**Executive Summary**

**Enterprise Release Management – Outline**

**Cognizant DevOps Practice**

**Swaminathan Ramamoorthy**

**Proprietary Notice**

Table of Contents

[1 What is Enterprise Release Management from DevOps perspective? 4](#_Toc455997270)

[1.1 Art of Release Management 4](#_Toc455997271)

[1.2 Enterprise Release Management (ERM) 4](#_Toc455997272)

[1.3 Basic tenets of ERM 4](#_Toc455997273)

[2 Why Enterprise Release Management is required? 5](#_Toc455997274)

[3 How to Organize Enterprise Release Management? 6](#_Toc455997275)

[4 Release Lifecycle Workflow 7](#_Toc455997276)

[5 Key Building Blocks of Release Planning: 8](#_Toc455997277)

[6 Key Focus areas in Enterprise Release Management 9](#_Toc455997278)

[6.1 Release Calendar 9](#_Toc455997279)

[6.2 Multiple Release Train & Train Seat Concept 10](#_Toc455997280)

[6.3 Release Standards and Process 11](#_Toc455997281)

[7 Using Enterprise Release Automation Tools: 12](#_Toc455997282)

[8 Recommendations for Successful ERM 12](#_Toc455997283)

[9 Related artifacts 13](#_Toc455997284)

[9.1 Enterprise Release Flow Diagram 13](#_Toc455997285)

[9.2 Use Case for Release Management 13](#_Toc455997286)

# What is Enterprise Release Management from DevOps perspective?

## Art of Release Management

We can portray Release Management is an art because the build and delivery of software might sounds like simple task, but it is not real scenario. The coordination between various teams, tracking of build versions, and transparency of releases is not for the faint of heart. There is no “straight forward or simple way to achieve a well-defined and smooth workflow for all the teams to follow. The release management process is always in a state of continual improvement. Release Management is all about handling the Risk Management.

Enterprise Release Management (ERM) isn’t a new concept, or even a new term. However, it is becoming more prominent, and it is starting to enter common usage amongst senior IT managers. Enterprise Release Management is also not an ITIL term, but it spans many of the Service Transition processes and capabilities, and is relevant for the context of complex and frequent project delivery, whilst supporting simpler technical and maintenance releases. Many organizations already have Release Managers, and the job description of a ‘Release Manager’ can vary wildly from one organization to the other, but very few have an ERM person or an ERM as a separate function.

## Enterprise Release Management (ERM)

The Enterprise Release Management (ERM) function is branded primarily by the *complex eco System,* *wide scope, and input variety of releases under its control*. It is not concerned with specific technologies, business units or projects, and giving your IT organization proactively on the forward view of all releases, and highlighting their integration points and dependencies. Enterprise Release management provides comprehensive functionality for using the process to conduct actual releases in your data center

## Basic tenets of ERM

Today it is common for every product to have a built-in configuration ability. Android, for example, is highly customizable at many layers, including features for development, testing, and end usage. When all the products have configuration built into them, a centralized body is required to ensure from beginning to end that requirements are made into a complete, sellable product.

We also need to consider how these requirements are transformed to code, tested, verified against the requirements, and then packaged to be released to the customer in a production environment.

Enterprise Release management provides these capabilities. It is the glue that holds all the key departments of a software development lifecycle together, including business and functional requirements, design, development, testing, deployment and ongoing maintenance.

The basic tenets of release management are as follows:

**Release Planning**: As a RMO, we need to plan ahead to address all these questions like:

* What are the systems?
* Is there a grouping of Systems?
* How many releases?
* Is there any defined release schedule?
* When to release?
* How much Overlap?

**Release Life Cycle**: Each Release will have a Life Cycle and phases (i.e.) Initiation, Planning, Build, Deployment and Close Out. Also we need to identify each phase by a key Milestones and Manage these release by a milestones

**Importance of Change Management**: Release managers are the gatekeepers to the production code. Any software code artifact that moves out of the organization has to be known to the release manager and should be approved by change management function.

**Release Metrics measurement and productivity:** With time, different metrics and measurement should be collected to help understand the throughput of releases given that the organization has some measurable skill set of people, technology, and current mature processes.

**Release Automation Tools:** RM Tools help organizations release software more quickly with higher quality. To automate the release process including management of the release process and visualization of the software delivery cycle, handoffs between stages of the process and automated deployment of software packages on environments.

# Why Enterprise Release Management is required?

**Enterprise Release management is so difficult, why do we need it?**

What benefits will come from the effort? Although it is not easy to implement in any organization the benefits of release management and DevOps out weight the effort and cost substantially.

There might be several reason why Enterprise Release Management adopt few strategies:

**Complex operational infrastructure.** The greater the complexity of your operational infrastructure the greater the risk that the release of new functionality into production will break something, hence the greater need for release management. Operational infrastructures become complex when there are many technologies in place, when there are different versions or configurations of those technologies, and when solutions are highly coupled to one another. Ideally you should strive to pay down this technical debt.

**Many delivery teams working in parallel**. Your operational infrastructure is a shared environment, or more accurately a collection of shared environments, that your IT delivery teams deploy into. As the number of delivery teams rises, the greater the chance that their release efforts will conflict with one another.

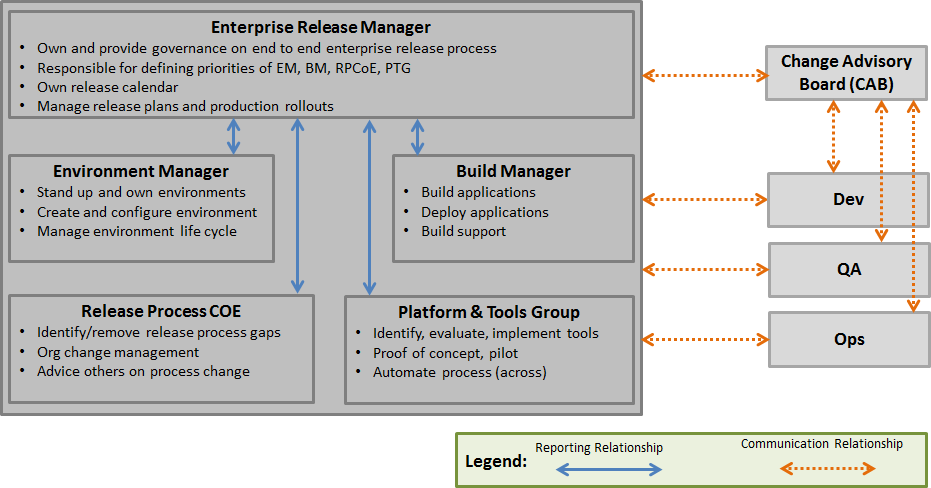
**IT delivery teams need help to release their solutions into production**. Your IT delivery teams, particularly new ones, may not have much experience deploying solutions into your operational environment. Your release management team can coach your IT delivery teams in effective release strategies, can guide them in ensuring that their solutions are production ready, and can help in the planning and coordination of their release efforts.

# How to Organize Enterprise Release Management?

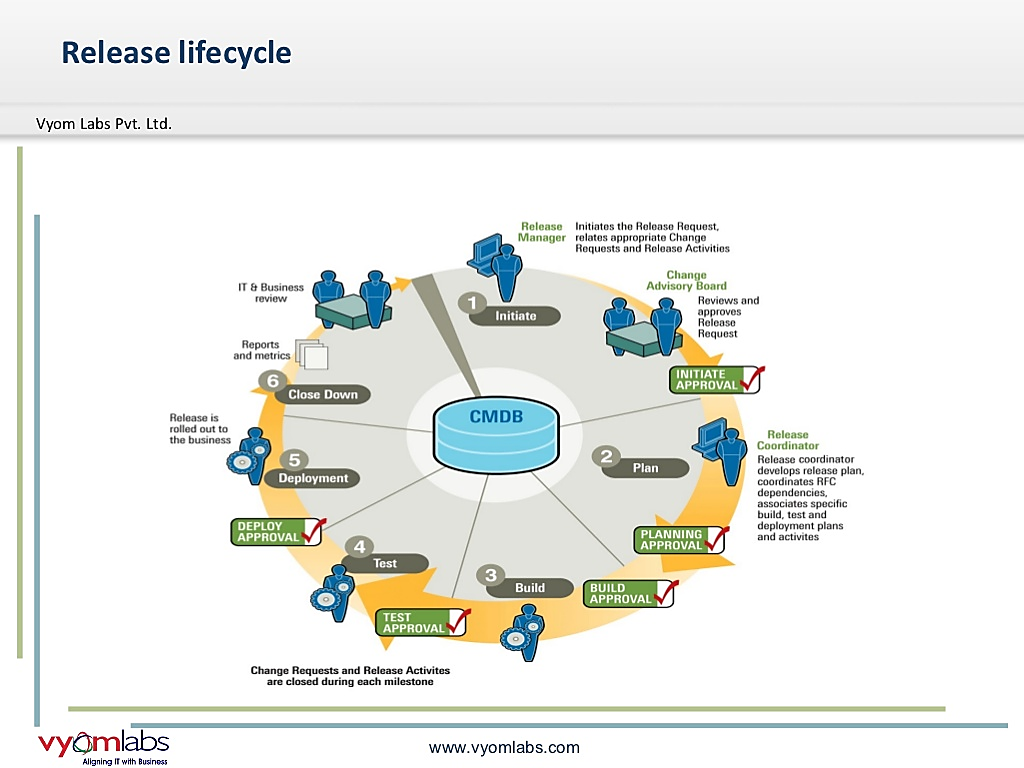
**There are few pointers to consider how to organize the Enterprise Release Management Team?**

Release Management should drive priorities and accountable for Release Enablement. Key Actors are like:

1. Enterprise Release Manager
2. Environment Manager
3. Build Manager
4. Release Process COE
5. Build Manager and CAB, DEV, QA & Ops



# Release Lifecycle Workflow

When using Release Management, here are the steps that you typically follow:

# Key Building Blocks of Release Planning:

We are dividing the release planning in 4 Foundation blocks like

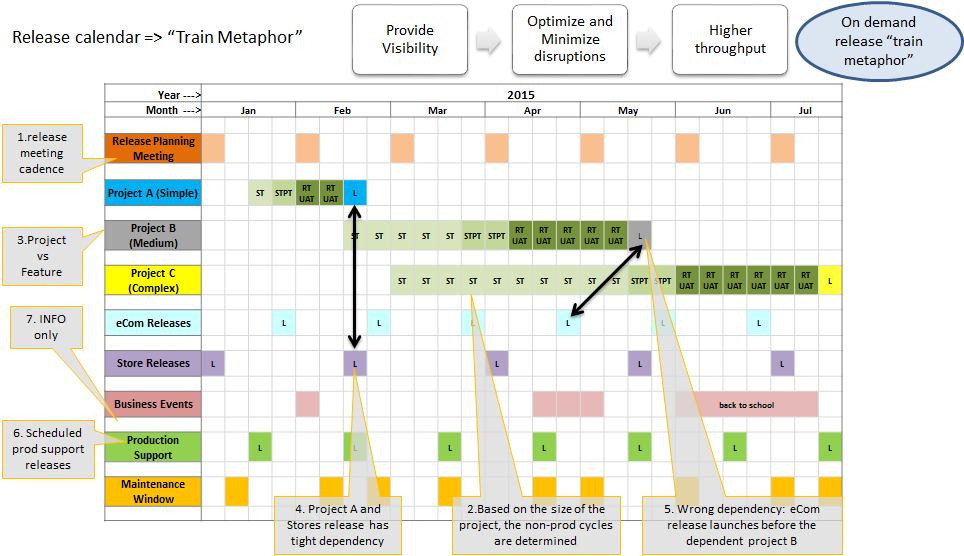
1. **Planning and Scheduling:** Under Planning and Scheduling, we are having few tasks like :
   * Intake Management
   * Prioritization
   * Communication cadence
   * Dependency management
   * Deployment plan
   * Release calendar
   * Release Train and Train Seat
2. **Different stages in Release Life Cycle**
   * Stage definitions
   * Entry and Exit criteria
   * Influencing factors like Environment and so on…
   * Delivery Governance
   * Build verification
3. **Post Feedback**
   * Retrospective discussion (What went well? & What did not go well?)
   * Root cause Analysis
   * Where to fix ?
   * Identify and delegate to block2
4. **Escalation and Emergency**
   * Rollback Strategy
   * War Room Discussion
   * Escalation and Goverence Control

# Key Focus areas in Enterprise Release Management

There are some focus areas to formalize the Release Management Process. An important point to be made is that there are several options available to you, each of which has its own advantages and disadvantages. No single approach is perfect, and no single approach works in all situations. You not only need to have choices, it’s good to have choice:

| Focus Areas | Outline |
| --- | --- |
| *Release Calendar* | *Easy accessible release calendar (SharePoint based). Create the calendar template with all specific inputs and workflow. But it might require Requires SharePoint development effort* |
|  | *Consolidation of releases schedules [smaller and similar] to reduce the overall release cycles* |
| *Time to Market* | *Better stage gates for release acceptance in line with existing SDLC / workflows* |
| *Release Train & Train Seat Concept* | *To define interval release dates planned in advance. Detailed write-up given below* |
| *Release Planning* | *Early release & capacity planning in line with Demand Management for the application releases* |
| *SOPs and Checklists* | *Checklists and SOPs inline to industry standards – Socialize the themes with project teams* |
|  | *Consolidate all checklists and SOPS under one repository for easy access across multiple deployments* |
| *Release Demand – Adhoc* | *Generate period metrics [KPI] to ensure and measure the efficiencies / forecasts.* |
| *Deployment Failures* | *Schedule deployments for lower environments based on environment availability and bandwidth to avoid failures* |
|  | *Ensure that a back out strategy is defined to make sure services are not compromised by new releases* |
| *Deployment Automation* | *Enable existing Continuous Integration towards auto deployment across lower environments via plugins and customized scripts (Distributed technologies)* |

## Release Calendar

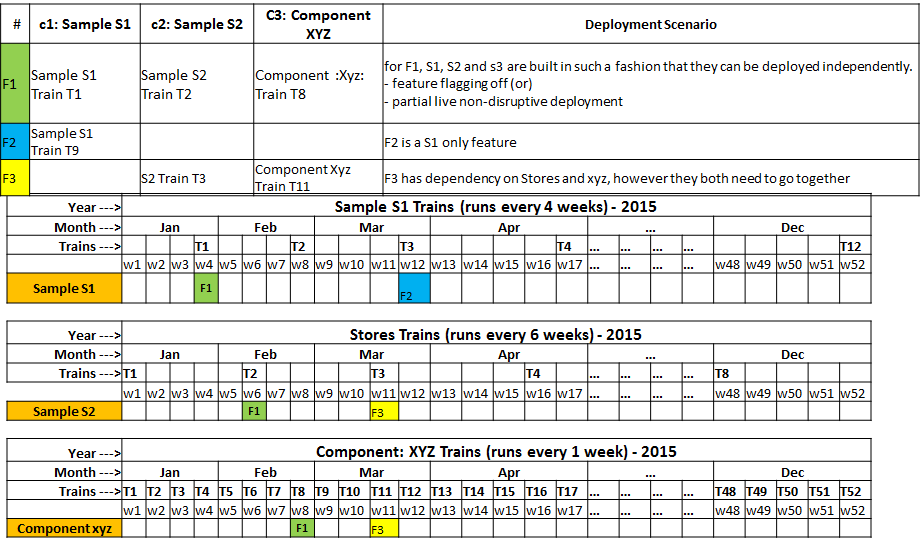
****

**Release Calendar > Why we require?**

* **Release discipline, effective release planning**
* **Phasing feature Delivery -**
* Optimizing/tightening QA cycles
* Lowered risk when we release in chunks
* **Enterprise Train Model**
* Individual project teams decide which train to pick in the enterprise train
* Organization knows, thinks and plans for “trains”
* **Multiple Factors determine “which Train?”**
* Business, Component ability, Component dependency
* QA cycle, Quality of product built, Prioritization

## Multiple Release Train & Train Seat Concept

* **Release Train :**
* Defined interval release dates planned in advance
* Indicates the target date in which the feature has to go live
* Each Train is assigned a specific seat capacity
* **Train Seat :**
* “Train Seat” is a unit of measure to define the project effort
* Allotment of project to a train is subject to availability of train seats in that train
* Each project is allotted target train number and number of train seats
* **How to break a big project drop?**
* Option 1: Find out possibilities of how the platform can be rolled out to production (released) and turned on end users to serve portion of the feature planned
* Option 2: When multiple functional components need to come together for a complete platform or feature release, when component 1 is ready it will be released and turned OFF
* Functional Architect and Technology Architect, by working with Dev Leads can find out possibilities of how the platform can be released in multiple bundles, instead of single release
* **Why multiple drops preferred?**
* Helps in phased out testing and production validation
* No bug is a Sev 1 or 2 bug during Trains 1,2,3
* QA Integration Env need not be end to end Integrated Env during initial trains

****

## Release Standards and Process

* Review the existing Release management Process
* Identifies opportunities to make tactical improvements to the release management process.
* Documentation of all processes.
* Creates release management standards to be followed. Educates external vendors on the release methodology and strategy.
* Collects feedback from the users of release management standards and identifies the need to modify or retire a standard based on the feedback.
* Integrates the release process with other IT processes including ITSM and Waterfall/Agile processes.
* Works closely with Change and Configuration Management team of the ITSM Unit to stay current on all change, configuration and internal controls standards and processes.
* Optimize the test environment usage
* Control all the activity through release stages
* Continuous Communications

# Using Enterprise Release Automation Tools:

ERA tools help cultivate DevOps best practices by providing a combination of automation, environment modeling and workflow management capabilities. These practices help teams deliver software rapidly, reliably and responsibly. ERA tools achieve a key DevOps goal of implementing continuous delivery with a large quantity of releases quickly.

**List of Key Release Automation Tools:**

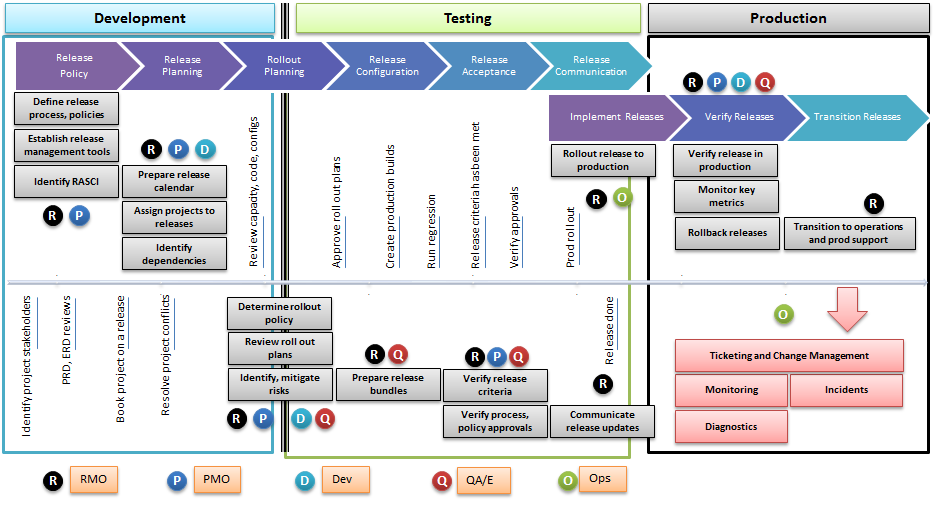
|  |  |
| --- | --- |
| ERA - Tool Name | Released By |
| Automic Release Automation | Automic |
| Release Lifecycle Management | BMC Software |
| CA Release Automation | CA Technologies |
| Clarive | Clarive |
| ElectricFlow | Electric Cloud |
| Codar | HP Software Division |
| UrbanCode Deploy & UrbanCode Release | IBM |
| BuildMaster | Inedo |
| Visual Studio Release Management | Microsoft |
| RapidDeploy | MidVision |
| Release Engineer | OpenMake Software |
| Serena Deployment Automation | Serena Software |
| Code Stream | VMware |
| XL Deploy & XL Release | XebiaLabs |

# Recommendations for Successful ERM

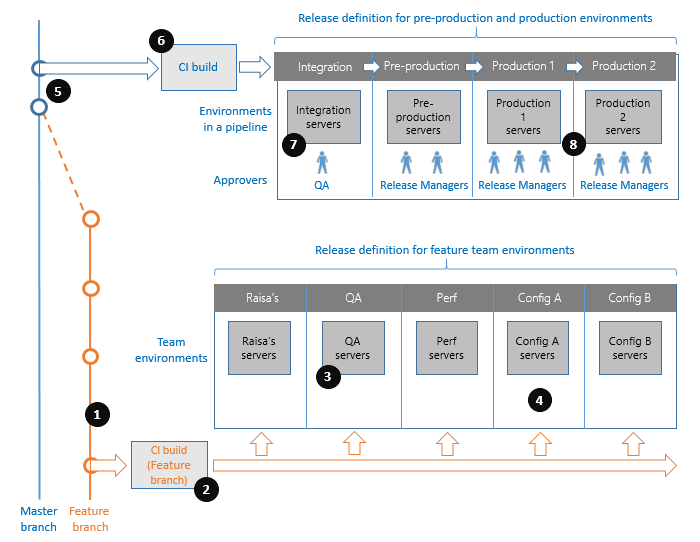
Release Management is all about controlling the release process, coordinating information about all your releases, securing the path to production and automating your release process and deployments. The following good practices are based on many years of implementing and observing software Release Management across enterprises. They are listed in no specific order:

# Related artifacts

## Enterprise Release Flow Diagram



## Use Case for Release Management



|  |  |
| --- | --- |
| Step | Description |
| #1 | You are part of a development team that follows an Agile software development process. During a Sprint, members of your team pick up product backlog items and commit their changes to a **Feature** branch |
| #2 | A CI build continuously integrates the changes in your **Feature** branch |
| #3 | Developers want new releases to be regularly deployed to a **Team QA environment** (3), where automated functional tests are run |
| #4 | In addition, they want to have a number of environments set up regularly with the latest code for **manual**, **performance**, or **multi-configuration testing** |
| #5 | At periodic intervals throughout the Sprint, your team integrates the **Feature** branch with the **Master** branch. |
| #6 | Related applications developed by other teams are constantly integrated into the master branch as well. To ensures that the commits into that branch are of high quality, a CI build of the master branch runs for each check-in |
| #7 | You want to deploy every new release in the master branch to an **Integration** environment and run functional tests |
| #8 | Your Release team manages a **Pre-production** environment and a series of **Production** environments. Early adopters of your applications are hosted in the first **Production** environment, while others are hosted in later **Production** environments |

**Conclusion of this given Use Case**:

At the end of every Sprint, the operations team wants to take a release that was successfully deployed to the **Integration** environment, and promote it through **Pre-production** and **Production** environments in turn. Upgrade tests are run in the **Pre-production** environment to minimize problems that might arise in the **Production** environment. You rely on feedback from early adopters to decide whether a release should be promoted from one **Production** environment to another.

Your release managers want to track the progress of all deployments. When a deployment fails, or when tests fail, developers need access to logs for immediate diagnosis. And your operations teams must be able to roll back a deployment if the fix for an issue will take too long.

When an older release is to be deployed, release managers need to understand all the features that will be rolled back. When the fix is finally ready, you want to track it all the way to **Production**, and you need to confirm that the issue your customers encountered is, indeed, fixed.