Domain 1: Introduction to Git and GitHub – Understand the basics of Git

1. Explain what Git is and why it is used

- Git is a Distributed Version Control System, or DVCS.
    - Distributed:
        - users do not require connection to a centralized version (though they can use a centralized version).
        - Instead, they can have their own copy.
    - Version control system: It tracks the files' history:
        - Who, what, why and when were changes made.
        - You can also add comments and issues.
- According to Wikipedia, there is no definitive answer as to why it is called "Git".
- You can issue commands:
    - Either through a command line, or
    - Using an application which includes a Graphical User Interface (GUI).
- It allows multiple people to work on a single project.
    - Not just in a single building or city, but also internationally.
- You can restore earlier versions if needed.
- Users can either work from a centralized version, or their own version.
    - They can extract parts of it, and then merge them back in.

2. Describe the basic Git workflow

- A basic Git workflow on your own computer includes the following:
    - If editing the main/master branch, add/edit files and commit (save) them.
    - Create a branch for your own use.
        - This is somewhere you can make changes without it currently affecting the main branch.
        - This will include all of the files and folders in that branch.
        - Ideally, you would have a separate branch of each change you are going to make.
            - When the changes are added to the main branch, you can add each different messages each time.
    - Make any edits to your own branch.
        - You can:
            - Create new files,
            - Edit existing files,
            - Rename files,
            - Move files to a new location, or

        - Delete files.
    - Merge your changes back to the main branch.
        - Don't merge everything that you haven't changed, as others may have changed them.
        - This allows Git to keep track of all changes, so the main branch has the latest version.
- Additionally, others may be making their own changes to the centralized branch, either online or on their computer. If this is the case:
    - You can pull the latest changes from the centralized version (remote -> own) to update your version.
    - You can then push back your own changes, maybe using a Push Request (PR).
    - This may need approval before the push can be finished.
- Local computer Git workflow
- GitHub workflow
- GitHub workflow

3. Understand the concept of repositories

- A repository contains:
    - All of the files and folders,
    - Every file's revision history.
- The advantages of using a repository are:
    - It allows you to manage your code over the project timeline.
    - Users can review your code, and add comments.
- Files are modified using commits.
    - This shows that the file contains at a particular point in time.
- Users can download the entirety of a repository.
- Repositories can either be public or private.
- It uses Branches, which contain folders and subfolders.
    - The original branch is called "main" in GitHub and "master" in Git.
- Repositories can be:
    - Cloned (duplicated),
    - Branched (part of the repository duplicated for separate development, without affecting the main branch), and
    - Merged (incorporates changes from one branch into another branch in the same repository).
- Files can be:
    - Committed (saved), and
    - Compared (different versions of code).

4. Explain the difference between local and remote repositories

- A remote repository is where your code is centrally stored.
    - This could be on GitHub or similar servers.
    - Changes are tracked.
    - You need to be connected to that remote repository to access it.
    - Multiple people can access it, depending on whether they have permission.
- A local repository is located on your own computer.
    - Changes are also tracked.
    - You do not need network access to work on this repository.
    - Generally, only you can access your repository.
        - If other users have access to your computer, they can also access it.
    - It can include your own commits and local history.
    - You can create a local repository by cloning a remote repository.
    - To send changes from a local repository to a remote repository, changes need to be pushed.

Domain 1: Introduction to Git and GitHub – Work with Git commands

5. Initialize a Git repository

- First of all, download and install Git from https://git-scm.com/downloads
    - It is available for Windows, Mac and Linux/Unix.
    - The latest version of the Windows download requires a 64-bit version.
        - The last full version of the Windows download which can be installed on a 32-bit computer is version 2.48.1 from February 2025.
        - This is because of lack of support of one of its components.
- Open the relevant application.
    - In Windows, open Git Bash (this is now an application on your computer).
        - You can increase the font by using Ctrl and plus.
        - You can decrease the font by using Ctrl and minus.
    - On Mac and Linux, open Terminal.
- You should then navigate to the appropriate folder.
    - In Windows, if that was the G folder on the C drive, then you can use:
        - cd c:/g
        - cd stands for “change directory”.
    - On a Mac, use:
        - //Users/user/g
        - cd can also work on a Mac.
    - On Linux, use

        - //home/user/g
- To make a folder, use mkdir (not md).
- Initialize this folder as a Git repository.
    - To do this, use: git init
        - This will not overwrite existing files.
        - It will add a hidden .git directory which allows for files to be tracked.
        - It has the following files:
            - Config – contains repository-specific Git settings,
            - Description – a description which is not used by Git itself, but by other programs,
            - HEAD – The current branch
        - It has the following subfolders.
            - Hooks – pre-commit and pre-push Git scripts
            - Info – contains a list of files to Git to ignore.
            - Objects – these are commits, trees, blobs and tags.
            - Refs/heads – this contains one file per local branch,
            - Refs/tags – this contains one tag (tracked point) per commit
            - Template or templates – this may include template files.
    - The current default is "master". However, you may want to name the main branch:
        - In Git 2.28.0 or later, use:
            - git init -b main
        - In Git 2.27.1 or earlier, use:
            - git init && git symbolic-ref HEAD refs/heads/main

6. Clone a repository

- You can use:
    - -b or --branch: clones a specific branch (instead of the main/master default)
- Cloning a repository is:
    - copying it into a newly created directory,
    - Creating and check out an initial branch.
- After the clone:
    - git fetch (no arguments) will update all the relevant remote-tracking branches with the latest changes, and
    - git pull (no arguments) will merge the remote master (main) branch into the current master (main) branch.

7. Add and commit changes

- Git uses an index which contains a snapshot of the "working tree".
    - Both are used interchangeably.

- After creating the repository, you should update the index with the existing files.
    - This is also called “staging”.
    - Enter: git add .
    - The “.” refers to the current directory and its subdirectories.
        - Instead of the “.”, you could also use a directory (DirName/FileName), or an extension (*.doc)
            - For example: git add "Documentation/*.txt" (the quotation marks should be used if using a *)
    - This command adds all changes in the current directory and subdirectories. Changes are:
        - New files,
        - Modified files,
        - Deleted files (if they were being tracked).
    - You can run this as many times as you want before committing changes.
- You can also add options such as:
    - -h = shows the options (you can use this with other commands). It is shown in pages.
        - Press space to see the next page.
        - To end the output, press q
        - Alternatively, you can use --no-pager
    - -n or --dry-run = shows whether the files exist or would be ignored. Does not actually add the files.
    - -v or --verbose = verbose
        - Says (for example “add ‘name of file’), instead of giving no output.
    - -f or --force = add previously ignored files
        - Files can be previously ignored, and added to a file called “.gitignore”.
    - -u or --update = update files where they have been changed (modified or deleted), but not added
    - -A (note: capital A) or --all = all files in the entire repo, not just in the current directory and subfolders.
        - It adds all files – modified, untracked and deleted.
        - So you can use git add -A
    - --ignore-errors = a single error does not stop the operation.
- To remove files from the index, use git rm (filename)
    - You can specify multiple files, either by:
        - *.tmp, or
        - File1.txt File2.txt

    - If a file has modified since last being commit, then it will not be removed from the index unless you use -f or --force.
    - To remove the output, use -q or --quiet.
- To move or rename a file, use git mv
    - It needs to be under source control for this to work.
    - This actually moves the file in your filing system.
    - -f or --force allows the file to be renamed or moved, even if the file already exists in the destination.
    - -n or --dry-run does a test as to what would happen. It does not move/rename files.
    - -v or --verbose – shows a longer output.
- To identify what changes have been made to files, you can use git diff.
    - This shows the difference between the current files and the index.
- To identify what files have been added, you can use git status. Options include:
    - -s or --short = short-format. The first character includes:
        - M = has been modified, ready to commit
        - R = renamed
        - ? = untracked
        - ! = ignored
    - The second character includes:
        - M = has been modified, not ready to commit
        - D = deleted, not ready to commit
        - ? = untracked
    - Note – a file can be MM, when one version is ready to commit, but it has since been modified.
    - -v or --verbose. Shows changes to the index.
- Prior to doing your first commit, you need to add your user identity.
    - git config --global user.email "your@email.com" AND
    - git config --global user.name "Your Name"
    - If you only want to add the user identity to a single repository, then omit the "--global".
        - You can also use --system and --local.
- To commit changes, use
    - git commit
    - This then opens an editor (for example, a Vim screen), where you can add a message after the #s.
    - Press:
        - i/a to insert before/after the cursor

        - o/O to insert a new line below/above
    - Then press Escape (you should see the – INSERT – disappear from the bottom)
    - Then enter :wq for write (save) and quit.
        - You can also use :x or ZZ
    - Alternatively, enter :q! to quit without saving the commit message.
        - You can also use ZQ
- You can use the following options:
    - -a or --all: commits all files which have modified and deleted.
    - -m or --message: add a message as part of the commit.
        - You can combine these into -am
    - -v or --verbose: shows a diff as part of the commit.
    - --amend: changes the most recent commit (either the message or the content)
- Once changes have been committed, you can see the various commits that have been done using the git log command.
    - git log
        - Press space to see the next page.
        - To end the output, press q
    - git log "Second File.txt“
    - It shows:
        - the commit hash,
        - Author,
        - Date, and
        - Message
- You can use the commit hash to show a prior or current version of the file:
    - git show #CommitHash#:"Second File.txt“
- For the current version, you can use:
    - git show HEAD:"Second File.txt"

8. Push and pull changes

- git push pushes commits from your local branch to a remote repository.
    - git push RemoteRepository LocalBranchName
    - It sends the branches which have the same name locally and remote.
- To push it to a different branch add a colon and then new Branch Name
    - git push RemoteRepository LocalBranchName:RemoteBranchName
- You can also use it to delete a remote branch, adding a colon before the branch name.
    - git push RemoteRepository :BranchName

9. Understand branching and merging

- A branch contains a separate stream of work (or version).
- Changes to a branch will be isolated to that branch – it will not affect other branches (including the "master" or "main" branch).
- You could have several branches of your repository.
    - Each branch would have a different function.
- To make a copy of a main repository, you "fork" the repository to create a branch.
- To list the branches, use:
    - git branch --list or git branch
- To create a new branch, use:
    - git branch NewBranchName
    - This will not switch to the new branch.
- To switch to a branch, use:
    - git switch NewBranchName
- To merge a branch:
    - Switch into the branch you want to merge into (not from).
    - Then you can use git merge NewBranchName
- Once a branch has been finished with, you can delete it, using
    - git branch -d NewBranchName, if all the changes have been merged, or
    - git branch -D NewBranchName, if the branch was not merged and you just want it deleted.

Domain 1: Introduction to Git and GitHub – Navigate GitHub

10. Create a GitHub account

- To create a GitHub account:
    - Go to https://www.github.com
    - Enter your email address and click Sign up.
    - Enter a new user name and password.
    - You can then verify your account.
- Two factor authentication (2FA) will be needed, either immediately or within 45 days of notification.
    - If you don't do so, then your account access will be limited.
- You can use:
    - A Time-based One-Time Password (TOTP) app such as 1Password, Authy and Microsoft Authenticator (but you can use other TOTP apps) which can generate passkeys.
        - These apps can also use technologies such as Windows Hello, Face ID or Touch ID to identify you.
    - Security keys (hardware devices),

    - GitHub Mobile app,
    - A mobile phone to receive a SMS or Text message (not recommended, as they do not provide the same level of protection).
- You can change your authentication method by:
    - Clicking on your profile picture,
    - Click on "Settings" in the new menu,
    - Going to "Password and authentication" on the left-hand side.
    - If you haven't previously used 2FA, then you can click on "Enable two-factor authentication" to start setting it up.
        - If you want to set it up later, you can do so in Settings – Password and authentication.
- You can then create repositories.

11. Create and manage repositories on GitHub

- Once you are logged in, you can then create a repository by clicking "Create repository".
    - The full name of the repository will be OwnerName/RepositoryName.
    - Because of this, you can use Repository names that others have used, because they will have different owners.
    - You can add a description (optional).
    - You should choose whether it is public or private.
        - Public means that everyone can see the repository.
        - Private means that you can choose who can see the repository.
        - Regardless, you can choose who can commit.
    - You can automatically add:
        - a README file, which can include a long description,
        - a .gitignore template. This shows the files not to track.
        - a license. This shows to others what they can do with your files.
- You can also create repositories using repository templates (see topic 16).
- Note - if this is an organization accounts (as opposed to a personal account), the following roles can be created:
    - Owner – can do anything in the organization. Ideally, there should be at least 2 owners.
    - Billing manager – view/edit billing information, and manage sponsorship. Doesn't have access to data.
    - Member – can create repositories and projects.

12. Understand the GitHub interface

- The GitHub interface was released in 2023.
- The first row is a site-wide menu "Global Navigation Bar", also known as the Global Header or Top Navigation Bar.
- On the left there is a hamburger icon. Clicking this opens a menu, with:

- Home.
    - This shows:
        - Copilot – you can ask questions and have conversations. The free plan gives you:
            - 50 Copilot Chat messages a month, and
            - 2,000 code suggestions a month.
        - Getting started
        - GitHub documentation
        - Recommendations, and
        - On the left-hand side “Top repositories”.
    - You can also get to Home by clicking on the cat icon (called “Mona”) in the top-left hand corner.
- Issues (see topic 24)
- Pull requests (see topic 13)
- Projects (see topics 25-26)
- Discussions (see topic 43)
- Codespaces
- Copilot, Microsoft’s version of AI
- Explore look at other people’s repositories, including:
    - Explore tab, which includes:
        - Topics that you have “starred” (added a star), and
        - Starred repositories.
    - Topics,
    - Trending,
    - Collections,
    - Events, and
    - GitHub Sponsors.
- Marketplace (where you can expand the functionality of GitHub), and
- Repositories.

- Looking at a repository, in the Global Navigation Bar you can see at the top:
    - The user name and repository name
    - A search bar
    - Chat/Open with Copilot
    - A + sign, together with a dropdown containing:
        - New/Import repository,
        - New Codespace,

        - New gist, and
        - New organization.
    - Your issues (a dot and a circle),
    - Your Pull Requests,
    - Your notifications, and
    - Your profile, including:
        - Your user account (together with two arrows for switching account),
        - Set status,
        - Your profile, repositories, Copilot, projects, stars, gists, organizations, enterprises and sponsors
        - Try Enterprise,
        - Feature preview,
        - Settings,
        - Links to GitHub Website, Docs, Support and Community, and
        - Sign out.
- Underneath the top bar, you will see the Repository Menu Bar for the current repository (also known as the Repository Navigation Menu or the Repo Tab Bar). There are links to:
    - Code (so you can see the files contained in the repository),
    - Issues,
    - Pull requests,
    - Actions,
    - Projects,
    - Wiki,
    - Security,
    - Insights and
    - Repository Settings (see topic 14)
- Note – the next two bars will disappear if you start looking at folders. To display them again, click on “Code” in the second top bar.
- In the third row from the top (in the Code tab) is the Repository Header (also known as the Repository Title Bar).
- You can see options for the current repository, which may include (depending on your repository settings):
    - The repository name
    - Whether the repository is public/private,
    - Pin the repository to your profile,
    - Watch menu – whether to receive notifications,
    - Forks, and

    - Starring repository, together with adding the repository to a list.
    - Depending on your repository settings, you may see additional items, such as "Use this template", or fewer items.
- Underneath, you can see the Branch and File Controls Bar:
    - The current branch (you can switch branches and tags)
    - Clickable links to the number of branches and tags.
    - A search bar for looking for files.
    - A "Add file" or + icon for adding files.
    - A "Code" button which gives:
        - The URL for cloning a repository, including opening with GitHub Desktop and Downloading a zip for the repository.
        - Codespaces.
- Underneath, you can see:
    - The files, together with the latest commit message and the time of the current commit. You can click on a file to open it.
    - The text of any README.md file and license. You can click on the icon to edit it.
- On the right-hand side, you can see:
    - The About information of the repository,
    - A wheel icon which opens the settings, allowing you to edit:
        - Description,
        - Website,
        - Topics,
        - Whether releases, packages and deployments to be included in the home page.
    - Releases and Packages.
- At bottom, there are links to:
    - GitHub Terms of Service,
    - GitHub General Privacy Statement,
    - GitHub Advanced Security, which contains two security add-ons,
    - GitHub Status (whether GitHub is online),
    - GitHub Documentation, and
    - GitHub Support.

13. Use GitHub issues and pull requests

- For pull requests, see topics 21 and 22.
- For issues: see topic 24.

Domain 2: Working with GitHub Repositories – Manage repository settings

14. Configure repository settings

- To access the repository settings, Owners can click on the "Settings" wheel icon (second row from the top which starts "Code").
    - Owners only – not collaborators or other users.
- The general tab includes:
    - A general section, includes:
        - The repository name, which you can change.
        - Whether to allow template to be used in repositories, so you can create new repositories with the same files and folders (see topic 16).
        - Whether to require contributors to confirm that their commits (saves) comply the terms for the repository.
    - The default branch – this is generally "main" on GitHub.
    - For public repositories only : An image for your repository's social media preview (minimum 640 x 320 pixels).
    - Whether to show tabs for:
        - Wikis – and whether only collaborators should edit it,
        - Issues (see topics 13 and 24), together with issue templates,
        - Sponsorships – how others can contribute money to your repository.
        - Discussions, and
        - Projects,

      together with whether it should preserved in the GitHub Archive program (for public repositories only).
    - Options regarding pull requests, archives, pushes (public repositories only) and issues.
    - The Danger Zone includes:
        - Changing repository visibility (public/private)
            - Note – your organization may have restricted who can change the repository visibility.
            - Any existing Forks will be detached (but kept with the same visibility).
            - If you change a repository from public to private:
                - advanced Security features may stop working, and
                - Stars and watchers will be removed.
        - Transferring ownership to another user or organization,
        - Archive the repository, so that it is read-only. This can be reversed.
        - Delete this repository. This cannot be reversed.

- There are options on the left-hand side for:
    - Access: Collaborators and moderation options,
    - Code and automation: Branches, tags, rules, actions, models, webhooks, Copilot, environments, codespaces and pages.
    - Security: Advanced Security, deploy keys and Secrets and variables
    - Integrations with GitHub Apps and who is notified by email when push events are triggered.

15. Set up repository permissions

- Only collaborators and owners can write to a repository.
- If a repository is public, everyone can read it.
    - If it is private, only collaborators can read it.
- Users need admin access to set up repository permissions.
- To set up permissions:
    - go to the Repository Settings, then
    - Collaborators (or Collaborators and Teams for an organization) on the left-hand side.
    - Under “Manage access”, click on “Add people” (or “Add teams”).
- You can then enter usernames and invite users to be collaborators.
    - They will receive a notification, and that user can “Accept invitation” or “Decline invitation”.
    - If accepted, the owner will be able to see:
        - The user’s public profile information,
        - Country of request origin,
        - IP address,
        - Certain activity and access level for the repository.
- Once collaborators have been added, they can be removed by clicking on the Trash icon.
- For public repositories, in Repository Settings – Access – Moderation Options, Owners (not contributors) also have the following options:
    - Stop New users from interacting (commenting, open issues or create pull requests) with the repository,
    - Stop New users and users who have been previously committed to the main branch, or
    - Stop everyone except owners and collaborators from interacting with the repository.
- These are meant to be temporary limitations only, and can be set for 1 or 3 days, 1 week, or 1 or 6 months.
- Owners (not collaborators) can do the following:
    - Invite collaborators,
    - Change the visibility to private/public,

    - Limit interactions,
    - Rename a branch, including the default "main" branch,
    - Merge a pull request on a protected branch,
    - Archive or Delete the repository,
    - Manage topics (and a few more).
- Both owners and collaborators can do the following (and more):
    - Fork the repository,
    - Rename branches (but not the default branch),
    - Create/edit/delete comments on commits, pull requests and issues,
    - Create/assign/close/re-open issues
    - Manage labels and milestones for issues/pull requests,
    - Mark an issue/pull request as a duplicate,
    - Create/merge/close pull requests,
    - Enable/disable auto-merge for a pull request,
    - Apply suggested changes to pull requests,
    - Create a pull requests from a fork,
    - Submit a review on a pull request that affects the mergeability of the pull request,
    - Create/edit releases/a wiki,
    - Act as a code owner,
    - Publish, view or install packages,
    - Remove themselves as collaborators.

16. Use repository templates

- Users with admin permissions for a particular repository can make it a template.
- This needs to be enabled in the Repository Settings – General – General – Template repository.
- Once enabled, anyone with access to the repository can create a new repository with the folders and files in the default "main" branch.
    - Note – it will not include any files stored using Git LFS (Large File Storage).
        - This generally includes audio samples, videos, datasets and some graphics.
    - Templates optionally can include the other repository branches.
- You can then click on "Use this template" and "Create a new repository".
    - You need to give the new Repository a name.
    - You can choose to copy all branches, and not just the default "main" branch.
    - You can optionally give it a description, and
    - You specify whether it is public or private.

Domain 2: Working with GitHub Repositories – Work with files in a repository

17. Add, edit, and delete files

- If you have write access to a repository, you can create new files.
    - If you only have read access, then GitHub can fork the project to your account, and then send a pull request to the original repository.
    - It used to be that file names created on GitHub can contain numbers, letters and hyphens only. However, now you can use symbols and spaces as well.
- To add a new file, click on "+ Add file", and then either select:
    - Create new file, or Upload files.
- If creating a new file:
    - Enter the file name – this can include subfolders, using the / symbol as part of the file name.
        - This is only way on the GitHub interface to directly add folders.
    - Edit the file
        - The three drop-downs are:
            - What happens if you press Tab. Do spaces get inserted, or tabs?
            - If spaces, then how many.
            - What is the Word wrap (when the text gets too long for a single line)? Is there "No wrap" or a "Soft wrap".
                - Note: with a soft wrap, the line numbers do not increment when the line is wrapped.
    - You can click "Preview" to look at the file, and then either:
        - Cancel changes, or
        - Commit changes. You should add:
            - A short commit message which shows what you have done.
            - An extended description (optional).
            - You can commit:
                - Either to the current branch, or
                - Create a new branch and start a pull request (see topics 21-22).
- Note – you should not add, commit or push any sensitive information, including personal data, or login information.
- To edit an existing file:
    - Navigate to the correct folder.
    - Click on the file.
    - Click on the pencil icon.

        - You can also click on the drop-down next to the pencil icon to open it in GitHub Desktop.
    - Make your changes.
    - You can then preview, cancel changes, or commit changes as before.
        - You can also create a new branch and start a pull request (see topics21-22).
- To move a file to a new location, and/or to rename an existing file:
    - Open it for editing.
    - Change the file name.
        - You can use the / symbol to move it into a subfolder.
        - To move it up a folder (into a parent folder):
            - enter ../ at the start of the file name, or
            - press Backspace at the beginning of the file name. This effectively removes the hidden / symbol, and merges the directory name with the file name.
    - Either commit or cancel the changes.
- To delete a file or folder:
    - Open the file or folder.
    - Click on the … in the top-right hand corner, and go to Delete file.
    - Then either Cancel or Commit the change as before.
        - You can also create a new branch and start a pull request (see topics21-22).

18. Understand file versioning

- When you commit changes that include a particular file, GitHub creates a new snapshot of that file's contents as part of the commit.
    - If a file hasn't changed, then GitHub uses a reference to that existing content, instead of replicating the snapshot.
- To see when changes were made:
    - Open the file.
    - Open the Blame view.
    - You will see who committed the final version for each line (who is to "blame").
    - For more recent versions, you can click on the version icon to see the Blame prior to that commit.
- To see the history for a file:
    - Open the file.
    - Click on "History" (top-right hand corner).
- You will see the commits which affect this file (latest first).
    - Together with the commit message.

        - The commits will have a 7 hexadecimal characters as a reference (numbers and letters from A to F). This is the first part of a SHA.
            - This is a unique ID or Secure Hash Algorithm (SHA-1).
            - The full SHA is 40 characters.
- You can then:
    - Filter for a user or a date range.
    - Click on an individual commit to see the changes.
    - Copy the full 40-character SHA
    - View the file after that commit, and
    - Browse the repository after that commit.
- If you click on an individual commit, you can see what files have been changed.
- For a file, you will see:
    - Lines which have been deleted (or changed) in red with a minus sign.
    - Lines which have been added (or changed) in green with a plus sign.
- If you click on the Wheel (settings) sign, you can:
    - Change the view from "Unified" (all changes in one view) to "Split" (the "before" and "after" shown in two separate panes).
    - Minimize comments
    - Hide whitespace and
    - Compact line height.
- You can also click on:
    - The document to preview this version of the file, or
    - The … to:
        - View the file. The file read-only, but you can:
            - copy or download the file, or
            - If you were on a branch, make/propose changes.
        - Ask Copilot about the differences ("diff") between these two versions of the file.
- You can also add comments.

19. Use GitHub Desktop for file management

- You can download GitHub Desktop on:
    - macOS 11.0+, and
    - Windows 10 or later, 64-bit.
- Go to https://github.com/apps/desktop to download it.
- Once installed and run, you can sign into your GitHub account by:
    - going to File – Options

- In the Accounts tab, click on "Sign into GitHub.com" (or "Sign into GitHub Enterprise" if you are using that plan).
- Then click "Continue with browser", sign in using your browser, and click "Authorize desktop". This will allow GitHub Desktop to:
    - Read and write to your public and private repositories.
    - Read and write all user data, and
    - Remove or edit GitHub Action Workflow files.

- To create a local repository on your computer, you can:
    - Fork (see topic 20) or Clone a remote repository on GitHub.
    - Create a new Repository on your local drive, or
    - Add an existing Repository from your local drive.
- If you select an existing repository and click Clone:
    - You can use the details in the URL tab, or
    - You can go to the GitHub.com (or Enterprise) table, and select a repository.
    - Either way, you can also adjust your local path.
- You can then open the repository in Visual Studio Code, Explorer, or GitHub.
- On the left-hand side, you can see changes (commits) and history, and
- On the top, you can:
    - See/change the current repository,
    - See/change the current branch (and pull requests)
    - See the time of the last fetch from the remote repository.

Domain 3: Collaboration Features – Collaborate using GitHub

20. Fork repositories

- You can use forks in two ways:
    - Instead of altering files directly, you can:
        - Fork the repository, to make a new branch,
        - Edit the files, and then
        - Create a pull request to the original repository's owner.
        - You could then delete the forked branch.
    - Use the original as the basis for your own repository.
- If you fork a repository, you can make draft changes without then changing the original (or "upstream") repository.
    - You can fork:
        - a public repository, or
        - a private repository if the owner permits forking
    - to

        - your personal account, or
        - an organization where you can create repositories. Note: Forking a private repository to an organization requires something other than GitHub Free.
    - Note: you cannot fork a repository that you own to your personal account.
    - These changes are done in a different branch.
        - This shows work separate from the default branch.
        - Changes can be incorporated into the default branch by using a pull request (see topics 21 and 22).
- The difference between duplicating and forking include:
    - Forks can use a pull request to the upstream repository. Cloning uses pushes to change the original.
    - You can bring in changes downstream to your fork using synchronization.
    - Forks inherit the upstream repository's restrictions.
- Both forks and clones can have their own members, branches, pull requests, tags/labels etc.
- To create a fork in GitHub:
    - Go to the repository,
    - Click on Fork – or click on the drop-down next to fork and select "Create a new fork".
    - Alternatively, try to edit a file in a repository where you do not have write permissions.
    - You will see under the forked name "forked from" and a hyperlink back to the upstream repository.
- You can then add, edit or delete files.
    - After doing these changes, you may see that "this branch is 1 commit ahead of" the upstream repository.
    - These edits will not be shown in the default fork.
- Forks cannot easily be made directly in GitHub Desktop.
    - Instead, you should clone a repository that you do not have write access to first.
    - Then try to push a change to the upstream repository.
    - You will then be prompted to fork the repository.
    - You can use the fork:
        - To contribute to the parent project, or
        - For your own purposes.

21. Create and manage pull requests

- A pull request or PR (not to be confused with a pull command) is a request to merge changes from one branch to another.
- Once created, collaborators can review and discuss the changes, and then decide whether to commit them.

- In GitHub, if you have made changes:
    - you can click on:
        - “Compare & pull request”, or
        - Contribute – Open Pull Request, or
        - Go to the Pull requests tab and click on “New pull request”.
    - You can then review where the repository is going from and to, and whether it can be automatically merged.
        - The repository/branch that it is going from is called the “head repository/branch”.
        - The repository/branch where it is going to is called the “base repository/branch”.
    - You can then add a title and description.
        - The description can mention contributors or teams in the upstream repository using the @ symbol.
        - You can use markdown, such as:
            - #, ## and ### for Headings 1-3.
            - *words* or _words_ for italic
            - **words** or __words__ (two underscores) for bold
            - <ins>underlined</ins> for underlined text.
            - ***words*** for bold and italic
            - ~words~ or ~~words~~ for strikethrough.
            - _{words} for subscript
            - ^{words} for superscript, and
            - > words to quote text
            - `words` (using a backtick) for code
            - [Microsoft] (www.Microsoft.com) for hyperlinks.
            - - words for bullet
        - However, you also use the icons to insert some of these as well.
        - You can also add HTML code inline (in the document)
            - For example: My link
        - If you want to use symbols such as * and _, instead of using them as bold or italic, then you need to prefix it with a \ backslash symbol.
        - You can also attach files, by clicking on the paperclip or going to “Paste, drop, or click to add files”.
    - If you have write access to the repository, you can also request a reviewer(s).
    - You can then:
        - Create a pull request, or
        - Create a draft pull request.

        - Draft PRs cannot be merged until marked ready for review.
        - Code owners are not automatically requested to review draft PRs.
    - You can subsequently change a draft PR to a PR, or a PR to a draft PR.
- To manage PRs:
    - Go to the "Pull requests" tab.
    - You can then see the PRs.
    - If you click on one of the PRs, there are the following tabs:
        - Conversation. This starts with your initial PR.
        - Commits. The commits which form part of the PR (and so which haven't actually been committed to the upstream repository and branch).
        - Any automated checks using GitHub Actions, and
        - The files which have been changed. Here you can click on the … and:
            - view, edit or delete the file.
            - Ask Copilot about the differences, or
            - Open in GitHub Desktop.
    - You can also:
        - unsubscribe – stop receiving notifications about this thread – or subscribe.
        - Add additional comments
        - Click on "Close pull request" to cancel it.
            - It can be subsequently reopened if needed.
- In GitHub Desktop:
    - Clone the repository that you don't have write access to.
    - Make changes in that repository.
    - Click on "Push origin".

22. Review and merge pull requests

- In the upstream branch, you can also click on the "Pull requests" tab.
- You will then see a list of the PRs.
    - You can also filter them by author, label, projects, milestones, reviews and assignee.
    - You can additionally filter them by:
        - Open issues and pull requests,
        - Your issues/pull requests,
        - Everything assigned to/mentioning you.
    - You can also sort them by date, number of comments, recently updated, best match, and most reactions.
- You can hover or click on a PR to see more information.

- Having clicked on it, you have the same things as when request a PR request (conversation, commits, checks and file changes).
- on the right-hand side you can manage:
    - Reviewers (up to 15 reviewers). These are people who should look at the code and then comment, approve, or request additional changes.
    - Assignees (up to 10 people, or 1 person for private repositories on the free plan). These are people who are responsible for the PR (the owner, or someone who should complete it), but not asked to comment on it.
    - Labels,
    - Projects,
    - Milestones, and
    - You may be able to link this pull request to an issue(s).
- If you have been added as a reviewer, you will see at the top of the screen "This pull request is waiting on your review".
    - See topic 30 for more.
- You can also scroll down and:
    - Add a comment, and click on "Close pull request" or "Comment" (which doesn't close it).
    - Click on "Close pull request".
    - Click on "Merge pull request". The dropdown may show ways to merge:
        - Create a merge commit. This adds all of the PR commits into the original branch using a merge commit.
        - Squash and merge. This remerges all the PR commits into one commit, and writes this commit into the original branch.
            - This new commit will have a default message, which is editable.
            - It will also have all of the original commit messages in the description.
        - Rebase and merge. This rewrites all of the changes into a new commit, with a different commit history and SHAs.
            - This cannot be done if the PR has merge conflicts.
            - It is also considered by GitHub to be "unsafe", as the results may be different compared to a merge commit.
- After a PR has been closed, you may choose to delete the relevant branch.
- Create a merge commit
- Squash and merge
- Rebase and merge

23. Use GitHub Actions for CI/CD

- See topic 28.

Domain 3: Collaboration Features – Use GitHub for project management

24. Create and manage issues

- Issues allow you to discuss work elements, like reporting bugs, suggesting new features, ideas, collect feedback.
    - You can expand this by adding sub-issues.
    - You don't need write access to create an issue; read access is fine (but you need to have another repository to which to write the issue).
    - Issues need to be enabled in the repository (administrators can disable it – see topic 14).
- To create or view issues:
    - click on the Issues tab in a repository, and click on New issue.
    - click on the … next to a comment, and click on "Reference in new issue", then click on "Create issue".
    - create an issue from a discussion, by clicking on "Create issue from discussion" on the right sidebar.
    - Create an issue from a project, by clicking on the + at the bottom of a table, group of items, or a column in board layout, and click on "Create new issue".
    - Using Copilot Chat on GitHub.
- You may have issue templates. If that is the case, you can use an issue template.
- Enter a title, and optionally:
    - add a description.
        - You can add a task list by prefacing list items with [ ], or click on the icon.
    - Paste, drop or click to add files,
    - Write with Copilot. This allows you to get suggestions for the description,
    - Add assignees, labels, projects and milestones.
- Then click "Create".
    - You could also check "Create more issues" to add additional issues.
- Once created, you can click on the … next to the issue, and select:
    - Copy link,
    - Quote reply,
    - Edit, and
    - Report content.
- You can check tasklist items, and add reactions to issues by clicking on the smiley face icon.
- You can add relationships, linking this issue with a parent issue.
- You can add comments, including using @mention.

    - You can also click on the … next to a tasklist, and you can move up/down, and convert to issue/sub-issue.
- You can also click on “Create sub-issue”.
    - There is a dropdown next to it, and you can select “Add existing issue”.
    - You can have up to 100 sub-issues per issue, and up to 8 levels of nested sub-issues.
    - To add sub-issues, you need at least repository triage permissions.
- Once the issue has been finished with, you can click on:
    - “Close issue”. You can select:
        - Close as completed – Done, closed, fixed, resolved,
        - Close as not planned – Won’t fix, can’t reproduce/replicate, stale
        - Cose a duplicate – you can then select the issue which is this a duplicate of.
    - Note:
        - You can close issues that you opened.
        - You can also close issues opened by others if you:
            - are the repository owner,
            - are a collaborator on personal account repositories, or
            - have at least triage permissions on organization repositories.
        - Issues can also be closed if they are linked to a Pull Request, and the Pull Request is merged.
    - “Delete issue”
        - Repository owners can delete issues, as can
        - Organization Repository admins.
        - The URL of a deleted issue will say that the web page cannot be found.
- You can also:
    - “Pin issue”
        - There can be up to three pinned issues per repository.
        - Pinned issues are shown above the issues list.
        - You need write access to pin an issue.
    - “Lock conversation”
        - Users without write access can no longer add new comments.
        - The conversation can be unlocked later.
    - “Transfer issue” to another repository
        - You need write permissions to both repositories.
- You can also subscribe/unsubscribe from notifications.
- If you go back to the list of issues and check one or more items, then you can:

    - Mark as open, completed, or not planned.
    - Add a label, assignee, project or milestone.
- Note: you can also create issues using GitHub Mobile.
    - GitHub Mobile is generally used for reading GitHub objects. However, you can also use it to create GitHub Issues and Discussions, to comment on and manage threads (if you have the relevant permission).

25a. Use labels

- To manage items, you can add labels to:
    - issues (topic 24),
    - Pull Requests (topics 21-22) and
    - discussions (topic 43).
- To see the list of currently used labels, go to Issues or Pull Requests – Labels. You can click on them to filter the list of (for example) issues.
- The default labels are:
    - Bug – problems,
    - Documentations – suggestions for improvements or additions
    - Duplicate – similar issues, Pull Requests, or discussions.
    - Enhancement – new feature requests
    - Good first issue – for first-time contributors. These issues are added into the repository's "contribute" page.
        - To see this page, use the URL and add "/contribute" afterwards.
    - Help wanted – someone wants help on an issue or Pull Request
    - Invalid – Item no longer relevant
    - Question – Needs more information, and
    - Wontfix – work will not continue on this item.
- You can add new labels by clicking on "New label". You can assign:
    - A name,
    - A description, and
    - A color, either from:
        - a list of 16 colors,
        - as a hexadecimal number (Red – Green – Blue), from #000000 (black) to #ffffff (white), or
        - a random color.
- You can edit or delete existing labels by clicking on the … next to those labels.
- Organizations can create, edit and delete default labels by going to:
    - Your profile picture (top-right hand corner) – Your organizations.
    - Settings – Repository – Repository defaults.

- To assign a label, go to an item, click on the wheel (settings) icon next to ”labels”.
    - You can assign multiple labels.
    - You can also click on “Edit labels”.
- Once a label has been assigned, you can click on it to see all of the issues which have the same label.
    - In the issues tab, you can also use the “Labels” drop-down to filter the list.
    - Clicking on multiple labels means that you are searching for issues which use both labels in the same issue.
    - You can change the filter to an OR by using either:
        - is:issue state:open (label:enhancement OR label:"good first issue")
        - is:issue state:open label:enhancement,"good first issue" – note: there is no gap between the comma and the next issue, and the word “label:” is not repeated.

25b. Use milestones

- Milestones can be used to track progress on multiple issues or pull requests.
    - You can assign the same milestone to multiple items, and see how they are progressing against an optional due date.
- To create a new milestone, go to Issues or Pull Requests – Milestones.
- You can then click on “New milestone” or, if this is your first milestone “Create a milestone”.
- You can then enter:
    - A title,
    - Due date (optional), and
    - Description (optional)
- And then click on “Create milestone”.
- If you go back to Issues or Pull Requests – Milestones, you can see all of the Milestones.
    - You can click on the … next to a milestone to: Edit, Close or Delete the milestone.
    - You can sort by:
        - Recently updated,
        - Furthest/closest due date,
        - Most/least complete,
        - Alphabetical/reverse alphabetical order,
        - Most/fewest issues.
- Clicking on one, you will see:
    - The due date
    - The percentage complete,
        - This is zero percent if you have not used the milestone.
    - The open issues/pull requests (which have been assigned).

- You can:
    - Alternate between open and closed issues,
    - Drag issues up and down to prioritise them.
        - You can't do this if there is more than 500 open issues in a milestone.
    - Select individual issue/pull requests and:
        - Mark it as: Open, Completed or Not Planned,
        - Add/remove labels,
        - Change the assignees,
        - Select a project, or
        - Set milestones.
    - Edit the milestone,
    - Close the milestone, or
    - Create a new issue.
- To assign a milestone:
    - Either:
        - Go to the list of issues or Pull Requests.
        - Check the relevant issue(s) or Pull Request(s),
        - Go to the Milestone dropdown,
        - Select the relevant milestone.
    - or
        - Open the relevant issue or Pull Request,
        - Click on the wheel (settings) icon next to Milestone
        - You can then click on a Milestone.
    - Note: You can only assign one milestone to an issue or Pull Request.
- You can:
    - Alternate between open and closed issues,
    - Drag issues up and down to prioritise them.
        - You can't do this if there is more than 500 open issues in a milestone.
    - Select individual issue/pull requests and:
        - Mark it as: Open, Completed or Not Planned,
        - Add/remove labels,
        - Change the assignees,
        - Select a project, or
        - Set milestones.
    - Edit the milestone,
    - Close the milestone, or

        - Create a new issue.
- To assign a milestone:
    - Either:
        - Go to the list of issues or Pull Requests.
        - Check the relevant issue(s) or Pull Request(s),
        - Go to the Milestone dropdown,
        - Select the relevant milestone.
    - or
        - Open the relevant issue or Pull Request,
        - Click on the wheel (settings) icon next to Milestone
        - You can then click on a Milestone.
    - Note: You can only assign one milestone to an issue or Pull Request.

26. Track progress with GitHub Projects

- See topics 33-35.

Domain 4: Modern Development – Implement DevOps practices

27. Understand the principles of DevOps

- DevOps allows for the delivery of software.
- It is short for Development and Operations. It is based on:
    - Collaboration – a team has a shared target, and can see everyone's priorities.
    - Automation – making tasks such as testing, deployment and monitoring easier and more accurate.
    - Continuous integration – merging code changes into a shared repository. Committing code to a shared repository for building projects and running tests.
        - More frequent commits may allow you to detect errors more quickly, and narrow the range of lines which may have the errors.
    - Continuous delivery – preparing code changes from development to test to production.
- It allows for:
    - Improved collaboration,
    - Increased efficiency and testing reliability,
    - Higher quality and security,
    - Faster release to users.
- The lifecycle includes:
    - Plan,
    - Code,
    - Continuous integration
        - Build,

        - Test,
        - Release,
    - Continuous delivery
        - Deploy,
        - Operate and
        - Monitor.
- Understand the principles of DevOps

28. Use GitHub Actions for automation

- GitHub Actions allows you to automate CI/CD.
    - It can build, test and deploy your code.
- Other uses:
    - You can also run tests automatically when changes are made to your repository.
    - When pull requests are made, they can be deployed to production.
    - You can also deploy the code to Azure, and build Infrastructure using the code (known as "Infrastructure as Code")
- They can be triggered by:
    - Events which occur in your workflow's repository, such as a PR commit,
    - Events which happen outside of the workflow's repository, and trigger a repository_dispatch event on GitHub,
    - At scheduled times, or
    - Manually.
- To view a list of templates, click on the Actions tab in a repository.
- You can build Docker containers, JavaScript and Composite Actions.
- Clicking on "Simple workflow" – Configure, you can see that workflow is written in YAML syntax.
    - "YAML" rhymes with Camel.
    - Initially stood for "Yet Another Markup Language", but is now "YAML Ain't Markup Language" (from https://yaml.org/), because its focus changed to data rather than Markup.
- It includes:
    - name – the name of the workflow.
    - on – shows the triggers, when the workflow will run, based on either a schedule or when something happens in a repository or when a workflow runs, and can include:
        - create/delete – when a branch or tag is created/deleted in the repository.
        - discussion – when a discussion is created, edited or answered
        - discussion_comment – when a comment in a discussion is created, edited or deleted.

- fork – when someone forks a repository.
- issues/label – when an issue/label is created or modified
- issue_comment – when an issue or Pull Request comment is created, edited or deleted.
- merge_group – a Pull request has been added to a merge queue.
- milestone – when a milestone is created, edited, deleted, closed or opened.
- public – when a repository's visibility has changed from private to public.
- pull_request – when an activity on a pull request happens.
    - Also: pull_request_review, pull_request_review_comment, pull_request_target
- push – when a commit or tag is pushed.
- schedule – at a specific time.
    - It uses cron to show when it should run, uses: minutes, hours, day of month (1-31), month (1-12 or JAN-DEC) and day of week (0 or SUN meaning Sunday).
    - 0,12 means 0 or 12.
    - 0-12 means 0, 1,2, 3, 4, 5, 6, 7, 8, 9, 10, 11 and 12.
    - * means all.
    - For example, the following means midnight and midday Monday to Friday.
        - schedule:
            - - cron: '0 0,12 * * 1-5'
- status – when the status of a Git commit changes. github.event.state can be error, failure, pending or success.
- workflow_run – when a workflow run has been requested, completed, or in_progress.

- jobs – these are the actions, which by default are run at the same time.
    - It then has a unique identifier (name), such as build or test.
    - runs-on defines the type of machine for the job – for example: ubuntu (Linux), windows, macos
    - steps contains the tasks, such as running commands, setup tasks, or actions.
        - uses defines an action to run
        - name is the name of the step
        - run shows the command-line program to be executed.

- GitHub Actions can be published in the GitHub Marketplace, so that others can download it.
    - Note: GitHub Marketplace can also contain:

    - Copilot Extensions,
    - AI Models, and
    - Apps which integrate with GitHub.
    - 29. Implement CI/CD pipelines
- #Go to Actions – Continuous integration – View all.#
- For Continuous Integration, you can build and test the following types of projects (among others):
    - Go,
    - Java with Ant, Gradle or Maven,
    - .NET,
    - Node.js,
    - PowerShell,
    - Python,
    - Ruby,
    - Rust,
    - Swift, and
    - Xamarin applications.
- Taking as an example the .NET template, clicking on Configure shows that it will:
    - Run when there is a “push” or “pull_request” (as shown in the “on” section). You could also have (for example):
        - when a fork event occurs,
        - when a label is created, opened, or labelled,
        - when issues are opened or closed,
        - when comments have been created or modified or
        - when another workflow is run.
    - Determine the push and pull request branches.
    - The appropriate .NET version.
    - Restores dependencies
    - Builds the project, and
    - Tests it. The results of these tests can be added to (for example) the Pull Request.

29. Implement CI/CD pipelines

- You can also add conditions using “if” – for example, if it is about a certain repository.
- You can add additional actions from the marketplace.
- For Continuous Deployment, you can deploy to (among others):
    - Node.js, Python, Java, .NET, PHP and Docker to Azure App Service,
    - Azure Static Web App,
    - Azure or Google Kubernetes Services and

    - Amazon Elastic Container Service.
- Looking at the Deploy Node.js to Azure Web App, it builds and then deploys.

Domain 4: Modern Development – Use GitHub for code review

30. Conduct code reviews

    - #Have a repository which has a collaborator which can review.
    - Create 2 issues.
    - Create a Pull Request which includes 2 files.#
- To conduct a code review, you can create a Pull Request, and add a reviewer.
    - The reviewer receives a notification.
    - You can see notifications by clicking on the inbox at the top-right hand corner.
    - You can filter on “Review requested”.
- The reviewer can then go to the Pull Request, and either:
    - go to the Files changed, or
    - Click on “Add your review”.
- To add comments to a line, click on the + which appears when you hover over the line.
    - You can add code suggestions by clicking on the +- icon.
    - Add the end, click on “Start review”.
- To add comments to a section, hover over the line where the + sign is, and then drag to other lines.
- You can click on the Comments icon to add comments to the entire file.
- Next to it, you can click on the … to:
    - Show/hide comments,
    - View, edit or Delete file, or
    - Open in desktop.
- Once you have concluded with a file, you can check “Viewed” to hide the details.
- When you have concluded, click on “Finish your review”, and select either:
    - Comment,
    - Approve, or
    - Request changes (which requires feedback to be sent back to you).
- And then click “Submit review”.
- If you requested changes, then you can go back to “Files changed” and:
    - Mark conversations as Resolved,
    - Commit suggestion, and/or
    - Add suggestion to batch (this adds the suggestion to an overall commit).
- If you have added suggestions to a batch, then you can click on “Commit suggestions” to commit them all.

- Then you can "Review changes", click on "Comment", add a comment and click "Submit review".
- Once the code review has been completed, you can then "Merge pull request" (or similar).
- If you want Pull Requests to automatically have somebody review code, you can use the CODEOWNERS file.
    - You create a CODEOWNERS file in:
        - the .github folder,
        - the root folder, or
        - the docs folder.
        - If you have multiple CODEOWNERS files, it will use the first one (in the above order).
    - The file must be under 3 Mb.
- Create the CODEOWNERS file in the following format:
    - Filename space/tab OwnerName. For example:
        - *.txt @myusername
    - # You can also add comments starting with a #, either at the start of the line or the middle.
    - The Filename can use wildcards, such as:
        - * - all files in the root folder and all subfolders
        - *.js – all files with a js extension, in the root folder and all subfolders
        - docs/* – all files in the docs folder (but not subfolders)
        - /docs/ – all files in the docs folder and also subfolders
    - In the event of multiple Filenames which match, the last match takes priority.
    - The OwnerName can be:
        - A username – @username
        - A team name - @organizationname/team-name
        - An email address – myemail@address.com (it cannot be used in managed user accounts).
        - There are be multiple OwnerNames, separated by a space. Approval from only one OwnerName is required, not all of them.
    - You should define an OwnerName for the CODEOWNERS file itself, to ensure that it cannot be changed without authorization.

31. Use GitHub's code review tools

- See topic 30.
- The GitHub code review tools include:
    - Pull requests,
    - Comments, suggestions and discussion tools,
    - Merge conflicts.

- When looking at a file, you can see the Blame view.
    - It shows the commit when that part of a file changed.
    - If you hover over that commit, you will who authored it.
- You can also add GitHub Copilot.
    - You can delegate open issues to GitHub Copilot.
        - It then creates and tests the response, and can then be developed further based on further responses.
        - It can use GitHub Actions to create the Pull Requests.
    - It can create a summary of changes that were made in a Pull Request.
        - It can also write Pull Request descriptions.
    - It can also analyze your code, find and fix mistakes.
    - You can also use Copilot Chat to ask coding-related questions, such as:
        - What is YAML?
        - Give me examples of cron.
        - Write JavaScript to give me the current time.
    - The cost is:
        - free for up to 50 chat requests and 2,000 code completions per month.
            - You can also use up to 50 premium requests. They can be used for CoPilot Chat and code completions.
        - $10 per month or $100 per year for unlimited chat and code completions, together with up to 300 premium requests.
            - You can also use code review and coding agents.
            - It is free for students, teachers, educational institutions and maintainers for popular open source project
        - $39 per month or $390 per year to access more models and up to 1,500 premium requests.
    - GitHub Copilot can be used in:
        - GitHub and GitHub Desktop,
        - GitHub Mobile (Copilot Chat only),
        - Visual Studio Code and Visual Studio,
        - Xcode,
        - JetBrains IDEs,
        - Neovim,
        - Azure Data Studio, and
        - Eclipse
    - GitHub CoPilot can be used to give you code suggestions and fixes, document code and create examples (“unit tests”).

31. Codespaces

- Codespaces are cloud-based development environments.
- They are hosted in a Docker container which runs Linux on a Virtual Machine.
    - They contain between 2 and 32 cores, 8-64 Gb Ram, and 32-128 Gb storage.
- Personal accounts include, free of charge:
    - GitHub Free: 120 core hours and an average of 15 Gb of storage and per month
        - As a Codespace needs a minimum of 2 cores, that is a maximum of 60 hours.
    - GitHub Pro: 180 core hours and an average of 20 Gb of storage and per month
- You will receive an email when you have used 75%, 90% and 100% of your free allowance.
    - If you have used your allowance, future use of Codespaces will be blocked unless you have a payment method saved and a budget set.
- The cost for additional usage is:
    - 9 US cents per core hour, and
    - 7 US cents per Gb (on average) per month, whether the Codespace is active or inactive.
        - However, the container based on the default image is not counted as used storage.
- To create a Codespace:
    - In a repository:
        - click on “Use this template” – “Open in a codespace”, or
        - click on Code – Codespaces tab – “Create codespace on [name of branch]”,
        - In a Pull Request, go to Files changed and click on either “Review in codespace” or Code - Codespaces tab – “Create codespace on [name of branch]”.
    - You can also click on the … next to the + in Codespaces to select:
        - New with options… - select the branch, region and machine type.
        - Configure the Dev container (devcontainer.json file).
        - Set up prebuild configurations.
            - This can speed up Codespace creations by running all of the tasks needed to build your development environment.
            - You can specify:
                - when the prebuild happens (every push, when the configuration files changes, or on a schedule).
                - which regions will be used.
                - how many prebuild template versions which can be retained.
                - who is going to be sent an email if a prebuild fails.

        - Manage codespaces
        - Share a codespace configuration
        - Open the documentation
    - It can also be created from a branch, and from a commit.
    - You can also create a Codespace using Visual Studio Code or the GitHub Command Line Interface (CLI).
    - You can create multiple codespaces for a single repository or branch.
        - There is a variable limit to the number of codespaces you can create.
        - You will get a warning if you try to exceed that limit, saying that you need to remove an existing codespace before creating a new one.
    - You can enable or disable Codespaces if you are the owner of an organization which uses a GitHub Team or Enterprise Cloud plan.
        - You can do this in the Settings of Your organizations.
    - When you create a Codespace, your repository will be cloned into the /workspaces folder.
- When the Codespace is created, then:
    - A Virtual Machine and storage are assigned,
    - The Docker container is created, which by default includes a web version of Visual Studio Code.
    - You are then connected to the codespace.
    - Additional commands may be created, based on the configuration in your devcontainer.json file.
    - Your codespace will, by default, be timed out after 30 minutes (this can be adjusted).
        - This happens when there is no user activity, such as typing or using the mouse.
- The Codespace uses a web version of the Visual Studio Code. It includes:
    - An activity bar on the left-hand side. This displays the Views.
    - Next is the Side bar, which shows your project files.
    - To the right is the editor, where you can edit your files.
    - Beneath there are the panels, where you can see output, debug information, and the Terminal for issuing commands.
    - At the very bottom is a Status bar, which contains information such as the branch name.
        - You can click on the branch name to change it.
    - At the right-hand side you may see Copilot, where you can ask questions.
        - You can open the Copilot inline chat by pressing Ctrl+I, if you have installed the GitHub.copilot-chat extension installed.
- In the Side bar, if you hover over the codespace, you will see icons to:

    - Add a new file,
    - Add a new folder,
    - Refresh the Explorer, and
    - Collapse the Folder in Explorer.
- If you click on the Activity bar hamburger icon, you can go to File, and then:
    - New Text File,
    - New File...,
    - Open File..., and
    - Open Folder...
- You can also click on a file in the Side bar, and then edit the file online.
- After making the changes:
    - You can view the changes by clicking on the "Source Control" view (the third icon in the Activity bar).
    - You can then click "Commit"
        - You can also click on the drop-down next to the Commit, and select "Commit & Push/Sync/Create Pull Request".
    - You can then "Publish Branch", and select the repository name.
    - You can also click on the ... at the top next to Source Control to show/hide Repositories, Changes and Graph.
    - Just underneath the "Source Control" title, you can hover over it and click on:
        - The check mark to Commit,
        - The speech bubble for Copilot Code Review,
        - The two circles and + for Create Pull Request,
        - The Refresh icon,
        - The ... for more options.
- When you have finished, you should click on the ... and click on "Stop codespace".
    - Otherwise the codespace will continue to run for another 30 minutes (by default), and you will incur charges for the running codespace.
    - You can also stop it in Visual Studio Code or in the CLI.
- You can see the list of Codespaces in a repository by going to Code – Codespaces tab.
    - You can also click on the hamburger icon at the top-left corner and click on Codespaces to see a list of all of the codespaces.
    - Here, you can also click on a repository to filter the list.
    - You can also create a
        - "New codespace" or
        - "Explore quick start templates" (you can also click on "See all"). You can then click on "Use this template", or click on the name of the codespace to view it in a repository.

- If you click on the … next to a Codespace, you can:
    - Rename the Codespace,
    - Export changes to a branch,
    - Change machine type (number of cores, for example),
    - Stop codespace,
    - Enable auto-delete codespace (they are deleted after having been stopped and are inactive for 30 days)
        - Codespaces would be deleted regardless of whether they have uncommited changes.
        - When a codespace will soon be auto-deleted, it will say "Expiring in _ days" in the list of codespaces.
    - Open in Browser, Visual Studio Code, or JuptyerLab,
    - Delete the Codespace
- You can change Codespace settings by:
    - Click on your profile picture,
    - Go to Settings,
    - On the left-hand side, click on Codespaces.
    - You can change:
        - Whether notifications will be sent when codespaces will soon be deleted due to inactivity.
        - Editor preference
            - Visual Studio Code (with the GitHub Codespaces extension),
            - Visual Studio Code for the Web, and
            - JupyterLab notebooks.
        - The default idle timeout (from 1-240 minutes; the default is 30 minutes),
        - The default retention period (from 1-30 days; the default is 30 days),
        - Region – either:
            - Set automatically, based on your IP address, or
            - Set manually: US East, US West, Europe West, Southeast Asia or Australia

32. Understand best practices for code reviews

- Reviewers should provide suggestions and highlight any issues.
    - They may also have a checklist of things to review.
    - You may wish to have smaller Pull Requests, which are easier and faster to review.
    - It should state what the Pull Request does.
- When you are reviewing code, you should consider issues about:
    - Security,

        - Performance,
        - Maintainability,
        - Compatibility,
        - Scalability,
        - Usability,
        - Accessibility,
        - Localization,
        - Legal/compliance,
        - Testability, and
        - Documentation (with the company's requirements).
- Some reviewers could be from Quality Assurance.
    - It may be useful to couple a QA person, who might not have sufficient coding experience by themselves, with a developer.
- You may have senior reviewers reviewing code written by more junior developers.
    - In larger teams, you might have multiple developers reviewing code.

Domain 5: Project Management – Manage projects with GitHub

33. Create and manage GitHub Projects

- GitHub Projects allow you to plan and track issues, Pull Requests and other items.
- To create a new Project, in a repository click on the "Projects" tab, and click on "+New project".
    - You can start with a:
        - New table – like a spreadsheet,
        - New board – a list with items in columns, and
        - New roadmap – items with start and target dates.
    - Instead, you can select a project template, such as:
        - Team planning,
        - Feature release,
        - Kanban,
        - Bug tracker,
        - Iterative development,
        - Product launch,
        - Road map, and
        - Team retrospective.
- Once a project has been created, you can create new views by clicking on "+ New view".
    - Either a Table, Board or Roadmap layout, or you can duplicate the current view.
- You can manage projects:

    - From a repository, by going to the "Projects" tab.
    - In all repositories, by clicking on the hamburger icon and going to "Projects", where you can see:
        - Recently viewed, and
        - Created by me.
- If you click on a … next to a project:
    - you can "Make a copy".
    - In a repository, you can "Remove project".
    - In all repositories, when looking at "Recently viewed", you can "Remove from recently viewed".

35. Integrate GitHub Projects with issues and pull requests

- You can add issues and pull requests by clicking on the + sign.
- You can then select:
    - Create new issues
        - You can select a repository, then
        - Select an issue template (such as the blank template), and
        - Create an issue, with a title, and optionally a description, assignee, label and milestone.
    - Add item from repository.
        - You can select a repository, then
        - Select one or more issues, and then click on "Add selected items".
- You can filter by keyword in the search box.
- You can drag issues up and down using the left-hand margin.
    - You can also hover over that margin, and click on the arrow to:
        - Move item. You can:
            - "Move item after", and select the item the current item will be after, or
            - "Move to position", and enter a position from 1 (at the top) to the number of items.
        - Archive – it will then be removed from the list of active projects, and
        - Delete from project – it will then be removed.
- You can also edit assignees and status. The statuses are:
    - Todo – this item has not been started,
    - In Progress – actively worked on, and
    - Done – Completed.
- You can click on the + in the columns, and show/hide columns. The columns are:
    - Title, assignees, and Status (by default – visible),

- Labels, Linked pull requests, milestones, repository, reviewers, parent issue, and sub-issues progress.

33. Create and manage GitHub Projects

- You can also add "+ New field", with a field type of:
    - Text,
    - Number,
    - Date,
    - Single select – this is a list of up to 50 options
    - Iteration
        - These are used to identity phases with repeating durations with a start and end date.
        - They can be used to plan upcoming work and group issues and pull requests.
        - They can include breaks.
- You can click on the … next to the column to:
    - Select the column,
    - Sort ascending/descending,
    - Filter by values (where you use a "Filter by keyword or by field" textbox to filter),
    - Group by – separates each unique variation of that column into separate groups,
    - Slice by values – this opens a pane which you can select by:
        - Assignees,
        - Status (or other custom fields),
        - Labels,
        - Milestone,
        - Repository,
        - Parent issue, and
        - No slicing (remove the slice).
    - Hide field,
    - Move column left/right, and
    - For custom fields, "Field settings".
- At the top-right hand corner, you can click on "Add status update" or the "Project details" icon to:
    - Add short description for the project,
    - A README file, and
    - Add an update.
        - The status updates are: Inactive, On track, At risk, Off track and Complete.
        - Enter start and target dates.

        - Add a description, and files.
- If you click on a dropdown next to the View tab, you can adjust:
    - The view (table, board or roadmap)
    - The fields which are shown,
    - Fields which are Grouped by,
    - Fields which are sorted by (you can select up to 2 fields),
    - Fields which are sliced by.
- You can also:
    - Generate chart. (You can also see this by clicking on the Insights icon.) By default, this shows a count of the items over time.
        - You can adjust the time range in the X-axis to:
            - 2 weeks,
            - 1 month,
            - 3 months,
            - The maximum range, or
            - A custom range.
        - You can click on “Configure” to:
            - Change the layout to:
                - Bar or stacked bar,
                - Column or stacked column,
                - Line, or
                - Stacked area.
            - Change the X-axis to:
                - Time
                - Assignees
                - Labels
                - Milestones
                - Parent issue
                - Repository and
                - Status.
            - Change the Y-axis to:
                - Count of items, or
                - Sum of a field.
            - Depending on the layout, you may also have a “Group by” option.
        - You can click on the drop-down next to the chart, and select:
            - Configure – which is the same as the Configure button,

        - Save changes to new chart, and
        - Discard.
    - Rename, move, duplicate or delete the view, and
    - Export the view data.
- If you click on a project, you can go to … - Workflows.
- The built-in automations are used as follows
    - When issues and Pull Requests in your project are closed, change the Status to Done, and
    - When Pull Requests in your project are merged, change the Status to Done,
    - When the status is updated to Done, close the issue, and
    - When an item in a project has sub-issued, add sub-issues to the project.
- By clicking on a workflow, you can:
    - Click on the pencil icon to rename the workflow,
    - Click on “Edit” to change the items (such as “issues” and “pull requests”).
- You can also edit the following workflows:
    - When an issue and Pull Request is added to the project, set the Status to Todo.
    - When it is reopened, set the Status to “In Progress”.
    - When a Pull Request has a review requesting changes, set the Status to “In Progress”.
    - When a Pull Request is approved, set the Status to “In Progress”.
    - Auto-archive an item when it has last been updated 2 weeks ago.
    - When a new item has been created or updated, add the item to the project.
- If you click on Edit for a non-active workflow then, after adjusting any parameters, you can click on “Save and turn on workflow”.
- If you click on a project, you can go to … - Settings.
    - In the “Project settings” tab, you can also adjust the Project name, Short description and README, and add files.
    - You can reorder custom fields, and click on the … to adjust their settings and use the “Advanced Move…”
    - You can make a copy of the project.
    - In the Danger zone, you can:
        - Adjust the visibility,
        - Close the project, and
        - Delete the project.
    - In the “Manage access” tab, you can:
        - Adjust the visibility,
        - Invite collaborators with:

        - Read-only role – can see the project,
        - Write mode – can see and make changes to the project, and
        - Admin – can see, make changes to, and add new collaborators to the project.
    - Note – this does not change whether collaborators can see the actual issues and pull requests.
    - If an admin, adjust collaborators' roles, or remove them.
- You can also adjust custom fields' properties.

34. Use project boards for task management

- You can use a project board by:
    - Creating a project and using a board view, or
    - Clicking on "+ New view" and selecting a Board layout.
- It shows all of the items by Status.
    - This is called a Kanban board.
    - You can drop the items from one column to another to change its status.
- You can filter at the top based on the:
    - Repository,
    - Assignees,
    - Label,
    - Is (an Issue, a Pull Request, Open, Closed, Draft and Merged),
    - Title,
    - Status,
    - Linked pull request,
    - Milestone,
    - Reviewers,
    - Parent issue, and
    - Sub-issues progress.
    - You can also Exclude (which adds a minus sign).
- You can click on the + at the bottom and start typing to:
    - Create new issue,
    - Create a draft, and
    - Add item from repository.
- Clicking on the drop-down arrow in the View tab, in addition to the options in the Table view, you can also:
    - Change the Column field, and
    - Change the Field sum (the calculation)

- If you click on the … next to the column, you can "Set limit" – specify the maximum number of issues and Pull Requests to be shown in a column.
    - There is no … in the "No status" column.

33. GitHub Projects – Roadmap view

- The Roadmap View also allows you to add dates.
- First of all, you need to create custom Date fields by clicking on "Date fields".
- You can then allocate the custom date fields to act as the Start date and/or Target date.
- You can add dates by clicking on the + near the issue/Pull Request. This will add it to today's date.
    - You can then drag it to the appropriate date.
    - You can also go to the Gantt chart, where a + will appear where your cursor is hovering.
- You can add markers for:
    - Milestones,
    - Start Dates and
    - Target Dates (which can look like an End date).
- You can:
    - Sort by up to 2 fields,
    - Change the Zoom level to Month, Quarter and Year,
    - Move to "Today", and
    - Scroll forward or backwards.
- If you click on the drop-down in the View tab, in addition to the options in the Table layout, you can:
    - Change the fields which are used for the Dates,
    - Change the Zoom level,
    - Truncate titles in the Gantt chart,
    - Show date fields on the left-hand side. Note: you cannot change which other fields are shown.

Domain 6: Privacy, Security, and Administration – Ensure repository security

36. Set up branch protection rules

- You can protect a branch or multiple branches to stop unwanted things from happening to it.
- To do this:
    - Click on the Settings (Wheel) icon in a repository.
    - Either:
        - Go to Branches – Add branch ruleset, or
        - Go to Rules – Rulesets – New ruleset – New branch ruleset.

- Enter a Ruleset Name.
- The default Enforcement status is “Disabled”, so it will not be enforced.
    - You can change it to “Active”, so that it is enforced.
- You can optionally create a Bypass list, so that they will not apply to roles, teams or apps.
- You should then select the Target branches by clicking on “Add target”, and selecting:
    - Include default branch,
    - Include all branches,
    - Include/exclude by pattern – for example:
        - *dev* will target all branches which include “dev”. * means any number of characters, but not including /s.
        - Prod/* will target all branches with prod, then /, then any characters (but not any additional / characters).
    - If you have multiple rules which affect the same branches:
        - the rules which mention the specific branch will have priority,
        - If there is still a tie, the one created first will have priority.

- Then you can set up the branch protection rules. They are:
    - Restrict creations/updates/deletions
        - Only users with bypass permission will be allowed to do this for the corresponding branches.
    - Require linear history.
        - This stops collaborators pushing merge commits; they must use squash merge or rebase merges instead. Note: the repository must allow these for this option to be used.
    - Require deployments to succeed before merging
        - For example, changes must be successfully deployed to a staging environment before merging to the default branch.
        - You can specify which environments can be used.
    - Require signed commits
        - This requires electronic signatures to be used
    - Require a pull request before merging
        - The following options are available:
            - Requires a number of approvals to be done before merging.
            - Dismiss stale Pull Request approvals when new commits are pushed
            - Require review from Code Owners
            - Require approval of the most recent reviewable push, by someone other than the person who pushed it

        - Require conversation resolution before merging
        - Request pull request review from Copilot.
        - Restrict merging methods to merge and/or squash and/or rebase.
    - Require status checks to pass. You can:
        - Require branches to be up to date before merging,
        - Do not require status checks on creation
    - Block force pushes
    - Require code scanning results
        - From external tools such as CodeQL.
- Once the rules have been set, press “Create”.
- Following creation, you can click on the … next to the rule and either export or delete the ruleset.
    - To edit it, press the ruleset.

37. Use security features like Dependabot

- In a repository, you can click on the Security tab.
    - The Security overview shows security alerts, and allows you to enable/disable/edit:
        - Security policy – create a markdown file to tell users about security status of the repository.
        - Security advisories,
        - Private vulnerability reporting – whether you can report security vulnerabilities to the owners/maintainers of the repository, privately.
        - Dependabot alerts – security vulnerabilities in the dependencies to your project.
        - Code scanning alerts – finds vulnerabilities/errors in your code which could be used by hackers
        - Secret scanning alerts – whether your API (Application Programming Interface) keys or tokens have been found.
    - On the left-hand side, you can look at:
        - Security policies reports,
        - Published security advisories,
        - Dependabot alerts,
        - Code scanning tool configuration, and
        - Secret scanning alerts.
- To enable Dependabot:
    - Click on the Settings (wheel) icon in a repository.
    - Go to Advanced Security.
- Dependabot gives you three things to monitor dependencies:

- Alerts – about vulnerabilities in the dependencies
    - This requires the dependency graph to be enabled.
- Security updates – automatically raises Pull Requests to update dependencies which have known security vulnerabilities.
    - This requires both the dependency graph and Dependabot alerts to be enabled.
    - You can also group all updates into one Pull Request.
- Version updates – keeps your dependencies current.

- To activate them, click on Enable.
    - Once enabled, you can click on the Wheel (configure) icon for Dependabot alerts. This allows you to:
        - "dismiss low-impact alerts for development-scoped dependencies", to reduce the number of alerts you receive.
        - Create custom rules.
    - Once alerts are enabled, you can also go to the Security tab in the repository, and go to "View Dependabot alerts".
- Other Advanced Security features include:
    - CodeQL analysis.
        - This is available for:
            - Public repositories, and
            - Organization-owned repositories on GitHub Team with GitHub Code Security enabled.
        - You can:
            - Use a default setup to allow CodeQL to scan your code.
            - Use advanced setup to customize the workflow, or
            - Use the Command Line Interface to scan it in an external CI (Continuous Integration) system.
    - Add third-party scanning tools,
    - Enable Copilot Autofix, both for CodeQL and for third-party scanning tools.
    - Change the alert severity level which would cause scans to fail. The default Severity levels are:
        - Security alert: High or more (which includes "Critical"),
        - Standard alert: Only errors (which doesn't include only warnings).
    - Looking for secrets in public repositories (this is on by default), and
    - Block commits than contain secrets (this is off by default).

38. Manage repository access and permissions

- See topic 15.

Domain 6: Privacy, Security, and Administration – Administer GitHub organizations

39. Create and manage organizations

- To create a new organization:
    - Click on your icon (top-right hand corner), then go to “Your organizations”.
    - Then click on “New organization”.
        - Note: “organization” is often shortened to “org”.
- Note: if you click on “Turn [me] into an organization” instead:
    - Your username will change from being a personal account to an organization.
    - There are a number of changes that will be made, such as:
        - You cannot sign back into the previously personal account.
        - Any user profiles become organization profiles.
        - Any repos previously owned by the user are now owned by the organization.
    - It cannot be undone.
        - It is advised that, instead of doing this, you:
            - optionally rename your existing personal account (if needed),
            - create a new organization, and
            - transfer your repositories to your new organization account.
- You can choose the plan for your organization:
    - Free – includes:
        - 2,000 CI/CD minutes per month for private repositories, and unlimited for public repositories.
        - 500 Mb of Packages storage for private repositories, and unlimited for public repositories.
    - Team - $4/user/month, which adds:
        - The ability to turn on or off GitHub Codespaces,
        - Protected branches,
        - Multiple reviewers in pull requests,
        - Draft pull requests,
        - Pages and wikis
            - Pages allow you to host a website about your project, company or yourself. It takes pages from a repository (like Microsoft learn).
            - Wiki pages host documentation,
        - 3,000 CI/CD minutes/month and 2Gb of packages storage for private repositories.
    - Enterprise - $21/user/month, which:

        - allows you to centrally manage policy and billing for multiple GitHub organizations
        - has a Service Level Agreement for 99.9% monthly uptime (a maximum about 42 minutes per month downtime).
        - enhances security and compliance
            - You can add Enterprise Managed Users – own and control your enterprise user accounts.
            - You can provision new user accounts from your identity provider (IdP), together with controlling usernames, profile data, organization membership and repository access. Your IdP include:
                - Active Directory Federation Services (AD FS),
                - Entra ID,
                - Okta,
                - OneLogin,
                - PingOne, and
                - Shibboleth
            - You can use SAML (Security Assertion Markup Language) Single Sign-on (SSO) to manage access to organizations. When you access resources, GitHub will redirect you to your identity provider to authenticate before going back to GitHub.
            - You can also enable team synchronization. This synchronizes a GitHub Team with an Identity Provider (IdP) Team, reducing the need for manual updates. You can use either Okta or Entra ID commercial tenants (not Gov Cloud). If you are using Entra ID, then it needs:
                - Global or Privileged Role administrator,
                - SAML Single Sign-on enabled (SSO)
                - Read all group memberships,
                - Read all users' full profiles, and
                - Sign in and read user profile.
    - Enterprise is available in two forms:
        - GitHub Enterprise Cloud (on the Internet, in a specific region on a unique subdomain) and
        - GitHub Enterprise Server (self-hosted).
- You should then enter:
    - Organization name.
    - Contact email address.
    - Whether the organization belongs to:
        - Your personal account, or

        - A business or institution and, if so, what its name is.
            - If this is the case, then that business/institution will then control the org.
    - You can also sign up for optional extras, like GitHub Copilot Business.
- You should then review and accept the GitHub Customer Agreement and Privacy Statement and click Next.
- You can optionally then add users to your organization as organization members.
    - You can search by username, full name or email address.
    - An organization can have as many members as is needed.
        - However, there may be some reduction in speed of access if there is more than 100,000 members.
    - Having done this, you can click "Complete setup" (or you can "Skip this step").
- An organization can own a maximum of 100,000 repositories.
- To manage organizations:
    - Either:
        - In a personal account, click on your icon (top-right hand corner) and go to "Your Organizations", or
        - Click on the "Mona" cat icon (top-left hand corner), click on the drop-down underneath it, and click on "Manage organization", or
        - Go to the organization (perhaps by Click on the "Mona" cat icon (top-left hand corner), click on the drop-down underneath it, selecting the organization, and going to "View organization".
    - Next to your organization, click on "Settings".
        - In this "General" tab, you can add email and social media details, and change the picture. You can also rename, archive or delete the organization.
    - Note – only organization owners and billing managers can see all the organization account settings.

40. Set up organization-level security

- To set up your organization-level security, in the Organization settings you click on "Member privileges". In this tab, you can select:
    - The minimum permissions for members, both in a repository and for projects (see topic 41).
    - Whether members can create public or private repositories.
        - This does not change the visibility of existing repositories.
        - If you want to restrict the creation of public repositories, you will need GitHub Enterprise Cloud.
    - Whether forking is allowed in private repositories.
        - It is always allowed for public repositories.

- Whether members can publish public and/or private pages (websites for your projects).
    - Private pages are only allowed using GitHub Enterprise Cloud.
- Whether outside collaborators can request access.
- Whether admin members or only organization owners can:
    - change the visibility of repositories,
    - Delete or transfer repositories,
    - Delete issues.
- Whether any member or only organization owners can create teams.

41. Manage teams and members

- You can see a list of the members by going to the People tab. Here you can:
    - Filter the list by:
        - Member name,
        - Two-factor authentication – either:
            - Everyone or disabled,
            - secure only (passkeys, security keys, authenticator apps, or the GitHub mobile app), or
            - either secure or insecure (for example, text or SMS).
        - Membership – owners or members who are not owners.
    - Export the list,
    - Invite new members
        - You can give the new member the role of member or owner.
            - Owners have full administrative rights to the organization and have complete access to all repositories and teams.
            - Members can see all other members, and can be granted access to repositories. They can also create new teams and repositories.
        - They can see the invitation in Settings – My Organizations.
    - Change whether:
        - Members' membership is displayed on their public profile or not (private).
    - See members' roles, and the number of teams and roles they have.
    - Find more information about a member by:
        - clicking on their link.
        - clicking on the … next to a member, then going to Manage.
    - Click on the … next to a member to:
        - If the invitation is pending, edit or cancel the invitation,
        - If the member has previously accessed, convert the member to be an outside collaborator, or remove from the organization.

- On the left-hand side, you can change from looking at members to look at:
    - Outside/pending collaborators,
    - Outstanding/failed invitations, and
    - Security Managers.
- You can see a list of the roles members have by clicking on "[number] roles". This takes you to the organization settings – Organization roles – Role assignments.
- If you click on "New role assignment", you can enter users or teams, and assign the following roles (full details of these roles can also be seen in Organization roles – Role management):
    - All-repository read – for non-code contributors to view/discuss your project.
    - All-repository triage – for contributors to manage issues and pull requests with no write access.
    - All-repository write – for contributors to push to the project
    - All-repository maintain – for contributors such as project managers who need to manage the repository, but not full admin access, so they cannot do sensitive or destructive actions.
    - All-repository admin – full access, including sensitive and destructive access such as managing security and deleting a repository.
    - Apps manager – manage an organization's GitHub Apps
    - CI/CD Admin – manage an organization's Actions policies, runners, runner groups, network configurations, secrets, variables, and usage metrics
    - Security manager – manage security policies, security alerts, and security configurations for an organization and all its repositories.
- Up to 5 Custom roles can also be created using GitHub Enterprise.
    - You can define specific permissions for these roles.
- If you have the All-repository read permission (or above), you can:
    - Pull requests
        - Pull from/fork the person or team's assigned repositories
        - Send pull requests from forks of the team's assigned repositories
        - Submit reviews on pull requests
    - Edit and delete their own comments
    - Issues
        - Open issues
        - Close issues they opened themselves, and reopen issues they closed themselves
        - Have an issue assigned to them
    - View published releases
    - View GitHub Actions workflow runs
    - Edit wikis in public repositories

        - Report abusive or spammy content
        - View and install packages
        - View rulesets for a repository
        - Create new discussions and comment on existing discussions
        - Codespaces
            - Create codespaces for private/public repositories
            - Create codespaces for private repositories with Codespaces secrets access
        - View code scanning alerts on pull requests
- If you have the All-repository Triage permission, you can also:
    - Apply/dismiss labels
    - Close, reopen, and assign all issues and pull requests
    - Apply milestones
    - Mark duplicate issues and pull requests
    - Request pull request reviews
    - Hide anyone's comments
    - Move a discussion to a different category
    - Lock and unlock discussions
    - Individually convert issues to discussions
    - Delete a discussion
- If you have the All-repository Write permission, you can also:
    - Edit wikis in private repositories
    - Create, edit, delete labels and milestones
    - Pull requests
        - Approve or request changes to a pull request with required reviews
        - Apply suggested changes to pull requests
        - Enable and disable auto-merge on a pull request
        - Merge a pull request
        - Mark a draft pull request as ready for review
        - Convert a pull request to a draft
    - Push to (write) the person or team's assigned repositories
    - Edit and delete anyone's comments on commits, pull requests, and issues
    - Lock conversations
    - Transfer issues
    - Act as a designated code owner for a repository
    - Create status checks

- GitHub Actions
    - Create, edit, run, re-run, and cancel GitHub Actions workflows
    - Create, update, and delete GitHub Actions secrets/variables using the REST API
- Create and edit releases, and view draft releases
- Publish packages
- Define code owners for a repository
- Rename a branch other than the repository's default branch
- Create and edit categories for GitHub Discussions
- Transfer a discussion to a new repository
- Manage pinned discussions
- Convert issues to discussions in bulk
- Dependabot and scanning alerts
    - Receive Dependabot alerts for insecure dependencies in a repository
    - Dismiss Dependabot alerts
    - List, dismiss, and delete code scanning alerts
    - View and dismiss secret scanning alerts in a repository

- If you have the All-repository Maintain permission, you can also:
    - Edit a repository's description
    - Manage topics
    - Enable wikis and restrict wiki editors
    - Configure pull request merges
    - Configure a publishing source for GitHub Pages
    - View content exclusion settings for GitHub Copilot
    - Push to protected branches (Doesn't apply to rulesets as these have a different bypass model)
    - Create and edit repository social cards
    - Limit interactions in a repository
    - Enable GitHub Discussions in a repository
- If you have the All-repository Admin permission, you can also:
    - Manage individual, team, and outside collaborator access to the repository
    - Create, update, and delete GitHub Actions secrets/variables on GitHub.com
    - Delete and restore packages
    - Manage branch protection rules and repository rulesets
    - Merge pull requests on protected branches, even if there are no approving reviews
    - Delete an issue

    - Add a repository to a team
    - Manage outside collaborator access to a repository
    - Change a repository's visibility
    - Make a repository a template
    - Change a repository's settings
    - Manage team and collaborator access to the repository
    - Edit the repository's default branch
    - Rename the repository's default branch
    - Manage webhooks and deploy keys
    - Manage the forking policy for a repository
    - Transfer repositories into the organization
    - Delete or transfer repositories out of the organization
    - Archive repositories
    - Display a sponsor button
    - Create autolink references to external resources, like Jira or Zendesk
    - Edit the custom property values for the repository
    - Create security advisories
    - Enable the dependency graph for a private repository
- You can also organize members into teams.
    - This can be used for managing access and sending notifications.
- To create a new Team, go to the Teams tab, and click on “New team”. You can then enter:
    - Team name and description,
    - Whether this team is contained within another team (“nested” from a “parent” team).
        - Nested teams can have different members compared to the parent team.
    - The team visibility:
        - Visible – the team can be viewed and @mentioned by every organization member (but not outside of the organization). This is the recommended team visibility.
        - Secret – only visible for team members and owners. You cannot have a secret team which is contained within another team.
    - Whether people will be notified when the team is @mentioned (using @Company/Team).
        - If this is disabled, members will still receive review notifications.
- To go to a Team page:
    - Click on the Teams tab, and
    - Click on the name of the team.

- Here you can:
    - Click on the Teams tab to add an existing team to be a child team.
    - Click on the Repositories tab to "Add repository", or check a repository to "Remove from team".
    - Change settings such as name, description, parent team (thereby moving the team to another visible, not secret, team), visibility, notifications, profile picture and delete the team by clicking on the Settings tab.
        - In "Code review" on the left-hand side, you can
            - Stop the entire team from being notified if both a team and individual members are requested to review,
            - Automatically route code review requests to team members (and specify additional settings).
        - You can also click on "Scheduled remainders" from a Slack channel.
- You will see the current members in the Members tab.
    - If you check one or more members, you can:
        - Change role to Maintainer or Member (this has no effect on Organization owners).
            - Maintainers can:
                - Change the team's name, description, and visibility,
                - Request to add a child team, or add/remove a parent team (thereby moving the team to another visible, not secret, team),
                - Set the team profile picture
                - Add/remove organization members to the team,
                - Remove the team's access to repositories,
                - Manage code review assignments for the team, and
                - Manage scheduled reminders for pull requests.
            - Members cannot do any of these actions.
        - Remove from team.
    - You can also click on "add a member".
        - You can subsequently change this member's role to be a maintainer.
- Users can see teams which they are a member or above in their personal dashboard.
    - They can then click on the team to get to the team page.
- Teams can be added to organization-owned repositories (but not personal repositories) by:
    - going to that repository,
    - clicking on "Settings" (wheel),
    - On the left-hand side, go to Access –Collaborators & teams.

        - If it says “Collaborators” and not “Collaborators & teams”, then it is a personal repository.
    - Click on “Add teams” (Not “Add people”), and select the team
    - Select the role that the team will have for that repository and click on “Add selection”.
- Once it has been added, then on that page:
    - A team’s role can be changed, and
    - You can remove the team from accessing that repository by clicking on the bin.
- If a team is a nested team, then they will automatically get the relevant permissions from the parent team.

Domain 7: Benefits of the GitHub Community – Engage with the GitHub community

42. Participate in open source projects

- Open Source Projects are projects where:
    - The source project is freely available,
    - Users may view, modify and adapt the project, and
    - You can share it for both commercial and non-commercial purposes.
- Advantages are:
    - As you can alter the code, it is both flexible and customizable.
    - Based on the latest technology.
    - Free cost, with no budget for technical vendors,
    - Allows for community support.
- Disadvantages are:
    - Can have a steeper learning curve,
    - Less polish – often in beta,
    - Potential liability and security issues, as there are no warranties.
- There are various licenses that you can use, include:
    - The MIT license – very few restrictions (just need to keep the license and copyright notice in any copies),
    - GNU General Public License (GPL) v2 – source code must be available for public use, and any adaptions must also be similarly licensed.
    - GNU General Public License (GPL) v3 – doesn’t require that source code must be available for public use, but any future adaptions must be similarly licensed.
    - Apache 2.0 – any major changes need to be logged.
- You may also hear about “innersource”. This is similar to open source projects, but its use and development is restricted to a particular organization.
- Advantages:
    - Increased contributions to improving the project, and solving bugs

        - You may wish to monitor this to see how well your project might be doing – but don't use this as the only thing you monitor.
    - Allows contributions from people in your organization
    - Don't need to start from scratch
    - Transparent
- You can find topics you might be interested in by going to:
    - https://github.com/topics, and searching for an appropriate topic or repository, or
    - https://forgoodfirstissue.github.com, and searching for an appropriate repository,
    - https://github.com/collections/choosing-projects or
    - https://github.com/firstcontributions/first-contributions for a sample repository.
- #see https://github.com/topics - Awesome Lists - 5th one down "avelino/awesome-go", which has a discussion#.
- Have a read of the Contribution Guidelines and README.
    - If you have sufficient permissions, you can:
        - Validate an issue or add additional context to an existing issue, and
        - Test a pull request.
    - Go to the Discussions.
- You can:
    - Organize planning events,
    - Design layouts for usability,
    - Write documentation,
    - Organize issues,
    - Answer questions,
    - Help other coders.
- If you want to keep up to date with changes or updates, you can:
    - Subscribe to:
        - An issue or a Pull Request, or a conversation within it, or
        - All activity in a repository (by clicking on "Watch" in the repository).
    - Follow a user or organization by:
        - Going to that user or organization, and
        - Clicking on Follow
        - This allows you to see public activity, such as new discussions or repositories.
        - Note: members of an enterprise with managed users can only follow other members of their enterprise.
- You can also create a gist – a code/note/snippet.
    - A snippet is a small piece of information. You can include markdown.

    - You create it by going to gist.github.com
- You can then add:
    - A description,
    - Filename, including extension
        - If you have a .md extension, then you will get a new "Preview" tab.
    - The code/note/snippet, and
    - Files.
- Then click on:
    - Create secret gist – these are available onto to people you give the URL to, or
    - Create public gist – available to everyone, including search engines.
- When you have created a gist, you can:
    - Add comments
    - See revisions,
    - In the Embed dropbox, you can:
        - Embed into your website,
        - Copy a shareable link,
        - Clone the gist.
        - The next icon copies the results.
    - Save it so you can use it in GitHub Desktop.
    - Download it as a Zip.
    - Subscribe/unsubscribe from updates,
    - Edit
        - When editing, you can also disable comments, make it public, or Delete it.
    - Delete, and
    - Star the gist.
    - You may also be able to fork it.

43. Use GitHub Discussions

- GitHub Discussions is a forum for a repository.
- Unlike GitHub Issues, they do not need to be tracked or related to documents.
- To enable it:
    - Go to your repository or organization.
    - Click on Settings (wheel).
        - If you can't see this icon for an organization, you do not have the appropriate permissions.
    - For a repository:
        - In "General" on the left-hand side, check "Set up discussions",

        - edit the template, and
        - click “Start discussion”.
    - For an organization Discussion:
        - In “Discussions” on the left-hand side, check “Enable discussions for this organization”,
        - Select a source repository.
            - This can be changed later – however, the discussions will not be transferred to the new repository.
- To create a new discussion:
    - Go to the Discussions tab.
    - Click on “New discussion” (read permission or above needed).
    - Select the category.
        - The default categories are: Announcements, General, Ideas, Pools, Q&A and Show and tell) and click “Get started”.
    - Enter a title, body and optionally add files and labels.
    - For polls, you can add a “Poll question” and a minimum of two “Poll options”.
        - You can click “+ Add an option” to have more than two options.
        - You can also delete the third and subsequent options.
    - You can then go into the poll and “Vote”.
- In the list of discussions, you can click on the up arrow (upvote the discussion).
    - You can click it again to un-upvote it.
- You can sort the discussions by:
    - Latest activity,
    - Date created, and
    - Top (most upvoted) over the past day, week, month, year, or all time.
- You can filter by label and by filter status (open, closed, locked/unlocked, answered/unanswered, and all).
- On the left-hand side, you can also filter by category, and see the users who have the most comments marked as an answer.
    - You can edit the categories by clicking on the pencil icon next to the word “Categories”.
    - You can then:
        - Add a new section
            - A category can belong to only one section, or no sections (and be at the top of the list).
        - Add a new Category, including adding the:
            - Category name,
            - Description,

        - Section, and
        - Discussion format. Choose from:
            - Open-ended discussion – no answer needed,
            - Question and Answer – vote on the best suggested answer,
            - Announcement – updates and news. It is started by maintainers and admins, but anyone can comment and reply, and
            - Poll.
    - edit or delete existing categories or sections.
- Note: when looking at an issue, you can also click on "Convert to discussion".
- In a discussion, depending on your permission, you can:
    - Manage labels
    - Subscribe/unsubscribe – receive/stop receiving notifications
    - View existing comments, and sort them: Oldest first, Newest first, or Top rated.
    - Lock/unlock the discussion (Triage permission needed)
        - This stops users without access to the repository from posting comments,
        - However, you can choose to "Allow reactions" (emojis).
    - Transfer the discussion to another repository that you own (Write permission needed)
    - Pin discussion to the top of the discussions page, or to the relevant category, for increased visibility,
        - Up to 4 important discussions can be pinned above the list of discussions.
    - Create issue from a discussion (may depend on whether it has been answered or locked),
    - Delete discussion (Triage permission needed),
    - Add comments, and click on "Comment" or "Close with comment".
    - Click on "Close discussion".
    - For existing comments, you can:
        - Click on the up arrow (upvote the comment).
        - Add an emoji reaction.
        - In Q&A categories, click "Mark as answer",
        - click on the … next to the comment and:
            - Copy link,
            - Quote reply,
            - You may be able to see "Reference in new issue".
            - Edit, Hide, and Delete comment, and
            - Report content, if they have gone against the GitHub Terms of Service or Community Guidelines, and

- Block non-member users (you can always remove member status from members). You can do this temporarily or permanently. This not only means that the user cannot add anything new, but they cannot
    - fork, watch, pin or star your organization's repositories, or
    - open issues,
    - Send, close, or merge pull requests,
    - Comment on issues, pull requests, or commits, or
    - Add/edit wiki pages.
    - If they have voted, their votes are removed.
    - Blocked users can be seen in the Organization/Personal Settings – Moderation – Blocked users.
- You will be able to see discussions that you have Created or Commented on by:
    - Clicking on the hamburger icon in the top-left hand corner, and
    - Clicking on Discussions.

44. Contribute to community projects

- A list of community projects can be found at:
    - https://github.com/github-community-projects and
    - https://github.com/topics/community-project
- Contributing suggests code contribution, rather than more general participation.
- You can:
    - Click on the Issues tab, go to Labels and click on "good first issue",
    - Create a suggest update by:
        - Fork the repository to create your own copy,
        - Optionally, clone the repository to your computer, and create a new branch,
        - update the forked/copied repository, and commit the changes,
        - Optionally, push your changes from your computer back to GitHub.
        - make a Pull Request
- You can see whether the files that should ideally be in a community project by going to a repository – Insights – Community Standards.
- Files that you should check before contributing include:
    - CONTRIBUTING.md – contribution guidelines.
    - CODE_OF_CONDUCT.md – standards for engaging in a community.
    - ISSUE_TEMPLATE.md – Issue template
    - LICENSE.md – the license for the repository.
    - README.md – information and an overview about yourself or your repository.
    - SECURITY.md – how to report security issues

    - SUPPORT.md – how to get help with the project
- You can also contribute to some projects by Sponsoring them.
    - You can generally make a monthly or a one-off monetary contribution.

