Checklist: Six power rules for integrating cloud security and GitOps

bridgecrew BY PRISMA® CLOUD

GitOps is an operating model that applies version control, collaboration tools, and CI/CD used in application development to infrastructure automation. In the GitOps model, the open source version control system Git is the single source of truth, making all changes visible and verifiable.

Below are **six impactful guidelines** for integrating cloud security and GitOps to increase release velocity while maintaining and improving your security posture.

1. Embed security scanning early

In GitOps, the only interface that developers interact with is Git. The rest of the integration and deployment process is automated, enabling complete version control and auditability of all infrastructure templates.

In this model, developers do not need access to the CI or CD platforms, nor the production environment. This limited access enforces the security principle of least privilege, shrinking the attack surface.

- ✓ Embed security into integrated development environments (IDEs) or as a pre-commit hook to surface feedback to developers similar to feedback in unit and integration tests.
- ✓ Scan your infrastructure codebase in the version control system (VCS) and continuous integration (CI) pipelines.

2. Enforce policies across the development cycle

It might seem redundant, but be sure to enforce security and compliance policies at every part of the GitOps flow. Scanning IaC templates before commit (and continuously through deployment) provides consistent, frequent feedback to help developers to catch and fix misconfigurations quickly.

- ✓ Periodically scan templates throughout the development lifecycle to catch new issues in existing code.

3. Provide quick, actionable feedback

To take advantage of GitOps speed and agility benefits, take action quickly when security feedback gets surfaced. Blocking builds or deployments without guidance creates friction—and developers will likely try to find workarounds. Be sure to keep developers in mind when embedding and enforcing guardrails.

- ✓ Ensure that security findings include prioritization and remediation guidance with offending line items.
- ✓ Employ an IDE extension or platform to provide the actual fixes.

4. Be aware of inevitable cloud drift

Managing cloud provisioning processes with infrastructure as code (IaC) creates a level of predictability. But no matter how airtight your IaC implementation is, out-of-band changes (known as drift) are inevitable.

As the source of truth and a key component of GitOps, IaC templates should be enforced by reconciliation loops and immutable infrastructure. These loops continuously check the desired state specified by the IaC templates against the actual state of the resources in production. Updates to infrastructure, such as operating system updates, wipe out the old instances, and stand up new ones.

- ✓ Create a drift detection strategy to pinpoint when drift becomes a risk.
- ✓ Implement drift detection automation relevant to your stack.
- ✓ Classify and route a response to the right individual or team.
- ✓ Only make changes to your cloud environment in IaC templates hosted and version-controlled in your repository.

5. Track progress over time

Checking infrastructure against policies and enforcing feedback improves the security of your codebase and deployed infrastructure over time. You can track your progress by measuring the number of existing misconfigurations, those patched, as well as new misconfigurations introduced. You know you're doing something right when these metrics decrease over time.

- ✓ Measure identified misconfigurations, patches, and new misconfigurations over a period of time, such as weekly or monthly.
- ✓ Create a benchmark or SLA for how long it takes to patch identified misconfigurations, and take steps to reduce this time to remediate.

6. Encourage collaboration with developer and security teams

In GitOps, collaboration is vital for success. Developers cannot blindly deploy infrastructure without passing the proper security guardrails, and security cannot assume those guardrails are perfect. Developers and security teams need to work together to iterate and improve on the process and tooling.

- ✓ Reduce noise by tuning feedback alongside security teams to determine what can pass through without remediation.
- ✓ Create a method for security teams to provide feedback on what is and isn't working. Developer teams to continue reinforcing the positive feedback loop.

Get started with Bridgecrew to:

Find and fix misconfigurations in cloud resources and IaC.

Enforce hundreds of built-in policies across security and compliance benchmarks.

Embed guardrails via IDE plugins, pre-commit hooks, and native VCS and CI/CD integrations.

GET STARTED FOR FREE