Chris Bookstore

Group Members
Edwin Tok Wei Liang
Ezekiel Ong Young
Jerome Goh Ting Chuan
Miguel Alonzo Ortega
Tay Wei Jie

Table of Contents

01

The Problem

The Problem

Bookstores in Singapore are declining

Prices of books are too expensive

Lack of books online

02

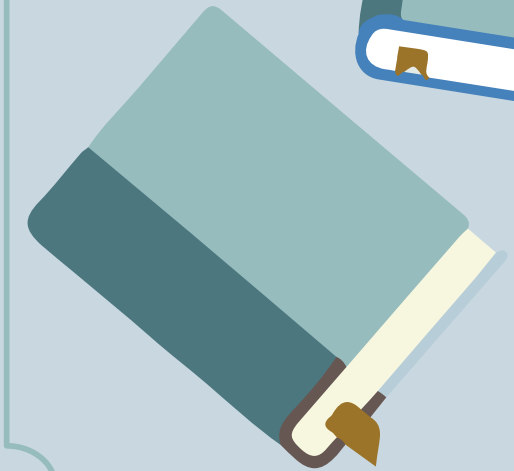
The Solution

The Solution

Bookstore

A place for users to buy books from the convenience of their homes

Atomic Microservices



Users Service

Login/Logout and Registration

Products Service

Product data and availability

Orders Service

Buying books

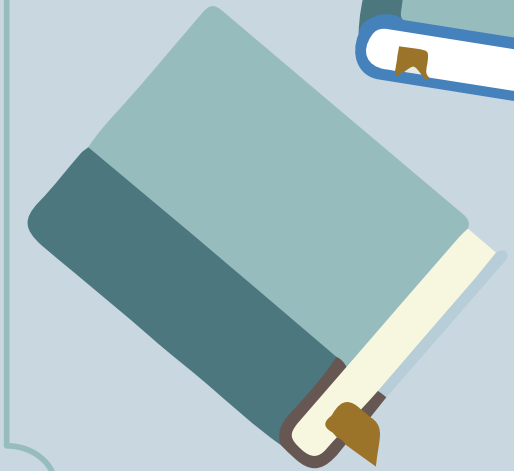
Notifications Service

Price-drop and order notifications

Wishlist Service

Storing wishlist

Composite Microservices



Place-order Service

Creation of orders

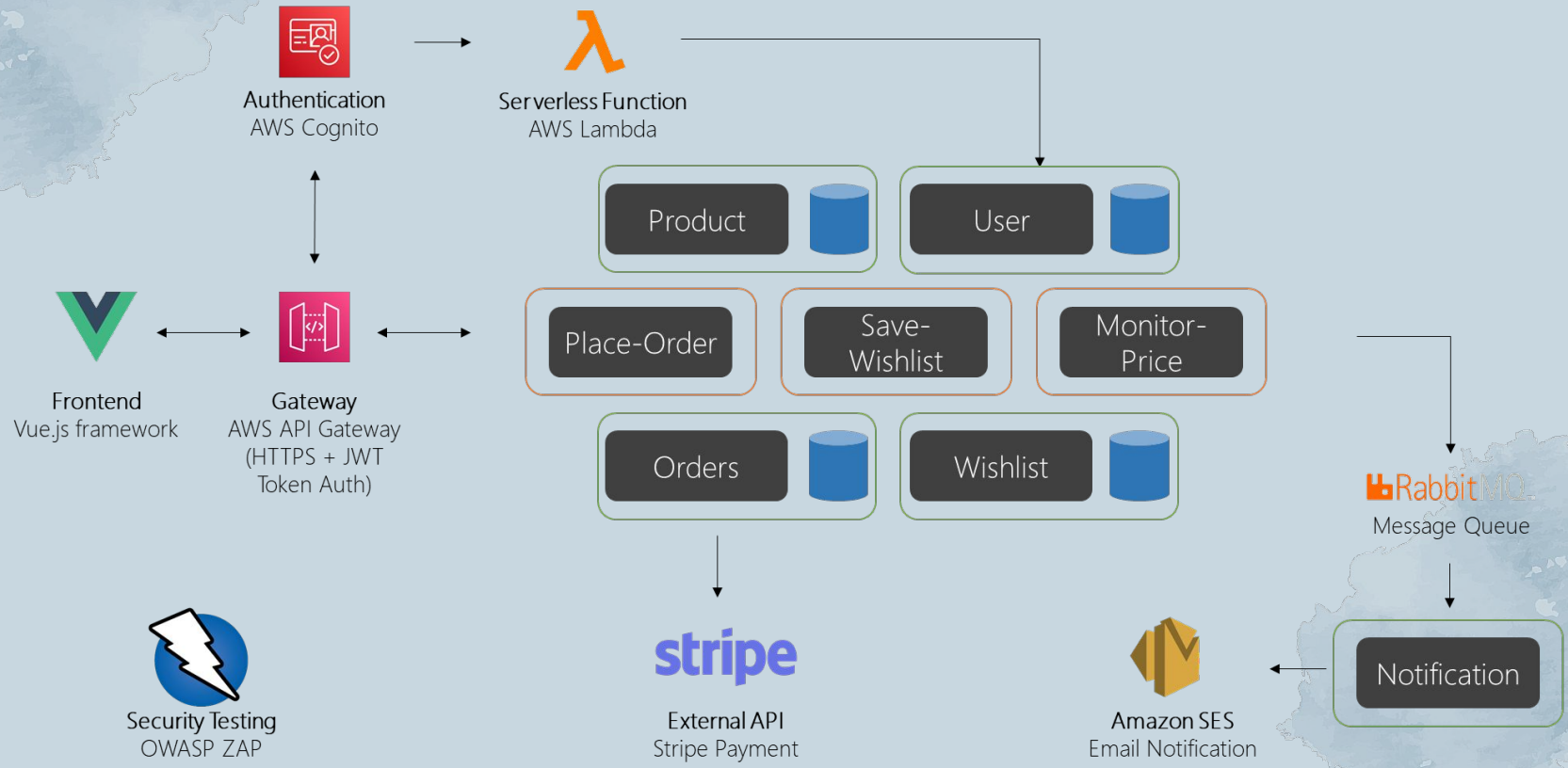
Save-wishlist Service

Saving wishlist

Monitor-price Service

Monitoring price changes

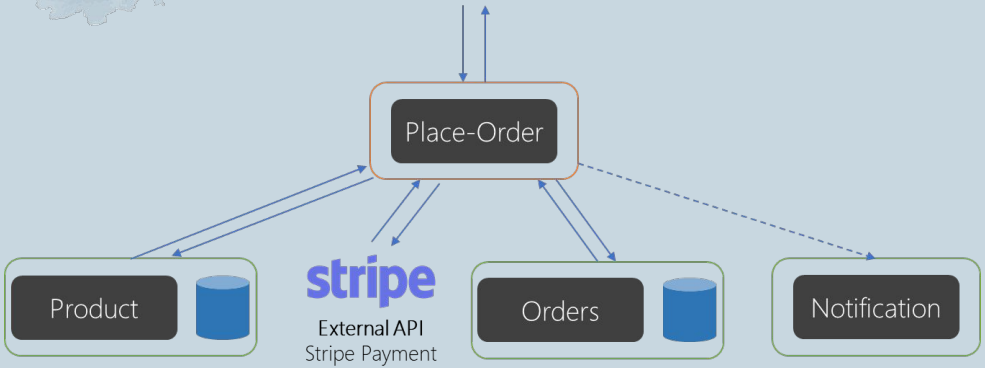
Overall Framework



03

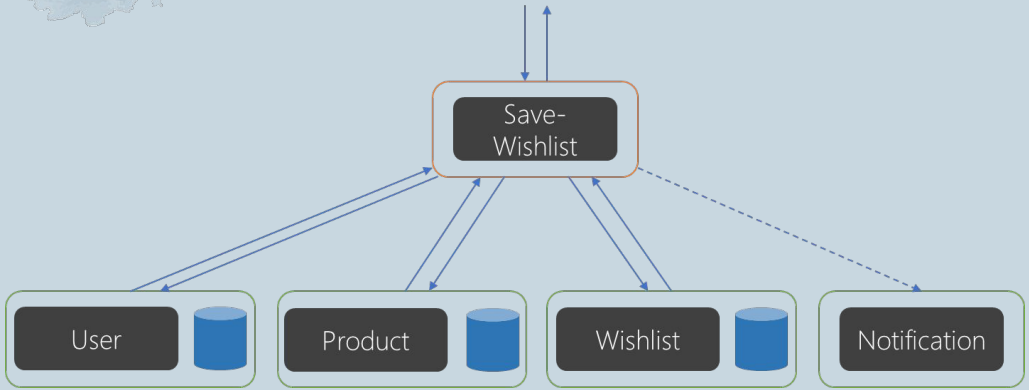
Key Scenarios

Ordering a product



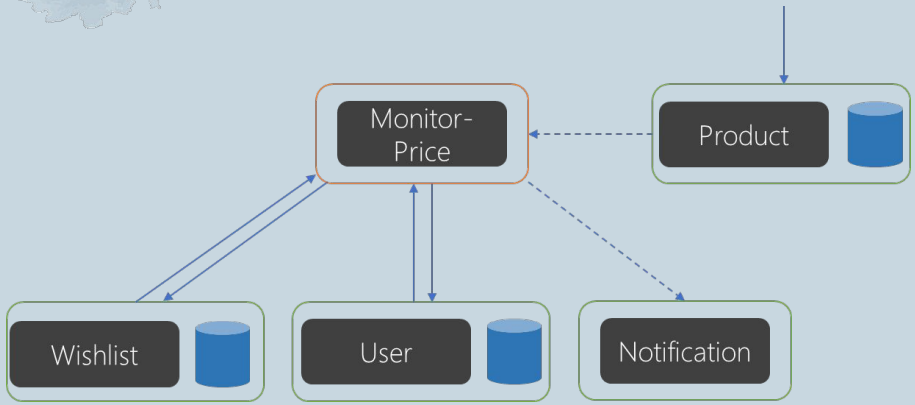
1. Users place an order on the frontend
2. Place-order service checks the product service for product availability
3. Place-order processes the payment using Stripe
4. If successful, the service creates the corresponding order in orders service
5. After which, notification service is informed asynchronously.

Making a wishlist



1. User selects to add book to wishlist in the frontend
2. Save-wishlist checks if user exists
3. Save-wishlist checks if book exists
4. If product and user is valid, book is added to wishlist
5. After which, notification service is informed asynchronously.

Notifying price drops



1. Book price is dropped using the product service
2. Product service sends a message to the monitor price service
3. The monitor price service contacts the wishlist service for the list of users that added the book
4. It also contacts the user service for the emails corresponding to the user IDs
5. Finally, it sends notifications to each of those emails

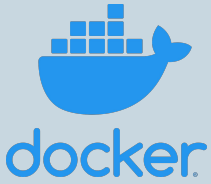
04

DevOps Practices and Tools

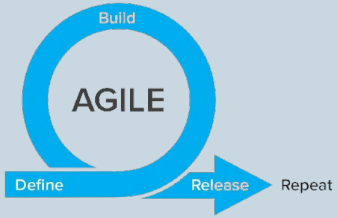
What DevOps practices we applied and what tools did we use to do these?

DevOps Practices

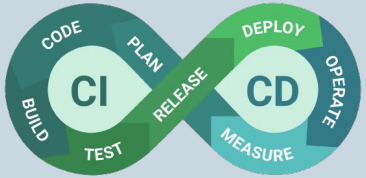
Utilisation of Microservices



Containerisation of Microservices



Agile Development



Continuous Integration and Deployment

DevOps Practices

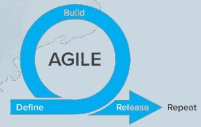
Utilisation of Microservices

- Decoupling of architecture
- Reusing of services - notifications
- AMQP and HTTP communications

Containerisation of Services

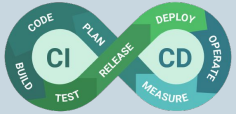
- Use of Dockerfiles and docker-compose for simple setup
- Portability and easier management of environments/dependencies
- Faster delivery and improved security

DevOps Practices



Agile Development

- Weekly sprint meetings to update progress and plan the next sprint
- Iterative and incremental sprints for developing microservices and UI
- Utilized agile tools such as kanban to keep track of progress



Continuous Integration and Deployment

- Early setup of CI/CD using Gitlab and AWS ECS to test changes
- Identify bugs and compatibility issues early to allow for a smoother development and subsequent release

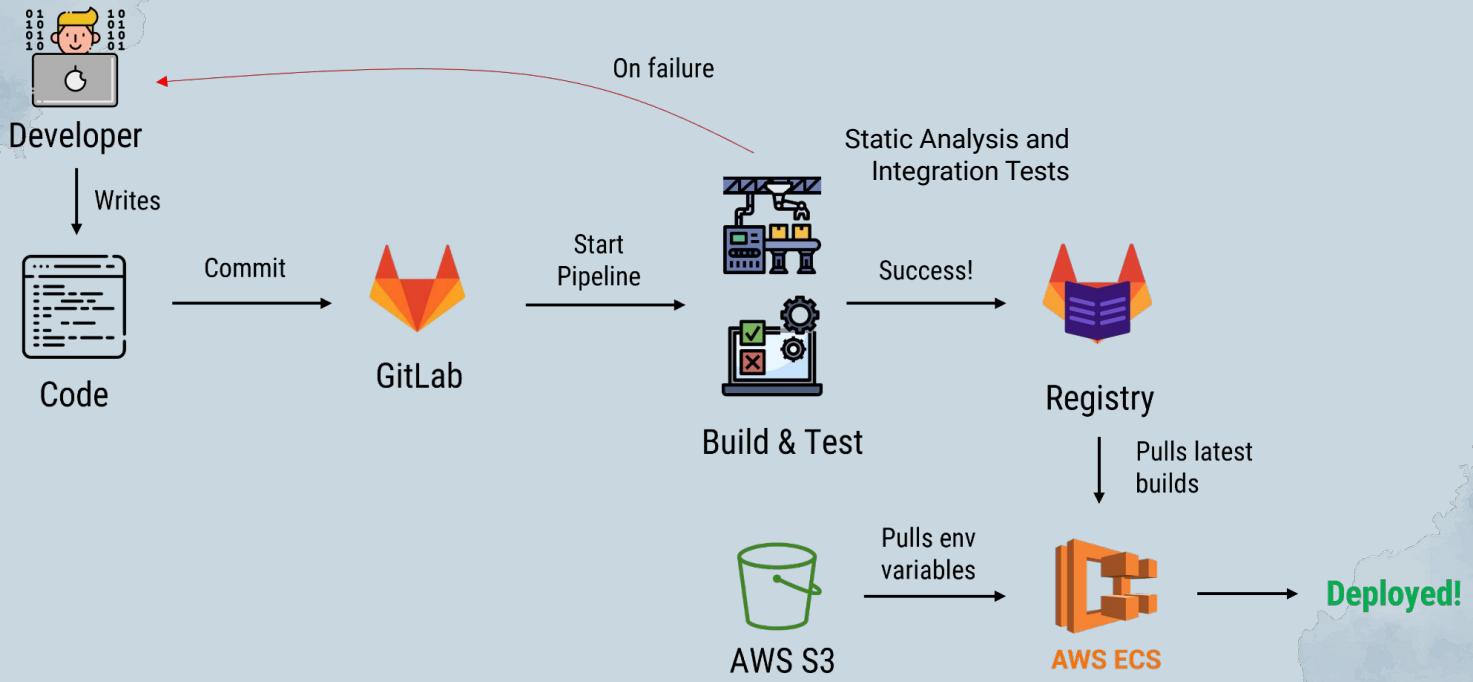
DevOps Tools

GitLab

Docker & Docker Compose

AWS ECS

CI/CD Pipeline



05

Demo

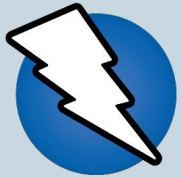
Watch us rock the house

06

Research

Bonus Features

Bonus Features



OWASP ZAP

Security Testing

AWS SES

Notification Service

https://

JWT Authentication

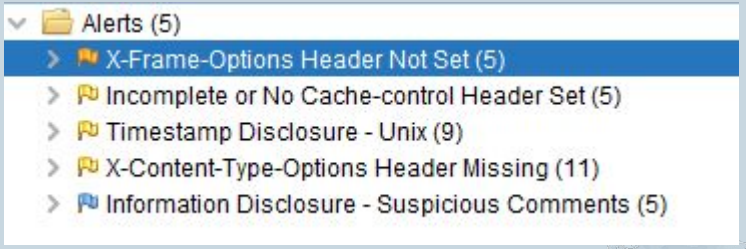
With AWS Cognito

Docker

End-to-End Testing

OWASP ZAP

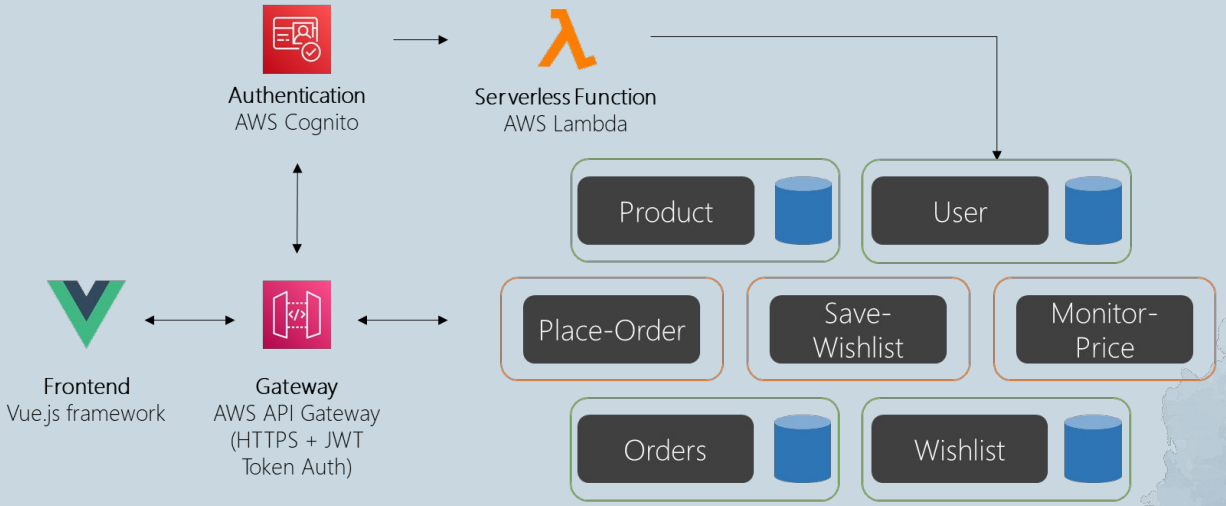
1. Performed basic level vulnerability assessment via OWASP ZAP attack mode scan
2. Yielded 5 alerts with below medium level risk including
3. Most of them were inaccurate upon further investigation
4. X-Frame-Option header was the only valid risk with potential vulnerabilities of clickjacking
5. Plan to remove vulnerability in next sprint :D



User Management

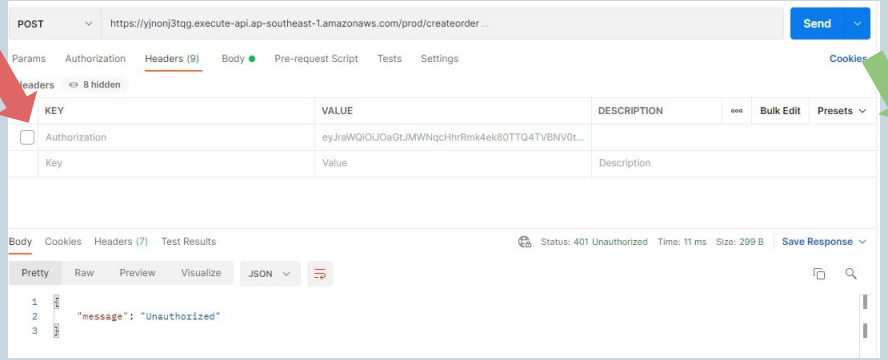
safe secure https://

- User management done with AWS Cognito User Pool
- AWS managed registration and signup
- Subsequent support for JWT token management with AWS API Gateway

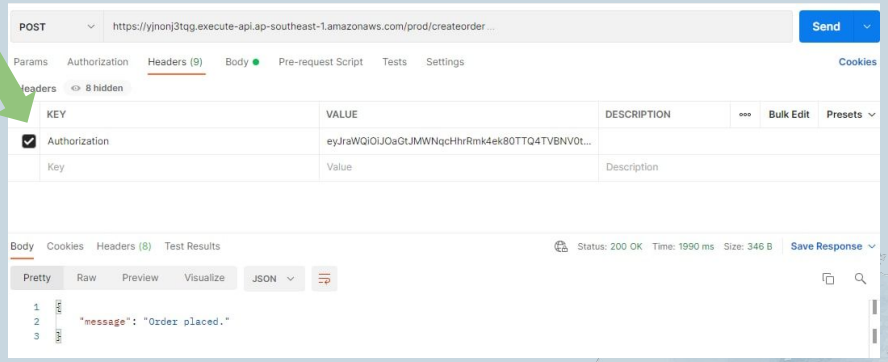


JWT Authentication

- Tokens issued by AWS Cognito
- Authentication is managed by AWS Gateway
- HTTPS provided via hosting with AWS Amplify (Provides CI/CD too :))



No Token Provided



Token Provided

AWS Simple Email Service

Amazon SES

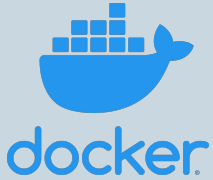
Notification service can send out emails to users upon events

- Confirmation of payments
- Price drop of wishlist items

Order confirmation

*AWS SES deployed on test environment, can only send to users registered in AWS SES

Docker Compose: End to End Testing



Fully automated end-to-end testing using docker compose to create the test environment and databases. Covering our 3 main use cases

Test Starting

```
order-test_1     | ============================= test session starts ==============================
order-test_1     | platform linux -- Python 3.10.0, pytest-6.2.5, py-1.11.0, pluggy-1.0.0
mysql_order_1    | mbind: Operation not permitted
order-test_1     | rootdir: /usr/src/app
order-test_1     | plugins: dependency-0.5.1
order-test_1     | collected 13 items
```

Test Results

```
order-test_1     |
order-test_1     | ============================== 13 passed in 23.23s ==============================
order-test_1 exited with code 0
```

Chris Bookstore

Group Members
Edwin Tok Wei Liang
Ezekiel Ong Young
Jerome Goh Ting Chuan
Miguel Alonzo Ortega
Tay Wei Jie