Installation and Interface of Apache NiFi

Installation methods

Docker based installation

There are several methods for installation of Apache NiFi.

The simplest one is with Docker.

Copy the contents of following file in some directory.

Run command `docker compose up -d`

```
services:
  nifi:
    cap_add:
      - NET_ADMIN
    image: apache/nifi
    container_name: nifi
    environment:
      - SINGLE_USER_CREDENTIALS_USERNAME=admin
      - SINGLE_USER_CREDENTIALS_PASSWORD=ctsBtRBKHRAx69EqUghvvgEvjnaLjFEB
      - NIFI_WEB_HTTP_PORT=8080
      - NIFI_WEB_HTTP_HOST=0.0.0.0
    ports:
      - "8080:8080/tcp"
    volumes:
      - nifi-conf:/opt/nifi/nifi-current/conf
      - nifi_flowfile:/opt/nifi/nifi-current/flowfile_repository
```

```
      - nifi_content:/opt/nifi/nifi-current/content_repository
    restart: unless-stopped

volumes:
  nifi-conf:
  nifi_flowfile:
  nifi_content:
```

Another way to create a nifi instance without docker-compose would be to run following commands

```
docker volume create nifi-conf
docker volume create nifi_flowfile
docker volume create nifi_content
docker run --cap-add NET_ADMIN --name nifi -e SINGLE_USER_CREDENTIALS_USERNAME=admin -e SI
```

Access on `http://localhost:8080/nifi`

Linux Native Installation

Download and extract binary

wget https://dlcdn.apache.org/nifi/1.27.0/nifi-1.27.0-bin.zip

Start using command

```
bin/nifi.sh start
```

FInd Generated UserID and Password inside `logs/nifi-app.log`

Access on `https://localhost:8080/nifi`

Directories and Their Functions

README

- This file usually contains introductory information about NiFi, installation instructions, quick start guides, and links to more detailed documentation.

LICENSE

- Contains the Apache License under which NiFi is distributed.

NOTICE

- Provides notification of the Apache Software Foundation's proprietary rights and includes attributions required by the terms of various open-source licenses.

bin/

- Contains executable scripts to start, stop, and configure NiFi. Main scripts include `nifi.sh` (or `nifi.bat` for Windows).

lib/

- Contains all the necessary libraries and dependencies that NiFi needs to run. This includes Java libraries, third-party libraries, and other runtime dependencies.

logs/

- Default directory for NiFi's log files. Important logs include `nifi-app.log`, `nifi-user.log`, and others for debugging and monitoring NiFi operations.

content_repository/

- **Function**: This repository is used by NiFi to manage the storage of raw content data as it flows through the system. This is crucial for handling large volumes of data efficiently.
- **Details**:
  - NiFi stores content for the FlowFiles here temporarily.
  - It's designed to handle large binary objects and allows NiFi to maintain reliability and data availability if processes need to handle data asynchronously or if the system experiences disruptions.
  - Configuration properties related to this repository can be found in the `nifi.properties` file, including retention policies and disk space management.

database_repository/

- **Function**: Stores metadata about the NiFi dataflow, including tracking processor states and provenance events.
- **Details**:
  - Contains internal databases that NiFi uses to track states and metadata.
  - Essential for maintaining the state of processors, data provenance, and the overall flow management.
  - Highly critical for recovery and ensuring the correctness and completeness of dataflows.
  - Configuration settings can also be found in `nifi.properties`.

flowfile_repository/

- **Function**: Stores the metadata for FlowFiles, which is crucial for managing and tracking the lifecycle of data as it flows through the system.
- **Details**:
  - Stores the attributes and state information of FlowFiles.
  - Helps NiFi manage state and maintain flowfile history, which is crucial for crash recovery and data provenance tracking.
  - Ensures data integrity and robustness by maintaining dataflow continuity even during system restarts or failures.
  - Configurable parameters, like the repository's storage location and related settings, are available in `nifi.properties`.

provenance_repository/

- **Function**: This repository holds the provenance data, which tracks the lineage and history of each FlowFile as it moves through the NiFi flow.
- **Details**:
  - Provides a detailed audit trail, showing where data originated, the transformations it underwent, and its final destination.
  - Essential for troubleshooting, auditing, and compliance.

- Stores vast amounts of data history while ensuring performance and enabling high-speed lookups and queries.
- Configuration includes settings for data retention, indexing, and storage paths, typically specified in `nifi.properties`.

state/

- This folder stores state management files, which are important for coordinating state among cluster nodes and to track the state of NiFi components, processors, and flows across restarts.

extensions/

- A directory where additional components, such as NAR files (NiFi Archive), can be placed to extend the functionality of NiFi with custom processors, controller services, etc.

work/

- Contains temporary files created during the processing of data flows. Used for temporary storage during various processing stages.

run/

- Contains runtime state and is used during the execution of the NiFi process. Typically includes PID files and other run-time specific data.

docs/

- This folder contains documentation files and resources for NiFi. This typically includes the user guide, API documentation, and more.

conf/

- Contains configuration files such as `nifi.properties`, `logback.xml`, and other configuration files that control various settings and behaviors of NiFi.

Configuration Files in `conf` Directory

1. **archive**
   - **Description**: This folder is used for archiving older versions of certain configurations such as `flow.xml.gz`.
2. **authorizations.xml**
   - **Description**: Defines the policies for authorizing user access to various NiFi resources. It describes who can do what within the NiFi instance.
3. **authorizers.xml**
   - **Description**: Configures the authorizer components of NiFi, which are responsible for authorizing user actions. This can include file-based authorization, LDAP-based authorization, etc.
4. **bootstrap-aws.conf**
   - **Description**: Contains bootstrap settings specific to AWS environments. This may include various configurations for integrating with AWS services.
5. **bootstrap-azure.conf**
   - **Description**: Contains bootstrap settings specific to Azure environments. This is used for configuring NiFi to run within Azure cloud services.
6. **bootstrap-gcp.conf**

- **Description**: Contains bootstrap settings specific to Google Cloud Platform environments. This includes settings for integrating with GCP services.

7. **bootstrap-hashicorp-vault.conf**
   - **Description**: Configuration file for integrating with HashiCorp Vault. This is used for obtaining secrets from Vault for secure storage and retrieval.
8. **bootstrap-notification-services.xml**
   - **Description**: Configuration for bootstrap-related notification services. This can be used to notify admins or other systems about NiFi startup, shutdown, or other lifecycle events.
9. **bootstrap.conf**
   - **Description**: Primary configuration for the NiFi bootstrap process. Contains settings for starting and stopping NiFi, JVM options, and other runtime configurations.
10. **flow.xml.gz**
    - **Description**: This file contains the serialized version of the actual data flow configuration. It's compressed in GZIP format and represents the state of the NiFi flow (processors, connections, controller services, etc.).
11. **flow.json.gz**
    - **Description**: Similar to `flow.xml.gz`, but uses a JSON format. It stores the flow configuration in a compressed JSON format.
12. **keystore.p12**
    - **Description**: A PKCS #12 file that contains security certificates and keys for establishing SSL/TLS connections. It's part of NiFi's security configuration.
13. **logback.xml**
    - **Description**: Configuration file for NiFi's logging framework (Logback). It defines what logs are generated, their format, and where they are stored.
14. **login-identity-providers.xml**
    - **Description**: Configures the various identity providers used for user authentication, such as LDAP, Kerberos, or other custom identity providers.
15. **nifi.properties**
    - **Description**: The primary configuration file for NiFi. It contains settings for ports, directories, clustering, security, state management, and numerous other configurations.
16. **state-management.xml**
    - **Description**: Configures how NiFi manages state, often used by stateful processors. This includes settings for local or distributed state management.
17. **stateless-logback.xml**
    - **Description**: Logging configuration specifically for Stateless NiFi, which is a lightweight version of NiFi that doesn't use the traditional flow file repository or other persistence mechanisms.
18. **stateless.properties**
    - **Description**: Properties specific to Stateless NiFi. This file configures how Stateless NiFi operates.
19. **truststore.p12**
    - **Description**: Another PKCS #12 file that contains trusted certificates. It's used to specify which certificates NiFi should trust for SSL/TLS connections.

20. **users.xml**
    - **Description**: Contains details about users, groups, and the mappings between them. This is part of the NiFi security configuration, specifying who is allowed to access the system.
21. **zookeeper.properties**
    - **Description**: Configuration file for ZooKeeper, which NiFi uses for managing clusters. It contains settings for clustering and coordination among multiple NiFi nodes.

nifi.properties configuration params

Core Properties

1. **nifi.flow.configuration.file=./conf/flow.xml.gz**
    - This property points to the configuration file that defines the dataflow within NiFi. It contains the state and configuration of NiFi flows.
2. **nifi.flow.configuration.json.file=./conf/flow.json.gz**
    - Similar to the above property, but specifies the path to the JSON format of the flow configuration file.
3. **nifi.flow.configuration.archive.enabled=true**
    - Enables archiving of older versions of the flow configuration file.
4. **nifi.flow.configuration.archive.dir=./conf/archive/**
    - Specifies the directory where archived versions of the flow configuration file are stored.
5. **nifi.flow.configuration.archive.max.time=30 days**
    - Retention period for archived flow configuration files. Older files are deleted after this period.
6. **nifi.flow.configuration.archive.max.storage=500 MB**
    - Maximum storage allowed for archived flow configuration files. Older files are deleted once this limit is reached.
7. **nifi.flowcontroller.autoResumeState=true**
    - Automatically resumes the flow controller state upon restart.
8. **nifi.flowcontroller.graceful.shutdown.period=10 sec**
    - Time period for gracefully shutting down the NiFi instance.
9. **nifi.flowservice.writedelay.interval=500 ms**
    - Time interval between consecutive writes to the flow configuration file.
10. **nifi.administrative.yield.duration=30 sec**
    - Time duration to wait before retrying to process a FlowFile after an administrative yield.
11. **nifi.bored.yield.duration=10 millis**
    - Interval NiFi processors will wait before checking for more work when no work is available.
12. **nifi.queue.backpressure.count=10000**
    - Maximum number of FlowFiles queued before backpressure is applied.
13. **nifi.queue.backpressure.size=1 GB**

- Maximum cumulative size of FlowFiles queued before backpressure is applied.

Security and Authorization

1. **nifi.authorizer.configuration.file=./conf/authorizers.xml**
   - Path to the configuration file that defines the authorizer components used to authorize user actions.
2. **nifi.login.identity.provider.configuration.file=./conf/login-identity-providers.xml**
   - Path to the configuration file defining the login identity providers used for user authentication.

UI and Appearance

1. **nifi.templates.directory=./conf/templates**
   - Directory where NiFi template files are stored.
2. **nifi.ui.banner.text=**
   - Provides a text banner at the top of the NiFi UI for notifications or status.
3. **nifi.ui.autorefresh.interval=30 sec**
   - Interval at which the NiFi UI should automatically refresh to show updated data.

NAR (NiFi Archive) Management

1. **nifi.nar.library.directory=./lib**
   - Directory for the core NiFi libraries.
2. **nifi.nar.library.autoload.directory=./extensions**
   - Directory where additional NAR files can be auto-loaded.
3. **nifi.nar.working.directory=./work/nar/**
   - Temporary working directory used when unpacking NAR files.

State Management

1. **nifi.state.management.configuration.file=./conf/state-management.xml**
   - Path to the configuration file for state management.
2. **nifi.state.management.provider.local=local-provider**
   - Identifier for the local state provider.
3. **nifi.state.management.provider.cluster=zk-provider**
   - Identifier for the cluster-wide state provider, used when NiFi is in a clustered setup.
4. **nifi.state.management.embedded.zookeeper.start=false**
   - Indicates whether NiFi should start an embedded ZooKeeper server.
5. **nifi.state.management.embedded.zookeeper.properties=./conf/zookeeper.properties**
   - Path to the ZooKeeper properties file, applicable if embedded ZooKeeper is started.

Repositories

Database Repository

1. **nifi.database.directory=./database_repository**

- Directory where the internal database repository is stored, holding metadata about dataflows.

FlowFile Repository

1. **nifi.flowfile.repository.implementation=org.apache.nifi.controller.repository.WriteAheadFlowFileRepository**
   - The implementation class for the FlowFile repository.
2. **nifi.flowfile.repository.directory=./flowfile_repository**
   - Directory for storing FlowFile repository data.
3. **nifi.flowfile.repository.checkpoint.interval=20 secs**
   - Interval at which the FlowFile repository is checkpointed.
4. **nifi.flowfile.repository.always.sync=false**
   - Determines whether FlowFile repository data should be fsynced to disk after every update.
5. **nifi.flowfile.repository.retain.orphaned.flowfiles=true**
   - Specifies whether to retain FlowFiles that are deemed orphaned.

Content Repository

1. **nifi.content.repository.implementation=org.apache.nifi.controller.repository.FileSystemRepository**
   - The implementation class for the Content repository.
2. **nifi.content.repository.directory.default=./content_repository**
   - Default directory for Content repository data.
3. **nifi.content.repository.archive.max.retention.period=7 days**
   - Maximum retention period for archived content.
4. **nifi.content.repository.archive.max.usage.percentage=50%**
   - Maximum disk usage percentage allowed for archived content before triggering cleanup.
5. **nifi.content.repository.archive.enabled=true**
   - Enables archiving of content.

Provenance Repository

1. **nifi.provenance.repository.implementation=org.apache.nifi.provenance.WriteAheadProvenanceRepository**
   - The implementation class for the Provenance repository.
2. **nifi.provenance.repository.directory.default=./provenance_repository**
   - Default directory for Provenance repository data.
3. **nifi.provenance.repository.max.storage.time=30 days**
   - Maximum retention period for provenance data.
4. **nifi.provenance.repository.max.storage.size=10 GB**
   - Maximum storage size for the provenance repository.
5. **nifi.provenance.repository.rollover.time=10 mins**
   - Interval for rolling over the provenance repository.
6. **nifi.provenance.repository.rollover.size=100 MB**

- Maximum size for rolling over the provenance repository.

7. **nifi.provenance.repository.query.threads=2**
   - Number of threads for processing provenance queries.
8. **nifi.provenance.repository.index.threads=2**
   - Number of threads for indexing provenance data.
9. **nifi.provenance.repository.compress.on.rollover=true**
   - Enables compression of provenance data upon rollover.
10. **nifi.provenance.repository.indexed.fields=EventType, FlowFileUUID, Filename, ProcessorID, Relationship**
    - Specifies the fields to index in the provenance data, making them searchable.

Cluster and Network Properties

1. **nifi.cluster.is.node=false**
   - Indicates whether this NiFi instance is part of a cluster.
2. **nifi.cluster.node.address=**
   - The address for this NiFi node when it is part of a cluster.
3. **nifi.cluster.node.protocol.port=**
   - The port used by this NiFi node for cluster communication.
4. **nifi.cluster.node.protocol.max.threads=50**
   - Maximum number of threads for handling cluster communication.
5. **nifi.web.http.host=**
   - The host address to bind the NiFi HTTP web interface.
6. **nifi.web.http.port=**
   - The port to bind the NiFi HTTP web interface.
7. **nifi.web.https.host=127.0.0.1**
   - The host address to bind the NiFi HTTPS web interface.
8. **nifi.web.https.port=8443**
   - The port to bind the NiFi HTTPS web interface.

Security Properties

1. **nifi.sensitive.props.key=8pyodA6hUQclLHa1UAArGMRdPOgqMSXR**
   - The key used to encrypt sensitive properties in the NiFi configuration files.
2. **nifi.security.keystore=./conf/keystore.p12**
   - Path to the keystore file used for SSL/TLS.
3. **nifi.security.keystoreType=PKCS12**
   - The type of keystore used.
4. **nifi.security.keystorePasswd=252506d03490c273383f61272bf1b154**
   - Password for the keystore.

5. **nifi.security.truststore=./conf/truststore.p12**
    - Path to the truststore file used for SSL/TLS.
6. **nifi.security.truststoreType=PKCS12**
    - The type of truststore used.
7. **nifi.security.truststorePasswd=6736666f14e7f369c10dff5e82bf6757**
    - Password for the truststore.

Others

1. **nifi.status.repository.questdb.persist.node.days=14**
    - Number of days to retain node status history in QuestDB.
2. **nifi.monitor.long.running.task.schedule=**
    - Schedule for monitoring long running tasks.
3. **nifi.diagnostics.on.shutdown.enabled=false**
    - Enables automatic diagnostics at shutdown.
4. **nifi.variable.registry.properties=**
    - External properties files for variable registry, supporting a comma-delimited list of file locations.