200 XP

# Knowledge check

3 minutes

## Check your knowledge

1. You want to grant a user the permissions required to add and remove organization members to and from a team. Which permission would you need to grant that user? *

- ○ The admin permission on a repository
- ○ The maintain permission on a repository
- ○ Organization billing manager
- ○ Team maintainer

✓ Correct! As a team maintainer, the user can add and remove organization members to and from a team.

2. As an organization owner, you want to ensure that everyone who is signed in to your corporate network can access the GitHub website without requiring a second sign-in. Which technology would you enable to accomplish this? *

- ○ Single sign-on

✓ Correct! Single sign-on is the right technology to allow network users to access the GitHub website without extra sign-ins.

- ○ Two-factor authentication
- ○ Personal Access Tokens
- ○ SSH keys

3. What's the appropriate repository permission level for contributors who need to actively push changes to your repository? *

- ○ admin
- ○ write

✓ Correct! The write permission is the appropriate permission level.

- ○ triage
- ○ maintain

## Next unit: Summary

Continue

200 XP

# Knowledge check

3 minutes

1. What type of user authentication is used to verify a user identity against a known identity provider? *

- [ ] Two-factor authentication (2FA)
- [ ] Time-based One-time Password (TOTP)
- [x] SAML Single Sign-on (SAML SSO)

  ✓ SAML authentication is a process used to verify user identity and credentials against a known identify provider.
- [ ] Short Message Service (SMS)

2. You're an admin and want to enable team synchronization for your organization. What installation permissions do you need to configure team synchronization for Microsoft Entra ID? *

- [ ] Provide the tenant URL
- [x] Read all users' full profiles

  ✓ To enable team synchronization for Microsoft Entra ID, the installation needs the following permissions: read all users' full profiles, sign in and read user profile, and read directory data.
- [ ] Generate a valid Single Sign-on for Web Systems (SSWS) token
- [ ] Enable SAML Single Sign-on (SSO)

  ✗ To enable team synchronization for Okta, you or your IdP administrator must: enable SAML SSO and SCIM for your organization using Okta, provide the tenant URL for your Okta instance, and generate a valid SSWS token with read-only admin permissions for your Okta installation as a service user.

3. Where does a user authenticate after enabling SAML Single sign-on? *

- [ ] With a GitHub login
- [ ] With the organization credentials

  ✗ Organization credentials are a different form of authentication.
- [x] With the Identity Provider (IdP)

  ✓ When a member accesses resources within an organization that uses SAML SSO, GitHub redirects the member to the IdP to authenticate.

4. What two-factor authentication method supports the secure backup of your authentication codes in the cloud? *

- [x] Time-based One-time Password (TOTP)

  ✓ TOTP apps support the secure backup of your authentication codes in the cloud, and can be restored if you lose access to your device.
- [ ] Short Message Service (SMS)
- [ ] Security Key

## Next unit: Summary

200 XP

# Knowledge check

5 minutes

## Check your knowledge

**1. Which directory is the clone placed after creating a Codespace? ***

- [ ] /workspaces directory

✔ **Correct. After you create a Codespace, the clone is placed into the /workspace directory.**

- [ ] /temp directory
- [ ] ~/.bashrc directory
- [ ] Linux directory

**2. What's the maximum number of Codespaces that you can create per repository or branch? ***

- [ ] You can only create two Codespaces.
- [ ] You can create a total of ten Codespaces.
- [ ] You can create a total of thirty Codespaces.
- [ ] You can create an unlimited number of Codespaces per repository or branch, depending upon available space. When you reach an upper amount of resources, a message displays that an existing Codespace needs to removed/deleted before a new Codespace can be created.

✔ **Correct. You can have an unlimited number of Codespaces per repository or even per branch. However, there are limits to the number of Codespaces that you can create and run at the same time.**

**3. What happens when Codespace loses internet connectivity? ***

- [ ] If the connection to the internet is lost while working in a Codespace, you aren't able to access your Codespace.

✔ **Correct. A Codespace requires an internet connection. If the connection to the internet is lost while working in a Codespace, you aren't able to access your Codespace.**

- [ ] Codespace doesn't require an internet connection. I can access my Codespace regardless if I lose connectivity.
- [ ] If you lose internet connection while working on your Codespace, your changes aren't saved.

**4. What defines the beginning of a Codespace's lifecycle? ***

- [ ] A Codespace's lifecycle begins when you create a Codespace and ends when you delete it.

✔ **Correct. A Codespace's lifecycle begins when you create a Codespace and ends when you delete it.**

- [ ] A Codespace's lifecycle begins immediately when GitHub is opened and ends when the software is closed.
- [ ] A Codespace's lifecycle begins when a repository is created and ends when it's deleted.

Next unit: Summary

200 XP

# Knowledge check

5 minutes

**1. How does GitHub Copilot work? ***

- [x] GitHub Copilot uses prompts and natural language text that you type to provide coding suggestions.

  ✓ **Correct! GitHub Copilot is trained on billions of lines of code. It turns natural language prompts into coding suggestions across dozens of languages.**

- [ ] GitHub Copilot uses lights, that you type, and it provides suggestions based on what you type.
- [ ] GitHub Copilot uses radio language, that you type, and it provides suggestions based on what you type.

**2. Is GitHub Copilot Free? ***

- [ ] Yes, it's free for everyone.
- [x] No, it’s a service you can sign up for that’s free for students to use but currently costs 10 dollars per month.

  ✓ **GitHub Copilot is free for verified students, teachers, and maintainers of popular open-source projects.**

- [ ] It's not free, even if you're a student or a teacher.

**3. How can you accept GitHub Copilot’s suggestions? ***

- [x] Press the `Tab` key.

  ✓ **Copilot offers you a suggestion, which appears as grey code if you use black as your text color. To accept the suggestion, you need to press the `Tab` key.**

- [ ] Press the `F1` key.
- [ ] Press the `F4` key.

**4. Identify which statement is valid and select the correct answer: ***

- [ ] A prompt, which is our output, is a collection of songs that tells our copilot what to generate.
- [x] A prompt, which is our input, is a collection of instructions or guidelines that tell our copilot what to generate.

  ✓ **A prompt is crucial in eliciting specific responses from Copilot. The prompt might be a comment that steers Copilot to generate code on your behalf or writing code that Copilot autocompletes.**

- [ ] A prompt, which is our document, is a collection of laptops that tells our Copilot what to generate.

**5. What does the quality of the output from GitHub Copilot depend on? ***

- [ ] Your code editor.
- [ ] How well your extensions were installed.
- [x] How well you crafted your prompt.

  ✓ **Designing an effective prompt is, therefore, crucial in ensuring we achieve our desired outcomes. You need to detail your prompt as much as possible.**

Check your answers

200 XP

# Knowledge check

5 minutes

## Check your knowledge

1. What is GitHub Copilot? *

- [x] GitHub Copilot is an AI pair programmer that you can use to get code suggestions.

✔ GitHub Copilot is an AI pair programmer that you can use to get suggestions for whole lines or entire functions right inside your editor.

- [ ] GitHub Copilot is OpenAI Codex, a new AI system created by OpenAI.
- [ ] GitHub Copilot is a JavaScript public repository and is one of the best supported languages.
- [ ] GitHub Copilot can write a comment describing logic and you can add your suggested code to implement the solution.

2. What are the supported integrated development environment extensions for GitHub Copilot? *

- [ ] Visual Studio Code and Visual Studio
- [ ] GitHub.com, Visual Studio Code, Visual Studio, Neovim, and JetBrains
- [x] Visual Studio Code, Visual Studio, Neovim, and JetBrains

✔ Correct! All of the IDEs listed have supported GitHub extensions.

3. What is the difference between GitHub Copilot Business and GitHub Copilot Enterprise? *

- [ ] GitHub Copilot Enterprise has code completions, while GitHub Copilot Business does not.
- [ ] GitHub Copilot Enterprise has chat in IDE and mobile, while GitHub Copilot Business does not.
- [x] GitHub Copilot Enterprise has an extra layer of personalization, with organization utilizing their own codebase to train GitHub Copilot.

✔ Correct! GitHub Copilot Enterprise allows organizations to have a smarter more tailored experience by utilizing their own codebase to train GitHub Copilot.

- [ ] GitHub Copilot Enterprise has an extra layer of security, with IP indemnity and enterprise-grade security, safety, and privacy.

✘ GitHub Copilot Business does have IP indemnity and enterprise-grade security, safety, and privacy.

## All units complete:

Complete module

200 XP

# Knowledge check

3 minutes

Choose the best response for each question, then select **Check your answers**.

## Check your knowledge

**1. Which of the following choices best describes the relationship between *open source* and *InnerSource* programs? ***

- Anyone can offer a contribution to an open source program, whereas InnerSource programs only accept contributions from members of the team that owns the repository.
- InnerSource programs are forked from open source programs by organizations that only use and maintain them privately moving forward.
- InnerSource programs are fundamentally the same as open source programs, except that their access is limited to people within their organization.

✓ **InnerSource offers all of the benefits of traditional open source patterns, but to a limited audience in order to protect intellectual property.**

**2. Suppose your team has been receiving some low-quality bug reports without enough information to properly diagnose. Which of the following choices is the best way to address the issue? ***

- Use GitHub Script to add a workflow action that automatically rejects any issues with a description fewer than 200 characters long.
- Add an `ISSUE_TEMPLATE.md` file that includes fields for reproduction steps, system properties, and instructions for generating and including important logs.

✓ **This file will ensure that anyone filing a bug knows what's expected of them at the moment they're writing the report.**

- Add a `CONTRIBUTING.md` file that clearly explains what's expected in bug reports, such as reproduction steps, system properties, and instructions for generating and including important logs.

**3. Suppose your team has been tracking data of all kinds since your InnerSource program went live three months ago. Which of the following metrics indicates your program is a great success? ***

- A dramatic rise in pull requests that address bugs in your software.

✓ **This metric indicates that more people are motivated to improve the quality of your software and are making the investment themselves.**

- A growing rate of bug reports that are quickly closed because they cannot be reproduced.
- A steady decline in new issues.

✗ **A decline in new issues may be an indication that users are satisfied with your software, or it could mean that they have low confidence that reporting issues and requesting features will produce progress. This metric alone is not enough to infer success.**

## Next unit: Summary

Continue

200 XP

# Knowledge check

3 minutes

Choose the best response for each question, then select **Check your answers**.

## Check your knowledge

**1. How does GitHub's top-level search bar differ from the search options available on repository tabs? ***

- ○ Other than being located in different parts of the user interface, they are otherwise the same.
- ○ They support different filter syntax options.
- ○ The top-level search bar supports searching everything across all of GitHub, whereas the repository tab searches are scoped to cover specific types in the current repository.

✓ **The top-level search allows the most flexibility, whereas the scoped tab searches provide popular filter dropdowns for easier refinement.**

**2. What does `git blame` do? ***

- ○ It creates a bug assigned to the last person who committed changes to the specified file.
- ○ It displays the commit history of the file.

✓ **Despite the accusatory name, `git blame` is just a command to display commit history.**

- ○ It reverts the effects of a `git praise` command.

**3. Suppose a bug issue is reported on your project, and you know which pull request introduced the problem. Which of the following options is not a cross-linking best practice? ***

- ○ Do not create cross-links when the root cause of the issue is already known.

✓ **It's a good practice to always add cross-links in case you or someone else needs the context later on.**

- ○ Add a comment to the bug report that includes the pull request's author by using an @mention.

✗ **This linking is a good practice, assuming the author is available to support the issue.**

- ○ Add a comment to the bug report that links the pull request to it using the #ID syntax.

## Next unit: Summary

Continue

200 XP

# Knowledge check

3 minutes

Choose the best response for each question, then select **Check your answers**.

## Check your knowledge

**1. Which of the following Markdown snippets would produce the text *Hello, world!* in bold italics? ***

- `*Hello, *world*!*`
- `**Hello, *world*!**`
- `***Hello, world!***`

  ✔ **Correct. Remember that you can also use underscores (`_`) instead of asterisks if you prefer.**
- `### Hello, world!`

**2. How do you print certain characters, like asterisks (`*`) and underscores (`_`), literally on your output? ***

- Use three in a row, like `***` or `___`.
- Escape them with a backslash, like `\*` or `\_`.

  ✔ **Correct. You can also escape other reserved characters, including `{` and `#`, using backslashes.**
- Unfortunately, this isn't supported at this time.

**3. Suppose there's an HTML snippet that you want to include on your GitHub Pages web site, but Markdown doesn't offer a way to render it. What should you do? ***

- Just add the HTML inline.

  ✔ **Correct. Markdown isn't a complete replacement for HTML. You might need to add HTML to get the final results you're looking for.**
- Cut the content. If it's not supported in Markdown, then it's probably not worth including.
- Open an issue that requests Markdown support for your specialized scenario.

  ✕ **Incorrect. This path would offer a low chance for success. Why not just put the HTML inline?**

## Next unit: Summary

Continue

200 XP

# Knowledge check

4 minutes

## Contribute to an open-source project using GitHub

**1. What is the best place on a GitHub repository to find where you can help a project? ***

- ○ The README file
- ○ The issues list

  ✓ **Correct. Existing issues with the project are listed here. They're also labeled according to help needed.**
- ○ The search bar
- ○ The LICENSE file

**2. What is the preferred way to ask for help or reviews on a pull request? ***

- ○ Send a negative or disrespectful comment to the project's maintainers via social media.
- ○ Create an issue
- ○ Add comment in the pull request

  ✓ **Correct. You can either directly request a review or add a comment to your pull request to discuss anything about it.**
- ○ Send an email to a random committer on the project

**3. What is needed before you can create a pull request on GitHub? ***

- ○ Send a patch file to maintainers via email
- ○ Clone a repo, commit changes, and force push
- ○ Get accepted as a team member
- ○ Fork a repo, clone it, commit changes, and push to your fork

  ✓ **Correct. While you can clone any public GitHub repository, by default you don't have the right to push any modifications. Fork the repository to create your own copy first.**

## Next unit: Summary

Continue

200 XP

# Knowledge check

3 minutes

## Check your knowledge

1. What Project descriptor automatically saves when you change it? *

- [x] Project name
  - ✓ The name of a GitHub Project is the only descriptor that automatically saves.
- [ ] Project description
- [ ] Project README

2. What does an iteration field help you do in Projects? *

- [ ] Allows you to keep track of the various changes made to an issue or pull request.
- [ ] Allows you to reverse the changes you made to your Project.
- [x] Allows you to create sequential phases of your project and group issues and pull requests based on the phase.
  - ✓ An iteration field helps you and your team organize your Project into different phases. The value of the field allows you to prioritize what comes first. Setting priorities allows you to implement timelines to keep you on track and accomplish your goals.

3. What field can you use in order to make a Priority grouping like High, Medium, and Low in your Project? *

- [ ] Date
- [x] Single select
  - ✓ The Single select field allows you to create multiple groups like Priority grouping in order to help you organize and prioritize your Project.
- [ ] Iteration field

4. What is the easiest way to add automation to your Project? *

- [ ] GraphQL API
- [x] Built-in Automation
  - ✓ Built-in automation built within Projects allows you a simple way to automate your Project.
- [ ] GitHub Actions

5. What is the name of the section where you can change the visibility of your Project, close your Project, or delete your Project? *

- [ ] Red zone
- [ ] Visibility and Access
- [x] Danger zone
  - ✓ Correct!

200 XP

# Knowledge check

3 minutes

Choose the best response for each question, then select **Check your answers**.

## Check your knowledge

**1. What is not a good reason to create a pull request? ***

- ○ You would like to receive feedback on prospective changes before merging your feature branch into `main`.
- ○ You want to merge your bug fix branch into `main`, but don't have permission.
- ○ Your branch can't be merged into `main` due to upstream changes made since you created it. Creating a pull request lets the other contributor know they need to pull their changes out so you can put yours in.

✓ **This isn't how pull requests work. Also, the etiquette is for you to be sure your branch can be cleanly merged into the base before creating a pull request.**

**2. How can you ensure that pull requests for a given area of the repository aren't merged unless certain users or teams approve? ***

- ○ Clearly explain the pull request policy in *CONTRIBUTING.md*.
- ○ Use a *CODEOWNERS* file and enable required reviews.

✓ **A *CODEOWNERS* file enables you to assign users or teams as required reviewers using the same syntax as *.gitingore* files.**

- ○ Add a table mapping directory paths to required users in *SECURITY.md*.

**3. You've been requested to review a pull request. As you read through it, you notice several minor coding errors and typos. How should you handle the review? ***

- ○ Start a review and fix obvious typos inline. Add comments in places that require further discussion or offer educational value. Complete the review with changes requested.

✓ **Contributors always appreciate when reviewers show an interest in getting their code merged.**

- ○ Leave single comments for each issue you come across, but don't change the code. For typos, include the correct spelling of the word as a reference. Approve the pull request if you trust the author to implement your suggestions.

✗ **You should only approve a pull request when it's ready to merge.**

- ○ Reject the pull request. We can't risk any bugs accidentally being merged into an important branch.

## Next unit: Summary

Continue

200 XP

# Knowledge check

3 minutes

Choose the best response for each question, then select **Check your answers.**

## Check your knowledge

**1. What's the best way to make sure you're integrating the most secure versions of your project dependencies? ***

- ○ Configure your package files to always use the latest versions of dependencies.
- ○ Check each project's security details closely before adding it to your dependencies by confirming its version status across multiple advisory sites.
- ◉ Enable **Dependabot** for your repository.

✓ **Dependabot scans your repository's dependency manifests and notifies you via pull request whenever a version you rely is marked as insecure.**

**2. Suppose one of your source projects relies on secrets kept in a folder called `.secrets`. You would like to make sure that the files kept in this folder on development machines aren't inadvertently committed to the repository. Which of these files best helps enforce this policy? ***

- ○ `SECURITY.md`
- ◉ `.gitignore`

✓ **`.gitignore` can be used to help enforce which files are included in commits by tools that respect it. However, the client enforces this policy and doesn't necessarily prevent users from committing files that violate policy.**

- ○ `CONTRIBUTING.md`

**3. What does secret scanning do? ***

- ◉ Looks for known secrets or credentials committed within the repository.

✓ **This approach is done to prevent the use of fraudulent behavior and to secure the integrity of any sensitive data.**

- ○ Analyzes and finds security vulnerabilities and errors in the code in a GitHub repository.
- ○ Secret scanning uses CodeQL to query your code as data.

## Next unit: Summary

Continue

200 XP

# Knowledge check

4 minutes

## Check your knowledge

**1. What's the difference between GitHub organization accounts and GitHub personal/user accounts? ***

- [x] Organizational accounts are shared accounts, while personal/user accounts are for individuals

  ✔ **Correct. Organizations are shared accounts where an unlimited number of people can collaborate across many projects at once. Unlike personal/user accounts, permissions with organization accounts are done at a tiered approach.**

- [ ] You pay more for organization accounts versus personal/user accounts
- [ ] They're exactly the same
- [ ] Personal/user accounts have more access than organization accounts

**2. What's the best reason to decide to upgrade to the GitHub Enterprise product? ***

- [ ] Because you want to use GitHub Actions and Codespaces
- [ ] Because your VP needs to use GitHub Insights
- [x] Because you want to centrally manage users and repositories across multiple organizations

  ✔ **Correct. GitHub Enterprise allows you to collect multiple organizations together for central management.**

- [ ] Because you want to use the team pull request reviewers feature.

**3. What's the purpose of a team? ***

- [ ] A team allows you to manage an organization account
- [ ] A team allows you to control permission levels for an enterprise
- [ ] A team allows a single user to sign in using different accounts credentials
- [x] A team is intended to reflect a company or group's structure. It's used to provide cascading access permissions and make it easy to notify all team members via mentions

  ✔ **Correct.**

**4. What's a function you can execute on GitHub Mobile? ***

- [ ] Check out branches with pull requests and view CI statuses

  ✘ **Incorrect. You can't do this on GitHub Mobile, but you can on GitHub Desktop.**

- [ ] Compare changed images
- [ ] Add and clone repositories
- [x] Manage, triage, and clear notifications from github.com

  ✔ **Correct. On GitHub Mobile you can manage, triage, and clear notifications from github.com.**

200 XP

# Knowledge check

3 minutes

Choose the best response for each question, then select **Check your answers**.

## Check your knowledge

**1. What is the best way to report a bug to a GitHub project? ***

- Send an email to a project owner.
- I don't bother reporting software bugs because there's no transparency and they never get fixed anyway.
- Search for the bug in the project's existing issues and create a new one if it hasn't been reported yet.

✓ **A project's issues are visible to anyone who has access to the project, so you might find a resolution is already planned or available. Otherwise, you can create and track the issue yourself.**

**2. Suppose you created a bug fix on a new branch and want it to become part of the next production build generated from the `main` branch. What should you do next? ***

- Copy your branch changes and commit them directly to the `main` branch.
- Create a pull request to merge your new branch into the `main` branch.

✓ **Pull requests are the correct way to communicate that commits are ready for review and ultimate inclusion on the `main` branch.**

- On second thought, maybe I won't share this fix. I'll just put it in my own private version of the source code.

✗ **You should reconsider offering your contribution to the original project. By getting your bug fix merged into the `main` branch, you'll benefit from it being automatically included in future builds.**

**3. Suppose you'd like to work with a project on GitHub, but you don't have write access to the project. What can you do to contribute? ***

- Fork the project's repository to your GitHub account, clone the forked repository to your local machine, push changes to your repository, and submit a pull request to the target (upstream) repository.

✓ **GitHub provides forking functionality designed to allow you to work with projects where you aren't an owner or don't have write access. Forking makes a remote copy of the project in your repository that you can then clone locally. To submit updates to the target repository (upstream repository), you can submit a pull request.**

- Clone the project to your local machine and push updates directly to the project repository.
- Use git commands to make a copy of the project so that you can work locally. Submit an issue to get your changes into the target repository.

## Next unit: Summary

Continue

200 XP

# Knowledge check

5 minutes

**1. Which of the following scenarios is a common use case for a version control system? ***

- ( ) Deleting earlier versions of a project or file, so you know you are working only with the most current file or data.

  ✗ **Incorrect. Version control systems purposefully retain earlier versions of a file or project, so you can access them, if needed.**

- ( ) Making experimental changes to your project in an isolated branch.

  ✓ **Correct! Using branches to create different sets of changes to a project is a key use case for version control.**

- ( ) Gathering feature requirements for a large project and communicating them to stakeholders.

**2. What is another name for a version control system? ***

- ( ) Version management software (VMS)
- ( ) Software control management (SCM) system
- ( ) Software configuration management (SCM) system

  ✓ **Correct!**

**3. What's the difference between Git and GitHub? ***

- ( ) Git lets you work with one or more local branches and push changes to a remote repository. GitHub acts as the remote repository, which is accessed through a website or command-line tools.

  ✓ **Correct! Git is the tool you can use to work with a local branch and push changes to a remote repository. GitHub acts as the remote repository.**

- ( ) Git is a distributed version control system (DVCS) that runs in the cloud. GitHub is an interface layer that provides access to Git technology.
- ( ) Git is used by an individual contributor. GitHub is used by multiple contributors to simplify group development work.

**4. What Git command gives information about how to use Git? ***

- ( ) `git init`
- ( ) `git status`
- ( ) `git help`

  ✓ **Correct! Use `git help` to view information about how to use Git.**

## Next unit: Summary

Continue >

100 XP

# Basic Git commands

10 minutes

Git works by remembering the changes to your files as if it's taking snapshots of your file system.

We'll cover a few basic commands to start tracking files in your repo. Then, you'll save your first "snapshot" for Git to compare against.

## git status

The first and most commonly used Git command is `git status`. You've already used it once, in the preceding exercise, to see that you had initialized your Git repo properly.

`git status` displays the state of the working tree (and of the staging area—we'll talk more about the staging area soon). It lets you see which changes are currently being tracked by Git, so you can decide whether you want to ask Git to take another snapshot.

## git add

`git add` is the command you use to tell Git to start keeping track of changes in certain files.

The technical term is *staging* these changes. You'll use `git add` to stage changes to prepare for a commit. All changes in files that have been added but not yet committed are stored in the *staging area*.

## git commit

After you've staged some changes for commit, you can save your work to a snapshot by invoking the `git commit` command.

*Commit* is both a verb and a noun. It has essentially the same meaning as when you commit to a plan or commit a change to a database. As a verb, committing changes means you put a copy (of the file, directory, or other "stuff") in the repository as a new version. As a noun, a commit is the small chunk of data that gives the changes you committed a unique identity. The data that's saved in a commit includes the author's name and e-mail address, the date, comments about what you did (and why), an optional digital signature, and the unique identifier of the preceding commit.

## git log

The `git log` command allows you to see information about previous commits. Each commit has a message attached to it (a commit message), and the `git log` command prints information about the most recent commits, like their time stamp, the author, and a commit message. This command helps you keep track of what you've been doing and what changes have been saved.

## git help

You've already tried out the `git help` command, but it's worth reminding you about. Use this command to easily get information about all the commands you've learned so far, and more.

Remember, each command comes with its *own* help page, too. You can find these help pages by typing `git <command> --help`. For example, `git commit --help` brings up a page that tells you more about the `git commit` command and how to use it.

---

## Next unit: Knowledge check

Continue