The Elastic Container Project for Security Research

Using Docker to stand up the Elastic Stack

By **Andrew Pease, Colson Wilhoit, Derek Ditch**

30 August 2022

English

Preamble

The Elastic Stack is a modular data analysis ecosystem. While this allows for engineering flexibility, it can be cumbersome to stand up a development instance for testing. The easiest way to stand up the Elastic Stack, is to use **Elastic Cloud** - it's completely turnkey. However, there could be situations where Elastic Cloud won't work for your testing environment. To help with this, this blog will provide you with the necessary information required in order to quickly and painlessly stand up a local, fully containerized, TLS-secured, Elastic Stack with Fleet and the Detection Engine enabled. You will be able to create a Fleet policy, install an Elastic Agent on a local host or VM, and send the data into your stack for monitoring or analysis.

This blog will cover the following:

- The Elastic Stack
- The Elastic Container project
- How to use the Elastic Container project
- How to navigate Kibana and use its related features for security research

> **Note about the Elastic Container Project**
>
> The Elastic Container Project is not sponsored or maintained by the company, Elastic. Design and implementation considerations for the project may not reflect Elastic's guidance on deploying a production-ready stack.

The Elastic Stack

The Elastic Stack is made up of several different components, each of which provide a distinct capability that can be utilized across a wide variety of use cases.

Elasticsearch

Elasticsearch is a distributed, RESTful search and analytics engine. As the heart of the Elastic Stack, it centrally stores your data for lightning-fast search, fine-tuned relevancy, and powerful analytics that scale with ease.

Kibana

Kibana is the user interface that lets you visualize your Elasticsearch data and manage the Elastic Stack.

The Elastic Agent

The Elastic Agent is the modular agent that allows you to collect data from an endpoint or act as a vehicle to ship data from 3rd party sources, like threat feeds. The Elastic Security integration for endpoints prevents ransomware and malware, detects advanced threats, and arms responders with vital investigative context.

The Elastic Container Project

The Elastic Container Project

As mentioned above, the Elastic Stack is modular which makes it very flexible for a wide variety of use cases but this can add complexity to the implementation.

The Elastic Container project is an open source project that uses Docker Compose as a way to stand up a fully-functional Elastic Stack for use in non-production environments. This project is not sponsored or maintained by the Elastic company.

Introduction

The **Elastic Container Project** includes three main components:

- Elasticsearch
- Kibana
- the Elastic Agent

The project leverages **Docker Compose**, which is a tool to build, integrate, and manage multiple Docker containers.

To simplify the management of the containers, the project includes a shell script that allows for the staging, starting, stopping, and destroying of the containers.

Additionally, the project makes use of self-signed TLS certificates between Elasticsearch and Kibana, Kibana and your web browser, the Elastic Agent and Elasticsearch, and the Elastic Agent and Kibana.

Prerequisites

The project was built and tested on Linux and macOS operating systems. If you are using Windows, you'll not be able to use the included shell script, but you can still run native Docker Compose commands and manually perform post-deployment steps.

While not thoroughly tested, it is recommended that you contribute 4 cores and 8 GB of RAM to Docker.

There are only a few packages you need to install:

- Docker
- Docker Compose
- jq
- Git
- cURL

macOS

If you're running on macOS, you can install the prerequisites using **Homebrew**, which is an open-source package management system for macOS. Check out the Homebrew site for information on installing it if needed.

```
brew install jq git
brew install --cask docker
```

Linux

If you're running on Linux, you can install the prerequisites using your package management system (**DNF**, **Yum**, or **APT**).

RPM-based distributions

```
dnf install jq git curl
```

Ubuntu

```
apt-get install jq git curl
```

You'll also need the Docker suite (including the **docker-compose-plugin**). Check out Docker's **installation instructions** for your OS'

Cloning the project repository

The Elastic Container project is stored on Github. As long as you have Git installed, you can collect it from your CLI of choice.

```
git clone https://github.com/peasead/elastic-container.git
cd elastic-container
```

This repository includes everything needed to stand up the Elastic Stack containers using a single shell script.

Setting credentials

Before proceeding, ensure you update the credentials for the Elastic and Kibana accounts in the **.env** file located in the root directory of the repository from their defaults of **changeme**.

The shell script

As mentioned above, the project includes a shell script that will simplify the management of the containers.

```
usage: ./elastic-container.sh [-v] (stage|start|stop|restart|status|help)
actions:
   stage     downloads all necessary images to local storage
   start     creates network and starts containers
   stop      stops running containers without removing them
   destroy   stops and removes the containers, the network and volumes created
   restart   simply restarts all the stack containers
   status    check the status of the stack containers
   help      print this message
 flags:
   -v        enable verbose output
```

Stage

This option downloads all of the containers from the Elastic Docker hub. This is useful if you are going to be building the project on a system that does not always have Internet access. This is not required, you can skip this option and move directly to the start option, which will download the containers.

```
$ ./elastic-container.sh stage
8.3.0: Pulling from elasticsearch/elasticsearch
7aabcb84784a: Already exists
e3f44495617d: Downloading [====> ]  916.5kB/11.26MB
52008db3f842: Download complete
551b59c59fdc: Downloading [>     ]  527.4kB/366.9MB
25ee26aa662e: Download complete
7a85d02d9264: Download complete
…
```

Start

This opinion will create the container network, download all of the required containers, set up the TLS certificates, and start and connect Elasticsearch, Kibana, and the Fleet server containers together. This option is a “quick start” to get the Elastic Stack up and running. If you have not changed your credentials in the .env file from the defaults, the script will exit.

```
$ ./elastic-container.sh start

Starting Elastic Stack network and containers
[+] Running 7/8
 ⠿ Network elastic-container_default  Created  0.0s
 ⠿ Volume "elastic-container_certs"  Created  0.0s
 ⠿ Volume "elastic-container_esdata01"  Created 0.0s
 ⠿ Volume "elastic-container_kibanadata"  Created  0.0s
 ⠿ Container elasticsearch-security-setup  Waiting  2.0s
 ⠿ Container elasticsearch  Created  0.0s
…
```

Stop

This option will stop all running containers in the project, but will not remove them.

```
$ ./elastic-container.sh stop

Stopping running containers.
[+] Running 4/4
 ⠿ Container elastic-agent  Stopped  0.0s
 ⠿ Container kibana  Stopped  0.0s
 ⠿ Container elasticsearch  Stopped  0.0s
 ⠿ Container elasticsearch-security-setup  Stopped
…
```

Destroy

This option will stop all running containers in the project, remove the container network, remove all data volumes, and remove all containers.

```
$ ./elastic-container.sh destroy

#####
Stopping and removing the containers, network, and volumes created.
#####
[+] Running 8/4
 ⠿ Container elastic-agent  Removed  0.0s
 ⠿ Container kibana  Removed  0.0s
 ⠿ Container elasticsearch  Removed  0.0s
 ⠿ Container elasticsearch-security-setup  Removed  0.3s
 ⠿ Volume elastic-container_esdata01  Removed  0.0s
 ⠿ Network elastic-container_default  Removed  0.1s
…
```

Restart

This option restarts all of the project containers.

```
$ ./elastic-container.sh restart

#####
Restarting all Elastic Stack components.
#####
Name  Command  State  Ports
--------------------------
elasticsearch  /bin/tini -- /usr/local/bi ...  Up (healthy)  0.0.0.0:9200->9200/tcp, 9300/tcp
fleet-server  /usr/bin/tini -- /usr/loca ...  Up  0.0.0.0:8220->8220/tcp
kibana  /bin/tini -- /usr/local/bi ...  Up (healthy)  0.0.0.0:5601->5601/tcp
```

Status

This option returns the status of the project containers.

```
$ ./elastic-container.sh status
Name  Command  State  Ports
--------------------------
elasticsearch  /bin/tini -- /usr/local/bi ...  Up (healthy)  0.0.0.0:9200->9200/tcp, 9300/tcp
fleet-server  /usr/bin/tini -- /usr/loca ...  Up  0.0.0.0:8220->8220/tcp
kibana  /bin/tini -- /usr/local/bi ...  Up (healthy)  0.0.0.0:5601->5601/tcp
```

Clear

This option clears all documents in the logs and metrics indices.

```
$ ./elastic-container.sh clear

Successfully cleared logs data stream
Successfully cleared metrics data stream
```

Help

This option provides instructions on using the shell script.

```
$ ./elastic-container.sh help

usage: ./elastic-container.sh [-v] (stage|start|stop|restart|status|help)
actions:
   stage downloads all necessary images to local storage
   start creates a container network and starts containers
   stop stops running containers without removing them
   destroy stops and removes the containers, the network and volumes created
   restart simply restarts all the stack containers
   status check the status of the stack containers
   clear all documents in logs and metrics indexes
   help print this message
flags:
   -v enable verbose output
```

Getting Started

Now that we've walked through the project overview and the shell script, let's go through the process of standing up your own stack.

Updating variables

All of the variables are controlled in an environment file (**.env**) that is at the root of the repository. The only things that you must change are the default usernames and passwords for **elastic** and **kibana**.

Open the **.env** file with whatever text editor you're most comfortable with and update the **ELASTIC_PASSWORD** and **KIBANA_PASSWORD** variables from **changeme** to something secure. If you do not update the credentials from the defaults in the **.env** file, the script will exit.

If you want to change the other variables (such as the stack version), you can do so in this file.

Starting the Elastic Stack

Starting the project containers is as simple as running the **elastic-container.sh** shell script with the start option.

```
$ ./elastic-container.sh start

Starting Elastic Stack network and containers
[+] Running 7/8
⠿ Network elastic-container_default Created 0.0s
⠿ Volume "elastic-container_certs" Created 0.0s
⠿ Volume "elastic-container_esdata01" Created 0.0s
⠿ Volume "elastic-container_kibanadata" Created 0.0s
⠿ Container elasticsearch-security-setup Waiting 2.0s
⠿ Container elasticsearch Created 0.0s
⠿ Container kibana Created 0.1s
⠿ Container fleet-server Created 0.2s

Attempting to enable the Detection Engine and Prebuilt-Detection Rules
Kibana is up. Proceeding
Detection engine enabled. Installing prepackaged rules.
Prepackaged rules installed!
Waiting 40 seconds for Fleet Server setup
Populating Fleet Settings
READY SET GO!
Browse to https://localhost:5601
Username: elastic
Passphrase: you-changed-me-from-the-default-right?
```

Accessing the Elastic Stack

Once the containers have all downloaded and started, you'll get an output that tells you to browse to **https://localhost:5601**.

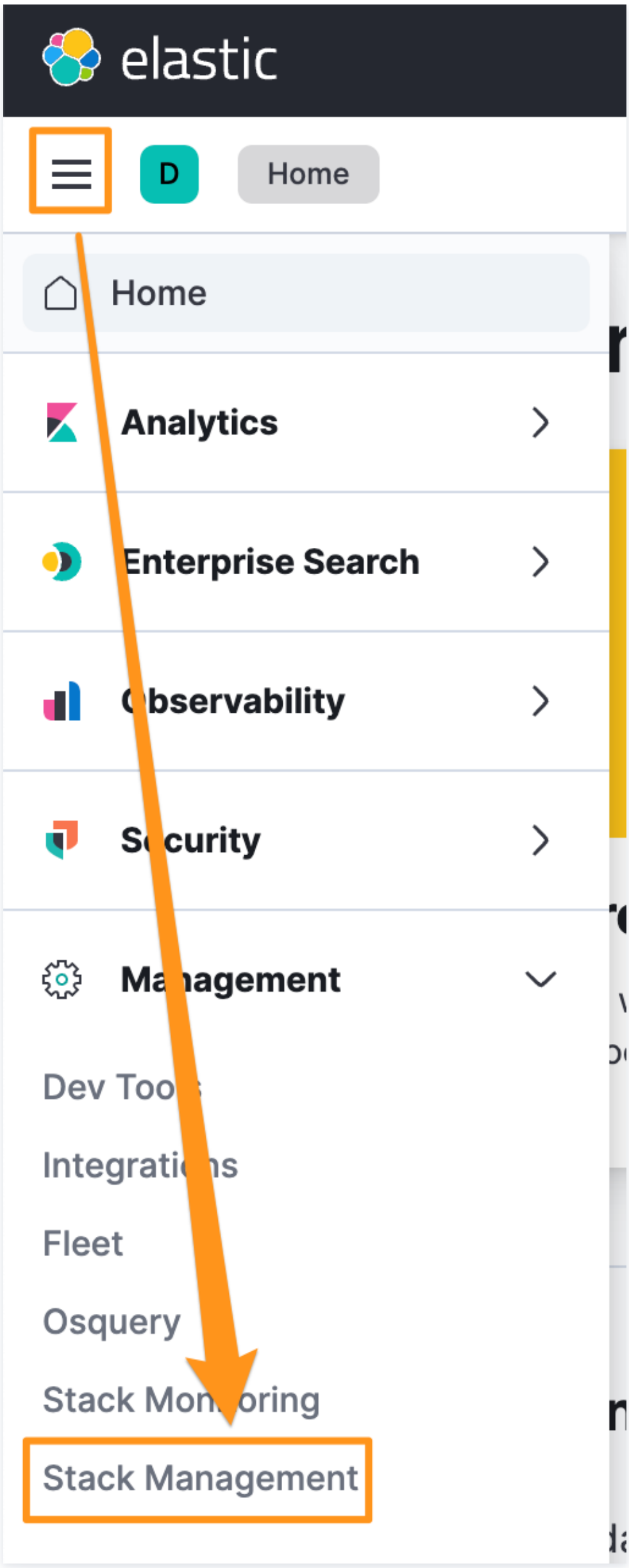
Note: You'll need to accept the self-signed TLS certificate.

Enabling the Platinum Features

Enabling the Platinum license features are completely optional. Security features, like anti-malware, EDR, EPP, etc. are included in the Basic license. Memory, behavior, and ransomware protections are Platinum license features. If you want to change your license, we can do that with the **.env** file or from within Kibana. You can update to Elastic Platinum for 30-days.

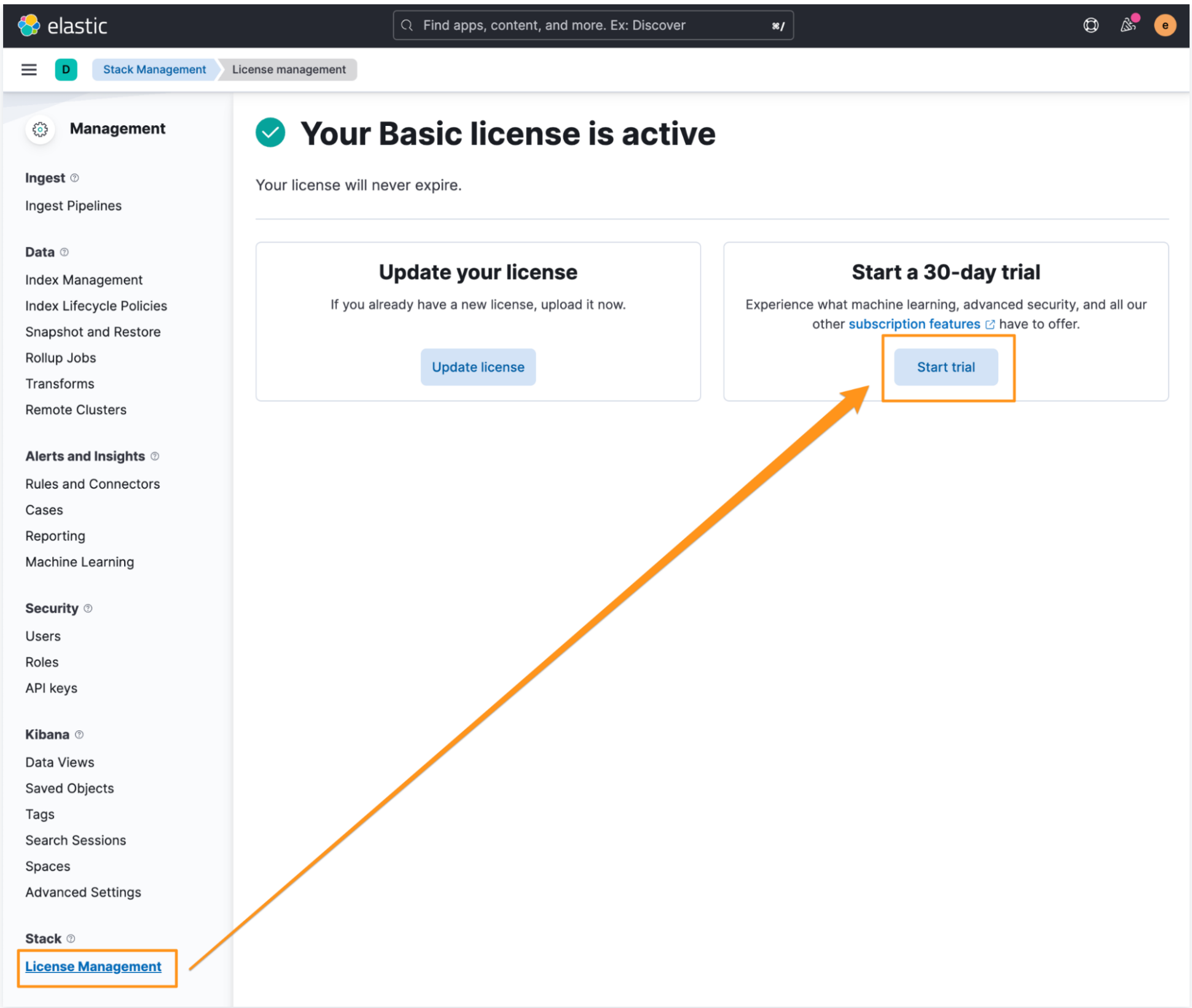
If you want to use the **.env** file so that the features are enabled when the stack is built, change **LICENSE=basic** to **LICENSE=trial** and then start the project as normal.

If you prefer to use Kibana, click on the hamburger menu, and then click on Stack Management.



Access Stack Management from Kibana

Click on License Management and then “Start a 30-day trial”.



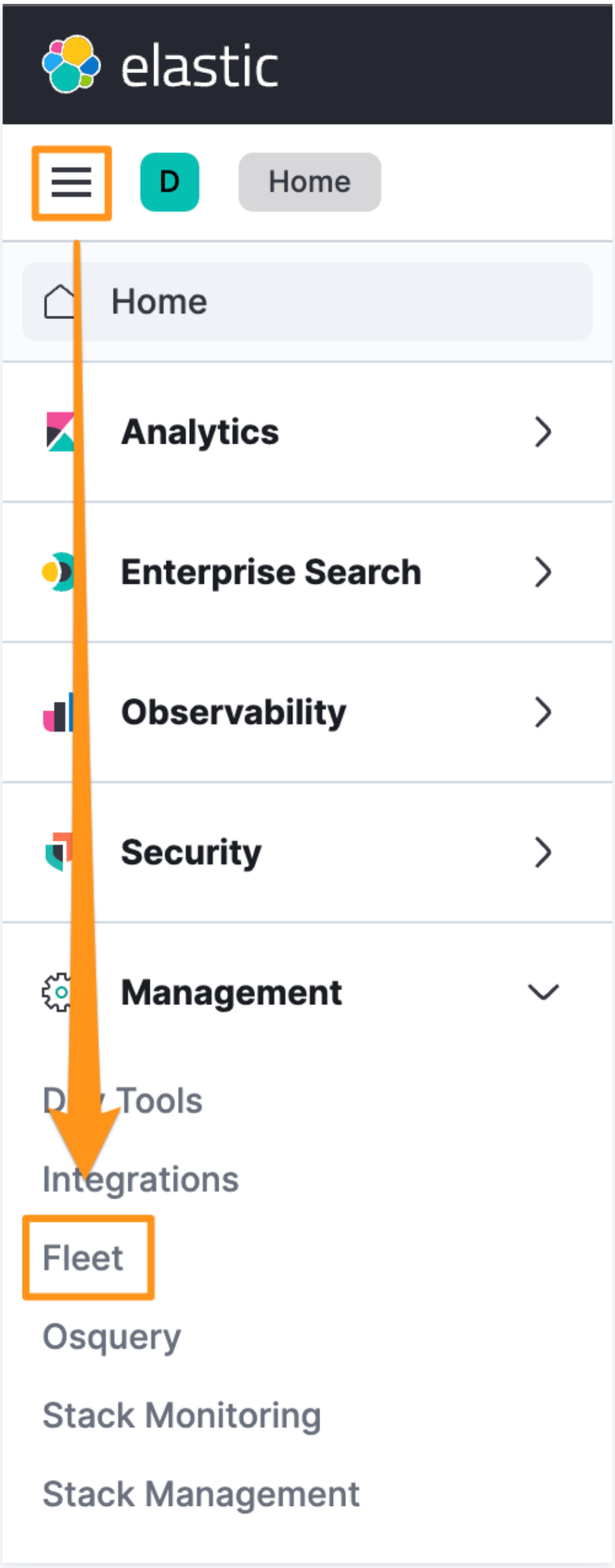
Start a 30-day trial

Creating a Fleet policy

Now that we have the entire Elastic Stack up and running, we can make a **Fleet** policy. Fleet is a subroutine of an **Elastic Agent** (which was built when we ran the **start** option in the shell script) that enables you to manage other Elastic Agents, policies, and integrations.

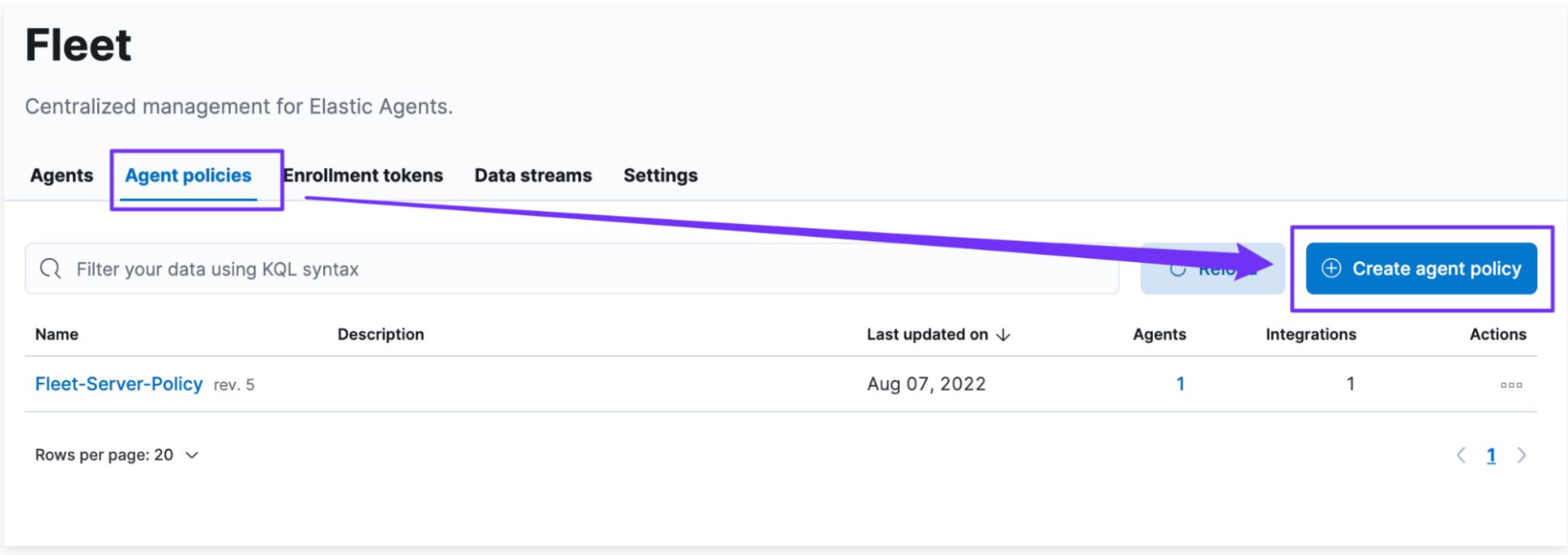
> **Training opportunities**
>
> Fleet is managed in Kibana, the UI that allows you to interact with data stored in Elasticsearch and manage your Elastic stack. If you're interested in learning more about Kibana, check out the **free training videos**.

Log into your Kibana instance and click on the "hamburger" menu on the top left, and navigate down to "Fleet", under the "Management" section.



Accessing Fleet

Next, click on the “Agent policies” tab and then the “Create agent policy” button.



Create agent policy

Give your new policy a name and a description (optional). Normally, we uncheck the “Collect agent logs” and “Collect agent metrics” options because it’s additional data going to the stack that we generally don’t need for our specific use-case. If you’re doing troubleshooting or interested in what’s happening behind the scenes, this data can help you understand that.

Create agent policy

Agent policies are used to manage settings across a group of agents. You can add integrations to your agent policy to specify what data your agents collect. When you edit an agent policy, you can use Fleet to deploy updates to a specified group of agents.

Name

Windows

Collect system logs and metrics

Advanced options

Description

Add a description of how this policy will be used.

Elastic Container Project Policy - Windows

Default namespace

Namespaces are a user-configurable arbitrary grouping that makes it easier to search for data and manage user permissions. A policy namespace is used to name its integration's data streams. Learn more.

default

Agent monitoring

Collecting monitoring logs and metrics will also create an Elastic Agent integration. Monitoring data will be written to the default namespace specified above.

Collect agent logs

Collect agent metrics

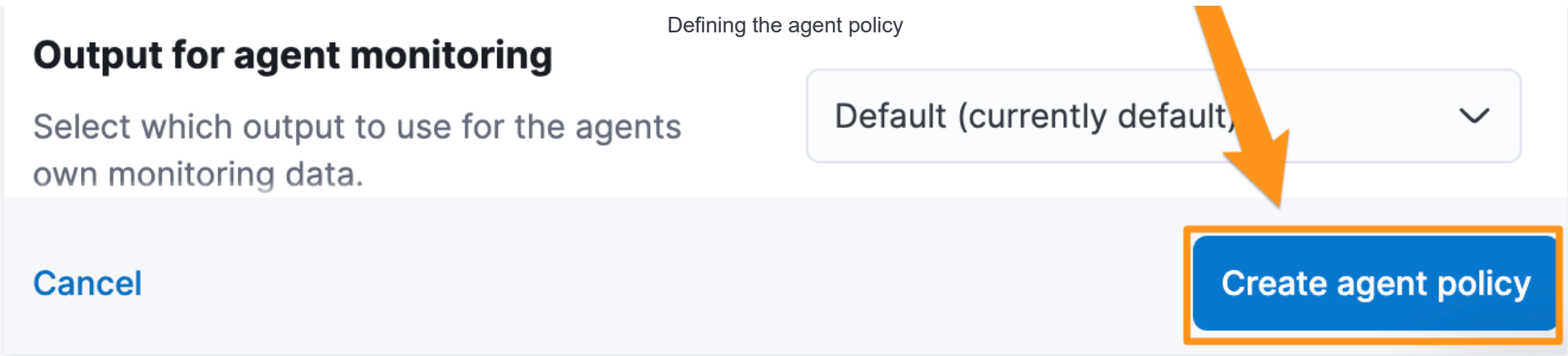
Unenrollment timeout

An optional timeout in seconds. If provided, an agent will automatically unenroll after being gone for this period of time.

Output for integrations

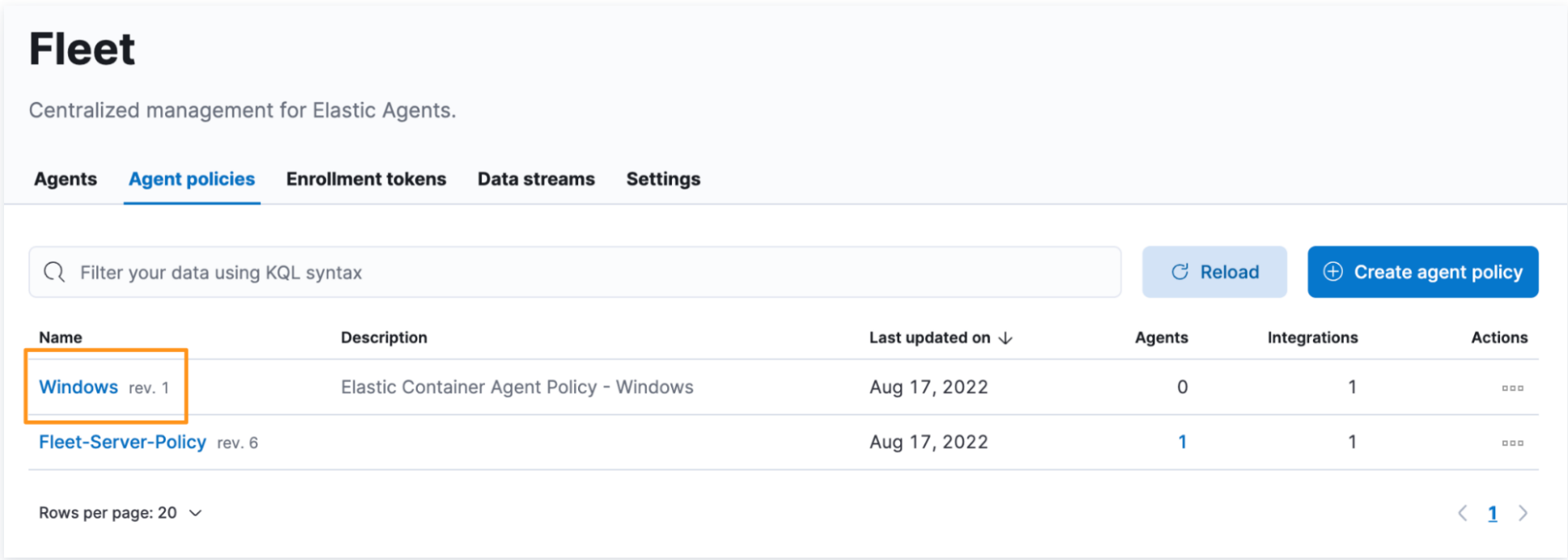
Select which output to use for data from integrations.

Default (currently default)

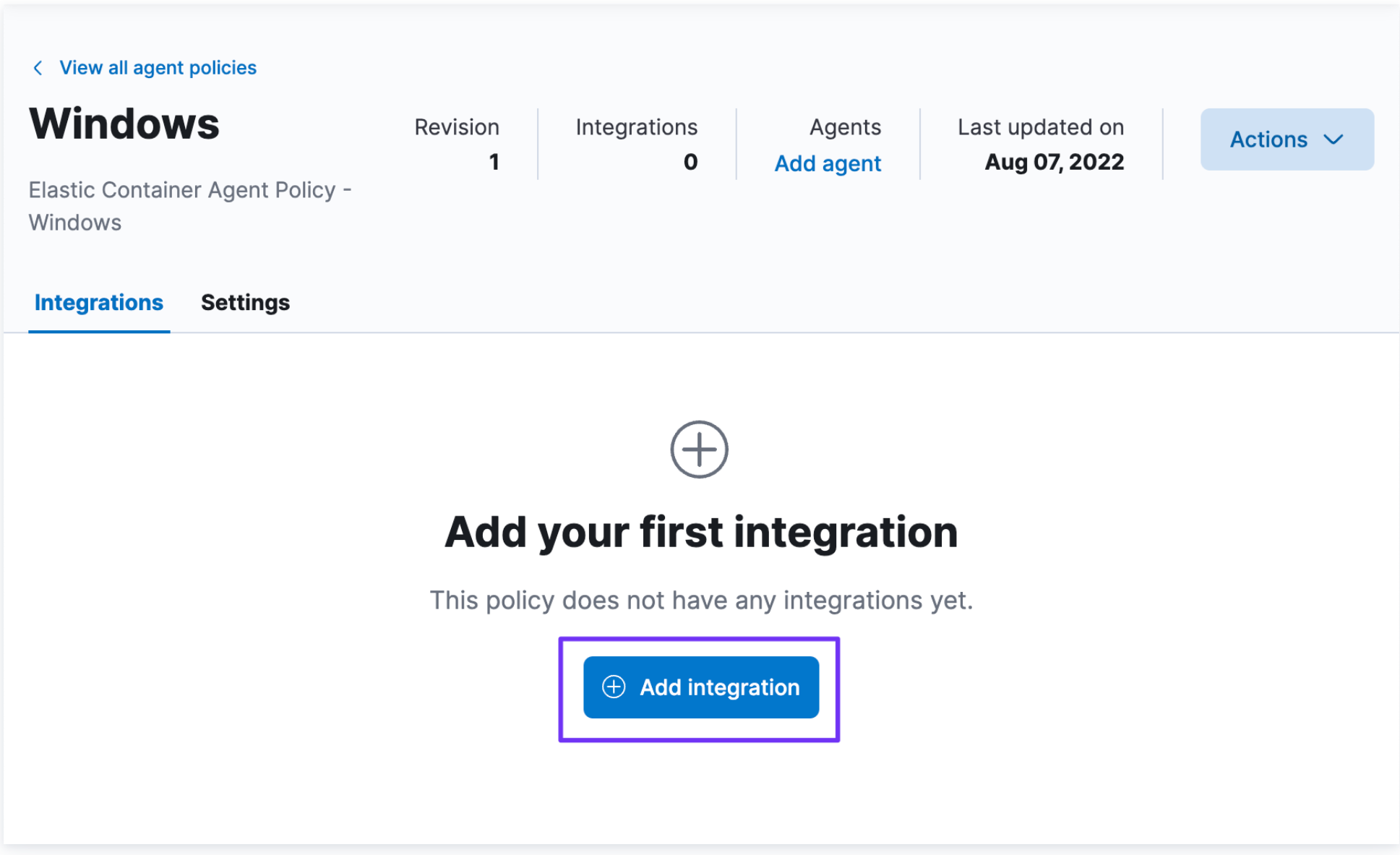


Defining the agent policy

Next, click on your new policy and the blue “Add integration” button.



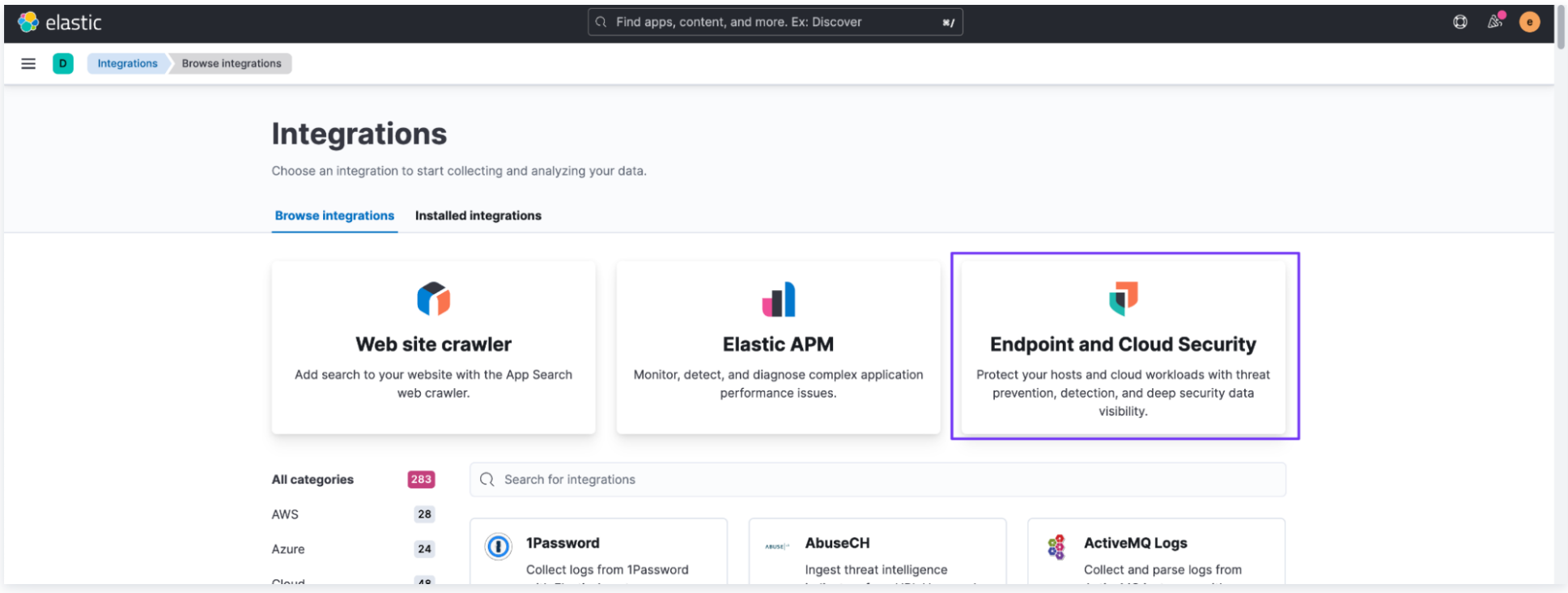
Open the Fleet policy



Add integrations

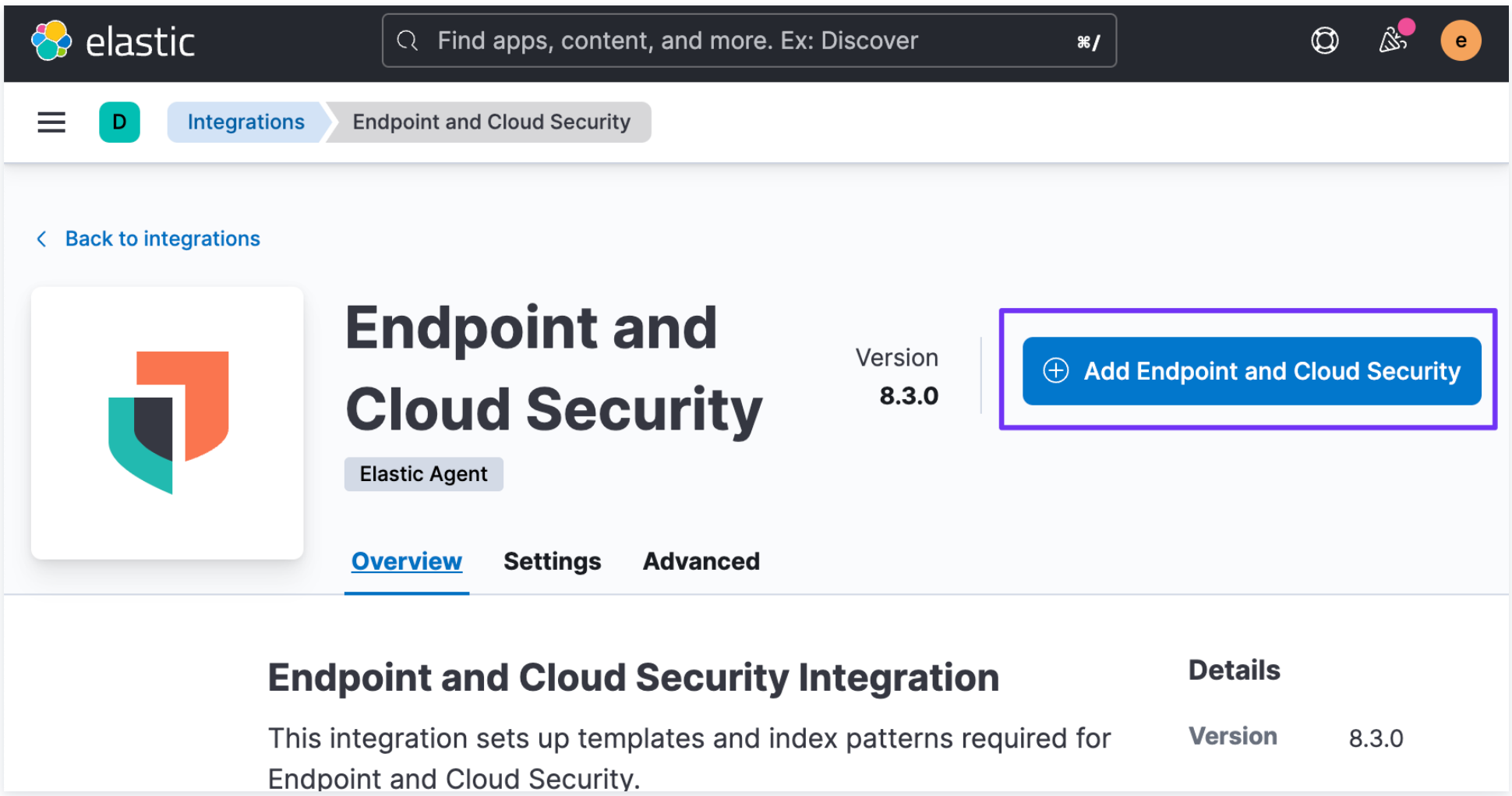
There are hundreds of integrations, but the ones that we’re most interested in for this blog are for Elastic Security.

To install Elastic Security, simply click on the tile on the main integrations page or search for “security”.



Endpoint and Cloud Security integration

Next, click the “Add Endpoint and Cloud Security” button to install this integration into the policy we just created.



Add Endpoint and Cloud Security

Name the integration and click the blue “Save and continue” button.

elastic Find apps, content, and more. Ex: Discover ⌘/

D Integrations Endpoint and Cloud ... Add integration Send feedback

Cancel

Add Endpoint and Cloud Security integration

Agent policy
Windows

Configure an integration for the selected agent policy.

1 Configure integration

Integration settings

Choose a name and description to help identify how this integration will be used.

Integration name
security

Description Optional

Advanced options

We'll save your integration with our recommended defaults. You can change this later by editing the Endpoint and Cloud Security integration within your agent policy.

2 Where to add this integration?

New hosts Existing hosts

Agent policy

Agent policies are used to manage a group of integrations across a set of agents.

Agent policy
Windows

0 agents are enrolled with the selected agent policy.

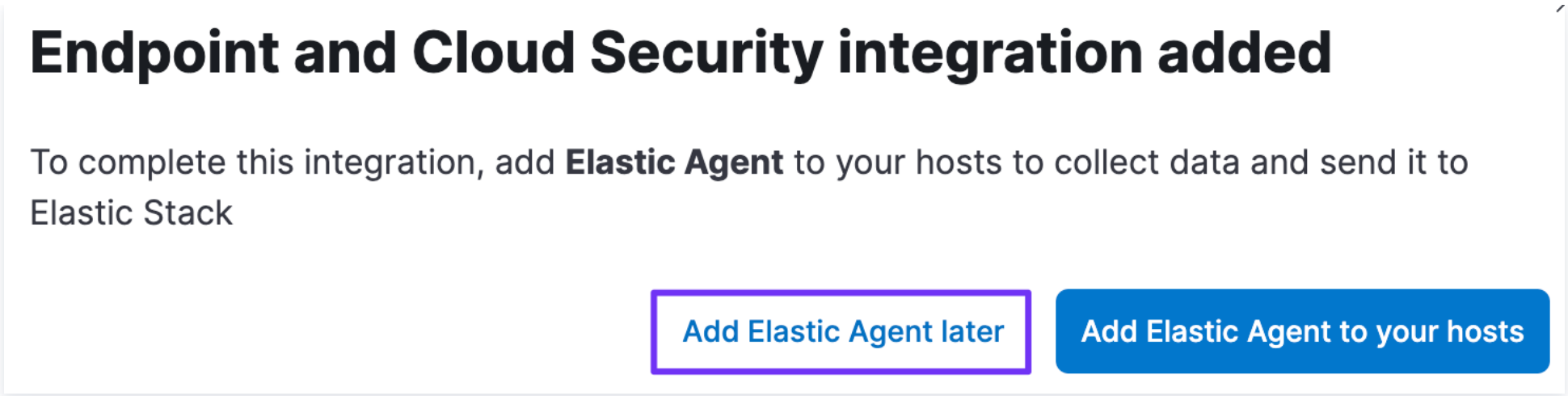
Cancel Save and continue

Save the integration to the policy

Are you using Sysmon?

While the Endpoint and Cloud Security and System integrations will collect security related logs, if you're using Sysmon on a Windows host, you may want to add the "Windows" integration to collect those logs.

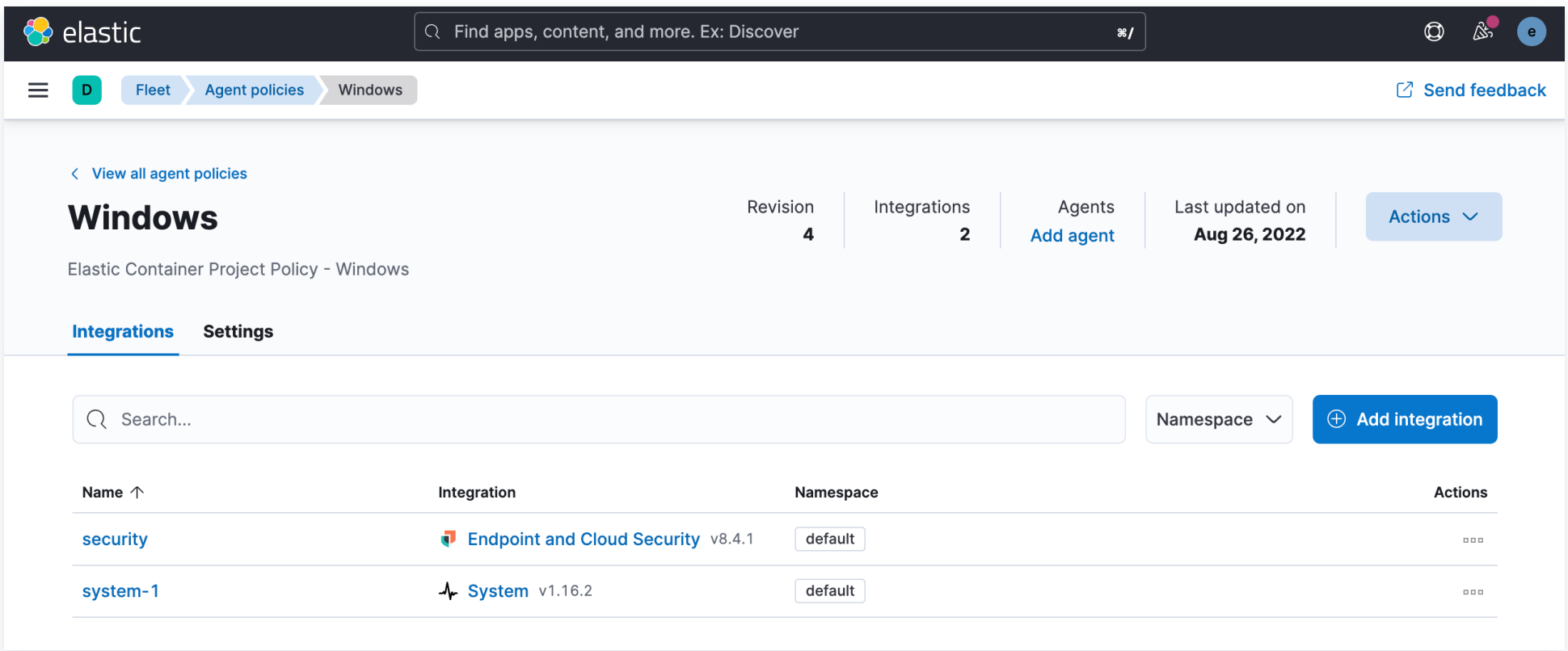
Once the integration is installed, you'll be prompted to add more Agents or to do that later. Select the "Add Elastic Agent later" option so we can make a few more changes to our policy.



Add Elastic Agents later

Now we'll be dropped back to our policy page.

We should have two integrations for our policy: **security** and **system-1**.

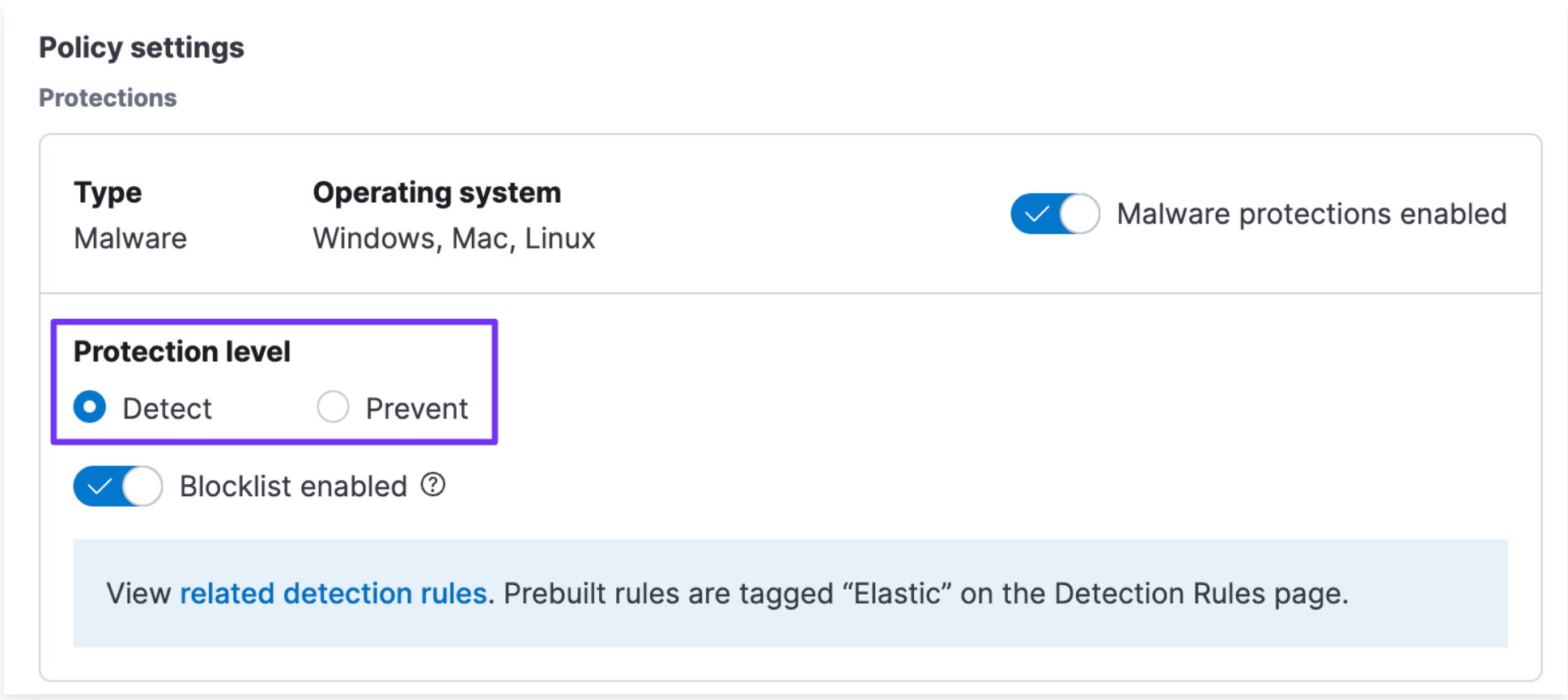


Reviewing the Windows policy

Before we add any agents, we'll want to set our Elastic Agent to Detect (so that it allows the malware to completely execute), register the Elastic Agent as a trusted AV solution (Windows only), and instruct the Endpoint and Cloud Security integration to collect memory samples from security events. This is tremendously helpful for "fileless" malware that injects directly into memory, like Cobalt Strike.

> **Additional malware beacon information**
>
> If you want to learn more about extracting malware beacons from events generated by the Elastic Agent, check out our other **publications** and **repositories**.

To allow the malware to continue to execute, on your "Windows" policy page, click on the name of the integration ("security" in our example), set the Protection level to "Detect".

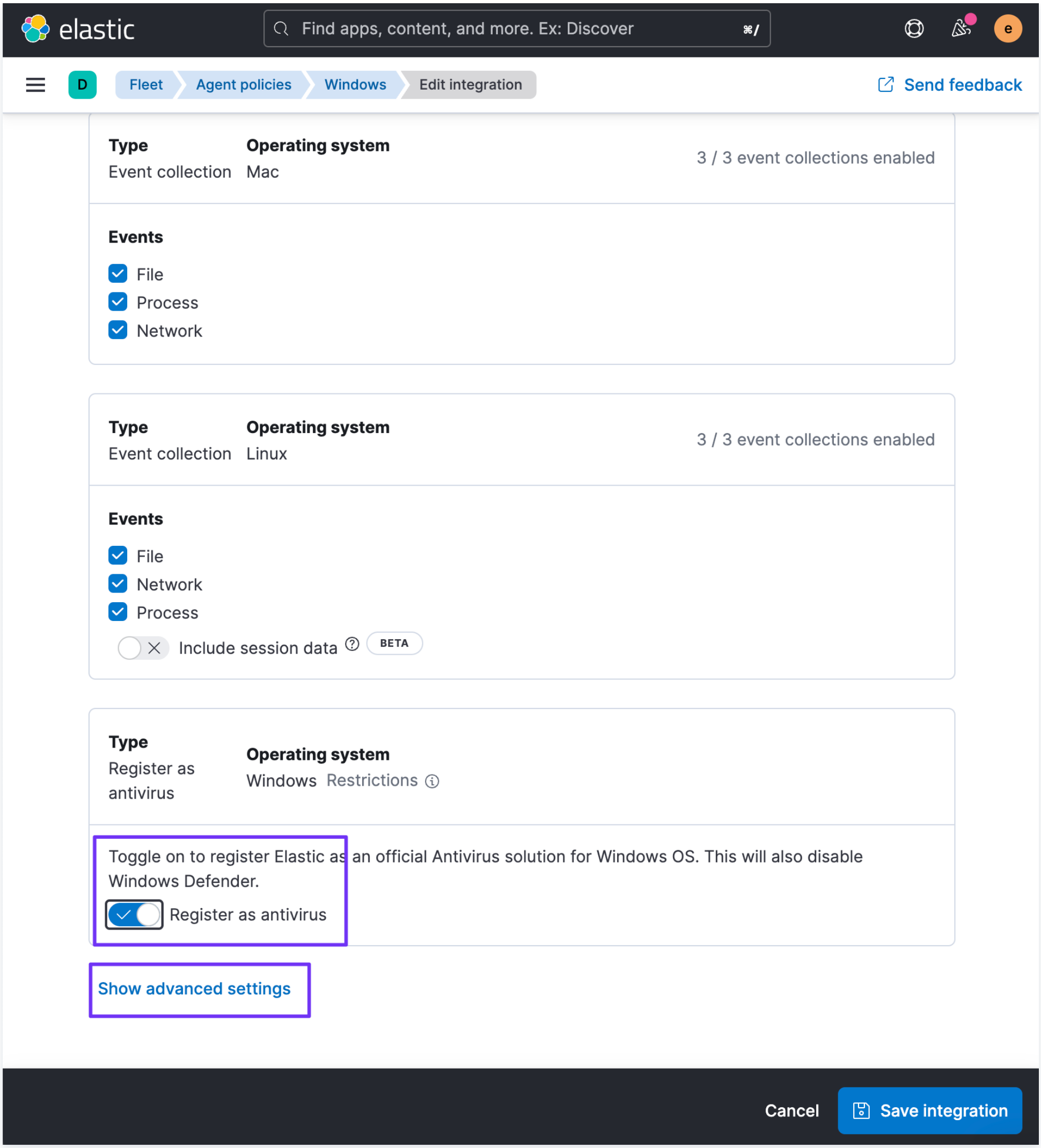


Setting the Protection level to Detect

Repeat these steps for the Ransomware, Memory threat protections, and Malicious behavior sections.

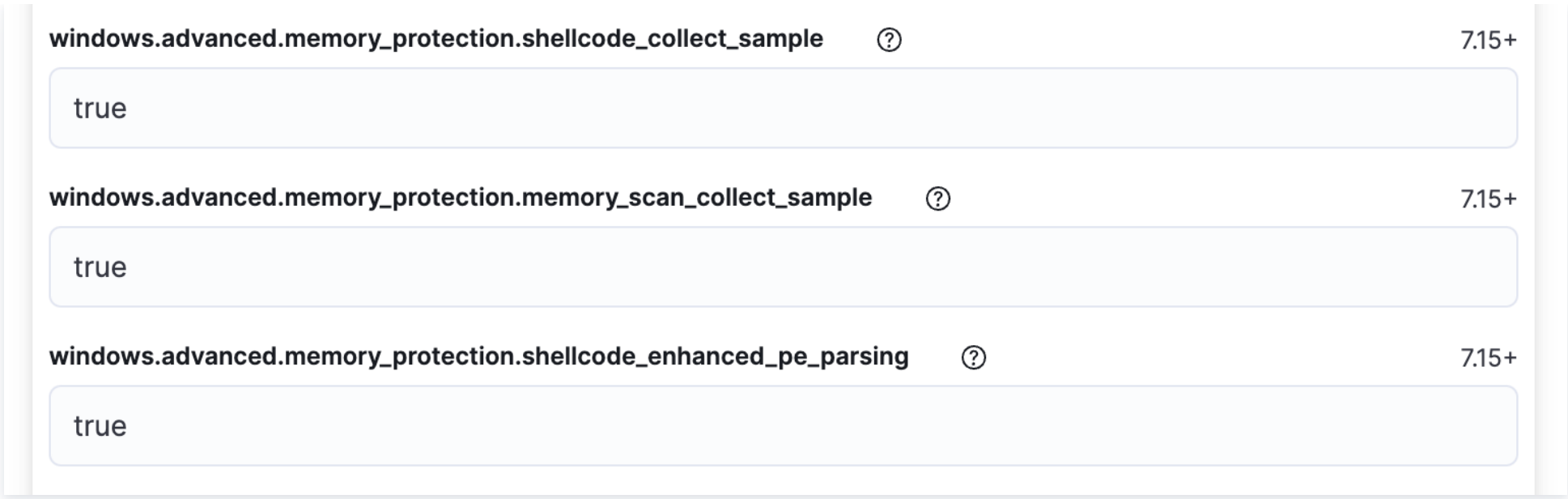
> **Detect vs. Prevent**
>
> We’re setting the Elastic Agent to Detect so that the malware we’re detonating will run completely so that we can analyze the entire execution chain. If you want the malware to be stopped, you can leave this in Prevent mode.

Next, scroll to the bottom and select the “Register as antivirus” toggle and click on the “Show advanced settings” hyperlink.



Register as antivirus

Scroll down to **windows.advanced.memory_protection.shellcode_collect_sample**, **windows.advanced.memory_protection.memory_scan_collect_sam** and **windows.advanced.memory_protection.shellcode_enhanced_pe_parsing** options and set the value to **true**.



Enabling sample collection

Note on data volume

As mentioned above, these steps are for labs, sandboxes, testing, etc. These settings can generate a lot of data, so setting these for production will need resourcing and sizing considerations.

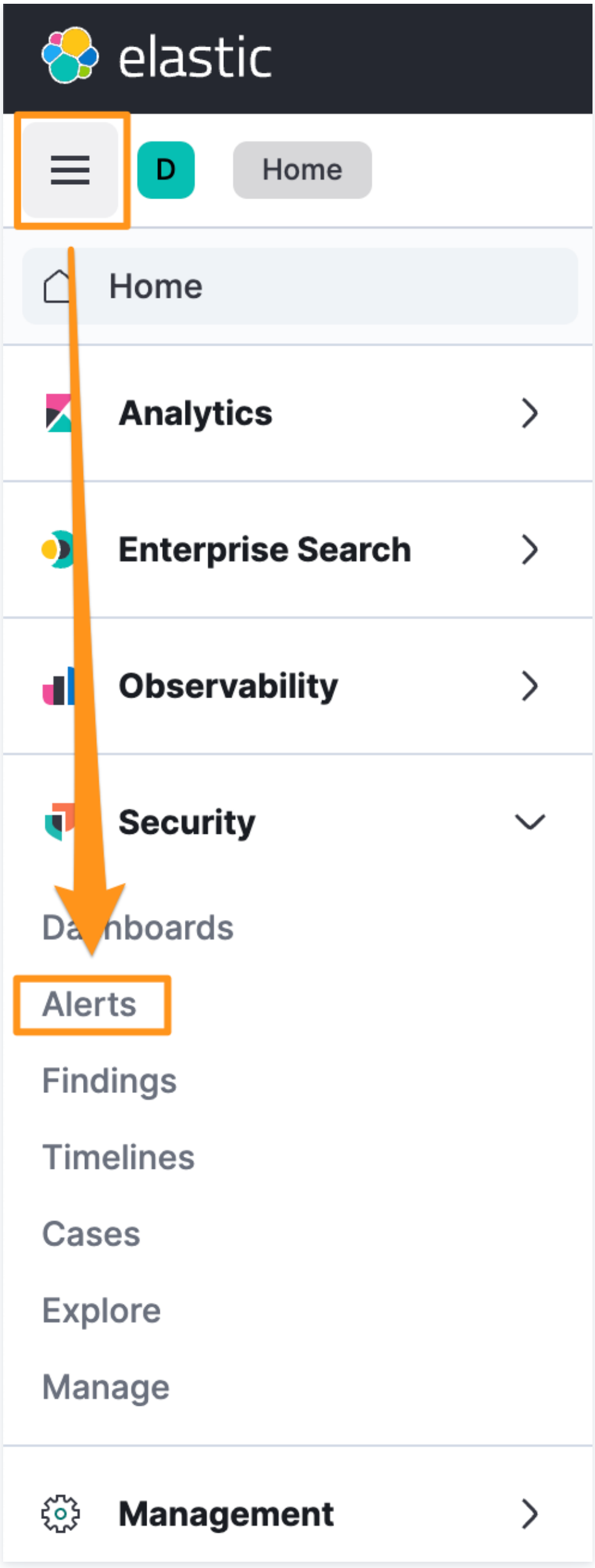
If you’re making a policy for Linux or macOS, repeat these for the proper OS.

Once we’re done with all of the post-installation configurations, we can click the blue Save integration button.

Enabling Elastic’s Prebuilt Detection Rules

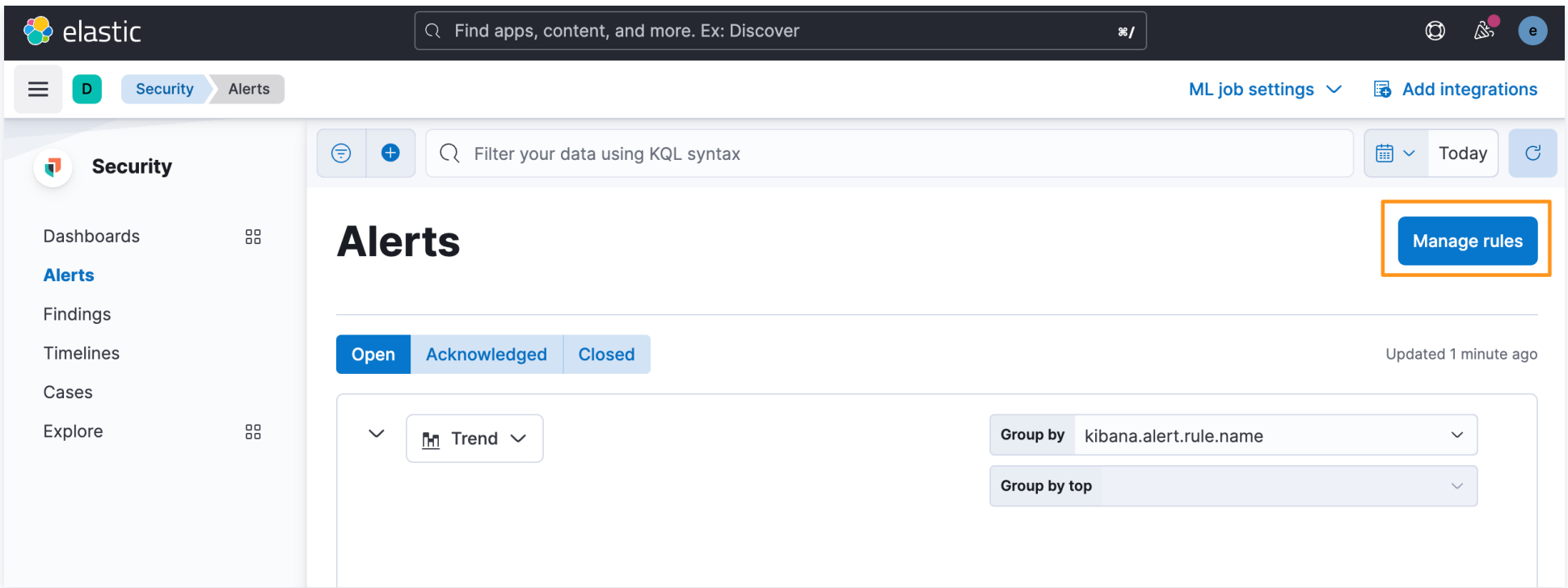
Now that we have created our Fleet agent policy we need to enable the set of pre-built detection rules associated with the OS or platform we will be deploying on (e.g Windows). To do this you will need to go to the Alerts page within the security app.

Click on the hamburger menu and select Alerts, under the Security solution.



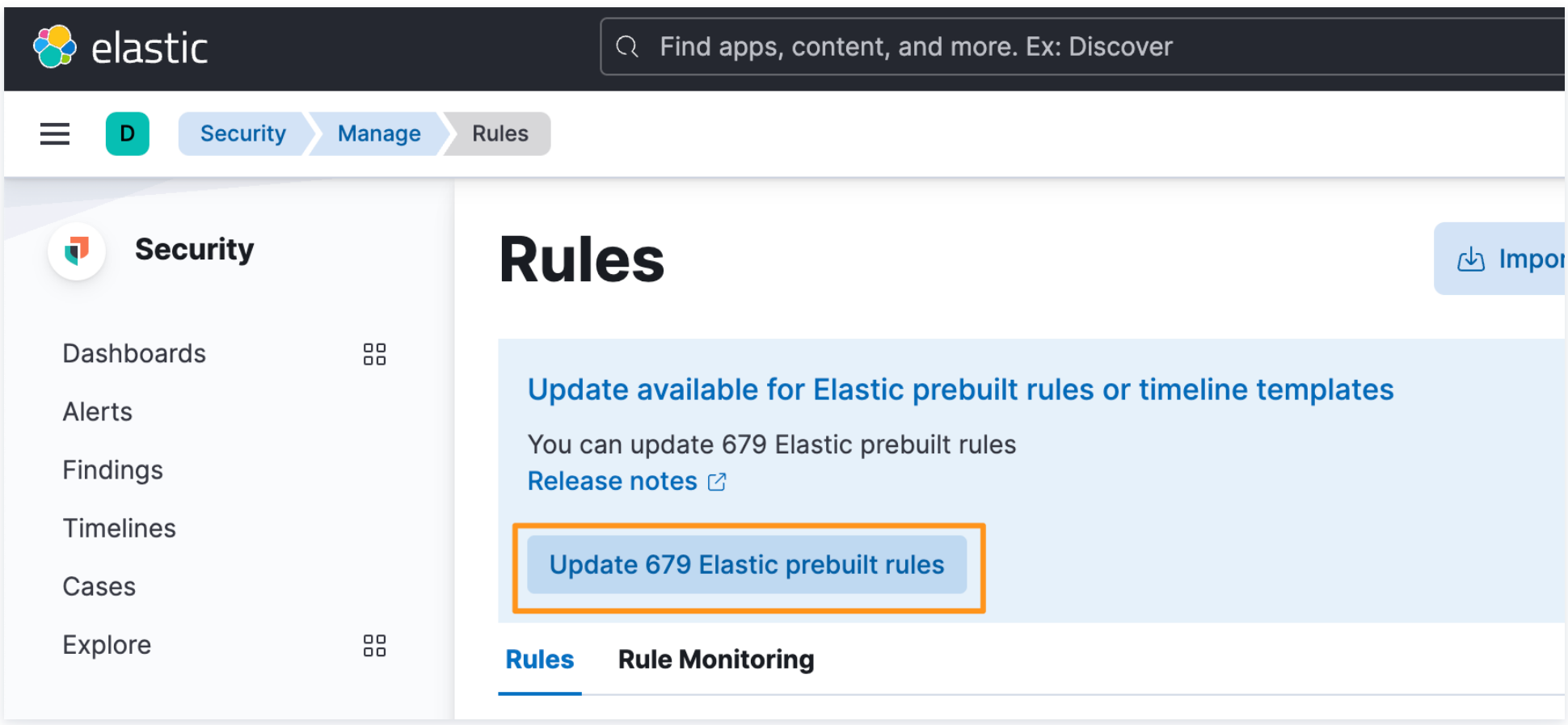
Access the Alerts section

Next, click on the blue Manage Rules button.



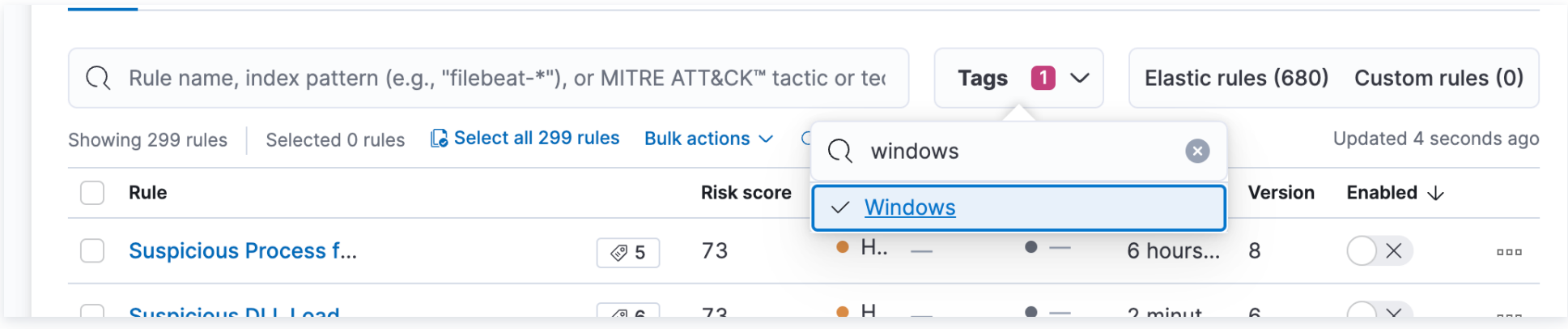
Access the Manage rules interface

Once on the Rules page you can update all of the prebuilt rules provided by Elastic by clicking on the “Update Elastic prebuilt rules” button. The update framework is enabled when you go into the “Manage rules” section for the first time, if the “Update Elastic prebuilt rules” button isn’t present, refresh the screen.



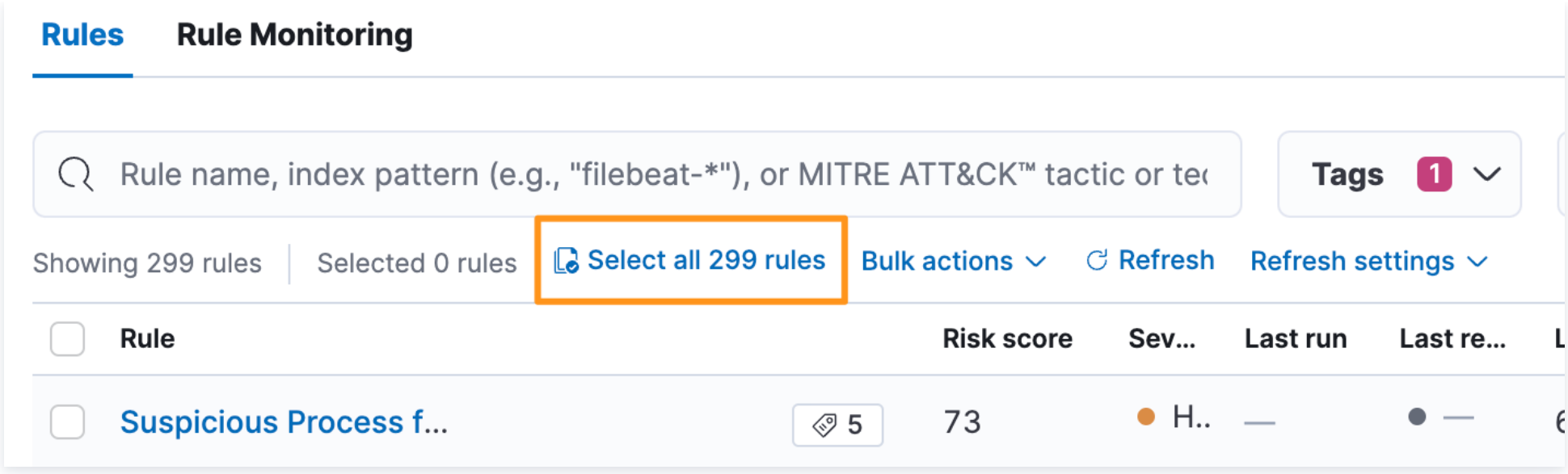
Update Elastic prebuilt rules

Once the rules have been updated, you can browse the available detection rules, search them by a number of different patterns or simply filter by tag, which is what we will do here by searching for Windows rules.



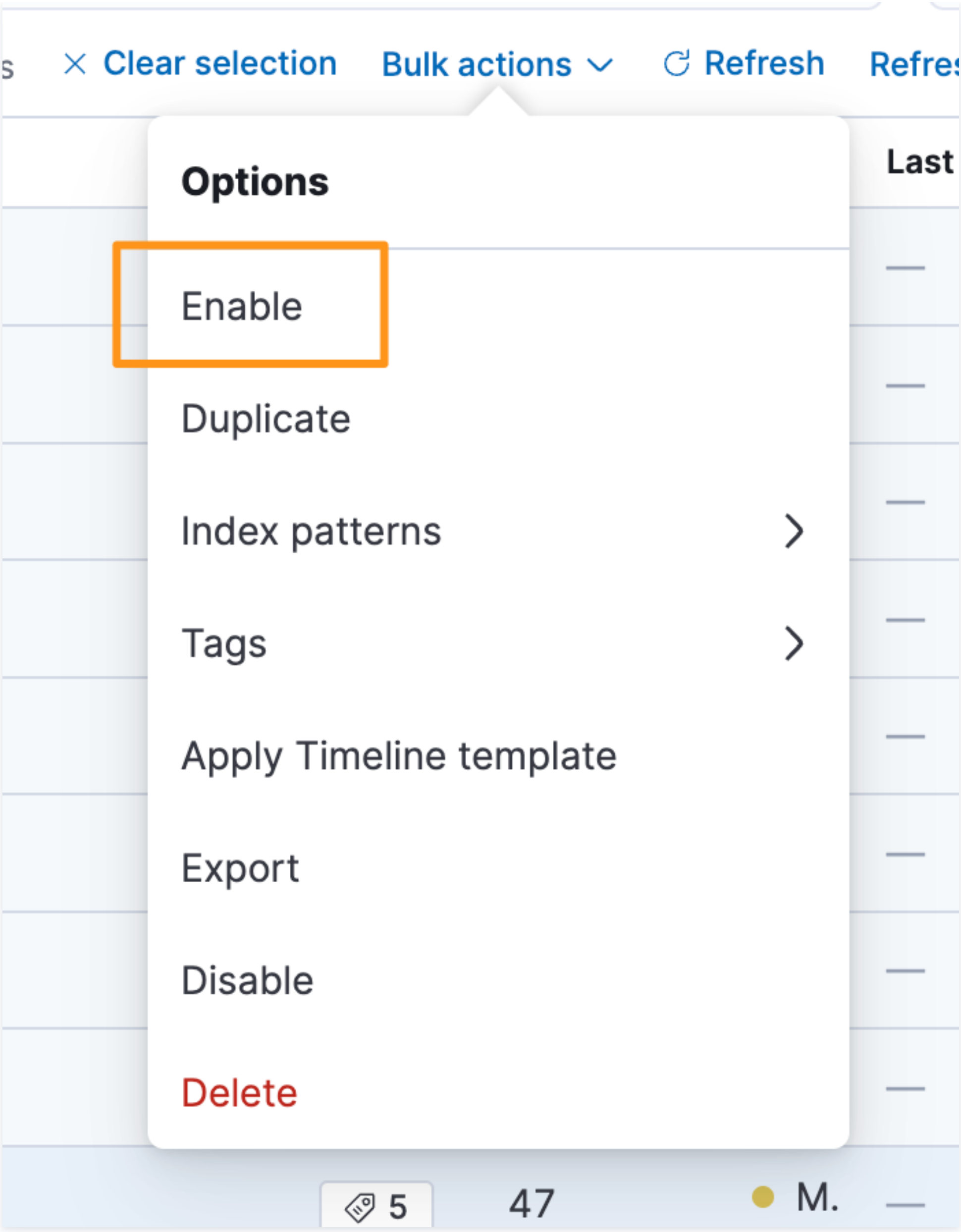
Filter for Windows rules

Now we can select all of the Windows rules.



Selecting all Windows rules

Once all of the rules have been selected, we can bulk enable them.



Bulk enable Windows rules

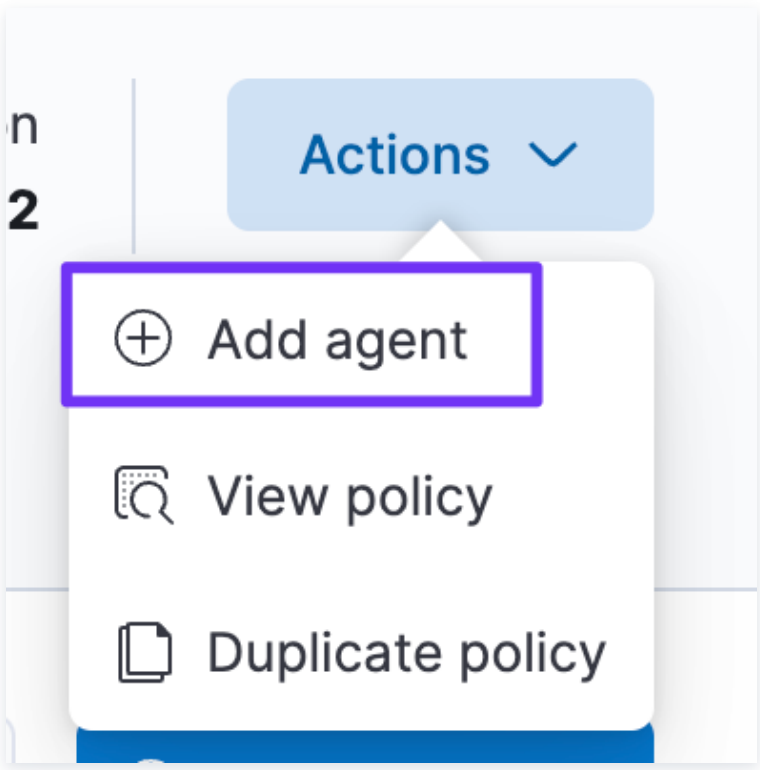
Performance considerations with bulk rule enablement

As the Elastic Container Project runs completely inside single Docker containers, performance impacts could be noticed if you enable all of the rules available. Explore the different rules and enable or disable them based on your infrastructure and use cases.

After we have enabled these rules they will be live and will be run against the data your endpoint agent sends into your stack. When the Detection Engine rules are triggered, they will be raised in the Alerts page in the Security Solution.

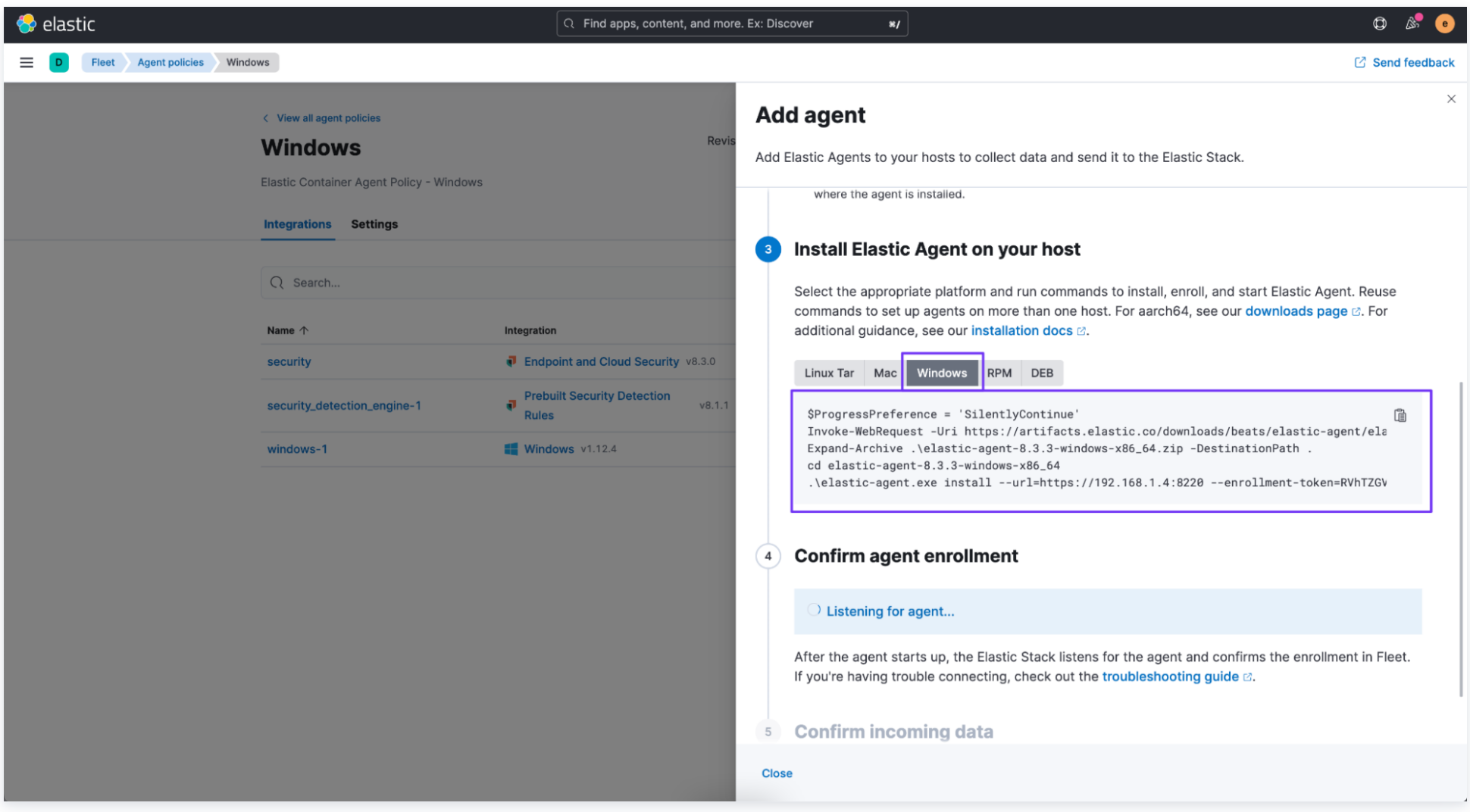
Enrolling an Elastic Agent

Still in Fleet, we have several ways to add an Elastic Agent. The most straightforward is from within the policy that we want to enroll an Elastic Agent into (otherwise you have to specify which policy you want to use). It doesn't really matter which approach you use, but clicking on the Actions button and then Add agent works from just about anywhere in Fleet.



Adding Elastic Agent

Scroll down and click on the OS that you're going to be installing the Elastic Agent on, and copy/paste the instructions directly into a terminal window on the host you're going to be installing the agent onto. Note, if you're using Windows, use a Powershell CLI that is running as (or elevated to) an account with administrative entitlements.



Powershell commands to add an Elastic Agent

Of note, because all of our TLS certificates are self-signed, we need to append the **–insecure** flag. This is unnecessary if you are using trusted certificates.

```
.\elastic-agent.exe install --url=https://[stack-ip]:8220 --enrollment-token=[token] --insecure
```

```
PS C:\users\variable\Desktop\elastic-agent-8.3.3-windows-x86_64> .\elastic-agent.exe install --url
https://172.20.10.4:8220 --enrollment-token=RVhTZGVZSUJqMEVzd3hjckVMcTM6ZUpHU2JJWWpTSUNGQ3EyWms4VG
FUQ== --insecure
Elastic Agent will be installed at C:\Program Files\Elastic\Agent and will run as a service. Do yo
 want to continue? [Y/n]:Y
{"log.level":"warn","@timestamp":"2022-08-13T13:40:31.275-0400","log.logger":"tls","log.origin":{"
ile.name":"tlscommon/tls_config.go","file.line":104},"message":"SSL/TLS verifications disabled.","
cs.version":"1.6.0"}
{"log.level":"info","@timestamp":"2022-08-13T13:40:31.589-0400","log.origin":{"file.name":"cmd/enr
ll_cmd.go","file.line":471},"message":"Starting enrollment to URL: https://172.20.10.4:8220/","ecs
version":"1.6.0"}
{"log.level":"warn","@timestamp":"2022-08-13T13:40:31.718-0400","log.logger":"tls","log.origin":{"
ile.name":"tlscommon/tls_config.go","file.line":104},"message":"SSL/TLS verifications disabled.","
cs.version":"1.6.0"}
{"log.level":"info","@timestamp":"2022-08-13T13:40:32.523-0400","log.origin":{"file.name":"cmd/enr
ll_cmd.go","file.line":273},"message":"Successfully triggered restart on running Elastic Agent.","
cs.version":"1.6.0"}
Successfully enrolled the Elastic Agent.
Elastic Agent has been successfully installed.
```

Enrolling the Elastic Agent into Fleet

Back in Kibana, we can see confirmation that the Elastic Agent installed on the host and that data is being recorded into Elasticsearch.

Add agent

Add Elastic Agents to your hosts to collect data and send it to the Elastic Stack.

Linux Tar | Mac | Windows | RPM | DEB

```
$ProgressPreference = 'SilentlyContinue'
Invoke-WebRequest -Uri https://artifacts.elastic.co/downloads/beats/elastic-agent/ela
Expand-Archive .\elastic-agent-8.3.3-windows-x86_64.zip -DestinationPath .
cd elastic-agent-8.3.3-windows-x86_64
.\elastic-agent.exe install --url=https://172.20.10.4:8220 --enrollment-token=RVhTZGV
```

Agent enrollment confirmed

✓ 1 agent has been enrolled.

View enrolled agents

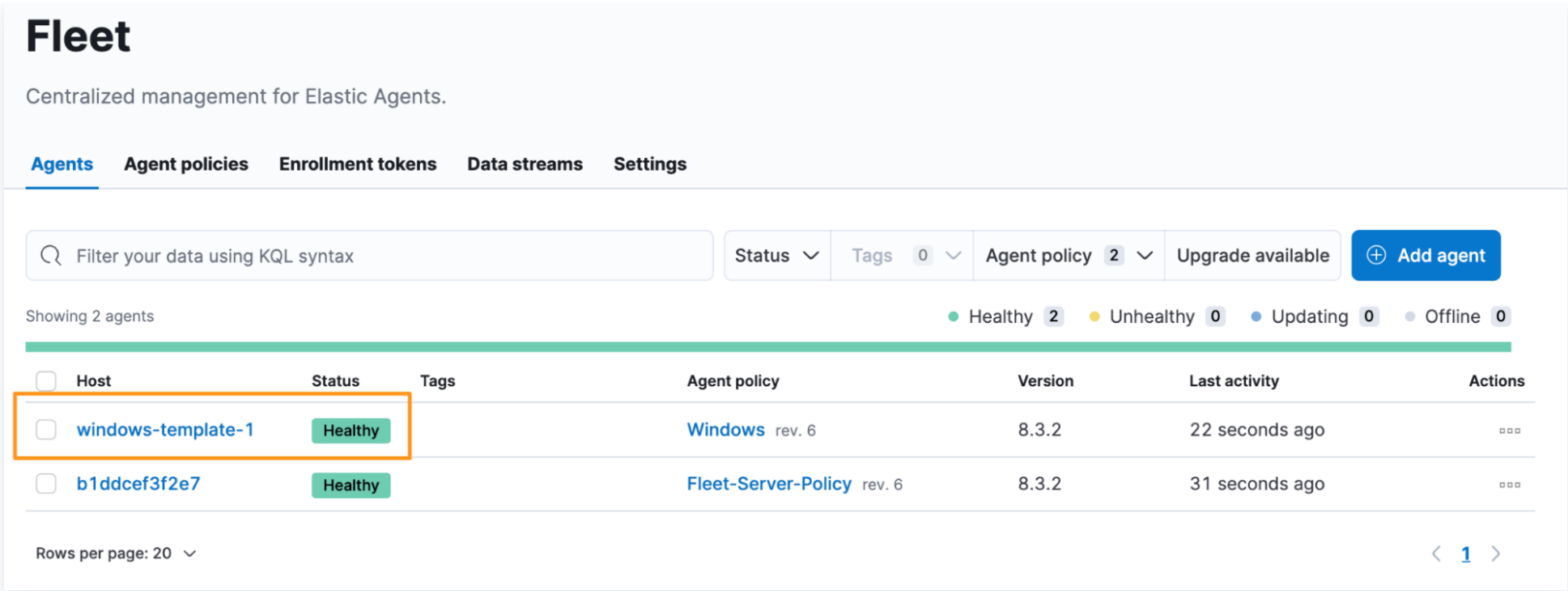
Incoming data confirmed

✓ Incoming data received from 1 of 1 recently enrolled agent.

Close

Verifying Elastic Agent enrollment

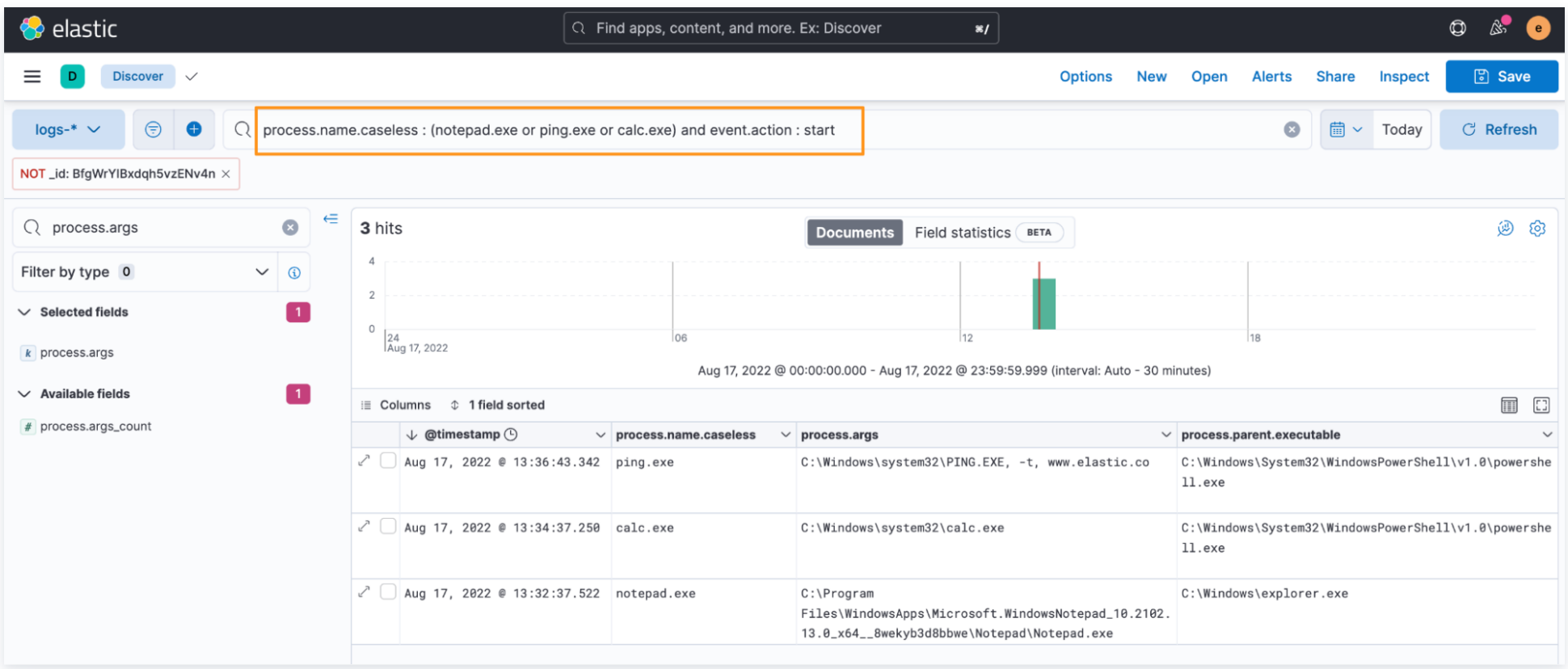
We can see that the Elastic Agent is reporting into Fleet and is healthy.



Verify Elastic Agent health

If we go into the Discover tab, we can see various event types reporting into Elasticsearch. We can generate some test data by opening **notepad.exe**, **calc.exe**, and **ping.exe -t www.elastic.co** on the host. From Discover, we can make a simple query to validate that we're seeing the data:

```
process.name.caseless : (notepad.exe or ping.exe or calc.exe)
```



Verifying data is being sent to Elasticsearch

Now that we've validated that we're seeing data. Let's fire some malware!

Test fire some malware

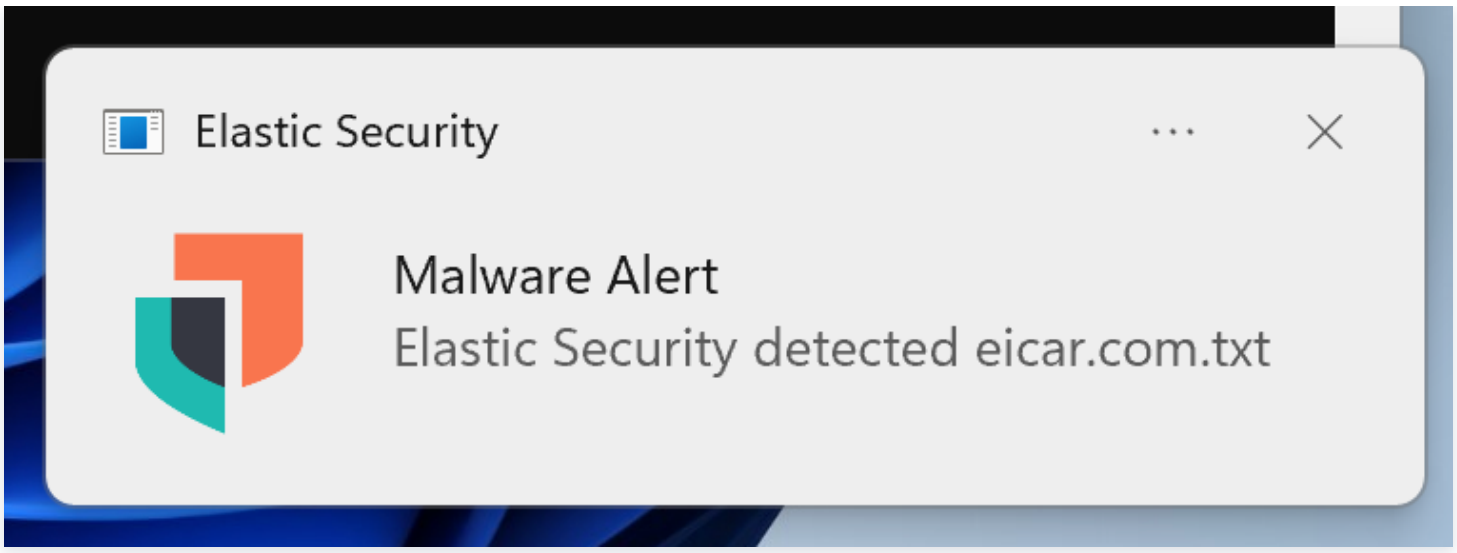
There are a lot of places you can download malware from, but for this test, we'll simply use the industry standard **EICAR anti malware test file** to check the functionality.

The EICAR test is a file that is universally identified by security vendors and is used to test the operation of anti malware software and platforms. It contains a single string and is non-malicious.

From within the Windows host, we'll use Powershell to download the EICAR file.

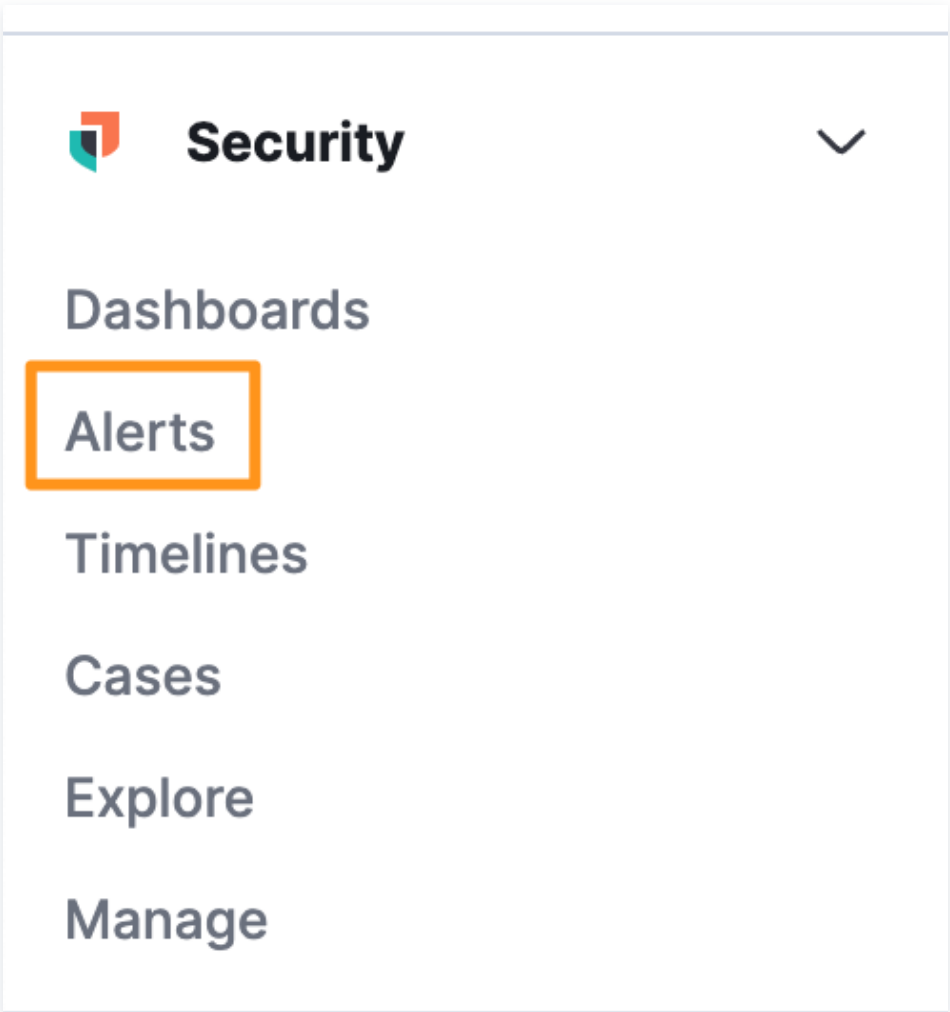
```
Invoke-WebRequest -Uri "https://secure.eicar.org/eicar.com.txt" -OutFile "eicar.txt"
```

As expected, the event was immediately identified by the Elastic Agent's security integration.



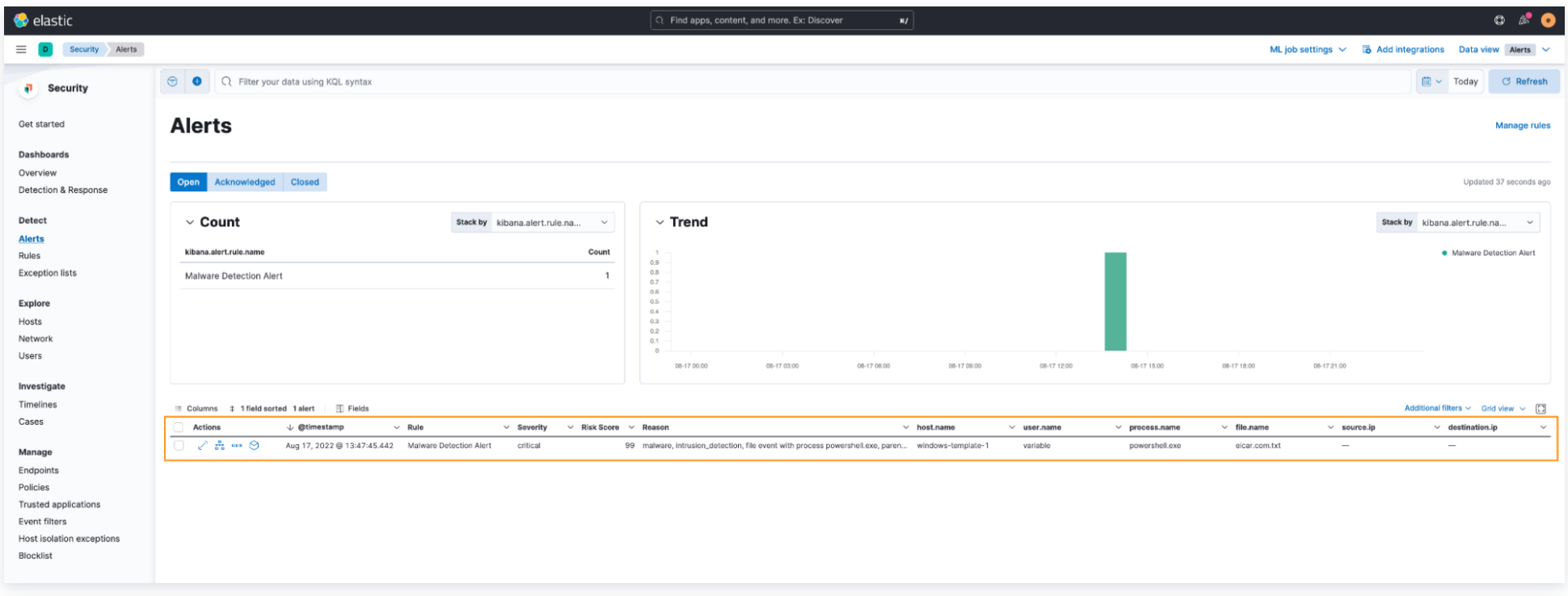
Elastic Security detected the EICAR test file

After a few minutes, the events are recorded into the Security Solution within Kibana. You can get there by clicking on the hamburger menu and then clicking on the Alerts section.



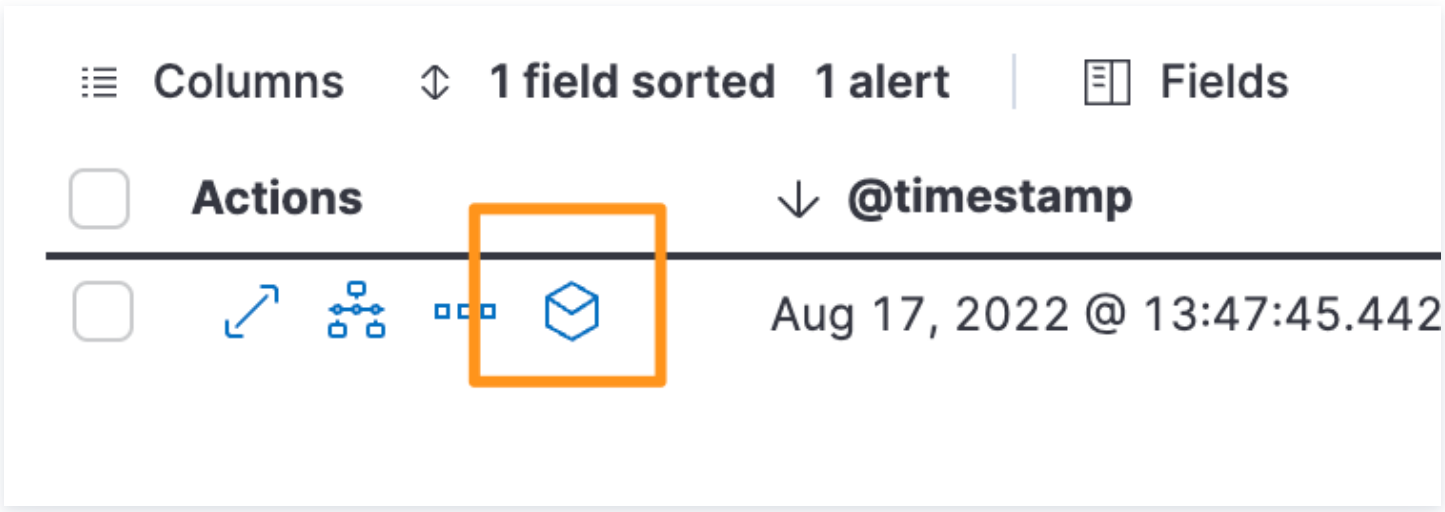
Viewing Security alerts

Here we can see the alert populated.



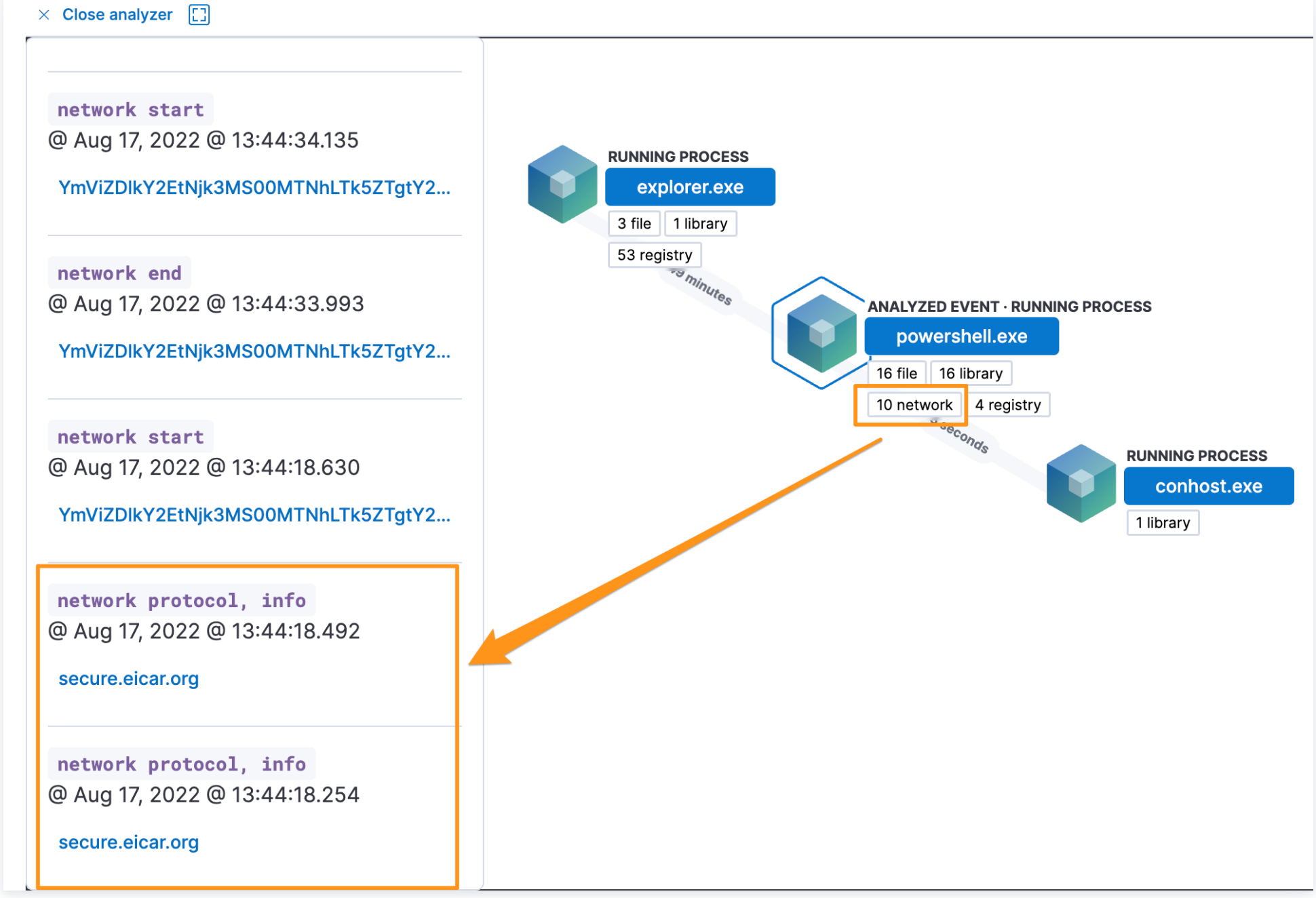
Alert in the Security Solution

If we click on the Analyzer button, we can dig into the event to identify the process that generated the event.



Analyzer button

In our example, we can see **powershell.exe** generated the event and this includes the correlated network events - **secure.eicar.org**, which is where the EICAR test file was downloaded from.



Analyzer view

Summary

In this publication, we introduced you to the Elastic Stack and an open source project that can be used to quickly and securely stand up the entire stack for testing, labs, and security research.

Kibana and the Security Solution are powerful tools that are built by incident responders, threat hunters, and intelligence analysts with security practitioners in mind. To learn more about how to use these tools, **Elastic has some great (free and paid) training** that can help learn how to use Kibana for threat hunting.