YDM1 — YDM1 TASK 1: VERSION CONTROL USING GIT FOR GITLAB

VERSION CONTROL — D197
PRFA — YDM1

TASK OVERVIEW SUBMISSIONS EVALUATION REPORT

COMPETENCIES

4072.1.1 : Implements Version Control
The learner implements version control processes and solutions to maintain source code.

INTRODUCTION

Version control, also referred to as revision control or source control (i.e., git), is an efficient way of managing changes to files over time. Managing files over time enable teams to collaborate freely because members can work on the same file at the same time and merge changes later. Version control and version control systems can be leveraged by anyone in an organization, from the development team to the Human Resources department and even the product development team.

For this task you will use the scenario below to create a GitLab repository and submit it along with screenshots showing the work completed.

SCENARIO

You have been hired as a consultant at a company. The company has a collection of files for their website that need modification to complete the project deliverables. However, the company's contract requires that all documents be saved to a source repository, and there are other consultants working concurrently. You have been asked to make upgrades to the website.

REQUIREMENTS

Your submission must be your original work. No more than a combined total of 30% of the submission and no more than a 10% match to any one individual source can be directly quoted or closely paraphrased from sources, even if cited correctly. The similarity report that is provided when you submit your task can be used as a guide.

You must use the rubric to direct the creation of your submission because it provides detailed criteria that will be used to evaluate your work. Each requirement below may be evaluated by more than one rubric aspect. The rubric aspect titles may contain hyperlinks to relevant portions of the course.

Help

*Tasks may **not** be submitted as cloud links, such as links to Google Docs, Google Slides, OneDrive, etc., unless specified in the task requirements. All other submissions must be file types that are uploaded and submitted as attachments (e.g., .docx, .pdf, .ppt).*

Create a private repository by doing the following:

> *Note: Screenshots should include the git response from the command prompt or the entire graph window.*
>
> *Note: All operations should be performed on your local machine using Git Bash or another shell unless GitLab is specifically mentioned.*

Prepare Your Repository With Initial Data On GitLab

A. Create a private repository in gitlab.com named "(yourstudentID) D197" that includes *all* the files contained in the attached "WGU-Hub.zip."
   1. Include the repository link in the "Comments to Evaluator" section when you submit your task.
   2. Add "WGU-Evaluation" as a member with reporter access to your repository on GitLab.
   3. Include a screenshot of your current repository graph in GitLab.

Clone Your Repository On Your Local Machine

B. Clone your remote repository to your local machine using the command line interface. Include a screenshot of the command line action and be sure to have your repository name visible in the command prompt.

Make Changes, Commit, and Push

C. Modify **three** HTML files you choose on the master branch by the doing the following:
   1. Commit *each* change with a short, meaningful message that explains *all* changes you have made to the **three** HTML files. Include a screenshot for *each* git command for *each* change and be sure to have your repository name visible in the command prompt.
   2. Push the branch to GitLab. Include a screenshot of the command line action and be sure to have your repository name visible in the command prompt.

Create A Branch and Make a Change

D. Modify a file on a new branch doing the following:
   - create a "README" branch using the command line interface
   - add your student ID to the README.md file
   - push the changes to the remote repository
   - include a screenshot of the command line action and be sure to have your repository name visible in the command prompt
   1. Include a screenshot of the current repository graph in GitLab after pushing changes.

Simulate A Merge Conflict

E. Introduce a merge conflict with the "README " branch by doing the following:
   - add the git version to the README .md file on the master branch
   - merge the "README" branch to the master branch
   - include a screenshot that demonstrates the conflict of this merge command line action and be sure to have your repository name visible in the command prompt
   1. Resolve the created conflict and push changes to the master branch in GitLab. Include a screenshot of the current repository graph in GitLab.

Tag A Branch

F. Specify a version for your repository by doing the following:
- tag the master branch V.1.0.0
- push the tag to GitLab
- include a screenshot of the command line action and be sure to have your repository name visible in the command prompt

1. Include a screenshot of the current repository graph in GitLab.

Add A Retrospective Directory To Repository

G. Add "retrospective" information by doing the following:
1. Create a directory on the master branch named "retrospective."
2. Create a log.txt that will contain the output of the git log command in the "retrospective" directory.

   Note: Use "git log > log.txt" to redirect output to the file.

3. Create a "summary.txt" file in the "retrospective" directory and include the following:
   - summarize how the merge conflict in part E1 was resolved
   - describe the **three** changes that you made in part C
4. Create a PDF file with screenshots and comments from parts A–F and place the PDF in the "retrospective" directory.
5. Push *all* changes made in parts G1–G4 to GitLab.

Final Step

H. Submit the zip file of your GitLab repository.

I. Acknowledge sources, if used, using in-text citations and references, for content that is quoted, paraphrased, or summarized.

J. Demonstrate professional communication in the content and presentation of your submission.

File Restrictions

File name may contain only letters, numbers, spaces, and these symbols: ! - _ . * ' ( )
File size limit: 400 MB
File types allowed: doc, docx, rtf, xls, xlsx, ppt, pptx, odt, pdf, txt, qt, mov, mpg, avi, mp3, wav, mp4, wma, flv, asf, mpeg, wmv, m4v, svg, tif, tiff, jpeg, jpg, gif, png, zip, rar, tar, 7z

RUBRIC

A:PRIVATE GITLAB REPOSITORY

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
A private repository is not created.	The repository created in gitlab.com does not contain *all* the files from "WGU-Hub.zip."	The private repository created in gitlab.com is named correctly and contains *all* files from "WGU-Hub.zip."

	Or the repository is not private or is named incorrectly.	

A1:REPOSITORY LINK

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
A repository link in the comment section is not included.	The repository link included in the "Comments to Evaluator" section is *either* broken or inaccurate (i.e., the link does not arrive at the appropriate repository).	The correct and accurate repository link is included in the "Comments to Evaluator" section.

A2:REPORTER ACCESS

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
No access to the repository link is added.	The "WGU-Evaluation" group is granted user-level access to the repository on GitLab but not reporter access.	The "WGU-Evaluation" group is added as a member with reporter-level access to the repository.

A3:REPOSITORY GRAPH SCREENSHOT

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
A screenshot of the current repository graph in GitLab is not included.	The screenshot of the current repository graph in GitLab is inaccurate. Or the screenshot does not show *all* the changes made in part A.	The screenshot of the current repository graph in GitLab is accurate and shows *all* the changes made in part A.

B:CLONING REMOTE REPOSITORY

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
A screenshot of the cloned remote repository is not provided.	The screenshot does not show that the remote repository was cloned or that the command interface was used to clone the repository. Or the command prompt does not include the repository name.	The screenshot shows that the remote repository was cloned, that the command line interface was used to clone the repository, and the command prompt includes the repository name.

C1:COMMIT WITH A MESSAGE

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
The screenshots do not show *any* HTML files.	Screenshots are not provided for *each* git command for *each* change. Or the screenshots do not show 1 or more changes committed to the branch, or 1 or more of the short, meaningful messages are incorrect. Or 1 or more of the messages do not logically explain the changes made to the 3 HTML files in part C. Or the command prompt does not include the repository name.	Screenshots are provided for *each* git command for *each* git change. *All* screenshots show *all* changes committed to the branch, and the short, meaningful messages are correct. *Each* of the messages logically explain the changes made to the 3 HTML files in part C. The command prompt includes the repository name.

C2:PUSH TO REMOTE

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
A screenshot of the command line actions is not included.	The screenshot of the command line actions does not show the push action of a branch to GitLab. Or the command prompt does not include the repository name.	The screenshot shows a correct push of the branch to GitLab and the command line actions. The command prompt includes the repository name.

D:MODIFY A FILE

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
A modified file on a new branch is not provided.	The file on a new branch is modified, but 1 or more of the given actions is not completed or they are completed incorrectly. Or a screenshot of the command line action is not provided. Or the command prompt does not include the repository name.	The file on a new branch is modified with *each* of the given actions correctly completed. The screenshot of the command line action is provided. The command prompt includes the repository name.

D1:PROOF OF REMOTE CHANGES

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT

A screenshot of the current repository graph is not included.	The screenshot does not show the current repository graph in GitLab after pushing changes. Or the command prompt does not include the repository name.	The screenshot shows the current repository graph in GitLab after pushing changes. The command prompt includes the repository name.

E:INTRODUCE A MERGE CONFLICT

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
A merge conflict is not introduced by performing *any* of the given actions.	The merge conflict introduced with the "README' branch does not perform *each* of the 3 given requirements.	The merge conflict is introduced with the "README" branch" and *each* of the 3 given actions are performed.

E1:RESOLUTION AND PROOF OF REMOTE CHANGES

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
A screenshot of the current repository graph is not included.	The screenshot of the current repository graph does not show that the conflict was resolved. Or the push changes are not reflected in the master branch in GitLab.	The screenshot of the current repository graph in GitLab shows the resolution of the conflict. The push changes are reflected in the master branch in GitLab.

F:TAG MASTER BRANCH

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
A version of the repository is not specified, or *any* of the given actions are not performed.	The version of the repository could not be specified because 1 or more of the 3 given actions are not completed.	The version of the repository is specified and each of the 3 given actions are completed.

F1:PROOF OF REMOTE CHANGES

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
A screenshot of the repository graph is not included.	The screenshot of the current repository graph in GitLab does not show the correct tag version label, or the command prompt	The screenshot of the current repository graph in GitLab has the correct tag version label and the command prompt includes the repository name.

	does not include the repository name.	

G1:RETROSPECTIVE DIRECTORY

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
A “retrospective” directory is not created.	The “retrospective” directory is not on the master branch.	The “retrospective” directory is created on the master branch.

G2:

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
A log.txt file is not created.	The log.txt file is not created in the “retrospective” directory or does not contain the git log output information in the file.	The log.txt file is created in the “retrospective” directory and contains the git log output information in the file.

G3:

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
A “summary.txt” file in the “retrospective” directory is not created.	The “summary.txt” file in the “retrospective” directory does not summarize how the merge conflict in part E3 was resolved in the “summary.txt” file. Or does not provide a description of the 3 changes that were made in part C.	The “summary.txt” file in the “retrospective” directory includes the summary of how the merge conflict in part E3 was resolved in the “summary.txt” file and includes the description of the 3 changes that were made in part C.

G4:PDF WITH SCREENSHOTS

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
A PDF with screenshots is not provided or is not placed in the “retrospective” directory.	The PDF is placed in the “retrospective” directory, but does not include *all* the screenshots or *all* comments from parts A–F.	The PDF contains *all* screenshots and *all* comments from parts A–F and is inserted in the “retrospective” directory.

G5:PUSH ALL CHANGES

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
No changes are pushed.	Not *all* changes made in parts G1–G4 are pushed from to GitLab, including the "retrospective" directory.	*All* changes from parts G1–G4 are pushed from part G to GitLab, including the "retrospective" directory.

H:ZIP FILE

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
No zip file is submitted.	The zip file of the GitLab repository is submitted but is missing screenshots, comments, or command line screenshots.	The zip file of the GitLab repository is submitted with *all* requested screenshots labeled with the letter of the corresponding prompt, *all* comments, and *all* command line screenshots have the candidate's repository name visible in the command prompt.

I:SOURCES

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
The submission does not include both in-text citations and a reference list for sources that are quoted, paraphrased, or summarized.	The submission includes in-text citations for sources that are quoted, paraphrased, or summarized and a reference list; however, the citations or reference list is incomplete or inaccurate.	The submission includes in-text citations for sources that are properly quoted, paraphrased, or summarized and a reference list that accurately identifies the author, date, title, and source location as available.

J:PROFESSIONAL COMMUNICATION

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
Content is unstructured, is disjointed, or contains pervasive errors in mechanics, usage, or grammar. Vocabulary or tone is unprofessional or distracts from the topic.	Content is poorly organized, is difficult to follow, or contains errors in mechanics, usage, or grammar that cause confusion. Terminology is misused or ineffective.	Content reflects attention to detail, is organized, and focuses on the main ideas as prescribed in the task or chosen by the candidate. Terminology is pertinent, is used correctly, and effectively conveys the intended meaning. Mechanics, usage, and grammar

promote accurate interpretation and understanding.

WEB LINKS

GitLab

SUPPORTING DOCUMENTS

WGU-Hub.zip