

# Kubernetes and CI/CD

Containerization of Continuous Integration and Continuous Delivery

BROUGHT TO YOU IN PARTNERSHIP WITH



# Table of Contents

3	Highlights & Introduction BY ANDRE LEE-MOYE
4	Key Research Findings BY THE EDITORIAL TEAM
8	Time Series Applied to CI/CD Pipeline Optimization BY DANIELLA PONTES
10	Where Kubernetes Is Headed in the Next Five Years BY STEFAN THORPE
16	Building a Development Ready Kubernetes Platform BY ANITA BUEHRLE
17	Effective Kubernetes Deployment Is More Than Choosing a CI/CD Product BY BOB RESELMAN
23	Diving Deeper Into Kubernetes and CI/CD

To sponsor a Trend Report:

Call: (919) 678-0300

Email: [sales@devada.com](mailto:sales@devada.com)

# Highlights & Introduction

By Andre Lee-Moye, Content Coordinator at DZone


Continuous integration and continuous delivery (CI/CD) have been synonymous with DevOps process transformations since the first efforts to condense the overall release pipeline. Continuous integration design practices result in software that is modular and can be repeatedly integrated with other components into a single source, while continuous delivery ensures that software is production-ready at all times. Together, CI/CD processes fulfill the promises of DevOps to create shorter time-to-market cycles, tighter feedback loops, and deployable products that can be updated in minutes instead of weeks.

There were few initial indications that container orchestration tool Kubernetes would become as widely adopted as it is. However, as an open source product with the veritable backing of Google's resources and community, Kubernetes entered the market as an easily accessible and authoritative tool. Since that introduction a mere five years ago, in 2014, Kubernetes has become a staple in the infrastructures of startups and top tech companies, as either a vanilla configuration or at the foundation of the managed container solutions of other cloud vendors.

The rise of Kubernetes came at a fortuitous time for organizations grappling with DevOps transformations and enterprise-wide CI/CD optimization, and the once-groundbreaking tool is now commonplace across the CI/CD lifecycle. In light of the prominence and growing necessity of Kubernetes in any discussion of CI/CD processes, we examine the present and future state of both Kubernetes and CI/CD individually, and together as cooperative and opposing entities.

The consideration of Kubernetes as an individual tool is seen within "Where Kubernetes Is Headed in the Next Five Years," as author Stefan Thorpe ponders how Kubernetes will be implemented in the near future. After a brief reflection of the meteoric rise of Kubernetes as the premier container orchestrator, Thorpe explains the continuing role of Kubernetes in creating common planes for both development and operations. He then explores its development as a more secure, full infrastructure management tool.

Bob Reselman's "Effective Kubernetes Deployment is More Than Choosing a CI/CD Product" details the stages of an effective CI/CD process as defined by the Build, Release, Run principle from the 12 Factor Application Methodology. Part of an overall warning against using Kubernetes as a panacea for a full CI/CD pipeline, Reselman notes how automation helps free developers from the more mundane tasks. With respect to the Dev/Prod Parity, he concludes by examining the hidden and clear dangers present in following an incomplete process.

Findings from our survey of software developers, engineers, and architects provide insight into the current state and future of Kubernetes and CI/CD, aligning closely with the predictions introduced in what follows. 

# Key Research Findings

By the Editorial Team, DZone

Working in a continuous integration and continuous delivery (CI/CD) environment is typically a major goal of DevOps teams. To package applications and products throughout the DevOps lifecycle, many teams are turning to containers. According to the DZone 2019 DevOps survey, half of all software organizations globally have an official, specified DevOps function. Repondents report the role of these teams is to “help the organization adopt the best continuous delivery tools” (70%), “introduce automation across the SDLC” (68%), “increase collaboration and break down silos between Dev and Ops” (56%), “develop and deliver software across the entire stack (54%). Given the broad mandate for DevOps, it’s not surprising to find an increasing interest in all aspects of the approach.

Container orchestration tools such as Kubernetes allow DevOps teams to manage these containers, leading to faster deployments and shorter time to market, while also satisfying heightened customer expectations and remaining competitive. To better understand the industry’s perspective on Kubernetes and CI/CD, we conducted a survey among DZone’s global audience.

## Adoption of Kubernetes

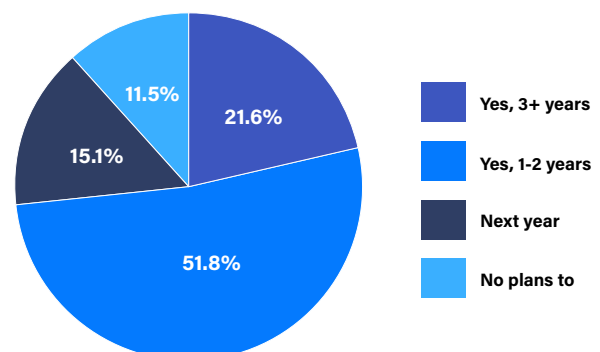
Three quarters (73.4%) of the companies represented in our survey are currently using Kubernetes, and most of them are recent converts to the use of the container-orchestration system. Only 21.6% have used Kubernetes for three years or more. The other 51.8% have adopted it in the past year.

We expect this trend to continue increasing. According to our research, another 15.8% of companies have plans to adopt Kubernetes in the next year. Only 11.5% said they have no plans to use Kubernetes. Kubernetes adoption is relatively consistent across the three regions of the world: APAC (74%), EMEA (73.7%), and NA (72.5%).

### TREND PREDICTIONS

- ▶ Kubernetes will continue to dominate as the most popular tool for deploying and managing containerized applications.
- ▶ Increasing adoption of Kubernetes will accelerate organizational productivity, growth, and scalability.
- ▶ More than two-thirds of organizations will achieve both continuous integration and continuous delivery in the near future.

Figure 1: Does your organization use Kubernetes?



What does vary, however, is how recently companies have adopted the system.

Over one-quarter of developers (27.1%) from APAC report that their company has been using Kubernetes for three or more years versus 19.4% in EMEA and 21.5% in NA. Only 15.7% of those in NA say that they have no plans to adopt Kubernetes.

## Environments for Kubernetes Use

Kubernetes is not limited to a single environment. The majority of users employ Kubernetes in:

- Production/deployment (82.0%).
- QA/testing (81.2%).
- Development (80.1%).
- Staging (73.5%).

Most companies (53.5%) use Kubernetes in all four environments. 21.1% use it in three of four, 13.5% use it in two of four, and 11.9% use it in only one. Those who use it in only one environment typically use it in production/deployment (56%).

**Table 1: Does your organization use Kubernetes in a DevOps environment?**

	TOTAL	APAC	EMEA	NA
YES	90.8%	87.3%	92.0%	91.9%
NO	9.2%	12.7%	8.0%	8.1%

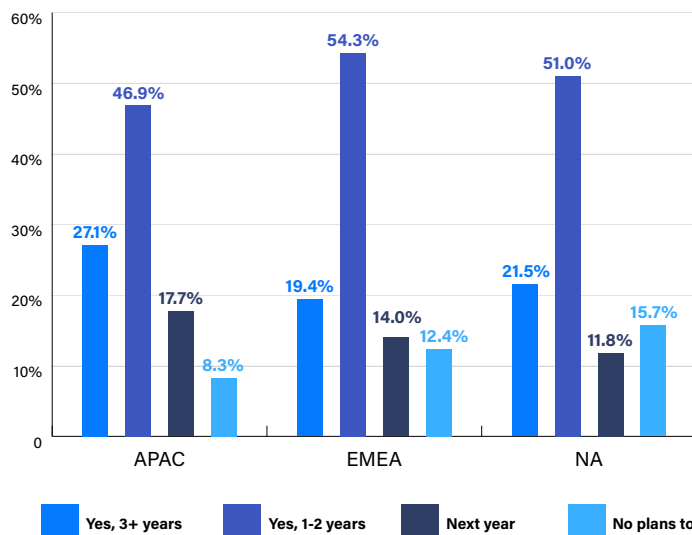
Globally, deploying code to production more often falls to the development team (44.2%) instead of the operations team (34.1%).

However, the global totals mask differences by region. In EMEA, the job is executed by development teams most often (49.3%), while in APAC, operations teams are responsible for deploying code (44.9%).

In NA, operations and development teams share more equal responsibility for deploying code.

Globally, release engineers are responsible less than 20% of the time. In NA, that's more frequent (27.1%) than in any other region. The QA team rarely manages that task in any region.

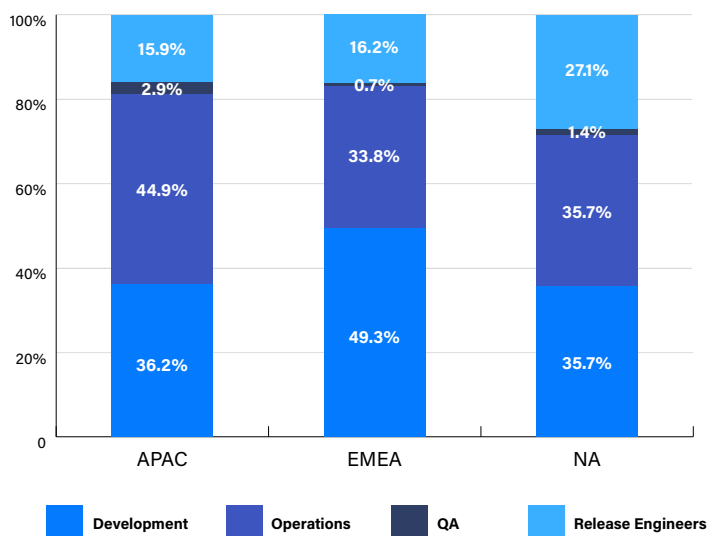
**Figure 2: Kubernetes Use by Region**



The development environment comes in second place with 30.6%, QA/testing comes in third with 11.1%, and only 2.8% use Kubernetes exclusively in their staging environment.

While Kubernetes may not be used across the entire software development lifecycle, those using Kubernetes (90.8%) say they use it in a DevOps environment.

**Figure 3: Regional difference in who deploys code**



## Benefits and Challenges of Kubernetes

Our developer audience tells us there are many reasons to adopt Kubernetes. The most cited reason for adoption is scalability (81.8%). Since scalability is crucial to a firm's ability to grow at a pace that maximizes performance and productivity, it's no surprise that many identify it as a major benefit of Kubernetes.

Surveyees selected several challenges of using containers:

- Lack of developer experience with containers (60.6%).
- Refactoring/rearchitecting legacy applications (48.7%).
- Application performance monitoring (42.5%).
- Monitoring (39.2%).
- Ensuring application and network security (35.7%).
- Storage scaling (30.7%).

A large majority of respondents say Kubernetes makes it much easier to employ containers, though Kubernetes doesn't help with the lack of developer experience with containers.

While some developers believe it does lessen the burden (21.8%), more believe it makes the situation harder (33.1%). Overwhelmingly, developers who use Kubernetes think that it makes working with containers easier (77.4%).

## Continuous Integration and Continuous Delivery

In the past year, there have been enormous gains in achieving continuous integration and continuous delivery in the pipeline. Continuous integration achievement leads continuous delivery, but the gap may be somewhat surprising.

This year, just over half (51.2%) report that they have achieved continuous integration, up from 30.6% only 12 months ago.

Continuous delivery has shown similar gains with 29.3% reporting it has been achieved — up from only 13.9% one year ago — and an even greater proportion expect to hit the mark soon.

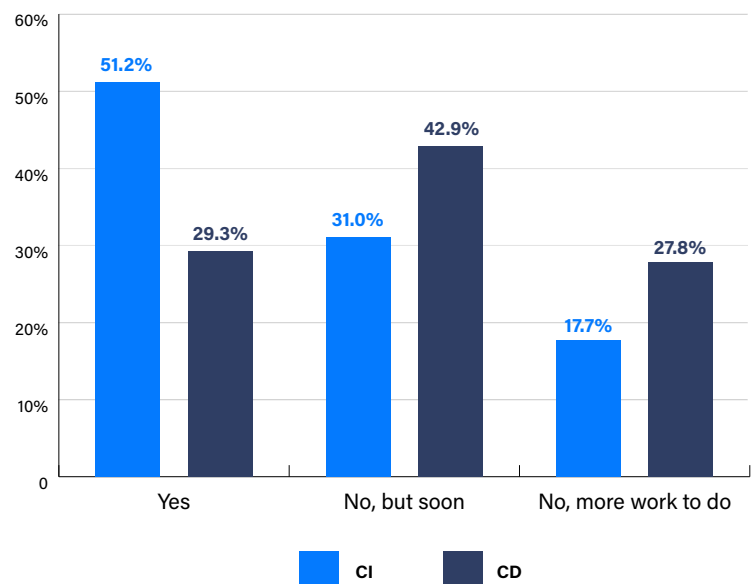
Not only is the CI/CD pipeline becoming a reality in an increasing number of organizations, those not at that stage expect to be there soon, 31% for continuous integration and 42.9% for continuous delivery at.

These projections suggest that 80% of companies will achieve continuous integration and 70% continuous delivery.

Table 2: Reasons for adopting Kubernetes

Scalability	81.8%
Works well with our CI/CD pipelines	65.3%
Fault tolerant	59.9%
Efficient resource management	58.5%
Open source	58.3%
Efficient application updates	56.9%
Multiple workload and deployment options	50.1%
Integration with major cloud providers	48.0%
Extensibility	40.1%
Good ecosystem of supporting tools	36.9%
Large user community for support	35.0%
Built-in security	25.2%
Works well with our PaaS	20.6%
Flat networking model	13.6%

Figure 4: Many more expect to achieve CI/CD soon




## Conclusion

The container revolution is moving quickly. The majority of technology organizations have embraced Kubernetes for container orchestration, most often within a DevOps environment. Achieving continuous integration and continuous delivery are on their way to becoming commonplace as well.

This suggests that most developers are now able to focus on individual layers in the technology stack and not on the entire infrastructure. Using Kubernetes helps developers better manage the increasing complexity, enabling organizations to scale for growth more easily. Combining the container orchestration system with a CI/CD pipeline provides more frequent software releases with improved quality, making both customers and developers happier.

## Methodology

DZone conducted a quantitative survey among its global audience of developers, architects, and software engineers during November of 2019. Results from 415 respondents informed the report. 

# Time Series Applied to CI/CD Pipeline Optimization

By **Daniella Pontes**, Sr. Product Marketing Manager at InfluxData

CI/CD, which stands for continuous integration and continuous delivery — and can also encompass the concept of continuous deployment — is at the heart of DevOps culture. By embracing the power of small, frequent code integrations with testing and workflow automation, organizations have managed to keep up with time-to-market pressure and user demand for better and faster applications and services.

Nonetheless, much more can still be achieved through holistic observation of the entire lifecycle of software releases. Organizations will not only gain further agility but also the confidence in meeting business needs and customer satisfaction in an optimal and consistent manner.

CI/CD monitoring — extended with enriched observation data — generates more insightful indicators derived from additional contextual and correlated information. By tapping into this rich data, each release cycle contributes to a better understanding of critical areas of code change and their impact coverage, thereby allowing continuous process improvement and optimization.

Furthermore, holistic observation and historic data analysis produce simple and low-hanging benefits to automation. Outcomes include identification of error-prone points; the reach of impact of changes on certain functions and UI aspects; better categorization of issues aligned with appropriate alerting and actions; adaptive workflow automation based on well-understood triggering events; and much more, since time series provides a limitless source of extracted value.

In summary, a lot can be learned from taking release cycle monitoring a step forward by sending a rich set of observed data to a [time series platform](#), in which such diverse types of data (numeric and non-numeric) can be stored long term and subjected to advanced analytics. Why not bring together metrics, logs, events, true/false outputs, errors, environment exceptions, RUM, and synthetic user monitoring to one place to visualize data from multiple perspectives?

With such an exploratory platform, monitoring testing results to review frequency of success and failures could, for instance, provide important feedback to the planning phase of a code change. Moreover, observing application quality and performance at various stages; keeping records of environment errors, events, and exception handling; and tracking the impact of changes on users all lead to the ultimate goal of achieving more mature practices — and consequently — safer and smoother continuous integration, delivery, and deployment of application software.



# A Customer Case Study: Playtech

By **Chris Churilo**, Director of Product Marketing at InfluxData

The nature of Playtech's operations — involving financial transactions in highly regulated markets — demanded high-level monitoring. High-level monitoring of their distributed system required analyzing financial data, which is time series data.

The architecture of the system is service oriented, the components have complex business logic, and there are different backend variants for different B2B clients (companies). As a result, Playtech achieved these tangible results:

- Detect service-start problems earlier because they see activities that are critical — and have special rules for that — begin to degrade (some services didn't start well previously when they made a deployment and had some mistakes in their configuration). Playtech can quickly catch numbers of incidents within a week, enabling very quick response.
- The system combines monitoring and alerting for business indicators (BIs) and Key Performance Indicators (KPIs) using the regularity of economical processes. This takes them beyond monitoring to observability for a deeper understanding of system behavior and causes, relationships between system components, and needed action.

*"Why InfluxDB? For Playtech, it was very important to have observability, to understand system behavior to predict possible outages and problems in the very early stages."*

**Aleksandr Tavgen**, Technical Architect at Playtech

## Product

InfluxDB

*Real-time visibility into stacks, sensors and systems*

## Category

Time Series Data Platform

## Release Schedule

Quarterly release cycles

## Open Source

Yes

## Strengths

- Built for developers
- Trusted by Ops
- Vital to business

## Notable Users

- Capital One
- PayPal
- Comcast
- Wayfair
- Optum Health

---

[influxdata.com](https://influxdata.com)

[influxdata.com/blog](https://influxdata.com/blog)

[@InfluxDB](https://twitter.com/InfluxDB)

# Where Kubernetes Is Headed in the Next Five Years

By **Stefan Thorpe**, Head of DevOps and Security at Cherre and CTO at Caylent

This article considers the future of Kubernetes and how the platform will evolve over the next five years. We examine:

- How Kubernetes originated and its soaring rise in popularity.
- The closer unification of infrastructure and application code.
- The current integration of core functions to improve the management of integral ops components.
- Early manifestations of a future Kubernetes platform-as-code functionality.

Blink, and you could miss it. Kubernetes' exponential growth to de facto choice for container orchestration has made it the fastest growing open source project in history. The foundation of Kubernetes is the orchestration and management of Linux containers to establish powerfully distributed systems for deploying multiple applications on hybrid cloud environments.

It's rise to the #1 super platform is scarcely credible — especially given that it essentially started as a homebrew project inspired by the best elements of Google's internal cluster management system, Borg, back in 2014.

## Historical Development and Rise in Popularity

Given its start in life, predicting the future of such a popular platform could be deemed futile, but factoring in Kubernetes' unprecedented support and huge technological patronage are what make the exercise interesting, to say the least. Never before has a lone open source project attracted such support, input, and collaboration from the software industry.

As well as making managing containerized workloads portable and extensible, Kubernetes' prestige lies in its customization capabilities and that it facilitates both declarative configuration and automation. These are all reasons why nowadays, no one is going to lose points for leveraging Kubernetes as their container orchestrator of choice for building for applications.

Its popularity is the reason it didn't exactly take long for the major public clouds to launch their own Kubernetes-based platforms (AKS, GKE, and EKS). And the on-premise guys weren't far behind with platforms based on Kubernetes as well (Openshift, Pivotal Container Service, etc.).

More and more PaaS systems are evolving into Platform-as-Code in Kubernetes. Inspired by trailblazers in Infrastructure-as-Code, such as AWS Cloud Formation and Terraform, these 'Platform-as-Code in Kubernetes tools' are highlighting the future of Kubernetes, which is slowly moving infrastructure code and application code closer and closer together.

## **Kubernetes Helps Dev and Ops Teams Optimize Workflows**

Traditionally, development and operations within the development pipeline comprise two very distinct skill sets: x and x. Over the last few years, two tooling approaches have formed. One set, which broadly covers PaaS, considers the end-to-end developer workflow while trying to remove any need for Ops engineers. The other set, generally covering Infrastructure-as-Code, examines ops difficulties in provisioning infrastructure through automation. Communication issues and technical incidents tend to occur in teams of dev and ops together over a lack of common tooling.

In more recent times — and with the increasing adoption of the DevOps methodology — ops teams have caught up with code automation, and devs are taking on more ops responsibilities to improve the seamless collaboration of workflow between the two departments. With the unification of infrastructure and application code in Platform-as-Code, there is a potential to overcome these issues with kubectl and Kubernetes YAML as the common language and tool for devs and ops working together. Optimizing development and operations together in an ‘as-Code’ approach leverages repeatability, collaboration, version control, and improved resource management across corresponding dev and ops elements.

Kubernetes Operators are already a hugely significant working element in the application build process. Designed to deliver automated application lifecycle management, Kubernetes Operators free devs up from having to deal with the time-consuming and complex nuts and bolts of deploying, scaling, reconfiguring, and upgrading applications. The future is not too far off from a time when Kubernetes Operators will be leveraged to manage not just the application lifecycle, but to include the underlying server, network, and storage infrastructure as well.

## **The Not-so-Distant Future of Kubernetes**

The next five years will see more teams leveraging platform-level functionality in an ‘as Code’ manner. Kubernetes will further enable us to describe an application and platform dependencies in a high-level, declarative representation, while at the same time supporting the deployment of the application stack in a repeatable manner via this description.

Early manifestations of this future Kubernetes platform-as-code functionality are indicated in the current integration of core processes in Kube, which make it easier to manage integral ops components. If you leverage platform elements such as Custom Resources in your application YAMLs with native Kubernetes resources like ConfigMap, Deployment, and Service, then you are already essentially customizing a platform for your YAML application code.

Just as when working with Helm and Kubernetes Secrets, we’re managing infrastructure code next to application code in a repeatable, shareable manner.

### **CONTINUED GROWTH TOWARD GREATER ACCESSIBILITY**

Over the next five years, I believe we’ll see the transition become more seamless as infrastructure and application code get even closer together in an ‘as Code’ approach, especially since Kubernetes is designed extensively to optimize configurability and flexibility. What this means at the moment is that there are hundreds of potential approaches for deploying a Kubernetes cluster.

Currently, Kubernetes is simple enough to set up; there is well-documented tooling and a solid do-it-yourself guide to follow. However, it becomes more complex to manage in production long-term without experience. I think the future will see the Kube community and vendor army focus on ease of use to support this transition and make adoptability much easier.


Kubernetes’ move from container orchestration to full infrastructure management is all but on the cards already. In an attempt to avoid lock-in with any one vendor or provider, users already optimize Kubernetes as a common abstraction layer for applications that run over physical, virtual, private, and public cloud environments.

The future will see users being able to leverage Kubernetes to manage converged workloads on third-party clouds, while retaining the freedom to deploy, run, and manage applications on their cloud of choice without requiring admins and devs to learn multiple APIs and environments. Kube will allow users to combine and manage compute, storage, and networking in a single framework in one location.

## **EVOLVING METHODS FOR SECURITY INTEGRATION**

Not to deviate too far from the main pathway of this article, but I believe another major aspect of Kubernetes that will see significant change over the next five years is security. Within Kubernetes, the whole method of security integration still confuses users. Even over the next 6-12 months, I believe we'll see more collaboration and input from the vendor army in the form of tools to improve this process as well as service offerings from within the Kube community.

## **Kubernetes and Future Innovation**

Despite Kubernetes having been around for several years now, there seems to be no indication of a change in its upward trajectory for future innovation. And it seems like there's no stopping Kubernetes dominating the future of infrastructure and application management through Platform-as-Code. Looking ahead over the next five years, I foresee the merging of these two trends playing a significant role in the software development world and for enterprise customers. 

# Ahoy, Kommander.



Deliver federated management and governance across disparate clusters for any on-premise or cloud Kubernetes distribution.

## KEY BENEFITS

- Consolidated Multi-Cluster Governance
- Simplified Operational Management
- Greater Management Flexibility
- Delivering Centralized Governance



**D2  
IQ**

Ensure smooth sailing for  
Kubernetes with D2iQ  
[Learn More →](#)

# Introducing Kommander

By **Rahul Dabke**, Director of Product Marketing and Analyst Relations at D2iQ

Developed to address the broad issues caused by Kubernetes cluster sprawl, D2iQ's Kommander delivers federal management, governance, and visibility across an organization's expansive use of Kubernetes clusters.

As various parts of the organization require new Kubernetes clusters and associated data services, Kommander simplifies the provisioning of new services and creates greater policy-driven consistency across Kubernetes clusters within the environment. This capability allows for greater standardization of Kubernetes to ensure an easier support path for application services.

Further, this level of broad control can help organizations meet risk and compliance demands as they govern how and where new application services are used, as well as who is able to engage in policy and operational needs of those services.

Key Benefits:

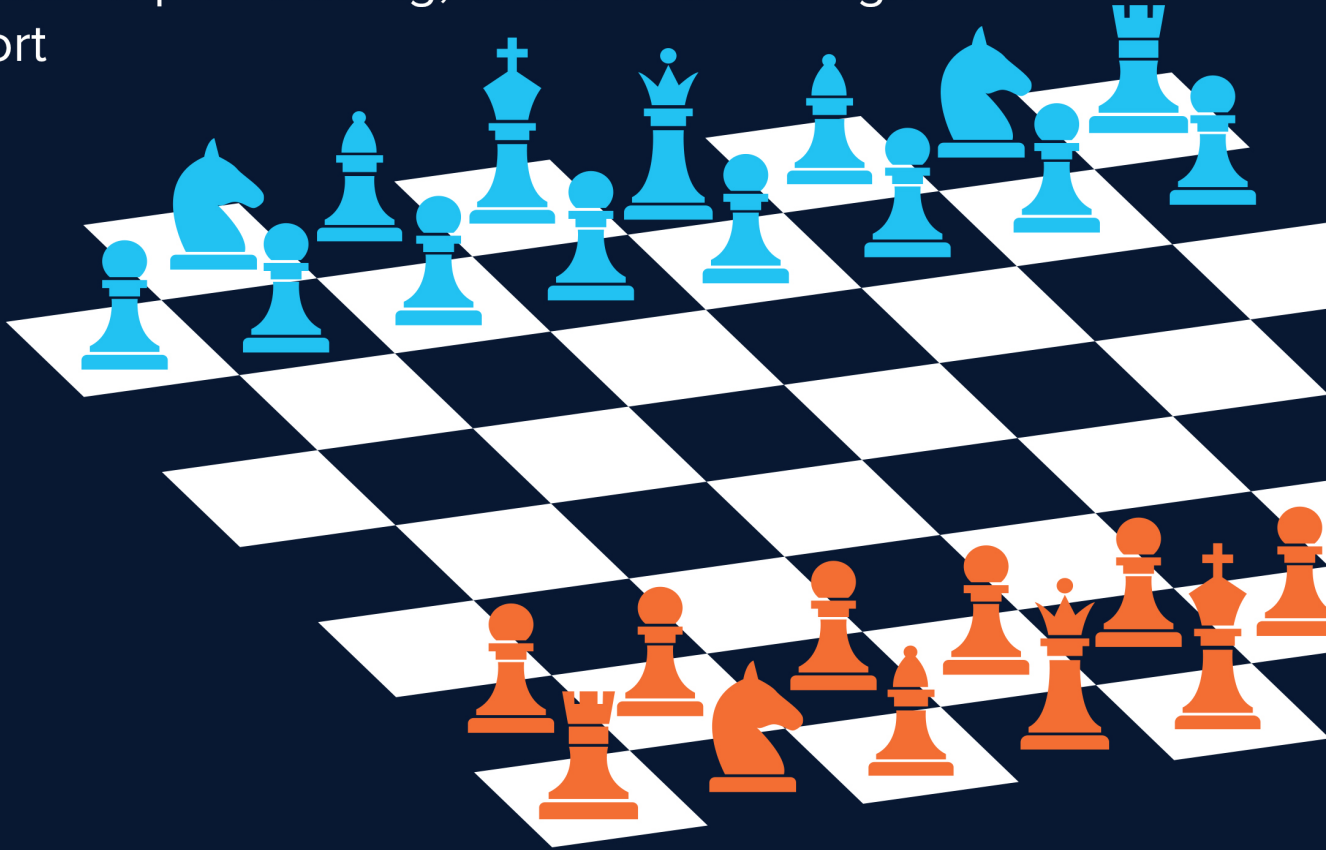
- **Governance and policy administration** – Centrally govern clusters and create lines of separation across various projects.
- **Reduced complexity** – Create greater configuration consistency across organizational clusters.
- **Unified infrastructure** – Deliver multi-cloud and hybrid experiences across a myriad of Kubernetes distributions.

## TREND PREDICTIONS

- ▶ Kubernetes cloud-native cluster sprawl will continue exploding.
- ▶ Cluster sizes will continue to grow, necessitating simplified management.
- ▶ Security and governance requirements will drive enterprise IT Policy.
- ▶ Best-in-class offerings for cloud-native technologies will emerge.
- ▶ Infrastructure and operations will need tools that automate orchestration and observability.

# Plan your next Kubernetes move

Weaveworks Quickstart: Accelerate your Kubernetes adoption with expert training, architecture design and support



## Kubernetes Automation

Day 2 Operations for Kubernetes doesn't have to be complex. Unlock developer productivity and operational efficiency with our GitOps workshop.



## Reference Architectures

Build an open architecture with validated designs used by industry leaders. Weaveworks' designs encapsulate the energy and innovation from the open source community.



## Proactive Support

24/7 SRE support at your fingertips. A dedicated support engineer guides you proactively through troubleshooting, upgrades and maintenance.

**Get Started**

# Building a Development-Ready Kubernetes Platform

By **Anita Buehrle**, Senior Content Lead at Weaveworks

Congratulations on starting your cloud-native journey. Your team has chosen the leading development and deployment framework that provides application portability, agility, and scalability. You began your journey with containers and now you're ready to deploy your container-based application at scale with Kubernetes. But at this point, you're faced with a bewildering array of software vendors, cloud providers, and open source projects that all promise painless, successful Kubernetes deployments.

How do you decide where to go from here? The key to success is a flexible and reproducible cloud-native platform that allows you to quickly adopt these new technologies in your infrastructure and to run workloads anywhere: on premise, in public clouds, or even in a hybrid-cloud environment.

"Cloud-native applications increase business agility and speed. But this requires a new runtime platform and environment for operating cloud-native applications reliably, securely, and at scale." Steve George, COO Weaveworks

Weaveworks Enterprise Kubernetes Platform reduces this complexity through automated configuration management and operations tooling. With GitOps configuration management, teams can define a standard installation of Kubernetes and automate the deployment of new nodes following standard templates. Preconfigured cluster templates let developers and operators define apps and update clusters add-ons with security patches, minimizing the YAML mess.

## GitOps Configuration Management Automation

With GitOps at the center of your operational model, application developers and cluster operators can spin up and manage production ready Kubernetes across environments with ease. GitOps can initiate a cluster patch or a minor version upgrade or add and remove cluster nodes all without having to rebuild your entire cluster from the ground up.

When your entire cluster configuration is stored in Git and managed with GitOps, you can reproduce the cluster in a repeatable and predictable way. This brings advantages when you are building test environments and pipelines, and producing clusters for different teams with the same base configuration, or improving your disaster recovery capability.

## Weaveworks Enterprise Kubernetes Platform

Increase application delivery at enterprise scale. Reduce the time, effort, and errors to create, update, and manage production ready clusters. Preconfigured dashboards allow you to understand clusters, verify and correct updates, and alert on incorrect states.

[Find out more about the Enterprise Kubernetes Platform.](#)



# Effective Kubernetes Deployment Is More Than Choosing a CI/CD Product

By **Bob Reselman**, Principal at CogArtTech

If your company's software stack runs under Kubernetes, an automated continuous integration/continuous deployment (CI/CD) pipeline is not a way to go — it's the only way to go. There's no way any group of people can manually support the hundreds, if not thousands, of active [API resources](#) that make up the typical Kubernetes cluster.

The need for CI/CD is apparent, not only for production release but for all steps that come before it, from dependency management to unit and performance testing at scale.

Yet many companies make the mistake of thinking that establishing an effective CI/CD process requires nothing more than choosing the right product with the right feature set that matches the technology stack for a given enterprise. While such criteria are reasonable, they put the cart before the horse.

The difficulty with CD is that it's process-intensive, particularly for a Kubernetes environment, and a fragmented deployment process is counter-productive to continuous deployment. Continuous deployment requires a release process that is uniform for all version releases all the time.

Without such uniformity, companies cannot release code at the velocities needed in today's business world. Thus, it's no surprise that many companies are reporting an inability to achieve effective CD. Process counts, particularly when it comes to supporting CI/CD under Kubernetes. The good news is that there's no need to reinvent the wheel. The principles for an effective CI/CD were laid out a while ago in the [12 Factor App Methodology](#).

A fragmented deployment process is counter-productive to continuous deployment

## TREND PREDICTIONS

In 2020, we can expect organizations to:

- ▶ Streamline deployment processes.
- ▶ Forego custom deployment processes.
- ▶ Adopt continuous delivery tools and frameworks.

As time has shown, companies that follow these principles have a better chance of realizing an effective CI/CD than those that take a tools-centric approach. In this article, we'll look at two key principles of the Methodology that have particular applications to any CI/CD process: Build, Release, Run and Dev/Prod Parity.

## The Elegance and Benefit of Build, Release, Run

One essential principle of the 12 Factor App Methodology is “Build, Release, Run,” which divides the deployment process into six steps that take place throughout all phases of the SDLC.

### PHASE ONE: BUILD

In the Build phase, a CI/CD management tool such as Jenkins, Travis CI, or Team City does an automated code checkout from a source control management repository that’s shared by all teams involved in the development effort. The code is converted into one, or even many, deployment artifacts (see Figure 1, callouts 1 and 2, below).

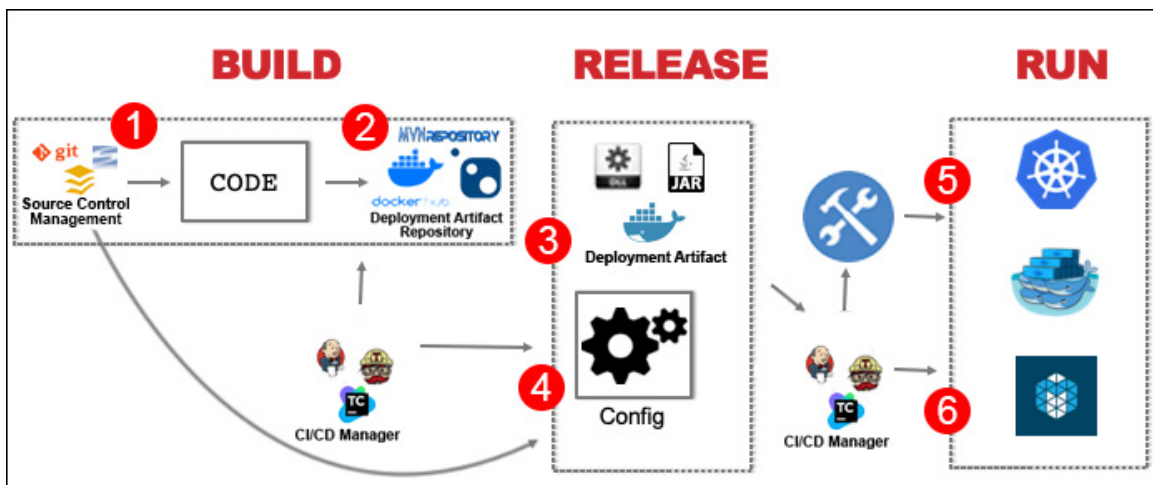


Figure 1: The Build, Release, Run process ensures reliable deployment of software independent of tool choice.

Examples of deployment artifacts include a Docker container, a .NET DLL, or a Java jar file. Once the deployment artifact is created, it’s stored in an artifact repository such as [Docker Hub](#), [NuGet](#), [MVNRepository](#), or any of the multitude of artifact repositories available. Both the deployment artifact(s) and artifact repository depend on the programming language(s) and software development framework(s) supported by the enterprise.

### PHASE TWO: RELEASE

Following the Build phase, during Release, the CI/CD tool aligns the release’s configuration information with the artifacts relevant to a particular release (see Figure 1, callouts 3 and 4, above.) Typically, the configuration information for a release is stored in the common source control management system.

**Example:** Let’s say a particular deployment is updating a feature in an application named MyService. The update to MyService would use a [Docker container image](#) named MyImage:1.1.1, for which MyImage is the image name and 1.1.1 is a tag that describes the version number relevant to the release. The image is stored in [Docker Hub](#).

The CI/CD management system checks out a configuration template that is stored in the source control management system (e.g., a [Kubernetes manifest file](#)). The CI/CD management system adjusts the configuration settings in the template to align with deployment artifact versions relevant to the release. The next step is to promote the configuration and its artifacts into a runtime environment, which is performed during the Run phase.

## PHASE THREE: RUN

Two steps occur during the run phase. First, the CI/CD Manager uses provisioning tools to create the actual runtime environment. The CI/CD Manager can use an infrastructure management tool such as [Terraform](#) to create a set of virtual machines in the cloud. Then they can use another configuration tool, [Ansible](#), for example, to install the necessary runtime binaries.

If the intended runtime environment is a Kubernetes cluster, Ansible will do the work of installing the binaries and setting up the cluster (see Figure 1, callout 5, above).

If the runtime environment exists already, this step is skipped. The CI/CD Manager moves on to the next step, installing the new artifacts into the runtime environment according to the configuration information relevant to the particular release. Typically, this involves applying information in a configuration file, such as a [Kubernetes manifest](#) or a [Puppet](#) configuration file.

Usually, a configuration file references the required deployment artifacts according to the network location of the artifact repository — either on the internet or in a private network. Then these artifacts are downloaded automatically from the network into the physical runtime environment (see Figure 1, callout 6, above).

## The Benefits of Build, Release, Run

Build, Release, Run is an automated process that ensures a high degree of reliability in terms of deployment consistency. It also provides an ongoing audit trail for the entire process, documenting what, where, and when each step occurred during the deployment process.

Humans are wonderfully creative and typically error prone. Thus, the best Kubernetes deployment processes are those that rely upon human ingenuity to design the process and machine automation to execute the multitude of redundant tasks that cannot tolerate mistakes in command line entries.

The best Kubernetes deployment processes are those that rely upon human ingenuity to design the process and machine automation to execute it

Automating deployment using the Build, Release, Run principle allows development teams to focus on designing a robust deployment process while leaving the grunt work to machine automation. This means no more weekends in the war room trying to fix problems created in a manual deployment process that was initiated by others.

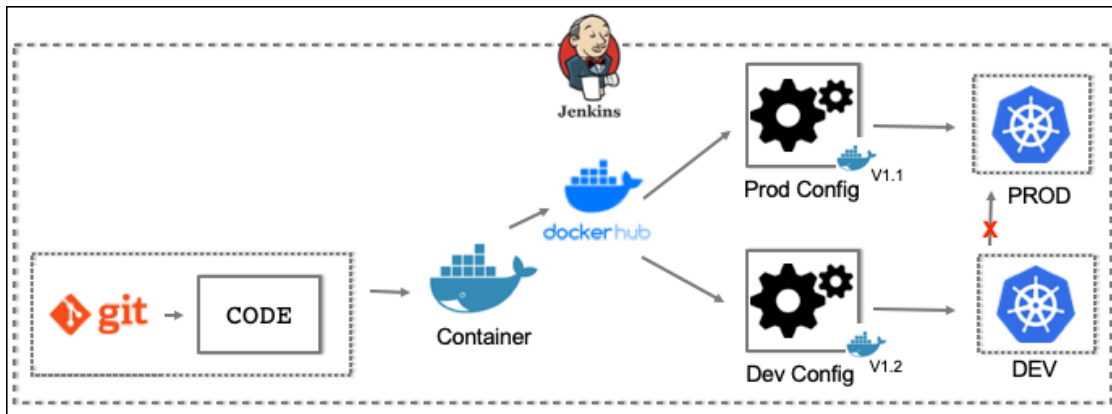
Build, Release, Run saves time and money, provided the process is followed beyond a single deployment path, as detailed by the Dev/Prod Parity principle within the 12 Factor App Methodology.

## Using Dev/Prod Parity to Avoid CI/CD Disasters

Although Build, Release, Run provides a high degree of consistency and efficiency in the deployment process, hazards can still exist if the process does not follow the Dev/Prod Parity principle closely.

When releasing software deployment versions, Dev/Prod Parity maintains that the automation scripts must follow the same deployment process and that the target of the deployment must be known and distinct.

**Example:** Below, Figure 2 shows the Build, Release, Run principle applied to two target Kubernetes environments — DEV and PROD. In the scenario above, PROD is running version 1.1 of a particular application. DEV is running version 1.2.



**Figure 2: TheDev/Prod Parity principle dictates a consistent deployment path to a distinct target.**

Once DEV version 1.2 is production-ready, escalating it into PROD only requires following the Build, Release, Run process again. This time, however, the process is targeting version 1.2 in the PROD environment. The version 1.2 configuration settings and artifacts in DEV should not be directly transferred into PROD. There must be parity in the escalation process. First target DEV. Then if all is well in DEV, run the process again but this time target PROD.

Following the same deployment process to a distinct target ensures consistent determination of the artifact(s) version used. It also provides a clear audit trail for the events in the escalation process. When you “jump” a release from DEV to PROD, you’re losing visibility. You’re hoping things work out for the best. Needless to say, hoping a piece of software works in production is not a good way to do business.

## The Future of Kubernetes Deployments

Tools are important, no doubt, but they are only as good as the processes in which they’re used. Adopting a process-first approach is a significant milestone in modern Kubernetes development. The ephemeral nature of Kubernetes requires that CI/CD processes must be predictable and easy to manage, otherwise even the simplest change in the system runs the risk of creating a significant roadblock at runtime.

As we look to the future, we’re very likely to see more companies forgo custom deployment processes and closely adopt Build, Release, Run and Dev/Prod Parity. Companies such as AWS, Google and Heroku, which are giants in the Kubernetes cloud native space, have led the way. And others will follow.

Following the deployment practices described in Build, Release, Run and Dev/Prod Parity will no longer be a nice thing to do. It will become essential practice for the health and wellbeing of the modern enterprise. 🎲

# How Jenkins X Makes K8s More Valuable

Viktor Farcic, Developer Advocate at CloudBees

CloudBees built Jenkins X, an open source CI/CD offering, solely on top of Kubernetes. Why did we limit ourselves to a single platform? Why do we need Kubernetes, and why does Kubernetes need Jenkins X?

CloudBees wanted to build a new CI/CD platform to meet today's challenges. It needed to be easy to use, scalable and fault tolerant. Building such a platform can take years, unless we take advantage of things that are already out there. One of those things is Kubernetes. It is a platform that provides all the building blocks needed to build a platform.

The Kubernetes "mission" is not to be an end-user tool. Instead, it is a platform designed for developers to build platforms. It allows developers to focus on what matters (functionality), while providing required features that are not differentiators. Being fault-tolerant, highly-available and scalable are not what differentiates one solution from another; they are the expected norm, today. Yet building those features is a daunting and time-consuming endeavor. That's where Kubernetes shines. It solves this and many other needs, allowing software companies to focus on functionality — their true differentiation.

Kubernetes is much more complex to understand and operate than any other traditional platform. That, by itself, scares many people. The time needed to learn basic Kubernetes is huge — and that is only the beginning. Users also need to figure out the ecosystem that surrounds it. We need to pick the right tools. We need to assemble them into a meaningful platform that fulfills our needs.

*How will we build container images? How will we package and deploy applications? Where will we store the information about the cluster and the environments?*

These are many questions that each of us must ask, and trying to answer each of them quickly leads to a myriad of solutions, without obvious answers.

That's where Jenkins X comes in. It simplifies the choices. It provides opinions based on the collective experience of the DevOps community. Jenkins X guides users through the maze of the Kubernetes ecosystem. It provides a solution for managing all the steps in the CI/CD application lifecycle. It makes Kubernetes boring (i.e., easy) instead of overwhelming. It makes Kubernetes useful for all, no matter how deeply (or not) developers understand the intricacies of the new cloud native ecosystem.

Jenkins X needed Kubernetes so that the community could focus on solving new problems. Kubernetes needs Jenkins X to guide people through the complexities of its ecosystem.

## Kubernetes and cloud native development simplified.

Your ideas are going to change the world. Nothing can stand in the way of your turning code and ideas into an impactful, finished application. Not even the complexity that accompanies the brave new world of cloud native app development.

CloudBees helps you leverage the power of Kubernetes for end-to-end application development. We bring order to cloud native chaos by uniting the silos of information and automation, and helping you scale CI/CD, DevOps and Software Delivery Management across your entire enterprise software portfolio, from mainframe to traditional to next-generation cloud native applications. We'll manage the CI/CD automation for you, so you can get busy building stuff that matters.

Develop cloud native apps with the offering that best fits your needs:



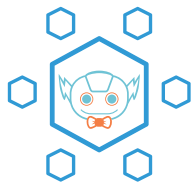
### CloudBees Core

Fully-featured CI/CD for traditional and modern applications. Built on Kubernetes to scale from small groups to multi-national corporations. [Try it for free.](#)



### CloudBees CI/CD powered by Jenkins X

For rapidly building and deploying cloud native applications to Kubernetes. Hosted by CloudBees and powered by Jenkins X. [Learn more and join the Preview Experience.](#)

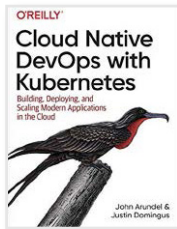


### CloudBees Jenkins X Distribution

For building cloud native applications, when open source satisfies the need. [Sign up](#) to get the battle-tested distribution of Jenkins X from CloudBees!

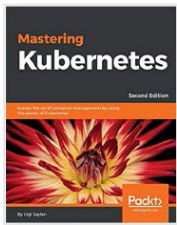
# Diving Deeper Into Kubernetes and CI/CD

## Books



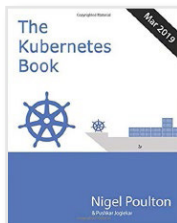
### Cloud Native DevOps With Kubernetes

Learn how the Kubernetes ecosystem can help you create reliable applications with scalable infrastructure.



### Mastering Kubernetes

This book covers concepts like the advantages and disadvantages of running K8s on various cloud providers versus bare metal, monitoring and troubleshooting clusters, and customizing K8s.



### The Kubernetes Book

Get a thorough introduction to Kubernetes, starting from the basics. This book is one of the most popular guides to Kubernetes for beginners.

## Refcards

**Monitoring Kubernetes** This Refcard outlines common challenges in monitoring Kubernetes, detailing the core components of the monitoring tool Prometheus.

**Advanced Kubernetes** This Refcard aims to deliver quickly accessible information for operators using any Kubernetes product.

**Securing Your Kubernetes Deployment** This Refcard will teach you the essentials of security in Kubernetes, addressing topics like container network access, user authorization, service token

## Zones

**Cloud** The Cloud Zone covers the host of providers and utilities that make cloud computing possible and push the limits (and savings) with which we can deploy, store, and host applications in a flexible manner. This Zone focuses on PaaS, infrastructures, containerization, security, scalability, and hosting servers.

**Integration** The Integration Zone focuses on communication architectures, message brokers, enterprise applications, ESBs, integration protocols, web services, service-oriented architecture (SOA), message-oriented middleware (MOM), and API management.

**Open Source** The Open Source Zone offers practical advice about transitioning from closed to open projects, creating enforceable codes of conduct, and making your first OSS contributions. This Zone encourages you to adopt an open-source mentality and shape the way open collaboration works.

**DevOps** DevOps is a cultural movement, supported by exciting new tools, that aims to encourage close cooperation within cross-disciplinary teams of developers and IT operations. This Zone is your hot spot for news and resources about continuous delivery, Puppet, Chef, Jenkins, and more.

## Podcasts

**PodCTL** Produced by Red Hat OpenShift, this podcast covers everything related to enterprise Kubernetes and OpenShift, from in-depth discussions on Operators to conference recaps.

**Deloitte on Cloud** This episode of the Deloitte on Cloud podcast dives into a few ways that organizations can use Kubernetes to standardize processes around cloud migration.

**Kubernetes Podcast from Google** Considering that Google produces it (and that Google also created Kubernetes in 2014), you might call this podcast a classic. Enjoy weekly interviews with prominent tech folks who work with K8s.



## INTRODUCING THE

# Cloud Zone

Container technologies have exploded in popularity, leading to diverse use cases and new and unexpected challenges. Developers are seeking best practices for container performance monitoring, data security, and more.

Keep a pulse on the industry with topics such as:

- Testing with containers
- Container performance monitoring
- Keeping containers simple
- Deploying containers in your organization

Visit the Zone



TUTORIALS



CASE STUDIES



BEST PRACTICES



CODE SNIPPETS