DEPLOYING VS RELEASING SOFTWARE: WHAT'S THE DIFFERENCE?



For all the work involved in the world of software development, it is only when a customer uses the software to achieve an outcome that the software's value is realized.

Every day, thousands of updates are made available on app stores, mostly pushing bug fixes, security updates, and performance improvements; sometimes making the latest features available for users to enjoy. App store intelligence firm App Annie reported that in the second quarter of 2020, app downloads reached a high of nearly 35 billion.

In service management, two terms, **deployment** and **release**, are often used interchangeably to describe rollout of these updates. But is there a difference between them? Let's investigate. (For the quick answer, skip ahead to the key difference between deploying and releasing software.)

What is deployment?

Deployment involves moving software from one controlled environment to another. According to the ITIL 4 practice guides,

> *An environment is a subset of IT infrastructure used for a particular purpose.*

The most common environments are:

- **Development.** Commonly referred to as dev, this is where developers build the code.
- **Integration.** Here, the new code is combined and validated that it works with existing code.
- **Test.** This is where both functional and non-functional tests are conducted on the merged

code to confirm it meets organization and customer requirements.

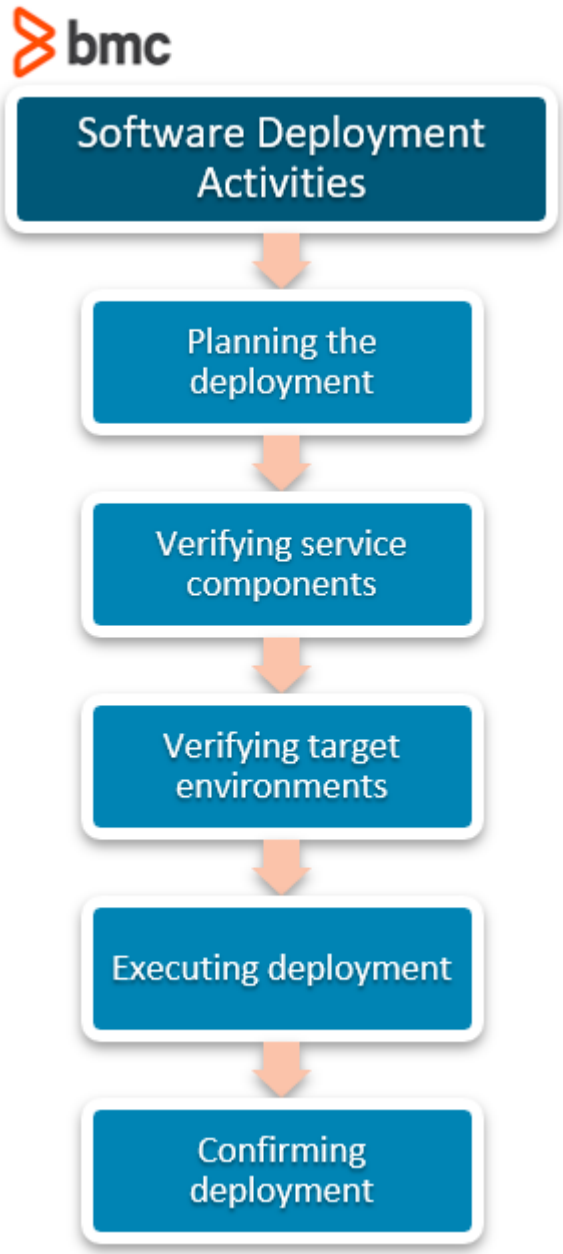
- **Staging.** This environment is used to test the software using real data to validate it is ready for use.
- **Production.** Commonly referred to as prod, this is where the software is made available to users.

Deployment is generally considered the final stage of the software development lifecycle (SDLC):



Deployment activities

The activities involved in deployment management include:



1. **Planning deployment**. Preparation tasks including authorization, alignment of resources, and scheduling.
2. **Verifying service components**. Unit and integration testing, with iterative fixing and retesting.
3. **Verifying target environments.** Validation to ensure that the host environments are ready for accepting the software packages.
4. **Executing deployment.** Pushing the software into the environment and conducting relevant system tests.
5. **Confirming deployment.** Acceptance testing to validate customer requirements are met. Post review and lessons learnt activities are also carried out here.

Deployment techniques

Traditionally, deployments were carried out using what is termed a big-bang approach where all features are released in one go. But currently, due to technology and risk management, rolling or phased deployments are preferred through gradual release across the environment over a period of time.

One common technique used in deployment is the blue-green approach. Here, two identical but separate production environments are used, with the current code running on the 'blue' environment, and the new code being deployed on the 'green' environment. Traffic is then switched from blue to green, and the performance of the new service or features is then monitored. In case of a major hitch, traffic is switched back to blue (like a rollback), allowing for fixing and remedies without affecting customers significantly.

Improving deployment activities

Deployment had been a challenging part of software development through the years, but lately has been made easier thanks to advances in practices such as DevOps. CI/CD (Continuous Integration/Continuous Delivery or Deployment) has resulted in faster deployment of software with fewer errors thanks in part to automation in integration, testing and deployment. Tools are fast becoming mainstream solutions in this area, such as:

- Jenkins
- Chef
- Bamboo
- GitLab

However, many organizations still use manual change approvals to trigger deployment to production environments, typically for two main reasons:

- To mitigate the risk involved in case live services are affected.
- To meet compliance needs.

What is software release?

According to the ISO/IEC 20000 standard definition, a release is:

> *A collection of one or more new or changed services or service components deployed into the live environment as a result of one or more changes.*

In other words, a release makes services and features available to users. More often than not, release management is more of a business responsibility than a technical responsibility. This is because the decisions on scheduling releases can be tied to business strategy from a revenue or portfolio management perspective.

A company can decide to release features based on an agreed marketing plan or stagger the releases to prevent cannibalizing existing products or to counter competitor activity. Features can also be released to different customers based on the company's product offerings, e.g. premium customers getting advanced functionalities.

Release categories

The most general categorization of releases is based on scope:

- The terms **major and minor** are used to describe a release in relation to the significance of change in code, service, and features. For example, a **major release** could see a software moving from version 2.4 to 3.1, while a **minor release** could be from 2.2.1 to 2.2.2.

- **Emergency release** is used to define a software version or package that is availed quickly to address a major issue especially from a security or performance perspective.
- **Maintenance release** for bug fixes and patches.
- **Feature release** for new or changed functionality.

Release techniques

Some release techniques used in software development include:

- Canary Release. This involves releasing new features to only a subset of users. Just like the canary birds are used to test poisonous gases in mines, this user group is the test case to unearth any issues, before the rest of the population is given access.
- Dark Launching. Similar to canary releases, dark launching involves activating features for a subset of users who are not aware of the new functionalities (hence "dark"). This can be done through the use of feature flags, that allow toggling on and off of features based on the needs of the organization.

Deployment vs Release: The key difference

The key distinction between these deployment and release is the business rationale. Deployment doesn't necessarily mean users have access to features, as can clearly be seen by the different environments involved. Some companies will release at the same time as deployment to production is taking place.

Others will choose to wait, thereby having the new features in production but not availed to users until the business decides.

Additional resources

For more on this topic, explore these resources:

- BMC DevOps Blog
- DevOps Guide, with 30+ articles
- Deployment Pipelines (CI/CD) in Software Engineering
- Continuous Delivery vs Deployment vs Integration: What's the Difference?
- Enterprise DevOps Solutions