Standardize secrets management across development and production with Gitlab and Hashicorp.

Businesses and their products / services across all industries have become increasingly dependent on software to connect with their audience and differentiate in the market. However, an increase in software dependency has led to security concerns as well. Security vulnerabilities in an organization can span from flawed code in an app at runtime to hacked / leaked access of credentials to unintended and unauthorized bad actors.

Enterprise security has largely consisted of built in, bolt on, or static operational security approaches. Historically, these processes have been managed manually without automation and are error prone.

Some example include:

- Network based ACL with perimeter firewalls
- Document based user operations
- Reactive security remediation

Covering all possible failure domains at all times is an ever-evolving effort which requires companies to be flexible and proactive which can lead to an increased demand on human resources -- particularly as system architectures become more complex with microservices and API driven.

Together, GitLab and Hashicorp provide a comprehensive solution to shift security left, automate operations, and standardize on secrets management across the SDLC.

- GitLab CI/CD with Security Scanning helps deliver vulnerability-free code
- Vault encrypts and manages access to sensitive data and credentials via APIs
- GitLab and Vault enforce policy and governance across DevSecOps workflows

Product Integration

Vault is a single security control plane for operations and infrastructure that many organizations have chosen to manage secrets, PKI Certificates and other sensitive data. Vault enables secure secret storage, dynamic secret generation, auto renewal and revocation of secrets. GitLab is a complete DevOps platform delivered as a single application for the SDLC process. Together, GitLab and Vault can be configured together to standardize the security of secrets, tokens, credentials, etc. across development and production.

GitLab serves as a center to automate security via codifying these approaches. Vault's REST API can be seamlessly invoked within the GitLab CI/CD pipelines and integrated into the automation scripts as necessary. Additionally, GitLab SCM can store and version control Sentinel policies for Vault Enterprise as a single source of truth across teams. GitLab security scanning and dashboards functionality provide actionable information within the DevSecOps workflow by shifting security left.

When used together, this joint solution provides an alternative to the error-prone, document-based collaboration methods used across teams and allows organization to migrate from bolting on security to shifting security left.

How It Works

Before an application can retrieve secrets from Vault, it needs to login using an Authentication method. Vault supports a range of Authentication methods to allow flexibility in identifying an application regardless of where it is deployed. GitLab CI Runners offer similar flexibility in being able to execute anywhere. They can seamlessly authenticate to Vault based on their identity such as a Cloud IAM Role, a Kubernetes service account token, a JSON Web Token (JWT) and so on.

In the case of the Vault JWT Auth Method, each GitLab CI job has JWT that is issued by GitLab.com or a GitLab EE/CE server instance. This JWT is provided as an environment variable named CI_JOB_JWT, and can be presented to Vault to authenticate GitLab CI worker processes. Vault will cryptographically validate the JWT by contacting GitLab JWKS endpoint (https://gitlab.example.com/-/jwks). A Vault ACL Policy will be assigned to the GitLab worker based on specific claims present on the JWT. For additional details on how to configure Vault and GitLab CI for using the JWT Auth method, read more here.

Alternatively, you can choose from a range of other Authentication methods to identify GitLab CI Runners. The outcome is that the secrets management process is automated and audited, with the principle of least privilege being enforced. For additional information on GitLab's Secrets Management product direction, read more here.

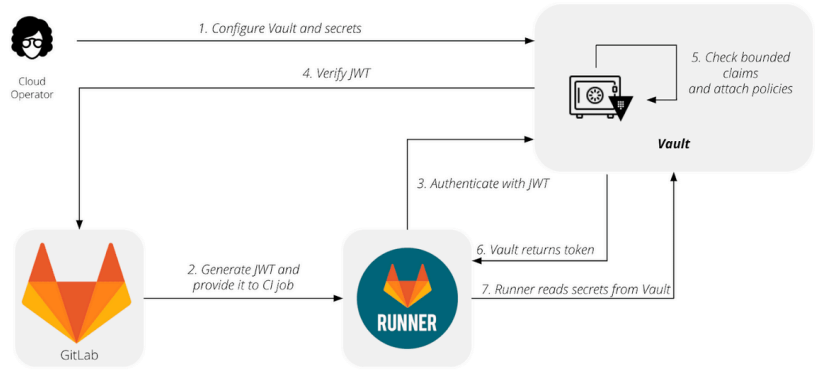

Figure 1: *Example of GitLab CI using Vault's JWT Authentication method to login and fetch secrets.*

Use Cases

Use Case 1: Standardize Secret Management across Dev and Prod.

A challenge in the application delivery lifecycle is to maintain consistency when promoting code from lower to higher environments, including how secrets are injected into applications. Vault can be configured as the general secrets management provider for GitLab across delivery into Dev and rollout into production.

When quickly spinning up secure development environments, users can install Vault into a Kubernetes clusters from within GitLab to better align with real production environments. From a Kubernetes cluster, a GitLab Runner can use the Service account JWT issued by Kubernetes to login to Vault and fetch secrets. Least privilege access can be enforced by mapping a Vault policy based on the Kubernetes service account name or the Namespace. Alternatively, practitioners using GItLab CI/CD can easily authenticate via the JWT Auth method described above. Using Either method, CI/CD processes can securely and automatically consume variables, secrets, service account credentials and even Cloud credentials from Vault as necessary.

Use Case 2: Cloud credentials for multi-Cloud provisioning and deployment

When provisioning to Cloud platforms at scale, a challenge is to issue temporary credentials that have least-privilege permissions. Vault solves this problem by managing the full lifecycle of credentials. This capability is known as Dynamic Secrets and it is especially useful when managing Cloud resources at scale, potentially across multiple Cloud platforms.

Jobs in GitLab CI/CD need Cloud API credentials to interact with platforms. GitLab workers can automatically login to Vault and retrieve dynamic Cloud credentials in a just-in-time model. Each Job is issued a unique credential that will be revoked by Vault after a pre-defined TTL, or once the Job is over.

In this model, there is no need for human operators to manually issue or rotate credentials, or configure credentials as CI variables. Developers can focus on writing code and not how credentials are issued for deploying their code. Similarly, operations and security teams can benefit from mitigating the risk of static, shared, plaintext credentials and being able to audit each credential issuance if needed.

Company Info

GitLab is a leader in DevOps platform solutions working to make it easier to manage CI/CD jobs. Check out our integration with HashiCorp Vault and our Secrets Management direction.