

# Safe Check-in Platform

Stay Safe-Check in Safely

Team 2, 5<sup>th</sup> 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

Physical Architecture & Design

04

Project Conduct

02

DevOps & Deployment Lifecycle

05

Logical Architecture & Design

03

DEMO!

06

## 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



### 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

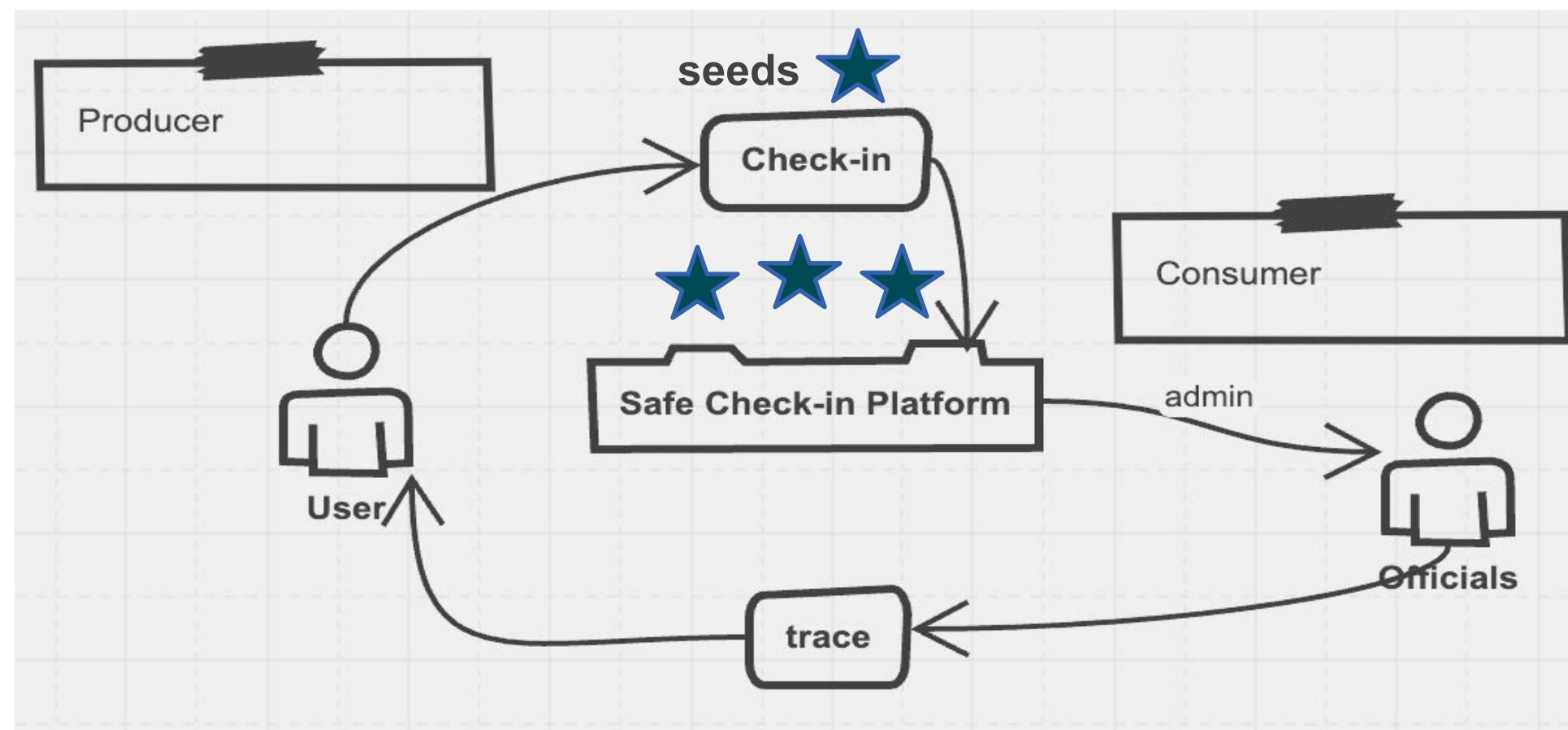
- Targeted deployment of Safe Distancing Ambassadors

# OVERVIEW 01

## – Platform Thinking



- All users who need to check-in before entering a site
- Check-in events, site information, government rules
- Government officials and safe ambassadors



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



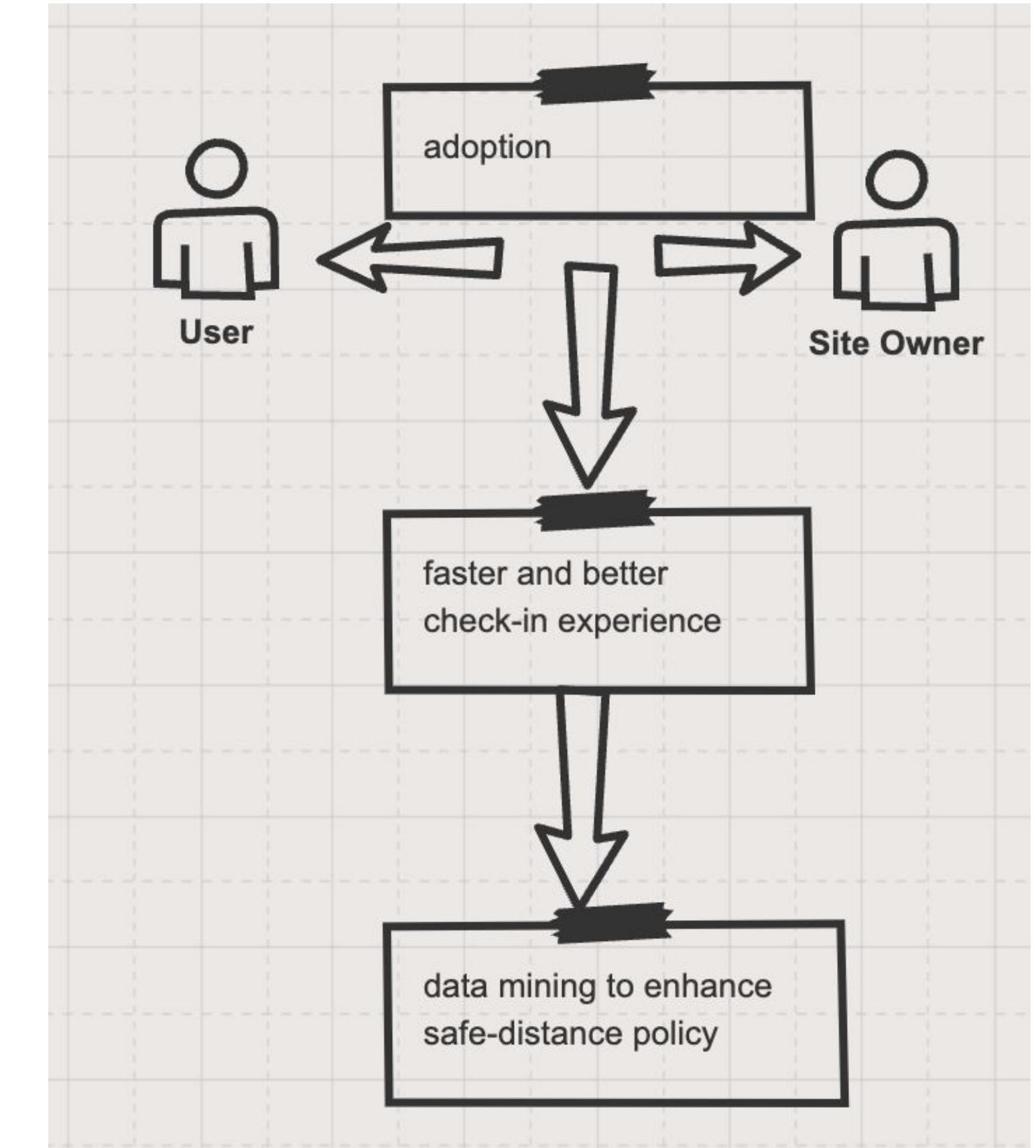
# OVERVIEW 01

## – Platform Thinking

### Scaling Platform

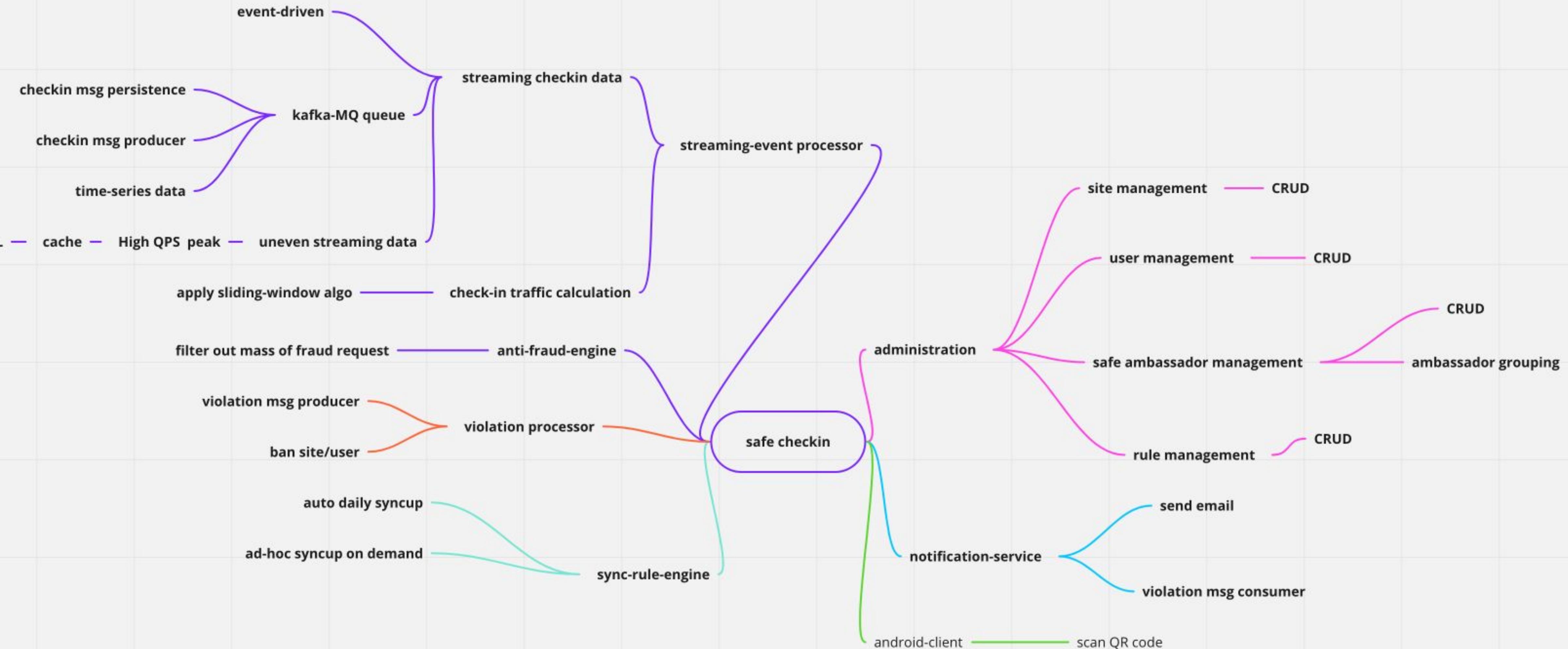
- First start with check-in regulation, and then we will scale our business to support covid contacting trace measures, apply big data to detect highly vulnerable sites, and deploy preventive measures

### Platform Ecosystem



# OVERVIEW 01

## – Functional Scope and Challenges

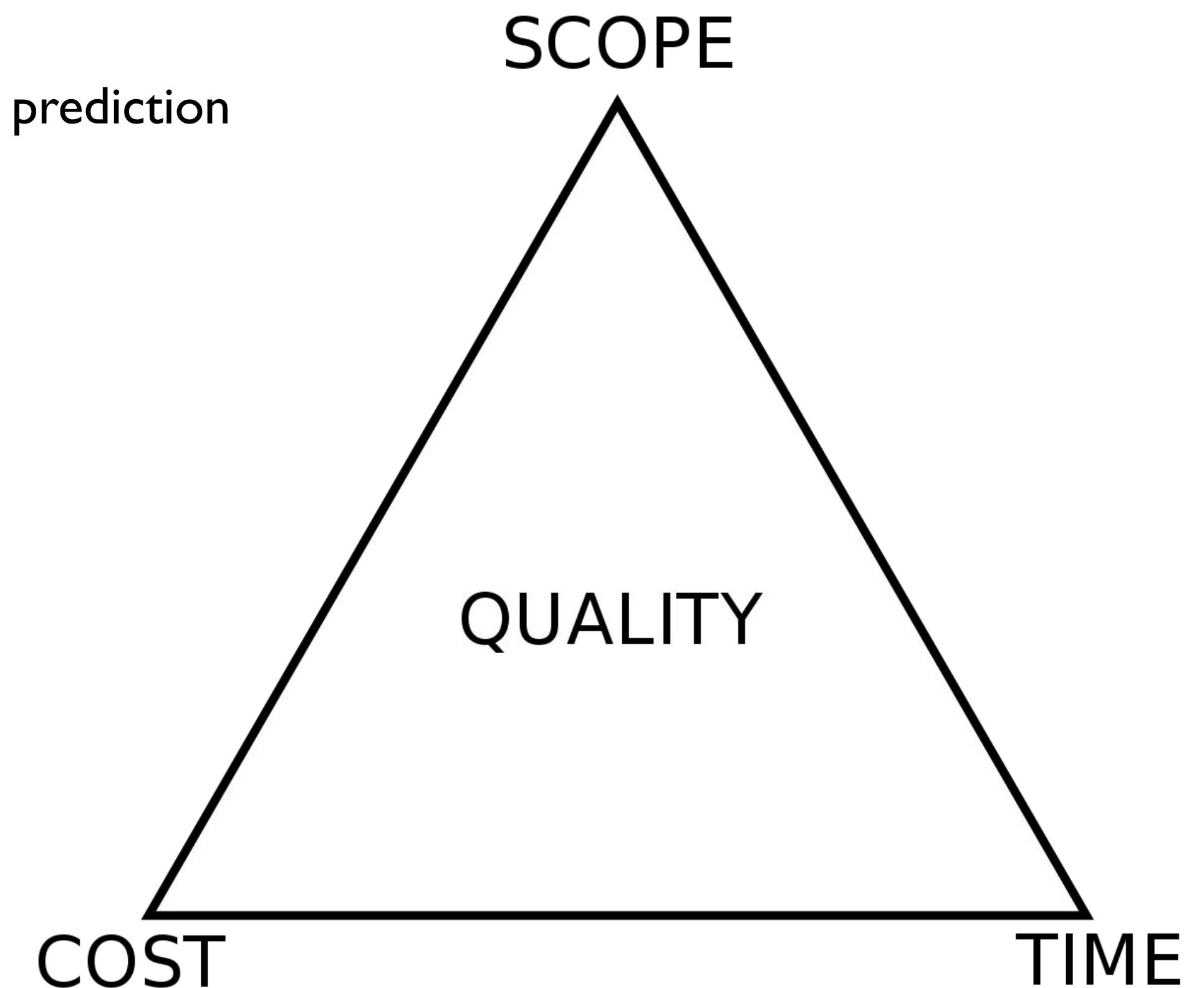


## OVERVIEW 01

### – Functional Out of Scope

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

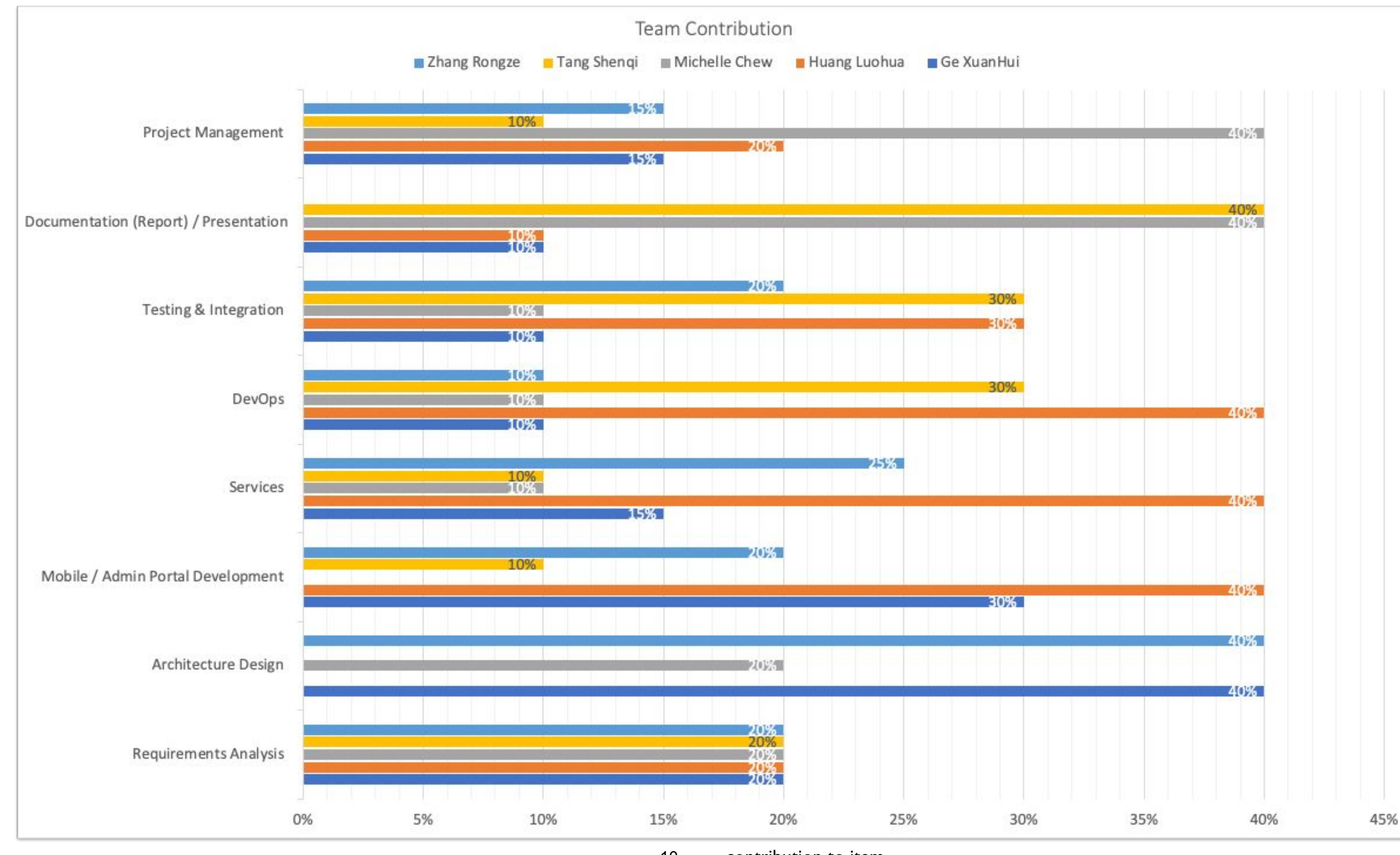
- Data Analytics of Check-in events, allowing for cloud mining cluster prediction
- Mobile App for Officials to conduct on-site check
- Token devices for check in
- Privacy concerns of Safe Check in data are accounted for



# Project Conduct

# Project Conduct 02

## – Team Contribution



# Project Conduct 02

## - WBS & Challenges

Work Breakdown Structure													
Project Title	Safe Check In Platform												
Date	5th May												
WBS Number	Task Title	Start Date	Due Date	Duration	% of Task	Week 1	Week 2	Week 3	Week 4	Week 1	Week 2	Week 3	Week 4
	Task Title	Start Date	End Date	Duration	Completion	M	T	W	T	F	S	S	M
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 Components			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 Preparation			0	100%								

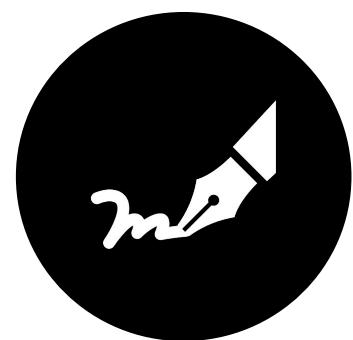
## Challenges

- 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

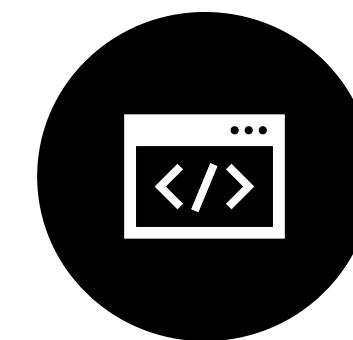


Define Mobile App and Admin Portal

- 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 CI/CD pipeline for build/deploy
- Manage infra and artifacts
- Conduct SCM for the platform

100%

### Testing & Integration



Integration testing, verify the functions

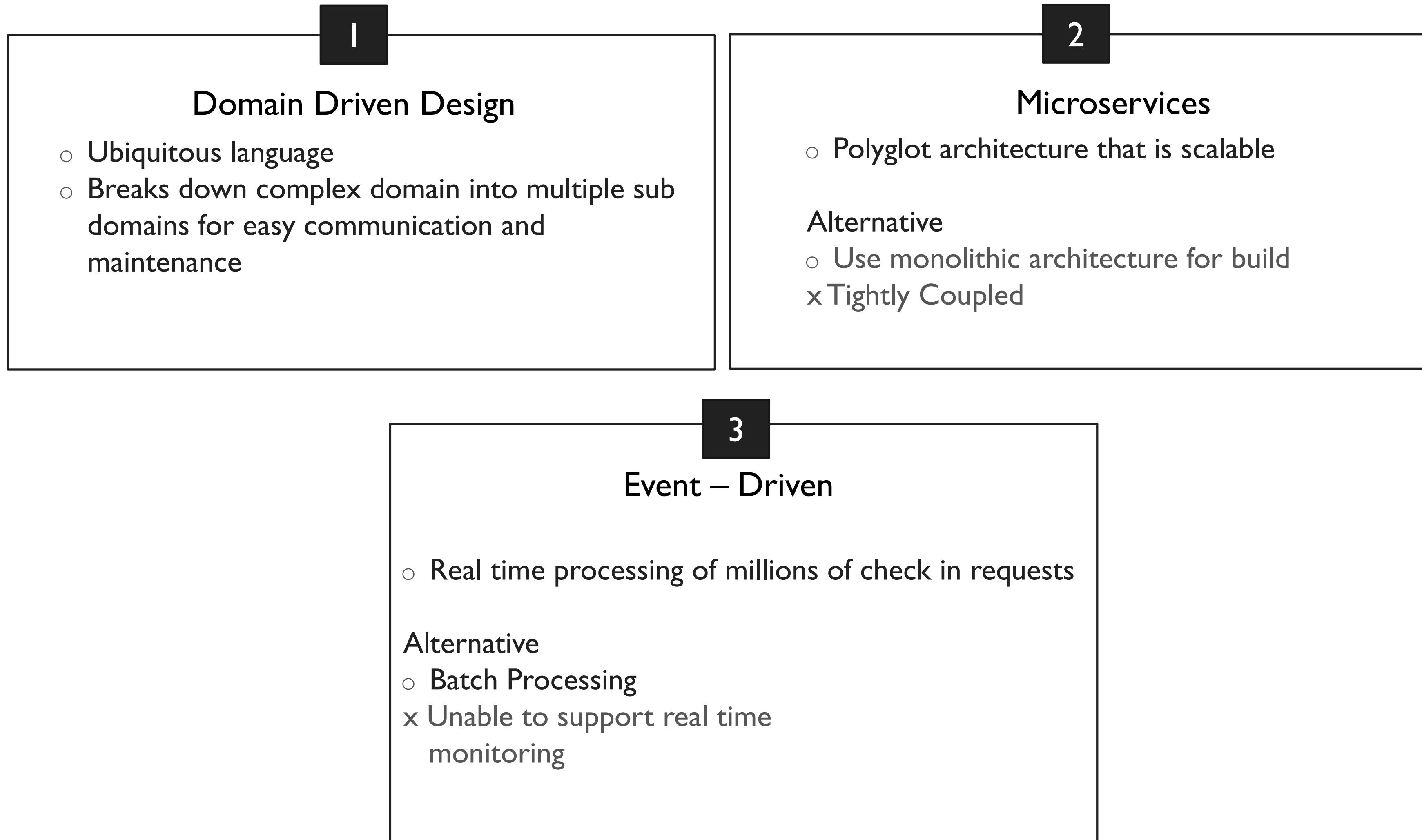
- Conduct integration testing for all components
- Verify all implemented functions and fix bugs

100%

# Logical Architecture & Design

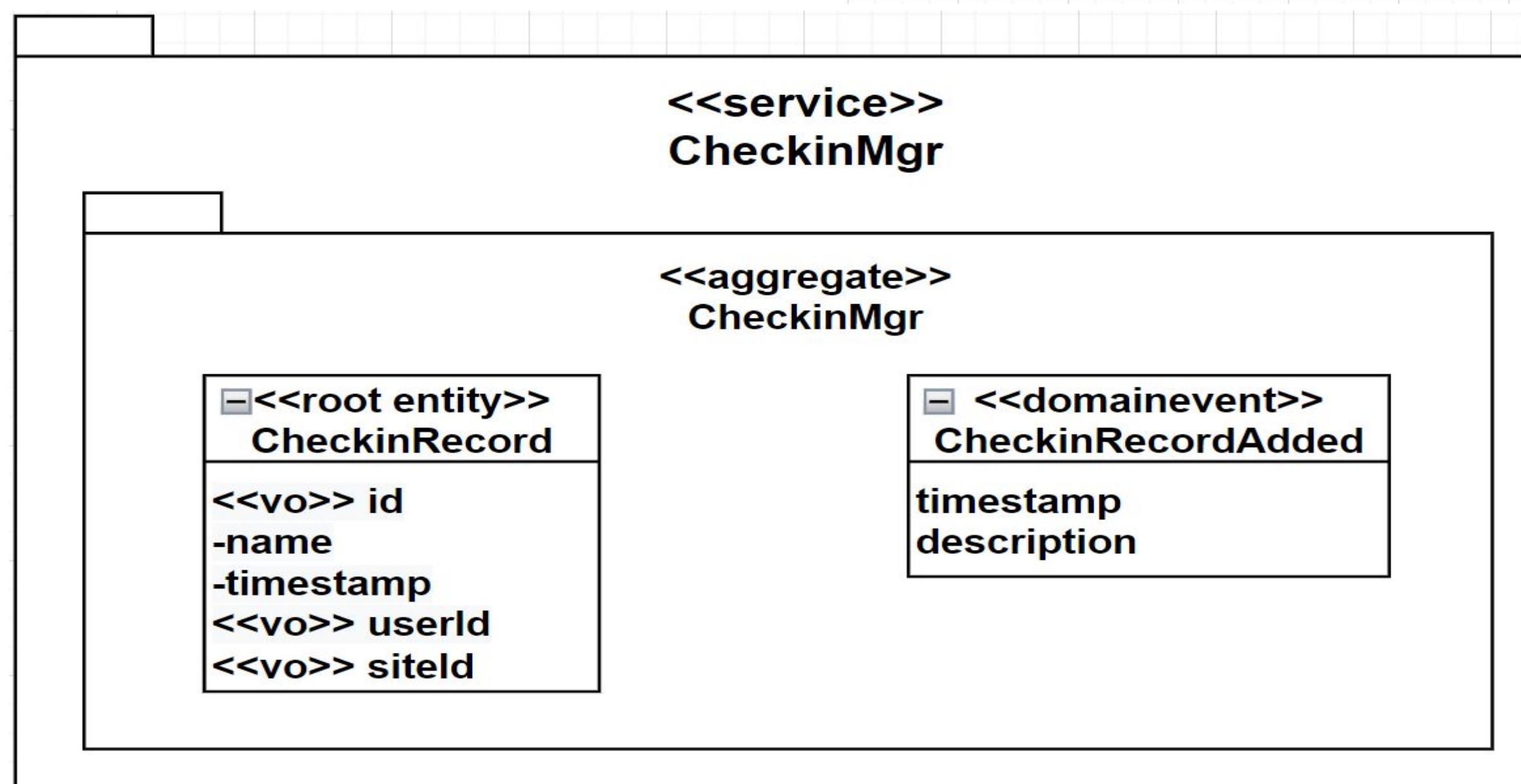
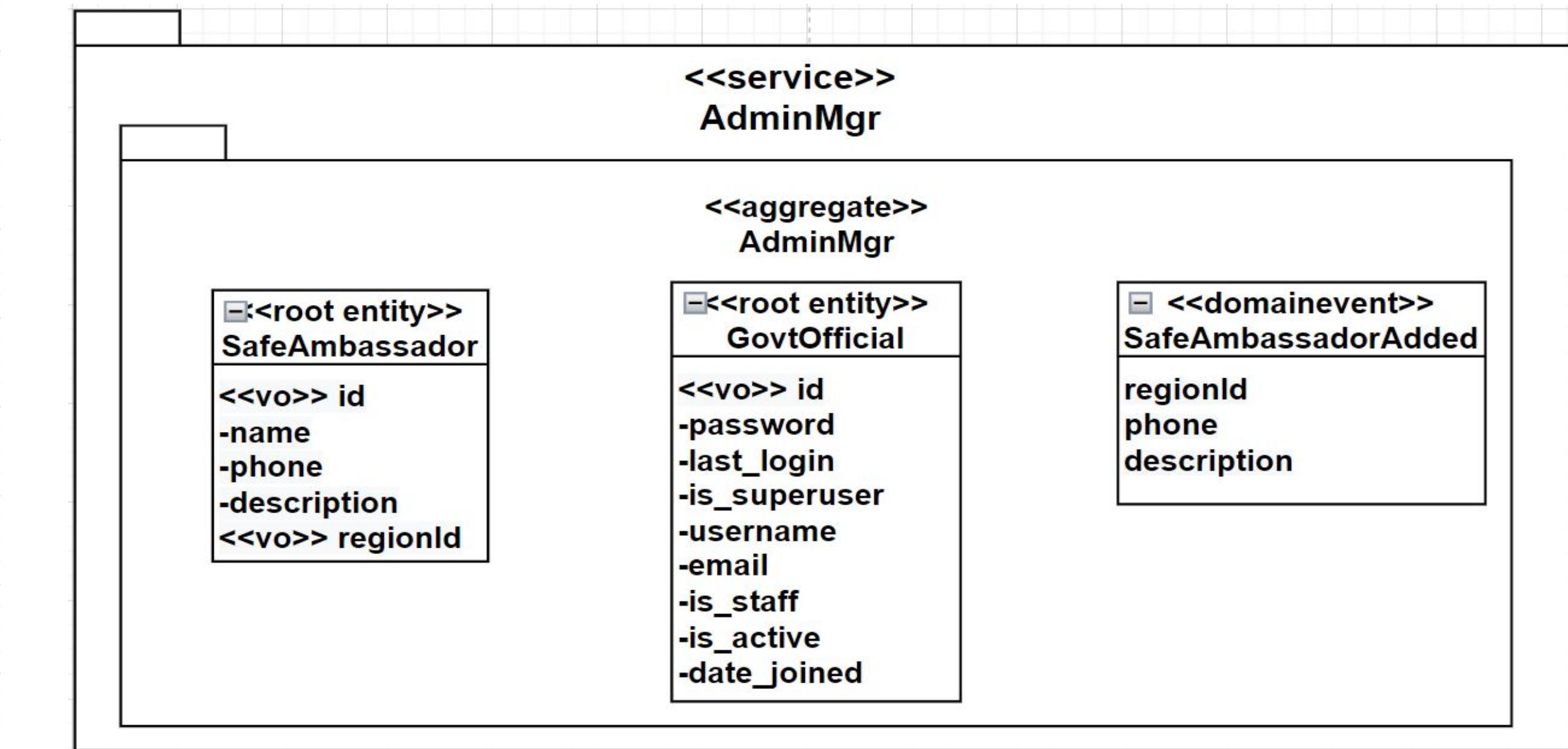
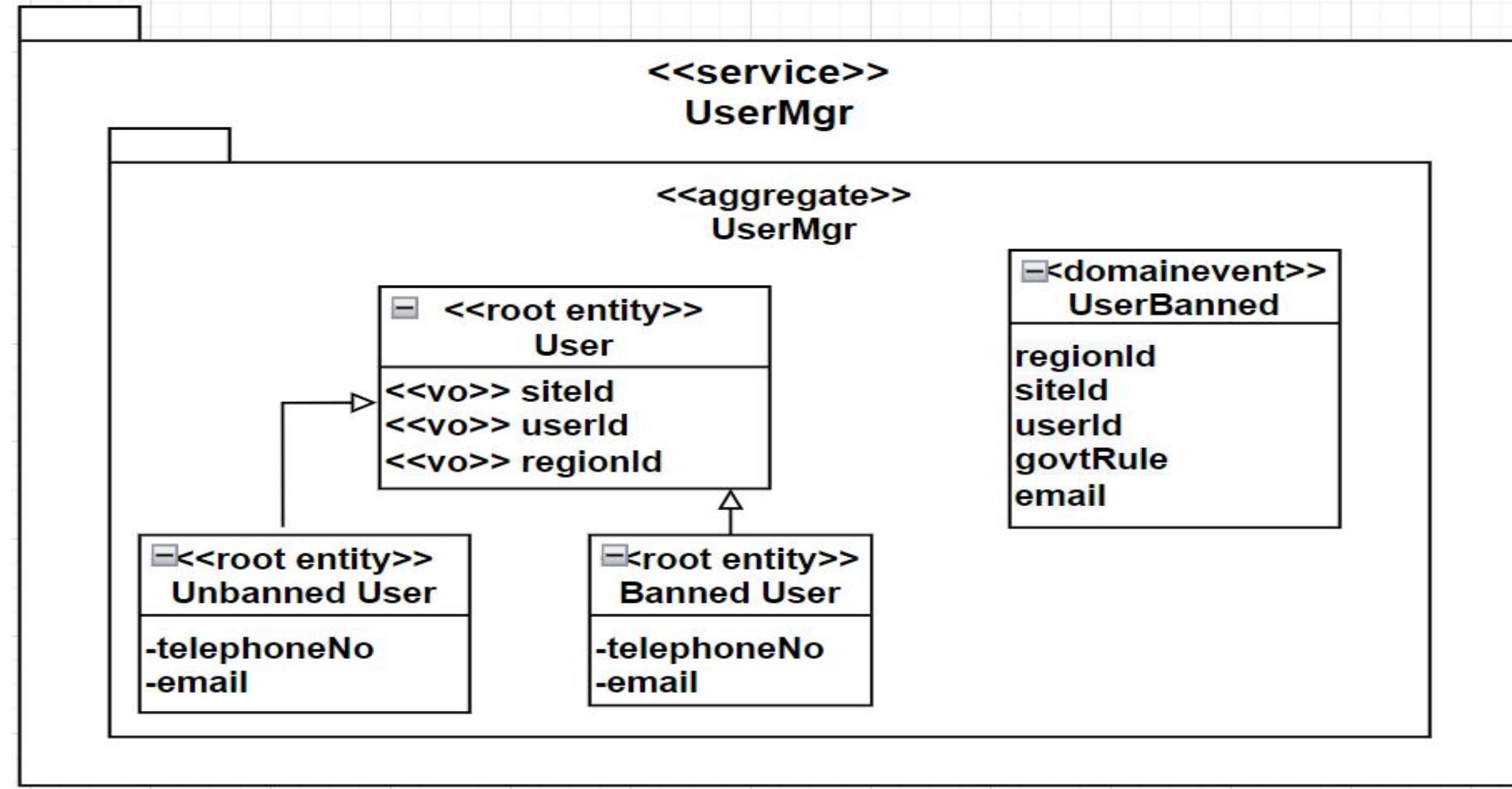
# Logical Architecture & Design 03

## – Key Architectural Decisions



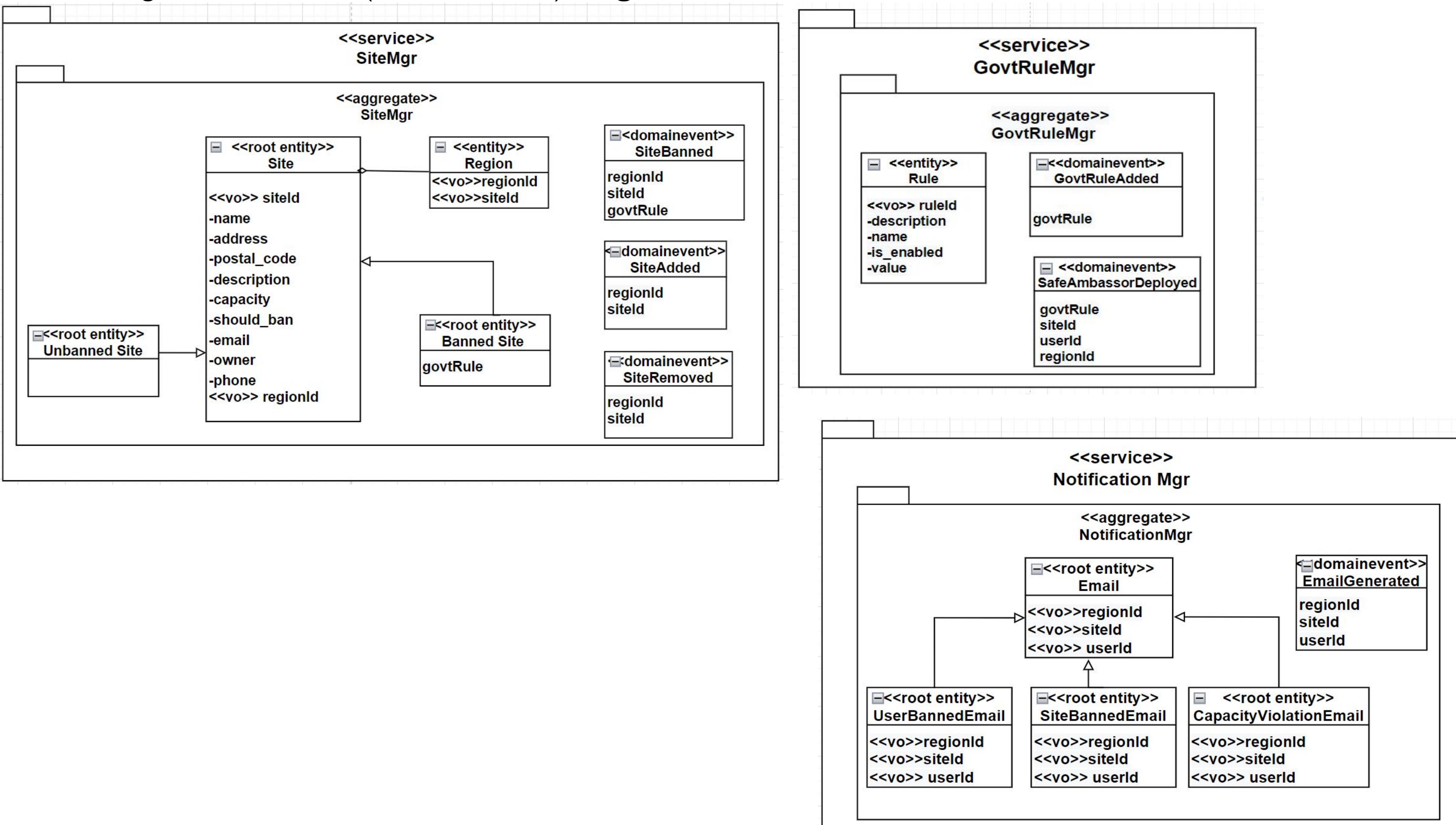
# Logical Architecture & Design 03

## – Logical Architecture (Domain Model) Producers and Consumers, Seeds



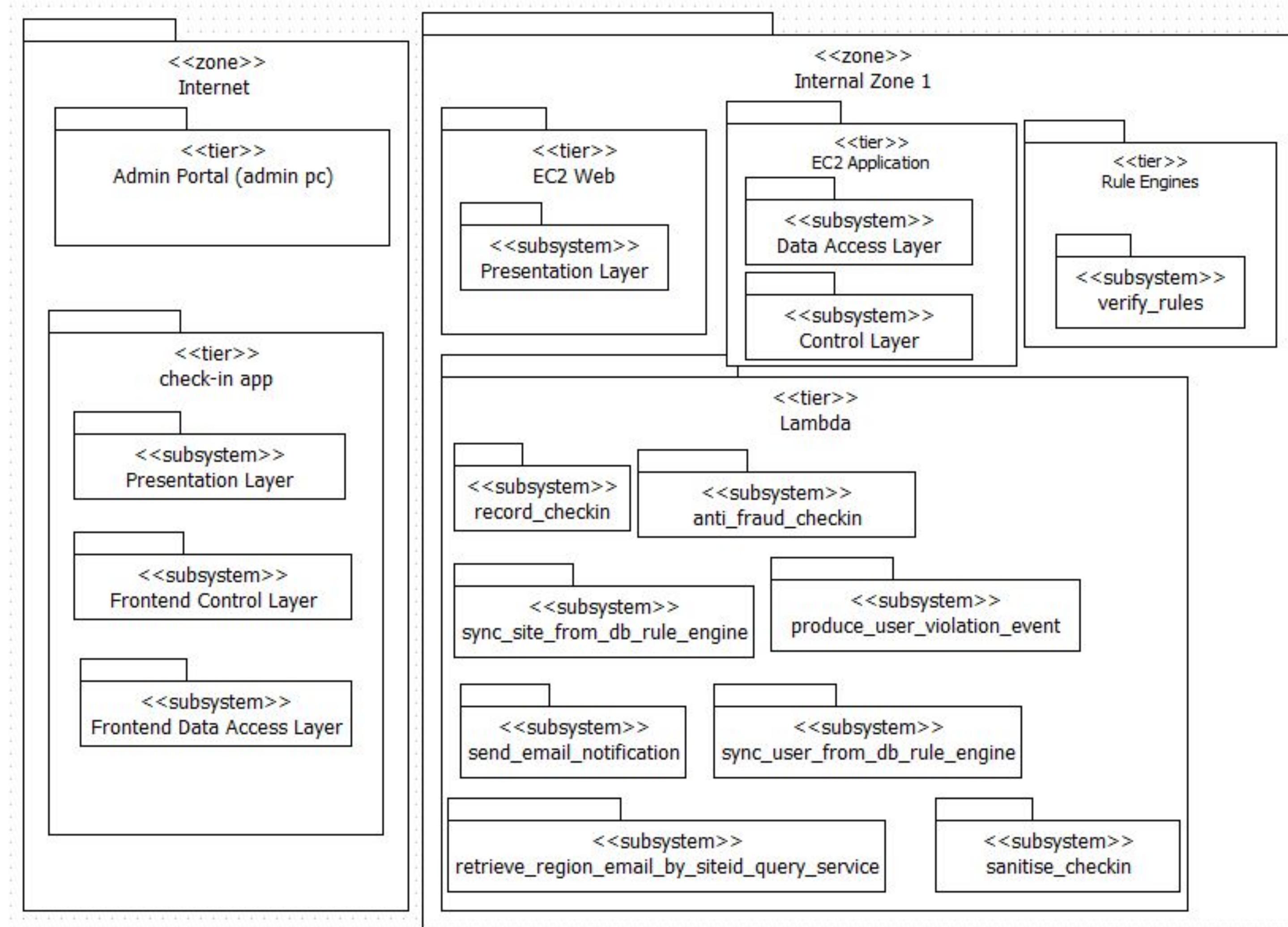
# Logical Architecture & Design 03

## – Logical Architecture (Domain Model) Magnet/Toolbox



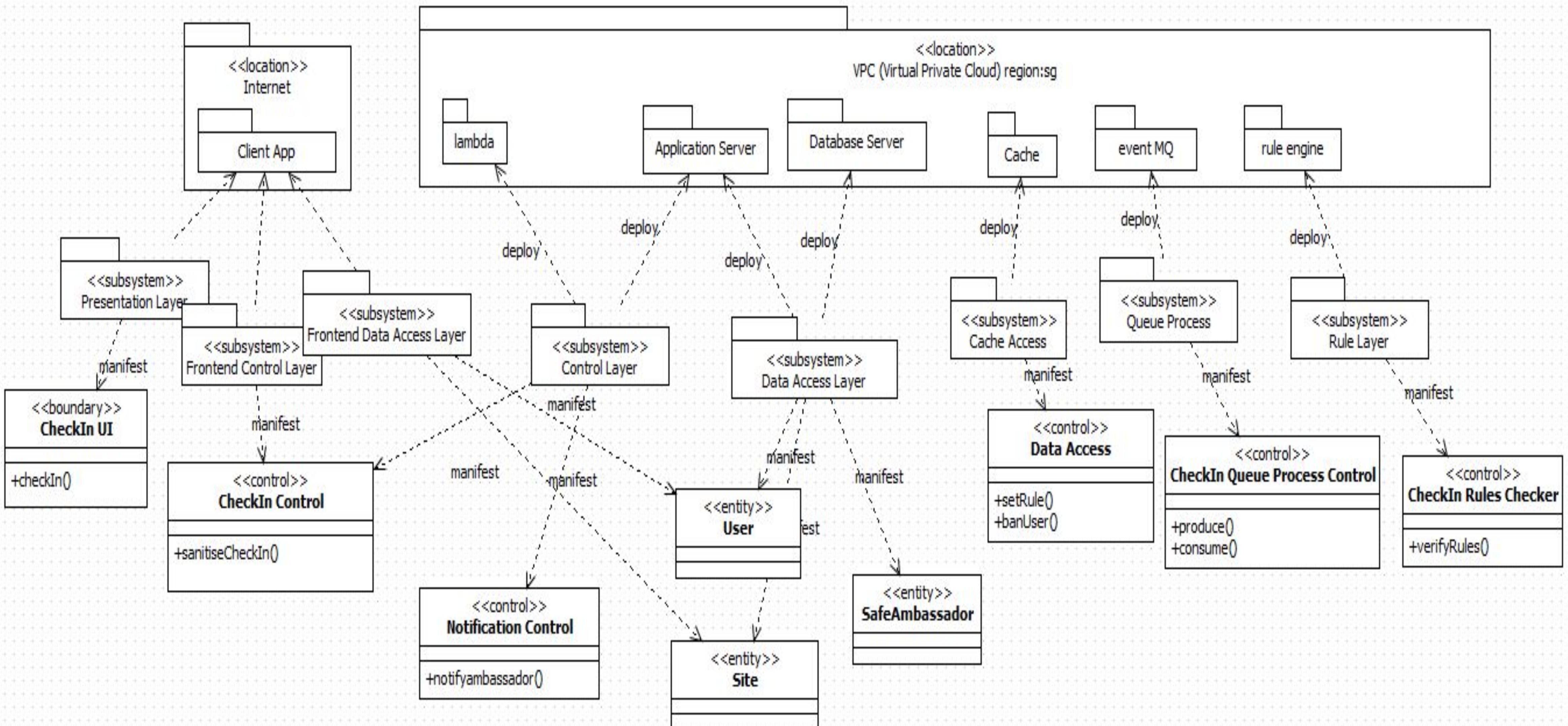
# Logical Architecture & Design 03

## – Overview



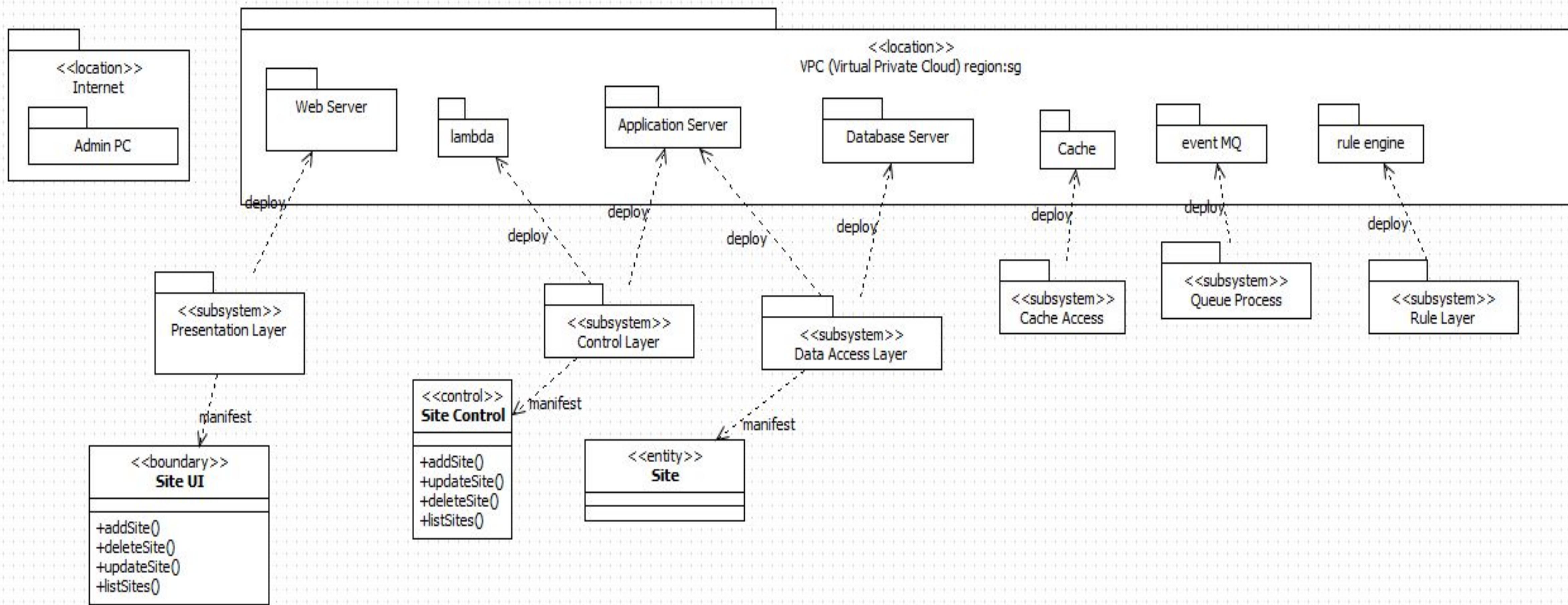
# Logical Architecture & Design 03

## – Detailed deployment diagram User Check- in



# Logical Architecture & Design 03

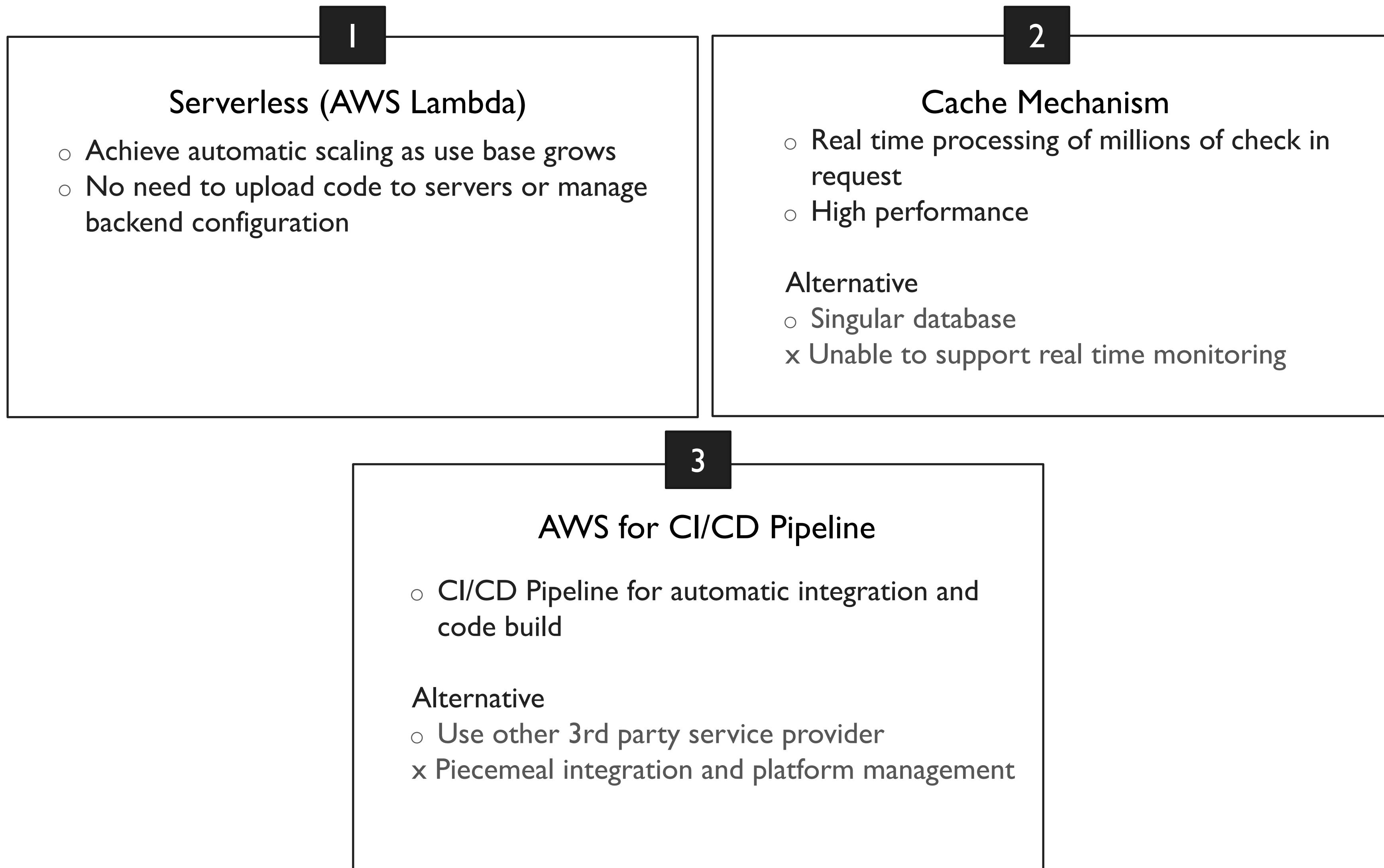
## – Detailed deployment diagram Admin Portal Site CRUD



# Physical Architecture & Design

# Physical Architecture & Design 04

## – Key Architectural Decisions

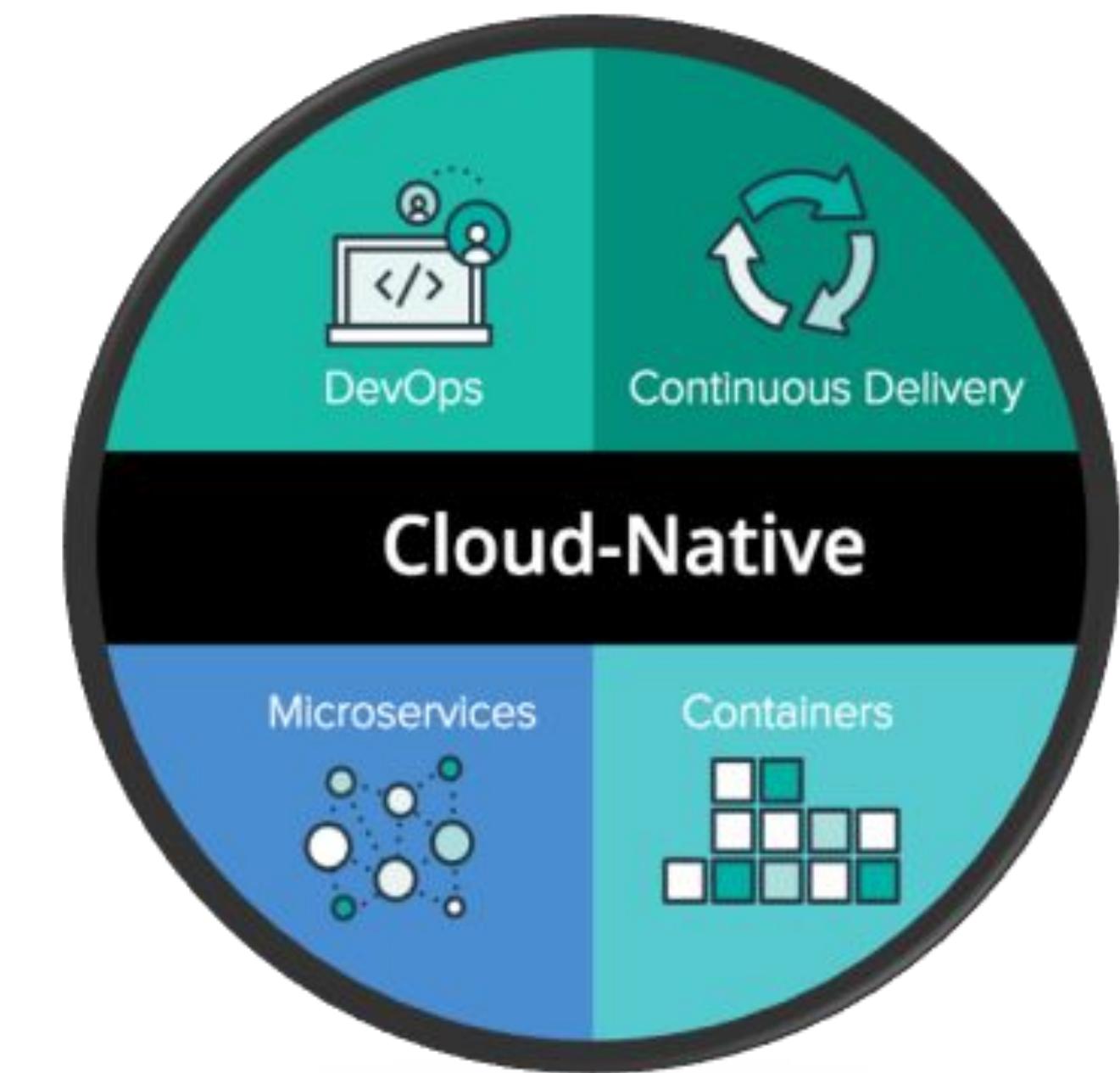
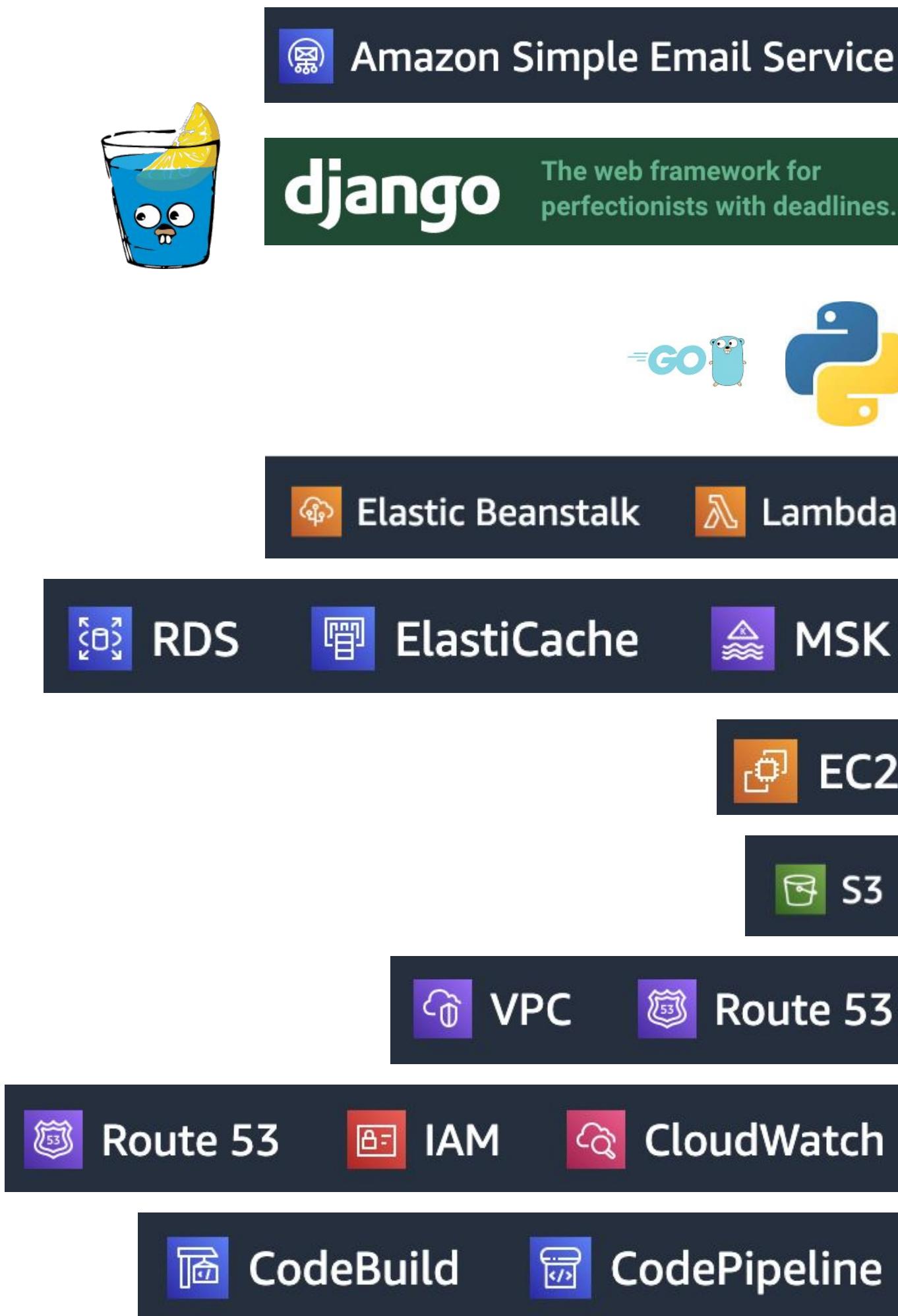


# Physical Architecture & Design 04

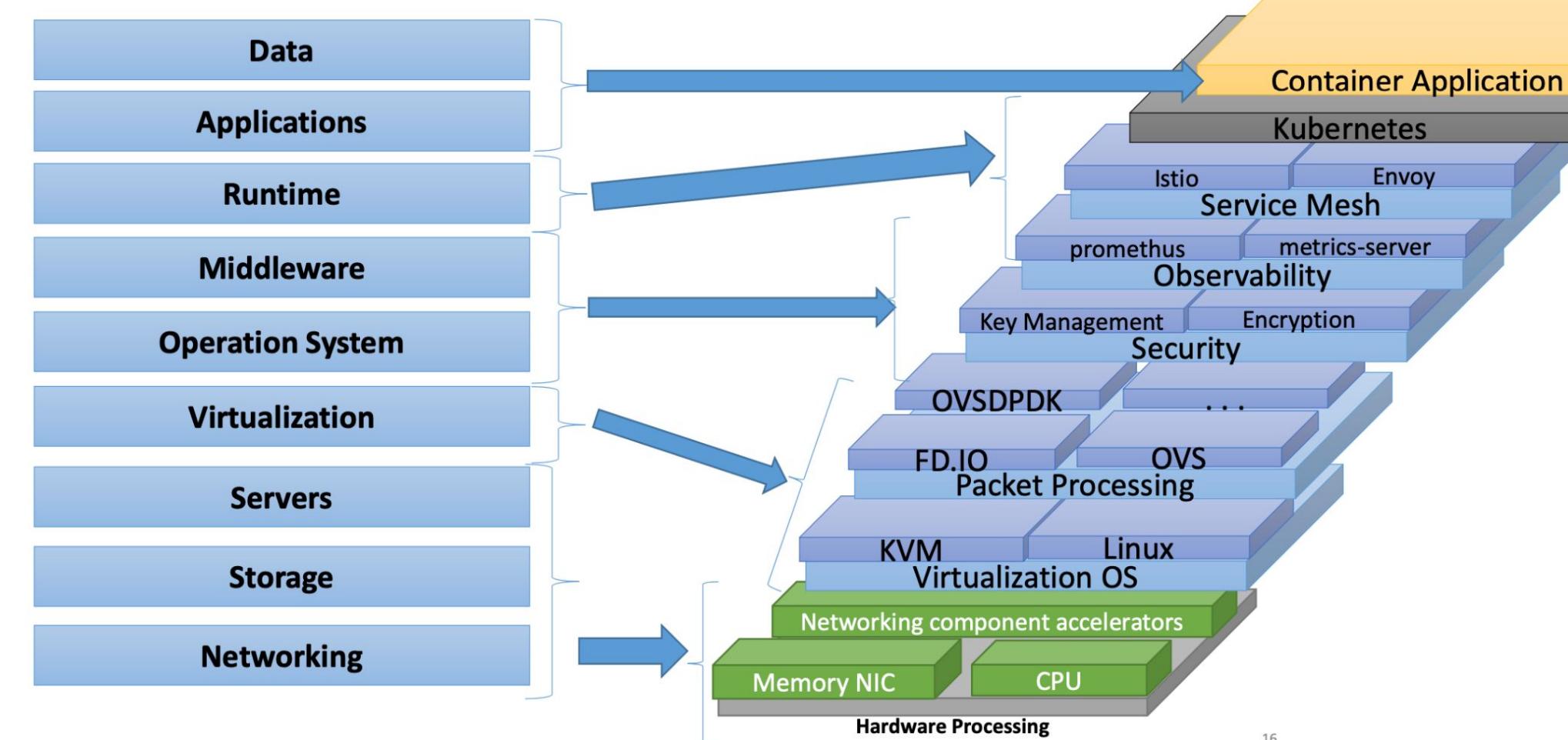
## – Tech Stacks

# Apply a Cloud Native Tech Stacks

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

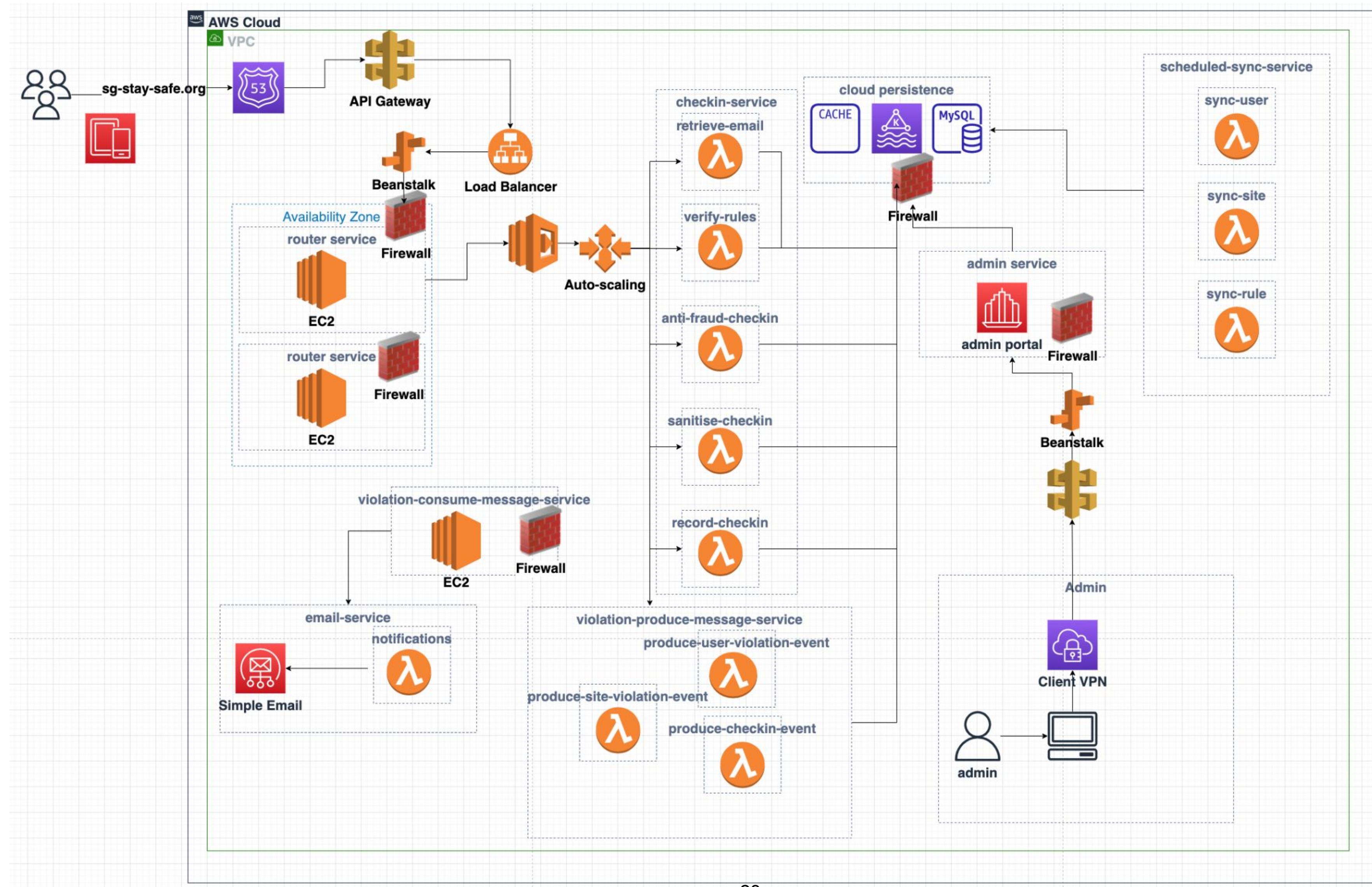


## Reference Architectures and Tech Stacks for Cloud Native



# Physical Architecture & Design 04

## – Physical Architecture Diagram



# Physical Architecture & Design 04

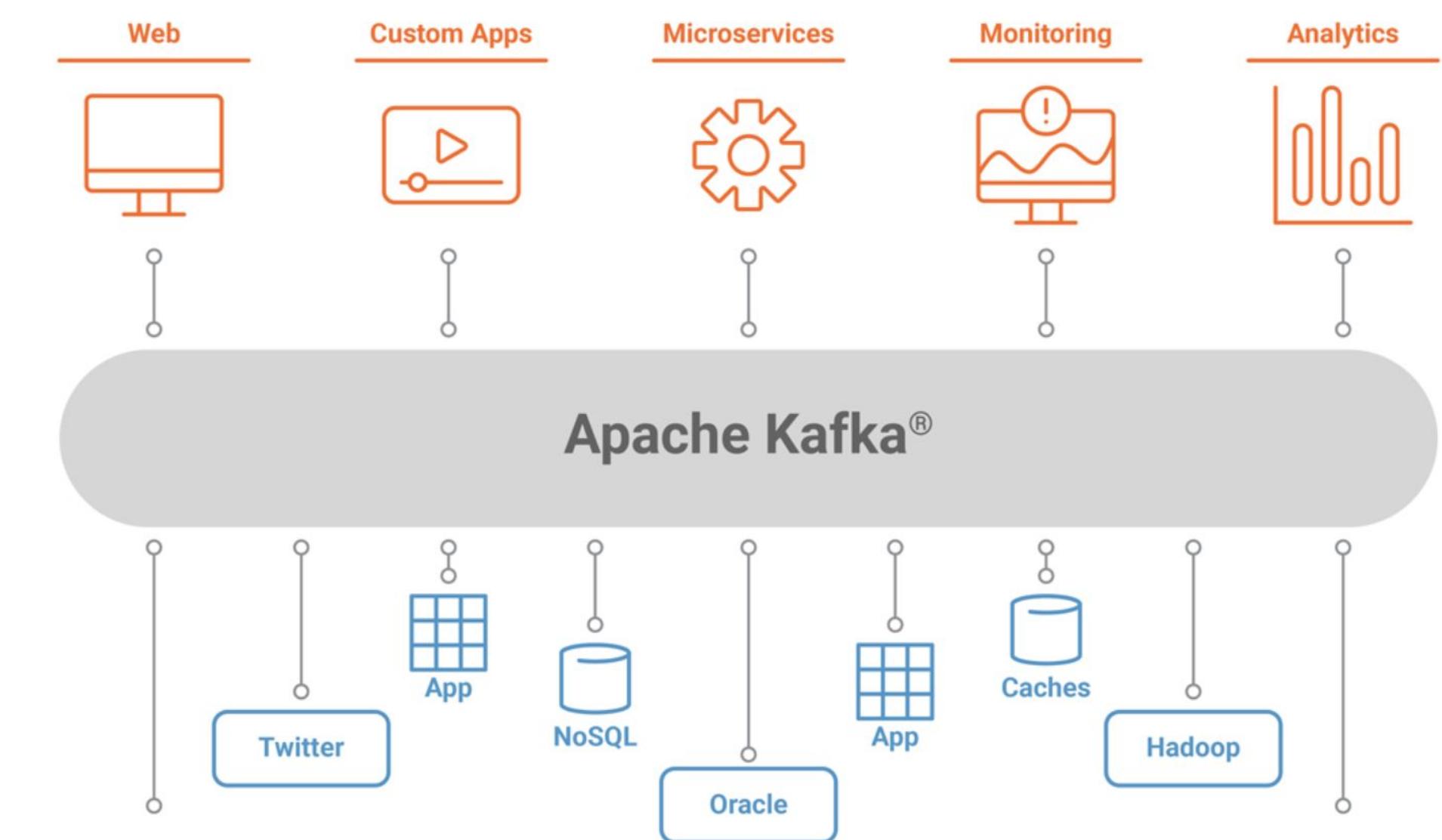
## – Persistence Design

## 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



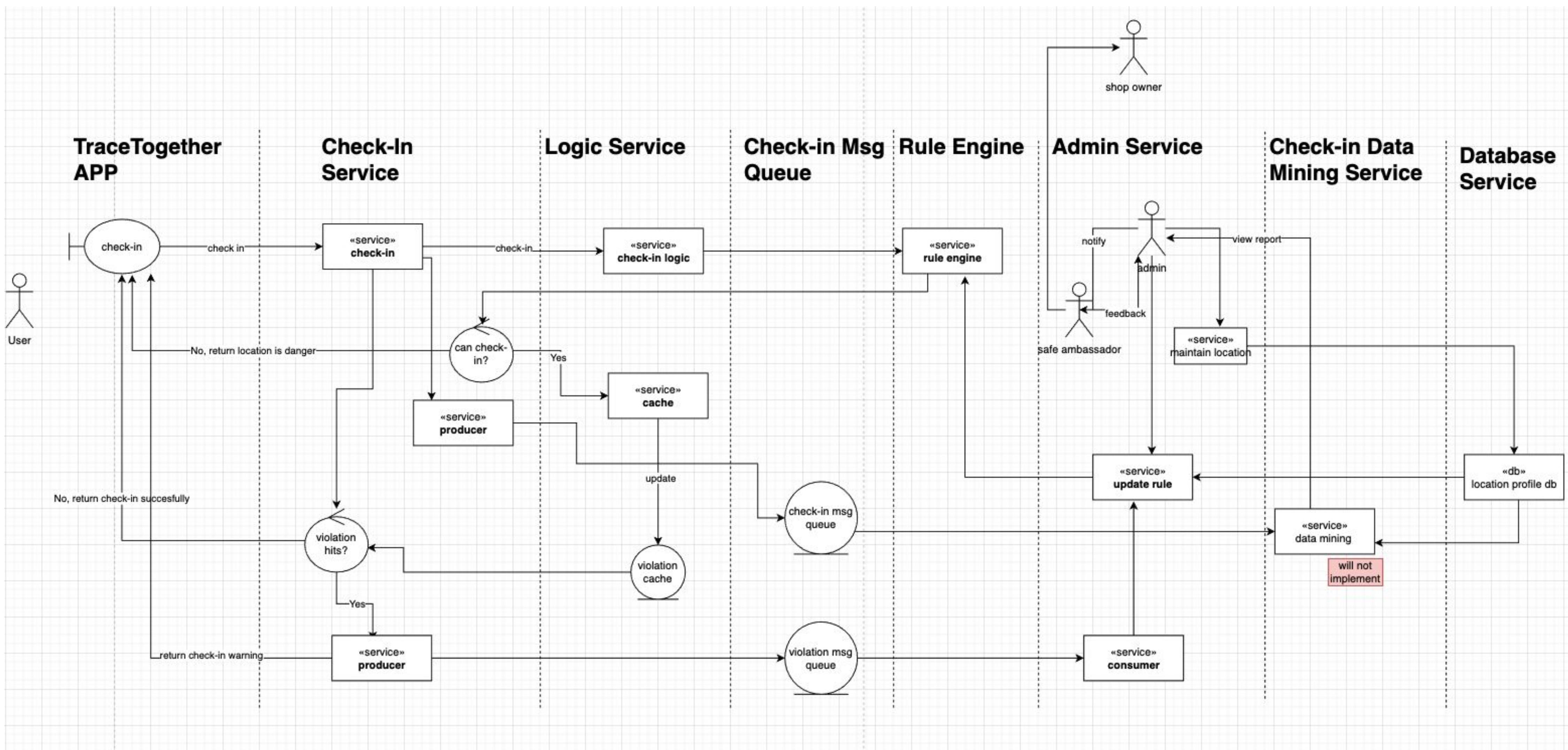
Decision outcome for [i]: Cache Storage



Decision outcome for [ii]: Kafka Msg Storage

# Physical Architecture & Design 04

## – User Banned Use Case



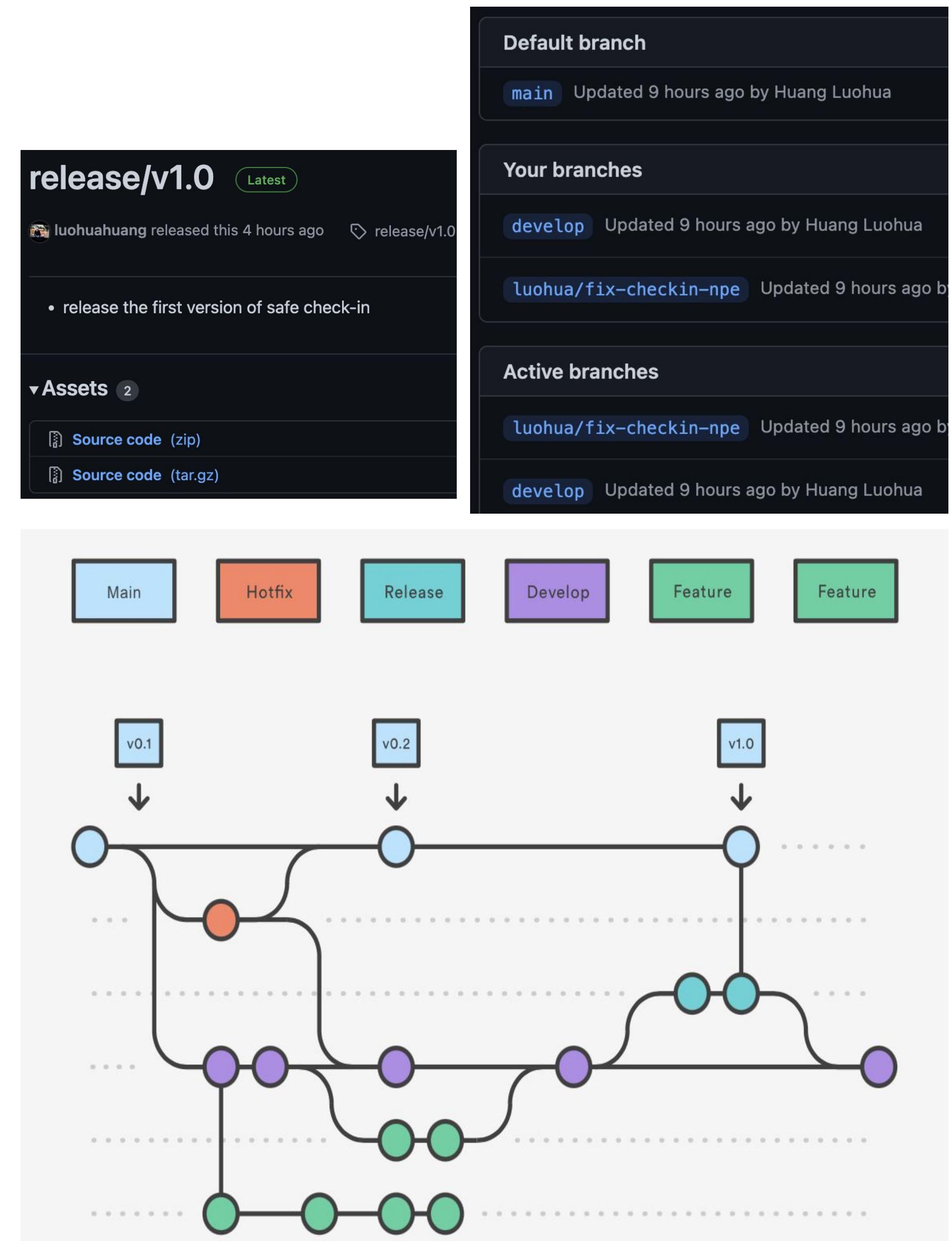
# DevOps and Development Lifecycle

# DevOps and Development Lifecycle 05

## – Source Code Management

# 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.



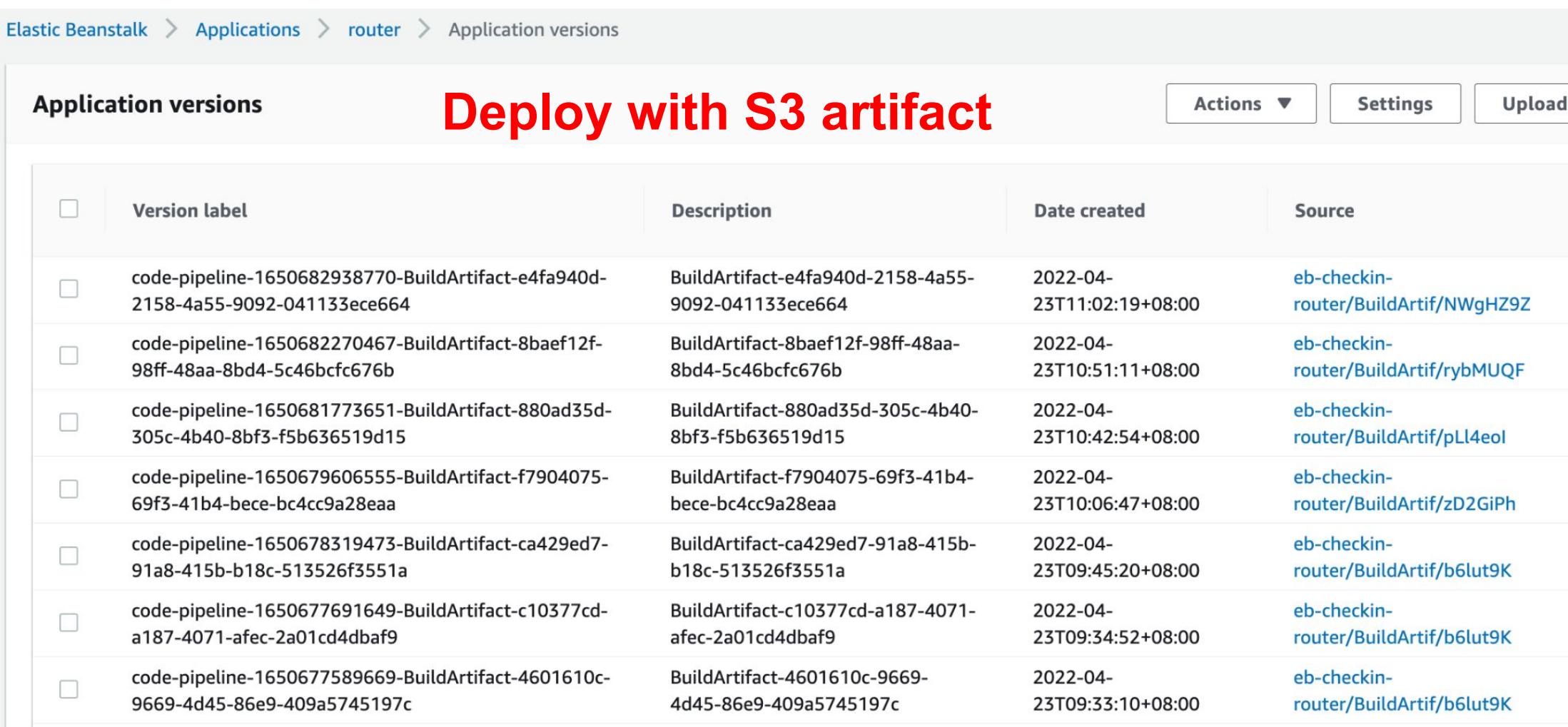
# DevOps and Development Lifecycle 05

## – Artifact Management

# Elastic artifact storage management

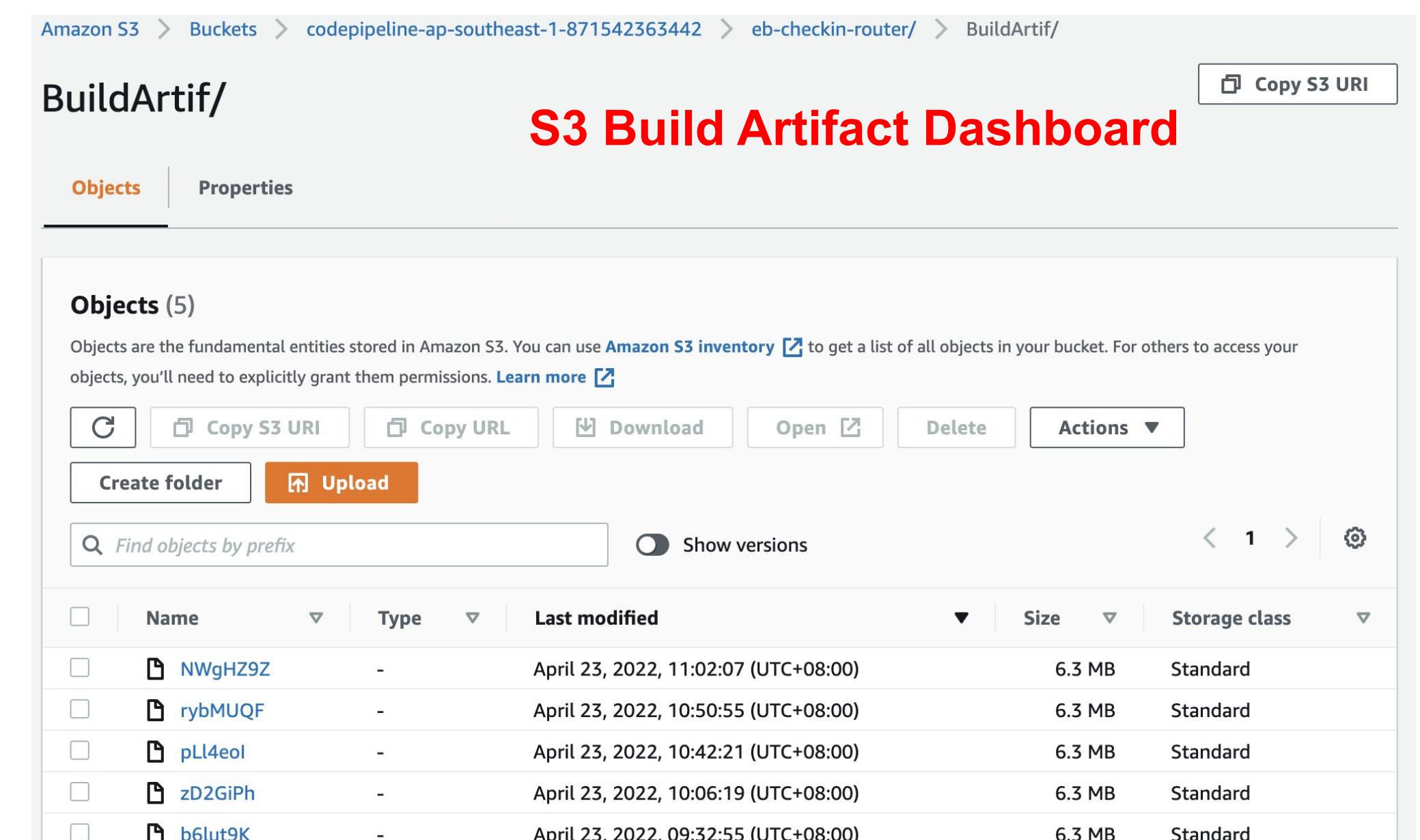
- Leverage S3 service for scalability, data availability, security, and performance
  - No more manual repository/storage management
  - Use Amazon S3 buckets to store and retrieve any amount of data
  - For each bucket, easier control access to it (who can create, delete, and list objects in the bucket)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



The screenshot shows the 'Application versions' section of the AWS Elastic Beanstalk console. It displays a table of deployed artifacts, each with a checkbox for 'Version label', a description, the date created, and the source. A red 'Deploy with S3 artifact' button is prominently displayed above the table.

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/pLl4eol
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



The screenshot shows the 'BuildArtif/' bucket in the AWS S3 console. It displays a list of objects, each with a checkbox, name, type, last modified, size, and storage class. A red 'S3 Build Artifact Dashboard' label is overlaid on the page.

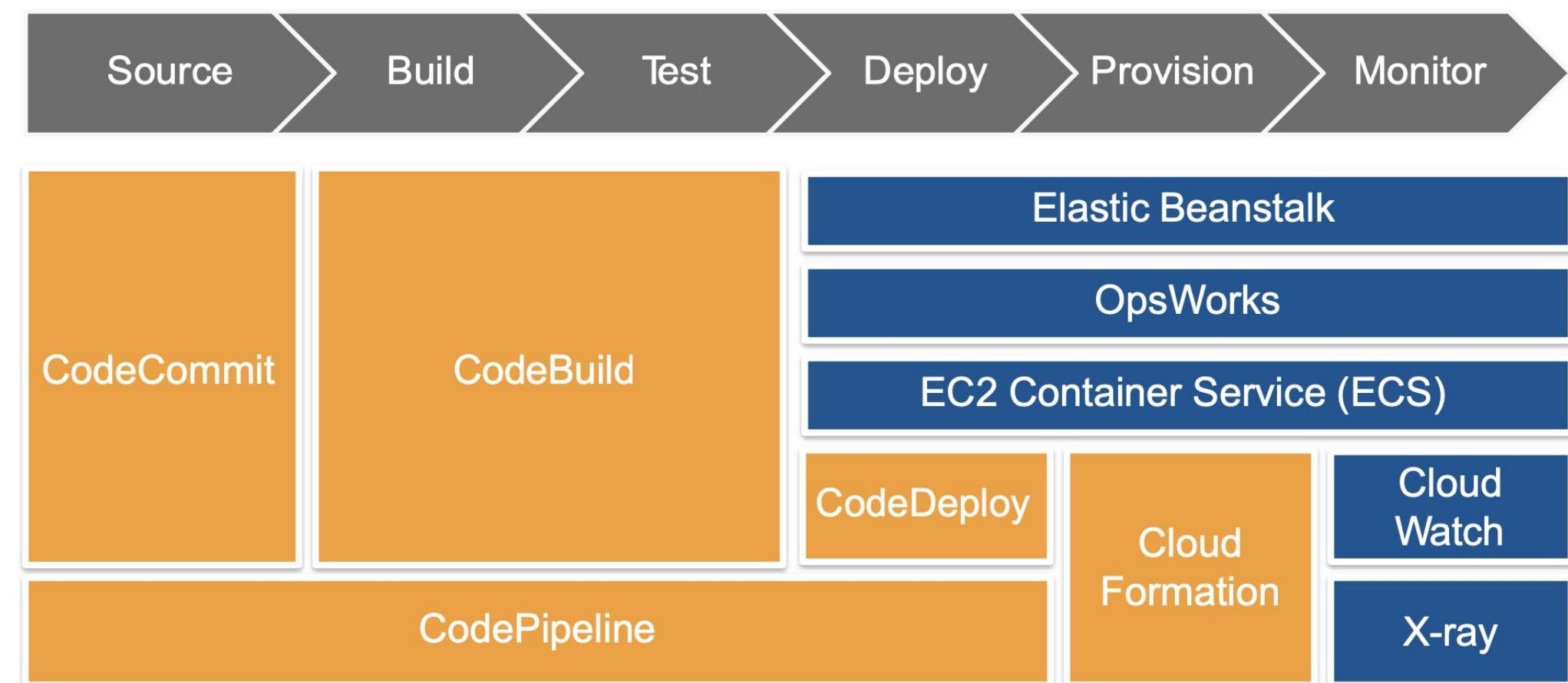
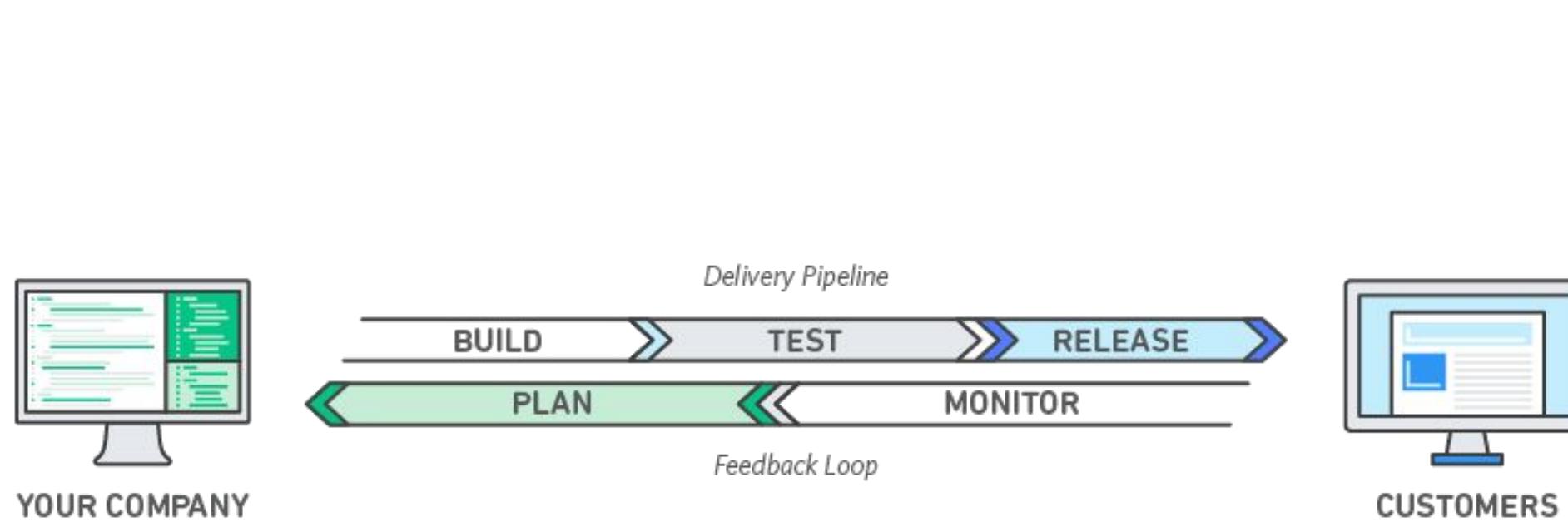
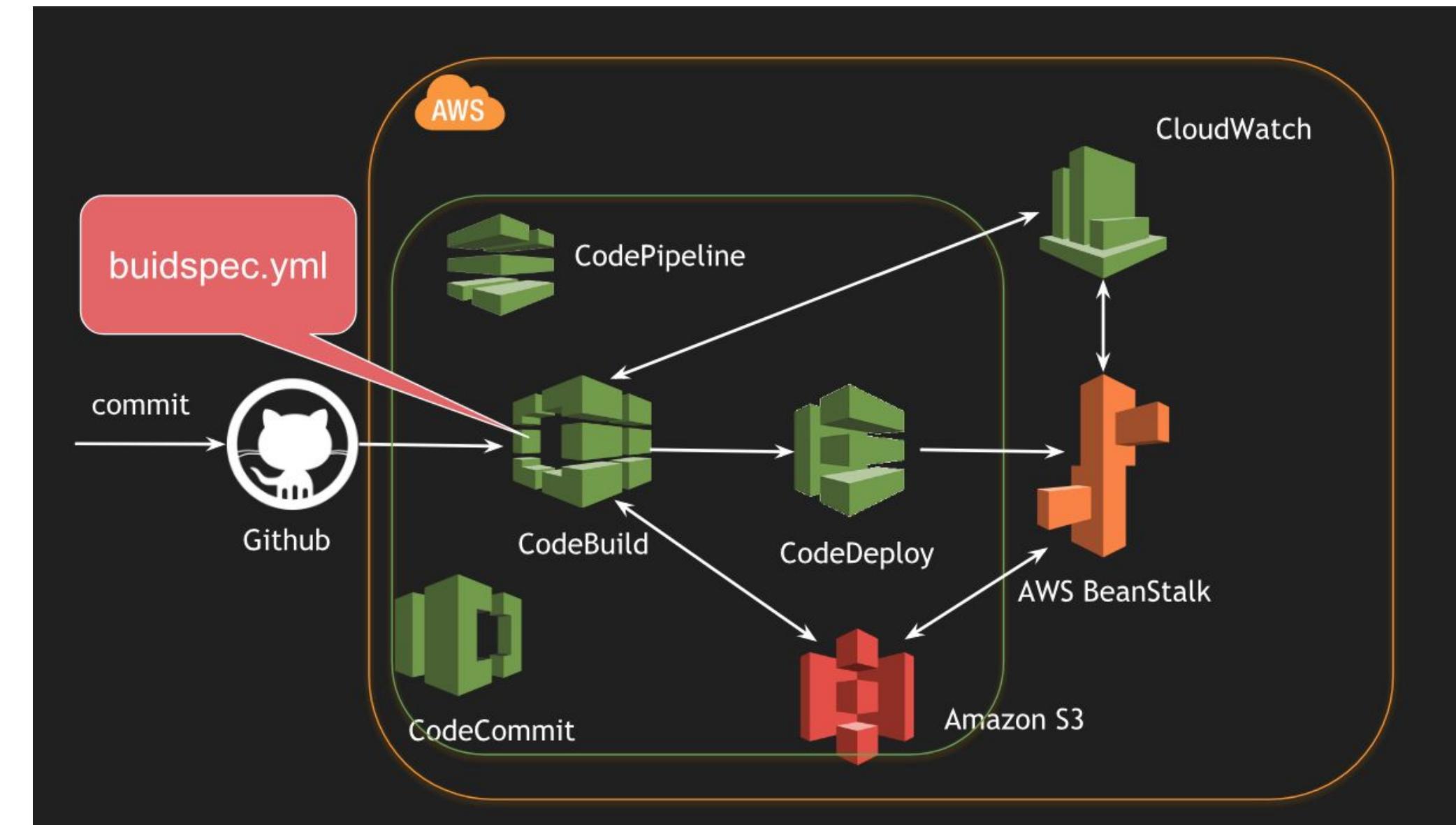
Name	Type	Last modified	Size	Storage class
NWgHZ9Z	-	April 23, 2022, 11:02:07 (UTC+08:00)	6.3 MB	Standard
rybMUQF	-	April 23, 2022, 10:50:55 (UTC+08:00)	6.3 MB	Standard
pLl4eol	-	April 23, 2022, 10:42:21 (UTC+08:00)	6.3 MB	Standard
zD2GiPh	-	April 23, 2022, 10:06:19 (UTC+08:00)	6.3 MB	Standard
b6lut9K	-	April 23, 2022, 09:32:55 (UTC+08:00)	6.3 MB	Standard

## DevOps and Development Lifecycle 05

### – DevOps Tools and Culture

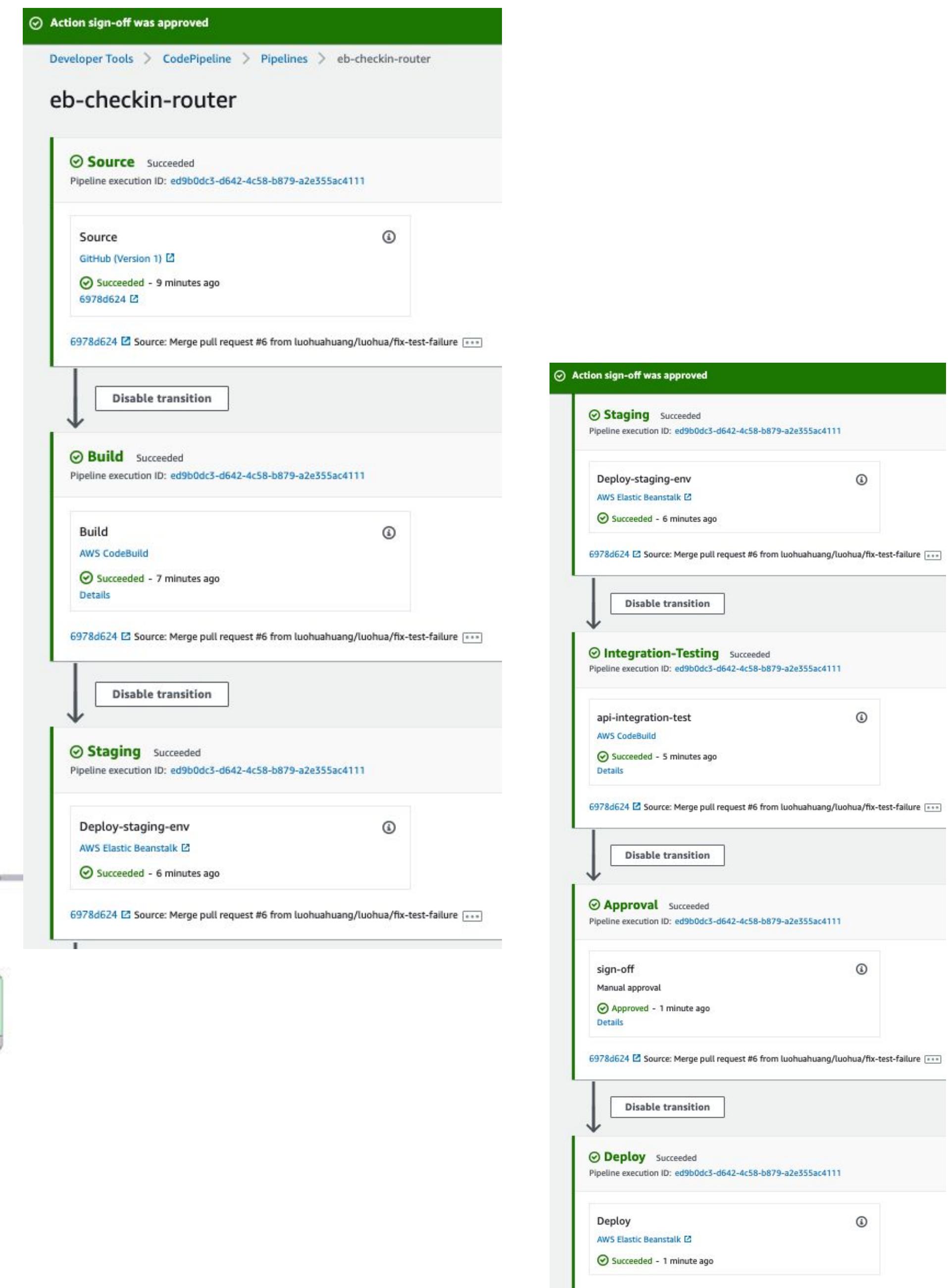
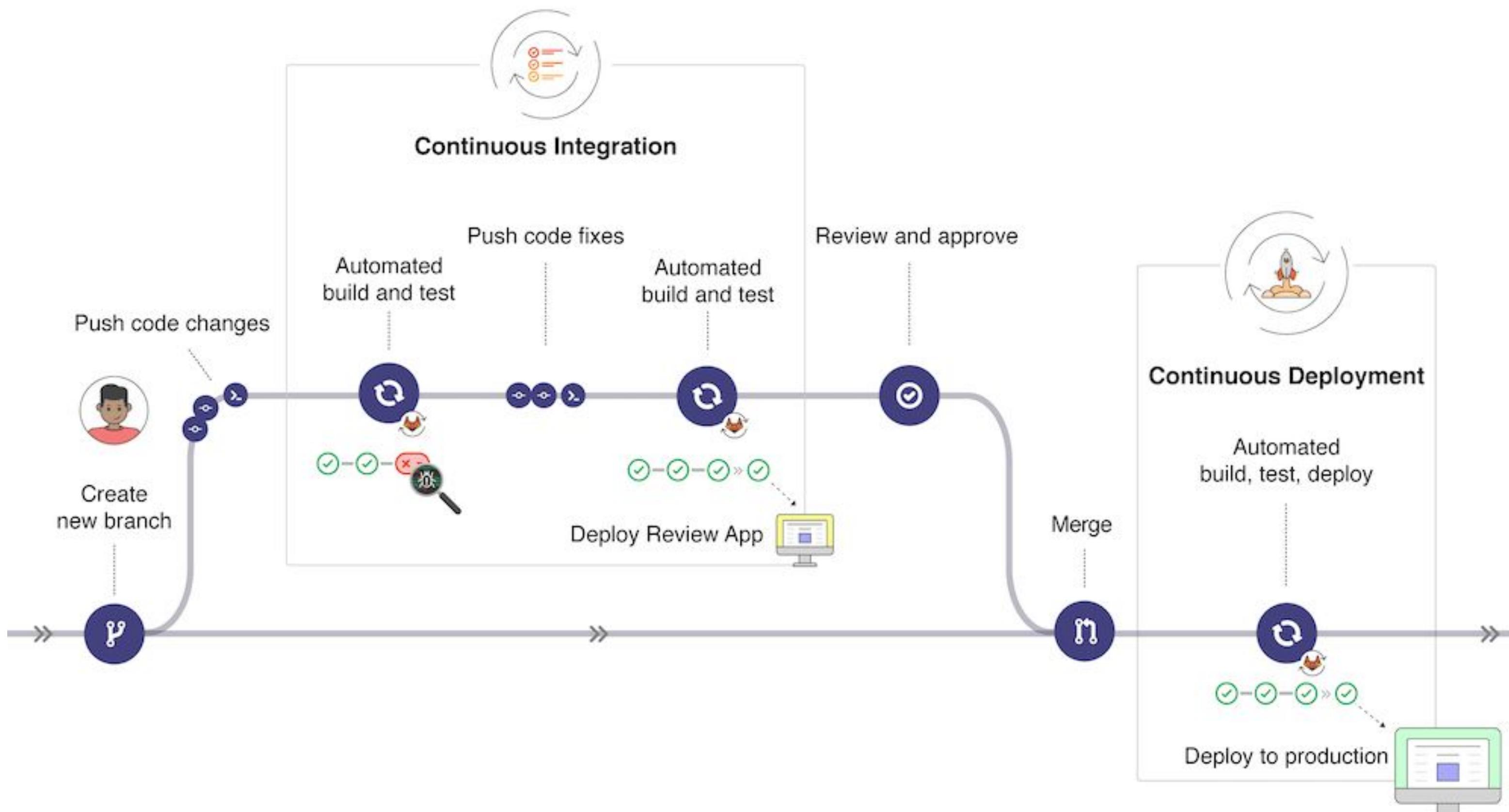
# Build an Automated, Agile and Observability DevOps

- 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

Build History					
	Build run	Status	Build number	Source version	Submitter
1	lambda-sanitiser-checkin:d0d5b452-9c78-41aa-b2a1-19fb7f67ce46	Succeeded	10	da6e11dce0100ed7d6083ac70c08c2d910e07f98	GitHub-Hookshot/85c02f5
2	lambda-sanitiser-checkin:970c8137-f228-4aae-a999-35ad5cf1f915	Succeeded	9	54dd527e139a9fda229084ff3c8cf7e8ac29b7a	GitHub-Hookshot/85c02f5
3	lambda-sanitiser-checkin:3c10dbf6-f6dd-440f-ad91-925bc9e15fd7	Succeeded	8	42801dc78c4c74b7e6b03d97cdbf0a8b06179dfe	GitHub-Hookshot/85c02f5
4	lambda-sanitiser-checkin:de471ad1-c71a-4665-baa5-20346e46255e	Succeeded	7	c7b9729a602fbf00e847b7742d752f6a697ab863	GitHub-Hookshot/85c02f5
5	lambda-sanitiser-checkin:77efc5f6-b4b3-48a7-8341-c704f4b30575	Succeeded	6	fab492477c561410c44c9376df8caf981b60e34e	GitHub-Hookshot/85c02f5
6	lambda-sanitiser-checkin:443e42d6-a4db-4f53-9a44-f308f06c4938	Succeeded	5	c8b7f2d81f02e424867bc582a871616ccbfac07a	GitHub-Hookshot/85c02f5
7	lambda-sanitiser-checkin:07755110-420d-4bf9-963a-84b999ae27a7	Succeeded	4	fd77469d650d3de2c7694a26276a63da8b814087	GitHub-Hookshot/85c02f5
8	lambda-sanitiser-checkin:9f2cdcd5-a98a-4220-971a-0d389627756a	Succeeded	3	35574c6106c9029752735c2a9bf68bfd89d21c47	GitHub-Hookshot/85c02f5
9	lambda-sanitiser-checkin:af002a0b-a454-46a0-b8de-fce65c7c9420	Succeeded	2	9a8a162ec2244d33b2f706f3ae1031349db2beec	GitHub-Hookshot/85c02f5
10	lambda-sanitiser-checkin:c9536785-94fc-4702-99cb-3cfa8d1f5f0f	Succeeded	1	b8a266693d6c7496bc52a0b3eaeb60f7f8927874	GitHub-Hookshot/85c02f5

```
producer/build.sh  produce.go  producer/buildspec.yml  retrieve-region-email-by-siteid/buildspec.yml  email/build.sh  email/buildspec.yml  retrieve-region-  
1  version: 0.2  
2  phases:  
3    install:  
4      runtime-versions:  
5        golang: 1.16  
6      commands:  
7        - echo "installing dependencies..."  
8        - go mod vendor  
9    build:  
10      commands:  
11        - echo "zipping deployment package..."  
12        - cd event/producer/  
13        - GOOS=linux GOARCH=amd64 go build -o produce-event-bin  
14        - zip -g deployment_package.zip produce-event-bin  
15    post_build:  
16      commands:  
17        - echo "updating lambda function..."  
18        - aws lambda update-function-code --function-name produce_checkin_event --zip-file fileb://deployment_package.zip  
19        - aws lambda update-function-code --function-name produce_userViolation_event --zip-file fileb://deployment_package.zip  
20        - aws lambda update-function-code --function-name produce_siteViolation_event --zip-file fileb://deployment_package.zip  
21        - echo "DONE"  
  
31
```

Build Projects					
Build projects					
Name	Source provider	Repository	Latest build status	Description	Last Modified
lambda-anti-fraud-checkin	GitHub	luohuahuang/sg-stay-safe.org	Succeeded	build & update anti-fraud checkin lambda function	6 minutes ago
lambda-record-checkin	GitHub	luohuahuang/sg-stay-safe.org	Succeeded	build & update record checkin lambda function	7 minutes ago
lambda-sanitiser-checkin	GitHub	luohuahuang/sg-stay-safe.org	Succeeded	build & update sanitiser checkin lambda function	7 minutes ago
lambda-sync-rule-from-db	GitHub	luohuahuang/sg-stay-safe.org	Succeeded	build & update sync rule from db lambda function	7 minutes ago
lambda-sync-site-from-db	GitHub	luohuahuang/sg-stay-safe.org	Succeeded	build & update sync site from db lambda function	8 minutes ago
lambda-user-site-from-db	GitHub	luohuahuang/sg-stay-safe.org	Succeeded	build & update sync user site from db lambda function	8 minutes ago
lambda-produce-events	GitHub	luohuahuang/sg-stay-safe.org	Succeeded	build & update three producer lambda functions: check-in, user-violation, and site-violation	9 minutes ago
lambda-notification-email	GitHub	luohuahuang/sg-stay-safe.org	Succeeded	build & update email notification lambda function	10 minutes ago
lambda-retrieve-region-email-by-siteid	GitHub	luohuahuang/sg-stay-safe.org	Succeeded	build & update retrieve email by siteid lambda function	10 minutes ago
lambda-verify-rules	GitHub	luohuahuang/sg-stay-safe.org	Succeeded	build & update verify-rule lambda function	10 minutes ago

## 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

Add more commits by pushing to the `luohua/api-test` branch on `luohuahuang/sg-stay-safe.org`.

Some checks haven't completed yet  
3 pending and 4 successful checks

## Continuous Build

AWS CodeBuild ap-southeast-1 (DEV-eb-checkin-router)	Pending — Build s...	Details
AWS CodeBuild ap-southeast-1 (lambda-sync-rule-from-db)	Pending — Bui...	Details
AWS CodeBuild ap-southeast-1 (lambda-sync-site-from-db)	Pending — Buil...	Details
AWS CodeBuild ap-southeast-1 (lambda-anti-fraud-checkin)	— Build succee...	Details
AWS CodeBuild ap-southeast-1 (lambda-notification-email)	— Build succeed...	Details
AWS CodeBuild ap-southeast-1 (lambda-verify-rules)	— Build succeeded for ...	Details

This branch has no conflicts with the base branch  
Merging can be performed automatically.

Merge pull request ▾ You can also open this in GitHub Desktop or view command line instructions.

```
72 [Container] 2022/05/02 08:44:58 Phase complete: INSTALL State: SUCCEEDED
73 [Container] 2022/05/02 08:44:58 Phase context status code: Message:
74 [Container] 2022/05/02 08:44:58 Entering phase PRE_BUILD
75 [Container] 2022/05/02 08:44:58 Phase complete: PRE_BUILD State: SUCCEEDED
76 [Container] 2022/05/02 08:44:58 Phase context status code: Message:
77 [Container] 2022/05/02 08:44:58 Entering phase BUILD
78 [Container] 2022/05/02 08:44:58 Running command echo "zipping deployment package.
79 zipping deployment package... Deploy
80
```

produce\_userViolationEvent

recordCheckin **Test env**

retrieveRegionEmailBySiteIdQuery

sanitiseCheckin

sendEmailNotification

syncRuleFromDbRuleEngine

syncSiteFromDbRuleEngine

syncUserFromDbRuleEngine

verifyRules

<input type="checkbox"/>	Function name
<input type="checkbox"/>	<b>Live env</b>
<input type="checkbox"/>	anti_fraud_checkin
<input type="checkbox"/>	live_anti_fraud_checkin
<input type="checkbox"/>	live_produce_checkin_event
<input type="checkbox"/>	live_produce_siteViolationEvent
<input type="checkbox"/>	live_produce_userViolationEvent
<input type="checkbox"/>	live_record_checkin
<input type="checkbox"/>	live_retrieveRegionEmailBySiteIdQueryService
<input type="checkbox"/>	live_sanitise_checkin
<input type="checkbox"/>	live_sendEmailNotification
<input type="checkbox"/>	live_syncRuleFromDbRuleEngine
<input type="checkbox"/>	live_syncSiteFromDbRuleEngine
<input type="checkbox"/>	live_syncUserFromDbRuleEngine
<input type="checkbox"/>	live_verifyRules

# 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
<input type="checkbox"/>	• (root)	100% (100%)	0	595.450
<input type="checkbox"/>	• check-in	100% (100%)	0	595.450
<input type="checkbox"/>	Test_checkin_anti_fraud_fresh_user_different_site_first_time_login	100% (100%)	0	0.290
<input type="checkbox"/>	Test_checkin_anti_fraud_fresh_user_different_site_revisit	100% (100%)	0	0.320
<input type="checkbox"/>	Test_checkin_anti_fraud_fresh_user_random_site_second_time_login	100% (100%)	0	0.170
<input type="checkbox"/>	Test_checkin_normal_fresh_user_random_site_first_time_login	100% (100%)	0	1.930
<input type="checkbox"/>	Test_checkin_normal_fresh_user_random_site_second_time_login_after_enough_time	100% (100%)	0	301.770
<input type="checkbox"/>	Test_checkin_normal_fresh_user_random_site_second_time_login_after_not_enough_time	100% (100%)	0	290.490
<input type="checkbox"/>	Test_checkin_sanitiser_anonymous_id_is_empty_only	100% (100%)	0	0.420
<input type="checkbox"/>	Test_checkin_sanitiser_corrupted_request	100% (100%)	0	0.000
<input type="checkbox"/>	Test_checkin_sanitiser_site_id_and_anonymous_id_both_empty	100% (100%)	0	0.030
<input type="checkbox"/>	Test_checkin_sanitiser_site_id_is_empty_only	100% (100%)	0	0.030

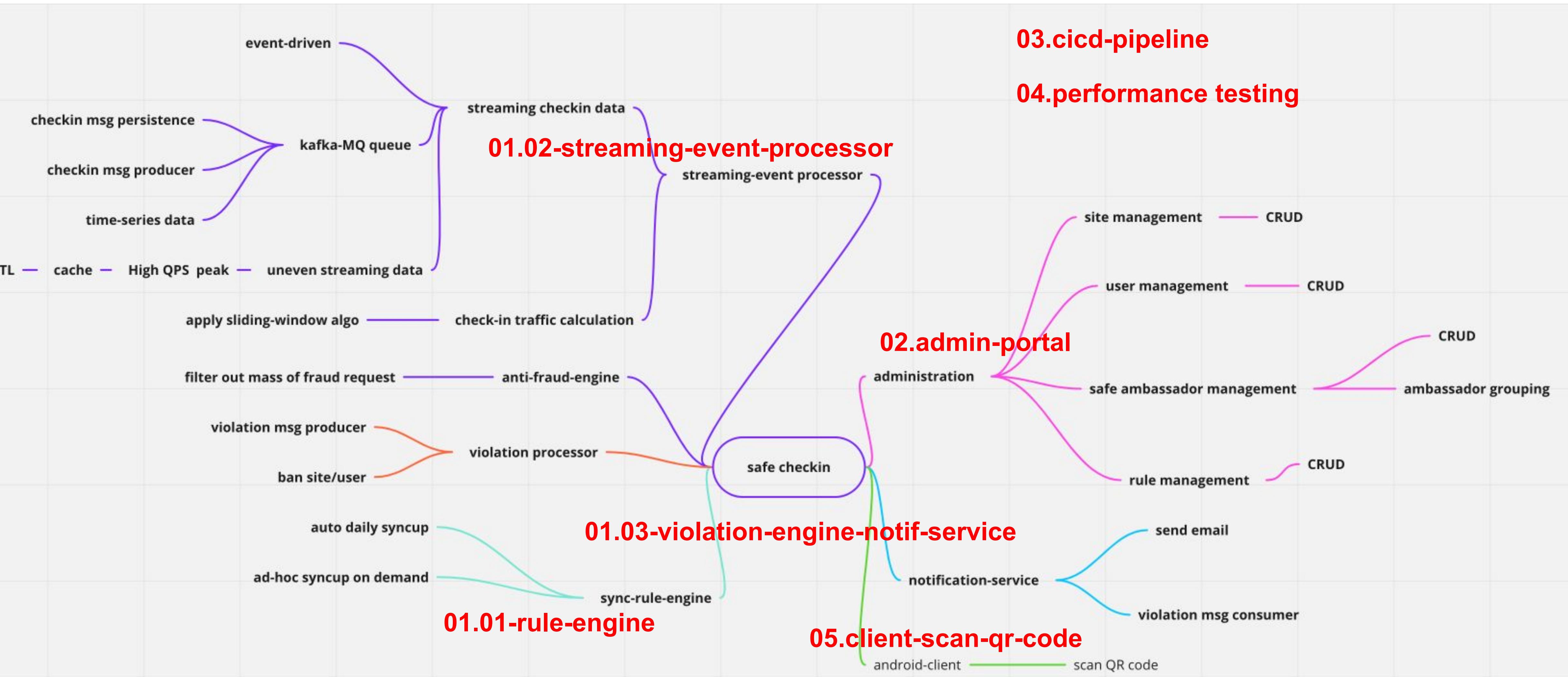
**APDEX (Application Performance Index)**

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		
<b>Total</b>	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

## 1. Overview

- a) Business Problem/Limitation of current Trace-Together app
- b) Platform Thinking Components
- c) Project Scope

## 2. Project Conduct

- a) Project Status and Challenges
- b) Team Contribution

## 3. Logical Architecture & Design

- a) Key Architectural Decisions
- b) Detailed deployment diagram
- c) Domain Driven Design Highlights

## 4. Physical Architecture & Design

- a) Key Architectural Decisions
- b) Physical Architecture Diagram
- c) Persistence design
- d) Detailed design of use case

## 5. DevOps & Deployment Lifecycle

- a) Source code & Artifacts
- b) Devops pipeline & procedures
- c) Technical findings and issues

## 6. DEMO!