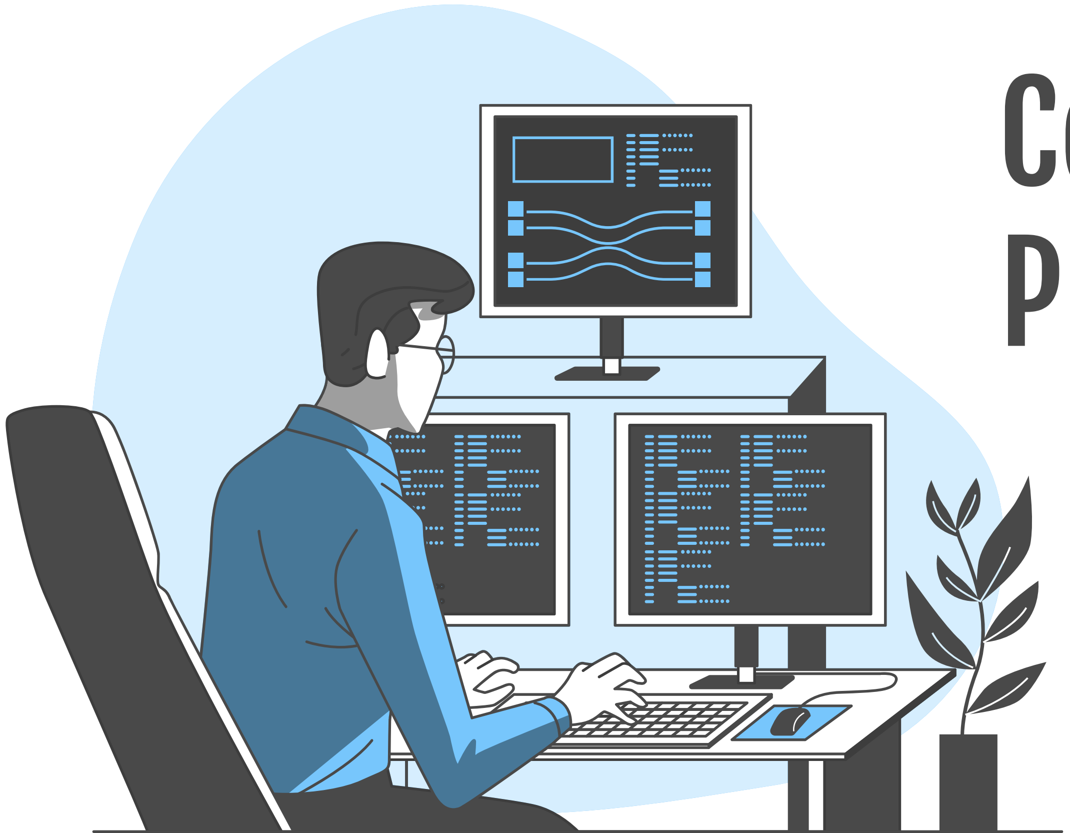


ComoClub Final Presentation

G2 Team 8

Bai He, Daniel, Jing Yi, Sean Jin, Timothy

Table of Contents

01
Key Use Cases

Use cases of our final application

Key Use Cases

01 Login

Verify if user exists in *Memberson CRM & AWS Cognito*

02 View list of experiences

List of experiences available along with its details *(7Rooms)*

03 Book an experience

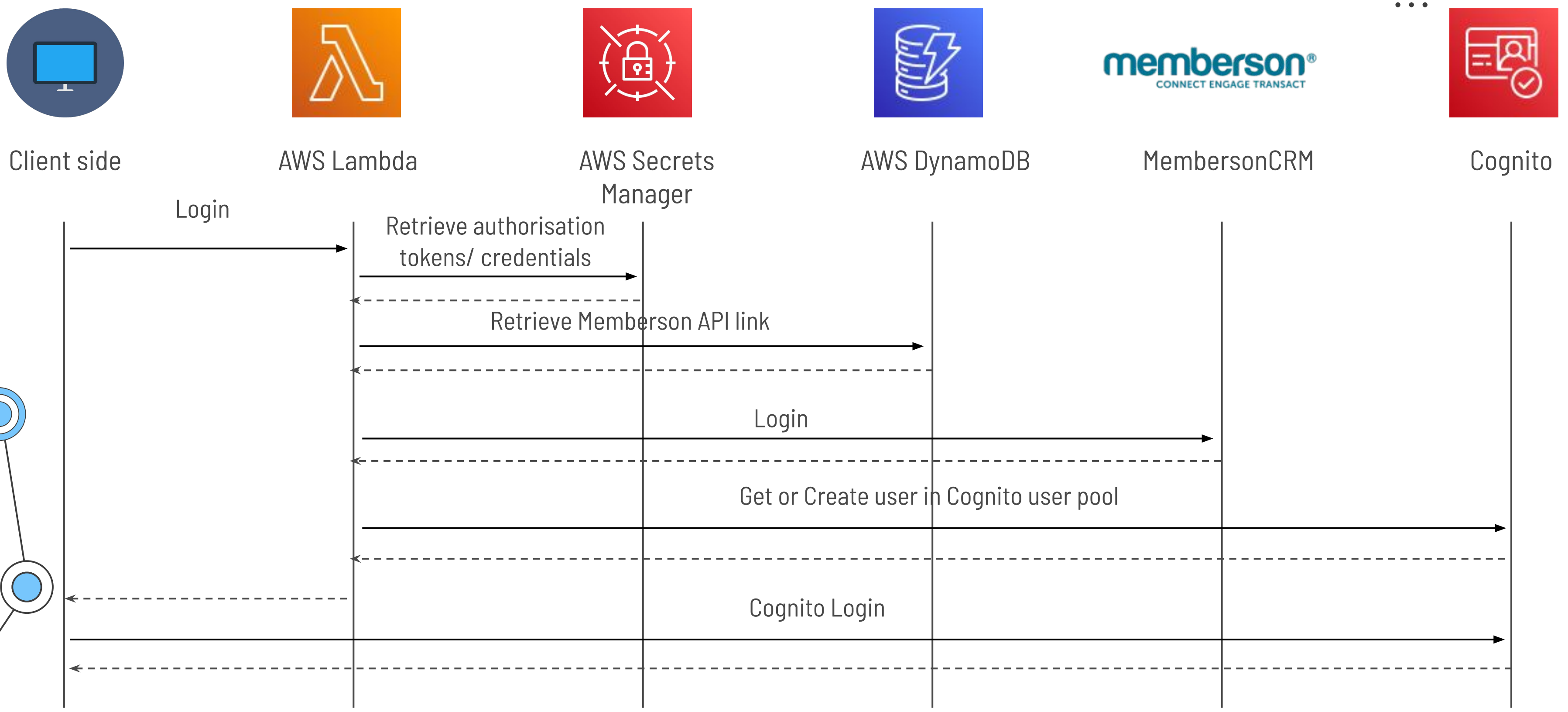
3.1 Payment by COMO Points *(Memberson CRM)*

3.2 Payment by Credit Card *(Stripe)*

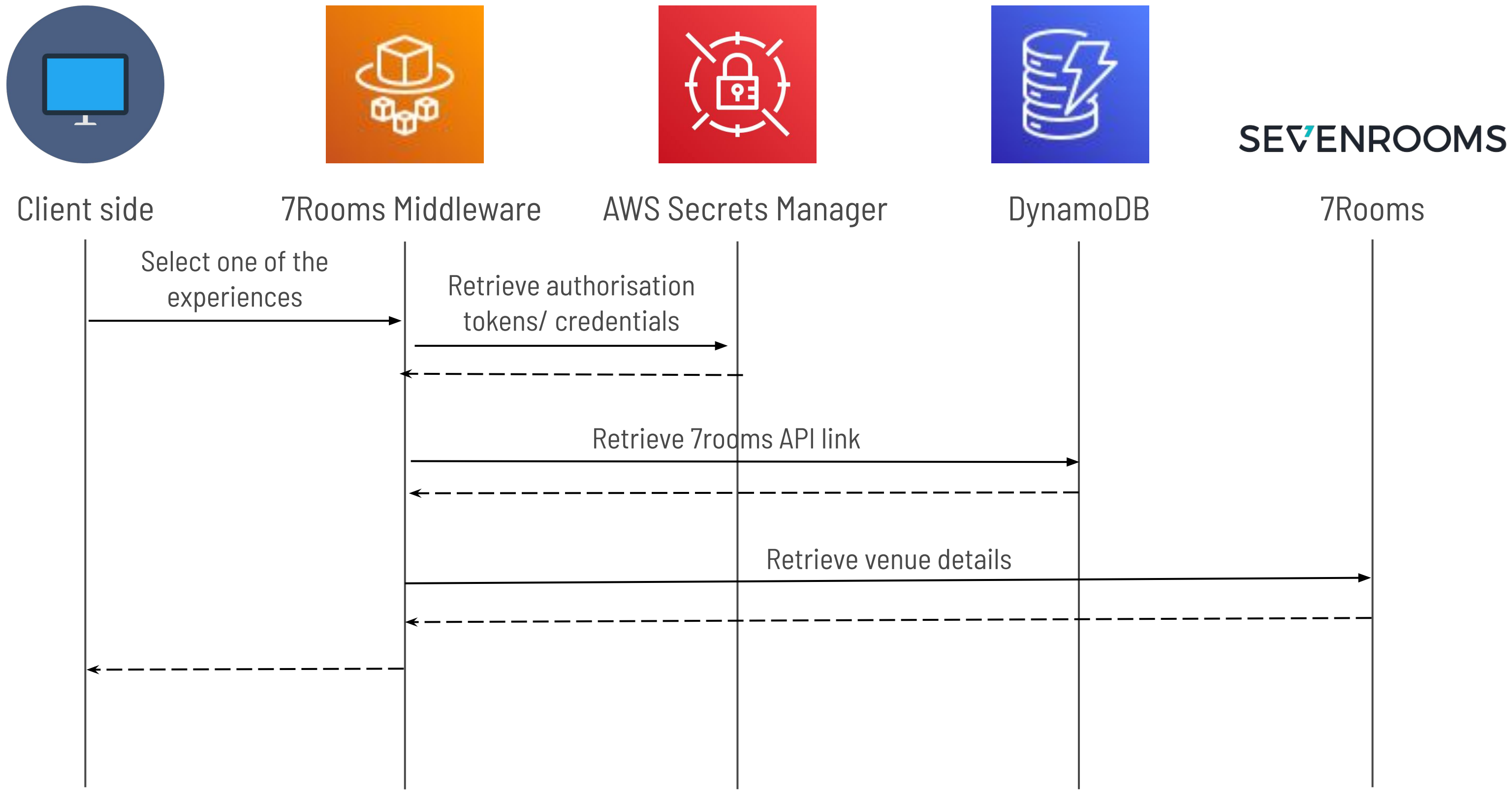
04 View list of user's bookings

List of confirmed bookings made by the user *(7Rooms)*

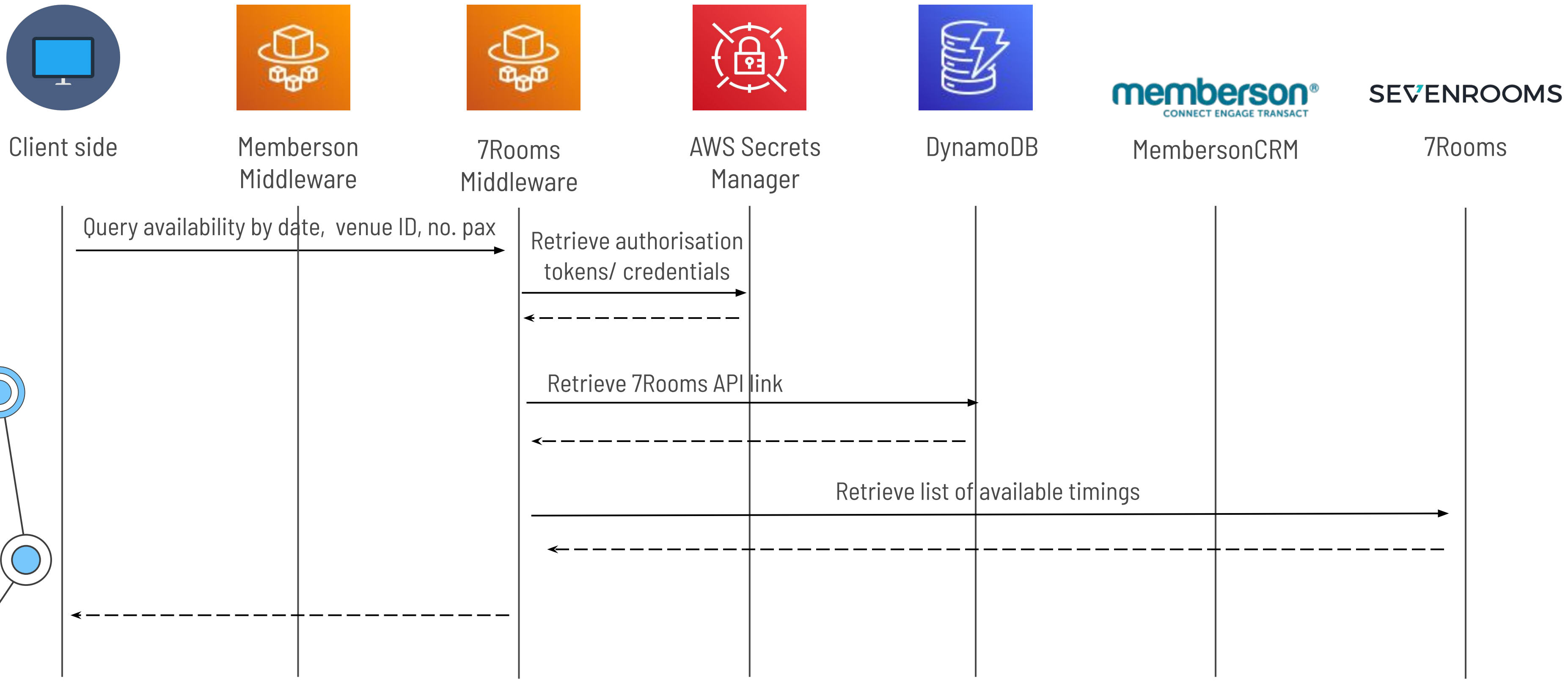
Key use case 1: User Login


Client side
AWS Lambda
AWS Secrets Manager
AWS DynamoDB
memberson CONNECT ENGAGE TRANSACT
MembersonCRM
Cognito
Login
Retrieve authorisation tokens/ credentials
Retrieve Memberson API link
Login
Get or Create user in Cognito user pool
Cognito Login

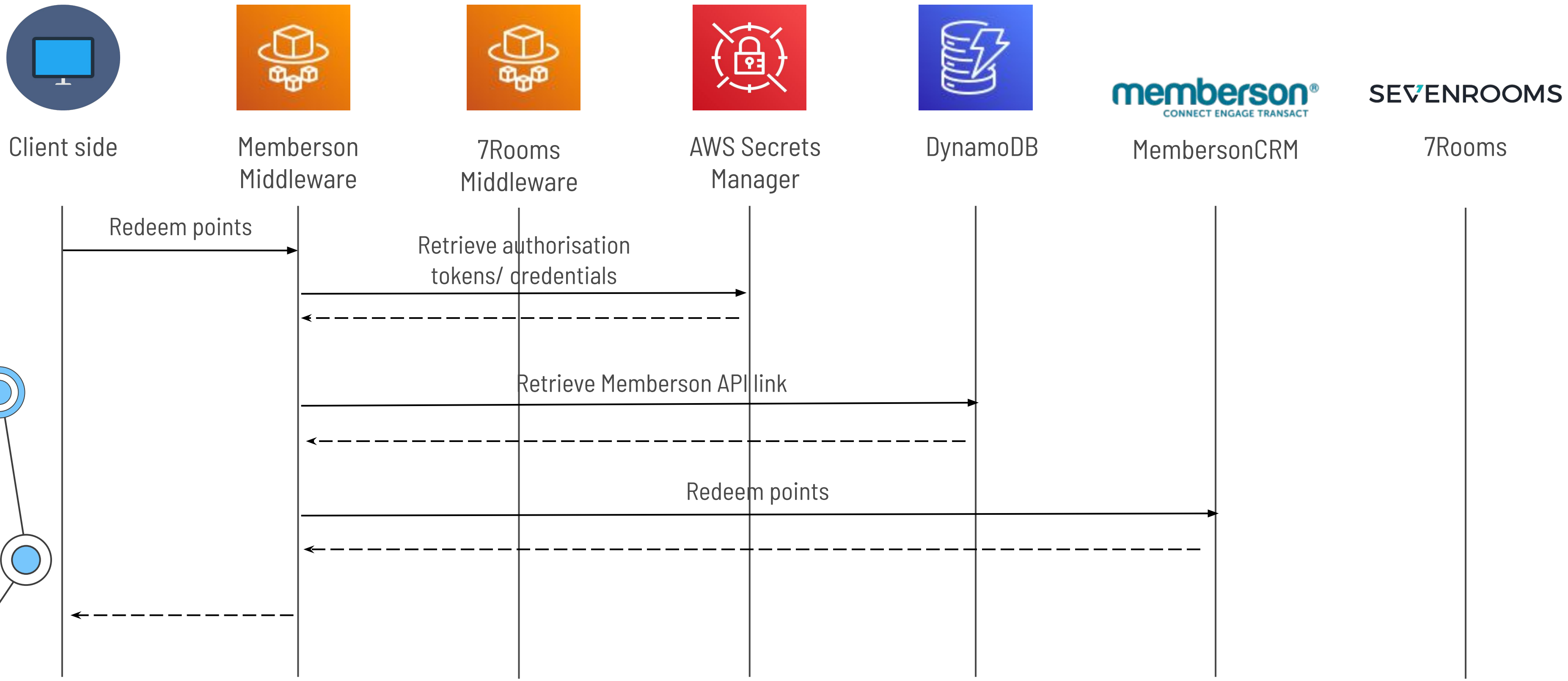
Key use case 2: View Experience



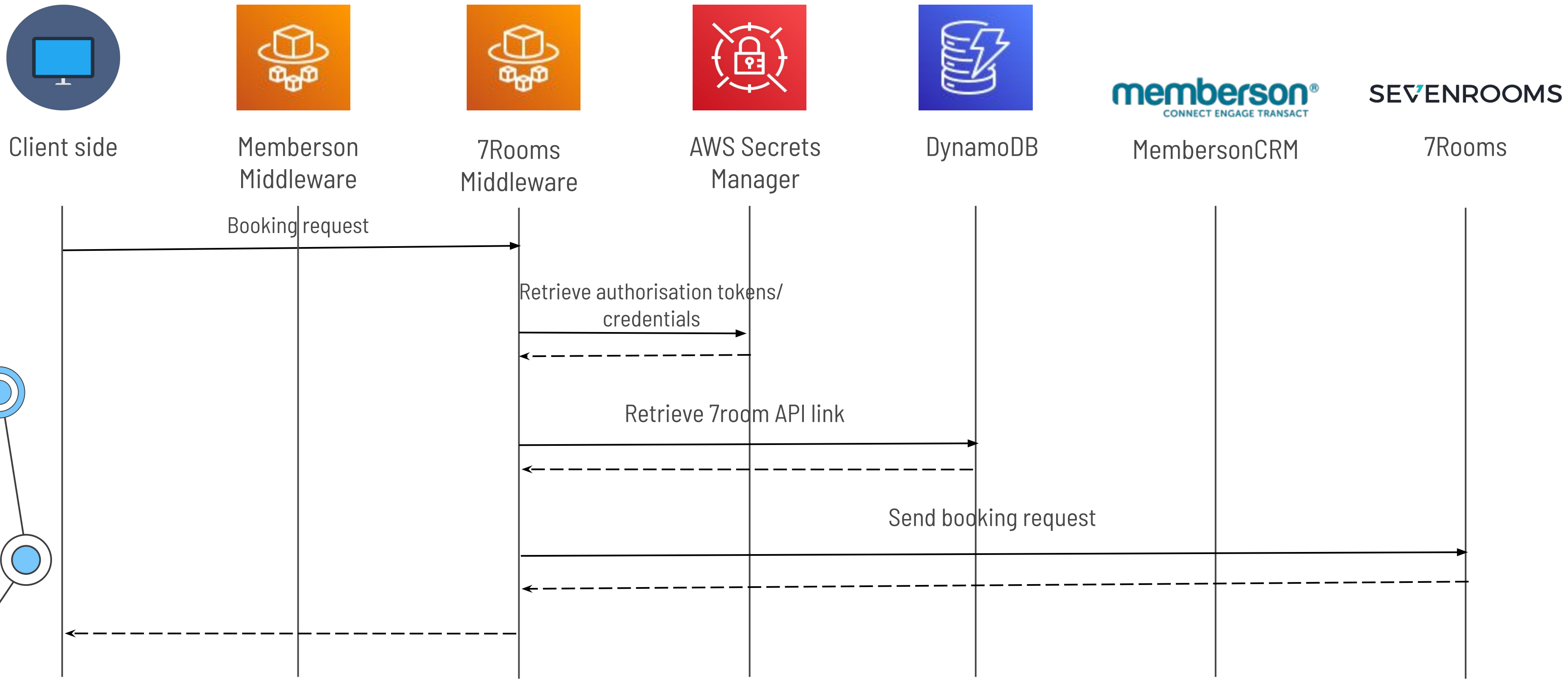
Key use case 3.1: Book an experience with points (Get availability)



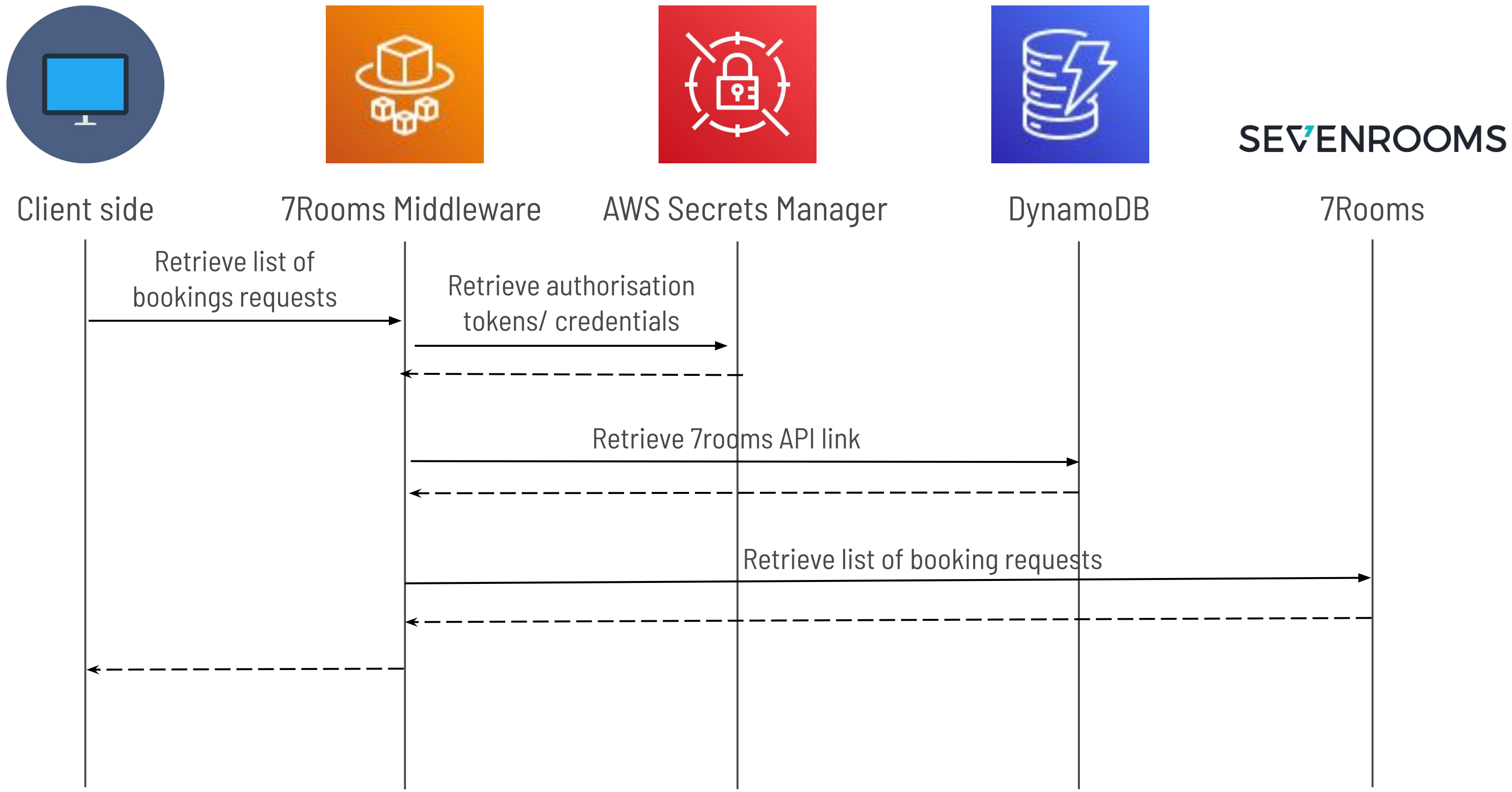
Key use case 3.1: Book an experience with points (Make payment)



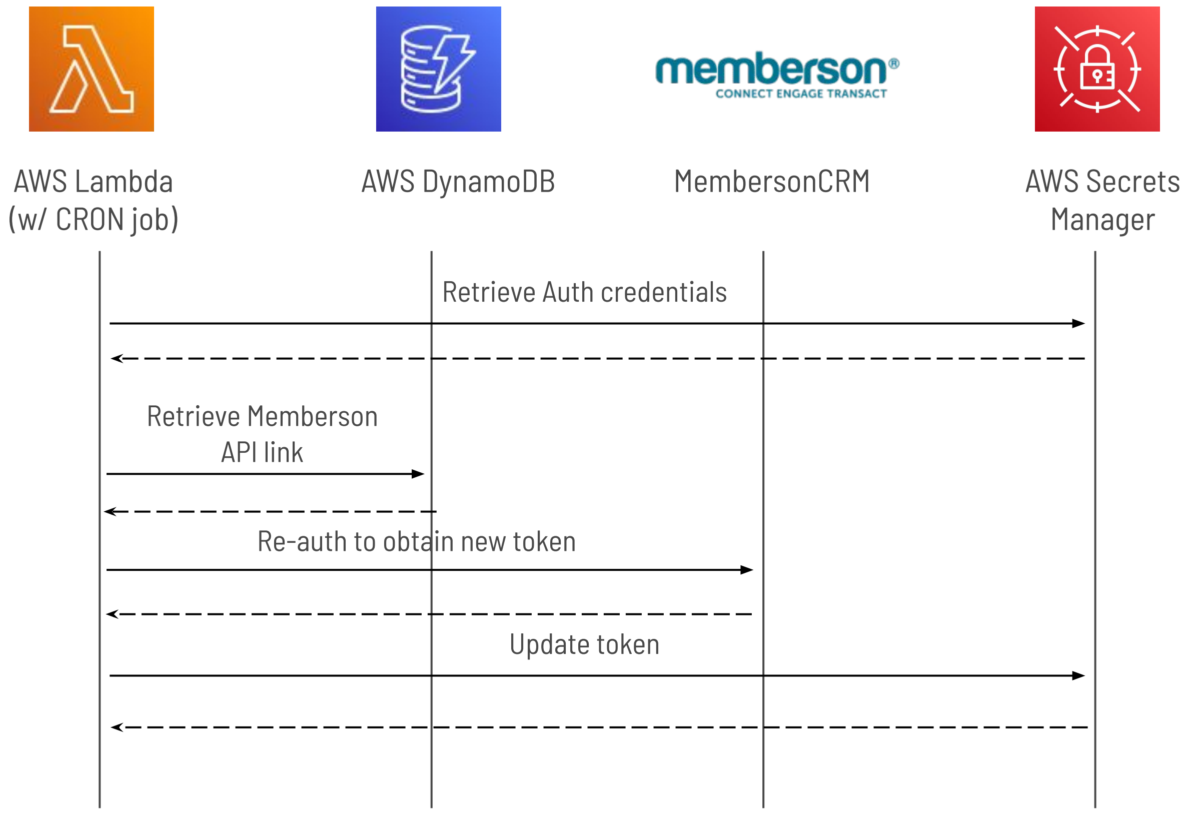
Key use case 3.1: Book an experience with points (Book experience)



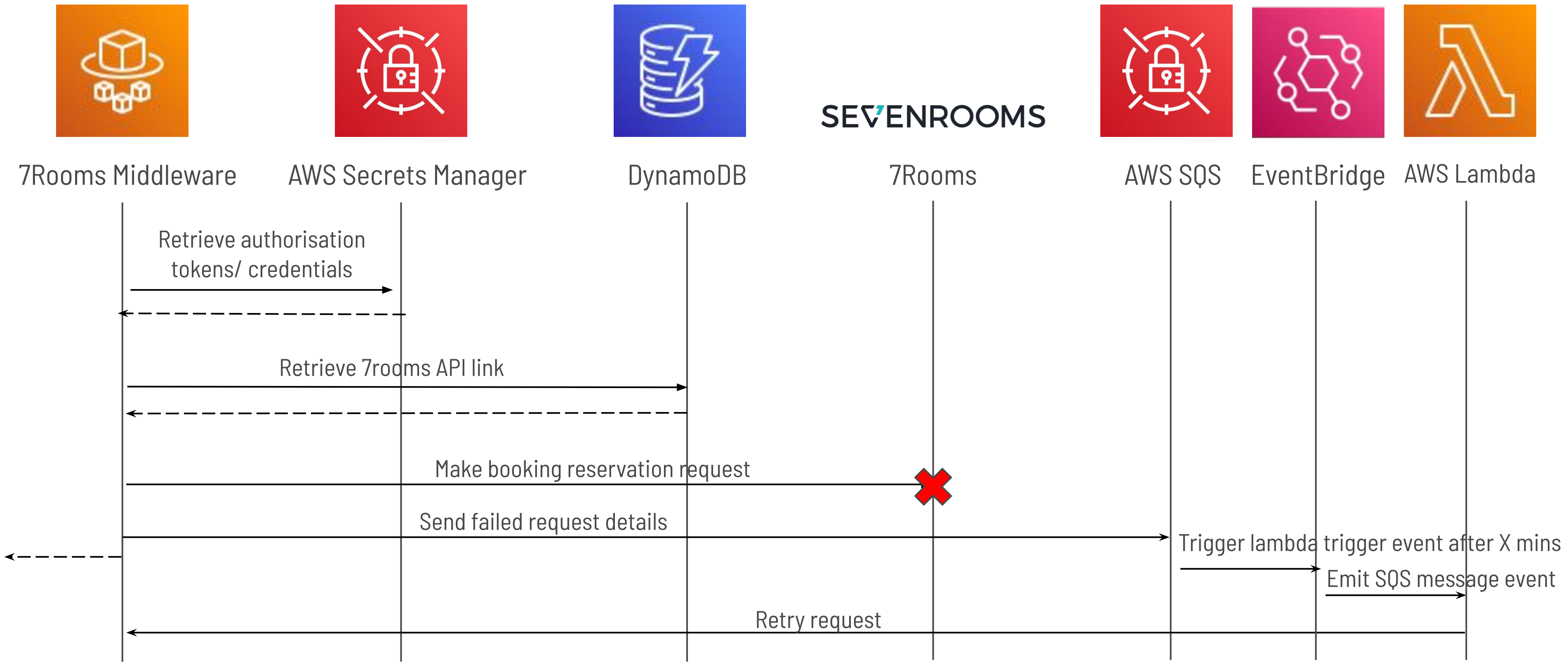
Key use case 4: View list of booking requests


SEVENROOMS
Client side
7Rooms Middleware
AWS Secrets Manager
DynamoDB
7Rooms
Retrieve list of bookings requests
Retrieve authorisation tokens/ credentials
Retrieve 7rooms API link
Retrieve list of booking requests

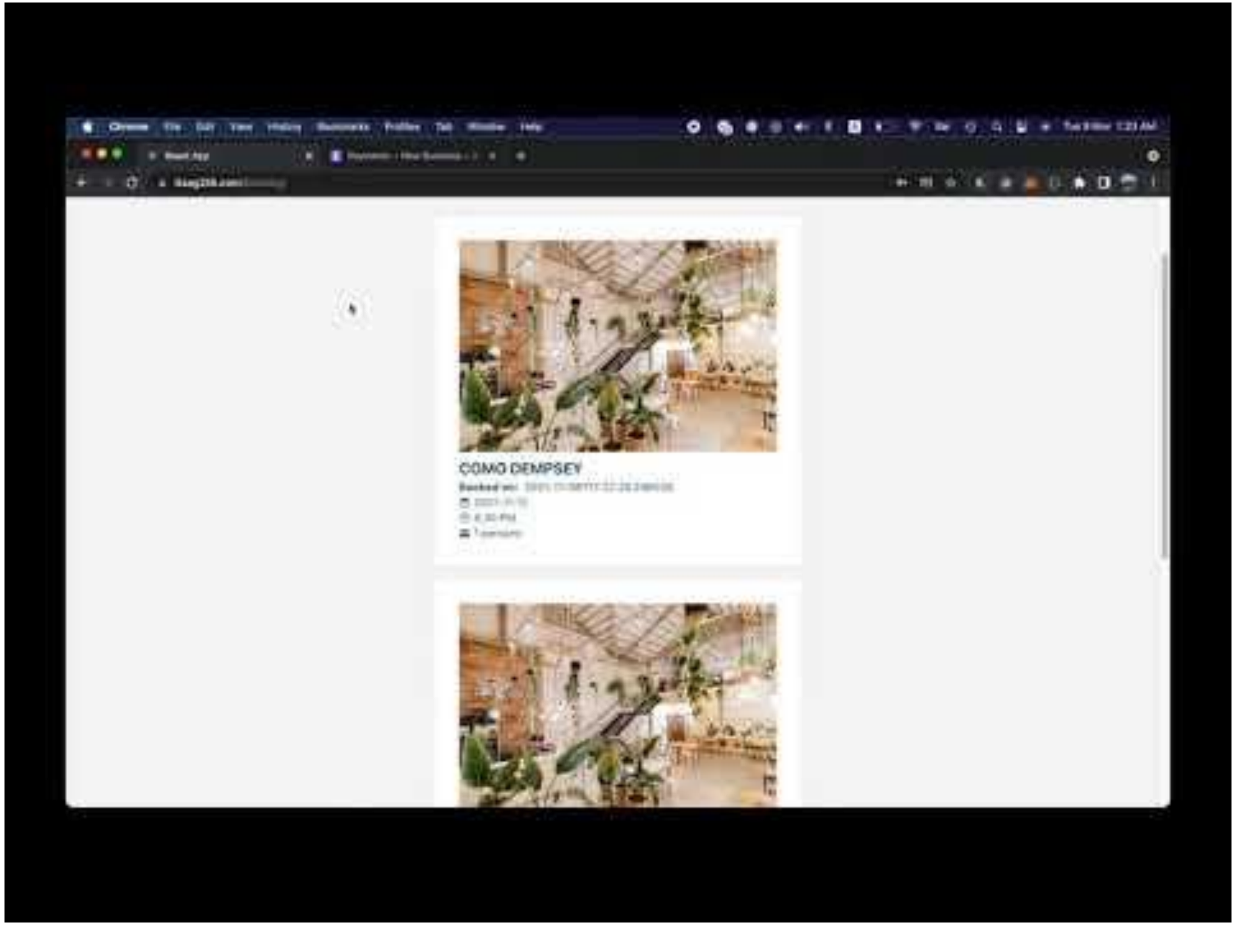
Token rotation


AWS Lambda (w/ CRON job)
AWS DynamoDB
memberson CONNECT ENGAGE TRANSACT
MembersonCRM
AWS Secrets Manager
Retrieve Auth credentials
Retrieve Memberson API link
Re-auth to obtain new token
Update token

Book Experience (Failover)



App demo



COMO DEMPSEY

Booked on: [illegible]

[illegible]

[illegible] PM

[illegible]

02

Quality Attributes

Non-functional requirements by COMO Club

Maintainability

System must be able to handle daily deployments with **no (<1 minute) downtime allowed.**

Automated health checks should be put in place for AWS Services

Availability

System must be **99.9% available** during **normal operating hours**

Mission critical systems must be able to **recover from failure** in **less than 15 minutes (RTPO)**

Security

Personal information should be **encrypted** for **data at rest, motion and in use**, closely following PCI, HIPAA, GDPR compliance

Performance

API round-trip time from a request to a response should be **less than 2 seconds under normal operations with 70 concurrent users**

API round-trip time from a request to a response should be **less than 3 seconds under peak operations with 300 concurrent users**

Scalability

System must be able to support **300 requests per second during the 6pm to 7pm peak hour**

System must be able to support **70 requests (on average) per second during normal operations throughout the day**

03

Key Architecture Decisions

Architecture Decision #1 – Microservices & ECS

Issue:

- Applications are **tightly coupled** and run as a **single service**
- Harder and more **resource-intensive** to implement changes

Alternatives:

- Monolithic Application

Justification

- **Less prone to failures**
- **More resilient** and **better performance**
- **ECS enabled microservice architecture**

Architecture Decision #2 – Authentication with authorization tokens

Issue:

- **Unsecured API endpoints** = easily compromised systems

Justification

- **AWS Cognito** is integrated with AWS Amplify and AWS API gateway to secure our frontend pages and our API endpoints
- Prevents our client facing endpoints from being publicly accessible

Architecture Decision #3 – Infrastructure as Code

Issue:

- Large-scale infrastructure involves a variety of components and configurations
- Configuring components one by one → **difficult to maintain**

Alternatives:

- CloudFormation

Justification

- **Provision infrastructure through code > manual processes**
- Terraform takes **shorter time** to implement support for new AWS Features & supports other cloud providers as well as third-party services

Architecture Decision #4 – Amazon SQS

Issue:

- **Availability of external APIs services** we used in our application is not within our control
- When an external API goes down, it can cause our services to fail

Alternatives:

- AWS MQ

Justification:

- Our middleware containers will push incoming requests from the user to AWS SQS when external APIs go down
- **More cost efficient** than AWS MQ for a **simple use case**

Architecture Decision #5 – VPC Endpoints

Issue:

- Servers running on private subnets **do not have internet access** and they will not be able to access AWS services

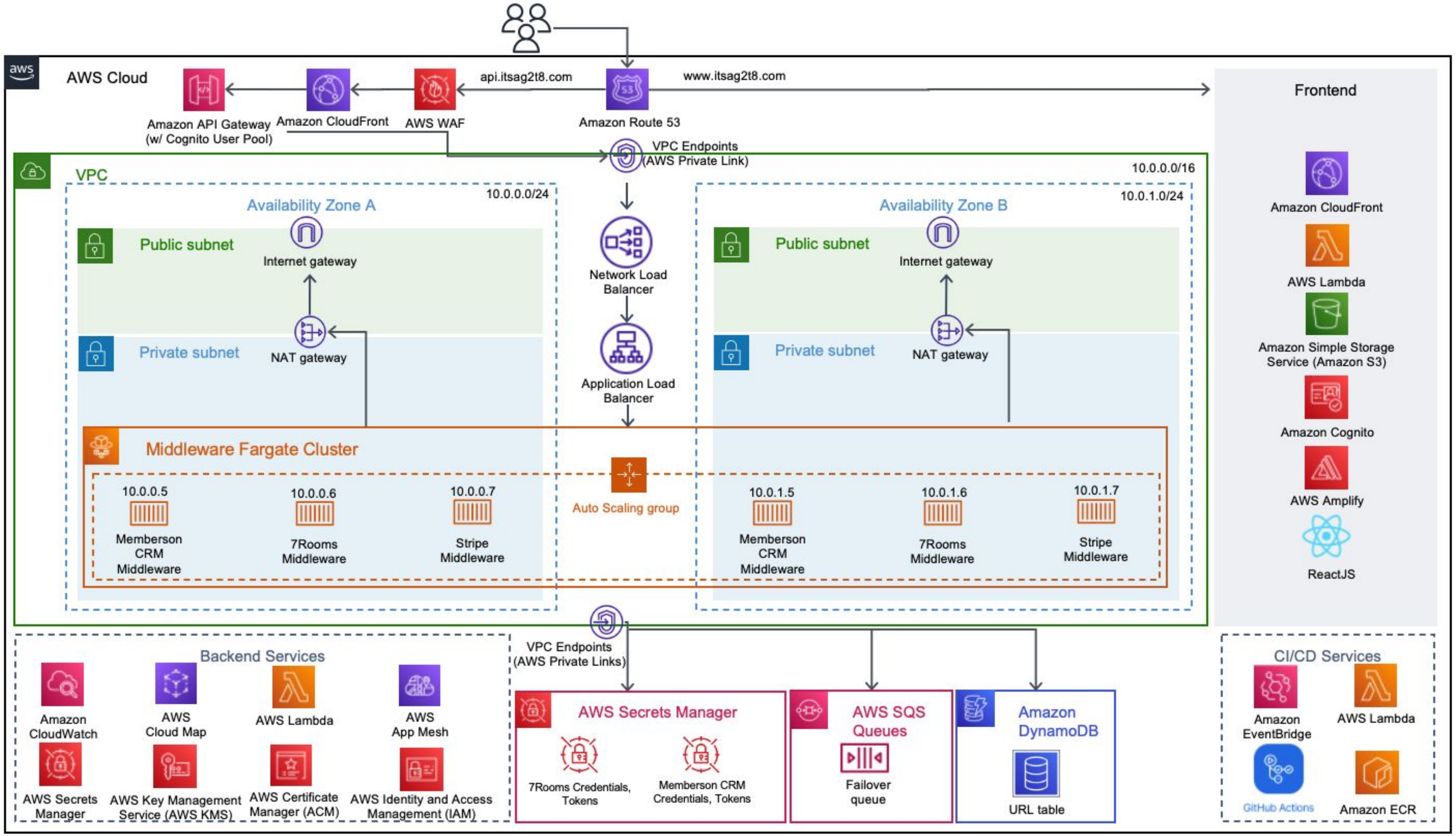
Alternatives:

- NAT Gateway

Justification:

- If only a NAT Gateway is provisioned, traffic will traverse the internet to connect to these AWS services
- Opens up attack surfaces which could compromise our system
- Through VPC Endpoints, **a secure connection** that is not exposed to the internet between our API gateway and our servers is created

04
Architecture Diagram

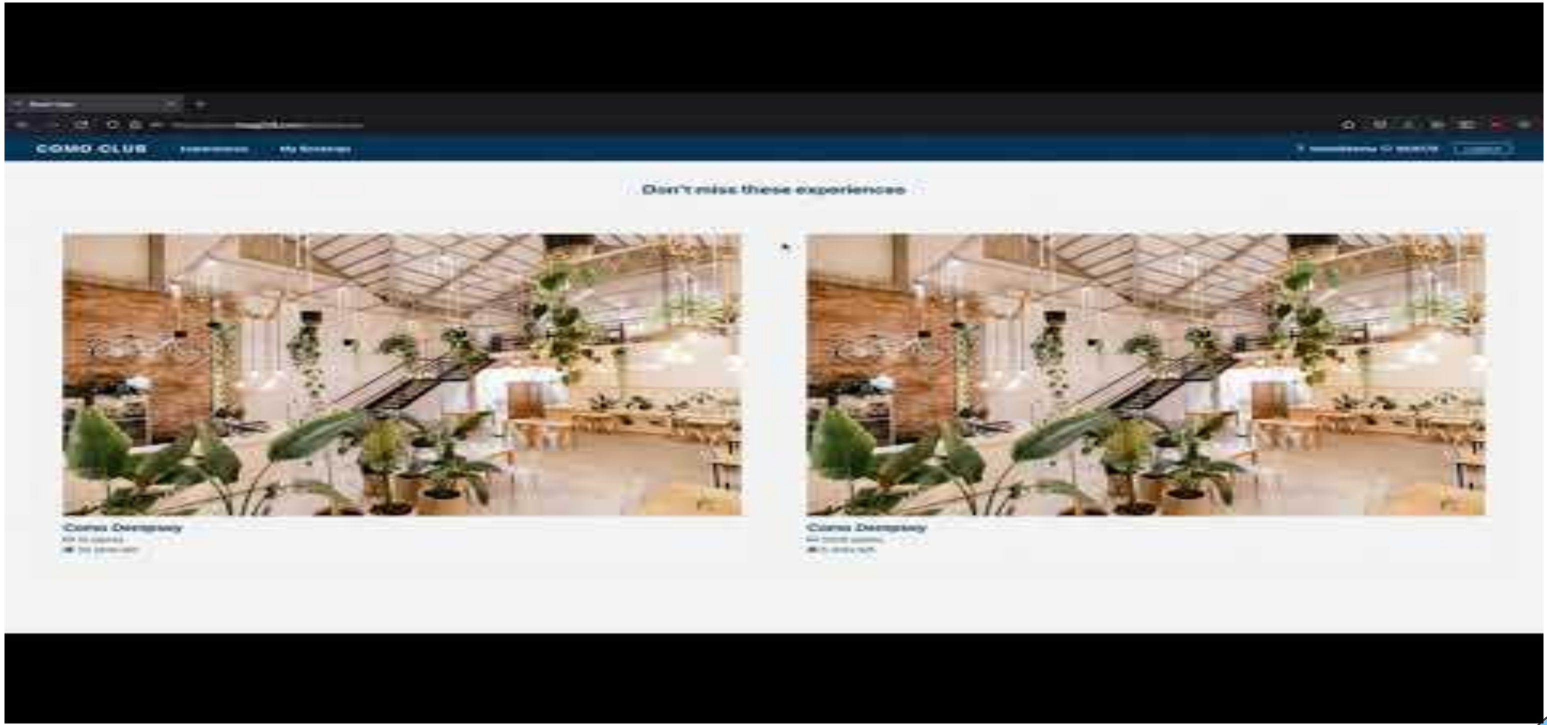

AWS Cloud
Amazon API Gateway (w/ Cognito User Pool)
Amazon CloudFront
AWS WAF
api.itsag2t8.com
Amazon Route 53
www.itsag2t8.com
Frontend
VPC Endpoints (AWS Private Link)
VPC
10.0.0.0/16
Availability Zone A
10.0.0.0/24
Public subnet
Internet gateway
Private subnet
NAT gateway
Network Load Balancer
Application Load Balancer
Availability Zone B
10.0.1.0/24
Public subnet
Internet gateway
Private subnet
NAT gateway
Middleware Fargate Cluster
Auto Scaling group
10.0.0.5
Memberson CRM Middleware
10.0.0.6
7Rooms Middleware
10.0.0.7
Stripe Middleware
10.0.1.5
Memberson CRM Middleware
10.0.1.6
7Rooms Middleware
10.0.1.7
Stripe Middleware
Amazon CloudFront
AWS Lambda
Amazon Simple Storage Service (Amazon S3)
Amazon Cognito
AWS Amplify
ReactJS
Backend Services
Amazon CloudWatch
AWS Cloud Map
AWS Lambda
AWS App Mesh
AWS Secrets Manager
AWS Key Management Service (AWS KMS)
AWS Certificate Manager (ACM)
AWS Identity and Access Management (IAM)
VPC Endpoints (AWS Private Links)
AWS Secrets Manager
7Rooms Credentials, Tokens
Memberson CRM Credentials, Tokens
AWS SQS Queues
Failover queue
Amazon DynamoDB
URL table
CI/CD Services
Amazon EventBridge
AWS Lambda
GitHub Actions
Amazon ECR

05

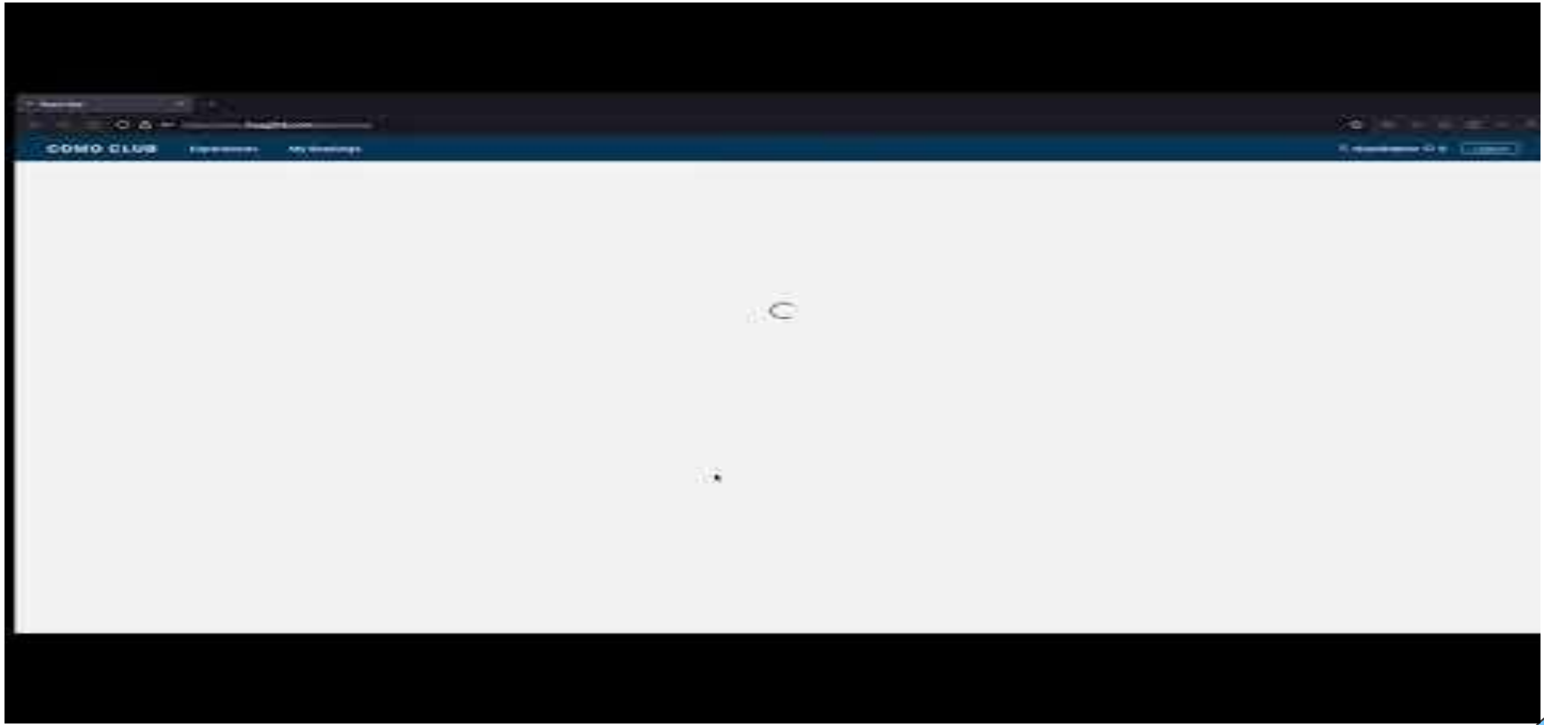
Availability

Designs for Availability

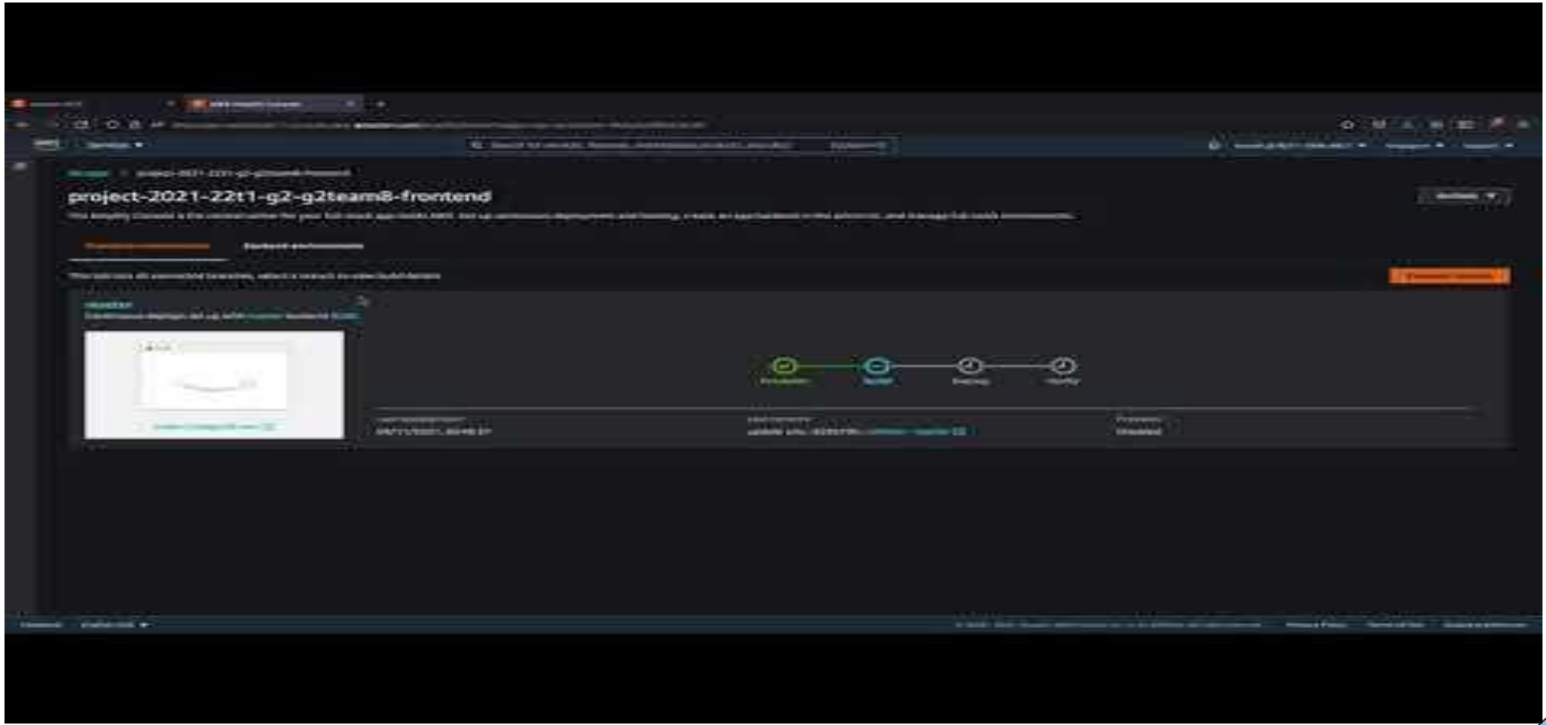
Scenario #1: 1 Availability zone down



Scenario #2: All containers down temporarily



Scenario #3: Disaster Recovery



project-2021-22t1-g2-g2team8-frontend

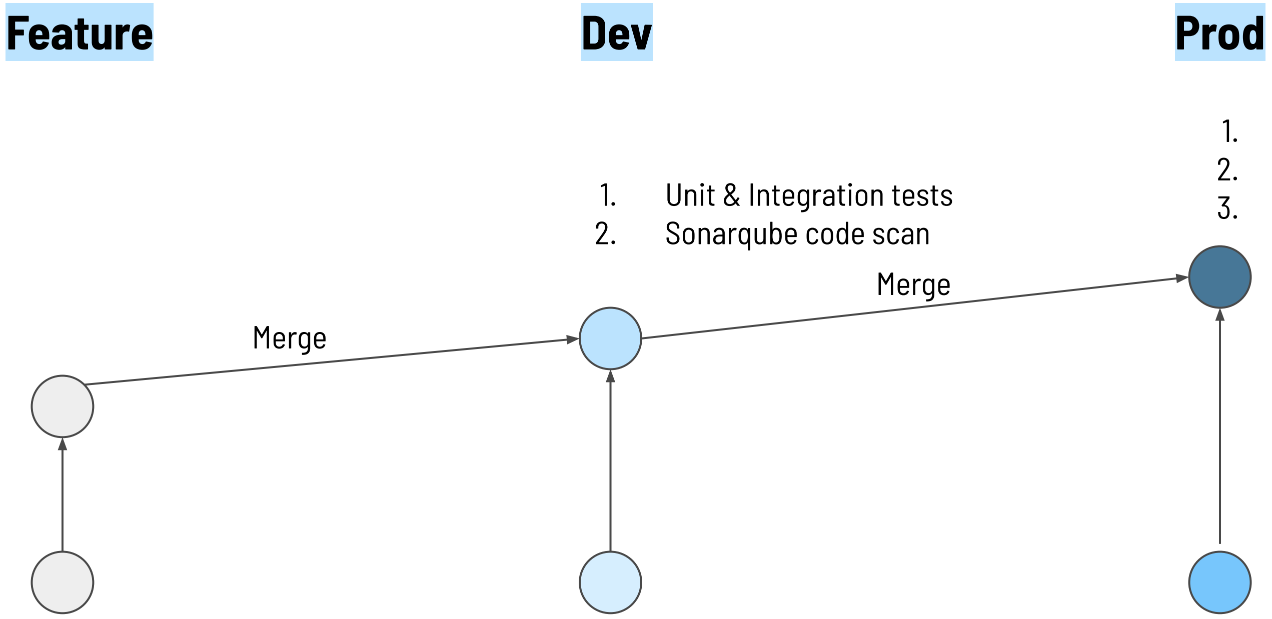
[illegible]

06

Maintainability

Our Development Strategy & CICD Pipeline

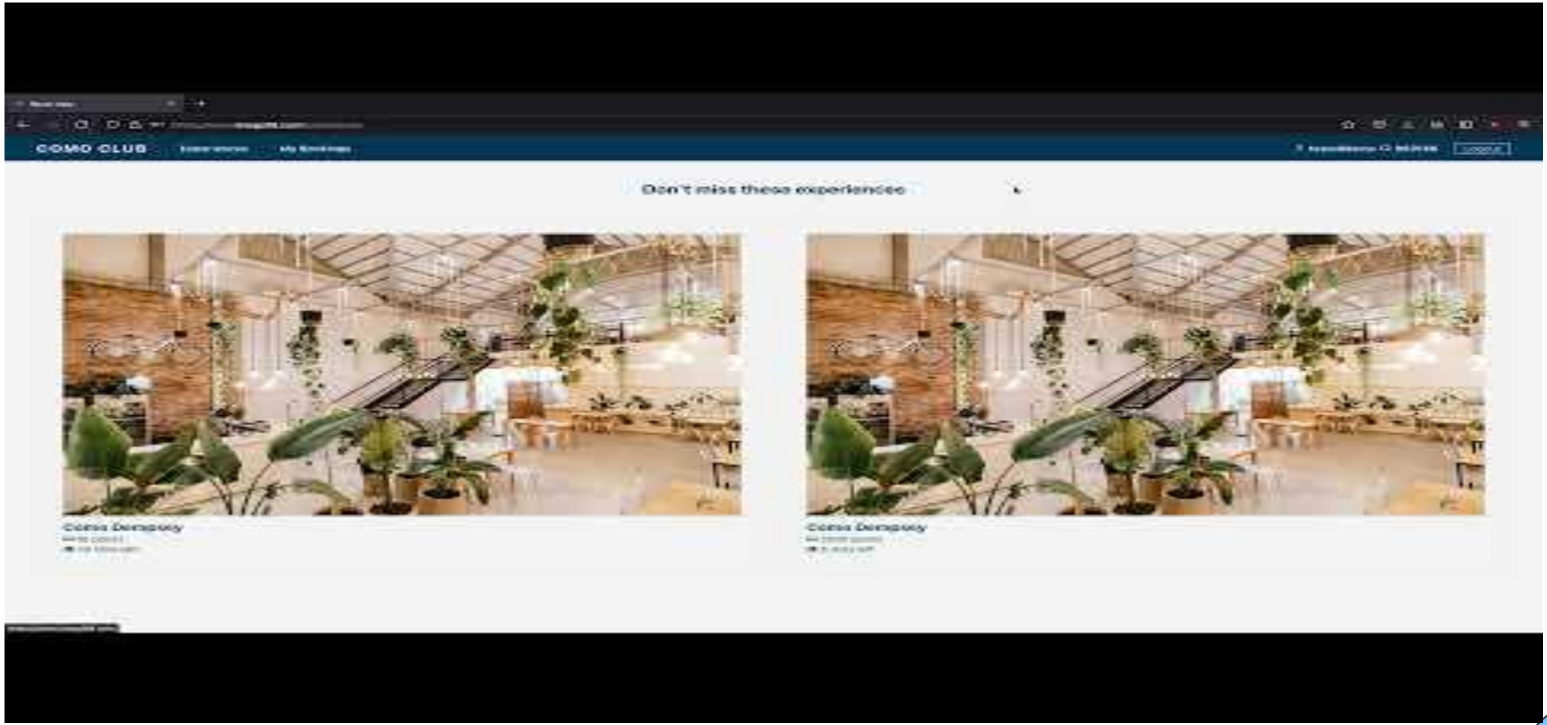
Development strategy



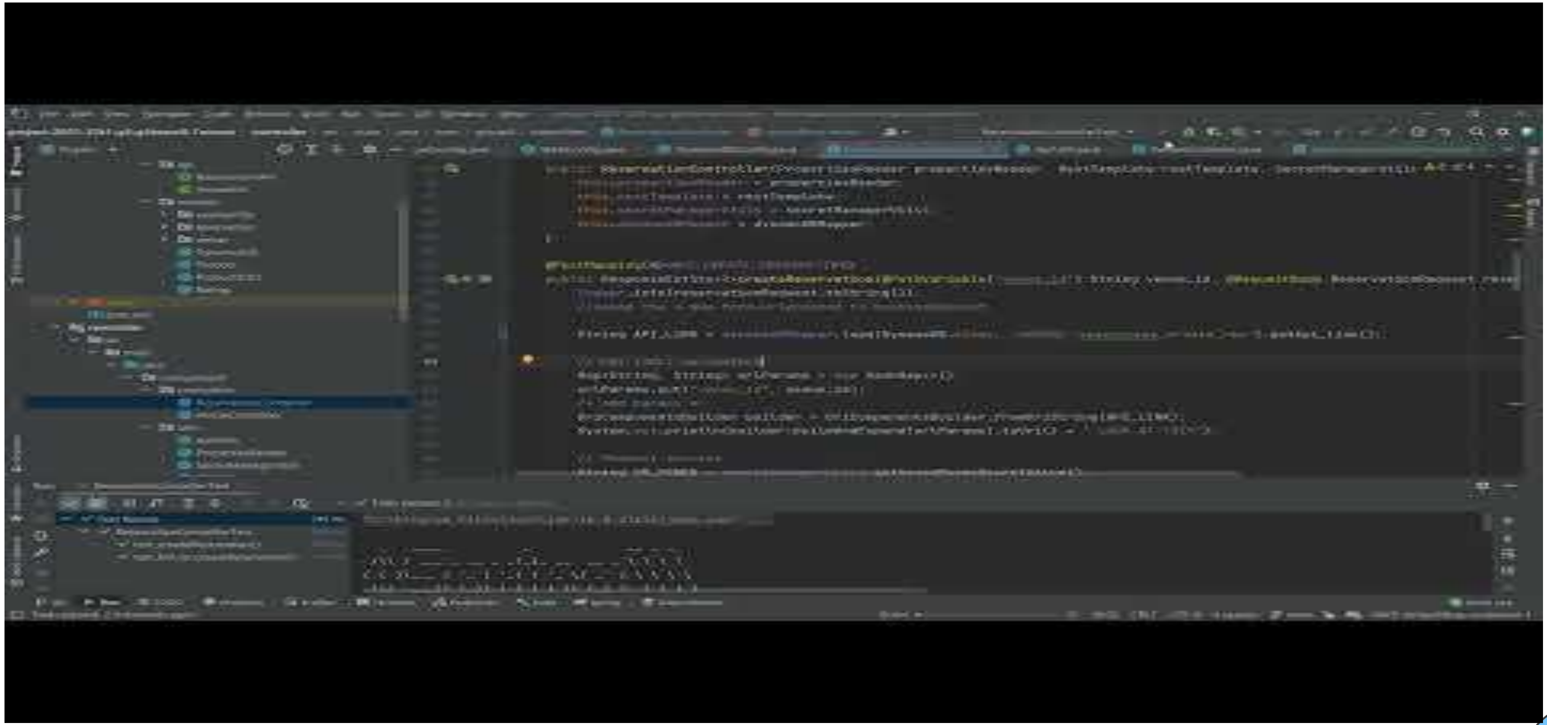
1. Unit & Integration tests
2. Sonarqube code scan

1. Unit & Integration tests
2. Sonarqube code scan
3. Deployment to AWS ECS

CI/CD demo


COMO CLUB
Experiences
My Bookings
Don't miss these experiences
Como Dempsey
Como Dempsey

Integration Test Demo



Architecture Design for Maintainability

API gateway is used to manage the client facing middleware endpoints

Easy onboarding with **terraform and CI/CD configuration templates**

Authentication and access control is implemented at the API Gateway using AWS Cognito

Enhanced logging with log4j2 and Cloudwatch enables future integration with **Elasticsearch, Logstash, Kibana (ELK) stack** for logs analysis

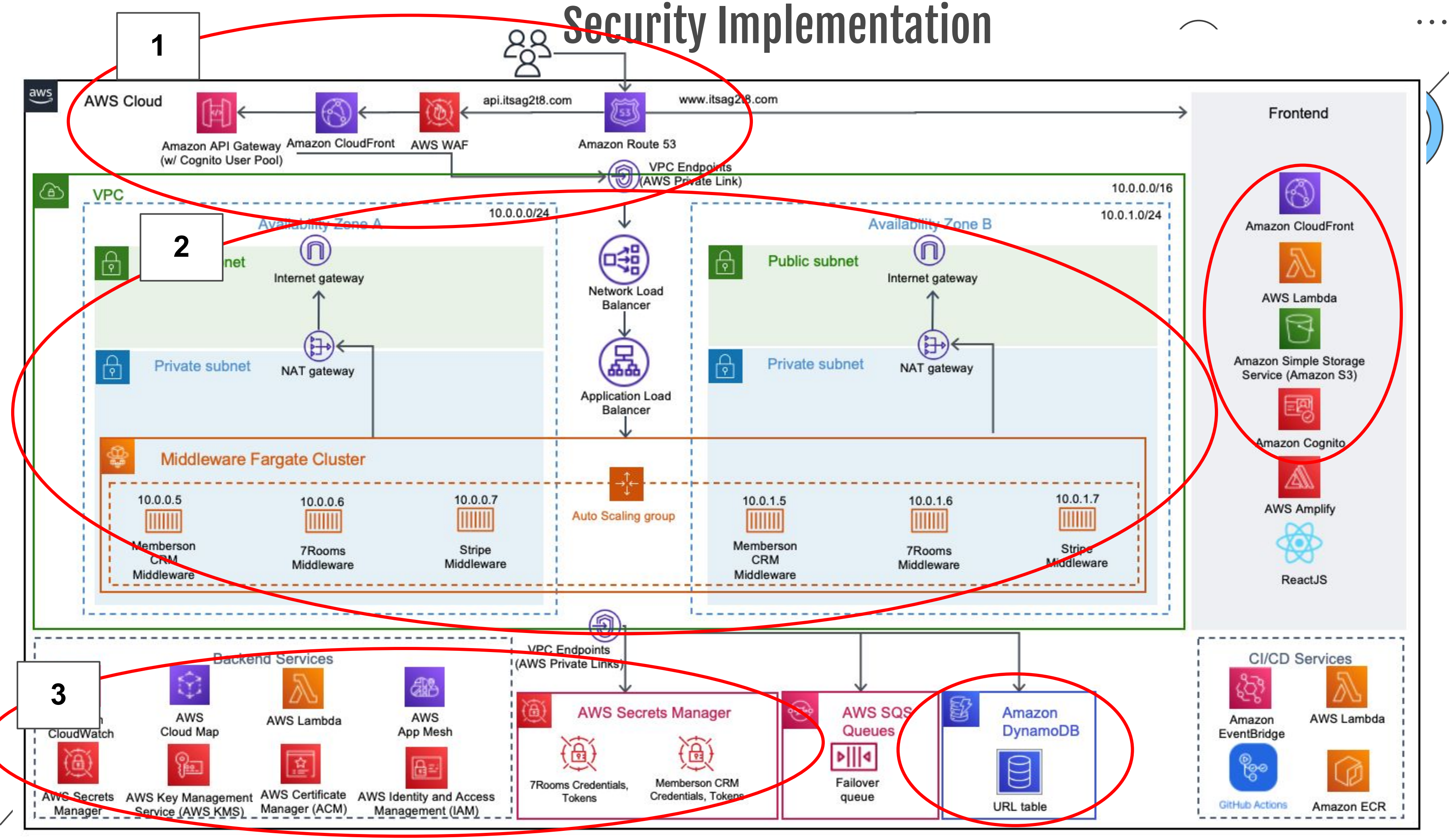
Gain increased **visibility** and **version control** over lambda microservices using **Terraform**

07

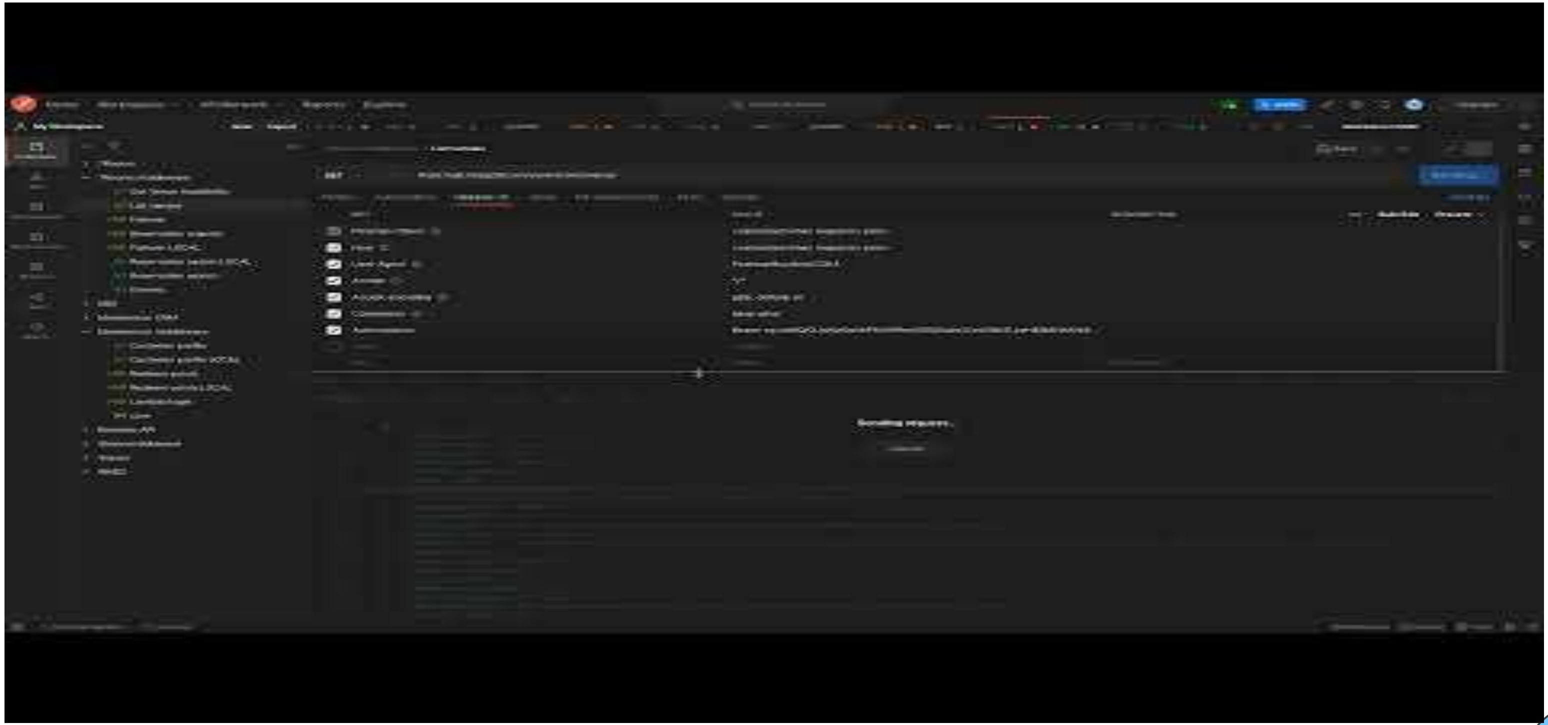
Security

Designs for Security

Security Implementation



WAF AWS Managed Rule Demo



08

Performance

Designs for Performance

Performance

CloudFront Cache

- Caches content at the edge

LB with Auto-scaling

- Traffic assigned evenly by LB based on availability
- Adjust the capacity with Autoscaling

Performance

Testing

- getCustomerProfile
- getAllVenues
- getVenueAvailability
- redeemPoints
- createReservationRequest
- searchReservationRequest

Test	Base	CloudFront Cache
70 Users | 1 Ramp up	2015ms	1011ms
300 Users | 1 Ramp up	7425ms	3055ms
300 Users | 30 Ramp up	5258ms	324ms

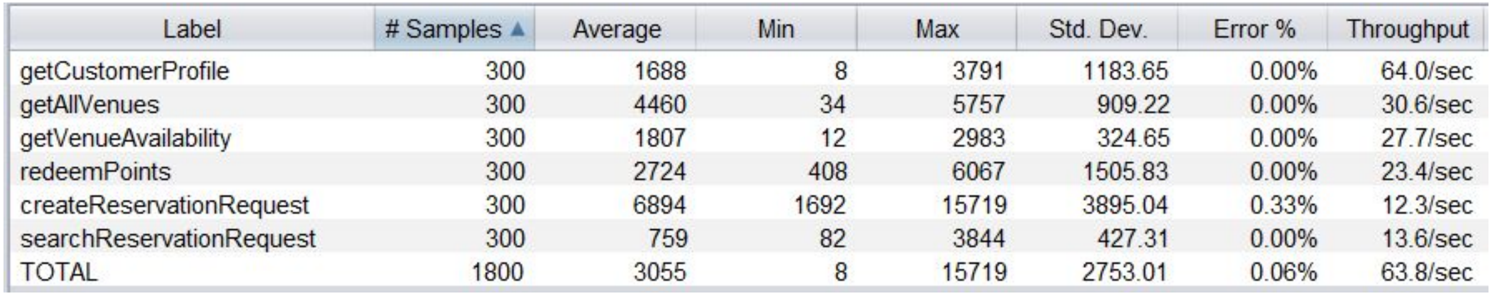
Performance

Testing

Quality requirement achieved:

- 300 hits within ~3s

- getCustomerProfile
- getAllVenues
- getVenueAvailability
- redeemPoints
- createReservationRequest
- searchReservationRequest



Label	# Samples ▲	Average	Min	Max	Std. Dev.	Error %	Throughput
getCustomerProfile	300	1688	8	3791	1183.65	0.00%	64.0/sec
getAllVenues	300	4460	34	5757	909.22	0.00%	30.6/sec
getVenueAvailability	300	1807	12	2983	324.65	0.00%	27.7/sec
redeemPoints	300	2724	408	6067	1505.83	0.00%	23.4/sec
createReservationRequest	300	6894	1692	15719	3895.04	0.33%	12.3/sec
searchReservationRequest	300	759	82	3844	427.31	0.00%	13.6/sec
TOTAL	1800	3055	8	15719	2753.01	0.06%	63.8/sec

09

Overall Costs

Overall Cost

Environment	Monthly Cost	First 12 Months Cost
Development	114.08	1,368.96
Production	516.32	6,195.79

Thank You!