Docs-as-code

Docs-as-code means “documentation as code.” It is an approach where documentation teams create and manage documentation using the same tools and workflows as engineers.

This approach aims to make documentation follow the same process as software development. It is used to document, collaborate, and maintain software documentation.

Here are the key concepts and practices involved:

1. **Version Control**: Documentation is stored in version control systems like Git, enabling collaboration, versioning, and traceability.
2. **Markup Languages**: Documentation is written in lightweight markup languages such as Markdown, reStructuredText, or AsciiDoc, making it easy to read and edit in plain text.
3. **Continuous Integration/Continuous Deployment (CI/CD)**: Automated pipelines are used to build, test, and deploy documentation, ensuring that it is always up-to-date and free of errors.
4. **Collaboration**: Using platforms like GitHub or GitLab, multiple contributors can work on the documentation simultaneously, with changes reviewed and merged through pull requests or merge requests.
5. **Automation**: Tasks such as spell checking, link checking, and formatting can be automated to maintain high-quality documentation.
6. **Single Source of Truth**: Documentation is integrated with the codebase, ensuring that it evolves alongside the software and reduces discrepancies between the code and its documentation.
7. **Documentation as Part of the Development Process**: Documentation is treated as an integral part of the development process, with developers writing and maintaining docs as they code.
8. **Infrastructure as Code (IaC)**: Similar principles apply to infrastructure documentation, where configuration files and infrastructure definitions are documented and versioned.
9. **Static Site Generators**: Tools like Jekyll, Hugo, or Sphinx are used to generate static websites from the documentation files, providing a professional and navigable output.
10. **Code Reviews**: Documentation changes undergo peer reviews to ensure accuracy, clarity, and completeness.

**Benefits**

* **Consistency**: Documentation is kept in sync with the codebase.
* **Collaboration**: Easier for teams to work together and contribute.
* **Automation**: Reduces manual errors and saves time.
* **Versioning**: Historical versions of documentation are preserved and accessible.

**Popular Tools**

* **Git**: For version control.
* **GitHub/GitLab**: For hosting repositories and collaboration.
* **Markdown/reStructuredText/AsciiDoc**: For writing documentation.
* **CI/CD Pipelines (e.g., GitHub Actions, GitLab CI)**: For automation.
* **Static Site Generators (e.g., Jekyll, Hugo, Sphinx)**: For publishing documentation.

Additional Information:

[Docs as Code](https://www.writethedocs.org/guide/docs-as-code/)

[Introduction To Docs As Code](https://www.everythingtechnicalwriting.com/dosc-as-code/)

[11 Top Code Documentation Tools](https://www.archbee.com/blog/code-documentation-tools)