# Java Programming Academy

Working with Git. Linux fundamentals

## About me

**Trayan Iliev**

– CEO of **IPT – Intellectual Products & Technologies**
http://www.iproduct.org

– Oracle® certified programmer 15+ Y

– end-to-end reactive fullstack apps with Java, ES6+, TypeScript, Angular, React and Vue.js

– 12+ years IT trainer: Spring, Java EE, Node.js, Express, GraphQL, SOA, REST, DDD & Reactive Microservices

– Voxxed Days, jPrime, Java2Days, jProfessionals, BGOUG, BGJUG, DEV.BG speaker

– Organizer  RoboLearn hackathons and  IoT enthusiast

## Course Schedule

- Block 1: 9:00 – 11:00
- Pause: 11:00 – 10:15
- Block 2: 11:15 – 13:15

## Where to Find The Code and Materials?

Java Academy projects and  examples are available @GitHub:

https://github.com/iproduct/java-fundamentals-2022.git

## Agenda for This Session

- Basic version control with Git
- Linux fundamentals - basic work with terminal

# Git

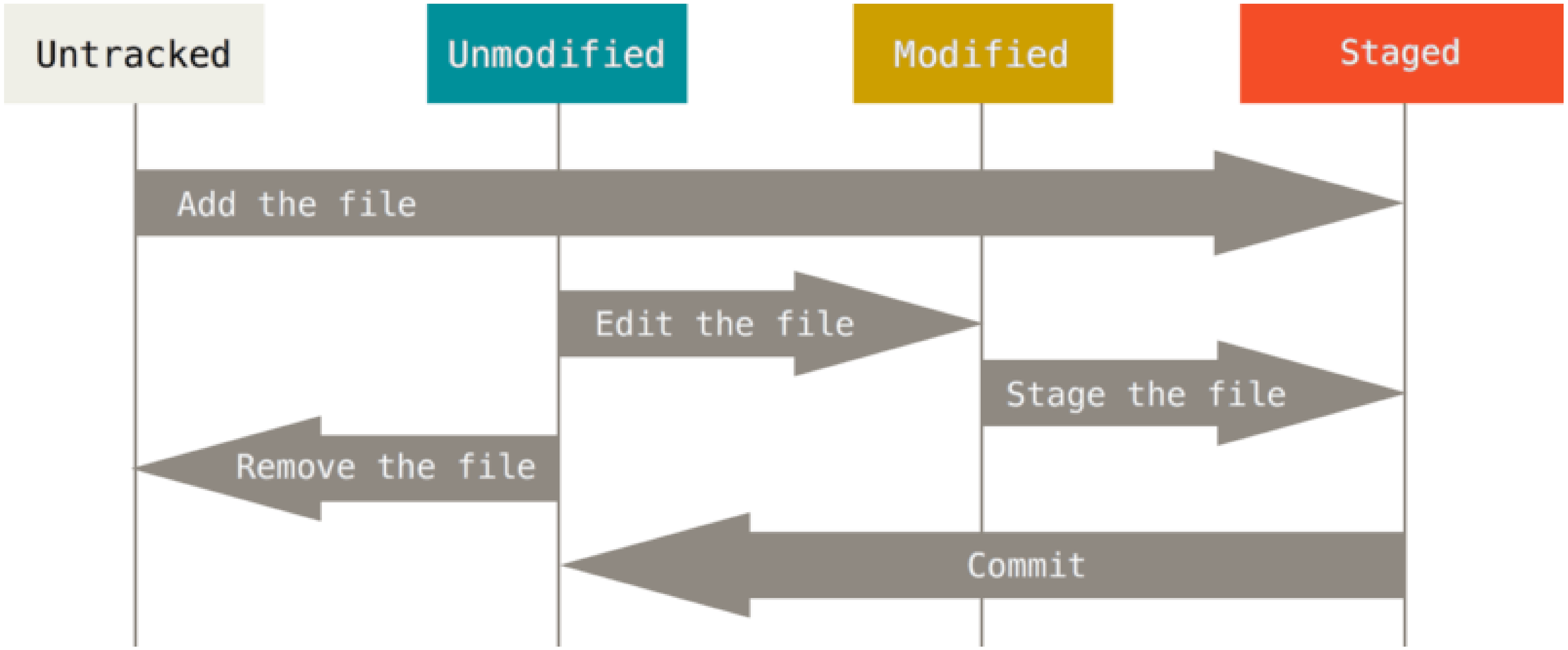
Materials from: https://git-scm.com/book/en/v2

License: Creative Commons Attribution Non Commercial Share Alike 3.0 license

## Social Coding using Git

- Version control systems and collaborative coding: CVS, SVN, Git
- Version control system – allows saving the code changes in a structured and manageable way, with ability to recover previous code state (rollback), experiments (branches), and changes synchronization (merge)
- A distinctive feature of Git is that the changes are kept locally in a form of momentary pictures (snapshots), instead of saving the list of changes – allows fast operations.
- Three stages: **Modified** → **Staged** → **Committed**

# Social Coding using Git



## Main Git Commands (1)

- Configuring Git

```
$ git config --global user.name "John Smith"
$ git config --global user.email jsmith@company.com
```

- Help information for a command

```
$ git help <command_verb>
```

- Creating new repository in an existing directory

```
$ git init
```

- Local cloning of a git repository

```
$ git clone <repository_url> [<local_folder>]
```

## Main Git Commands (2)

- Adding new files – Staging и Commit

```
$ git add *.java
$ git add README.txt
$ git commit -m ”initial commit of MyProject”
```

- Information about the status of the files in the project

```
$ git status
```

- Showing changes in the files

```
$ git diff
```

- Ignoring files – file **.gitignore**

```
$ cat .gitignore
```

## Main Git Commands (3)

- Removing files

```
$ git rm README.txt
$ git commit -m ”removing README file from project”
```

- Renaming files

```
$ git mv README.txt README
```

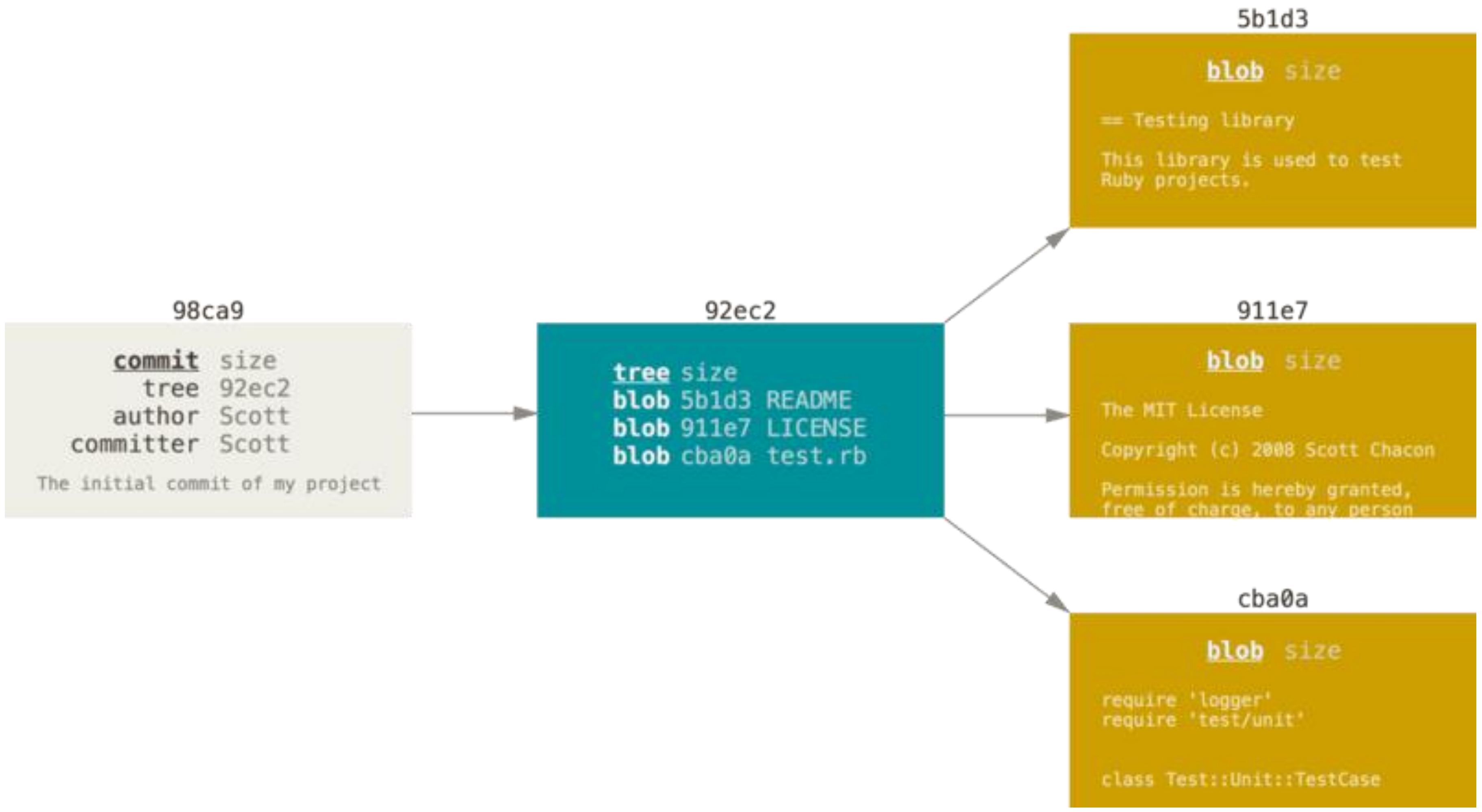
- For more information:

http://git-scm.com/book/en/Git-Basics-Recording-Changes-to-the-Repository

- Example Git project:

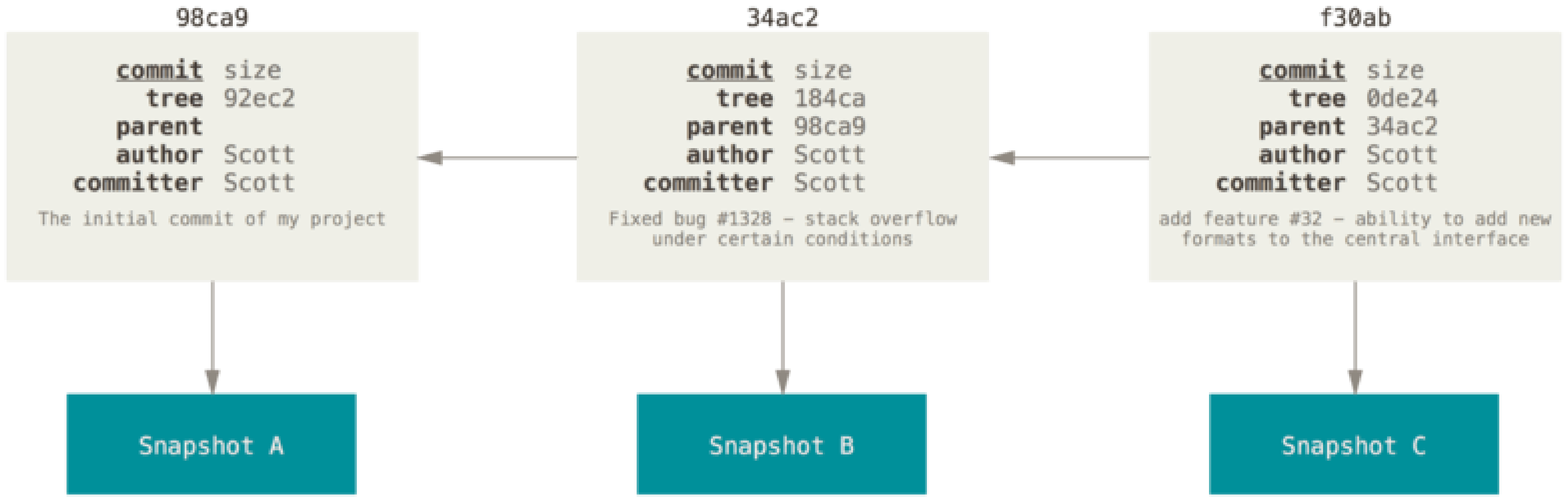
https://github.com/iproduct/java-fundamentals-2022.git

## Git Blobs

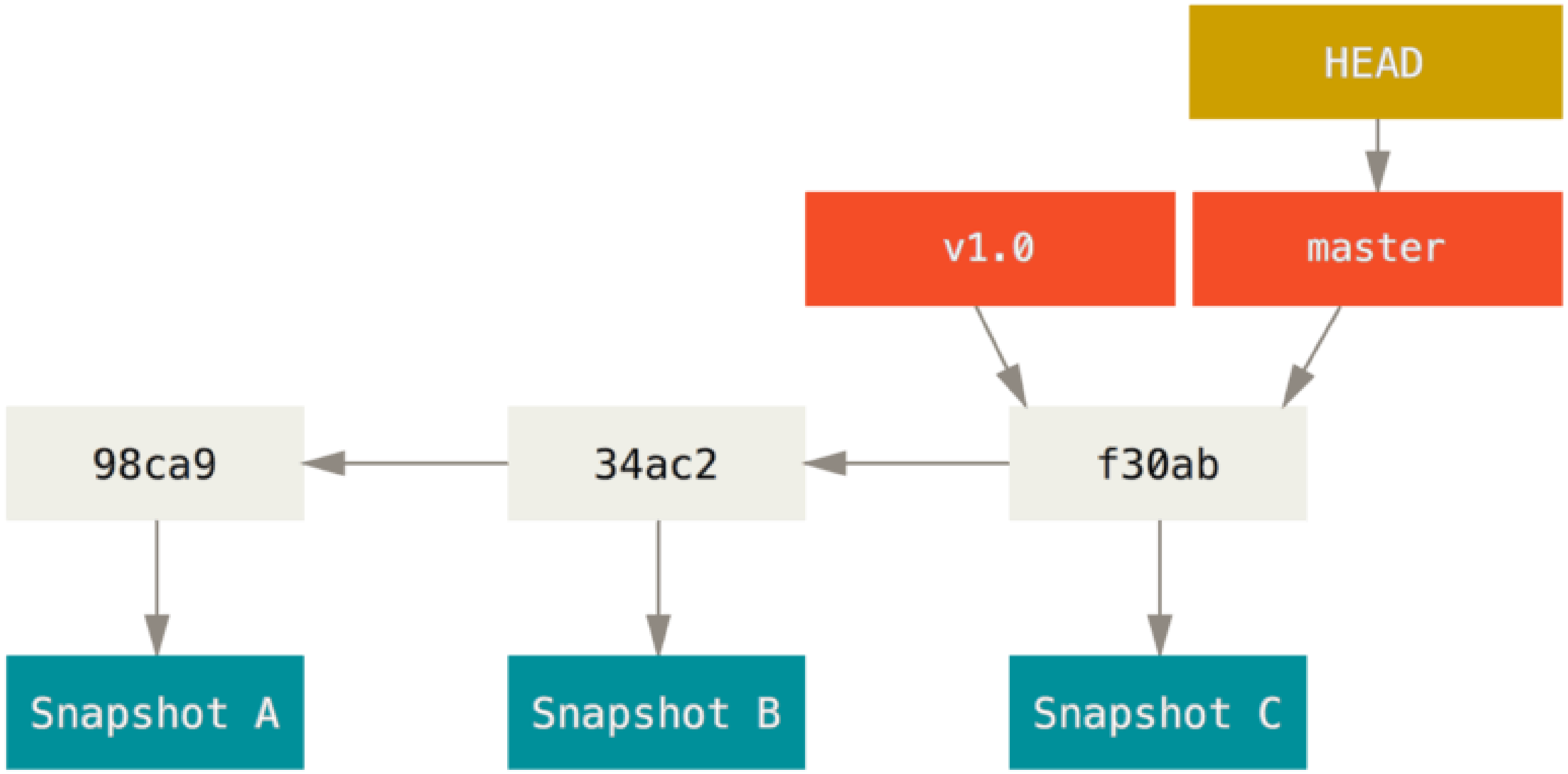


98ca9
commit size
tree 92ec2
author Scott
committer Scott
The initial commit of my project
92ec2
tree size
blob 5b1d3 README
blob 911e7 LICENSE
blob cba0a test.rb
5b1d3
blob size
== Testing library
This library is used to test Ruby projects.
911e7
blob size
The MIT License
Copyright (c) 2008 Scott Chacon
Permission is hereby granted, free of charge, to any person
cba0a
blob size
require 'logger'
require 'test/unit'
class Test::Unit::TestCase

## Git Commits

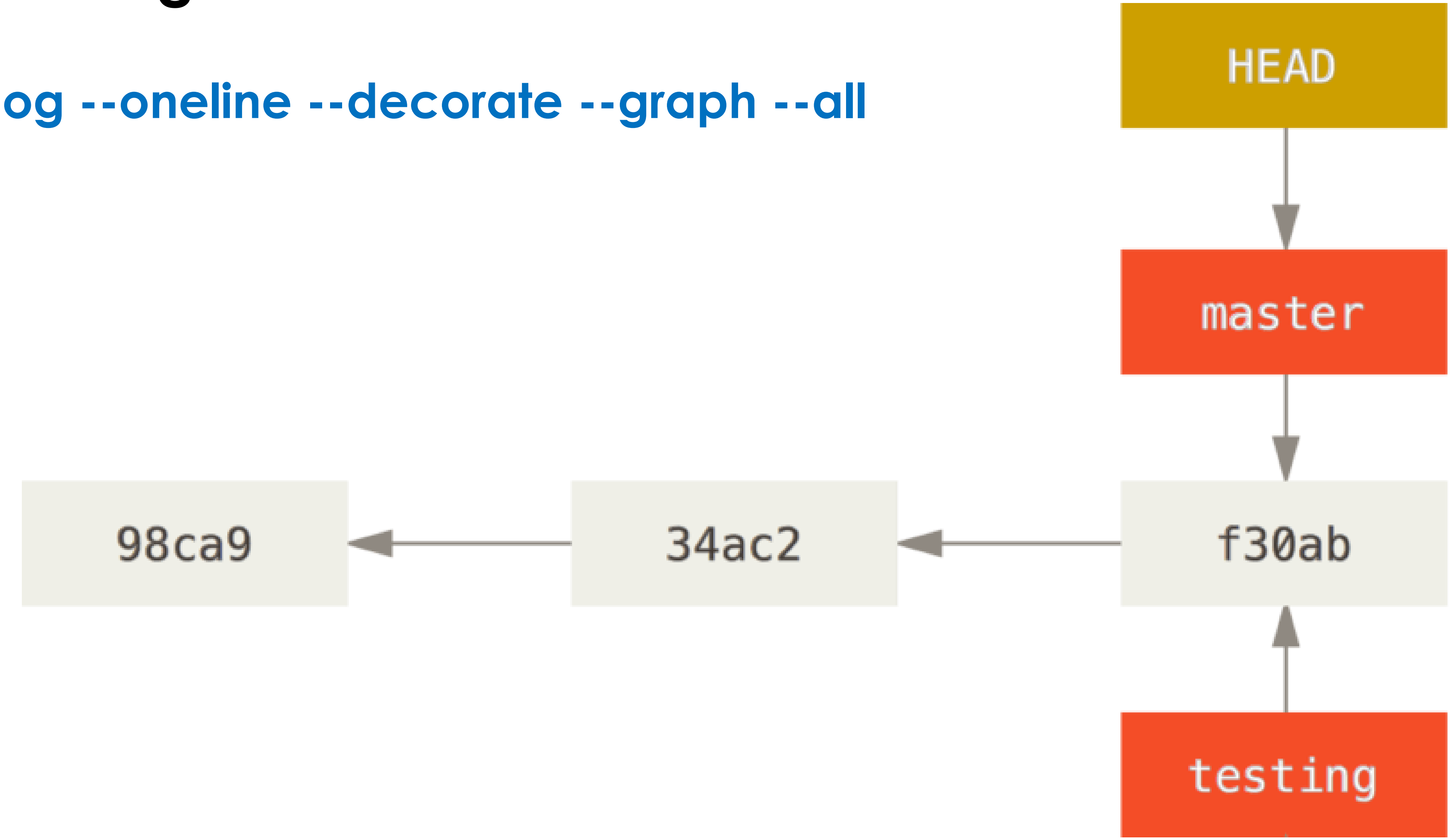

98ca9
commit size
tree 92ec2
parent
author Scott
committer Scott
The initial commit of my project
34ac2
commit size
tree 184ca
parent 98ca9
author Scott
committer Scott
Fixed bug #1328 - stack overflow under certain conditions
f30ab
commit size
tree 0de24
parent 34ac2
author Scott
committer Scott
add feature #32 - ability to add new formats to the central interface
Snapshot A
Snapshot B
Snapshot C

## Head and Branches

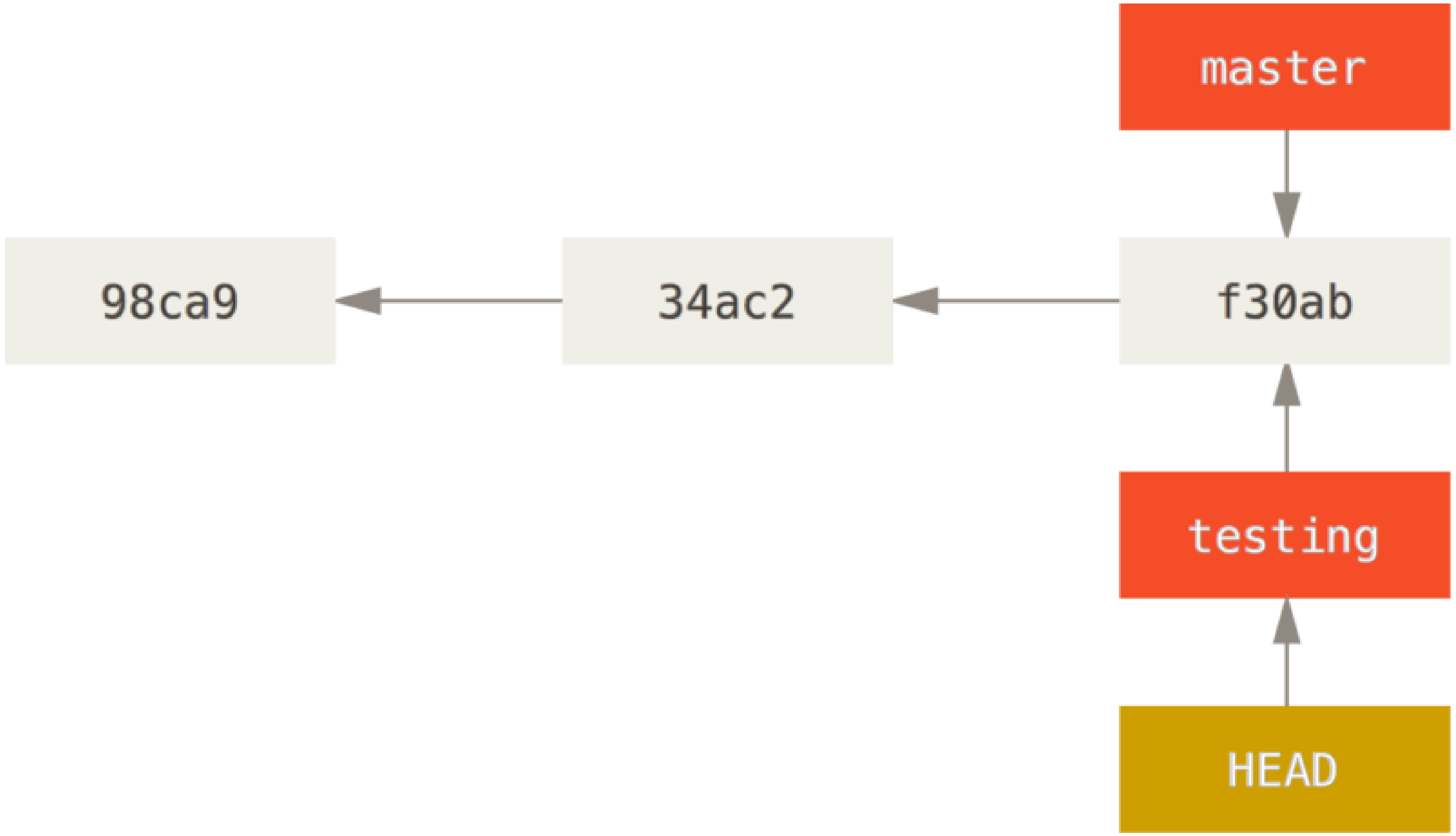


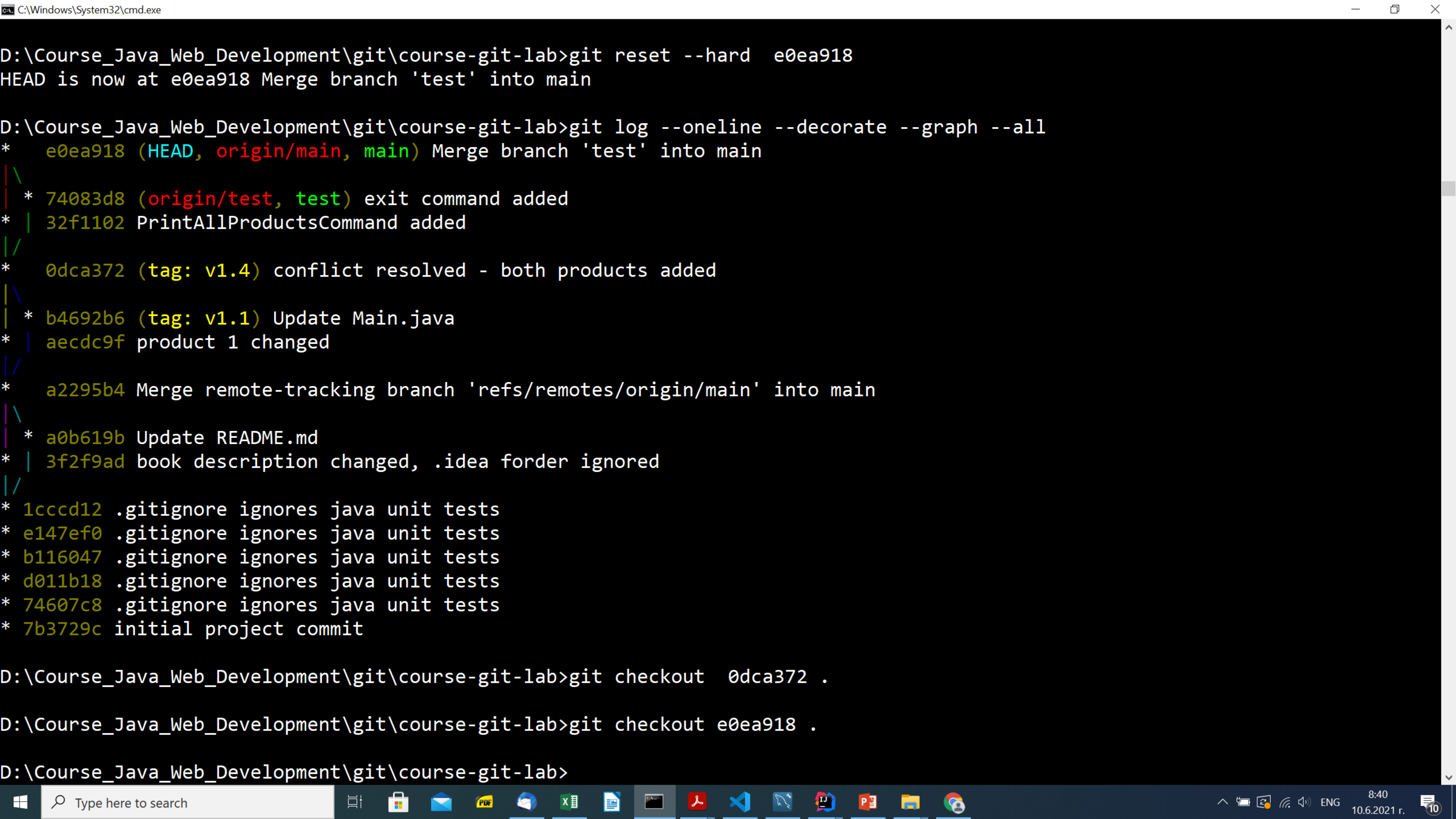
## Branching

git log --oneline --decorate --graph --all



## Switching Branches - https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell





C:\Windows\System32\cmd.exe

```
D:\Course_Java_Web_Development\git\course-git-lab>git reset --hard  e0ea918
HEAD is now at e0ea918 Merge branch 'test' into main

D:\Course_Java_Web_Development\git\course-git-lab>git log --oneline --decorate --graph --all
*   e0ea918 (HEAD, origin/main, main) Merge branch 'test' into main
|\
| * 74083d8 (origin/test, test) exit command added
* | 32f1102 PrintAllProductsCommand added
|/
*   0dca372 (tag: v1.4) conflict resolved - both products added
|\
| * b4692b6 (tag: v1.1) Update Main.java
* | aecdc9f product 1 changed
|/
*   a2295b4 Merge remote-tracking branch 'refs/remotes/origin/main' into main
|\
| * a0b619b Update README.md
* | 3f2f9ad book description changed, .idea forder ignored
|/
* 1cccd12 .gitignore ignores java unit tests
* e147ef0 .gitignore ignores java unit tests
* b116047 .gitignore ignores java unit tests
* d011b18 .gitignore ignores java unit tests
* 74607c8 .gitignore ignores java unit tests
* 7b3729c initial project commit

D:\Course_Java_Web_Development\git\course-git-lab>git checkout  0dca372 .

D:\Course_Java_Web_Development\git\course-git-lab>git checkout e0ea918 .

D:\Course_Java_Web_Development\git\course-git-lab>
```

## Resources

- Pro Git book – https://git-scm.com/book/en/v2

## Introduction to Linux

## Why Linux

- Linux is a powerful operating system.
- Many web sites use Linux as the operating system
- Tolerant of a range of hardware platforms without special configuration.
- Free platform
- Flexible and reliable
- Easier to access low-level interfaces
- Good forensic qualities

## Linux Statistics [https://writersblocklive.com/blog/linux-statistics/]

- 54.2% of the most powerful supercomputers operated on Linux in 2020.
- 90% of public cloud workloads are run on Linux.
- Android constitutes 71.93% of the operating system market share.
- Linux makes up only 1.30% of the desktop and laptop operating system market share.
- According to 83.1% of professional developers, Linux is the most loved platform.
- 59% of Ubuntu users prefer the English language.
- In 2021, the Linux kernel counts 27.8 million lines of code.
- Linux games on Steam account for 50,361.

## Recommended Linux Reading

- There are many good books on system administration.

- Recommended : UNIX SYSTEM ADMINISTRATION HANDBOOK: Third Edition – EVI NEMETH et all Prentice Hall, ISBN 0-13-020601-6

## Linux Flavours

- There are many flavours of Linux.
- There are many Linux distributions including:
  - Fedora
  - Redhat
  - Novell SUSE
  - Gentoo
- Different Linux distributions have their strengths:
  - Redhat/Fedora is the market leader for the Server Market
  - Ubuntu/Debian is a strong contender for the desktop market.
  - CAINE (Computer Aided INvestigative Environment) is an Italian GNU/Linux live distribution created as a Digital Forensics project - uses Ubuntu.

## Why A Command Prompt?

- Almost any Linux distribution has a graphical interface (GUI).

PROS:

- It is faster, easier, and more powerful to use commands at a command prompt to configure a server.

CONS:

- Command interface does mean a steep learning curve.
- Editing in the console is not so convenient

## Command Line Text Editors

Editing in the console is not so convenient, but there are editors working in console that provide mouse handling and menus – e.g.:

- Vim - extensively configurable, cross-platform, and a highly efficient text editor.
- GNU Emacs - undoubtedly one of the oldest and versatile text editor out there. In case you didn't know, it was created by GNU Project founder Richard Stallman
- Nano - when it comes to simplicity, Nano is the one. Unlike Vim or Emacs, it is suitable for beginners to get used to quickly.
- ne – The Nice Editor - when compared to the classic and popular text editors, the nice editor is a good alternative which tries to offer advanced functionalities and making it easier to use them.
- Tilde - Tilde is a terminal-based text editor tailored for users who are normally used to GUI applications.

## Telnet in the virtual machines

- Telnet is quite clever and usually no matter what OS and keyboard you have things just seem to “work”.
- Sometimes however telnet gets confused.
- If you ever have a problem where cursor keys stop working, or your editor corrupts the screen try these magic commands (you don’t type the “>”):

```
> export TERM=vt100

> tset
```

## Useful commands:

- ls
- cat
- cal
- date
- pwd
- more
- cd

- mkdir
- cp
- mv
- rm
- rmdir
- man

## The Basics

- Before your machine operates it must BOOT.
- As it boots things are started up.
- Only when the boot process completes will the system be fully operational.
- When you are finished, a machine can be shutdown or halted.
  - Shutdown – does it nicely and cleanly
  - HALT – pulls the power out the back.

## The PROMPT

- Once you log into your machine, you are at the prompt. Here you can perform your commands.
- Everything on linux is either a file or a directory.
- A file which is executed becomes a process.
- Processes can be seen as files too.
- Devices, such as scanners and hard drives are also files.

## > ls /

```
bin   dev  home  lost+found  mnt   root  selinux
tmp  var boot  etc  lib   misc proc  sbin
sys  usr
```

- Directories use / in linux (like Windows uses \).
- No volumes in linux (like C: or A: )
- / is called the root directory.
- ls splits the files either by line or in this case by tabs.

## Directories

- /bin : This contains commands a user can run, like ‘ls’, but which might be needed during boot.
- /dev : This contains devices, like the mouse.
- /home : This is where users store their files.
- /tmp : Temporary storage for users and the system
- /var : System files which can change.
- /etc : System config files which don’t change
- /lib : Where all the system libraries live
- /proc : Files which represent the running system (like processes).
- /sbin : Commands which only an administrator would want.
- /usr : Commands which are never needed during bootup.

## > cal

```
Su Mo Tu We Th Fr Sa
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31
```

## Redirection

- Ending a command with “>” - its output goes to a file.
- Ending a command with “<“ - its input comes from a file.

```
$ ls
a
$ cal > b
$ ls
a  b
$ cat b
Su Mo Tu We Th Fr Sa
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31
```

## Prompts

- When explaining commands, we usually put a prompt character before it to make it clear that the command has to be typed.
- You can set the prompt to anything, but the prompts like $ or > are common.
- Don’t type the first > or $ you see:

```
$ ls
$ cal
> ls
> cal
```

## Parameters

- Some commands change behaviours with different parameters.
- If a parameter relates to a file, then it is called a “parameter”.
- However, if the parameter changes the behavour of the program, it is instead called an “option” or “flag”.

## Flags

```
$ cal
    August 2008
Su Mo Tu We Th Fr Sa
                1  2
 3  4  5  6  7  8  9
...

$ cal -m
    August 2008
Mo Tu We Th Fr Sa Su
             1  2  3
 4  5  6  7  8  9 10
...
```

## Man pages

- If you don’t know what options or flags are possible for a command, use “man”
- For instance, to find out what flags cal uses, do:

```
$ man cal
```

- To get out of man, press “q”. Space shows you more of the information.

## Man -k

- You can keyword search for commands
- For instance, what commands show a calendar?

```
$ man -k calendar
cal             (1)  - displays a calendar
cal             (1p) - print a calendar
difftime        (3p) - compute the difference…
```

## Directories

```
$ ls
a  b
$ mkdir d1
$ ls
a  b  d1
$ cd d1
$ pwd
/home/demo/d1
```

```
$ pwd
/home/demo/d1
$ cd ..
$ pwd
/home/demo/
$ ls
a  b  d1
$ rmdir d1
$ ls
a  b
```

## Directory characters

- Absolute location (Starts with “/”)

```
cat /home/demo/z1

cat ~demo/z1
```

- Relative location (where z2 is a directory)

```
cd /home

cat demo/u1

cd /home/demo/u2

cat ../z1
```

## Wildcards

- Parameters which match filenames don’t have to be complete. You can pattern match with the characters “?” for a single character and “*” for a number of characters.

```
$ ls
aaa  aab  abb
$ ls aa?
aaa  aab
$ ls a*
aaa  aab  abb
```

## Wildcard [set]

- You can pattern match with a set of characters. For instance, you want files which end with a or b.

```
$ ls

aaa  aab  aac zzb zzc

$ ls aa[ab]

aaa  aab

$ ls *[ab]

aaa  aab  zzb
```

## Thank’s for Your Attention!

Trayan Iliev

IPT – Intellectual Products & Technologies

http://iproduct.org/

https://github.com/iproduct

https://twitter.com/trayaniliev

https://www.facebook.com/IPT.EACAD