Amar Lojo

Solutions Engineer at HashiCorp
he/him

amar@hashicorp.com
GitHub: github.com/lomar92

Packer & HCP Vault Better Together

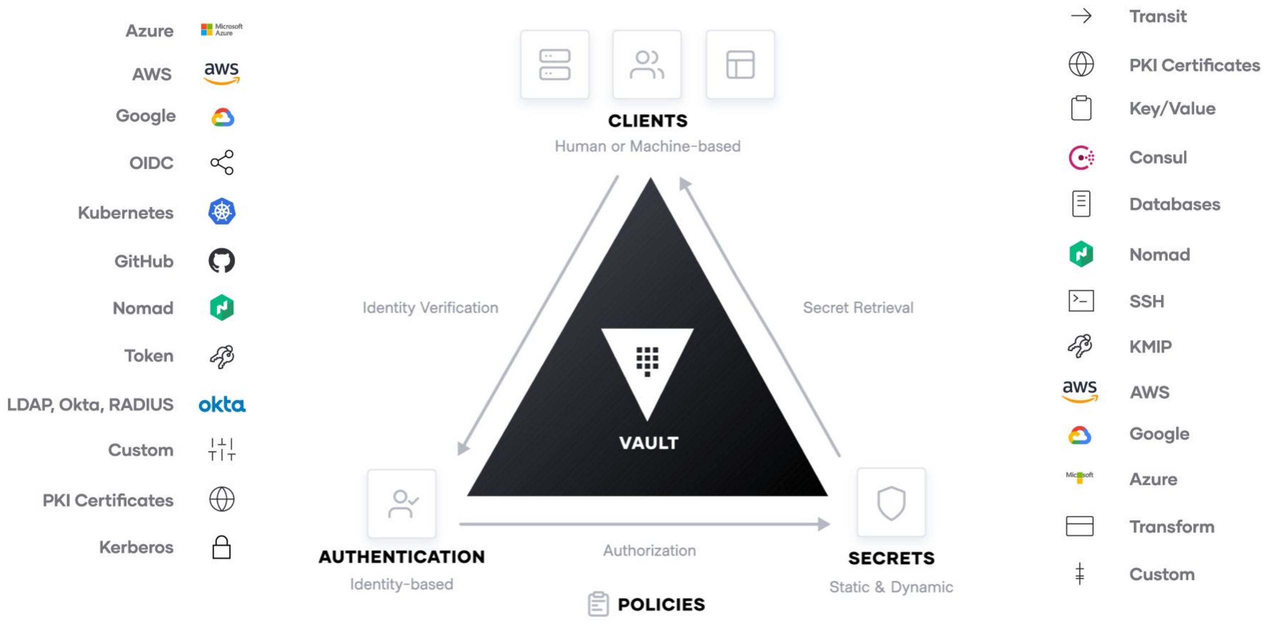
GitHub | AWS Secrets | Vault Management

Problem Statement

- Secrets Management für CI CD/GitHub Actions
- Secrets sind verteilt in Repos und Pipelines
- AWS Secrets sind statisch
- GitHub Auth Engine für Vault nicht ausreichend
- Zero Trust
- Einfaches Management von Vault

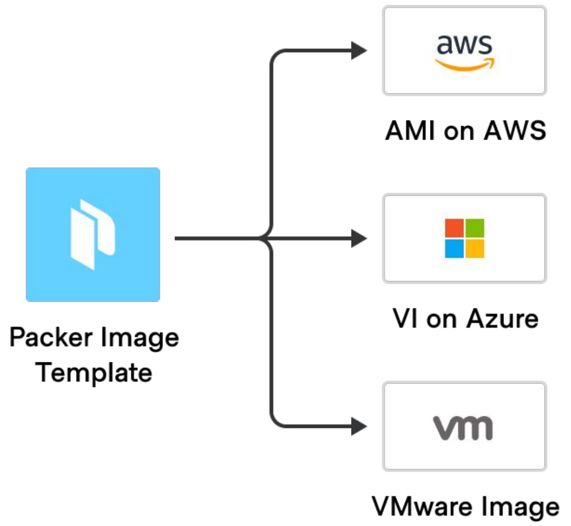
Introduction

HashiCorp Vault



Introduction

Packer



- **Easy to read and write:** Leverages HCL2, making it easier to use alongside Terraform
- **Extensible:** Use one language to manage images across clouds and integrate with other configuration tools
- **Open source:** Use existing community templates, or write your own

HashiCorp Packer

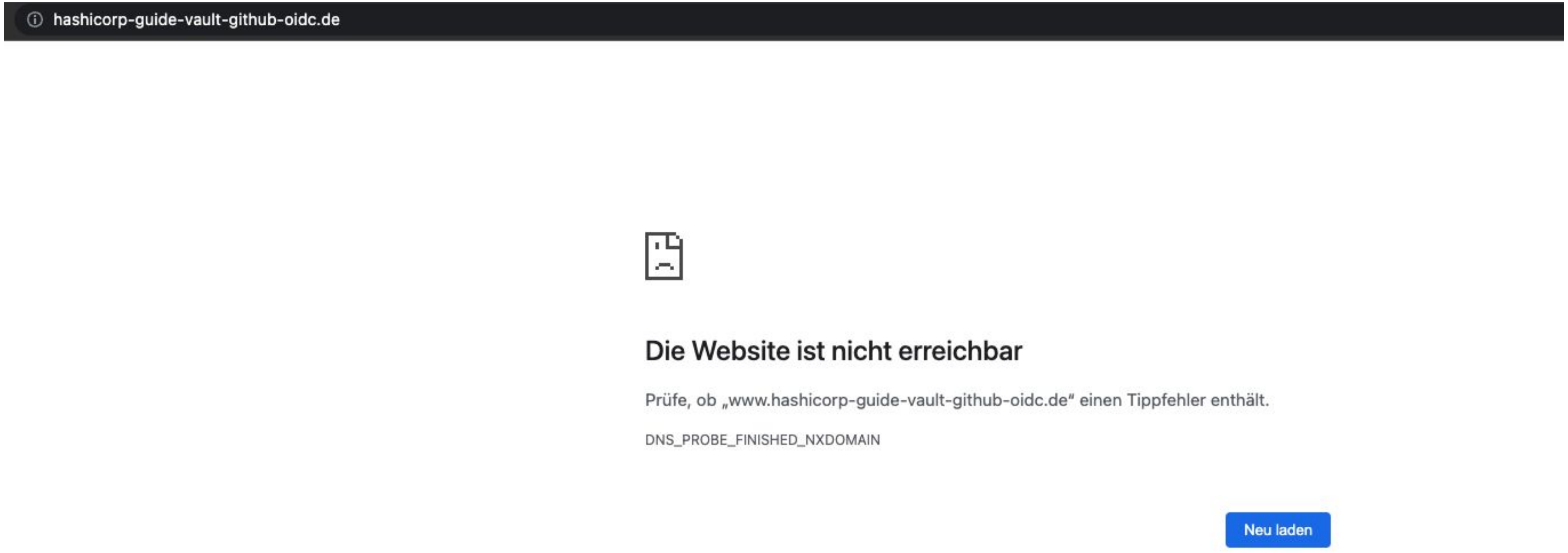
Creating a Golden Image Pipeline with HCP Packer & Terraform Cloud

Amar Lojo
Solutions Engineer

14. Dezember 2021

Creating a Golden Image Pipeline with HCP Packer & Terraform Cloud Demo

Wie lösen wir es?


hashicorp-guide-vault-github-oidc.de

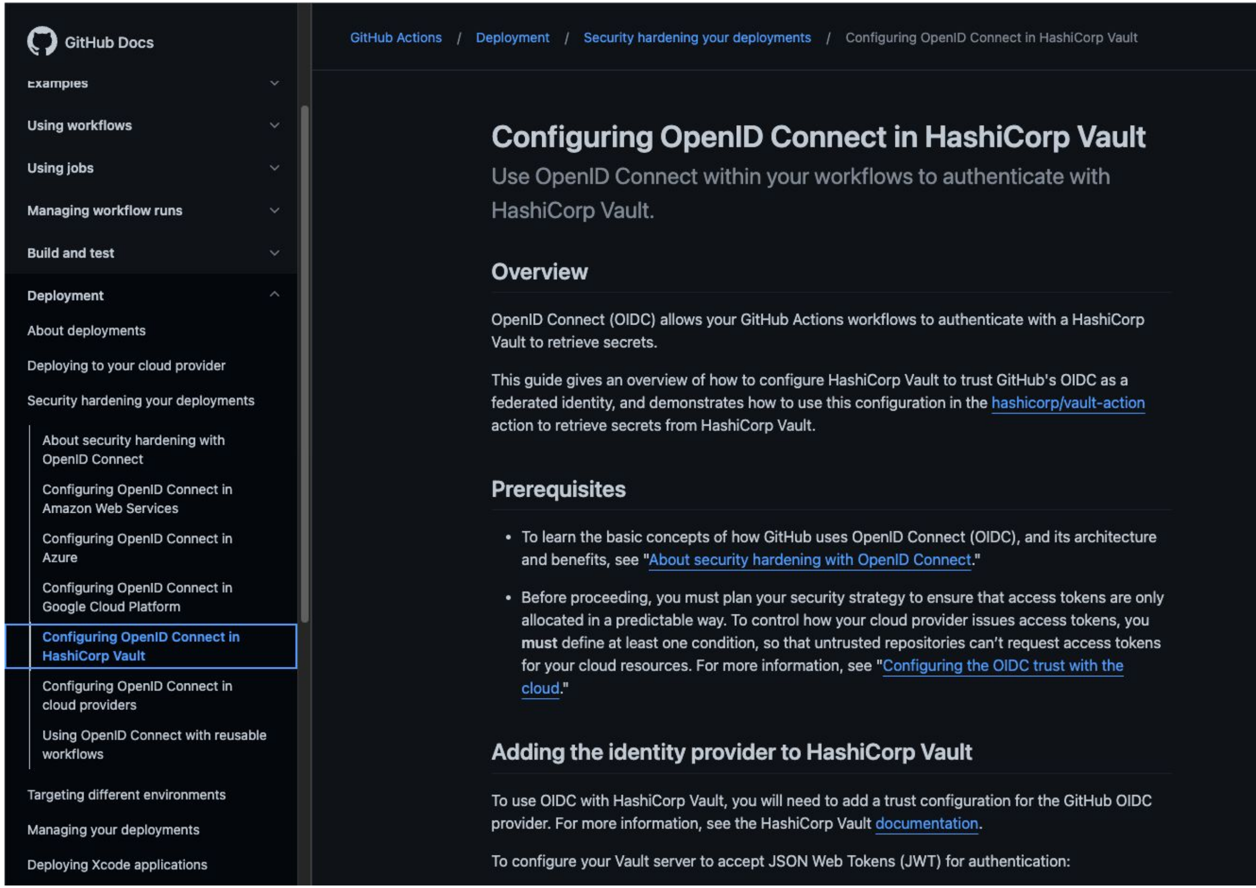
Die Website ist nicht erreichbar

Prüfe, ob „www.hashicorp-guide-vault-github-oidc.de“ einen Tippfehler enthält.

DNS_PROBE_FINISHED_NXDOMAIN

GitHub Doku

- [GitHub OIDC]()
- HashiCorp Vault support?



GitHub Docs

Examples
Using workflows
Using jobs
Managing workflow runs
Build and test
Deployment
About deployments
Deploying to your cloud provider
Security hardening your deployments
About security hardening with OpenID Connect
Configuring OpenID Connect in Amazon Web Services
Configuring OpenID Connect in Azure
Configuring OpenID Connect in Google Cloud Platform
Configuring OpenID Connect in HashiCorp Vault
Configuring OpenID Connect in cloud providers
Using OpenID Connect with reusable workflows
Targeting different environments
Managing your deployments
Deploying Xcode applications

GitHub Actions / Deployment / Security hardening your deployments / Configuring OpenID Connect in HashiCorp Vault

Configuring OpenID Connect in HashiCorp Vault

Use OpenID Connect within your workflows to authenticate with HashiCorp Vault.

Overview

OpenID Connect (OIDC) allows your GitHub Actions workflows to authenticate with a HashiCorp Vault to retrieve secrets.

This guide gives an overview of how to configure HashiCorp Vault to trust GitHub's OIDC as a federated identity, and demonstrates how to use this configuration in the hashicorp/vault-action action to retrieve secrets from HashiCorp Vault.

Prerequisites

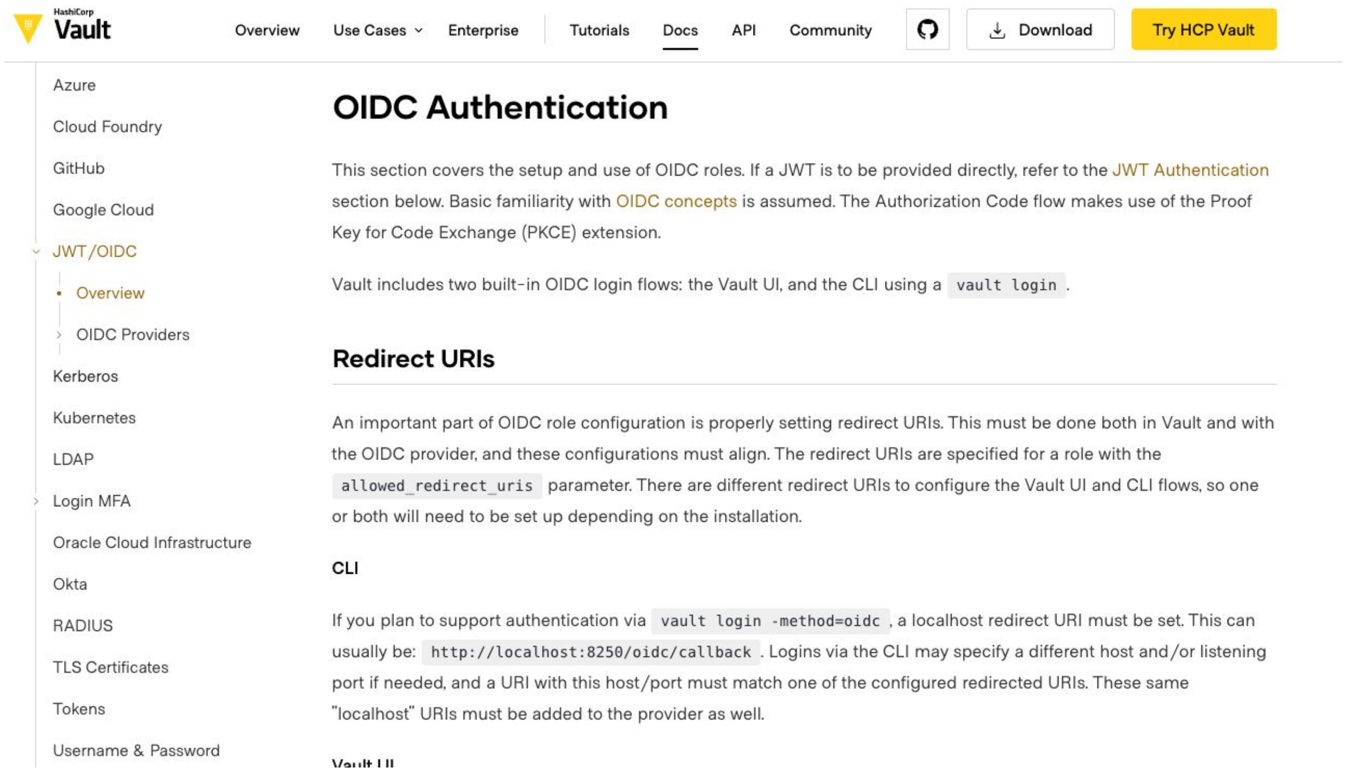
- To learn the basic concepts of how GitHub uses OpenID Connect (OIDC), and its architecture and benefits, see "About security hardening with OpenID Connect."
- Before proceeding, you must plan your security strategy to ensure that access tokens are only allocated in a predictable way. To control how your cloud provider issues access tokens, you **must** define at least one condition, so that untrusted repositories can't request access tokens for your cloud resources. For more information, see "Configuring the OIDC trust with the cloud."

Adding the identity provider to HashiCorp Vault

To use OIDC with HashiCorp Vault, you will need to add a trust configuration for the GitHub OIDC provider. For more information, see the HashiCorp Vault documentation.

To configure your Vault server to accept JSON Web Tokens (JWT) for authentication:

HashiCorp Doku



OIDC Authentication

This section covers the setup and use of OIDC roles. If a JWT is to be provided directly, refer to the JWT Authentication section below. Basic familiarity with OIDC concepts is assumed. The Authorization Code flow makes use of the Proof Key for Code Exchange (PKCE) extension.

Vault includes two built-in OIDC login flows: the Vault UI, and the CLI using a `vault login`.

Redirect URIs

An important part of OIDC role configuration is properly setting redirect URIs. This must be done both in Vault and with the OIDC provider, and these configurations must align. The redirect URIs are specified for a role with the `allowed_redirect_uris` parameter. There are different redirect URIs to configure the Vault UI and CLI flows, so one or both will need to be set up depending on the installation.

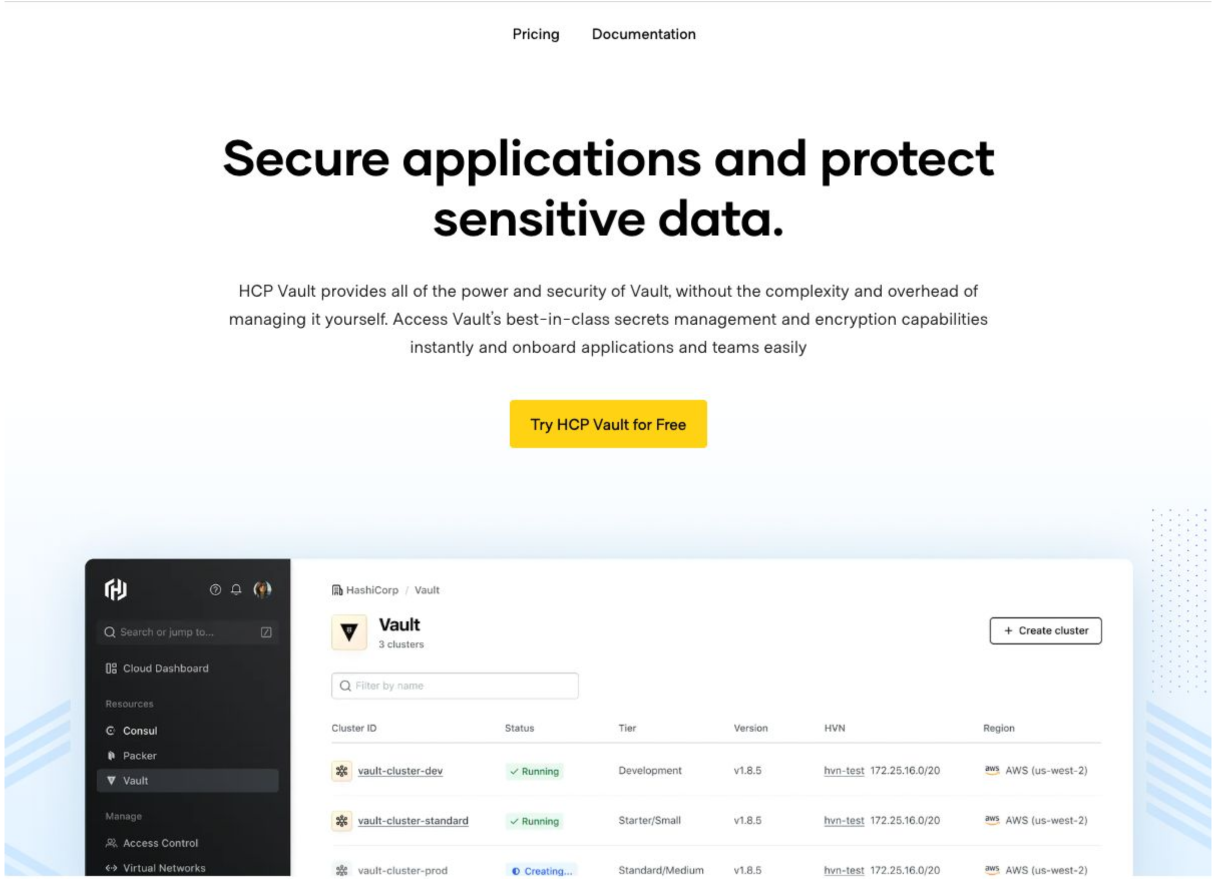
CLI

If you plan to support authentication via `vault login -method=oidc`, a localhost redirect URI must be set. This can usually be: `http://localhost:8250/oidc/callback`. Logins via the CLI may specify a different host and/or listening port if needed, and a URI with this host/port must match one of the configured redirected URIs. These same "localhost" URIs must be added to the provider as well.

Vault UI

Vault OIDC: https://www.vaultproject.io/docs/auth/jwt

Was muss konfiguriert werden?



Pricing Documentation

Secure applications and protect sensitive data.

HCP Vault provides all of the power and security of Vault, without the complexity and overhead of managing it yourself. Access Vault's best-in-class secrets management and encryption capabilities instantly and onboard applications and teams easily

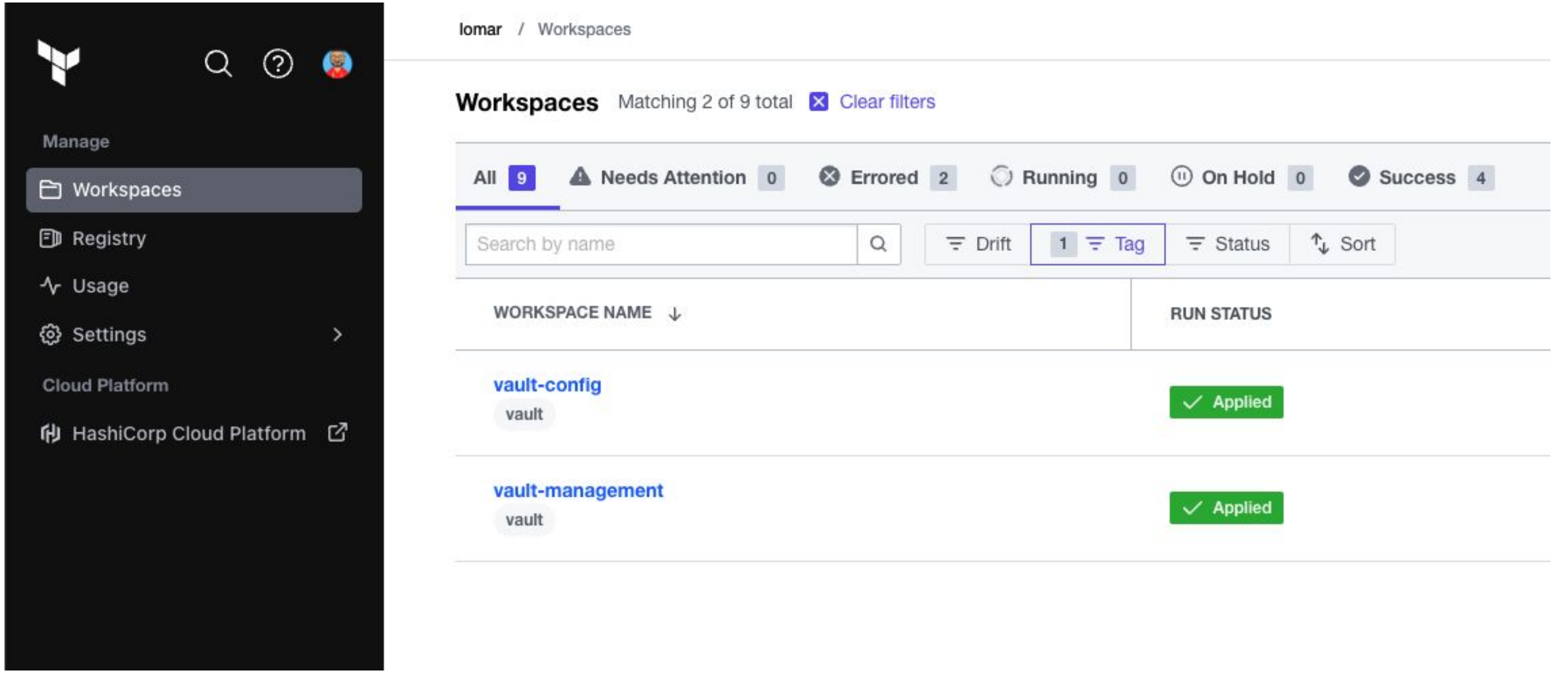
Try HCP Vault for Free

Vault
3 clusters

+ Create cluster

Filter by name

Cluster ID	Status	Tier	Version	HVN	Region
vault-cluster-dev	✓ Running	Development	v1.8.5	hvn-test 172.25.16.0/20	AWS (us-west-2)
vault-cluster-standard	✓ Running	Starter/Small	v1.8.5	hvn-test 172.25.16.0/20	AWS (us-west-2)
vault-cluster-prod	Creating...	Standard/Medium	v1.8.5	hvn-test 172.25.16.0/20	AWS (us-west-2)



lomar / Workspaces

Workspaces Matching 2 of 9 total Clear filters

WORKSPACE NAME	RUN STATUS
vault-config vault	✓ Applied
vault-management vault	✓ Applied

Auth Engine in HCP Vault

CODE EDITOR

```
resource "vault_jwt_auth_backend" "github_actions" {
 description        = "This is GitHub Actions JWT"
 path               = "jwt"
 oidc_discovery_url = "https://token.actions.githubusercontent.com"
 bound_issuer       = "https://token.actions.githubusercontent.com"
 default_role       = "github-actions"
 namespace          = data.terraform_remote_state.vault_infra.outputs.namespace
}
```

Auth Engine in HashiCorp Vault

CODE EDITOR

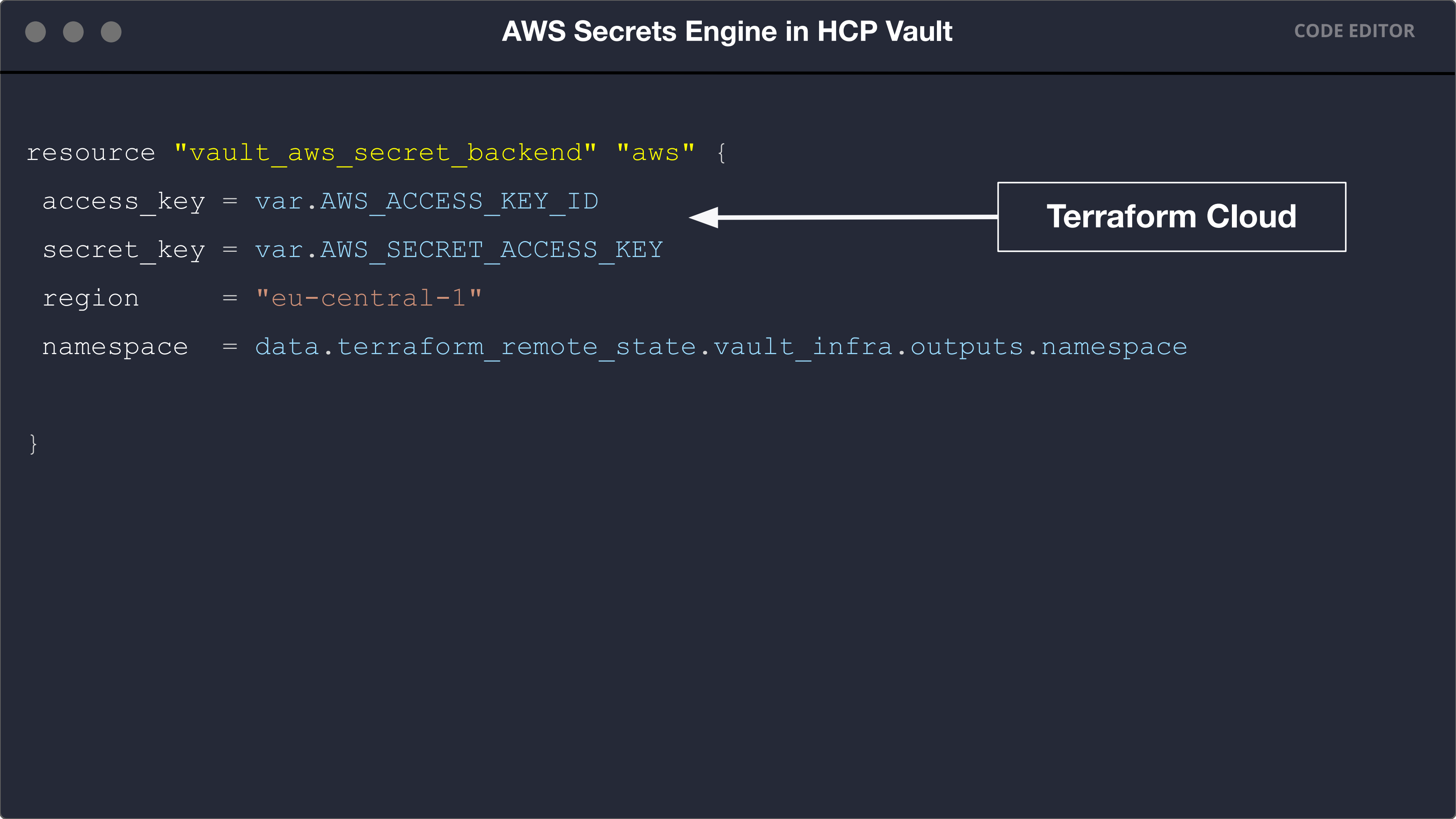
```
resource "vault_jwt_auth_backend_role" "github_actions"{
 backend        = vault_jwt_auth_backend.github_actions.path
 role_name      = "github-actions"
 token_policies = ["vault-actions"]

 bound_claims_type = "glob"
 bound_claims = {
   sub : "repo:hashicorp-dach/tf-vault-packer:ref:refs/*"
 }

 bound_audiences = ["https://github.com/hashicorp-dach"]

 user_claim = "workflow"
 role_type  = "jwt"
 token_ttl  = "1800"
 namespace  = data.terraform_remote_state.vault_infra.outputs.namespace
}
```

AWS Secrets Engine in HCP Vault



```
resource "vault_aws_secret_backend" "aws" {
 access_key = var.AWS_ACCESS_KEY_ID
 secret_key = var.AWS_SECRET_ACCESS_KEY
 region     = "eu-central-1"
 namespace  = data.terraform_remote_state.vault_infra.outputs.namespace

}
```

AWS Secrets Engine in HCP Vault

CODE EDITOR

```
resource "vault_aws_secret_backend_role" "role"  {
 backend         = vault_aws_secret_backend .aws.path
 name            = "github-actions-aws"
 credential_type = "iam_user"
 namespace       = data.terraform_remote_state .vault_infra .outputs .namespace

 policy_document = <<EOT
{
 "Version": "2012-10-17",
 "Statement": [
   {
     "Effect": "Allow",
     "Action": "ec2:*",
     "Resource": "*"
   }
 ]
```

Last Step

GitHub Konfiguration?

GitHub + GitHub Actions

Notes & Lessons Learned

- AWS Credential Management nach Least Privilege Ansatz
- AWS IAM oder AWS STS credentials werden unterschiedlich für die Pipeline verarbeitet bzw. zur Verfügung gestellt. “AWS Consistency”
- AWS Secrets Engine: Konfigurieren und anschließend Secrets rotieren.
- Vault mithilfe von Terraform managen
- GitHub Repo zum selber probieren: https://github.com/lomar92/tf-vault-packer

hug HashiCorp User Groups

hugs@hashicorp.com | learn.hashicorp.com | discuss.hashicorp.com

Thank You

hugs@hashicorp.com | learn.hashicorp.com | discuss.hashicorp.com