SIX OBSTACLES TO SUCCESSFUL DEVOPS

WHY A BINARY REPOSITORY MANAGER IS ESSENTIAL
Every company is now a software company, even yours.

In today’s enterprises, software is your company’s everyday face, whether through the desktop, the cloud, or a mobile device, to all parts of the globe.

Cars are computers on wheels. Thermostats are data terminals. Banks live in your phone.

In this new world, software updates serve customer’s demands. Each one you deliver is your opportunity to renew -- or, if botched, destroy -- their trust. How can you make every update top-notch at top speed?

Repository Management to the Rescue

That’s why DevOps is important for your organization. When you speed the delivery of quality software, customers get what they clamor for, and you can respond rapidly to shifts in market demand.

DevOps speeds the delivery of quality software by reducing friction as it moves between stages and stakeholders for testing, evaluation, and release. Identifying and resolving these pain points forges successful DevOps.

A binary repository manager can help keep those builds moving. It stores your artifacts, but keeps information about them too, reducing uncertainty and enabling your automation tools to run freely and fast.

These six obstacles to speedy release are typically faced in software delivery pipelines. A binary repository manager can help resolve them, for swift, constant updates from code to customers.
Obstacle 1: X
Knowing About Your Builds

Your developer teams can produce many builds every day. How will you keep track of them all?

Without a versatile solution, you may know which build is most current, but not which build is best. Nor can you reliably trace through their history, or where their many parts came from.

When your builds fail, can you identify and roll back the problem parts? You’ll want to pinpoint which builds are having problems and where they occur in the build process so you or your developers can quickly provide a fix.

Resolution: ✓
A Common System of Record

A central home for your builds is a single source of truth for all artifacts moving through your pipeline. Organizing and versioning your build outcomes into repositories means you can readily find the best-functioning, most current build.

A repository manager that tracks where artifacts are used and their prior versions provides a rich set of data that helps you trace everything to its source or ancestor. You can quickly see the differences from one version to the next, gain visibility into how each was made, and find insights that help you fix your builds when they go wrong.
Obstacle 2: **X**

**Manual Processes**

Every place in your pipeline that requires a human to step in opens a risk. Individual signoffs add delays. Redundant rebuilds for release introduce uncertainty. Scripts for tool management or build deployment that have to be manually changed, maintained, and executed cost time and are prone to mistakes.

Any of these costly blocks slow getting the right code to the end user.

**Resolution: ✔**

**Automation and Process Management**

A central binary manager that holds your builds and artifacts offers convenience. But one that collects intelligence about them as well gives you power. The more you know about your binaries, the better you can automate, and enable your build production tools to make smart decisions that can unify and speed your delivery pipeline all the way to deployment.

Your repositories solution will need to provide your build tools with a rich, versatile interface for queries and commands, so they can do their work without your intervention. If it uses a standard, platform-independent interchange technology, such as REST APIs, you’ll be free to choose whichever CI server fits you best.

Once you can automate your pipeline, you can be better assured that every build released into production adhered to the same process, and conforms to a common standard.
Obstacle 3: Wild Dependencies

Developers draw dependencies for many languages and technologies, each with its own requirements and interface. How will you manage them all?

These external resources can change at any time, and quality control ranges from fierce oversight to none at all. How can you be certain of what’s in every build? How can you reliably reproduce one? What harmful changes might sneak in?

What’s more, your build processes can’t run any faster than your link to those remote resources. A heavy load slows builds down; an outage forces your otherwise sound builds to fail.

Resolution: Dependency Management

Bring your dependencies under control with local repositories that proxy the remote resources where dependencies are stored. With a locally-held cache of those dependencies, the version you need is always available to complete your builds, and to do so at top speed.

Even better, once your binary management system governs your dependencies, it can maintain the same kind of information about them as other artifacts. Track their history and usage, and always know which version of a dependency is employed in every build.
Obstacle 4: Moving Builds Through Your Pipeline

Many pipelines require a fresh complete or partial build of the code at each staging transition of testing, validation, and release. Each new build takes additional time, and might require each stakeholder to manually evaluate and launch. Even worse, as developers continue to change shared code, each rebuild introduces uncertainties that require repeating the same quality checks in each stage.

Once a build has passed through checkpoints, how do you physically advance it to the next stage? A manual process to promote a build to the next staging location is prone to errors. And you’ll need a way to communicate the status of that build to the entire team as it transitions through your pipeline.

Resolution: Dependency Management

Bring your dependencies under control with local repositories that proxy the remote resources where dependencies are stored. With a locally-held cache of those dependencies, the version you need is always available to complete your builds, and to do so at top speed.

Even better, once your binary management system governs your dependencies, it can maintain the same kind of information about them as other artifacts. Track their history and usage, and always know which version of a dependency is employed in every build.
Obstacle 5: Meeting Growing Demands

You need to operate big today, and bigger tomorrow. A heavy load from many line-of-business teams can slow down your entire development pipeline, while any single point of failure in your infrastructure can be catastrophic.

Geographically distributed teams need to be able to always reach the same pipeline resources at the same speed. And any interruption of service to update or to upgrade capacity wastes vital production time.

Resolution: Enterprise Readiness

Enterprise-ready solutions provide the flexibility and muscle that can adapt as you grow.

A repository manager that can work in the cloud can help infinitely scale costs of storage and computing with your needs. The more cloud providers your binary management tool can work with, the greater control you’ll have. And a SaaS subscription option ensures your resources are always available and up-to-date.

A high availability, active/active clustering configuration can assure repository responsiveness under load. That redundancy also provides failover for disaster recovery, and enables expansion and maintenance with zero downtime.

A binary manager that supports multi-site replication can provide the regional proximity distributed teams need to share the resources of their pipelines speedily across the globe.
Obstacle 6: **X**  
Cost of Adapting to Change

Reaching all your customers means developing in many languages, across many runtimes. One department may write code for the cloud in Go, another for mobile devices in Java. Yet each technology has its own requirements and supporting tools.

What kind of infrastructure will you use for DevOps? Today it might make the most sense to run securely in your own data center. Tomorrow you may need the flexibility of the cloud, or to combine them for the benefits of each. You’ll want to be free to choose the vendors that fit your requirements best, and to change nimbly when your needs shift.

**Resolution: 🔄**  
**A Universal Solution**

A universal binary repository manager automates your delivery pipeline no matter what language used or platform run on. Control through REST APIs enable working with the tools you already use.

As the core of your DevOps system, your binary repository manager must function equally well in the cloud as on your own servers on-premises. A solution that can easily promote builds across those environments enables DevOps in a powerful hybrid cloud. And integrated support for all major providers, such as AWS, Google Cloud and Azure, empowers a multicloud strategy that prevents vendor lock-in.

You should be able to choose how to pay for it, too. Whether you need the fixed licensing costs or a flexible SaaS subscription, a solution available as both will free you to build the systems you use now and in the future.
Next Steps

A fully featured binary repository manager will help you automate your software delivery pipeline and guide you to a new way of work. It can provide you control over and insight into your processes so you can resolve problems as they arise and continuously improve your methods. When robustly designed, your repository manager can flexibly adapt to the special needs of your organization.

Just as important, you’ll need a solutions provider that can be a good partner in your DevOps journey. They should be knowledgeable about different approaches and industry trends. Most of all, you’ll want a partner that’s invested in your success.

About JFrog

JFrog is the creator of Artifactory, the heart of the end-to-end Universal DevOps platform for automating, managing, securing, distributing, and monitoring all types of binaries.

Artifactory is trusted by more than 5,500 customers, including 70% of the Fortune 100. The world's top brands, such as Amazon, Facebook, Google, Netflix, Uber, VMware, and Spotify depend on JFrog to manage their binaries for their mission-critical applications.