# GitHub User Guide for MCTs

One of the issues that MCTs face when teaching courses like 20533C: Implementing Microsoft Azure Infrastructure Solutions is that cloud services like Azure are updated frequently. This means that the labs provided in courses like this also need to be updated frequently to match the service changes. In addition, because of the frequency of the changes and the fact that there might be no notification when these changes are occurring, it can be difficult for the development team to rapidly identify and address any lab changes.

To address these issues, we are publishing the lab instructions and lab files for this course on GitHub to allow for collaboration between the course authors and MCTs to keep the content current with changes in the Azure platform. By using GitHub, MCTs can provide feedback and provide suggestions for changes to the labs. The course authors can then very rapidly update lab steps and lab scripts.

When you prepare to teach this course, you should always ensure that you are using the latest lab steps and scripts. You can do this by downloading the appropriate files from GitHub whenever you prepare to teach this course.

This document is a user guide designed for MCTs who are new to GitHub. This guide provides steps for connecting to GitHub, downloading and printing your course materials, updating the scripts used by students in the labs, as well as showing you how you can contribute to keeping this course content current.

### GitHub Terminology

Using GitHub introduces terminology that might be new to you. The following are some terms and concepts throughout this document. For a full list of GitHub terms, see the (GitHub Glossary)[https://help.GitHub.com/articles/GitHub-glossary/]

* **Git** and **GitHub** – Git is an open source change tracking program. GitHub is a site/solution built on top of Git. There are other websites and solutions that use Git as their backend. GitHub is used primarily for open source (public) development projects, and it is free for those projects. If you want to use GitHub for projects that are not open source (private), you must sign up for a paid version.
* **Repo** or **Repository** – Each project in GitHub is in a repository, or Repo for short. A repository contains all of a project’s files, including documentation, and supports revision history. A repository can be either public or private. Also, you can have a **local** copy of the repo on your computer hard drive, or use the **online** repo within GitHub.
* **Markdown** – A text file format for creating documentation. It is text based and very simple to update, making it easy to collaborate. It is rendered by GitHub as HTML.
* **GitHub flavored markdown (GFM)** – There are many variations, or flavors, of Markdown. GitHub version, commonly referred to as GFM, is one of the most common variations of Markdown. For details on GFM and all the ways you can ‘markup’ GFM documents, see [Getting started with writing and formatting on GitHub](https://help.github.com/articles/getting-started-with-writing-and-formatting-on-github/).
* **Fork** – A copy of someone else’s repo that lives under your GitHub account, in comparison to a Branch, which lives in the original repo. See **Branch**.
* **Branch** – A branch is a copy of a repository that lives in the same repository as the original. A branch can be **merged** with the original.
* **Fetch** – Getting a copy of the latest changes from an online repo. A fetch does not **merge** changes.
* **Pull** – Fetching the latest changes from an online repo and **merging** them with any local changes.
* **Merge** – Getting changes from one branch and applying them to another. This includes changes from an online repo and applying the changes to the local version of that repo
* **Pull request** – A set of proposed changes to a repo submitted by a user that can either be accepted or rejected by owners or collaborators of a repo.
* **Push** – Sending (or submitting) your local changes to the online repo.
* **Collaborator** – A GitHub user that has permissions to add, delete, or change the content of a repository.

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

The Microsoft Learning team has created a solution that allows the Microsoft team to regularly publish updated lab and lab answer keys (LAKs) to GitHub. That solution also includes a script and tools to allow you to print the labs and lab answer keys from Microsoft Word .docx files. In order to use this solution, you need to perform the following steps the first time you wish to download and print the lab files:

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

* Pandoc 1.13.2
* PowerShell Community Extensions 3.2.0

Once you have signed up for GitHub and installed 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 Visual Studio, VS Code, or any of the Git command line tools widely available.

**Note** GitHub has both 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, see the GitHub [Hello Word](https://guides.github.com/activities/hello-world/) guide to get started.

## Prerequisites

Using GitHub and the Microsoft Learning courseware lab solution requires the following prerequisites:

## 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

1. In your browser, navigate to [https://GitHub.com/](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 password.
4. In the **Create a password** text box, enter a password that meets the complexity requirements of GitHub.
5. Click **Sign up for GitHub**.
6. In the **Welcome to GitHub** page, make sure that **Unlimited public repositories for free** is selected.
7. Click **Finish sign up**.
8. GitHub sends a confirmation email to the email address you provided. When you receive the email, open it and click **Verify email address**.

### Installing GitHub Desktop

GitHub Desktop provides a graphical user interface (GUI) for GitHub, where you can perform the most common functions. There are some operations that can only be performed from the command line, but none of those are needed for downloading and printing lab files.

#### To install GitHub Desktop

1. In your browser, navigate to [https://desktop.GitHub.com/](https://desktop.github.com/).
2. Click **Download GitHub Desktop**.
3. When the **GitHubSetup.exe** file has downloaded, double-click the file to start the setup or click **Run** if prompted by Internet Explorer.
4. In the **Application Install - Security Warning** dialog, click **Install**.
5. Close GitHub Desktop.

### Installing Pandoc 1.13.2

Pandoc is a tool for converting files in one format to another. It can read many formats including GFM. It is used to output Microsoft Word's .docx format. Pandoc is the tool behind the Microsoft Learning-provided scripts that create Word documents from the Markdown versions of the lab files. If you do not install Pandoc, the document creation script fails.

#### To install Pandox 1.13.2

1. In your browser, navigate to [https://GitHub.com/jgm/pandoc/releases](https://github.com/jgm/pandoc/releases/tag/1.13.2).
2. Scroll to the bottom of the page.
3. Click **pandoc-1.13.2-windows.msi**.
4. When the **pandoc-1.13.2-windows.msi** file has downloaded, double-click the file to start the setup or click **Run** if prompted by Internet Explorer.
5. In the **Pandox 1.13.2 Setup** dialog, review the License Agreement, select **I accept the terms in the License Agreeement**, and click **Next**.
6. 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 3.2.0 is the most current (as of 6/16/2016) version of PSCX. PSCX is used to create the .zip files that contain your .docx files. If you do not have the extensions installed, the document creation script fails.

#### To install PSCX 3.2.0

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 prompted by Internet Explorer.
4. In the **PowerShell Community Extensions 3.2.0 Setup** dialog, review the License Agreement, select **I accept the terms in the License Agreeement**, and click **Install**.
5. In the **User Account Control** dialog, if it appears, click **Yes**.
6. Click **Finish**.

**Important**: After installing both 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. Each repo contains the following folders:

* **Allfiles** - The folder contains any supporting files for the labs. This is the same as the Allfiles folder that would appear on the VHD for student VMs.
* **Build** - The folder contains the Windows PowerShell script and supporting files for creating Word documents from the lab Markdown files. –
* **Instructions** - the folder contains the lab files and lab answer key files, formatted as Markdown.

You need all of these folders in order to print the lab files.

### Downloading the latest course lab materials

In order to build Word documents from the Markdown files, you need to [clone](https://help.github.com/articles/cloning-a-repository/) or get a copy of the repo on your local computer. In order to clone the files, you need to know where on GitHub the course files are located. You can find the files by searching for the course number in GitHub, using the **Search GitHub** search box on the [GitHub home page](http://github.com), or by browsing through the repos under the [Microsoft Learning organization](https://github.com/MicrosoftLearning) page on GitHub.

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

1. In your browser, navigate to the online repo in GitHub. For example, the online repo for course 20532: Developing Microsoft Azure Solutions, is located at <https://GitHub.com/MicrosoftLearning/20532-DevelopingMicrosoftAzureSolutions>.
2. On the repo page, click **Clone or download**.
3. In the **Clone with HTTPS** dialog, click **Open in Desktop**.
4. In the **Internet Explorer** confirmation dialog, click **Allow** (or the equivalent for your browser).
5. Switch to GitHub Desktop.
6. In the **Browse For Folder** dialog, select a folder as the root for the local repo and click **Ok**.

If you plan to clone several repos, you can choose one common folder in this step. A subfolder for each repo is created.

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

After you have cloned the repo the first time, you can simply open GitHub Desktop, select the repository, and click **Sync** to get the latest files.

For more information on syncing your repo, see [Working with your remote repository on GitHub or GitHub Enterprise](https://help.github.com/desktop/guides/contributing/working-with-your-remote-repository-on-github-or-github-enterprise/).

### Printing the lab and LAK files

In order to print the lab and LAK files, they need to be converted to Word documents. Microsoft Learning has provided a Windows PowerShell script to automate the task. In addition to creating Word documents, the script also packages the word documents into .zip files. At the same time, it creates a .zip file, containing supporting files for the labs for when you need to setup the lab environment.

The Windows PowerShell script is located in the folder and is named Pandoc.ps1. The folder also contains template.docx, which the script uses to provide the formatting 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 you cloned. For example:

..\Documents\GitHub\20532-DevelopingMicrosoftAzureSolutions\Build

1. Right-click the file **pandoc.ps1**, and click **Run with PowerShell**.
2. In the **Windows PowerShell** window, if you receive an **Execution Policy Change** prompt, type **Y** and 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,

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

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

To avoid 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, scripts that you run have the power to make disruptive things to your computer.

#### To print the lab files

* Open the lab files in Microsoft Word to print them.

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

You can receive notifications whenever there are updates to a GitHub repo. You have several ways to sign-up for notifications. It should not be a surprise that many of the ways you can receive notifications are also related to many of the ways you can collaborate on a project. To receive notifications, you can:

* **Watch repositories** - Watching a repository subscribes you to notifications for any new **pull requests** or **issues** that are created for the repository. You automatically watch any repository you create or where you are a collaborator.
* **Pull request** - When you create a pull request, proposing the owners of a repo accept a change you have made, 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 on someone else's Pull request, you automatically subscribe to the forum related to the comment, or you can subscribe to the forum.
* **Issues** - An issue is a suggestion, question, or request that pertains to the repository. Each issue has its own discussion. You can subscribe to issues or you are automatically subscribed to Issues you create.
* **Mentions** - When another user mentions you in a conversation, using your GitHub user name (@\_\_username\_\_) you are automatically subscribed to the discussion.

You can modify how and when you receive notifications. You can also 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** the repo. You can do that, even if you have not Cloned a local copy.

#### To Watch a repository

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

#### To unWatch a repository

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

You can also choose the **Ignoring** option under the **Watch** drop-down list. However, this means you receive **no** notifications, even if someone @mentions you in a discussion.

### Suggesting changes and collaborating on a repo

Using GitHub provides an easy-to-use method of collaborating with Microsoft Learning on the courses you are interested in.

You can modify your own copy of the lab materials and submit the change to Microsoft Learning so that they can incorporate your updates:

* If you find a mistake in a lab,
* the UI has changed since the lab was created,
* or, if you think that the lab can be improved.

You do this by branching the repo, making updates in your branch, and then submitting a Pull request to the main (master) branch. Microsoft Learning, other MCTs, and other GitHub users, can review your changes and comment on them.

You can review and comment on the changes other users make. Microsoft Learning can then approve and merge changes into the master branch. When they do, any user who is watching the repo is notified of the change.

#### To create a repo Branch

1. In Internet Explorer, navigate to the repo on GitHub.
2. Click **Branch :branchname**, and from the **Branches** list, select the branch you want to copy.

If there is only one branch, the Branch drop-down shows **Branch: master** and the only Branch available is **master**.

1. In the blank text box, type the name of the branch you want to create.
2. 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 you want to delete, click **Delete this branch** icon.

After you have created a Branch, you can clone the files to your local repo and update them on your computer and commit (check in) the changes from within GitHub Desktop, or in the case of Markdown or other text files, you can edit them within GitHub and commit the changes online.

#### To commit changes using GitHub Desktop

1. Open GitHub Desktop.
2. Select the repo containing your changes.
3. Click **Changes**.
4. Select the changes you want to commit.
5. In the **Summary** text box, write a **short** description of the change.
6. In the **Description** text box, provide a more detailed description of the change, if needed.
7. Click **Commit to master**.
8. 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 repo on GitHub.
2. Select the file you want to edit.
3. Click the **Edit this file** icon.
4. Make your changes in the **Edit file** tab of the webpage.
5. Click **Preview changes** to view your proposed changes without committing them.
6. Under **Commit changes**, in the **Update filename** text box, enter a short description of the change.
7. In the **Add an optional extended description...** text box, enter a more detailed description of the change, if needed.
8. Click **Commit changes**.

#### To create a pull request

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

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

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

1. In Internet Explorer, navigate to the repo on GitHub.
2. Click **Pull requests n**, where **n** is the number of active pull requests.
3. Select the pull request you want to review.
4. On the **Write** tab, in the **Leave a comment** text box, provide your comment.
5. 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.

1. In the **source diff** section, select the change you want to comment on by clicking the **+** that appears when the mouse hovers over the change.
2. On the **Write** tab, in the **Comment** text box, provide your comment.
3. Click **Comment**.
4. If you wish to provide an overall comment on the commit:
5. Under **n comments on commit**, where **n** is the number of comments submitted, under the **Write** tab, in the **Leave a comment** text box, type your comment.
6. Click **Comment on this commit**.

You can also 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 repo on GitHub.
2. Click **Issues**.
3. Click **New issue**.
4. In the **Title** text box, enter the title for the issue.
5. In the **Leave a comment** text box, enter a description of the issue or suggestion.
6. Click **Submit new issue**.

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

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

Whenever you create an issue, or a comment on a pull request or commit, you can also bring other GitHub users or teams into the conversation. You do this by **mentioning** them in the body of the comment. 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 repo on GitHub.
2. Create your comment or issue as described previously.
3. In the comment text box, type **@**, followed by the user or team name, within the comment.

When you type the **@** symbol, a list appears that contains GitHub users who are collaborators on the projector and anyone who is participating in comments on the project. The list uses autocomplete as you type to filter the list.