# GitHub User Guide for MCTs

Cloud services, such as Office 365 and Microsoft Azure, are updated frequently. This leads to issues for Microsoft Certified Trainers (MCTs) when they teach courses, such as *10993:* *Integrating On-Premises Identity Infrastructure with Microsoft Azure*, because lab steps change frequently as the service changes. Due to the frequency of the changes and the fact that there may not be any notification when changes occur, it can be difficult for the course development team to rapidly identify and address any lab changes.

To address these issues, we are using GitHub to publish the lab steps and lab scripts for courses that cover cloud services like Microsoft Azure. Using GitHub allows for collaboration between the course’s authors and MCTs to keep the content current with Microsoft Azure platform changes. Using GitHub allows the MCTs to provide feedback and suggestions for lab changes, and then the course authors can update lab steps and scripts quickly and relatively easily.

When you prepare to teach these courses, you should ensure that you are using the latest lab steps and scripts by downloading the appropriate files from GitHub.

This user guide is for MCTs who are new to GitHub, and it provides steps for connecting to GitHub, downloading and printing course materials, updating the scripts that students use in labs, and explaining how you can help ensure that this course’s content remains current.

**Note:** It is strongly recommended that MCTs and Partners access these materials and in turn, provide them separately to students. Pointing students directly to GitHub to access lab steps as part of an ongoing class will require them to access yet another UI as part of the course, contributing to a confusing experience for the student. An explanation to the student regarding why they are receiving separate lab instructions can highlight the nature of an always-changing cloud-based interface and platform. Microsoft Learning support for accessing files on GitHub and support for navigation of the GitHub site is limited to MCTs teaching this course only.

### GitHub terminology

GitHub introduces terminology that might be new to you, and the following list includes terms and concepts that this document uses. However, for a full list of GitHub terms, refer to the “GitHub Glossary” at <https://help.github.com/articles/GitHub-glossary/>.

*Git* and *GitHub*: *Git* is an open-source, change-tracking program, and *GitHub* is a site/solution built on Git. There are other websites and solutions that use Git as their backend. You would use GitHub primarily for open-source (public) development projects, and it is free for those projects. However, if you want to use GitHub for projects that are private, and not open source, you must sign up for a paid version.

*Repo* or *Repository*: Each project in GitHub is in a repository, or *repo*. A repo contains all of a project’s files, including documentation, and it supports revision history. A repository can be public or private, and you can have a local copy of the repo on your computer hard drive, or you can use the repo within GitHub.

*Markdown*: This is a text-file format that you can use for creating documentation. It is text-based and very simple to update, which makes it easy to use during collaboration. GitHub then renders it as HTML.

*GitHub flavored markdown (GFM)*: There are many variations, or flavors, of the Markdown file format. The GitHub version, commonly referred to as *GFM*, is one of the most common variations of Markdown. For more information about GFM and how you can use the Markup format for your GFM documents, refer to “Getting started with writing and formatting on GitHub” at https://help.github.com/articles/getting-started-with-writing-and-formatting-on-github/.

*Fork*: This is a copy of another repo that resides in your GitHub account, in comparison to a *branch*, which lives in the original repo. See “Branch” directly below.

*Branch*: This is a copy of a repository that resides in the same repository as the original. You can merge a branch with the original.

*Fetch*: This is the process of retrieving a copy of the latest changes from an online repo. However, a fetch *does not* merge changes.

*Pull*: This is the process of fetching the latest changes from an online repo and merging them with local changes.

*Merge*: This isthe process of fetching changes from one branch and applying them to another. This includes retrieving changes from an online repo, and then applying them to that repo’s local version.

*Pull request*: This is a set of proposed changes to a repo that a user submit, and a repo’s owners or collaborators then can accept or reject the pull request.

*Push*: This is the process of sendingor submitting your local changes to the online repo.

*Collaborator*: This is a GitHub user that has permissions to add, delete, or change a repo’s content.

## Overview of Microsoft Learning's GitHub solution for course labs

The Microsoft Learning team has created a solution that allows them to publish updated lab and lab answer keys (LAKs) and updated lab scripts regularly to GitHub. The solution also includes a script and tools that you can use to print labs and lab answer keys from Microsoft Word .docx files. However, if you want to use this solution, you must perform several steps the first time that you download and print lab files. A later section of this file details these steps, which include that you must:

1. Sign-up for a GitHub account.
2. Install the GitHub Desktop.
3. Install the prerequisite software:

* Pandoc version 1.19.2
* Windows PowerShell Community Extensions 3.2.0

Once you sign up for GitHub and install the prerequisite software, the steps for downloading and printing the course-lab materials are the same for each course.

You can use a variety of tools that support Git with GitHub, including Microsoft Visual Studio, Visual Studio Code, or any of the Git command-line tools that are available online.

**Note:** GitHub has a desktop client and a command-line interface. Throughout this document, we use the desktop client. If GitHub and Git are new concepts, and you would like a more in-depth introduction, refer to the “GitHub [Hello World](https://guides.github.com/activities/hello-world/) guide” at <https://guides.github.com/activities/hello-world/> .

## Prerequisites

The following section details the prerequisites for using GitHub and the Microsoft Learning courseware lab solution.

### Signing up for a GitHub account

In order to clone a repo or collaborate with Microsoft Learning on GitHub, you need to sign up for a GitHub account.

To sign up for a GitHub account, perform the following steps.

1. In your browser, navigate to <https://GitHub.com/>.
2. In the **Pick a username** text box, enter a unique user name.
3. In the **Your email address** text box, enter your email address.
4. In the **Create a password** text box, enter a password that meets GitHub’s complexity requirements.
5. Click **Sign up for GitHub**.
6. In the **Welcome to GitHub** page, make sure that **Unlimited public repositories for free** check box is selected.
7. Click **Finish sign up**.

GitHub will send a confirmation email to the email address that you provide. You must open the email, and then click **Verify email address**.

### Installing GitHub Desktop

GitHub Desktop provides a graphical user interface (GUI) for GitHub, and you can use it to perform most common functions. There are some operations that you can perform only from a command line, but none of these operations are necessary for downloading and printing lab files.

To install the GitHub Desktop, perform the following steps:

1. In your browser, navigate to <https://desktop.GitHub.com/>.
2. Click **Download for Windows**.
3. When the **GitHubDesktopSetup.exe** file has downloaded, double-click the file to start the setup, or click **Run** if you receive a prompt from Internet Explorer.
4. In the **Application Install - Security Warning** dialog box, click **Install**.
5. Close GitHub Desktop.

### Installing Pandoc version 1.19.2

Pandoc is a tool that you can use to convert files from one format to another. It can read many formats, including GFM, and you use it output Microsoft Word's .docx format. Pandoc is the tool behind the scripts that Microsoft Learning provides to create Word documents from the Markdown file format of the lab files. If you do not install Pandoc, the document-creation script fails.

To install Pandoc, perform the following steps:

1. In your browser, navigate to <https://github.com/jgm/pandoc/releases/tag/1.19.2>.
2. Click **pandoc-1.19.2-windows.msi**.
3. When the **pandoc-1.19.2-windows.msi** file has downloaded, double-click the file to start the setup, or click **Run** if you receive a prompt from Internet Explorer.
4. In the **Pandoc 1.19.2 Setup** dialog, review the License Agreement, select **I accept the terms in the License Agreeement**, and then click **Install**.
5. Click **Finish**.

### Installing PowerShell Community Extensions 3.2.0

PowerShell Community Extensions (PSCX) is an open-source project that extends Windows PowerShell with scripts, cmdlets, functions, and other features. PSCX version 3.2.0 is the most current (as of 6/16/2016) PSCX version. You use PSCX to create the .zip files that contain your .docx files. Please note, if you do not install these extensions, the document-creation script fails.

To install PSCX 3.2.0, perform the following steps:

1. In your browser, navigate to [http://pscx.codeplex.com/releases](http://pscx.codeplex.com/releases/view/133199).
2. Under **RECOMMENDED DOWNLOAD**, click **Pscx-3.2.0.msi**.
3. When the **Pscx-3.2.0.msi** file has downloaded, double-click the file to start the setup, or click **Run** if you receive a prompt from Internet Explorer.
4. In the **PowerShell Community Extensions 3.2.0 Setup** dialog box, review the License Agreement, select **I accept the terms in the License Agreeement**, and then click **Install**.
5. If the **User Account Control** dialog box appears, click **Yes**.
6. Click **Finish**.

**Important:** After you install Pandoc and PSCX, you must restart your computer to complete the installation. If you do not restart your computer, the document-creation script might fail.

## Downloading and printing lab files

The labs are stored on GitHub in a repo. The structure for each Microsoft Learning course repo is similar, and each contains the following folders:

**Allfiles:** This folder contains any supporting files for the labs. This is the same as the Allfiles folder that would appear on the virtual hard disk for virtual machines (VMs).

**Build**: This folder contains the Windows PowerShell script and supporting files for creating Word documents from the lab files that are in the Markdown format.

**Instructions:** This foldercontains the lab files and LAK files, which are in the Markdown format.

You need all of these folders if you want to print the lab files.

### Downloading the latest materials for course labs

If you want to build Word documents from Markdown files, you must [clone](https://help.github.com/articles/cloning-a-repository/) or fetch a copy of the repo on your local computer. If you want to clone the files, you must know the GitHub location of the course files. You can use the **Search GitHub** search box on the GitHub home page to search for these files by using the course number. You also can browse through the repos under the [Microsoft Learning organization](https://github.com/MicrosoftLearning) page on GitHub. The Microsoft Learning page on GitHub is located at <https://github.com/MicrosoftLearning/>.

#### To clone the course repo to your local machine

1. In your browser, navigate to the online repo in GitHub.
2. On the **repo** page, click **Clone or download**.
3. In the **Clone with HTTPS** dialog box, click **Open in Desktop**.
4. In the **Internet Explorer** confirmation dialog box, click **Allow** (or the equivalent for your browser).
5. Switch to GitHub Desktop.
6. In the **Browse For Folder** dialog box, select a folder as the root for the local repo, and then click **Ok**.

If you plan to clone several repos, you can choose one common folder in this step. This creates a subfolder for each repo.

1. In the **Repositories** list, right-click the repository name, and then click **Open in Explorer** to view the local files.

After you clone a repo the first time, on subsequent visits, you can open GitHub Desktop, select the repository, and then click **Sync** to retrieve the latest files.

**Note:** For more information about synchronizing your repo, refer to Working with your remote repository on GitHub or GitHub Enterprise at <https://help.github.com/desktop/guides/contributing/working-with-your-remote-repository-on-github-or-github-enterprise/>.

### Printing the lab and LAK files

If you want to print lab and LAK files, you must convert them Word documents first. Microsoft Learning has a Windows PowerShell script that automates this task. The script creates the Word documents, and then packages the Word documents into .zip files. At the same time, it creates a .zip file that contains the lab’s supporting files such as scripts and text files, which you will need when you set up your lab environment.

The Windows PowerShell script, Pandoc.ps1, is in the **\Build** folder. The folder also contains template.docx, which the script uses to format files in Word. Do not alter the template.docx file.

#### To convert the lab files and create the Zip packages:

1. In **File Explorer,** navigate to the folder in the repo that you cloned, such as

..\Documents\GitHub\10993-IntegratingOn-PremisesIdentitywithAzure\Build.

1. Right-click the file **pandoc.ps1**, and then click **Run with PowerShell**.
2. In the **Windows PowerShell** window, if you receive an **Execution Policy Change** prompt, type **Y**, and then press Enter.
3. When you receive the **What is the current version?** prompt, enter a short string or number to uniquely identify the .zip files that is built.

**Note:** The **current version** string is added to the name of the .zip file.

1. Switch to File Explorer, and in the **\Build** folder, select the .zip files that you just were created. The file names will be **allfiles-vversion.zip** and **\*\*lab\_instructions-v\_\_version\_\_.zip\*\***.
2. Move these files to a new location, so that you can avoid attempting to add them to the repo inadvertently as part of a Pull request.

**Note:** To avoid receiving the **Execution Policy Change** prompt, you can change the [**Set-ExecutionPolicy**](https://technet.microsoft.com/en-us/library/ee176961.aspx) setting in Windows PowerShell to execute scripts without restriction. After changing the **ExecutionPolicy** property, be aware now scripts that you run have the power to make disruptive things happen to your computer.

#### To print the lab files:

* Open the lab files in Microsoft Word, and then print them by using the Word print functionality.

## Receiving update notifications, suggesting changes, and collaborating on projects

You can configure your GitHub experience so that you receive notifications when updates occur to a GitHub repo. There are several ways in which you can sign up for notifications, and many of them relate to the many ways that you can collaborate on a project. To receive notifications, you can:

*Watch repositories*: When you watch a repository, GitHub subscribes you automatically to notifications for any new pull requests or issues that are created for that specific repository. You automatically watch any repository that you create or for which you are a collaborator.

*Pull request*: When you create a pull request, and propose that the owners of a repo accept a change that you make, you automatically subscribe to receive notifications for the related discussion about the pull request. In order to create a Pull request you must first create a branch.

*Comments*: When you make comments about another person’s pull request, GitHub subscribes you automatically to the forum that pertains to that comment, or you can subscribe to the forum manually.

*Issues*: An issue is a suggestion, question, or request that pertains to a repository. Each issue has its own discussion, and you can subscribe to issues, or GitHub subscribes you automatically to issues that you create.

*Mentions*: When another user mentions you in a conversation, using your GitHub user name (@*username*), GitHub subscribes you automatically to the discussion.

You can modify how and when you receive notifications, and you also can unsubscribe to any or all discussions.

### Watching a repo

The simplest way to make sure you know about any changes to a repo is to **watch** it. You can do that, even if you do not clone a local copy of the repo.

To watch a repo, perform the following steps:

1. In Internet Explorer, navigate to the repo on GitHub.
2. Click **Watch**, and then under **Notifications**, select **Watching**.

To quit watching a repo, perform the following steps:

1. In Internet Explorer, navigate to the repo on GitHub.
2. Click **Watch**, and then under Notifications, select **Not watching**.

**Note:** You can select the **Ignoring** option under the **Watch** drop-down list. However, this means that you receive *no* notifications, even if another user includes you in a discussion with the mention functionality and your GitHub user name. Therefore, you should be careful configuring the **Ignoring** option.

### Suggesting changes and collaborating on a repo

GitHub makes it easy to collaborate with other Microsoft Learning users on the courses in which you are interested.

You can modify your own copy of the lab materials, and then submit your changes to Microsoft Learning so that they can incorporate your updates. You might want to modify your lab materials if:

You find a mistake in a lab.

The UI has changed since the lab was created.

You think that the lab needs improvements or modifications.

To modify lab materials, you should branch the repo, make updates in your branch, and then submit a pull request to the main (master) branch. This allows Microsoft Learning staff, and other MCTs and GitHub users to review, and comment on, your changes.

You can review and comment on changes that other users make, and Microsoft Learning staff then approves and merges these changes into the master branch. This action notifies any user who is watching the repo that a change has occurred.

#### To create a repo branch:

1. In Internet Explorer, navigate to the repo on GitHub.
2. Click **Branch : *branchname***, and then from the **Branches** list, select the branch you want to copy.
3. If there is only one branch, the **Branch** drop-down list shows **Branch: master**, and the only branch that is available is **master**.
4. In the blank text box, type the name of the branch that you want to create.
5. Click **Create branch: *new branch name*** when it appears.

#### To delete a repo branch:

1. In Internet Explorer, navigate to the repo on GitHub.
2. Click ***n* branches**, where ***n*** is the number of existing branches.
3. On the **Branches** page, in the row for the branch that you want to delete, click **Delete this branch** icon.

After you have created a Branch, you can clone the files to your local repo, update them on your computer, and then check in the changes from the GitHub Desktop. If you are working with Markdown or other text files, you can edit them in GitHub, and then check in the changes online.

#### To commit changes by using GitHub Desktop:

1. Open GitHub Desktop.
2. Select the repo that contains your changes, and then click **Changes**.
3. Select the changes that you want to commit, and then in the **Summary** text box, write a short description of the change.
4. In the **Description** text box, write a more-detailed description of the change, if necessary.
5. Click **Commit to master**, and then click **Sync** to push the local changes to the online repo.

#### To edit files and commit changes in the online repo:

1. In Internet Explorer, navigate to the applicable repo on GitHub, and then select the file that you want to edit.
2. Click the **Edit this file** icon.
3. Make your changes in the **Edit file** tab of the webpage, and then click **Preview changes** to view your proposed changes, without committing them.
4. Under **Commit changes**, in the **Update *filename*** text box, enter a short description of the changes.
5. In the **Add an optional extended description...** text box, enter a more detailed description of the change, if necessary, and then click **Commit changes**.

#### To create a pull request:

1. In Internet Explorer, navigate to the applicable repo on GitHub.
2. Click **Branch:*branchname***, and then in the **Branches** list, select the branch for which you want to create a pull request.
3. Click **New pull request**, and then on the **Open a pull request** page, in the **Title** text box, update the name of the pull request, if necessary.
4. On the **Write** tab, in the **Leave a comment** text box, provide a description of the proposed change, and then click **Create pull request**.

As we noted previously, you also can comment on pull requests and proposed changes (commits) that other users make. When you comment on a commit, you view a source diff of the file, and you then you can comment on specific changes on a line-by-line basis or on the entire commit.

#### To review and comment on a pull request:

1. In Internet Explorer, navigate to the applicable repo on GitHub.
2. Click **Pull requests *n***, where ***n*** is the number of active pull requests.
3. Select the pull request that you want to review, and then on the **Write** tab, in the **Leave a comment** text box, input your comment.
4. Click **Comment**.

#### To review and comment on a commit:

1. In Internet Explorer, navigate to the repo on GitHub.
2. Click ***n* commits**, where ***n*** is the number of commits that have been submitted. If you want to review the latest commit, you can select the title/short description of the commit from file list.
3. In the **source diff** section, select the change on which you want to comment by clicking the plus sign (**+)** that appears when the mouse hovers over the applicable change.
4. On the **Write** tab, in the **Comment** text box, provide your comment.
5. Click **Comment**.

If you wish to provide an overall comment on the commit, under ***n* comments on commit**, where ***n*** is the number of comments submitted, and then under the **Write** tab, in the **Leave a comment** text box, type your comment, and Click **Comment on this commit**.

You also can make suggestions about an overall project, by submitting an Issue or commenting on an existing Issue.

#### To submit an Issue:

1. In Internet Explorer, navigate to the applicable repo on GitHub.
2. Click **Issues**, and then click **New issue**.
3. In the **Title** text box, enter the title for the issue, and then in the **Leave a comment** text box, type a description of the issue or suggestion.
4. Click **Submit new issue**.

#### To review and comment on an existing issue:

1. In Internet Explorer, navigate to the applicable repo on GitHub.
2. Click **Issues**, and then select the title of the issue that you want to review.
3. On the **Issue name** page, on the **Write** tab, in the **Leave a comment** text box, type your comment.
4. Click **Comment**.

Whenever you create an issue, or a comment on a pull request or commit, you also can include other GitHub users or teams into the conversation by performing a **mention** of themin the comment’s body. If you are familiar with Twitter, this feature will look very familiar.

#### To mention a GitHub user in a comment:

1. In Internet Explorer, navigate to the applicable repo on GitHub.
2. Create your comment or issue, as described previously, and then in the **comment** text box, type **@**, followed by the user or team name, within the comment.

**Note:** When you type the **@** symbol, a list appears that contains GitHub users who are collaborators on the applicable project and anyone who is participating in the project’s comments. The list uses autocomplete as you type, so that you can filter the list easily.