env0

Infrastructure as Code 101

DevOps Practice: Modern Infrastructure as Code Frameworks

eBook

What is Infrastructure as Code and why does it matter?

Infrastructure as Code (IaC) has transformed the way IT infrastructure is set up and managed, provisioning and managing data centers through machine-readable definition files rather than through physical hardware configuration or interactive configuration tools. With IaC, infrastructure is provisioned with code instead of by manually clicking buttons on a web console. Automatic code can set up servers quickly, without manual intervention.

IaC is an emerging and evolving concept, a powerful technology that allows you to provision and manage any cloud resource in an automated, declarative way. You can create a consistent, repeatable workflow, enabling wider-scale deployments across a range of resources.

Today, organizations are still figuring out how best to implement new IaC practices in their existing DevOps frameworks. This article covers a variety of options for multiple frameworks to support even the most demanding business-critical environments.

Infrastructure as Code: pros and cons

IaC has made IT more efficient than ever before, solving numerous IT challenges and enabling new capabilities such as:

| Recreating environments

It used to be challenging to recreate an identical environment after a deployment because the systems with which it interacted also had to be updated. With IaC, users can recreate infrastructure from scratch, and on-demand, simply by replaying code. The pipeline uses a prescribed set of parameters for deployment and creates a new environment that is identical in terms of the number of hosts, networks, data centers, clusters, and data stores, every time it's run. The infrastructure code can even be versioned with the product, making it easy for engineers to recreate the infrastructure as it was when a previous version of the product was released.

| Minimizes errors

IaC minimizes the need for manual management, reducing the risk of human error. Rather than depending on engineers to remember past configurations or respond to failures, everything is in the code, under version control. When changes go to production, the infrastructure code is checked in a code review or in a review by a gatekeeper.

| Supports teamwork and collaboration

Using IaC, engineers don't have to deal with problems caused by conflicting changes in a shared environment. IaC makes it easier to work as a team and to share code with colleagues and other teams, so they can utilize it to set up their own environments. Using a version control system, different teams can each work on a separate piece of the infrastructure, rolling out their changes in a controlled manner.

| Reduce CapEx cloud expenditures

Before Amazon AWS and other cloud providers, infrastructure investments like servers, network cables, and physical space were made upfront, and if demand dropped, there was no way to minimize the expense. The shift to the cloud reduced the CapEx expenses, and IaC has reduced them even further. With IaC, a developer writes code and configuration management instructions that trigger actions according to actual need and accurately reflect the structure of the real operating environment. IaC lets you manage your environments easily, and automatically deactivates environments you no longer need.

While IaC has clear advantages, it also presents unique challenges.

| Integration with management tools

In order to harness the full benefits of IaC, it must be integrated into all processes, including CI/CD workflows, notification tools like Slack, security tools, system administration, IT operations, and DevOps, with well-documented policies and procedures. Without full integration, errors can quickly spread across the system.

| Longer turnaround

When using IaC, every change has to be coded, tested and reviewed before it is applied. Changes are more complex and must be planned carefully to avoid significant downtime.

| Lack of oversight of expenses

Since IaC deploys infrastructure automatically, it can be hard to keep track of expenses. Development teams are often unaware of the financial ramifications of their code, and expenses can build up quickly without monitoring tools that are designed for IaC.

Infrastructure as Code Main Frameworks and Tools

Framework	Languages	Maintained by
Terraform	HCL / JSON	HashiCorp
CloudFormation	YAML / JSON	AWS
AWS CDK	TypeScript, JavaScript, Python, Java, and C#	AWS
ARM Templates	JSON, Bicep (Preview)	Azure
Google cloud deployment manager	YAML	Google
Terragrunt	HCL / JSON	Gruntwork
Pulumi	Node.js, Python, .NET Core, and Go	Pulumi
Serverless Framework	YAML	Serverless
Crossplane	YAML	Upbound

IaC framework review

There are many IaC tools out there, but let us focus on the most common ones, dividing them into two groups. We will compare each one to the alternatives, to determine whether it is a good choice.

Some IaC tools were I left out of this analysis. Here is why:

Terragrunt: Terragrunt is a wrapper on top of Terraform, and is not chosen independently. If you are already using Terraform, you may consider using Terragrunt to DRY out your code and add flexibility and extensibility, but it can not be considered a stand-alone tool.

AWS CDK: Despite having some strong advantages over CloudFormation, it ultimately comes down to the language selections you get with CDK. It also allows you to choose Terraform CDK, which was announced recently.

Google cloud deployment manager: It is not very popular these days, and if you are on GCP, you will likely prefer Terraform/Pulumi to Google cloud deployment manager.

Serverless framework: This is an IaC tool for serverless development. It is maintained by Serverless (the company) and is a very popular open-source with ~40K stars, a huge community, and available plugins. It supports many service providers like AWS, Azure, and GCP as well as Knative, Kubeless, and others. Despite being very popular, it only targets a specific use case, serverless programming, making it less relevant to this analysis.

Configuration management tools like Chef, Puppet, and Ansible; However, are not IaC tools and thus cannot be compared with IaC tools we will be discussing here.

Cloud-specific IaC

AWS CloudFormation

Overview

One of the oldest IaC tools, CloudFormation was released in 2010 by AWS to provision and manage AWS resources using code. Although not open source, it is used widely by AWS customers. It has many built-in features like templates, stack sets, a web console, a CLI, and an API you can communicate with and can be used with S3 to store your templates.

AWS recently announced a new service called Proton that utilizes CloudFormation to create a more self-service approach for running applications and integrating with CI/CD pipelines and a wider range of resource provisioning. CloudFormation enables custom resources and allows you to write code for resources outside of the AWS ecosystem, for example Auth0 or Sendgrid.

Pros

AWS integration
CloudFormation integrates perfectly with all other AWS resources, including IAM roles, security standards, events, and other AWS services.

Management solutions
CloudFormation includes a management solution with concepts like CloudFormation templates and stack sets, and there are many sample templates out there.

It also includes out-of-the-box history and audit logs in the CloudFormation console for each deployment.

No cost
CloudFormation is free of charge—you pay only for third-party resources.

Cons

AWS-specific
Like the other tools in this section, CloudFormation is AWS-specific, and although it does support custom resources, it is not structured to support multi-cloud deployments.

Unlike other IaC tools, CloudFormation is a closed system managed by AWS, which means that the CloudFormation utilities open-source community is small compared to its rivals.

Declarative language
CloudFormation uses YAML or JSON as a declarative language, which has many built-in limitations.

Missing AWS resources
There are many missing AWS resources or settings inside a resource. As this is not an open-source project, not much can be done about it.

No formal approval flow
There is no formal approval flow, and no policy-as-code or testing framework with which to easily integrate.

AWS CloudFormation

| Summary

If you are an AWS-only company, CloudFormation may be the right choice for you, as it offers many features out of the box at no additional cost. Nowadays, however, most companies seek multi-cloud approach. With the emerging SaaS provider companies, there may be better solutions to meet those same requirements.

Azure resources manager (ARM)

| Overview

Azure has its own IaC tool to provision resources. ARM is not an open-source project, but it has many out-of-the-box features and it integrates with the entire Microsoft ecosystem and offers templates. Microsoft also recently created a new open-source project called Bicep, which is a new DSL that is supposed to be better and less verbose than what the standard JSON-schema ARM templates offer.

| Pros

Template options
ARM templates can be broken down into smaller templates and linked together during deployment, making it easy to control and manage a large codebase and DRY it out. ARM templates can also be extended using deployment scripts (Bash or PowerShell) to give you flexibility in full deployment.

Testing and preview
A testing tool kit is available using PowerShell to test your code and make sure you've followed recommended guidelines before deploying. ARM templates also validate your code before deploying it.
You can preview changes before deploying a template, giving you full visibility into which resources will be created, updated, or deleted.

Microsoft integration
ARM has some great integrations with the entire Microsoft and Azure ecosystem at no extra cost. For example, it includes policy-as-code using Azure Policy, CI/CD using Azure pipelines, blueprint creation, and management from the Azure portal, CLI, API, or your visual studio. It also includes out-of-the-box history and audit logs in the Azure portal for each deployment.

Bicep
Bicep, a new DSL for ARM templates, is an open-source tool to help you reduce the verbose nature of JSON.

| Cons

Azure-only
ARM templates are Azure-specific, and although you can extend them with the deployment scripts, they are not made for provisioning resources outside of Azure.

JSON-based
ARM is JSON-based (Bicep is a DSL) that is very verbose by nature and has many limitations such as condition statements, loops, and more.

No approval flow
Although you can see a preview of the changes, there is no formal approval flow out of the box.

Limited community
Unlike other IaC tools, ARM templates are a closed system managed by Microsoft, which means that the ARM templates utilities open-source community is small compared with its rivals.

Azure resources manager (ARM)

| Summary

If your company mainly uses the Microsoft ecosystem and runs only on Azure, choosing ARM templates is a very wise decision. It offers lots of features out of the box, including testing and previewing changes, and allows you to easily integrate it with other Azure services. However, the main disadvantage is that it is not a multi-cloud IaC tool, and it doesn't allow you to provision other SaaS vendors (not even GitHub, which is owned by Microsoft).

Terraform

| Overview

Terraform is an open-source project by HashiCorp, launched in 2014. It is based on HCL (HashiCorp configuration language) and is the most widely adopted IaC tool out there. It is a CLI tool that can help manage and provision external resources such as public cloud infrastructure, private cloud infrastructure, network appliances, and SaaS and PaaS vendors. Terraform uses a provider concept that is built on top of the core code, which means that you can basically write a Terraform provider for almost anything. HashiCorp maintains most of the official providers (such as AWS, Azure, GCP, and more), but there are also many open-source providers maintained by others, such as IBM Cloud, Alicloud, and more. HashiCorp maintains a public registry for all of its official, verified, and community providers.

| Pros

Lots of providers

The open-source project is maintained by a company and a huge community of developers and has the biggest community on this list. That means you will be able to get a lot of information about the tools and best practices on how to use them.

Terraform has ~1000 providers for almost anything you may be using, even SaaS vendors like GitHub, GitLab, Auth0, Datadog, and more. In addition, you can write your own Terraform provider. It also has ~4.5K open-source modules you can use.

Management capabilities

Terraform's remote backend lets you manage your deployment states and locks when working as part of a team. Its workspaces let you to manage multiple environments on the same code base, and the modules allow you to share code within your organization, or even publicly, maintain a large codebase, and DRY out your code.

| Cons

It's a CLI tool

Terraform is a CLI tool - it lacks many of the basic features needed to manage deployments at scale. That means you will either need to build and use a lot of utilities, or use other tools like env0, which can manage your Terraform orchestration (deployment history, audit logs, RBAC and approval workflow, etc.) for you.

Language

Terraform supports HCL or JSON, but since HCL is a declarative language, it lacks a lot of imperative language features, even though it has come a long way with condition expressions.

In addition, it may mean that some developers will need to learn a new language.

Bugs (lots of them)

Terraform has lots of bugs, with around ~13K issues. To compare, AWS Terraform provider has 2.3K issues - Azure provider has 1.4K issues.

Terraform

| Pros

Utilities
As Terraform is widespread and open source, there are a variety of utilities you can use with it such as OPA for policy-as-code, checkov and tfsec for security scanning, Infracost for cost estimation, and more. Terragrunt is another widely adopted open-source tool that is built on top of Terraform.

Visibility
Terraform's plan allows you to preview changes before deploying your code for full visibility into the resources that will be created, updated, or deleted.

| Cons

No rollback mechanism
Some of the other IaC tools have a built-in rollback mechanism if something goes wrong during a deployment, but Terraform does not. You might end up with a corrupted state file that needs to be fixed manually, creating a drift between the code and state file and the actual provider.

Updates for provider features aren't available from day 1
Since Terraform supports a lot of providers, it takes time for them to support new provider features. For example, if AWS releases a new service, CloudFormation will probably have support for it before Terraform does.

| Summary

Terraform is a great IaC tool that supports a multi-cloud approach. Its vast community and adoption probably make it the best choice for most companies. It also provides a lot of value when managing numerous SaaS or PaaS vendors and going all-in with the "everything as code" mindset. However, if you use Terraform you will need to either build or use another solution like env0 to support and manage a large-scale organization and deployments.

Pulumi

| Overview

Pulumi was founded in 2017, after the major adoption of Terraform. It took a different approach, creating IaC with support for common languages like Node.js, Python, .Net, and Go. It is similar to Terraform but built its SaaS solution on top of its open-source from day one. Pulimi is becoming more and more popular and has ~8K GitHub stars. Pulimi began with minimal support for the public cloud vendors but it is closing the gap. It is a young product, but looks very promising.

| Pros

Imperative language

Pulumi was created for imperative languages, not declarative languages (HCL, YAML, and JSON), which is a major benefit when coding your infrastructure deployment. Since it is based on modern languages, it has out-of-the-box testing frameworks, making it very easy to test your code.

Management capabilities

Like Terraform, Pulumi's remote backend lets you to manage your deployment states and locks when working as a team. You can manage multiple environments on the same code base, and share code within your organization, or even publicly, as well as maintain a large codebase, and DRY out your code.

Concepts and visibility

It has many concepts that help you manage your code and environments like projects, programs, and stacks, and the preview allows you to see the changes before deploying your code for full visibility of the resources that will be created, updated, or deleted.

| Cons

Immature

Since Pulumi is a younger tool, it lacks the richness of other tools when it comes to supporting providers or the resources within each provider. For example, it might have an AWS provider but be missing many resources in AWS.

The Pulumi community is still not huge, which means that the utilities and open-source around it are limited, and only a few providers support Pulumi out of the box.

Limited options

CrossGuard, which is relatively new and is not widespread, is the policy-as-code for Pulumi and works only for Pulumi. In addition, if you want to manage Pulumi deployment at scale you are bound to Pulumi SaaS with limited options.

No rollback mechanism

Some other IaC tools have a built-in rollback mechanism if something goes wrong during a deployment, but like Terraform, Pulumi does not. Therefore, you might end up with a corrupted state file that needs to be fixed manually, creating a drift between the code and state file and the actual provider.

Pulumi

| Cons

Slow updates for provider features
Since Pulumi supports a lot of providers, it takes time for them to support new provider features. For example, if AWS releases a new service, Cloudformation will probably have support for it long before Pulumi, and Terraform will probably get it sooner as its community is bigger.

Focus on migration path
As other IaC tools are already widely adopted, Pulumi is making a major effort to create a migration path from other IaC tools like Terraform, Kubernetes, and ARM templates. The company is focusing extensive resources on those tools instead of focusing on its core product, which impacts quality.

| Summary

Pulumi is a fast-growing IaC framework with significant advantages, mainly regarding language, that may provide a better learning curve compared with other tools. If you are just starting your journey in the IaC space, Pulumi can be an easy way in. However, if you have a large ecosystem you need to support with a lot of resources, in its current state, Pulumi might not be the best option for you.

Crossplane

| Overview

Crossplane is a CNCF sandbox open-source IaC project built by Upbound. It leverages Kubernetes and the Kubernetes ecosystem, extending the Kubernetes API to manage and compose infrastructure. Crossplane is the “new kid on the block” and was first released in December 2020. It is a fast-growing community with ~3.2K GitHub stars and ~60 contributors.

| Pros

Kubernetes
Crossplane is based on Kubernetes, which means you get all the benefits of Kubernetes out of the box, like RBAC, configuration management, etc. It also means that you get the power of the huge Kubernetes community and utilities and integration with tools that are based on Kubernetes like ArgoCD.

State management
Crossplane manages its state inside your Kubernetes cluster using etcd, which is much more reliable and secure, and therefore highly available and fault-tolerant. One of the key benefits from this is auto-sync on the state, which can help solve drifts out of the box.

| Cons

Immature
Since Crossplane is a very young tool, it lacks the richness of other tools when it comes to supporting providers or the resources within each provider. It currently only supports the four big cloud providers (AWS, Azure, Aliclouds and GCP), and doesn’t have the full support of all of their resources.

No formal approval flow
As Crossplane doesn’t include a preview of what it’s going to perform, you can’t build an approval flow mechanism for it. There is no policy-as-code or testing framework with which it can easily integrate.

It runs inside a Kubernetes cluster
Crossplane can only run inside a Kubernetes cluster. Although Kubernetes is widely adopted, getting started with Crossplane is still more difficult, and it is harder to run it locally and test it. It also means that you need to configure it and maintain its Kubernetes resources.

| Summary

Crossplane is the newest IaC tool out there. It leverages the Kubernetes ecosystem which adds numerous advantages but also increases its complexity. As a new tool, it is not yet mature, and supports limited providers and resources. Therefore, you should consider Crossplane as an option to manage your infrastructure only if you are familiar with Kubernetes and don't need to support a large number of resources.

The env0 Advantage

There are a lot of robust IaC tools out there, each with distinct advantages and disadvantages. There is no “one size fits all” approach—the right choice for your company depends on its size, current platforms in use, and employee expertise. Whichever tool you choose, when you get to scale and have to manage a large number of cloud resources and pull requests, you need more than a standard IaC platform.

One of the key challenges is continuous integration (CI) in the world of IaC. When you create a pull request with code changes, you will want to make sure the code is valid, and to understand what resources will be removed, updated, and/or created as a result of the change. It is critical to understand how the change will affect the real, live environment, so you will need to run your changes against a running environment for each commit. It can get heavy.

env0 solves this problem with a one-stop-shop that gives DevOps everything they need for the pull request in one place. With env0, you get immediate notifications about pull requests and their contexts so that as a reviewer, you can be sure the code is correct and of high quality before it gets merged into master. It gives you easy visibility into the GitOps workflows of infrastructure changes and delivers the Terraform or Terragrunt plans to all pull requests so you can approve the infrastructure change requests before applying changes on environments (production, staging, and dev). env0 also makes the process of reviewing a Terraform plan easier for both admins and team members.

Another challenge in IaC is environment drift that can lead to costly business waste. This happens because teams build against a staging or development environment and then, upon deployment, find that the production environment is out of sync, which can lead to a time-consuming investigation of why and what is missing. env0 solves the problem with a centralized, consistent remote-run cloud credentials process that avoids local deployments.

env0 also gives you detailed visibility into requested changes in your cloud resources, and full visibility of your cloud deployment history so you can better investigate infrastructure changes and identify issues. env0 even sends you immediate alerts about issues and policy violations that need your attention, along with actionable recommendations on how to fix them.

An additional problem in IaC is that when many people have write access to the cloud account, granting, changing, and revoking access to resources can be difficult (too many cooks in the kitchen). env0 allows you to define and enforce granular policies for how your organization provisions infrastructure. It offers a centralized, consistent remote-run cloud credential process that allows you to establish RBAC controls for who can apply changes and how, and define groups of users that match your organization's real-world teams, assigning them only the permissions they need.

In summary, if you’ve chosen Terraform or Terragrunt and as your IaC framework (Pulumi coming soon) and are working at scale, env0 provides out-of-the-box visibility into the actual cost of environments, projects, teams, and users and significantly simplifies the governance of cloud deployments.

About env0

env0’s collaborative remote-run workflow management platform automates and simplifies the governance of cloud deployments for Terraform, Terragrunt and IaC frameworks.

env0