Microsoft Sentinel for Developers

Laura Kokkarinen & Thomas Vochten

Who are we?

Thomas Vochten

Workplace Solution Lead, MVP

Sopra Steria, Belgium

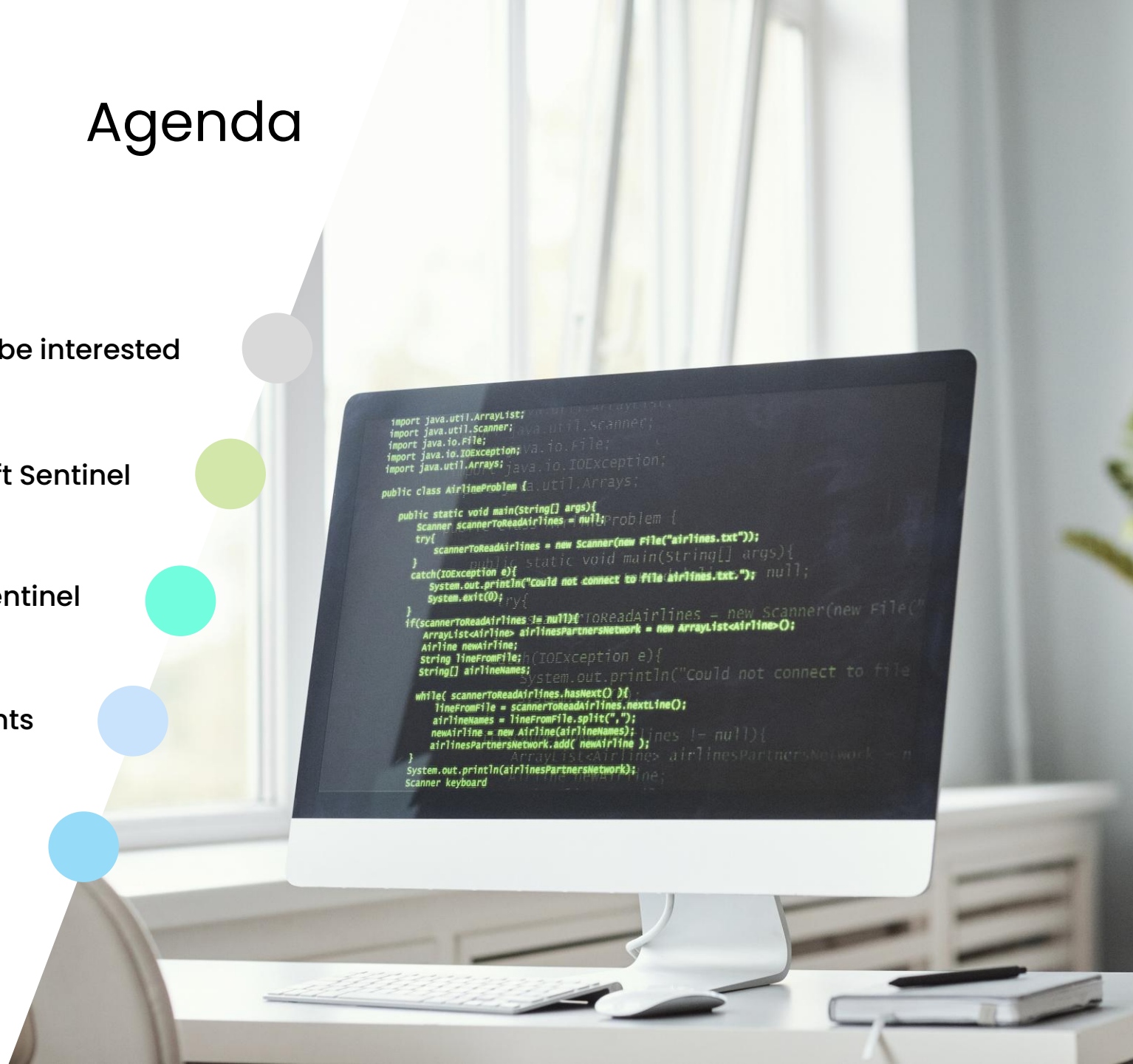
Laura Kokkarinen

Software Architect, MVP

Sulava, Finland

Agenda

- Why you should be interested
- A quick tour around Microsoft Sentinel
- Plugging in a custom app to Sentinel
- Detecting incidents
- Automating responses



Now what?

You deployed your app.
Do you really know what's going on?

Some Real-World Security Scenarios

- Compromised accounts
- API key misuse
- Brute-force login attempts
- Session hijacking
- Account abuse from unknown IPs
- Use of expired or forged JWT tokens
- Sudden spike in data exports or downloads
- Malicious actors misusing your app to get a foot in the door
- Access to production systems from unusual locations
- Chained low-severity vulnerabilities used for privilege escalation
- …

But isn’t that an operations problem?

What do you need to know as a developer and why

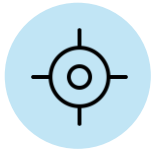
Security Logging and Monitoring

- Monitoring depends on quality logging!
- OWASP Top 10: Security Logging and Monitoring Failures
- “Shift Left”
  “Defense in depth”
- Sentinel helps us adhere to the security requirements

- *Events, such as logins, failed logins, and high-value transactions, are not logged.*
- *Warnings and errors generate no, inadequate, or unclear log messages.*
- *Logs are only stored locally.*
- *Logs of applications and APIs are not **monitored for suspicious activity**.*
- *No **alerting thresholds** and **response escalation processes** are in place.*
- *Penetration testing and DAST scanning do not trigger alerts.*
- *The application cannot detect, escalate, or alert for active attacks in **real-time**.*

https://owasp.org/Top10

Microsoft Sentinel to the Rescue



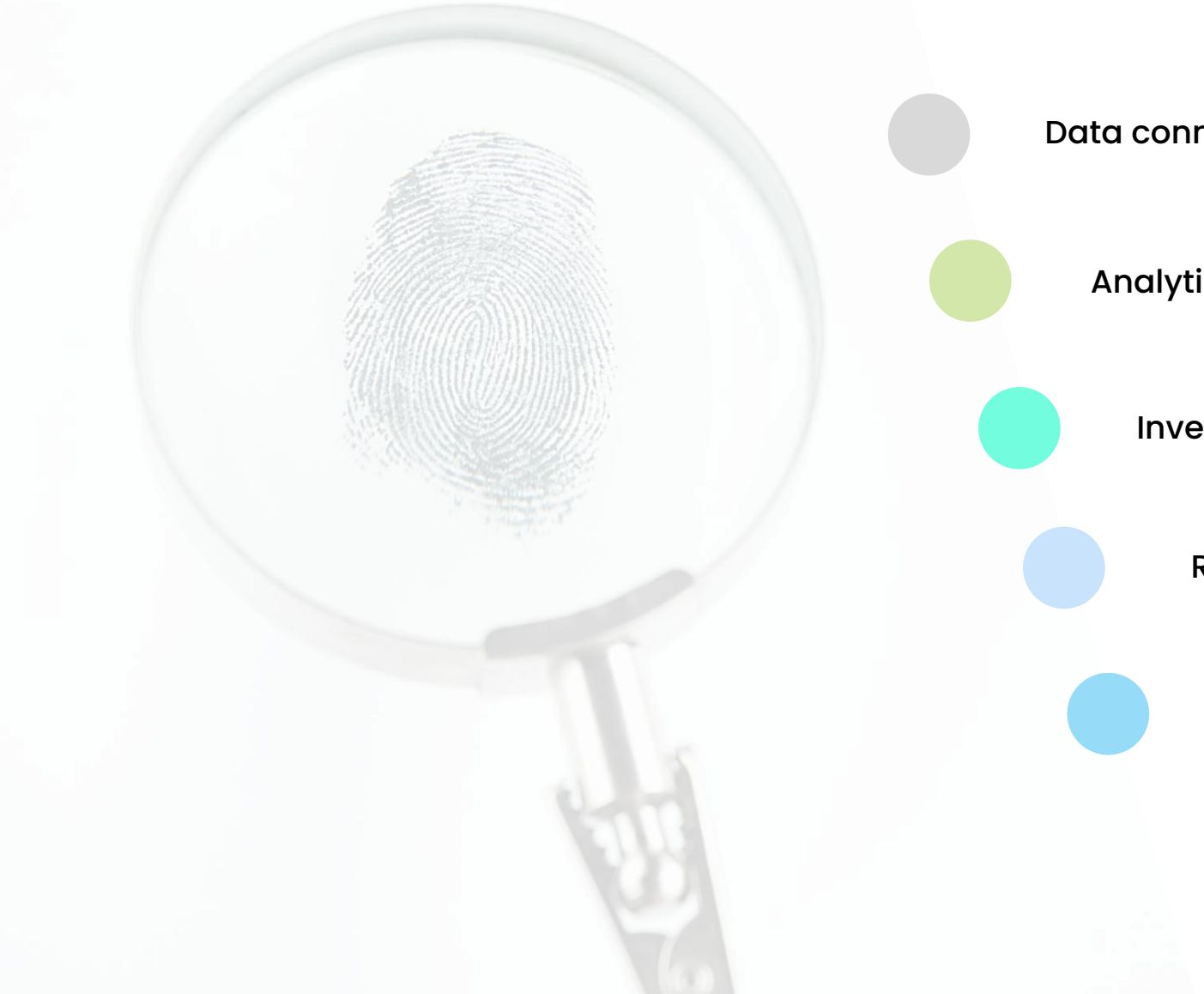
Unified SIEM + SOAR

Built on Azure Monitor and Log Analytics

Correlates logs and detects threats

Automates incident response

Scalable and cloud-native

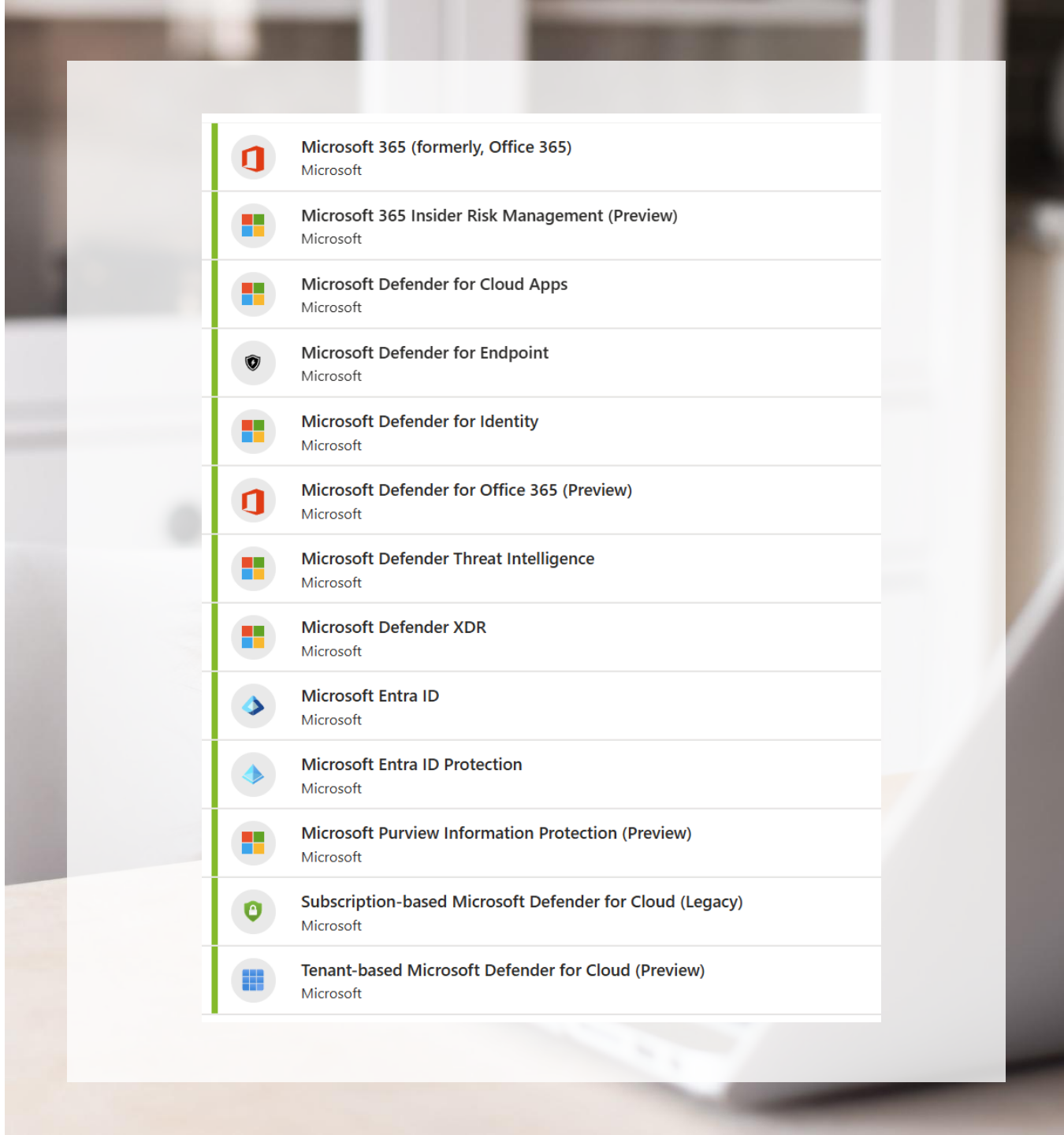


Key Components & Capabilities

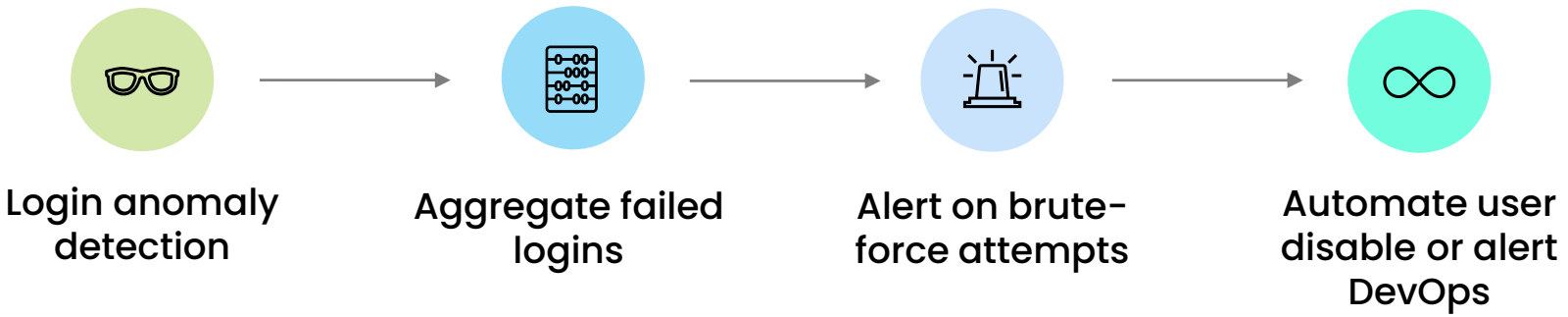
- Data connectors and ingestion
- Analytics and detection rules
- Investigation tools and dashboards
- Response automation (Logic Apps, Functions)
- Built-in AI: Fusion, UEBA, anomaly detection

Data connectors

- 100+ built-in connectors
- Native support for most Microsoft services
- Templates for rules and workbooks
- Custom ingestion through an API, CEF or Syslog



Example:
Typical Detection & Response Flow



Which events to log?

- Use threat modeling to identify what events to log for your specific application and business case
- Log enough but not too much

Authentication and authorization failures

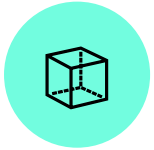
Accessing resources, performing operations

Session management failures

Invalid session token, JWT validation failures

Input validation failures

Invalid params/values, payloads, outputs from external services



Deserialization failures

Invalid or unexpected serialized data

Use of higher-risk functionality

All admin actions, sensitive data, user-generated content, legal etc.

Suspicious activity

Attempts to bypass control flow, excessive usage

What information to log?

- Data to exclude or handle carefully: sensitive data, secrets, session identifiers, internal system information
- Log enough but not too much

When (UTC)

- 2020-05-27T14:30:00Z
- Consistency

Who (user or system)

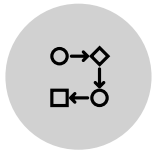
- User type
- User ID
- Source IP address
- User agent
- Session ID (hashed!)

What (event details)

- Event type and severity
- Performed action
- Event description
- Affected resource
- Action outcome

Where (system location)

- App ID and address
- Client's geolocation
- Code location



How (technical details)

- Request details
- Response details
- Correlation ID

Log entry structure and best practices

- Log all events with a consistent structure (JSON schema)
- Validate and sanitize all data originating from clients and external applications before processing and logging
- Ensure proper encoding (e.g., UTF-8) to avoid log corruption


```
{
    "datetime": "2025-05-10T14:30:00Z",
    "event_timestamp": "2025-05-10T14:29:55Z",
    "appid": "myapp-v1.2.3",
    "event": "AUTHN_login_fail",
    "severity": "WARN",
    "description": "User login failed due to incorrect password",
    "user_id": "9ed6e181-c4e5-4c9f-b7a6-6344f39a77bb",
    "user_type": "authenticated_user",
    "source_ip": "192.168.1.1",
    "useragent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)",
    "host_ip": "10.0.0.1",
    "hostname": "auth.myapp.com",
    "protocol": "HTTPS",
    "port": 443,
    "request_uri": "/login",
    "request_method": "POST",
    "result_status": "FAILED",
    "reason": "Incorrect credentials",
    "http_status_code": 401,
    "region": "West Europe",
    "geo": "Europe",
    "correlation_id": "bf3e8dc2-cabe-443f-89c2-a2191f2ca112",
}
```

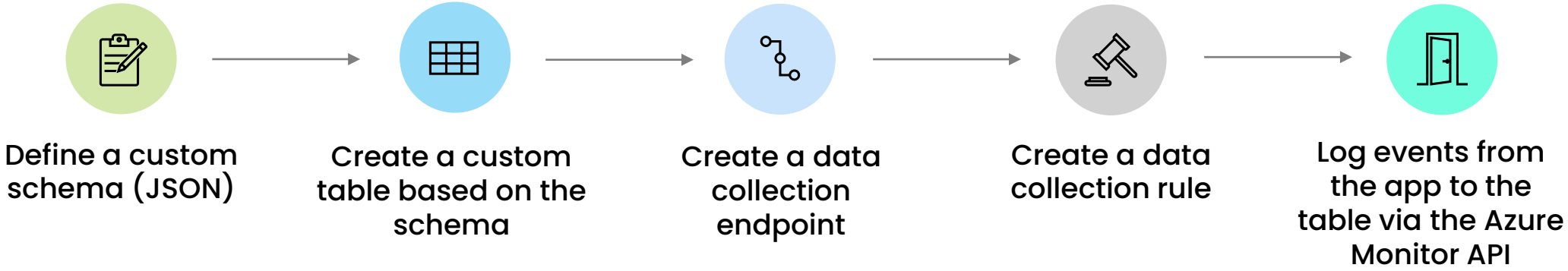
Application Insights

- Connect Application Insights to the Log Analytics Workspace used by Microsoft Sentinel
- **Pros:** Fast and easy
- **Cons:** Other than security events end up on the Sentinel workspace, resulting in higher cost

Azure Monitor API

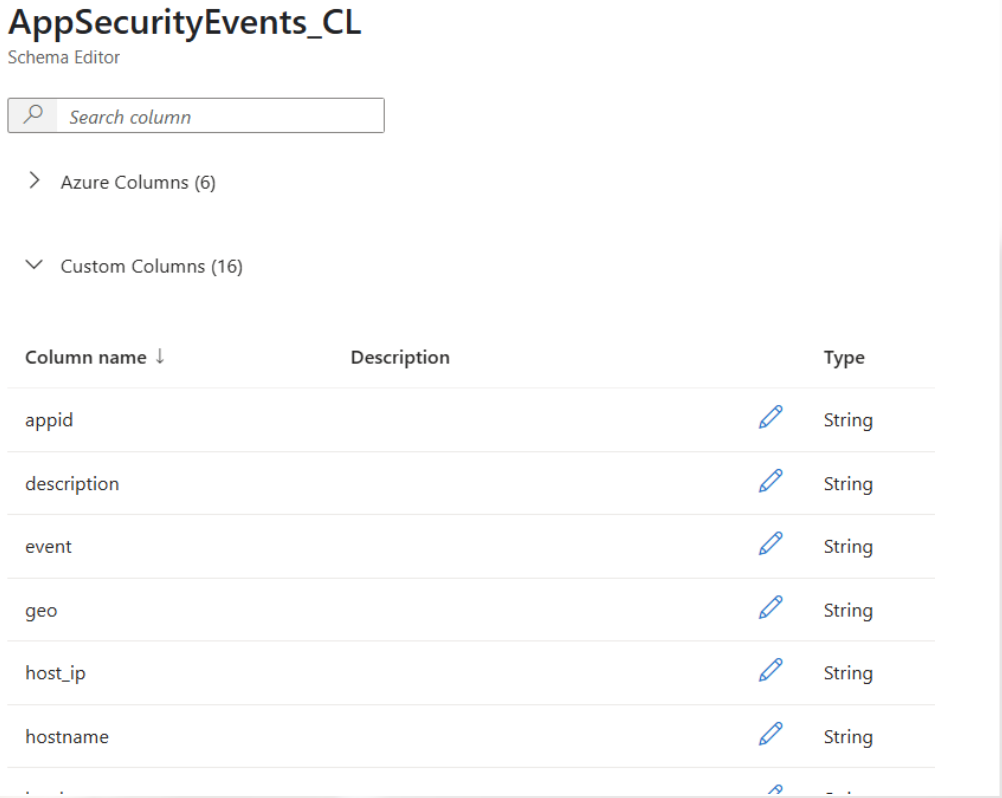
- Send log entries to the Log Analytics Workspace used by Microsoft Sentinel via Azure Monitor REST API
- **Pros:** Control what log entries to send to Sentinel, hence cost efficient to monitor
- **Cons:** Requires more effort to initially implement

Enable logging to Sentinel workspace via REST API



From schema to custom log analytics table


```
[
  {
    "timestamp": "2025-01-01T01:01:01.0000000Z",
    "appid": "00000000-0000-0000-0000-000000000000",
    "region": "West Europe",
    "geo": "Europe",
    "level": "Information",
    "event": "AUTHN_login_success:00000000-0000-0000-0000-000000000000",
    "description": "User 00000000-0000-0000-0000-000000000000 logged in
    successfully.",
    "host_ip": "127.0.0.1",
    "port": 1234,
    "request_method": "POST",
    "protocol": "https",
    "hostname": "azure.functions.com",
    "request_uri": "/api/LogEvent",
    "source_ip": "127.0.0.1",
    "useragent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
    AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36"
  }
]
```

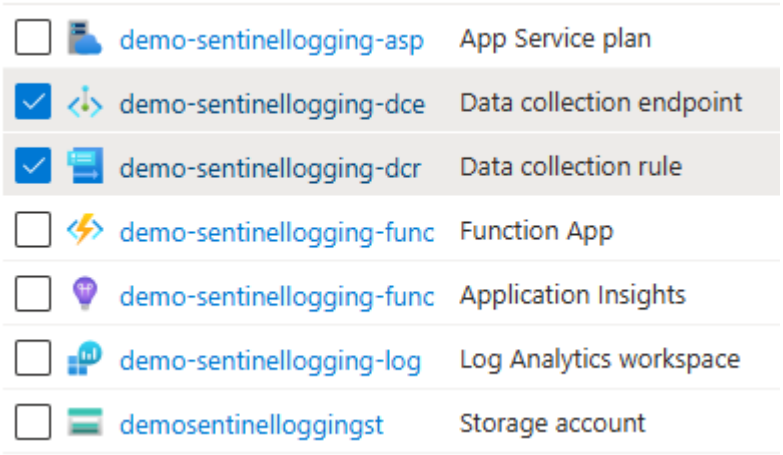
Data collection endpoints & rules

- Needed when calling the Azure Monitor Logs Ingestion API

- **DCE:** Log Ingestion
- **DCR:** Access to custom table

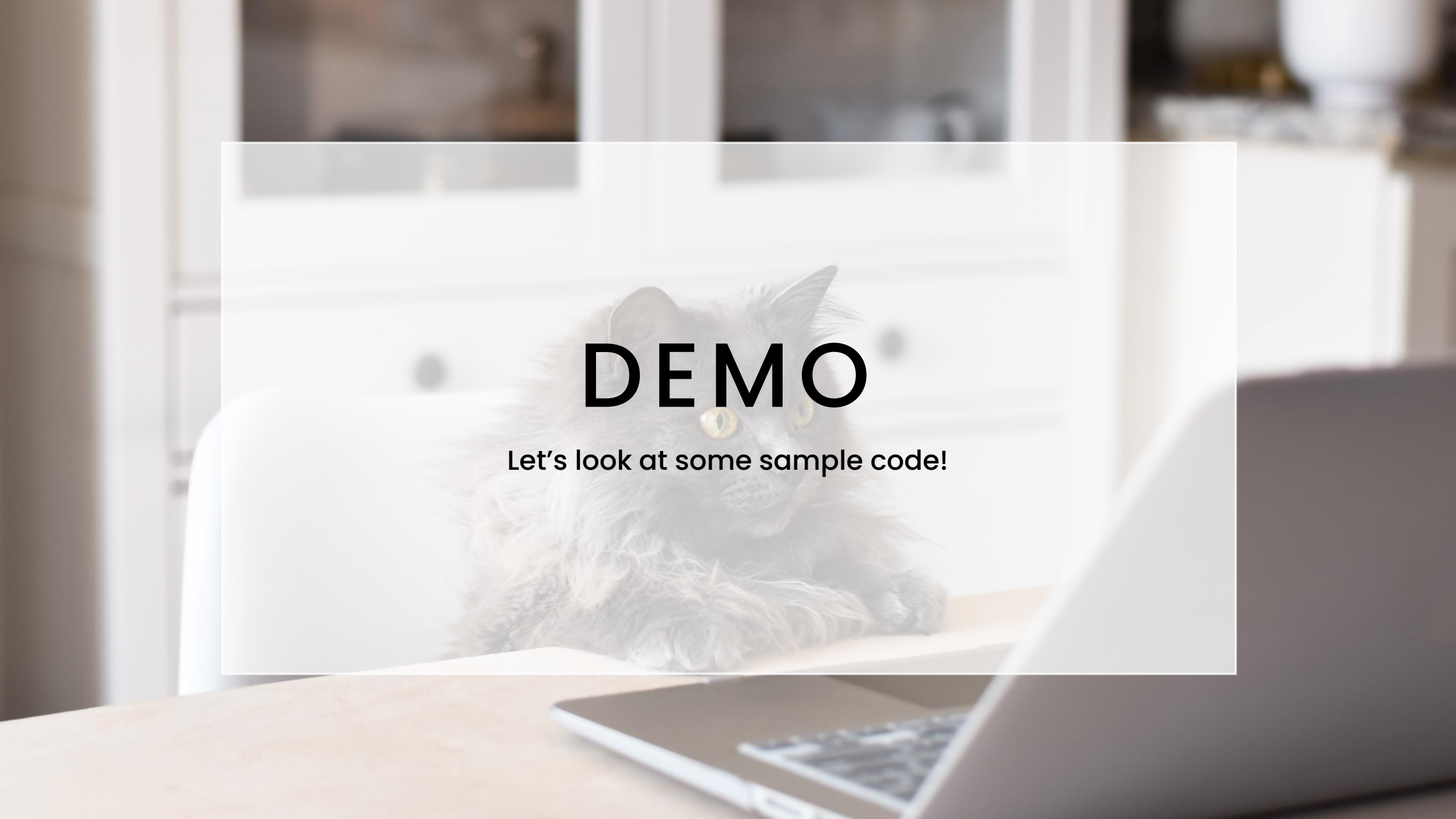
- **Important:** Grant your app *"Monitoring Metrics Publisher"* permissions to the data collection rule.

- https://github.com/LauraKokkarinen/FunctionApp.SentinelLogging



- DCE: Logs Ingestion URL

- DCR: Immutable ID
- DCR: Data Source (table name)



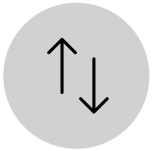
DEMO

Let’s look at some sample code!

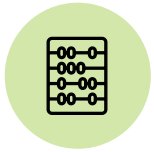
KQL Query Basics

Querying

Filtering



Sorting



Aggregation

Summarization

KQL Query Basics

```
1  AppSecurityEvents_CL
2  | project TimeGenerated, event, level
```

Results | Chart | Add bookmark

TimeGenerated [UTC]	event	level
5/6/2025, 1:30:53.163 PM	sequence_fail:00000000-0000-0000-0000-000000000000	Critical
5/6/2025, 1:30:53.100 PM	privilege_permissions_changed:00000000-0000-0000-0000-000000000000	Warning
5/6/2025, 1:30:53.059 PM	malicious_direct_reference:00000000-0000-0000-0000-000000000000	Critical
5/6/2025, 1:30:52.951 PM	malicious_cors:00000000-0000-0000-0000-000000000000,illegal.origin.com	Critical
5/6/2025, 1:30:52.901 PM	malicious_attack_tool:00000000-0000-0000-0000-000000000000,Nikto	Critical
5/6/2025, 1:30:52.781 PM	malicious_extraneous:00000000-0000-0000-0000-000000000000,creditcardnum	Critical
5/6/2025, 1:30:52.657 PM	malicious_excess_404:00000000-0000-0000-0000-000000000000	Warning
5/6/2025, 1:30:52.607 PM	input_validation_fail:00000000-0000-0000-0000-000000000000,date_of_birth	Warning
5/6/2025, 1:30:52.563 PM	upload_delete:00000000-0000-0000-0000-000000000000,1234567890	Information
5/6/2025, 1:30:52.509 PM	**upload_validation:file.png,virusscan,FAILED**	**Critical**

TimeGenerated [UTC]	2025-05-06T13:30:52.5090415Z
event	upload_validation:file.png,virusscan,FAILED
level	Critical

KQL to Detect Impossible Travel

```
AppSecurityEvents_CL
| where event startswith "AUTHN_login_success"
| extend user_id = tostring(split(event, ":")[1])
| summarize
    login_count = count(),
    countries = dcount(geo),
    regions = make_set(region),
    hosts = make_set(host_ip)
    by user_id, bin(timestamp, 1h)
| where login_count >= 2 and countries > 1
```

KQL with Threat Intelligence Integration

```
AppSecurityEvents_CL
| where timestamp > ago(1d)
| where isnotempty(source_ip)
| extend source_ip_str = tostring(source_ip)
| join kind=inner (
    ThreatIntelligenceIndicator
    | where TimeGenerated > ago(1d)
    | where isnotempty(NetworkIP)
    | extend ti_ip = tostring(NetworkIP)
    )
    on $left.source_ip_str == $right.ti_ip
| project
    timestamp,
    source_ip,
    event,
    description,
    ThreatType,
    ConfidenceScore,
    Description
```

Detecting and alerting on events

Turn KQL into scheduled (or near real-time) detection

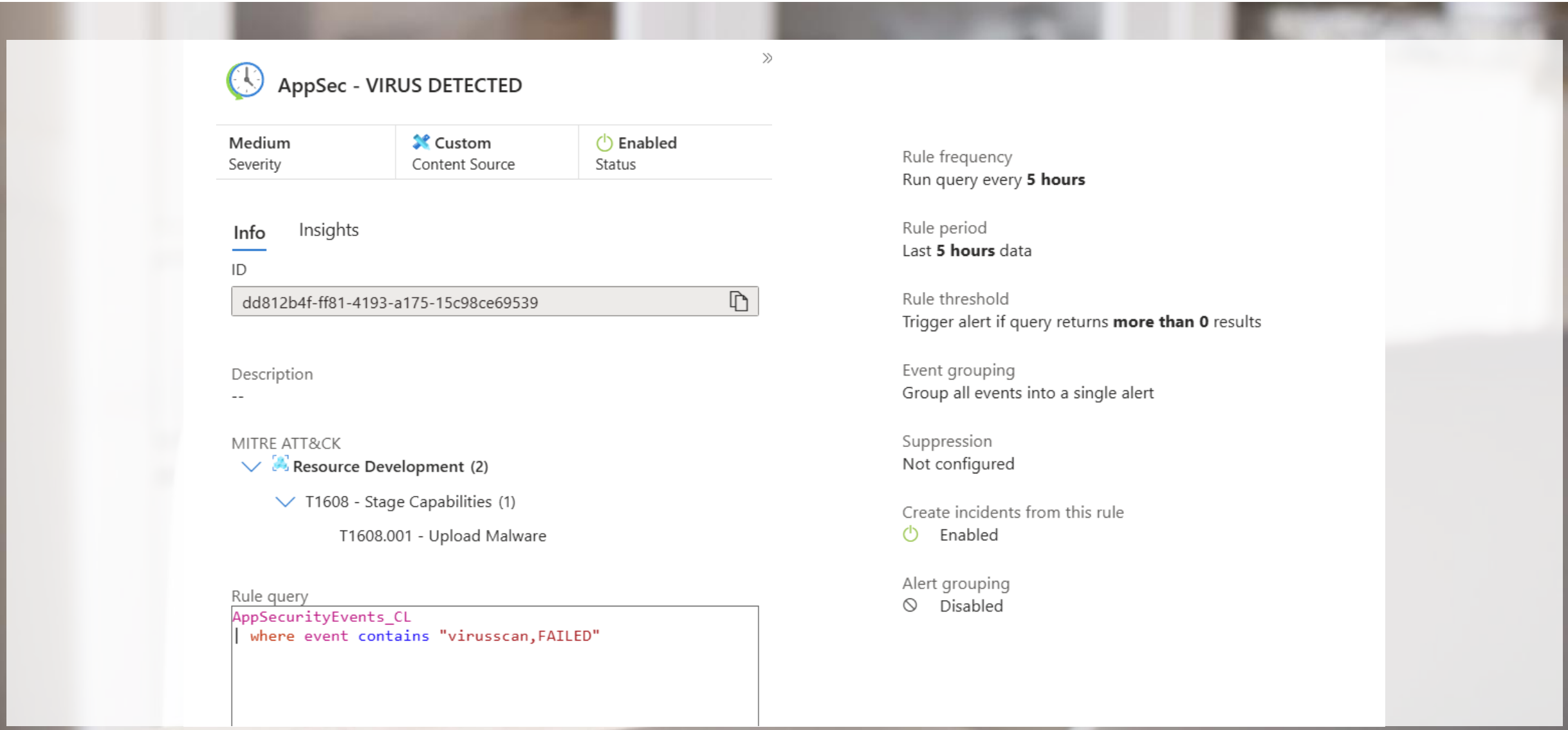
Define thresholds and frequency

Generate alerts

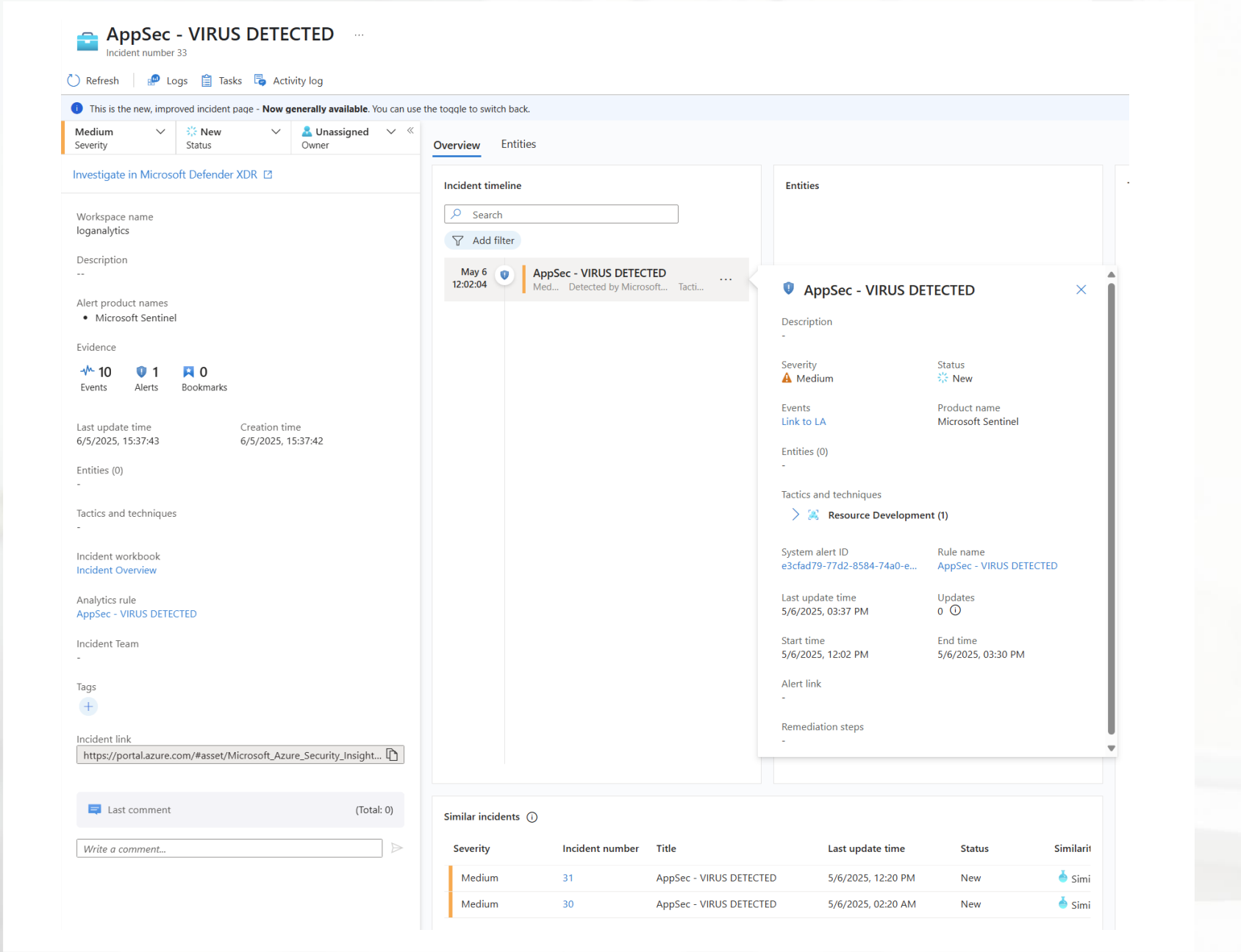
Group alerts into incidents

Classification (MITRE ATT&CK)

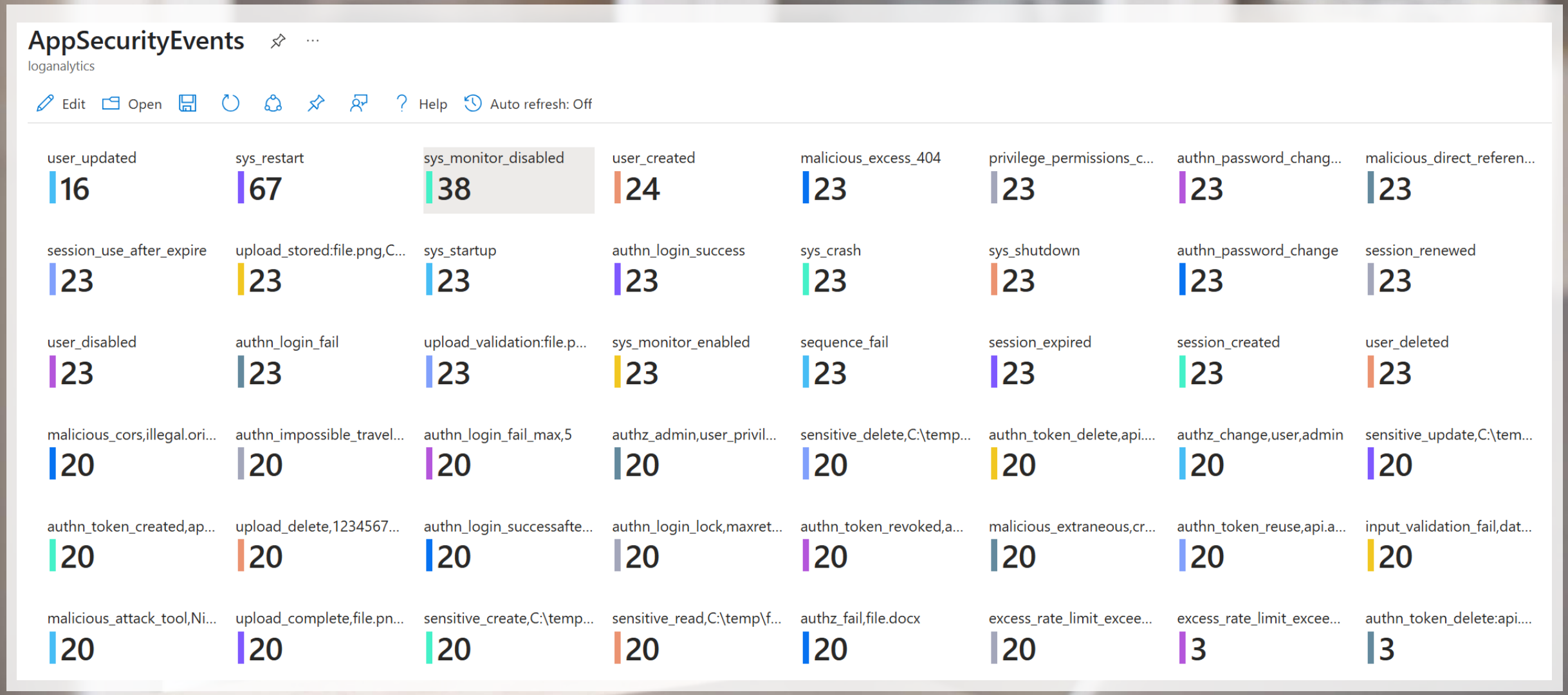
Creating an analytics rule



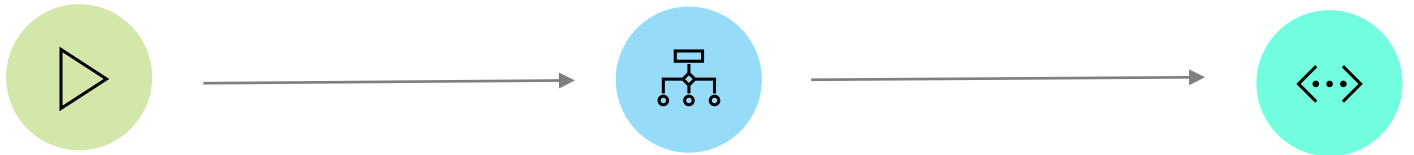
Managing Incidents



Custom Workbooks



Automating a response



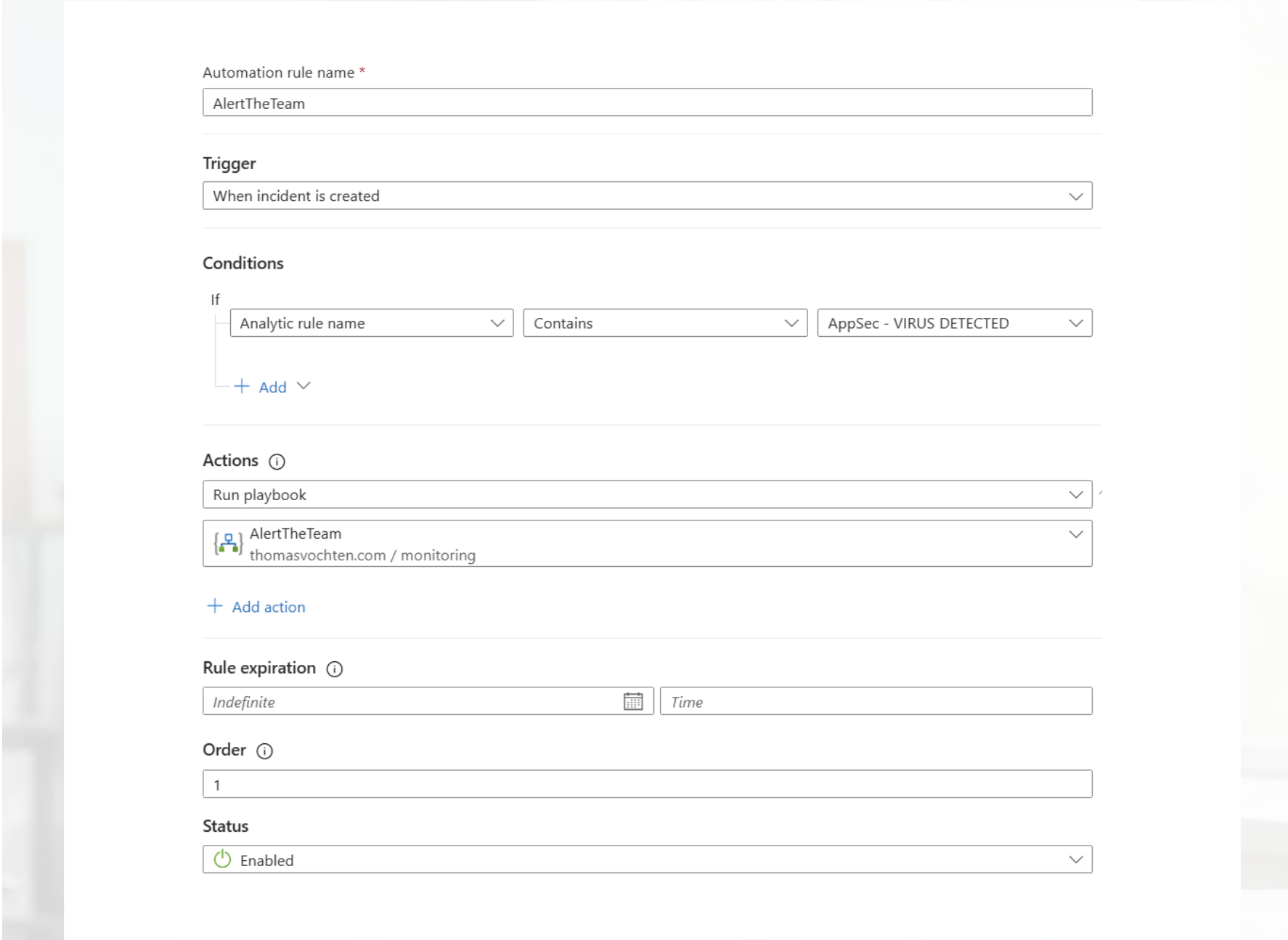
Automation rules define triggers and actions

Playbooks (Logic Apps) contain & execute your logic

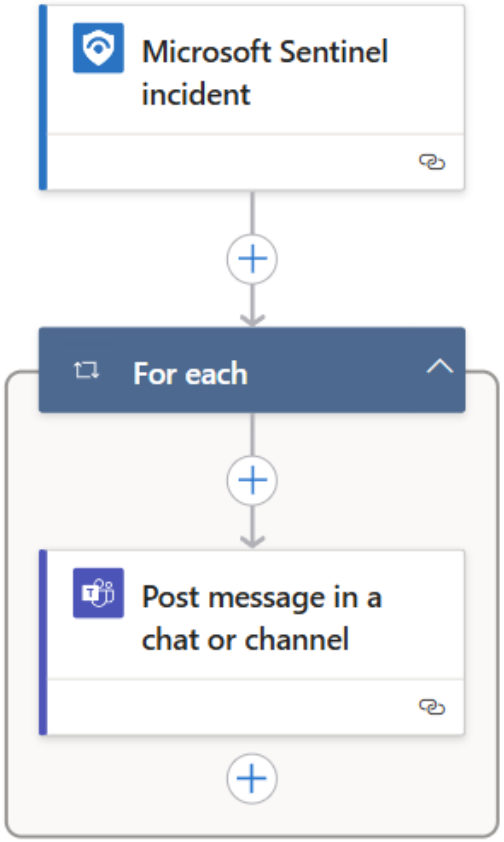
- Playbook templates
- Create your own

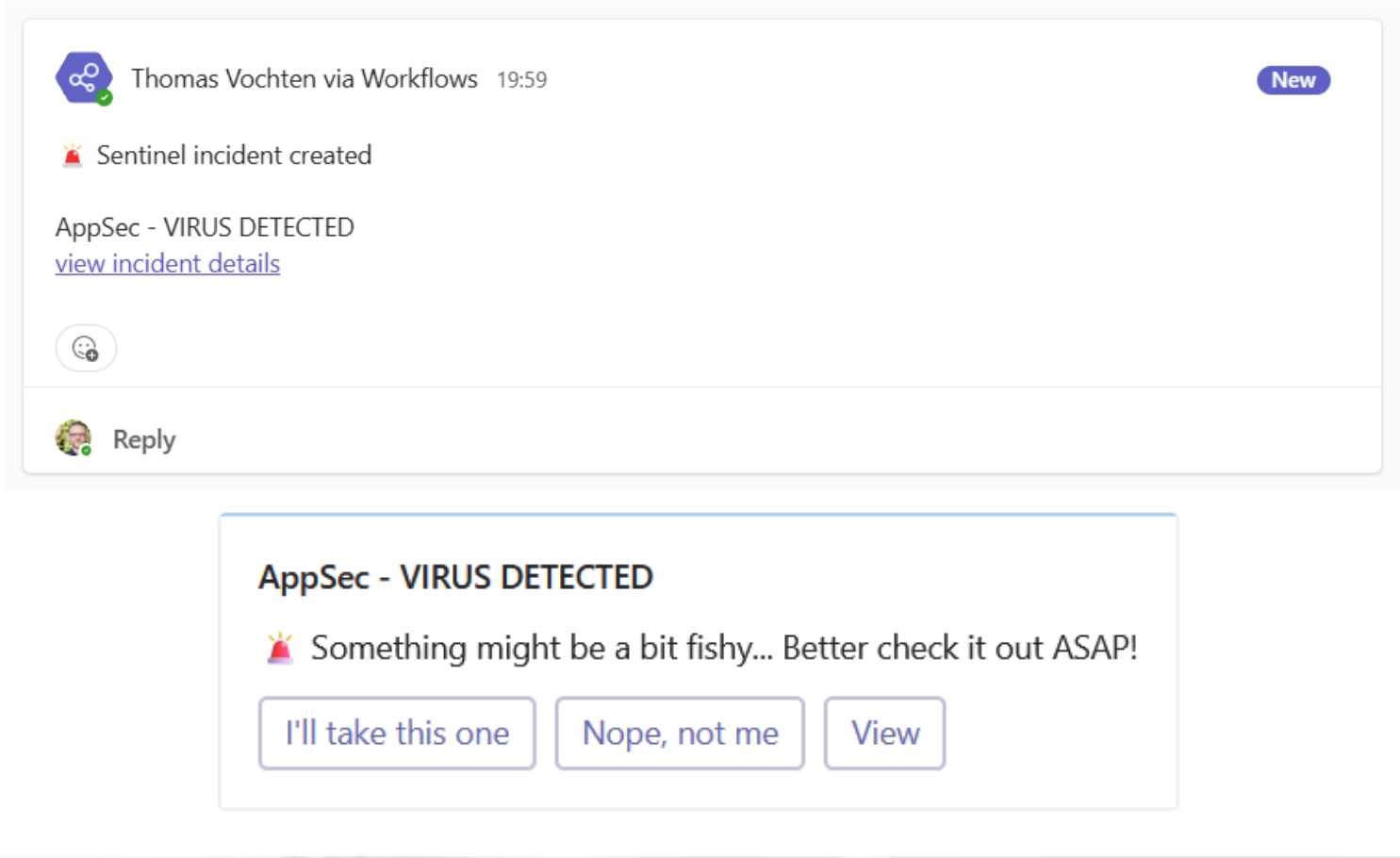
Call services or your *own app*

Automation Rules

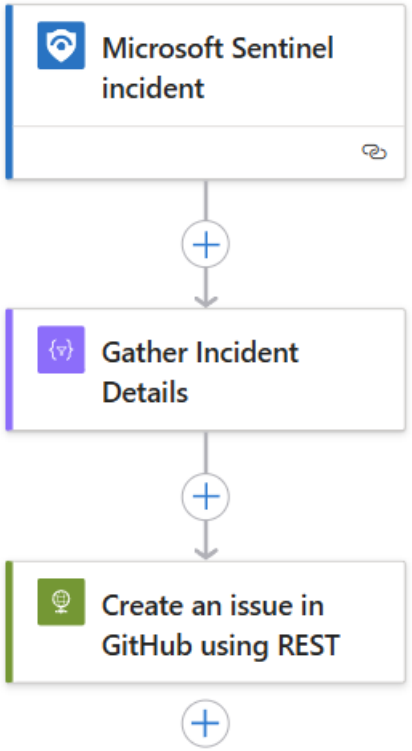


Playbook Example: Posting in Teams





Playbook Example: Creating a GitHub Issue



Sentinel Security Incident #6

Open

thomasvochten opened now

Security Incident Detected

Timestamp: 2025-05-07T09:37:52.69Z
Severity: Medium

Error: AppSec - VIRUS DETECTED

Details:
No details available.

Entities:

This issue was automatically generated by Microsoft Sentinel.

Create sub-issue

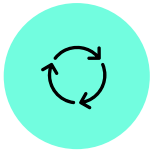
thomasvochten added runtime-attack security-log autogenerated now

Possibilities are endless

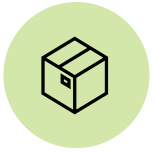
- Enrich IPs with Threat Intelligence
- Enrich incidents with geolocation or tags
- Send adaptive cards to Teams for interactivity
- Auto-close low-risk incidents
- Add a comment & assign to a specific developer
- Block IP or domain in your (web application) firewall
- Switch a feature flag based on certain abuse
- Generate a summary PDF and save to SharePoint
- Tag the application version that was live during the attack
- Annotate code in GitHub
- …

What about the costs?

Balancing insights versus cost and noise

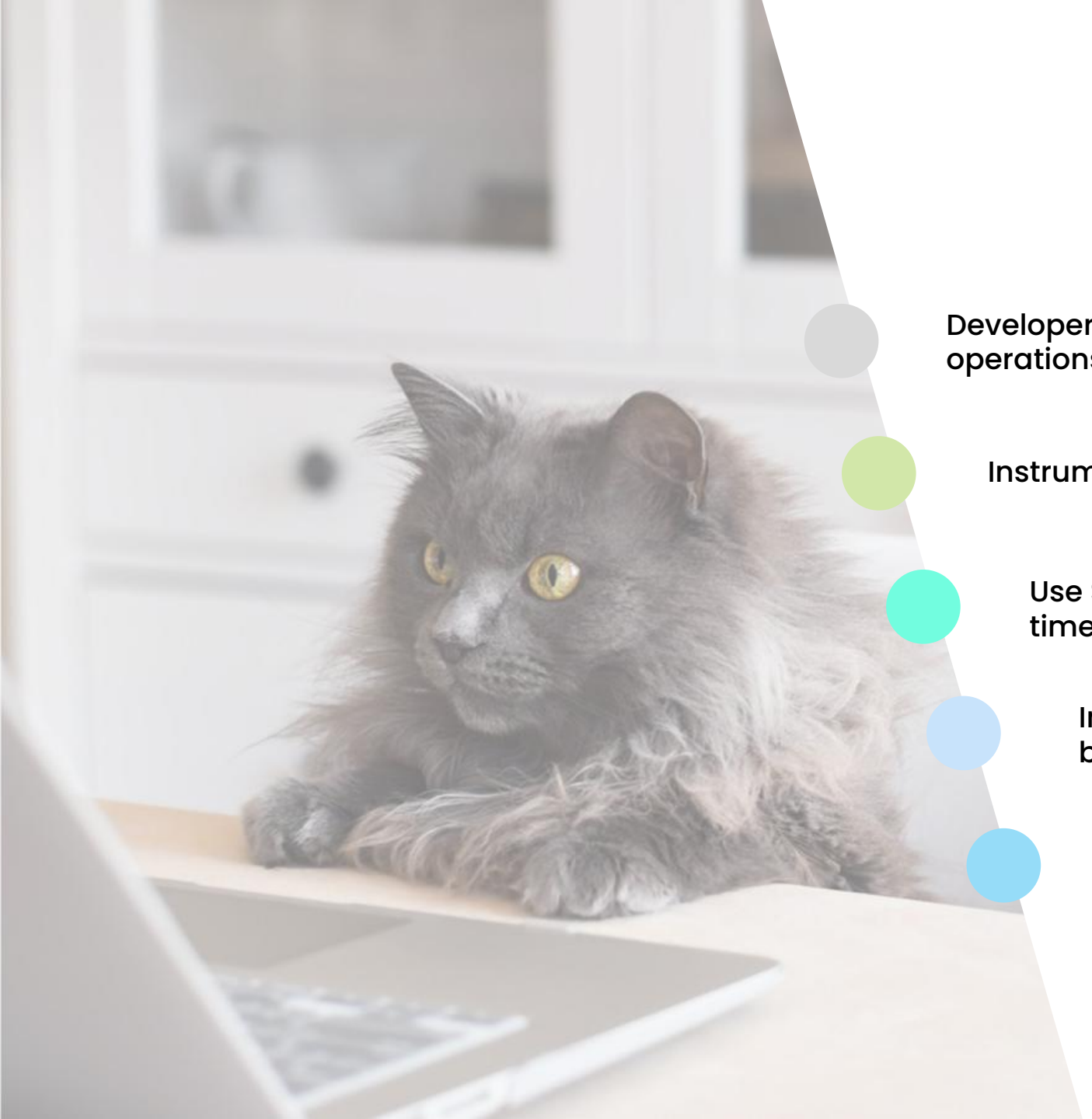


Managing data ingestion effectively



Optimizing log retention policies

Pay-as-you-go vs commitment tier pricing



Key takeaways

- Developers play a vital role in modern security operations
- Instrument your app for security-relevant logging
- Use Sentinel to detect and respond in near real-time
- Integrate the intelligence that Sentinel provides back in your development cycle (SDL)
- Start small: log smart, query smarter, automate what matters