ENTERPRISE DEVOPS:

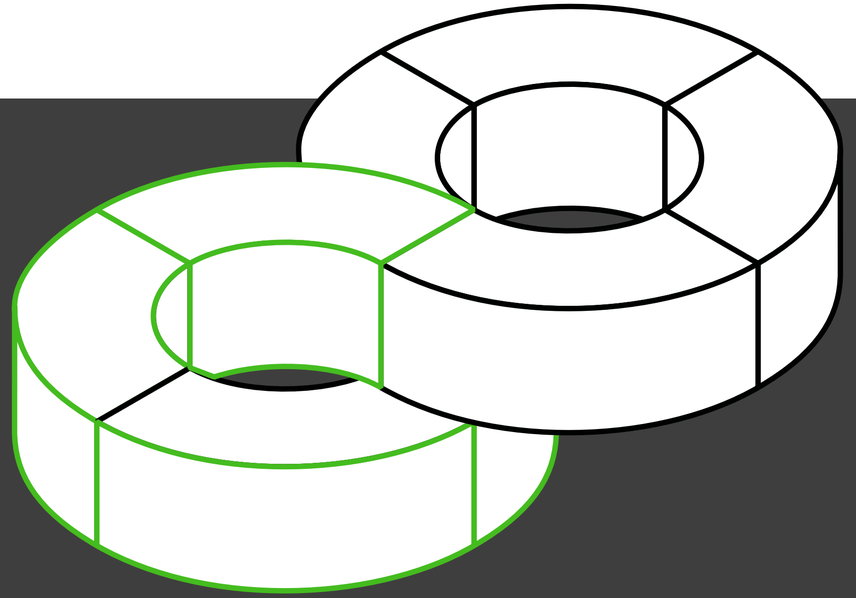
5 KEYS TO SUCCESS WITH DEVOPS AT SCALE

After getting a taste of DevOps' benefits, enterprises naturally seek to widen its adoption. However, the tooling and processes that work for small-scale use cases often fall short when teams try to scale DevOps efforts. You must support all your different teams, toolsets, applications, processes, workflows, release cycles and pipelines — both legacy and cloud native. Otherwise, you can end up with a haphazard mishmash of automation silos or DevOps tools and processes, with widely varying degrees of quality, security, velocity, and — ultimately — success.

And success is critical, since software now underpins and facilitates virtually all business processes. DevOps — with its iterative, collaborative approach to application development and delivery — sits at the center of this new value chain. Implementing it at scale requires the right structure, processes and tools.

In this ebook, we'll outline the five key principles for scaling DevOps effectively across your organization.

- Central management of end-to-end DevOps ▶
- Secured from the start ▶
- Future proof with cloud native ▶
- Pipelines as code ▶
- Think global, act local ▶



FIVE KEY PRINCIPLES FOR ENTERPRISE DEVOPS SUCCESS

A pioneer and leader in enterprise DevOps, JFrog knows what it takes to scale DevOps effectively across an organization. With more than 6,000 customers, including many of the world's largest enterprises, across all verticals, we know a thing or two about DevOps at scale.

We've partnered with these huge organizations as they embrace DevOps — and now, cloud-native modernization — to deliver high-quality software frequently and at scale. Here are five key principles our customers have put into practice when scaling DevOps in the enterprise.

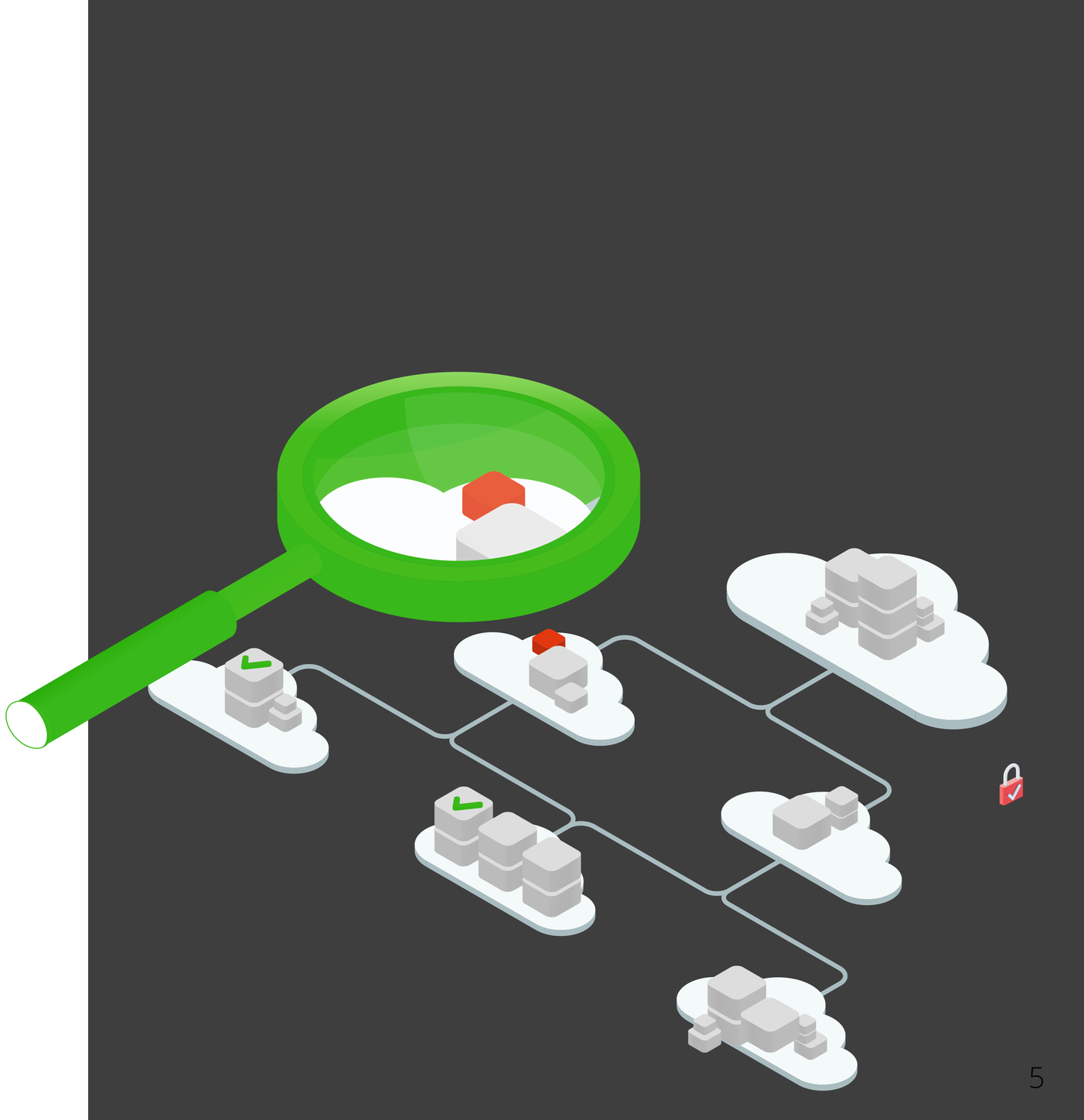
CENTRAL MANAGEMENT OF END-TO-END DEVOPS PROCESSES AND THEIR OUTPUT

An end-to-end central DevOps platform should include management of binaries, container images, CI/CD pipelines, security and compliance, and software distribution to last-mile deployments across runtime environments, edges, and ”things.” Currently, many CI/CD tools let you manage either the automation processes or their outcomes (builds and binaries), but not both, and do not support all types of binaries and technologies either.

It’s essential to have a DevOps platform that allows this central artifact management from one solution. A unified experience provides clear visibility and a single source of truth for your entire SDLC and software assets. This speeds up software delivery, improves code quality, security, and governance, and lets you take action and trigger automated processes over your dependency downloads, repositories, deployments, builds, pipelines and releases.

By managing both delivery processes, and delivery assets and outputs from a single, end-to-end DevOps solution, and not having to “context switch” between disparate tools, you’ll be able to:

- **Ensure consistency and traceability** of all your artifacts throughout the software lifecycle, as they flow through your pipelines from development to production.
- **Have a universal repository and single source of truth** for different types of binaries, container images, environments, processes, point tools and more.
- **Manage security and compliance**, across all tools, processes, artifacts and repositories, including third party ones, to ensure governance throughout the organization.
- **Gain full visibility across your entire pipeline and organization.** No more siloed processes or snowflake configurations.



SECURED FROM THE START — WITH BUILT-IN DEVSECOPS AND 'SHIFT LEFT' CAPABILITIES

Enterprise DevOps must incorporate security and compliance checks across the entire software lifecycle — from development to deployment to production. Anywhere between 60 percent and 80 percent of application code is made up of third-party open source components. Using OSS dependencies as part of our applications greatly accelerates developer time-to-value and productivity by re-using existing components available in the ecosystem. However, these dependencies often contain security vulnerabilities and misconfigurations, license compliance issues, or other governance risks.

Enterprises today must manage unprecedented software growth. They're producing more and more artifacts and applications — all using OSS components. In addition, the continued adoption of microservices-based, cloud-native applications further widens the attack surface due to the myriad of connected services, and to the fact that each container image can contain dozens of layers with hundreds of OSS dependencies.

This scenario gets compounded by the pressing need to patch these vulnerabilities more and more quickly, to try and stay a step ahead of the bad guys, and by the fact that for every 200 developers, the typical enterprise has only one security analyst.

Trying to tack on security testing at the end of the software development lifecycle creates a bottleneck and slows down delivery. Because security testing and scanning can be a pain, they tend to be pushed towards the end of the process, when they become a bottleneck.

THE SOLUTION?

- Security and compliance must be first-class citizens, enabled by default, as an integral part of your chosen DevOps platform. No more tool/context switching, or remembering to run a test or initiate a scan.
- Security must be universal, meaning it supports all types of binaries, including cloud-native artifacts such as container images, and is tightly integrated throughout the artifact lifecycle and CI/CD pipeline.
- Deep recursive, scanning is required across all dependencies, including container images — from the application layer through the operating system (OS) layer. Beware of tools that can't automatically scan and identify the granular dependency tree all the way to the OS level without heavy lifting from developers.
- Scanning must be continuous and automated — at the DB level — across all repositories and production instances, and not just triggered by a pipeline. Why? Many dependencies may already be in 'the system' and across existing applications/builds. DB-level scanning ensures that you're alerted to newly-disclosed vulnerabilities and can patch them — even in older applications or re-usable packages that aren't part of an 'active' current CI/CD delivery pipeline. Beyond DB-level scanning, you should be able to trigger security scanning and security gates as part of your CI/CD automation as another validation stage.
- Enable "shift left." Your application security solution must have integrations with your IDE, so that you can identify and patch vulnerabilities as early as possible in the process — while you're developing the application. The sooner you catch them the cheaper it is to fix them — rather than waiting until the code is ready to be built, let alone shipped to production.
- Set up governance rules for security and compliance policies, with the flexibility to adjust the scope of enforcement and the follow-up actions depending on the software development stage and your company's DevSecOps rollout plan. Your policies may be more or less strict depending on your teams, applications, use cases, DevSecOps maturation, and risk.

FUTURE PROOF WITH CLOUD NATIVE: THE MODERNIZATION IMPERATIVE

As you modernize your applications to take advantage of modern cloud-native patterns and technologies like Kubernetes, remember that you still need to be able to update your legacy applications — and that not all your applications are containerized microservices (yet).

An enterprise DevOps platform should support both cloud-native and legacy applications, so that you can manage the entire lifecycle for either type of application. This includes their binaries, CI/CD processes, security scanning, and more — without having to switch to a separate tool.

Similarly, your DevOps platform has to be hybrid and multi-cloud, so that you can both consume it and use it to manage delivery pipelines across mixed environments of on-prem, private and public cloud, multi-cloud, and edge infrastructure.

PIPELINES AS CODE

The ability to define Pipelines-as-Code increases developer productivity and helps to scale DevOps efforts. You can store pipeline definitions in your source control, and this makes them shareable (so you can collaborate with your team), versionable, reusable, auditable, and reproducible.

This eliminates redundant work among developers (so that teams don't need to re-invent the wheel to create their CI/CD automation), and also allows you to standardize and use vetted automation processes across the organization.

Furthermore, it also enables your teams to continuously evolve your delivery pipeline like a product — hardening, improving, and enhancing your processes with each consequent version.

Let's look at the benefits and some best practices for Pipelines-as-Code in more detail:

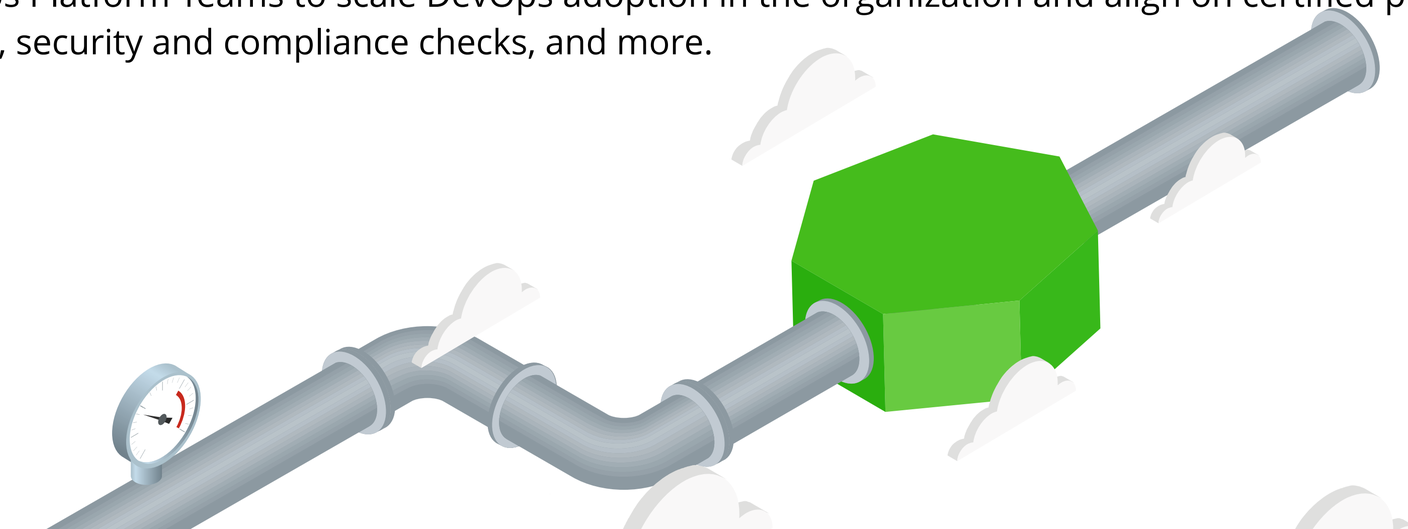
- Re-use and standardize across teams: Pipelines-as-Code let you model and define the building blocks that DevOps teams need to standardize for their workflows and processes across the organization. This has two main benefits:
  - Increased speed and developer productivity through the re-use and sharing of automation building blocks — including objects, processes, secrets, resources, configurations, policies, security tests, conditional execution, and more.
  - Improved quality and governance by standardizing on approved, hardened, processes that are consistent across the organization. These efforts eliminate snowflake configurations, automation silos, disparate processes and error-prone scripts that can introduce drift and quality concerns.

- Use parameters in your pipelines' code. Remember that in order to allow different teams to leverage the same Pipelines-as-Code, these building blocks should be parameterized so that every team can dynamically call their appropriate inputs (such as secrets, resources, environment configurations parameters, etc...)

- Use declarative automation. Pipelines-as-Code building blocks should be preferably declarative, so they can be more easily defined and extended, avoiding "spaghetti", error-prone scripting or heavy lifting. Declarative pipelines are also more suitable for cloud-native environments such as Kubernetes.

- Modernize legacy workflows. Since large organizations likely have many legacy scripts and technical debt, you need a flexible DevOps platform to grandfather legacy CI/CD technologies (such as old build tools) and custom scripts (such as your Perl/Bash automation.) These need to be supported in your Pipelines-as-Code scripts as well, with standard steps or integrations that call external systems or custom code. This way, these legacy scripts can be triggered and orchestrated by a modern CI/CD solution, for end-to-end automation, until you have time to refactor or modernize them.

- Empower DevOps Platform Teams to scale DevOps adoption in the organization and align on certified processes, promotion gates, security and compliance checks, and more.



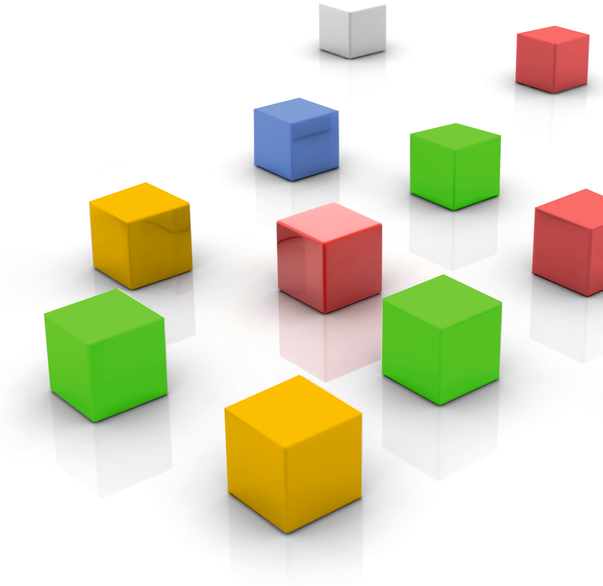
THINK GLOBAL, ACT LOCAL, AND A WORD ON PLATFORM TEAMS

System-level thinking is a key tenet of DevOps. It means thinking systematically and holistically about your entire DevOps ecosystem and your software delivery practices — encompassing your teams, culture, value streams, processes, and technology and architecture choices.

To scale DevOps in the enterprise, you need to take into consideration your organization-wide processes and tools, while allowing for flexibility, agility, and freedom of choice to empower developers and enable team autonomy and speed. Your DevOps platform must allow you to “think global, act local.” For example, you should be able to enforce different security policies, or to integrate specific best-of-breed point tools, for certain use cases.

This approach reduces drift and prevents you from ending up with snowflake workflows or configurations, while also ensuring compliance as well as flexibility to support any tools or process in your DevOps ecosystem — for both legacy and cloud-native app delivery. This is where DevOps platform teams come in!

Application delivery is critical for organizations’ competitive advantage today. With scale and growth in deployment frequency, many enterprises realize that it is impractical to continue to manage delivery with ad-hoc/siloed solutions for different teams.



DevOps Platform Teams (also known as Platform Ops or Delivery Services) are responsible for selecting the tools and delivery infrastructure to enable DevOps-as-a-Service for all internal teams — with shared tools, standardized processes, and consistent governance. They develop, harden and certify the automation building blocks that teams can consume from a central repo. They also further enhance these processes as new requirements emerge. As such, they help improve speed, productivity, reliability, and TCO, by:

- Leveraging Pipelines-as-Code to scale and accelerate DevOps adoption in the organization, while ensuring governance, compliance and auditability.

- Managing a catalog of self-service node pulls and secrets rotations to enable developers to provision environments with consistent configurations, in a secure way.

- Enabling central access to a shared repository of "certified" packages, to be consumed by developers. The platform team hardens and secures OSS dependencies, and certifying them as compliant with the organization's governance policies. This eliminates re-work and enables re-use and standardization around shared, secured, artifacts.

- Managing the sharing of binaries between teams/stages of the pipeline and software distribution to last-mile deployment in a centralized way — to optimize for throughput, network utilization, and also ensuring security and governance. Managing Access Controls (RBAC) in a centralized way to ensure compliance and auditability — with the ability to scope the level of access to all approved components including repositories, artifacts, tools, pipeline processes, approvals, environments, and more.

The JFrog Platform is an end-to-end DevOps solution used by some of the largest organizations in the world to speed up application delivery while improving quality and security.

The Platform helps you streamline, secure and scale your SDLC end-to-end, spanning binaries management, container images, CI/CD, application security, and distribution.

Start your free cloud trial to see the JFrog Platform for yourself!

www.jfrog.com

www.facebook.com/artifrog/

www.twitter.com/jfrog

www.linkedin.com/company/jfrog-ltd

The Liquid Software Company