Installing Vault

Vault is platform agnostic....meaning it can be run on many different underlying platforms, such as:

Kubernetes

Cloud-based Machines (AWS Instances, Azure Virtual Machines)

VMware Virtual Machines

Physical Servers

A Laptop

Installing Vault

Vault is also available for many operating systems...

- macOS
- Windows
- Linux
- FreeBSD
- NetBSD
- OpenBSD
- Solaris

Installing Vault

Order of Operations

1. Install Vault
2. Create Configuration File
3. Initialize Vault
4. Unseal Vault

Installing Vault

- So where do I download Vault?
  - https://developer.hashicorp.com/vault
  - releases.hashicorp.com/vault
- Ideally, you should install Vault using your preferred package manager (apt, yum, homebrew, chocolately (community supported))

Terminal

```
$ sudo yum install -y yum-utils shadow-utils
$ sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
$ sudo yum -y install vault
```

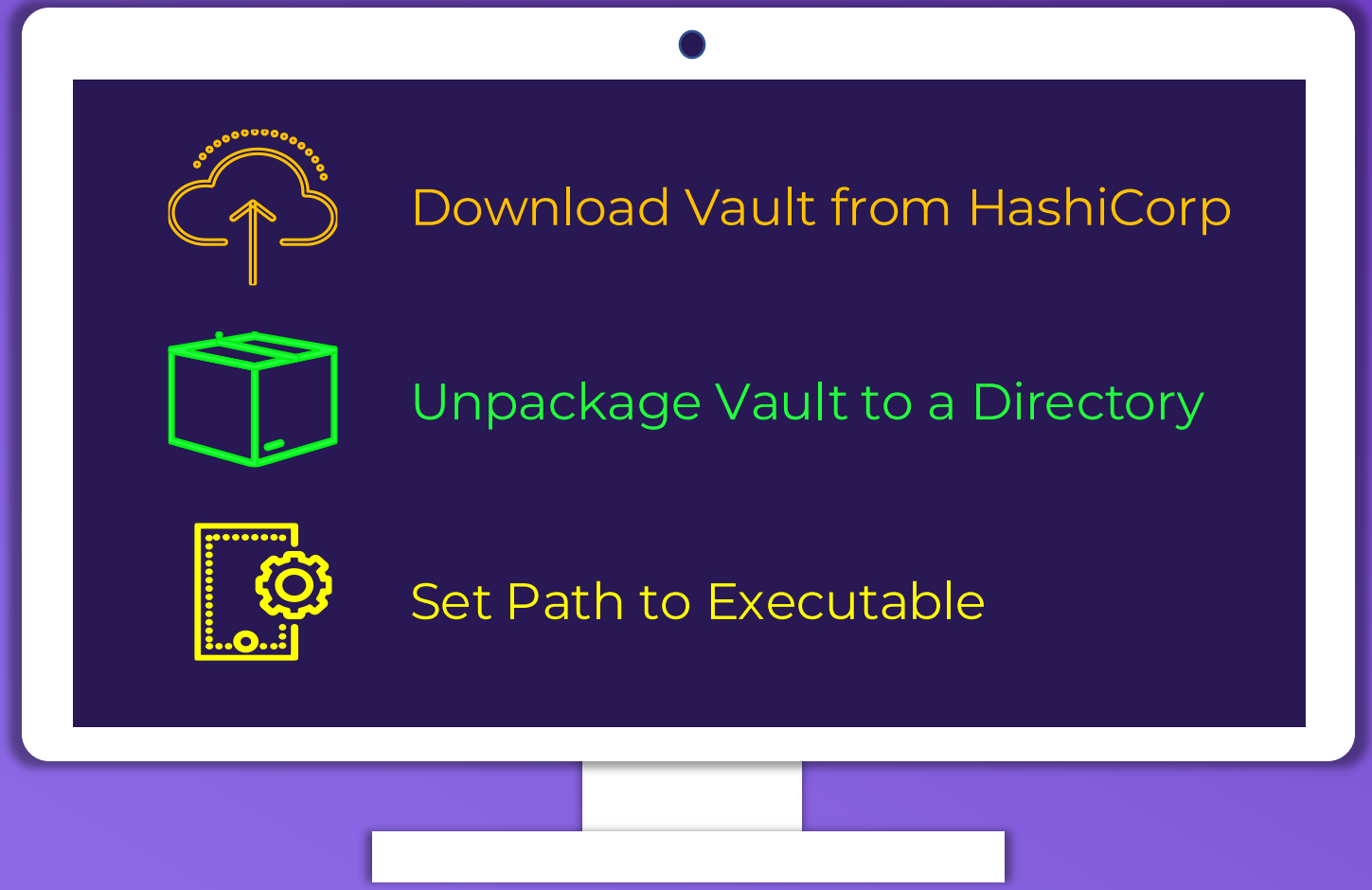
- Use the Vault Helm Chart to install/configure Vault on Kubernetes

Terminal

```
$ helm install vault hashicorp/vault
```

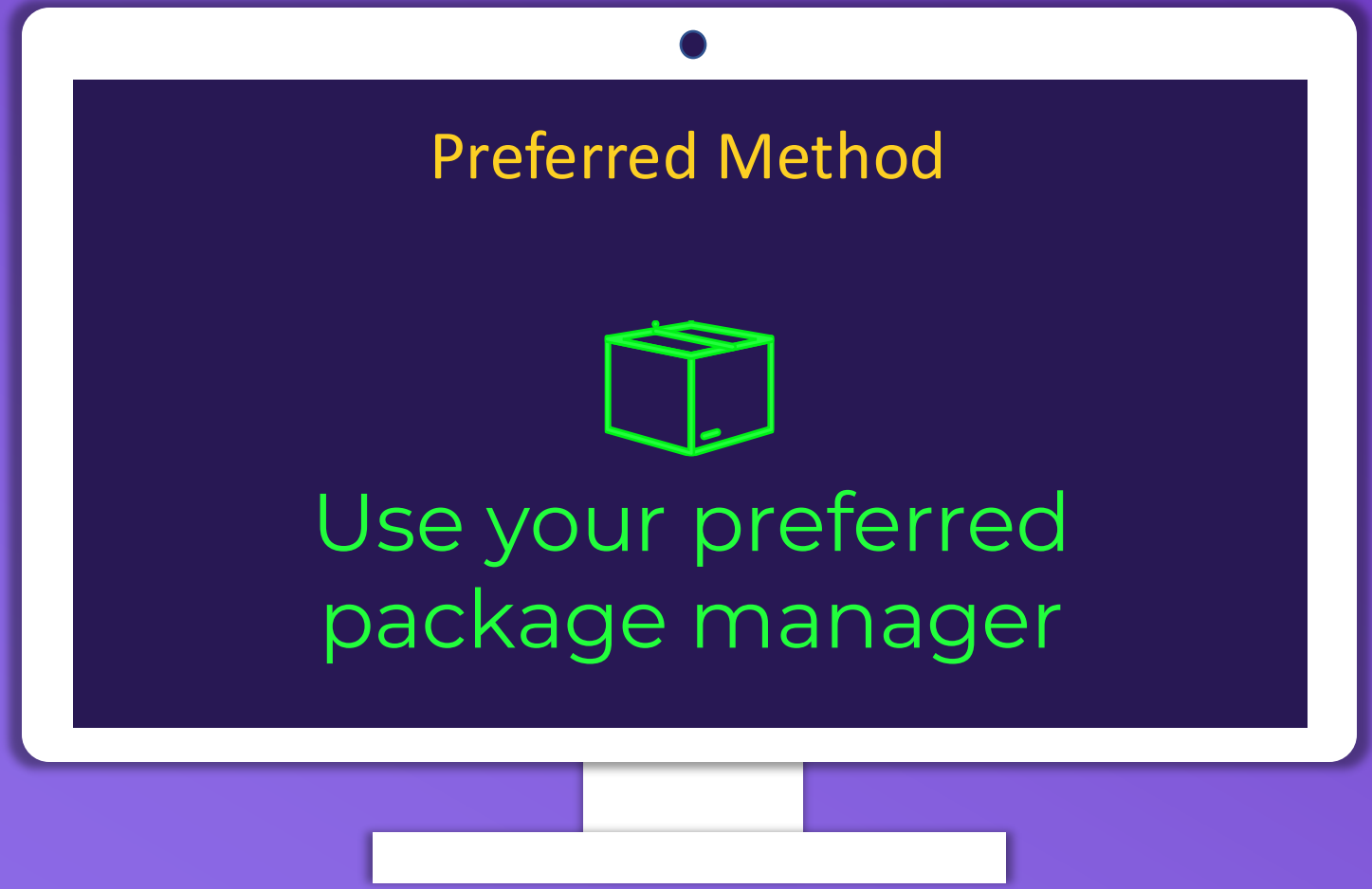
Installing Vault

Manual Installation



Installing Vault

Package Manager



Running Vault Dev Server

- Quickly run Vault without configuration
- Automatically initialized and unsealed
- Enables the UI – available at localhost
- Provides an Unseal Key
- Automatically logs in as root

- Non-Persistent – Runs in memory
- Insecure – doesn't use TLS
- Sets the listener to 127.0.0.1:8200
- Mounts a K/V v2 Secret Engine
- Provides a root token

NEVER USE DEV SERVER MODE IN PRODUCTION!

Where Would I Use Dev Server?

Dev Server Mode

- Proof of Concepts
- New Development Integrations
- Testing New Features of Vault
- Experimenting with Features

Running Vault Dev Server

TERMINAL

```
$ vault server -dev

==> Vault server configuration:

Administrative Namespace:
             Api Address: http://127.0.0.1:8200
                     Cgo: disabled
         Cluster Address: https://127.0.0.1:8201
              Go Version: go1.23.8
              Listener 1: tcp (addr: "127.0.0.1:8200", cluster address: "127.0.0.1:8201",
disable_request_limiter: "false", max_request_duration: "1m30s", max_request_size:
"33554432", tls: "disabled")
               Log Level:
                   Mlock: supported: false, enabled: false
           Recovery Mode: false
                 Storage: inmem
                 Version: Vault v1.19.3, built 2025-04-29T10:34:52Z
             Version Sha: a2de3bb7bcf4a073cbb8724863a5a88d3c2f83da

==> Vault server started! Log data will stream:
```

Running Vault Server in Production

- Deploy one or more persistent nodes via configuration file
- Use a storage backend that meets the requirements
- Multiple Vault nodes will be configured as a cluster
- Deploy close to your applications
- Most likely, you'll automate the provisioning of Vault

Running Vault Server in Production

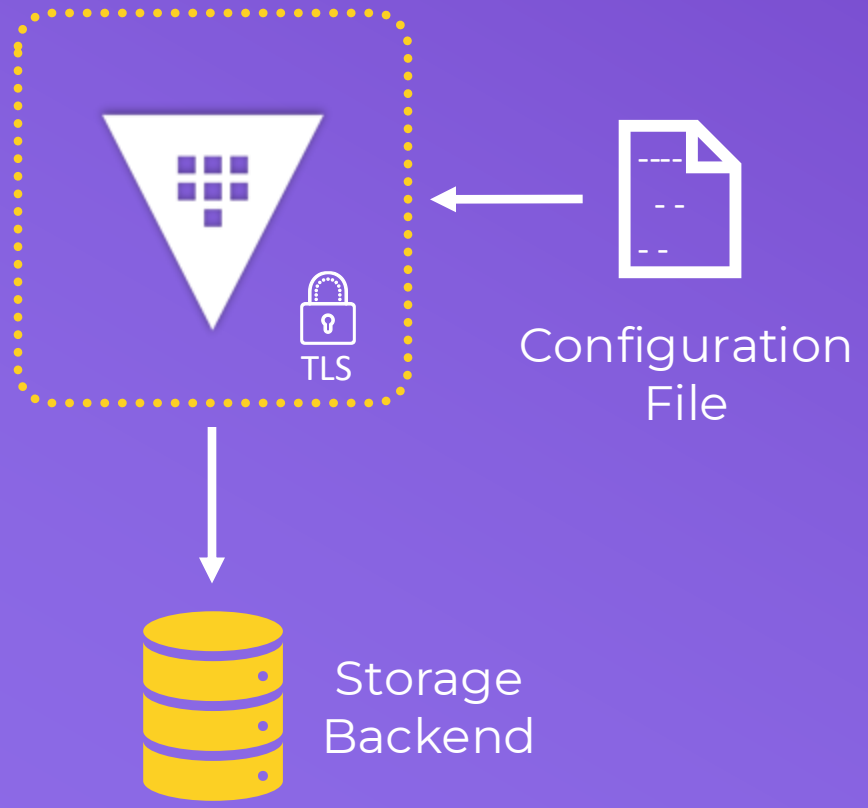
- To start Vault, run the vault server –config=<file> command
- In a production environment, you'll have a service manager executing and managing the Vault service (systemctl, Windows Service Manager, etc)
- For Linux, you also need a systemd file to manage the service for Vault

Running Vault Server in Production

- Systemd for a Vault service:
  - https://github.com/btkrausen/hashicorp/blob/master/vault/config_files/vault.service
- Systemd file for a Consul Server:
  - https://github.com/btkrausen/hashicorp/blob/master/consul/consul.service
- Systemd for a Consul client (that would run on the Vault node):
  - https://github.com/btkrausen/hashicorp/blob/master/vault/config_files/consul-client.json

Running Vault Server in Production

Single Node

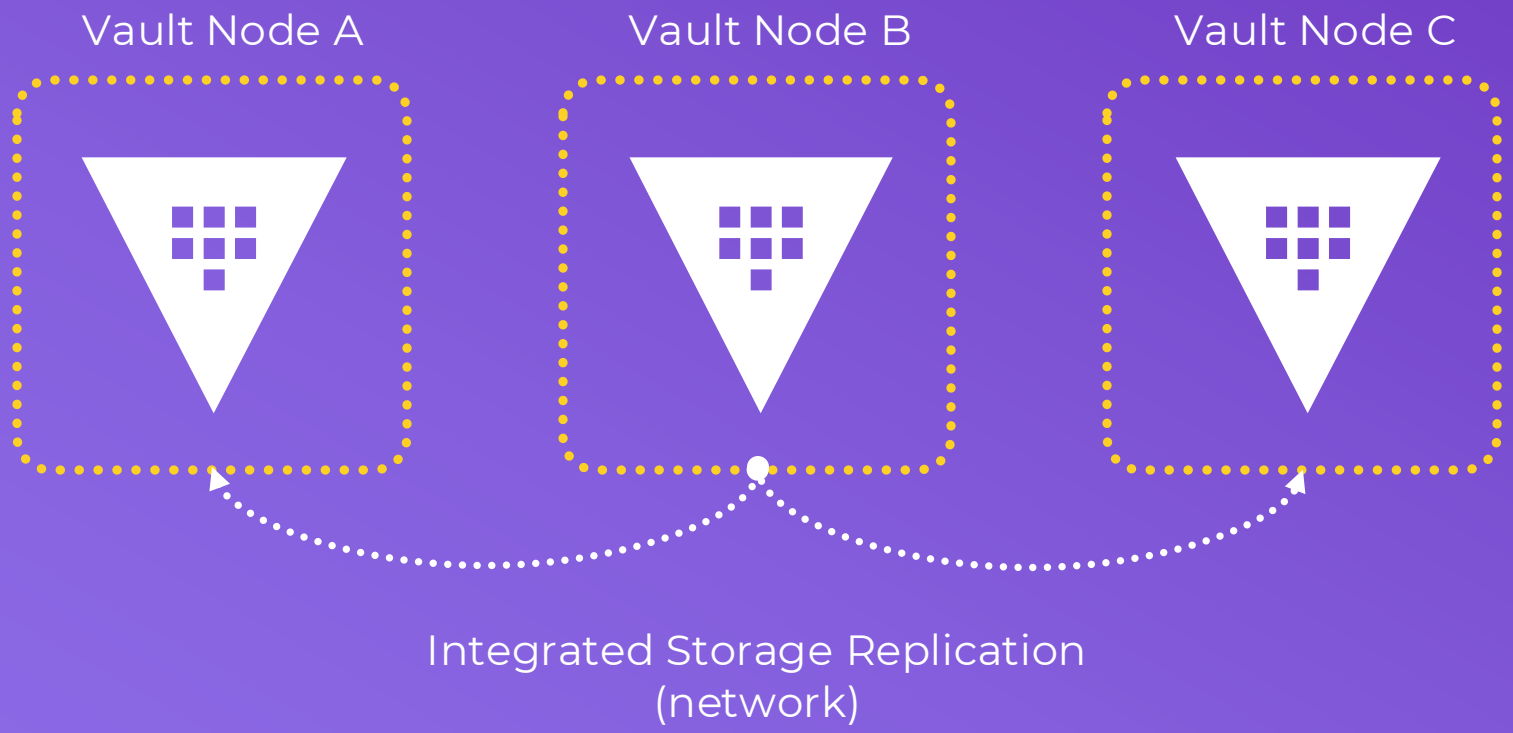


Not a Recommended Architecture

- No Redundancy
- No Scalability

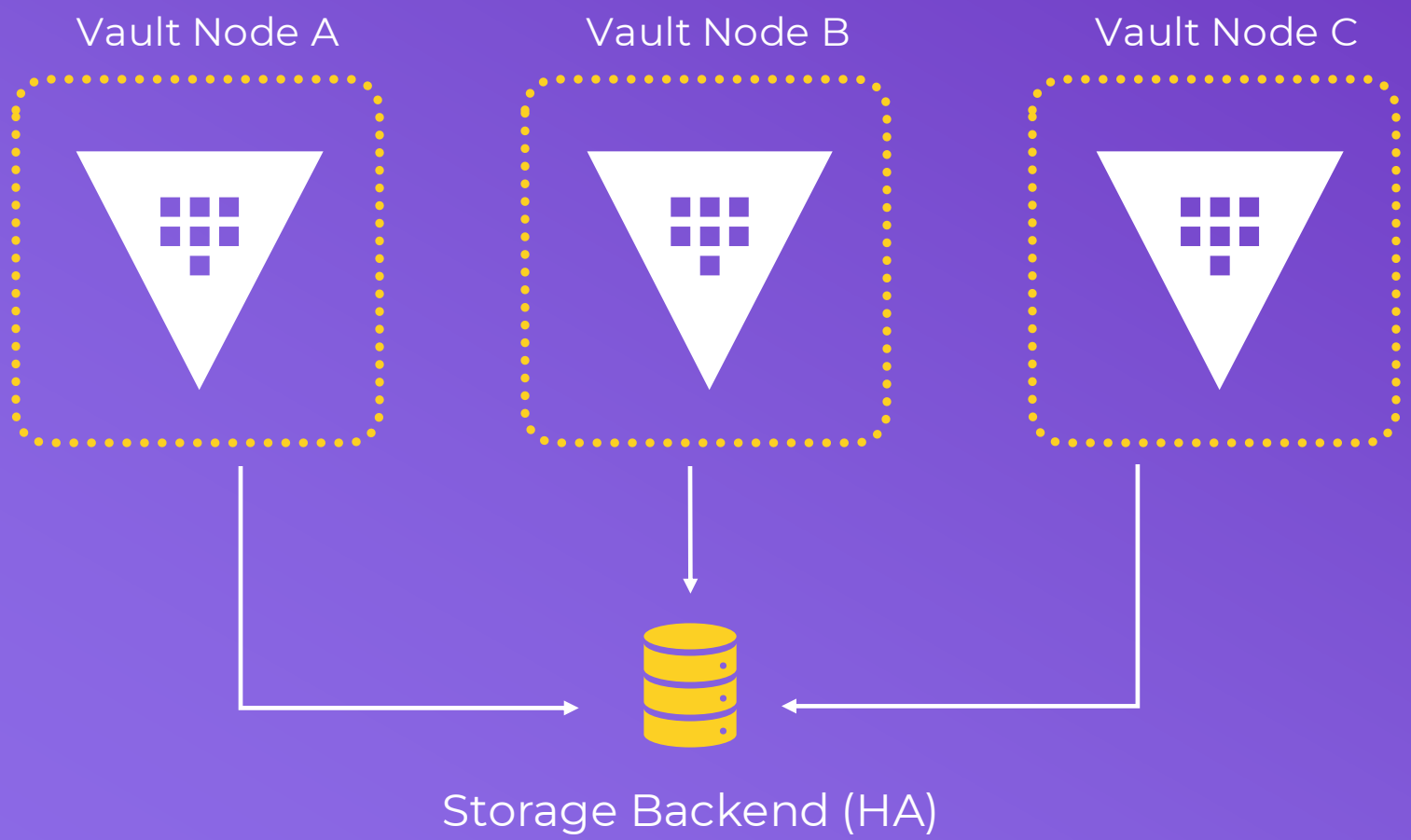
Running Vault Server in Production

Multi-Node Vault Cluster (with Integrated Storage)



Running Vault Server in Production

Multi-Node Vault Cluster (with External Storage)



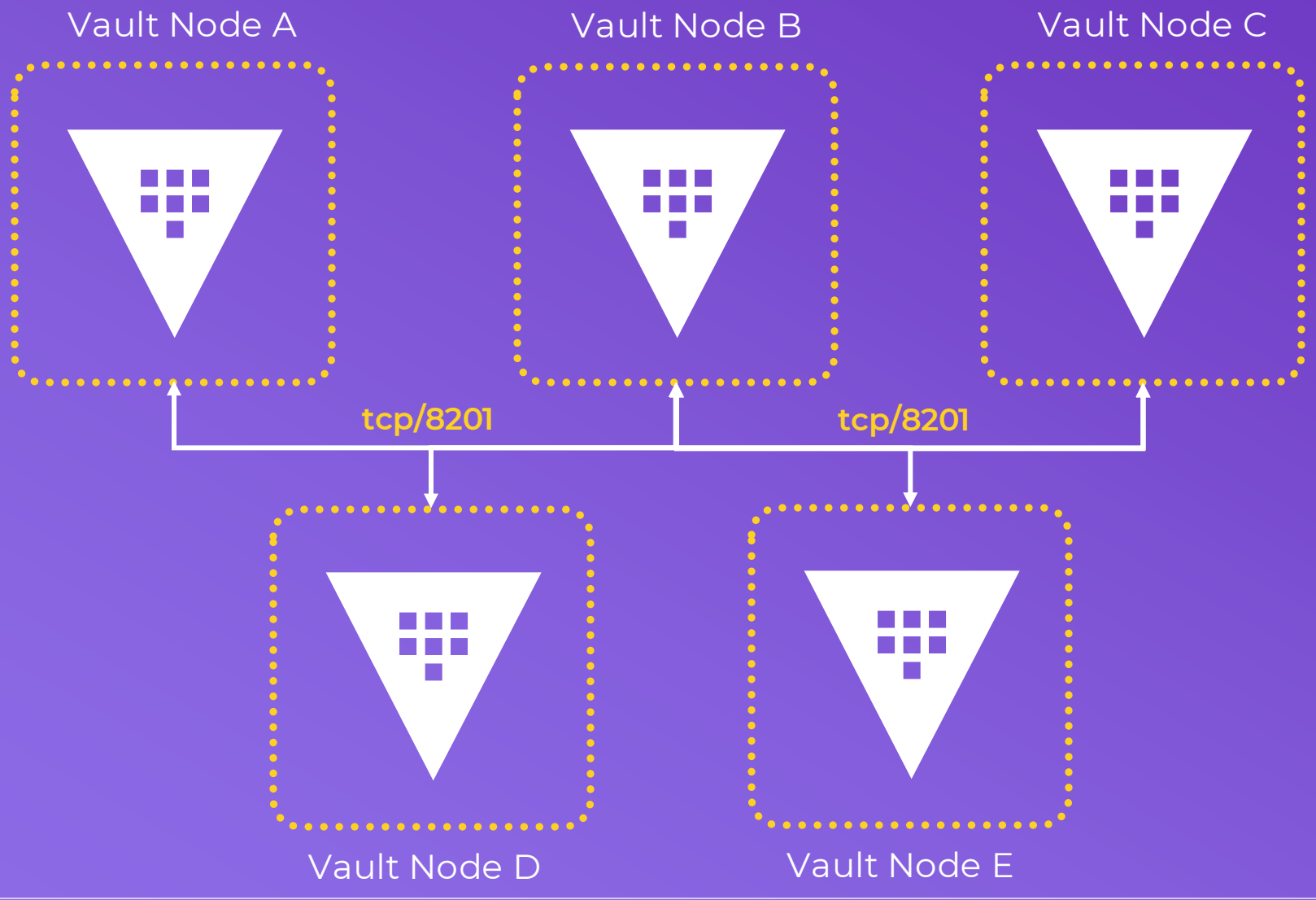
Deploying the Integrated Storage Backend

- Vault Internal Storage Option
- Leverages Raft Consensus Protocol
- All Vault nodes have a copy of the data
- Eliminates Network Hop to Consul
- Supports High Availability
- Only need to troubleshoot Vault
- Built-in Snapshots For Data Retention
- HashiCorp Supported

Deploying the Integrated Storage Backend

- Integrated Storage (aka Raft) allows Vault nodes to provide its own replicated storage across the Vault nodes within a cluster
- Define a local path to store replicated data
- All data is replicated among all nodes in the cluster
- Eliminates the need to run a Consul cluster and manage it

Deploying the Integrated Storage Backend



Deploying the Integrated Storage Backend

Example Vault Server Configuration File

vault.hcl

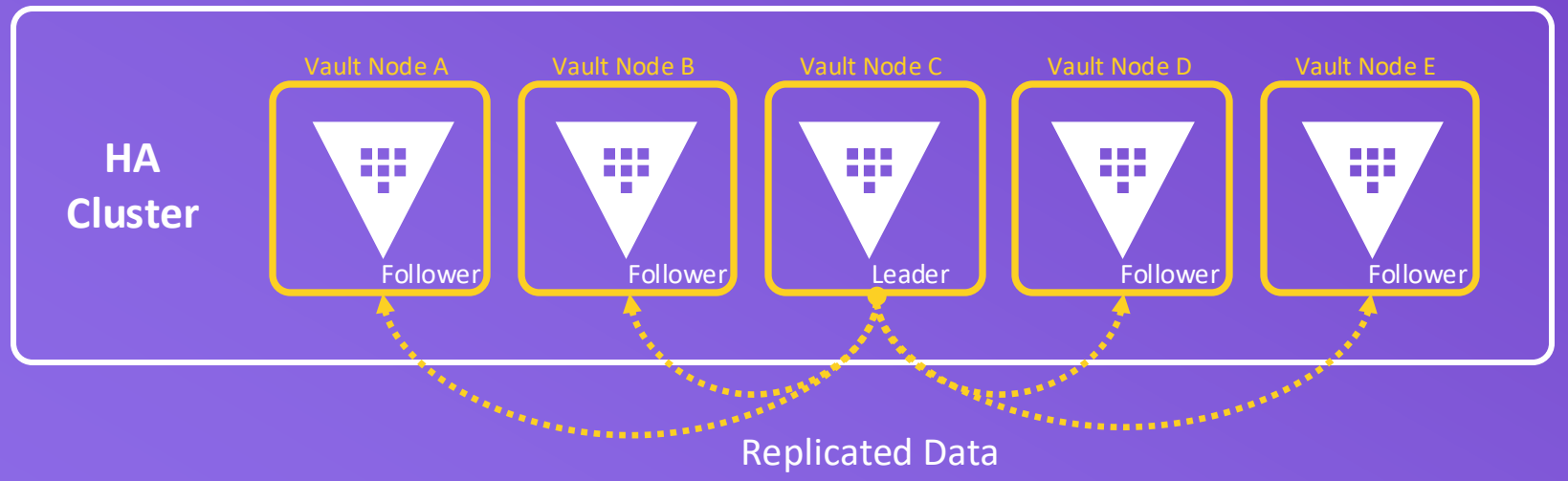
```
storage "raft" {
  path    = "/opt/vault/data"
  node_id = "node-a-us-east-1.example.com"
  retry_join {
    auto_join = "provider=aws region=us-east-1 tag_key=vault tag_value=us-east-1"
  }
}

listener "tcp" {
 address = "0.0.0.0:8200"
 cluster_address = "0.0.0.0:8201"
 tls_disable = 0
 tls_cert_file = "/etc/vault.d/client.pem"
...
```

Deploying the Integrated Storage Backend

Manually join standby nodes to the cluster using the CLI:

TERMINAL

```
$ vault operator raft join https://active_node.example.com:8200
```



Deploying the Integrated Storage Backend

List the cluster members

TERMINAL

```
$ vault operator raft list-peers

Node       Address             State       Voter
----       -------             -----       -----
vault_1    10.0.101.22:8201    leader      true
vault_2    10.0.101.23:8201    follower    true
vault_3    10.0.101.24:8201    follower    true
vault_4    10.0.101.25:8201    follower    true
vault_5    10.0.101.26:8201    follower    true
```