How Infrastructure as Code (IaC) manages complex infrastructures

BY IAN BUCHANAN

Summary: *Infrastructure as Code (IaC) is an IT infrastructure management process that applies best practices from DevOps software development to the management of cloud infrastructure resources.*

The rise of hardware virtualization in the mid-2000s spawned new opportunities of cloud infrastructure hosting. Cloud hosting providers began offering access to dynamic Infrastructure as a Service (IaaS) platforms. As these platforms grew and began offering more complex infrastructure assets, the complexity of a traditional systems administration role also grew. The need to rapidly configure and manage complex cloud infrastructures quickly became a challenge.

The idea of Infrastructure as Code (IaC), or modeling infrastructure with code, was spurred on by the success of CI/CD. DevOps proved how productive it was to commit code to a Git repository and then apply feature branches and pull request workflows. The automation these workflows brought to software development helped reduce the new complexity of cloud systems administration.

What is Infrastructure as Code?

Infrastructure as Code is an IT infrastructure management process that applies best practices from DevOps software development to the management of cloud infrastructure resources. Applicable infrastructure resources are virtual machines, networks, load balancers, databases, and other networked applications.

IaC is a form of configuration management that codifies an organization's infrastructure resources into text files. These infrastructure files are then committed to a version control system like Git. The version control repository enables feature branch and pull request workflows, which are foundational dependencies of CI/CD.

Infrastructure as Code is made possible by the rise of cloud infrastructure hosting platforms, specifically IaaS platforms. IaaS allows on-demand provisioning and requisition of cloud resources through remote APIs, which set the template for properties committed to the infrastructure configuration files. IaC's automation features can take the configuration files and run them against the remote IaaS APIs.

Once a team has committed infrastructure configuration to version control, they can apply CI/CD practices to infrastructure changes. Infrastructure updates can follow a DevOps workflow. If a team member has edited one of the configuration text files, pull requests and code review workflows can be used to audit and verify the correctness of the edits. Furthermore, a DevOps-enabled infrastructure as code system will utilize automatic infrastructure deployments and rollbacks.

Why is Infrastructure as Code important?

IaC evolved to help solve the problem of "environment drift". Cloud applications usually have separate deployment environments for the stages of their release lifecycle. It's common to have development, staging, and production environments. These environments are composed of networked resources like application servers, load balancers, and databases. Environment drift occurs when the infrastructure between these different environments falls out of sync.

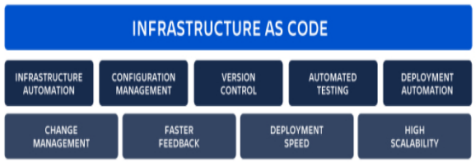
Without IaC, infrastructure management can be a disorganized and fragile process. System administrators manually connect to remote cloud providers and use API or web dashboards to provision new hardware and resources. This manual workflow does not give a holistic view of the application infrastructure. Administrators may manually make changes to one environment and forget to follow through on the other. This is how environment drift happens.

Environment drift becomes an expensive business waste. Bugs and failures happen because teams build against a staging or development environment and then find upon deployment that the production environment is out of sync, which leads to a time-consuming investigation of why and what is missing.

Without IaC, manual infrastructure management is a slow process. If a required infrastructure change is identified due to environment drift, spikes in traffic, or some other issue, it can take an unknown amount of time for a systems administrator to react and adapt. This leads to outages and customer frustration. With IaC in place, infrastructure can automatically adapt to changes in configuration and react to spikes in traffic with auto-scaling features.

Infrastructure as Code brings more oversight and visibility to manual systems administration. With the infrastructure configuration files committed to a central version control repository, all team members can view and edit infrastructure data. This enables powerful auditing capabilities. For example, if your team undergoes a PCI compliance audit, you'll need to know if a specific part of your infrastructure is using SSL encryption. With IaC you can quickly see how SSL is configured and execute the code to make sure the live infrastructure matches the configuration files, which identifies if SSL is enabled. The version control commit history also acts as a log to review when it was added or removed.

How does Infrastructure as Code work?



There are a few dependencies that need to be in place to fully achieve Infrastructure as Code.

Remote accessible hosting or IaaS cloud hosting platform
The first and foremost dependency is remote accessible hosting. The configuration management tool needs to connect to and modify the remote host. If the remote infrastructure is self-managed your team needs to ensure the configuration management tool has access. The IaaS-enabled cloud hosting platform offers APIs that allow users to automatically create, delete, and modify infrastructure resources on demand. These APIs can also be accessed by configuration management tools to further automate these tasks. Some examples of popular IaaS platforms are Digital Ocean, Amazon AWS, and Microsoft Azure.

Configuration management platform
The next requirement to complete IaC is a tools suite that connects to the IaaS APIs and automates common tasks. A team can create a set of scripts and tools. However it would be a lot of work, future maintenance, and probably have a low return on investment. There are already many open source configuration management platforms that solve this problem, including Terraform, Ansible, Salt Stack, and Chef.

Version control system
A configuration management platform uses human and machine-readable text files written in a markup language like YAML to declare tasks and sequences for the platform to execute. These text files can be treated like application code files and stored in a version control system repository. The repository acts as a central source of truth and enables

pull requests and code review. The most popular version control system is Git.

With these dependencies in place, let's consider an example scenario in which a developer wants to add a new application service to a system. This scenario helps demonstrate an IaC workflow

1. The developer edits a YAML configuration text file in their configuration management platform of choice, Terraform. The edits specify that a new hosting server is needed.
2. The developer commits edits to a feature branch in the Git repository. Because the project's Git repository is hosted on Bitbucket, the developer opens up a pull request. Another team member reviews the pull request and becomes aware of the new infrastructure changes. The team member approves the pull request and the developer then merges the commit to the main branch of the repository.
3. At this point, the configuration platform is needed to execute an update. The update can be manually triggered by the developer. Because the team is using Bitbucket they also have access to Bitbucket Pipelines and can automate this step with a pipeline.
4. Upon execution Terraform interfaces with the team's IaaS. Terraform executes a series of commands against the IaaS API to bring the IaaS up to date with the expected infrastructure configuration.

In conclusion...

IaC is a highly productive form of configuration management that focuses on automating cloud IT infrastructure management. Once IaC is in place it can be used to achieve levels of CI/CD automation for changes to a project's infrastructure. IaC enables many beneficial insights into communication and transparency around infrastructure changes. IaC requires a set of dependencies like hosting platforms and automation tools, that are widely available from modern hosting companies.

IAN BUCHANAN

While Ian has broad and deep experience with both Java and .NET, he's best known as a champion of agile methods in large enterprises. He's currently focused on the emerging DevOps culture and the tools for enabling better continuous integration, continuous delivery, and data analysis. During his career, he's successfully managed enterprise software development tools in all phases of their lifecycle. He has driven organization-wide process improvement with results of greater productivity, higher quality, and improved customer satisfaction. He has built multi-national teams that value self-direction and self-organization. When not speaking or coding, you can find Ian indulging his passions in parsers, meta-programming, and domain-specific languages. Follow Ian at @devpartisan.

Subscribe

Up Next
Cloud Bursting →