EP64: How to improve API performance

ALEX XU
JUN 17, 2023

252 9 11 Share ...

This week's system design refresher:

- How Discord Stores TRILLIONS of Messages (YouTube video)
- Netflix's Overall Architecture
- How to improve API performance
- Branching strategies
- Key data terms
- ByteByteGo is looking for guest posts

2023 State of the Java Ecosystem Report by New Relic (Sponsored)

new relic
2023 State of the Java Ecosystem
An in-depth look at one of the most popular programming languages

Get an in-depth look at one of the most popular programming languages in New Relic's 2023 State of the Java Ecosystem report.

You'll get insight into:

- The most used Java versions in production
- The most popular JDK vendors
- The rise of containers
- The most common heap size configurations
- The most used garbage collection algorithms

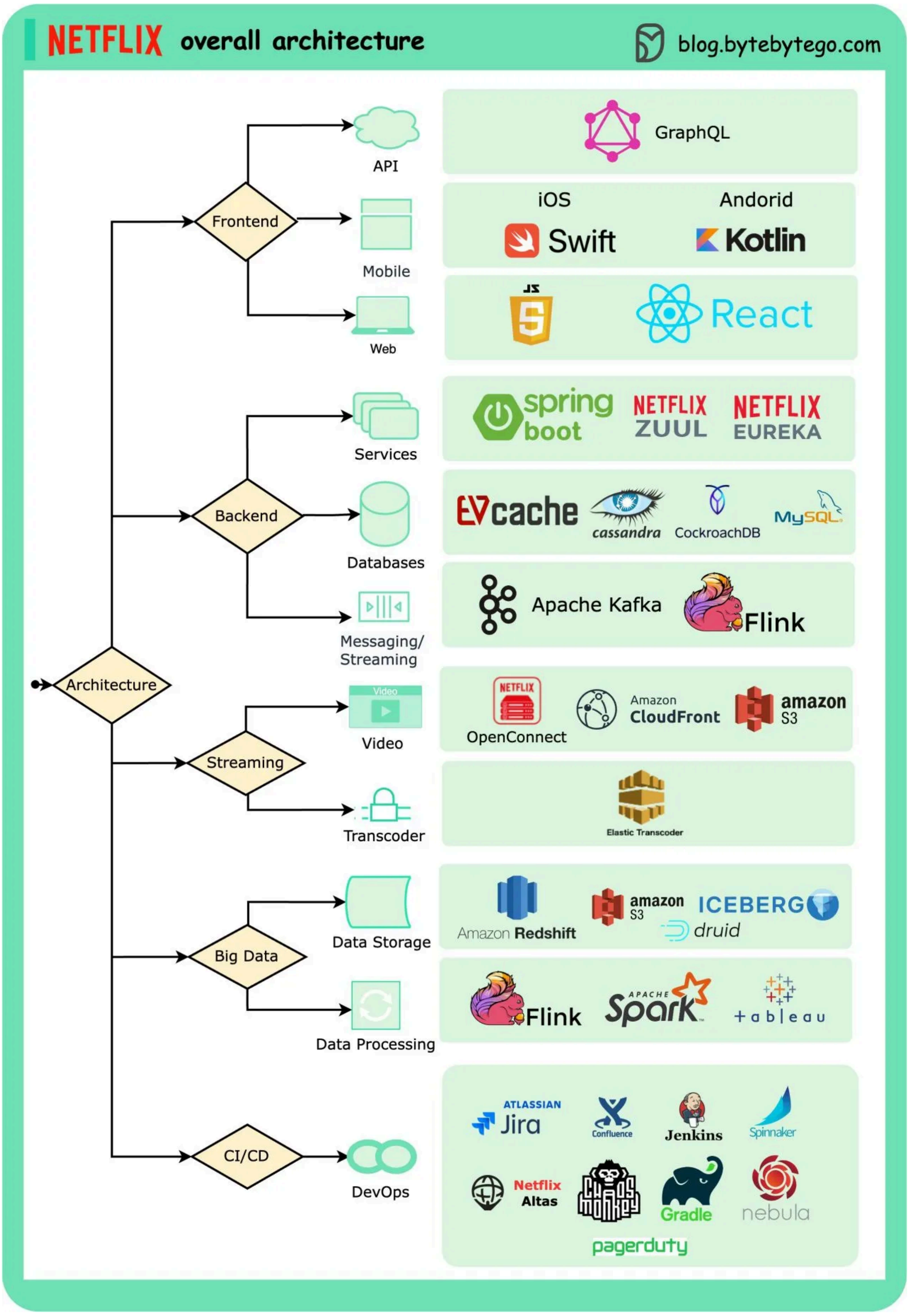
Highlights from the report:

- Java 17 user adoption grew 430% in one year
- Java 14 is the most popular non-LTS version
- Amazon is now the most popular JDK vendor
- Containers rule everything around us

How Discord Stores TRILLIONS of Messages

Netflix's Overall Architecture

This post is based on research from many Netflix engineering blogs and open-source projects. If you come across any inaccuracies, please feel free to inform us.


NETFLIX overall architecture
blog.bytebytego.com
Architecture
Frontend
API
GraphQL
Mobile
iOS
Swift
Andorid
Kotlin
Web
JS
React
Backend
Services
spring boot
NETFLIX ZUUL
NETFLIX EUREKA
Databases
EVcache
cassandra
CockroachDB
MySQL
Messaging/ Streaming
Apache Kafka
Flink
Streaming
Video
NETFLIX OpenConnect
Amazon CloudFront
amazon S3
Transcoder
Elastic Transcoder
Big Data
Data Storage
Amazon Redshift
amazon S3
ICEBERG
druid
Data Processing
Flink
Apache Spark
tableau
CI/CD
DevOps
ATLASSIAN Jira
Confluence
Jenkins
Spinnaker
Netflix Altas
Chaos Monkey
Gradle
nebula
pagerduty

Mobile and web: Netflix has adopted Swift and Kotlin to build native mobile apps. For its web application, it uses React.

Frontend/server communication: Netflix uses GraphQL.

Backend services: Netflix relies on ZUUL, Eureka, the Spring Boot framework, and other technologies.

Databases: Netflix utilizes EV cache, Cassandra, CockroachDB, and other databases.

Messaging/streaming: Netflix employs Apache Kafka and Fink for messaging and streaming purposes.

Video storage: Netflix uses S3 and Open Connect for video storage.

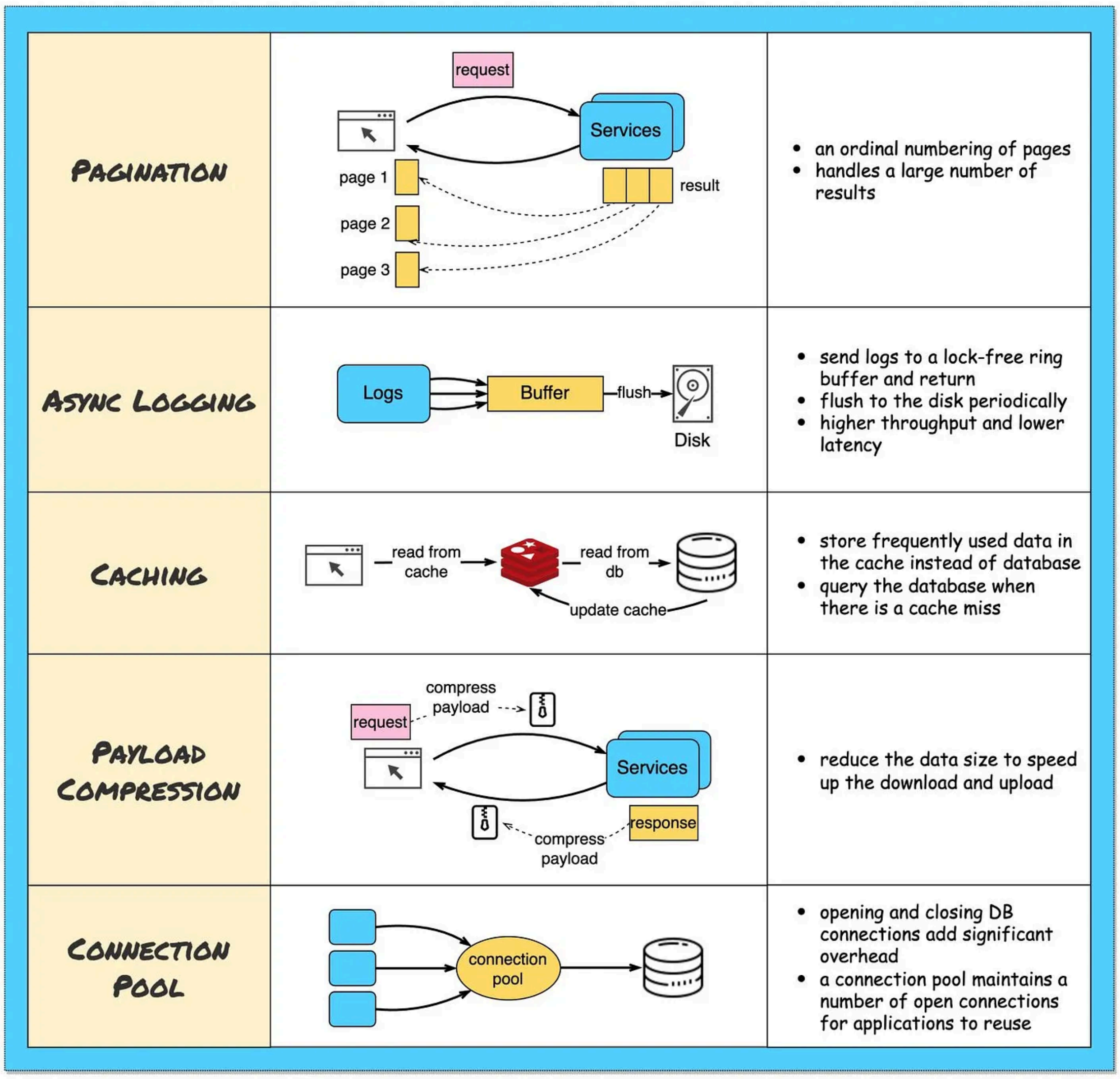
Data processing: Netflix utilizes Flink and Spark for data processing, which is then visualized using Tableau. Redshift is used for processing structured data warehouse information.

CI/CD: Netflix employs various tools such as JIRA, Confluence, PagerDuty, Jenkins, Gradle, Chaos Monkey, Spinnaker, Altas, and more for CI/CD processes.

How to improve API performance

The diagram below shows 5 common tricks to improve API performance.

How to Improve API Performance?



PAGINATION	• an ordinal numbering of pages • handles a large number of results
ASYNC LOGGING	• send logs to a lock-free ring buffer and return • flush to the disk periodically • higher throughput and lower latency
CACHING	• store frequently used data in the cache instead of database • query the database when there is a cache miss
PAYLOAD COMPRESSION	• reduce the data size to speed up the download and upload
CONNECTION POOL	• opening and closing DB connections add significant overhead • a connection pool maintains a number of open connections for applications to reuse

Reference: Rapid API

1. Pagination
   This is a common optimization when the size of the result is large. The results are streaming back to the client to improve the service responsiveness.

2. Asynchronous Logging
   Synchronous logging deals with the disk for every call and can slow down the system. Asynchronous logging sends logs to a lock-free buffer first and immediately returns. The logs will be flushed to the disk periodically. This significantly reduces the I/O overhead.

3. Caching
   We can cache frequently accessed data into a cache. The client can query the

cache first instead of visiting the database directly. If there is a cache miss, the client can query from the database. Caches like Redis store data in memory, so the data access is much faster than the database.

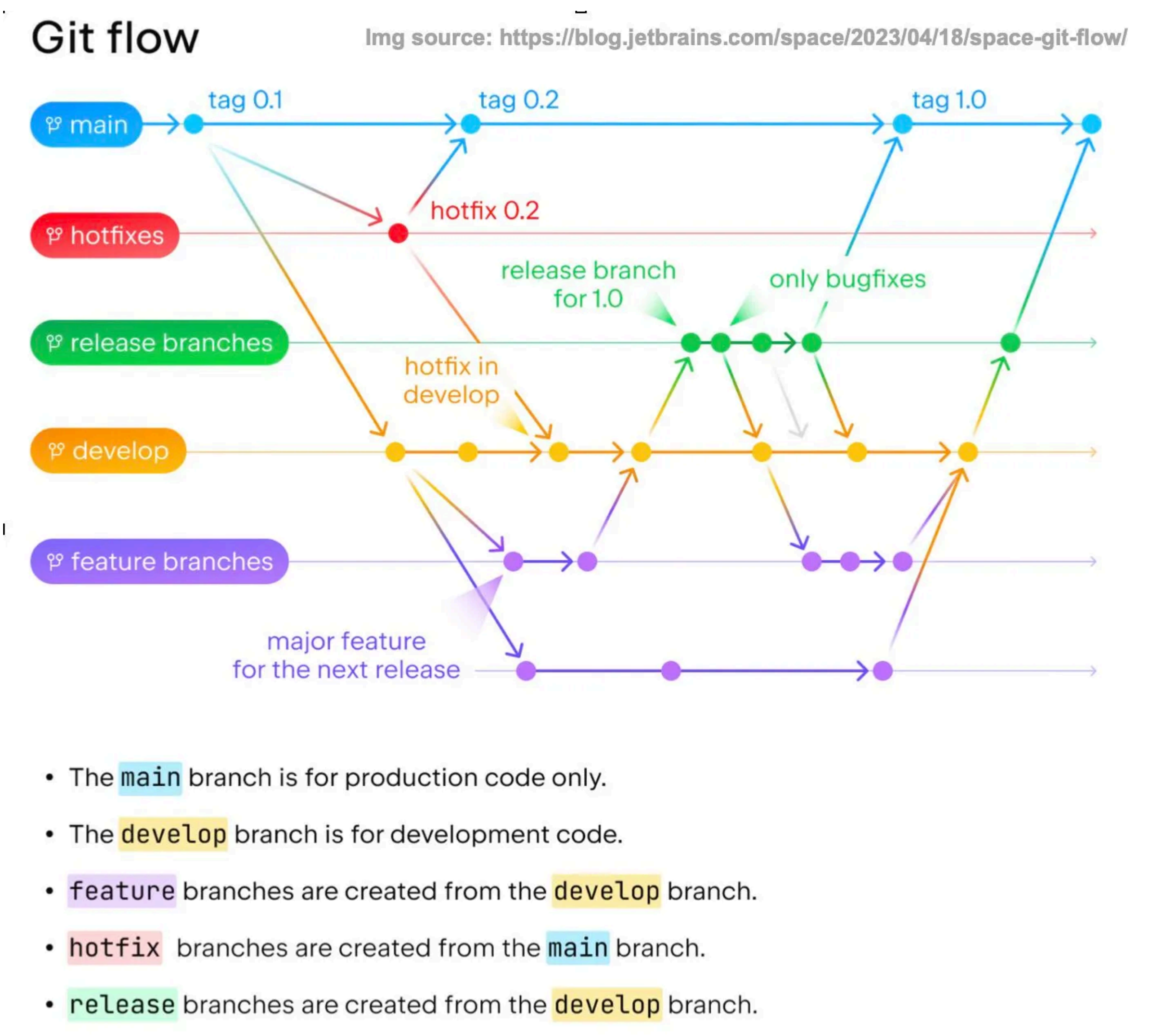
4. Payload Compression
The requests and responses can be compressed using gzip etc so that the transmitted data size is much smaller. This speeds up the upload and download.

5. Connection Pool
When accessing resources, we often need to load data from the database. Opening the closing db connections add significant overhead. So we should connect to the db via a pool of open connections. The connection pool is responsible for managing the connection lifecycle.

Over to you: What other tricks do you use to improve API performance?

What branching strategies does your team use?

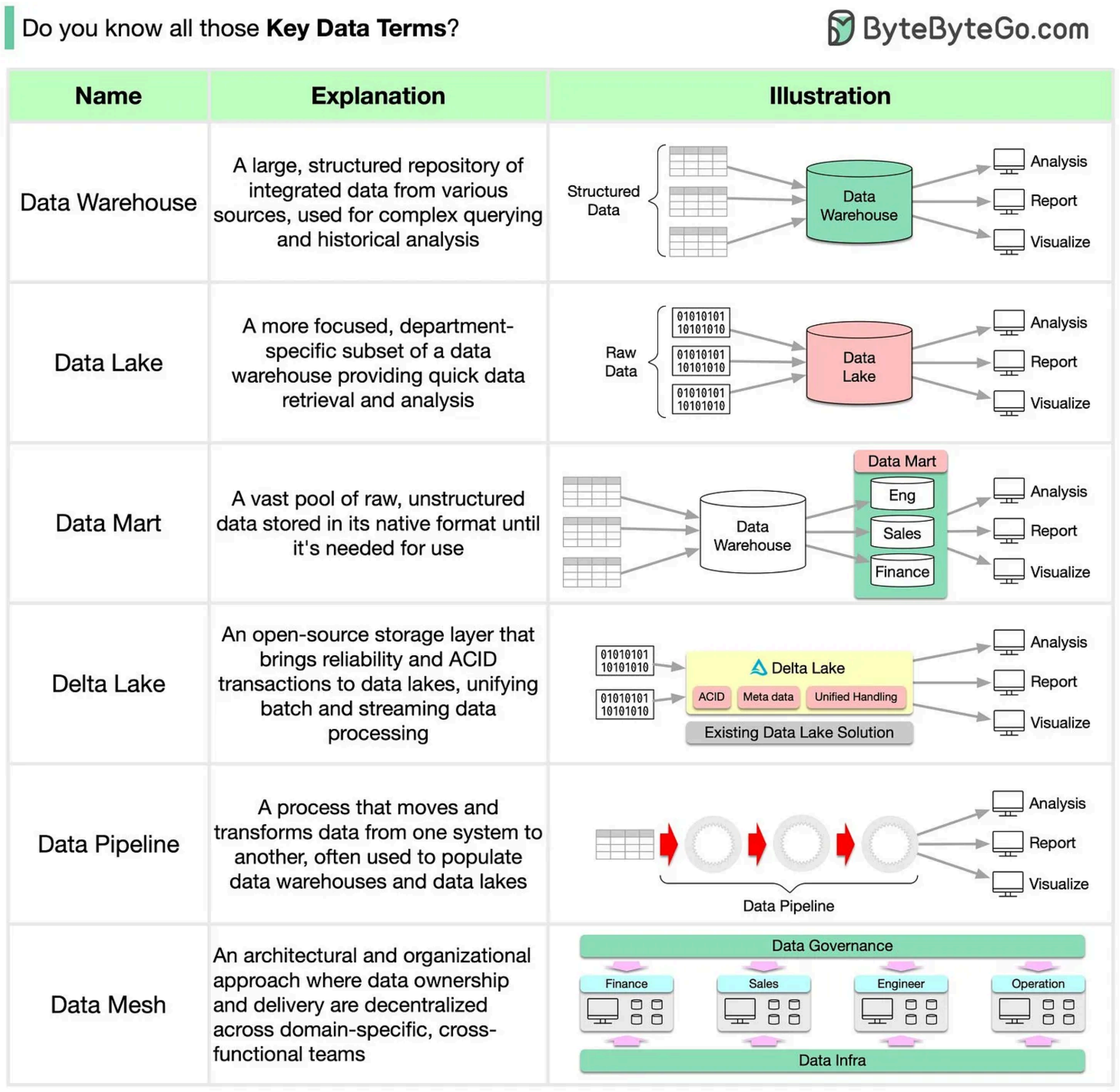
Teams often employ various branching strategies for managing their code, such as Git flow, feature branches, and trunk-based development.



- The `main` branch is for production code only.
- The `develop` branch is for development code.
- `feature` branches are created from the `develop` branch.
- `hotfix` branches are created from the `main` branch.
- `release` branches are created from the `develop` branch.

Out of these options, Git flow or its variations are the most widely favored methods. The illustration by Jetbrains explains how it works.

Data is used everywhere, but do you know all the commonly used data terms?



Do you know all those **Key Data Terms**? ByteByteGo.com

Name	Explanation	Illustration
Data Warehouse	A large, structured repository of integrated data from various sources, used for complex querying and historical analysis	Structured Data → Data Warehouse → Analysis, Report, Visualize
Data Lake	A more focused, department-specific subset of a data warehouse providing quick data retrieval and analysis	Raw Data → Data Lake → Analysis, Report, Visualize
Data Mart	A vast pool of raw, unstructured data stored in its native format until it's needed for use	Data Warehouse → Data Mart (Eng, Sales, Finance) → Analysis, Report, Visualize
Delta Lake	An open-source storage layer that brings reliability and ACID transactions to data lakes, unifying batch and streaming data processing	Delta Lake (ACID, Meta data, Unified Handling), Existing Data Lake Solution → Analysis, Report, Visualize
Data Pipeline	A process that moves and transforms data from one system to another, often used to populate data warehouses and data lakes	Data Pipeline → Analysis, Report, Visualize
Data Mesh	An architectural and organizational approach where data ownership and delivery are decentralized across domain-specific, cross-functional teams	Data Governance; Finance, Sales, Engineer, Operation; Data Infra

- Data Warehouse: A large, structured repository of integrated data from various sources, used for complex querying and historical analysis. \
- Data Mart: A more focused, department-specific subset of a data warehouse providing quick data retrieval and analysis.
- Data Lake: A vast pool of raw, unstructured data stored in its native format until it's needed for use.
- Delta Lake: An open-source storage layer that brings reliability and ACID transactions to data lakes, unifying batch, and streaming data processing.
- Data Pipeline: A process that moves and transforms data from one system to another, often used to populate data warehouses and data lakes.

- Data Mesh: An architectural and organizational approach where data ownership and delivery are decentralized across domain-specific, cross-functional teams.

Over to you: do you know the difference between a data engineer and a data scientist?

ByteByteGo is looking for guest posts/authors

I'm looking for guest posts/authors for ByteByteGo's social media or newsletter (> 1.5 million audiences).

ByteByteGo's social media channels reach an extensive audience of over 1.5 million individuals worldwide.

Qualifications:

- The ideal person will have read and heard enough of our content and can explain complex technical topics with simple illustrations.
- Proficient research skills in a subject you are knowledgeable about.
- Ability to cover topics that appeal to a wide audience. This is important.

Submit your pitch here.

9 Comments

Big Tech Digest Big Tech Digest Jun 18, 2023 · *edited Jun 27, 2023*

Other ways of improving API performance:

- Avoiding the thread-per-request model and using an event loop instead. Using non-blocking APIs whenever possible on the backend.

- Offer filtering and field projection capabilities in your API. This way, your clients will only fetch the data they're interested in, avoiding overfetching.

- For larger payloads, using HTTP chunked encoding or gRPC streaming to transfer data piece by piece.

- Be careful with pagination with SQL databases as limit and offset operations are costly. See: https://stackoverflow.com/questions/4481388/why-does-mysql-higher-limit-offset-slow-the-query-down

- Limit the number of requests. Use batching instead.

- Make sure HTTP/2 is enabled.

- JSON parsing is costly. Switch to binary formats like Protobuf.

- Use PATCH instead of PUT to update just a subset of the resource.

PS. Subscribe to my free newsletter that delivers a collection of links to the latest engineering blog posts from Big Tech companies and startups twice a month: https://bigtechdigest.substack.com/

LIKE (7) REPLY SHARE

3 replies

Venkata Jun 17, 2023

I think the definitions provided in data key terms for data lake and data mart are different in picture and text description

LIKE (4) REPLY SHARE

2 replies

7 more comments...