Google Cloud

Key considerations when writing policy as code

Throughout your journey, you've learned that Policy as Code, or PaC, defines, manages, and automates policies, rules, and conditions using a high-level programming language. PAC is a subset of infrastructure as code. In this reading, you'll learn more about PaC use cases, and key considerations when writing PaC.

PaC use cases:

Creating and setting up infrastructure

You can use PaC to help you automate governance. To do this, first establish your governance policies, then explain why you're choosing these specific controls, and how they relate to your risk management program. It's important to note that the PaC must align with the governance already in place. Next, you can use PaC to automate policy checks around security, operations, and compliance. PaC makes the application of policy, procedures, and standards easier and more predictable. Application becomes easier because it can be automated with code. And, it becomes more predictable because policy does not have to be applied manually each time.

Authorization access control

You can also use PaC to enforce policies for authorization of a service. For example, you can make a policy to enforce multi-factor authorization (MFA) for all users. Or, you can create a policy to lock accounts after a certain number of unsuccessful login attempts.

Audits and compliance

Organization, industry, and government standards usually require specific controls for cloud services. PaC allows you to codify these controls early in the development process. Once PaC is aligned with governance, PaC can be used to audit applied controls to align with stated policies.

Then, applied controls can be audited to ensure that the PaC was successful, and that the code achieves what was intended.

Decision logs showing the policy, input, and query results enable you to audit policy decisions.

Key considerations when writing PaC

- First, ensure that the PaC is aligned with your organization's stated policy. For example, if your company's policy states that all PHI must be encrypted while not in use, the security team would need to check if the PaC is applying cryptographic services, like Key Management Services (KMS) to the data stores containing PHI.
- Second, ensure that the PaC will work as intended. Like with any code being deployed, this usually means to first deploy it to testing and staging environments. It can also be deployed with controls set to operate in Dry Run mode as a way to ensure it runs properly. Dry Run mode means that controls will be logged like they're enforced, but will not actually be enforced yet.
- Third, ensure that the fully tested set of controls is then applied into the production environment in a controlled way, like a staged rollout.
- Fourth, ensure that all controls deployed are also being monitored and logged. A set of detective controls need to be a part of any control set, and they also need to be tested.
- Fifth, controls should always be updated on the golden PaC build. The golden PaC build refers to the idea that there is a single best build of the controls that will be implemented over and over. The controls should not be updated manually, because manual entry can cause misconfiguration, and defeats the purpose of PaC.

Key takeaways

PaC automates policy by using high level programming languages, like GoLang used by Terraform. PaC also helps you automate governance, authorization and access controls, and audits and compliance. To write effective PaC that does NOT interfere with expected business functionality of applications, you'll need to ensure that your PaC is up to date, functions as expected, and is aligned with governance.