

Effortless Secrets Management in Kubernetes: Streamlining App Deployment with GitOps and ArgoCD using the HashiCorp Vault Injector



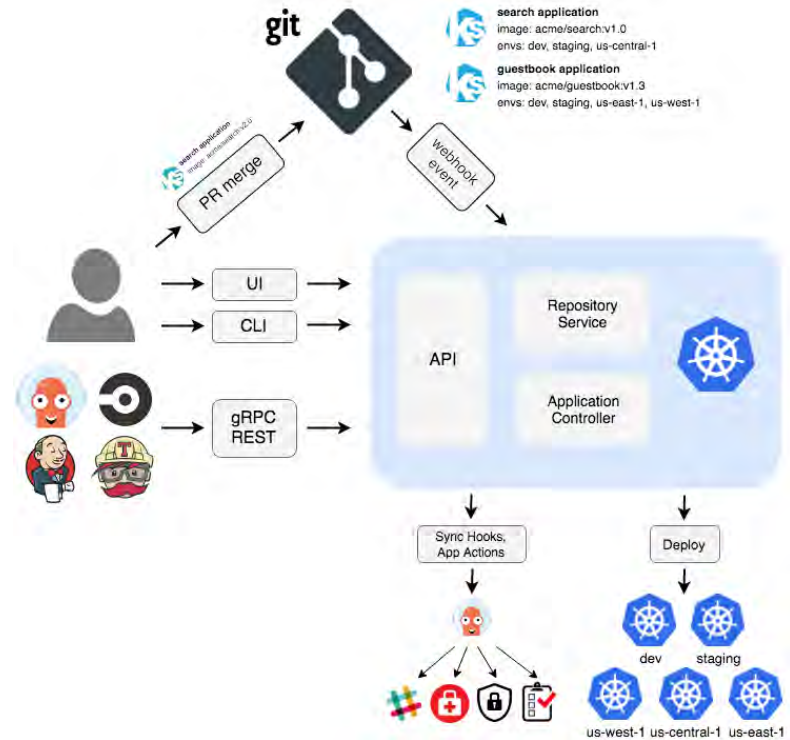
Presenter:
Sam Gabrail

Intro to GitOps with ArgoCD



Intro to GitOps with ArgoCD

- **ArgoCD** is implemented as a **Kubernetes controller** which **continuously monitors** running **applications**
- It **compares** the current, **live state** against the **desired target state** (as specified in the **Git repo**)
- A deployed application whose **live state deviates** from the **target state** is considered **OutOfSync**
- ArgoCD **reports & visualizes** the **differences**, while providing facilities to automatically or manually **sync** the live state back to the desired target state
- Any **modifications** made to the **desired target state** in the **Git repo** can be **automatically** applied and reflected in the specified **target environments**



Taken from the official docs <https://argo-cd.readthedocs.io/en/stable/>



School App Introduction



School App Introduction

A simple demo app for online courses

The screenshot displays the School App interface. At the top, there is a dark blue header with a hamburger menu icon and the text "School App". Below the header, a green "ADD COURSE" button is visible. The main content area is divided into two sections: "Courses" and "Students".

Courses

Title ↓	Author	Price	Actions
HashiCorp Vault 201 - Vault for Apps in Kubernetes	Sam Gabrail	139	

Rows per page: 10 | 1-1 of 1

Students

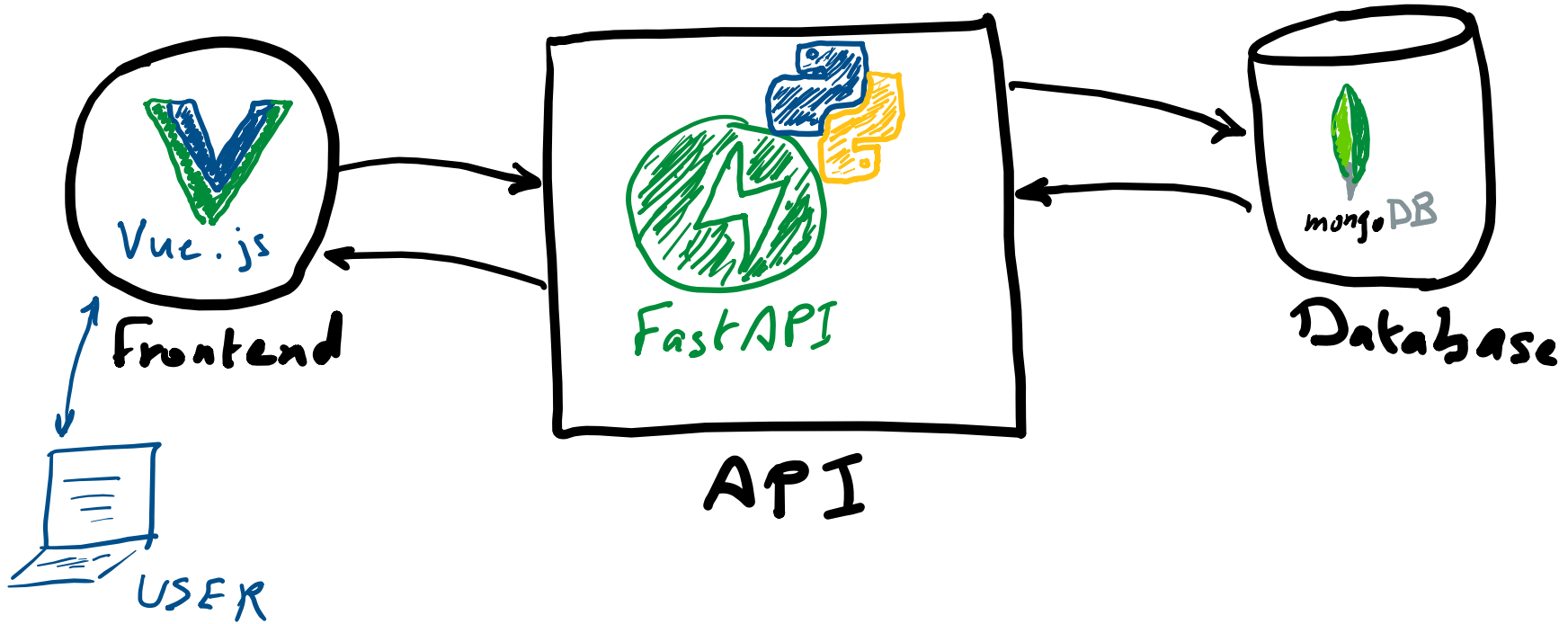
First Name ↓	Last Name	Email	Address	Credit Card	Course Enrolled	Actions
Sam	Gabrail	me@email.com	1234 Main St	1234-5678-9012-3456	HashiCorp Vault 201 - Vault for Apps in Kubernetes	

Rows per page: 10 | 1-1 of 1

The screenshot displays the School App interface showing a course detail view. At the top, there is a dark blue header with a hamburger menu icon and the text "School App". Below the header, the word "Courses" is displayed. The main content area shows a course card for "HashiCorp Vault 201 - Vault for Apps in Kubernetes" by Sam Gabrail. The card features a header image with the HashiCorp logo and a portrait of Sam Gabrail. Below the image, the course title and author are listed, along with a description: "Learn how to use HashiCorp Vault for your applications in Kubernetes". A green "ENROLL" button is visible at the bottom right of the card. The author's name "Sam Gabrail" and price "139" are also shown. At the bottom of the page, there is a dark blue footer with "HOME" and "ABOUT" links, and copyright information: "© SchoolApp 2022 - A Demo App by TeKanAid Solutions Inc. Version v0.5.8".



School App Components



School App K8s Output

```
(* | docker-desktop:schoolapp)
```

```
Gabrail-Windows:sam:~$kubectl get all
```

NAME	READY	STATUS	RESTARTS	AGE
pod/api-64f45f88f7-5tfst	1/1	Running	0	4m41s
pod/frontend-7fd5f756f5-md2x8	1/1	Running	0	4m16s
pod/schoolapp-mongodb-6cdf54d797-crs9f	1/1	Running	6 (7d23h ago)	32d

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/api	ClusterIP	10.98.188.115	<none>	5000/TCP	32d
service/frontend	ClusterIP	10.107.27.89	<none>	8080/TCP	32d
service/schoolapp-mongodb	ClusterIP	10.99.91.72	<none>	27017/TCP	32d

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/api	1/1	1	1	32d
deployment.apps/frontend	1/1	1	1	32d
deployment.apps/schoolapp-mongodb	1/1	1	1	32d

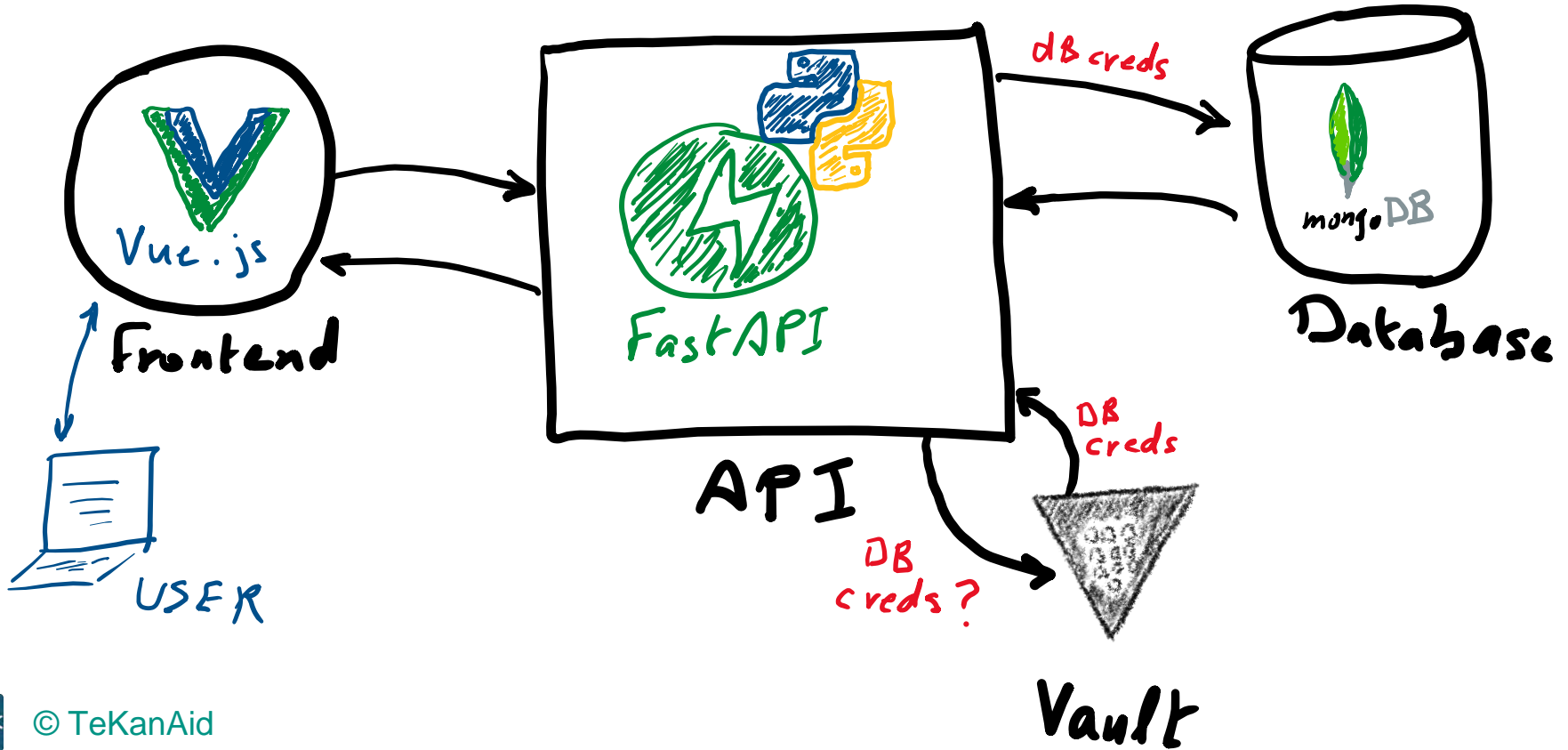
NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/api-64f45f88f7	1	1	1	4m41s
replicaset.apps/frontend-7fd5f756f5	1	1	1	4m16s
replicaset.apps/schoolapp-mongodb-6cdf54d797	1	1	1	32d



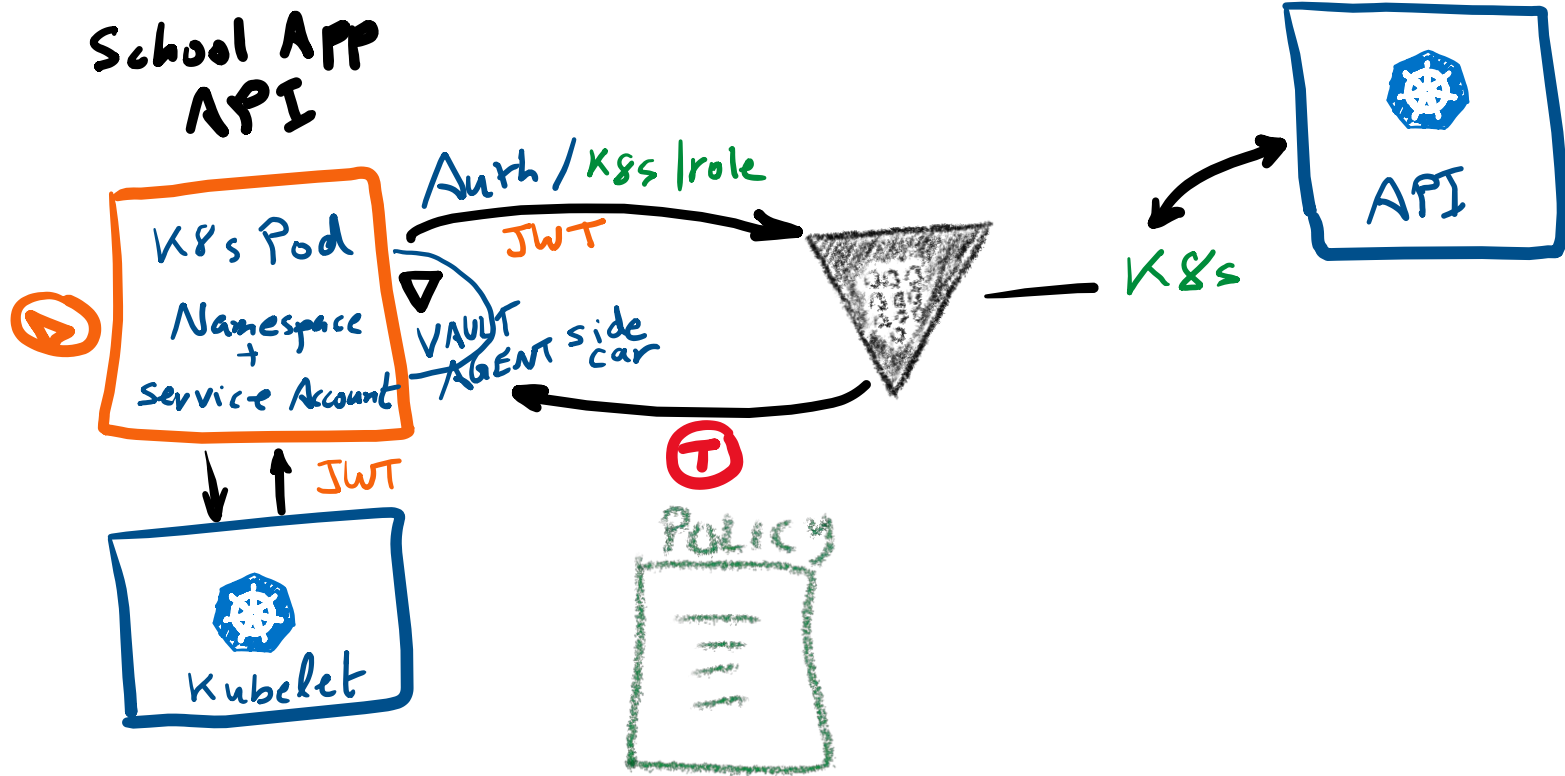
Add Vault to the School App



Add Vault to the School App



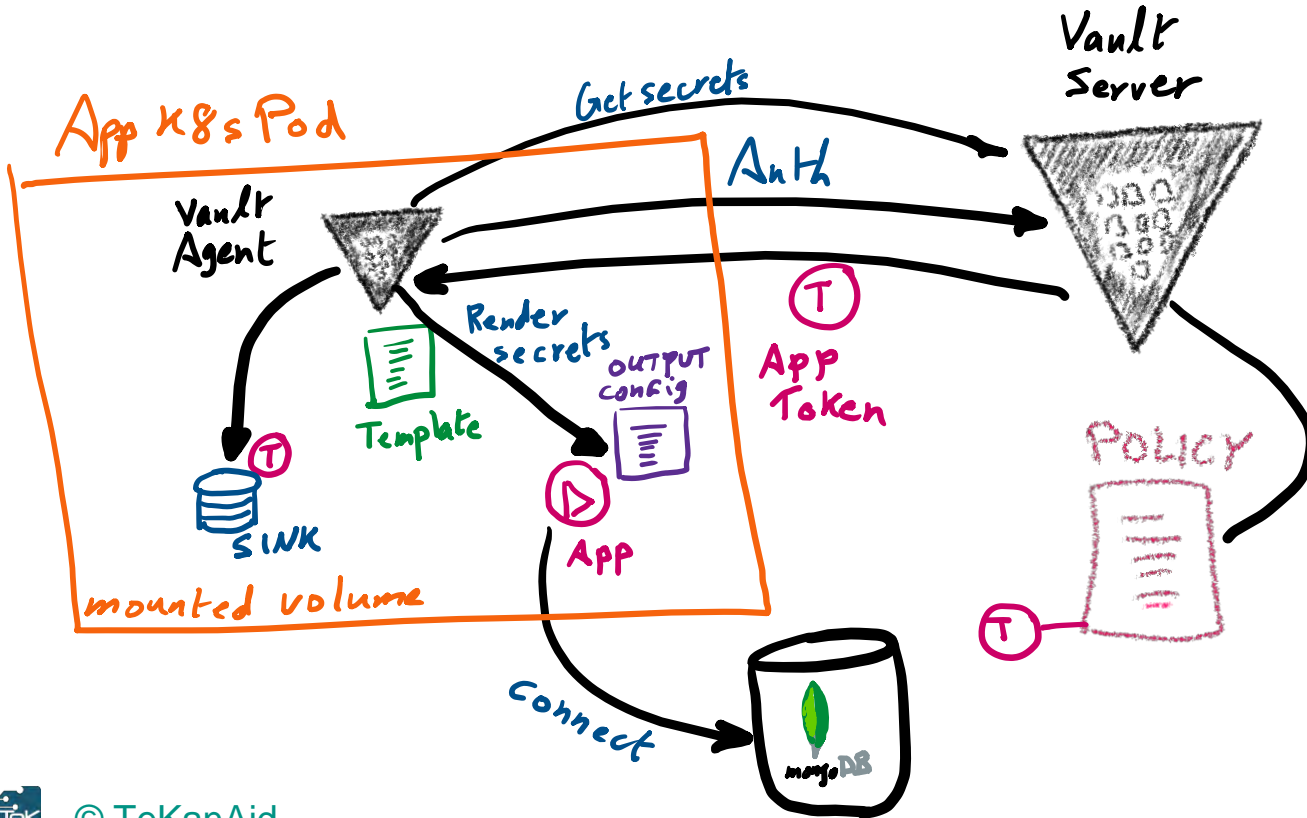
Kubernetes Auth Method



The Vault Agent Sidecar Injector Overview



Vault Agent Workflow in K8s



The Vault Agent Sidecar Injector Overview

- The Vault Agent Injector **alters pod specifications** to include **Vault Agent containers** that render **Vault secrets** to a **shared memory volume** using **Vault Agent Templates**
- **App containers** within the pod can **consume Vault secrets from the shared volume** without being Vault aware
- The injector is a Kubernetes Mutation Webhook Controller
- It works by **intercepting** pod **CREATE** and **UPDATE** events in Kubernetes
- The controller parses the event and looks for the metadata **annotation** `vault.hashicorp.com/agent-inject: true`
- If found, the controller will **alter** the pod specification based on **other annotations** present



Mutation Effects

- **Every container** in the **pod** will be configured to **mount a shared memory volume**. This volume is mounted to `/vault/secrets` by default and will be used by the Vault Agent containers for **sharing secrets** with the **other containers** in the **pod**
- Two types of Vault Agent containers can be injected: **init** and **sidecar**
- The **init** container will **prepopulate** the shared memory volume with the requested secrets **prior** to the other **containers starting**
- The **sidecar** container will continue to authenticate and render secrets to the same location as the pod runs
- Using **annotations**, the **init** and **sidecar** containers **may be disabled**

Good for cronjobs

Good for long-lived containers



Vault Agent Config to Render Secrets

- There are **two methods** of configuring the Vault Agent containers to render secrets:
 - the `vault.hashicorp.com/agent-inject-secret` **annotation**, or
 - a **configuration map** containing Vault Agent configuration files
- Only **one** of these methods may be used **at any time**
- The **configuration map** can provide **more details** for configuration beyond the annotations



School App Annotations Example

```
annotations:  
  vault.hashicorp.com/agent-inject: "true"  
  vault.hashicorp.com/agent-inject-token: "true"  
  vault.hashicorp.com/agent-inject-status: "update"  
  vault.hashicorp.com/role: "schoolapp"  
  vault.hashicorp.com/secret-volume-path: "/app/secrets/"
```

Configures Vault Agent to share the Vault token with other containers in the pod. This is helpful for Vault aware apps that need to communicate directly with Vault but require auto-authentication provided by Vault Agent

Blocks further mutations by adding the value injected to the pod after a successful mutation.

Will render the Vault token in the file called: /app/secrets/token

- Full list of Annotations:

<https://www.vaultproject.io/docs/platform/k8s/injector/annotations>



Vault Agent Templates

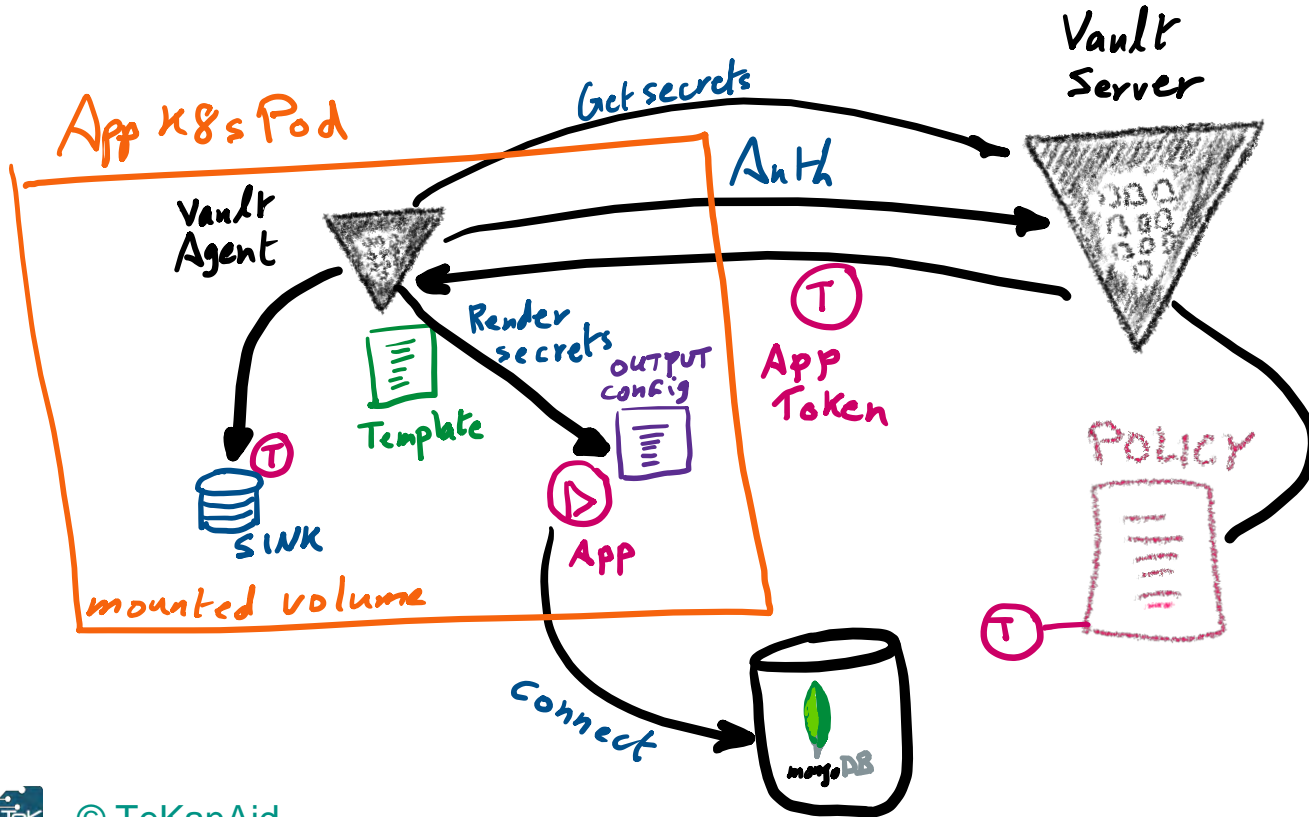


Vault Agent Templates Overview

- **Vault Agent** uses the **Consul Template** project to render secrets
- This is useful for **Vault unaware apps**
- The **app's file system** has the **secrets dropped** in by **Vault**
- The **app doesn't talk** to **Vault** directly
- All it needs to know is what **files to find the secrets** in



Vault Agent Templates Workflow



Vault Agent Templates with Annotations

```
annotations:  
  vault.hashicorp.com/agent-inject: "true"  
  vault.hashicorp.com/agent-inject-token: "true"  
  vault.hashicorp.com/agent-inject-status: "update"  
  vault.hashicorp.com/role: "schoolapp"  
  vault.hashicorp.com/secret-volume-path: "/app/secrets/"  
  vault.hashicorp.com/agent-inject-secret-schoolapp-mongodb-username:  
"internal/data/schoolapp/mongodb"  
  vault.hashicorp.com/agent-inject-secret-schoolapp-mongodb-password:  
"internal/data/schoolapp/mongodb"  
  vault.hashicorp.com/agent-inject-template-schoolapp-mongodb-username: |  
  {{- with secret "internal/data/schoolapp/mongodb" -}}  
  {{ .Data.data.schoolapp_DB_USERNAME }}  
  {{- end -}}  
  vault.hashicorp.com/agent-inject-template-schoolapp-mongodb-password: |  
  {{- with secret "internal/data/schoolapp/mongodb" -}}  
  {{ .Data.data.schoolapp_DB_PASSWORD }}  
  {{- end -}}
```

Expected Output in Container:
cat /app/secrets/schoolapp-
mongodb-username
schoolapp
cat /app/secrets/schoolapp-
mongodb-password
mongoRootPass



Demo

