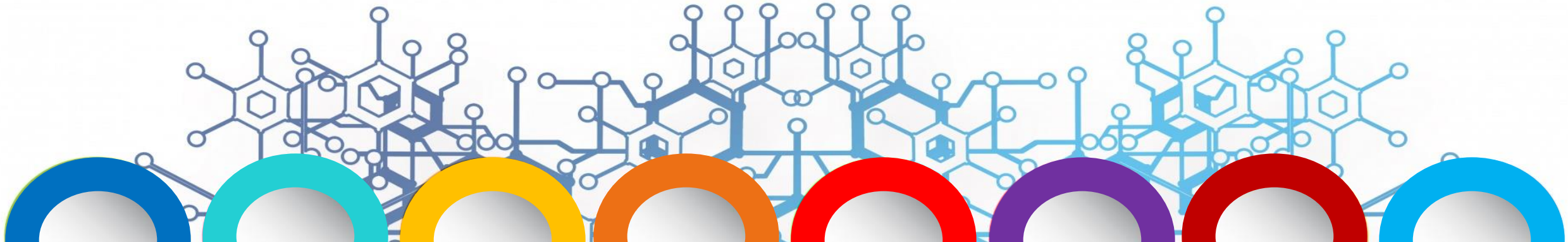
Popular Applications



Applications & Architectures

A sample set of cloud applications

- Resource usage model
- Challenges and opportunities while moving towards serverless cloud
- System architecture, programming model, design and deployment framework

Application	*Description*
Feature-rich Scalable Web Application	***Microservices Architecture***
Real-Time Video	*On-the-fly video encoding*
Unstructured Data Processing	***Big data processing on large clusters***
Linear Algebra	*Large scale linear algebra*
ML Pipelines	***ML Training at scale***
Databases	*OLTP*

From [edited]: *Cloud Programming Simplified: A Berkeley View on Serverless Computing (2019)*

Application Architectures

- Defines the relation of different components of the system (application, middleware, data stores) that work together for an application

[Traditional]
- N-Tier

[Popular]
- Microservices
- Big Compute
- Big Data
- Event-Driven
- Web-Queue-Worker

Relation?

Event-Bus Broker Blackboard
Primary-Secondary Interpreter
MVVM Monolithic Pipe-Filter
Client-Server Peer to Peer
Model-View-ViewModel
Layered Model-View-Controller

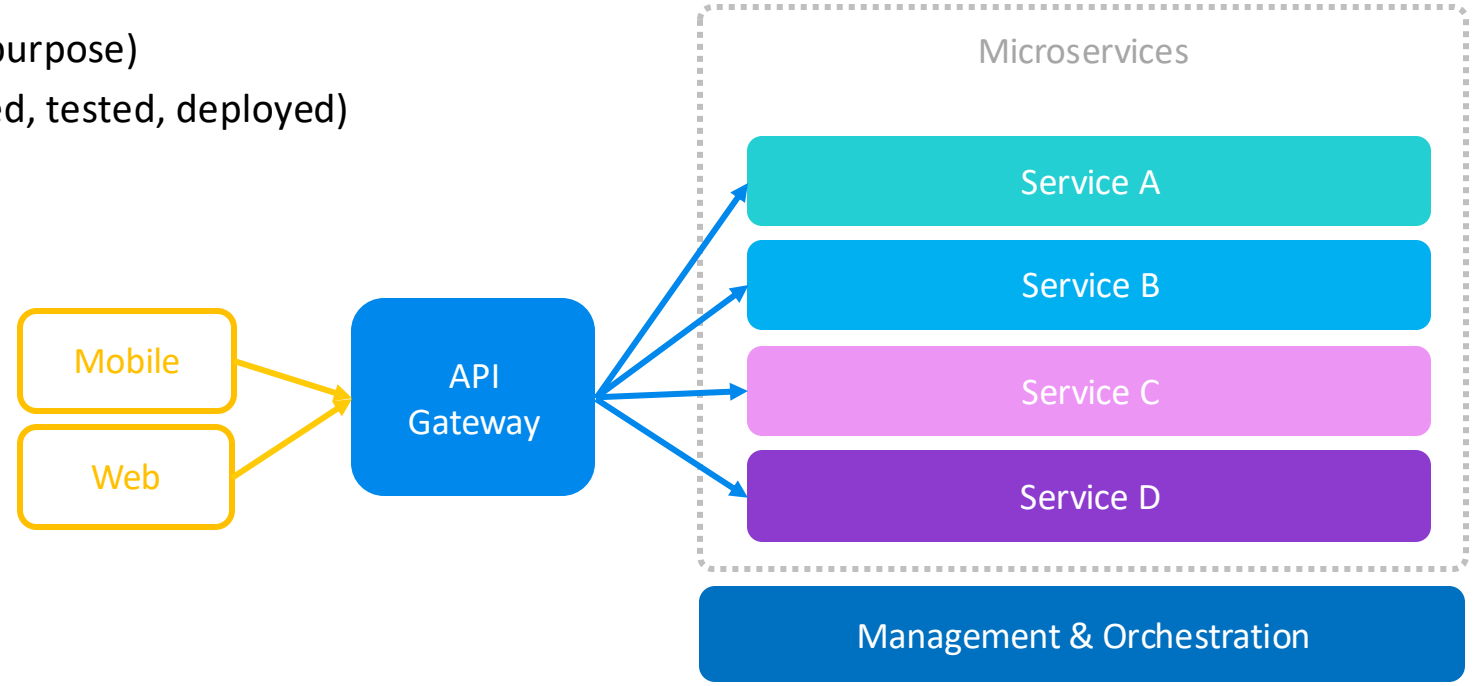
*Different architectures could be implemented using different **frameworks** and **technologies***

Refernece: *https://learn.microsoft.com/en-us/azure/architecture/guide/architecture-styles/ (Page Date: 13/6/2024, Last Accessed: Feb 2025)*

Microservices

- Context: Server-side enterprise application

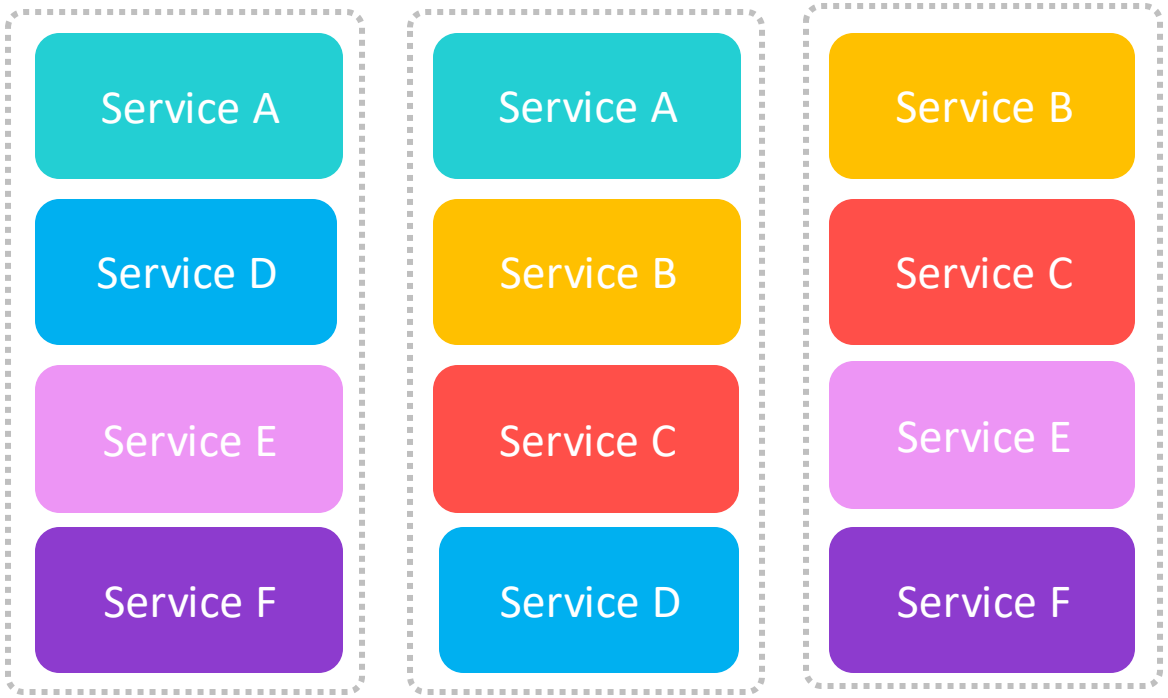
- Characteristics
  - Small (Cohesion around business purpose)
  - Autonomous (Separately developed, tested, deployed)
  - Single responsibility
  - Independent of other services



Reference: https://docs.microsoft.com/en-us/azure/architecture/

Microservices

- Each service is a separate codebase
- Not shared
  - Technology stack
  - Libraries
  - Frameworks
- Hide details from other services
- Services can be deployed independently
  - No application rebuilt



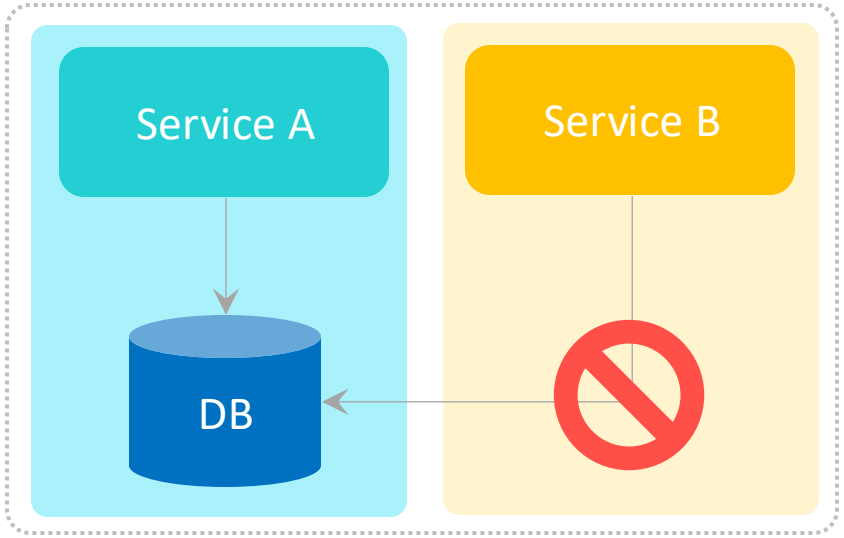
Coupling

- Domain Coupling
  - One Microservice the user of the functionality of the other Microservice
- Pass through Coupling
  - One microservice passes data to another microservice because it is needed by some other further downstream microservice
- Common Coupling
  - Two or more microservices make use of a common set of data
- Content Coupling
  - An upstream service reaches into the internals of a downstream service and changes its internal state
- Temporal Coupling
  - One microservice needs to call another microservice in a synchronous way

Microservices

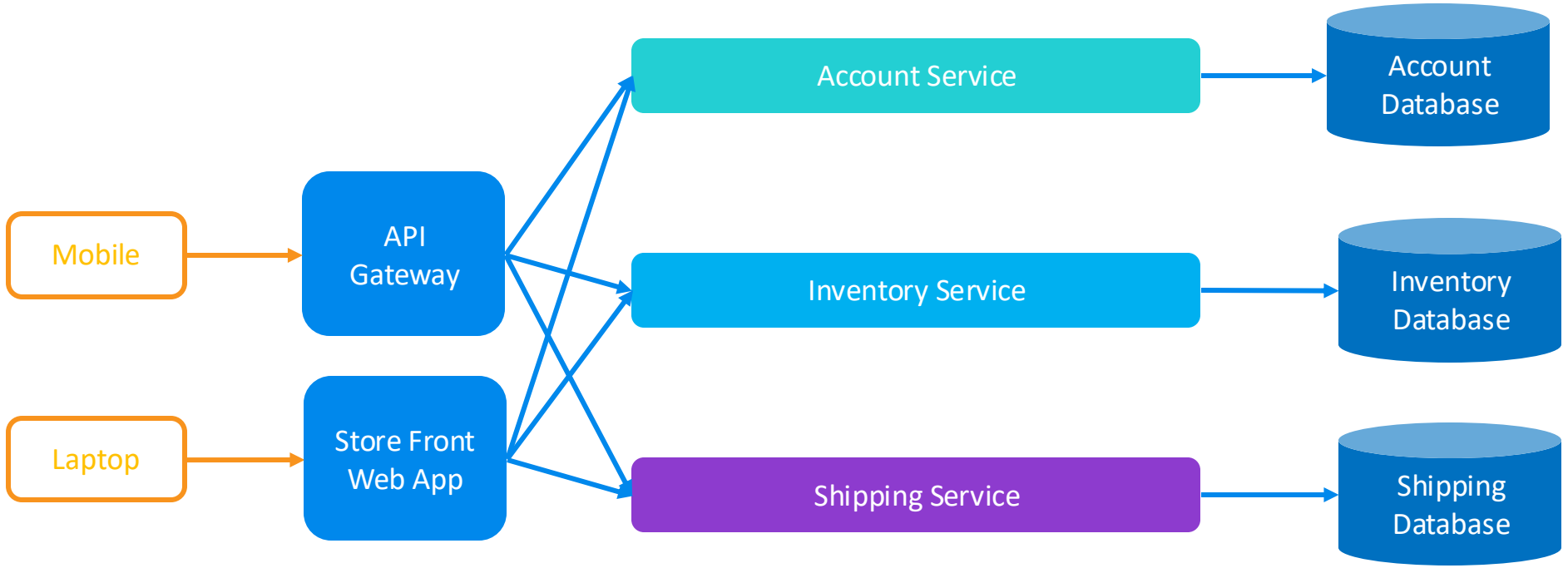
- Services are responsible for persisting their own data or external state
  - Data is private to the service that owns it
  - Services do not share data
  - No data persistence layer
  - Leads to polyglot persistence

- Challenges
  - Redundancy
    - Same data in multiple places
  - Tracking Updates (Eventual Consistency)
    - Use events (aggregation & batching)
    - What if strong consistency needed? One service source



Reference: https://docs.microsoft.com/en-us/azure/architecture/

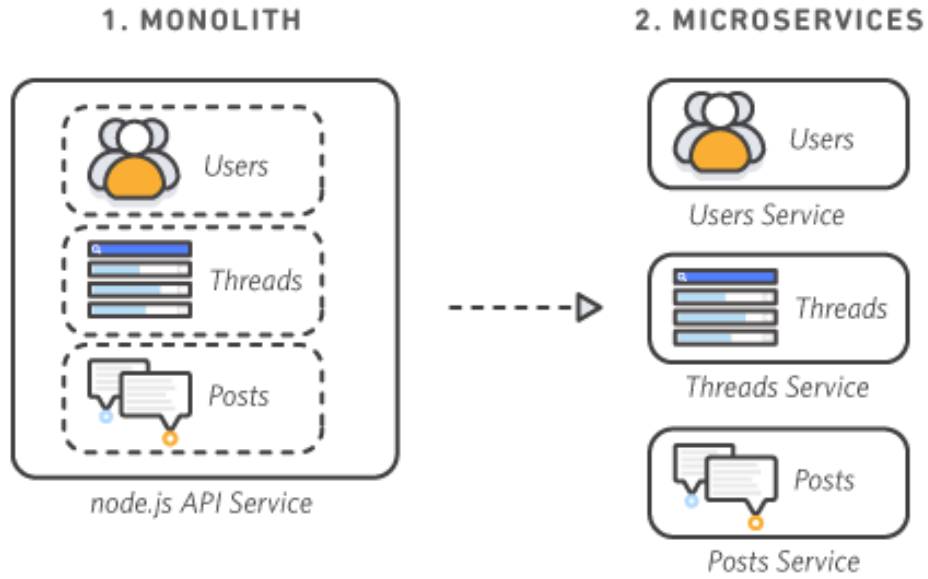
Microservices: Example



Reference for the example E-Commerce Application: https://www.atatus.com/blog/microservices-architecture/ (Last accessed: Feb 2024)

Examples

- Well-Known Live Services
  - Amazon
  - Netflix
  - eBay
  - LinkedIn
  - Uber



Code examples

Azure Drone Delivery Example
https://github.com/mspnp/microservices-reference-implementation

Oracle microservices deployment K8 example
https://docs.oracle.com/en/solutions/deploy-microservices/index.html
https://github.com/oracle-devrel/terraform-oci-arch-microservice-oke

Additional Examples
https://eventuate.io/exampleapps.html

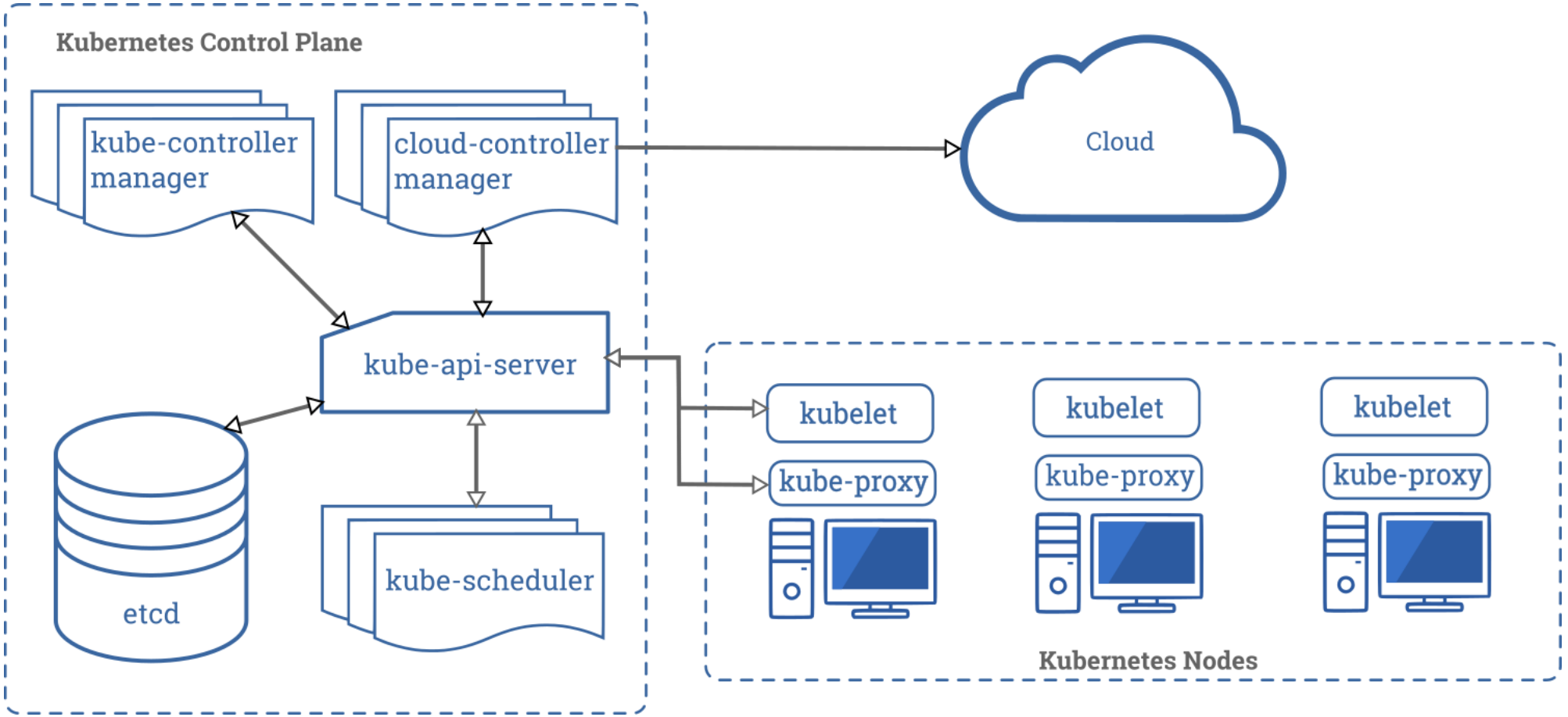
Picture from Amazon: https://aws.amazon.com/getting-started/hands-on/break-monolith-app-microservices-ecs-docker-ec2/ (Accessed: Feb 2023)

Kubernetes (Greek word meaning helmsman or pilot)

- Open-source platform for managing containerized workloads
- Kubernetes Components
  - Master (Control Plane)
    - API Server
    - etcd: Consistent and highly-available key value store
    - Kube scheduler: Pod to node assignment
    - Kube-controller manager: Control Plane component that runs controller processes
    - Cloud-controller manager: Runs controllers that interact with the underlying cloud providers
  - Nodes (Minions): A worker machine (May be a VM or physical machine, depending on the cluster)
    - Container (Most often Docker)
    - Kubelet: An agent that runs on each node in the cluster and makes sure that containers are running in a pod
    - Kube-proxy: A network proxy that runs on each node in your cluster
  - Workloads
    - Pod: Basic execution unit of a Kubernetes application
    - Controllers: Controller can create and manage multiple Pods

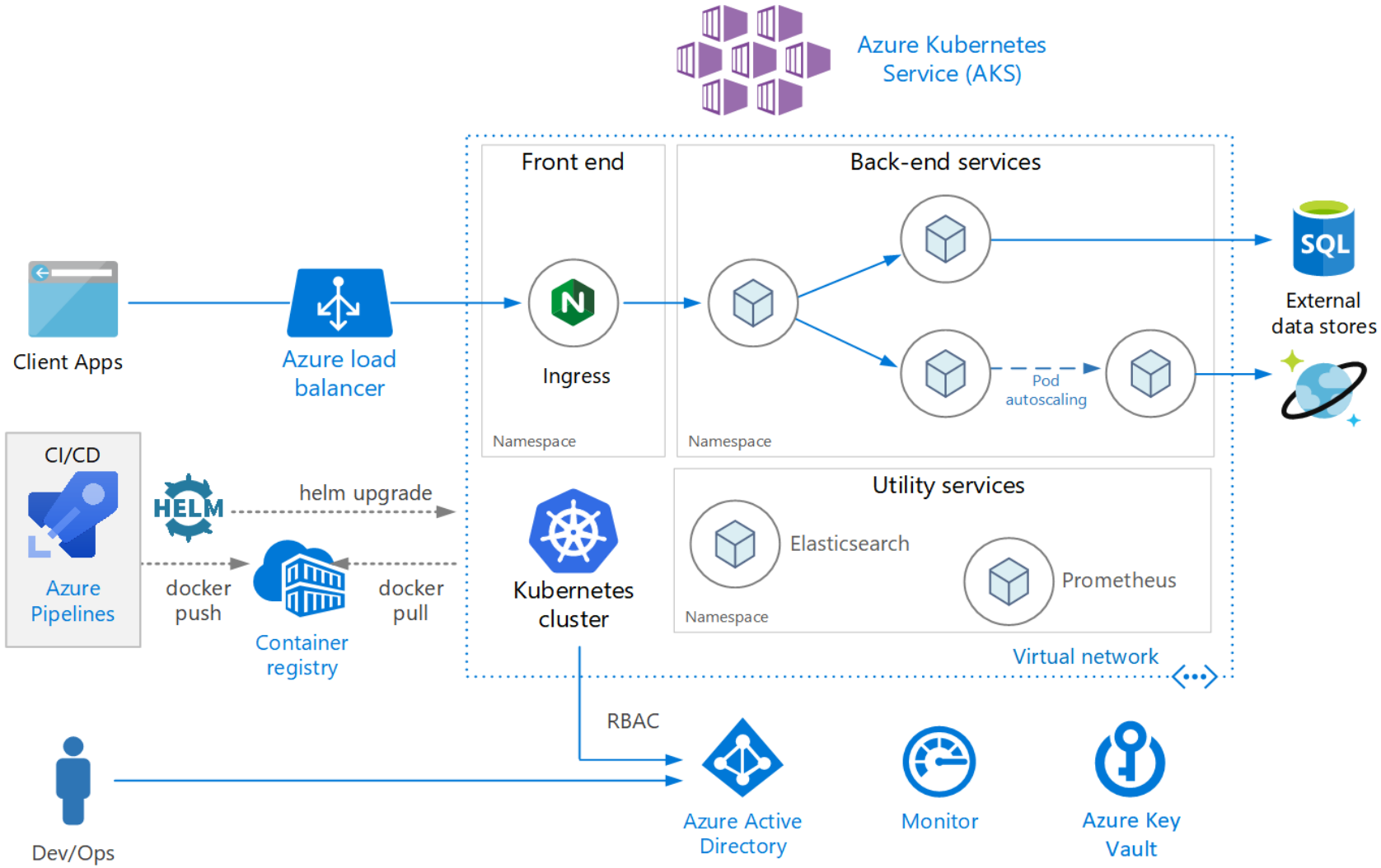
Reference: *https://kubernetes.io/docs/*

Deployment: Kubernetes



Reference: https://kubernetes.io/docs/

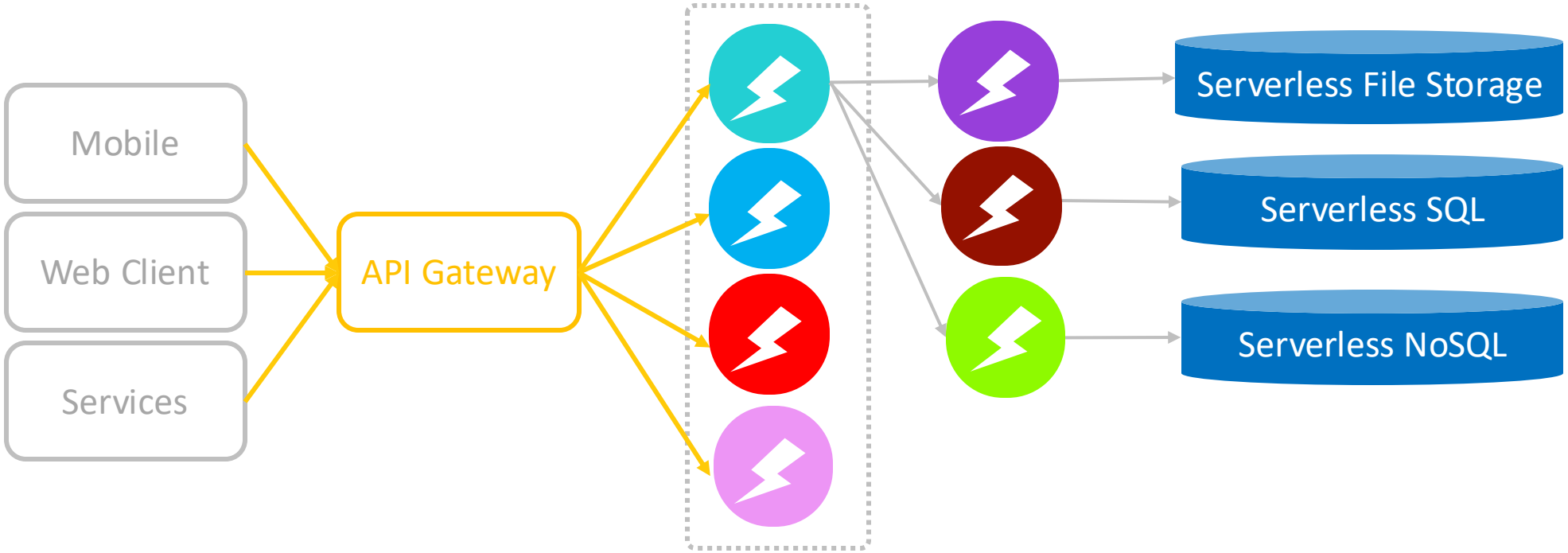
Deployment: K8



Reference: https://docs.microsoft.com/

Code: https://github.com/mspnp/microservices-reference-implementation

Serverless


Mobile
Web Client
Services
API Gateway
Serverless File Storage
Serverless SQL
Serverless NoSQL

Knative

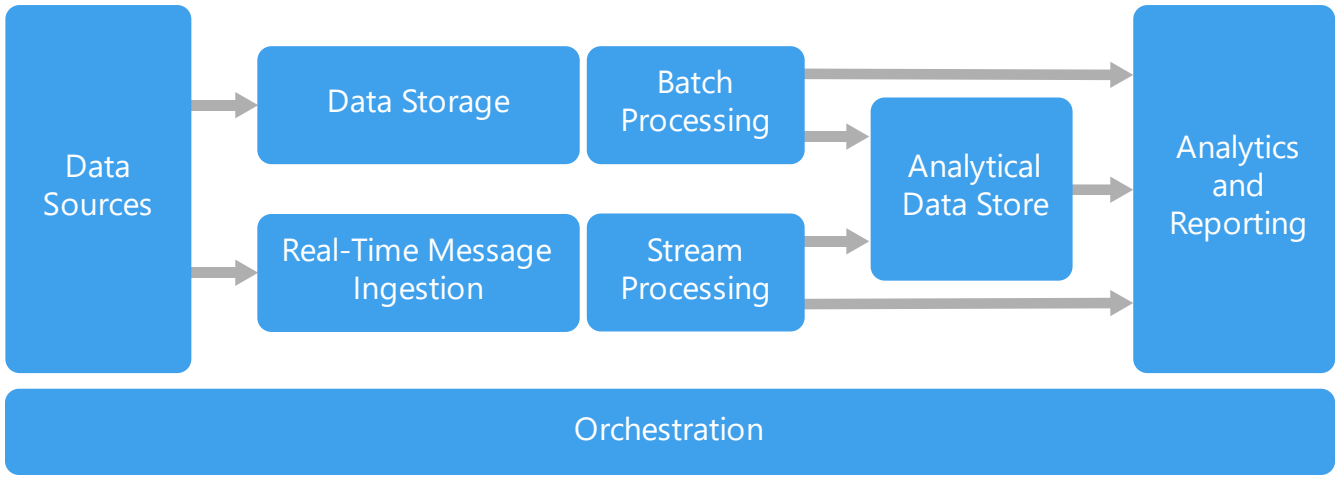
- Kubernetes Extension with set of middleware components
- Kubernetes-native APIs for deploying serverless-style functions
- Components
  - Eventing: Management and delivery of events
  - Serving: Request-driven compute that can scale to zero

Reference: https://knative.dev/

Data Processing on the Cloud

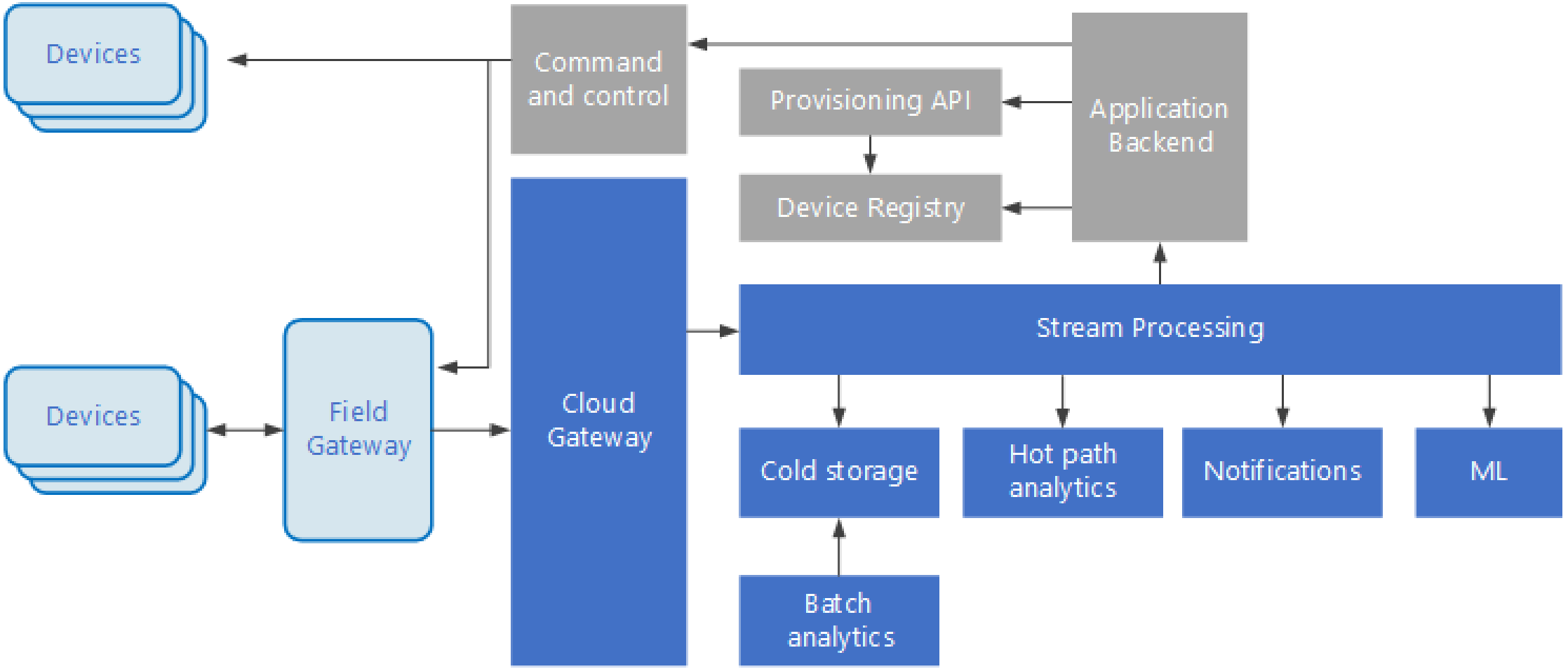
- Data Infrastructures
  - Batch Processing
  - Streaming Data
  - Data Integration
  - Cloud Native Platforms

- Architecture elements
  - Ingestion
  - Transformation
  - Storage



Reference: IBM Documentation, ***Figure from:*** https://docs.microsoft.com/en-us/azure/architecture/

Example: IoT Applications



Reference & Figure: https://docs.microsoft.com/en-us/azure/architecture/

Machine Learning Pipelines

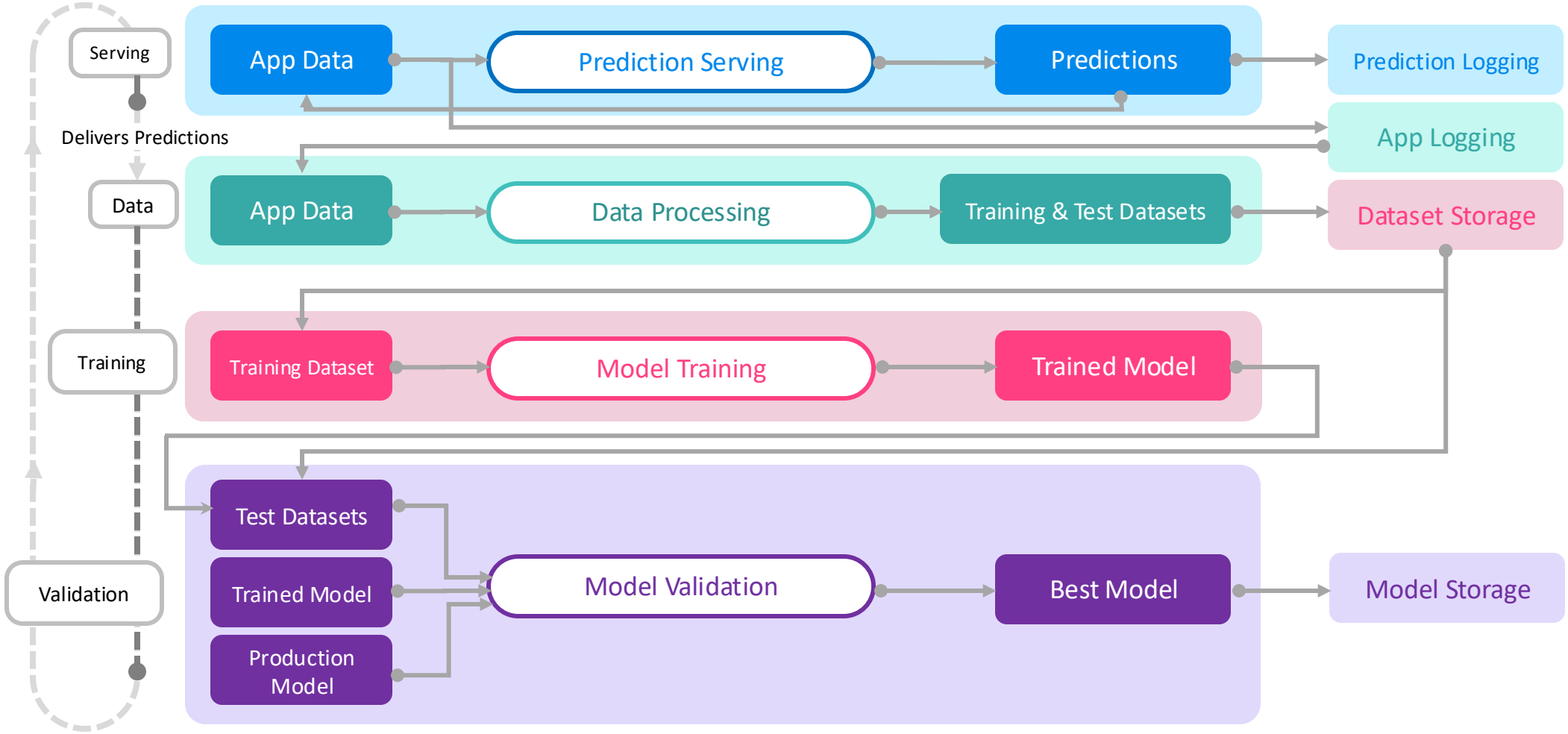
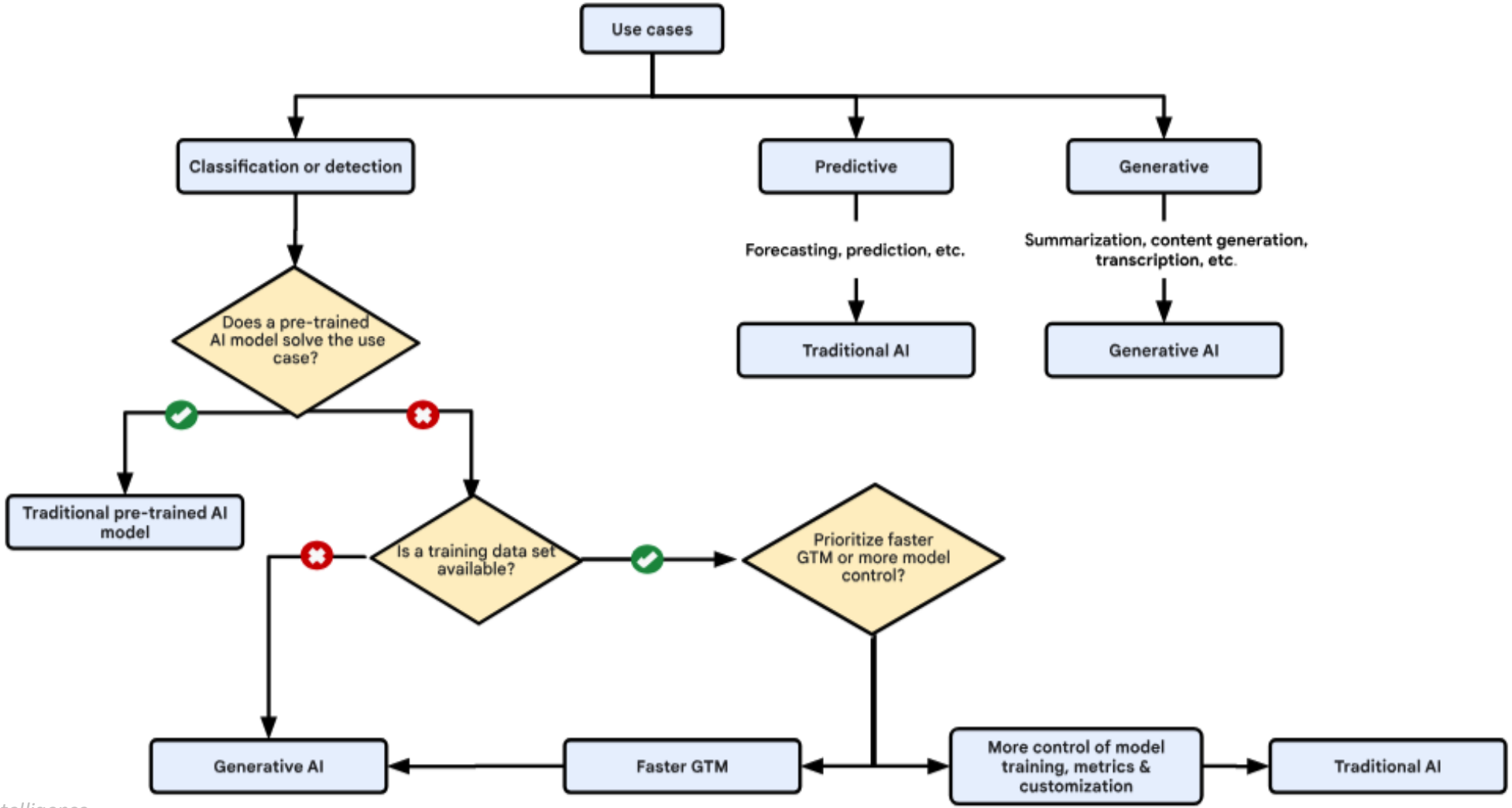


Figure [edited] From: Google Cloud Documentation: https://developers.google.com/machine-learning/managing-ml-projects/pipelines (Last Accessed: Feb 2025)

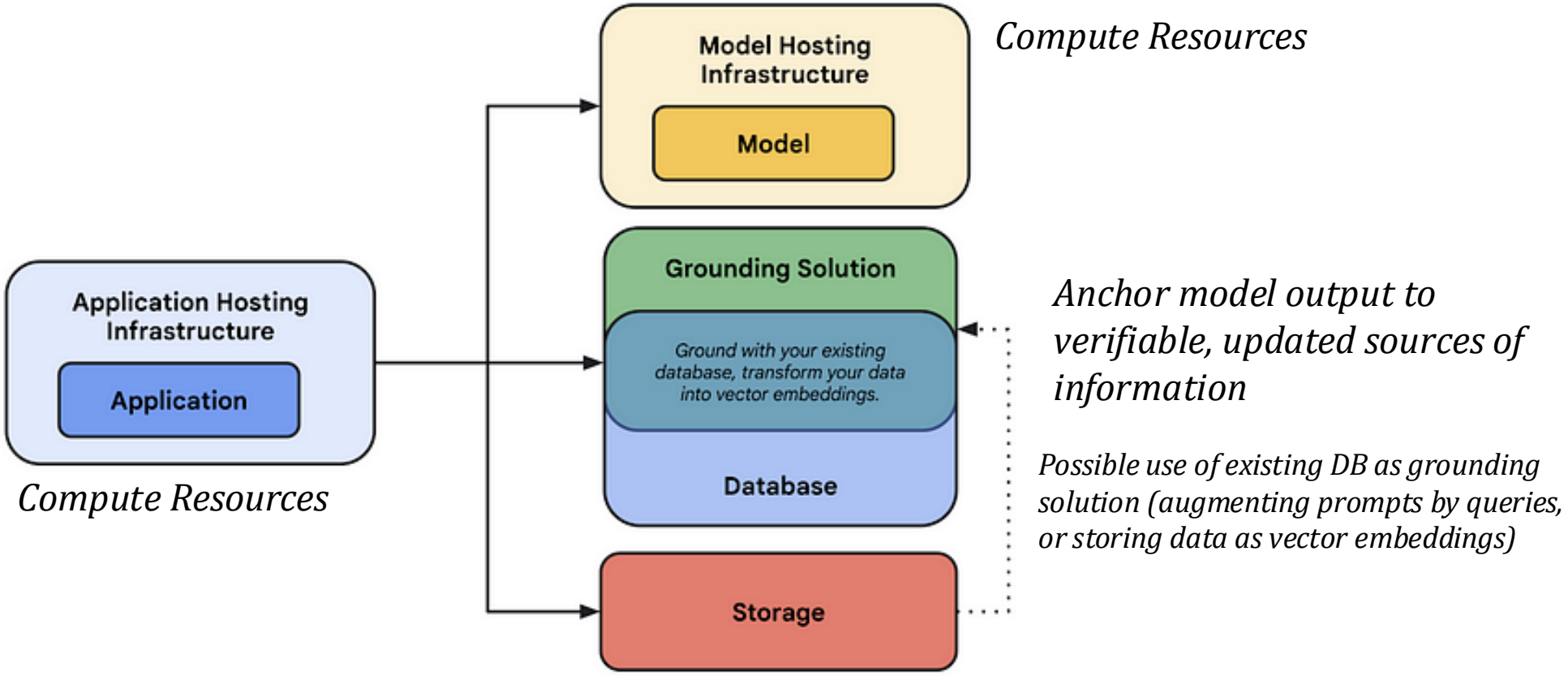
Traditional or Generative



AI: *Artificial Intelligence*

Figure From: *Google Cloud Documentation & https://cloud.google.com/docs/ai-ml/generative-ai/generative-ai-or-traditional-ai (accessed Feb 2025)*

Architecting Applications



Architecting Generative Artificial Intelligence (GenAI) Applications

AI: *Artificial Intelligence,* ***GenAI:*** *Generative Artificial Intelligence*

Figure From: *Google Cloud Documentation & https://medium.com/google-cloud/architecting-genai-applications-with-google-cloud-b38c9cbc66e0 (accessed Feb 2025)*

Summary

- Applications we build using the cloud
  - Popular Use Cases

- What's next?
  - Programming Models & Frameworks
    - Cloud Enablers
      - Resources abstraction
      - Programming abstraction
      - Scalability
        - Parallelization
        - Resource sharing
        - Instance selection
        - Coordination
        - Load-balancing
      - Fault-tolerance

Acknowledgement

The list of resources used in preparation of this slide set are provided on:

https://canvas.sfu.ca/courses/88212/pages/references

Pictures and quoted resources are mentioned in each use.