Flux7

White Paper

Optimized Code Delivery Pipelines

Accelerate Innovation and Code Quality with DevOps

Flux7

Optimized Code Delivery Pipelines

Accelerate Innovation and Code Quality with DevOps Automation

Many developers steeped in the world of agile startups view continuous delivery (CD) pipelines as an accepted standard requirement for software development. Yet many companies, particularly large enterprises with traditional infrastructure, still struggle to make this approach a standard part of their development process. Whether you are an enterprise looking to make CD pipelines a standard project element to increase agility and speed time to market, or if you are looking to simply implement code delivery pipeline best practices, this paper will show you how to deliver business value through DevOps-based automation that grows developer output and strategic contributions.

"In Gartner's 2017 Enterprise DevOps survey 2017[1], 41% of the participating respondents report that their organizations are using DevOps. And a further 40% say that their organizations are piloting/planning to implement by year-end 2018." As a cornerstone of a solid DevOps initiative, it's not surprising then that more than 28% of respondents to an Evans Data Corporation[2] survey reported that their organization uses automated code delivery across all their projects, with an additional 37% reporting that their organization employs the strategy for at least some projects.

Driving the significant interest in -- and adoption of -- DevOps-based automated code delivery is its value in:

- Accelerating innovation by creating code delivery pipelines that promote innovation and enable the easy, efficient delivery of quality software.
- Empowering developers to autonomously try new ideas and make architecture changes; and fearlessly break and recreate dev and test environments without worrying about consequences. Greater risk can bring greater reward and distinct competitive advantage.
- Reducing the cost of failure and iteration time while improving continuity, all of which can significantly improve the process of creating quality software.
- Allows teams to run concurrent (parallel) environments for A/B comparisons. And, enables them to create QA and build environments without having to wait for resources, speeding time to market.
- Optimizing developer resources, which assists in addressing and reducing the common need to attract, train and retain skilled programmers.
- Minimizing friction between developers and IT at every step of the process; enabling change in the overall production environment can help reduce overall system resources and costs.

[1]Enabling the Bimodal Enterprise With DevOps Practices Primer for 2018, 1/3/18, Ian Head | Joachim Herschmann
[2]SDTimes, https://sdtimes.com/continuous-delivery/sd-times-blog-continuous-delivery-adoption-rates-rising/

Efficient code delivery pipelines that balance security and agility can be achieved by using automation, coupled with cloud infrastructure. Together these technologies and processes can deliver big dividends in the form of lower development costs, higher quality results and faster times to launch. By focusing on optimizing a delivery mechanism for value, versus just delivering more code, code delivery pipelines can meaningfully and measurably impact business objectives.[3]

"DevOps is an approach that generates tremendous interest due to its ability to accelerate MTTV (mean time to value). Indeed, based on 2016 survey data, 66% of respondents indicated that they achieved faster realization of value, but only 49% reported meeting or exceeding the cycle times requested by customers."

- Gartner[3]

As a result, this paper seeks to show technical managers, operations professionals and developers the attributes of a well architected framework for continuous delivery and how it can help accelerate delivery to meet and exceed customer expectations. We'll also show how individual pieces of the pipeline can be implemented and orchestrated.

Continuous Delivery in Modern Development Environments

Continuous delivery is a design practice where developers use automation to accelerate the software delivery process. It is a high-value process because it enables programmers to cut the overall cost of software development and faster delivery of high-quality code makes it easier for companies to improve customer satisfaction, capture new markets and gain a competitive advantage.

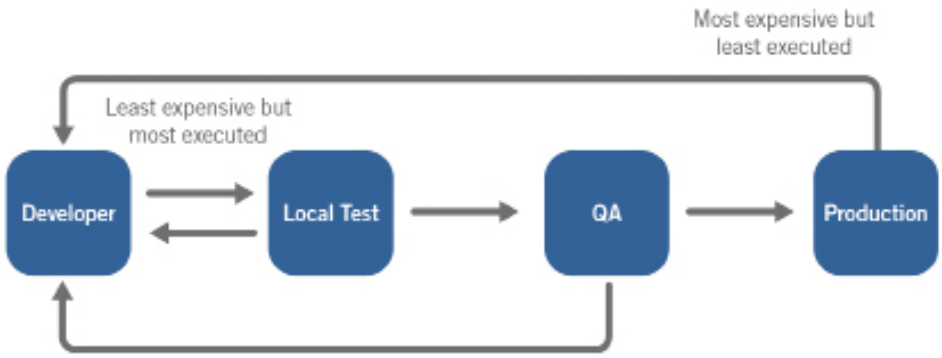
To optimize this development model, companies need to automate their build, integration and testing processes as well as provisioning and retiring infrastructure components.

Organizations will need to leverage infrastructure resources (virtual machines, storage devices and networking assets) to automate the build, integration, and testing the code under development.

The Process of Delivering Value

Code delivery starts from a feature or bug request filed with a developer. The developer takes existing code, modifies it, does local testing, and commits it to the code repository. The code in the repository is then tested by QA and tagged for production, where IT or operations professionals deploy the code to production servers.

[3] Gartner DevOps Requires Faster Organizational Learning, 3/31/17, Geroge Spafford, Katherine Lord



The diagram above describes the full code delivery pipeline.

This creates loops at the local testing, QA, and production stages of the process. Each of these loops has a cost, directly affected by developer productivity and how long it takes to execute each loop.

The local testing loop executes most frequently, so it's critical to make this part of the testing process fast. Production bugs are the least frequent but most expensive for businesses when they exist. Consider the cost of a significant disruption to the workflow, or the negative customer experience, or the inability to deliver a service, or security vulnerabilities. Imagine if you could keepi delivery time and costs under control while minimizing total effort at *every* stage.

Ideally, code should be pushed in smaller batches. Yet, the downside to this approach is the overhead of extra movement that smaller batches creates and the room for error as more and more things move through the process. To reduce the overhead of too many pushes, automation can play an important and meaningful role. Indeed, automation makes it cheaper, faster and more reliable to do things in smaller batches, and is what helps enable streamlined delivery.

Flux7 Approach to Optimized Pipelines

When development environments are optimized, the following attributes are occurring at each stage of the process:

Code to Develop

Committing code often: Micro-agility is achieved when developers test and commit frequently. This approach means that feedback from QA is receive every few minutes rather than once a day and bugs can be tacked and quickly addressed.

An unambiguous process: The latest development is clear, and it's clear where developers commit their code after making changes.

Local Test Loop

Quick to run a test locally: Test should be easy to fully run locally, and is run as a subset of regular QA testing.

The law of PQR: The development environment is:

- **Production-like.** Development and testing environments mimic the production environments very closely.
- **Quick:** Fast and easy to set-up; push button deployment.
- **Repeatable:** Recreate environments that will produce the same results every time.

Promoting Code From Development to QA

QA Ready. Any code pushed by a developer automatically becomes ready for QA.

QA Loop

Automation and Reporting. QA is automated or streamlined and executed in small batches rather than in multi-week long cycles. Reports from QA are communicated to developers immediately. Commits that fail QA are rejected automatically.

Promotion From QA to Production

One click. Promoting to production is a one-click process. There's no ambiguity in which code has gone through QA and is ready for production. Promotion can be achieved through tagging, if needed.

Production Deployment

Full Automation. Deployment is fully automated. It automatically applies any configuration changes made by the developer and approved by QA. Deployment does not lead to system or application downtime and i human errors are greatly reduced.

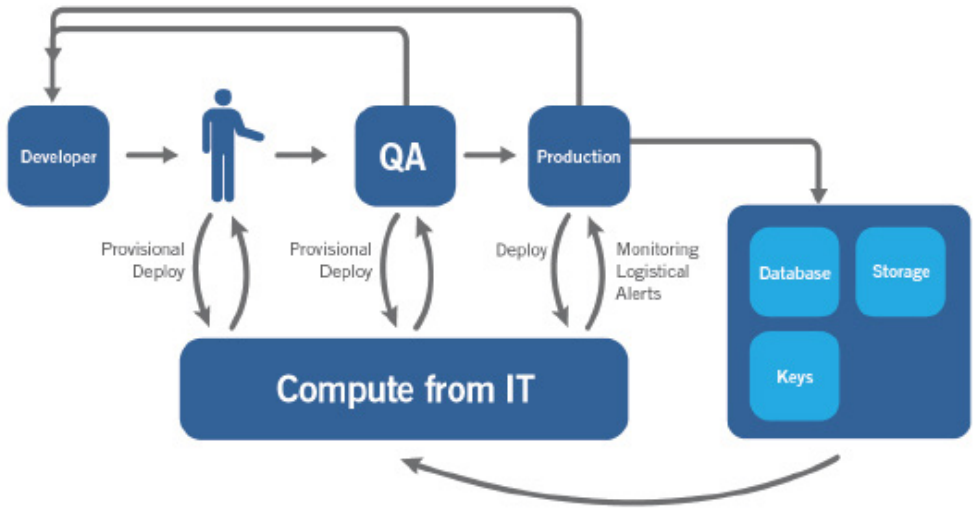
Production Iterations

Monitoring. Monitoring and alerts should be implemented so that you (and not your customers) are the first to know about errors. In production, code is monitored for off-specification system behavior and components that don't function properly.

Alerts. Monitoring is nearly useless without alerts. But, too many alerts are as bad as too few. In production, every alert should lead to an action, which fixes a problem. If there is no problem, a clearly written note of investigations performed should be made. Then, alerts are updated to prevent them from firing again.

Alerts are classified by three preset rules:

- Alerts that are handled automatically but tallied.
- Alerts to be fixed by the ops staff.
- Alerts in application logs that should be sent to developers. Normally, production servers are not changed manually unless drift monitoring is used.



Case Study: Dev Workflow as a Modernization Test

A large US healthcare software vendor with on-premise IT was interested in modernization, but wanted to start the process by focusing on a discrete area where there was low risk and a high return on investment was expected: optimizing development workflows.

The organization specifically wanted to address the following in its new development pipelines:

- Provisioning infrastructure required manual intervention by operations, and caused developers to wait until compute power was provided.
- To work efficiently, many users needed access to development and QA environments at the same time.
- Healthcare information compliance regulations required them to generate audit trails as changes were made to the company's code or infrastructure.

They needed a solution that would help them:

- Build automated controls for provisioning and release to production.
- Automate the delivery pipeline from development to production.
- Provide a proof of concept with a sample application.

Solution

The optimized developer workflow provided an agile, scalable development and QA infrastructure. Originally, because of compliance concerns, the company was reluctant to move from on-premise infrastructure. The final architected solution implemented the same level of controls on-premise while gaining the agility of the cloud and leveraged Docker containers to provide self-service IT capabilities to developers. As the development and QA teams grow, cloud-based services could be scaled to support development activities, without the need for extra capital expenses or over-purchasing and under-using compute power.

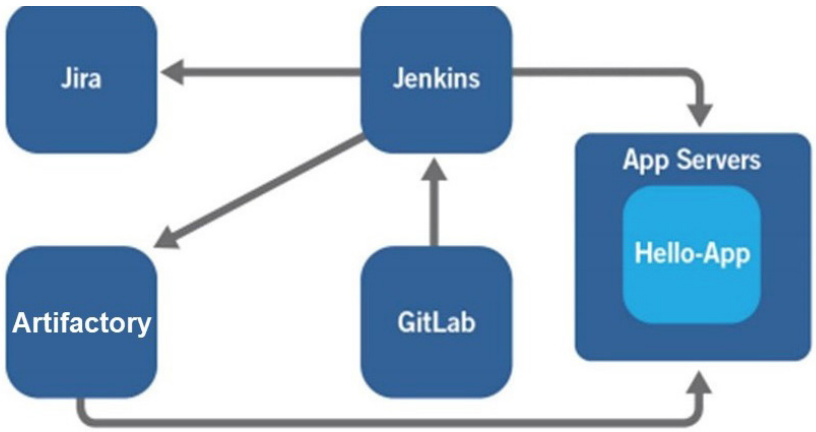
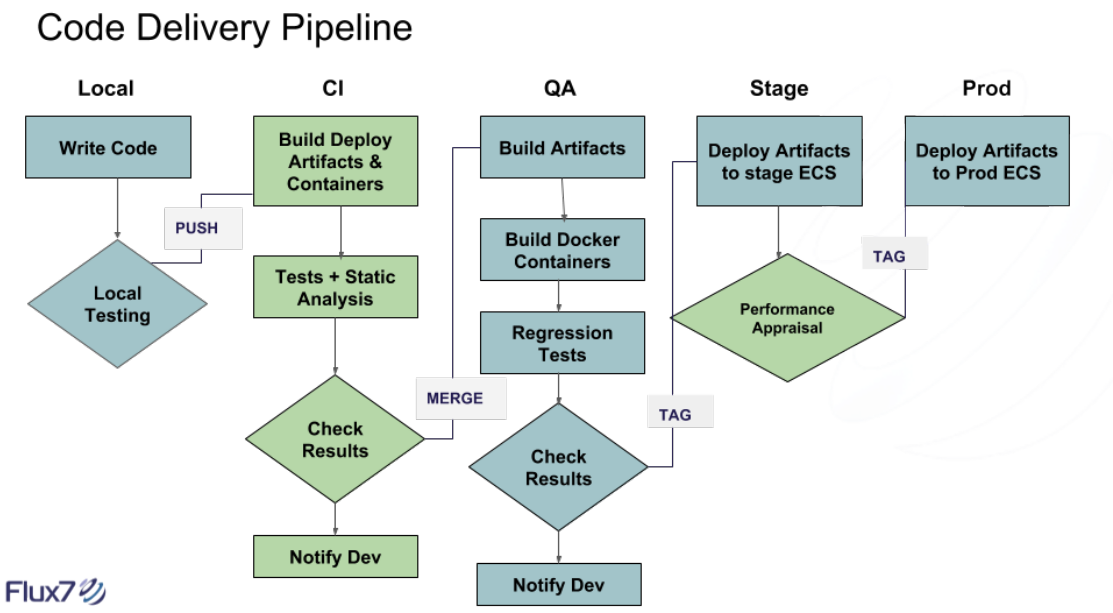


Diagram 2: The new, optimized code delivery pipeline

Now, the new development and QA infrastructure and processes are managed mainly by scripts, not engineers. The optimized pipeline features:

- An audit trail and dashboard that makes monitoring and alerts easy to view and manage.
- Support for a rapid deployment strategy and new agility that resulted in higher developer productivity.
- Cloud and container technology that speeds time to market of new applications and features.
- The ability to mitigate risk with manual checks of services in the pipeline, balancing automation with human approvals to effectively manage risk.



Case Study: AWS DevOps Automation

DevOps automation like automated Code Delivery Pipelines can be more easily achieved when coupled with cloud technology. For example, for a Fortune 500 enterprise, the Flux7 DevOps team set up an automated code delivery pipeline using CloudFormation, AWS CodeCommit, AWS CodePipeline and Jenkins. Developers would use a Git flow methodology, and once they merged code to the master branch in CodeCommit, polling would alert to a new commit. CodePipeline would create a CloudFormation ChangeSet, and queue it up for review by the Ops team. Once the Ops team approved the change set by manually pressing a button, CodePipeline would trigger a CloudFormation stack update. This allowed for very high full-stack agility while not losing central governance.

Aptly named, AWS CodePipeline is AWS' continuous delivery service. It builds, tests, and deploys code every time there is a code change.

AWS CodePipeline allows developers to model release processes through a series of stages - tailorable to your team's models. AWS CodePipeline orchestrates each step, from building from a source, to software testing and deployment.

AWS CodePipeline integrates with AWS services such as AWS CodeCommit, Amazon S3, AWS CodeBuild, AWS Code Deploy, AWS CloudFormation and more.

Conclusion: Value of Optimized Developer Workflows

Gartner[4] finds that "more than 70% of organizations report that the rate of change has increased, and is accelerating." Many leaders in technology organizations are moving to Agile and DevOps as methods of improving the speed of delivery, of which an automated code delivery pipeline is foundational. Optimized development pipelines deliver significant value by minimizing the time and costs of local test, QA and production processes and by promoting code from one development stage to the next.

When you engage in this process, you reduce the time it takes to identify problems and their causes. And when you automate development steps, you dramatically reduce the cumbersome and time-consuming tasks that slow down the release of software code. At Flux7, efficient developer workflows in a cloud-based environment are the foundation of pipeline optimization. Their importance relates to the need to:

- **Control ever-increasing developer labor costs.** Your monthly developer labor cost is almost guaranteed to exceed the monthly cost of your cloud server setup. Keeping your developers working efficiently means you get the most out of that labor and can focus that on what projects are bringing the most value to the business.
- **Get a clear understanding of developer activities.** When we interview developers and IT team members, what emerges from these discussions is astonishing. Companies learn that they really don't know as much as they thought they did about their employee's day-to-day activities and workflow.
- **Find developers with the required skill sets.** Hiring the right talent is costly and time-consuming. If you can help your developers work more efficiently, you can avoid the search costs. And, building a reputation as a company that supports developers helps attract new talent.
- **Accelerate time to market.** It's possible to improve the performance of less productive developers by accelerating their work process with optimized code delivery pipelines and by building best practices in through automation.
- **Avoid rework-related slowdowns.** Rework comes at a cost of time and effort. Automating development processes can avoid many of the human errors that cause rework.

An effective code delivery process is highly valuable as it reduces risk from each release, ensures that rogue code is not introduced, and helps organizations effectively manage global deployments. Most importantly, optimized code delivery pipelines have the ability to significantly impact business objectives and business outcomes, making DevOps a strategic contributor to achieving business imperatives.

[4] Choosing the Right Tools for Your DevOps Toolchain, 11/9/17, Christopher Little and Jim Scheibmeir

About the Author

Aater Suleman, Ph.D. is co-founder and CEO of Flux7, an Austin-based IT consultancy. He is recognized internationally for his innovative work, speaking engagements and published papers on computer engineering. Suleman uses his extensive background in computer hardware, performance optimization and software development to create self-healing cloud infrastructure frameworks. These products enable organizations to use infrastructure as a service with minimum effort and cost. A passionate researcher and active mentor, Suleman continues as a member of the faculty of his alma mater, the University of Texas, Austin.

About Flux7

As DevOps and AWS experts, Flux7 offers a suite of solutions that help organizations design, build, own and manage IT modernization projects. Focused on architecting and optimizing their clients' AWS infrastructure and training internal IT teams to manage their own infrastructure, Flux7 solutions are rooted in DevOps best practices. Flux7 has delivered hundreds of agile, right-sized projects to satisfied customers across industries, creating a well-architected core from which these business can own and expand their IT modernization. For more information, please visit http://flux7.com.

Copyright