

Hands-on Workbook

Source Code Management with Git

Table of Contents

1. [Exercise 2.1: Install Git 3](#_Toc498517727)
2. [Exercise 3.1: Working in GitHub 5](#_Toc498517728)
3. [Exercise 3.2: Using Git Basic Commands 9](#_Toc498517729)
4. [Exercise 4.1: Using Git Advance Commands 12](#_Toc498517730)
5. [Exercise 4.2: Using Git Advance Commands 16](#_Toc498517731)

Module 2

# Exercise 2.1: Install Git

## Scenario

|  |
| --- |
| To get started hands on for Git, we need to install GIT based on OS we use. |

## Steps

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | **Installation Commands:**  Following table lists the commands to install Git in different operating systems:   |  |  | | --- | --- | | **Operating System** | **Command to Install** | | Linux | Run the following command:  sudo yum install git-all | | Debian-based like Ubuntu | Run the following command:  sudo apt-get install git-all | | Windows | Access the following link:  <http://git-scm.com/download/win>  The download will start automatically.  PFA doc. For git installing git instructions. | |
| 2 | **Initial Configuration Setup:**  To set an identity, use the following commands:  git config --global user.name  git config --global user.email  Example:  git config --global user.name “Rakhi Parashar"  git config --global user.email rakhi.parashar[@accenture.com](mailto:johndoe@example.com) |
| 3 | **To view all the configurations, use the following command:**  git config –list |
| 4 | **Initializing Git repository:**  To initialize Git repository run the following commands:  git init (to initialize git repository)  cd .git (go to git directory and verifying GIT repository initialization)  dir (to list all files and directories .git contains) |
| ------- End of Exercise ------- | |

Module 3

# Exercise 3.1: Working in GitHub

## Scenario

|  |
| --- |
| Create an account in GitHub. Login to the GitHub and create a new repository as NewProject and branch gitbash. Create and add the file to the project. Make some changes in the code and then commit your changes in the master as well develop. |

## Steps

|  |  |
| --- | --- |
| 1 | **Creating a GitHub Account:**   1. Login to <https://github.com/join> in new browser. 2. Create new account by giving username, email address and password and login. |
| 2 | **Creating Github Repository:**  1. Click **New repository** to create a repository.  2. Provide name of new Repository Example: **SampleProject.**  3. Select for **Public** button.  4. Check “**Initialize this repository with a README**”  5. Click **Create repository**.    Please review your repository created as below: |
| 3 | **Adding a file & editing an existing file:**   1. Click **Create new file**. 2. Name the file. For this exercise “Samplecode.sh” 3. Add lines of code.   For example:  echo “welcome to sample project”     1. Give comments. 2. Ensure the at you can committing in the Master branch. 3. Click **Commit new file**.     **View the new files added**     1. Click any file that had been created in the repository or click on the default **README.md** file which was created along with the repository. 2. Click the **Edit this file** icon. 3. Edit the file content in the **Edit file** tab. 4. Click **Preview changes** to view the changes. The new content will be displayed in Green. 5. Give a meaningful comment when committing which explains the changes made to the file.   Note: You can choose to commit the changes to the same branch or to a new branch.   1. Click **Commit changes** to propose the file changes. |
| ------- End of Exercise ------- | |

# Exercise 3.2: Using Git Basic Commands

## Scenario

|  |
| --- |
| Clone the repository that we created in GitHub into local machine. Add a file to the repository and commit the changes in the repository. |

## Steps

|  |  |
| --- | --- |
| 1 | **Clone the repository**  Execute the following command in order to clone the repository into local machine which we created in GitHub:  $ git clone <https://github.com/rakhar/SampleProject> or  $ git clone <https://github.com/Accenture/adop-doa-materials.git>  $ git checkout -b develop  The result of using the command is as follows: |
| 2 | **Adding file to a repository**  Create a text file in GIT initialized folder and save the changes to the staging using the command, “git add <filename>” as shown in the following screen capture. |
| 3 | **Committing the changes to the Repository**  To commit the changes to the GIT repository, execute the following command:  $ git commit –m “initial version”  The changes committed to the repository is shown in the following screen capture: |
| 4 | **Using GIT Push Command**  Run the following command to push commit on the repository:  $ git push –u origin master |
| 5 | **Tagging a repository**   |  |  | | --- | --- | | **Commands** | **Usage** | | $git tag v1 | Assign a tag to the current version of the file. | | $git checkout v1^ | Assign a tag to the previous version, instead of mentioning the hash while checking out. | | $git tag v0 | Name the previous version as v0. | | $git tag | View all the tags. | | $git show v1 | View a tagged version of a file. | |
| 6 | **Viewing the log details of the repository**  To log into the GIT repository, execute the following command and see the result:  $ git log |
| 7 | **Synchronizing local & remote repository**  See remote repository in GIT bash with the following command:  $ git remote –v  Use the following command to rename an existing remote repository:  $ git remote rename origin destination    **Note**: The Origin is the default repository created when a project is created. |
| 8 | **Using pull command to update the changes to the remote repository**  To pull the changes to local repository from master, run the following command.  $ git pull    **Note**: If there are changes updated to the remote repository (could be by a different user),  the changes can be pulled into/updated to the local repository using the GIT Pull command. |
| ------- End of Exercise ------- | |

Module 4

# Exercise 4.1: Using Git Advance Commands

## Scenario

|  |
| --- |
| **Task 1**: Create a branch devTest, pull the changes from the master branch over here and then make some more changes, save those changes and then commit your changes. Go to develop branch and then merge the changes of devTest branch in develop branch.  **Task 2**: Compare both the branches develop and devTest with master, if any file is missing from the master then fetch that file and place in both branches (develop and devTest). Use commands like grep, log, show, status, diff and bisect for analyzing history.  **Task 3**: Add a new file in the working tree and make the changes and commit. Post this copy this file to another file File2 and then rename the newly added file. Finally remove the File2 from the working tree. Check status of the working tree.  **Task 4**: Clone the sample-java-project.git project from git repository. Open pom.xml and remove the first few lines. OR you can create any other bug and commit the changes. Now use binary search to find the commit that introduced a bug. Hint: use Bisect and blame command. |

## Steps

|  |  |
| --- | --- |
| 1 | **Scenario Task**   * Create a branch devTest, pull the changes from the master branch over here and then make some more changes, save those changes and then commit your changes. * Go to develop branch and then merge the changes of devTest branch in develop branch.   Following steps and screen captures should help you to accomplish the considered task:   1. Run the command in gitbash “**git checkout -b devTest**” for creating new branch. 2. Use command “**git pull**” to get the latest code from master in this new devTest branch. 3. Make changes to any of the file for example: README.md file. 4. Save and commit the changes. 5. Checkout develop branch using command “**git checkout develop**”. 6. Merge the changes of devTest branch to develop branch using command “**git merge devTest**”. |
| 2 | **Scenario Task**  Compare both the branches develop and devTest with master, if any file is missing from the master then fetch that file and place in both branches (develop and devTest).  Use commands like grep, log, show, status, diff etc. |
| 3 | **Scenario Task**   * Add a new file in the working tree and make the changes and commit. * Post this copy this file to another file File2 and then rename the newly added file. * Finally remove the File2 from the working tree. Check status of the working tree. * Also now if you don’t want to commit these changes just stash/save your changes and also give them tags. * Use log for logging.   Following steps should help you to accomplish the considered task:   1. To create new file use command “**touch Test2**” 2. Make changes to the file and save it. 3. Run the following command:   git add Test2  git commit -m “initial” and “git push”   1. Use command “**cp Test2 Test3**” to copy files. 2. Rename Test3 as NewTest by executing the following command:   **mv Test3 NewTest**   1. Remove Test2 “**rm -f Test2**” |
| 4 | **Scenario Task**  Following steps and screen captures should help you to accomplish the considered task:   1. Git clone https://github.com/skeeto/sample-java-project.git 2. Open **pom.xml** and remove few lines. 3. Commit the changes. 4. Now use Bisect and Blame command to find which commit has introduced the bug and who has done. |
| ------- End of Exercise ------- | |

# Exercise 4.2: Using Git Advance Commands

## Scenario

|  |
| --- |
| Following are the key scenarios where you need to use some of the other Git Advance Commands:   * Creating an alias for most frequently used command based on the history of commit. * Identifying the file status by displaying which stage is a file in Git * Merging without using merge command * Keeping the changes in local branch that you want to commit later. * Committing only specific commits from master to your current * Removing untracked files and directories |

## Steps

|  |  |
| --- | --- |
| 1 | **Scenario Task - Creating an alias for most frequently used command based on the history of commit.**  Alan wants to create an alias for most frequently used command based on the history of commit. He creates an alias by using the following command to save his writing time.  Command: git config –global alias.XZ   * Example: git config --global alias.X "log -1 HEAD"   After creating alias, the command can be as follows.   * git X instead of git log -1 HEAD. |
| 2 | **Scenario Task - Identifying the file status by displaying which stage is a file in Git**  Alan has his local files with similar names. He has forgotten what was the last action taken on those files. He wants to display which stage is a file in GIT.  Command: git status –s    “**M**” indicates file is modified and the file needs to staged for commit. |
| 3 | **Scenario Task - Merging without using merge command**  Alan wants to merge the repository without using merge command.  Pull upstream changes with rebase instead of merge. |
| 4 | **Scenario Task - Keeping the changes in local branch that you want to commit later**  Alan wants to keep the changes in local branch and do not want to commit right now, and may wish to commit later.  Stash : Temporarily save/stash the changes in current branch  Command : git stash  git status  git list  git stash pop |
| 5 | **Scenario Task - Committing only specific commits from master to your current**  Alan wants commit only specific commits from master to the current branch.   * Merge a cherry-picked remote commit with your branch * Command: git cherry-pick <commitSHA> * Git cherry –v SampleBranchName * Git cherry –v master |
| 6 | **Scenario Task - Removing untracked files and directories**  Alan wants to remove untracked files and directories.  Git clean –f (remove untracked files)  Git clean –fd (remove untracked files/directories)  Git clean –nfd (list all which will be removed) |
| ------- End of Exercise ------- | |