Google Cloud

Terraform and cloud security automation

So far, you've learned that Terraform is an infrastructure as code (IaC) tool that lets you define resources in a declarative way. One of the many things Terraform can do is use automation to keep your resources secure. In this reading, you'll learn more about Terraform and how it can help you with security. You'll also get some tips and best practices, and learn how Terraform can automate policy using policy-as-code (PaC).

Basics of IaC and Terraform in security

IaC can deploy code that has security measures built in as a baseline. This means that the code is "secure by default." To achieve this baseline, you need to apply security configurations to hosts, orchestration platforms, and supporting infrastructure during the creation of the infrastructure.

IaC has many benefits for security. For example, IaC helps ensure your infrastructure deployments are consistent. And, with IaC, secure configurations can be enforced, and are immutable. This means there's less chance for new backdoors or vulnerabilities. And, IaC lowers the chances your access controls are left wide open.

Terraform is a tool that helps deliver consistent infrastructure. It manages resources like compute, networking, and storage. It also manages some high level components like Software as a Service (SaaS) features, and domain name system (DNS) entries. An IaC tool like Terraform allows you to deploy infrastructure with secure configurations. This prevents your infrastructure from being changed, and avoids drift in the hosts, networks, and platforms you're using.

There are several tool options that can help you set up IaC, but Terraform is one of the most popular choices. As an added bonus, Terraform works with a number of major cloud providers, including Google Cloud.

Security with Terraform

Before you attempt to deploy IaC with Terraform, consider these questions:

- What does the infrastructure you're deploying need?
- What security configurations are required?
- How can you keep monitoring and auditing your infrastructure to ensure it hasn't been changed after initial deployment?

When deploying IaC with Terraform, you need to continuously audit:

- Role-based access controls (RBACs)
- Root access, and any other privileged access
- Secrets; used or embedded
- Definitions of services
- Any connections allowed from the network, including ingress (entering) or egress (exiting) traffic
- Misconfigurations that can lead to meddler-in-the-middle attacks

Have your team build everything you need into your scripts. If you must use a custom script, be sure to document your reasoning and include a plan to depreciate the custom script. During this process, there are some essential steps that can set you up for success, including:

- Properly documenting your process
- Explaining the reasoning behind your decisions
- Thoroughly testing scripts
- Committing scripts to version control

If you have to apply any changes temporarily without scripts, for example through the CSP's GUI, make sure these changes are also reflected in your scripts. Or, these changes will likely be reverted when the scripts are executed the next time. This helps ensure traceability and accountability.

Note: You can use Terraform resources and modules directly from your cloud provider because your provider has likely secured and curated them.

Terraform vital concepts

Here are some useful concepts for understanding how Terraform works:

- Workspaces are security boundaries within an organization. Things like variables, log output, state, and secure shell (SSH) keys are all within a workspace.
- You can group related workspaces in your organization using projects.
- You can grant permission for teams to do things like reading, planning, writing, or admin.
- You can set up a customized set of permissions.

Key takeaways

Remember, it's important to consider what your infrastructure's security needs are before deploying infrastructure using Terraform. If you decide Terraform is the best option, it's beneficial to implement any changes to the environment in scope through Terraform. This will help you avoid inadvertent roll-backs of manually applied changes when the scripts run the next time. Understanding Terraform, how it works, and when it applies will increase your organizational efficiency, and bolster your security posture.