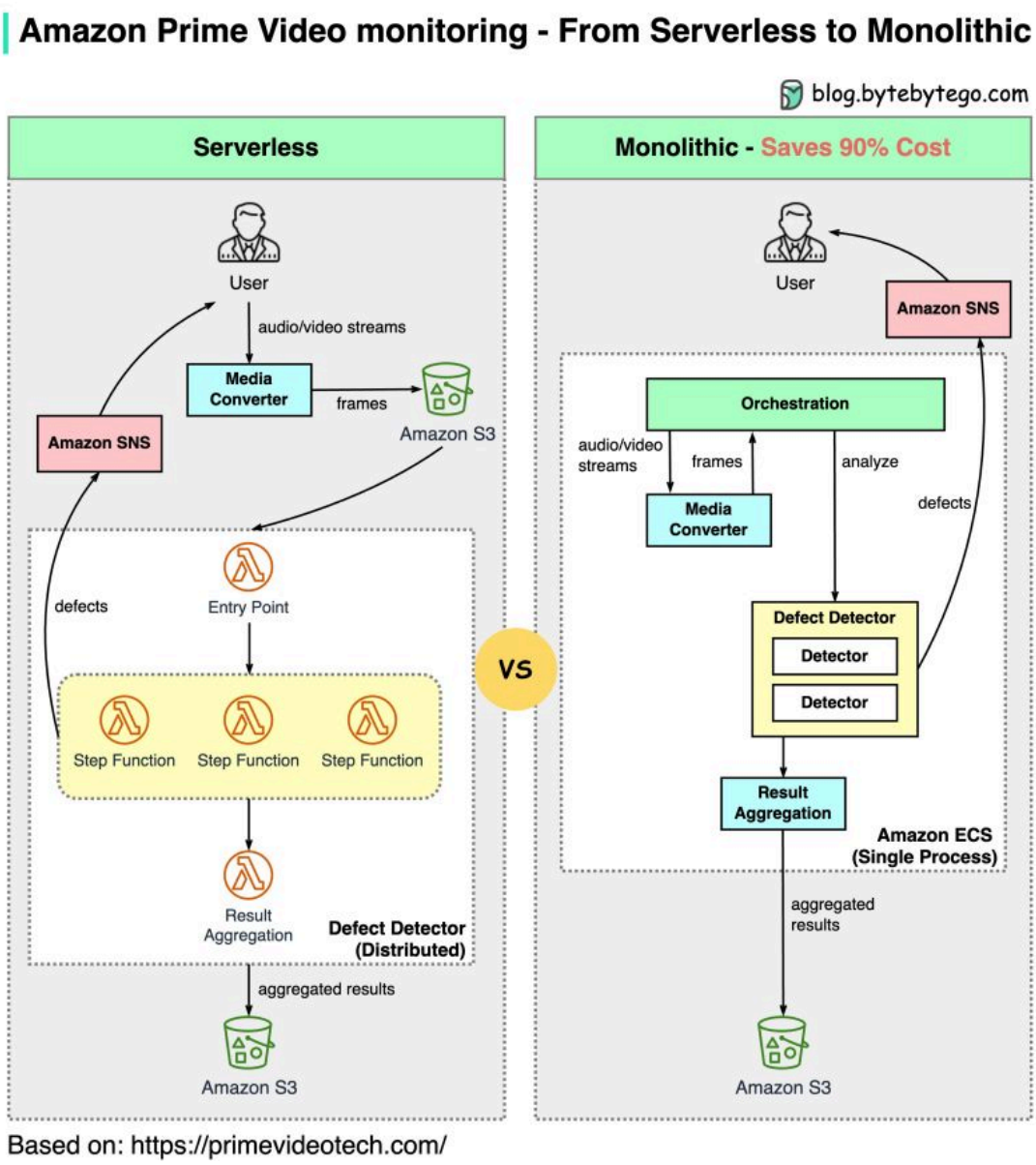
- 1980s
  Since the 1980s, machine learning started to pick up and was used for classification. The training was conducted on a small range of data.

- 1990s - 2000s
  Since the 1990s, neural networks started to imitate human brains for labeling and training. There are generally 3 types:
  - CNN (Convolutional Neural Network): often used in visual-related tasks.
  - RNN (Recurrent Neural Network): useful in natural language tasks
  - GAN (Generative Adversarial Network): comprised of two networks(Generative and Discriminative). This is a generative model that can generate novel images that look alike.

- 2017
  “Attention is all you need” represents the foundation of generative AI. The transformer model greatly shortens the training time by parallelism.

- 2018 - Now
  In this stage, due to the major progress of the transformer model, we see various models train on a massive amount of data. Human demonstration becomes the learning content of the model. We’ve seen many AI writers that can write articles, news, technical docs, and even code. This has great commercial value as well and sets off a global whirlwind.

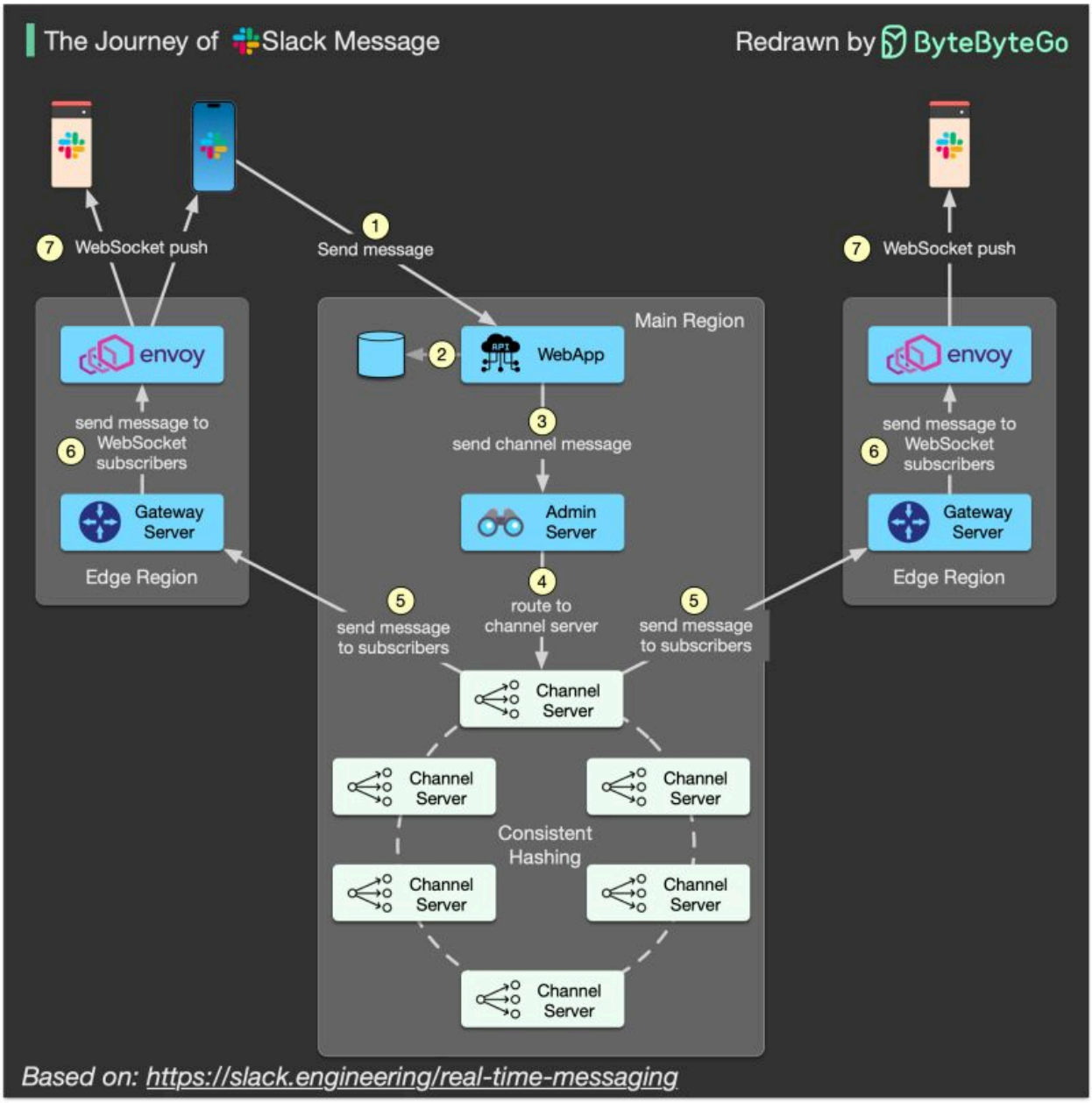
Why did Amazon Prime Video monitoring move from serverless to monolithic? How can it save 90% cost?



In this video, we will talk about:
- What is Amazon Prime Video Monitoring Service
- What is the problem with the old serverless architecture
- How the monolithic architecture saves 90% cost
- What did Amazon leaders say about this

Watch and subscribe here: https://lnkd.in/eFaVeRij

What is the journey of a Slack message?



In a recent technical article, Slack explains how its real-time messaging framework works. Here is my short summary:

- Because there are too many channels, the Channel Server (CS) uses consistent hashing to allocate millions of channels to many channel servers.
- Slack messages are delivered through WebApp and Admin Server to the correct Channel Server.

- Through Gate Server and Envoy (a proxy), the Channel Server will push messages to message receivers.
- Message receivers use WebSocket, which is a bi-directional messaging mechanism, so they are able to receive updates in real-time.

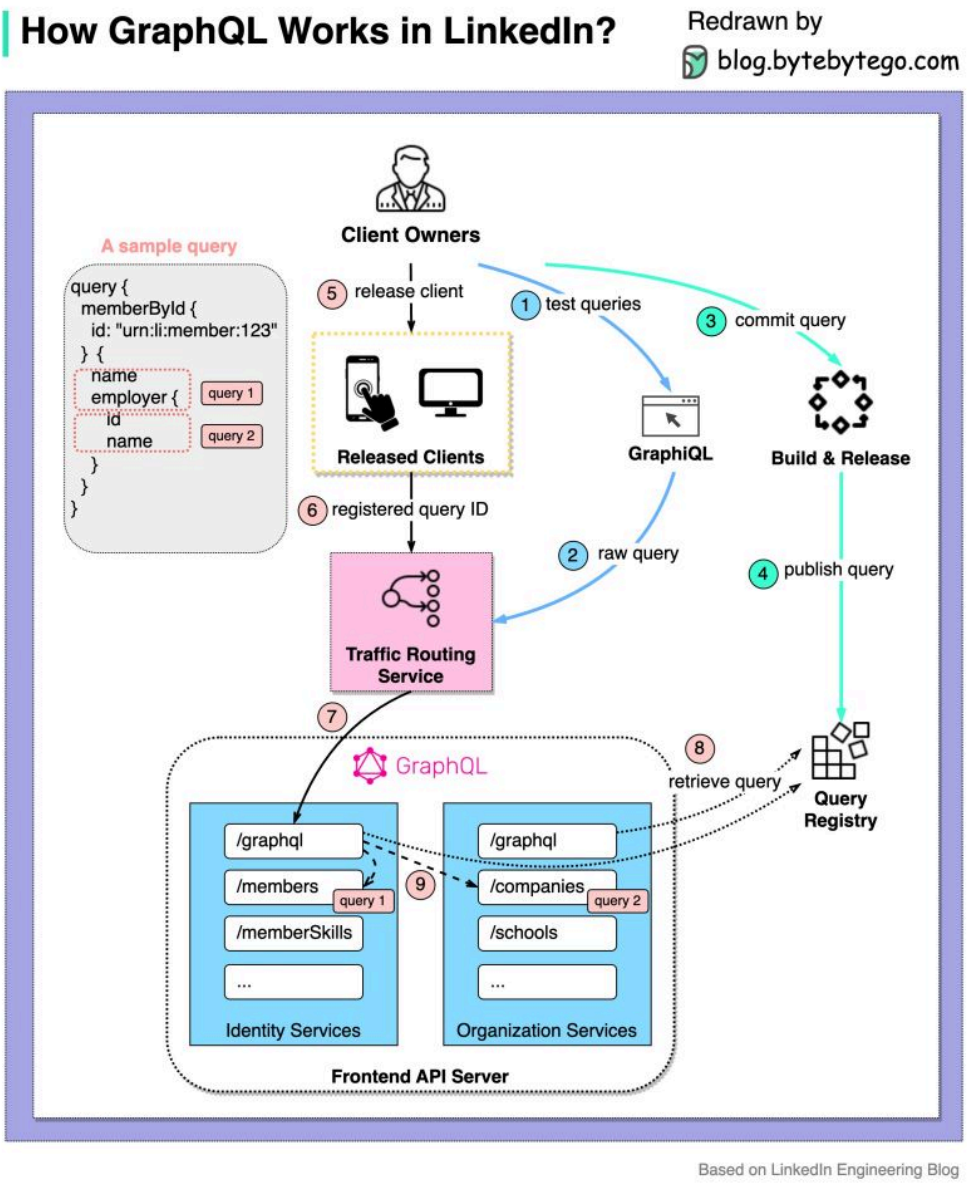
A Slack message travels through five important servers:

- WebApp: define the API that a Slack client could use
- Admin Server (AS): find the correct Channel Server using channel ID
- Channel Server (CS): maintain the history of message channel
- Gateway Server (GS): deployed in each geographic region. Maintain WebSocket channel subscription
- Envoy: service proxy for cloud-native applications

Over to you: The Channel Servers could go down. Since they use consistent hashing, how might they recover?

How does GraphQL work in the real world?

The diagram below shows how LinkedIn adopts GraphQL.



“Moving to GraphQL was a huge initiative that changed the development workflow for thousands of engineers...” [1]

The overall workflow after adopting GraphQL has 3 parts:

- Part 1 - Edit and Test a Query
  Steps 1-2: The client-side developer develops a query and tests with backend services.

- Part 2 - Register a Query
  Steps 3-4: The client-side developer commits the query and publishes the query to the query registry.

- Part 3 - Use in Production
  Step 5: The query is released together with the client code.
  Steps 6-7: The routing metadata is included with each registered query. The metadata is used at the traffic routing tier to route the incoming requests to the correct service cluster.
  Step 8: The registered queries are cached at service runtime.
  Step 9: The sample query goes to the identity service first to retrieve members and then goes to the organization service to retrieve company information.

LinkedIn doesn't deploy a GraphQL gateway for two reasons:

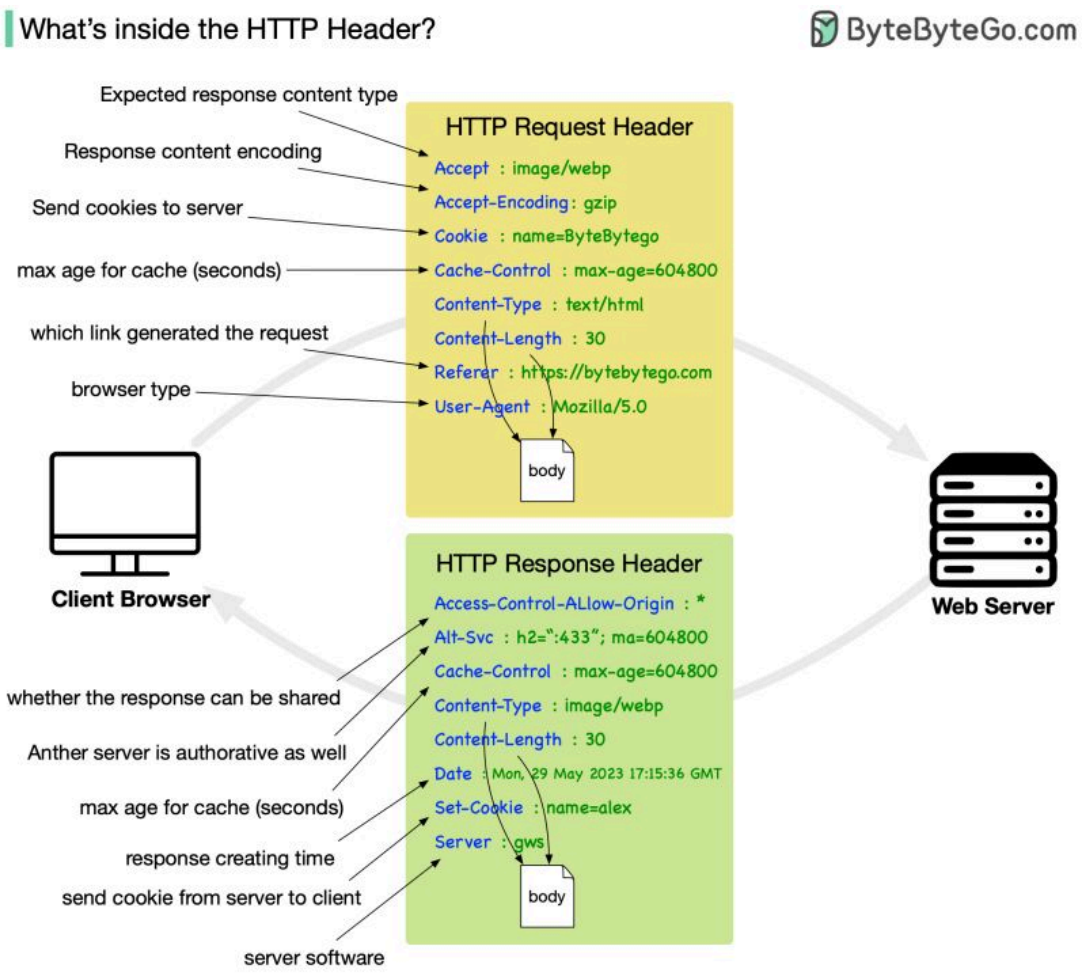
1. Prevent an additional network hop
2. Avoid single point of failure

👉Over to you: How are GraphQL queries managed in your project?

Reference: How LinkedIn Adopted A GraphQL Architecture for Product Development

Important Things About HTTP Headers You May Not Know!

HTTP requests are like asking for something from a server, and HTTP responses are the server's replies. It's like sending a message and receiving a reply.

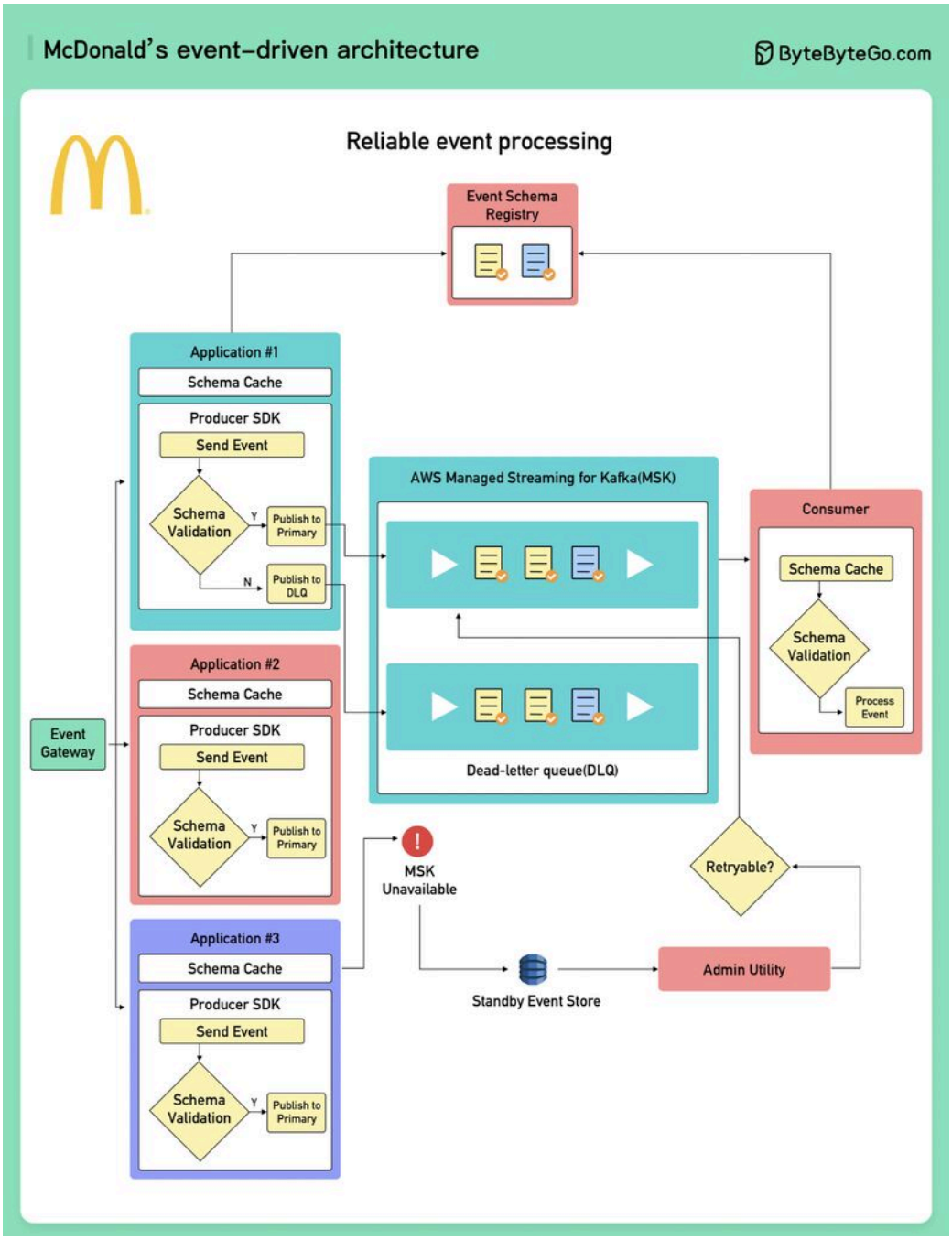


An HTTP request header is an extra piece of information you include when making a request, such as what kind of data you are sending or who you are. In response headers, the server provides information about the response it is sending you, such as what type of data you're receiving or if you have special instructions.

A header serves a vital role in enabling client-server communication when building RESTful applications. In order to send the right information with their requests and interpret the server's responses correctly, you need to understand these headers.

👉Over to you: the header “referer” is a typo. Do you know what the correct name is?

Think you know everything about McDonald's? What about its event-driven architecture 😬?



McDonald's standardizes events using the following components:

- An event registry to define a standardized schema.
- Custom software development kits (SDKs) to process events and handle errors.

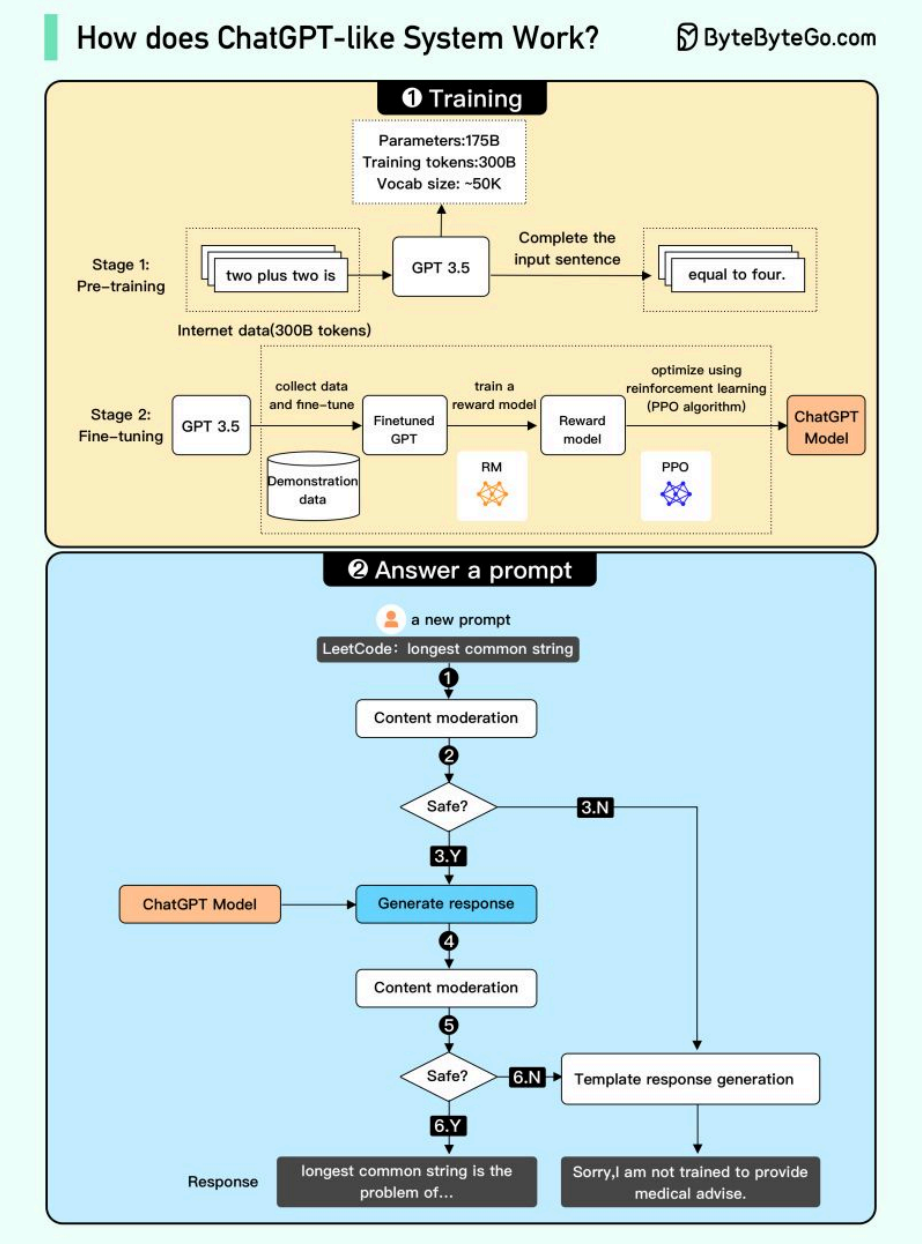
- An event gateway that performs identity authentication and authorization.
- Utilities and tools to fix events, keep the cluster healthy, and perform administrative tasks.

To scale event processing, McDonald uses a regional architecture that provides global availability based on AWS. Within a region, producers shard events by domains, and each domain is processed by an MSK cluster. The cluster auto-scales based on MSK metrics (e.g., CPU usage), and the auto-scale workflow is based on step-functions and re-assignment tasks.

How ChatGPT works technically

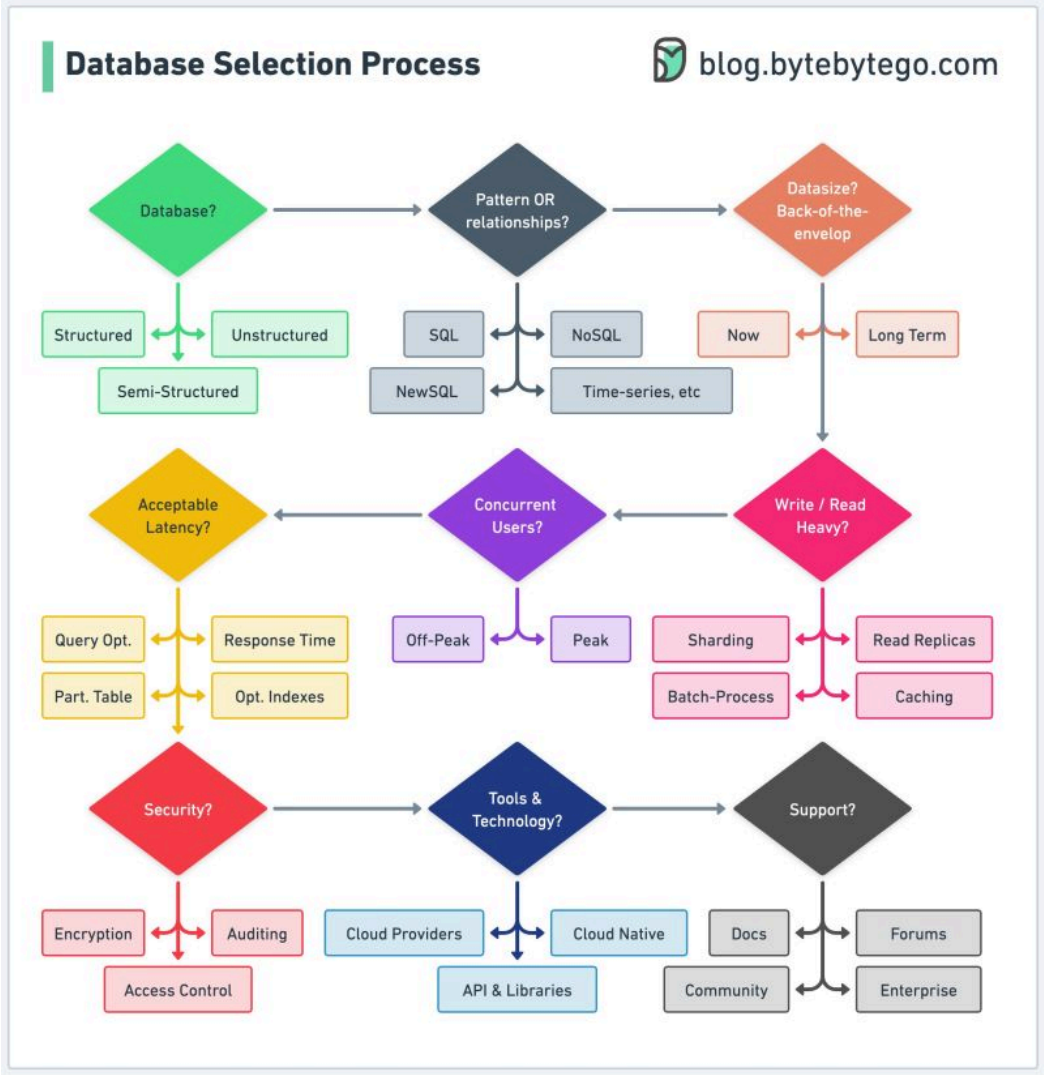
We attempted to explain how it works in this video. We will cover:

- Large Language Model
- GPT-3.5
- Fine-tuning
- Prompt engineering
- How to answer a prompt



Watch and subscribe here (YouTube video): https://lnkd.in/eNAUnWup

Choosing the right database is probably the most important technical decision a company will make.



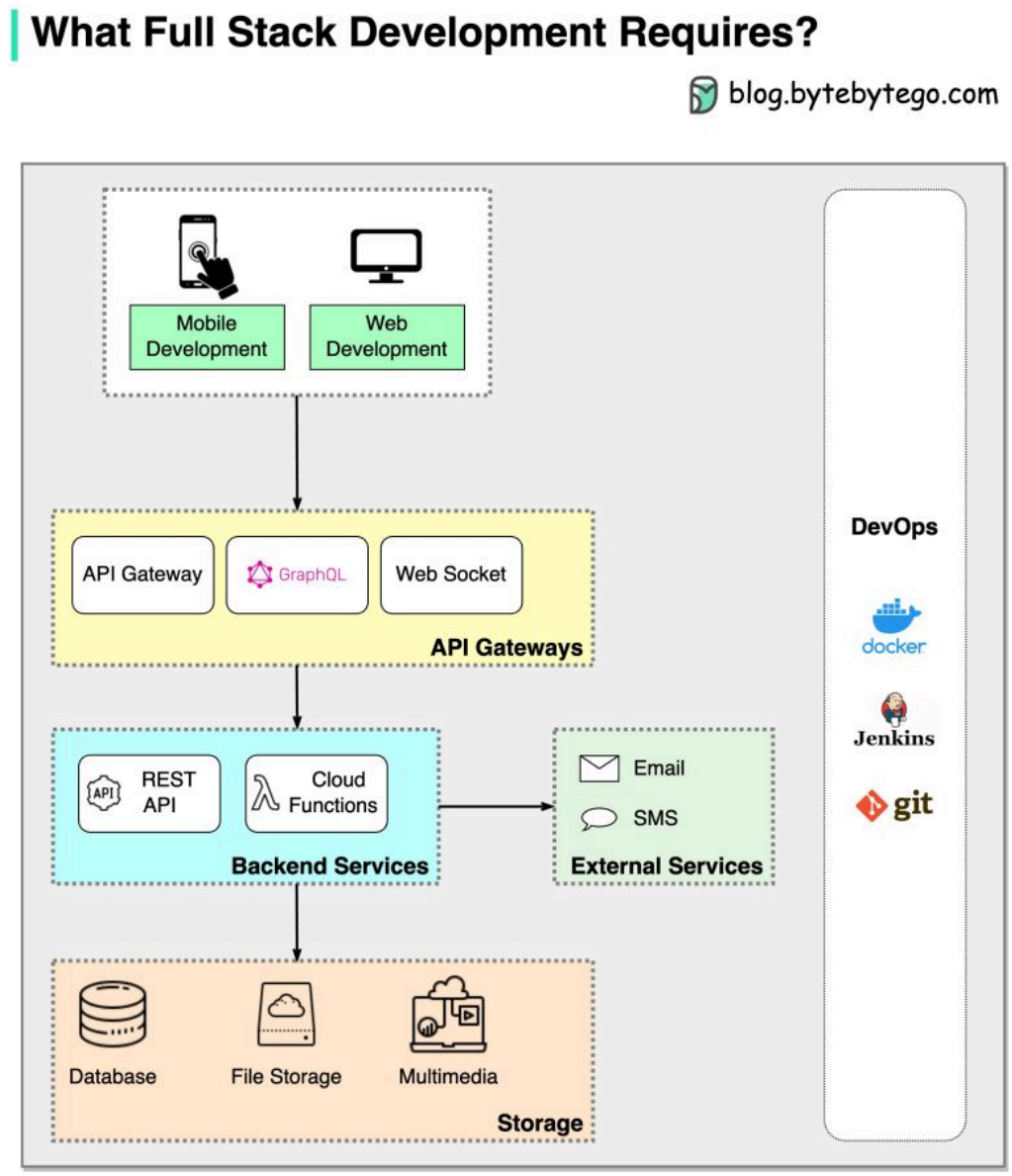
In this three-part newsletter series, we will dive deep into:

- Examining the types of data our project will handle.
- Considering the expected volume of data the project will generate.
- Evaluating the anticipated number of concurrent users or connections.
- Carefully assessing performance and security requirements.
- Considering any existing systems, tools, or technologies that will need to integrate with the chosen database.

Over to you: What kinds of databases have you used before, and why were they chosen?

How do you become a full-stack developer?

The diagram shows a simplified possible full-stack tech stack.



Full stack development involves developing and managing all layers of a software application, from user interfaces to storage.

Full-stack developers need to have a broad range of technical skills including:

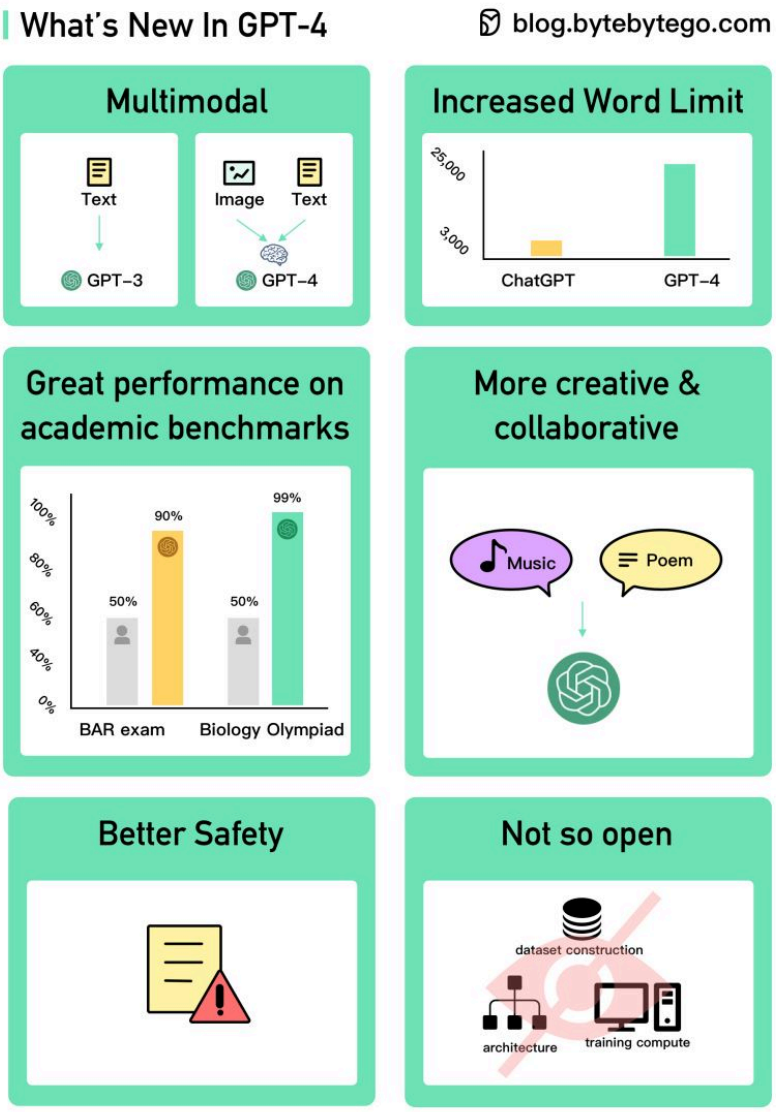
- Front-end development - HTML, Javascript, CSS, popular frameworks (React, Vue).

- API gateways - REST API gateway, GraphQL, web socket, webhook. Basic knowledge of firewall, reverse proxy, and load balancer.
- Backend development - Sever-side languages (Java, Python, Ruby), API designs, serverless cloud interactions.
- Storage - Relational databases, NoSQL databases, file storage, multimedia storage.
- External Services - Email and SMS interactions.
- DevOps skills - Full stack developers need to take care of the full lifecycle of development, including testing, deployment, and troubleshooting.

Over to you: What's your favorite full-stack setup?

What's New in GPT-4

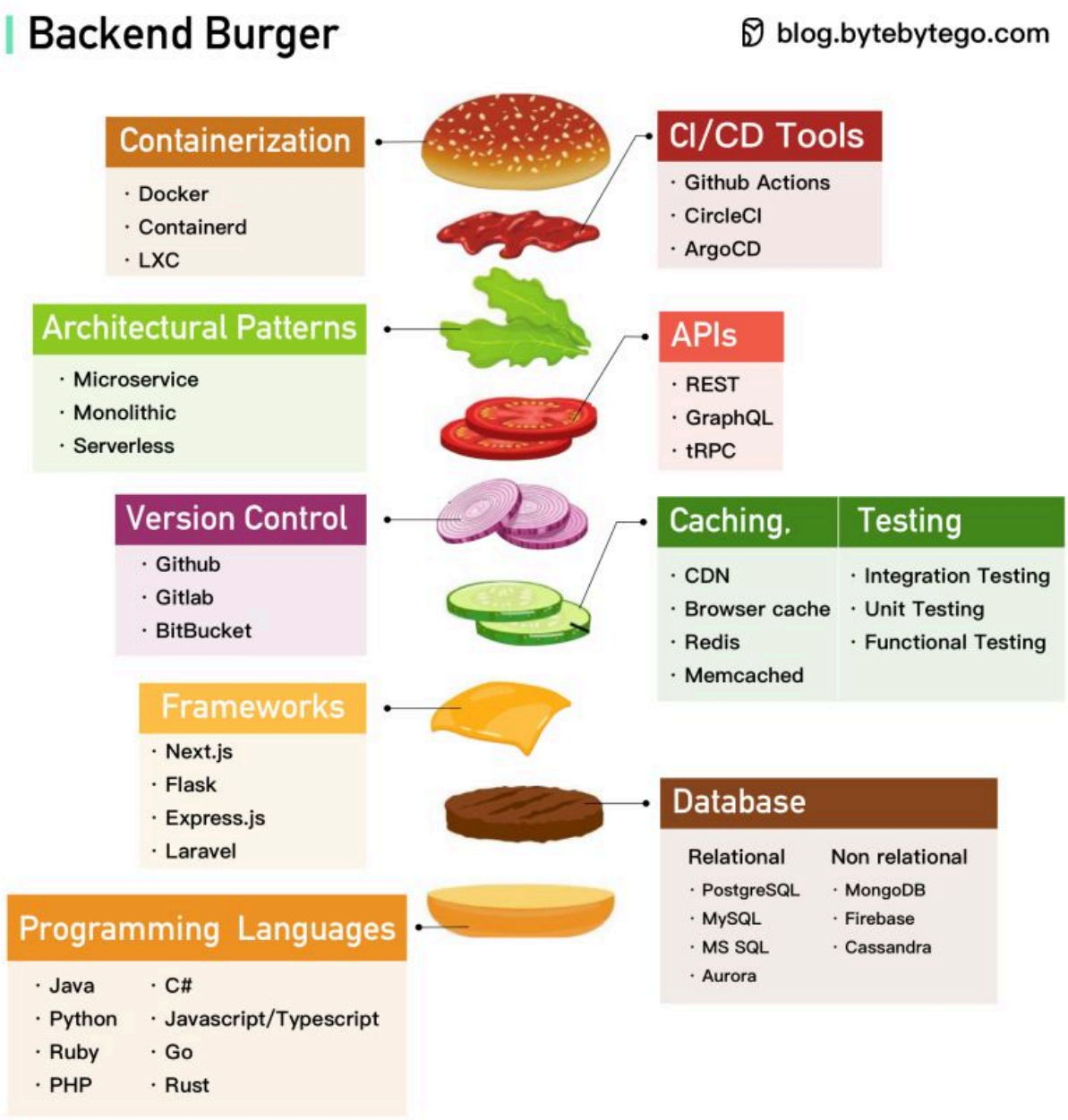
AI is evolving at a scary pace. I dove deep into the GPT-4 Technical Report and some videos, and here's what's fresh.



- Multimodal: support both image and text
- Increased word limit to 25,000
- Human-level performance on academic benchmarks
- More creative & collaborative
- Better safety
- Not so open: no further details about the architecture, hardware, training compute, etc.

Backend Burger

Everyone loves burgers, whether it's a full stack burger, a frontend burger, or a backend burger.

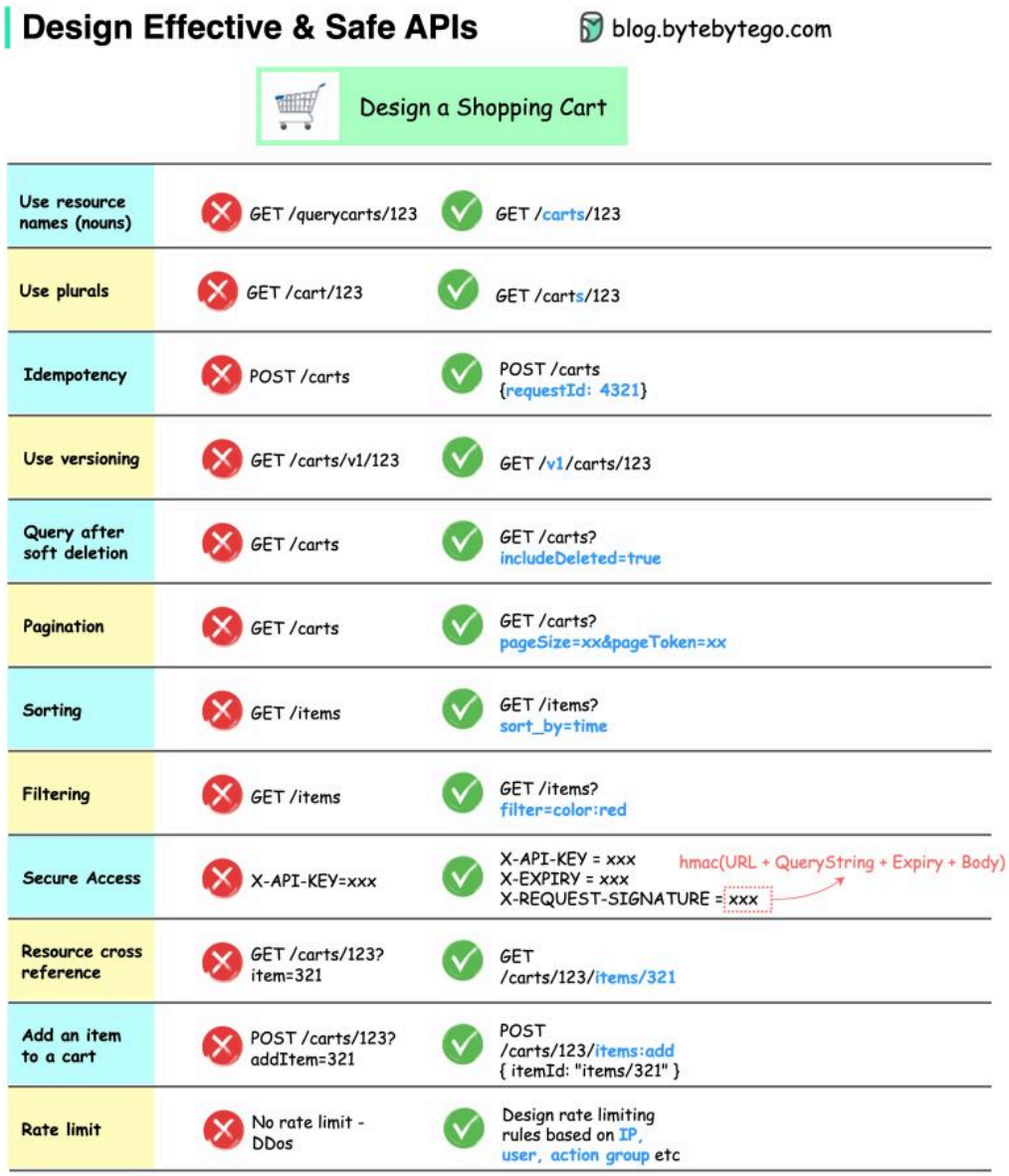


While the origin of this innovative burger is unknown, a comparable full-stack burger was shared on Reddit four years ago. We want to give a special shout-out to the original creators.

Watch & subscribe full video here: https://lnkd.in/eFKe4gHd

How do we design effective and safe APIs?

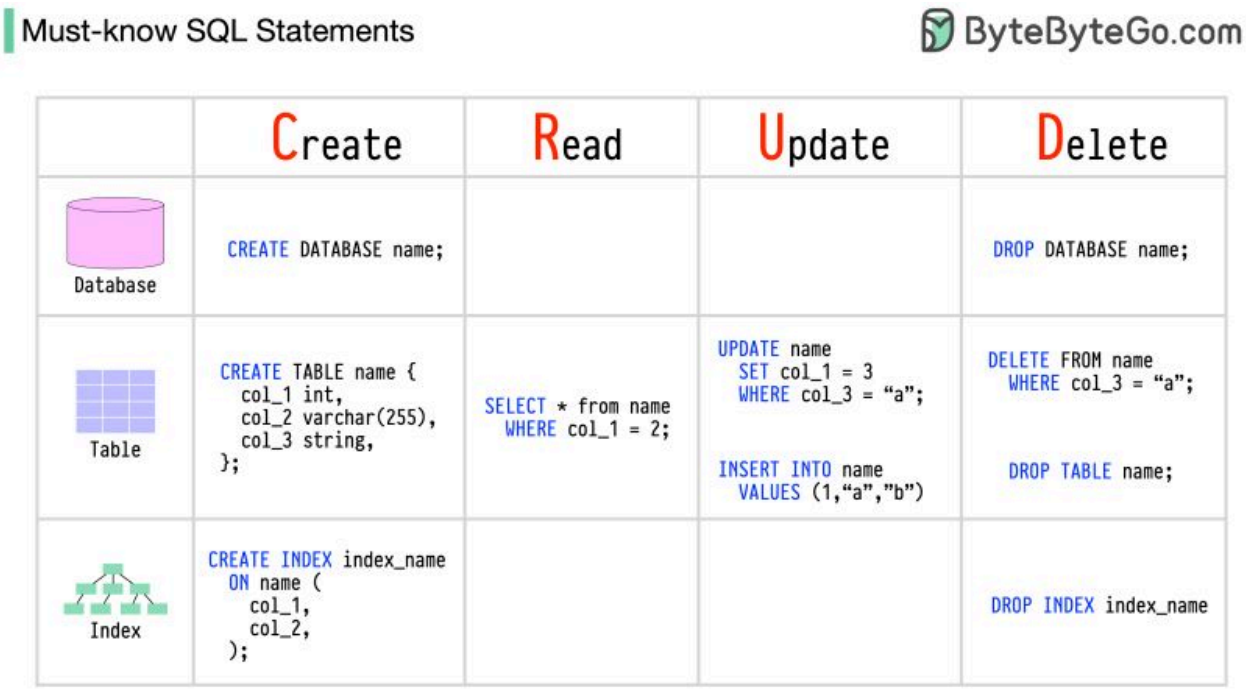
The diagram below shows typical API designs with a shopping cart example.



Note that API design is not just URL path design. Most of the time, we need to choose the proper resource names, identifiers, and path patterns. It is equally important to design proper HTTP header fields or to design effective rate-limiting rules within the API gateway.

Over to you: What are the most interesting APIs you've designed?

Which SQL statements are most commonly used?



A database consists of three types of objects:

- Database
- Table
- Index

Each object type has four operations (known as CRUD):

- Create
- Read
- Update
- Delete

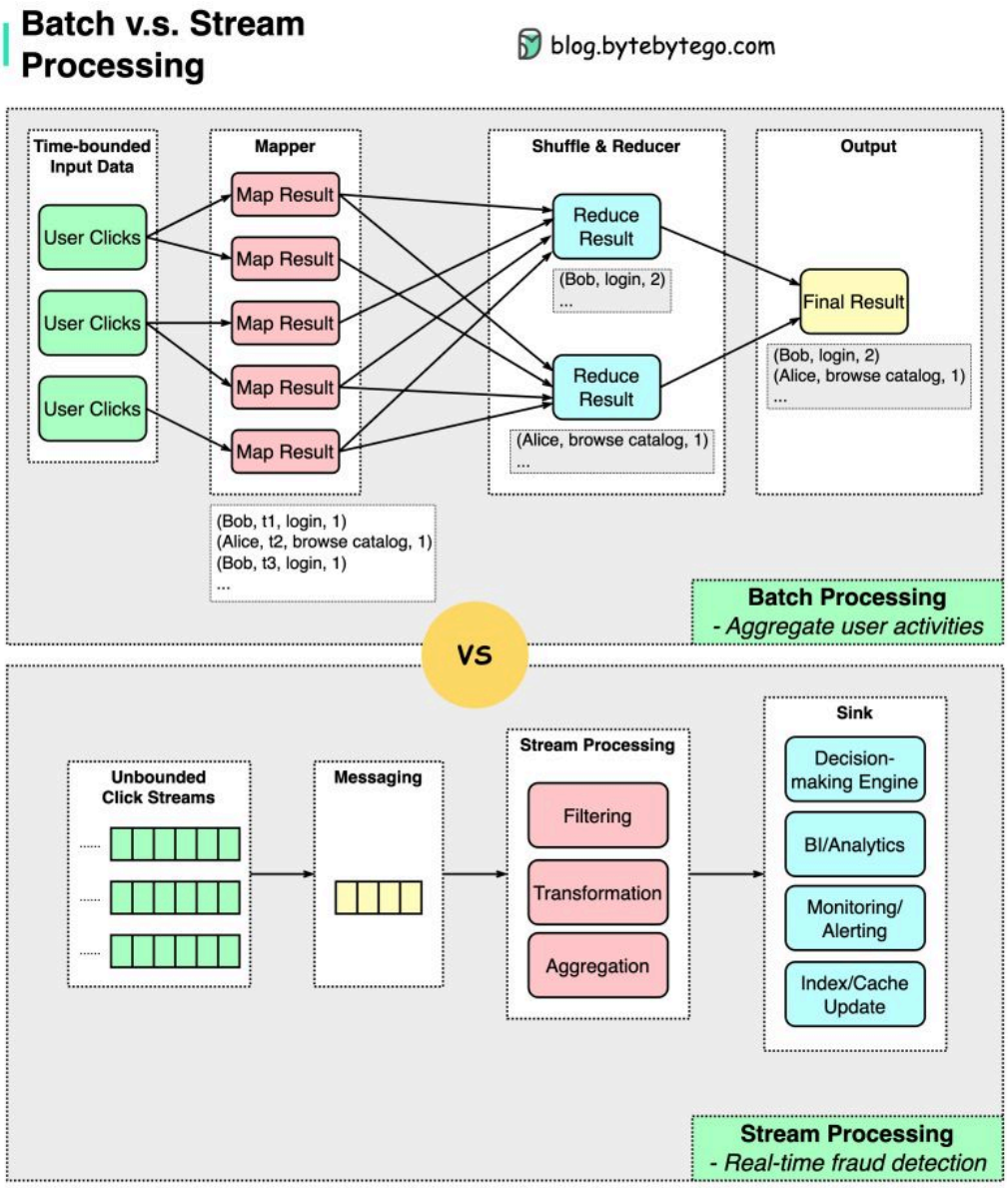
Therefore, there are a total of 12 categories of SQL statements. Some categories have been excluded from the graph because they are less commonly used. It is highly recommended that you become familiar with the remaining categories.

Over to you: I did not mention SQL statements related to transactions. In which categories do you think they should be included?

Two common data processing models: Batch v.s. Stream Processing. What are the differences?

The diagram below shows a typical scenario with user clicks:

- Batch Processing: We aggregate user click activities at end of the day.
- Stream Processing: We detect potential frauds with the user click streams in real-time.



Both processing models are used in big data processing. The major differences are:

1. Input

Batch processing works on time-bounded data, which means there is an end to the input data.

Stream processing works on data streams, which doesn't have a boundary.

2. Timelineness
   Batch processing is used in scenarios where the data doesn't need to be processed in real-time.

   Stream processing can produce processing results as the data is generated.

3. Output
   Batch processing usually generates one-off results, for example, reports.

   Stream processing's outputs can pipe into fraud decision-making engines, monitoring tools, analytics tools, or index/cache updaters.

4. Fault tolerance
   Batch processing tolerates faults better as the batch can be replayed on a fixed set of input data.

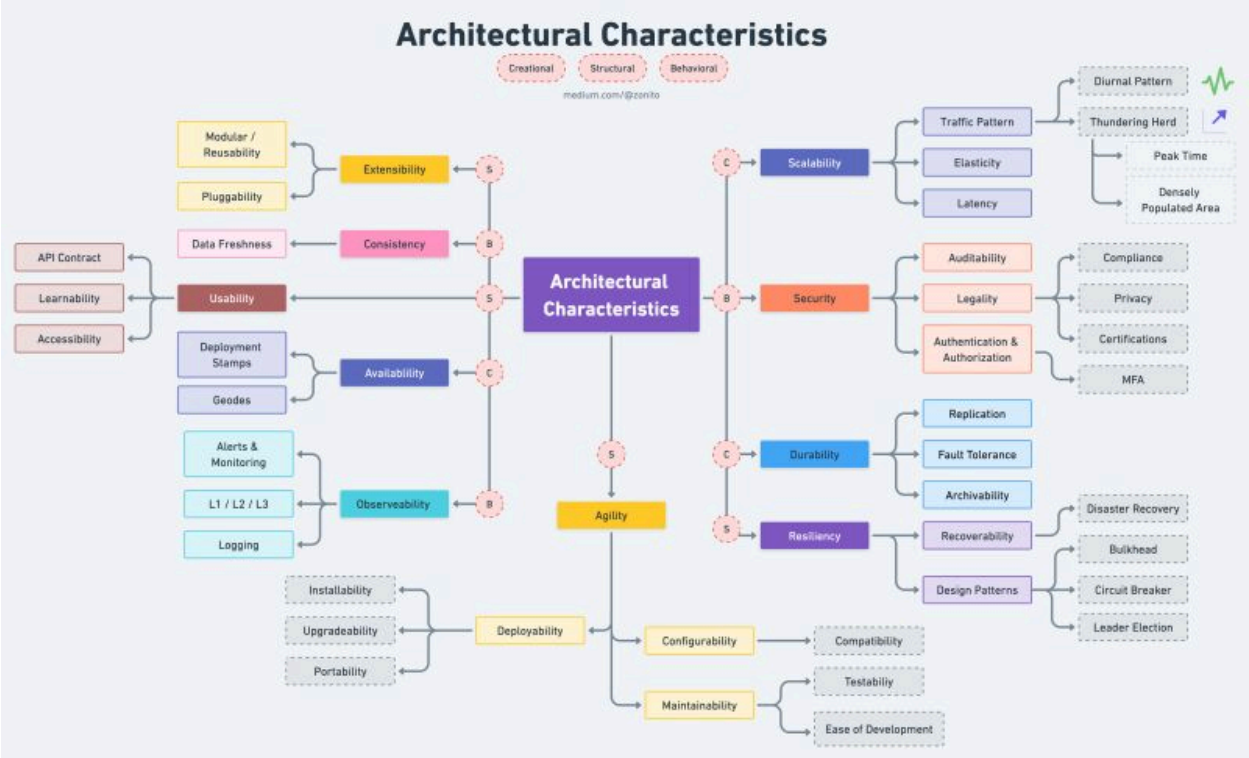
Stream processing is more challenging as the input data keeps flowing in. There are some approaches to solve this:

a. Microbatching which splits the data stream into smaller blocks (used in Spark);
b. Checkpoint which generates a mark every few seconds to roll back to (used in Flink).

👉Over to you: Have you worked on stream processing systems?

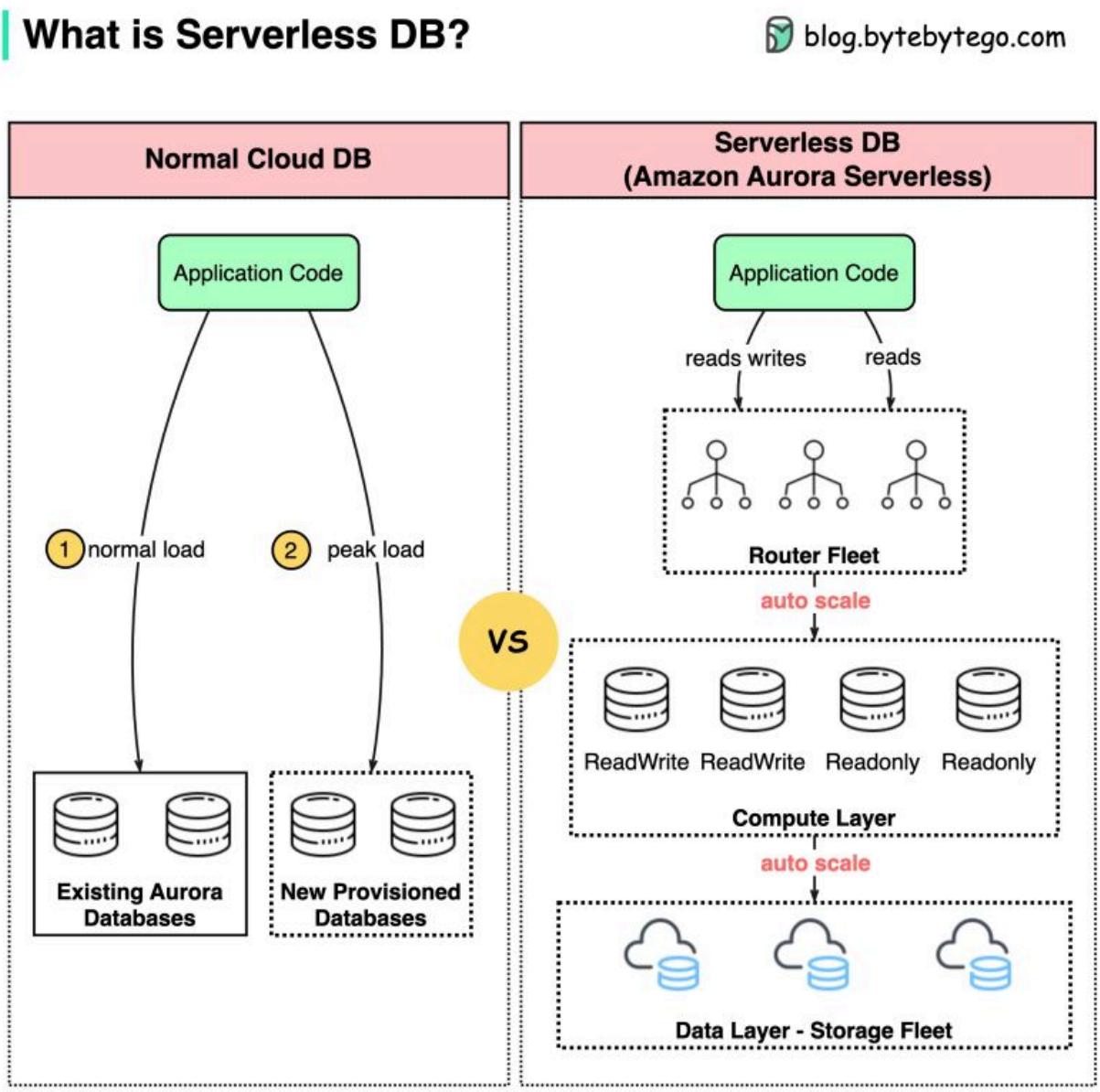
Top 10 Architecture Characteristics / Non-Functional Requirements with Cheatsheet

Did we miss anything?



Written by Love Sharma, our guest author. You can find the full article here.

Are serverless databases the future? How do serverless databases differ from traditional cloud databases?



Amazon Aurora Serverless, depicted in the diagram below, is a configuration that is auto-scaling and available on-demand for Amazon Aurora.

- Aurora Serverless has the ability to scale capacity automatically up or down as per business requirements. For example, an eCommerce website preparing for a major promotion can scale the load to multiple databases within a few milliseconds. In comparison to regular cloud databases, which necessitate the provision and

administration of database instances, Aurora Serverless can automatically start up and shut down.

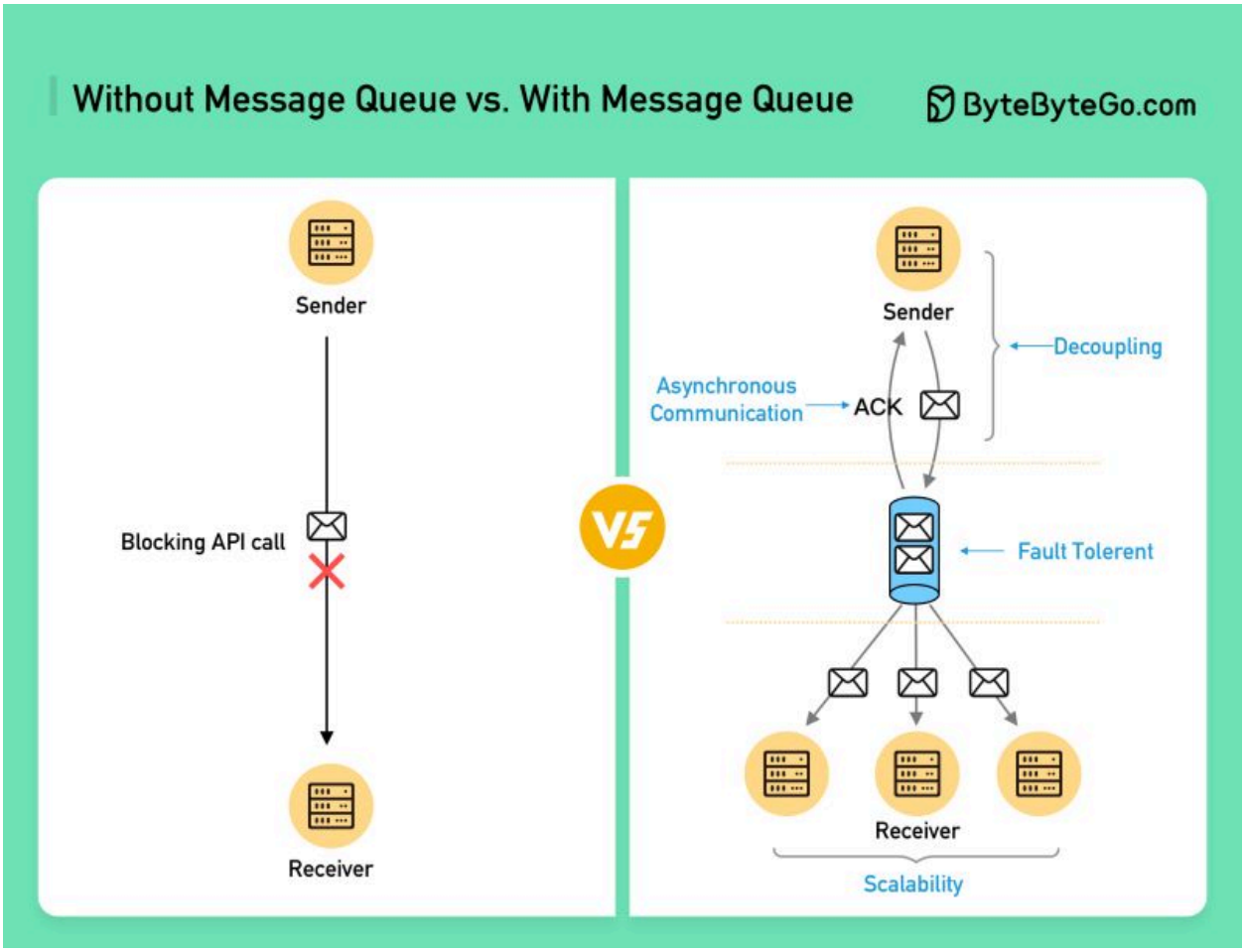
- By decoupling the compute layer from the data storage layer, Aurora Serverless is able to charge fees in a more precise manner. Additionally, Aurora Serverless can be a combination of provisioned and serverless instances, enabling existing provisioned databases to become a part of the serverless pool.

👉Over to you: Have you used a serverless DB? Does it save cost?

Reference: Amazon Aurora Serverless

Why do we need message brokers 👇?

Message brokers play a crucial role when building distributed systems or microservices to improve their performance, scalability, and maintainability.



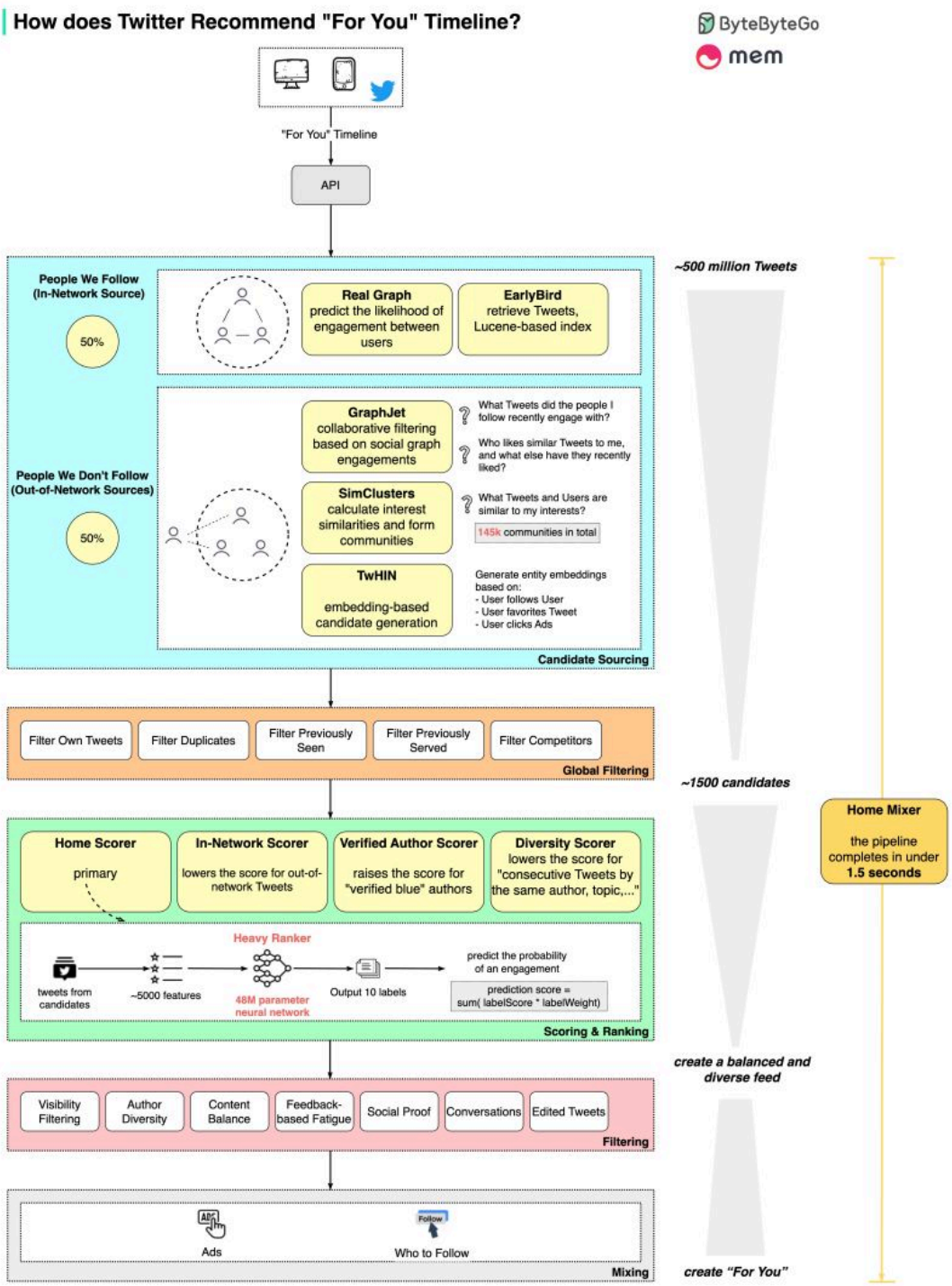
- Decoupling: Message brokers promote independent development, deployment, and scaling by creating a separation between software components. The result is easier maintenance and troubleshooting.
- Asynchronous communication: A message broker allows components to communicate without waiting for responses, making the system more efficient and enabling effective load balancing.
- Message brokers ensure that messages are not lost during component failures by providing buffering and message persistence.
- Scalability: Message brokers can manage a high volume of messages, allowing your system to scale horizontally by adding more instances of the message broker as needed.

To summarize, a message broker can improve efficiency, scalability, and reliability in your architecture. Considering the use of a message broker can greatly benefit the long-term success of your application. Always think about the bigger picture, and how your design choices will affect the overall project.

Over to you: there is a term called pub/sub. Are you familiar with their meanings?

How does Twitter recommend “For You” Timeline in 1.5 seconds?

We spent a few days analyzing it. The diagram below shows the detailed pipeline based on the open-sourced algorithm.



The process involves 5 stages:

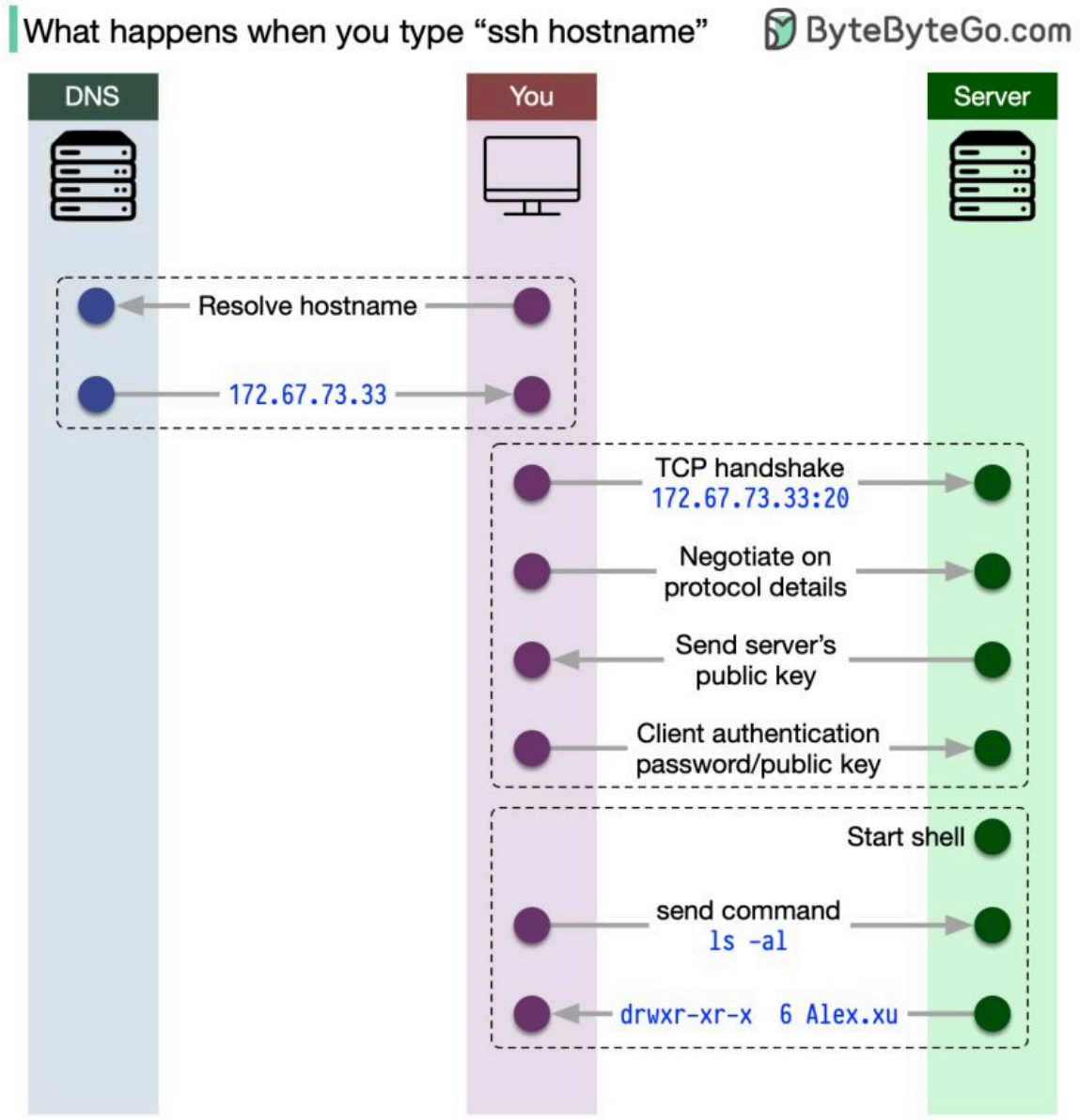
- Candidate Sourcing ~ start with 500 million Tweets
- Global Filtering ~ down to 1500 candidates
- Scoring & Ranking ~ 48M parameter neural network, Twitter Blue boost
- Filtering ~ to achieve author and content diversity
- Mixing ~ with Ads recommendation and Who to Follow

The post was jointly created by ByteByteGo and Mem
Special thanks Scott Mackie , founding engineer at Mem, for putting this together.

Mem is building the world's first knowledge assistant. In next week's ByteByteGo guest newsletter, Mem will be sharing lessons they've learned from their extensive work with large language models and building AI-native infrastructure.

Popular interview question: what happens when you type “ssh hostname”?

In the 1990s, Secure Shell was developed to provide a secure alternative to Telnet for remote system access and management. Using SSH is a great way to set up secure communication between client and server because it uses a secure protocol.



The following happens when you type "ssh hostname":

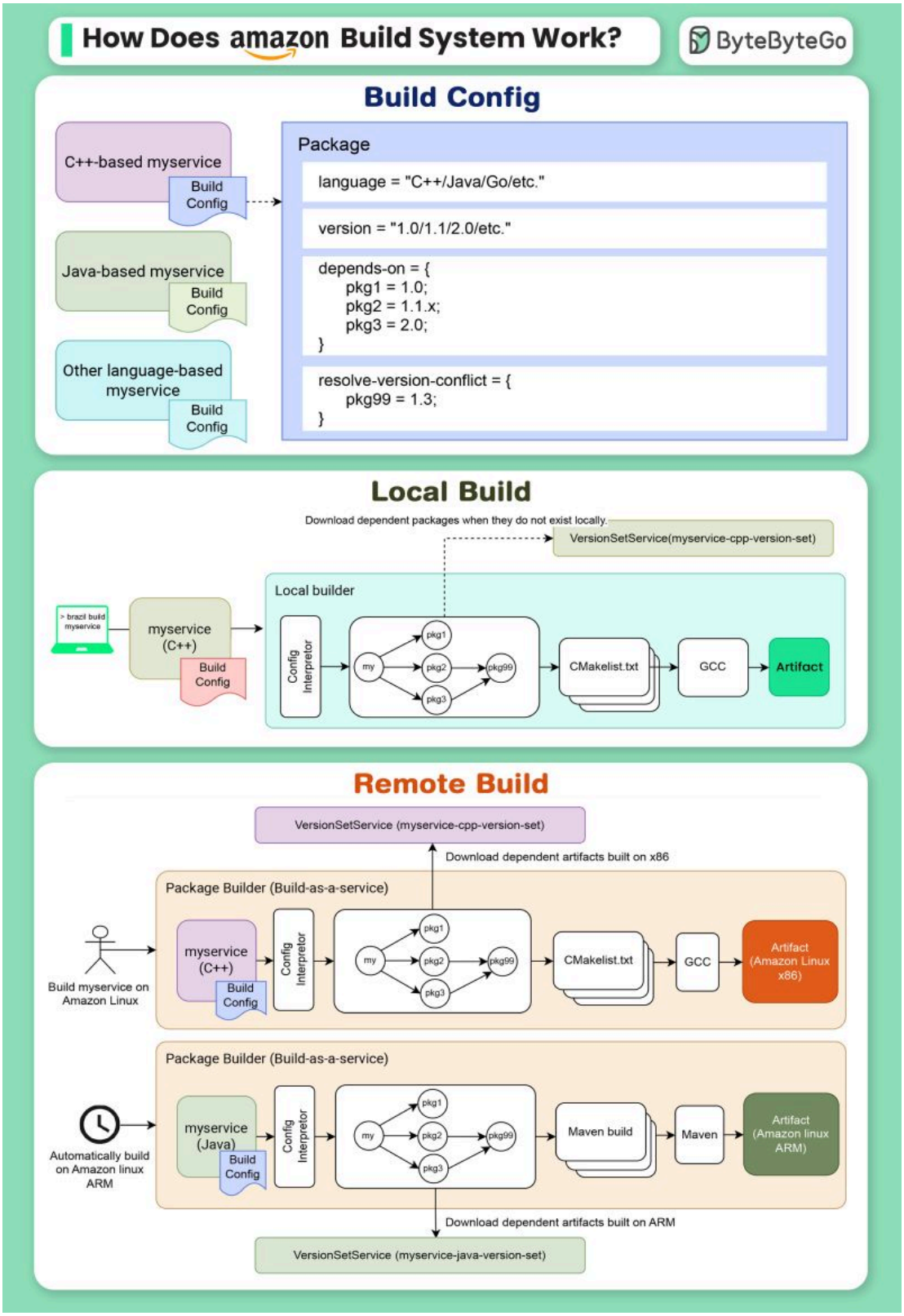
- Hostname resolution: Convert the hostname to an IP address using DNS or the local hosts file.
- SSH client initialization: Connect to the remote SSH server.
- TCP handshake: Establish a reliable connection.

- Protocol negotiation: Agree on the SSH protocol version and encryption algorithms.
- Key exchange: Generate a shared secret key securely.
- Server authentication: Verify the server's public key.
- User authentication: Authenticate using a password, public key, or another method.
- Session establishment: Create an encrypted SSH session and access the remote system.

Make sure you always use key-based authentication with SSH for better security, and learn SSH configuration files and options to customize your experience. Keep up with best practices and security recommendations to ensure a secure and efficient remote access experience.

Over to you: can you tell the difference between SSH, SSL, and TLS?

Discover Amazon's innovative build system - Brazil.



Amazon's ownership model requires each team to manage its own repositories, which allows for more rapid innovation. Amazon has created a unique build system, known as Brazil, to enhance productivity and empower Amazon’s micro-repo driven collaboration. This system is certainly worth examining!

With Brazil, developers can focus on developing the code and create a simple-to-understand build configuration file. The build system will then process the output artifact repeatedly and consistently. The build config minimizes the build requirement, including language, versioning, dependencies, major versions, and lastly, how to resolve version conflicts.

For local builds, the Brazil build tool interprets the build configuration as a Directed Acyclic Graph (DAG), retrieves packages from the myservice’s private space (VersionSet) called myservice-cpp-version-set, generates the language-specific build configuration, and employs the specific build tool to produce the output artifact.

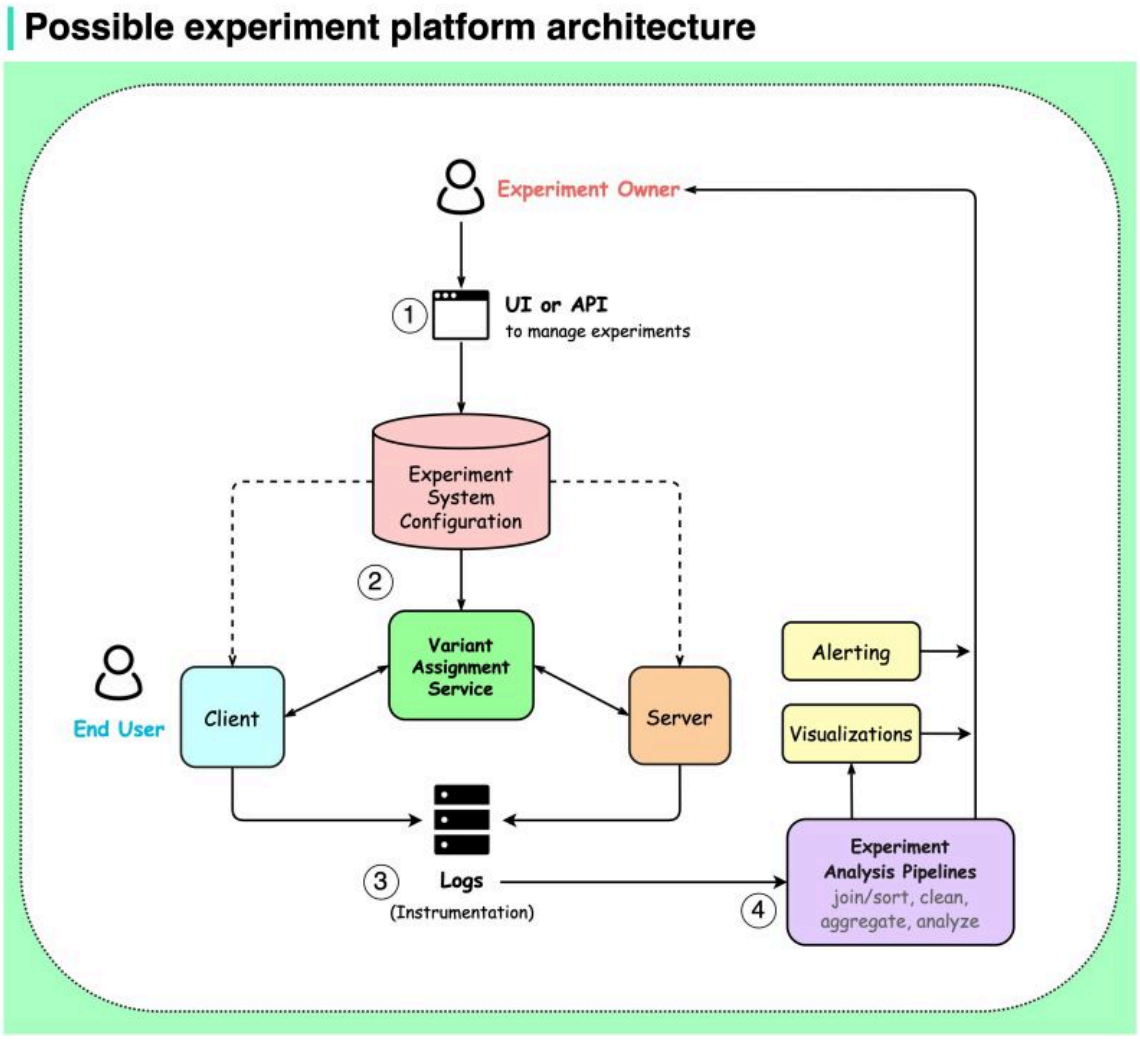
A version set is a collection of package versions that offers a private space for the package and its dependencies. When a new package dependency is introduced, it must also be merged into this private space. There is a default version set called "live," which serves as a public space where anyone can publish any version.

Remotely, the package builder service provides an intuitive experience by selecting a version set and building targets. This service supports Amazon Linux on x86, x64, and ARM. Builds can be initiated manually or automatically upon a new commit to the master branch. The package builder guarantees build consistency and reproducibility, with each build process being snapshotted and the output artifact versioned.

Over to you - which type of build system did you use?

Possible Experiment Platform Architecture

The architecture of a potential experiment platform is depicted in the diagram below. This content of the visual is from the book: "Trustworthy Online Controlled Experiments" (redrawn by me). The platform contains 4 high-level components.

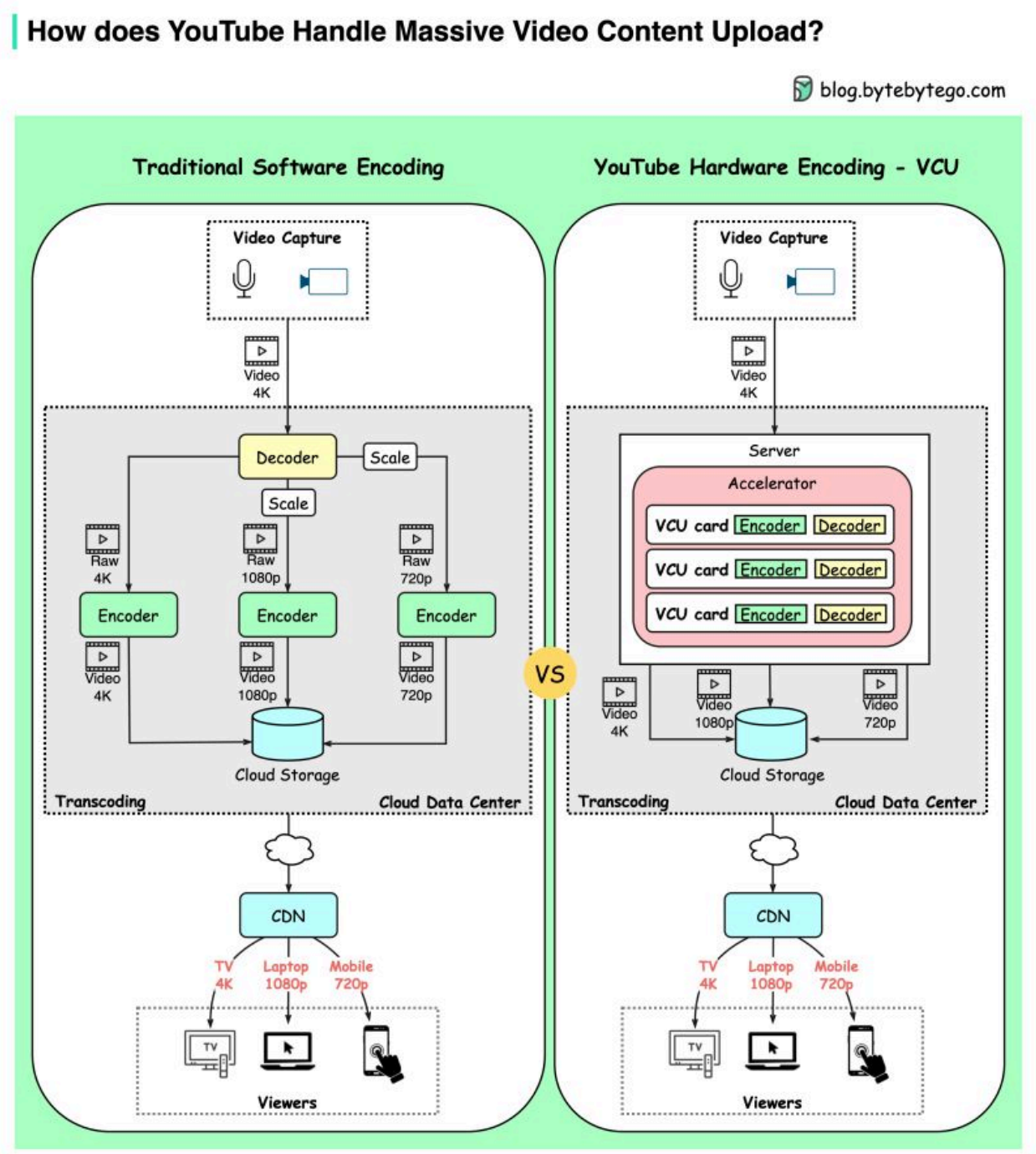


1. Experiment definition, setup, and management via a UI. They are stored in the experiment system configuration.
2. Experiment deployment to both the server and client-side (covers variant assignment and parameterization as well).
3. Experiment instrumentation.
4. Experiment analysis.

The book's author Ronny Kohavi also teaches a live Zoom class on Accelerating Innovation with A/B Testing. The class focuses on concepts, culture, trust, limitations, and build vs. buy. You can learn more about it here: https://lnkd.in/eFHVuAKq

YouTube handles 500+ hours of video content uploads every minute on average. How does it manage this?

The diagram below shows YouTube's innovative hardware encoding published in 2021.



- Traditional Software Encoding

YouTube's mission is to transcode raw video into different compression rates to adapt to different viewing devices - mobile(720p), laptop(1080p), or high-resolution TV(4k).

Creators upload a massive amount of video content on YouTube every minute. Especially during the COVID-19 pandemic, video consumption is greatly increased as people are sheltered at home. Software-based encoding became slow and costly. This means there was a need for a specialized processing brain tailored made for video encoding/decoding.

- YouTube's Transcoding Brain - VCU

  Like GPU or TPU was used for graphics or machine learning calculations, YouTube developed VCU (Video transCoding Unit) for warehouse-scale video processing.

  Each cluster has a number of VCU accelerated servers. Each server has multiple accelerator trays, each containing multiple VCU cards. Each card has encoders, decoders, etc. [1]

  VCU cluster generates video content with different resolutions and stores it in cloud storage.

  This new design brought 20-33x improvements in computing efficiency compared to the previous optimized system. [2]

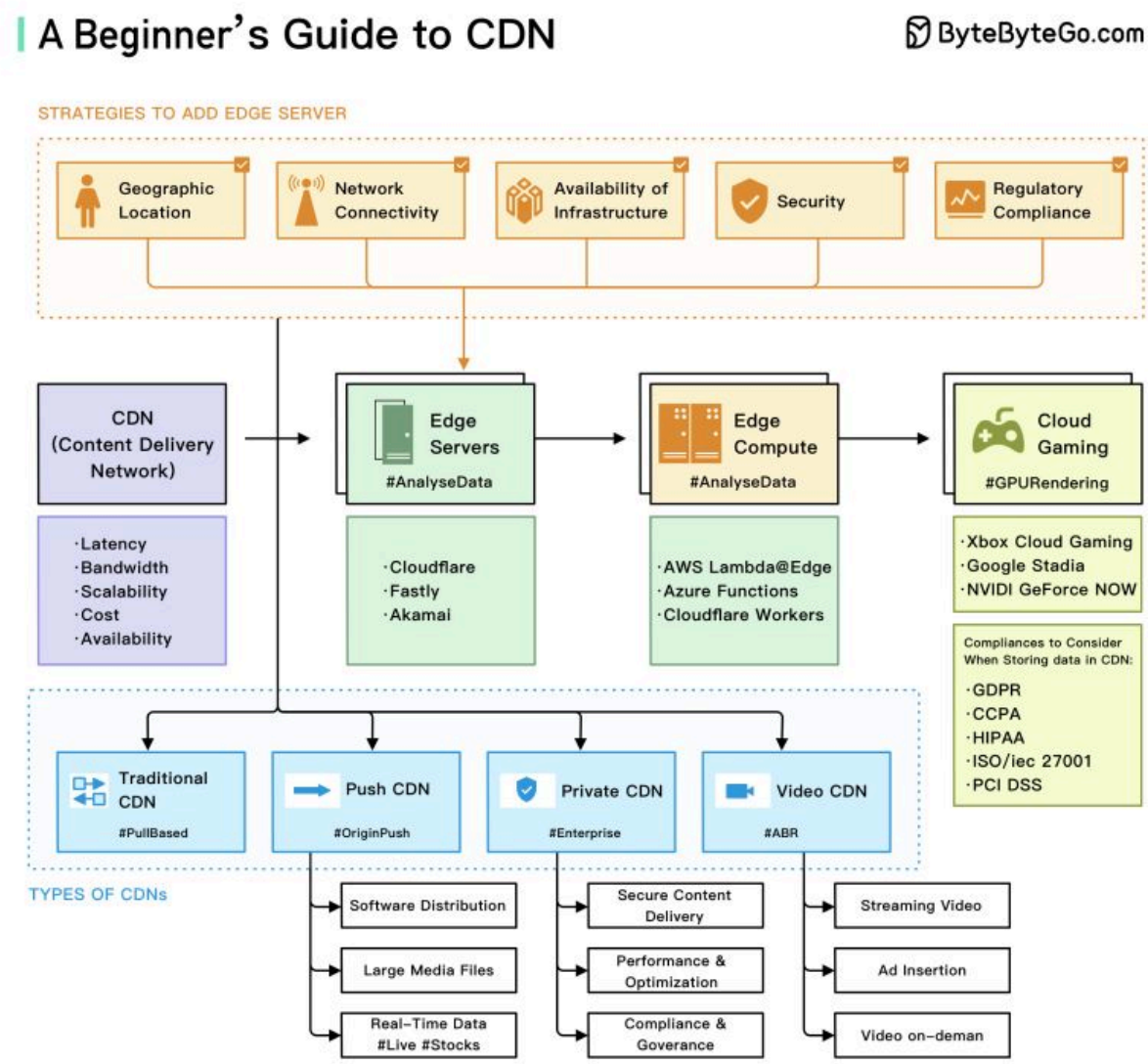
👉 Over to you: Why is a specialized chip so much faster than a software-based solution?

Reference:
[1] Warehouse-scale video acceleration: co-design and deployment in the wild
[2] Reimagining video infrastructure to empower YouTube

A beginner's guide to CDN (Content Delivery Network)

A guest post by Love Sharma. You can read the full article here.

CDNs are distributed server networks that help improve the performance, reliability, and security of content delivery on the internet.



The Overall CDN Diagram explains:

Edge servers are located closer to the end user than traditional servers, which helps reduce latency and improve website performance.

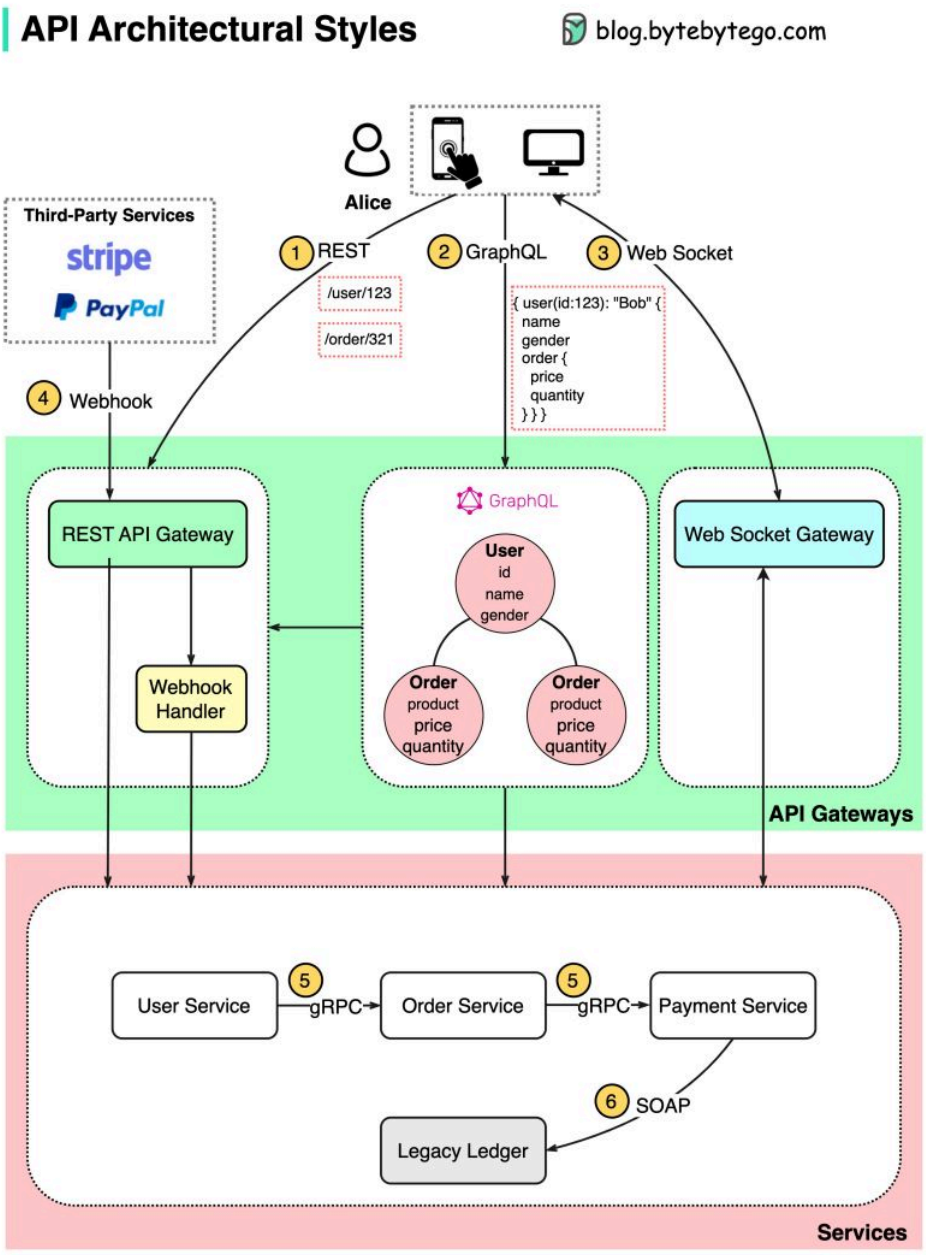
Edge computing is a type of computing that processes data closer to the end user rather than in a centralized data center. This helps to reduce latency and improve the performance of applications that require real-time processing, such as video streaming or online gaming.

Cloud gaming is online gaming that uses cloud computing to provide users with high-quality, low-latency gaming experiences.

Together, these technologies are transforming how we access and consume digital content. By providing faster, more reliable, and more immersive experiences for users, they are helping to drive the growth of the digital economy and create new opportunities for businesses and consumers alike.

What are the API architectural styles?

The diagram below shows the common API architectural styles in one picture.



1. REST
   Proposed in 2000, REST is the most used style. It is often used between front-end clients and back-end services. It is compliant with 6 architectural constraints. The payload format can be JSON, XML, HTML, or plain text.

2. GraphQL

GraphQL was proposed in 2015 by Meta. It provides a schema and type system, suitable for complex systems where the relationships between entities are graph-like. For example, in the diagram below, GraphQL can retrieve user and order information in one call, while in REST this needs multiple calls.

GraphQL is not a replacement for REST. It can be built upon existing REST services.

3. Web socket
   Web socket is a protocol that provides full-duplex communications over TCP. The clients establish web sockets to receive real-time updates from the back-end services. Unlike REST, which always “pulls” data, web socket enables data to be “pushed”.

4. Webhook
   Webhooks are usually used by third-party asynchronous API calls. In the diagram below, for example, we use Stripe or Paypal for payment channels and register a webhook for payment results. When a third-party payment service is done, it notifies the payment service if the payment is successful or failed. Webhook calls are usually part of the system’s state machine.

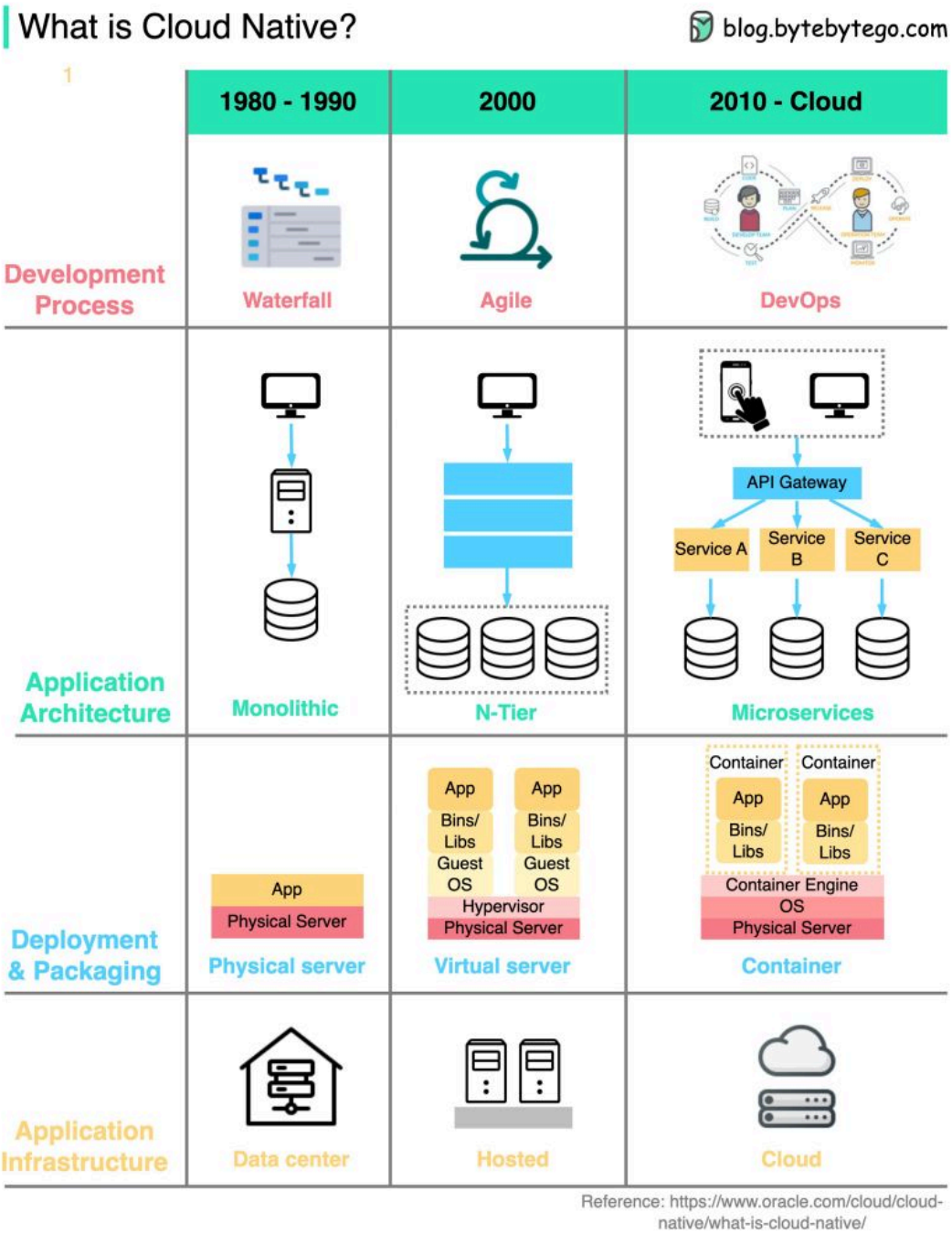
5. gRPC
   Released in 2016, gRPC is used for communications among microservices. gRPC library handles encoding/decoding and data transmission.

6. SOAP
   SOAP stands for Simple Object Access Protocol. Its payload is XML only, suitable for communications between internal systems.

👉 Over to you: What API architectural styles have you used?

Cloud-native vs. Cloud computing

The term "Cloud Native" seemed to first appear around 10 years ago when Netflix discussed their web-scale application architecture at a 2013 AWS re:Invent talk.



Reference: https://www.oracle.com/cloud/cloud-native/what-is-cloud-native/

At that time, the meaning of the term was likely different than it is today. However, one thing remains the same: there were no clear definitions for it then, and there still are not any clear definitions now. It means different things to different people.

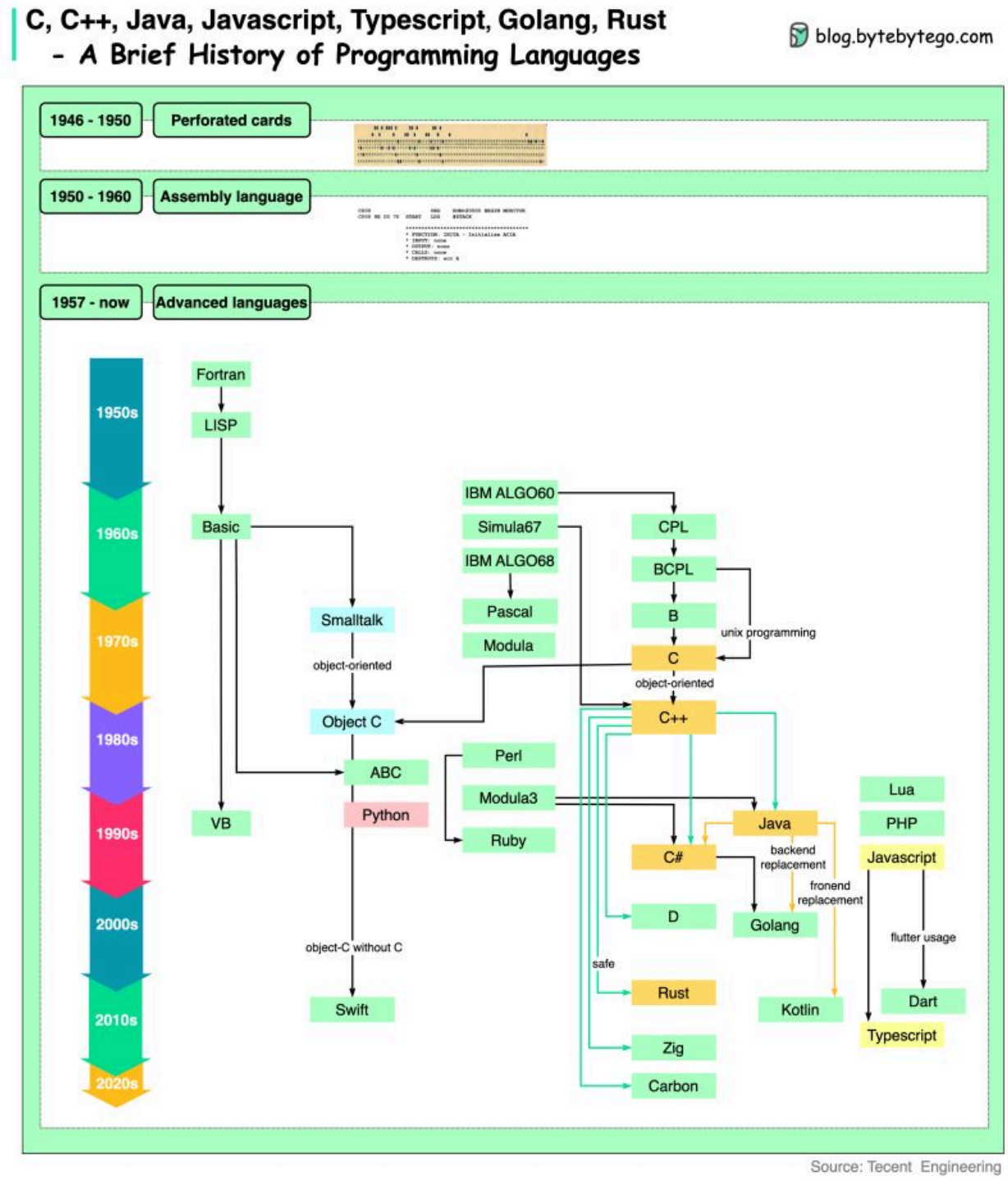
In this video, we provide our interpretation of the term "Cloud Native" and discuss when it is important.

Watch and subscribe here: https://lnkd.in/evAqzU9V

C, C++, Java, Javascript, Typescript, Golang, Rust…

How do programming languages evolve for the past 70 years?

The diagram below shows a brief history of programming languages.

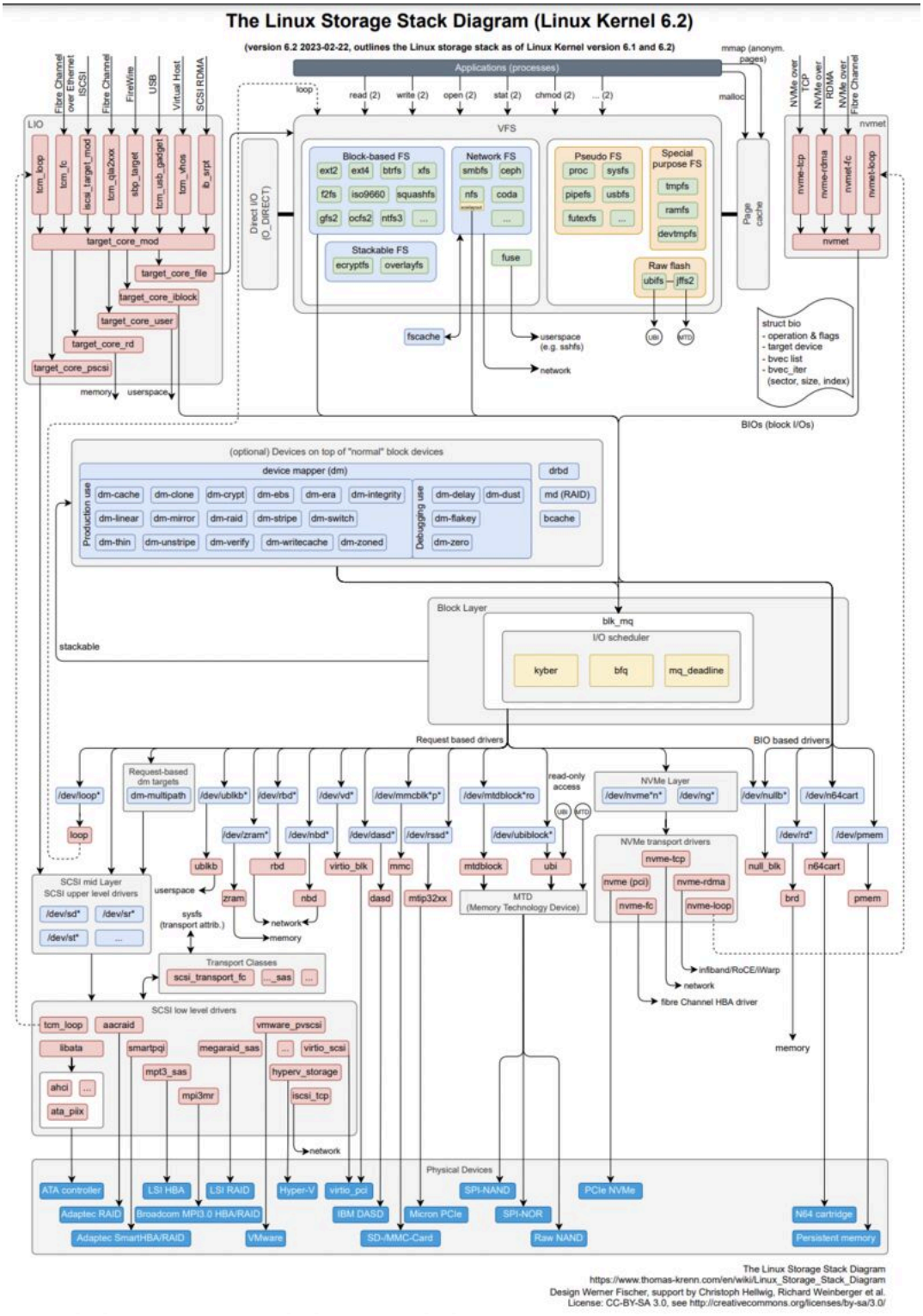


- Perforated cards were the first generation of programming languages. Assembly languages, which are machine-oriented, are the second generation of programming languages. Third-generation languages, which are human-oriented, have been around since 1957.
- Early languages like Fortran and LISP proposed garbage collection, recursion, exceptions. These features still exist in modern programming languages.
- In 1972, two influential languages were born: Smalltalk and C. Smalltalk greatly influenced scripting languages and client-side languages. C language was developed for unix programming.
- In the 1980s, object-oriented languages became popular because of its advantage in graphic user interfaces. Object-C and C++ are two famous ones.
- In the 1990s, the PCs became cheaper. The programming languages at this stage emphasized security and simplicity. Python was born in this decade. It was easy to learn and extend and it quickly gained popularity. In 1995, Java, Javascript, PHP and Ruby were born.
- In 2000, C# was released by Microsoft. Although it was bundled with .NET framework, this language carried a lot of advanced features.
- A number of languages were developed in the 2010s to improve C++ or Java. In the C++ family, we have D, Rust, Zig and most recently Carbon. In the Java family, we have Golang and Kotlin. The use of Flutter made Dart popular, and Typescript was developed to be fully compatible with Javascript. Also, Apple finally released Swift to replace Object-C.

👉 Over to you: What's your favorite language and why? Will AI change the way we use programming languages?

The Linux Storage Stack Diagram shows the layout of the the Linux storage stack

Diagram by Thomas-Krenn



Breaking down what's going on with the Silicon Valley Bank (SVB) collapse