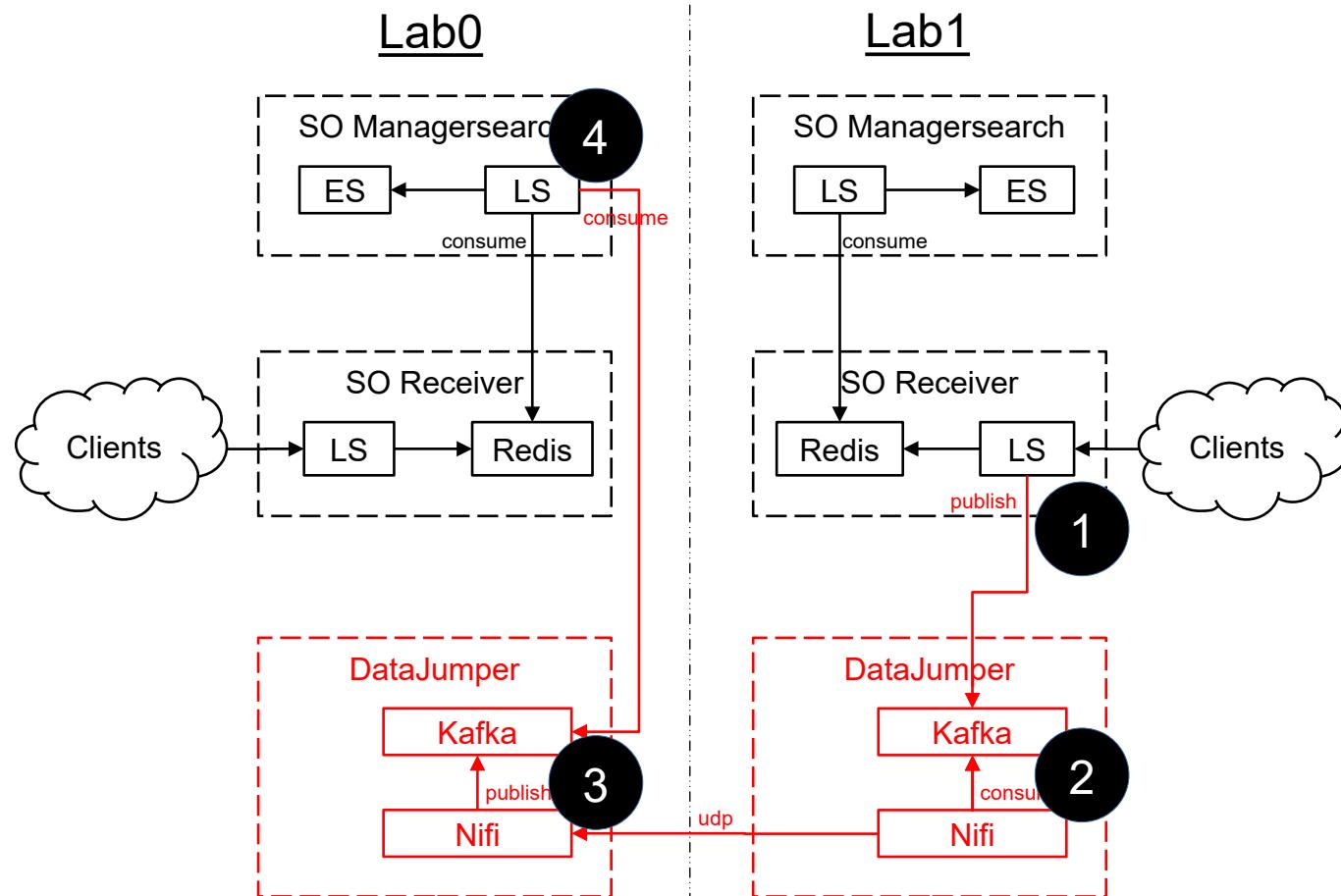


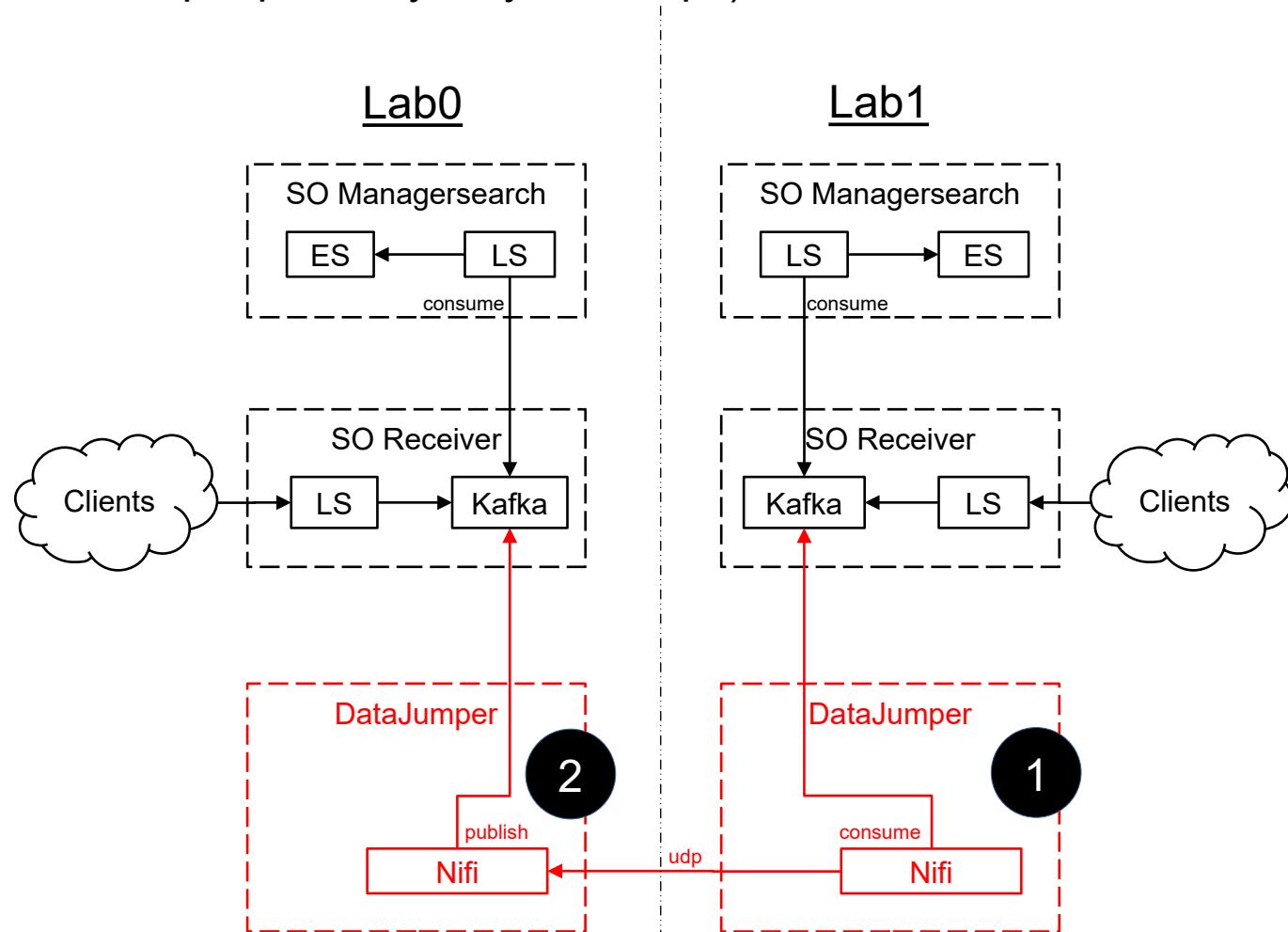
Aggregate multiple independent Security Onion envs.

A HOWTO in four steps



Aggregate multiple independent Security Onion envs.

Pro version (out of scope, probably only two steps)



Lab1 configuration

Based on the Open Source
version of Security Onion

0. Elastic namespaces

In Elastic Fleet, modify the value of the namespace field that each event will get, so we can distinguish between events from different environments. Because every event will have this value, we can realise environment specific alerts and alert suppression rules with a single ruleset.

The screenshot shows the Elastic Fleet interface with the 'Fleet' tab selected. The 'endpoints-initial' policy is displayed, which is an 'Initial Endpoint Policy'. The 'Integrations' tab is active. A search bar at the top has 'Search...' placeholder text. Below it, a 'Namespace' dropdown is set to 'so-recv1' and a blue 'Add integration' button is visible. The main table lists various integration policies and their details:

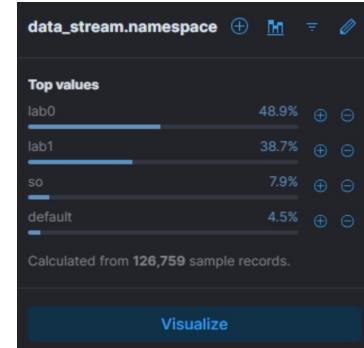
Integration policy	Integration	Namespace	Output	Actions
elastic-defend-endpoints	Elastic Defend v8.17.0	lab1 ⓘ	so-recv1 ⓘ	...
osquery-endpoints	Osquery Manager v1.16.0	lab1 ⓘ	so-recv1 ⓘ	...
system-endpoints	System v1.67.0	lab1 ⓘ	so-recv1 ⓘ	...
windows-defender ⓘ	Custom Windows Event Logs v2.4.0	default	so-recv1 ⓘ	...
windows-endpoints	Windows v2.5.0	lab1 ⓘ	so-recv1 ⓘ	...

(Unfortunately it isn't configurable for every integration policy, like e.g. the windows-defender integration)

The end result will be alerts generated using a single ruleset in Lab0 for both Lab1 and Lab0 that still distinguish between environments.

The screenshot shows the Kibana Security interface with the 'Alerts' tab selected. On the left, a sidebar lists 'Dashboards', 'Rules', 'Alerts' (selected), 'Attack discovery', 'Findings', 'Timelines', 'Intelligence', 'Explore', 'Get started', and 'Manage'. The main area has a search bar at the top. Below it, a 'Severity levels' section shows 9 Medium alerts and 4 Low alerts. A large circular gauge indicates 13 alerts. To the right, a 'Fields' section is visible, followed by a table of 13 alerts. The table columns are: Actions, @timestamp, data_stream_, Rule, Assignees, Severity, and Risk Score. Each row contains a set of icons for actions, a timestamp, a redacted stream name, a rule name, an assignee, a severity level, and a risk score. The first few rows show alerts from Nov 17, 2025, and the last few rows show alerts from Nov 7, 2025.

Actions	@timestamp	data_stream_	Rule	Assignees	Severity	Risk Score
...	Nov 17, 2025 @ 16:33:01.876	lab1	Disable Windows Firewall R...		medium	47
...	Nov 17, 2025 @ 15:43:01.556	lab1	Disable Windows Firewall R...		medium	47
...	Nov 17, 2025 @ 13:08:01.045	lab0	Disable Windows Firewall R...		medium	47
...	Nov 13, 2025 @ 16:32:51.840	lab1	Windows Event Logs Cleared		low	21
...	Nov 13, 2025 @ 16:32:51.837	lab1	Windows Event Logs Cleared		low	21
...	Nov 13, 2025 @ 16:32:51.834	lab1	Windows Event Logs Cleared		low	21
...	Nov 13, 2025 @ 16:32:51.830	lab1	Windows Event Logs Cleared		low	21
...	Nov 13, 2025 @ 15:22:52.066	lab1	Potential Privilege Escalati...		medium	47
...	Nov 13, 2025 @ 15:22:52.062	lab1	Potential Privilege Escalati...		medium	47
...	Nov 13, 2025 @ 14:17:48.824	lab1	Disable Windows Firewall R...		medium	47
...	Nov 7, 2025 @ 11:12:32.228	lab1	Disable Windows Firewall R...		medium	47
...	Nov 7, 2025 @ 09:41:09.716	lab0	Disable Windows Firewall R...		medium	47
...	Nov 7, 2025 @ 09:41:09.707	lab0	Disable Windows Firewall R...		medium	47



Discover - Elastic

Not secure https://192.168.2.40/kibana/app/discover#/_g/filters>[]/_query[language<query>query]refreshInterval(pause, value60000)timeFrom(now-14d,to... Update

elastic Discover Find apps, content, and more. Try ES|QL Inspect Alerts + Save

Data view alerts-security-alerts-default.apm... Filter your data using KQL syntax Last 14 days

data.stream.namespace: lab1 ✘

Auto interval No breakdown

Nov 11, 2025 @ 15:26:05.072 - Nov 25, 2025 @ 15:26:05.072 (interval: Auto - 12 hours)

Documents (650,840) Patterns Field statistics Sort Fields

Filter bar: Search field name: [] Popular fields []

- data.stream.namespace
- host.hostname
- host.name

Available fields []

- @indexstamp
- @version
- @xpack.ephemeral_id
- agent.id
- agent.name
- agent.type
- agent.version
- audience
- component.binary
- component.dataset
- component.id
- component.state
- component.state
- component.type
- correlated_id
- correlated_id
- data.stream.dataset
- data.stream.namespace
- data.stream.type

Add a field

Rows per page: 100

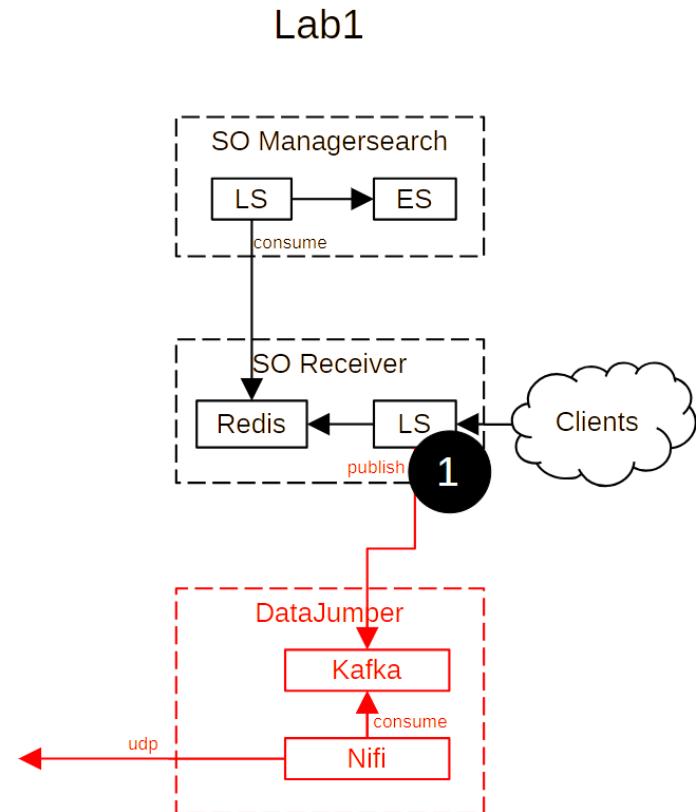
Search results are limited to 100 documents. Load more

1. Kafka output

OVERVIEW

The Redis config in SO is set up for load balancing using a simple queue, not a pub/sub channel. That means that reading the same Redis queue will remove events and lead to dataloss. Instead, we add a custom Logstash pipeline to the Receiver. This does duplicate events, into our own Kafka Topic. SO support customizing Logstash pipelines for these SO roles:

- Receiver
- Manager
- Search
- Fleet



1. Kafka output

a) Security Onion Console

The screenshot shows the Security Onion Console interface. The left sidebar has a dark theme with various navigation options like Overview, Alerts, Dashboards, Hunt, Cases, PCAP, Grid, Downloads, Administration, Users, Grid Members, Configuration, License Key, Tools, Kibana, Elastic Fleet, Osquery Manager, InfluxDB, CyberChef, and Navigator. The Configuration section is currently selected. The main panel shows a list of pipeline configurations under the logstash > defined_pipelines > receiver [adv] group. The list includes custom0, custom1, custom2, custom3, custom4, fleet, and manager. A modal window is open over the list, showing the current grid value which includes so/0011_input_endgame.conf, so/0012_input_elastic_agent.conf.jinja, so/0013_input_lumberjack_fleet.conf, so/9999_output_redis.conf.jinja, and custom/lab1_kafka.conf. Below the modal, there are buttons for DUPLICATE and VIEW DEFAULT. At the bottom of the main panel, there's a search bar with the placeholder "Select a node to modify".

b) File on Manager

/opt/so/saltstack/local/salt/logstash/pipelines/config/custom/lab1_kafka.conf

```
output {
    kafka {
        codec => json
        bootstrap_servers => "192.168.37.129:9092"
        topic_id => "lab1"
    }
}
```

c) Sync Grid

d) Check on Receiver via SSH:

```
sudo salt-call pillar.get logstash:defined_pipelines
```

```
[admin@so-recv1 ~]$ sudo salt-call pillar.get logstash:defined_pipelines
local:
-----
receiver:
  - so/0011_input_endgame.conf
  - so/0012_input_elastic_agent.conf.jinja
  - so/0013_input_lumberjack_fleet.conf
  - so/9999_output_redis.conf.jinja
  - custom/lab1_kafka.conf
[admin@so-recv1 ~]$
```

e) Don't want to wait 15 minutes? Execute on the Receiver:

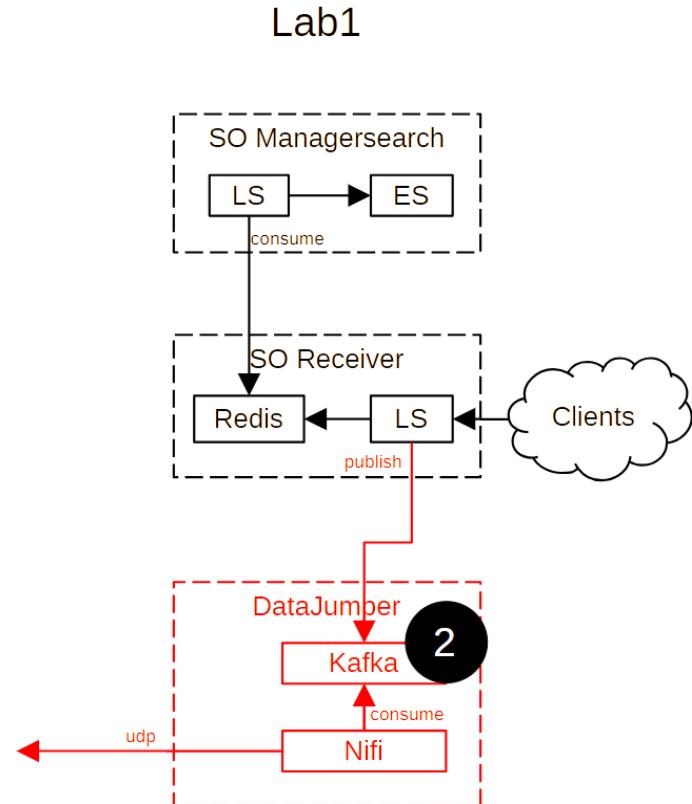
```
sudo salt-call state.highstate
```

2. Kafka and Nifi

OVERVIEW

The Logstash pipeline from the previous step duplicates events to a new Kafka Topic. To receive these events and transmit them to Lab0, we install a Linux system with:

- 1)Podman
- 2)Kafka container
- 3)Nifi container



2. Kafka and Nifi

a) On an internet connected Linux system:

```
podman pull apache/kafka:latest  
podman pull apache/nifi:latest
```

```
podman save -o apache-kafka.tar apache/kafka:latest  
podman save -o apache-nifi.tar apache/nifi:latest
```

b) Transfer the 2 tar files to Lab1

c) On the Lab1 Linux system:

```
podman load -i /tmp/apache-kafka.tar  
podman load -i /tmp/apache-nifi.tar
```

```
podman images
```

2. Kafka and Nifi

d) Start Kafka container (no authentication, auto topic creation)

```
podman run --detach --name kafka --hostname kafka -p 9092:9092 \
-e KAFKA_NODE_ID=1 \
-e KAFKA_PROCESS_ROLES=broker,controller \
-e KAFKA_CONTROLLER_LISTENER_NAMES=CONTROLLER \
-e KAFKA_CONTROLLER_QUORUM_VOTERS=1@localhost:9093 \
-e KAFKA_LISTENERS=A://:9092,CONTROLLER://:9093 \
-e KAFKA_LISTENER_SECURITY_PROTOCOL_MAP=A:PLAINTEXT,CONTROLLER:PLAINTEXT \
-e KAFKA_ADVERTISED_LISTENERS=A://192.168.37.129:9092 \
-e KAFKA_INTER_BROKER_LISTENER_NAME=A \
-e KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR=1 \
-e KAFKA_TRANSACTION_STATE_LOG_REPLICATION_FACTOR=1 \
-e KAFKA_TRANSACTION_STATE_LOG_MIN_ISR=1 \
-e KAFKA_NUM_PARTITIONS=1 \
apache/kafka:latest
```

e) Check logs for successful start

```
podman logs -f kafka
```

```
[...knip...]
[2025-11-03 10:26:53,014] INFO Kafka version: 4.0.0 (org.apache.kafka.common.utils.AppInfoParser)
[2025-11-03 10:26:53,017] INFO Kafka commitId: 985bc99521dd22bb (org.apache.kafka.common.utils.AppInfoParser)
[2025-11-03 10:26:53,019] INFO Kafka startTimeMs: 1762165613014 (org.apache.kafka.common.utils.AppInfoParser)
[2025-11-03 10:26:53,023] INFO [KafkaRaftServer nodeId=1] Kafka Server started (kafka.server.KafkaRaftServer)
```

f) Open local firewall

```
firewall-cmd --add-port=9092/tcp --permanent
firewall-cmd --reload
```

2. Kafka and Nifi

g) Start Nifi container

```
podman run --name nifi --detach -p 8443:8443 \
-e SINGLE_USER_CREDENTIALS_USERNAME=admin \
-e SINGLE_USER_CREDENTIALS_PASSWORD=eenwachtwoordmetminimaal12karakters \
-e NIFI_WEB_HTTPS_PORT=8443 \
apache/nifi:latest
```

h) Check logs for successful start (can take a while)

```
podman logs -f nifi
```

```
[...knip...]
2025-11-06 10:54:08,040 INFO [main] org.eclipse.jetty.server.Server Started oejs.Server@13a268cd{STARTING}[12.0.27,sto=0] @230155ms
2025-11-06 10:54:08,060 INFO [main] org.apache.nifi.web.server.JettyServer Started Server on https://906f398d8249:8443/nifi
2025-11-06 10:54:08,212 INFO [main] o.a.n.runtime.StandardManagementServer Started Management Server on http://127.0.0.1:52020
2025-11-06 10:54:08,232 INFO [main] org.apache.nifi.runtime.Application Started Application in 226.876 seconds (226876498547 ns)
```

i) Open local firewall

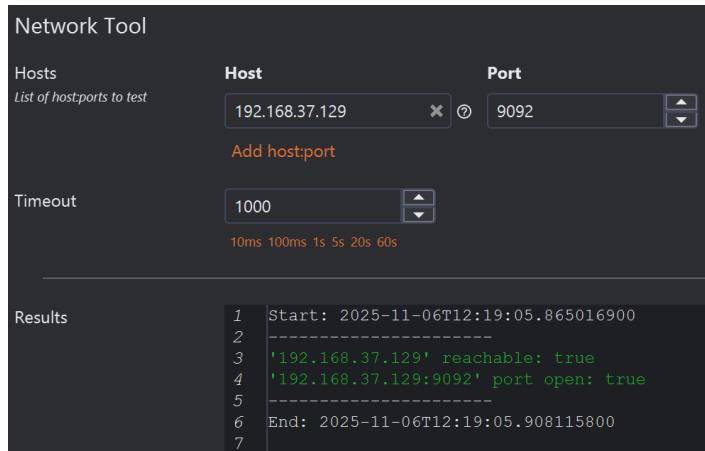
```
firewall-cmd --add-port=8443/tcp --permanent
firewall-cmd --reload
```

j) Add “906f398d8249” to the hosts file on your lab workstations (Otherwise you can't configure Nifi pipelines)

2. Kafka and Nifi

k) Install KafkIO (Linux, Windows, macOS) or some other Kafka GUI tool

l) Check connectivity
(Add Cluster → Check network reachability)



2. Kafka and Nifi

m) Are there messages in the lab1 topic?

The screenshot shows the Confluent Cloud UI interface for managing a Kafka cluster named "Lab1 Kafka". The left sidebar navigation includes "Add Cluster", "Kafka Clusters", "Overview", "Brokers", "Topics" (which is selected), "Consumers", "Schema Registry", "ACL", "Kafka Connect", and "ksqlDB". The main content area is titled "Lab1 Kafka" and shows the "Topics" tab. The "lab1" topic is highlighted with an orange background. The table below lists 5,000 messages received, with the last message shown having an offset of 2216045, partition 0, and timestamp 2025-11-06T11:22:49.618Z. The message key is <null> and the value is a JSON object starting with {"mess...". The bottom status bar indicates "Cluster: Lab1 Kafka; status: connected" and memory usage "Mem used: 138.23 MB | Max: 29.50 GB".

Offset	Partition	Size (serialized)	Time	Key	Value
2216048	0	1 KB	2025-11-06T11:22:49.618Z	<null>	{"mess...
2216047	0	1 KB	2025-11-06T11:22:49.618Z	<null>	{"mess...
2216046	0	1 KB	2025-11-06T11:22:49.618Z	<null>	{"mess...
2216045	0	1 KB	2025-11-06T11:22:49.618Z	<null>	{"mess...
2216044	0	1 KB	2025-11-06T11:22:49.618Z	<null>	{"mess...

Showing: 5,000 | Received: 5,000 | Received size: 9.1M

2216045 0 FWD 2025-11-06T11:22:49.618Z <null> {"mess...

Key Message Headers Raw

Search event/message... ↑ ↓ 0 found Display as: STRING

Topics: 4

Cluster: Lab1 Kafka; status: connected Mem used: 138.23 MB | Max: 29.50 GB

2. Kafka and Nifi

n) Is logstash listed as a consumer Group ID?

The screenshot shows the Confluent Cloud UI interface for managing a Kafka cluster named "Lab1 Kafka". The left sidebar menu is visible, with "Consumers" selected. The main panel displays a table of consumers:

Group ID	Number of Members	Number of Topics	Consumer Lag	Coordinator	State
logstash	1	1	0	1	STABLE
nifi1	1	1	0	1	STABLE

At the bottom of the UI, there is a "Health Log" section showing Kafka status messages:

```
12:24:10 Topics (visible): 4
12:24:40 Lab1 Kafka
12:24:40 Status: online
12:24:40 Last error: none
12:24:40 Controller ID: 1
12:24:40 Topics (visible): 4
12:25:10
12:25:10 Lab1 Kafka
12:25:10 Status: online
12:25:10 Last error: none
12:25:10 Controller ID: 1
12:25:10 Topics (visible): 4
```

Autoscroll health checkbox is checked.

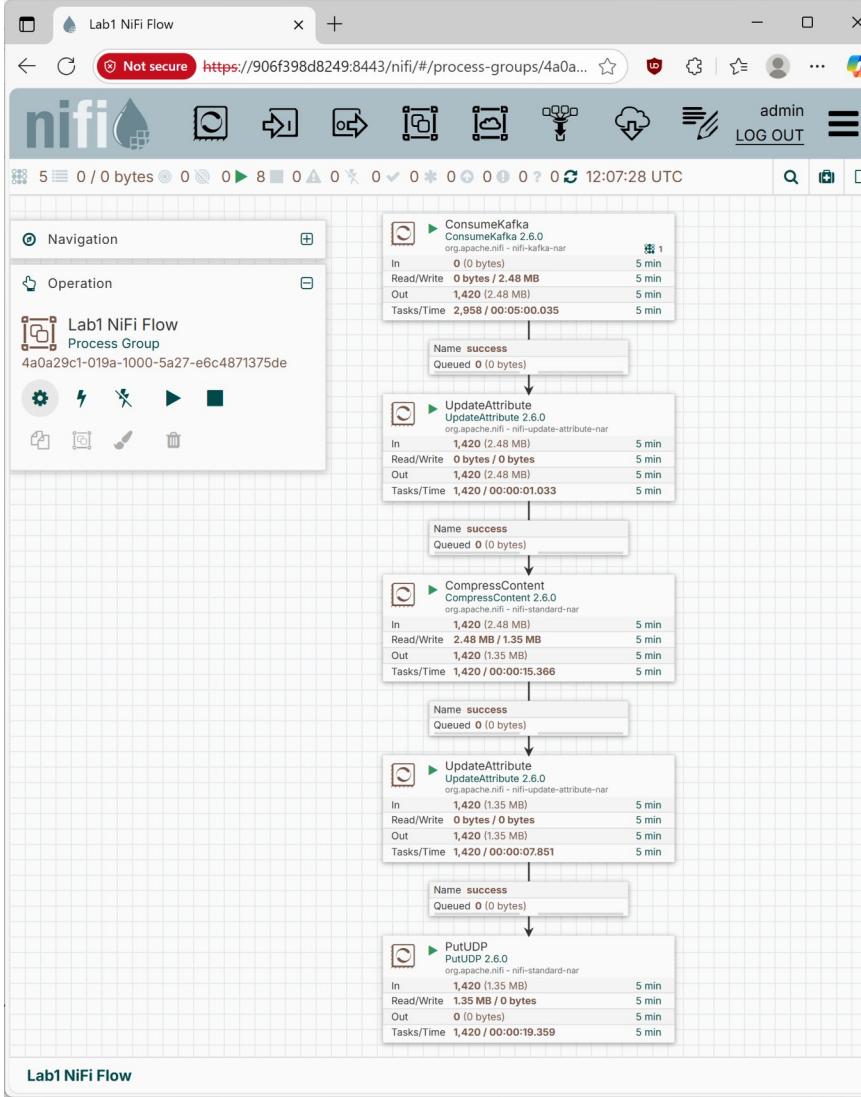
Cluster: Lab1 Kafka; status: connected

In the subsequent steps we will create an additional subscriber that uses a different group id to consume events from this topic.

2. Kafka and Nifi

o) Configurere dataflow in Nifi by adding, configururing and linking these processors:

- 1)ConsumeKafka
- 2)UpdateAttribute
- 3)CompressContent
- 4)UpdateAttribute
- 5)PutUDP



➡ (Read lab1 topic)

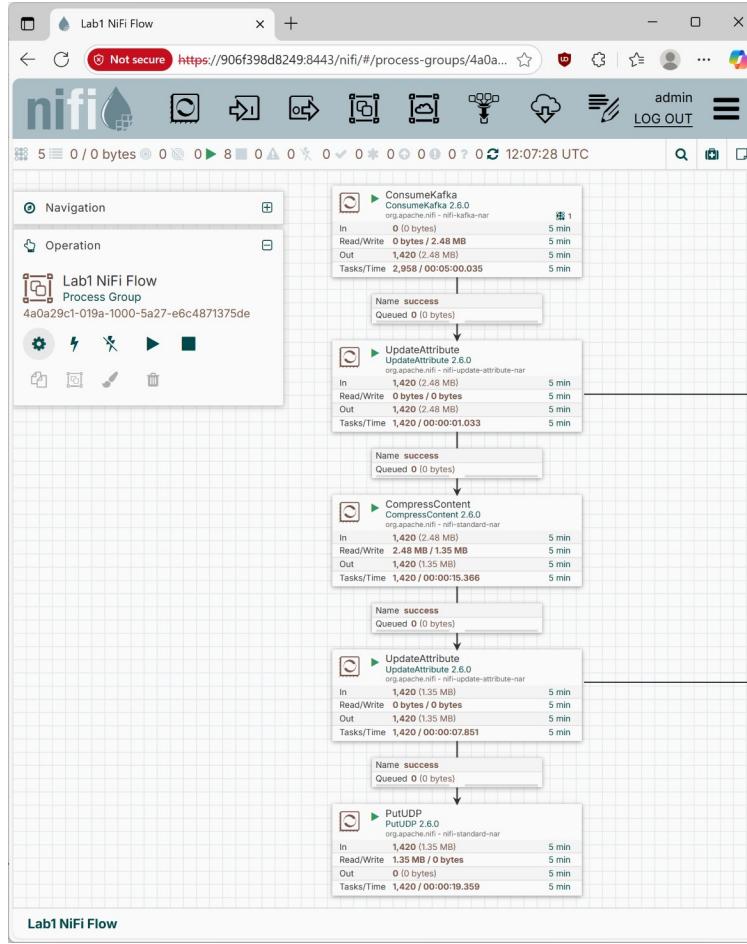
➡ (Keep a running max stats of the message size. Should remain below 64kb UDP datagram size)

➡ (Compress using gzip just in case)

➡ (Keep a running max stats of the message size. Should remain below 64kb UDP datagram size)

➡ (Transmit events to Lab0 using UDP)

2. Kafka and Nifi



Component State

UpdateAttribute

Description: Gives the option to store values not only on the FlowFile but as stateful variables to be referenced in a recursive manner.

State Deletion Policy: When clearing state this component only supports deleting the entire state.

Filter

Displaying 1 of 1

Key ↑	Value
maxSize	16075

Component State

UpdateAttribute

Description: Gives the option to store values not only on the FlowFile but as stateful variables to be referenced in a recursive manner.

State Deletion Policy: When clearing state this component only supports deleting the entire state.

Filter

Displaying 1 of 1

Key ↑	Value
maxSize	3530

The only use of the UpdateAttribute processors is keeping an eye on the max event size using “View State”. This needs to stay under 64 kb because it won’t fit in one UDP datagram. We could create elaborate contraptions to split here in Lab1 and merge in Lab0 using Nifi but so far we haven’t seen events that warrant that, so we compress instead just to be sure.

2. Kafka and Nifi

1. ConsumeKafka

Processor Details | ConsumeKafka 2.6.0

Settings Scheduling Properties

Required field

Property	Value
Kafka Connection Service	Kafka3ConnectionService
Group ID	nifi1
Topic Format	names
Topics	lab1

Controller Service Details

Settings Properties

Required field

Property	Value
Bootstrap Servers	192.168.37.129:9092
Security Protocol	PLAINTEXT
Transaction Isolation Level	Read Committed
Max Poll Records	10000
Client Timeout	60 sec
Max Metadata Wait Time	5 sec
Acknowledgment Wait Time	5 sec

2. UpdateAttributes

Processor Details | UpdateAttribute 2.6.0

Settings Scheduling Properties

Required field

Property	Value
Delete Attributes Expression	No value set
Store State	Store state locally
Stateful Variables Initial Value	0
Cache Value Lookup Cache Size	100

Advanced

Use original FlowFile for matching rules

Search

bereken_maxSize

fallback_set_maxSize

Conditions

Expression: \${fileSize:toNumber():gt(\${getStateValue('maxSize'):toNumber()})}

Actions

Attribute	Value
maxSize	\${fileSize}

Conditions

Expression: \${fileSize:gt(-1)}

Actions

Attribute	Value
maxSize	\${getStateValue('maxSize')}

2. Kafka and Nifi

3. CompressContent

Processor Details CompressContent 2.6.0		
Settings	Scheduling	Properties
Required field		
Property	Value	
Mode	compress	
Compression Format	gzip	
Compression Level	1	
Update Filename	false	

5. PutUDP

Edit Processor PutUDP 2.6.0		
Settings	Scheduling	Properties
Required field		
Property	Value	
Hostname	192.168.2.69	
Port	12345	
Max Size of Socket Send Buffer	1 MB	
Idle Connection Expiration	15 seconds	
Timeout	10 seconds	

Create a static arp entry aan for IP 192.168.2.69, because arp who-has requests will fail with a data diode and we won't reach the other side:
arp -s 192.168.2.69 00:c1:de:ad:be:ef

Configuration Lab0

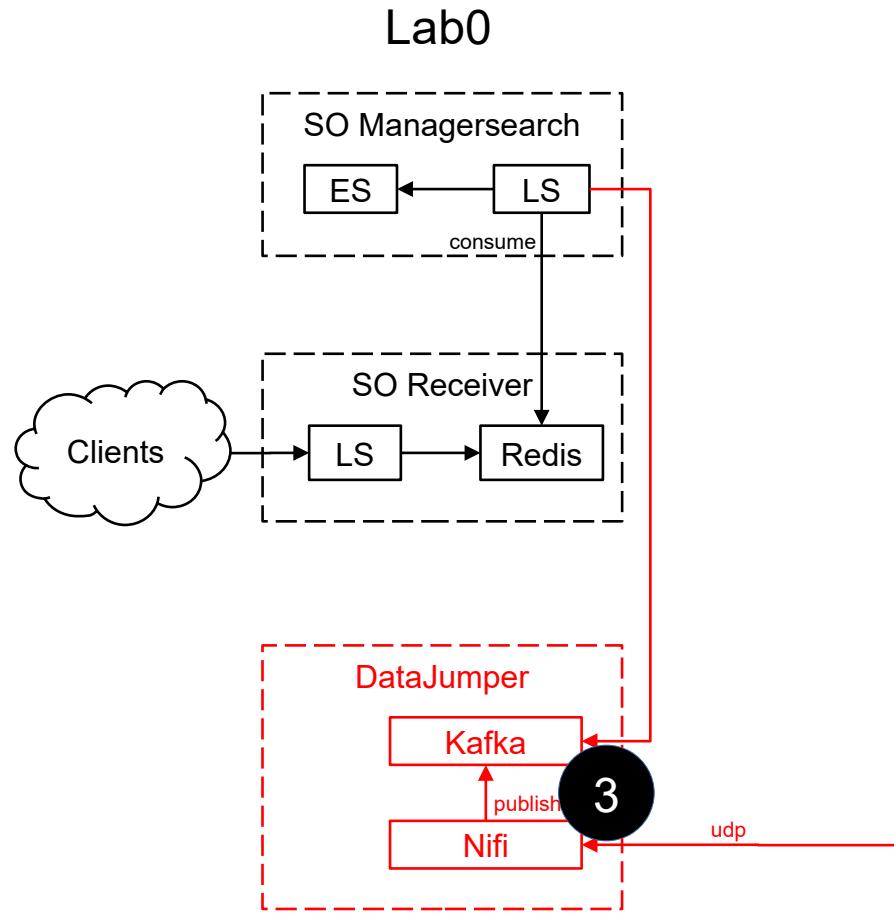
Based on the Open Source
Version of Security Onion

3. Nifi and Kafka

OVERVIEW

Nifi in Lab0:

- receives events via UDP
- publishes to a Kafka topic.



3. Nifi and Kafka

a) In Elastic Fleet change the namespace to lab0.

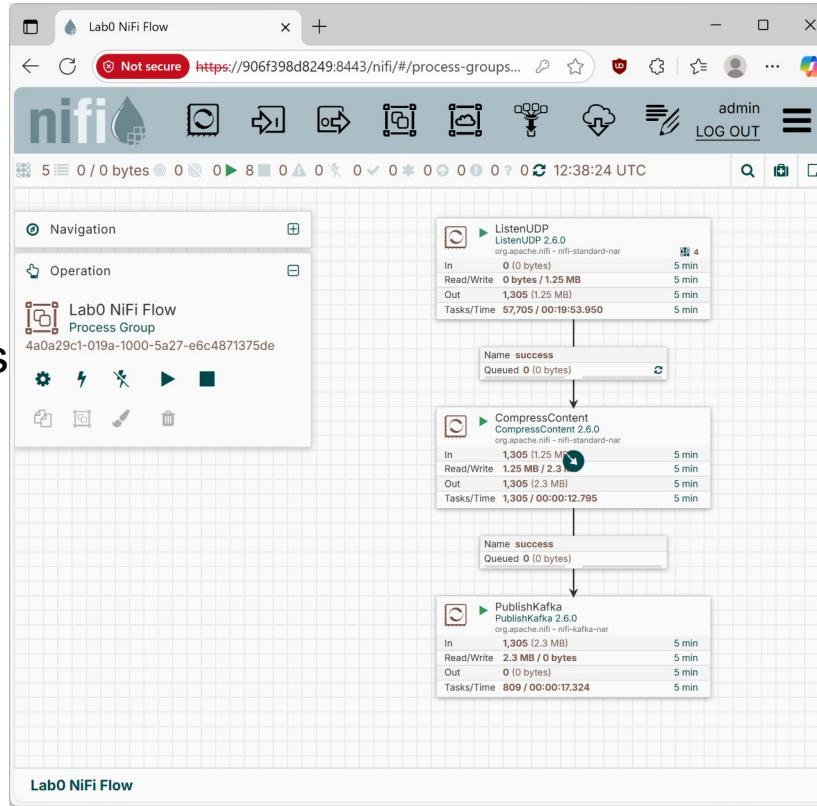
b) Install Nifi and Kafka containers like in the previous step 2.

Add the listening UDP port to the podman run args:

```
-p 12345:12345/udp
```

c) Configure dataflow in Nifi by adding, configuring and linking these processors:

- 1)ListenUDP
- 2)CompressContent
- 3)PublishKafka



◀ (Listen on a UDP port)

◀ (Decompress the events)

◀ (Publish to lab0 Kafka topic)

3. Nifi and Kafka

1. ListenUDP

Processor Details | ListenUDP 2.6.0

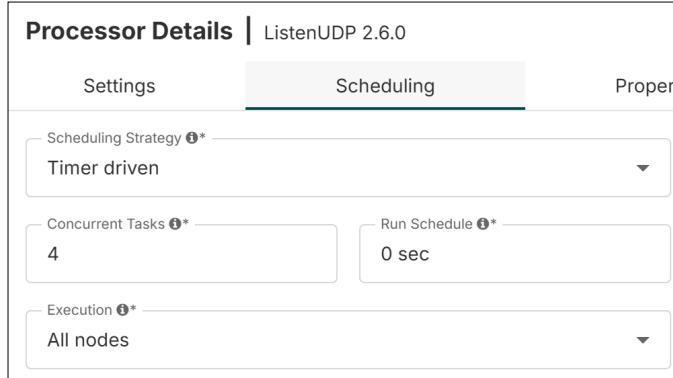
Settings Scheduling Properties

Scheduling Strategy ⓘ* — Timer driven

Concurrent Tasks ⓘ* — 4

Run Schedule ⓘ* — 0 sec

Execution ⓘ* — All nodes

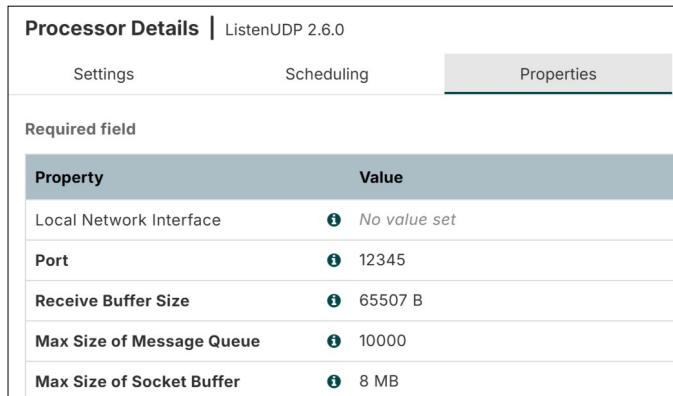


Processor Details | ListenUDP 2.6.0

Settings Scheduling Properties

Required field

Property	Value
Local Network Interface	ⓘ No value set
Port	ⓘ 12345
Receive Buffer Size	ⓘ 65507 B
Max Size of Message Queue	ⓘ 10000
Max Size of Socket Buffer	ⓘ 8 MB



Raise the kernel max receive buffer to prevent packet loss [0]:

```
# sysctl -w net.core.rmem_max=8388608  
net.core.rmem_max = 8388608  
#
```

[0] https://github.com/Vrolijk/OSDD/blob/main/packetloss_explained.md

3. Nifi and Kafka

2. CompressContent

Processor Details CompressContent 2.6.0		
Settings	Scheduling	Properties
Required field		
Property	Value	
Mode	decompress	
Compression Format	gzip	
Update Filename	false	

3. PublishKafka

Processor Details PublishKafka 2.6.0		
Settings	Scheduling	Properties
Required field		
Property	Value	
Kafka Connection Service	Kafka3ConnectionService	
Topic Name	lab1_udp	

↓

Edit Controller Service	
Settings	Properties
Required field	
Property	Value
Bootstrap Servers	192.168.2.69:9092
Security Protocol	PLAINTEXT
Transaction Isolation Level	Read Committed
Max Poll Records	10000
Client Timeout	60 sec
Max Metadata Wait Time	5 sec
Acknowledgment Wait Time	5 sec

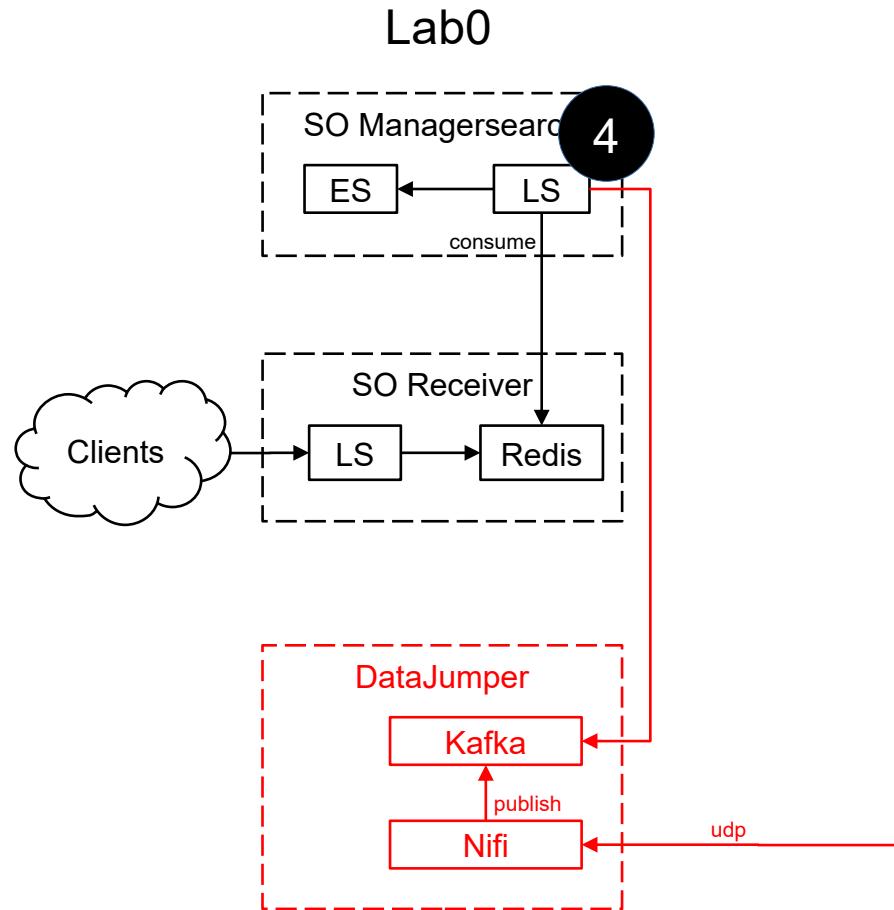
4. Kafka input

OVERVIEW

Add a custom Logstash pipeline that consumes events from the Lab1 Kafka Topic via:

- Security Onion Console
- Manager: /opt/so/saltstack/local/...

In contrast to Lab1, we add it so the Search role pipeline instead of the Receiver pipeline.



4. Kafka input

a) Security Onion Console

The screenshot shows the Security Onion Grid Configuration interface. On the left, the navigation sidebar includes options like Overview, Alerts, Dashboards, Hunt, Cases, Detections, PCAP, Grid, Downloads, Administration, Users, Grid Members, Configuration (which is selected), and License Key. The main area is titled "Grid Configuration" and shows a list of pipeline configurations assigned to the "logstash" group. The "defined_pipelines" section contains several entries: custom0, custom1, custom2, custom3, custom4, fleet, manager, receiver, and a highlighted entry "custom/lab1_udp.conf". A modal window is open over the list, containing buttons for "DUPLICATE" and "VIEW DEFAULT". Below the list, a "Current Grid Value" box displays the paths of the selected files: "so/0900_input_redis.conf.jinja", "so/9805_output_elastic_agent.conf.jinja", "so/9900_output_endgame.conf.jinja", and "custom/lab1_udp.conf". At the bottom of the main area, there is a dropdown menu labeled "Select a node to modify".

b) File on Manager

/opt/so/saltstack/local/salt/logstash/pipelines/config/custom/lab1_udp.conf

```
input {
  kafka {
    bootstrap_servers => "192.168.2.69:9092"
    topics => ["lab1_udp"]
  }
}
```

c) Sync Grid

d) Check on Managersearch via SSH:

```
sudo salt-call pillar.get logstash:defined_pipelines
```

```
[admin@manager ~]$ sudo salt-call pillar.get logstash:defined_pipelines
[sudo] password for admin:
local:
-----
search:
  - so/0900_input_redis.conf.jinja
  - so/9805_output_elastic_agent.conf.jinja
  - so/9900_output_endgame.conf.jinja
  - custom/lab1_udp.conf
```

e) Dont want to wait 15 minutes? Execute on Managersearch:

```
sudo salt-call state.highstate
```

DISCLAIMER

This is a lab-scale proof of concept

- No encryption
- No authentication
- No RBAC
- No persistence
- No redundancy, retry or fail-over
- No scaling