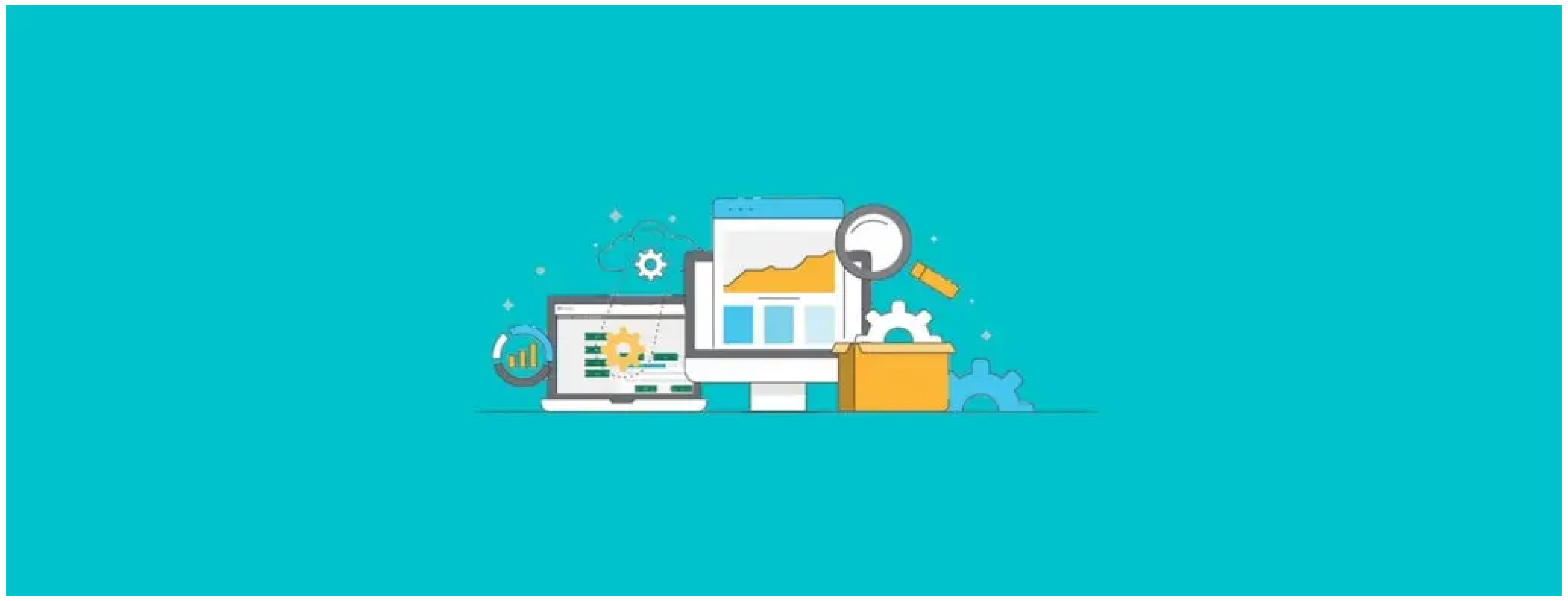
D — DEVOPS

Release Management Explained: Dev To Prod Deployment Process

by **Nivedha Varadharajan** · April 29, 2020



If you are into DevOps it is very important to understand release management. There is no single template for release management. It can differ from organization to organization.

In this article, I will explain what release management is and one of the processes we adopted for our implementations.

What is Release Management?

Release management is a structured model. Planning, scheduling, and controlling the project in each stage or environment with testing and deployments.

It makes application deployments as a stable and smooth one. It avoids half-baked releases by constantly following a process and approval mechanism.

Release management is required for the new project or even changes to an existing project. It prevents deployment delays and last-minute issues as it goes by a well-defined process.

With paradigms like DevOps, we can achieve a smooth transition of application from one state to another with collaboration, automation, and well-defined feedback loops.

Release Process Components

1. **Release Model:** A specific release process from project planning to delivery.
2. **Release Policy:** The definition of release standards, types and requirements to set the release process
3. **Release Template:** A single and repeatable workflow to release process that includes human and automated activities. In other words, a release management checklist.
4. **Release Plan:** Activities involved deploying a release to production
5. **Major Release:** Include many release units that have a higher and critical business impact
6. **Minor Release:** Include fewer release units that do not include critical components

Release Models

Design the release model based on the customer's requirements.

For example, In our previous releases, process checkpoints, validation, and verification were not seamless.

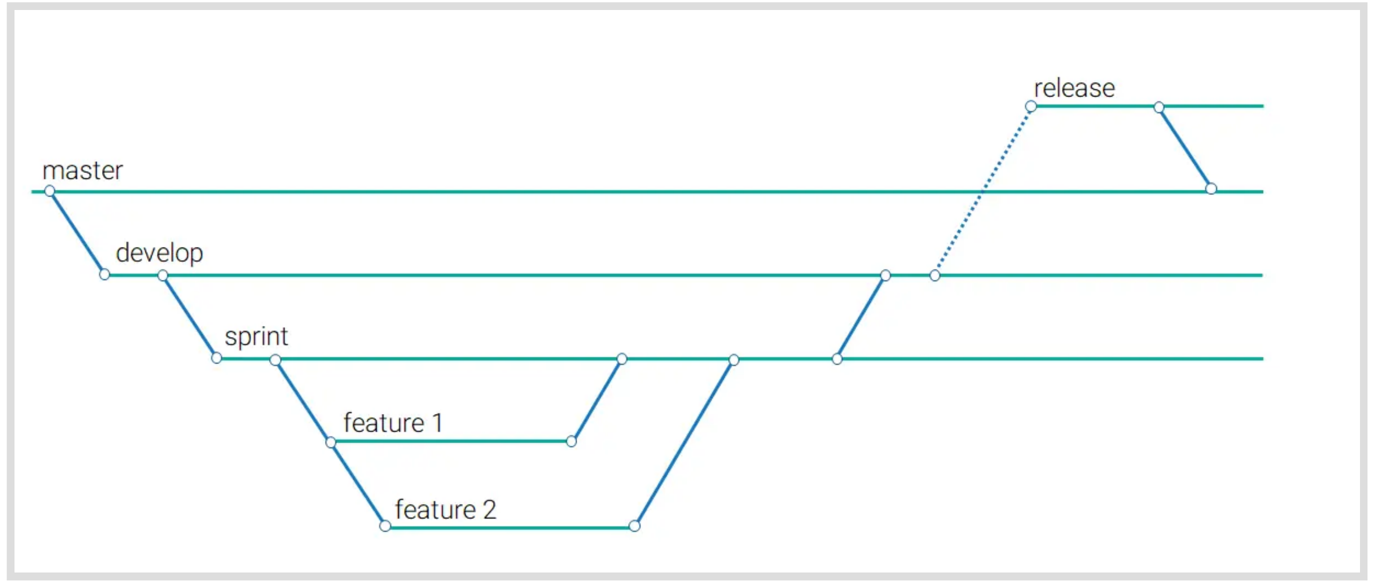
We customized the release model to overcome the issues. Below, given the comparison table for the previous vs current release process

Previous Model	**Current Model**
Branching Strategy: Developers working on the	**Branching Strategy**: Developers working on their feature branch Checkpoint

Review)Automated jobs are trigger for each commit	are trigger based on the pull request (Create, Update and Merge)CI build status in Bitbucket
CI Test Classes are optional. Lagging code coverage. There is no custom script for deployment. Maintaining the Build file manually	**CI** Test Classes are mandatory (Test classes in runtests.txt) Better Code Coverage A custom script for deployment (manifest.txt to generate package.xml) Generate the build file automatically

Release Branching Strategy

In version control systems, Branching is a technique that make a copy of source code.

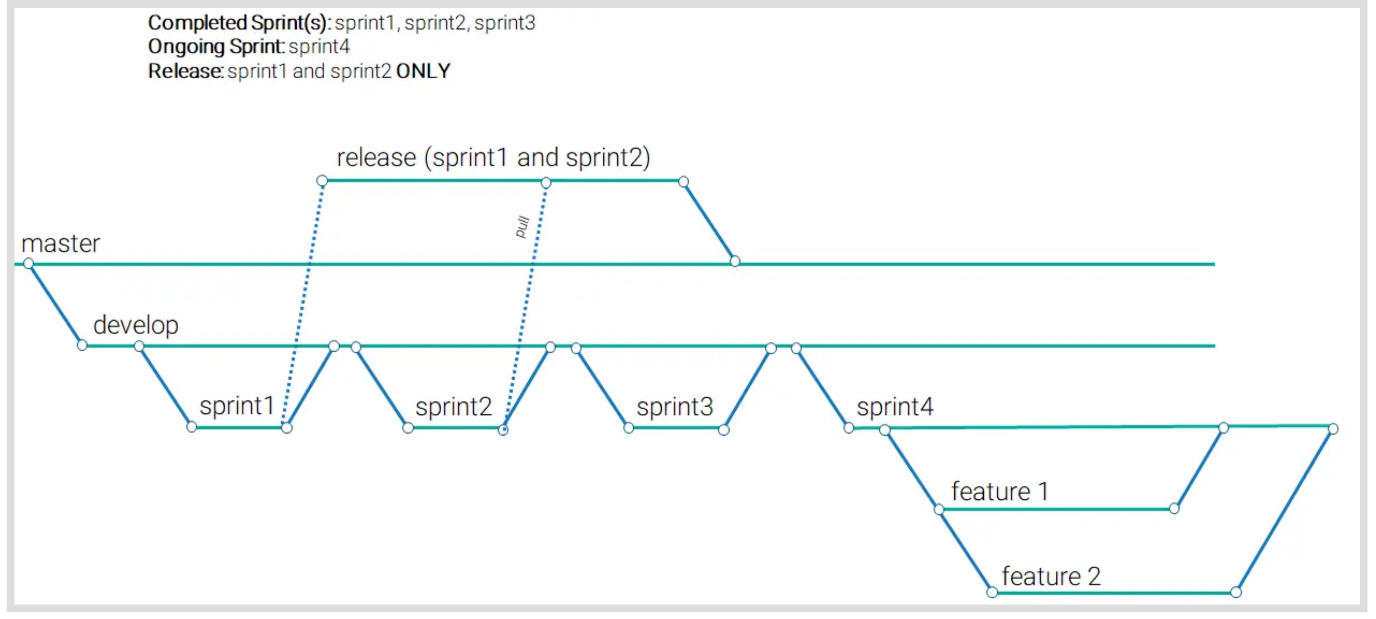


In this Release model master contains the production copy. Create a `develop` copy from the master. Develop branch contains all the developed components.

In an Agile model, the stories are aligned as a sprint. So, create a sprint copy (feature branch) from the develop branch. Based on the stories the developer needs to create their own user story branch from sprint copy.

Once the developers complete their stories, they need to create a pull request from their story branch to sprint copy. The reviewer takes the responsibility to review and merge the code.

The develop branch contains all the completed sprint copies. Once we plan for a higher environment release need to create a release branch from the develop branch. Based on the project release we can also create a release copy from sprint itself.

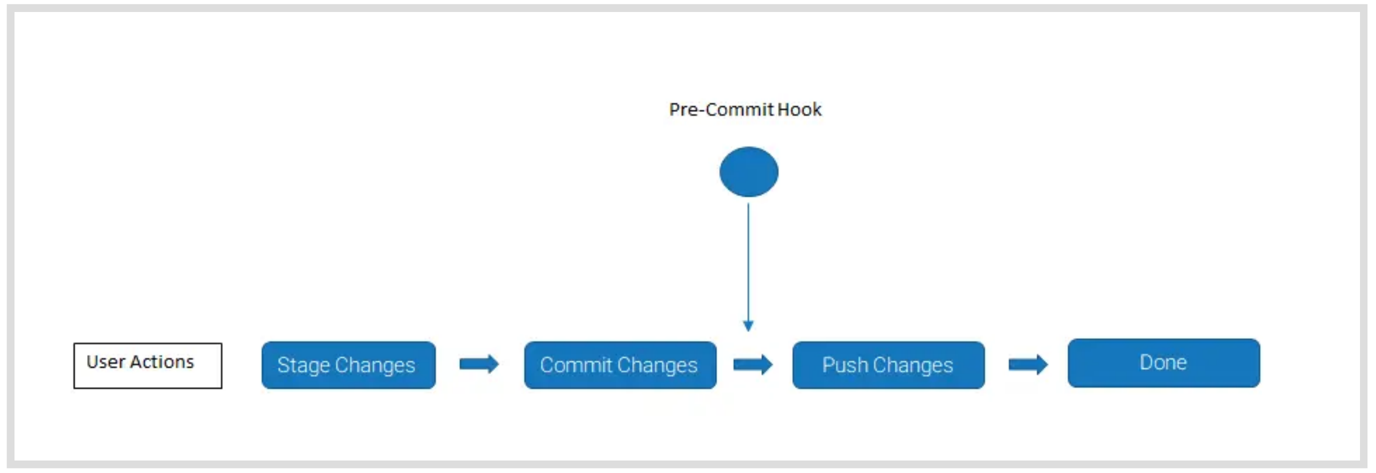

Completed Sprint(s): sprint1, sprint2, sprint3
Ongoing Sprint: sprint4
Release: sprint1 and sprint2 ONLY
release (sprint1 and sprint2)
master
develop
sprint1
sprint2
sprint3
sprint4
feature 1
feature 2

Importance of Branching Strategy

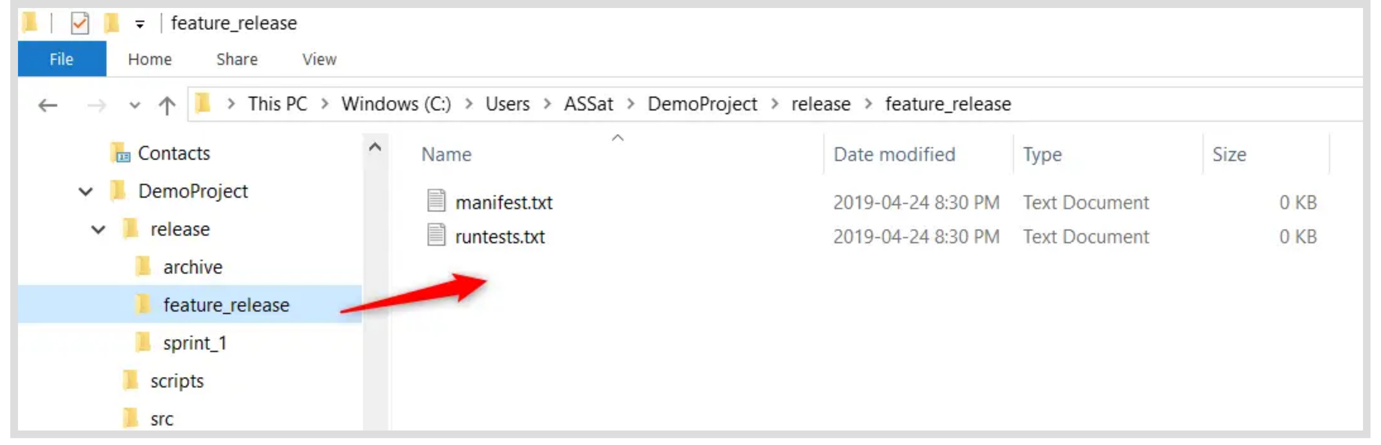
- It provides better code management. We can track the changes by time, person, and versions.
- Developers can work on their stories independently by creating a separate story branch.
- Work between current and existing tracks easily because maintaining the code by versioning.
- collaborate better and spend less time managing version control and more time developing code.
- Merging will happen only via a pull request that gives a checkpoint to review the code.
- Proper branching will provide a smooth development and fast release.

Git-Hook

Once the developer cloned their branch into local, they need to install the git-hook(pre-commit) script. It is used to track the current committed components in a text file called manifest.txt. Also, it generated the runtests.txt file. In that developers need to maintain the related test classes to get the successful deployment.



Folder Structure:



Name	Date modified	Type	Size
manifest.txt	2019-04-24 8:30 PM	Text Document	0 KB
runtests.txt	2019-04-24 8:30 PM	Text Document	0 KB

Manifest:

- Automatic change log
- Key file for any build jobs, used to generate the package.xml
- Can be modified manually if required

Runtest:

- Developer must list down the test class
- Key file for any build jobs, used to generate the ant build.xml file
- Need at least one test to run the build

Release Management Process Overview

In this section, we will look at a release management example from source code to release phase.

Source Code Management (SCM):

Source control is the practice of tracking and managing changes to code. It provides a history of code development. Each version will be shown with the time and includes the person's name when made the change.

Development phase:

The developer completes their ticket and creates a pull request from the user story branch to the sprint branch. Once they create a pull request, Jenkins Dev-Int validate job will get trigger automatically.

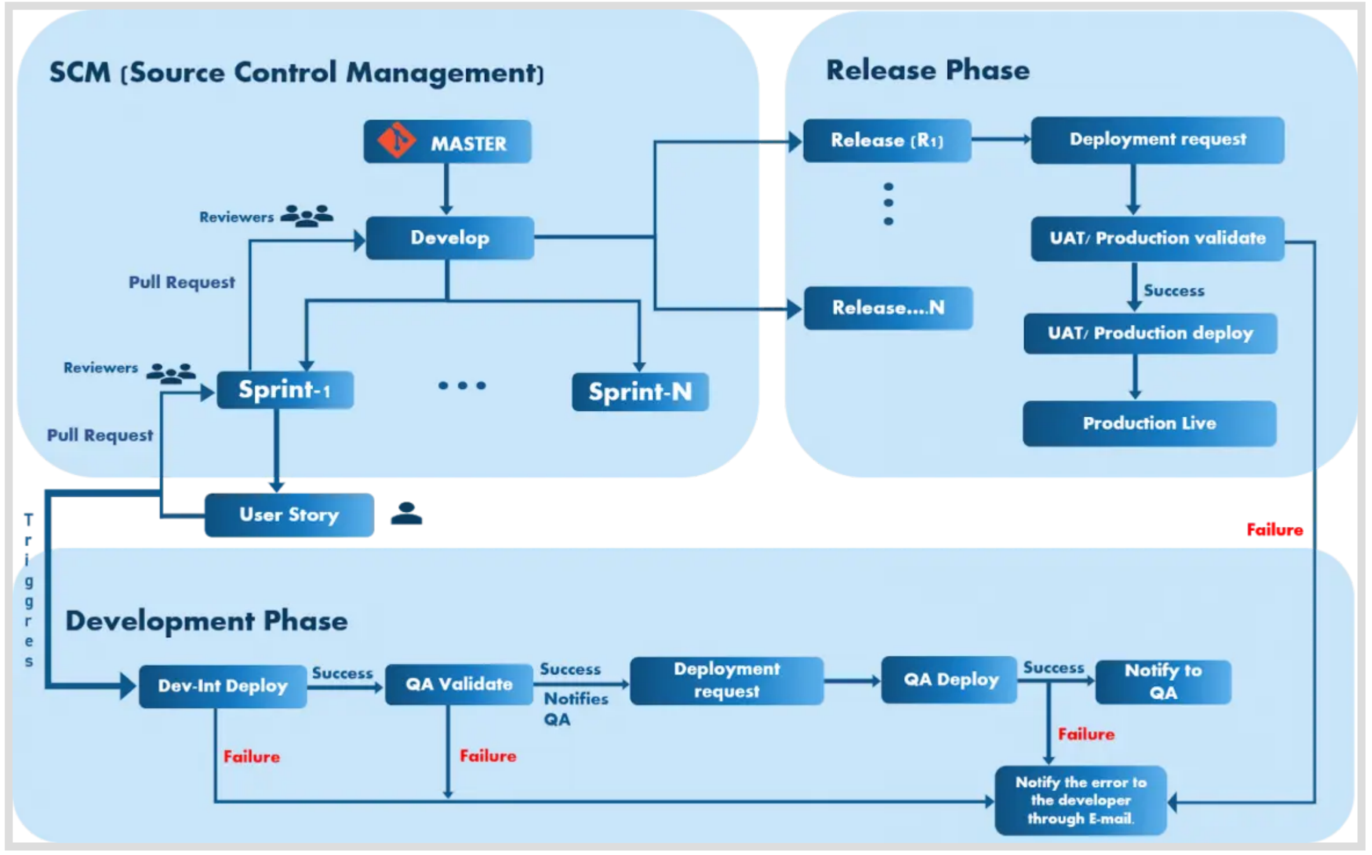
Dev-Int: Dev Integration sandbox that is used for unit testing

Reviewers will review the code and merges the pull request only when Dev-Int validate job gets success. Based on the job status the reviewer started to review the code and merge the user story copy into sprint copy.

At the time Dev-Int deploy job will get a trigger and the components present in the sprint branch will get deployed to Dev-Int sandbox. QA validate job will get triggered once Dev-Int is successfully deployed.

After the QA validation gets success, the QA team requests the DevOps team to deploy the sprint branch components to the QA sandbox. DevOps team will deploy and notify QA team about the success or failures. (This notification process can be automated)

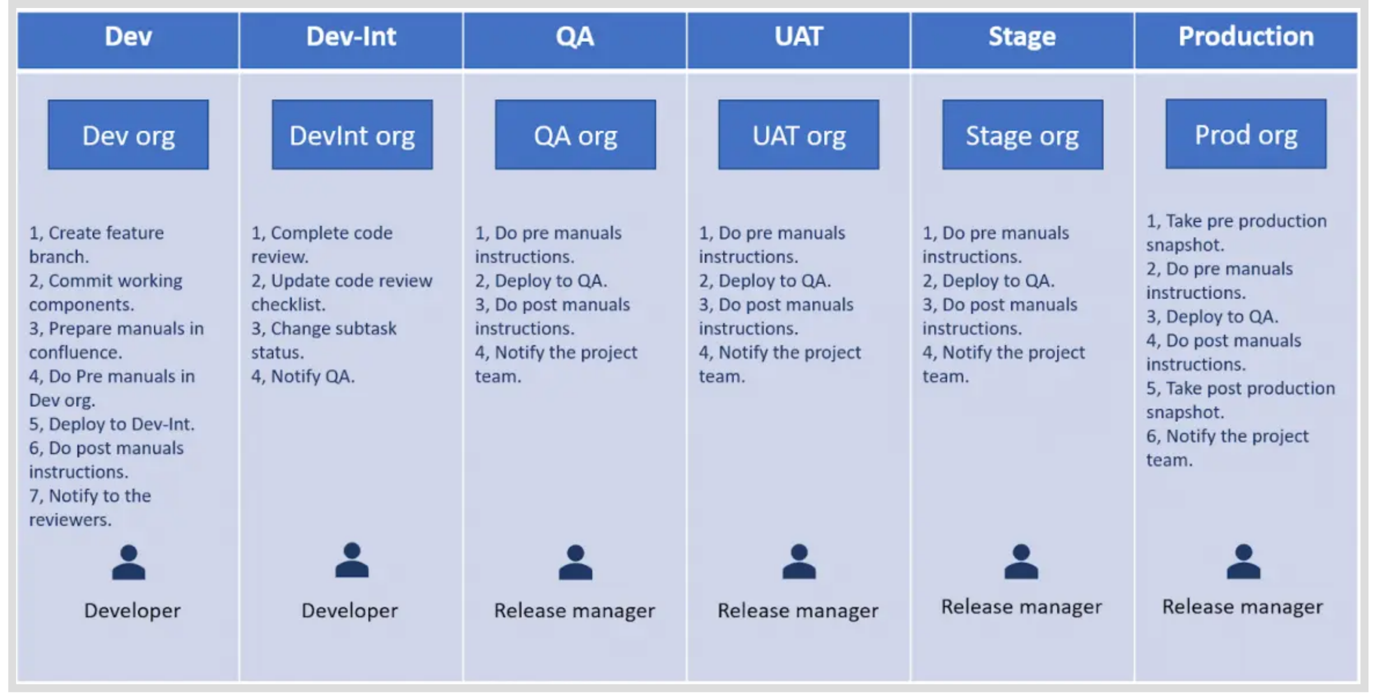
After completion of each sprint that will be merged with the develop branch via pull requests.



Release phase:

The successful completion of sprints will be merged into the develop branch. A release branch will be created from the develop branch when we need higher environment deployments.

Roles and Responsibilities:



Dev	Dev-Int	QA	UAT	Stage	Production
Dev org	DevInt org	QA org	UAT org	Stage org	Prod org
1, Create feature branch. 2, Commit working components. 3, Prepare manuals in confluence. 4, Do Pre manuals in Dev org. 5, Deploy to Dev-Int. 6, Do post manuals instructions. 7, Notify to the reviewers.	1, Complete code review. 2, Update code review checklist. 3, Change subtask status. 4, Notify QA.	1, Do pre manuals instructions. 2, Deploy to QA. 3, Do post manuals instructions. 4, Notify the project team.	1, Do pre manuals instructions. 2, Deploy to QA. 3, Do post manuals instructions. 4, Notify the project team.	1, Do pre manuals instructions. 2, Deploy to QA. 3, Do post manuals instructions. 4, Notify the project team.	1, Take pre production snapshot. 2, Do pre manuals instructions. 3, Deploy to QA. 4, Do post manuals instructions. 5, Take post production snapshot. 6, Notify the project team.
Developer	Developer	Release manager	Release manager	Release manager	Release manager

Benefits

- Track the changes over source code using SCM.
- Help to find and fix the defects fast during the deployments.
- Test classes are mandatory to deploy the component from the lower environment onwards. So, developers can find test failures in the early stages.
- Increase the code coverage.
- DevOps takes the responsibility to do the manuals. Developers need to document the manuals in teamwork or confluence. So, it will reduce the chances of missing the pre/post manuals.
- Increase productivity.
- In CI server the DevInt deployment and QA validation are automated ones. It will reduce the overall deployment time.

Risks (If missed to follow)

- No tracking on changes
- Overwriting each other's changes
- Risk of missing manual changes
- Less productivity
- More defect
- More rework

Release Management Tools

However, some oraganizations will make use of their exitisng tools and open source tools for their release process.

Here are some popular tools in the release management segment.

1. Spinnaker [Opensource Continuous Delivery Tool]
2. Go CD [Open Source]
3. XL release [Enterprise]
4. Harness [Enterprise – For Container-Based application release]

Conclusion

Release management presents a tremendous opportunity to drive continuous improvement in software delivery throughout an enterprise.

It provides greater visibility and traceability throughout the release. This release process gives the efficiency of release planning, review the process to accelerate and improve software delivery.

TAGS: Release Management

1 SHARES: SHARE 1 TWEET

Nivedha Varadharajan

Nivedha Varadharajan is a DevOps Engineer living in Bangalore. She has experience in constructing the release process and managing the deployments for salesforce. Also, she has hands-on experience in the cloud (AWS, Azure). Nivedha has good knowledge to improve the release process based on customer needs. You can connect her via LinkedIn

VIEW COMMENTS (3)

YOU MAY ALSO LIKE

C — COMMON

Sitting On the Fence About DevOps Integration? Don't Miss Your Chance to Jump on the Bandwagon industry giants such as Netflix, Etsy and…

D — DEVOPS

What is DevOps? What Does it Really Mean?

by **Bibin Wilson** · April 8, 2020

This article explains what DevOps really means and how automation tools can help teams in adopting DevOps practice.…

D — DEVOPS

How To Create Jenkins Shared Library For Pipelines

by **devopscube** · July 7, 2019

In this tutorial, you will learn how to create a basic shared library and integrate it with Jenkins…

D — DEVOPS

Git Basics Every Developer and Administrator Should Know

by **devopscube** · October 20, 2019

Version control systems are repositories used to version your code/scripts collaboratively with the advantages of tracking changes as…

D — DEVOPS

How To Setup Google Provider and Backend For Terraform

by **devopscube** · November 3, 2018

D — DEVOPS

List of the best programming languages to learn in 2017

by **devopscube** · December 2, 2016

Programming Languages… AH! There are so many!! And it is really confusing to figure out which language you…