Safe Check-in Platform

Stay Safe-Check in Safely

Team 2, 5th May

Ge Xuanhui | Huang Luohua | Michelle Chew |
Tang Shenqi | Zhang Rongze

Timestamp: 8 AM Sunday Morning – 24 April

Contents Page

Points to discuss:

OVERVIEW 01
– Business Problem & Context

BUSINESS PROBLEM

Limited platform functionality

- Business logic based on vaccination status
- Lack of real time site capacity management

High cost of contact tracing

- Random deployment of safe distancing ambassadors
- Manual analysis of cluster analysis due to lack of centralized server

CONTEXT

Using Check in Platform to combat COVID-19/Future Disease X by:

- Allows for real time site capacity monitoring
- Enforcement of covid rules through a portal
- Reducing high cost of contact tracing - SDAs will be informed in the event of a violation of these rules

safe
check in

Introducing Safe Check-in Platform

Enhanced monitoring features

- Site capacity managed on a real time basis
- Ability to add business rules based on new regulation

Lower cost of contact tracing

- Targeted deployment of Safe Distancing Ambassadors

OVERVIEW 01
– Platform Thinking

PRODUCERS

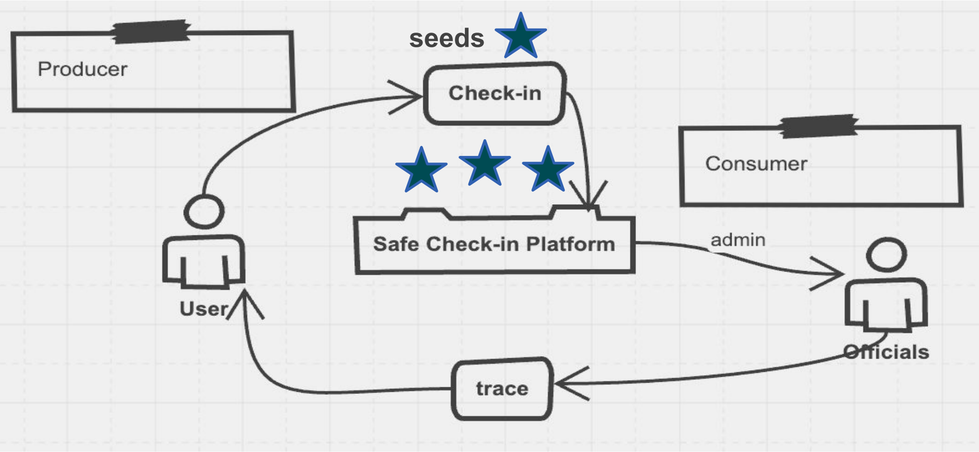
- All users who need to check-in before entering a site

SEEDS

- Check-in events, site information, government rules

CONSUMERS

- Government officials and safe ambassadors



MAGNET

- Enables users to check in safely get feedback of **real-time check-in** status
- Enables users to easily **control** the no of checked in people and **enforce** covid measures
- Enables govt officials to better **deploy** SDAs

TOOLBOX

- Enables users to do check- in from **mobile app** (QR code)
- **Rule engine** to manage regulation rules
- **Admin portal** to update banned/unbanned users/sites
- **Email notification** service to inform SDAs where a violation takes place
- **Caching of user info** upon check in

MATCH MAKER

- **Real Time** Data Analytics
- **Predictive Analytics** to forecast potentially crowded sites

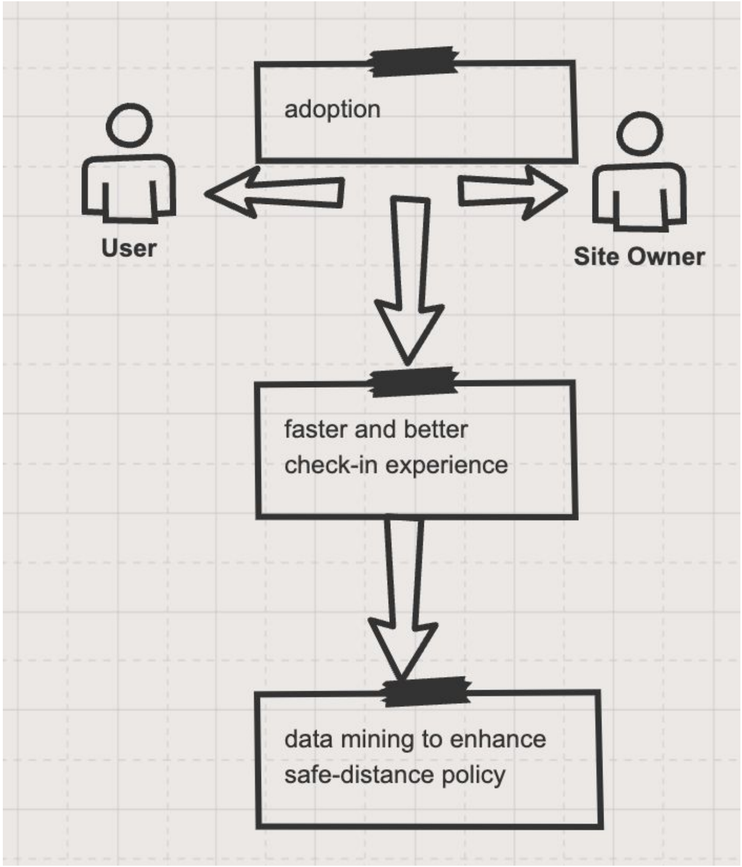
OVERVIEW 01
– Platform Thinking

Scaling Platform

- First start with check-in regulation, and then we will scale our business to support covid contacting trace measures, apply big data to detect highly vulnerable sites, and deploy preventive measures

Platform Ecosystem

- Cloud Platform: AWS
- Elastic Beanstalk, AWS Lambda Serverless, API Gateway
- Access and Role: VPC, Private Subnet, Role-based Lambda Execution
  Predictive analytics to forecast potentially crowded sites

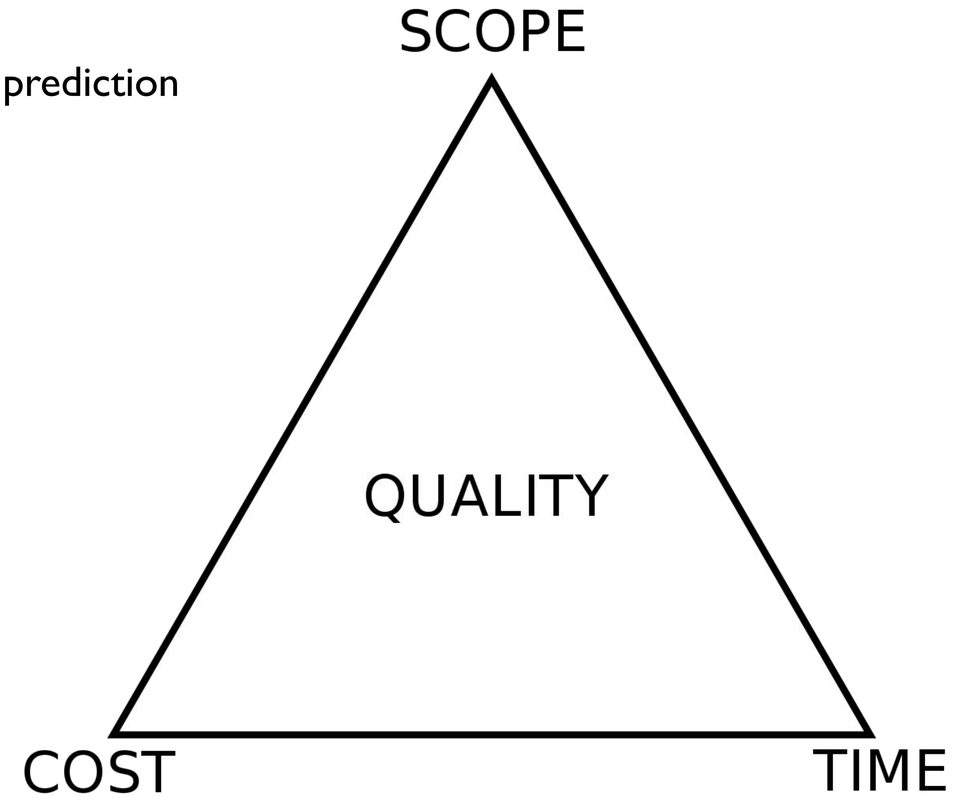


OVERVIEW 01
– Functional Scope and Challenges

OVERVIEW 01
– Functional Out of Scope

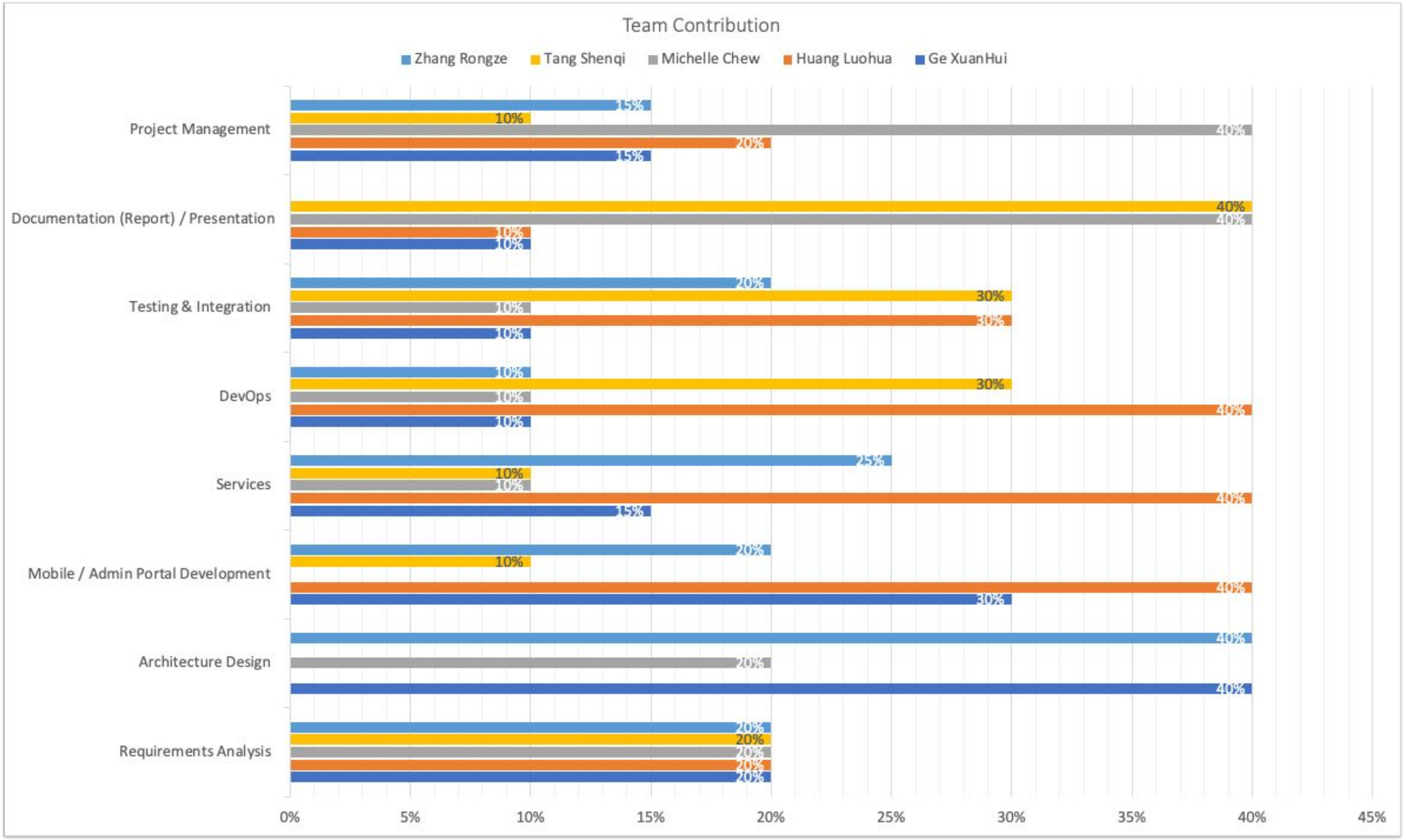
Due to time and resource constraints, the following features are out of scope

- Data Analytics of Check-in events, allowing for cloud mining cluster prediction
- Mobile App for Officials to conduct on-site check
- Token devices for check in
- Privacy concerns of Safe Check in data are accounted for



Project Conduct

Project Conduct 02
– Team Contribution



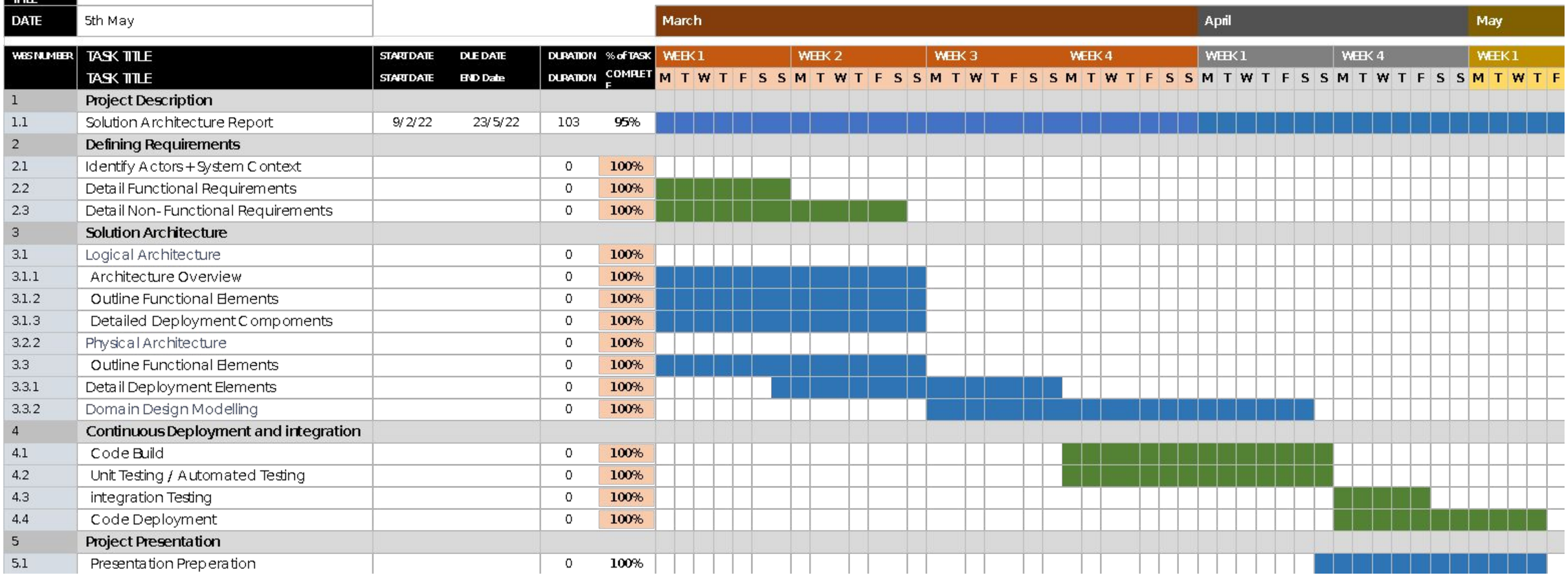
10 contribution to item

Project Conduct 02 – WBS & Challenges

WORK BREAKDOWN STRUCTURE

PROJECT TITLE	Safe Check In Platorm
DATE	5th May

WBS NUMBER	TASK TITLE	START DATE	DUE DATE	DURATION	% of TASK
	TASK TITLE	START DATE	END Date	DURATION	COMPLETE
1	**Project Description**				
1.1	Solution Architecture Report	9/2/22	23/5/22	103	95%
2	**Defining Requirements**				
2.1	Identify Actors + System Context			0	100%
2.2	Detail Functional Requirements			0	100%
2.3	Detail Non-Functional Requirements			0	100%
3	**Solution Architecture**				
3.1	Logical Architecture			0	100%
3.1.1	Architecture Overview			0	100%
3.1.2	Outline Functional Elements			0	100%
3.1.3	Detailed Deployment Compoments			0	100%
3.2.2	Physical Architecture			0	100%
3.3	Outline Functional Elements			0	100%
3.3.1	Detail Deployment Elements			0	100%
3.3.2	Domain Design Modelling			0	100%
4	**Continuous Deployment and integration**				
4.1	Code Build			0	100%
4.2	Unit Testing / Automated Testing			0	100%
4.3	integration Testing			0	100%
4.4	Code Deployment			0	100%
5	**Project Presentation**				
5.1	Presentation Preperation			0	100%



Challenges

- Covid Times – Devops culture allows for collaboration
- Learning of new cloud tools for implementation – AWS

Project Conduct 02 – Status

Requirement Analysis

Define requirement, business analysis, and MVP

- Discuss and define business requirements
- Analysis use cases and MVP of the platform

100%

Tech Design

Architecting, Tech Decisions, and Tooling

- Develop logical and physical architecture diagram
- Setup IDE and Dev environments
- Create tools

100%

Services

Develop backend services and APIs

- Develop backend API/Lambda services
- Self-testing
- Integrate with FE
- Performance tuning and debug

100%

Mobile & Admin Portal

Define Mobile App and Admin Portal

- Develop mobile App for Check-in
- Develop Admin Portal for administration
- Integrate with BE
- Performance tuning and debug

100%

DevOps

Define pipeline for build/deploy, conduct SCM

- Define CICD pipeline for build/deploy
- Manage infra and artifacts
- Conduct SCM for the platform

100%

Testing & Integration

Integration testing, verify the functions

- Conduct integration testing for all components
- Verify all implemented functions and fix bugs

100%

Logical Architecture & Design

Logical Architecture & Design 03
– Key Architectural Decisions

1

Domain Driven Design

- Ubiquitous language
- Breaks down complex domain into multiple sub domains for easy communication and maintenance

2

Microservices

- Polyglot architecture that is scalable

Alternative

- Use monolithic architecture for build

x Tightly Coupled

3

Event – Driven

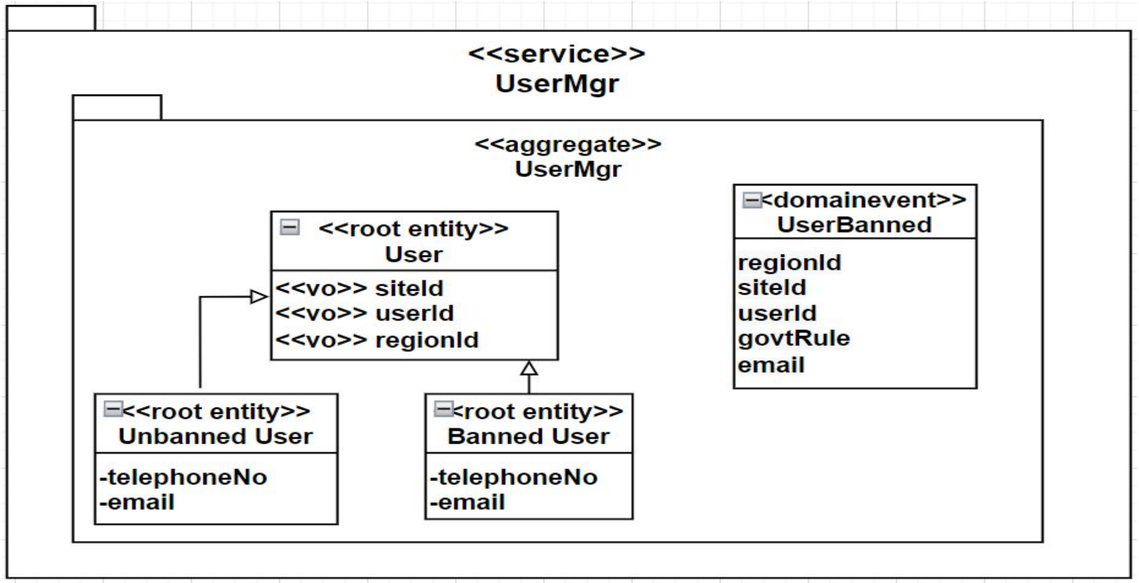
- Real time processing of millions of check in requests

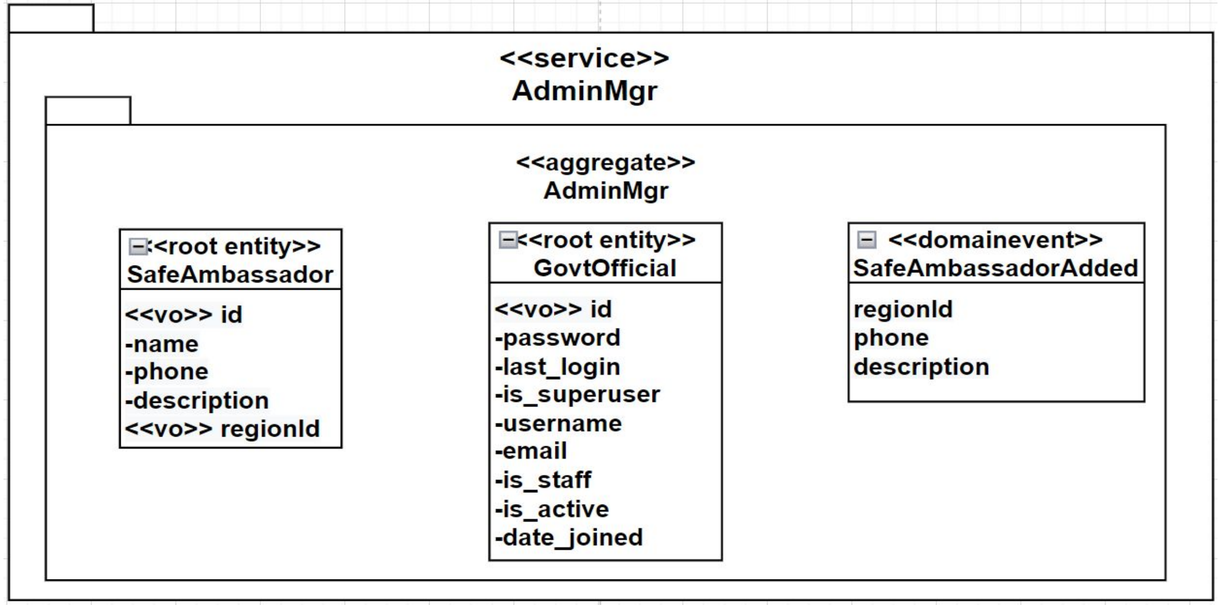
Alternative

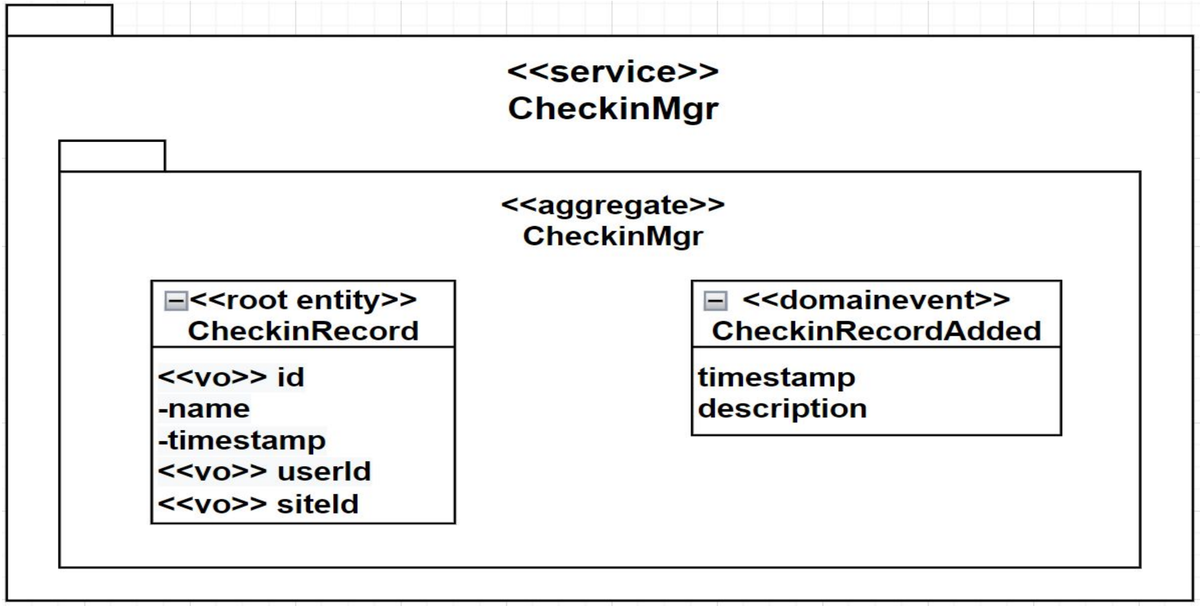
- Batch Processing

x Unable to support real time monitoring

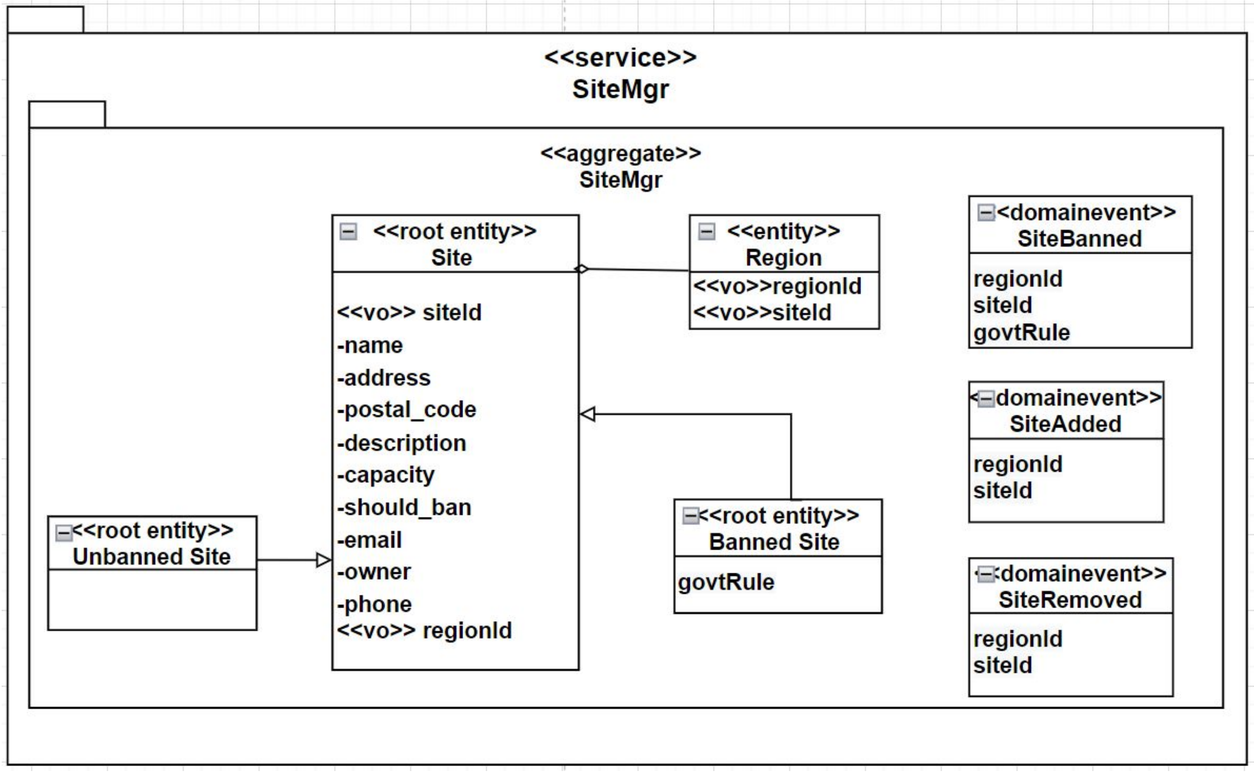
Logical Architecture & Design 03
– Logical Architecture (Domain Model) **Producers and Consumers, Seeds**

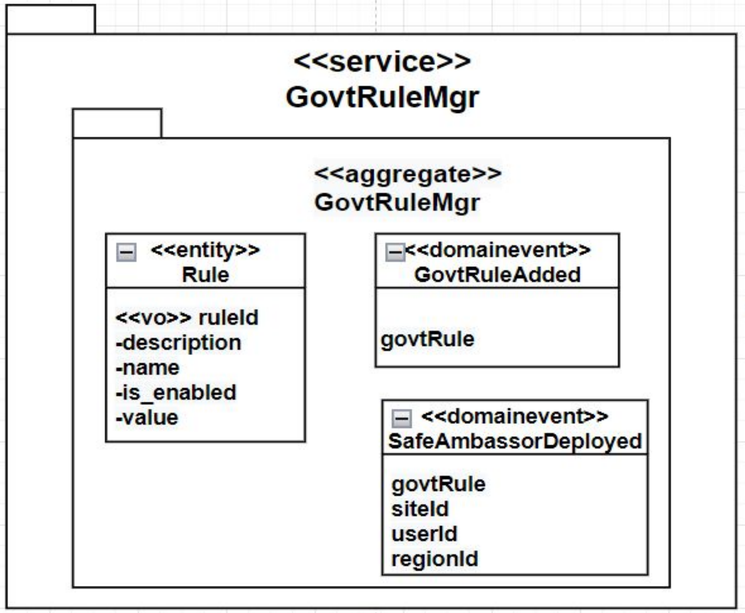


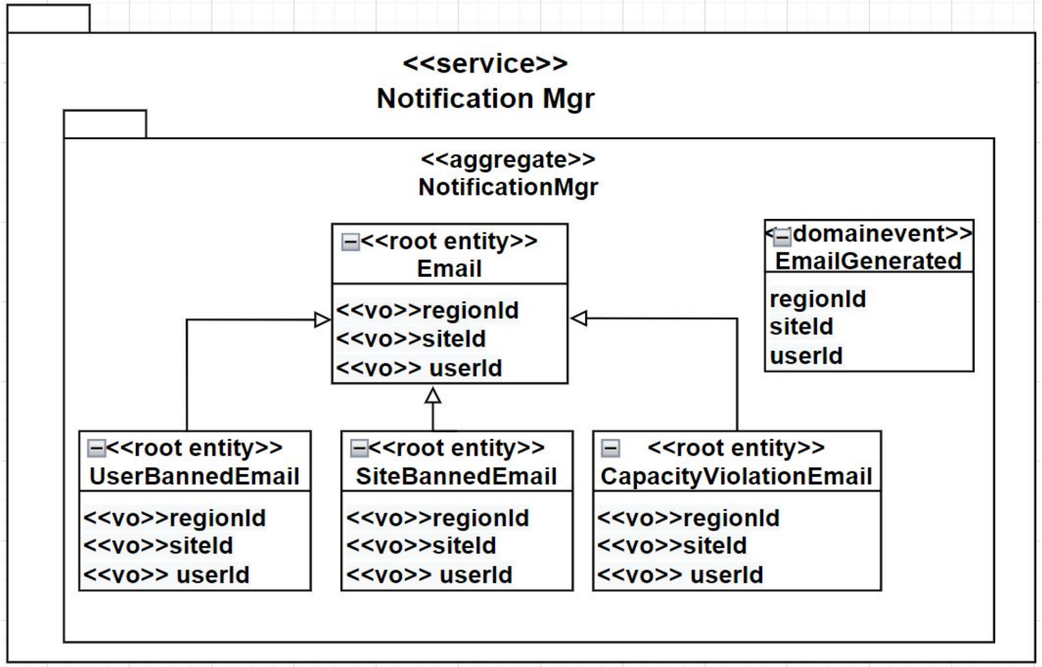




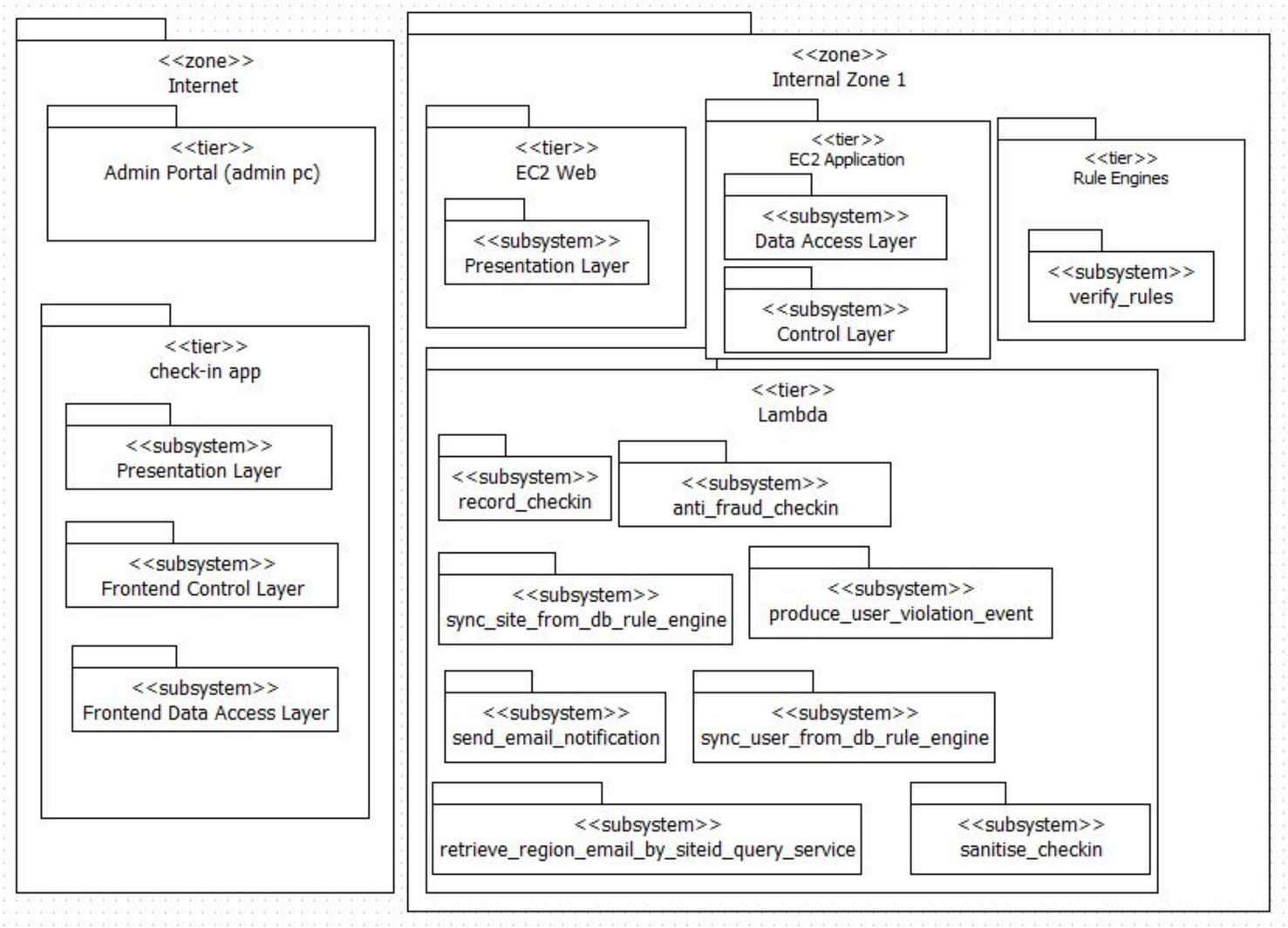
Logical Architecture & Design 03
– Logical Architecture (Domain Model) **Magnet/Toolbox**



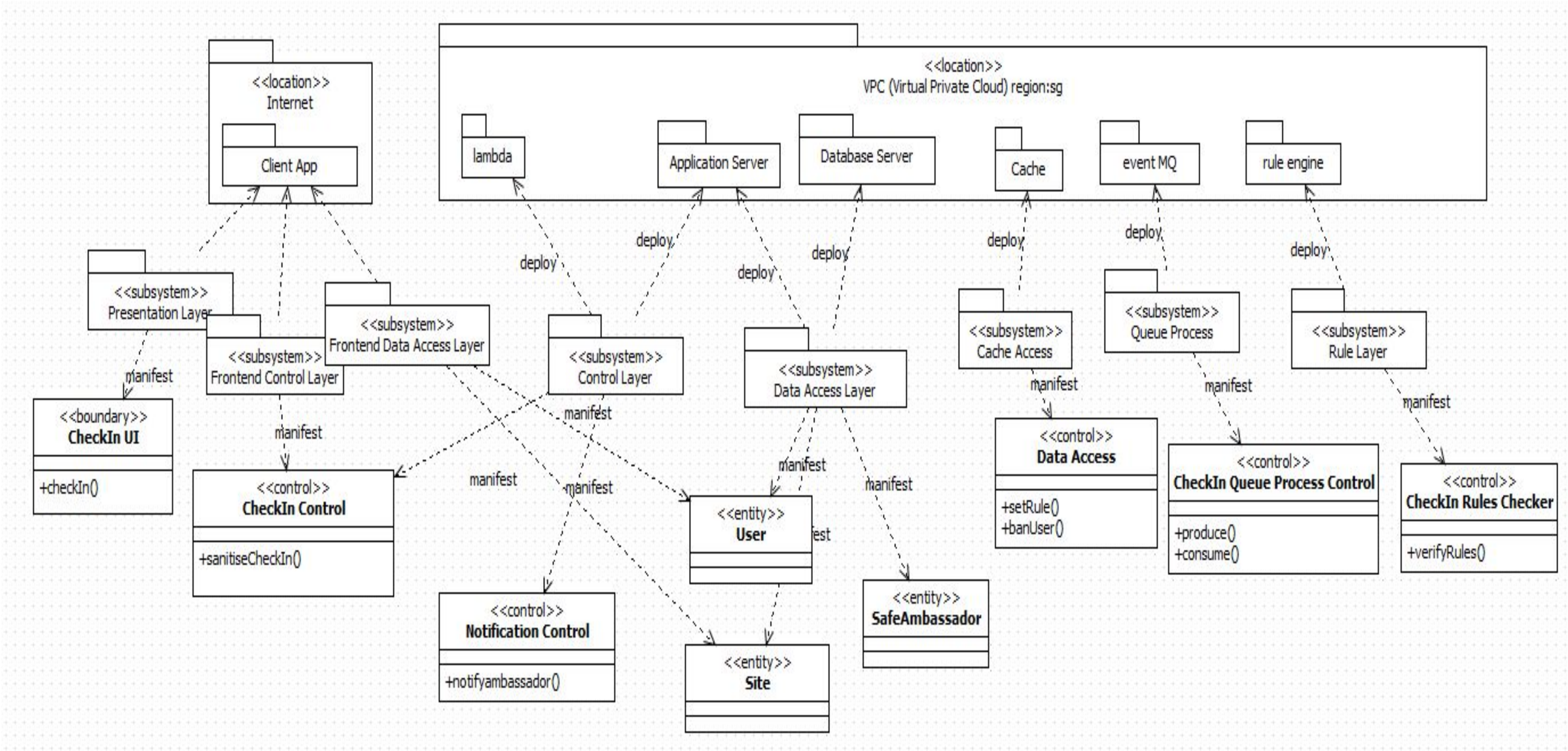




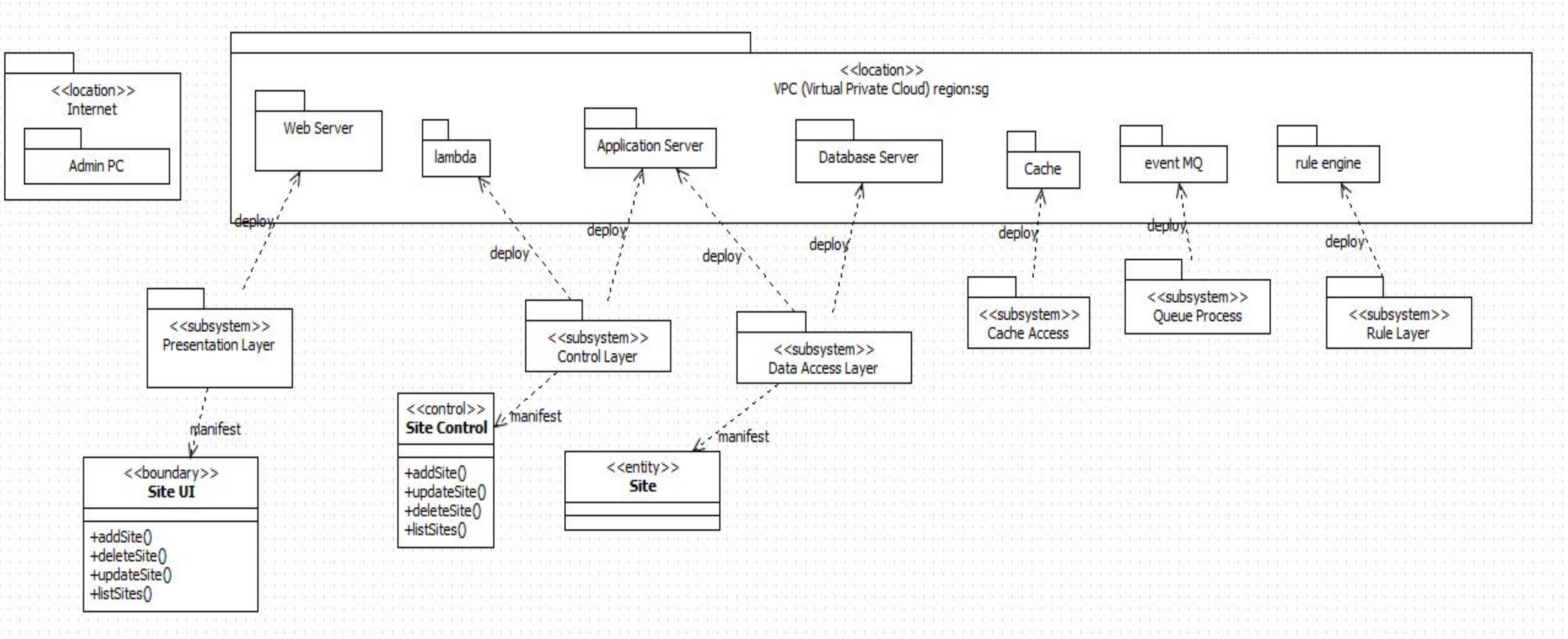
Logical Architecture & Design 03
– Overview



Logical Architecture & Design 03
– Detailed deployment diagram User Check- in



Logical Architecture & Design 03
– Detailed deployment diagram Admin Portal Site CRUD



Physical Architecture & Design

Physical Architecture & Design 04
– Key Architectural Decisions

1

Serverless (AWS Lambda)

- Achieve automatic scaling as use base grows
- No need to upload code to servers or manage backend configuration

2

Cache Mechanism

- Real time processing of millions of check in request
- High performance

Alternative

- Singular database

x Unable to support real time monitoring

3

AWS for CI/CD Pipeline

- CI/CD Pipeline for automatic integration and code build

Alternative

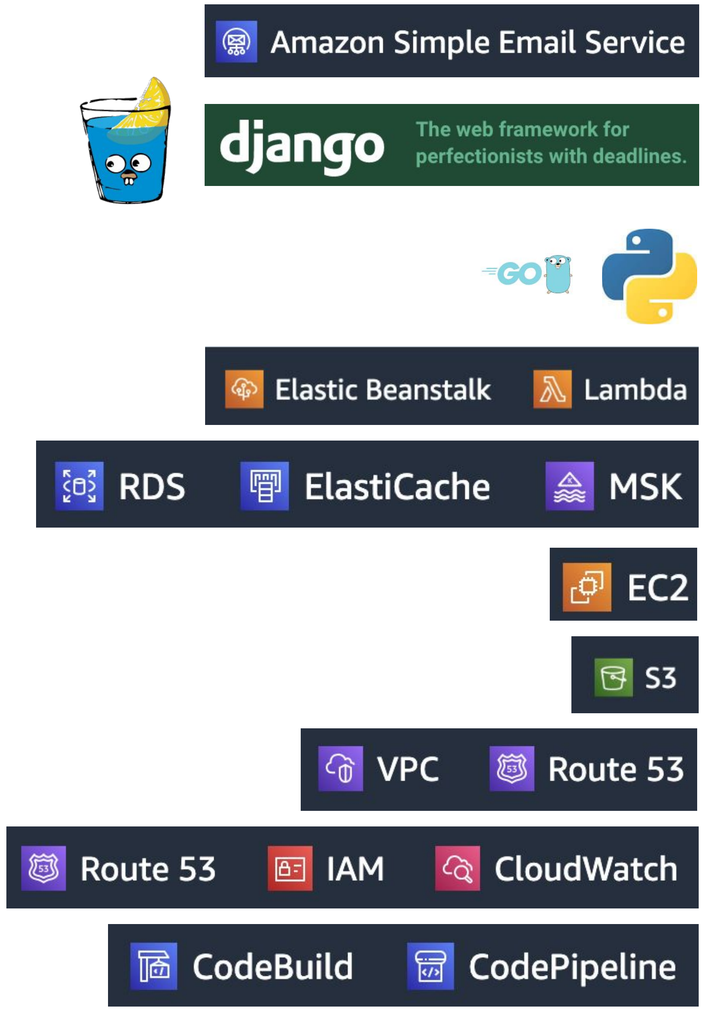
- Use other 3rd party service provider

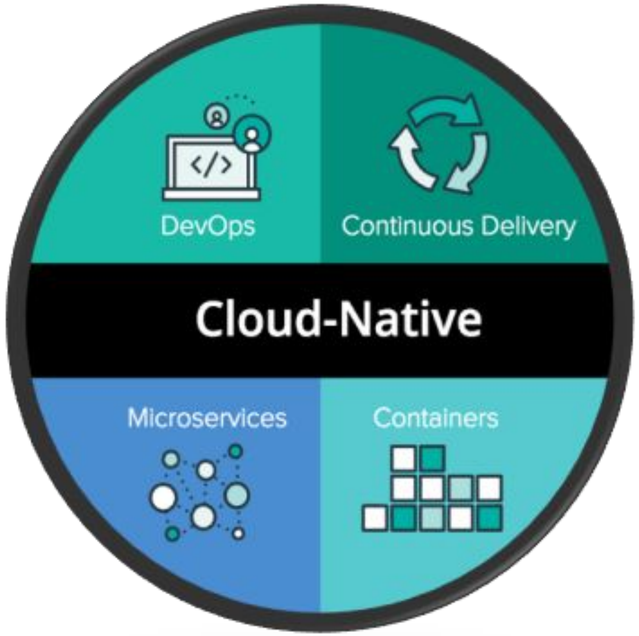
x Piecemeal integration and platform management

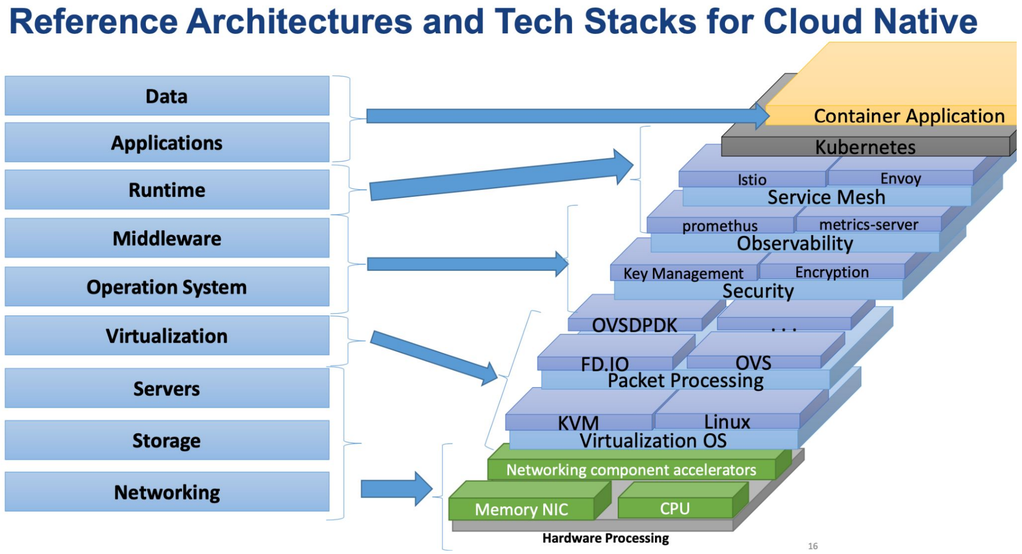
Physical Architecture & Design 04
– Tech Stacks

Apply a Cloud Native Tech Stacks

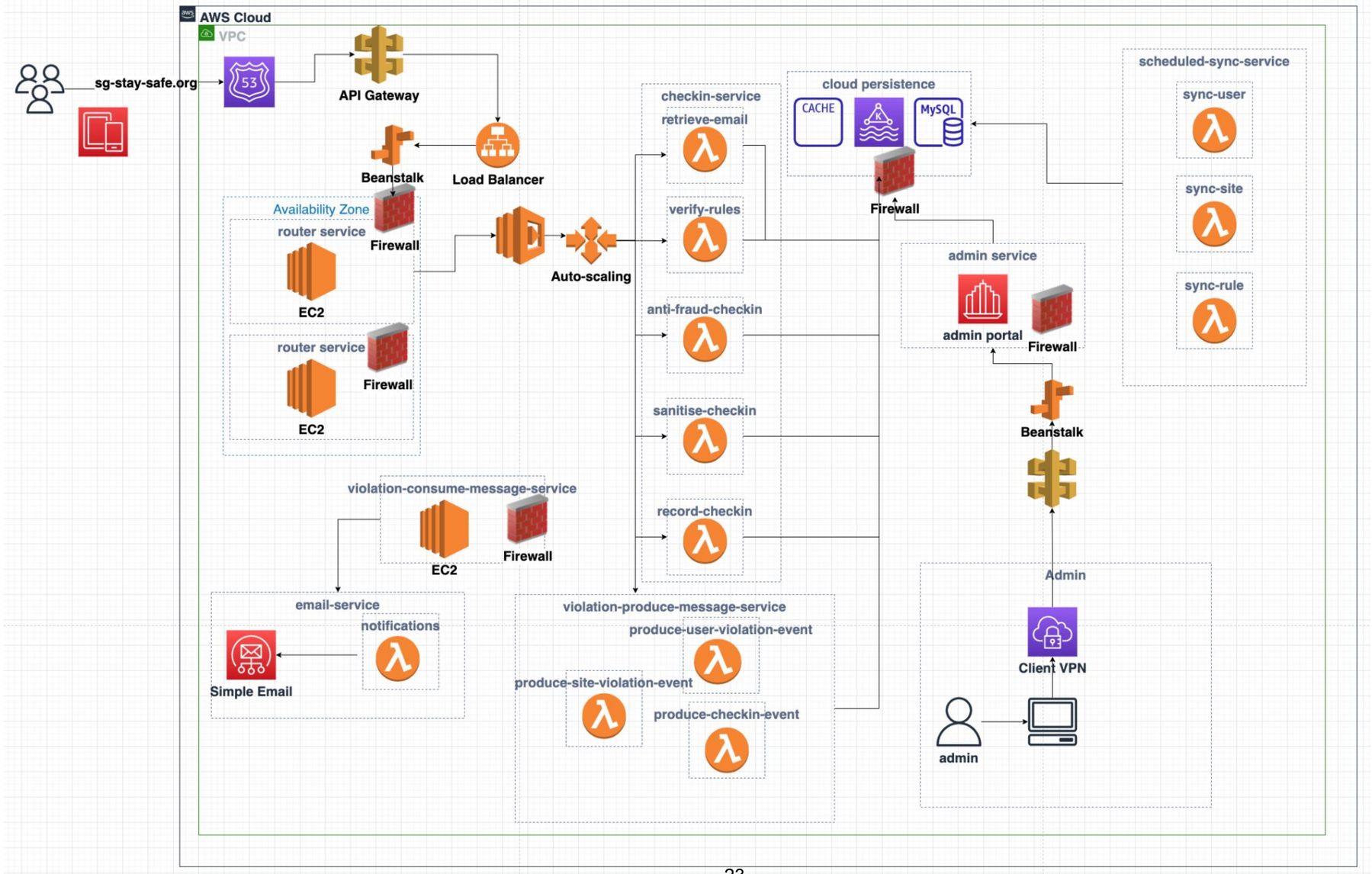
- Applications
  - Simple Email Service
- Framework
  - Go Gin, Django
- Runtime
  - Golang, Python
- Middleware
  - Elastic Beanstalk, Lambda
- Cloud Persistence
  - RDS, ElasticCache, MSK
- Virtualization, OS & Servers
  - Ubuntu, EC2
- Storage
  - S3
- Networking
  - VPC, Route53
- Auth, Monitoring and Logging
  - IAM, CloudWatch
- DevOps
  - CodeBuild & CodePipeline







Physical Architecture & Design 04
– Physical Architecture Diagram

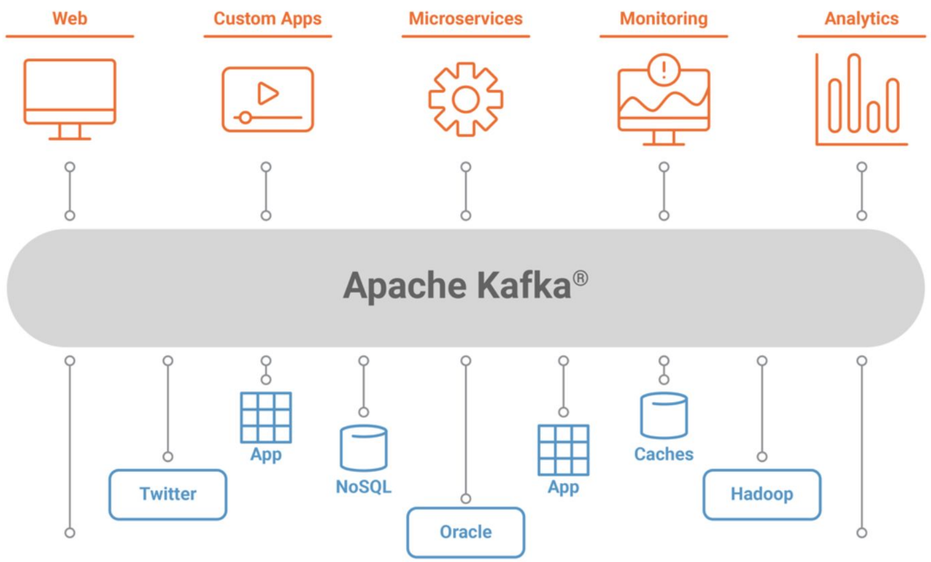


Physical Architecture & Design 04
– Persistence Design

Cloud Persistence

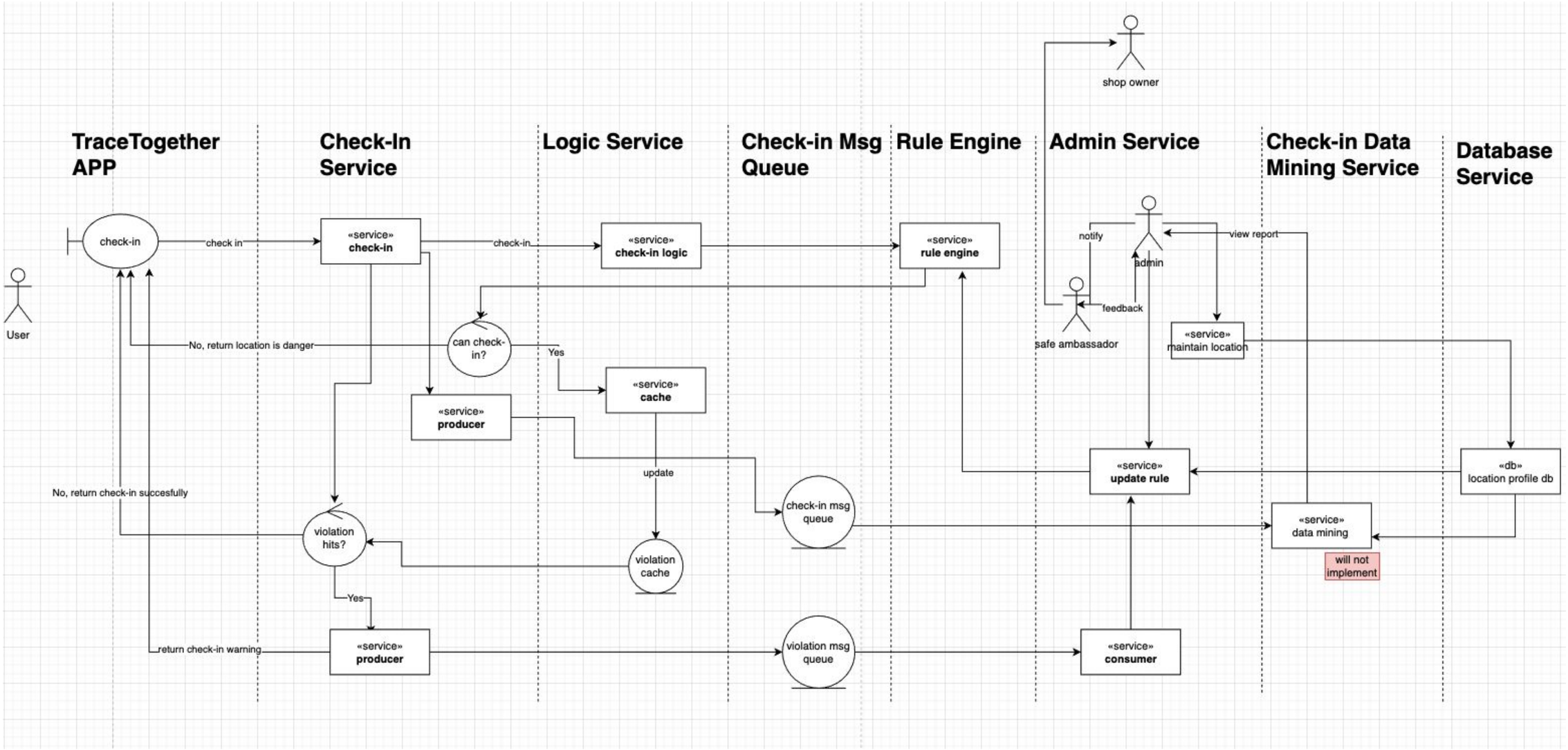
- Variety: The need for Non-static Schema[i]
  - Our check-in seeds are not in the traditional form of relational mode.
  - Data is stored in NoSQL schema
- Variability: The need for Agility and Heterogeneity[ii]
  - Data needs to be processed by different technologies many times in a day for different purpose.
  - Our system persist all data in Apache Kafka to fulfill the variability
- Volume of Data: The need for Scalability[i][ii]
  - Relational database is not friendly to horizontal scalability
  - Our business requires elastic horizontal scaling for scaling up and down
  - horizontal scaling increases the need for caching
- Velocity: The need for higher performance[i]
  - Our data model doesn't highly rely on data integration and constraints
  - Cache has extremely higher performance in key/value data read/write

Decision outcome for [i]: Cache Storage



Decision outcome for [ii]: Kafka Msg Storage

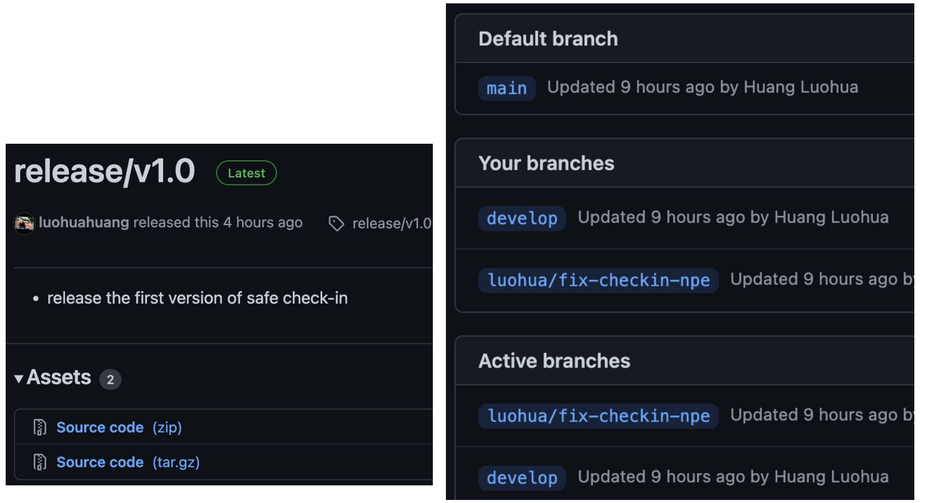
Physical Architecture & Design 04
– User Banned Use Case

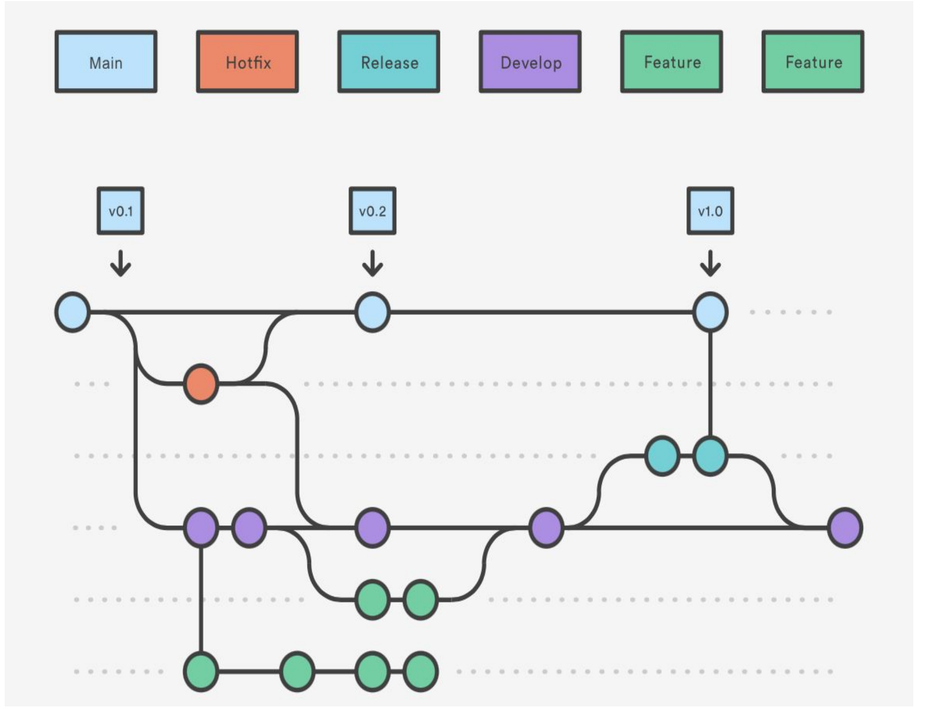


DevOps and Development Lifecycle

Apply Gitflow Workflow Branching Strategy

- Gitflow is designed for projects that have a scheduled release cycle and for the DevOps best practice of continuous delivery
- “master” (a.k.a “main”) branch
  - Always ensure that the repository’s release is from this branch
  - It stores the official release history
- “develop” (a.k.a “integrate”) branch
  - Derived from master branch
  - Development purpose for integrating multiple features for the next upcoming release
- “hotfix” branch
  - Used to quickly patch production releases
  - Hotfix branches are a lot like release branches and feature branches except they're based on main instead of develop
- “feature” branches
  - Derived from develop branch
  - Dev’s own dedicated branch for his/her ongoing development of a ticket
- releases/tags
  - Releases are deployable software iterations the team can package and make available for a wider audience to download and use
  - Releases are based on Git tags, which mark a specific point in our repository's history.





Elastic artifact storage management

- Leverage S3 service for scalability, data availability, security, and performance
  - No more manual repository/storage management
  - Use Amazon S3 buckets to store and retrieve any amount of data
  - For each bucket, easier control access to it (who can create, delete, and list objects in the bucket)
  - Restful API access to storage

```
13        - mkdir bin/
14        - GOOS=linux GOARCH=amd64 go build -o bin/application
15    post_build:
16      commands:
17        - mkdir -p ${CODEBUILD_SRC_DIR}/bin/
18        - cp bin/application ${CODEBUILD_SRC_DIR}/bin/
19  artifacts:
20    files:
21      - bin/**/*
```

Upload to S3 post build

Elastic Beanstalk > Applications > router > Application versions

Application versions — Deploy with S3 artifact — Actions | Settings | Upload

Version label	Description	Date created	Source
code-pipeline-1650682938770-BuildArtifact-e4fa940d-2158-4a55-9092-041133ece664	BuildArtifact-e4fa940d-2158-4a55-9092-041133ece664	2022-04-23T11:02:19+08:00	eb-checkin-router/BuildArtif/NWgHZ9Z
code-pipeline-1650682270467-BuildArtifact-8baef12f-98ff-48aa-8bd4-5c46bcfc676b	BuildArtifact-8baef12f-98ff-48aa-8bd4-5c46bcfc676b	2022-04-23T10:51:11+08:00	eb-checkin-router/BuildArtif/rybMUQF
code-pipeline-1650681773651-BuildArtifact-880ad35d-305c-4b40-8bf3-f5b636519d15	BuildArtifact-880ad35d-305c-4b40-8bf3-f5b636519d15	2022-04-23T10:42:54+08:00	eb-checkin-router/BuildArtif/pLl4eoI
code-pipeline-1650679606555-BuildArtifact-f7904075-69f3-41b4-bece-bc4cc9a28eaa	BuildArtifact-f7904075-69f3-41b4-bece-bc4cc9a28eaa	2022-04-23T10:06:47+08:00	eb-checkin-router/BuildArtif/zD2GiPh
code-pipeline-1650678319473-BuildArtifact-ca429ed7-91a8-415b-b18c-513526f3551a	BuildArtifact-ca429ed7-91a8-415b-b18c-513526f3551a	2022-04-23T09:45:20+08:00	eb-checkin-router/BuildArtif/b6lut9K
code-pipeline-1650677691649-BuildArtifact-c10377cd-a187-4071-afec-2a01cd4dbaf9	BuildArtifact-c10377cd-a187-4071-afec-2a01cd4dbaf9	2022-04-23T09:34:52+08:00	eb-checkin-router/BuildArtif/b6lut9K
code-pipeline-1650677589669-BuildArtifact-4601610c-9669-4d45-86e9-409a5745197c	BuildArtifact-4601610c-9669-4d45-86e9-409a5745197c	2022-04-23T09:33:10+08:00	eb-checkin-router/BuildArtif/b6lut9K

Amazon S3 > Buckets > codepipeline-ap-southeast-1-871542363442 > eb-checkin-router/ > BuildArtif/

BuildArtif/ — S3 Build Artifact Dashboard — Copy S3 URI

Objects | Properties

Objects (5)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

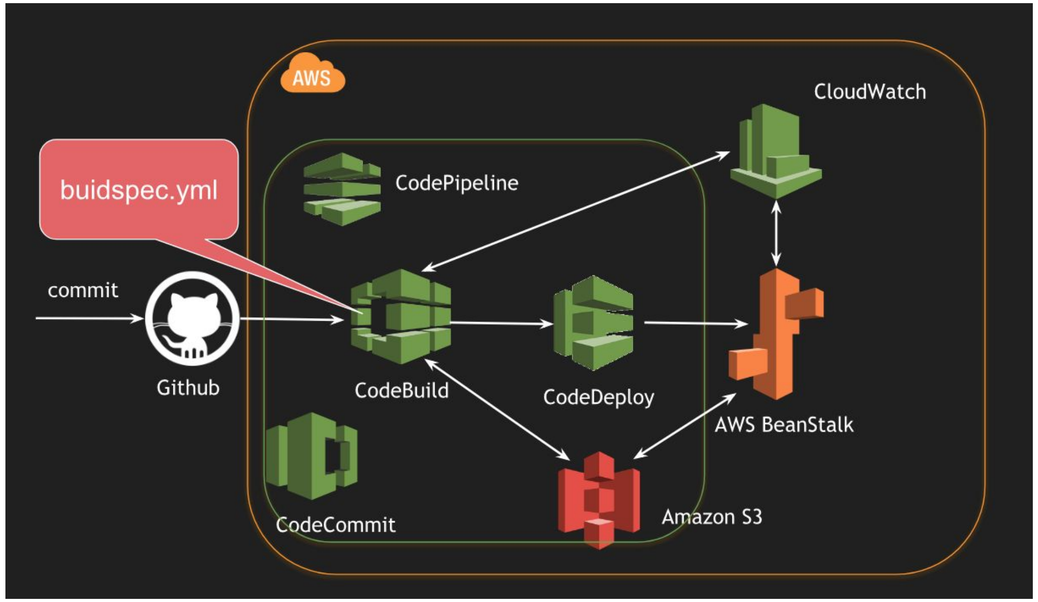
Copy S3 URI | Copy URL | Download | Open | Delete | Actions | Create folder | Upload

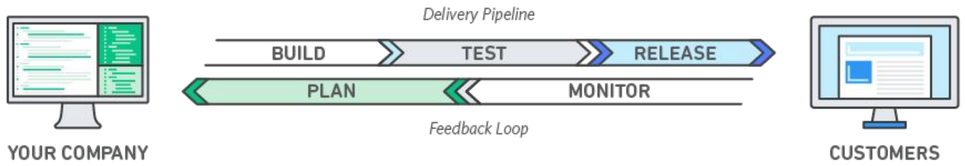
Find objects by prefix — Show versions — 1

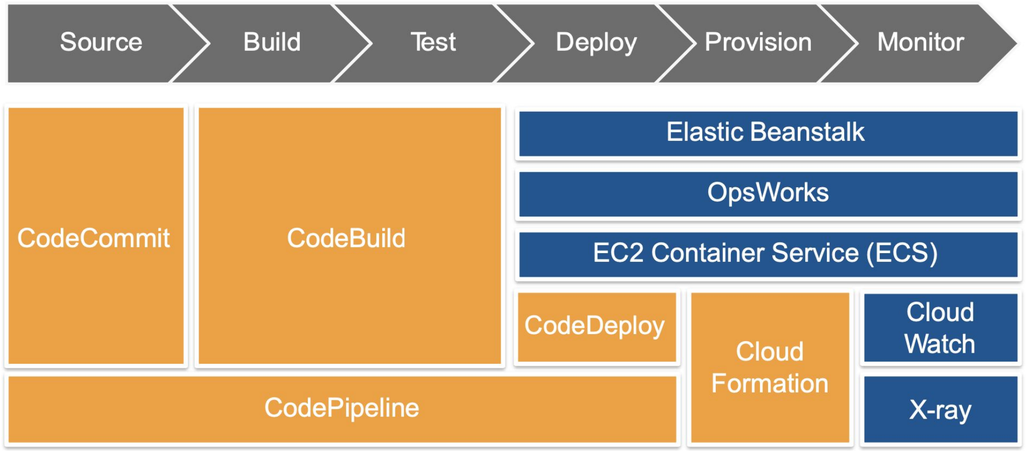
Name	Type	Last modified	Size	Storage class
NWgHZ9Z	-	April 23, 2022, 11:02:07 (UTC+08:00)	6.3 MB	Standard
rybMUQF	-	April 23, 2022, 10:50:55 (UTC+08:00)	6.3 MB	Standard
pLl4eoI	-	April 23, 2022, 10:42:21 (UTC+08:00)	6.3 MB	Standard
zD2GiPh	-	April 23, 2022, 10:06:19 (UTC+08:00)	6.3 MB	Standard
b6lut9K	-	April 23, 2022, 09:32:55 (UTC+08:00)	6.3 MB	Standard

Build an Automated, Agile and Observability DevOps

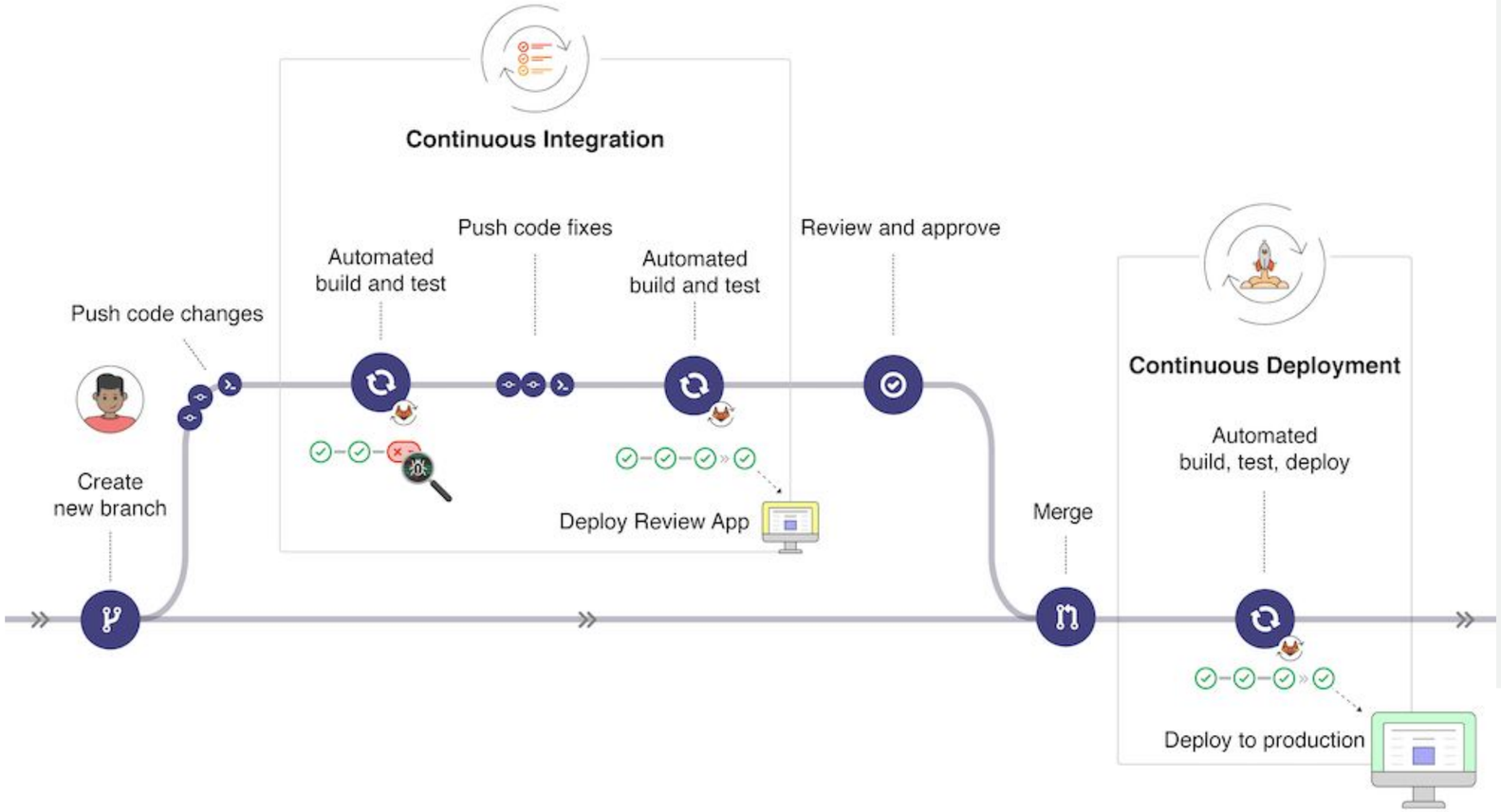
- Leverage S3 service for scalability, data availability, security, and performance
  - Continuous Integration
    - Github -> buildspec.yml -> CodeBuild -> S3
  - Continuous Delivery
    - S3 -> CodeDeploy/CodePipeline -> AWS Beanstalk
  - Monitoring and Logging
    - All logs -> CloudWatch
  - Communication and Collaboration
    - We embrace DevOps cultural model, which emphasizes values such as ownership and accountability. We collaborate closely, share responsibilities, and combine workflows

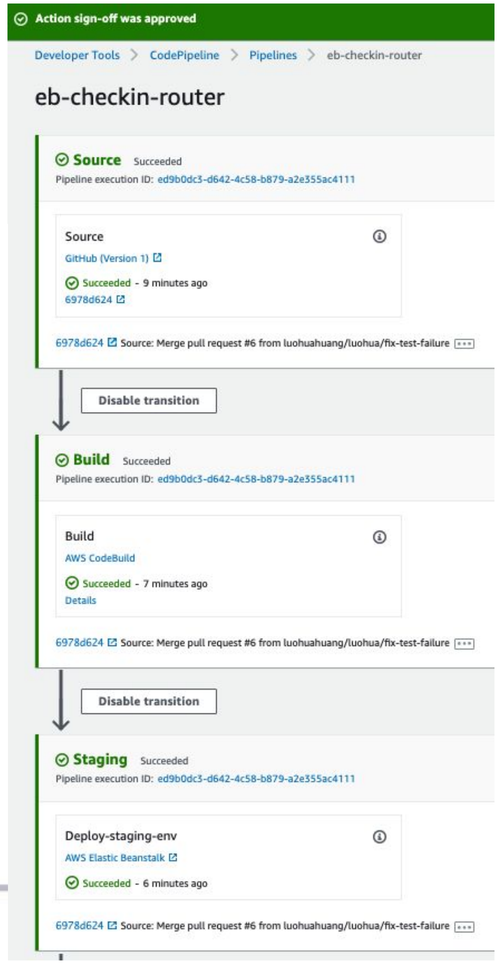


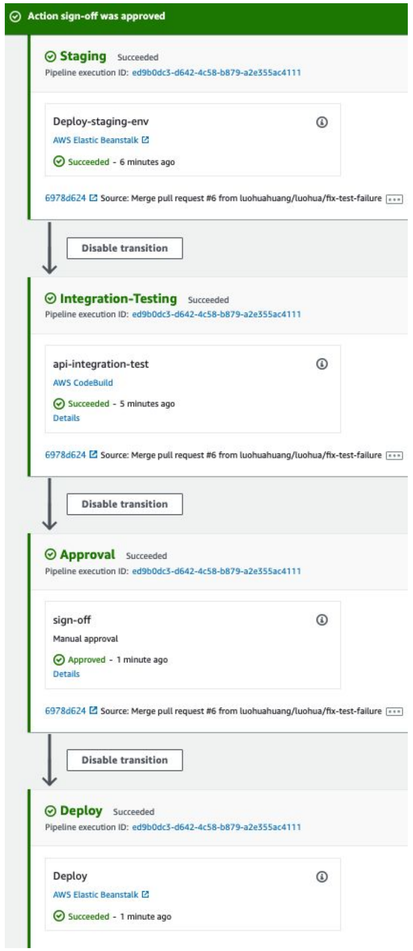




DevOps and Development Lifecycle 05 – CICD, SDLC and Pipeline



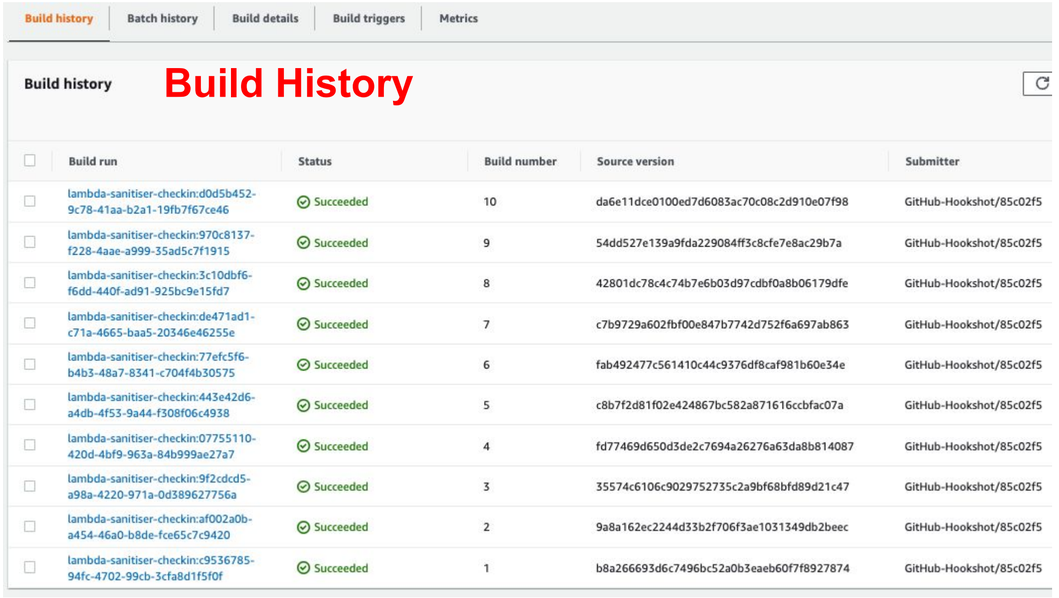


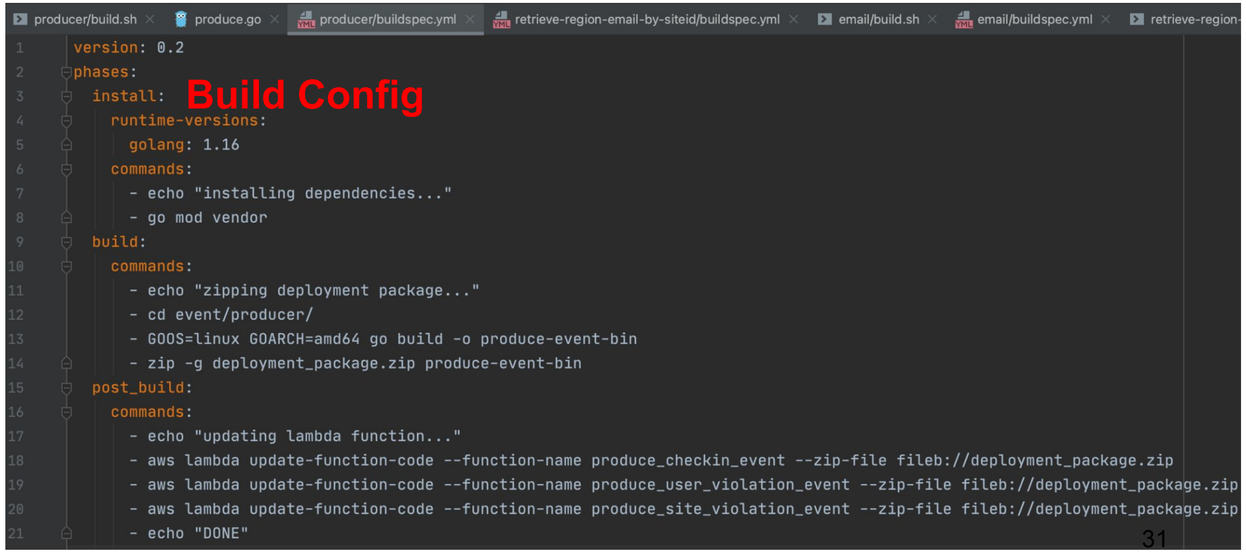


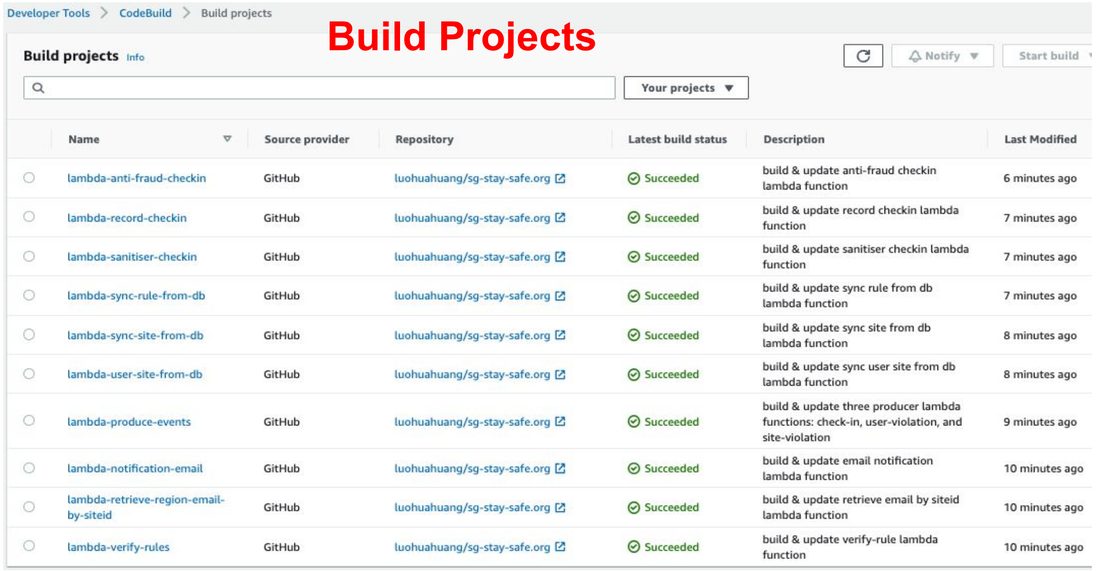
DevOps and Development Lifecycle 05
– Code Build

Build and test code with elastic scaling

- Leverage AWS CodeBuild to setup fully managed continuous integration services
  - No need to provision, manage, and scale our own build servers
  - CodeBuild scales continuously and processes multiple builds concurrently, no more builds waiting in a queue
  - Compiles source code, and update lambda functions in a pipeline



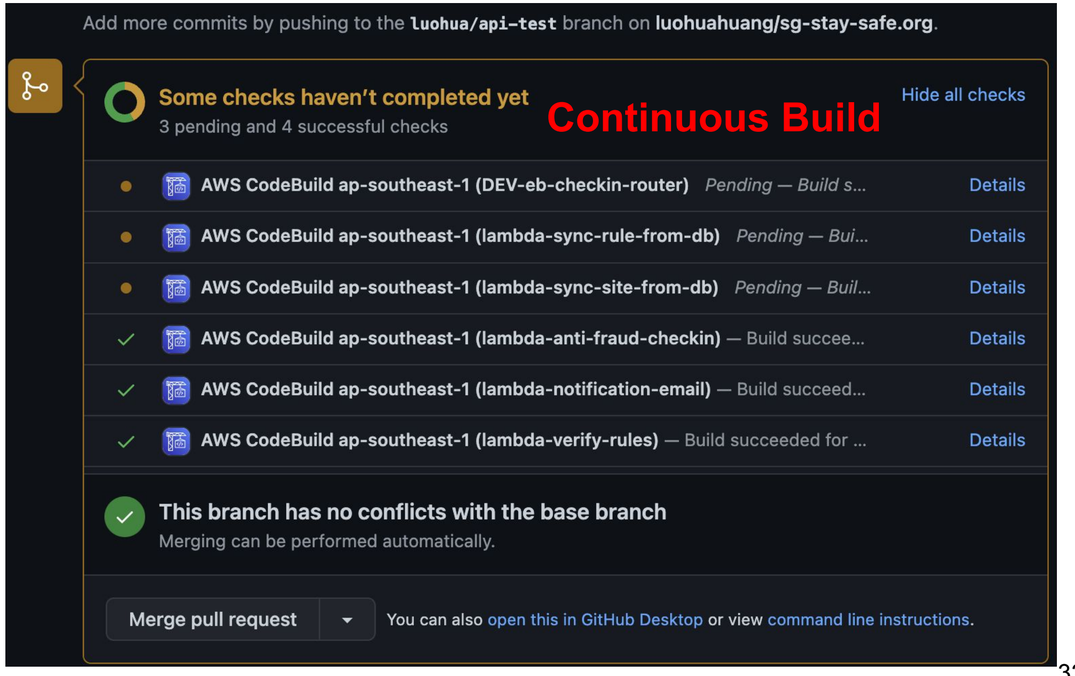


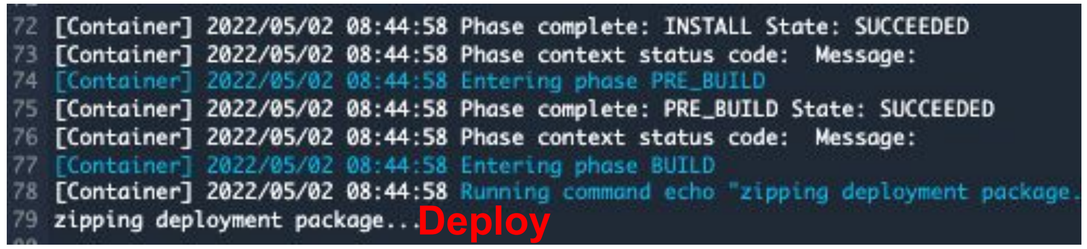


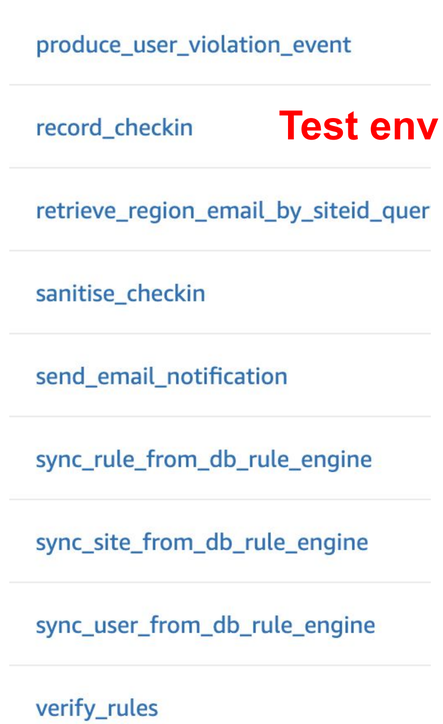
DevOps and Development Lifecycle 05
– Code Pipeline

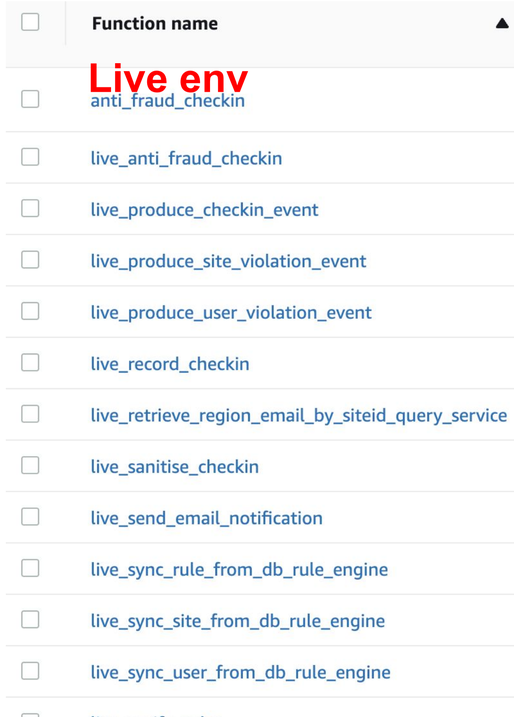
Continuous Build, Deploy, Test and Integration

- Leverage AWS CodePipeline to setup continuous integration and continuous delivery service
  - Fast and reliable application and infrastructure updates
  - Builds, tests, and deploys the code every time there is a code change, based on the release process models we define
  - Configurable workflow and easy to integrate









DevOps and Development Lifecycle 05
– Code Pipeline

Continuous API Testing and Performance Test

Options | Download Test (CSV) | Search: Test/Class/Package

Chart	Package/Class/Testmethod	Passed	Transitions	1
☐	(root)	100% (100%)	0	595.450
☐	check-in	100% (100%)	0	595.450
☐	Test_checkin_anti_fraud_fresh_user_different_site_first_time_login	100% (100%)	0	0.290
☐	Test_checkin_anti_fraud_fresh_user_different_site_revisit	100% (100%)	0	0.320
☐	Test_checkin_anti_fraud_fresh_user_random_site_second_time_login	100% (100%)	0	0.170
☐	Test_checkin_normal_fresh_user_random_site_first_time_login	100% (100%)	0	1.930
☐	Test_checkin_normal_fresh_user_random_site_second_time_login_after_enough_time	100% (100%)	0	301.770
☐	Test_checkin_normal_fresh_user_random_site_second_time_login_after_not_enough_time	100% (100%)	0	290.490
☐	Test_checkin_sanitiser_anonymous_id_is_empty_only	100% (100%)	0	0.420
☐	Test_checkin_sanitiser_corrupted_request	100% (100%)	0	0.000
☐	Test_checkin_sanitiser_site_id_and_anonymous_id_both_empty	100% (100%)	0	0.030
☐	Test_checkin_sanitiser_site_id_is_empty_only	100% (100%)	0	0.030

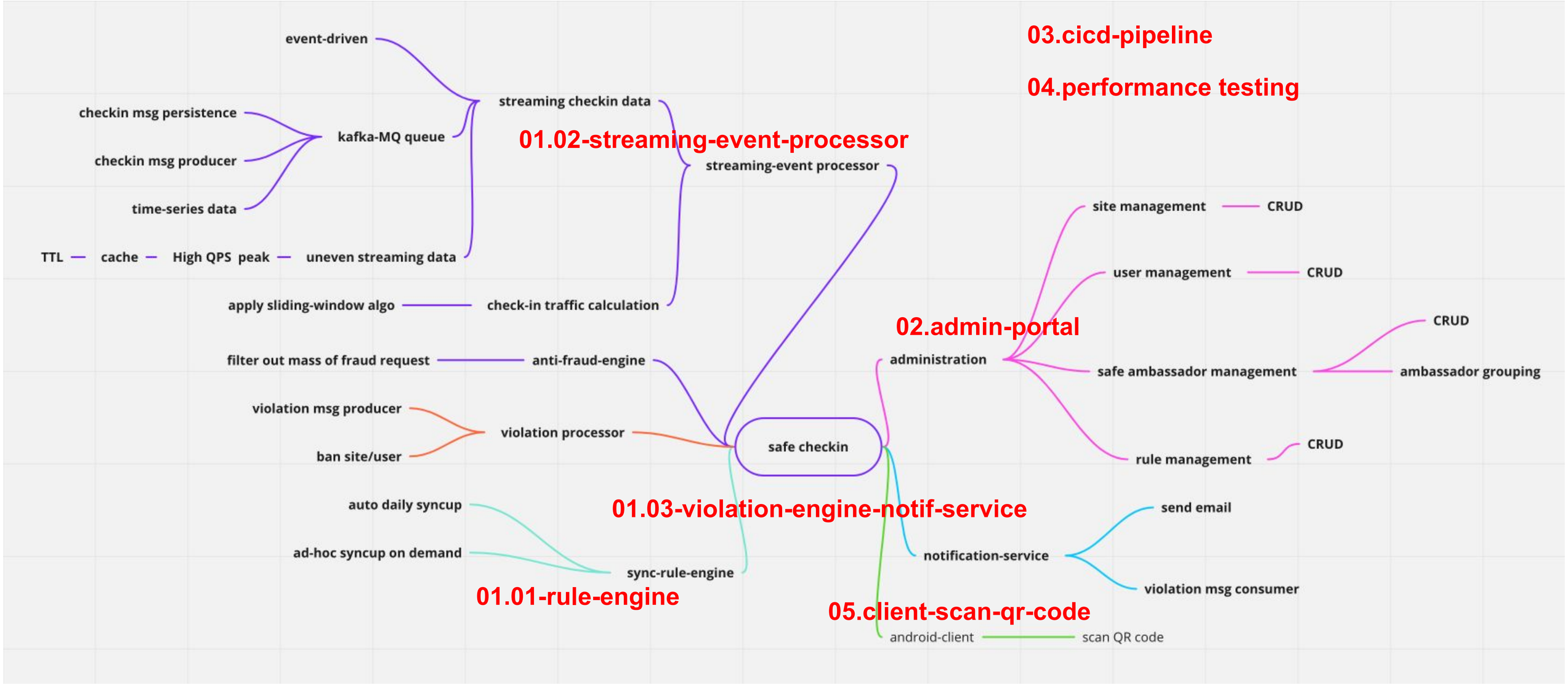
APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
0.002	500 ms	1 sec 500 ms	Total
0.002	500 ms	1 sec 500 ms	check-in

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	1000	998	99.80%	67.61	41	1086	54.00	85.00	157.80	288.95	16.72	4.21	4.64
check-in	1000	998	99.80%	67.61	41	1086	54.00	85.00	157.80	288.95	16.72	4.21	4.64

Demo

– Demo video: demo-video-for-all.mp4 (1.25x speed, elapses 9:29). All original videos available from here



Contents Page