aio CLI to manage CP4AIOps

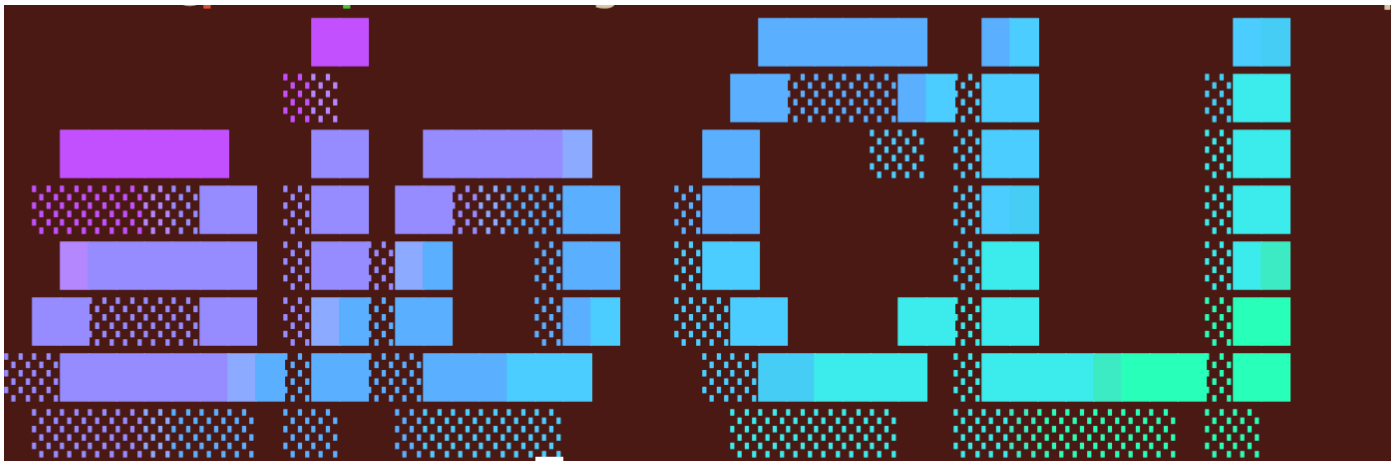


Table of Contents

Note : in these version, I ported aio program to **MacOS**, **Linux** and **Windows**. 64 bits only.

Purpose

The **aio** command line is a small program that helps you to log in, switch between openshift clusters and then **manage** CP4WAIOPS different data (metrics, topologies, events, stories ...). Logs are under implementation. The intend of this command is to provide a command line to help ingeneers to quickly view, ingest or delete data in CP4WAIOPS during tests, PoCs or MVPs.

Installation

Put the aio program into a binary library that is located in your PATH. For example for Linux or MacOS:

```
cp aio-linux-amd64 /usr/local/bin/aio
cd /usr/local/bin
chmod +x aio
```

You also need to have : **oc** CLI installed. Optionnaly **ibmcloud** if you defined a ROKS cluster on IBM Cloud.

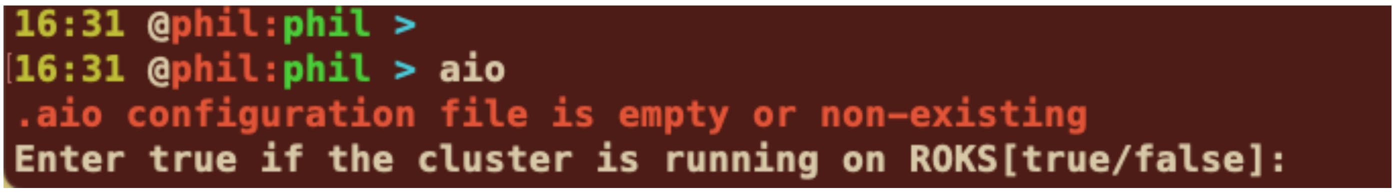
1- Connecting to any OpenShift with aio CLI

When you have to manage multiple ROKS or OCP environments for managing different AI managers, it is not easy to remenbers users, passwords, account keys, API keys ... So I decided to write a small program called **aio** that will make your life better.

Because I was switching from one ROKS or OCP to another one so many times during the day, I felt it was so painful to recall the credentials, I have decided to store all the credentials in a file (called .aio in your home directory)

Also, the **first time,** you will type **aio** on your terminal, the **~/.aio** file is empty. So it will ask you to **add a new OpenShift cluster** set of credentials.

The first step is to specify **true** (ROKS) or **false** (OCP) :

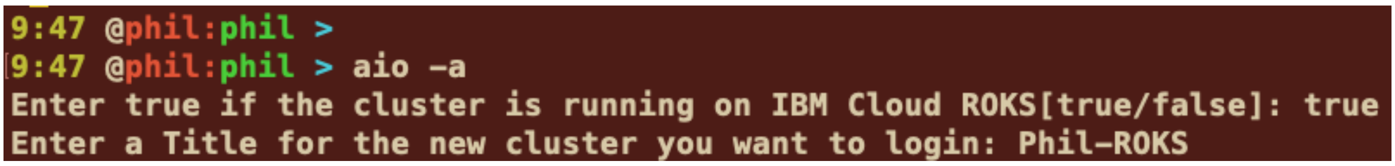


```
16:31 @phil:phil >
16:31 @phil:phil > aio
.aio configuration file is empty or non-existing
Enter true if the cluster is running on ROKS[true/false]:
```

IBM Cloud OpenShift Cluster (ROKS) configuration

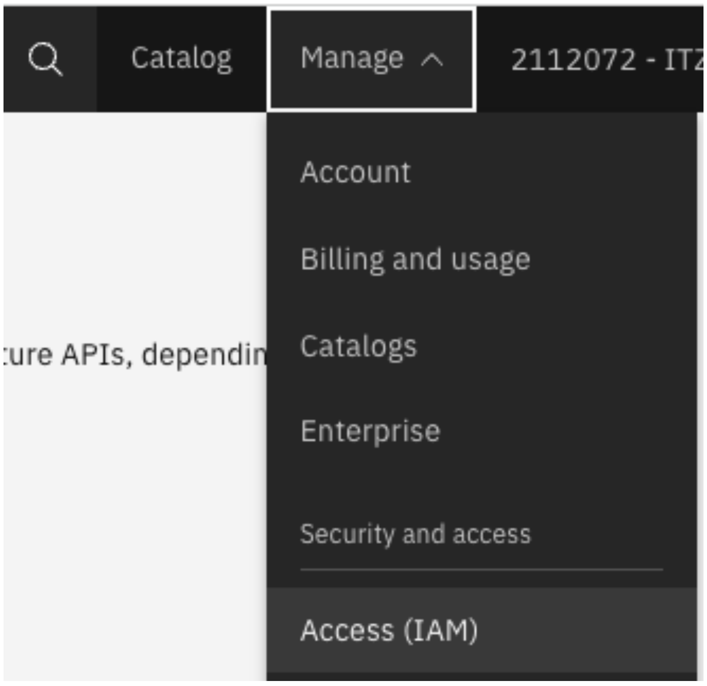
(If you use a standard OCP see next section)

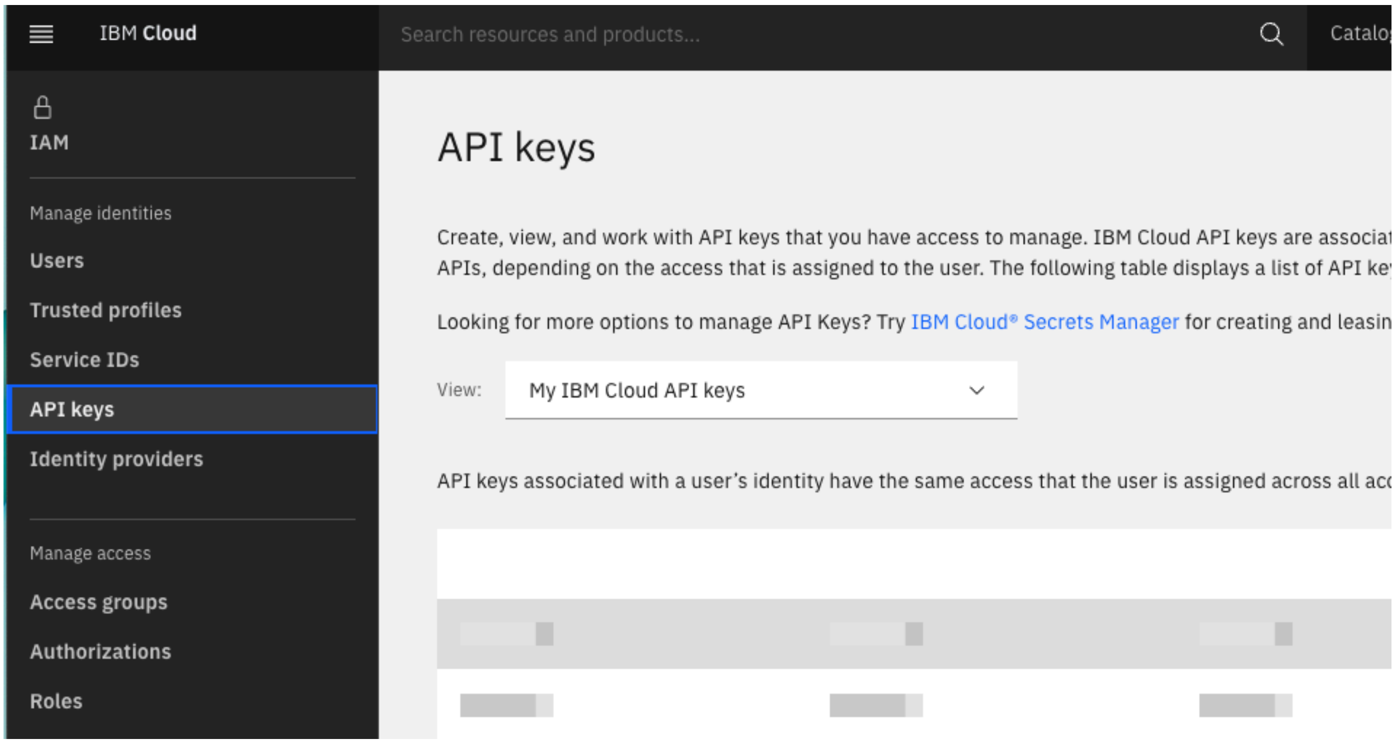
You should specify **true** for example if you have an OpenShift Cluster running on **IBM Cloud** and then the title is just a name that represents the cluster (you can pick **any name that you want**).



```
9:47 @phil:phil >
9:47 @phil:phil > aio -a
Enter true if the cluster is running on IBM Cloud ROKS[true/false]: true
Enter a Title for the new cluster you want to login: Phil-ROKS
```

Then you need to provide the account API Key: you can find this key in the IBM Cloud console. Be sure to select the right target **account** (on the top right) where your cluster is located. If necessary, it is easy to **create a new key** by selecting `Manage > IAM > API Key > Create` . Don't forget to download the key and copy -paste the key when asked. If you already have that key, just provide the existing key.





API keys

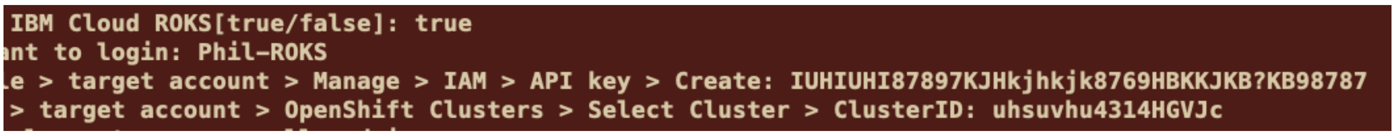
Create, view, and work with API keys that you have access to manage. IBM Cloud API keys are associat
APIs, depending on the access that is assigned to the user. The following table displays a list of API ke

Looking for more options to manage API Keys? Try IBM Cloud® Secrets Manager for creating and leasin

API keys associated with a user's identity have the same access that the user is assigned across all acc

You will need to provide an **Account key** (this key can be generated on IBM Cloud in the Manage > IAM > API key). Put this key when requested.

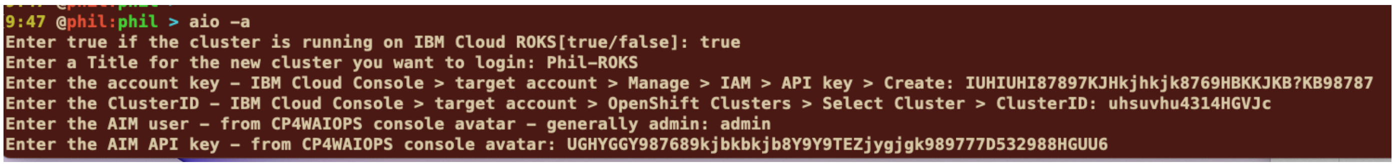
Then you will be asked for the **clusterID** : this ID is provided in the IBM Cloud console, on the OpenShift Cluster of your choice.



```
IBM Cloud ROKS[true/false]: true
nt to login: Phil-ROKS
e > target account > Manage > IAM > API key > Create: IUHIUHI87897KJHkjhkjk8769HBKKJKB?KB98787
> target account > OpenShift Clusters > Select Cluster > ClusterID: uhsuvhu4314HGVJc
```

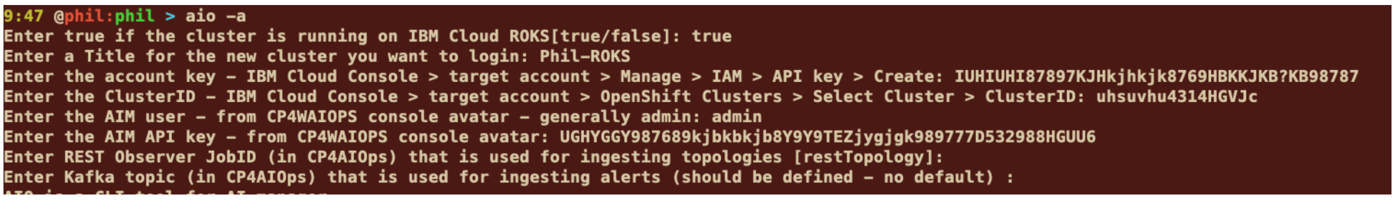
Finally the 2 last parameters are optional if you don't have installed CP4WAIOps and concern the connexion to **AI Manager** (CP4WAIOPS). The **admin** or any other user necessary to connect to AI Manager. `admin` is generally the user's name that you will use. Of course, you can get the key once the AI manager has been created.

You will be asked to provide the AI Manager API key (which is also optional) : to get this key just go to the top right icon (avatar) and click on settings > API Key. You can regenerate a new one if necessary.



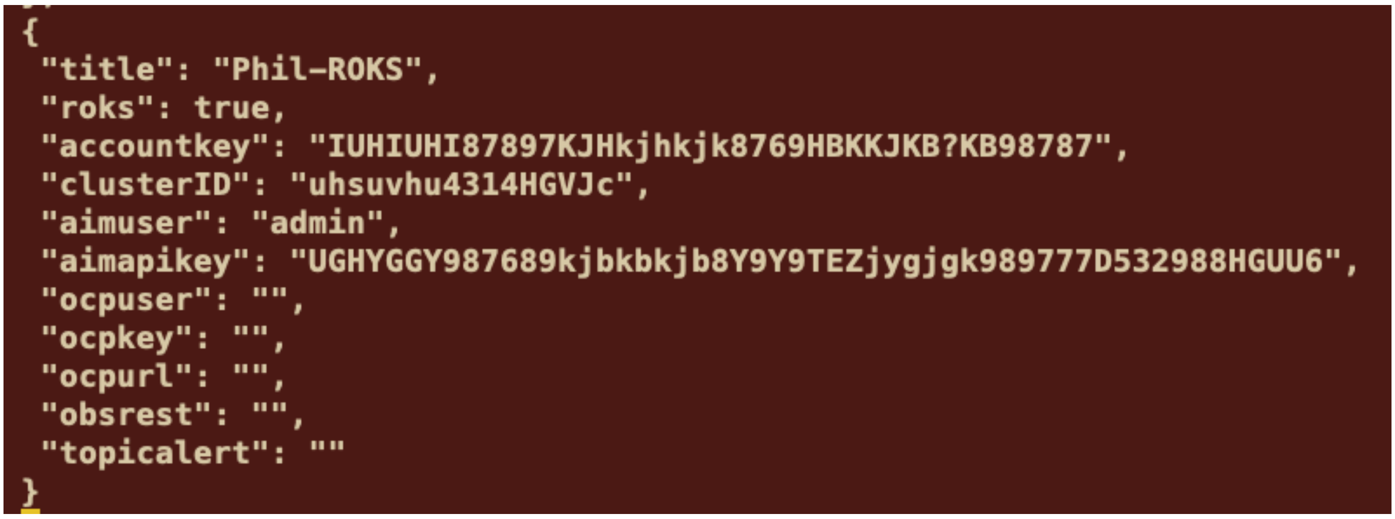
```
9:47 @phil:phil > aio -a
Enter true if the cluster is running on IBM Cloud ROKS[true/false]: true
Enter a Title for the new cluster you want to login: Phil-ROKS
Enter the account key - IBM Cloud Console > target account > Manage > IAM > API key > Create: IUHIUHI87897KJHkjhkjk8769HBKKJKB?KB98787
Enter the ClusterID - IBM Cloud Console > target account > OpenShift Clusters > Select Cluster > ClusterID: uhsuvhu4314HGVJc
Enter the AIM user - from CP4WAIOPS console avatar - generally admin: admin
Enter the AIM API key - from CP4WAIOPS console avatar: UGHYGGY987689kjbkbkjb8Y9Y9TEZjygjgk989777D532988HGUU6
```

Finally fill the jobID and the Topic if you know them at the time of configuration:



```
9:47 @phil:phil > aio -a
Enter true if the cluster is running on IBM Cloud ROKS[true/false]: true
Enter a Title for the new cluster you want to login: Phil-ROKS
Enter the account key - IBM Cloud Console > target account > Manage > IAM > API key > Create: IUHIUHI87897KJHkjhkjk8769HBKKJKB?KB98787
Enter the ClusterID - IBM Cloud Console > target account > OpenShift Clusters > Select Cluster > ClusterID: uhsuvhu4314HGVJc
Enter the AIM user - from CP4WAIOPS console avatar - generally admin: admin
Enter the AIM API key - from CP4WAIOPS console avatar: UGHYGGY987689kjbkbkjb8Y9Y9TEZjygjgk989777D532988HGUU6
Enter REST Observer JobID (in CP4AIOps) that is used for ingesting topologies [restTopology]:
Enter Kafka topic (in CP4AIOps) that is used for ingesting alerts (should be defined - no default) :
```

You can also visualize the **~/.aio** file and you can also modify it directly if needed.

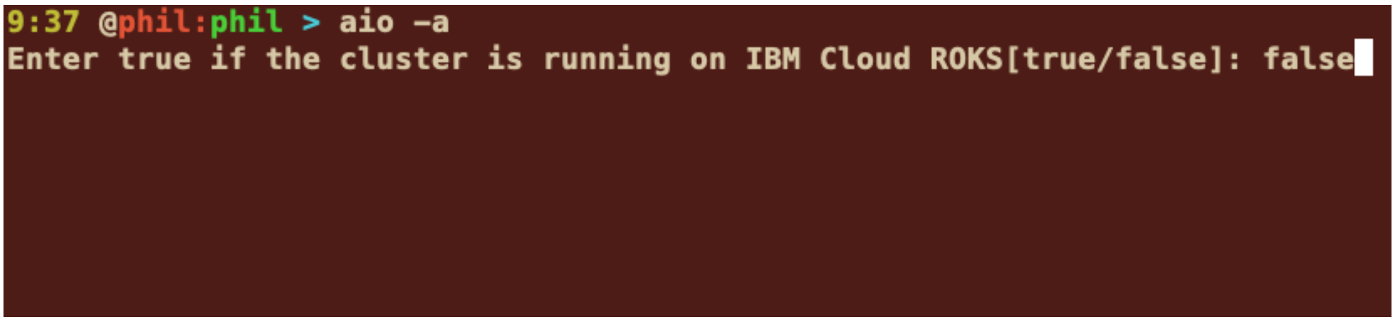


```
{
 "title": "Phil-ROKS",
 "roks": true,
 "accountkey": "IUHIUHI87897KJHkjhkjk8769HBKKJKB?KB98787",
 "clusterID": "uhsuvhu4314HGVJc",
 "aimuser": "admin",
 "aimapikey": "UGHYGGY987689kjbkbkjb8Y9Y9TEZjygjgk989777D532988HGUU6",
 "ocpuser": "",
 "ocpkey": "",
 "ocpurl": "",
 "obsrest": "",
 "topicalert": ""
}
```

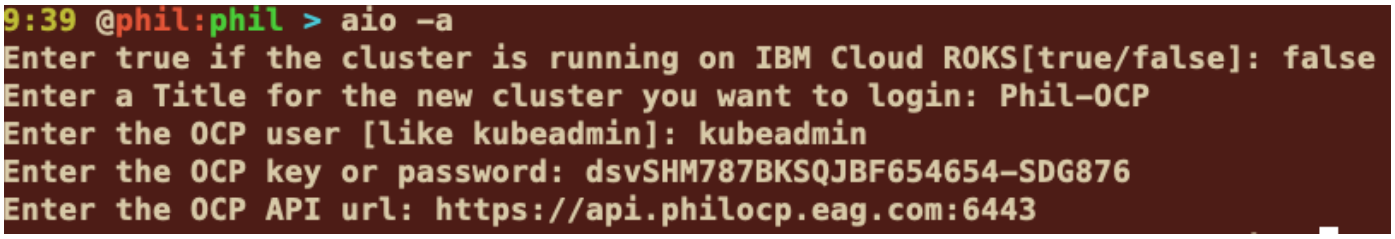
You can also type **aio -l** to list the different parameters.

OCP configuration

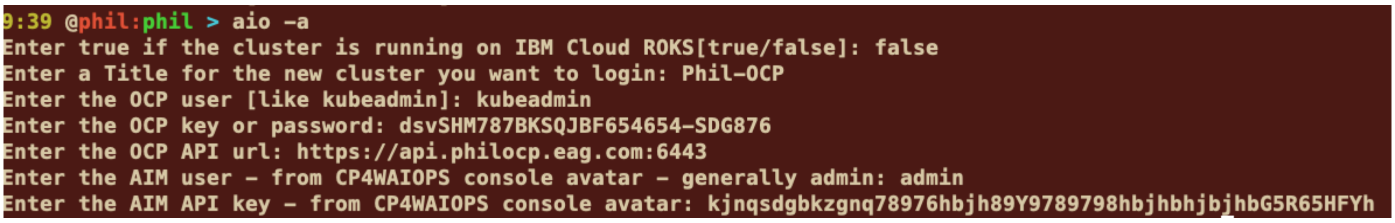
If you are using an **OCP cluster,** instead of ROKS (false), you will use the **ocpuser, ocpkey and ocpurl** parameters instead of **accountkey and clusterID.** Depending on the the **roks=false**, this will enable to collect automatically these parameters.


```
9:37 @phil:phil > aio -a
Enter true if the cluster is running on IBM Cloud ROKS[true/false]: false
```

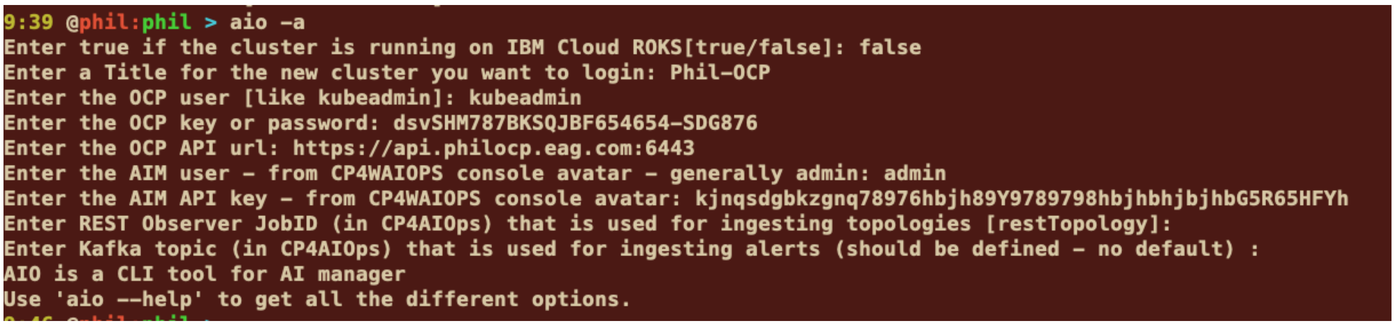
Then provide **title, ocpuser, ocpkey and ocpurl** parameters


```
9:39 @phil:phil > aio -a
Enter true if the cluster is running on IBM Cloud ROKS[true/false]: false
Enter a Title for the new cluster you want to login: Phil-OCP
Enter the OCP user [like kubeadmin]: kubeadmin
Enter the OCP key or password: dsvSHM787BKSQJBF654654-SDG876
Enter the OCP API url: https://api.philocp.eag.com:6443
```

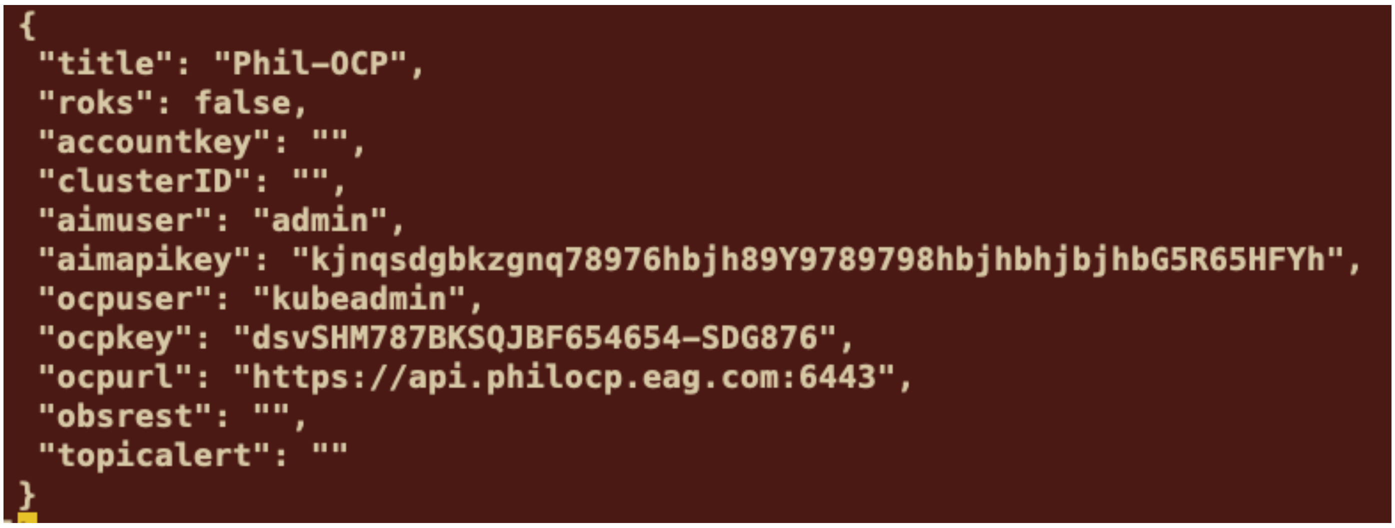
Then the admin and API key for CP4WAIOPS (if known at the time of configuration - otherwise specify none):


```
9:39 @phil:phil > aio -a
Enter true if the cluster is running on IBM Cloud ROKS[true/false]: false
Enter a Title for the new cluster you want to login: Phil-OCP
Enter the OCP user [like kubeadmin]: kubeadmin
Enter the OCP key or password: dsvSHM787BKSQJBF654654-SDG876
Enter the OCP API url: https://api.philocp.eag.com:6443
Enter the AIM user - from CP4WAIOPS console avatar - generally admin: admin
Enter the AIM API key - from CP4WAIOPS console avatar: kjnqsdgbkzgnq78976hbjh89Y9789798hbjhbhjbjhbG5R65HFYh
```

Finally if you know the JobID and Kafka Topic, specify them. Otherwise press enter and specify these names later.


```
9:39 @phil:phil > aio -a
Enter true if the cluster is running on IBM Cloud ROKS[true/false]: false
Enter a Title for the new cluster you want to login: Phil-OCP
Enter the OCP user [like kubeadmin]: kubeadmin
Enter the OCP key or password: dsvSHM787BKSQJBF654654-SDG876
Enter the OCP API url: https://api.philocp.eag.com:6443
Enter the AIM user - from CP4WAIOPS console avatar - generally admin: admin
Enter the AIM API key - from CP4WAIOPS console avatar: kjnqsdgbkzgnq78976hbjh89Y9789798hbjhbhjbjhbG5R65HFYh
Enter REST Observer JobID (in CP4AIOps) that is used for ingesting topologies [restTopology]:
Enter Kafka topic (in CP4AIOps) that is used for ingesting alerts (should be defined - no default) :
AIO is a CLI tool for AI manager
Use 'aio --help' to get all the different options.
```

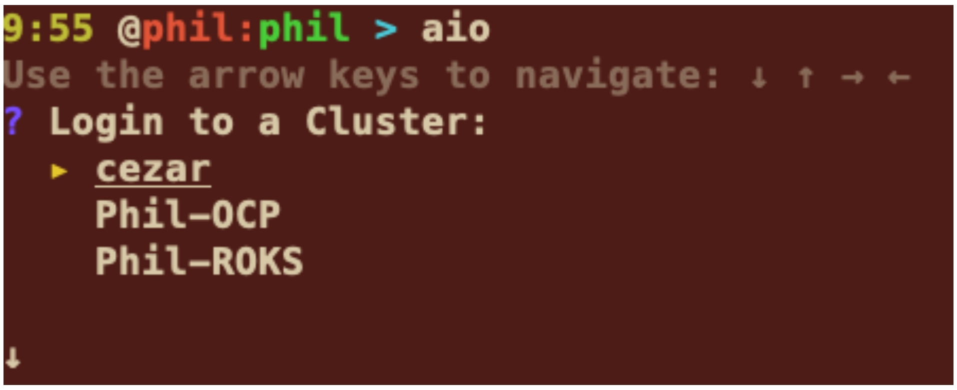
You can also modify / print ~/.aio when you want:


```
{
 "title": "Phil-OCP",
 "roks": false,
 "accountkey": "",
 "clusterID": "",
 "aimuser": "admin",
 "aimapikey": "kjnqsdgbkzgnq78976hbjh89Y9789798hbjhbhjbjhbG5R65HFYh",
 "ocpuser": "kubeadmin",
 "ocpkey": "dsvSHM787BKSQJBF654654-SDG876",
 "ocpurl": "https://api.philocp.eag.com:6443",
 "obsrest": "",
 "topicalert": ""
}
```

Using aio to select and login any OpenShift Cluster

I didn't implemented a delete parameter for aio CLI to delete a specific cluster entry. But you can manually edit .aio.

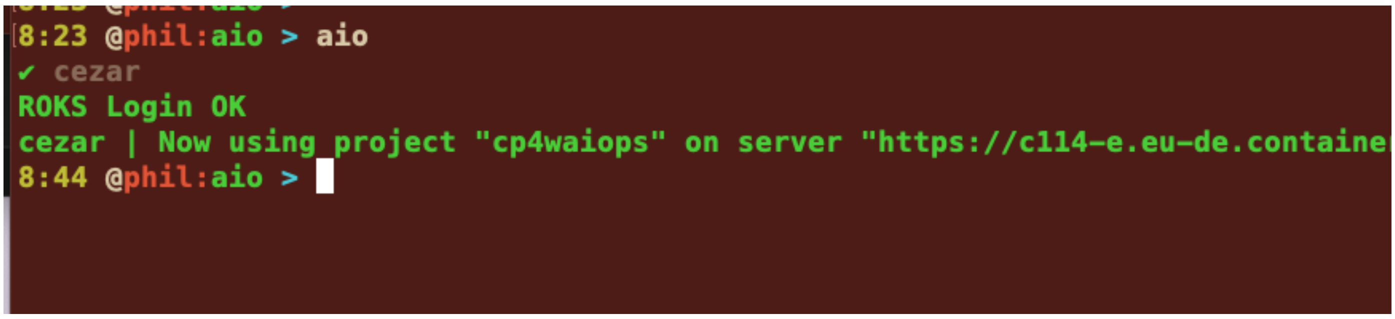
Now let's play with **aio**.When you just type aio, you will get the list of all your registred clusters (ROKS or OCP):



```
9:55 @phil:phil > aio
Use the arrow keys to navigate: ↓ ↑ → ←
? Login to a Cluster:
  ▸ cezar
    Phil-OCP
    Phil-ROKS

↓
```

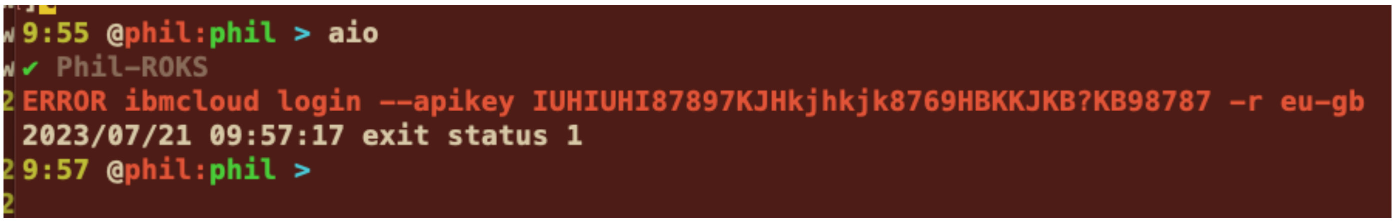
You can just navigate to the Cluster (with cursors and the yellow pointer) that you want and press ENTER. I picked **cezar** for example then be patient, 10 or 20 seconds generally:

If successful, **aio** will display some information:



```
8:23 @phil:aio > aio
✔ cezar
ROKS Login OK
cezar | Now using project "cp4waiops" on server "https://c114-e.eu-de.containe
8:44 @phil:aio > 
```

If not successful, you will receive a **red message**:

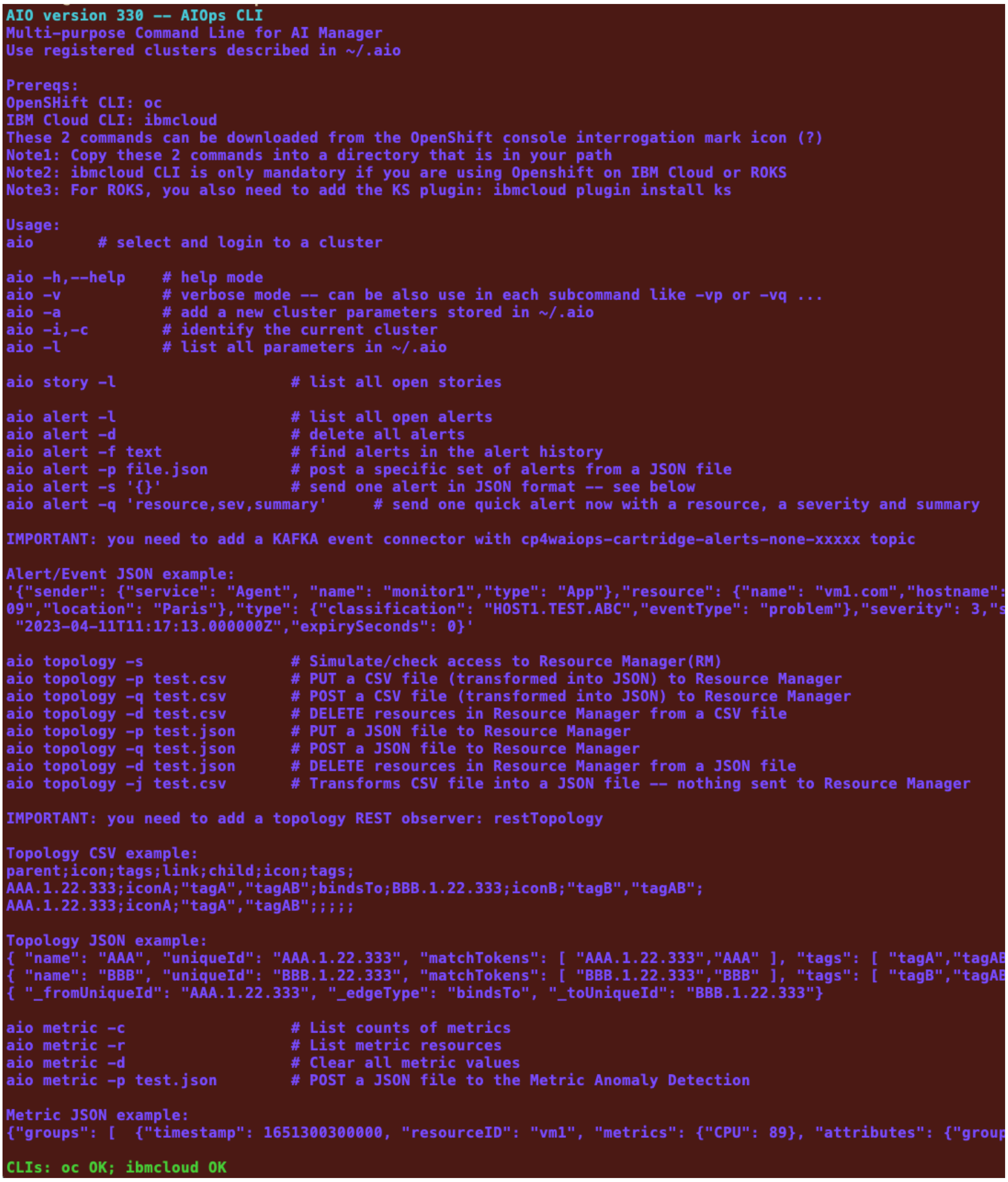


```
9:55 @phil:phil > aio
✔ Phil-ROKS
ERROR ibmcloud login --apikey IUHIUHI87897KJHkjhkjk8769HBKKJKB?KB98787 -r eu-gb
2023/07/21 09:57:17 exit status 1
9:57 @phil:phil >
```

> Remember that aio is using oc CLI and ibmcloud CLI (if using ROKS).

Help

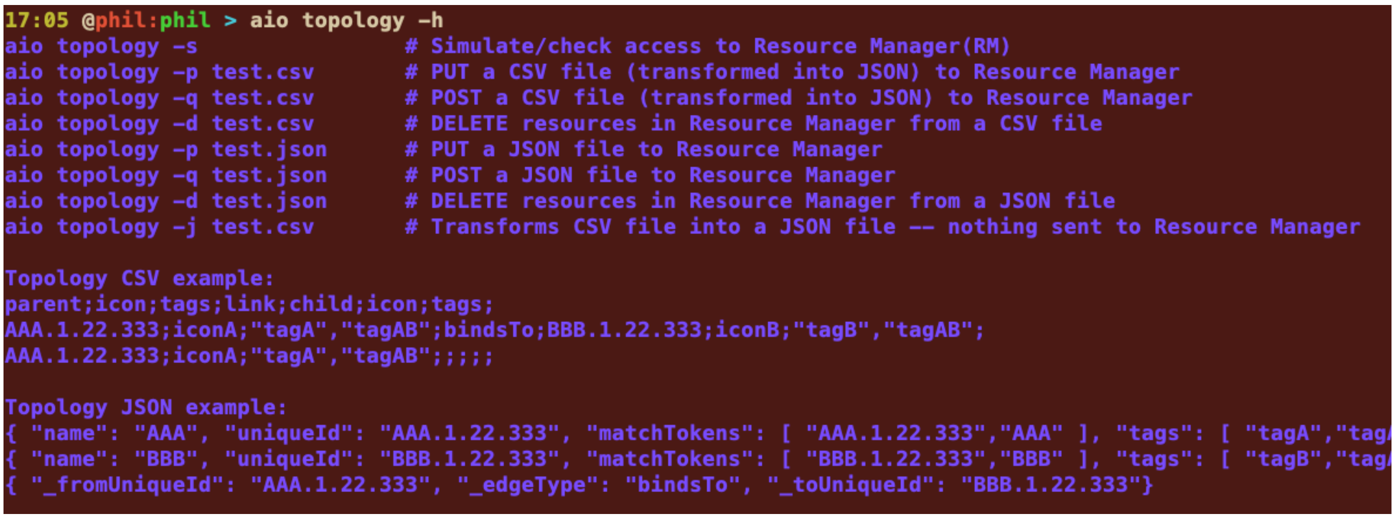
If you have some problems, you can use the help : `aio -h`



```
AIO version 330 -- AIOps CLI
Multi-purpose Command Line for AI Manager
Use registered clusters described in ~/.aio

Prereqs:
OpenSHift CLI: oc
IBM Cloud CLI: ibmcloud
These 2 commands can be downloaded from the OpenShift console interrogation mark icon (?)
Note1: Copy these 2 commands into a directory that is in your path
Note2: ibmcloud CLI is only mandatory if you are using Openshift on IBM Cloud or ROKS
Note3: For ROKS, you also need to add the KS plugin: ibmcloud plugin install ks

Usage:
aio       # select and login to a cluster

aio -h,--help    # help mode
aio -v           # verbose mode -- can be also use in each subcommand like -vp or -vq ...
aio -a           # add a new cluster parameters stored in ~/.aio
aio -i,-c        # identify the current cluster
aio -l           # list all parameters in ~/.aio

aio story -l                   # list all open stories

aio alert -l                   # list all open alerts
aio alert -d                   # delete all alerts
aio alert -f text              # find alerts in the alert history
aio alert -p file.json         # post a specific set of alerts from a JSON file
aio alert -s '{}'              # send one alert in JSON format -- see below
aio alert -q 'resource,sev,summary'     # send one quick alert now with a resource, a severity and summary

IMPORTANT: you need to add a KAFKA event connector with cp4waiops-cartridge-alerts-none-xxxxx topic

Alert/Event JSON example:
'{"sender": {"service": "Agent", "name": "monitor1","type": "App"},"resource": {"name": "vm1.com","hostname":
09","location": "Paris"},"type": {"classification": "HOST1.TEST.ABC","eventType": "problem"},"severity": 3,"s
 "2023-04-11T11:17:13.000000Z","expirySeconds": 0}'

aio topology -s                # Simulate/check access to Resource Manager(RM)
aio topology -p test.csv       # PUT a CSV file (transformed into JSON) to Resource Manager
aio topology -q test.csv       # POST a CSV file (transformed into JSON) to Resource Manager
aio topology -d test.csv       # DELETE resources in Resource Manager from a CSV file
aio topology -p test.json      # PUT a JSON file to Resource Manager
aio topology -q test.json      # POST a JSON file to Resource Manager
aio topology -d test.json      # DELETE resources in Resource Manager from a JSON file
aio topology -j test.csv       # Transforms CSV file into a JSON file -- nothing sent to Resource Manager

IMPORTANT: you need to add a topology REST observer: restTopology

Topology CSV example:
parent;icon;tags;link;child;icon;tags;
AAA.1.22.333;iconA;"tagA","tagAB";bindsTo;BBB.1.22.333;iconB;"tagB","tagAB";
AAA.1.22.333;iconA;"tagA","tagAB";;;;;

Topology JSON example:
{ "name": "AAA", "uniqueId": "AAA.1.22.333", "matchTokens": [ "AAA.1.22.333","AAA" ], "tags": [ "tagA","tagAB
{ "name": "BBB", "uniqueId": "BBB.1.22.333", "matchTokens": [ "BBB.1.22.333","BBB" ], "tags": [ "tagB","tagAB
{ "_fromUniqueId": "AAA.1.22.333", "_edgeType": "bindsTo", "_toUniqueId": "BBB.1.22.333"}

aio metric -c                  # List counts of metrics
aio metric -r                  # List metric resources
aio metric -d                  # Clear all metric values
aio metric -p test.json        # POST a JSON file to the Metric Anomaly Detection

Metric JSON example:
{"groups": [  {"timestamp": 1651300300000, "resourceID": "vm1", "metrics": {"CPU": 89}, "attributes": {"group

CLIs: oc OK; ibmcloud OK
```

Subcommand help

You can also ask for some help on specific topic like **topology** or **topo** by using `aio topo -h`



```
17:05 @phil:phil > aio topology -h
aio topology -s               # Simulate/check access to Resource Manager(RM)
aio topology -p test.csv      # PUT a CSV file (transformed into JSON) to Resource Manager
aio topology -q test.csv      # POST a CSV file (transformed into JSON) to Resource Manager
aio topology -d test.csv      # DELETE resources in Resource Manager from a CSV file
aio topology -p test.json     # PUT a JSON file to Resource Manager
aio topology -q test.json     # POST a JSON file to Resource Manager
aio topology -d test.json     # DELETE resources in Resource Manager from a JSON file
aio topology -j test.csv      # Transforms CSV file into a JSON file -- nothing sent to Resource Manager

Topology CSV example:
parent;icon;tags;link;child;icon;tags;
AAA.1.22.333;iconA;"tagA","tagAB";bindsTo;BBB.1.22.333;iconB;"tagB","tagAB";
AAA.1.22.333;iconA;"tagA","tagAB";;;;;

Topology JSON example:
{ "name": "AAA", "uniqueId": "AAA.1.22.333", "matchTokens": [ "AAA.1.22.333","AAA" ], "tags": [ "tagA","tag
{ "name": "BBB", "uniqueId": "BBB.1.22.333", "matchTokens": [ "BBB.1.22.333","BBB" ], "tags": [ "tagB","tag
{ "_fromUniqueId": "AAA.1.22.333", "_edgeType": "bindsTo", "_toUniqueId": "BBB.1.22.333"}
```

For instance, the **verbose** mode is very useful to find out a **connexion problem**. Verbose mode is used in conjonction with a subcommand and another flag (like `aio metric -vp` or `aio topo -vd` for example).

Shortcuts

Each subcommand has also shortcuts like :

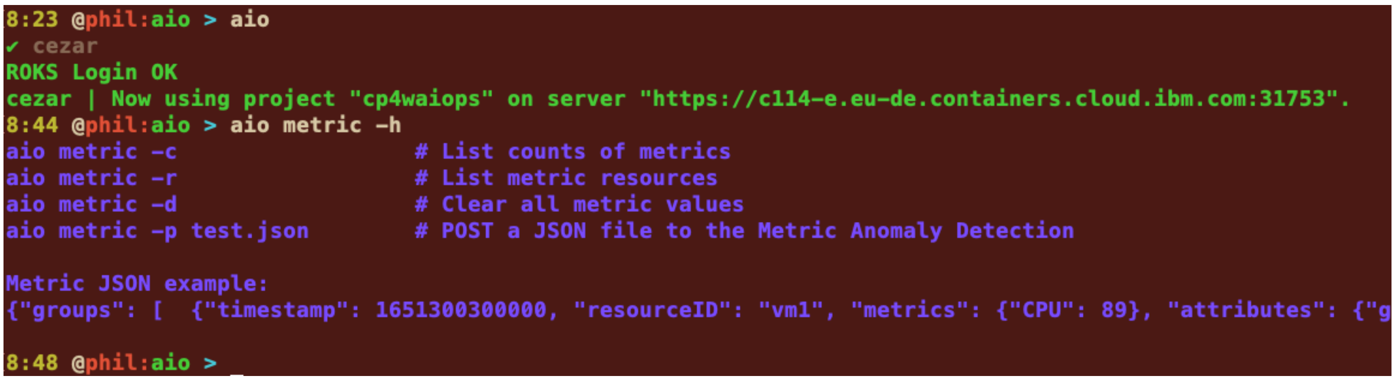
- alerts, alerts, a
- metrics, metric, m
- topologies, topology, topo, t
- stories, story, s

2- Ingesting metrics

When you want to ingest some metrics to the Metric Manager in CP4AIOps, you need to take care about a lot of informations like credentials, REST APIs, routes, password ...

So to avoid all these problems, you can use **aio metric** subcommand that encapsulates all this complexity into the aio CLI.

You can start with `aio metric -h` to see the program help. Be sure to be connected.



```
8:23 @phil:aio > aio
✔ cezar
ROKS Login OK
cezar | Now using project "cp4waiops" on server "https://c114-e.eu-de.containers.cloud.ibm.com:31753".
8:44 @phil:aio > aio metric -h
aio metric -c                  # List counts of metrics
aio metric -r                  # List metric resources
aio metric -d                  # Clear all metric values
aio metric -p test.json        # POST a JSON file to the Metric Anomaly Detection

Metric JSON example:
{"groups": [  {"timestamp": 1651300300000, "resourceID": "vm1", "metrics": {"CPU": 89}, "attributes": {"g

8:48 @phil:aio > _
```

The current thing that we are doing, we can send a json file to the metric manager. This JSON file contains all the metric definitions that you need to ingest. In our case, we look at `OutBoundBytes` that corresponds to a network interface metric.

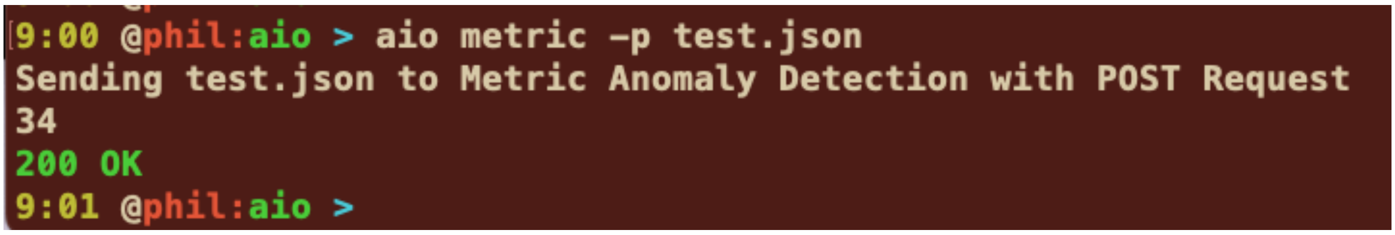
Prepare a file call test.json with the following content:

```
{
  "groups": [
    {
      "timestamp": 1651300300000,
      "resourceID": "VM000101",
      "metrics": { "OutBoundBytes": 10089
        },
      "attributes": {
        "group": "TEST",
        "node": "VM000101",
        "interface": "Interface101"
       }
    },
    {
      "timestamp": 1651300600000,
      "resourceID": "VM000101",
      "metrics": { "OutBoundBytes": 10197
        },
      "attributes": {
        "group": "TEST",
        "node": "VM000101",
        "interface": "Interface101"
       }
    }
  ]
}
```

This JSON file will create 2 occurences of one metric for the same resource `VM000101`.

Now, you can use the following command to ingest the JSON content into Metric Manager:

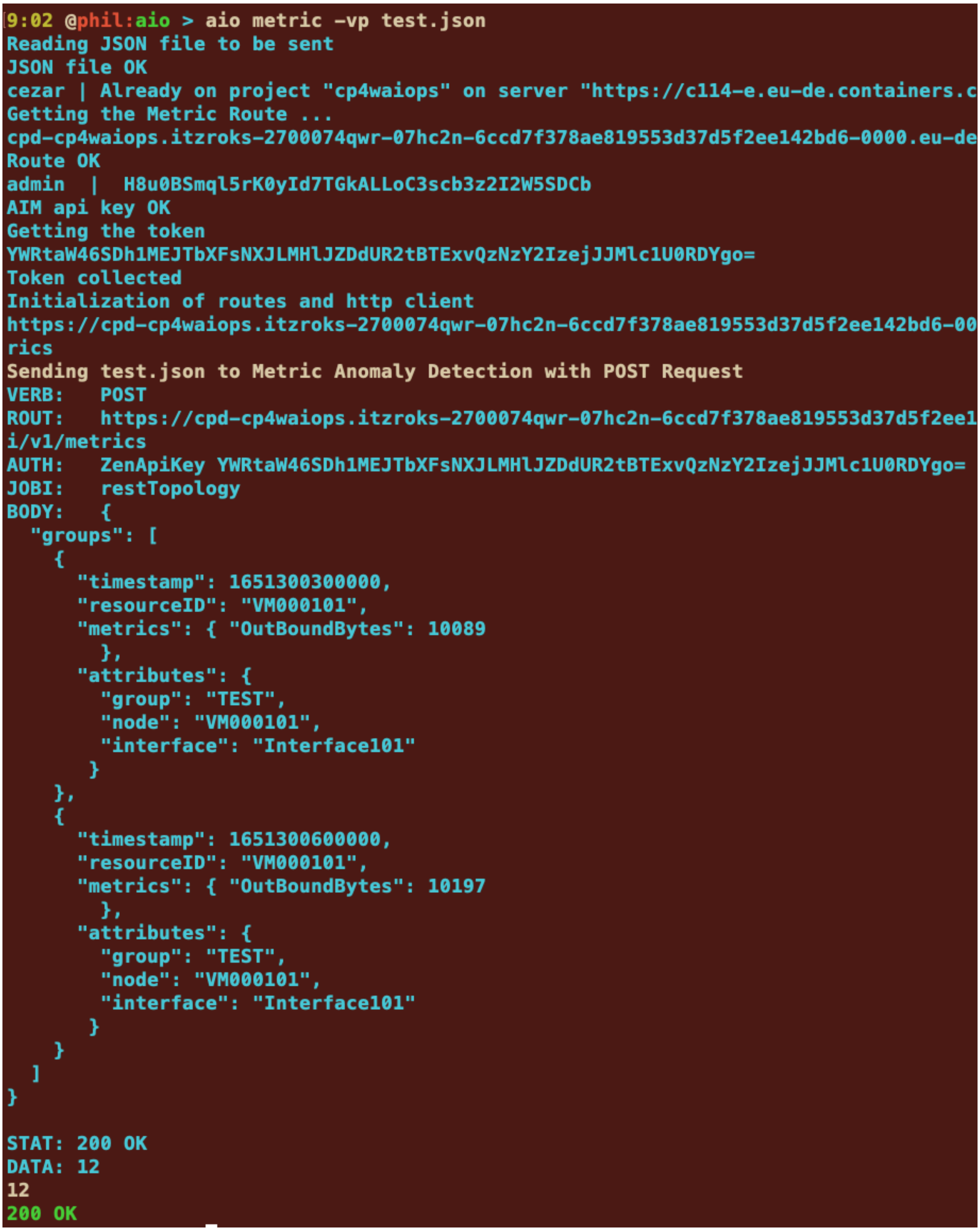
```
aio metric -f test.json
```



```
9:00 @phil:aio > aio metric -p test.json
Sending test.json to Metric Anomaly Detection with POST Request
34
200 OK
9:01 @phil:aio >
```

In case of error, you can use the verbose mode to look at the details by specifying the **-v** in the command line. The result will show all the different steps (token, routes, API ...):

```
aio metric -vp test.json
```



```
9:02 @phil:aio > aio metric -vp test.json
Reading JSON file to be sent
JSON file OK
cezar | Already on project "cp4waiops" on server "https://c114-e.eu-de.containers.c
Getting the Metric Route ...
cpd-cp4waiops.itzroks-2700074qwr-07hc2n-6ccd7f378ae819553d37d5f2ee142bd6-0000.eu-de
Route OK
admin  |  H8u0BSmql5rK0yId7TGkALLoC3scb3z2I2W5SDCb
AIM api key OK
Getting the token
YWRtaW46SDh1MEJTbXFsNXJLMHlJZDdUR2tBTExvQzNzY2IzejJJMlc1U0RDYgo=
Token collected
Initialization of routes and http client
https://cpd-cp4waiops.itzroks-2700074qwr-07hc2n-6ccd7f378ae819553d37d5f2ee142bd6-00
rics
Sending test.json to Metric Anomaly Detection with POST Request
VERB:   POST
ROUT:   https://cpd-cp4waiops.itzroks-2700074qwr-07hc2n-6ccd7f378ae819553d37d5f2ee1
i/v1/metrics
AUTH:   ZenApiKey YWRtaW46SDh1MEJTbXFsNXJLMHlJZDdUR2tBTExvQzNzY2IzejJJMlc1U0RDYgo=
JOBI:   restTopology
BODY:   {
  "groups": [
    {
      "timestamp": 1651300300000,
      "resourceID": "VM000101",
      "metrics": { "OutBoundBytes": 10089
        },
      "attributes": {
        "group": "TEST",
        "node": "VM000101",
        "interface": "Interface101"
       }
    },
    {
      "timestamp": 1651300600000,
      "resourceID": "VM000101",
      "metrics": { "OutBoundBytes": 10197
        },
      "attributes": {
        "group": "TEST",
        "node": "VM000101",
        "interface": "Interface101"
       }
    }
  ]
}

STAT: 200 OK
DATA: 12
12
200 OK
```

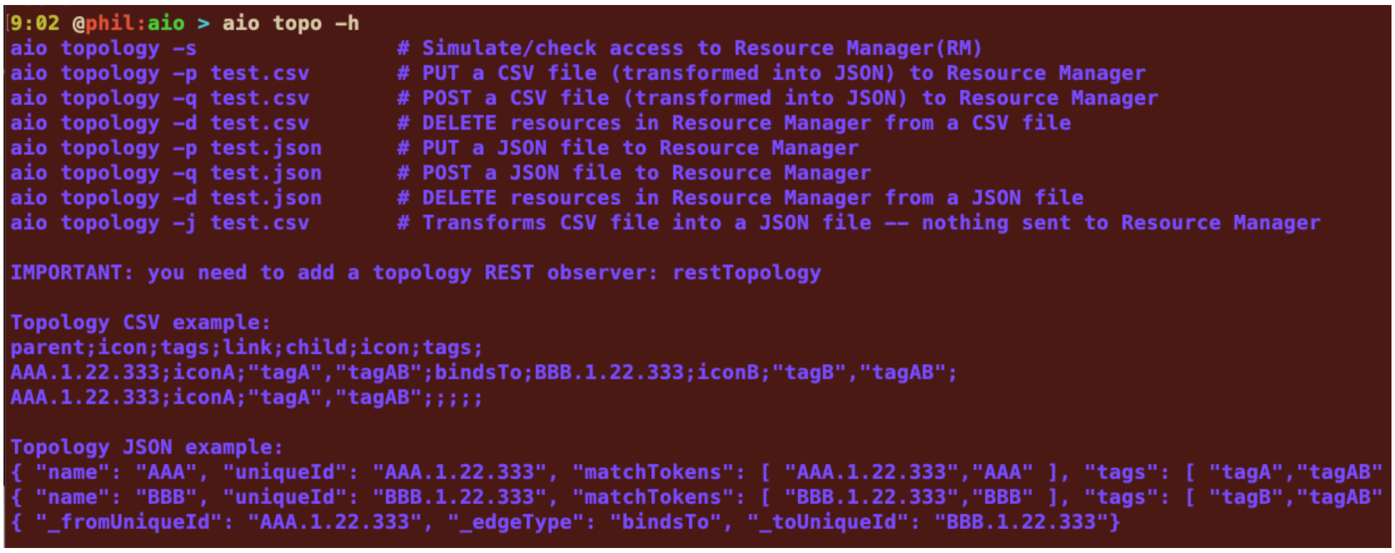
3- Ingesting and managing topologies

When managing topologies with AI Manager, you also have to provide API keys and routes and REST APIs.

So to avoid this problem, you can use `aio topology` which help you to connect and send (and manage) REST API requests to the AI Manager in the Topology manager.

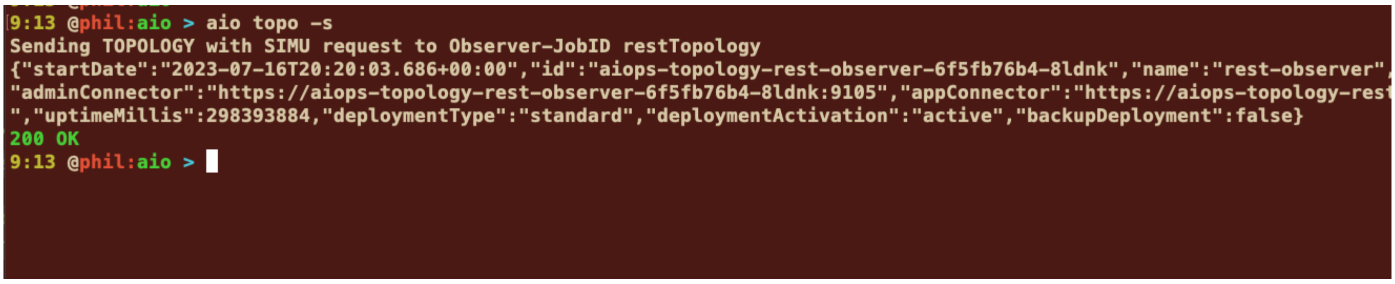
> **Important** you must use **aio** to be connected to the cluster. Also provide the **AIM user and API key** with correct values.

Use `aio topology -h` or `aio topo -h` to see the help for this feature:



```
9:02 @phil:aio > aio topo -h
aio topology -s               # Simulate/check access to Resource Manager(RM)
aio topology -p test.csv      # PUT a CSV file (transformed into JSON) to Resource Manager
aio topology -q test.csv      # POST a CSV file (transformed into JSON) to Resource Manager
aio topology -d test.csv      # DELETE resources in Resource Manager from a CSV file
aio topology -p test.json     # PUT a JSON file to Resource Manager
aio topology -q test.json     # POST a JSON file to Resource Manager
aio topology -d test.json     # DELETE resources in Resource Manager from a JSON file
aio topology -j test.csv      # Transforms CSV file into a JSON file -- nothing sent to Resource Manager

IMPORTANT: you need to add a topology REST observer: restTopology

Topology CSV example:
parent;icon;tags;link;child;icon;tags;
AAA.1.22.333;iconA;"tagA","tagAB";bindsTo;BBB.1.22.333;iconB;"tagB","tagAB";
AAA.1.22.333;iconA;"tagA","tagAB";;;;;

Topology JSON example:
{ "name": "AAA", "uniqueId": "AAA.1.22.333", "matchTokens": [ "AAA.1.22.333","AAA" ], "tags": [ "tagA","tagAB"
{ "name": "BBB", "uniqueId": "BBB.1.22.333", "matchTokens": [ "BBB.1.22.333","BBB" ], "tags": [ "tagB","tagAB"
{ "_fromUniqueId": "AAA.1.22.333", "_edgeType": "bindsTo", "_toUniqueId": "BBB.1.22.333"}
```

The first thing that we want to do with aio topo is to check that the routes and rest observers have been well defined on the AI manager. Use `aio topo -s` to check the **info service is OK** or not. See below a good example where you didn't get a connection with the cluster :



```
9:13 @phil:aio > aio topo -s
Sending TOPOLOGY with SIMU request to Observer-JobID restTopology
{"startDate":"2023-07-16T20:20:03.686+00:00","id":"aiops-topology-rest-observer-6f5fb76b4-8ldnk","name":"rest-observer",
"adminConnector":"https://aiops-topology-rest-observer-6f5fb76b4-8ldnk:9105","appConnector":"https://aiops-topology-rest
","uptimeMillis":298393884,"deploymentType":"standard","deploymentActivation":"active","backupDeployment":false}
200 OK
9:13 @phil:aio > 
```

> **IMPORTANT**: to ingest topologies with the REST API, we need to define a REST Observer in the Cloud Pak. The default name for this Observer is `restTopology` but you can also change that name in the aio configuration file.

Once you checked that the cluster and observers can be accessed with good credentials, you can use this program to ingest new topologies: there are 2 different ways : thru a JSON file or thru a csv file.

Let's take a basic csv file containing (ab.csv): only one line but you can add several lines.

```
A.com.RHEL4;router;"INVOICES","front";connectedTo;B.com.RHEL5;server;"INVOICES","app";
```

A.com.RHEL4: is the parent member, this long name will be the unique ID. A will be the **displayed** name (first part before the dot. If no dot in the file, the displayed name is the same as the unique ID).

router: is the icon used for the A

bindsTo: is the relationship used between A and B

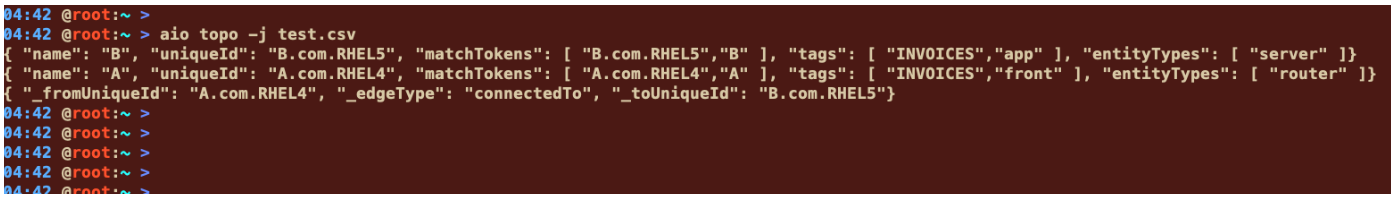
server: is the icon used for B

The other parameters are tags for each resource. You can have one or multiple tags separated by commas.

Before sending that content, **aio** will translate that into a JSON content that will be used to be sent to the Topology manager.

To see the JSON content, type the following command:

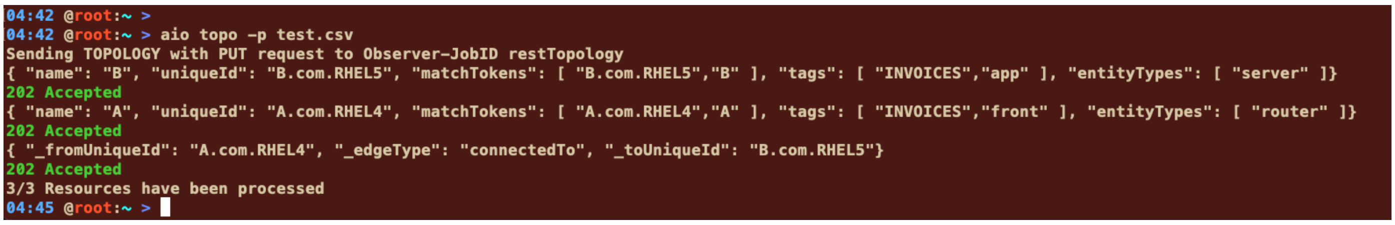
```
aio topo -j test.csv
```



```
04:42 @root:~ >
04:42 @root:~ > aio topo -j test.csv
{ "name": "B", "uniqueId": "B.com.RHEL5", "matchTokens": [ "B.com.RHEL5","B" ], "tags": [ "INVOICES","app" ], "entityTypes": [ "server" ]}
{ "name": "A", "uniqueId": "A.com.RHEL4", "matchTokens": [ "A.com.RHEL4","A" ], "tags": [ "INVOICES","front" ], "entityTypes": [ "router" ]}
{ "_fromUniqueId": "A.com.RHEL4", "_edgeType": "connectedTo", "_toUniqueId": "B.com.RHEL5"}
04:42 @root:~ >
04:42 @root:~ >
04:42 @root:~ >
04:42 @root:~ >
04:42 @root:~ >
```

Now let's send this small topology to AI Manager:

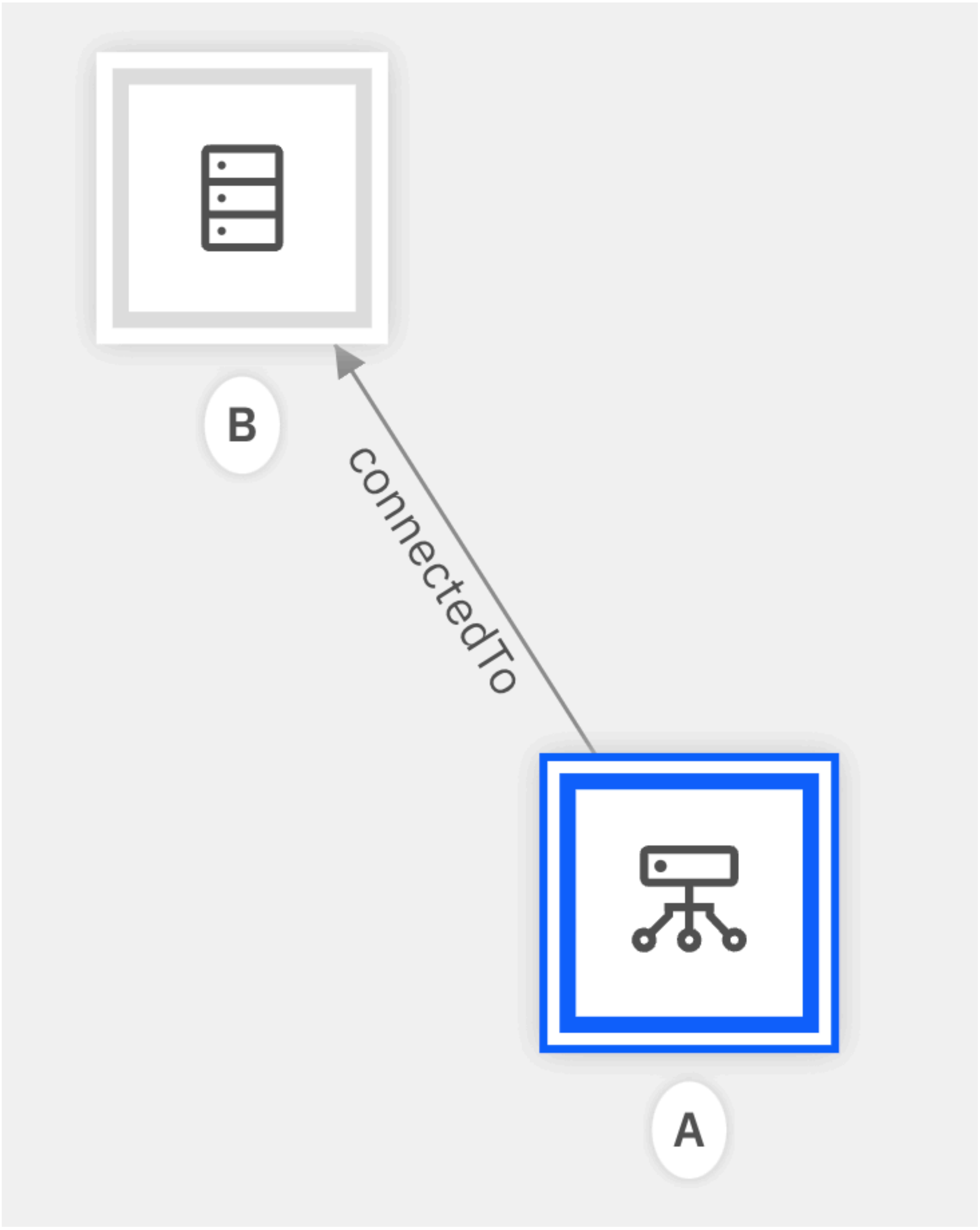
```
aio topo -p test.csv
```



```
04:42 @root:~ >
04:42 @root:~ > aio topo -p test.csv
Sending TOPOLOGY with PUT request to Observer-JobID restTopology
{ "name": "B", "uniqueId": "B.com.RHEL5", "matchTokens": [ "B.com.RHEL5","B" ], "tags": [ "INVOICES","app" ], "entityTypes": [ "server" ]}
202 Accepted
{ "name": "A", "uniqueId": "A.com.RHEL4", "matchTokens": [ "A.com.RHEL4","A" ], "tags": [ "INVOICES","front" ], "entityTypes": [ "router" ]}
202 Accepted
{ "_fromUniqueId": "A.com.RHEL4", "_edgeType": "connectedTo", "_toUniqueId": "B.com.RHEL5"}
202 Accepted
3/3 Resources have been processed
04:45 @root:~ >
```

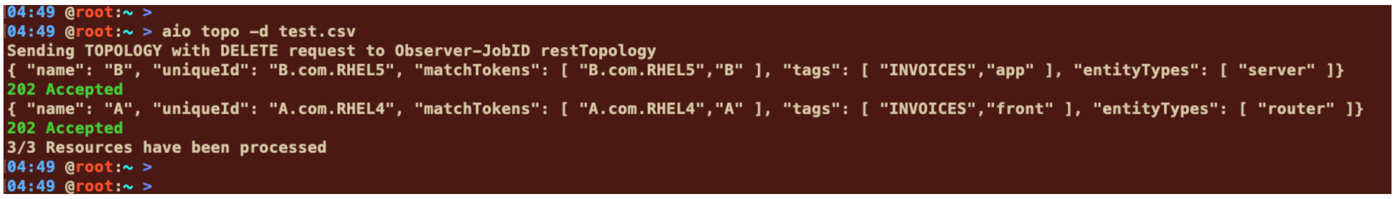
We created 3 resources (A and B and a connection) connected together with a **connectedTo** relationship. The **202 Accepted** return code is normally expected.

Go to the AI manager console and look at Resource Management:



You can also delete the following resources with the following command:

```
aio topo -d test.csv
```



```
04:49 @root:~ >
04:49 @root:~ > aio topo -d test.csv
Sending TOPOLOGY with DELETE request to Observer-JobID restTopology
{ "name": "B", "uniqueId": "B.com.RHEL5", "matchTokens": [ "B.com.RHEL5","B" ], "tags": [ "INVOICES","app" ], "entityTypes": [ "server" ]}
202 Accepted
{ "name": "A", "uniqueId": "A.com.RHEL4", "matchTokens": [ "A.com.RHEL4","A" ], "tags": [ "INVOICES","front" ], "entityTypes": [ "router" ]}
202 Accepted
3/3 Resources have been processed
04:49 @root:~ >
04:49 @root:~ >
```

> **In case of error**: if you received a red message, you can use the verbose mode (**-vp** or **-vd**) to check what is going wrong.
>
> The connection (or edge or relationship) is automatically destroyed when the 2 resources have been suppressed.

You can also ingest directly JSON file. Lets create a test.json by using the `-j` parameter:

```
aio topo -j test.csv > test.json
```

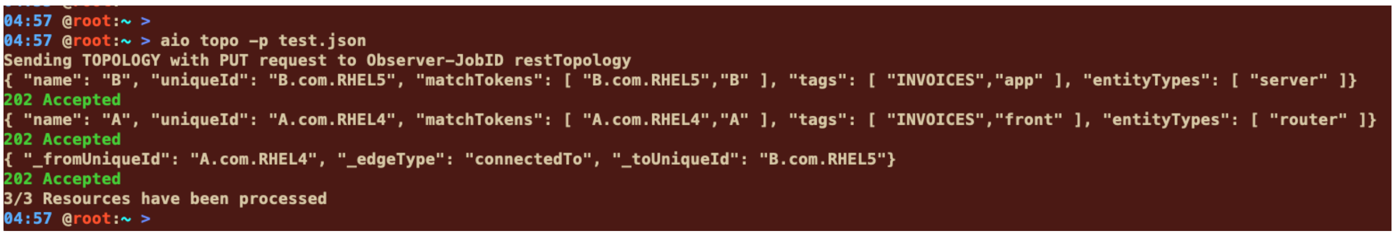
This will generate a file with the following content:

```
{ "name": "B", "uniqueId": "B.com.RHEL5", "matchTokens": [ "B.com.RHEL5","B" ], "tags": [ "INVOICES","app" ], "entityTypes": [ "server" ]}
{ "name": "A", "uniqueId": "A.com.RHEL4", "matchTokens": [ "A.com.RHEL4","A" ], "tags": [ "INVOICES","front" ], "entityTypes": [ "router" ]}
{ "_fromUniqueId": "A.com.RHEL4", "_edgeType": "connectedTo", "_toUniqueId": "B.com.RHEL5"}
```

This content is re-creating the 2 servers (A and B) and a connection between them.

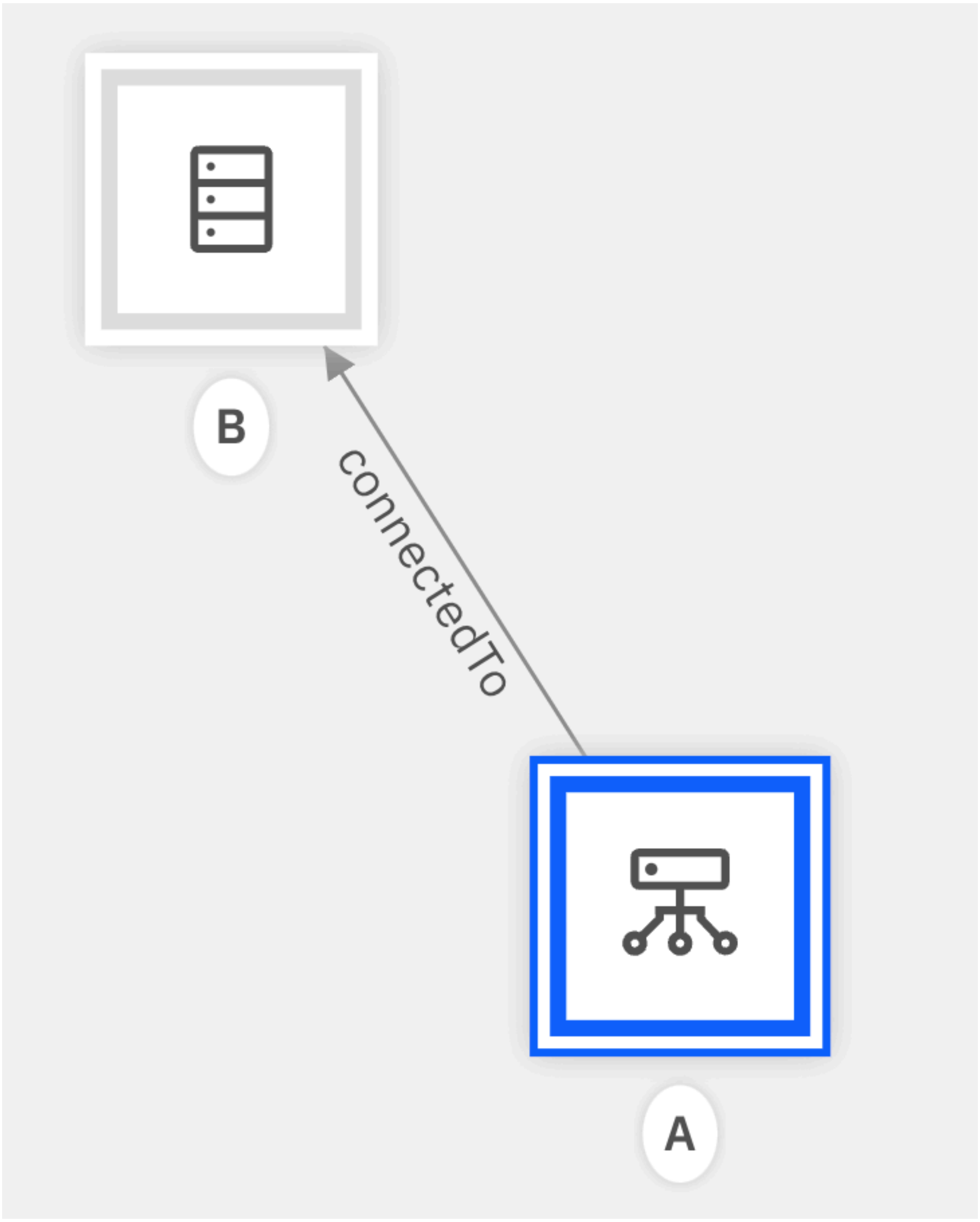
Use the following command to send that file to the Topology Manager:

```
aio topo -p test.json
```



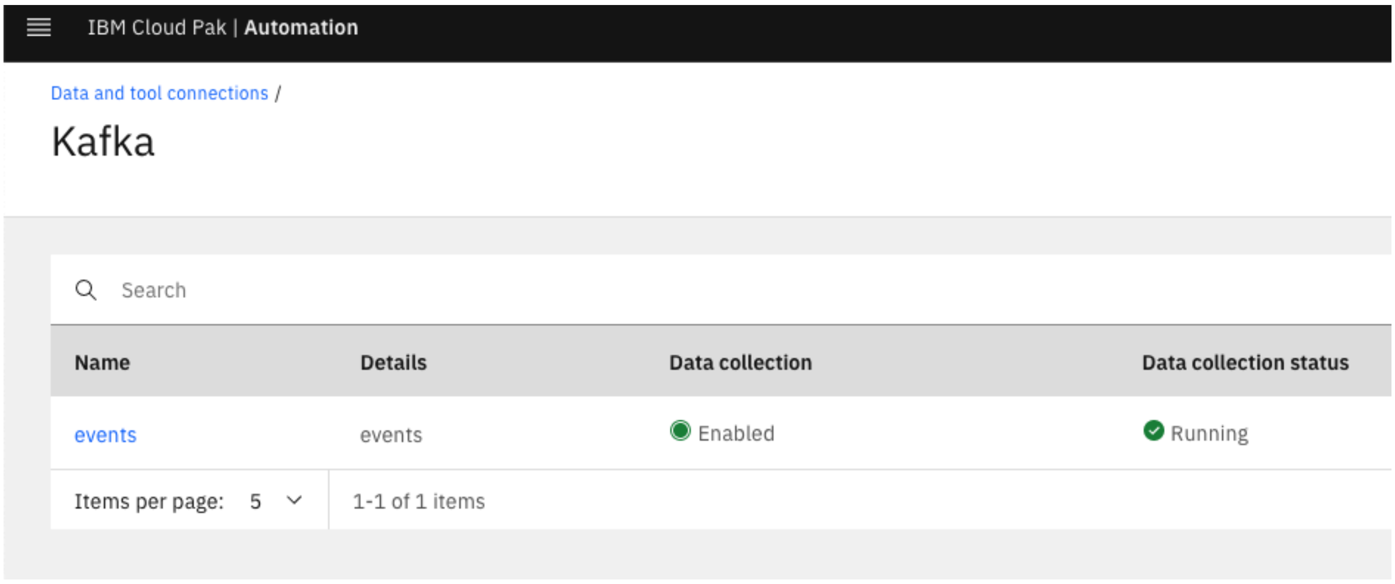
```
04:57 @root:~ >
04:57 @root:~ > aio topo -p test.json
Sending TOPOLOGY with PUT request to Observer-JobID restTopology
{ "name": "B", "uniqueId": "B.com.RHEL5", "matchTokens": [ "B.com.RHEL5","B" ], "tags": [ "INVOICES","app" ], "entityTypes": [ "server" ]}
202 Accepted
{ "name": "A", "uniqueId": "A.com.RHEL4", "matchTokens": [ "A.com.RHEL4","A" ], "tags": [ "INVOICES","front" ], "entityTypes": [ "router" ]}
202 Accepted
{ "_fromUniqueId": "A.com.RHEL4", "_edgeType": "connectedTo", "_toUniqueId": "B.com.RHEL5"}
202 Accepted
3/3 Resources have been processed
04:57 @root:~ >
```

If you go to the AI Manager Console > Resource Manager :



4- Ingesting events and alerts

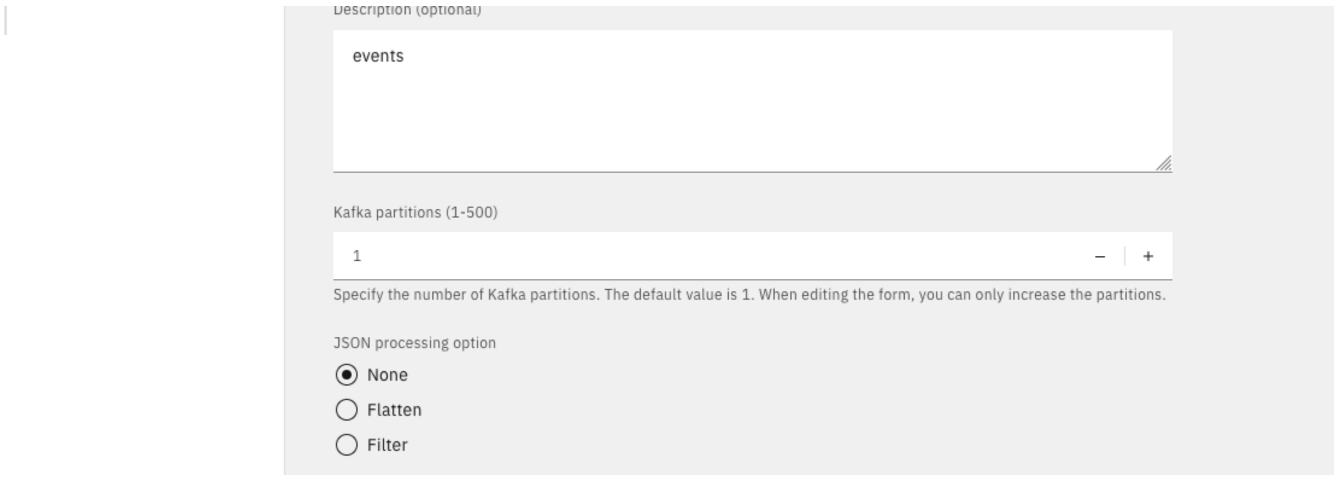
In that case, we want to send some events or alerts to the Cloud Pak. You need to define a **KAFKA connector** in the Cloud Pack web console. Let call this Kafka connector **events**



Name	Details	Data collection	Data collection status
events	events	● Enabled	✓ Running

Items per page: 5 1-1 of 1 items

You must get the Kafka topic in the connector **definitions**:


Description (optional)

events

Kafka partitions (1-500)

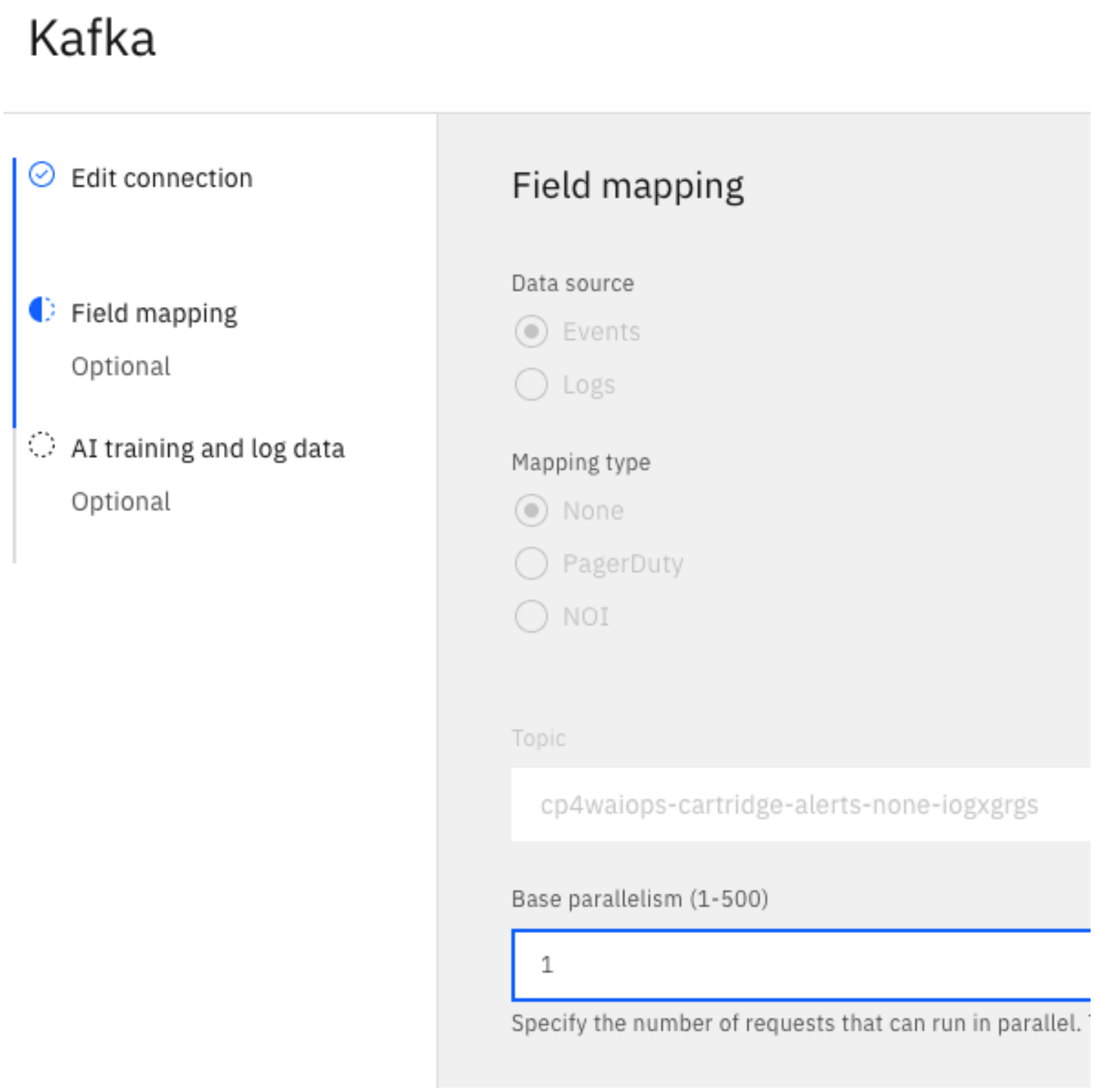
1

Specify the number of Kafka partitions. The default value is 1. When editing the form, you can only increase the partitions.

JSON processing option

[✓ None] Flatten Filter

Take a note of the **topic**:



Kafka

- Edit connection
- Field mapping
  Optional
- AI training and log data
  Optional

Field mapping

Data source
(● Events) ○ Logs

Mapping type
(● None) ○ PagerDuty ○ NOI

Topic
cp4waiops-cartridge-alerts-none-iogxgrgs

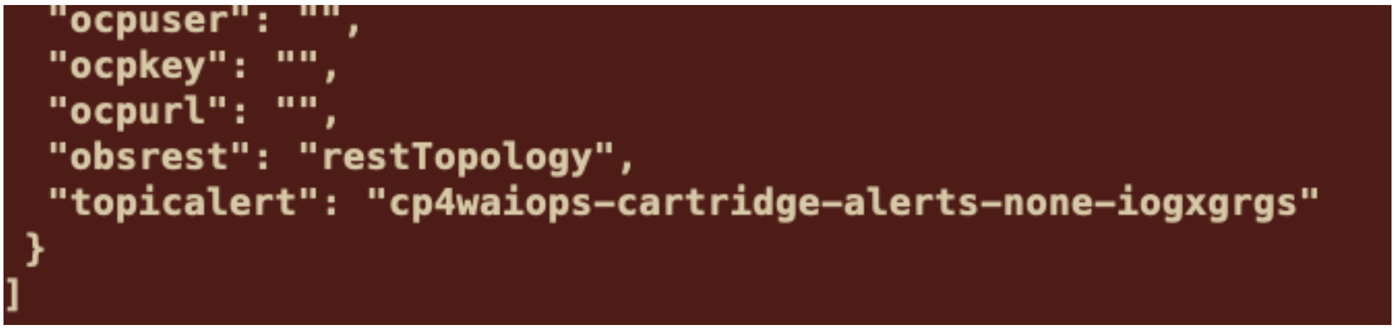
Base parallelism (1-500)
1
Specify the number of requests that can run in parallel.

In that case, the topic is :

```
cp4waiops-cartridge-alerts-none-iogxgrgs
```

You can also define this topic in the **.aio** file to be used by aio.

```
cat ~/.aio
```

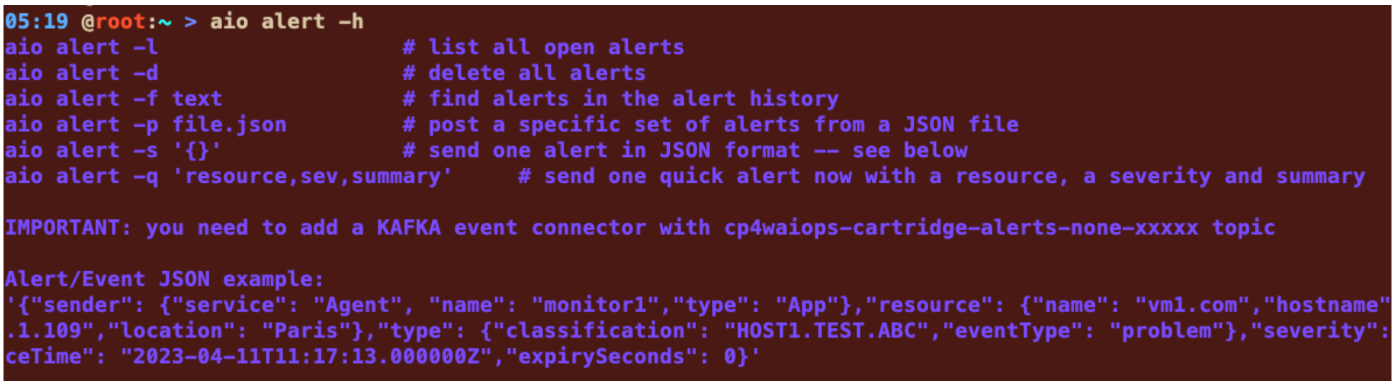


```
  "ocpuser": "",
  "ocpkey": "",
  "ocpurl": "",
  "obsrest": "restTopology",
  "topicalert": "cp4waiops-cartridge-alerts-none-iogxgrgs"
 }
]
```

If this topic is not specified, then aio will search for the first `*-alerts-none*` topic in the Kafka topic list.

Use the help to get the instruction to list or ingest alerts:

```
aio alert -h
```



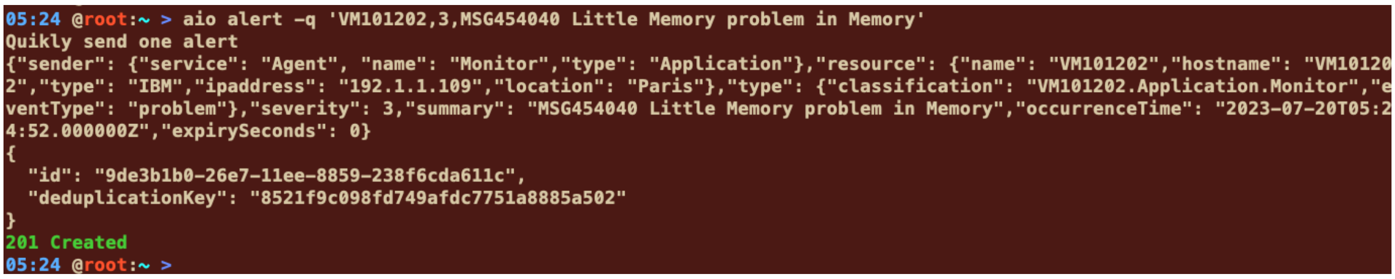
```
05:19 @root:~ > aio alert -h
aio alert -l                   # list all open alerts
aio alert -d                   # delete all alerts
aio alert -f text              # find alerts in the alert history
aio alert -p file.json         # post a specific set of alerts from a JSON file
aio alert -s '{}'              # send one alert in JSON format -- see below
aio alert -q 'resource,sev,summary'     # send one quick alert now with a resource, a severity and summary

IMPORTANT: you need to add a KAFKA event connector with cp4waiops-cartridge-alerts-none-xxxxx topic

Alert/Event JSON example:
'{"sender": {"service": "Agent", "name": "monitor1","type": "App"},"resource": {"name": "vm1.com","hostname"
.1.109","location": "Paris"},"type": {"classification": "HOST1.TEST.ABC","eventType": "problem"},"severity":
ceTime": "2023-04-11T11:17:13.000000Z","expirySeconds": 0}'
```

There are different ways to ingest alerts. The quick way that can be used is especially to check the Kafka topic is fine:

- VM101202 is a resource name;
- 3 is a severity
- MSG454040 Little Memory problem in Memory   is a summary message

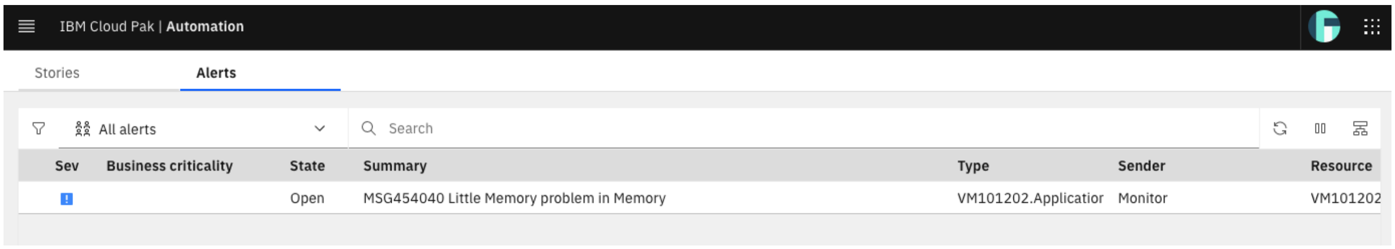
Let's start with a quick event:

```
aio alert -q 'VM101202,3,MSG454040 Little Memory problem in Memory'
```



```
05:24 @root:~ > aio alert -q 'VM101202,3,MSG454040 Little Memory problem in Memory'
Quikly send one alert
{"sender": {"service": "Agent", "name": "Monitor","type": "Application"},"resource": {"name": "VM101202","hostname": "VM10120
2","type": "IBM","ipaddress": "192.1.1.109","location": "Paris"},"type": {"classification": "VM101202.Application.Monitor","e
ventType": "problem"},"severity": 3,"summary": "MSG454040 Little Memory problem in Memory","occurrenceTime": "2023-07-20T05:2
4:52.000000Z","expirySeconds": 0}
{
  "id": "9de3b1b0-26e7-11ee-8859-238f6cda611c",
  "deduplicationKey": "8521f9c098fd749afdc7751a8885a502"
}
201 Created
05:24 @root:~ >
```

Then you can go to the Cloud Pak web console and look at the alerts:



Sev	Business criticality	State	Summary	Type	Sender	Resource
!		Open	MSG454040 Little Memory problem in Memory	VM101202.Applicatior	Monitor	VM101202

Let's try another way : using a JSON payload that contains almost all the necessary parameters.

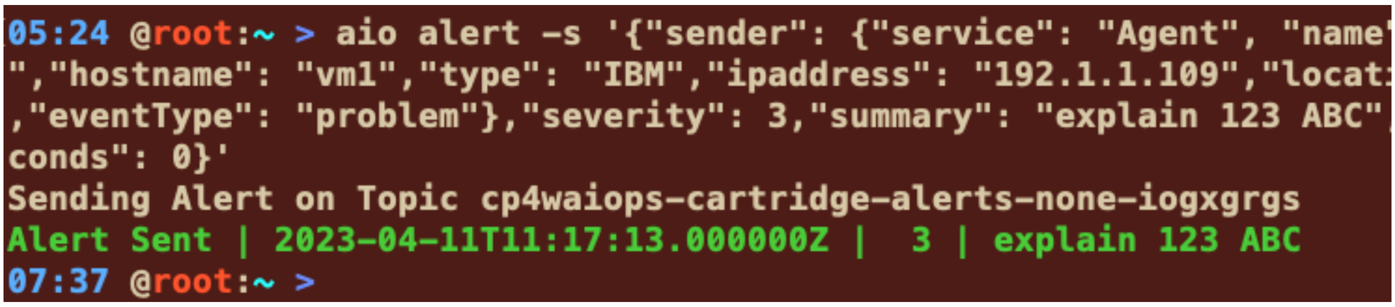
Let's prepare a JSON payload :

```
'{"sender": {"service": "Agent", "name": "monitor1","type": "App"},"resource": {"name":
"vm1.com","hostname": "vm1","type": "IBM","ipaddress": "192.1.1.109","location":
"Paris"},"type": {"classification": "HOST1.TEST.ABC","eventType":
"problem"},"severity": 3,"summary": "explain 123 ABC","occurrenceTime": "2023-07-
20T11:17:13.000000Z","expirySeconds": 0}'
```

Don't forget to modify the **occurenceTime** with an updated **date**.

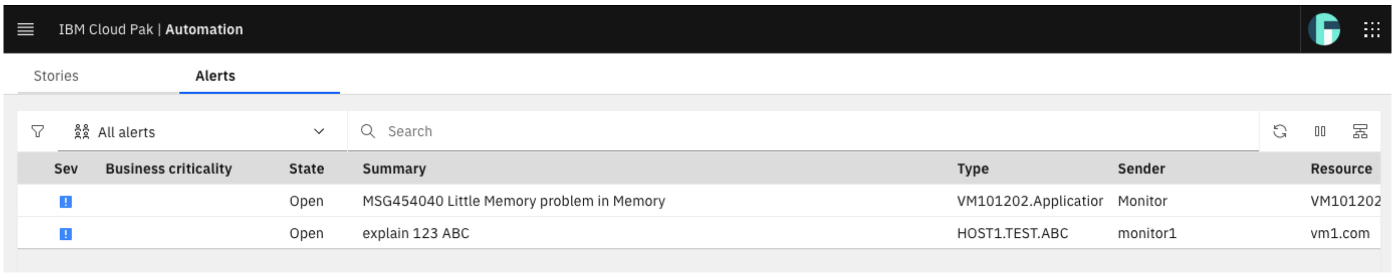
Now use the following command:

```
ai alert -s '{"sender": {"service": "Agent", "name": "monitor1","type":
"App"},"resource": {"name": "vm1.com","hostname": "vm1","type": "IBM","ipaddress":
"192.1.1.109","location": "Paris"},"type": {"classification":
"HOST1.TEST.ABC","eventType": "problem"},"severity": 3,"summary": "explain 123
ABC","occurrenceTime": "2023-07-20T11:17:13.000000Z","expirySeconds": 0}'
```



```
05:24 @root:~ > aio alert -s '{"sender": {"service": "Agent", "name'
","hostname": "vm1","type": "IBM","ipaddress": "192.1.1.109","locat
,"eventType": "problem"},"severity": 3,"summary": "explain 123 ABC",
conds": 0}'
Sending Alert on Topic cp4waiops-cartridge-alerts-none-iogxgrgs
Alert Sent | 2023-04-11T11:17:13.000000Z |  3 | explain 123 ABC
07:37 @root:~ >
```

Then look to the Cloud Pak web console:

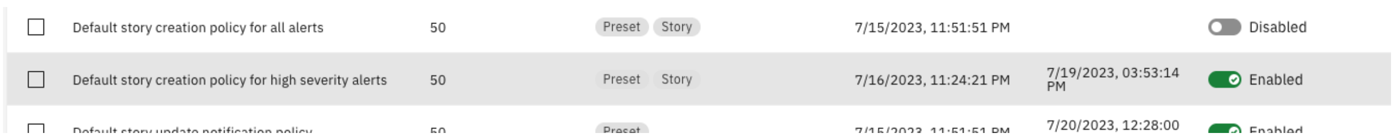


Sev	Business criticality	State	Summary	Type	Sender	Resource
!		Open	MSG454040 Little Memory problem in Memory	VM101202.Applicatior	Monitor	VM101202
!		Open	explain 123 ABC	HOST1.TEST.ABC	monitor1	vm1.com

Now if we want to generate a story, let us go to the polocies :

```
Menu > Operate > Automations > Policies
```

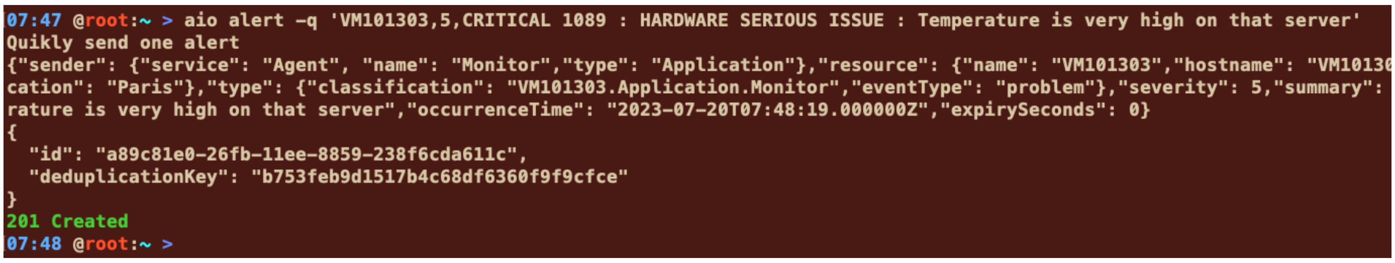
Activate the `Default story creation policy for high severity alerts`



☐	Default story creation policy for all alerts	50	Preset Story	7/15/2023, 11:51:51 PM		Disabled
☐	Default story creation policy for high severity alerts	50	Preset Story	7/16/2023, 11:24:21 PM	7/19/2023, 03:53:14 PM	Enabled
☐	Default story update notification policy	50	Preset	7/15/2023, 11:51:51 PM	7/20/2023, 12:28:00	Enabled

We must trigger a severity 5 or 6 alert to generate a story:

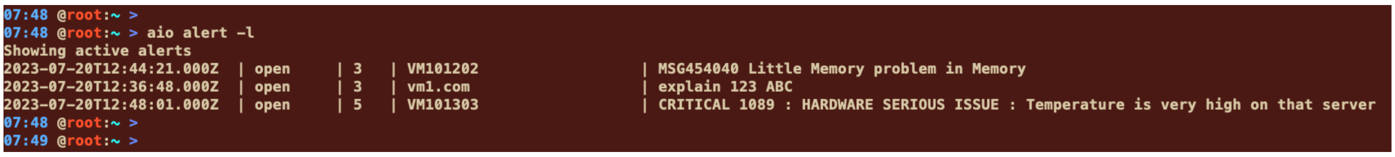
```
aio alert -q 'VM101303,5,CRITICAL 1089 : HARDWARE SERIOUS ISSUE : Temperature is very
high on that server'
```



```
07:47 @root:~ > aio alert -q 'VM101303,5,CRITICAL 1089 : HARDWARE SERIOUS ISSUE : Temperature is very high on that server'
Quikly send one alert
{"sender": {"service": "Agent", "name": "Monitor","type": "Application"},"resource": {"name": "VM101303","hostname": "VM10130
cation": "Paris"},"type": {"classification": "VM101303.Application.Monitor","eventType": "problem"},"severity": 5,"summary":
rature is very high on that server","occurrenceTime": "2023-07-20T07:48:19.000000Z","expirySeconds": 0}
{
  "id": "a89c81e0-26fb-11ee-8859-238f6cda611c",
  "deduplicationKey": "b753feb9d1517b4c68df6360f9f9cfce"
}
201 Created
07:48 @root:~ >
```

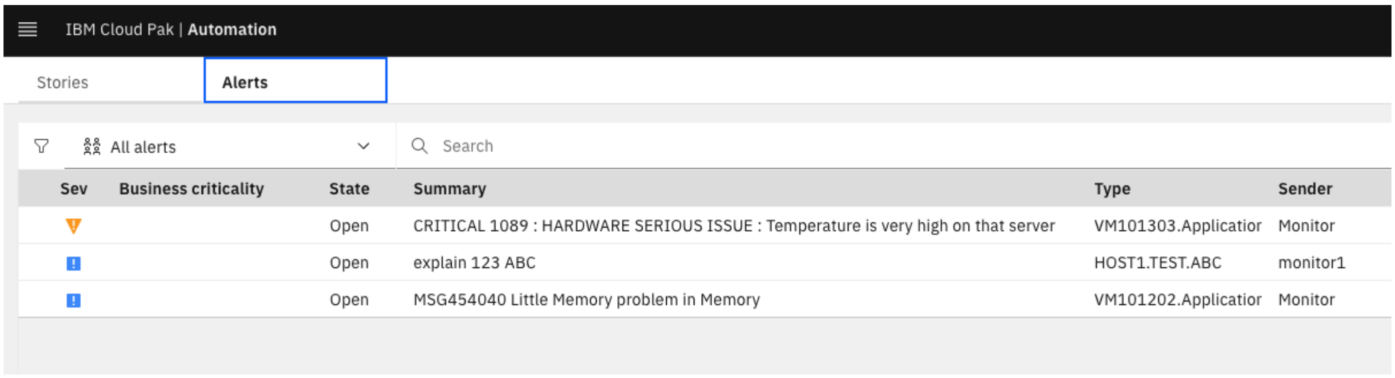
You can look to the alert list:

```
aio alert -l
```



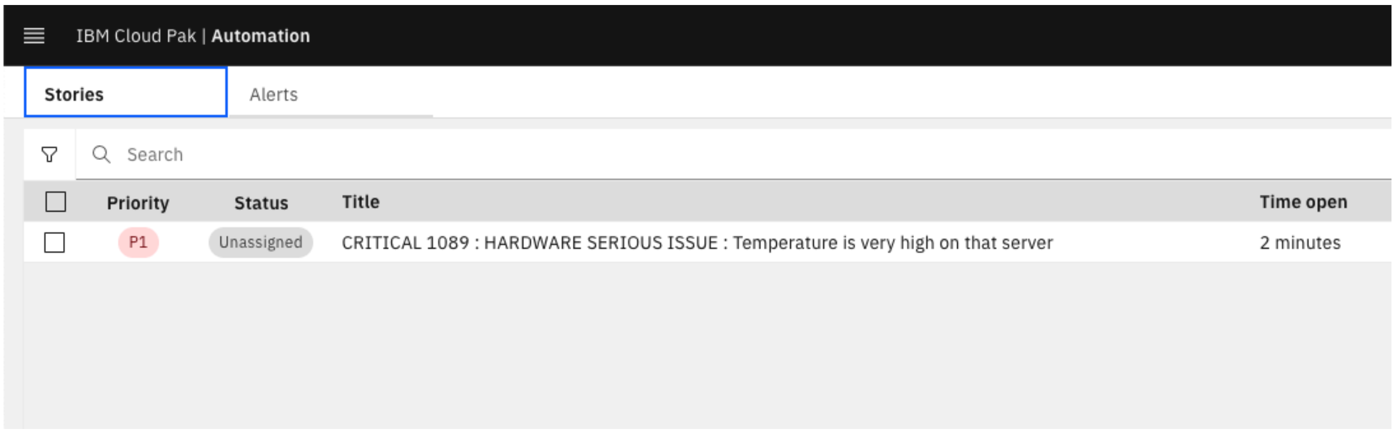
```
07:48 @root:~ >
07:48 @root:~ > aio alert -l
Showing active alerts
2023-07-20T12:44:21.000Z  | open     | 3   | VM101202                        | MSG454040 Little Memory problem in Memory
2023-07-20T12:36:48.000Z  | open     | 3   | vm1.com                         | explain 123 ABC
2023-07-20T12:48:01.000Z  | open     | 5   | VM101303                        | CRITICAL 1089 : HARDWARE SERIOUS ISSUE : Temperature is very high on that server
07:48 @root:~ >
07:49 @root:~ >
```

or



Sev	Business criticality	State	Summary	Type	Sender
▼		Open	CRITICAL 1089 : HARDWARE SERIOUS ISSUE : Temperature is very high on that server	VM101303.Applicatior	Monitor
!		Open	explain 123 ABC	HOST1.TEST.ABC	monitor1
!		Open	MSG454040 Little Memory problem in Memory	VM101202.Applicatior	Monitor

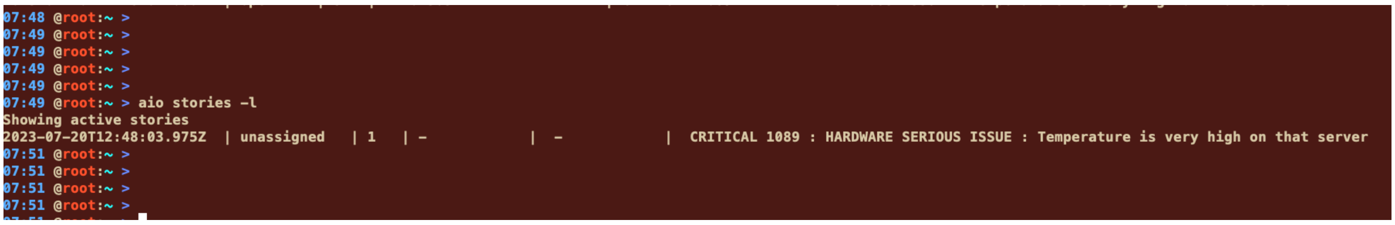
or you can also look at the stories:



	Priority	Status	Title	Time open
☐	P1	Unassigned	CRITICAL 1089 : HARDWARE SERIOUS ISSUE : Temperature is very high on that server	2 minutes

or

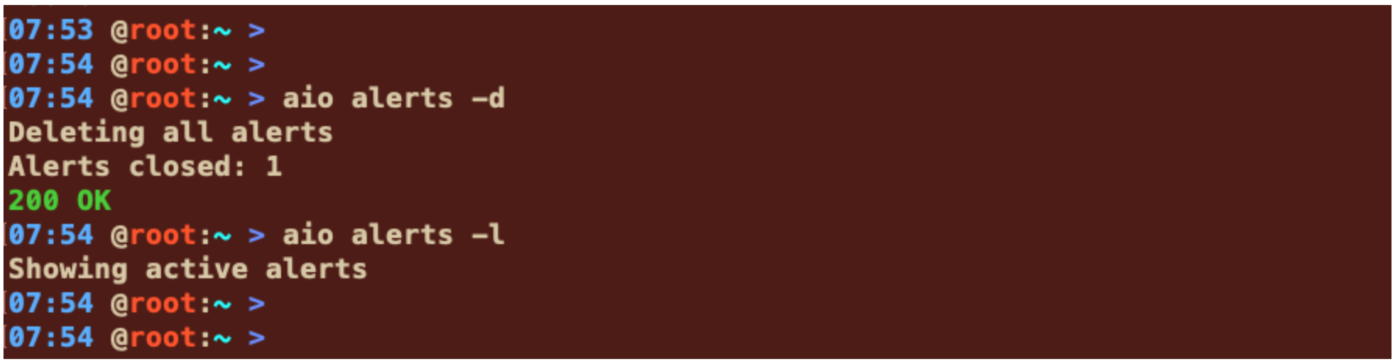
```
aio stories -l
```



```
07:48 @root:~ >
07:49 @root:~ >
07:49 @root:~ >
07:49 @root:~ >
07:49 @root:~ >
07:49 @root:~ > aio stories -l
Showing active stories
2023-07-20T12:48:03.975Z  | unassigned   | 1   | -           |  -           |  CRITICAL 1089 : HARDWARE SERIOUS ISSUE : Temperature is very high on that server
07:51 @root:~ >
07:51 @root:~ >
07:51 @root:~ >
07:51 @root:~ >
```

Then you can delete all the alerts (it takes 2 minutes to complete) after a short answer:

```
aio alerts -d
```



```
07:53 @root:~ >
07:54 @root:~ >
07:54 @root:~ > aio alerts -d
Deleting all alerts
Alerts closed: 1
200 OK
07:54 @root:~ > aio alerts -l
Showing active alerts
07:54 @root:~ >
07:54 @root:~ >
```

Conclusion

With this tool you can switch from one cluster to another one easily, you can ingest new files into the metric manager or define new topologies on an easier way and ingest new alerts.

Thanks.