Member-only story

Complete Roadmap to becoming a Data Engineer

Vishal Barvaliya · Follow
Published in Towards Data Engineering
25 min read · Mar 18, 2023

Listen Share

Data engineering is a rapidly growing field that has become increasingly important in the age of big data. A data engineer is responsible for designing, building, and maintaining the infrastructure that allows for the collection, processing, and storage of large amounts of data. Data engineers work closely with data scientists and analysts to ensure that the data they need is available in the right format and at the right time. In this article, we will discuss a complete roadmap to becoming a data engineer.

Photo by Scott Graham on Unsplash

Pre-requisites

As a data engineer, you will need to work with various operating systems, and Linux is a popular choice due to its robustness, flexibility, and availability of open-source tools. Here are some important operating system and Linux topics to learn:

1. **Operating System Fundamentals:** You should have a good understanding of the fundamentals of operating systems, such as process management, memory management, file systems, and networking.

2. **Linux Commands:** You should learn about essential Linux commands such as navigating the file system, creating and managing files and directories, using pipes and filters, managing users and permissions, and managing processes.

3. **Shell Scripting:** Shell scripting is used to automate tasks in Linux. You should learn how to write shell scripts using Bash or other shell

languages for data processing, file manipulation, and system administration.

Step 1: Learn the basics of programming

The first step to becoming a data engineer is to learn the basics of programming. The most common programming languages used in data engineering are Python, Java, and Scala. Start by learning one of these languages and becoming proficient in it ***(I would recommend starting with Python as it's the most widely used programming language in data engineering)***. You should learn how to write code, debug, and optimize your code for performance. You should also learn basic data structures and algorithms.

Python

Python is one of the most popular programming languages used in data engineering. As a data engineer, you will be responsible for designing, building, and maintaining the infrastructure for collecting, processing, and storing large amounts of data. Here are some important Python topics to learn to become a data engineer:

1. **Basic Python Programming:** You should have a solid understanding of the fundamentals of programming in Python, including data types **(strings, integers, floats, etc.**), control structures **(if-else statements, for and while loops)**, **functions, modules, and libraries.** You should be familiar with Python's built-in functions and data types, as well as popular libraries such as **NumPy, Pandas, and Matplotlib.**

- Understanding **data types** such as integers, floats, strings, and booleans
- Using **control structures** such as if/else statements and loops
- Creating **functions** to perform specific tasks

- Importing **modules** and **libraries** to extend Python's capabilities
- Understanding and using **built-in data structures** such as lists, tuples, and dictionaries

2. Object-Oriented Programming (OOP): Object-oriented programming is a programming paradigm that revolves around the concept of objects, which have attributes and methods. To use OOP in Python, you should learn **how to create classes and objects, inheritance** (the ability of a class to inherit methods and attributes from its parent class), **polymorphism** (the ability of different objects to respond to the same method in different ways), and **encapsulation** (the ability to hide implementation details of a class from outside code).

- **Creating classes** to define objects and their attributes
- **Using inheritance** to create new classes from existing ones
- **Polymorphism** to use the same method with different objects
- **Encapsulation** to protect data and methods from external access

3. Data Structures: Data structures are used to store and organize data in a way that is efficient for processing. In Python, you should learn about **lists** (ordered, mutable sequences of values), **tuples** (ordered, immutable sequences of values), **dictionaries** (unordered collections of key-value pairs), and sets (unordered collections of unique values). You should also learn how to manipulate these data structures using built-in functions and methods.

- Understanding and using **lists** to store and manipulate collections of data
- Using **tuples** to store data that should not be changed
- Creating and using **dictionaries** to store key-value pairs

- Using **sets** to store unique items

4. File Input and Output: Reading and writing data to files is a common task in data engineering. In Python, you should learn how to use the built-in file I/O functions to read and write data to files, as well as how to handle errors that may occur during file I/O operations.

- **Opening and closing files** using Python's built-in file I/O functions
- **Reading data from files** using various file modes
- **Writing data to files** using various file modes
- Working with **CSV** and **JSON** and other file formats

5. Regular Expressions: Regular expressions **(regex)** are a powerful tool for pattern matching and text manipulation. In Python, you should learn how to use regex to search for and extract data from text using patterns.

- Understanding and using **regular expressions** to match patterns in strings
- Using **special characters** and **metacharacters** to refine patterns
- Using regular expressions in Python to **extract data from text**

6. Exception Handling: Exception handling is a way to handle errors and exceptions that may occur during the execution of a program. In Python, you should learn how to use **try-except blocks** to catch and handle exceptions, as well as how to raise your own exceptions when necessary.

- Using **try-except blocks** to handle errors and exceptions
- Creating **custom exceptions** to handle specific cases

- Using **finally blocks** to perform cleanup actions

7. Functional Programming: Functional programming is a programming paradigm that focuses on the use of functions to create modular, reusable code. In Python, you should learn about **lambda functions** (anonymous functions that can be defined on the fly), as well as built-in functions such as **map** (apply a function to each item in a sequence), **filter** (return a sequence of items that satisfy a condition), and **reduce** (apply a function to a sequence to reduce it to a single value).

- Understanding and using **lambda functions** to create anonymous functions
- Using **map, filter, and reduce functions** to manipulate data
- **Comprehensions** to create new data structures in a concise way

8. Concurrency and Parallelism: Concurrency and parallelism are techniques for executing multiple tasks at the same time, which is essential for processing large datasets efficiently. In Python, you should learn about **threading** (a way to run multiple threads of execution in a single process), **multiprocessing** (a way to run multiple processes in parallel), and **asynchronous programming** (a way to execute tasks without blocking other tasks). You should also be familiar with libraries such as **asyncio** and **concurrent**. futures, which provide tools for asynchronous and parallel programming in Python.

- Using **threading** to create and manage multiple threads in Python
- Using **multiprocessing** to execute multiple processes in parallel
- Understanding **asynchronous** programming and using **async/await syntax** to write **asynchronous** code

By learning and mastering these Python topics, a data engineer can effectively manipulate data, build and maintain data pipelines, and create scalable and efficient data processing systems.

Must-do Python topics for Data Engineers

Python is the most commonly used programming language for data engineering. Python has a large and...

medium.com

Step 2: Learn SQL

SQL is the standard language used for querying and manipulating data in relational databases. As a data engineer, you will be working with databases a lot, so it is important to learn SQL. You should be able to write SQL queries to extract data from databases, and you should also learn how to optimize your queries for performance.

Here are some important SQL topics that a data engineer should know:

1. Data Definition Language (DDL):

DDL is used to define and modify the structure of a database, such as creating tables and defining constraints. A data engineer should know how to use SQL commands such as CREATE, ALTER, and DROP to define and modify database objects.

- Creating tables using **CREATE TABLE** command
- Modifying tables using **ALTER TABLE** command
- Dropping tables using the **DROP TABLE** command
- Adding and dropping columns using **ALTER TABLE** command
- Creating and modifying constraints using **ALTER TABLE** command

2. Data Manipulation Language (DML):

DML is used to manipulate the data in a database, such as inserting, updating, and deleting records. A data engineer should know how to use SQL commands such as SELECT, INSERT, UPDATE, and DELETE to manipulate data.

- Inserting data into a table using the **INSERT** command
- Updating data in a table using the **UPDATE** command
- Deleting data from a table using the **DELETE** command
- Querying data from a table using the **SELECT** command
- Filtering data using the **WHERE** clause
- Sorting data using **ORDER BY** clause
- Grouping data using **GROUP BY** clause
- Filtering grouped data using the **HAVING** clause

3. Joins:

Joins are used to combine data from multiple tables. A data engineer should know how to use SQL commands such as INNER JOIN, LEFT JOIN, and RIGHT JOIN to join tables.

- **Inner join** to select rows that have matching values in both tables
- **Left join** to select all rows from the left table and matching rows from the right table
- **Right join** to select all rows from the right table and matching rows from the left table
- **Full outer join** to select all rows from both tables

4. Subqueries:

Subqueries are queries that are embedded inside another query. A data engineer should know how to use subqueries to create complex queries that involve multiple tables.

- **Correlated subqueries** to reference values from the outer query
- **Non-correlated subqueries** to execute independently of the outer query
- **Scalar subqueries** to return a single value
- **Multiple-row subqueries** to return multiple rows

5. Aggregation Functions:

Aggregation functions are used to calculate summary statistics on data, such as the average, maximum, and minimum values. A data engineer should know how to use SQL commands such as SUM, AVG, MAX, and MIN to calculate summary statistics.

- **COUNT** to count the number of rows in a table or a group
- **SUM** to calculate the sum of a column or a group of rows
- **AVG** to calculate the average of a column or a group of rows
- **MAX** to find the maximum value of a column or a group of rows
- **MIN** to find the minimum value of a column or a group of rows

6. Query Optimization:

Query optimization is the process of improving the performance of SQL queries. A data engineer should know how to use techniques such as indexing, partitioning, and denormalization to optimize SQL queries.

- Creating **indexes** to **speed up** SELECT queries
- **Partitioning** large tables to improve performance

- Denormalizing tables to **reduce JOINs**
- Using **EXPLAIN** command to analyze the query execution plan

7. Transactions:

Transactions are used to ensure the integrity of data in a database by grouping multiple SQL statements into a single atomic unit. A data engineer should know how to use SQL commands such as BEGIN TRANSACTION, COMMIT, and ROLLBACK to manage transactions.

- Starting a transaction using **BEGIN TRANSACTION** command
- Committing a transaction using the **COMMIT** command
- Rolling back a transaction using the **ROLLBACK** command

8. Stored Procedures:

Stored procedures are precompiled SQL statements that can be called from other SQL statements. A data engineer should know how to create and use stored procedures to simplify complex queries and improve performance.

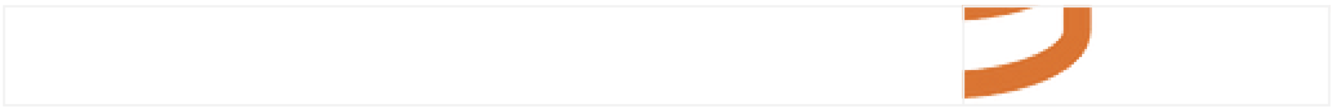
- Creating a stored procedure using **CREATE PROCEDURE** command
- Executing a stored procedure using **EXECUTE** command
- **Passing parameters** to a stored procedure
- **Returning values** from a stored procedure

By mastering these SQL topics, a data engineer can effectively manipulate data, optimize SQL queries, and create efficient and scalable databases.

Must-do SQL topics for Data Engineers

1. Data Modeling.

medium.com



Step 3: Learn Data Structures & Algorithms (Average Level, No Hard level):

Data structures and algorithms are fundamental concepts that every data engineer should know. Here is a brief explanation of some of the most important topics:

1. **Arrays:** An array is a collection of similar data types. Data engineers should know how to create, manipulate and traverse arrays.
2. **Strings**: A string is a sequence of characters. Data engineers should know how to perform string manipulation, substring searches and pattern matching.
3. **Linked List:** A linked list is a linear data structure that consists of nodes where each node contains a data field and a reference to the next node. Data engineers should know how to create, traverse and manipulate linked lists.
4. **Stack**: A stack is a data structure that allows data to be inserted and removed from one end only. Data engineers should know how to implement stack operations like push, pop and peek.
5. **Queue**: A queue is a data structure that allows data to be inserted at one end and removed from the other end. Data engineers should know how to implement queue operations like enqueue and dequeue.
6. **Tree**: A tree is a hierarchical data structure consisting of nodes, where each node has a parent node and zero or more child nodes. Data engineers should know about the different types of trees, like binary trees and their traversals.

7. **Graph:** A graph is a non-linear data structure consisting of nodes and edges. Data engineers should know about the different types of graphs and their representations, like adjacency matrices and adjacency lists.

8. **Dynamic Programming:** Dynamic programming is a technique to solve problems by breaking them down into smaller sub-problems and solving each sub-problem only once. Data engineers should know how to apply dynamic programming to solve problems.

9. **Searching:** Searching is the process of finding a specific element in a collection of elements. Data engineers should know about linear and binary search algorithms.

10. **Sorting:** Sorting is the process of arranging data in a particular order. Data engineers should know about different sorting algorithms like the bubble sort, insertion sort, quicksort, and merge sort.

Step 4: Learn data modeling and schema design

Data modeling and schema design are important skills for a data engineer. You should learn how to design data models that are efficient and scalable. You should also learn how to design database schemas that are optimized for performance and ease of use.

1. Entity Relationship Diagrams (ERDs):

Entity Relationship Diagrams are used to model the relationships between different entities in a database. Entities are things that you want to store data about, such as customers, products, or orders. Attributes describe the characteristics of an entity, such as the customer’s name, address, or phone number. Relationships define how entities are related to each other, such as one-to-one, one-to-many, or many-to-many.

- Understanding and creating ERDs to model data relationships
- Identifying entities, attributes, and relationships between them
- Using ERDs to create database schemas

2. Normalization:

Normalization is a technique used to organize data in a database to reduce redundancy and ensure data consistency. There are different forms of normalization, such as the first normal form (1NF), second normal form (2NF), third normal form (3NF), and so on. Each form has a set of rules to ensure that the data is organized correctly and there are no data anomalies or inconsistencies.

- Understanding normalization and its importance in data modeling
- The different forms of normalization, from the first normal form (1NF) to the Boyce-Codd normal form (BCNF)
- Applying normalization to ensure data consistency and reduce redundancy in the database

3. Dimensional Modeling:

Dimensional modeling is a technique used in data warehousing to model data in a way that is optimized for analytical queries. It involves identifying fact tables and dimension tables and creating star schemas or snowflake schemas. Fact tables contain the measures or metrics that you want to analyze, such as sales, revenue, or clicks. Dimension tables contain the attributes that describe the context of the fact table, such as time, product, or location.

- Understanding dimensional modeling and its use in data warehousing
- Identifying fact tables and dimension tables

- Creating star schemas and snowflake schemas

4. Data Integrity:

Data integrity ensures that data in a database is accurate and consistent. This can be achieved by defining constraints such as primary keys, foreign keys, and unique constraints. Primary keys are unique identifiers for each record in a table, foreign keys ensure that the relationships between tables are maintained, and unique constraints ensure that no two records have the same value for a particular attribute.

- Ensuring data integrity by defining constraints such as primary keys, foreign keys, and unique constraints
- Using triggers and stored procedures to enforce business rules

5. Performance Optimization:

Performance optimization involves designing the database schema in a way that optimizes query performance. This can include partitioning large tables into smaller ones, creating indexes to speed up data retrieval, and optimizing queries to minimize the amount of data that needs to be scanned.

- Optimizing database performance through schema design
- Partitioning large tables to improve query performance
- Creating indexes to speed up data retrieval

6. NoSQL Databases:

NoSQL databases are non-relational databases that provide flexibility, scalability, and high availability. They are used in applications where traditional relational databases may not be the best fit. There are different types of NoSQL databases, such as document-based, key-value, and graph databases, each with its own strengths and weaknesses.

- Understanding NoSQL databases and their use cases
- Working with document-based, key-value, and graph databases
- Understanding the trade-offs between consistency, availability, and partition tolerance

By mastering these data modeling and schema design topics, a data engineer can create efficient and effective database schemas that support data analysis and decision-making. They can also design and build data warehouses, work with NoSQL databases, and ensure data integrity and performance.

Step 5: Learn data warehousing

Data warehousing is the process of storing data in a centralized repository that can be easily accessed and analyzed. You should learn how to design and build data warehouses using tools like Amazon Redshift, Snowflake, or Google BigQuery. You should also learn how to optimize your data warehouses for performance and scalability.

1. ETL (Extract, Transform, Load):

ETL is the process of extracting data from various sources, transforming it into a format suitable for analysis, and then loading it into a data warehouse. The different stages of ETL include:

- **Data profiling:** This involves analyzing the quality and consistency of the data being extracted, such as identifying missing or duplicate records.
- **Data cleansing:** This is the process of cleaning up the data to remove errors, inconsistencies, and other anomalies.
- **Data transformation:** This involves converting the data into a format that can be used for analysis, such as converting data types, creating new fields, and aggregating data.

- **Data integration:** This is the final step in ETL and involves loading the transformed data into a data warehouse.

2. Data Modeling:

Data modeling is the process of designing the structure of a database or data warehouse. There are three main types of data models:

- **Conceptual data model:** This is a high-level view of the data and describes the relationships between different entities.
- **Logical data model:** This is a more detailed view of the data and includes the tables, columns, and relationships between them.
- **Physical data model:** This is the actual implementation of the data model and includes details such as data types, indexes, and

into a single, unified view. The different stages of data integration include:

- **Data profiling:** This involves analyzing the quality and consistency of the data being integrated.
- **Data mapping:** This is the process of identifying how data from different sources should be mapped to each other.
- **Data transformation:** This involves converting the data into a format that can be used for analysis, such as converting data types, creating new fields, and aggregating data.

4. Dimensional Modeling:

Dimensional modeling is a technique used for designing data warehouses that are optimized for reporting and analysis. The key concepts of dimensional modeling include:

- **Fact tables:** These contain the measures or metrics that are being analyzed, such as sales revenue or customer counts.
- **Dimension tables:** These contain the attributes or dimensions that provide context for the measures, such as date, product, or location.
- **Star schema:** This is a simple dimensional model where a single fact table is connected to multiple dimension tables.
- **Snowflake schema:** This is a more complex dimensional model where the dimension tables are normalized to reduce redundancy.

5. OLAP (Online Analytical Processing):

OLAP is a technology used for analyzing large, multidimensional datasets. Some key concepts of OLAP include:

- **Multidimensional cubes:** These are structures that allow data to be analyzed across multiple dimensions, such as time, product, and location.
- **Slicing and dicing:** This is the ability to analyze data by selecting a subset of the dimensions and measures.
- **Drill down/drill up:** This is the ability to analyze data at different levels of granularity, such as moving from monthly to daily data.

6. Data Governance:

Data governance is the process of managing the availability, usability, integrity, and security of the data used in an organization. Some key concepts of data governance include:

- **Data policies:** These are rules and guidelines that define how data should be managed, such as data retention policies or data access policies.

- **Data quality:** This is the process of ensuring that data is accurate, complete, and consistent.
- **Compliance:** This involves ensuring that the data meets legal and regulatory requirements, such as GDPR or HIPAA.

Data Warehousing: A Guide for Data Engineers
Introduction:
medium.com

Step 6: Basic Terminologies in Big Data:

1. **Big Data:** Refers to extremely large datasets that cannot be processed by traditional data processing tools.
2. **5 V's of Big Data:** The 5 V's of Big Data are Volume, Velocity, Variety, Veracity, and Value. They represent the key characteristics of Big Data that make it different from traditional data.
3. **Distributed Computation:** In Big Data, computation is distributed across multiple machines in a cluster. This allows for faster processing of large datasets.
4. **Distributed Storage:** Big Data is stored across multiple machines in a cluster. This allows for better scalability and reliability.
5. **Vertical vs Horizontal Scaling:** Vertical scaling involves adding more resources to a single machine, while horizontal scaling involves adding more machines to a cluster.
6. **Commodity Hardware:** Refers to inexpensive, off-the-shelf hardware that is used in clusters for Big Data processing.
7. **Clusters:** A cluster is a group of machines that work together to process and store Big Data.

8. **File Formats:** Refers to the different file formats used for storing Big Data, including CSV, JSON, AVRO, Parquet, and ORC.
9. **Structured Data:** Refers to data that is organized in a specific format, such as a database.
10. **Unstructured Data:** Refers to data that does not have a specific format, such as text files or social media posts.
11. **Semi-Structured Data:** Refers to data that has some structure, but not a fully defined schema, such as XML or JSON data.

Step 7: Learn Big Data Frameworks

Big data technologies are a set of tools and frameworks that are used to process and analyze large datasets. Some of the most popular big data technologies are Hadoop, Spark, Hive, and Kafka. You should learn how to use these tools and become proficient in them. You should also learn how to work with distributed systems and how to optimize your code for performance in a distributed environment.

I. Apache Hadoop (Architecture Understanding Most Imp)

Apache Hadoop is a framework for distributed processing of large data sets across clusters of computers. It consists of the Hadoop Distributed File System (HDFS) for distributed storage, MapReduce for distributed processing, and YARN (Yet Another Resource Negotiator) for cluster resource management.

Topics to learn

- HDFS
- Map-Reduce
- Yarn

HDFS

1. **HDFS architecture:** Understanding the architecture of HDFS, including the Namenode, Datanode, and Secondary Namenode.
2. **HDFS commands:** Learning important HDFS commands for managing files and directories in HDFS, such as ls, cp, mv, rm, mkdir, and chmod.
3. **Data replication:** Understanding data replication in HDFS and how it helps in providing fault tolerance.
4. **Block size:** Understanding the importance of block size in HDFS and how it affects the performance and storage utilization.
5. **HDFS security:** Understanding the security features of HDFS, including authentication, authorization, and encryption.
6. **HDFS federation:** Learning about HDFS federation, which allows multiple independent HDFS namespaces to be served by a single HDFS cluster.
7. **HDFS high availability:** Understanding how to configure HDFS for high availability, which ensures that the Namenode remains available even in case of failure.
8. **HDFS performance tuning:** Learning how to tune HDFS performance by adjusting parameters such as block size, replication factor, and buffer sizes.
9. **Hadoop cluster monitoring**: Learning how to monitor the Hadoop cluster and the HDFS filesystem using tools such as Hadoop logs, JMX, and Hadoop metrics.
10. **HDFS backups:** Understanding the importance of taking regular backups of the HDFS filesystem and learning how to perform backups using tools such as DistCp.

Map-Reduce

Since Map-Reduce is replaced by Apache Spark so we don't need to learn much about Map-Reduce so, understanding architecture is enough no need to learn about map-reduce jobs and all.

YARN

1. **YARN Architecture:** Understanding the architecture of YARN is important to get a clear understanding of how YARN works and how it interacts with other Hadoop components.

2. **YARN Resource Management:** YARN is responsible for resource management, so it's important to understand how it allocates resources to different applications running on the cluster.

3. **YARN Node Manager:** NodeManager is a YARN component responsible for managing the resources on a single node. Understanding NodeManager is important to troubleshoot resource allocation issues.

4. **YARN Application Master:** The Application Master is responsible for managing the lifecycle of an application and coordinating with the Resource Manager for resource allocation.

5. **YARN Containers:** Containers are the fundamental unit of computation in YARN. They encapsulate the application code and necessary resources and are launched on different nodes in the cluster.

6. **YARN Scheduling:** Understanding the different scheduling policies and how to configure them is important to ensure optimal resource utilization.

7. **YARN High Availability:** High availability is crucial for ensuring that the Resource Manager is always available, even in the event of a failure. Understanding how to configure and set up YARN for high availability is important for production environments.

8. **YARN Security:** YARN provides a number of security features, such as authentication and authorization. Understanding how to configure and use these security features is important for secure cluster operation.

II. Apache Hive:

To work with Hive effectively, you should learn the following topics:

- **How to load data in different file formats:** Hive supports various file formats, such as CSV, JSON, AVRO, Parquet, and ORC. You should learn how to load data from these file formats into Hive tables.
- **Internal Tables:** Hive supports two types of tables: Internal Tables and External Tables. Internal tables are managed by Hive, and the data is stored in a Hive-managed directory. You should learn how to create and use internal tables in Hive.
- **External Tables:** External tables are not managed by Hive, and the data is stored in an external directory. You should learn how to create and use external tables in Hive.
- **Querying table data stored in HDFS:** HiveQL is similar to SQL, and you can use it to query data stored in HDFS. You should learn how to write HiveQL queries to retrieve data from Hive tables.
- **Partitioning:** Hive supports partitioning, which allows you to divide large tables into smaller, more manageable parts based on one or more columns. You should learn how to create partitioned tables in Hive.
- **Bucketing:** Bucketing is a technique to further divide data into smaller, more manageable parts. You should learn how to create bucketed tables in Hive.

- **Map-Side Join:** Map-side join is an optimization technique that improves the performance of join operations in Hive. You should learn how to use map-side join in Hive.
- **Sorted-Merge Join:** Sorted-merge join is another optimization technique that improves the performance of join operations in Hive. You should learn how to use sorted-merge join in Hive.
- **UDFs in Hive:** User-Defined Functions (UDFs) allow you to extend the functionality of Hive by creating custom functions. You should learn how to create and use UDFs in Hive.
- **SerDe in Hive:** SerDe stands for Serializer/Deserializer, which is used to serialize and deserialize data in Hive. You should learn how to use SerDe in Hive to work with non-standard data formats.

III Apache Spark (Most Important):

Apache Spark is an open-source distributed computing system used for big data processing and analytics. It provides an interface for programming entire clusters with implicit data parallelism and fault tolerance.

Some important topics to learn for data engineers in Apache Spark include:

1. **Spark Core:** This is the foundation of the Apache Spark framework, and it includes the basic components needed to run Spark applications. The topics to learn under Spark Core include:

- RDDs (Resilient Distributed Datasets)
- Transformations and Actions
- Spark Context
- Shared Variables

- Caching

2. Spark SQL: This module provides support for structured and semi-structured data. The topics to learn under Spark SQL include:

- DataFrame and Dataset API
- Spark SQL Functions
- Joins and Aggregations
- Window Functions

3. Spark Streaming: This module enables the processing of real-time streaming data. The topics to learn under Spark Streaming include:

- DStreams (Discretized Streams)
- Window Operations
- Stateful Operations
- Spark Streaming Sources and Sinks

4. Spark MLlib: This module provides support for machine learning algorithms. The topics to learn under Spark MLlib include:

- Basic Statistics
- Regression and Classification
- Clustering
- Collaborative Filtering

5. GraphX: This module provides support for graph processing and analytics. The topics to learn under GraphX include:

- Graph Operators and Algorithms
- Pregel API
- GraphFrames API

6. Deployment: This includes topics related to deploying Spark applications in production environments. The topics to learn under Deployment include:

- Cluster Managers (YARN, Mesos, Kubernetes)
- Cluster Configurations
- Resource Management and Scheduling
- Monitoring and Logging

7. Optimization: This includes topics related to optimizing Spark applications for performance. The topics to learn under Optimization include:

- Memory Management
- Caching Strategies
- Partitioning and Shuffling
- Broadcast Variables and Accumulators

8. Integration: This includes topics related to integrating Spark with other tools and frameworks. The topics to learn under Integration include:

- Apache Hadoop and HDFS
- Apache Kafka

- Apache Cassandra
- Apache Hive and Impala

These are the key topics to learn for Apache Spark. However, it's worth noting that the Spark ecosystem is constantly evolving, so it's important to stay up-to-date with the latest releases and developments.

Overall, becoming proficient in Apache Spark involves learning its core concepts, understanding its architecture, and mastering its various modules and libraries.

Must-Do Apache Spark Topics for Data Engineering Interviews

Apache Spark is an open-source big data processing framework that provides a flexible and powerful...

medium.com

Step 8: Learn cloud technologies

Cloud technologies are becoming increasingly popular for storing and processing data. Some of the most popular cloud platforms for data engineering are AWS, Azure, and Google Cloud. You should learn how to use these platforms and become proficient in them. You should also learn how to work with cloud-based databases and how to optimize your code for performance in a cloud environment.

AWS

As a data engineer, it is important to have knowledge of cloud computing platforms, especially Amazon Web Services (AWS), which is one of the most popular cloud platforms. Here are some important AWS topics to learn:

1. **AWS Fundamentals:** Understand the basics of AWS and its services, such as EC2, S3, VPC, IAM, etc.
2. **AWS Data Storage Services:** Learn about AWS data storage services, such as S3, EBS, EFS, and Glacier, and their use cases.
3. **AWS Compute Services:** Learn about AWS compute services, such as EC2, Lambda, Elastic Beanstalk, and their use cases.
4. **AWS Database Services:** Learn about AWS database services, such as RDS, DynamoDB, Redshift, and their use cases.
5. **AWS Big Data Services:** Learn about AWS big data services, such as EMR, Kinesis, Glue, and their use cases.
6. **AWS Networking Services:** Learn about AWS networking services, such as VPC, Direct Connect, Route 53, and their use cases.
7. **AWS Security:** Understand the security aspects of AWS, such as IAM, KMS, Security Groups, and best practices for securing data in AWS.
8. **AWS Monitoring and Logging:** Learn about AWS monitoring and logging services, such as CloudWatch, CloudTrail, and their use cases.
9. **AWS Certification:** Consider obtaining AWS certifications to demonstrate your knowledge and skills in AWS, such as the AWS Certified Data Analytics — Specialty certification.

Overall, having a strong understanding of AWS and its services can greatly benefit a data engineer’s career, as more and more organizations are moving their data and workloads to the cloud.

Azure

1. **Azure Data Factory:** A cloud-based ETL service that enables the creation, scheduling, and orchestration of data pipelines that move

and transform data from various sources.

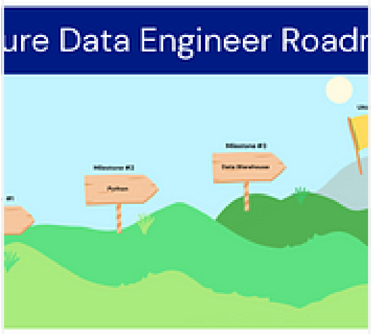
2. **Azure Databricks:** A collaborative, cloud-based platform for data engineering, machine learning, and analytics. It is built on Apache Spark and provides an interactive workspace for data exploration, experimentation, and model development.
3. **Azure Cosmos DB:** A globally distributed, multi-model database service that supports NoSQL data models, including document, graph, key-value, and column family.
4. **Azure Synapse Analytics:** A cloud-based analytics service that brings together big data and data warehousing. It provides a unified experience for data integration, big data analytics, and data warehousing.
5. **Azure SQL Database:** A fully managed relational database service that provides high availability, security, and scalability for mission-critical applications.
6. **Azure HDInsight:** A fully managed cloud service that makes it easy to process big data using popular open-source frameworks such as Hadoop, Spark, Hive, and HBase.
7. **Azure Stream Analytics:** A real-time analytics service that enables the processing of streaming data from various sources, including IoT devices, social media, and logs.
8. **Azure Machine Learning:** A cloud-based machine learning service that enables the creation, deployment, and management of machine learning models.
9. **Azure Data Lake Storage:** A scalable and secure cloud-based data lake that enables the storage and analysis of large amounts of unstructured, semi-structured, and structured data.

10. **Azure Event Hubs:** A highly scalable, real-time data ingestion service that can collect and process millions of events per second from various sources.

Complete Roadmap to become Azure Data Engineer

As the amount of data generated by businesses continues to grow exponentially, the need for skilled...

medium.com



5 Essential Azure Services for Data Engineers in 2023

Data engineering is a field that deals with managing, processing, and storing data. With the advent of big...

medium.com

GCP

1. **Google Cloud Storage:** Learn how to store and retrieve data using Google Cloud Storage. This includes creating buckets, uploading files, and setting access controls

2. **Google BigQuery:** Learn how to work with BigQuery, which is a fully-managed, serverless data warehouse that enables super-fast SQL queries using the processing power of Google's infrastructure.

3. **Google Cloud Dataflow:** Learn how to use Dataflow, which is a fully-managed service for creating data processing pipelines. You can use Dataflow to transform and enrich data, and to stream data into BigQuery, Cloud Storage, or other data stores

4. **Google Cloud Pub/Sub:** Learn how to use Pub/Sub, which is a messaging service that allows you to send and receive messages

between independent applications. You can use Pub/Sub to build real-time data processing pipelines

5. **Google Cloud Dataproc:** Learn how to use Dataproc, which is a fully-managed service for running Apache Hadoop, Apache Spark, and other big data tools on GCP
6. **Google Cloud SQL:** Learn how to use Cloud SQL, which is a fully-managed relational database service that makes it easy to set up, maintain, manage, and administer your MySQL, PostgreSQL, and SQL Server databases on GCP.
7. **Google Cloud Spanner:** Learn how to use Cloud Spanner, which is a globally-distributed, strongly consistent, and horizontally-scalable relational database service that lets you store and query structured data with SQL semantics.
8. **Google Cloud Composer:** Learn how to use Composer, which is a fully-managed service for creating and managing workflows built on Apache Airflow.
9. **Google Cloud Functions:** Learn how to use Cloud Functions, which is a serverless platform that lets you run your code in response to events and automatically manages the underlying infrastructure for you.
10. **Google Cloud Kubernetes Engine:** Learn how to use Kubernetes Engine, which is a fully-managed service for deploying, managing, and scaling containerized applications on GCP.

Topics to learn for each of these services include, but are not limited to, understanding the service architecture, learning how to interact with the service through the command line or API, setting up access controls, and optimizing performance.