Assessment of Monolith to Microservices Migration in AWS

Prepared by: Papu Bhattacharya
Date: 05/12/16

Revision History

Revision #	Description	Initials	Date
0.1 Draft	First Draft for Review	PB	03/11/2016
1.0	Final Document	PB	05/12/2016

Contents

1 INTRODUCTION 3

1.1 PURPOSE 3
1.2 BACKGROUND AND GOALS 3

2 BUSINESS READINESS 3

2.1 BUSINESS ENVIRONMENT 3
2.1.1 Business Transformation Challenges 3
2.1.2 Pain points 3
2.1.3 Risk tolerance 3
2.1.4 Resistance to change 4
2.2 BUSINESS IMPACT 4
2.2.1 Customer Experience 4
2.2.2 Revenue 4
2.2.3 Cost 4
2.2.4 Time to Market 4
2.3 BUSINESS PROCESS 4
2.3.1 Procurement 4
2.3.2 IT Process 5

3 EXISTING INFRA AND APPLICATION REVIEW 5

3.1 CURRENT APPLICATION ARCHITECTURE 5
3.2 360 DEGREE ANALYSIS 5
3.2.1 Scalability 6
3.2.2 High Availability 7
3.2.3 Management 8
3.2.4 Security 9
3.2.5 Performance 13
3.2.6 Backup & DR 14
3.3 EXPECTED KEY BENEFITS ON MIGRATION 15

4 MIGRATION PATH 15

4.1 ROADMAP 15
4.1.1 High Level Future State 15
4.1.1 Platform Architecture Future State 16
4.1.1 Project Execution Roadmap 16
4.2 PROPOSED SOLUTION ARCHITECTURE 17
4.2.1 Brief Microservice deployment Architecture 17
4.3 SECURITY & COMPLIANCE RISK REPORT 18
4.4 PROJECT PLAN & SOW FOR IMPLEMENTATION 18
4.4.1 Project Plan 18
4.5 DETAILED POC 18

1 Introduction

1.1 Purpose

The purpose of this document is to detail the review findings and plan for monolith to microservices readiness program for ABC CMS platform.

1.2 Background and Goals

There is a business need to **consolidate and centralize vehicle marketing data** in the vehicle domain and improve business processes by creating micro web services in AWS cloud platform. Our goal is to have a detail review and plan to asses:

1) Business Readiness
2) Existing Infrastructure and Applications Review
3) Migration Path

2 Business Readiness

2.1 Business Environment

2.1.1 Business Transformation Challenges

- Increase agility to accelerate business – Ability to bring changes rapidly.
- Increase optimization and be in control of costs – a need to do more with less spending
- Location Transparent – Make all business process location transparent.
- Reduce risk – the need to improve the availability, security governance, and the compliance of business support system (BSS) and operational support system (OSS) applications, and standardize the IT infrastructure.

2.1.2 Pain points

- No centralized source of truth for Vehicle Domain Marketing Content, Configuration Rules, & Assets.
- Data Governance – Multiple web services provides same data that serves multiple systems
- Current services are difficult to scale horizontally
- Public service with no service authorizations – cannot identify service consumers.
- Maintenance release has a long QA and regression test cycle and difficult to test every application functionality and scenario.
- Daily batch job to replicate data and manual process to update and import data
- UI Integration is too complex with client side JavaScript API.
- Significantly high capex on hardware and operation team

2.1.3 Risk tolerance

- There are GAPS in overall understanding of the microservices and its relevant technologies within the technical team. As a risk mitigation, we recommend the following:
    - Perform AS IS migration in cloud in first phase
    - Conduct trainings on cloud and Microservices
    - Perform microservices transformation in second phase

2.1.4 Resistance to change

- We observed cultural readiness towards cloud transformation amongst the stakeholders.

2.2 Business Impact

2.2.1 Customer Experience

- API Based, fully discoverable, centralized services for Vehicle Marketing Content, Rules for multiple languages and regions (GST/SET)
- Domain driven, container based micro services architecture leveraging fully native AWS Stack (Cloud first guiding principle)
- Robust and secure authentication and authorization via API Gateway.
- Establish Continuous Integration and Continuous Delivery Pipeline
- Eliminate monolithic nature of the EFC Platform
- Enable faster time to market to deliver new content and features quickly
- Simplify complexity of integration with restful API services. Single consumer integration point for marketing content and rules.
- Response time is deterministic within Performance SLA
- Scales horizontally to any workload
- Operate in auto pilot mode, maximum automation achieved

2.2.2 Revenue

- Elastic infrastructure, grows and shrink with demand, it will reduce 60% existing hardware expenses.
- The Automation effort would reduce 80% operation expense.
- The proposed architecture has no assumption of software license, this will reduce 100% license cost
- It will deliver 10X agility, 5X reliability, and 10X operational efficiency

2.2.3 Cost

- The complete migration project would take 60-man months effort. (Detail effort estimation sheet attached)
- TCO will be reduced to 70%, because of 80% operation cost, 60% Hardware Cost, 100% License cost saving.

2.2.4 Time to Market

- It will reduce the current maintenance release cycle (1 month) to 1 week.

2.3 Business Process

2.3.1 Procurement

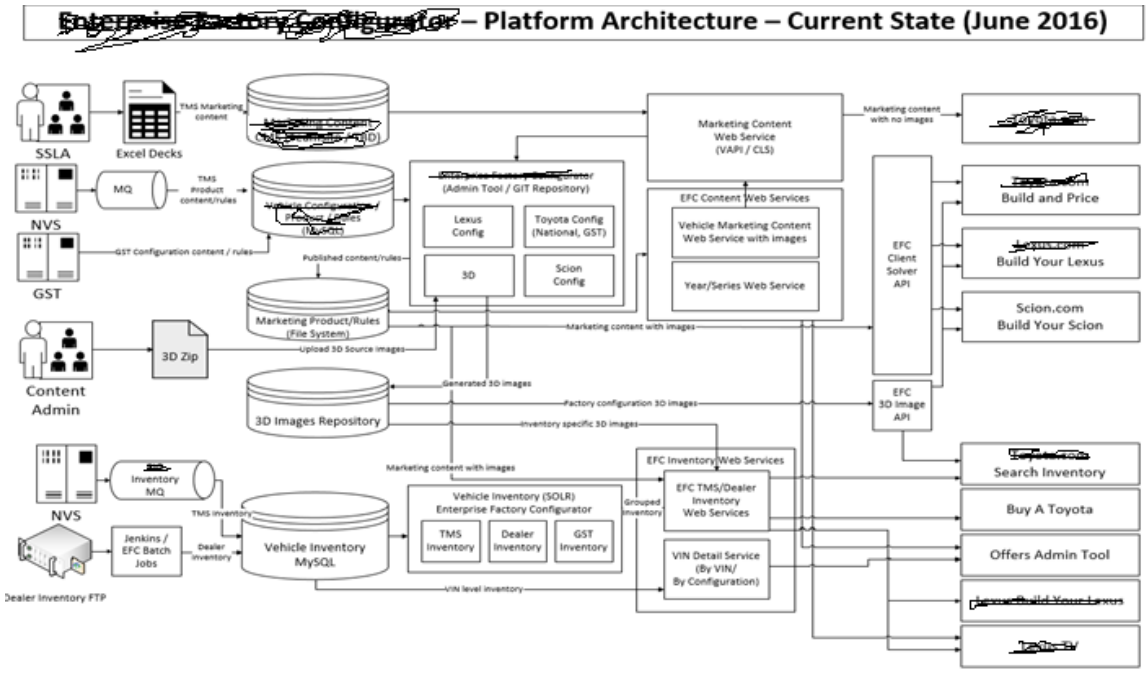
- A central team(DEIS) is in place that create the AWS accounts for product team, the tests and POC's are done in XYZ account. So, there won't be any major procurement cycle.

2.3.2 IT Process

- Procure AWS environment from DEIS team
- Use existing OSS tools setup for EFC product
- Complete the migration paths in AWS environment (As per the details plan)
- Transition the operations to DEIS team
-

3 Existing Infra and Application Review

3.1 Current Application Architecture



3.2 360 Degree Analysis

Y= Yes , N = Non Compliance , NA – Not Applicable or out of scope

3.2.1 Scalability

	Criterion	Compliance	Comment
Application Scalability	The application is service driven composed of stateless components	N	It's currently monolith – all services composed in single war
	The business logic uses an isolated / independent cache cluster	N	It uses internal L2 cache with ehcache
	Individual components can be scaled horizontally	N	It's monolith, bundled together
	The application uses a separate search / indexing cluster for search purpose	N	It uses solr , but not in separated clustered & elastic way
	The large data processing layer is parallel and horizontally scalable	N	It's processing a huge json datasets inside a single process in single node.
	The application uses the right combinations of SQL/NOSQL/CACHE/ GRAPH database	N	Since it manages contents inside git and git is managing database in filesystem, we may not need this.
	The application uses a queuing cluster to process async task management	N	The application uses quartz job engine on single node, does not use a queuing cluster for job processing.
	Separate database is maintained for transaction & reporting	N	Use same database
	The application stores document and content on a scalable distributed filesystem or inside a distributed CMS	N	Maintained in DAS SSD . Also, git filesystem is not distributed
	The application streams its real-time data to separate bigdata layer to process the analytics	N	Does not handle much real-time data

	The application uses a scalable api gateway cluster to handle api requests	N	API handling is internal
Storage Scalability	Storage is context Aware i.e for block/ document / object – optimal storage is designed. Having a content aware stoarge design -- for small files / DB blocks /Application objects / OS objects / READ sensitive / Write sensitive	NA	Out of scope
	At host level, optimal volume management practices are followed to support auto scaling and high density	NA	Out of scope
	Using distributed storage (lustre / CEPH) with appropriate node level fs - btrfs / zfs to scale out horizontally	NA	Out of scope
	Using cloud storage effectively to scale out	NA	Out of scope
Network Scalability	WAN/CORE/AGGREAGTION/ACCESS layer switch comply with proper design methodology to scale out	NA	Out of scope
	Service cluster network -- grow with vlan tagging	NA	Out of scope
	Hardware Load Balancer scaling	NA	Out of scope

3.2.2 High Availability

	Criterion	Compliance	Comment
Application Availability	The application is susceptible for n-node failover	N	Web & App server is currently active / active dual node , webserver has sticky session enabled. It's recommended to have a stateless n-node failover enabled
	The Datastore (database / file system) is distributed and susceptible for n-node failure	N	Database is dual node active /passive failover mode.
	The application and database comply to the disaster recovery requirement	N	It does not support site failover

3.2.3 Management

	Criterion	Compliance	Comment
Deployment	Supports Blue / Green Deployment of Application/ Databases without downtime	N	The upgrade requires a system level reboot
	Platform and infrastructure can be provisioned is software defined manner	N	Platform provisioning is primarily manual, with some automation for OS provisioning.
	OS update /patches /hardening process is automatic	Y	In place
	Stage wise promotion of build process is in place	N	
	Service orchestration templates are available through which a chain of services can be deployed and upgraded	NC	Quick build jobs deploy application component 1 by 1
Operation Support System	Monitoring Tool in place which can monitor the following: • Service cluster performance • Application services performance • Application availability • Database performance • Load balancer statistics • Network traffic • Physical server performance • Virtual Server performance • Storage performance	Y	Most of the metrices get collected through app dynamics tools
	Central Log Analysis Tools are present	Y	ELK is present
	Auto scaling of services and cluster are present	N	Scaling is manual
	Infrastructure Audit / Trails tools are present	N	Some security audit is in place, Trailing is absent
	Automation scripts are present which does regular and preventive maintenance in response to system nature	N	Most regular maintenance jobs are ticket based
	Monitoring tools generates right notification /alerts based on rules	Y	Present

	Application support resiliency through resilience architecture (circuit breaker sort of) or with some tools (like monit)	N	Not present
ALM	Integrated ALM tools are present for product development	Y	Atlassian ALM tools are in place with right level of integration
	Unit / Integration / Regression / Performance / Code security / Code quality testing is integrated in continuous integration environment	N	Performance & Code Security testing is not integrated with CI process
	The codebase can be easily managed and be extended	N	The GWT and JavaScript layer has become highly cohesive and it is difficult to manage and extend.

3.2.4 Security

	Criterion	Compliance	Comment
Application Security	• Application comply with the security checks below Transport— • Check SSL Version, Algorithms, Key length • Check for Digital Certificate Validity (Duration, Signature and CN) • Check credentials only delivered over HTTPS • Check that the login form is delivered over HTTPS • Check session tokens only delivered over HTTPS • Check if HTTP Strict Transport Security (HSTS) in use • Test ability to forge requests • Test Web Messaging (HTML5) • Check CORS implementation (HTML5) Data Validation: • Test for HTML Injection • Test for SQL Injection • Test for LDAP Injection • Test for ORM Injection • Test for XML Injection • Test for XXE Injection • Test for SSI Injection • Test for XPath Injection • Test for XQuery Injection	NA	Out of Scope

	• Test for IMAP/SMTP Injection • Test for Code Injection • Test for Expression Language Injection • Test for Command Injection • Test for NoSQL injection • Test for Reflected Cross Site Scripting • Test for Stored Cross Site Scripting • Test for DOM based Cross Site Scripting • Test for Cross Site Flashing • Test for Overflow (Stack, Heap and Integer) • Test for Format String • Test for incubated vulnerabilities • Test for HTTP Splitting/Smuggling • Test for HTTP Verb Tampering • Test for Open Redirection • Test for Local File Inclusion • Test for Remote File Inclusion • Compare client-side and server-side validation rules • Test for HTTP parameter pollution • Test for auto-binding • Test for Mass Assignment • Test for NULL/Invalid Session Cookie • Test for integrity of data • Test for the Circumvention of Work Flows • Test Defenses Against Application Mis-use • Test That a Function or Feature Cannot Be Used Outside Of Limits • Test for Process Timing • Test for Web Storage SQL injection (HTML5) • Check Offline Web Application Denial of Service • Test for anti-automation • Test for account lockout • Test for HTTP protocol DoS • Test for SQL wildcard DoS Authentication • Test password quality rules • Test remember me functionality • Test password reset and/or recovery • Test password change process • Test CAPTCHA • Test multi factor authentication • Test for logout functionality presence • Test for default logins • Test for out-of channel notification of account lockouts		

	and successful password changes • Test for consistent authentication across applications with shared authentication schema / SSO and alternative channels • Test for Weak security question/answer • Test for user enumeration • Test for authentication bypass • Test for brute force protection • Test for Credentials Transported over an Encrypted Channel • Test for cache management on HTTP (eg Pragma, Expires, Max-age) • Test for user-accessible authentication history Session Management • Establish how session management is handled in the application (eg, tokens in cookies, token in URL) • Check session tokens for cookie flags (httpOnly and secure) • Check session cookie scope (path and domain) • Check session cookie duration (expires and max-age) • Check session termination after a maximum lifetime • Check session termination after relative timeout • Check session termination after logout • Test to see if users can have multiple simultaneous sessions • Test session cookies for randomness • Confirm that new session tokens are issued on login, role change and logout • Test for consistent session management across applications with shared session management • Test for session puzzling • Test for CSRF and clickjacking Authorization: • Test for path traversal • Test for vertical Access control problems (a.k.a. Privilege Escalation) • Test for horizontal Access control problems (between two users at the same privilege level) • Test for missing authorization • Test for Insecure Direct Object Reference Vulnerable Functionality File Upload Related: • Test that acceptable file types are whitelisted and non-whitelisted types are rejected		

	• Test that file size limits, upload frequency and total file counts are defined and are enforced • Test that file contents match the defined file type • Test that all file uploads have Anti-Virus scanning in-place. • Test upload of malicious files • Test that unsafe filenames are sanitized • Test that uploaded files are not directly accessible within the web root • Test that uploaded files are not served on the same hostname/port • Test that files and other media are integrated with the authentication and authorization schemas Payments • Test for default or guessable password • Test for Injection vulnerabilities • Test for Buffer Overflows • Test for Insecure Cryptographic Storage • Test for Insufficient Transport Layer Protection • Test for Improper Error Handling • Test for all vulnerabilities with a CVSS v2 score > 4.0 • Test for Authentication and Authorization issues • Test for CSRF		
	API consumers access the API with multifactor authentication	Y	It uses app key and certificates
	Code & Libraries Security	Y	Verracode tools checks in place
Database Security	Check the application database from the following perspectives: • Access control • Auditing • Authentication • Encryption • Integrity controls • Backup Security • Tokenization with key server	NA	Out of Scope

Network Security	Tool based backbone network assessment for: • Network configuration backup • Standard based Compliance & Governance • Intrusion and threat protection • Vulnerability • Firewall and load balancer security	NA	Out of Scope
	Tool based audits of basics network security: • Private LAN traffic flow analysis • Isolation through appropriate Network Access Control List • DMZ and Non-DMZ isolation • Outbound / Inbound traffic in private network passes through NAT gateway		

3.2.5 Performance

	Criterion	Compliance	Comment
Application Performance	Key business functionalities meet business SLA's: Following are the lists of key functionalities- • Admin UI load • Bulk data upload • Indexing based series / model / year • Rule engine execution with composed rule •	N	The specified functionality did not meet business performance SLA's
	Application performance benchmark testing is done in continuous integration stage with appropriate load data set	N	Testing is done with load data set, but that is not integrated with CI testing.
	Application profiling is done at development environment as well as the data is available from production environment.	N	Microlevel functional profiling is absent
	Application static data is delivered through a geographically distributed cached cluster	N	Currently static data is not distributed through any CDN
Storage Performance	Read / write performance benchmarking is done for all storage (application and database storage) with appropriate load simulation	NA	Out of scope
	The backend storage network can transmit large data set within performance SLA	NA	Out of scope
	At host level, in case of TCP traffic, storage network interface is	NA	Out of scope

	separated from any other network (private/public/admin)		
	cloud storage gateways meet performance SLA's	NA	Out of scope

3.2.6 Backup & DR

	Criterion	Compliance	Comment
	Test that • A documented backup and recovery procedure exist • All critical data set is identified for backups • Appropriate backup type is defined for critical data • Appropriate backup schedule and window exist • Data storing mechanism is defined • Retention policy is defined • Automated backup exists with tools • Backup monitoring through tool exist • Delete obsolete backups to ensure optimal performance • Validate and verify backups without doing actual restores. • Archiving and staging process exists • Databases restore testing. • Validating restores on data integrity and performance • Is there any critical data that is ONLY stored on-site? (no off-site backup) • Utilize off-site storage • periodically perform test restores of critical data • Are on-site backups protected from fire, flood, theft, etc? • Are redundant copies of data kept mitigating loss? • Can backups be remotely accessed? • backing up done at the disk block level, or only files & data • Identified data that should not be included in the backup process • keep redundant copies of data • Appropriate level of encryption is used to transfer data • Backup comply regulatory compliance • Get notifications of success or errors in backup on a daily basis • Acceptable recovery time for critical data is defined • Recover lost data from a remote location within SLA • When changes are made to application software, are they applied to the off-site copy • Seamless DR transition in case site failure	NA	Out of Scope

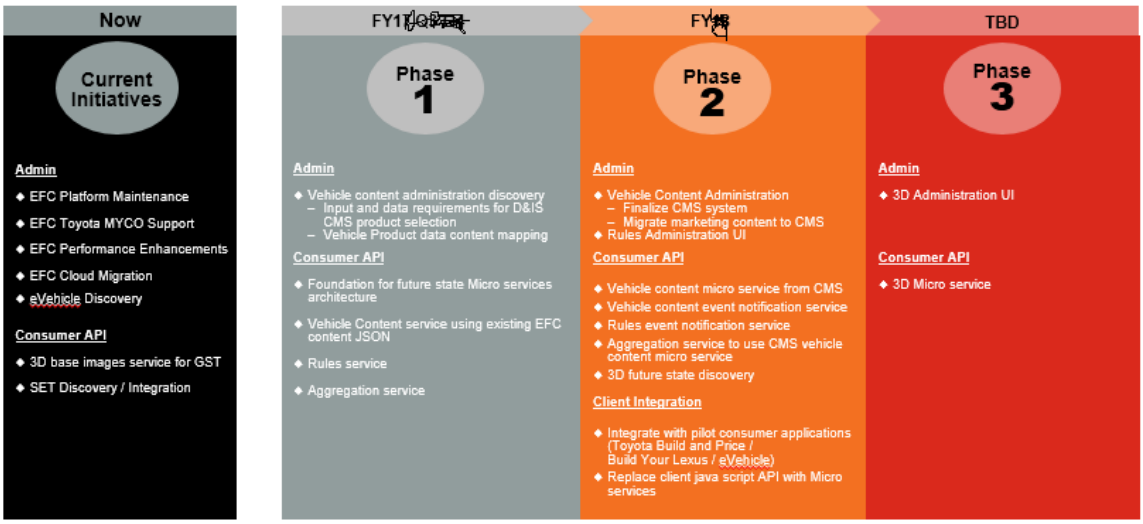
3.3 Expected Key Benefits on Migration

Scalability	Horizontal scalability at services (spring boot services) & cluster (ECS cluster) and auxiliary services (cache / search / nosql / EFS) level
HA	N- Node failover ability at services and filesystem data
Management	80% reduction on operation, 70% reduction on Hardware expense, repeated platform provisioning through code, 1 touch release upgrade without downtime, hardware changes happens in controlled manner, achieves agility in releases through integration of deployment & test process.
Security	SDN based network security and IAM based access control improves secure access. Separate API gateway ensures secure data access, improves governance and data integrity
Performance	Auxiliary services are elastic, removal of thick javascript layer for business logic helps better performance
Backup & DR	Migration architecture takes care of cross site recovery objective

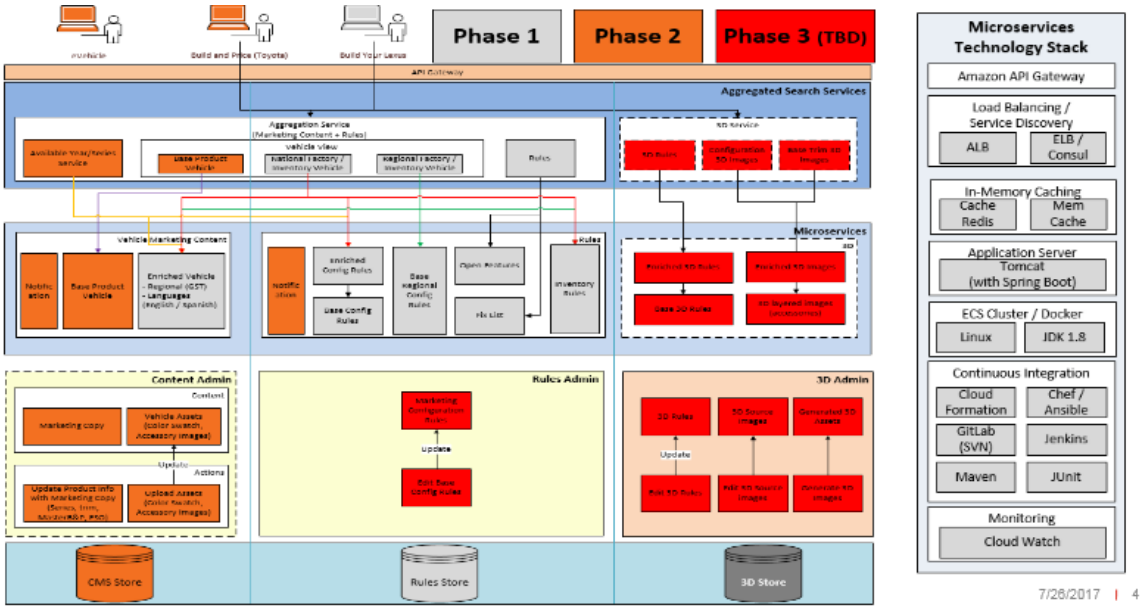
4 Migration Path

4.1 RoadMap

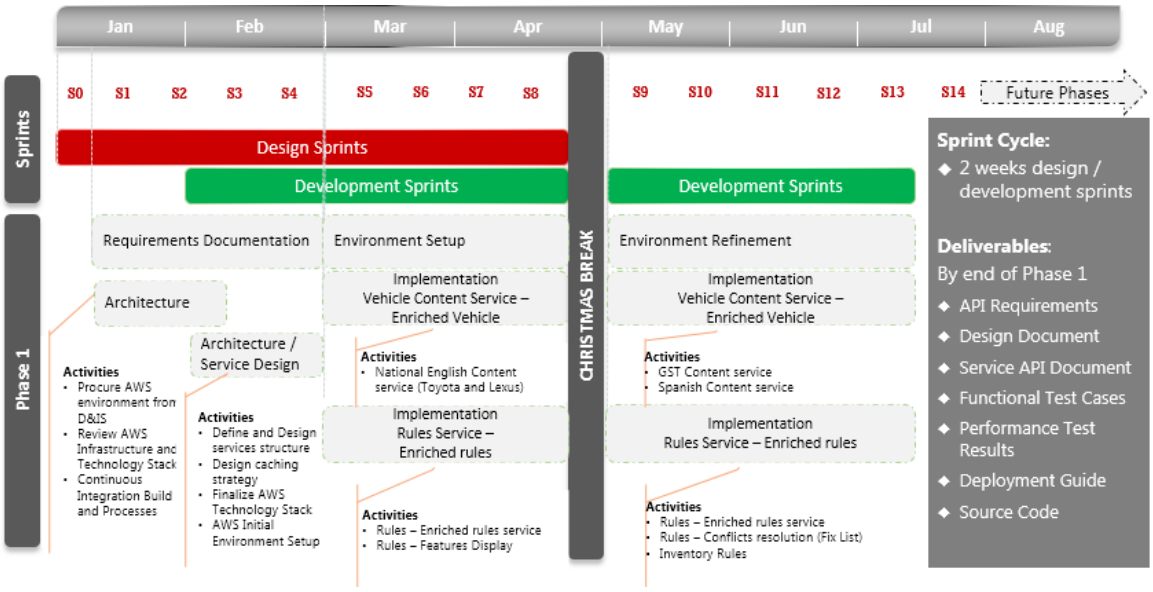
4.1.1 High Level Future State



4.1.1 Platform Architecture Future State

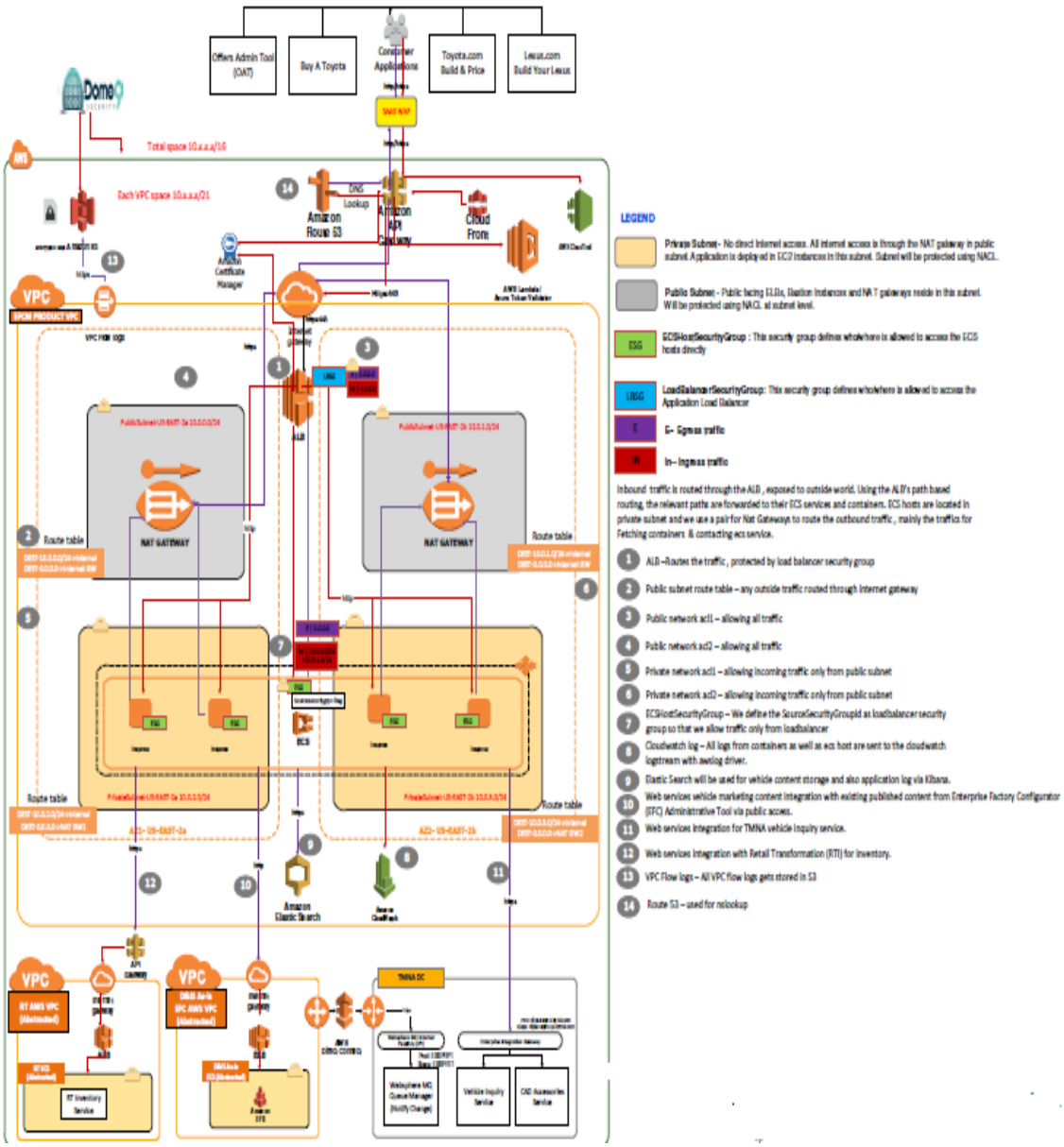


4.1.1 Project Execution Roadmap



4.2 Proposed Solution Architecture

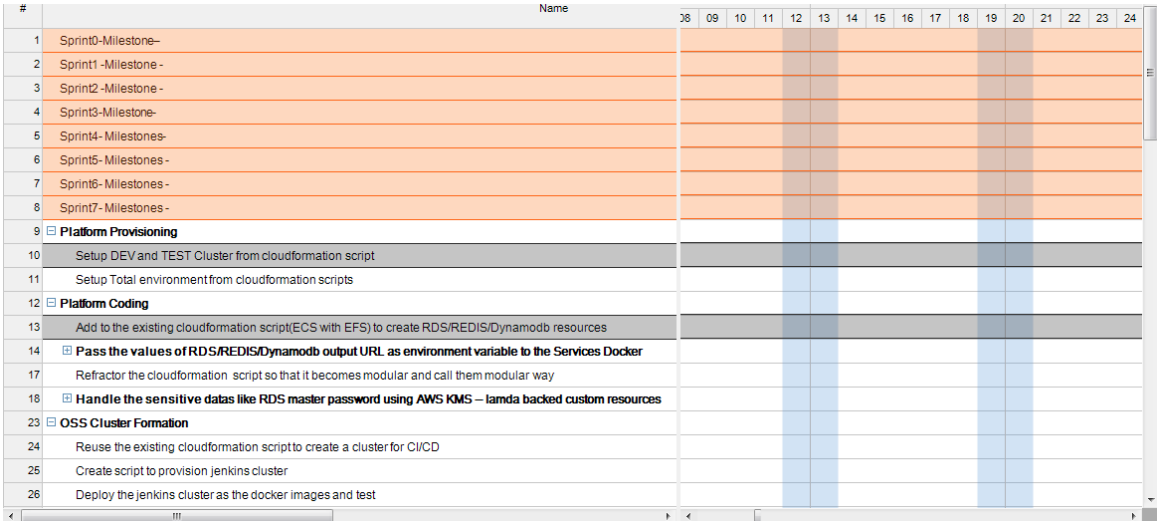
4.2.1 Brief Microservice deployment Architecture



4.3 Security & Compliance risk report

4.4 Project Plan & SOW for implementation

4.4.1 Project Plan



#	Name	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	Sprint0-Milestone–																	
2	Sprint1 -Milestone -																	
3	Sprint2 -Milestone -																	
4	Sprint3-Milestone-																	
5	Sprint4- Milestones-																	
6	Sprint5- Milestones -																	
7	Sprint6- Milestones -																	
8	Sprint7- Milestones -																	
9	**Platform Provisioning**																	
10	Setup DEV and TEST Cluster from cloudformation script																	
11	Setup Total environment from cloudformation scripts																	
12	**Platform Coding**																	
13	Add to the existing cloudformation script(ECS with EFS) to create RDS/REDIS/Dynamodb resources																	
14	**Pass the values of RDS/REDIS/Dynamodb output URL as environment variable to the Services Docker**																	
17	Refractor the cloudformation script so that it becomes modular and call them modular way																	
18	**Handle the sensitive datas like RDS master password using AWS KMS – lamda backed custom resources**																	
23	**OSS Cluster Formation**																	
24	Reuse the existing cloudformation script to create a cluster for CI/CD																	
25	Create script to provision jenkins cluster																	
26	Deploy the jenkins cluster as the docker images and test																	

4.5 Detailed POC

Attached separately