**Software Configuration Management**

# **Part 1: Configuration-As-Code in DevOps**

***Define Configuration-As-Code in DevOps***

There are different definitions of Configuration-as-Code and Software Configuration Management (SCM). The difference lies in the area where the code is relevant. My favorite definition of SCM is:

Software configuration management (SCM) comprises factors such as configuration identification, configuration control, status accounting, review, build management, process management, and team work (Dart 1992)[1]. SCM practices taken as a whole define how an organization builds and releases products, and identifies and tracks changes.

Berczuk and Appleton [2]

We can see how that definition changes depending on what we refer to. For example, Smith [3] defines configuration management as *the process used by your organization to establish and maintain conformity across your various applications*.

Version Control Systems (VCS) allow you to record file changes over time. Many types of VCS tools exist nowadays. Unfortunately, not everything is ready to be configured as code, but over time, tooling appears with efforts from both Open-Source and Companies. One prominent example is infrastructure. During the 2000s, you could purchase server time via the web and connect to these remote computers. Deploying changes to a remote computer took time and effort. In 2007, Amazon saw an opportunity and created Amazon Web Services. Amazon provided companies with access to develop services in the cloud via their own HTTP APIs. Eventually, they made a suite of tools to run over these processes. The ability to create code to automate Cloud Infrastructure creation became known as Infrastructure-as-Code (IaC).

My favorite definition of "as-code" appears in The DevOps Handbook by Kim, Humble, Debois, and Willis [4]:

* All application code and dependencies.
* Any script to create data-related information.
* All environment creation tools and artifacts (e.g., IaC)
* Container creation files.
* Tests of all kinds, manual or automated.
* Scripts to package code, deploy, migrate, or provision.
* All artifacts (documentation, too).
* All cloud configuration files (more IaC).
* Anything related to infrastructure (like firewall configurations).

## ***Importance of Configuration-As-Code in DevOps***

Many studies have shown that "all top companies use VCS" [4] and that "Well-architected principles include IaC." [5] [6] Some of them are mentioned in the DevOps Handbook as well. However, the most important reason is not related to statistics or studies of the best players in the industry. The most important reason is related to *Downtime*. In the book "Think Like a CTO," Alan Williamson [7] establishes a simple formula for a Disaster Recovery strategy:

* What is your company's downtime tolerance?
* During recovery, do you need full or partial service?
* Will you be failing forward or bouncing back?

Failing forward means creating a new primary service to replace your failing service.

During a Disaster Event, emotions are high, options are low, and pressure is astronomical. Coming up with these answers is not easy in any workplace, as most CEOs will want to demand zero Downtime. Zero Downtime is possible but, most of the time, hugely prohibitive for any project. It's essential to figure out budgets, leading us to "How fast do we need to be back online?" Every second means money, and the easiest way to bring everything back up is to ensure everything is in Code. I have taken deployments from days of manual changes to one-click with predictable rollouts. The more things we configure as Code in the VCS, the faster we will recover from a disaster event, and the more we minimize (and sometimes nullify) the losses.

# **Part 2: Support Tools for SCM Functions**

## ***Git Software Configuration Management Tool***

The Git software configuration management tool supports many features that enable users to manage and track changes to their codebase effectively. The following is a summary of the support tools provided by the tool for each of the SCM functions (identification, control, auditing, status accounting):

* Identification: Git provides a way to identify file modifications in a project. The tracking helps users track changes made to the codebase over time. The tool uses a hash function to create a unique identifier for each file, and it uses the hash to compare the current state of the file with its previous version.
* Control: Git allows users to control who can modify files and how those modifications are applied. The control ensures that the codebase remains consistent and reliable. The tool also uses permissions and branching to allow users to collaborate effectively.
* Auditing: Git allows users to audit changes made to the codebase, which helps to identify any errors or issues found during development. The tool provides a way to track changes made to files over time, including who made the changes and when.
* Status accounting: Git allows users to track the status of files in a project. The status of each file could be: added, deleted, or modified. And the status helps users understand the codebase's current state and plan for future changes. The tool uses a tree-like structure to represent the hierarchy of files and directories in the codebase, which makes it easy to track changes made to individual files or groups of files.

The Git software configuration management tool is widely used in industry and academia due to its ease of use, flexibility, and scalability. Many developers use it to manage their codebase effectively and collaborate with other team members. [8] The tool provides support tools that enable users to manage and track changes made to their codebase effectively, including identification, control, auditing, and status accounting.

As for the commands that help us achieve these goals they are done via the following:

|  |  |
| --- | --- |
| Command | Description |
| git commit | Identifies the files that have been modified in a project. This helps users to track changes made to the codebase over time. |
| git branch | Controls who can modify files and how those modifications are applied. This ensures that the codebase remains consistent and reliable. |
| git log | Audits changes made to the codebase, which helps to identify any errors or issues that may have been introduced during development. |
| git status | Tracks the status of files in a project, including whether they have been modified, added, or deleted. This helps users to understand the current state of the codebase and to plan for future changes. |

# **References**

|  |  |
| --- | --- |
| [1] | S. A. Dart, "The past, present, and future of configuration management," in *IFIP Congress (1)*, 1992. |
| [2] | S. P. Berczuk and B. Appleton, Software configuration management patterns, 10. print. ed., Boston: Addison-Wesley, 2008. |
| [3] | J. Smith, Operations Anti-Patterns, DevOps Solutions, New York: Manning Publications Co. LLC, 2020. |
| [4] | G. Kim, The DevOps handbook, Second edition ed., J. Humble, P. Debois, J. Willis, N. Forsgren and J. Allspaw, Eds., Portland, OR: IT Revolution Press, LLC, 2021. |
| [5] | *OPS05-BP03 Use configuration management systems - Operational Excellence Pillar — docs.aws.amazon.com,* 2024. |
| [6] | PageWriter-MSFT, *Operational Excellence cloud design principles - Microsoft Azure Well-Architected Framework — learn.microsoft.com.* |
| [7] | A. Williamson, Think Like a CTO, New York: Manning, 2023. |
| [8] | R. Umali, Learn Git in a Month of Lunches, New York: Manning Publications Co. LLC, 2015. |
| [9] | C. Davis, Agile Metrics in Action, New York: Manning Publications Co. LLC, 2015. |
| [10] | S. Chacon, Pro Git, 2nd ed. ed., B. Straub, Ed., Berkeley, CA: Apress L. P., 2014. |
| [11] | J. Loeliger and M. McCullough, Version Control with Git Powerful Tools and Techniques for Collaborative Software Development, O'Reilly Media, Incorporated, 2012, p. 456. |