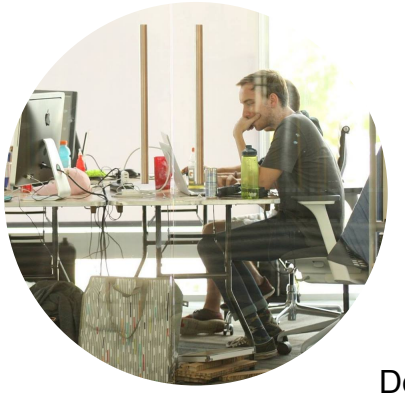
Expresso

Team 17-34

Advisor: Bill Adamowski

Team Members

Jonny Krysh



Lucas Collins

Jake Long

Derek Yu

Garret Meier

Overview

Project Goal and Scope

Infrastructure & Design Overview

Backend

Frontend

Completed Work

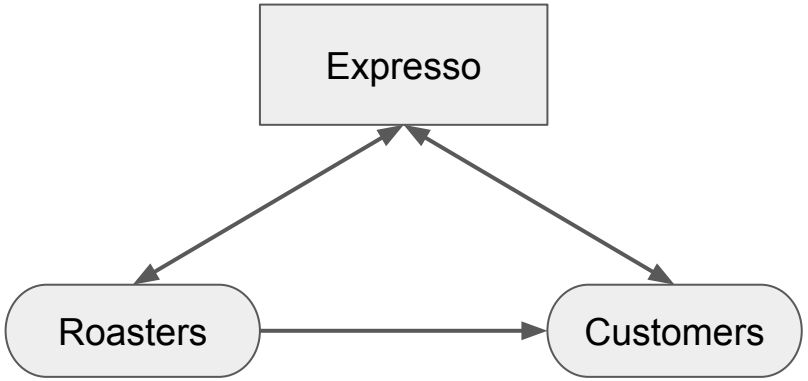
Overview
Scope

Coffee subscription

Target local roasters

Quick delivery

Easily customizable



Overview
Goals

Utilize good architecture patterns

Build reliable distributed systems

Allow:

- Roasters to post and fulfill orders
- Customers to subscribe and receive beans

Overview
Design Decisions

Decentralized

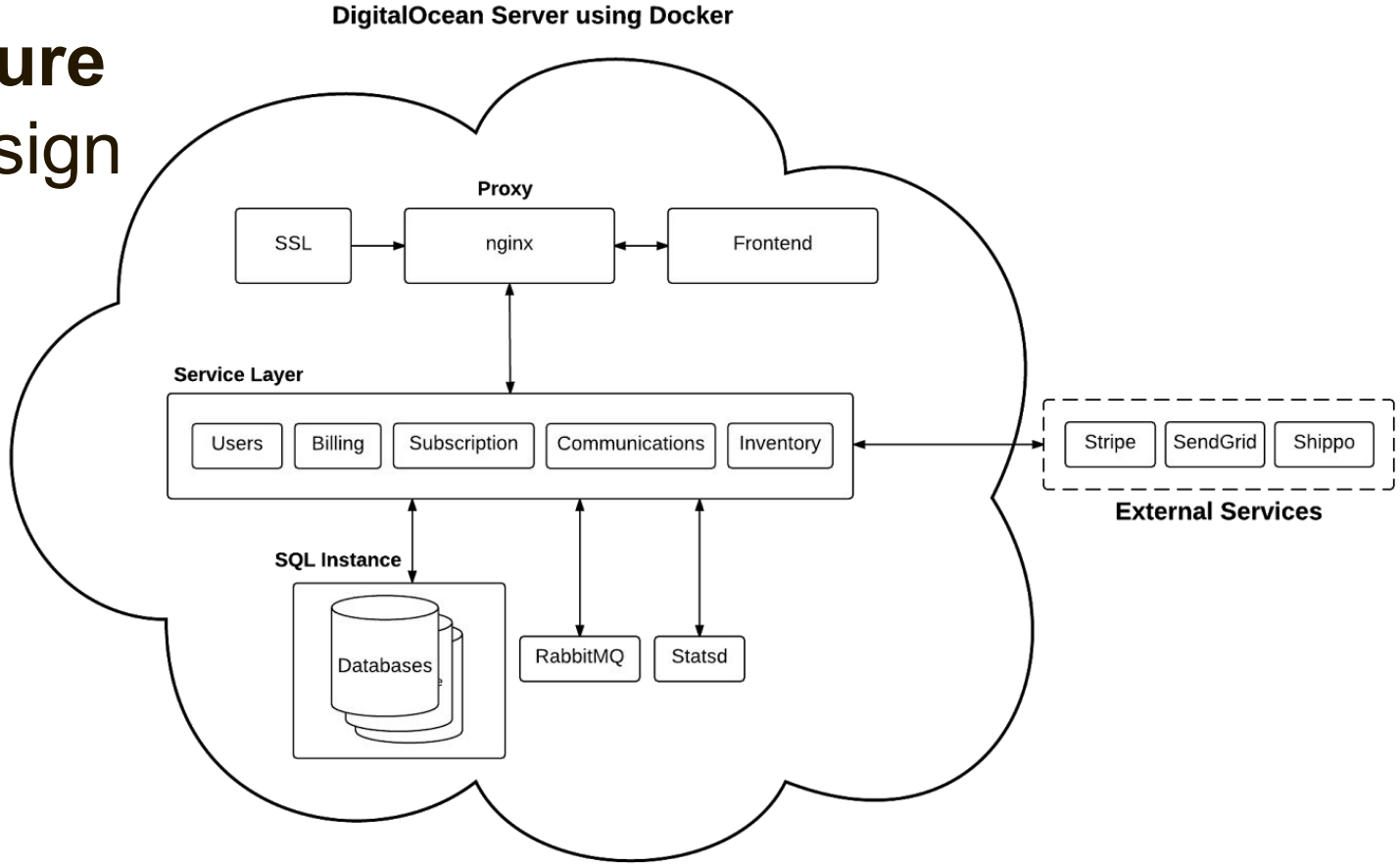
- Beans will be fresher when shipped
- Saves time and money by not needing to ship to middle-man warehouse

Microservice

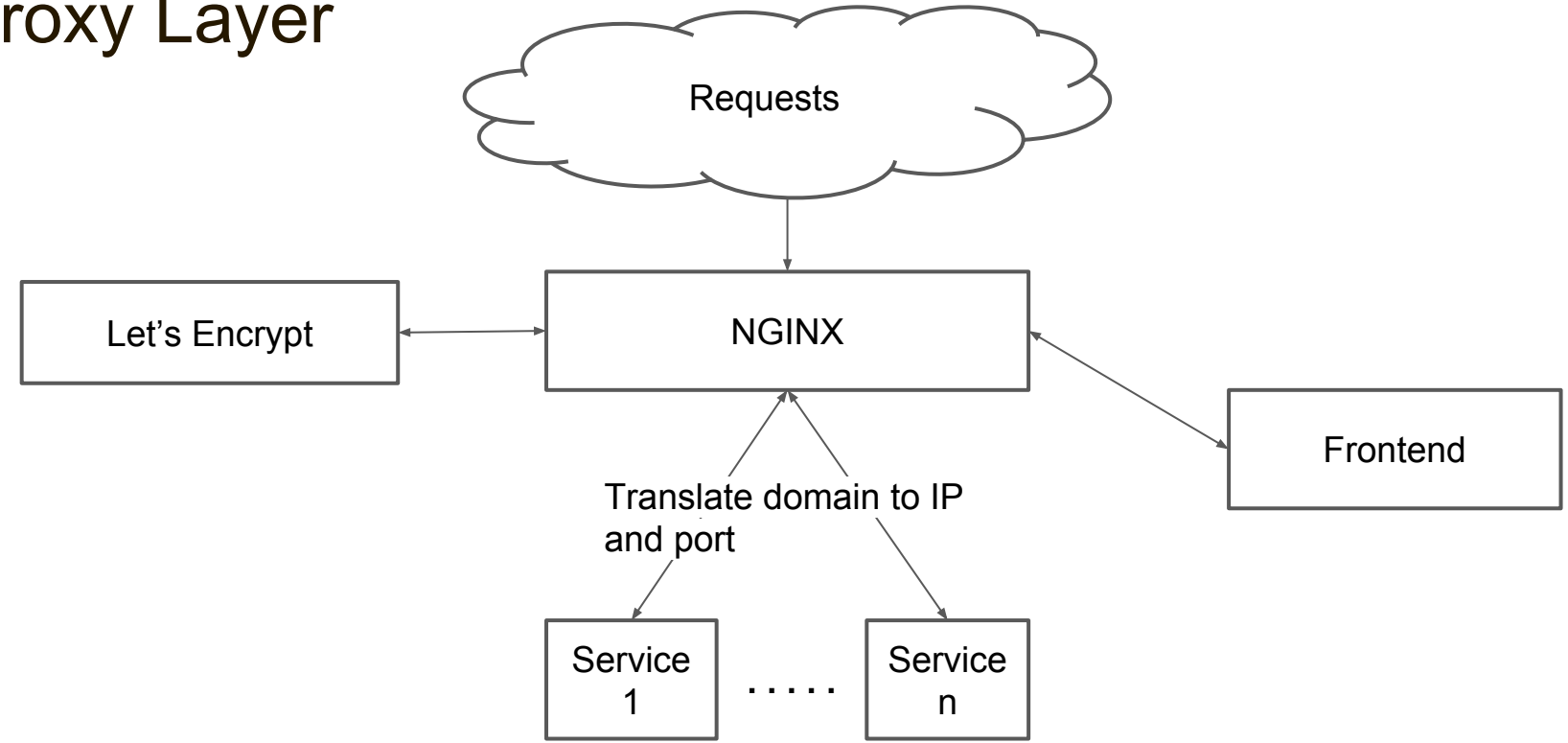
- Easily Scalable
- Easier to deploy
- Easier to split work between team members

Infrastructure
Current Design

Infrastructure
Current Design

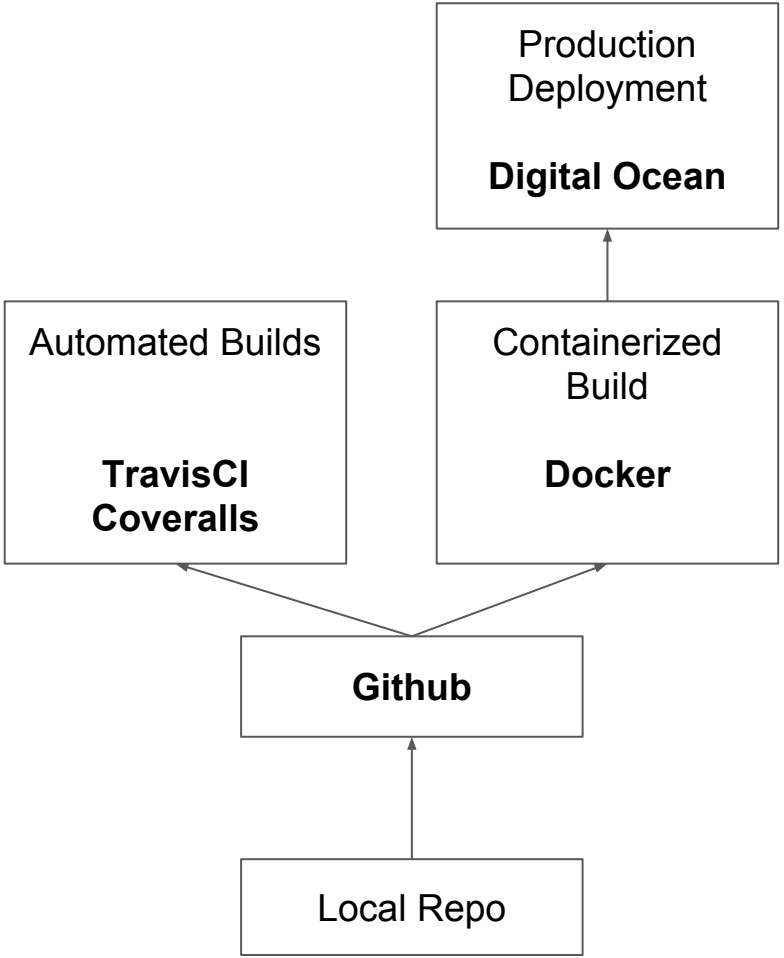


Infrastructure
Proxy Layer



Infrastructure
Deployment

1. Commit Locally
2. Push Updates
3. Test build and get coverage
4. Build production image
5. Deploy.



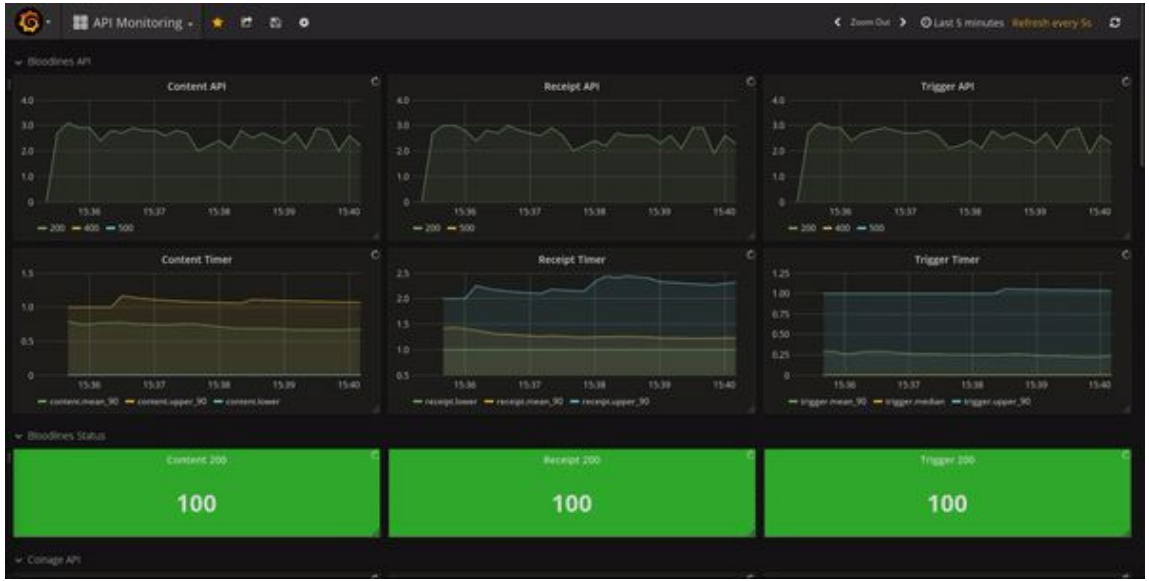
Infrastructure
Testing

Challenges:

- Integrating separate services
- Monitoring service behavior

Goals:

- Obtain useful metrics
- Display service errors

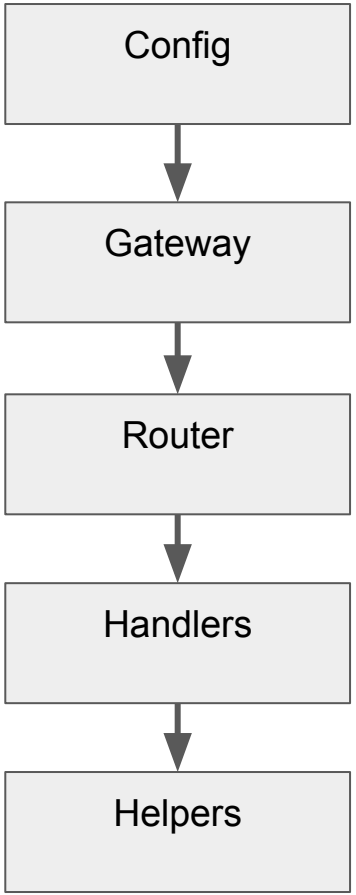


Backend
Service Layer

Backend
Service Layer

Backend
Architecture

1. Leverage microservice
2. Route requests
3. Handle requests
4. Integrate database manipulation per request



Backend
Testing

Mock Tools

- mockery
- testify
- httpmock
- sql-mock

Primary Targets

- Gateways
- Helpers
- Routers

General Tools

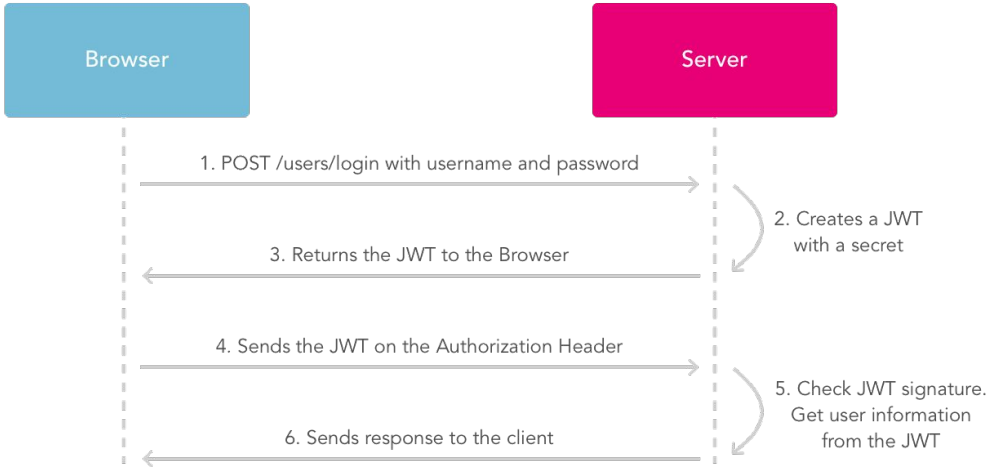
- Postman

Backend
Challenges

- Code reuse
  - Each service needs similar boilerplate code
- Integration testing
  - Needs mocks for testing
  - Docker containers
- Dependency management
  - Services depending on each other

Backend
Security

- Passwords hashed with bcrypt
  - Adaptive for hardware improvements
- JSON Web Token (JWT)
  - Sent in header of HTTP request
  - Stores information about the user
- Payment information
  - Use Stripe to manage billing information



Source: https://jwt.io/introduction/

Frontend
Architecture

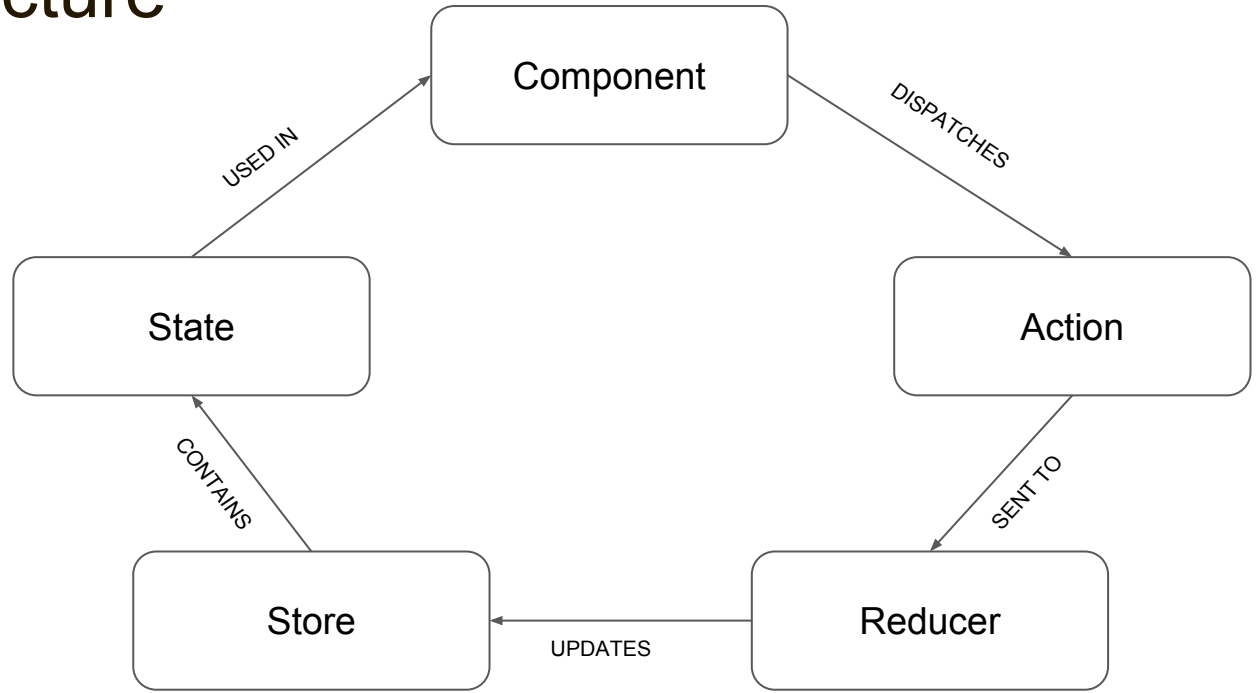
Frontend
Architecture

Used **CreateReactApp** React starter pack for environment set-up

NodeJS with **Webpack** handles all configuration and building / bundling

React, **Redux**, and (React) **Router** for all views, logic, and app state

Frontend
Architecture



Frontend
Challenges

- Microservices can make it difficult to aggregate data on the frontend
  - Multiple requests for one set of data
  - Fixed on server side
- Consistent styling with Tachyons library
  - CSS Framework for class names as styles
- Login authentication and persistent sessions

Completed Work

23092 lines of code

(**16249** of Go and **6843** of JS)

1129 git commits over **6** repos

79 closed PRs

66 resolved issues

Completed Work

- ☑ Utilize modern architecture practices
- ☑ Build reliable distributed systems
- ☑ Allow Roasters to post and fulfill orders
- ☑ Allow Customers to subscribe and receive beans
- ☑ Build Minimum Viable Product

expresso.store