# Intermediate Version Control: Knowledge-based Questions

**Assessment Resources:**

|  |
| --- |
| Marking key available for lecturer via Blackboard.  Students may refer to the lecture material in formulating their answers.  Students may use Git Bash or Git CMD to help answer questions |

**Assessment Instructions:**

|  |
| --- |
| Students must attempt all questions. Answer succinctly using full sentences. At most two paragraphs are expected per answer.  All answers must be at the student’s own words – use of ChatGPT or other AI tools is **strictly** prohibited.  Assessment must be completed in class unless alternative arrangements are made.  You will be given up to 180 minutes to complete the assessment |

Answer **all** the questions:

1. **Distributed and Centralised Version Control Systems**: Can you differentiate between distributed and centralized version control systems? Provide an example of each.

Centralised version control system has a copy of the source code which everyone looks at and checksout parts of it to work on which would lock other for working on it. Once you are finished your code can be merged onto this copy. Subversion is an example

Distributed version control system ,there may still be a version that everyone works against but each person has a full copy of the source code that can be worked on locally. Git Is an example.

1. **Web-Based DevOps Lifecycle Tools**: What are web-based DevOps lifecycle tools, and why are they important in a development environment? Provide an example of such a tool.

Github is a web based DevOps life cycle tool used for continuations intermigration and continuations deployment it is helpful because the development and feed back is quicker.

1. **DevOps Automation Tools**: Explain the role of DevOps automation tools in development environments. Provide an example of an automation tool you would typically use.

DevOps automeation tools do all the repetive task so they are so you can easly do CI/CD. GitHub Actions is an example

1. **Branching, Pull and Push Commands**: Briefly describe the following Git commands: branching, pull, and push. How do these commands facilitate collaboration in local and remote environments?

Branching - when working loaclly branch the code when working on a new feature and merge it back to main once feature is complete. It helps in organising you code.

Pull – fetches and merges from a remote repository your repository

Push – sends commits from your repository to the remote repository

1. **Key Policies and Procedures**: What are some key policies, procedures, and documentation required to use version control systems in development environments?

Things that should be covered in policies, procedures and documents are guidelines on commit messages, branch naming conversions and how to merge.

1. **Repositories and Working Copies**: Explain the concepts of repositories and working copies in version control systems. How do they contribute to the overall development workflow?

Repositories are the centerlise code that everyone Is pulls from and sends code too. The working copies are what the user is using and editing. Once completed the something sends to the repositoriy for everyone to use.

1. **Merging and Merge Tools**: What does the process of merging involve in version control systems? Can you name and describe a commonly used merge tool?

Merging involves intergreating changes from two branches.This is done by first being in the you want to get the changes. Then git merge branch you want to

1. **Working and Indiscriminate Commits**: What are working and indiscriminate commits? How can indiscriminate commits affect a project?

Indiscriminate commits is where you commit when ever you make a change, not documented or tested. Typo here ,formatting there, committing without a substantial reason. It put a lot of noise into the repo so using the log is harder.

1. **Resolving Conflict and Backout Changes**: Describe the process of resolving conflicts in version control systems. How can you back out changes if necessary?

To resolve conflicts git will open a text editor ,default is vi my is set to vscode, the conflicts will be highlighted. Then you choose the one, either what is there or the incoming. To back out there is a revet command or you can checkout the commit before the one you mangled and go from there.

1. **Creating Repositories and Branch Workflows**: Briefly explain the principles and techniques of creating repositories and branch workflows in a version control system.

To create a repository navigate to a folder in terminal and use the git init command. When I want to do a task, like adding a feature or fixing a bug, I will create a branch and name it appropriately. Do the task and merge it back into the main once it done.

Git init ,git switch -c <name> ,git merge

1. **Version Control Best Practices**: What are some industry-standard best practices when using version control systems?

Committing needs to be related to each other. Commit messages need to be detailed but concise. Branches need to be clean, focused and well maintained. Commit often, how often depends but a little more is usually better than less.

1. **Distributed Version Control Best Practices**: What are some industry-standard best practices specifically for using distributed version control systems?

Commit frequently, Use local tor complex merge and rebase tasks, utilise pull required for code review.

1. **Workflow Processes**: What workflow processes are applicable when using version control systems? How do they contribute to efficient development?

Organise work by features and fixes, write clear and consistent names for commits and branching.They contribute to efficient development by when you return to a project a lot less mental energy is spent understanding what is going on.

1. **Integration of DevOps Tools**: How can version control systems integrate with other DevOps tools? Provide an example of such integration.

There are issue trackers, CI/CD tools and code review tools are examples of such integration. An example of one of these is GitHub Actions which is a integrates GitHub for CI/CD

1. **Roles and Responsibilities**: What are the roles and responsibilities of a developer when using version control systems in a team setting?

There are two roles the contributor and maintainer, they basically do the best practices. The contributor maintains there local repositories, use clear and consistent naming for branches and commits, make sure pull requests are well-tested and documented. Maintainer need to look after the repositories that individual contributor work against, They vet pull request and only pull if up to standard. Maintainer also have all the responsibilities of a contributor.

1. **Version Control Systems Impact**: How do version control systems impact the speed and quality of software development?

Version control system make development of software faster and better code quality. This is because a lot of people can be working against the same code base without the risk of breaking it, faster. Build on what you had before and iterate the code without risk of breaking it.

1. **Version Control Systems Security**: How do version control systems contribute to the security of a software development project?

Version Control systems like GitHub have access control, tt also track changes, these two stop unauthorised changes and if they are made can revert because the changes are tracked. Version control make development more agile so when a vulnerability found it can be fixed quickly.

1. **Version Control Systems and Testing**: How do version control systems support testing and debugging processes in software development?

Version control system help testing and debugging by tracking changes, so it a change is made that is bad it can be reverted. Enable parallel testing of branches, which speeds up debugging. Supports CI/CD practices ,this reduces the feed back loop which helps find and fix bugs quicker.

1. **Continuous Integration and Continuous Deployment**: How does version control relate to the practices of Continuous Integration and Continuous Deployment?

Version control is needed to because to makes it posable to have many developers work on a project at the same time, Then pull there changes into the project often and send to deployment.

1. **Version Control Systems and Agile Methodologies**: How do version control systems support Agile methodologies in software development?

Version control supports agile methodologies by doing incremental development, lots of integrations which makes the feed back loop shorter and gives the ability to respond to changes quickly