Dyskrecja DevOpsa

Co robimy w ukryciu

Michał Borkowski

https://norasoft.eu/talks/secrecy-of-devops/

michal.borkowski@xebia.com

O czym dzisiaj?

- co robimy w ukryciu i dlaczego się kryjemy?
- jak zabezpieczać środowisko lokalne?
- ale jak to automatyzować?
- usługi zewnętrzne
- co z produkcją?

Niejawne informacje

Niejawne informacje

jak przekazywać i gdzie trzymać

Niejawne informacje

jak przekazywać i gdzie trzymać

- w środowisku lokalnym
- w czasie buildu
- w czasie wykonania

Po co się starać?

Po co się starać?

BBC

Po co się starać?

BBC

Because

Po co się starać?

BBC

Because

Benedict

Po co się starać?

BBC

Because

Benedict

Cumberbatch

Thank you!

https://norasoft.eu/talks/secrecy-of-devops/

michal.borkowski@xebia.com

Sekrety projektu

Projekt

```
def main():
    with open('secret.yml', 'r') as secret:
        secrets = yaml.load(secret, yaml.Loader)

        message = f'''
        Login credentials:
        username: {secrets['secret']['username']}
        password: {secrets['secret']['password']}
        '''
        print(message)
```

Wersja alpha

secret.yml

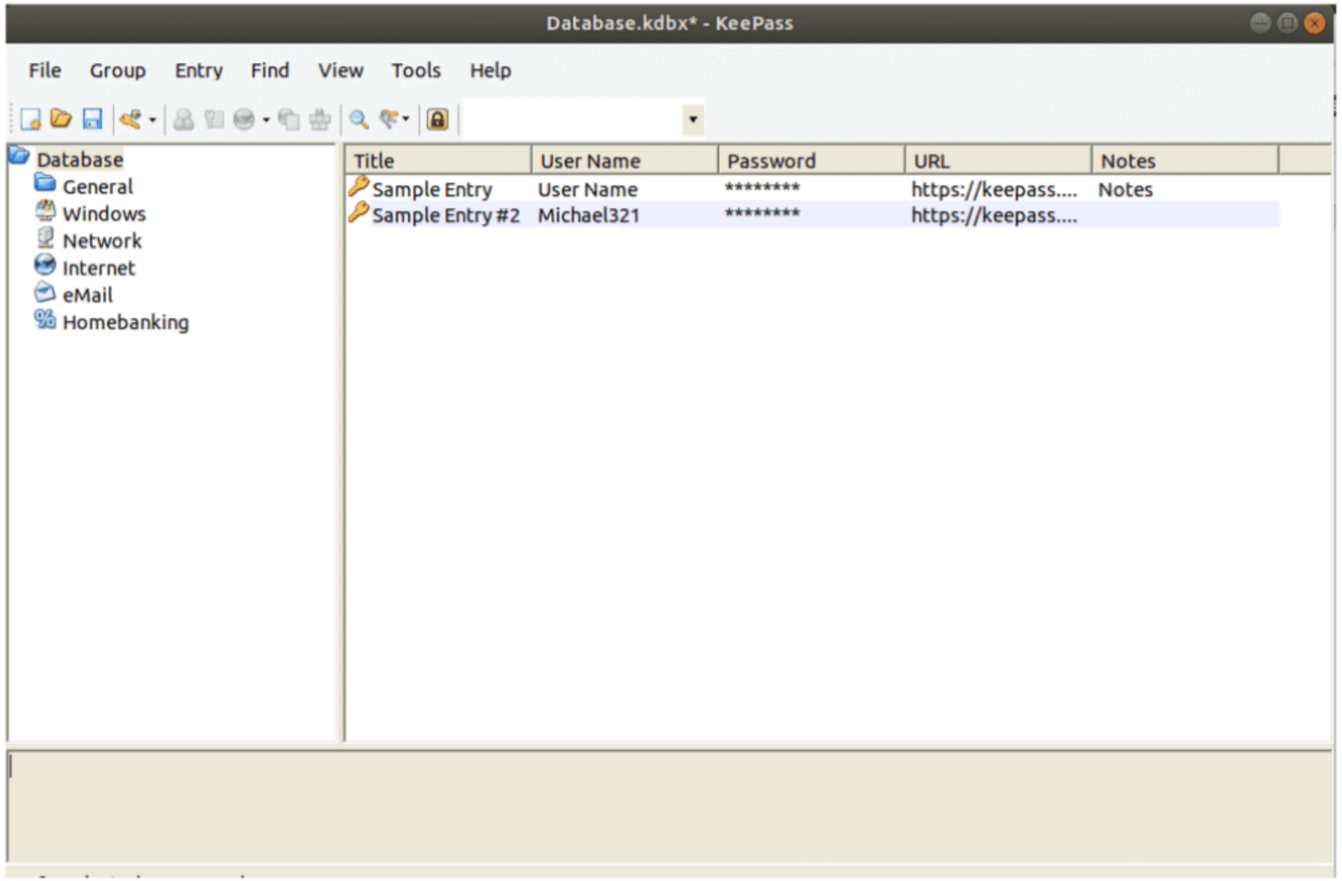
```
secret:
  username: user
  password: secretpassword
```

Wersja alpha

secret.example.yml

```
secret:
  username: <not-my-user>
  password: <not-a-password>
```

Skąd brać wartości?



Wersja beta

secret.yml.gpg

```
{
       #Yc ã°8 ÿÒa É¨J@}o÷ ®±ÄMoHõ Mjßõjÿ
¿5· ,h iV®>  g  ¹ wÛ¯ Û ? lËè èë ·f'[é
Ðè§$Ü{ 0é ç[äp  úg°°×®K¾ ¾ ãb`Ë¦ö
```

Wersja beta

secret.yml.gpg

```
{
       #Yc ã°8 ÿÒa É¨J@}o÷ ®±ÄMoHõ Mjßõjÿ
¿5· ,h iV®>  g  ¹ wÛ¯ Û ? lËè èë ·f'[é
Đè§$Ü{ 0é ç[äp  úg°°×®K¾ ¾ ãb`Ë¦ö
```

encrypt.sh

```
gpg --symmetric --cipher-algo AES256 --batch
--passphrase=very-secret-passphrase secret.yml
```

Wersja beta

secret.yml.gpg

```
{
       #Yc ã°8 ÿÒa É¨J@}o÷ ®±ÄMoHõ Mjßõjÿ
¿5· ,h iV®>  g  ¹ wÛ¯ Û ? lËè èë ·f'[é
Ðè§$Ü{ 0é ç[äp  úg°°×®K¾ ¾ ãb`Ë¦ö
```

encrypt.sh

```
gpg --symmetric --cipher-algo AES256 --batch
--passphrase=very-secret-passphrase secret.yml
```

decrypt.sh

```
gpg --batch --yes --decrypt --passphrase=very-secret-passphrase
--output secret.yml secret.yml.gpg
```

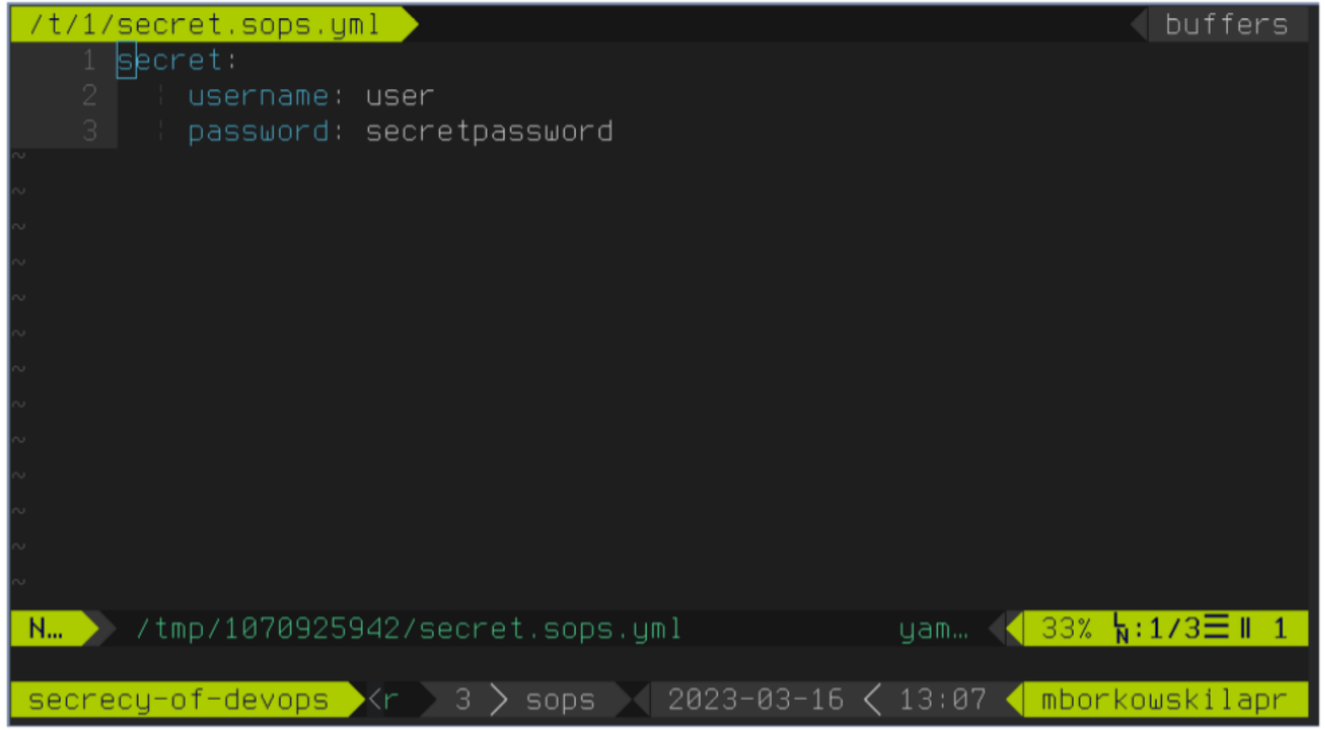
Wersja Release Candidate

decrypt.sh

```
gpg --batch --yes --decrypt --passphrase="$PASSPHRASE"
--output secret.yml secret.yml.gpg
```

Można wygodniej

```
export SOPS_PGP_FP="85D77543B3D624B63CEA9E6DBC17301B491B3F21,E60892BB9BD89A69F759A1A0A3
sops secret.sops.yml
```



Można wygodniej

encrypt.sh

```
export SOPS_PGP_FP="85D77543B3D624B63CEA9E6DBC17301B491B3F21,E60892BB9BD89A69F759A1A0A3
sops --encrypt --output secret.sops.yml secret.yml
```

decrypt.sh

```
sops --decrypt --output secret.yml secret.sops.yml
```

SOPS

secret.sops.yml

```
secret:
    username: ENC[AES256_GCM,data:4n63PA==,iv:7SuAX7DpWZ2JOA53N3YW6W1t5p/n+Xt/uHpil
    password: ENC[AES256_GCM,data:1txQttm652mqfKckRwU=,iv:SiirAhTIbJteeTYjIsnWCyE3C
sops:
    kms: []
    gcp_kms: []
    azure_kv: []
    hc_vault: []
    age: []
    lastmodified: "2022-12-22T07:17:45Z"
    mac: ENC[AES256_GCM,data:LiOeyDvh3LUzRd5YKiiJeGj+YrVj1F/XQHRsgYkJrqOeSzA5Vb2607
    pgp:
        - created_at: "2022-12-22T07:17:45Z"
          enc: |
            -----BEGIN PGP MESSAGE-----
```

SOPS

secret.sops.yml

```
secret:
    username: ENC[AES256_GCM,data:4n63PA==,iv:7SuAX7DpWZ2JOA53N3YW6W1t5p/n+Xt/uHpi
    password: ENC[AES256_GCM,data:1txQttm652mqfKckRwU=,iv:SiirAhTIbJteeTYjIsnWCyE3
sops:
    kms: []
    gcp_kms: []
    azure_kv: []
    hc_vault: []
    age: []
    lastmodified: "2022-12-22T07:17:45Z"
    mac: ENC[AES256_GCM,data:LiOeyDvh3LUzRd5YKiiJeGj+YrVj1F/XQHRsgYkJrqOeSzA5Vb260
    pgp:
        - created_at: "2022-12-22T07:17:45Z"
          enc: |
            -----BEGIN PGP MESSAGE-----
```

Jak to automatyzować?

Powrót do wersji alpha

.github/workflows/build.yml

```yaml
name: build application
on:
  workflow_dispatch:

jobs:
  run:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - run: ./decrypt.sh
        env:
          PASSPHRASE: very-secret-passphrase
      - uses: actions/upload-artifact@v3
        with:
          name: python-app
          path: .
```

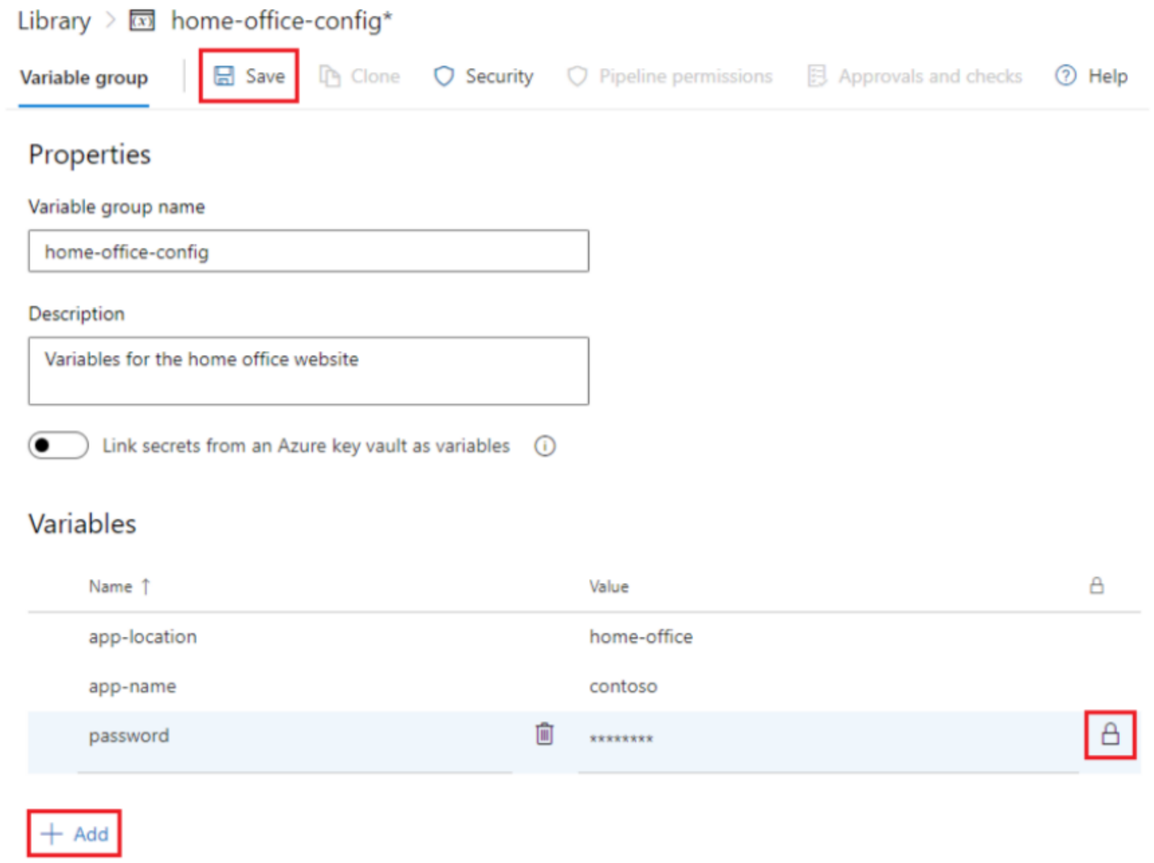
CI umie utrzymać sekret

```yaml
name: build application
on:
  workflow_dispatch:

jobs:
  run:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - run: |
        echo "
        secret:
          username: ${{ secrets.USERNAME }}
          password: ${{ secrets.PASSWORD }}
        " > secret.yml
      - uses: actions/upload-artifact@v3
        with:
          name: python-app
          path: .
```

Można też połączyć najlepsze rozwiązania

```yaml
name: build application
on:
  workflow_dispatch:

jobs:
  run:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - run: ./decrypt.sh
        env:
          PASSPHRASE: ${{ secrets.ENCRYPTION_PASSPHRASE }}
      - uses: actions/upload-artifact@v3
        with:
          name: python-app
          path: .
```

Czy sekrety są bezpieczne?



Czy sekrety są bezpieczne?

yml Copy

```yaml
steps:
- powershell: |
      Write-Host "My first secret variable is $env:FOO_ONE"
      $env:FOO_ONE -eq "foo"
  env:
    FOO_ONE: $(SecretOne)
- bash: |
    echo "My second secret variable: $FOO_TWO"
    if [ "$FOO_TWO" = "bar" ]; then
        echo "Strings are equal."
    else
        echo "Strings are not equal."
    fi
  env:
    FOO_TWO: $(SecretTwo)
```

The pipeline outputs:

Copy

```
My first secret variable is ***
True
My second secret variable: ***
Strings are equal.
```

Czy sekrety są bezpieczne?

workflow

```yaml
jobs:
  run:
    runs-on: ubuntu-latest
    steps:
      - run: echo ${{ secrets.PASSWORD }}
```

output

```
***
```

Czy sekrety są bezpieczne?

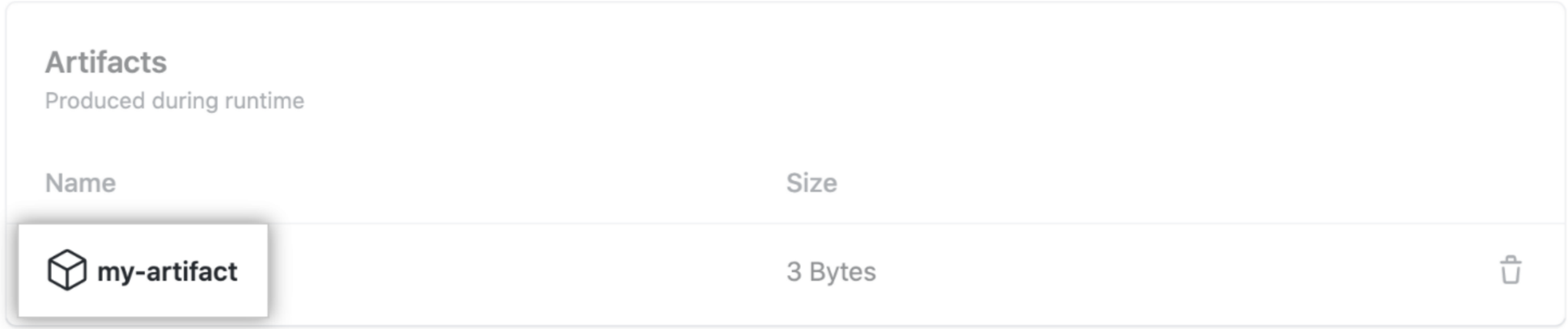
workflow

```yaml
jobs:
  run:
    runs-on: ubuntu-latest
    steps:
      - run: echo ${{ secrets.PASSWORD }} | base64 | base64
```

output

```
YzJWamNtVjBjR0Z6YzNkdmNtUUsK
```

Czy sekrety są bezpieczne?



Są jeszcze inne sekrety w buildach

```
1  FROM ubuntu:22.04
2
3  WORKDIR /app
4
5  ARG NEXUS_USER
6  ARG NEXUS_PASS
7
8  RUN pip install --extra-index-url \
9    "${NEXUS_USER}:${NEXUS_PASS}@https://nexus.company.com/repository" \
10   -r requirements.txt
```

```
docker build --build-arg NEXUS_USER=${USER} --build-arg NEXUS_PASS=${SECRET_PASSWORD} -
```

Są jeszcze inne sekrety w buildach

```dockerfile
FROM ubuntu:22.04

WORKDIR /app

ARG NEXUS_USER
ARG NEXUS_PASS

RUN pip install --extra-index-url \
  "${NEXUS_USER}:${NEXUS_PASS}@https://nexus.company.com/repository" \
  -r requirements.txt
```

```
docker build --build-arg NEXUS_USER=${USER} --build-arg NEXUS_PASS=${SECRET_PASSWORD} -
```

Są jeszcze inne sekrety w buildach

```
1  FROM ubuntu:22.04
2
3  WORKDIR /app
4
5  ARG NEXUS_USER
6  ARG NEXUS_PASS
7
8  RUN pip install --extra-index-url \
9    "${NEXUS_USER}:${NEXUS_PASS}@https://nexus.company.com/repository" \
10   -r requirements.txt
```

```
docker build --build-arg NEXUS_USER=${USER} --build-arg NEXUS_PASS=${SECRET_PASSWORD} -
```

Moment... Co?

```
~ » docker history secure-hello-pass

IMAGE          CREATED         CREATED BY                                      SIZE
c30b9a8c9c63   6 months ago    |2 NEXUS_PASS=pass NEXUS_USER=user /bin/sh -…   93B
b8e56c886555   6 months ago    /bin/sh -c #(nop)  ARG NEXUS_PASS               0B
...
```

```
sha256:c30b9a8c9c639ddea85f1574f3a323601431d4e20d471ac7413927a440c0e15c
6 months ago
|2 NEXUS_PASS=pass NEXUS_USER=user /bin/sh -c pip install
--extra-index-url "https://${NEXUS_USER}:${NEXUS_PASS}@nexus.company.com/repository"
93B
```

Moment... Co?

```
~ » docker history secure-hello-pass

IMAGE          CREATED         CREATED BY                                      SIZE
c30b9a8c9c63   6 months ago    |2 NEXUS_PASS=pass NEXUS_USER=user /bin/sh -…   93B
b8e56c886555   6 months ago    /bin/sh -c #(nop)  ARG NEXUS_PASS               0B
...
```

```
sha256:c30b9a8c9c639ddea85f1574f3a323601431d4e20d471ac7413927a440c0e15c
6 months ago
|2 NEXUS_PASS=pass NEXUS_USER=user /bin/sh -c pip install
--extra-index-url "https://${NEXUS_USER}:${NEXUS_PASS}@nexus.company.com/repository"
93B
```

Moment... Co?

```
~ » docker history secure-hello-pass

IMAGE          CREATED         CREATED BY                                      SIZE
c30b9a8c9c63   6 months ago    |2 NEXUS_PASS=pass NEXUS_USER=user /bin/sh -…   93B
b8e56c886555   6 months ago    /bin/sh -c #(nop)  ARG NEXUS_PASS               0B
...
```

```
sha256:c30b9a8c9c639ddea85f1574f3a323601431d4e20d471ac7413927a440c0e15c
6 months ago
|2 NEXUS_PASS=pass NEXUS_USER=user /bin/sh -c pip install
--extra-index-url "https://${NEXUS_USER}:${NEXUS_PASS}@nexus.company.com/repository"
93B
```

To też można zrobić dobrze

```
1 FROM ubuntu:22.04
2
3 WORKDIR /app
4
5 RUN --mount=type=secret,id=netrc,uid=0 NETRC=/run/secrets/netrc \
6   pip install --extra-index-url "https://nexus.company.com/repository" \
7   -r requirements.txt
```

```
export DOCKER_BUILDKIT=1
export CUSTOM_NETRC="machine host
login user
password pass
"
docker build --secret id=netrc,env=CUSTOM_NETRC -t secure-pass .
```

To też można zrobić dobrze

```
1 FROM ubuntu:22.04
2
3 WORKDIR /app
4
5 RUN --mount=type=secret,id=netrc,uid=0 NETRC=/run/secrets/netrc \
6   pip install --extra-index-url "https://nexus.company.com/repository" \
7   -r requirements.txt
```

```
export DOCKER_BUILDKIT=1
export CUSTOM_NETRC="machine host
login user
password pass
"
docker build --secret id=netrc,env=CUSTOM_NETRC -t secure-pass .
```

Niech inni martwią się
o moje sekrety

AWS Secrets Manager

```yaml
  AWS_ROLE: arn:aws:iam::000000000000:role/secret-role

jobs:
  run:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: AWS Credentials
        uses: aws-actions/configure-aws-credentials@v1-node16
        with:
          aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
          aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
          role-to-assume: ${{ env.AWS_ROLE }}
          aws-region: eu-west-1
      - name: get credentials
        uses: aws-actions/aws-secretsmanager-get-secrets@v1
        with:
          secret-ids: |
            USERNAME, /secret/app/username
            PASSWORD, /secret/app/secret_password
       - run: |
        echo "
        secret:
```

AWS Secrets Manager

```yaml
      - uses: actions/checkout@v3
      - name: AWS Credentials
        uses: aws-actions/configure-aws-credentials@v1-node16
        with:
          aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
          aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
          role-to-assume: ${{ env.AWS_ROLE }}
          aws-region: eu-west-1
      - name: get credentials
        uses: aws-actions/aws-secretsmanager-get-secrets@v1
        with:
          secret-ids: |
            USERNAME, /secret/app/username
            PASSWORD, /secret/app/secret_password
       - run: |
        echo "
        secret:
          username: ${{ env.USERNAME }}
          password: ${{ env.PASSWORD }}
        " > secret.yml
       - uses: actions/upload-artifact@v3
        with:
          name: python-app
```

AWS Secrets Manager

```
      - name: AWS Credentials
        uses: aws-actions/configure-aws-credentials@v1-node16
        with:
          aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
          aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
          role-to-assume: ${{ env.AWS_ROLE }}
          aws-region: eu-west-1
      - name: get credentials
        uses: aws-actions/aws-secretsmanager-get-secrets@v1
        with:
          secret-ids: |
            USERNAME, /secret/app/username
            PASSWORD, /secret/app/secret_password
      - run: |
        echo "
        secret:
          username: ${{ env.USERNAME }}
          password: ${{ env.PASSWORD }}
        " > secret.yml
      - uses: actions/upload-artifact@v3
        with:
          name: python-app
          path: .
```

Vault

```yaml
jobs:
  run:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Import Secrets
        id: secrets
        uses: hashicorp/vault-action@v2
        with:
          url: https://vault.mycompany.com:8200
          token: ${{ secrets.VAULT_TOKEN }}
          caCertificate: ${{ secrets.VAULT_CA_CERT }}
          secrets: |
              secret/app/credentials username | USERNAME ;
              secret/app/credentials secretPassword | PASSWORD ;
      - run: |
        echo "
        secret:
          username: ${{ steps.secrets.outputs.USERNAME }}
          password: ${{ steps.secrets.outputs.PASSWORD }}
        " > secret.yml
      - uses: actions/upload-artifact@v3
```

Vault

```yaml
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Import Secrets
        id: secrets
        uses: hashicorp/vault-action@v2
        with:
          url: https://vault.mycompany.com:8200
          token: ${{ secrets.VAULT_TOKEN }}
          caCertificate: ${{ secrets.VAULT_CA_CERT }}
          secrets: |
              secret/app/credentials username | USERNAME ;
              secret/app/credentials secretPassword | PASSWORD ;
      - run: |
        echo "
        secret:
          username: ${{ steps.secrets.outputs.USERNAME }}
          password: ${{ steps.secrets.outputs.PASSWORD }}
        " > secret.yml
      - uses: actions/upload-artifact@v3
        with:
          name: python-app
          path: .
```

A z kodu?

```python
import yaml

def main():
    with open('secret.yml', 'r') as secret:
        secrets = yaml.load(secret, yaml.Loader)

        message = f'''
        Login credentials:
        username: {secrets['secret']['username']}
        password: {secrets['secret']['password']}
        '''
        print(message)

if __name__ == '__main__':
    main()
```

A z kodu?

```python
import yaml

def main():
    with open('secret.yml', 'r') as secret:
        secrets = yaml.load(secret, yaml.Loader)

        message = f'''
        Login credentials:
        username: {secrets['secret']['username']}
        password: {secrets['secret']['password']}
        '''
        print(message)

if __name__ == '__main__':
    main()
```

A z kodu?

```python
import boto3

def main():
    # ignore credentials - provide config in runtime
    client = boto3.client('secretsmanager')

    username = client.get_secret_value(
            SecretId='/secret/app/username')['SecretString']
    password = client.get_secret_value(
            SecretId='/secret/app/secret_password')['SecretString']

    message = f'''
    Login credentials:
    username: {username}
    password: {password}
    '''
    print(message)

if __name__ == '__main__':
    main()
```

A z kodu?

```python
import boto3

def main():
    # ignore credentials - provide config in runtime
    client = boto3.client('secretsmanager')

    username = client.get_secret_value(
            SecretId='/secret/app/username')['SecretString']
    password = client.get_secret_value(
            SecretId='/secret/app/secret_password')['SecretString']

    message = f'''
    Login credentials:
    username: {username}
    password: {password}
    '''
    print(message)

if __name__ == '__main__':
    main()
```

From code?

```python
import os
from azure.keyvault.secrets import SecretClient
from azure.identity import DefaultAzureCredential

def main():
    keyVaultName = os.environ["KEY_VAULT_NAME"]
    KVUri = f"https://{keyVaultName}.vault.azure.net"

    credential = DefaultAzureCredential()
    client = SecretClient(vault_url=KVUri, credential=credential)

    username = client.get_secret('/secret/app/username').value
    password = client.get_secret('/secret/app/secret_password').value

    message = f'''
    Login credentials:
    username: {username}
    password: {password}
    '''
    print(message)

if __name__ == '__main__':
    main()
```

From code?

```python
import os
from azure.keyvault.secrets import SecretClient
from azure.identity import DefaultAzureCredential

def main():
    keyVaultName = os.environ["KEY_VAULT_NAME"]
    KVUri = f"https://{keyVaultName}.vault.azure.net"

    credential = DefaultAzureCredential()
    client = SecretClient(vault_url=KVUri, credential=credential)

    username = client.get_secret('/secret/app/username').value
    password = client.get_secret('/secret/app/secret_password').value

    message = f'''
    Login credentials:
    username: {username}
    password: {password}
    '''
    print(message)

if __name__ == '__main__':
    main()
```

A z kodu?

```python
import os
import hvac

def main():
    # provide token with an ENV variable
    vault_token = os.getenv('VAULT_TOKEN', default='dev-only-token')
    client = hvac.Client(
            url='https://vault.mycompany.com:8200',
            token=vault_token)

    secret_value = client.secrets.kv.read_secret_version(path='secret/app/credentials')

    username = secret_value['data']['data']['username']
    password = secret_value['data']['data']['secretPassword']

    message = f'''
    Login credentials:
    username: {username}
    password: {password}
    '''
    print(message)

if __name__ == '__main__':
```

A z kodu?

```python
import os
import hvac

def main():
    # provide token with an ENV variable
    vault_token = os.getenv('VAULT_TOKEN', default='dev-only-token')
    client = hvac.Client(
            url='https://vault.mycompany.com:8200',
            token=vault_token)

    secret_value = client.secrets.kv.read_secret_version(path='secret/app/credentials')

    username = secret_value['data']['data']['username']
    password = secret_value['data']['data']['secretPassword']

    message = f'''
    Login credentials:
    username: {username}
    password: {password}
    '''
    print(message)

if __name__ == '__main__':
```

Co dalej?

K8S

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: very-secret-python-app
  labels:
    name: secret-app
spec:
  containers:
  - name: python-app
    image: my-python-app
    env:
    - name: NOT_SO_SECRET
      value: visible-value
    - name: SECRET_API_KEY
      value: why-you-no-secret
```

K8S

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: very-secret-python-app
  labels:
    name: secret-app
spec:
  containers:
  - name: python-app
    image: my-python-app
    env:
    - name: NOT_SO_SECRET
      value: visible-value
    - name: SECRET_API_KEY
      value: why-you-no-secret
```

K8S

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: very-secret-python-app
  labels:
    name: secret-app
spec:
  volumes:
  - name: secret
    secret:
      secretName: python-app-secret
  containers:
  - name: python-app
    image: my-python-app
    volumeMounts:
    - name: secret
```

K8S

```
3  metadata:
4    name: very-secret-python-app
5    labels:
6      name: secret-app
7  spec:
8    volumes:
9    - name: secret
10     secret:
11       secretName: python-app-secret
12   containers:
13   - name: python-app
14     image: my-python-app
15     volumeMounts:
16     - name: secret
17       readOnly: true
```

K8S

```
10      secret:
11        secretName: python-app-secret
12    containers:
13    - name: python-app
14      image: my-python-app
15      volumeMounts:
16      - name: secret
17        readOnly: true
18        mountPath: "/app/secret.yml"
19      env:
20      - name: NOT_SO_SECRET
21        value: visible-value
22      - name: SECRET_API_KEY
23        valueFrom:
24          secretKeyRef:
```

K8S

```
12    containers:
13    - name: python-app
14      image: my-python-app
15      volumeMounts:
16      - name: secret
17        readOnly: true
18        mountPath: "/app/secret.yml"
19      env:
20      - name: NOT_SO_SECRET
21        value: visible-value
22      - name: SECRET_API_KEY
23        valueFrom:
24          secretKeyRef:
25            name: python-app-api-key
26            key: api-key
```

K8S enterprise

```yaml
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: vault-api-key
spec:
  provider: vault
  parameters:
    roleName: "csi"
    vaultAddress: "http://vault.vault:8200"
    objects:
      - secretPath: "secret/data/api_key"
        objectName: "api"
        secretKey: "api"
```

```yaml
volumes:
  - name: my-secret-api-key
    csi:
      driver: secrets-store.csi.k8s.io
      readOnly: true
      volumeAttributes:
        secretProviderClass: "vault-api-key"
```

K8S enterprise

```yaml
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: vault-api-key
spec:
  provider: vault
  parameters:
    roleName: "csi"
    vaultAddress: "http://vault.vault:8200"
    objects:
      - secretPath: "secret/data/api_key"
        objectName: "api"
        secretKey: "api"
```

```yaml
volumes:
  - name: my-secret-api-key
    csi:
      driver: secrets-store.csi.k8s.io
      readOnly: true
      volumeAttributes:
        secretProviderClass: "vault-api-key"
```

K8S enterprise

```yaml
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: vault-api-key
spec:
  provider: vault
  parameters:
    roleName: "csi"
    vaultAddress: "http://vault.vault:8200"
    objects:
      - secretPath: "secret/data/api_key"
        objectName: "api"
        secretKey: "api"
```

```yaml
volumes:
  - name: my-secret-api-key
    csi:
      driver: secrets-store.csi.k8s.io
      readOnly: true
      volumeAttributes:
        secretProviderClass: "vault-api-key"
```

K8S enterprise

```yaml
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: vault-api-key
spec:
  provider: vault
  parameters:
    roleName: "csi"
    vaultAddress: "http://vault.vault:8200"
    objects:
      - secretPath: "secret/data/api_key"
        objectName: "api"
        secretKey: "api"
```

```yaml
volumes:
  - name: my-secret-api-key
    csi:
      driver: secrets-store.csi.k8s.io
      readOnly: true
      volumeAttributes:
        secretProviderClass: "vault-api-key"
```

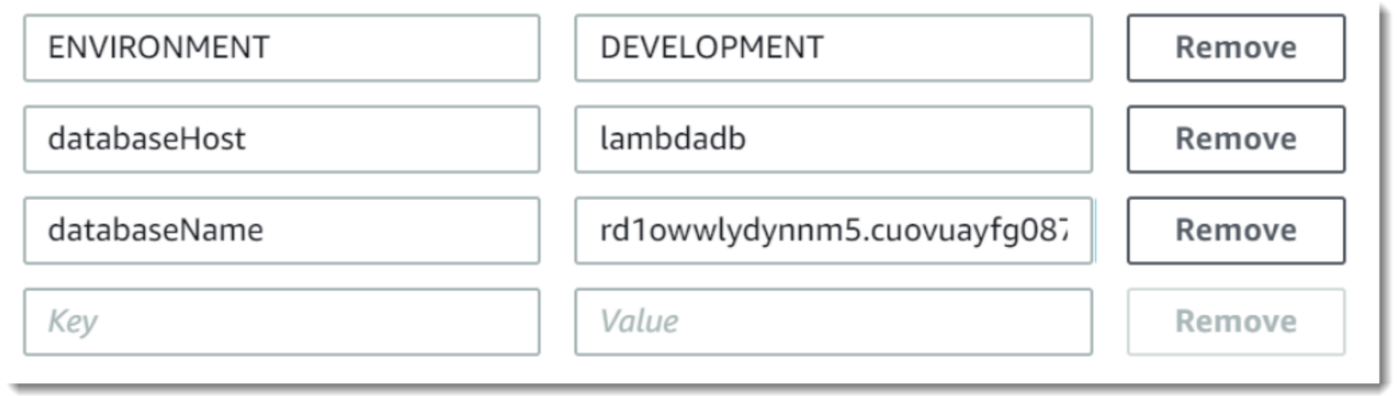
K8S enterprise

```yaml
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: vault-api-key
spec:
  provider: vault
  parameters:
    roleName: "csi"
    vaultAddress: "http://vault.vault:8200"
    objects:
      - secretPath: "secret/data/api_key"
        objectName: "api"
        secretKey: "api"
```

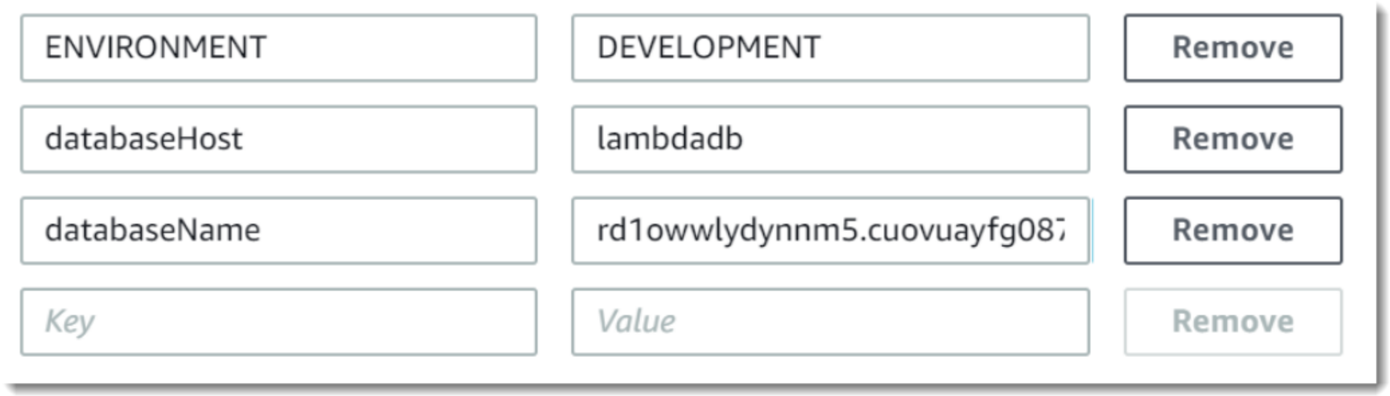
```yaml
volumes:
  - name: my-secret-api-key
    csi:
      driver: secrets-store.csi.k8s.io
      readOnly: true
      volumeAttributes:
        secretProviderClass: "vault-api-key"
```

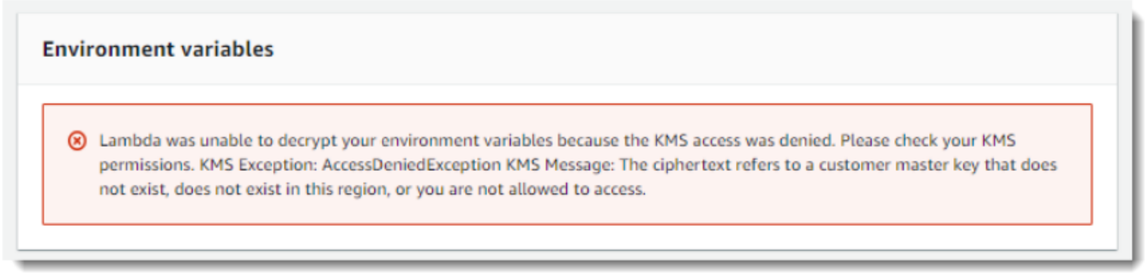
Lambda

Lambda


ENVIRONMENT
DEVELOPMENT
Remove
databaseHost
lambdadb
Remove
databaseName
rd1owwlydynnm5.cuovuayfg087
Remove
Key
Value
Remove

Lambda


ENVIRONMENT
DEVELOPMENT
Remove
databaseHost
lambdadb
Remove
databaseName
rd1owwlydynnm5.cuovuayfg087
Remove
Key
Value
Remove


Environment variables
Lambda was unable to decrypt your environment variables because the KMS access was denied. Please check your KMS permissions. KMS Exception: AccessDeniedException KMS Message: The ciphertext refers to a customer master key that does not exist, does not exist in this region, or you are not allowed to access.

Lambda

```
secrets_client = boto3.client('secretsmanager')
secret_arn = 'python_app_auth_token'
...

auth_token = secrets_client.get_secret_value(SecretId=secret_arn).get('SecretString')
```

Lambda

```
secrets_client = boto3.client('secretsmanager')
secret_arn = 'python_app_auth_token'
...

auth_token = secrets_client.get_secret_value(SecretId=secret_arn).get('SecretString')
```

```
aws lambda update-function-configuration \
    --function-name my-function \
    --layers arn:aws:lambda:eu-west-1:015030872274:layer:AWS-Parameters-and-Secrets-Lam
```

Lambda

```
secrets_client = boto3.client('secretsmanager')
secret_arn = 'python_app_auth_token'
...

auth_token = secrets_client.get_secret_value(SecretId=secret_arn).get('SecretString')
```

```
aws lambda update-function-configuration \
    --function-name my-function \
    --layers arn:aws:lambda:eu-west-1:015030872274:layer:AWS-Parameters-and-Secrets-Lam
```

```
from aws_lambda_powertools.utilities import parameters
...

auth_token = parameters.get_secret('python_app_auth_token')
```

Logi

Logi

Jeżeli sekrety przekazujemy jako zmienne środowiskowe...

Logi

Jeżeli sekrety przekazujemy jako zmienne środowiskowe...

...a każda biblioteka której używamy ma dostęp do całości środowiska procesu...

Logi

Jeżeli sekrety przekazujemy jako zmienne środowiskowe...

...a każda biblioteka której używamy ma dostęp do całości środowiska procesu...

...to co pojawi się w DEBUG logach?

Zaufanie

Wersja 1.9 edycja PRO

```yaml
name: build application
on:
  workflow_dispatch:

env:
  AWS_ROLE: arn:aws:iam::000000000000:role/secret-role

jobs:
  run:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: AWS Credentials
        uses: aws-actions/configure-aws-credentials@v1-node16
        with:
          aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
          aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
          role-to-assume: ${{ env.AWS_ROLE }}
          aws-region: eu-west-1
      - name: get credentials
        uses: aws-actions/aws-secretsmanager-get-secrets@v1
        with:
          secret-ids: |
```

Wersja 1.9 edycja PRO

```yaml
name: build application
on:
  workflow_dispatch:

env:
  AWS_ROLE: arn:aws:iam::000000000000:role/secret-role

jobs:
  run:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: AWS Credentials
        uses: aws-actions/configure-aws-credentials@v1-node16
        with:
          aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
          aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
          role-to-assume: ${{ env.AWS_ROLE }}
          aws-region: eu-west-1
      - name: get credentials
        uses: aws-actions/aws-secretsmanager-get-secrets@v1
        with:
          secret-ids: |
```

Wersja 1.9 edycja PRO

```yaml
  AWS_ROLE: arn:aws:iam::000000000000:role/secret-role

jobs:
  run:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: AWS Credentials
        uses: aws-actions/configure-aws-credentials@v1-node16
        with:
          aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
          aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
          role-to-assume: ${{ env.AWS_ROLE }}
          aws-region: eu-west-1
      - name: get credentials
        uses: aws-actions/aws-secretsmanager-get-secrets@v1
        with:
          secret-ids: |
            USERNAME, /secret/app/username
            PASSWORD, /secret/app/secret_password
       - run: |
        echo "
        secret:
```

Wersja 2.0 edycja PRO

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::000000000000:oidc-provider/token.actions.githubus
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringLike": {
                    "token.actions.githubusercontent.com:sub": [
                        "repo:my-org/secret-project:*",
                        "repo:my-org/even-more-secret-project:*",
                    ]
                },
                "StringEquals": {
                    "token.actions.githubusercontent.com:aud": "sts.amazonaws.com"
                }
            }
        }
    ]
```

Wersja 2.0 edycja PRO

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::000000000000:oidc-provider/token.actions.githubus
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringLike": {
                    "token.actions.githubusercontent.com:sub": [
                        "repo:my-org/secret-project:*",
                        "repo:my-org/even-more-secret-project:*",
                    ]
                },
                "StringEquals": {
                    "token.actions.githubusercontent.com:aud": "sts.amazonaws.com"
                }
            }
        }
    ]
```

Wersja 2.0 edycja PRO

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::000000000000:oidc-provider/token.actions.githubus
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringLike": {
                    "token.actions.githubusercontent.com:sub": [
                        "repo:my-org/secret-project:*",
                        "repo:my-org/even-more-secret-project:*",
                    ]
                },
                "StringEquals": {
                    "token.actions.githubusercontent.com:aud": "sts.amazonaws.com"
                }
            }
        }
    ]
}
```

Wersja 2.0 edycja PRO

```yaml
name: build application
on:
  workflow_dispatch:

env:
  AWS_ROLE: arn:aws:iam::000000000000:role/secret-role

permissions:
  id-token: write
  contents: read

jobs:
  run:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: AWS Credentials
        uses: aws-actions/configure-aws-credentials@v1-node16
        with:
          role-to-assume: ${{ env.AWS_ROLE }}
          aws-region: eu-west-1
      - name: get credentials
        uses: aws-actions/aws-secretsmanager-get-secrets@v1
```

Wersja 2.0 edycja PRO

```yaml
name: build application
on:
  workflow_dispatch:

env:
  AWS_ROLE: arn:aws:iam::000000000000:role/secret-role

permissions:
  id-token: write
  contents: read

jobs:
  run:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: AWS Credentials
        uses: aws-actions/configure-aws-credentials@v1-node16
        with:
          role-to-assume: ${{ env.AWS_ROLE }}
          aws-region: eu-west-1
      - name: get credentials
        uses: aws-actions/aws-secretsmanager-get-secrets@v1
```

Wersja 2.0 edycja PRO

```yaml
name: build application
on:
  workflow_dispatch:

env:
  AWS_ROLE: arn:aws:iam::000000000000:role/secret-role

permissions:
  id-token: write
  contents: read

jobs:
  run:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: AWS Credentials
        uses: aws-actions/configure-aws-credentials@v1-node16
        with:
          role-to-assume: ${{ env.AWS_ROLE }}
          aws-region: eu-west-1
      - name: get credentials
        uses: aws-actions/aws-secretsmanager-get-secrets@v1
```

Wersja 2.0 edycja PRO

```yaml
permissions:
  id-token: write
  contents: read

jobs:
  run:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: AWS Credentials
        uses: aws-actions/configure-aws-credentials@v1-node16
        with:
          role-to-assume: ${{ env.AWS_ROLE }}
          aws-region: eu-west-1
      - name: get credentials
        uses: aws-actions/aws-secretsmanager-get-secrets@v1
        with:
          secret-ids: |
            USERNAME, /secret/app/username
            PASSWORD, /secret/app/secret_password
       - run: |
        echo "
        secret:
```

Wersja 2.0 edycja PRO

```
14      runs-on: ubuntu-latest
15      steps:
16        - uses: actions/checkout@v3
17        - name: AWS Credentials
18          uses: aws-actions/configure-aws-credentials@v1-node16
19          with:
20            role-to-assume: ${{ env.AWS_ROLE }}
21            aws-region: eu-west-1
22        - name: get credentials
23          uses: aws-actions/aws-secretsmanager-get-secrets@v1
24          with:
25            secret-ids: |
26              USERNAME, /secret/app/username
27              PASSWORD, /secret/app/secret_password
28         - run: |
29          echo "
30          secret:
31            username: ${{ env.USERNAME }}
32            password: ${{ env.PASSWORD }}
33          " > secret.yml
34         - uses: actions/upload-artifact@v3
35          with:
36            name: python-app
```

Żadnych sekretów

Żadnych sekretów

Żadnych tajemnic

Żadnych sekretów

Żadnych tajemnic

Tylko **ZAUFANIE**

Materiały dla cierpiących na bezsenność

- https://github.com/mozilla/sops - Sops - narzędzie do edycji/zarządzania szyfrowanych sekretów
- https://docs.github.com/en/actions/security-guides/encrypted-secrets - instrukcja github na temat sekretów
  (Storing large secrets mówi o użyciu gpg)
- https://github.com/marketplace/actions/aws-secrets-manager-github-action - użyj AWS Secret w buildzie GH
- https://github.com/marketplace/actions/vault-secrets - użyj sekretu z Hashicorp Vault w buildzie GH
- https://pythonspeed.com/articles/docker-build-secrets/ - o używaniu sekretów w buildach dockera
- https://docs.github.com/en/actions/deployment/security-hardening-your-deployments/configuring-openid-connect-in-amazon-web-services - o usuwaniu credentiali do AWS z GH actions
- https://learn.microsoft.com/en-us/azure/developer/github/connect-from-azure?tabs=azure-cli%2Clinux - bez credentiali w Azure
- https://kubernetes.io/docs/concepts/configuration/secret/ - sekrety w K8S
- https://secrets-store-csi-driver.sigs.k8s.io/introduction.html - sekrety w K8S z secret managera
- https://aws.amazon.com/blogs/compute/securely-retrieving-secrets-with-aws-lambda/ - wydajne użycie sekretów w lambdach

I pamiętajcie

I pamiętajcie

BBC