Introduction

Completed

- 1 minute

You can think of source control as an essential everyday practice. Versioning is a standard part of the developer's routine and, if used correctly, can save organizations enormous costs and resources.

The source control is a common-sense aspect of programming. It is essential from time to time to look at why we do what we do and how versioning impacts the entire value stream at an organization.

This module introduces you to the basics of source control, exploring benefits and best practices.

Learning objectives

After completing this module, students and professionals can:

- Understand source control.
- Apply best practices for source control.
- Describe the benefits of using source control.

Prerequisites

- Understanding of what DevOps is and its concepts.
- Familiarity with version control principles is helpful but is not necessary.
- Beneficial to have experience in an organization that delivers software.

Continue

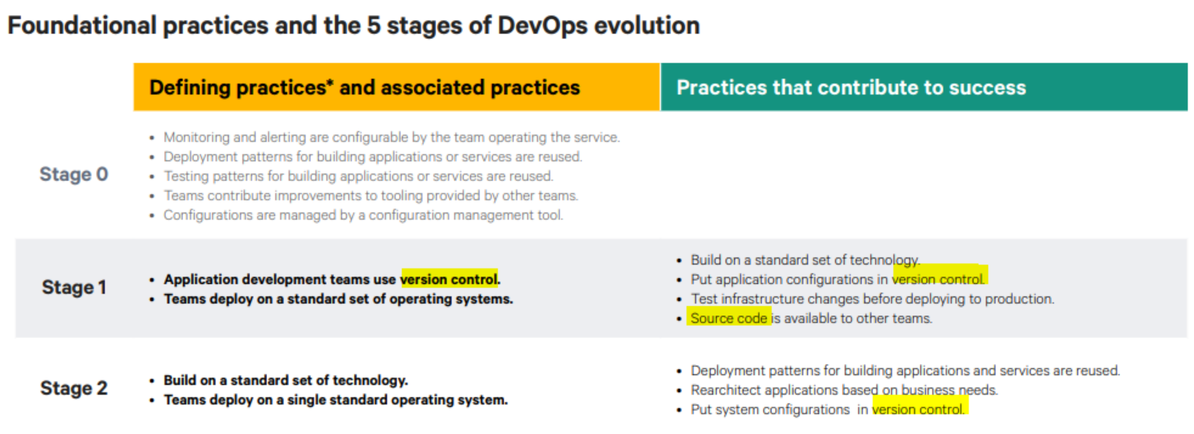
Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Explore DevOps foundational practices

Completed

- 3 minutes

The State of DevOps Report 2021 highlights version control in almost all stages of DevOps evolution.

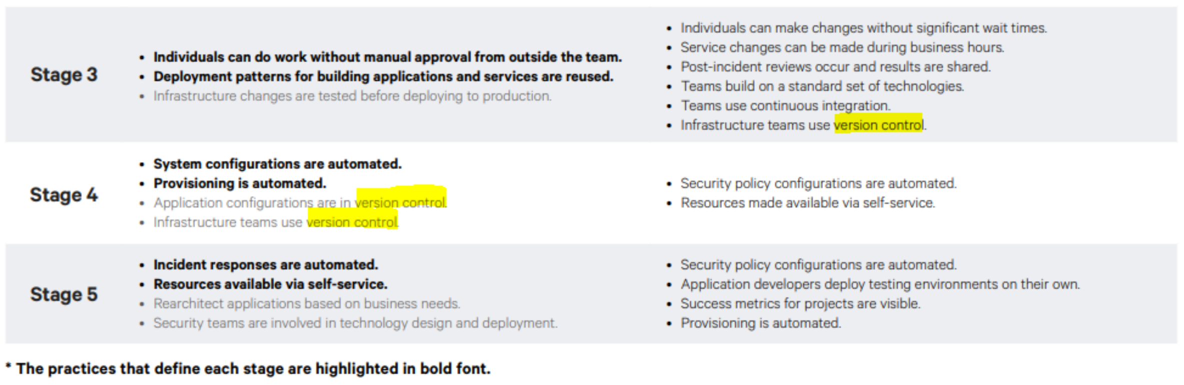


Foundational practices and the 5 stages of DevOps evolution

	Defining practices* and associated practices	Practices that contribute to success
Stage 0	• Monitoring and alerting are configurable by the team operating the service. • Deployment patterns for building applications or services are reused. • Testing patterns for building applications or services are reused. • Teams contribute improvements to tooling provided by other teams. • Configurations are managed by a configuration management tool.	
Stage 1	• **Application development teams use version control.** • **Teams deploy on a standard set of operating systems.**	• Build on a standard set of technology. • Put application configurations in version control. • Test infrastructure changes before deploying to production. • Source code is available to other teams.
Stage 2	• **Build on a standard set of technology.** • **Teams deploy on a single standard operating system.**	• Deployment patterns for building applications and services are reused. • Rearchitect applications based on business needs. • Put system configurations in version control.

It is helpful for non-developers in an organization to understand the fundamentals of the discipline.

Also, it is so deeply rooted in the daily life of software engineers. Essential if those individuals decide which version control tools and platforms to use.



Stage 3	• **Individuals can do work without manual approval from outside the team.** • **Deployment patterns for building applications and services are reused.** • Infrastructure changes are tested before deploying to production.	• Individuals can make changes without significant wait times. • Service changes can be made during business hours. • Post-incident reviews occur and results are shared. • Teams build on a standard set of technologies. • Teams use continuous integration. • Infrastructure teams use version control.
Stage 4	• **System configurations are automated.** • **Provisioning is automated.** • Application configurations are in version control. • Infrastructure teams use version control.	• Security policy configurations are automated. • Resources made available via self-service.
Stage 5	• **Incident responses are automated.** • **Resources available via self-service.** • Rearchitect applications based on business needs. • Security teams are involved in technology design and deployment.	• Security policy configurations are automated. • Application developers deploy testing environments on their own. • Success metrics for projects are visible. • Provisioning is automated.

*** The practices that define each stage are highlighted in bold font.**

Version control is essential for all software development projects and is vital at large businesses and enterprises.

Enterprises have many stakeholders. For example:

- Distributed teams.
- Strict processes and workflows.
- Siloed organizations.
- Hierarchical organizations.

All those characteristics represent coordination and integration challenges when it comes to merging and deploying code.

Companies within highly regulated industries need a practical way to ensure that all standards are met appropriately and mitigate risk—for example, banking and healthcare.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

What is source control?

Completed

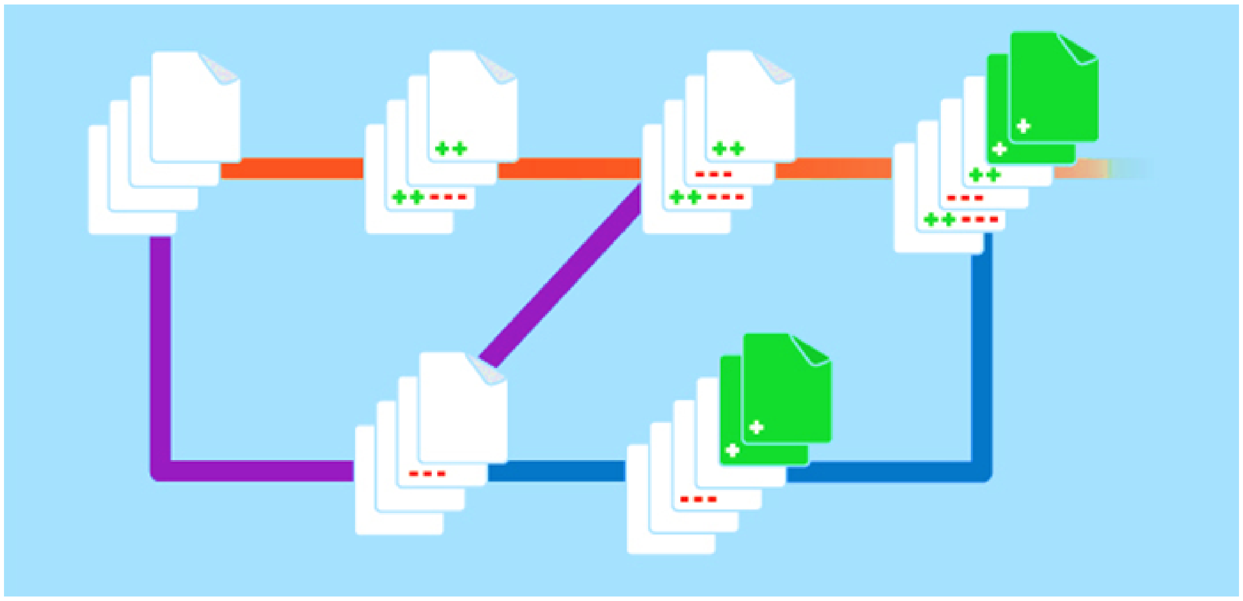
- 2 minutes

A Source control system (or version control system) allows developers to collaborate on code and track changes. Use version control to save your work and coordinate code changes across your team. Source control is an essential tool for multi-developer projects.

The version control system saves a snapshot of your files (history) so that you can review and even roll back to any version of your code with ease. Also, it helps to resolve conflicts when merging contributions from multiple sources.

For most software teams, the source code is a repository of invaluable knowledge and understanding about the problem domain that the developers have collected and refined through careful effort.

Source control protects source code from catastrophe and the casual degradation of human error and unintended consequences.



Without version control, you're tempted to keep multiple copies of code on your computer. It could be dangerous. Easy to change or delete a file in the wrong code copy, potentially losing work.

Version control systems solve this problem by managing all versions of your code but presenting you with a single version at a time.

Tools and processes alone aren't enough to accomplish the above, such as adopting Agile, Continuous Integration, and DevOps. Believe it or not, all rely on a solid version control practice.

Version control is about keeping track of every change to software assets—tracking and managing the who, what, and when. Version control is the first step needed to assure quality at the source, ensure flow and pull value, and focus on the process. All of these create value not just for the software teams but ultimately for the customer.

Version control is a solution for managing and saving changes made to any manually created assets. If changes are made to the source code, you can go back in time and easily roll back to previous-working versions.

Version control tools will enable you to see who made changes, when, and what exactly was changed.

Version control also makes experimenting easy and, most importantly, makes collaboration possible. Without version control, collaborating over source code would be a painful operation.

There are several perspectives on version control.

- For developers, it's a daily enabler for work and collaboration to happen. It's part of the daily job, one of the most-used tools.
- For management, the critical value of version control is in:
  - IP security.
  - Risk management.
  - Time-to-market speed through Continuous Delivery, where version control is a fundamental enabler.

Continue

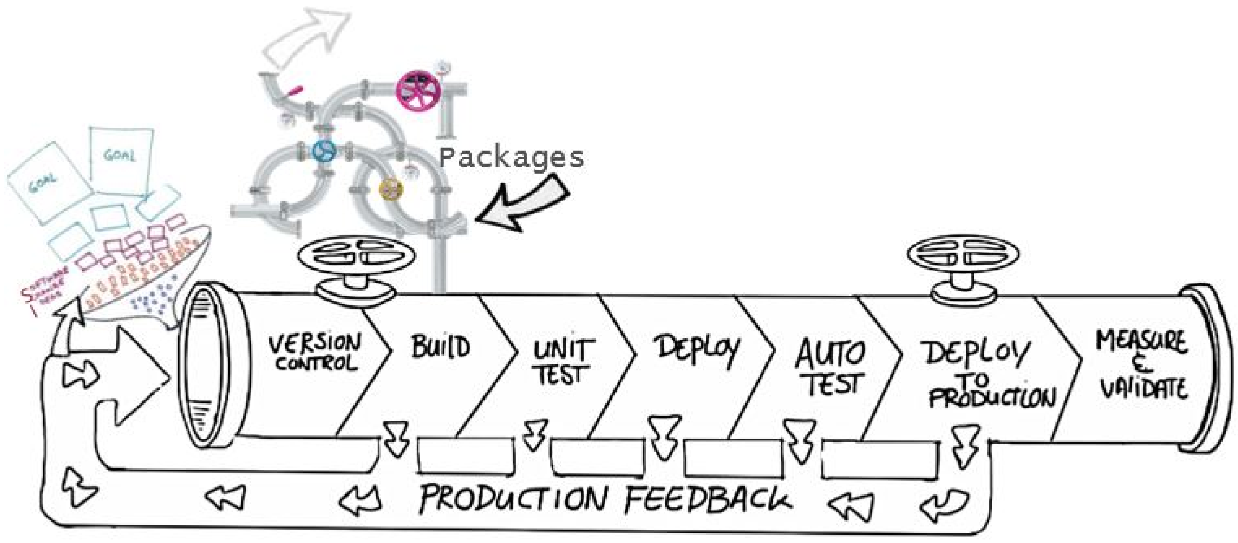
Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Explore benefits of source control

Completed

- 3 minutes

"Code does not exist unless it is committed into source control. Source control is the fundamental enabler of continuous delivery."



Whether writing code professionally or personally, you should always version your code using a source control management system. Some of the advantages of using source control are,

- **Create workflows** . Version control workflows prevent the chaos of everyone using their development process with different and incompatible tools. Version control systems provide process enforcement and permissions, so everyone stays on the same page.
- **Work with versions** . Every version has a description in the form of a comment. These descriptions help you follow changes in your code by

version instead of by individual file changes. Code stored in versions can be viewed and restored from version control at any time as needed. It makes it easy to base new work on any version of code.

- **Collaboration** . Version control synchronizes versions and makes sure that your changes do not conflict with other changes from your team. Your team relies on version control to help resolve and prevent conflicts, even when people make changes simultaneously.
- **Maintains history of changes** . Version control keeps a record of changes as your team saves new versions of your code. This history can be reviewed to find out who, why, and when changes were made. The history gives you the confidence to experiment since you can roll back to a previous good version at any time. The history lets your base work from any code version, such as fixing a bug in an earlier release.
- **Automate tasks** . Version control automation features save your team time and generate consistent results. Automate testing, code analysis, and deployment when new versions are saved to version control.

Common software development values

- **Reusability** – why do the same thing twice? Reuse of code is a common practice and makes building on existing assets simpler.
- **Traceability** – Audits are not just for fun; in many industries, it is a legal matter. All activities must be traced, and managers can produce reports when needed. Traceability also makes debugging and identifying root cause easier. Additionally, it helps with feature reuse as developers can link requirements to implementation.
- **Manageability** – Can team leaders define and enforce workflows, review rules, create quality gates and enforce QA throughout the lifecycle?
- **Efficiency** – are we using the right resources for the job, minimizing time and effort? This one is self-explanatory.
- **Collaboration** – When teams work together, quality tends to improve. We catch one another's mistakes and can build on each other's strengths.
- **Learning** – Organizations benefit when they invest in employees learning and growing. It is important for onboarding new team members, the lifelong learning of seasoned members, and the

opportunity for workers to contribute to the bottom line and the industry.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Explore best practices for source control

Completed

- 2 minutes

- **Make small changes** . In other words, commit early and commit often. Be careful not to commit any unfinished work that could break the build.
- **Do not commit personal files** . It could include application settings or SSH keys. Often personal files are committed accidentally but cause problems later when other team members work on the same code.
- **Update often and right before pushing to avoid merge conflicts** .
- **Verify your code change before pushing it to a repository** ; ensure it compiles and tests are passing.
- **Pay close attention to commit messages, as it will tell you why a change was made** . Consider committing messages as a mini form of documentation for the change.
- **Link code changes to work items** . It will concretely link what was created to why it was created—or modified by providing traceability across requirements and code changes.
- **No matter your background or preferences, be a team player and follow agreed conventions and workflows** . Consistency is essential and helps ensure quality, making it easier for team members to pick up where you left off, review your code, debug, and so on.

Using version control of some kind is necessary for any organization, and following the guidelines can help developers avoid needless time spent fixing errors and mistakes.

These practices also help organizations reap more significant benefits from having a good version control system.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Knowledge check

Completed

- 5 minutes

Choose the best response for each question. Then select **Check your answers** .

Check your knowledge

1.

Which of the following choices isn't a benefit of source control?

○

Manageability.

○

Efficiency.

○

Accountability.

2.

Which of the following choices isn't a source control best practice?

○

Make small changes.

○

Commit personal and secure files.

○

Link code changes to work items.

3.

Which of the following choices correctly describes one of the most valuable version control features?

○

Version control is a solution for managing and saving changes made to manually created assets. If you make changes to the source code, you can go back in time and easily roll back to previous-working versions.

○

Version control is a solution for automatically incrementing the version number for deployments.

○

Version control is a solution for managing and saving changes made to manually created assets. If you make changes to the source code, be careful, you can't go back in time and easily roll back to previous-working versions.

Check your answers

You must answer all questions before checking your work.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Summary

Completed

- 1 minute

This module explored the basics of source control and how to work with it daily to benefit team collaboration and code maintenance.

You learned how to describe the benefits and usage of:

- Understand source control.
- Apply best practices for source control.
- Describe the benefits of using source control.

Learn more

- Understand source control - Azure DevOps .
- Using source control in your codespace - GitHub Docs .

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .