Infrastructure as code and cloud-native security

So far, you've learned that infrastructure as code (IaC) is the use of code to provision and manage infrastructure. In this reading, you'll learn more about how to use IaC, and some of the ways it can help with security issues. You'll also learn about the declarative and imperative approaches to IaC.

Infrastructure as code (IaC)

IaC uses automation to create cloud resources. Cloud computing uses a large number of components, so there are many details for cloud security professionals to manage. Luckily, IaC can manage these details, keep your infrastructure consistent, and reduce human error.

The declarative approach

There are two ways to approach IaC: the declarative approach and the imperative approach. The declarative approach defines what resources to include, and what properties those resources should have.

The benefits of the declarative approach include:

- Keeping a list of a system objects' current state
- Improving the ease of managing the resources in scope across their lifecycle, including taking down the infrastructure
- Providing the desired states of the resources in the configuration;the tool compares the desired state to the current states of the resources, and adjusts them to match what's defined as the desired state

The imperative approach

The imperative approach defines the commands needed to achieve the desired configuration.

The commands in the configuration need to be executed in the correct order to achieve the desired state.

Declarative vs imperative tools

The capability of declarative and imperative tools is the same. Both types of tools automatically provision infrastructure and apply any changes you specify. The difference between these tools is how this process happens. Declarative tools don't need to know the current state, while imperative tools need to know the current state of each infrastructure component to determine whether to create or modify it.

For declarative tools, you need to specify what you want to achieve, and where you want it to be implemented, so the tool and service provider can make that outcome happen. With declarative tools, the outcome you specify is the outcome you get. With imperative tools, you need to provide step-by-step instructions on what process you want to be completed, and how you want it completed. If you provide these instructions correctly, there won't be any errors.

Two examples of declarative IaC tools are Terraform and Deployment Manager. The standard approach used in most tools, including Terraform, is declarative.

How infrastructure as code supports security

IaC helps build a strong foundation for your infrastructure. It also lets you control changes and more efficiently configure your cloud environment. And, it can help with security by performing some tasks for you. For example, IaC lets you use automation, drift detection, and source control.

- Automation is the use of technology to reduce human and manual effort to perform common and repetitive tasks.
- Because resources change from their original and desired states over time, sometimes things are added that introduce vulnerabilities. Drift detection identifies when changes have been made.
- Source control is similar to software code. Configuration can be kept in a source control system where changes are tracked and managed, and rollback can be performed if needed.

IaC templates

There are many IaC templates you can use to create IaC. Ready made IaC templates can save you time, but you should always scan the templates for potential security risks. There are many templates available, but if you use one with vulnerabilities, you're likely introducing security problems into your infrastructure instead of making your infrastructure more secure.

IaC can use scanning tools like Terrascan, Checkov and TFLint to scan templates against policies. This can help find compliance violations and configurations that introduce security violations. If you're using pipelines like continuous integration/continuous delivery (CI/CD), you can scan IaC templates as part of those pipelines using control gates, and stop templates with potential risks.

Key takeaways

The declarative approach to IaC defines the desired state that your resources should have. The imperative approach to IaC defines the commands you need, and the order they need to be executed in to achieve the configuration you want. Infrastructure as code, coupled with automation, can help you with security tasks, including detecting drift and doing scans.