Basics

Downloading Git

- We can download Git for free at http://git-scm.com/downloads.

  Git is available for various operating systems.

Git Initiation

- The first step would initializing the folder we want Git to track its files.

```
Last login: Wed Jan  8 12:19:30 on ttys001
h-MacBook-Pro-sl-haim-3:~ haimmichael$ mkdir ourprojectfiles
h-MacBook-Pro-sl-haim-3:~ haimmichael$ cd ourprojectfiles
h-MacBook-Pro-sl-haim-3:ourprojectfiles haimmichael$ git init
Initialized empty Git repository in /Users/haimmichael/ourprojectfiles/.git/
h-MacBook-Pro-sl-haim-3:ourprojectfiles haimmichael$
```

Git Initiation

- The initiation process creates the .git directory inside the directory we chose to initialize.
- The .git directory is usually read-only and hidden in order to protect it from accidental deletion.
- Git uses .git directory for saving our files history and their changes.

Git Configuration

- We should configure Git before we start using it. The minimum required configuration is setting the user.name and the user.email in order to allow Git to associate each change with a specific user.

Git Configuration

- We use the word `config` with `git` in order to set up the configuration. In order to set up a global value we should add `--global` parameter. In order to set up a local value we should add the `--local` parameter. Global configuration will apply for all repositories.

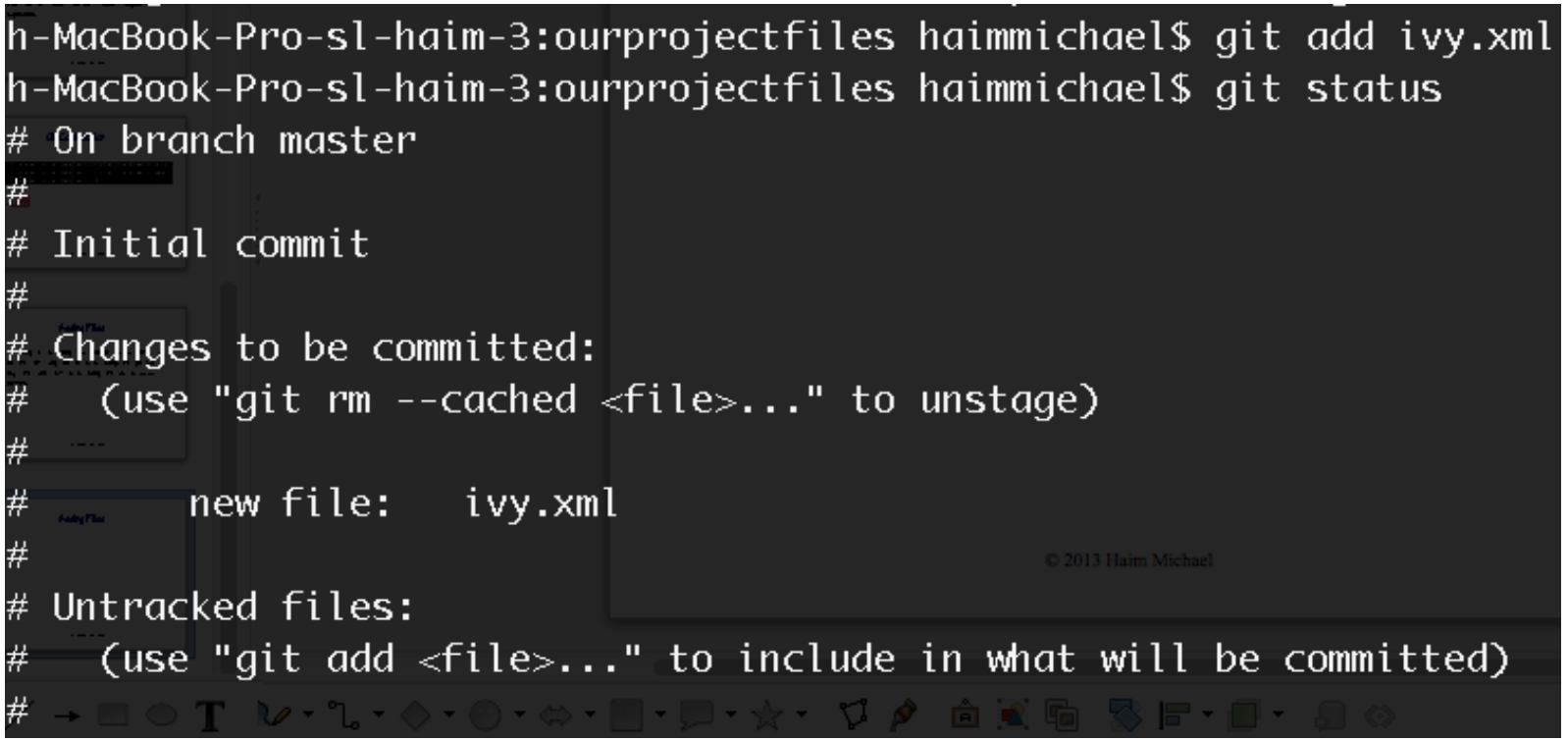
Git Configuration

```
h-MacBook-Pro-sl-haim-3:ourprojectfiles haimmichael$ git config --global user.name "haim mich
ael"
h-MacBook-Pro-sl-haim-3:ourprojectfiles haimmichael$ git config --global user.email "haim.mic
hael@gmail.com"
h-MacBook-Pro-sl-haim-3:ourprojectfiles haimmichael$
```

Adding Files

- The repository is the directory we configured to be tracked by Git. We can easily add files to the repository. We just need to copy and paste them to the repository and use the `git add` command.

Adding Files



The `git status` Command

- Using `git status` we can get a detailed list of all changes that took place in our repository.

The `git status` Command

```
h-MacBook-Pro-sl-haim-3:ourprojectfiles haimmichael$ git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#       new file:   ivy.xml
#       new file:   tamarhaim.jpg
#
# Changes not staged for commit:
#   (use "git add/rm <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       deleted:    ivy.xml
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       .DS_Store
```

Committing The Added Files

- When we add a file to the repository and make a commit, Git registers the new file. Any further commits will be tracked as changes to the very same file.
- In order to commit the added files we should use the `git commit` command.

Committing The Added Files

```
h-MacBook-Pro-sl-haim-3:ourprojectfiles haimmichael$ git commit -m "my first com
mit.. part of a small demo"
[master (root-commit) f5d4bf3] my first commit.. part of a small demo
 5 files changed, 44 insertions(+)
 create mode 100644 ivy.xml
 create mode 100644 lifemichael_200x200.png
 create mode 100644 lifemichael_200x200_2.png
 create mode 100644 lifemichael_200x200_3.png
 create mode 100644 lifemichael_for_hit_blog.png
```

The `git log` Command

- Using the `git log` command we can get a list of all commits. Each commit is associated with an id, user.name and user.email.

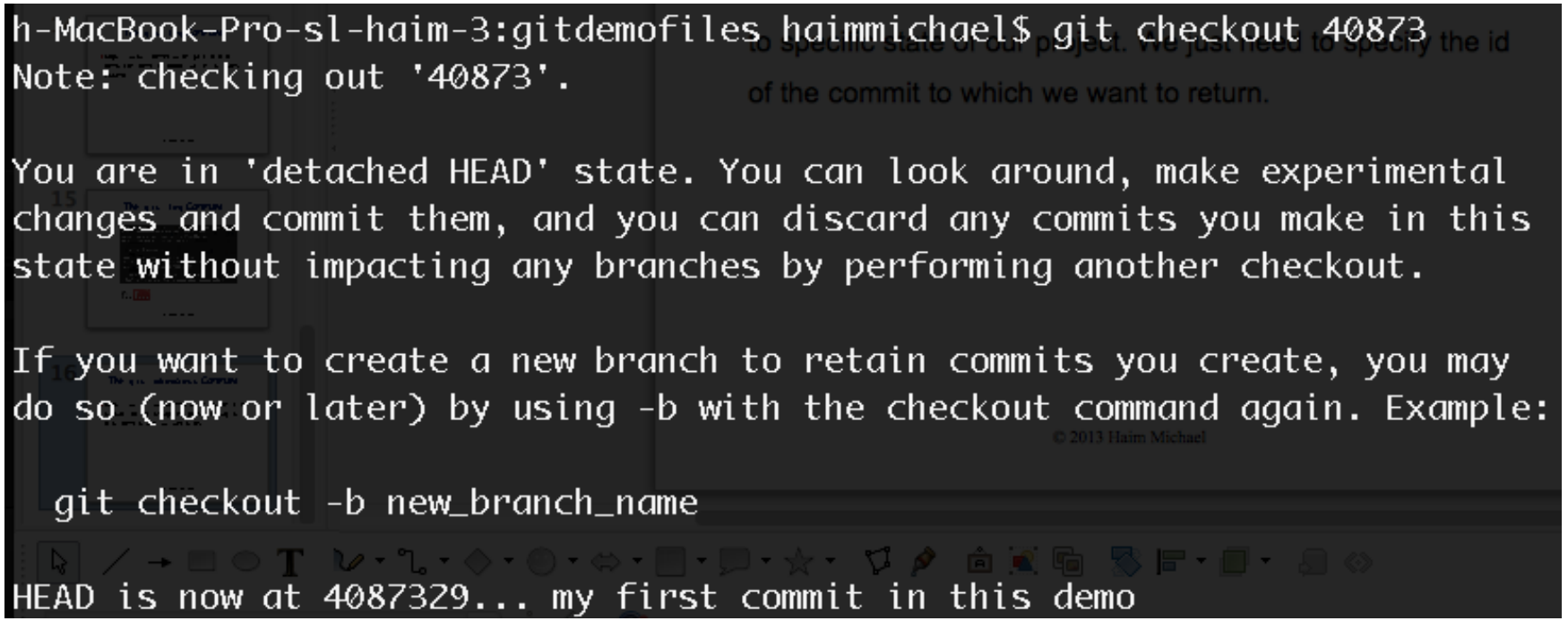
The `git log` Command

```
h-MacBook-Pro-sl-haim-3:ourprojectfiles haimmichael$ git log
commit 622797dadc28c1b5c1b534c938d4184999e1d723
Author: haim michael <haim.michael@gmail.com>
Date:   Wed Jan 8 16:11:18 2014 +0200

    my third commit...

commit f5d4bf33d441affec49c0b5eaf79d93afc93fc99
Author: haim michael <haim.michael@gmail.com>
Date:   Wed Jan 8 16:00:14 2014 +0200

    my first commit.. part of a small demo
h-MacBook-Pro-sl-haim-3:ourprojectfiles haimmichael$
```

The `git checkout` Command

- Using the `git checkout` command we can go back in time to specific state of our project. We just need to specify the id of the commit to which we want to return.
- When we check back to a previous commit we are hanging in the air. Any change to our files now will be lost once we go back to the master. We go back to the master using the git commit master command.

The `git checkout` Command



The `git checkout master` Command

- Using the `git checkout master` command we can go back to the master, the latest version.

The `git checkout master` Command

```
h-MacBook-Pro-sl-haim-3:thenewrepository haimmichael$ git checkout master
Previous HEAD position was ec55ac5... the first commit
Switched to branch 'master'
h-MacBook-Pro-sl-haim-3:thenewrepository haimmichael$
```

The `git reset` Command

- Resetting is a permanent travel back in time. There are three types of reset: soft, hard and mixed. Ignoring all changes after a specific commit can be achieved using hard reset.

The `git reset` Command

```
h-MacBook-Pro-sl-haim-3:thenewrepository haimmichael$ git reset --hard ec55a
HEAD is now at ec55ac5 the first commit
h-MacBook-Pro-sl-haim-3:thenewrepository haimmichael$
```

The `git help` Command

- Using the git help command we can get detailed help. We can get a detailed list of all the commands and a detailed help for a specific one.

The `git help` Command

```
h-MacBook-Pro-sl-haim-3:thenewrepository haimmichael$ git help
usage: git [--version] [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p|--paginate|--no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           [-c name=value] [--help]
           <command> [<args>]

The most commonly used git commands are:
   add         Add file contents to the index
   bisect      Find by binary search the change that introduced a bug
   branch      List, create, or delete branches
   checkout    Checkout a branch or paths to the working tree
   clone       Clone a repository into a new directory
   commit      Record changes to the repository
   diff        Show changes between commits, commit and working tree, etc
   fetch       Download objects and refs from another repository
   grep        Print lines matching a pattern
   init        Create an empty git repository or reinitialize an existing one
   log         Show commit logs
   merge       Join two or more development histories together
   mv          Move or rename a file, a directory, or a symlink
   pull        Fetch from and merge with another repository or a local branch
```