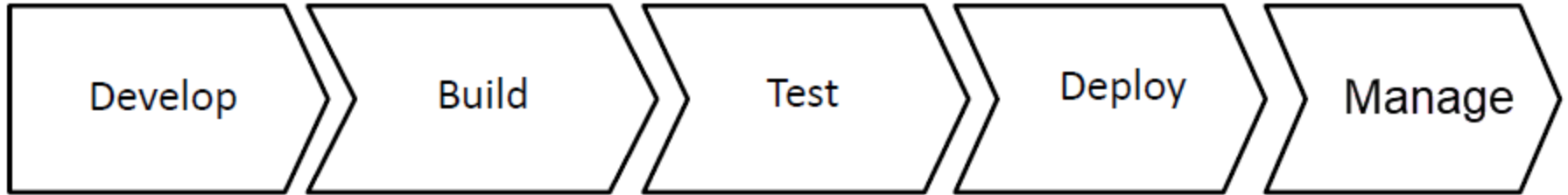




Web App Deployment & Scaling

Software Release Steps



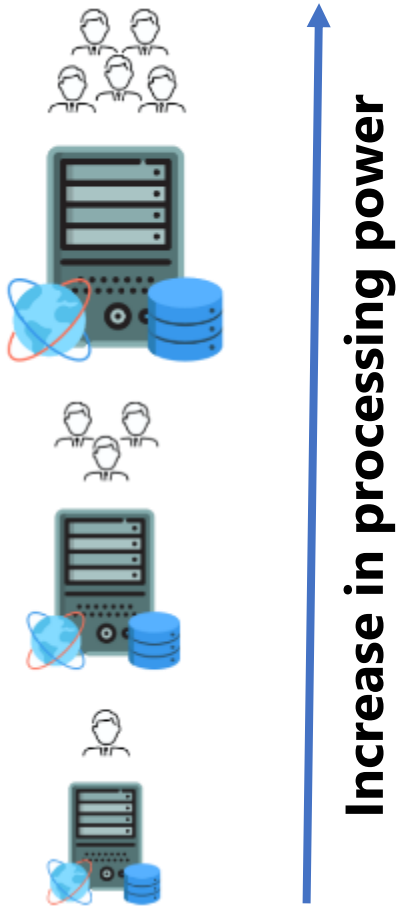
Manage deployed app includes:

- **Collect:** Metrics (response time per Url, CPU/memory usage) and logs
- **Monitor:** Alarms and dashboards
- **Act:** Autoscaling to meet fluctuating demands and the desired performance
- **Analyze:** Trends and metrics (e.g., app usage per Url)

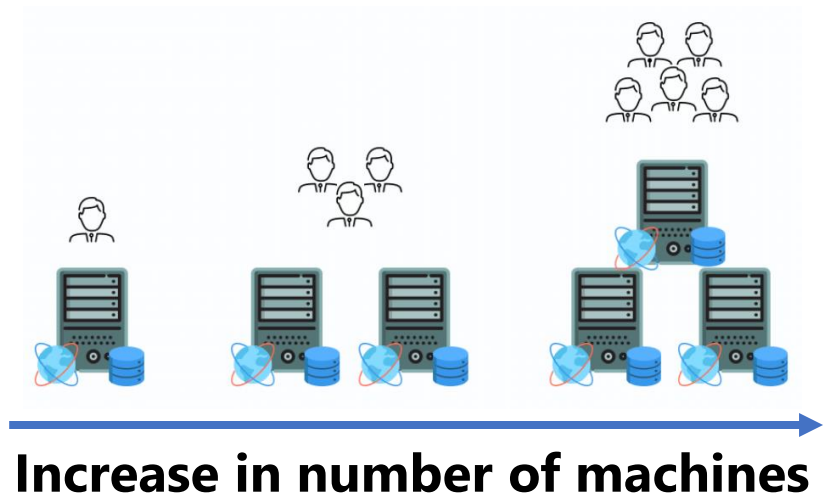
Scaling

- As the number of incoming requests increases, the server may not be able accommodate a growing load
 - The server will eventually burn out and the app crashes!
 - Even the most powerful server still has limited RAM and CPU capacity
- Managing Web application scalability is required
 - Scalability = the ability of a web app to deal with increasing load without breaking down + maintain the desired performance
 - App architecture should allow smooth scaling to accommodate increased load

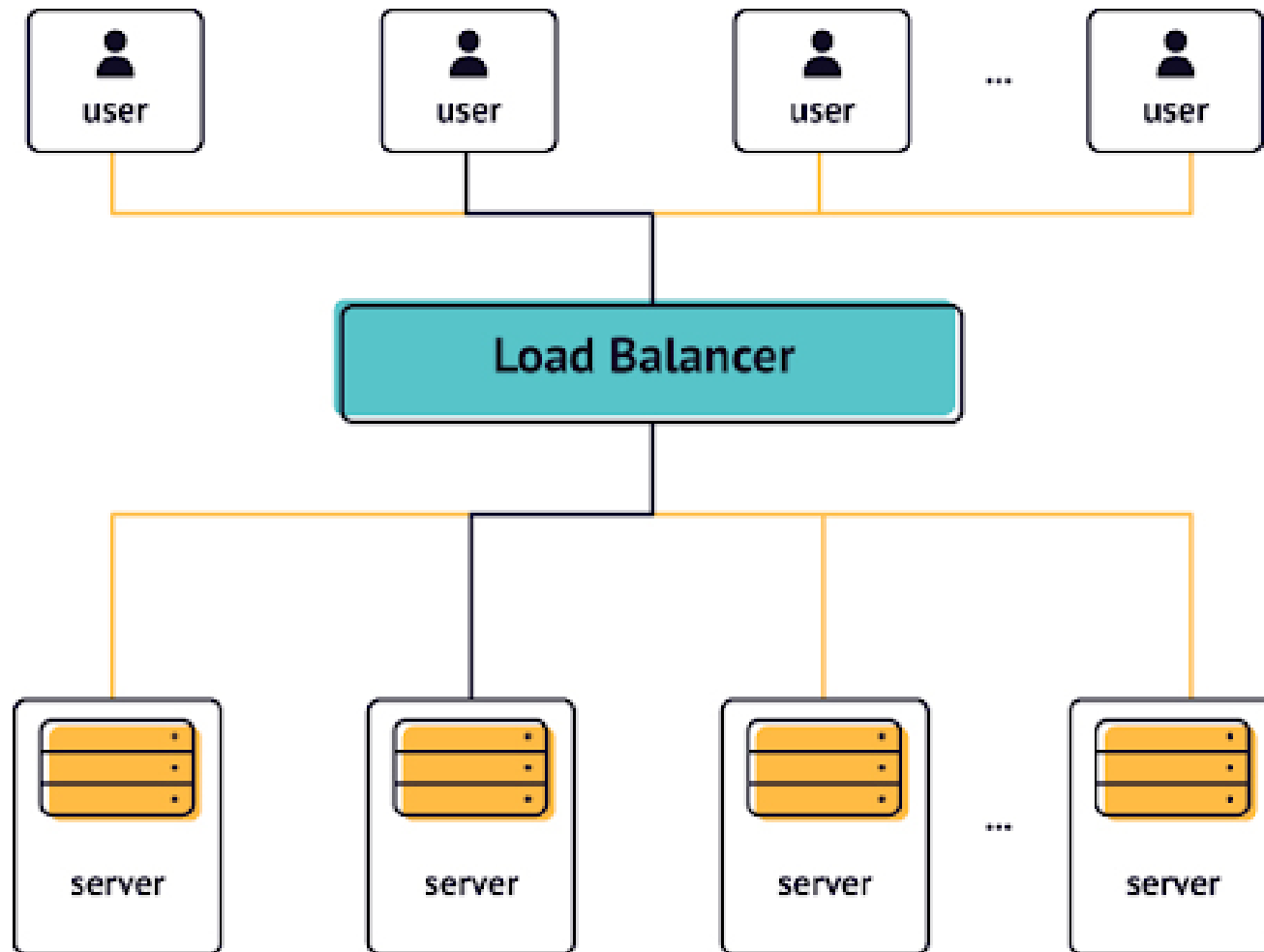
Vertical Scaling



Horizontal Scaling

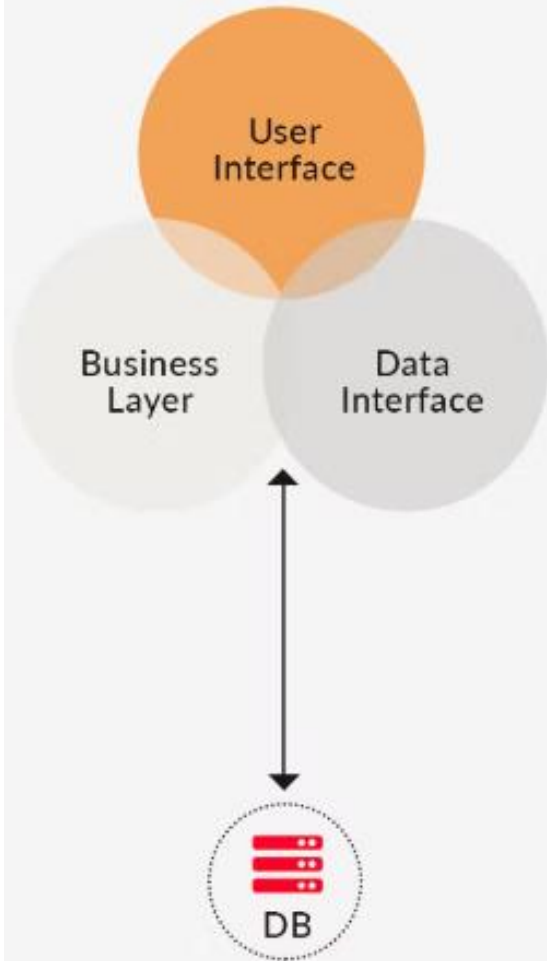


Horizontal Scaling

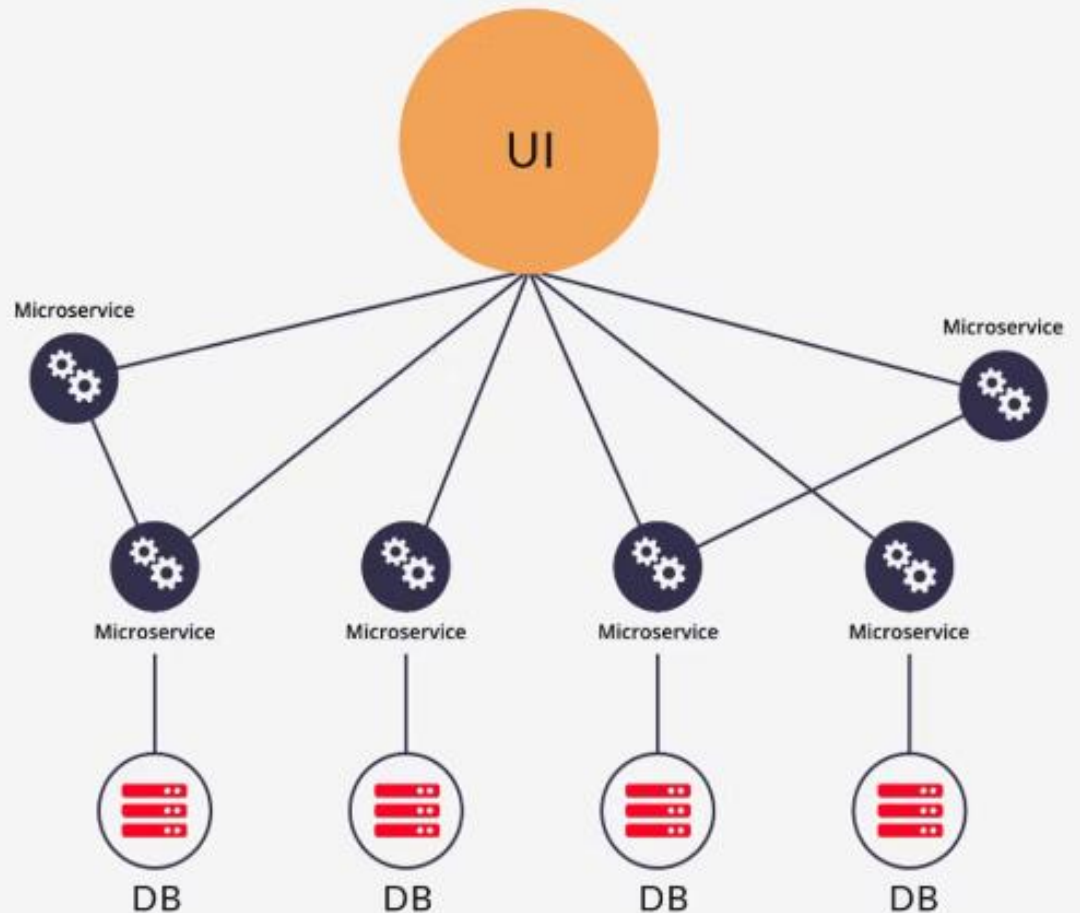


Scaling is easier for modular app design

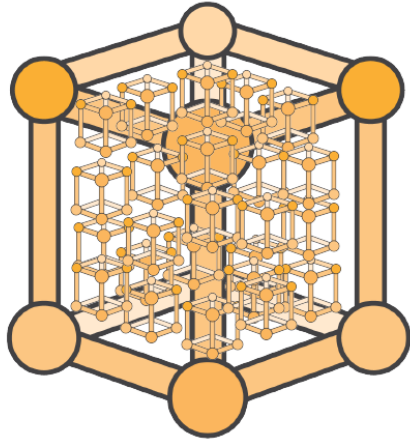
Monolithic Architecture



Microservices Architecture



Monolithic Architecture vs. Microservices Architecture



User interface

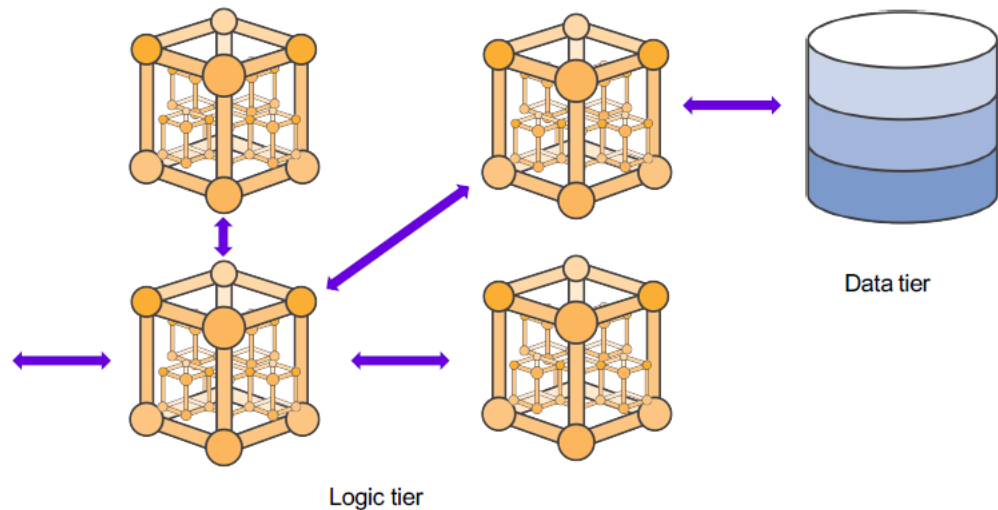
Business logic

Data access

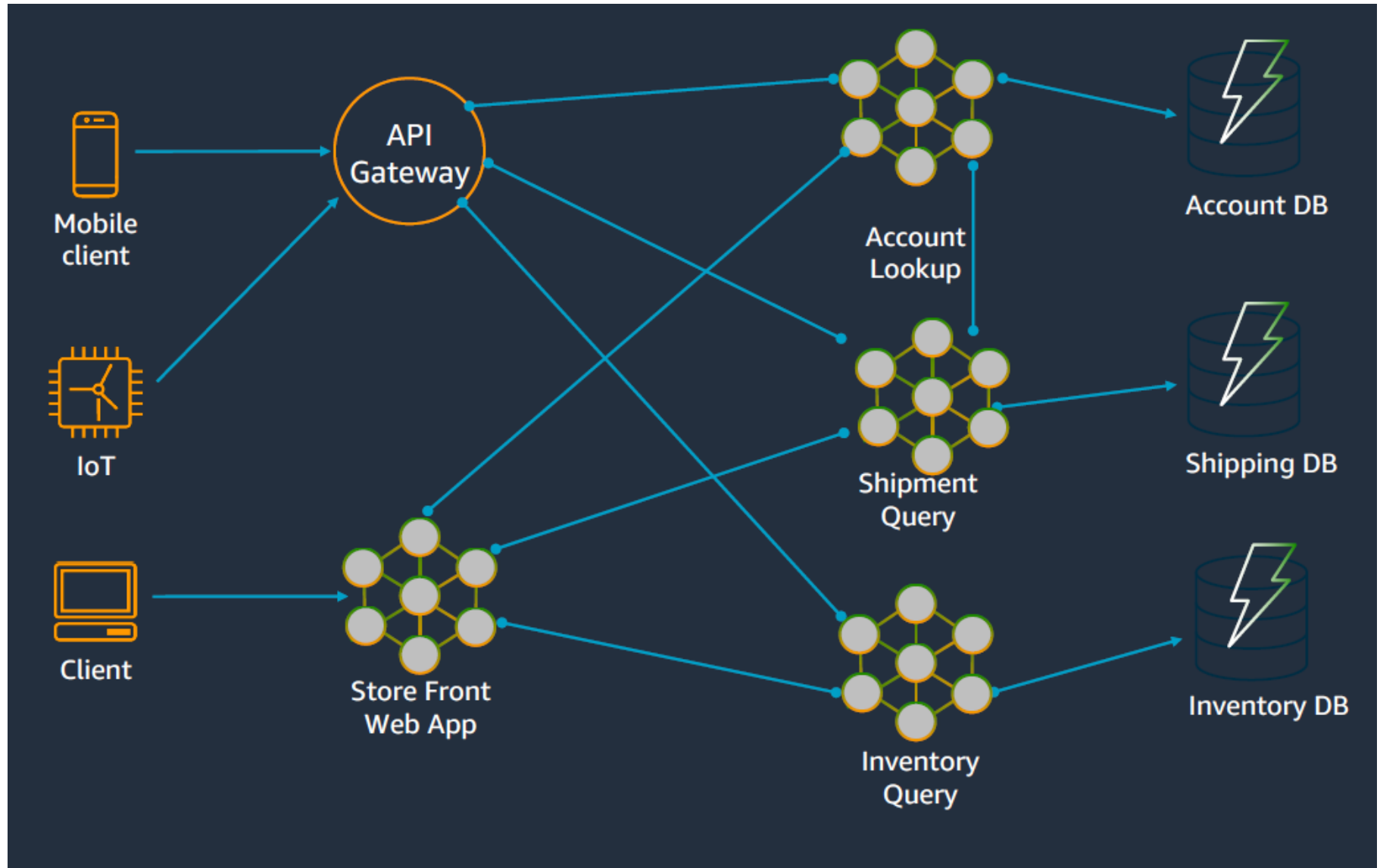
- Services can be deployed separately
- Scaled independently



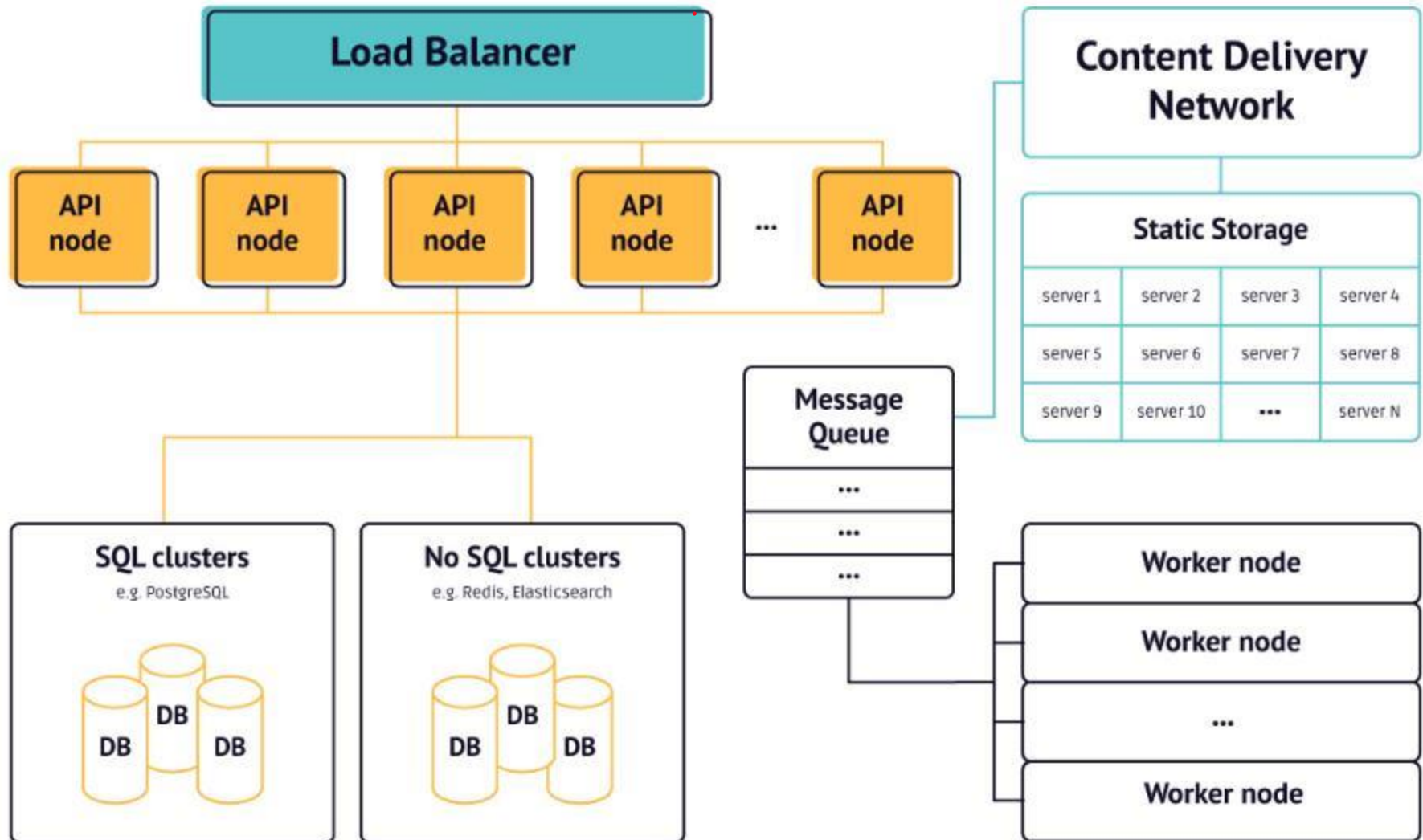
Presentation tier



Microservices Architecture Example



Strategies for Scaling Web Apps



Server Types

- An API server for handling priority requests (e.g., authentication, loading the feed, displaying comments)
- A cluster of database servers for storing dynamic data
- A Static Storage Server for storing BLOB data (images, audio, and other files):
 - Content Delivery Network (CDN) with distributed caching servers located all around the world that promptly deliver content to the users
- The Workers for managing complex tasks that are not required to be done in real-time (e.g., uploading a video on YouTube, convert word to pdf, etc.)

Fundamental problems in Scaling



increasing throughput

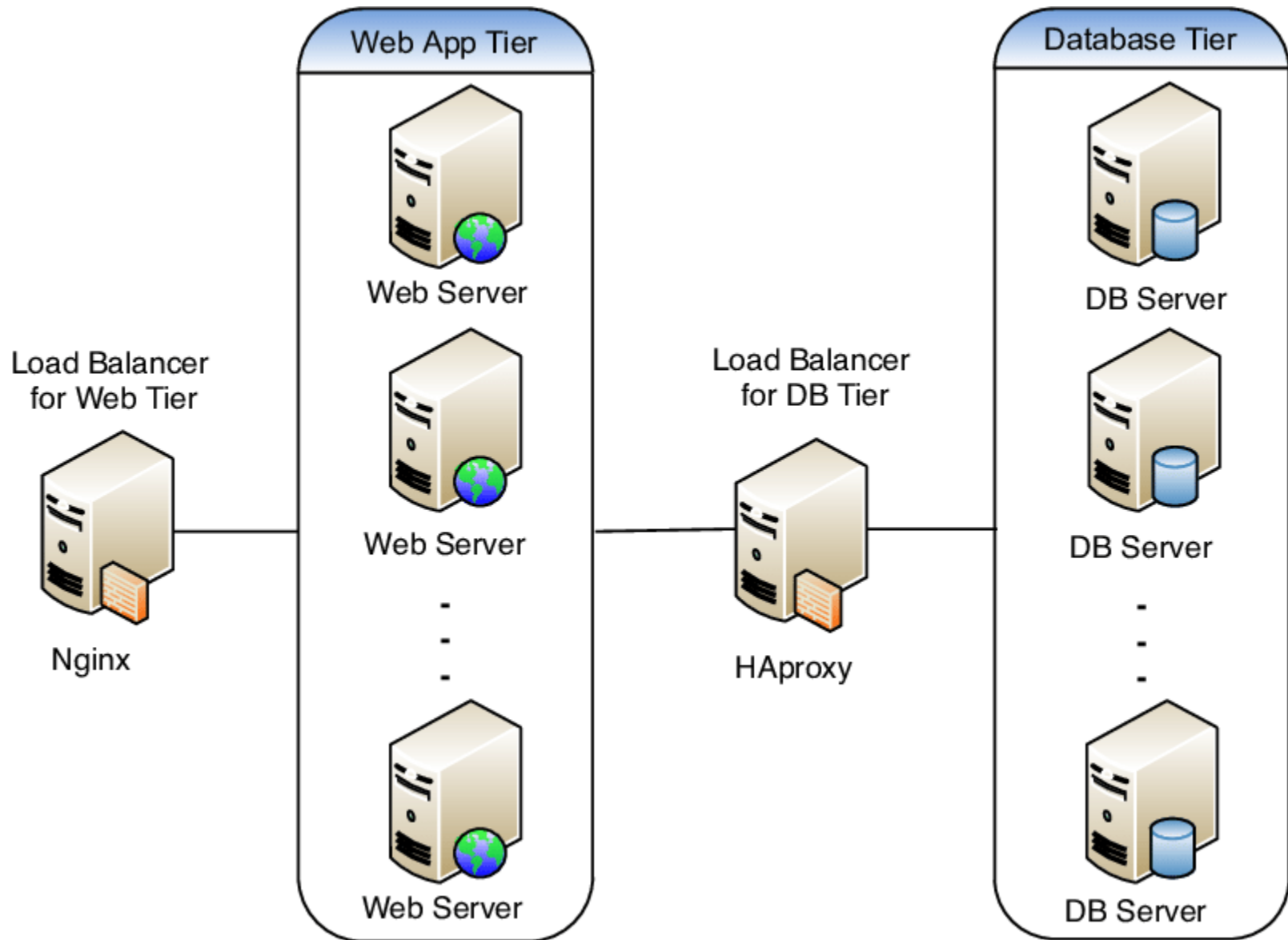
horizontal scaling, network configuration



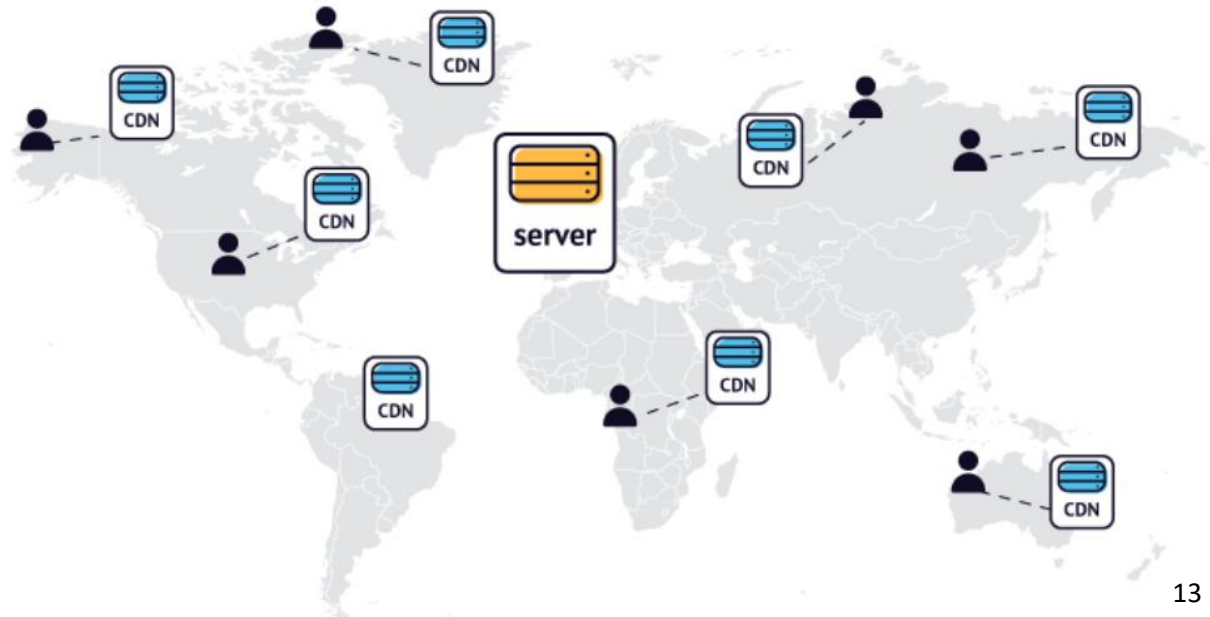
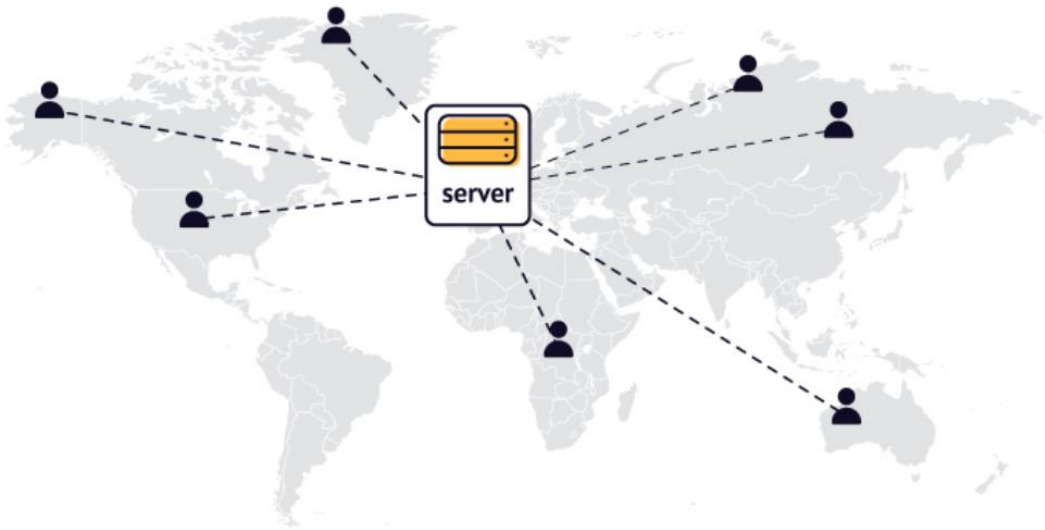
reducing latency

caching, geo-distribution

Typical App Deployment Architecture



Without CDN vs. With CDN



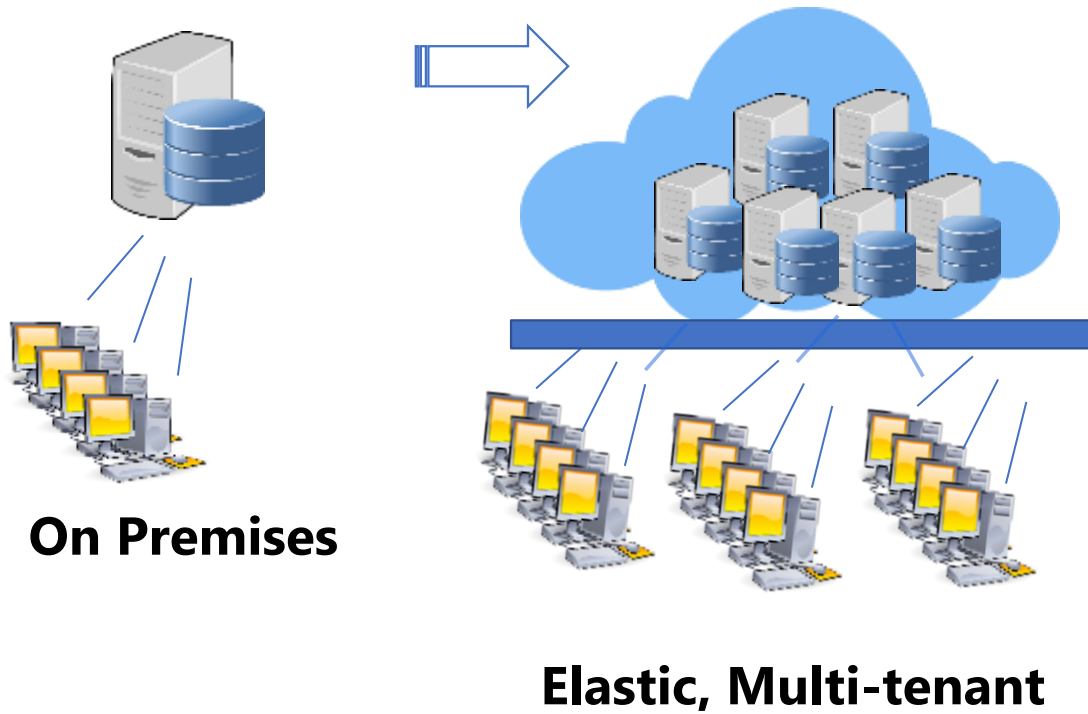
On-premise vs Cloud Hosting

- An on-premise cloud requires that you hire and manage your own datacenter while a hybrid model relies on third parties to keep up operations
 - On-Premise = Hosting data and applications in servers within an organization and manages them internally
+ full control but costly
- Cloud hosting = hosting data and applications third on party servers
 - Pay on per use basis
 - Save money through cutting down on staffing costs, infrastructure and maintenance costs
 - E.g. Microsoft Azure, Google Cloud Platform, Amazon Web Services

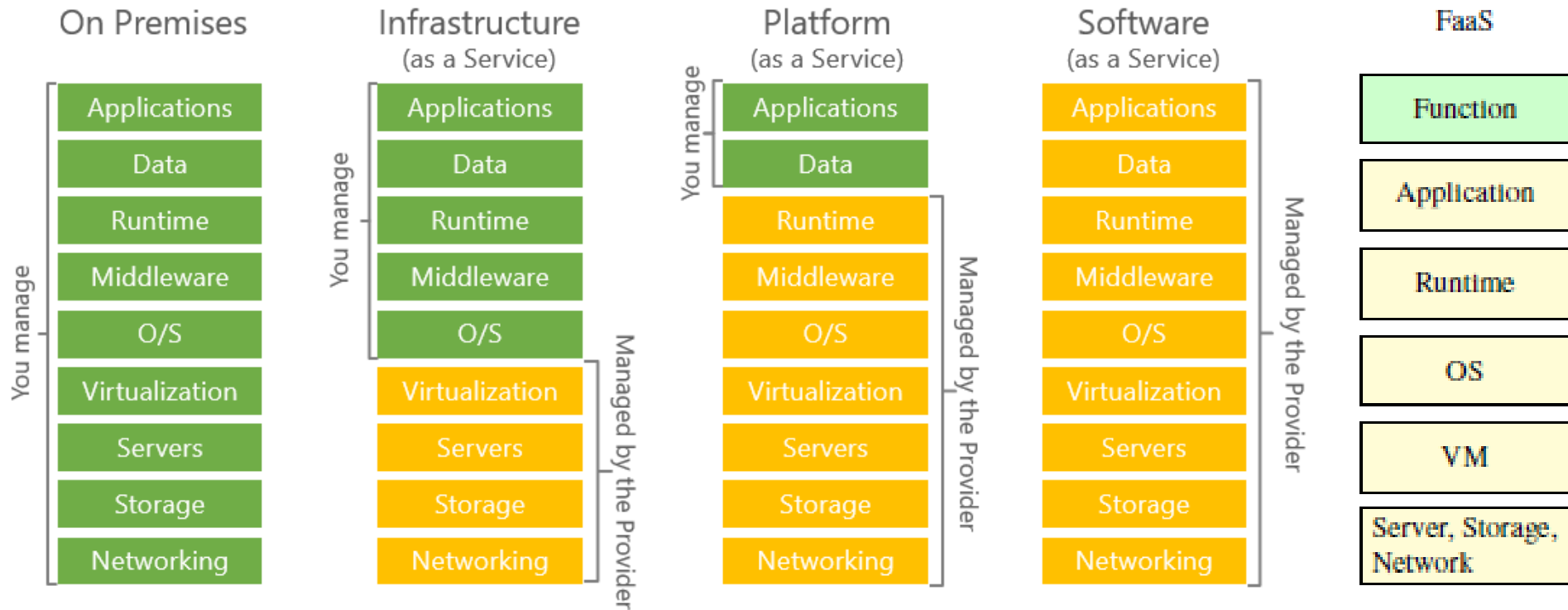
What is Cloud computing?

- External compute resources
 - No assets owned
- Access delivered over the network
- Clients are 'tenants', not owners
- Scalable without purchasing new equipment
 - Ideally dynamic
- Financial model is by usage (vs. by asset)

A shift ...



Cloud Service Models



Cloud Service Models

- Cloud Infrastructure as a Service (IaaS)
 - Rent processing, storage, network capacity, and other fundamental computing resources (Basically a hosted, rented data-center)
- Cloud Platform as a Service (PaaS)
 - Deploy customer-created applications to a cloud
 - Frameworks and platforms (OS, Management, Core storage and replication are provided)
- Cloud Software as a Service (SaaS)
 - Use provider's applications over a network (e.g., Salesforces, Microsoft Office, SharePoint)
- Function as a Service
 - Serverless computing – resources are only used on-demand to run a function. There is no VM being held by the user. e.g., Lambdas

Cloud Infrastructure

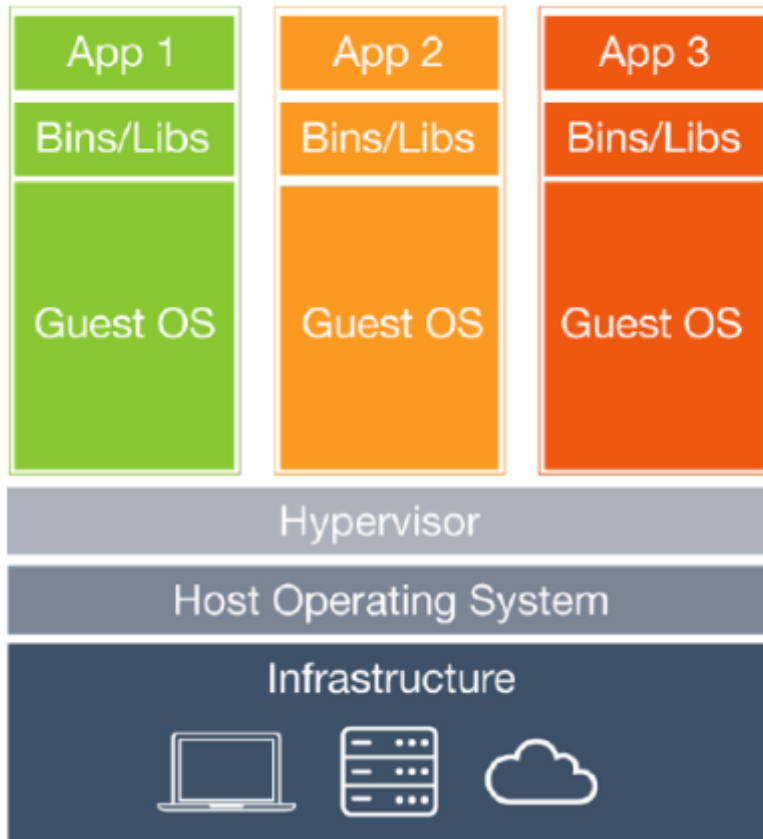
SaaS

FaaS

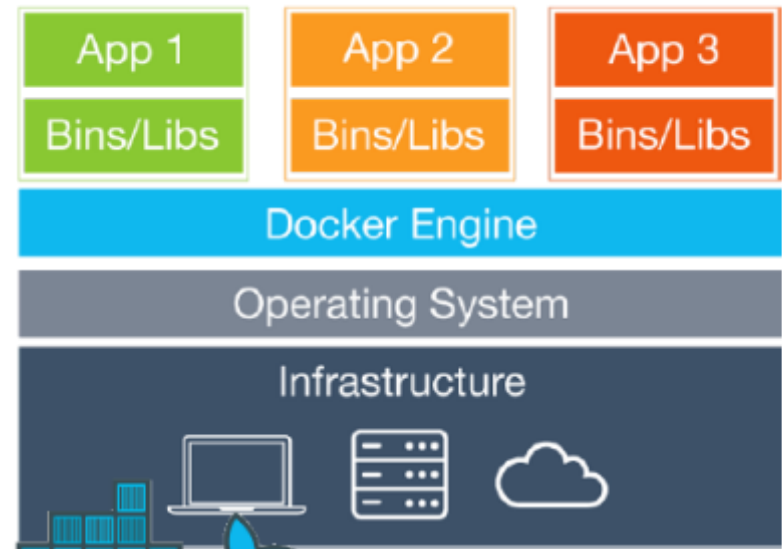
PaaS

IaaS

Virtualization Options



Virtual Machines



Containers

Cloud Deployment models

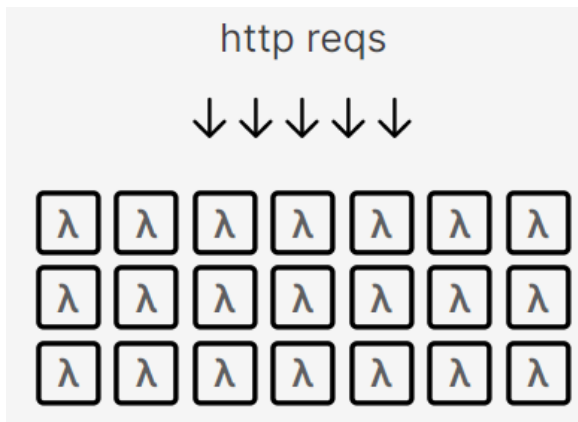
There are multiple deployment models to fit business needs

- Public Cloud
 - For the masses
- Private Cloud
 - Enterprise owned, but using Cloud Technology
 - Can be local, or remote (but segregated)
- Hybrid Cloud
 - Combines Public and Private

FaaS – Function as Service

- Function as a Service

- Serverless computing: resources are only used on-demand to run a function. There is no VM being held by the user
- Allow building apps faster without managing infrastructure
- Event-driven compute: Functions triggered by events



λ . ∞ -scalable

λ . no ops needed

λ . pay for the compute you need
(down to 100ms)

FaaS = Serverless Functions

- General-purpose FaaS providers: [AWS Lambda](#), [Google Cloud Functions](#), [Azure Functions](#)
 - Deployment platforms + SmartCDNs such as **Vercel** and **Netlify**
 - Offer CDN that is also capable of executing code
 - Better DX, less configurable
 - Serverless Functions are usually deployed and invoked in one region, so requests sometimes travel far to reach the database before they can deliver a response to the user
- => This limitation can be addressed using edge functions

<https://fauna.com/blog/comparison-faas-providers>

Serverless Function Examples



File Processing



Handle a Web Request


Edge Functions

- Edge functions = if a form of serverless compute functions that execute closer to globally distributed users
 - Edge Functions are duplicated and deployed across a global network of data centers
 - Edge functions aim to minimize latency by executing geographically nearest to the request, or near the database => their network requests travel shorter distances
 - Kind of CDN that is also capable of executing code
- Personalized dynamic content: You can use custom code in the Edge Functions to deliver content based on which data center on the Edge Network a user is invoking a Function in

Edge Functions

```
// pages/api/hello.js
export const config = { runtime: 'edge' }
export default function handler(req)
{
  return new Response("Hello World");
}
```

- Edge Function can optionally specify which region the function should execute in
 - Using the config object that the Edge Function exports

 pages/api/hello.js

```
export const config = {
  runtime: 'edge', // this is a pre-requisite
  regions: ['iad1'], // only execute this function on iad1
};
```