WHAT IS A CONTAINER PIPELINE?

Containers have revolutionized the virtualization market by segmenting processes, drastically increasing portability, and saving space and time. These lightweight, safe spaces coincide beautifully with the DevOps methodology by bringing cohesion and clarity to all stages of the build, test, and deploy a model of computer coding. Pipeline or “pipelining” brings automation to the forefront by breaking down continuous integration and continuous delivery (CI/CD) into the individual stages of the build. By doing so, Devs can incrementally test, assess, and redeploy computer code without impeding workflow throughout the project.

In a software-centric world moving at breakneck speed, the need for agility and rapid release of application support and updates is at a premium. Pipeline provides time-efficient tools, reducing manual, error-prone deployment work related to any software project.

How Pipelining Works

A pipeline is a workflow strategy enveloping automation to produce software in an effective timely process. First, automated tests are generated with every code change or check-in. Next, a code analysis runs. If the code makes it through the quality control gates and then testing, automatic deployment is initiated. Finally, acceptance tests run against the code. When applied optimally, new automation begins at every stage of code check-in. Only one build is tested at a time. If a segment goes red or status updates occur, Devs are notified from source control to end-user.This aids in preventing compound errors that can occur through integration and testing the application’s smallest components.

Pipelines can be constructed in a host of ways. Through container pipelines, teams can define the steps as well as the environments for coding tests, builds, and deploys. The process will be further

defined by the processing tools available.

Stages of Developing Container Pipelines

Here are some processes for developing container pipelines, allowing you to implement a pipeline that's fully effective and useful. Typically, a common four-stage process would include the following stages: Commit and Build, Automated Acceptance, Continuous Deployment, and Production Release.

The details of your processes will depend upon the size of your team, your coding language, the platforms used, and the individual project itself. However, by establishing processes, you can keep your pipeline clear, managed, and stable.

Let's briefly look at each development stage.

Commit and Build

The beginning of any pipeline starts with submitting the newest version of the code into the CI/CD system. At each check-in, the new code is containerized and put through a battery of tests. It's tested continuously until a report of failure or binary artifacts generates. The continuous testing further compiles assemblies to be used in subsequent stages. Testing should only take five to ten minutes. This quick testing confirms whether the specific build of the code is fit for production, thereby eliminating wasted time on broken versions.

Many different tests are used in this stage. Unit tests and A/B tests are just two examples of the many available. Unit tests determine the accuracy of the method and function to eliminate bugs and improve overall code design. A/B tests compare two components or variants of code side-by-side to determine which performs better. These tests are also known as split testing.

Your team may decide on varying pre-commit processes when testing code. However the processes are developed, these early containerization tests occurring in the first phase of this process encourage early and frequent feedback. Such assessments are critical in spotting errors in the code change behavior or errors cropping up in the testing itself. Once all testing specifications are satisfied, the code moves to the next pipeline stage--the automated acceptance stage.

Automated Acceptance

Automated acceptance tests are pivotal in the continuous delivery pipeline strategy. These tests provide specific insight and perspective into the user experience in an environment similar to productions. Environments intended to replicate the application during production are built to test the business acceptance of the tested code/function.

Automated acceptance tests are often referred to as "smoke testing," which is the examination of basic functionality. For example, you can test button response and address any flaws early. Being made aware of these failures quickly allows Devs to eliminate issues down the road.

Continuous Deployment

In the third stage, Devs are heavily integral. Cycles of monitoring and interacting with software to determine behavior and code refinement. In the continuous deployment stage, your team should

run tests such as:

1. Functional Tests: Tests that are performed to test complete application functionality.
2. Regression Tests: Tests comparing new code against earlier versions to ensure correct function.
3. Stress Tests: Tests performance in less than favorable conditions

Continuous deployment and Continuous delivery are not synonymous; they are different tests. Deployment features software updates that automatically push to production. Delivery manually pushes software updates to production when the team decides to do so. These software updates often occur when the product is ready for production release, the last stage of the process. Teams will need to confer with customers on their approach.

Production Release

This is the end loop of the pipeline. As the code heads into a production release, the live application has been screened, the code has been verified to the standardized specs, and the software is ready for release. Pipelines can be labeled as "boring," but in a good way. In developing your container pipeline, you want it to be reliable and consistent, with no unanticipated surprises. Fully-developed containers eliminate wasted resources, lag time, and lack of continuity between teams at different stages of development.

The timeframe of the development stage often depends upon the business and its goals. However, by establishing processes, no matter your project's timeline, you'll create top-notch software that can be distributed to the end-user more quickly. If you'd like to learn more about DevOps and the application of container pipelines, contact BMC to assist in crafting the application software pipeline your team deserves. Discover how our specific DevOps services will streamline your workflow and workforce, with proven practices designed around businesses' high-speed release and delivery demands.