

Week4

Markdown

This section is inspired by [Markdown Guide - Basic Syntax](#).

0. Philosophy
 - Easy-to-read and Easy-to-write
1. Headings
2. Paragraphs
 - Indentation
3. Line Breaks
 - Two `\n` or Two white spaces
 - For line break, use trailing white space or the HTML tag at the end of the line
4. Emphasis
 - Bold
 - Italic
 - Bold and Italic
5. Blockquotes
 - Single block
 - Multiline
 - Nested Blockquotes
 - Blockquotes with Other Elements
6. Lists
 - Ordered List
 - Unordered List
 - Adding Elements in Lists
 - Four Spaces or One Tab
 - Nested List
 - Four Spaces or One Tab
7. Code
 - Single Line code
 - Code Blocks
 - Escaping Backticks
8. Horizontal Rules
9. Links
 - Adding Titles
 - Formatting Links
 - Reference-style Links
10. Images
 - Adding Titles
11. Escaping Characters

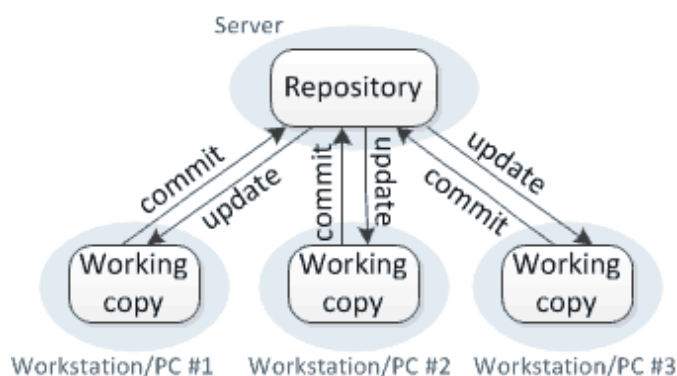
Git

This article is inspired by [Missing Semester - Version Control \(Git\)](#) and [Version control concepts and best practices](#).

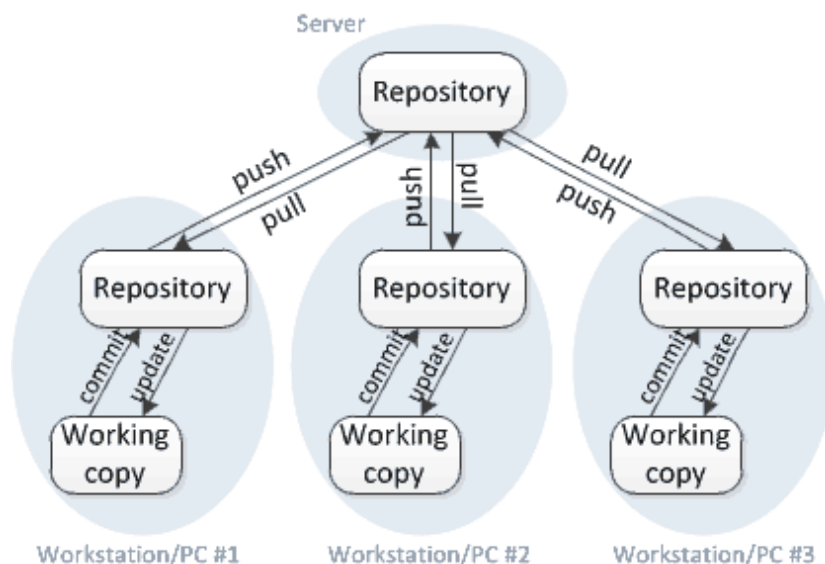
1. Pre-Git history

- What is a Version control systems (VCSs)?
- Purpose of VCS
- Git vs. SVN

Centralized version control



Distributed version control



2. Brief Introduction to Git

- Repositories and working copies

3. Model of Git

- Snapshot
- Linear History
- Data model, as pseudocode
- sha1
- Master and HEAD
- Staging Area

4. Git commands
 - Basics
 - Branching and merging
 - Remotes
 - Undo
5. Advanced Commands
 - `blame`
 - `rebase`
 - `stash`
 - `.gitignore`
6. Best practice to use Git
 - Don't commit generated files
 - Write a good commit messages (cz)
7. More to explore: submodules, gpg, pre-commit hook...

Shell Scripting

This section is inspired by [Shell Scripting for Beginners – How to Write Bash Scripts in Linux](#).

0. Shebang revisited
1. Declare variables in Shell
2. Arithmetic Expressions
3. How to read user input
4. Numeric Comparison logical operators
5. If and AND + OR
6. Loop
 - Looping with Numbers
 - Looping with Strings
 - While Loop
7. Function and Arguments

Exercises for Shell Scripting

1. Write bash functions `marco` and `polo` that do the following. Whenever you execute `marco` the current working directory should be saved in some manner, then when you execute `polo`, no matter what directory you are in, `polo` should `cd` you back to the directory where you executed `marco`. For ease of debugging you can write the code in a file `marco.sh` and (re)load the definitions to your shell by executing `source marco.sh`.
2. Write a `tiny-echo` bash function that prints all the arguments it received to the terminal. You should not use `echo` in your function. (`printf` may be a helpful function.)
3. Say you have a command that fails rarely. In order to debug it you need to capture its output but it can be time consuming to get a failure run. Write a bash script that runs the following script until it fails and captures its standard output and error streams to files and prints everything at the end. Bonus points if you can also report how many runs it took for the script to fail.

```
#!/usr/bin/env bash

n=$(( RANDOM % 100 ))

if [[ n -eq 42 ]]; then
  echo "Something went wrong"
  >&2 echo "The error was using magic numbers"
  exit 1
fi

echo "Everything went according to plan"
```

License

MIT License

Copyright (c) 2022 Yuxiang Qiu

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.