

# Detecting Azure Hybrid Machine Attack Paths with Graph Theory

Author: Shawn Woods, [Shawn.Douglas.Woods@gmail.com](mailto:Shawn.Douglas.Woods@gmail.com)

Advisor: Michael Long

Accepted: December 19<sup>th</sup>, 2024

## Abstract

Today's on-premises and cloud environments are ever-growing and becoming increasingly complex. Attackers know this and can and will exploit this fact, pivoting from network to network. Identity and access management is more critical than ever with hybrid cloud environments. Proper privileges must be assigned according to least privilege principles; if they are not, this is where the problem starts. Attack path mapping and graph databases offer a solution that can highlight potential paths to compromise. Through simple Cypher queries, defenders can observe the potential risks within their environments and mitigate them as needed. This research extends the data collected by the security tool BloodHound to uncover hidden connections between on-premises devices and their cloud identities within an Azure environment. The research offers insights into how organizations can utilize standard tools to add context to their attack maps.

## 1. Introduction

A picture is worth a thousand words, a phrase that underscores the power of images to communicate complex ideas quickly and effectively. While words require context and understanding for comprehension, images allow us to instantly grasp relationships, patterns, and meanings that might otherwise take pages of text to convey. In incident response, identity management, and cybersecurity in general, understanding intricate connections is critical; visual tools can simplify these connections, leading to quicker, more insightful decisions.

In cybersecurity, adopting a graph-based perspective is increasingly recognized as commonplace for addressing complex challenges. John Lambert, a cybersecurity engineer at Microsoft, emphasized this by stating, “Defenders think in lists. Attackers think in graphs. As long as this is true, attackers win” (Lambert, 2015). Defenders can better identify vulnerabilities, anticipate attack paths, and strengthen security by visualizing networks as interconnected graphs.

Year over year, hybrid cloud environments have become increasingly popular in modern IT infrastructure, enabling organizations to harness the scalability and innovation of public cloud platforms while retaining control over critical on-premises resources (Franklin, 2023; Reese, 2022; Vailshery, 2024). However, these setups introduce unique security challenges, requiring the synchronization of data, services, and identities across different systems (SentinelOne, 2024). Due to their complex nature, these systems can invite misconfigurations and over-permissioned identities, leading to unforeseen vulnerabilities. Effectively understanding and visualizing the interconnections of the hybrid cloud is crucial for identifying potential risks and strengthening an organization’s security posture.

Graph theory provides a robust framework for this visualization. It models entities, such as computers or users, as nodes and their relationships as edges (e.g., “Admin of,” “Synced to”) (Vaidehi Joshi, 2017). Graph theory can employ algorithms like Dijkstra’s for the shortest path between nodes and centrality algorithms to highlight high-risk pathways and critical nodes within networks (Neo4j, n.d.-a; Neo4j, n.d.-b). For instance, the shortest path algorithm can reveal the quickest routes attackers might exploit

to escalate privileges. In contrast, centrality algorithms can identify essential identities that serve as critical access points.

The OWASP Top 10 list for 2021 highlights insecure design and security misconfiguration as the third and fifth most critical security risks, respectively (OWASP, 2021). The growing complexity of modern environments exacerbates these risks, where misaligned permissions and insecure configurations can introduce vulnerabilities. Graph theory can address these challenges by offering a way to visualize and analyze interconnected hybrid cloud systems.

This research explores the application of graph theory in hybrid Azure cloud environments to evaluate its effectiveness in highlighting potential risks, such as over-permissioned device identities and excessive user access privileges. By leveraging graph-based algorithms and additional data collection, the study aims to uncover hidden relationships in these complex environments, providing cybersecurity professionals with actionable steps to understand better and address identity management and security challenges in their hybrid environments.

## **1.1. Background**

### **1.1.1. Cloud and Hybrid Cloud**

Cloud technologies are nearly universal; some resources suggest that as much as 98% of all companies utilize them (Baron et al., 2023). The 2024 State of the Cloud Report from Flexera, based on a survey of 753 businesses, revealed that 73% of surveyed companies are adopting hybrid cloud strategies (2024 State of the Cloud Report, 2024). Figure 1 below from the 2024 State of the Cloud Report shows the widespread adoption of the hybrid cloud.

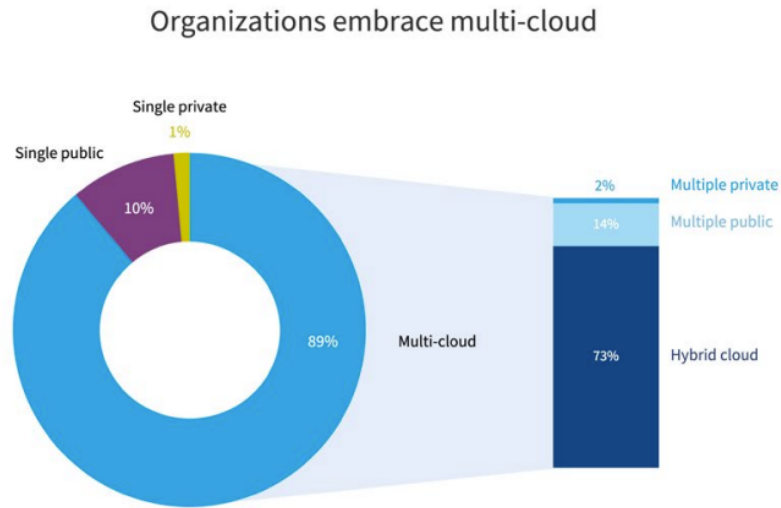


Figure 1: Flexera State of the Cloud Report on Adoption of Hybrid Cloud

Google defines a hybrid cloud as a computing environment where applications run using a combination of on-premises resources in a business's data center and resources in a public cloud (Google Cloud, 2024). A hybrid environment requires connectivity between on-premises and cloud resources, enabling synchronization, data movement, and access to cloud technologies. Many organizations adopt hybrid cloud setups to optimize costs, meet regulatory requirements, and leverage additional cloud provider technologies (Microsoft, 2023a).

However, the integration of on-premises and cloud environments introduces unique security challenges. Hybrid setups necessitate additional overhead and training for security practitioners, as misconfigurations and excessive permissions can create vulnerabilities. John Lambert (2021) summarized:

Attackers seek to turn illegitimate access into legitimate access. Your network often provides all the accesses and capabilities the attacker needs — because after all, you need to manage the network too. If they obtain your legitimate credentials, they can use the tools and means you have put in place to achieve their goals. So while malware and exploits may play some part in their toolkit, attackers are just IT with different goals.

This quote encapsulates the challenges of defending hybrid environments, where every user and system account becomes a potential access point. In these environments, identity is everything (Elmiger et al., 2023). A comprehensive view of the entire network is critical to identifying risks and preventing attacks. Graph-based approaches can help security practitioners pinpoint previously unseen paths to compromise by visualizing complex interconnections within hybrid environments.

### 1.1.2. Graph Theory and Graph Databases

Graph theory, founded in the 18th century by Swiss mathematician Leonhard Euler, provides a framework for representing relationships as nodes and edges. Euler established the foundation of graph theory in 1736 through his work on the Seven Bridges of Königsberg problem, which asked whether one could traverse all seven bridges in the city without crossing any bridge more than once. Euler demonstrated that the problem had no solution by introducing the concept of a graph, representing landmasses as vertices (nodes) and bridges as edges (Paoletti, 2011).

The evolution of graph databases, critical for implementing graph theory in practical applications, began in the 1970s with early experiments in network database models, as explored in Richard Phillips' work on graphical entities (Phillips, 1977). These models demonstrated the ability to represent and query graphical relationships. Phillips introduced concepts like adjacency matrices, representing all the connections between nodes in a graph, and graphical relational operators to query adjacency and proximity. These tools were crucial for enabling pathfinding and neighbor-based queries, laying the foundation for modern graph databases.

The rise of NoSQL technologies in the 2000s brought graph databases into mainstream use, enabling them to handle interconnected data structures at scale efficiently. Neo4j, released in 2007, became one of the first commercially viable graph database platforms. Later, in 2011, Neo4j started offering intuitive querying capabilities through its Cypher query language. This innovation marked a turning point in adopting graph databases for various applications, including cybersecurity, and in 2015, the openCypher project standardized Cypher as the query language for multiple graph database systems (Francis et al., 2018).

Graph databases are the backbone of attack graphs, which visualize systems and connections to identify vulnerabilities (Elmiger et al., 2023). These attack graphs help highlight paths that attackers might exploit to compromise systems. Tools like BloodHound use graph theory and graph databases to map relationships between on-premises resources and, more recently, cloud identities to uncover privilege escalation paths. In August 2024, the BloodHound team extended its capabilities to link Azure Entra ID Users with on-premises Active Directory Users, exposing hybrid attack paths. However, significant gaps remain in mapping other critical identities within hybrid environments.

This paper addresses these gaps by developing queries to bridge on-premises and cloud environments, incorporating additional identities into the analysis. By leveraging graph databases and graph theory, this research aims to enhance the visualization and understanding of complex hybrid environments, enabling better identification of vulnerabilities and potential attack paths.

## **2. Research Method**

This research will specifically examine the ability of graph theory and graph databases to show potential attack paths concerning device identities in hybrid cloud environments. First, the results of a standard cybersecurity tool, BloodHound, will be examined. Then, the data gathered will be further enriched using Microsoft Azure and Graph PowerShell modules. Enriching the data with additional data points and examining the new dataset in a graph database to observe any new relationships between resources that may indicate an attack path.

### **2.1. Environment**

The test environment, seen in Figure 2, reflected an actual Microsoft hybrid cloud setup. VMware Workstation Pro 17.6.1 hosted the on-premises environment comprised of two Windows Server 2022 machines: an Active Directory Domain Controller and an application server representing a web server or SQL server in a real environment.

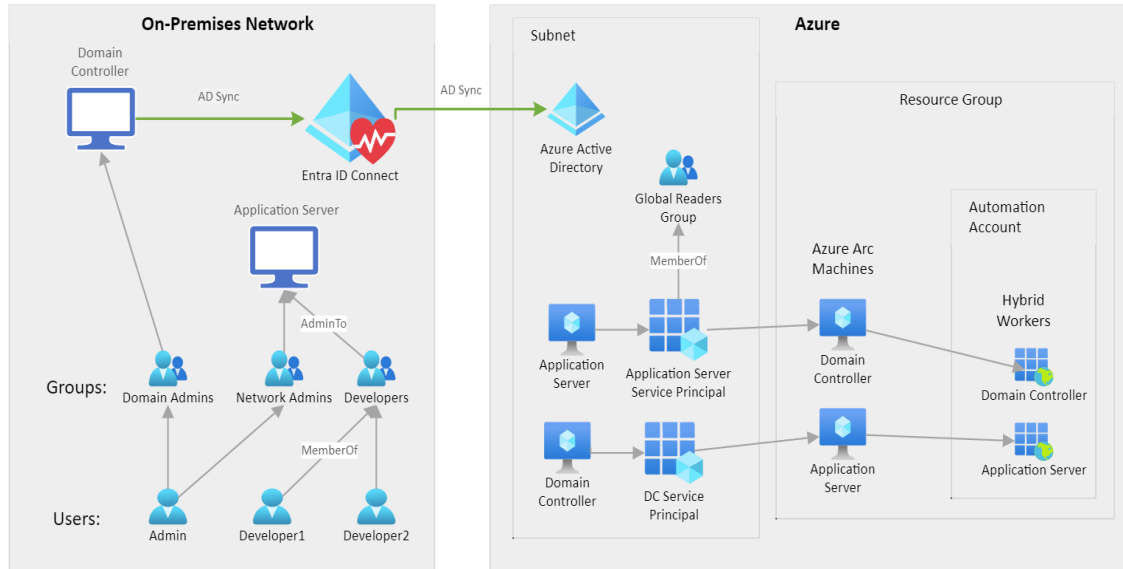


Figure 2: On-premises and Cloud Environment

The test domain contained multiple custom user accounts and security groups. The setup created a developers and network admins group with administrative access to the application server. The domain admin group had administrative access to the domain controller.

A Microsoft Entra tenant and an Azure subscription were required to use Entra Connect and Azure resources (Microsoft, 2023b). A free Azure subscription with credit was acquired to integrate with Microsoft Azure and Entra ID. Microsoft Entra Connect was installed and configured on the domain controller to sync identities to the Entra ID tenant. A new resource group was created in the Azure tenant to onboard the on-premises machines to Azure Arc. The onboarding process creates an identity known as an enterprise application or service principal in the Entra ID tenant.

An Azure Automation Account was created inside the resource group. Hybrid-worker groups were made inside the Automation Account—one for the domain controller and one for the application server. These groups are linked to the Azure Arc machine. The automation account allows for the execution of Runbooks, which are scripts or workflows stored within the Automation Account for automation on-premises or in the cloud. Adding the Azure Arc machines to the hybrid worker groups installs the Hybrid Worker Extension on the Azure Arc machine. The identity for the application server was

granted a Global Reader role within Entra ID. Below is a table containing the components of the environment.

| Component                      | Count | Description   |
|--------------------------------|-------|---|
| <b>Server 2022 Machines</b>    | 2     | Includes Domain Controller, Application Server  |
| <b>AD Domain</b>               | 1     | On-premises Active Directory environment  |
| <b>Users</b>                   | 10    | Representing employees of the network   |
| <b>On-premises Groups</b>      | 3     | Developers, Network Admins, Domain Admins   |
| <b>Azure Arc Connections</b>   | 2     | All server machines are connected to Azure Arc for centralized management and monitoring.   |
| <b>Entra ID</b>                | 1     | Integrated with on-premises Active Directory  |
| <b>Automation Account</b>      | 1     | Azure Automation Account connected to automate routine tasks across the hybrid environment.   |
| <b>Hybrid Worker Group</b>     | 2     | A group within the Automation account linked to Azure Arc machines  |
| <b>BloodHound/Neo4j Server</b> | 1     | Ubuntu server, with Docker Community Edition and Docker Compose, installed to initialize the BloodHound Community Edition environment |
| <b>Data Collection Server</b>  | 1     | Windows 2022 Server with SharpHound and AzureHound collectors   |

Table 1: Outline of On-premises and Cloud Environment

The environment for BloodHound consisted of an Ubuntu 24.04 machine to host the BloodHound components and a Windows Server 2022 machine to collect information from the test environment. The Windows machine had the collectors for BloodHound on it, AzureHound version 2.2.1, and SharpHound Version 2.5.8. Installed on the Windows machine were Microsoft PowerShell and the Az PowerShell module to connect to and collect data from the Azure environment. Installed on the Ubuntu machine were Docker Community Edition, Docker Compose, and BloodHound. The original Docker Compose file (linked in Appendix A) installed and configured the BloodHound environment. This deployed the environment with BloodHound Version 6.2.0-rc5, Neo4j Version 4.4, and Postgres SQL Version 16. The docker file had to be modified, and the changed section is



also included in Appendix A. Additions had to be made to install the Awesome Procedures on Cypher (APOC) add-on library to allow for importing additional JSON data. See Figure 3 for these customizations.

```
graph-db:
  image: docker.io/library/neo4j:4.4
  environment:
    - NEO4J_AUTH=${NEO4J_USER:-neo4j}/${NEO4J_SECRET:-bloodhoundcommunityedition}
    - NEO4J_dbms_allow_upgrade=${NEO4J_ALLOW_UPGRADE:-true}
    - NEO4JLABS_PLUGINS=["apoc"]
    - NEO4J_dbms_security_procedures_unrestricted=apoc.*
    - NEO4J_dbms_security_procedures_allowlist=apoc.*
    - NEO4J_apoc_import_file_enabled=true
    - NEO4J_apoc_export_file_enabled=true
```

Figure 3: Additional Configuration to Install APOC Library in Neo4j Docker Container

## 2.2. Testing Variables

This research manipulated and measured specific variables to evaluate the efficacy of graph theory in identifying new attack paths within hybrid cloud environments. The manipulated variable is the dataset used for analysis. Two configurations were tested: the default dataset collected using AzureHound and SharpHound, part of BloodHound's standard collectors, and an enriched dataset, which included additional nodes, properties, and relationships imported into Neo4j using the APOC library.

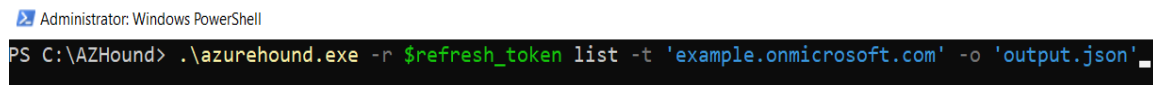
The measured variable is the number and type of hybrid device attack paths identified within each dataset. This includes paths demonstrating privilege escalation, lateral movement, and hybrid attack vectors involving on-premises devices and Azure identities. By comparing the default dataset with the enriched dataset, this research evaluates the impact of additional data collection and relationship mapping on the ability to uncover previously unseen attack paths.

## 2.3. Data Collection

### 2.3.1. Default BloodHound Collection

BloodHound is a cybersecurity tool that helps organizations understand and protect their networks by mapping out the connections between people, computers, and permissions within an environment. It identifies pathways and critical points in a network where a malicious actor or rogue employee could move around and increase their permissions. The solution comprises a few primary parts: collectors, a Neo4j Database, and the BloodHound application. There are two collectors: AzureHound and SharpHound.

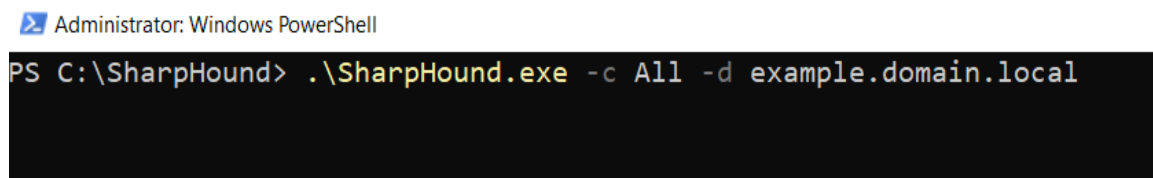
AzureHound uses Microsoft APIs to query an Azure environment. The SharpHound executable must be run with the tenant, the list command, and either a username and password or a refresh token. The list command tells the program to retrieve as much information as possible from Entra ID and Azure. Figure 4 shows the command used to collect information from the test environment in this paper.



```
Administrator: Windows PowerShell
PS C:\AZHound> .\azurehound.exe -r $refresh_token list -t 'example.onmicrosoft.com' -o 'output.json'
```

Figure 4: Execution of AzureHound

SharpHound is used to examine an Active Directory domain. It should be run in a location with the best line of sight to the computers and domain controllers. It should also be run with an account with enough permissions to collect all the relevant information. Figure 5 shows the command used to collect information from the test environment; it was run as a domain administrator.



```
Administrator: Windows PowerShell
PS C:\SharpHound> .\SharpHound.exe -c All -d example.domain.local
```

Figure 5: Execution of SharpHound

These applications, executed in the proper location and with the appropriate permissions, generate JSON files that contain information on the nodes and relationships within an on-premises Active Directory domain and Azure tenant. The BloodHound application allows the importation of the JSON files created by the collectors. After the data are imported, they are analyzed by BloodHound and sent to the Neo4j database. They are now ready for examination through the BloodHound web interface. BloodHound offers some prebuilt queries and provides the ability to view relationships between two nodes and query the data directly using Cypher.

### **2.3.2. Custom Data Collection Procedure**

As previously mentioned, Neo4j is a database designed to store and analyze relationships between data points, often called a graph database. Unlike traditional databases that use tables to organize information, Neo4j uses a structure similar to a web or network. In Neo4j, data is represented as nodes (which can be people, servers, or identities), their properties (key-value pairs used for additional data about nodes), and relationships (connections between nodes). This makes it especially useful for understanding connections, patterns, and networks. These data are searchable by using a language called Cypher.

Cypher is a query language similar to SQL queries but used for graphs. It provides a way to match patterns and relationships based on ASCII art syntax. That means it is simple to understand a query by looking at it, using parentheses to identify nodes, square brackets to identify relationships, and lines or arrows to show the directionality or not of a relationship. Neo4j and Cypher allow for the integration of plugins, such as the Awesome Procedures on Cypher (APOC) add-on library. This add-on library extends the Cypher language for data integration and conversion. The scope of the use case for APOC in this paper is for the procedure `apoc.load.json`, which allows importing JSON files into the Neo4j database through Cypher queries. This plugin was used to add additional data to the default data collected by BloodHound.

To visualize paths from on-premises devices to Azure devices, Azure Service Principals, and Azure Arc Machines, additional data needed to be gathered and imported into the Neo4j database. The following outlines the data collection procedures taken.

## On-Premises Computer to AZDevice

**Observation:** The initial dataset collected by BloodHound revealed that while the AZDevice node contained a deviceid property, it did not capture the corresponding objectguid from on-premises devices, preventing the creation of a meaningful link between these nodes.

**Data Collection Command:** To address this, the following PowerShell command was used to query the on-premises domain, retrieving the objectguid and distinguishedname of all devices:

```
get-adcomputer -Filter * -properties objectguid,distinguishedname
|select objectguid, distinguishedname |
convertto-json |
Out-File C:\users\swoods\Desktop\computers.json
```

**Importing into Neo4j:** The retrieved JSON file was imported into Neo4j using the following Cypher command:

```
CALL apoc.load.json("file:///import/computers.json")
YIELD value
WITH TOUPPER(value.distinguishedname) as dn, value
MATCH (computer:Computer)
WHERE TOUPPER(computer.distinguishedname) = dn
SET computer.objectguid = value.objectguid
RETURN computer;
```

**Relationship Creation:** A directional relationship OnPremToAzDevice was created between on-premises devices and Azure devices using this Cypher command:

```
MATCH (op:Computer), (dev:AZDevice)
WHERE op.objectguid = dev.deviceid
MERGE (op)-[:OnPremToAZDevice]->(dev)
RETURN op, dev;
```

## Azure Device to Hybrid Machine to Service Principal

**Observation:** The initial dataset collected by BloodHound did not create Azure Arc machine nodes in the Neo4j database. The lack of the Azure Arc machine neglects the link between an on-premises device, an Azure device, and the associated service principal.

**Creating Hybrid Machine Nodes:** Azure Arc Machines represent critical points of connection between on-premises and cloud resources. The following PowerShell script gathered relevant information about Azure Connected Machines, including their properties and associated automation accounts:

```
$connectedmachines = get-azconnectedmachine
$transformedmachines = foreach ($connectedmachine in
$connectedmachines) {
    $extensions = Get-AzConnectedMachineExtension -MachineName
        $connectedmachine.DisplayName -ResourceGroupName
        $connectedmachine.ResourceGroupName
    $hybridworkerextension = $extensions |
        where { $_.InstanceViewName -eq
            "HybridWorkerExtension"}
    $automationaccounturl = if ($hybridworkerextension -ne
    $null) {
        $hybridworkerextension.Setting |
        ConvertFrom-Json |
        select -expandproperty AutomationAccountURL
    } else { $null }
    $connectedmachine | Select `
        AdFqdn,
        AgentConfigurationConfigMode,
        AgentVersion,
        DisplayName,
        DnsFqdn,
        DomainName,
        Fqdn,
        Id,
        IdentityPrincipalId,
        IdentityTenantId,
        IdentityType,
        Name,
        ResourceGroupName,
        Status,
        Type,
        @{name = "AutomationAccountURL";
        expression = {$automationaccounturl}}
```

```
}
```

**Node Creation Command:** The following Cypher command created the HybridMachines nodes and their properties in the Neo4j database:

```
CALL apoc.load.json("file:///import/devices.json")
YIELD value
MERGE (device:HybridMachines {Id: value.Id})
SET device += {
    AdFqdn: value.AdFqdn,
    AgentConfigurationConfigMode:
    value.AgentConfigurationConfigMode, AgentVersion:
    value.AgentVersion,
    DisplayName: value.DisplayName,
    DnsFqdn: value.DnsFqdn,
    DomainName: value.DomainName,
    Fqdn: value.Fqdn,
    IdentityPrincipalId: value.IdentityPrincipalId,
    IdentityTenantId: value.IdentityTenantId,
    IdentityType: value.IdentityType,
    Name: value.Name,
    ResourceGroupName: value.ResourceGroupName,
    Status: value.Status,
    Type: value.Type }
```

**Relationship Creation:** In the comparison between Azure Devices and Hybrid Machines, the only common property that can be used to link them is the device display name. However, since both types of devices are synchronized from Active Directory, they will have the same value for this property

```
MATCH (azd:AZDevice), (azhb:HybridMachines)
WHERE TOLOWER(azd.displayname) = TOLOWER(azhb.DisplayName)
MERGE (azd)-[:AZDeviceToHybridMachine]->(azhb);
```

**Relationship Creation:** Between Hybrid Machines and their Service Principals:

```
MATCH (azhb:HybridMachines), (asp:AZServicePrincipal)
WHERE TOLOWER(azhb.IdentityPrincipalId) = TOLOWER(asp.objectid)
MERGE (azhb)-[:HybridMachineToAZServicePrincipal]->(asp);
```

### 3. Findings and Discussion

The findings from this research underscore the effectiveness of graph theory in identifying hybrid attack paths within complex cloud environments. By leveraging graph-based modeling and analysis, this study demonstrated how the enriched dataset, with additional nodes, properties, and relationships, enabled the identification of undetectable attack paths in the default BloodHound dataset. Graph theory’s inherent ability to model interconnected systems allowed a deeper exploration of privilege escalation, lateral movement, and hybrid attack vectors spanning on-premises devices and Azure identities.

#### 3.1. BloodHound Defaults

As previously mentioned, in August 2024, the team behind BloodHound started adding hybrid cloud attack paths to the program. However, as shown in Figure 6, these capabilities are limited to identifying user-to-user attack paths across Active Directory and Azure Entra ID.

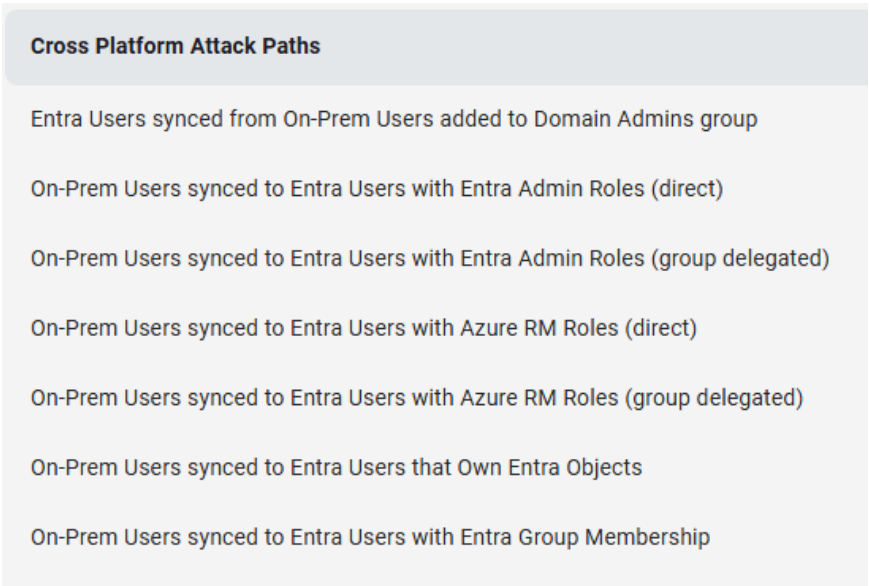


Figure 6: BloodHound Cross Platform Attack Paths

While these prebuilt queries are helpful, they neglect vital areas, such as device-based attack paths and other hybrid identity connections. At the time of this research, BloodHound does not natively support paths involving devices, service principals, or

other critical Azure components. This limitation highlights the need for custom enhancements to fully map the hybrid attack surface.

### 3.2. Enriched Dataset

The enriched dataset, created by integrating custom PowerShell scripts and Neo4j's APOC library, addressed these limitations by introducing new nodes and relationships, allowing for the visualization and analysis of attack paths involving on-premises devices, Azure resources, and hybrid configurations.

#### 4.2.1. New Node

A new node within the Neo4j database was created to allow for the creation of specific attack paths, which will be discussed in the next section. This node was not captured in the original dataset.

##### **Node: HybridMachines**

**Description:** The HybridMachines node captures Azure Arc machines, which connect on-premises devices with Azure. They allow for running automation tasks through Azure Automation Account Runbooks, remote sessions via SSH with Azure command line tools, and other management tasks. The added node encapsulates critical properties such as DisplayName, IdentityPrincipalId, and AutomationAccountURL, which were crucial in creating a new relationship. An example of the new node can be seen in Figure 7.



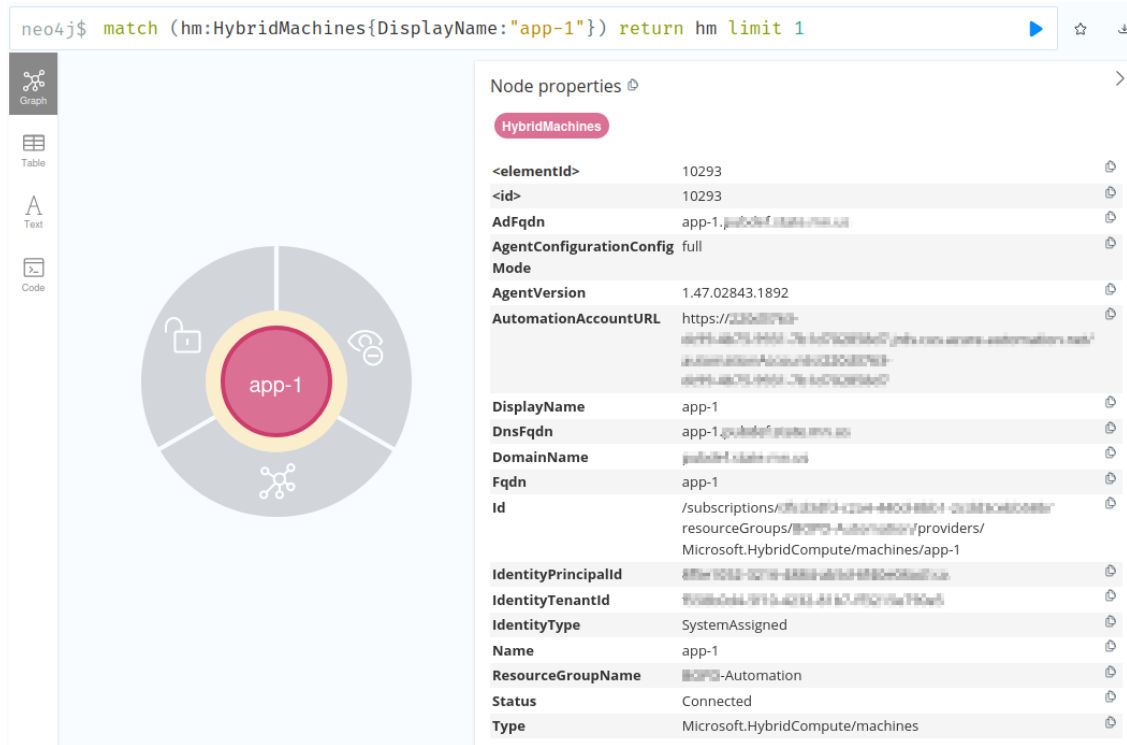


Figure 7: HybridMachine Node in Neo4j

**Significance:** This node was essential, as it facilitated a link between an Azure device and the Azure service principal created for each Azure Arc machine.

### 3.2.2. New Relationships

The enriched dataset also introduced new relationships that connected the newly added nodes and allowed for observing new hybrid attack paths. These relationships are:

### Relationship 1: OnPremToAzDevice

**Description:** This relationship links an on-premises device to its corresponding Azure Device, as seen in Figure 8.

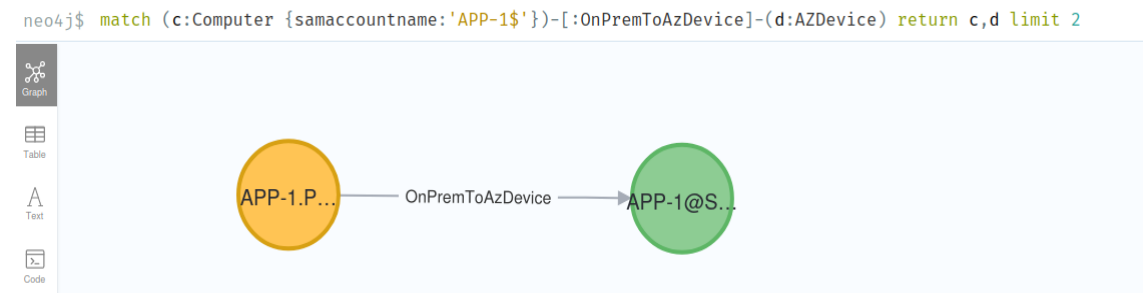


Figure 8: OnPremToAzDevice Relationship

**Significance:** This relationship is the foundational link for tracing attack paths from on-premises devices to Azure.

**Creation:** This relationship was created using the Active Directory objectguid attribute from an on-premises computer object and the Azure deviceid. Microsoft Entra Connect, by default, syncs these attributes to the Microsoft Entra ID environment.

### Relationship 2: AZDeviceToHybridMachine

**Description:** This relationship connects Azure devices to Azure Arc hybrid machines, and this relationship builds off the previously created HybridMachine node. An example of this relationship can be seen in Figure 9.

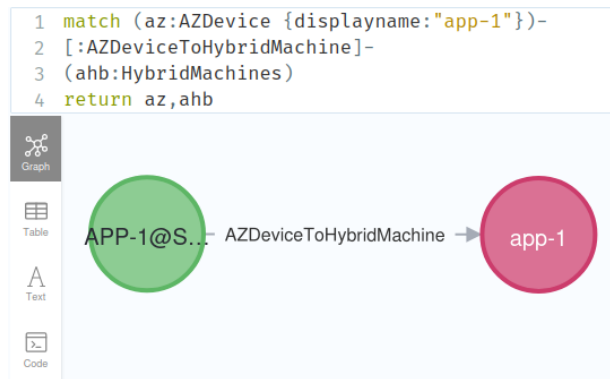


Figure 9: AZDeviceToHybridMachine Relationship

**Significance:** The link between the Azure device and the hybrid machine starts to reveal the true path and possibilities of hybrid attack paths in Azure.

**Creation:** Established by matching the displayname attribute between Azure devices and hybrid machines.

### Relationship 3: HybridMachineToAZServicePrincipal

**Description:** This relationship links hybrid machines to their corresponding Azure service principals, as seen in Figure 10.

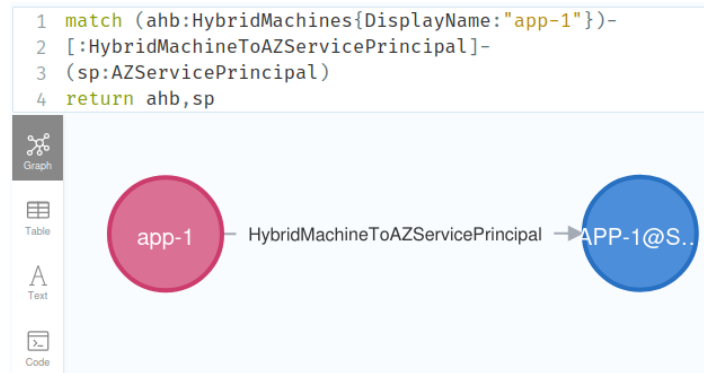


Figure 10: HybridMachineToAZServicePrincipal Relationship

**Significance:** Allows the exploration of paths where hybrid machines are a pivot point to compromise Azure service principals with privileged roles.

**Creation:** This relationship was created through a meaningful link between the hybrid machine IdentityPrincipalId property and the service principal objectid property.

### 3.2.3. New Attack Path Identified

#### Path 1: On-premises User to Azure Role through Hybrid Machine

**Description:** This attack path was created using the newly created HybridMachines node and relationships by following the flow below:

User → MemberOf → Group → AdminTo → Device → SyncedTo → AZDevice → SyncedTo → HybridMachine → SyncedTo → ServicePrincipal → AZHasRole → Azure Role

**Significance:** This path is significant because it answers the main research question. By creating a new node and multiple relationships, an attack path within a hybrid environment could be observed. The path can be seen in Figure 11.

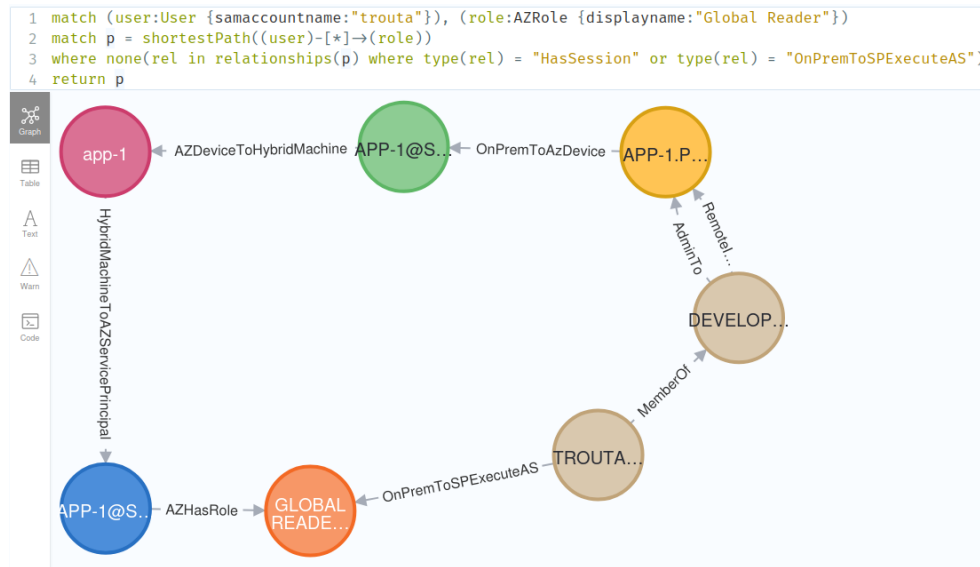
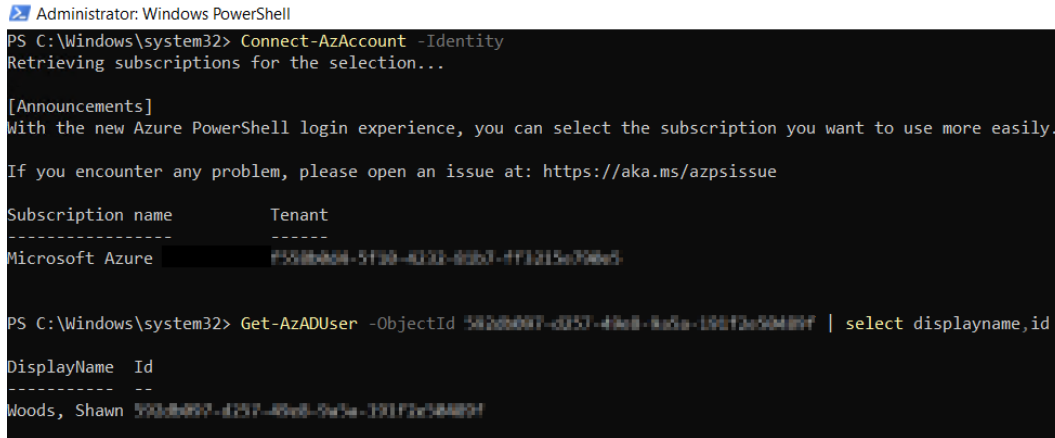


Figure 11: Shortest Path to AZRole through HybridMachine Node

**Implications:** This path is interesting because it highlights an attack path of how a user with administrative access to an on-premises device could exploit the Hybrid Instance Metadata Service (HIMDS) to get to an Azure role. This service manages the connection to Azure and the connected machine's identity (Microsoft, 2024b). There are many ways to use this service's authentication token to connect to Microsoft Azure or Entra ID. Some PowerShell modules allow using the managed identity directly. To access this service and retrieve a token, a user must be part of the local Administrators group or the Hybrid Agent Extension Applications group (Microsoft, 2024a).

Depending on the role's permissions, this could result in significant security risks, including lateral movement across the hybrid environment or tenant-level compromise. An example of this is seen in Figure 12. In the image, simply passing the identity parameter to Connect-AzAccount, an administrative user on app-1 can connect using the managed identity and retrieve information about Azure AD users since it has been granted the Global Reader role.



```

Administrator: Windows PowerShell
PS C:\Windows\system32> Connect-AzAccount -Identity
Retrieving subscriptions for the selection...

[Announcements]
With the new Azure PowerShell login experience, you can select the subscription you want to use more easily.
If you encounter any problem, please open an issue at: https://aka.ms/azpsissue

Subscription name      Tenant
-----
Microsoft Azure        7595b661-5f30-4233-b060-4f1215e700e5

PS C:\Windows\system32> Get-AzADUser -ObjectId 5820b007-d257-49a8-ba5e-1d6f2e50489f | select displayname,id

DisplayName Id
-----
Woods, Shawn 5820b007-d257-49a8-ba5e-1d6f2e50489f
    
```

Figure 12: Connecting to Azure Using the Managed Identity of a Hybrid Machine

This path also allows for the creation of a new relationship, as seen in Figure 12. The relationship OnPremToSPExecuteAS was created as a test to simplify all the steps in the path described above. As seen in the image, if this relationship is created, it can streamline the resulting graph exponentially.

However, some granularity is lost when doing this, as the full path is not outlined.

## 4. Recommendations and Implications

### 4.1. Recommendations

Based on the findings of this research, the following recommendations are proposed for practitioners and organizations managing hybrid cloud environments:

**Leverage Enhanced Graph-Based Tools:** Organizations should invest in tools or extend existing solutions like BloodHound with enriched datasets to comprehensively map out hybrid attack paths. This includes incorporating new nodes and relationships like HybridMachines and OnPremToAzDevice. The code presented in this paper may be used to gather the same data points and create the same nodes.

**Monitor and Manage Privileged Identities:** Continuous monitoring of privileged identities, including service principals and hybrid machine identities, is essential to prevent misuse and limit attack surfaces. Repeatedly collecting data and importing them

into attack path mapping tools can update the nodes as the environment changes. This is important so that the data is representative and does not stagnate.

**Harden Hybrid Instances:** Strengthen security around the HIMDS by restricting access to local administrator accounts and ensuring that roles and permissions are tightly scoped. Doing this ensures they are less likely to be used as pivot points for attackers.

**Adopt Least Privilege Principles:** Regularly audit permissions and adopt least privilege principles to minimize potential attack vectors. For instance, review and remove unnecessary Azure role assignments for hybrid identities.

**Integrate Graph Insights into Incident Response:** Incorporate graph-based visualizations into incident response plans to enable quicker identification and remediation of potential attack paths.

## 4.2. Future Research

Future studies should expand on this work because of the prevalence of hybrid environments. The scope of the research and time constraints limited the addition of other nodes and relationships to the graph database. However, this leaves many avenues for further investigation. Some ideas for future exploration are as follows.

**Broader Dataset Integration:** Extend the dataset to include additional relationships to other Azure components, such as Key Vaults, Storage Accounts, and Kubernetes clusters, to uncover further hybrid attack paths. Further Key Vault integration would be pertinent to this research, as the Key Vault service has a dedicated API for data plane operations that takes a bearer token (Robbins, 2024). A framework like BARK could use a hybrid machine with excessive permissions to read secrets, manage keys, or decrypt data by leveraging the token from HIMDS.

**New Tool Development and Use:** Custom tool creation based on this research to streamline the process of enriching and analyzing hybrid cloud environments would help tremendously. Further research in this domain should test the efficacy of tools like Microsoft's newly released Security Exposure Management in identifying hybrid attack paths.

## 5. Conclusion

This study confirms that it is possible to use graph theory to show meaningful links between on-premises and cloud environments. The additional data gathered enhanced a standard cybersecurity tool to highlight relevant attack paths in hybrid environments. Hybrid attack paths are becoming more prevalent. Microsoft has already observed threat actors leveraging hybrid attack paths. Threat actor group Storm-0501 compromised the Entra Connect service accounts to pivot from on-premises to the cloud for data exfiltration and ransomware deployment (Microsoft Threat Intelligence, 2024).

Organizations should proactively address these risks by leveraging attack path tools and tailoring data ingestion to meet their needs. Regularly analyzing tool outputs and applying appropriate remediations can strengthen defenses against hybrid attacks. Aligning these efforts with Zero Trust principles, such as enforcing strict identity verification and applying least-privilege access, can significantly enhance security. This strategy improves organizational resilience, bolsters regulatory compliance, streamlines incident response, and provides better protection against advanced persistent threats.

## References

- 2024 State of the Cloud Report*. (2024). Flexera. Retrieved from <https://resources.flexera.com/web/pdf/Flexera-State-of-the-Cloud-Report-2024.pdf>
- Baron, H., Leach, T., & John Yeoh. (2023, June 5). *State of Financial Services in Cloud*. Cloudsecurityalliance.org. Retrieved from <https://cloudsecurityalliance.org/artifacts/state-of-financial-services-in-cloud>
- Elmiger, M., Lemoudden, M., Pitropakis, N., & Buchanan, W. J. (2023). Start thinking in graphs: using graphs to address critical attack paths in a Microsoft cloud tenant. *International Journal of Information Security*, 23. <https://doi.org/10.1007/s10207-023-00751-6>
- Francis, N., Taylor, A., Green, A., Guagliardo, P., Libkin, L., Lindaaker, T., Marsault, V., Plantikow, S., Rydberg, M., & Selmer, P. (2018). Cypher: An Evolving Query Language for Property Graphs. *Proceedings of the 2018 International Conference on Management of Data - SIGMOD '18*. <https://doi.org/10.1145/3183713.3190657>
- Franklin, B. (2023, January 27). *40 cloud computing stats and trends to know in 2023*. Google Cloud Blog. Retrieved from <https://cloud.google.com/blog/transform/top-cloud-computing-trends-facts-statistics-2023>
- Google Cloud. (2024). *What is Hybrid Cloud?* Google Cloud. Retrieved from <https://cloud.google.com/learn/what-is-hybrid-cloud>
- Lambert, J. (2015, April 26). Defenders think in lists. Attackers think in graphs. As long as this is true, attackers win. Github. Retrieved from <https://github.com/JohnLaTwC/Shared/blob/master/Defenders%20think%20in%20lists.%20Attackers%20think%20in%20graphs.%20As%20long%20as%20this%20is%20true,%20attackers%20win.md>
- Lambert, J. (2021, November 21). *Defender's Mindset*. Medium. <https://medium.com/@johnlatwc/defenders-mindset-319854d10aaa>



Microsoft. (2023a, June 9). *Introduction to hybrid and multicloud*. Learn.microsoft.com. Retrieved November 27, 2024, from <https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/scenarios/hybrid/>

Microsoft. (2023b, November 6). *Microsoft Entra Connect: Prerequisites and hardware*. Learn.microsoft.com. Retrieved November 27, 2024, from <https://learn.microsoft.com/en-us/entra/identity/hybrid/connect/how-to-connect-install-prerequisites>

Microsoft. (2024a, September 19). *Authenticate against Azure resources with Azure Arc-enabled servers*. Microsoft.com. Retrieved November 27, 2024, <https://learn.microsoft.com/en-us/azure/azure-arc/servers/managed-identity-authentication>

Microsoft. (2024b, November 14). *Overview of the Azure Connected Machine agent*. Microsoft.com. Retrieved November 27, 2024, from <https://learn.microsoft.com/en-us/azure/azure-arc/servers/agent-overview>

Microsoft Threat Intelligence. (2024, September 26). *Storm-0501: Ransomware attacks expanding to hybrid cloud environments*. Microsoft Security Blog; Microsoft Threat Intelligence. <https://www.microsoft.com/en-us/security/blog/2024/09/26/storm-0501-ransomware-attacks-expanding-to-hybrid-cloud-environments/>

Neo4j. (n.d.-a). *Betweenness Centrality - Neo4j Graph Data Science*. Neo4j Graph Database Platform. Retrieved November 26, 2024, from <https://neo4j.com/docs/graph-data-science/current/algorithms/betweenness-centrality/>

Neo4j. (n.d.-b). *Dijkstra Source-Target - Neo4j Graph Data Science*. Neo4j Graph Database Platform. Retrieved November 26, 2024, from <https://neo4j.com/docs/graph-data-science/current/algorithms/dijkstra-source-target/>

OWASP. (2021). *OWASP Top 10: 2021*. OWASP. Retrieved November 27, 2024, from <https://owasp.org/Top10/>

- Phillips, R. L. (1977). A query language for a network data base with graphical entities. *ACM SIGGRAPH Computer Graphics*, 11(2), 179-185.  
doi:10.1145/965141.563891
- Reese, H. (2022, May 25). *Report: 82% of IT leaders are adopting the hybrid cloud*. TechRepublic. <https://www.techrepublic.com/article/report-82-of-it-leaders-are-adopting-the-hybrid-cloud/>
- Robbins, A. (2024, November 20). *Azure Key Vault Tradecraft with BARK*. Medium; SpecterOps. <https://posts.specterops.io/azure-key-vault-tradecraft-with-bark-24163abc8de3>
- SentinelOne. (2024, November 6). *Top 6 Hybrid Cloud Security Challenges*. SentinelOne. <https://www.sentinelone.com/cybersecurity-101/cloud-security/hybrid-cloud-security-challenges/>
- Vaidehi Joshi. (2017, March 20). *A Gentle Introduction To Graph Theory*. Medium. <https://medium.com/basecs/a-gentle-introduction-to-graph-theory-77969829ead8>
- Vailshery, L. (2024, September 17). *Hybrid cloud - statistics & facts*. Statista. Retrieved November 27, 2024, from <https://www.statista.com/topics/7914/hybrid-cloud/>

## Appendix A: Source Code for Original Docker File

<https://raw.githubusercontent.com/SpecterOps/BloodHound/refs/heads/main/examples/docker-compose/docker-compose.yml> contains the original Docker File used to deploy the BloodHound environment. The modified code used in this paper is below. Only the graph-db section was modified.

```
graph-db:
  image: docker.io/library/neo4j:4.4
  environment:
    - NEO4J_AUTH=${NEO4J_USER:-neo4j}/${NEO4J_SECRET:-bloodhoundcommunityedition}
    - NEO4J_dbms_allow_upgrade=${NEO4J_ALLOW_UPGRADE:-true}
    - NEO4JLABS_PLUGINS=["apoc"]
    - NEO4J_dbms_security_procedures_unrestricted=apoc.*
    - NEO4J_dbms_security_procedures_allowlist=apoc.*
    - NEO4J_apoc_import_file_enabled=true
    - NEO4J_apoc_export_file_enabled=true
    - NEO4J_dbms_memory_heap_initial_size=2G
    - NEO4J_dbms_memory_heap_max_size=8G
  ports:
    - 127.0.0.1:${NEO4J_DB_PORT:-7687}:7687
    - 127.0.0.1:${NEO4J_WEB_PORT:-7474}:7474
  volumes:
    - ${NEO4J_DATA_MOUNT:-neo4j-data}:/data
    - /home/swoods/Desktop:/import
  healthcheck:
    test:
      [
        "CMD-SHELL",
        "wget -O /dev/null -q http://localhost:7474 || exit 1"
      ]
    interval: 10s
    timeout: 5s
    retries: 5
    start_period: 30s
```