White paper

Web application/ Mobile application

Web Application

AWS Cloud

EC2

S3

Route53

EFS

SES

SMS

WorkMail

Security

IAM

Microservice + Serverless (Lambda function)

Frontend framework:

Backend design:

(Apache server) Elastic beam

Distributed server:

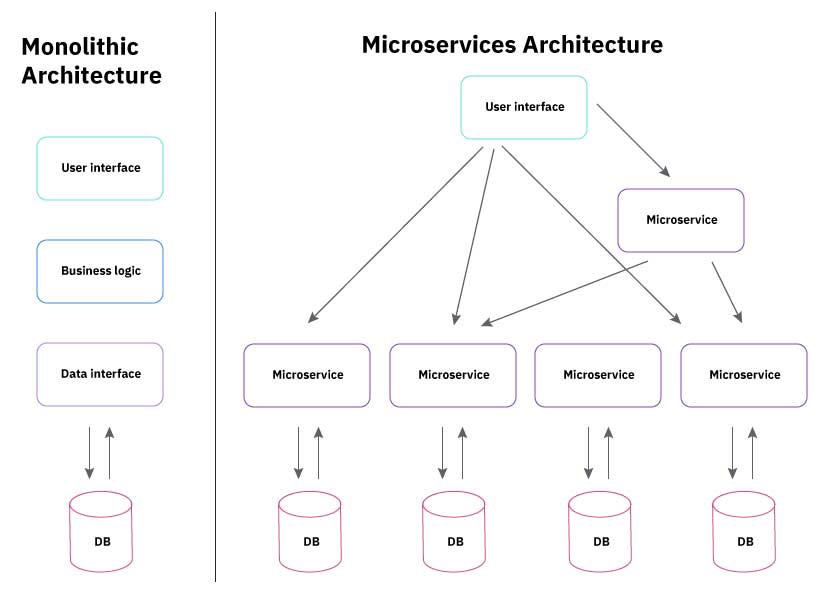
Node:

Elastic server

Angular + Java + SQL

IBM microservices definition: Link

The microservice approach encourages enterprises to become more agile, with cross-functional teams responsible for each service. Microservice architecture structures the application as a set of loosely coupled, collaborating services. Microservices are inherently distributed systems. Implementing such a company structure, as in Spotify or Netflix, can allow you to adapt and test new ideas quickly, and build strong ownership feelings across the teams.



Most large-scale web sites including Netflix, Amazon and eBay have evolved from a monolithic architecture to a microservice architecture.

Cadenza First problem: App crashes

We started the modularising complete Cadenza.tech application as follows

a. User management

b. Product management

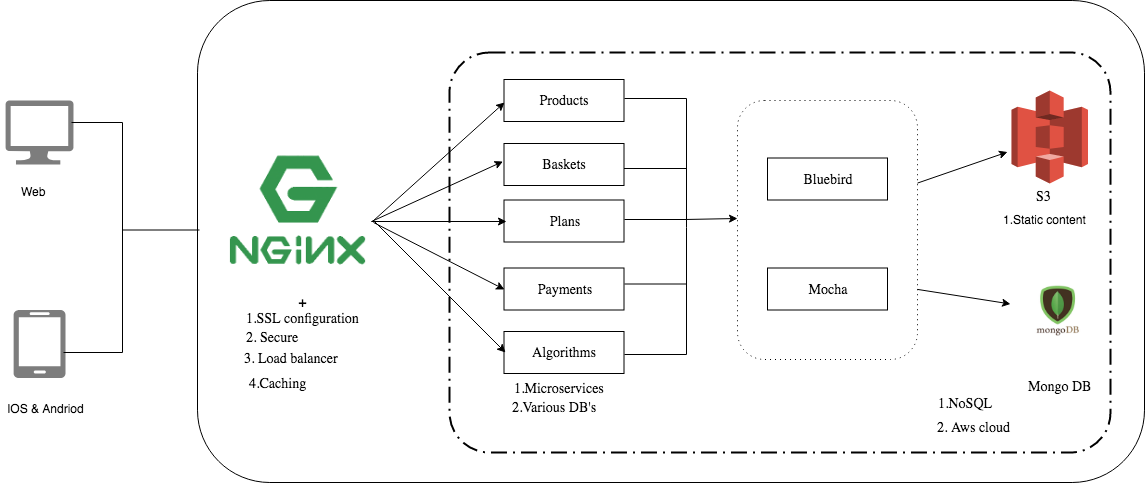
c. Order management

d. Plans management

e. Payment management

f. Algorithms

Cadenza2.0 beta architecture



Each category had almost 5 or 6 APIs. By adopting micro-services pattern, we had great extent advantages such as.

1. Had an opportunity to code services independent of programming language & database. Eliminates any long-term commitment to a technology stack.

We wrote payment services in python and user management in NodeJS.

2. Multiple teams worked in parallel

Frontend, backend and server (DevOps) teams worked in parallel. Teams loved coding microservices, as they were quite fine-grained versions. Debugging was easy peasy.

3. Enabled the continuous delivery and deployment of large, complex applications.

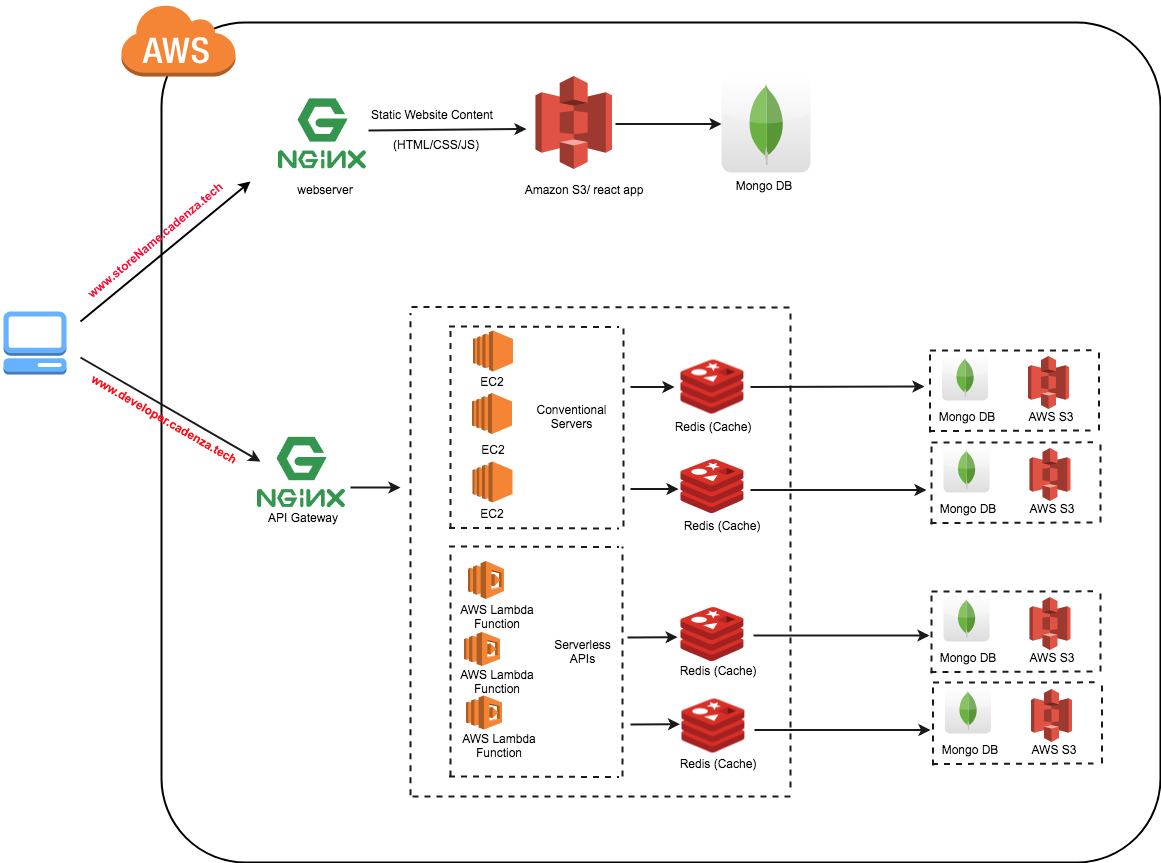
Cadenza second problem: Huge costs

Microservices can be easily hosted in the cloud, Docker environments, or any other server platform, and can be very easily scaled as each service can live on its own server(s), VPS (es) etc.

When dividing a monolithic platform into smaller pieces, each of them should be easy to deploy in an automatic way. Nowadays, we see two main concepts that facilitates such a process — containerization and serverless architecture.

By using Docker, we are able to easily run our Microservice oriented applications on our developer machines and then run the same setup on pre- and production servers. Read more

Serverless/ FaaS(Function-as-a-Service) is easy to learn and implement, and it allows you to reduce the time to market. It also allowed us to reduce costs, and to scale easily.

Cadenza2.2 beta architecture diagram

We’ve used both conventional servers for NodeJS/Python/Java APIs deployments and AWS Lambda for serverless APIs. This hybrid architecture helped us in cutting down huge costs. Monthly cloud recurring costs reduced from 40k to ~7k. This was one of the biggest achievements at Cadenza.

Cadenza.tech scaled to serve 10k+ concurrent users.

Adaptation of Master-Slave database strategy was one the important decisions in reducing the costs. Soon I will be writing blog on it.