ambassador

Platforms, People, & Prosecco

Best Practices for Adopting Kubernetes

DRINK YOUR OWN

EST. 2014

CHAMPAGNE

BOSTON, MA

Toast to ‘drinking your own champagne’ and using our own tools for platform engineering and better developer experiences.

Whatever you call it – ‘dogfooding’ or the slightly upmarket ‘**drinking your own champagne**’ – using your own tools can lead to both engineering and operational excellence internally while delivering better products externally.

Cloud-native teams adopting and using Kubernetes and CNCF technologies face a number of challenges: potential architectural complexity, tool sprawl and the paralysis of choice, not always having a clear overview of what services are running or what’s going wrong when things fail, and other critical, but broader, considerations like security. Companies spring up every day with new tools promising to simplify the increasingly tangled cloud-native space, either with a single piece of software or a concept like a developer control plane.

Using our own products:

Getting tipsy on our own drinks

Ambassador Labs is no different. We've developed solutions, such as the Kubernetes-native Ambassador Edge Stack API gateway and Telepresence, which lets developers and teams access remote environments locally for code, testing and debugging. What sets Ambassador apart is that we consume our own products every day as part of our normal work.

Using our own products not only makes us more productive, it helps us experience our products as consumers – discovering, for example, how a specific feature or function might work for real-world users in their own environments. Each time we take a sip of our champagne, we understand our users and their pains a bit better, infusing the product with what cloud luminary Kelsey Hightower calls "empathetic engineering", i.e., by using our own products, we can design software that is more usable and reliable for users.

Ultimately drinking more of our own champagne has a dual set of benefits: internally, using our own tools enables us to look out for, and eliminate, friction points in our application, and by extension, boost developer efficiency. Externally, our tools may deliver similar benefits for organizations adopting them but can also enable faster feature releases and better products for end users.

ambassador EDGE STACK

ambassador TELEPRESENCE

Internal benefits:

Lift a glass to developer productivity, faster feedback loops and operational excellence

How can you tell whether your champagne is any good? At Ambassador, we measure the impact of using our internal tools and examine the less quantifiable benefits we gain as a result. These measurements look at performance based on business outcomes, not on individual contributors and their vanity metrics, such as lines of code created, number of pull requests, and so on. This sets the developer and the team up for efficiency, real feedback and an understanding of how their work impacts the business.

Being plugged into what both the business and end users need makes the developer's job clearer and outlines what kind of feedback will be most useful in contributing to both developer productivity and developer experience.

Paving a path to developer productivity

The term "developer productivity" can mean different things, but for Ambassador, it means that our culture and our tools empower engineers to deliver business value fast. And one way to achieve this rapid time-to-value is by drinking our own champagne.

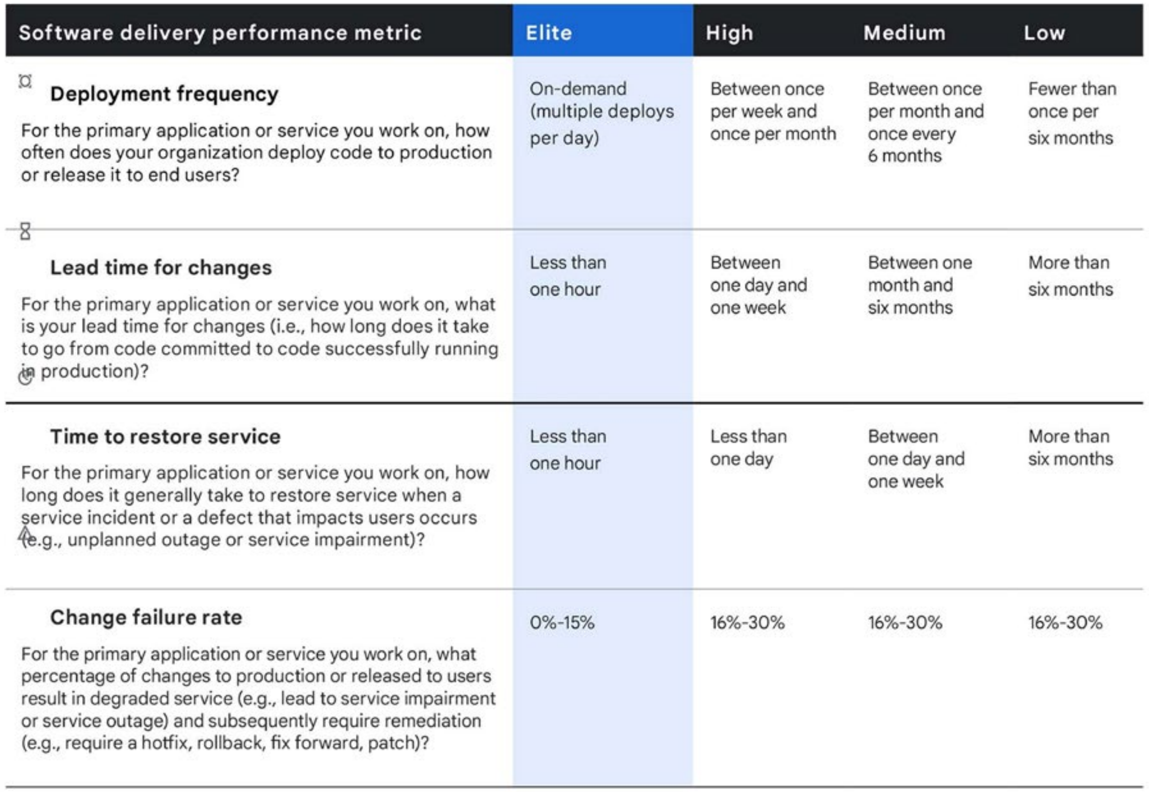
In our case, the champagne drinking starts from developer onboarding. We use Telepresence and Ambassador Cloud heavily, which kicks things off the right way from a developer's day one and continues to help them build their confidence throughout their time as an Ambassador developer. The Ambassador toolkit enables faster engineering onboarding, helping new developers get their development environment set up immediately, often running our cloud application locally and intercepting requests on day one and opening pull requests on day two.

For the Ambassador team, using our own tools every day ensures that improvement is embedded in developers' workflow. And as developers become more familiar with the tools and platform available, our experience shows that developers can reduce their inner and outer dev loop execution time and incrementally build productivity gains alongside faster feedback loops.

Achieving operational excellence

Without setting some clear KPIs it's impossible to measure what "operational excellence" is for your organization. For Ambassador, we look at, for example, the number of incidents we experience, service downtime, software regressions, mean time to detect and mean time to recover. We also tie performance back to the business goals, the core of which is shipping software safely and fast to production.

This leads us to evaluating our performance with [DevOps Research and Assessment](#) (DORA) metrics, which measure deployment frequency, lead time for changes, change failure rate, and time to restore service. These keep us on the right track and currently our DORA metrics rank Ambassador in the "elite tier" when compared to similar organizations.



Software delivery performance metric	Elite	High	Medium	Low
Deployment frequency For the primary application or service you work on, how often does your organization deploy code to production or release it to end users?	On-demand (multiple deploys per day)	Between once per week and once per month	Between once per month and once every 6 months	Fewer than once per six months
Lead time for changes For the primary application or service you work on, what is your lead time for changes (i.e., how long does it take to go from code committed to code successfully running in production)?	Less than one hour	Between one day and one week	Between one month and six months	More than six months
Time to restore service For the primary application or service you work on, how long does it generally take to restore service when a service incident or a defect that impacts users occurs (e.g., unplanned outage or service impairment)?	Less than one hour	Less than one day	Between one day and one week	More than six months
Change failure rate For the primary application or service you work on, what percentage of changes to production or released to users result in degraded service (e.g., lead to service impairment or service outage) and subsequently require remediation (e.g., require a hotfix, rollback, fix forward, patch)?	0%-15%	16%-30%	16%-30%	16%-30%

Source: https://cloud.google.com/blog/products/devops-sre/using-the-four-keys-to-measure-your-devops-performance

External benefits:

Better products in end-users' hands – faster

"Drinking our own champagne allows us to have confidence in what solutions we put into the world. When we use our own tools, we may be using the most common use case, which is not as complicated as the use cases of many of our customers," according to Ambassador Senior Engineering Manager, Steve Barlow. "When the baseline use case is solid, we are very sure that when someone has a problem with Telepresence, like 'is this a bug?', 'is this a missing feature?', 'is this a strange interaction with a load balancer we haven't encountered before?', we can more easily and quickly identify fixes and alleviate points of friction
for users."

Putting our champagne out on the market might be a scary proposition, but when developers work on and use the tools day in and day out, they build confidence that they are in fact creating valuable products that will solve external developers' or companies' problems. Every organization, for example, struggles with sharing and gathering feedback in a cloud-native environment, and Telepresence addresses that. Ambassador's VP of Engineering, Katie Wilde, explained that she was a champion of Ambassador tools at her previous role at Buffer, and came to Ambassador recognizing how other companies could benefit from the challenges Ambassador's products tackle – first and foremost, unblocking engineers and removing friction.

"Like most engineering orgs, we need to share and review a lot of our work. We also do a lot of cross-functional work at Ambassador, and there's nothing like Telepresence to be able to help colleagues share their work and get fast feedback without friction," Katie explained.

A platform engineering sample platter:

A smörgåsbord of developer control planes hands – faster

Bringing together the fundamental enablers of increased developer productivity and better developer experience, the common denominator for Ambassador as well as other organizations is a developer control plane or opinionated developer platform that helps ease the developer journey from day one on the job. In many cases, and certainly in ours, "champagne-drinking experiences" (weaving our own tools into the developer platform) are all about reducing friction to be able to:

- Create a better developer experience
- Ship software faster, safely

And we are not alone here. Platform engineering has become something of a buzzy topic, but for good reason. Every day we hear of another organization building a platform (and more formal polls and surveys bear out this observation). Everyone wants to reduce complexity and enable developer self-service, which is exactly what platform engineering is meant to do. There is no one single way to get to the "right" developer platform, but figuring out what you need to achieve, as we have done, smooths the developer journey and improves your product(s) in the end.

What is platform engineering?

Just so there's no confusion, what do we mean by platform engineering? Our friends at Humanitec shared a definition of platform engineering that resonated not only with us, but appeared in similar forms in almost every conversation we have had with engineers, architects, SREs and developers:

"Platform engineering is the discipline of ***designing and building toolchains and workflows that enable self-service capabilities for software engineering organizations in the cloud-native era****. Platform engineers provide an integrated product most often referred to as an "Internal Developer Platform" covering the operational necessities of the entire lifecycle of an application."*

Developer platforms are usually built with these goals at their core, but the challenge is getting the balance right for the majority of developers. This is often when teams can have the most influence by infusing their platforms with their own champagne – tooling that they know can really change the developer experience for the better. And at the same time, look at how a platform helps cement or influence the development and company culture, too.

Building your platform:

Bottling the champagne

If the consensus holds that a developer platform delivers a reduction in developer toil and a boost to productivity as well as a "seamless path to production", as Bo Daley of Zipcar described good developer platforms, it certainly means that you're not just drinking champagne but actively creating and bottling it.

How? In many cases you will be building (and using) some of your own tools. And then you will be looking for the external tools that will augment the platform you're building, that is, creating your opinionated developer platform as it suits your organization, business goals and workflows. And this probably includes, as we experienced and Bo echoed, an onboarding path for new developers to contribute immediately. The platform becomes your champagne bottling operation because while it makes developer experience consistent in your own organization, the platform you create will most likely be unique to your organization.

Conclusion:

Let the champagne flow

While we conclude that “drinking your own champagne” is very much shaped by tools and platforms, the bigger picture depicts the way companies create their own culture. And the champagne, while encompassing the tools as ingredients, is much more than the sum of its parts. The best vintage is one in which all the ingredients – the people, their experience, their expertise – meld to make it what it is, and more about producing the champagne than drinking it. But that’s why culture is so important – it’s the “mash” from which well-functioning tools and processes are created.

And once we have created those tools, processes and platforms, we drink our own champagne for a number of reasons, but two stand out.

First, the developer experience, as we have written before, is much more than just technology and tools. It also encompasses the aforementioned culture and support. But supporting the developer experience is also very much about the technology and tools developers use day in and day out. We have tested the hypothesis that developer control platforms offer built-in paths that loosely define workflows, guardrails and tools that provide safety and confidence. The platform experience created should, as Crystal Hirschorn, Director of Engineering - Infrastructure, SRE and Developer Experience at Snyk, said, “be a pleasure to use and make a developer’s work easier. The goal is to reduce developer toil and pain, and maybe even bring joy”.

In addition, making our own tools a part of that platform delivers not only greater efficiency for developers but also the second of the reasons why we are committed to drinking our own champagne: users.

We want to ensure that users (anyone outside of Ambassador) have as good a champagne experience externally when they use our products and reap the same benefits – or better – than we do. By using our products ourselves, we remain closer to users, and have empathy for the user experience, leading to better products that reach the market faster.

ambassador

K8s in 8

CHALLENGE

Take the challenge for a chance to win an iPhone 14, a Pluralsight subscription, and other prizes.