1. What is git? Give us a brief history on it.

Git is a distributed version control system (DVCS). Git was developed in 2005 by the Linux kernel development community when their previous VCS (Bitkeeper) lost its free-of-charge status. The Linux community was able to develop Git from lessons learned while using Bitkeeper. Git was designed on the following principles: speed, simplicity, support for non-linear development, fully distributed, and able to handle large project efficiently.

1. What are the differences between git and SVN?

Git is decentralized. Unlike svn, git allows us to commit files to our local repository without using the internet. Once an internet connection is reestablished, the user would be able to commit that work to the main repository.

1. What is the purpose of the .gitignore file?

Gitignore is a file that specifies intentionally untracked files that Git should ignore. The purpose of gitignore files is to ensure that certain files not tracked by Git will remain untracked. Typically, you would want to add files like your logs or items that are generated automatically by your build system that you wouldn’t normally want to keep versioning history on.

1. Explain the following commands:



* 1. Commit – Records changes to the local repository
  2. Push – Allows you to share changes made to your local repository with others that are accessing the remote repository
  3. Pull – Fetches changes made to the remote repository and merges those changes with the current branch of your local repository (updates your local repository with changes from the remote repository)
  4. Merge – Merges one or more branches into your current branch while preserving the ancestry of each commit history
  5. Rebase – Allows you to easily change a series of commits, reordering, editing, or squashing commits together into a single commit (used to rewrite the history of commits)
  6. Fetch – Fetches from the remote repository and creates a local copy of the remote branch, does not merge it with the rest of the repository

1. What is the stash, and how is it used?

The Git stash command allows you to “stash” your changes on a project instead of having to commit unfinished work in the event that you want to switch branches to work on something else. It stashes it away onto a stack and is able to be called later by popping it off.

1. What GUI programs are available for Windows? Mac?

Windows – Git Extensions

Mac – GitX-dev, Tower, Gitbox

Windows and Mac – GitEye, Source Tree, git-cola, SmartGit

1. What repository providers are available online?

Assembla, Beanstalk, Bettercodes.org, Bitbucket.org, CloudHost, Codebase, CloudForge, Deveo, GitEnterprise, GitHub, GitLab Cloud, Gitorious, Pikacode, ProjectLocker, Repo.or.cz, SourceForge.net, Unfuddle

1. What are best practices for branching when developing several features at once?

* Try to avoid committing unfinished work to your repository’s master working branch.
* Create a branch any time you begin non-trivial work including features and complex bug-fixes.
* Don’t forget to delete feature branches once they were merged into stable branch.
* Use your repository’s default working branch as your “development” branch.
* Create a branch for each environment, including staging and production.
* Never merge into an environment branch unless you are ready to deploy to that environment.

1. What is and how do you make a pull request?

A pull request fetches changes made to the remote repository by others and tries to merge the changes to the current branch of your local repository.

Example: git pull origin

Give an example of how to:

1. Fork a public, open-source, project for your use
2. Merge an update to that project into your fork

Here is example

1. Creating and merging a pull request into a repository