The `openmsipython` Software Package and Open MSI Directory Stream Service

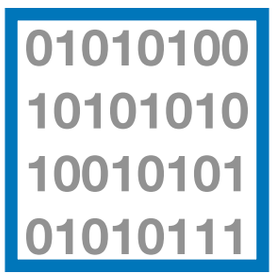
Maggie Eminizer
DMREF monthly meeting, May 7th 2021

Institute for Data Intensive Engineering and Science (IDIES), JHU Physics & Astronomy
Work supported by Open MSI (NSF DMREF award #1921959)

The `openmsipython` software package

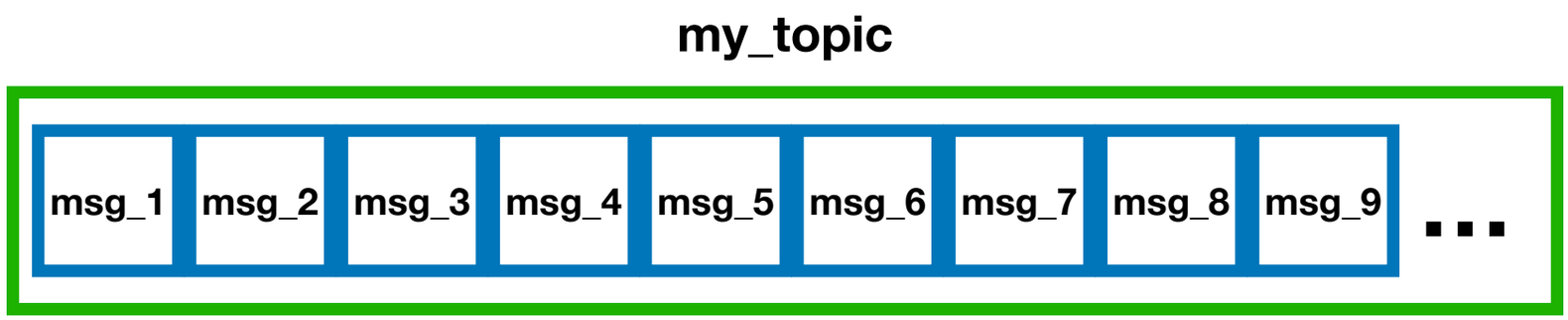
- Github: https://github.com/openmsi/openmsipython
- Python implementations of data streaming applications using Apache Kafka
- Current applications:
  - Upload files to a cloud-based server
    - break files into “messages”
    - produce those messages to a topic on a Kafka cluster
  - Download files from the cloud-based server
    - Reconstruct files locally from their messages stored in the topic
- How we intend to use this
  - Stream data from computers connected to lab instruments
  - Maintain a central & persistent data repository
  - Accelerate pace of development by automatically running analyses as soon as data are available
- Today I want to show you all how to use what I’ve written to set up an easy and reliable way to get data files uploaded to the cloud as part of your workflow

Apache Kafka, briefly



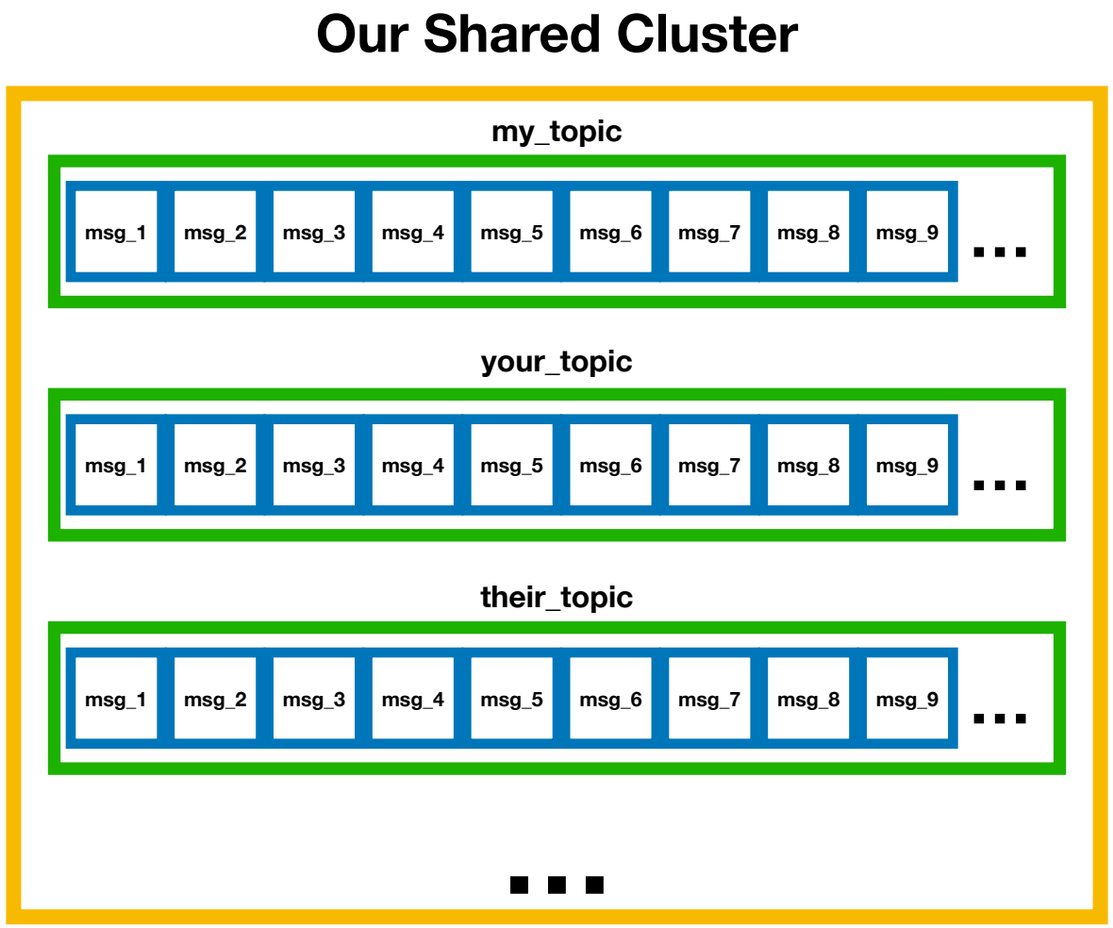
“message”
(some packet of binary data)

Apache Kafka, briefly



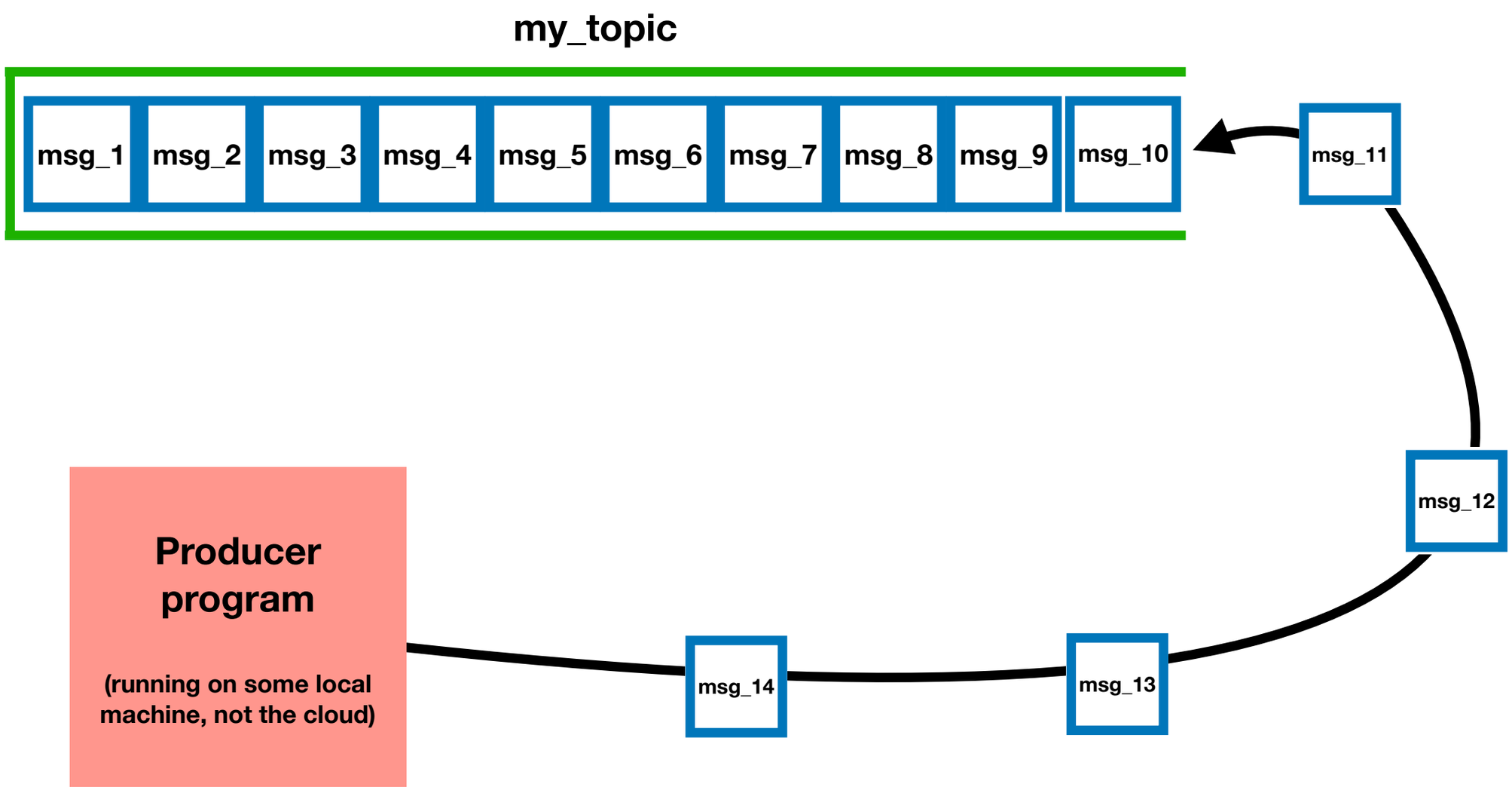
“topic”
(persistent, ordered, append-only log of messages)

Apache Kafka, briefly



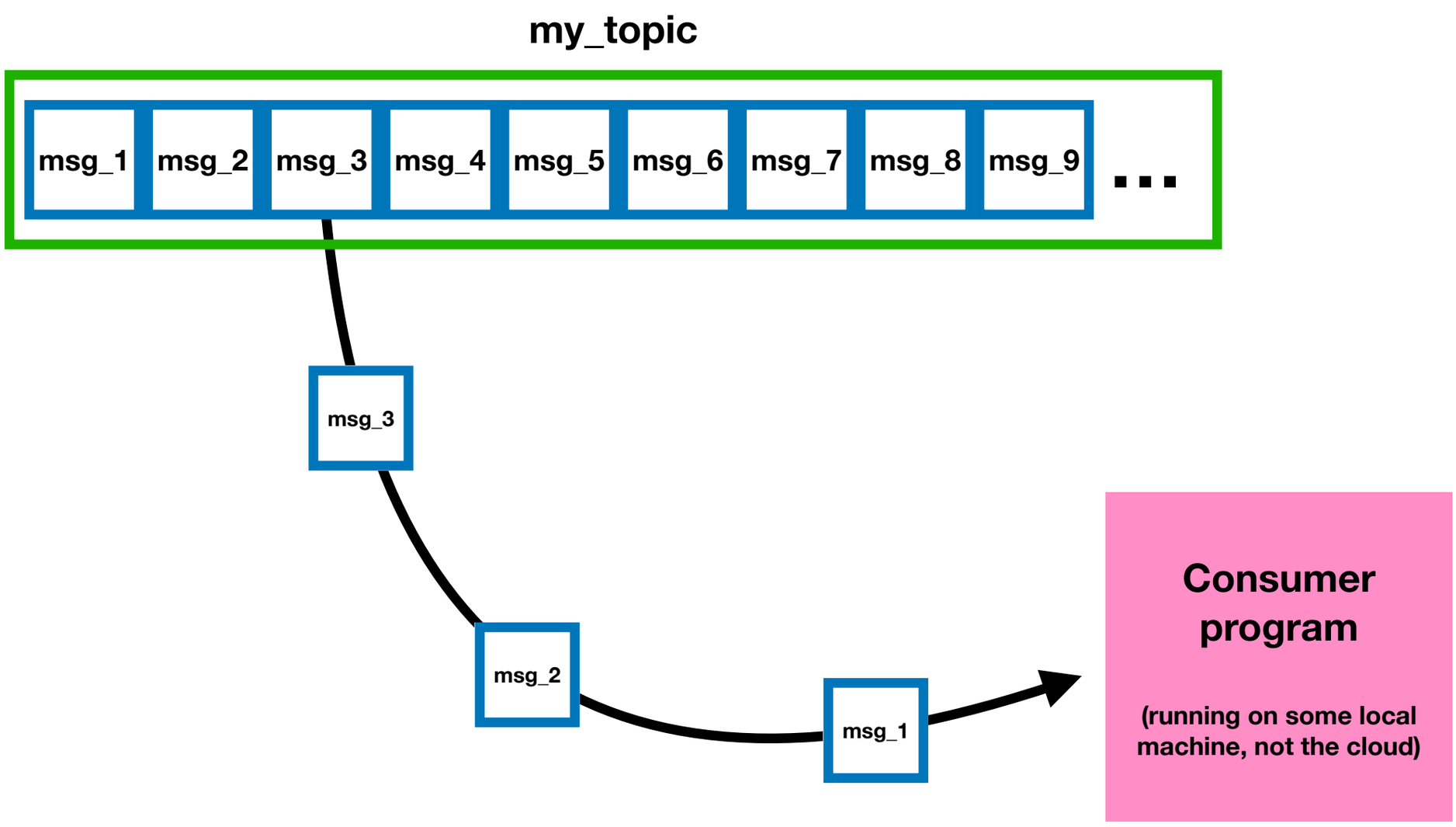
“cluster”
(collection of topics, plus bookkeeping/management)

Apache Kafka, briefly



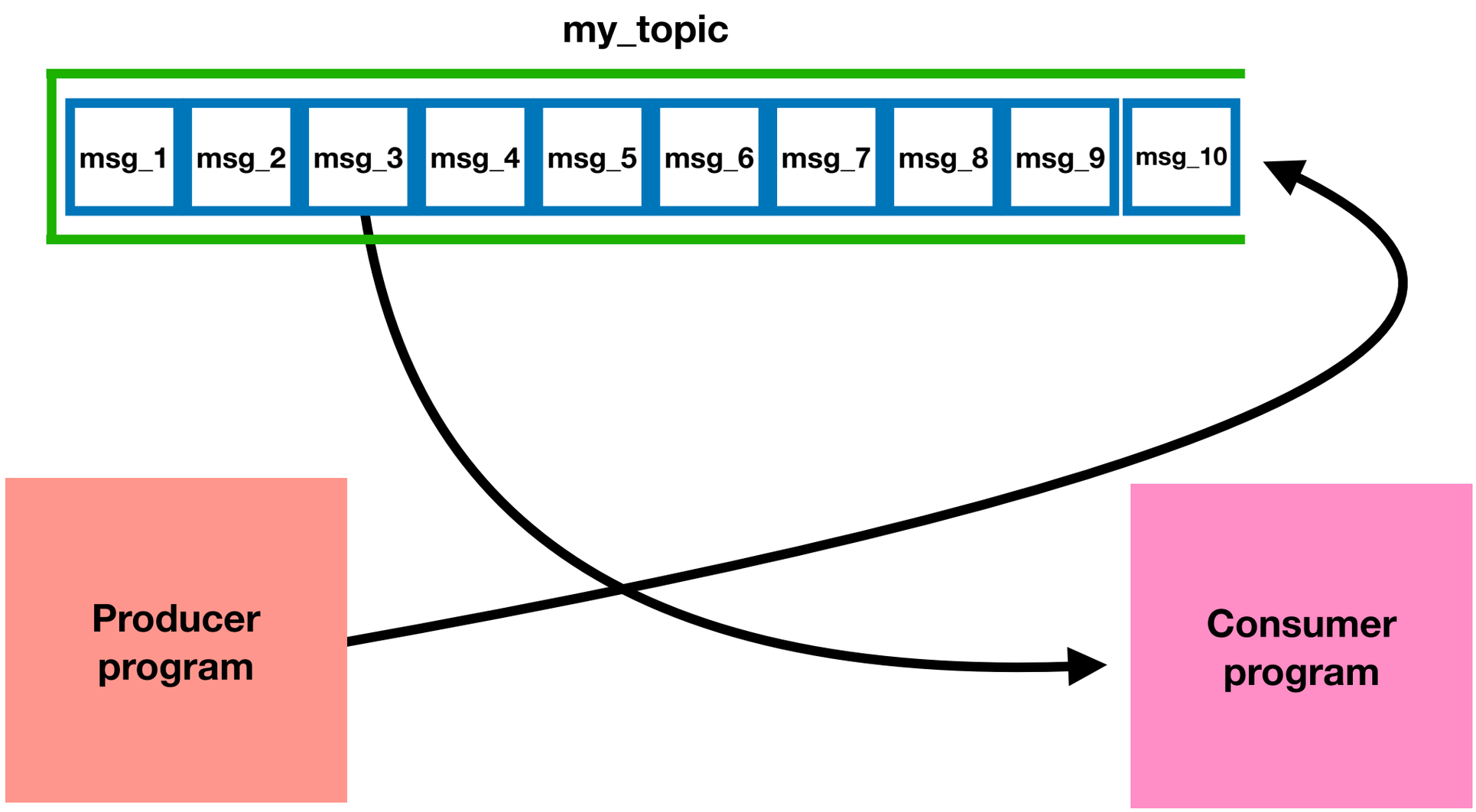
“Producers”
programs that add messages to topics

Apache Kafka, briefly



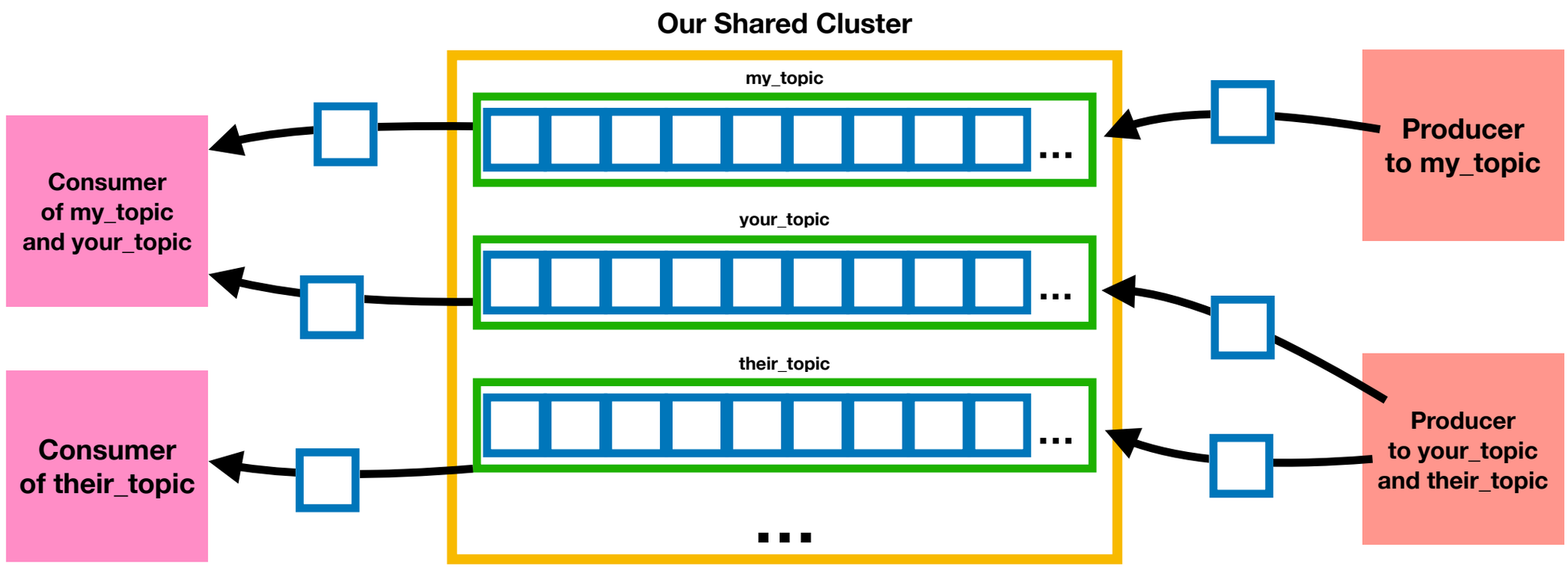
“Consumers”
programs that read the messages in topics

Apache Kafka, briefly



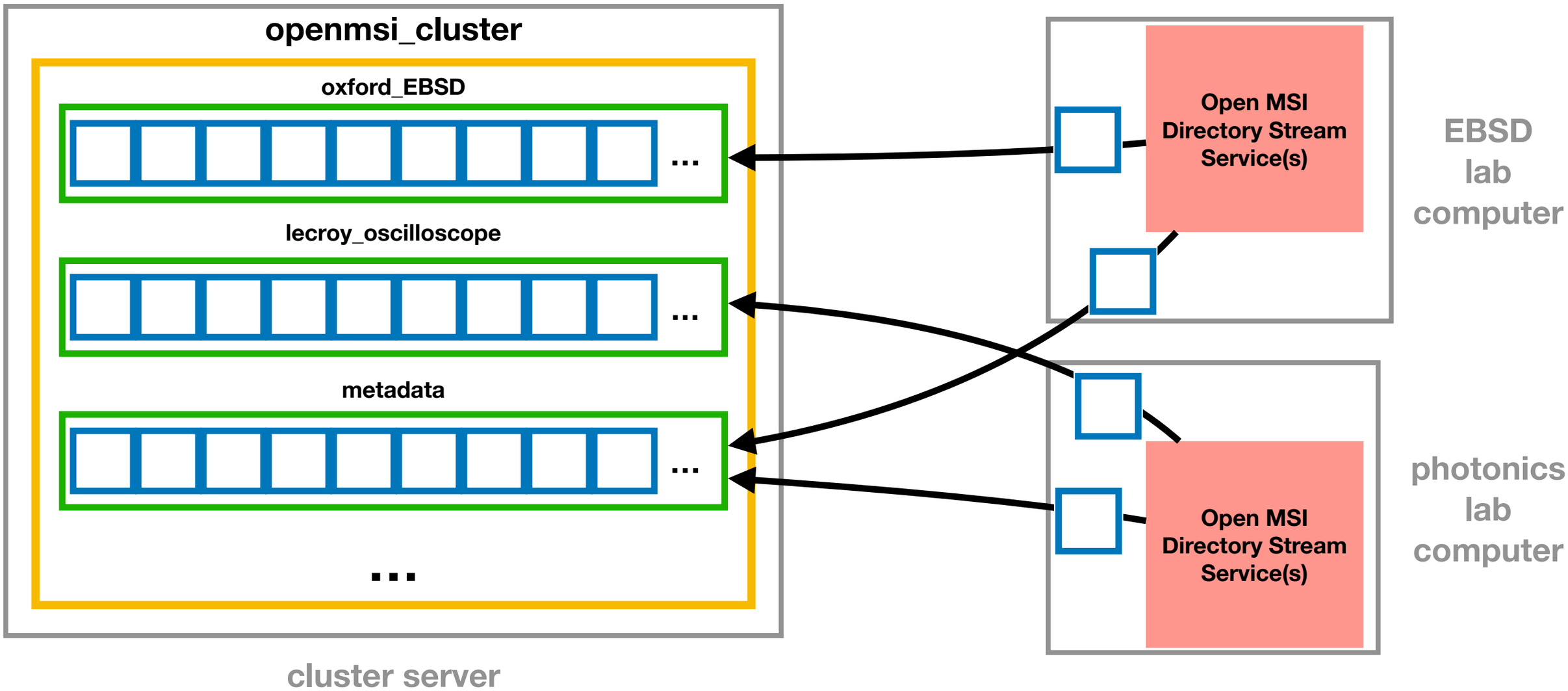
Producers and Consumers
are completely independent of one another
(this is a surprisingly powerful guarantee)

Apache Kafka, briefly



- An ecosystem of data flow
- Extremely up-to-date
- Persistent, safe, reliable, and flexible
- Using Apache Kafka to store, transfer, and (eventually) analyze data in real time will accelerate the pace of communication between groups that focus on disparate parts of larger projects

The Open MSI Directory Stream Service, Conceptually

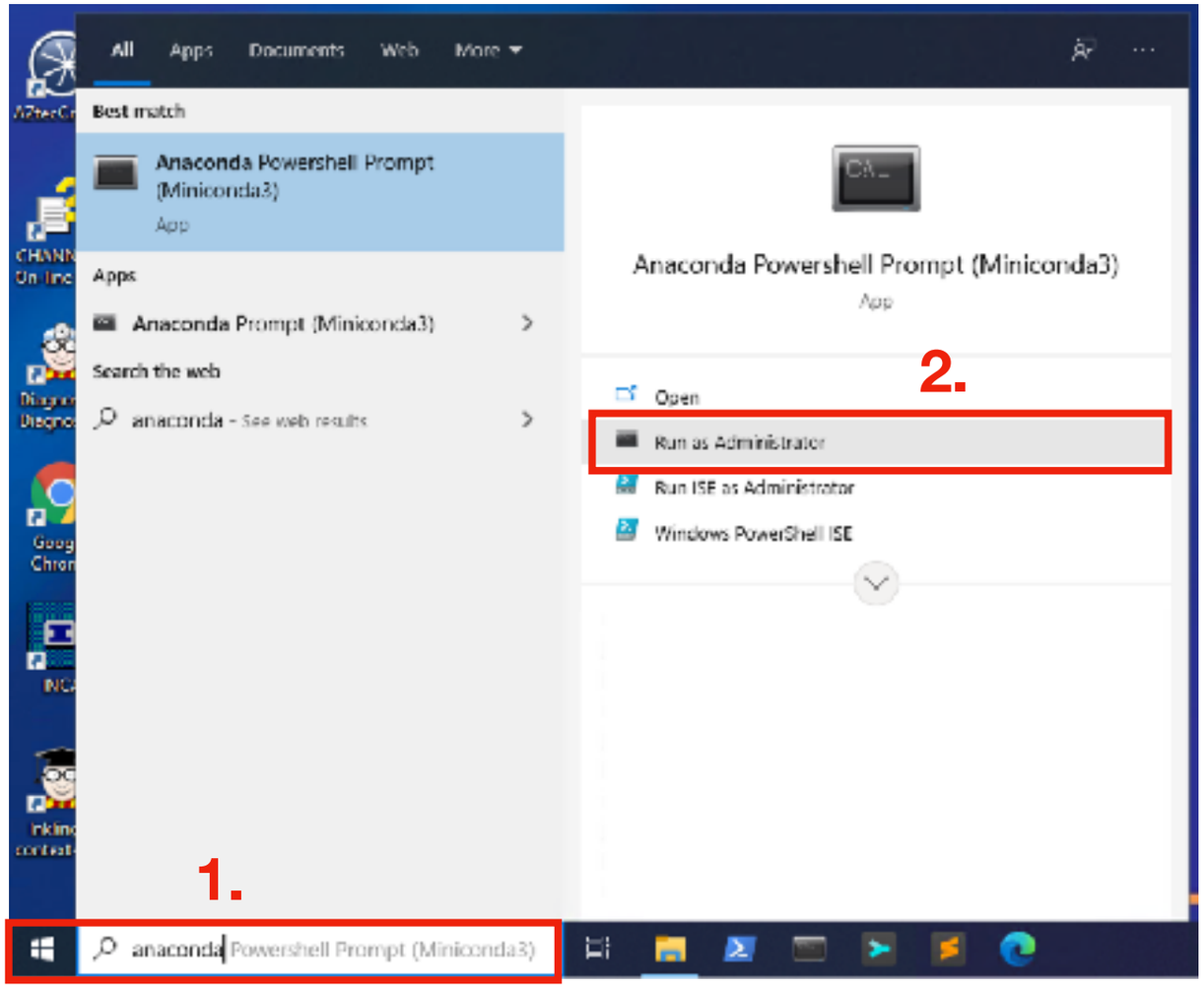


- Open MSI Directory Stream Service is a custom Producer program packaged as a Windows Service
- Open MSI Directory Stream Service gets installed on lab computers
    - Starts watching directories for new files to be added
    - Code will run for as long as the machine is booted, and won't stop unless you tell it to
    - Once installed by an admin account it will work on any other user account
- Data files added to watched directories are automatically uploaded to topic(s) you choose in the cluster
- Data files can be reconstructed on other machines, analyses can happen as soon as files are available, etc.

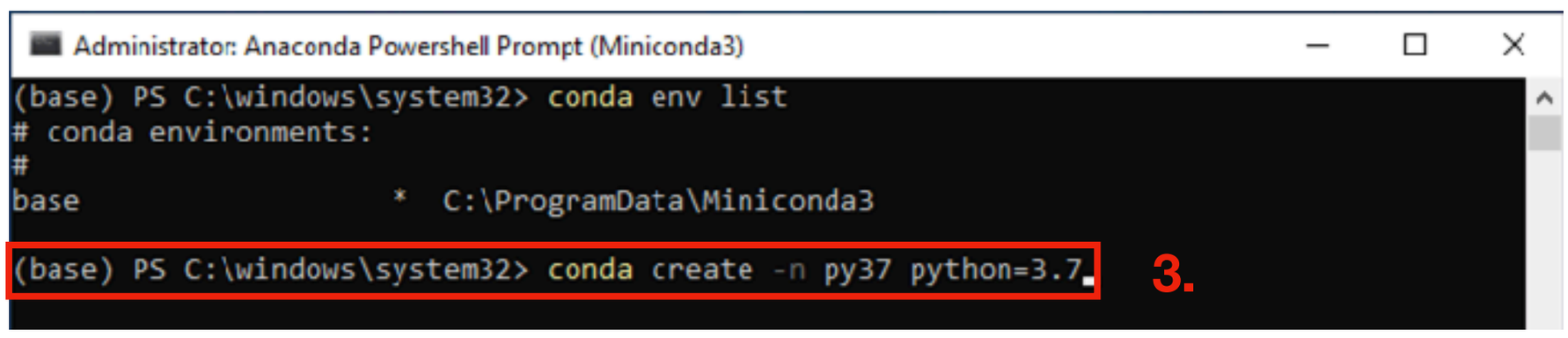
The Open MSI Directory Stream Service, Practically

- Need an environment (miniconda3 is an excellent option) with Python 3.7, Git, and `pip`
- Clone the Python_code Github repository
- Install the package in the repo using `pip`
- Write (or use) a small configuration file to tell the code:
  - which directory to watch for data files
  - which topic on the cluster to produce them to
- Install the Open MSI Directory Stream Service (managed using NSSM)
- Put data files into the watched directory while you work
  - Files you put in the directory will be produced to the topic on the cluster
  - The Service will start running when the machine is booted, and stay running until you tell it to stop
- Instructions on how to do the above (and more!) are available on the GitHub, and I'll walk you through it right now, too! (I have screenshots in these slides for reference, also.)

The Open MSI Directory Stream Service, Practically

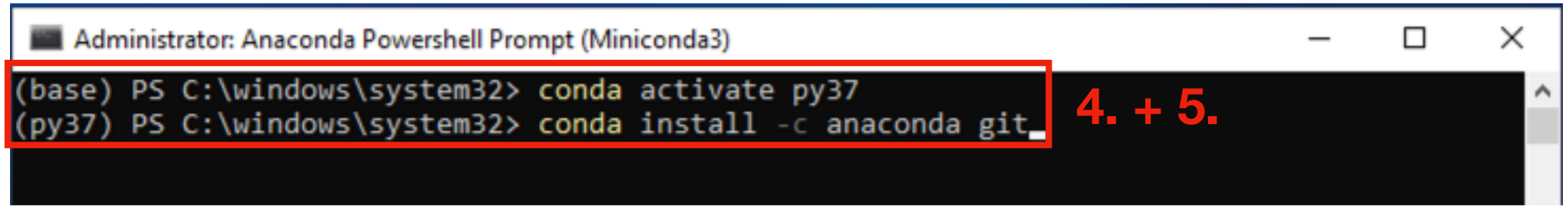


1. Search Start Menu* for Anaconda Powershell Prompt (Miniconda3)
2. Run as an Administrator
3. Create a new environment with Python 3.7

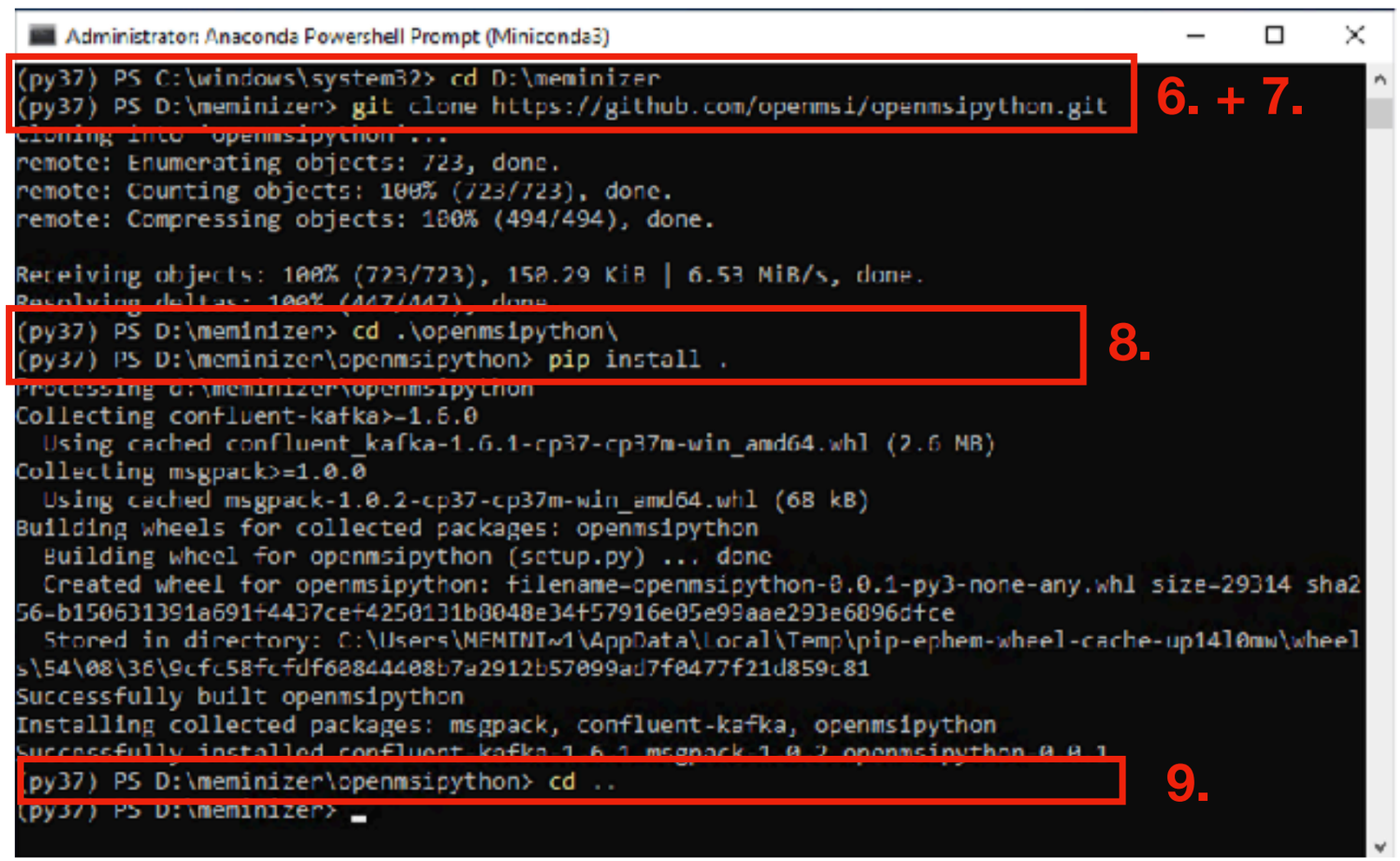


*If you don't already have Miniconda3 installed on the machine, you can get it here: https://docs.conda.io/en/latest/miniconda.html

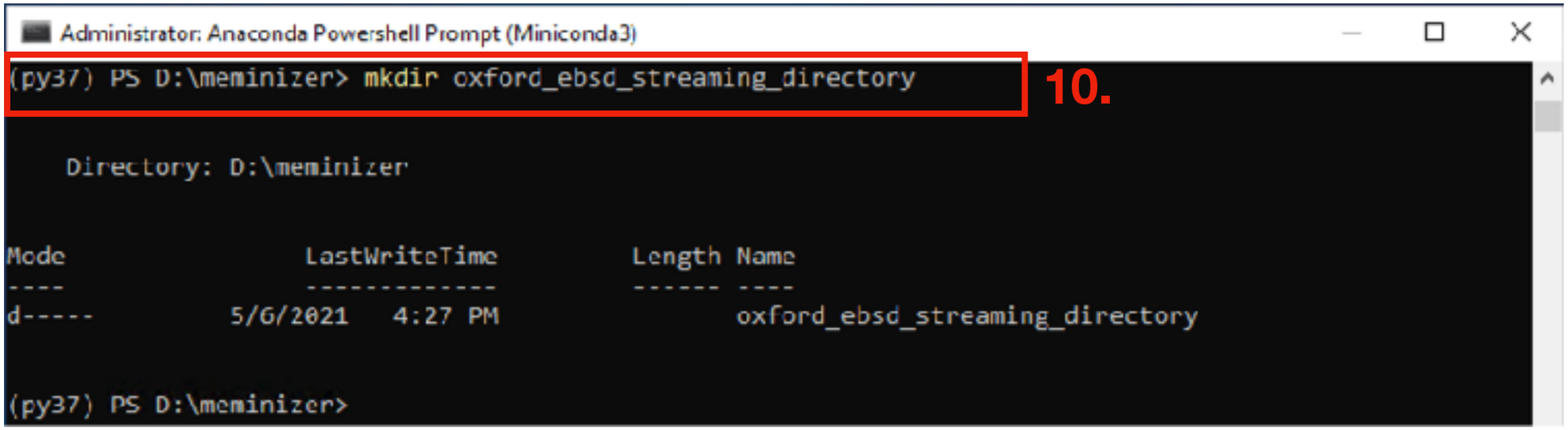
The Open MSI Directory Stream Service, Practically

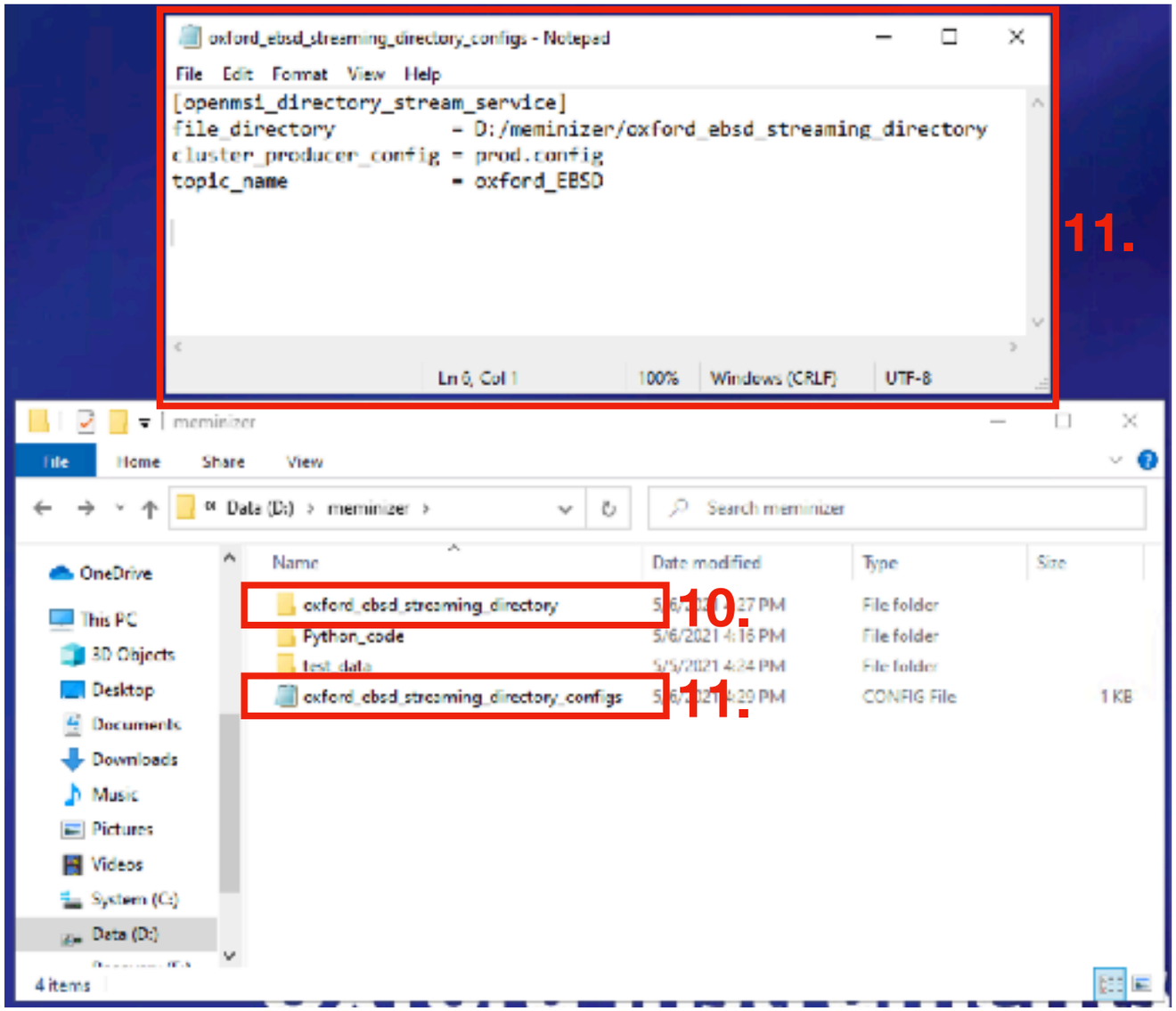


4. Switch to the Python3.7 environment
5. Install `git`
6. Navigate to where you want to store the code (anywhere is fine)
7. Clone the Github repo
8. Install the package in the repo with `pip`
9. Back out of the repo directory



The Open MSI Directory Stream Service, Practically

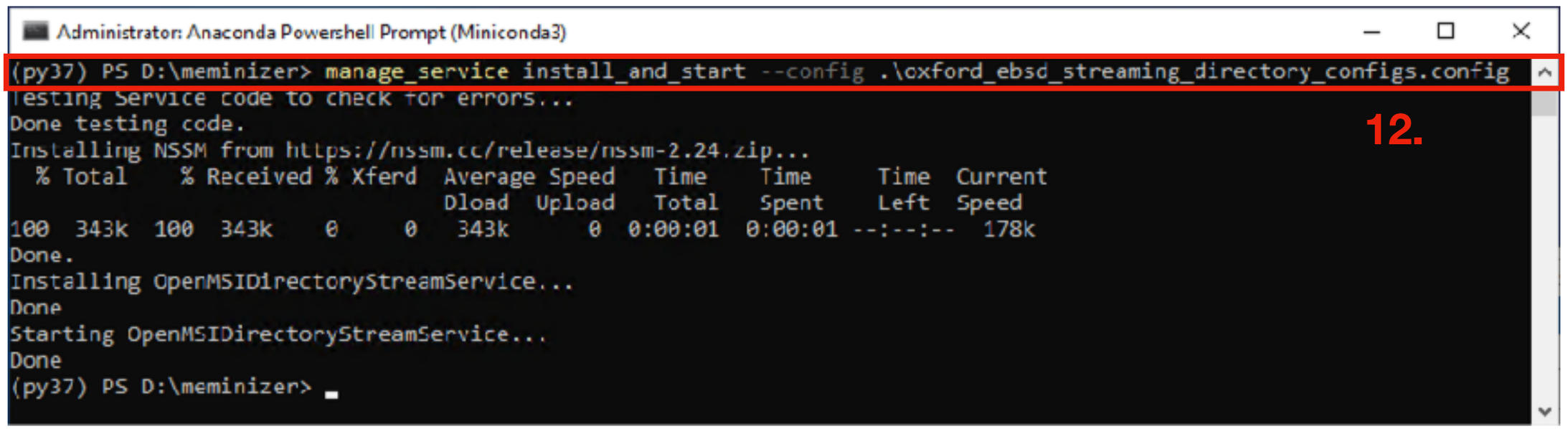




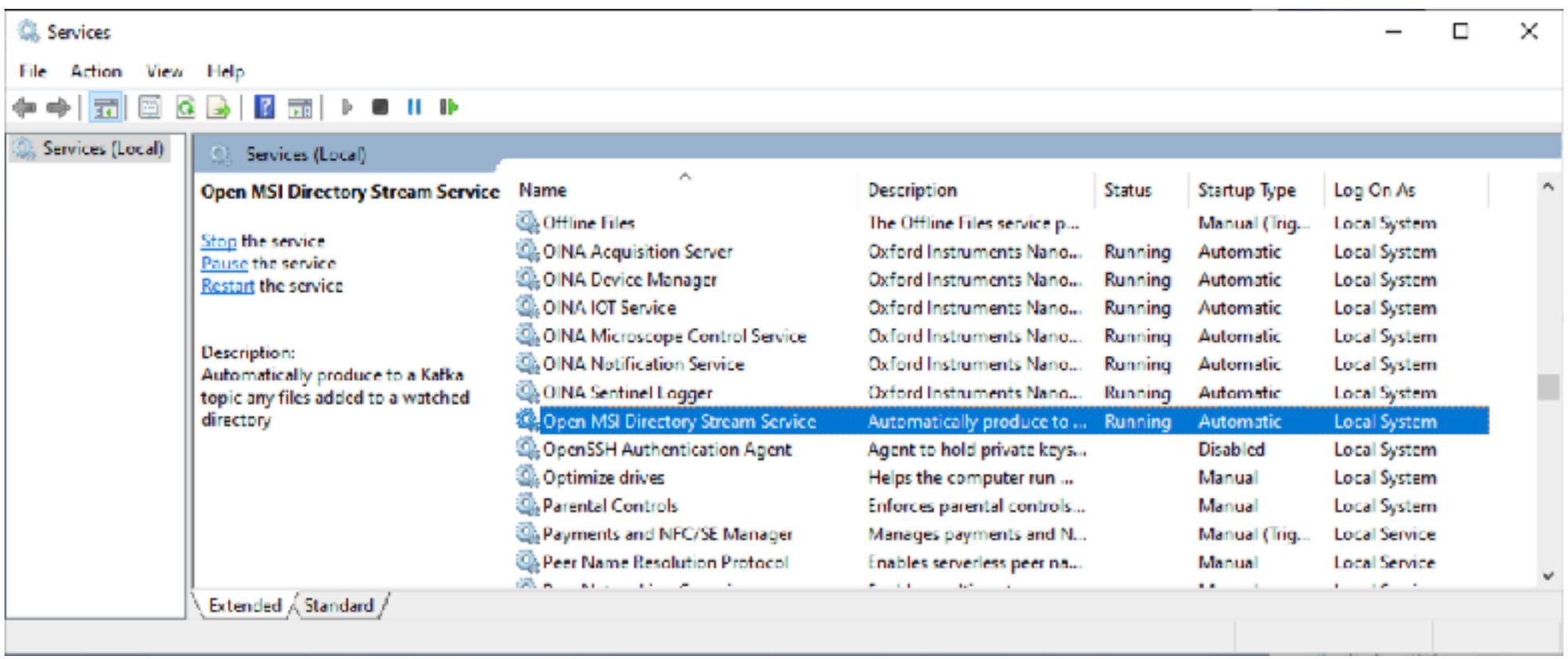
10. Create a directory to be watched for new data files being added

11. Set up a config file pointing to that directory

The Open MSI Directory Stream Service, Practically

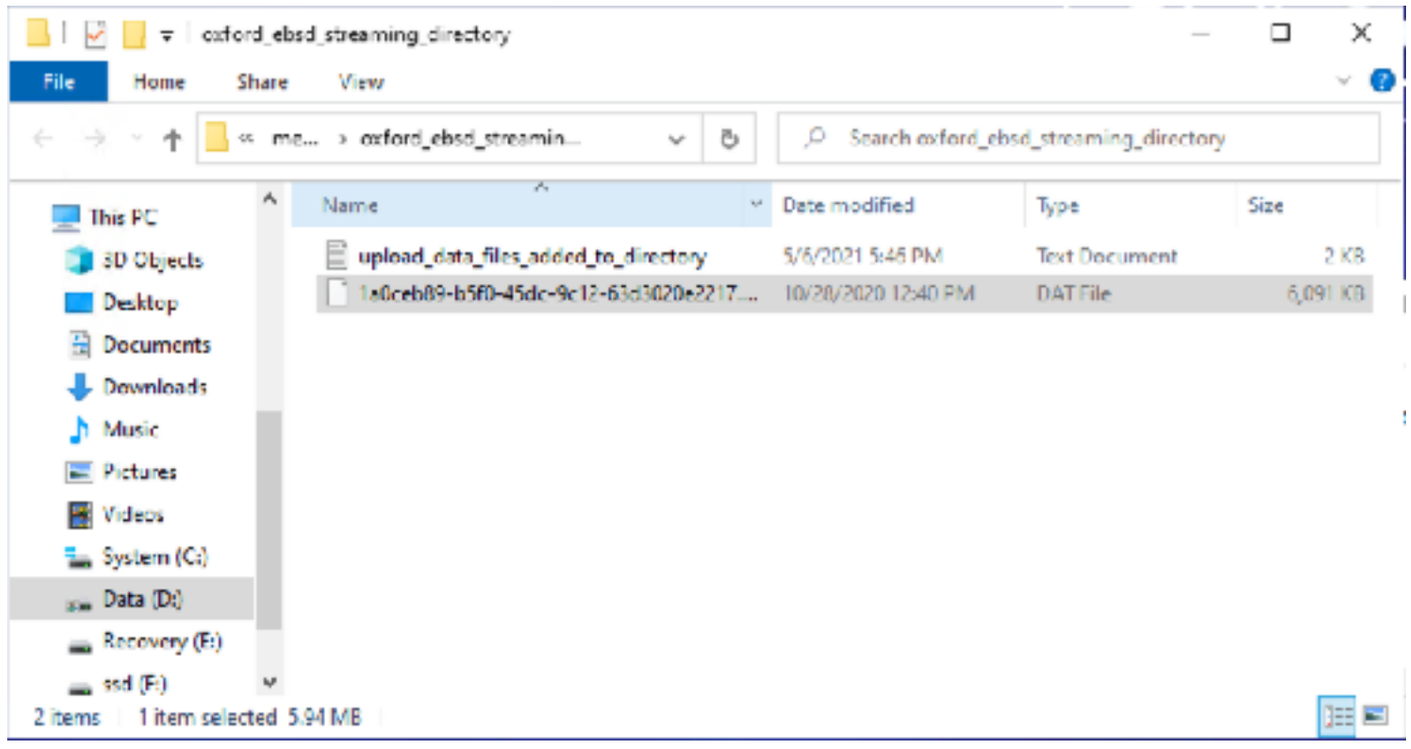


12. Install and start the Service running



The Open MSI Directory Stream Service, Practically

13. Add files to the directory you created :)




```
(py37) zinc20 17:50:55 11 ~/dmref
> reconstruct_data_files test_ebsd_reco --config prod --topic_name oxford_EBSD
[reconstruct_data_files at 2021-05-06 17:52:25] Listening for files to reconstruct in /home/nemiriz1/dmref/test_ebsd_reco
[reconstruct_data_files at 2021-05-06 17:52:25] Will listen for files from the oxford_EBSD topic using 5 threads
[reconstruct_data_files at 2021-05-06 17:52:49] File 1a0ceb89-b5f0-45dc-9c12-63d3020e2217.dat successfully reconstructed locally from stream
[reconstruct_data_files at 2021-05-06 17:52:55] .
c
[reconstruct_data_files at 2021-05-06 17:53:04] File reconstructor writing to /home/nemiriz1/dmref/test_ebsd_reco shut down
[reconstruct_data_files at 2021-05-06 17:53:04] 301 total messages were consumed and the following 1 file was successfully reconstructed from 2021-05-06 17:52:
25.586799 to 2021-05-06 17:53:04.021270
        D:\meminizer\oxford_ebsd_streaming_directory\1a0ceb89-b5f0-45dc-9c12-63d3020e2217.dat
(py37) zinc20 17:53:04 12 ~/dmref
>
```