ITIL Maps Nikhil Aditya Rao Balkunje

Release and Deployment Management

ABSTRACT:
This paper highlights key factors for automating the Release design , Release plan, Release build,Deployment ,Early Life Support and Release Closure

Table of Contents

1 Introduction

An application release involves the deployment of an entire application or its components/modules to production, ready to be used by end customers. The completion of this process is a significant milestone for any business. Today's applications typically contain hundreds to thousands, of components for application servers, databases, web servers, messaging middleware, and authorization services. Each component must be installed and configured correctly to work with the versions of every other component making up the application baseline. Coupled with this complexity is the ever increasing volume of application releases.

The current scenario of application development to deployment is represented below:

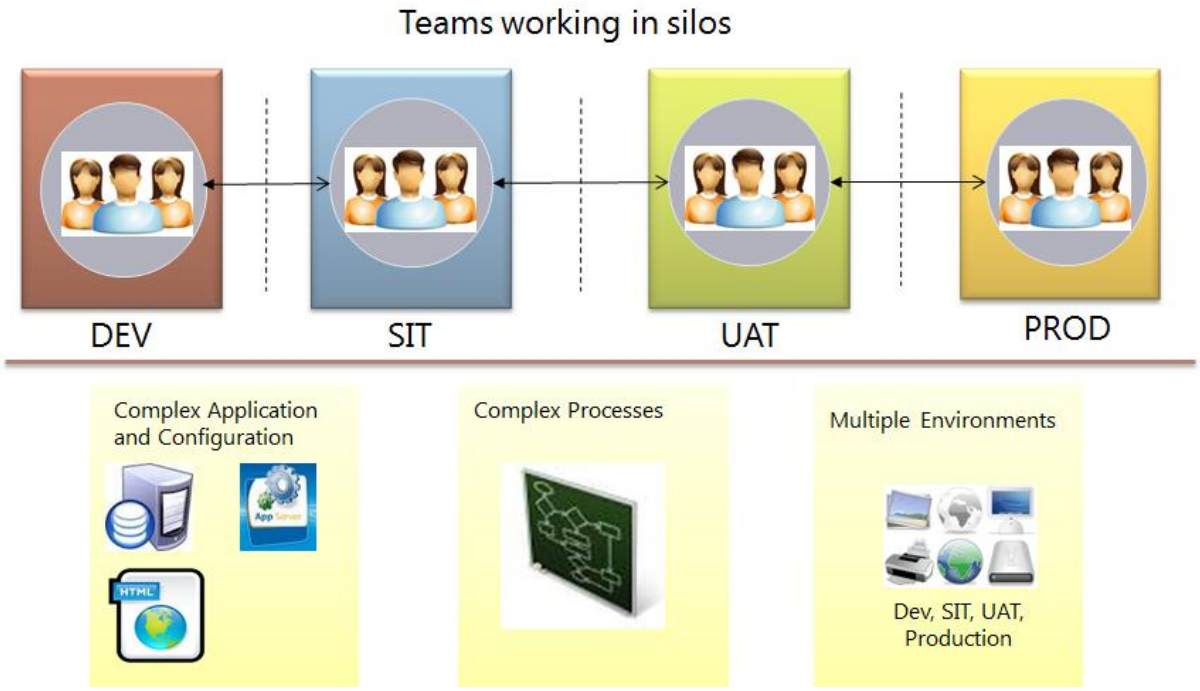

Figure 1 Current Scenario

2 Objective of Release Management

The objective of release management is

The first point is about ensuring that an existing working application is not replaced with a non-working one. The second point is about not introducing a working application which breaks other working applications or components.

In real life, we see both these cases happening frequently. So, to reiterate, the agenda of release management is totally production, operation, infrastructure availability focused.

3 Understanding application release and deployment challenges

Existing manual and semi-automated approaches result in uncoordinated, time-consuming, error-prone and non-scalable release mechanisms draining both effort and budgets. The typical problems faced by enterprise release and deployment management functions are:

High volume and frequency of releases

Typical banks have on an average more than 1000 changes, production issues and enhancements to be done and moved to production. Assuming an average of 4 environments and number changes/fixes/enhancements are delivered incrementally, the number of promotions is easily closer to a 5 digit number.

Wide technology stack coupled with heterogeneous environments

Added to these is the wide technology stack- horizontally and vertically. i.e. each application spans across multiple technologies. Portfolio of applications is spread across multiple technologies.

Expert Overload

Due to the multiple technologies involved, Manual (or partially manual) deployment processes remain dependent on experts with specific application knowledge and technologies. Crucial deployment knowledge is in the heads of people, not stored and automated in systems.

Excessive costs and delays

Manual deployments involving multiple experts across multiple environments is time consuming. This translates to higher costs and schedule delays.

Outages

According to a 2010 Gartner, Inc. Research report, “through 2015, 80% of mission critical outages will be caused by people and process issues, and more than 50% of those outages are caused by change/configuration/release integration and handoff issues.

Bulky deployment manuals or Custom Scripts

Manual deployments are guided by bulky manuals and contain endless sequences of deployment steps. It is extremely difficult to keep these manuals up-to-date each time a change occurs in the physical infrastructure, middleware or the application itself. Custom scripts created to automate the deployments steps are also tedious and need to be updated when there is a change.

Non-orchestrated tool sets

Release and Deployment activities are managed using a plethora of tools. These are not orchestrated to provide a seamless end to end release and deployment process

Lack of visibility

Organizations are unable to trace whether contents of the package being promoted to various environments are the baseline contents (e.g. whether the UAT certified build is the same one that is deployed into Production)

4 Solution to the release and deployment challenges

Application release and deployment automation is the automation of the release planning, control and entire deployment process from build all the way to production. A Release Automation platform is a toolset used to enable a more accurate, reliable and accelerated release and deployment experience for business and to offer an overall simplification of the complex release management process.

The automation platform typically covers

- **Planning** of release, dependencies and schedules
- **Monitoring and control** of environments, consistency and access
- **Automation of deployment**, rollback and configuration

Let us see how an automation platform changes the way release and deployment happens.

4.1 Release Planning

In manual release management, Planning is done using excel sheets. It is very difficult to track the multitude of tasks to be completed for each release, their dependencies. Also it is very difficult to map to the environments that will be used and any potential conflicts.

The automation platform should enable planning out the release calendar providing a holistic view of all upcoming releases. The various tasks under each release can be scheduled and the dependencies mapped. The dependencies between releases can be mapped to enable identification and resolution of release conflicts

Test environments required for a particular release can be reserved. This enables a view of what is happening on each environment at any point of time. Configuration parameters for the particular release are defined; this provides the flexibility to customize the deployment according to that particular release context. The process flow can be customized based on release types.

4.2 Release Control

In manual release and deployment activities multiple personnel need to have access to environments, component storage locations. This spawns a number of issues related to environment inconsistencies, security and compliance. Tracking the reason for a release failure is a time consuming activity. The dynamic schedule changes to various releases also create potential resource conflicts which are difficult to detect.

The release control aspect of the automation platform enables tracking the progress of releases and identification of both process and application issues. The platform identifies issues such as, a release package not having an associated application version or configuration files required to guide the roll out, also alerts are provided for rollout issues, resource conflicts if environments are used for something else.

The platform provides a clear audit trail mechanism of who approved a particular release, when was the release promoted to a particular environment, issues if any, roll back status etc making compliance a breeze.

4.3 Automation Deployment

When software developers decide their build package is ready, it is automatically passed to the solution. The build package then moves through a pre-defined software development lifecycle, complete with all the compliance-savvy functions like automated process, workflows, notifications and approvals. During the entire process, release and deployment teams can easily track changes and provide the necessary compliance reports. From the time the build is assembled and imported into the system, the solution tracks, manages, and deploys the application through automated workflows and best practices.

The core aspect of the release and deployment automation is the Deployment model. The majority of the platforms enable definition of a deployment model which consists of three distinct sub models.

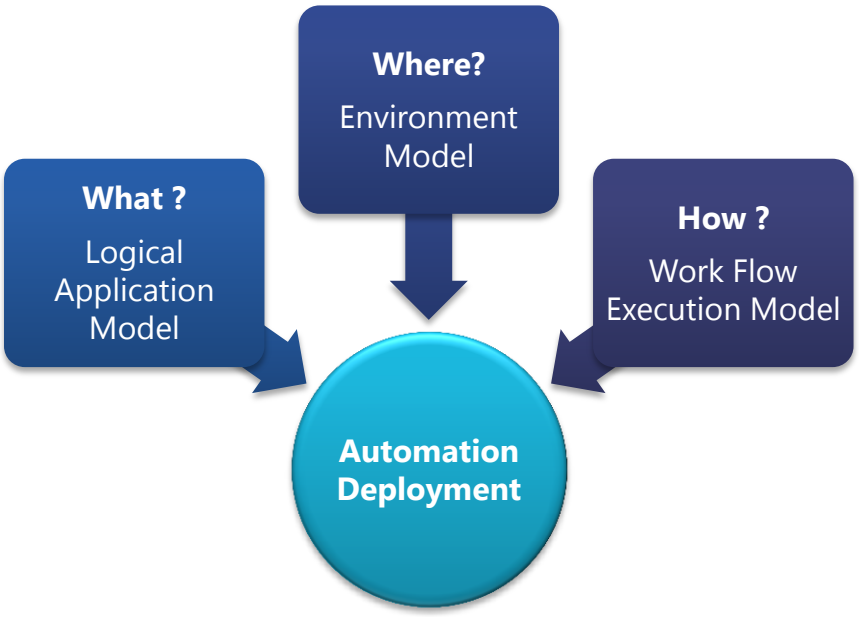

Figure 2 Automation Deployment

The first sub model is the logical application model, which defines “What” needs to be deployed.

The second sub model is the Environment model, which defines “Where” the release component(s) need to be deployed.

The third sub model is the Work flow execution model, which defines “How”, i.e the order in which the release components (s) need to be deployed.

Application Model

The logical application model is not about packaging of physical binaries, it is the logical packaging. The logical model captures what are the different types of components that comprise the release and their mapping to the specific container/server type. This model captures -

- Dependencies on middle ware, database or previous version of the same applications and how they are related to each other.
- Configuration status, i.e. the versions of the application, where the binaries are organized, which versions are sunset and the sequence of patches or upgrades that have to be installed to get to the current version.
- Inventory list of components and the versions of the component(s) that make up a particular release.

Environment Model

Enterprises have different types of environments such as Dev, SIT, UAT, Pre-Prod and Productions. The details of these environments along with their setting such as the number of different servers, interfaces etc. are captured as part of the environment model. These need to be modeled so that it can affect the work flow as it is executed.

Profiles for deployment of components to each container/server type example database server, app server etc. are also captured as part of the environment model.

Workflow Model

As part of the application and environment models, components of particular applications along with their various environments are identified. Most of the enterprises have different release types, for example customization release, support release, non-functional and emergency releases with each release type having its own route to live. The work flow execution model defines the sequence of release activities by each release type.

The work flow can be easily defined using pre-defined standardized business process functions with a drag and drop interface. For each of these steps a threshold for success/failure along with recovery steps for failure conditions can be defined. You have options to set breaks at each of these steps to pause the deployment and examine the logs.

The workflow execution model uses the data in application and environment models to repeat the required activities without the need for complex if-then-else logic. The platform is intelligent to repeat steps based on the number of servers, skip a server update if the release does not impact the server.

To sum it up, this model enables encapsulating the entire set of deployment data and facilitates a repeatable, scalable and secure deployment process.

5 Pre-Requisites of an effective Automation Platform

An effective application release automation platform should be able to:

Version Control Integration

Internal as well as Vendor development teams use different version control tools like Subversion, CVS, and Microsoft Visual SourceSafe. The automation platform should preferably have capabilities to pick up baseline code from any of these version control tools and initiate the build processes. This enables vendor development teams to continue using their preferred version control tools while release automation toolset handles moving the build package through the release lifecycle.

Automation of all movement along the path to production

The automation platform should automate From the time the build is created to final deployment, the promotion/ hand-offs across users, applications and environments should be automated. This operation should be completely configurable using simple point-and –click functions to meet the individual business requirements that the organization has established.

Point and click distribution and deployment

For companies managing multi-platform, distributed environments, the solution should automatically deploy all necessary components to the appropriate target location throughout the development lifecycle. It should gather, package, distribute and install application components at each stage of the lifecycle.

Total Visibility of Objects and Package contents

From the minute a build is created, ability to trace all information associated with a valid package from a revision number to a tag or label, as it moves through the entire development lifecycle. This includes who approved, who touched it, what happened to it, where it went next, etc.

Deployment rollback features

Rollback recovery options are absolutely necessary. When it comes to application development, deployment is often the stage where disaster will strike. Rollback features allow users to simply label the last state of the application pre-deployment before moving into production.

Parallel development and conflict resolutions

The simultaneous development of multiple versions of software application can be complex and difficult to manage. All too often, development teams are faced with conflicts between versions, which can confuse and disrupt even the best managed project teams. The right solution should allow quick identification of version conflicts using descriptive and structured status tags (active, pending, cleared etc) so that developers can clearly see and resolve existing conflicts at the appropriate time.

Secure and consistent distribution packages

The right solution should eliminate the security concerns and inconsistencies of manual file transfers, coordinate arrival regardless of location or system(windows, Linux, IBM i), log all code and content transfers for audit purposes and limit access to objects by user.

Workflow and change management

The solution should automate, accelerate and enforce workflow and change management process.

Software Configuration management

The solution should include functions that automatically build organize and maintain a central inventory of all application components.

Release merging capabilities

The right solution should allow seamless merging of releases without losing version and historical information in the merged release. Companies with scheduled release and tight controls, software houses who want to consolidate releases without losing the object's identifying mark(the version number'), and financial institutions with strict auditing procedures should benefit from the "merge to parent" functionality .

Flexibility

The solution selected should be both open and flexible in nature with integration efforts, tool options and adoption of process methodologies. This is essential in enforcing process across release management. It should plug in to variety of tools so users can pursuer best-of-breed strategy.

Simple Version Control Integration

Release management process also need to support simple version control tools like Subversion, CVS, Microsoft Visual SourceSafe and Microsoft team foundation. Most of them are great for keeping track of all the various version of file, they work fast and developers love them. The solution allows software teams to keep using their own version control tools while it handles moving the build package through a pre-defined software development lifecycle.

6 Benefits

The Benefits enterprises achieve by automation release and deployment are:

- Shorten application release and service times, for both routine and non-routine tasks across the application portfolio.
- Provide centralized control and automatic execution of application release tasks such as rollouts, patches, hot fixes and rollbacks.
- Streamline and coordinate processes across users, applications and environments (Dev, QA, Ops).
- Support automation across heterogeneous infrastructures, including physical, virtual and cloud environments.
- Scale application service workload capacities.
- Provide granular audits and application service reports.
- Provide a sophisticated and comprehensive dashboard of release trends, enabling high-level IT managers to monitor and audit the deployment process.
- Enable seamless integration with existing automation and monitoring tools.

The table below is an example of before and after automation status of release and deployment issues faced by an enterprise.

Issue	Before Automation	After Automation	Business Impact
Effort spent per release on release management and deployment	~10 days	~2 days	80%
Errors encountered from Build to Release/Deploy	10+ errors/cycle	~0 errors/cycle	90+%
Audit cycle time for application changes(who, what, how, why, when)	Days	Minutes	90%
Time to trouble shoot problems	Days	Minutes	90%

Table 1 Before and after Automation

7 Conclusion

Release & Deployment Automation Platforms help organizations leverage automation to release changes more often, release more changes each time, reduce the time required to deploy applications, and as a result free up senior technical personnel to drive business innovation, instead of spending so much time monitoring application deployments.