Installing Vault

- Vault is platform agnostic….meaning it can be run on many different underlying platforms
  - 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

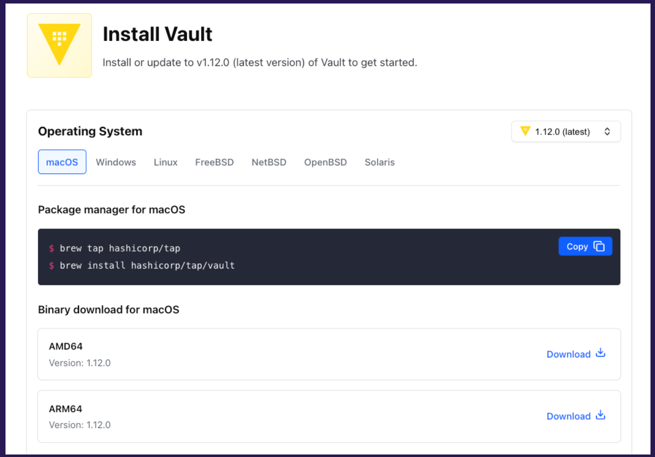
2 Create Configuration File

3 Initialize Vault

4 Unseal Vault

Installing Vault

- So where do I download Vault?
  - vaultproject.io
  - releases.hashicorp.com/vault



Install Vault

Install or update to v1.12.0 (latest version) of Vault to get started.

Operating System — 1.12.0 (latest)

macOS | Windows | Linux | FreeBSD | NetBSD | OpenBSD | Solaris

Package manager for macOS

```
$ brew tap hashicorp/tap
$ brew install hashicorp/tap/vault
```

Binary download for macOS

AMD64 Version: 1.12.0	Download
ARM64 Version: 1.12.0	Download

- You can also download/install Vault using your preferred package manager as well (apt, yum, even homebrew (community supported))

Terminal

```
$ sudo apt update && sudo apt install gpg
$ wget -O- https://apt.releases.hashicorp.com/gpg | gpg --dearmor | sudo tee /usr/share/keyrings/hashicorp-archive-keyring.gpg >/dev/null
$ gpg --no-default-keyring --keyring /usr/share/keyrings/hashicorp-archive-keyring.gpg –fingerprint
$ echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb_release -cs) main" |
    sudo tee /etc/apt/sources.list.d/hashicorp.list
$ sudo apt update && sudo apt install vault
```

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

Terminal

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

Installing Vault

Download Vault from HashiCorp

Unpackage Vault to a Directory

Set Path to Executable

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

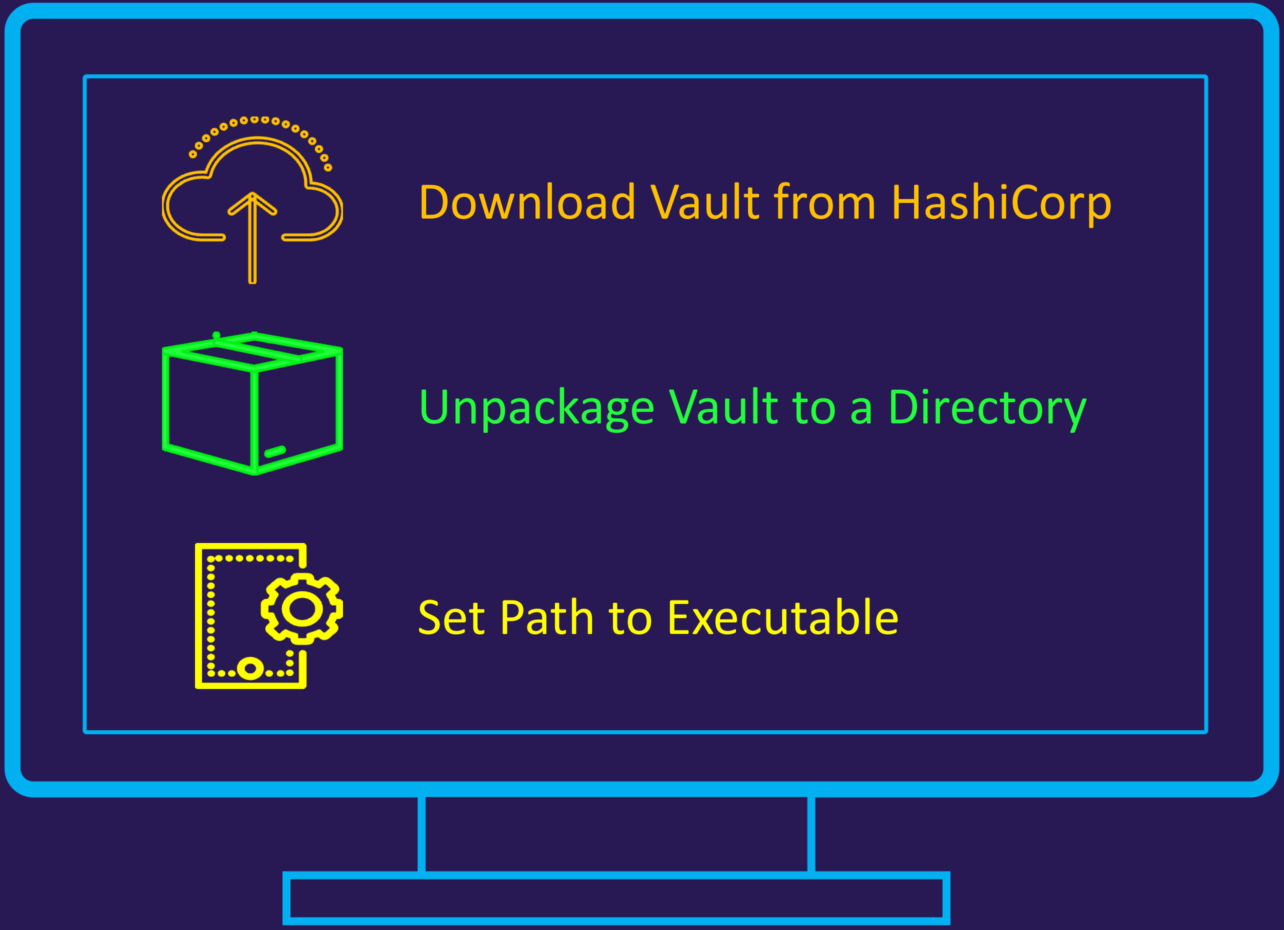
NEVER USE DEV SERVER MODE IN PRODUCTION!

Where Would I Use Dev Server?

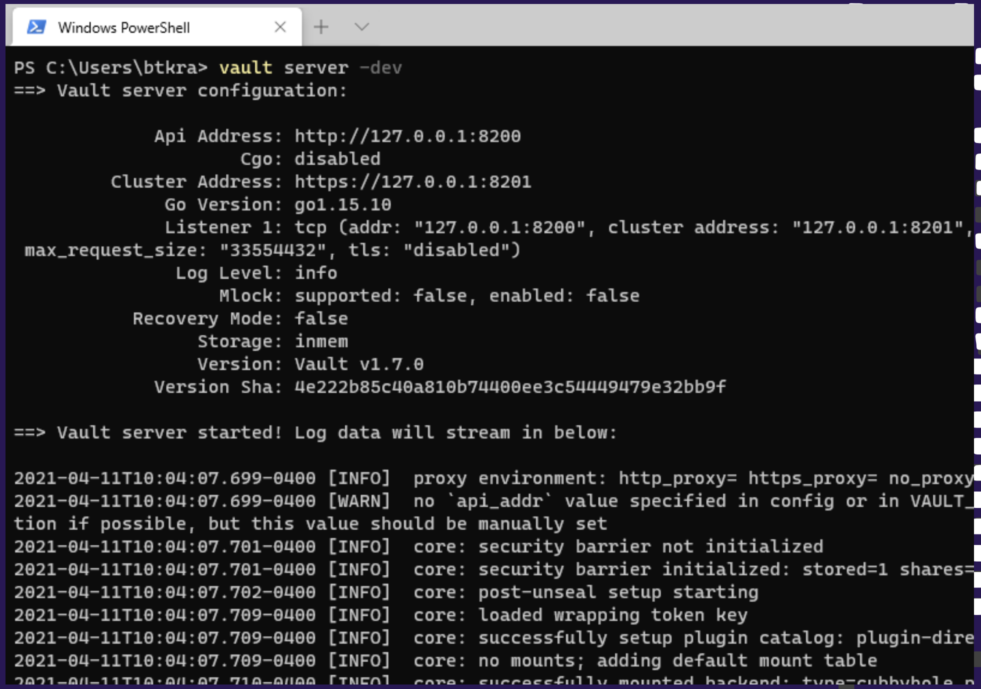
Dev Server Mode

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

Installing Vault



$ vault server -dev

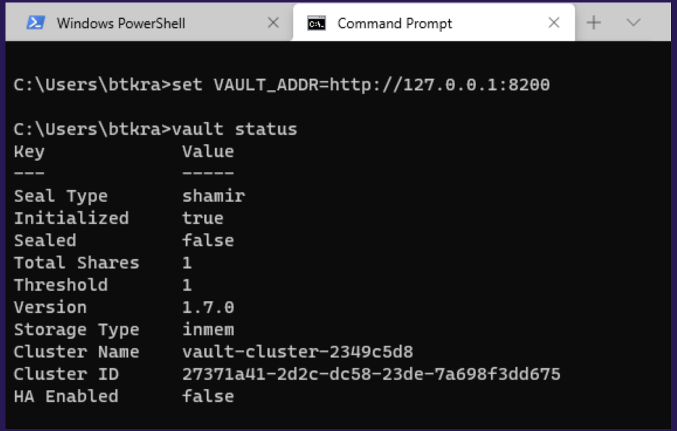


```
PS C:\Users\btkra> vault server -dev
==> Vault server configuration:

             Api Address: http://127.0.0.1:8200
                     Cgo: disabled
         Cluster Address: https://127.0.0.1:8201
              Go Version: go1.15.10
              Listener 1: tcp (addr: "127.0.0.1:8200", cluster address: "127.0.0.1:8201",
 max_request_size: "33554432", tls: "disabled")
               Log Level: info
                   Mlock: supported: false, enabled: false
           Recovery Mode: false
                 Storage: inmem
                 Version: Vault v1.7.0
             Version Sha: 4e222b85c40a810b74400ee3c54449479e32bb9f

==> Vault server started! Log data will stream in below:

2021-04-11T10:04:07.699-0400 [INFO]  proxy environment: http_proxy= https_proxy= no_proxy
2021-04-11T10:04:07.699-0400 [WARN]  no `api_addr` value specified in config or in VAULT_
tion if possible, but this value should be manually set
2021-04-11T10:04:07.701-0400 [INFO]  core: security barrier not initialized
2021-04-11T10:04:07.701-0400 [INFO]  core: security barrier initialized: stored=1 shares=
2021-04-11T10:04:07.702-0400 [INFO]  core: post-unseal setup starting
2021-04-11T10:04:07.709-0400 [INFO]  core: loaded wrapping token key
2021-04-11T10:04:07.709-0400 [INFO]  core: successfully setup plugin catalog: plugin-dire
2021-04-11T10:04:07.709-0400 [INFO]  core: no mounts; adding default mount table
2021-04-11T10:04:07.710-0400 [INFO]  core: successfully mounted backend: type=cubbyhole p
```



```
C:\Users\btkra>set VAULT_ADDR=http://127.0.0.1:8200

C:\Users\btkra>vault status
Key             Value
---             -----
Seal Type       shamir
Initialized     true
Sealed          false
Total Shares    1
Threshold       1
Version         1.7.0
Storage Type    inmem
Cluster Name    vault-cluster-2349c5d8
Cluster ID      27371a41-2d2c-dc58-23de-7a698f3dd675
HA Enabled      false
```

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

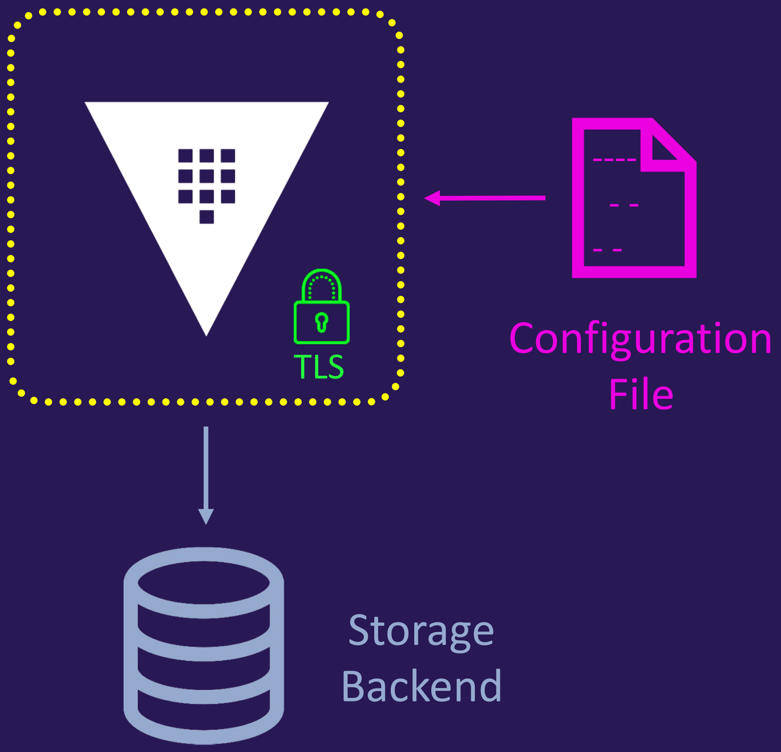
- To start Vault, run the `vault server -config=<file>` command
- In a production environment, you'll have a service manager executing and managing the Vault service (systemctl, Windows Service Manager, etc.)
- For Linux, you also need a systemd file to manage the service for Vault (and Consul if you're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

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

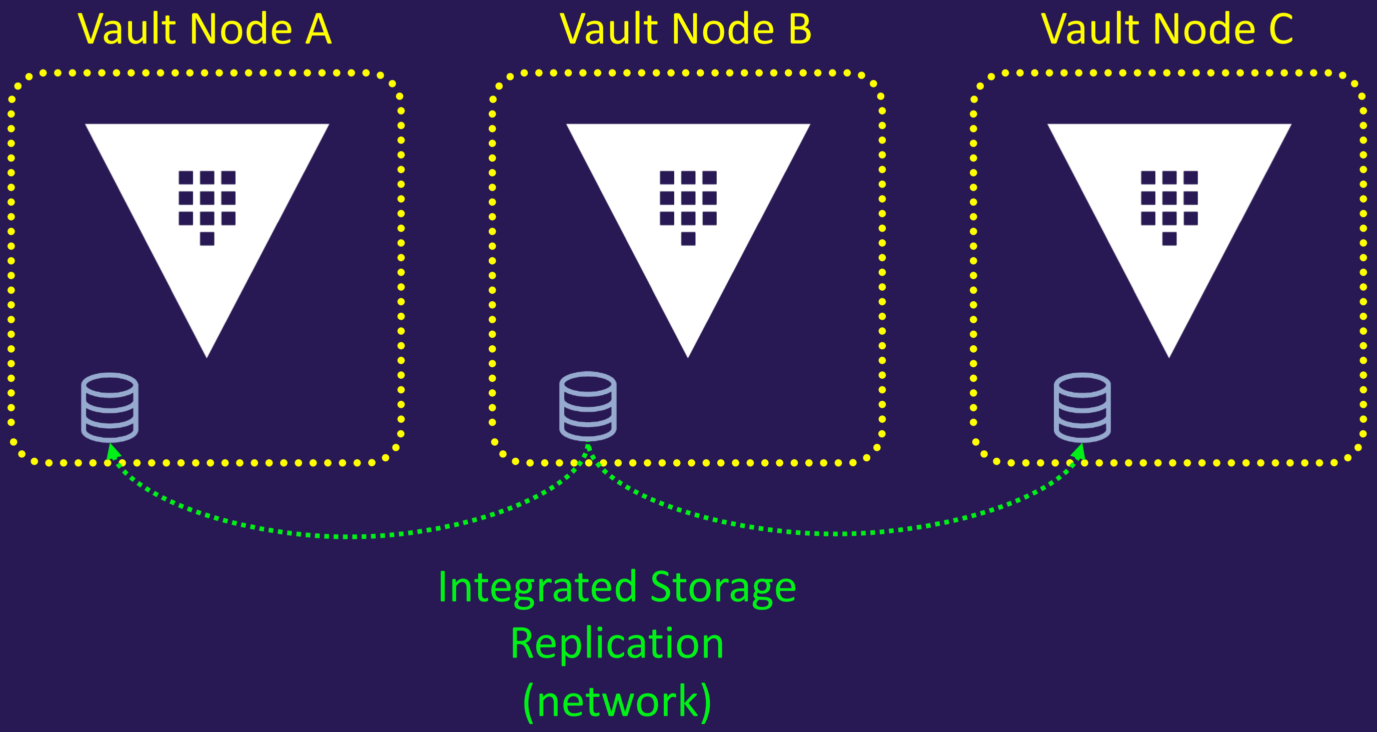


Not a Recommended Architecture

- No Redundancy
- No Scalability

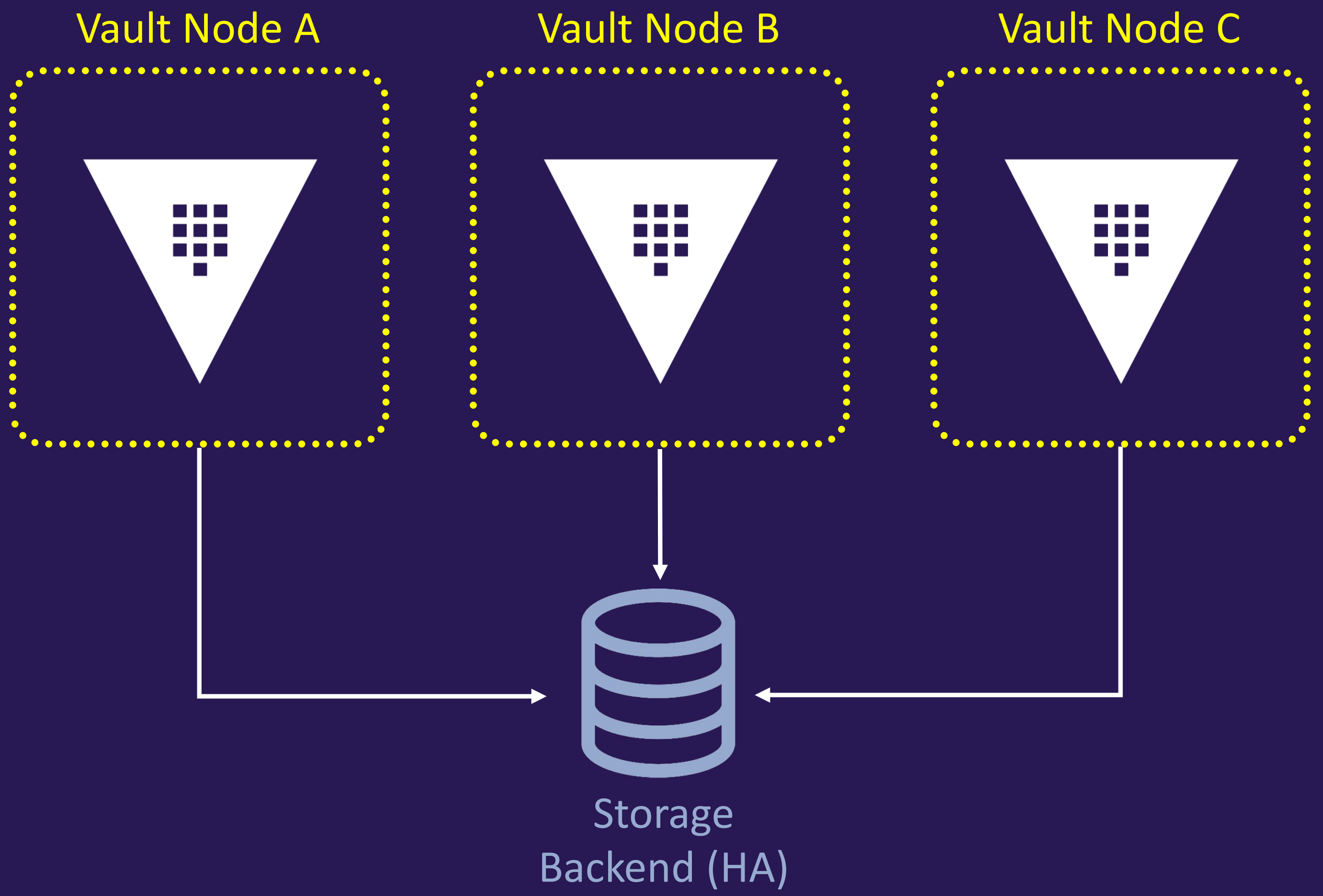
Running Vault Server in Production

Multi-Node Vault Cluster (with Integrated Storage)



Running Vault Server in Production

Multi-Node Vault Cluster (with external storage backend)



Running Vault Server in Production

Step-by-Step Manual Install

1. Download Vault from HashiCorp
2. Unpackage Vault to a Directory
3. Set Path to Executable
4. Add Configuration File & Customize
5. Create Systemd Service File
6. Download Consul from HashiCorp
7. Configure and Join Consul Cluster
8. Launch Vault Service

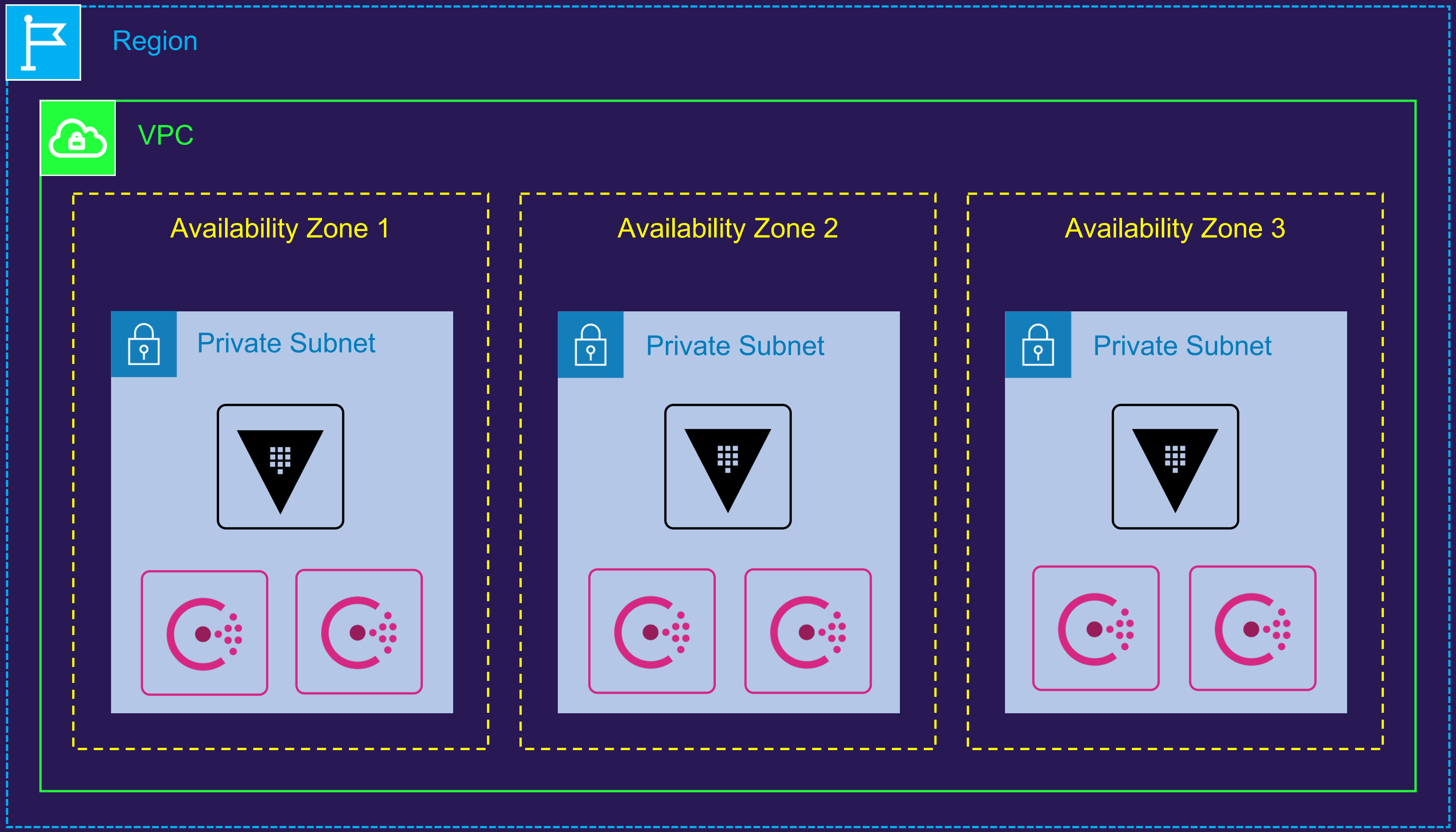
Deploying the Consul Storage Backend

- Provides Durable K/V Storage For Vault
- Can Independently Scale Backend
- Easy To Automate
- Built-in Integration Between Consul/Vault
- Supports High Availability
- Distributed System
- Built-in Snapshots For Data Retention
- HashiCorp Supported

Deploying the Consul Storage Backend

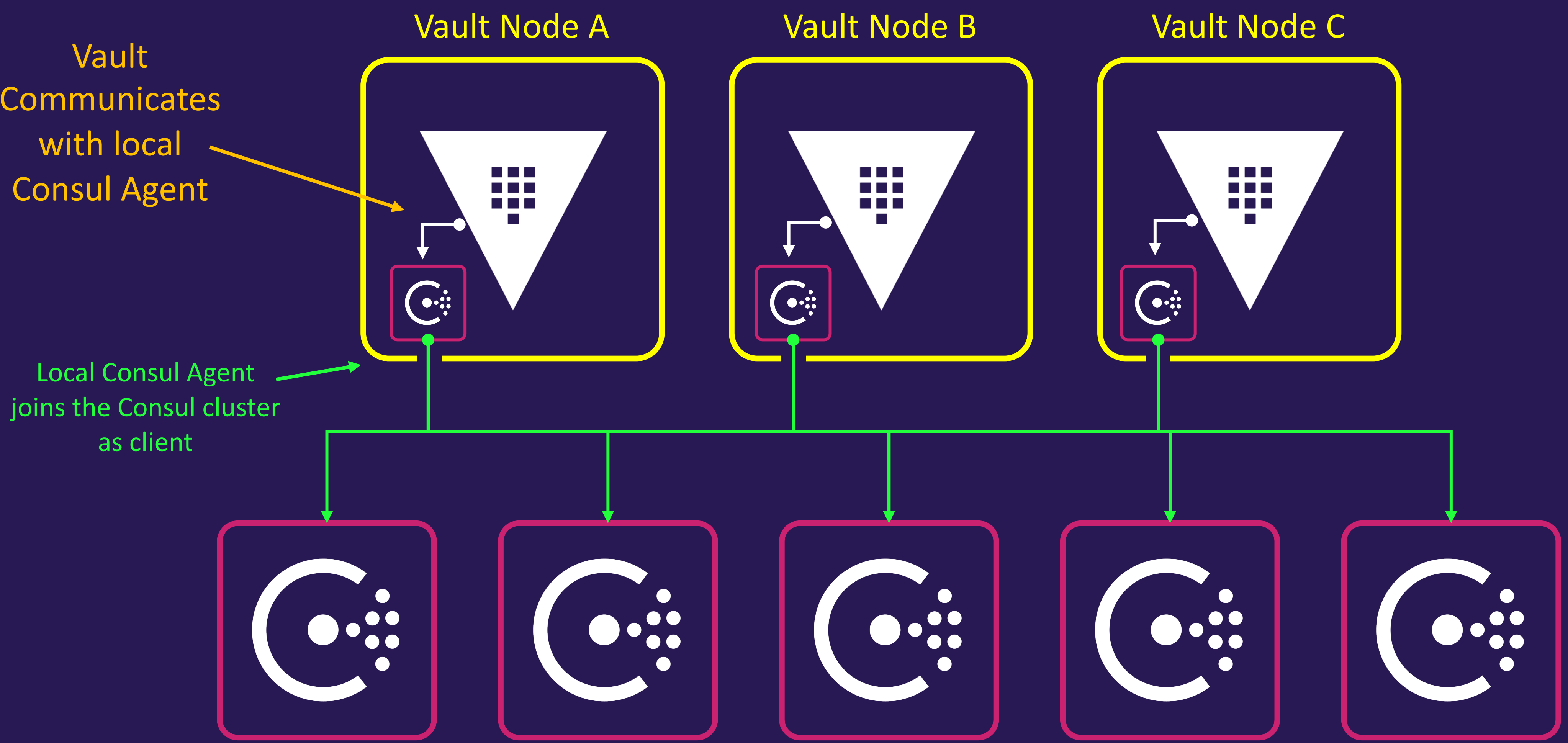
- Consul is deployed using multiple nodes and configured as a cluster
- Clusters are deployed in odd numbers (for voting members)
- All data is replicated among all nodes in the cluster
- A leader election promotes a single Consul node as the leader
- The leader accepts new logs entries and replicates to all other nodes
- Consul cluster for Vault storage backend shouldn't be used for Consul functions in a production setting

Deploying the Consul Storage Backend



Special Install of Consul using Redundancy Zones

Deploying the Consul Storage Backend


Vault Node A
Vault Node B
Vault Node C
Vault Communicates with local Consul Agent
Local Consul Agent joins the Consul cluster as client

Deploying the Consul Storage Backend

Example Consul Server Configuration File

```
 storage "consul" {
  address = "127.0.0.1:8500"
  path    = "vault/"
  token   = "1a2b3c4d-1234-abdc-1234-1a2b3c4d5e6a"
}
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"
 tls_key_file = "/etc/vault.d/cert.key"
 tls_disable_client_certs = "true"
}
seal "awskms" {
  region = "us-east-1"
  kms_key_id = "12345678-abcd-1234-abcd-123456789101",
  endpoint = "example.kms.us-east-1.vpce.amazonaws.com"
}
api_addr = "https://vault-us-east-1.example.com:8200"
cluster_addr = " https://node-a-us-east-1.example.com:8201"
cluster_name = "vault-prod-us-east-1"
ui = true
log_level = "INFO"
```

https://github.com/btkrausen/hashicorp/blob/master/vault/config_files/vault.hcl

Deploying the Consul Storage Backend

Example Consul Server Configuration File

```
{
 "log_level": "INFO",
 "server": true,
 "key_file": "/etc/consul.d/cert.key",
 "cert_file": "/etc/consul.d/client.pem",
 "ca_file": "/etc/consul.d/chain.pem",
 "verify_incoming": true,
 "verify_outgoing": true,
 "verify_server_hostname": true,
 "ui": true,
 "encrypt": "xxxxxxxxxxxxxx",
 "leave_on_terminate": true,
 "data_dir": "/opt/consul/data",
 "datacenter": "us-east-1",
 "client_addr": "0.0.0.0",
 "bind_addr": "10.11.11.11",
 "advertise_addr": "10.11.11.11",
 "bootstrap_expect": 5,
 "retry_join": ["provider=aws tag_key=Environment-Name tag_value=consul-cluster region=us-east-1"],
 "enable_syslog": true,
 "acl": {
    "enabled": true,
    "default_policy": "deny",
    "down_policy": "extend-cache",
    "tokens": {
      "agent": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
  }
},
  "performance": {
   "raft_multiplier": 1
 }
}
```

https://github.com/btkrausen/hashicorp/blob/master/consul/config.hcl

Looking for More on Consul?

For a deeper dive on Consul, check out my dedicated course on Consul:

Getting Started with HashiCorp Consul

Coupons Available on github.com/btkrausen/hashicorp

Deploying the Integrated Storage Backend

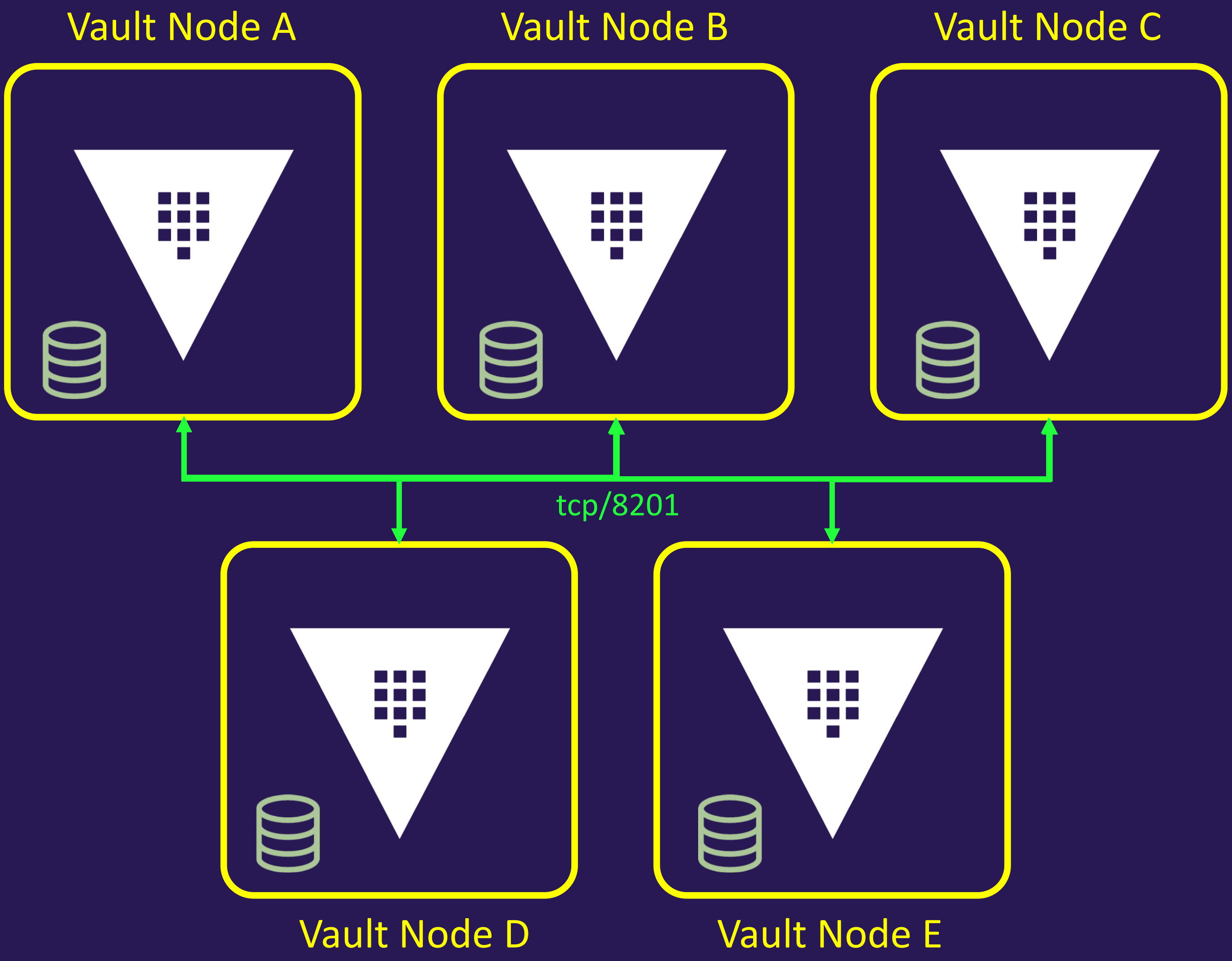
- Vault Internal Storage Option
- Leverages Raft Consensus Protocol
- All Vault nodes have copy of Vault's Data
- Eliminates Network Hop to Consul

- 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 also run a Consul cluster and manage it

Deploying the Integrated Storage Backend



Deploying the Integrated Storage Backend

Example Vault Server Configuration File

```
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"
 tls_key_file = "/etc/vault.d/cert.key"
 tls_disable_client_certs = "true"
}
seal "awskms" {
  region = "us-east-1"
  kms_key_id = "12345678-abcd-1234-abcd-123456789101",
  endpoint = "example.kms.us-east-1.vpce.amazonaws.com"
}
api_addr = "https://vault-us-east-1.example.com:8200"
cluster_addr = " https://node-a-us-east-1.example.com:8201"
cluster_name = "vault-prod-us-east-1"
ui = true
log_level = "INFO"
```

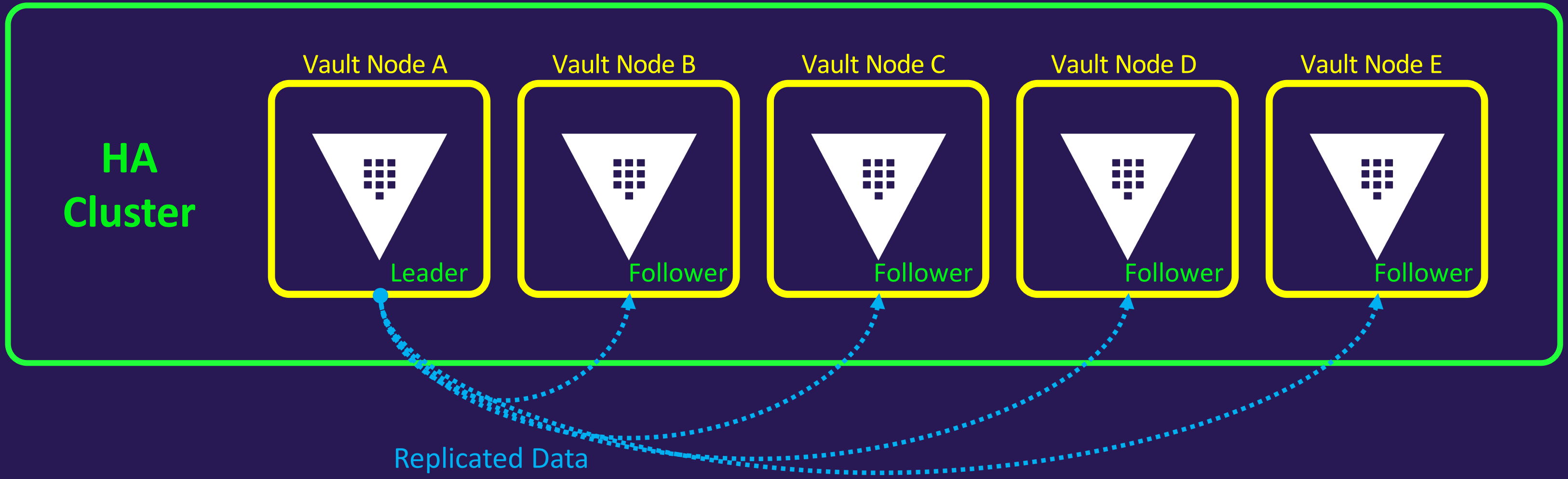
https://github.com/btkrausen/hashicorp/blob/master/vault/config_files/vault_int_storage.hcl

Deploying the Integrated Storage Backend

- Manually join standby nodes to the cluster using the CLI:

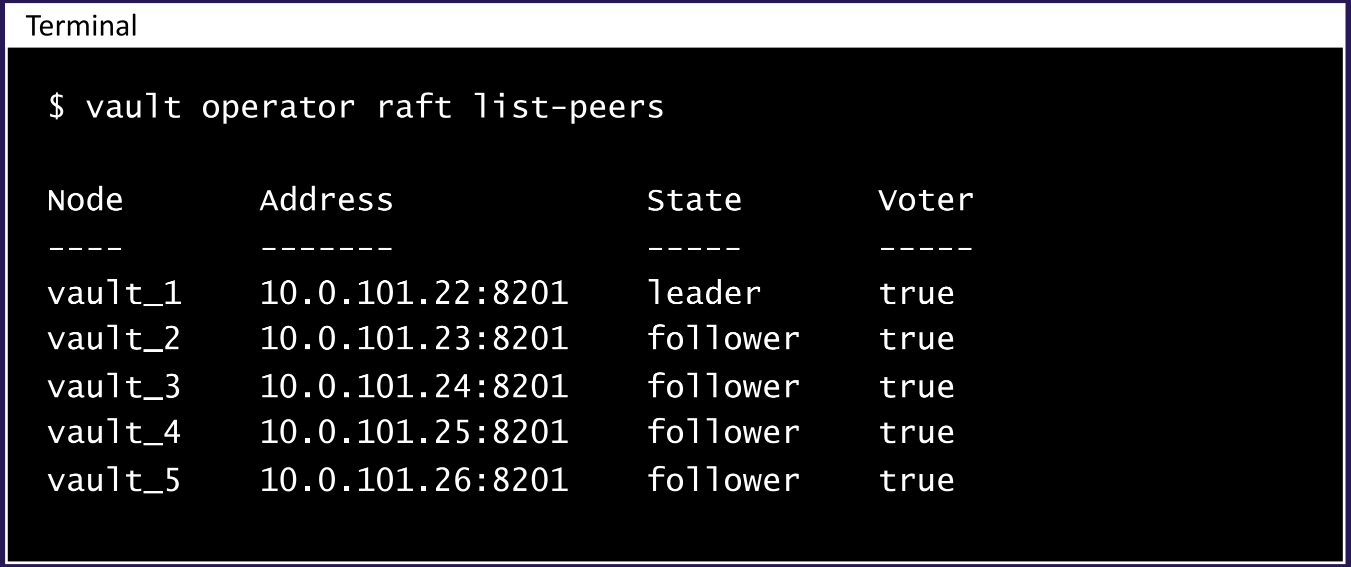
Terminal

```
$ 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
```

END OF SECTION