1. Design and implement processes and communications
└ 1.3 Configure collaboration and communication
└ 1.3.3 Automate documentation from Git history

---

1. What are the benefits of automating documentation from Git history?
2. Which tools can generate documentation from Git commit history?
3. How do you use conventional commits to structure auto-generated documentation?
4. What is the typical process to generate changelogs automatically from Git history?
5. How do you configure a CI pipeline to automate changelog generation?
6. What is the role of commit message standards in documentation automation?
7. How can release notes be generated automatically from pull requests or commits?
8. Which GitHub Actions are commonly used for automated documentation?
9. How can you integrate auto-generated documentation with project wikis or Markdown files?
10. What are best practices for maintaining accuracy in automated documentation?

---

1. What are the benefits of automating documentation from Git history?

- Ensures documentation is always up to date with code changes
- Reduces manual effort and human error
- Improves traceability and transparency for releases
- Provides a reliable audit trail for changes

---

2. Which tools can generate documentation from Git commit history?

- Conventional Changelog
- *GitHub Actions* (e.g., release-drafter, changelog-generator)
- GitVersion
- *GitLab* Release Notes
- *Azure DevOps* Release Notes Extension

---

3. How do you use conventional commits to structure auto-generated documentation?

- Enforce commit messages to follow the Conventional Commits format (e.g., feat:, fix:, docs:)
- Tools parse commit messages to categorize and format changelogs automatically
- Enables automation tools to extract features, fixes, and breaking changes for documentation

---

4. What is the typical process to generate changelogs automatically from Git history?

- Enforce commit message standards (e.g., Conventional Commits)
- Use a changelog tool (e.g., conventional-changelog) in CI or manually
- Tool scans Git history since last tag/release
- Generates or updates changelog file with categorized changes

---

5. How do you configure a CI pipeline to automate changelog generation?

- Add a changelog generation step (using tools like conventional-changelog, release-drafter) in your CI workflow (*GitHub Actions*, *Azure Pipelines*, etc.)
- Trigger the step on events like merges to main or new release tags
- Output or commit the changelog file as an artifact or into the repo

6. What is the role of commit message standards in documentation automation?

- Commit message standards enable automated tools to identify the type and scope of changes
- Structured messages allow accurate and meaningful release notes and changelogs
- Inconsistent messages lead to incomplete or misleading documentation

---

7. How can release notes be generated automatically from pull requests or commits?

- Use tools (e.g., release-drafter, *GitHub* release notes) that scan merged pull requests or commits
- Configure templates to map PR titles or commit messages to release note sections
- Trigger note generation on release/tag creation events in CI

---

8. Which GitHub Actions are commonly used for automated documentation?

- release-drafter/release-drafter
- conventional-changelog/standard-version

---

9. How can you integrate auto-generated documentation with project wikis or Markdown files?

- Output changelog or release notes to a *Markdown* file (e.g., CHANGELOG.md)
- Commit the file to the repo so it appears in project docs
- Use CI to sync the content into wiki pages or documentation sites

---

10. What are best practices for maintaining accuracy in automated documentation?

- Enforce strict commit and PR message conventions
- Review generated docs as part of the release process
- Automate generation as part of CI for consistency
- Regularly update automation tooling and templates