

NIFI-GUIDE

Objectives

- Core NiFi Terminologies
- More on FlowFiles of NiFi
- Processors and Connections
- Processor Category
- Types of Processors Available in NiFi - List of processors shared to be covered
- Connection configuration
- Processor Configuration Settings
- Processor Configuration Scheduling
- Processor Configuration Property
- Processor Relationship in NiFi
- Connection Queue & Back Pressure in NiFi

Introduction to Apache NiFi

- An Open Source Data Distribution
- Framework for managing complex dataflows
- Provides a way to move data from one place/system to another place / system
- Making transformations and routing decisions as necessary in real time streaming
- Scalable directed graphs of data routing, transformation, and system mediation logic
- Automate the flow of data between systems
- E.g.: JSON -> Database, FTP-> Hadoop, Kafka -> ElasticSearch, etc...
- Data could be anything like log files, HTTP request, XML, CSV, Audio, Video, Telemetry data
- Drag and drop interface
- Focus on configuration of processors (i.e. what matters only to the user)
- Scalable across a cluster of machines
- Guaranteed Delivery / No Data Loss
- Data Buffering / Back Pressure / Prioritization Queuing / Latency vs Throughput
- Templates for rapid development and deployment

Apache NiFi use cases:

What Apache NiFi is good at:

- Reliable and secure transfer of data between systems
- Delivery of data from sources to analytic platforms
- Enrichment and preparation of data:
 - 1 Conversion between formats
 - 2 Parsing
 - 3 Routing decisions

What Apache NiFi shouldn't be used for:

- 1 Distributed Computation
- 2 Complex Event processing
- 3 Joins, rolling windows, aggregates operations

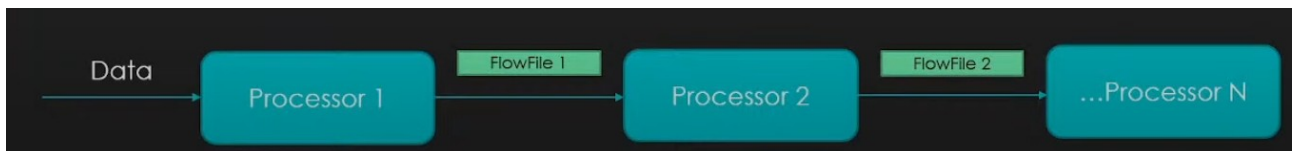
NIFI-GUIDE

Core NiFi Terminologies: Flow File

- It's basically the data
- Comprised of two elements:
 - Content: the data itself
 - Attributes: Key value pairs associated with the data (creation date, filename, UUID etc....)
- Gets persisted to disk after creation
- A FlowFile represents each object moving through the system and for each one, NiFi keeps track of a map of key/value pair attribute strings and its associated content of zero or more bytes.

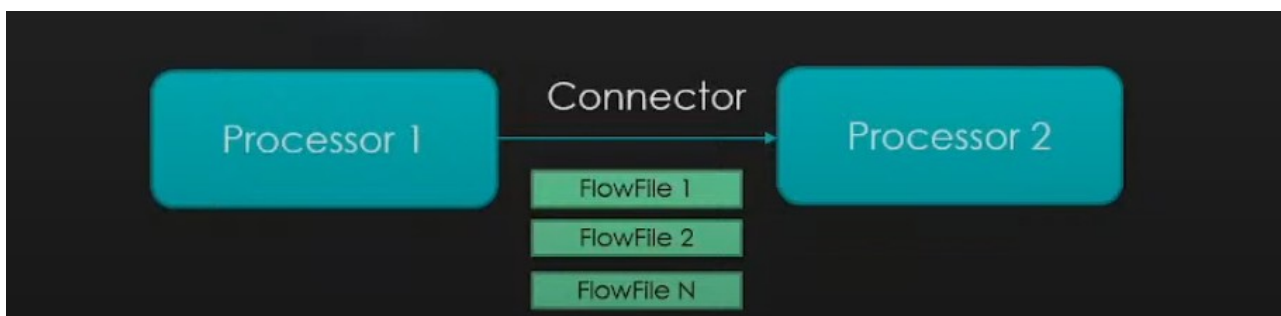
Core NiFi Terminologies: Processor

- Applies a set of transformations and rules to FlowFiles, to generate new FlowFiles
- Any processor can process any FlowFile
- Processors are passing FlowFile references to each other to advance the data processing
- They are all running in parallel (different threads)



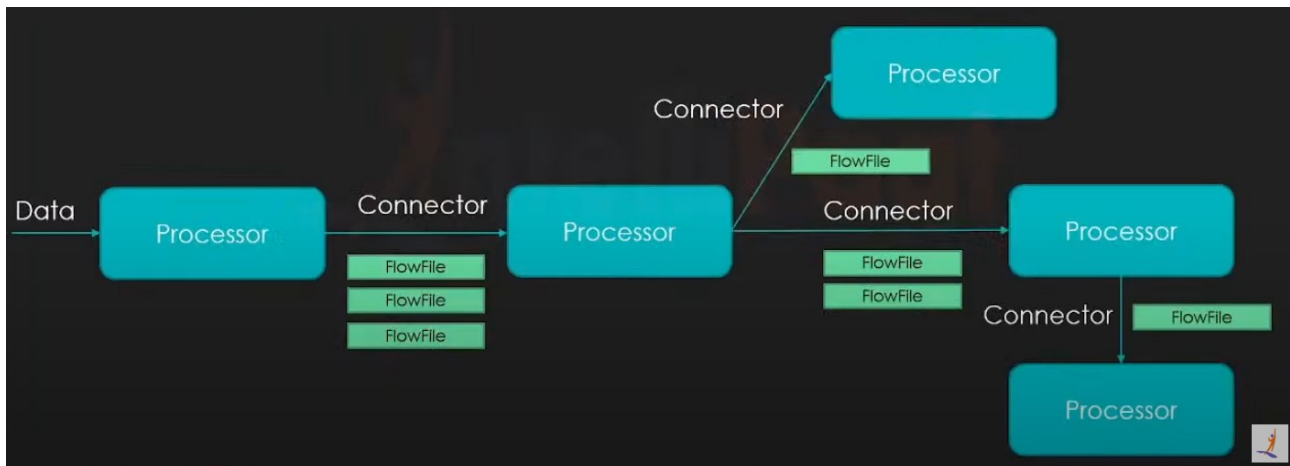
Core NiFi Terminologies: Connector

- It's basically a queue of all the FlowFiles that are yet to be processed by Processor
- Defines rules about how FlowFiles are prioritized (which ones first, which ones not at all)
- Can define backpressure to avoid overflow in the system



All Connected: Flowfile, Processors, Connectors

NIFI-GUIDE



NiFi: Categorization of processors

- **Over 286 bundled processors (as of 1.11.4 version)**
 - **Data Transformation:** ReplaceText, JoltTransformJSON
 - **Routing and Mediation:** RouteOnAttribute, RouteOnContent, ControlRate...
 - **Database Access:** ExecuteSQL, ConvertJSONToSQL, PutSQL...
 - **Attribute Extraction:** Evaluate JsonPath, ExtractText, UpdateAttribute...
 - **System Interaction:** ExecuteProcess
 - **Data Ingestion:** GetFile, GetFTP, GetHTTP, GetHDFS, ListenUDP, GetKafka...
 - **Sending Data:** PutFile, PutFTP, PutKafka, PutEmail...
 - **Splitting and Aggregation:** SplitText, Split.Json, SplitXml, MergeContent...
 - **HTTP:** GetHTTP, ListenHTTP, PostHTTP...
 - **AWS:** FetchS3Object, PutS3Object, PutSNS, GetSQS

NIFI: Connection Configuration

Connecting Components

- Connecting two components and acts like a buffer /queue.
- **Settings**
 - 1 **Flowfile Expiration** -Data not processed in timely fashion will be removed from the system. Zero Secs that is the default value, indicates data will never expire.
 - 2 Expiration can be done effectively with prioritizes.

//new page

Right click on processor and select option **configuration** then you will see window like below

NIFI-GUIDE

The screenshot shows the 'Edit Processor' dialog for 'ExecuteScript 2.4.0'. It has three tabs: 'Settings', 'Scheduling', and 'Properties'. The 'Settings' tab is active and contains the following fields:

- Name***: A text input field containing 'ExecuteScript'.
- Id**: A text input field containing 'b4d1ae64-0196-1000-fc5c-22f61807c6fc'.
- Type**: A text input field containing 'ExecuteScript 2.4.0'.
- Bundle**: A text input field containing 'org.apache.nifi - nifi-scripting-nar'.
- Penalty Duration ⓘ***: A text input field containing '30 sec'.
- Yield Duration ⓘ***: A text input field containing '1 sec'.
- Bulletin Level ⓘ***: A dropdown menu currently showing 'WARN'.

Settings Tab Fields Explained:

1. Name

- **What it is:** The name of the processor.
- **Purpose:** Helps you identify it on the canvas.
- **Tip:** Use a meaningful name like `ChunkingScript` or `JSONFormatter`.

2. Id

- **What it is:** A unique UUID automatically assigned to the processor.
- **Purpose:** Used internally by NiFi to track this processor.
- **Note:** You cannot change it.

3. Type

- **What it is:** The processor type — in this case, `ExecuteScript`.
- **Version:** 2.4.0
- **Purpose:** Tells NiFi which processor logic to use.

NIFI-GUIDE

4. Bundle

- **What it is:** Identifies the NiFi component group that contains the processor.
- **Example:**
`org.apache.nifi - nifi-scripting-nar`
- **Purpose:** Shows which NAR (NiFi Archive) this processor belongs to.

5. Penalty Duration

- **Default:** 30 sec
- **What it is:** Time to wait before the same flow file is retried if it fails (e.g., exception in script).
- **Purpose:** Prevents retrying too fast and overloading the system.
- **Example:** If a script fails, the flow file is penalized for 30 seconds.

6. Yield Duration

- **Default:** 1 sec
- **What it is:** If the processor encounters a general error, it will stop processing for this time.
- **Purpose:** Prevents continuous failure loops.
- **Example:** If the script has no input, it will wait 1 second before trying again.

7. Bulletin Level

- **Default:** WARN
- **What it is:** The minimum log level at which messages will be shown in the NiFi UI.
- **Options:** DEBUG, INFO, WARN, ERROR
- **Purpose:** Controls how verbose the NiFi UI bulletins will be for this processor.
- **Tip:** Set to DEBUG temporarily if you're troubleshooting a script.

NIFI-GUIDE

Scheduling Tab Fields Explained:

Edit Processor | ExecuteScript 2.4.0

Settings **Scheduling** Properties Relationships

Scheduling Strategy ⓘ*
Timer driven ▼

Concurrent Tasks ⓘ*
1

Run Schedule ⓘ*
0 sec

Execution ⓘ*
All nodes ▼

1. Scheduling Strategy

- **Value shown:** Timer driven
- **What it is:** Defines how the processor is triggered.
- **Options:**
 - **Timer driven:** Runs periodically based on the Run Schedule (default and most common).
 - **Event driven:** Runs when triggered by system events (rarely used).
 - **CRON driven:** Runs based on a CRON expression (e.g., "0 0 * * *" for hourly).
- **Use:** Timer driven is suitable when you want the processor to run at fixed intervals.

2. Concurrent Tasks

- **Value shown:** 1

NIFI-GUIDE

- **What it is:** Number of threads NiFi can use to run this processor in parallel.
- **Use:**
 - Increase it (e.g., 2, 4, 8...) if your processor can handle parallelism (e.g., multiple flow files).
 - Keep it 1 for sequential processing or when using shared resources like files.

3. Run Schedule

- **Value shown:** 0 sec
- **What it is:** Delay between each run of the processor.
- **Use:**
 - 0 sec = run as fast as possible (immediate after each run).
 - 5 sec = wait 5 seconds between each run.
- **Tip:** Add delay if the processor is resource-intensive or hits an external API.

4. Execution

- **Value shown:** All nodes
- **What it is:** Determines on which cluster node(s) the processor will run.
- **Options:**
 - All nodes: Run on every node in a NiFi cluster.
 - Primary node: Run only on the designated primary node.
- **Use:**
 - Use All nodes if the processor handles independent data.
 - Use Primary node when writing to shared destinations like databases or filesystems to avoid conflicts.

NIFI-GUIDE

The screenshot shows the 'Edit Connection' dialog in Apache NiFi, with the 'Settings' tab selected. The dialog is divided into two main sections: 'Details' and 'Settings'. The 'Details' section on the left contains fields for 'Name' (empty), 'Id' (b4d4e93d-0196-1000-3b52-26e566ad76d3), 'FlowFile Expiration' (0 sec), 'Back Pressure Object Threshold' (10000), 'Size Threshold' (1 GB), and 'Load Balance Strategy' (Do not load balance). The 'Settings' section on the right contains 'Available Prioritizers' (FirstInFirstOutPrioritizer, NewestFlowFileFirstPri..., OldestFlowFileFirstPrio..., PriorityAttributePrioriti...) and 'Selected Prioritizers' (empty dashed box).

Edit Connection

Details Settings

Name

Id
b4d4e93d-0196-1000-3b52-26e566ad76d3

FlowFile Expiration ⓘ*
0 sec

Back Pressure Object Threshold ⓘ*
10000

Size Threshold ⓘ*
1 GB

Load Balance Strategy ⓘ*
Do not load balance ▼

Available Prioritizers ⓘ

FirstInFirstOutPrioritizer

NewestFlowFileFirstPri...

OldestFlowFileFirstPrio...

PriorityAttributePrioriti...

Selected Prioritizers ⓘ

Edit Connection – Settings Tab Explained:

1. Name

- Optional field to give your connection a custom name.
- Helpful for identifying connections between processors in complex flows.

2. Id

- A unique identifier automatically assigned to the connection.
- Used internally by NiFi; typically not editable.

3. FlowFile Expiration

- **Value shown:** 0 sec
- Determines how long a FlowFile can remain in this connection queue.
- **0 sec** means it will never expire.
- Set a time limit (e.g., 5 min) if you want to drop stale files.

NIFI-GUIDE

4. Back Pressure Object Threshold

- **Value shown:** 10000
- Max number of FlowFiles allowed in this connection queue before **back pressure** is applied.
- Once the limit is reached, upstream processors will stop pushing new FlowFiles.

5. Size Threshold

- **Value shown:** 1 GB
- Maximum total size of FlowFiles allowed in the queue before back pressure is triggered.
- Helps avoid memory/disk overload.

6. Load Balance Strategy

- **Value shown:** Do not load balance
- Controls how FlowFiles are distributed across nodes in a NiFi cluster.
- **Options:**
 - Do not load balance: Default, no balancing.
 - Round Robin: Distributes FlowFiles evenly across nodes.
 - Single Node: Sends all FlowFiles to one node.
 - Partition by Attribute: Uses a FlowFile attribute to group files per node.

7. Available Prioritizers

Controls the **order** in which FlowFiles are retrieved from the queue. You can select one or more to customize behavior.

Options:

- **FirstInFirstOutPrioritizer:** Oldest files processed first (default).
- **NewestFlowFileFirstPrioritizer:** Most recent files processed first.
- **OldestFlowFileFirstPrioritizer:** Explicitly enforces old-first logic.
- **PriorityAttributePrioritizer:** Prioritize based on a custom FlowFile attribute (e.g., "priority=high")

NIFI-GUIDE

//new page

Working with Attributes in flow file ..

An Example

- A file that is picked up from a local file system, the FlowFile would have an attribute called filename that reflected the name of the file on the file system. Additionally, the FlowFile will have a path attribute that reflects the directory on the file system that this file lived in. Th FlowFile will also have an attribute named uuid, which is a unique identifier for this FlowFile.

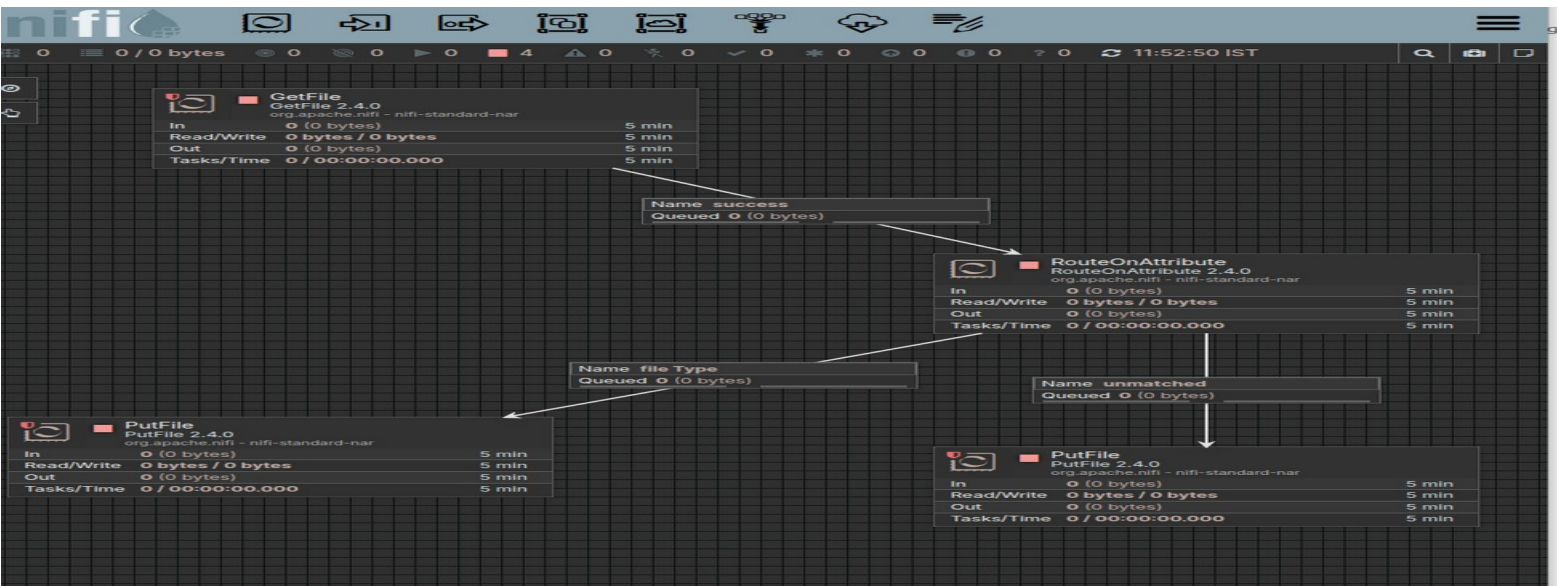
Other Core Attributes

- Absolute Path (absolute.path): The FlowFile's absolute path indicates the absolute directory to which a FlowFile belongs and does not contain the filename.
- Priority (priority): A numeric value Indicating the FlowFile priority.
- MIME Type (mime.type): The MIME Type of this FlowFile.
- Discard Reason (discard.reason): Specifies the reason that a FlowFile is being discarded.
- Alternate Identifier (alternate.Identifier): Indicates an identifier other than the FlowFile's UUID that is known to refer to this FlowFile.

Working with Attributes in flow file – Hands-On

- GetFile Processor
- RoutingOnAttribute Processor
- PutFile Processor JSON (Route JSON Files)
- PutFile Processor CSV(Route CSV Files)**see**
https://www.youtube.com/watch?v=himL2kRQA4c&t=25s&ab_channel=Intellipaati
from timestamp 1:19:40 to timestamp 1:34:00

NIFI-GUIDE



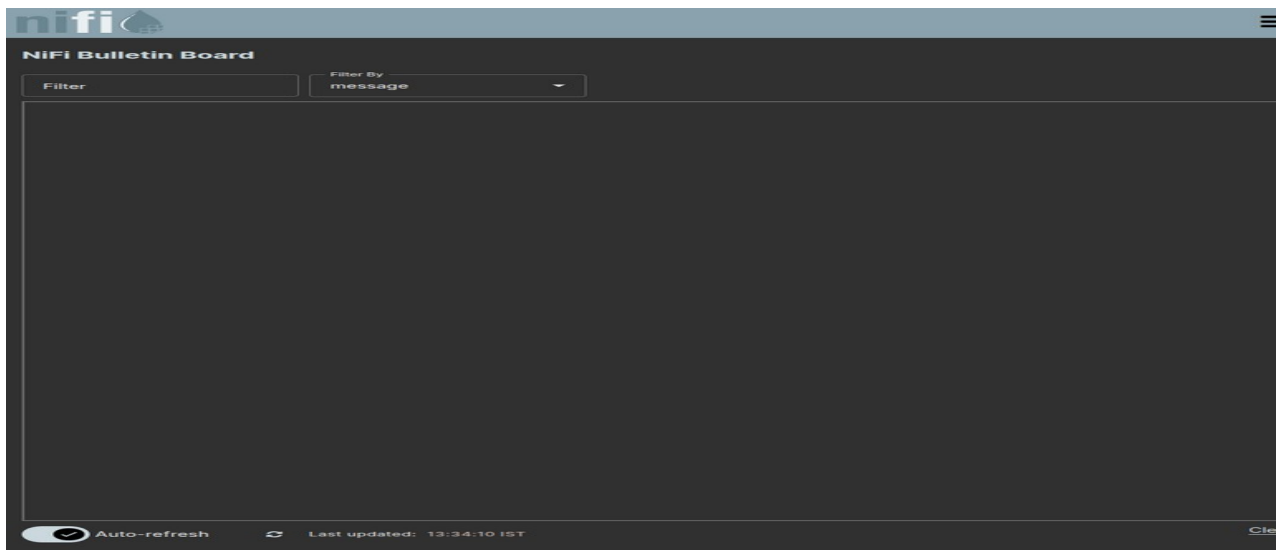
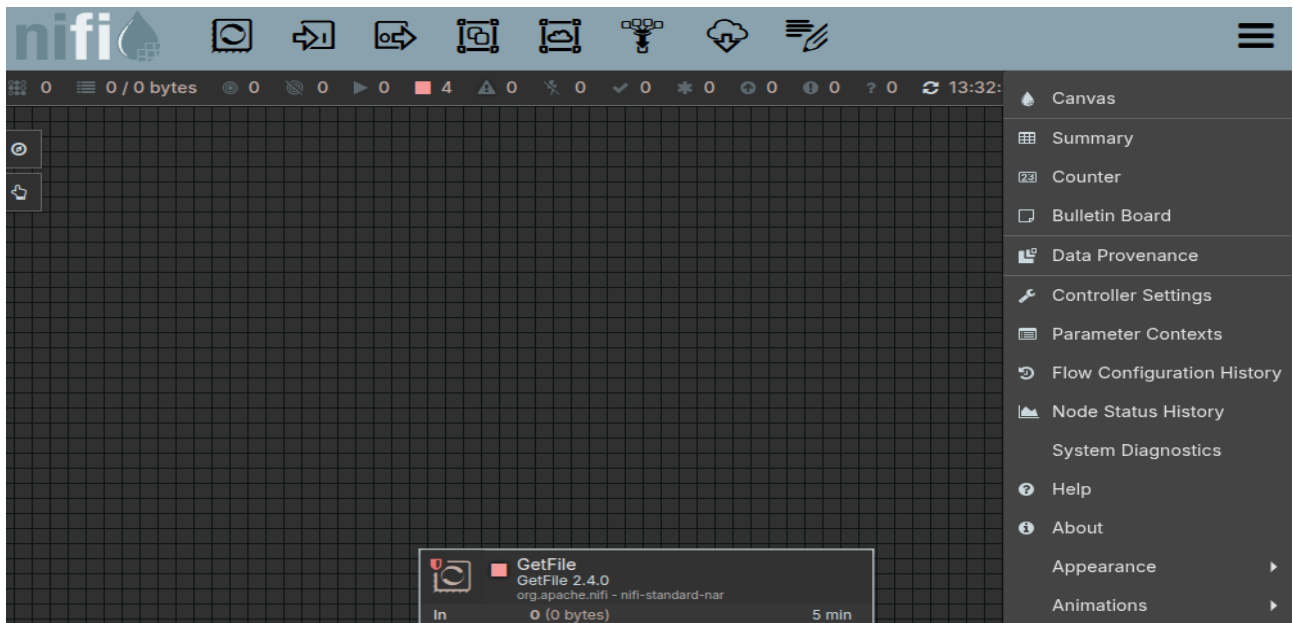
Log Configuration and Monitoring Logs

- There is one configuration file in NiFi that manages all the logging operations performed by NiFi: this file is in the configuration directory (NIFI_HOME/conf) and is named logback.
- It is good to know that this file can be modified "on-the-fly" and it won't be required to restart NiFi for the modifications to be taken into account.
- **Logging helps in debugging and monitoring NiFi.**

How to See Logs

click on top right bar(three liner symbol) then click bulletin Board. There you can see logs.

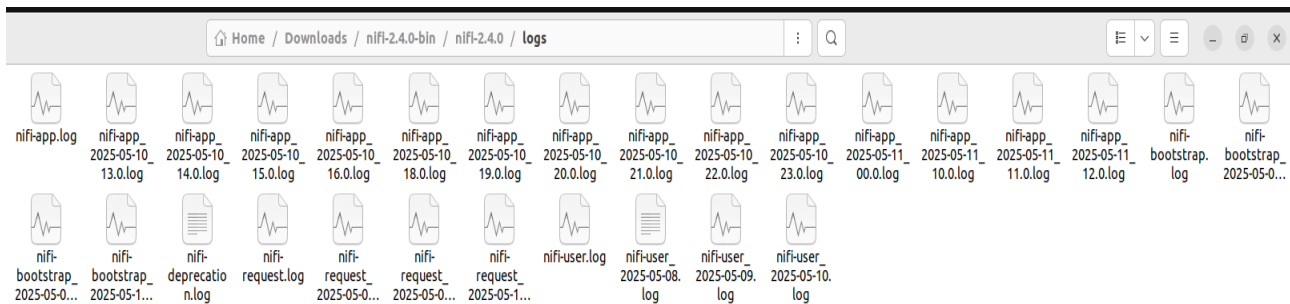
NIFI-GUIDE



->other way

you can follow path shown in below image (initial directory location depends on your downloaded location)

NIFI-GUIDE



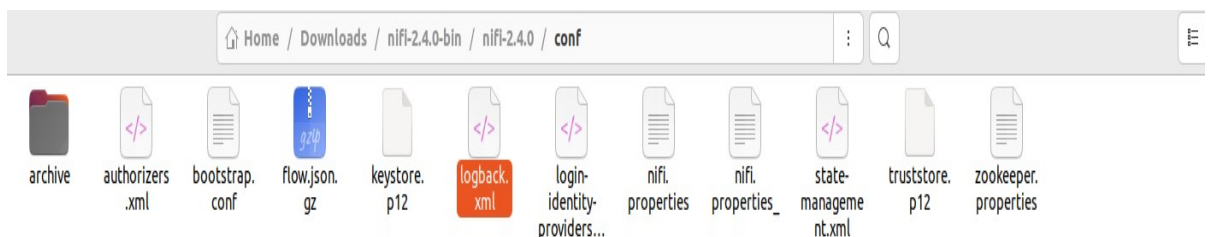
here you can see your application logs click on nifi-app.log file

Log Configuration and Monitoring Hands-on

- Improve on previous data flow. Add additional attribute (key and value pair) to flowfile if the file type is not JSON type. Please use **LogAttribute processor** as the destination and write logs to a different file for monitoring purpose.
- You need to put appropriate configuration in **logback.xml** file for the same.
- https://www.youtube.com/watch?v=himL2kRQA4c&t=25s&ab_channel=Intellipaat from timestamp 2:01:03 to timestamp 2:21:00

logback.xml

- `logback.xml` is the configuration file used by Logback, a powerful and flexible logging framework for Java applications. It's the successor to Log4j and is often used in projects like Apache NiFi, Spring Boot, and others for managing logging behavior.
- you can follow path shown in below image (initial directory location depends on your downloaded location)



NIFI-GUIDE

```
<appender name="APP_FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
  <file>${org.apache.nifi.bootstrap.config.log.dir}/nifi-app.log</file>
  <rollingPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
    <!--
      For daily rollover, use 'app_%d.log'.
      For hourly rollover, use 'app_%d{yyyy-MM-dd_HH}.log'.
      To GZIP rolled files, replace '.log' with '.log.gz'.
      To ZIP rolled files, replace '.log' with '.log.zip'.
    -->
    <fileNamePattern>${org.apache.nifi.bootstrap.config.log.dir}/nifi-app_%d{yyyy-MM-dd_HH}.
i.log</fileNamePattern>
    <!-- Control the maximum size of each log file before rolling over -->
    <maxFileSize>100MB</maxFileSize>
    <!-- Control the maximum number of log archive files kept and asynchronously delete older
files -->
    <maxHistory>30</maxHistory>
    <!-- Control the total size of all log archive files for this appender -->
    <totalSizeCap>3GB</totalSizeCap>
    <!-- Log files exceeding maximum settings will be rolled on startup -->
    <cleanHistoryOnStart>true</cleanHistoryOnStart>
  </rollingPolicy>
  <immediateFlush>true</immediateFlush>
  <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
    <pattern>%date %level [%thread] %logger{40} %msg%n</pattern>
  </encoder>
</appender>
```

This `logback.xml` snippet configures a rolling file appender for logging in Apache NiFi. Below is a breakdown of the configuration:

Appender Definition:

```
<appender name="APP_FILE"
class="ch.qos.logback.core.rolling.RollingFileAppender">
```

- `name="APP_FILE"`: Identifier for this appender.
- `RollingFileAppender`: Writes logs to a file and rolls them over based on size or time.

Log File Location:

```
<file>${org.apache.nifi.bootstrap.config.log.dir}/nifi-app.log</file>
```

- Writes logs to the file `nifi-app.log` inside the directory specified by `org.apache.nifi.bootstrap.config.log.dir`.
- This property is typically defined in `nifi.properties` and resolves to a directory like `/opt/nifi/logs`.

Rolling Policy:

```
<rollingPolicy
class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
```

This policy combines both time-based and size-based rollover. Logs will roll over either when a time period ends or when the file size exceeds a defined limit.

Filename Pattern for Rolled Logs:

NIFI-GUIDE

```
<fileNamePattern>${org.apache.nifi.bootstrap.config.log.dir}/nifi-app_%d{yyyy-MM-dd_HH}.log</fileNamePattern>
```

- Log files will be rolled over with names like: /path/to/logs/nifi-app_2025-05-11_10.log

File Size and Retention Settings:

```
<maxFileSize>100MB</maxFileSize>
<maxHistory>30</maxHistory>
<totalSizeCap>3GB</totalSizeCap>
```

- `maxFileSize`: Rolls over when file exceeds 100 MB.
- `maxHistory`: Keeps up to 30 archived log files.
- `totalSizeCap`: Deletes oldest logs if total size of all log files exceeds 3 GB.

Cleanup Behavior:

```
<cleanHistoryOnStart>true</cleanHistoryOnStart>
```

- Deletes old logs at startup if they exceed `maxHistory` or `totalSizeCap`.

Flush Behavior:

```
<immediateFlush>true</immediateFlush>
```

- Ensures that log entries are written to disk immediately without buffering.

Log Format (Encoder):

```
<encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
  <pattern>%date %level [%thread] %logger{40} %msg%n</pattern>
</encoder>
```

- Formats log entries with date, log level, thread name, logger name, and the actual message.
- Example log line:
2025-05-11 10:45:00 INFO [main]
o.a.n.c.StandardProcessScheduler Starting processor...

NIFI-GUIDE

Navigate to bottom of logback.xml file

```
288  
289 <root level="INFO">  
290   <appender-ref ref="APP_FILE" />  
291 </root>  
292
```

- `level="INFO"`: Sets the minimum log level for all logs. Levels below INFO (e.g., DEBUG) will not be logged.
- `<appender-ref ref="APP_FILE" />`: Binds the root logger to the APP_FILE appender, meaning all logs will be written to the `nifi-app.log` file.

Summary:

This configuration logs messages to `nifi-app.log`, rolls over logs hourly or when file size exceeds 100 MB, retains up to 30 log files or 3 GB total, and formats each log entry with relevant metadata.

Source:

1) Follow below link after timestamp 2:21:00 for further Exploration of Some More Processes

https://www.youtube.com/watch?v=himL2kRQA4c&t=25s&ab_channel=Intellipaat

2) short playlist to follow

https://youtube.com/playlist?list=PL55symSEWBbMBSnNW_Aboh2TpYkNIFMgb&si=2L9RDcX1NSEd6wmN