## What is Apache NiFi?

Put simply, NiFi was built to automate the flow of data between systems. While the term '**data flow**' is used in a variety of contexts, we use it here to mean the automated and managed flow of information between systems. This problem space has been around ever since enterprises had more than one system, where some of the systems created data and some of the systems consumed data.

Some of the high-level challenges of dataflow include:

**Systems fail**

**Data access exceeds capacity to consume**

**Boundary conditions are mere suggestions**

**What is noise one day becomes signal the next**

**Systems evolve at different rates**

**Compliance and security**

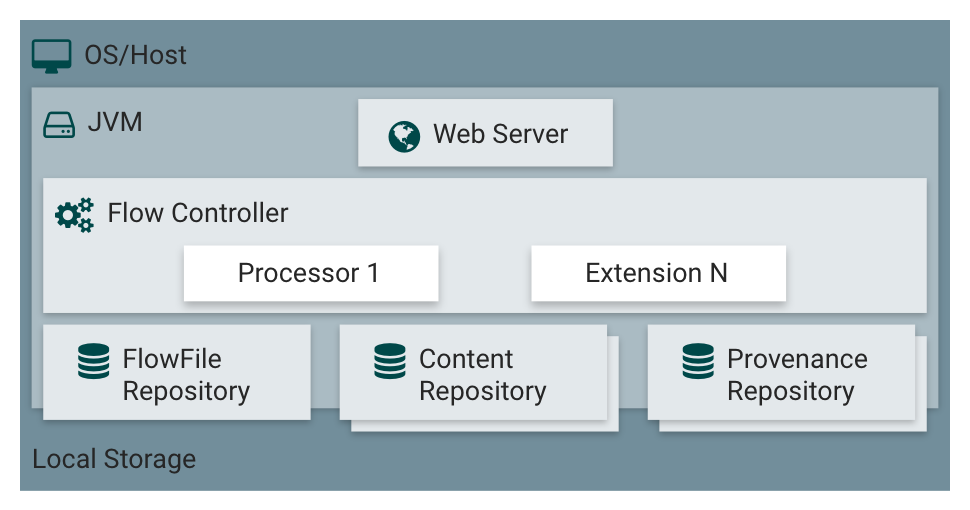
**Continuous improvement occurs in production**

## The core concepts of NiFi

NiFi’s fundamental design concepts closely relate to the main ideas of Flow Based Programming [[fbp]](https://nifi.apache.org/docs/nifi-docs/html/overview.html#fbp).

| **NiFi Term** | **FBP Term** | **Description** |
| --- | --- | --- |
| FlowFile | Information Packet | 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. |
| FlowFile Processor | Black Box | Processors actually perform the work. In [[eip]](https://nifi.apache.org/docs/nifi-docs/html/overview.html#eip) terms a processor is doing some combination of data routing, transformation, or mediation between systems. Processors have access to attributes of a given FlowFile and its content stream. Processors can operate on zero or more FlowFiles in a given unit of work and either commit that work or rollback. |
| Connection | Bounded Buffer | Connections provide the actual linkage between processors. These act as queues and allow various processes to interact at differing rates. These queues can be prioritized dynamically and can have upper bounds on load, which enable back pressure. |
| Flow Controller | Scheduler | The Flow Controller maintains the knowledge of how processes connect and manages the threads and allocations thereof which all processes use. The Flow Controller acts as the broker facilitating the exchange of FlowFiles between processors. |
| Process Group | subnet | A Process Group is a specific set of processes and their connections, which can receive data via input ports and send data out via output ports. In this manner, process groups allow creation of entirely new components simply by composition of other components. |

## NiFi Architecture



NiFi executes within a JVM on a host operating system. The primary components of NiFi on the JVM are as follows:

**Web Server**

The purpose of the web server is to host NiFi’s HTTP-based command and control API.

**Flow Controller**

The flow controller is the brains of the operation. It provides threads for extensions to run on, and manages the schedule of when extensions receive resources to execute.

**Extensions**

There are various types of NiFi extensions which are described in other documents. The key point here is that extensions operate and execute within the JVM.

**FlowFile Repository**

The FlowFile Repository is where NiFi keeps track of the state of what it knows about a given FlowFile that is presently active in the flow. The implementation of the repository is pluggable. The default approach is a persistent Write-Ahead Log located on a specified disk partition.

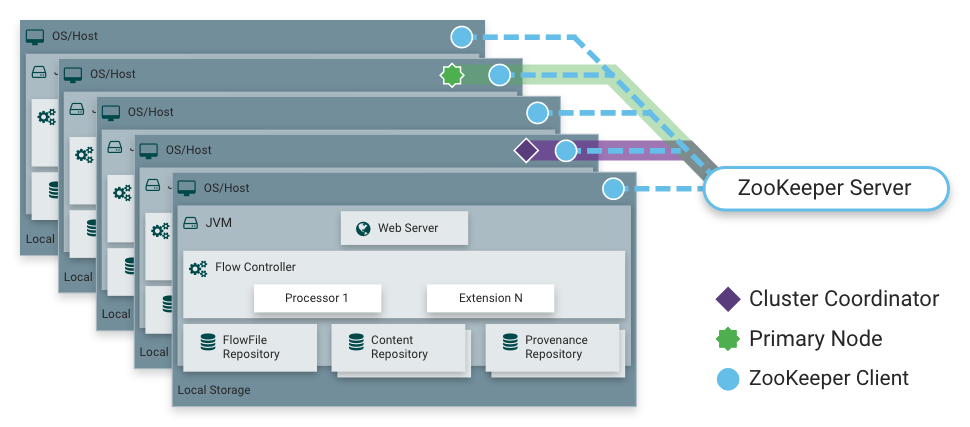
**Content Repository**

The Content Repository is where the actual content bytes of a given FlowFile live. The implementation of the repository is pluggable. The default approach is a fairly simple mechanism, which stores blocks of data in the file system. More than one file system storage location can be specified so as to get different physical partitions engaged to reduce contention on any single volume.

**Provenance Repository**

The Provenance Repository is where all provenance event data is stored. The repository construct is pluggable with the default implementation being to use one or more physical disk volumes. Within each location event data is indexed and searchable.

NiFi is also able to operate within a cluster.



Each node in a NiFi cluster performs the same tasks on the data, but each operates on a different set of data. Apache ZooKeeper elects a single node as the Cluster Coordinator, and failover is handled automatically by ZooKeeper. All cluster nodes report heartbeat and status information to the Cluster Coordinator. The Cluster Coordinator is responsible for disconnecting and connecting nodes. Additionally, every cluster has one Primary Node, also elected by ZooKeeper. As a DataFlow manager, you can interact with the NiFi cluster through the user interface (UI) of any node. Any change you make is replicated to all nodes in the cluster, allowing for multiple entry points.

## High Level Overview of Key NiFi Features

**This section provides a 20,000 foot view of NiFi’s cornerstone fundamentals, so that you can understand the Apache NiFi big picture, and some of its most interesting features. The key features categories include flow management, ease of use, security, extensible architecture, and flexible scaling model.**

## Terminology Used in This Guide

In order to talk about NiFi, there are a few key terms that readers should be familiar with. We will explain those NiFi-specific terms here, at a high level.

**FlowFile**: Each piece of "User Data" (i.e., data that the user brings into NiFi for processing and distribution) is referred to as a FlowFile. A FlowFile is made up of two parts: Attributes and Content. The Content is the User Data itself. Attributes are key-value pairs that are associated with the User Data.

**Processor**: The Processor is the NiFi component that is responsible for creating, sending, receiving, transforming, routing, splitting, merging, and processing FlowFiles. It is the most important building block available to NiFi users to build their dataflows.

**NiFi is an open source data flow framework. It is highly automated for flow of data between systems. It works as a data transporter between data producer and data consumer. Producer means the system that generates data and consumer means the other system that consumes data. NiFi ensures to solve high complexity, scalability, maintainability and other major challenges of a Big Data pipeline.**

**NiFi is used extensively in Energy and Utilities, Financial Services, Telecommunication , Healthcare and Life Sciences, Retail Supply Chain, Manufacturing and many others.**

**Commonly used sources are data repositories, flat files, XML, JSON, SFTP location, web servers, HDFS and many others.**

**Destinations can be S3, NAS, HDFS, SFTP, Web Servers, RDBMS, Kafka etc.,**

**—---------------------------------------------------------------------------------------------------------**

**Installation of NIFI**

brew install nifi

or

<https://nifi.apache.org/download.html>

navigate to the directory where NiFi was installed :-

To run NiFi in the foreground, run

bin/nifi.sh run

To run NiFi in the background, instead run

bin/nifi.sh start

To check the status and see if NiFi is currently running, execute the command bin/nifi.sh status

NiFi can be shutdown by executing the command

bin/nifi.sh stop

Issuing bin/nifi.sh start executes the nifi.sh script that starts NiFi in the background and then exits.

If you want nifi.sh to wait for NiFi to finish scheduling all components before exiting, use the --wait-for-init flag with an optional timeout specified in seconds: bin/nifi.sh start --wait-for-init 120.

If the timeout is not provided, the default timeout of 15 minutes will be used.

If NiFi was installed with Homebrew, run the commands nifi start or nifi stop from anywhere in your file system to start or stop NiFi.

### 

### Installing as a Service

To install the application as a service, navigate to the installation directory in a Terminal window and execute the command bin/nifi.sh install to install the service with the default name nifi.

To specify a custom name for the service, execute the command with an optional second argument that is the name of the service. For example, to install NiFi as a service with the name dataflow, use the command bin/nifi.sh install dataflow.

Once installed, the service can be started and stopped using the appropriate commands, such as sudo service nifi start and sudo service nifi stop. Additionally, the running status can be checked via sudo service nifi status.

## Started NiFi. Now What?

The default installation generates a random username and password, writing the generated values to the application log. The application log is located in logs/nifi-app.log under the installation directory.

The log file will contain lines with Generated Username [USERNAME] and Generated Password [PASSWORD] indicating the credentials needed for access. Search the application log for those lines and record the generated values in a secure location.

The following command can be used to change the username and password:

$ ./bin/nifi.sh set-single-user-credentials <username> <password>

Now that NiFi has been started, we can bring up the User Interface (UI) in order to create and monitor our dataflow. To get started, open a web browser and navigate to <https://localhost:8443/nifi>.

The port can be changed by editing the nifi.properties file in the NiFi conf directory, but the default port is 8443.

—------------------------------------------------------------------------------------------------------------------

## Overview

The NiFi Toolkit contains several command line utilities to setup and support NiFi in standalone and clustered environments. The utilities include:

* CLI — The cli tool enables administrators to interact with NiFi and NiFi Registry instances to automate tasks such as deploying versioned flows and managing process groups and cluster nodes.
* Encrypt Config — The encrypt-config tool encrypts the sensitive keys in the *nifi.properties* file to facilitate the setup of a secure NiFi instance.
* File Manager — The file-manager tool enables administrators to backup, install or restore a NiFi installation from backup.
* Flow Analyzer — The flow-analyzer tool produces a report that helps administrators understand the max amount of data which can be stored in backpressure for a given flow.
* Node Manager — The node-manager tool enables administrators to perform status checks on nodes as well as the ability to connect, disconnect, or remove nodes from the cluster.
* Notify — The notify tool enables administrators to send bulletins to the NiFi UI.
* S2S — The s2s tool enables administrators to send data into or out of NiFi flows over site-to-site.
* TLS Toolkit — The tls-toolkit utility generates the required keystores, truststore, and relevant configuration files to facilitate the setup of a secure NiFi instance.
* ZooKeeper Migrator — The zk-migrator tool enables administrators to:
  + move ZooKeeper information from one ZooKeeper cluster to another
  + migrate ZooKeeper node ownership

## NiFi CLI

This tool offers a CLI focused on interacting with NiFi and NiFi Registry in order to automate tasks, such as deploying flows from a NIFi Registy to a NiFi instance or managing process groups and cluster nodes.

### Usage

The CLI toolkit can be executed in standalone mode to execute a single command, or interactive mode to enter an interactive shell.

To execute a single command:

./bin/cli.sh <command> <args>

To launch the interactive shell:

./bin/cli.sh

To show help:

./bin/cli.sh -h

The following are available commands:

demo quick-import

nifi current-user

nifi cluster-summary

nifi connect-node

nifi delete-node

nifi disconnect-node

nifi get-root-id

nifi get-node

nifi get-nodes

nifi offload-node

nifi list-reg-clients

nifi create-reg-client

nifi update-reg-client

nifi get-reg-client-id

nifi pg-import

nifi pg-start

nifi pg-stop

nifi pg-create

nifi pg-get-vars

nifi pg-set-var

nifi pg-get-version

nifi pg-change-version

nifi pg-get-all-versions

nifi pg-list

nifi pg-status

nifi pg-get-services

nifi pg-create-service

nifi pg-enable-services

nifi pg-disable-services

nifi pg-get-param-context

nifi pg-set-param-context

nifi pg-replace

nifi get-services

nifi get-service

nifi create-service

nifi enable-services

nifi disable-services

nifi get-reporting-tasks

nifi get-reporting-task

nifi create-reporting-task

nifi delete-reporting-task

nifi start-reporting-tasks

nifi stop-reporting-tasks

nifi list-users

nifi create-user

nifi list-user-groups

nifi create-user-group

nifi update-user-group

nifi get-policy

nifi update-policy

nifi list-templates

nifi download-template

nifi upload-template

nifi list-param-contexts

nifi get-param-context

nifi create-param-context

nifi delete-param-context

nifi set-inherited-param-contexts

nifi remove-inherited-param-contexts

nifi set-param-provider-reference

nifi remove-param-provider-reference

nifi set-param

nifi delete-param

nifi export-param-context

nifi import-param-context

nifi merge-param-context

nifi list-param-providers

nifi get-param-provider

nifi create-param-provider

nifi delete-param-provider

nifi fetch-params

nifi set-param-provider-property

nifi get-access-token

nifi get-access-token-spnego

nifi logout-access-token

registry current-user

registry list-buckets

registry create-bucket

registry delete-bucket

registry list-flows

registry create-flow

registry delete-flow

registry list-flow-versions

registry export-flow-version

registry import-flow-version

registry sync-flow-versions

registry transfer-flow-version

registry diff-flow-versions

registry upload-bundle

registry upload-bundles

registry list-bundle-groups

registry list-bundle-artifacts

registry list-bundle-versions

registry download-bundle

registry get-bundle-checksum

registry list-extension-tags

registry list-extensions

registry list-users

registry create-user

registry update-user

registry list-user-groups

registry create-user-group

registry update-user-group

registry get-policy

registry update-policy

registry update-bucket-policy

registry get-access-token

registry get-access-token-spnego

registry logout-access-token

session keys

session show

session get

session set

session remove

session clear

exit

help

To display extensive help for a specific command:

./bin/cli.sh <command> -h