Git

Document Update: 16.11.2017

Table of Contents

Git References

- https://git-scm.com
- https://git-scm.com/book/en/v2 (Pro Git book)
- https://github.com/

Installation

Git "config"

Setup your user name and email address:

```
$ git config --global user.name "John Doe"

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

Working with a Local Repository

Create a new Git repository

Create a new empty Git repository in the current directory:

```
$ git init
```

Create a new folder named "git-demo" with an empty Git repository:

```
$ git init git-demo
```

Git "add"

Add files to a repository:

```
$ git add xyz.txt

$ git add *.xlsx
```

Git "commit"

Commit a single file to your local repository:

```
$ git commit xyz.txt -m "Some commit text"
```

Note:

- you **must** supply a commit text
- commit is always to your local repository only

Git "status"

View the status of a repository:

```
$ git status
```

Example:

C:\Work\WebDev\repo\playground\WebVisionera>git status

On branch sandbox

nothing to commit, working tree clean

Working with a Remote Repository

Put your own local repository to a server

Create a (local) bare repository from your local repository:

```
$ git clone --bare play-it play-it.git
```

Copy the bare repository to a server:

```
$ xcopy play-it.git \\gitsrv\repo\test /e
```

Add (connect) the remote server to your repository:

```
$ git remote add origin \\gitsrv\repo\git-docs
```

Display the remote servers:

```
$ git remote -v
```

Git "push" / "pull"

Now you can "push" your changes to the server:

```
$ git push origin master
```

"Pull" from the remote server:

```
$ git pull origin master
```

Git "clone"

Anyone can now clone the repo from the (remote) server to local storage:

```
$ git clone //gitsrv/git-test/git-docs C:\work\myrepos\git-docs
```

Clone a repository from github:

```
$ git clone "https://github.com/ussebatz/angular"
```

Put your repository to GitHub

Create an empty repo on GitHub (e.g. "tour-of-heroes").

Add the repository:

```
$ git remote add origin https://github.com/ussebatz/tour-of-heroes
```

Push the local repo to GitHub:

```
$ git push -u origin master
```

Branches

Create a new branch:

```
$ git branch testing
```

Switch to a branch (in order to work on that branch):

```
$ git checkout testing
```

Create a branch and switch to that branch:

```
$ git checkout -b branch53
```

Delete a branch:

```
$ git branch -d branch53
```

View current branches:

```
$ git branch
```

Push branch "TAC102":

```
$ git push origin TAC102
```

Examples

Show available branches:

```
C:\Work\WebDev\repo\playground\WebVisionera>git branch
  master
* sandbox
```

Switch to branch 'master':

```
C:\Work\WebDev\repo\playground\WebVisionera>git checkout master
Switched to branch 'master'
Your branch and 'origin/master' have diverged,
and have 4 and 24 different commits each, respectively.
  (use "git pull" to merge the remote branch into yours)
```

Git and Visual Studio Code

Step-1: Create a new Angular project

```
$ ng new play-info
```

VS Code will create the Angular project including some required files. By default VS Code will also automatically setup a new Git repository in the new folder.

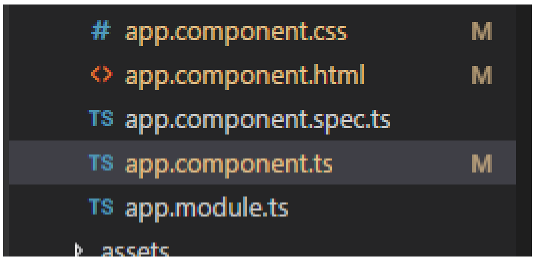
The repository will include all the required source files, but will not include generated files (e.g. "node_modules").

The files and folders that are not to be included into the Git repository are listed in ".gitignore".

You can open the new project in VS Code.

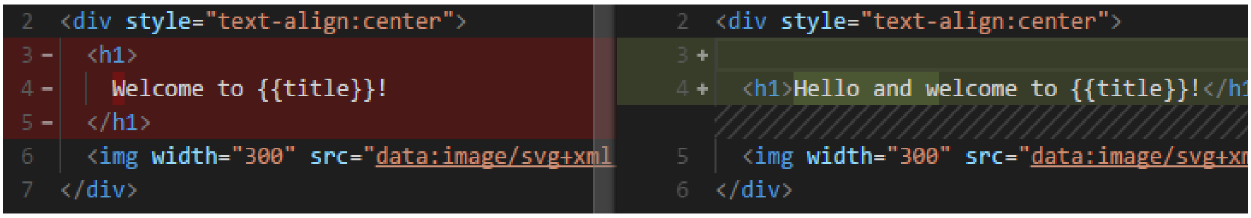
Step-2: Make changes

Make changes to some files. The changed files are indicated "M" (modified).



```
# app.component.css        M
<> app.component.html      M
TS app.component.spec.ts
TS app.component.ts        M
TS app.module.ts
 ▸ assets
```

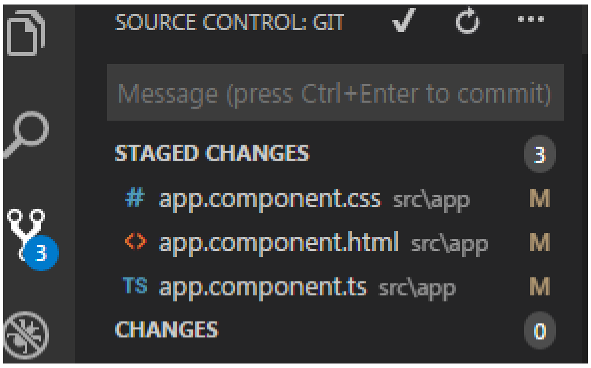
You can view the differences.



```
  <div style="text-align:center">
-   <h1>
-     Welcome to {{title}}!
-   </h1>
    <img width="300" src="data:image/svg+xml
  </div>
```

```
  <div style="text-align:center">
+
+   <h1>Hello and welcome to {{title}}!</h1
    <img width="300" src="data:image/svg+xm
  </div>
```

Step-3: Stage changes



SOURCE CONTROL: GIT ✓ ↻ ···

Message (press Ctrl+Enter to commit)

STAGED CHANGES 3
- # app.component.css src\app M
- <> app.component.html src\app M
- TS app.component.ts src\app M

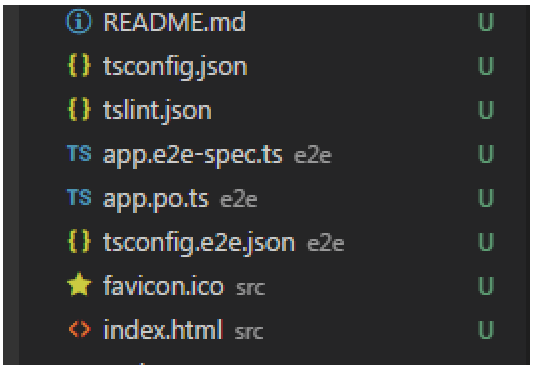
CHANGES 0

Step-4: Commit changed files

After commit there are no changed files.

Step-5: Create new files

Create new files, new files are indicated as "U" (untracked):



README.md		U
tsconfig.json		U
tslint.json		U
app.e2e-spec.ts	e2e	U
app.po.ts	e2e	U
tsconfig.e2e.json	e2e	U
favicon.ico	src	U
index.html	src	U

Stage changes, then the new file is indicated "A" (added).

Commit changes.

Ignoring files

Create a .gitignore file.

TODO

Command summary

Git Command	**Purpose**
git config –global user.name "Herbert"	Configure a user.
git config –global user.email "email@address"	Configure a user's e-mail.
git config --list	List the current configuration settings.
git init	Create a new repository in the current folder.
git init Project1	Create a new folder 'Project1' and init a git repository in that folder.
git clone https://github.com/libgit2/libgit2	Clone an existing repository.
git add file1.txt	Add a single file to the project.
git add *.cpp	Add multiple files.
git add css	Add a folder.
git rm somefile.txt	Remove a file
git log	View commit history.
git status	
"fetch"	Download from remote repository (but do not merge).
"pull"	Download and merge.
git commit -m"Some Message"	Takes all the file contents that have been staged with *git add* and records a new permanent snapshot in the database. Adds a message to the commit.