ADDO

ALL DAY DEVOPS

NOVEMBER 6, 2019

Shrivatsa Upadhye

@iShrivatsa

The 3 Musketeers: Jenkins, Terraform, Vault

whoami (Who am I?)

- Public Cloud Advocate @VMWare
- Focused on Cloud and Application Security
- Part of CloudJourneyIO team
- Love Cars and Pizza

Follow me on
Twitter/Medium/GitHub :
@ishrivatsa

Use Case

Automate deployment of cloud infrastructure

- Generate access tokens based on identity (user or machine)
- Securely storing the credentials for the cloud (AWS access key and secret key) along with application secrets
- Destroy the access keys once they are used for deployment to avoid any misuse
- Auditing the usage of credentials
- Ability to provision to any cloud provider via a pipeline

Jenkins

- Free and Open-source automation server
- Plugins available for integration with including Terraform, AWS, Azure etc.
- Ability to build sophisticated pipeline
- Supports Pipeline as Code

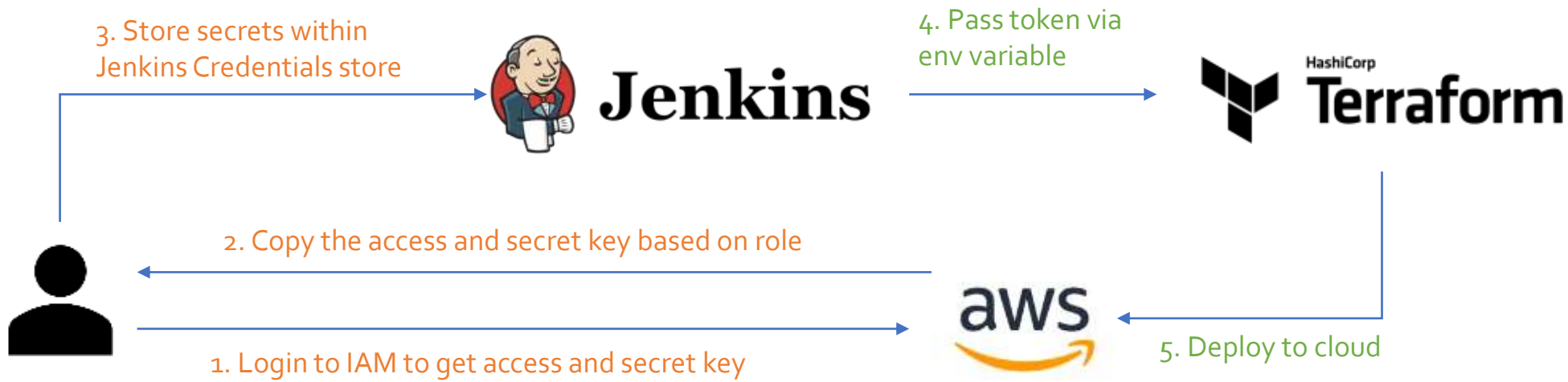
HashiCorp Terraform

- Free and Open-source infrastructure provisioning tool
- Provider plugins available for major cloud providers as well as on-prem platforms like vSphere
- Provides Infrastructure as Code (HCL)
- Support for Remote state management

HashiCorp Vault

- Free and Open-source secrets store
- Secrets engine (with encryption) supported for AWS, Azure, GCP along with K8s, simple key-value store and other services
- Authentication method like GitHub, AWS, LDAP, Okta etc., supported

Secrets in Jenkins Credential Store

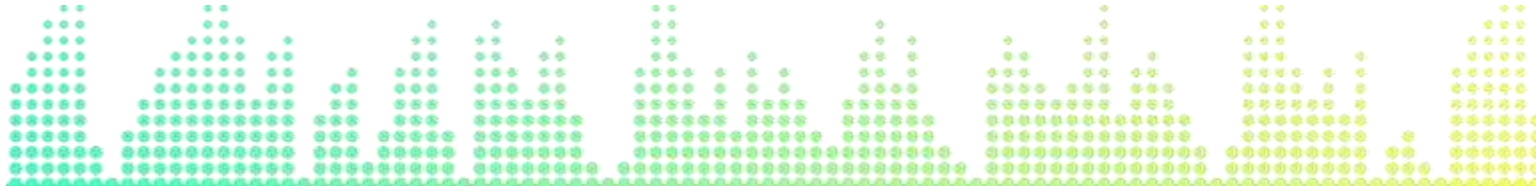


Why Not Jenkins Credentials Store?

- No ability to revoke the keys
- Need to manage different plugins for every cloud provider and application type
- No secrets/token lifecycle management like generation of secrets based on user/machine identity
- For applications, Hardcoding of secrets is **NOT** a good practice as there is the risk of these secrets being exposed on git repositories.

Why Vault?

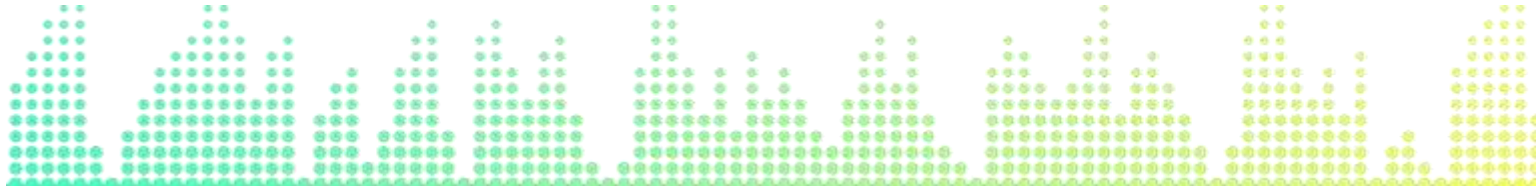
1. Centralized secrets store with encryption
2. Granular policies designed to control permissions to every key stored within the vault
3. Lease Period - Duration can be set for how long the access to secrets are allowed
4. Cloud Agnostic i.e. Works with AWS, Azure, and GCP , with support for dynamic secrets
5. Integrations for various opensource and licensed databases like Influx, Mongo, Hana as well as Kubernetes secrets



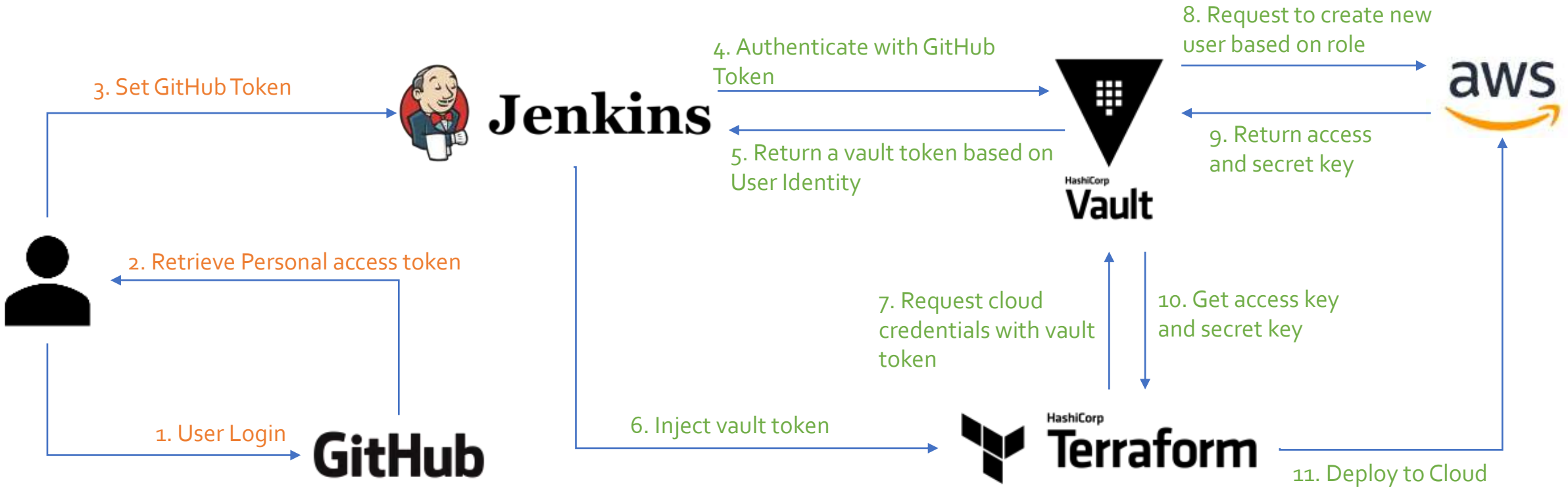
So.... how can you implement it?

There are 2 Primary ways of implementing this with vault

Method 1	Method 2
User-Identity based authentication	AppRole based authentication
Uses LDAP, GitHub, Okta etc., for authn and returns a token	Uses Application identity and needs role_id and secret_id to return token
Policies can be designed for orgs/teams/user	Policies can be designed based on application type
Provides audit logs of credential usage based on user identity	Provides audit logs of credential usage based on app identity

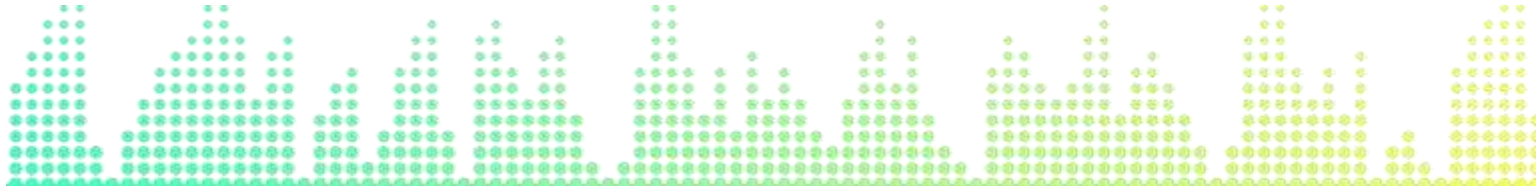


Vault with GitHub authentication



https://github.com/ishrivatsa/terraform-vault-jenkins

Demo



Summary

- Use Vault to store credentials for apps and infra
- Use Vault policies to restrict access to credentials
- Select 3rd Party Auth or AppRole authentication based on use case
- Always, version control the pipeline as code, terraform and vault policy templates

Thank You

www.cloudjourney.io | @cloudjourneyio

@iShrivatsa

For Code and Templates:
https://github.com/ishrivatsa/terraform-vault-jenkins

ADDO
ALL DAY DEVOPS