RESTaurant Reservation

Cloud Computing Project 2021

- Leonardo Emili
- Alessio Luciani
- Andrea Trianni
- Emanuele Mercanti

Agenda

01

Introduction

- Task and goals
- Technology stack

02

Architecture

- Microservice Decomposition
- UserAuth
- RestaurantAuth
- RestaurantData
- Booking
- Web App

03

Testing

- Unit Test
- Integration Test
- Scalability

Introduction

Task and goals

Technology stack

Task and Goals

- Allow users to reserve a table for lunch/dinner
- Pick the best place where to eat (i.e. ratings-based system)
- Allow users to create a new profile to enter the system
- Separation between user and restaurant access

Technical challenges

- Prioritize security-related aspects (i.e. access tokens, password generation)
- High-availability
- Fault-tolerant system
- Business oriented system (i.e. promote elasticity to match user demand)
- Seamless integration of different system components

Technology Stack

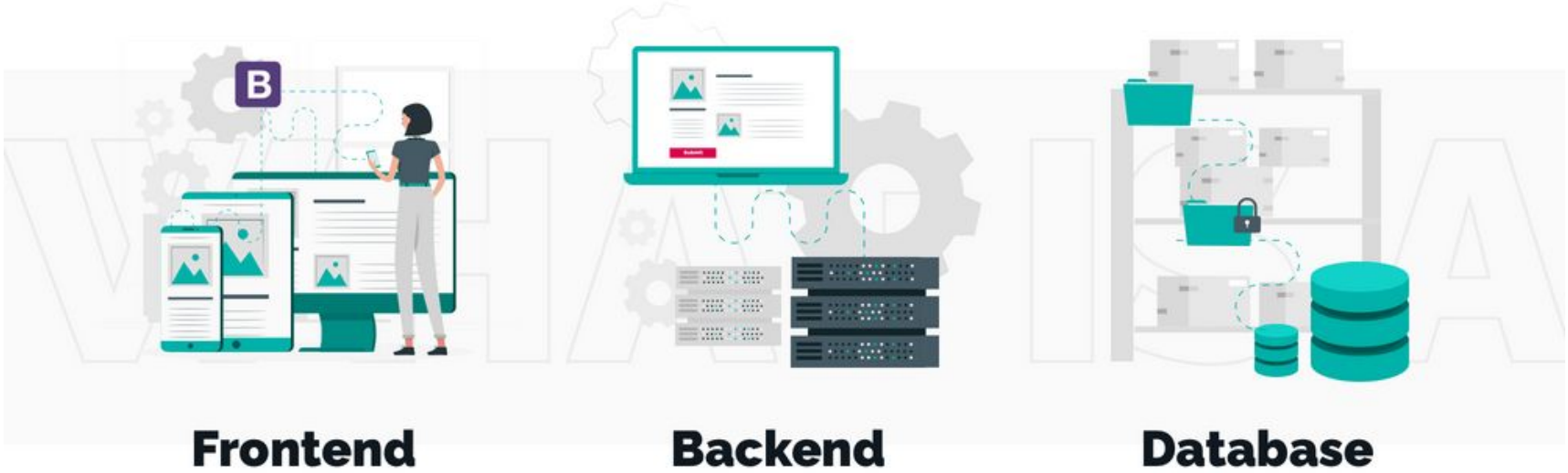


Image taken from [link]

Technology Stack - Backend

- Docker for OS-level virtualization and Docker swarm as orchestration tool
- Technology independence across services
- Backbone of the system using Node.js and Flask
- Ad-hoc monitoring tools (i.e. cAdvisor, Prometheus, Grafana)
- Stress test using Pumba
- Minor scripting using Python

Technology Stack - Database

- NoSQL approach to scale horizontally
- Document-oriented database using MongoDB
- Dedicated databases to each service

Technology Stack - Frontend

- Simple showcase application built using HTML and CSS
- Vue.js as frontend framework (i.e. Javascript)
- Ajax for asynchronous web requests
- Localstorage to store key/value pairs in a web browser

Architecture

Microservice Decomposition

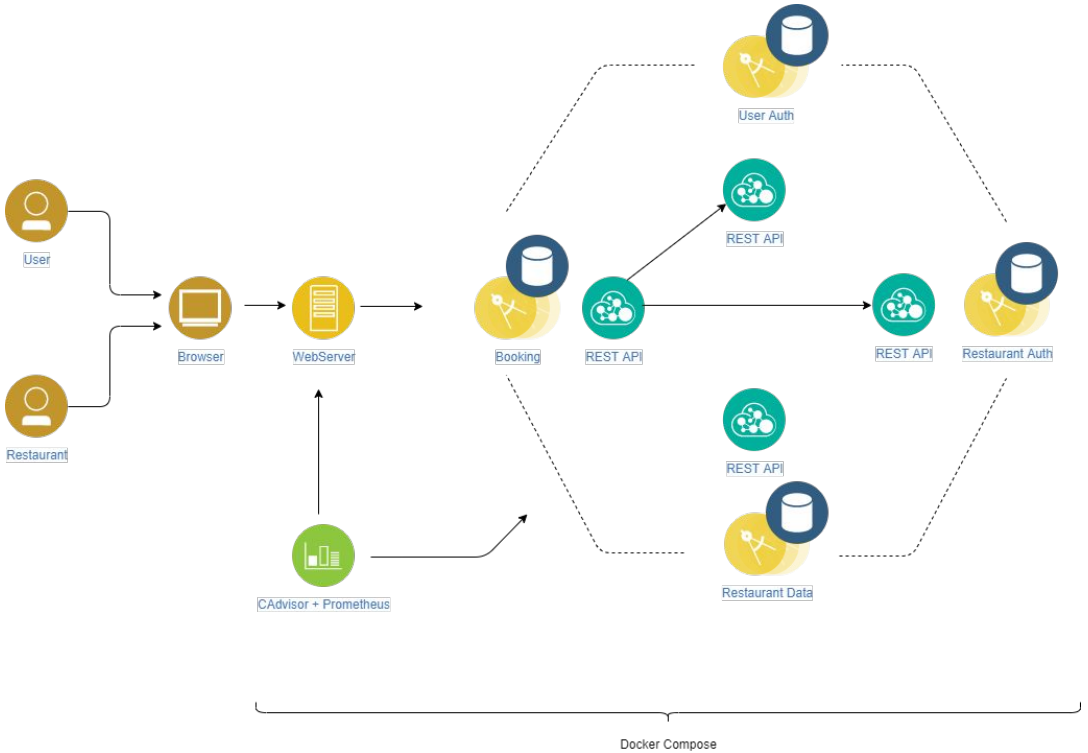
User Auth

Restaurant Data

Booking

WebApp

Microservice Decomposition



User Auth: overview

What does it do?

- Exposes an interface for authentication needs of the user
- Provides identity validation for operations that need session validation

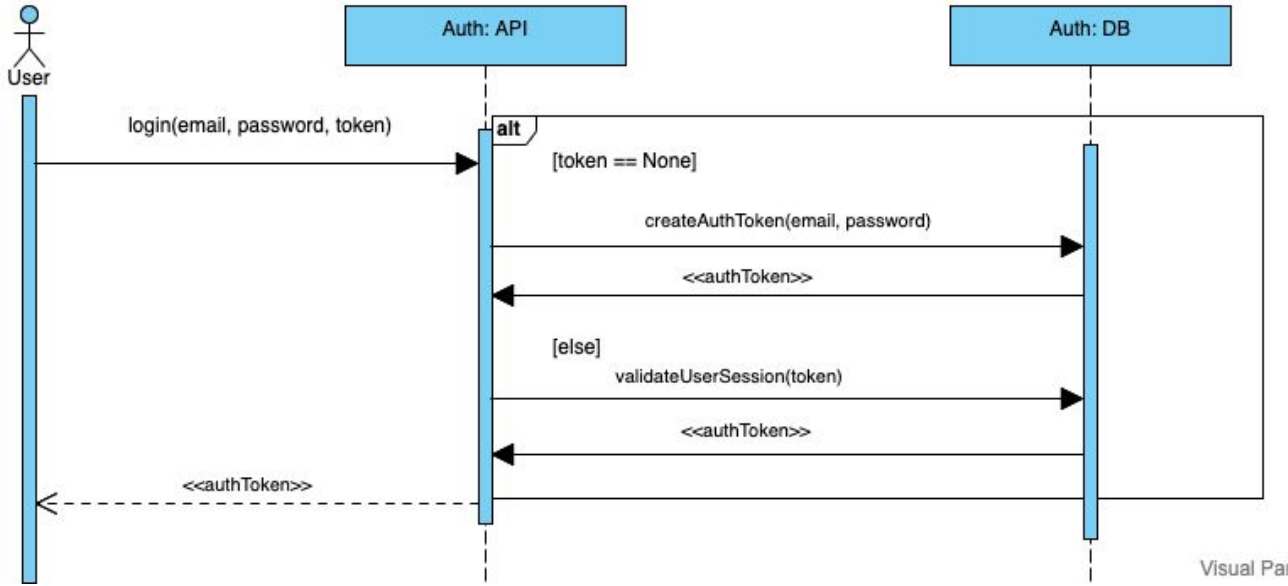
Functionalities

- Login
- Registration
- Session validation

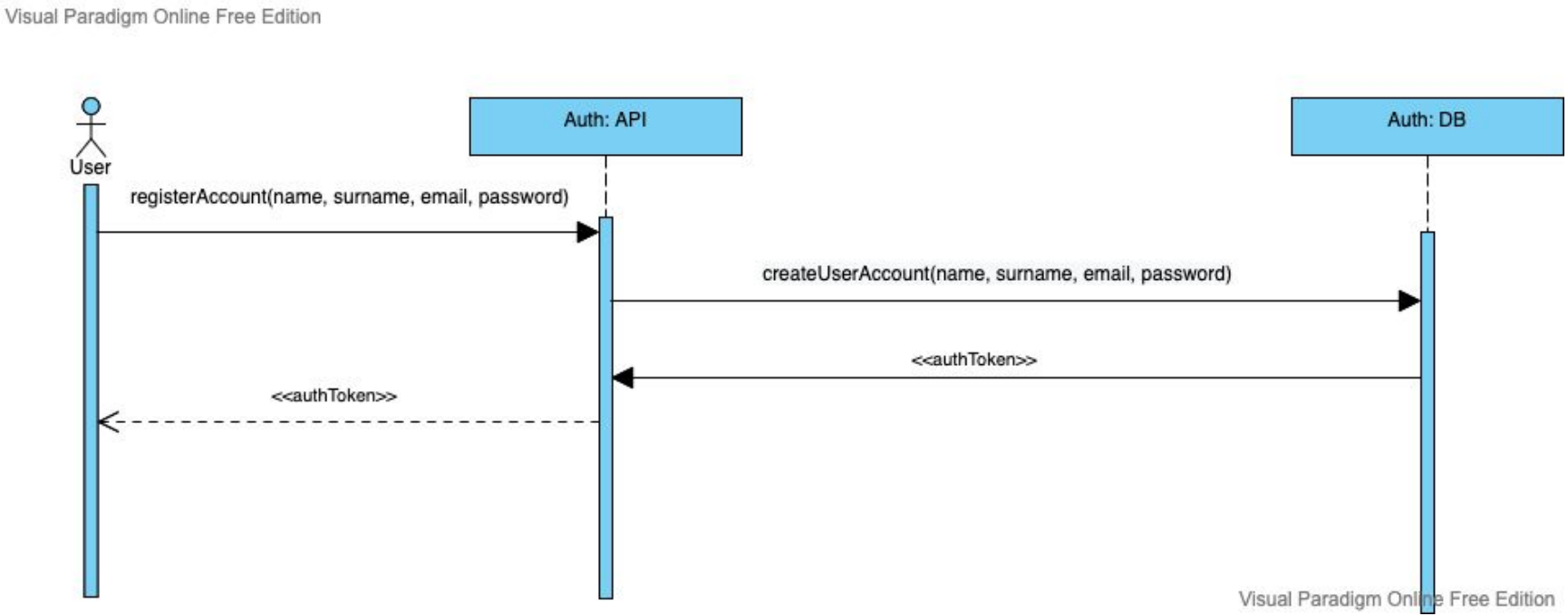
Used technologies

- Typescript on Node.js
- MongoDB

User Auth: login



User Auth: registration



Restaurant Auth

What does it do?

- Exposes an interface for authentication needs of the restaurant
- Provides identity validation for operations that need session validation

Functionalities

- Login
- Registration
- Session validation

Used technologies

- Typescript on Node.js
- MongoDB

Restaurant Data: overview

What does it do?

- Exposes an interface to search among different restaurants

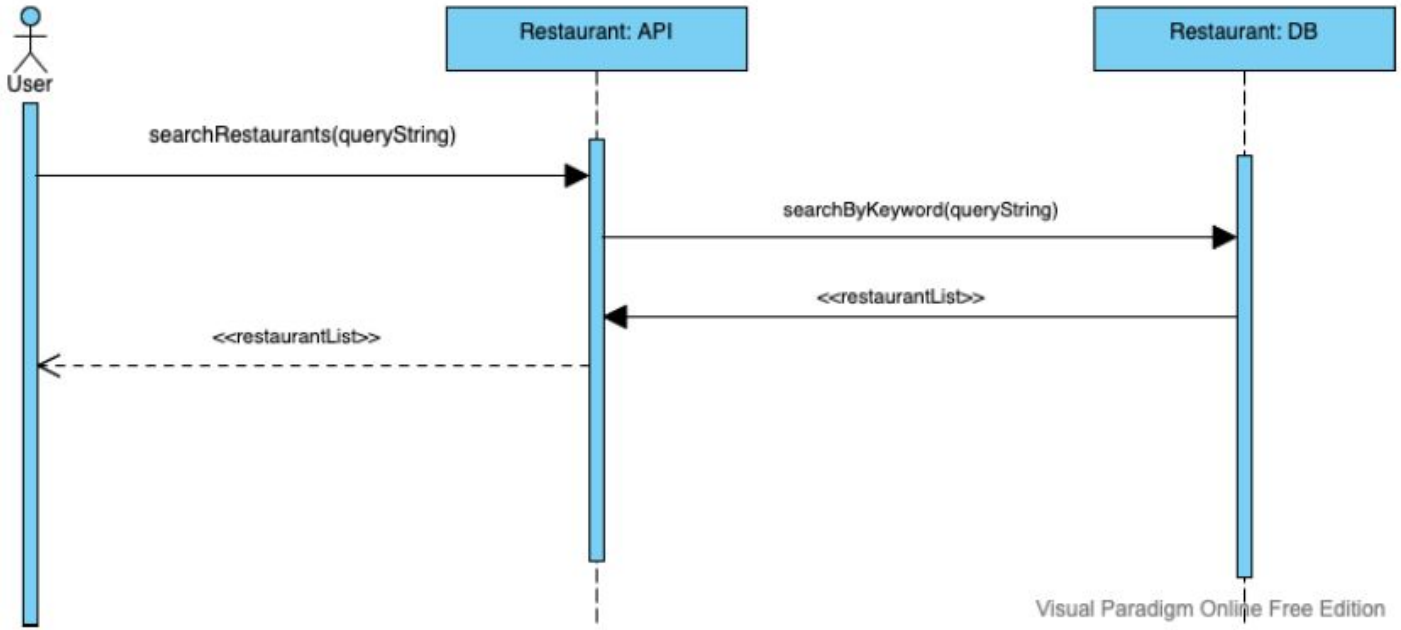
Functionalities

- Users search one or more restaurants based on a text string (name, item on menu, address..)

Used technologies

- Python
- MongoDB

Restaurant Data: search



Booking: overview

What does it do?

- Takes care of the tables reservations
- Used by both users and restaurants users

Functionalities

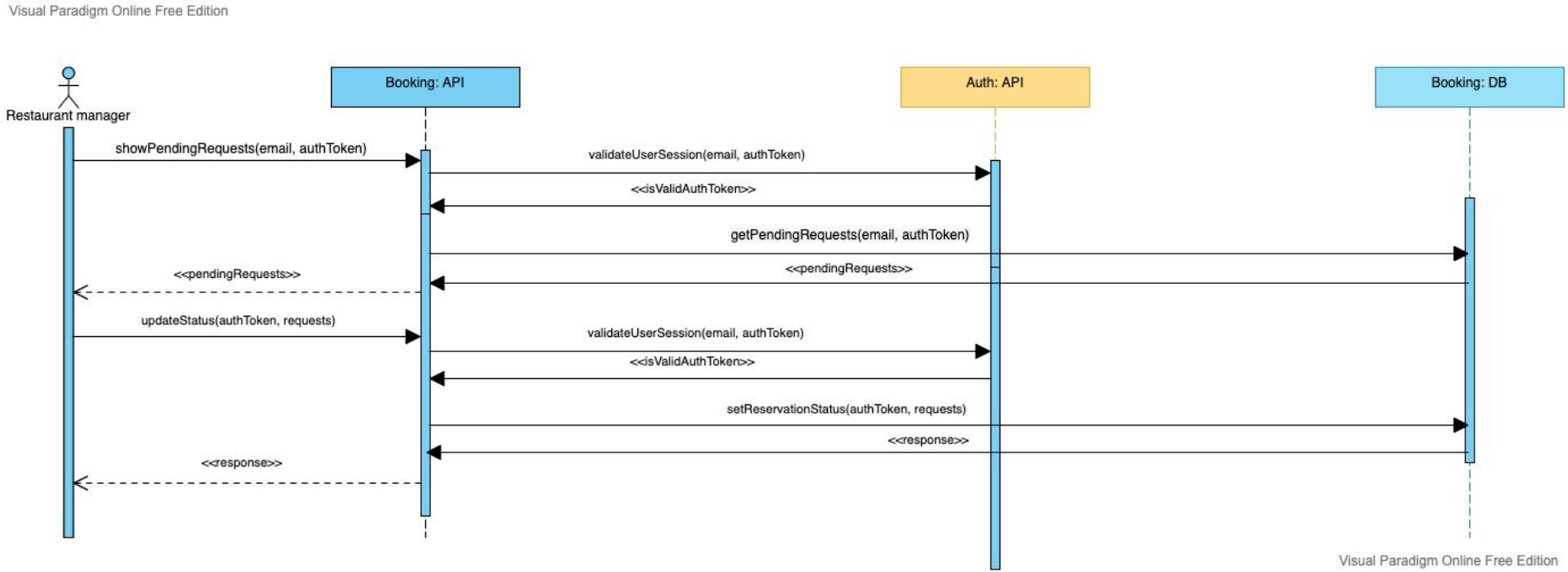
- User send a request for a reservation to a restaurant, restaurant accepts/refuses reservations
- Retrieves the list of reservations for a user/restaurant

Used technologies

- Python
- MongoDB

Booking: Reserve

Booking: Change status



Additional features

Possible microservices to add in the future:

- Reviews: allow users to leave a review and a rating
- Promo Manager: allow restaurants to issues promotions
- Notification manager: sends emails to users and restaurants to notify of a reservation

Web App

Web App

- Simple overview over the whole system
- Modular system (e.g. multiple views and Vue components)
- Enforce type checking using typescript
- Thin client approach

Web App

- Passwords are not stored on the client-side
- Random generation of authentication tokens is performed on the server-side
- Silent sign-in using authentication tokens (session remains open despite the browser status)
- Allow restaurant owners to accept/reject booking requests

Welcome to the explore page!

risto

ristorante buono

ristorante etnico

ristorante cattivo

Testing

Unit Test

Integration Test

Scalability

Unit Test

- Test microservices individually using simple REST calls
- Use example data to test the single functionalities

Integration Test

- Test microservices together while running in Docker Compose
- Access client webapp and test the overall functioning of the system

Scalability

- Assess scalability of the system as the load changes
- Implement an auto-scaling mechanism that acts in response to resource usage changes
- Scale the docker service by adding and removing microservice instances

Scalability - Auto-scaler

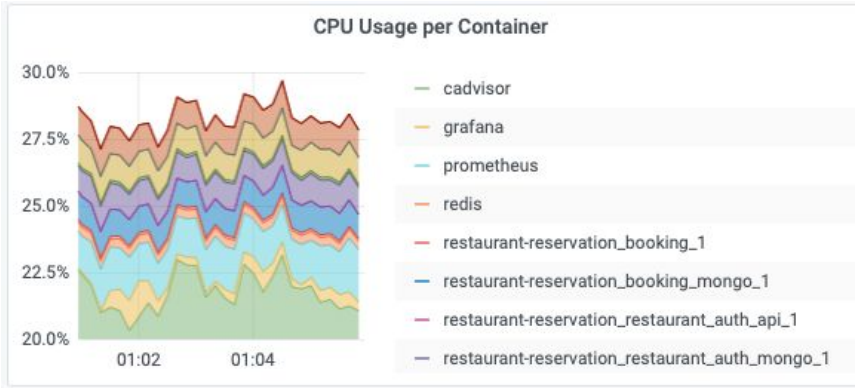
- Monitor the complexive CPU usage of all instances of a microservice with a fixed cadence using “docker stats”
- If the usage goes above a certain threshold, scale the service up with “docker-compose scale”
- If it goes below a lower threshold, scale the service down with “docker-compose scale”

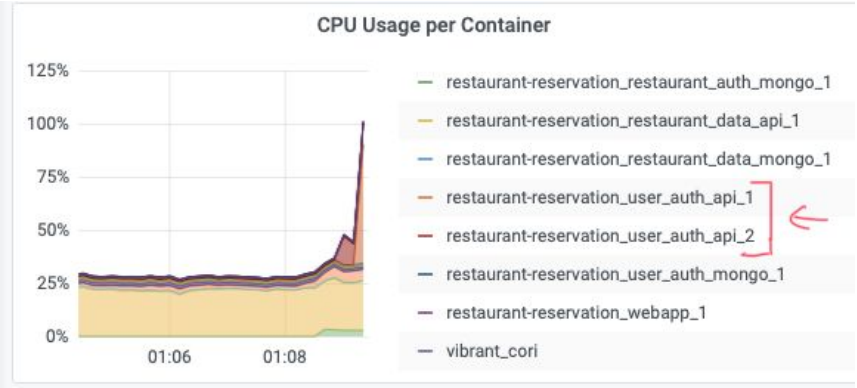
Scalability - Monitoring tools

- cAdvisor is used to extract metrics from the containers
- Prometheus is used to query cAdvisor and collect the necessary metrics
- Grafana is used to show the metrics on charts and dashboards interactively
- Metrics are temporarily stored on Redis

Scalability test

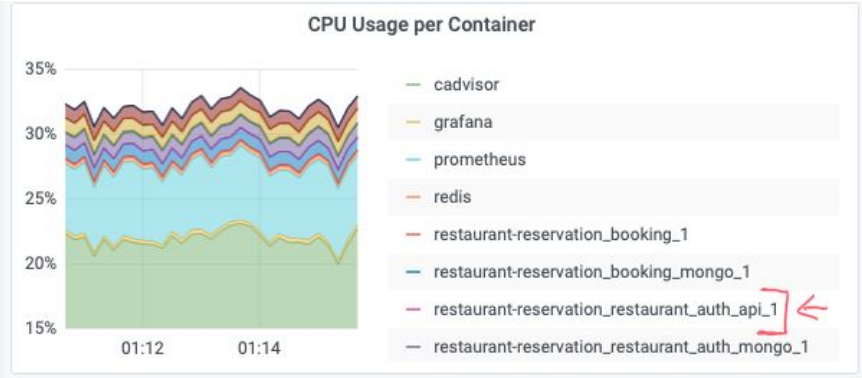
- Pumba tool used to leverage the Linux stress-ng mechanism and send workloads to Docker containers

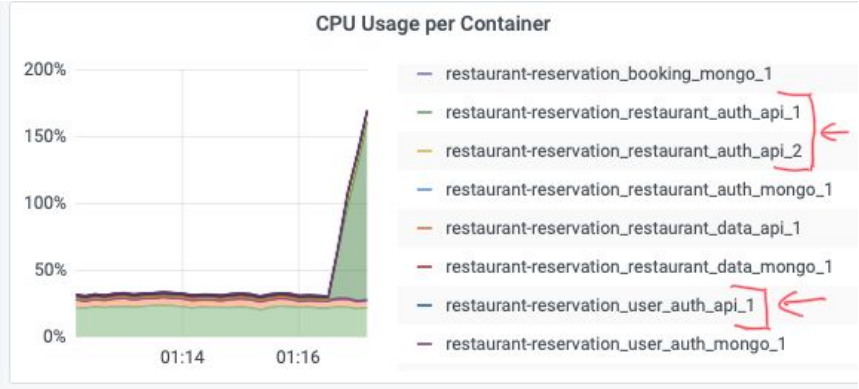




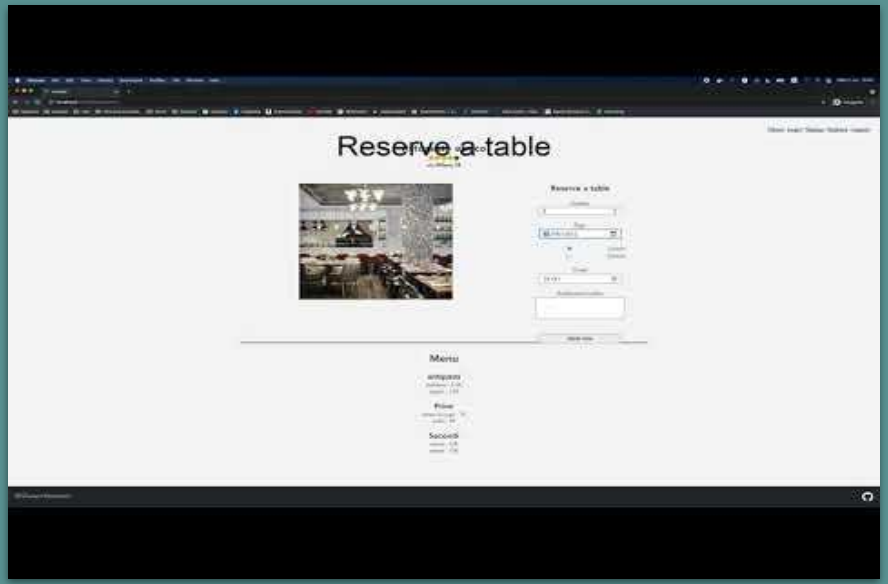
Scalability test (2)

- Scale down User Auth and scale up Restaurant Auth





Live Demo



Thank you