**Pig Latin**

Pig is a high-level programming language useful for analyzing large data sets. A pig was a result of development effort at Yahoo!

* Extract
* Transform
* Load
* Research and data processing

In a MapReduce framework, programs need to be translated into a series of Map and Reduce stages. However, this is not a programming model which data analysts are familiar with. So, in order to bridge this gap, an abstraction called Pig was built on top of Hadoop.

Apache Pig enables people to focus more on **analyzing bulk data sets and to spend less time writing Map-Reduce programs.**Similar to Pigs, who eat anything, the Pig programming language is designed to work upon any kind of data. That's why the name, Pig.

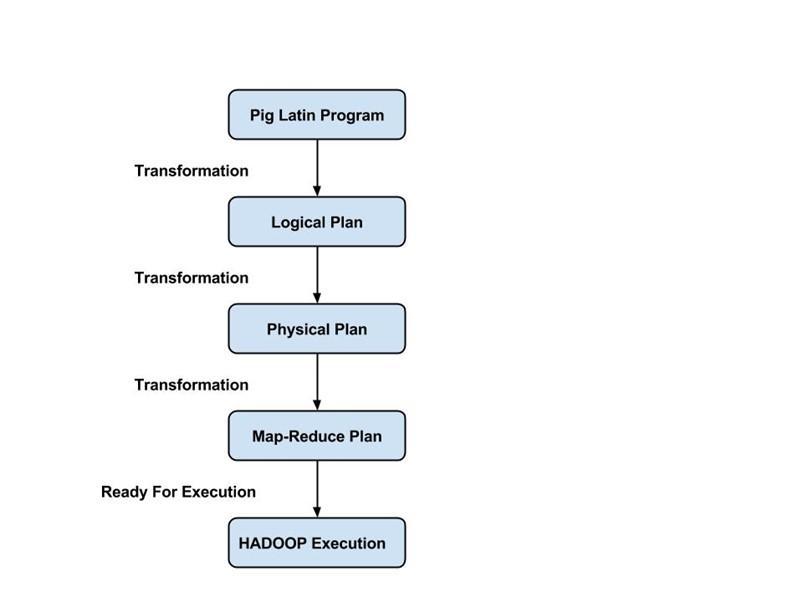
**Pig Architecture**

Pig consists of two components:

1. **Pig Latin,** which is a language
2. **A runtime environment,** for running PigLatin programs.

A Pig Latin program consists of a series of operations or transformations which are applied to the input data to produce output. These operations describe a data flow which is translated into an executable representation, by Pig execution environment. Underneath, results of these transformations are series of MapReduce jobs which a programmer is unaware of. So, in a way, Pig allows the programmer to focus on data rather than the nature of execution.

PigLatin is a relatively stiffened language which uses familiar keywords from data processing e.g., Join, Group and Filter.

[](https://www.guru99.com/images/Big_Data/061114_1128_INTRODUCTIO2.jpg)

PIG Architecture

**Execution modes:**

Pig has two execution modes:

1. Local mode: In this mode, Pig runs in a single JVM and makes use of local file system. This mode is suitable only for analysis of small datasets using Pig
2. Map Reduce mode: In this mode, queries written in Pig Latin are translated into MapReduce jobs and are run on a Hadoop cluster (cluster may be pseudo or fully distributed). MapReduce mode with the fully distributed cluster is useful of running Pig on large datasets.

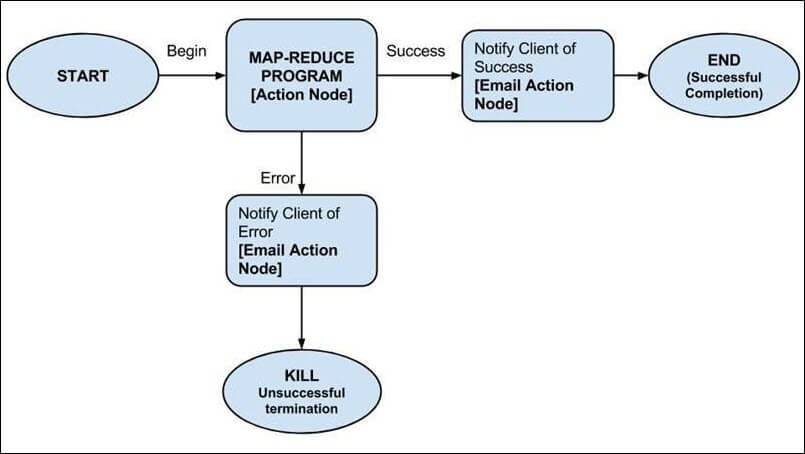
## ****What is OOZIE?****

Apache Oozie is a workflow scheduler for Hadoop. It is a system which runs the workflow of dependent jobs. Here, users are permitted to create **Directed Acyclic Graphs** of workflows, which can be run in parallel and sequentially in Hadoop.

It consists of two parts:

* Workflow engine: Responsibility of a workflow engine is to store and run workflows composed of Hadoop jobs e.g., MapReduce, Pig, Hive.
* **Coordinator engine**: It runs workflow jobs based on predefined schedules and availability of data.

Oozie is scalable and can manage the timely execution of thousands of workflows (each consisting of dozens of jobs) in a Hadoop cluster.

[[](https://www.guru99.com/images/Big_Data/061114_1137_LearnOOZIEi1.jpg)](https://www.guru99.com/images/Big_Data/061114_1137_LearnOOZIEi1.jpg)

Oozie is very much flexible, as well. One can easily start, stop, suspend and rerun jobs. Oozie makes it very easy to rerun failed workflows. One can easily understand how difficult it can be to catch up missed or failed jobs due to downtime or failure. It is even possible to skip a specific failed node.

**Why use Oozie?**

The main purpose of using Oozie is to manage different type of jobs being processed in Hadoop system.

Dependencies between jobs are specified by a user in the form of Directed Acyclic Graphs. Oozie consumes this information and takes care of their execution in the correct order as specified in a workflow. That way user's time to manage complete workflow is saved. In addition, Oozie has a provision to specify the frequency of execution of a particular job.

**Features of Oozie**

* Oozie has client API and command line interface which can be used to launch, control and monitor job from Java application.
* Using its Web Service APIs one can control jobs from anywhere.
* Oozie has provision to execute jobs which are scheduled to run periodically.
* Oozie has provision to send email notifications upon completion of jobs.

## ****How does OOZIE work?****

Oozie runs as a service in the cluster and clients submit workflow definitions for immediate or later processing.

Oozie workflow consists of **action nodes** and **control-flow nodes**.

An**action node** represents a workflow task, e.g., moving files into HDFS, running a MapReduce, Pig or[Hive](https://www.guru99.com/hive-tutorials.html)jobs, importing data using Sqoop or running a shell script of a program written in Java.

A**control-flow node** controls the workflow execution between actions by allowing constructs like conditional logic wherein different branches may be followed depending on the result of earlier action node.

**Start Node**, **End Node**, and **Error Node** fall under this category of nodes.

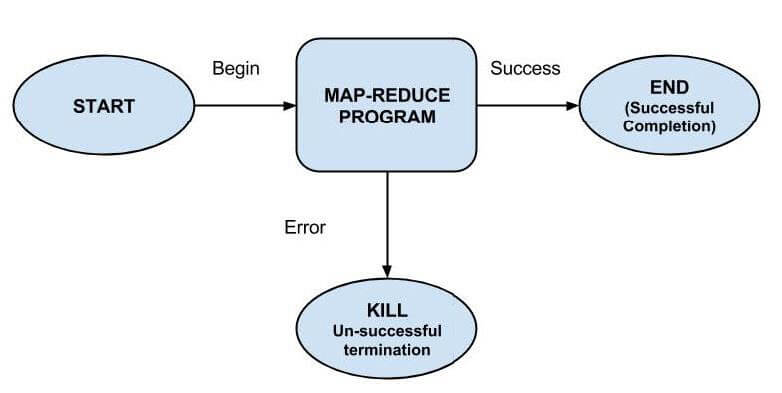
**Start Node,** designates the start of the workflow job.

**End Node,** signals end of the job.

**Error Node** designates the occurrence of an error and corresponding error message to be printed.

At the end of execution of a workflow, HTTP callback is used by Oozie to update the client with the workflow status. Entry-to or exit from an action node may also trigger the callback.

## ****Example Workflow Diagram****

[](https://www.guru99.com/images/pimgpsh_fullsize_distr.jpg)