Hadoop Tutorials

Daniel Lanza

Zbigniew Baranowski

4 sessions

- Hadoop Foundations (today)
- **Data Ingestion (20-July)**
- Spark (3-Aug)
- Data analytic tools and techniques (31-Aug)

Hands-on setup

- 12 node virtualized cluster
  - 8GB of RAM, 4 cores per node
  - 20GB of SSD storage per node
- Access (haperf10[1-12].cern.ch)
  - Everybody who subscribed should have the access
  - Try: ssh haperf10* 'hdfs dfs -ls‘
- List of commands and queries to be used

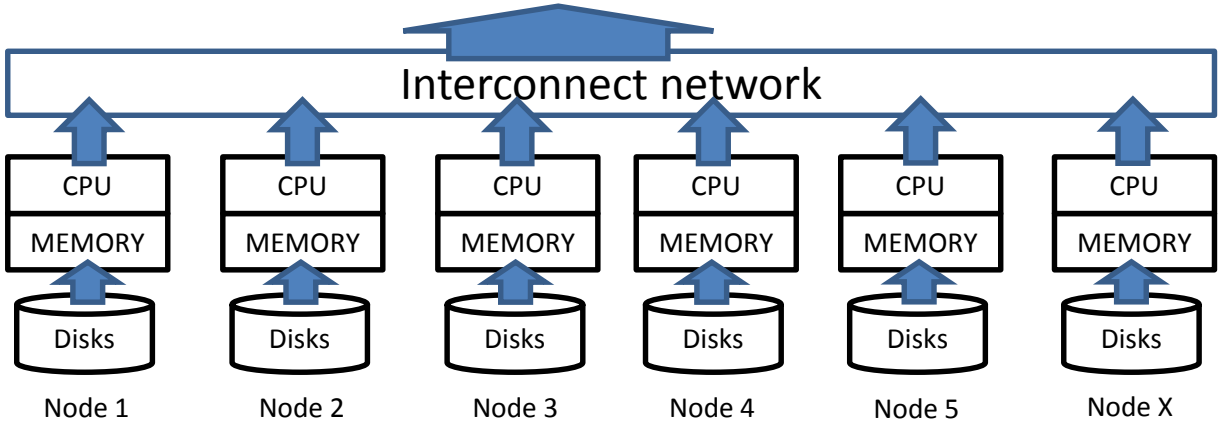
$> ssh haper10*
$> kinit
$> git clone https://:@gitlab.cern.ch:8443/db/hadoop-tutorials-2016.git
or
$> sh /tmp/clone-hadoop-tutorials-repo

- Alternative environment:

http://www.cloudera.com/downloads/quickstart_vms/5-7.html

Recap of the 1st session

- A framework for large scale data processing
- Data locality (shared nothing) – scales out



Recap of the 1st session

- Hadoop is distributed platform for large scale data processing
- For aggregations and reporting SQL is very handy – no need to be a Java expert
- In order to achieve good performance and optimal resources utilization
  - Partition your data – spread them across multiple directories
  - Use compact data formats - (Avro or Parquet)

Data Ingestion to Hadoop

Goals for today

- What are the challenges in storing data on Hadoop
- How to decrease data latency – ingestion in near-real-time
- How to ensure scalability and no data losses
- Learn about commonly used ingestion tools

What are the challenges?

- Variety of data sources
  - Databases
  - Web
  - REST
  - Logs
  - Whatever…

Files in batch

Streaming data

- Not all of them are necessary producing files…
- HDFS is a file system, not a database
  - You need to store files
- Extraction-Tranformation-Loading tools needed

Data ingestion types

- Batch ingestion
  - Data are already produced and available to store on Hadoop (archive logs, files produced by external systems, RDBMS)
  - Typically big chunks of data
- Real time ingestion
  - Data are continuously produced
  - Streaming sources

Batch ingestion

- **hdfs dfs –put** or HDFS API
  - sends file from local system to HDFS
  - file is sent sequentially
- Custom programs with using HDFS API
- **Kite SDK**
  - sends (text) files and encodes in Avro, Parquet or store in HBase
  - multithreaded
- **Apache Sqoop** – loading data from external relational databases

About Kite

- High level data API for Hadoop
- Two steps to store your date
  - Create **dataset** - configure how to store the data
    - Data schema, partitioning strategy
    - File format: JSON,Parquet, Avto, Hbase
    - dataset metadata: on HDFS or in Hive (as a table)
  - Import the data
    - From local file system, or HDFS

KiteSDK – Hand-on

- Loading a CSV data to HDFS into parquet format

0) get a CSV data (**hadoop-tutorials-2016/2_data_ingestion/0_batch_ingestion/kite**)

```
hdfs dfs –get /tmp/ratings.csv .
```

1) infer schema from the data (script: ./1_get_schema)

```
$ kite-dataset csv-schema ratings.csv --record-name ratings -o ratings.avsc
$ cat ratings.avsc
```

2) create data partitioning policy (script: ./2_create_part_file)

```
$ echo “[
  {"type": "year", "source": "timestamp"}
]”>> partition.policy.json
```

3) create a datastore on HDFS (script: ./3_create_datastore)

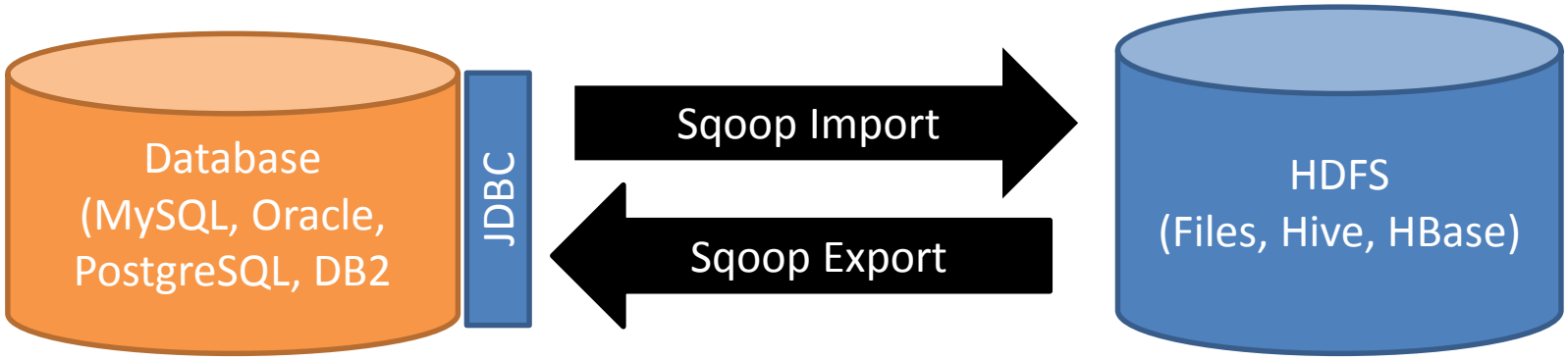
```
$  kite-dataset create dataset:hdfs:/user/zbaranow/datasets/ratings –schema \ ratings.avsc --format parquet --partition-by partition.policy.json
```

4) load the data (script: ./4_load_data)

```
$ kite-dataset csv-import ratings.csv --delimiter ',' \
dataset:hdfs:/user/zbaranow/datasets/ratings
```

About Apache Sqoop

- Tool to transfer data between structured databases and Hadoop



- Sqoop tasks are implemented as **map reduce** jobs – can scale on the cluster

Tips for Sqoop

- For big data exports (>1G) use dedicated connectors for target database
  - Logic that understand how to read data efficiently from target db system
- Do not use too many mappers/sessions
  - Excessive number of concurrent sessions can kill a database
  - max 10 mappers/sessions
- Remember: Sqoop does not retransfer (automatically) updated data

How to run a Sqoop job

Example (target Oracle):

```
sqoop import \                  #impoting from DB to HDFS
--direct
--connect jdbc:oracle:thin:@itrac5110:10121/PIMT_RAC51.cern.ch \
--username meetup \  #database user
--table meetup_data #table name to be imported
-P \
--num-mappers 2 \                  #number of parallel sessions
--target-dir meetup_data_sqooop \    #target HDFS directory
```

Sqoop hands-on

- Run a sqoop job to export meetup data from Oracle database
  - in a text format

```
cd hadoop-tutorials-2016/2_data_ingestion/0_batch_ingestion/sqoop
./1_run_sqoop_import
```

  - (custom) incremental import to parquet,

    !build-in incremental import jobs does not support direct connectors

```
./2_create_sqoop_job
./3_run_job
```

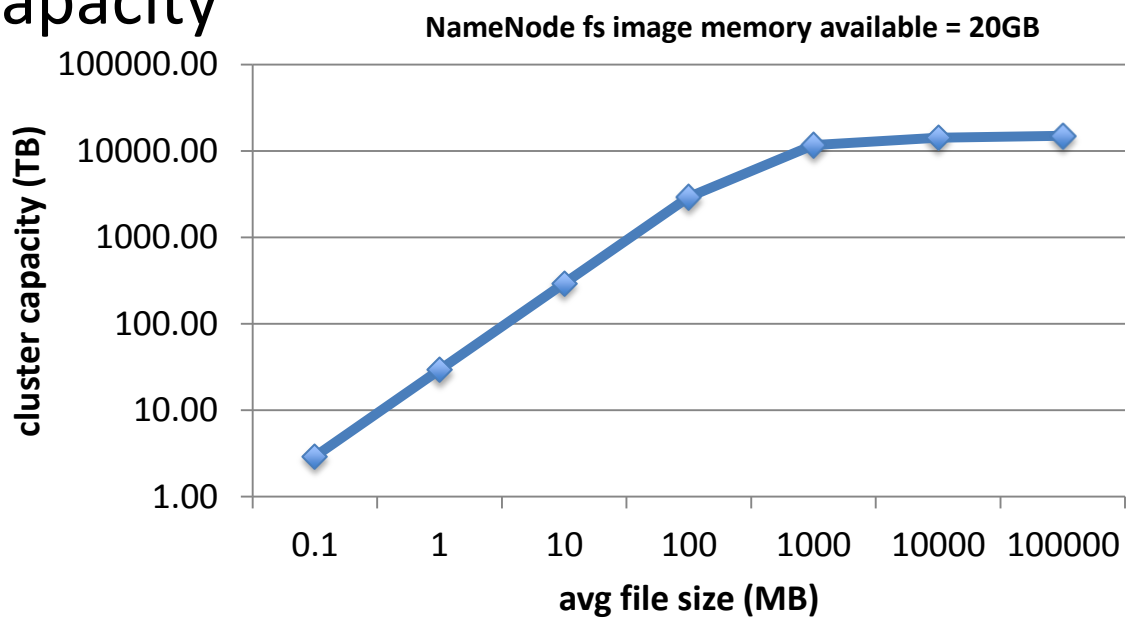
‘Real’ time ingestion to HDFS

- More challenging than batch ingestion
  - ETL on fly
- There is always a **latency** when storing to HDFS
  - data streams has to be materialized in files
  - creating a file per a stream event will kill HDFS
  - events has to be written in groups

What are the challenges ?

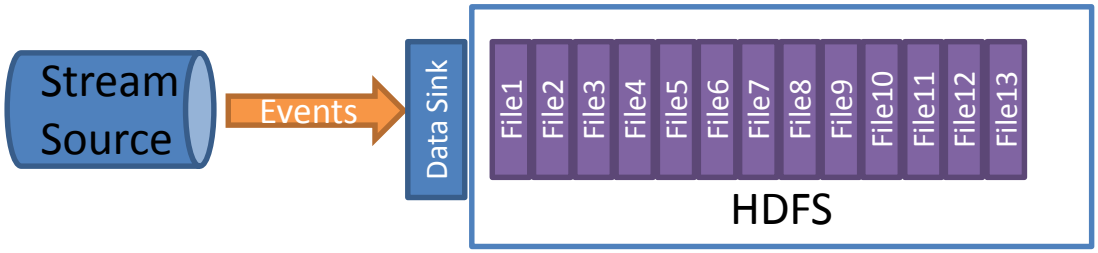
- Hadoop works efficiently with big files
  - (> a block size=128MB)
  - placing big data in small files will degrade processing performance (typically one worker per file)
  - and reduce hdfs capacity

File meta size = ~125B
Directory meta size = ~155B
Block meta size = ~184B

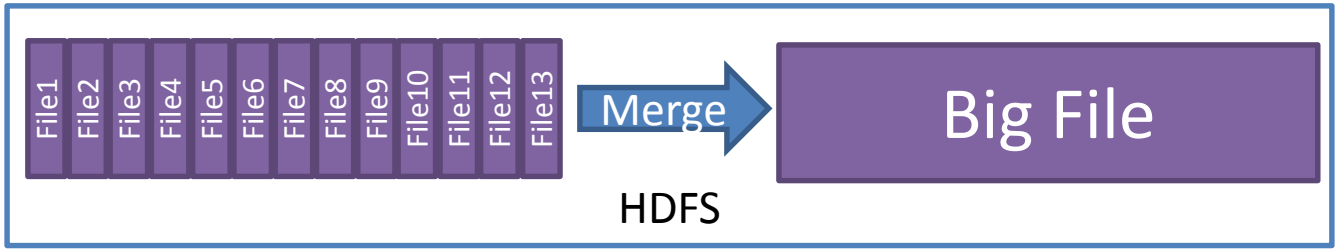


Solutions for low latency (1)

- 1) Writing small files to HDFS
  - Decreases data latency ☺
  - Creates many files ☹



- 2) Compacting them periodically



Compacting examples

- For text files (just merging with MR)

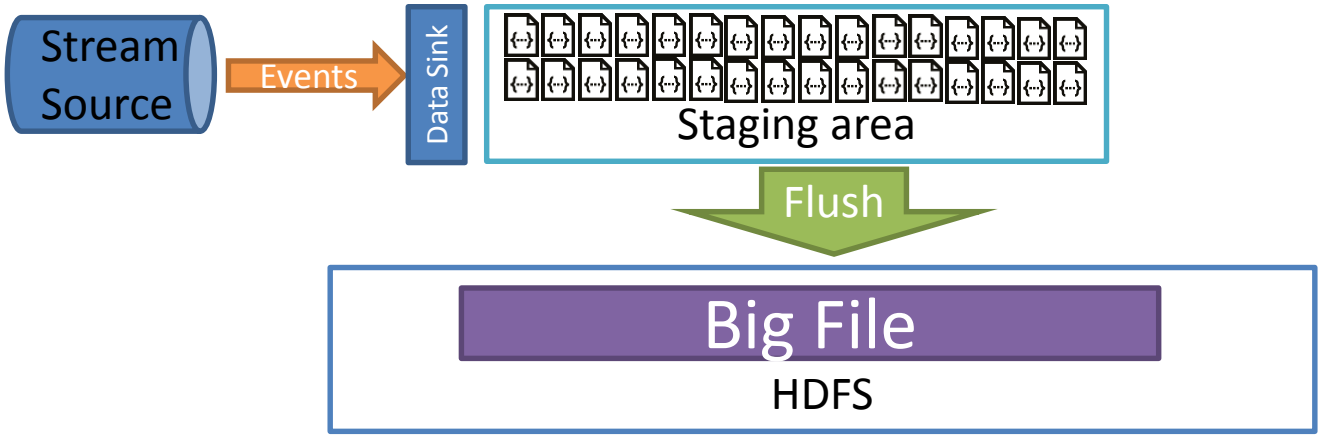
```
hadoop jar hadoop-streaming.jar \
 -Dmapred.reduce.tasks=1
-input meetup-data/2016/07/13  \
-output meetup-data/2016/07/13/merged \
   -mapper cat \
   -reducer cat
```

- For parquet/avro (using Spark)

```
val df = sqlContext.parquetFile("meetup-data/2016/07/13/*.parquet")
val dfp=df.repartition(1)
dfp.saveAsParquetFile("meetup-data/2016/07/13/merge")
```

Solutions for low latency (2)

- 1) Stage data into staging buffers
  - Making them immediately accessible to access



- 2) Flush buffers periodically to HDFS files
  - Requires two access path to the data (buffers + HDFS)

How to sink data streams to HDFS

- There are specialized tools
  - **Apache Flume**

  - LinkedIn Gobblin

  - Apache Nifi

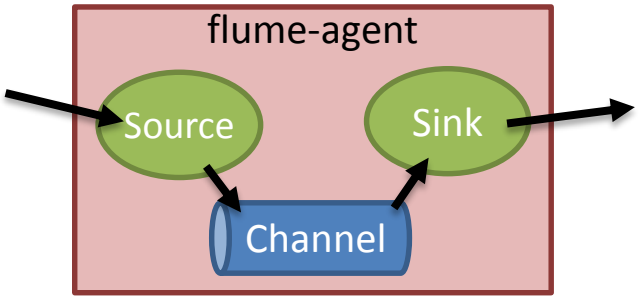
  - and more

Apache Flume

- Data transfer engine driven by events
  - Flume events
    - Headers
    - Body (byte array)
- Data can be
  - Collected
  - Processed (interceptors)
  - Aggregated
- Main features
  - Distributed
    - Agents can be placed on different machines
  - Reliable
    - Transactions

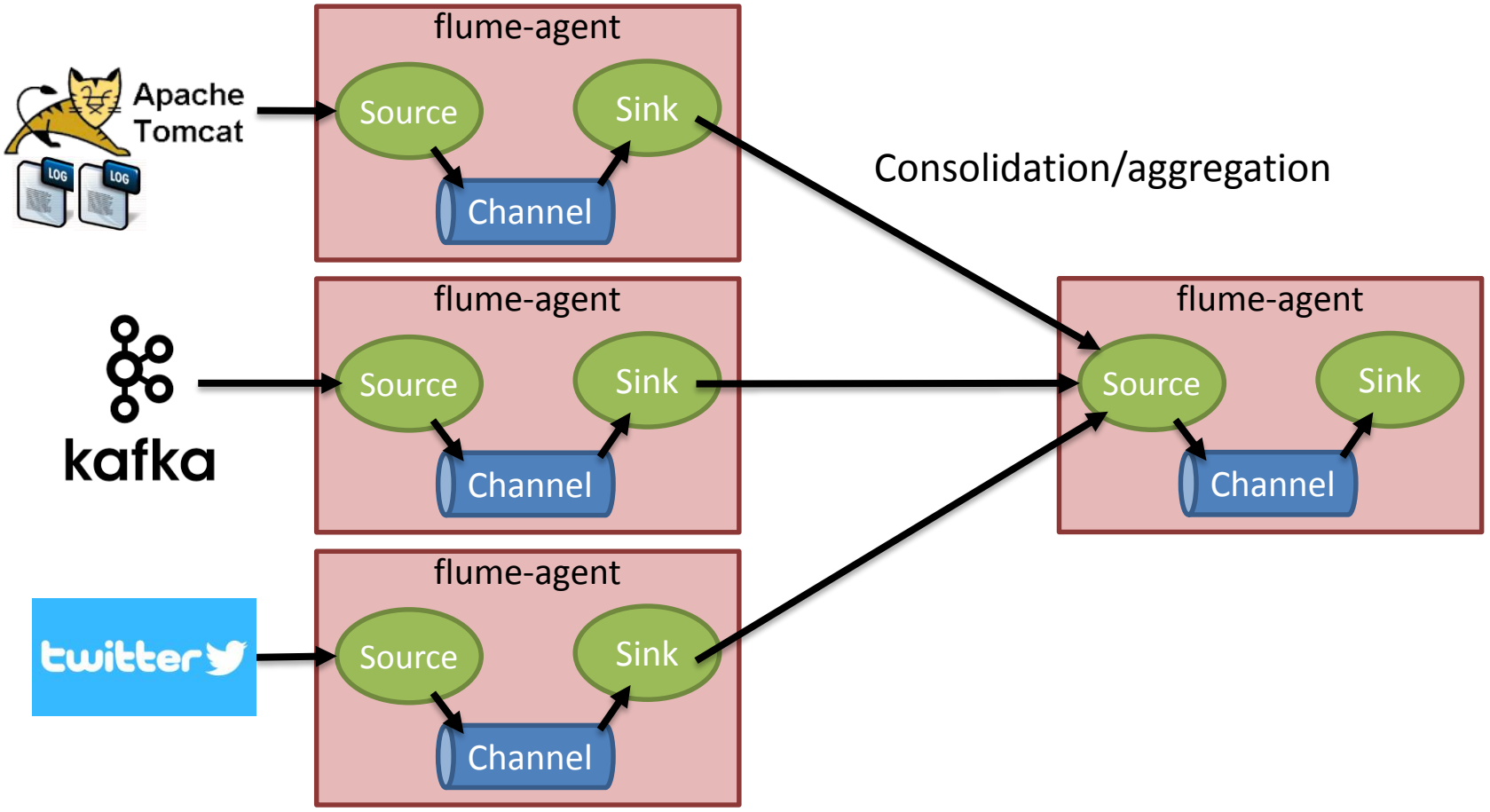
Flume agent

- Agents has at least
  - A source
    - Files/Logs/Directories, Kafka, Twitter
    - STDOUT from a program
    - …
    - Custom (a.g. JDBCSource)
    - Note: could have interceptors
  - A channel
    - Memory
    - File
    - Kafka
    - JDBC
    - …
    - Custom
  - A sink
    - HDFS, HBase,
    - Kafka, ElasticSearch
    - ….
    - Custom



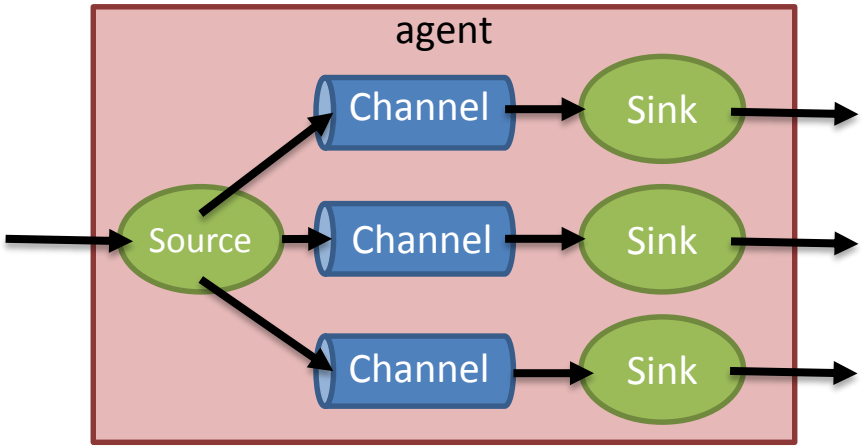
Flume data flow

- Multiple agents can be deployed
  - On the same machine or in a distributed manner



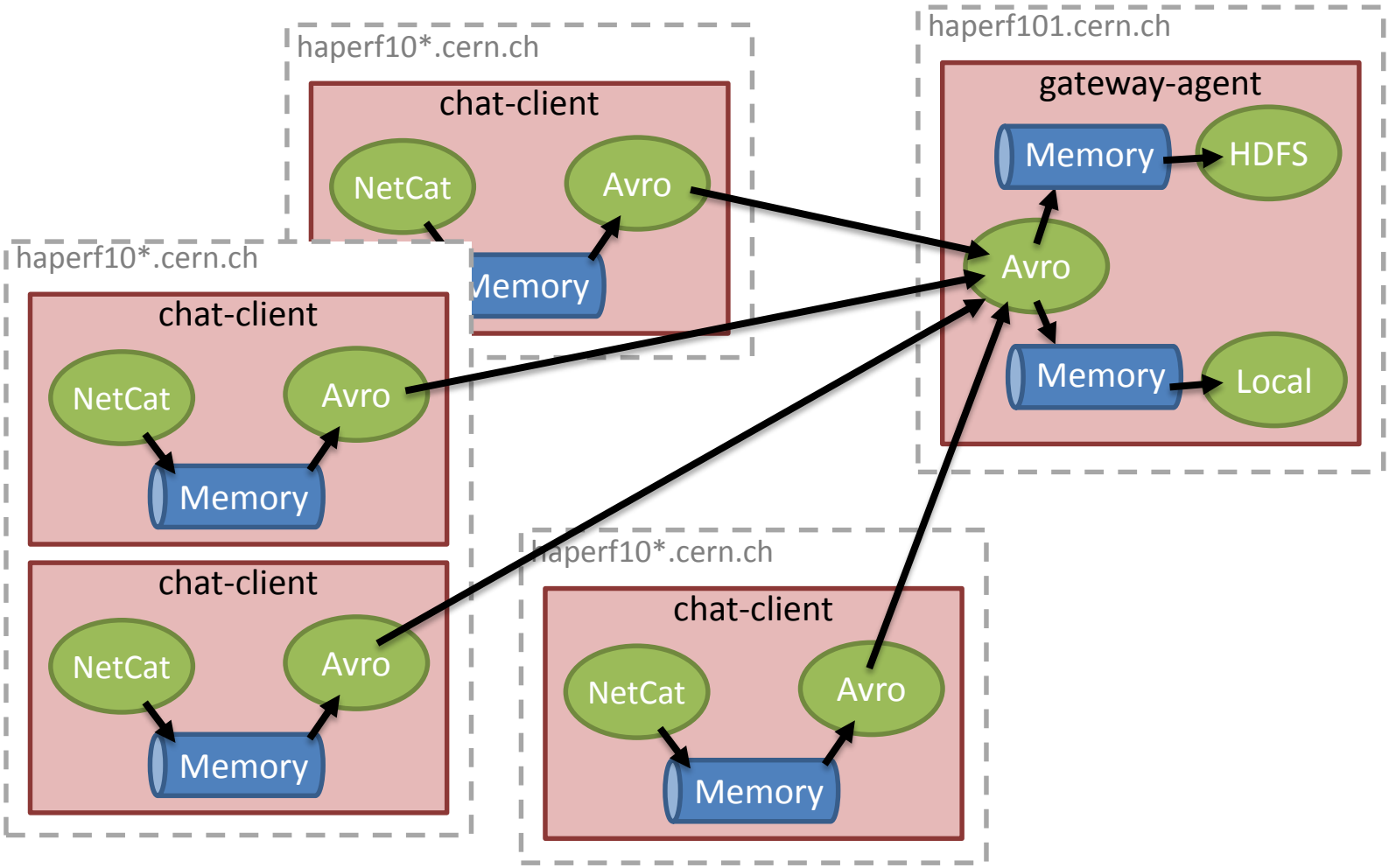
Flume data flow

- Agents can have more than one data flow
  - Replication
  - Multiplexing



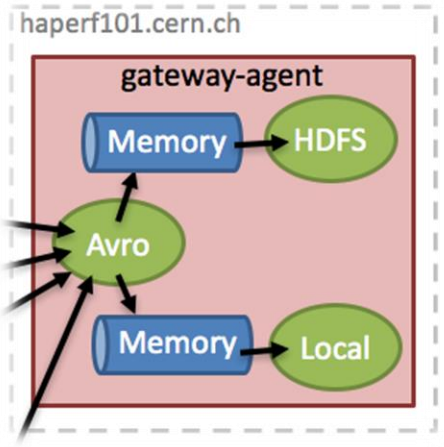
Flume hands-on

- Flume chat



Flume hands-on

- Flume chat gateway




```
# Name the components on this agent
gateway-agent.sources = avro_source
gateway-agent.channels = hdfs_channel local_channel
gateway-agent.sinks = hdfs_sink local_sink

# Configure source
gateway-agent.sources.avro_source.type = avro
gateway-agent.sources.avro_source.selector.type = replicating
gateway-agent.sources.avro_source.channels = hdfs_channel local_channel
gateway-agent.sources.avro_source.bind = 0.0.0.0
gateway-agent.sources.avro_source.port = 12123
gateway-agent.sources.avro_source.interceptors = addtimestamp
gateway-agent.sources.avro_source.interceptors.addtimestamp.type = AddTimestampInterceptor$Builder

# Use a channel which buffers events in memory
gateway-agent.channels.hdfs_channel.type = memory
gateway-agent.channels.hdfs_channel.capacity = 10000
gateway-agent.channels.hdfs_channel.transactionCapacity = 100

gateway-agent.channels.local_channel.type = memory
gateway-agent.channels.local_channel.capacity = 10000
gateway-agent.channels.local_channel.transactionCapacity = 100

# Describe the HDFS sink
gateway-agent.sinks.hdfs_sink.type = hdfs
gateway-agent.sinks.hdfs_sink.channel = hdfs_channel
gateway-agent.sinks.hdfs_sink.hdfs.fileType = DataStream
gateway-agent.sinks.hdfs_sink.hdfs.path = hdfs://haperf100.cern.ch:8020/tmp/flume-chat-data/
gateway-agent.sinks.hdfs_sink.hdfs.rollCount = 100
gateway-agent.sinks.hdfs_sink.hdfs.rollSize = 1000000

# Describe the local files sink
gateway-agent.sinks.local_sink.type = file_roll
gateway-agent.sinks.local_sink.channel = local_channel
gateway-agent.sinks.local_sink.sink.directory = /tmp/flume-chat-data/
gateway-agent.sinks.local_sink.batchSize = 10
```

Flume hands-on

- Flume chat gateway (AddTimestampInterceptor)

```
public class AddTimestampInterceptor implements Interceptor {

    public static String DATE_FORMAT = "dd/MM/yyyy HH:mm:ss";

    private DateFormat format;

    @Override
    public void initialize() {
        format = new SimpleDateFormat(DATE_FORMAT);
    }

    @Override
    public Event intercept(Event event) {
        String body = new String(event.getBody());

        String body_with_timestamp = "[" + format.format(new Date()) + "] " + body;

        return EventBuilder.withBody(body_with_timestamp.getBytes(), event.getHeaders());
    }
```

Flume hands-on: chat gateway

- Clone repository and go to gateway directory

```
git clone https://:@gitlab.cern.ch:8443/db/hadoop-tutorials-2016.git
cd hadoop-tutorials-2016/2_data_ingestion/1_flume_chat_gateway/
```

- Compile and run it

Done by us on haperf101.cern.ch!

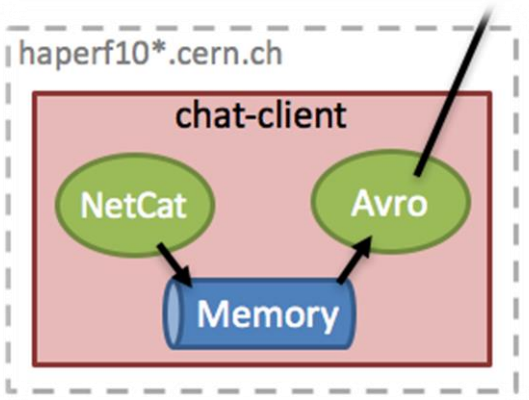
```
./compile
./run-agent
```

- After running client we can see the data

```
cat /tmp/flume-chat-data/*
hdfs dfs -cat "/tmp/flume-chat-data/*"
```

Flume hands-on

- Flume chat client




```
# Name the components on this agent
chat-client.sources = netcat_source
chat-client.channels = memory_channel
chat-client.sinks = avro_sink

# Configure source
chat-client.sources.netcat_source.type = netcat
chat-client.sources.netcat_source.channels = memory_channel
chat-client.sources.netcat_source.bind = 0.0.0.0
chat-client.sources.netcat_source.port = 1234
chat-client.sources.netcat_source.interceptors = adduser
chat-client.sources.netcat_source.interceptors.adduser.type = AddUserInterceptor$Builder

# Use a channel which buffers events in memory
chat-client.channels.memory_channel.type = memory
chat-client.channels.memory_channel.capacity = 1000
chat-client.channels.memory_channel.transactionCapacity = 100

# Describe the sink
chat-client.sinks.avro_sink.type = avro
chat-client.sinks.avro_sink.channel = memory_channel
chat-client.sinks.avro_sink.hostname = haperf101.cern.ch
chat-client.sinks.avro_sink.port = 12123
```

Flume hands-on

- Flume chat client (AddUserInterceptor)

```
public class AddUserInterceptor implements Interceptor {


    @Override
    public void initialize() {
        username = System.getProperty("user.name");
    }

    @Override
    public Event intercept(Event event) {
        String body = new String(event.getBody());

        String body_with_username = "(" + username + "): " + body;

        return EventBuilder.withBody(body_with_username.getBytes(), event.getHeaders());
    }
```

Flume hands-on: chat client

- Go to client directory (on any machine)

```
ssh haperf10*.cern.ch
cd hadoop-tutorials-2016/2_data_ingestion/1_flume_chat_client/
```

- Compile and run it

```
./compile
./run-agent
```

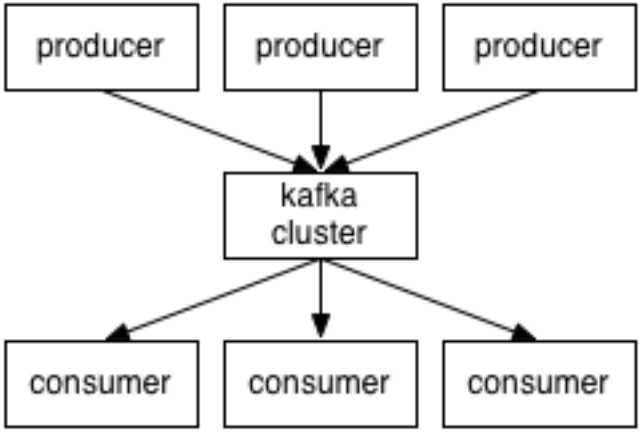
- Initialize chat (different terminal)

```
cd hadoop-tutorials-2016/2_data_ingestion/1_flume_chat_client/
./init_chat          # Ctrl + ] and quit to exit
```

Is Apache Flume enough?

- Yes, for simple use cases
- Performance limited by a single machine
  - Consolidating multiple sources -> requires a lot of resources
  - Multiplexing of sinks -> duplicating data in channels
- If HDFS in down for maintenance
  - Flume channel can be full quickly
- It does not provide **high availability**
  - flume agent machine is a single point of failure
  - if it breaks we will lose data
- Solution?
- Stage data in a reliable distributed event broker
- **Apache Kafka**

Apache Kafka

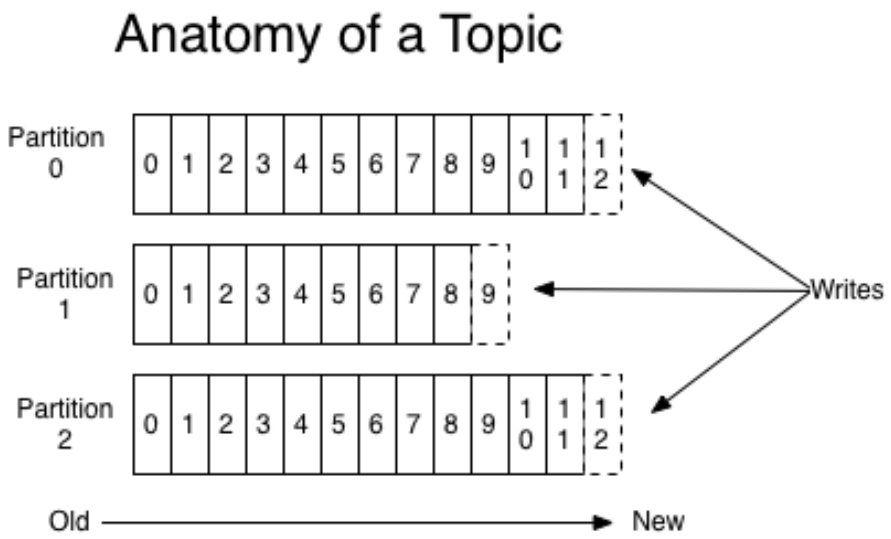


- Messages broker
  - Topics

- Main features
  - Distributed
    - Instances can be deployed on different machines
  - Scalable
    - Topic could have many partitions
  - Reliable
    - Partitions are replicated
    - Messages can be acknowledged

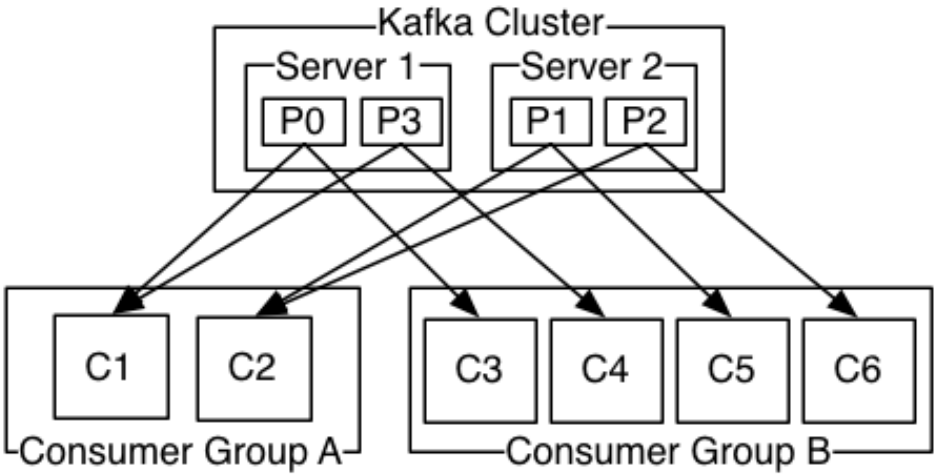
Apache Kafka

- Topics
  - Partitions
    - Replicated
    - One is the leader
    - Message written depending on the message key
    - Data retention can be limited by size or time



Apache Kafka

- Consumer groups
  - Offset is kept in Zookeeper

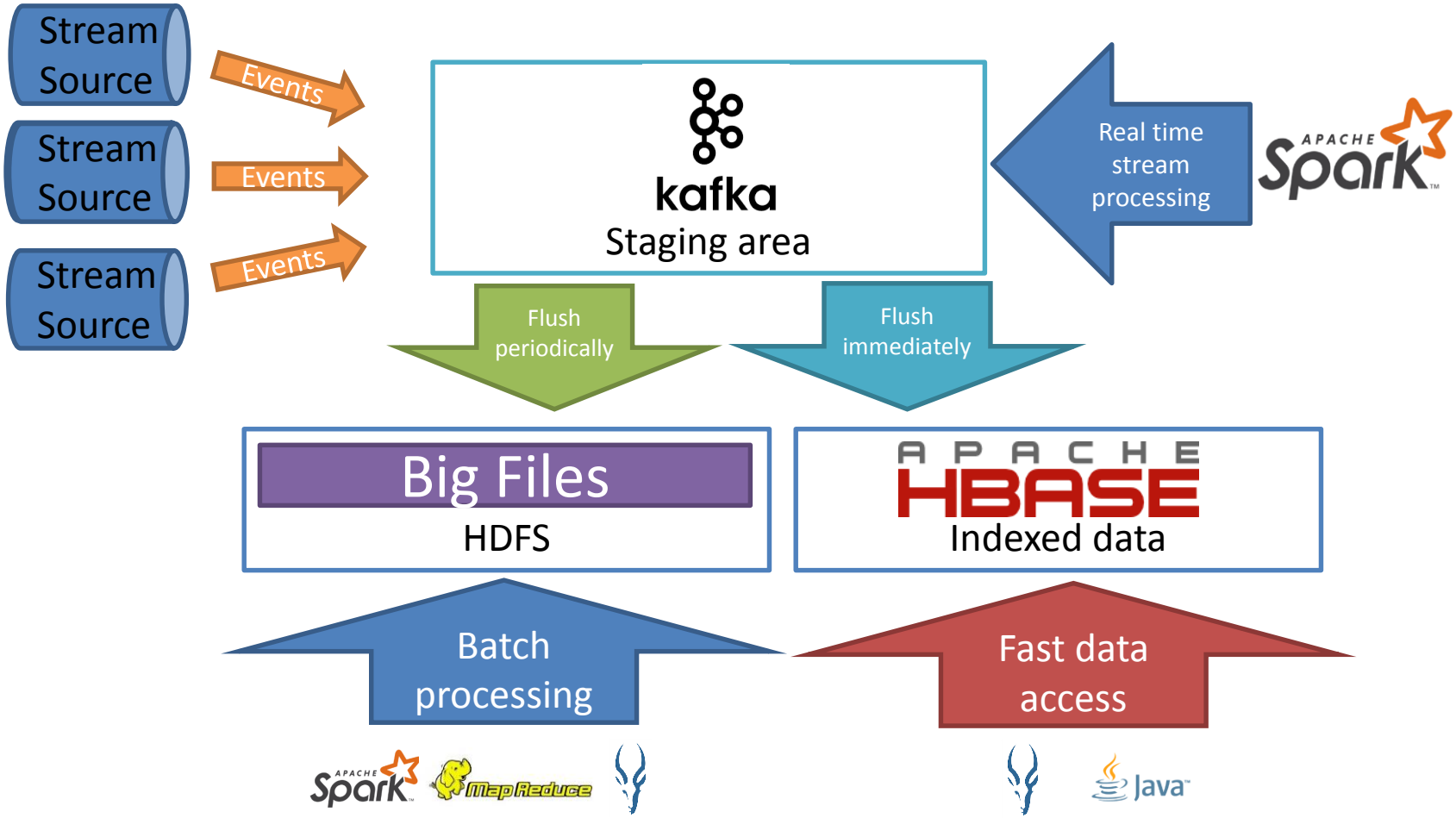


Apache Kafka – how to use it

- Flume out-of-the-box can use Kafka as
  - Source, Channel, Sink
- Other ingestion or processing tools support Kafka
  - Spark, Gobblin, Storm…
- Custom implementation of producer and consumer
  - Java API, Scala, C++, Python

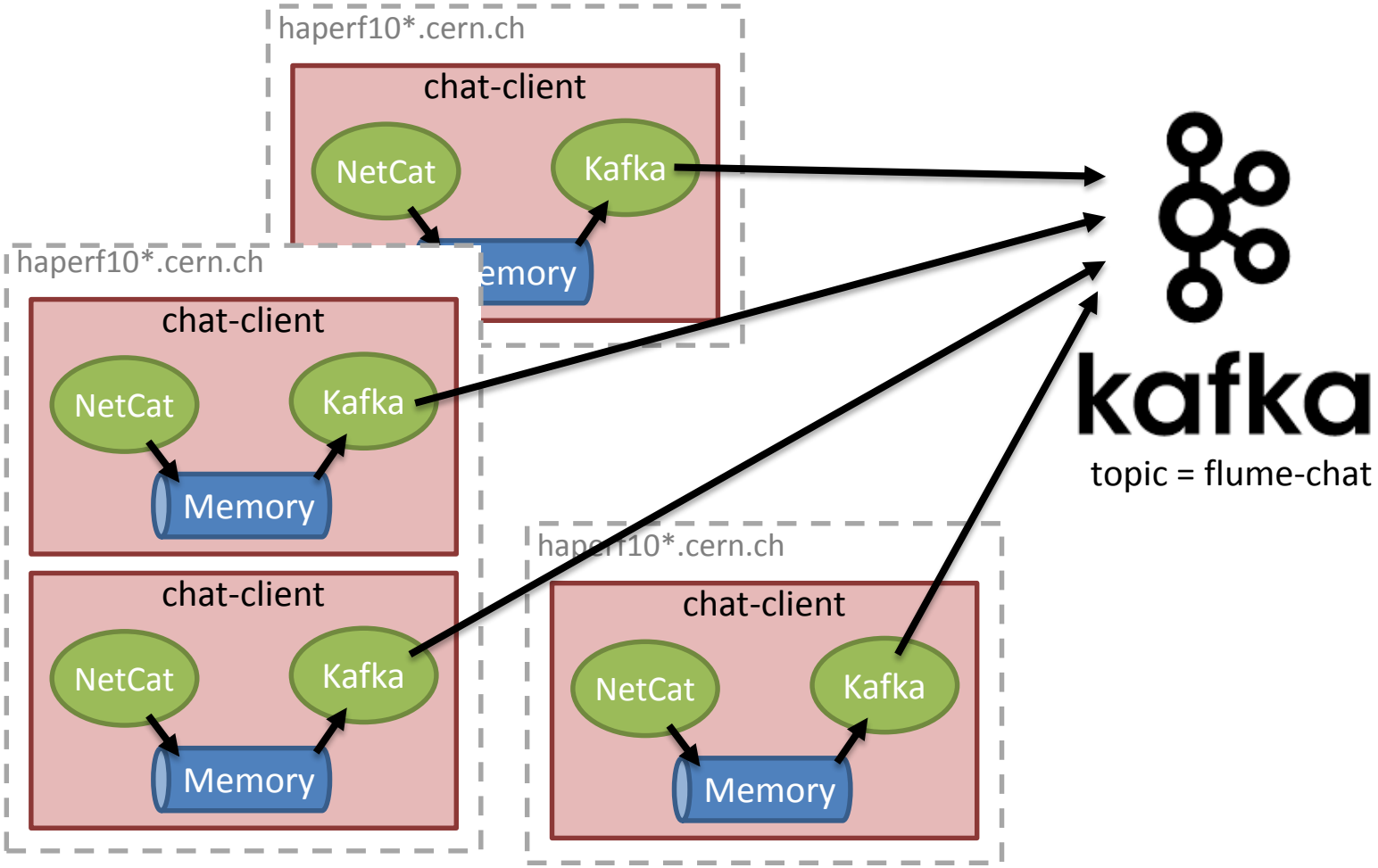
How Kafka can improve data ingestion

- As reliable big staging area



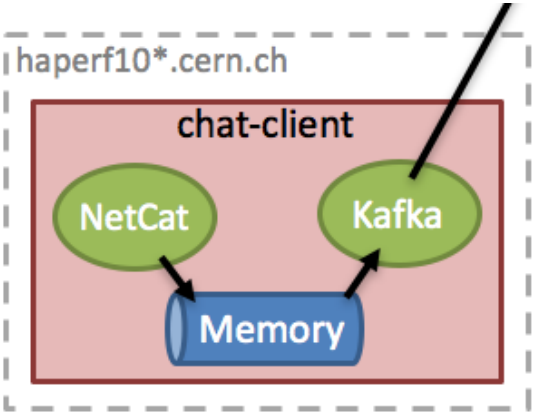
Flume hands-on

- Flume chat to Kafka



Flume hands-on

- Flume chat client to Kafka




```
# Name the components on this agent
chat-client.sources = netcat_source
chat-client.channels = memory_channel
chat-client.sinks = kafka_sink

# Configure source
chat-client.sources.netcat_source.type = netcat
chat-client.sources.netcat_source.channels = memory_channel
chat-client.sources.netcat_source.bind = 0.0.0.0
chat-client.sources.netcat_source.port = 1234
chat-client.sources.netcat_source.interceptors = adduser addtimestamp
chat-client.sources.netcat_source.interceptors.adduser.type = AddUserInterceptor$Builder
chat-client.sources.netcat_source.interceptors.addtimestamp.type = AddTimestampInterceptor$Builder

# Use a channel which buffers events in memory
chat-client.channels.memory_channel.type = memory
chat-client.channels.memory_channel.capacity = 1000
chat-client.channels.memory_channel.transactionCapacity = 100

# Describe the sink
chat-client.sinks.kafka_sink.type = org.apache.flume.sink.kafka.KafkaSink
chat-client.sinks.kafka_sink.channel = memory_channel
chat-client.sinks.kafka_sink.brokerList = haperf111.cern.ch:9092,haperf107.cern.ch:9092
chat-client.sinks.kafka_sink.topic = flume-chat
chat-client.sinks.kafka_sink.batchSize = 1
```

Flume hands-on: chat client to Kafka

- Go to client directory (on any machine)

```
ssh haperf10*.cern.ch
cd hadoop-tutorials-2016/2_data_ingestion/2_flume_chat_client_to_kafka/
```

- Compile and run it

```
./compile
./run-agent
```

- Initialize chat (different terminal)

```
cd hadoop-tutorials-2016/2_data_ingestion/2_flume_chat_client_to_kafka/
./consume-kafka-topic &
./init_chat      # Ctrl + ] and quit to exit
```

- Kill background process

```
./stop_kafka_consumer
```

Streaming data

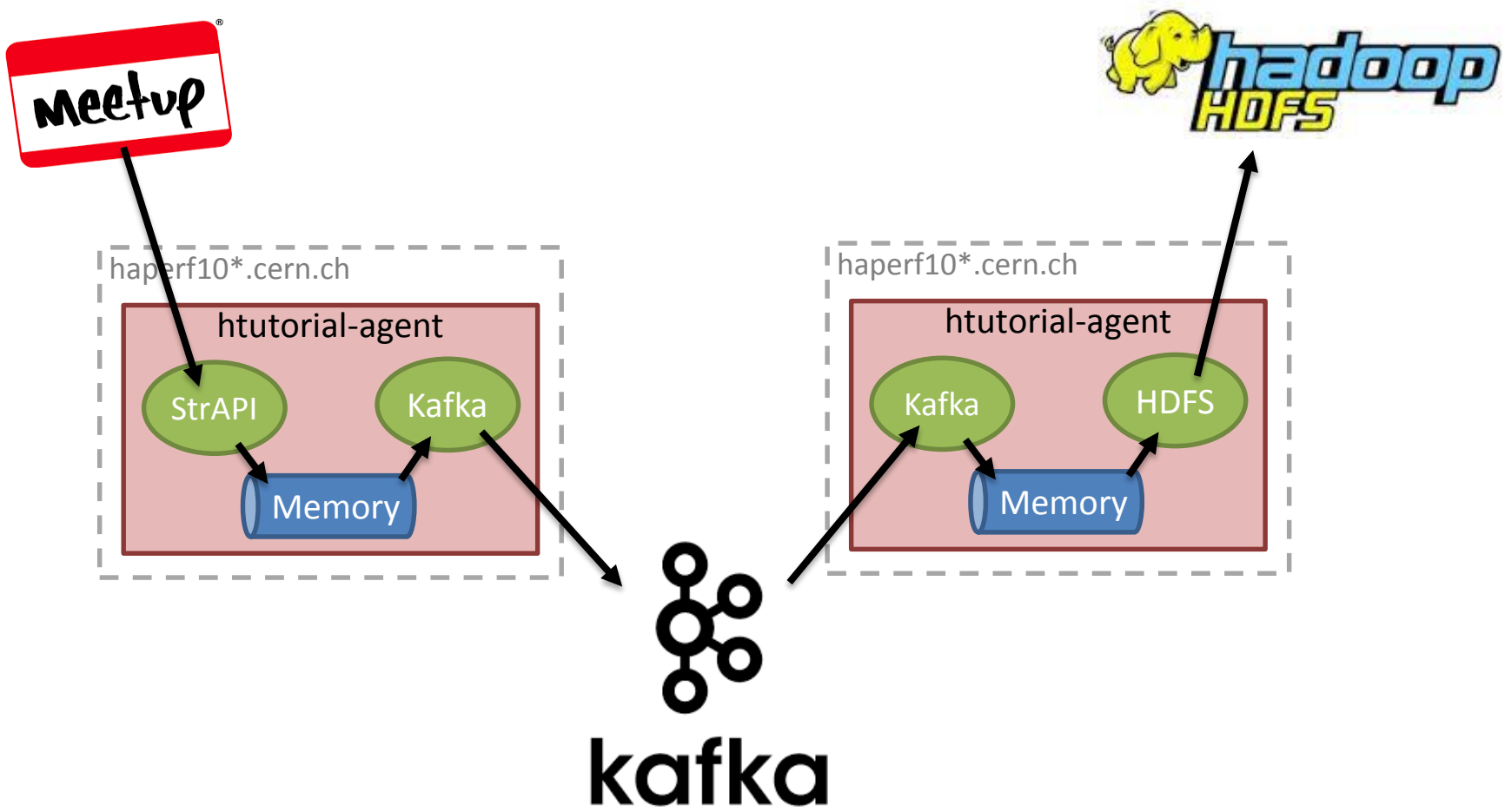
- Source

  - Meetups are: neighbours getting together to learn something, do something, share something…

- Streaming API
  - curl -s http://stream.meetup.com/2/rsvps

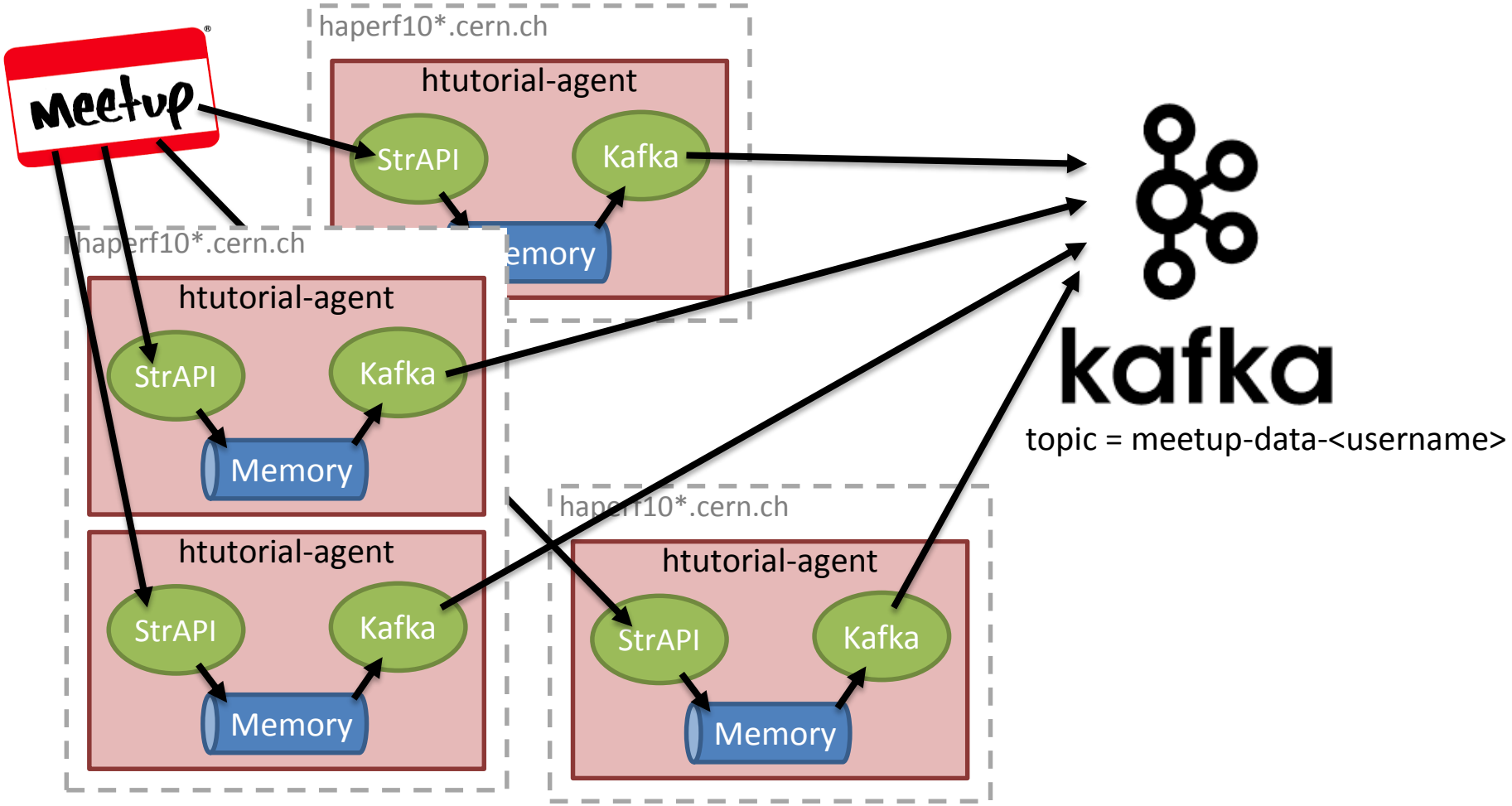
Flume hands-on

- Kafka as persistent buffer



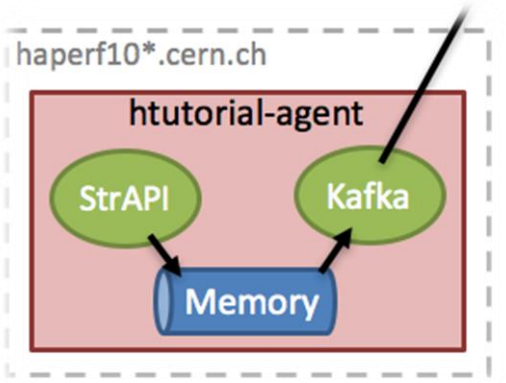
Flume hands-on

- Streaming data from Meetup to Kafka



Flume hands-on

- Streaming data from Meetup to Kafka




```
# Name the components on this agent
htutorial-agent.sources = meetup_source
htutorial-agent.channels = memory_channel
htutorial-agent.sinks = kafka_sink

# Configure source
htutorial-agent.sources.meetup_source.type = StreamingAPISource
htutorial-agent.sources.meetup_source.channels = memory_channel
htutorial-agent.sources.meetup_source.url = http://stream.meetup.com/2/rsvps
htutorial-agent.sources.meetup_source.batch.size = 5
htutorial-agent.sources.meetup_source.interceptors = addtimestamp
htutorial-agent.sources.meetup_source.interceptors.addtimestamp.type = timestamp

# Use a channel which buffers events in memory
htutorial-agent.channels.memory_channel.type = memory
htutorial-agent.channels.memory_channel.capacity = 1000
htutorial-agent.channels.memory_channel.transactionCapacity = 100

# Describe the sink
htutorial-agent.sinks.kafka_sink.type = org.apache.flume.sink.kafka.KafkaSink
htutorial-agent.sinks.kafka_sink.channel = memory_channel
htutorial-agent.sinks.kafka_sink.brokerList = haperf111.cern.ch:9092,haperf107.cern.ch:9092
htutorial-agent.sinks.kafka_sink.topic = meetup-data-<username>
htutorial-agent.sinks.kafka_sink.batchSize =
```

Flume hands-on

- Streaming data from Meetup to Kafka
  - StreamingAPISource

```
public class StreamingAPISource
    extends AbstractSource implements Configurable, PollableSource {

        @Override
        public Status process() throws EventDeliveryException {

            List<Event> events = new LinkedList<>();

            String line;
            try {
                connect();

                while ((line = reader.readLine()) != null && events.size() < batch_size) {
                    events.add(EventBuilder.withBody(line.getBytes()));
                }

                getChannelProcessor().processEventBatch(events);

                return Status.READY;
            } catch (IOException e) {
                if(connection != null)
                    connection.disconnect();
                connection = null;

                return Status.BACKOFF;
            }
        }
```

Flume hands-on: Meetup to Kafka

- Go to client directory (on any machine)

```
ssh haperf10*.cern.ch
cd hadoop-tutorials-2016/2_data_ingestion/3_meetup_to_kafka/
```

- Compile and run it

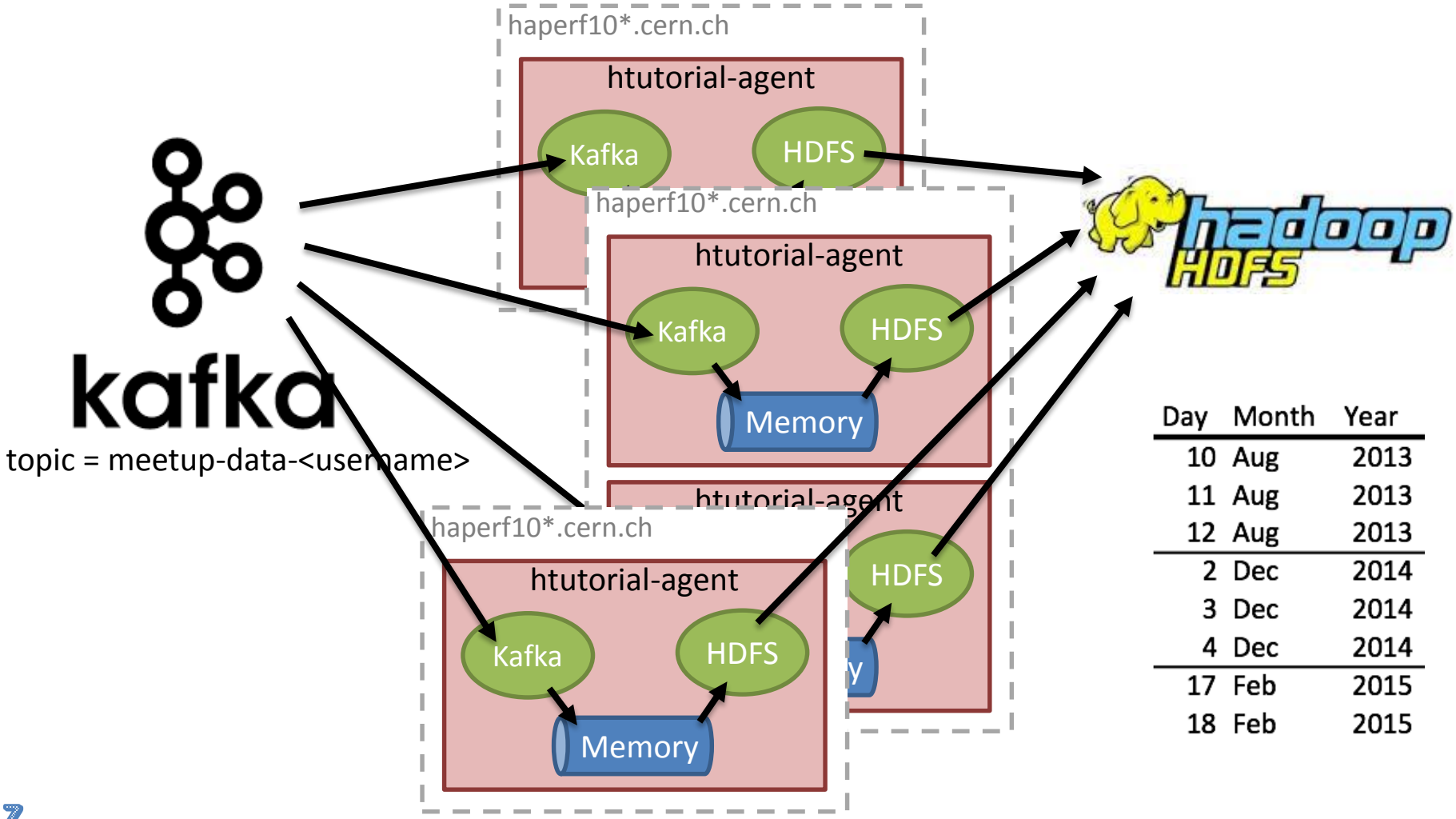
```
./compile
./run-agent
```

- Initialize chat (different terminal)

```
cd hadoop-tutorials-2016/2_data_ingestion/3_meetup_to_kafka/
./consume-kafka-topic
```

Flume hands-on

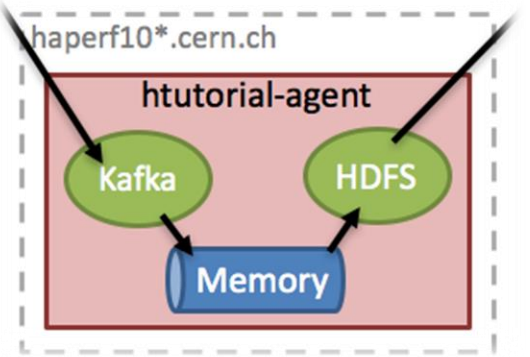
- From Kafka to partitioned data into HDFS



Day	Month	Year
10	Aug	2013
11	Aug	2013
12	Aug	2013
2	Dec	2014
3	Dec	2014
4	Dec	2014
17	Feb	2015
18	Feb	2015

Flume hands-on

- From Kafka to partitioned data into HDFS




```
# Name the components on this agent
htutorial-agent.sources = kafka_source
htutorial-agent.channels = memory_channel
htutorial-agent.sinks = hdfs_sink

# Configure source
htutorial-agent.sources.kafka_source.type = org.apache.flume.source.kafka.KafkaSource
htutorial-agent.sources.kafka_source.channels = memory_channel
htutorial-agent.sources.kafka_source.zookeeperConnect = haperf104:2181,haperf105:2181
htutorial-agent.sources.kafka_source.topic = meetup-data-<username>

# Use a channel which buffers events in memory
htutorial-agent.channels.memory_channel.type = memory
htutorial-agent.channels.memory_channel.capacity = 10000
htutorial-agent.channels.memory_channel.transactionCapacity = 1000

# Describe the sink
htutorial-agent.sinks.hdfs_sink.type = hdfs
htutorial-agent.sinks.hdfs_sink.channel = memory_channel
htutorial-agent.sinks.hdfs_sink.hdfs.fileType = DataStream
htutorial-agent.sinks.hdfs_sink.hdfs.path = hdfs://haperf100.cern.ch:8020/user/<username>/meetup-
data/year=%Y/month=%m/day=%d/
htutorial-agent.sinks.hdfs_sink.hdfs.rollCount = 100
htutorial-agent.sinks.hdfs_sink.hdfs.rollSize = 1000000
```

Flume hands-on: Kafka to HDFS

- Go to client directory (on any machine)

```
ssh haperf10*.cern.ch
cd hadoop-tutorials-2016/2_data_ingestion/3_kafka_to_part_hdfs
```

- Run it

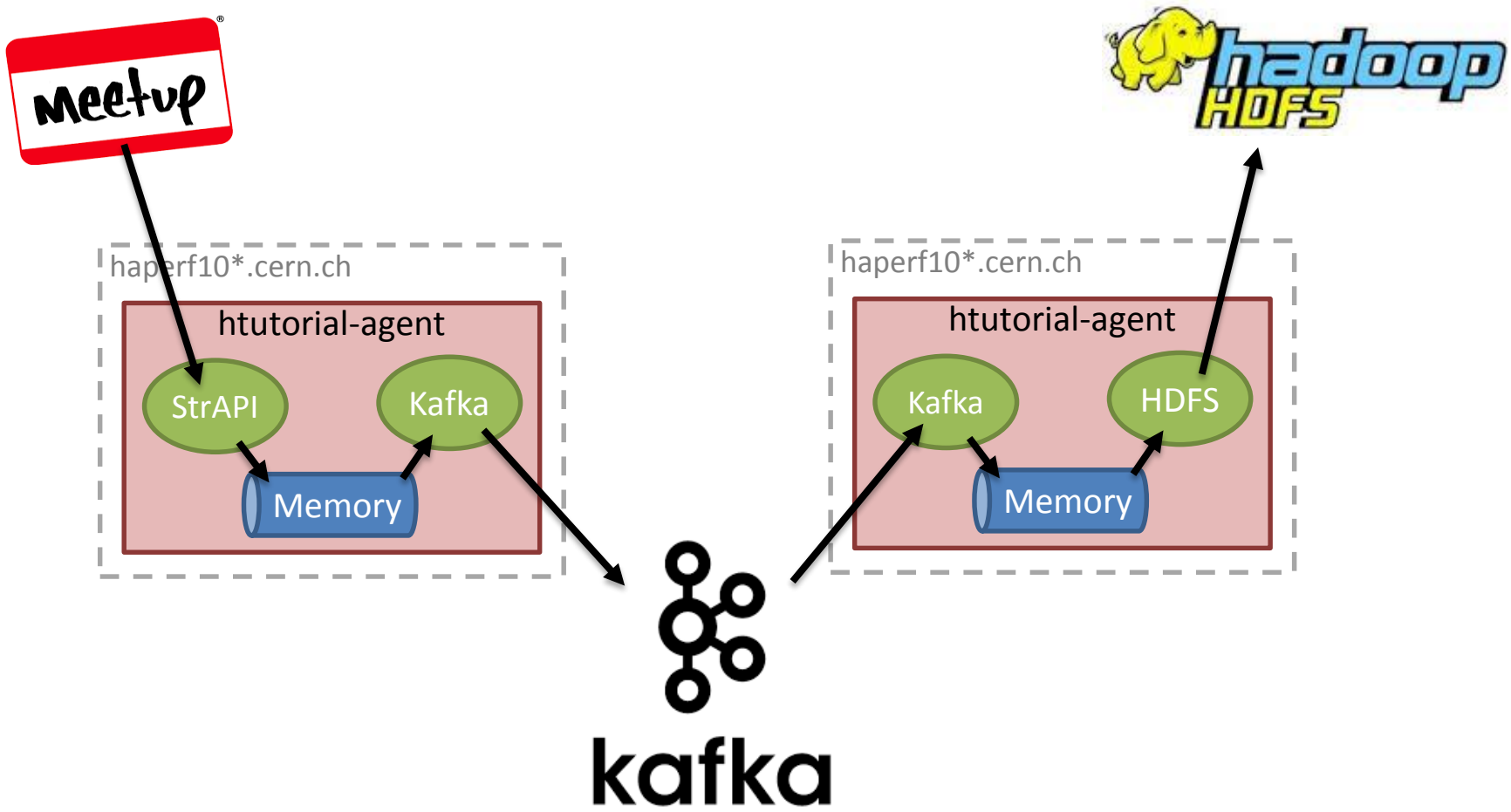
```
./run-agent
```

- Check data is landing (different terminal)

```
cd hadoop-tutorials-2016/2_data_ingestion/3_kafka_to_part_hdfs/
hdfs dfs -cat meetup-data/year=2016/month=07/day=20/*
```

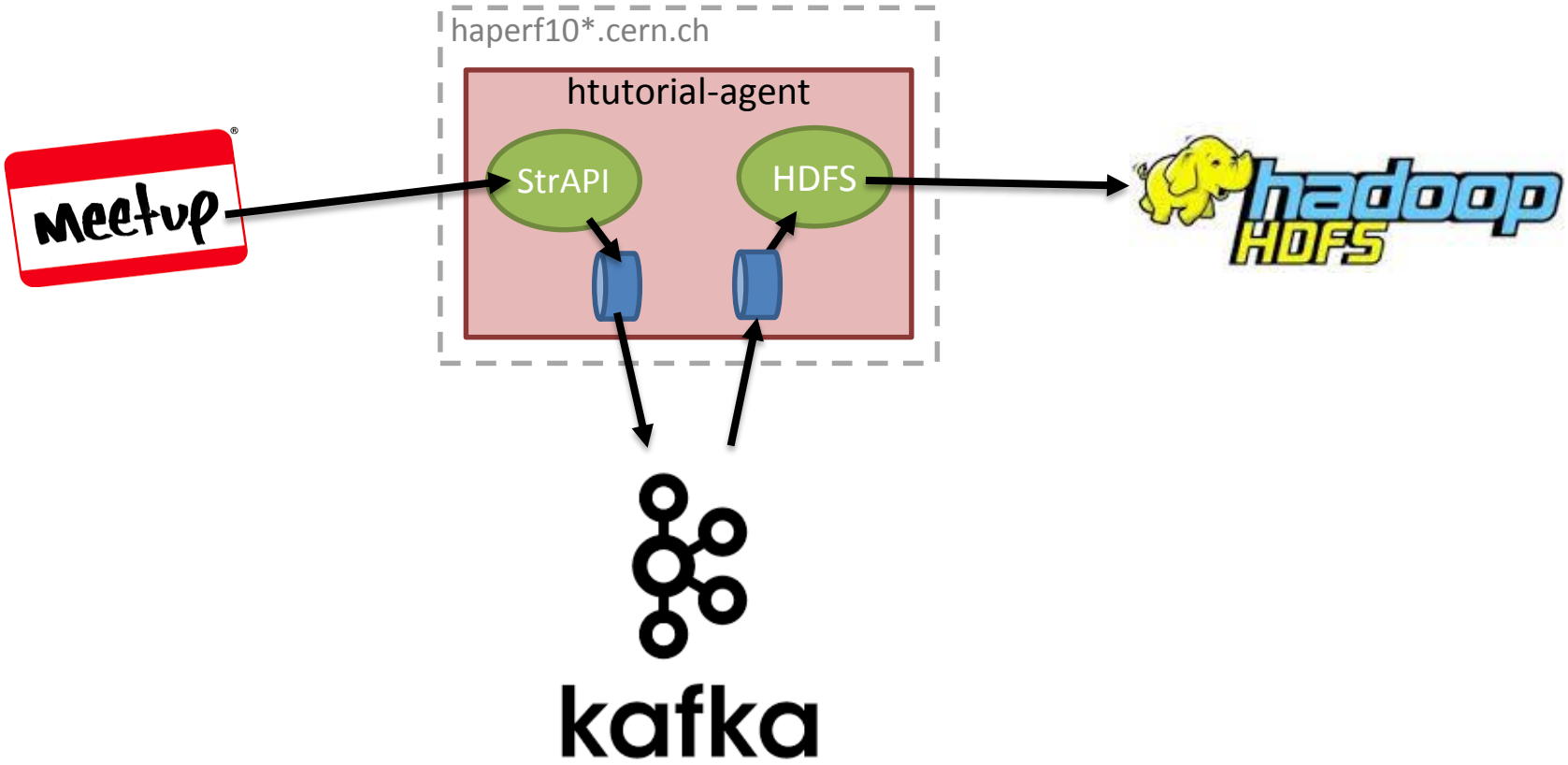
Flume hands-on

- Can we simplify the previous architecture?



Flume hands-on

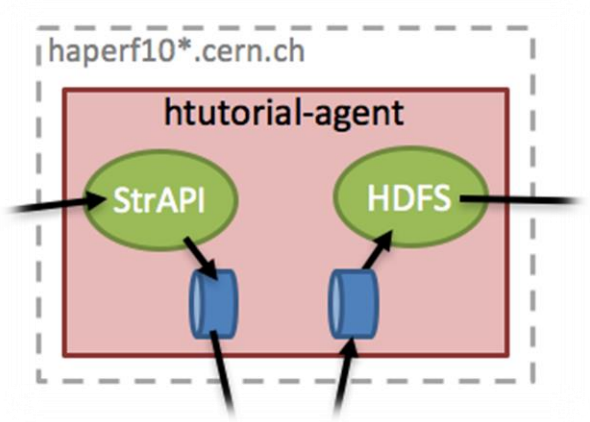
- Kafka as channel



topic = flume-channel-<username>

Flume hands-on

- Kafka as channel




```
# Name the components on this agent
htutorial-agent.sources = meetup_source
htutorial-agent.channels = kafka_channel
htutorial-agent.sinks = hdfs_sink

# Configure source
htutorial-agent.sources.meetup_source.type = StreamingAPISource
htutorial-agent.sources.meetup_source.channels = kafka_channel
htutorial-agent.sources.meetup_source.url = http://stream.meetup.com/2/rsvps
htutorial-agent.sources.meetup_source.batch.size = 5

# Use a channel which buffers events in Kafka
htutorial-agent.channels.kafka_channel.type = org.apache.flume.channel.kafka.KafkaChannel
htutorial-agent.channels.kafka_channel.topic = flume-channel-<username>
htutorial-agent.channels.kafka_channel.brokerList = haperf111.cern.ch:9092,haperf107.cern.ch:9092
htutorial-agent.channels.kafka_channel.zookeeperConnect = haperf104:2181,haperf105:2181

# Describe the sink
htutorial-agent.sinks.hdfs_sink.type = hdfs
htutorial-agent.sinks.hdfs_sink.channel = kafka_channel
htutorial-agent.sinks.hdfs_sink.hdfs.fileType = DataStream
htutorial-agent.sinks.hdfs_sink.hdfs.path = hdfs://haperf100.cern.ch/user/<username>/meetup-data
htutorial-agent.sinks.hdfs_sink.hdfs.rollCount = 100
htutorial-agent.sinks.hdfs_sink.hdfs.rollSize = 1000000
```

Flume hands-on: Kafka as channel

- Go to client directory (on any machine)

```
ssh haperf10*.cern.ch
cd hadoop-tutorials-2016/2_data_ingestion/4_meetup_kafka_as_channel/
```

- Compile and run it

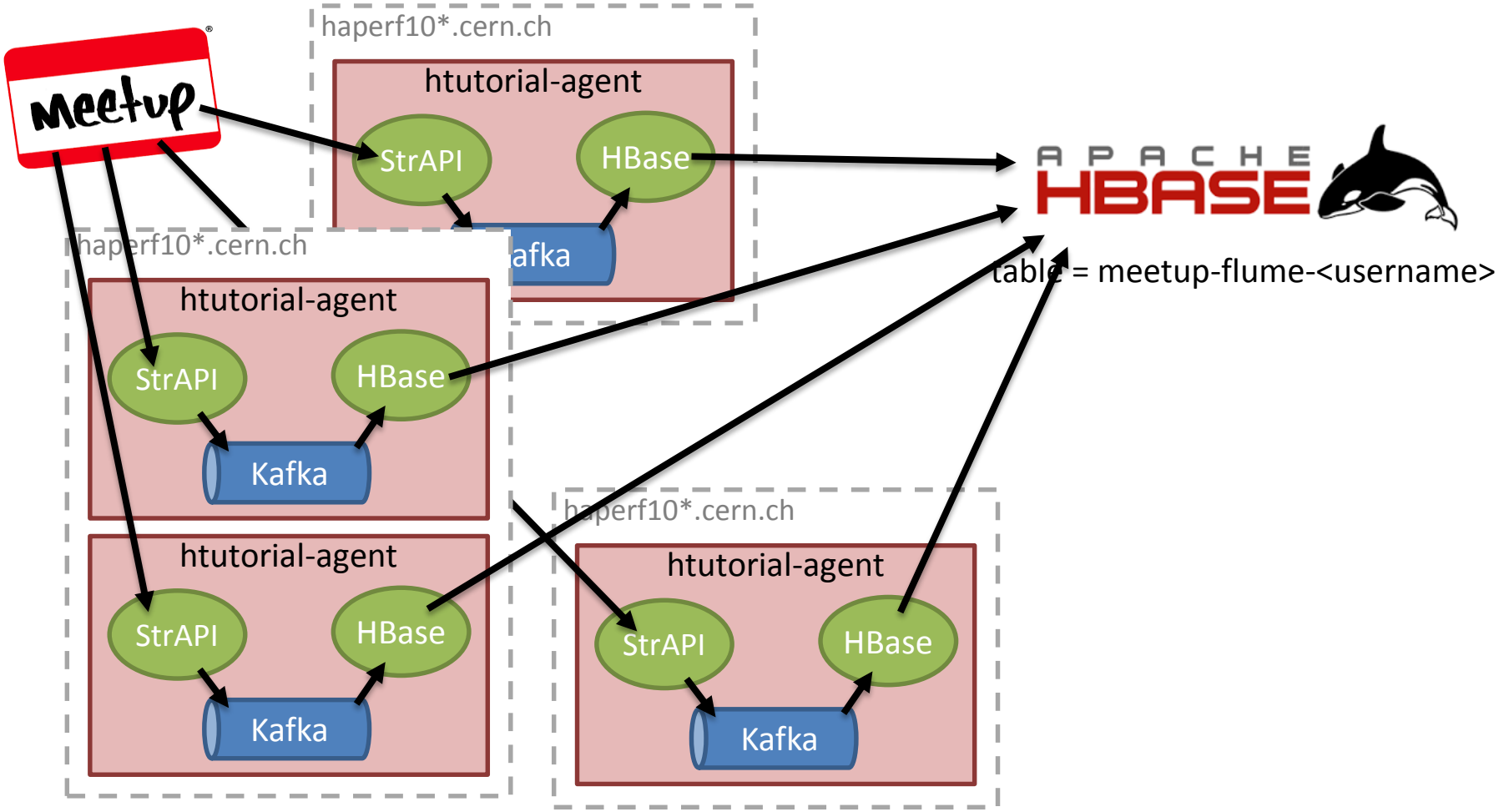
```
./compile
./run-agent
```

- See data coming (different terminal)

```
cd hadoop-tutorials-2016/2_data_ingestion/4_meetup_kafka_as_channel/
./consume-kafka-topic
hdfs dfs -cat "meetup-data/*"
```

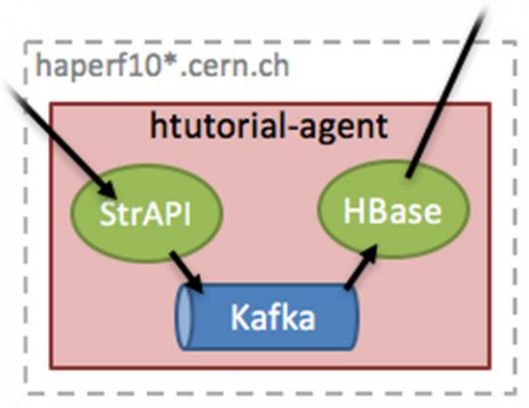
Flume hands-on

- Streaming data from Meetup to HBase



Flume hands-on

- Streaming data from Meetup to HBase




```
# Name the components on this agent
htutorial-agent.sources = meetup_source
htutorial-agent.channels = kafka_channel
htutorial-agent.sinks = hbase_sink

# Configure source
htutorial-agent.sources.meetup_source.type = StreamingAPISource
htutorial-agent.sources.meetup_source.channels = kafka_channel
htutorial-agent.sources.meetup_source.url = http://stream.meetup.com/2/rsvps
htutorial-agent.sources.meetup_source.batch.size = 5

# Use a channel which buffers events in Kafka
htutorial-agent.channels.kafka_channel.type = org.apache.flume.channel.kafka.KafkaChannel
htutorial-agent.channels.kafka_channel.topic = flume-channel-<username>
htutorial-agent.channels.kafka_channel.brokerList = haperf111.cern.ch:9092,haperf107.cern.ch:9092
htutorial-agent.channels.kafka_channel.zookeeperConnect = haperf104:2181,haperf105:2181

# Describe the sink
htutorial-agent.sinks.hbase_sink.type = hbase
htutorial-agent.sinks.hbase_sink.channel = kafka_channel
htutorial-agent.sinks.hbase_sink.table = meetup_flume_<username>
htutorial-agent.sinks.hbase_sink.columnFamily = event
htutorial-agent.sinks.hbase_sink.batchSize = 10
```

Flume hands-on: Meetup to HBase

- Go to client directory (on any machine)

```
ssh haperf10*.cern.ch
cd hadoop-tutorials-2016/2_data_ingestion/5_meetup_to_hbase/
```

- Compile, create table and run it

```
./compile
./create-hbase-table
./run-agent
```

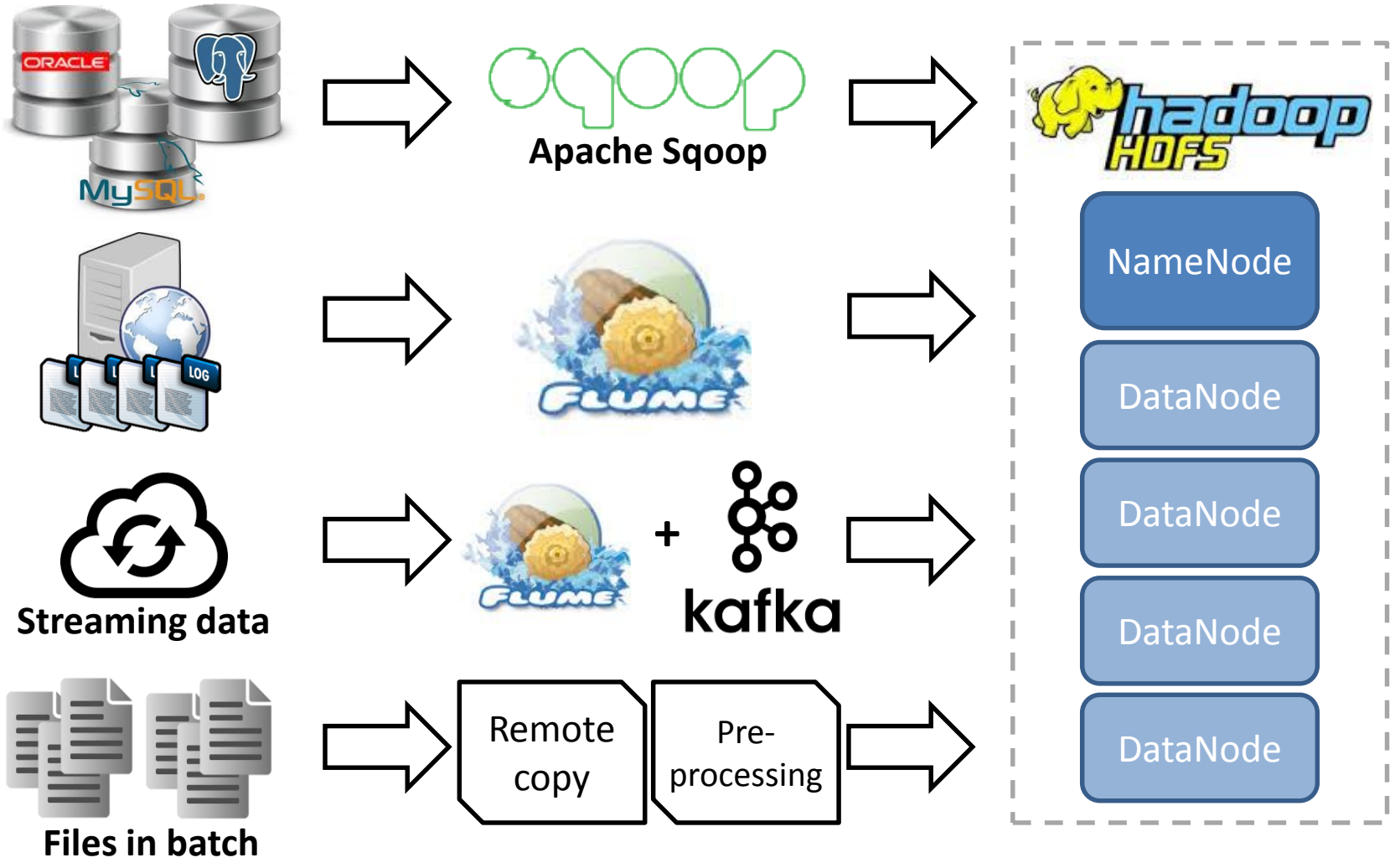
- See data coming (different terminal)

```
hbase shell
    > scan 'meetup_flume_<username>'
```

Summary

- There are many tools that can help you in ingesting data to Hadoop
- Batch ingestion is easy
  - Sqoop, Kite, HDFS API
- Real time ingestion is more complex
  - 2 phases needed
- Flume + Kafka for reliable, scalable data ingestion
  - can help to integrate data from multiple sources in near real time
  - **not only for Hadoop**

Summary – when to use what



Questions & feedback

it-dep-db-sas@cern.ch