Published in 4th Coffee

Tiexin Guo Follow

Nov 2, 2022 · 9 min read · ✦ · ▶ Listen

Save

10 New DevOps Tools to Watch in 2023

Photo by Matt Ragland on Unsplash

With less than two months left in 2022, today, I'd like to look at some new (relatively) DevOps tools we might want to follow in the next year because they might boost engineering productivity to the next level.

> *Author's note: it's worth noting that not all of them are "brand new" tools released recently; most of them are probably tried out by the CNCF end-user community. It's recommended to look at these tools when you have a specific need for the technology in your projects.*

Without further adieu, let's get started.

1 Pulumi

Let's start with the basics of DevOps: Infrastructure.

First things first, Pulumi is an Infrastructure as Code (IaC) tool, just like Terraform, AWS CDK, CDK for Terraform, etc.

Nowadays, although Terraform probably has become the most popular choice for IaC, there are some drawbacks:

- You'd have to learn a new "language" that is HCL(HashiCorp Config Language).
- HCL isn't precisely a "language", per se, or at least not a very powerful one. For example, you can't use a `for` loop on a module until late 2020.

Enter Pulumi.

What is it, then?

If you are familiar with AWS CDK, it's easy for you to understand: it's that. Except it's universal (at least it tries to be) and compatible with every cloud.

If you don't know AWS CDK, think this: Pulumi allows you to manage your infrastructure with programming languages that you already know, eliminating the overhead of learning yet another configuration language.

Who's Pulumi for? Great question.

If you are already familiar with some programming language, like TypeScript, Python, Go, C#, Java, etc., but you don't really want to learn yet another language that is HCL, Pulumi might be for you. If you are using AWS, technically, you can use AWS CDK too, but if you plan to orchestrate a hybrid cloud architecture, Pulumi makes more sense.

If you are already using Terraform heavily, but you are tired of the limitations of HCL, and you don't really like using `count` and built-in functions that lower the readability of your infrastructure code, you could also give Pulumi a try.

This tool isn't "new" anymore; it has more than 14k stars on GitHub. But it's newer than Terraform. If it solves a specific problem you have, give it a try.

2 SOPS

SOPS, short for **S**ecrets **OP**eration**S**, is an open-source text file editor that encrypts/decrypts files automagically.

Emphasis on the text editor, encryption, and automation.

Typically, when you want to encrypt a text file, this is what you do:

- Use your favorite editor for writing, editing, and manipulating the text data, and save it as a file.
- Use an encryption/decryption tool to encrypt the whole file.

When you need to read the encrypted file:

- First, you must decrypt the file using an encryption/decryption tool.
- Open the decrypted file (now it's a regular text file) with a text editor of your choice.

The drawback of this "normal" process is obvious: you need two tools (an editor and an encryption/decryption tool) for one job.

You probably see where I'm going with this, and you are right: SOPS is for that.

In short, it can be integrated with many encryption services (like HashiCorp Vault, AWS KMS, etc.) to encrypt your secret files automatically, making using a git repo to store secrets possible and easy for collaboration.

Intrigued? Read more on it here, which even has a quick demo/tutorial that you could try yourself.

3 Trivy

Containerization and 12-factor apps have become so popular nowadays that they are your first thoughts when you want to build/deploy an app. Since we rely on container images heavily for our cloud-native workload, the importance of container image security is rising all the time: any container created from an image inherits all its characteristics — including security vulnerabilities, misconfigurations, or even malware.

Trivy is a security scanner. It is reliable, fast, effortless, and works wherever you need it. Trivy has different scanners that look for various security issues, and the most famous use case is for container image Known vulnerabilities (CVEs) scanning.

You can run it as a CLI tool locally to scan your local container image and other artifacts before pushing it to a container registry or deploying your application.

Moreover, Trivy is designed to be used in CI and can be easily integrated with your CI pipelines, thus perfectly fitting in the “continuous everything” DevOps mindset.

4 Cluster API

> *Cluster API is a Kubernetes sub-project focused on providing declarative APIs and tooling to simplify provisioning, upgrading, and operating multiple Kubernetes clusters.*
>
> *Started by the Kubernetes Special Interest Group (SIG) Cluster Lifecycle, the Cluster API project uses Kubernetes-style APIs and patterns to automate cluster lifecycle management for platform operators. The supporting infrastructure, like virtual machines, networks,*

> *load balancers, and VPCs, as well as the Kubernetes cluster configuration are all defined in the same way that application developers operate deploying and managing their workloads. This enables consistent and repeatable cluster deployments across a wide variety of infrastructure environments.*

If the official definition baffles you, think this: you can run one `kubectl apply` command to create a K8s cluster, and it works for AWS, Azure, DigitalOcean, Docker, GCP, OpenStack, and more.

No need for creating Terraform modules (or worse, trying to figure out all the parameters of somebody else's modules) for K8s clusters, no need to figure out how to use `eksctl` for AWS and something else for another cloud; simply `kubectl apply` to create clusters. Sounds impressive, right? I know. That's why it's on the top-10-tools-to-watch list.

5 Linkerd

Linkerd is the world's lightest and fastest service mesh (at least, that's what they say). What's a service mesh? A service mesh is a dedicated infrastructure layer for making service-to-service communication safe, fast, and reliable.

The ease of use is where Linkerd really shines. You can install it with one line of command. That's the end of this paragraph. I don't know what more to say here; it's that simple.

But let's talk more.

The setup is fast. Even the docker images are small, so they get pulled faster.

The architecture isn't drastically different. There is a control plane and a data plane, where the control plane is a set of services in charge of telemetry, API, providing control data to the data plane proxies, etc., and the data plane has proxies that run next to each service instance. Check out the official doc here if you wish to know more details.

Istio and AWS App Mesh use the open-source envoy proxy, a high-performance C++ distributed proxy designed for single services and applications. It's a complex general-purpose proxy. Linkerd, on the other hand, uses a specifically designed proxy written in Rust to be as small, lightweight, and safe as possible. I'm not here to judge which is the best and safest language, C++ or Rust, but as a modern language with a specific way to manage memory (ownership instead of garbage collection), Rust sure has an edge.

For multi-cluster management, unlike Istio, Linkerd uses a service mirroring mechanism. The setup is also relatively simple, almost just like a single-cluster setup, except you have to do it twice plus a multi-cluster control plane.

To summarize, Linkerd is a different kind of service mesh: ultra-light, ultra-simple, and ultra-powerful. Linkerd adds security, observability, and reliability to Kubernetes without the complexity. It isn't exactly a new tool either, but if the features fit your needs and you like simplicity, give it a try.

6 GitHub Actions

GitHub Actions is yet another CI.

Why GitHub Actions, then?

Well, for one, it is in the CNCF tech radar (and in the "assess" stage, making it a "new" tool), so we kind of have to give it a good look.

For another, CI interacts with your code a lot, and by nature, GitHub Actions interacts with your GitHub repos easily. No more trouble integrating your CI with your code repos.

Another benefit for start-ups is: GitHub Actions has some free quota, so when you just launched a new product, the free quota might be more than enough, making it completely free. You probably don't need to register some extra self-hosted runners for quite a long time, and you save the costs of running some VMs in some cloud for your own infrastructure just for the CI part.

7 Tekton

Tekton is yet another CI (I know, I copy-pasted this line from the previous section).

The key features are:

- You can run it in a K8s cluster.

Besides, Tekton lets you build, test, and deploy across multiple environments, such as VMs or serverless. You can also deploy across various cloud providers or hybrid environments using Tekton pipelines.

Should you use it? My take is, if:

- you have to “own” your CI system (for example, using GitHub Actions free quota isn’t an option to you for some reason);
- you already are using K8s;
- you like the way how you interact with K8s;

Then give Tekton a try.

The installation is simple; you can get it up and running quickly.

8 HashiCorp Harness

Harness is yet another CI, but it’s more than that.

It’s from HashiCorp, a name we are already familiar with, and it combines a few things into one:

- CI
- CD/GitOps
- feature flags

- cloud costs

Harness offers hosted virtual machines (VMs) to run your builds. With Harness Cloud, you can build your code worry-free on the infrastructure that Harness provides. You can spend less time and effort maintaining infrastructure and focus on developing great software instead.

889 | 16

In Harness, Continuous Delivery is modeled using Pipelines and Stages. In each stage, you define what you want to deploy using Services, where you want to deploy it using Environments, and how you want to deploy it using Execution steps.

Harness GitOps lets you perform GitOps deployments in Harness. You define the desired state of the service you want to deploy in your Git manifest and then use Harness GitOps to sync the state with your live Kubernetes cluster.

Harness Feature Flags (FF) is a feature management solution that lets you change your software's functionality without deploying new code. It allows you to hide code or behavior without shipping new software versions. A feature flag is like a powerful `if` statement.

In short, if you want a SaaS CI/CD/FeatureFlags all in one place, this is the one to look at.

9 Thanos

First, a little bit on Prometheus's local storage:

> *Prometheus's local storage is not intended to be durable long-term storage; external solutions offer extended retention and data durability*.

Although we can set a long data retention period like years with `storage.tsdb.retention`, the question remains on scale and planning. With years of high-resolution probes, processing very long queries can take a lot of memory. It also comes down to scale: for example, a `rate()` function over one year with a 15-second scrape interval requires 2.1 million samples or about 2.6MiB of data. And that's only for a single metric.

If you have a small infrastructure, there is nothing wrong with adjusting the retention time to years; the current TSDB implementation is perfectly able to handle this. For larger applications, consider a larger distributed TSDB.

And Thanos is a solution that solves this problem: it is an open-source, highly available Prometheus setup with long-term storage capabilities, focusing on long-term storage. If you have already met issues with Prometheus storage, try Thanos.

10 HashiCorp Sentinel

Finally, let's talk Sentinel.

Policy-as-code is an approach to policy management in which policies are defined, updated, shared, and enforced using code, and Sentinel is HashiCorp's take on that.

Since Sentinel is from HashiCorp, it integrates well with HashiCorp's other products. So, if you are a heavy user of Terraform, Vault, Consul, or Nomad and want to try Policy-as-Code, Sentinel is just the right tool for you.

To give a few concrete examples of what Sentinel policies can do:

- Do not allow Cloud resources to be provisioned without tags with Terraform.
- Ensure that modification of critical Vault data can only be performed by authorized sysops with valid MFA.
- Only allow Docker workloads in Nomad.
- Consul keys can only be updated during business hours.

A short code sample:

```
import “tfplan/v2” as tfplan

aws_instances = filter tfplan.resource_changes as _, rc {
  rc.mode is “managed” and
  rc.type is “aws_instance” and
  rc.change.actions is not “delete”
}
```

```
main = rule {
  all aws_instances as _, instance {
    (instance.change.after.tags else {}) is not empty
  }
}
```

Pretty self-explanatory, right? Get the AWS instances from a Terraform plan; tags can't be empty after the change(unless you are trying to delete the instance).

If you are interested in Policy-as-Code, please stay tuned; I'll publish an introduction article on this topic soon.

Summary

A quick categorization of all the mentioned tools in this article:

- Infrastructure-as-Code: Pulumi
- Security: SOPS, Trivy
- K8s/multi-cluster: Cluster API, Linkerd
- CI/CD: GitHub Actions, Tekton, HashiCorp Harness
- Monitoring: Thanos
- Policy-as-Code: HashiCorp Sentinel

If you like this article, please give it a like, comment, and subscribe! See you in the next one.

Dev Ops Software Engineering Cloud Computing Software Development

Platform Engineering

Get an email whenever Tiexin Guo publishes yet another great article on DevOps.

Confession: not a big fan of email notifications. However, please subscribe via email if you don't want to miss out on a new article on DevOps and the cloud. I'm sorry, and thanks.

Your email

Subscribe

By signing up, you will create a Medium account if you don't already have one. Review our Privacy Policy for more information about our privacy practices.