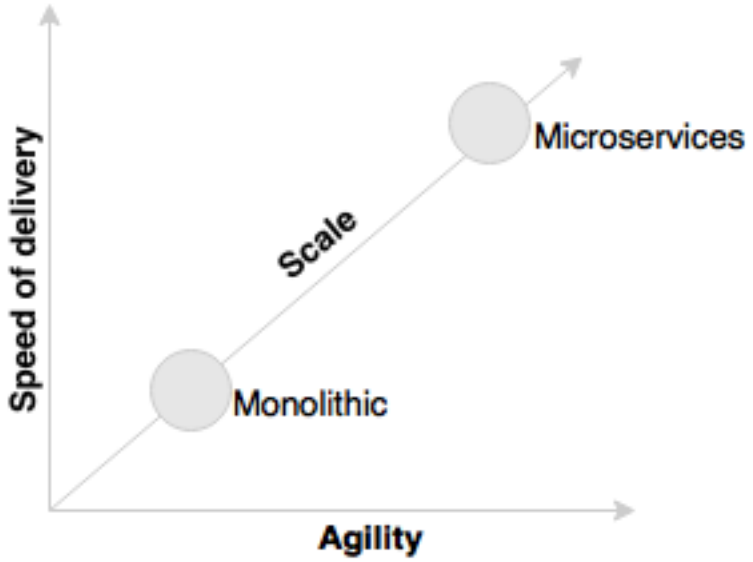
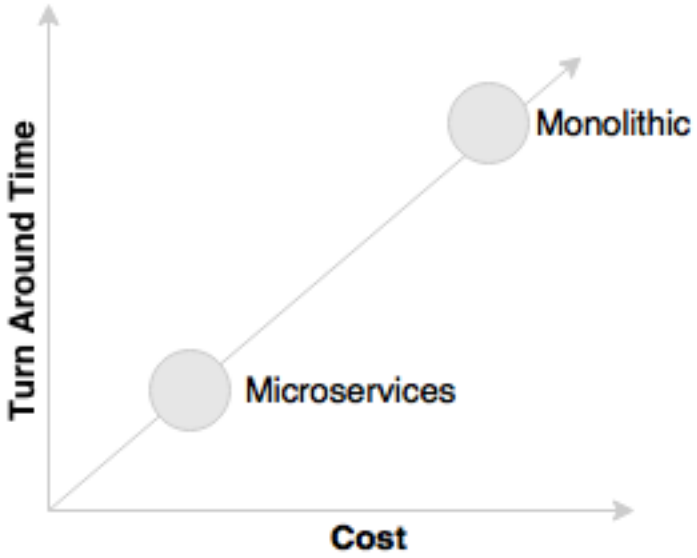
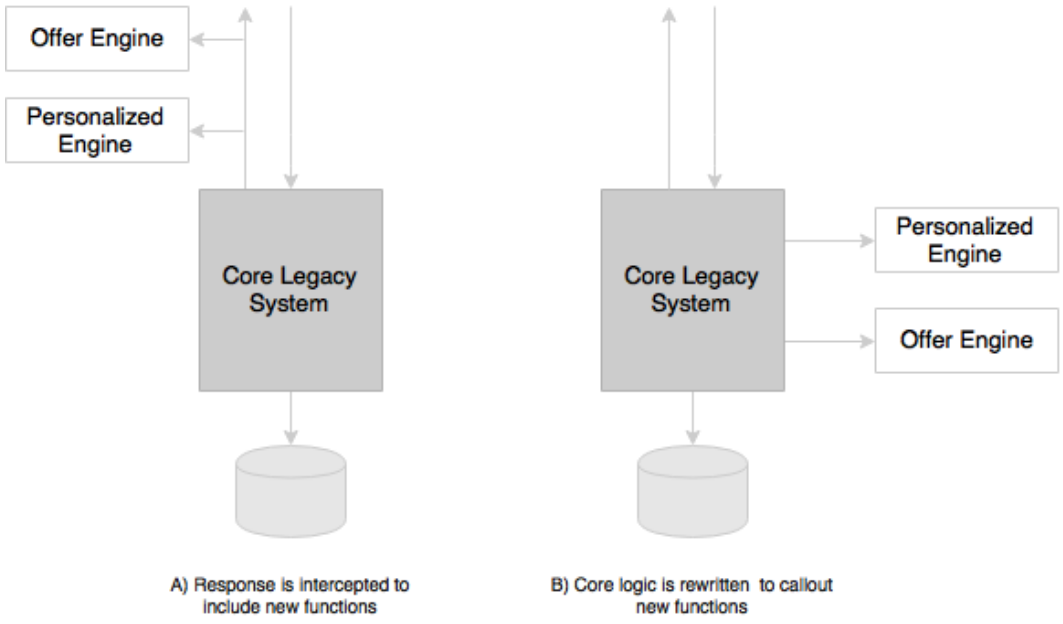
Chapter 1: Demystifying Microservices

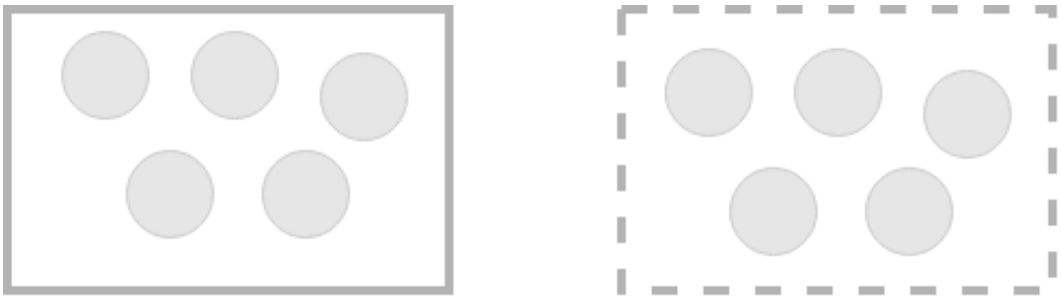






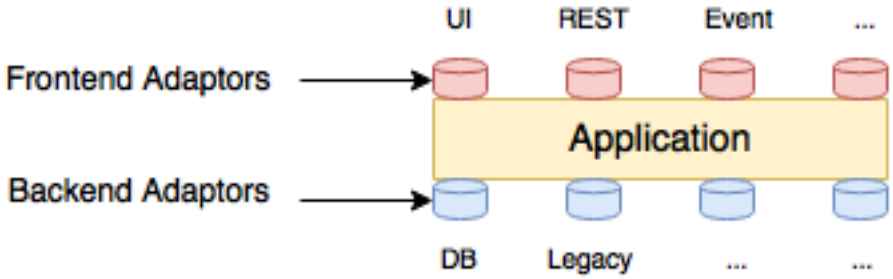
A) Response is intercepted to include new functions

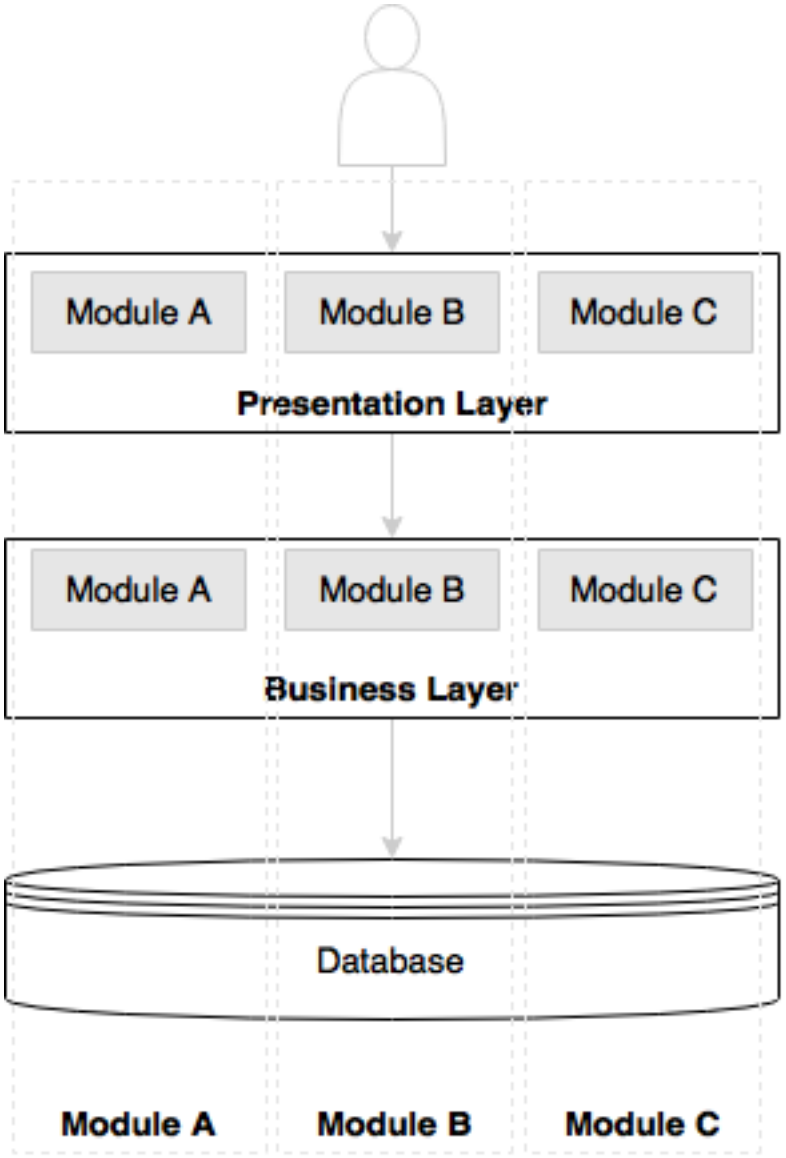
B) Core logic is rewritten to callout new functions

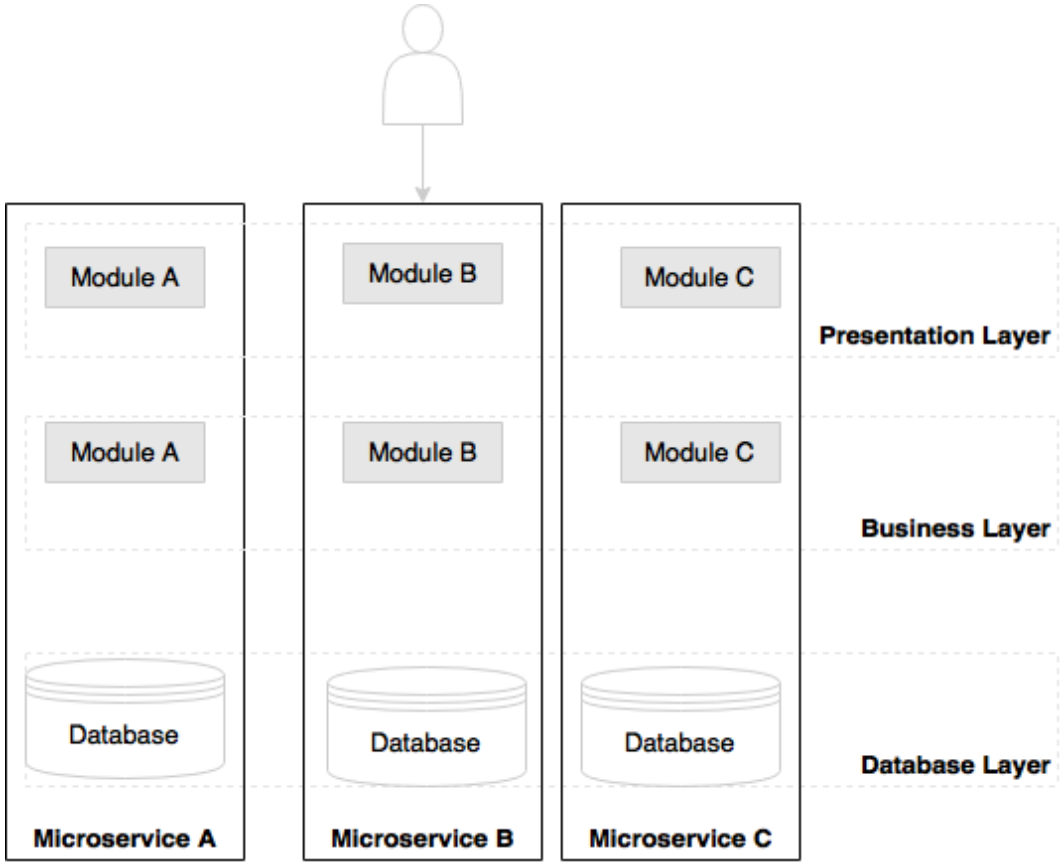


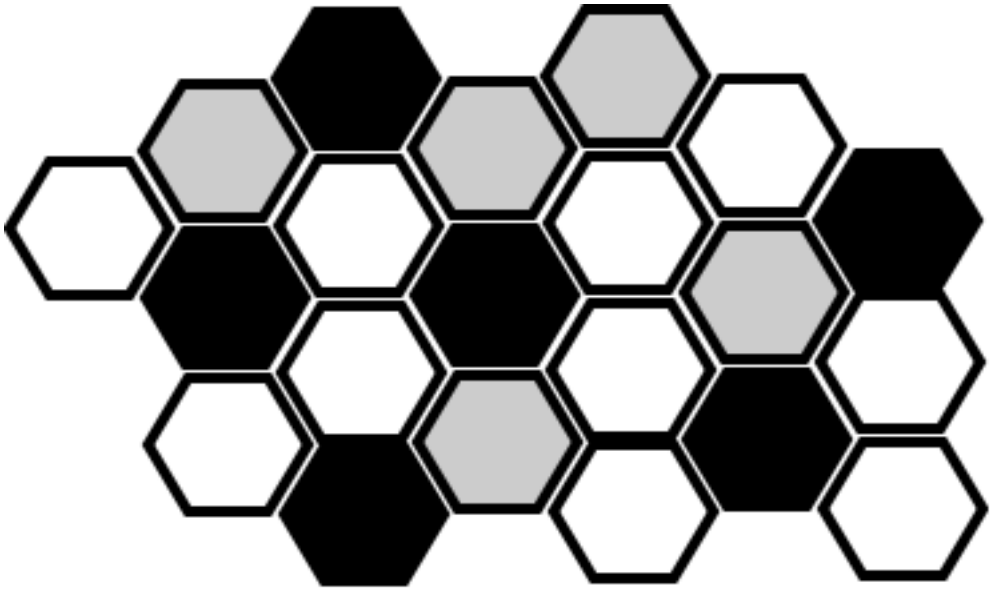
Monolithic Architecture

Microservices Architecture




Module A
Module B
Module C
Presentation Layer
Module A
Module B
Module C
Business Layer
Database
Module A
Module B
Module C


Module A
Module B
Module C
Presentation Layer
Module A
Module B
Module C
Business Layer
Database
Database
Database
Database Layer
Microservice A
Microservice B
Microservice C



Customer

Product

Order

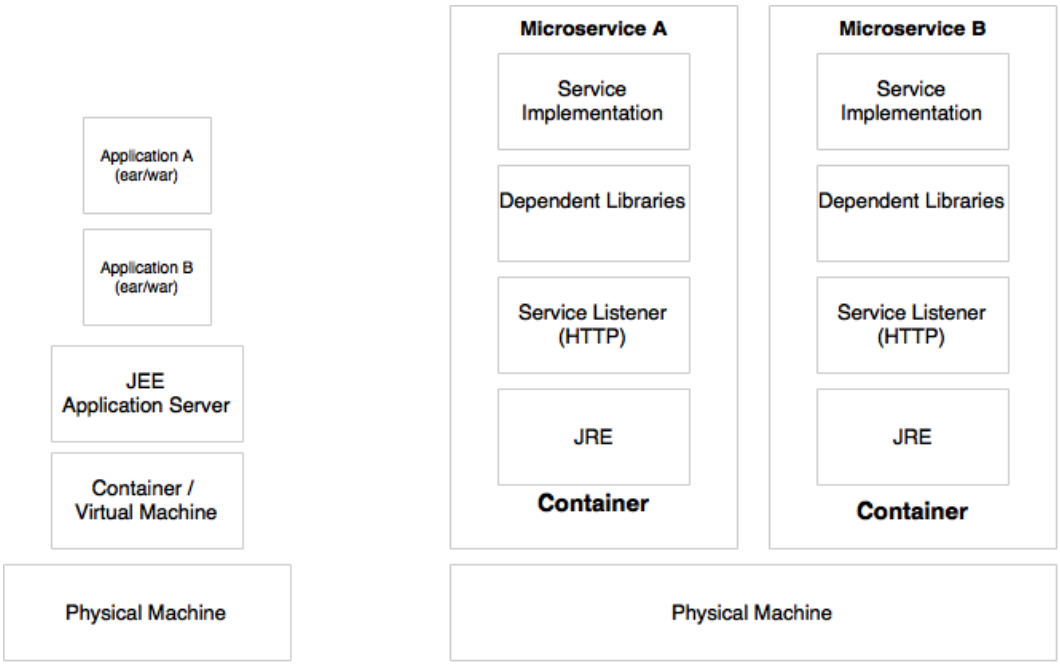
Multiple Responsibility Monolithic App

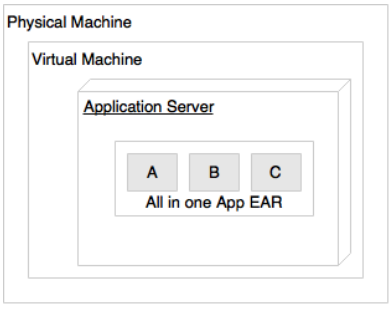
Customer

Product

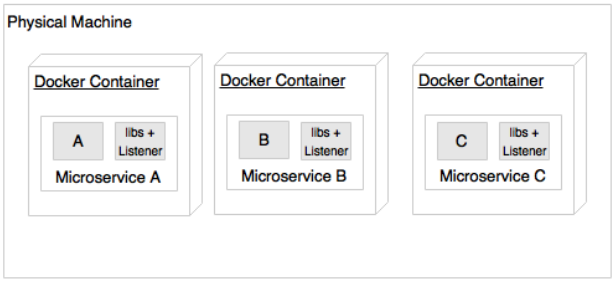
Order

Single Responsibility Microservices

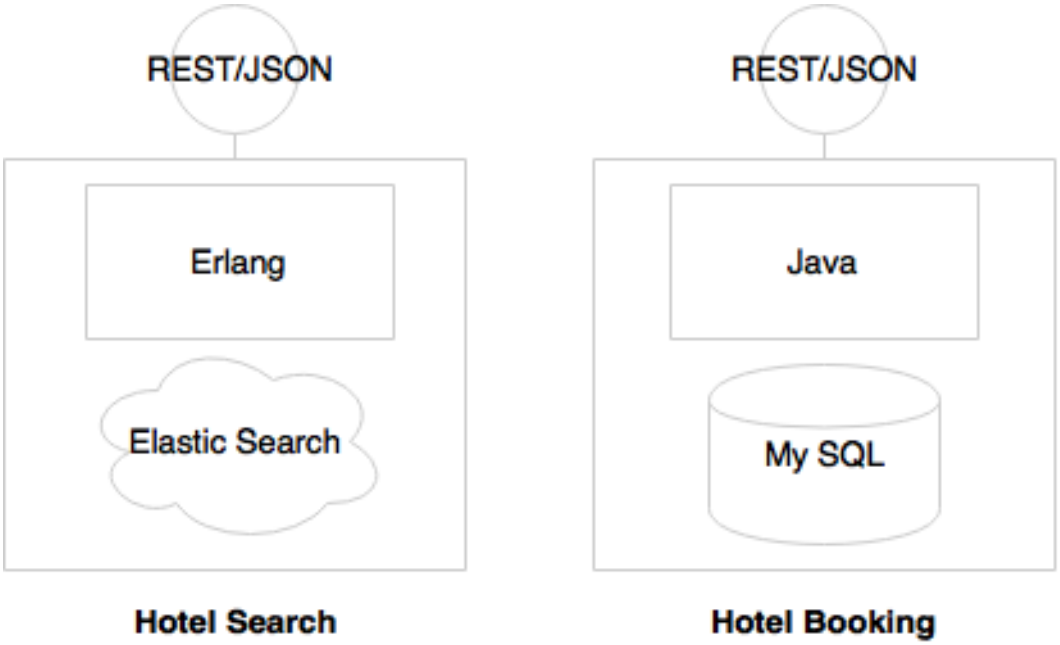




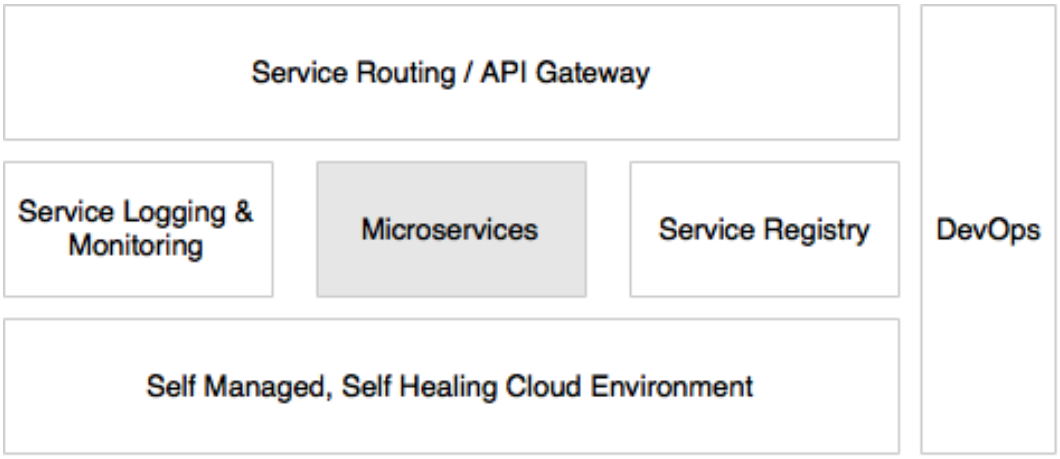
Tradiitonal Deployment

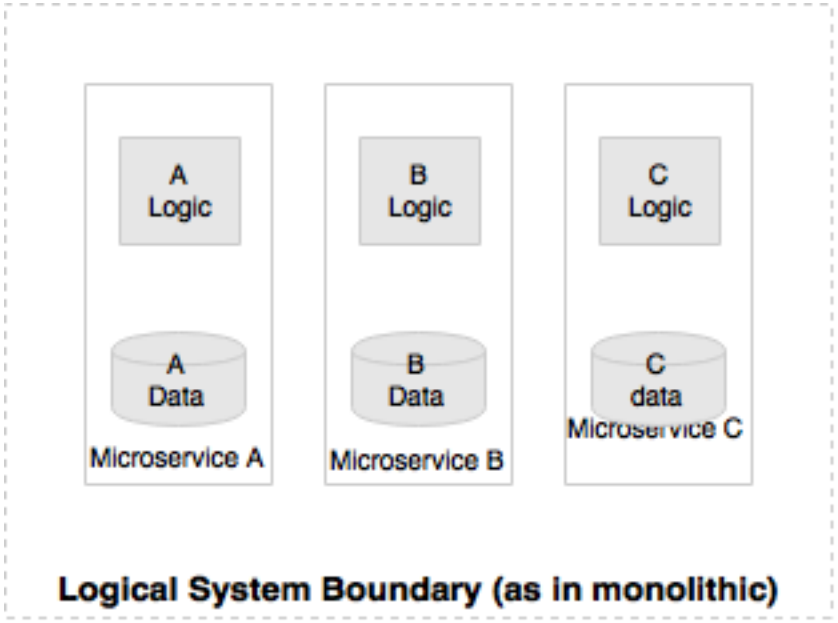


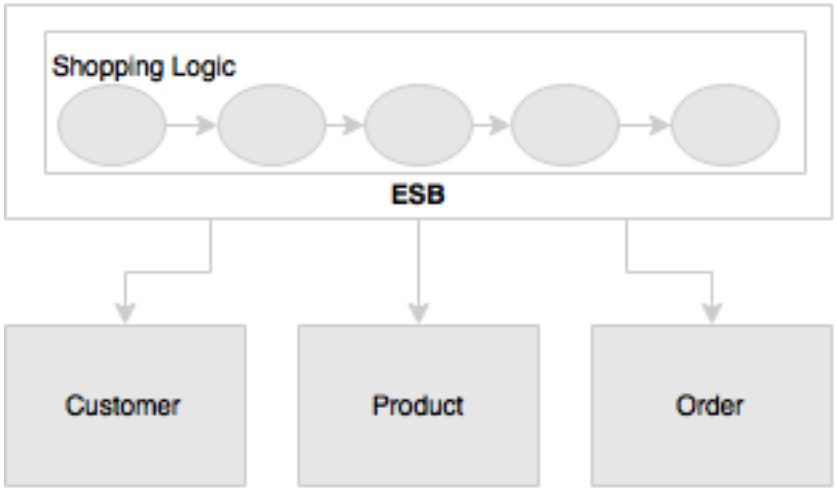
Microservices Deployment

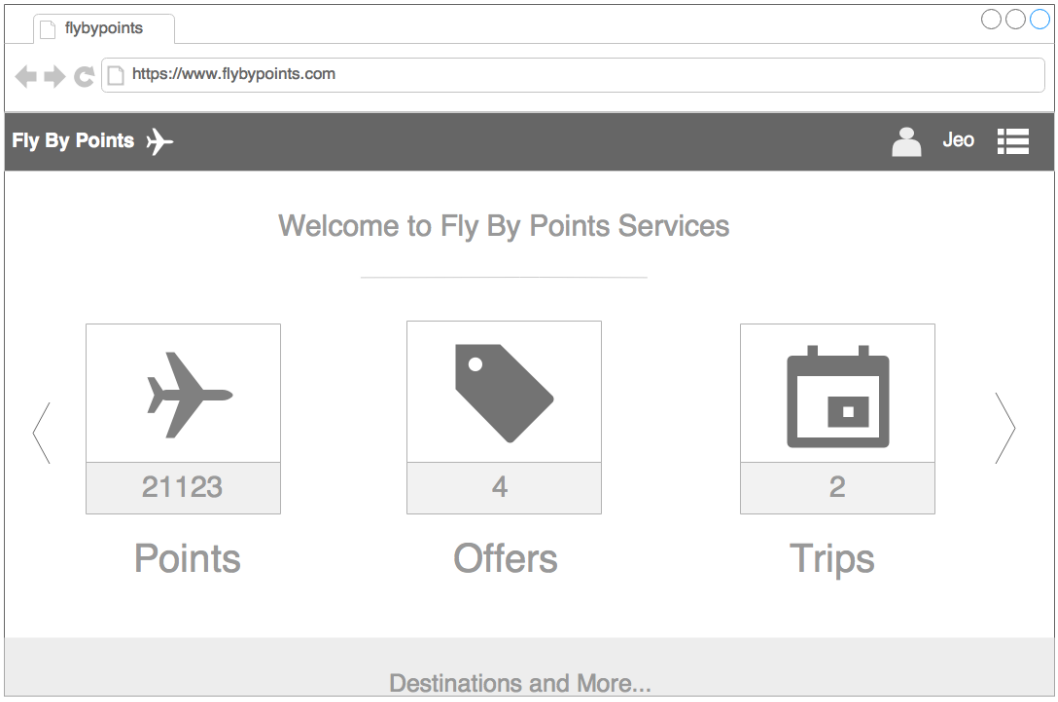

REST/JSON
Erlang
Elastic Search
Hotel Search
REST/JSON
Java
My SQL
Hotel Booking

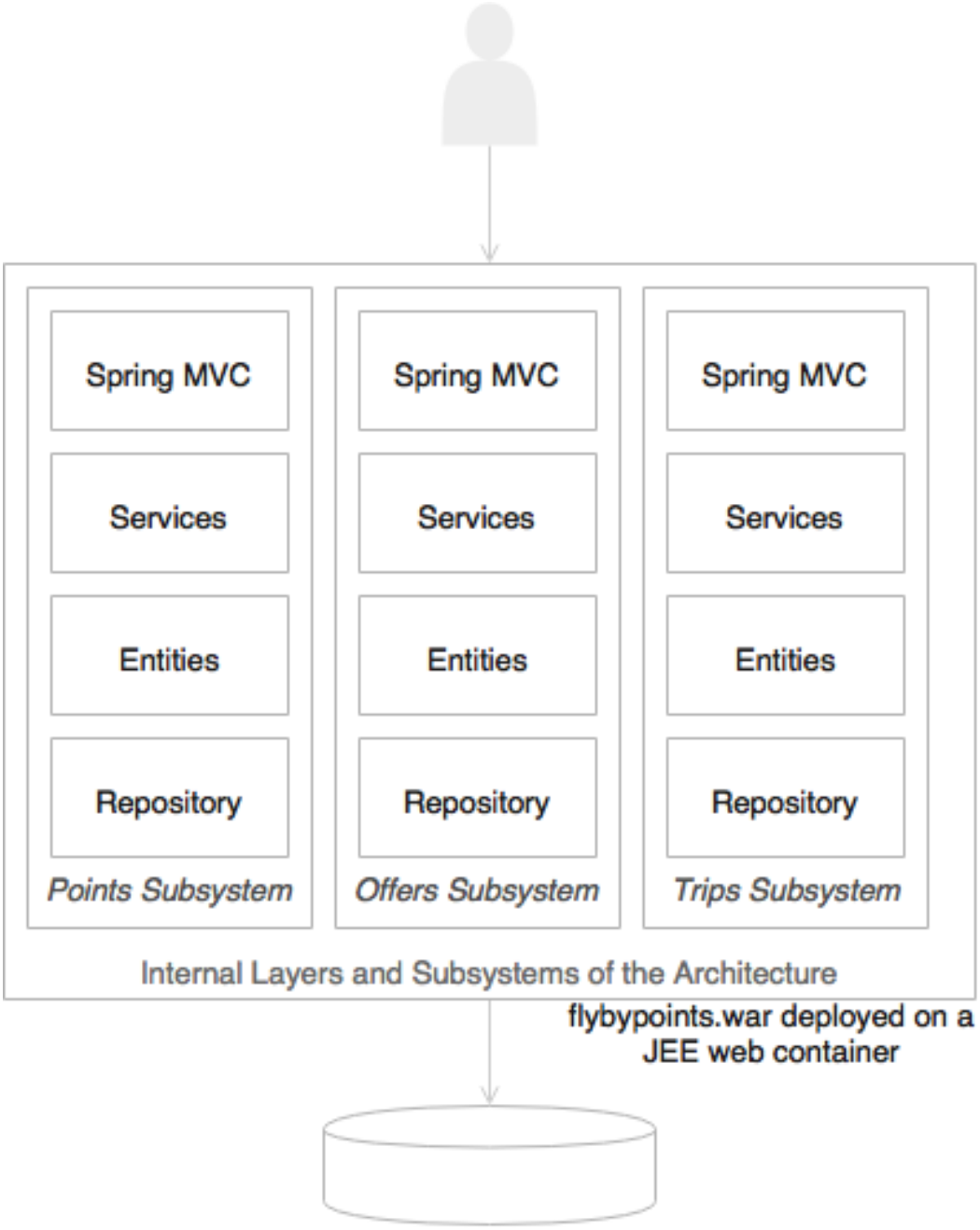
Automated Continous Integration
Automated Testing
Automated Infrastructure Provisioning
Automated Deployment

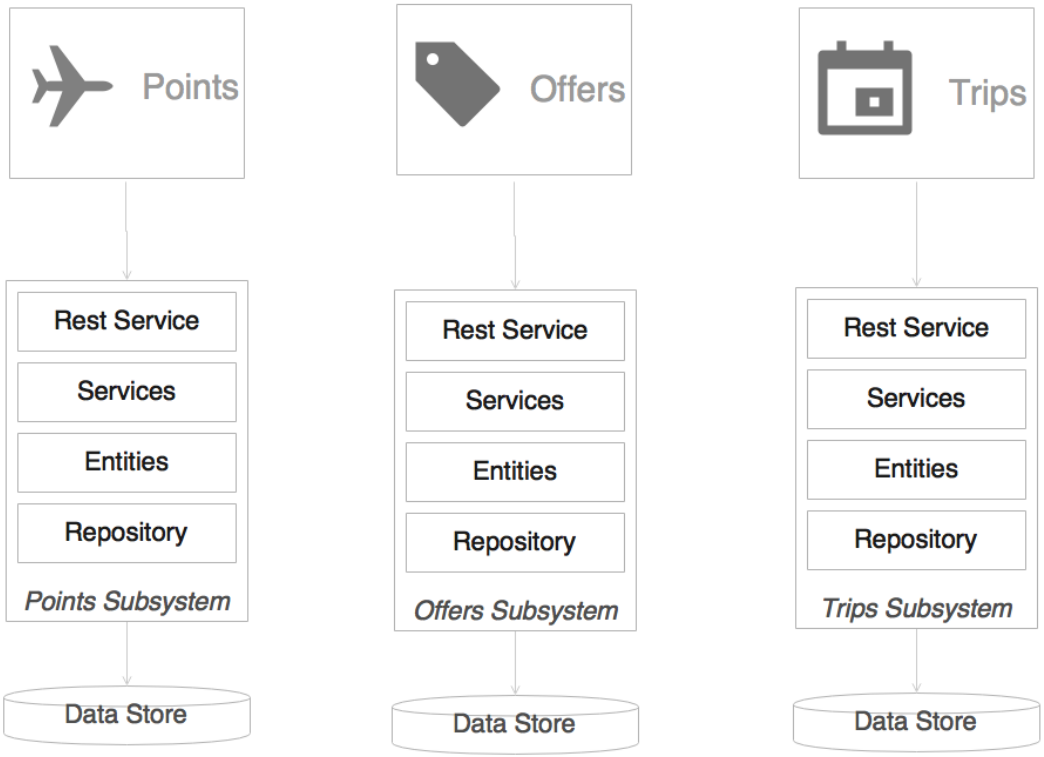

Service Routing / API Gateway
Service Logging & Monitoring
Microservices
Service Registry
DevOps
Self Managed, Self Healing Cloud Environment

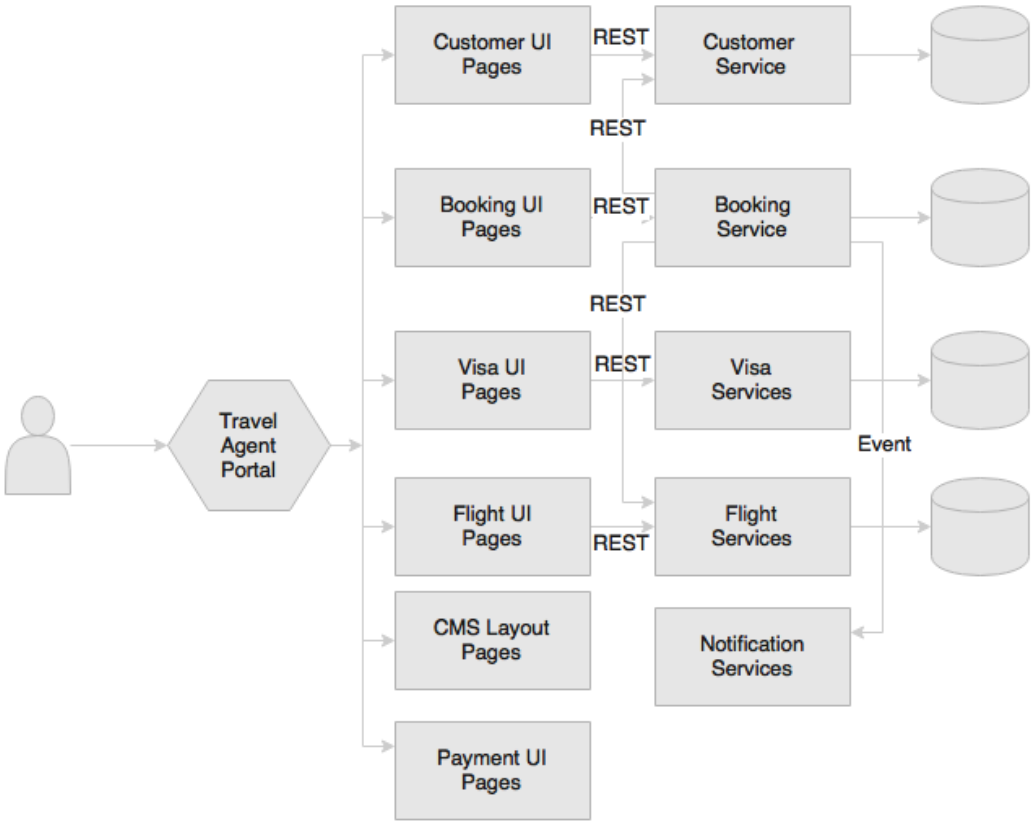

A Logic
A Data
Microservice A
B Logic
B Data
Microservice B
C Logic
C data
Microservice C
Logical System Boundary (as in monolithic)

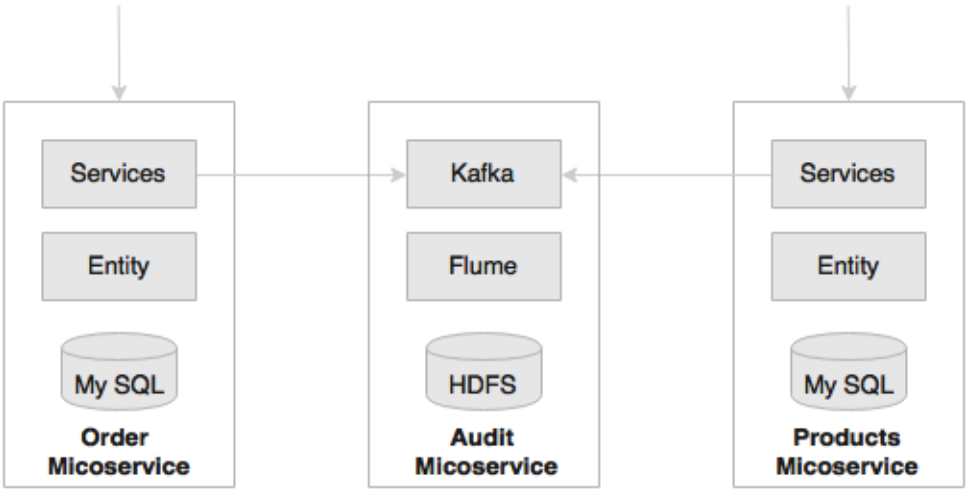

Shopping Logic
ESB
Customer
Product
Order

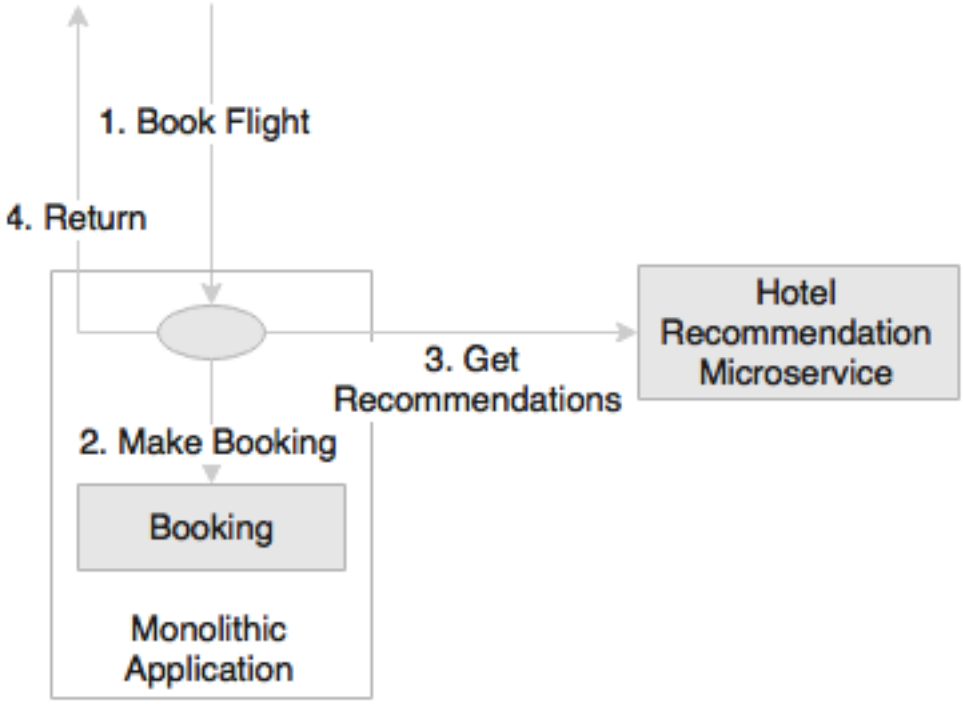

flybypoints
https://www.flybypoints.com
Fly By Points
Jeo
Welcome to Fly By Points Services
21123
Points
4
Offers
2
Trips
Destinations and More...

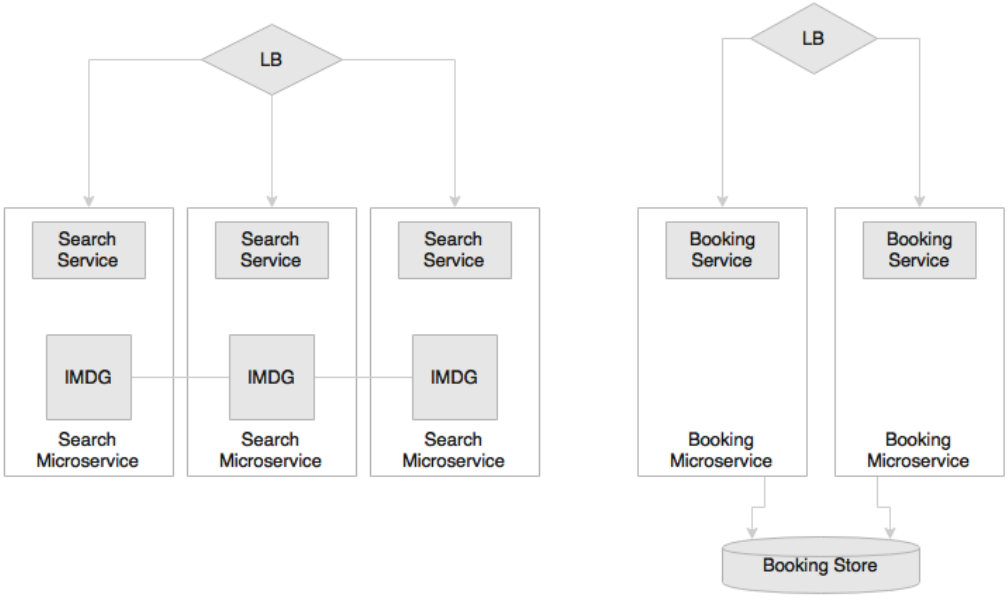

Spring MVC
Services
Entities
Repository
Points Subsystem
Spring MVC
Services
Entities
Repository
Offers Subsystem
Spring MVC
Services
Entities
Repository
Trips Subsystem
Internal Layers and Subsystems of the Architecture
flybypoints.war deployed on a
JEE web container

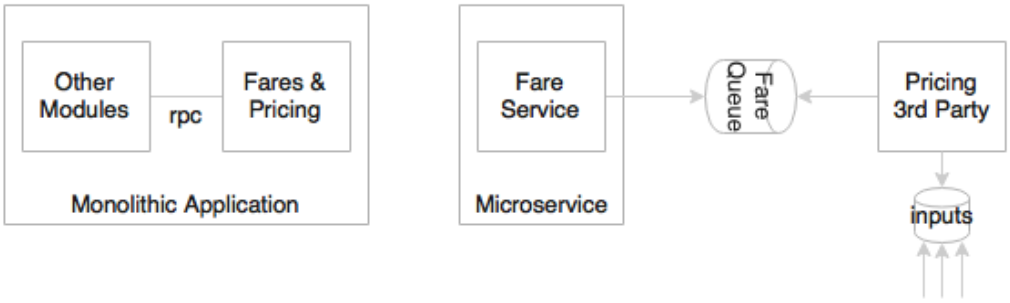

Points
Rest Service
Services
Entities
Repository
Points Subsystem
Data Store
Offers
Rest Service
Services
Entities
Repository
Offers Subsystem
Data Store
Trips
Rest Service
Services
Entities
Repository
Trips Subsystem
Data Store

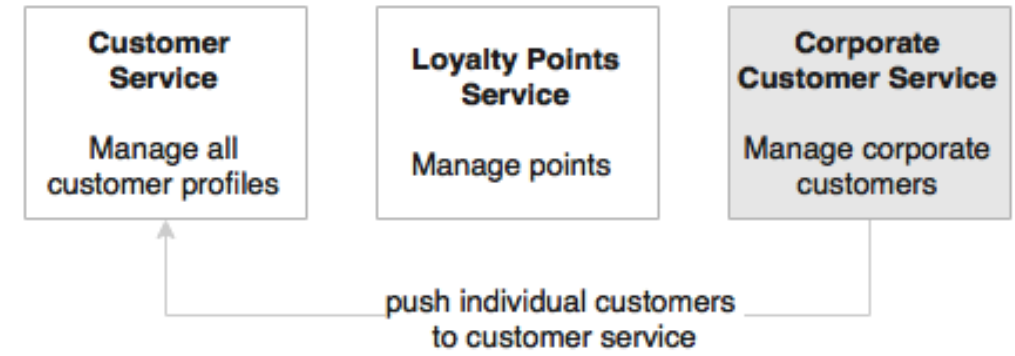

Customer UI Pages
REST
Customer Service
REST
Booking UI Pages
REST
Booking Service
REST
Visa UI Pages
REST
Visa Services
Event
Travel Agent Portal
Flight UI Pages
REST
Flight Services
CMS Layout Pages
Notification Services
Payment UI Pages

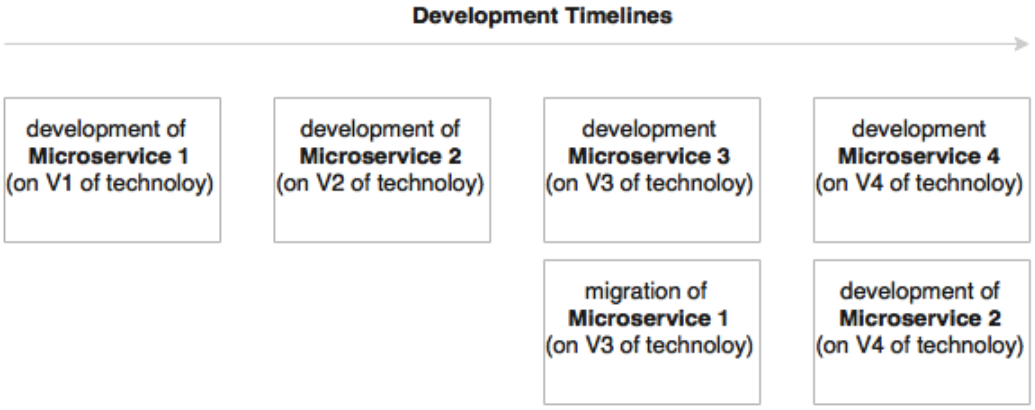

Services
Entity
My SQL
Order
Micoservice
Kafka
Flume
HDFS
Audit
Micoservice
Services
Entity
My SQL
Products
Micoservice

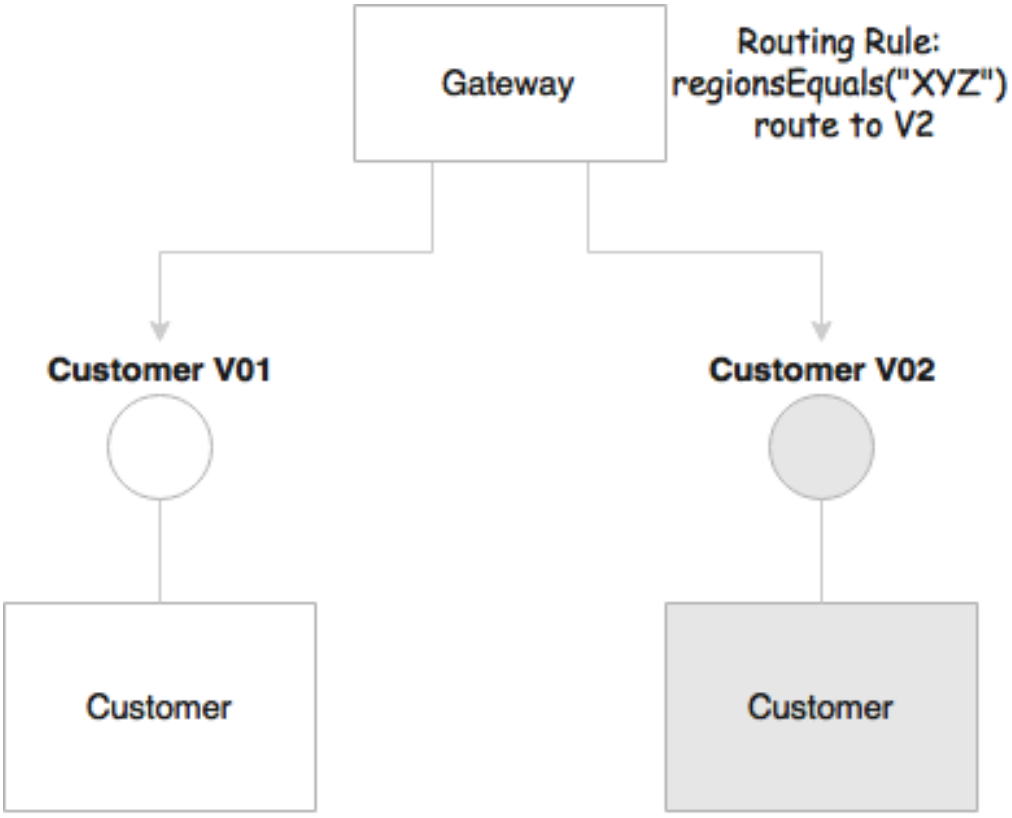

1. Book Flight
4. Return
Hotel
Recommendation
Microservice
3. Get
Recommendations
2. Make Booking
Booking
Monolithic
Application

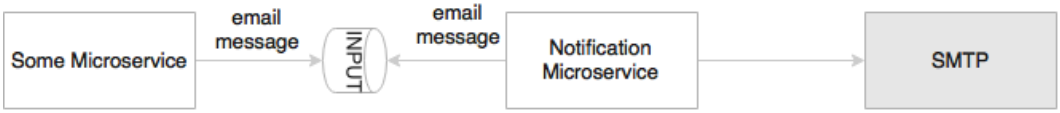

LB
Search Service
Search Service
Search Service
IMDG
IMDG
IMDG
Search Microservice
Search Microservice
Search Microservice
LB
Booking Service
Booking Service
Booking Microservice
Booking Microservice
Booking Store


Other Modules
rpc
Fares & Pricing
Monolithic Application
Fare Service
Fare Queue
Pricing 3rd Party
Microservice
inputs

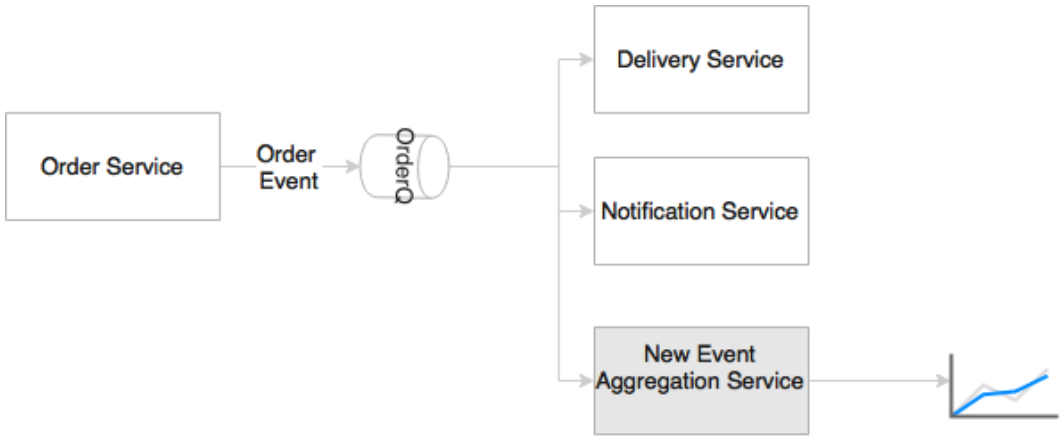

Customer Service
Manage all customer profiles
Loyalty Points Service
Manage points
Corporate Customer Service
Manage corporate customers
push individual customers to customer service


Development Timelines
development of Microservice 1 (on V1 of technoloy)
development of Microservice 2 (on V2 of technoloy)
development Microservice 3 (on V3 of technoloy)
development Microservice 4 (on V4 of technoloy)
migration of Microservice 1 (on V3 of technoloy)
development of Microservice 2 (on V4 of technoloy)

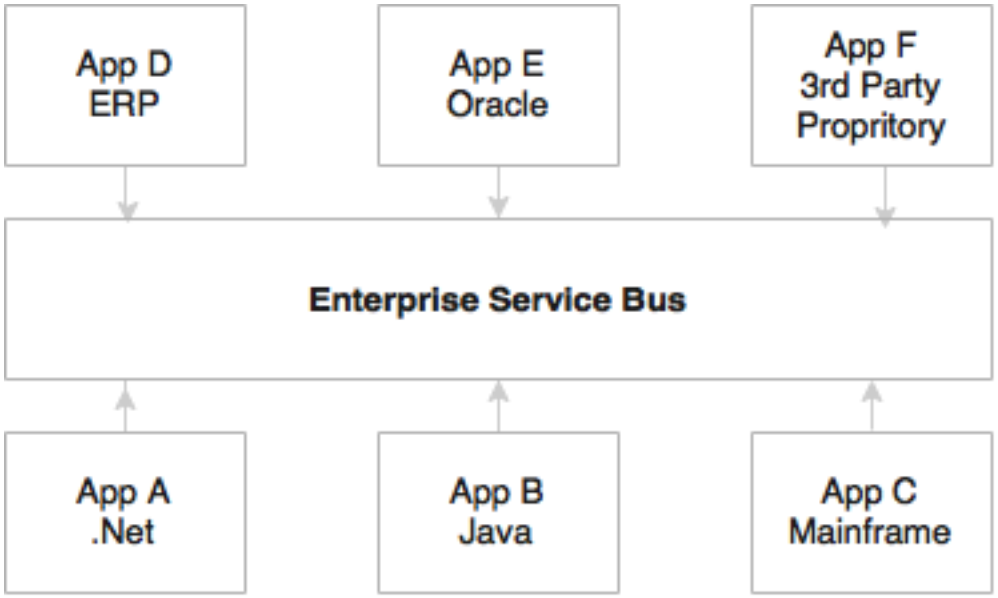

Gateway
Routing Rule:
regionsEquals("XYZ")
route to V2
Customer V01
Customer V02
Customer
Customer

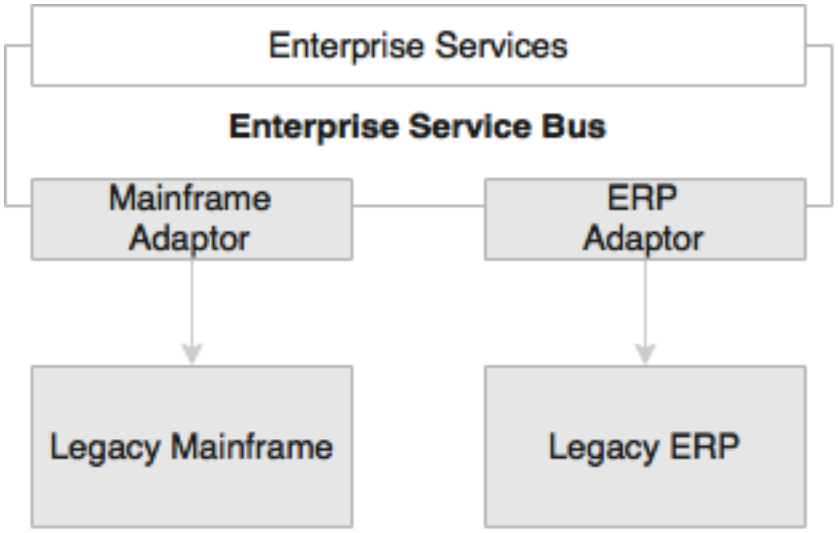

Some Microservice
email message
INPUT
email message
Notification Microservice
SMTP

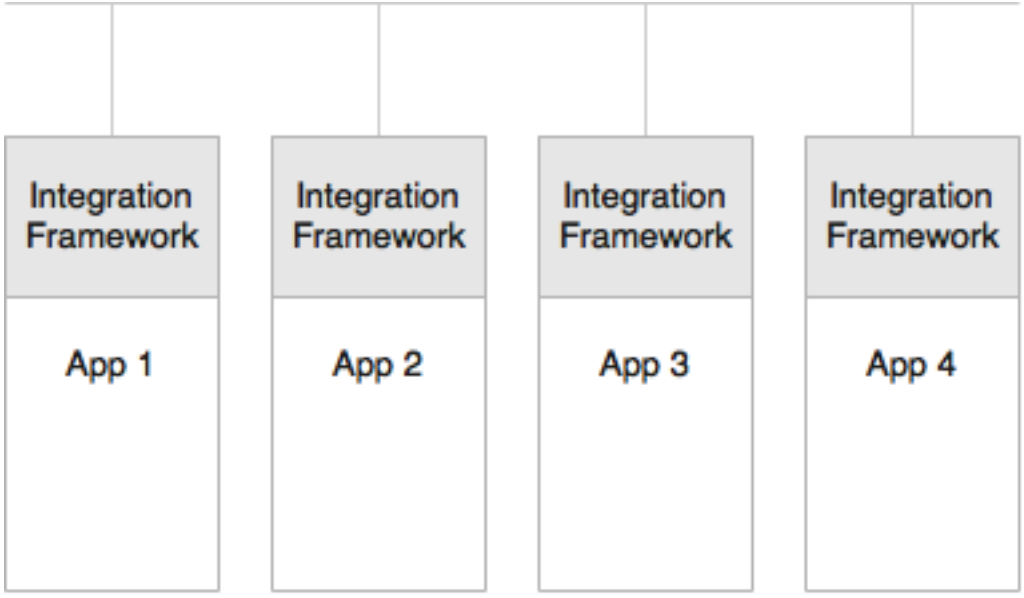
Some Microservice
email message
INPUT
email message
Personalization Microservice
email message
OUTPUT
email message
Notification Microservice
SMTP

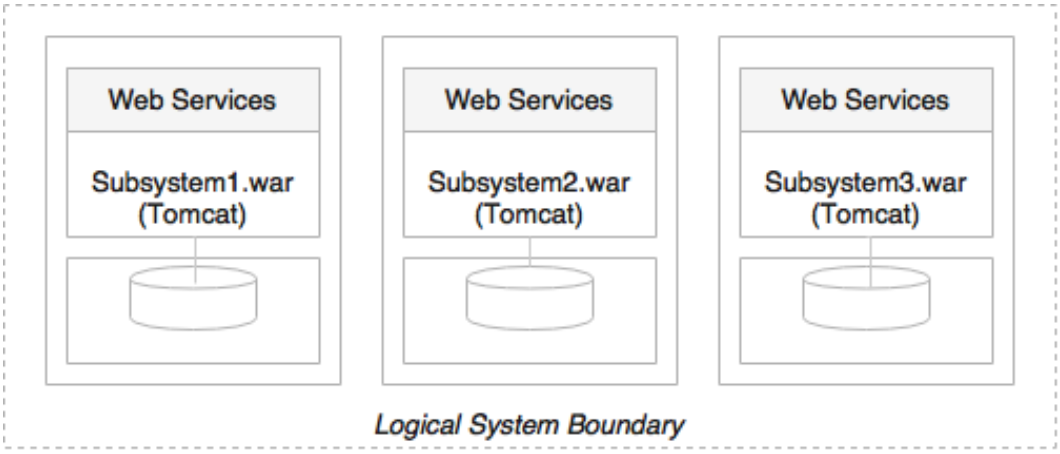

Order Service
Order
Event
OrderQ
Delivery Service
Notification Service
New Event
Aggregation Service

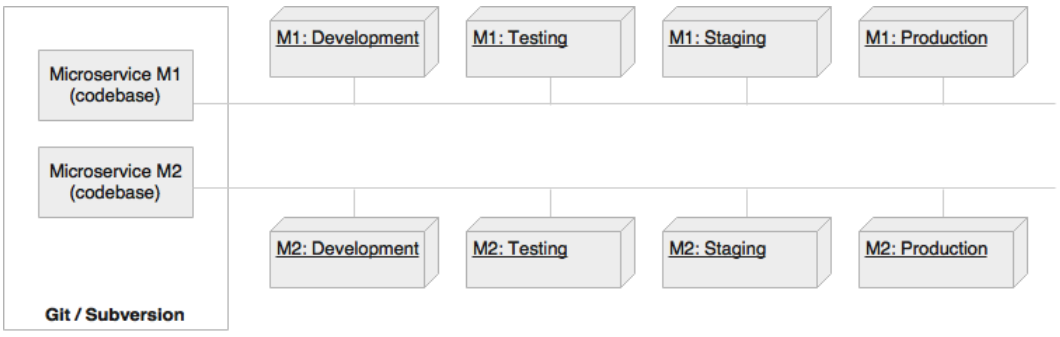
Chapter 2: Related Architecture Styles and Use Cases

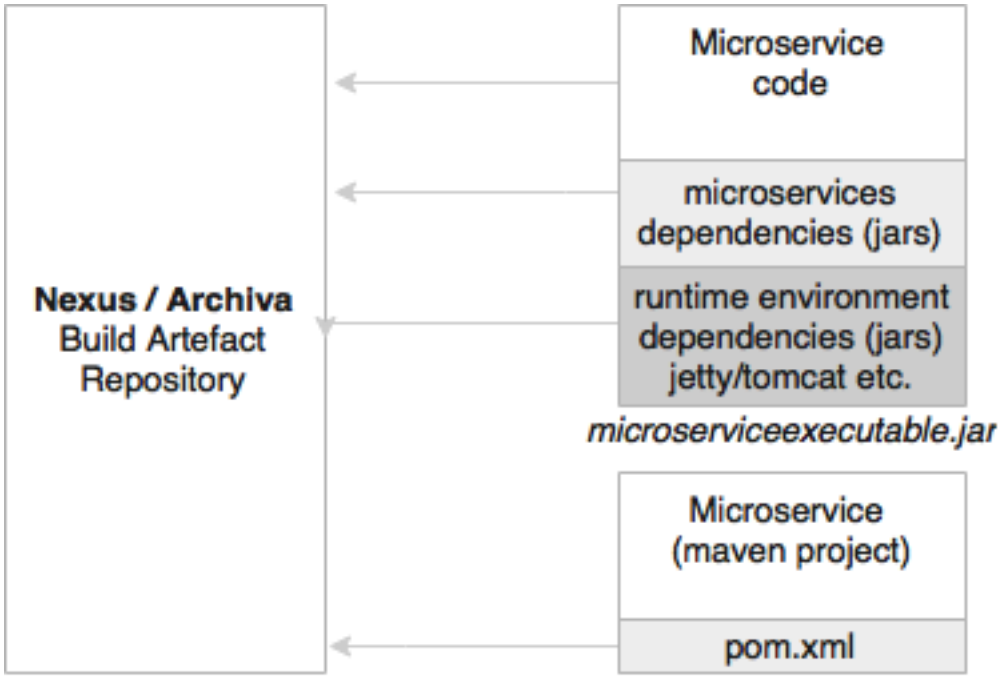


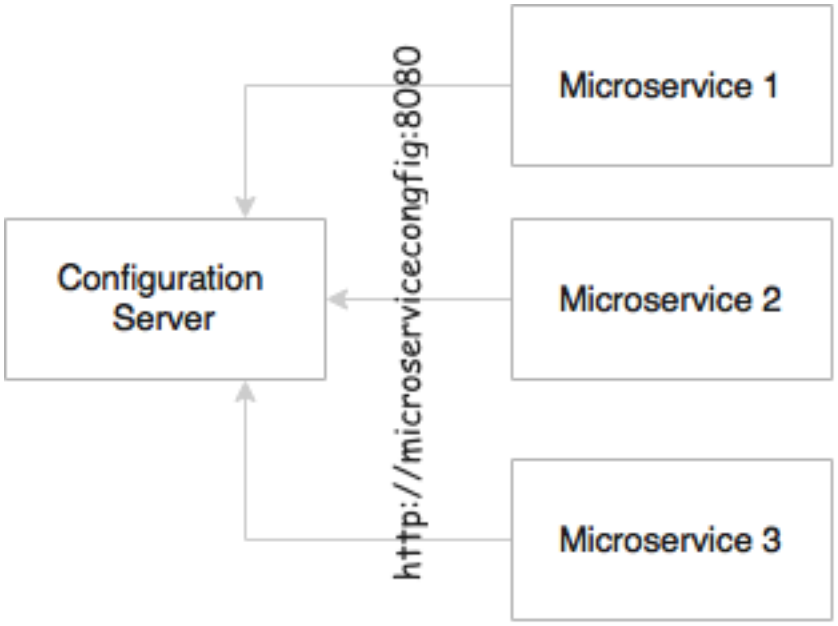

Enterprise Services
Enterprise Service Bus
Mainframe Adaptor
ERP Adaptor
Legacy Mainframe
Legacy ERP

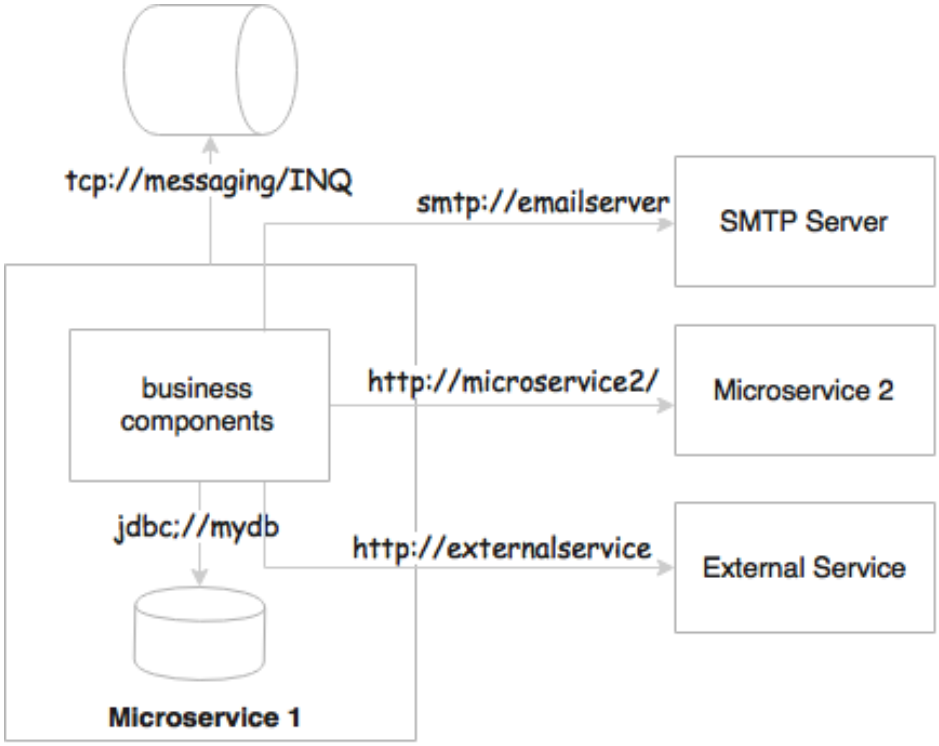

Integration Framework
Integration Framework
Integration Framework
Integration Framework
App 1
App 2
App 3
App 4

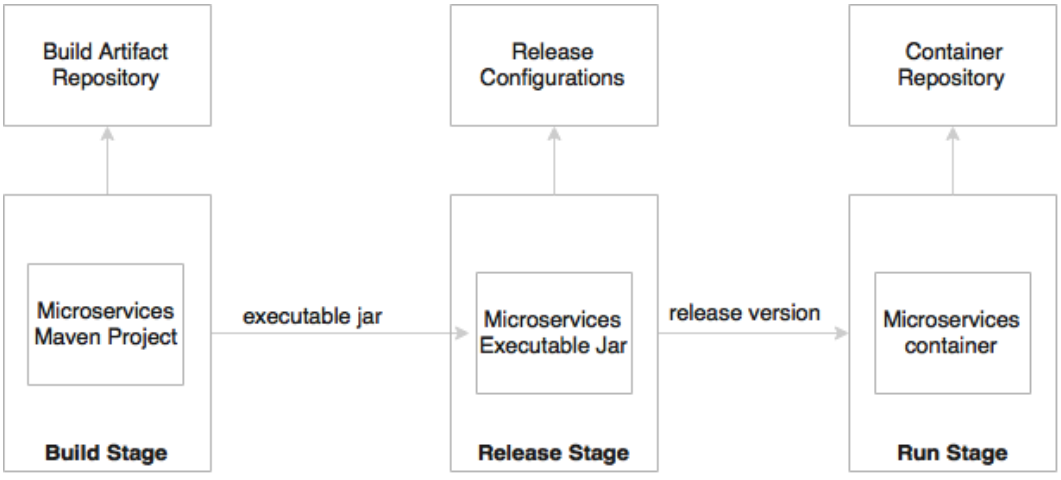

Web Services
Subsystem1.war
(Tomcat)
Web Services
Subsystem2.war
(Tomcat)
Web Services
Subsystem3.war
(Tomcat)
Logical System Boundary

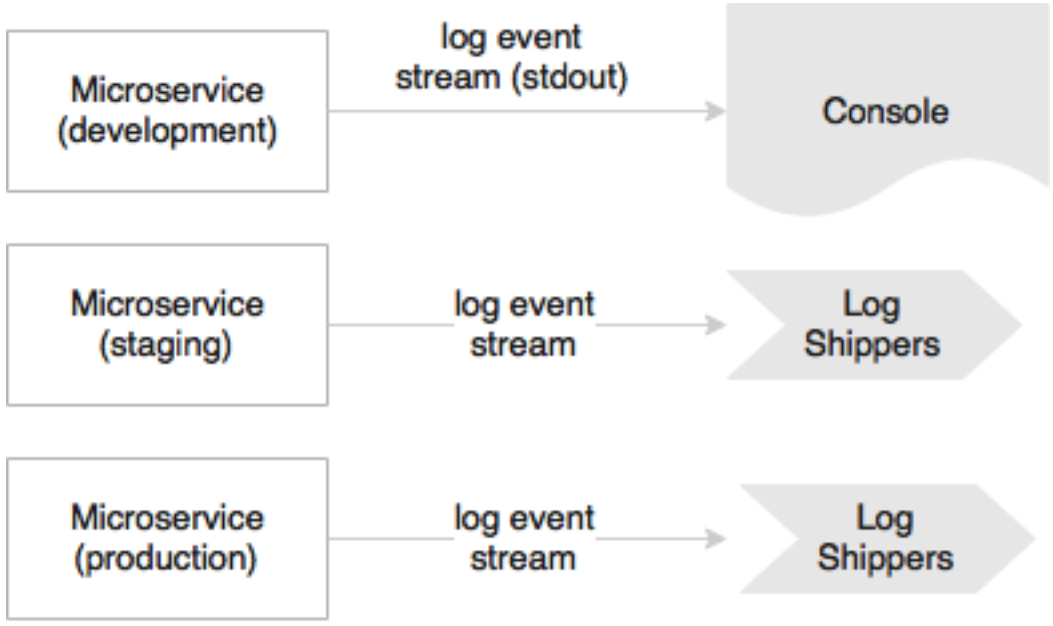

Microservice M1
(codebase)
Microservice M2
(codebase)
Git / Subversion
M1: Development
M1: Testing
M1: Staging
M1: Production
M2: Development
M2: Testing
M2: Staging
M2: Production

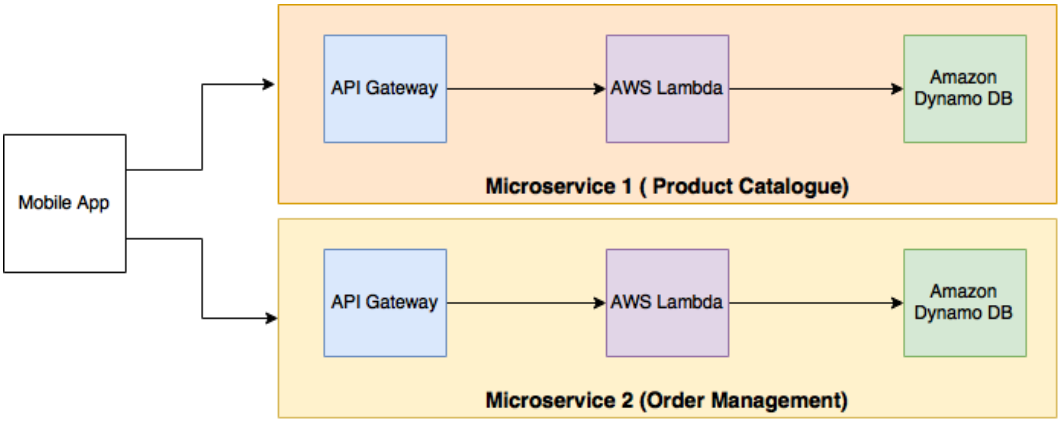

Nexus / Archiva
Build Artefact
Repository
Microservice
code
microservices
dependencies (jars)
runtime environment
dependencies (jars)
jetty/tomcat etc.
microserviceexecutable.jar
Microservice
(maven project)
pom.xml

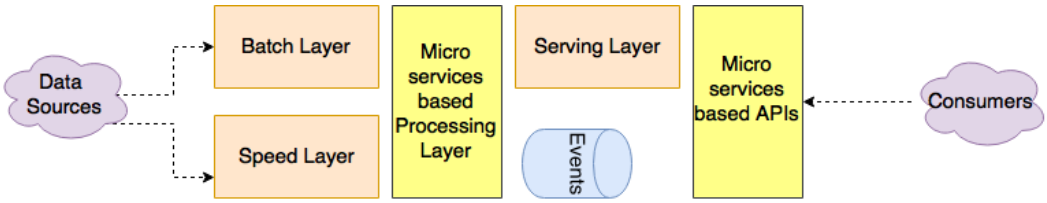

Microservice 1
Configuration Server
Microservice 2
http://microservicecongfig:8080
Microservice 3

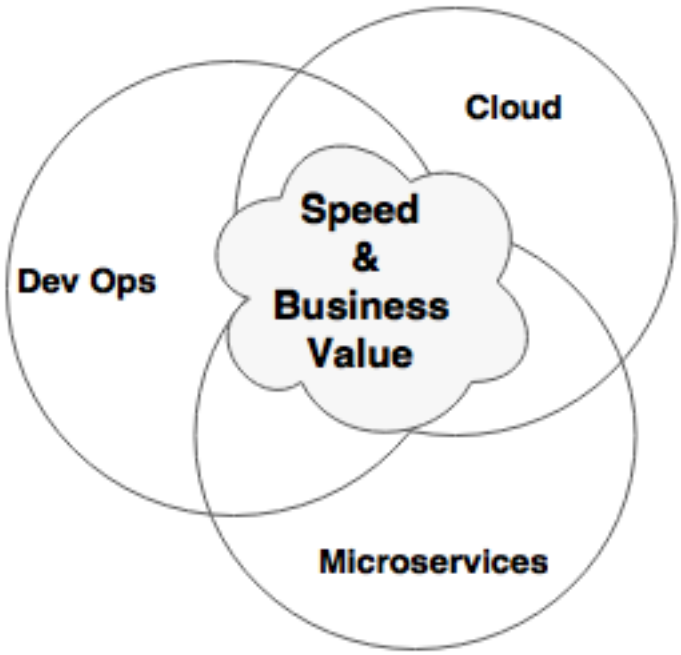

tcp://messaging/INQ
smtp://emailserver
SMTP Server
business components
http://microservice2/
Microservice 2
jdbc://mydb
http://externalservice
External Service
Microservice 1

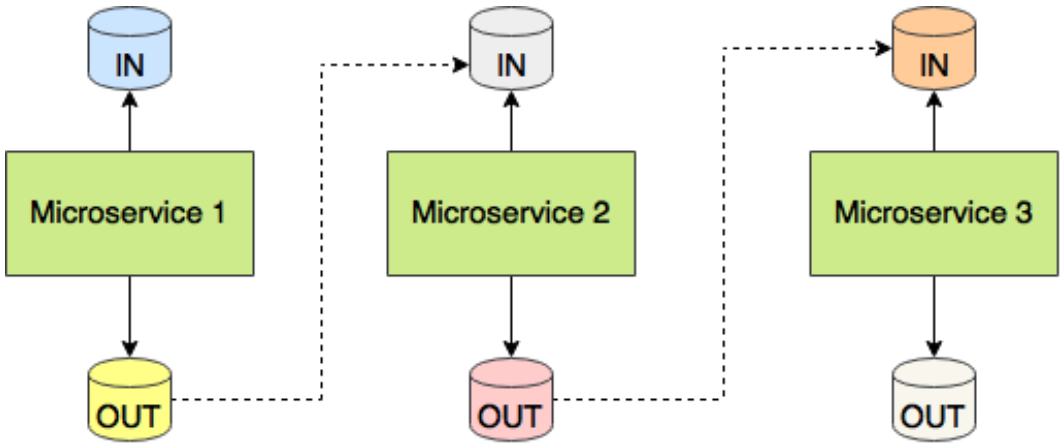

Build Artifact Repository
Release Configurations
Container Repository
Microservices Maven Project
executable jar
Microservices Executable Jar
release version
Microservices container
Build Stage
Release Stage
Run Stage

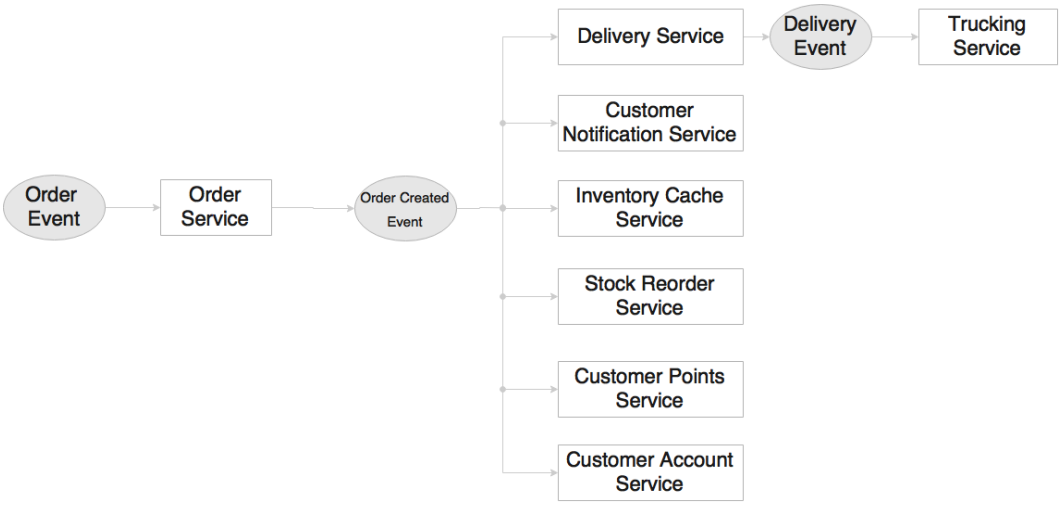

Microservice (development)
log event stream (stdout)
Console
Microservice (staging)
log event stream
Log Shippers
Microservice (production)
log event stream
Log Shippers


Mobile App
API Gateway
AWS Lambda
Amazon Dynamo DB
Microservice 1 (Product Catalogue)
API Gateway
AWS Lambda
Amazon Dynamo DB
Microservice 2 (Order Management)

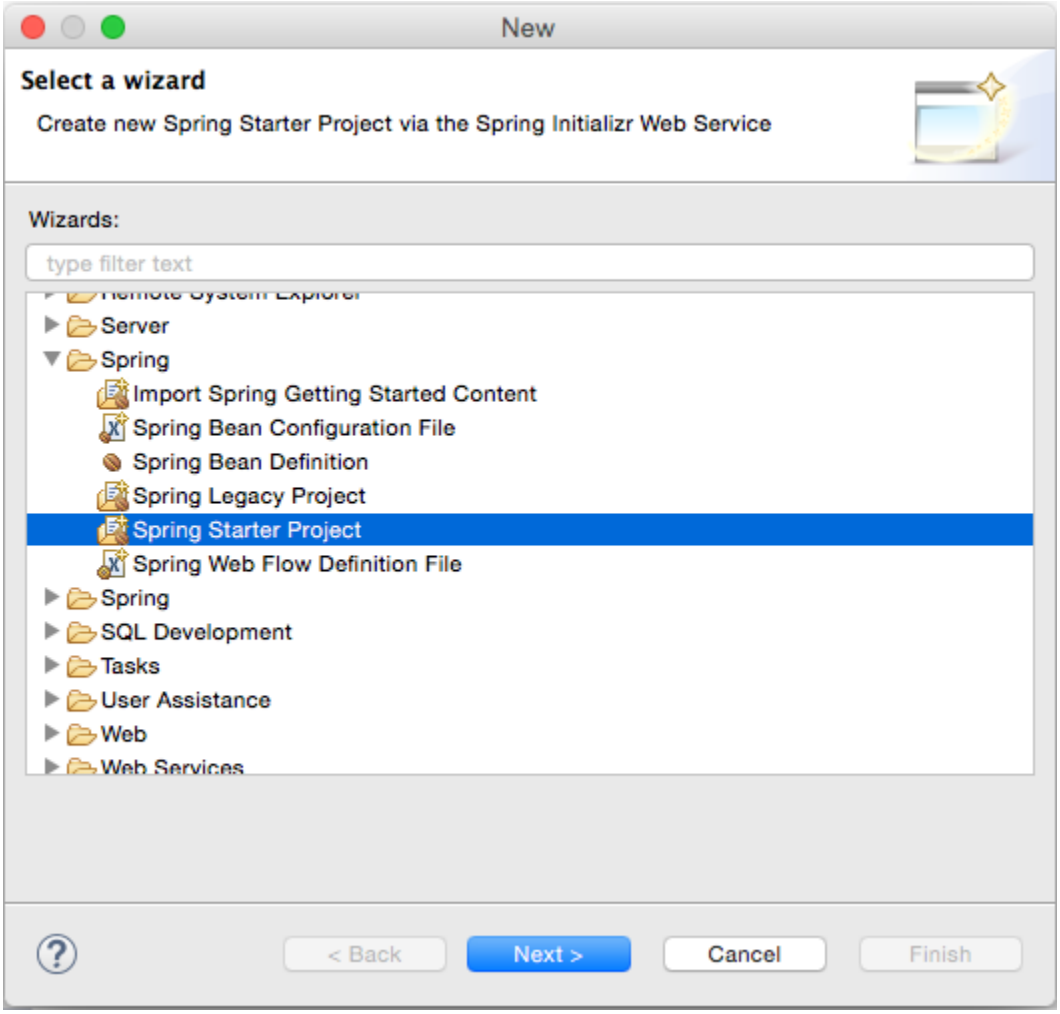

Data Sources
Batch Layer
Speed Layer
Micro services based Processing Layer
Serving Layer
Events
Micro services based APIs
Consumers

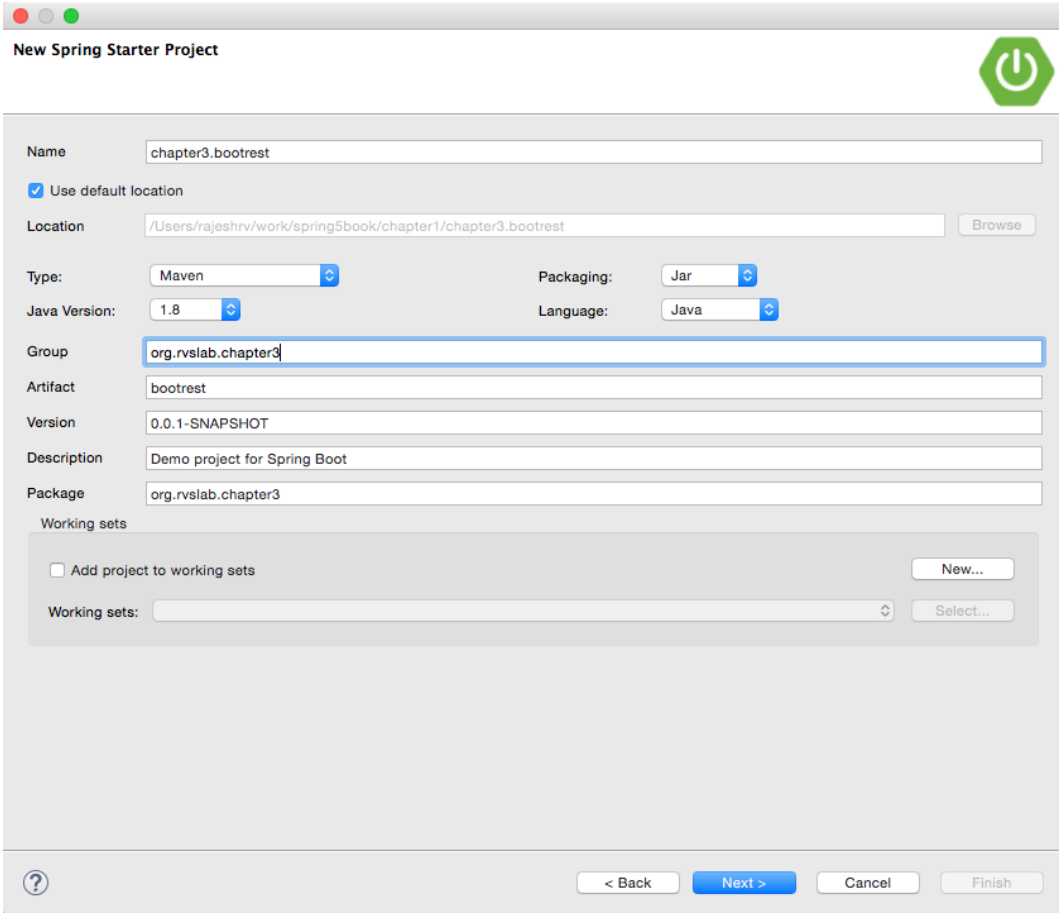

Cloud
Dev Ops
Speed & Business Value
Microservices

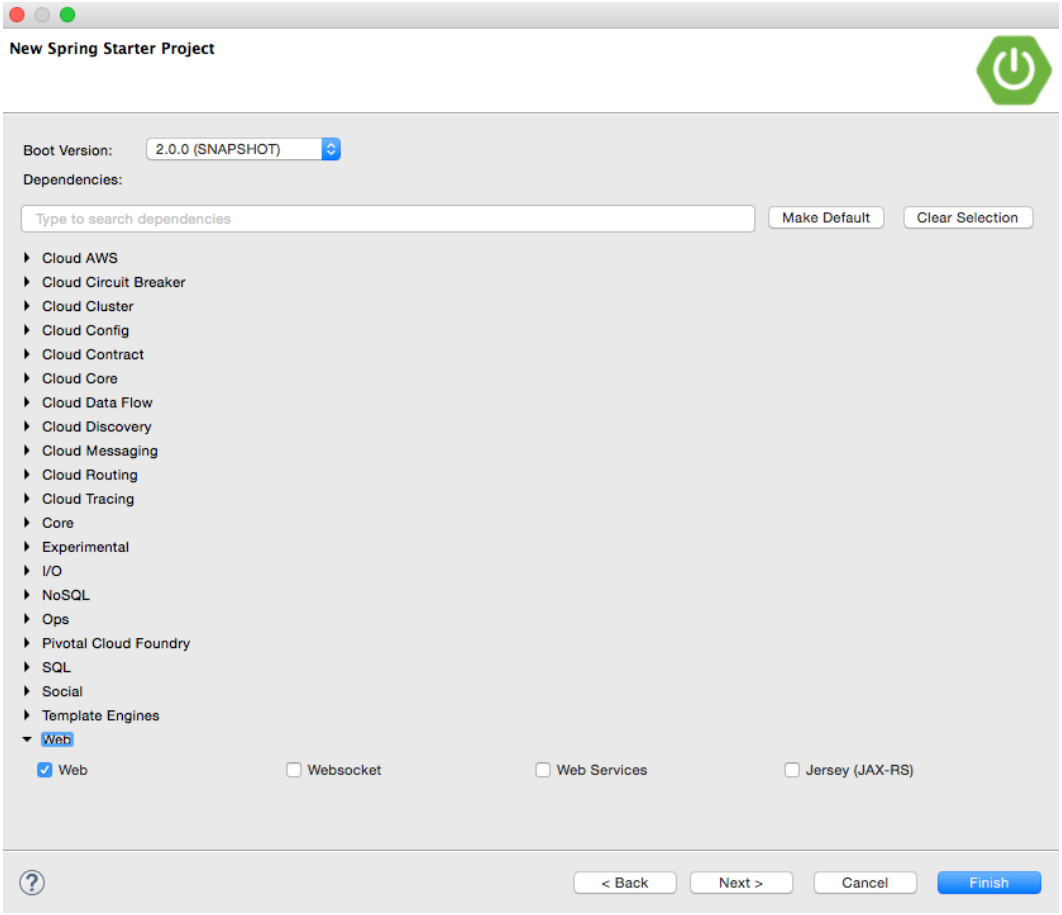

IN
Microservice 1
OUT
IN
Microservice 2
OUT
IN
Microservice 3
OUT

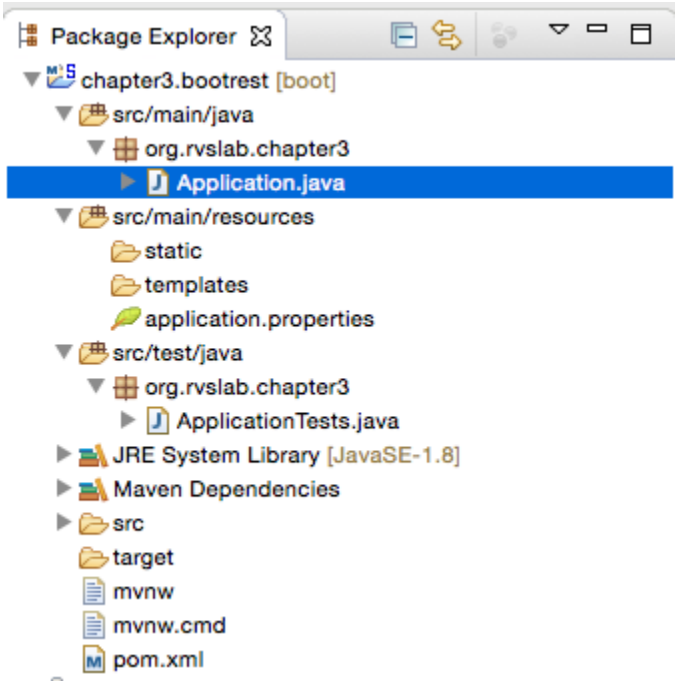

Order Event
Order Service
Order Created Event
Delivery Service
Delivery Event
Trucking Service
Customer Notification Service
Inventory Cache Service
Stock Reorder Service
Customer Points Service
Customer Account Service

Chapter 3: Building Microservices with Spring Boot



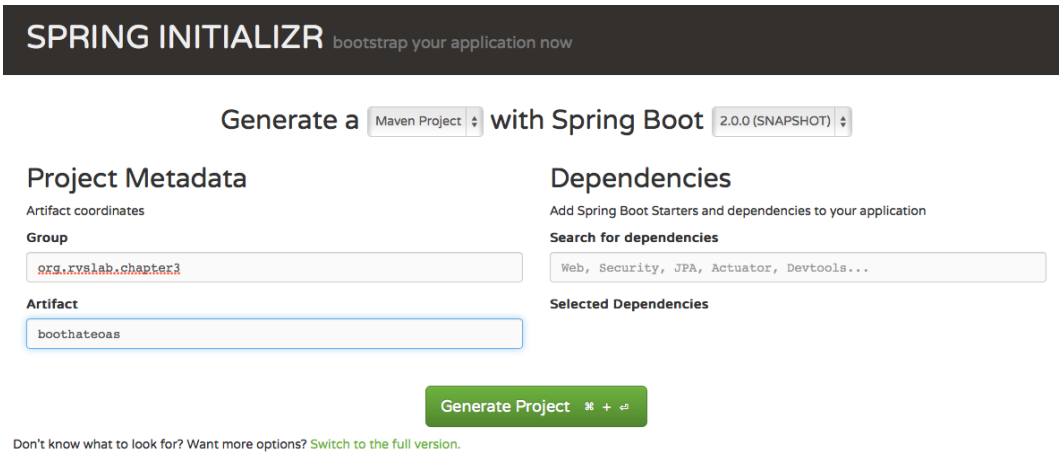

New Spring Starter Project
Name
chapter3.bootrest
Use default location
Location
/Users/rajeshrv/work/spring5book/chapter1/chapter3.bootrest
Browse
Type:
Maven
Packaging:
Jar
Java Version:
1.8
Language:
Java
Group
org.rvslab.chapter3
Artifact
bootrest
Version
0.0.1-SNAPSHOT
Description
Demo project for Spring Boot
Package
org.rvslab.chapter3
Working sets
Add project to working sets
New...
Working sets:
Select...
< Back
Next >
Cancel
Finish


New Spring Starter Project
Boot Version:
2.0.0 (SNAPSHOT)
Dependencies:
Type to search dependencies
Make Default
Clear Selection
Cloud AWS
Cloud Circuit Breaker
Cloud Cluster
Cloud Config
Cloud Contract
Cloud Core
Cloud Data Flow
Cloud Discovery
Cloud Messaging
Cloud Routing
Cloud Tracing
Core
Experimental
I/O
NoSQL
Ops
Pivotal Cloud Foundry
SQL
Social
Template Engines
Web
Web
Websocket
Web Services
Jersey (JAX-RS)
< Back
Next >
Cancel
Finish



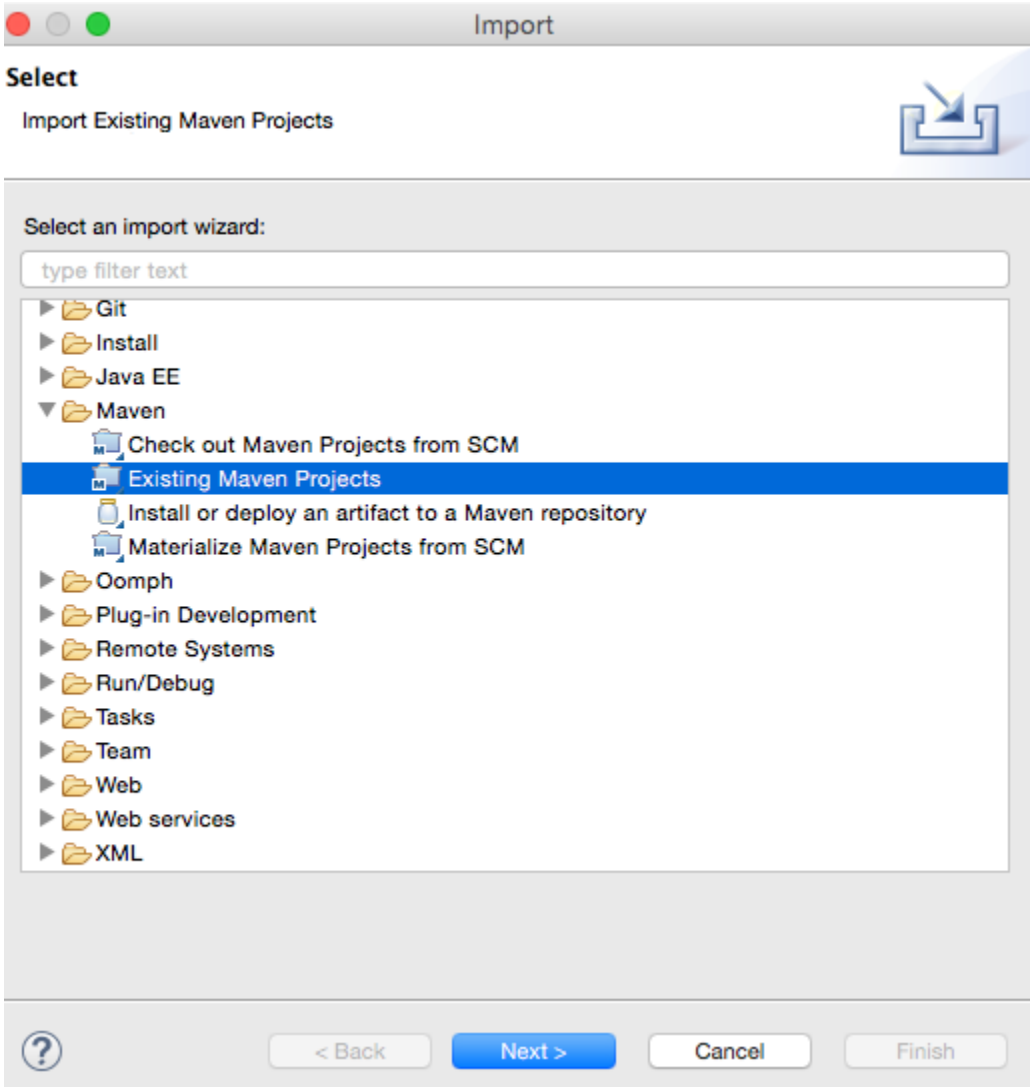

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::  (v2.0.0.BUILD-SNAPSHOT)

2016-12-16 13:40:35.807  INFO 32654 --- [           main] org.rvslab.chapter3.Application          :
Starting Application on rvslab.local with PID 32654 (/Users/rajeshrv/work/spring5book/chapter3/
```

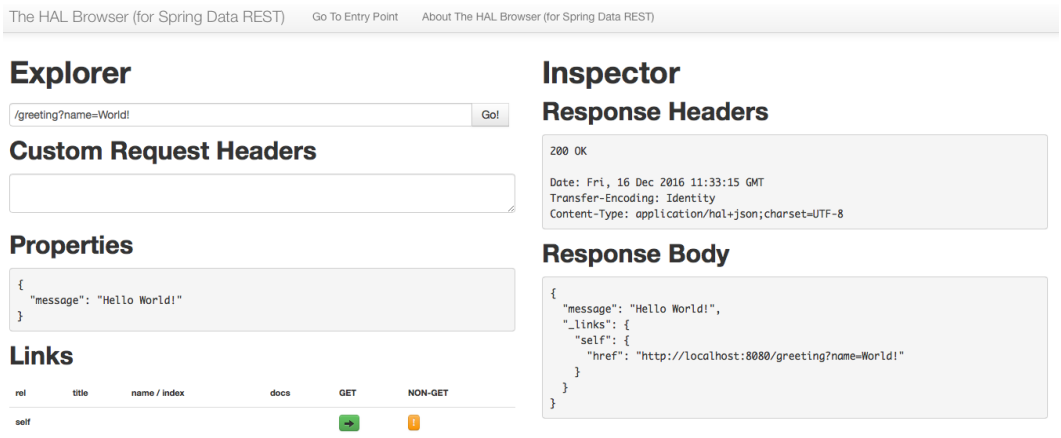
```
{"message":"Hello World!"}
```

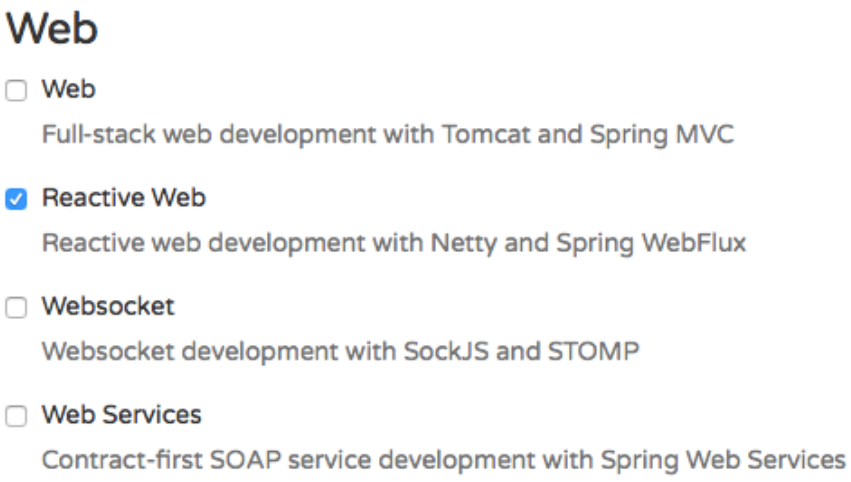

SPRING INITIALIZR bootstrap your application now
Generate a Maven Project with Spring Boot 2.0.0 (SNAPSHOT)
Project Metadata
Artifact coordinates
Group
org.rvslab.chapter3
Artifact
boothateoas
Dependencies
Add Spring Boot Starters and dependencies to your application
Search for dependencies
Web, Security, JPA, Actuator, Devtools...
Selected Dependencies
Generate Project ⌘ + ⏎
Don't know what to look for? Want more options? Switch to the full version.

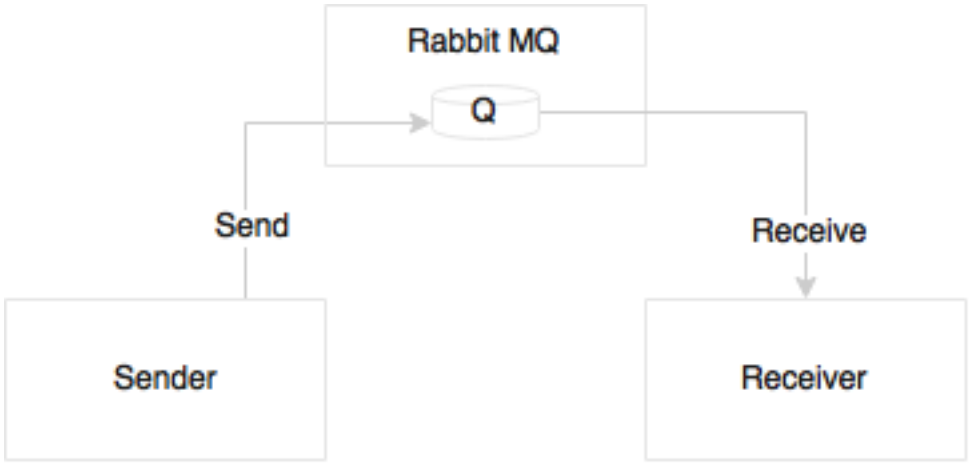
Web

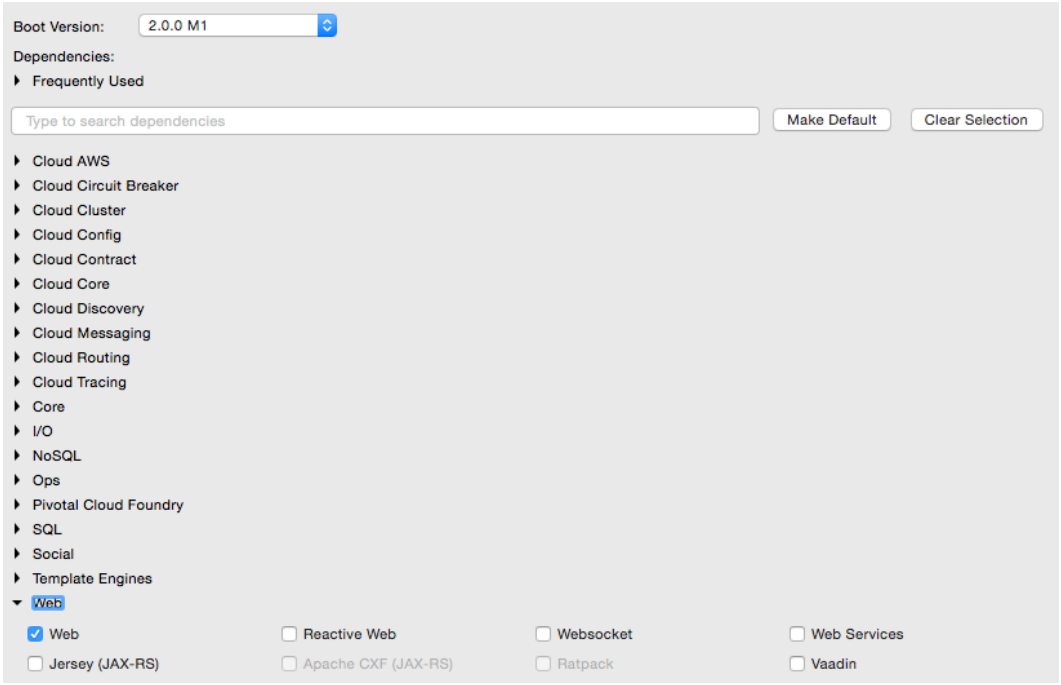
- [x] **Web**
  Full-stack web development with Tomcat and Spring MVC
- [ ] **Websocket**
  Websocket development with SockJS and STOMP
- [ ] **Web Services**
  Contract-first SOAP service development with Spring Web Services
- [ ] **Jersey (JAX-RS)**
  RESTful Web Services framework
- [ ] **Ratpack**
  Spring Boot integration for the Ratpack framework
- [ ] **Vaadin**
  Vaadin java web application framework
- [ ] **Rest Repositories**
  Exposing Spring Data repositories over REST via spring-data-rest-webmvc
- [x] **HATEOAS**
  HATEOAS-based RESTful services
- [x] **Rest Repositories HAL Browser**
  Browsing Spring Data REST repositories in your browser
- [ ] **Mobile**
  Simplify the development of mobile web applications with spring-mobile
- [ ] **REST Docs**
  Document RESTful services by combining hand-written and auto-generated

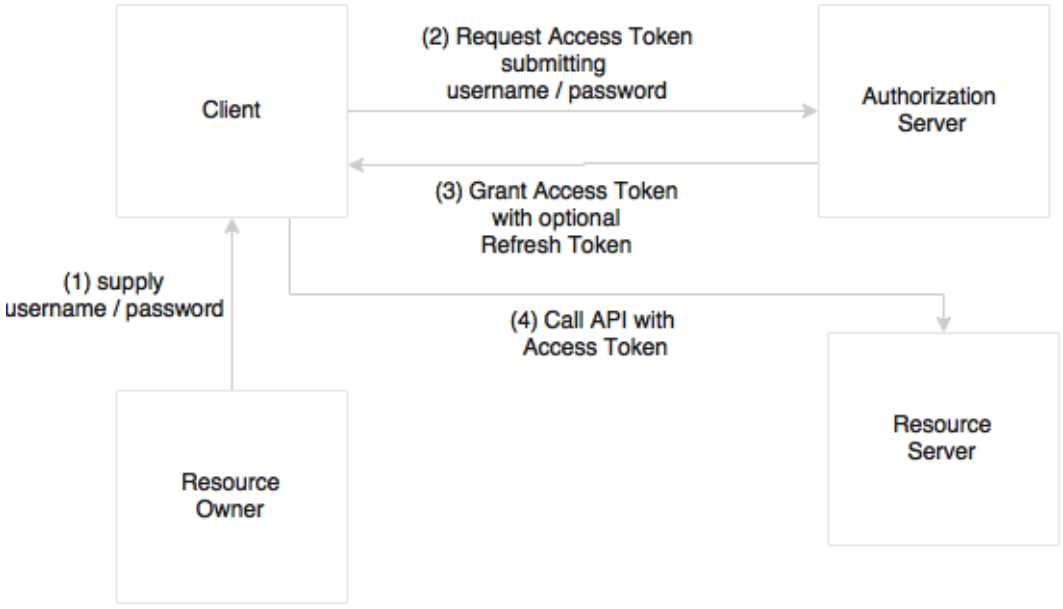

Import
Select
Import Existing Maven Projects
Select an import wizard:
type filter text
Git
Install
Java EE
Maven
Check out Maven Projects from SCM
Existing Maven Projects
Install or deploy an artifact to a Maven repository
Materialize Maven Projects from SCM
Oomph
Plug-in Development
Remote Systems
Run/Debug
Tasks
Team
Web
Web services
XML
< Back
Next >
Cancel
Finish

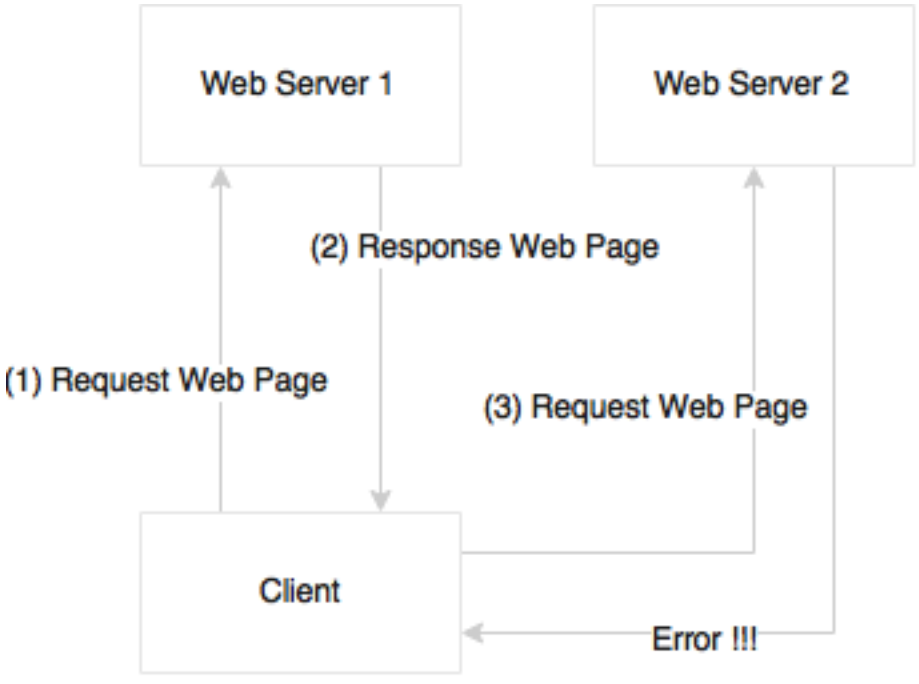

The HAL Browser (for Spring Data REST)
Go To Entry Point
About The HAL Browser (for Spring Data REST)
Explorer
/greeting?name=World!
Go!
Custom Request Headers
Properties
{
"message": "Hello World!"
}
Links
rel
title
name / index
docs
GET
NON-GET
self
Inspector
Response Headers
200 OK
Date: Fri, 16 Dec 2016 11:33:15 GMT
Transfer-Encoding: Identity
Content-Type: application/hal+json;charset=UTF-8
Response Body
{
"message": "Hello World!",
"_links": {
"self": {
"href": "http://localhost:8080/greeting?name=World!"
}
}
}


Web
Web
Full-stack web development with Tomcat and Spring MVC
Reactive Web
Reactive web development with Netty and Spring WebFlux
Websocket
Websocket development with SockJS and STOMP
Web Services
Contract-first SOAP service development with Spring Web Services


Rabbit MQ
Q
Send
Receive
Sender
Receiver

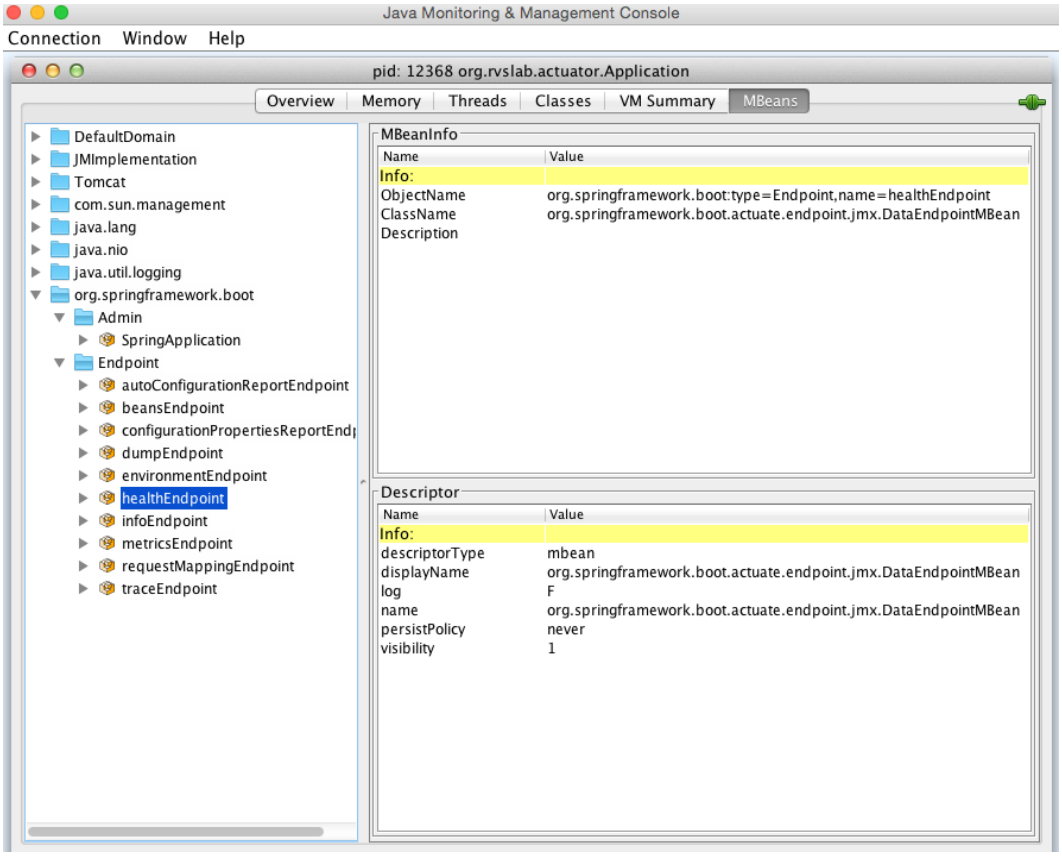

Boot Version:
2.0.0 M1
Dependencies:
Frequently Used
Type to search dependencies
Make Default
Clear Selection
Cloud AWS
Cloud Circuit Breaker
Cloud Cluster
Cloud Config
Cloud Contract
Cloud Core
Cloud Discovery
Cloud Messaging
Cloud Routing
Cloud Tracing
Core
I/O
NoSQL
Ops
Pivotal Cloud Foundry
SQL
Social
Template Engines
Web
Web
Reactive Web
Websocket
Web Services
Jersey (JAX-RS)
Apache CXF (JAX-RS)
Ratpack
Vaadin

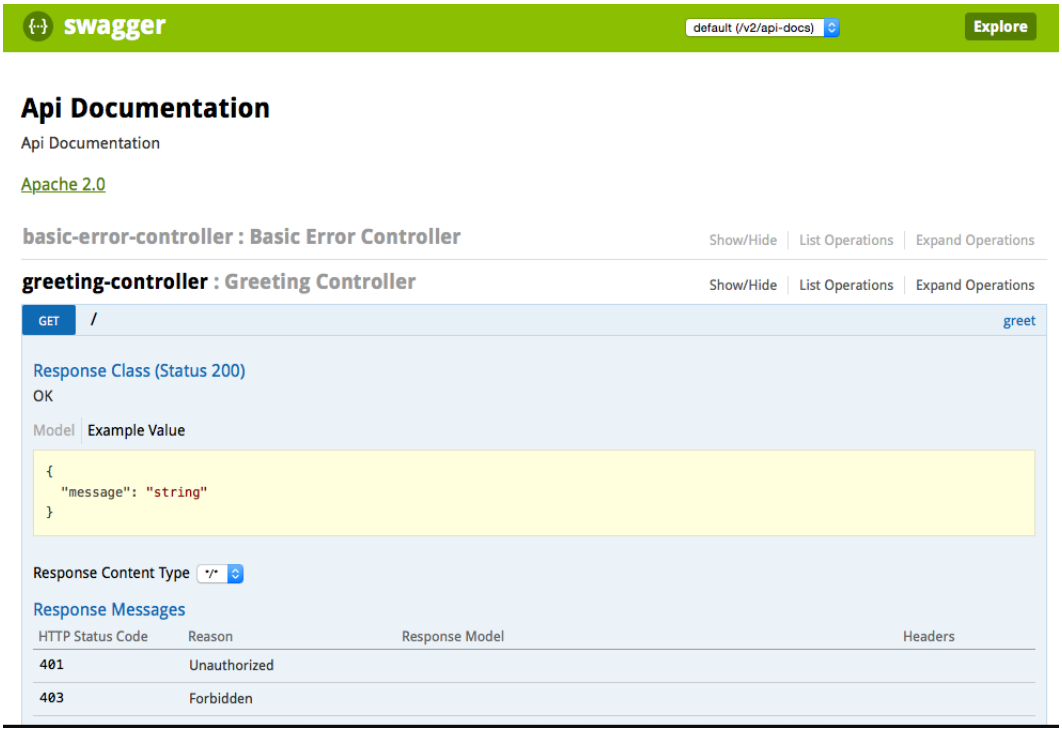

Client
(2) Request Access Token
submitting
username / password
Authorization
Server
(3) Grant Access Token
with optional
Refresh Token
(1) supply
username / password
(4) Call API with
Access Token
Resource
Server
Resource
Owner

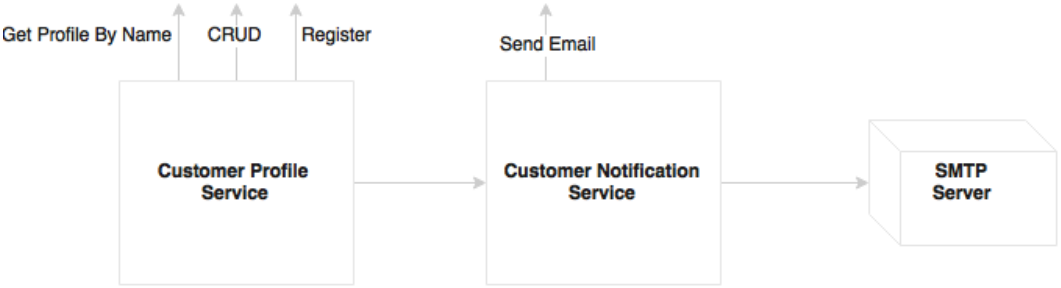

Web Server 1
Web Server 2
(2) Response Web Page
(1) Request Web Page
(3) Request Web Page
Client
Error !!!

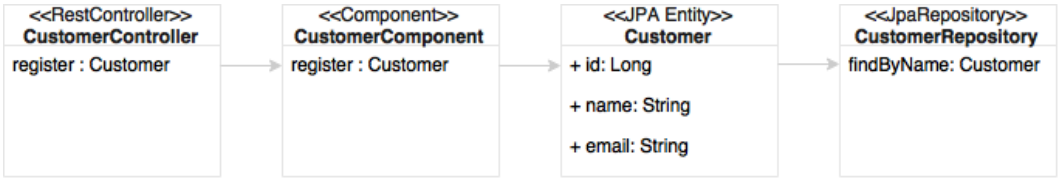
Links

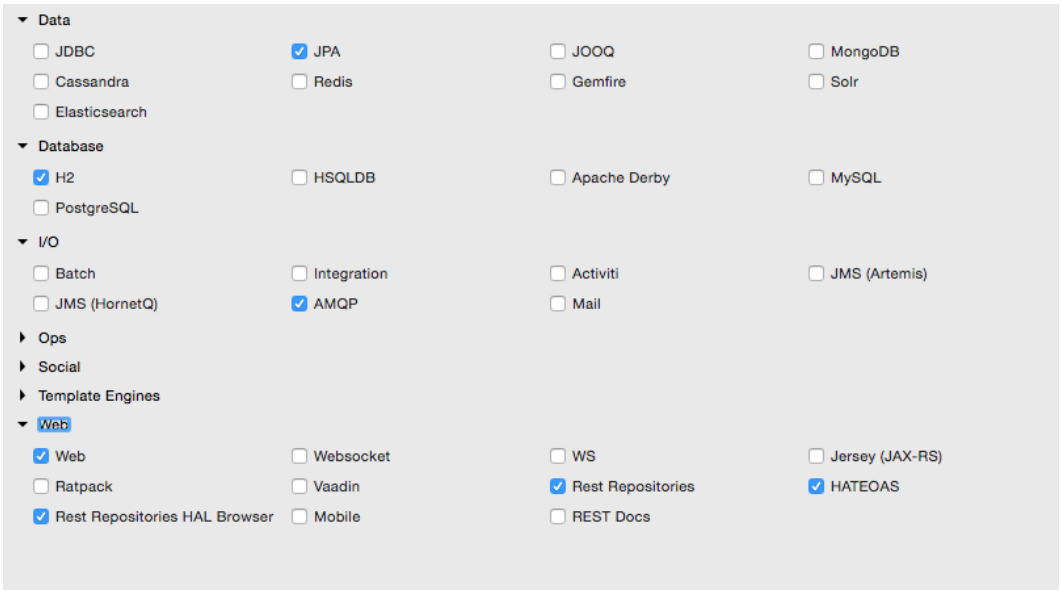
rel	title	name / index	docs	GET	NON-GET
self				➡	!
health				➡	!
trace				➡	!
dump				➡	!
loggers				➡	!
configprops				➡	!
beans				➡	!
info				➡	!
autoconfig				➡	!
env				➡	!
metrics				➡	!
mappings				➡	!
auditevents				➡	!
heapdump				➡	!

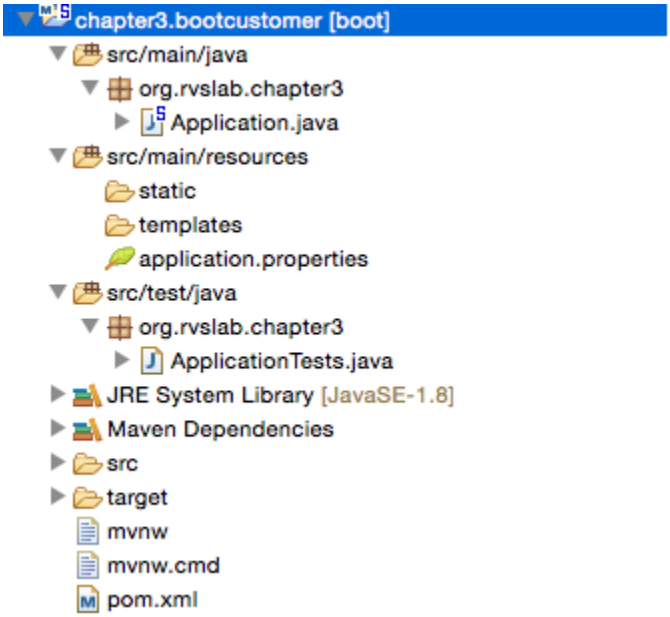

Java Monitoring & Management Console
Connection
Window
Help
pid: 12368 org.rvslab.actuator.Application
Overview
Memory
Threads
Classes
VM Summary
MBeans
DefaultDomain
JMImplementation
Tomcat
com.sun.management
java.lang
java.nio
java.util.logging
org.springframework.boot
Admin
SpringApplication
Endpoint
autoConfigurationReportEndpoint
beansEndpoint
configurationPropertiesReportEndp
dumpEndpoint
environmentEndpoint
healthEndpoint
infoEndpoint
metricsEndpoint
requestMappingEndpoint
traceEndpoint
MBeanInfo
Name
Value
Info:
ObjectName
org.springframework.boot:type=Endpoint,name=healthEndpoint
ClassName
org.springframework.boot.actuate.endpoint.jmx.DataEndpointMBean
Description
Descriptor
Name
Value
Info:
descriptorType
mbean
displayName
org.springframework.boot.actuate.endpoint.jmx.DataEndpointMBean
log
F
name
org.springframework.boot.actuate.endpoint.jmx.DataEndpointMBean
persistPolicy
never
visibility
1


swagger
default (/v2/api-docs)
Explore
Api Documentation
Api Documentation
Apache 2.0
basic-error-controller : Basic Error Controller
Show/Hide
List Operations
Expand Operations
greeting-controller : Greeting Controller
Show/Hide
List Operations
Expand Operations
GET
/
greet
Response Class (Status 200)
OK
Model
Example Value
{
"message": "string"
}
Response Content Type
Response Messages
HTTP Status Code
Reason
Response Model
Headers
401
Unauthorized
403
Forbidden


Get Profile By Name
CRUD
Register
Send Email
Customer Profile Service
Customer Notification Service
SMTP Server


<<RestController>>
CustomerController
register : Customer
<<Component>>
CustomerComponent
register : Customer
<<JPA Entity>>
Customer
+ id: Long
+ name: String
+ email: String
<<JpaRepository>>
CustomerRepository
findByName: Customer


Data
JDBC
JPA
JOOQ
MongoDB
Cassandra
Redis
Gemfire
Solr
Elasticsearch
Database
H2
HSQLDB
Apache Derby
MySQL
PostgreSQL
I/O
Batch
Integration
Activiti
JMS (Artemis)
JMS (HornetQ)
AMQP
Mail
Ops
Social
Template Engines
Web
Web
Websocket
WS
Jersey (JAX-RS)
Ratpack
Vaadin
Rest Repositories
HATEOAS
Rest Repositories HAL Browser
Mobile
REST Docs

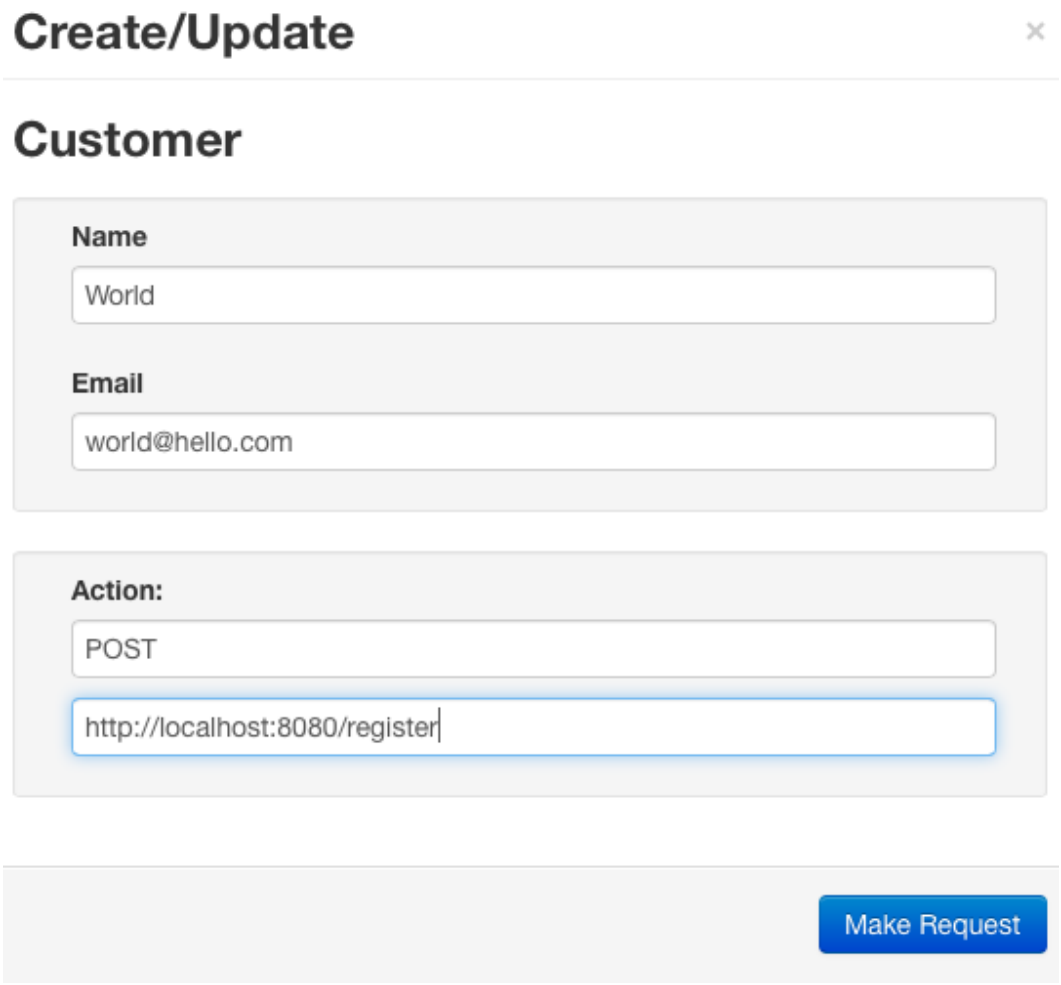


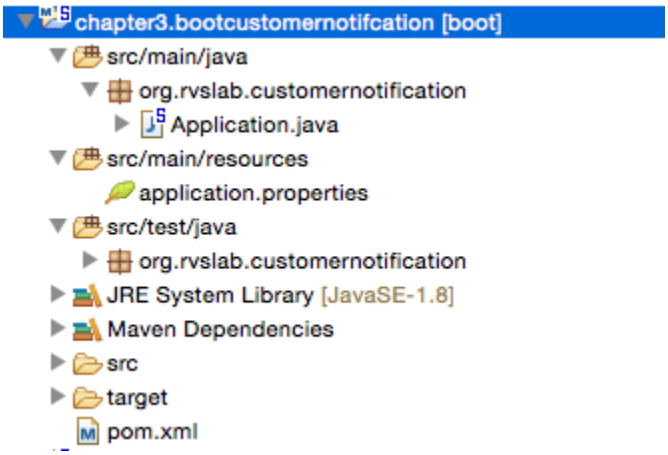
Links

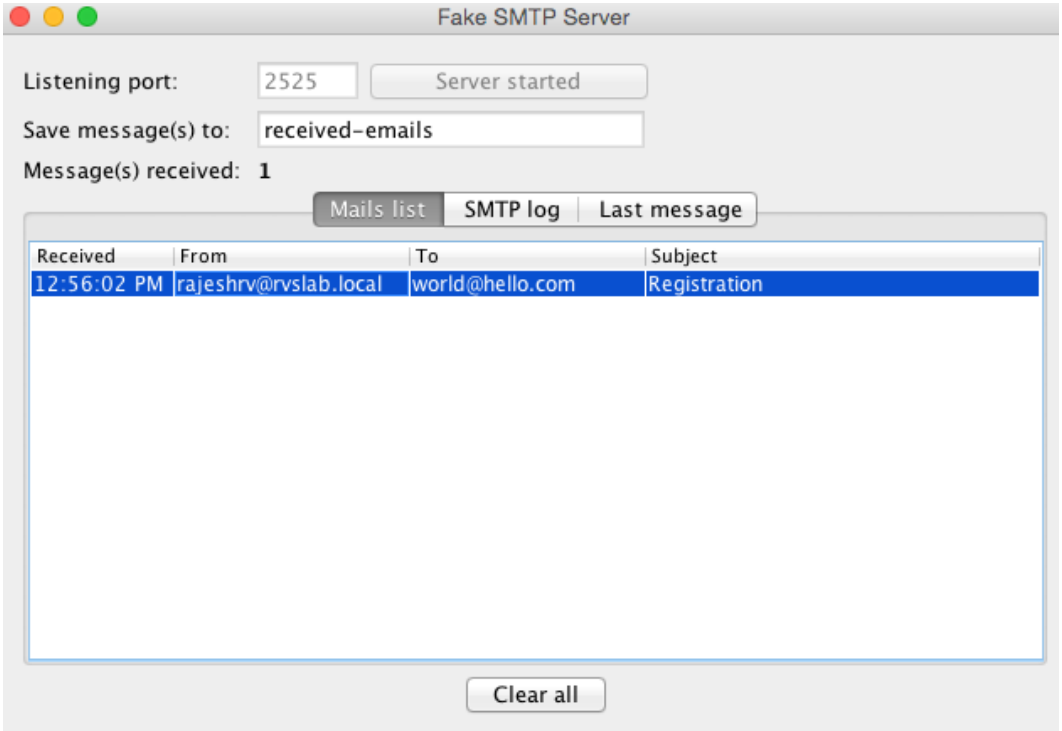
rel	title	name / index	docs	GET	NON-GET
first				→	!
prev				→	!
self				→	!
next				→	!
last				→	!
profile				→	!
search				→	!

Links

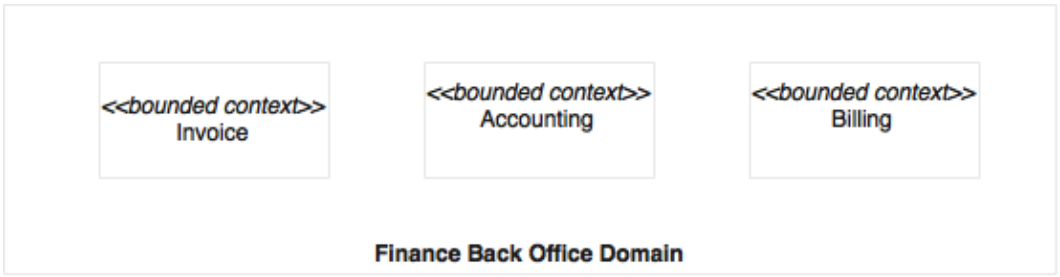
rel	title	name / index	docs	GET	NON-GET
self				→	!
profile				→	! Perform non-GET rec
search				→	!

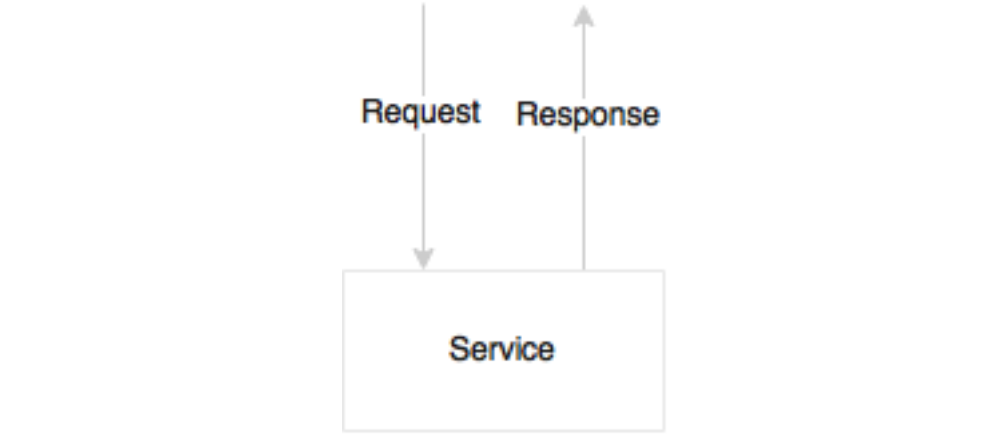

Create/Update
Customer
Name
World
Email
world@hello.com
Action:
POST
http://localhost:8080/register
Make Request

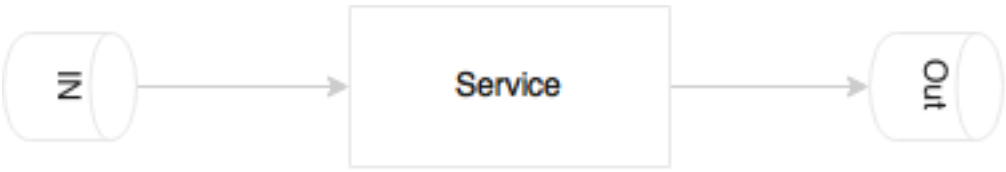

chapter3.bootcustomernotifcation [boot]
src/main/java
org.rvslab.customernotification
Application.java
src/main/resources
application.properties
src/test/java
org.rvslab.customernotification
JRE System Library [JavaSE-1.8]
Maven Dependencies
src
target
pom.xml


Fake SMTP Server
Listening port:
2525
Server started
Save message(s) to:
received-emails
Message(s) received: 1
Mails list
SMTP log
Last message
Received
From
To
Subject
12:56:02 PM
rajeshrv@rvslab.local
world@hello.com
Registration
Clear all

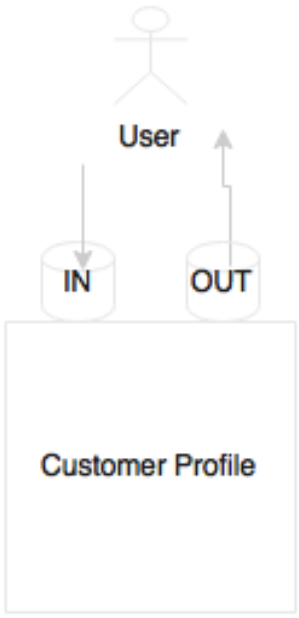
Chapter 4: Applying Microservices Concepts



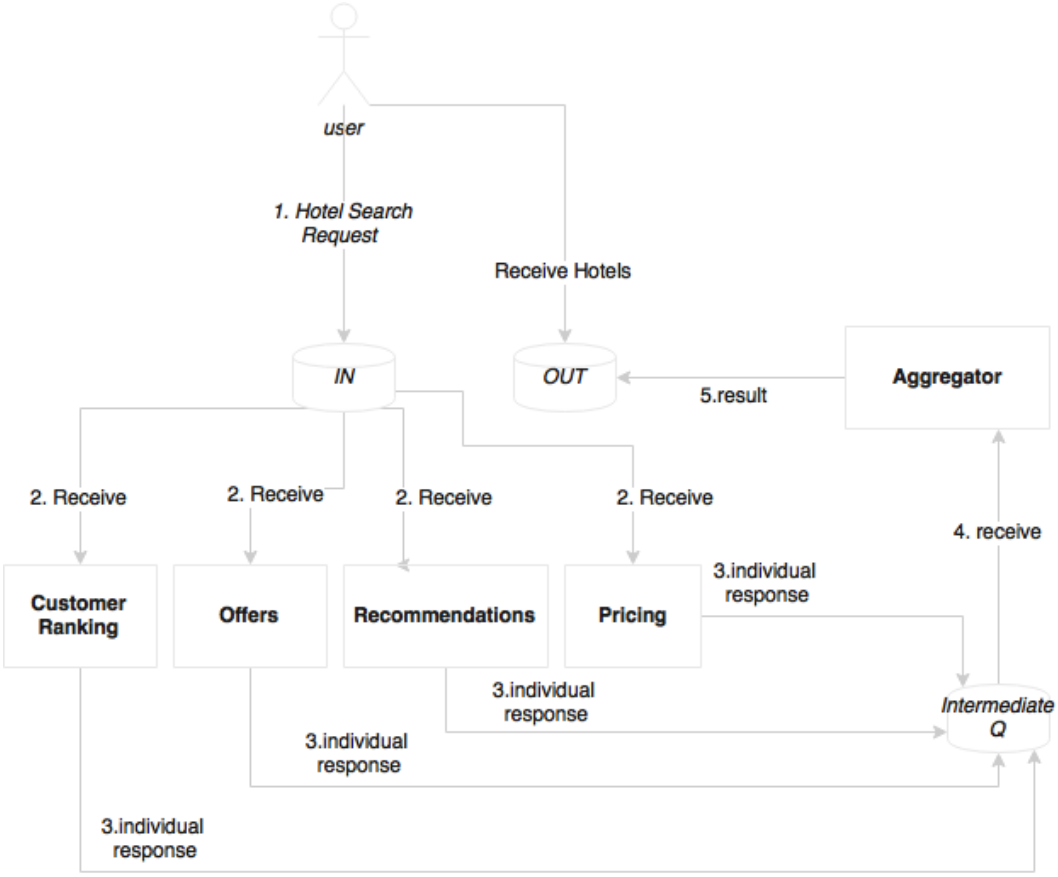


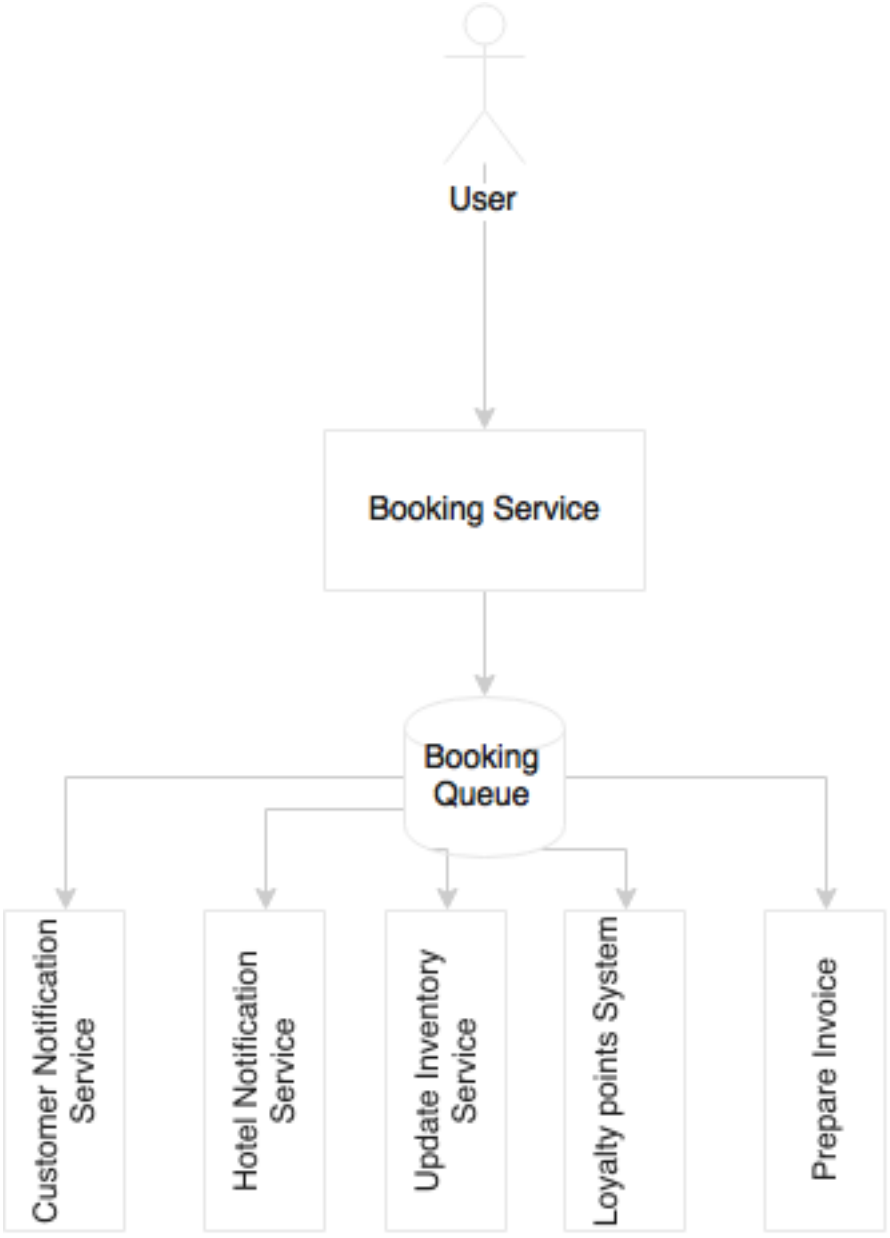


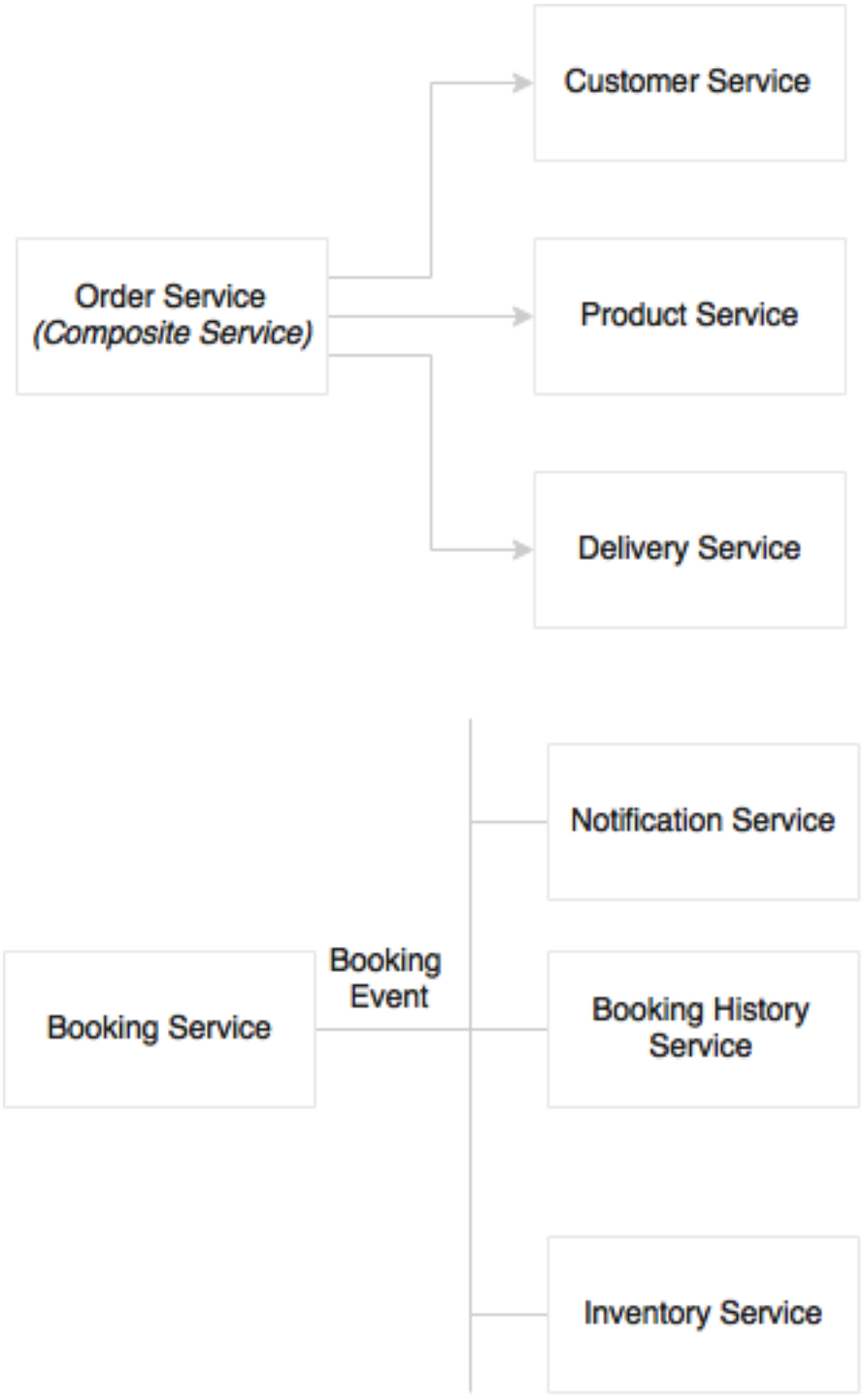
A) synchronous
request - response

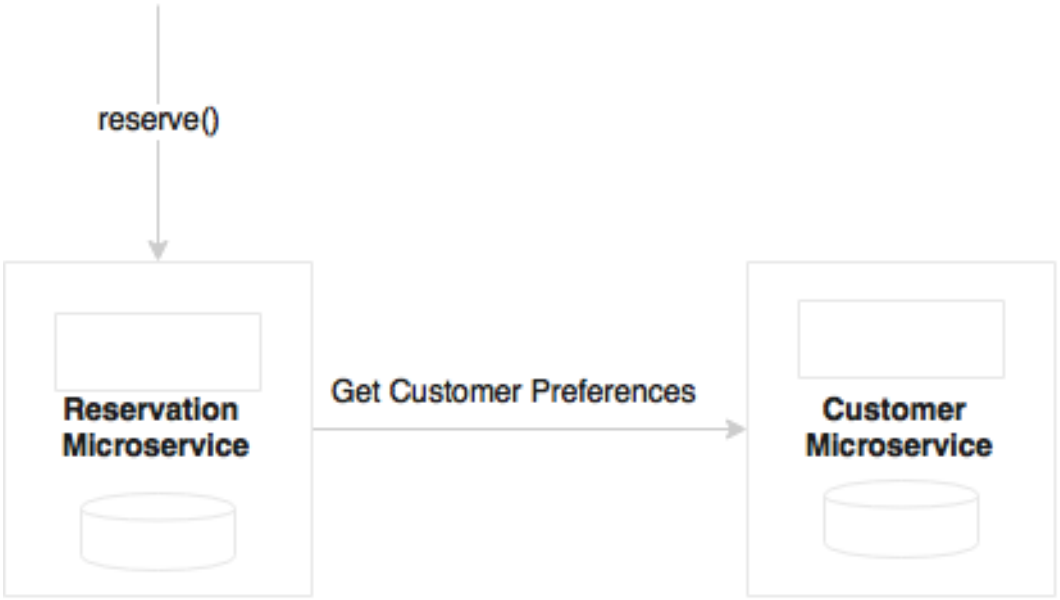


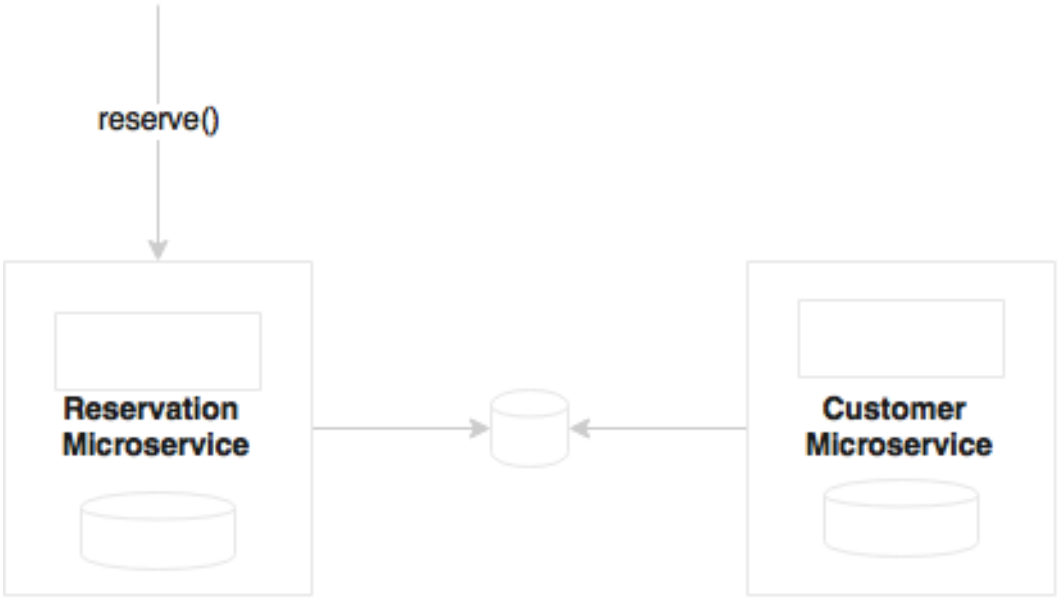
B) asynchronous
request - response

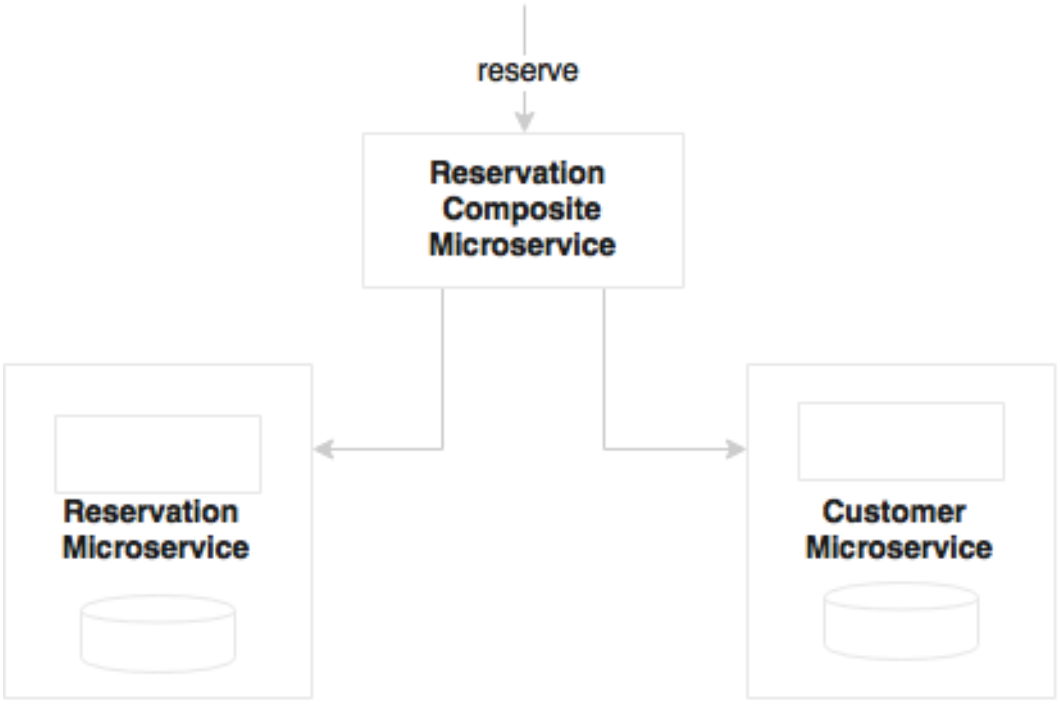

user
1. Hotel Search Request
Receive Hotels
IN
OUT
Aggregator
5.result
2. Receive
2. Receive
2. Receive
2. Receive
4. receive
Customer Ranking
Offers
Recommendations
Pricing
3.individual response
3.individual response
Intermediate Q
3.individual response
3.individual response

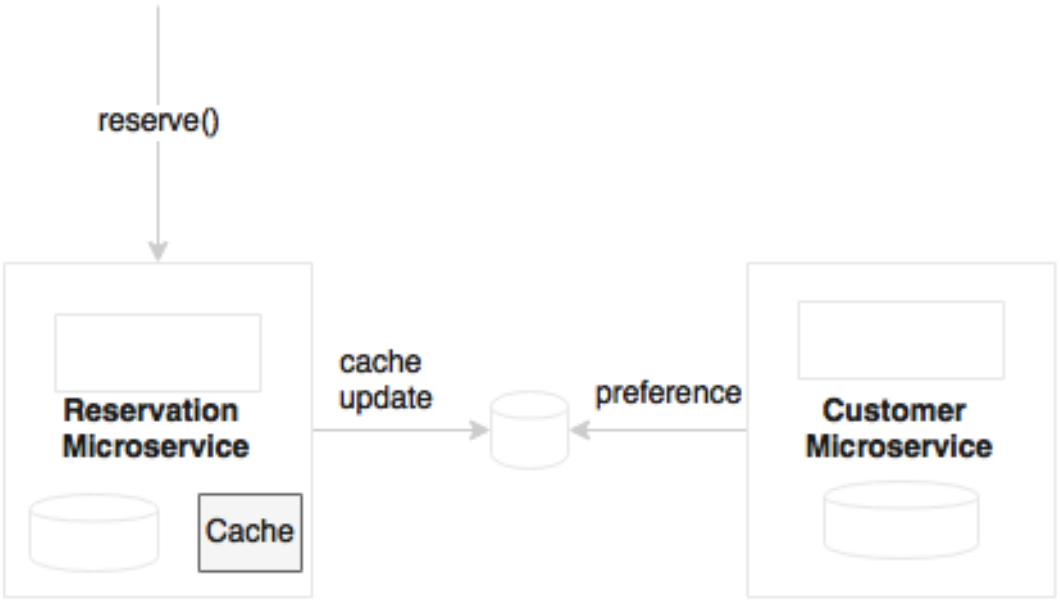

User
Booking Service
Booking Queue
Customer Notification Service
Hotel Notification Service
Update Inventory Service
Loyalty points System
Prepare Invoice

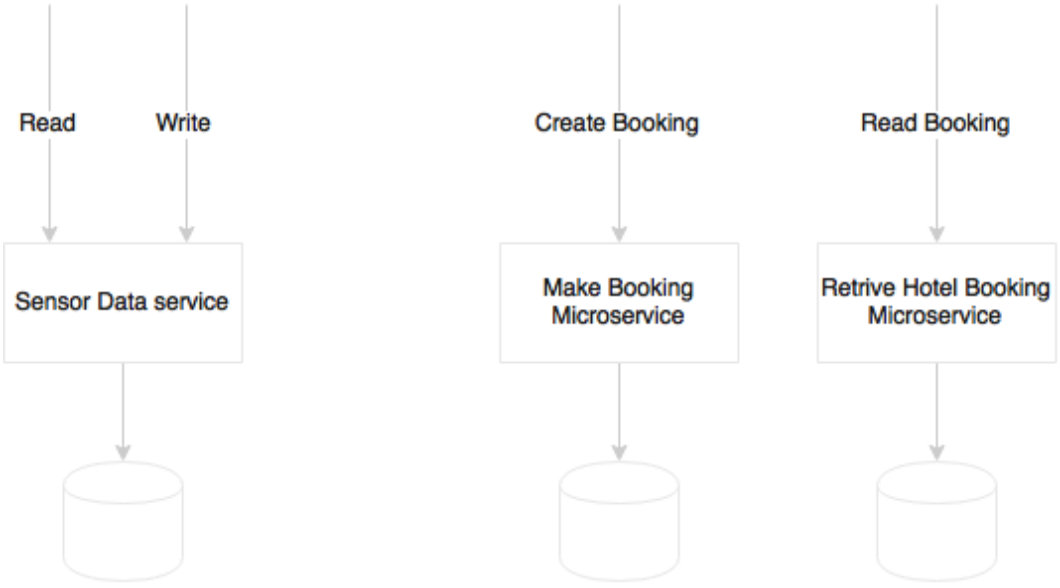

Customer Service
Order Service
(Composite Service)
Product Service
Delivery Service
Notification Service
Booking Service
Booking
Event
Booking History
Service
Inventory Service

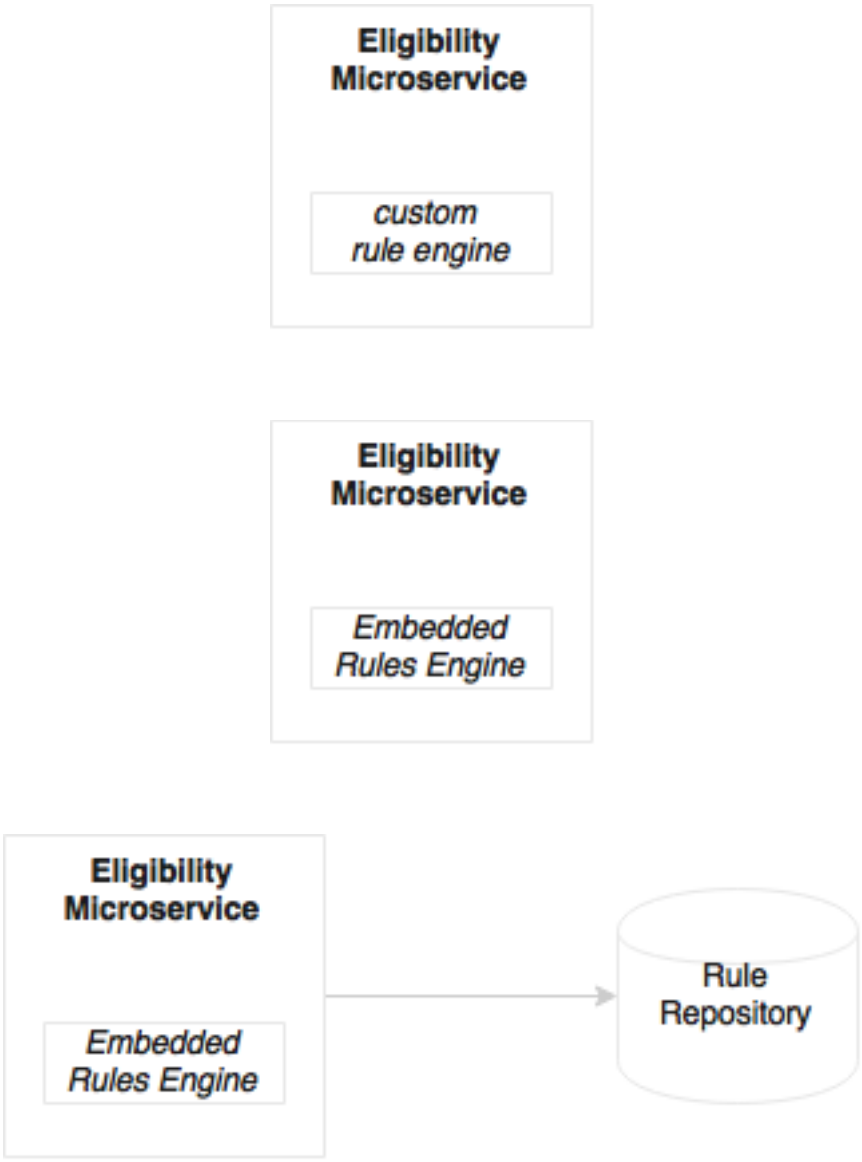

reserve()
Reservation Microservice
Get Customer Preferences
Customer Microservice

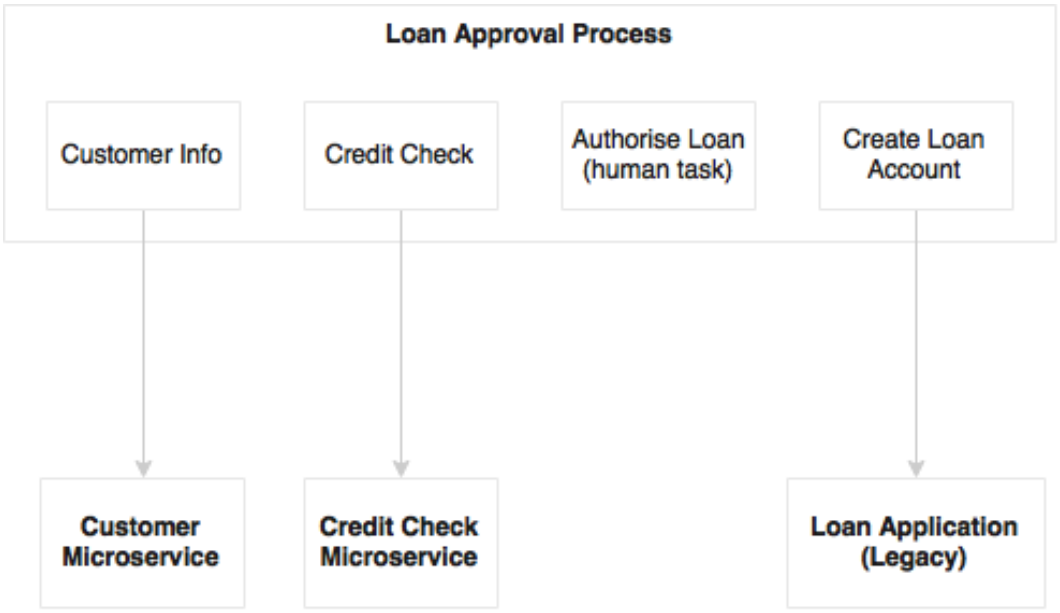

reserve()
Reservation Microservice
Customer Microservice

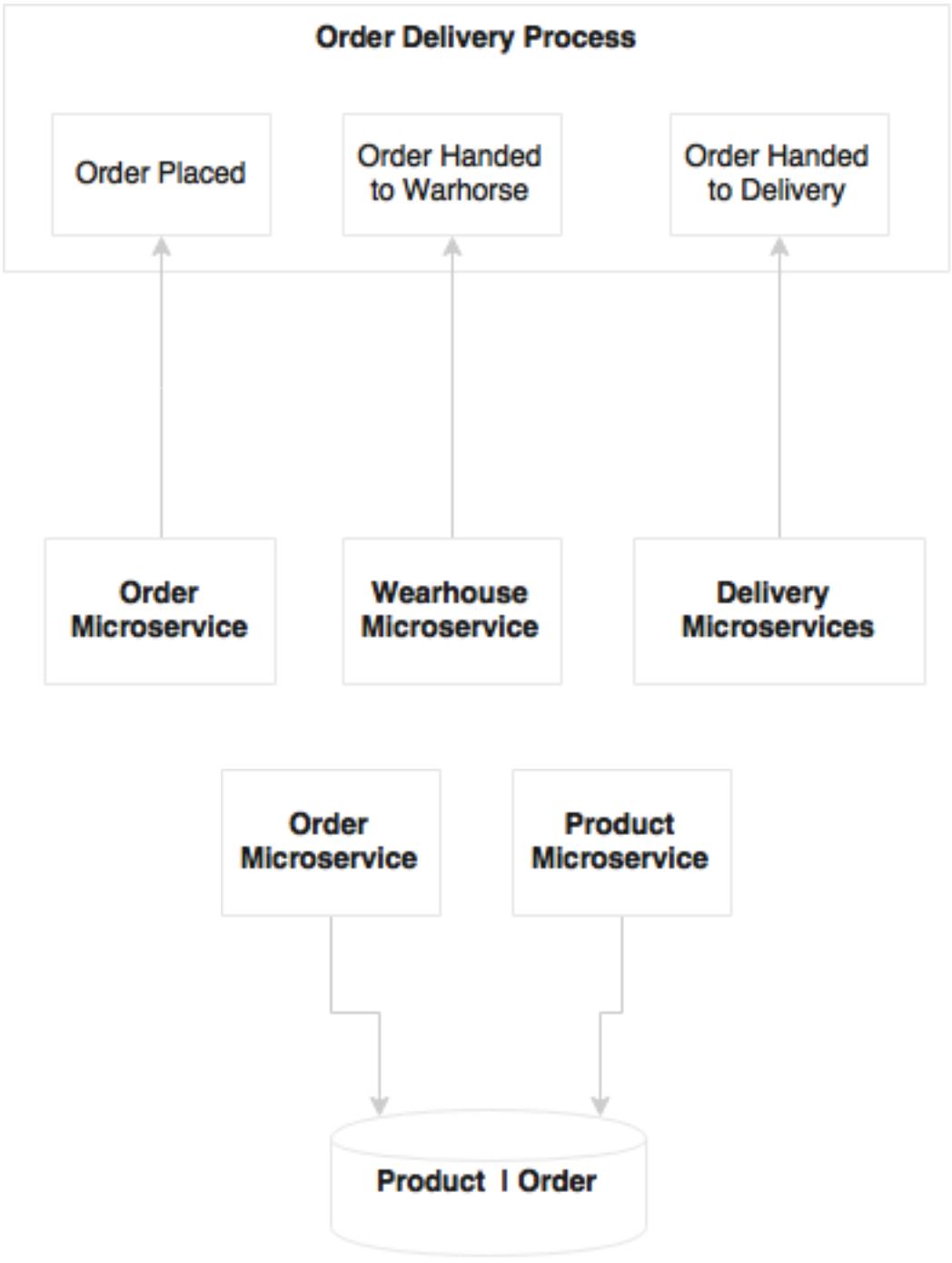

reserve
Reservation
Composite
Microservice
Reservation
Microservice
Customer
Microservice

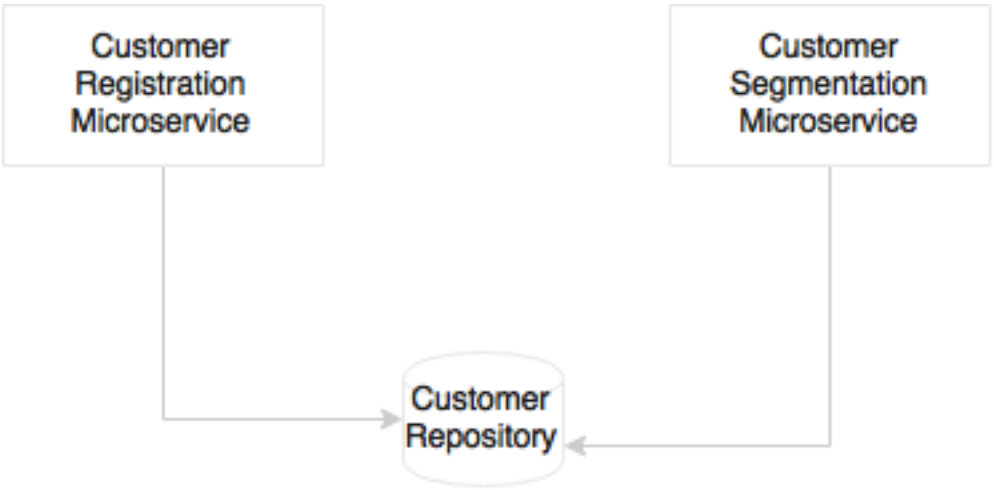

reserve()
Reservation Microservice
Cache
cache update
preference
Customer Microservice

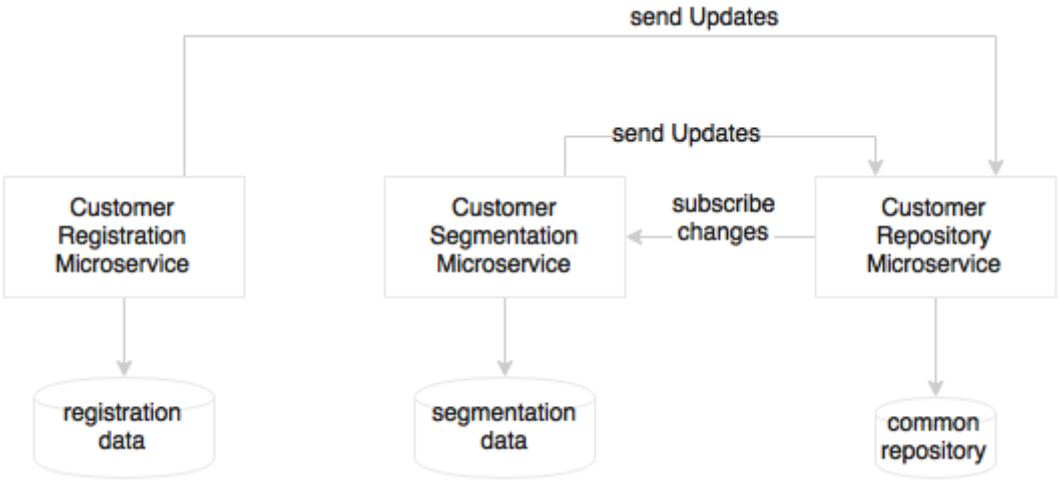

Read
Write
Sensor Data service
Create Booking
Make Booking Microservice
Read Booking
Retrive Hotel Booking Microservice

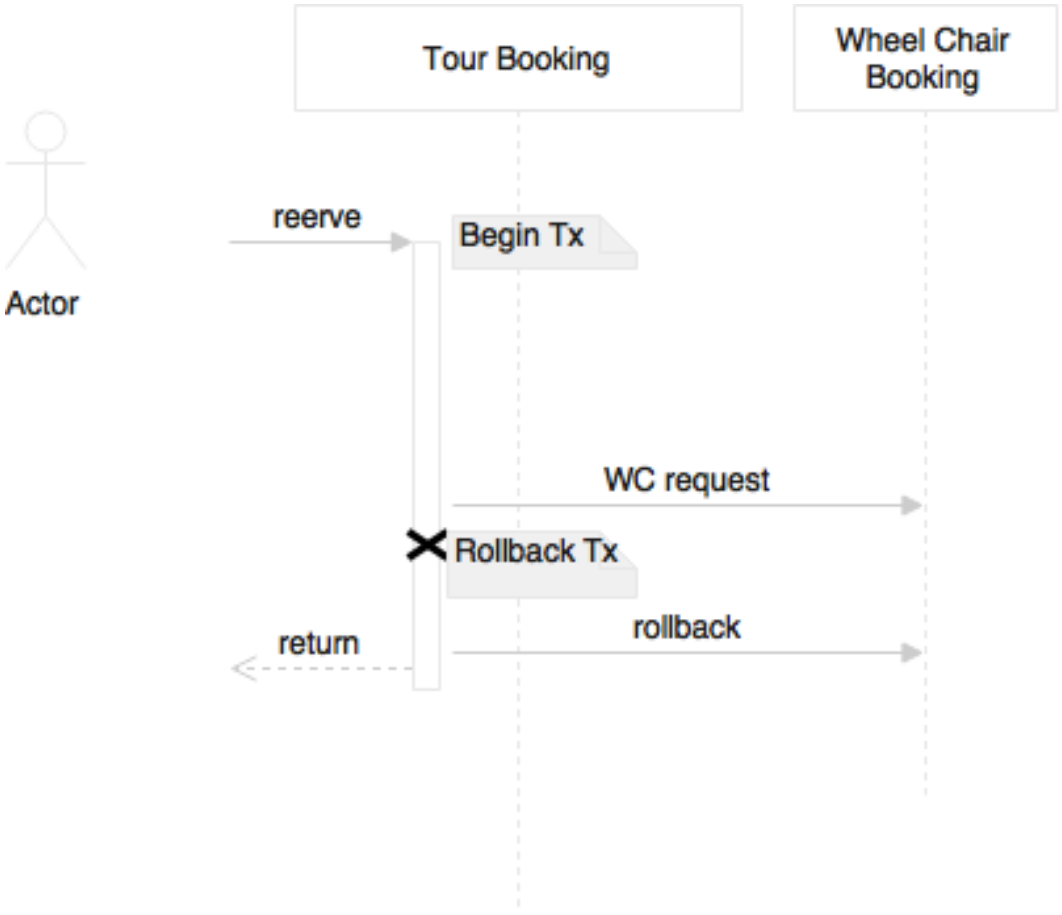

Eligibility Microservice
custom rule engine
Eligibility Microservice
Embedded Rules Engine
Eligibility Microservice
Embedded Rules Engine
Rule Repository


Loan Approval Process
Customer Info
Credit Check
Authorise Loan (human task)
Create Loan Account
Customer Microservice
Credit Check Microservice
Loan Application (Legacy)


Order Delivery Process
Order Placed
Order Handed to Warhorse
Order Handed to Delivery
Order Microservice
Wearhouse Microservice
Delivery Microservices
Order Microservice
Product Microservice
Product I Order


Customer
Registration
Microservice
Customer
Segmentation
Microservice
Customer
Repository


send Updates
send Updates
Customer
Registration
Microservice
Customer
Segmentation
Microservice
subscribe
changes
Customer
Repository
Microservice
registration
data
segmentation
data
common
repository

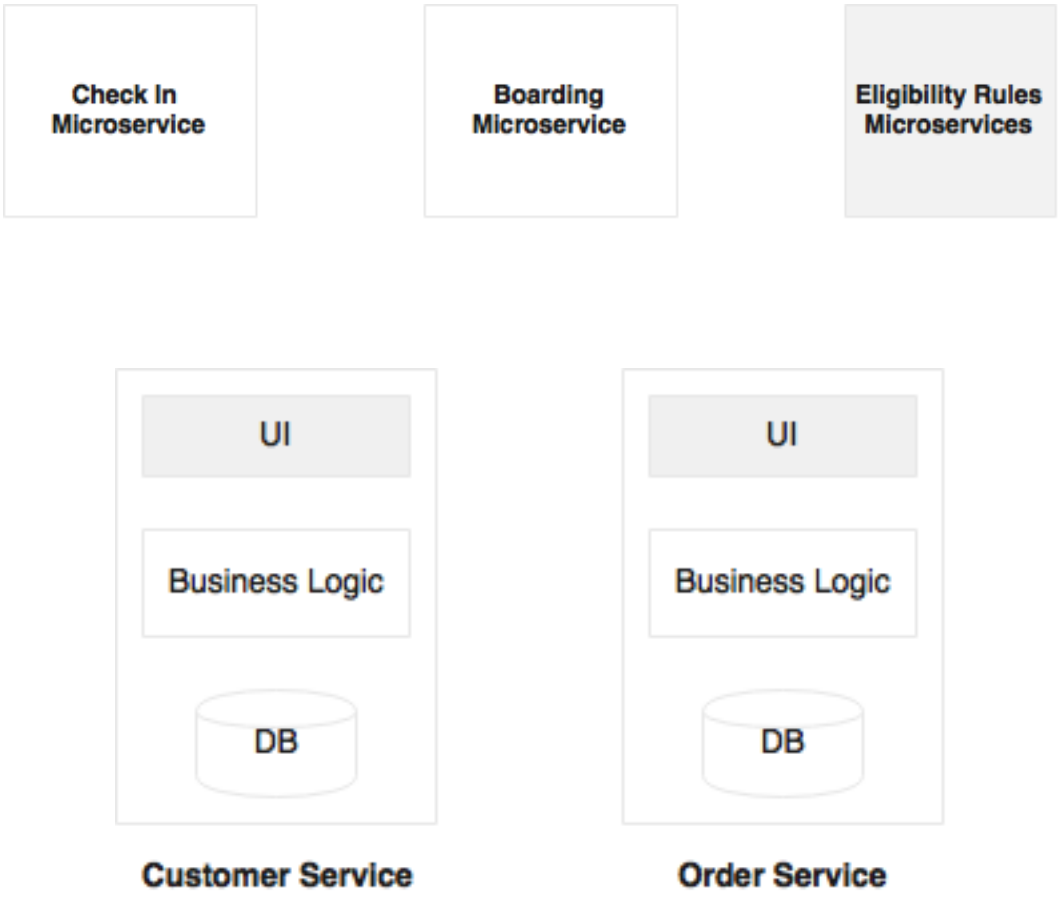


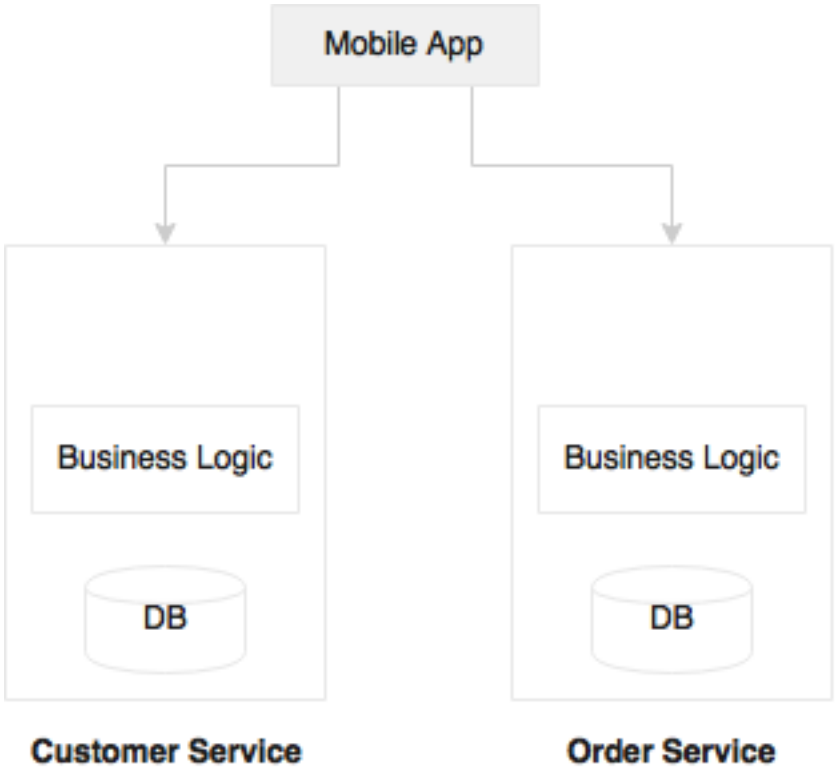
Eligibility Rules

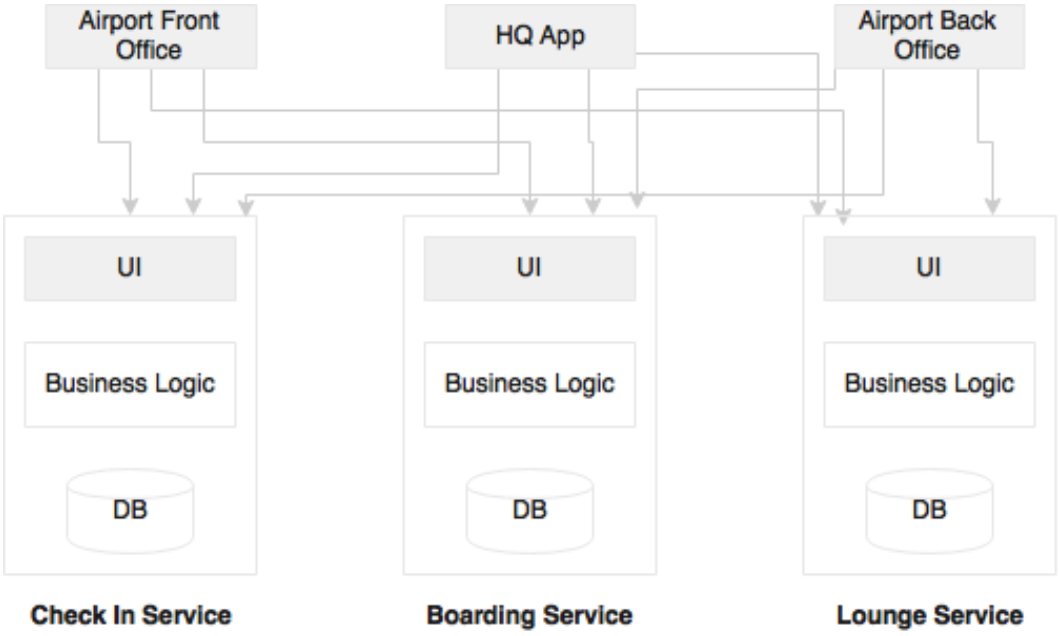
Check In Microservice

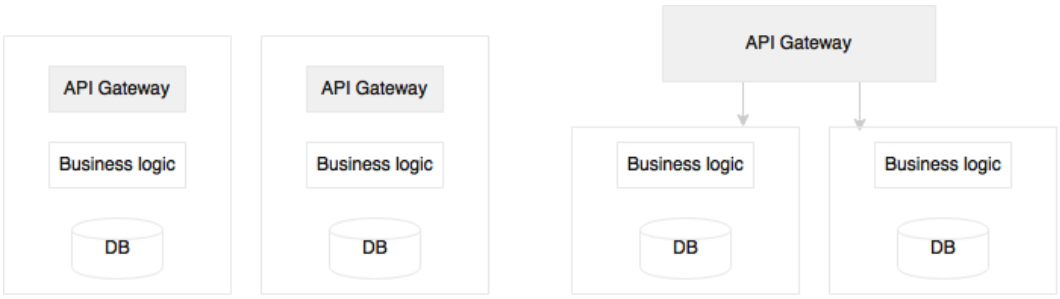
Eligibility Rules

Boarding Microservice


Check In Microservice
Boarding Microservice
Eligibility Rules Microservices
UI
Business Logic
DB
Customer Service
UI
Business Logic
DB
Order Service

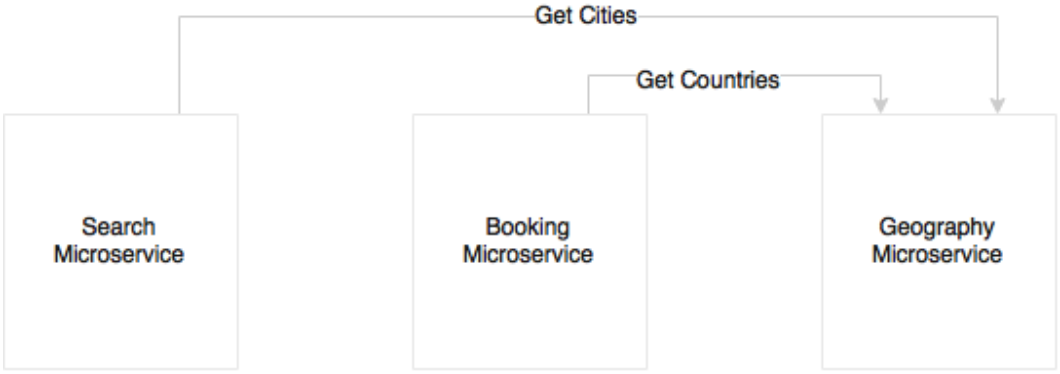

Mobile App
Business Logic
DB
Business Logic
DB
Customer Service
Order Service



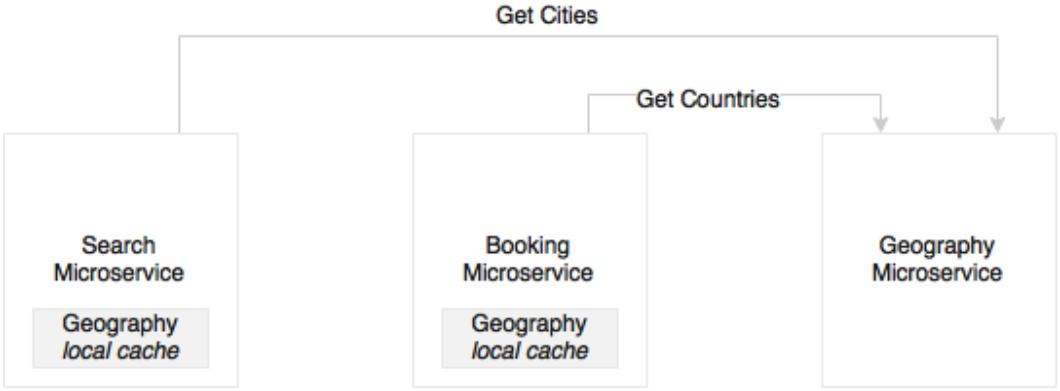


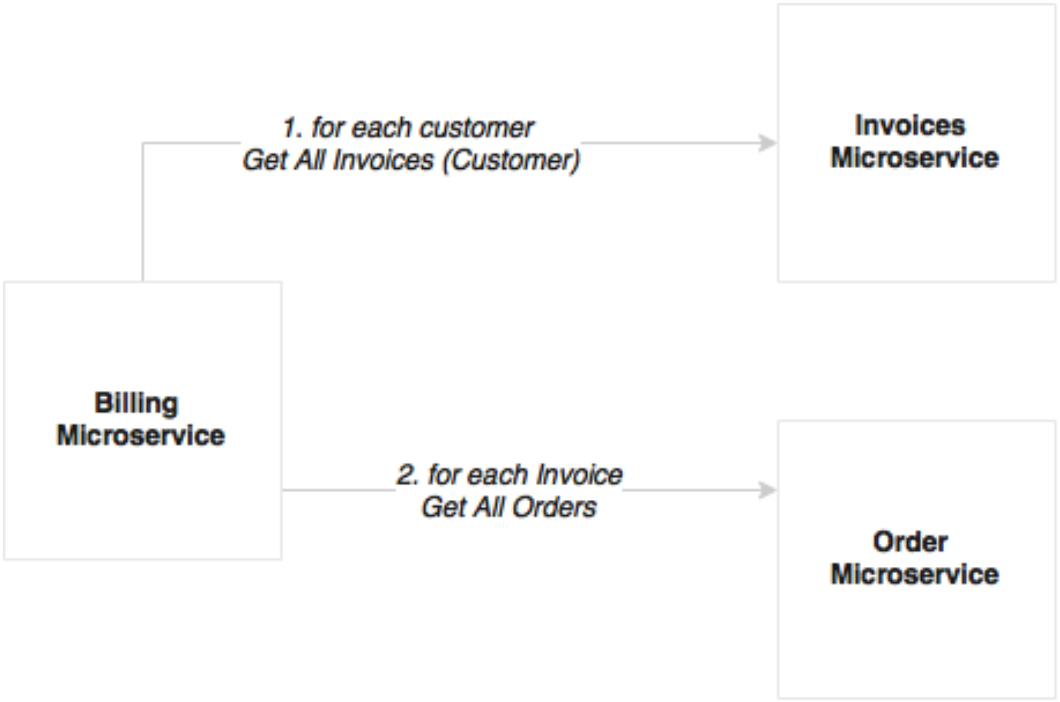
A) API gateway is part of the micro service

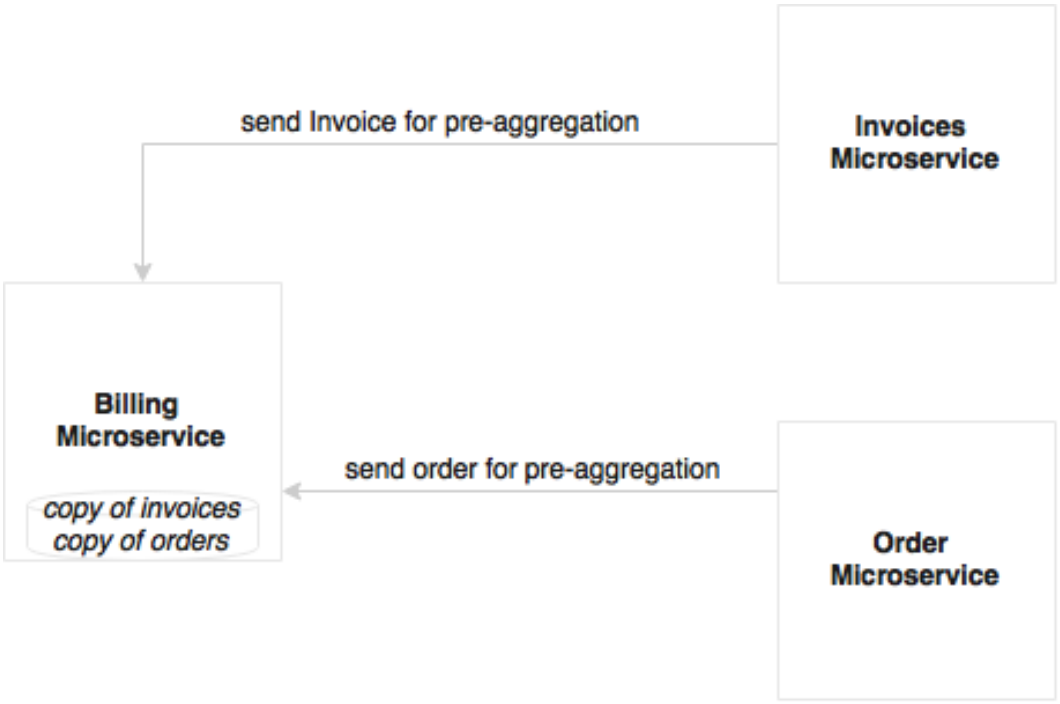
B) Common API gateway


Get Cities
Get Countries
Search Microservice
Booking Microservice
Geography Microservice

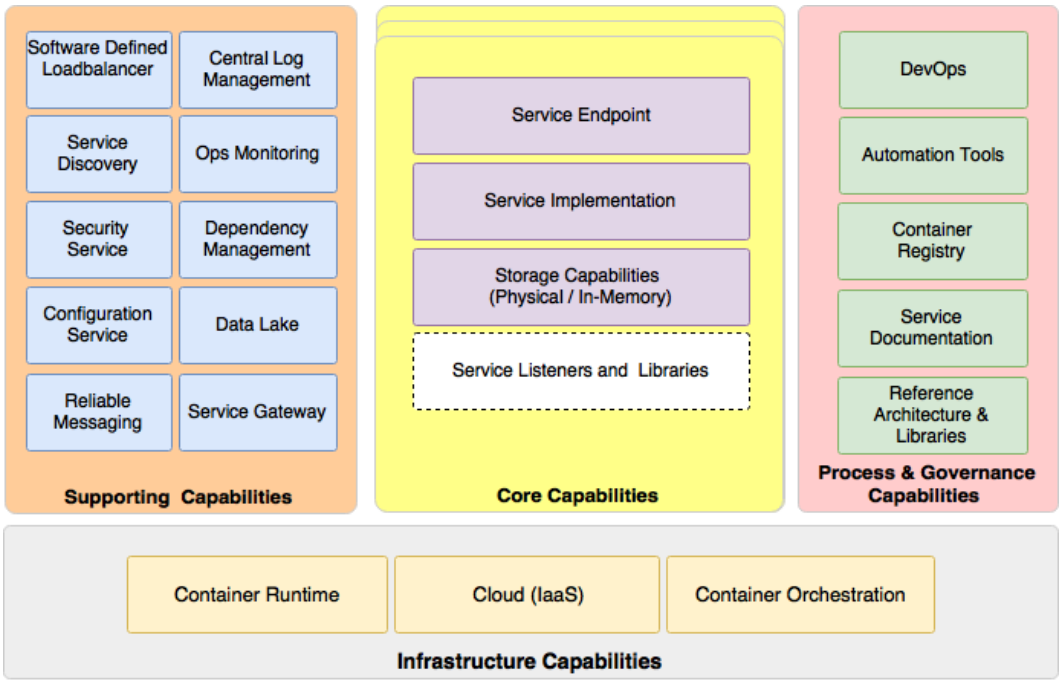
Search Microservice
Geography
Booking Microservice
Geography


Get Cities
Get Countries
Search Microservice
Geography local cache
Booking Microservice
Geography local cache
Geography Microservice

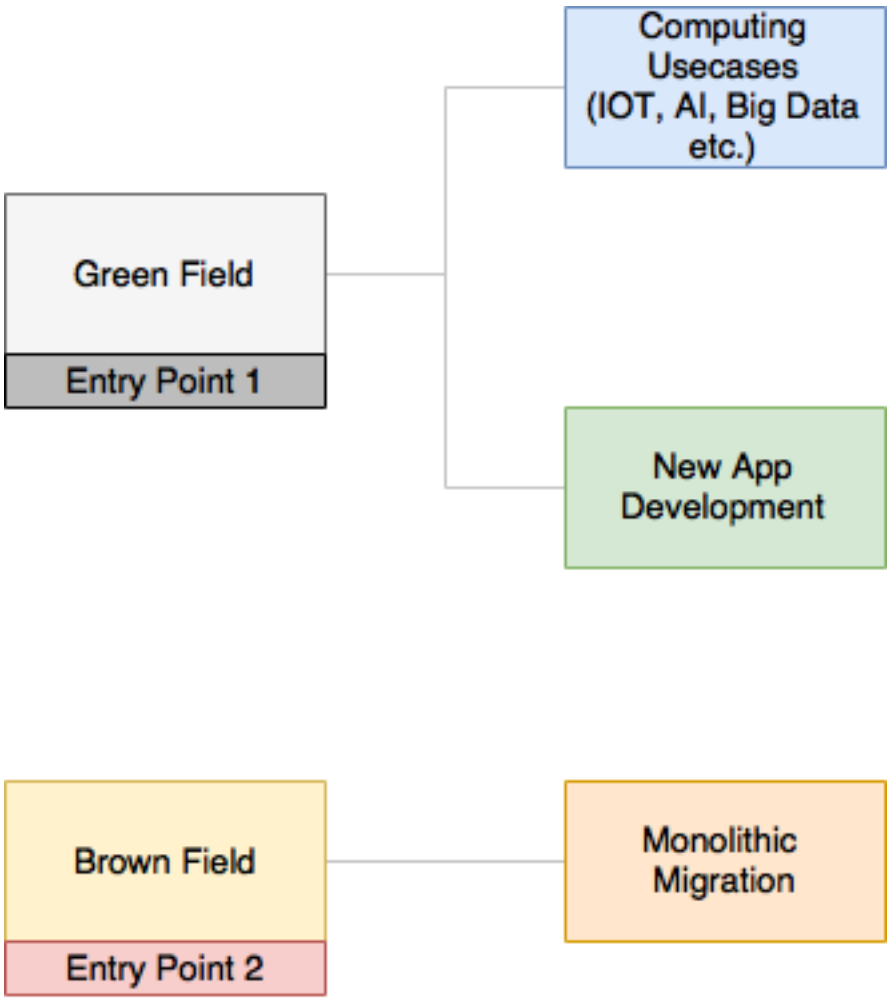

1. for each customer
Get All Invoices (Customer)
Invoices Microservice
Billing Microservice
2. for each Invoice
Get All Orders
Order Microservice


send Invoice for pre-aggregation
Invoices Microservice
Billing Microservice
copy of invoices
copy of orders
send order for pre-aggregation
Order Microservice

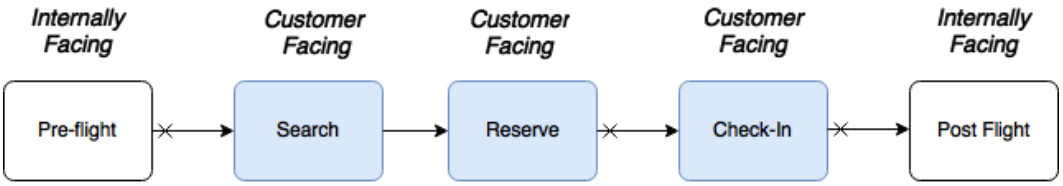
Chapter 5: Microservices Capability Model

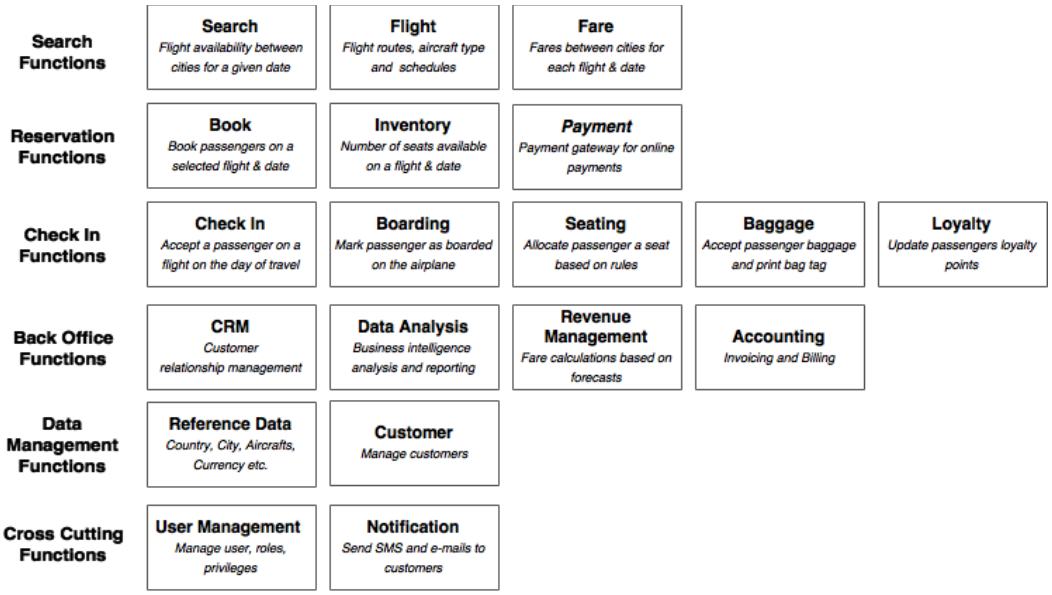


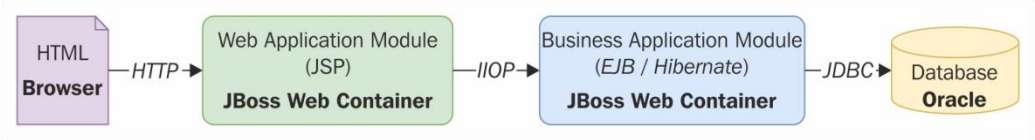
	Level 0 Traditional	Level 1 Basic	Level 2 Intermediate	Level 3 Advanced
Application	Monolithic	Service Oriented Integrations	Service Oriented Applications	API Centric
Database	One Size Fit All Enterprise DB	Enterprise DB + No SQLs and Light databases	Polyglot, DBaaS	Matured Data Lake / Near Realtime Analytics
Infrastructure	Physical Machines	Virtualization	Cloud	Containers
Monitoring	Infrastrucure	App & Infra Monitoring	APMs	APM & Central Log Management
Process	Waterfall	Agile and CI	CI & CD	DevOps

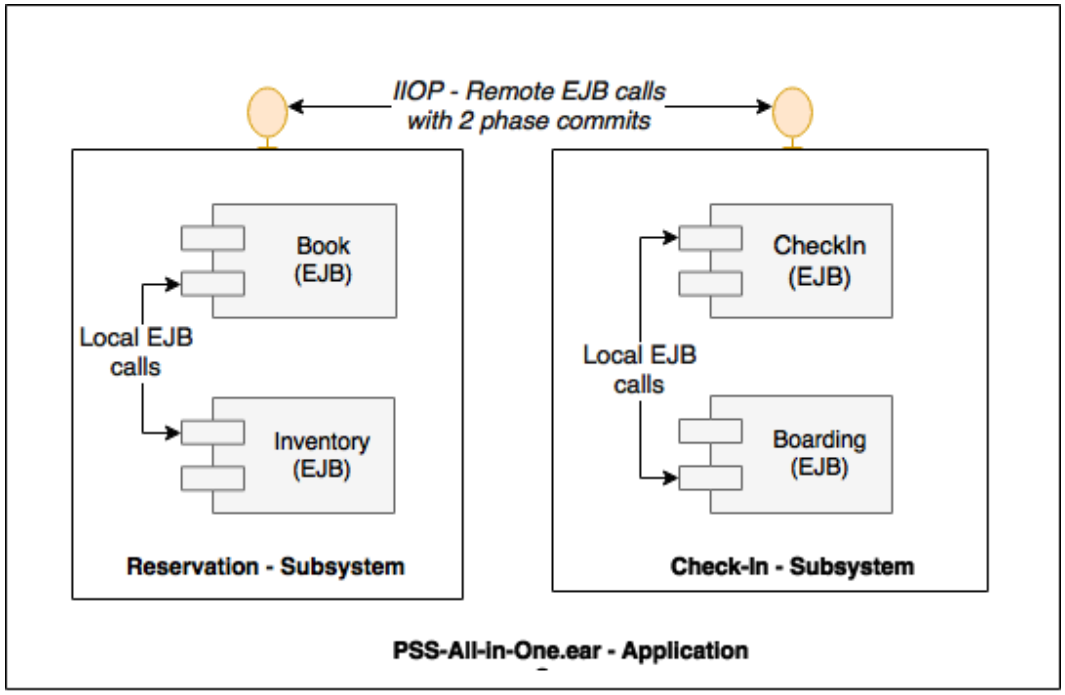

Computing Usecases (IOT, AI, Big Data etc.)
Green Field
Entry Point 1
New App Development
Brown Field
Entry Point 2
Monolithic Migration

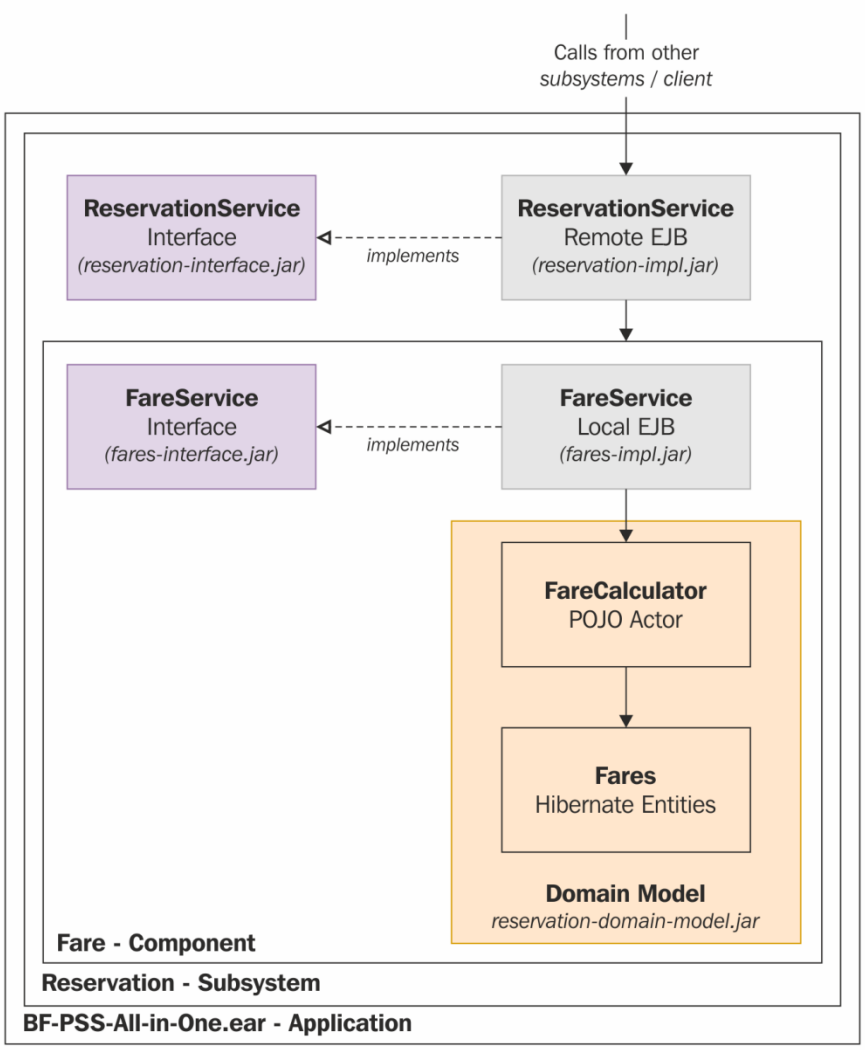
Chapter 6: Microservices Evolution – A Case Study

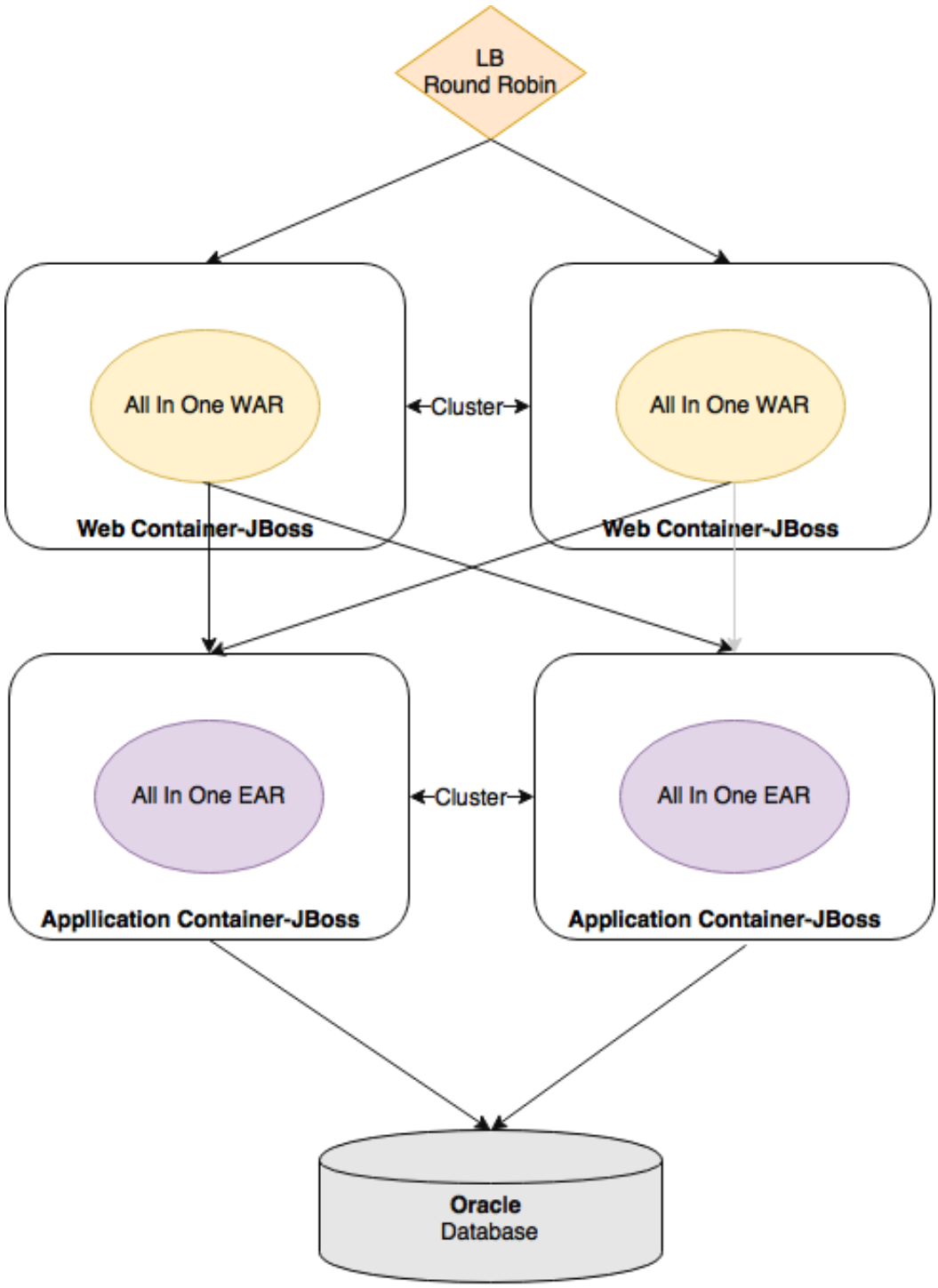


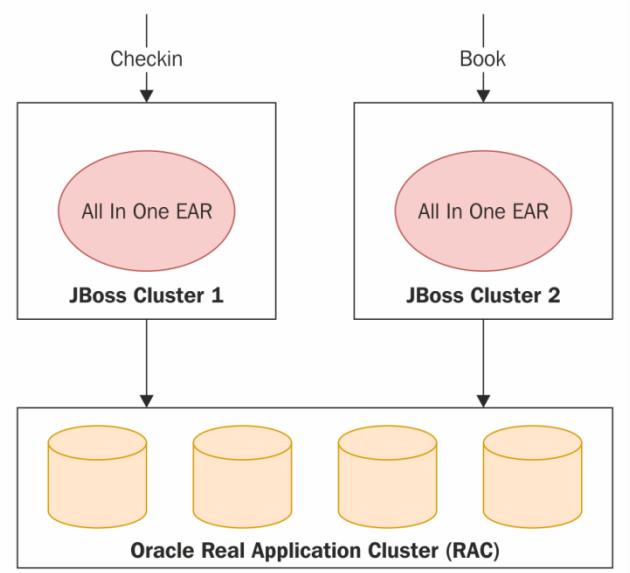


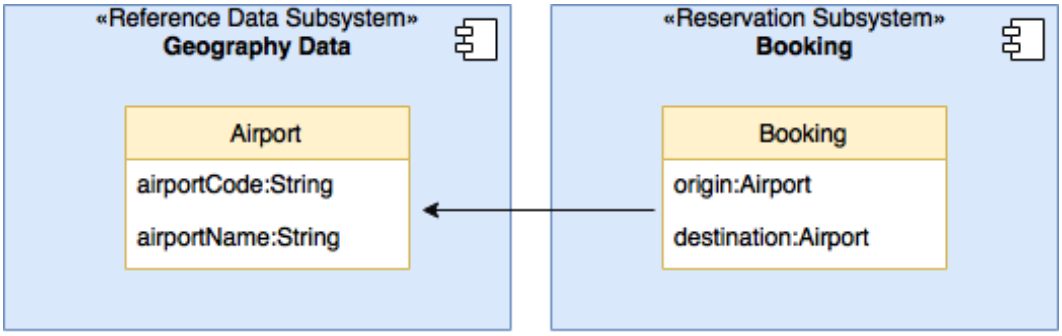


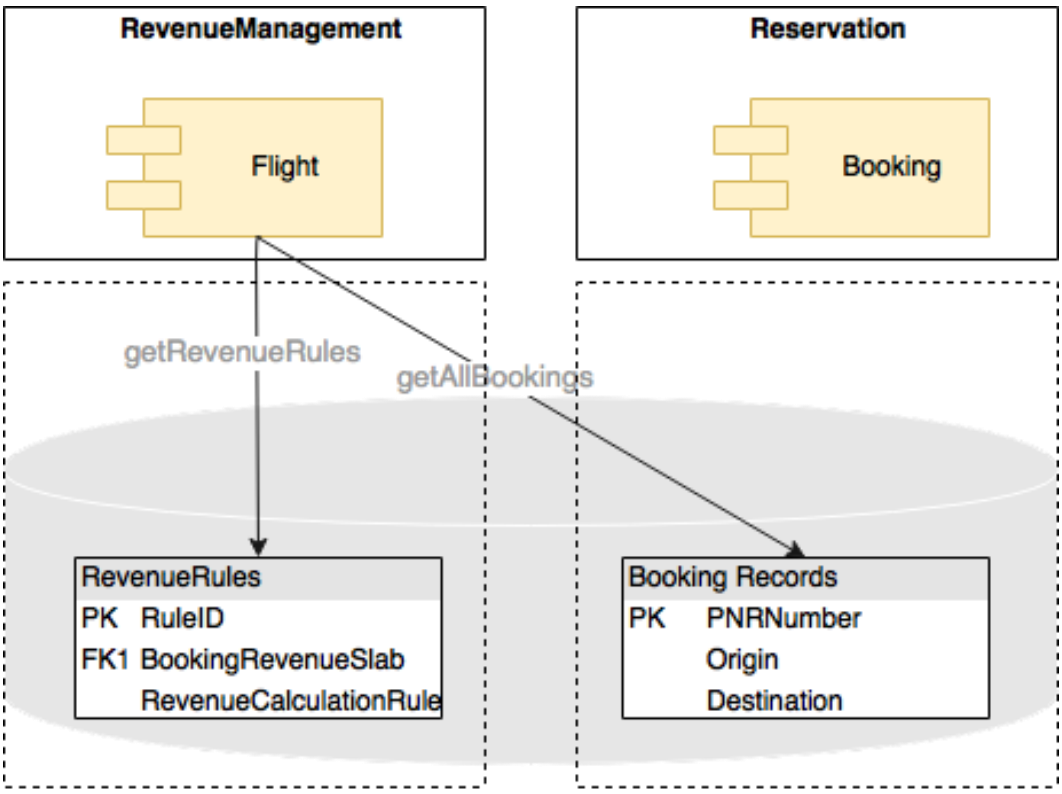

IIOP - Remote EJB calls
with 2 phase commits
Book
(EJB)
Local EJB
calls
Inventory
(EJB)
Reservation - Subsystem
CheckIn
(EJB)
Local EJB
calls
Boarding
(EJB)
Check-In - Subsystem
PSS-All-in-One.ear - Application

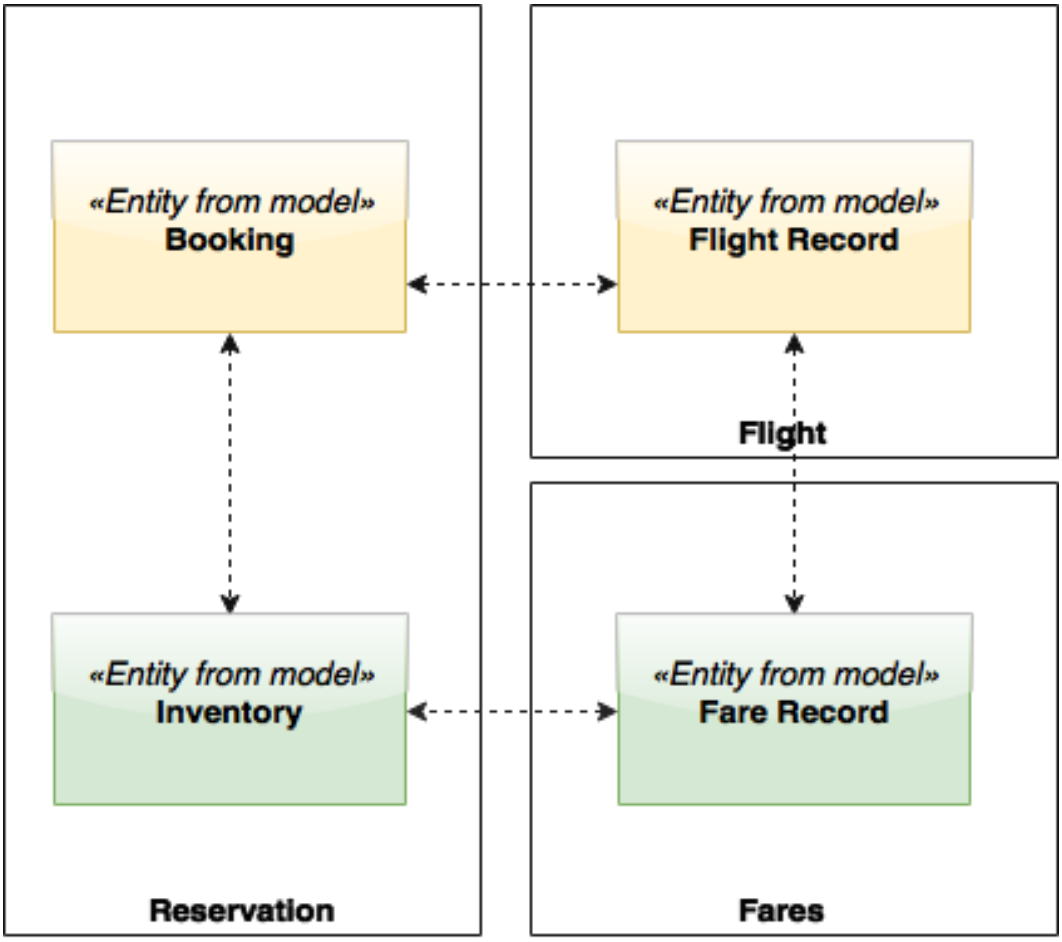

Calls from other
subsystems / client
ReservationService
Interface
(reservation-interface.jar)
implements
ReservationService
Remote EJB
(reservation-impl.jar)
FareService
Interface
(fares-interface.jar)
implements
FareService
Local EJB
(fares-impl.jar)
FareCalculator
POJO Actor
Fares
Hibernate Entities
Domain Model
reservation-domain-model.jar
Fare - Component
Reservation - Subsystem
BF-PSS-All-in-One.ear - Application

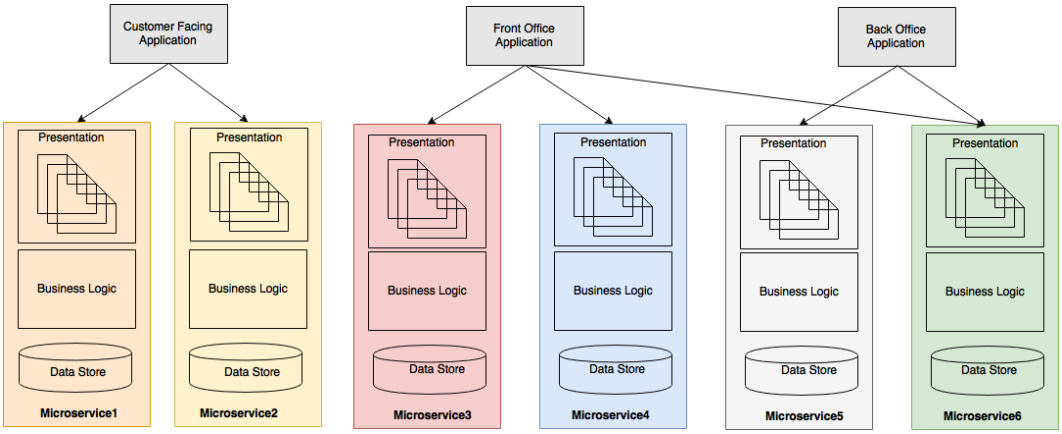

LB
Round Robin
All In One WAR
Cluster
All In One WAR
Web Container-JBoss
Web Container-JBoss
All In One EAR
Cluster
All In One EAR
Application Container-JBoss
Application Container-JBoss
Oracle
Database

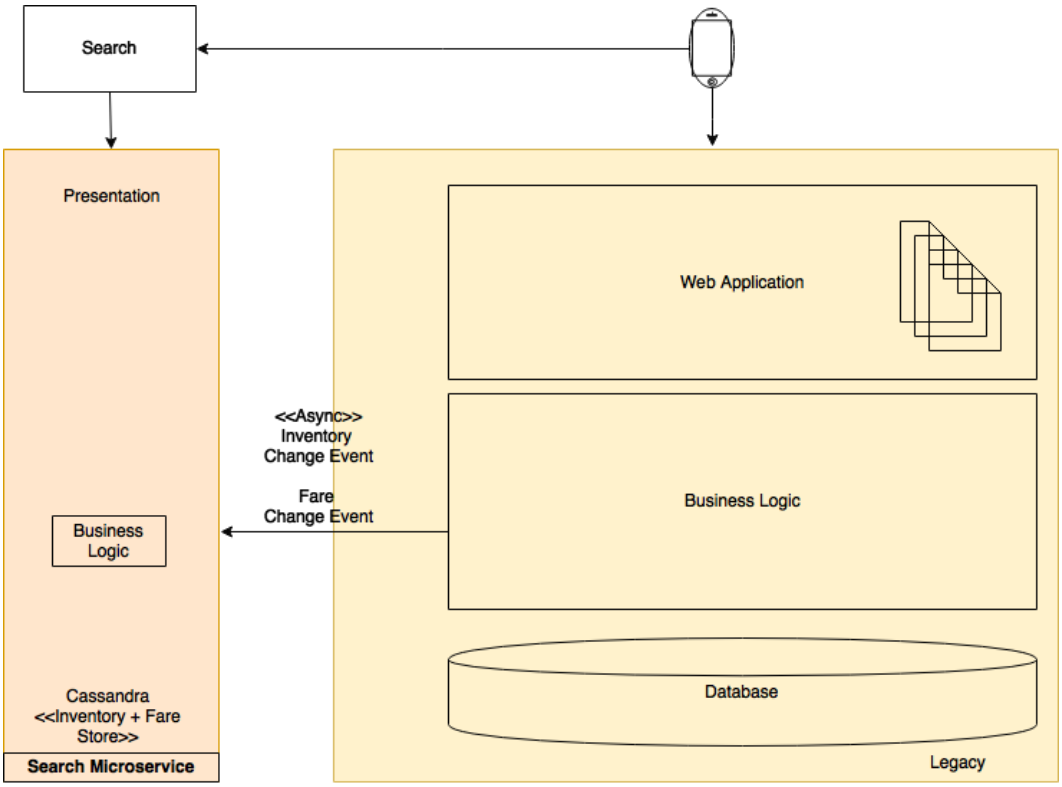

Checkin
Book
All In One EAR
All In One EAR
JBoss Cluster 1
JBoss Cluster 2
Oracle Real Application Cluster (RAC)

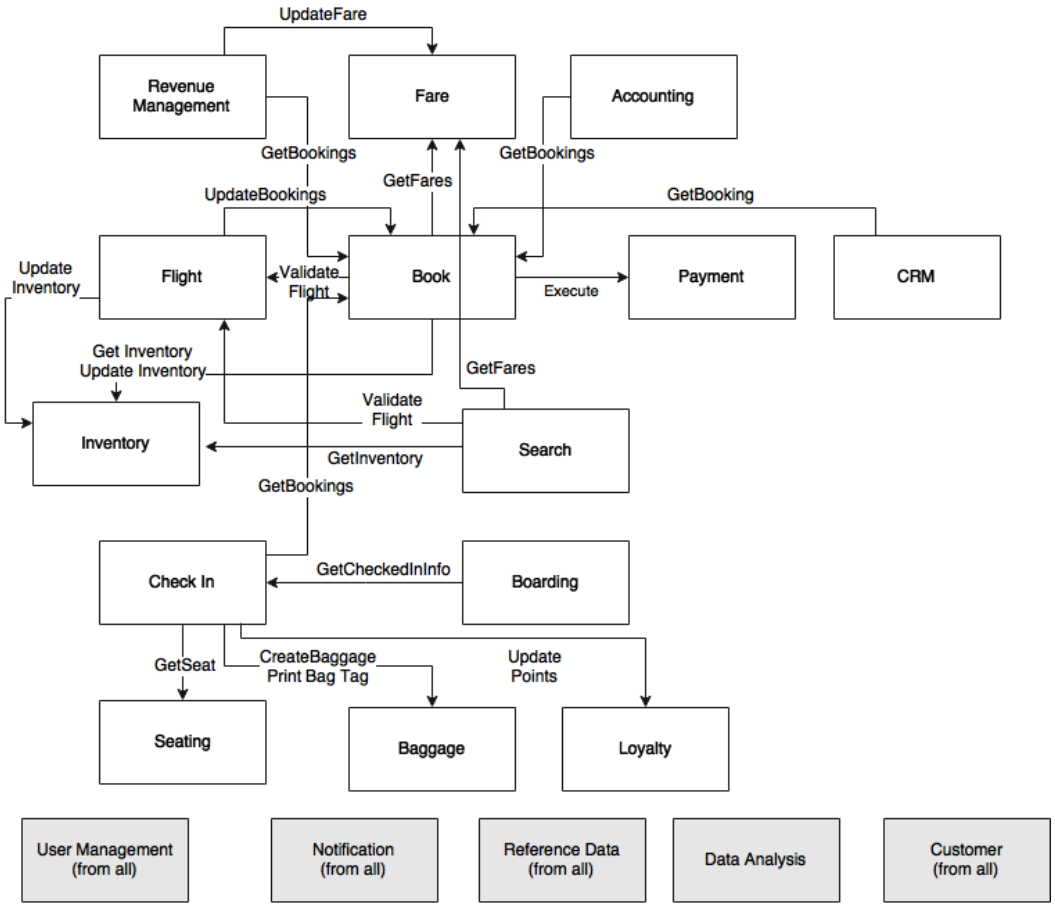

«Reference Data Subsystem»
Geography Data
Airport
airportCode:String
airportName:String
«Reservation Subsystem»
Booking
Booking
origin:Airport
destination:Airport

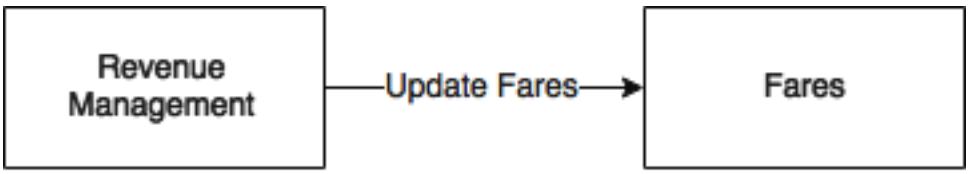

RevenueManagement
Flight
Reservation
Booking
getRevenueRules
getAllBookings
RevenueRules
PK RuleID
FK1 BookingRevenueSlab
RevenueCalculationRule
Booking Records
PK PNRNumber
Origin
Destination

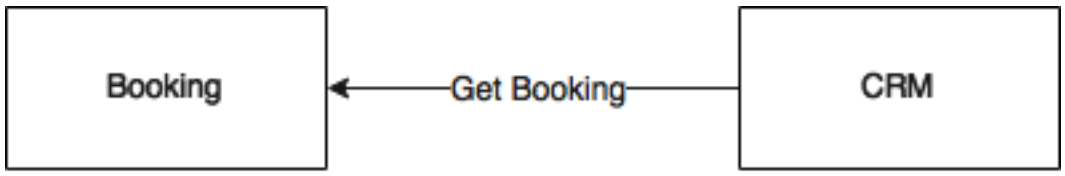

«Entity from model»
Booking
«Entity from model»
Flight Record
Flight
«Entity from model»
Inventory
«Entity from model»
Fare Record
Reservation
Fares

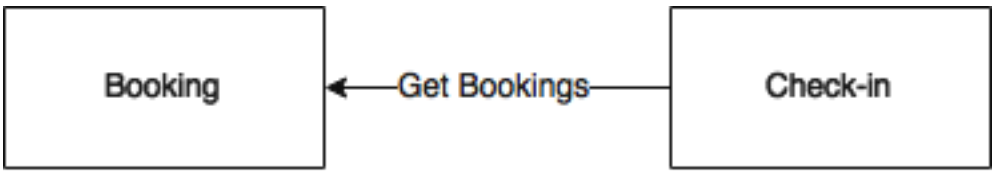

Customer Facing Application
Front Office Application
Back Office Application
Presentation
Business Logic
Data Store
Microservice1
Microservice2
Microservice3
Microservice4
Microservice5
Microservice6

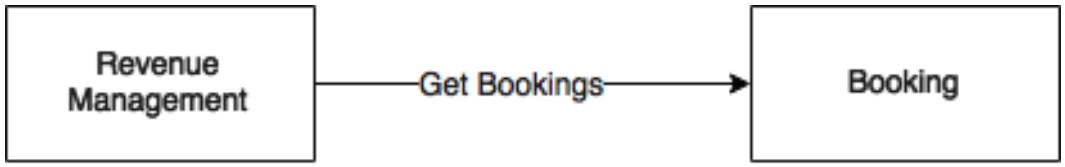

Search
Presentation
Business Logic
Cassandra
<<Inventory + Fare Store>>
Search Microservice
<<Async>>
Inventory
Change Event
Fare
Change Event
Web Application
Business Logic
Database
Legacy

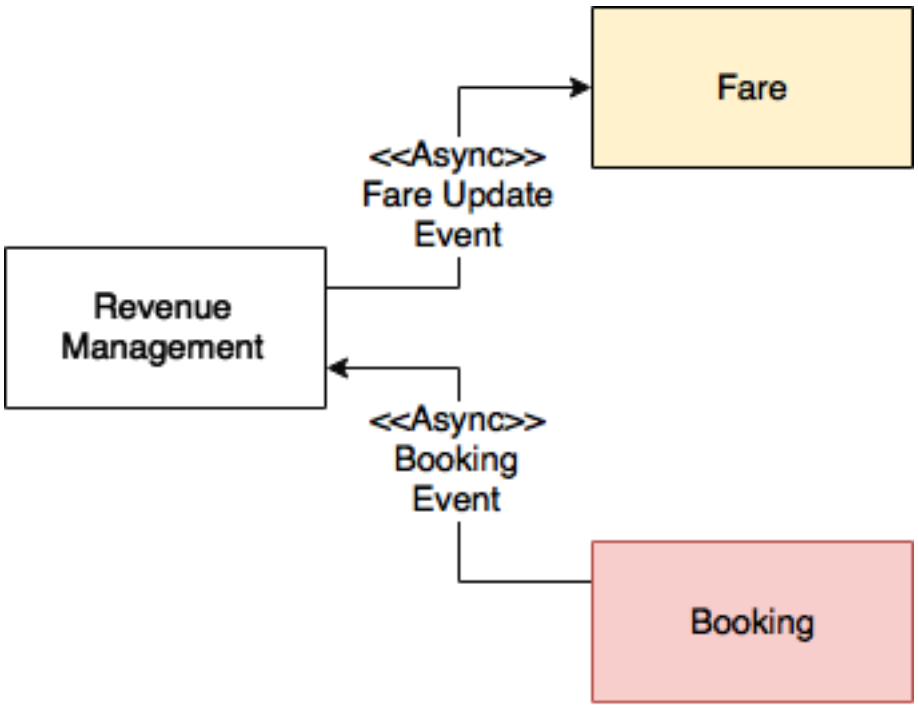

UpdateFare
Revenue Management
Fare
Accounting
GetBookings
GetFares
GetBookings
UpdateBookings
GetBooking
Update Inventory
Flight
Validate Flight
Book
Execute
Payment
CRM
Get Inventory
Update Inventory
GetFares
Validate Flight
Inventory
Search
GetInventory
GetBookings
Check In
GetCheckedInInfo
Boarding
GetSeat
CreateBaggage
Print Bag Tag
Update Points
Seating
Baggage
Loyalty
User Management (from all)
Notification (from all)
Reference Data (from all)
Data Analysis
Customer (from all)

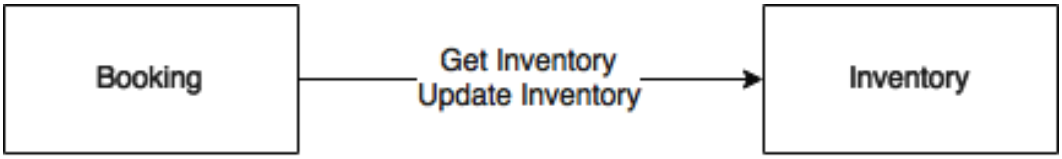

Revenue Management
Update Fares
Fares

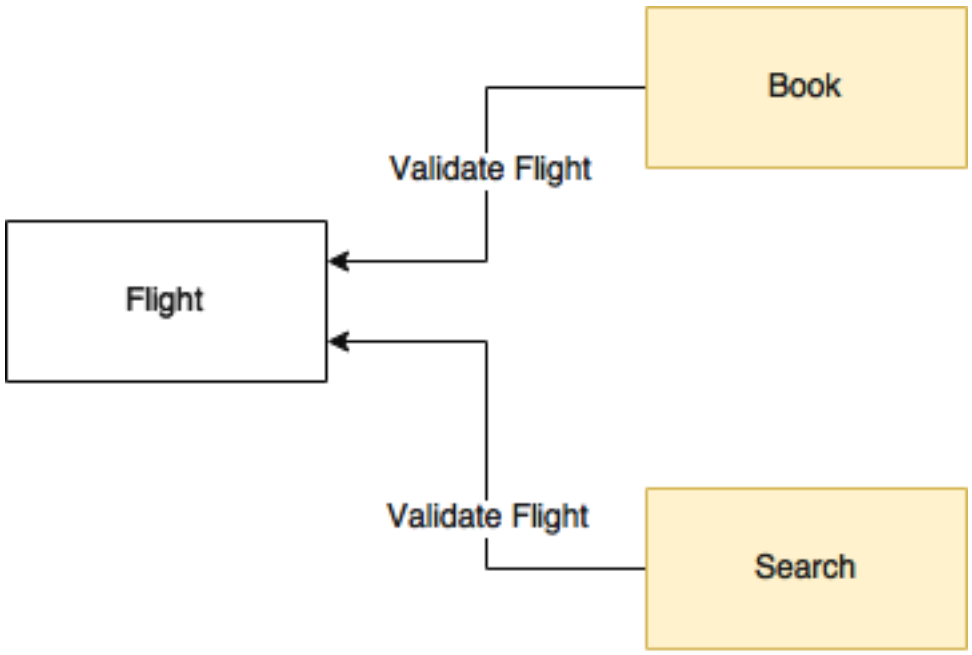

Booking
Get Booking
CRM

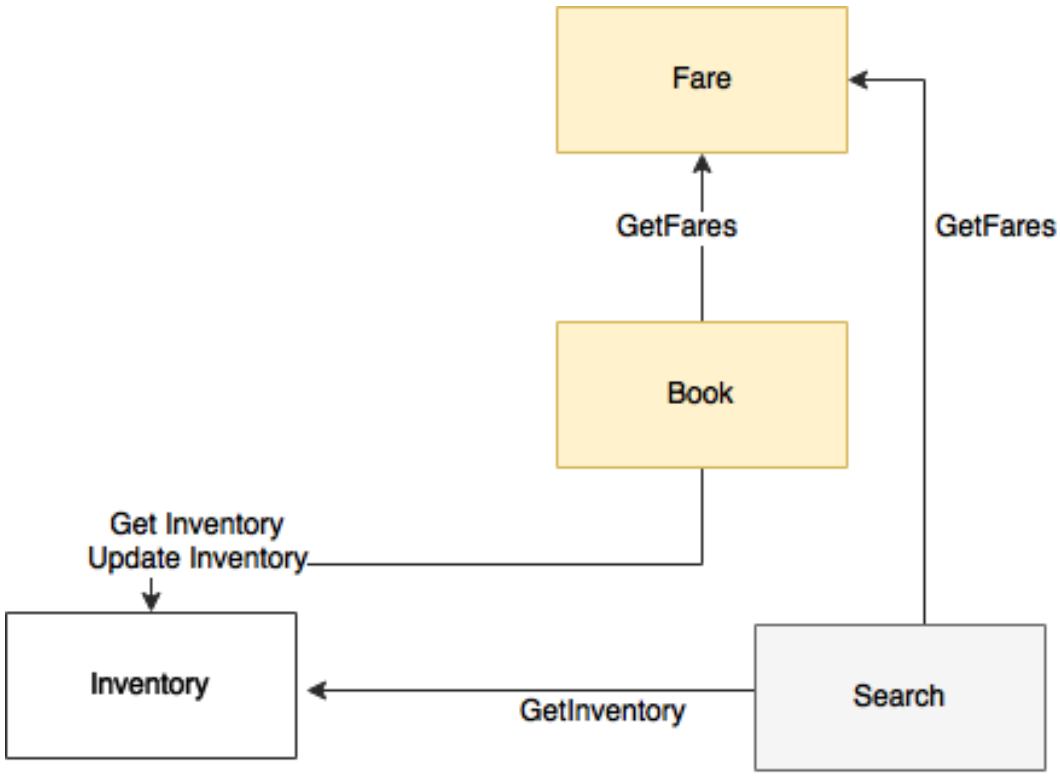

Booking
Get Bookings
Check-in

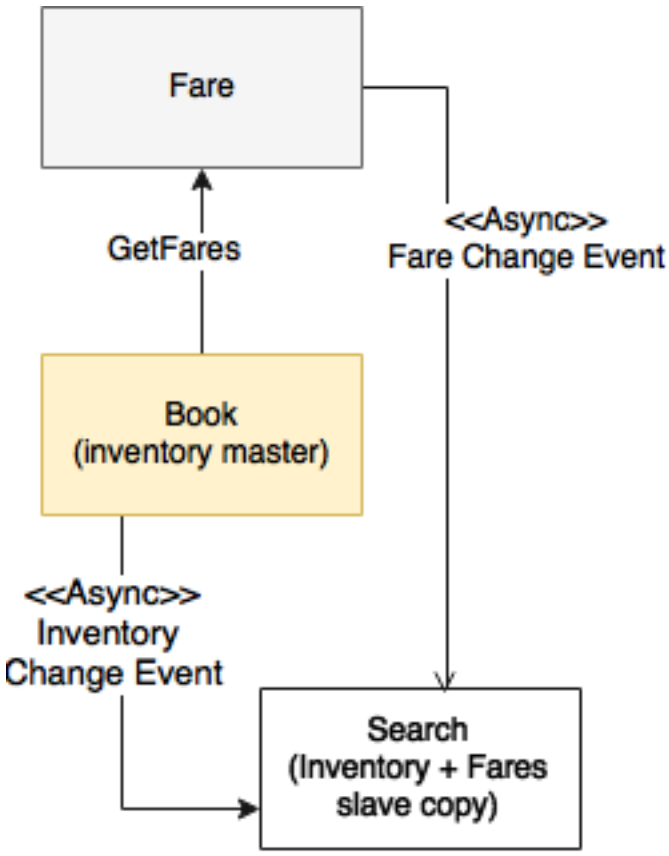

Revenue Management
Get Bookings
Booking

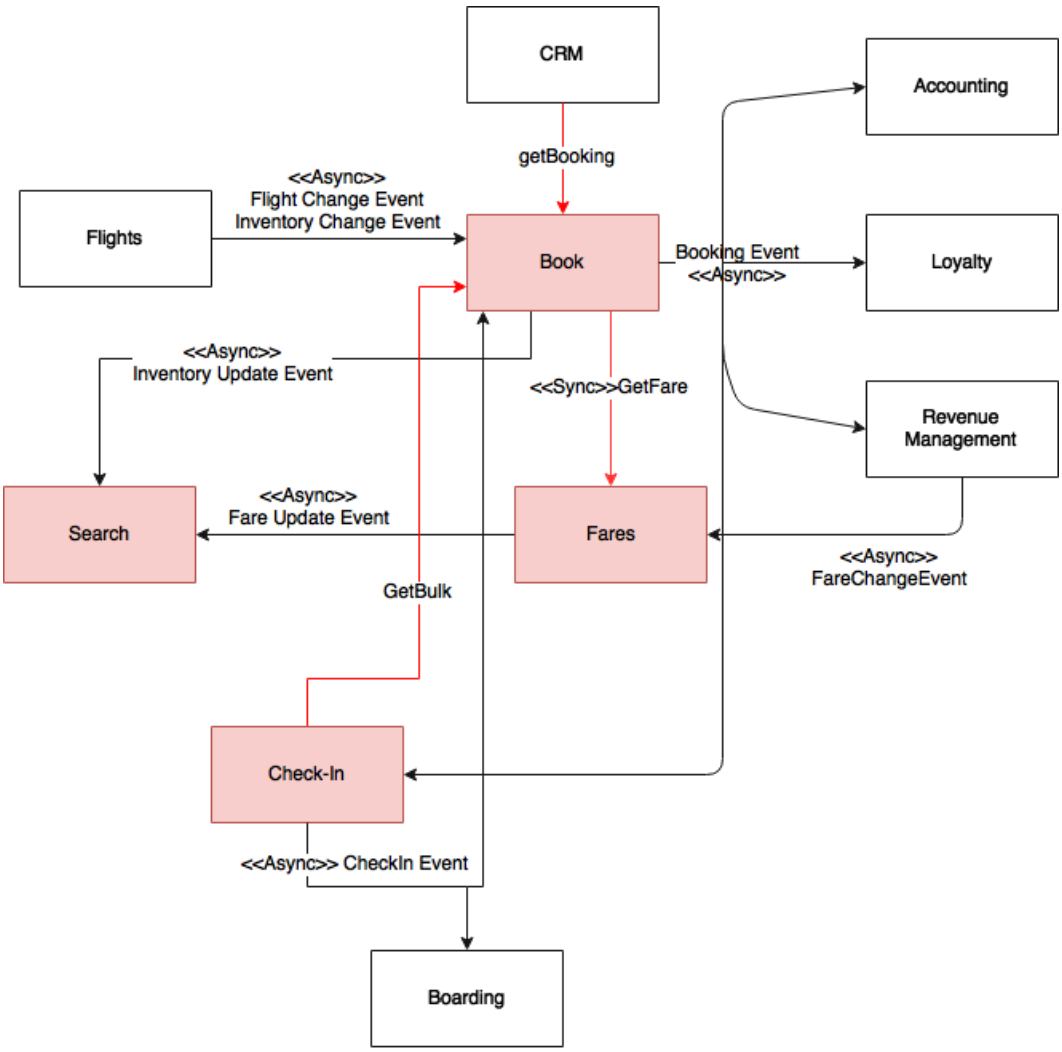

Fare
<<Async>>
Fare Update
Event
Revenue
Management
<<Async>>
Booking
Event
Booking

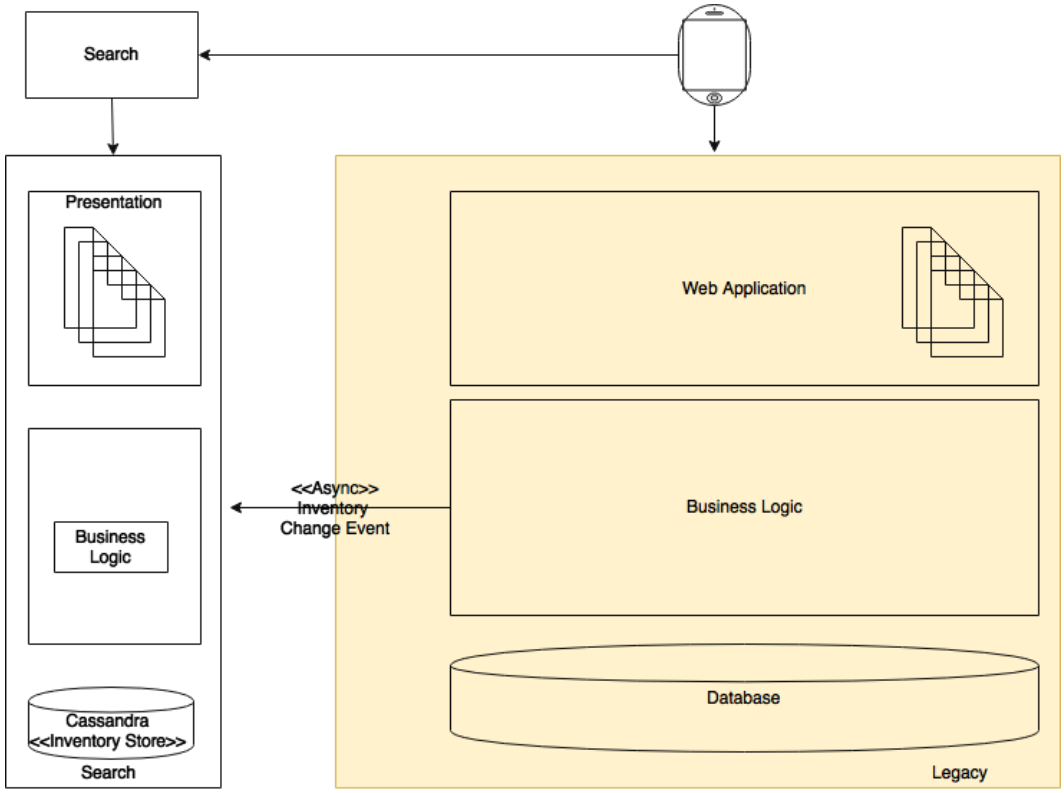

Booking
Get Inventory
Update Inventory
Inventory

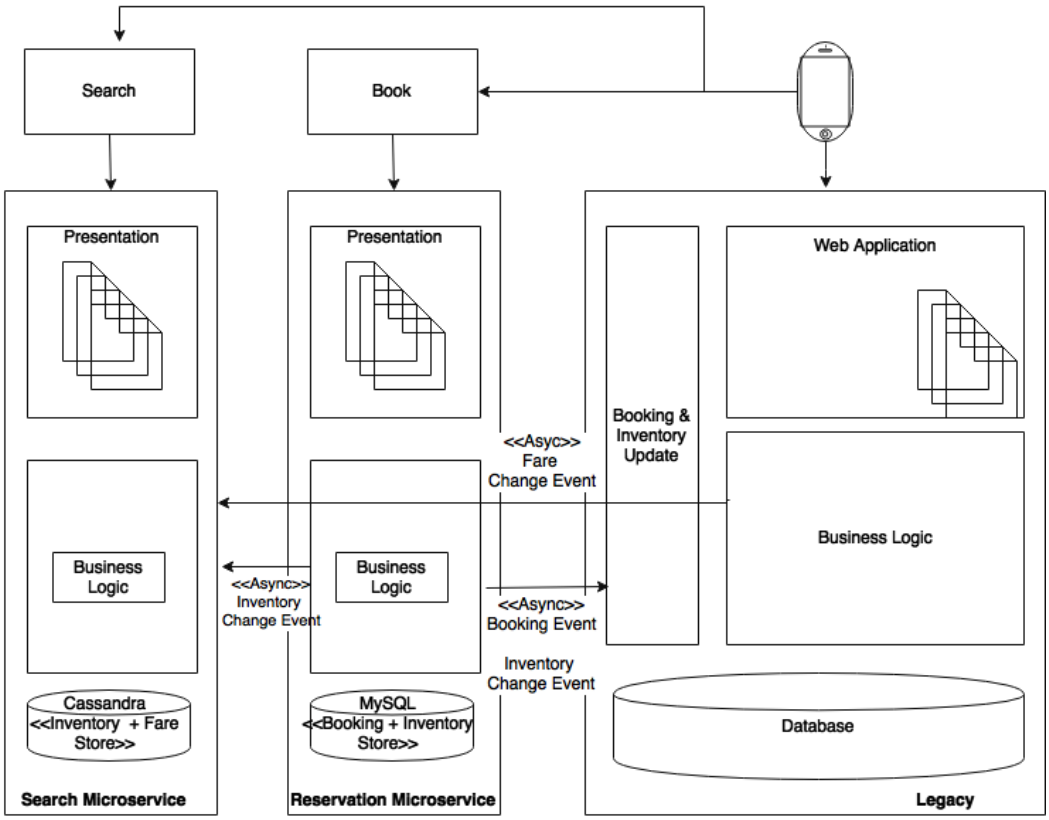

Book
Validate Flight
Flight
Validate Flight
Search

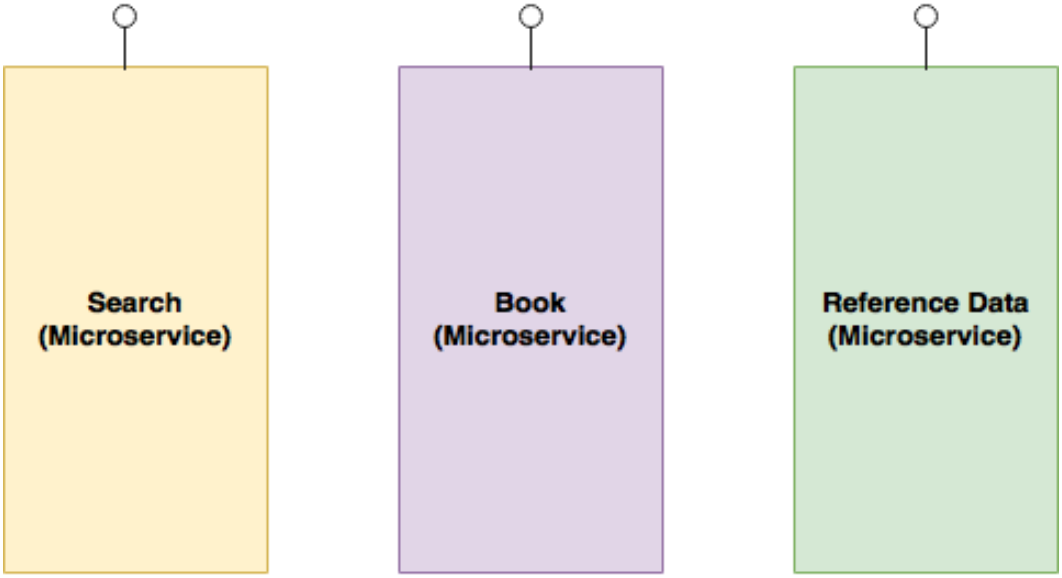

Fare
GetFares
GetFares
Book
Get Inventory
Update Inventory
Inventory
GetInventory
Search

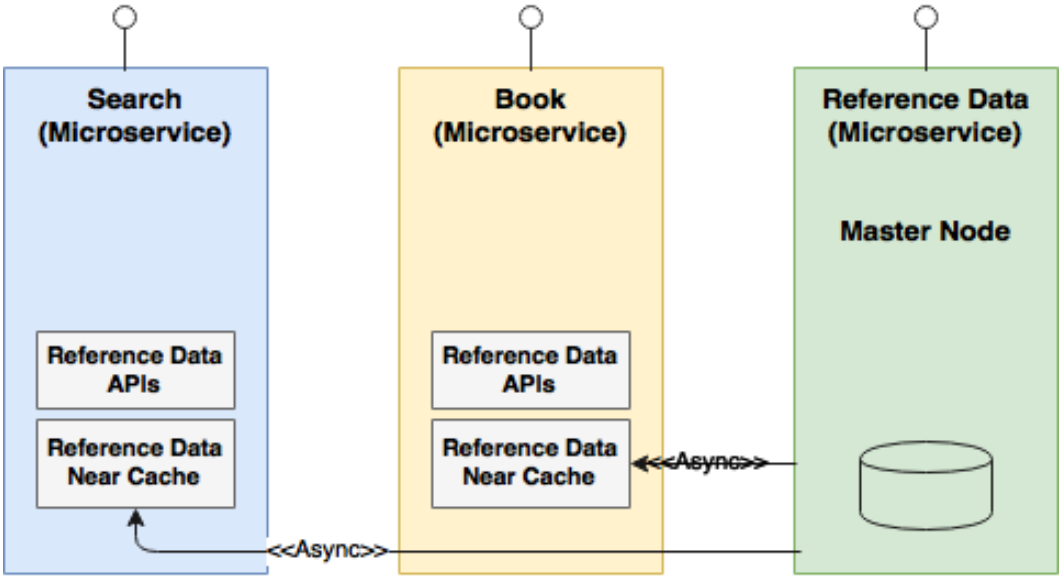

Fare
GetFares
<<Async>>
Fare Change Event
Book
(inventory master)
<<Async>>
Inventory
Change Event
Search
(Inventory + Fares
slave copy)

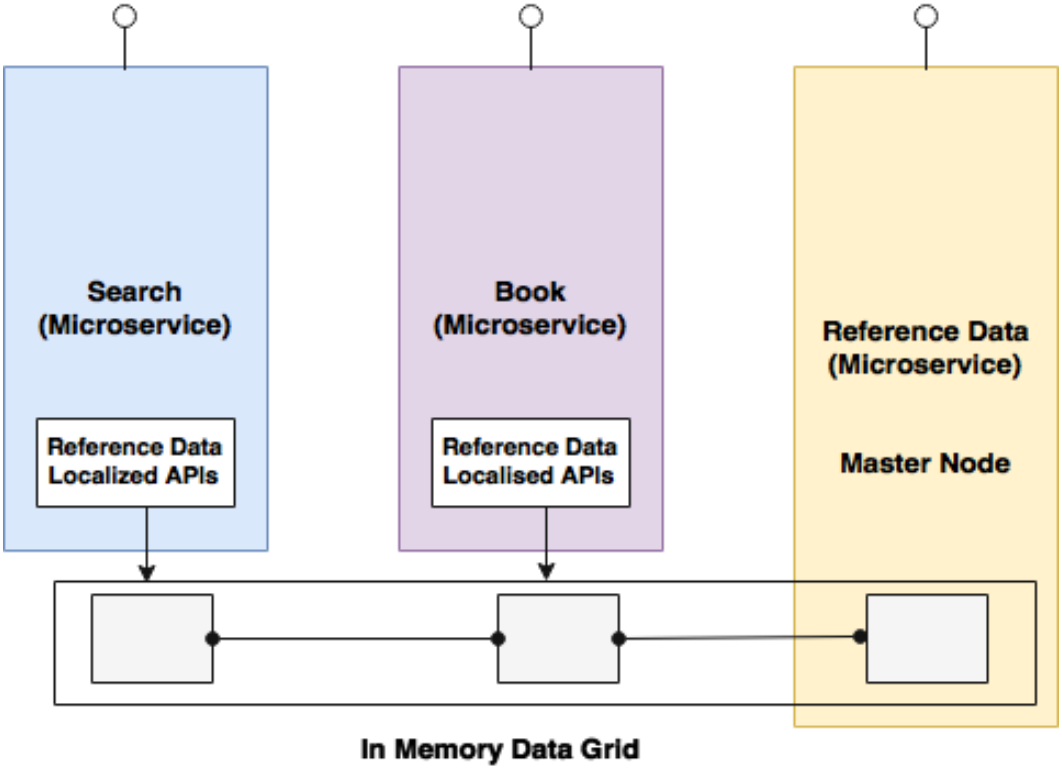

CRM
Accounting
getBooking
<<Async>>
Flight Change Event
Inventory Change Event
Flights
Book
Booking Event
<<Async>>
Loyalty
<<Async>>
Inventory Update Event
<<Sync>>GetFare
Revenue
Management
Search
<<Async>>
Fare Update Event
Fares
<<Async>>
FareChangeEvent
GetBulk
Check-In
<<Async>> CheckIn Event
Boarding

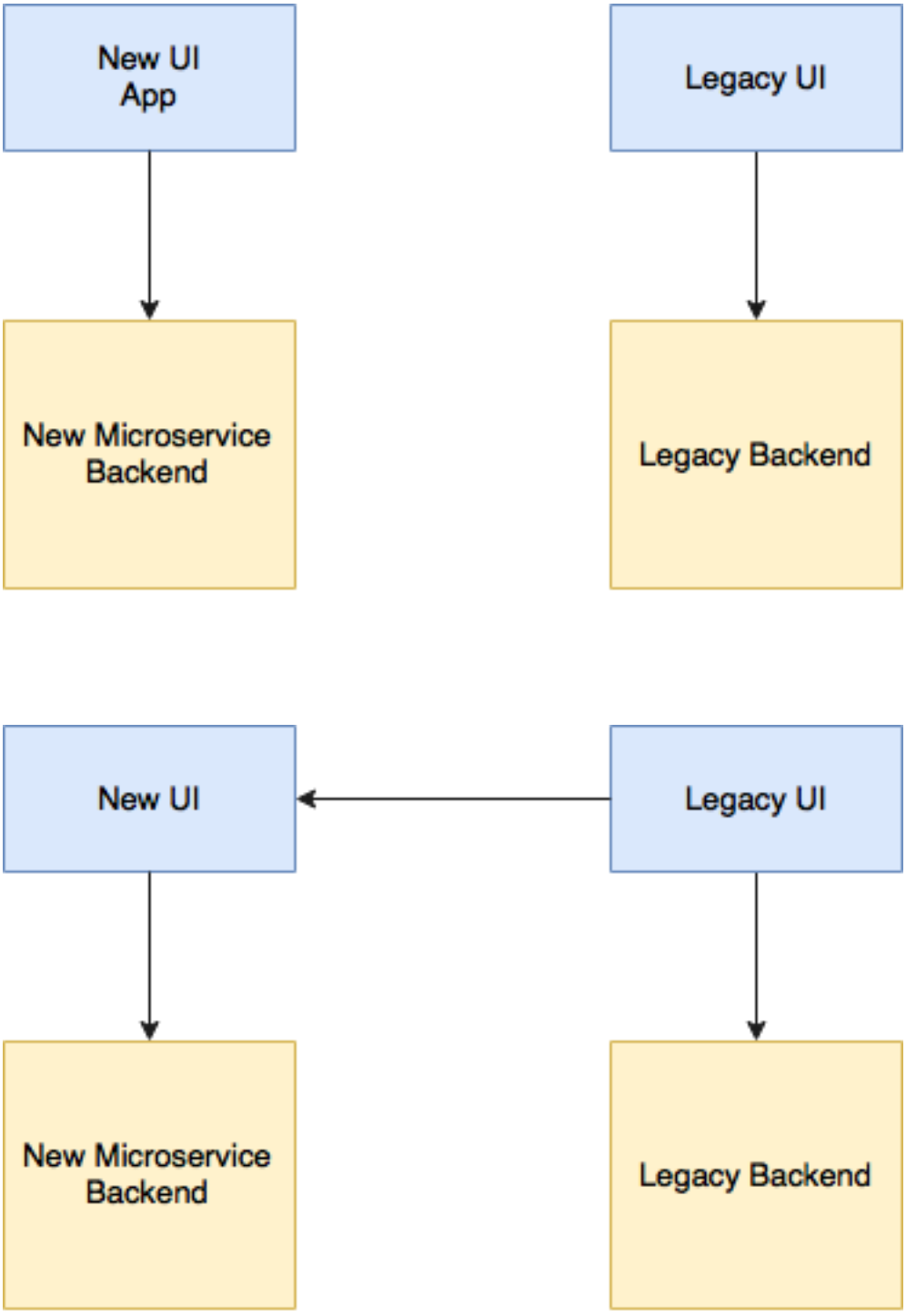

Search
Presentation
Business Logic
Cassandra
<<Inventory Store>>
Search
<<Async>>
Inventory
Change Event
Web Application
Business Logic
Database
Legacy

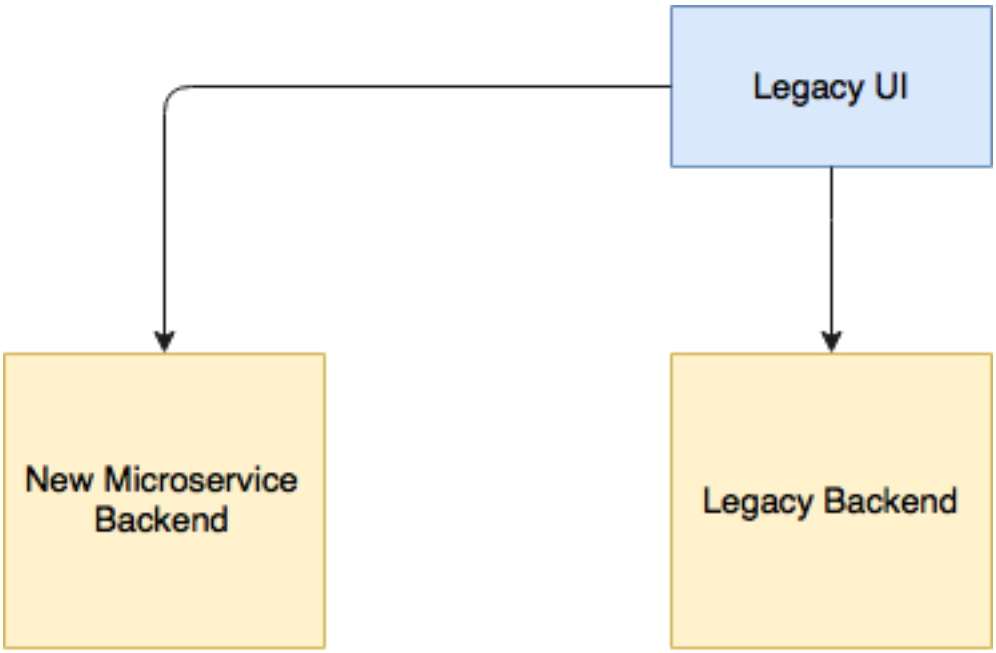

Search
Book
Presentation
Presentation
Web Application
Booking &
Inventory
Update
<<Asyc>>
Fare
Change Event
Business Logic
Business
Logic
<<Async>>
Inventory
Change Event
Business
Logic
<<Async>>
Booking Event
Inventory
Change Event
Cassandra
<<Inventory + Fare
Store>>
MySQL
<<Booking + Inventory
Store>>
Database
Search Microservice
Reservation Microservice
Legacy

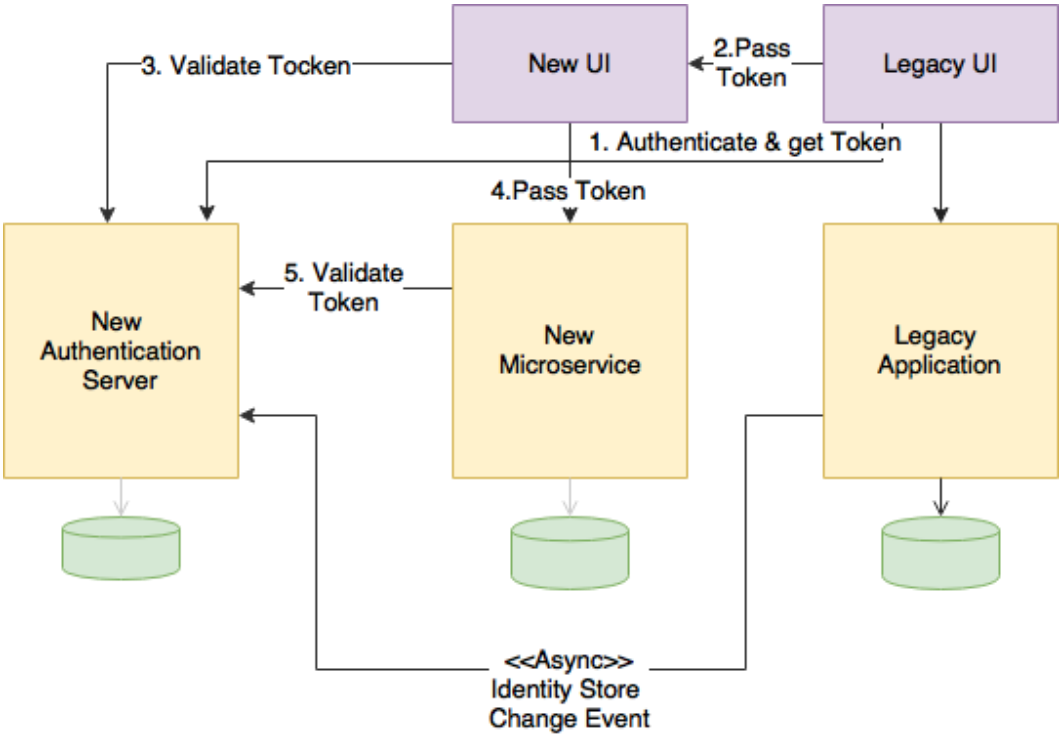

Search
(Microservice)
Book
(Microservice)
Reference Data
(Microservice)

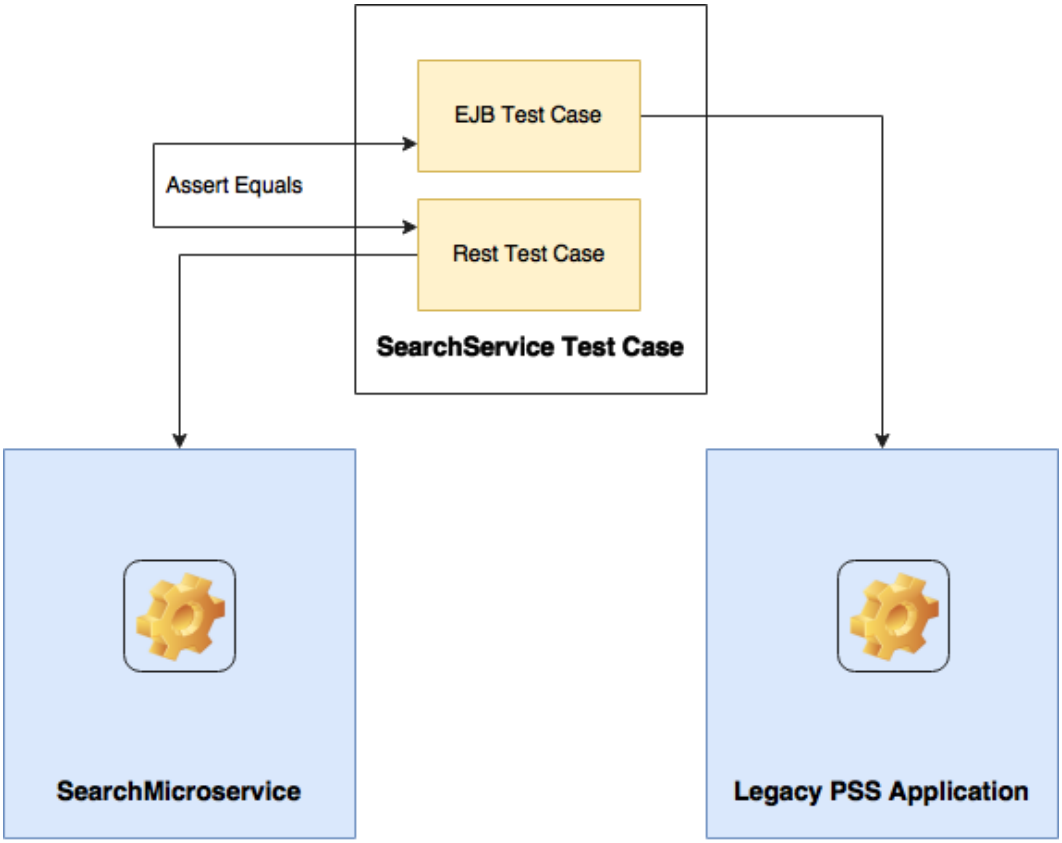

Search
(Microservice)
Reference Data
APIs
Reference Data
Near Cache
Book
(Microservice)
Reference Data
APIs
Reference Data
Near Cache
Reference Data
(Microservice)
Master Node
<<Async>>
<<Async>>

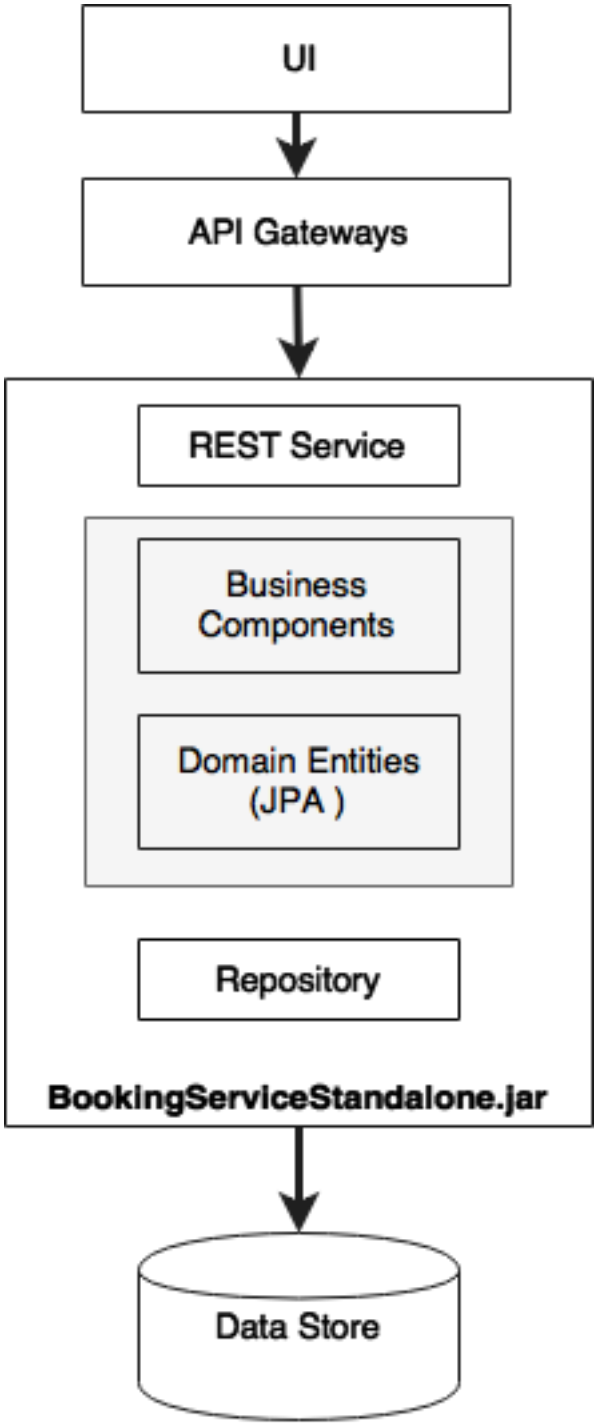

Search
(Microservice)
Reference Data
Localized APIs
Book
(Microservice)
Reference Data
Localised APIs
Reference Data
(Microservice)
Master Node
In Memory Data Grid

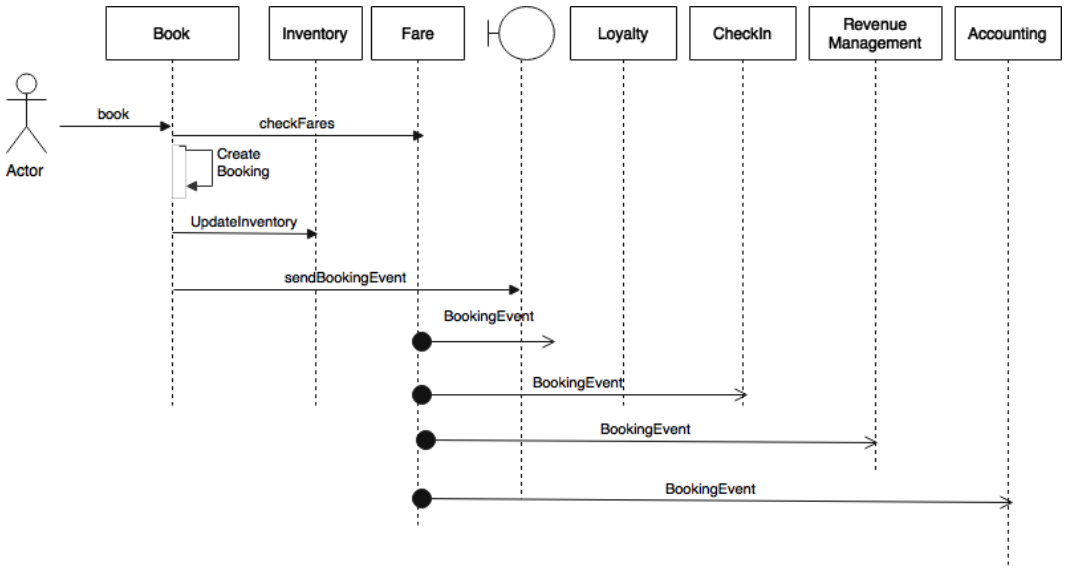

New UI
App
Legacy UI
New Microservice
Backend
Legacy Backend
New UI
Legacy UI
New Microservice
Backend
Legacy Backend

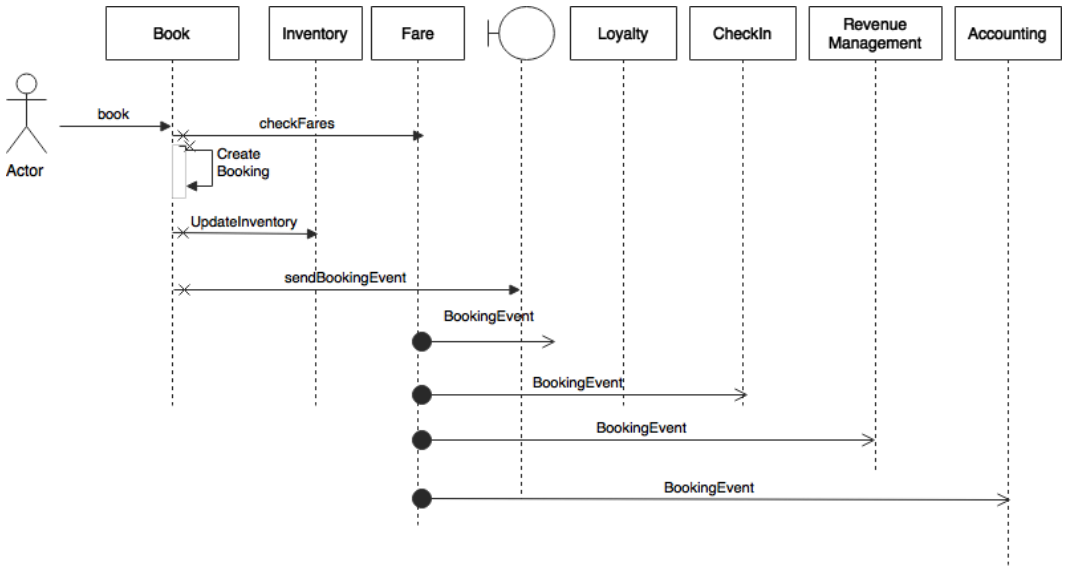

Legacy UI
New Microservice Backend
Legacy Backend

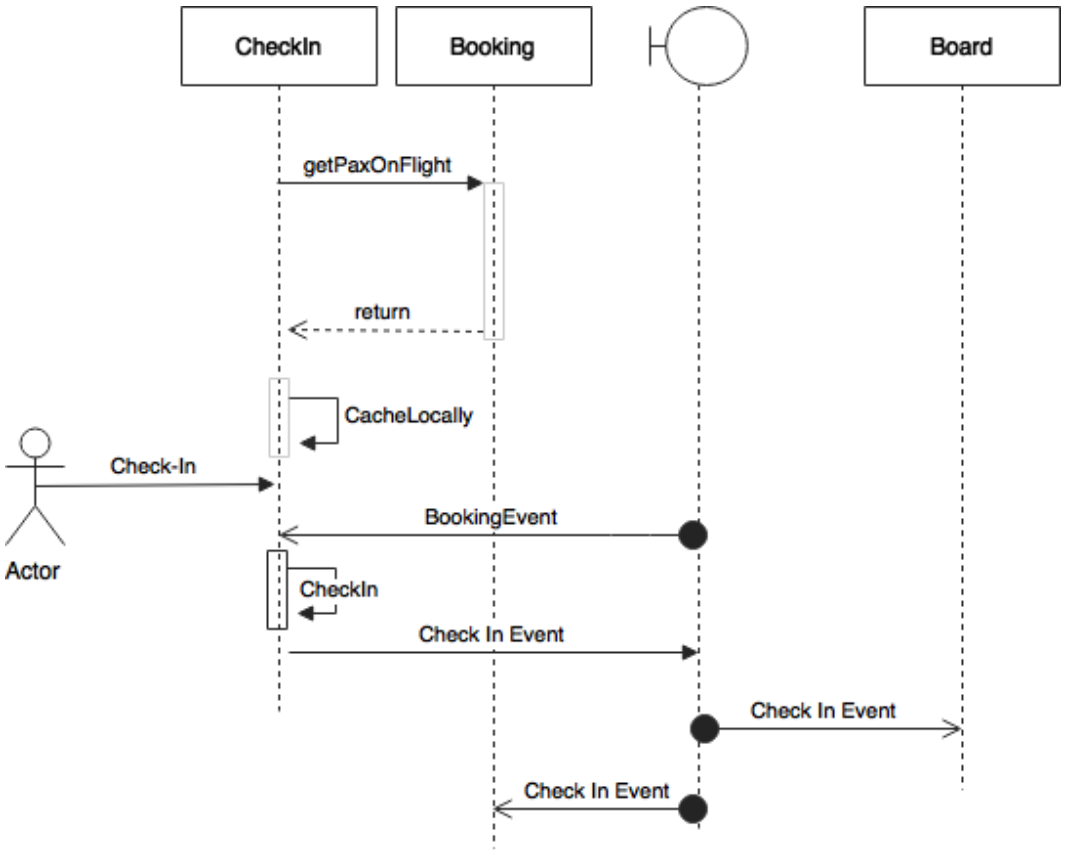

New UI
2.Pass Token
Legacy UI
3. Validate Tocken
1. Authenticate & get Token
4.Pass Token
5. Validate Token
New Authentication Server
New Microservice
Legacy Application
<<Async>>
Identity Store
Change Event

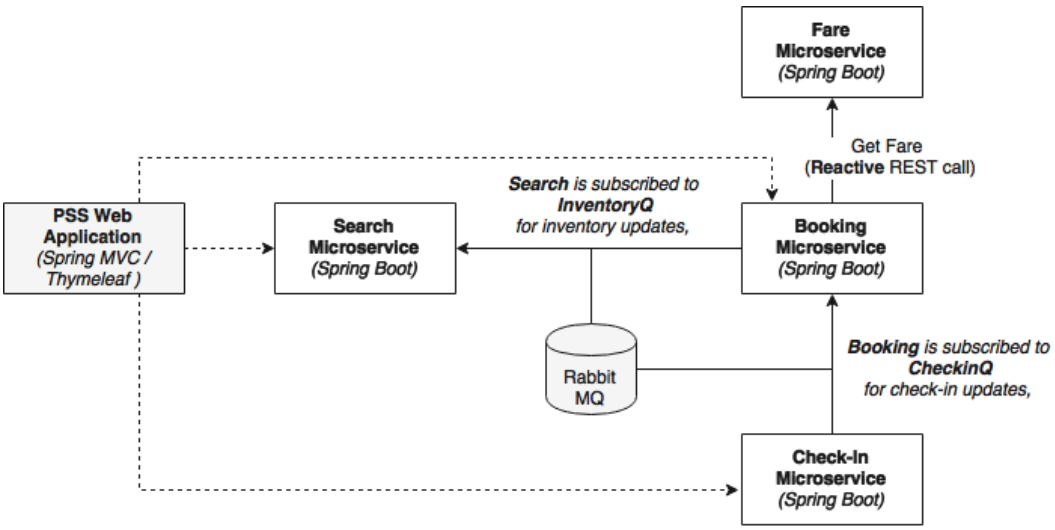

EJB Test Case
Assert Equals
Rest Test Case
SearchService Test Case
SearchMicroservice
Legacy PSS Application

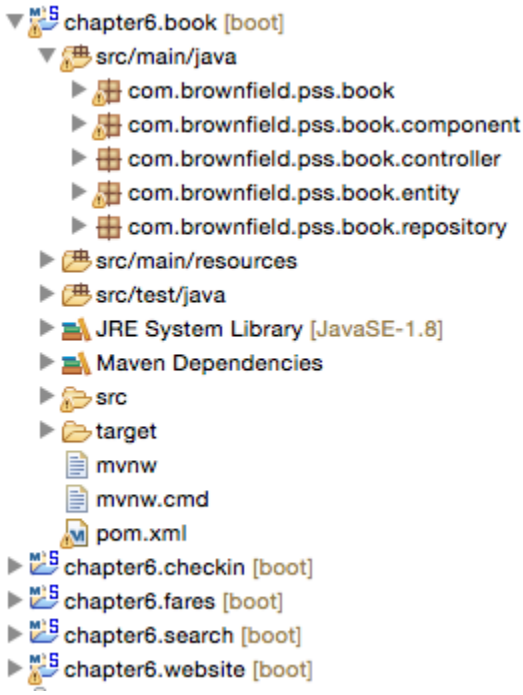

UI
API Gateways
REST Service
Business Components
Domain Entities (JPA)
Repository
BookingServiceStandalone.jar
Data Store

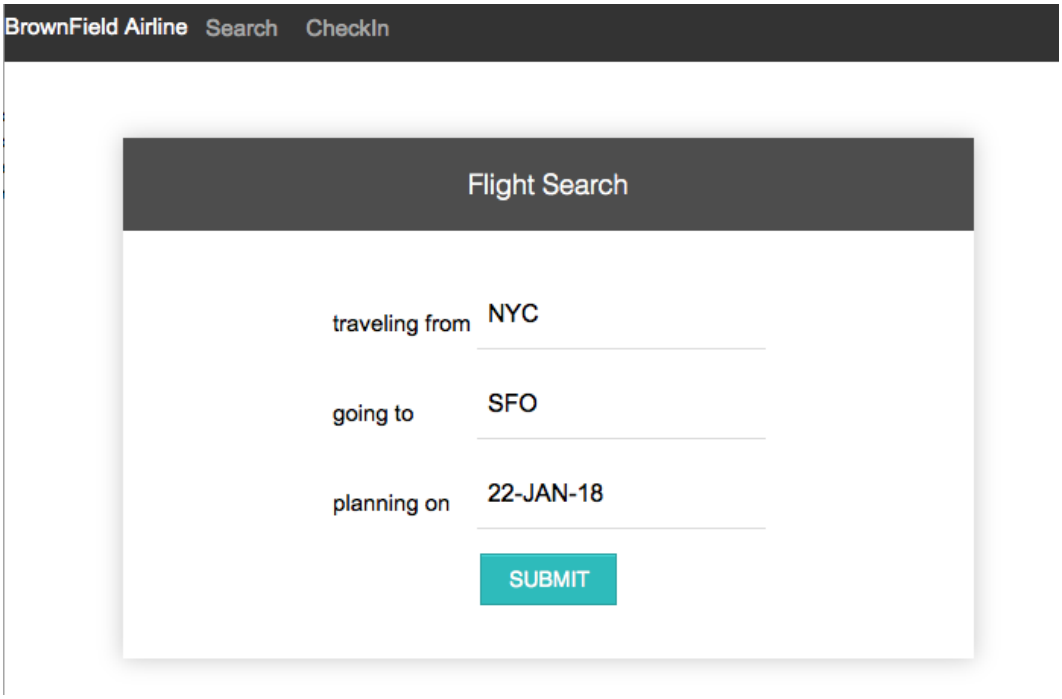

Book
Inventory
Fare
Loyalty
CheckIn
Revenue Management
Accounting
Actor
book
checkFares
Create Booking
UpdateInventory
sendBookingEvent
BookingEvent
BookingEvent
BookingEvent
BookingEvent

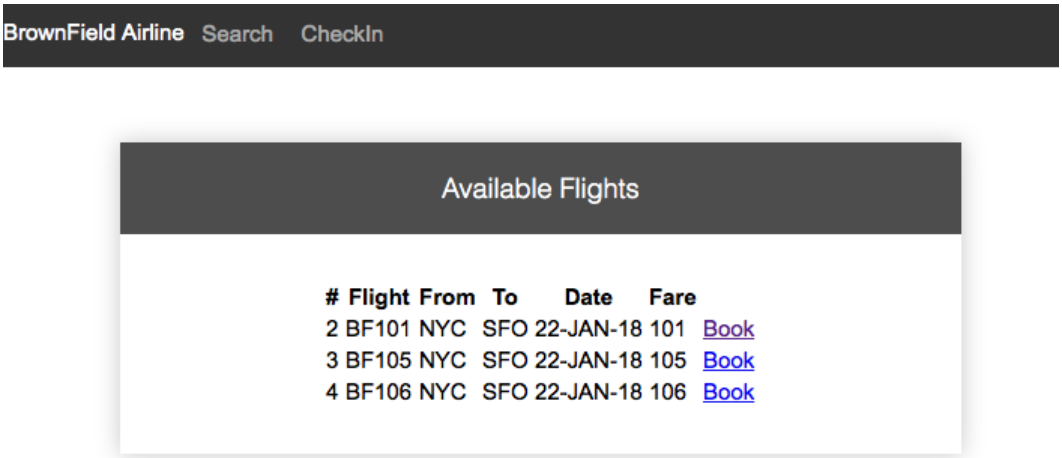

Book
Inventory
Fare
Loyalty
CheckIn
Revenue Management
Accounting
Actor
book
checkFares
Create Booking
UpdateInventory
sendBookingEvent
BookingEvent
BookingEvent
BookingEvent
BookingEvent

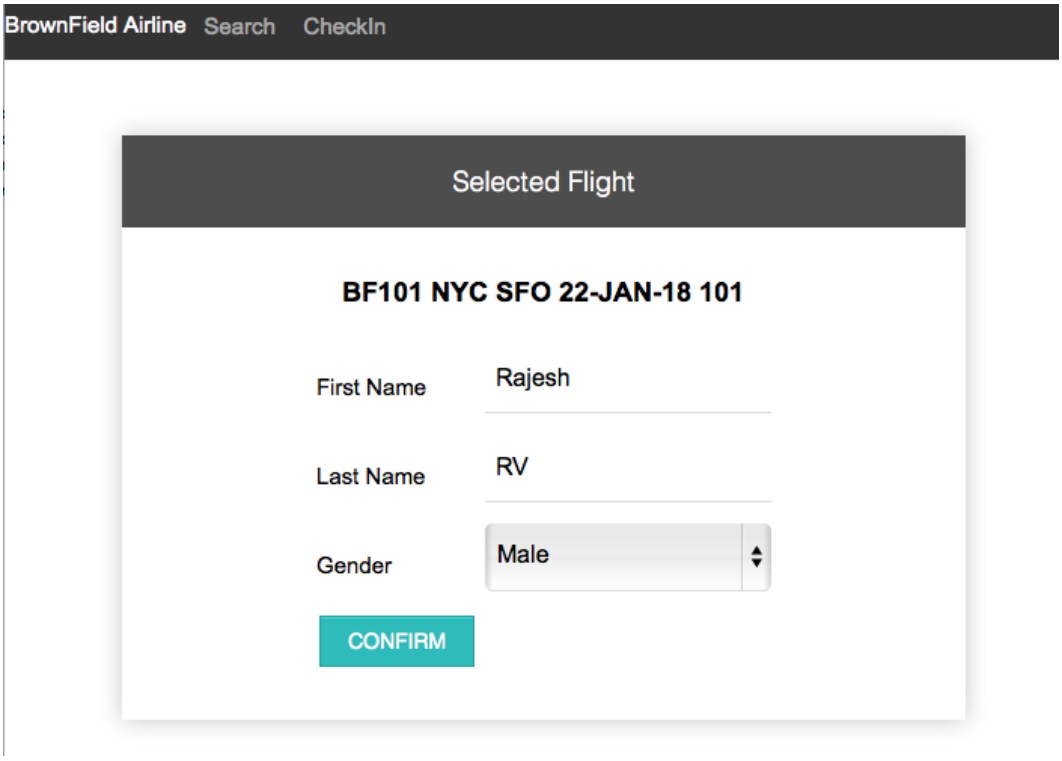

CheckIn
Booking
Board
getPaxOnFlight
return
CacheLocally
Check-In
Actor
BookingEvent
CheckIn
Check In Event
Check In Event
Check In Event

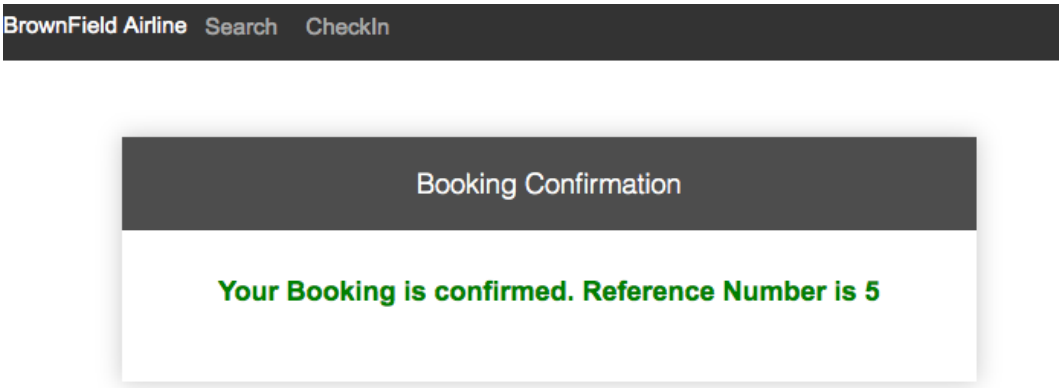

Fare
Microservice
(Spring Boot)
Get Fare
(Reactive REST call)
Search is subscribed to
InventoryQ
for inventory updates,
PSS Web
Application
(Spring MVC /
Thymeleaf)
Search
Microservice
(Spring Boot)
Booking
Microservice
(Spring Boot)
Rabbit
MQ
Booking is subscribed to
CheckinQ
for check-in updates,
Check-In
Microservice
(Spring Boot)

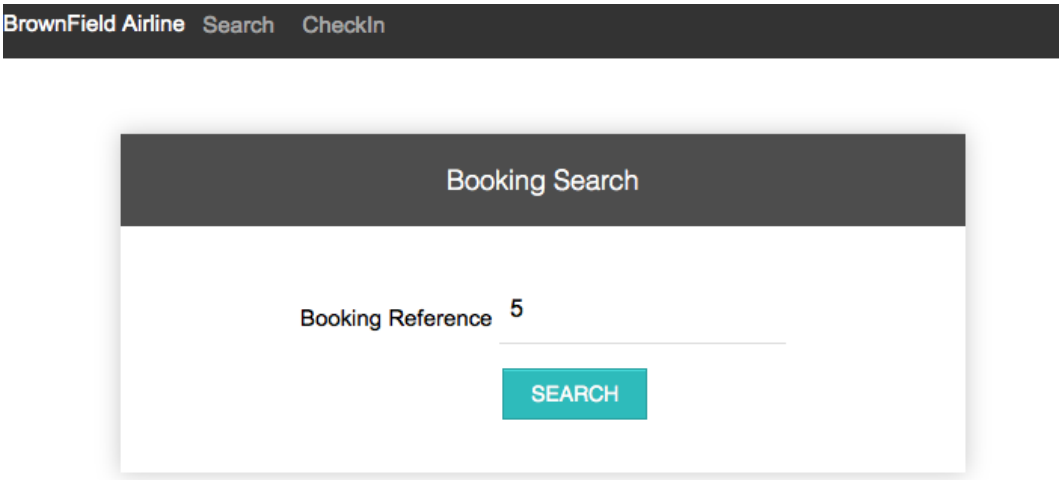

chapter6.book [boot]
src/main/java
com.brownfield.pss.book
com.brownfield.pss.book.component
com.brownfield.pss.book.controller
com.brownfield.pss.book.entity
com.brownfield.pss.book.repository
src/main/resources
src/test/java
JRE System Library [JavaSE-1.8]
Maven Dependencies
src
target
mvnw
mvnw.cmd
pom.xml
chapter6.checkin [boot]
chapter6.fares [boot]
chapter6.search [boot]
chapter6.website [boot]

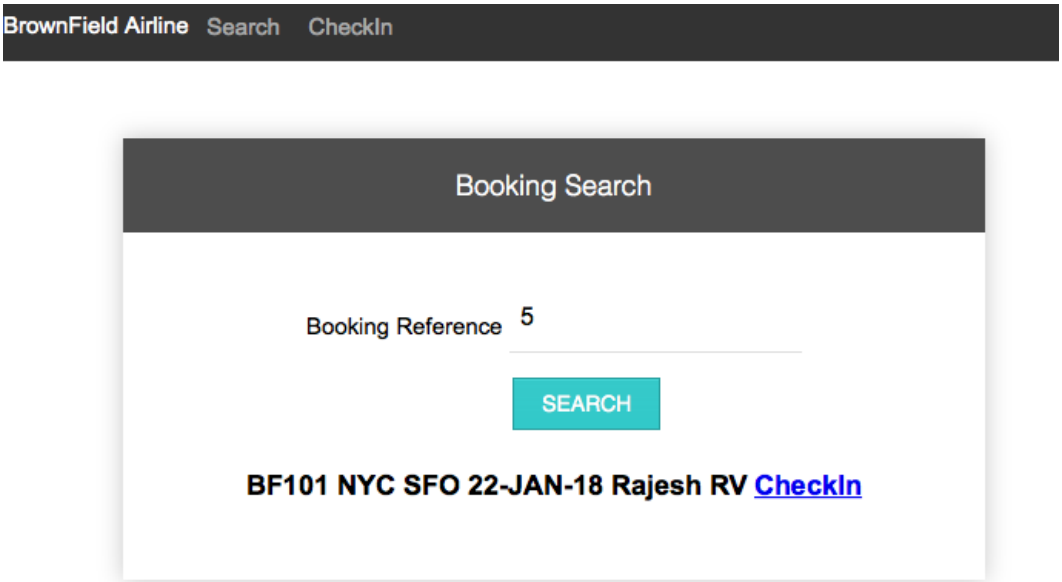

BrownField Airline
Search
CheckIn
Flight Search
traveling from
NYC
going to
SFO
planning on
22-JAN-18
SUBMIT


BrownField Airline
Search
CheckIn
Available Flights
Flight From To Date Fare
2 BF101 NYC SFO 22-JAN-18 101 Book
3 BF105 NYC SFO 22-JAN-18 105 Book
4 BF106 NYC SFO 22-JAN-18 106 Book


BrownField Airline
Search
CheckIn
Selected Flight
BF101 NYC SFO 22-JAN-18 101
First Name
Rajesh
Last Name
RV
Gender
Male
CONFIRM


BrownField Airline
Search
CheckIn
Booking Confirmation
Your Booking is confirmed. Reference Number is 5


BrownField Airline
Search
CheckIn
Booking Search
Booking Reference
5
SEARCH

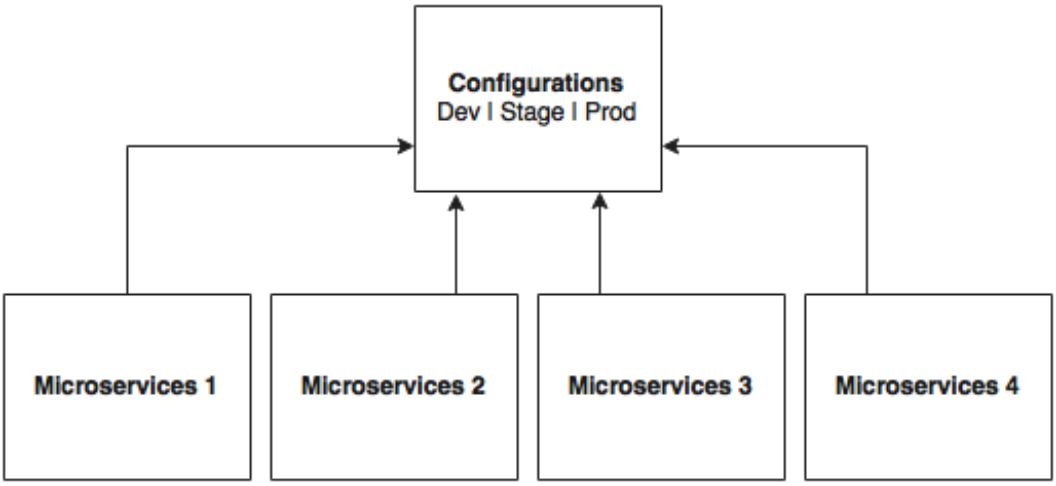

BrownField Airline
Search
CheckIn
Booking Search
Booking Reference
5
SEARCH
BF101 NYC SFO 22-JAN-18 Rajesh RV CheckIn

BrownField Airline Search CheckIn

Check In Confirmation

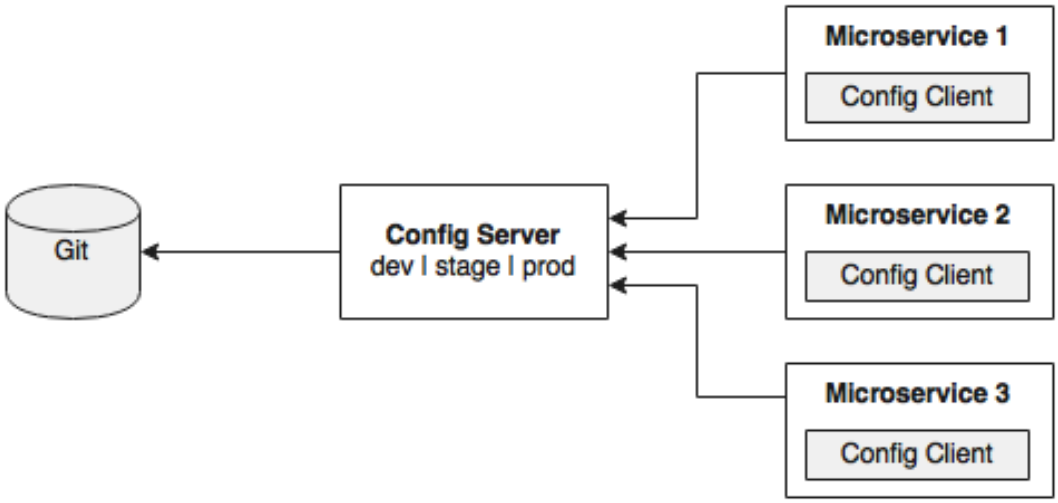
Checked In, Seat Number is 28c , checkin id is 5

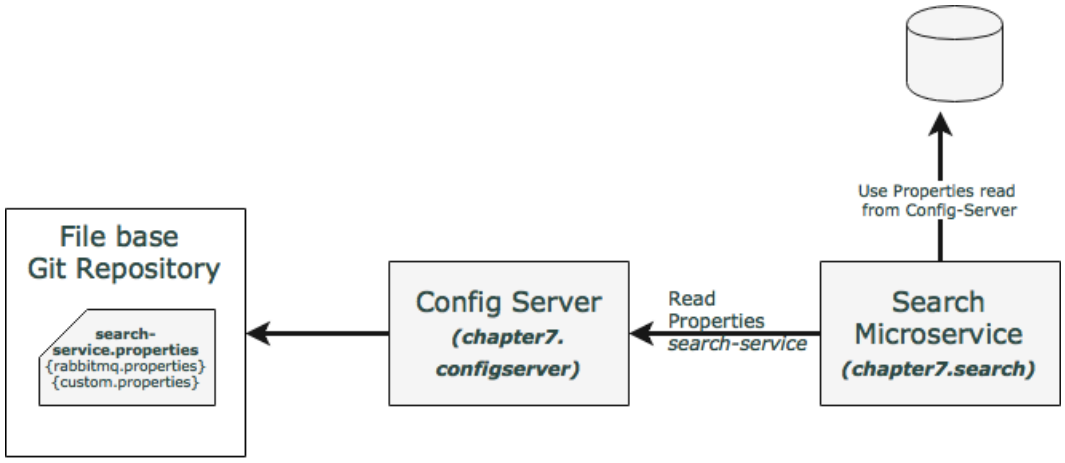
Chapter 7: Scale Microservices with Spring Cloud Components

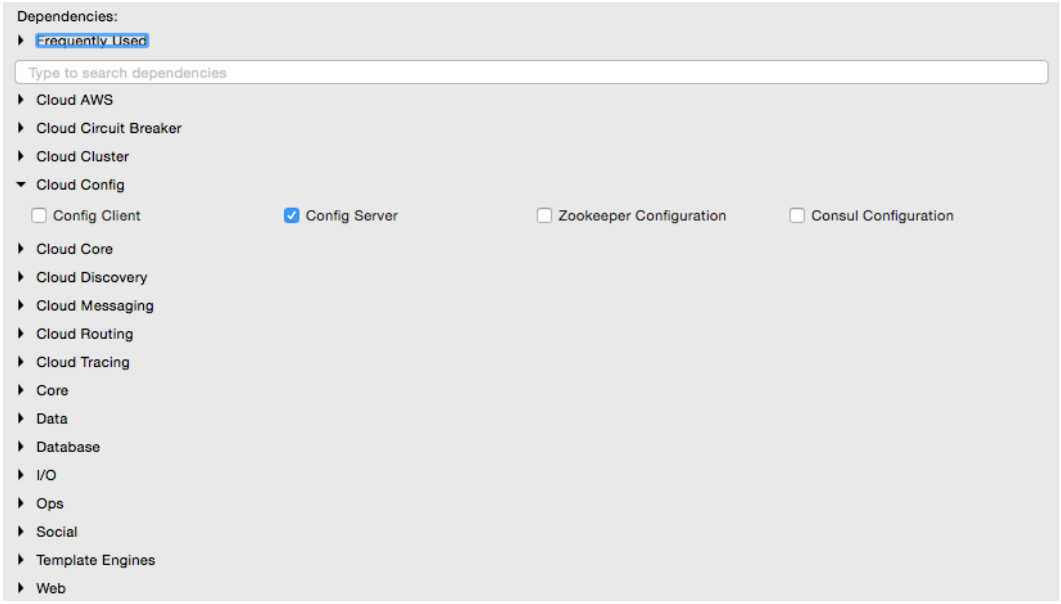


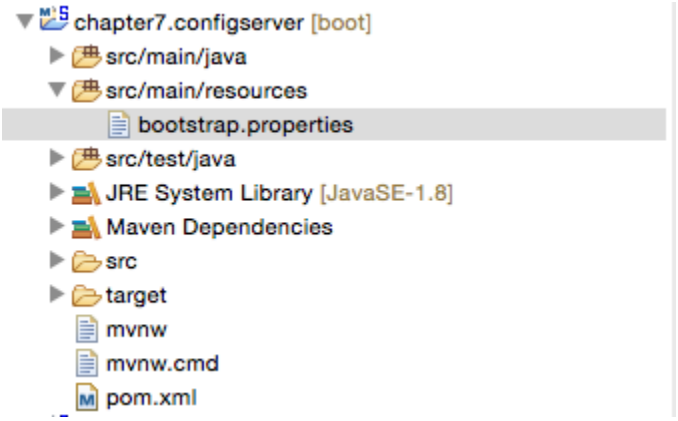
Cloud Config

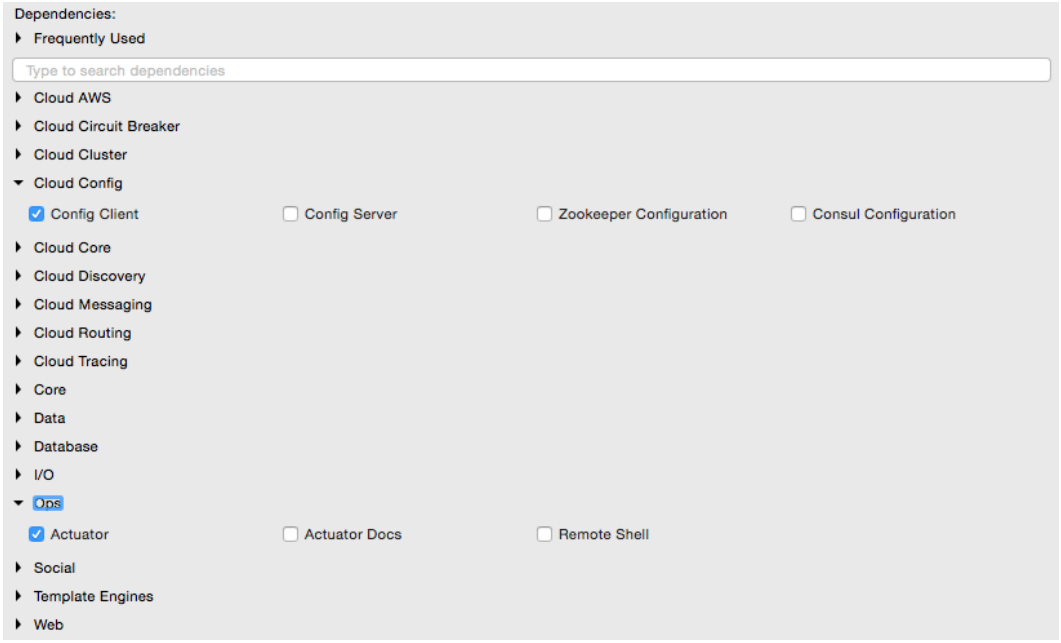
- ☐ Config Client
  spring-cloud-config Client
- ☐ Config Server
  Central management for configuration via a git or svn backend
- ☐ Zookeeper Configuration
  Configuration management with Zookeeper and spring-cloud-zookeeper-config
- ☐ Consul Configuration
  Configuration management with Hashicorp Consul

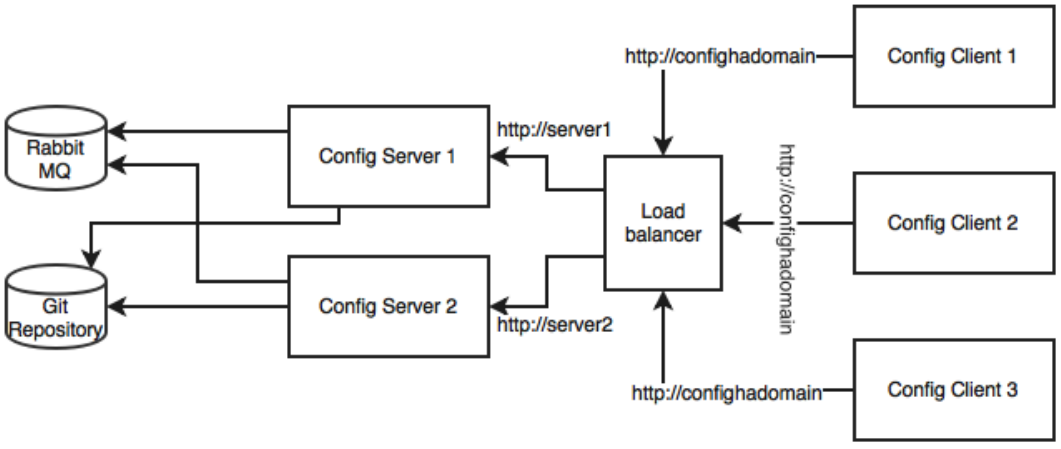

Microservice 1
Config Client
Microservice 2
Config Client
Microservice 3
Config Client
Git
Config Server
dev I stage I prod


Use Properties read
from Config-Server
File base
Git Repository
search-
service.properties
{rabbitmq.properties}
{custom.properties}
Config Server
(chapter7.
configserver)
Read
Properties
search-service
Search
Microservice
(chapter7.search)


Dependencies:
Frequently Used
Type to search dependencies
Cloud AWS
Cloud Circuit Breaker
Cloud Cluster
Cloud Config
Config Client
Config Server
Zookeeper Configuration
Consul Configuration
Cloud Core
Cloud Discovery
Cloud Messaging
Cloud Routing
Cloud Tracing
Core
Data
Database
I/O
Ops
Social
Template Engines
Web


chapter7.configserver [boot]
src/main/java
src/main/resources
bootstrap.properties
src/test/java
JRE System Library [JavaSE-1.8]
Maven Dependencies
src
target
mvnw
mvnw.cmd
pom.xml


Dependencies:
Frequently Used
Type to search dependencies
Cloud AWS
Cloud Circuit Breaker
Cloud Cluster
Cloud Config
Config Client
Config Server
Zookeeper Configuration
Consul Configuration
Cloud Core
Cloud Discovery
Cloud Messaging
Cloud Routing
Cloud Tracing
Core
Data
Database
I/O
Ops
Actuator
Actuator Docs
Remote Shell
Social
Template Engines
Web


http://confighadomain
Config Client 1
Rabbit MQ
Config Server 1
http://server1
Load balancer
http://confighadomain
Config Client 2
Git Repository
Config Server 2
http://server2
http://confighadomain
Config Client 3

Cloud Discovery

- [ ] Eureka Discovery

  Service discovery using spring-cloud-netflix and Eureka

- [ ] Eureka Server

  spring-cloud-netflix Eureka Server

- [ ] Zookeeper Discovery

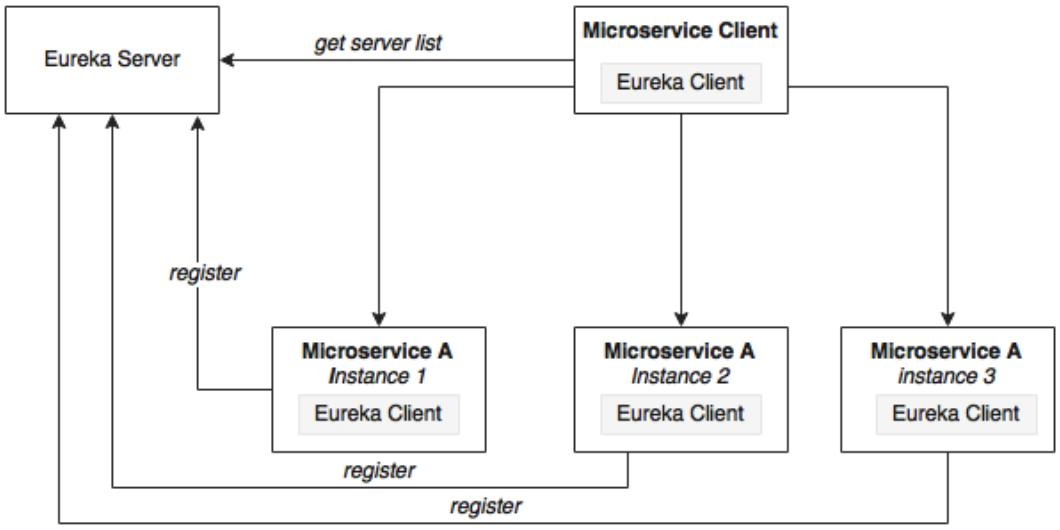
  Service discovery with Zookeeper and spring-cloud-zookeeper-discovery

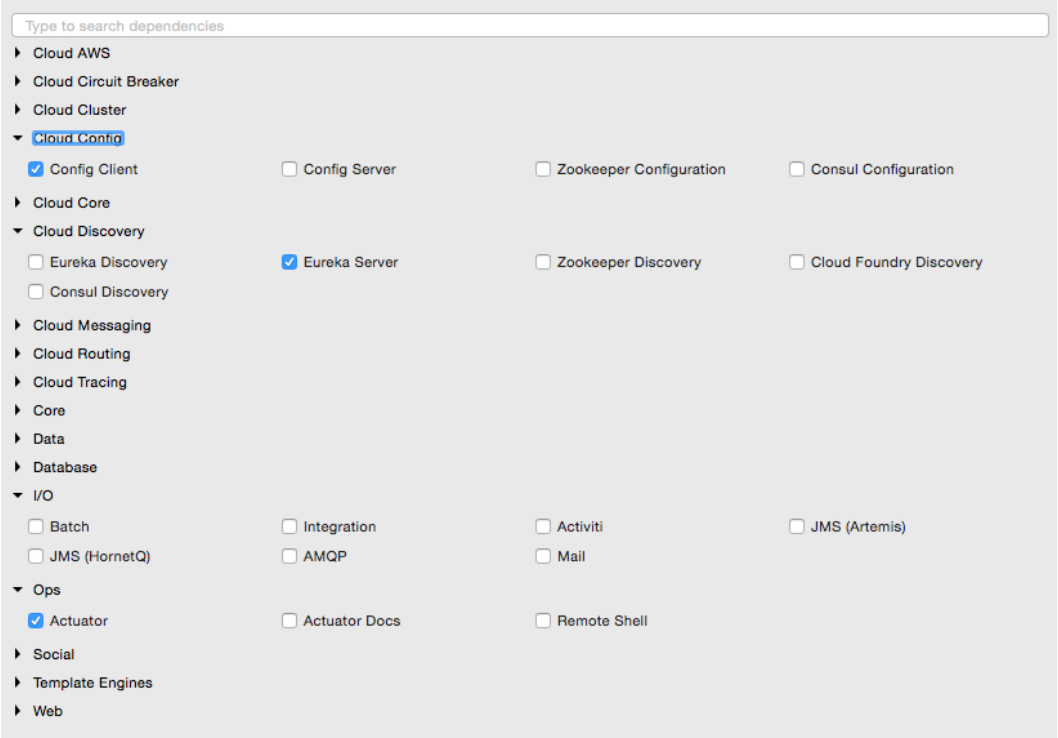
- [ ] Cloud Foundry Discovery

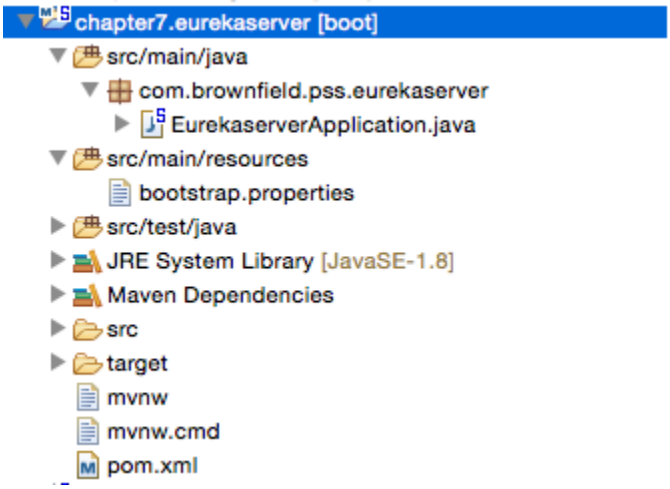
  Service discovery with Cloud Foundry

- [ ] Consul Discovery

  Service discovery with Hashicorp Consul

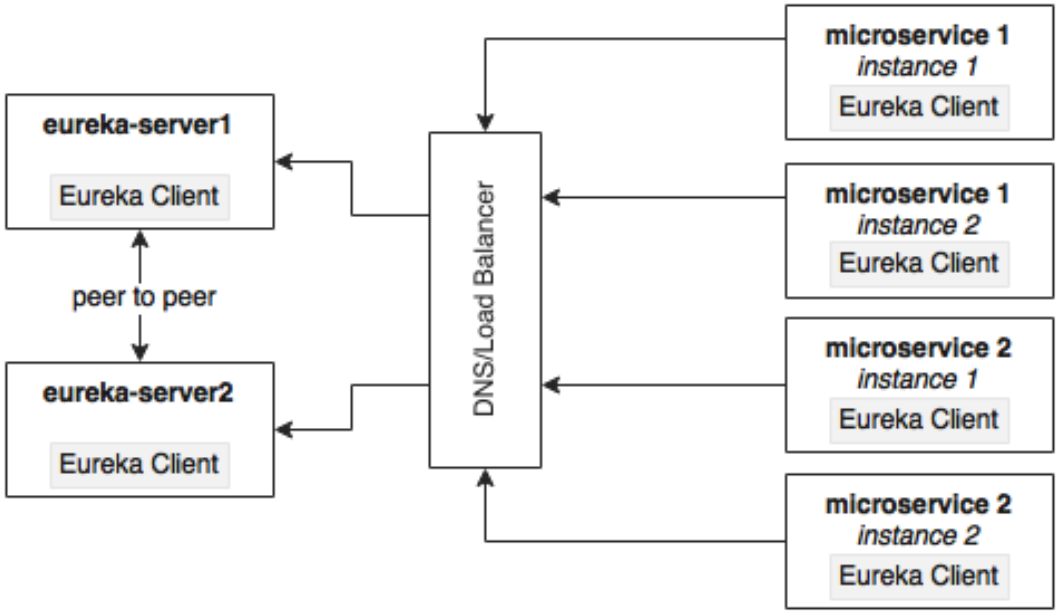


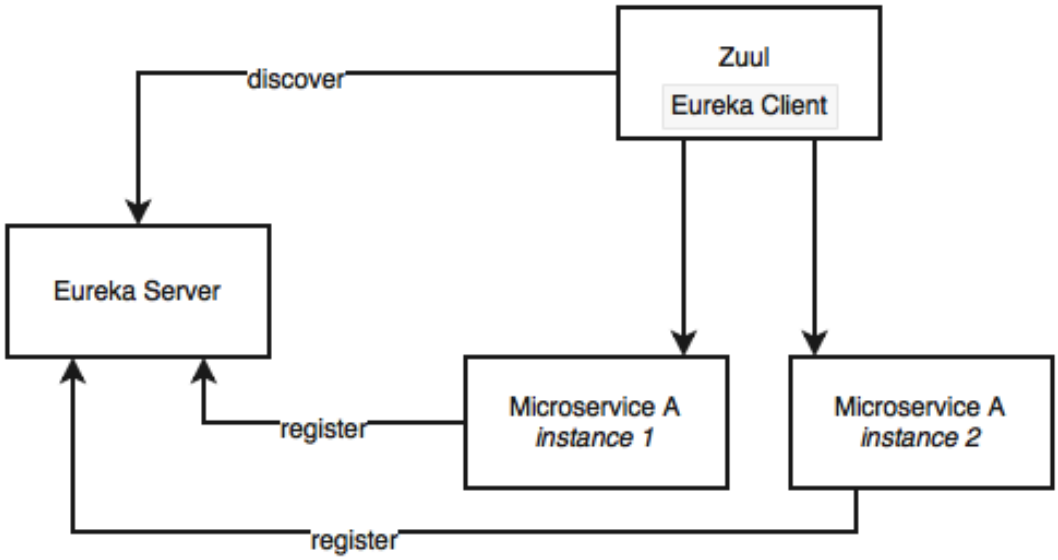

Type to search dependencies
Cloud AWS
Cloud Circuit Breaker
Cloud Cluster
Cloud Config
Config Client
Config Server
Zookeeper Configuration
Consul Configuration
Cloud Core
Cloud Discovery
Eureka Discovery
Eureka Server
Zookeeper Discovery
Cloud Foundry Discovery
Consul Discovery
Cloud Messaging
Cloud Routing
Cloud Tracing
Core
Data
Database
I/O
Batch
Integration
Activiti
JMS (Artemis)
JMS (HornetQ)
AMQP
Mail
Ops
Actuator
Actuator Docs
Remote Shell
Social
Template Engines
Web

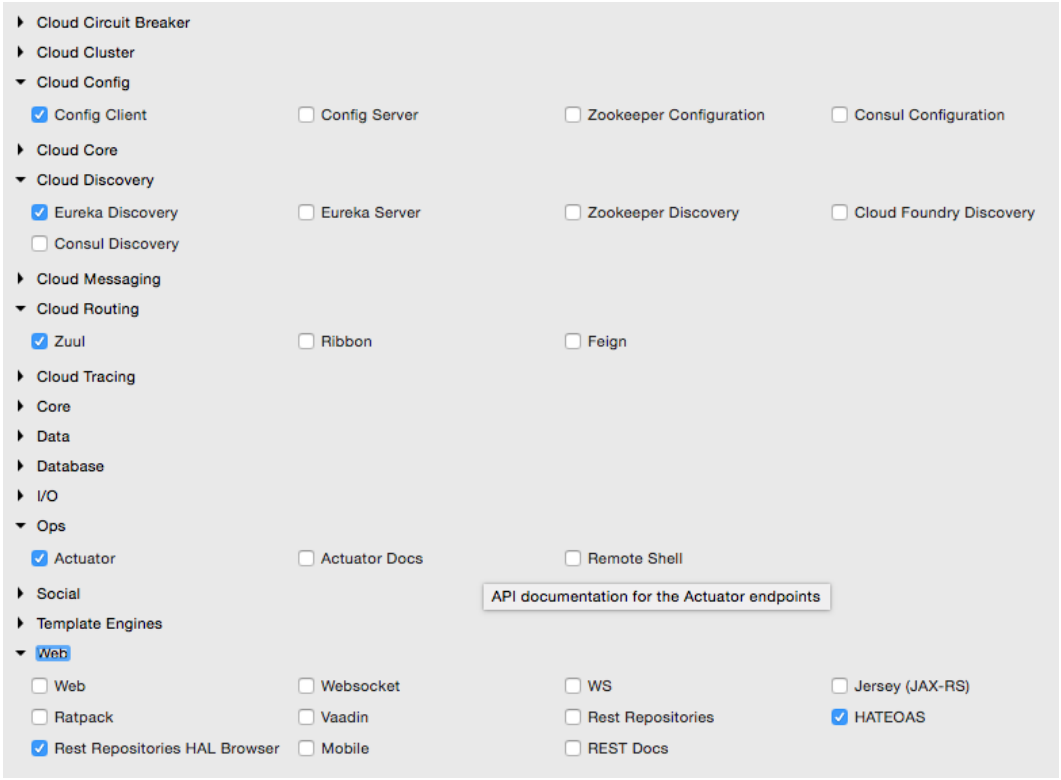

chapter7.eurekaserver [boot]
src/main/java
com.brownfield.pss.eurekaserver
EurekaserverApplication.java
src/main/resources
bootstrap.properties
src/test/java
JRE System Library [JavaSE-1.8]
Maven Dependencies
src
target
mvnw
mvnw.cmd
pom.xml

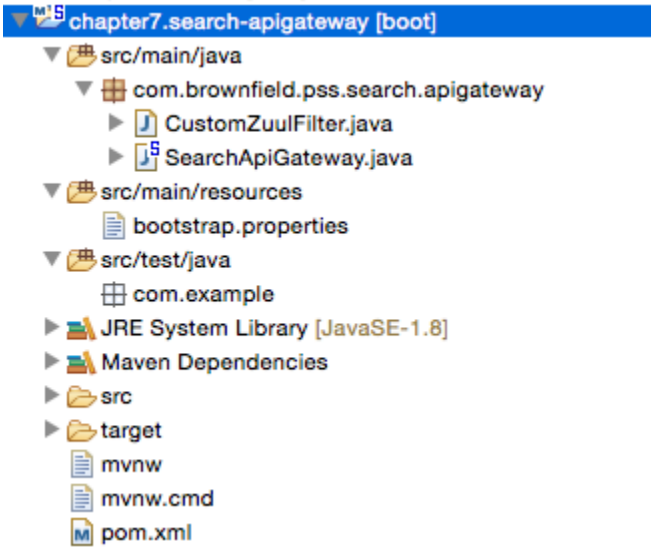
Instances currently registered with Eureka

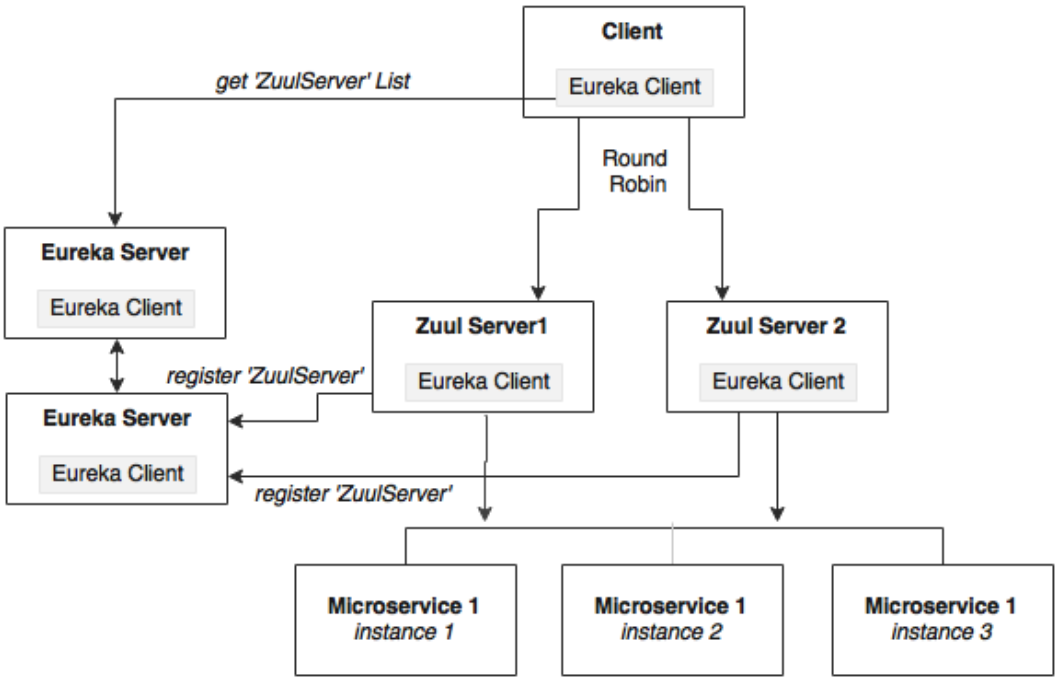
Application	AMIs	Availability Zones	Status
BOOK-SERVICE	n/a (1)	(1)	UP (1) - 192.168.0.102:book-service:8060
CHECKIN-SERVICE	n/a (1)	(1)	UP (1) - 192.168.0.102:checkin-service:8070
FARES-SERVICE	n/a (1)	(1)	UP (1) - 192.168.0.102:fares-service:8080
SEARCH-SERVICE	n/a (1)	(1)	UP (1) - 192.168.0.102:search-service:8090

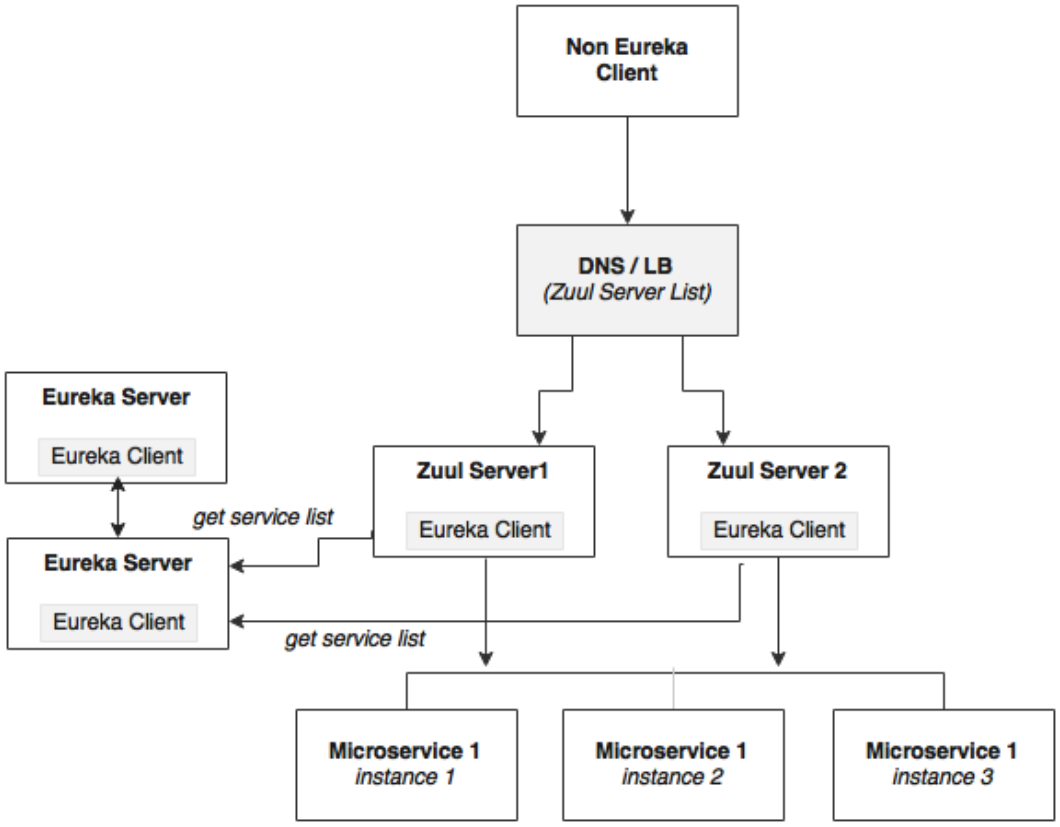

microservice 1
instance 1
Eureka Client
eureka-server1
Eureka Client
microservice 1
instance 2
Eureka Client
peer to peer
DNS/Load Balancer
microservice 2
instance 1
Eureka Client
eureka-server2
Eureka Client
microservice 2
instance 2
Eureka Client

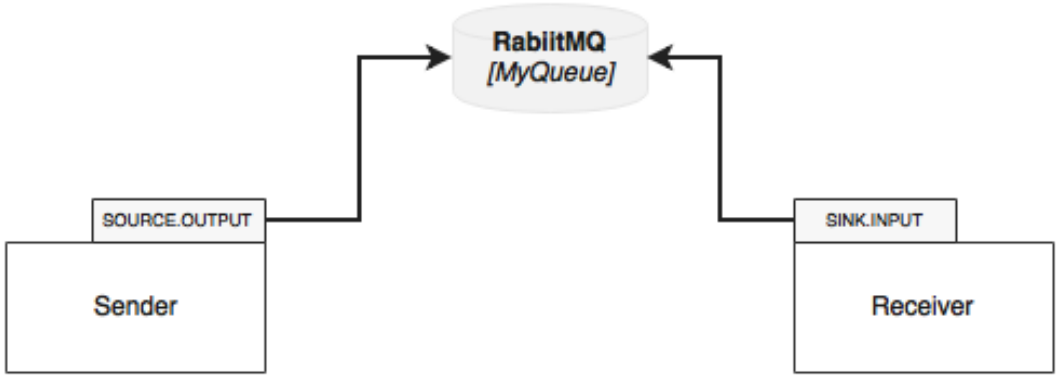

Zuul
Eureka Client
discover
Eureka Server
register
Microservice A
instance 1
Microservice A
instance 2
register

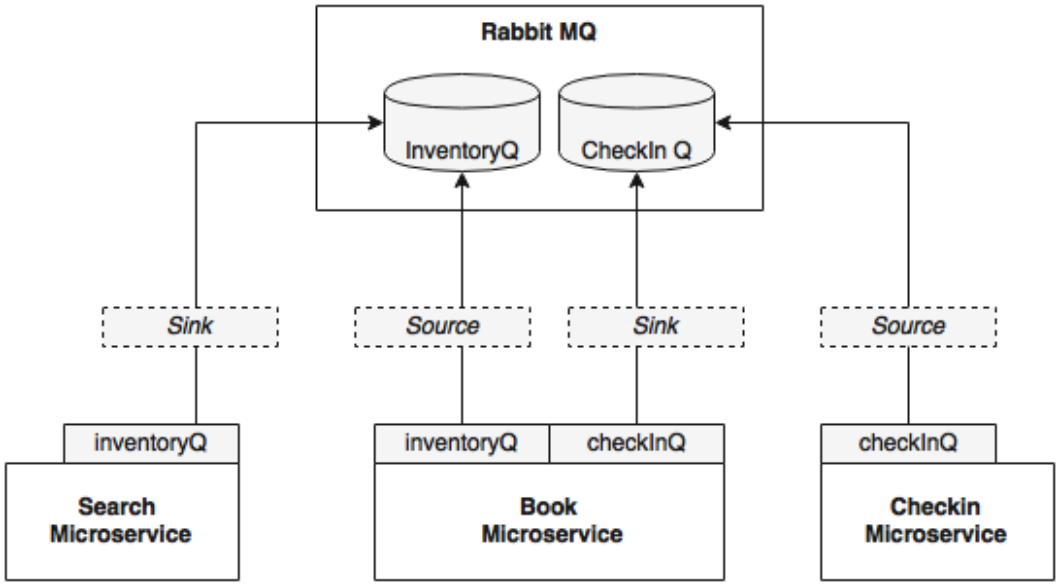

Cloud Circuit Breaker
Cloud Cluster
Cloud Config
Config Client
Config Server
Zookeeper Configuration
Consul Configuration
Cloud Core
Cloud Discovery
Eureka Discovery
Eureka Server
Zookeeper Discovery
Cloud Foundry Discovery
Consul Discovery
Cloud Messaging
Cloud Routing
Zuul
Ribbon
Feign
Cloud Tracing
Core
Data
Database
I/O
Ops
Actuator
Actuator Docs
Remote Shell
Social
API documentation for the Actuator endpoints
Template Engines
Web
Web
Websocket
WS
Jersey (JAX-RS)
Ratpack
Vaadin
Rest Repositories
HATEOAS
Rest Repositories HAL Browser
Mobile
REST Docs

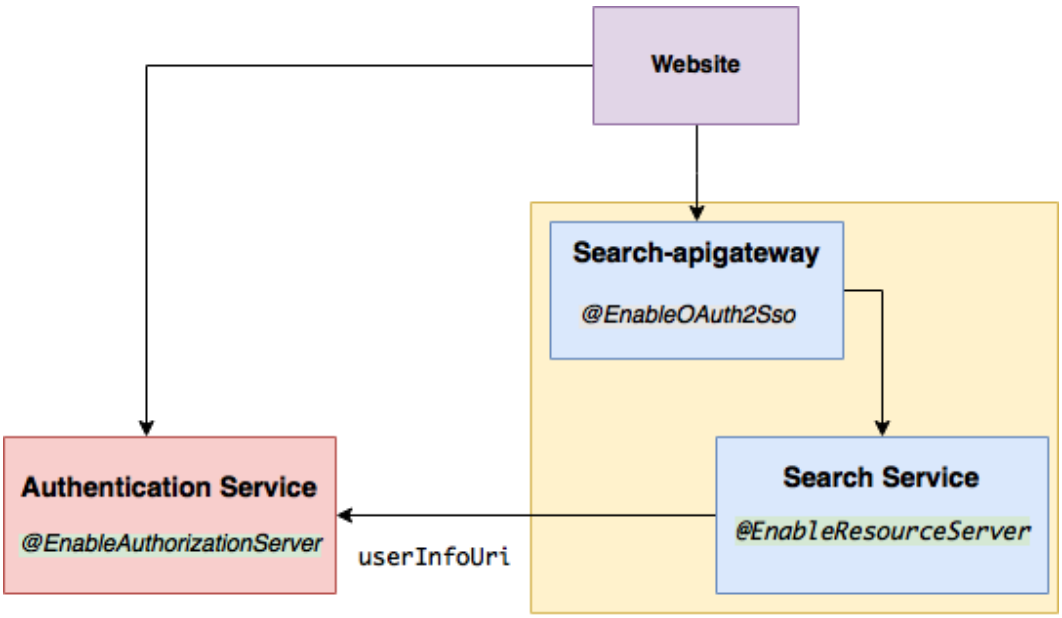

chapter7.search-apigateway [boot]
src/main/java
com.brownfield.pss.search.apigateway
CustomZuulFilter.java
SearchApiGateway.java
src/main/resources
bootstrap.properties
src/test/java
com.example
JRE System Library [JavaSE-1.8]
Maven Dependencies
src
target
mvnw
mvnw.cmd
pom.xml

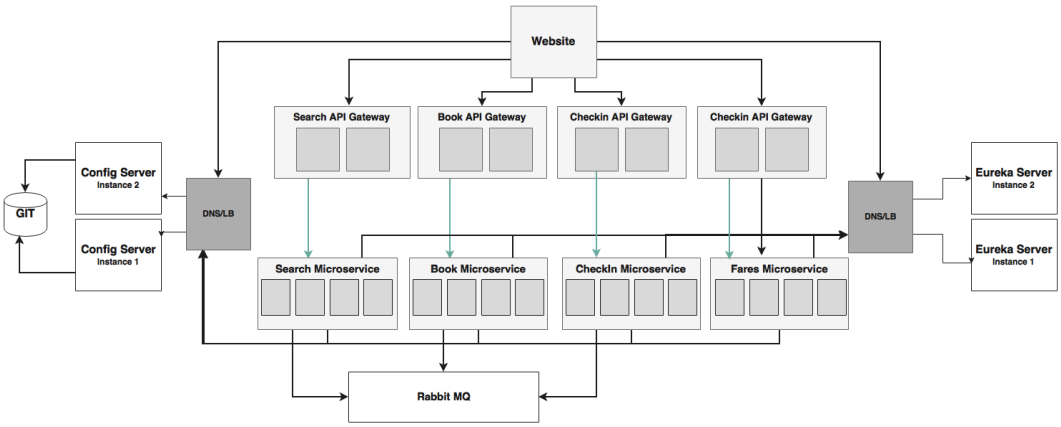

Client
Eureka Client
get 'ZuulServer' List
Round
Robin
Eureka Server
Eureka Client
Zuul Server1
Eureka Client
Zuul Server 2
Eureka Client
register 'ZuulServer'
Eureka Server
Eureka Client
register 'ZuulServer'
Microservice 1
instance 1
Microservice 1
instance 2
Microservice 1
instance 3


Non Eureka Client
DNS / LB
(Zuul Server List)
Eureka Server
Eureka Client
Zuul Server1
Eureka Client
Zuul Server 2
Eureka Client
get service list
Eureka Server
Eureka Client
get service list
Microservice 1
instance 1
Microservice 1
instance 2
Microservice 1
instance 3

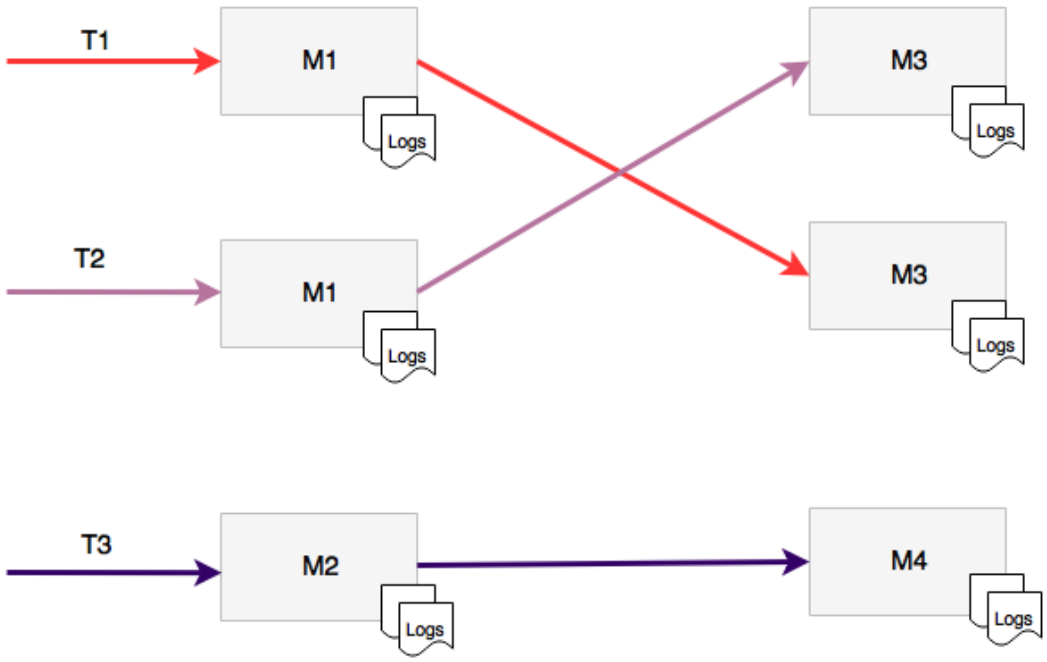

RabiitMQ
[MyQueue]
SOURCE.OUTPUT
Sender
SINK.INPUT
Receiver

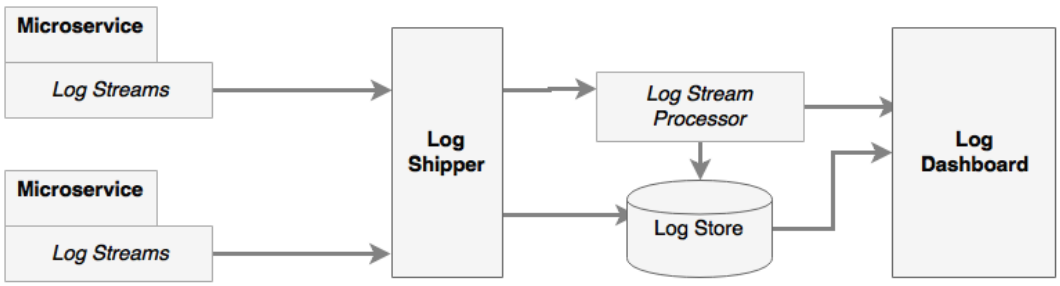

Rabbit MQ
InventoryQ
CheckIn Q
Sink
Source
Sink
Source
inventoryQ
Search Microservice
inventoryQ
checkInQ
Book Microservice
checkInQ
Checkin Microservice


Website
Search-apigateway
@EnableOAuth2Sso
Authentication Service
@EnableAuthorizationServer
userInfoUri
Search Service
@EnableResourceServer

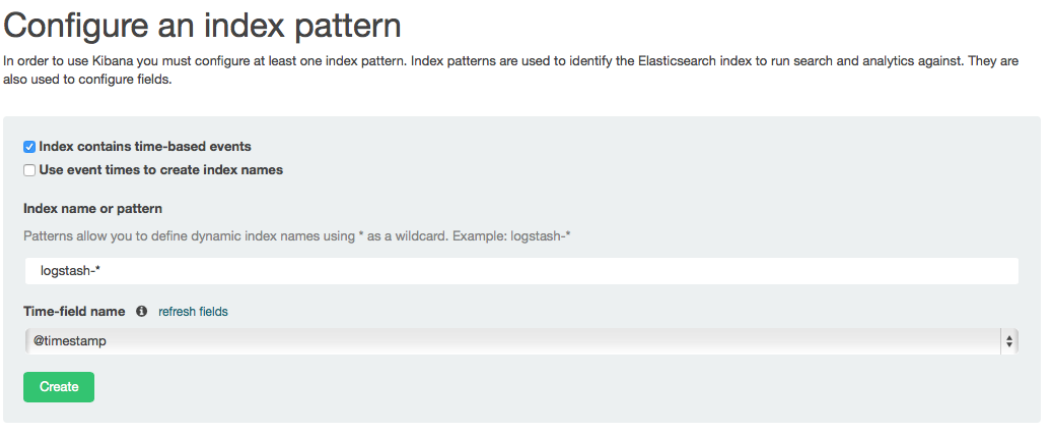

Website
Search API Gateway
Book API Gateway
Checkin API Gateway
Checkin API Gateway
Config Server
Instance 2
GIT
Config Server
Instance 1
DNS/LB
DNS/LB
Eureka Server
Instance 2
Eureka Server
Instance 1
Search Microservice
Book Microservice
CheckIn Microservice
Fares Microservice
Rabbit MQ

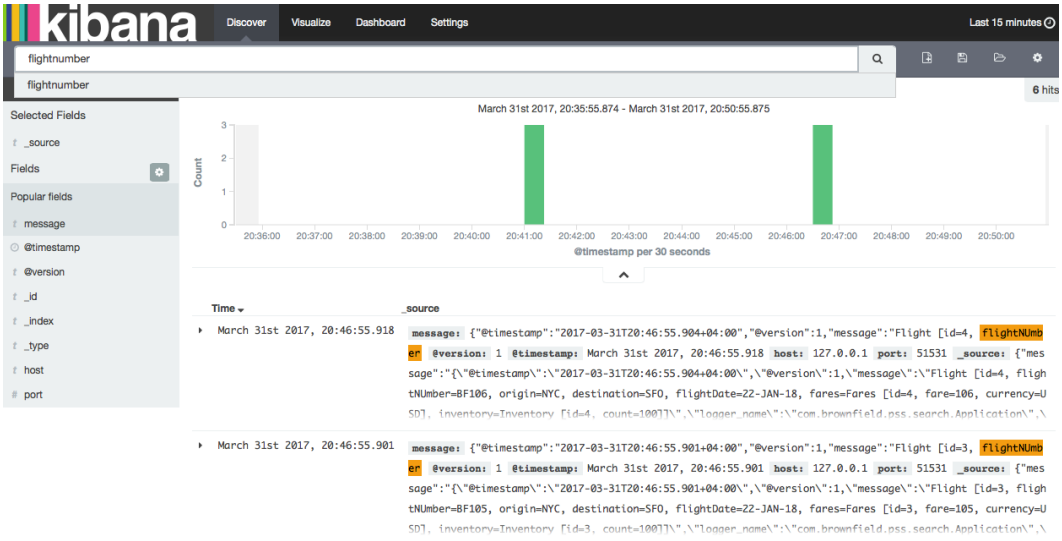
Chapter 8: Logging and Monitoring Microservices



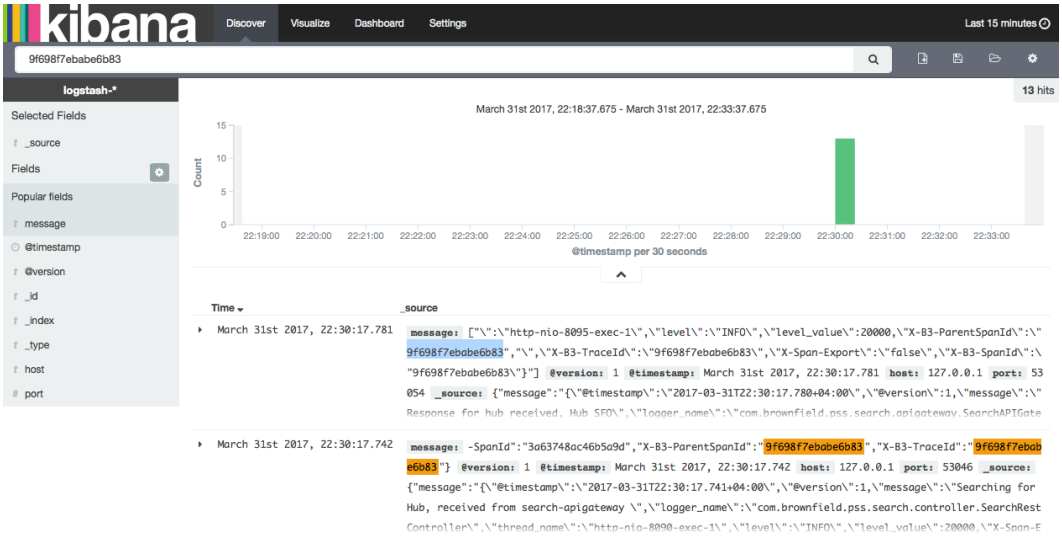


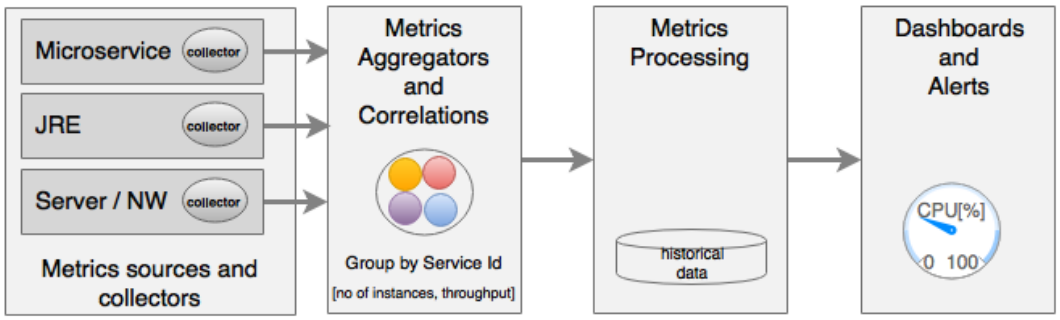
Microservice
Logback / Log 4j appenders
Logstash
(shipper)
Elasticsearch
(store)
Kibana
(dashboard)

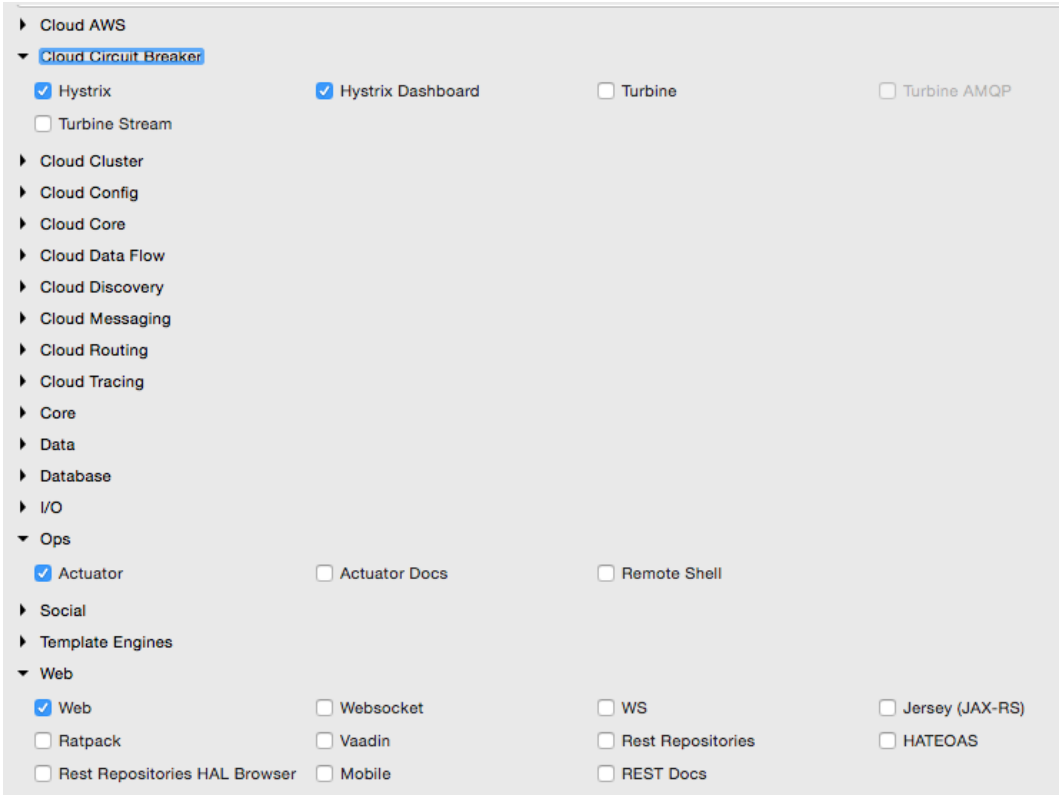

Configure an index pattern
In order to use Kibana you must configure at least one index pattern. Index patterns are used to identify the Elasticsearch index to run search and analytics against. They are also used to configure fields.
Index contains time-based events
Use event times to create index names
Index name or pattern
Patterns allow you to define dynamic index names using * as a wildcard. Example: logstash-*
logstash-*
Time-field name
refresh fields
@timestamp
Create


kibana
Discover
Visualize
Dashboard
Settings
Last 15 minutes
flightnumber
flightnumber
6 hits
Selected Fields
_source
Fields
Popular fields
message
@timestamp
@version
_id
_index
_type
host
port
March 31st 2017, 20:35:55.874 - March 31st 2017, 20:50:55.875
Count
3
2
1
0
20:36:00
20:37:00
20:38:00
20:39:00
20:40:00
20:41:00
20:42:00
20:43:00
20:44:00
20:45:00
20:46:00
20:47:00
20:48:00
20:49:00
20:50:00
@timestamp per 30 seconds
Time
_source
March 31st 2017, 20:46:55.918
message: {"@timestamp":"2017-03-31T20:46:55.904+04:00","@version":1,"message":"Flight [id=4, flightNUmber @version: 1 @timestamp: March 31st 2017, 20:46:55.918 host: 127.0.0.1 port: 51531 _source: {"message":"{\"@timestamp\":\"2017-03-31T20:46:55.904+04:00\",\"@version\":1,\"message\":\"Flight [id=4, flightNUmber=BF106, origin=NYC, destination=SFO, flightDate=22-JAN-18, fares=Fares [id=4, fare=106, currency=USD], inventory=Inventory [id=4, count=100]]\",\"logger_name\":\"com.brownfield.pss.search.Application\",\
March 31st 2017, 20:46:55.901
message: {"@timestamp":"2017-03-31T20:46:55.901+04:00","@version":1,"message":"Flight [id=3, flightNUmber @version: 1 @timestamp: March 31st 2017, 20:46:55.901 host: 127.0.0.1 port: 51531 _source: {"message":"{\"@timestamp\":\"2017-03-31T20:46:55.901+04:00\",\"@version\":1,\"message\":\"Flight [id=3, flightNUmber=BF105, origin=NYC, destination=SFO, flightDate=22-JAN-18, fares=Fares [id=3, fare=105, currency=USD], inventory=Inventory [id=3, count=100]]\",\"logger_name\":\"com.brownfield.pss.search.Application\",\

[Trace-Id=1, Span-Id = 1
Microservice 1
[Trace-Id=1, Span-Id = 2
Microservice 2
[Trace-Id=1, Span-Id = 3
Microservice 3


kibana
Discover
Visualize
Dashboard
Settings
Last 15 minutes
9f698f7ebabe6b83
logstash-*
Selected Fields
_source
Fields
Popular fields
message
@timestamp
@version
_id
_index
_type
host
port
13 hits
March 31st 2017, 22:18:37.675 - March 31st 2017, 22:33:37.675
Count
@timestamp per 30 seconds
Time
_source
March 31st 2017, 22:30:17.781
March 31st 2017, 22:30:17.742


Microservice
collector
JRE
collector
Server / NW
collector
Metrics sources and collectors
Metrics Aggregators and Correlations
Group by Service Id
[no of instances, throughput]
Metrics Processing
historical data
Dashboards and Alerts
CPU[%]
0 100


Cloud AWS
Cloud Circuit Breaker
Hystrix
Hystrix Dashboard
Turbine
Turbine AMQP
Turbine Stream
Cloud Cluster
Cloud Config
Cloud Core
Cloud Data Flow
Cloud Discovery
Cloud Messaging
Cloud Routing
Cloud Tracing
Core
Data
Database
I/O
Ops
Actuator
Actuator Docs
Remote Shell
Social
Template Engines
Web
Web
Websocket
WS
Jersey (JAX-RS)
Ratpack
Vaadin
Rest Repositories
HATEOAS
Rest Repositories HAL Browser
Mobile
REST Docs

Hystrix Dashboard

http://localhost:8095/hystrix.stream

Cluster via Turbine (default cluster): http://turbine-hostname:port/turbine.stream
Cluster via Turbine (custom cluster): http://turbine-hostname:port/turbine.stream?cluster=[clusterName]
Single Hystrix App: http://hystrix-app:port/hystrix.stream

Delay: 2000 ms Title: SearchAPIGateway

Monitor Stream

Hystrix Stream: SearchAPIGateway

Circuit

Sort: Error then Volume | Alphabetical | Volume | Error | Mean | Median | 90 | 99 | 99.5

Success | Short-Circuited | Timeout | Rejected | Failure | Error %

getHub

6 0 0 0 0.0 %

Host: 0.6/s
Cluster: 0.6/s
Circuit Closed

Hosts	1	90th	16ms
Median	12ms	99th	19ms
Mean	12ms	99.5th	19ms

Thread Pools

Sort: Alphabetical | Volume |

SearchAPIGatewayComponent

Host: 0.6/s
Cluster: 0.6/s

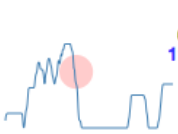
Active	0	Max Active	1
Queued	0	Executions	6
Pool Size	10	Queue Size	5

Hystrix Stream: SearchAPIGateway

HYSTRIX
DEFEND YOUR APP

Circuit Sort: Error then Volume | Alphabetical | Volume | Error | Mean | Median | 90 | 99 | 99.5

Success | Short-Circuited | Timeout | Rejected | Failure | Error %



getHub

0 | 0 | 100.0 %
11 | 0
1

Host: 1.2/s
Cluster: 1.2/s
Circuit Open

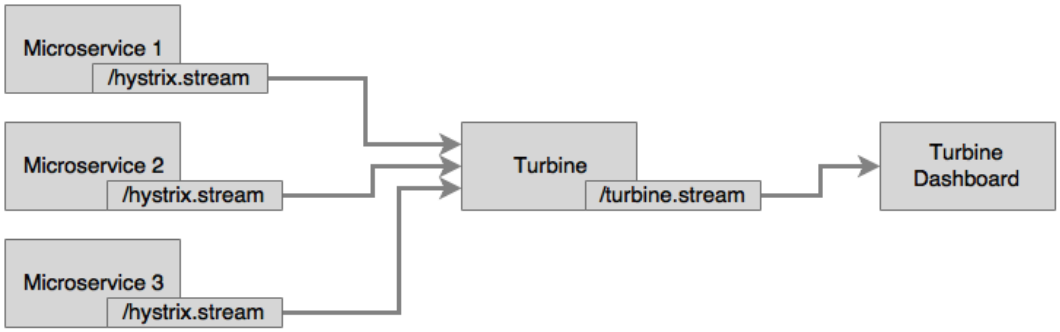
Hosts	1	90th	0ms
Median	0ms	99th	0ms
Mean	0ms	99.5th	0ms

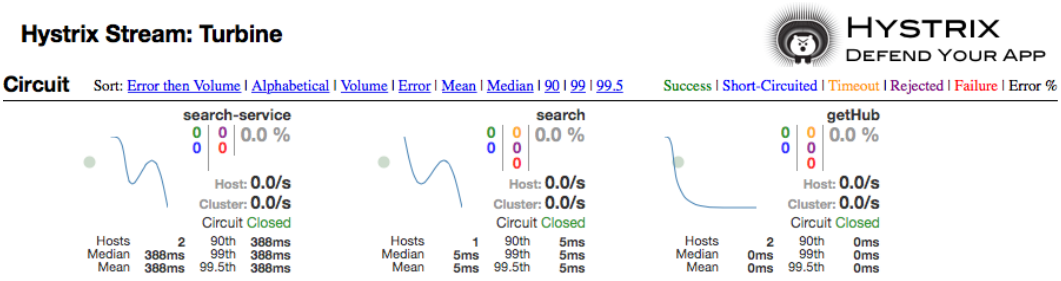
Thread Pools Sort: Alphabetical | Volume |

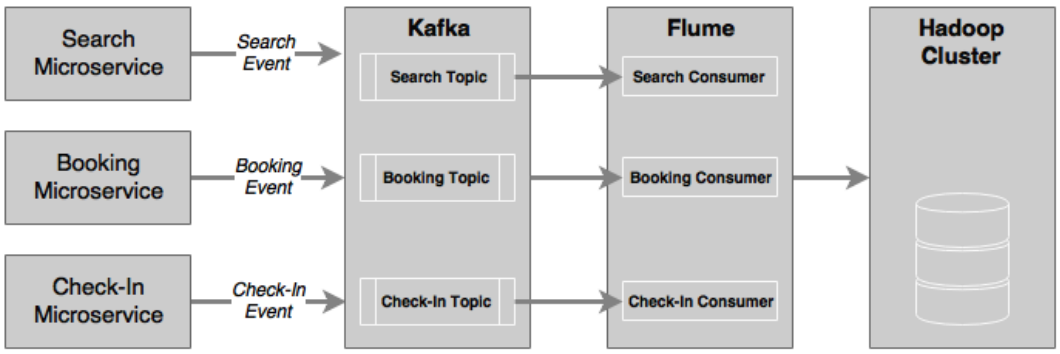
SearchAPIGatewayComponent

Host: 0.1/s
Cluster: 0.1/s

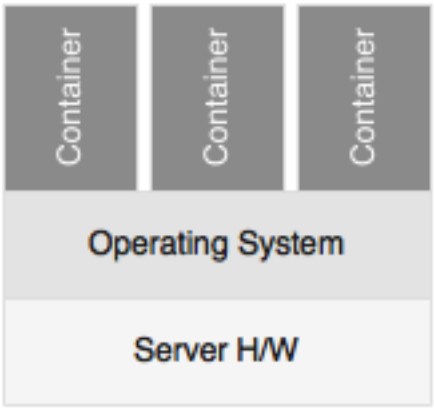
Active	0	Max Active	1
Queued	0	Executions	1
Pool Size	10	Queue Size	5

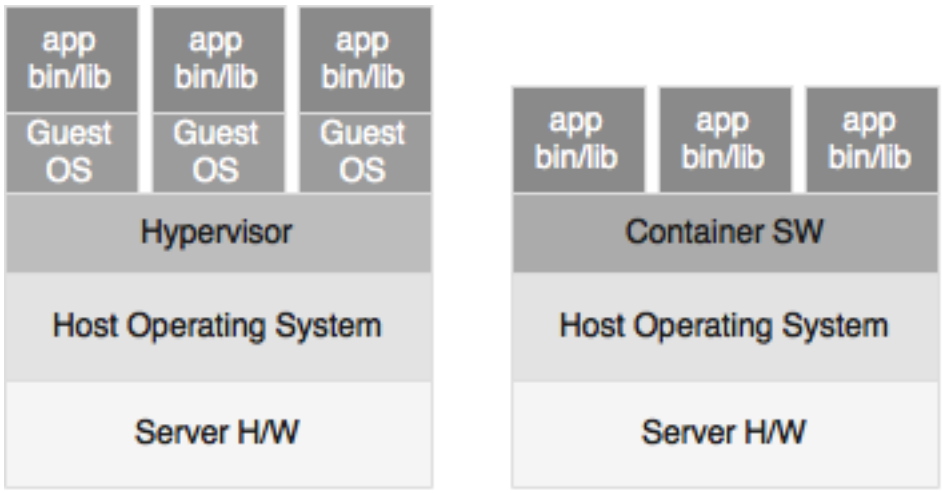



Hystrix Stream: Turbine
HYSTRIX
DEFEND YOUR APP
Circuit
Sort: Error then Volume | Alphabetical | Volume | Error | Mean | Median | 90 | 99 | 99.5
Success | Short-Circuited | Timeout | Rejected | Failure | Error %
search-service
0.0 %
Host: 0.0/s
Cluster: 0.0/s
Circuit Closed
Hosts 2
Median 388ms
Mean 388ms
90th 388ms
99th 388ms
99.5th 388ms
search
0.0 %
Host: 0.0/s
Cluster: 0.0/s
Circuit Closed
Hosts 1
Median 5ms
Mean 5ms
90th 5ms
99th 5ms
99.5th 5ms
getHub
0.0 %
Host: 0.0/s
Cluster: 0.0/s
Circuit Closed
Hosts 2
Median 0ms
Mean 0ms
90th 0ms
99th 0ms
99.5th 0ms
Thread Pools
Sort: Alphabetical | Volume |
SearchComponent
Host: 0.0/s
Cluster: 0.0/s
Active 0
Queued 0
Pool Size 2
Max Active 0
Executions 0
Queue Size 5
SearchAPIGatewayComponent
Host: 0.0/s
Cluster: 0.0/s
Active 0
Queued 0
Pool Size 5
Max Active 0
Executions 0
Queue Size 5


Search Microservice
Search Event
Booking Microservice
Booking Event
Check-In Microservice
Check-In Event
Kafka
Search Topic
Booking Topic
Check-In Topic
Flume
Search Consumer
Booking Consumer
Check-In Consumer
Hadoop Cluster

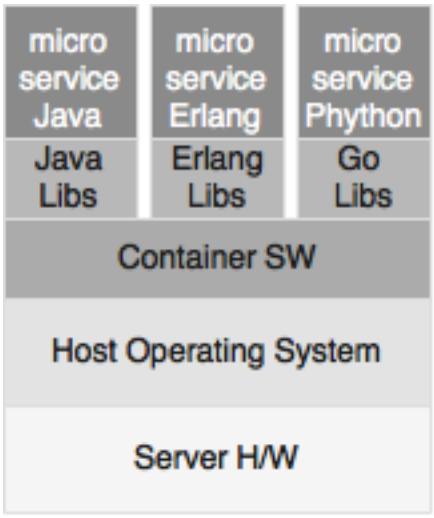
Chapter 9: Containerizing Microservices with Docker

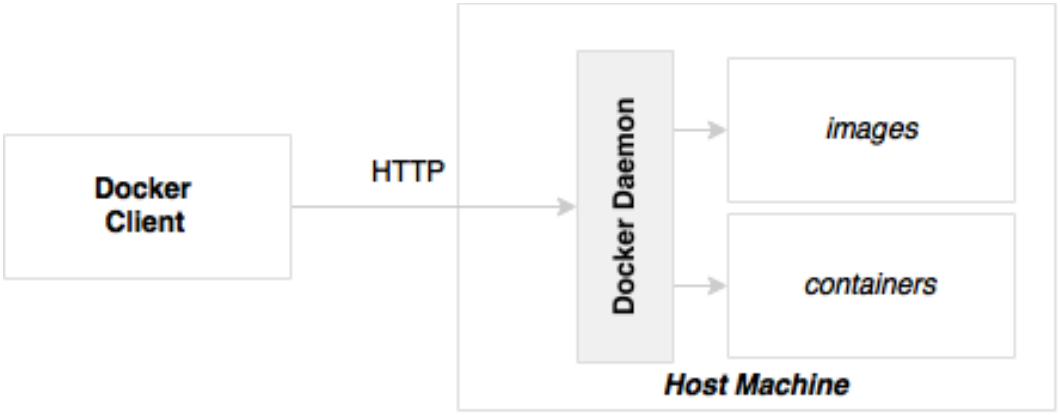




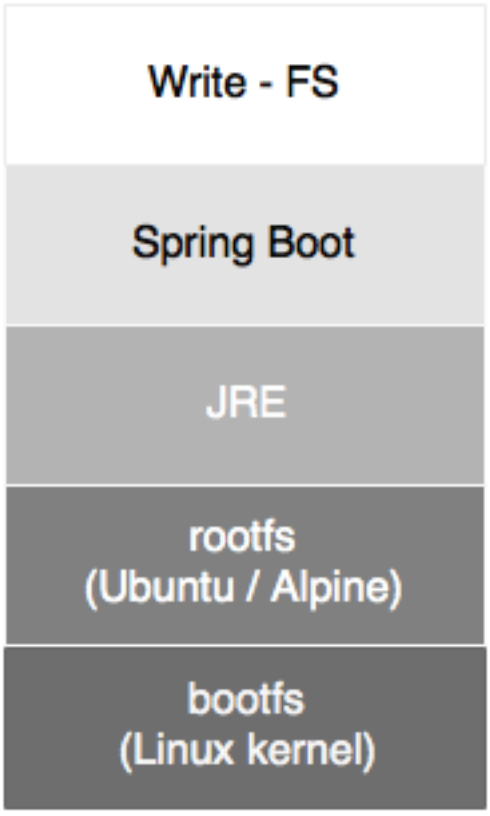
A) Virtual Machine Stack

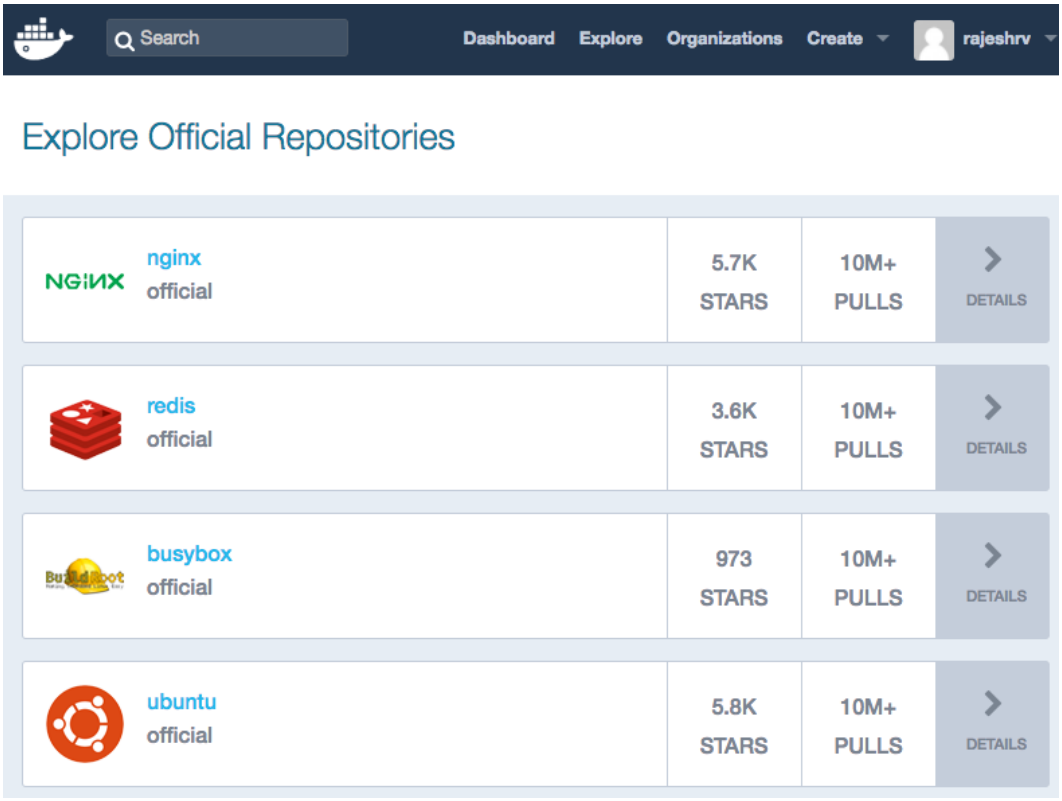
B) Container Stack


micro service Java
micro service Erlang
micro service Phython
Java Libs
Erlang Libs
Go Libs
Container SW
Host Operating System
Server H/W


Docker Client
HTTP
Docker Daemon
images
containers
Host Machine

Spring Boot
Application Jar
Java Runtime
Operating System


Write - FS
Spring Boot
JRE
rootfs
(Ubuntu / Alpine)
bootfs
(Linux kernel)

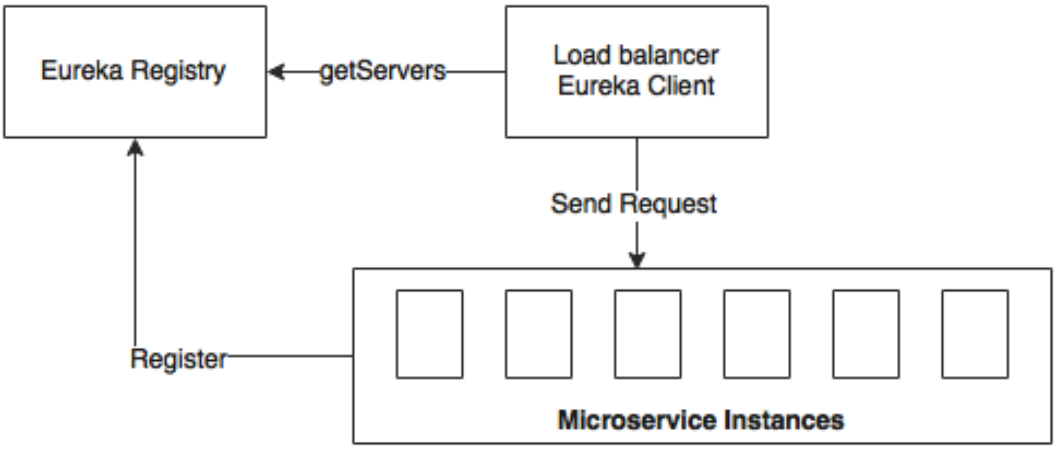

Search
Dashboard
Explore
Organizations
Create
rajeshrv
Explore Official Repositories
nginx
official
5.7K
STARS
10M+
PULLS
DETAILS
redis
official
3.6K
STARS
10M+
PULLS
DETAILS
busybox
official
973
STARS
10M+
PULLS
DETAILS
ubuntu
official
5.8K
STARS
10M+
PULLS
DETAILS

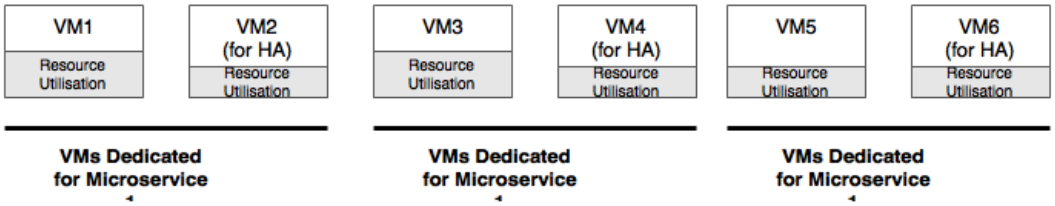
```
rvslab:chapter9.search rajeshrv$ docker build -t search:1.0 .
Sending build context to Docker daemon 57.01 MB
Step 1/5 : FROM frolvlad/alpine-oraclejdk8
latest: Pulling from frolvlad/alpine-oraclejdk8
627beaf3eaaf: Pull complete
95a531c0fa10: Pull complete
b03e476748e7: Pull complete
Digest: sha256:8ad40ff024bff6df43e3fa7e7d0974e31f6b3f346c666285e275afee72c74fcd
Status: Downloaded newer image for frolvlad/alpine-oraclejdk8:latest
 ---> f656c77f5536
Step 2/5 : VOLUME /tmp
 ---> Running in c816f2b47568
 ---> e6028f6a76bc
Removing intermediate container c816f2b47568
Step 3/5 : ADD target/search-1.0.jar search.jar
 ---> 39f28a242676
Removing intermediate container b4463e6220fc
Step 4/5 : EXPOSE 8090
 ---> Running in 2c25a35d20ea
 ---> d0738b1fb63a
Removing intermediate container 2c25a35d20ea
Step 5/5 : ENTRYPOINT java -jar /search.jar
 ---> Running in 095e60d75f13
 ---> a9f2ae1252c2
Removing intermediate container 095e60d75f13
Successfully built a9f2ae1252c2
```

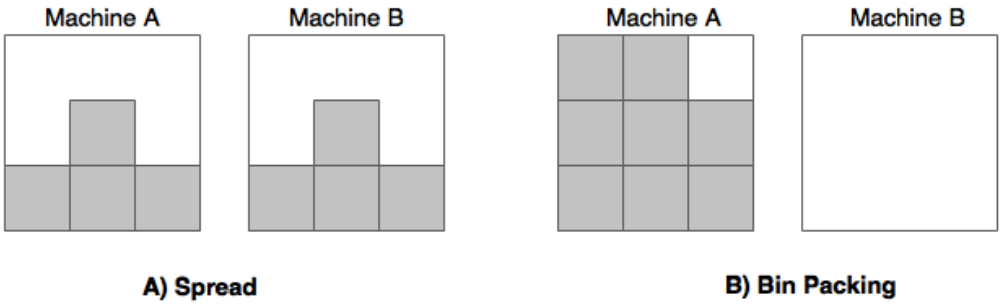
```
rvslab:chapter9.website rajeshrv$ docker images
REPOSITORY                    TAG                 IMAGE ID            CREATED             SIZE
book                          1.0                 8c0dbbe5ffc4        11 minutes ago      203 MB
checkin                       1.0                 2ee2d759fecd        12 minutes ago      209 MB
fares                         1.0                 13275668b4ea        12 minutes ago      202 MB
website                       1.0                 b4b3c7d59ff8        13 minutes ago      187 MB
search                        1.0                 8f42cd4d1f86        13 minutes ago      203 MB
```

```
rvslab:chapter9.website rajeshrv$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                    NAMES
9a11b3478e28        book:1.0            "java -jar /book.jar"    36 seconds ago      Up 34 seconds       0.0.0.0:8060->8060/tcp   hardcore_booth
2e629addd32b        website:1.0         "java -jar /websit..."   12 minutes ago      Up 12 minutes       0.0.0.0:8001->8001/tcp   boring_mcclintock
bf0b4436a387        checkin:1.0         "java -jar /checki..."   12 minutes ago      Up 12 minutes       0.0.0.0:8070->8070/tcp   angry_darwin
02d9c291b8b8        fares:1.0           "java -jar /fares.jar"   13 minutes ago      Up 13 minutes       0.0.0.0:8080->8080/tcp   affectionate_sinoussi
8a40e771dbe2        search:1.0          "java -jar /search..."   14 minutes ago      Up 14 minutes       0.0.0.0:8090->8090/tcp   loving_edison
```

Chapter 10: Scaling Dockerized Microservices with Mesos and Marathon

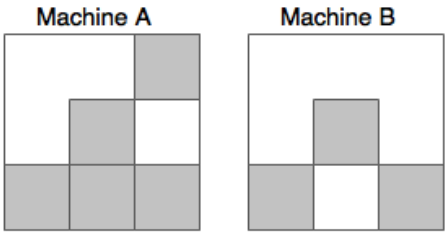




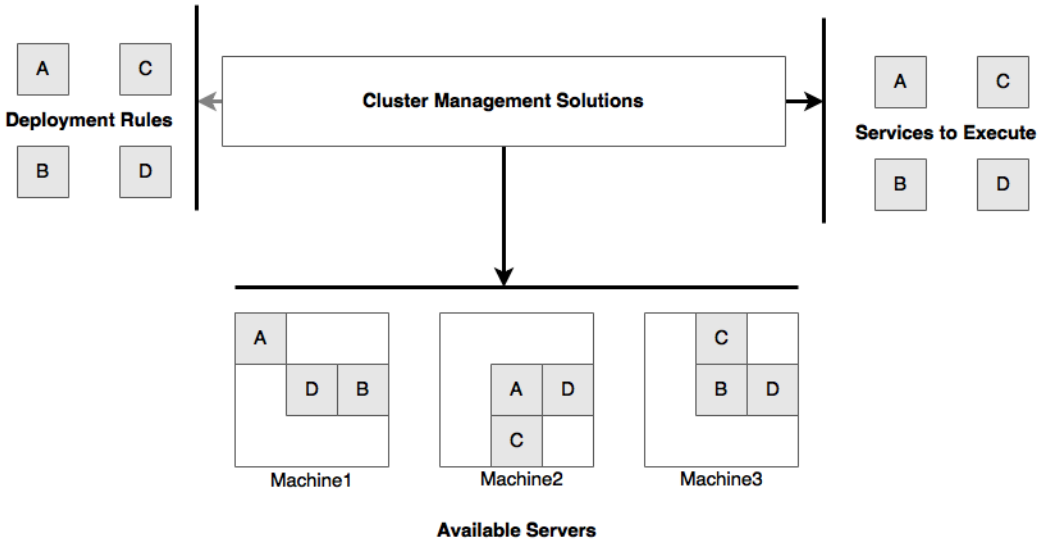


A) Spread

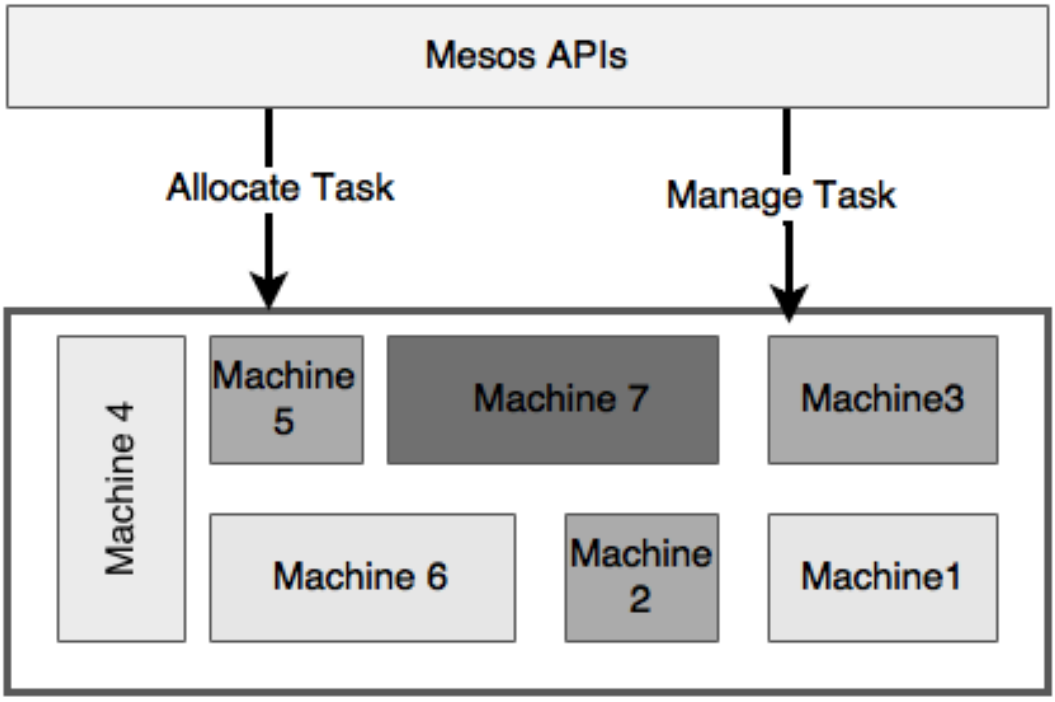
B) Bin Packing



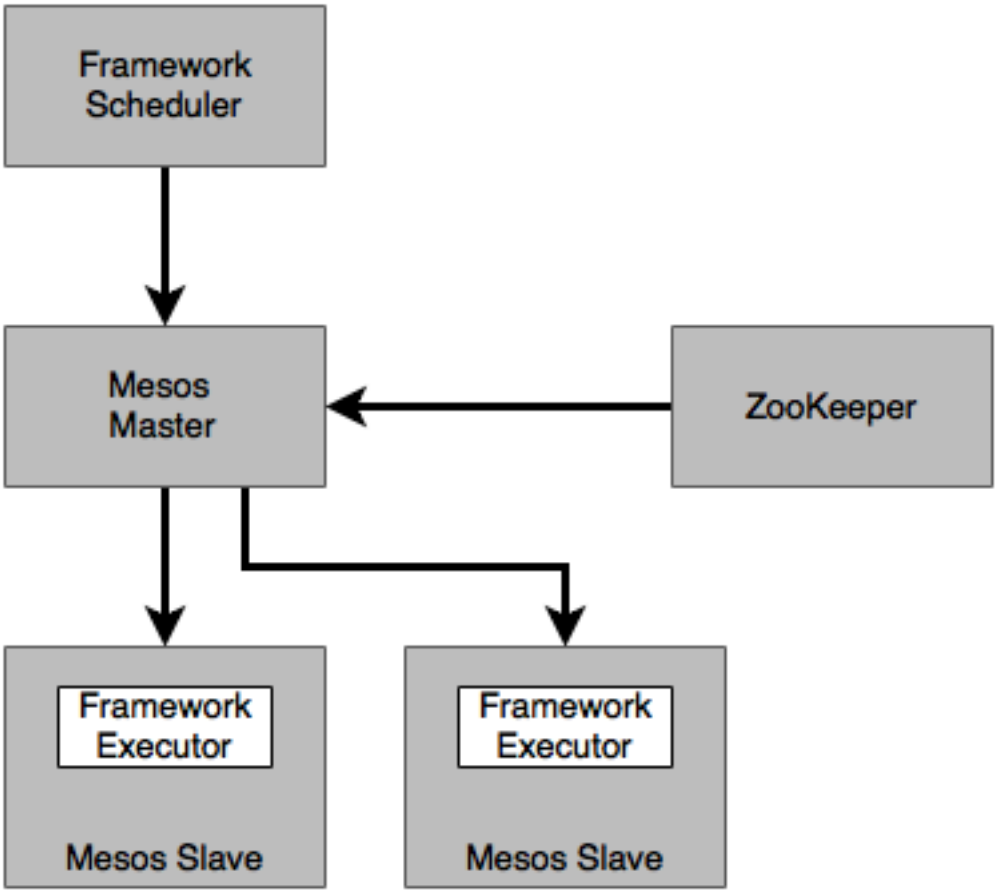
C) Random

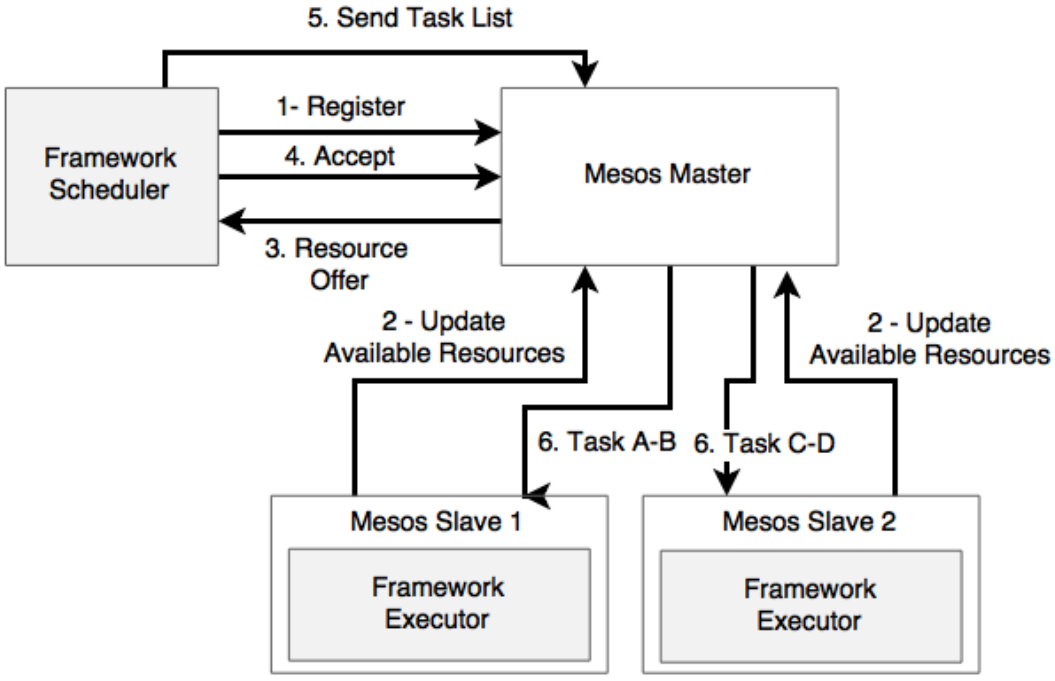


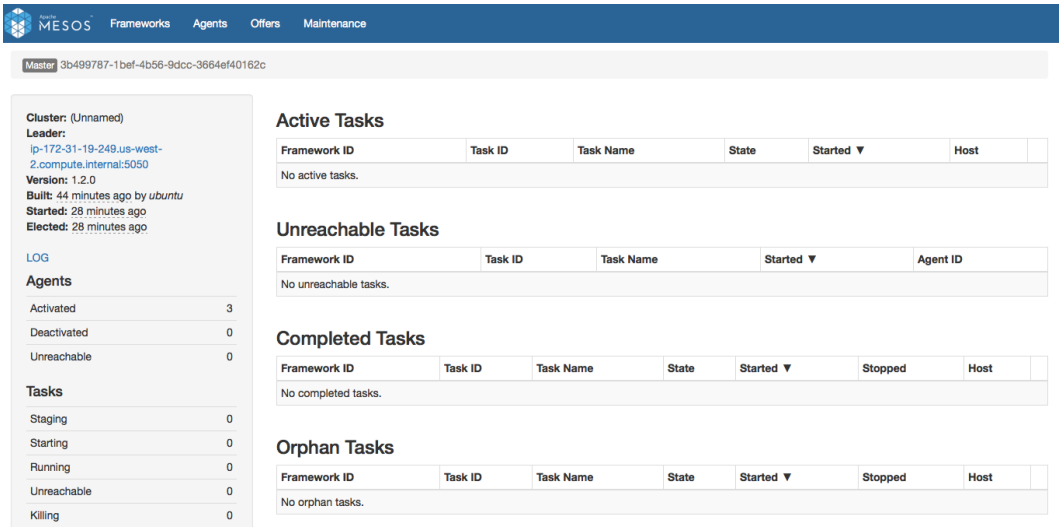
Available Servers

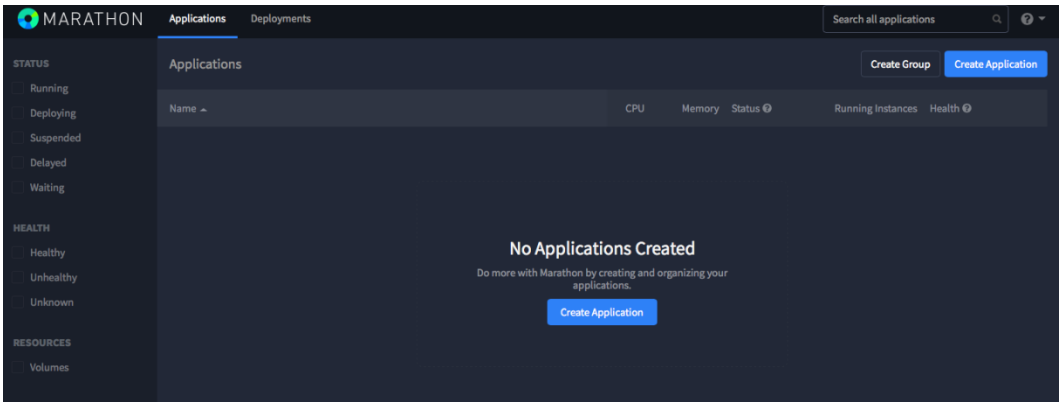


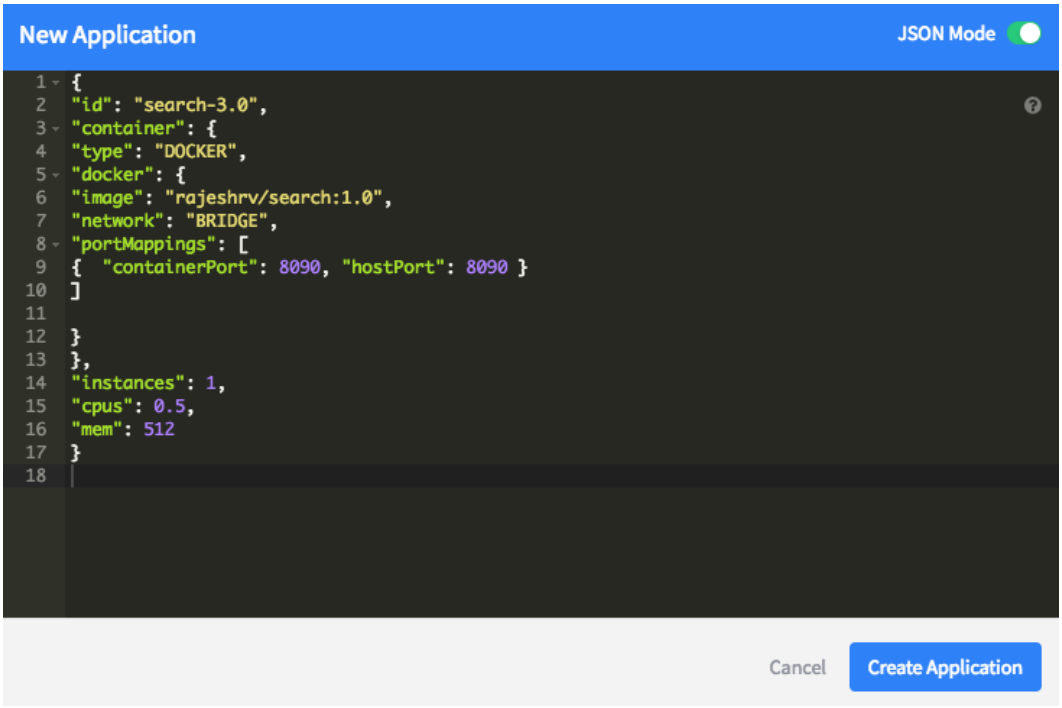
Cluster As A Single Machine

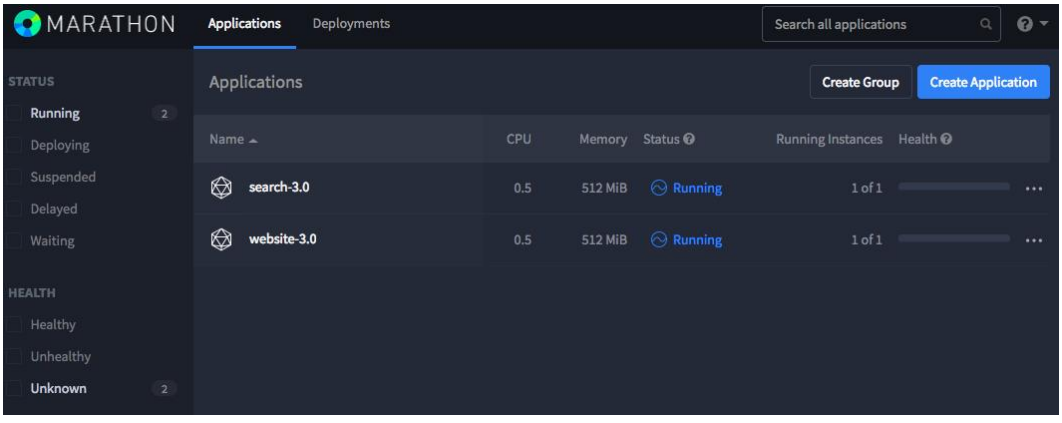

Framework
Scheduler
Mesos
Master
ZooKeeper
Framework
Executor
Mesos Slave
Framework
Executor
Mesos Slave

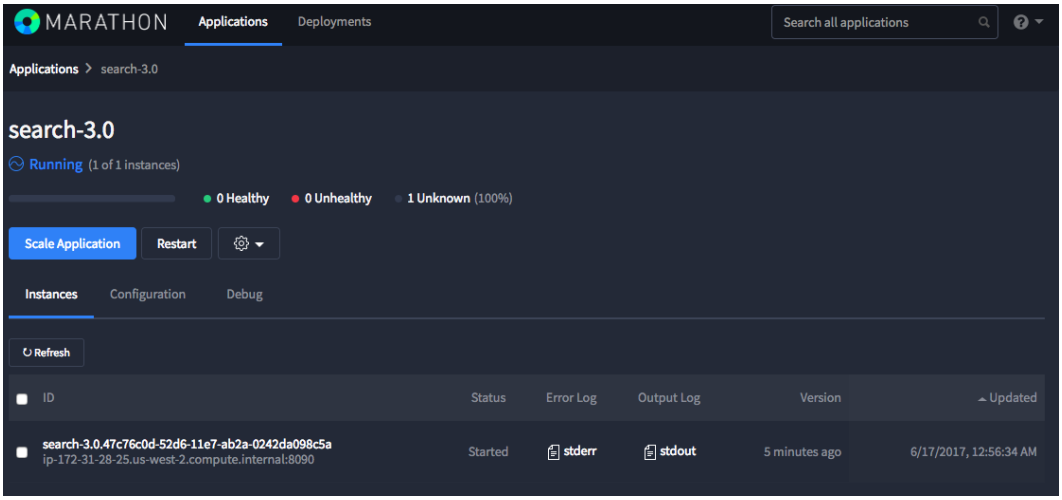

5. Send Task List
Framework Scheduler
1- Register
4. Accept
3. Resource Offer
Mesos Master
2 - Update Available Resources
2 - Update Available Resources
6. Task A-B
6. Task C-D
Mesos Slave 1
Framework Executor
Mesos Slave 2
Framework Executor


MESOS
Frameworks
Agents
Offers
Maintenance
Master 3b499787-1bef-4b56-9dcc-3664ef40162c
Cluster: (Unnamed)
Leader: ip-172-31-19-249.us-west-2.compute.internal:5050
Version: 1.2.0
Built: 44 minutes ago by ubuntu
Started: 28 minutes ago
Elected: 28 minutes ago
LOG
Agents
Activated 3
Deactivated 0
Unreachable 0
Tasks
Staging 0
Starting 0
Running 0
Unreachable 0
Killing 0
Active Tasks
Framework ID | Task ID | Task Name | State | Started ▼ | Host
No active tasks.
Unreachable Tasks
Framework ID | Task ID | Task Name | Started ▼ | Agent ID
No unreachable tasks.
Completed Tasks
Framework ID | Task ID | Task Name | State | Started ▼ | Stopped | Host
No completed tasks.
Orphan Tasks
Framework ID | Task ID | Task Name | State | Started ▼ | Stopped | Host
No orphan tasks.

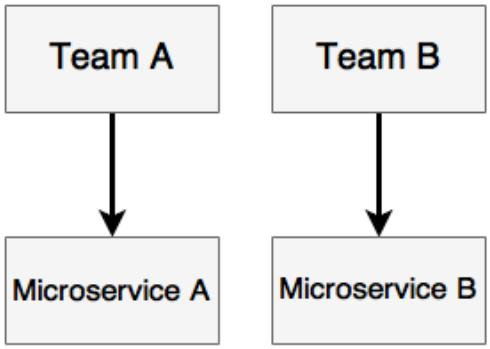

MARATHON
Applications
Deployments
Search all applications
STATUS
Running
Deploying
Suspended
Delayed
Waiting
HEALTH
Healthy
Unhealthy
Unknown
RESOURCES
Volumes
Applications
Create Group
Create Application
Name
CPU
Memory
Status
Running Instances
Health
No Applications Created
Do more with Marathon by creating and organizing your applications.
Create Application


New Application
JSON Mode
{
"id": "search-3.0",
"container": {
"type": "DOCKER",
"docker": {
"image": "rajeshrv/search:1.0",
"network": "BRIDGE",
"portMappings": [
{ "containerPort": 8090, "hostPort": 8090 }
]
}
},
"instances": 1,
"cpus": 0.5,
"mem": 512
}
Cancel
Create Application

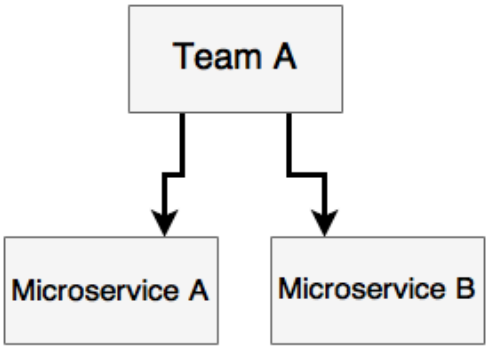

MARATHON
Applications
Deployments
Search all applications
STATUS
Running 2
Deploying
Suspended
Delayed
Waiting
HEALTH
Healthy
Unhealthy
Unknown 2
Applications
Create Group
Create Application
Name
CPU
Memory
Status
Running Instances
Health
search-3.0 0.5 512 MiB Running 1 of 1
website-3.0 0.5 512 MiB Running 1 of 1


MARATHON
Applications
Deployments
Search all applications
Applications > search-3.0
search-3.0
Running (1 of 1 instances)
0 Healthy
0 Unhealthy
1 Unknown (100%)
Scale Application
Restart
Instances
Configuration
Debug
Refresh
ID
Status
Error Log
Output Log
Version
Updated
search-3.0.47c76c0d-52d6-11e7-ab2a-0242da098c5a
ip-172-31-28-25.us-west-2.compute.internal:8090
Started
stderr
stdout
5 minutes ago
6/17/2017, 12:56:34 AM

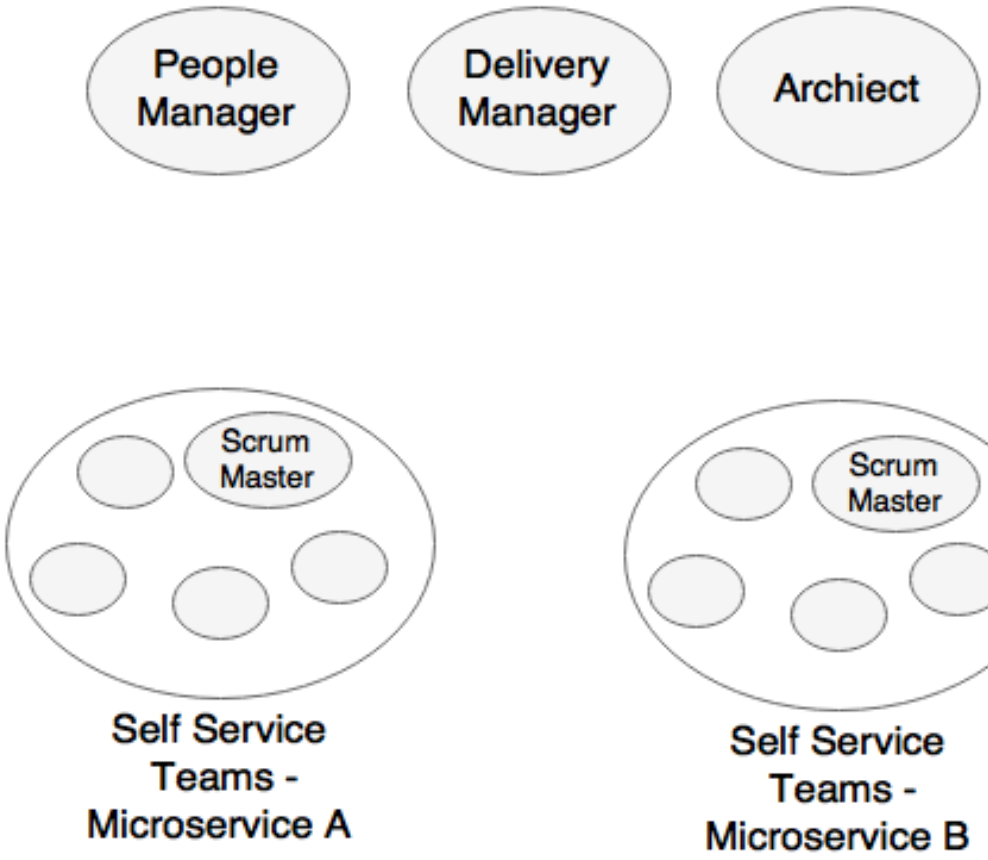
Chapter 11: Microservice Development Life Cycle

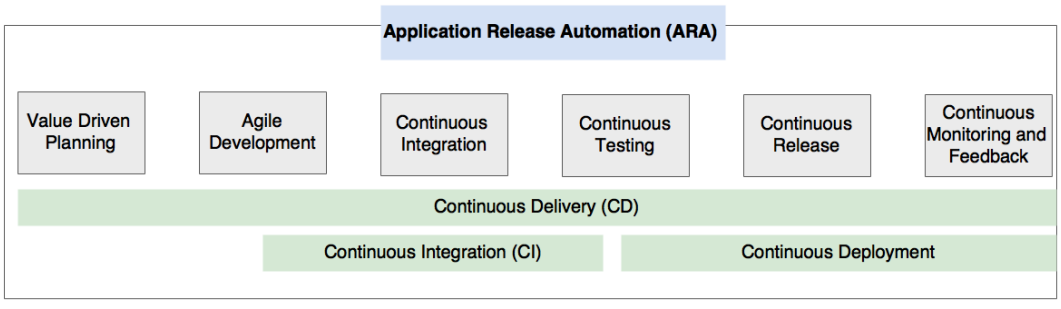


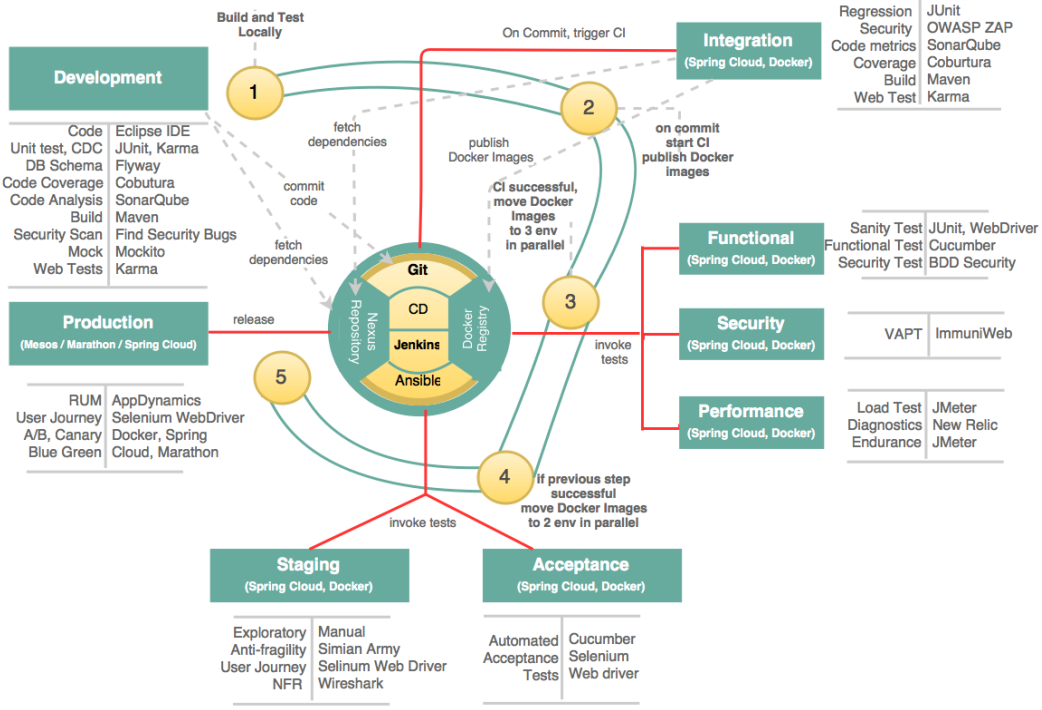
A) One team per microservice



A) One team per group of microservice representing related business capabilities


People Manager
Delivery Manager
Archiect
Scrum Master
Self Service Teams - Microservice A
Scrum Master
Self Service Teams - Microservice B


Application Release Automation (ARA)
Value Driven Planning
Agile Development
Continuous Integration
Continuous Testing
Continuous Release
Continuous Monitoring and Feedback
Continuous Delivery (CD)
Continuous Integration (CI)
Continuous Deployment


Build and Test Locally
On Commit, trigger CI
Development
Integration
(Spring Cloud, Docker)
Regression JUnit
Security OWASP ZAP
Code metrics SonarQube
Coverage Coburtura
Build Maven
Web Test Karma
1
2
Code Eclipse IDE
Unit test, CDC JUnit, Karma
DB Schema Flyway
Code Coverage Cobutura
Code Analysis SonarQube
Build Maven
Security Scan Find Security Bugs
Mock Mockito
Web Tests Karma
fetch dependencies
publish Docker Images
on commit start CI publish Docker images
commit code
CI successful, move Docker Images to 3 env in parallel
fetch dependencies
Git
CD
Jenkins
Ansible
Nexus Repository
Docker Registry
Functional
(Spring Cloud, Docker)
Sanity Test JUnit, WebDriver
Functional Test Cucumber
Security Test BDD Security
3
Production
(Mesos / Marathon / Spring Cloud)
release
Security
(Spring Cloud, Docker)
VAPT ImmuniWeb
invoke tests
5
RUM AppDynamics
User Journey Selenium WebDriver
A/B, Canary Docker, Spring
Blue Green Cloud, Marathon
Performance
(Spring Cloud, Docker)
Load Test JMeter
Diagnostics New Relic
Endurance JMeter
4
If previous step successful move Docker Images to 2 env in parallel
invoke tests
Staging
(Spring Cloud, Docker)
Acceptance
(Spring Cloud, Docker)
Exploratory Manual
Anti-fragility Simian Army
User Journey Selinum Web Driver
NFR Wireshark
Automated Acceptance Tests Cucumber
Selenium
Web driver