Course: Software Configuration Management

Week 11: Software Release Management and Delivery – Part 1

Lecturer: Yimer Amedie (MSc.)
Addis Ababa Science and Technology University, Ethiopia

November, 2025

Contents

Figure 1: Concepts of SCM
(Source: OpenAI, 2025)

- Introduction to software release management.
- Software build types and automation.
- Version control and tagging.
- Software delivery and deployment methods.
- Release planning, approval, and risk management.
- Summary

Learning Outcomes

After completing this lesson, you will be able to:

- Define software release management concepts.
- Identify build types and their purposes.
- Apply version control and tagging in releases.
- Explain CI/CD and release pipeline processes.
- Implement best practices for reliable software delivery.

Introduction

Figure 2: Software Dev't process (Source: HRF07 (2025, August 20). Plan, code, build, test, release, deploy, operate, monitor [icon]. Devops Icon Set Vector Concept Icon Stock Vector (Royalty Free) 2668094693 | Shutterstock)

- According to the (Leon, 2015)
  - A **release** refers to distributing software outside the development activity.
  - **Main goal:** make the software available to its end users.
  - Releases can target **internal users** or **external customers**.
  - Closely related to build management
    - A release is a production build.
  - Includes deployment and version tracking processes.

Key Activities in Release Management

- **Identification**, **packaging**, and **delivery** of software elements.
  - Includes executables, documentation, release notes, and configuration data.
- **Timing of release** is crucial.
  - Software **changes** continuously
  - Decisions depend on **problem severity**, **fault density**, and **market factors**.
- Packaging selects correct product items and variants for delivery.

Software Build – Definition and Purpose

- A software build is a compiled version of source code.
  - Converts code into executable form for testing or release.
  - Integral to the release management process.
  - Ensures consistent and reproducible software versions.
  - Managed through automated or manual build tools.

The process of converting source code into a standalone, executable program, and the resulting software artifact.

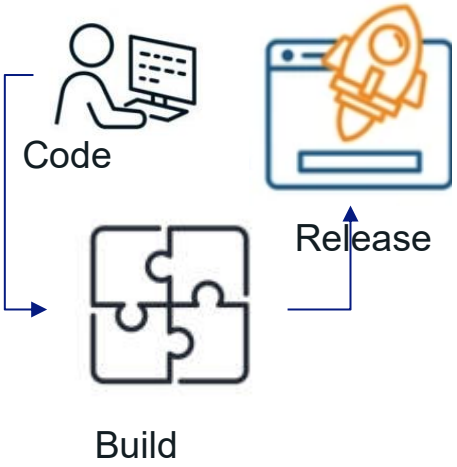
Involves:-
compiling code, running automated tests, linking libraries, and packaging the final product,
which can then be distributed to users

Software Build – Types

There are different types of build in software development (Kv, 2023)

Incremental Builds	Compiling only the parts of the code that have changed since the last build.
Continuous Builds	All files in a project are compiled and tested automatically at regular intervals.
Full Builds	Compiling all files in a project every time takes longer
Automated Builds	Entire process is automated Configured to run regularly at certain times or locations
Release Builds	performed before an application is released to production

Build Management and Release Linkage



- Build management ensures consistent outputs from source code.
- Release management depends on reliable builds.
- Build artifacts are stored in repositories.
- Version identifiers link builds to releases.
- Build automation enhances traceability and speed

Build Automation – Overview

- Build automation removes manual intervention.
- Tools like Jenkins, Maven, or Gradle manage builds.
- Enhances speed and consistency in delivery.
- Supports Continuous Integration and Continuous Delivery (CI/CD).
- Reduces human error in the build process.

Benefits of Build Automation

- Automated builds bring substantial benefits to software teams:
  - Ensures repeatable and reliable builds.
  - Speeds up integration and testing cycles.
  - Reduces configuration and version errors.
  - Supports continuous delivery models.
  - Frees developers for higher-value tasks

Version Control in Release Management

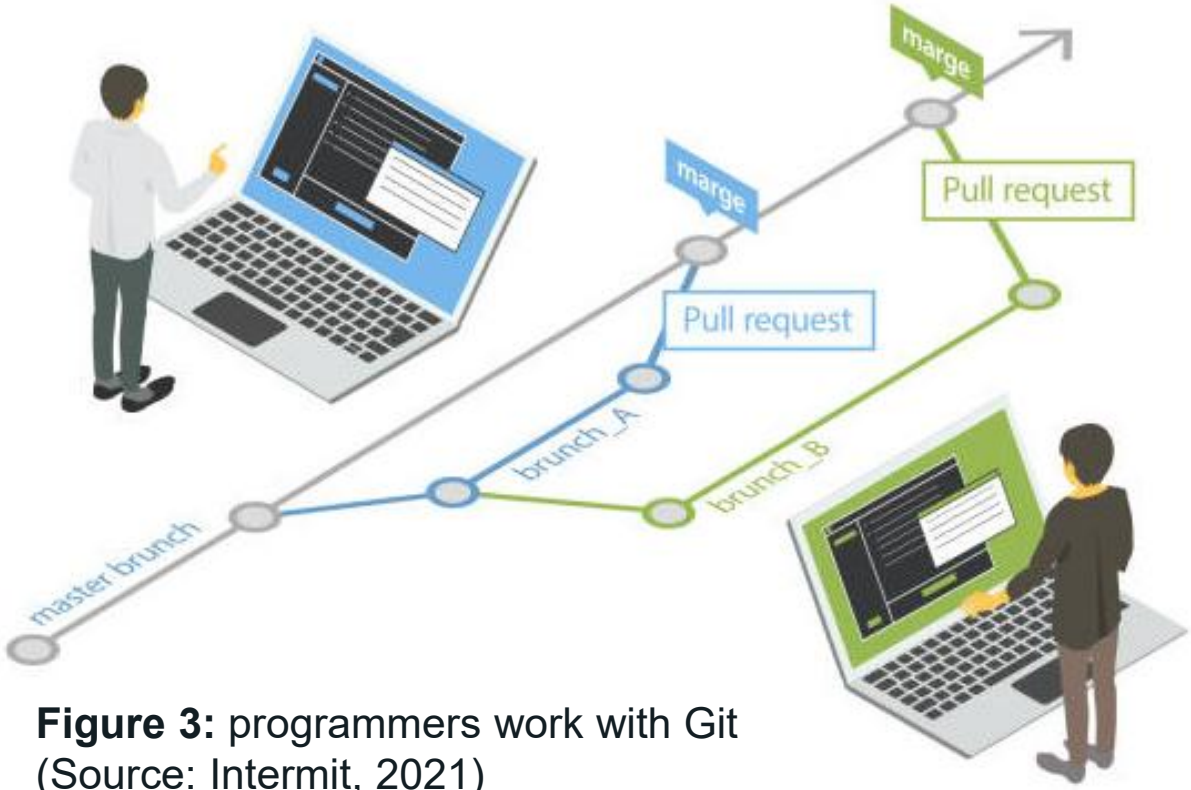


Figure 3: programmers work with Git
(Source: Intermit, 2021)

Version Control in Release Management … (2)

- Tracks code and configuration changes over time.
- Tools:
  - **Git, Subversion**
- Enables rollback to stable versions.
- Supports branch-based release strategies.
- Integral to build and release reproducibility

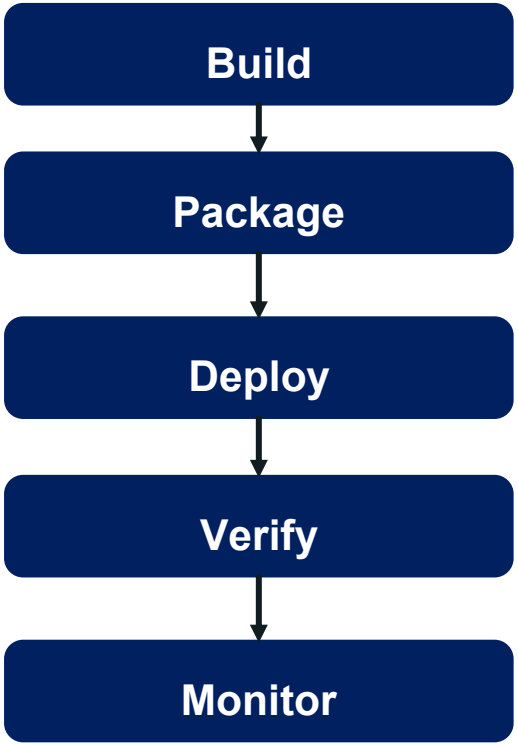
Importance of Version Tagging

- Version tagging is a best practice in managing software releases.
  - Tags mark specific points in repository history.
  - Used to identify official release versions.
  - Prevents confusion between builds and patches.
  - Helps reproduce past releases accurately.
  - Common convention: **v1.0.0, v2.1.3**, etc.

Software Delivery Process

- The software delivery process encompasses the entire journey of a build from development through deployment into production.
  - Moves software from development to user environments.
  - Involves build, package, deploy, and verify phases.
  - Must ensure integrity during each transition.
  - Requires coordination between multiple teams.
  - Supported by automation and release planning tools.

Phases of the Software Delivery Process



Compile and assemble code into deployable artifacts.

Group executables and configuration files

Move to target environment (test or production)

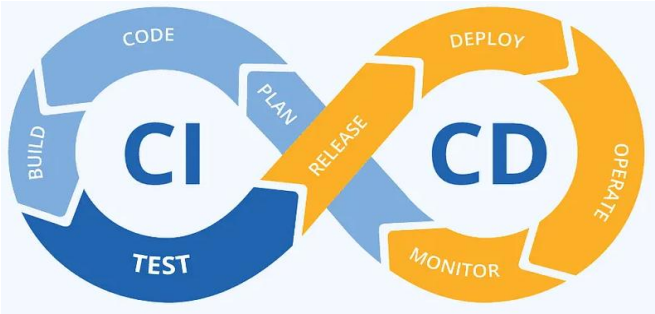
Ensure correct operation post-deployment

Track release health and performance

Deployment Methods

- Software deployment can take various forms depending on scale and risk tolerance.
  - Manual deployment through scripts or file transfers.
  - Automated deployment using pipelines or orchestration tools.
  - Blue-green deployment for smooth version transitions.
  - Rolling updates minimize downtime.
  - Canary releases allow limited testing in production.

Continuous Integration and Delivery (CI/CD)



- CI automates build and test after every code commit.
- CD extends automation to deployment and release.
- Enables frequent and reliable software delivery.
- Supports collaboration across development and operations.
- Reduces release risk and improves feedback loops

Figure 4: CI/CD (Source: Bhari, 2024)

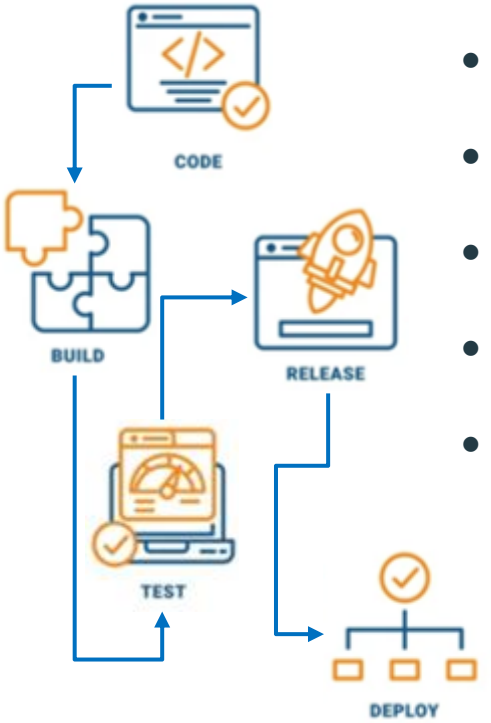
Packaging and Configuration Data

- The packaging phase transforms build outputs into deployable bundles.
  - Packaging combines binaries, scripts, and resources.
  - Configuration data adapts software to its environment.
  - Packaging ensures repeatable deployments.
  - Must include documentation and release notes.
  - SCM tools track package composition and changes.

Version Control Strategies for Releases

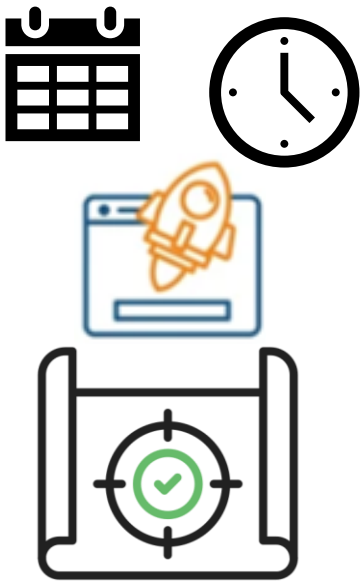
- Version control strategies determine how teams manage multiple streams of work efficiently.
  - Branching isolates release, development, and maintenance lines.
  - Merging integrates fixes into active versions.
  - Tags mark stable release points.
  - Feature branches support parallel development.
  - Proper strategy prevents integration conflicts

Build and Release Pipeline



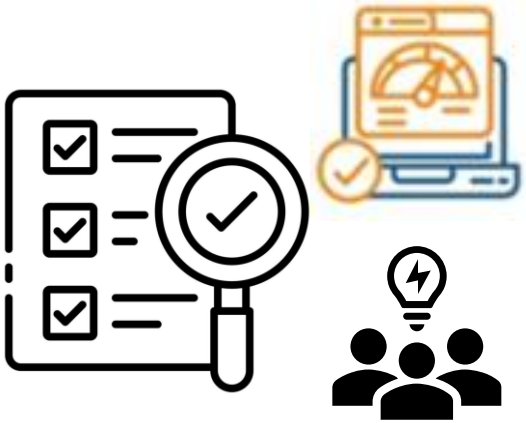
- Defines automated stages from code to deployment.
- Includes build, test, approval, and release steps.
- Pipelines ensure consistency and repeatability.
- Integrate version control and artifact repositories.
- Visualizes the flow of software delivery.

Release Planning and Scheduling



- Defines release scope, timing, and dependencies.
- Aligns development and testing milestones.
- Balances feature readiness with stability.
- Considers external factors (market, compliance).
- Documented in release calendars or roadmaps.

Criteria for Releasing Software



- Must meet predefined acceptance and quality criteria.
- Passes testing and configuration audits.
- Approved by the Change Control Board (CCB).
- Version documentation must be complete.
- Risks of release are evaluated and mitigated.

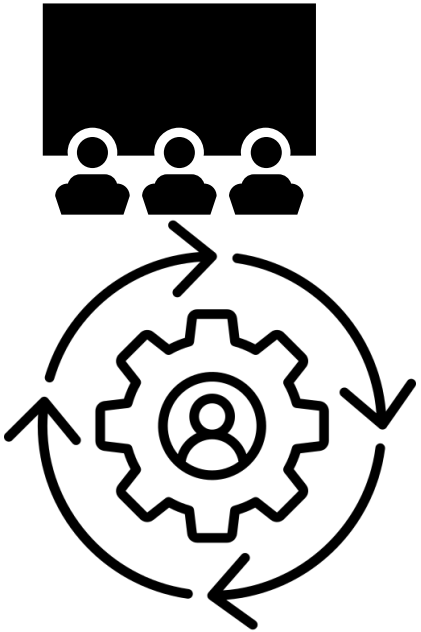
Documentation in Release Management

- Documentation plays a crucial role in effective release management.
  - Release notes summarize features and fixes.
  - Installation and upgrade guides support users.
  - Version Description Document (VDD) details contents.
  - Configuration records track software structure.
  - Documents must align with approved versions.

Approval and Sign-off Process

- Verifies completeness of release components.
- Confirms alignment with change requests.
- Requires testing and quality assurance reports.
- CCB provides final release authorization.
- Prevents unverified versions from deployment.

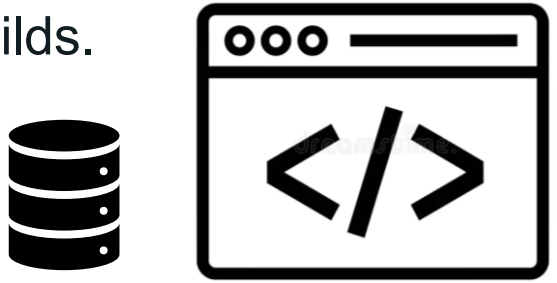
Role of the Change Control Board (CCB)



- Reviews and approves proposed release changes.
- Ensures alignment with organizational goals.
- Evaluates risk, cost, and benefit of changes.
- Authorizes emergency or interim releases when needed.
- Maintains accountability for release decisions.

Software Library in Release Management

- Stores approved and version-controlled software components.
- Divided into development, test, and production baselines.
- Ensures secure and traceable access to builds.
- Facilitates reproducibility of releases.
- Managed by SCM tools and policies.



Release Packaging and Variants

- Software releases often come in multiple variants tailored for different platforms or user environments, such as Windows, macOS, or Linux.
  - Packages include executables, documentation, and metadata.
  - Variants address different platforms or languages.
  - Each variant must maintain identical functionality.
  - Incorrect variant delivery causes customer dissatisfaction.
  - SCM tools help manage variant tracking.

Release Delivery Methods

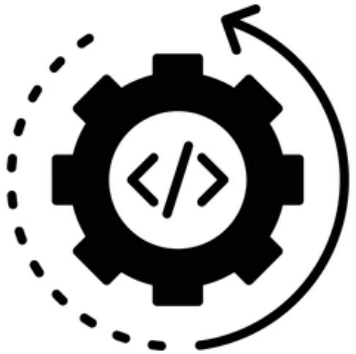
- The delivery mechanism must guarantee version integrity, security, and user accessibility while minimizing disruption.
  - Physical media (rarely used today).
  - Network-based delivery or FTP servers.
  - Cloud-based and containerized deployments.
  - App stores and web portals for user distribution.
  - Automated delivery through CI/CD pipelines

Release Risk Management

- Identifies risks before, during, and after release.
- Common risks:
  - **Integration failures, downtime, data loss.**
- Requires contingency and rollback plans.
- Automated testing mitigates delivery risks.
- Post-release monitoring ensures stability.

Rollback and Recovery Procedures

- Rollback procedures are critical safeguards in release management.
  - Rollback returns system to previous stable state.
  - Used when a release introduces major issues.
  - Requires proper version control and backups.
  - Must be tested and documented in advance.
  - Essential for high-availability systems.



Post-Release Activities

- The post-release activities are:
  - Monitor system performance and user feedback.
  - Document lessons learned and improvement areas.
  - Update issue trackers and backlog with findings.
  - Archive release artifacts and reports.
  - Prepare for maintenance or follow-up releases.

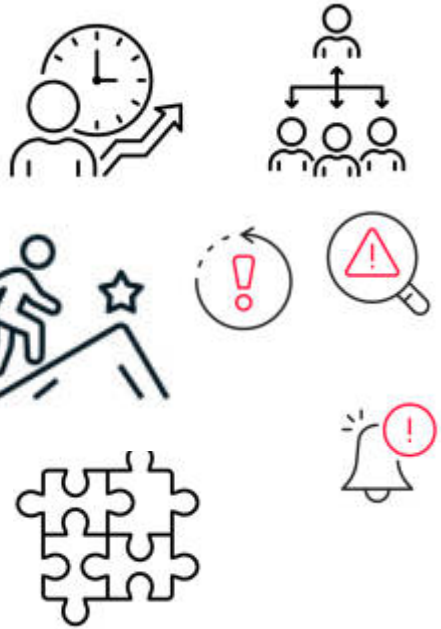
Metrics for Release Management

- Measuring performance is vital for improving release management.
  - Release frequency and success rate.
  - Mean time to recover (MTTR).
  - Deployment duration and error rate.
  - Number of post-release defects.
  - User satisfaction and performance indicators.

Continuous Delivery Pipelines

- Continuous delivery pipelines:
  - Automates code integration, testing, and release.
  - Ensures every build is deployable at any time.
  - Enables frequent, smaller releases.
  - Improves speed and reliability.
  - Supports agile and DevOps environments.

Challenges in Release Management



- Coordination across distributed teams.
- Managing multiple environments and dependencies.
- Ensuring rollback readiness.
- Balancing speed with quality.
- Maintaining accurate documentation and traceability

Best Practices for Effective Release Management

- Successful release management relies on discipline and automation.
- The best practices are to:
  - Automate builds, tests, and deployments.
  - Maintain strict version control policies.
  - Standardize packaging and documentation.
  - Conduct release readiness reviews.
  - Continuously monitor and improve processes.

Integration with Software Configuration Management – SCM

- Release management extends SCM principles, applying its principles to the delivery phase.
  - Ensures consistency across baselines and versions.
  - Builds depend on configuration-controlled items.
  - Audits verify correctness before release.
  - Strengthens the overall change management framework.

Summary

- Release management bridges development and operations.
- Build automation and CI/CD improve speed and reliability.
- Version control ensures traceable and reproducible releases.
- Documentation, approval, and audits guarantee quality.
- Continuous improvement strengthens future releases.

References

1. Leon, A. (2015). Software Configuration Management Handbook (3rd ed.). Norwood: Artechhouse. Retrieved September 4, 2025.
2. OpenAI. (2025, September 4). SCM history and concepts [AI-generated image]. ChatGPT (Sora). https://chat.openai.com/
3. Kv. (2023, July 19). What Is a Build in Software Development? Retrieved October 31, 2025, from Hackernoon website: https://hackernoon.com/what-is-a-build-in-software-development
4. Interemit (2021, July 24). Programmers work with Git [illustration]. iStock. Programmers Work With Git Isometric Vector Illustration Stock Illustration - Download Image Now – iStock
5. Bharti, A. (2024, August 25). *Understanding CI and CD: The Cornerstones of Modern Software Development*. Retrieved October 31, 2025, from medium.com: https://medium.com/@arunb9525/understanding-ci-and-cd-the-cornerstones-of-modern-software-development-ced62ccf1d59

Thank You!

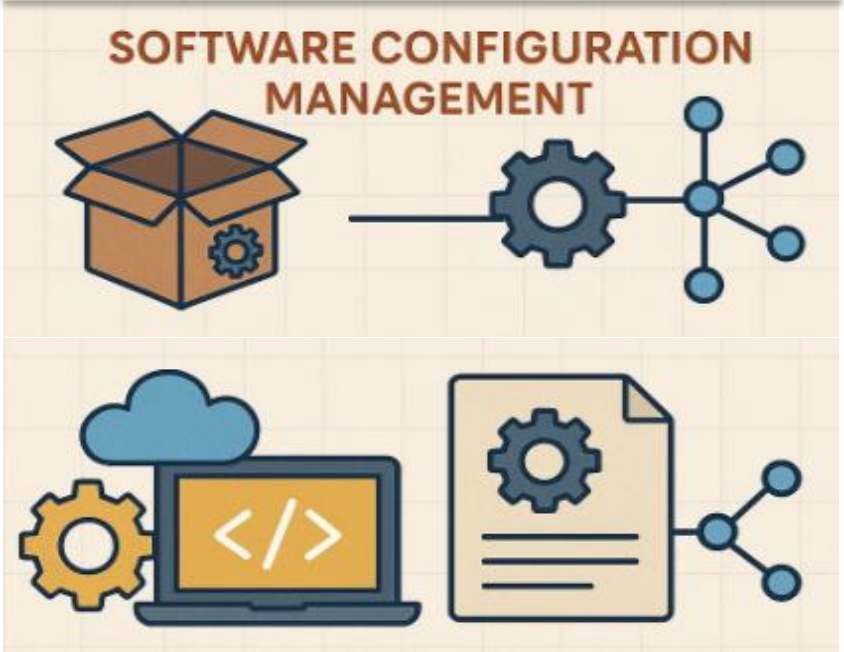


Figure 5: Concepts of SCM (Source: OpenAI, 2025)

NEXT LECTURE

Software Release Management and Delivery – Part 2