WHITE PAPER

RELEASE MANAGEMENT: PART 6

How Release Management Works: An Overview

- Deliver value faster to compete in the digital age.
- Reduce release risk by making it visible and ensuring teams complete the tasks that ensure quality, stability, security and compliance without slowing down.
- Learn key terms and concepts in release management.

PLUTORA

We've always had to release something to live to realize its value but a few key things have changed in recent history:

A Brief History of Release Management

Releases have shrunk: Digital transformation requires moving from waterfall, project-oriented ways of working to agile/DevOps, product-oriented ways of working. This means that large batches of requirements that are developed, tested and released become small, ideally-single flow items.

Releases have become more frequent: Because we are releasing in smaller batches, we are doing it more frequently. This allows us to be more experimental and receive faster feedback on whether the work we just made available to the customer has been received as intended – or not, and further action needs to be taken.

Automation is available: For a long time releases required a great deal of manual effort such as building (virtual) machines/ environments, performing tests, preparing and deploying release packages, opening ports, updating tickets, requesting permissions – often many, many steps. Individuals frequently wrote scripts to automate commonly repetitive steps – but these scripts often proliferated becoming a 'script monster' where only the creator knew the intricacies of how they were built and so what looked like an IT hero, was actually a Single Point of Failure (SPOF). Now, there is automation available from end-to-end in every release process (the DevOps toolchain) and release management tools like Plutora can manage all these processes across every team in an organization.

Continuous Integration avoids merge hell: Relating to both agile and automation, the aim is to continually deliver value. This is achieved by always having software in a releasable state thanks to trunk based development and continuous testing (continuous integration) where all developers commit code at least daily to trunk and every time they do so, a build is automated along with functional tests such as unit, integration and user acceptance as well as non-functional tests for security and performance.

The focus is on flow: Plutora is also a value stream management platform so it can use data from releases to ensure teams are able to see where they can optimize their release process and improve time to release, value and learning.

IT Service Management (ITSM) has always included the release management process and associated processes like change management, but they need to be adjusted to satisfy the demands in digitally transforming companies for increasing the speed and frequency of releases to compete in a disrupted world.

In an ideal world, all teams and systems would have sufficient autonomy and be sufficiently loosely coupled so that no dependencies exist, reducing a great deal of risk around the release process. Sadly, few organizations currently live in the ideal world. As teams continue to work on breaking dependencies, release management provides a platform to manage the transition to a truly autonomous organization.

Challenges that Release Management Addresses

Release management supports businesses who want to deliver value faster to compete in the digital age.

As teams attempt to release more frequently and faster, they often encounter higher levels of incidents and increased need for release nights and weekends. All this unplanned and out of hours work leads to burnout. Effective release management reduces the change failures associated with higher release and deployment frequency, easing the burden on the technology delivery teams and allowing them to do more useful things, like develop more value, pay down technical debt and improve delivery platforms.

Release failure isn't just costly for the teams; it's bad for the customers and when they have poor experience, it has a direct effect on organizational performance as brand reputation dips, usage lowers and customer retention and acquisition falters.

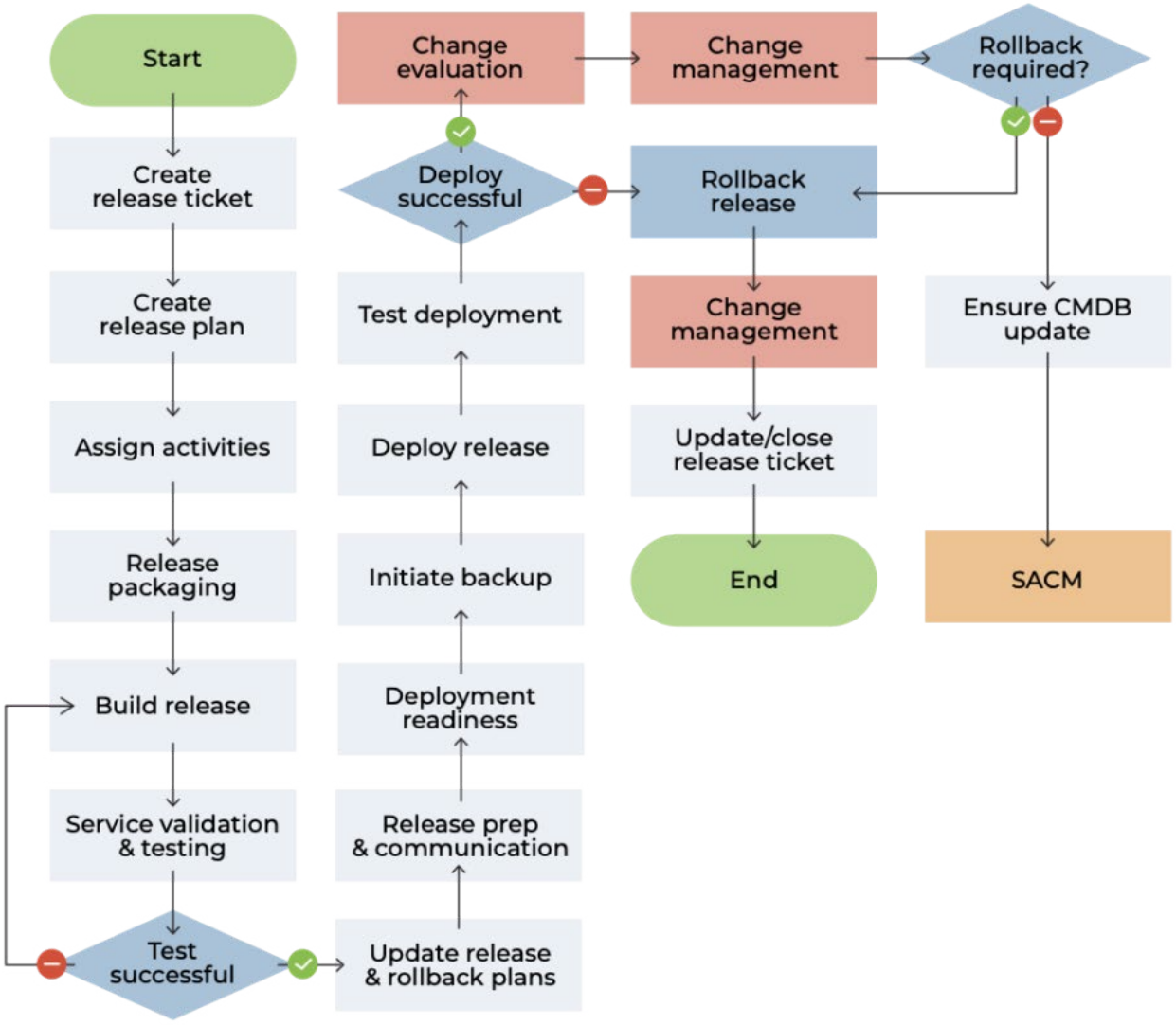
Value stream teams continually optimize the flow of work to their customers and the subsequent release of value. The release process is a key step in the value stream, so release management supports value stream management by identifying and removing waste and rework and unplanned work caused when the process fails.

How Release Management Works: An Overview

Release management aims to reduce risks associated with releases by making them visible and ensuring teams complete the tasks that ensure quality, stability, security and compliance (safety) without slowing them down. This is achieved by following a number of steps:

1. Identify your value streams: These are your products or services and each will have their own release process which may often have dependencies on other teams' activities.

2. Map each value streams' release process: A process map visually expresses the tasks in your release process. Here's an example:



3. Create release templates: These define the steps, tasks, or checklists and organize them into phases and gates and associated timelines.

4. Coordinate releases: A release schedule is essential for organizations with multiple teams working on multiple systems delivering changes.

It indicates that planning for the delivery of releases and changes has occurred and visualizes the contentions and dependencies around releases, providing the mechanism to make decisions around release dates and test environment allocations.

5. Standardize releases: Look for patterns within your portfolio of release processes and see how you can categorize them. Also, look for patterns that will allow you to understand and standardize tasks in the release process across all value streams.

6. Streamline releases: Each team must be empowered to self-discover improvements in their release process as part of their value stream. Local discoveries must be shared to become global improvements.

Key Terms in Release Management

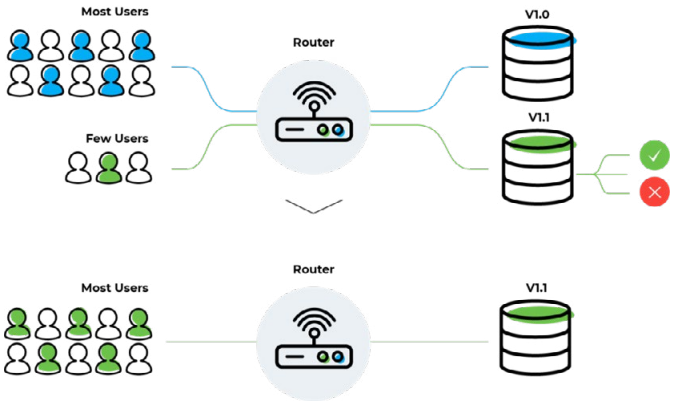
Application Release Automation (ARA) or Orchestration (ARO): Controlled continuous delivery pipeline capabilities including automation (release upon code commit), environment modeling (end-to-end pipeline stages, and deploy application binaries, packages, or other artifacts to target environments), and release coordination (project, calendar, and scheduling management, integration with change control and/or IT service support management).

Artifact: Any element in software development including documentation, test plans, images, data files, and executable modules. Some of these artifacts will form part of the release.

Backlog: Requirements for a system expressed as a prioritized list of product backlog items usually in the form of 'User Stories'. The product backlog is prioritized by the Product Owner and should include both functional and non-functional requirements. The delivery of these user stories form a release.

Batch Sizes: Refers to the volume of features involved in a single code release.

Canary Release: A canary release (also called a canary test) is a push of code changes to a small number of end-users who have not volunteered to test anything. Similar to incremental rollout, it is where a small portion of the user base is updated to a new version first. This subset, the canaries, then serve as the proverbial "canary in the coal mine". If something goes wrong then a release is rolled back and only a small subset of the users are impacted.



Change Failure Rate: A measure of the percentage of failed/rolled back changes released.

Change Lead Time: A measure of the time from a request for a change to the delivery of the change.

Change Management: The process that controls all changes throughout their lifecycle.

Continuous Delivery: A methodology that focuses on making sure software is always in a releasable state throughout its lifecycle.

Continuous Delivery Pipeline: A continuous delivery pipeline refers to the series of processes that are performed on product changes in stages. A change is injected at the beginning of the pipeline. A change may be new versions of code, data, or images for applications. Each stage processes the artifacts resulting from the prior stage. The last stage results in deployment to production.

Continuous Deployment: A set of practices that enable every change that passes automated tests to be automatically deployed to production.

Continuous Flow: Smoothly moving people or products from the first step of a process to the last with minimal (or no) buffers between steps.

Continuous Integration (CI): A development practice that requires developers to merge their code into trunk or master ideally at least daily and perform tests (i.e. unit, integration, and acceptance) at every code commit.

Deployment: The installation of a specified version of software to a given environment (e.g., promoting a new build into production).

Deployment Frequency: How often an enhancement to a value stream's product or service is deployed.

DevOps Toolchain: The tools needed to support a DevOps continuous development and delivery cycle from idea through release to value realization.

Flow: How people, products, or information move through a process. Flow of Value: A form of map that shows the end-to-end value stream.

IT Service Management: Adopting a process approach towards management, focusing on customer needs and IT services for customers rather than IT systems, and stressing continual improvement.

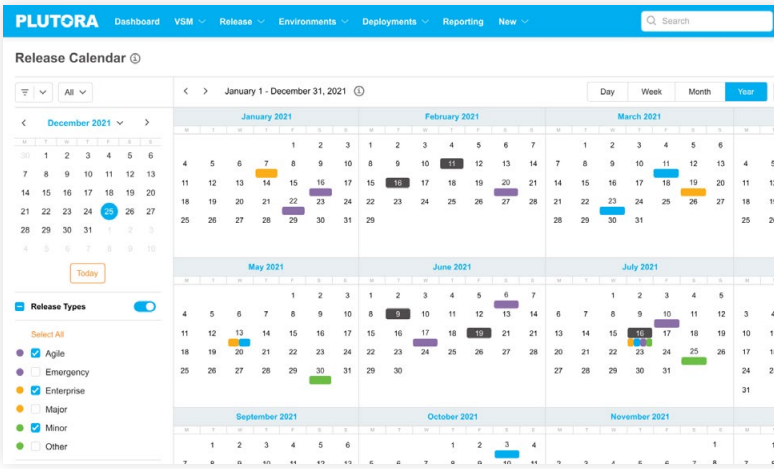
Mean Time Between Deploys: Used to measure deployment frequency.

Non-Functional Requirements: Requirements that specify criteria that can be used to judge the operation of a system, rather than specific behaviors or functions (e.g., availability, reliability, maintainability, supportability); qualities of a system.

Release: Software that is built, tested, and deployed into the production environment.

Release Acceptance Criteria: Measurable attributes for a release package that determine whether a release candidate is acceptable for deployment to customers.

Release Calendar: A rolled up view of all releases occurred, happening and in plan at any point in time.



Release Candidate: A release package that has been prepared for deployment, may or may not have passed the Release.

Release Frequency: How often an enhancement to a value stream's product or service is released.

Release Governance: The controls and automation (security, compliance, or otherwise) that ensure your releases are managed in an auditable and trackable way, in order to meet the need of the business to understand what is changing.

Release Management: The process that manages releases and underpins

Continuous Delivery and the deployment pipeline.

Release Manager: A role accountable for the overall quality of the release management process. There may be a team of release managers covering multiple value streams, or teams may have someone dedicated or not to the role.

Release Orchestration: Typically a deployment pipeline used to detect any changes that will lead to problems in production. Orchestrating other tools will identify performance, security, or usability issues.

Release Package: A combination of one or more release units deployed together as a single release due to interdependencies, scheduling, or business priorities

Release Pipeline: A specific release process from feature planning to delivery.

Release Plan: A forecast of the activities planned to deploy a release to the production environment.

Release Policy: The definition of release types, standards, governance requirements for an organization.

Release Process: All the tasks (manual, automated, human and technical) needed to make completed changes available to the customer.

Release Process Map: A visual representation of the release process that can be collaborated on within and between teams.

Release Template: A framework document that can be completed with the steps, tasks, or checklists associated with a release (and related releases) that organizes them into phases and gates and associated timelines.

Release Train: The release train is a technique for coordinating releases across multiple teams or components that have runtime dependencies.

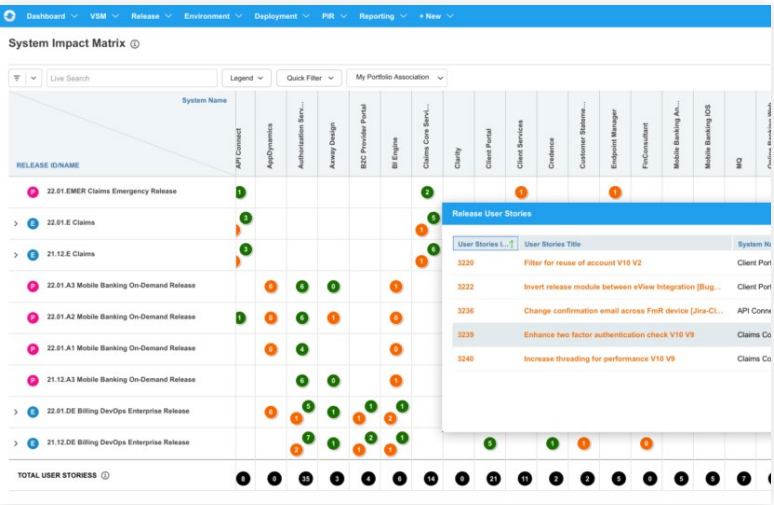
All releases happen on a fixed and reliable schedule regardless of whether all expected features are ready (the train doesn't wait for you – if you miss it you wait for the next one).

Release Unit: The set of artifacts released together to implement a specific feature.

Risk: A possible event that could cause harm or loss or affect an organization's ability to achieve its objectives. The management of risk consists of three activities: identifying risks, analyzing risks, and managing risks.

The probable frequency and probable magnitude of future loss. Pertains to a possible event that could cause harm or loss or affect an organization's ability to execute or achieve its objectives.

System Impact Matrix: A diagram that shows where releases contain shared components that represent risk.



Time to Learning: The time between conceiving an idea and learning how it was received based on customer feedback.

Time to Market: The period of time between when an idea is conceived and when it is available to customers.

Time to Release: The time between conceiving an idea and releasing the change that satisfies the Definition of Done to the live, production environment.

Unit Test: The purpose of the test is to verify code logic.

User Stories: These express requirements in the product backlog and are likely to be tagged with unique identifiers that will become useful as you automate.

Value Stream: All of the activities needed to go from a customer request to a delivered product or service.

Value Stream Management: Value Stream Management is a combination of people, processes, and technology that maps, optimizes, visualizes, measures, and governs business value flow through heterogeneous software delivery pipelines from idea through development and into production.

Value Stream Management Platform (VSMP): Software that manages value streams.

Key Concepts in Release Management

Release management exists because of dependencies. If each team had autonomy and no systems were interconnected, then, assuming each team was correctly ensuring their own quality, security, stability and compliance, there would be no need to manage releases.

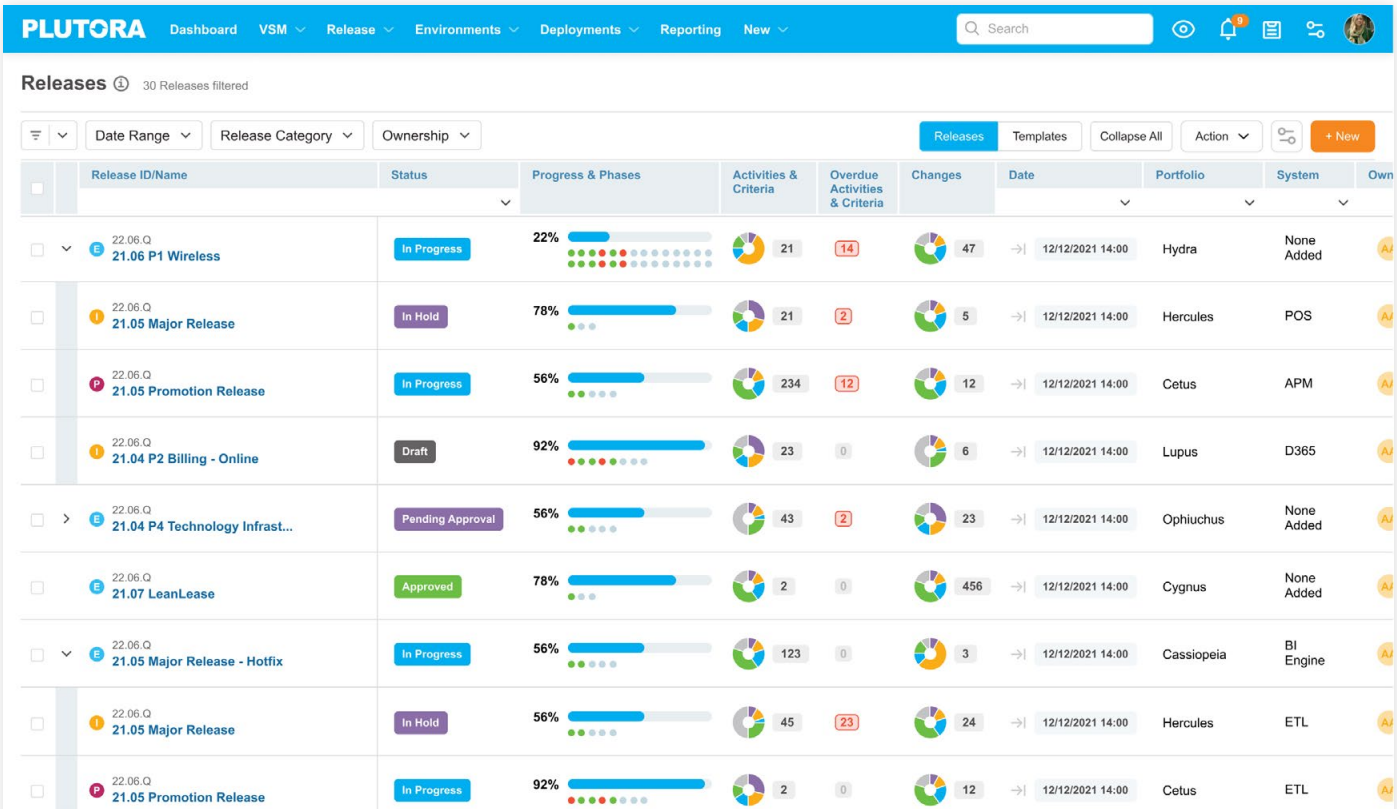
Release managers exist to help teams ensure their activities, performed in the spirit of moving their own team forward, don't negatively impact other teams.

Release calendars exist to provide visibility across an organization of all the teams' activities so that risks can be surfaced and mitigated.

Release nights and weekends occur because releases are dangerous and need to happen at times when they are less likely to be customer impacting. Staff have to be at work or on call out of hours during this time and they are a key cause of burnout, dissatisfaction and employee disengagement in the industry.

Release management supports the transition to a place where:

- Dependencies are identified, managed and ultimately removed where possible
- Automation delivers continuous compliance and the focus is on value flow
- Releases are frequent, small and risk-free



Want to learn more about release management?

Check our series of white papers about Release Management to learn everything you need to know.

1. What is a Software Release?
2. What is Release Management
3. The Benefits of Release Management
4. Release Management Techniques for Process Improvements
5. How Release Management Works: An Overview
6. Release Management, DevOps, and Agile

Visit www.plutora.com/software-release-management to learn more.

About Plutora

Plutora, the market leader of value stream management solutions for enterprise IT, improves the speed and quality of software creation by capturing, visualizing and analyzing critical indicators of every aspect of the delivery process. Plutora orchestrates release pipelines across a diverse ecosystem of development methodologies, manages hybrid test environments, correlates data from existing toolchains, and incorporates test metrics gathered at every step. The Plutora Platform ensures organizational alignment of software development with business strategy and provides visibility, analytics and a system of insights into the entire value stream, guiding continuous improvement through the measured outcomes of each effort.

PLUTORA

Learn more: www.plutora.com
Email: contact@plutora.com