resources

- Git homepage: git-scm.com
- Stackoverflow: stackoverflow.com/questions/tagged/git
- GitHub: github.com – popular Git hosting on web
- Bitbucket: bitbucket.org – popular Git hosting on web