Lesson 5

Data Pipelines

In this lesson we will talk about:

Data pipelines. Origins

Data Capture, Transforming & Analysing

Raw Data. Data Aggregation

Pipeline efficiency

Basic statistical functions

Origins

2025. Present time

IBM

"A data pipeline is a method in which raw data is ingested from various data sources, transformed and then ported to a data store, such as a data lake or data warehouse, for analysis."

"Data pipelines act as the piping for data science projects or business intelligence dashboards. Data can be sourced through a wide variety of places: APIs, SQL and NoSQL databases, files."

"During sourcing, data lineage is tracked to document the relationship between enterprise data in various business and IT applications, for example, where data is currently and how its stored in an environment, such as on-premises, in a data lake or in a data warehouse."

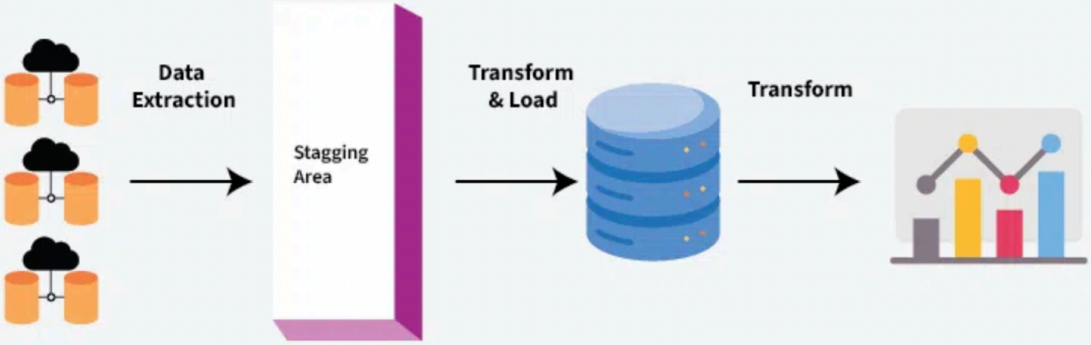
Source: ibm.com

www.geeksforgeeks.org

"Data Pipeline deals with information that is flowing from one end to another. In simple words, we can say collecting the data from various resources than processing it as per requirement and transferring it to the destination by following some sequential activities."

Source: www.geeksforgeeks.org

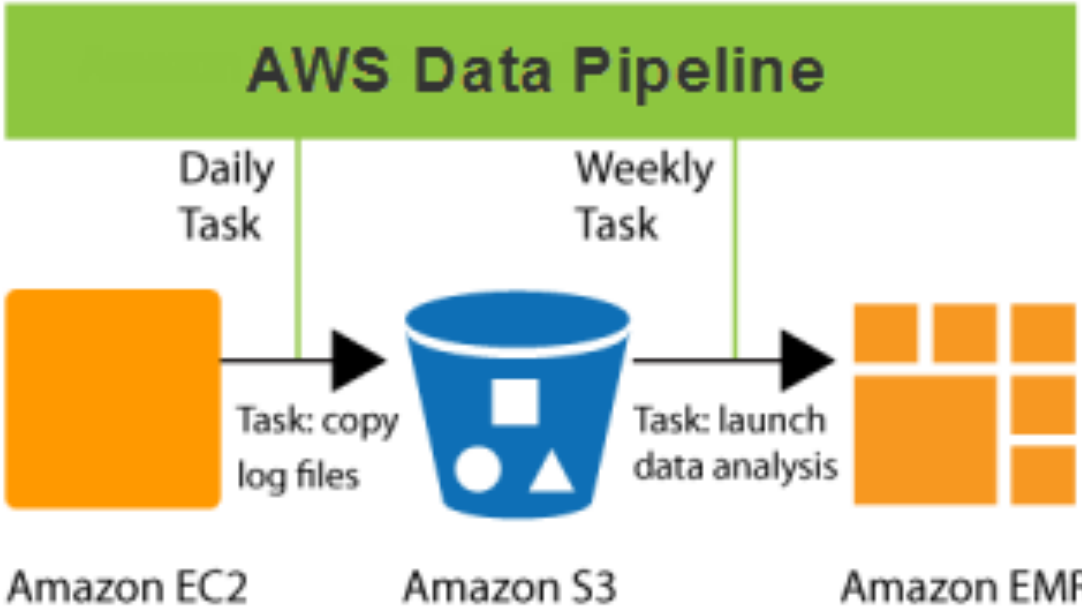
Data Pipeline Overview



AMAZON

"A data pipeline is a series of processing steps to prepare enterprise data for analysis. Organizations have a large volume of data from various sources like applications, Internet of Things (IoT) devices, and other digital channels. However, raw data is useless; it must be moved, sorted, filtered, reformatted, and analyzed for business intelligence. A data pipeline includes various technologies to verify, summarize, and find patterns in data to inform business decisions."

"Just like a water pipeline moves water from the reservoir to your taps, a data pipeline moves data from the collection point to storage. A data pipeline extracts data from a source, makes changes, then saves it in a specific destination. We explain the critical components of data pipeline architecture below."

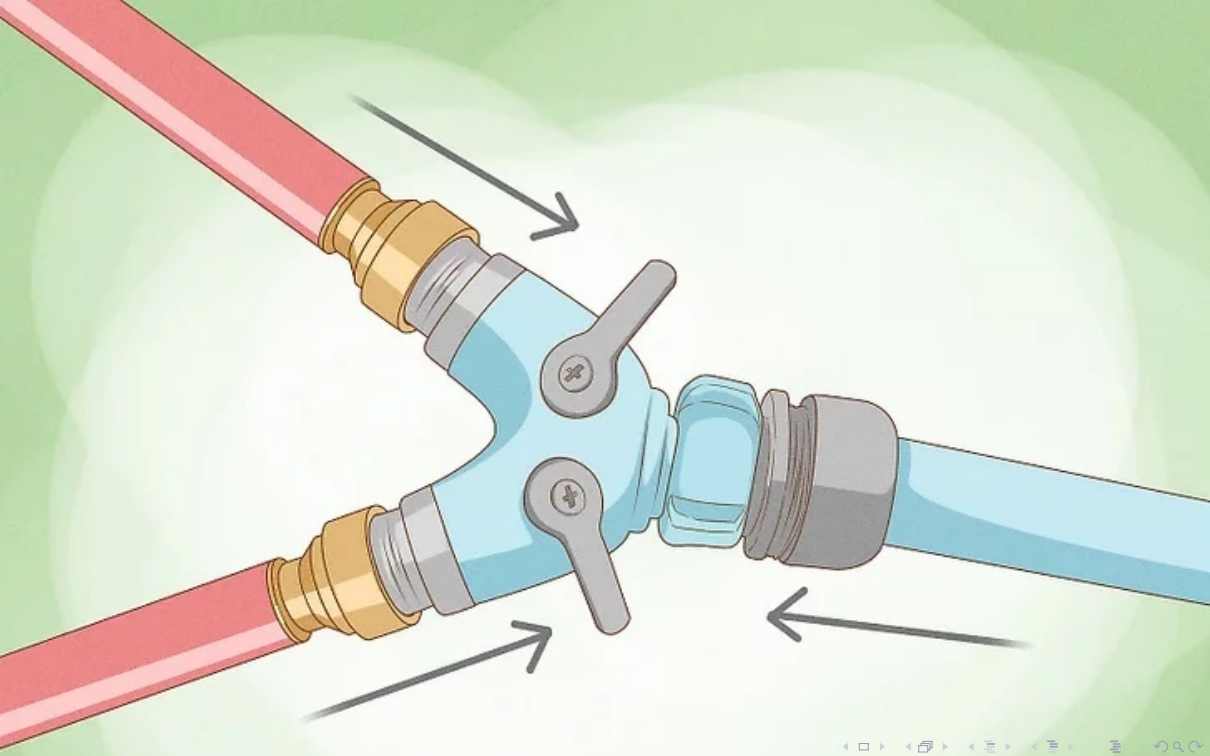

AWS Data Pipeline
Daily Task
Weekly Task
Task: copy log files
Task: launch data analysis
Amazon EC2
Amazon S3
Amazon EMR

So, is a data pipeline a program or what? Lets rollback time...

1969. UNIX

UNIX pipelines

As early as 1964 Douglas McIlroy thought about a mechanism how we could connect programs same way like we connect and combine garden hoses together.

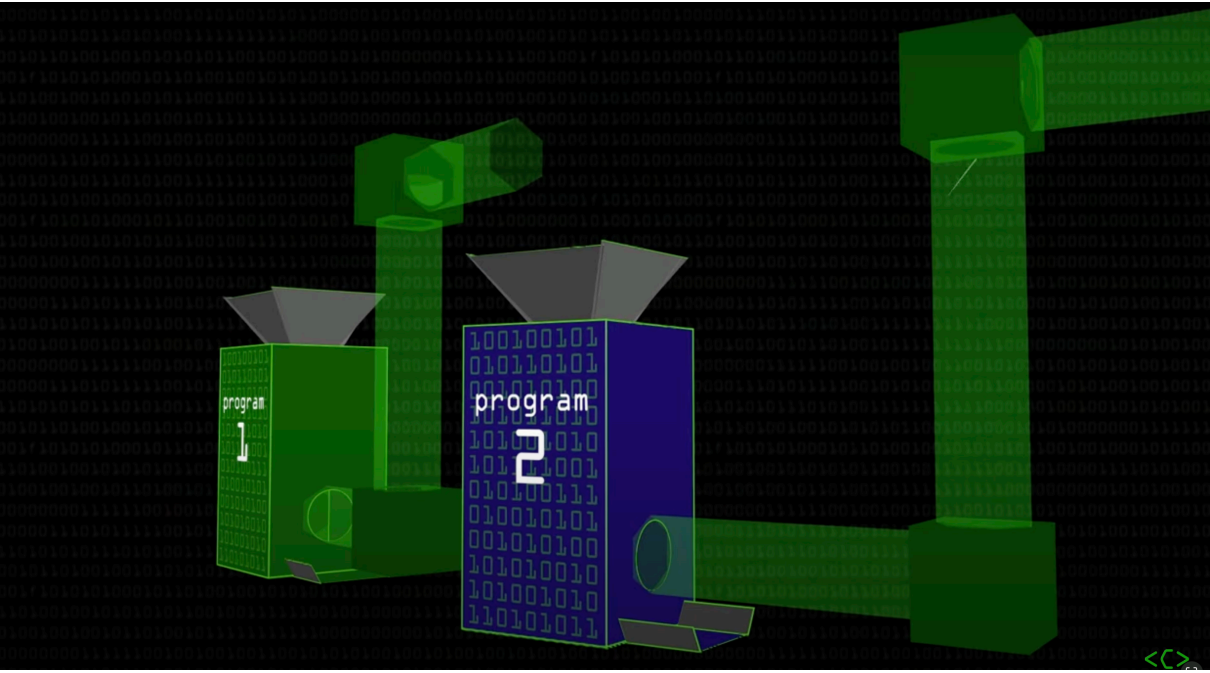


Connecting Garden Hoses

- it should connect different hoses
- as efficiently as possible
- with no leaks
- covering all the garden

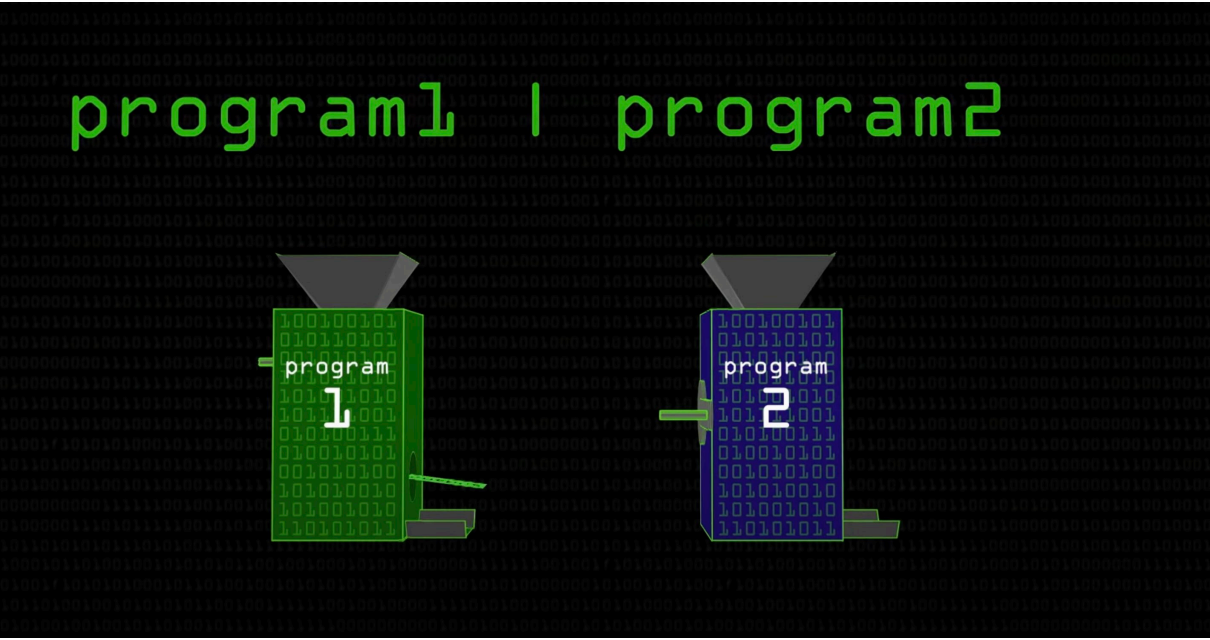
UNIX pipelines

The very first data pipeline was implemented for UNIX based operating systems. An operator '—' was used to connect one program to another program.


program
1
program
2

"A pipeline is a mechanism for connecting the output of one program directly and conveniently into the input of another program."

— *Brian Kernighan, UNIX father*


program1 | program2
program
1
program
2

UNIX pipelines

But why connect different programs instead of having a single one processing all the input?

The pipeline: is connecting one or many programs together! It is not a single monolithic program.

Idea was to have small, **modular** programs which can connect together to solve a problem very efficient and as modular as possible

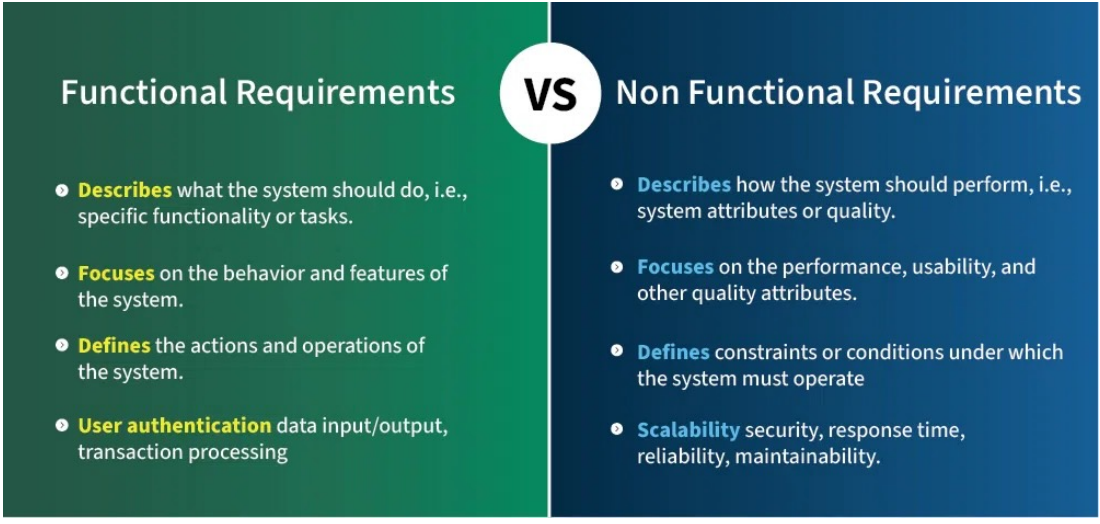
Small programs vs. A big ones

- smaller programs are easy to develop
- and are simple and cheaper to maintain
- as well the amount of data might not fit one program
- no temporary files or data in between

Functional specification

Helps us to understand what our program will do. Its highly important to write first a specification of our program before implementing and developing anything.

A **functional software specification** is a written description of what our program is supposed to do.


Functional Requirements
VS
Non Functional Requirements
Describes what the system should do, i.e., specific functionality or tasks.
Focuses on the behavior and features of the system.
Defines the actions and operations of the system.
User authentication data input/output, transaction processing
Describes how the system should perform, i.e., system attributes or quality.
Focuses on the performance, usability, and other quality attributes.
Defines constraints or conditions under which the system must operate
Scalability security, response time, reliability, maintainability.

Functional specification

Helps us to understand what our program will do. Its highly important to write first a specification of our program before implementing and developing anything.

The specification – what our program does.

But how should we describe this specification? How can we be precise? What does it mean to be precise?

Imprecision can lead to **ERRORS!**

Precise specifications

Its difficult to be precise using English or Chinese. Thats why in science and engineering precise specifications have adopted basic maths to describe the specifications.

Basic math

Elementary mathematics. Propositional logic.

Propositional Logic

Elementary algebra is the mathematics of real numbers and the operators $+$, $-$, $*$ (multiplication), and $/$ (division). Propositional logic is the mathematics of the two Boolean values TRUE and FALSE and the five operators whose names (and common pronunciations) are

$\wedge$	conjunction (and)	$\Rightarrow$	implication (implies)
$\vee$	disjunction (or)	$\equiv$	equivalence (is equivalent to)
$\neg$	negation (not)		

Propositional Logic, cont.

To learn how to compute with numbers, you had to memorize addition and multiplication tables and algorithms for calculating with multidigit numbers. Propositional logic is much simpler, since there are only two values, TRUE and FALSE. To learn how to compute with these values, all you need to know are the following definitions of the five Boolean operators:

- $\wedge$ $F \wedge G$ equals TRUE iff both F and G equal TRUE.
- $\vee$ $F \vee G$ equals TRUE iff F or G equals TRUE (or both do).
- $\neg$ $\neg F$ equals TRUE iff F equals FALSE.
- $\Rightarrow$ $F \Rightarrow G$ equals TRUE iff F equals FALSE or G equals TRUE (or both).
- $\equiv$ $F \equiv G$ equals TRUE iff F and G both equal TRUE or both equal FALSE.

Formal methods

In computer science, **formal methods** are mathematically rigorous techniques for the specification, development, analysis, and verification of software and hardware systems.[1] The use of formal methods for software and hardware design is motivated by the expectation that, as in other engineering disciplines, performing appropriate mathematical analysis can contribute to the reliability and robustness of a design.[2]

Formal methods employ a variety of theoretical computer science fundamentals, including logic calculi, formal languages, automata theory, control theory, program semantics, type systems, and type theory.[3]

Specifications using formal methods

Formal methods may be used to give a formal description of the system to be developed, at whatever level of detail desired. Further formal methods may depend on this specification to synthesize a program or to verify the correctness of a system.

Alternatively, specification may be the only stage in which formal methods is used. By writing a specification, ambiguities in the informal requirements can be discovered and resolved. Additionally, engineers can use a formal specification as a reference to guide their development processes.[4]

Formal verification is the use of software tools to prove properties of a formal specification, or to prove that a formal model of a system implementation satisfies its specification.

Once a formal specification has been developed, the specification may be used as the basis for proving properties of the specification.

Why maths?

- Quantify output behavior
- Predict behavior of systems as design time
- Understand safety margins
- Understand system interactions

Verification: Are we building the product right?

Validation: Are we building the right product?

"Building the product right" checks that the specifications are correctly implemented by the system while "building the right product" refers back to the user's needs. Ideally, formal methods provide a mathematical guarantee that software meets its specifications.

Industry


03 | 20 | 2015
DRAFT 29837611-0004
SCALE: 01 : 01
amazon
web services™
ENGINEERING TEAM
AWS-0910
PROJECT
AMAZON WEB SERVICES
A3WWSI6

Industry: Amazon, Microsoft, NASA, ESA, Intel

DOI:10.1145/2699417

Engineers use TLA+ to prevent serious but subtle bugs from reaching production.

BY CHRIS NEWCOMBE, TIM RATH, FAN ZHANG, BOGDAN MUNTEANU, MARC BROOKER, AND MICHAEL DEARDEUFF

How Amazon Web Services Uses Formal Methods

TLA+ Specifications of the Consistency Guarantees Provided by Cosmos DB