BY CHRISTIAAN NIEUWLAAT

INFRASTRUCTURE AS CODE

Every software solution needs to run on an **infrastructure**. Even the so-called serverless solutions need to run on an infrastructure, which in that case is abstracted from the user. Traditionally, infrastructure consists of bare metal servers and networking appliances, and runs on-premise or (managed) in a rack in a datacenter.

To configure, run and maintain these servers and appliances, we needed **seasoned ops engineers**, who would do endless patching and configuration updates. Doing their magic always took a significant amount of time and knowledge.

Through the modern technology of virtualization, we now have access to things that we weren't able to do traditionally. We can have:

- ✓ **Virtualized infrastructure with virtualized machines, appliances, and containers**
- ✓ **Cloud computing for highly scalable, reliable infrastructure**
- ✓ **Managed infrastructure services such as the popular Infrastructure as a Service (IaaS) and Platform as a service (PaaS)**

These technologies make it possible **to easily scaffold** an entire infrastructure with all required servers and appliances. In just a few clicks, it is possible to roll out a new infrastructure, bells and whistles included, albeit all virtualized. This is where **infrastructure as code** (IaC) comes in to make your life deploying virtual infrastructure even easier and automated. By leveraging IaC principles, infrastructure is handled integrally within the (secure) **software development lifecycle** (SDLC) where changes are made safely and easily.

MANAGING INFRASTRUCTURE IN THE MODERN AGE

Thanks to the advancements in technology, we now have a totally different approach to obtaining and managing infrastructure than we used to do. In the "old" days, we needed to determine beforehand what hardware and infrastructure our application landscape should run on. From determining the (minimum) server hardware requirements to how many network appliances we should purchase to create **a safe application environment**.

When we had compiled this "grocery list" of hardware and appliances, you would go to one or more hardware vendors to purchase the needed components, wait for delivery, put it all in 19" racks in an on-site or managed datacenter, hook up the network connectivity and configure everything to be secure and fully functional in the required way.

This had some **downsides**, for example:

- Determining the right sized hardware and backbone is difficult
- Hardware and appliances are costly to purchase
- An existing infra does not easily scale when demand grows
- You would need specialized operations engineers to configure, fine-tune and maintain the environment (including patching, proactive monitoring, etcetera)
- Hardware ages and needs to be replaced when it malfunctions or gets outdated

Nowadays, due to the advancement in virtualization, we can easily create a virtualized infrastructure in software with nearly the same performance just by using a few tools. We still create the "grocery list" to determine requirements but we then don't need to go shopping at hardware stores. We simply place our order in code after which a specialized suite of software virtually constructs the required environment in a manner minutes at our cloud provider or on our own on-prem hardware.

Our "grocery list" is a codified version of wat we need, hence we call this approach infrastructure as code.

INFRASTRUCTURE AS CODE

As the name implies, we can describe our required infrastructure in a codified form. This way we can declare what resources we need in whichever configuration we want, and let the software do the heavy lifting to create the infrastructure in compliance with the described state.

Due to the infrastructure being code, we can manage it in the same way we do with application code and integrate it fully within our application development life cycle. In short, we can do everything with our infrastructure like we do with our application code. For instance, we can include the code in a version control system such as git or mercurial, (automatically) test the infrastructure scripts, and integrate the infrastructure deployment in the delivery pipeline easily.

GETTING IT **DONE** ✓

IaC relies heavily on tooling to be effective. Though it's entirely possible to create virtualized infrastructure using shell scripts and REST calls, there are better tools available specifically for IaC. The best known tool for this purpose is probably Terraform by Hashicorp, in which you specify the goal infrastructure in plain text files. Terraform has a plethora of (community-provided) provider modules, and is thus able to create infrastructure for multiple cloud providers (and on-site cloud systems like VMWare and OpenStack). The downside is that you'll have to learn the Terraform syntax to use Terraform for your infrastructure.

When you're coming from a developer mindset, you might want to use a tool that can create your infrastructure using an already known programming language. In this case you can use multiple different approaches:

1. **DIY: Create all infra using cloud-provider specific API calls (not recommended)**
2. **Use a vendor-provided library (such as the AWS Cloud Development Kit (CDK))**
3. **Use an IaC tool such as Pulumi (which is comparable to Terraform, but uses an already known programming language instead of a yaml specification)**

Using the developer-focused approach to Infrastructure as Code, will give you more control over your infrastructure and how it will be provisioned. You can use all flow control mechanisms provided by the chosen programming language to determine what the goal infrastructure being provisioned will be. This way, highly complex, logic-driven, infrastructures can be provisioned.

IMMUTABLE INFRASTRUCTURE

The power of IaC opens a world of opportunities. One of these is to create so-called immutable infrastructure. In its most basic sense, immutable infrastructure is the concept that next to no changes or configurations happen outside of the infrastructure source code: **the infrastructure is integrally provided on (every) deploy of your application landscape**. In essence, the existing infrastructure is taken down and a fresh infrastructure is being created and configured on every deployment, so the application(s) run on an infrastructure that is tailored to its needs and in a known good state.

Note! When using infrastructure as code or immutable infrastructure, keep in mind that **all resources are taken down on a destroy or a redeploy.** If you have data that needs to be persisted across deployments, please store this on a shared storage medium that isn't taken down.

TOOLS INDEX

When using Infrastructure as code, or immutable infrastructure, there are multiple tools working together to make this experience as easy as possible. Depending on the target platform on which the virtualized infrastructure is being created, there are different tools to use. The following list presents some valuable options when using IaC:

Git (AWS CodeCommit, Azure DevOps repos) provides a fully featured source control and versioning system that helps track changes to code over time. This eases tracking infrastructural changes.

Delivery pipelines (AWS CodePipeline, Azure Pipelines, GitLab pipelines, Jenkins/BlueOcean) provide the ability to build, test and deliver all code (including infrastructure code) in an orchestrated way.

Terraform* / Pulumi acts as the orchestration tool for rolling out the proposed goal infrastructure. These tools track changes in infrastructure state, so only necessary changes are rolled out incrementally.

* When using Terraform as orchestrator, we advise to use TerraTest to run integration tests on the infrastructure rolled out by Terraform.

START YOUR FORAY INTO THE WORLD OF ENDLESS POSSIBILITIES

As stated earlier, infrastructure as code opens up a world of possibilities. Almost everything is possible, from creating a repeatable, consistent way of provisioning a sound infrastructure once, to entirely integrating this automated way of provisioning infra in a multi-tenant cloud application, where on the creation of a tenant, their entire environment including all virtualized hardware and appliances, gets “automagically” provisioned in a matter of minutes.

Grasp these possibilities now, and start using IaC in your next project. There is a slight learning curve, though this doesn’t weigh up to the benefits you reap when using this technology. If you need assistance or guidance, get in touch, we’ve got your back if needed.

THE WORLD OF IAC IS YOURS,

NOW GO AND EXPLORE...