NIC

EMPOWER

November 13-15, Oslo Spektrum

Karim El-Melhaoui, O3 Cyber

Azure Security Assessments: Lessons from the Field

NIC
EMPOWER

Quick intro

- Karim El-Melhaoui (karim@o3c.no)
- Principal at O3 Cyber (o3c.no)
- CloudSec Researcher
- Azure MVP but I love **most** clouds

This is a deep dive session

1. Introduction to assessment and Scoping
2. Discovery, Data Collection and Tooling
3. Writing our own Tools
4. Attack Paths
5. Research Release: Abusing Azure Data Factory
6. Research Preview: OIDC Attack Path Analyzer

Introduction to a Cloud Security Assessment

NIC
EMPOWER

What is an assessment

- Infrastructure and Architecture review
  - Access and Authentication assessment
  - Configuration Analysis
  - Exposure and Vulnerabilities

NIC
EMPOWER

Goal

- Uncover critical design flaws
- Identify Toxic Combinations (Attack Paths)
- Assess the efficacy of controls
- Find exposed vulnerabilities

Scoping

Scoping an assessment: Technical

- Service Coverage
  - Azure Resource Manager
  - Microsoft Graph
  - Azure Services
  - GitHub / Azure DevOps / CI tool
- Platform usage

NIC
EMPOWER

Scoping an assessment: Organizational

- Business Context

- Expectations

NIC
EMPOWER

What we need to know

- Third-parties we should expect to see
- Privileged Access procedures
- Segmentation policy
- Guardrails / Governance procedures
- Tooling

NIC
EMPOWER

Permissions

- Non-instrusive:
  - Global Reader
  - Reader on Tenant Root Group
  - Reader in GitHub
- Intrusive: Same as a ‘DevOps’ user

Discovery / Data Collection

Open-source tools

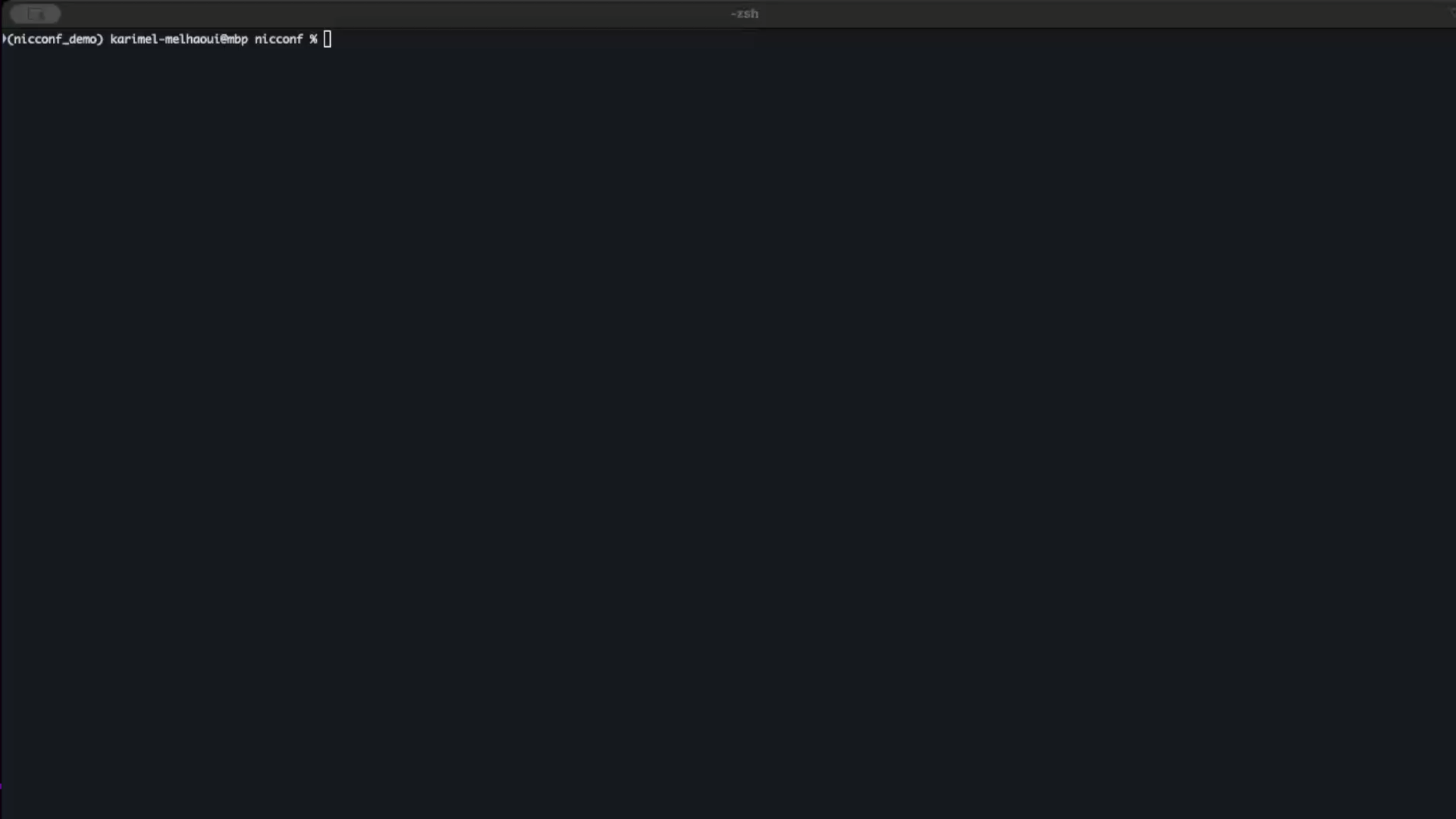
Tools

PROWLER

Prowler

- Multi-Cloud
- Built in Python with SDK’s
- Compliance Benchmarks ++

Compliance scan with Prowler


-zsh
(nicconf_demo) karimel-melhaoui@mbp nicconf %

Why do we complicate things beyond this?

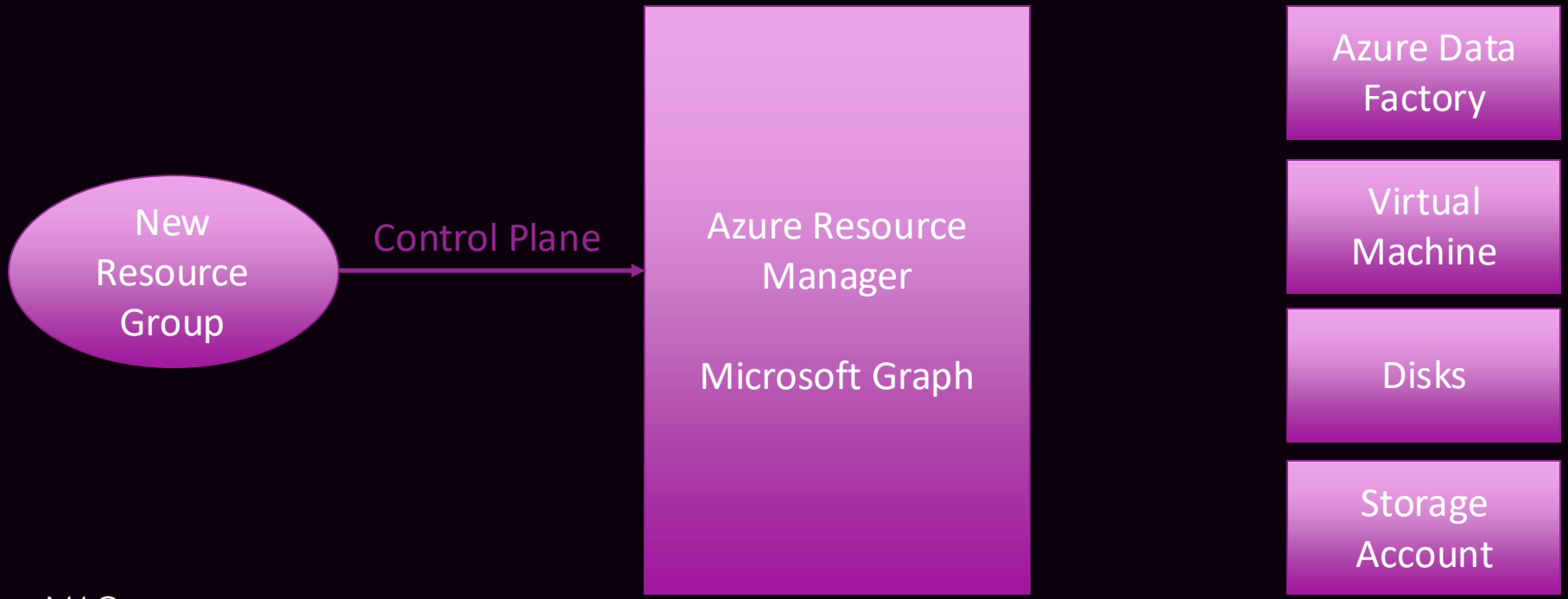
What the client **shouldn’t** care about

- Compliance for the sake of being compliant
- Unencrypted disks
- Isolated configuration findings

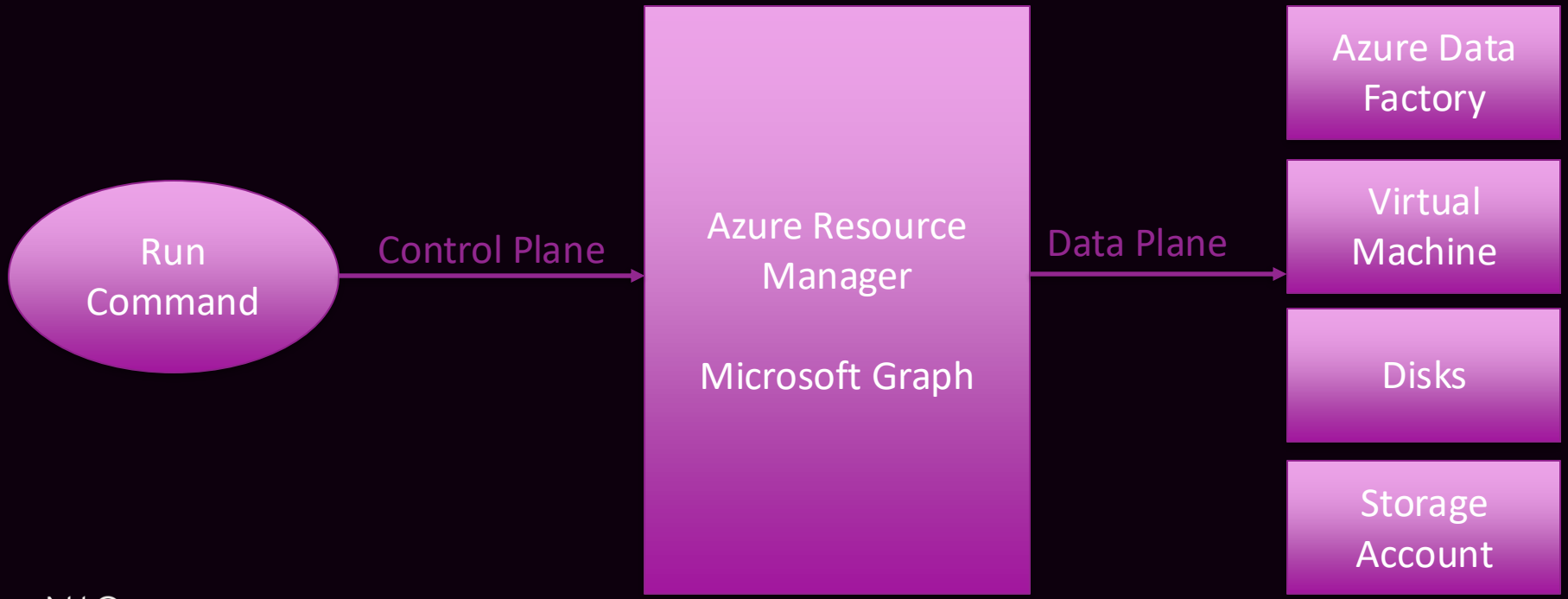
* For the record, I love Prowler and use it all the time!

Writing our own tools

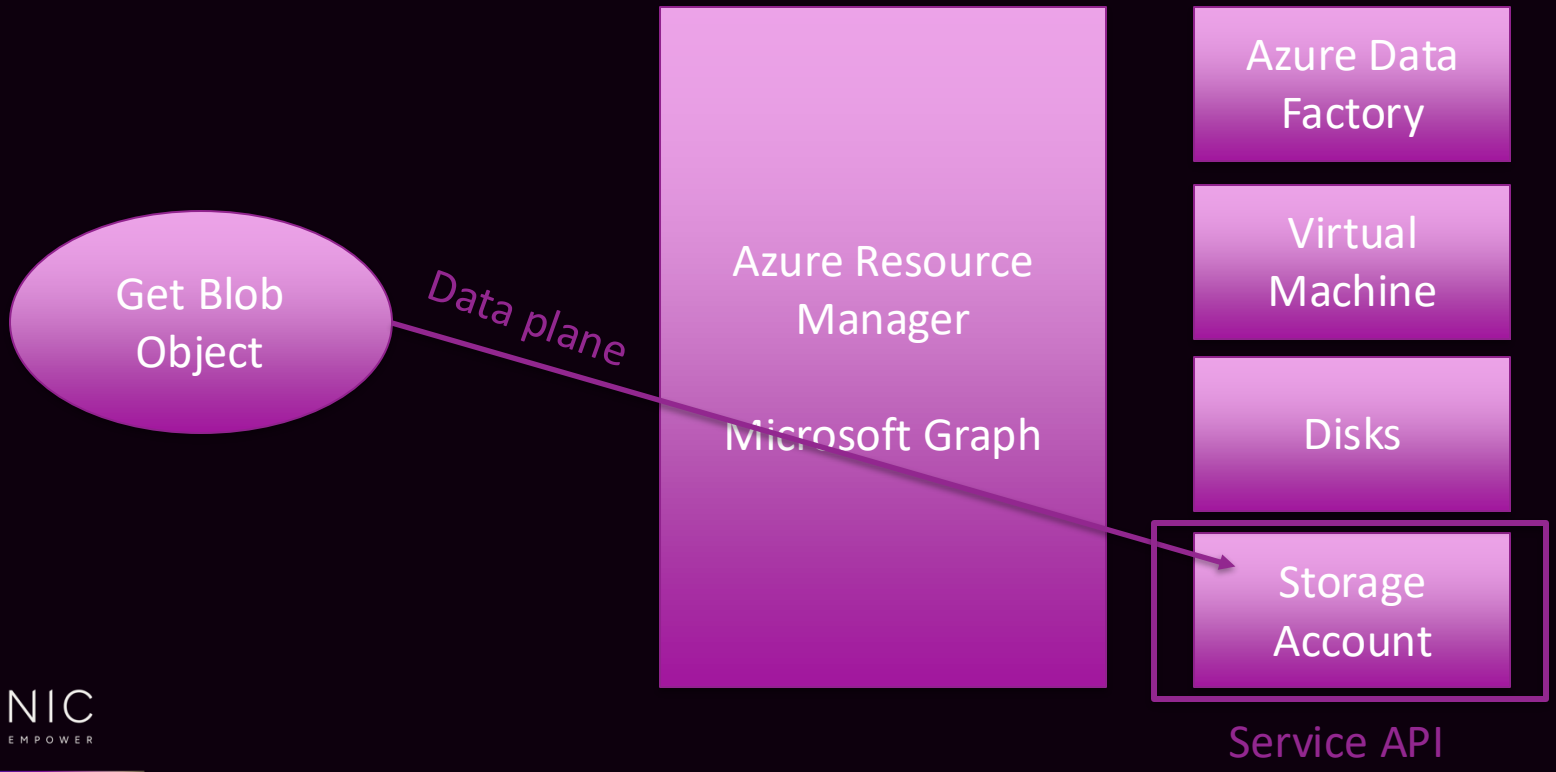
Control Plane vs. Data Plane



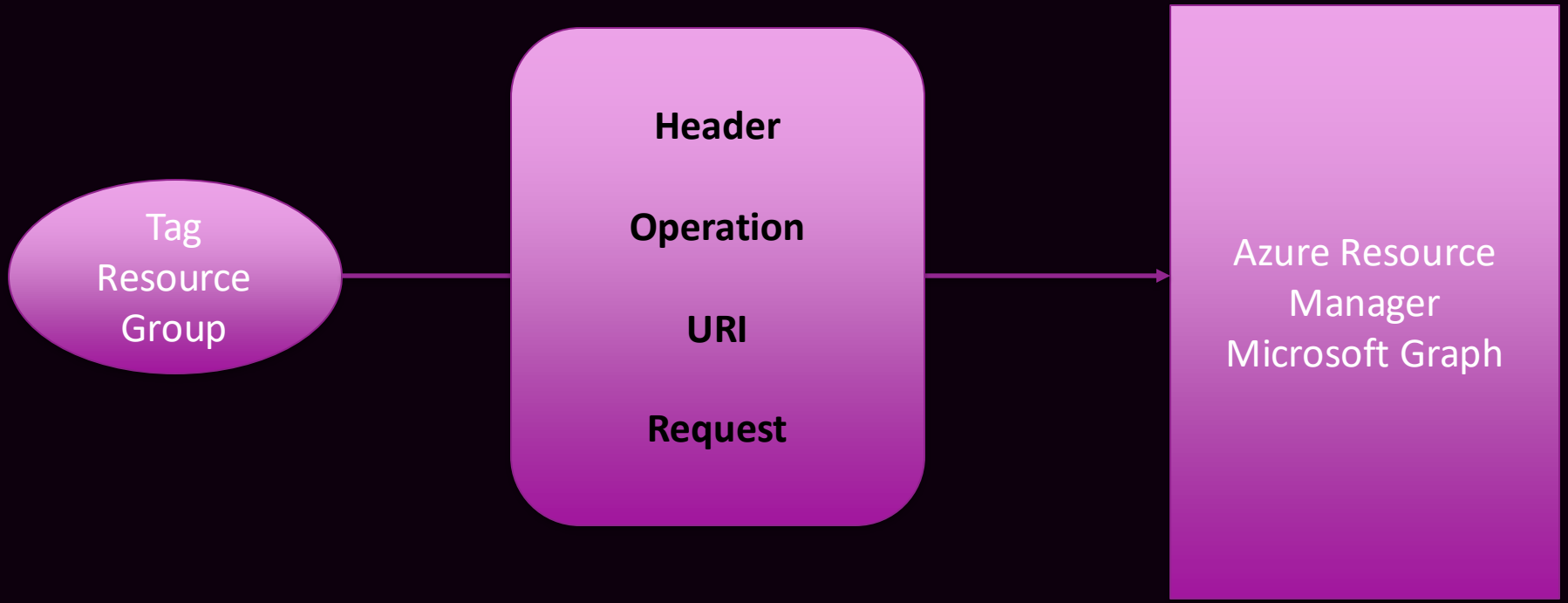
Control Plane vs. Data Plane



Control Plane vs. Data Plane



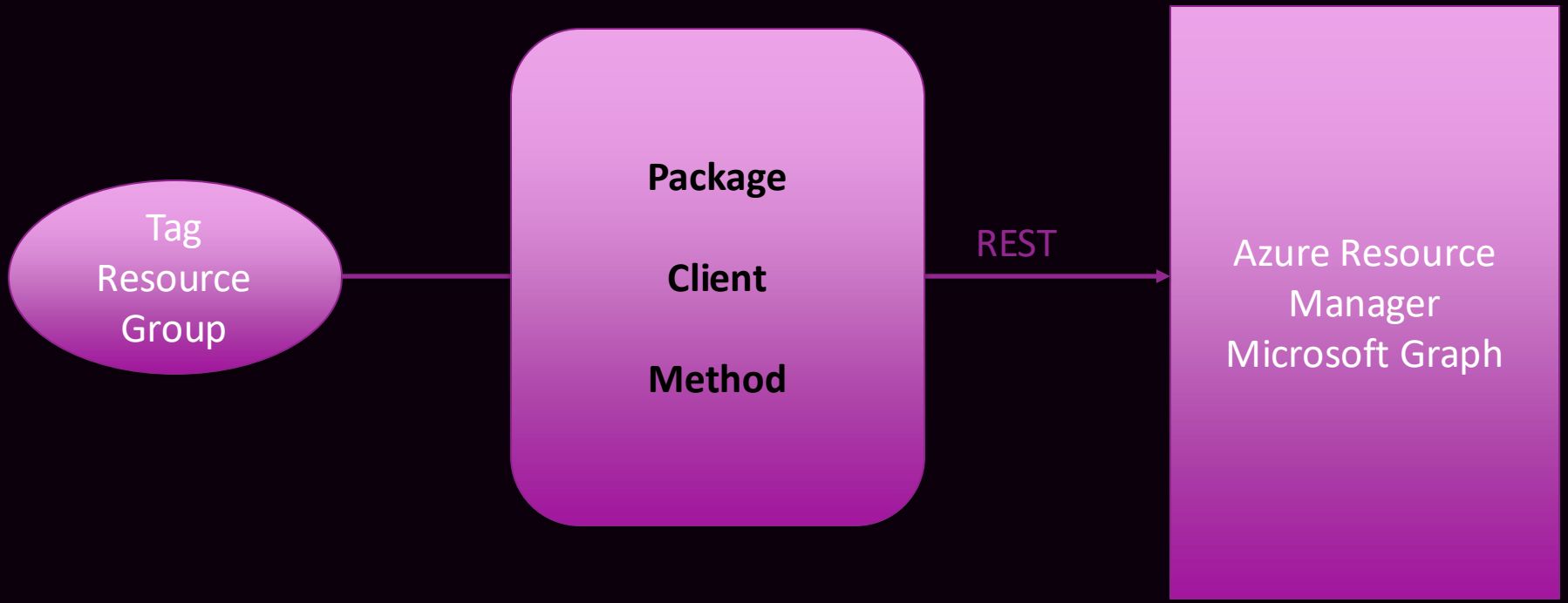
REST


Tag
Resource
Group
Header
Operation
URI
Request
Azure Resource
Manager
Microsoft Graph

REST

- Full flexibility and control
- Language independent
- Low-Level operations

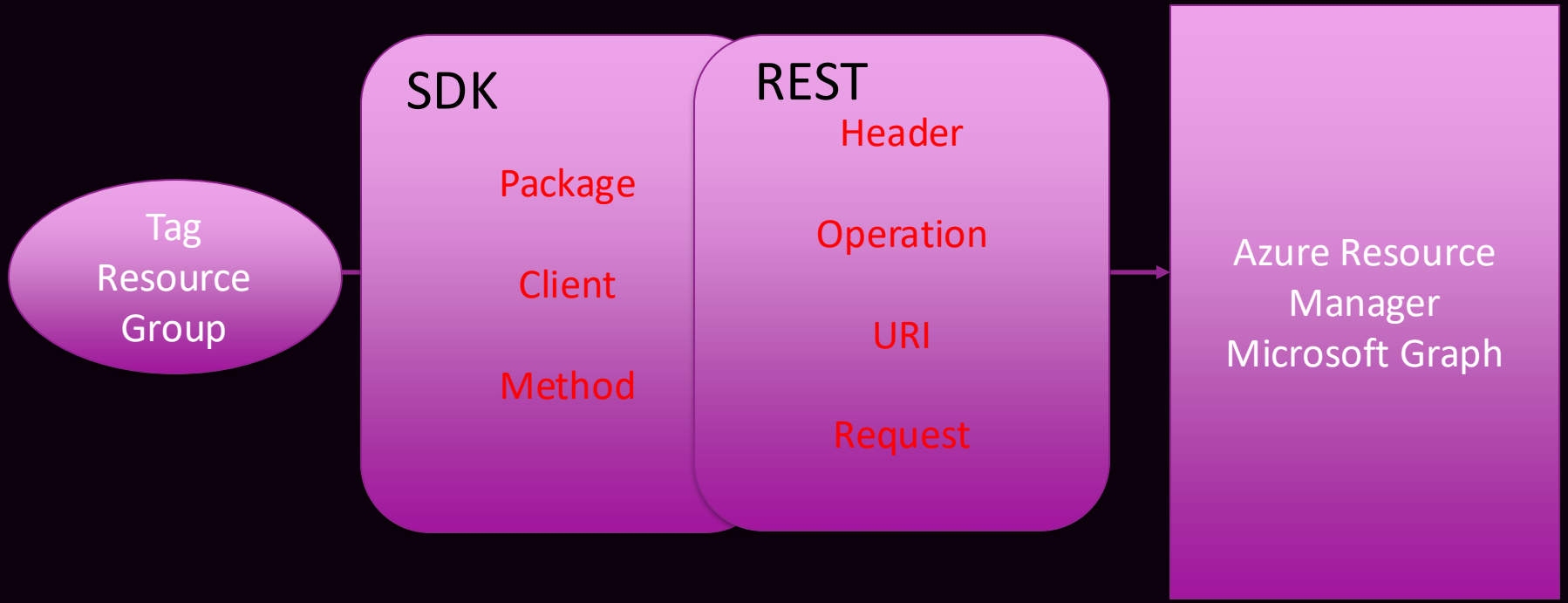
SDK


Tag
Resource
Group
Package
Client
Method
REST
Azure Resource
Manager
Microsoft Graph

SDK

- Built-in Authentication Handling
- Automatic retry and error handling
- Language specific

Best of Both Worlds



End Goal

MSAL + Python

Graph Token
ARM Token

Authentication

import msal

```python
class AuthClientBase:

    def __init__(self, client_id, client_credential, tenant_id, scope):
        self.app = msal.ConfidentialClientApplication(
            client_id=client_id,
            client_credential=client_credential,
            authority=f"https://login.microsoftonline.com/{tenant_id}",
        )
        self.scope = scope

    def get_token(self):
        result = self.app.acquire_token_silent(scopes=[self.scope], account=None)
        if not result:
            result = self.app.acquire_token_for_client(scopes=[self.scope])

        if "access_token" in result:
            return result["access_token"]
        else:
            raise Exception("Failed to obtain access token")
```

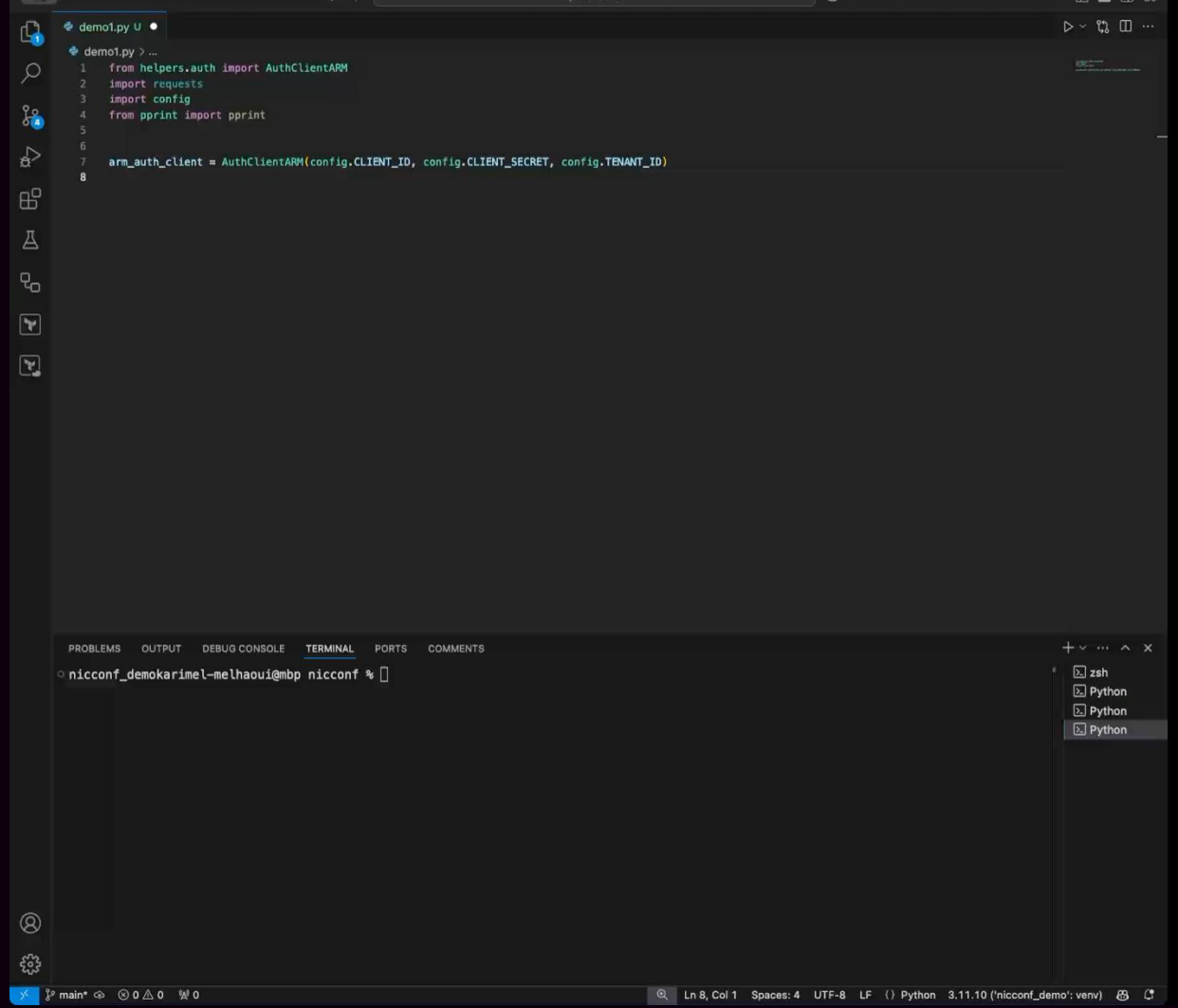
NIC
EMPOWER

Authentication

```python
class AuthClientGraph(AuthClientBase):
    def __init__(self, client_id, client_credential, tenant_id):
        super().__init__(
            client_id,
            client_credential,
            tenant_id,
            "https://graph.microsoft.com/.default",
        )


class AuthClientARM(AuthClientBase):
    def __init__(self, client_id, client_credential, tenant_id):
        super().__init__(
            client_id,
            client_credential,
            tenant_id,
            "https://management.azure.com/.default",
        )
```

NIC
EMPOWER


demo1.py U
demo1.py > ...
1 from helpers.auth import AuthClientARM
2 import requests
3 import config
4 from pprint import pprint
5
6
7 arm_auth_client = AuthClientARM(config.CLIENT_ID, config.CLIENT_SECRET, config.TENANT_ID)
8
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
COMMENTS
nicconf_demokarimel-melhaoui@mbp nicconf %
zsh
Python
Python
Python
main*
Ln 8, Col 1
Spaces: 4
UTF-8
LF
Python
3.11.10 ('nicconf_demo': venv)

NIC
EMPOWER

Resource Groups

```python
def get_resource_groups(arm_auth_client, subscription):
    """
    Fetch the list of resource groups for a specific subscription from the Azure Management API.

    Args:
        arm_auth_client: The authentication client to use for fetching the token.
        subscription (str): The subscription ID to fetch resource groups for.

    Returns:
        List[Dict]: A list of dictionaries containing the resource group details.
    """
    token = arm_auth_client.get_token()
    url = f"https://management.azure.com/subscriptions/{subscription}/resourceGroups?api-version=2020-01-01"
    headers = {"Authorization": f"Bearer {token}"}
    response = requests.get(url, headers=headers)
    if response.status_code == 200:
        return response.json().get("value", [])
    else:
        print(
            f"Error fetching resource groups: {response.status_code} - {response.text}"
        )
        return []
```

Resource Group Role Assignments

```python
def get_rg_role_assignment(arm_auth_client, subscription, resource_group):
    """
    Fetch the list of role assignments for a specific resource group from the Azure Management API.

    Args:
        arm_auth_client: The authentication client to use for fetching the token.
        subscription (str): The subscription ID to fetch role assignments for.
        resource_group (str): The resource group ID to fetch role assignments for.

    Returns:
        List[Dict]: A list of dictionaries containing the role assignment details.
    """
    token = arm_auth_client.get_token()
    url = f"https://management.azure.com{resource_group}/providers/Microsoft.Authorization/roleAssignments?api-version=2022-04-01"
    headers = {"Authorization": f"Bearer {token}"}
    response = requests.get(url, headers=headers)
    if response.status_code == 200:
        return response.json().get("value", [])
    else:
        print(
            f"Error fetching resource group role assignments: {response.status_code} - {response.text}"
        )
        return []
```

NIC
EMPOWER

Resources

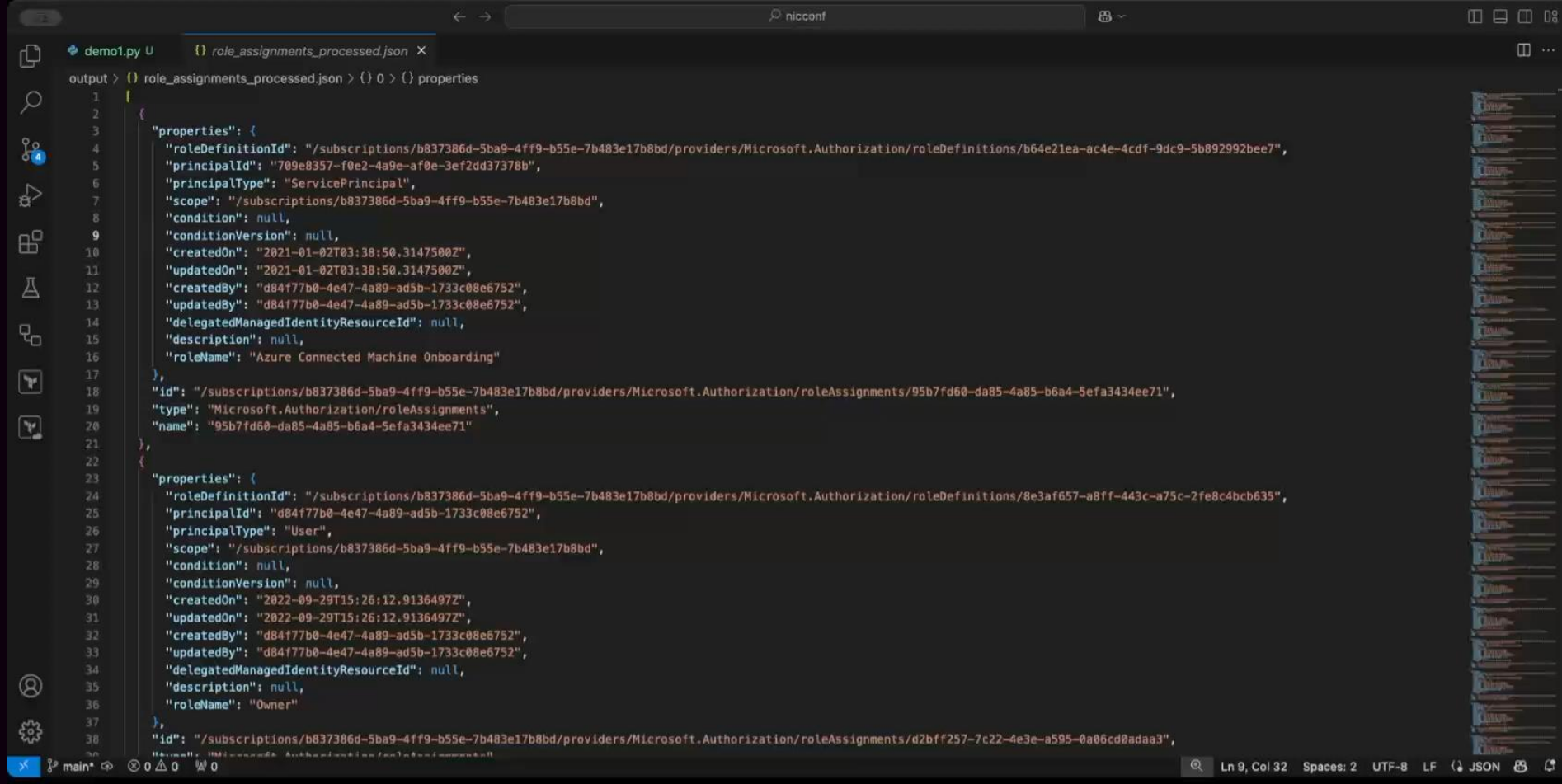
```python
def get_resources(arm_auth_client, subscription, resource_group):
    """
    Fetch the list of resources for a specific resource group from the Azure Management API.

    Args:
        arm_auth_client: The authentication client to use for fetching the token.
        subscription (str): The subscription ID to fetch resources for.
        resource_group (str): The resource group ID to fetch resources for.

    Returns:
        List[Dict]: A list of dictionaries containing the resource details.
    """
    token = arm_auth_client.get_token()
    url = f"https://management.azure.com{resource_group}/resources?api-version=2021-04-01"
    headers = {"Authorization": f"Bearer {token}"}
    response = requests.get(url, headers=headers)
    if response.status_code == 200:
        return response.json().get("value", [])
    else:
        print(f"Error fetching resources: {response.status_code} - {response.text}")
        return []
```

What if we want to get ALL DATA?

Live demo: Getting all the data




```
[
  {
    "properties": {
      "roleDefinitionId": "/subscriptions/b837386d-5ba9-4ff9-b55e-7b483e17b8bd/providers/Microsoft.Authorization/roleDefinitions/b64e21ea-ac4e-4cdf-9dc9-5b892992bee7",
      "principalId": "709e8357-f0e2-4a9e-af0e-3ef2dd37378b",
      "principalType": "ServicePrincipal",
      "scope": "/subscriptions/b837386d-5ba9-4ff9-b55e-7b483e17b8bd",
      "condition": null,
      "conditionVersion": null,
      "createdOn": "2021-01-02T03:38:50.3147500Z",
      "updatedOn": "2021-01-02T03:38:50.3147500Z",
      "createdBy": "d84f77b0-4e47-4a89-ad5b-1733c08e6752",
      "updatedBy": "d84f77b0-4e47-4a89-ad5b-1733c08e6752",
      "delegatedManagedIdentityResourceId": null,
      "description": null,
      "roleName": "Azure Connected Machine Onboarding"
    },
    "id": "/subscriptions/b837386d-5ba9-4ff9-b55e-7b483e17b8bd/providers/Microsoft.Authorization/roleAssignments/95b7fd60-da85-4a85-b6a4-5efa3434ee71",
    "type": "Microsoft.Authorization/roleAssignments",
    "name": "95b7fd60-da85-4a85-b6a4-5efa3434ee71"
  },
  {
    "properties": {
      "roleDefinitionId": "/subscriptions/b837386d-5ba9-4ff9-b55e-7b483e17b8bd/providers/Microsoft.Authorization/roleDefinitions/8e3af657-a8ff-443c-a75c-2fe8c4bcb635",
      "principalId": "d84f77b0-4e47-4a89-ad5b-1733c08e6752",
      "principalType": "User",
      "scope": "/subscriptions/b837386d-5ba9-4ff9-b55e-7b483e17b8bd",
      "condition": null,
      "conditionVersion": null,
      "createdOn": "2022-09-29T15:26:12.9136497Z",
      "updatedOn": "2022-09-29T15:26:12.9136497Z",
      "createdBy": "d84f77b0-4e47-4a89-ad5b-1733c08e6752",
      "updatedBy": "d84f77b0-4e47-4a89-ad5b-1733c08e6752",
      "delegatedManagedIdentityResourceId": null,
      "description": null,
      "roleName": "Owner"
    },
    "id": "/subscriptions/b837386d-5ba9-4ff9-b55e-7b483e17b8bd/providers/Microsoft.Authorization/roleAssignments/d2bff257-7c22-4e3e-a595-0a06cd0adaa3",
```

I could have just used PowerShell or Az cli?

… No. Because

- Full control of what I want to request
- Returns the full object
- Extensible
- Interact outside of what Microsoft wants you to.

Classic Admins in SDK? Nope.

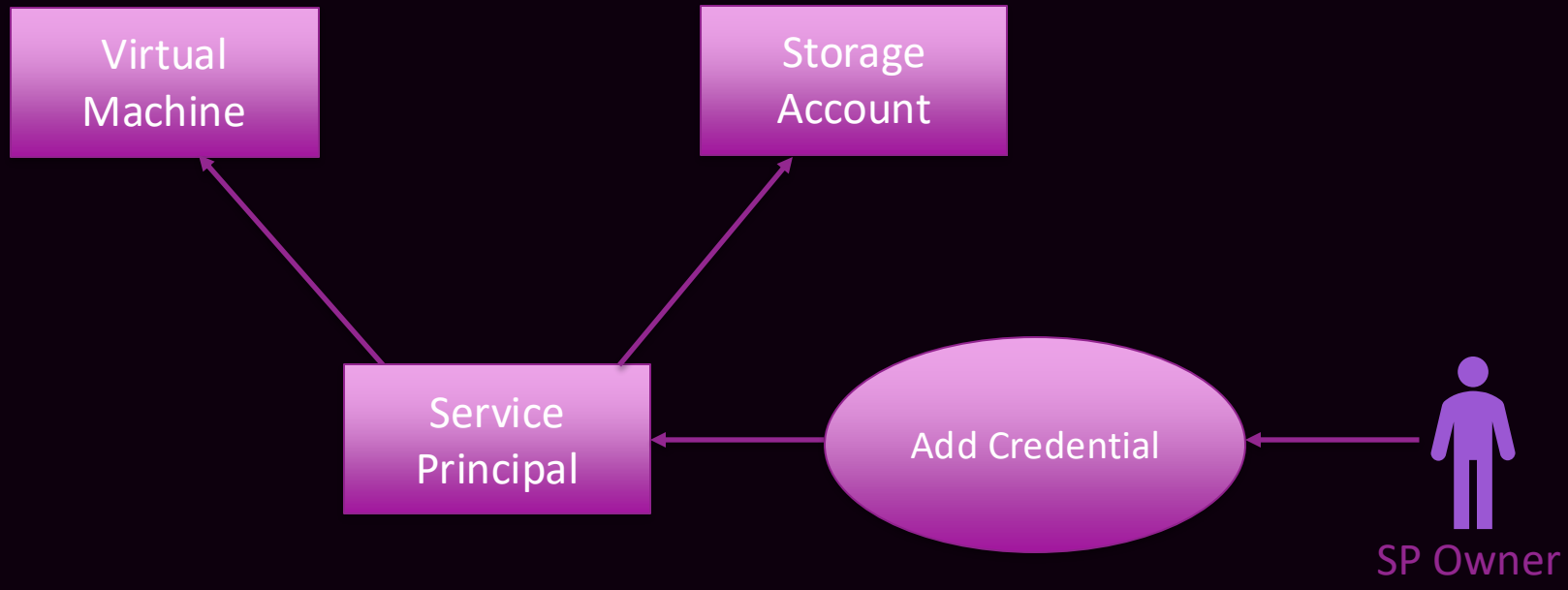
```python
def get_classic_admins(arm_auth_client, subscription):
    token = arm_auth_client.get_token()
    url = f"https://management.azure.com{subscription}/providers/Microsoft.Authorization/classicAdministrators?api-version=2015-07-01"
    headers = {"Authorization": f"Bearer {token}"}
    response = requests.get(url, headers=headers)
    if response.status_code == 200:
        return response.json().get("value", [])
    else:
        print(
            f"Error fetching subscription role assignments: {response.status_code} - {response.text}"
        )
        return []
```

NIC
EMPOWER

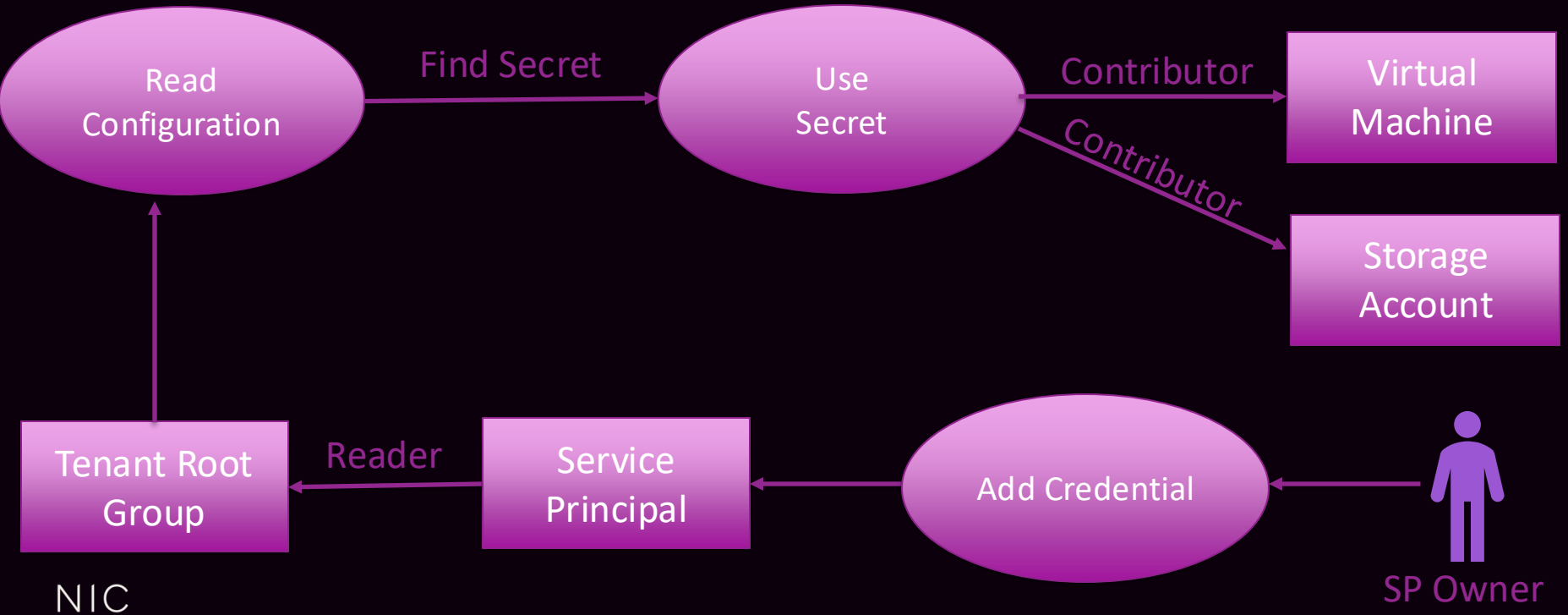
Let’s dive deeper.

Attack Path Introduction

Simple Attack Path



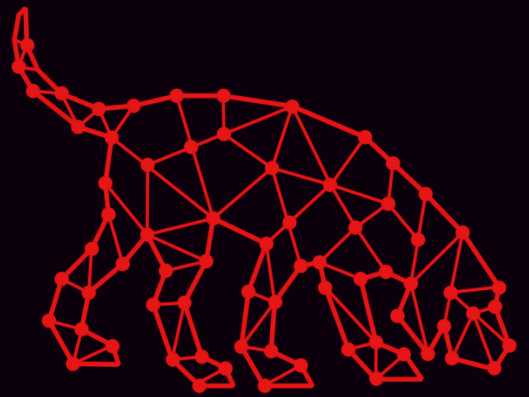
Attack Path


Read Configuration
Find Secret
Use Secret
Contributor
Virtual Machine
Contributor
Storage Account
Tenant Root Group
Reader
Service Principal
Add Credential
SP Owner

What causes an Attack Path?

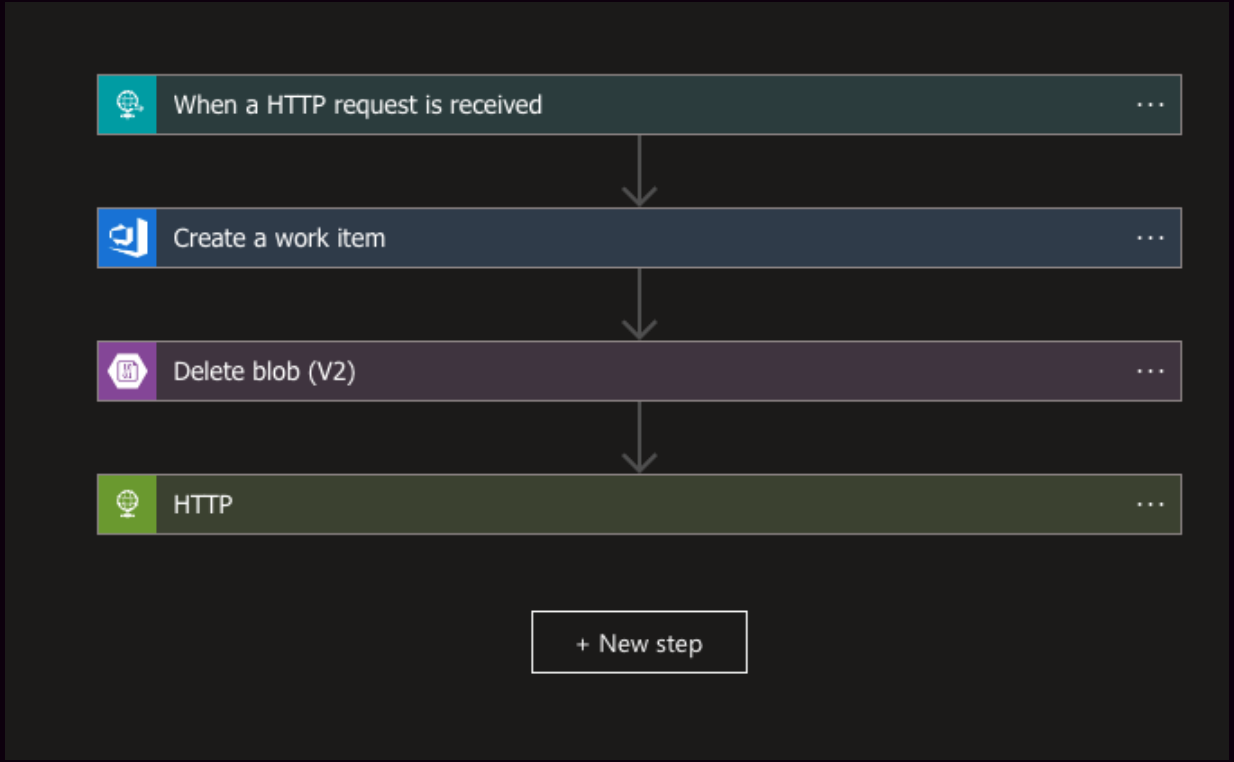
- Serverless with privileged Managed Identities
- Clear-text secrets
- Roles such as Application Administrator, Helpdesk Admin
- Users that are Owners on applications
- Exposed endpoints

Attack Path Tools



Finding Secrets

Logic Apps



NIC
EMPOWER

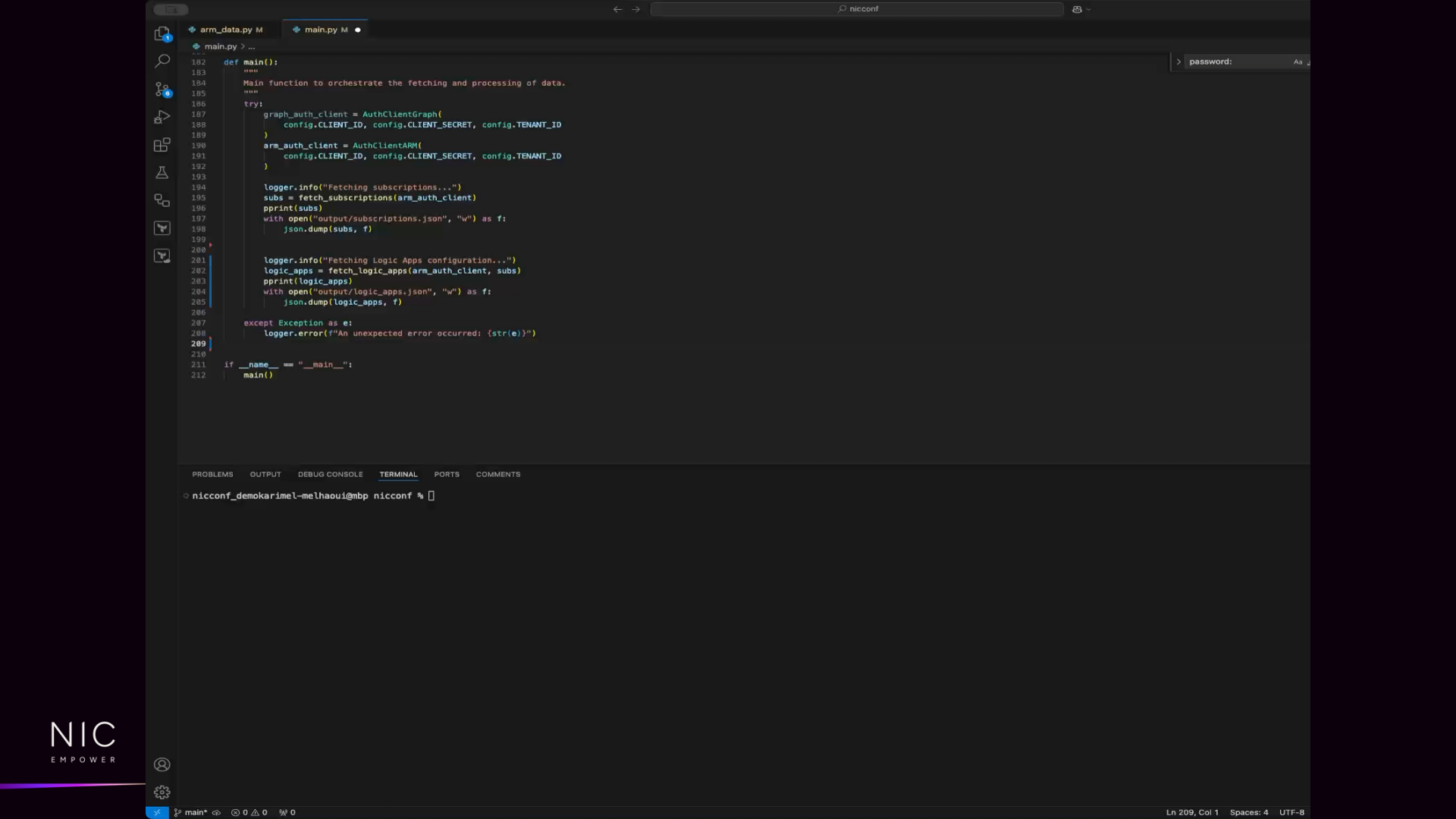
Logic Apps

```python
def get_logic_apps_configuration(arm_auth_client, subscription, resource_group):
    """
    Fetch the configuration of Logic Apps for a specific resource group from the Azure Management API.

    Args:
        arm_auth_client: The authentication client to use for fetching the token.
        subscription (str): The subscription ID to fetch Logic Apps for.
        resource_group (str): The resource group ID to fetch Logic Apps for.

    Returns:
        List[Dict]: A list of dictionaries containing the Logic Apps configuration details.
    """
    token = arm_auth_client.get_token()
    url = f"https://management.azure.com/{resource_group}/providers/Microsoft.Logic/workflows?api-version=2019-05-01"
    headers = {"Authorization": f"Bearer {token}"}
    response = requests.get(url, headers=headers)
    if response.status_code == 200:
        return response.json().get("value", [])
    else:
        print(f"Error fetching Logic Apps configuration: {response.status_code} - {response.text}")
        return []
```

NIC
EMPOWER


nicconf
arm_data.py M
main.py M
main.py > ...
password:
def main():
"""
Main function to orchestrate the fetching and processing of data.
"""
try:
graph_auth_client = AuthClientGraph(
config.CLIENT_ID, config.CLIENT_SECRET, config.TENANT_ID
)
arm_auth_client = AuthClientARM(
config.CLIENT_ID, config.CLIENT_SECRET, config.TENANT_ID
)
logger.info("Fetching subscriptions...")
subs = fetch_subscriptions(arm_auth_client)
pprint(subs)
with open("output/subscriptions.json", "w") as f:
json.dump(subs, f)
logger.info("Fetching Logic Apps configuration...")
logic_apps = fetch_logic_apps(arm_auth_client, subs)
pprint(logic_apps)
with open("output/logic_apps.json", "w") as f:
json.dump(logic_apps, f)
except Exception as e:
logger.error(f"An unexpected error occurred: {str(e)}")
if __name__ == "__main__":
main()
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
COMMENTS
nicconf_demokarimel-melhaoui@mbp nicconf %
main*
Ln 209, Col 1
Spaces: 4
UTF-8

NIC
EMPOWER

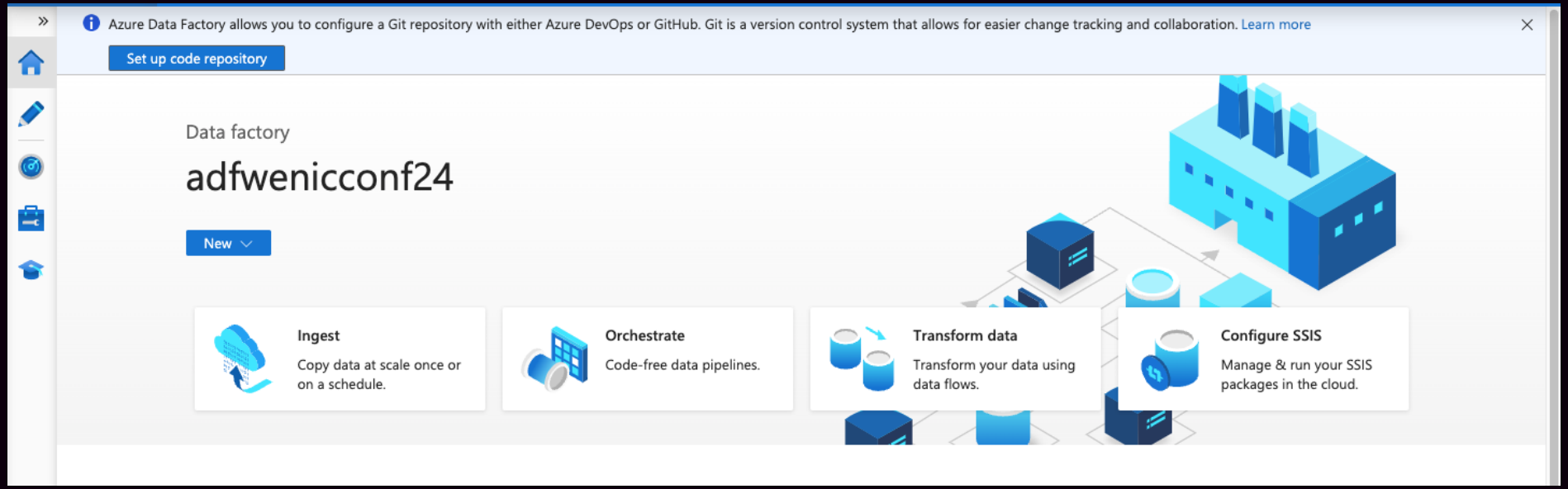
This is ALWAYS the case.

More Secrets

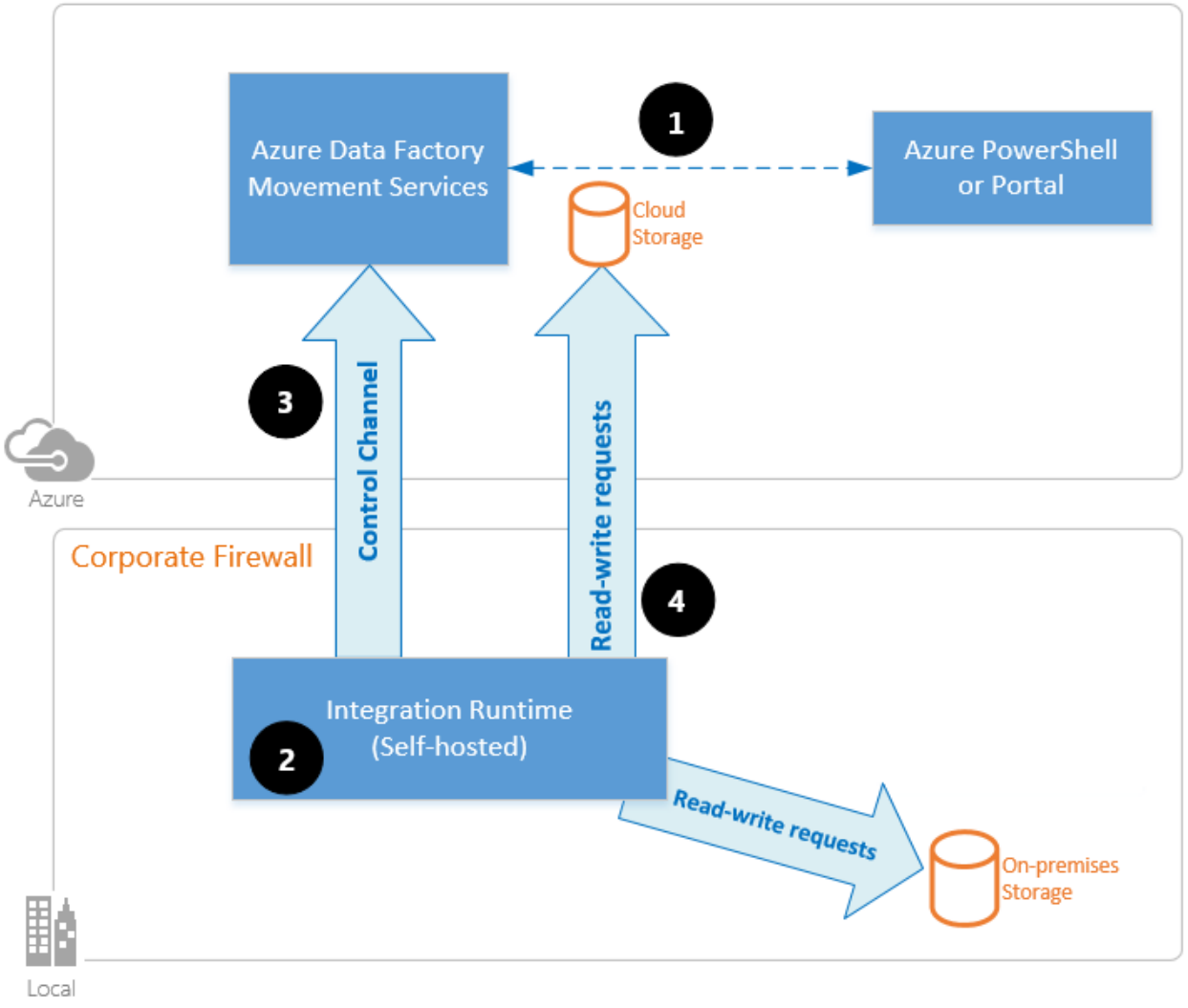
More secrets, means more attack paths

Abusing Azure Data Factory

Azure Data Factory



NIC
EMPOWER


Azure Data Factory
Movement Services
1
Azure PowerShell
or Portal
Cloud
Storage
3
Control Channel
Read-write requests
Azure
Corporate Firewall
4
Integration Runtime
(Self-hosted)
2
Read-write requests
On-premises
Storage
Local

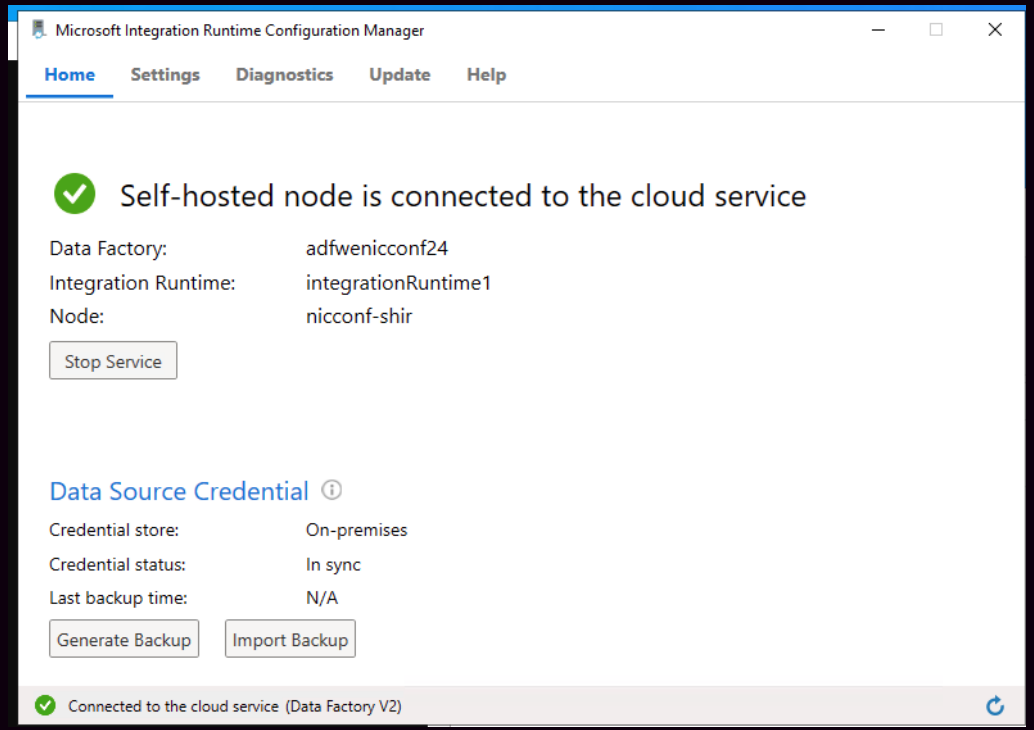
Azure Data Factory

- Azure Data Factory can execute jobs through Self-Hosted Integration Runtime

- Self-Hosted Integration Runtime Downloads Secrets

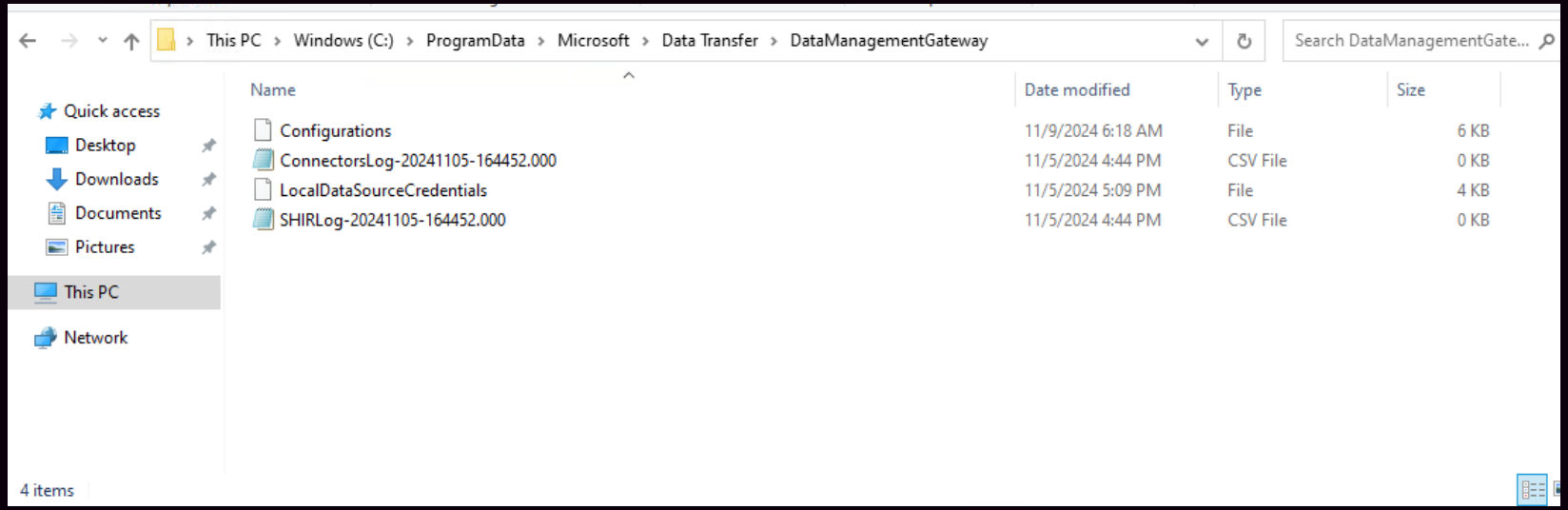
Exploiting SHIR

SHIR



NIC
EMPOWER

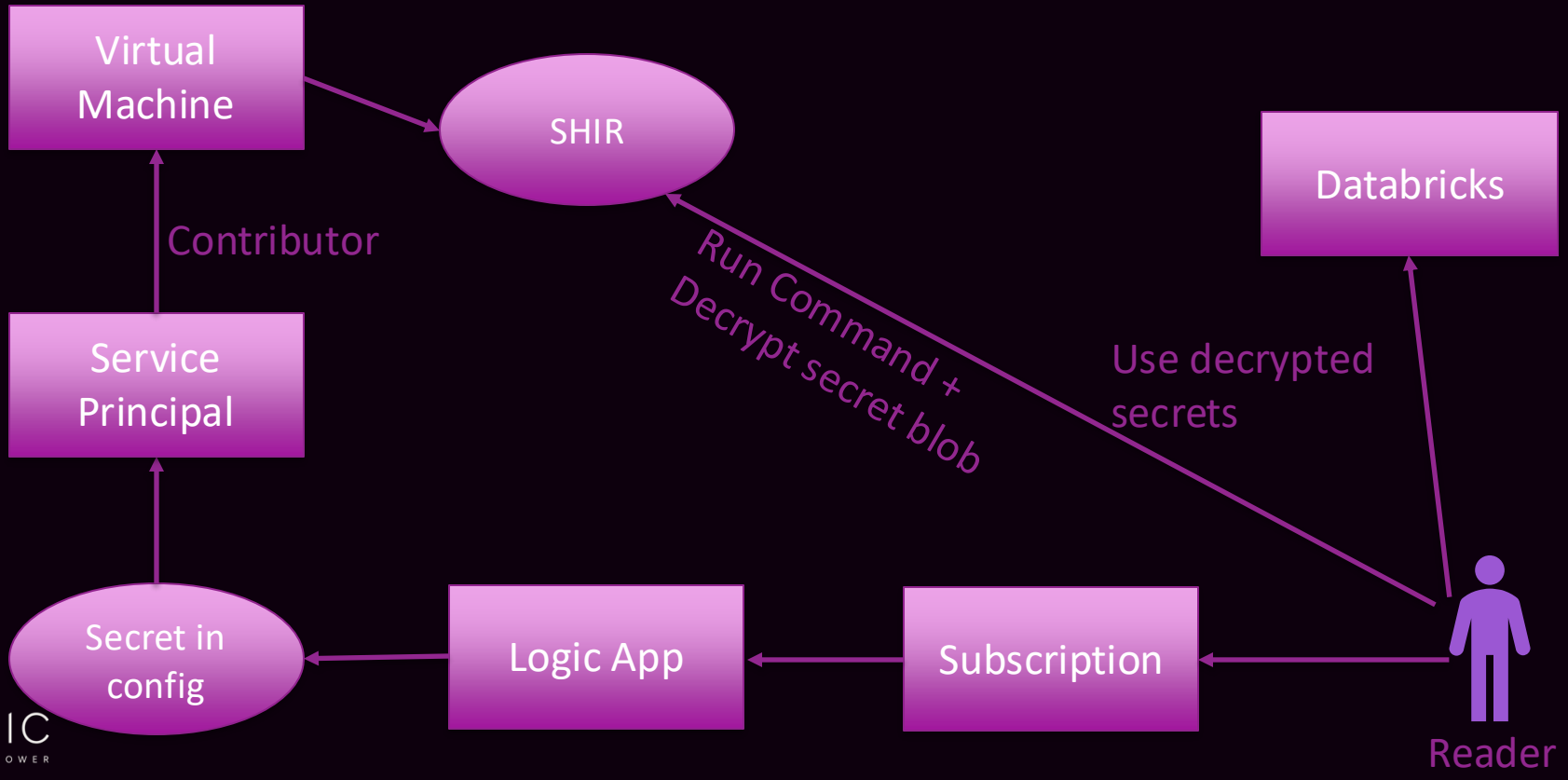
SHIR



NIC
EMPOWER


Recycle Bin
Microsoft Edge
dddd

Data Factory



Azure Data Factory - Abusing the Self-Hosted Integration Runtime

During a recent Cloud Security Assessment, we obtained **Contributor** permissions on a Virtual Machine (VM) configured as a Self-Hosted Integration Runtime (SHIR) instance in an Azure Subscription. This setup raised a question: could we leverage our access to pivot toward more valuable data? To explore this, we recreated the SHIR configuration in our lab environment and used Azure's **Run Command** to modify settings and gain full access to the host machine.

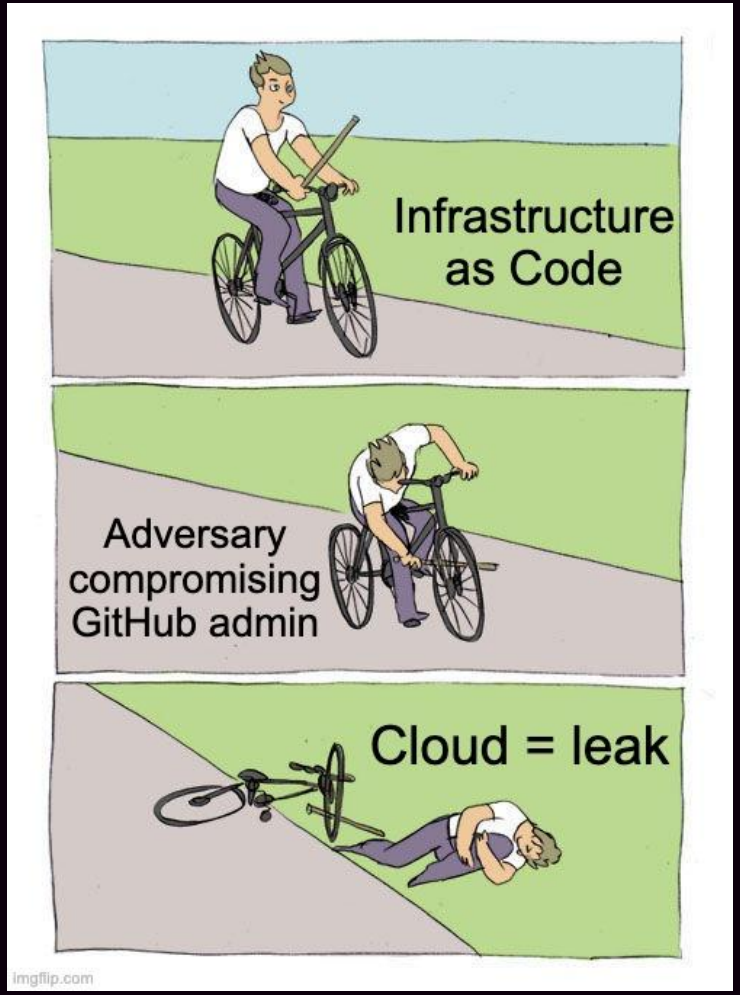
Observing SHIR Credentials with Procmon

Once inside the SHIR host, we ran **Procmon** to monitor activity as we created a new connection in Azure Data Factory that used the SHIR instance. During this process, we observed the creation of a file named `LocalDataSourceCredentials` in the following location:

```
C:\Windows\ProgramData\Microsoft\Data Transfer\DataManagementGateway
```

Going from Code to Cloud

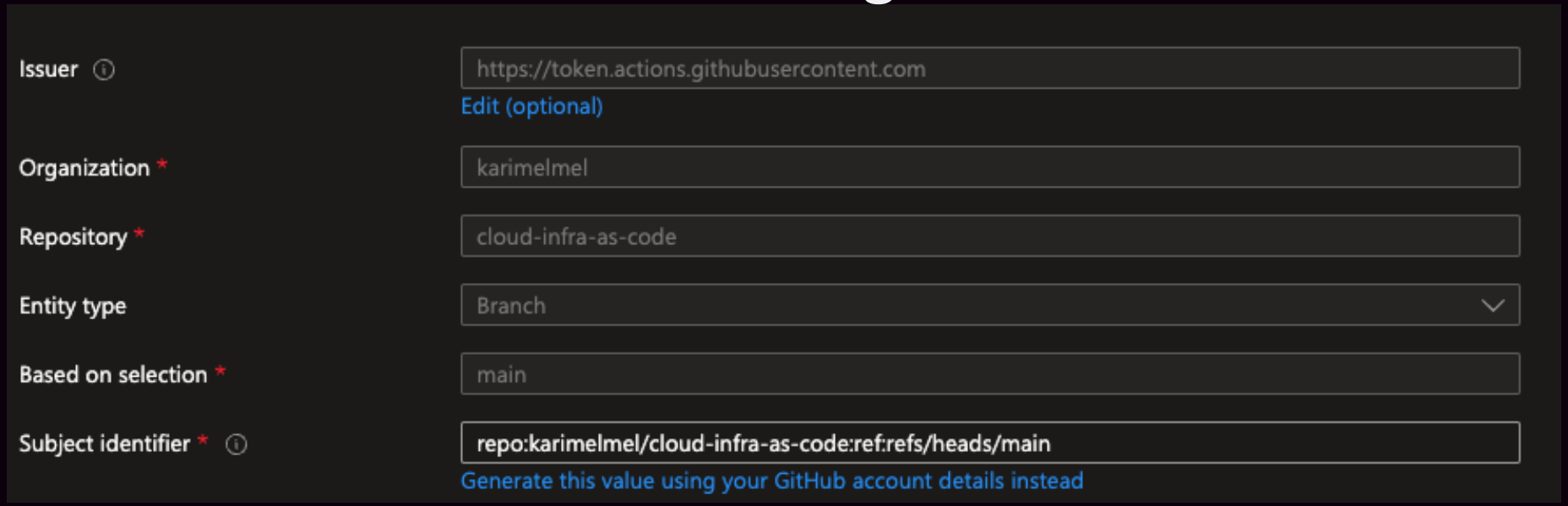
NIC
EMPOWER


Infrastructure as Code
Adversary compromising GitHub admin
Cloud = leak
imgflip.com

NIC
EMPOWER

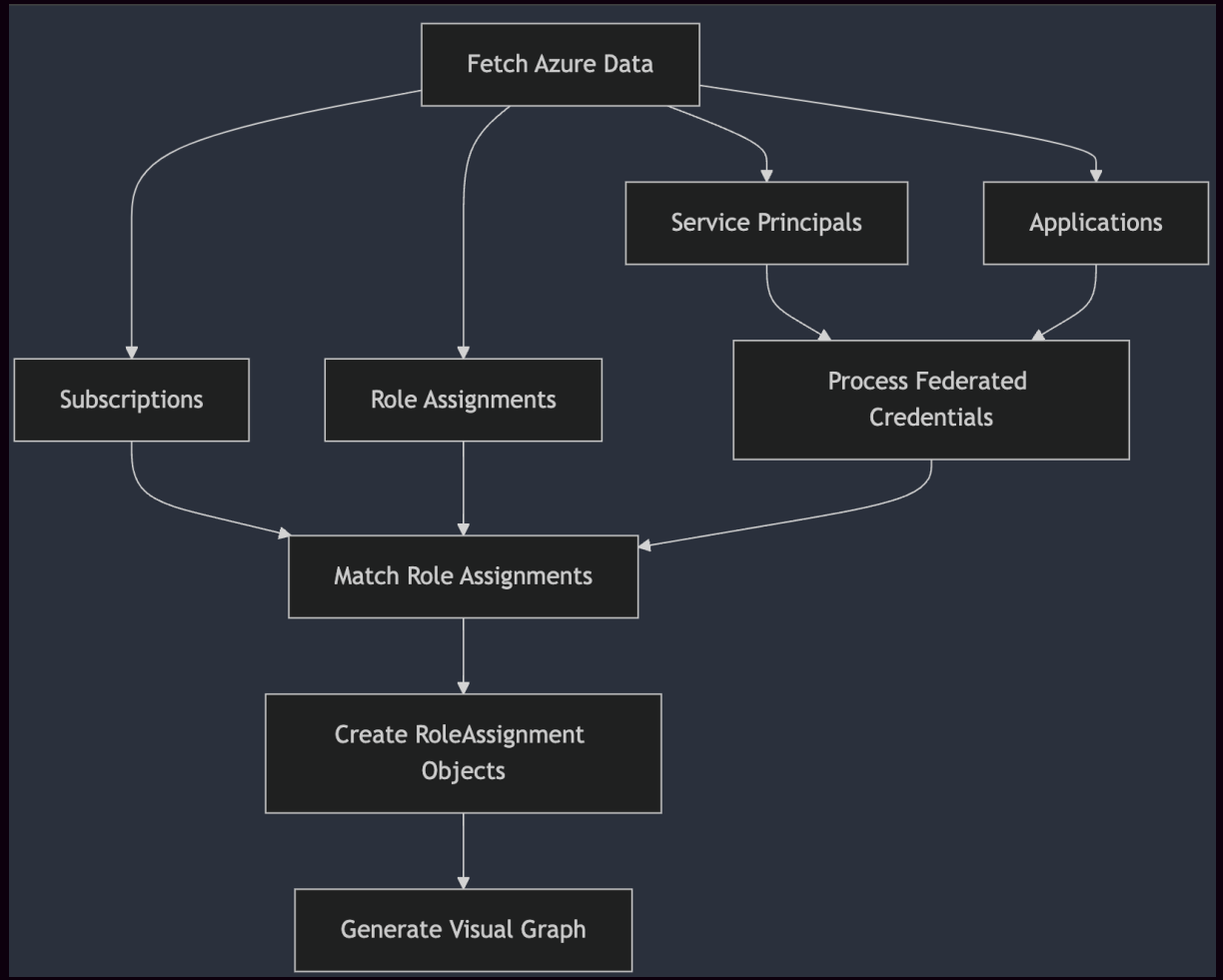
Open ID Connect

OIDC Config flaw



repo:org/repo:pull_request repo:org/* repo:org/repo:<branch>

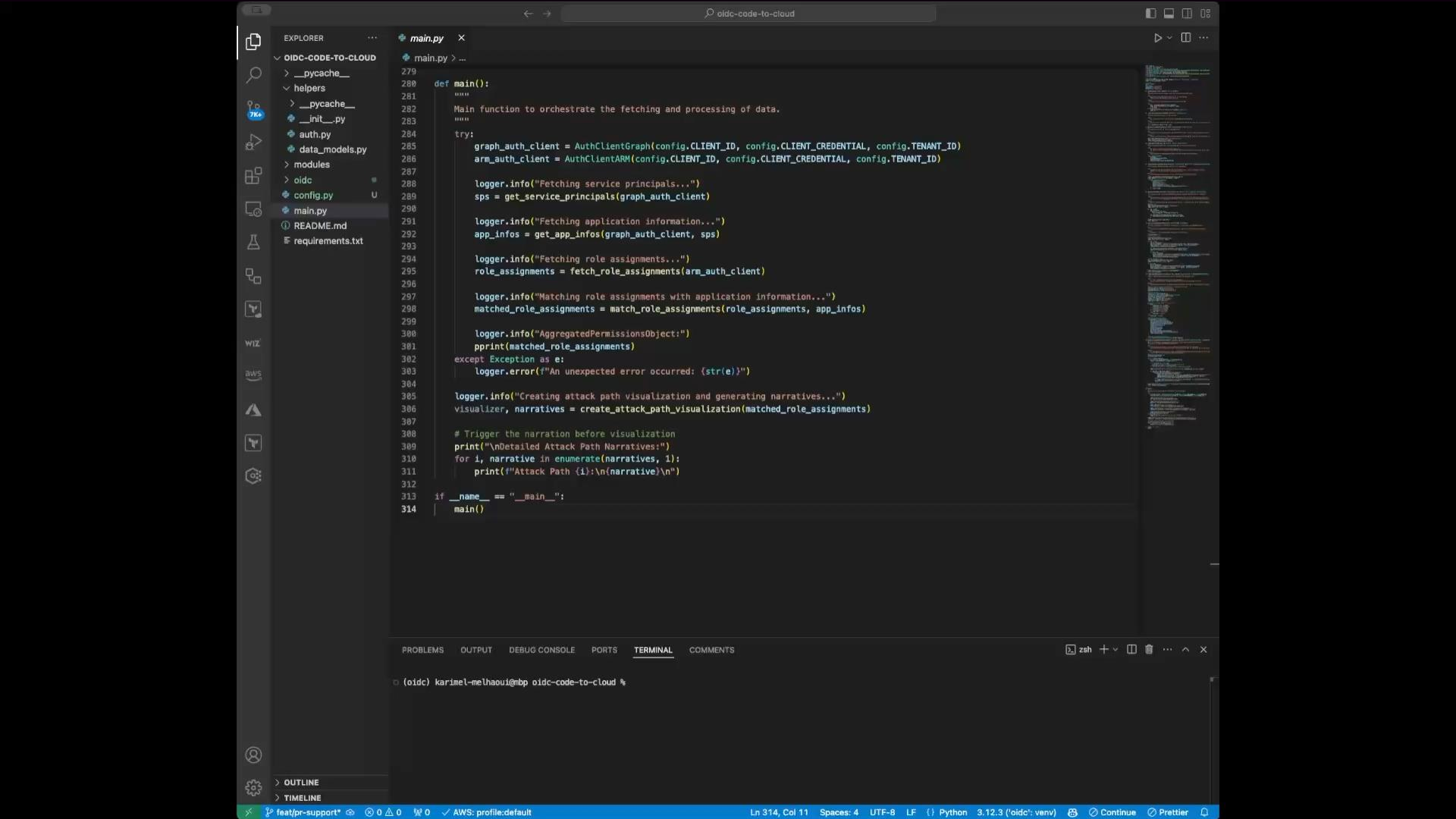
NIC
EMPOWER


Fetch Azure Data
Service Principals
Applications
Subscriptions
Role Assignments
Process Federated Credentials
Match Role Assignments
Create RoleAssignment Objects
Generate Visual Graph

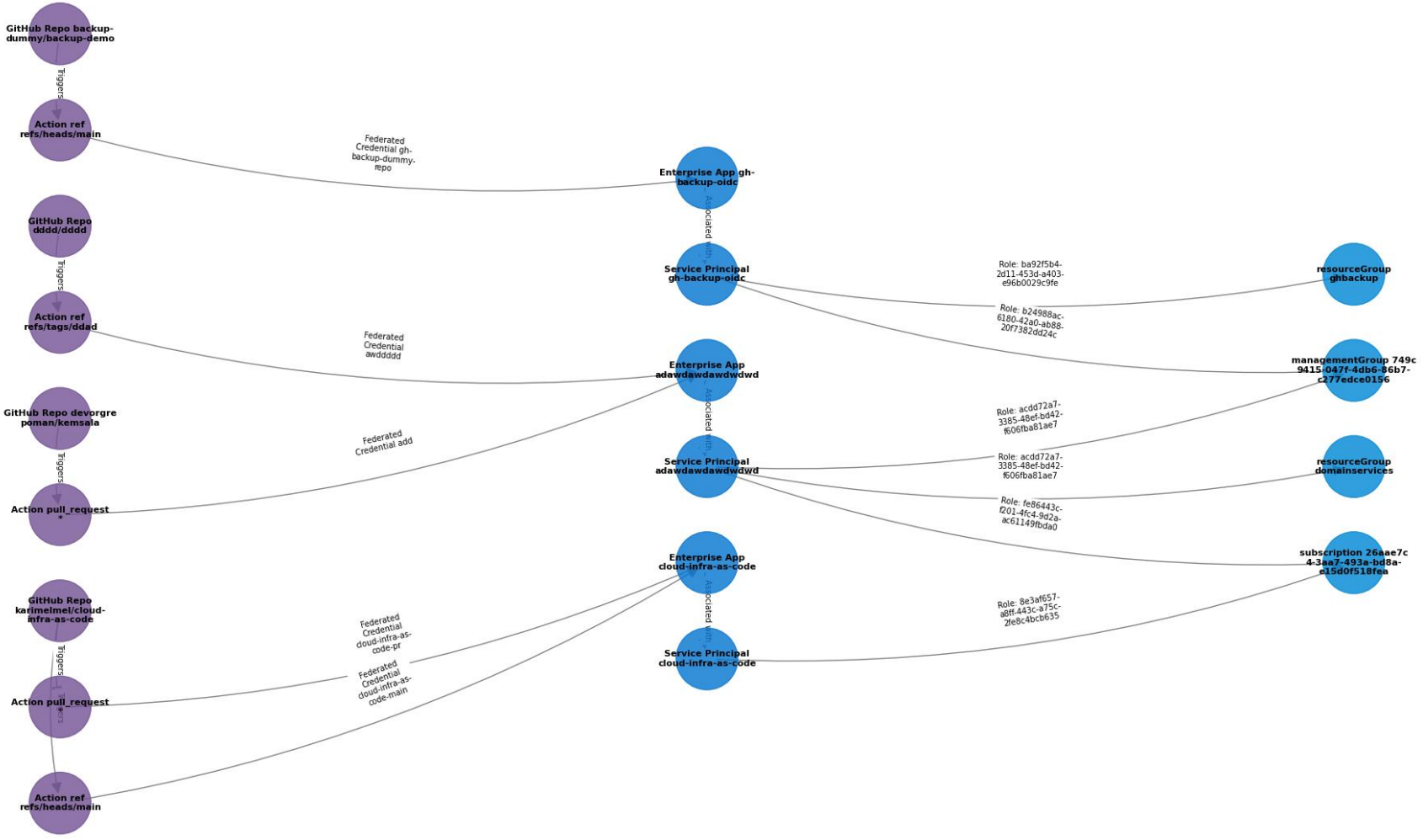
NIC
EMPOWER

Finding all your misconfigured OIDC

*Full session and tool release scheduled for Q1 ‘25


oidc-code-to-cloud
EXPLORER
OIDC-CODE-TO-CLOUD
__pycache__
helpers
__pycache__
__init__.py
auth.py
data_models.py
modules
oidc
config.py
main.py
README.md
requirements.txt
main.py
main.py > ...
279
280 def main():
281 """
282 Main function to orchestrate the fetching and processing of data.
283 """
284 try:
285 graph_auth_client = AuthClientGraph(config.CLIENT_ID, config.CLIENT_CREDENTIAL, config.TENANT_ID)
286 arm_auth_client = AuthClientARM(config.CLIENT_ID, config.CLIENT_CREDENTIAL, config.TENANT_ID)
287
288 logger.info("Fetching service principals...")
289 sps = get_service_principals(graph_auth_client)
290
291 logger.info("Fetching application information...")
292 app_infos = get_app_infos(graph_auth_client, sps)
293
294 logger.info("Fetching role assignments...")
295 role_assignments = fetch_role_assignments(arm_auth_client)
296
297 logger.info("Matching role assignments with application information...")
298 matched_role_assignments = match_role_assignments(role_assignments, app_infos)
299
300 logger.info("AggregatedPermissionsObject:")
301 pprint(matched_role_assignments)
302 except Exception as e:
303 logger.error(f"An unexpected error occurred: {str(e)}")
304
305 logger.info("Creating attack path visualization and generating narratives...")
306 visualizer, narratives = create_attack_path_visualization(matched_role_assignments)
307
308 # Trigger the narration before visualization
309 print("\nDetailed Attack Path Narratives:")
310 for i, narrative in enumerate(narratives, 1):
311 print(f"Attack Path {i}:\n{narrative}\n")
312
313 if __name__ == "__main__":
314 main()
PROBLEMS
OUTPUT
DEBUG CONSOLE
PORTS
TERMINAL
COMMENTS
zsh
(oidc) karimel-melhaoui@mbp oidc-code-to-cloud %
OUTLINE
TIMELINE
feat/pr-support*
0 0
0
AWS: profile:default
Ln 314, Col 11
Spaces: 4
UTF-8
LF
Python
3.12.3 ('oidc': venv)
Continue
Prettier

Attack Path: GitHub to Azure via Entra ID



Reporting

NIC
EMPOWER

Report output

- Impact
- Reproducibility
- Technical details vs. executive summary
- Actionable improvements, not one-click remediations

Thank you for watching!

karim@o3c.no

NIC
EMPOWER