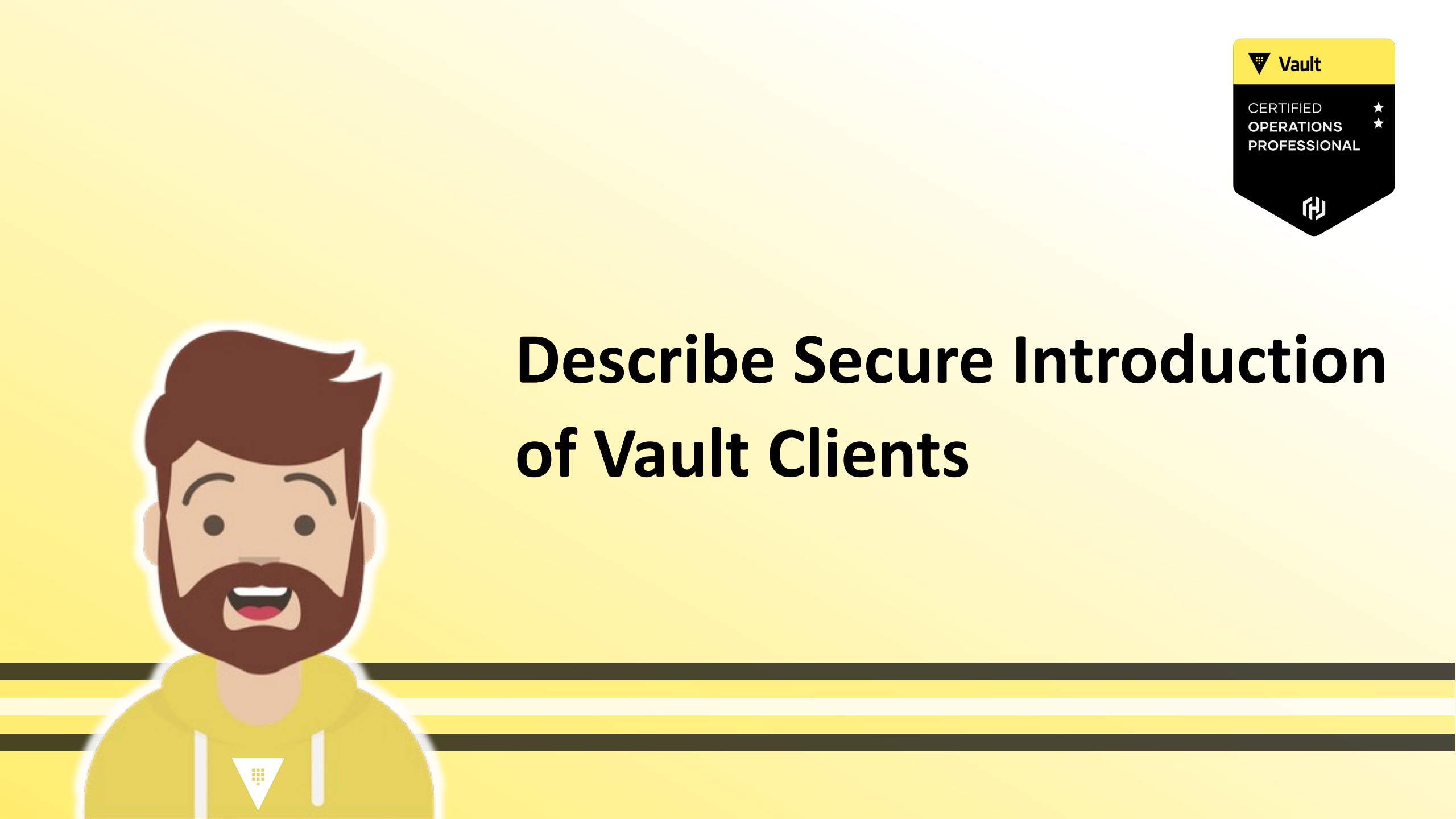


Describe Secure Introduction of Vault Clients

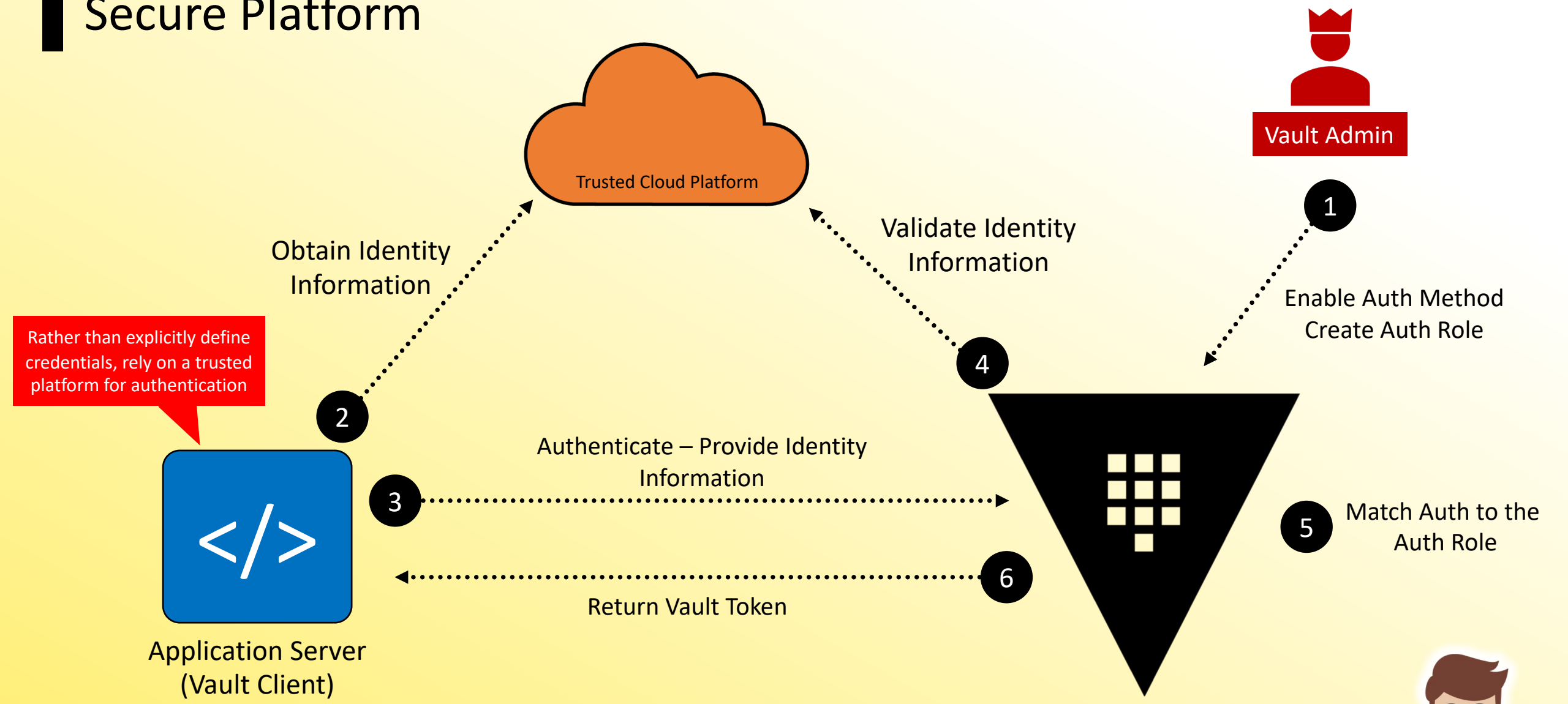
What is Secret Zero?

- Secret zero is essentially the "first secret" needed to obtain other secrets
  - Example: 1Password or LastPass
- In Vault, this is either the authentication credentials or a Vault token
- Once we have secret zero, we can potentially obtain other credentials. Unfortunately, it also allows for an unauthorized user to elevate privileges in the organization
- The goal is to introduce secret zero in the most secure fashion but only when it's needed for the application to use it

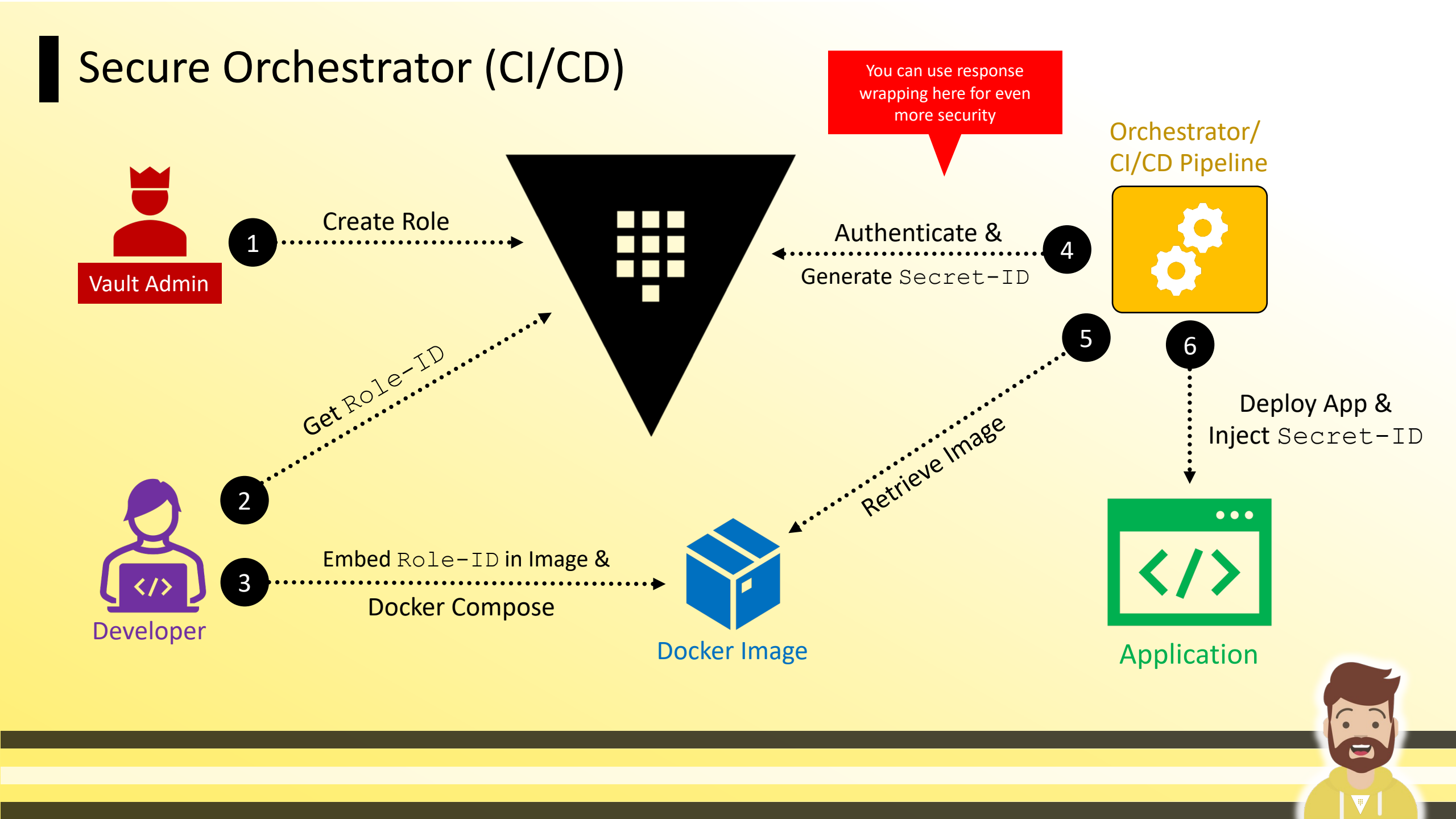
Secure Introduction Goals

1. Use unique credentials for each application instance provisioned
2. Limit your exposure if a credential is compromised
3. Stop hardcoding credentials within the application codebase
4. Reduce the TTL of the credentials used by applications and reduce long-lived creds
5. Distribute credentials securely and only at runtime
6. Use a trusted platform to verify the identities of clients
7. Employ a trusted orchestrator that is already authenticated to Vault to inject secrets

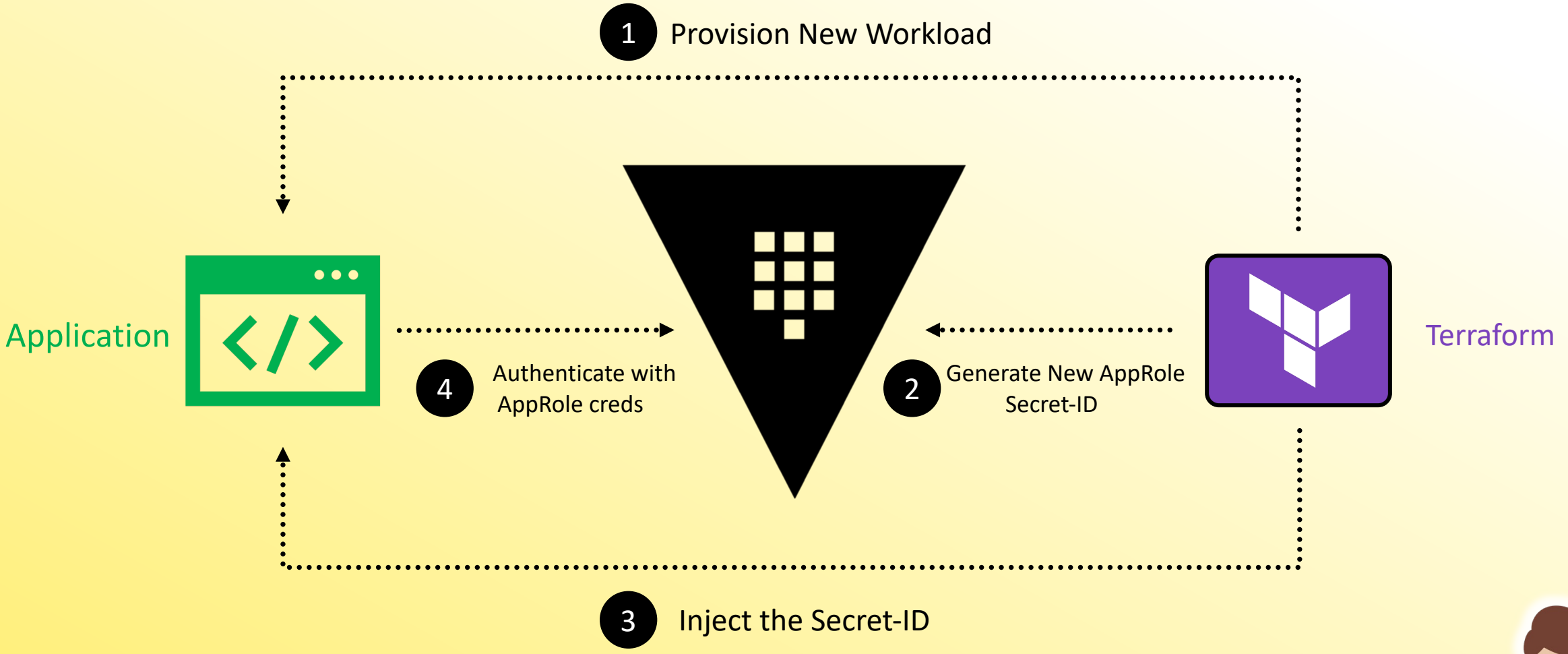
Secure Platform



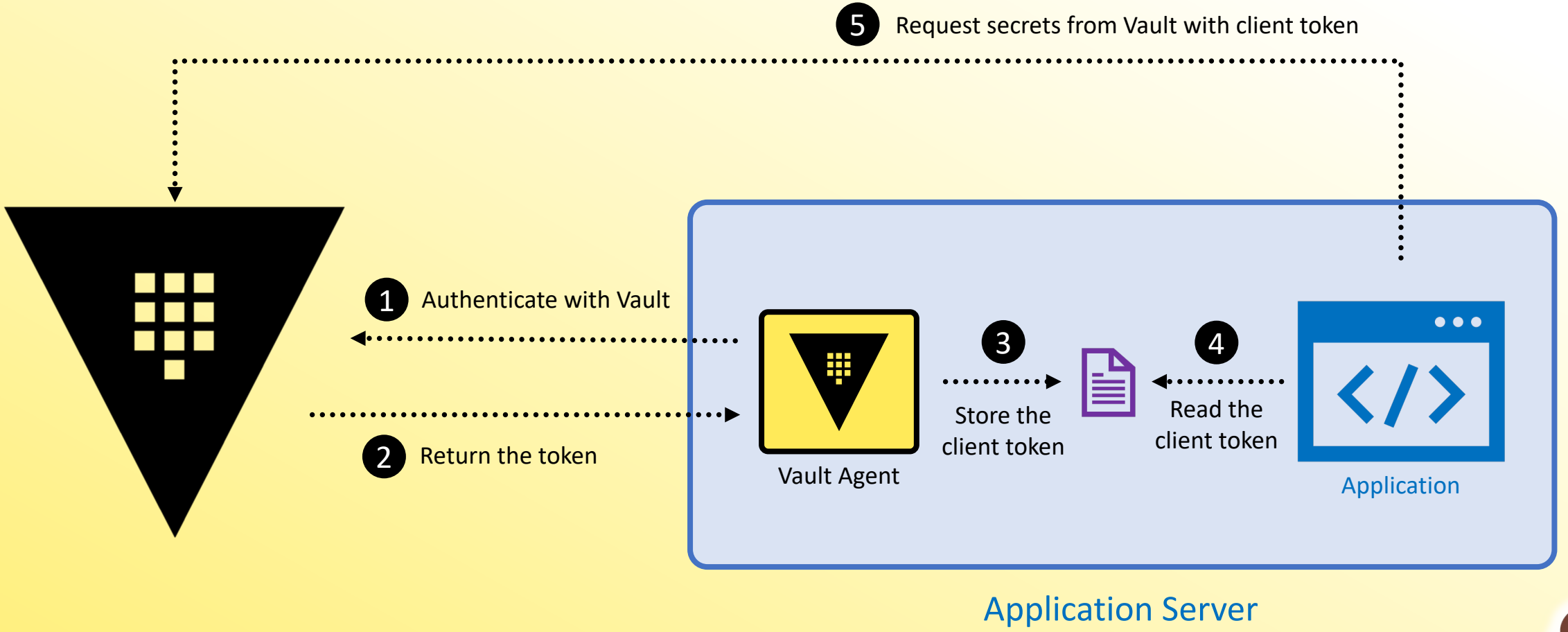
Secure Orchestrator (CI/CD)



Secure Orchestrator (Terraform)



Vault Agent – Auto Auth

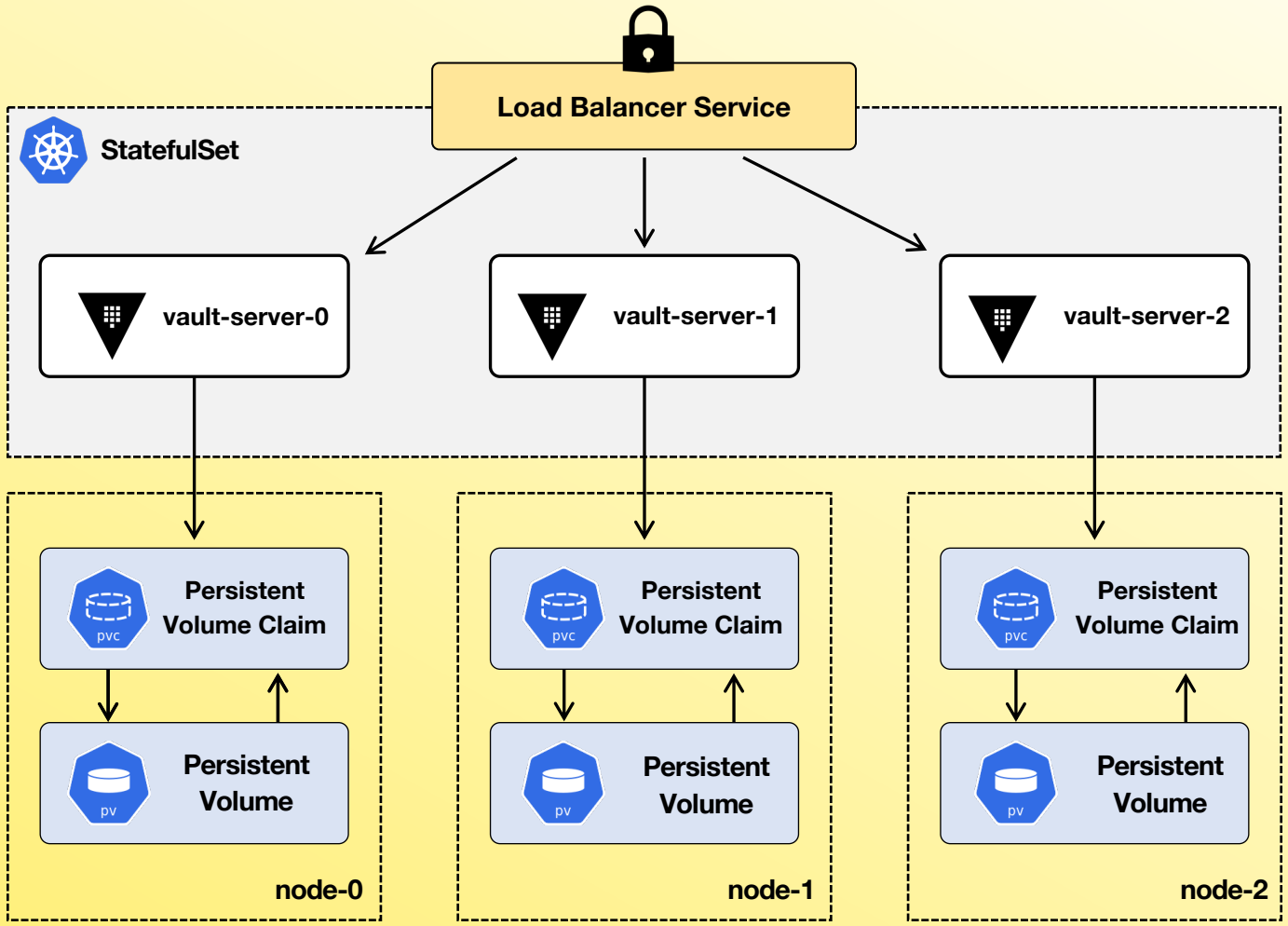


Describe the Security Implications of Running Vault in Kubernetes

Running Vault on Kubernetes

- As a consultant, I'm seeing more and more customers looking to deploy Vault on Kubernetes, including EKS, AKS, GKE, and OpenShift
- The easiest way to deploy Vault on Kubernetes is to use the official Helm chart
- The Vault security model assumes that Vault will be run on VMs/physical hardware and not necessarily containers, so HashiCorp provides additional recommendations specifically for containerization

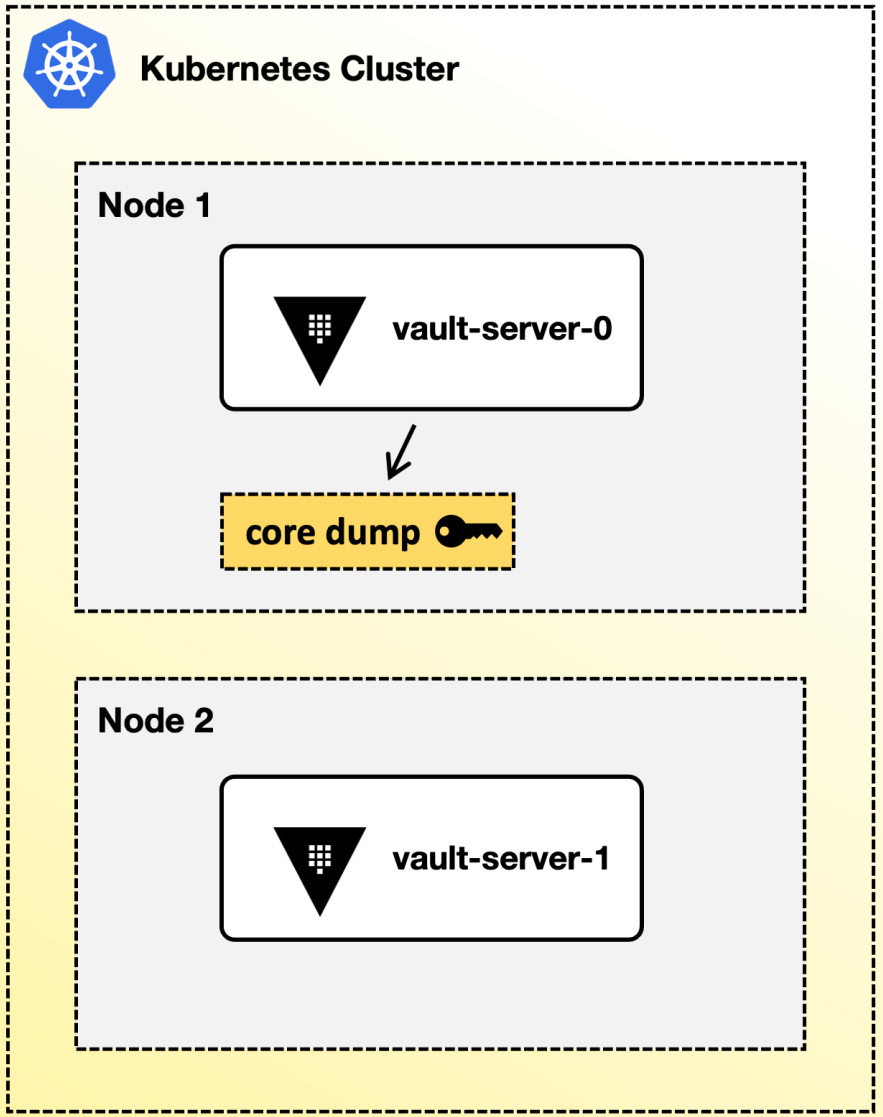
TLS – End-to-End-Encryption



- Don't offload TLS at the load balancer
- Ensures end-to-end encryption from the client to the Vault node
- Use TLS certificates signed by a trusted Certificate Authority (CA)
- Require TLS 1.2+

Disable Core Dumps

- Most commonly, Vault pods are scheduled to run on a separate cluster to reduce/eliminate shared resources
- Core dump files may include Vault's encryption keys
- Ensure `RLIMIT_CORE` is set to 0 or use the `ulimit` command with the core flag (`ulimit -c 0`) inside the container to ensure your container processes can't core dump.



Ensure mlock is Enabled

- Memory lock ensures memory from a process on a Linux system isn't swapped to disk. Additional configurations are needed for containerized deployments

- The process that starts the container that runs the mlock call must have IPC_LOCK capabilities

Terminal

```
securityContext:
    runAsNonRoot: true
    runAsUser: 1000
    capabilities:
      add: ["IPC_LOCK"]
```

Container Supervisor

- If your container starts as root, the processes that might escape that container will also have root on the node

- Mitigations can be used to prevent starting your container as root
  - SecurityContext → runAsNonRoot
  - PodSecurityContext → runAsNonRoot

Terminal

```
apiVersion: v1
kind: Pod
metadata:
  name: hello-world
spec:
  containers:
  # specification of the pod's containers
  # ...
  securityContext:
    readOnlyRootFilesystem: true
    runAsNonRoot: true
```

Don't Run Vault as Root

- Vault is designed to run as an unprivileged user – regardless of the platform
- Elevated privileges can potentially expose the Vault process memory and allow access to Vault encryption keys