DEPLOYMENT PIPELINES (CI/CD) IN SOFTWARE ENGINEERING

On any Software Engineering team, a pipeline is a set of automated processes that allow developers and DevOps professionals to reliably and efficiently compile, build, and deploy their code to their production compute platforms.

There is no hard and fast rule stating how a pipeline should look and the tools it must utilise. However, the most common components of a pipeline are:

- Build automation/continuous integration
- Test automation
- Deploy automation

A pipeline generally consists of a set of tools which are normally broken down into the following categories;

- Source Control
- Build tools
- Containerisation
- Configuration Management
- Monitoring

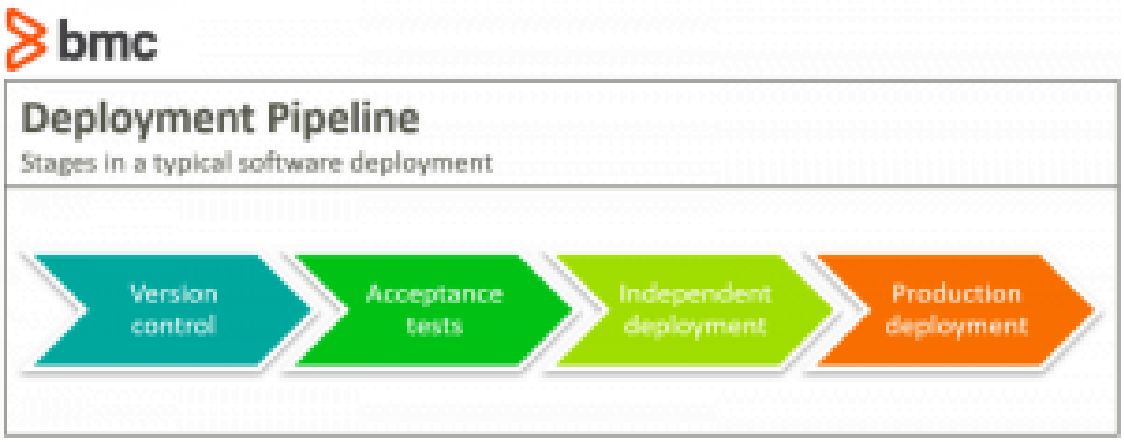
The key objective of a Software Delivery Pipeline is automation with no manual steps or changes required in or between any steps of the pipeline. Human error can and does occur when carrying out these boring and repetitive tasks manually and ultimately does affect the ability to meet

deliverables and potentially SLAs due to botched deployments.

Deployment Pipeline

A Deployment pipeline is the process of taking code from version control and making it readily available to users of your application in an automated fashion. When a team of developers are working on projects or features they need a reliable and efficient way to build, test and deploy their work. Historically, this would have been a manual process involving lots of communication and a lot of human error.

The stages of a typical deployment pipeline are as follows.



Version Control

Software Developers working on their code generally commit their changes into source control (e.g. GitHub). When a commit to source control is made a the first stage of the deployment pipeline is started which triggers:

- Code compilation
- Unit tests
- Code analysis
- Installer creation

If all of these steps complete successfully the executables are assembled into binaries and stored into an artefact repository for later use.

Acceptance Tests

Acceptance testing is a process of running a series of tests over compiled/built code to test against the predefined acceptance criteria set by the business.

Independent Deployment

An independent deployment is the process of deploying the compiled and tested artefacts onto development environments. Development environments should be (ideally) a carbon copy of your production environments or very similar at worst. This allows the software to be functionally tested on production like infrastructure ready for any further automated or manual testing.

Production Deployment

This process is normally handed by the Operations or DevOps team. This should be a very similar process to independent deployments and should deliver the code to live production servers. Typically this process would involve either Blue/Green deployments or canary releases to allow for zero down time deployments and easy version roll backs in the event of unpredicted issues. In situations where there are no zero down time deployment abilities release windows are normally negotiated with the business.

Continuous Integration & Continuous Delivery Pipelines

Continuous Integration (CI) is a practice in which developers check their code into a version controlled repository several times per day. Automated build pipelines are triggered by these check ins which allow for fast and easy to locate error detection.

The key benefits of CI are:

- Smaller changes are easier to integrate into larger code bases.
- Easier for other team members to see what you have been working on
- Bugs in larger pieces of work are identified early making them easier to fix resulting in less debugging work
- Consistent code compile/build testing
- Fewer integration issues allowing rapid code delivery

Continuous Delivery (CD) is the process which allows developers and operations engineers to deliver bug fixes, features and configuration changes into production reliably, quickly and sustainably. Continuous delivery offers the benefit of code delivery pipelines that are routinely carried out that can be performed on demand with confidence.

The benefits of CD are:

- **Lower-risk releases.** Blue/Green deployments and canary releases allow for zero downtime deployments which are not detectable by users and make rolling back to a previous release relatively pain free.
- **Faster bug fixes & feature delivery.** With CI & CD when features or bug fixes are finished, and have passed the acceptance and integration tests, a CD pipeline allows these to be quickly delivered into production.
- **Cost savings.** Continuous Delivery allows teams to work on features and bug fixes in small batches which means user feedback is received much quicker. This allows for changes to be made along the way thus reducing the overall time and cost of a project.

Blue/Green Deployments

Utilisation of a Blue/Green Deployment process reduces risk and down time by creating a mirror copy your production environment naming one Blue and one Green. Only one of the environments is live at any given time serving live production traffic.

During a deployment, software is deployed to the non-live environment - meaning live production traffic is unaffected during the process. Tests are run against this currently non-live environment and once all tests have satisfied the predefined criteria traffic routing is switched to the non-live

environment making it live.

The process is repeated in the next deployment with the original live environment now becoming non-live.

Canary Deployments

Different from Blue/Green deployments, Canary Deployments do not rely on duplicate environments to be running in parallel. Canary Deployments roll out a release to a specific number or percentage of users/servers to allow for live production testing before continuing to roll out the release across all users/servers.

The prime benefit of canary releases is the ability to detect failures early and roll back changes limiting the number of affected users/services in the event of exceptions and failures.

Tools for automating software quality

There are many different tools that you can use to build CI/CD pipelines outlined below, all of which can be used to build reliable and robust CI/CD pipelines with the added bonus of being able to get started for free!

- Jenkins
- Azure DevOps (formerly VSTS)
- CodeShip

In summary, CI is the automated process to enable software development teams to check in and verify the quality and ability to compile of their code. CD allows Development and Operations teams to reliably and efficiently delivery new features and bug fixes to their end uses in an automated fashion.

Related reading

- BMC DevOps Blog
- DevOps Guide, a series of 30+ articles on DevOps
- Continuous Delivery vs Deployment vs Integration: What's the Difference?
- Deploying vs Releasing Software: What's The Difference?
- Testing Automation Explained: Why & How To Automate Testing
- Shift Left Testing: What, Why & How To Shift Left