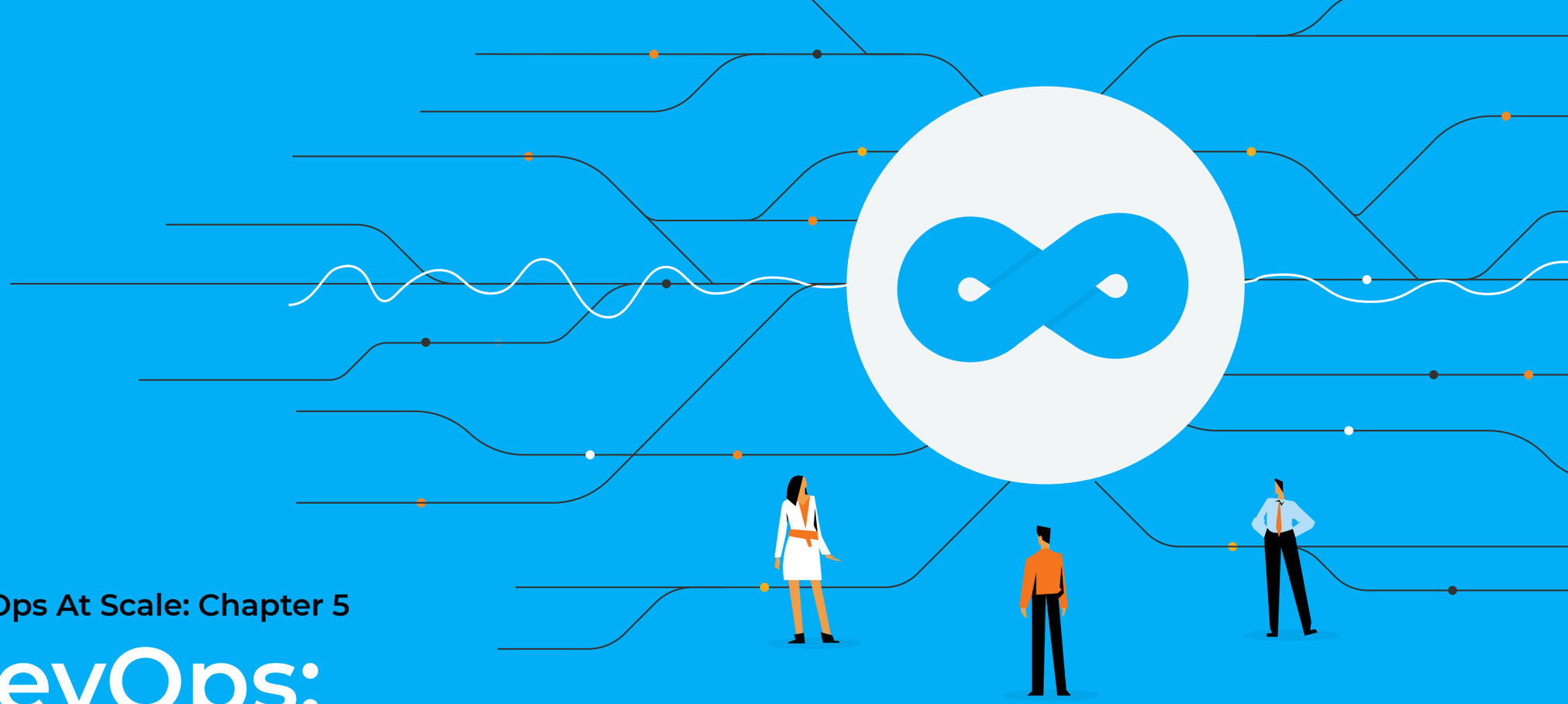


DevOps At Scale: Chapter 5

DevOps: Deployment Pipeline

PLUTORA

DevOps has been gaining immense popularity in the recent past as IT decision-makers across the globe have started realizing the benefits that it offers. Powered by automation and aided by cross-departmental collaboration, DevOps has taken the software development world by storm. However, not a lot of IT executives are aware of the nitty-gritty of a DevOps pipeline. It might be that way because the different concepts involving a DevOps pipeline are often not well defined.

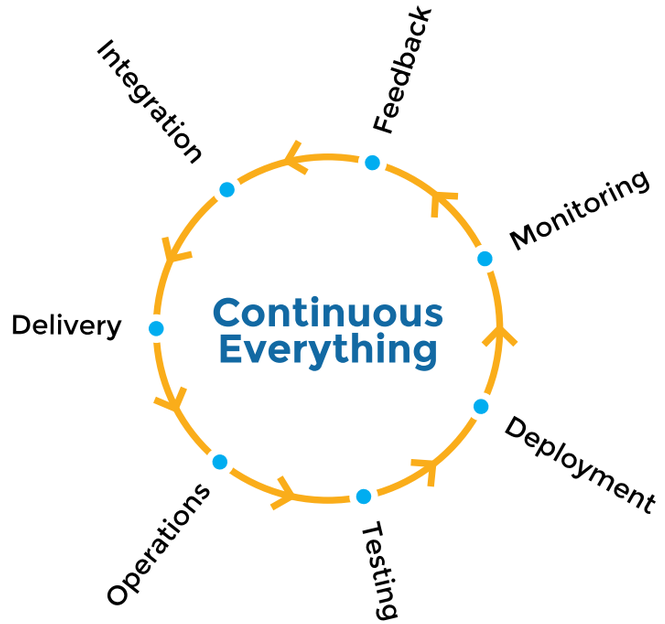
This resource article attempts to break down the different components of a sample DevOps pipeline and explains what a pipeline looks like in the enterprise.

Different Phases in a Typical DevOps Pipeline

The core of a DevOps pipeline constitutes the following: continuous integration/continuous delivery (CI/CD), continuous testing (CT), continuous deployment, continuous monitoring, continuous feedback, and continuous operations. Let's delve into what these concepts mean and how they serve as building blocks for DevOps.

Continuous Integration/Continuous Delivery (CI/CD)

Before continuous integration (CI) was in place, developers built the application features in silos and submitted them separately. The concept of CI has completely changed how developers go about sharing their code changes with the master branch. With CI, the system frequently integrates the code changes into a central repository several times a day. As a result, merging the different code changes becomes easier and is also less time-consuming.



You'll also encounter integration bugs early, and the sooner you spot them, the easier it is to work on resolving them.

Continuous delivery (CD) is about incremental delivery of updates/software to production. While serving as an extension of CI, CD enables you to automate your entire software release operation. It allows you to look beyond just the unit tests and perform other tests such as integration tests and UI tests. As a result, the developers can perform a more comprehensive validation on updates to ensure bug-free deployment. With CD in place, you

increase the frequency of releasing new features. Consequently, it enhances the customer feedback loop, thereby creating the opportunity for better customer involvement.

Thus, CI/CD serve as linchpins to any DevOps pipeline.

Continuous Testing (CT)

Continuous testing (CT) is another key component of a DevOps pipeline. With continuous testing, you can perform automated tests on the code integrations accumulated during the continuous integration phase. Besides ensuring high-quality application development, continuous testing also evaluates the release's risks before it proceeds to the delivery pipeline. Barring the script development part, continuous testing doesn't require any other manual intervention. Testers write the test scripts before the commencement of coding. As a result, once the code integration happens, the tests begin to run one after the other automatically.

Continuous Deployment

There's an element of ambiguity when people talk about continuous delivery and continuous deployment. People often interchange the two terms although there's a substantial difference between them.

Continuous deployment succeeds continuous delivery, and the updates that successfully pass through the automated testing are released into production automatically. As a result, it enables multiple production deployments in a single day. While the goal of continuous delivery is to make your software ready for its release instantly, the actual job of pushing it into production is manual. That's where continuous deployment comes into the picture. And, as mentioned earlier, if the updates can be deployed, they'll be deployed automatically through continuous deployment.

Continuous Monitoring

Monitoring your systems and environment is crucial to ensure optimal application performance. In the production environment, the operations team leverages continuous monitoring to validate if the environment is stable and that the applications do what they're supposed to do. Rather than monitoring only their systems, DevOps encourages them to monitor applications too. With continuous monitoring in place, you can continuously keep a tab on your application performance. The data thus gathered from monitoring application performance and issues can be used to discover trends and also identify areas of improvement.

Continuous Feedback

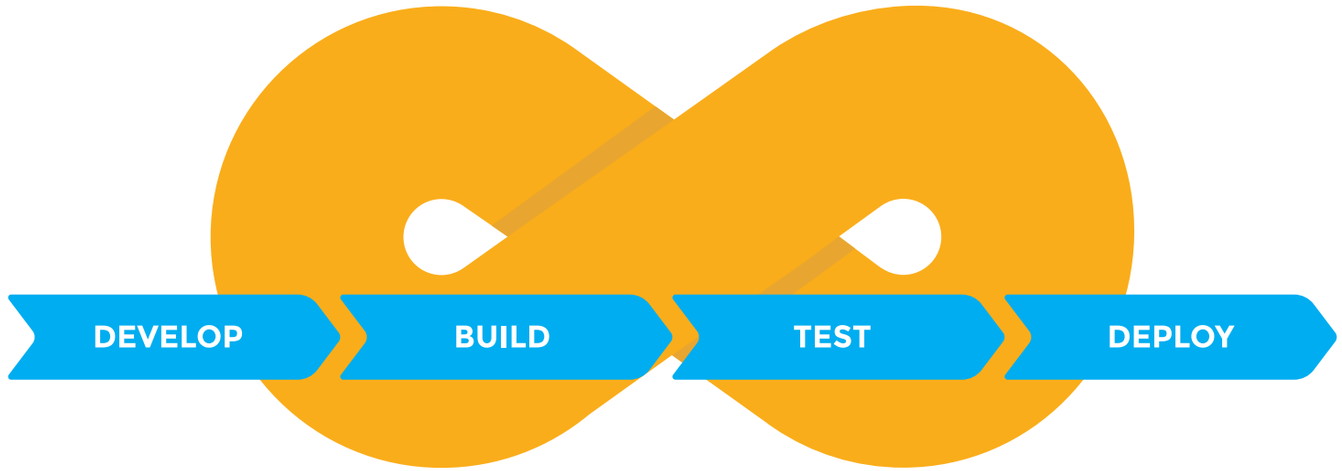
People often overlook continuous feedback in a DevOps pipeline, and it doesn't get as much limelight as the other components. However, continuous feedback is equally valuable. In fact, the purpose of continuous feedback resonates very well with one of the core DevOps goals—product improvement through customers'/stakeholders' feedback. Merely delivering your applications faster doesn't equate to successful business outcomes or increased end-user satisfaction. You'll have to ensure that you and your end users are on the

same page with your releases. That's exactly what continuous feedback can help you do, and that's why it's an important DevOps component.

Continuous Operations

Continuous operations is a relatively newer concept. According to Gartner, continuous operations is defined as "Those characteristics of a data-processing system that reduce or eliminate the need for planned downtime, such as scheduled maintenance. One element of 24-hour-a-day, seven-day-a-week operation." The goal of continuous operations is to effectively manage hardware as well as software changes so that there's only minimal interruption to the end users.

Setting up continuous operations in your DevOps pipeline will cost you a lot. However, considering the massive advantage that it brings to the table—minimizing core systems' unavailability—shelling out a lot of money for it will probably be justified in the long run.



Stages of a DevOps Pipeline

Although it's common to find variations in DevOps pipeline representation, the basic stages include develop, build, test and deploy.

Develop

At this stage, the developers write the software code and push it into a source control repository. After the code passes through the repository, source code integration takes place. There are several code repository hosting services available on the market, along with an underlying version control system. Selecting the best repository can be tricky as it depends on different factors, such as your project and team size, its release schedules, and so on.

Build

In the subsequent stage, "build," the application is built by using the integrated code in the source code repository from the previous phase.

Test

The next step in the DevOps pipeline is “test,” wherein the testers execute different tests such as system tests, functional tests, and unit tests on the build from the last phase. If any issues are found at this phase, then such issues are sent back to the developer for resolution.

Deploy

The final stage is “deploy,” where once the production environment is created and is configured, the final version of the build is deployed.

Thus, the above-discussed simple DevOps pipeline starts from code being checked into the source control repository until its deployment to the end users in the final stage. There’s also a feedback loop that connects all the mentioned stages and ensures that the process of application delivery is on the move at all times.

Deployment Pipeline Automation

DevOps calls for automating everything that can be automated. And except for some rare instances, automation is generally a good idea. There are several deployment pipeline automation tools available on the market. And such tools automate code validation and delivery across the entire life cycle. The obvious benefit of automating your deployment pipeline is that you drastically minimize the time taken for deployment execution. Writing deployment scripts for specific applications consumes a lot of time. However, when you keep working on a product for extended periods, you’ll be able to modify the existing process to automate even more application deployments.

Steps Involved in DevOps Pipeline Implementation

If you’re considering doing DevOps or are in the initial stages of implementation, you should know that a lot of things go into implementing a DevOps pipeline from scratch. There’s no single correct answer to the question “How exactly do I implement DevOps?” It depends on different factors such as the size of the organization, the budget, the toolsets, and the business goals expected out of the implementation, to name a few. This section of the article discusses some of the common steps that involve any DevOps pipeline implementation.

Chalk Out and Establish the DevOps Strategy

Before you commence your DevOps journey, it’s first important to clearly define and establish the DevOps strategy. Also, up-front planning goes a long way for a successful transition to DevOps. Remember that, at its core, DevOps is a mindset. So, it’s about the cultural shift in people as much as the other processes and tools. That’s why a best practice at this stage would be bringing together the people from different departments and facilities to collaborate and work toward the unified goal—speeding up the SDLC while ensuring high software quality. You should also ensure sufficient provisioning of IT infrastructure through infrastructure as code (IaC).

Stick to Agile Principles

Following agile principles in addition to DevOps methodologies can be a great move. While they're two different software development methodologies, they generally complement each other. So, a coexistence of agile and DevOps can be beneficial to organizations. Marrying agile with DevOps should result in an increase in bug-free code and minimized average development time. Agile focuses on software delivery in iterations. When you also use CI/CD for each of those iterations, you accelerate time to market as well.

Do Continuous Everything

Saying "continuous everything is the mantra of DevOps success" is justified. If you're looking to achieve maturity in your DevOps practice, you should strongly consider adopting at least CI/CD and CT workflows. Continuity is very central when it comes to DevOps as these processes ensure that the code quality and deliverable time are maintained across each stage of the pipeline.

There are several other best practices to have at the back of the mind during DevOps implementation. As much as possible, try to use open-source tools to facilitate a smooth integration. Choose your DevOps projects wisely. It's also a good idea to focus on the cultural mindset and establishing standard metrics to measure your DevOps success.

Managing CI/CD From the Application Portfolio Management Perspective

Application portfolio management (APM) was developed during the period of the critical millennium bug. And since then, it has come a long, long way. Today, APM is highly important as it facilitates enterprises to boost revenue through digital transformation.

APM has become an integral element of DevOps as it breaks down the intricacies to improve speed and agility. Big organizations find it hard to adopt the present technology, facing many practical challenges of modernizing and deploying their application without disturbing their existing portfolio. To bring out the best outcome in this situation, managing a CI/CD pipeline with APM is essential. If DevOps implementation results in better agility, accuracy, and speed, APM brings about an obvious business value to organizations with DevOps and strives to produce flawless results in the CI/CD environment.

APM Implementation to Handle Delivery

Here's how you can leverage APM to manage your delivery operation:

Structure the Process

To design the process, you need to collect data on the present IT conditions and the insights on applications and their functionalities to build the roadmap for a high-revenue business. With the gathered data, you can segregate them into relevant and irrelevant data to structure the process to avoid redundancy.

Evaluate

Assess the applications to generate a detailed report on the usage and functionality of the

applications. You'll have to look into every aspect of the application to make any changes in the future. For instance, if you want to upgrade an application, you can easily identify its present dependency and impact to manage the transition.

Define IT Transformation

Create and test plans for a wide comparison involving risks, quality, and feasibility. With the relevant outcomes, choose the plan that best suits your company and implement that transformation process.

As you adopt APM, you can drive DevOps to improve quality and enhance reliability. Further, by rationalizing the application portfolio, you can help companies significantly reduce cost and avoid redundant applications.

As one of our key features, Plutora provides oversights into CI/CD pipelines for product/release/portfolio managers, and even CIOs to some extent, so that they can see what features are currently deployed in a release.

Conclusion

Hopefully, this resource has touched on the key elements of a DevOps pipeline and has taken you a little closer to the DevOps pipeline ocean. The concepts explained in this paper constitute only a small chunk of a complete DevOps pipeline. As you'd know by now, you have to have your hands on a lot of things at once to build a DevOps pipeline from scratch. But once the pipeline is in place, it'll completely redefine how you go about building your software and how you deploy it. And, of course, you'll reap a lot of business and technical benefits in the long run.

If you're considering going forward with a DevOps deployment pipeline or aren't sure of how to achieve pipeline maturity, our service offerings can help you out!

Want to learn more about DevOps?

Check our series of white papers about DevOps to learn
from the foundations, to the cultural change and maturity model.

1. What is DevOps?
2. The Benefits of DevOps Strategy
3. DevOps Methodology: Aligning your Organization
4. DevOps Tools: Why You Need Them
5. The DevOps Maturity Model
6. DevOps Pipeline: The Functional Building Blocks
7. Mastering the DevOps Process
8. Making your DevOps + Agile Transformation a Success

Visit www.plutora.com/devops to learn more.

About Plutora

Plutora, the market leader of value stream management solutions for enterprise IT, improves the speed and quality of software creation by capturing, visualizing and analyzing critical indicators of every aspect of the delivery process. Plutora orchestrates release pipelines across a diverse ecosystem of development methodologies, manages hybrid test environments, correlates data from existing toolchains, and incorporates test metrics gathered at every step. The Plutora Platform ensures organizational alignment of software development with business strategy and provides visibility, analytics and a system of insights into the entire value stream, guiding continuous improvement through the measured outcomes of each effort.

PLUTORA®

Learn more: www.plutora.com
Email: contact@plutora.com