Release Engineering

A guideline for successful software release

Jayashree Nagaraja and Vadiraj Thayur
Jayashree_Nagaraja@McAfee.com , Vadiraj_Thayur@McAfee.com

Abstract

Releasing software to the world is as important as developing the software. All the effort put into developing the software could go in vain if the right process is not followed for software release. This whitepaper is an effort to highlight some good practices that could be followed to ensure software in the form of tools, executable or products are delivered in an efficient manner to reduce the probability of mistakes to minimum. This whitepaper also covers releasing software in the SaaS (Software-as-a-Service, which is termed as Security-as-a-Service within McAfee) world and also touches upon other commonly used means of releasing software.

Release throttling is another aspect covered in the whitepaper. This is the process of controlling the availability of newly released software to the world so that any initial issues are found and fixed with minimal impact to the customer base. This aims at preventing issues from flowing to the greater customer base thus increasing customer satisfaction and also reducing the effort in fixing the issues out in the wild.

Biography

Vadiraj Thayur is a Sr.Technical QA Lead at McAfee, currently working in the McAfee India Center in Bangalore. He has been working for the past 8+ years in different QA roles on Enterprise as well as SaaS products. He has also owned Release Engineering for the McAfee SaaS product for a couple of years.

Vadiraj is a Bachelor of Engineering in Information Science from VTU, Karnataka, India. He also holds an M.S. in Quality Management from BITS Pilani, India.

Jayashree Nagaraja is a QA Engineer at McAfee, currently working in the McAfee India Center in Bangalore. She has an overall Software QA experience of 4+ years. She has been working for the past 2+ years in a QA Engineer role on McAfee SaaS product. She currently owns the Release Engineering for that product as well. Prior to McAfee, she has worked for 2+ years in Mindtree Ltd, Bangalore in a QA Engineer role.

Jayashree has a Bachelor's degree in Computer Science from Bangalore University in India.

1 Introduction

Release Engineering is a very critical and important part, like any other stage of the PLC (Product Life Cycle). The efforts put in by everyone, right from conceptualization to development to testing could go in vain with one mistake in releasing the builds. The criticality of this task doubles if the process involves release to the live servers (on-premise or OEM or SaaS or to manufacturing unit). For instance, in McAfee SaaS (Security-as-a-Service), any released build goes live within a couple of hours and could typically reach millions of users within a day; a case where a lot of caution is necessary while releasing. The most important thing to be ensured by the Release Engineer is “Release the right files, Release the files right”. In simple words, it means that the engineer needs to ascertain that the files being released are the correct ones.

Another key area of Release Engineering is to control the release process in order to provide the newer version/patch to different sets of customers one after the other. This method helps the engineering team to monitor the release process and the status (success or failure) of the release for different customers.

This write up is an effort to highlight the guidelines/best practices to be followed for a successful software release.

2 Need for a ‘Release Engineering’ process

Build release process is generally considered as a simple task where builds are copied from the build room to various live servers with the help of ‘file copy jobs/scripts’. Of course it is a build copy task, but not as simple as it is misunderstood. Release Engineering is a process in itself which has well defined stages before the build goes live and reach the customers. It consists of simple, yet very important steps which are to be followed religiously to ensure a less error prone release.

Any new software which goes live (On-premise/Saas) or released to manufacturing unit or to OEM partner could get into two kinds of issues. One being errors/faults in the new software and the other being wrong files and configuration reaching the customers. Both the issues can cause enough damage to the customers in terms of data loss, unstable environment, loss of productivity/time/money etc, which in turn causes loss of business to the company which is releasing the software, damage its reputation, affect customer satisfaction and result in competitor advantage. The release process which we follow should be capable of preventing such situations in the first place and cater quick solutions to both kinds of problems in order to avoid the spread of damage. The simplest solution to both the problems is to rollback the changes and switch to the stable version of the software. The process we follow should ensure least turnaround time in such critical situations. However, in case of release to manufacturing unit or to OEM partner, if a post production issue occurs, it is not feasible to switch back to the older version of the product, as the release reaches the customers through the hardware, in the form of CD/DVD kit or pre-installed on the PC along with the Operating System. In such situations where the spread of issues cannot be controlled, preventing the issues by following a systematic release process would be a better choice.

There should be a lot of thought process involved in designing a release process which is methodical to prevent errors, which foresees the occurrences of post production issues, which has the ability to incrementally provide the new software/version to customers and has a backup plan to halt the availability of erroneous files without causing much damage. Hence, any software release calls for a better Release Management System and it is worth the effort!

3 Release Engineering Process

The Release Engineering process can be split into 3 stages: Pre-release process, Release process and the Post release process.

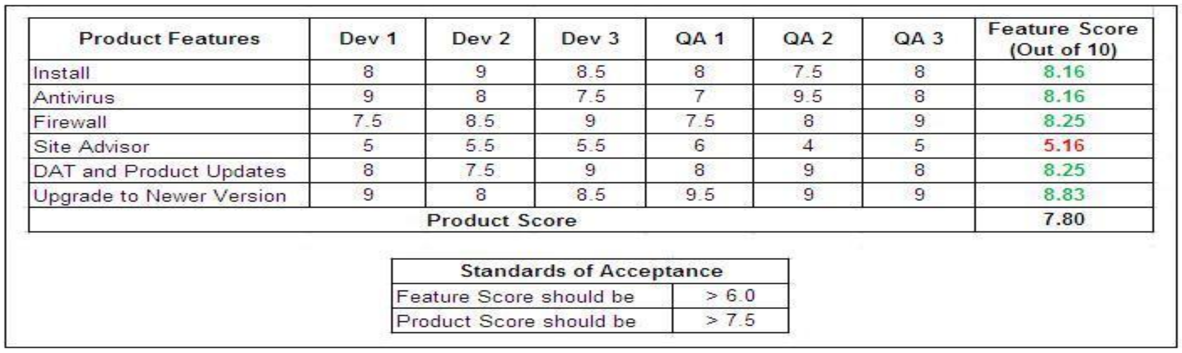
3.1 Pre-Release Process

3.1.1 Team Approval: Once the build has passed QA, it is necessary to get a "GO" for the release from all the members of the team including the Engineers, Managers, Product managers, Program managers and all others concerned. It is also a process of recording the confidence level on different aspects of the product release from concerned members of the team. For instance,

- Engineers (Dev and QA) express the confidence level on the features and functionalities of the new product,
- Managers acknowledge that all the committed features are implemented and certified,
- Product Managers certify that the new product caters the intended solution to the requirements/problems of the customers (based on the engineering confidence),
- Program Managers, who tracks the release timelines, approve the release dates.

There has to be a logical way of recording confidence level from the Dev and QA, as it forms the basis of consent from other members in the team. Easiest method is to ask each Dev and QA in the team to rate different features of the product on a scale of 10, based on the development, test and regression effort put in so far; defects logged, fixed and deferred; performance of the feature etc. The individual rating can then be aggregated to come up with the Feature Score for each of the features, which can again be averaged to arrive at the Product Score, which speaks the confidence level of the team. As the ratings of different features are aggregated, a poor rating for one of the features could be subsided by a good rating for the others. So it is important to set a standard of acceptance or a cut off number for individual feature rating as well as the Feature Score and the Product Score, below which an alarm has to be raised or a strong reasoning has to be provided to accept the dip in rating.

Pictorial Representation of Recording Confidence Level of Engineering Team



Product Features	Dev 1	Dev 2	Dev 3	QA 1	QA 2	QA 3	Feature Score (Out of 10)
Install	8	9	8.5	8	7.5	8	8.16
Antivirus	9	8	7.5	7	9.5	8	8.16
Firewall	7.5	8.5	9	7.5	8	9	8.25
Site Advisor	5	5.5	5.5	6	4	5	5.16
DAT and Product Updates	8	7.5	9	8	9	8	8.25
Upgrade to Newer Version	9	8	8.5	9.5	9	9	8.83
Product Score							7.80

Standards of Acceptance	
Feature Score should be	> 6.0
Product Score should be	> 7.5

This is just an illustration. Numbers and features are just indicative

It is important to note that, if there is a "NO GO" from any of the team members for any reason, it is necessary to find out the criticality of the issue being stated in order to decide if the product should be released now or postponed. Hence any release should be approved by the entire team before it reaches the customers.

3.1.2 Informing the stakeholders: Once the build has passed QA, all the stakeholders (both internal and external) including the Marketing Team, Sales Team, Support Team, Manufacturing Unit or OEM Partner, Infrastructure/Data Center Team, Partners and Customers should be intimated on the new product version and the details of the release. It is a necessary step to initiate the action items related to the release, on the part of the stakeholders (if any). For instance,

- ➢ The Marketing and Sales Team should be aware of the new features and functionalities of the product to market/sell the right solutions to the customers,
- ➢ The Support Team should also be trained on the product features to guide the customers on product usage. It is also important to publish the Product Guide, List of Known Issues, Troubleshooting Tips and the Supportability Document to effectively solve the customer issues,
- ➢ The Manufacturing Unit or the OEM Partner should be intimated for hardware readiness in order to reach the customers on time,
- ➢ The Infrastructure Team or the Data Center Team should be informed on the required configuration changes (Ex: Web Server setup) on the live servers to support the new product/version,
- ➢ Lastly, Partners and Customers should be intimated on the release, especially in cases where administrator/user interaction is expected for successful install or upgrade.

3.2 Release Process

The release process discussed here consists of various 'Best Practices' which can be adopted for any kind of software release. It is important to ensure that these steps are followed sequentially and methodically (as applicable).

3.2.1 AV Scanning: One very essential process that needs to be performed as part of the release process is to scan the release candidate build to ensure that none of the files being released get detected as malicious by any prominent Antivirus software in the market. To accomplish this task, two or three 'Release Servers' which are used for the release process should be installed with at least two different Antivirus software with latest signature updates and the build should be scanned to ensure that there are no infections. This step is of high importance for any software product release, as it ensures that the build does not contain any malicious files that affect the customers and that there are no malware patterns in the build. If any of the files in the build are modified or cleaned or deleted by the Antivirus software, install/upgrade/any other functionality of the product may not function as expected when it reaches the end user.

3.2.2 Build Archiving: Good archiving is another secret to successful release. It is necessary that the current files and configuration on the live servers are backed up on the Release Servers before starting the release process. All previously released versions of the product (at least the versions which are still being supported) should be archived and stored in a build repository. Release logs and hash/comparison results can also be archived. These will prove very useful in situations requiring a rollback and also to trace back post release issues. They are also useful in situations when a patch has to be released on a supported older version of the software, as it provides a quick reference to the base build on which the files can be patched and tested on the QA environment before releasing. Hence it saves a lot of time and confusion while dealing with older releases.

3.2.3 Release to Staging Environment: Releasing the product to Beta servers to get feedback on the new product/version is a common practice in any software release process. However, the environment and the configuration might be entirely different on beta and live servers, which may lead to undiscovered configuration issues on production environment. To overcome this problem, it is a good practice to stage the final build on an environment which is equivalent to live servers and validate through a couple of basic tests to gain confidence on the files and configuration which is set to go live. The risk of post production issues due to configuration errors can be largely reduced with this step.

3.2.4 Release to Live Servers: A software release can be of various types including release to an On-Premise setup, to SaaS environment (Cloud), Release to Manufacturing Unit and Release to OEM Partner. For any kind of release, there has to be a systematic way of execution of this step which is designed with utmost caution. Considering that all the above mentioned steps are followed before reaching this stage, a generalized method which can be adopted by all different kinds of release is discussed here.

- **Release Sandbox:** Make a copy of only the build (or files) which is being released as part of the current release on the Release Server.

- **Release Candidate:** Make a copy of current live build (from the archived build repository), patch the new files onto it and make the required configuration changes (if any) to get the Release Candidate version ready.

- **Build Comparison:** Compare the current live build (as in the archived build repository) with the Release Candidate build to check for any major discrepancies. The difference set should be exactly same as the files in the release sandbox.

- **Copy files to live servers:** The build which is ready to be released should be copied from the Release Servers on to the Live Servers with the help of file copy scripts. The scripts that are used for copying the files are very critical. They should be tested a couple of times against dummy builds and servers to gain confidence that they achieve the correct results. As the files are copied from server to server during the release process, the copy needs to be verified at each and every step. This can be done by file/build comparison. One very common way of comparing binaries is by hashing. The files in both the locations are hashed separately and the hashes are compared. Typically, the hashes should match exactly otherwise, strong reasoning would be required to explain why the difference in hashes is acceptable. Many hashing tools are available which could be used. Alternately, some languages like PERL also provide hash modules. Sometimes, when a couple of files are being released to production, to achieve a quick turnaround, file comparison tools like Beyond Compare can be used as well. They can be used to perform file/folder level and binary comparisons to ensure the correctness of files and configuration.

3.2.5 Post Production Validation: Release process is not complete until the release has been verified with a couple of basic validations from the live servers. This step may simply involve a download of a file, a basic installation of the product or a simple upgrade from previous version to the newer version, to ensure that the release has achieved the intended results.

3.3 Post Release Process

The post release process is the last and equally important stage of software release. It consists of two major steps, one being Phased Rollout of the software and the other being Post Production Validation.

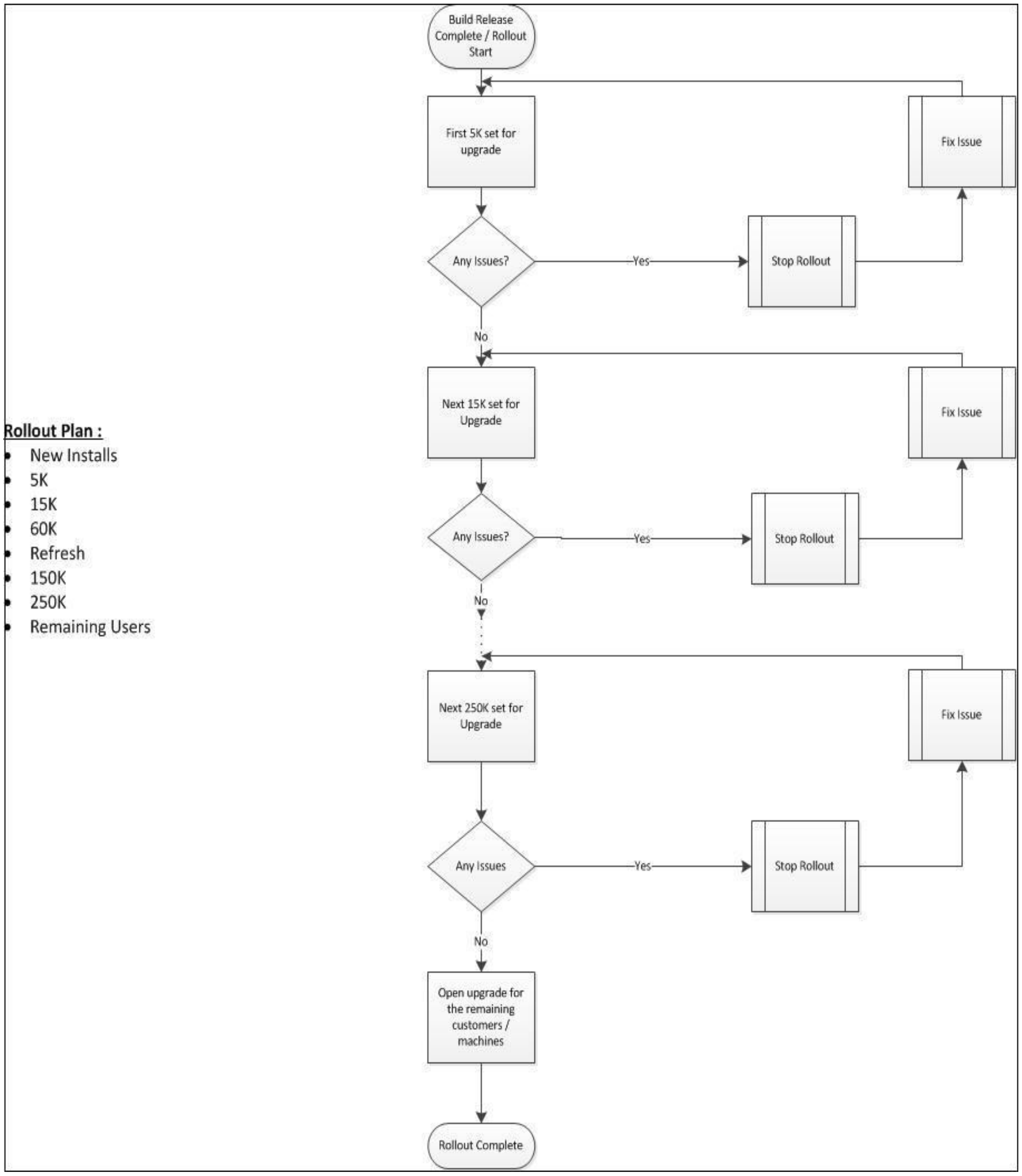
3.3.1 Phased Rollout: Once the build is successfully released, making the software available to all customers at once might be risky. If in case there are issues with the new release, all the customers will end up running into the same problem at once, which might require a bigger effort to deliver rectification. It is important to control and manage the availability of the new product version to the customers. The advisable method is to incrementally upgrade smaller sets of customers by rolling out the product in a phased manner. The importance of Phase Rollout is felt mostly by the 'SaaS based product teams', as any change which is released on the cloud would be available to all the customers immediately unless controlled. The other forms of release (like CD, OEM, Uploading onto a website etc.) would also benefit from the phased rollout process. The Phase Rollout mechanism is also helpful in cases where a specific hot fix has to be provided to only a few set of customers in a scenario where thousands of other unaffected customers are also pointing to the same live server on the production setup. The Phase Rollout of software is a very handy process which could be explained as below:

- **Identify the phases:** The customer base should be broken up into smaller segments to which the product should be released in a sequential manner. The breakup can be done based on various criteria like: Language, node / license count, complexity of their environment, products they use, Location etc. For instance, phased rollout for a software can be done based on languages, the English customers could be given the software first, then to German and French and then to Japanese and Chinese and so on. If the criterion is just based on numbers, the customers can be chosen for upgrade as, 5k customers at once, then 25k customers and then 100k customers etc.

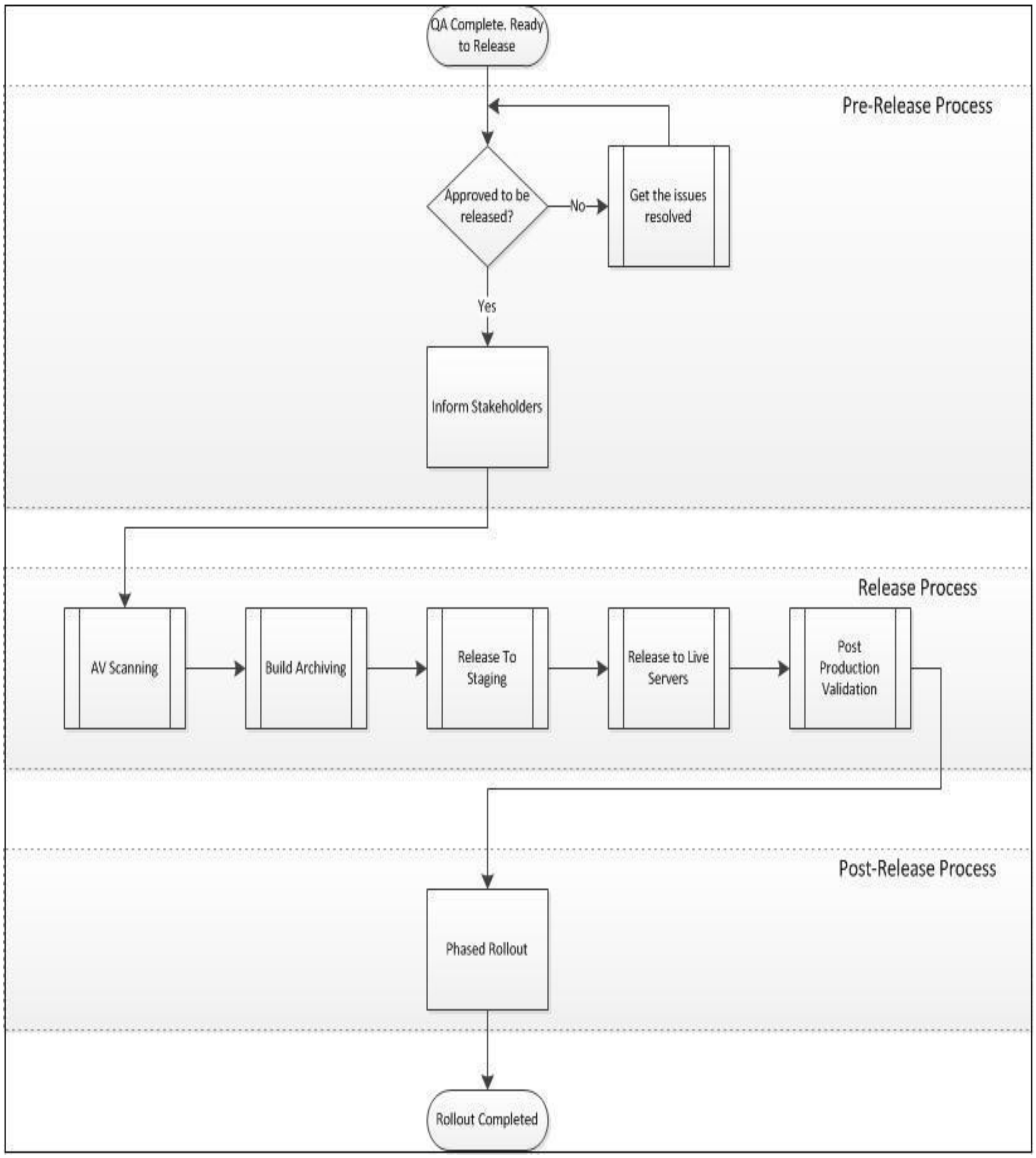
- **Rollout stage-by-stage:** Once the different phases are identified, the software should be rolled out in that manner. It is good to keep customers informed if they are being included in any phase of rollout so that they get prepared to move to the new version. In order to incrementally upgrade the customers, a backend (database) which is designed to support the rollout is a good thing to have. In this way, the customers to be included in phases can be selected in the database. Before beginning the phased rollout, it **i**s a good practice to test it in the QA environment to validate the correctness.

- **Review and continue the release:** After the first segment of customers receive the product updates, review the status of the release in terms of feedback from the customers. If there are any issues reported, they need to be fixed and verified on the customer environment before proceeding with the next phase. If there are no issues observed, the next set of customers should be upgraded and so on. With couple of such cycles, if there is enough confidence gain on the new product, software can be opened up for upgrade/download to all the remaining customers.

Pictorial Representation of Phased Rollout Process



4 Work Flow Diagram of Release Engineering Process



5 Role of a Release Engineer

The secret of Release Engineering is to 'Release correct files and configuration onto correct servers'. This is possible only if the release engineer is always up to date on the changes in the product, current requirements/specifications/configurations for the release and the production setup. It is also the responsibility of the release engineer to co-ordinate between the teams which owns the different aspects of the product, to ensure that all the related components are available on time and the integration functions as expected. For instance, to adopt Phased Rollout for a release, the DBA should be informed well before and the scripts required to split the customer base and rollout the product should be ready. Release Engineer play a vital role in discovering issues related to files and configuration on the production setup. Hence, a release engineer should be keen on finding the right answers for any small discrepancy observed in any stage of the release process.

In all, a good Release Engineer requires the following:

- ➢ Good knowledge of the product,
- ➢ Good knowledge of the production servers,
- ➢ Good knowledge of the release process,
- ➢ Hands-on experience on the tools used,
- ➢ Lot of emphasis to minor details,
- ➢ Patience,
- ➢ A positive questioning attitude.

6 Conclusion

Any software release demands a well-thought Release Engineering Process, to prevent the problems rather than finding a solution after the damage is caused. Here are a few advantages of following a good Release Engineering Process:

- ➢ Copying files on to intermediate locations on the Release Servers helps the Release Engineer validate the correctness of files at every step (less error prone),
- ➢ Release to staging environment provides an opportunity for pre-production validation,
- ➢ Rollback operation is handy with file and configuration backup,
- ➢ File archiving helps quick future references,
- ➢ Phased rollout helps early discovery of issues and prevents damage on larger customer base,
- ➢ Along with all these processes, a good release engineer is required to adhere to the processes and improve them over time.

If not planned and managed correctly, release process becomes very lengthy, messy, time consuming and risky.

For any kind of software release, the essence of Release Engineering remains the same:
"Deliver the correct files at the correct time in the correct way to the correct location"!!!

7 References

None