



1ST EDITION

Security Orchestration, Automation, and Response for Security Analysts

Learn the secrets of SOAR to improve MTTA and MTTR
and strengthen your organization's security posture



BENJAMIN KOVACEVIC

Foreword by Nicholas DiCola, Vice President of Customers, Zero Networks

Security Orchestration, Automation, and Response for Security Analysts

Learn the secrets of SOAR to improve MTTA and MTTR
and strengthen your organization's security posture

Benjamin Kovacevic



BIRMINGHAM—MUMBAI

Security Orchestration, Automation, and Response for Security Analysts

Copyright © 2023 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing or its dealers and distributors, will be held liable for any damages caused or alleged to have been caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

Group Product Manager: Pavan Ramchandani

Publishing Product Manager: Prachi Sawant

Senior Editor: Athikho Sapuni Rishana

Technical Editor: Arjun Varma

Copy Editor: Safis Editing

Project Coordinator: Ashwin Kharwa

Proofreader: Safis Editing

Indexer: Hemangini Bari

Production Designer: Nilesh Mohite

Marketing Coordinator: Shruthi Shetty, Marylou De Mello

Business Development Executive: Prathamesh Walse

First published: July 2023

Production reference: 1230623

Published by Packt Publishing Ltd.

Livery Place

35 Livery Street

Birmingham

B3 2PB, UK.

978-1-80324-291-0

www.packtpub.com

Contributors

About the author

Benjamin Kovacevic is a cybersecurity enthusiast with hands-on experience with Microsoft XDR and SIEM platforms. Currently working with Microsoft Sentinel as a product manager, he focuses on the SOAR component of Microsoft Sentinel and works on new capabilities that help SOC's improve their investigations and responses. Benjamin constantly works to improve his knowledge about cybersecurity and also shares his knowledge about Microsoft SOAR. He is the author of Microsoft Sentinel Automation training blog, as well as many other blog posts containing tips and tricks to get started quickly with Microsoft Sentinel Automation.

Benjamin is originally from Bosnia and Herzegovina, but he currently resides in Ireland with his wife and two sons.

I want to thank my wife, Dzenana, and my sons, Adi and Mak. Thank you for all the sacrifices you have made and for supporting me through this journey. Also, a big thanks to all the people who have made a big impact on my security journey!

5

Introducing Microsoft Sentinel Automation

In the previous chapter, we introduced a few SOAR tools and some of the main features we can utilize in our day-to-day operations. We showcased what incident management, investigation, automation, and reporting look like in real tools and offered some directions on how to utilize them.

This chapter will focus on Microsoft Sentinel automation, and we will dive deep into each element when working with it. We will discuss automation rules, playbooks, their elements and permissions, and prepare you for hands-on examples that will be covered in *Chapters 6 to 8*.

In this chapter, we will discuss the following:

- The purpose of Microsoft Sentinel automation
- All about automation rules
- All about playbooks
- Monitoring automation rules and playbook health

The purpose of Microsoft Sentinel automation

Microsoft Sentinel automation's purpose, like the purpose of all automation, is to take repetitive tasks and transform them into automated tasks. In the SOC, some of the topics automation focuses on are as follows:

- **Enrichment:** When an incident is created, we want to enrich it with additional data. This will save SOC analysts time as they will have this enrichment as soon as they pick up the incident. For example, when an incident with an IP address is created, we can run an automatic playbook to enrich the incident with TI data about whether the IP is known to be malicious or not.

- **Initial triage and incident suppression:** This accompanies enrichment as we can utilize the results of that to decide whether we want to auto-close an incident if the IP is internal and behavior is expected, or transfer it to tier 2 if the IP address is known to be malicious.
- **Orchestration:** This is more oriented to orchestrating incident assignments or notifying SOC analysts that an incident has been created or assigned to them. For example, we can create automation that will send a Microsoft Teams chat message to the user using adaptive cards on incident creation, from where we can utilize SOC analyst input to even auto-close incidents if the information that's shared warrants making that decision.
- **Response:** SOC analysts used to struggle when they needed to block an IP address, isolate a machine, block a user or reset their password, and so on. Because SOC analysts wouldn't usually have permission to access a firewall, active directory, or EDR solution, they would need to raise an internal ticket or ping the network or system administrator to help out. By running a playbook, that task can be done for them automatically. This is crucial for many modern threats, where the time it takes to contain the threat must be minimal.

To perform any of the preceding tasks, Microsoft Sentinel uses two different automation methods:

- **Automation rules:** These are used to manage automation in Microsoft Sentinel centrally. Automation rules contains triggers, conditions, and actions that dictate how an automation rule will respond. In the next section, *All about automation rules*, we will dive deep into this feature.
- **Playbooks:** Playbooks are a list of actions that will be performed on an incident. This can include enrichment, response, remediation, and much more. We will cover playbooks in more detail later, in the *All about playbooks* section.

All about automation rules

As mentioned previously, automation rules can be created to manage automation in Microsoft Sentinel centrally. But how can we do that?

Automation rules in Microsoft Sentinel have three main aspects:

- Triggers
- Conditions
- Actions

Automation rules are sorted in order, which is a critical element since all automation rules will run from the lowest order number (for example, 1) to the highest (for example, 55), and they will run sequentially.

However, before we go into more detail about triggers, conditions, and actions, let's familiarize ourselves with the **graphical user interface (GUI)** of Microsoft Sentinel automation rules and learn more about permissions.

Navigating the automation rule GUI

Microsoft Sentinel automation rules are located under the **Automation** tab in the **Automation rules** sub-menu. In this menu, we have the option to create an automation rule, edit an automation rule, enable or disable an automation rule, move it up or down, remove an automation rule, as well as filter automation rules by analytic rules, actions, triggers, statuses, who created them, and when they were last modified.

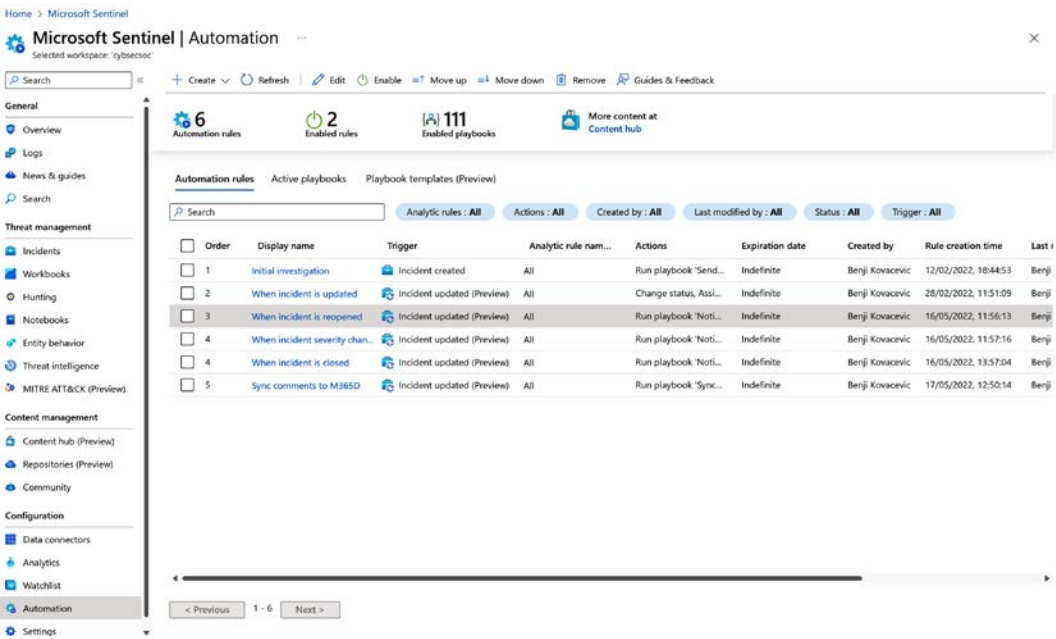


Figure 5.1 – Microsoft Sentinel automation rules

We need to click on **Create** and select **Automation rule** to create an automation rule.

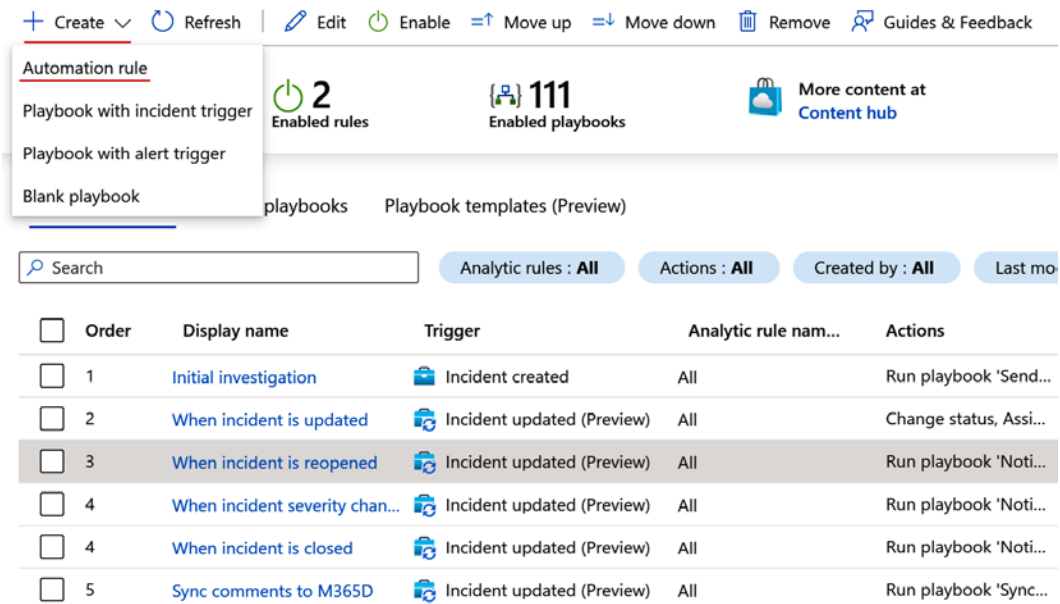


Figure 5.2 – Creating a new automation rule

From here, we must navigate to the **Create new automation rule** window; this is where we can start creating this new rule.

Create new automation rule

×

Automation rule name

Trigger

When incident is created

▼

Conditions

If

Analytic rule name

Contains

▼

All

▼

+ Add

▼

Actions ⓘ

▼



+ Add action

Rule expiration ⓘ

Indefinite



Time

Order ⓘ

2

Apply

Cancel

Figure 5.3 – The Create new automation rule wizard

However, there are other ways to create or edit automation rules:

- When creating an analytic rule using the **Analytics rule wizard** area, under the **Automated response** tab, we can see what automation rules will be triggered when an incident is created. We can also create a new automation rule specific to this rule. We can choose between any trigger at this stage.

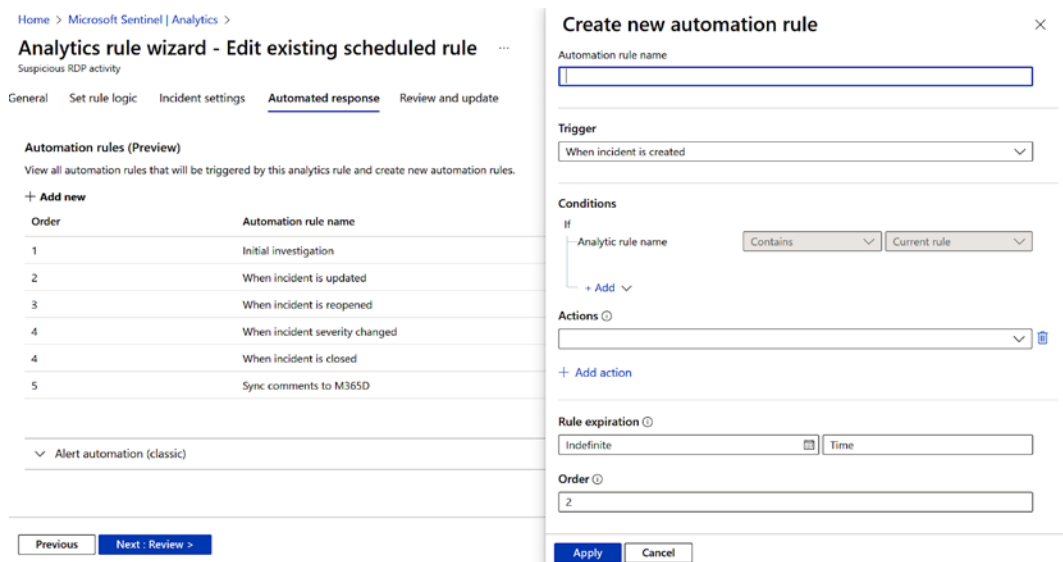


Figure 5.4 – Creating an automation rule when creating the analytic rule

- From the **Incidents** page, we can select an incident and, from the right menu, under **Actions**, select **Create automation rule**.

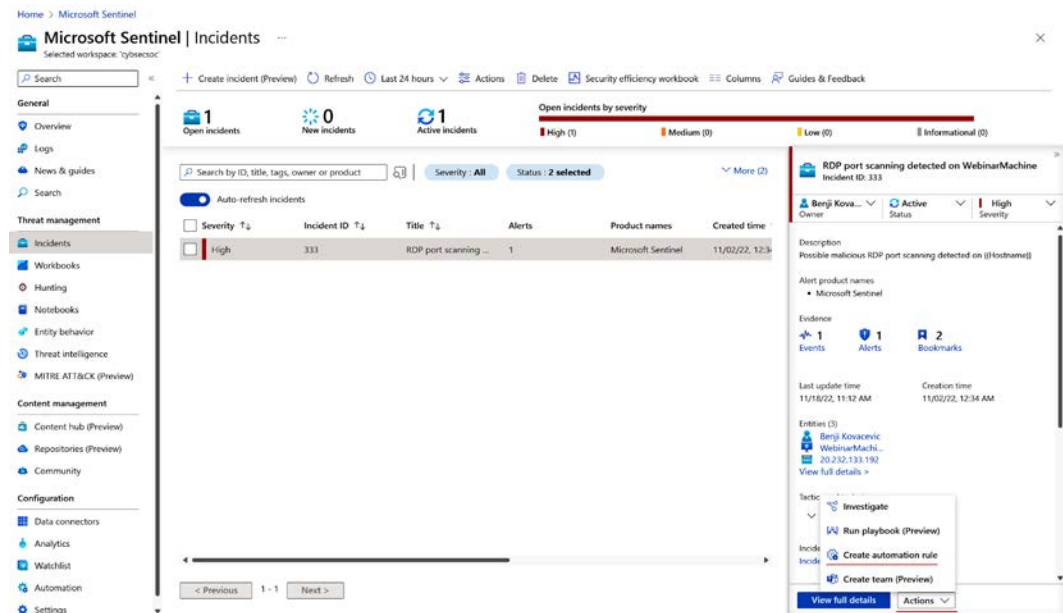


Figure 5.5 – Creating an automation rule from the Incidents page

Using this method, automation rule conditions will be filled with the data that was detected in the incident itself.

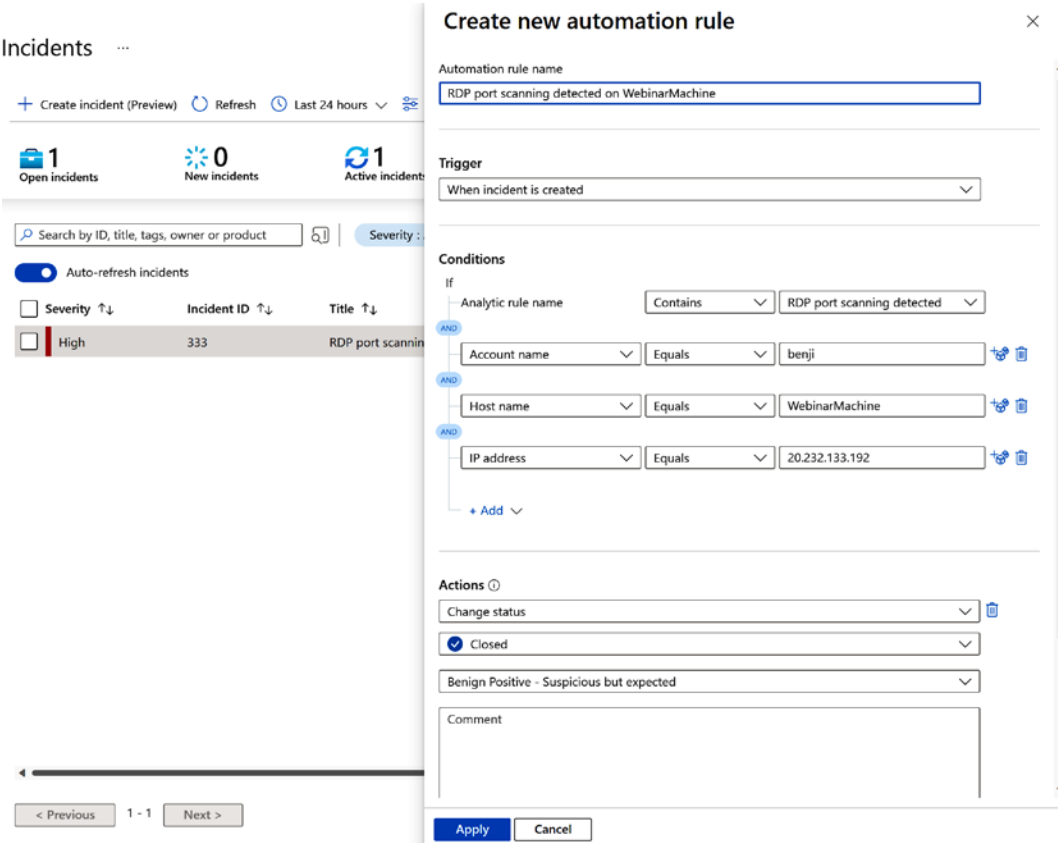


Figure 5.6 – Automation rule configuration

- We can follow the same steps to create an automation rule from the **Incident investigation** page.

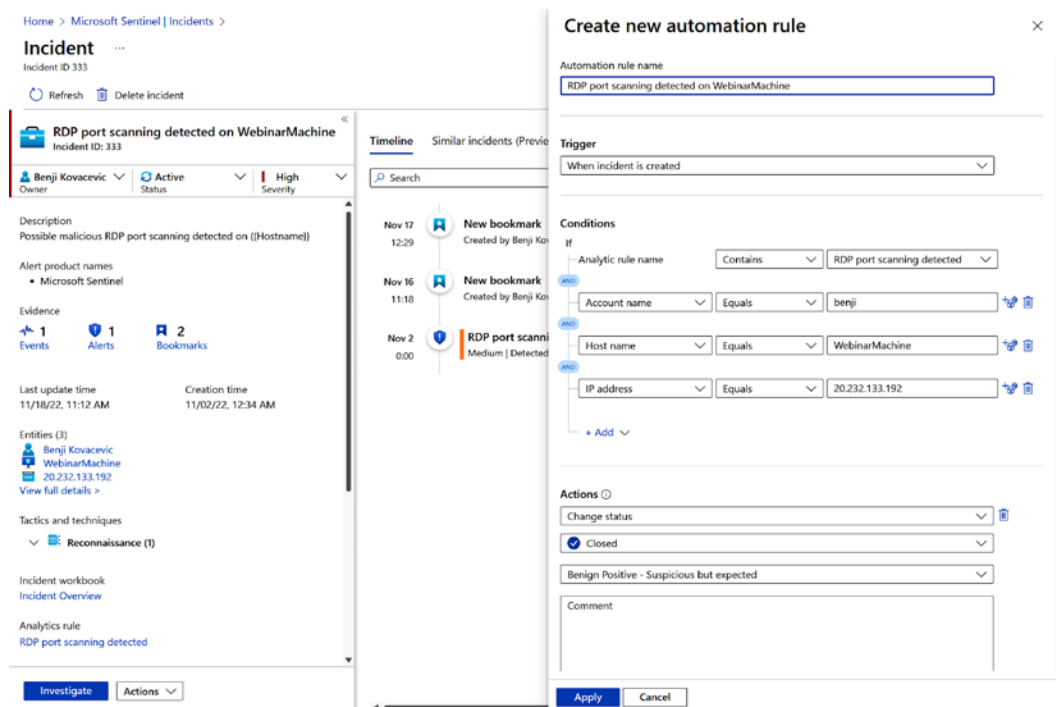


Figure 5.7 – Creating an automation rule from the Incident Overview page

To be able to create automation rules or playbooks, users must have the right permissions to create or edit them. Let's go through the permissions for Microsoft Sentinel automation.

Permissions

To create automation rules, a user needs to have a **Microsoft Sentinel Responder** or **Microsoft Sentinel Contributor** role assigned.

There is one more special role connected to Microsoft Sentinel automation rules – **Microsoft Sentinel Automation Contributor**. This is not a user role but instead a role that needs to be assigned to a Microsoft Sentinel identity so that an automation rule can run a playbook as an action. This is assigned to Microsoft Azure resource groups, which is where playbooks reside. For example, if we have five resource groups that contain playbooks, we want to have the option to attach them as an action; we need to assign this permission to all five Microsoft Azure resource groups.

To assign this permission, we need to go to the Microsoft Sentinel instance, go to **Settings**, then **Settings** again, and then, under **Playbook permissions**, click on **Configure permissions**. In the next window, choose the resource groups you want to assign the **Microsoft Sentinel Automation Contributor** role.

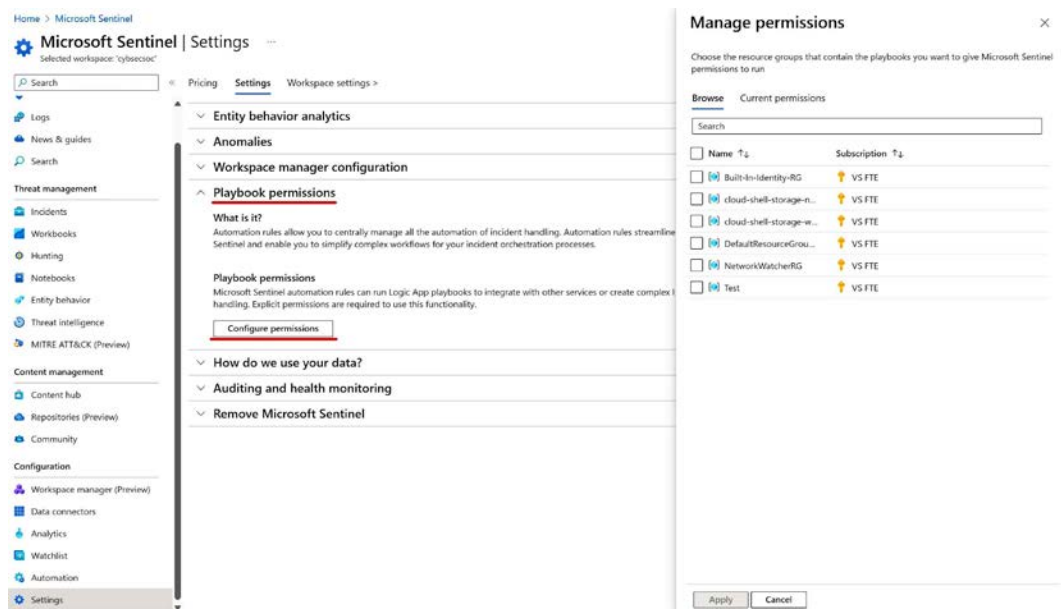


Figure 5.8 – Microsoft Sentinel playbook permissions configuration

Once the permission is applied on the resource group level, we can attach playbooks from that resource group to an automation rule as an action.

Triggers

Triggers are used to define when an automation rule runs. For automation rules, we have three different triggers:

- **When incident is created:** This supports a complete list of conditions
- **When incident is updated (Preview):** This supports a complete list of conditions, plus conditions associated with information about updated data
- **When alert is created (Preview):** This supports only one condition (an analytic rule name)

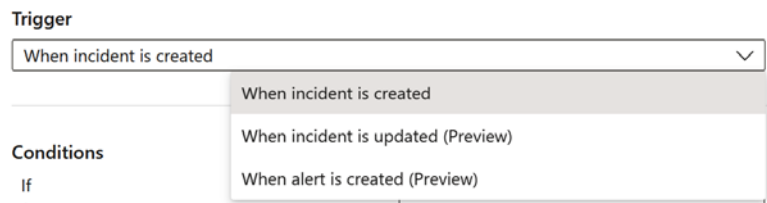


Figure 5.9 – Automation rule triggers

Conditions

Once a trigger has been set, we need to set conditions. Conditions are used to filter what incidents we want to run specific actions on, as we don't want the same actions on all incidents. What's important to note here is that actions will run only if all conditions are met. Automation rules support **OR** and **AND** condition grouping, which allows us to create more detailed automation.

An analytic rule name is one condition that cannot be removed and is used across all triggers. The evaluation supports **Contains** and **Does not contain** options, while for values, we can choose all analytic rules or run on only specific analytic rules created in Microsoft Sentinel. When we choose **All** as a value for incident creation and update, this will also run on synchronized incidents from tools such as Microsoft 365 Defender and Microsoft Defender for Cloud.

Since we have multiple triggers in automation rules, let's see what conditions are supported for each.

Conditions associated with the "When incident is created" trigger

The **When incident is created** trigger can only check the current state of the values of an incident. If we evaluate the same incident in multiple automation rules, the current state can change if we update the incident in a previous automation rule. For example, if the severity of incident creation is set to **Medium**, that will be the current state for the first automation rule. Suppose, in the automation rule, we take action to change the severity to **High**. In that case, the current value for severity in the following automation rule (that is, automation rule number 3) that will run on this same incident will be **High**.

The following conditions can be used with the **When an incident is created** trigger:

- **Incident properties:** For example, analytic rule name, title, description, severity, owner, status, tactics, and custom details
- **Entity properties:** For example, account name, account domain, filename, file hash, hostname, IP address, IoT device, mail message details, URL, and many more

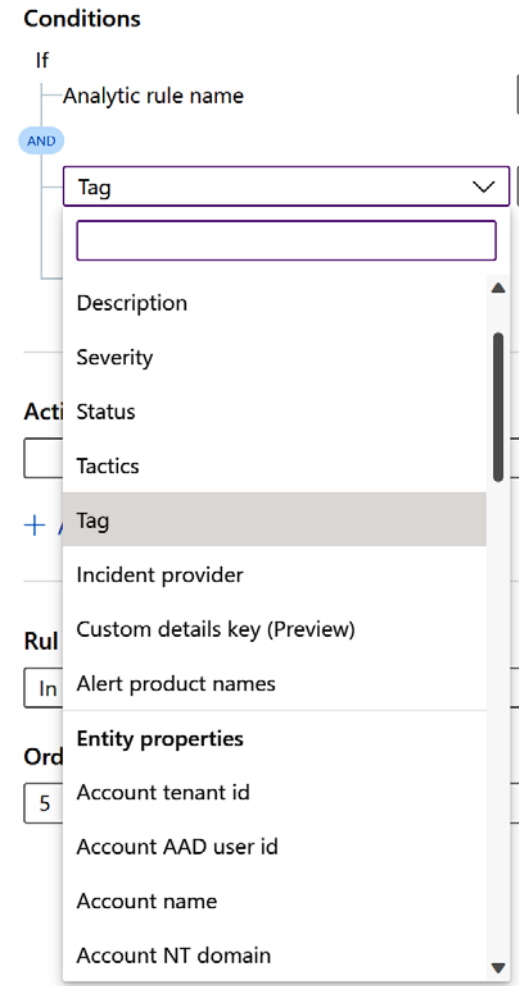


Figure 5.10 – Automation rule condition values

Based on the condition selected, we can use one of the following evaluation methods:

- **Equals** or **Does not equal**
- **Contains** or **Does not contain**
- **Starts with** or **Does not start with**
- **Ends with** or **Does not end with**

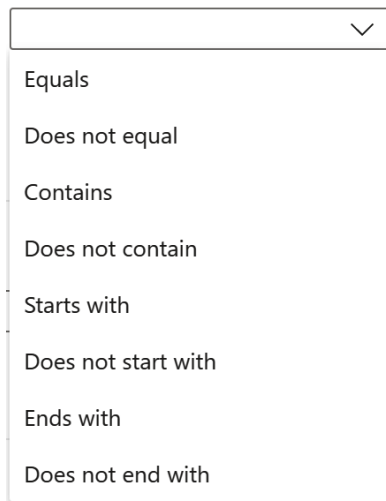


Figure 5.11 – Automation rule condition options

The final element of the condition is the value itself. Here, we can have clear text input (such as an incident title or description) or the option to select pre-existing values (such as incident severity or status). We can also add multiple inputs to the same condition.

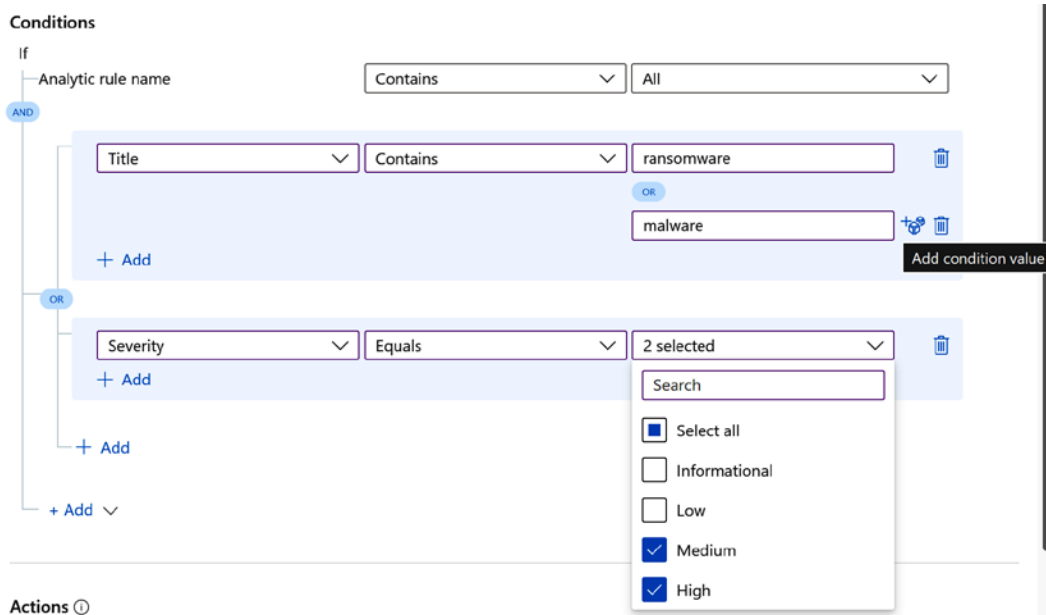


Figure 5.12 – An automation rule condition example

Conditions associated with the “When incident is updated” trigger

If we update an incident with one of the supported conditions, we can trigger automation rules based on that. This gives us complete control over automation scenarios to support incident creation and updates.

Conditions with an incident update trigger support **current state** and **state changes**. Conditions include those from the incident creation trigger, plus the following:

- **Owner**
- **Updated by**
- **Alerts**
- **Comments**

They also include **state change** values for the following:

- **Severity**
- **Status**
- **Tactics**
- **Tag**

The following evaluation methods can be added with **state change**:

- **Changed** (owner, severity, and status)
- **Changed To** (severity and status)
- **Changed From** (severity and status)
- **Added** (alerts, comments, tactics, and tags)

The only difference is **Updated by**, which uses the current state for evaluation. We can select one of the following values from the dropdown:

- **Application**
- **User** (a manual change of a field by a specific user)
- **Alert grouping** (adding an alert to the incident)
- **Playbook** (a change made by a playbook run)
- **Automation rule** (a change made by an automation rule run)
- **Microsoft 365 Defender** (for updates to incidents made by bidirectional incident synchronization from Microsoft 365 Defender)

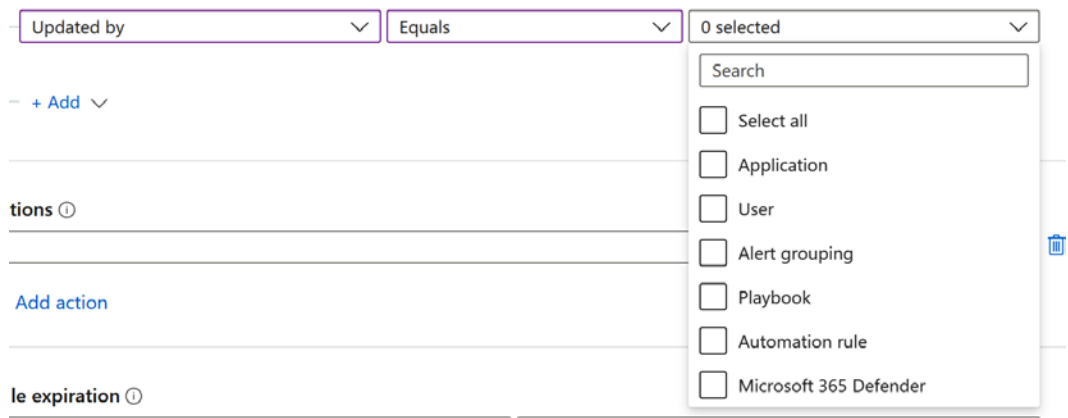


Figure 5.13 – The Updated by automation rule values

Conditions associated with the “When alert is created” trigger

The **When alert is created** trigger only supports analytic rules created in Microsoft Sentinel, and the only supported condition is **Analytic rule name**.

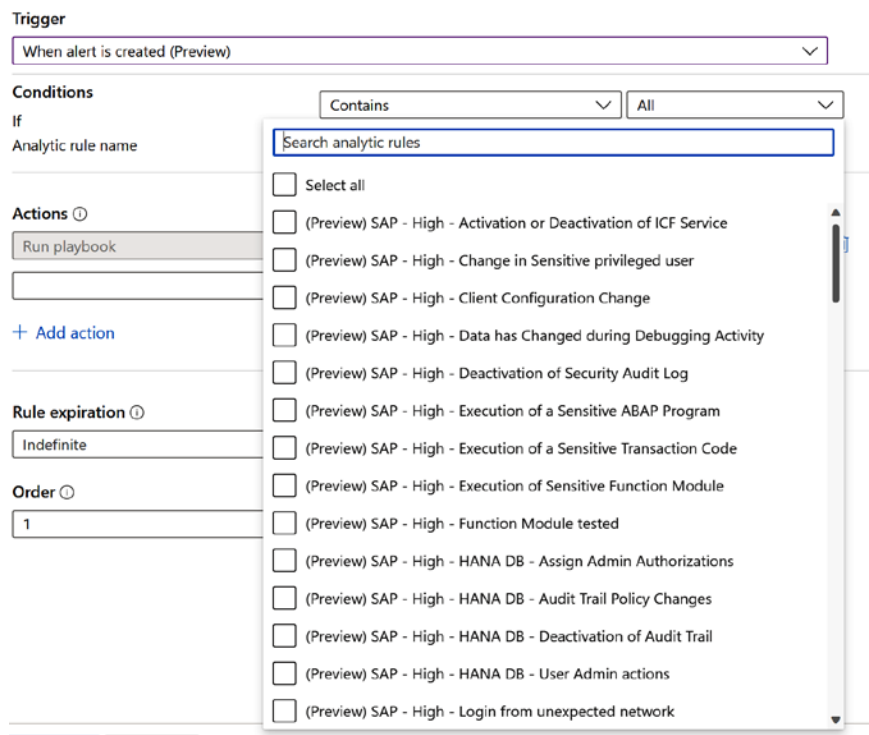


Figure 5.14 – An automation rule alert creation condition

After choosing one of the triggers (incident creation, incident update, or alert creation) and configuring the conditions, if all conditions we configured are met, we can run one or more actions on an incident.

Actions

The following actions are supported for automation rules:

- **Run playbook:**

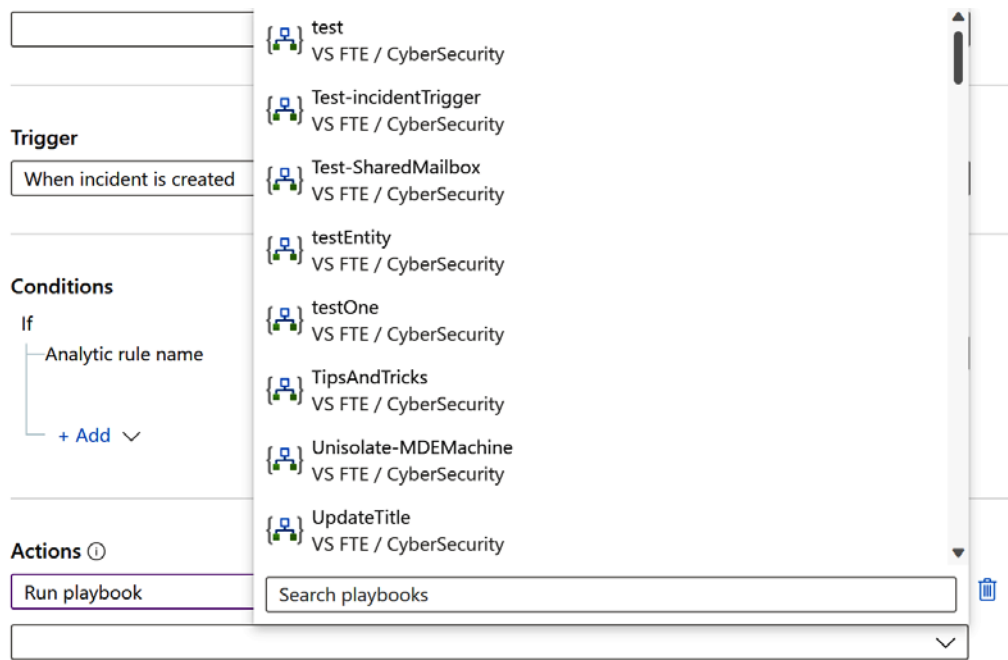


Figure 5.15 – Run playbook action

- **Change status:**

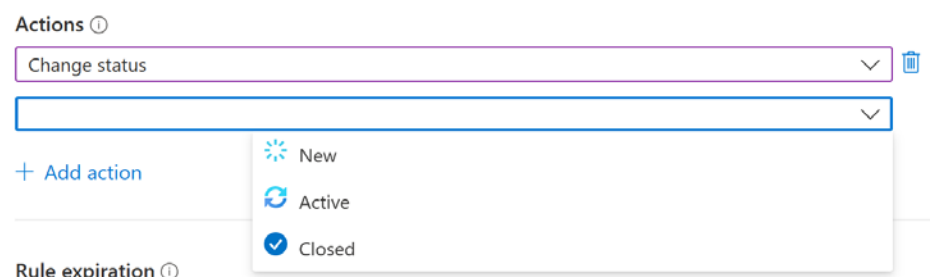


Figure 5.16 – Change status action

- **Change severity:**

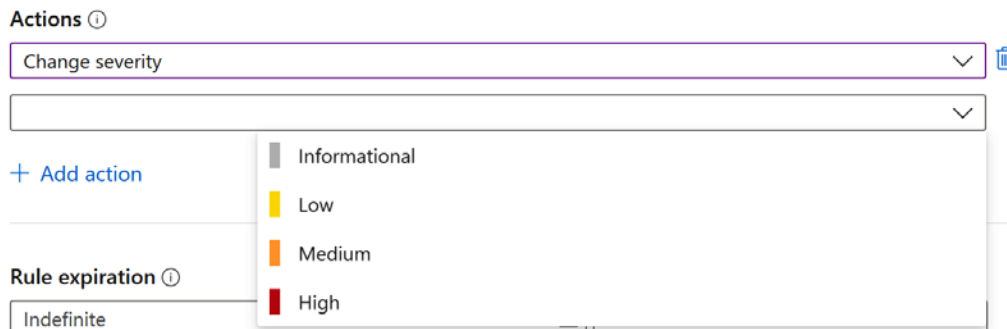


Figure 5.17 – Change severity action

- **Assign owner:**

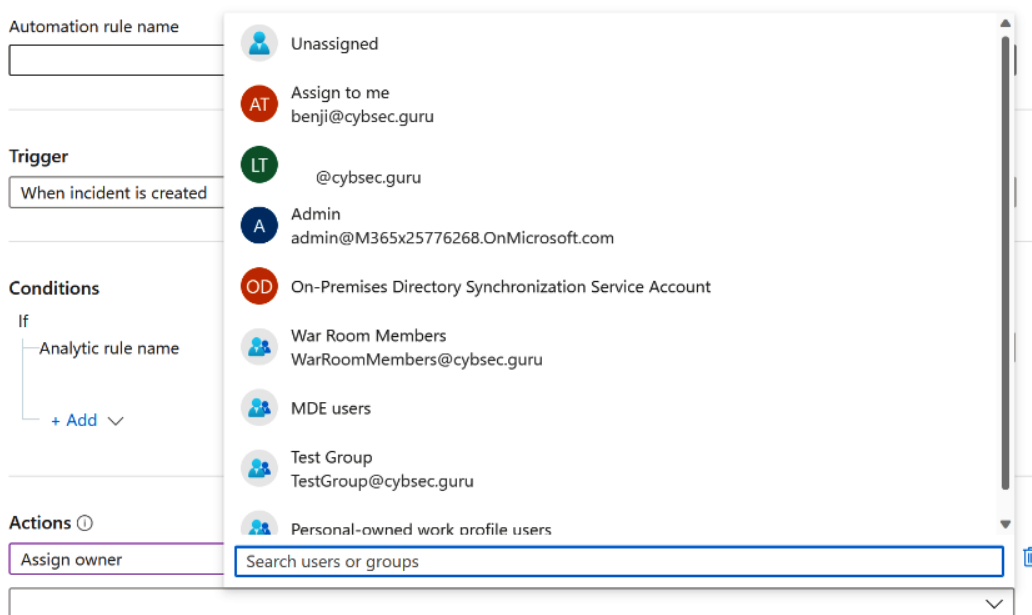


Figure 5.18 – Assign owner action

- **Add tags:**

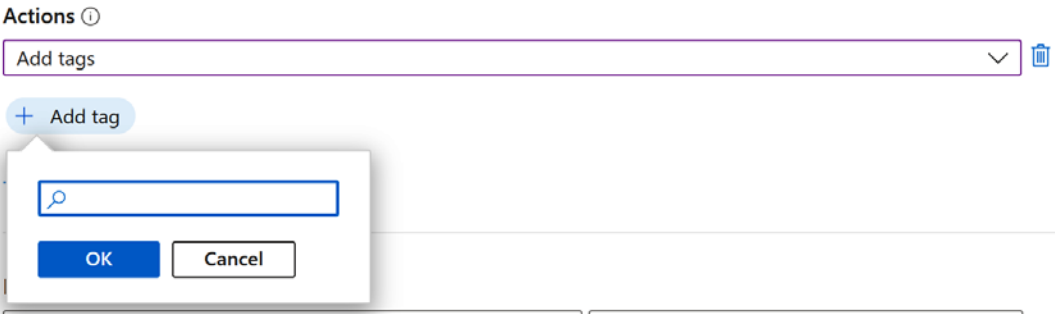


Figure 5.19 – Add tags action

Important note

The **Run playbook** action is the only action that's available when utilizing the **When alert is created** trigger.

While automation rule triggers, conditions, and actions are fields, we will always want to configure to have effective automation rules; we can also configure rule expiration and order.

Rule expiration and order

As mentioned previously, we have two additional steps we can configure when creating automation rules:

- **Rule expiration:** This is if we are creating an automation rule that will be active for only a specific period – for example, if we are performing penetration testing. We want to auto-close incidents created by and during penetration testing and disable the automation rule at a specific date and time.
- **Order:** This is what execution order number we want to configure an automation rule as. To recall, all automation rules are run via an order number, from a lower number to a higher one, sequentially.

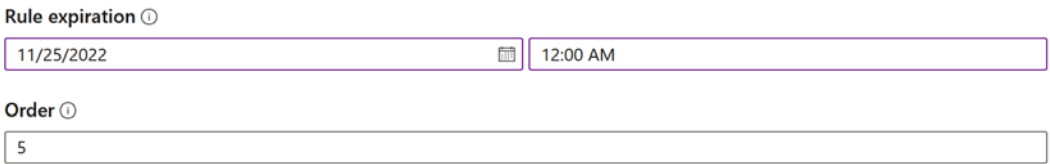


Figure 5.20 – Automation rule expiration and order configuration

With that, we have covered all the major elements of automation rules and how they work. In the next section, we will look at playbooks and their main building blocks in Microsoft Sentinel automation in more detail.

All about playbooks

Playbooks are a list of actions that will be performed on the incident. They can include enrichment, response, remediation, and much more. To achieve this, Microsoft Sentinel utilizes a Microsoft Azure solution called **Logic Apps** – a platform used to create and run automated workflows. This platform uses low- or no-code and focuses more on visual design. However, those who prefer to code more can utilize coding mode as well. Because of this, it is common to hear people refer to Microsoft Sentinel playbooks as Logic Apps.

There are two different types of Logic Apps that Microsoft Sentinel supports:

- **Logic Apps Consumption:** This is a single playbook that has only one workflow. It supports templates and custom connectors and is widely integrated into Microsoft Sentinel with template support. Logic Apps Consumption shares the same backend resources across different customer tenants. We will use the Logic Apps Consumption model in our hands-on examples.
- **Logic Apps Standard:** This is a single Logic App that can have multiple workflows. It doesn't support templates and custom connectors, which is why Microsoft Sentinel doesn't have playbook templates created in Logic Apps Standard. In Logic Apps Standard, workflows in the same Logic App share the same backend resources, and they are not shared across different Logic Apps like they are with Logic Apps Consumption. It's also important to note that when creating a Logic Apps Standard playbook, it must be stateful and cannot utilize private endpoints – Microsoft Sentinel does not support these scenarios at the time of writing.

Microsoft Sentinel is a unified way to run a playbook, and it will make no difference whether Logic Apps Consumption or Logic Apps Standard is used.

Navigating the playbooks GUI

Microsoft Sentinel playbooks are located under the **Automation** tab in the **Active playbooks** sub-menu. In this menu, we have the option to create a playbook, open playbook details to edit or manage it, enable or disable a playbook, delete a playbook, as well as to filter playbooks by status, trigger kind, subscription, resource group, plan, and source name. If we have deployed the playbook using built-in templates, we will also get information on whether an update is available.

The screenshot displays the Microsoft Sentinel Automation interface. The left sidebar contains navigation links for General (Overview, Logs, News & guides, Search), Threat management (Incidents, Workbooks, Hunting, Notebooks, Entity behavior, Threat intelligence, MITRE ATT&CK), Content management (Content hub, Repositories, Community), Configuration (Data connectors, Analytics, Watchlist), and Automation (Settings). The main panel shows the 'Automation rules' section with a sub-menu for 'Active playbooks'. At the top, there are counts for Automation rules (6), Enabled rules (2), and Enabled playbooks (110). Below this, a table lists the active playbooks with columns for Name, Status, Plan, Trigger kind, Subscription, Resource group, Location, Source name, and Tags. The table shows 11 playbooks, including 'UPDATE AVAILABLE', 'UPDATE AVAILABLE', 'test', 'Test-alertTrigger', 'test-createWatchlist', 'Test-incidentTrigger', 'Test-SharedMailbox', 'testEntity', 'testOne', 'TipsAndTricks', 'Unisolate-MDEMachine', 'Update-VIPUsers-Wat...', 'Update-Watchlist-Wit...', and 'UpdateTitle'. The first two rows have a yellow 'UPDATE AVAILABLE' badge. The table is filtered by Status: All, Trigger kind: All, Subscription: VS FTE, Resource group: All, Plan: All, and Source name: All. The page number is 1 of 3.

Name	Status	Plan	Trigger kind	Subscription	Resource group	Location	Source name	Tags
UPDATE AVAILABLE	Enabled	Consumption	Microsoft Senti...	VS FTE	CyberSecurity	East US	Gallery Content	LogicAppsCatego...
UPDATE AVAILABLE	Enabled	Consumption	Microsoft Senti...	VS FTE	CyberSecurity	East US	Gallery Content	LogicAppsCatego...
test	Disabled	Consumption	Microsoft Senti...	VS FTE	CyberSecurity	East US	Custom Content	
Test-alertTrigger	Enabled	Consumption	Microsoft Senti...	VS FTE	CyberSecurity	East US	Custom Content	
test-createWatchlist	Enabled	Consumption	Microsoft Senti...	VS FTE	CyberSecurity	East US	Custom Content	
Test-incidentTrigger	Enabled	Consumption	Microsoft Senti...	VS FTE	CyberSecurity	East US	Custom Content	
Test-SharedMailbox	Enabled	Consumption	Microsoft Senti...	VS FTE	CyberSecurity	East US	Custom Content	
testEntity	Enabled	Consumption	Microsoft Senti...	VS FTE	CyberSecurity	East US	Custom Content	
testOne	Enabled	Consumption	Microsoft Senti...	VS FTE	CyberSecurity	East US	Custom Content	
TipsAndTricks	Enabled	Consumption	Microsoft Senti...	VS FTE	CyberSecurity	East US	Custom Content	
Unisolate-MDEMachine...	Enabled	Consumption	Microsoft Senti...	VS FTE	CyberSecurity	East US	Custom Content	LogicAppsCatego...
Update-VIPUsers-Wat...	Enabled	Consumption	Using Microsoft...	VS FTE	CyberSecurity	East US	Custom Content	LogicAppsCatego...
Update-Watchlist-Wit...	Enabled	Consumption	Using Microsoft...	VS FTE	CyberSecurity	East US	Custom Content	
UpdateTitle	Enabled	Consumption	Microsoft Senti...	VS FTE	CyberSecurity	East US	Custom Content	

Figure 5.21 – Microsoft Sentinel playbooks

Playbooks support templates; all deployed templates can be found in the **Playbooks templates (Preview)** sub-menu. We can deploy any playbook, from templates to active state. We can filter templates by trigger, Logic App connector, entities, tags, and source name. If we have already deployed a playbook template, we will see a notification stating that a specific playbook is in use.

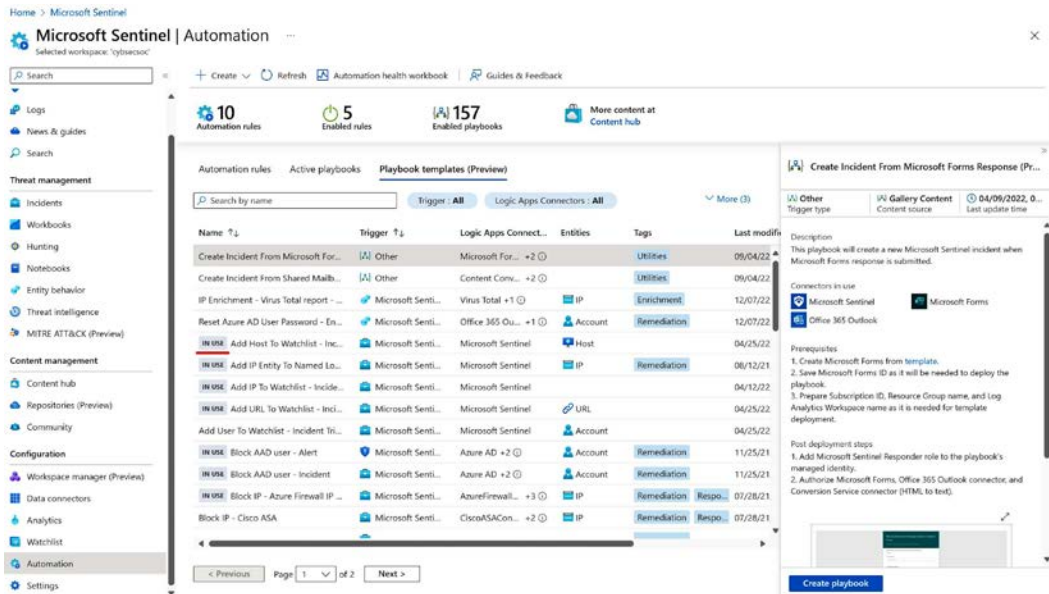


Figure 5.22 – Microsoft Sentinel playbook templates

To access all templates in Microsoft Sentinel, we can utilize **Content hub** and the available solutions, where we can filter, among others, by the solution we need or solutions with playbook templates.

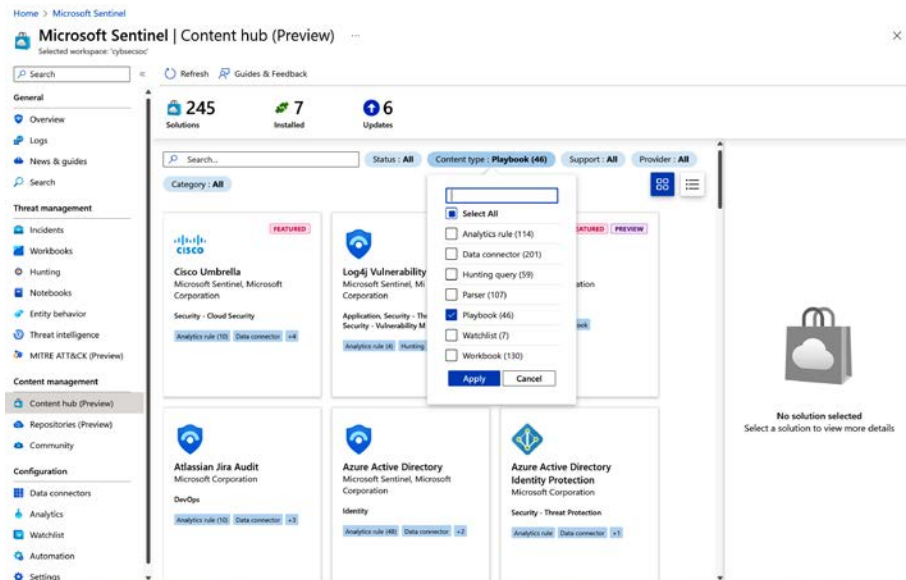


Figure 5.23 – Microsoft Sentinel – Content hub

More templates can be found on GitHub and can be easily deployed to Microsoft Sentinel since they utilize **Azure Resource Manager (ARM)** templates. Microsoft Sentinel has an official repository with lots of content available that is ready to be deployed. The link to the official repository is <https://github.com/Azure/Azure-Sentinel>.

To create a new playbook, go to the **Automation** tab, click **Create**, and select one of the following options:

- **Playbook with incident trigger**
- **Playbook with alert trigger**
- **Blank playbook**

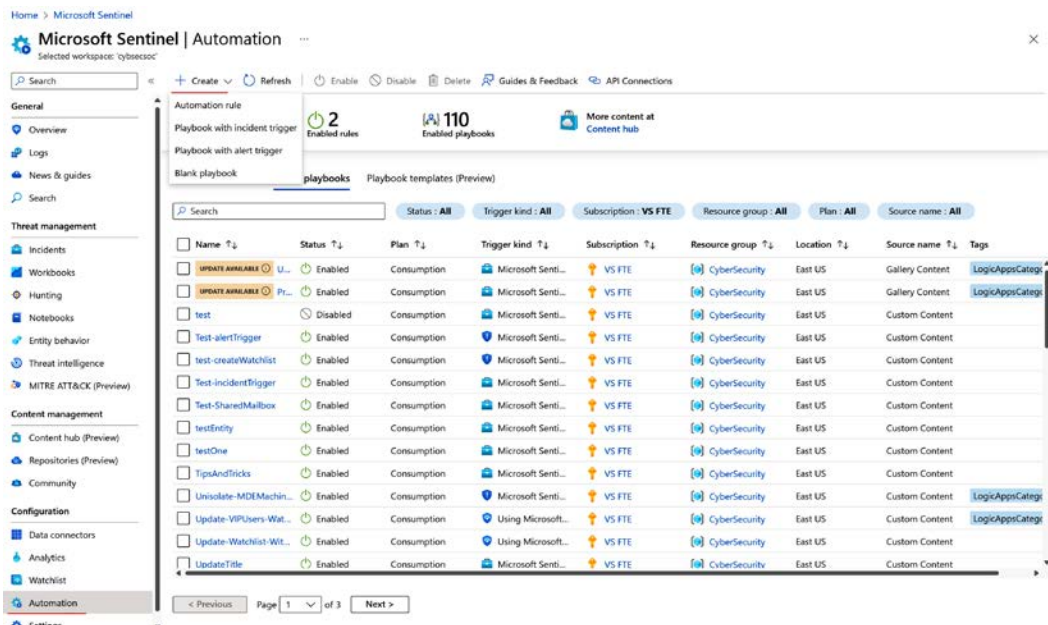


Figure 5.24 – Creating a new Microsoft Sentinel playbook

If we select **Playbook with incident trigger** or **Playbook with alert trigger**, we will create a **Logic Apps Consumption** Logic App. The first view is where we enter basic information, such as what subscription and resource we want to deploy the playbook in, the region, and the playbook's name. We can also enable diagnostic settings, which we will cover in the *Monitoring automation rules and playbook health* section:

[Home](#) > [Microsoft Sentinel | Automation](#) >

Create playbook ...

1 Basics 2 Connections 3 Review and create

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *

Resource group * [Create new](#)

Region *

Playbook name *

☐ Enable diagnostics logs in Log Analytics ⓘ

Log Analytics workspace

☐ Associate with integration service environment ⓘ

Integration service environment

[Next : Connections >](#)

Figure 5.25 – The Create playbook wizard

In the next window, we can select how we want to authenticate a Microsoft Sentinel connection. By default, playbook creation will enable a **system-assigned managed identity** from the playbook and utilize it (the recommended method). However, we can utilize any pre-existing connection or change it once the playbook has been deployed.

Home > Microsoft Sentinel | Automation >

Create playbook ...

✓ Basics **2 Connections** ③ Review and create

For each connector this playbook uses, you can choose to use an existing connection from another playbook. Otherwise, you must create a new connection and authenticate when you are brought to the Logic Apps designer after your playbook is deployed.

^

 Microsoft Sentinel

Connect with managed identity

Connect with managed identity

FailedTrigger

benji@cybsec.guru

ttt

Previous

Next : Review and create >

Figure 5.26 – Creating a new playbook – Connections

The last step is to review the configuration and create our playbook.

Home > Microsoft Sentinel | Automation >

Create playbook ...

✓ Basics ✓ Connections **③ Review and create**

Basics

Subscription	VS FTE
Resource group	CyberSecurity
Region	East US
Playbook name	SOAR
Diagnostics logs workspace	Disabled
Integration service environment	Disabled

Connections

 Microsoft Sentinel

Connect with managed identity

i **Note:** Grant permissions to the managed identity after deployment.

Previous

Create and continue to designer

Figure 5.27 – Creating a new playbook – Review and create

Once the playbook has been deployed, we can navigate to **Logic app designer**, where we can start our playbook design.

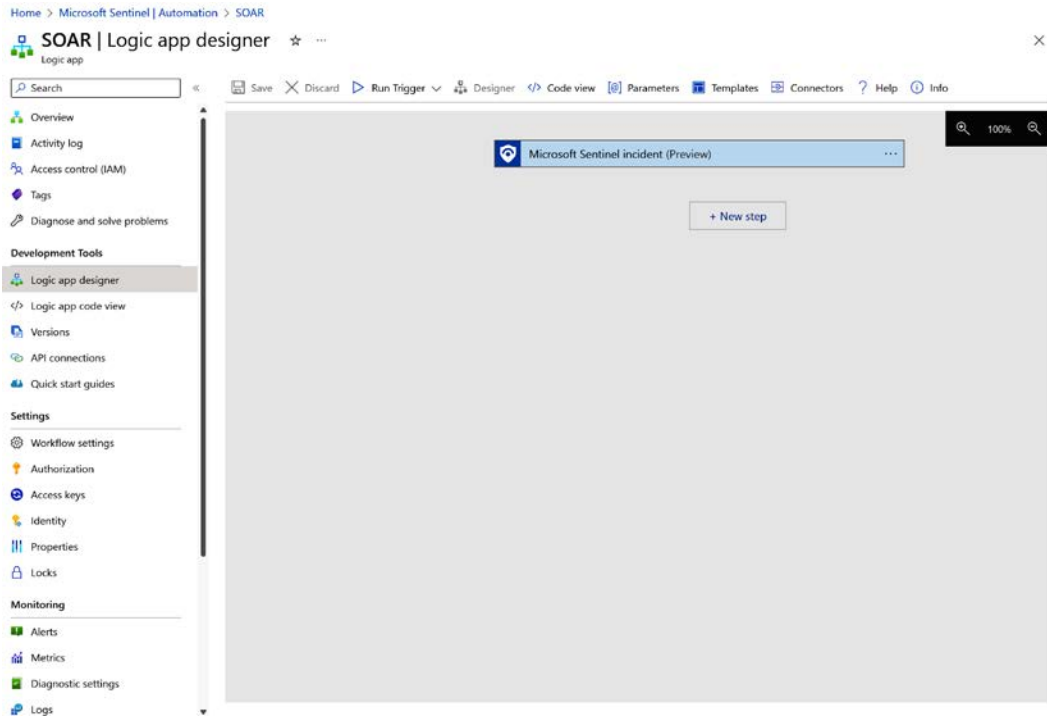


Figure 5.28 – The Logic app designer view

From here, we can also access the code view of the playbook if we prefer to work with code.

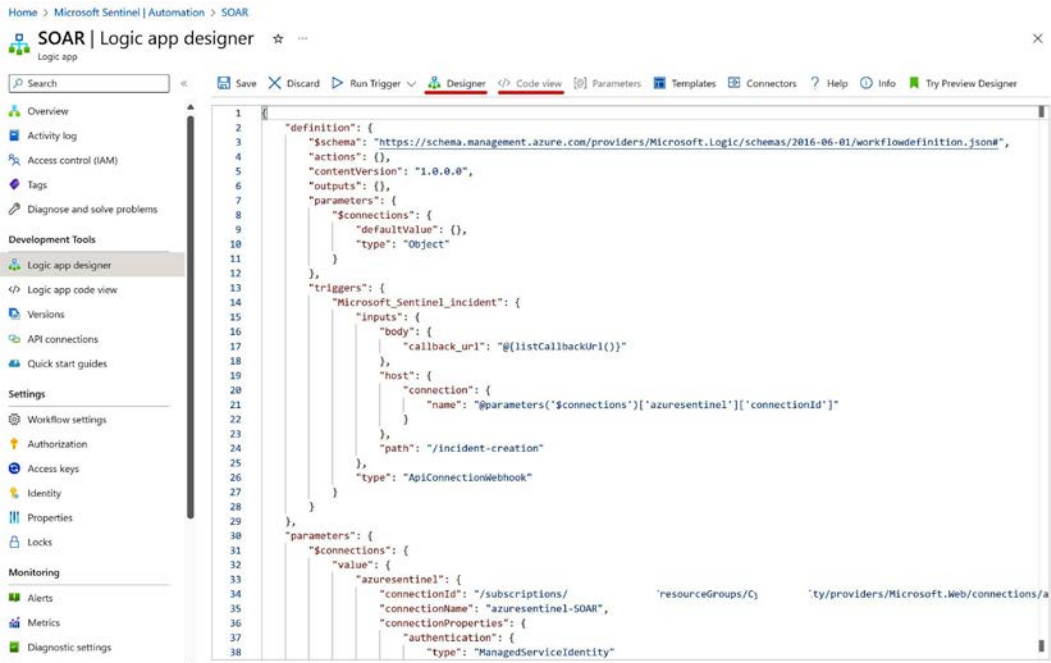


Figure 5.29 – The Logic App designer code view

We will go through the whole process of creation and explanation in *Chapters 6 to 8*, where we will cover hands-on examples.

When we want to create a blank playbook, we can choose between creating a **Logic Apps Standard** or **Logic Apps Consumption** Logic App. We can also utilize any other trigger available in Logic Apps, such as a recurrence to perform a regular playbook run, and a Microsoft Forms trigger to create an incident when a new form is filled in and when an email is received.

[Home](#) >

Create Logic App

Basics Hosting Monitoring Tags Review + create

Create a logic app, which lets you group workflows as a logical unit for easier management, deployment and sharing of resources. Workflows let you connect your business-critical apps and services with Azure Logic Apps, automating your workflows without writing a single line of code.

Project Details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ VS FTE

Resource Group * ⓘ CyberSecurity

[Create new](#)

Instance Details

Logic App name * SOARStandard .azurewebsites.net

Publish * ☒ Workflow ☐ Docker Container

Region * Central US

Not finding your App Service Plan? Try a different region or select your App Service Environment.

Plan

The plan type you choose dictates how your app scales, what features are enabled, and how it is priced. [Learn more](#)

Plan type * ☒ **Standard:** Best for enterprise-level, serverless applications, with event-based scaling and networking isolation.

☐ **Consumption:** Best for entry-level. Pay only as much as your workflow runs.

Windows Plan (Central US) * ⓘ (New) ASP-CyberSecurity-8dfe

[Create new](#)

Pricing plan * **Workflow Standard WS1**
210 total ACU, 3.5 GB memory
[Change size](#)

Zone redundancy

An App Service plan can be deployed as a zone redundant service in the regions that support it. This is a deployment time only decision. You can't make an App Service plan zone redundant after it has been deployed [Learn more](#)

[Review + create](#) [< Previous](#) [Next : Hosting >](#)

Figure 5.30 – Creating a playbook using Logic Apps Standard

To run the created playbook, we have a few options:

- Attach the playbook to an automation rule for automatic triggering (which will require the **Microsoft Sentinel Automation Contributor** role to be assigned to a Microsoft Sentinel identity; more about this in the subsequent *Permissions* section)
- Run the playbook manually on the incident (which will require the **Microsoft Sentinel Automation Contributor** role assigned to a Microsoft Sentinel identity)
- Run the playbook manually on the alert

To create, edit, and run playbooks in Microsoft Sentinel, you will need certain permissions to perform these actions. Let's go through the different permissions users can have and what users can perform with these actions.

Permissions

Important note

To understand this segment better, I suggest that you have a basic understanding of permissions on Azure and Azure RBAC. A great starting point is the official documentation: <https://learn.microsoft.com/azure/role-based-access-control/>.

There are a few different permissions that users can utilize based on the actions they need to perform when working with Microsoft Sentinel playbooks:

- **Logic Apps Contributor:** This gives you permission to manage Logic Apps and run playbooks, but you cannot change access to them (there is standard role separation in Azure, and only the **Owner** or **User Access Administrator** role can perform this action).
- **Logic App Operator:** This gives you permission to read, enable, or disable a playbook, but you cannot edit, update, or run playbooks.
- **Microsoft Sentinel Contributor:** This permits you to attach a playbook to an analytic rule, among other Microsoft Sentinel permissions.
- **Microsoft Sentinel Responder:** This permits you to run playbooks manually, among other Microsoft Sentinel permissions.
- **Microsoft Sentinel Playbook Operator:** This permits you to list and run playbooks manually.

We covered the **Microsoft Sentinel Automation Contributor** role earlier in this chapter, so we will not look at it in detail again.

To be able to create and utilize playbooks, it is important to understand how they work. In the next section, we will cover the main aspects of Logic Apps.

Logic Apps connectors and authentication

Under the hood, Logic Apps uses API calls to connect with Microsoft and non-Microsoft solutions. Those API calls can be wrapped in a Logic Apps connector, giving us more straightforward configuration and authentication. These connectors are as follows:

- **Managed connectors:** These are available in a Logic App out of the box. They contain triggers and actions for specific products or services, such as the Microsoft Sentinel connector. There are hundreds of managed connectors for Microsoft products and services, as well as for non-Microsoft products and services.
- **Custom connectors:** If some product or service still doesn't have a built-in connector in Logic Apps, you have the option to create and utilize a custom connector in your environment. It is also possible to share those custom connectors with others, and some of them are utilized with Microsoft Sentinel. When using custom connectors, you must utilize the Logic App Consumption model since Logic App Standard doesn't support custom connectors at the time of writing.

But what if there is neither a built-in nor custom connector?

In this case, we can utilize an HTTP connector that will allow us to connect to a product or solution using direct API calls. Examples of these HTTP calls will be covered in *Chapter 9*.

Important note

Data connectors in Microsoft Sentinel aren't the same as Logic Apps connectors.

Data connectors in Microsoft Sentinel are used to ingest logs into a Log Analytics workspace, and we can utilize those logs to create detection rules, hunt for data, and so on. Some examples of these logs include Syslog data, security event data from Windows Server, and sign-in logs from Azure AD.

Logic Apps connectors are API calls to products and services so that we can perform specific actions. Examples of these API calls are a call to Azure AD to block a user, a call to an EDR solution to isolate the machine, and an API call to the TI solution to get IP address information.

But wait! When making an API call, don't we need to provide authentication? Is this supported in Microsoft Sentinel playbooks?

Yes! If we need to authenticate managed or custom connectors, there is a way to do this. For non-Microsoft products and services, these can be usernames and passwords, API tokens, and so on. These authentications are saved as API connections and can be accessed from a playbook. Once created, these API connections can also be utilized in other playbooks; they are not specific to one playbook.

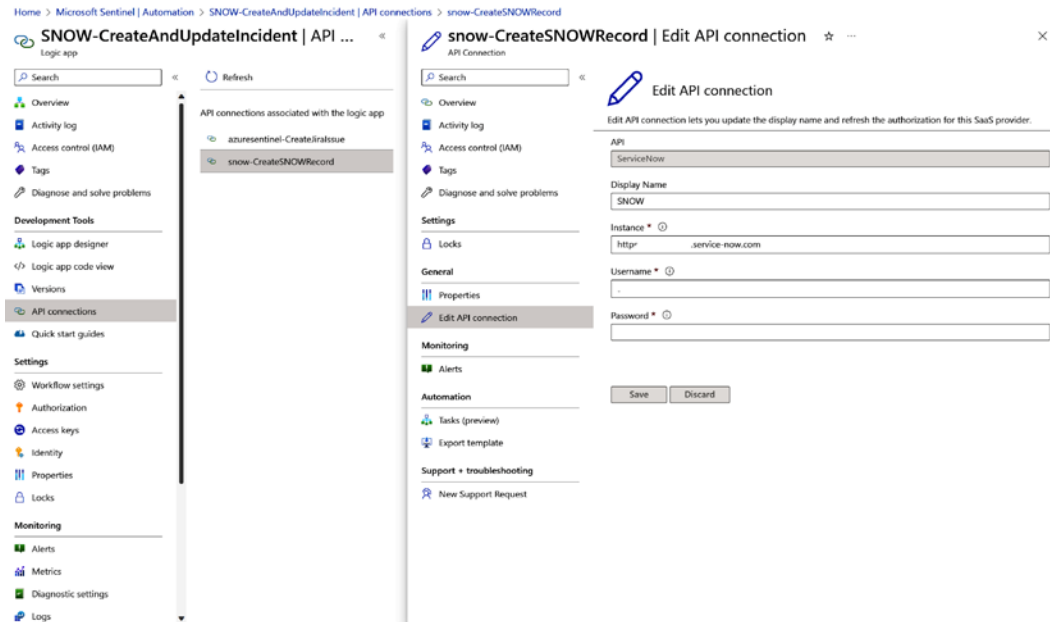


Figure 5.31 – Managing an API connection in a playbook

If we use HTTP calls, this information will be inserted into the header or body of the API call, as per the instructions of the product or service.

In terms of Microsoft services and products, playbooks support three types of API authentication. Let's look at them in detail.

System-assigned managed identity

This is the preferred option and is utilized by default when creating playbooks using **Create playbook with an incident trigger** or **Create playbook with an alert trigger**. Each playbook has its own system-assigned managed identity that can be enabled, and this identity can be utilized only by this specific playbook. This connection cannot be shared among playbooks. A managed identity also provides an option for the least privileged approach.

It's important to note that not all connectors in Logic Apps support managed identities. For example, a Microsoft Sentinel connector supports them, while Microsoft Teams and Office 365 Outlook do not.

To add permissions to a managed identity, you will need to go to the **Identity** tab in Logic Apps.

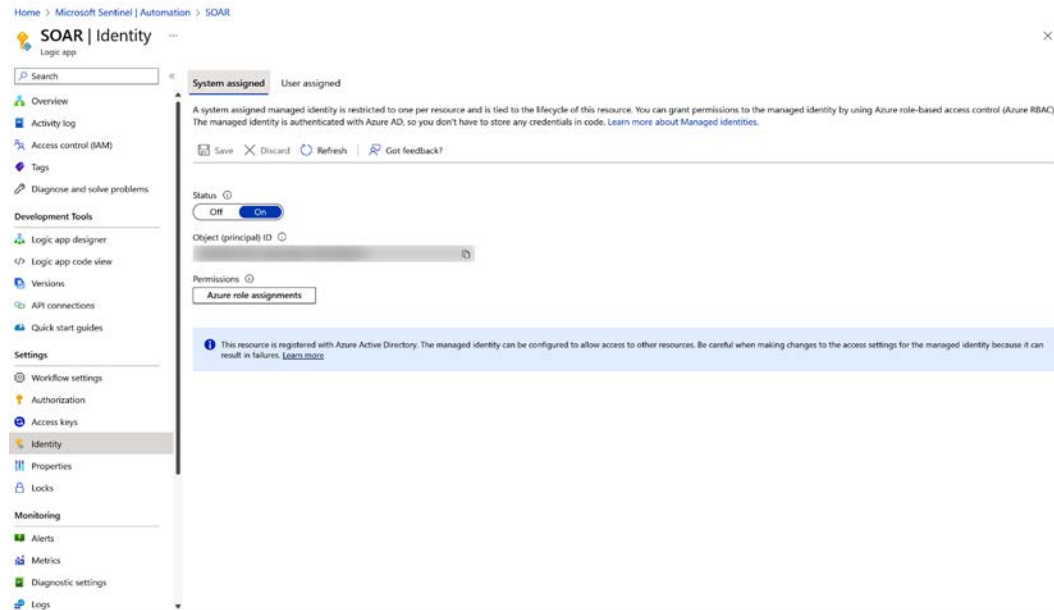


Figure 5.32 – Enabling a playbook’s managed identity

Once you are in the **Identity** section, you need to select **Azure role assignments** and then **Add role assignments** to assign an Azure permission to the managed identity.

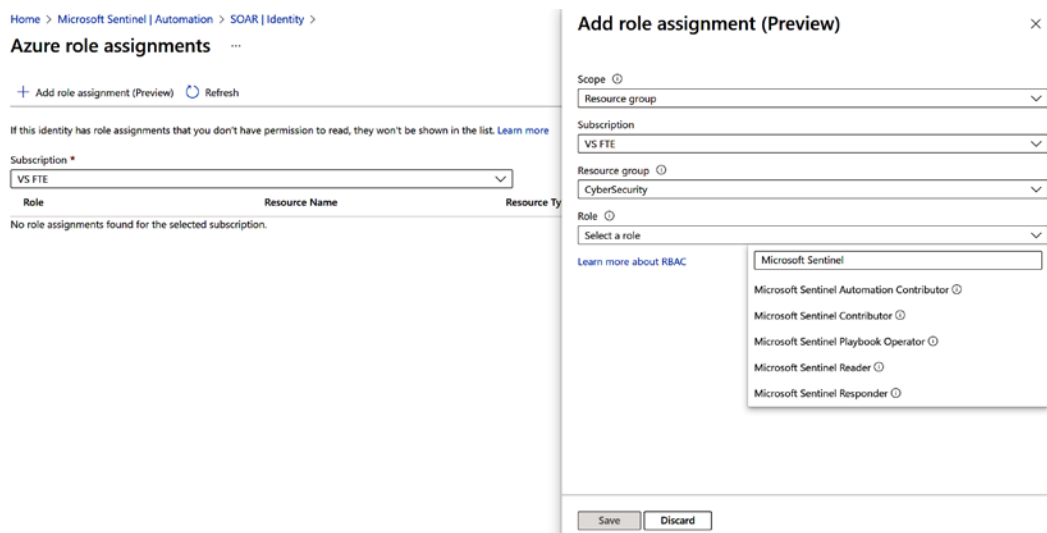
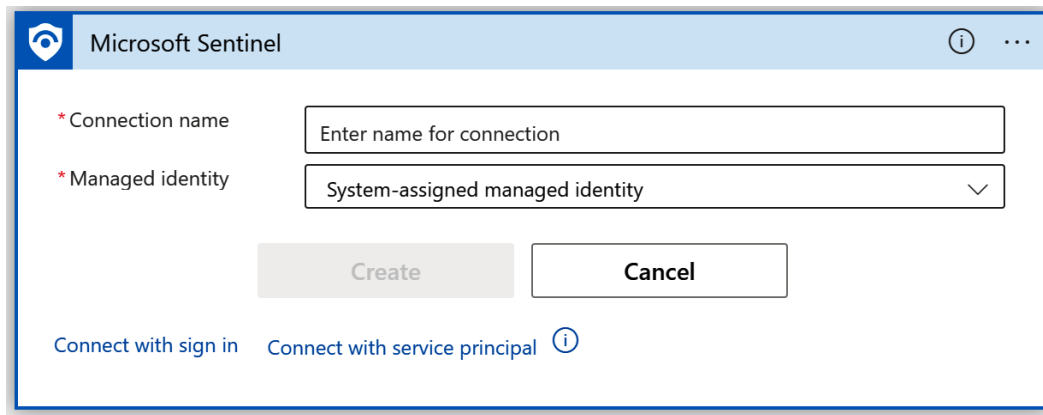


Figure 5.33 – Assigning a permission to a playbook’s managed identity

To authenticate a playbook trigger or action with a managed identity, first, fill in **Connection name**, then for connecting, select **System-assigned managed identity** from the drop-down list, and click **Create**.



The screenshot shows a 'Microsoft Sentinel' dialog box for creating a new connection. It features two required fields: '* Connection name' with a text input containing 'Enter name for connection', and '* Managed identity' with a dropdown menu currently showing 'System-assigned managed identity'. Below these fields are two buttons: 'Create' (disabled) and 'Cancel'. At the bottom, there are two links: 'Connect with sign in' and 'Connect with service principal', followed by an information icon.

Figure 5.34 – Connecting to a Logic Apps connector using a managed identity

Service principals

Service principals can be created on the Azure AD administrator page by registering an application. Once we have created an application, we will need to save a **Tenant ID** and **Application ID** in a secure space. These will be needed for authentication purposes, as well as to create a secret for the application and save it in a secure space, such as Azure Key Vault.

To register an application, we need to enter the application's name and specify whether the application is single- or multi-tenant.

[Dashboard](#) > [CybSec Guru | App registrations](#) >

Register an application ...

* Name

The user-facing display name for this application (this can be changed later).

SOAR ✓

Supported account types

Who can use this application or access this API?

- ☒ Accounts in this organizational directory only (CybSec Guru only - Single tenant)
- ☐ Accounts in any organizational directory (Any Azure AD directory - Multitenant)
- ☐ Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)
- ☐ Personal Microsoft accounts only

[Help me choose...](#)

Redirect URI (optional)

We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

Select a platform ▼

e.g. <https://example.com/auth>

Register an app you're working on here. Integrate gallery apps and other apps from outside your organization by adding from [Enterprise applications](#).

By proceeding, you agree to the [Microsoft Platform Policies](#) ⓘ

Register

Figure 5.35 – Creating a new service principal using Azure AD App registrations

Once the service principal has been created, we can assign API permissions to it, such as the Microsoft Graph Security API and Microsoft 365 Defender.

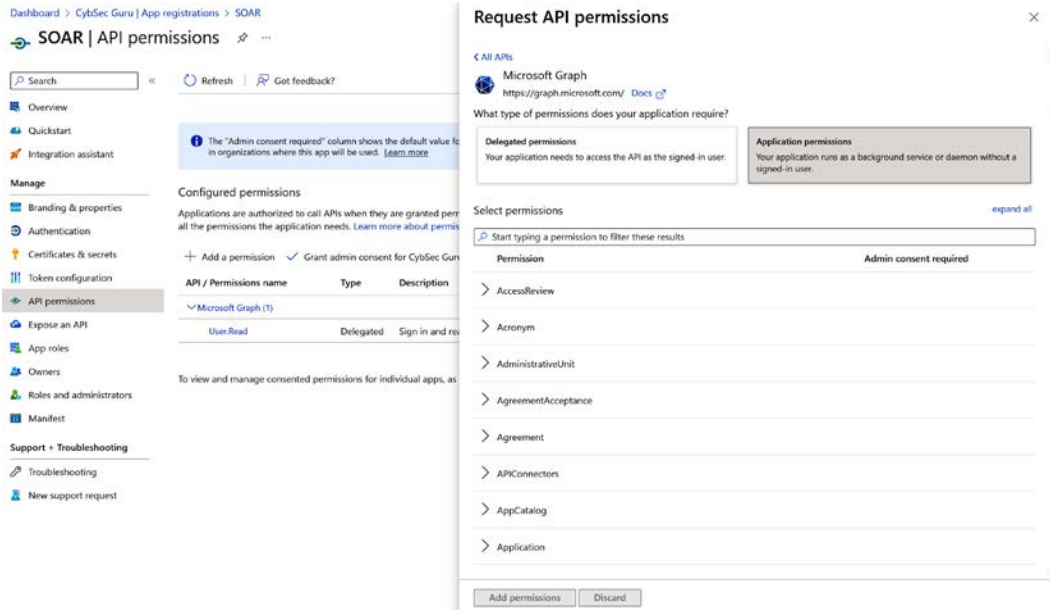


Figure 5.36 – Assigning API permissions to a service principal

We can also assign any Azure AD or Azure **Role Based Access Control (RBAC)** role to a service principal. For example, suppose we want the option to query data in a Log Analytics workspace. In that case, we can utilize the **Azure Monitor Logs** connector, which does not support a managed identity at the time of writing but does support service principals. Therefore, we will assign the Log Analytics Reader permission to the service principal and authenticate our connector with it. Another important note about using a service principal is that it can be reused across multiple playbooks as it is not playbook-specific. This is because it's a system-assigned managed identity.

The screenshot shows the 'Azure Monitor Logs' connection configuration dialog. It has a title bar with the Azure logo and the text 'Azure Monitor Logs'. Inside, there are four input fields: 'Connection name' (placeholder: 'Enter name for connection'), 'Client ID' (placeholder: 'Client (or Application) ID of the Azure Active Directory application.'), 'Client Secret' (placeholder: 'Client secret of the Azure Active Directory application.'), and 'Tenant' (placeholder: 'The tenant ID of the Azure Active Directory application.'). Below the fields are 'Create' and 'Cancel' buttons. At the bottom left, there is a link that says 'Connect with sign in'.

Figure 5.37 – Connecting to the Logic Apps connector using a service principal

User identity

The last option that we will cover is user identity. This option allows any user from your organization to authenticate the connection. The user carrying out the authentication must have the permission to perform the needed action (for example, to block a user in Azure Active Directory or to isolate a host in EDR) to authenticate the API connection. If the user doesn't have the permission, the playbook will fail on this step. This is also a shared connection. Once we create this API connection, it can be utilized across multiple playbooks – by a specific user or any user with permission to create and edit playbooks.

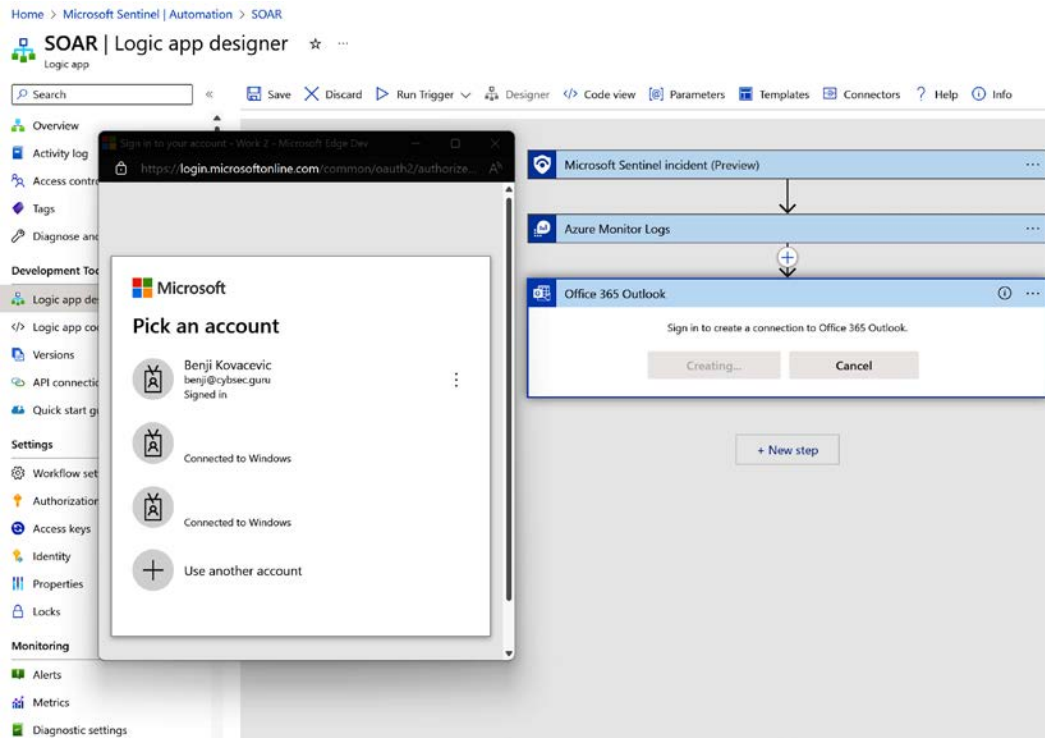


Figure 5.38 – Connecting to the Logic Apps connector using a user's identity

What's important to note here is that the user who authenticates this connection must have permission to perform the action (for example, to block a user in Azure Active Directory or isolate a host in EDR) when the playbook is running. Some organizations use **Privileged Identity Management (PIM)** and users are only assigned permission when needed. In this scenario, it is advisable to not use user identity for any action that can run while the user doesn't have permission active.

With that, let's shift focus and get to know triggers a little better.

Triggers

Triggers are used to define on what event a playbook will be triggered to run. It is always the first step when we create our playbook. Triggers also define the scheme that the playbook expects when it is triggered.

In Microsoft Sentinel, we have two primary triggers, with the third one still under development at the time of writing and, therefore, it will not be covered in this book.

The following triggers are available in Microsoft Sentinel:

- **Microsoft Sentinel alert:** This receives alert data as input
- **Microsoft Sentinel incident:** This receives incident data as input
- **Microsoft Sentinel entity:** Under development

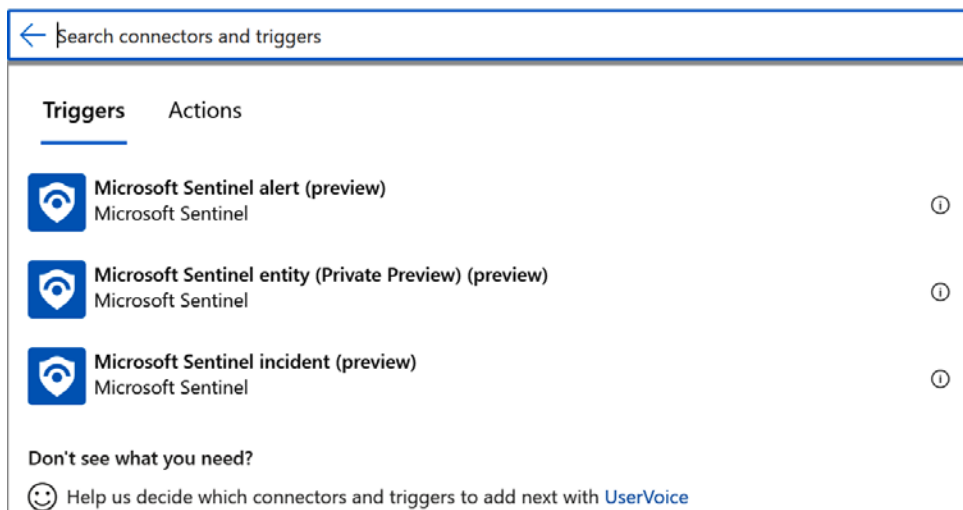


Figure 5.39 – Microsoft Sentinel playbook triggers

Since Microsoft Sentinel utilizes Logic Apps to create playbooks, we can utilize many more triggers that are not connected to Microsoft Sentinel. Some of them are as follows:

- **Schedule triggers:** Runs a playbook every n minutes, hours, or days, or at a specific time on specific days
- **Microsoft Forms triggers:** Creates a manual incident when a Microsoft Forms form is submitted
- **Office 365 Outlook:** Creates a manual incident when an email is received in a shared mailbox, and so on

Let's now focus on the rest of the playbook steps.

Actions

Once we have configured a trigger, we need to configure actions that will be performed when a playbook is triggered. We can utilize data that arrives with the trigger (incident data alongside the incident trigger, for example) to focus on specific information from it. These actions can be run sequentially, in parallel, or under complex conditions.

Microsoft Sentinel's native actions, among others, include the following:

- **Add comment to incident**
- **Bookmarks:** Create or update a bookmark
- **Create incident**
- **Entities:** Get accounts/hosts/IPs/URLs
- **Watchlist:** Create a new watchlist with data (raw content) and so on

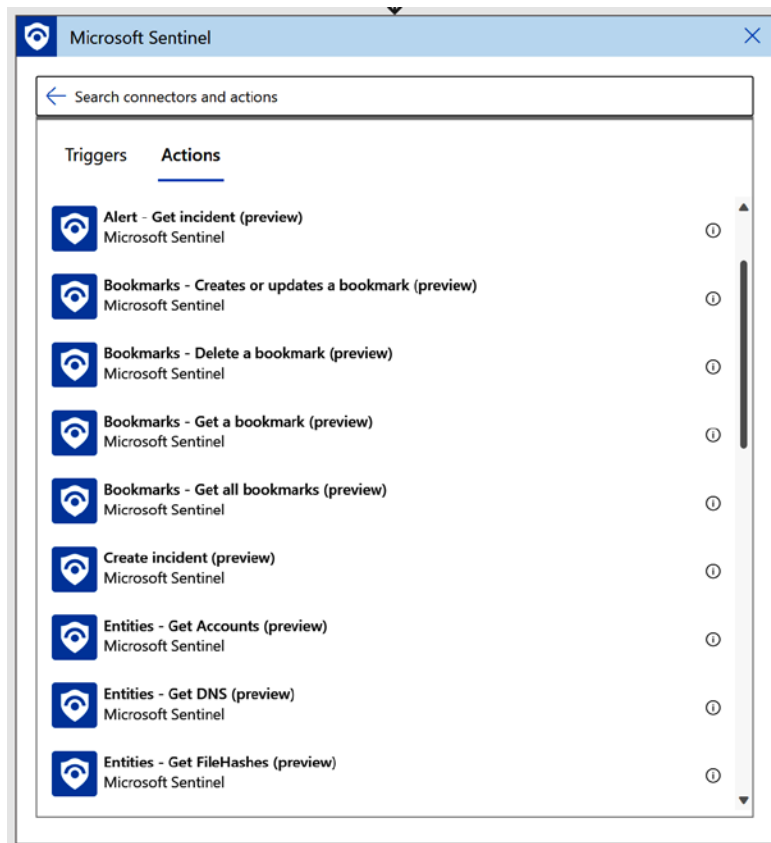


Figure 5.40 – Microsoft Sentinel playbook actions

It's important to know that each action defines its scheme, and we can utilize data from triggers and previous actions to define data in the action we are working on now. To access this data, we can utilize dynamic content. This will be covered in the next section.

Some other actions we can create are as follows:

- **Azure Active Directory:** An action to block a user, reset their password, and revoke the session
- **Azure Firewall:** An action to block an IP
- **Palo Alto/Fortinet/CheckPoint...:** An action to block an IP
- **ServiceNow/Jira:** Sends incident data and creates a record
- **VirusTotal:** Gets the IP/URL/file hash enrichment data

Some other actions native to Logic Apps that we need to be aware of are as follows:

- **Condition:** Here, we can define which block of actions to execute based on condition evaluation. For example, if the host starts with *admin*, we can auto-close the incident; if not, we can isolate the device.
- **For each:** If the data that we are working on is an array (set of data) and we want to act on each piece of data in the dataset.
- **Switch:** Similar to **Condition**, but we can have multiple paths.
- **HTTP:** To make an API call to a product or service if a native action is not available in Logic Apps.
- **Parse JSON:** To parse a result received by an HTTP call.
- **Create HTML table:** To be used when creating an email response and you want to create an HTML table containing data.
- **Set variable:** If we want to set a variable that can be used on multiple instances in a playbook.

We will cover these actions in more depth in *Chapter 9*.

To make playbook actions utilize dynamic data, we can utilize dynamic content in Logic Apps.

Dynamic content

Dynamic content refers to temporary fields in a playbook run; these are created by triggers and actions. The only rule is that using these temporary fields is only possible for triggers and actions that happened before the action we are working on occurred. For example, when a playbook is triggered with a Microsoft Sentinel incident trigger, the output that's received will contain all the necessary data about the incident, such as its severity, status, incident number, incident URL, entities, and alerts. Using dynamic content, we can specify these values in the next action. For example, we might want to get a list of IPs, so we can use the **Entities - Get IPs** action; as input, we can use the **Entities** dynamic content that we received from the Microsoft Sentinel incident trigger.

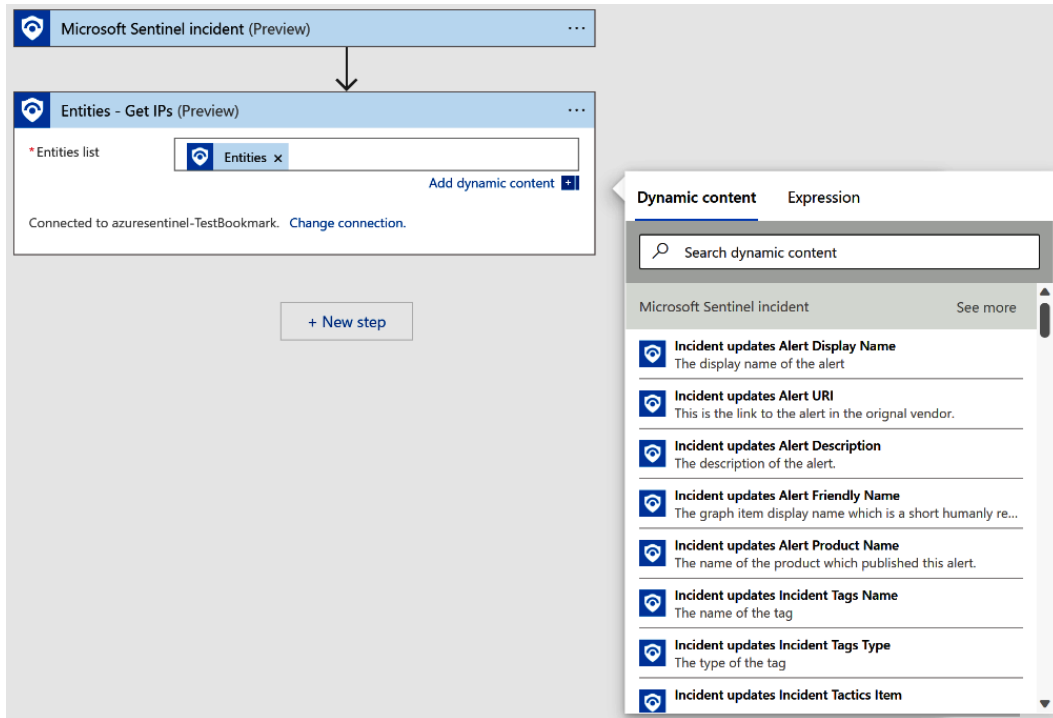


Figure 5.41 – Dynamic content in Microsoft Sentinel playbooks

If some data is not exposed in **Dynamic content** or we want to join two values, we can use expressions. An example of an expression is when we have an array containing alerts and we want to list alert names from that dataset; we can use a **join** expression to perform this action.

We will cover more on dynamic content and expressions in the hands-on examples in *Chapters 6 to 9*, which will feature common tips and tricks for working with Microsoft Sentinel playbooks.

Monitoring automation rules and playbook health

Microsoft Sentinel automation has a native way of monitoring the health of automation rules and playbook triggers. This monitoring can be enabled from the **Settings** page of Microsoft Sentinel, under **Health monitoring**.

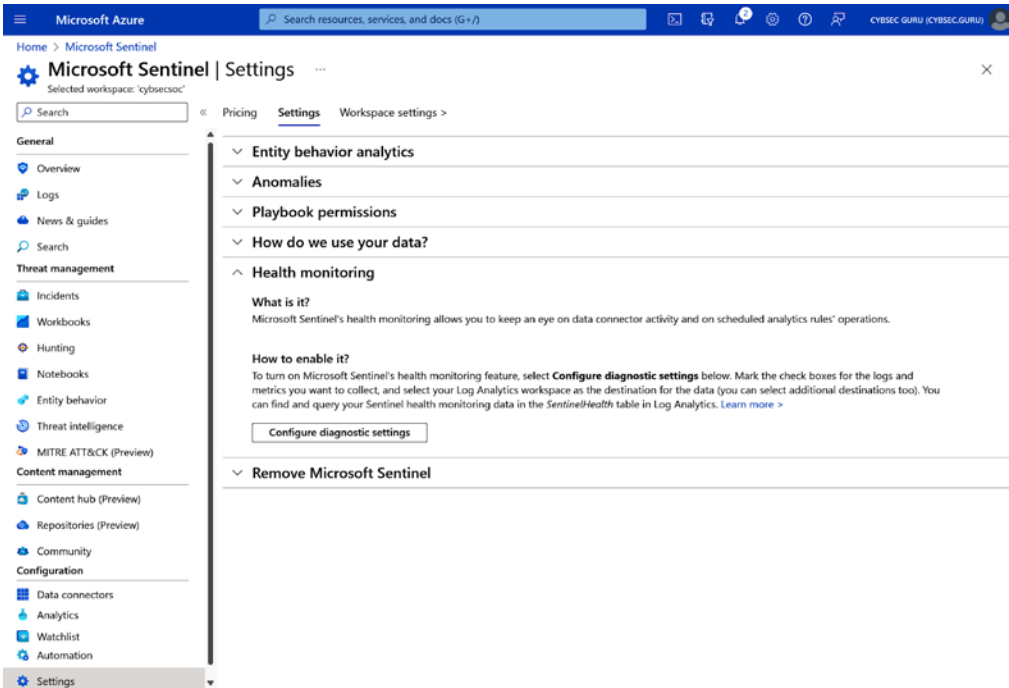


Figure 5.42 – Microsoft Sentinel – Health Monitoring configuration

We need to enable the **Automation** diagnostic settings and send them to the Log Analytics workspace where Microsoft Sentinel is enabled.

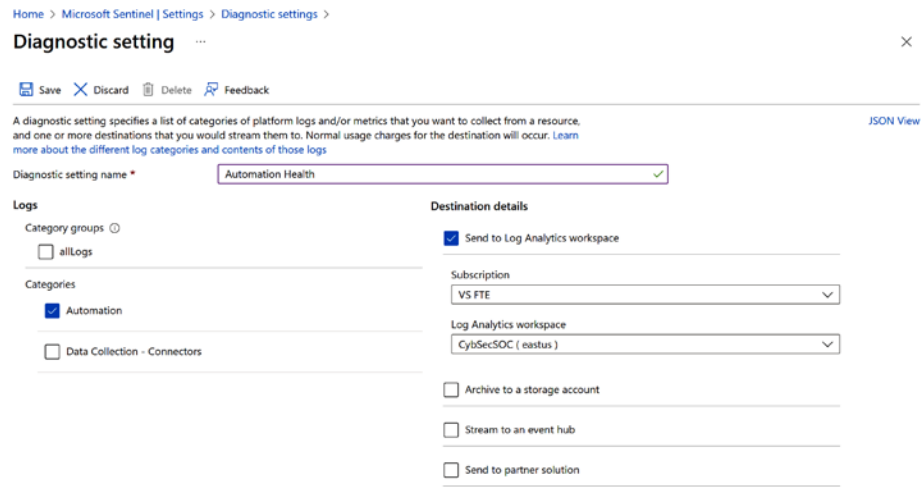


Figure 5.43 – The Automation Health monitoring configuration wizard

These diagnostic settings will be saved in the **SentinelHealth** table in Microsoft Sentinel so that we can query statuses using KQL.

A sample KQL query that you can use to get this data is as follows:

SentinelHealth

| where SentinelResourceType in ("Playbook", "Automation rule")

The screenshot displays the Microsoft Sentinel | Logs interface. On the left is a navigation pane with categories like General, Threat management, Content management, and Configuration. The main area shows a KQL query editor with the query: `SentinelHealth | where SentinelResourceType in ("Playbook", "Automation rule")`. Below the editor, the 'Results' tab is active, showing a table of log entries. The first entry is expanded, revealing detailed properties for an 'Automation rule run'.

TimeGenerated [UTC]	OperationName	SentinelResourceId	SentinelResourceName	Status
11/17/2022, 12:29:32.217 PM	Automation rule run	/subscriptions/...	Initial investigation	Success

Property	Value
TenantId	2022-11-17T12:29:32.2173122Z
TimeGenerated [UTC]	2022-11-17T12:29:32.2173122Z
OperationName	Automation rule run
SentinelResourceId	/subscriptions/...
SentinelResourceName	Initial investigation
Status	Success
Description	Rule executed successfully, triggering all actions.
Reason	Automation rule completed run.
WorkspaceId	/resourceGroups/.../providers/Microsoft.OperationalInsights/workspaces/...
SentinelResourceType	Automation rule
RecordId	2ba8236e-606d-4dc9-851f-26fa84689c09
ExtendedProperties	{ "TriggeredOn": "Incident", "TriggeredWhen": "Created", "ActionsTriggeredSuccessfully": 3, "TotalActions": 3, "TriggeredPlaybooks": ["Work..."] }
Type	SentinelHealth

Figure 5.44 – Querying the SentinelHealth table in the Microsoft Sentinel | Logs tab

We can also utilize the **Automation Health** workbook that’s available in workbook templates to get a detailed report about our automation health in Microsoft Sentinel.

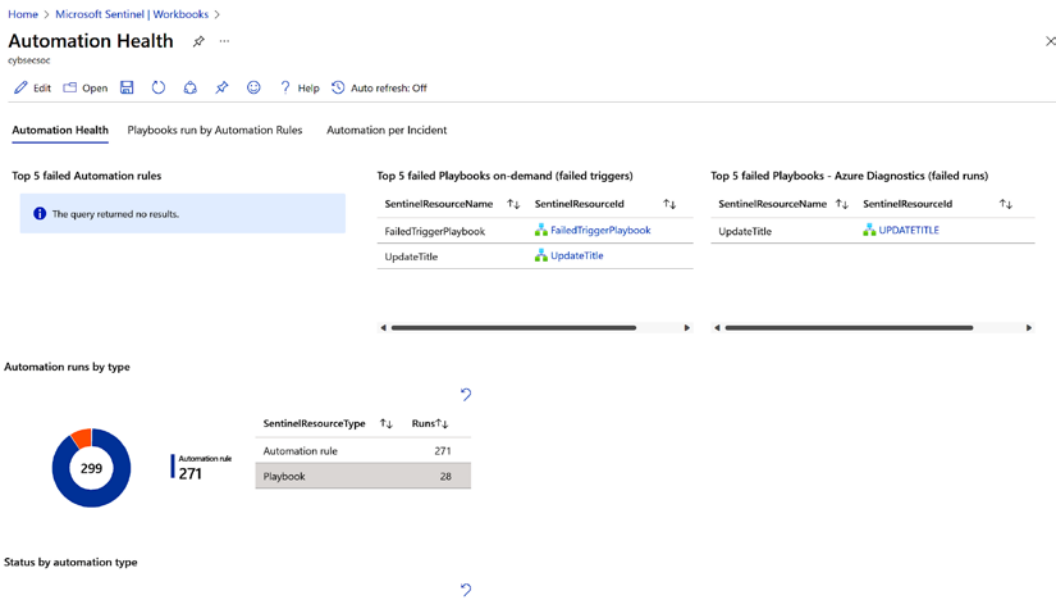


Figure 5.45 – Utilizing the Microsoft Sentinel Automation Health workbook

Automation Health only monitors a playbook trigger – in other words, it only checks whether the playbook triggered successfully, not the whole playbook run. To monitor whether the playbook runs successfully or not, we have to utilize diagnostic settings at the playbook level. A diagnostic setting can be configured when creating a playbook.

[Home](#) > [Microsoft Sentinel | Automation](#) >

Create playbook ...

1 Basics 2 Connections 3 Review and create

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * VS FTE 

Resource group * CyberSecurity 
[Create new](#)

Region * East US 

Playbook name *

☒ **Enable diagnostics logs in Log Analytics** 

Log Analytics workspace cybsecsoc 

☐ Associate with integration service environment 

Integration service environment 

Next : Connections >

Figure 5.46 – Enabling diagnostic settings when creating a new playbook

A diagnostic setting can also be created in the playbook itself.

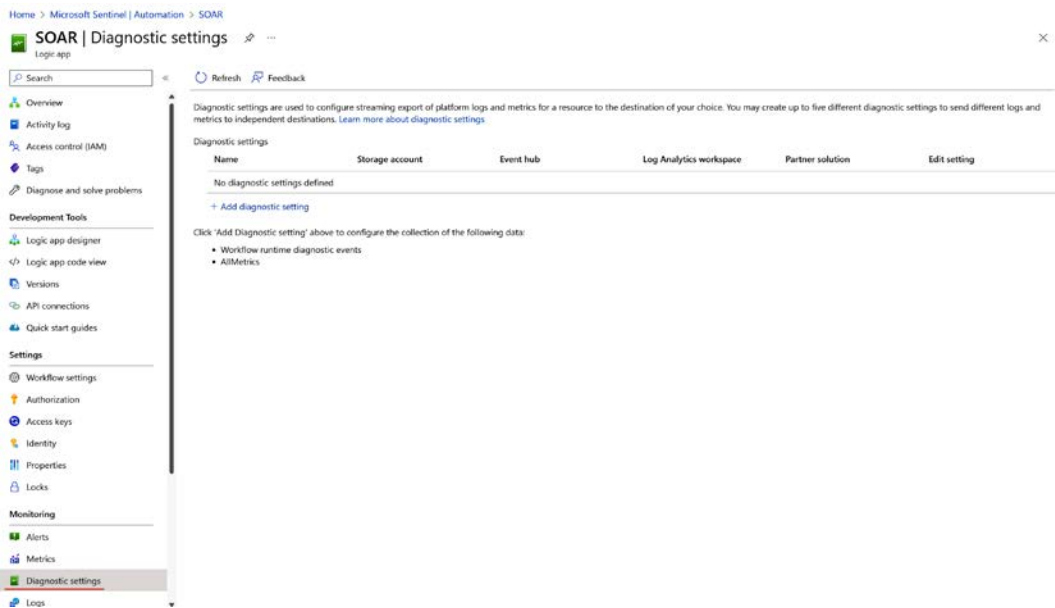


Figure 5.47 – Adding a diagnostic setting to an existing playbook

Once we’ve done this, we need to select the **Send to Log Analytics workspace** box, where we have Microsoft Sentinel enabled.

Home > Microsoft Sentinel | Automation > SOAR | Diagnostic settings >

Diagnostic setting

Save Discard Delete Feedback

A diagnostic setting specifies a list of categories of platform logs and/or metrics that you want to collect from a resource, and one or more destinations that you would stream them to. Normal usage charges for the destination will occur. [Learn more about the different log categories and contents of those logs](#)

Diagnostic setting name * SOAR playbook diagnostic settings ✓

Logs

Category groups ⓘ

☒ allLogs

Categories

☒ Workflow runtime diagnostic events

Metrics

☐ AllMetrics

Destination details

☒ Send to Log Analytics workspace

Subscription

VS FTE

Log Analytics workspace

CybSecSOC (eastus)

☐ Archive to a storage account

☐ Stream to an event hub

☐ Send to partner solution

Figure 5.48 – The Diagnostic setting configuration wizard of a playbook

This data will be saved in the **AzureDiagnostics** table. We can query it using KQL:

```
AzureDiagnostics
| where OperationName == "Microsoft.Logic/workflows/workflowRunCompleted"
```

The preceding KQL query will give you the following output:

The screenshot shows the Microsoft Sentinel | Logs interface. The query editor displays the following query:

```
1 AzureDiagnostics
2 | where OperationName == "Microsoft.Logic/workflows/workflowRunCompleted"
```

The results table shows the following data:

TimeGenerated [UTC]	ResourceId	Category	ResourceGroup	SubscriptionId
11/4/2022 2:59:43.740 PM	/SUBSCRIPTIONS/	WorkflowRuntime	CYBERSECURITY	803140b9-666a-
TenantId				
TimeGenerated [UTC]	2022-11-04T14:59:43.740564Z			
ResourceId	/SUBSCRIPTIONS/	RESOURCEGROUPS/	/PROVIDERS/MICROSO	
Category	WorkflowRuntime			
ResourceGroup	C	Y		
SubscriptionId				
ResourceProvider	MICROSOFT.LOGIC			
Resource	DBSL	US4		
ResourceType	WORKFLOWRUNS			
OperationName	Microsoft.Logic/workflows/workflowRunCompleted			
Level	Error			
status_s	Failed			
startTime_s [UTC]	2022-11-04T14:59:41.5032886Z			
endTime_s [UTC]	2022-11-04T14:59:43.7405762Z			
workflowId_s	/SUBSCRIPTIONS/			
resource_location_s	eastus			

Figure 5.49 – The queried playbook’s diagnostic settings in the Microsoft Sentinel | Logs tab

We can also join and compare data from the **AzureDiagnostics** and **SentinelHealth** tables to check whether the playbook triggered by the automation rule or the one we triggered manually had a successful run. This can be done by comparing the **runId** columns in both tables and joining them on the same **runId** since it is unique for each playbook run.

Here is a KQL example:

```
SentinelHealth
| where SentinelResourceType == "Automation rule"
| mv-expand TriggeredPlaybooks = ExtendedProperties.TriggeredPlaybooks
| extend runId = tostring(TriggeredPlaybooks.RunId)
| join (AzureDiagnostics
| where OperationName == "Microsoft.Logic/workflows/workflowRunCompleted"
| project
    resource_runId_s,
    playbookName = resource_workflowName_s,
    playbookRunStatus = status_s)
on $left.runId == $right.resource_runId_s
| project
    RecordId,
    TimeGenerated,
    AutomationRuleName= SentinelResourceName,
    AutomationRuleStatus = Status,
```

```
Description,  
workflowRunId = runId,  
playbookName,  
playbookRunStatus
```

The preceding KQL query will give you the following output:

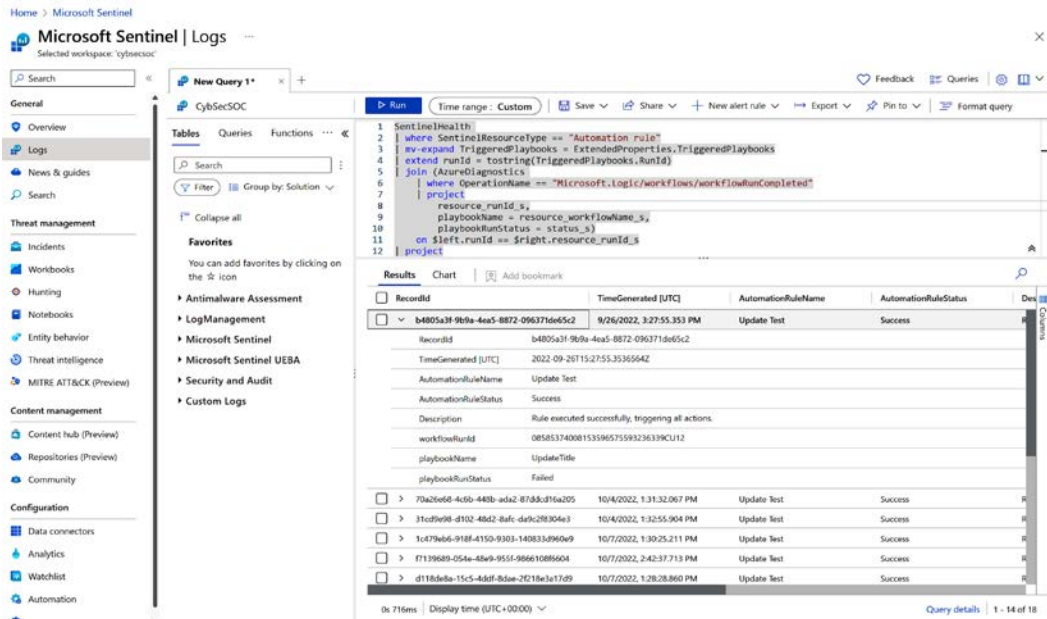


Figure 5.50 – Querying AzureDiagnostics and SentinelHealth in the Microsoft Sentinel | Logs tab

With automation rule and playbook health monitoring covered, let's wrap up this chapter.

Summary

In this chapter, we dug deep into Microsoft Sentinel automation and dissected each element. First, we focused on automation rules and their main elements – triggers, conditions, and actions – and how they define automation rule runs. We also covered permissions and ways to create automation rules. Then, we moved on to the topic of playbooks, where we focused on their main elements – triggers, actions, and dynamic content – as well as underlying information such as connectors, permissions, and authentication methods.

At the end of this chapter, we focused on the critical topic of automation health and how to monitor it using Microsoft Sentinel functionalities.

In the next chapter, we will begin our hands-on examples. We will focus on enriching incidents so that we can speed up MTTA and MTTR in Microsoft Sentinel.