VAULT

Administrator Guide | v2.4 | July 2025

1. Production Architecture

A production Vault deployment requires careful planning for high availability, disaster recovery, and security hardening.

1.1 Recommended Architecture

Component	Recommendation
Cluster size	5 nodes (tolerates 2 failures)
Storage backend	Integrated Raft (no external dependencies)
Load balancer	Active node only — do not load-balance across all nodes
TLS	Valid certificate from trusted CA — no self-signed in production
Unseal	Auto Unseal with AWS KMS, Azure Key Vault, or GCP Cloud KMS
Backups	Raft snapshots to S3/GCS/Azure Blob — at minimum daily

1.2 Production Configuration

Example vault.hcl for a production node:

```
ui = true cluster_name = "vault-prod"  storage "raft" {   path    = "/vault/data"
node_id = "vault-prod-1"    retry_join {     auto_join             = "provider=aws
tag_key=vault-cluster tag_value=prod"     auto_join_scheme      = "https"
leader_tls_servername = "vault.internal.example.com"   } }  seal "awskms" {   region
= "us-east-1"   kms_key_id = "alias/vault-unseal-key" }  listener "tcp" {   address
= "0.0.0.0:8200"   cluster_address    = "0.0.0.0:8201"   tls_cert_file      =
"/vault/tls/vault.crt"   tls_key_file       = "/vault/tls/vault.key"   tls_min_version
= "tls12" }  telemetry {   prometheus_retention_time = "30s"   disable_hostname
= false }  api_addr     = "https://vault.internal.example.com:8200" cluster_addr =
"https://vault-prod-1.internal:8201"
```

2. Initialization

2.1 Initialize Vault

Initialize the cluster from any node. This is done only once.

```
vault operator init -key-shares=5 -key-threshold=3 -format=json > vault-init.json
```

⚠ vault-init.json contains the unseal keys and root token. Store the unseal keys in separate, secure locations (hardware security modules, different geographic locations, or with different administrators). Never store all keys together.

2.2 Initial Configuration Checklist

1. Generate a root token and immediately revoke it after bootstrapping
2. Enable audit logging to at least two destinations
3. Enable AppRole or another non-root auth method
4. Create a break-glass policy and token with a long TTL
5. Disable the root token: vault token revoke <root-token>
6. Test the unseal procedure with a controlled failover

3. Access Control Administration

3.1 Policy Management

```
# List all policies vault policy list  # View a policy vault policy read myapp-policy  # Delete a policy vault policy delete old-policy
```

3.2 Token Management

```
# List all token accessors (no token values exposed) vault list auth/token/accessors  # Look up a token by accessor vault token lookup -accessor <accessor>  # Revoke all tokens with a specific policy vault token revoke -mode=path auth/token/create/myapp-role
```

3.3 Entity and Group Management

Vault Identity enables managing user identities across auth methods.

```
# Create an entity vault write identity/entity name=johndoe policies=developer-policy  # Create a group vault write identity/group name=developers policies=developer-policy \   member_entity_ids=<entity-id>
```

4. Secrets Engine Administration

4.1 Mount Management

```
# List all mounts vault secrets list -detailed  # Enable a new mount vault secrets enable -path=app-secrets -description="App secrets" kv-v2  # Tune a mount (set default lease TTL) vault secrets tune -default-lease-ttl=8h -max-lease-ttl=24h app-secrets  # Disable a mount (destructive — data is lost) vault secrets disable old-secrets
```

⚠ Disabling a secrets engine destroys all secrets stored in it. Back up data before disabling any mount.

4.2 Key Rotation

```
# Rotate the encryption key for a transit mount vault write -f transit/keys/mykey/rotate
# Configure automatic rotation vault write transit/keys/mykey/config
\   auto_rotate_period=2160h   # 90 days
```

5. Backup and Disaster Recovery

5.1 Automated Raft Snapshots

```
# Schedule daily snapshots via cron 0 2 * * * vault operator raft snapshot save /backup/vault-$(date +\%Y\%m\%d).snap && \   aws s3 cp /backup/vault-$(date +\%Y\%m\%d).snap s3://vault-backups/
```

5.2 Restore from Snapshot

```
# Restore to a new cluster vault operator raft snapshot restore vault-20250715.snap
```

5.3 Disaster Recovery Runbook

7. Identify the incident (node failure, data corruption, accidental deletion)
8. If cluster is unsealed and healthy, promote a standby node as leader
9. If cluster is compromised, restore from the latest Raft snapshot to new nodes
10. Rotate all secrets and credentials after a security incident
11. Review audit logs to identify the scope of any unauthorized access

6. Monitoring and Alerting

6.1 Critical Alerts

Alert	Condition	Action
Vault sealed	vault.core.active = 0	Immediate: trigger auto-unseal or manually unseal
All audit devices failed	vault.audit.log_request_failure > 0	Immediate: Vault stops accepting requests
High token count	vault.token.count > 50,000	Review for token leaks; revoke orphaned tokens
Lease expiry approaching	vault.expire.num_leases > threshold	Verify applications are renewing leases
Raft leadership change	vault.raft.leader_changes > 0	Investigate network stability between nodes

7. Security Hardening

7.1 OS-Level Hardening

- Run Vault as a dedicated non-root user (e.g., vault:vault)
- Use systemd memory locking (LimitMEMLOCK=infinity) to prevent swapping sensitive data
- Restrict file permissions: chmod 640 /vault/config/vault.hcl
- Enable kernel memory protection (mlock syscall)

7.2 Network Hardening

- Use a firewall to restrict access to ports 8200 (API) and 8201 (cluster)
- Enable mutual TLS between Vault nodes
- Place Vault behind a load balancer with strict health checks
- Use private DNS for cluster_addr — never expose cluster port publicly

7.3 Vault Configuration Hardening

- Disable the UI if not required: ui = false
- Enable raw storage endpoint protection
- Set max_lease_ttl to the minimum needed for your use case
- Regularly audit and rotate the Vault encryption key