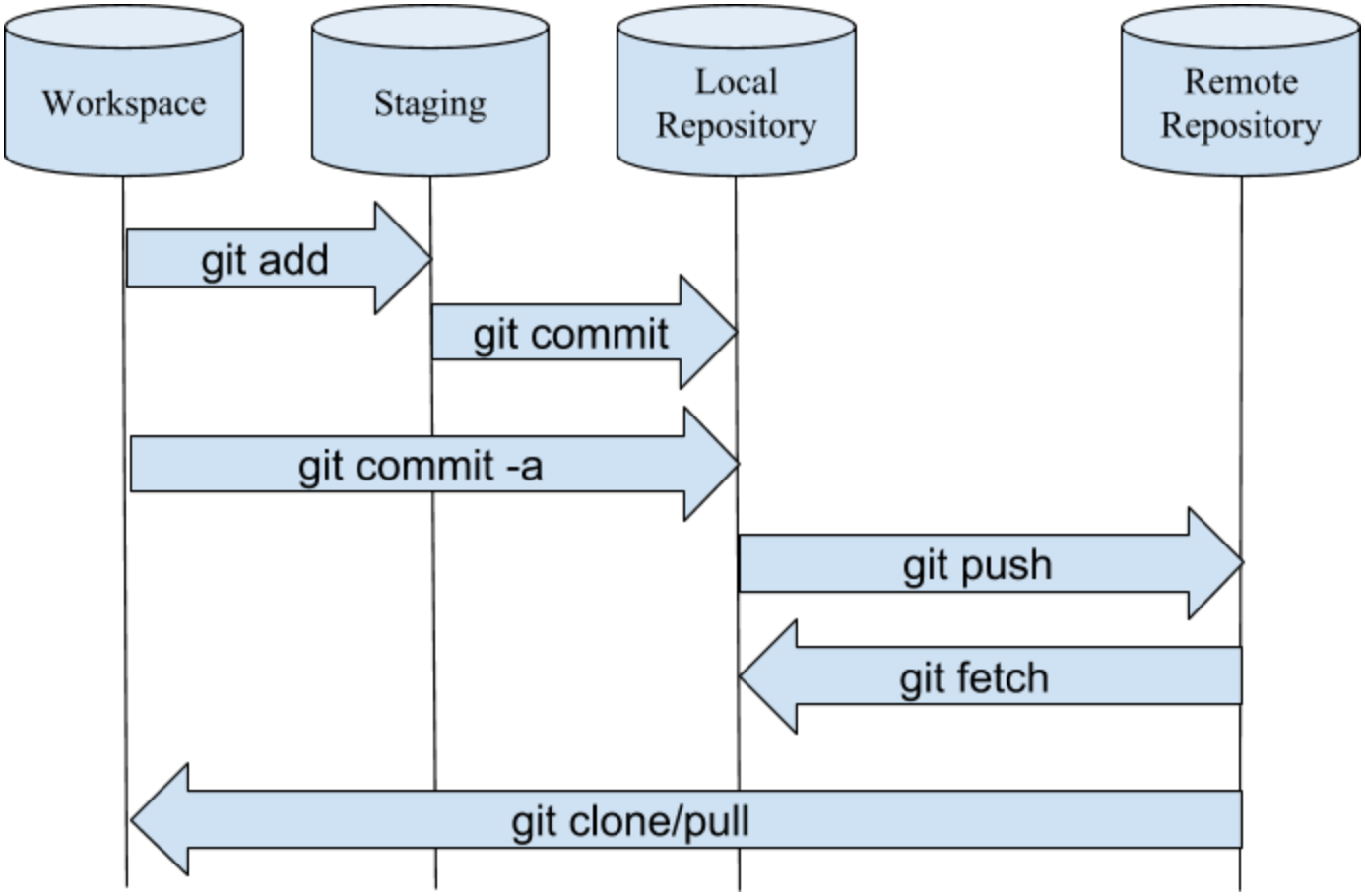
M. Mücahid Benlioğlu
Mert Koşan

Git Tutorial

Version control system that tracks the changes in files and helps group of people to work together in a project easily.

Architecture



Commands

Some commands, theirs usage and practice are retrieved from
http://guides.beanstalkapp.com/version-control/common-git-commands.html

We are also introducing some missing part of above link. We are going to share the codes and commands which we have done in the workshop as well.

git init

It creates empty Git repository. You should make this command to use other commands of git.

Usage:

```
# change directory to codebase
$ cd /file/path/to/code

# make directory a git repository
$ git init
```

Practice:

```
# change directory to codebase
$ cd /Users/computer-name/Documents/website

# make directory a git repository
$ git init
Initialized empty Git repository in /Users/computer-name/Documents/website/.git/
```

git add

Add files to staging area for Git. Files, that will be committed, should be added to staging area before. You can add all files, specific file or whole folder.

Usage:

```
$ git add <file or directory name>
```

Practice:

```
# To add all files not staged:
$ git add .

# To stage a specific file:
$ git add index.html

# To stage an entire directory:
$ git add css
```

git commit

Record the changes of the files to local repository and each commit has own unique ID. Commit message also is important to understand changes and explanation of the commits.

Usage:

```
# Adding a commit with message
$ git commit -m "Commit message in quotes"
```

Practice:

```
$ git commit -m "My first commit message"
[SecretTesting 0254c3d] My first commit message
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 homepage/index.html
```

git diff

Show the difference between the areas (commits, working directory, staging area etc.).

Usage:

```
# Shows the difference between workspace and staging
$ git diff

# Shows the difference between workspace and commit
$ git diff HEAD

# Shows the difference between staging and commit
$ git diff --cached
```

Practice:

```
# Show difference between workspace and staging area
$ git diff
diff --git a/file b/file
index 8baef1b..5f5521f 100644
--- a/file
+++ b/file
@@ -1 +1,2 @@
 abc
+def
```

git status

Returns the current state of the repository. It shows untracked files or files which are ready to commit. If there are no files like these, it shows working directory is clean.

Usage:

```
$ git status
```

Practice:

```
# Message when files have not been staged (git add)
$ git status
On branch SecretTesting
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        homepage/index.html

# Message when files have been not been committed (git commit)
$ git status
On branch SecretTesting
Your branch is up-to-date with 'origin/SecretTesting'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   homepage/index.html

# Message when all files have been staged and committed
$ git status
On branch SecretTesting
nothing to commit, working directory clean
```

git config

There are many configuration that should or may be set in Git. There is a --global flag that set/get settings of all repositories on your computer. Without a --global flag, git config applies only current repository. Two essential configurations are username and email.

Usage:

```
$ git config <setting> <command>
```

Practice:

```
# Running git config globally
$ git config --global user.email "my@emailaddress.com"
$ git config --global user.name "Brian Kerr"

# Running git config on the current repository settings
$ git config user.email "my@emailaddress.com"
$ git config user.name "Brian Kerr"
```

git branch

In Git, there can be other branches alongside of the master branch. This command shows current branch where you are working on, creates or deletes branches.

Usage:

```
# Create a new branch
$ git branch <branch_name>

# List all remote or local branches
$ git branch -a

# Delete a local branch
$ git branch -d <branch_name>

# Delete a remote branch_name
$ git push origin --delete <branch_name>
```

Practice:

```
# Create a new branch
$ git branch new_feature

# List branches
$ git branch -a
* SecretTesting
  new_feature
  remotes/origin/stable
  remotes/origin/staging
  remotes/origin/master -> origin/SecretTesting

# Delete a branch
$ git branch -d new_feature
Deleted branch new_feature (was 0254c3d).
```

git checkout

It changes to branch that you are working.

Usage:

```
# Checkout an existing branch
$ git checkout <branch_name>

# Checkout and create a new branch with that name
$ git checkout -b <new_branch>
```

Practice:

```
# Switching to branch 'new_feature'
$ git checkout new_feature
Switched to branch 'new_feature'

# Creating and switching to branch 'staging'
$ git checkout -b staging
Switched to a new branch 'staging'
```

git merge

This combines the branches into one branch.

Usage:

```
# Merge changes into current branch
$ git merge <branch_name>
```

Practice:

```
# Merge changes into current branch
$ git merge new_feature
Updating 0254c3d..4c0f37c
Fast-forward
 homepage/index.html | 297 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 1 file changed, 297 insertions(+)
 create mode 100644 homepage/index.html
```

Remote repository commands:

git remote:

It connects local repository to remote repository. A remote repository can take name that helps not to remember repository URL.

Usage:

```
# Add remote repository
$ git remote <command> <remote_name> <remote_URL>

# List named remote repositories
$ git remote -v
```

Practice:

```
# Adding a remote repository with the name of beanstalk
$ git remote add origin git@account_name.git.beanstalkapp.com:/acccount_name/repository_name.git

# List named remote repositories
$ git remote -v
origin git@account_name.git.beanstalkapp.com:/acccount_name/repository_name.git (fetch)
origin git@account_name.git.beanstalkapp.com:/acccount_name/repository_name.git (push)
```

git clone:

It creates local copy of an available remote repository.

Usage:

```
$ git clone <remote_URL>
```

Practice:

```
$ git clone git@account_name.git.beanstalkapp.com:/acccount_name/repository_name.git
Cloning into 'repository_name'...
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 5 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (5/5), 3.08 KiB | 0 bytes/s, done.
Checking connectivity... done.
```

git pull:

It gets latest version of a repository. It pulls the changes of given branch.

Usage:

```
$ git pull <remote_URL/remote_name> <branch_name>
```

Practice:

```
# Pull from named remote
$ git pull origin staging
From account_name.git.beanstalkapp.com:/account_name/repository_name
 * branch            staging    -> FETCH_HEAD
 * [new branch]      staging    -> origin/staging
Already up-to-date.

# Pull from URL (not frequently used)
$ git pull git@account_name.git.beanstalkapp.com:/acccount_name/repository_name.git staging
From account_name.git.beanstalkapp.com:/account_name/repository_name
 * branch            staging    -> FETCH_HEAD
 * [new branch]      staging    -> origin/staging
Already up-to-date.
```

git push:

It sends local changes/commits to remote repository. It needs branch name and remote name as parameters.

Usage:

```
$ git push <remote_URL/remote_name> <branch>

# Push all local branches to remote repository
$ git push –all
```

Practice:

```
# Push a specific branch to a remote with named remote
$ git push origin staging
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 734 bytes | 0 bytes/s, done.
Total 5 (delta 2), reused 0 (delta 0)
To git@account_name.git.beanstalkapp.com:/acccount_name/repository_name.git
   ad189cb..0254c3d  SecretTesting -> SecretTesting

# Push all local branches to remote repository
$ git push --all
Counting objects: 4, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 373 bytes | 0 bytes/s, done.
Total 4 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To git@account_name.git.beanstalkapp.com:/acccount_name/repository_name.git
   0d56917..948ac97  master -> master
   ad189cb..0254c3d  SecretTesting -> SecretTesting
```

git fetch:

It refreshes the local repository with remote repository. It gets new branches/existing branches with new data.

Usage:

```
# Fetch all data from remote repository to local repository
$ git fetch <remote_URL/remote_name>
```

Practice:

```
# Fetch data from remote called origin
$ git fetch origin
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 3
Unpacking objects: 100% (3/3), done.
From https://github.com/cs-workshops/git-tutorial
* [new branch] GitIntro -> origin/GitIntro
```

git log:

It shows commit history for the current repository.

Usage:

```
# Show entire git log
$ git log

# Show git log with date pameters
$ git log --<after/before/since/until>=<date>

# Show git log based on commit author
$ git log --<author>="Author Name"
```

Practice:

```
# Show entire git log
$ git log
commit 4c0f37c711623d20fc60b9cbcf393d515945952f
Author: Brian Kerr <my@emailaddress.com>
Date:   Tue Oct 25 17:46:11 2016 -0500

    Updating the wording of the homepage footer

commit 0254c3da3add4ebe9d7e1f2e76f015a209e1ef67
Author: Ashley Harpp <my@emailaddress.com>
Date:   Wed Oct 19 16:27:27 2016 -0500

    My first commit message

# Show git log with date pameters
$ git log --before="Oct 20"
commit 0254c3da3add4ebe9d7e1f2e76f015a209e1ef67
Author: Ashley Harpp <my@emailaddress.com>
Date:   Wed Oct 19 16:27:27 2016 -0500

    My first commit message

# Show git log based on commit author
$ git log --author="Brian Kerr"
commit 4c0f37c711623d20fc60b9cbcf393d515945952f
Author: Brian Kerr <my@emailaddress.com>
Date:   Tue Oct 25 17:46:11 2016 -0500

    Updating the wording of the homepage footer
```