Implementation of New Security Features in CMSWEB Kubernetes Cluster at CERN

Aamir Ali[1], Aroosha Pervaiz[2], Muhammad Imran[2], Valentin Kuznetsov[3], Spyridon Trigazis[1], Andreas Pfeiffer[1] and Marco Mascheroni[4]
On behalf of CMS Collaboration

[1]CERN, Geneva, Switzerland
[2]National Centre for Physics, Pakistan
[3]Cornell University, USA
[4]University of California San Diego, USA

Email for queries: muhammad.imran@cern.ch

Agenda

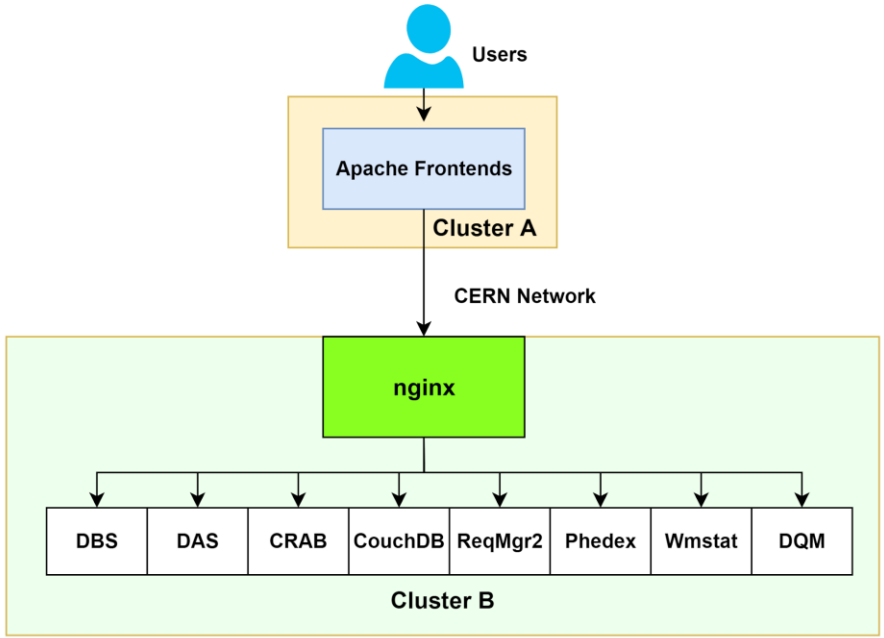
- Introduction
- CMSWEB Architecture
  - No of Requests
- Network Policies
  - Network Policy Example
- Open Policy Agent
  - OPA Gatekeeper
  - OPA Policies Examples
  - Deployment of OPA Policies with HELM
- Vault
  - Vault High Level Overview
  - Vault Integration with Kubernetes
  - Deployment Script
- Conclusion

Introduction

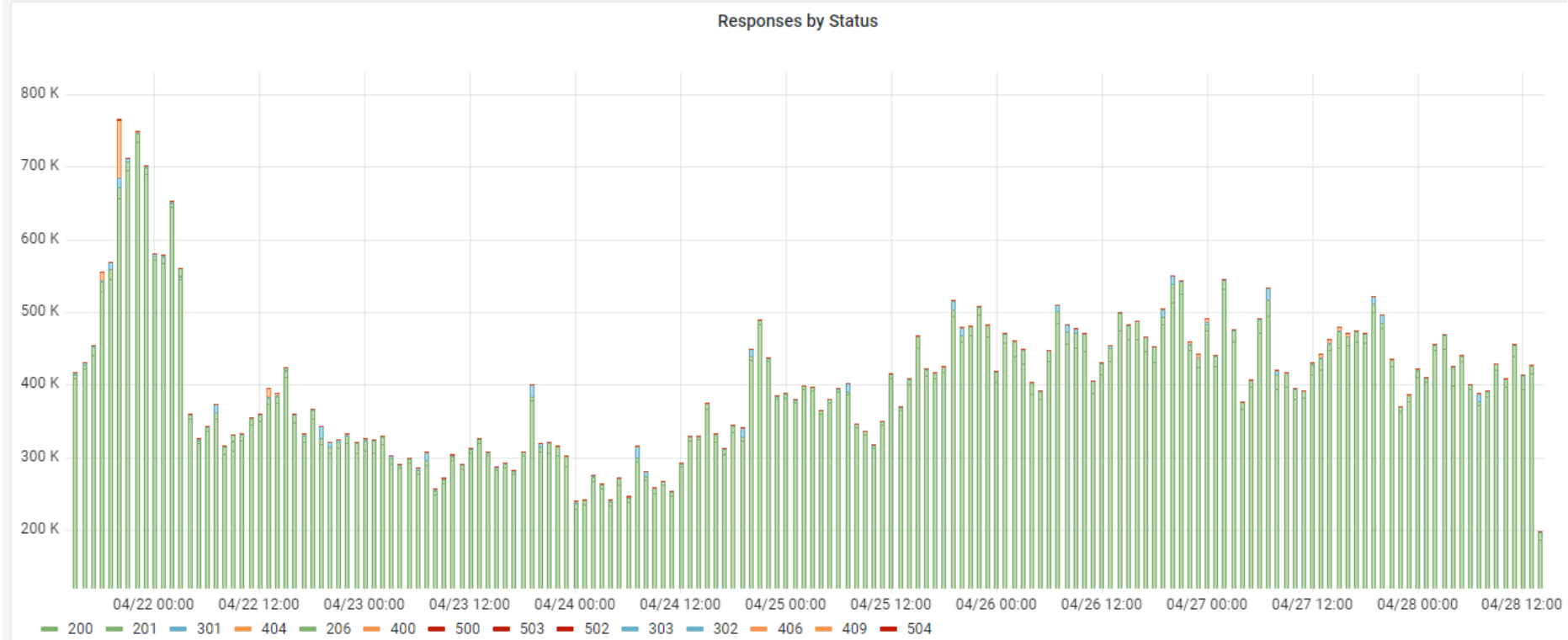
- The CMSWEB cluster is pivotal to the activities of the CMS experiment, as it hosts critical services required for the operational needs of the experiment.
- The security of these services and the corresponding data is crucial to CMS.
  - Any malicious attack can compromise the availability of our services.
  - Therefore, it is important to construct a robust security infrastructure.
- In this talk, we discuss some new security features introduced to the CMSWEB kubernetes ("k8s") cluster.
- The new features include:
  - The implementation of network policies,
  - Deployment of Open Policy Agent (OPA), and enforcement of OPA policies
  - The integration of Vault.

CMSWEB Architecture

- It has the two components:
  - **frontend cluster**
  - **backend cluster**
- The Apache frontend performs x509 certificates authentication and redirects the request to the backend cluster.
- On the backend cluster, the ingress controller has basic redirect rules to the appropriate services and only allows requests from the frontend cluster.



No of Requests



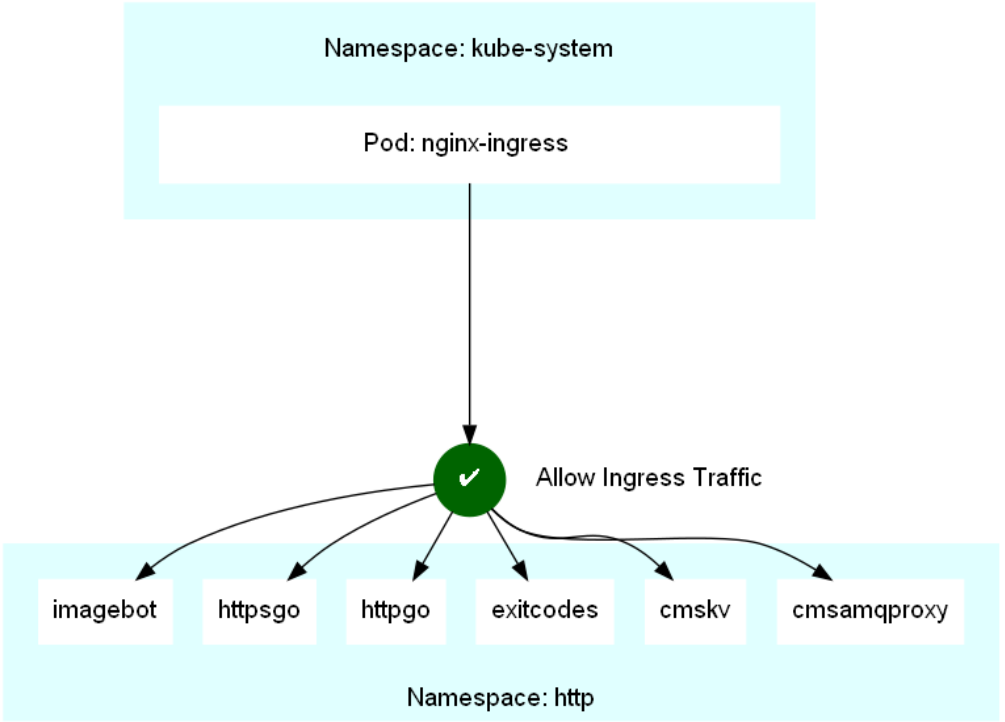
- CMSWEB cluster receives 0.5 million to 1 million requests per hour

1- Network Policies

- Network Policies are an application-centric construct which allow you to specify how a pod is allowed to communicate with various network "entities"
- They act as an inside-the-cluster firewall to limit the network communication between the pods to the minimum necessary, and its dynamic nature allows us to work with microservices.
- The network policies can be applied to following entities:
  - Other pods that are allowed
  - Namespaces that are allowed
  - IP blocks

Network Policy Example

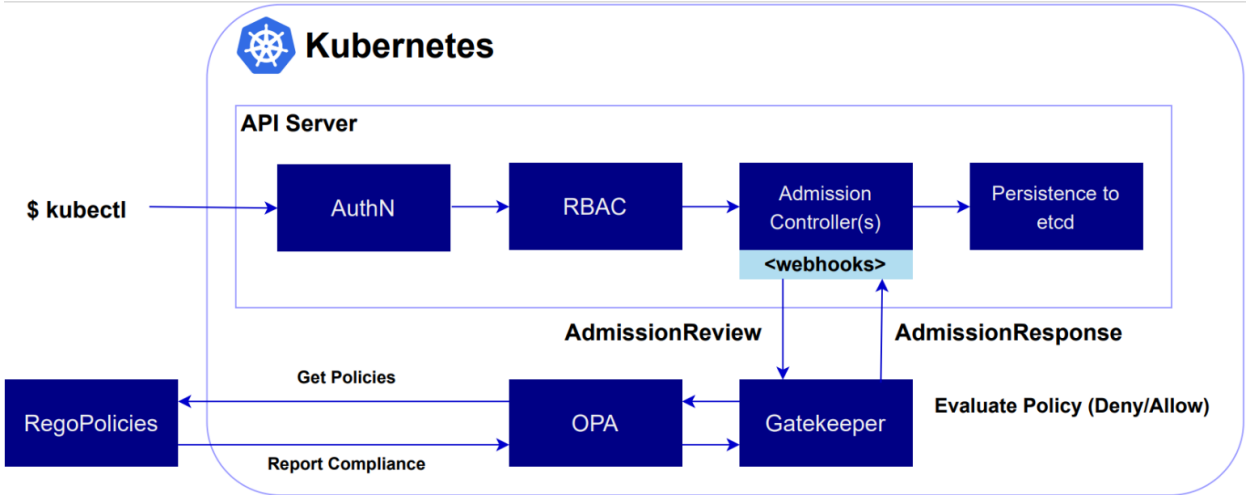
- Backend services pods in backend cluster will only receive incoming traffic from nginx-ingress pods that act as proxy between frontend and backend services.
- All other incoming traffic to all services pods is blocked.
- Additionally, nginx-ingress will only allow traffic from frontend cluster using IP addresses whitelisting feature of ingresses.



2- Open Policy Agent (OPA)

- An open source, general-purpose policy engine that unifies policy enforcement across the stack.
- It enables developers, operations, compliance and security teams to build and enforce consistent authorization policy at scale, enhancing security and reducing the manual burden placed on staff as a result.
- OPA lets you decouple policy from that software service.
- It uses Rego language for implementation of policies
- Common use cases include
  - application and microservice authorization,
  - Kubernetes admission control,
  - infrastructure policies and configuration management.

OPA Gatekeeper



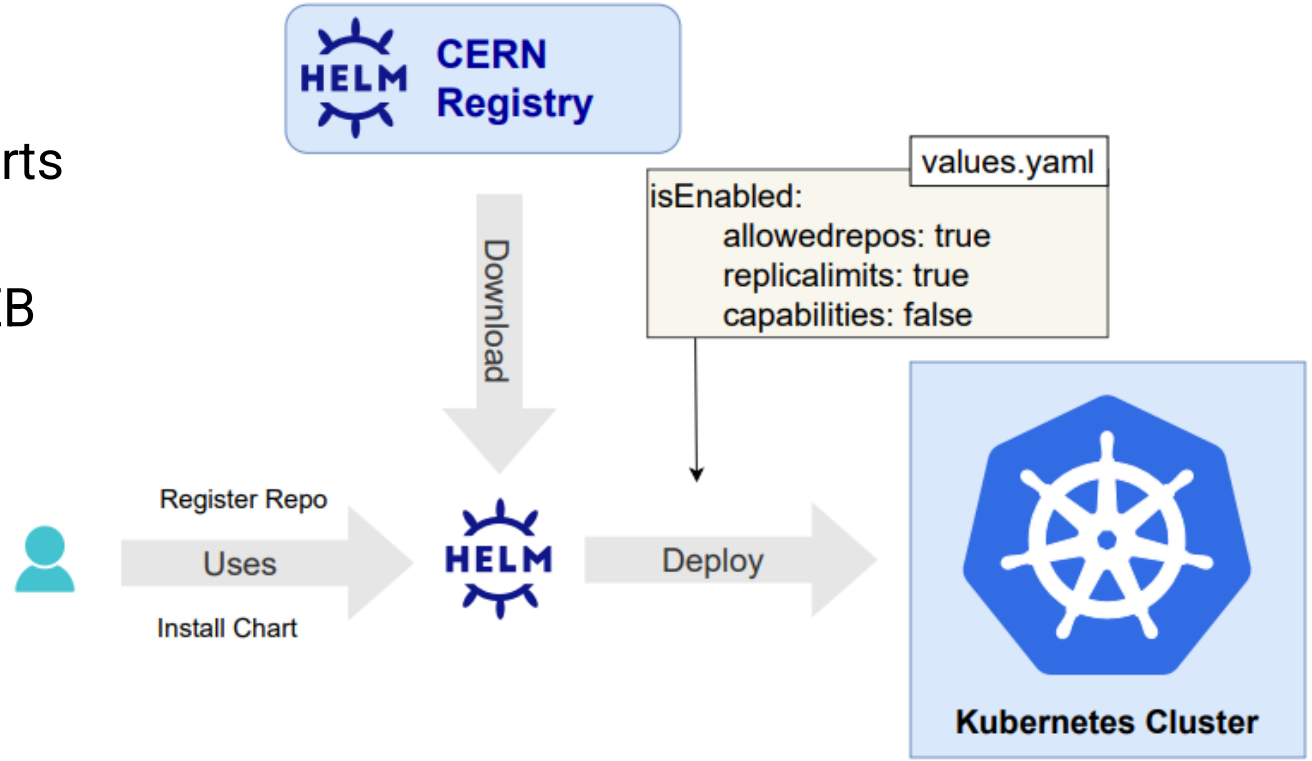
- When performing create, update, and delete operations on objects in Kubernetes, Admission Controllers enforce policies.
- Kubernetes enables the decoupling of policy decisions from the inner workings of the API Server.
- OPA Gatekeeper is a specialized implementation of OPA that provides integration with Kubernetes using Dynamic Admission Control and custom resource definitions to allow the Kubernetes cluster administrator and other users to create policy templates and specific policy instances

OPA Polices Examples

Policy	Description
Allowed Repos	Requires container images to be downloaded from specific repositories only.
Container Limits	Requires that containers have established memory and CPU limits, and that limitation be kept within the designated maximum values.
Container Requests	Requires that containers have their memory and CPU requests set, as well as that requests, stay under the permitted upper limits.
Container Resource Ratios	Restricts the maximum ratio of container resource limits to requests.
Disallow Anonymous	Prohibits assigning ClusterRole and Role resources to the system:unauthenticated group and system:anonymous user.
Replica Limits	Requires that objects (Deployments, ReplicaSets, etc.) indicate the number of replicas within specified ranges
Required Probes	Requires the readiness and/or liveness probes to be present in the Pods.
Host Namespaces	Prevents pod containers from sharing host PID and IPC namespaces.

Deployment of OPA Polices with HELM

- We created helm charts to incorporate OPA policies with CMSWEB K8s cluster

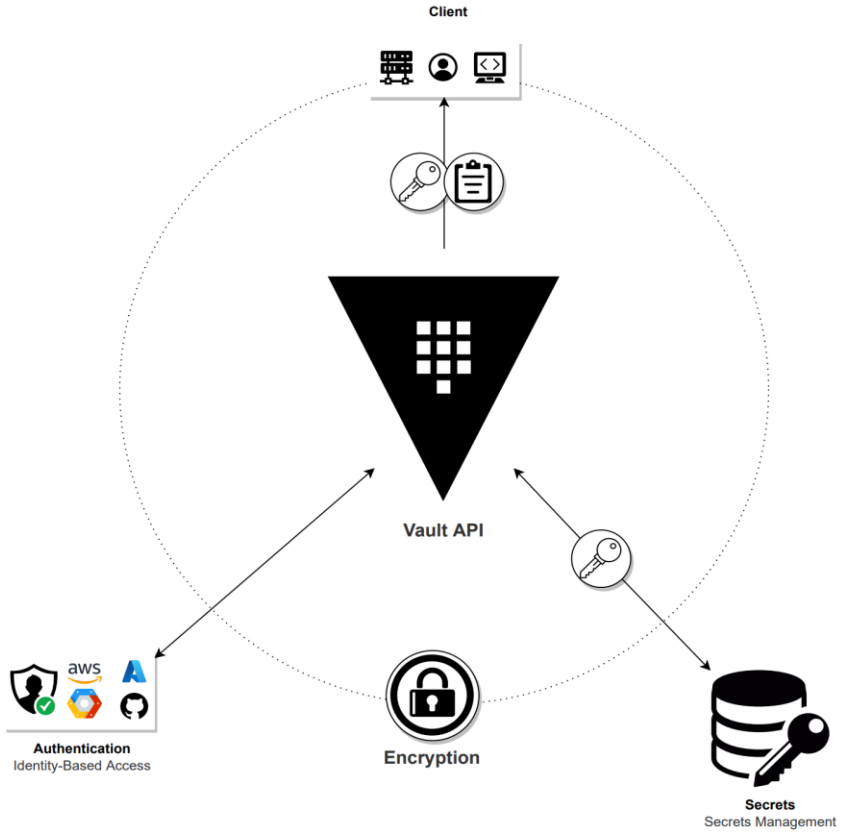


3- Vault

- HashiCorp Vault is an identity-based secrets and encryption management system
- Vault secures, stores, and tightly controls access to tokens, passwords, certificates, API keys, and other secrets in modern computing

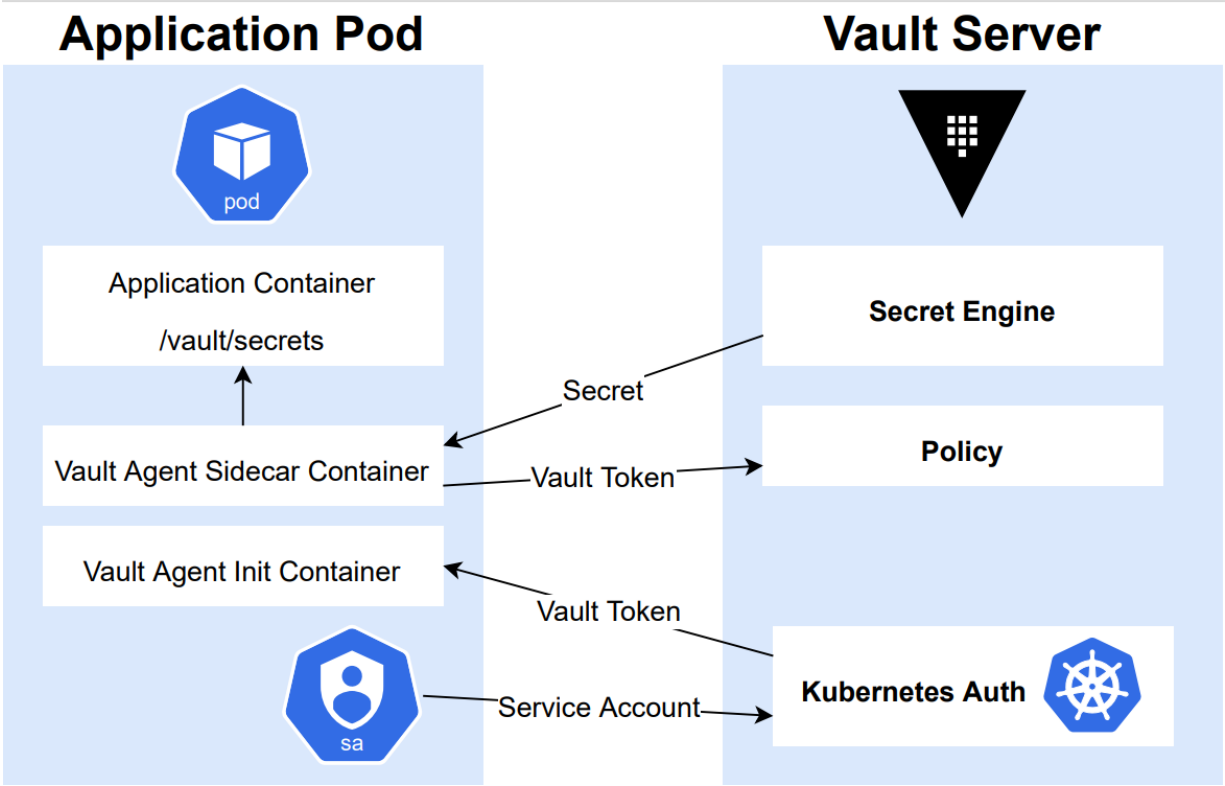
Vault High Level Overview

- Before granting users, machines, or applications access to secrets or storing sensitive data, it verifies and authorizes the clients (users, apps, and machines) via APIs.



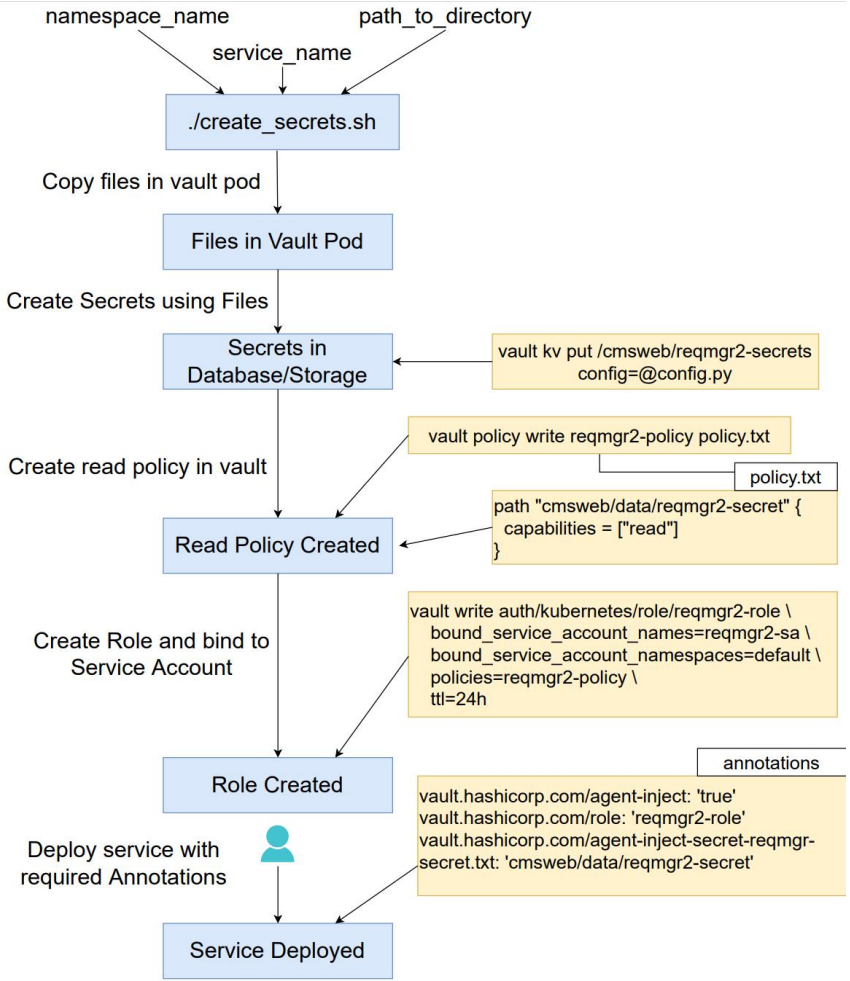
Vault integration with Kubernetes

- We deployed Vault using the helm chart provided by HashiCorp with custom configurations.
- Vault is deployed as a stand-alone server inside a pod in the vault namespace.
- The storage is statically created from OpenStack to keep data in case of accidental removal of the Vault server.
- The Kubernetes is configured as the authenticator. For secrets in CMSWEB, an encryption engine kv-v2 is enabled.
- A Vault Agent container that renders Vault secrets to a shared memory volume is added to the pod specs by the Vault Sidecar Agent Injector using the sidecar pattern.



Deployment Script

- To create secrets in CMSWEB, we have created a deployment script.
- The script requires 3 arguments: the namespace, service, and path to the directory that contains files.
- It copies files into the vault pod so it's accessible by the Vault CLI.
- Then it creates a secret using the service name and appending it with "-secrets".
- The secret is required to be key-value pair so names of files are used as keys and content as values.
- The script creates a policy that allows reading the secret.
- Finally, the script creates a role to bind policy with the service account.
- After running the script, the user is required to add annotations to the service pods for the Vault Agent Injector.



Conclusion

- In this talk, we discuss some new security features introduced to the CMSWEB kubernetes cluster.
- The new features include:
  - The implementation of network policies,
  - Deployment of Open Policy Agent (OPA), and enforcement of OPA policies
  - The integration of Vault.
- The new features enhances security and robustness in CMSWEB infrastructure.
  - Network policies are in-house guards against code vulnerabilities.
  - OPA Gatekeeper further ensures that nobody can trick the system by implementing fine-grained control over different activities that is not possible otherwise
  - Vault encrypts the secret and tightly controls access to the secret so a user, machine, or service has to authorize itself first before accessing the secret.