Latest Version: 6.0

Question: 1

How are commits related to pull requests?

A. Commits are made on a branch that can have a linked pull request.
B. Commits can only be made after a pull request is created.
C. Commits can only be made before a pull request is created.
D. Commits are made on a pull request that can have a linked branch.

Answer: A

Explanation:
Commits and pull requests (PRs) are fundamental concepts in Git and GitHub workflows, particularly in collaborative software development.
Commits:
Commits are individual changes or updates made to the codebase. Each commit is identified by a unique SHA-1 hash and typically includes a commit message describing the changes.
Commits are made to a specific branch in the repository. The branch could be the main branch, or more commonly, a feature branch created for specific work or a feature.
Pull Requests (PRs):
A pull request is a mechanism for developers to notify team members that a branch is ready to be merged into another branch, usually the main branch.
PRs are used to review code, discuss changes, and make improvements before the branch is merged into the target branch.
Relationship Between Commits and PRs:
Option A is correct because commits are made on a branch, and this branch can have a pull request associated with it. The pull request tracks the branch's commits and allows for code review before merging into the target branch.
Commits can be added to the branch both before and after the pull request is created. Any new commits pushed to the branch are automatically included in the pull request.
Incorrect Options:
Option B is incorrect because commits can be made both before and after a pull request is created.
Option C is incorrect because it suggests that commits can only be made before a pull request is created, which is not true.
Option D is incorrect because commits are not made on a pull request; they are made on a branch. The pull request links a branch to another branch (e.g., feature branch to the main branch).
Reference:
GitHub Documentation: About Pull Requests
GitHub Docs: Understanding the GitHub Flow
Git Documentation: Git Basics - Getting a Git Repository

Question: 2

What is the difference between an organization member and an outside collaborator?

A. Organization base permissions do not apply to outside collaborators.
B. Two-factor authentication (2FA) is not required for outside collaborators.
C. Outside collaborators cannot be given the admin role on a repository.
D. Outside collaborators do not consume paid licenses.

Answer: A

Explanation:
In GitHub, an organization member is a user who has been added to an organization and is subject to the organization's base permissions and policies. An outside collaborator is a user who is not a member of the organization but has been granted access to one or more repositories within the organization.
Here's the difference between an organization member and an outside collaborator:
Organization Members:
Members are subject to the organization's base permissions, which apply across all repositories within the organization. These permissions might include read, write, or admin access, depending on what has been set as the default.
Members consume paid licenses if the organization is on a paid plan.
Members are required to have two-factor authentication (2FA) if the organization enforces it.
Outside Collaborators:
Outside collaborators do not have organization-wide permissions. They only have access to specific repositories to which they have been granted permission. This means organization base permissions do not apply to them (making option A correct).
Outside collaborators do not consume paid licenses. They are only counted toward the license if they are made organization members.
Outside collaborators can be granted any level of permission, including the admin role on specific repositories.
Two-factor authentication (2FA) can be enforced for outside collaborators at the repository level, depending on the organization's security settings.
Given this information, option A is the correct answer: "Organization base permissions do not apply to outside collaborators."
Reference:
GitHub Documentation: Roles in an organization
GitHub Documentation: About outside collaborators
GitHub Documentation: Managing repository access for your organization

Question: 3

What are the defining features of Git?

A. Distributed version control, open source software, and being designed for handling projects of any size with efficiency
B. Sequential version control, cloud-based hosting service, and being designed for collaboration on large projects
C. Low-cost local branching, convenient staging areas, multiple workflows, and being designed for managing small projects
D. Centralized version control, proprietary software, and being designed for small projects

Answer: A

Explanation:
Git is a widely-used version control system that has several defining features:
Distributed Version Control:
Git is a distributed version control system, meaning that every developer has a full copy of the entire repository, including its history, on their local machine. This enables greater flexibility, as work can be done offline and each user has access to the full project history.
Open Source Software:
Git is open-source, meaning its source code is freely available for use, modification, and distribution. This fosters a large community of users and contributors who continuously improve the software.
Efficiency with Large Projects:
Git is designed to handle projects of any size with speed and efficiency. It can manage large codebases and many contributors without significant performance degradation, making it suitable for everything from small personal projects to large, complex software systems.
Incorrect Options:
Option B is incorrect because Git is not a sequential version control system, nor is it inherently tied to cloud-based services. GitHub, GitLab, and other platforms offer cloud hosting for Git repositories, but Git itself is a version control tool.
Option C is incorrect because Git is not limited to small projects; it is designed to scale efficiently, and the other features mentioned are only partial descriptions of Git's capabilities.
Option D is incorrect because Git is not a centralized version control system; it is distributed. Additionally, Git is open-source, not proprietary, and is used for projects of all sizes.
Reference:
Pro Git Book: What is Git?
Git Documentation: Distributed Version Control
GitHub Docs: Understanding the Git Workflow

Question: 4

Who can be assigned to an Issue or pull request?
(Each answer presents a complete solution. Choose two.)

A. Anyone who has an enterprise GitHub account
B. Anyone who has commented on the Issue or pull request
C. Anyone who has a personal GitHub account
D. Anyone with write permissions to the repository

Answer: B, D

Explanation:
In GitHub, issues and pull requests (PRs) are essential tools for managing work and collaboration in a project. Assigning individuals to these issues or PRs is a way to indicate responsibility for addressing the issue or completing the PR.
Anyone with write permissions to the repository:
Users who have write permissions to a repository can be assigned to issues and pull requests. Write permissions allow users to push changes to the repository, create branches, and modify issues and pull requests. Assigning them to an issue or PR ensures they are recognized as responsible for the task.
Anyone who has commented on the Issue or pull request:
GitHub allows you to assign issues or pull requests to users who have already engaged with the discussion by commenting on it. This feature is particularly useful for quickly assigning tasks to those who are already involved in the conversation.
Incorrect Options:
Option A is incorrect because having an enterprise GitHub account alone does not necessarily grant the ability to be assigned to issues or PRs. Permission to assign is based on repository-specific roles and permissions.
Option C is incorrect because not all personal GitHub accounts can be assigned to issues or PRs. The user needs either write permissions to the repository or must have commented on the issue or PR.
Reference:
GitHub Docs: Assigning Issues and Pull Requests
GitHub Docs: Permission Levels for a Repository
This detailed explanation provides clarity on GitHub's assignment mechanics for issues and pull requests, reflecting the platform's collaborative nature.

Question: 5

Which of the following is the best GitHub feature for long-form documentation for a project?

A. Insights
B. Pull Requests
C. Projects
D. Wikis

Answer: D

Explanation:
GitHub offers a variety of features for different aspects of project management and documentation. For long-form documentation, the best feature is Wikis. Wikis in GitHub allow you to create detailed, structured documentation that is easy to navigate and edit. Each repository in GitHub can have its own Wiki, which acts as a space for collaborators to maintain project documentation, guides, manuals, or any other long-form content.
Wikis are specifically designed to host extensive documentation in a way that is easy to reference and edit over time. They support Markdown, allowing you to format your documentation effectively. Unlike

the other options, Wikis are explicitly intended for the purpose of long-form content, making them the best choice for this use case.