Vault (Production-Style) & Integrations — Hands-On Cheat Sheet

Your Name

November 1, 2025

Read Me First

Scope. Pragmatic, production-*style* demo (not fully hardened). TLS is disabled in several steps for learning only; enable TLS and hardening before real use.
Dependencies. Vault CLI, Docker Compose, a terminal with `bash`, and `pdflatex` with `minted` (Pygments).

1 Production-Style Server: Minimal Config & Boot

Create `vault/config.hcl`:

```
storage "file" {
  path = "/var/lib/vault"
}

listener "tcp" {
  address     = "127.0.0.1:8200"
  tls_disable = 1   # LEARNING ONLY. Use TLS in real deployments.
}

disable_mlock = true
ui            = true
```

Start Vault (may require sudo):

```
sudo vault server -config=vault/config.hcl
```

In a second terminal, point the CLI at the server:

```
export VAULT_ADDR=http://127.0.0.1:8200
vault status    # should show "initialized: false"
```

Initialize & Unseal (Shamir)

```
vault operator init
# Copy: 5 unseal keys (key shards) + initial root token. Store separately.

# Provide 3 distinct unseal keys (example threshold 3 of 5):
vault operator unseal
vault operator unseal
vault operator unseal
```

```
# Login with the initial root token:
vault login <ROOT_TOKEN>
vault secrets list
```

2 Training Stack via Docker Compose

From the exercises root:

```
docker-compose -v
docker-compose up -d     # starts: vaultserver, ssh, mariadb, jenkins
export VAULT_ADDR=http://0.0.0.0:8200
vault status             # not initialized yet (first boot)
vault operator init
vault operator unseal    # 3x with different keys
vault login <ROOT_TOKEN>
vault secrets list
```

Jenkins UI (demo)

Open `http://localhost:8080`. (Setup demo occurs later.)

Stop/Start

```
docker-compose stop
docker-compose up -d     # re-unseal after restart
```

3 SSH Secrets Engine (One-Time Password / OTP)

Enable Engine & Role

```
vault login <ROOT_TOKEN>
vault secrets enable ssh

# Create role "admin" that issues OTP for user "vaultuser"
vault write ssh/roles/admin \
  key_type=otp \
  default_user=vaultuser \
  cidr_list="0.0.0.0/0,0.0.0.0/0"
```

Prepare SSH Server (inside the SSH container)

SSH in (password: `vaultpwd`):

```
ssh root@localhost -p 3000
```

Install `vault-ssh-helper` and test (script provided in exercises):

```
. ./vault-ssh-setup.sh
vault-ssh-helper -verify-only -config=/etc/vault-ssh-helper.d/config.hcl
```

Update PAM to call the helper (learning demo):

```
nano /etc/pam.d/sshd
# comment "auth include common-auth"
# add:
# auth requisite pam_exec.so quiet expose_authtok log=/var/log/vault-ssh-helper.log
↪  /usr/local/bin/vault-ssh-helper -config=/etc/vault-ssh-helper.d/config.hcl
# auth sufficient pam_exec.so quiet expose_authtok log=/var/log/vault-ssh-helper.log
↪  /usr/local/bin/vault-ssh-helper -config=/etc/vault-ssh-helper.d/config.hcl
```

Create the target login user:

```
adduser vaultuser
# (password value irrelevant for OTP demo)
exit
```

Request OTP and Log In

```
# Request an OTP for role "admin" to the SSH server's bridge IP:
vault write ssh/creds/admin ip=172.16.238.20
# => copy 'key' field (one-time password)

ssh vaultuser@localhost -p 3000   # paste OTP when prompted
# Try again with same OTP -> should FAIL (one-time)
```

4 Database Secrets Engine (MariaDB) — Dynamic Users

Enable & Configure Connection

```
docker-compose down && docker-compose up -d   # clean slate as needed
vault operator init && vault operator unseal && vault login <ROOT_TOKEN>

vault secrets enable database

# Configure DB connection (compatible plugin for MariaDB)
vault write database/config/my-mariadb \
  plugin_name=mysql-database-plugin \
  allowed_roles="datareader,datawriter" \
  connection_url="{{username}}:{{password}}@tcp(mariadb:3306)/" \
  username="root" password="mysql"
```

Define Roles (Capabilities via SQL)

```
# Read-only role
vault write database/roles/datareader \
  db_name=my-mariadb \
  creation_statements="CREATE USER '{{name}}' IDENTIFIED BY '{{password}}'; GRANT SELECT
  ↪  ON *.* TO '{{name}}';" \
  default_ttl=1h max_ttl=24h

# Read-write role
vault write database/roles/datawriter \
  db_name=my-mariadb \
  creation_statements="CREATE USER '{{name}}' IDENTIFIED BY '{{password}}'; GRANT ALL ON
  ↪  *.* TO '{{name}}';" \
  default_ttl=1h max_ttl=24h
```

Policies & Short-Lived Tokens

```
# Minimal policies (inline HCL examples, shown as plain text for portability)
cat > datareader.hcl <<'HCL'
path "database/creds/datareader" {
  capabilities = ["read"]
}
HCL

cat > datawriter.hcl <<'HCL'
path "database/creds/datawriter" {
  capabilities = ["read"]
}
HCL

vault policy write datareader datareader.hcl
vault policy write datawriter datawriter.hcl

# Issue tokens bound to policies
vault token create -policy=datareader   # copy token A
vault token create -policy=datawriter   # copy token B
```

Generate Credentials and Use Them

Reader session

```
vault login <TOKEN_A_DATAREADER>
vault read database/creds/datareader     # yields username/password

# Use MySQL CLI in SSH container
ssh root@localhost -p 3000    # password: vaultpwd
mysql -u <DB_USER> -p<DB_PASS> -h mariadb
# Inside MySQL:
#   SHOW DATABASES;
#   CREATE DATABASE testDB;   # should fail (read-only)
```

Writer session

```
vault login <TOKEN_B_DATAWRITER>
vault read database/creds/datawriter     # yields username/password

ssh root@localhost -p 3000
mysql -u <DB_USER> -p<DB_PASS> -h mariadb
# Inside MySQL:
#   CREATE DATABASE testDB;   # should succeed
#   SHOW DATABASES;
```

5 Security Notes (Before Real Use)

- **Always enable TLS** on listeners; manage certs/keys securely.
- Store unseal keys separately (use *auto-unseal* with a KMS/HSM).
- Replace root token with tightly-scoped tokens tied to policies.
- Lock down Docker networks/ports; segregate secrets backends.
- Instrument audit devices; monitor leases and revocations.

Troubleshooting

- `vault status` shows `sealed=true`: run `vault operator unseal` with distinct keys until threshold met.
- CLI fails to connect: ensure `VAULT_ADDR` is set and listener address matches.
- OTP rejected: remember it's single-use; request a fresh OTP.
- DB auth fails: verify plugin, host `mariadb`, root password, and role SQL grants.