|  |
| --- |
|  |
| Git  Muhammad Waseem Akhtar |
| Maintaining the version of codes |
|  |

Table of Contents

[1 Diffing Files 1](#_Toc162689553)

[2 Applying Changes 1](#_Toc162689554)

[2.1 Practical Application of diff and patch 2](#_Toc162689555)

[2.2 Study guide: diff and patch 2](#_Toc162689556)

[3 Version Control System (VCS) 3](#_Toc162689557)

[3.1 Version Control and Automation 3](#_Toc162689558)

[3.2 What is Git 3](#_Toc162689559)

[3.3 Using Git 4](#_Toc162689560)

[4 Review 6](#_Toc162689561)

[4.1 Version Control Wrap-Up 6](#_Toc162689562)

[4.2 Qwiklabs guidelines and troubleshooting steps 6](#_Toc162689563)

[4.3 Qwiklabs assessment on Git 6](#_Toc162689564)

[5 References 7](#_Toc162689565)

1. Diffing Files

|  |  |
| --- | --- |
| **Commands** | **Descriptions** |
| cat rearrange1.py  cat rearrange2.py | Concatenate: Shows the contents of the file |
| diff rearrange1.py rearrange2.py  diff validations1.py validations2.py | Show line numners the difference between two files |
| diff -u validations1.py validations2.py | Show line numners and contents in the unified difference(human understandable) between two files |
| wdiff -u validations1.py validations2.py | Word-by-word differences shown using color |
| meld -u validations1.py validations2.py | // |
| KDiff3 -u validations1.py validations2.py | Graphical diff b/w files or directories |
| vimdiff3 -u validations1.py validations2.py | // |

1. Applying Changes

|  |  |
| --- | --- |
| **Commands** | **Descriptions** |
| cat cpu\_usage.py | Display contents |
| diff -u Old\_file new\_file > change.diff | Generate a file ”change.diff” with the contents of diff -u |
| cat cpu\_usage.diff | display the contents of the file named "cpu\_usage.diff" in the terminal window |
| patch cpu\_usage.py < cpuusage.diff | applies the changes contained in a diff file to another file |

* 1. Practical Application of diff and patch

|  |  |
| --- | --- |
| **Commands** | **Descriptions** |
| cp disk\_usage.py disk\_usage\_original.py  cp disk\_usage.py disk\_usage\_fixed.py | Copy one file to another |
| ./disk\_usage\_fixed.py  diff -u disk\_usage\_original.py disk\_usage\_fixed.py > disk\_usage.diff  cat disk\_usage.diff | -compares two files  and then send the differnce to disk\_usage.diff  -See disk\_usage.diff |
| patch disk\_usage.py < disk\_usage.diff | patching file disk\_usage.py |
| ./disk\_usage.py | Check the file |

* 1. Study guide: diff and patch

|  |  |
| --- | --- |
| **Commands** | **Descriptions** |
| ~$ cat menu1.txt  Menu1:  Apples  Bananas  Oranges  Pears  ~$ cat menu2.txt  Menu:  Apples  Bananas  Grapes  Strawberries  ~$ diff -u menu1.txt menu2.txt  --- menu1.txt   2019-12-16 18:46:13.794879924 +0900  +++ menu2.txt   2019-12-16 18:46:42.090995670 +0900  @@ -1,6 +1,6 @@  -Menu1:  +Menu:     Apples   Bananas  -Oranges  -Pears  +Grapes  +Strawberries |  |
| ~$ cat hello\_world.txt  Hello World  ~$ cat hello\_world\_long.txt  Hello World  It's a wonderful day!  ~$ diff -u hello\_world.txt hello\_world\_long.txt  --- hello\_world.txt     2019-12-16 19:24:12.556102821 +0900  +++ hello\_world\_long.txt        2019-12-16 19:24:38.944207773 +0900  @@ -1 +1,3 @@   Hello World  +  +It's a wonderful day!  ~$ diff -u hello\_world.txt hello\_world\_long.txt > hello\_world.diff  ~$ patch hello\_world.txt < hello\_world.diff  patching file hello\_world.txt  ~$ cat hello\_world.txt  Hello World  It's a wonderful day! |  |

## Resources for more information

There are other interesting patch and diff commands such as patch -p1 and diff -r . For more information on these commands, check out the following resources:

* <http://man7.org/linux/man-pages/man1/diff.1.html>
* <http://man7.org/linux/man-pages/man1/patch.1.html>

1. Version Control System (VCS)/Sorce Control Management (SCM)/git

|  |  |
| --- | --- |
| **Commands** | **Descriptions** |
| Git-scsm.com | Official git website |
| git –version | Check if git is installed |
| suddo apt install git | Install git on linux |
| Notpad++, atom |  |

* 1. Version Control and Automation

|  |  |
| --- | --- |
| **Commands** | **Descriptions** |
| <https://git-scm.com/doc> | Check out the following links for more information: |
| <https://www.mercurial-scm.org/> |
| <https://subversion.apache.org/> |
| <https://en.wikipedia.org/wiki/Version_control> |

* 1. Using Git

|  |  |
| --- | --- |
| **Commands** | **Descriptions** |
| git config --global user.email "me@example.com" |  |
| git config --global user.name "My name" |  |
| mkdir checks |  |
| cd checks |  |
| git init | Initialize empty git repo in the current directory |
| ls -la | Check if this directory exits |
| ls -l |  |
| ls -l .git/ | Check what files exists in this directory |
| cp ../disk\_usage.py . | Copy disk\_usage.py in the current directory |
| ls -l |  |
| git add disk\_usage.py | Enable git to track file and add it to the staging area |
| git status | Get information about current working tree and pending changes |
| git commit  Add new dis\_usage check.  And exit the aeditor saving info | By this command we tell git that we want to save our changes |

* + 1. Tracking Files

|  |  |
| --- | --- |
| **Commands** | **Descriptions** |
| cd checks  ls -l | Goto the checks directory and list the items in it |
| git status | Check If it requires commit |
| atom disk\_usage.py  git status |  |
| git commit -m 'Add periods to the end of sentences.' |  |
| git status |  |

* + 1. First Step With Git

|  |  |
| --- | --- |
| **Commands** | **Descriptions** |
| Modified file🡪staged file🡪commit file |  |
| touch test.py | Create an empty test.py into the current directory |
| Atom test.py |  |
| Git status |  |
| Git add test.py | Commit the changes-the file is currently at the stage area |
| Git commit -m ‘Add period to the end of sentences.’ | Committed the stage changes |
| Git status |  |
|  |  |

* + 1. The Basic Git workflow

|  |  |
| --- | --- |
| **Commands** | **Descriptions** |
| mkdir scripts  cd scripts  git init |  |
| git config -l | Config it and see the infor about username and email |
| #!/usr/bin/env python3  def main():      pass  main() |  |
| chmod +x all\_checks.py  git status |  |
| Create an empty all\_checks.  # Please enter the commit message for your changes. Lines starting  # with '#' will be ignored, and an empty message aborts the commit.  #  # On branch master  #  # Initial commit  #  # Changes to be committed:  #       new file:   all\_checks.py  # |  |
| #!/usr/bin/env python3  import os  def check\_reboot():      """Returns True if the computer has a pending reboot."""      return os.path.exists("/run/reboot-required")  def main():      pass  main() |  |
| git status | to view pending changes in a Git repository |
| git add all\_checks.py  git status | track our file in the list of changes to be committed |
| git commit -m 'Add a check\_reboot function' |  |
|  |  |

* + 1. Anatomy of Content Message

|  |  |
| --- | --- |
| **Commands** | **Descriptions** |
| cat example\_commit.txt | Detailed infor about changes |
| cd scripts  git log | Short infor about changes (review the commit history for our project) |
|  |  |
|  |  |

* + 1. Study Guide-Git

|  |  |
| --- | --- |
| **Commands** | **Descriptions** |
| **Terms and definitions from Course 3, Module 1** **Commit:** A command to make edits to multiple files and treat that collection of edits as a single change  **Commit files:** A stage where the changes made to files are safely stored in a snapshot in the Git directory  **Commit message:** A summary and description with contextual information on the parts of the code or configuration of the commit change  **Diff:** A command to find the differences between two files  **DNS zone file:** A configuration file that specifies the mappings between IP addresses and host names in your network  **Git:** A free open source version control system available for installation on Unix based platforms, Windows and macOS  **Git directory:** A database for a Git project that stores the changes and the change history  **Git log:** A log that displays commit messages  **Git staging area:** A file maintained by Git that contains all the information about what files and changes are going to go into the next commit  **Modified files:** A stage where changes have been made to a file, but the have not been stored or committed  **Patch:** A command that can detect that there were changes made to the file and will do its best to apply the changes  **Repository:** An organization system of files that contain separate software projects  **Source Control Management (SCM):** A tool similar to VCS to store source code  **Stage files:** A stage where the changes to files are ready to be committed  **Tracked:** A file’s changes are recorded  **Untracked:** A file’s changes are not recorded  **Version control systems (VCS):** A tool to safely test code before releasing it, allow multiple people collaborate on the same coding projects together, and stores the history of that code and configuration | |

1. Review
   1. Version Control Wrap-Up

|  |  |
| --- | --- |
| **Commands** | **Descriptions** |
|  |  |
|  |  |
|  |  |
|  |  |

* 1. Qwiklabs guidelines and troubleshooting steps

|  |  |
| --- | --- |
| **Commands** | **Descriptions** |
|  |  |
|  |  |
|  |  |
|  |  |

* 1. Qwiklabs assessment on Git

|  |  |
| --- | --- |
| **Commands** | **Descriptions** |
|  |  |
|  |  |
|  |  |
|  |  |

1. References

[01] Akhtar, Muhammad Waseem, et al. "The shift to 6G communications: Vision and requirements." *Human-centric Computing and Information Sciences* 10 (2020): 1-27.

[02] Akhtar, Muhammad Waseem, et al. "The shift to 6G communications: Vision and requirements." *Human-centric Computing and Information Sciences* 10 (2020): 1-27.

[03] Akhtar, Muhammad Waseem, et al. "The shift to 6G communications: Vision and requirements." *Human-centric Computing and Information Sciences* 10 (2020): 1-27.

[04] Akhtar, Muhammad Waseem, et al. "The shift to 6G communications: Vision and requirements." *Human-centric Computing and Information Sciences* 10 (2020): 1-27.

[05] Akhtar, Muhammad Waseem, et al. "The shift to 6G communications: Vision and requirements." *Human-centric Computing and Information Sciences* 10 (2020): 1-27.

[06] Akhtar, Muhammad Waseem, et al. "The shift to 6G communications: Vision and requirements." *Human-centric Computing and Information Sciences* 10 (2020): 1-27.

[07] Akhtar, Muhammad Waseem, et al. "The shift to 6G communications: Vision and requirements." *Human-centric Computing and Information Sciences* 10 (2020): 1-27.

[08] Akhtar, Muhammad Waseem, et al. "The shift to 6G communications: Vision and requirements." *Human-centric Computing and Information Sciences* 10 (2020): 1-27.

[09] Akhtar, Muhammad Waseem, et al. "The shift to 6G communications: Vision and requirements." *Human-centric Computing and Information Sciences* 10 (2020): 1-27.

[10] Akhtar, Muhammad Waseem, et al. "The shift to 6G communications: Vision and requirements." *Human-centric Computing and Information Sciences* 10 (2020): 1-27.

[11] Akhtar, Muhammad Waseem, et al. "The shift to 6G communications: Vision and requirements." *Human-centric Computing and Information Sciences* 10 (2020): 1-27.

[12] Akhtar, Muhammad Waseem, et al. "The shift to 6G communications: Vision and requirements." *Human-centric Computing and Information Sciences* 10 (2020): 1-27.

[13] Akhtar, Muhammad Waseem, et al. "The shift to 6G communications: Vision and requirements." *Human-centric Computing and Information Sciences* 10 (2020): 1-27.