System Design Fundamentals

Software Systems Development

IIIT Hyderabad

Content

Introduction

Building blocks

Horizontal vs Vertical scaling

Monolithic vs Microservices architecture

Containerization using Docker

Deployment Scenarios

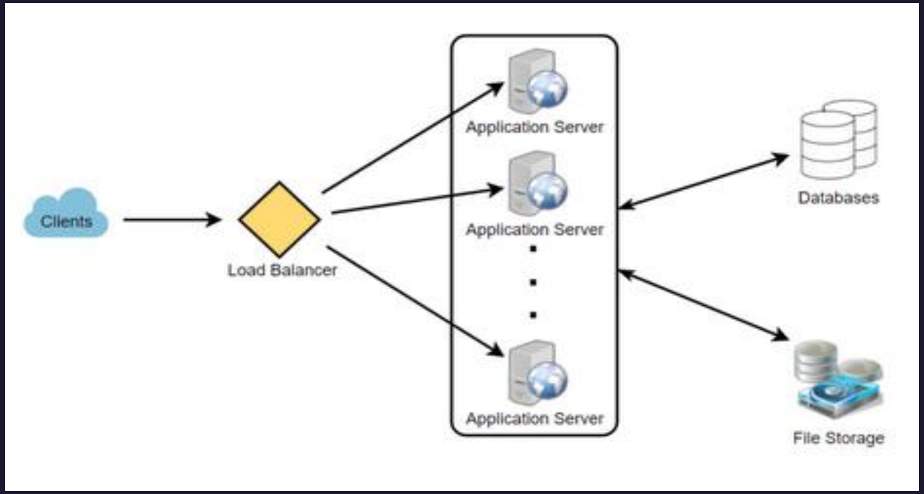
Proxy Server

Redundancy and Replication

Cloud computing services categories

Some Examples of System Designs

What is System Design?



- The **process of defining the architecture, interfaces, and data for a system** that satisfies specific requirements.

Gather business needs or requirements (Functional and non-functional) → Estimation of important parts (storage, bandwidth etc.) → Data Flow → Prepare detailed architecture design → Identify and resolve bottlenecks

- Design Methods :

Architectural Design

Views

Models

Behaviour

Infrastructure

Logic Design

Data flow

Input/output of system

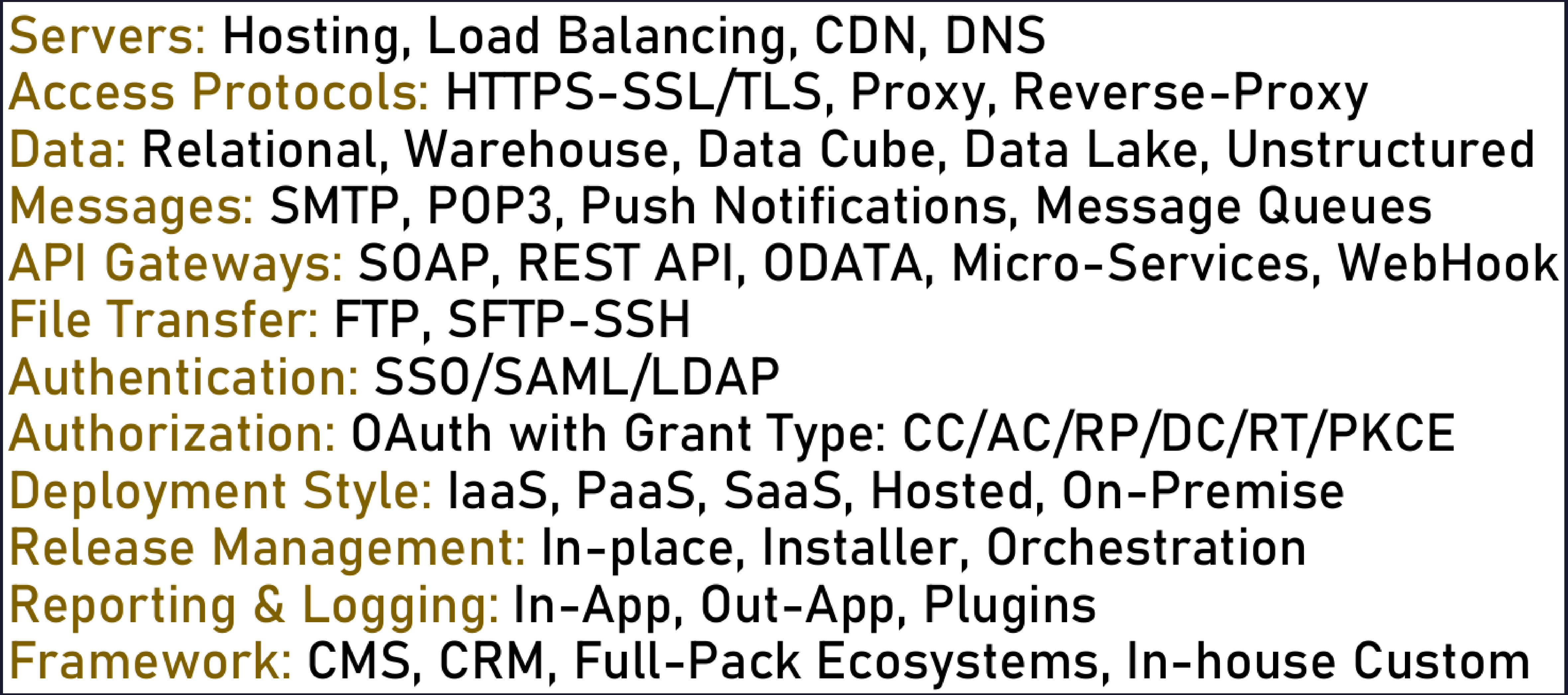
Physical Design

How User add information

How System shows information to users

How data is stored

Building blocks of a software system

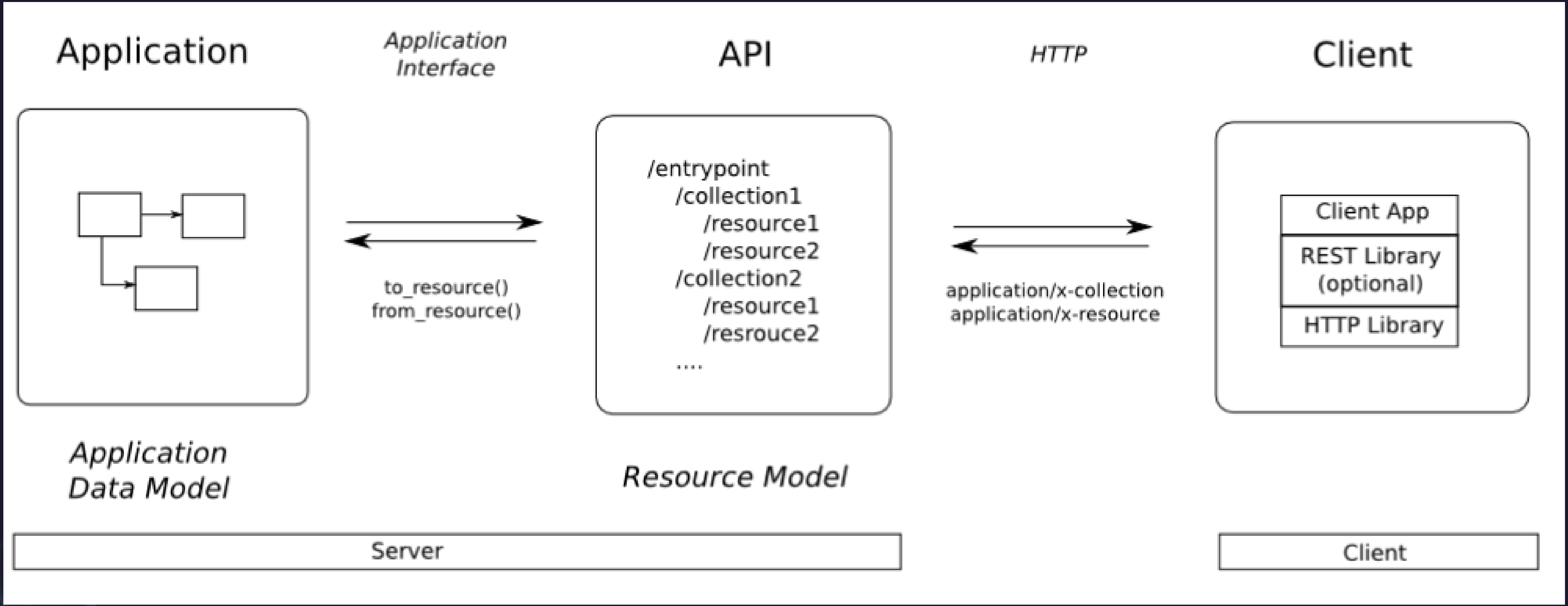


Servers: Hosting, Load Balancing, CDN, DNS
Access Protocols: HTTPS-SSL/TLS, Proxy, Reverse-Proxy
Data: Relational, Warehouse, Data Cube, Data Lake, Unstructured
Messages: SMTP, POP3, Push Notifications, Message Queues
API Gateways: SOAP, REST API, ODATA, Micro-Services, WebHook
File Transfer: FTP, SFTP-SSH
Authentication: SSO/SAML/LDAP
Authorization: OAuth with Grant Type: CC/AC/RP/DC/RT/PKCE
Deployment Style: IaaS, PaaS, SaaS, Hosted, On-Premise
Release Management: In-place, Installer, Orchestration
Reporting & Logging: In-App, Out-App, Plugins
Framework: CMS, CRM, Full-Pack Ecosystems, In-house Custom

Common terms

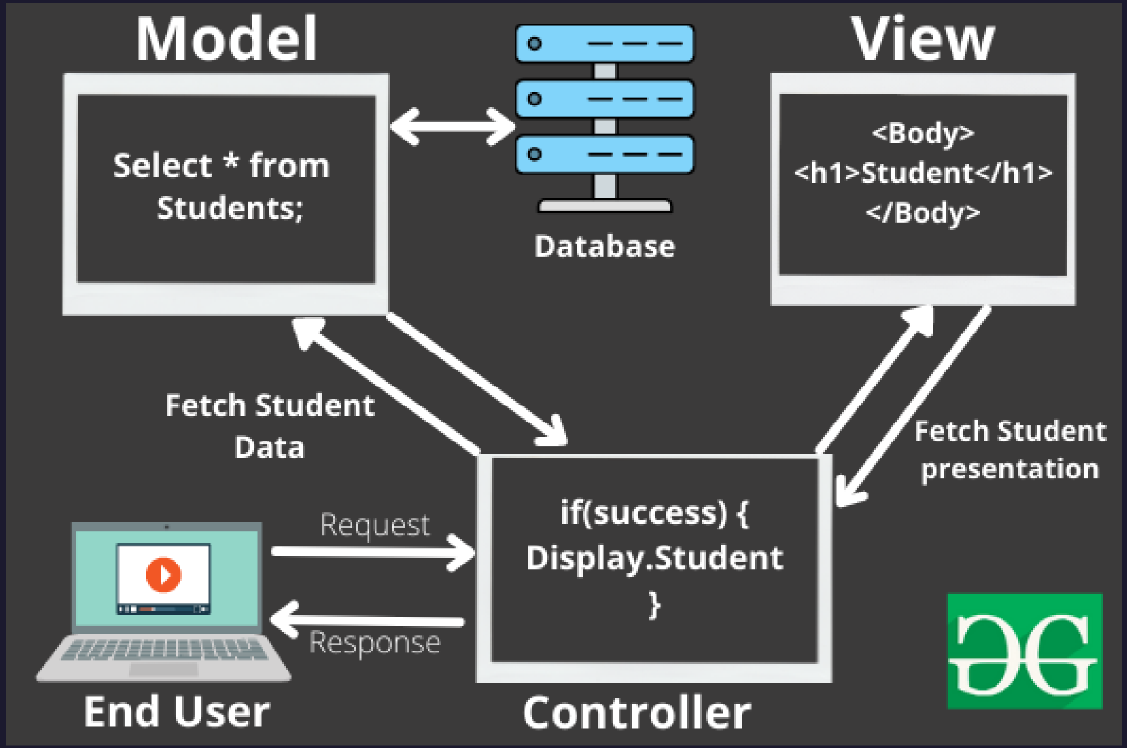
- **Scalability** : property of a system to handle a growing amount of work. An example is a search engine, which must support increasing numbers of users, and the number of topics it indexes.
- **Reliability** : probability of failure-free operation of a computer program in a specified environment for a specified time.
- **Security** : refers to a set of practices that help protect software applications and digital solutions from attackers.
- **Performance** : availability, end-user experience, resource utilization, reliability, and responsiveness of your software application.
- **Consistency** : provides context that is understandable for most of us, so we can transfer our knowledge from one product we use to another.
- **Service vs product** : Service is more oriented towards a particular client needs whereas product is targeted for general users.

Client Server Model

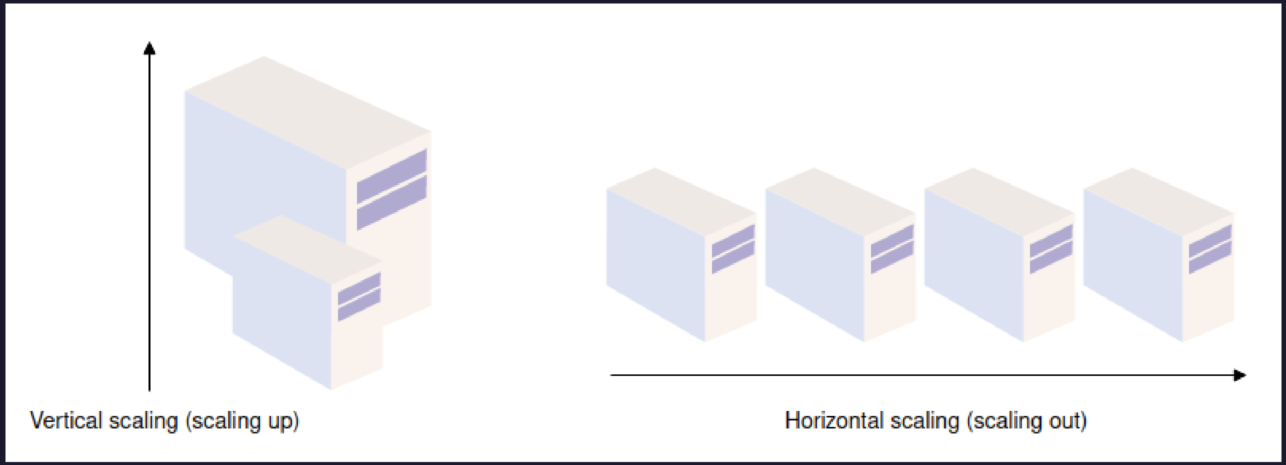


Model View Controller (MVC) framework

- The **Model-View-Controller (MVC)** framework is an architectural/design pattern that separates an application into three main logical components **Model**, **View**, and **Controller.**



Horizontal Vs Vertical Scaling


Vertical scaling (scaling up)
Horizontal scaling (scaling out)

Horizontal

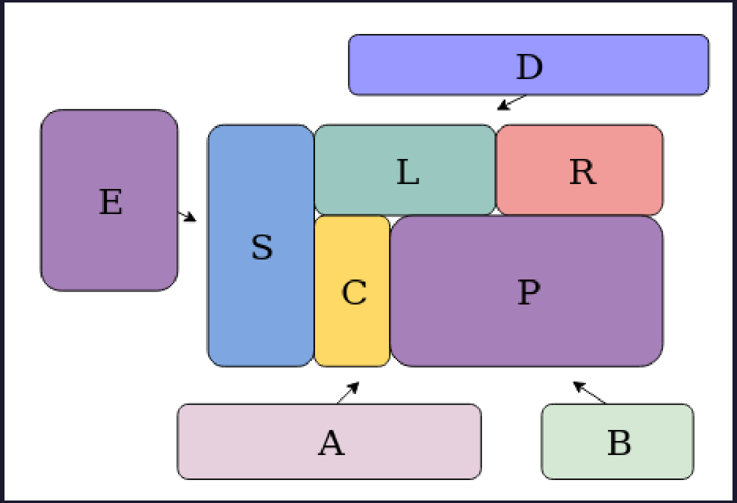
- add more machines in parallel to deal with the increasing requirements.
- need load balancing to distribute the load across the system.
- Data inconsistency is a drawback.

Vertical

- uses one huge machine that handles all requests and improves response time and throughput.
- no load balancing.
- single point of failure.
- Hardware limitations.

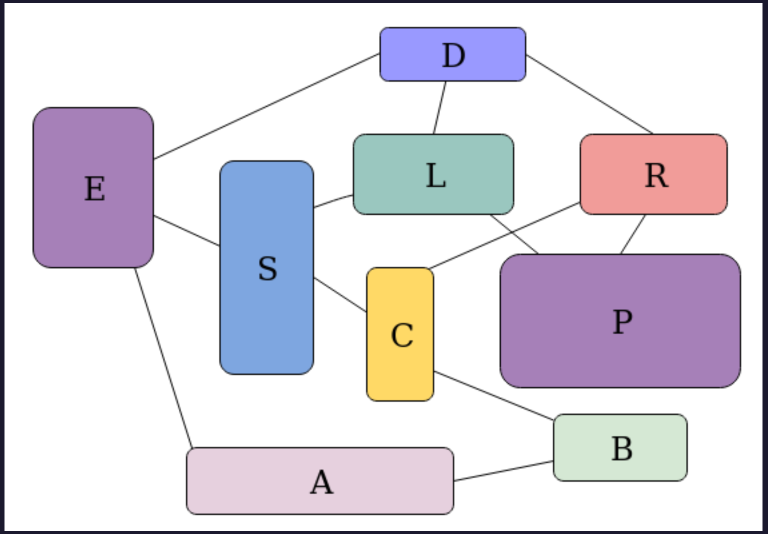
Monolithic Architecture

- In the 1990s, internet companies would develop server-side systems with a single code base and application.
- For ex. online store has a series of features: searches, likes, ratings, etc.
- As the application expands, more features are “packaged” into a single application.
- Challenges :
  - **Dependence** : components share libraries, change in one affects other and the risk of bugs increases.
  - **Deployment** : If a feature ever stops working, the system must be shut down and re-deployed consuming extensive resources and time.
  - **Framework or language** : difficult to migrate technologies.
- Preferred for small team size and for quicker launch.



Microservices Architecture

- It is an architectural style where services are split up so that they are self-contained and self-deployable.
- Each service (likes, search, ratings, catalog, purchase, etc.) is packaged in its own container and communicates with other applications through APIs.
- Benefits :
    - Improves Scalability and Productivity
    - Integrates well with legacy systems
    - Sustainable development
    - Cross-functionality
- Drawbacks :
    - Deployment requires more effort
    - Testing must be independent
- Preferred for more scalable application and when company is large and plans to grow.



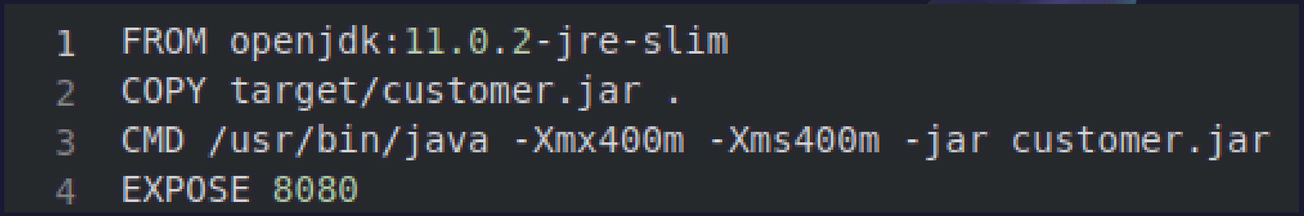
Docker

- Docker is a lightweight solution to deploying microservices. A microservice can be packed into a **Docker image** and isolated as a **Docker container**. This way, you can build an application that is independent of your host environment.

- Docker containers share the kernel of the operating system on the Docker host.

- They don't contain a guest OS for each container and rely on the underlying OS kernel, which makes the containers lightweight.

- To use Docker with microservices, you need to create Docker images via files named *Dockerfile*.



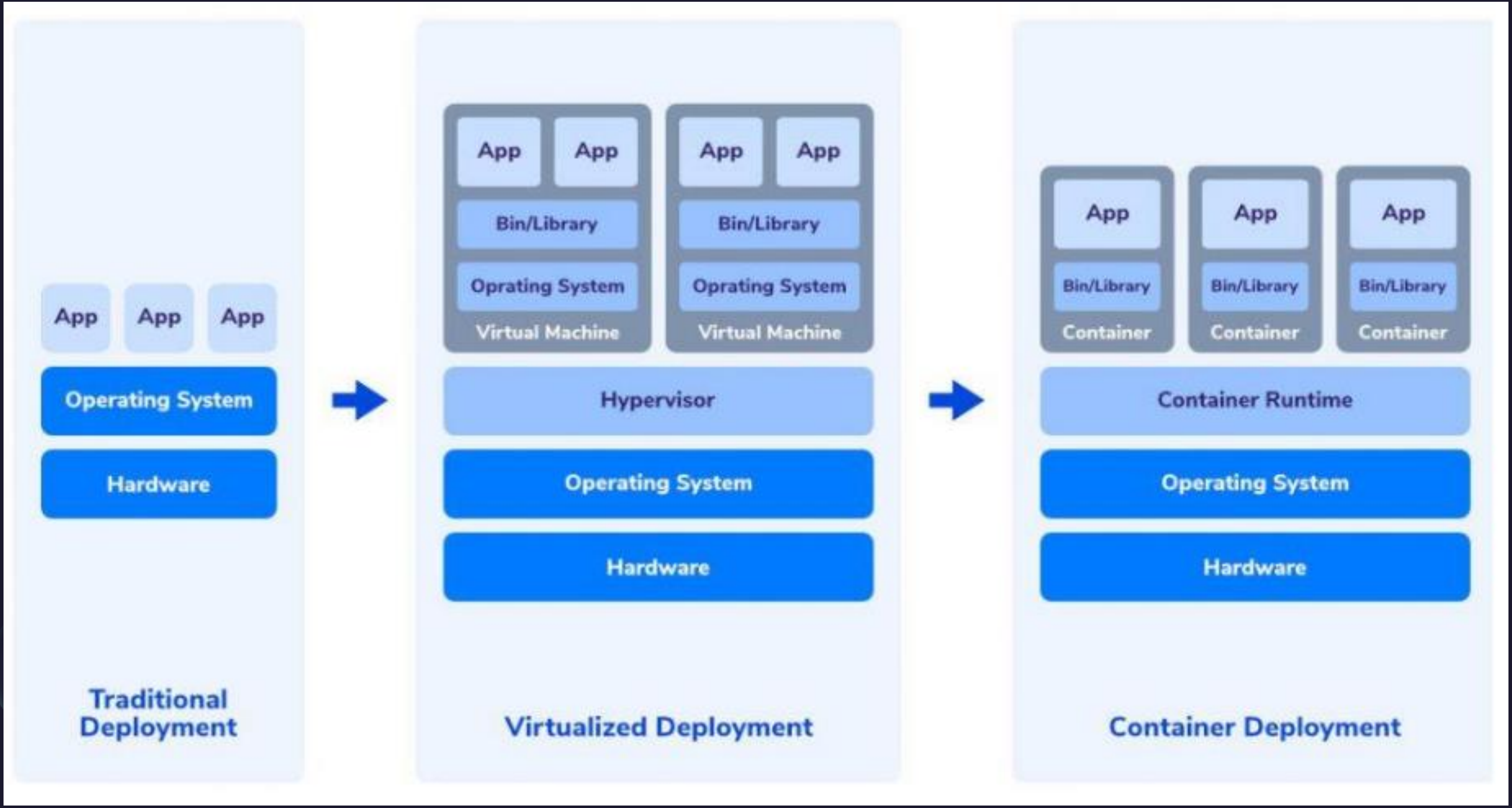
```
FROM openjdk:11.0.2-jre-slim
COPY target/customer.jar .
CMD /usr/bin/java -Xmx400m -Xms400m -jar customer.jar
EXPOSE 8080
```

- **Virtual Machines (VMs)** run on Hypervisors, which allow multiple Virtual Machines to run on a single machine along with its own operating system.

- Each VM has its own copy of an operating system along with the application and necessary binaries, which makes it significantly larger, and it requires more resources.

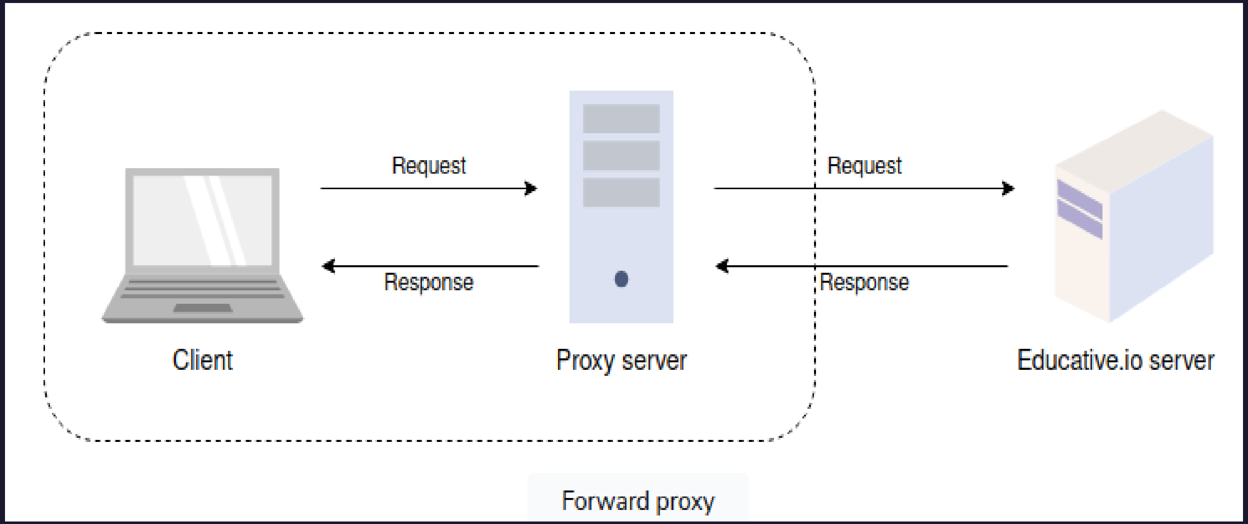
- VM provide Hardware-level process isolation and are slow to boot.

Deployment Scenarios



Proxy Server

- A **proxy server** acts as a *channel* between the *user* and the *internet*. It separates the *end user* from the *website* they're browsing.
- Whenever a user sends a request for an address from the end server, the traffic flows through a proxy server on its way to the address. When the request comes back to the user, it flows back through the same proxy server which then forwards it to the user.
- Benefits :
    - Improved security
    - Improved privacy
    - Access to blocked resources
    - Control on internet usage of employees and children
    - Cache data to speed up requests



- **Virtual Private Network (VPN)** is like proxy server with a basic difference that it encrypts the browsing data or any data you send or receive.

Redundancy and replication

Redundancy

Redundancy is the process of **duplicating critical components of a system** with the intention of increasing a system's reliability or overall performance

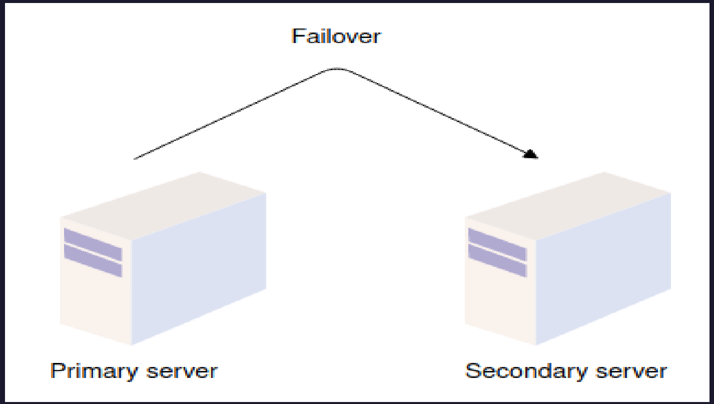
It plays a critical role in removing single points of failure in a system and providing backups when needed or fail-safe.

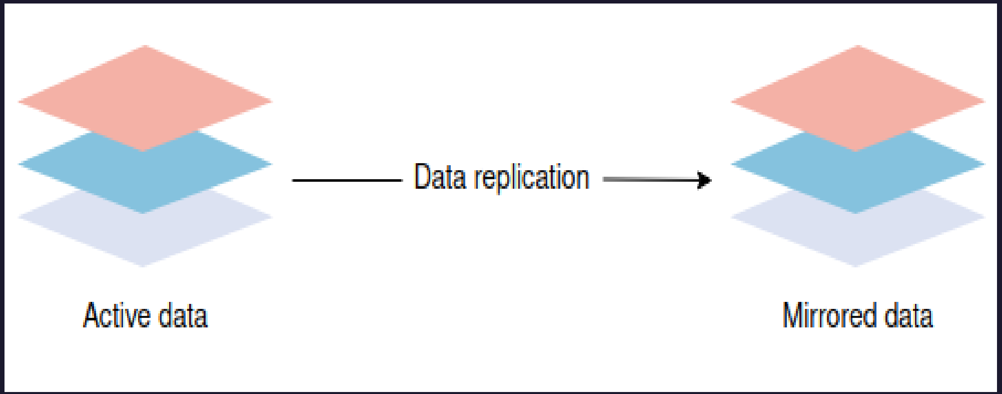
For instance, if we have two instances of a service running in production and one of those instances fails, the system can fail over to another one.

Replication

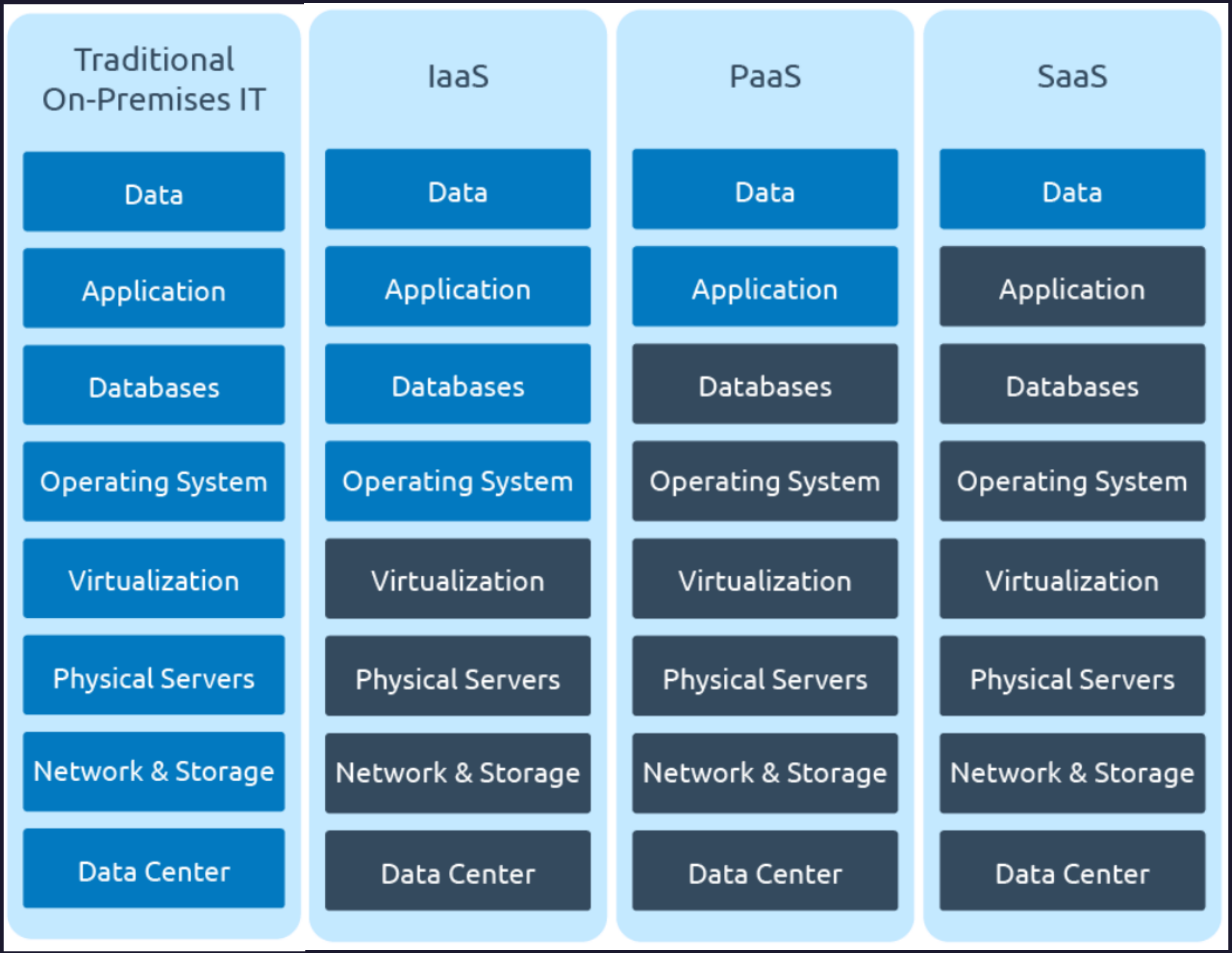
Replication is the process of **sharing information to ensure consistency between redundant resources**

The primary server receives all of the updates, and those updates pass through the replica servers





Cloud Computing Services categories

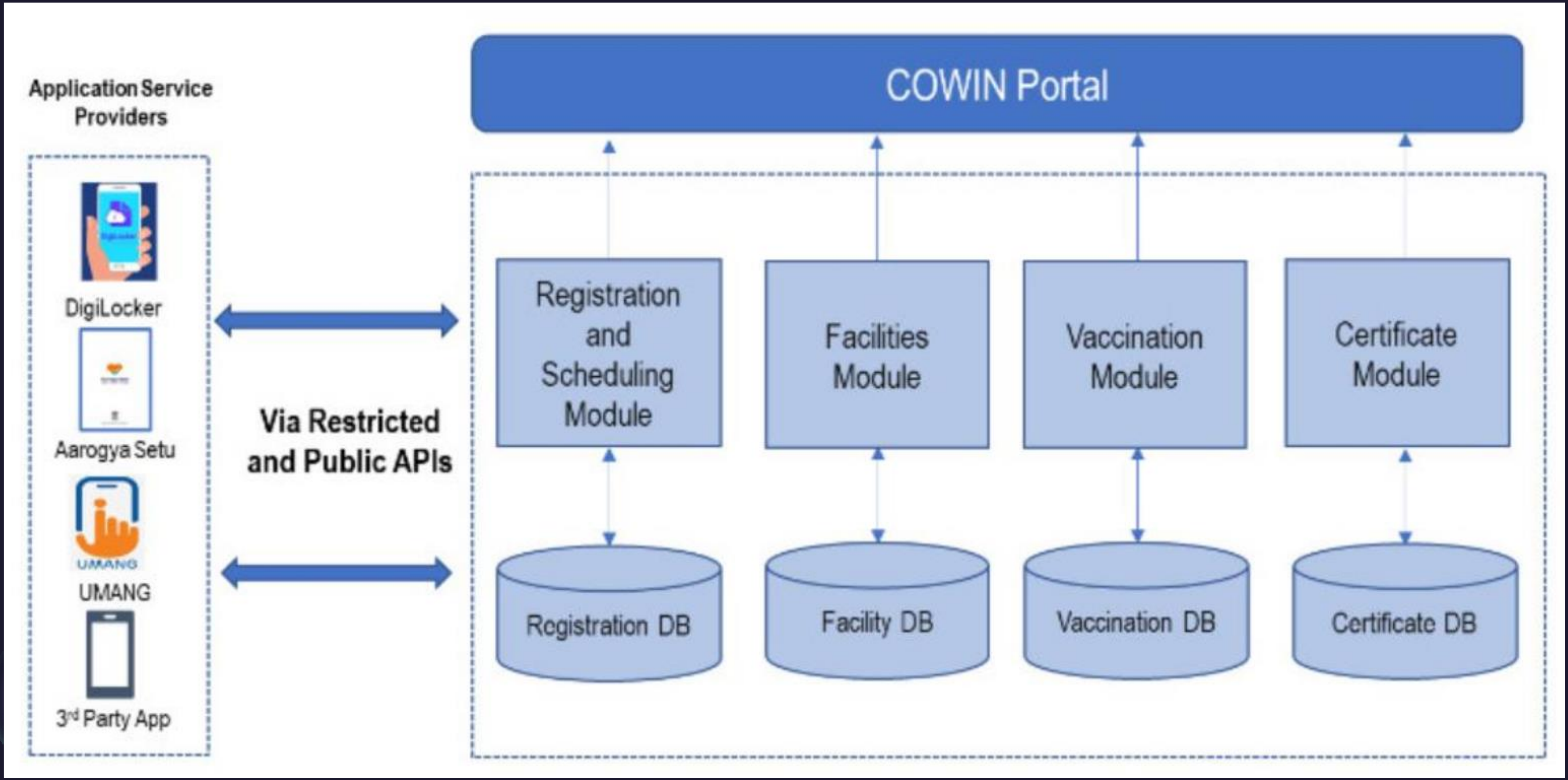


Traditional On-Premises IT	IaaS	PaaS	SaaS
Data	Data	Data	Data
Application	Application	Application	Application
Databases	Databases	Databases	Databases
Operating System	Operating System	Operating System	Operating System
Virtualization	Virtualization	Virtualization	Virtualization
Physical Servers	Physical Servers	Physical Servers	Physical Servers
Network & Storage	Network & Storage	Network & Storage	Network & Storage
Data Center	Data Center	Data Center	Data Center

- **IaaS** : Amazon Web Services, Microsoft Azure (VM), and Google Compute Engine
- **PaaS** : AWS Elastic Beanstalk, Google App Engine, and Adobe Commerce
- **SaaS** : Gmail, Slack, and Microsoft Office 365

⬛ Provider-Supplied 🟦 Self-Managed

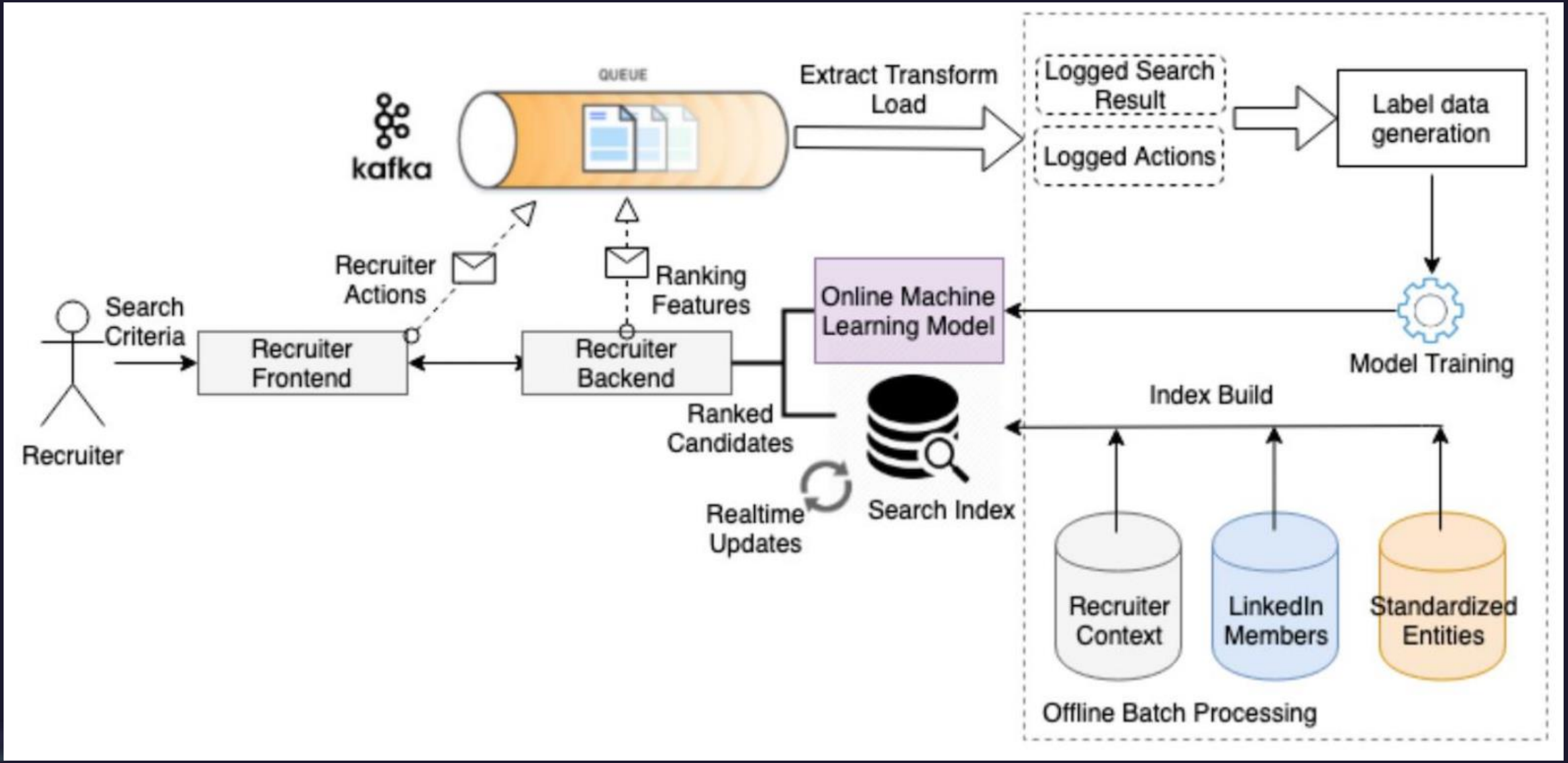
COWIN – Mass Vaccination – India Stack



Source : https://directory.apisetu.gov.in/api-collection/cowin

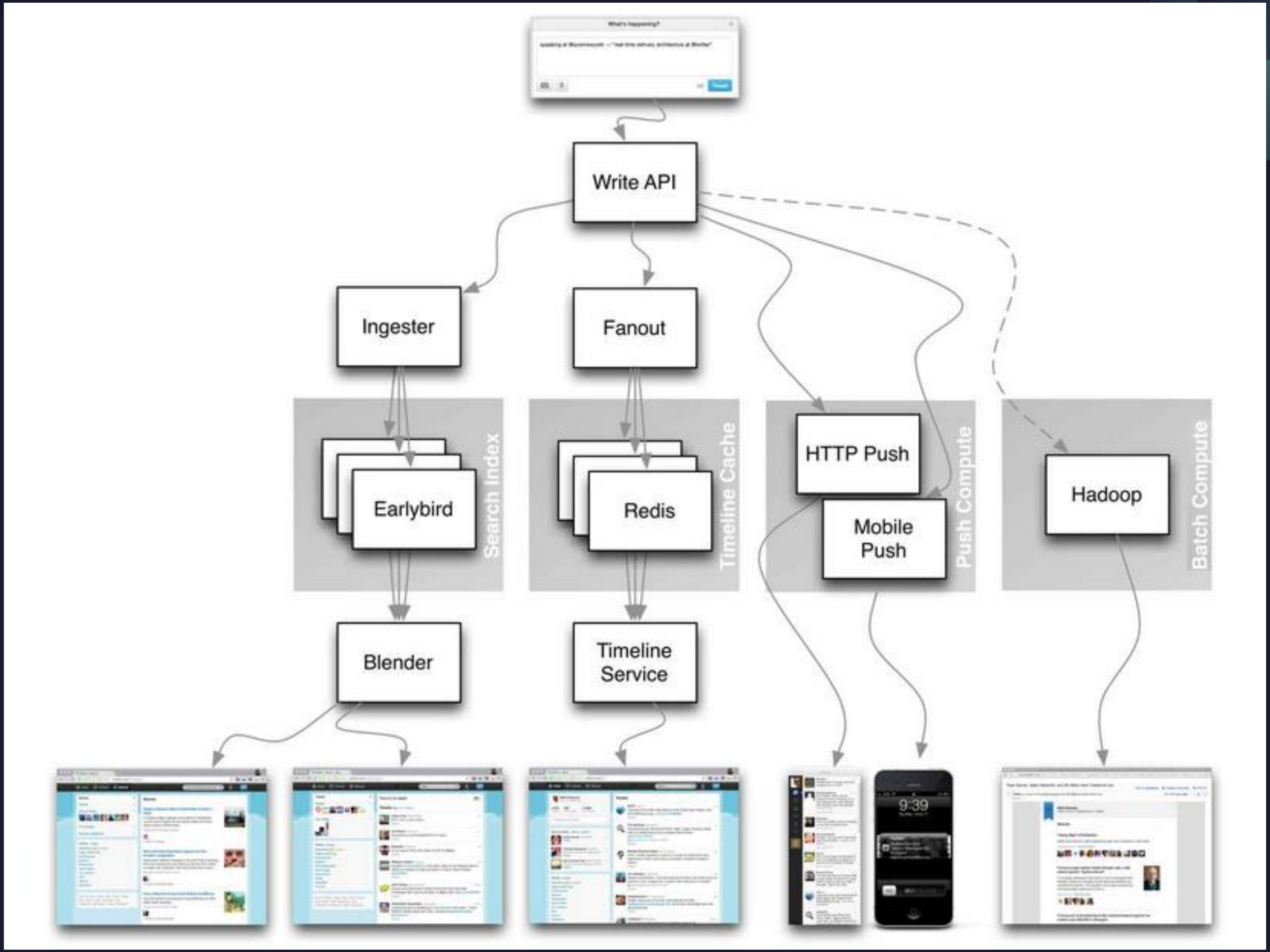
LinkedIn Recruiting Module

(Talent Search architecture and flow)



Source : https://engineering.linkedin.com/blog/2019/04/ai-behind-linkedin-recruiter-search-and-recommendation-systems

Twitter – System Design



Source : https://www.infoq.com/presentations/Twitter-Timeline-Scalability/

References

- https://www.educative.io/blog/complete-guide-to-system-design
- https://www.infoq.com/presentations/Twitter-Timeline-Scalability/
- https://www.geeksforgeeks.org/introduction-to-docker/

Thank you