Automação Completa 1Password: Guia End-to-End

Versão: 1.0
Data: 2025-10-22
Autor: Luiz Sena

1. Overview e Arquitetura

1.1. Objetivo

Automação end-to-end de **secrets management** usando 1Password para gerenciar credenciais de infraestrutura, aplicações e CI/CD de forma segura, escalável e auditável[137][139][^144].

1.2. Componentes Principais

Componente	Descrição	Caso de Uso
1Password Connect Server	API REST self-hosted	Unlimited requests, cache local, baixa latência[137][146]
1Password CLI v2	Ferramenta command-line	Automação de scripts, CI/CD, secret references[139][142]
1Password Service Accounts	Autenticação de máquinas	CI/CD sem interação humana[144][151]
1Password Kubernetes Operator	Sincronização K8s	Auto-sync secrets para Kubernetes[^141]

1.3. Casos de Uso

- ✓ Provisionar secrets em **CI/CD pipelines** (GitHub Actions, GitLab CI, Jenkins)
- ✓ Sincronizar secrets com **Kubernetes** automaticamente
- ✓ Automatizar gestão de credenciais de infraestrutura (AWS, databases, APIs)
- ✓ Integrar secrets em aplicações via **API/SDK** (Python, Go, JS)
- ✓ Rotação automática de secrets

2. Arquitetura

2.1. Camadas

Vault Layer (Source of Truth)

Vaults compartilhados organizados por:

Vault	Conteúdo
`Infra Prod`	Secrets de produção
`Infra Staging`	Secrets de staging
`Infra Dev`	Secrets de desenvolvimento
`CI/CD`	Secrets de pipelines
`Databases`	Credenciais de databases

Best Practices:

- Princípio de **least privilege**
- Segregação por ambiente
- Nomenclatura consistente
- Tags para categorização

Connect Layer (API REST)

Componentes:

- `connect-api`: serve REST API
- `connect-sync`: sincroniza com 1Password

Vantagens:[137][144]

- Unlimited requests (sem rate limits)
- Baixa latência (cache local)
- Self-hosted (controle total)
- Alta disponibilidade (deploy redundante)

Deployment Options:

- Docker Compose
- Kubernetes (Helm Chart)
- Railway/Cloud Run
- AWS ECS/Fargate

Automation Layer

- 1Password CLI v2
- SDKs (Python, Go, JavaScript)
- Shell scripts
- Terraform/Pulumi providers
- CI/CD integrations

2.2. Fluxo de Dados

1. Secrets criados/atualizados em **1Password Vault**
2. **Connect Server** sincroniza mudanças (connect-sync)
3. Aplicações consultam via **Connect API/CLI**
4. Secrets injetados em runtime (env vars, config files)
5. **Logs de auditoria** mantidos automaticamente

3. Setup Connect Server

3.1. Step 1: Criar Secrets Automation Workflow

Via Web[^146]

1. Acessar `https://my.1password.com`
2. Developer > Directory > Infrastructure Secrets Management
3. Create a Connect server
4. Selecionar vaults com acesso
5. Baixar `1password-credentials.json`
6. Copiar access token

Via CLI[^146]

```
# Criar Connect server
op connect server create "My Connect Server" --vaults="Infra Prod,Infra Staging"

# Gerar token
op connect token create "My Connect Server" --server="My Connect Server"

# Salvar token no 1Password
op item create --category=password --title="Connect Token" --vault="Private" password=&lt
```

Outputs:

- `1password-credentials.json` (para deploy do server)

- Access token (para autenticação de apps)

3.2. Step 2: Deploy Connect Server

Docker Compose[137][140]

```
version: '3.8'

services:
  connect-api:
    image: 1password/connect-api:latest
    ports:
      - "8080:8080"
    environment:
      - OP_SESSION=/home/opuser/.op/data/1password-credentials.json
    volumes:
      - ./1password-credentials.json:/home/opuser/.op/data/1password-credentials.json:ro
      - op-data:/home/opuser/.op/data
    depends_on:
      - connect-sync

  connect-sync:
    image: 1password/connect-sync:latest
    environment:
      - OP_SESSION=/home/opuser/.op/data/1password-credentials.json
    volumes:
      - ./1password-credentials.json:/home/opuser/.op/data/1password-credentials.json:ro
      - op-data:/home/opuser/.op/data

volumes:
  op-data:
```

```
# Subir o Connect Server
docker-compose up -d

# Validar
curl -H "Authorization: Bearer &lt;TOKEN&gt;" http://localhost:8080/v1/health
```

Kubernetes Helm[141][146]

```
# Adicionar repo Helm
helm repo add 1password https://1password.github.io/connect-helm-charts
helm repo update

# Criar secret com credentials
kubectl create namespace 1password
kubectl create secret generic op-credentials \
  --from-file=1password-credentials.json \
  -n 1password

# Instalar Connect + Operator
```

```
helm install connect 1password/connect \
  --set-file connect.credentials=1password-credentials.json \
  --set operator.create=true \
  --set operator.token.value=&lt;ACCESS_TOKEN&gt; \
  --namespace 1password
```

3.3. Step 3: Configurar Access Tokens

Best Practices:[^142]

- 1 token por aplicação (isolamento)
- Least privilege: acesso apenas aos vaults necessários
- Tokens de curta duração para CI/CD (rotação)
- Armazenar tokens em 1Password (não hardcode)

```
# Listar tokens existentes
op connect token list --server="My Connect Server"

# Criar novo token
op connect token create "App Production" --server="My Connect Server"

# Revogar token
op connect token revoke &lt;TOKEN_ID&gt; --server="My Connect Server"
```

4. Automação via CLI

4.1. Instalação[^139]

```
# macOS
brew install 1password-cli

# Linux
curl -sSO https://downloads.1password.com/linux/debian/amd64/stable/1password-cli-latest-
sudo dpkg -i 1password-cli-latest-amd64.deb

# Windows
winget install 1Password.CLI

# Docker
docker pull 1password/op:2
```

4.2. Autenticação

Biometric (Local)[^139]

```
# Conectar ao app 1Password desktop
eval $(op signin)

# Usar biometria (Touch ID, Apple Watch)
op vault list  # Solicita biometria automaticamente
```

Service Account (CI/CD)[144][151]

```
# Usar Service Account (sem interação humana)
export OP_SERVICE_ACCOUNT_TOKEN=&lt;TOKEN&gt;

# Comandos funcionam automaticamente
op vault list
op item get "Database Password" --vault="Infra Prod"
```

Connect Server[^139]

```
# Configurar para usar Connect Server
export OP_CONNECT_HOST=http://localhost:8080
export OP_CONNECT_TOKEN=&lt;ACCESS_TOKEN&gt;

# Comandos usam Connect automaticamente
op vault list
op item get "API Key" --vault="CI/CD"
```

4.3. Operações Comuns

Criar Item[^142]

```
# Template JSON
cat &gt; template.json &lt;&lt;EOF
{
  "title": "Database Production",
  "category": "login",
  "fields": [
    {"id": "username", "label": "username", "value": "admin"},
    {"id": "password", "label": "password", "value": "super-secret-123"},
    {"id": "hostname", "label": "hostname", "value": "db.prod.example.com"},
    {"id": "port", "label": "port", "value": "5432"}
  ]
}
EOF

# Criar item via template
cat template.json | op item create --vault="Infra Prod"

# Criar item inline
```

```
op item create \
  --category=password \
  --title="API Token" \
  --vault="CI/CD" \
  token=ghp_abc123xyz
```

Buscar Secret[136][139]

```
# Buscar password de um item
op item get "Database Production" --vault="Infra Prod" --fields password

# Buscar campo específico
op item get "Database Production" --vault="Infra Prod" --fields hostname

# Output JSON completo
op item get "Database Production" --vault="Infra Prod" --format=json | jq
```

Listar Items[^148]

```
# Listar todos os items de um vault
op item list --vault="Infra Prod"

# Filtrar por categoria
op item list --categories=password --vault="CI/CD"

# Output JSON para processamento
op item list --vault="Infra Prod" --format=json | jq '.[] | {title, id}'
```

Atualizar Secret

```
# Atualizar campo específico
op item edit "API Token" --vault="CI/CD" token=new-token-value

# Atualizar múltiplos campos
op item edit "Database Production" \
  password=new-password \
  hostname=db-new.prod.example.com
```

5. Integração CI/CD

5.1. GitHub Actions[139][147]

Setup:

1. Adicionar Service Account token aos Secrets
2. Settings > Secrets > Actions > New repository secret
3. Name: `OP_SERVICE_ACCOUNT_TOKEN`

4. Value: `ops_<token>`

Workflow:

```
name: Deploy with 1Password Secrets

on:
  push:
    branches: [main]

jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3

      - name: Install 1Password CLI
        uses: 1password/install-cli-action@v1

      - name: Load secrets from 1Password
        env:
          OP_SERVICE_ACCOUNT_TOKEN: ${{ secrets.OP_SERVICE_ACCOUNT_TOKEN }}
        run: |
          # Export secrets como env vars
          export AWS_ACCESS_KEY_ID=$(op item get "AWS Credentials" --vault="CI/CD" --fiel
          export AWS_SECRET_ACCESS_KEY=$(op item get "AWS Credentials" --vault="CI/CD" --

          # Ou usar secret references
          op run --env-file=".env.op" -- npm run deploy

      - name: Deploy to production
        run: |
          # Secrets já disponíveis como env vars
          ./deploy.sh
```

5.2. GitLab CI

```
stages:
  - deploy

deploy_prod:
  stage: deploy
  image: 1password/op:2
  variables:
    OP_SERVICE_ACCOUNT_TOKEN: $OP_SERVICE_ACCOUNT_TOKEN
  script:
    - op vault list
    - export DB_PASSWORD=$(op item get "Database Prod" --vault="Infra Prod" --fields pass
    - ./deploy.sh
  only:
    - main
```

5.3. Jenkins

```
pipeline {
    agent any

    environment {
        OP_SERVICE_ACCOUNT_TOKEN = credentials('op-service-account-token')
    }

    stages {
        stage('Load Secrets') {
            steps {
                sh '''
                    curl -sSO https://downloads.1password.com/linux/debian/amd64/stable/1
                    sudo dpkg -i 1password-cli-latest-amd64.deb

                    export DB_PASSWORD=$(op item get "Database Prod" --vault="Infra Prod'
                    echo "Secret loaded successfully"
                '''
            }
        }

        stage('Deploy') {
            steps {
                sh './deploy.sh'
            }
        }
    }
}
```

6. Integração Kubernetes

6.1. 1Password Operator[^141]

Instalação: Já incluído se instalou Connect via Helm com `--set operator.create=true`

Uso Básico

1. Criar secret no 1Password

Item no vault `Infra Prod` com campos: `username`, `password`, `hostname`

2. Criar OnePasswordItem CRD:

```
apiVersion: onepassword.com/v1
kind: OnePasswordItem
metadata:
  name: database-credentials
  namespace: production
spec:
  itemPath: "vaults/Infra Prod/items/Database Production"
```

3. Usar secret no Pod:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: app
  namespace: production
spec:
  template:
    spec:
      containers:
      - name: app
        image: myapp:latest
        env:
        - name: DB_USERNAME
          valueFrom:
            secretKeyRef:
              name: database-credentials
              key: username
        - name: DB_PASSWORD
          valueFrom:
            secretKeyRef:
              name: database-credentials
              key: password
```

Auto-Restart on Update[^141]

Operator detecta mudanças e restart pods automaticamente:

```
metadata:
  annotations:
    operator.1password.io/auto-restart: "true"
```

7. Secret References

7.1. Formato[139][147]

```
op://&lt;vault&gt;/&lt;item&gt;[/&lt;section&gt;]/&lt;field&gt;
```

7.2. Uso com Env Vars[^154]

Arquivo `.env.op`**:**

```
DATABASE_URL=op://Infra Prod/Database Production/connection_string
API_KEY=op://CI/CD/External API/api_key
AWS_ACCESS_KEY_ID=op://Infra Prod/AWS Credentials/access_key_id
```

Carregar secrets:

```
# Injetar secrets no ambiente
op run --env-file=.env.op -- npm start

# Ou export direto
op run --env-file=.env.op -- env
```

7.3. Uso com Config Files

Template `config.yaml.op`:

```
database:
  host: op://Infra Prod/Database Production/hostname
  port: op://Infra Prod/Database Production/port
  username: op://Infra Prod/Database Production/username
  password: op://Infra Prod/Database Production/password

api:
  key: op://CI/CD/External API/api_key
  endpoint: https://api.example.com
```

Injetar secrets:

```
# Gerar config.yaml com secrets
op inject -i config.yaml.op -o config.yaml

# Ou usar direto
cat config.yaml.op | op inject | kubectl apply -f -
```

8. SDKs Programáticos

8.1. Python

```
pip install onepassword-sdk
```

```
from onepassword import Client, OP_CONNECT_HOST, OP_CONNECT_TOKEN
import os

# Configurar client
client = Client(
    url=os.environ.get('OP_CONNECT_HOST'),
    token=os.environ.get('OP_CONNECT_TOKEN')
)

# Buscar secret
item = client.items.get('Database Production', vault='Infra Prod')
```

```
password = item.fields['password'].value

# Listar vaults
vaults = client.vaults.list()
for vault in vaults:
    print(f"Vault: {vault.name}")

# Criar item
new_item = client.items.create(
    vault='CI/CD',
    title='New API Token',
    category='password',
    fields=[
        {'label': 'token', 'value': 'abc123xyz'}
    ]
)
```

8.2. Go

```
go get github.com/1Password/connect-sdk-go
```

```
package main

import (
    "context"
    "fmt"
    "os"

    "github.com/1Password/connect-sdk-go/connect"
    "github.com/1Password/connect-sdk-go/onepassword"
)

func main() {
    client := connect.NewClient(
        os.Getenv("OP_CONNECT_HOST"),
        os.Getenv("OP_CONNECT_TOKEN"),
    )

    // Get item
    item, err := client.GetItem("Database Production", "Infra Prod")
    if err != nil {
        panic(err)
    }

    password := item.Fields[^0].Value
    fmt.Printf("Password: %s\n", password)
}
```

8.3. JavaScript

```
npm install @1password/connect
```

```
import { OnePasswordConnect } from '@1password/connect';

const op = OnePasswordConnect({
  serverURL: process.env.OP_CONNECT_HOST,
  token: process.env.OP_CONNECT_TOKEN,
  keepAlive: true
});

// Get secret
const item = await op.getItem('Database Production', 'Infra Prod');
const password = item.fields.find(f =&gt; f.label === 'password').value;

// List vaults
const vaults = await op.listVaults();
vaults.forEach(vault =&gt; {
  console.log(`Vault: ${vault.name}`);
});
```

9. Best Practices

9.1. Least Privilege[^142]

- ✓ 1 token por aplicação/serviço
- ✓ Acesso apenas aos vaults necessários
- ✓ Service Accounts em vez de contas pessoais
- ✓ Revogar tokens não utilizados

9.2. Rotação de Secrets

- ✓ Rotação automática via scripts
- ✓ Monitorar idade dos secrets
- ✓ Alertar quando expiração próxima
- ✓ Documentar procedimentos de emergência

9.3. Armazenamento de Tokens[^142]

- ✗ **NUNCA** hardcode tokens em código
- ✗ **NUNCA** commit tokens no Git
- ✓ Armazenar tokens em 1Password
- ✓ Usar env vars em runtime

- ✓ Tokens de curta duração para CI/CD

9.4. Auditoria

- ✓ Ativar Events API para logs
- ✓ Monitorar acesso aos secrets
- ✓ Alertas para padrões suspeitos
- ✓ Revisão periódica de permissões

10. Troubleshooting

10.1. Connect Server Não Responde

```
# Verificar logs
docker logs connect-api
docker logs connect-sync

# Testar conectividade
curl http://localhost:8080/v1/health

# Validar credentials
cat 1password-credentials.json | jq
```

10.2. CLI Não Autentica

```
# Verificar sessão
op whoami

# Re-signin
eval $(op signin)

# Verificar env vars
echo $OP_SERVICE_ACCOUNT_TOKEN
echo $OP_CONNECT_HOST
echo $OP_CONNECT_TOKEN
```

10.3. Kubernetes Secrets Não Sincronizam

```
# Verificar Operator logs
kubectl logs -n 1password -l app=connect-operator

# Verificar OnePasswordItem status
kubectl describe onepassworditem database-credentials -n production

# Validar itemPath format
# Correto: vaults/&lt;VAULT_NAME&gt;/items/&lt;ITEM_NAME&gt;
```

Referências

- 1Password Connect Server: https://developer.1password.com/docs/connect/
- 1Password CLI: https://developer.1password.com/docs/cli
- Best Practices: https://developer.1password.com/docs/cli/best-practices/
- Kubernetes Operator: https://developer.1password.com/docs/k8s/
- Service Accounts: https://developer.1password.com/docs/service-accounts/
  [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20]

⁂

1. https://austincloud.guru/2018/11/27/1password-cli-tricks/
2. https://forum.keyboardmaestro.com/t/best-securest-way-to-pass-data-from-1password-cli/25974
3. https://developer.1password.com/docs/connect/get-started/
4. https://1password.com/resources/guides/managing-developer-secrets-with-1password/
5. https://developer.1password.com/docs/cli/reference
6. https://github.com/1Password/connect
7. https://engineering.kraken.tech/news/2024/09/02/onepassword-connect.html
8. https://www.1password.community/discussions/developers/is-it-possible-to-use-1password-cli-on-cicd-scripts/87399
9. https://mockoon.com/mock-samples/1passwordlocal-connect/
10. https://1password.com/developers/secrets-management
11. https://www.gruntwork.io/blog/how-to-securely-store-secrets-in-1password-cli-and-load-them-into-your-zsh-shell-when-needed
12. https://developer.1password.com/docs/connect/
13. https://hub.docker.com/r/1password/connect-api
14. https://hoop.dev/blog/the-simplest-way-to-make-1password-aws-secrets-manager-work-like-it-should/
15. https://developer.1password.com/docs/cli
16. https://railway.com/deploy/1password-connect-api-and-server
17. https://blog.enapi.com/secrets-management-in-kubernetes-with-1password-92e3f8f08ce0
18. https://developer.1password.com/docs/cli/best-practices/
19. https://developer.1password.com/docs/connect/api-reference/
20. https://developer.1password.com/docs/secrets-automation/