**Monolithic**

monolithic design is interconnected and interdependent. the application is managed in internal layers or libraries. These layers could be for security, reporting, analytics, user interface, or data access. The layers are tightly connected and dependent on each other.

Benefits:

simplicity and less cross-cutting of features

Drawbacks:

design becomes more complex and difficult to modify

adapting to new technology which would mean re-writing the whole application

**Service Oriented Architecture (SOA)**

designed and built with a service provider and consumer approach.The services provided work as a discrete unit of functionality, integrated seamlessly, and are easily reused. three components:

interface - defines how a service provider will execute requests from a service consumer

contract - defines how the service provider and service consumer should interact

implementation - the service code

Benefits:

increase reliability and support parallel development due to a fixed contract between the services

Drawbacks:

an become complex and obstruct rapid application

expensive investment suits enterprise teams, who can invest the required resources and expertise.

Eg. banking system

Difference between SOA and microservice is microservices are fine-grained with independent data storage for each, that is, the bounded context.

**Microservices**

Microservices make each application component its own service, and each service communicates via an API, microservices allow application components to use different technology stacks, microservices enable individual components to scale in response to demand, microservices lessen risks associated with change because components can iterate independently, and failures in one service do not necessarily impact other services. the data in a microservice is not shared with other services, which helps you to scale individual microservices independently.

***Patterns***

1]the single-page application (or SPA)

web interfaces began incorporating all functionality into single-page applications. The user enters through one interface that never reloads the landing page or navigates away from that initial experience.

Benefits:

simplifies the front-end experience

2] Strangler Pattern

Steps in this pattern are:

transform-This creates a parallel new site on a cloud platform or within your existing environment.

coexist- This leaves the existing site functional and live for a specified time. It incrementally redirects from the current location to the new site for newly implemented functionality.

eliminate- This removes the outdated functionality from the existing site or stops maintaining that functionality when you redirect traffic from the original site.

3]service discovery pattern helps applications and services discover each other

***API Gateway***

the door to your backend services, it enables you to plug additional services while providing your client with unified access to your services, it hides the complexity of your backend services from the client, it makes it easier to scale or replace your backend services

Benefits:

It insulates the clients from how the application is partitioned into microservices.

In other words, it simplifies the client side by moving the

logic for calling multiple services from the client to the API Gateway.

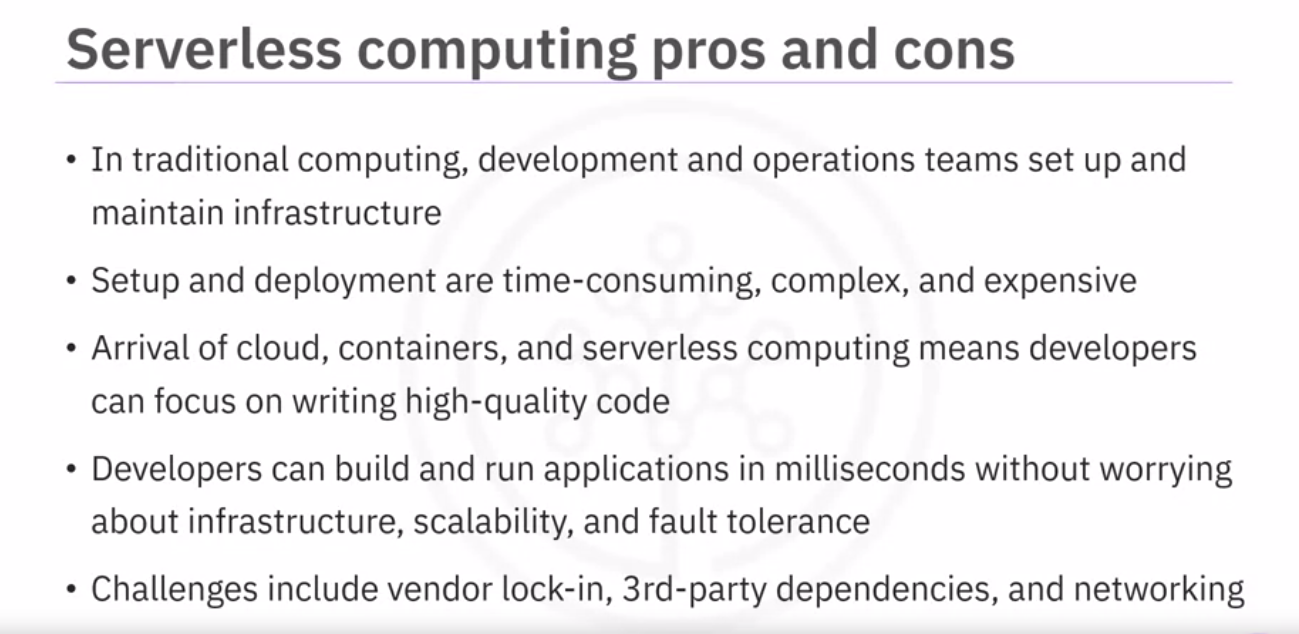
It also provides the optimal API for each client, regardless of who the client is.

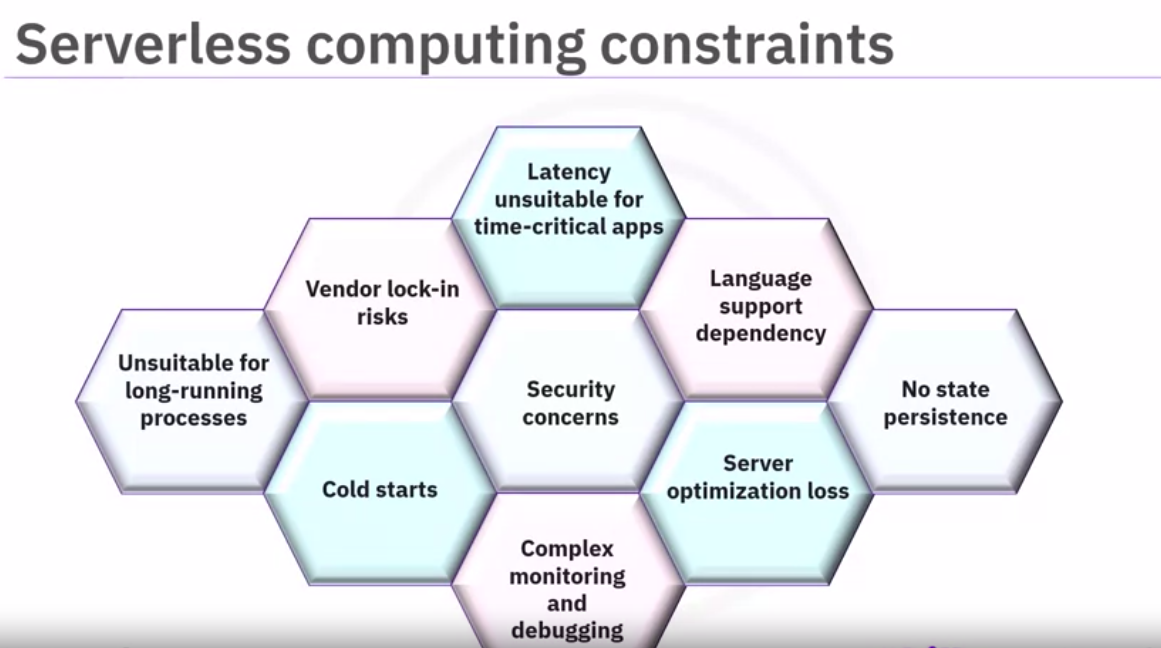
It reduces the number of requests or round trips.

irrespective of how your microservices communicate internally, an API Gateway

will provide a standard protocol to communicate with the outside world.

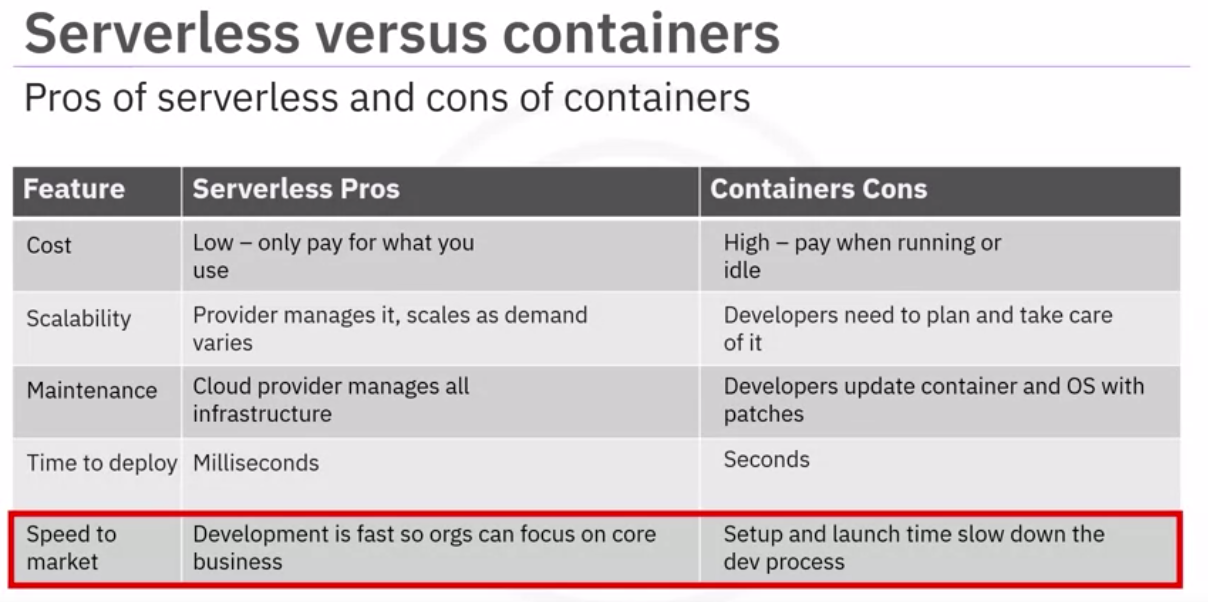
***Serverless***

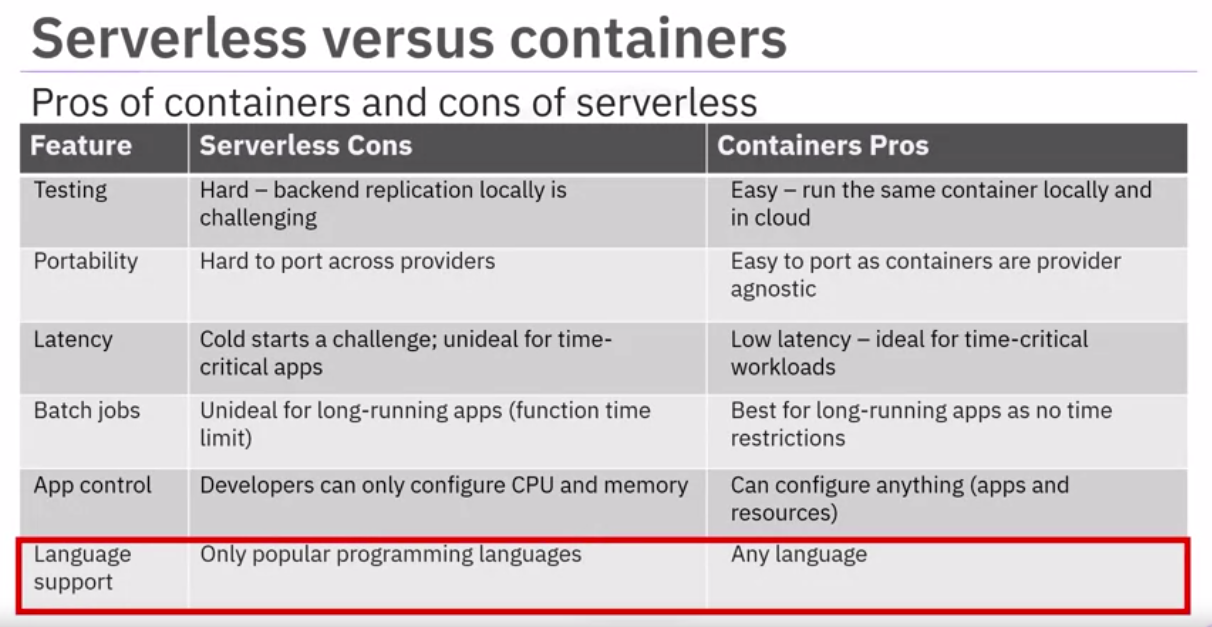




follow the standard industry advice:

“Build serverless first. If needed, move to containers”.





**FaaS (function as a service)**

FaaS is a type of cloud-computing service that

allows you to execute code in response to events without a complex infrastructure,

FaaS is a subset of serverless computing that

creates applications in the form of multiple functions,

FaaS can be deployed on hybrid clouds as well as on-premises environments,

FaaS is billed on the time taken to run functions and not on server instance sizes,

a serverless stack is comprised of FaaS, BaaS, and an API Gateway,

and there are several different managed and self-managed FaaS platforms to choose from.

***Serverless***

The Serverless Framework is a free and open-source web framework written using Node.js.

Functions are triggered by events.

The Web Application reference architecture is event-driven that uses:

AWS Lambda

Amazon API Gateway

Amazon DynamoDB

Amazon Cognito

Amazon Amplify Console

The use cases for serverless web applications are Event streaming, Post-processing, and Multi-language.

Popular serverless platforms: AWS lambda, google cloud function, MS Azure cloud