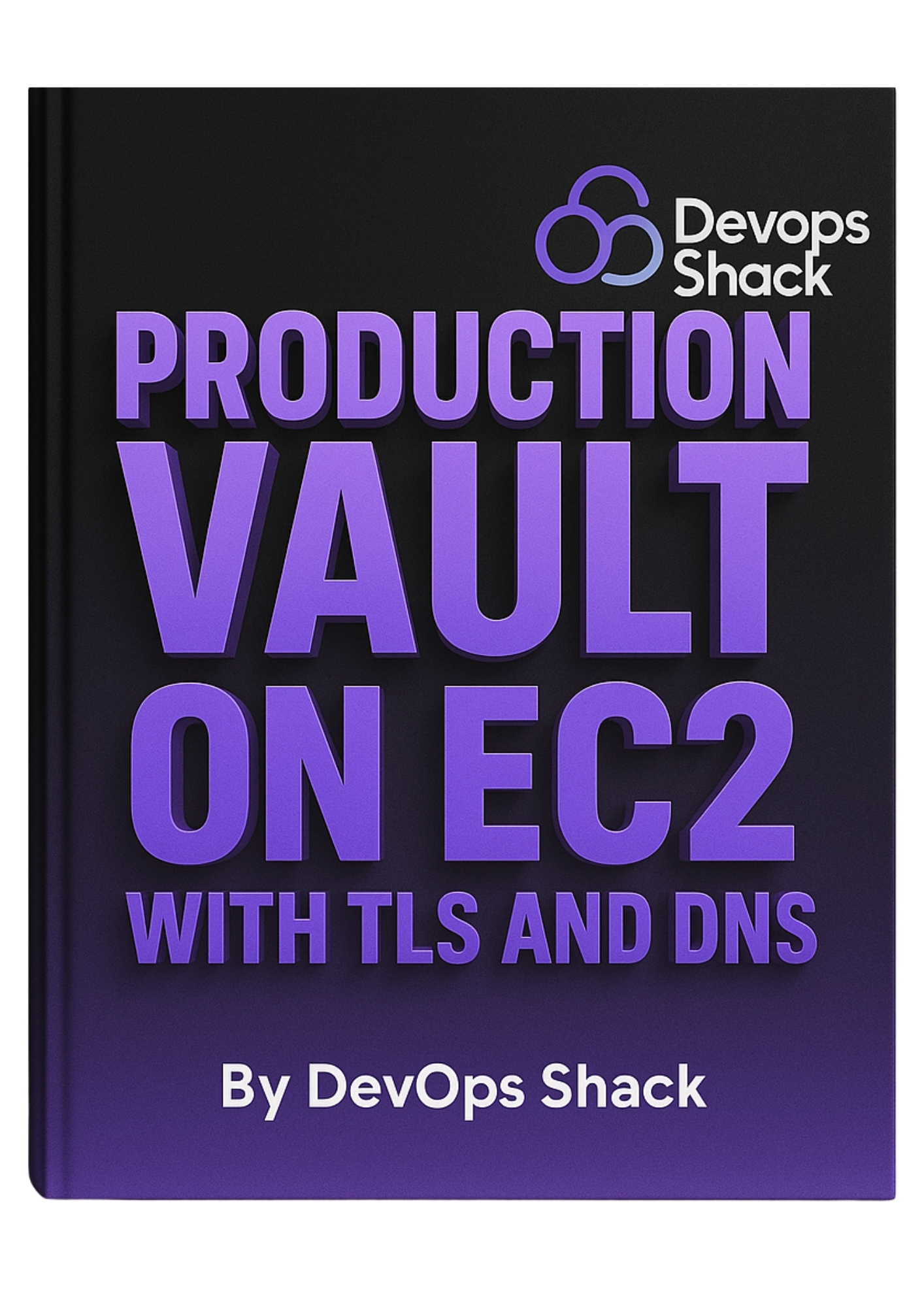
****

**Complete Production Setup of HashiCorp Vault on EC2 with TLS and DNS**

**✅ Step 1: Install Required Packages**

sudo apt update && sudo apt upgrade -y

sudo apt install -y unzip curl jq gnupg snapd

Install Certbot:

sudo snap install core; sudo snap refresh core

sudo snap install --classic certbot

sudo ln -s /snap/bin/certbot /usr/bin/certbot

**✅ Step 2: Install Vault (Official Repo)**

curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg

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 -y

**✅ Step 3: Create Vault User & Directories**

sudo useradd --system --home /etc/vault.d --shell /bin/false vault

sudo mkdir -p /opt/vault/data

sudo mkdir -p /etc/vault.d

**✅ Step 4: Generate TLS Cert with Let’s Encrypt**

Make sure your DNS (vault.skjptpp.in) points to the EC2 public IP.

Then run:

sudo certbot certonly --standalone -d vault.skjptpp.in

This creates certs at:  
/etc/letsencrypt/live/vault.skjptpp.in/fullchain.pem  
/etc/letsencrypt/live/vault.skjptpp.in/privkey.pem

**❗️Fix 1: Vault Fails to Read Let's Encrypt Certs (Permission Denied)**

**✅ Fix: Copy Certs to Vault Directory & Fix Permissions**

sudo cp /etc/letsencrypt/live/vault.skjptpp.in/fullchain.pem /etc/vault.d/vault.crt

sudo cp /etc/letsencrypt/live/vault.skjptpp.in/privkey.pem /etc/vault.d/vault.key

sudo chown vault:vault /etc/vault.d/vault.\*

sudo chmod 640 /etc/vault.d/vault.key

sudo chmod 644 /etc/vault.d/vault.crt

**✅ Step 5: Configure Vault (/etc/vault.d/vault.hcl)**

storage "file" {

path = "/opt/vault/data"

}

listener "tcp" {

address = "0.0.0.0:8200"

tls\_cert\_file = "/etc/vault.d/vault.crt"

tls\_key\_file = "/etc/vault.d/vault.key"

}

ui = true

disable\_mlock = true

api\_addr = "https://vault.skjptpp.in:8200"

cluster\_addr = "https://172.31.42.120:8201"

Replace <private-ip> with EC2’s internal IP (hostname -I).

**✅ Step 6: Vault systemd Service File**

sudo tee /etc/systemd/system/vault.service > /dev/null <<EOF

[Unit]

Description=Vault Server

Requires=network-online.target

After=network-online.target

[Service]

User=vault

Group=vault

ProtectSystem=full

ProtectHome=read-only

PrivateTmp=yes

PrivateDevices=yes

SecureBits=keep-caps

AmbientCapabilities=CAP\_IPC\_LOCK

Capabilities=CAP\_IPC\_LOCK+ep

CapabilityBoundingSet=CAP\_SYSLOG CAP\_IPC\_LOCK

ExecStart=/usr/bin/vault server -config=/etc/vault.d/vault.hcl

ExecReload=/bin/kill --signal HUP \$MAINPID

KillMode=process

Restart=on-failure

LimitNOFILE=65536

LimitMEMLOCK=infinity

[Install]

WantedBy=multi-user.target

EOF

**✅ Step 7: Start Vault**

sudo systemctl daemon-reexec

sudo systemctl enable vault

sudo systemctl start vault

sudo systemctl status vault

**✅ Step 8: Export VAULT\_ADDR**

export VAULT\_ADDR=https://vault.skjptpp.in:8200

Add it permanently (optional):

echo 'export VAULT\_ADDR=https://vault.skjptpp.in:8200' >> ~/.bashrc

source ~/.bashrc

**✅ Step 9: Initialize Vault**

vault operator init -key-shares=5 -key-threshold=3

⚠️ Save the 5 unseal keys and root token securely!

**✅ Step 10: Unseal Vault**

Use 3 of the 5 keys:

vault operator unseal <key-1>

vault operator unseal <key-2>

vault operator unseal <key-3>

Check:

vault status

**✅ Step 11: Login to Vault**

vault login <initial-root-token>

**✅ Step 12: Enable Secrets Engine**

vault secrets enable -path=secret kv-v2

Store and retrieve secrets:

vault kv put secret/myapp username=admin password=supersecure123

vault kv get secret/myapp

**# Create a Read-Only Token in Vault for Jenkins**

Login to Vault:

vault login <root-token>

Create a policy for Jenkins:

# jenkins-read-policy.hcl

path "secret/data/jenkins/\*" {

capabilities = ["read"]

}

Save and apply:

vault policy write jenkins-read jenkins-read-policy.hcl

Create a token using that policy:

vault token create -policy=jenkins-read -ttl=24h

vault kv put secret/jenkins/docker username=mydockeruser password=MyP@ssword123

pipeline {

agent any

environment {

VAULT\_ADDR = 'https://vault.skjptpp.in:8200'

}

stages {

stage('Use Vault Secrets') {

steps {

script {

withVault([

vaultSecrets: [

[

path: 'secret/jenkins/docker',

engineVersion: 2,

secretValues: [

[vaultKey: 'username', envVar: 'DOCKER\_USERNAME'],

[vaultKey: 'password', envVar: 'DOCKER\_PASSWORD']

]

]

],

vaultCredentialId: 'vault-cred'

]) {

sh '''

echo "Username from Vault: $DOCKER\_USERNAME"

echo "Password from Vault: $DOCKER\_PASSWORD"

'''

}

}

}

}

}

}

**Golbal Usage Of Secrets**

pipeline {

agent any

environment {

// Declare placeholders to avoid undeclared variable issues in later stages

DOCKER\_USERNAME = ''

DOCKER\_PASSWORD = ''

}

stages {

stage('Fetch Secrets from Vault') {

steps {

script {

withVault(

configuration: [

disableChildPoliciesOverride: false,

engineVersion: 2,

timeout: 60,

vaultCredentialId: 'vault-token',

vaultUrl: 'https://vault.skjptpp.in:8200'

],

vaultSecrets: [[

engineVersion: 2,

path: 'secret/jenkins/docker',

secretValues: [

[envVar: 'DOCKER\_USERNAME', vaultKey: 'username'],

[envVar: 'DOCKER\_PASSWORD', vaultKey: 'password']

]

]]

) {

// Capture to env-scoped vars so we can use them outside

env.DOCKER\_USERNAME = env.DOCKER\_USERNAME

env.DOCKER\_PASSWORD = env.DOCKER\_PASSWORD

echo "Fetched DOCKER\_USERNAME: ${env.DOCKER\_USERNAME}" // avoid printing secrets in real pipelines

}

}

}

}

stage('Use Secrets') {

steps {

script {

echo "Using secret outside: ${env.DOCKER\_USERNAME}" // just for demonstration

}

}

}

}

}

**🔁 Bonus: Renew Cert Automatically**

sudo crontab -e

Add:

0 3 \* \* \* certbot renew --post-hook "systemctl restart vault"