Final Project
CMPE-281 Cloud Technologies

Picasso

Picassa

A Twitter-like Marketplace for Pictures

Team Ace-traordinary:

- Ravijeta Kommalapati
- Ramya Dharwada
- Nasrajan Jalin
- Masi Nazarian

San Jose State University | Department of Computer Engineering
Professor Paul Nguyen
Spring 2019

Outline

Picassa

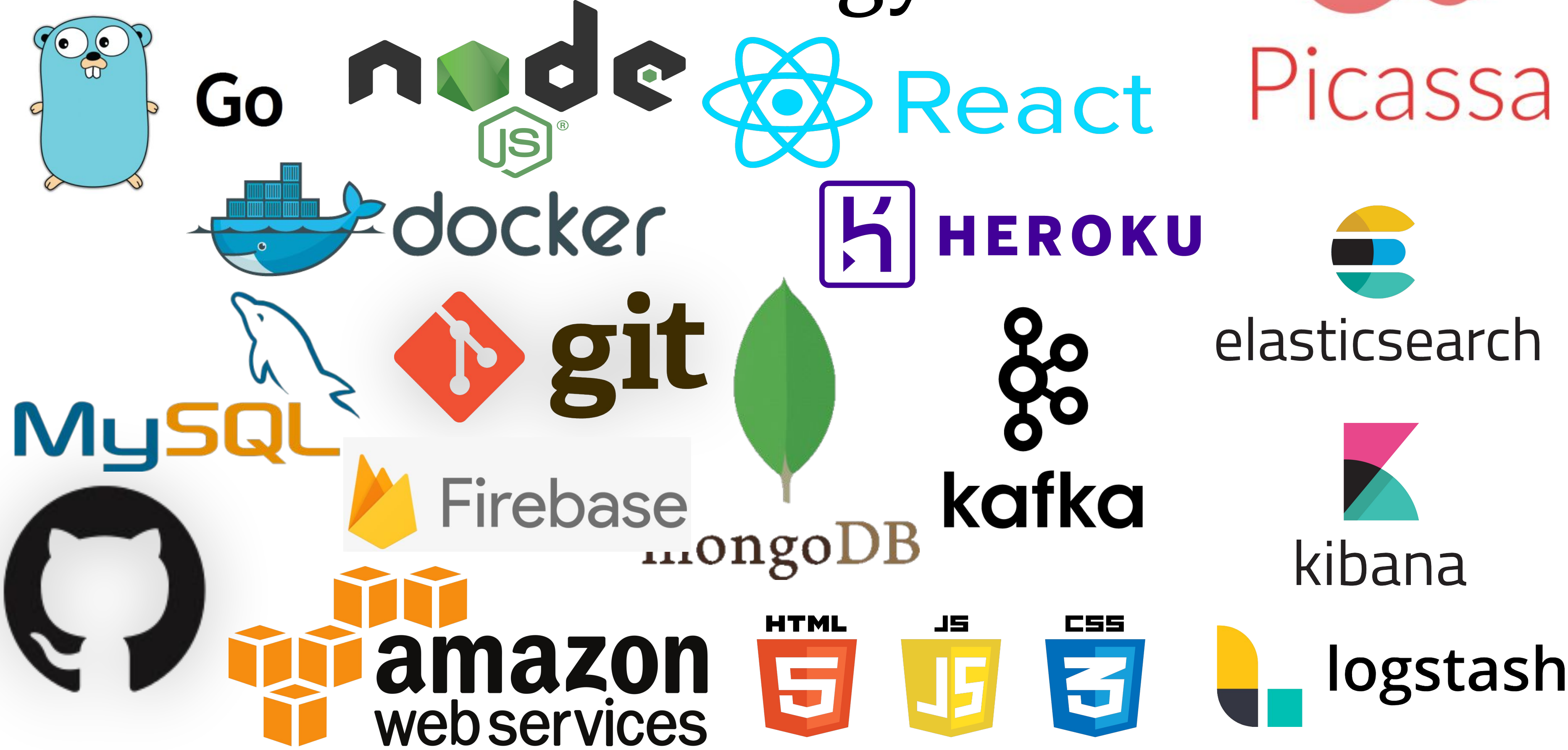
- Problem Statement & Overview
- UI Design
- Architecture
- Database
- Demo
- Challenges
- Q&A

Picasso
A Twitter-like Marketplace for Pictures

Picassa

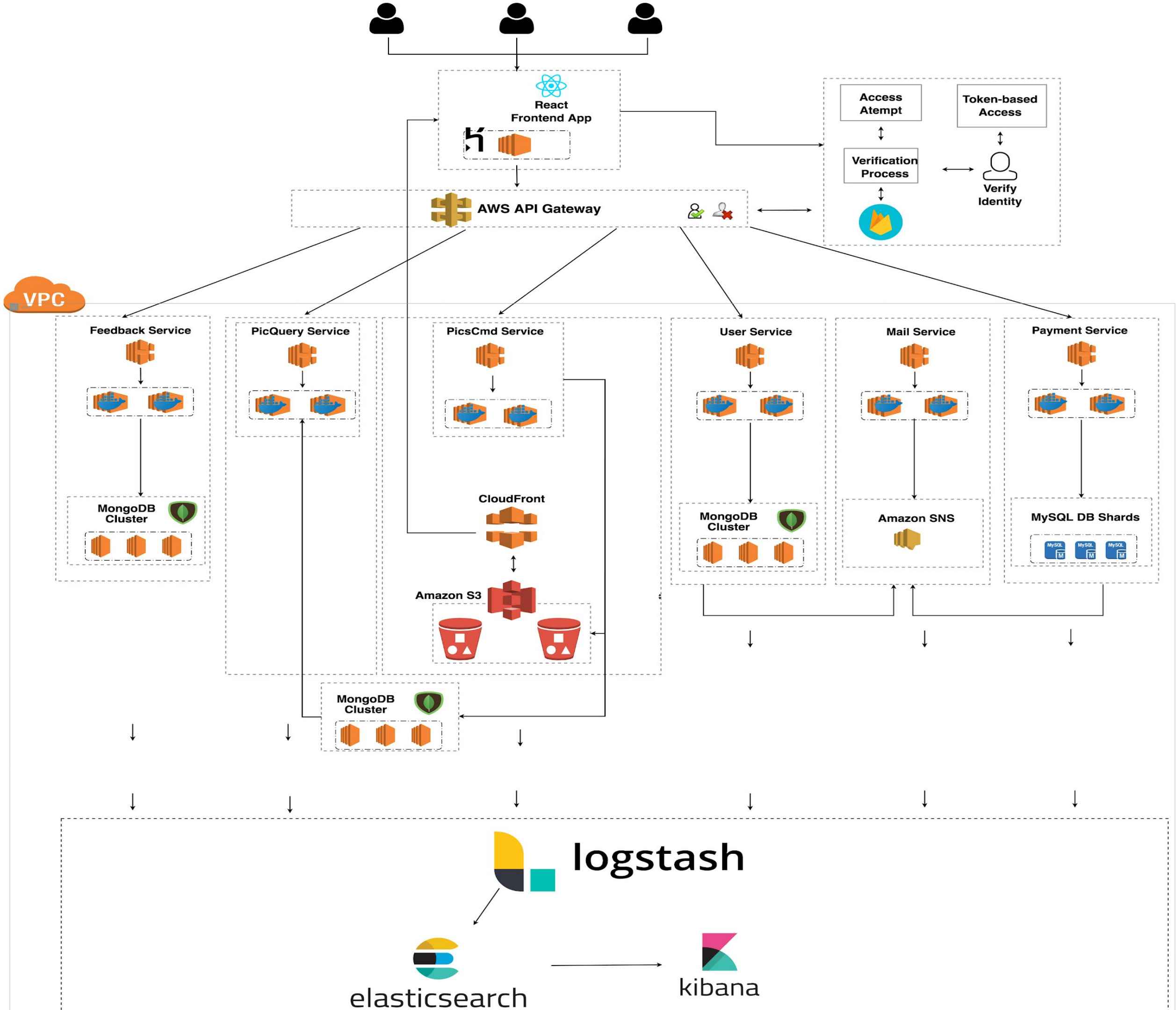
- Register as a user and log into your Picasso account
- Upload an original picture to our database for other users to explore and purchase
- Explore pictures uploaded by other users in one place and purchase the original size of pictures that you like
- Pay for picture purchases through our Payment service
- Express you feedback about pictures by giving it a “Like”

Picasso :: Technology Stack



Picasso :: Architecture

- ReactJs | Heroku
- CQRS Pattern
- Logging
- MongoDB Cluster
- MySQL Cluster
- Amazon S3
- Amazon Cloudfront
- Docker
- Amazon SNS
- Firebase Authentication
- AWS API Gateway
- ELK Stack
- Stripe Payment Gateway



Picassa

Design

Features

- Frontend; ReactJS - Deployed to Heroku
- Backend - Five Golang Microservices:
  - User service
    - Mail Service
  - Payment Service
  - Pictures Command Service
  - Pictures Query Service
  - Reactions Service

Wow Factors

- CQRS Pattern
- Logging - ElasticSearch
- MongoDB & MySQL Database Sharding
- AWS API Gateway
- Firebase Authentication

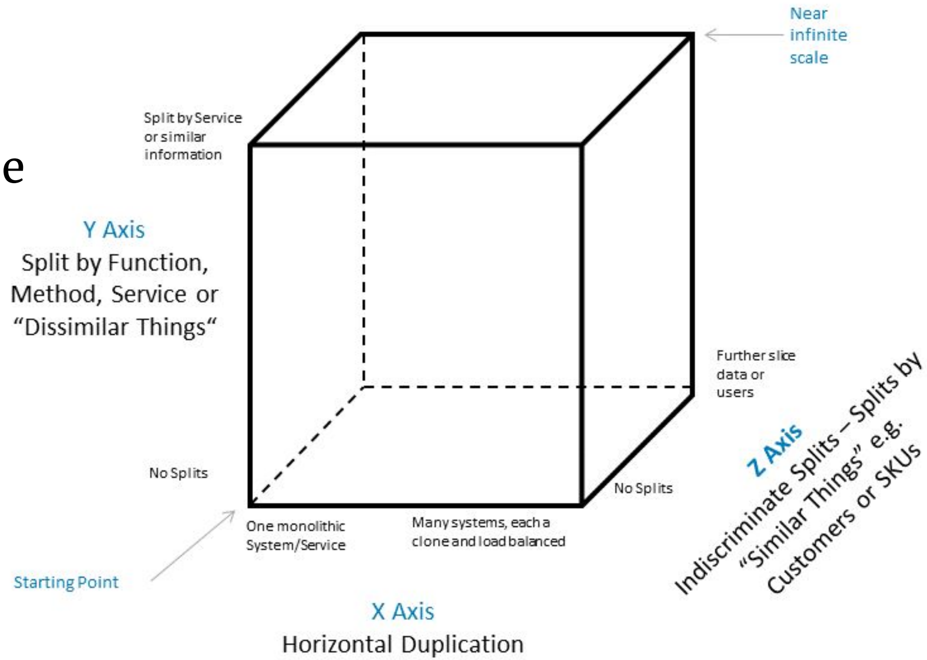
Did Not Work Well:

- Confluent Kafka
- Dashboard: ELK Stack; i.e. Elasticsearch, Kibana, Logstash

Picassa

AKF Scale Cube

- **X-Axis**
  - Load balanced multiple instances for each microservice
- **Y-Axis**
  - Microservices structure to decouple functionalities
- **Z-Axis**
  - Database sharding for Reactions, Payments service


Near infinite scale
Split by Service or similar information
Y Axis
Split by Function, Method, Service or "Dissimilar Things"
Further slice data or users
No Splits
No Splits
Z Axis
Indiscriminate Splits – Splits by "Similar Things" e.g. Customers or SKUs
One monolithic System/Service
Many systems, each a clone and load balanced
Starting Point
X Axis
Horizontal Duplication

Picassa

Demo [↗]

Picassa

Challenges

HEROKU

elasticsearch

kafka

- Confluent Kafka cluster on AWS
- ELK Stack (Elasticsearch / Logstash / Kibana)
- Deployment of frontend to Heroku
- MySQL Fabric for sharding

kibana

logstash

References

- Picasso GitHub Repository [Link]
- Picasso Frontend App Deployment on Heroku [Link]

Q&A

Thank You! :)