**PRINCIPLE OF DEVOPS**

**LAB-04**

**NAME**: KRITTIKA V

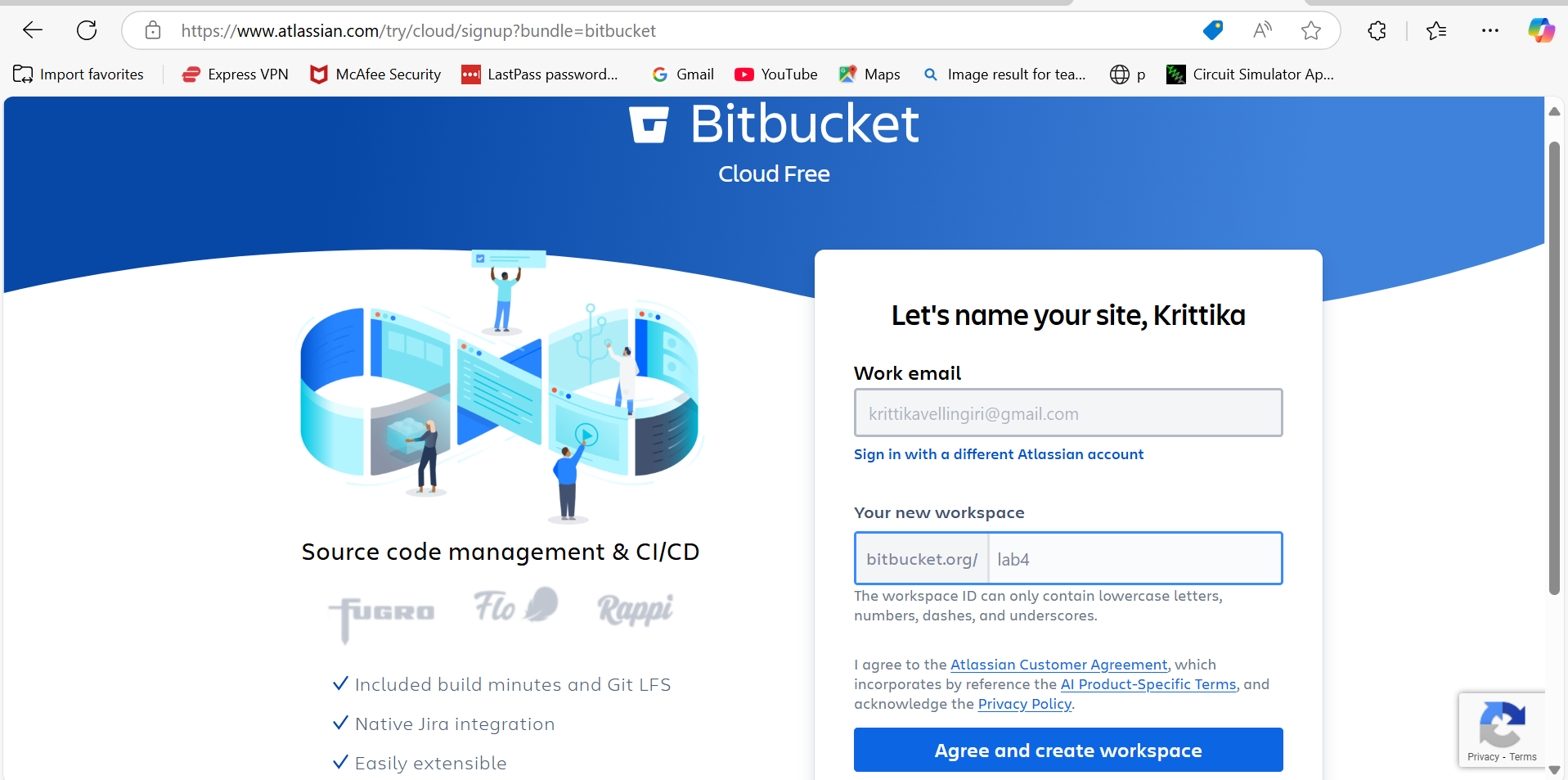
**ROLL NO:** 22BCS053

**Implement BitBucket Operations using Git.**

**Experiment Steps:**

**Step 1: Creating a Repository**

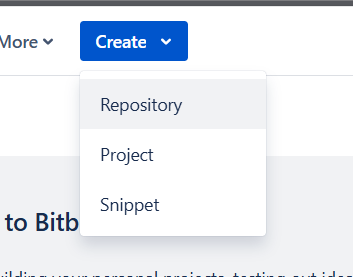
* Sign in to your Bitbucket account.



A screenshot of a computer

Description automatically generated

* Click the "Create" button to create a new repository.



* Choose a repository name, visibility (public or private), and other settings.
* Click "Create repository."

A screenshot of a computer

Description automatically generated

**Step 2: Cloning a Repository**

* Open your terminal or command prompt.
* Navigate to the directory where you want to clone the repository.
* Copy the repository URL from Bitbucket.
* Run the following command:

***git clone <repository\_url>***

* Replace <repository\_url> with the URL you copied from Bitbucket.
* This will clone the repository to your local machine.

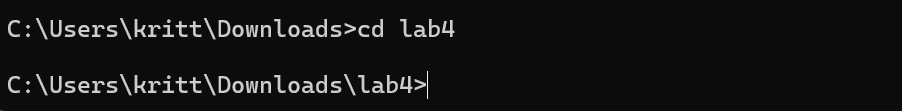
A screen shot of a computer

Description automatically generated

**Step 3: Making Changes and Creating a Branch**

* Navigate into the cloned repository:

***cd <repository\_name>***



* Create a new text file named "example.txt" using a text editor.
* Add some content to the "example.txt" file.
* Save the file and return to the command line.

A screenshot of a computer

Description automatically generated

* Check the status of the repository:

***git status***

A screen shot of a computer

Description automatically generated

* Stage the changes for commit:

***git add example.txt***

* Commit the changes with a descriptive message:

***git commit -m "Add content to example.txt"***

A computer screen with white text

Description automatically generated

* Create a new branch named "feature":

***git branch feature***

* Switch to the "feature" branch:
* ***git checkout feature***

A black screen with white text

Description automatically generated

**Step 4: Pushing Changes to Bitbucket**

* Add Repository URL in a variable

***git remote add origin  <repository\_url>***

* Replace <repository\_url> with the URL you copied from Bitbucket.



* Push the "feature" branch to Bitbucket:

***git push origin feature***

* Check your Bitbucket repository to confirm that the new branch "feature" is available.

A screenshot of a computer program

Description automatically generated

**Step 5: Collaborating through Pull Requests**

1. Create a pull request on Bitbucket:

* Go to the repository on Bitbucket.

A screenshot of a computer

Description automatically generated

* Click on "Create pull request."
* Choose the source branch ("feature") and the target branch ("main" or "master").
* Review the changes and click "Create pull request."

1. Review and merge the pull request:
   * Add a title and description for the pull request.
   * Assign reviewers if needed.

A screenshot of a computer

Description automatically generated

* + Once the pull request is approved, merge it into the target branch.

A screenshot of a computer

Description automatically generated

click merge

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Step 6: Syncing Changes**

* After the pull request is merged, update your local repository:

***git checkout main***

A black screen with white text

Description automatically generated

***git pull origin main***

A computer screen with white text

Description automatically generated

**Conclusion:**

This experiment provided you with practical experience in performing Bitbucket operations using Git commands. You learned how to create repositories, clone them to your local machine, make changes, create branches, push changes to Bitbucket, collaborate through pull requests, and synchronise changes with remote repositories. These skills are essential for effective collaboration and version control in software development projects using Bitbucket and Git.

**Questions/Exercises:**

**Q.1 What is Bitbucket, and how does it fit into the DevOps landscape?**

Bitbucket is a Git-based repository management tool developed by Atlassian that enables teams to plan, collaborate, and deploy code efficiently. It supports source code hosting, version control, and CI/CD pipelines. In the DevOps landscape, Bitbucket plays a crucial role by integrating with other tools to streamline the software development lifecycle. It allows for automated builds, testing, deployment, and seamless collaboration among developers, contributing to faster and more reliable software delivery.

**Q.2 Explain the concept of branching in Bitbucket and its significance in collaborative development.**

Branching in Bitbucket refers to creating isolated copies of the codebase for development, bug fixes, or experimentation without affecting the main code. It allows teams to work on multiple features or issues simultaneously. Common branch types include:

* **Feature branches**: For new features.
* **Bugfix branches**: For resolving bugs.
* **Hotfix branches**: For urgent fixes.

Significance in collaborative development:

* **Parallel Work**: Developers can work on different tasks without conflicts.
* **Code Isolation**: Reduces risks by isolating changes until they are ready for merging.
* **Structured Workflow**: Aligns with Gitflow and other branching models to streamline development processes.

**Q.3 What are pull requests in Bitbucket, and how do they facilitate code review and collaboration?**

A **pull request** in Bitbucket is a formal mechanism for proposing, reviewing, and discussing changes before merging them into the target branch. Developers submit pull requests to notify team members of the changes made in a branch.

Pull requests facilitate:

* **Code Review**: Peers and team leads can review changes for quality, consistency, and security.
* **Collaboration**: Allows for discussion and suggestions via inline comments.
* **Accountability**: Maintains a record of changes, discussions, and decisions.
* **Pre-Merge Validation**: Enables integration with CI/CD pipelines to run tests and checks before merging.

**Q.4 How can you integrate code quality analysis and security scanning tools into Bitbucket's CI/CD pipelines?**

Bitbucket integrates with code quality and security scanning tools through its CI/CD pipelines. Steps to integrate these tools include:

1. **Add Scanning Tools**: Use tools like **SonarQube**, **CodeQL**, or **Snyk** in the pipeline configuration file (bitbucket-pipelines.yml).
2. **Pipeline Configuration**: Define steps to run static code analysis, linting, and security scans.
3. **Artifacts and Reports**: Configure the pipeline to generate reports and artifacts for analysis.
4. **Fail Conditions**: Set thresholds for code quality or security issues to fail the pipeline if violated.
5. **Notifications**: Enable alerts or integrations with Jira/Slack for detected issues.

**Q.5 What are merge strategies in Bitbucket, and how do they affect the merging process during pull requests?**

Merge strategies determine how changes from a branch are incorporated into the target branch. Bitbucket supports three main strategies:

1. **Merge Commit**:
   * Creates a new commit combining the source and target branches.
   * **Use case**: Retains full history and context of the merge.
2. **Squash Merge**:
   * Combines all commits in the source branch into a single commit before merging.
   * **Use case**: Produces a cleaner history, ideal for small, incremental changes.
3. **Fast-Forward**:
   * Moves the target branch pointer forward if there are no diverging commits.
   * **Use case**: Useful when linear history is preferred.

**Impact on Pull Requests**:

* **History Management**: The chosen strategy affects the clarity and readability of the Git history.
* **Conflict Resolution**: Influences how conflicts are detected and resolved.
* **Team Preferences**: Aligning merge strategies with team practices ensures consistency.