All-En-Route Deployment Architecture

1. Deployment overview

This **document** depicts **deployment guidelines** for the All-En-Route system. The All-En-Route system will be a **distributed system** with its **user applications communicating over internet with** its **geographically replicated core system**.
The document provides guidelines for deployment with example option of Kubernetes v1.28 cluster deployment in Oracle cloud native environment (OCNE) v1.7. The hardware components table in section 3.7 is also example sizings with example component cardinalaties for example option of Kubernetes v1.28 cluster deployment in OCNE v1.7 with generalized working procedures described in section 3.1 and 3.2. The actual cluster and cloud platform for staging and production deployment will be selected after evaluation of available options.

2. Deployment topology

The All-En-Route user application requests will be routed through load balancer and API gateway to appropriate All-En-Route core service. The load balancer will reside in dual firewall **DMZ** for optimal security.

The **load balancer** will distribute the traffic among the running instances of **API gateway**. The All-En-Route core service instances will run as containerized native cloud microservice instances to allow resources and cardinality configuration changes based on load once operational. Functionality of maintaining the desired configuration will be provided by the underlying cluster infrastructure.

The core services with respective data stores will run in multiple sites on cloud platform across the country. In each site, configured number of instances of each core service will be hosted as containers in **clustered cloud deployment**. The core services will expose REST API entry points and setup push notifications for client application interactions.

The **data stores** will also be initially sized based on estimated data movement. To support various types of data formats, **NoSQL database instances** will be configured for the data stores.

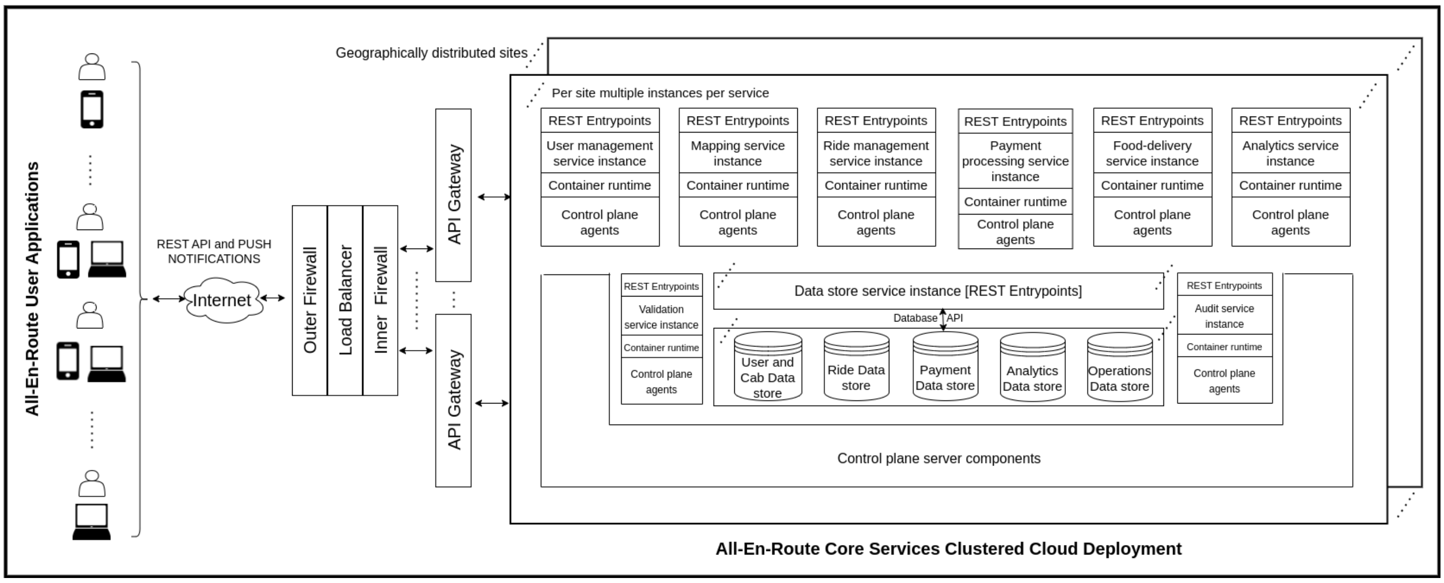
All-En-Route User Applications
REST API and PUSH NOTIFICATIONS
Internet
Outer Firewall
Load Balancer
Inner Firewall
API Gateway
API Gateway
Geographically distributed sites
Per site multiple instances per service
REST Entrypoints
User management service instance
Mapping service instance
Ride management service instance
Payment processing service instance
Food-delivery service instance
Analytics service instance
Container runtime
Control plane agents
Validation service instance
Data store service instance [REST Entrypoints]
Database API
User and Cab Data store
Ride Data store
Payment Data store
Analytics Data store
Operations Data store
Audit service instance
Control plane server components
All-En-Route Core Services Clustered Cloud Deployment

Fig 1: All-En-Route high level deployment architecture

2.1 System components

The All-En-Route **user applications** for various roles will be **mobile application** running natively as Android or iOS application on mobile device **and/or thin web client application** running on laptop, as described in “Section 2 of Software Architecture” document.

The **core system** will need to be **modular, secured, available anytime anywhere across** the **country** with **low latency, easy to maintain, easy to deploy** and **scalable**. Given these quality attribute requirements, the core system will need to follow service (**microservices**) oriented architecture. **For anticipated scale, operational ease, agility, flexibility** and **optimal resource utilization**, All-En-Route core services **will** need to **run in cluster in cloud environment**.

2.2 Scale and availability

Scalewise, **daily millions of active passengers, few hundred thousands of active cab providers,** and **few thousands of active operations staff** will be interacting with the core system. **For low latency, core system** will be **geographically replicated** in multiple designated locations. In **each location**, the **core system** will be **deployed in clustered environment in the cloud**. To manage the **scale** requirements, the **maximum resources** and **number of instances per service** will be **configured initially** based **on estimated load** and **later** will be **tuned for actual load**.

The **availability** of the service resources, instances and dependencies configured in the cluster configuration will be **managed and maintained by underlying cluster infrastructure running the microservices**. For instance, if the services are deployed as containers in kubernetes (k8s) cluster, the k8s control plane will manage and maintain service resources and instances configured for required concurrency and redundancy for high availability. The availability of 99.999% will be targeted using relevant hardware, software, monitoring, redundancy and diagnostic capability.

3. Deployment components

	Elements		
		Hardware	**Software**
Components	**Compute**	Compute nodes [Specification: vcpu cores and memory capacity]	Operating system Cluster software HTTP/HTTPS server software NoSQL DBMS software
	Storage	SAN storage [Specification: IO throughput and storage capacity]	Storage OS Storage configuration and management software Backup and restore software Disaster recovery (replication) software
	Network	Outer and inner firewall Load balancer API Gateway [Specification: Network bandwidth]	Network OS Firewall software Load Balancer software API Gateway software

Table 1: Core and paltform services (clustered cloud deployment) components per site location

3.1 Compute node hardware

For **production deployment**, per site location, initial compute node hardware sizing for **maximum number of vcpus** will depend on the maximum number of service instances to be run on the node. Initial maximum number of instances for a service will be arrived at using the estimated number of requests to be handled by the service per second at peak load and service throughput in terms of requests per second. Similary, compute node hardware sizing for **maximum memory capacity** will depend on the estimated peak load data processed and persisted by the service instances running on the node. Per site location, the service instances will also be distributed across the cluster nodes in the cloud to achieve required resilience. The underlying cluster technology will need to provide for this, for instance, k8s inherently supports this.

Staging deployment sizing will be scaled down version (if possible and/or required) of production deployment but large enough to cover the concurrency and redundancy and mimic the topological aspects of the production deployment from testing perspective.

For **development**, a single node deployment with one instance of each service will be made available. For **testing**, multinode (three node) deployment with multiple instances of each service will be made available.

3.2 Compute node software

Following **software elements** will need to run on each **compute node**:

- **Operating system software**: The compute nodes will run latest default OS distribution supported in preselected cloud platform. The OS installations will need to maintain the OS patch level and security patch levels as and when recommended by the cloud platform.
- **Cluster software**: The cluster software will need to be installed with predecided redundancy for master nodes and rest of the compute nodes will be worker nodes. The worker nodes will host All-En-Route microservices instances allocated by control software running on master nodes. For instance, in k8s cluster deployment, control software will run on predesignated master nodes and container runtime will run on all (master and worker) nodes.
- **HTTP/HTTPS server software**: This will be available in the REST API framework used by the application service.
- **NoSQL DBMS software**: The application data stores will be allocated required storage as configured in the cluster configuration. The NoSQL database to use will be predetermined after evaluating options (such as MongoDB or Cassandra). The DataStoreService will be used to abstract access to the NoSQL data store instances. The cluster infrastructure will provide for DNS based service discovery, IP tables based load balancing (round robin) and traffic distribution to data store replicas. The DataStoreService will interact with data stores using database APIs. The service level access tokens for the data store access will be managed using cluster configurations.

3.3 Storage hardware

Multipathed block level **SAN storage** will be used for data stores to provide resiliency and low latency IOs. The storage will be configured with RAID-6 (or at least RAID-5) for better performance and fault tolerance with reasonable resource provisioning.

The **sizing of storage hardware** will be initially based on the estimated data volumes (persisted), backup and replication requirements of the platform and will be tuned to actual volume once operational. The maximum throughput requirements will be determined based on the estimated collective maximum IO per second (IOPS) to be done by all the All-En-Route application services, backup and replication and will be tuned to actual IOPS once operational.

3.4 Storage software

Following **software elements** will need to run **on DataStoreService compute nodes**:

- **Storage configuration and management software**: The storage configuration and management software will be needed to configure, allocate and manage the application data stores and provide access control for the same.
- **Backup and restore software**: The data backup and restore software will need to support the Recovery Point Objective (RPO) and Recovery Time Objective (RTO) requirements for All-En-Route system. The periodic backup batch jobs will be configured by the system administrator.
- **Disaster recovery (replication) software**: The storage software will need to optionally support for the replication functionality in case customer plans to configure for disaster recovery. The periodic replication for data stores will be configured by the system administrator.

3.5 Network hardware

The core services will be deployed in multiple locations / sites (like capital city of each state). In each site, the internal network or **Virtual Private Cloud (VPC)** will be protected using dual (outer and inner) firewall **Demilitarized Zone (DMZ)**.

- **Outer and inner firewall**: The outer firewall secures ingress from internet to DMZ and inner firewall isolates VPC from DMZ.
- **Load Balancer**: The load balancer will expose the public IP from the DMZ through outer firewall. To avoid being a single point of failure, it will be configured in active standby setup.
- **API Gateway**: The API gateway will act as reverse proxy. For concurrency and to avoid being a single point of failure, it will be deployed with at least two replicas for high availability.

Depending on the initial estimates of the ingress of user application requests and push notifications, **network bandwidth** required will be determined and tuned to actual load once operational.

3.6 Network software

Following **software elements** will need to run on network hardware:

- **Network OS**: Each network appliance will run supported default network OS to provide required framework for its designated functionality.
- **Firewall software**: The firewalls enhance security by implementing multiple security checkpoints and controls for incoming and outgoing traffic for VPC.
- **Load Balancer software**: The load balancer will provide traffic distribution with predefined policy such as round robin, weighted round robin or least connections.
- **API Gateway software**: The API gateway will provide API routing and security to VPC against various online threats and attacks, rate limiting, caching, and audit logging functionality.

The networking configuration for VPC will be managed by the cluster infrastructure.

3.7 Hardware components

The hardware components table in this section is example sizings with example component cardinalaties for example option of kubernetes v1.28 cluster deployment in OCNE v1.7 with generalized working procedures described in section 3.1 and 3.2.

S. No.	Component ID	Component Name	Count	Component description
1	FW	Firewall	4	Outer firewalls (1 active, 1 standby) Inner firewalls (1 active, 1 standby) Example: F5 Big-IP iSeries Firewall Secure VPC from internet
2	LB	Load balancer	2	Load balancer (1 active, 1 standby) Example: F5 Big-IP iSeries Load balancer Distribute incoming traffic
3	AG	API Gateway	2	API gateway (2 instances for concurrency) Example: F5 Big-IP iSeries API Gateway Route the API calls to appropriate services

4	CCN-UM CCN-MS CCN-RM CCN-PP CCN-FD CCN-AN CCN-DS CCN-AS CCN-VS	Container compute nodes	30	Service container compute nodes: 10 service instances for data store service (2 instances per data store) 3 service instances (containers) for each of mapping service (CCN-MS), ride management (CCN-RM), payment processing (CCN-PP), food-delivery (CCN-FD) 2 service instances for each of validation (CCN-VS), analytics (CCN-AN), user management (CCN-UM) and audit (CCN-AS) Example: Kubernetes worker node in OCNE Specification: 1 CPU cores (Intel VT-capable CPU), 8GB RAM, 1GB Ethernet NIC XFS file system (the default file system for Oracle Linux) 15GB hard disk space in the /var directory XFS mount-point /var/lib/containers Container compute nodes run application core service instances in the containers
5	CM	Cluster master nodes	2	Example: Kubernetes master node (control plane node) in OCNE 4 CPU cores (Intel VT-capable CPU), 16GB RAM, 1GB Ethernet NIC XFS file system (the default file system for Oracle Linux) 40GB hard disk space in the /var directory Cluster master nodes run the cluster control plane components
6	STOR	Data store storage	5	Example: Seagate Exos CORVAULT 4U106 and IBM SAN32C-6 SAN Switch Storage for data stores

Table 2: Hardware components

3.8 Application Software Components

S. No.	Component ID	Component Name	Installation H/W Component ID	Open source or proprietary?	Component description
1	UMSC	User management service container	CCN-UM	Proprietary	User management service User account and profile management
2	MSC	Mapping service container	CCN-MS	Proprietary	Mapping service Locations, routes and ETA information
3	RMSC	Ride management service container	CCN-RM	Proprietary	Ride management service Ride create, assignment, start, finish, cancel
4	PPSC	Payment processing service container	CCN-PP	Proprietary	Payment processing service Payment transaction
5	FDSC	Food-delivery service container	CCN-FD	Proprietary	Food-delivery service Food stops and option listing and order placement
6	ANSC	Analytics service container	CCN-AN	Proprietary	Analytics service Basic and analytics charts configuration and management
7	DSSC	Data store service container	CCN-DS	Proprietary	Data store service Data store, retrieve, update and delete
8	ASC	Audit service container	CCN-AS	Proprietary	Audit service System activities logging

9	VSC	Validation service container	CCN-VS	Proprietary	Validation service Phone, email validations

Table 3: Application Software Components (for each deployment unit)

3.3 System Software Components

The system software table in this section is listing of softwares for example option of kubernetes v1.28 cluster deployment in OCNE v1.7 with generalized working procedures described in section 3.1 and 3.2.

S. No.	Component ID	Component Name	Installation H/W Component ID	Open source or proprietary?	Component description
1	CSS	Container system software	CCN-UM CCN-MS CCN-RM CCN-PP CCN-FD CCN-AN CCN-DS CCN-AS CCN-VS	Open source	HTTP/HTTPS server software (for REST API entrypoints) K8S Container runtime K8S Cluster control plane agents Oracle Linux v9 Provide the container runtime and control plane agent side functionality
2	FWS	Firewall software	FW	Proprietary	F5 BIG-IP software stack on F5 TMOS operating system Secure VPC from internet
3	LBS	Load balancer software	LB	Proprietary	F5 BIG-IP software stack on F5 TMOS operating system Distribute incoming traffic
4	AGS	API gateway software	AG	Proprietary	F5 BIG-IP software stack on F5 TMOS operating system Route the calls to approprate core service
5	CMS	Cluster master node software	CM	Open source	Kubernetes v1.28 Cluster infrastructure
6	STORSW	Data store storage	STOR	Proprietary	Example: Seagate Exos CORVAULT 4U106 drivers and management software and IBM SAN32C-6 SAN Switch drivers and management software SAN configuration and data store access

Table 4: System (Infrastructure) Software Components

4. Deployment process

The All-En-Route software **installables, upgrade and patches** (fixes) will be made available in the release software images repository for the system administrator to download. The software images will be downloaded and installed as per the All-En-Route **installation guide** following the documented **best practices**.
The installations, upgrades and patches will be **first applied in staging environment**. After testing it in staging for stipulated time, it will be **applied to production environment**.
The **best practices** for **setting up cluster configuration for required concurrency and redundancy** will be followed. The services will be upgraded in rolling fashion, by upgrading one instance at a time.

5. Security considerations

The user **RBAC** will be setup for predefined user accounts, users created by online registration or when system administrator configures system users. The data **encryption** will also be configured for user and cab data store and payment data store for securing data at rest. All the REST APIs will use https protocol for secured interaction and transactions. The security in the VPC will use **security tokens** for authentication and authorization for service entrypoint invocations and service-to-data-store communication.

All the users' personally identifiable information (**PII**) and **financial transactions** will be **encrypted** and **passwords** will be **one-way-hashed** in user application itself before being transfered on-the-wire and persisted at-rest in data stores in VPC for **security** and **privacy** reasons.

6. Monitoring and logging

The system administrator will setup cloud native or opensource (such as Prometheus) **monitoring tools for monitoring system health, performance and capacity**. All the components will **log the activities using common audit service** to facilitate system auditing and **diagnostics** when needed.

7. Disaster recovery and backup

The **data stores** will be **backed up periodically** and **replicated** to designated centralized locations **for disaster recovery** reasons. **Analytics functionality** will **operate** in central locations **on full data** for the **analytics across all sites**. The **knowledge base data** will also be **replicated** and merged across all sites for optimal issue resolutions.
The **periodic backup job**s and configurations for regular data backups will be setup by system administrator. The **restore procedures** and **disaster recovery strategies** and procedures for recovering from system failures or disasters will also be setup and configured as automated jobs to be run on demand or triggered on an event basis by system administrator.

8. References

[1] Software Systems Architecture
[2] Microservices Architecture
[3] Deployment Patterns in Microservices Architecture
[4] Oracle Cloud Native Environment
[5] BIG-IP System
[6] Exos E 4U106
[7] IBM Storage Networking SAN32C-6 Fabric Switch
[8] Microservices architecture on Azure Kubernetes Service
[9] Understanding RPO and RTO
[10] Data replication