100 Intermediate Level Git MCQ Questions with Answers

What is Git?

1. **Which of the following best describes Git?** A) A centralized version control system B) A distributed version control system C) A remote code hosting platform D) An integrated development environment **Answer: B) A distributed version control system**
2. **In which year was Git first released?** A) 2001 B) 2005 C) 2008 D) 2010 **Answer: B) 2005**
3. **Who created Git?** A) Linus Torvalds B) Bill Gates C) Dennis Ritchie D) James Gosling **Answer: A) Linus Torvalds**
4. **Which of the following is NOT a characteristic of Git?** A) Fast operation B) Distributed development C) Mandatory central repository D) Data integrity **Answer: C) Mandatory central repository**
5. **What is the file extension of a git repository's configuration file?** A) .gitignore B) .gitrc C) .gitconfig D) .git/config **Answer: D) .git/config**
6. **Which of the following terms best describes Git's data model?** A) Delta-based B) Snapshot-based C) Block-based D) Stream-based **Answer: B) Snapshot-based**
7. **What does SHA-1 refer to in Git?** A) A security protocol B) A compression algorithm C) A hash function used to identify commits D) A network transmission protocol **Answer: C) A hash function used to identify commits**
8. **What is a Git repository?** A) An online backup of your code B) A collection of code files stored on GitHub C) A data structure storing metadata and objects for a project D) A folder containing only source code files **Answer: C) A data structure storing metadata and objects for a project**

9. **In Git terminology, what is a "working tree"?** A) The hierarchy of branches in a repository B) The directory on your filesystem where your files are stored C) A representation of commit history D) A visual graph of the Git workflow **Answer: B) The directory on your filesystem where your files are stored**
10. **Which of these statements about Git is FALSE?** A) Git stores the entire history of a project B) Git requires constant internet connectivity C) Git tracks content rather than files D) Git uses a directed acyclic graph for commit history **Answer: B) Git requires constant internet connectivity**

How to Install Git on Windows?

11. **What is the recommended way to install Git on Windows?** A) Using npm install B) Using the official Git for Windows installer C) Via Windows Store only D) Through Docker containers **Answer: B) Using the official Git for Windows installer**
12. **Which of these is NOT included in a standard Git for Windows installation?** A) Git Bash B) Git GUI C) Git Visual Studio Code Extension D) Git CMD **Answer: C) Git Visual Studio Code Extension**
13. **When installing Git on Windows, which setting is recommended for line ending conversions?** A) Checkout Windows-style, commit Unix-style B) Checkout as-is, commit as-is C) Checkout Unix-style, commit Windows-style D) Convert all line endings to Windows format **Answer: A) Checkout Windows-style, commit Unix-style**
14. **What does the "Git Credential Manager" do when installed with Git for Windows?** A) Stores passwords in plain text B) Integrates with Windows authentication C) Securely stores authentication credentials for remote repositories D) Creates random passwords for repositories **Answer: C) Securely stores authentication credentials for remote repositories**
15. **After installing Git on Windows, how can you verify the installation was successful?** A) Check if a Git icon appears on the desktop B) Run `git --version` in command prompt C)

Look for Git in the Windows Registry D) Restart the computer and check for Git in Start Menu **Answer: B) Run `git --version` in command prompt**

16. **Which of these shells can be used with Git for Windows?** A) Bash only B) PowerShell only C) CMD only D) All of the above **Answer: D) All of the above**
17. **What does the "Git LFS" option in the Git for Windows installer provide?** A) Low Frequency Synchronization B) Large File Storage C) Local File System integration D) Limited Feature Set **Answer: B) Large File Storage**
18. **Which component enables Git integration with Windows Explorer?** A) Git Explorer Extension B) Windows Git Client C) Git Shell Extensions D) Git Credential Manager **Answer: C) Git Shell Extensions**
19. **What is the default terminal emulator that comes with Git for Windows?** A) PowerShell B) MinTTY C) Windows Terminal D) Command Prompt **Answer: B) MinTTY**
20. **Which of these PATH environment settings is recommended during Git for Windows installation?** A) Use Git from Git Bash only B) Use Git from the Windows Command Prompt C) Use Git and Unix tools from the Windows Command Prompt D) All of the above are valid options depending on your needs **Answer: D) All of the above are valid options depending on your needs**

What is GitHub?

21. **Which of the following best describes GitHub?** A) A version control system B) A cloud-based hosting service for Git repositories C) A programming language D) A local development environment **Answer: B) A cloud-based hosting service for Git repositories**
22. **What is a GitHub "fork"?** A) A hardware device for Git hosting B) A branch within a repository C) A personal copy of someone else's repository D) A merge conflict resolution tool **Answer: C) A personal copy of someone else's repository**

23. **Which feature allows you to propose changes to a repository on GitHub?** A) Issue B) Pull Request C) Push Request D) Commit Request **Answer: B) Pull Request**
24. **What is GitHub Actions primarily used for?** A) Authenticating users B) Automating workflows and CI/CD pipelines C) Creating documentation D) Managing billing information **Answer: B) Automating workflows and CI/CD pipelines**
25. **What is GitHub Pages used for?** A) Storing large files B) Hosting static websites directly from a repository C) Managing project issues D) Repository analytics **Answer: B) Hosting static websites directly from a repository**
26. **Which of these is NOT a valid GitHub repository visibility option?** A) Public B) Private C) Internal D) Protected **Answer: D) Protected**
27. **What is a GitHub Gist?** A) A mini-repository for sharing code snippets B) A GitHub mascot C) A type of GitHub organization D) A premium feature for enterprise accounts **Answer: A) A mini-repository for sharing code snippets**
28. **What is GitHub Codespaces?** A) A code review tool B) A cloud-based development environment C) A file compression utility D) A version control protocol **Answer: B) A cloud-based development environment**
29. **What does the GitHub "Dependabot" feature do?** A) Manages repository dependencies B) Automatically fixes bugs in your code C) Monitors pull request activity D) Deploys code to production **Answer: A) Manages repository dependencies**
30. **Which of these is NOT part of GitHub's social coding features?** A) Following users B) Starring repositories C) Direct messaging D) Watching repositories **Answer: C) Direct messaging**

Git Commands

31. **Which Git command creates a new local repository?** A) `git create` B) `git init` C) `git start` D) `git new` **Answer: B) `git init`**

32. **What does the command `git status` do?** A) Shows the connection status to remote repository B) Shows the working tree status C) Shows the status of all branches D) Shows the status of the Git application **Answer: B) Shows the working tree status**
33. **Which command stages changes for a specific file?** A) `git stage filename` B) `git add filename` C) `git commit filename` D) `git track filename` **Answer: B) `git add filename`**
34. **What does `git commit -m "message"` do?** A) Adds a message to the last commit B) Modifies the previous commit message C) Creates a new commit with the specified message D) Sends a message to other repository contributors **Answer: C) Creates a new commit with the specified message**
35. **Which command shows the commit history?** A) `git history` B) `git log` C) `git show` D) `git commits` **Answer: B) `git log`**
36. **What does `git reset --hard HEAD~1` do?** A) Resets the working directory to the state of the last commit B) Removes the last commit and discards changes C) Hard resets the repository to its initial state D) Resets the permissions of the HEAD file **Answer: B) Removes the last commit and discards changes**
37. **Which command creates a new branch?** A) `git create-branch branch-name` B) `git branch branch-name` C) `git checkout -b branch-name` D) Both B and C are correct **Answer: D) Both B and C are correct**
38. **What does `git checkout -- filename` do?** A) Creates a new file B) Changes to a different branch C) Discards changes in the working directory for that file D) Shows the file's commit history **Answer: C) Discards changes in the working directory for that file**
39. **Which command is used to download changes from a remote repository without merging them?** A) `git pull` B) `git fetch` C) `git download` D) `git sync` **Answer: B) `git fetch`**

40. **What does `git stash` do?** A) Permanently deletes uncommitted changes B) Temporarily stores uncommitted changes to clean the working directory C) Compresses the Git repository D) Backs up committed changes **Answer: B) Temporarily stores uncommitted changes to clean the working directory**
41. **Which command shows the differences between the working directory and the index?** A) `git compare` B) `git diff` C) `git changes` D) `git status -v` **Answer: B) `git diff`**
42. **What does `git rebase` typically do?** A) Changes the base directory of a repository B) Rebuilds the database of commits C) Reapplies commits on top of another base tip D) Restores the repository from a backup **Answer: C) Reapplies commits on top of another base tip**
43. **Which command is used to tag a specific commit?** A) `git marker` B) `git tag` C) `git label` D) `git version` **Answer: B) `git tag`**
44. **What does `git cherry-pick <commit-hash>` do?** A) Removes a specific commit B) Applies the changes from a specific commit to the current branch C) Finds "cherry" commits that are easy to merge D) Picks a random commit for review **Answer: B) Applies the changes from a specific commit to the current branch**
45. **Which command configures global user information for Git?** A) `git --global user.config` B) `git profile --set` C) `git config --global user.name "Name"` D) `git init --user "Name"` **Answer: C) `git config --global user.name "Name"`**

Git vs. GitHub

46. **Which of the following correctly describes the relationship between Git and GitHub?** A) Git is a frontend for GitHub B) GitHub is a version control system, Git is a hosting platform C) Git is a version control system, GitHub is a hosting platform D) They are different names for the same software **Answer: C) Git is a version control system, GitHub is a hosting platform**

47. **Which of these features is specific to GitHub and NOT a core Git feature?** A) Commit history B) Pull requests C) Branching D) Merging **Answer: B) Pull requests**
48. **What can you do with Git that you cannot do with GitHub alone?** A) Host public repositories B) Work on code offline without internet C) Create organizations D) Track issues **Answer: B) Work on code offline without internet**
49. **Which is a valid alternative to GitHub for hosting Git repositories?** A) Subversion B) Mercurial C) GitLab D) CVS **Answer: C) GitLab**
50. **How do Git commits differ from GitHub commits?** A) They use different hash algorithms B) There is no difference; GitHub just displays Git commits C) Git commits are local, GitHub commits are remote D) GitHub commits include additional metadata for the web interface **Answer: B) There is no difference; GitHub just displays Git commits**
51. **Which feature is NOT provided natively by Git but is available on GitHub?** A) Code reviews B) Commit history C) Branch creation D) Conflict resolution **Answer: A) Code reviews**
52. **What would be a valid reason to use Git without GitHub?** A) When you need pull requests B) When working on private code that shouldn't be hosted online C) When you need issue tracking D) When you need continuous integration **Answer: B) When working on private code that shouldn't be hosted online**
53. **Which authentication method is specific to GitHub and not part of core Git?** A) SSH keys B) Personal Access Tokens C) HTTPS D) Git credentials **Answer: B) Personal Access Tokens**
54. **In terms of collaboration, which statement is most accurate?** A) Git has more collaboration features than GitHub B) GitHub extends Git's collaboration capabilities with additional features C) Git and GitHub have identical collaboration features D) Neither Git nor GitHub are designed for collaboration **Answer: B) GitHub extends Git's collaboration capabilities with additional features**
55. **Which operation can be performed with Git but NOT directly through the GitHub web interface?** A) Creating branches B) Interactive rebase C) Viewing commit history D) Editing files

Answer: B) Interactive rebase

What is GitLab?

56. **Which of the following best describes GitLab?** A) A Git command-line tool B) A web-based DevOps platform that provides Git repository management C) A lightweight alternative to Git D) A graphical user interface for Git **Answer: B) A web-based DevOps platform that provides Git repository management**
57. **What differentiates GitLab from GitHub?** A) GitLab cannot host private repositories B) GitLab can be self-hosted on your own servers C) GitLab doesn't support Git version control D) GitLab is only available for Linux **Answer: B) GitLab can be self-hosted on your own servers**
58. **Which of the following is a feature of GitLab CI/CD?** A) Automatic code review B) Automatic repository creation C) Automated testing and deployment pipelines D) Automatic code generation **Answer: C) Automated testing and deployment pipelines**
59. **What is GitLab Runner?** A) A physical device that hosts GitLab B) An application that runs CI/CD jobs C) A programming competition hosted by GitLab D) A tool for measuring repository performance **Answer: B) An application that runs CI/CD jobs**
60. **In GitLab, what is a "merge request"?** A) A request to create a new repository B) GitLab's equivalent of a GitHub pull request C) A request to join a project D) A request to merge two GitLab accounts **Answer: B) GitLab's equivalent of a GitHub pull request**
61. **What file defines GitLab CI/CD pipeline configuration?** A) gitlab-config.yml B) .gitlab.yml C) .gitlab-ci.yml D) ci-config.yaml **Answer: C) .gitlab-ci.yml**
62. **Which is NOT an edition of GitLab?** A) Community Edition B) Enterprise Edition C) Ultimate Edition D) Professional Edition **Answer: D) Professional Edition**
63. **What is GitLab Pages used for?** A) Documentation management B) Static website hosting C) Page view analytics D) File storage **Answer: B) Static website hosting**

64. **What feature allows GitLab to automatically detect and report on security vulnerabilities?** A) GitLab Monitor B) GitLab Security Scanner C) GitLab Secure D) GitLab Guardian **Answer: C) GitLab Secure**

65. **Which of these is unique to GitLab compared to GitHub?** A) Issue tracking B) Built-in container registry C) Repository forking D) Pull/merge requests **Answer: B) Built-in container registry**

Git Clone Commands

66. **What does the `git clone` command do?** A) Creates a duplicate branch within the same repository B) Creates a local copy of a remote repository C) Creates a new empty repository D) Copies only the latest commit from a repository **Answer: B) Creates a local copy of a remote repository**

67. **How do you clone a specific branch from a remote repository?** A) `git clone -b branch-name repository-url` B) `git clone --branch=branch-name repository-url` C) Both A and B D) `git clone repository-url/branch-name` **Answer: C) Both A and B**

68. **What is a shallow clone in Git?** A) A clone with limited permissions B) A clone with truncated history C) A clone of only specific files D) A clone without remote references **Answer: B) A clone with truncated history**

69. **Which command creates a shallow clone with only the most recent commit?** A) `git clone --depth=1 repository-url` B) `git clone --shallow repository-url` C) `git clone --recent repository-url` D) `git clone --latest repository-url` **Answer: A) `git clone --depth=1 repository-url`**

70. **What does `git clone --bare` do?** A) Clones without checking out a working copy B) Clones without any Git metadata C) Clones only the .git directory without the working directory D) Clones without any branches **Answer: C) Clones only the .git directory without the working directory**

71. **How can you clone a repository to a specific directory?** A) `git clone repository-url directory-name` B) `git clone --dir=directory-name repository-url` C) `git clone repository-url --target=directory-name` D) `git clone --path directory-name repository-url` **Answer: A) `git clone repository-url directory-name`**
72. **What does the `--mirror` option do when cloning?** A) Creates a mirror image of all files B) Clones all remote references and keeps them updated C) Creates a read-only copy D) Duplicates the repository in multiple locations **Answer: B) Clones all remote references and keeps them updated**
73. **Which option limits the clone operation's bandwidth usage?** A) `--throttle` B) `--limit` C) `--bandwidth` D) `--filter` **Answer: D) `--filter`**
74. **What happens when you clone a repository using HTTPS URL?** A) You will always need to provide username and password B) You might need to provide credentials depending on the repository's visibility C) Authentication is never required D) SSH keys are automatically used **Answer: B) You might need to provide credentials depending on the repository's visibility**
75. **How can you clone a Git repository without automatically checking out any files?** A) `git clone --no-checkout repository-url` B) `git clone --empty repository-url` C) `git clone --metadata-only repository-url` D) `git clone --no-files repository-url` **Answer: A) `git clone --no-checkout repository-url`**

Git Push Commands

76. **What does the command `git push origin main` do?** A) Pulls changes from the main branch B) Pushes committed local changes to the main branch on the origin remote C) Creates a new main branch on the origin remote D) Merges the origin branch into the main branch **Answer: B) Pushes committed local changes to the main branch on the origin remote**

77. **What is the purpose of the `--force` or `-f` option with git push?** A) Increases the upload speed B) Pushes only specific commits C) Forces the update even if it's not a fast-forward D) Creates a new branch if it doesn't exist **Answer: C) Forces the update even if it's not a fast-forward**

78. **What does `git push --tags` do?** A) Pushes all local tags to the remote repository B) Tags the push with a timestamp C) Creates a new tag for the push D) Pushes all branches with their tags **Answer: A) Pushes all local tags to the remote repository**

79. **How do you push all local branches to a remote repository?** A) `git push origin all` B) `git push --all origin` C) `git push origin --branches` D) `git push origin master` **Answer: B) `git push --all origin`**

80. **What does `git push -u origin feature-branch` do?** A) Updates the feature-branch on origin B) Pushes and sets upstream tracking for feature-branch C) Creates a new upstream branch D) Pushes uncommitted changes to feature-branch **Answer: B) Pushes and sets upstream tracking for feature-branch**

81. **Which command would you use to push a specific commit to a remote branch?** A) `git push origin commit-hash:branch-name` B) `git push origin @commit-hash` C) `git push origin HEAD~1:branch-name` D) `git push --commit=hash origin branch-name` **Answer: A) `git push origin commit-hash:branch-name`**

82. **What does `git push --dry-run` do?** A) Pushes only dry code without comments B) Shows what would be pushed without actually pushing C) Pushes without requiring review D) Pushes only code changes, not binary files **Answer: B) Shows what would be pushed without actually pushing**

83. **When would you use `git push --force-with-lease`?** A) When pushing to a protected branch B) To force push only if the remote branch hasn't been updated C) To push to multiple

remotes simultaneously D) To push with specific file permissions **Answer: B) To force push only if the remote branch hasn't been updated**

84. **What does `git push origin :branch-name` do?** A) Pushes the local branch to remote B) Renames the remote branch C) Deletes the remote branch D) Creates an empty branch on remote **Answer: C) Deletes the remote branch**
85. **Which of these is NOT a valid Git push configuration?** A) simple B) matching C) upstream D) automatic **Answer: D) automatic**

Git Pull Commands

86. **What does the `git pull` command do?** A) Fetches from and merges with another repository or branch B) Only downloads changes from a remote repository C) Removes changes from your local repository D) Uploads local changes to a remote repository **Answer: A) Fetches from and merges with another repository or branch**
87. **Which command is equivalent to `git pull origin main`?** A) `git fetch origin main && git rebase origin/main` B) `git fetch origin main && git merge origin/main` C) `git checkout origin/main` D) `git update origin main` **Answer: B) `git fetch origin main && git merge origin/main`**
88. **What does `git pull --rebase` do?** A) Fetches changes and rebases instead of merging B) Creates a new branch before pulling C) Only pulls if rebase is necessary D) Rebases the remote branch onto local **Answer: A) Fetches changes and rebases instead of merging**
89. **How can you pull from a remote repository without merging changes?** A) `git pull --no-merge` B) `git fetch` C) `git pull --dry-run` D) `git pull --download-only` **Answer: B) `git fetch`**
90. **What does `git pull --ff-only` do?** A) Fast-forwards the branch if possible, otherwise fails B) Always performs a fast-forward merge C) Pulls only fast files (text files) D) Creates a fast branch

for testing **Answer: A) Fast-forwards the branch if possible, otherwise fails**

91. **Which of these is NOT a valid option for git pull?** A) `--quiet` B) `--verbose` C) `--fast` D) `--autostash` **Answer: C) `--fast`**

92. **What happens if you run `git pull` when you have uncommitted changes?** A) Git automatically stashes your changes and reapplies them after pull B) Git refuses to pull if the changes conflict with remote changes C) Git discards your local changes D) Git creates a new branch for your changes **Answer: B) Git refuses to pull if the changes conflict with remote changes**

93. **How can you pull changes while automatically stashing and reapplying local changes?** A) `git pull --stash` B) `git pull --autostash` C) `git pull --preserve` D) `git pull --keep-local` **Answer: B) `git pull --autostash`**

94. **What is the difference between `git pull` and `git pull --prune`?** A) `--prune` removes any remote-tracking references that no longer exist on the remote B) `--prune` removes all local branches C) `--prune` optimizes the repository size after pulling D) `--prune` pulls only modified files, not new files **Answer: A) `--prune` removes any remote-tracking references that no longer exist on the remote**

95. **Which command would pull from all remotes at once?** A) `git pull --all` B) `git pull --remotes` C) This is not possible with a single command D) `git pull origin other-remote another-remote` **Answer: C) This is not possible with a single command**

Git History, Branching and Merging, and Conflict Resolution

96. **What command is used to view a graph of the commit history?** A) `git graph` B) `git log --graph` C) `git history --visual` D) `git show --tree` **Answer: B) `git log --graph`**

97. **Which strategy does Git use by default when merging branches?** A) Fast-forward if possible, recursive if not B) Always create a merge commit C) Rebase and apply D) Cherry-pick

commits **Answer: A) Fast-forward if possible, recursive if not**

98. **What identifies a merge conflict in a file?** A) Error messages in the terminal B) Conflict markers like `<<<<<<<`, `=======`, and `>>>>>>>` C) Files marked with "C" in git status D) Red highlighting in Git GUI **Answer: B) Conflict markers like `<<<<<<<`, `=======`, and `>>>>>>>`**

99. **What is a "fast-forward" merge in Git?** A) A merge that happens quickly B) A merge where the head pointer simply moves forward C) A merge that skips commit verification D) A merge that only applies recent commits **Answer: B) A merge where the head pointer simply moves forward**

100. **Which command is used to abort a merge that has conflicts?** A) `git merge --abort` B) `git reset --merge` C) `git cancel-merge` D) `git checkout --abort` **Answer: A) `git merge --abort`**