Valet or Self-Park? Keys to Optimizing CI for Cloud Environments

Discussing the tradeoffs of convenience, control and performance when considering continuous integration deployment options.

Jason English

Principal Analyst, Intellyx

March 2021

Remember valet parking?

Back when we used to go to crowded restaurants and events, we had the choice of either handing our car keys over to a friendly valet who will park and return the car to us for a fee -- or struggling to locate our own spot and hoofing it back and forth to the door.

Almost every modern business model is defined by its digital presence, driven by software, and designed with cloud computing in mind. Thus, the software pipeline, including developed code and Infrastructure-as-Code (IaC) as utilized in a continuous integration (CI) solution, is in effect the company van for taking development to the cloud.

CI is the transportation that drives us where we need our digital business to go – from concept to customer.

Let's say we've already decided ***where*** to go for cloud resources -- picking from our own private cloud-like resources, and Azure, Google Cloud or AWS for instance. Now ***how*** should we ideally park our CI when we get there?

Should we hand the CI keys over to a valet or full-service SaaS provider? Or should we self-park, and manage an 'in-house' implementation of the pipeline in our cloud infrastructure of choice? This paper will explore the tradeoffs and benefits of both approaches.

Is cloud driving CI to extremes?

There's no longer a question about ***if*** we should move any business application to cloud, but ***how*** we should do it.

Depending on which of the many recent cloud industry surveys you trust, as many as 83 percent of companies are now running application workloads in public cloud infrastructure (AWS, Google, Azure, others) and as many as 69 percent are additionally opting for some combination of hybrid public/private cloud infrastructures.

As IT leadership strategically decides to modernize and migrate the enterprise's application estate to run in cloud infrastructure, teams must make shared responsibility model considerations about where specific applications should reside, store data and run workloads. In this sense, your development teams' CI pipeline is just another application.

So should you turn the CI keys over to a full SaaS provider, or keep it in house?

There are benefits and pitfalls to each of these two opposite approaches:

Full SaaS CI Pipeline

This approach takes advantage of the natural desire of many companies to leverage managed services for every other aspect of their company's business (CRM, HR apps, etc.) as much as possible.

Benefits of Full SaaS CI include:

- **Simplified procurement structure** from a per-user perspective. The complexity of cloud storage and compute, plus any software licensing and support costs, are encapsulated in the per-seat price.
- **Easier horizontal scaling** to rapidly add more users to teams with ready-to-go CI environments already configured
- **Built-in support** means developers will get automatic CI software updates, and they can call on the SaaS vendor's support organization rather than opening a ticket with IT Ops or the cloud provider org.
- **Easy for CSPs / partners to manage** multiple client accounts and pass along the CI functionality through a turnkey service partner and development vendor

interface.

Potential Full SaaS CI pitfalls include:

- **Reusability of build automation** may not be as simple as expected, if certain settings or parameters are locked in by the provider.
- **Optimal cloud resource utilization** may not be a top priority of the SaaS provider in orchestrating, juicing the performance, and closing a highly variable flow of jobs.
- **Multi-layered pricing** offers a limited number of standardized resource levels and OS options to developers with less opportunity for usage-based or custom configuration
- **Security dependency** on the vendor's best practices may not sync up with the development org's concerns over where source code goes, and the privacy of queued and running build jobs.

Self-Hosted CI Pipeline

In this approach, the vendor or community simply provides the CI tool, and leaves server software installation, management and orchestration up to the consuming development organization.

Self-Hosted CI Benefits include:

- **Potentially cheaper procurement** of CI packages, low or 'free' license costs or open source tools with or without vendor or minimal support from cloud IaaS provider
- **Supports a do-it-yourself ethic** since IT and dev teams can specify both orchestration and job manifests which can be completely tailored to the requirements of the company
- **Maintain internal security settings** to protect code, project data and secrets within running CI workloads according to the company's governance needs.

Self-Hosted CI potential pitfalls include:

- **Higher potential cloud overhead costs** as instances may need to be pre-allocated, and capacity may not be released immediately upon consumption. Additional effort should be put into cost containment procedures.
- **Higher potential labor costs** as teams manage more of the CI process, write/maintain environment configuration YAML code, report results and collate documentation for each project. Custom-built assets may be less reusable.
- **Security management complexity** across several different pipelines can be problematic – it basically makes your developers/architects change hats as sysadmins to control user/group access. Furthermore, you may not want to keep secrets on your self-managed build servers.

Measuring toward a happy medium

We started by exploring the pros and cons of two diametrically opposed approaches to managing CI in cloud. But as it turns out, we're almost always seeking a middle ground -- the ability to pick and choose a 'goldilocks' option that isn't too hot, or too cold for the specific criteria we want to focus on.

The total cost of ownership (TCO) and ROI figures of a CI transformation seldom tell the whole story, as the overall efficiency of software delivery has a much greater impact on the agility and performance of a digital business than anything we can measure at the pipeline level.

Time to value (TTV) measures how easy it is to transition the existing development suite to its next cloud-based life. The ability to reuse existing repositories, build configurations and test suites has a big impact on productive start time, as does reducing the learning curve for unfamiliar tools and languages.

If the CI transition process requires more effort than sticking with the status quo for a few weeks with little additional impact to show for it, the project may be doomed from the outset. After all, if it isn't far more convenient, performant, secure or cost-efficient than what is working today -- it's probably not worth changing!

One early web commerce vendor, Shopify, encountered just this quandary, as they had to grow their development organization to a globally distributed team responsible for delivering new features, updates and bug fixes to a production environment encompassing dozens of unique products serving half a million discrete customers.

The company recognized that the 40 minutes of wait time it was taking for each build job with its existing CI process was starting to cascade into even more delays due to interdependency and a stated goal of thousands of small releases a day.

Using Buildkite's hybrid CI model, the company centrally orchestrated as many as 10,000 concurrent jobs from its SaaS management interface, while using its lightweight agents to auto-scale Google Cloud-based workloads toward its stated goal of completing build jobs in less than 10 minutes.

Sharing control in a hybrid CI model

A hybrid CI model allows the consuming development organization to divide responsibility for how a CI pipeline is managed in the same way the broader IT organization applies the IaaS cloud vendors' shared responsibility models.

We want to offload the capabilities that are most efficiently handled and automated by SaaS CI services in the cloud, without losing grasp of how our development and operations teams know the software builds themselves should be declared, validated and delivered to target infrastructure, according to their unique application requirements.

This natural division of labor gives our teams total control of the parts they care most about -- coding, building, testing and ultimately running and monitoring applications in cloud or hybrid IT infrastructure.

To accomplish all of the above at cloud scale, the CI service should act as the valet, and "automate all the things" it is good at. Developers should never need to handhold the orchestration of the build service across thousands of jobs, including authorization, hosting, scaling, and integration steps.

Even an expert software configuration vendor like Chef recommends hybrid CI pipelines to their own customers, as developers can define dynamic pipeline steps within their own source code, then layer on their own custom functionality. At build time, Buildkite orchestrates the builds and dispatches lightweight agents to scale workloads on the customer's service infrastructure of choice.

The open source Elastic CI Stack for AWS solution provides an example of this model -- developers can declare their own desired build infrastructure as code, then turn the keys over to a valet which utilizes AWS's innate auto-scaling capabilities for massively fast and parallel CI build execution. Buildkite-defined defaults make these capabilities accessible to anyone, and configurable parameters give DevOps practitioners a way to customize to taste.

Performance vs. reliability concerns

We often think of performance in IT operational terms of 'speeds and feeds,' but in CI build environments, reliability is the true benchmark of performance.

After all, your team can automate thousands of builds in the cloud, but if any aspect of the process is compromised, you might have to start over with a whole new batch. Your effective utilization of cloud infrastructure drops below 0 percent, and teams are saddled with rework.

Test builds in particular have to work flawlessly every time, both for positive confirmation that the delivered configuration passes as specified, and for reporting negative test results in order to fail builds and trigger exception workflows.

Leading customer service and support application vendor Intercom was dealing with just such a problem in testing two primary applications hosted in its own Amazon EC2 instance: a Ruby-on-rails monolith with most of the firm's back-end logic, and a Javascript customer front-end app.

With as many as 150 agile deployments targeted a day, testing became a major headache. Their two self-hosted CI pipelines for test builds were interrupted with resource outages, and up to 50% of the tests would fail due to environment failures -- rather than failures of the application or test itself.

By orchestrating tests using Buildkite agents on their own performance-tuned cloud infrastructure, Intercom was able to auto-scale up to tens of thousands of builds a day, while the tests themselves could rapidly retry or report back to development teams if a real problem emerged.

We often think of performance in IT operational terms of 'speeds and feeds,' but in CI build environments, reliability is the true benchmark of performance.

Wrapping up security risk

Clearly for ease of use, we want the DevSecOps team to inherit the company's SSO and authorization procedures, while declaring who can view, modify or execute software build pipelines.

In a software-driven business, code delivery is not only a transport for ongoing revenue and operations, it encapsulates all of the innovative ideas and intellectual property the company delivers to market. Who wants to give away the keys to that 'shiny new car' to just anyone?

Buildkite's hybrid CI model allows source code to stay behind the firewall in the GitHub or Bitbucket account, bringing it into the mix at build time using self-hosted agents that can be activated upon a pull request. The agent reports build results back to Buildkite, and your code never leaves your environments.

The same goes for secrets, which will go on to hold sensitive customer information in production. Developers can link up to a leading secrets storage service like AWS Secrets Manager or Hashicorp Vault, to test the validity of integrations without ever directly handling private data or sharing secrets with an additional party.

The Intellyx Take

Technology leaders with big plans for delivering software in the cloud can often find themselves circling the CI parking lot -- trying to decide between the self-managed pipelines they outgrew, and newer SaaS options that offer operational uncertainty.

Maybe the ideal hybrid CI process should feel like parking your own car, but having the valet bring the parking spot to the entrance, if that's even possible.

In selecting a strategy for moving CI to cloud, don't just prioritize cloud cost concerns, or the 'speeds and feeds' of raw performance, but where your team's valuable time should best be spent.

You want your team to be the experts at knowing how to build your own app -- not building a better CI mousetrap. And you want your cloud provider to do what it's best at -- easy provisioning and elastic scaling to provide highly performant, and secure resilient infrastructure for fast, reliable build workloads.

About the Author

Jason "JE" English (@bluefug) is Principal Analyst and CMO at **Intellyx**, a boutique analyst firm covering digital transformation. His writing is focused on how agile collaboration between customers, partners and employees can accelerate innovation.

In addition to several leadership roles in supply chain, interactive and cloud computing companies, Jason led marketing efforts for the development, testing and virtualization software company ITKO, from its bootstrap startup days, through a successful acquisition by CA in 2011. JE co-authored the book *Service Virtualization: Reality is Overrated* to capture the then-novel practice of test environment simulation for Agile development, and more than 60 thousand copies are in circulation today.