1. Outline the steps involved in creating a new branch and pushing changes in the Forking

workflow.

=>

1) fork the repository :

Go to original repository and click on fork button .this create a copy of repository under your account.

2)

Clone the forked repository

Git clone “forked repo link”

3) create a new branch

4)make changes

5) stage and commit changes:-

Git add.

Git commit -m ”add new features”

6) push changes to your fork

Git push origin feature-name

7) create a pull request

Go to forked repo in git hub and switch to branch you just pushed.

Click on “new pull request”

Select the base repo and branch

Review the changes

Click on “create pull request” to submit the pull request

8) now your pulled request is approved and merged into repository by reviwer

What are feature branches in Git, and why are they useful for development teams?

Feature branches in Git are separate branches created to work on specific features or enhancements in a software project. Each feature branch typically corresponds to a single new feature, bug fix, or improvement that the development team is working on. Feature branches are created from the main or development branch and are merged back into the main codebase once the feature is completed and tested.

Here's why feature branches are useful for development teams:

1. \*\*Isolation of Changes:\*\* Feature branches allow developers to work on different features or fixes in isolation from each other. This isolation prevents conflicts between changes made by different team members and provides a clean environment for developing and testing new features.

2. \*\*Parallel Development:\*\* Feature branches enable parallel development of multiple features simultaneously. Different team members can work on separate features without blocking each other's progress, leading to increased productivity and faster delivery of new features.

3. \*\*Easy Collaboration:\*\* Feature branches facilitate collaboration among team members by providing a centralized platform for sharing code changes. Developers can easily collaborate on feature branches, review each other's code, and provide feedback before merging the changes into the main codebase.

4. \*\*Improved Code Quality:\*\* Feature branches encourage developers to focus on implementing and testing specific features comprehensively. By working in isolation, developers can ensure that their changes meet the requirements and quality standards before merging them into the main codebase, leading to higher overall code quality.

5. \*\*Clear Versioning and History:\*\* Feature branches provide a clear and organized versioning structure for the project. Each feature branch represents a distinct set of changes, making it easier to track the development progress and understand the history of the codebase. This clarity enhances transparency and facilitates troubleshooting and debugging.

6. \*\*Risk Mitigation:\*\* Feature branches minimize the risk associated with introducing new features or changes into the main codebase. By developing and testing features in isolation, developers can identify and address issues early in the development process, reducing the likelihood of introducing bugs or regressions into the production environment.

Overall, feature branches in Git offer development teams a structured and efficient approach to managing and implementing new features, bug fixes, and enhancements in a collaborative and organized manner. They promote parallel development, code quality, collaboration, and risk mitigation, ultimately contributing to the success of the software project.

When is the release branching strategy typically employed, and what are its benefits?

The release branching strategy is typically employed when a software project is preparing for a new release. It involves creating a separate branch from the main development branch (e.g., master or main) specifically for the release process. This release branch serves as a stable environment for finalizing and testing the release candidate before it is deployed to production.

Here are the benefits of the release branching strategy:

1. \*\*Isolation of Release Preparation:\*\* By creating a separate release branch, development teams can isolate the release preparation process from ongoing development work. This ensures that the main development branch remains stable and unaffected by last-minute changes or fixes intended for the release.

2. \*\*Focused Bug Fixes:\*\* The release branch allows for focused bug fixes and enhancements targeted specifically for the upcoming release. Developers can address critical issues and make final adjustments without disrupting other ongoing development efforts.

3. \*\*Testing and Validation:\*\* The release branch provides a controlled environment for rigorous testing and validation of the release candidate. Quality assurance teams can thoroughly test the functionality, performance, and stability of the software before it is deployed to production, reducing the risk of introducing bugs or regressions.

4. \*\*Clear Versioning:\*\* The release branch serves as a clear marker for the version of the software being prepared for release. It allows developers and stakeholders to easily identify and track the progress of the release candidate, facilitating communication and coordination among team members.

5. \*\*Rollback and Hotfixes:\*\* In the event of critical issues or unexpected problems discovered during testing, the release branch enables quick rollback to a previous stable state. It also provides a controlled environment for applying hotfixes or emergency patches to address urgent issues without affecting the main development branch.

6. \*\*Release Documentation:\*\* The release branch can be used to finalize release documentation, such as release notes, user guides, and installation instructions. This ensures that all necessary documentation is up-to-date and ready for distribution alongside the release.

Overall, the release branching strategy helps streamline the release process, improve quality control, and minimize disruptions to ongoing development work, resulting in smoother and more reliable software releases.