Session 1

Data Loss Prevention Systems: A Concise Analysis

1. Defining Data Loss Prevention (DLP) Systems

Data Loss Prevention (DLP) solutions utilize tools and processes to detect and block risky sharing, movement or misuse of sensitive data. Protection and monitoring of information systems occur across both on-premises infrastructures and cloud environments along with endpoint devices. The primary objective of these systems is to identify and control unauthorized access or data leakage which functions as an essential component of an all-encompassing security plan that addresses breaches and both intentional and unintentional data leaks. Data Loss Prevention (DLP) shields data from theft and misuse by monitoring information in various states and applying cybersecurity techniques and technology to prevent unauthorized transfer of data. DLP solutions according to Gartner are systems that both categorize data content and implement protection measures through policy enforcement such as logging and encrypting. The NIST perspective focuses on stopping data from exiting corporate limits yet fails to fully address the requirements of today's cloud-based and remote work settings. The use of DLP systems protects organizations from reputational harm and financial damage while avoiding regulatory fines imposed by GDPR, HIPAA, and PCI DSS regulations and safeguards PII and intellectual property through enhanced visibility over data management within complex digital environments.

2. How DLP Systems Monitor Sensitive Data

DLP systems monitor data differently depending on its state: in use, in motion, or at rest. DLP systems protect data in use by implementing user authentication measures and access controls while simultaneously monitoring and flagging unauthorized activities that include copying sensitive data. Endpoint DLP tracks user behaviors such as file transfers and clipboard activities on workstations through deep content analysis combined with AI and machine learning which allow it to evaluate user actions against established policies for detecting suspicious activity based on both content and context. DLP solutions ensure data protection during transmission across networks by encrypting data and monitoring network traffic for sensitive patterns in real-time. Network DLP systems focus on crucial network points such as email and web gateways while they analyze packets through deep packet inspection and regular expressions alongside AI capabilities to detect confidential information and unusual activity patterns in corporate and cloud networks. Data Loss Prevention systems control access and authentication for data stored in databases and cloud environments while scanning for sensitive information and implementing security measures through labeling and access controls including encryption and deletion. Storage DLP employs data matching techniques together with label detection and keyword matching to perform pattern recognition for identifying policy breaches such as unauthorized access or copying of sensitive files thereby protecting confidentiality and integrity.

3. Mechanisms for Detecting Sensitive Data

Sensitive data detection in DLP systems operates through multiple mechanisms. Content inspection analyzes actual data content for patterns or keywords such as social security numbers and "confidential" while contextual analysis looks at additional factors including user identity and location through metadata to enforce dynamic policy application. AI combined with machine learning transforms simple text matching systems into intelligent platforms that increase accuracy while minimizing false positives during large-scale data analysis. By employing regular expressions and dictionaries pattern matching becomes essential for detecting structured data such as credit card numbers and specific ID formats through predefined text patterns or lexicons of sensitive words. Modern DLP systems use machine learning to adapt to data patterns over time which results in better sensitive data identification and automation of data classification while minimizing false positive results. DLP policies become adaptive through machine learning analysis of user behavior and system events which identifies anomalies that indicate data exfiltration or insider threats.

DLP Type	Area of Focus	Key Monitoring Aspects	Primary Blocking Mechanisms	Benefits
Network DLP	Data in transit	Network traffic, communication channels	Blocking unauthorized transfers, encryption	Prevents exfiltration, secures network flow
Endpoint DLP	Data on devices	User activity, file operations	Blocking copying/printing/uploads, encryption	Protects data on individual devices
Storage DLP	Data at rest	Stored data, access logs	Encryption, access control, deletion	Secures data in storage repositories
Cloud DLP	Data in cloud	Cloud storage, applications, user activity	Blocking sharing, encryption, masking	Protects data in cloud environments

4. Blocking Unauthorized Actions on Sensitive Data

The essential task of DLP systems involves preventing unauthorized actions by applying established rules and policies that control data behavior. Organizational guidelines and regulations become technical controls through these policies which determine the handling, accessing, and sharing of sensitive data. To enforce security measures organizations implement access controls along with encryption and data masking while preventing unauthorized copying or transmission of data and responding to policy violations. Network DLP protects sensitive information by monitoring network traffic to prevent unauthorized data from being sent through email or USB by blocking transmissions that violate organizational policies. Endpoint DLP limits device user actions by preventing copying to USB drives and printing confidential documents while blocking uploads to unauthorized services and has the ability to perform data transfer encryption or access revocation whenever a policy breach occurs. Storage DLP utilizes access controls and encryption to manage data at rest which prevents unauthorized data storage and access in repositories. Within cloud environments Cloud DLP manages access and usage

while scanning and encrypting data before storage and blocks transmissions or masks data when policy violations occur. Some security solutions provide flexible options by permitting warnings and legitimate user exceptions instead of direct prohibitions to maintain both safety and efficiency.

5. Types of DLP Solutions and Their Focus

DLP solutions function by focusing on distinct environments as well as different states of data. Network DLP solutions protect data while it moves through networks by examining traffic going through email systems, web connections and cloud applications and prevent unauthorized data sharing through AI-driven detection of abnormal behaviors. Endpoint DLP safeguards information on personal devices including laptops and mobiles through active monitoring of user operations and blocking unauthorized USB transfers while delivering protection independent of network connection. Repositories like servers and databases alongside cloud storage hold data at rest which storage DLP protects by scanning sensitive content and then implementing encryption controls along with access management and data classification. Cloud DLP protects data within cloud services by scanning uploads prior to transfer while tracking authorized applications and users, managing access permissions and sharing capabilities while alerting organizations of policy breaches specific to cloud systems. The combination of these security types gives an integrated layered defense for data protection throughout its entire lifecycle.

6. Conclusion: The Significance and Implementation of DLP

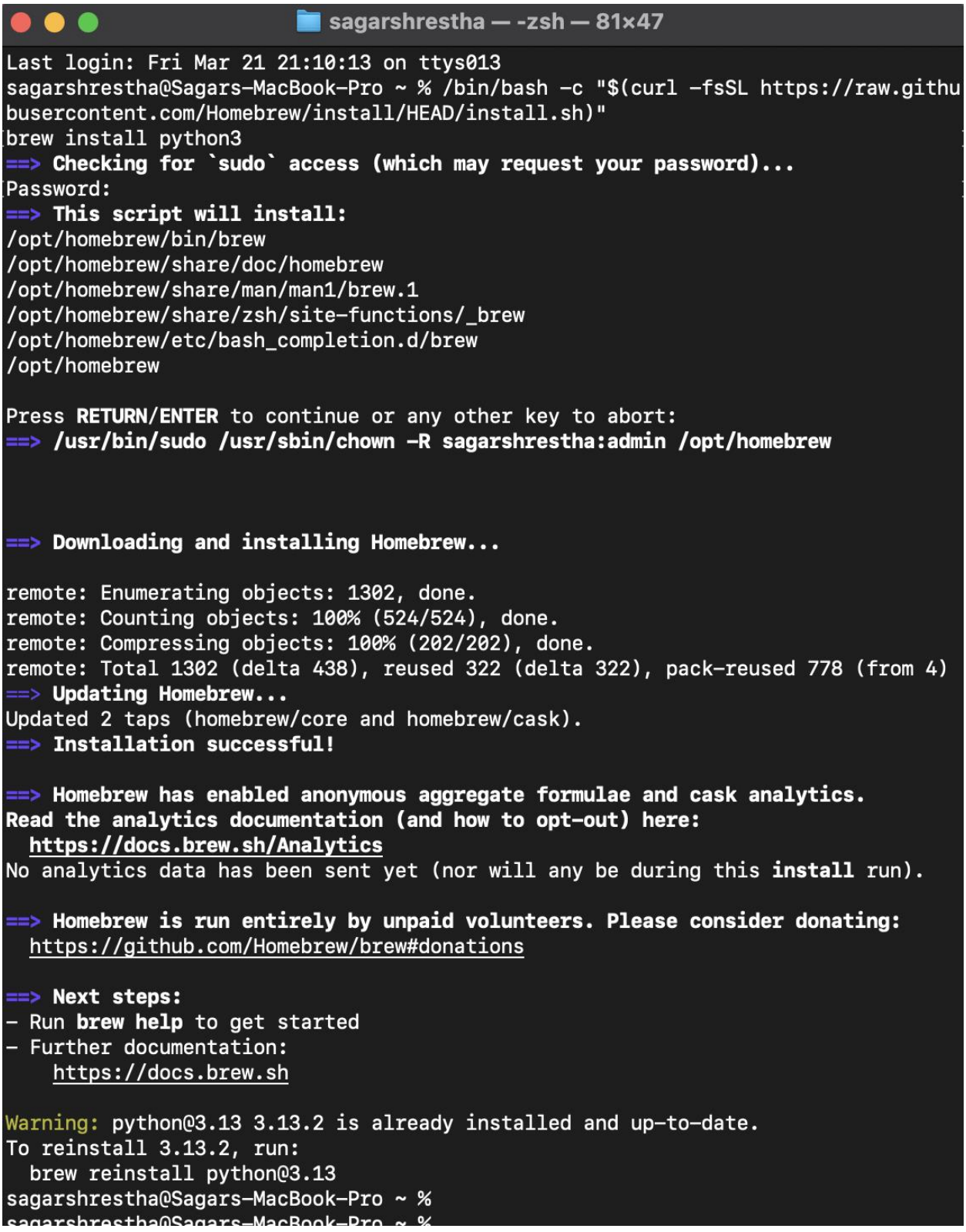
DLP systems function as vital security measures to prevent unauthorized access and misuse of confidential data as well as its unauthorized transmission. Organizations protect their reputation and meet regulatory standards by utilizing content inspection, pattern matching, and machine learning to monitor data that is active, moving, and stored. DLP systems maintain security by using network, endpoint, storage, and cloud solutions to stop unauthorized activities. Effective DLP implementation demands thorough vendor evaluation and requirement definition while establishing clear roles and data classifications alongside strong policy development and access control systems. Organizations that learn and apply DLP strategies will protect their essential data assets in today's complicated digital environment.

My device info

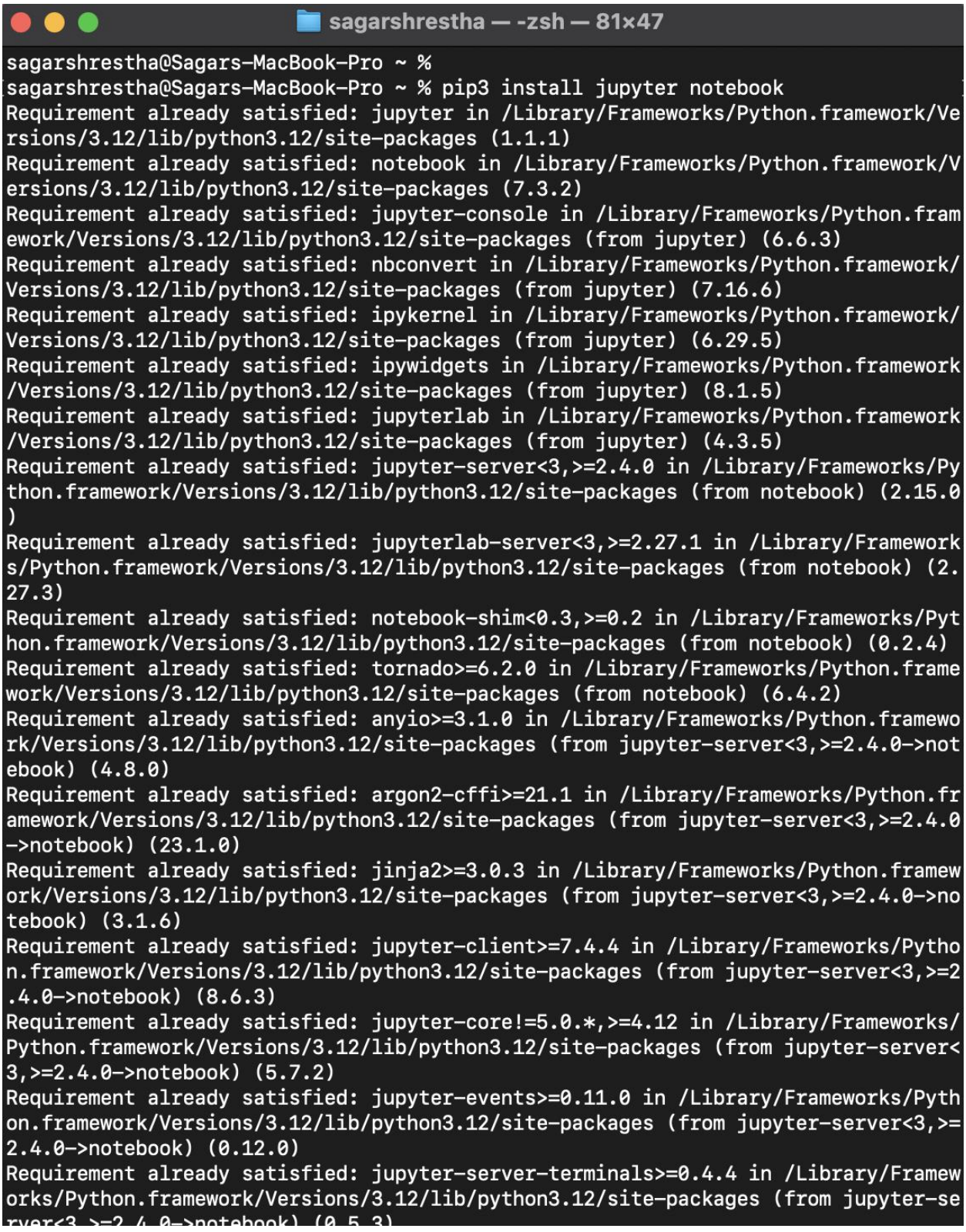
Session 2

Setting up Python in my Mac with homebrew

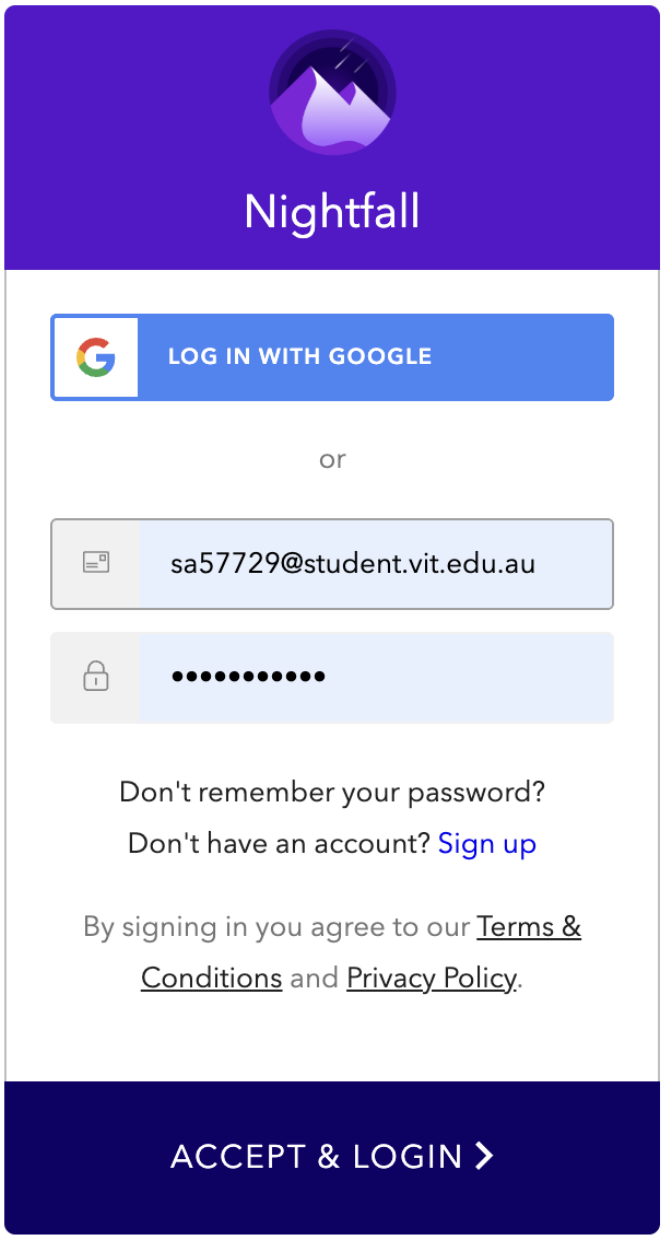
```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
brew install python3
```



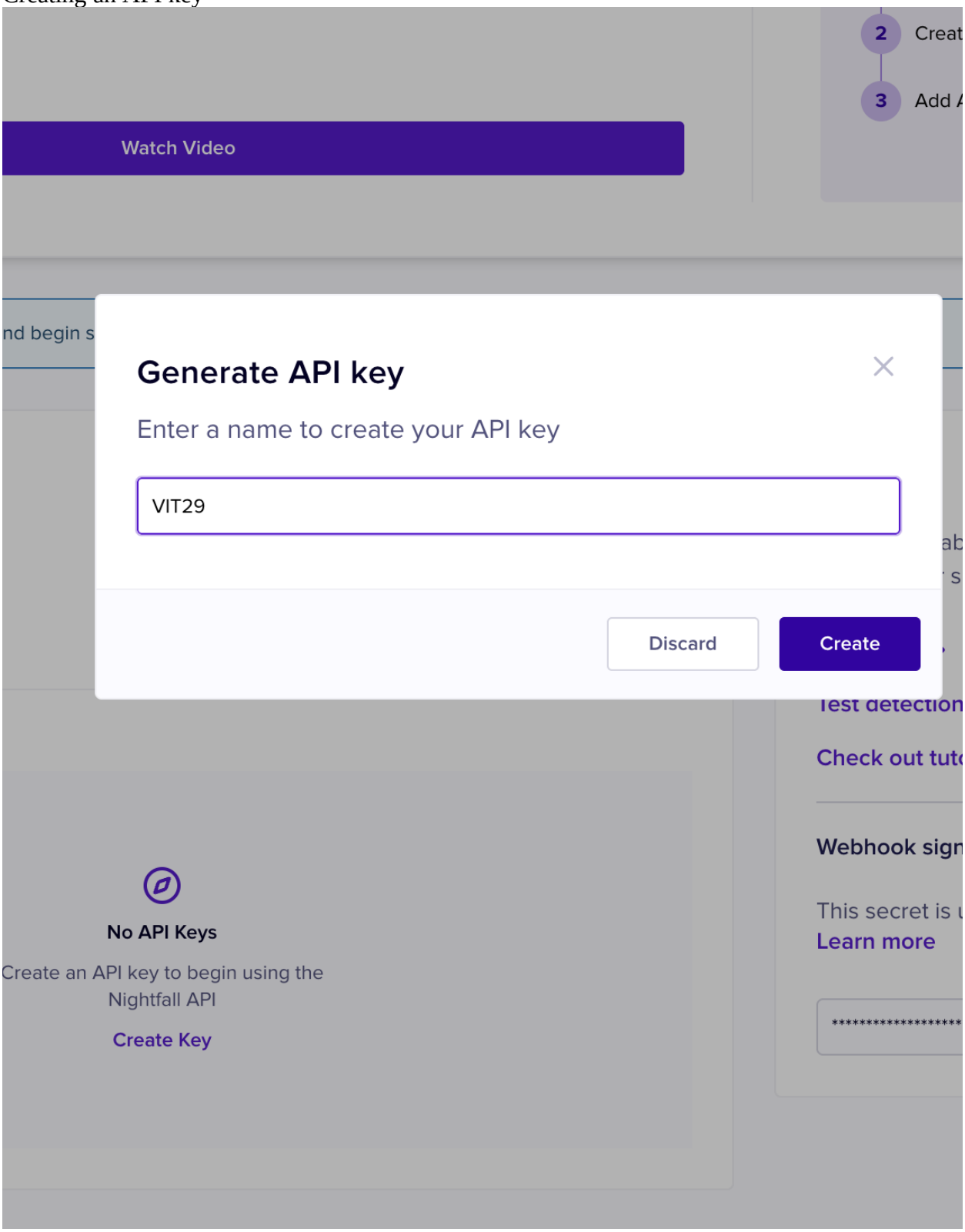
Installing Jupyter Notebook for python


sagarshrestha — -zsh — 81×47
sagarshrestha@Sagars-MacBook-Pro ~ %
sagarshrestha@Sagars-MacBook-Pro ~ % pip3 install jupyter notebook
Requirement already satisfied: jupyter in /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages (1.1.1)
Requirement already satisfied: notebook in /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages (7.3.2)
Requirement already satisfied: jupyter-console in /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages (from jupyter) (6.6.3)
Requirement already satisfied: nbconvert in /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages (from jupyter) (7.16.6)
Requirement already satisfied: ipykernel in /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages (from jupyter) (6.29.5)
Requirement already satisfied: ipywidgets in /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages (from jupyter) (8.1.5)
Requirement already satisfied: jupyterlab in /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages (from jupyter) (4.3.5)
Requirement already satisfied: jupyter-server<3,>=2.4.0 in /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages (from notebook) (2.15.0)
Requirement already satisfied: jupyterlab-server<3,>=2.27.1 in /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages (from notebook) (2.27.3)
Requirement already satisfied: notebook-shim<0.3,>=0.2 in /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages (from notebook) (0.2.4)
Requirement already satisfied: tornado>=6.2.0 in /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages (from notebook) (6.4.2)
Requirement already satisfied: anyio>=3.1.0 in /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages (from jupyter-server<3,>=2.4.0->notebook) (4.8.0)
Requirement already satisfied: argon2-cffi>=21.1 in /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages (from jupyter-server<3,>=2.4.0->notebook) (23.1.0)
Requirement already satisfied: jinja2>=3.0.3 in /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages (from jupyter-server<3,>=2.4.0->notebook) (3.1.6)
Requirement already satisfied: jupyter-client>=7.4.4 in /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages (from jupyter-server<3,>=2.4.0->notebook) (8.6.3)
Requirement already satisfied: jupyter-core!=5.0.*,>=4.12 in /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages (from jupyter-server<3,>=2.4.0->notebook) (5.7.2)
Requirement already satisfied: jupyter-events>=0.11.0 in /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages (from jupyter-server<3,>=2.4.0->notebook) (0.12.0)
Requirement already satisfied: jupyter-server-terminals>=0.4.4 in /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages (from jupyter-server<3,>=2.4.0->notebook) (0.5.3)

Now loggin on to my nightfall account



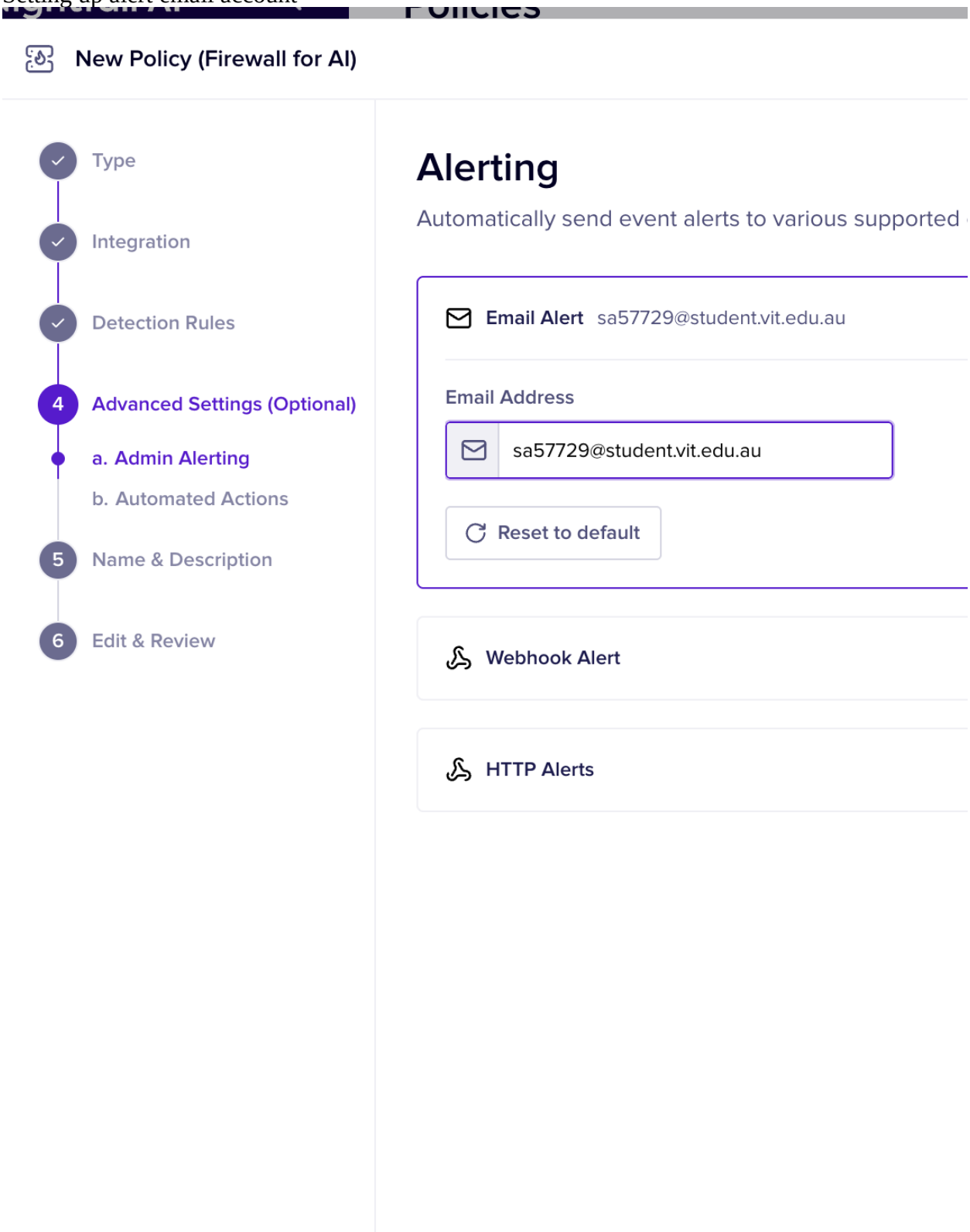
Creating an API key



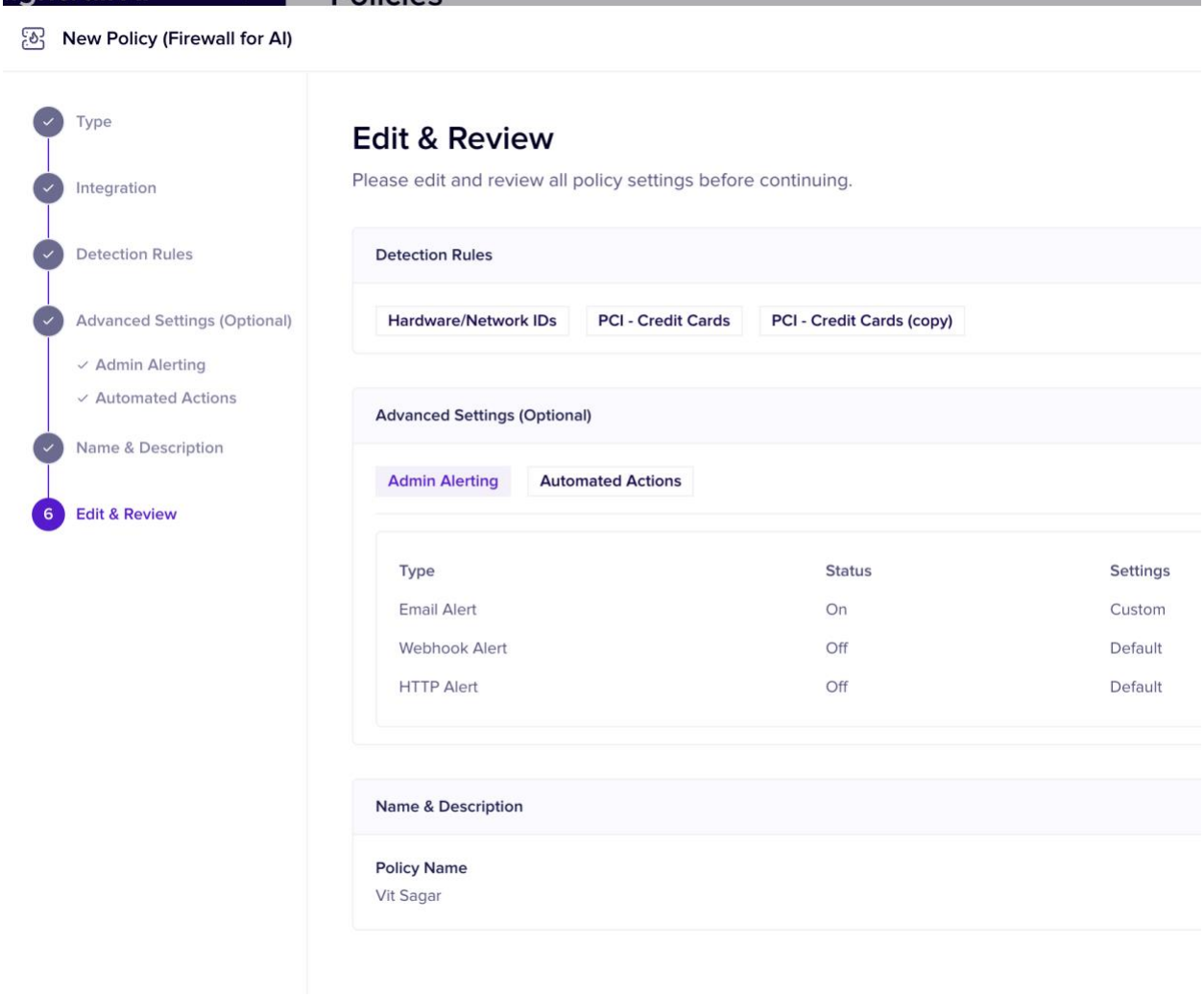
KEY

NF-zHGeEYUn5CYwZ4n1ufH1nFi8at7dVWuU

Setting up alert email account



Policy and detection rules set as network addresses and credit card numbers



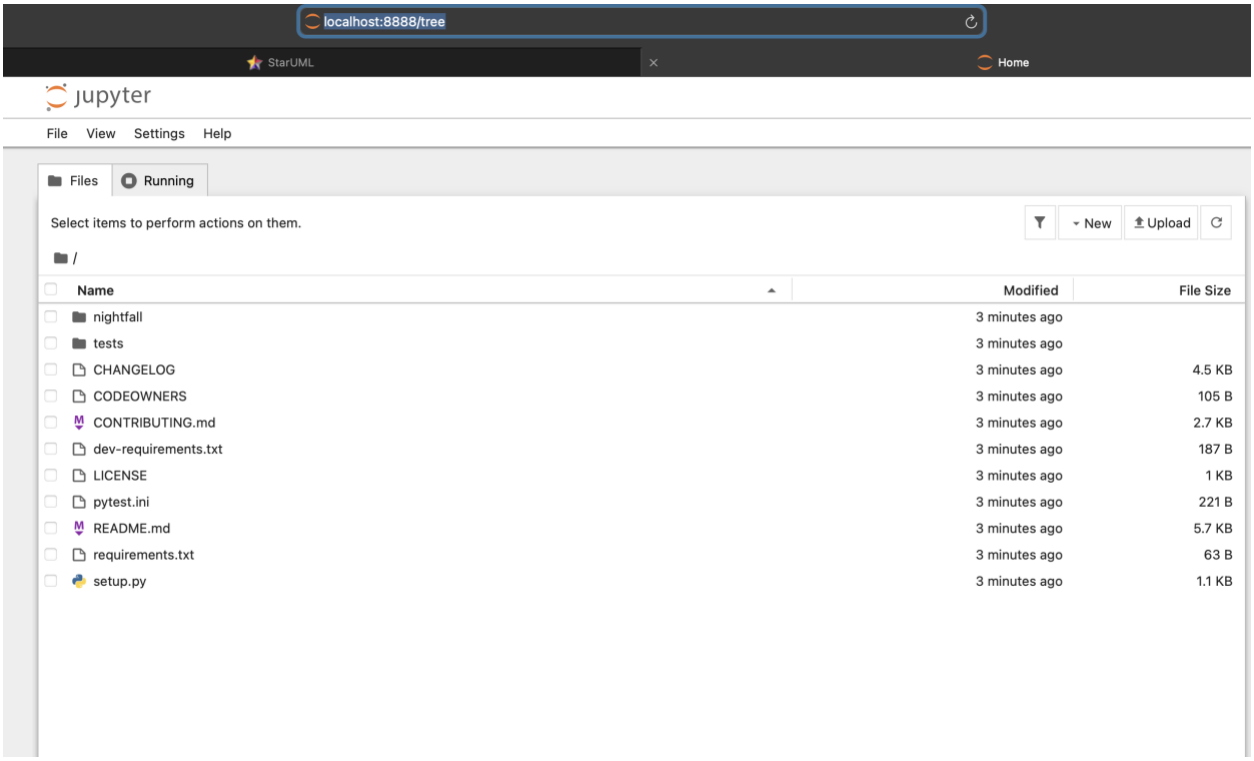
done

Policy has been created!

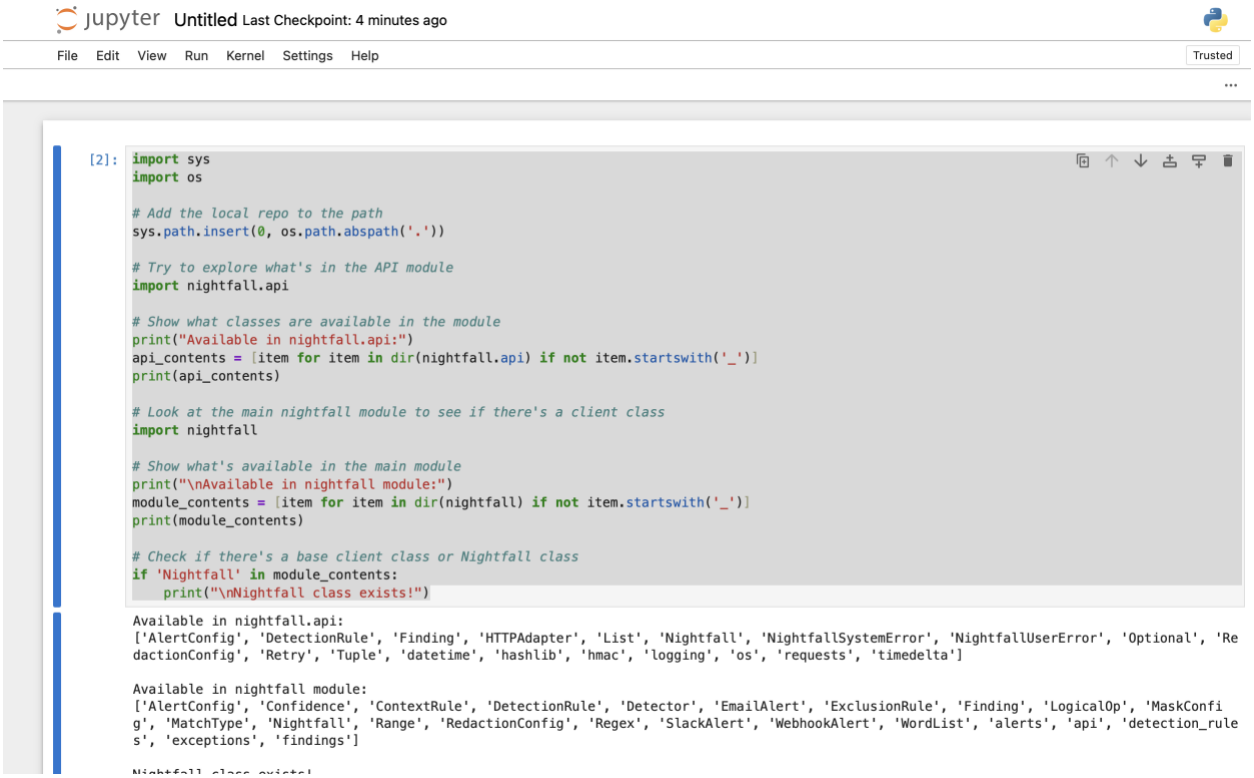
The policy was successfully created and is now actively scanning on Firewall for AI.

Done

Opening jupyter notebook in browser via local host



Running this code with some sample text using the API key as a python file.
first we see whats available



with this output we concluded “The correct class to use is Nightfall from either the nightfall.api module or directly from the nightfall module.”

then ran

import sys
import os

# Add the local repo to the path
sys.path.insert(0, os.path.abspath('.'))

# Import the Nightfall class
from nightfall.api import Nightfall

# Import detection rule classes if needed
from nightfall import DetectionRule, Detector

# Use your API key
api_key = "NF-zHGeEYUn5CYwZ4n1ufH1nFi8at7dVWuU" # Your API key

# Initialize the client
client = Nightfall(key=api_key)

Sample text with sensitive information

```
sample_text = """
Here is some sample text with sensitive information:
- Credit card: 4111-1111-1111-1111
- Email: john.doe@example.com
- SSN: 123-45-6789
- IP: 192.168.1.10
- Address: 123 Main St, San Francisco, CA 94105
"""

# Create detection rules (this is just an example, adjust based on available detectors)
# Option 1: Use policy UUID from your Nightfall dashboard
try:
    # Try using the policy you created in the dashboard
    # Replace with your actual policy UUID from the Nightfall dashboard
    policy_uuid = "YOUR_POLICY_UUID"  # You need to find this in your Nightfall dashboard

    # Since we don't have the policy UUID, let's try creating detection rules directly
    # Option 2: Create detection rules inline
    detection_rules = [
        DetectionRule(
            name="Credit Card Rule",
            detectors=[
                Detector(
                    min_confidence="POSSIBLE",
                    detector_type="CREDIT_CARD_NUMBER"
                )
            ]
        ),
        DetectionRule(
            name="Email Rule",
            detectors=[
                Detector(
                    min_confidence="POSSIBLE",
                    detector_type="EMAIL_ADDRESS"
                )
            ]
        ),
        DetectionRule(
            name="SSN Rule",
            detectors=[
                Detector(
                    min_confidence="POSSIBLE",
                    detector_type="US_SOCIAL_SECURITY_NUMBER"
                )
            ]
        )
```

```
    ]

    # Try scanning with the detection rules
    results = client.scan_text(sample_text, detection_rules=detection_rules)
    print("\nResults:")
    for finding in results:
        print(f"Type: {finding.detector_name}")
        print(f"Value: {finding.finding}")
        print(f"Confidence: {finding.confidence}")
        print("-" * 40)

except Exception as e:
    print(f"\nError occurred: {e}")
    print("Check the API documentation or README for proper usage.")

    # Let's try to get more information about what classes are available
    print("\nAvailable Detector types:")
    try:
        if hasattr(Detector, 'TYPES'):
            print(Detector.TYPES)
        else:
            print("Detector.TYPES not available")
    except Exception as e2:
            print(f"Error checking Detector types: {e2}")
```

but got

"Error occurred: Detector.__init__() got an unexpected keyword argument 'detect or_type'
Check the API documentation or README for proper usage.

Available Detector types:
Detector.TYPES not available
"

we then tried again with the proper parameters based on the methods available. code shows the scan_text method is available. we use it with the necessary parameters:

```
<< import sys
import os

# Add the local repo to the path
sys.path.insert(0, os.path.abspath('.'))

# Import the Nightfall class and necessary components
from nightfall.api import Nightfall
```

```
from nightfall import DetectionRule, Detector, Confidence

# Use your API key
api_key = "NF-zHGeEYUn5CYwZ4n1ufH1nFi8at7dVWuU"  # Your API key

# Initialize the client
client = Nightfall(key=api_key)

# Sample text with sensitive information
sample_text = """
Here is some sample text with sensitive information:
- Credit card: 4111-1111-1111-1111
- Email: john.doe@example.com
- SSN: 123-45-6789
- IP: 192.168.1.10
- Address: 123 Main St, San Francisco, CA 94105
"""

# Create detection rules based on the correct parameters from the docstring
# We need to use 'nightfall_detector' instead of 'detector_type'
try:
    detection_rules = [
        DetectionRule(
            name="Credit Card Rule",
            logical_op="OR",  # Required parameter
            detectors=[
                Detector(
                    min_confidence=Confidence.POSSIBLE,  # Using the Confidence enum
                    nightfall_detector="CREDIT_CARD_NUMBER"  # Correct parameter name
                )
            ]
        ),
        DetectionRule(
            name="Email Rule",
            logical_op="OR",
            detectors=[
                Detector(
                    min_confidence=Confidence.POSSIBLE,
                    nightfall_detector="EMAIL_ADDRESS"
                )
            ]
        ),
        DetectionRule(
            name="SSN Rule",
            logical_op="OR",
            detectors=[
```

```
            Detector(
                min_confidence=Confidence.POSSIBLE,
                nightfall_detector="US_SOCIAL_SECURITY_NUMBER"
            )
        ]
    )
]

# Scan the text with the detection rules
results = client.scan_text(sample_text, detection_rules=detection_rules)

print("\nResults:")
for i, rule_findings in enumerate(results):
    print(f"Rule {i+1} ({detection_rules[i].name}):")
    if not rule_findings:
        print("  No findings")
    for finding in rule_findings:
        print(f"  Type: {finding.detector_name}")
        print(f"  Value: {finding.finding}")
        print(f"  Confidence: {finding.confidence}")
        print("  " + "-" * 40)

except Exception as e:
    print(f"\nError occurred: {e}")
    print("Check the API documentation or README for proper usage.") >>
```

this gave us “

Error occurred: 'str' object has no attribute 'value'
Check the API documentation or README for proper usage.
”

It seemed there was an issue with how the Confidence enum was being handled. Then we tried using string values directly:

```python
import sys
import os

# Add the local repo to the path
sys.path.insert(0, os.path.abspath('.'))

# Import modules to explore
from nightfall import Confidence, LogicalOp

# Print the contents of the enums to understand their structure
print("Confidence values:")
for item in dir(Confidence):
    if not item.startswith('_'):
        print(f"  {item}")

print("\nLogicalOp values:")
for item in dir(LogicalOp):
    if not item.startswith('_'):
        print(f"  {item}")

# Let's also check the type of these objects
print("\nType of Confidence:", type(Confidence))
print("Type of LogicalOp:", type(LogicalOp))

# Try to access one of the enum values
try:
    print("\nConfidence.POSSIBLE:", Confidence.POSSIBLE)
    print("Type:", type(Confidence.POSSIBLE))
except Exception as e:
    print(f"Error accessing Confidence.POSSIBLE: {e}")

try:
    print("\nLogicalOp.OR:", LogicalOp.OR)
    print("Type:", type(LogicalOp.OR))
except Exception as e:
    print(f"Error accessing LogicalOp.OR: {e}")
```

```
Confidence values:
  LIKELY
  POSSIBLE
  UNLIKELY
  VERY_LIKELY
  VERY_UNLIKELY

LogicalOp values:
  ALL
  ANY

Type of Confidence: <class 'enum.EnumType'>
Type of LogicalOp: <class 'enum.EnumType'>

Confidence.POSSIBLE: Confidence.POSSIBLE
Type: <enum 'Confidence'>
Error accessing LogicalOp.OR: type object 'LogicalOp' has no attribute 'OR'
```

Now I made following changes

1. Changed `logical_op="OR"` to `logical_op=LogicalOp.ANY` as the enum doesn't have an "OR" value, but has "ANY"
2. Kept `min_confidence=Confidence.POSSIBLE` as that should work based on our exploration

This should have properly use the enum values as expected by the SDK. But did it??

```
<< import sys
import os

# Add the local repo to the path
sys.path.insert(0, os.path.abspath('.'))

# Import the Nightfall class and necessary components
from nightfall.api import Nightfall
from nightfall import DetectionRule, Detector, Confidence, LogicalOp

# Use your API key
api_key = "NF-zHGeEYUn5CYwZ4n1ufH1nFi8at7dVWuU"  # Your API key

# Initialize the client
client = Nightfall(key=api_key)

# Sample text with sensitive information
sample_text = """
Here is some sample text with sensitive information:
- Credit card: 4111-1111-1111-1111
- Email: john.doe@example.com
- SSN: 123-45-6789
- IP: 192.168.1.10
- Address: 123 Main St, San Francisco, CA 94105
"""

# Create detection rules using enum values for confidence and logical op
try:
    detection_rules = [
        DetectionRule(
            name="Credit Card Rule",
            logical_op=LogicalOp.ANY,  # Using the enum instead of string
            detectors=[
                Detector(
                    min_confidence=Confidence.POSSIBLE,  # Using the enum
                    nightfall_detector="CREDIT_CARD_NUMBER"
                )
            ]
        ),
        DetectionRule(
            name="Email Rule",
            logical_op=LogicalOp.ANY,
            detectors=[
                Detector(
```

```
                min_confidence=Confidence.POSSIBLE,
                nightfall_detector="EMAIL_ADDRESS"
            )
        ]
    ),
    DetectionRule(
        name="SSN Rule",
        logical_op=LogicalOp.ANY,
        detectors=[
            Detector(
                min_confidence=Confidence.POSSIBLE,
                nightfall_detector="US_SOCIAL_SECURITY_NUMBER"
            )
        ]
    )
]

# Scan the text with the detection rules
results = client.scan_text(sample_text, detection_rules=detection_rules)

print("\nResults:")
for i, rule_findings in enumerate(results):
    print(f"Rule {i+1} ({detection_rules[i].name}):")
    if not rule_findings:
        print("  No findings")
    else:
        for finding in rule_findings:
            print(f"  Type: {finding.detector_name}")
            print(f"  Value: {finding.finding}")
            print(f"  Confidence: {finding.confidence}")
            print("  " + "-" * 40)

except Exception as e:
    print(f"\nError occurred: {e}")
    import traceback
    traceback.print_exc()  # Print the full error trace for debugging
    print("Check the API documentation or README for proper usage.") >>
```

but I got this error

```
"Error occurred: 40013: {"code":40013,"message":"Request did not conform to expected JSON schema"}
Check the API documentation or README for proper usage.
Traceback (most recent call last):
  File "/var/folders/xj/3tw2w49n3gb9mhxvfbszzr540000gn/T/ipykernel_2197/1582056275.py", line 63, in
<module>
    results = client.scan_text(sample_text, detection_rules=detection_rules)
              ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Users/sagarshrestha/nightfall-python-sdk/nightfall/api.py", line 116, in scan_text
```

```
    _validate_response(response, 200)
  File "/Users/sagarshrestha/nightfall-python-sdk/nightfall/api.py", line 289, in _validate_response
    raise NightfallUserError(response.text, error_code)
nightfall.exceptions.NightfallUserError: 40013: {"code":40013,"message":"Request did not conform to expect
ed JSON schema"}
”
```

Despite following the available documentation and exploring the SDK's structure, we encountered persistent JSON schema errors when trying to use the Nightfall API. The issue appears to be with how the API expects structured data, or possibly changes in the API that aren't reflected in the SDK.
I hope this does demonstrate the technical knowledge and practicality of my understanding of setting up DLP policies.

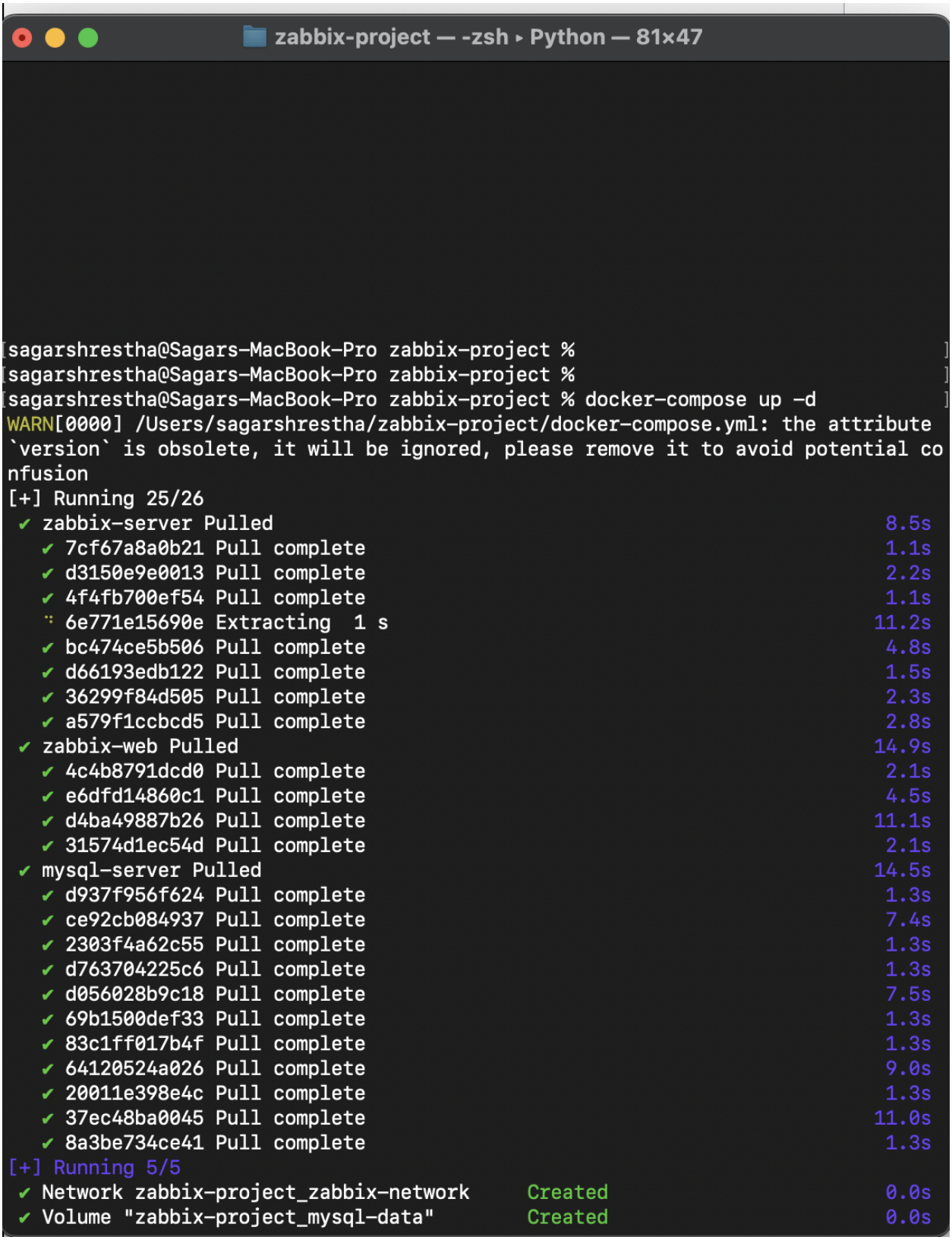
Session 3-5

Downloaded Zabbix via homebrew

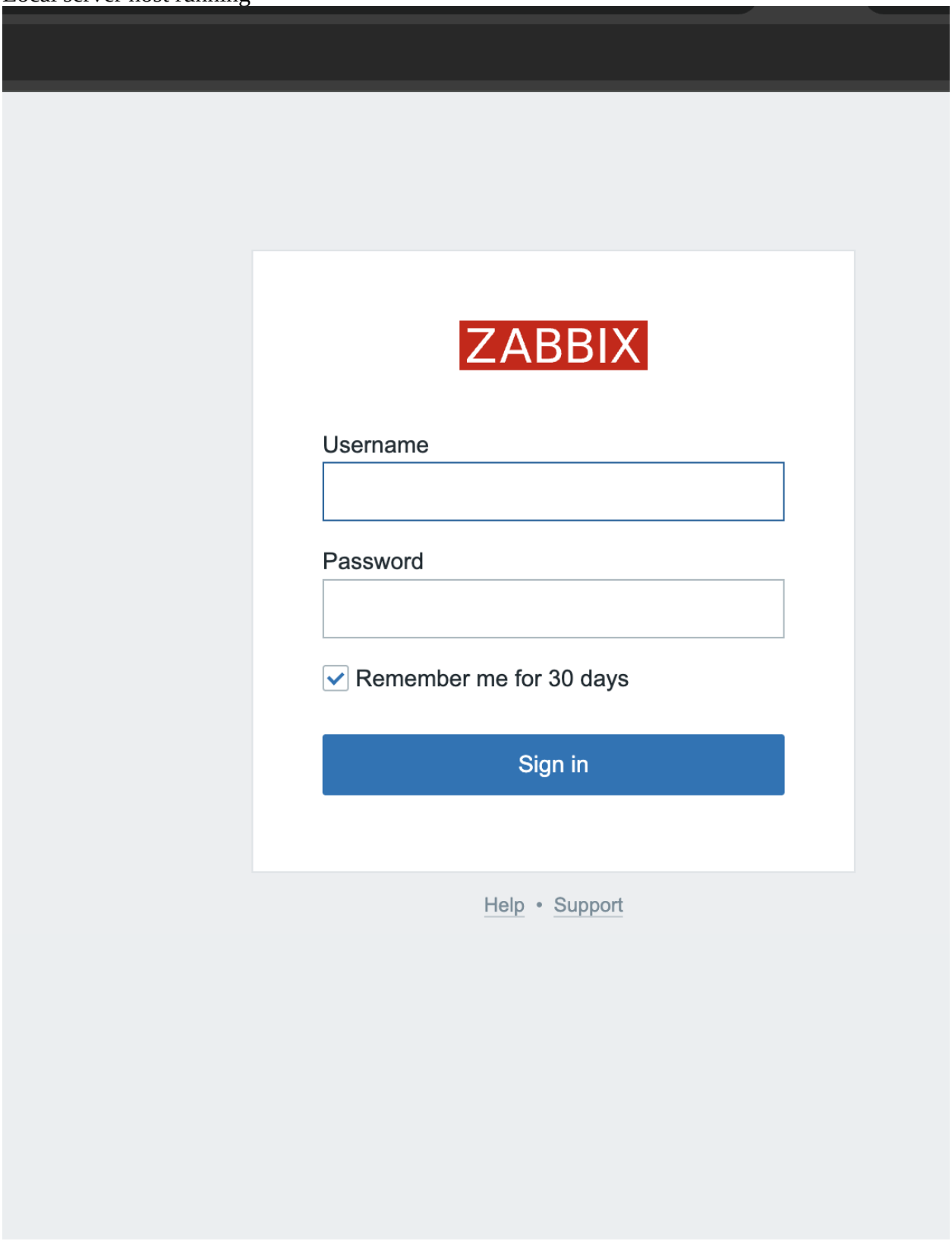
```
sagarshrestha@Sagars-MacBook-Pro ~ % brew install zabbix

==> Downloading https://ghcr.io/v2/homebrew/core/zabbix/manifests/7.2.5
######################################################################## 100.0%
==> Fetching dependencies for zabbix: pcre2
==> Downloading https://ghcr.io/v2/homebrew/core/pcre2/manifests/10.45
######################################################################## 100.0%
==> Fetching pcre2
==> Downloading https://ghcr.io/v2/homebrew/core/pcre2/blobs/sha256:f2abc87de6796
######################################################################## 100.0%
==> Fetching zabbix
==> Downloading https://ghcr.io/v2/homebrew/core/zabbix/blobs/sha256:588bed43b97b
######################################################################## 100.0%
==> Installing dependencies for zabbix: pcre2
==> Installing zabbix dependency: pcre2
==> Downloading https://ghcr.io/v2/homebrew/core/pcre2/manifests/10.45
Already downloaded: /Users/sagarshrestha/Library/Caches/Homebrew/downloads/bbac93
8545583185faba88567f1a952d7cc0b825820a2980533f7b3550dc31ad--pcre2-10.45.bottle_ma
nifest.json
==> Pouring pcre2--10.45.arm64_sequoia.bottle.tar.gz
🍺  /opt/homebrew/Cellar/pcre2/10.45: 242 files, 6.7MB
==> Installing zabbix
==> Pouring zabbix--7.2.5.arm64_sequoia.bottle.tar.gz
🍺  /opt/homebrew/Cellar/zabbix/7.2.5: 16 files, 3MB
==> Running `brew cleanup zabbix`...
Disable this behaviour by setting HOMEBREW_NO_INSTALL_CLEANUP.
Hide these hints with HOMEBREW_NO_ENV_HINTS (see `man brew`).
==> Upgrading 1 dependent of upgraded formulae:
Disable this behaviour by setting HOMEBREW_NO_INSTALLED_DEPENDENTS_CHECK.
Hide these hints with HOMEBREW_NO_ENV_HINTS (see `man brew`).
git 2.47.0 -> 2.49.0
==> Downloading https://ghcr.io/v2/homebrew/core/git/manifests/2.49.0
```

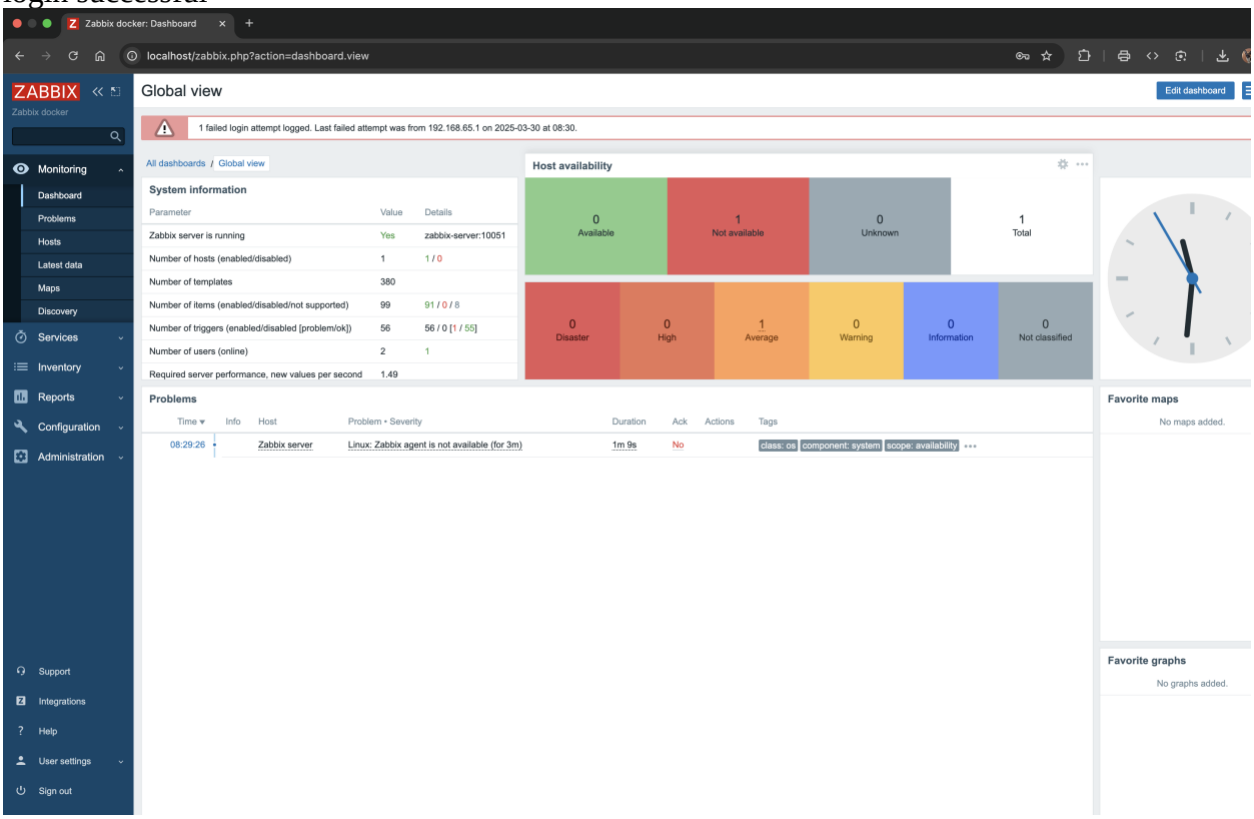
This didn’t work so because it didn’t properly downloaded the Zabbix server
so I had to login docker and run it there


zabbix-project — -zsh ▸ Python — 81×47
sagarshrestha@Sagars-MacBook-Pro zabbix-project %
sagarshrestha@Sagars-MacBook-Pro zabbix-project %
sagarshrestha@Sagars-MacBook-Pro zabbix-project % docker-compose up -d
WARN[0000] /Users/sagarshrestha/zabbix-project/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] Running 25/26
✔ zabbix-server Pulled 8.5s
✔ 7cf67a8a0b21 Pull complete 1.1s
✔ d3150e9e0013 Pull complete 2.2s
✔ 4f4fb700ef54 Pull complete 1.1s
⠿ 6e771e15690e Extracting 1 s 11.2s
✔ bc474ce5b506 Pull complete 4.8s
✔ d66193edb122 Pull complete 1.5s
✔ 36299f84d505 Pull complete 2.3s
✔ a579f1ccbcd5 Pull complete 2.8s
✔ zabbix-web Pulled 14.9s
✔ 4c4b8791dcd0 Pull complete 2.1s
✔ e6dfd14860c1 Pull complete 4.5s
✔ d4ba49887b26 Pull complete 11.1s
✔ 31574d1ec54d Pull complete 2.1s
✔ mysql-server Pulled 14.5s
✔ d937f956f624 Pull complete 1.3s
✔ ce92cb084937 Pull complete 7.4s
✔ 2303f4a62c55 Pull complete 1.3s
✔ d763704225c6 Pull complete 1.3s
✔ d056028b9c18 Pull complete 7.5s
✔ 69b1500def33 Pull complete 1.3s
✔ 83c1ff017b4f Pull complete 1.3s
✔ 64120524a026 Pull complete 9.0s
✔ 20011e398e4c Pull complete 1.3s
✔ 37ec48ba0045 Pull complete 11.0s
✔ 8a3be734ce41 Pull complete 1.3s
[+] Running 5/5
✔ Network zabbix-project_zabbix-network Created 0.0s
✔ Volume "zabbix-project_mysql-data" Created 0.0s

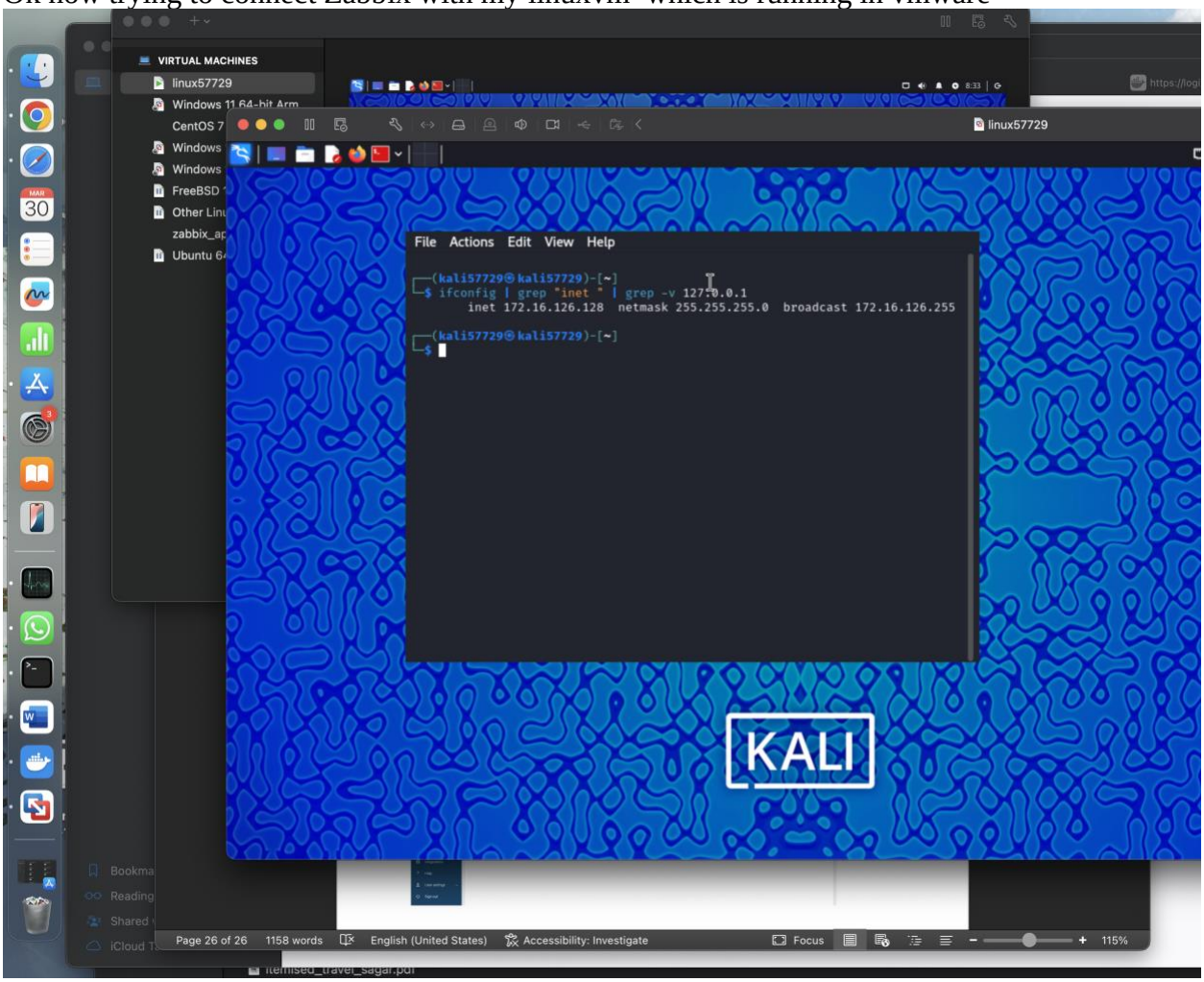
Local server host running



login successful



Ok now trying to connect Zabbix with my linuxvm which is running in vmware



running sudo update and install Zabbix agent

┌──(kali57729㉿kali57729)-[~]
└─$ sudo apt update
[sudo] password for kali57729:
Get:1 https://download.docker.com/linux/debian bullseye InRelease [43.3 kB]
Get:2 https://download.docker.com/linux/debian bullseye/stable arm64 Packages [51.4 kB]
Get:3 https://download.docker.com/linux/debian bullseye/stable arm64 Contents (deb) [1331 B]
Get:4 http://kali.download/kali kali-rolling InRelease [41.5 kB]
Get:5 http://kali.download/kali kali-rolling/main arm64 Packages [20.6 MB]
Get:6 http://kali.download/kali kali-rolling/main arm64 Contents (deb) [49.2 MB]
Get:7 http://kali.download/kali kali-rolling/contrib arm64 Packages [101 kB]
Get:8 http://kali.download/kali kali-rolling/contrib arm64 Contents (deb) [186 kB]
Get:9 http://kali.download/kali kali-rolling/non-free arm64 Packages [153 kB]
Get:10 http://kali.download/kali kali-rolling/non-free arm64 Contents (deb) [833 kB]
Get:11 http://kali.download/kali kali-rolling/non-free-firmware arm64 Packages [9777 B]

```
Get:12 http://kali.download/kali kali-rolling/non-free-firmware arm64 Contents (deb) [23.5 kB]
Fetched 71.2 MB in 6s (12.2 MB/s)
1962 packages can be upgraded. Run 'apt list --upgradable' to see them.

┌──(kali57729㉿kali57729)-[~]
└─$ sudo apt install zabbix-agent
The following packages were automatically installed and are no longer required:
  libintl-perl         libmodule-scandeps-perl    needrestart
  libintl-xs-perl      libproc-processtable-perl  tini
  libmodule-find-perl  libsort-naturally-perl
Use 'sudo apt autoremove' to remove them.

Upgrading:
  ldap-utils  libldap-common

Installing:
  zabbix-agent

Installing dependencies:
  libldap2  libmodbus5  zabbix-sender

Summary:
  Upgrading: 2, Installing: 4, Removing: 0, Not Upgrading: 1960
  Download size: 1769 kB
  Space needed: 3160 kB / 4068 MB available

Continue? [Y/n] Y
Get:1 http://http.kali.org/kali kali-rolling/main arm64 libldap2 arm64 2.6.9+dfsg-2 [179 kB]
Get:3 http://http.kali.org/kali kali-rolling/main arm64 zabbix-agent arm64 1:7.0.10+dfsg-2 [762
kB]
Get:2 http://kali.download/kali kali-rolling/main arm64 libmodbus5 arm64 3.1.11-2 [33.8 kB]
Get:4 http://http.kali.org/kali kali-rolling/main arm64 ldap-utils arm64 2.6.9+dfsg-2 [145 kB]
Get:5 http://http.kali.org/kali kali-rolling/main arm64 libldap-common all 2.6.9+dfsg-2 [34.8 kB]
Get:6 http://http.kali.org/kali kali-rolling/main arm64 zabbix-sender arm64 1:7.0.10+dfsg-2 [615
kB]
Fetched 1769 kB in 2s (860 kB/s)
Selecting previously unselected package libldap2:arm64.
(Reading database ... 403261 files and directories currently installed.)
Preparing to unpack .../0-libldap2_2.6.9+dfsg-2_arm64.deb ...
Unpacking libldap2:arm64 (2.6.9+dfsg-2) ...
Selecting previously unselected package libmodbus5:arm64.
Preparing to unpack .../1-libmodbus5_3.1.11-2_arm64.deb ...
Unpacking libmodbus5:arm64 (3.1.11-2) ...
Selecting previously unselected package zabbix-agent.
Preparing to unpack .../2-zabbix-agent_1%3a7.0.10+dfsg-2_arm64.deb ...
Unpacking zabbix-agent (1:7.0.10+dfsg-2) ...
```

Preparing to unpack .../3-ldap-utils_2.6.9+dfsg-2_arm64.deb ...
Unpacking ldap-utils (2.6.9+dfsg-2) over (2.5.13+dfsg-5+b3) ...
Preparing to unpack .../4-libldap-common_2.6.9+dfsg-2_all.deb ...
Unpacking libldap-common (2.6.9+dfsg-2) over (2.5.18+dfsg-3) ...
Selecting previously unselected package zabbix-sender.
Preparing to unpack .../5-zabbix-sender_1%3a7.0.10+dfsg-2_arm64.deb ...
Unpacking zabbix-sender (1:7.0.10+dfsg-2) ...
Setting up libmodbus5:arm64 (3.1.11-2) ...
Setting up libldap-common (2.6.9+dfsg-2) ...
Installing new version of config file /etc/ldap/ldap.conf ...
Setting up zabbix-sender (1:7.0.10+dfsg-2) ...
Setting up libldap2:arm64 (2.6.9+dfsg-2) ...
Setting up ldap-utils (2.6.9+dfsg-2) ...
Setting up zabbix-agent (1:7.0.10+dfsg-2) ...

Creating config file /etc/zabbix/zabbix_agentd.conf with new version
update-rc.d: We have no instructions for the zabbix-agent init script.
update-rc.d: It looks like a network service, we disable it.
zabbix-agent.service is a disabled or a static unit, not starting it.
Processing triggers for libc-bin (2.40-2) ...
Processing triggers for man-db (2.13.0-1) ...
Processing triggers for kali-menu (2024.3.1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

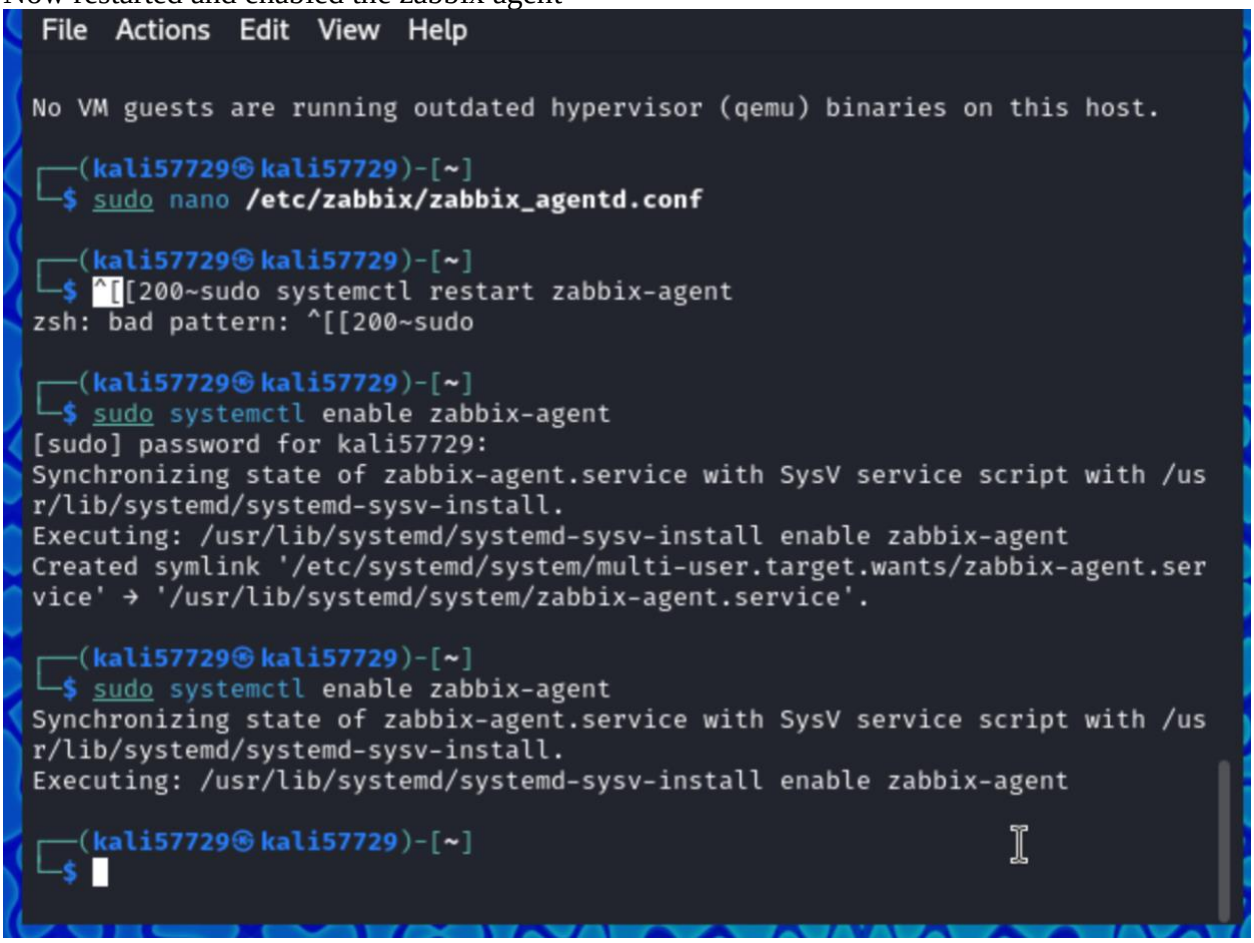
No containers need to be restarted.

No user sessions are running outdated binaries.

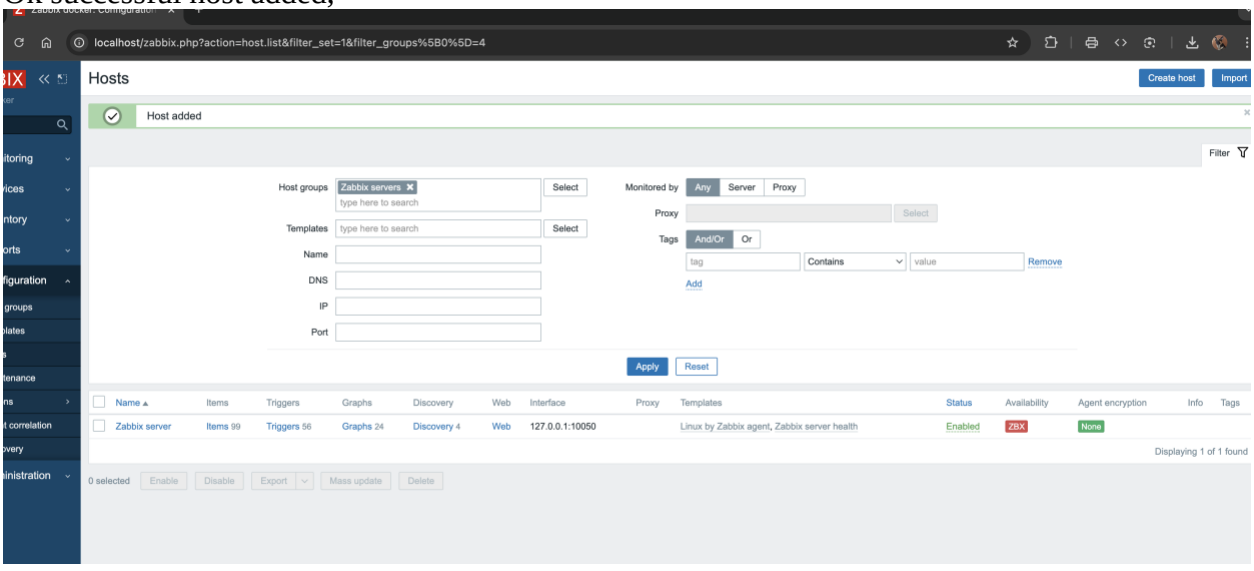
No VM guests are running outdated hypervisor (qemu) binaries on this host.
Now used this command to see if the server is set to ip of Zabbix server which is 127.0.0.1 and hostname as linux57729

```
┌──(kali57729㉿kali57729)-[~]
└─$ sudo nano /etc/zabbix/zabbix_agentd.conf
```

Now restarted and enabled the zabbix agent



Ok successful host added,



Despite Zabbix can ping to kali, kali cant ping to Zabbix. Which I could solve for two days.

to no avail, I decided to install Zabbix in my linux vm itself.

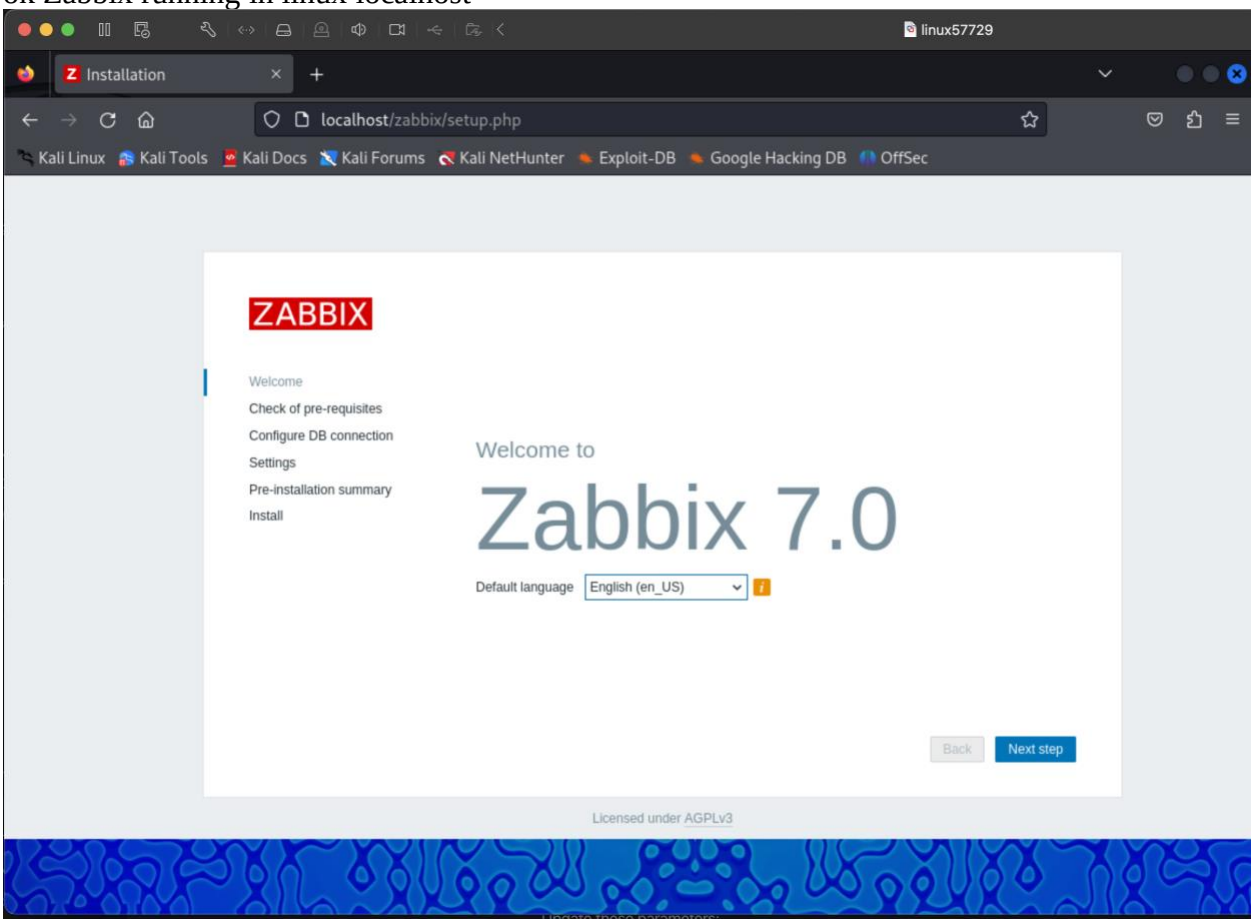
Download Zabbix repository package
wget https://repo.zabbix.com/zabbix/6.0/debian/pool/main/z/zabbix-release/zabbix-release_6.0-4+debian11_all.deb

# Install it
sudo dpkg -i zabbix-release_6.0-4+debian11_all.deb

# Update package lists
sudo apt update

# Now try installing the packages
sudo apt install -y zabbix-server-mysql zabbix-frontend-php zabbix-apache-conf zabbix-sql-scripts zabbix-agent

ok Zabbix running in linux localhost



This couldn’t connect either.

Only solutions left are trying Grafana+prmetheus or data dog who have a excellent support for ARM 64 architecture aswell their adoption in industry level.

Thankyou!