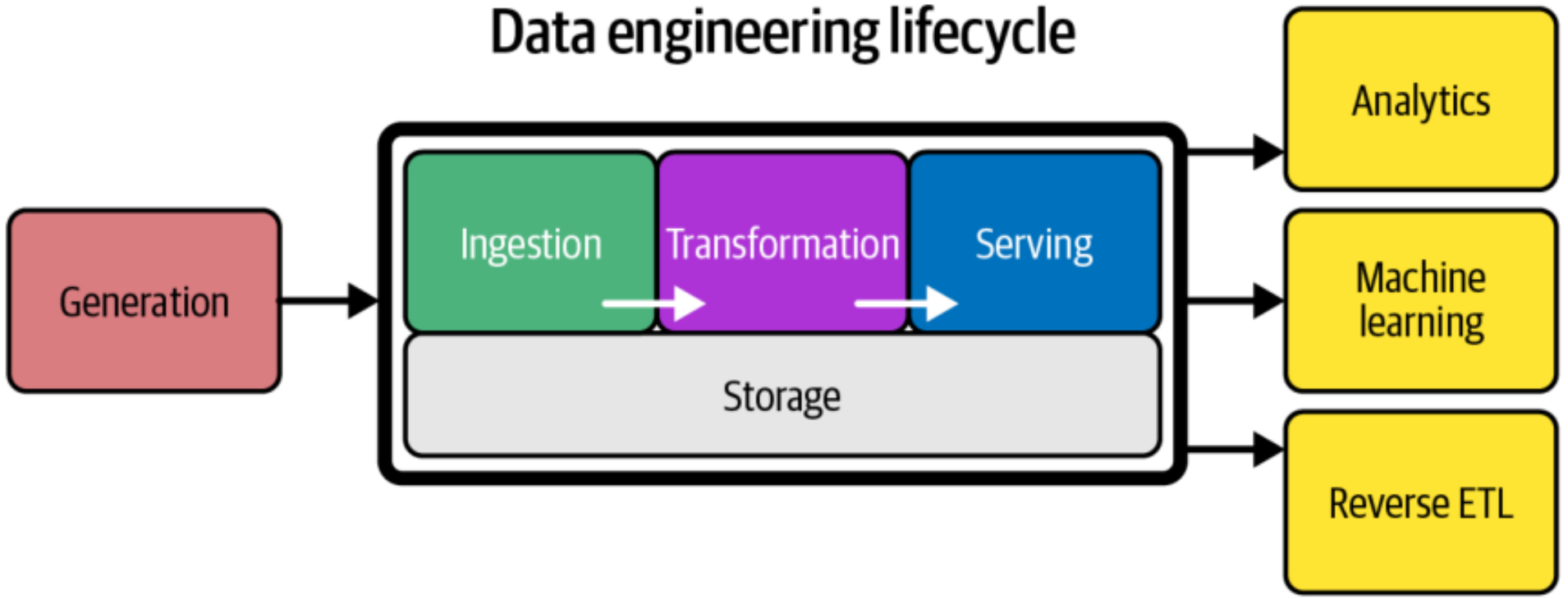
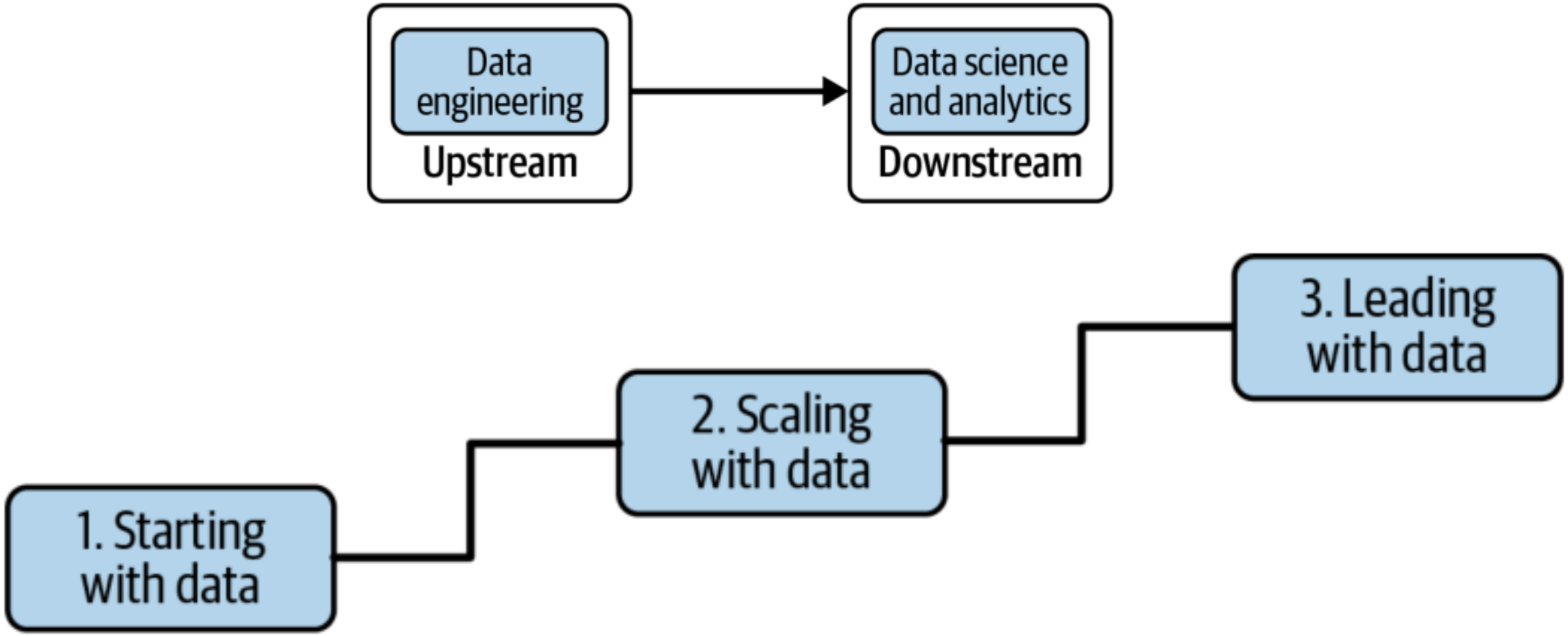
Data Engineering Theory and advices

A high level view of Data Engineering ecosystem:



Up and Downstream and Data Maturity Model:



A data Engineer must understand:

1. **داده:** جنبه‌های مختلف داده و مدیریت آنرا بشناسد.
2. **تکنولوژی:** ابزارهای مختلف کار با دیتا و ارتباطشان با یکدیگر را درک نماید.

Two types of data Engineers:

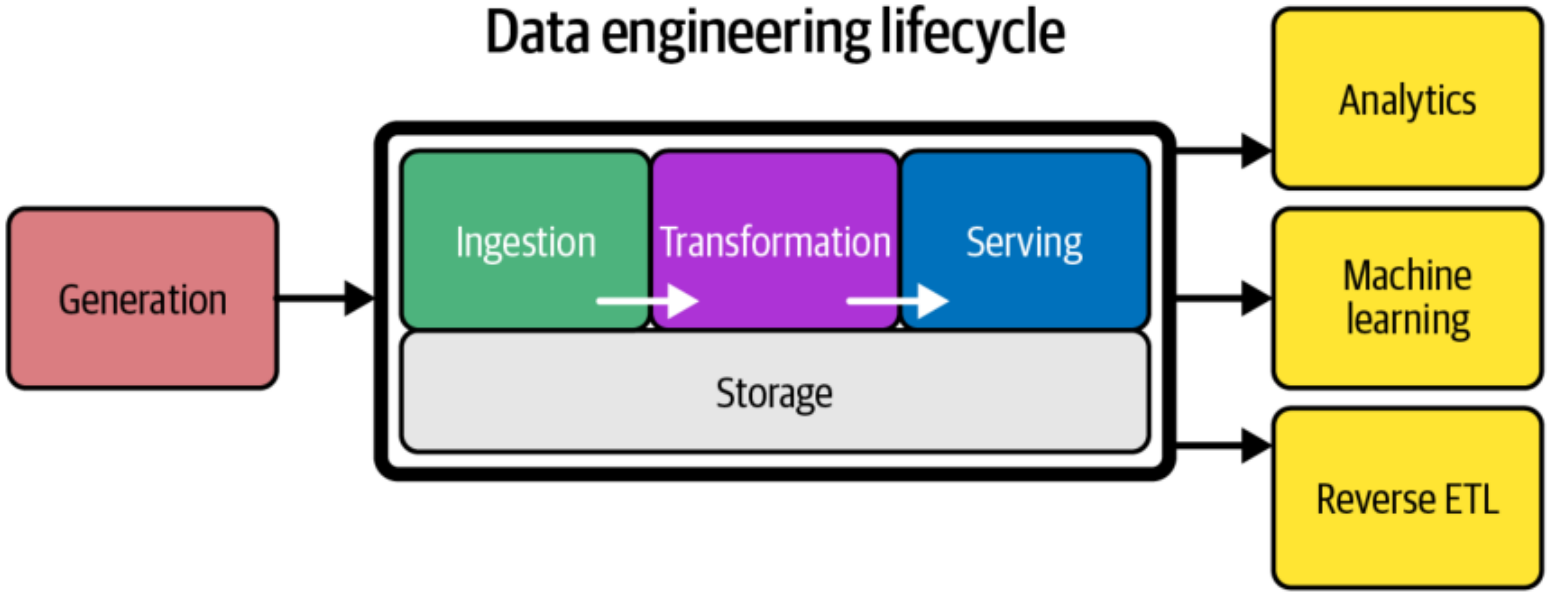
Type A (A stands for abstraction):

مهندسین داده در این سطح و مرحله، از انجام کارهای سخت و پیچیده پرهیز می‌نمایند. آنها تلاش می‌کنند که معماری‌های مختلف داده را تا حد ممکن بصورت مجرد (روی کاغذ پیاده کنند که به مرحله‌ی واقعیت هنوز نرسیده است) تهیه و تدوین نمایند. این سطح از مهندسین عمومن چیزی را ابداع نمی‌کنند و از ابزارهای آماده استفاده می‌کنند.

Type B (B stands for build):

سازندگان اصلی پایپ‌لاین‌ها، این سطح از مهندسین هستند. در اکثر موارد به ابداع سیستم‌های جدید می‌پردازند.

Five stages in data Engineering lifecycle:



1- Generation: Source Systems:

منبع داده می‌تواند یک دستگاه IoT، یک دیتابیس، کلیک‌های یک وبسایت و یا هر چیز دیگر باشد. مهندسین داده باید درک درستی از منابع داده پیدا کنند و نحوه‌ی کار با آنها را بخوبی بشناسند. دقت کنید که مهندس داده مالک یا کنترل کننده‌ی منابع دیتا نیست.

2- Storage:

یکی از پیچیده‌ترین (Most Complicated) مراحل در مهندسی داده، مرحله‌ی ذخیره‌سازی داده‌هاست. چون تا حد زیادی به پول و مصرف منابع مالی مرتبط است. در فاز ذخیره‌سازی داده، باید فراوانی دسترسی به دیتا (Data Access Frequency) را خوب بفهمیم.

Hot Data

- این دسته از داده‌ها، عمومن هر روزه بازیابی شده و مورد استفاده قرار می‌گیرند

Lukewarm Data

- این دسته از داده‌ها، شاید هفته یا ماهی یکبار مورد استفاده قرار بگیرند

Cold Data

- این دسته از داده‌ها بندرت مورد استفاده یا کوئری قرار خواهند گرفت. ولی اطلاعاتی هستند که شاید بعدها بدرد بخورند. پس مجبور هستیم که آنها را ذخیره کنیم

Data Storage in cloud environment is VERY CHEAP

but data retrieval is expensive!

انتخاب راه‌حل برای ذخیره‌سازی خیلی مهم است. این مورد به استفاده‌های ما از دیتا، حجم داده، فراوانی دریافت (هضم) داده و فرمت دیتاهای ورودی وابسته است.

3- Ingestion:

بیشترین گیر و خطا (یا حتی بن‌بست) در مرحله‌ی هضم دیتا از منابع مختلف داده رخ می‌دهد.

بطور کلی در مرحله‌ی هضم، دو مفهوم مهم وجود دارد

Batch:

بعنوان مثال در پایان روز کاری، اطلاعات فروش را در آن روز مورد پردازش قرار می‌دهیم (هضم بصورت بچ از منابع دیتا (دیتابیس) به سیستم پردازشی).

Streaming:

انتقال دیتا به سیستم‌های پائین‌دستی (Downstream) بصورت پیوسته و در لحظه (دیتای در لحظه، یعنی بلافاصله پس از تولید به پائین‌دستی‌ها برسد)

زمانی که منابع محاسباتی خوب نبودند، پردازش بصورت بچ فراگیر بود. اما امروزه دریافت و پردازش استریم طرفداران بیشتری دارد.

در بخش هضم، دو واژه‌ی دیگر نیز وجود دارد و حائز اهمیت می‌باشند:

Push vs. Pull:

در مدل پوش، یک منبع داده، دیتا را بسمت هدف ارسال می‌کند (می‌نویسد). هدف می‌تواند یک دیتابیس، دیتالیک و یا یک سیستم فایلی باشد. در مدل پول، داده از منبع دیتا بازیافت (retrieved) می‌شود.

4- Transformation:

در فاز تبدیل، داده‌های ما به ارزش برای مصرف کننده‌ی پائین‌دستی بدل می‌گردند. چون پائین‌دستی‌ها از هر داده‌ای نمی‌توانند بهره ببرند. در یک سازمان، به محض اینکه دیتاساینتیست‌ها تشخیص دادند که از چه ویژگی‌هایی از دیتا، قرار است استفاده نمایند، تیم مهندسی داده فرایند استخراج این ویژگی‌ها و در اختیار قرار دادن آنها به سیستم تحلیل را بصورت اتوماتیک تبدیل می‌نمایند.

5- Serving Data

در اختیار مصرف‌کنندگان قرار دادن دیتا (سرو دیتا) یکی از هیجان‌انگیزترین فازها در مهندسی داده است. باید در این فاز بررسی کنیم که چه کسانی و چه چیزهایی ممکن است مصرف کننده‌ی داده باشند. چند مورد را در ادامه می‌بینیم:

I. Analytics

II. ML

III. Reverse ETL (New tech and trend)

یک مثال از ای‌تی‌ال وارونه اینگونه است؛ دیتا از سمت دیتاورهوز بسمت تیم مارکتینگ می‌رود برای استفاده‌ی بهینه، در واقع برای ارسال ایمیل‌های تبلیغاتی برای مشتریان متعهد سازمان.

Designing a Good Data Architecture

معماری داده، شرحی از ساختارها و تعاملات مولفه‌های اصلی و منابع دیتایی یک سازمان است.

"Never shoot for the best architecture, but rather the least worst architecture"

Mark Richards and Neal Ford, Fundamentals of software architecture

یک معماری داده‌ی خوب، در خدمت بیزینس و سازمان است و مولفه‌هایی دارد که قابل استفاده‌ی مجدد هستند. در حالی که یک تعادل مطلوبی بین عملکرد و هزینه را برقرار می‌سازد.

9 Principles of Data Engineering Architecture

1- Choose common components wisely:

نیاز به ابداع و طراحی مولفه‌های جدید نیست. عمومن تیم‌ها تشویق می‌شوند که از مولفه‌های معمول و مرسوم استفاده کنند.

2- Plan for Failure

حتمن بحث failure یا مشکل را در طراحی خود در نظر بگیرید. بقول آقای ووگل (مدیر تکنولوژی آمازون):

Everything fails, all the time

موارد زیر را در طراحی خود (در این فاز) در نظر بگیرید:

Availability, Reliability, Recovery time objective(RTO), Recovery point objective (RPO)

RTO: بیشترین زمان قابل قبول برای از دسترس خارج شدن یک سرویس

RPO: بیشترین مقدار از دست‌رفت داده بعد از هر اتفاق بد

3- Architect for Scalability:

بار فعلی سیستم‌های خود را اندازه‌گیری کنید. نوسانات و افزایش‌های شدید را تخمین بزنید و با توجه به این موارد بار وارده را در چند سال آتی برآورد کنید. اینگونه بهتر می‌توانید یک سیستم و معماری طراحی کنید که پایدارتر و کاراتر و مقیاس‌پذیرتر باشد.

4- Architect is leadership:

5- Always be architecting:

6- Build Loosely coupled systems:

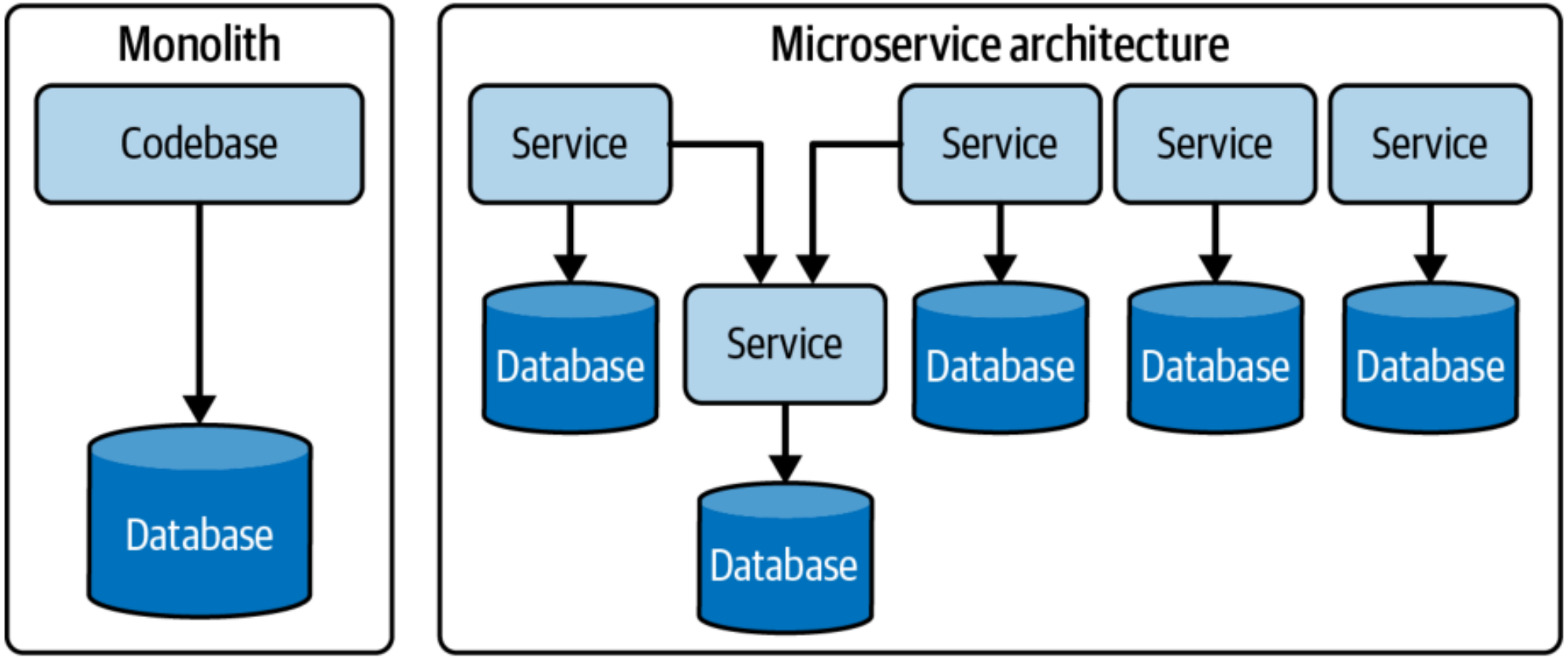
7- Make reversible Decisions:

8- Prioritize Security:

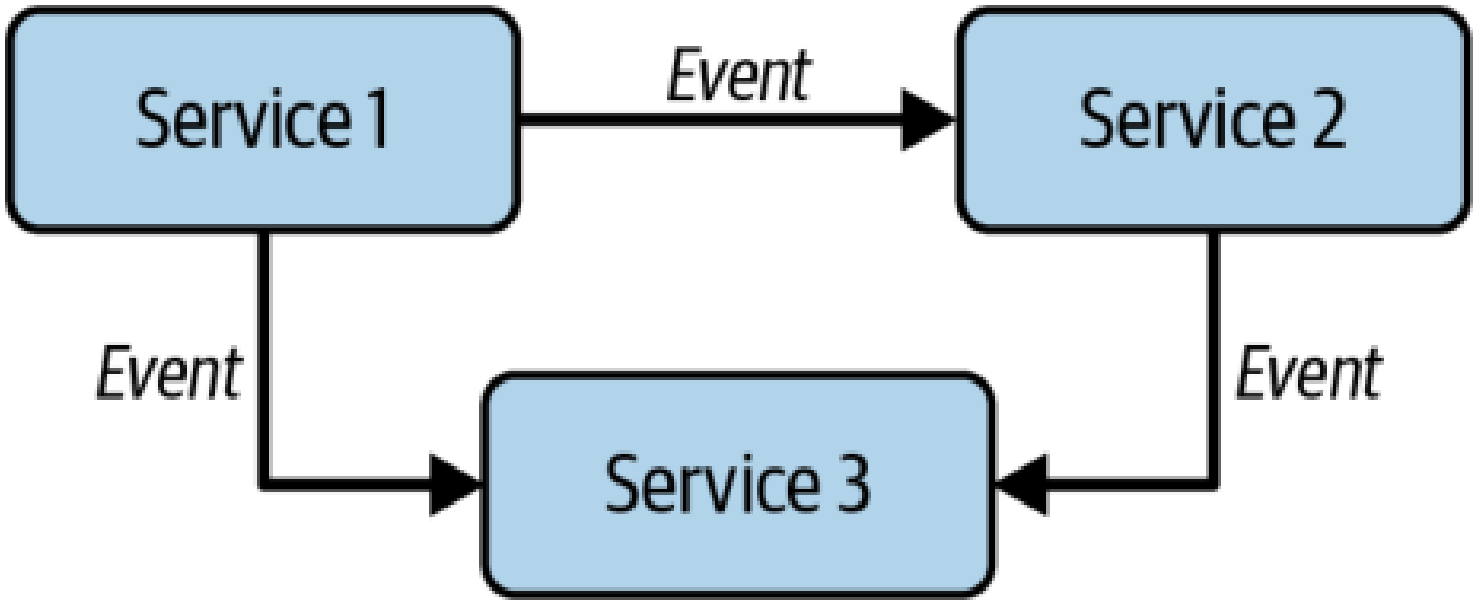
9- Embrace FinOps:

خرج و مخارج کلاد را درست مدیریت کنید.

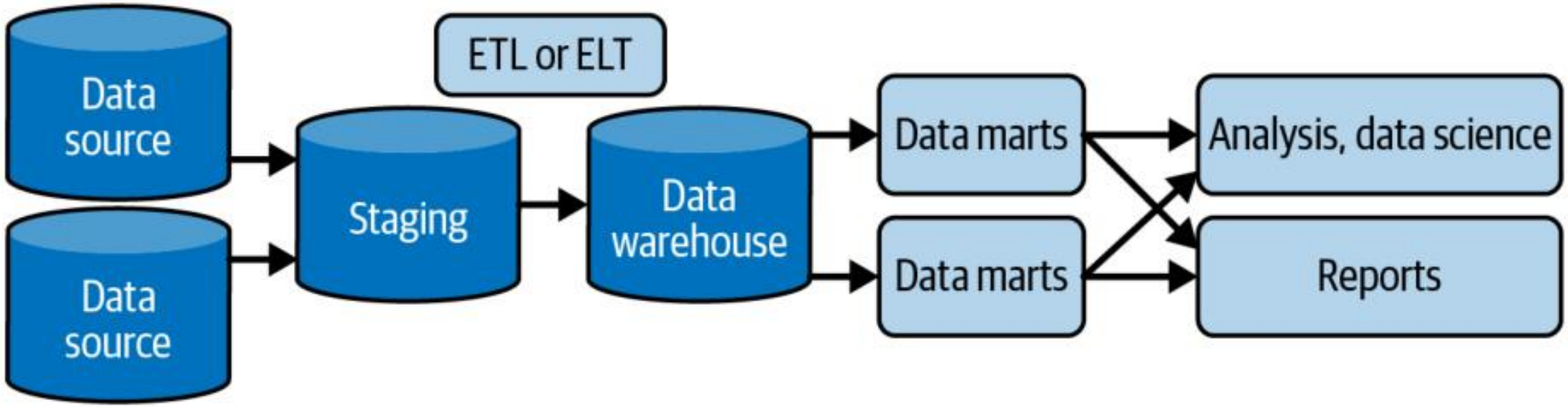
Two Main Architectures:



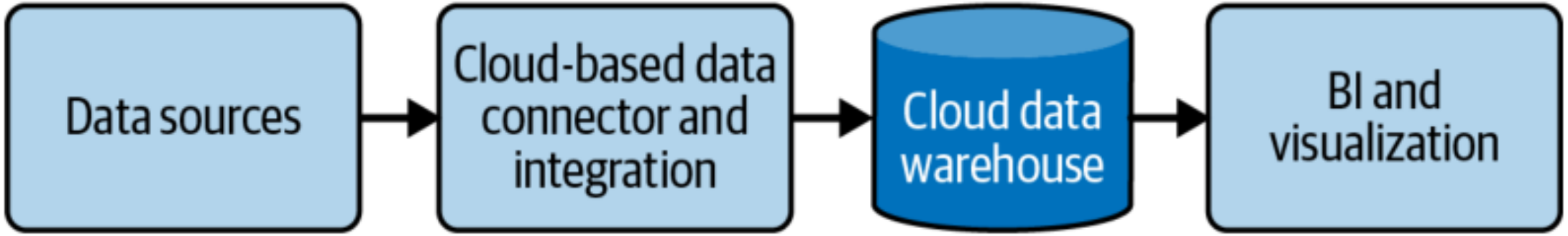
Event-Driven Architecture:



Data warehouse and data marts:



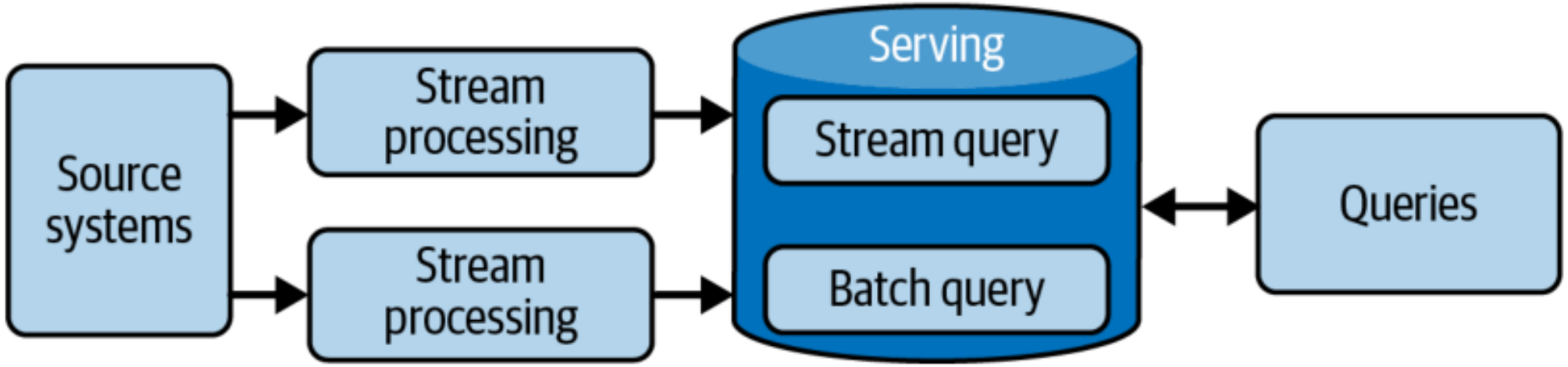
Modern Data Stack:



Lambda Architecture:

در معماری لَمدا، ما هر سه مولفه‌ی بچ، استریم و سرو داده را در کنار هم داریم که بطور مستقل از یکدیگر، کار می‌کنند.

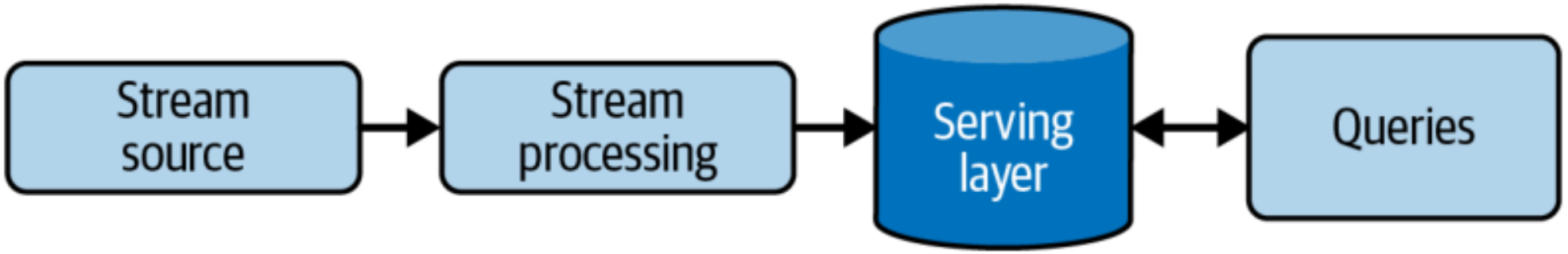
عیب: مدیریت دو سیستم بچ و استریم همزمان سخت است



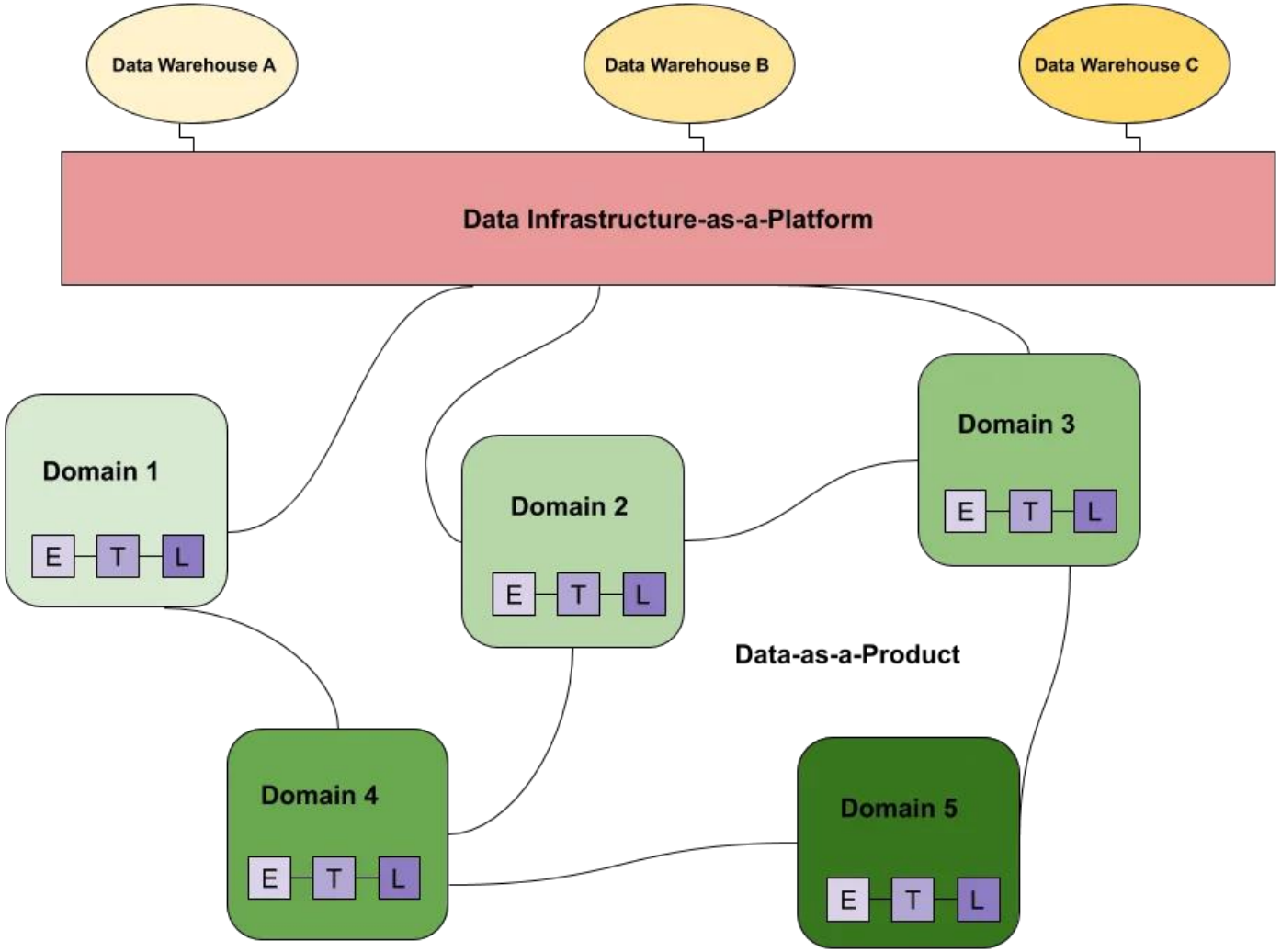
Kappa Architecture:

در معماری کاپا، ما تنها از یک پلت‌فرم پردازش استریم برای هندل کردن تمام کارها استفاده می‌کنیم.

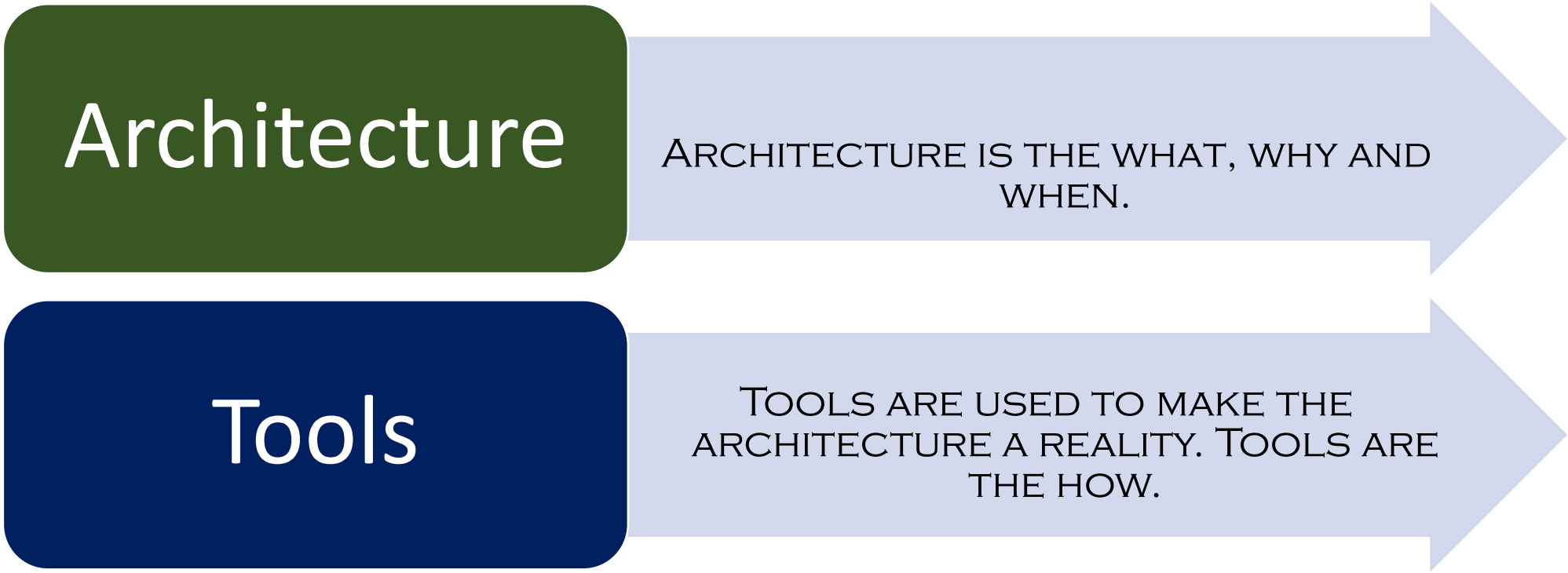
عیب: سخت بودن در عمل و گران بودن در پیاده‌سازی



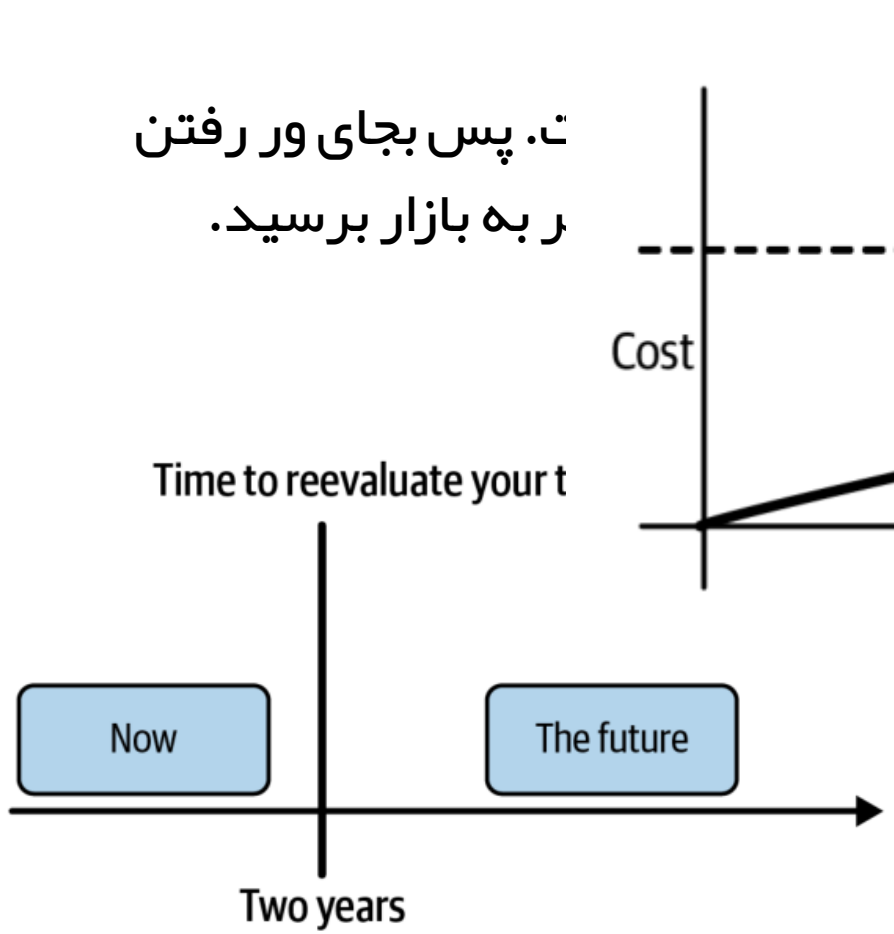
Data Mesh Architecture:



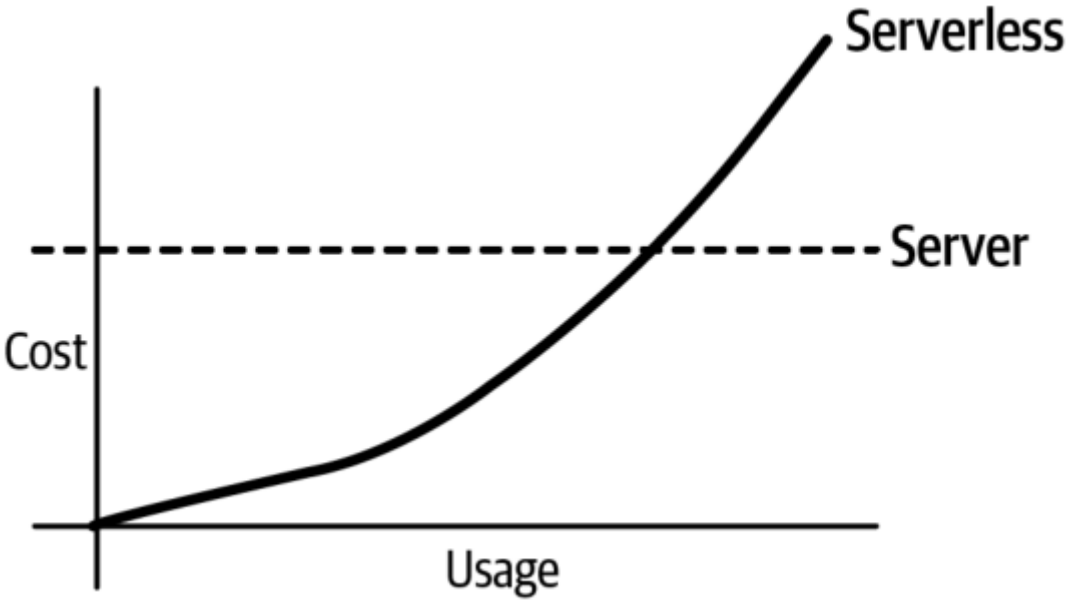
Difference Between Architecture and Tool:



Some pieces of advice:



ت. پس بجای ور رفتن
ر به بازار برسید.



1. تیم‌های کوچک، نبای
2. در دنیای تکنولوژی
   بیهوده با ابزارها، از
3. قیمت مهمترین ملاک
4. هر دو سال یکبار در
5. ساختن و توسعه وق
   محسوب می‌گردد.
6. تکنولوژی‌ها می‌آیند و می‌روند. توان تغییر ابزارها بسیار د
7. استفاده از سرو یا بدون سرور؟! انتخاب کنید.

Some pieces of advice:

1. تیم‌های کوچک، نباید کارهای پیچیده‌ی کمپانی‌های بزرگ را تقلید کنند.
2. در دنیای تکنولوژی سرعت ورود به بازار (Speed to market) همیشه برنده است. پس بجای ور رفتن بیهوده با ابزارها، از همان تکنولوژی‌های روز و مرسوم استفاده کنید تا زودتر به بازار برسید.
3. قیمت مهمترین ملاک برای انتخاب تکنولوژی است.
4. هر دو سال یکبار در استفاده از ابزارهای خود تجدید نظر کنید.
5. ساختن و توسعه وقتی معنی دارد که برای کسب و کار سودمند باشد. در غیراینصورت اتلاف وقت محسوب می‌گردد.
6. تکنولوژی‌ها می‌آیند و می‌روند. توان تغییر ابزارها بسیار مهم هستند.
7. استفاده از سرو یا بدون سرور؟! انتخاب کنید.

(Four + One) places to run our Technologies:

1- On premises

2- Cloud

3- Hybrid Cloud (On prem + cloud)

4- Multicloud (AWS + Azure + GCP +...)

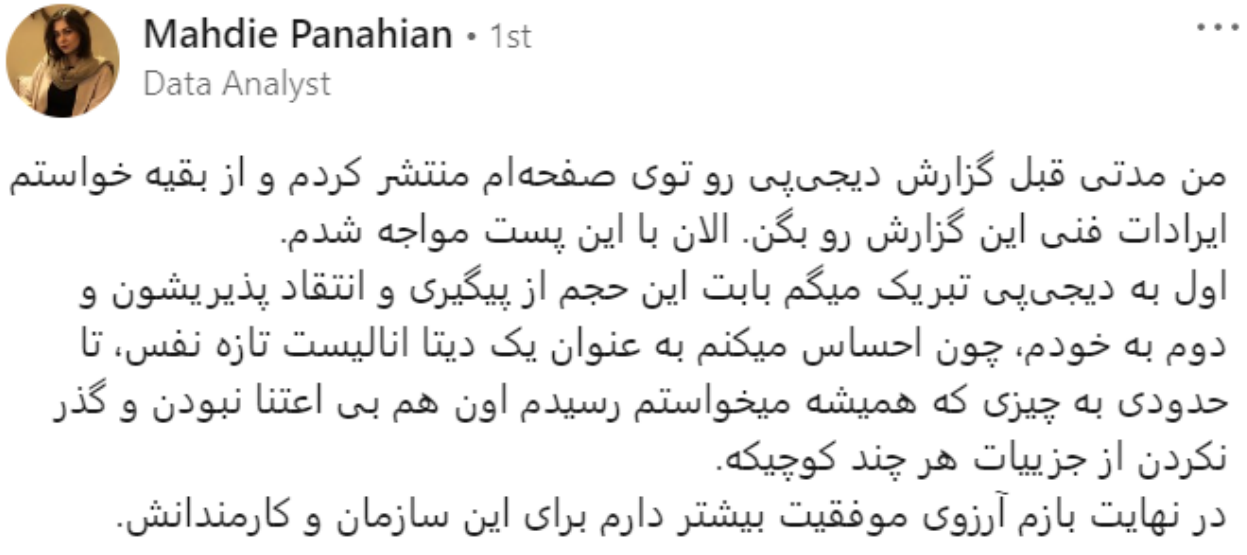
disadvantage: handling different panel is hard!

FIVE- Cloud of Clouds (Snowflake- It offers just one single panel)

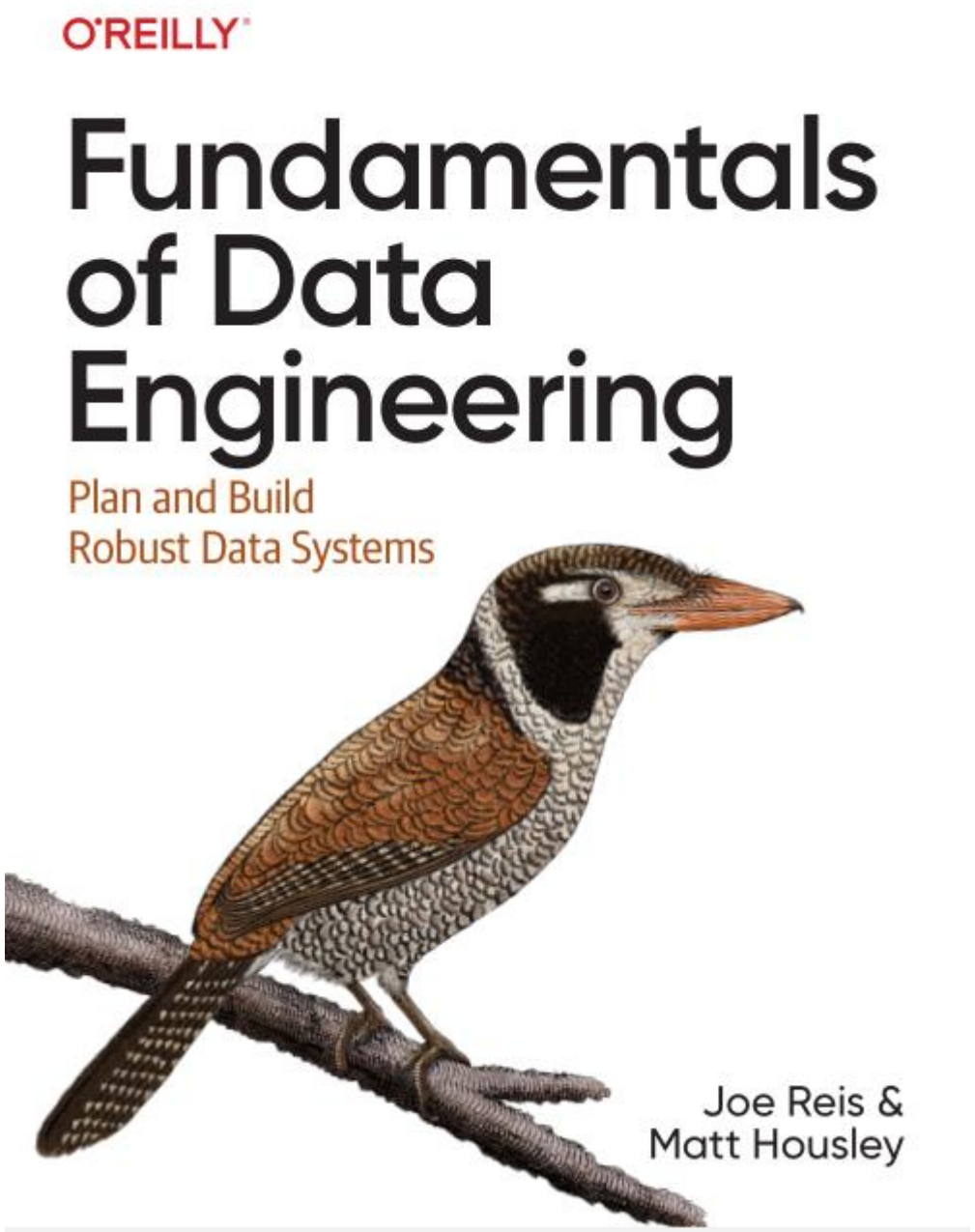
Know three awesome guys in the world of DE and Cloud

1- Joe Reis

2- Stephen Maarak (+2M students in Udemy= 100 Bilion Toman 😯)



1. تخصصی کار کنید
2. کلاد را جدی بگیرید
3. جاوا یاد بگیرید
4. از کار بزرگان ایراد بگیرد و نظر دهید



This talk is brought to you using the first four chapters (out of eleven) of this awesome Book.