# Workshop Kafka Connect

The code for this workshop is located at:

Git clone

## Preparations: Docker and Docker-Compose

This workshop requires a working docker and docker-compose.

### Linux

#docker

curl -fsSL https://get.docker.com -o get-docker.sh

sudo sh get-docker.sh

#docker-compose

sudo curl -L "https://github.com/docker/compose/releases/download/1.23.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose

Make sure the user who is going to use docker is in the docker group

### Windows

Enable Hyper-V: <https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/quick-start/enable-hyper-v>

Install Chocolatey: <https://chocolatey.org/>

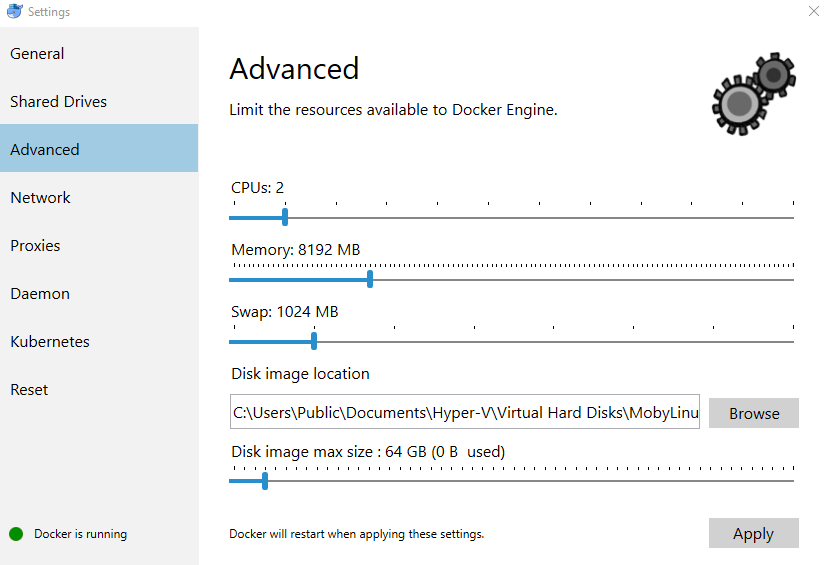
Install Docker, Docker Compose and Curl

choco install docker-desktop

choco install docker-compose

choco install curl

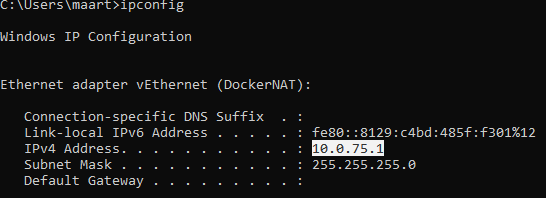
Make sure the Hyper-V VM which runs Docker has at least 8Gb of memory:



Determine the IP by which you can access the Docker host:

ipconfig

Lookup DockerNAT and use the IPv4 Address to access exposed ports from Docker containers



You can also execute the following Powershell command:

(Get-NetIPAddress | Where-Object { $\_.InterfaceAlias -eq "vEthernet (DockerNAT)" -and $\_.AddressFamily -eq "IPv4"}).IPAddress

For executing PowerShell scripts:

Set-ExecutionPolicy -ExecutionPolicy Unrestricted

## Running Elasticsearch

Elasticsearch has an additional requirement as explained here: <https://www.elastic.co/guide/en/elasticsearch/reference/current/docker.html> under ‘Production Mode’. Below the required commands.

### Linux

sudo sysctl -w vm.max\_map\_count=262144

### Windows

docker run -it --rm --privileged --pid=host justincormack/nsenter1

sysctl -w vm.max\_map\_count=262144

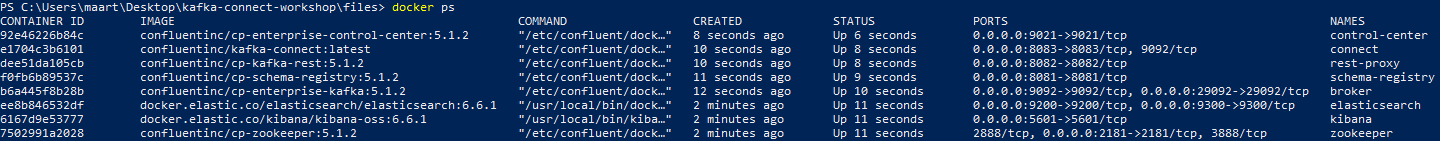
## Starting the environment

The environment can be started by going to the ‘files’ folder inside an elevated Powershell and

docker-compose up

It consists of:

docker ps



You can browse the different interfaces at (replace localhost with the IP of the DockerNAT interface for Windows):

Elasticsearch (REST API only): <http://localhost:9200>

Kibana: <http://localhost:5601>

Connect (REST API only): <http://localhost:8083>

Control Center: <http://localhost:9021>

Kafka can be accessed with for example Kafkatool to browse messages. Kafka Tool can be installed from: <http://www.kafkatool.com/> (different versions are available)

## Kafka Connect

The Kafka Connect container is created by the supplied Dockerfile. It consists of Kafka Connect, configured to run in distributed mode and it contains a file outputas.txt.

This file has been created by monitoring filesystem events during the starting of an application server and will serve as input for Kafka Connect FileStreamSource connector. The Kafka topic connect-test is filled with the data from the file. Next the ElasticSearchSink connector is used to put this data into ElasticSearch. Kibana is a dashboard on top of ElasticSearch.

### Load the filesourceconnector

Examine the file filesource.json and the respective load script below

Linux

sh ./loadfilesource.sh

Windows

loadfilesource.ps1

### Load the ElasticSearchSink connector

Examine the file elasticsearchsink.json

Linux

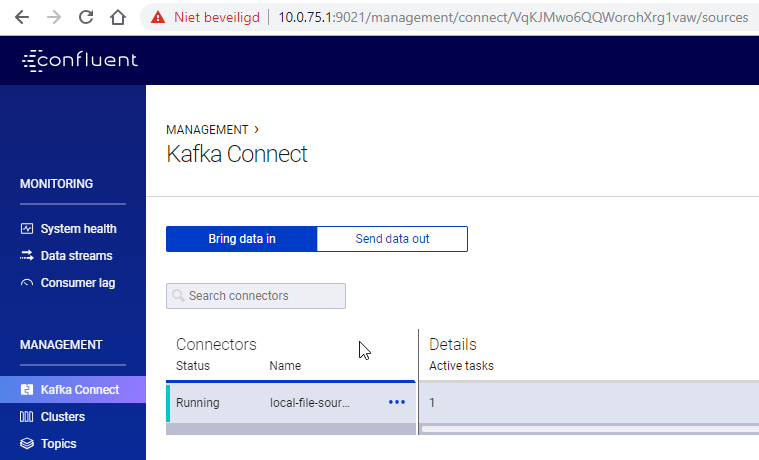
sh ./loadelasticsearchsink.sh

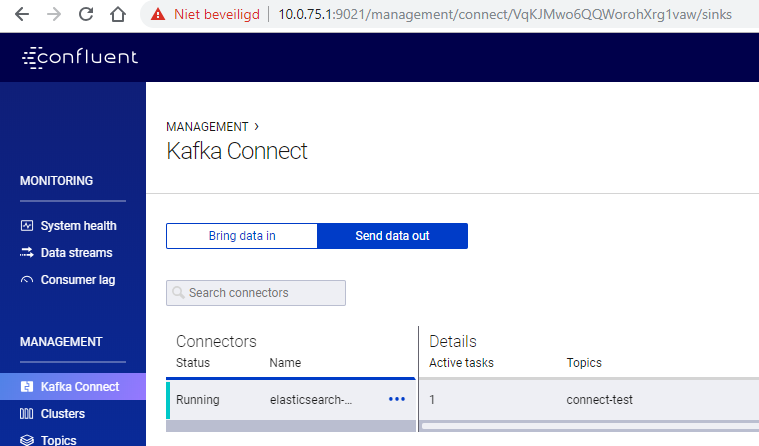
Windows

loadelasticsearchsink.ps1

### Examine the results

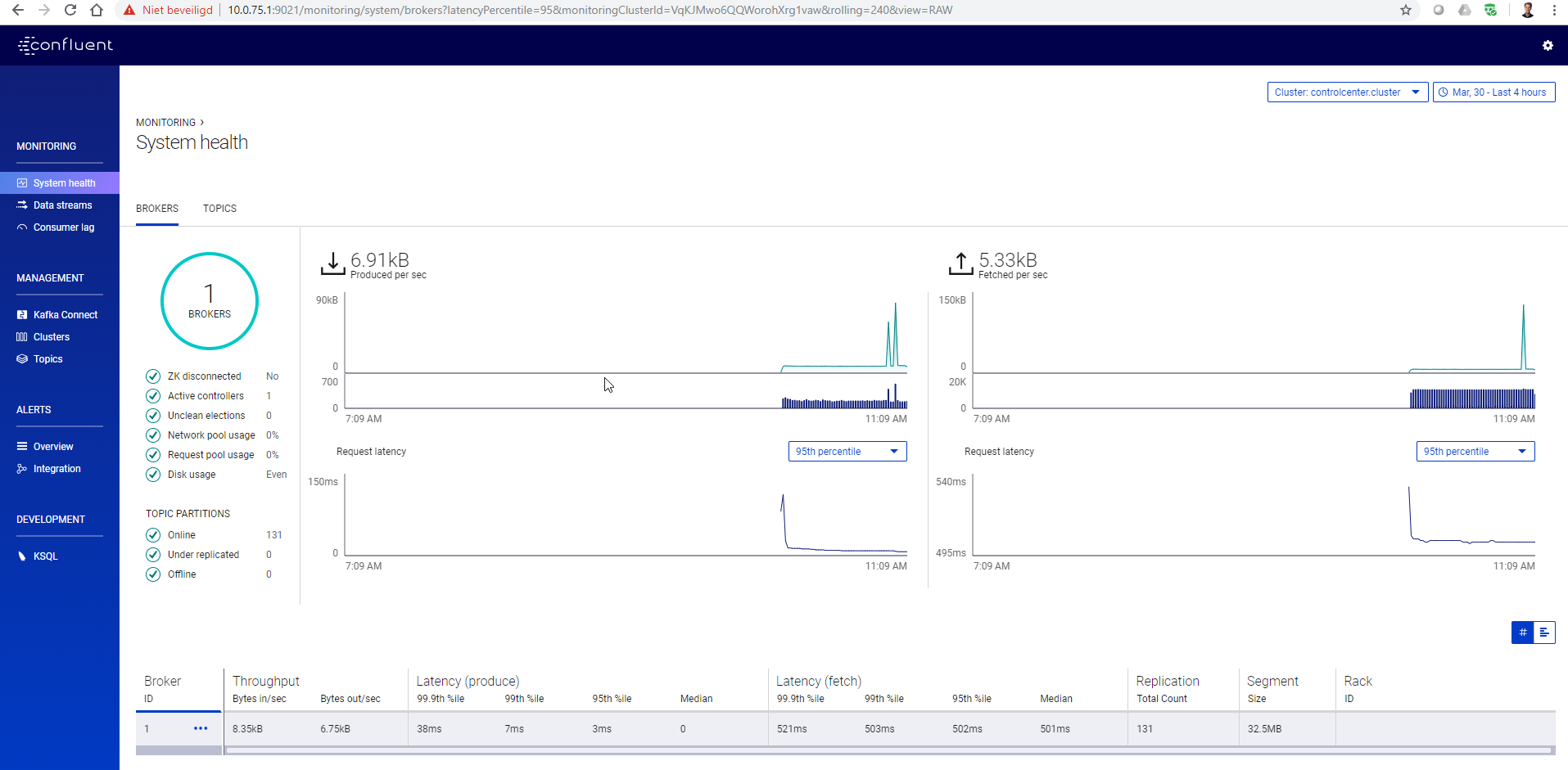
Open the Control Center and confirm the presence of the connectors:



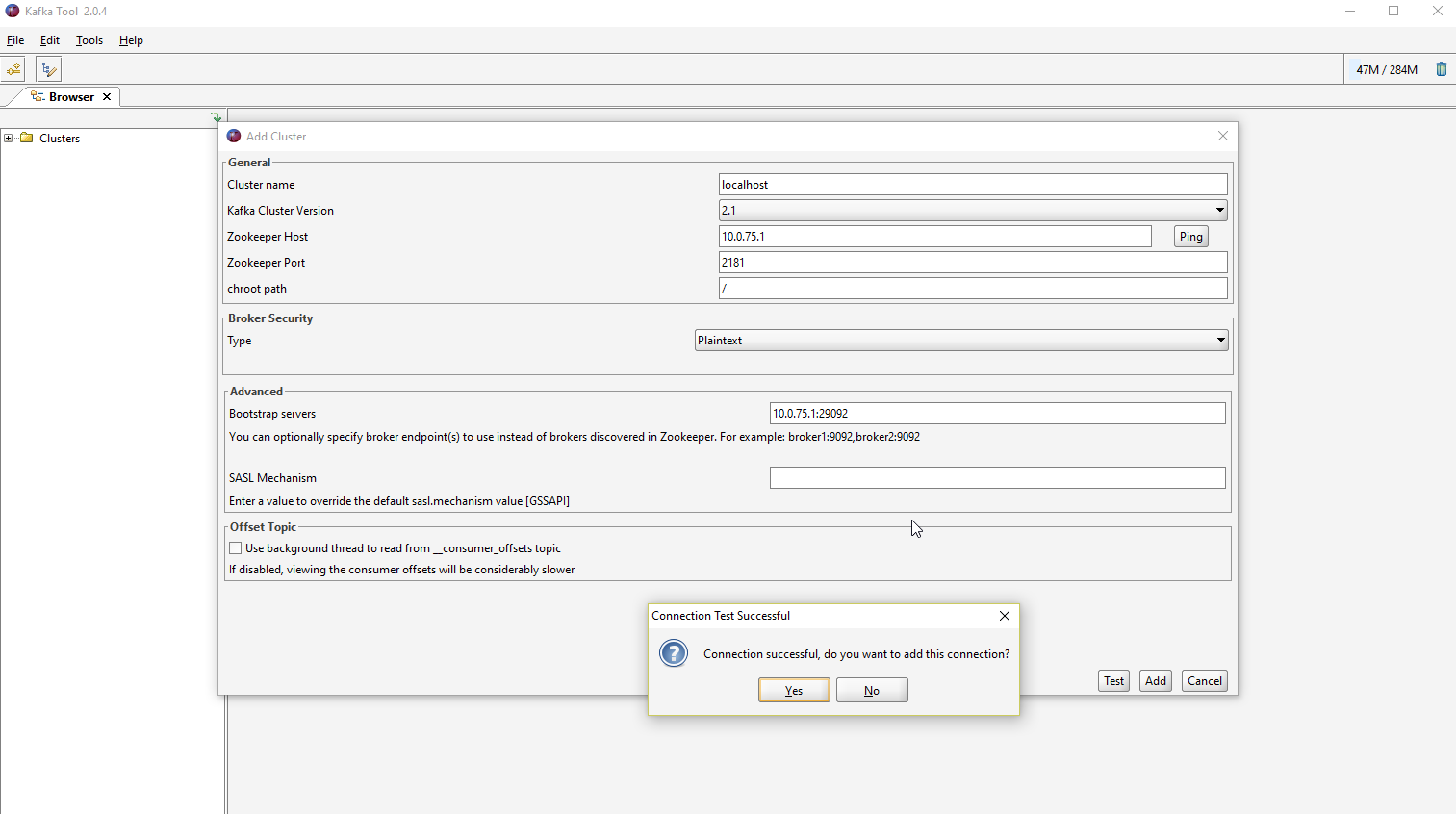


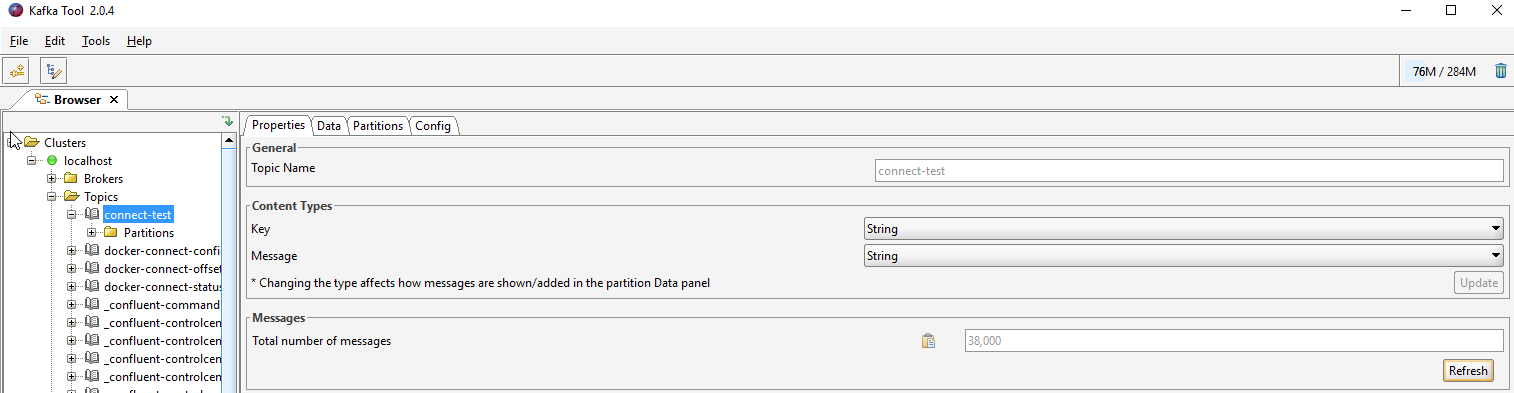
Confirm the activity of the connect-test topic

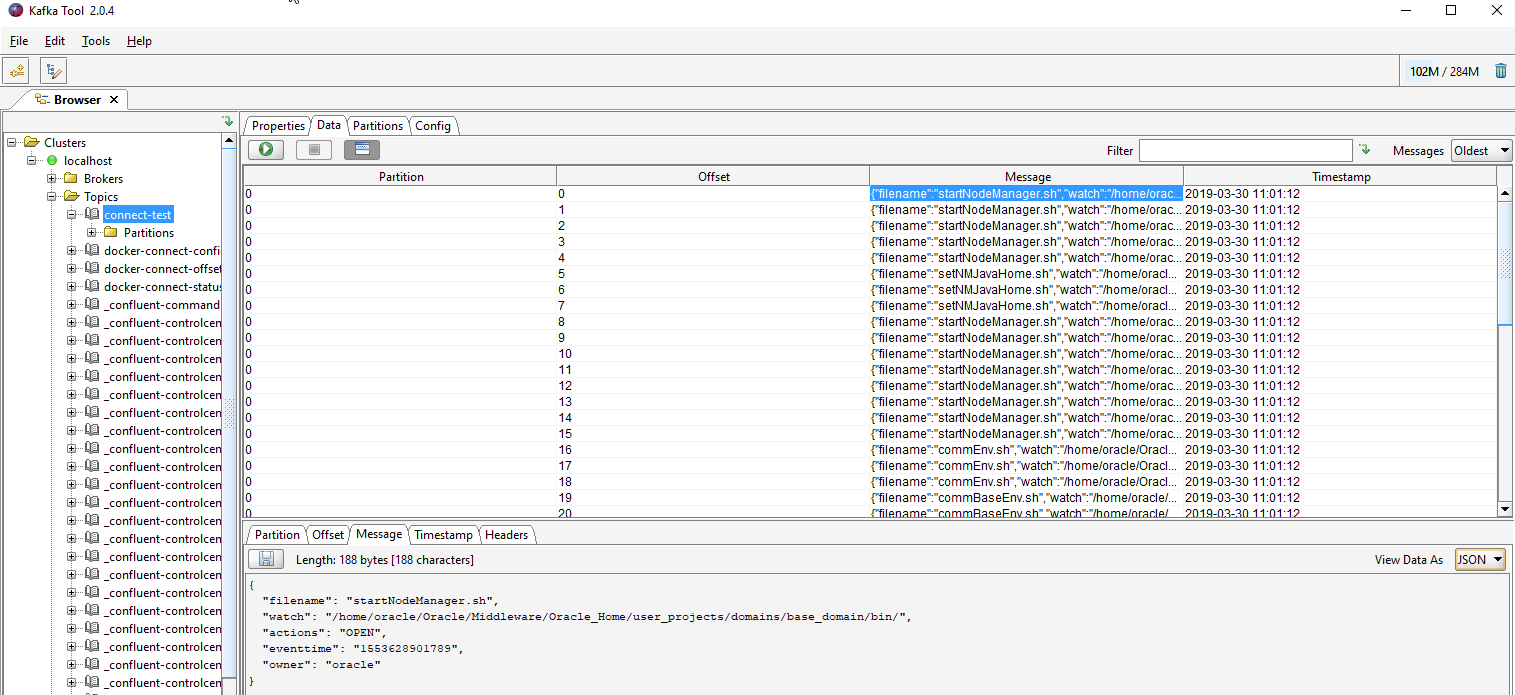




Look at the messages on the topic connect-test using Kafka Tool







## Kibana

Open Kibana and examine the data from ElasticSearch

