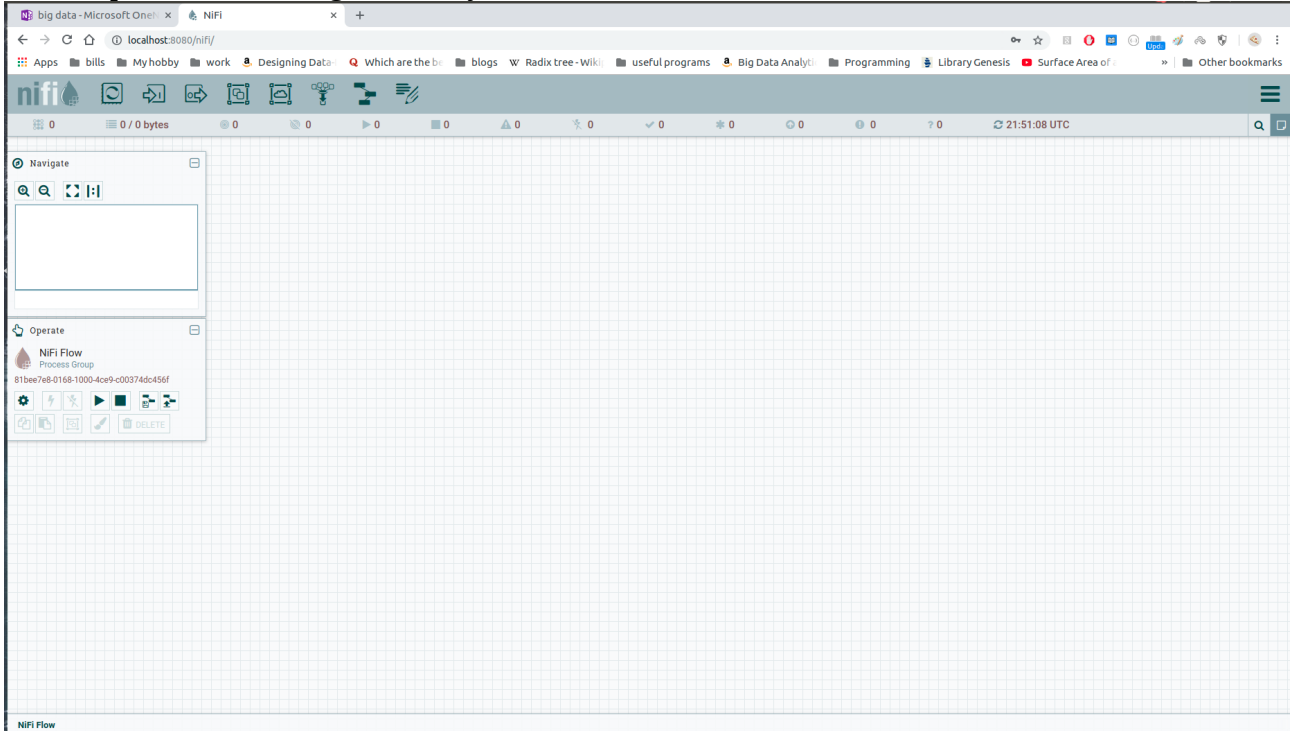


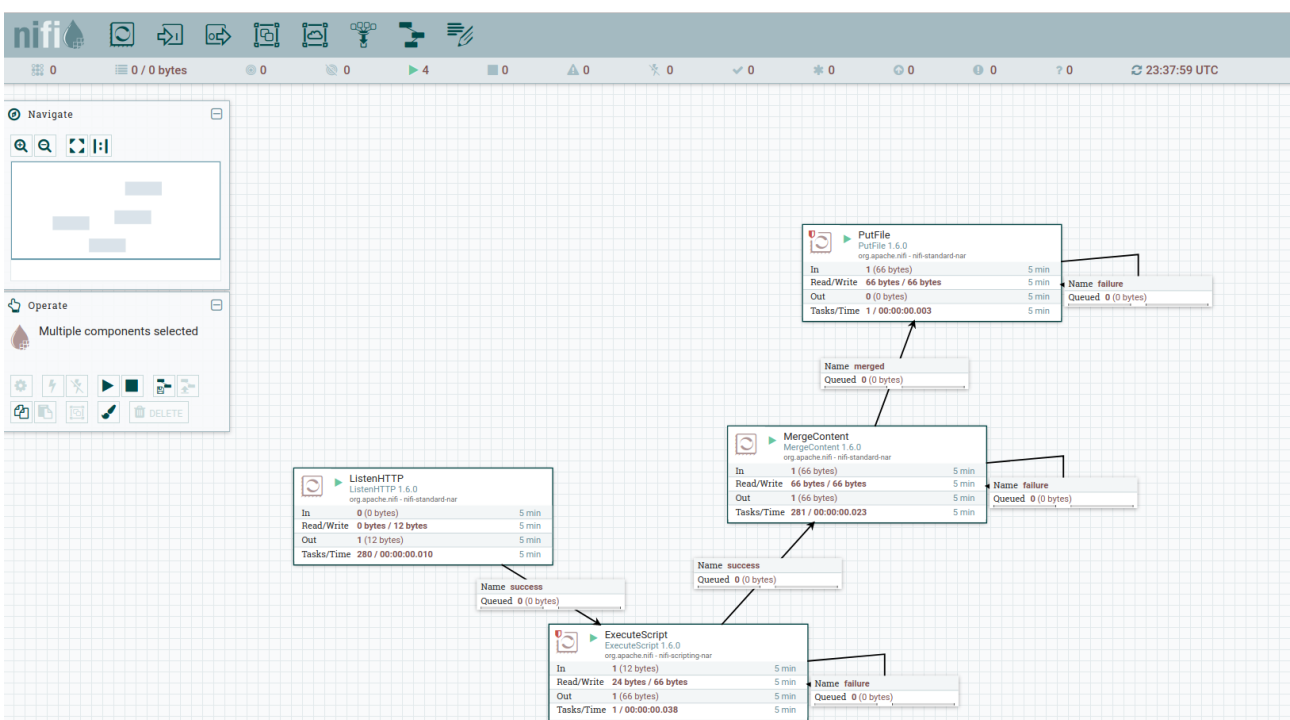
## Apache Nifi report.

So, first of all I downloaded the docker image, using the “docker pull apache/nifi:1.6.0” command. Then I launched a container with that image on port 8080, using the “docker run -p 8080:8080 apache/nifi:1.6.0” command.

Here’s apache nifi running within my browser:



Next up, I added the same processors as described in the lecture, except between the **ListenHTTP** processor and **MergeContentProcessor** I added an **ExecuteScript** processor. Here’s my screenshot of this setup:



Inside the **ExecuteScript** processor, I added a Groovy script that transforms our message into json format, adding a timestamp. So, if we receive a message “Hello” at 5 am, the json will look like this: **{“timestamp”:”5 am”, “message”:”Hello”}**

Here is my script:

```
import org.apache.commons.io.IOUtils
import java.nio.charset.StandardCharsets
import java.time.LocalDateTime

flowFile = session.get()
if(!flowFile)return
def text = ''

session.read(flowFile, {inputStream ->
    text = IOUtils.toString(inputStream, StandardCharsets.UTF_8)
} as InputStreamCallback)

def outputMessage = '{"timestamp":"' + LocalDateTime.now().toString() + '\", \"message\":\"' + text + '\"}'

flowFile = session.write(flowFile, {inputStream, outputStream ->
    text = IOUtils.toString(inputStream, StandardCharsets.UTF_8)
    outputStream.write(outputMessage.getBytes(StandardCharsets.UTF_8))
} as StreamCallback)

session.transfer(flowFile, REL_SUCCESS)
```

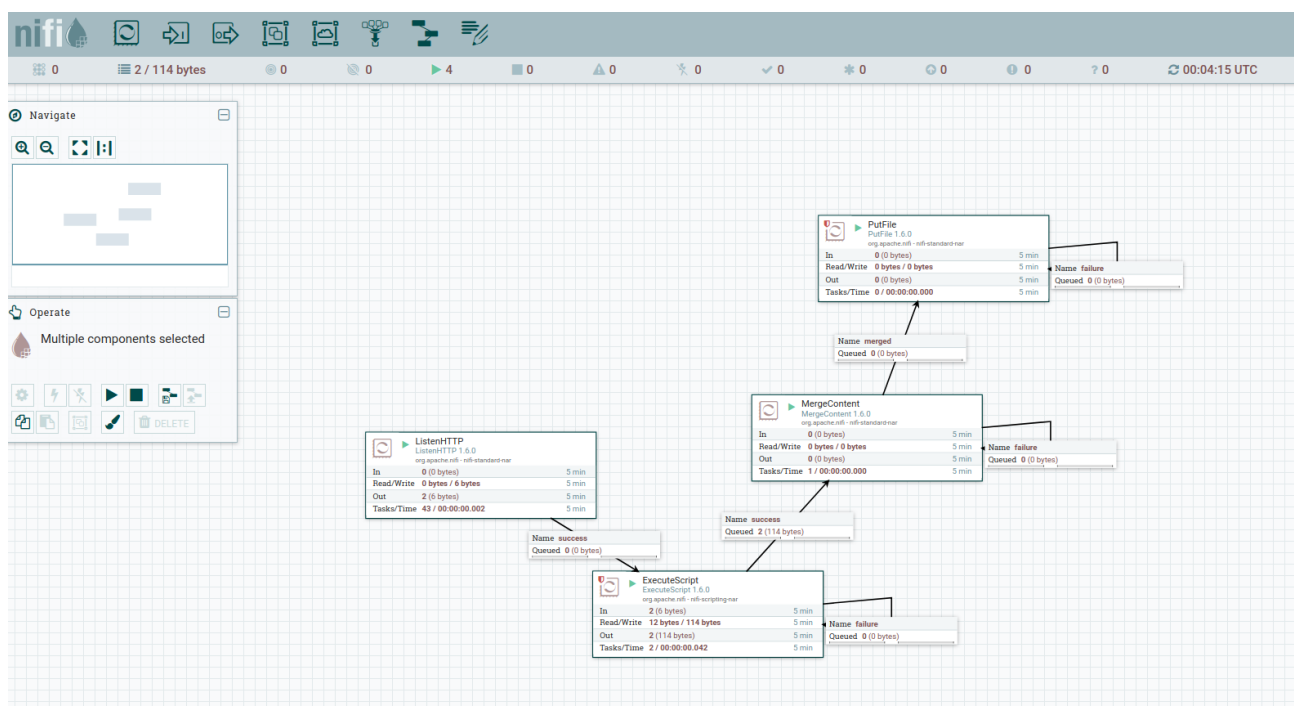
In order to stall messages in a queue, so that they can be collected into batches I changed **MergeContent** processor’s scheduling settings to this (note the **60 sec** setting):

SETTINGS	SCHEDULING	PROPERTIES	COMMENTS
<b>Scheduling Strategy</b> ? Timer driven		<b>Run Duration</b> ? 00:00:00.000	
<b>Concurrent Tasks</b> ? 1		<b>Run Schedule</b> ? 60 sec	

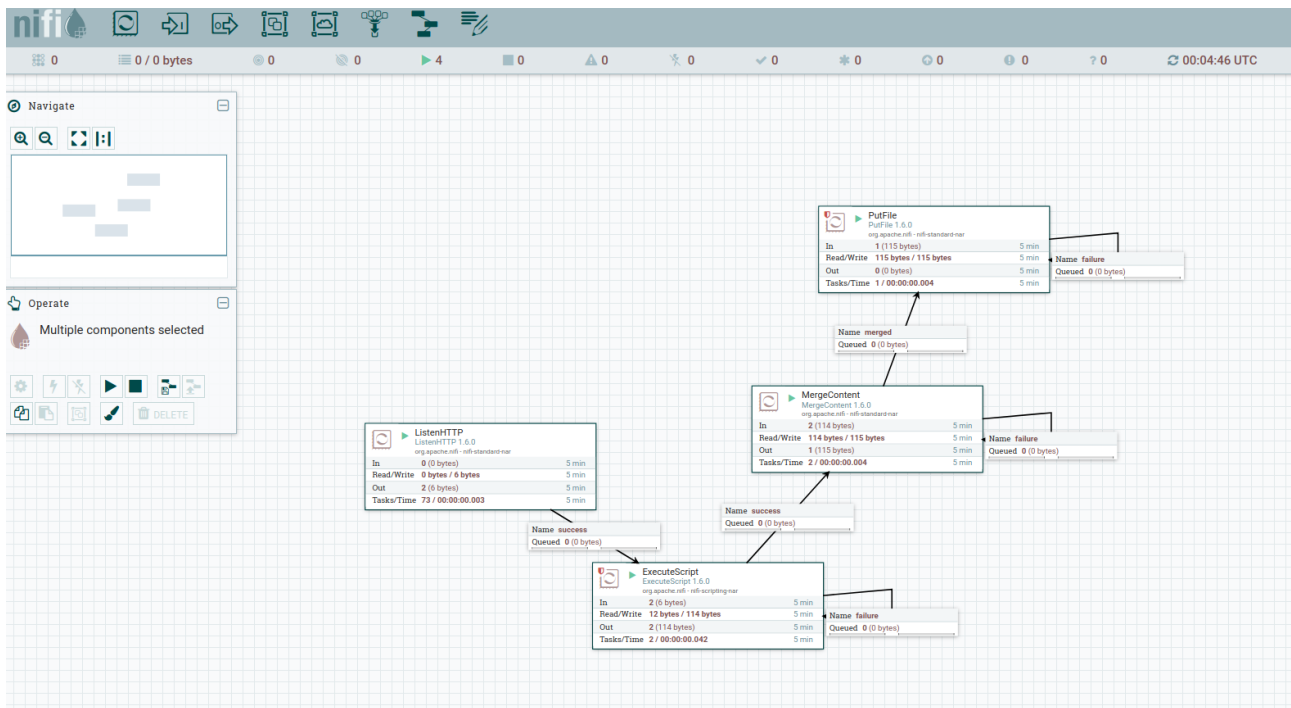
After I did that and sent 2 messages like so:

- curl -X POST -d "one" <http://127.0.0.1:4141/data>
- curl -X POST -d "two" <http://127.0.0.1:4141/data>

I was able to see **MergeContent** processor stalling these 2 messages in a queue, until the current minute was up:



From the picture above you can see the 2 queued messages. And this is the picture after that minute was up:



As you can see, the 2 merged messages were processed into 1, and written to disk.

Then I sent the third message, and this is the contents of my /tmp/nifi folder:

```
nifi@cca79f2af03d:/opt/nifi/nifi-1.6.0$ ls /tmp/nifi
21501741934280 21550473693423
nifi@cca79f2af03d:/opt/nifi/nifi-1.6.0$
```

These and the contents of those 2 files:

```
nifi@cca79f2af03d:/opt/nifi/nifi-1.6.0$ cat /tmp/nifi/21501741934280
{"timestamp":"2019-01-25T00:03:45.716", "message":"one"}
{"timestamp":"2019-01-25T00:04:11.970", "message":"two"}nifi@cca79f2af03d:/opt/nifi/nifi-1.6.0$
nifi@cca79f2af03d:/opt/nifi/nifi-1.6.0$ cat /tmp/nifi/21550473693423
{"timestamp":"2019-01-25T00:05:22.390", "message":"three"}nifi@cca79f2af03d:/opt/nifi/nifi-1.6.0$
```

As expected, the first 2 messages were batched into one, and the third one is by itself.

## Exporting templates.

If we go to Global Menu → Templates, we will see every one of our saved templates. In order to download them, we need to click on the **Download** button which is on the right:

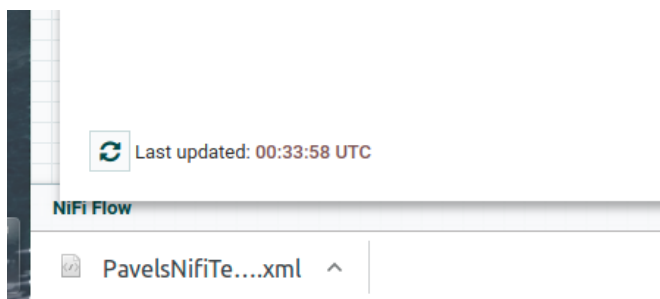
NiFi Templates

Displaying 1 of 1

pa by name

Date/Time	Name	Description	Process Group Id
01/24/2019 23:53:13 UTC	PavelsNiFiTemplate	A nifi template that listens for post requests on an HTTP port, timestamps then, and pu...	81bee7e8-0168-1000-4ce9-c00374dc456f

Here's the downloaded template:



Once we open it, we are able to see this content:

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```

- <template encoding-version="1.2">
- <description>
  A nifi template that listens for post requests on an HTTP port, timestamps then, and puts them into the HDFS, batching them every minute.
</description>
<groupId>81bee7e8-0168-1000-4ce9-c00374dc456f</groupId>
<name>PavelsNifiTemplate</name>
- <snippet>
- <connections>
  <id>e46d7d0c-181c-32a2-0000-000000000000</id>
  <parentGroupId>401503f7-e217-34f3-0000-000000000000</parentGroupId>
  <backPressureDataSizeThreshold>1 GB</backPressureDataSizeThreshold>
  <backPressureObjectThreshold>10000</backPressureObjectThreshold>
- <destination>
  <groupId>401503f7-e217-34f3-0000-000000000000</groupId>
  <id>48390f21-d20c-3b77-0000-000000000000</id>
  <type>PROCESSOR</type>
</destination>
<flowFileExpiration>0 sec</flowFileExpiration>
<labelIndex>1</labelIndex>
<name/>
<selectedRelationships>success</selectedRelationships>
- <source>
  <groupId>401503f7-e217-34f3-0000-000000000000</groupId>
  <id>bd343ba3-3abd-3a0d-0000-000000000000</id>
  <type>PROCESSOR</type>
</source>
<zIndex>0</zIndex>
</connections>
- <connections>
  <id>fdc40a37-8a01-33ca-0000-000000000000</id>
  <parentGroupId>401503f7-e217-34f3-0000-000000000000</parentGroupId>
  <backPressureDataSizeThreshold>1 GB</backPressureDataSizeThreshold>
  <backPressureObjectThreshold>10000</backPressureObjectThreshold>
- <destination>
  <groupId>401503f7-e217-34f3-0000-000000000000</groupId>
  <id>bd343ba3-3abd-3a0d-0000-000000000000</id>
  <type>PROCESSOR</type>
</destination>
<flowFileExpiration>0 sec</flowFileExpiration>
<labelIndex>1</labelIndex>

```

QED.