THE DATA ENGINEERING SKILLS & TOOLS GUIDE

ANDREAS KRETZ
Founder & CEO

LearnDataEngineering.com

CONTENT

If you have any questions regarding this document either:

- Write me an email to: hello@learndataengienering.com
- Or join our Discord server for direct contact: https://discord.gg/Wxy2mQA7Fy

SKILLS GUIDE
FOR DATA ENGINEERS

What skills do Data Engineers have? And what's the difference between a Junior and Senior Data Engineer? In the matrix below I show you examples for Junior Data Engineer, Data Engineer, Senior Data Engineer, Data Architect and Machine Learning Engineer.

Roles as columns and skill categories as rows.

One difference between a junior and full professional role that is worth mentioning is the level of self-sufficiency. A junior most of the time gets told what to do while professional and senior roles usually develop their goals & work-packages themselves.

You'll see that Senior Data Engineer and Data Architect are quite on the same level. That means you need experience as a Data Engineer to do Architecture.

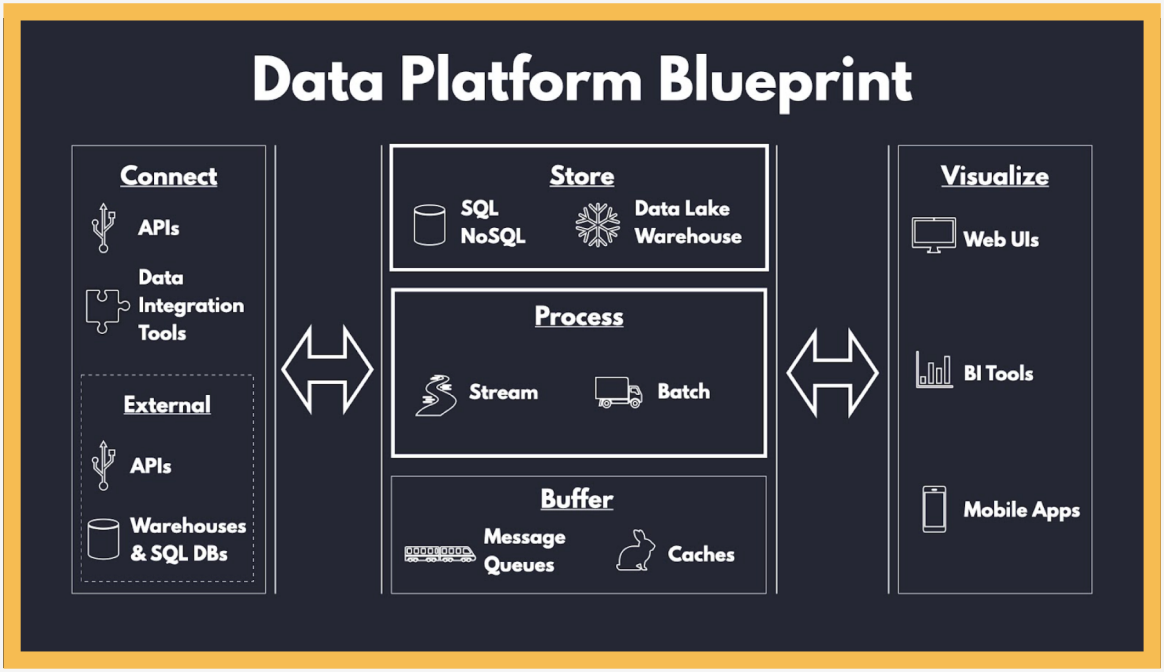
Data Engineer and ML Engineer also are roughly on the same level. The ML Engineer just has this added ML knowledge and focus

SKILLS GUIDE FOR DATA ENGINEERS

Andreas Kretz LearnDataEngineering.com	Junior Data Engineer	Data Engineer	Senior Engineer	Data Architect	ML Engineer
Min. Experience	0 Years	2 Years	5+	5+	2 Years
Main Work	Implementation	Implementation	Implementation, Design, Agile manager	Design	Implementation
Agile Development	Gets tasks assigned	Develops own tasks	Manages tasks for whole team	Develops own tasks	Develops own tasks
Team Leadership	-	Mentor for Juniors	Engineering Team	Mentor for Juniors	Mentor for Juniors
Collaboration	With Team	With Team & Customers	Team, Customers, Business Lead	Team, Customers, Business Lead	With Team & Customers, Data Scientists (teams)
Project management	-	-	Yes	Yes	-
Software Development	Mid Level developer	Experienced developer in 1 or 2 languages	Experienced developer 1 or 2 + basics in multiple languages	Experienced developer (Not needed but nice to have)	Experienced developer in 1 or 2 languages
SQL	Basic: queries & understand DB design	Complex: queries & DB do db design	Complex: queries & DB do db design, optimization	Complex: queries & DB do db design, optimization	Complex: queries & DB Design
Machine Learning	Basic understanding of process	Basic understanding of process	Complex understanding of tools and processes	Complex understanding of tools and processes	Complex understanding of tools and processes, can create own ML models (Part time Data Scientist?)
CI/CD	Usage	Usage & Implement CI/CD processes	Usage & Implement CI/CD processes	Design CI/CD strategy	Usage & Implement CI/CD processes for ML
DevOps	Basic DevOps skills using tools	Implementation of DevOps pipelines	Design & Implementation of DevOps strategy	Design DevOps strategy	Implementation of DevOps pipelines
Cloud Platforms	Single Platform & Key Open Source Tools	Multiple Platforms & Open Source Tools	Multiple Platforms & Open Source Tools	Expert in multiple platforms & open source tools	Multiple Platforms & Open Source Tools
Domain knowledge	Broad knowledge	Advanced knowledge in domain	Expert knowledge in domain	Expert knowledge in domain	Advanced knowledge in domain
Tool knowledge	Basic Knowledge of key tools	Advanced knowledge of key tools	Advanced knowledge of wide range of tools, derive solutions across platforms	Advanced knowledge of wide range of tools, derive solutions across platforms	Advanced knowledge of key tools
Platform Strategy	-	-	Design or support for Architect & Internal dev strategy	Design	-
Platform Design	-	Design complex platform for distinct use cases end to end	Design Complex platforms for wide range of use cases end to end	Design Complex platforms for wide range of use cases end to end	-
Pipeline Design	Develop individual pipelines (simple)	Develop Complex pipelines	Design complex pipelines, select tools	Design complex pipelines, select tools	Develop Complex pipelines
Data Modeling / Schema Design	Simple modeling & changes on existing data structures	Complex modeling for use case from ground up	Complex modeling of whole end to end pipelines	Complex modeling of whole end to end pipelines	Simple modeling & changes on existing data structures
Budget	-	Pipeline responsible	Team responsible	Planning for whole platform (big picture)	Pipeline responsible
Platform Security	Usage	Implementation	Implementation & Design	Design	Implementation
Data Protection	Usage	Implementation	Implementation & Design	Design	Implementation
APIs	Usage + implementation	Implementation & Design	Implementation & Design	Design	Implementation & Design
Processing Frameworks	Basic: SaaS, Containerization, Distributed frameworks	Advanced: SaaS, Containerization & Distributed frameworks	Expert: SaaS, Containerization & Distributed frameworks	Expert: SaaS, Containerization & Distributed frameworks	Advanced: SaaS, Containerization & Distributed frameworks
Data Stores	Usage and simple design of NoSQL stores	Use and choose the right available NoSQL store for distinct problems	Evaluate and choose data stores for complex problems in line with platform strategy	Evaluate and choose data stores for complex problems in line with platform strategy	Use and choose the right NoSQL store for distinct problems
Data Lake	Usage & simple implementation	Usage & complex Implementation	Design & implement governance and lineage	Design Governance and Lineage	Usage & complex Implementation

DATA

PLATFORM BLUEPRINT



Before you start doing projects you need to understand the typical parts of a data platform. This platform blueprint is very important as it helps you understand how a platform works. I split the blueprint into the following phases. To build a data pipeline you jut need to combine the phases.

Connect

The connection phase is the first phase where the data comes in. This could be a hosted API gateway where clients send data to. This data then flows through your platform.

You'll also find here very often data integration tools that access existing (external) systems such as data warehouses, relational SQL databases or external APIs. They extract the data and feed it into your platform and pipelines.

Buffer

In the middle of the blueprint you have the main processing and storage functions of your platform.

If the source in the connection is constantly pushing data in then you need some kind of buffer to process it dynamically. That's where message queues come in. They prevent what is called backpressure, where data is coming in faster than your processing can handle.

Buffers are also very good if you have multiple processings working with the same data. You feed the data into the queue once and multiple consumers can work with it in parallel.
Message queues help you to manage the flow of data within your platform without having to work with files.

Process

The processing framework is where things actually happen. This is where data is taken either from storage or from a message queue, processed and stored again.

It's almost the most important part of the whole platform, because without a processing framework, nothing happens. You need a way to transform and analyze data, and that's what it's there for.

Within the processing framework, there are 2 different types of processing: stream processing and batch processing. Batch processing is where you find extract transform and load (ETL) jobs. After processing data goes into a storage.

Store

The store phase is where you put the data in and store it. Here you find all kinds of data stores: Relational Databases, NoSQL databases, Data Warehouses, Data Lakes. You will find the different types again further below organized into OLTP and OLAP data stores.

The stored data is typically either used again by the processing framework, for a visualization tool or API that serves data to a client.

Visualize

After storing your data, you need to visualize it. That's where tools like web interfaces, Business intelligence tools or monitoring dashboards come in. Something where actually a user is working with and visualizes the data.

TOOLS GUIDE
A HELPFUL OVERVIEW

No that you know the phases of a data platform, what are the tools that you insert in every phase?

On the next page, you can see which tools are needed in each part of the platform blueprint and for which kind of platform they come to use.

I ordered the **columns** into cloud native and cloud agnostic (independent from a specific cloud). Then I split the two into Categories.

Cloud native I split into the main clouds: AWS, Azure and Google Cloud Platform (GCP). Cloud Agnostic I split into open source and vendors. This wasn't actually not that easy, because some vendors (like Databricks) are also cloud native, but I hope you get the point.

As **rows** I used the phases of the blueprint I showed above and added sub categories for the tools.

You are going to find some tools in multiple categories. That's completely normal. Some have multiple functionalities.

With this, you also have a great overview of helpful tool combinations on clouds, such as Lambda, Glue and ECS for the processing frameworks on AWS. The guide also allows you to find open source or vendor alternatives to these cloud native tools.

After this guide you find the two main types of pipelines people build. You can find concrete examples further below under "example projects and pipelines" how to arrange tools together to pipelines

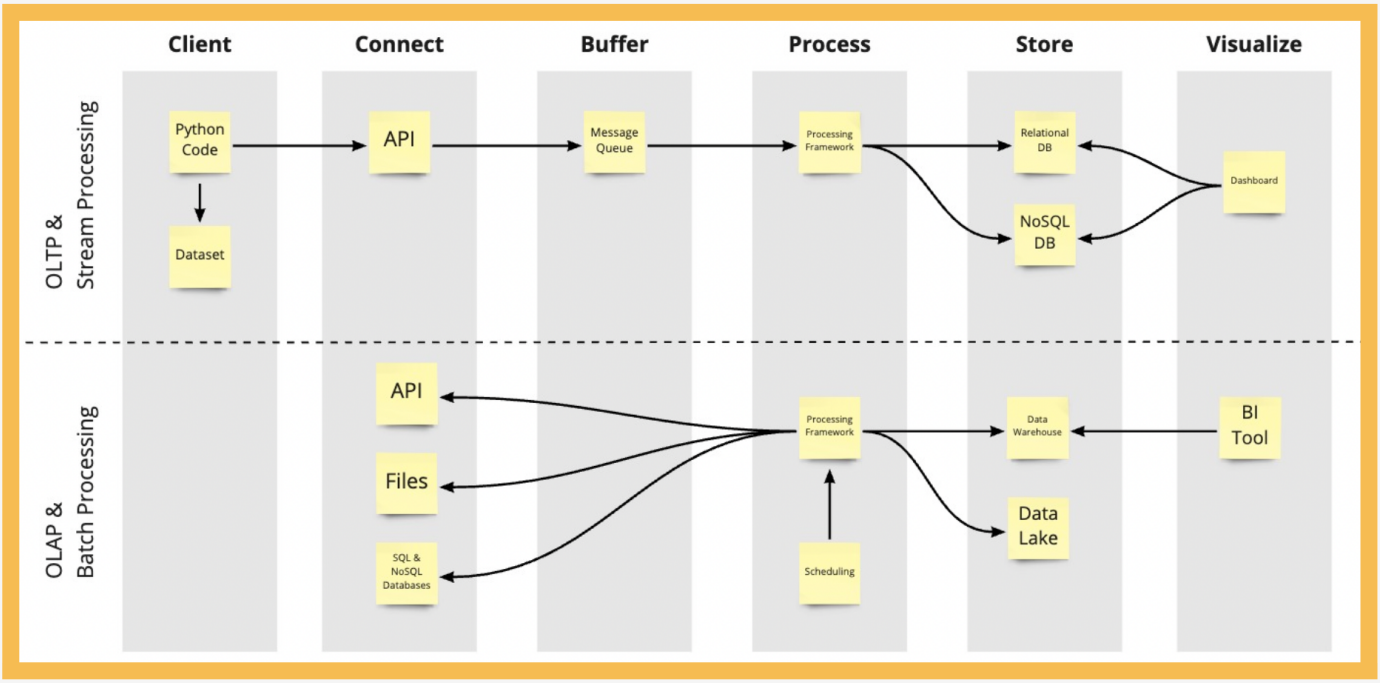
TOOLS GUIDE

by Andreas Kretz LearnDataEngineering.com		Cloud Native			Cloud Agnostic	
		AWS	**Azure**	**GCP**	**Open Source**	**Vendor**
Connect	**APIs**	API Gateway	API Management	API Gateway	FastAPI Flask Django GraphQL	
	Data Integration Tools	AWS Glue Stepfunctions	Data Factory	Cloud Data Fusion Dataprep	Airbyte Apache Nifi	Fivetran Streamsets Informatica Talend Hevo Data Streamsets Trifacta
Buffer	**Message Queues**	Kinesis Kinesis Firehose SQS	EventHub	Pub/Sub	Apache Kafka RabbitMQ Redis (Pub/Sub)	
Process	**Processing**	Lambda Stepfunctions Glue ECS EMR EKS	Functions Data Factory	Cloud Function DataFlow DataProc	Python Apache Spark Apache Flink Docker Kubernetes dbt Logstash	Databricks Alteryx Exasol Matillion
	Scheduling	EventBridge Cloudwatch	Azure Logic Apps Azure Batch	Cloud Scheduler Cloud Batch Cloud Workflows Cloud Composer	Airflow Luigi Dagster Perfect	Astronomer
	Machine Learning	SageMaker	Azure ML Synapse Analytics	Cloud Datalab ML, AutoML Vertex AI BigQueryML	Tensorflow PyTorch Keras	
Store	**OLTP + File Stores**	S3 RDS DynamoDB Timestream	CosmosDB Azure SQL DB Blob Storage	Cloud Storage CloudSpanner CloudSQL BigTable Firestore	MySQL MongoDB Postgres Elasticsearch Redis (K/V store) TimescaleDB	Microsoft SQL Server Oracle SQL DB
	OLAP	Redshift + Redshift Spectrum Athena	Azure Synapse Azure Data Explorer	BigQuery	Apache Druid InfluxDB Presto Apache Hive	Snowflake Databricks (Delta Lake) Oracle DW IBM Db2 SAP HANA Rockset
	Data Catalog	AWS Glue Data Catalog	Azure Data Catalog	Cloud Data Catalog	Amundsen Hive Metastore	Attacama
Visualize	**BI Tools**	Quicksight	(PowerBI)	Data Studio Looker	Streamlit Grafana Kibana Apache Superset Metabase	Tableau QlikSense PowerBI
Bonus	**Various Categories**	**Data Observability** Kensu.io Anomalo	**Data Quality** Great Expectations TIBCO SODA SQL	**CI/CD** Jenkins Gitlab GitHub Actions		

TYPICAL PIPELINES
OF DATA PLATFORMS

You most likely want to start building and arranging the tools from above. Before you do that you need to understand the difference between Online Transaction Processing (OLTP) and Online Analytical Processing (OLAP).

These are two different processes that many people mix up. They actually represent two different parts of a data platform: OLTP part for business processes & OLAP for analytics use cases.



Transactional Pipelines - The Transactional Part of a Platform

The OLTP part of a data platform is often used for realizing business processes (see upper part in the image above). That's why OLTP is more event driven and often uses streaming pipelines to process the incoming transactions (data) very quickly.

OLTP databases are ACID and CRUD compliant. They are the choice for efficiently storing transactions like store purchases.

You open a transaction, insert or modify your data and if everything is correct the transaction is confirmed and it's completed. If the transaction fails, the store rolls back your transaction and nothing has happened.

User interfaces use them very often to, for instance, visualizing your purchase history.

Another example is a factory where you produce goods. The production line queries from the database the details of what to produce next. After an item has been produced, the transactional database records that this item has been produced and some information about the production process.

Analytical Pipelines - The Analytics Part of a Platform

The analytics part of the platform is usually the part where data is coming in in larger chunks. Like once a day or once an hour. ELT jobs that run on a schedule are very often used to get the data into an OLAP store.

The data in OLAP is often less structured than in OLTP, which means you are more flexible. In OLAP stores, the data is usually simply copied in via copy commands and not via transactions.

Analysts have all the data at their disposal and are able to look at it from a completely different perspective - from a holistic perspective. They gain insights into larger amounts of data.

OLAP stores help analysts to answer questions like: Identify the customers with the highest value purchases in a month or a year. Another question could be to find the best-selling products in the shop's inventory by month or quarter.

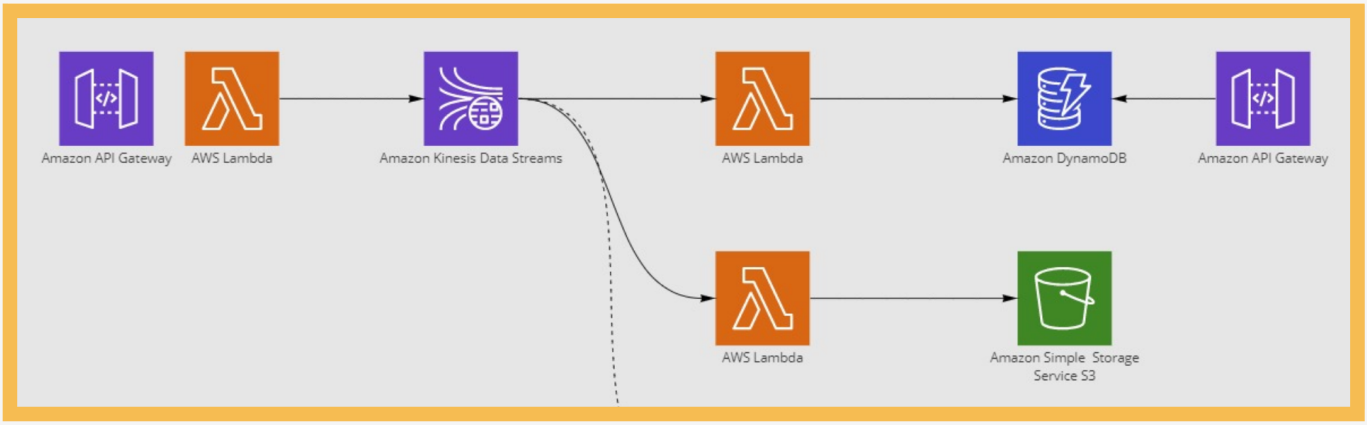
EXAMPLE PROJECTS
& PIPELINES

Now that you have learned everything about the most important parts of building a platform, the pipelines, OLTP and OLAP, as well as all the tools you can work with, you can start with your very own data engineering project.

Here is an overview of some example platform and pipeline use cases which we see all the time in the real world and which I teach to my students via hands-on project courses:

Transactional Pipelines

Data Engineering on AWS



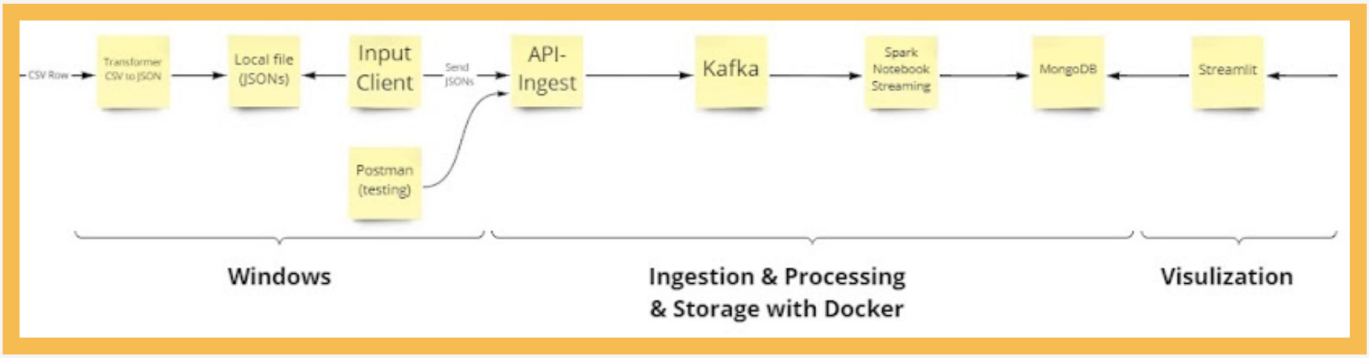
Connect	Buffer	Process	Store	Visualize
API Gateway	Kinesis Data Stream	Lambda	S3 DynamoDB	API Gateway

The AWS project is perfect for everyone who wants to start with Cloud platforms. Currently, AWS is the most used platform for data processing. It is really great to use, especially for those people who are new in their Data Engineering job or looking for one.

Working through AWS, you learn how to set up a complete end-to-end project with a streaming and analytics pipeline. Based on the project, you learn how to model data and which AWS tools are important, such as Lambda, API Gateway, Glue, Redshift Kinesis and DynamoDB.

Link to the project in our Data Engineering Academy:
https://learndataengineering.com/p/data-engineering-on-aws

Document Streaming with Kafka, Spark and MongoDB



Connect	Buffer	Process	Store	Visualize
FastAPI	Apache Kafka	Apache Spark	MongoDB	Streamlit

This full end-to-end example project uses e-commerce data that contains invoices for customers and items on these invoices.

Here, your goal is to ingest the invoices one by one, as they are created, and visualize them in a user interface. The technologies you will use are FastAPI, Apache Kafka, Apache Spark, MongoDB and Streamlit - tools you can learn in the academy individually, which I recommend before getting into this project.

Among other things, you learn about each step of how to build your pipeline and which tools to use at which point and get to know how to work with Apache Spark structured streaming in connection with MongoDB.

Also, you set up your own data visualization and create an interactive user interface with Streamlit to select and view invoices for customers and the items on these invoices.

Sounds like the right project for you? Here you can find it in my academy: https://learndataengineering.com/p/document-streaming

Analytical Pipelines

AWS ETL Pipeline to Redshift

Connect	Buffer	Process	Store	Visualize
---	---	AWS Glue	S3 Redshift	PowerBI

This project is a ETL pipeline where we take files out of S3 and put the contents into Redshift. For visualization we use PowerBI. (This is part of the Data Engineering on AWS project)

As you are not using an API here, the data is stored as a file within S3 as clients usually send in their data as files. This is very typical for an ETL job, where the data is extracted from S3 and then transformed and stored.

A typical tool to use here is AWS Glue. With this tool, you can write Apache Spark jobs, which pull the data from S3 and then transform and store them into the destination, for which we recommend Redshift. Beside the Glue jobs, another part you should make use of is the data catalog. It catalogs the csv files within S3 as well as the data in Redshift. Thus, it becomes very easy to automatically configure and generate a Spark job in AWS Glue, which sends the data to Redshift.

As said before, the recommended storage for the analytical part is Redshift, from which the data analysts can evaluate and visualize the data. Best to use here is a single staging table in which the data is stored.

The typical method here would be to use Redshift as your analytics database, to which you ideally connect a BI tool such as Power BI, for example.

Modern Data Warehouses

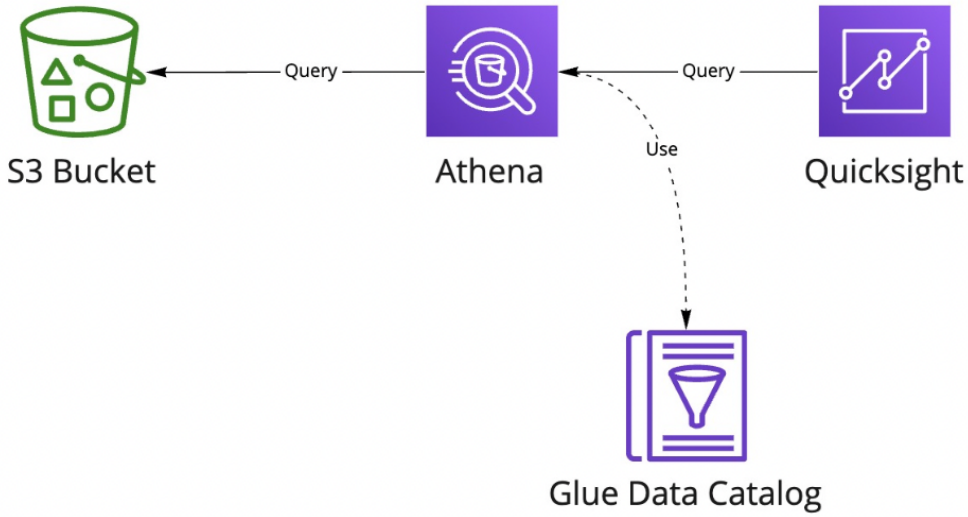
Google Cloud

Connect	Buffer	Process	Store	Visualize
---	---	---	Cloud Storage BigQuery	Data Studio

On GCP you can start by configuring Cloud Storage, BigQuery and Data Studio on the Google Cloud Platform (GCP). Put a file into the lake, create a BigQuery table and a Quicksight report that you can share with anyone you want.

Modern Data Warehouses

AWS



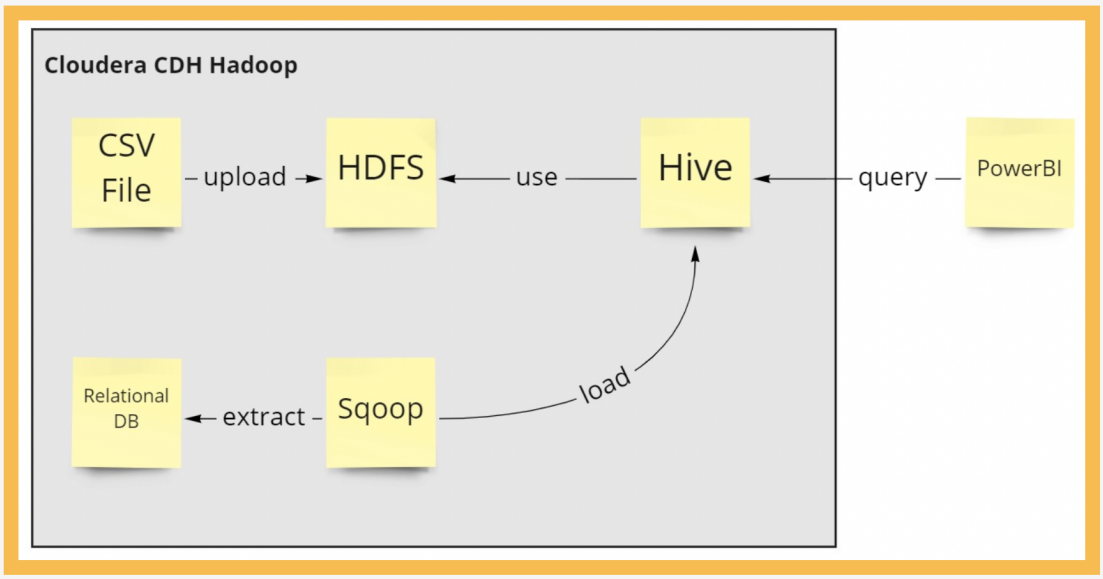
Connect	Buffer	Process	Store	Visualize
---	---	---	S3 Athena Glue Data Catalog	Quicksight

Or go into AWS to set up a manual data lake integration through S3, Athena and Quicksight. After that you can change the integration to using the Glue Data catalog.

As a BONUS LESSON I share in my course how you can do what you did with AWS Athena through Redshift Spectrum.

Here you can find the complete course in my Data Engineering Academy: https://learndataengineering.com/p/modern-data-warehouses

Hadoop Data Warehousing with Hive



Connect	Buffer	Process	Store	Visualize
Sqoop Relational DB	---	---	HDFS Hive	PowerBI

Hadoop is a Java-based and open source framework that handles all sorts of storage and processing for Big Data across clusters of computers using simple programming models.

Working through this project you will learn and master the Hadoop architecture and its components like HDFS, YARN, MapReduce, Hive and Sqoop. Understand the detailed concepts of the Hadoop Eco-system along with hands-on labs as well as the most important Hadoop commands. Also, learn how to implement and use each component to solve real business problems!

Learn how to store and query your data with Sqoop, Hive, and MySQL, write Hive queries and Hadoop commands, manage Big Data on a cluster with HDFS and MapReduce and manage your cluster with YARN and Hue.

If you think that's the perfect project for you, get right into it: https://learndataengineering.com/p/data-engineering-with-hadoop

DATA ENGINEERING ACADEMY

The best place to learn Data Engineering!

Perfect for becoming a Data Engineer or add Data Engineering to your skillset

- Huge step by step Data Engineering course
- Trusted by over 900 students
- 12 months or unlimited access to all courses incl. future ones during your subscription
- 5 theory courses on the most important engineering techniques
- 10 hands-on courses the most important tools for engineers
- 8 hands-on example data engineering projects on major platforms like AWS, Azure, open source platforms and Hadoop
- Over 44 hours of video content
- Prepared source codes
- Private Discord community
- Associate Data Engineer Certification and course certificates

Contact us for more information about the Academy or advertising opportunities:
hello@learndataengineering.com