Cloud Infrastructure as Code (IaC) generation of SIEM and log collection infrastructure

Name	Szymon Kawecki
Student Number	C00286043
Course	Year 4, Bachelor (Hons) in SE606 Cybercrime & IT Security
Project Supervisor	Hisain Elshafi
Project Type	Student Specified Project

Table of Contents

Project Outline

The purpose of this project is to design a front-end graphical user interface (GUI) that guides an end user through a modular generation of Infrastructure as Code (IaC) for cloud-deployed resources for the back end of an industry-standard SIEM with additional log collectors.

Technologies

Below are the technologies and reasoning behind said technologies used in this project.

1. **Microsoft Azure**
   For reasoning see section: Cloud Provider – Amazon AWS or Microsoft Azure?
2. **Visual Studio Code**
   Simple and consistent IDE and Code Editor with a Marketplace featuring a vast number of extensions to aid with the development of this project including **Azure Functions**, **HashiCorp Terraform**, and more...
3. **Python (version 3.11)**
   This version is considered the most stable version, before 3.12 which is not yet fully supported nor endorsed by Microsoft use on the Azure platform. Python is a popular, high-level programming language favoured over other programming languages such as TypeScript for its human readability.
4. **Terraform**
   Terraform by HashiCorp is an open-source Infrastructure as Code (IaC) tool with multi-vendor support that allows for the creation of cloud and on-premises infrastructure.
   For reasoning see section: IaC Provider - Terraform or Azure ARM
5. **CustomTkinter**
   CustomTkinter is a powerful Python UI library used to create the Graphical User Interface (GUI) for this project. It is clean, professional and easily integrated with Python code and IaC modules such as Terraform.
   For reasoning see section: Graphical User Interface - CustomTkinter

Problem Statement

With the up and coming, ever-developing cyber threat to companies; it is almost crucial to staff at least 1 Cybersecurity personnel in each business above 50 employees. As per tradition, companies should seek to deploy a Security Information and Event Management (SIEM) in place (more on that in section: What is: a SIEM?).

The process of deploying a SIEM and its back-end infrastructure is seen as technically challenging and therefore companies usually must resort to hiring 3rd party companies to not only aid with deployment but handle the entire deployment. This service is usually costly but does however bring additional positive factors such as active monitoring of the infrastructure and more.

This sort of service also usually comes at a cost that not every company can afford, especially outside of the existing costs of running a SIEM and said infrastructure.

Target Audience

The purpose of this project, outside of the Brief Description seen above is to allow companies or private person(s) to generate Infrastructure as Code (more on that in section: What is: Infrastructure as Code (IaC)?) to help with this deployment, ideally without external assistance; to save costs and protect the hosts and data contained within said company.

Metrics

1. Success rate of the generation Infrastructure as Code (IaC)
    a. How many times does the generated Infrastructure as Code successfully deploy via the target cloud provider?
    Desired result: This metric should be very high.
    b. How modular in the generation? Can the end-user easily add or remove optional pieces of infrastructure to reach their target goal?
    Desired result: This metric should read Easy in the realm of modularity.
2. Resource accuracy. How often does the infrastructure as Code generate the true, requested resources?
    Desired result: Very high percentage metric. The app should effectively almost never generate inaccurate resources or configuration.
3. How often if at all does the Graphical User Interface crash?
    Desired result: This metric should be none or minimal.
4. take to manually deploy said infrastructure.
5. How easy it is to read the code?
    Desired results: High readability to therefore support manual changes by the end-user or translation into another cloud infrastructure provider.

Precedents

1. Cloud Curate by Quali (https://www.quali.com/cloud-curate/ is a product available to aid with the generation of Infrastructure as Code (IaC). This platform lets you select specific resources based on the cloud provider and configure as you go.

   How does this compare? Although this is a Infrastructure as Code generator, it does not provide a simple, modular blueprint for deployment the back-end infrastructure of a SOC without a sizeable learning curve and custom configuration by the client, as well as designing the architecture by themself. The product is also behind a paywall which means more costs for the end-user.

2. SketchTF or Sketch Terraform (https://www.cloudsketcher.com/sketch-TF.html) is a platform that translates architectural diagrams into Infrastructure as Code via Terraform.

   How does this compare? Although this is a solution for non-tech-savvy end-users; it merely mirrors the required infrastructure and not the configuration. There are no attempts made at importing any sort of infrastructure configuration to allow for the infrastructure to properly function, as well as no blueprint – which once again requires a learning curve to design.

3. AWS CDK (https://docs.aws.amazon.com/cdk/v2/guide/home.html or Amazon Web Services Cloud Development Kit allows developers to write Infrastructure as Code in another, high-level language such as C++ which can then be translated across the board to other languages such as Hashicorp Language for Terraform.

   How does this compare? Steep learning curve; end-user needs experience using a high-level language followed by the infrastructure required behind a SOC/SIEM to build code that could be reliably translated.

Mandatory Deliverables

1. Graphical User Interface (GUI), that will act as the primary interaction layer between the end-user and the system. Main function is to provide a modular, intuitive, and user-friendly front-end interface that abstracts away the technical complexity of the IaC generator.

2. The back-end generation of Infrastructure as Code (IaC), which will be the back-end core logic that translates the end-user's selection from the GUI into IaC, written in Hashicorp Language using Terraform.

3. Deploy all the back-end infrastructure and including the configuration of the SIEM (Microsoft Sentinel in this case) and all other required resources to make the system operational.

4. An additional, non-native log collector via Azure Function Apps (more on this in section: Architectural layout (Infrastructure)), as this could be expanded into a library of said non-native collectors in the future for the end user to pick from.

Exceptional Deliverables

1. Project is fully functional; can generate Infrastructure as Code (IaC) without any internal errors and can be deployed on the target cloud provider without issues.
2. Additional log collector and example setup of Windows Security Events (Winevent).

Development plan

Due dates

Project specification is due on the **24th of October 2025**.

Research document is due on the **5th of December 2025**.

Presentation is due on the **5th of January 2026**.

Final product and report are due on the **24th of April 2026**.

Development phases

Development phase 1:

The first phase of development will be 9 weeks long (included but excluding holidays) and occur from the **5th of December 2025** to the **9th of February 2026**. This phase will cover the development of the additional, non-native log collector as well as the pieces of Infrastructure as Code (IaC) for all the infrastructure required for this project, including configurations.

Development phase 2:

The second phase of development will be 5 weeks long and will last until the **16th of March**. This phase will cover the graphical user interface (GUI) portion of this project. It will involve the full back-end creation of the IaC generator as well as the GUI, including design.

Development phase 3:

The last phase of development will be 4 weeks long and will last until the **6th of April**. This phase will feature thorough testing and ensuring the stability of the developed product as well as the Exceptional Deliverables including ensuring it is fully exceptional and attempting to complete the task of developing a log collector for Windows Security Events (Winevent).

Components of the project

Security Information and Events Management (SIEM)

A SIEM is an on-premises or cloud-based platform that gathers and stores log data from several systems, usually security logs. It makes compliance reporting and long-term analysis easier. The platform's main strength is its real time monitoring and analysis of events, along with its ability to correlate events and produce alerts to identify suspicious activity or attacks with the use of analytics (detections, or patterns that the SIEM looks for).

Security Operations Center (SOC)

A Security Operations Center (SOC) is typically a company's department, or external party, in charge of tracking, identifying and evaluating cybersecurity issues. Think of it as a 24/7 continuous monitoring (based on staffing) where security analysts surveil over al digital activity. It typically involves continuous monitoring through a SIEM (see the correlation here?), threat detection, incident response, and gathering threat intelligence from existing logs and incidents to create a way to anticipate and protect against future attacks.

Although this project does not directly provide a SOC (unfortunately, we cannot generate that!), it does provide the foundation structure for any company to bring up a SIEM and task employees with the monitoring and further development of their security system.

Infrastructure as Code (IaC)

Infrastructure as Code (IaC) is a modern practice that allows for the management and provisioning of IT infrastructure, in this case to deploy a SIEM and its backend infrastructure, using code instead of a manual process. This also allows it to remain

modular and be adapted throughout multiple different cloud providers (multi-vendor support). It allows for creating automated infrastructure deployments, consistency as it eliminates human error in configuration, version control, scalability, and disaster recovery as it also acts like a backup in some capacity.

Cloud Provider – Amazon AWS or Microsoft Azure?

Amazon Web Services (AWS) is the “world’s most comprehension and broadly adopted cloud.” It has over 120 Availability Zones with 38 Geographic Regions. AWS offers the clients the opportunity to build and host cloud infrastructure in the datacentres with a unique panel.

Microsoft Azure is a cloud platform used for building, deployment, and management of cloud solutions and infrastructure. It has over 300 datacentres in more than 60 regions worldwide. The platform also features an online platform for management with in-built security.

Both platforms became a pivotal shift from on-premises datacentres to cloud computing. AWS is the oldest and most mature cloud provider in the world with the largest global market share and customer base, which means it is sometimes considered more stable and battle-tested with many third-party integrations.

However, Microsoft Azure is the preferred choice in this project as it provides its own native SIEM offering, namely Microsoft Sentinel which will allow this project to cover the IaC deployment of a SIEM, not only its infrastructure. It is better for most organizations as they typically already use products such as 365, Defender (Windows Antivirus, this is a huge factor!), Intune, Entra ID and more. They have built in SOAR (Security Orchestration, Automation and Response) playbooks which are used for building SIEM alerts for Microsoft Sentinel.

IaC Provider – Terraform or Azure ARM

Microsoft Azure allows you to deploy infrastructure and update configurations by sending JSON-formatted messages featuring instructions directly or by inputting them into the ARM-based template deployer in the Microsoft Azure portal.

Terraform on the other hand, allows you to create Infrastructure as Code (IaC) directly, as actual code. This means that you can use the Hashicorp Language (HCL) and for example, the AzureRM provider which provides a systematic code template for every resource on the Azure platform.

The positive thing about Terraform, as opposed to directly sending Microsoft Azure RM (ARM) templates, is that there is a higher level of state-tracking. This means that Terraform keeps a Terraform “state” (.tfstate) file that allows you to continue

modifying configuration, update, add and delete infrastructure without overwriting critical infrastructure. Anything deployed via Terraform will continue to be managed by terraform but can also be managed by logging into the Microsoft Azure portal.

Overall, Terraform is the solution we are looking for here and will allow for multi-vendor support as the code itself can be adjusted to fit most other cloud providers Hashicorp Language templates. HCL is also much more readable compared to the JSON format ARM templates provided by Microsoft and other cloud providers.

Terraform is highly modular while ARM templates are less intuitive and prefer nested templates. Terraform also automatically calculates resource dependencies, a comprehensive testing “terraform plan” command that allows you to see what changes you are pushing without pushing them, and a huge community based eco-system of assistance and modules.

Graphical User Interface – CustomTkinter

The development for the front-end Graphical User Interface (GUI) in the project will occur with CustomTkinter. CustomTkinter looks modern, clean and uses Python only (no web or JavaScript). It is much easier to learn than PyQt but more advanced than standalone Tkinter. It allows you to build front-end features such as buttons, dropdowns, tabs, frames etc but also allows you to add back-end logic easily! It integrates and connects easily to Python modules to generate IaC such as Terraform. It works cross-platform on Windows, macOS and Linux with no additional changes. Other GUI modules such as PyQt6 as more powerful and professional but have a much steeper learning curve and can lead to many more errors.

You can see documentation on CustomTkinter here:
https://customtkinter.tomschimansky.com/

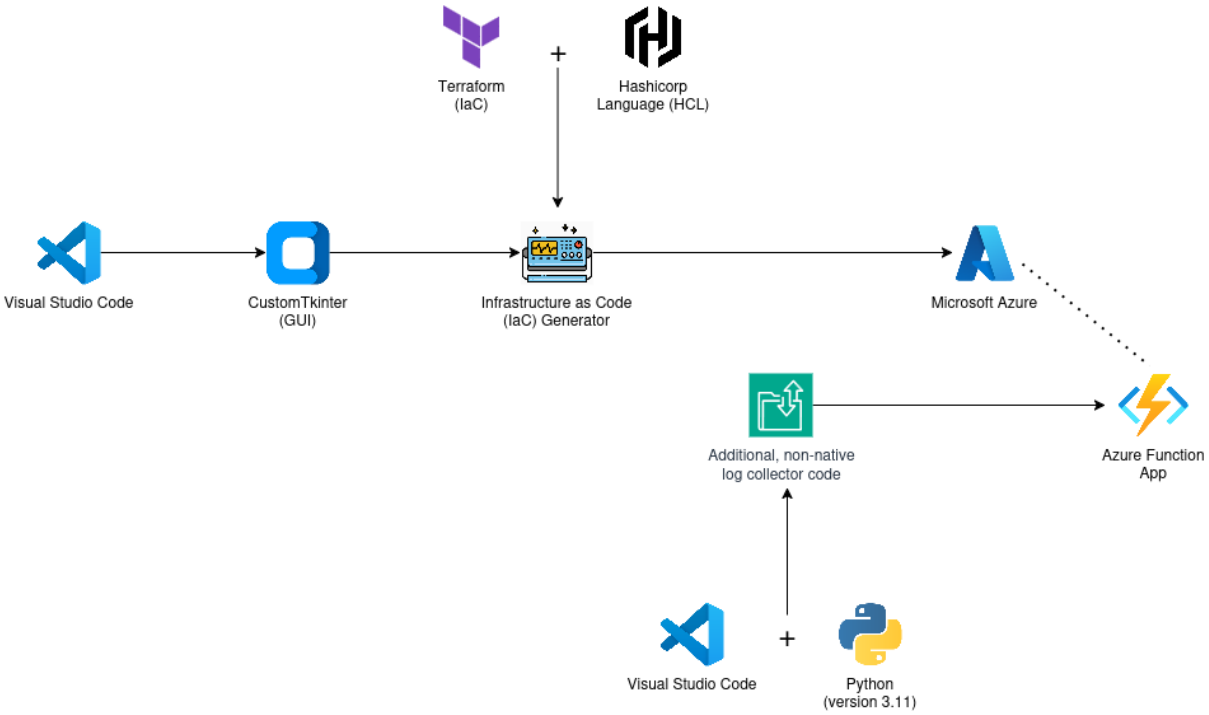
Technical layout

As mentioned above, there are multiple different technical elements that go into this project.

1. Graphical User Interface (GUI) via CustomTkinter.
2. Back-end Infrastructure as Code (IaC) generator using Terraform with Hashicorp Language (HCL) and the AzureRM provider.
3. Additional, non-native log collector:
    a. Via Azure Functions Apps, see next section.
    b. Using Python version 3.11.

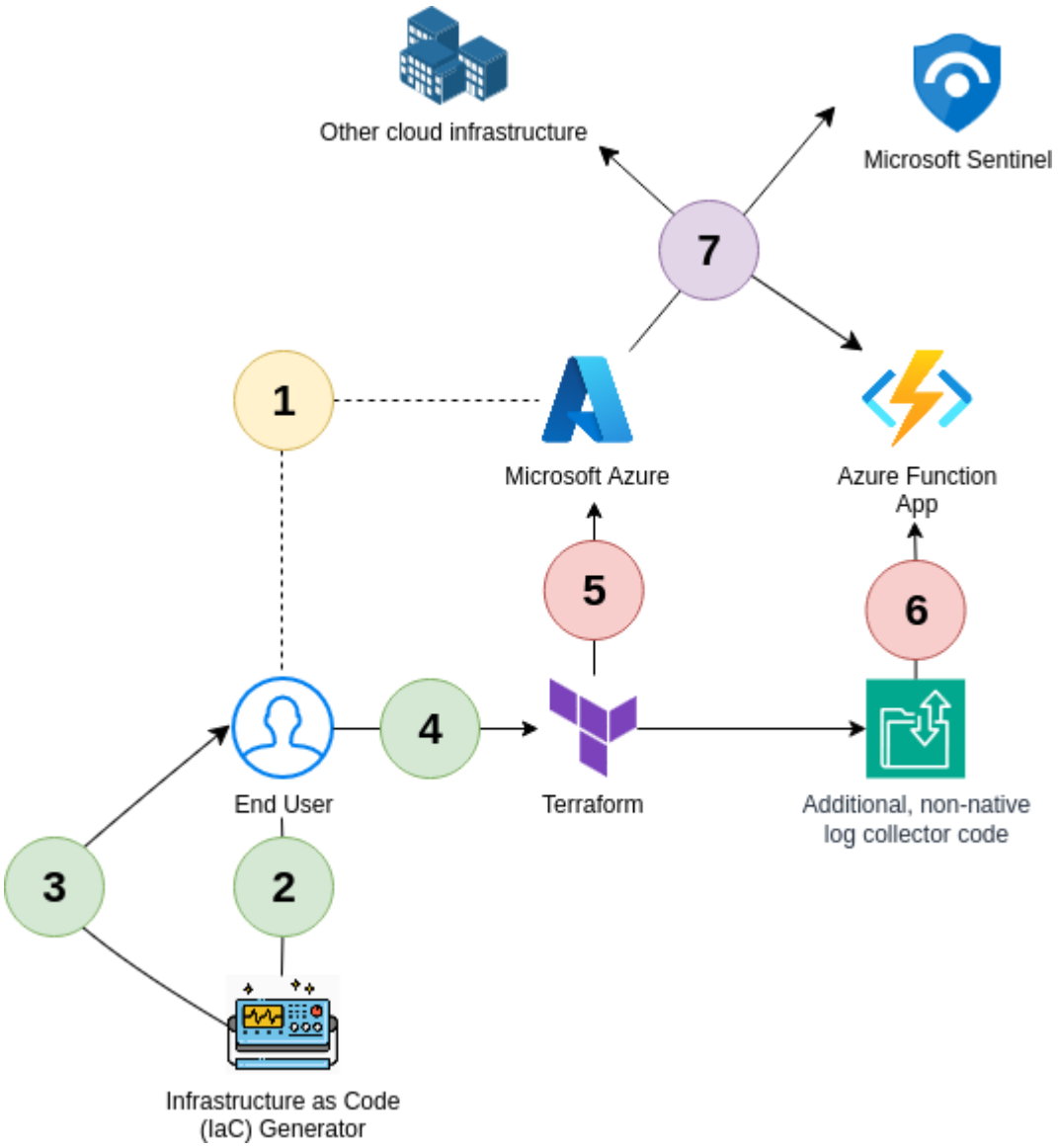
Below is an architectural diagram of what this setup is intended to look like. Note, this diagram does not include all the Azure resources for this project which are covered in the next section.

There will also be security considerations made during the research document to determine the best, possible way to isolate the log traffic coming into the Azure Function App to prevent it being infiltrated. Other security considerations will also be considered depending on whether the Exceptional Deliverables are met, more specifically the ingestion of Windows Security Events. This will include but not be limited to keeping everything in a virtual network, load balancing, isolating inbound and outbound source/destination IPs as well as ports, and other considerations such as (a) Network Address Translation gateway(s).



Use case

This section depicts a diagram which was constructed to show how the end user will use this project for its intended purpose.



1. End User authenticates with Microsoft Azure using AzCLI.

2. End User interacts with the IaC Generator to generate desired infrastructure.

3. Terraform code output is given to the user.

4. End User uses simple Terraform commands to deploy the IaC.

5. Collector code automatically available, if selected, with the IaC.

6. Log collector uploaded (if selected).

7. All selected and other cloud infrastructure is deployed, e.g. Microsoft Sentinel (SIEM).

Conclusion

One important step toward making the deployment of complicated cybersecurity infrastructure in cloud environments simpler is the creation of the Cloud Infrastructure as Code (IaC) Generation of SIEM and Log Collection Infrastructure project. With the use of a Python-based CustomTkinter graphical user interface (GUI) and Terraform's modular flexibility, this project seeks to close the gap between technical complexity and real-world usability. This solution will make it possible for companies, particularly those with limited resources or a lack of cybersecurity staff, to set up scalable, dependable SIEM and log collection systems on Microsoft Azure without extra technical knowledge on the matter.

Organizations can accomplish version-controlled and repeatable deployments, minimize disturbance to their infrastructure as it changes, and avoid human configuration errors by implementing Infrastructure as Code. The system's value is further increased by the addition of more non-native log collectors and the possibility of adding more features, which gives the system flexibility for future integration with a variety of data sources outside of the Azure ecosystem.

To guarantee functionality and accessibility, I made several important design and technological choices. Microsoft Azure was chosen as the cloud provider due to its native SIEM solution, Microsoft Sentinel which allows for direct and seamless integration of security infrastructure. Terraform was selected as the Infrastructure as Code (IaC) provider for its modularity, readability, and ability to manage infrastructure state effectively. These are features that make it superior to Azure ARM templates for this use case.

The GUI is being created with CustomTkinter because of its simple design and native Python integration, while Python 3.11 was chosen for its reliability and Azure compatibility. When combined, these choices produce a modular, intuitive architecture that lets users effectively build, implement, and maintain SIEM infrastructure without requiring extensive knowledge of the cloud.

In conclusion, this project offers a useful, cost-effective, and instructive architecture for the deployment of automated SIEM infrastructure. It illustrates how up-and-coming cybersecurity professionals may use current approaches, especially Infrastructure as Code, to boost defensive capabilities and operational effectiveness in a threat landscape that is changing quickly.

Developing more log ingestion modules, expanding multi-cloud compatibility, and investigating increased automation via AI-assisted configuration generation are possible future projects. When combined, these improvements have the potential to turn this prototype into a complete platform for the deployment of secure, scalable, and easily accessible cloud security.

References

1. Microsoft, (n.d.) What is Azure? Microsoft Corporation. Available at: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-azure (Accessed: 8 October 2025).
2. Microsoft, (2025) Visual Studio Code. Microsoft Corporation. Available at: https://learn.microsoft.com/en-us/shows/visual-studio-code/ (Accessed: 13 October 2025).
3. Python, (n.d.) What is Python? Executive Summary. Python Software Foundation. Available at: https://www.python.org/doc/essays/blurb/ (Accessed: 5 October 2025).
4. Developer Service, (2025) CustomTkinter - A Complete Tutorial. DEV Community. Available at: https://dev.to/devasservice/customtkinter-a-complete-tutorial-4527 (Accessed: 10 October 2025).
5. Microsoft, (n.d.) What is SIEM? Microsoft Corporation. Available at: https://www.microsoft.com/en-us/security/business/security-101/what-is-siem (Accessed: 12 October 2025).
6. Scapicchio, M. et al., (n.d.) What is a security operations centre (SOC)? IBM. Available at: https://www.ibm.com/think/topics/security-operations-center (Accessed: 9 October 2025).
7. RedHat, (2025) What is Infrastructure as Code (IaC)? RedHat. Available at: https://www.redhat.com/en/topics/automation/what-is-infrastructure-as-code-iac (Accessed: 6 October 2025).
8. Amazon, (n.d.) Why AWS? Amazon Web Services. Available at: https://aws.amazon.com/what-is-aws/ (Accessed: 11 October 2025).
9. Hashicorp, (n.d.) What is Terraform? Hashicorp. Available at: https://developer.hashicorp.com/terraform/intro (Accessed: 7 October 2025).
10. Microsoft, (2025) What is Azure Resource Manager? Microsoft Corporation. Available at: https://learn.microsoft.com/en-us/azure/azure-resource-manager/management/overview (Accessed: 14 October 2025).
11. Amazon Web Services, (n.d.) What is the AWS CDK? Amazon Web Services. Available at: https://docs.aws.amazon.com/cdk/v2/guide/home.html (Accessed: 5 October 2025).