Common Services Platform Architecture

For

General Services Administration (GSA)

Integrated Award Environment (IAE) Common Services Platform (CSP)

Version 1.2

Task Order GSQ0014AJ0101

Submitted by:

Booz | Allen | Hamilton

March 25, 2015

Revision History

Revision History			
Date	Description	Version	Author
Oct 8, 2014	Initial Draft	0.1	Booz Allen Hamilton
Nov 7, 2014	First Version (with comments incorporated)	1.0	Booz Allen Hamilton
Dec 19, 2014	Updates to First Version (based on v1.0 feedback)	1.1	Booz Allen Hamilton
Mar 25, 2015	Updates corresponding with Release 1	1.2	Booz Allen Hamilton

Table of Contents

..... vii
1. Introduction 1
1.1. Document Organization 1
2. High-Level Architecture 2
2.1. Conceptual Architecture 2
2.2. Logical Architecture Layers 3
2.3. Logical Deployment Architecture 9
2.4. CSP Architecture Scope 12
3. Hosting 15
3.1. Site Map 15
3.2. Logical Architecture 17
3.2.1. Logical Architecture 17
3.2.2. Assumptions and Constraints 18
3.3. Use Cases 18
3.3.1. HST-1 Create an Organization 20
3.3.2. HST-2 Create a Project 21
3.3.3. HST-3 Create an Application 22
3.3.4. HST-4 Deploy to Component/Minimum-Capacity (MC)/Production-Like IAE CS Containers 23
3.3.5. HST-5 Deploy to Production 24
3.3.6. HST-6 Development Cycle Flow 24
3.3.7. HST-7 High-Level Deployment Use Case 25
3.3.8. HST-8 Upgrade a Host 26
3.3.9. HST-9 Application Hosting with Docker Containers 27
3.3.10. HST-10 Application Description File 31
3.3.11. HST-11 Patching 32
3.3.12. HST-12 Deployment of a legacy Application on IaaS 33
4. API Management 35
4.1. Site Map 35
4.2. Logical Architecture 37
4.2.1. Assumptions and Constraints 38
4.3. Physical Architecture 38
4.3.1. Portal and Gateway 39
4.3.2. Enterprise Service Bus 39
4.4. Use Cases 40
4.4.1. API-1 Publishing an API 40
5. Data and Search 43
5.1. Site Map 44
5.2. Logical Architecture 46
5.2.1. Data Logical Architecture 46
5.2.2. Assumptions and Constraints 47
5.3. Physical Architecture 48
5.4. Use Cases 51
5.4.1. DAT-1 Install Core & Essential Data Platform Tools 53
5.4.2. DAT-4 Data Export from Data Lake to Data Stores 54
5.4.3. DAT-5 Enable Boolean Search Capability 54
5.4.4. DAT-6 Search Specific Index via Elastic Search API 55

5.4.5. DAT-7 Elastic Search Auto Indexing 55
5.4.6. DAT-8 Migration of Redacted data from Prod to Non-Prod Container 56
5.4.7. DAT-9 Relational Data Continuous Integration / Deployment 1 57
5.4.8. DAT-10 Relational Data Continuous Integration / Deployment 2 58
6. Common Components 59
6.1. Site Map 59
6.2. Logical Architecture 60
6.2.1. Logical Architecture 60
6.2.2. Marketplace Logical Architecture 60
6.2.3. Assumptions and Constraints 61
6.3. Physical Architecture 61
6.4. Use Cases 62
6.4.1. CC-1: Create JIRA Account 63
6.4.2. CC-2: Log-in to CSP Marketplace UI 63
6.4.3. CC-3: Create/Manage Entitlements 64
6.4.4. CC-4: Social Widget Customizations 64
6.4.5. CC-5: Project Metrics 65
6.4.6. CC-6: Search 65
6.4.7. CC-7: Order Placement 65
6.4.8. CC-8: Create CSP Portal 66
6.4.9. CC-9: User Account Management 66
6.4.10. CC-10: Project Status Report 66
6.4.11. CC-11: CSP Marketplace UI (User Dashboard) 67
6.4.12. CC-12: Continuous Integration 68
6.4.13. CC-13: Load Testing 68
6.4.14. CC-14: Browser Testing 69
6.4.15. CC-15: Kibana UI Search 69
6.4.16. CC-16: Configure Kibana 70
6.4.17. CC-17: Configure Logstash 70
6.4.18. CC-18: Configure Logstash Forwarder 71
7. Reporting and Visualization 72
7.1. Site Map 72
7.2. Logical Architecture 73
7.2.1. Logical Architecture 73
7.2.2. Assumptions and Constraints 73
7.3. Physical Architecture 74
7.3.1. Physical Architecture 74
7.4. Use Cases 74
7.4.1. RV-1 Reporting and Visualization Developer Use Case 75
7.4.2. RV-2 Reporting and Visualization User Use Case 76
8. Appendices 77
8.1. Glossary of Acronyms 77
8.2. Term Catalog 80

List of Figures

1. Reference Architecture for the Integrated Awards Environment - Common Services Platform 2
2. Logical Architecture Layers for the Integrated Award Environment - Common Services Platform 3
3. Logical Deployment Architecture for Integrated Awards Environment - Common Services Platform 9
4. Hosting Services Site Map 16
5. Hosting Services Logical Architecture 17
6. Hosting Services Use Case Create an Organization 20
7. Hosting Services Use Case Create a Project 21
8. Hosting Services Use Case Create an Application 22
9. Hosting Services Use Case Deploy to Container 23
10. Hosting Services Use Case Deploy to Production 24
11. Hosting Services Use Case Development Cycle Flow 25
12. Hosting Services Use Case High- Level Deployment 25
13. Hosting Services Use Case Upgrade a Host 26
14. Hosting Services Use Case Upgrade a Docker Container 27
15. Hosting Services Use Case Phase 1 Simple Docker Deployment 29
16. Hosting Services Use Case Phase 1 IAE CSP Container 30
17. Hosting Services Use Case Scaling Docker Containers 30
18. Hosting Services Use Case Phase 2 Scaling Docker Containers Per App Within Container 31
19. Hosting Services Use Case Patching 32
20. Hosting Services Use Deployment of a Legacy Application on IaaS 33
21. API Management Sitemap 35
22. API Management Gateway Sitemap 36
23. API Management ESB Sitemap 37
24. API Management ESB Functional Stack 38
25. API Management High-Level ESB 39
26. API Management Use Case Publishing an API 41
27. API Management Use Case High-Level ESB Logging Architecture 41
28. API Management Use Case Queue High Availability Configuration 42
29. Data and Search Data Lake Sitemap 44
30. Data and Search Relational and Document Storage Tools Sitemap 44
31. Data and Search Search Tools Sitemap 45
32. Data and Search Logical Architecture 46
33. Data and Search Search Logical Architecture 47
34. Data and Search Physical Data and Search Architecture 48
35. Data and Search Search Indexing and High Availability 48
36. Data and Search Data Lake High Availability 49
37. Data and Search Automated ETL Process 50
38. Data and Search Use Case Install Core and Essential Data Platform Tools 53
39. Data and Search Use Case Create Relational Database 53
40. Data and Search Data Import from Data Stores to Data Lake 53
41. Data and Search Use Case Data Export from Data Lake to Data Stores 54
42. Data and Search Enable Boolean Search Capability 54
43. Data and Search Search Specific Index via Elastic Search API 55
44. Data and Search Elastic Search Auto Indexing 55

45. Data and Search Use Case Database Migration from Prod to Prod-Like 56
46. Data and Search Use Case Relational Data Continuous Integration Deployment 1 57
47. Data and Search Use Case Relational Data Continuous Integration Deployment 2 58
48. Common Components Sitemap 59
49. Common Components Logical Architecture 60
50. Common Components Physical Architecture 61
51. Common Components Use Case Create JIRA Account 63
52. Common Components Use Case Login to CSP Marketplace UI 63
53. Common Components Use Case Create Manage Entitlement 64
54. Common Components Use Case Widget Customization 64
55. Common Components Use Case Project Metrics 65
56. Common Components Use Case Search 65
57. Common Components Use Case Order Placement 65
58. Common Components Use Case Create CSP Portal 66
59. Common Components Use Case User Account Management 66
60. Common Components Use Case Project Status Report 1 66
61. Common Components Use Case Project Status Report 2 67
62. Common Components Use Case CSP Marketplace UI User Dashboard 67
63. Common Components Use Case Continuous Integration 68
64. Common Components Use Case Load Testing 68
65. Common Components Use Case Browser Testing 69
66. Common Components Use Case Kibana UI Search 69
67. Common Components Use Case Configure Kibana 70
68. Common Components Use Case Configure Logstash 70
69. Common Components Use Case Configure Logstash Forwarder 71
70. Reporting and Visualization Sitemap 72
71. Reporting and Visualization Logical Architecture 73
72. Reporting and Visualization Physical Architecture 74
73. Reporting and Visualization Use Case Developer Use of Reporting and Visualization 75
74. Reporting and Visualization Use Case User Use of Reporting and Visualization 76

List of Tables

1. Logical Architecture Layers 4
2. High-Level Service Groupings 9
3. Production Shared Services, CSP-Wide Services, Per-Container Services, and Per-Application Services 10
4. Non-Production Container Services 11
5. Architecture Scope 12
6. Hosting Services Site Map 16
7. Hosting Use Case Summary Listing 18
8. Docker Deployment Phases 28
9. API Management Use Case Listing 40
10. Data and Search Use Case Listing 51
11. Common Components Use Case Listing 62
12. Reporting and Visualization Use Case Listing 74
13. Glossary of Acronyms Listing 77
14. Term Catalog Listing 80

1. Introduction

The purpose of this document is to provide an architectural view of the Common Services Platform (CSP). This platform will serve as the foundational capability, which the future IAE Core Applications will be developed upon. The architecture depicted in this document describes the components of the CSP that includes hosting, data and search, reports and visualization, identity and access management (IAM), and application programming interface (API) management. The architecture for the CSP adheres to the IAE's architectural principles and the document also provides the product selection for each of the architectural components.

1.1. Document Organization

The document is organized in the following structure:

- High-Level Architecture describes the overall Common Services Platform architecture in conceptual, logical and physical architectures.
- Component Architecture sections are organized in the structure below for each of the components that make up the Common Services Platform that includes: Hosting, API Management, Data and Search, Common Components, Reporting and Visualization, and Identity Access Management.
  - Site Map provides a high-level architectural scope overview of each component
  - Logical Architecture describes interconnections and flow of data between the different components at the logical architectural level
  - Physical Architecture describes the tools and product selection that underlines the logical architecture
  - Use Cases provide the flow of steps and interaction with various actors
- Appendices provide additional context to the document
  - Software Version List provides a list of all software versions used in the solution
  - Glossary provides definitions for any acronyms used in the document
  - Term Catalog provides definitions for any terms used in the document

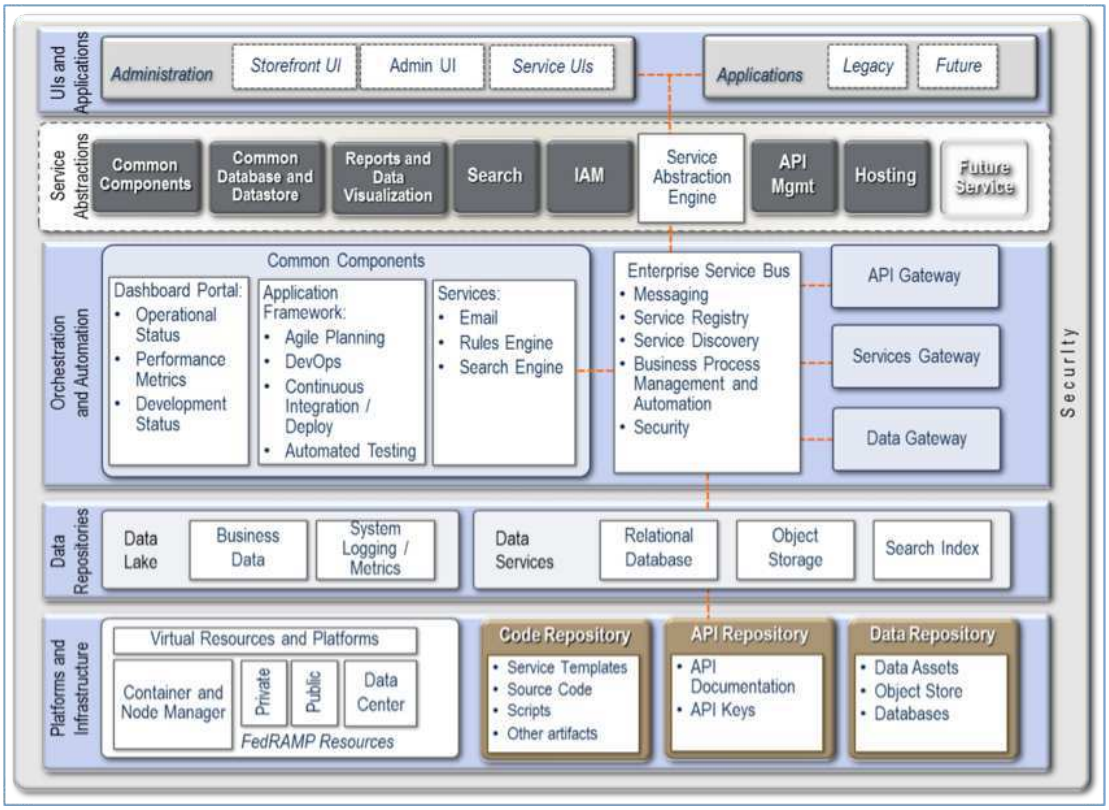
2. High-Level Architecture

The purpose of the high-level architecture is to describe the key functions and logical relationship of components of the Common Services Platform (CSP). These concepts are described in the context of the conceptual architecture, the logical architecture, and the logical deployment architecture.

The conceptual architecture illustrates the high-level approaches to accomplish the architecture principles of the IAE. The logical architecture provides a mapping of the primary business functions to each of the six layers of the CSP architecture. The logical deployment architecture aligns the functions to individual groups logically separated from one another.

2.1. Conceptual Architecture

Figure 1. Reference Architecture for the Integrated Awards Environment - Common Services Platform



The conceptual architecture in Figure 1, above, demonstrates the key concepts of the architectural approach to achieve the common services platform. This is divided into Platforms and Infrastructure, Data Repositories, Orchestration and Automation, Service Abstractions, and UIs and Applications.

The Platforms and Infrastructure layer is responsible for the management of hosting resources and repositories. The hosting resources support the deployment of both legacy applications as well as modernized Core Applications running on the CSP provided application platform. The repositories provide complete infrastructure-as-code and support application and hosting infrastructure, application programming interfaces, and data repositories.

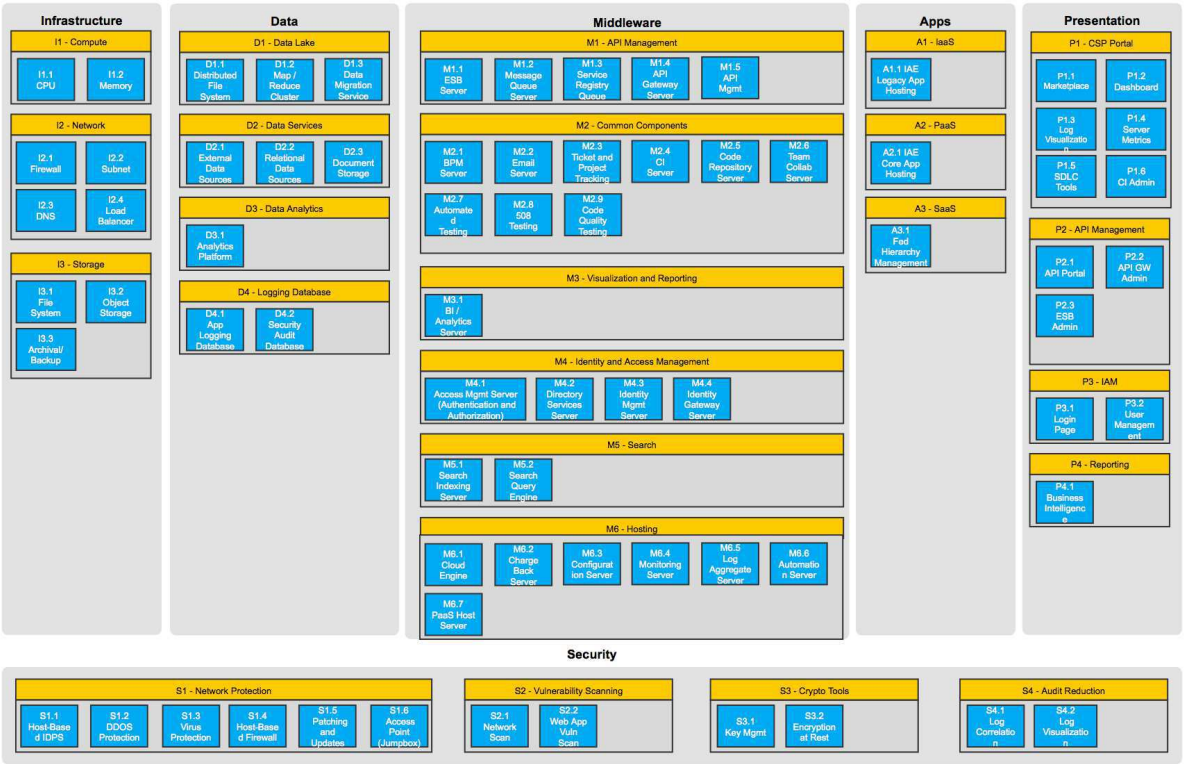
The Data Repositories layer provides the necessary services to ensure that the data-as-an-asset objectives of the IAE are met. This includes dedicated data services for applications such as Relational Database and Object Storage. The Data Lake provides the ability to process both structured and unstructured business data aggregated from across the platform as well as from external data sources.

The Orchestration and Automation layer provides the complete set of common components to support development and operations teams. Users access the common components through a single-pane-of-glass providing access to the marketplace to manage development environments, development tools to manage the agile process, and additional services such as email and rules engines. The Enterprise Service Bus is a service available in the marketplace that supports the development of enterprise integration patterns by core applications in order to integrate with legacy services.

The Service Abstraction Engine controls the provisioning of resources for consumption. The UI and Applications layer provides the presentation for capabilities provided by common services, core application, and legacy application end-points.

2.2. Logical Architecture Layers

Figure 2. Logical Architecture Layers for the Integrated Award Environment - Common Services Platform



Each component of the logical architecture layers is described in detail in the table below.

Table 1. Logical Architecture Layers

ID	CSP Architecture Layer	Components	Sub-components	Description
I1.1	Infrastructure	Compute	CPU	Component to provision CPU resources
I1.2	Infrastructure	Compute	Memory	Component to provision Memory resources
I2.1	Infrastructure	Network	Firewall	Configuration of Open / Closed Ports, Source / Destination IPs
I2.2	Infrastructure	Network	Subnet	Configuration of IP ranges and routing within network
I2.3	Infrastructure	Network	DNS	Configuration of Domain Name Resolution to IP Addresses Services
I2.4	Infrastructure	Network	Load Balancer	Provisioning/Configuration of load balancers
I3.1	Infrastructure	Storage	File System	Provisioning of block storage
I3.2	Infrastructure	Storage	Object Storage	Provisioning of object storage
I3.3	Infrastructure	Storage	Archival / Backup	Provisioning of Backup and Archival storage
D1.1	Data	Data Lake	Distributed File System	Underlying storage component of the Data Lake
D1.2	Data	Data Lake	Map / Reduce Cluster	Horizontal scaling component for evaluating information in the Data Lake
D1.3	Data	Data Lake	Data Migration Service	Component for import/export of information in/out of the Data Lake
D2.1	Data	Data Services	External Data Sources	Data sources to ingest into data lake
D2.2	Data	Data Services	Internal Data Sources	Relational Database used by Core Apps
D2.3	Data	Data Services	Data Services API	Document storage used by Core Apps
D3.1	Data	Data Analytics	Analytics Platform	Data Quality Metrics and Reports
D4.1	Data	Logging Database	App Logging Database	Unified collection of application logs
D4.2	Data	Logging Database	Security Audit Database	Unified collection of security event logs
M1.1	Middleware	API Management	ESB Server	The Enterprise Service Bus (ESB) supports the development of enterprise integration patterns by core applications in order to integrate with legacy services.

ID	CSP Architecture Layer	Components	Sub-components	Description
M1.2	Middleware	API Management	Message Queue Server	The component that holds the messages as they wait to be processed
M1.3	Middleware	API Management	Service Registry Queue	Service Registry manages a directory of services
M1.4	Middleware	API Management	API Gateway Server	The API Gateway provides high volume capabilities for security, auditing, monitoring and stateless orchestration and mediation of both REST and SOAP services at the perimeter of the CSP. This helps reduce unnecessary traffic from reaching the ESB
M1.5	Middleware	API Management	API Management	The API Management component provides an API Portal for services discovery and documentation as well as SLA enforcement and service management functions.
M2.1	Middleware	Common Components	BPM Server	The Business Process Management (BPM) server that manages the workflows
M2.2	Middleware	Common Components	Email Server	An Simple Mail Transfer Protocol (SMTP) server
M2.3	Middleware	Common Components	Ticket and Project Tracking	The ticket and project tracking manages requirements, defects/enhancements, and the Agile software development lifecycle
M2.4	Middleware	Common Components	CI Server	Jenkins Server that manages the build, test, and deployment of the builds
M2.5	Middleware	Common Components	Code Repository Server	The server that stores and versions the source code
M2.6	Middleware	Common Components	Team Collab Server	Wiki for team collaboration and document sharing
M2.7	Middleware	Common Components	Automated Testing	Provides developers with automated testing tools for such as for unit, integration, regression, and performance testing.
M2.8	Middleware	Common Components	508 Testing	Provides testing for 508 compliance
M2.9	Middleware	Common Components	Code Quality Testing	Provides the ability to perform static code analysis to ensure code quality against metrics such as software bugs or security defects.

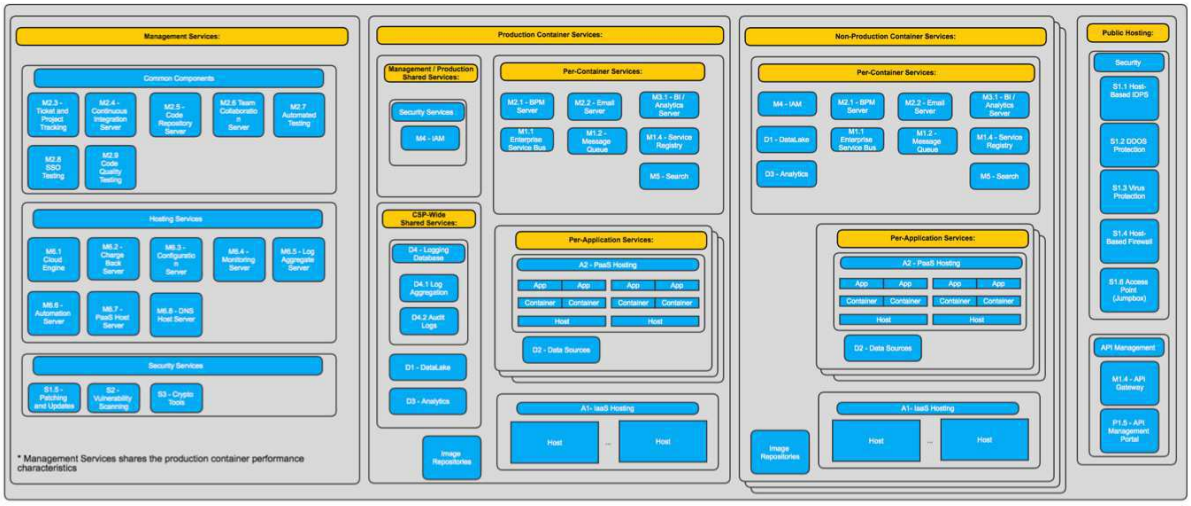
ID	CSP Architecture Layer	Components	Sub-components	Description
M3.1	Middleware	BI / Analytics Server	BI / Analytics Server	The server that analyzes the data and provides reporting features likes dashboards
M4.1	Middleware	IAM	Access Management	Provides open source Authentication, Authorization, Entitlement and Federation software
M4.2	Middleware	IAM	Directory Services Server	Provides open source LDAP directory services in Java.
M4.3	Middleware	IAM	Identity Management Server	Provides flexible, open source services for automating management of the identity life cycle.
M4.4	Middleware	IAM	Identity Gateway Server	Provides high-performance reverse proxy server with specialized session management and credential replay functionality
M5.1	Middleware	Search	Search Indexing Server	The component of the search solution that manages the indexing of content.
M5.2	Middleware	Search	Search Query Engine	The component of the search solution which executes search queries
M6.1	Middleware	Hosting	Cloud Engine	Manage the provisioning of environments based on infrastructure as code.Ê
M6.2	Middleware	Hosting	Charge Back Server	Provide cost metrics and forecasting based on usage
M6.3	Middleware	Hosting	Configuration Server	The server that keeps track of all of the infrastructure
M6.4	Middleware	Hosting	Monitoring Server	The server that keeps track of the health of the entire infrastructure and sends alerts when needed
M6.5	Middleware	Hosting	Log Aggregation Server	The server that stores and analyzes all of the logs from all of the different components of the platform
M6.6	Middleware	Hosting	Automation Server	Execute configuration management scripts
M6.7	Middleware	Hosting	PaaS Host Server	The server that operates an application server necessary to deploy Core Applications.
A1.1	Apps	IaaS	IAE Legacy Application Hosting	The IAE legacy applications are hosted on the CSP in accordance to unique hosting requirements

ID	CSP Architecture Layer	Components	Sub-components	Description
A2.1	Apps	PaaS	IAE Core Application Hosting	The IAE Core Applications are hosted on a application server provided by the CSP.
A3.1	Apps	SaaS	Fed Hierarchy Management	The IAE Fed Hierarchy Management application is an application provided by the Common Service Platform that implement domain-specific business functions and is deployed similar to Core Applications.
P1.1	Presentation	CSP Portal	Marketplace	The marketplace allows management of the development environments.
P1.2	Presentation	CSP Portal	Dashboard	The dashboard provides a consolidated view of important parts of the CSP
P1.3	Presentation	CSP Portal	Log Visualization	The log visualization component provides a graphical rendering to analyze trends in system logs.
P1.4	Presentation	CSP Portal	Server Metrics	The server metrics component provides monitoring metrics to understand system performance and operational state of the environment.
P1.5	Presentation	CSP Portal	SDLC Tools	The SDLC toolset allows teams to effectively manage the software lifecycle.
P.1.6	Presentation	CSP Portal	CI Admin	The Continuous Integration administrative interface allows build tasks to be managed
P2.1	Presentation	API Management	API Portal	The API Portal provides administrative and end-user management of APIs.
P2.2	Presentation	API Management	API GW	The API Gateway provides high-volume security validation of APIs
P2.3	Presentation	API Management	ESB	The ESB supports implementation of enterprise integration patterns to support legacy interfaces
P3.1	Presentation	IAM	Login Page	The Login Page provides a user interface for authentication
P3.2	Presentation	IAM	User Management	The user management component allows user accounts to be provisioned and updated.
P4.1	Presentation	Reporting	Business Intelligence	Business Intelligence provides the ability to perform reporting against business objects.

ID	CSP Architecture Layer	Components	Sub-components	Description
S1.1	Security	Network Protection	Host-Based IDPS	Host-Based Intrusion Detection and Prevention Service
S1.2	Security	Network Protection	DDOS Protection	Distributed Denial of Service Protection
S1.3	Security	Network Protection	Virus Protection	The component that scans files sent over the network for viruses.
S1.4	Security	Network Protection	Host-Based Firewall	The software that runs on individual hosts and configures open and closed ports.
S1.5	Security	Network Protection	Patching and Updates	The component that provides patch management and updates primarily for operating system leveraged across the platform.
S1.6	Security	Network Protection	Access Point (Jumpbox)	The server through which administrators must login in order to gain administrative access to hosts within the CSP
S2.1	Security	Vulnerability Scanning	Network Scan	A scanning tool focused on detecting open ports and network vulnerabilities.
S2.2	Security	Vulnerability Scanning	Web App Vulnerability Scanning	A scanning tool focused on detecting the presence of vulnerabilities in web applications such as those described in the OWASP Top 10.
S3.1	Security	Crypto Tools	Key Management	A component responsible for management of encryption keys.
S3.2	Security	Crypto Tools	Encryption at Rest	The component responsible for ensuring that data at rest is properly encrypted.
S4.1	Security	Audit Reduction	Log Correlation	The component that performs unified logging
S4.2	Security	Audit Reduction	Log Visualization	The log visualization component provides a graphical interface to analyze system logs.

2.3. Logical Deployment Architecture

Figure 3. Logical Deployment Architecture for Integrated Awards Environment - Common Services Platform



The logical deployment architecture in Figure 3, above, describes at a high-level how services are grouped, the degree to which they are made accessible to services operating in other groups, and the number of instances of a service that may need to be provisioned. The high-level services groupings are comprised of Management Services, Production Container Services, Non-Production Container Services, and Public Hosting Services. Within the Production Container Services, additional groupings consist of Management / Production Shared Services, CSP-Wide Shared Services, Per-Container Services, and Per-Application services. The Non-Production Container Services consists of Per-Container Services and Per-Application Services.

A detailed description of each of the high-level service groupings is provided in Table 2.

Table 2. High-Level Service Groupings

Group Name	Description
Management Services	The Management Services group contains the set of services needed to administer and operate the Common Services Platform. This includes the Common Components tools for development and testing, the hosting services that manage containers and nodes, and the security services provide patching and vulnerability scanning.
	The Management Services environment operates with the same performance characteristics as the Production Container.
	Each of the services within the Management Services group are provisioned once per Common Services Platform instance.
Production Container Services	The Production Container services are provisioned and operated in an instance of the Production Container environment. There will be only one instance of a Production Container operating within the CSP. The Production Container instance will operate to ensure

Group Name	Description
	the contained services meet the SLA objectives in accordance with the definition of the Production Container.
Non-Production Container Services	The Non-Production objectives services are those that operate in an instance of a Production-Like, Minimal-Capacity, or Component Container environment. There may be multiple instances of these non-production containers. A Non-Production container will operate to ensure the contained services meet the SLA in accordance with the definition of that container type of either Production-Like, Minimal-Capacity, or Component.
Public Hosting Services	The Public Hosting services group contains the set of services necessary to securely expose end-points for external consumption. This includes security services such as Intrusion Detection and Prevention and API Management services such as the API Gateway and API Management Portal.
	Each of the services within the Public Hosting Services group are provisioned once per Common Services Platform instance.

Production Shared Services, CSP-Wide Services, Per-Container Services, and Per-Application Services as described by Table 3.

Table 3. Production Shared Services, CSP-Wide Services, Per-Container Services, and Per-Application Services

Group Name	Description
Management / Production Shared Services	The Management / Production Shared Services group contains the set of services that are accessible to both the Management Services environment and the Production Container environment. There is only one service in this category, which is the IAM service. This ensures that the same IAM services for user identities, credentials, and entitlements can be leveraged by services in the Management Services group such as Common Components tools as well as the services in the Production Container Services group such as Core Applications.
	Each of the services within the Management / Production Shared Services group will be provisioned once per Production Container instance.
CSP-Wide Services	The Common Services Wide Shared Services group contains the set of services which are accessible to services across the common services platform but hosted in the Production Container instance.
	This includes primarily services for logging. For example, logs produced by Core Applications in either the Production Container Environment and Non-Production Container Environments will contribute logs to this single set of logging services.
	Note that the Data Lake is included in CSP-Wide Services group. This is for the purpose of the storage of logging information

Group Name	Description
	across Container Environments. In the future, we may consider re-categorizing the Data Lake service into the Per-Container Services group of the Production Container Services if this is unclear.
	Each of the services within the CSP-Wide Services will be provisioned once per Production Container instance.
Per-Container Services	The Per-Container Services group contains the set of services which are only accessible to other services within the Production Container services group. For example, Core Application services with APIs are routed through the Enterprise Service Bus.
	Each of the services within the Per-Container Services will be instantiated once per Production Container instance.
Per-Application Services	The Per-Application Services group contains the set of services that are made available only to core applications deployed on the PaaS hosting environment within a container.
	The services within this category consist of the PaaS Hosting, the Relational Database Management System and the Document Storage. For example, each application is provisioned with a dedicated relational database.
	Each of the services within the Per-Applications services group are provisioned once per application. There can be multiple instances of the Per-Applications Services group within each container environment instance.

Services within the Non-Production Container Services group are further divided into Per-Container Services and Per-Application Services as described by Table 4.

Table 4. Non-Production Container Services

Group Name	Description
Production Container Services	The Per-Container Services group contains the set of services that are only accessible to other services within the Production Container services group. For example, Core Applications may expose APIs that are routed through the Enterprise Service Bus.
	Each of the services within the Per-Container Services will be instantiated once per Production Container instance.
	For each instance of a Non-Production container, a dedicated instance of the Data Lake, IAM, and ESB services are provided to facilitate development and testing.
Non-Production Container Services	The Per-Application Services group contains the set of services that are made available only to core applications deployed on the PaaS hosting environment within a container.
	The services within this category consist of the PaaS Hosting, the Relational Database Management System and the Document

Group Name	Description
	Storage. For example, each application is provisioned with a dedicated relational database.
	Each of the services within the Per-Applications services group are provisioned once per application. There can be multiple instances of the Per-Applications Services group within each container environment instance.

2.4. CSP Architecture Scope

It is important to define the scope of responsibilities in terms of the Common Services Platform and Core Applications. The scope is defined in terms of the following services:

- Hosting Services
- Data Services
- API Management
- Reporting and Visualization
- Identity Access Management
- Common Components

These scopes are summarized in Table 5, below.

Table 5. Architecture Scope

Services	Common Services Platform	Core Applications
Hosting	Responsible for providing hosting services for Core Applications. This includes establishing the IAE CSP container performance characteristics, SLAs on the containers themselves, providing the catalog of application runtime environments, standard sizing options for containers, securing the operating systems and providing logging, monitoring and security scanning services.	Responsible for configuring the performance characteristics of the IAE CS Container Environments. This includes parameters such as: • Choosing Linux Container attributes (CPU cores, Memory, Network ports) • Application runtime selection from catalog provided by CSP • Metrics selected from available monitoring from CSP • Auto-scaling options
Data Services	Responsible for providing Relational Database Management, Object Storage, and Data Lake services. This includes providing a mechanism for developers to do the following: • Provide developers with a standard and process for placing database scripts in a repository	Responsible for developing Business Specific APIs, and defining schemas for relational databases. This includes: • Development of the relational database data model, schema, and create scripts

Services	Common Services Platform	Core Applications
	to ensure execution upon deployment • Support developers to understand the success/failure of script execution upon deployment • Allow developers to create/ update the database schema (Tables, Constraints, Indexes, etc.) through scripts added to the repository. • Allow developers to define base data scripts to be loaded/updated through scripts added to the repository • Allow developers to resolve the end-point to the data services	• Development of the relational database base data scripts required for the application to run properly. • Development of application code to leverage the data services appropriately.
API Management	Responsible for providing API Management and API gateway services as well as the Enterprise Service Bus to support service needs of Core Applications and common services. This includes providing a mechanism for developers to do the following: • Provide developers with a standard for placing API specification files in a repository to ensure registration upon deployment • Support developers to understand the success/failure of API registration. • Supporting the ability to determine whether an API should be exposed for consumption internally and/or externally Providing developers with a strategy for resolving end-points of available services	Responsible for defining service descriptions exposed by the application that conforms to the specification of the common services platform. This includes: • Development of the API specification files • Development of application code to implement the API
Reporting and Visualization	Responsible for providing business intelligence reporting of business	Responsible for the development of the business intelligence report objects leveraging the

Services	Common Services Platform	Core Applications
	data and visualization of log data ingested into the Data Lake. This includes providing a mechanism for users to do the following: • Develop reports to be deployed into the reporting and visualization service • Execute reports within the reporting and visualization service	necessary thick-client development environment. Responsible for producing logging statements within core application code.
Identity Access Management	Responsible for providing identity management and policy management to include SSO portal and user management as well as validating authentication and authorization of actions performed on the platform.	Responsible for role based access controls within application code based on entitlements provided to the application common services platform.
Common Components	Responsible for providing common components such as development tools to support the software development life cycle of core application developers, services such as email and business process management, as well as services that span all applications such as federal hierarchy.	Responsible for populating Software Development Life Cycle (SDLC) tools with project data, for development of business rules deployed to the business process management system, and for leveraging the capabilities provided as services exposed through the ESB by the common services platform.

3. Hosting

The hosting service is responsible for the automation, orchestration, management, operations, and security of the infrastructure and platform as a service capabilities in support of all the other IAE CSP services and respective Core Applications. Hosting will provide the foundation to support the infrastructure needs of the platform services. The hosting service is currently built on the FISMA Moderate and FedRAMP compliant Amazon Web Services (AWS) cloud but the modular design enables it to be re-deployed on other FedRAMP-compliant cloud service providers. The modular functional components also make for easier plug and play of future technologies and products that can optimize the hosting service. The hosting service has been designed and built with open source first, one of the guiding principles of the GSA CIO. The components of the hosting service can expose APIs for end-to-end automation, helping to achieve an Automate Always operations approach. The automation and orchestration capabilities are further enhanced through infrastructure as code tools, extending the Agile approach to infrastructure and providing a true DevOps solution. These features, along with continuous integration, help provide the IAE CSP with a continuous delivery capability that will enable developers to promote thoroughly tested code to production quickly.

3.1. Site Map

The hosting service depicted in Figure 4 provides a construct to manage applications grouped by project and organization. An organization is considered to be a group or team; for example, one organization is GSA IAE. Additional sub-organizations can be defined to allow individual contractors to develop on the platform

A project is a collection of applications to support services as part of the organization. An application is a single deployable unit with one or more dependencies. Each application will have its own Git repository, build process and Docker container images for repeatable deployment to the various containers (i.e. Component, Minimal-Capacity, Prod-like, Production).

Figure 4. Hosting Services Site Map

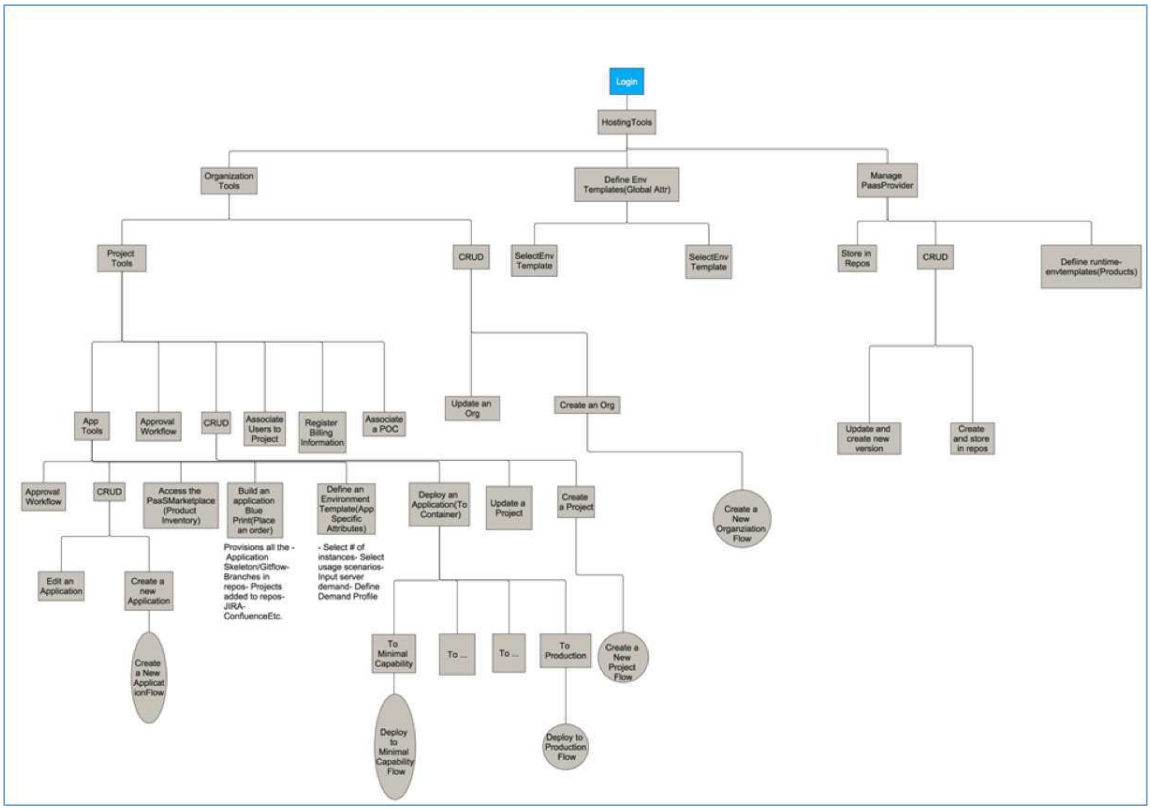


Table 6 below provides a breakdown of the main levels of the Hosting Services Site Map in Figure 4.

Table 6. Hosting Services Site Map

Hosting Site Map Main Level	Description
Manage PaaS Provider	The hosting services will provide an API and eventually a UI allowing the PaaS administrators to operate the PaaS environment including managing cloud instances/virtual machines, Operating System (OS) configuration, Platform as a Service (PaaS) configuration, baseline images, PaaS application runtime images and operational workflows and processes.
Define an Environment Template	The hosting services will provide the capability for PaaS administrators to define the global environment templates for the IAE container types (component, minimal-capability, production-like, and production). These templates will provide all the global attributes that affect applications on the platform for each container. Application developers cannot change these parameters.
Organization Tools	The hosting service will provide a unified CSP Portal web user interface to allow Common Services users to do Create, Read, Update, and

Hosting Site Map Main Level	Description
	Delete (CRUD) operations and get information on organizations, projects and applications that they have access to.
	The CSP Portal also manages the environment templates and fills in the specifics of the environment specific to that project and application. Environment templates will be used by users to define and select the parameters that make up the various IAE container parameters for each type of container. For example, one could define the testing demand profile for the minimal-capability container associated with that application. These parameters are project and application specific. The default values and global parameters will be set by PaaS administrators specified in the Site Map under Hosting Services/Define an Environment Template above.

3.2. Logical Architecture

3.2.1. Logical Architecture

Figure 5. Hosting Services Logical Architecture

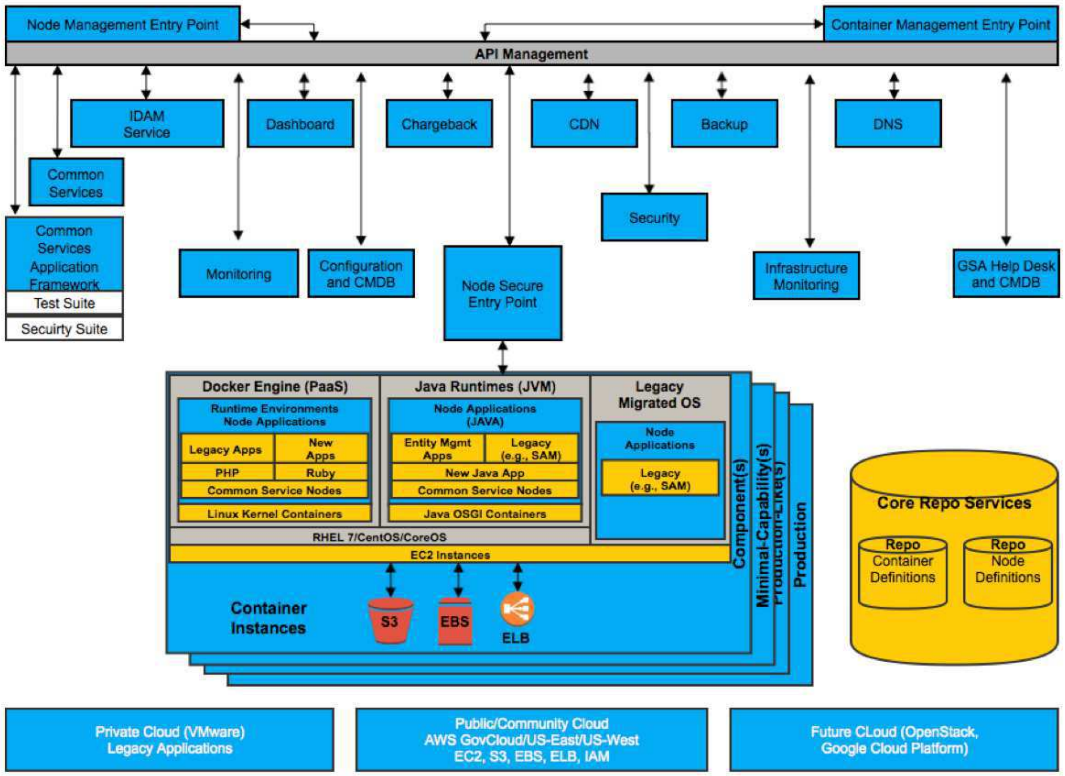


Figure 5 provides a high-level logical architecture of the proposed GSA IAE hosting environment and the major components that will provide the features and capabilities for the Common Services Platform. Each software or tool that is identified in Figure 5 can be tied back to the hosting requirements. This modular hosting environment leverages multiple open-source tools and all components expose APIs for end-to-end automation. The architecture has also been designed to support multi-cloud management so that the GSA can integrate private and other public clouds in the future.

3.2.2. Assumptions and Constraints

Assumptions:

- There is no International Traffic in Arms Regulations (ITAR) constraint so the hosting environment will be deployed in AWS US East/West regions.
- The uptime requirement for the Common Service platform is 99.8%.
- The recovery time objective (RTO) and recovery point objective (RPO) for the Common Service platform will be defined at a later date.

Constraints:

- Developers will need to use new workflow to deploy applications
- Existing applications will need to be modified to use the new deployment process

3.3. Use Cases

Table 7. Hosting Use Case Summary Listing

ID	Use Case Name	Description	Actors	Pre-condition	Post-condition
HST-1	Create an Organization	Describes the flow of how an organization is created on the platform	Project Admins/ Developers	User has access	An organization is created. Common components are setup for organization.
HST-2	Create a Project	Describes how a project is created on the platform	Project Admin/ Developers	User has access. An organization exists	A project is created
HST-3	Create an Application	Describes how an application is created on the platform	Project Admin/ Developers	User has access. A project exists	An application development environment is created. Common components are setup for the application. DevOps tools

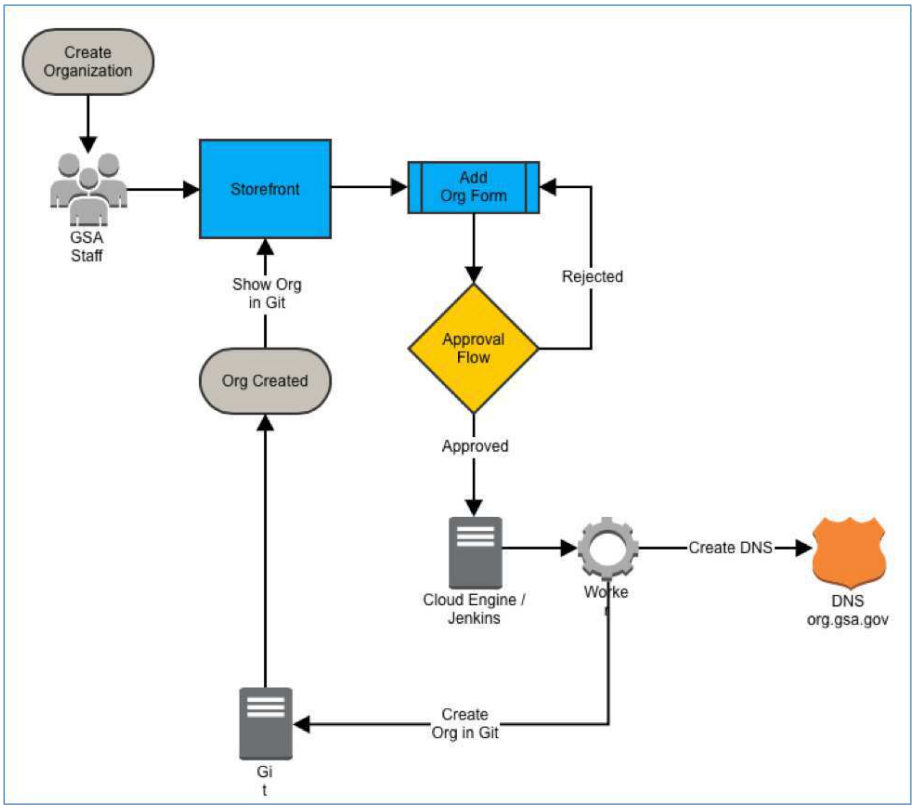
ID	Use Case Name	Description	Actors	Pre-condition	Post-condition
					are setup for application.
HST-4	Deploy to Component/ Minimum-capacity/ Production-Like IAE CS Containers	Deploying an application to the minimum capacity container	Project Developers	An application is created in the storefront. Developers have access. Developers have code to commit	Application is deployed in and monitored in IAE CS Containers
HST-5	Deploy to Production	Deploying an application to the production container	Project Developers	An application is created in the storefront. Developers have access. Developers have code to commit	Application is deployed in and monitored in IAE CS Production Containers
HST-6	Development Cycle Flow	Describes how developer continuously works with the platform	Project Developer	Active application development	Developer
HST-7	High-Level Deployment Use Case	General process of deploying applications to the platform	Project Developer	Application is ready for deployment	Application has been deployed to specified environment
HST-8	Upgrade a Host	Upgrading a server using Chef	PaaS Administrator	Identify needed software updates or patches	Host is upgraded
HST-9	Application Hosting with Docker Containers	First and second phase of Docker Platform as a Service from the point of view of an application	N/A	None	Static Docker containers for applications
HST-10	Application Description File	Summarizes the use of JSON files	N/A	None	JSON file
HST-11	Patching	Describes how patching of	PaaS Administrator	Clear vulnerability	Patched hosts

ID	Use Case Name	Description	Actors	Pre-condition	Post-condition
		hosts will be achieved		remediation steps and patches are available	
HST-12	Deployment of a Legacy Application on IaaS	Provides process overview for migration of Legacy Applications	N/A	None	Migrated Legacy Application

3.3.1. HST-1 Create an Organization

The use case shown in Figure 6, below, describes the flow for creating a new organization through the CSP Portal. The diagram describes part of the Hosting Service site map reflecting how an application owner/developer can create the initial organization that will encapsulate their projects and applications. Organizations, projects and applications are defined at the beginning of the hosting section.

Figure 6. Hosting Services Use Case Create an Organization

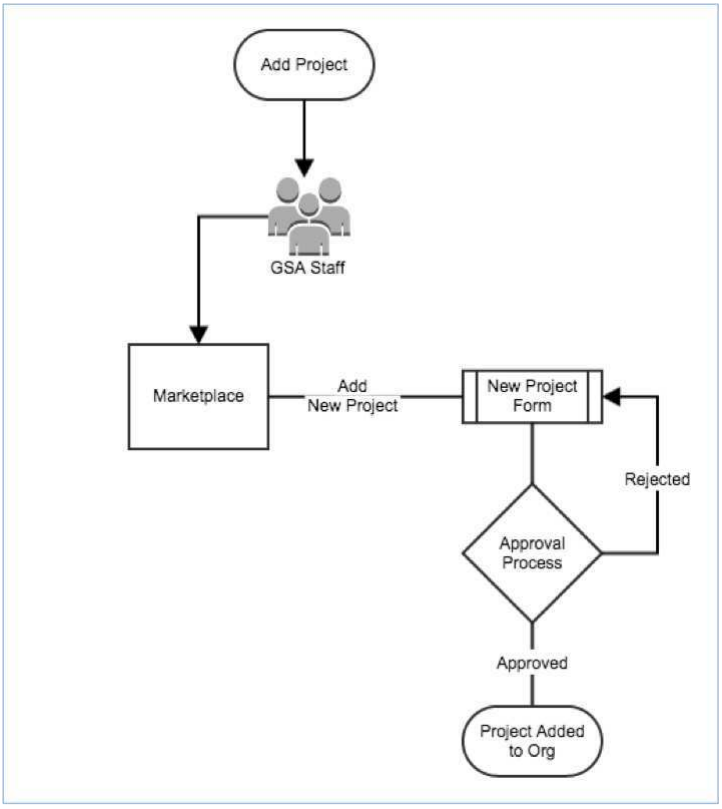


In order for an organization to be created, the creator must first have log in credentials. Once authenticated, the creator fills out the Organization form, and upon administrative approval of that form the automation engine will automatically provision the organization in the Code Repository and will update the Organization's status in the marketplace.

3.3.2. HST-2 Create a Project

The use case shown in Figure 7, below, describes the flow of creating a new project through the marketplace. This describes part of the Hosting Service site map reflecting how an application owner/ developer can create a new project flow within an existing organization that will encapsulate their applications. Organizations, projects and applications are defined at the beginning of the hosting section.

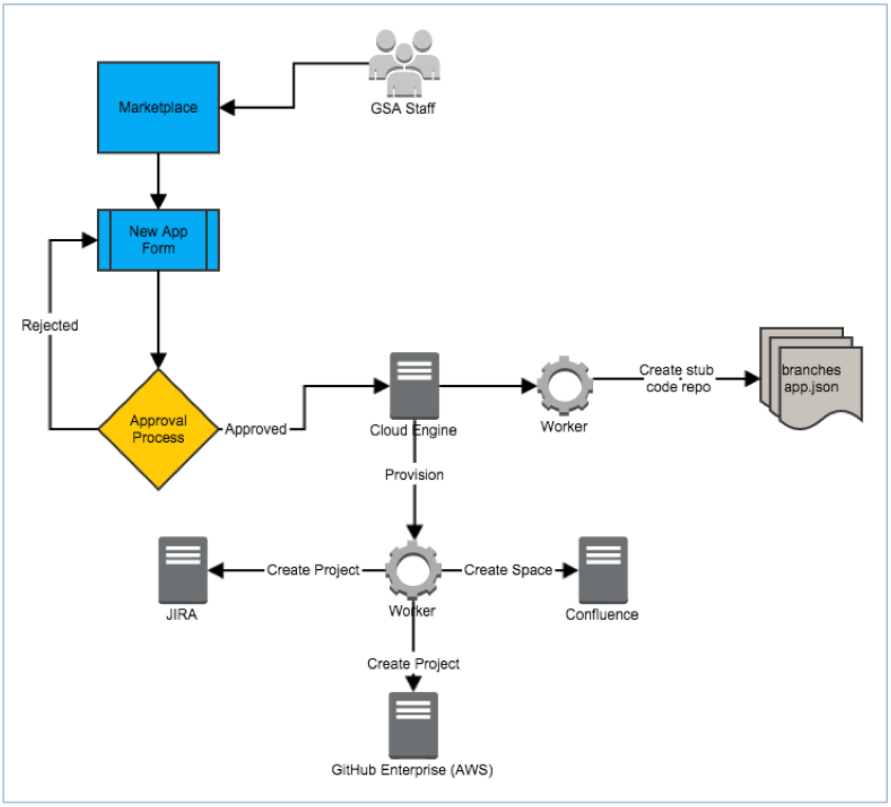
Figure 7. Hosting Services Use Case Create a Project



In order for a project to be created, the creator must first have log in credentials. Once authenticated, the creator fills out the Project form, and upon administrative approval of that form the project is made available in the marketplace under the associated organization.

3.3.3. HST-3 Create an Application

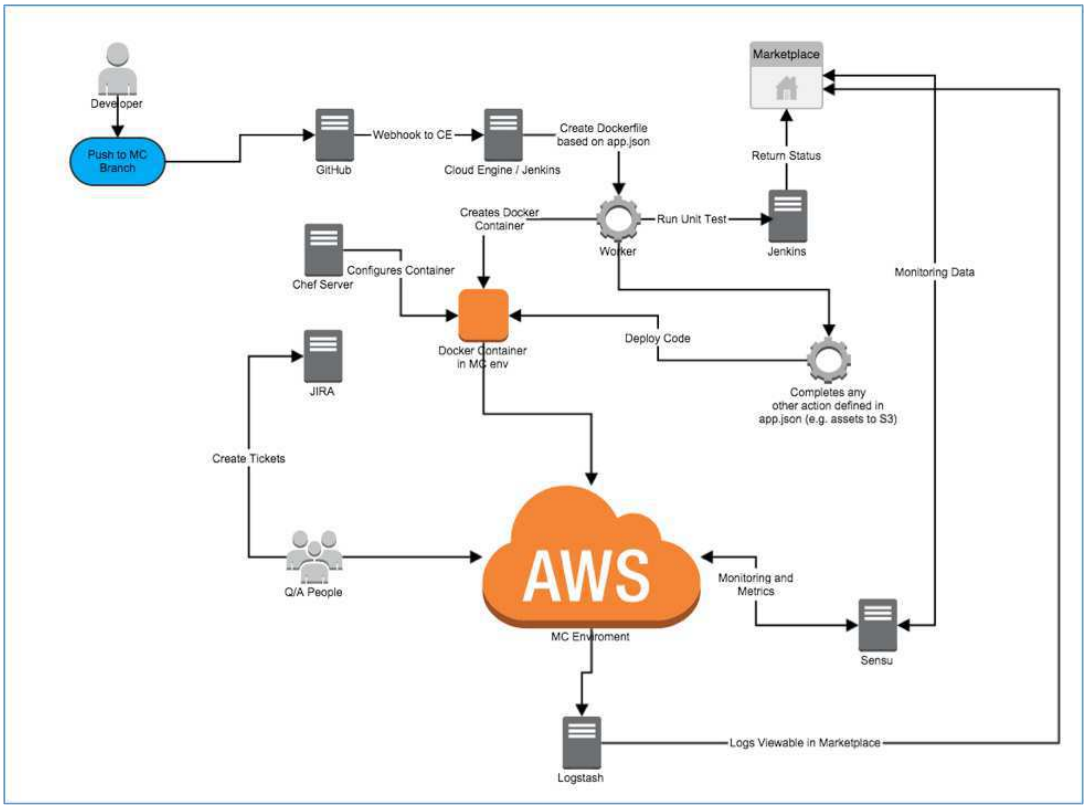
Figure 8. Hosting Services Use Case Create an Application



In order for an application to be created, the creator must first have log in credentials. Once authenticated, the creator fills out the new application form, and upon administrative approval of that form the automation engine will automatically provision new branches with the application template pre-populated in the repository. In addition, the Cloud Provision engine also provisions and configures the app in JIRA, Confluence, and Git. This process is shown in Figure 8, above.

3.3.4. HST-4 Deploy to Component/Minimum-Capacity (MC)/Production-Like IAE CS Containers

Figure 9. Hosting Services Use Case Deploy to Container



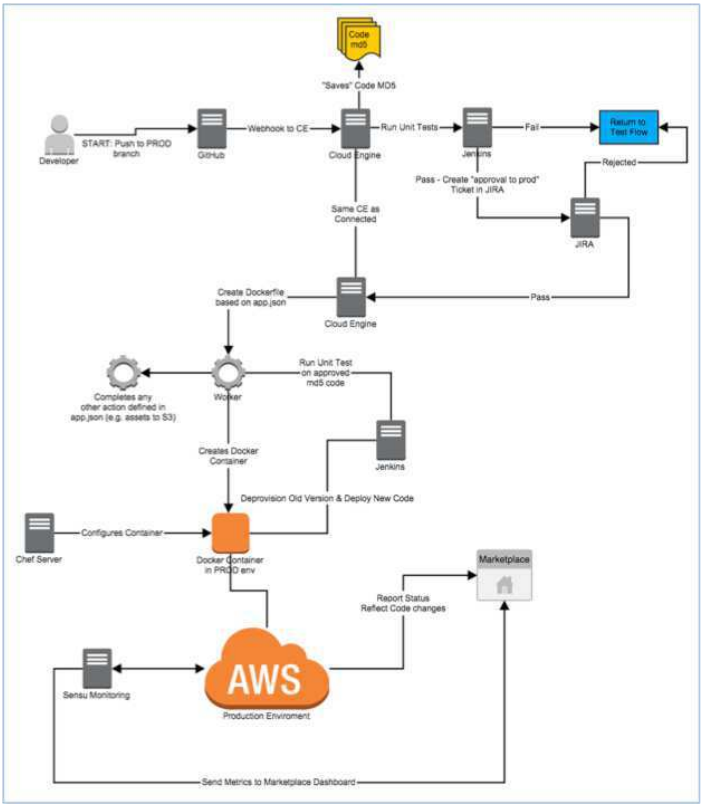
The use case shown in Figure 9, above, describes how the developer will deploy their application code to the component, minimal capacity, and production-like containers. Core Applications and shared CSP services will follow this development use case. Gitflow branching strategy will be used to ensure automated deployments and to facilitate code moving from one environment to the next. This branching strategy is flexible enough to allow developers to have their own feature branch naming strategy and also maintain common named branches for each type of environment deployment.

This process can be broken down into the following six steps:

1. Developer commits code to MC branch in repos
2. GitHub hook initiates build and deployment in CI server
3. Worker creates Docker container based on app metadata (Infrastructure as code
4. Environment in AWS provisions instance of Docker image
5. Instance is connected with monitoring and logging services
6. Issues with deployment logged in JIRA

3.3.5. HST-5 Deploy to Production

Figure 10. Hosting Services Use Case Deploy to Production



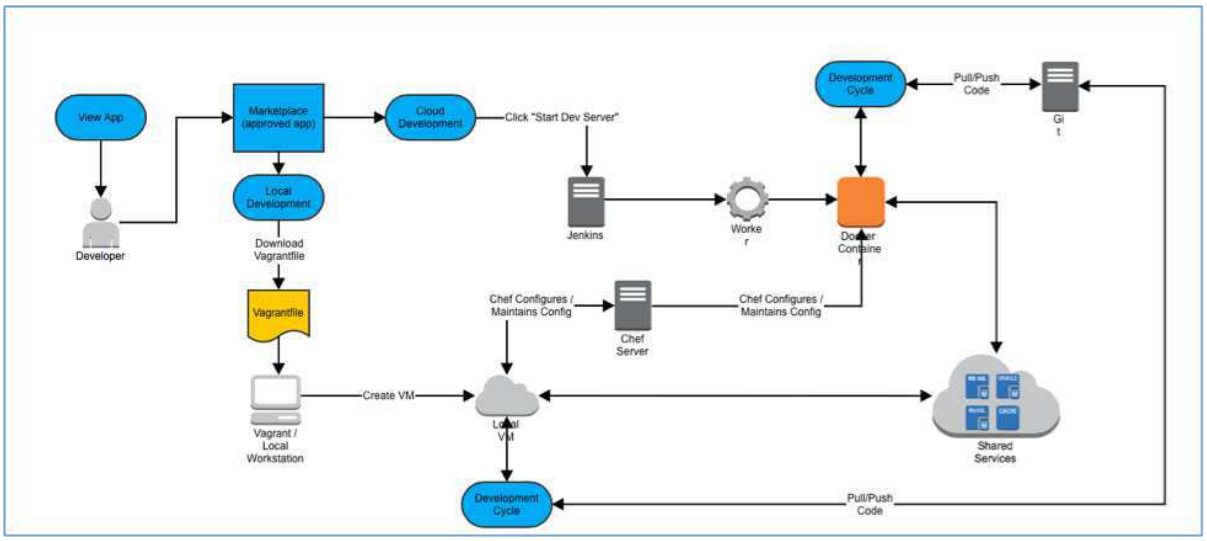
The deployment to production use case is described in Figure 10. Core Applications and shared CSP services will follow this development use case. This process can be broken down into the following five steps:

1. Developer commits code to prod branch in repository
2. GitHub hook initiates build and deployment in CI server
3. Docker image deployed to AWS production environment similar to MC use case
4. Logging and Metrics are enabled
5. Marketplace updated to reflect currently deployed version

3.3.6. HST-6 Development Cycle Flow

This use case allows for a developer to either potentially work locally using Vagrant files, or with peers leveraging the cloud-based hosting environment

Figure 11. Hosting Services Use Case Development Cycle Flow

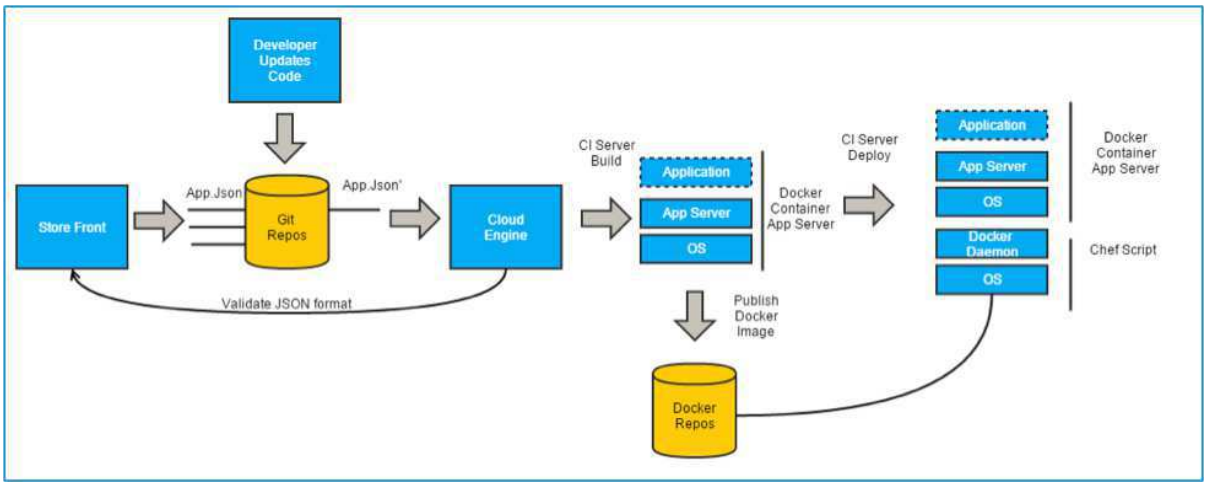


The development cycle in Figure 11 shows the following five major steps in the cycle.

1. View App in Marketplace
2. Download Image for local testing (Optional)
3. Provision instance of Development Environment
4. Assemble application node from code repository
5. Deploy node into container

3.3.7. HST-7 High-Level Deployment Use Case

Figure 12. Hosting Services Use Case High- Level Deployment



High-Level Deployment Use Case from Figure 12:

1. App Developer creates application in CSP Portal which provisions Git with App.json specification
2. App Developer commits updates to Git repos
3. Cloud Engine validates App.json and initiates CI build of Docker Image including application files

4. Docker Image stored in Docker repository
5. CI deploy process creates host via Chef and Docker Container loaded from repos and instantiated on host.

3.3.8. HST-8 Upgrade a Host

Figure 13. Hosting Services Use Case Upgrade a Host

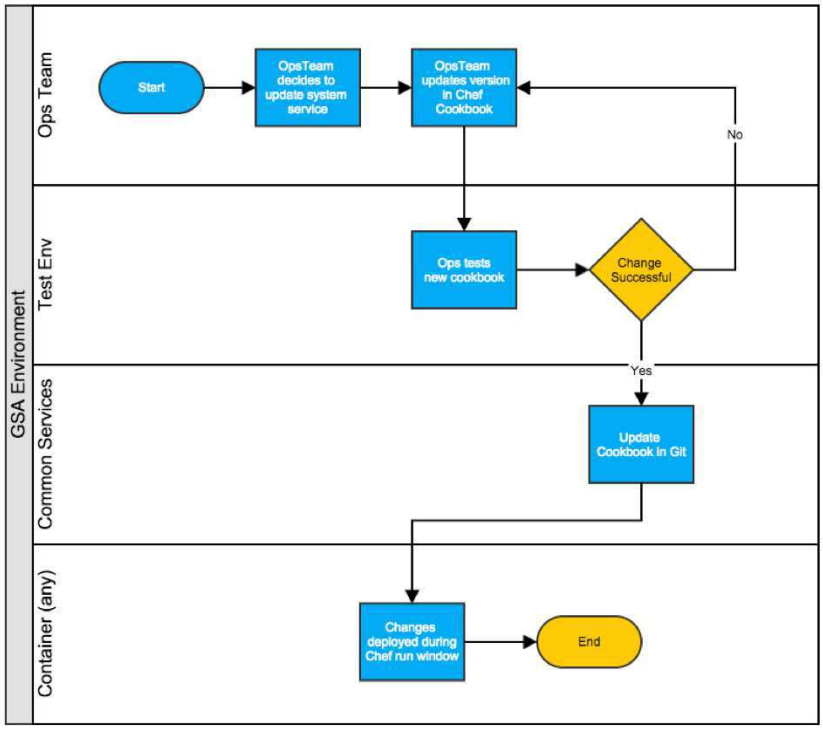


Figure 14. Hosting Services Use Case Upgrade a Docker Container

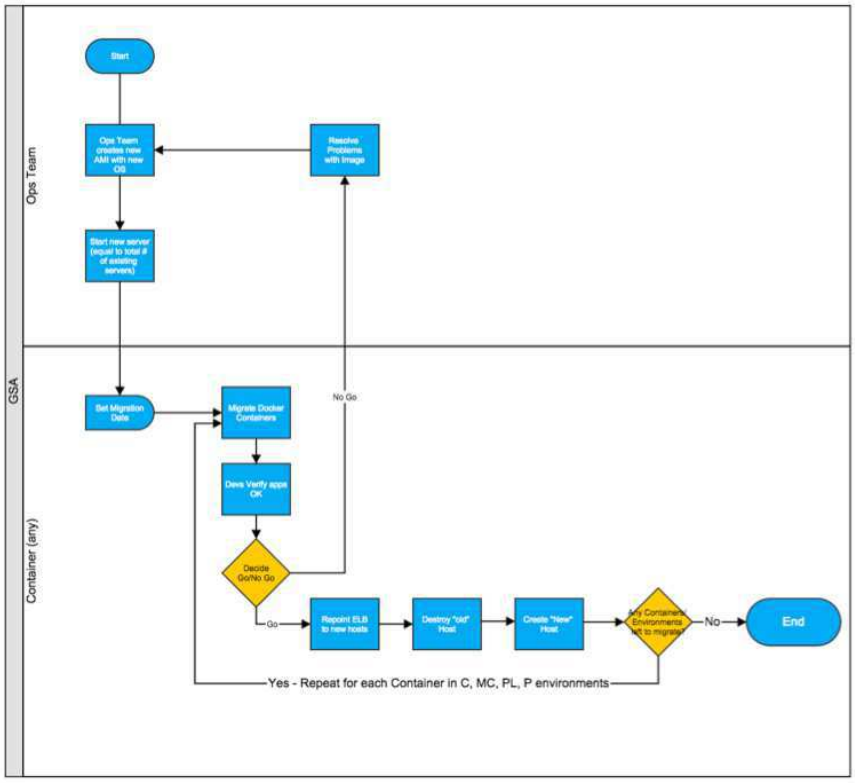


Figure 13 and Figure 14 provide a high-level overview of the upgrade process, for both a Docker Host and a Docker container. This includes:

- Versioned based on the pre-existing host
- Old versions continue to be tracked
- Initiate the deployment to staging environment for IV&V
- Validate the application is working
- Initiate the deployment to production
  - New hosts are added to the pool (Via Chef Cookbook)
  - New hosts operate simultaneous to existing hosts
  - Old hosts are terminated
  - Deployment complete

3.3.9. HST-9 Application Hosting with Docker Containers

A phased approach to hosting Core Applications with Docker containers balances scalability features with the complexity and maturity of management solutions. The phases of this approach are described below in Table 8.

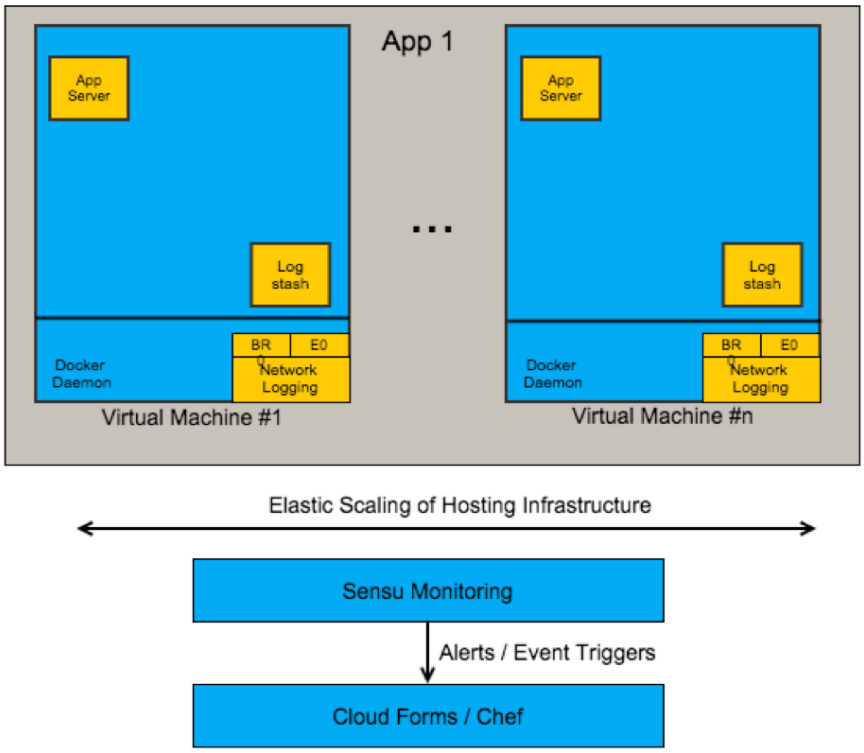
Table 8. Docker Deployment Phases

Phase	Description	Features:	Pros/Cons:
Phase 1	Simple Docker deployment	A fixed number of Docker containers per host. One elastically scaling group of hosts per application deployed to the IAE CS Container	Pros: • Simplifies deployment. • Use of containers supports portability objective and future configurations. Cons: • Does not take advantage of density of host in a dynamic fashion.
Phase 2	Scaling Docker containers within a host	A variable number of Docker containers that scale within a host. One elastically scaling group of hosts per application deployed to a IAE CS Container	Pros: • Takes advantage of density of host in a dynamic fashion Cons: • More complex deployment compared to phase 1

These phases will be correlated to the appropriate release as part of the release planning activities.

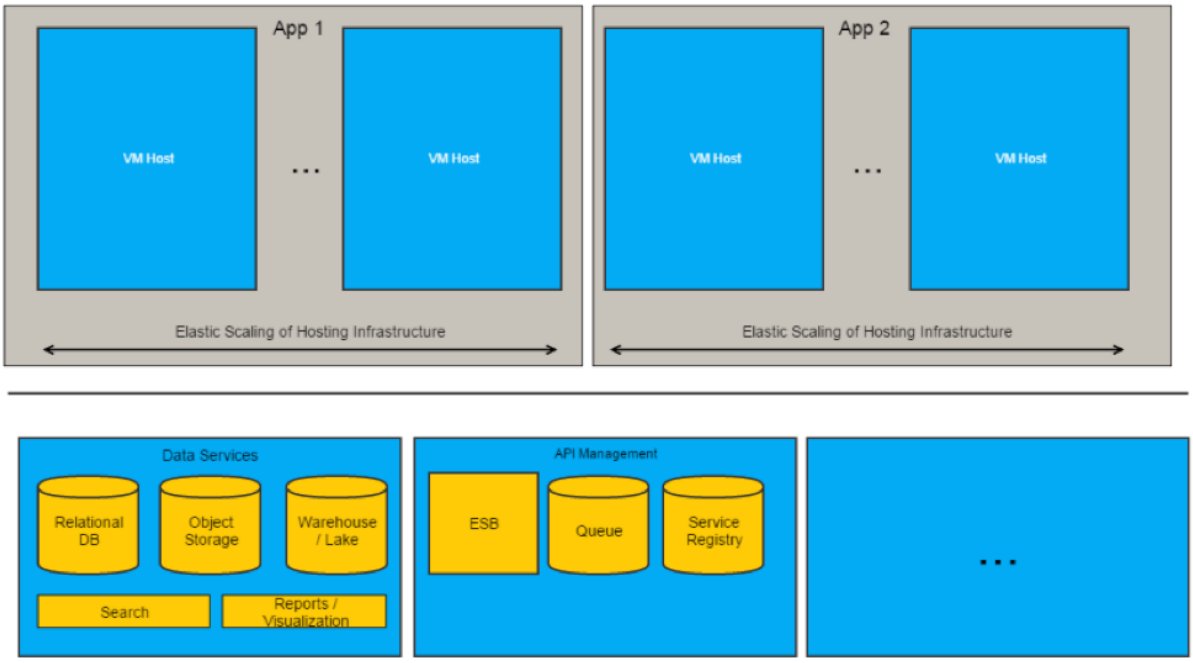
3.4.9.1 Phase 1 – Simple Static Docker Deployment per Application

Figure 15. Hosting Services Use Case Phase 1 Simple Docker Deployment



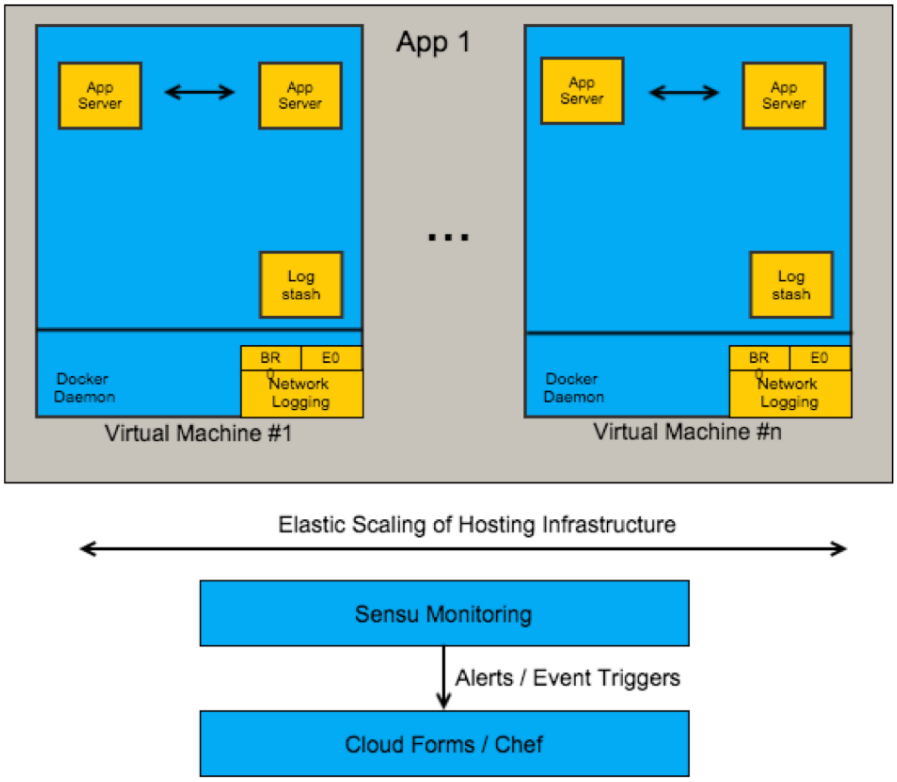
As shown in Figure 15, each host will be configured via Chef automation to spin up a fixed number of Docker containers. The scaling will occur at the host level based on monitoring from Sensu. These events will trigger scaling behaviors in CloudForms / Chef. A Log Stash instance deployed as a container will aggregate logs from all other Docker containers. Network traffic across the Docker containers and the host will be logged.

Figure 16. Hosting Services Use Case Phase 1 IAE CSP Container



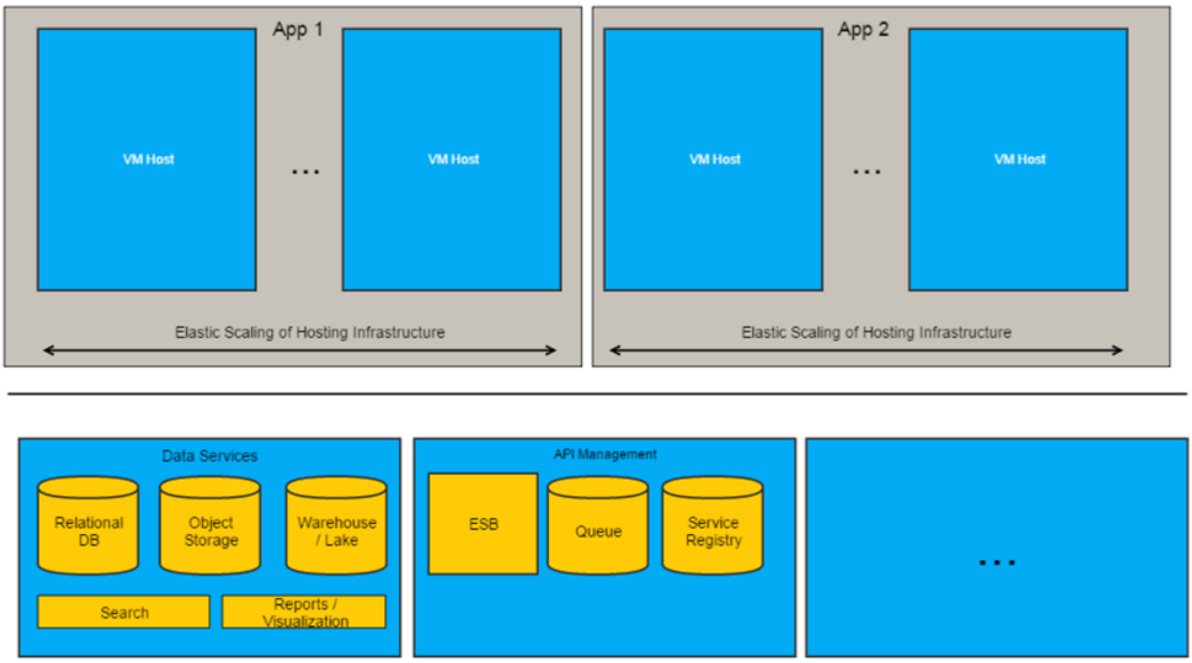
As shown in Figure 16, Each GSA container will include multiple groups of hosting servers to support each individual application or application type. Examples of application types are: App 1 = Java, App 2 = Ruby. The Core Applications will be able to access the Common Services provided by the platform. The common services will be managed initially at the host / Virtual Machine (VM) level

Figure 17. Hosting Services Use Case Scaling Docker Containers



As shown in Figure 17, scaling within a host will help to ensure the appropriate density of Docker containers is achieved to maximize performance.

Figure 18. Hosting Services Use Case Phase 2 Scaling Docker Containers Per App Within Container



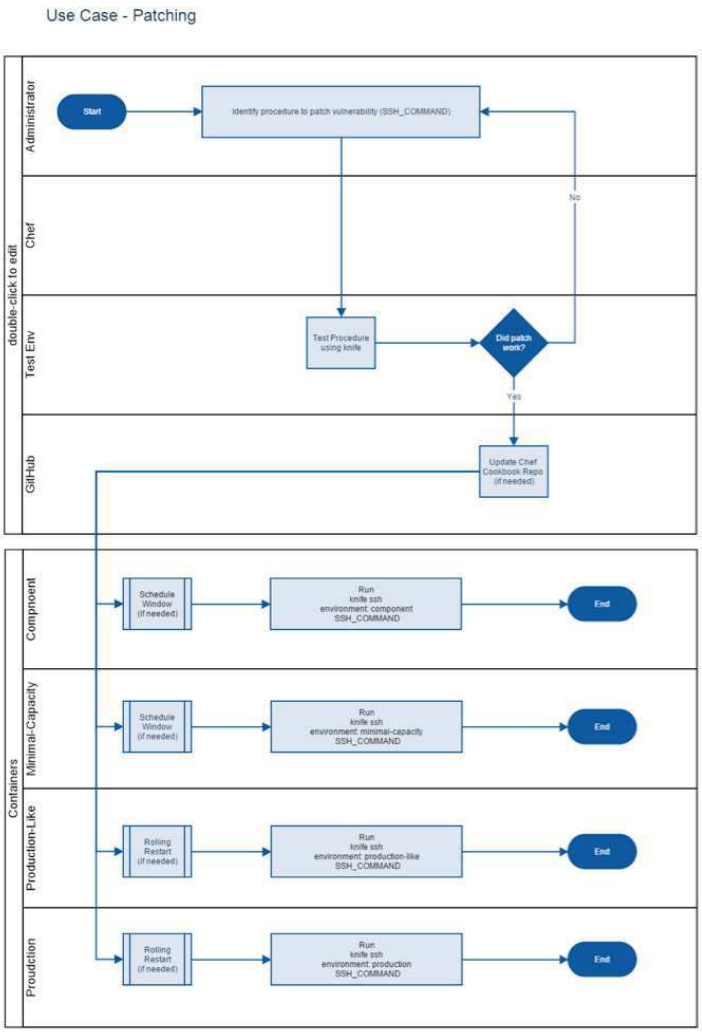
No changes to management of hosts compared with phase 1, as shown in Figure 18.

3.3.10. HST-10 Application Description File

The App.json file within each application’s root enables developers to define their applications' details, setup configurations, and Docker containers in a structured way. This will allow developers to have a good deal of control, while maintaining security, and system reliability by taking the information specified in the App.json file and processing it during deployment; allowing system admins to set limits, and check for any potential issues prior to deployment.

3.3.11. HST-11 Patching

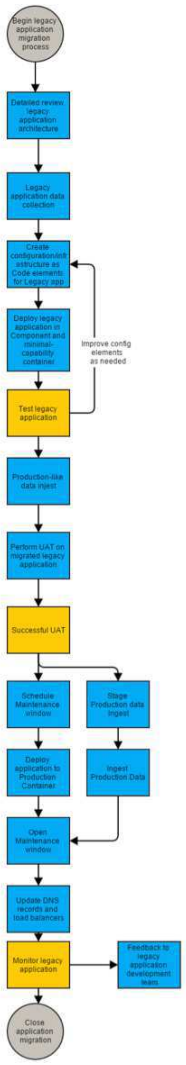
Figure 19. Hosting Services Use Case Patching



This is the Hosting use case for patching the Infrastructure as a Service (IaaS) and PaaS environments, shown in Figure 19, above. The use case shows how the solution will use the configuration management toolset (Chef) to patch a vulnerability, create the relevant patching commands, and then execute those changes in each environment.

3.3.12. HST-12 Deployment of a legacy Application on IaaS

Figure 20. Hosting Services Use Deployment of a Legacy Application on IaaS



The legacy application deployment process is detailed in Figure 20, above. The IAE CS Operations Team/Legacy Application teams perform detailed reviews of the legacy application architecture, collect information on operating system environment and version (e.g. RHEL 6.x), runtime environments and versions (e.g. JDK 1.7), runtime dependencies and libraries, configuration elements (conf files, xml config files), database requirements (e.g. Oracle version features), network architecture, load balancing, session management and firewall configurations. Once the detailed review is complete, an initial set of chef cookbooks are used to deploy the OS environment, runtime environment, configuration files and database environment. The application is deployed using another chef cookbook in the component container and tested. Once application functionality is successfully validated, the legacy application is deployed to the minimal-capacity container. After successful validation, production-like data is ingested into the application through the data services ingest service (e.g. scoop) and deployed in the prod-like container. User Acceptance Testing (UAT) is performed on the legacy application and once complete, the application is deployed to the production container with production data ingested through the data services ingest service. A maintenance window is

opened for the cutover of the legacy application and the DNS and load balancers are reconfigured accordingly.

4. API Management

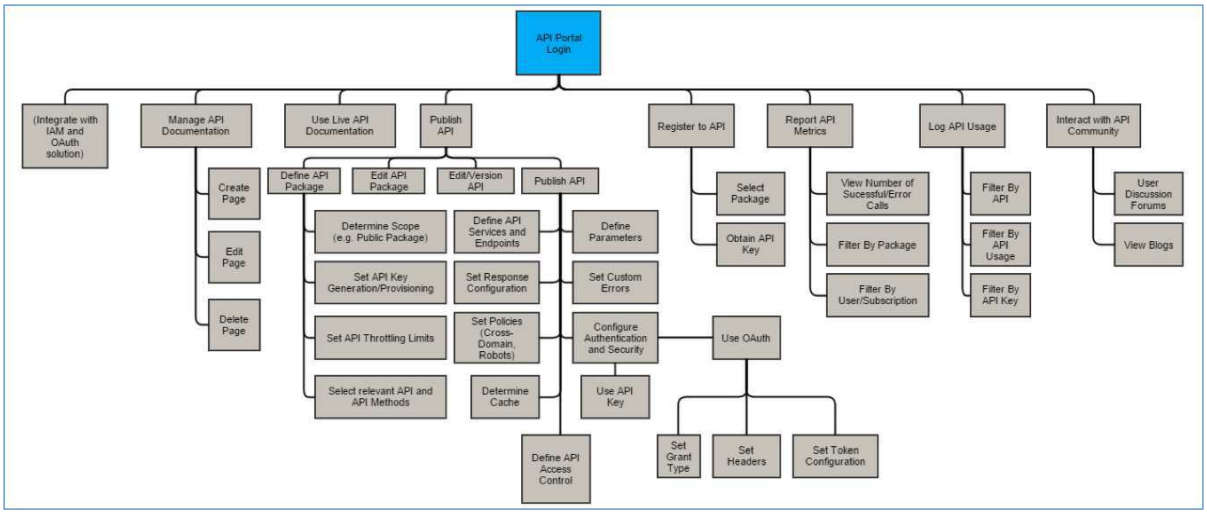
The CSP leverages services and APIs that offer an integrated set of capabilities and functionality that address CSP requirements. Core application services and common services, such as databases and IAM, are made available to other systems through APIs. The following sections describe the architecture for the API Management capability.

4.1. Site Map

API Management user features and functionality can be divided into three areas:

- An API Management portal where administrators and developers publish, view, and subscribe to APIs;
- An API Management gateway that validates API keys and subscriptions and that performs additional security checks before routing requests; and
- An Enterprise Service Bus (ESB) that supports the full set of Enterprise Integration Patterns (EIP).

Figure 21. API Management Sitemap



The API Management portal shown in Figure 21, integrates with the CSP identity and access management solution (IAM) for authenticating and authorizing portal users. The portal serves two primary types of users, API providers and API consumers.

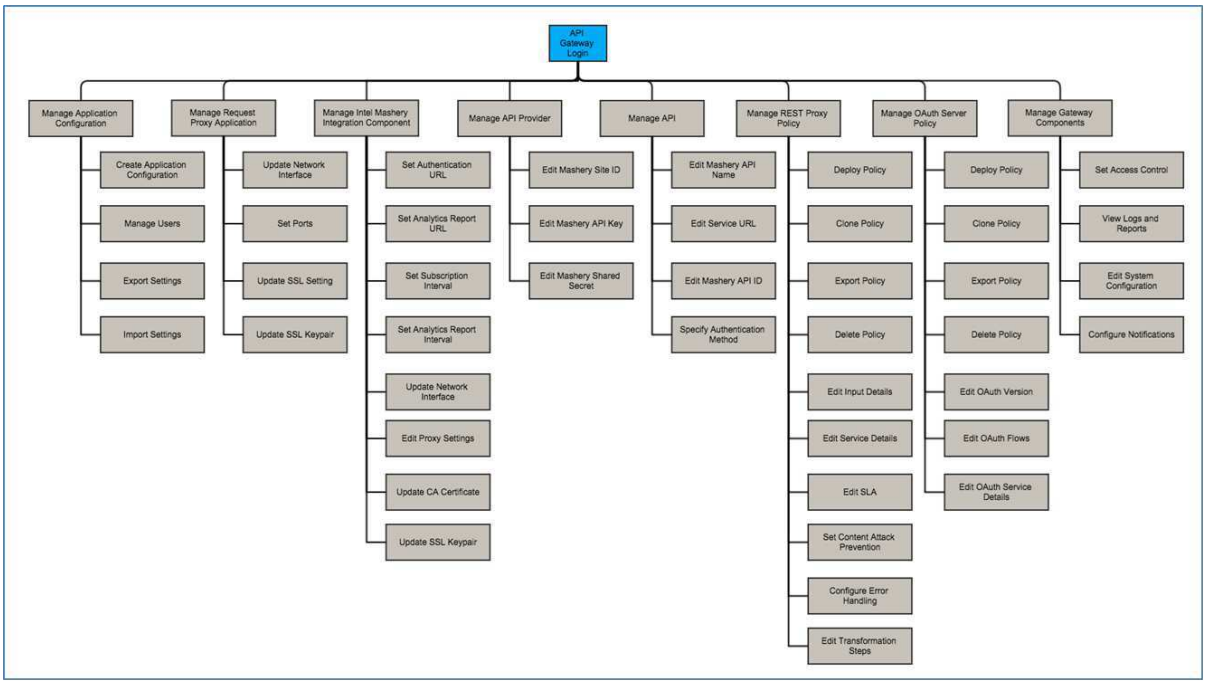
API Providers:

An API provider uses the portal to publish APIs and offer different levels of subscription packages for using the API. A subscription package sets the maximum number of concurrent requests and maximum number of requests per month that are available to subscribers. API publication and configuration options include setting API service endpoints, selecting caching options, and defining API security (e.g. requiring that consuming applications use OAuth). An API provider also publishes documentation that describes the API, its uses, example requests, and example responses.

API Consumers:

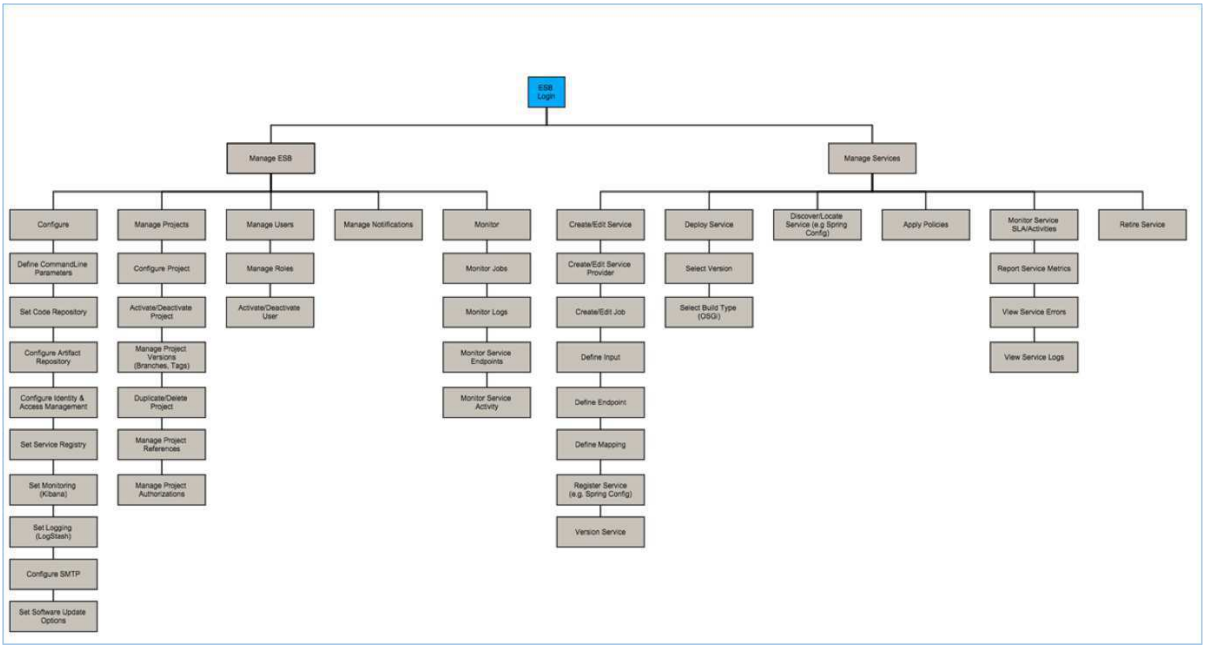
API consumers are those who wish to gain access to the functionality exposed through the API Management portal. These API consumers see the available APIs published on the portal and subscribe to relevant packages to obtain API subscription keys. A subscription key allows a consuming application or system to utilize the API up to the usage levels defined in the package. An API consumer may also experience the API through a user interface that supports test requests and displays corresponding API responses.

Figure 22. API Management Gateway Sitemap



The API gateway manages API calls for end-user applications as well as system-to-system integration as depicted in Figure 22, above. The gateway acts as a checkpoint for validating API subscription keys and usage limits (e.g. maximum number of concurrent requests). In addition, the gateway verifies that a request does not contain malicious content to ensure that system security is maintained prior to routing the request to the CSP ESB. Gateway configuration covers integration with an API portal, API policies such as OAuth security and error handling, and settings for network interfaces and service routing.

Figure 23. API Management ESB Sitemap



As shown in Figure 23 above, services can be registered with the ESB to be consumed by other applications and services within the CSP. Managing an ESB includes functionality to cover configuration, ESB project management, user management, notifications, and monitoring. ESB configuration includes defining parameters and settings for identity and access management, service registry, monitoring, and logging. ESB project management offers capabilities to organize service development activities into projects that are deployed and implemented on the ESB. User management allows ESB administrators to define roles and assign ESB users to those roles. ESB monitoring functionality enables administrators to see details about existing jobs, logs, service endpoints, and activity.

From a developer perspective, the ESB provides a solution for discovering, creating, and deploying services. Creating a service involves defining service endpoints, inputs, outputs, and mapping. Once a service is created, a developer registers the service in the ESB service registry, which makes the service discoverable and available for use. Service registration and discovery can take place programmatically or through service configuration files (e.g. Spring config). Registered services are monitored by the ESB, with reports and analytics available to ESB administrators and other authorized users.

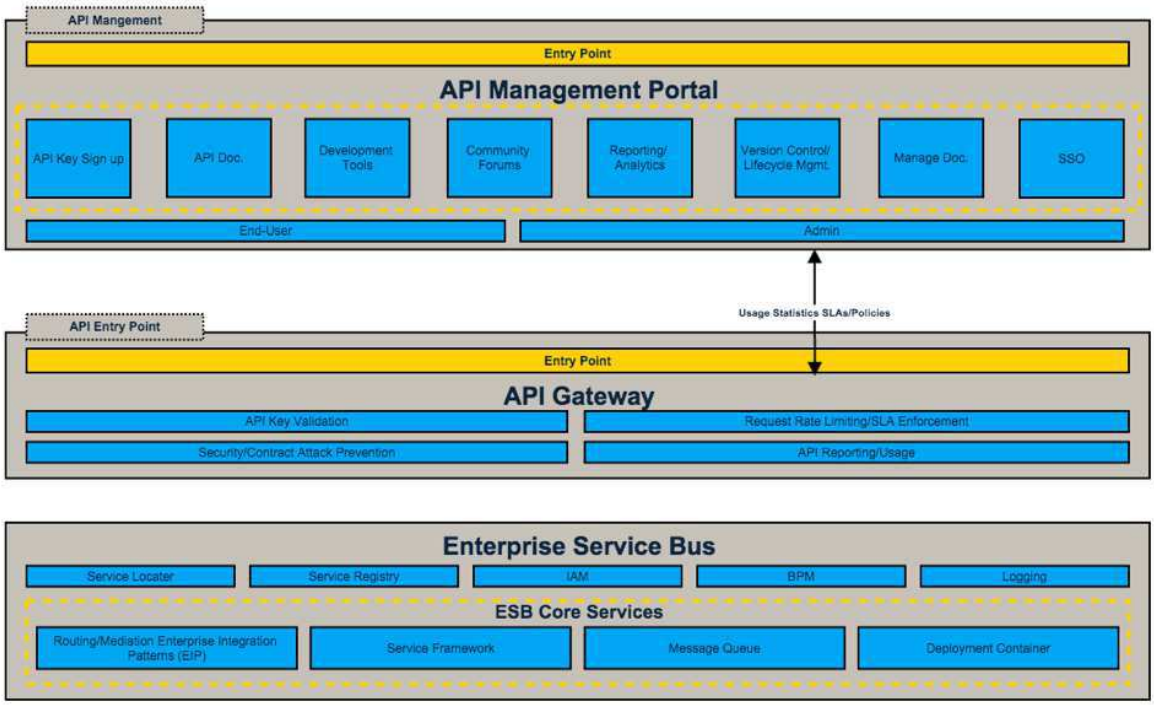
4.2. Logical Architecture

Building on the CSP logical architecture, the API Management functional stack, as seen in Figure 24, is separated into portal, gateway, and enterprise service bus (ESB) areas. The Portal provides the user-facing web interface to API Management features, such as the ability to browse API documentation, obtain API keys through API subscriptions, and view API analytics and reports. Working in conjunction with the portal, the gateway checks and validates API requests against subscription details and applies enterprise security policies across APIs in a consistent manner. The use of API keys and subscriptions allows for fine-grained tracking of requests and transactions.

Version management of APIs is supported by both the API Management Portal as well as the ESB. For example, two versions of the same API may be supported simultaneously on the CSP to support

users of the current version of an API as well as to expose a new version of the API. A version of an API in the API Management Portal may be linked to a version of an API exposed through the ESB and provided by the Core Applications.

Figure 24. API Management ESB Functional Stack



4.2.1. Assumptions and Constraints

Assumptions:

- REST/SOAP: Newly migrated Core Applications should expose RESTful APIs. However, legacy applications re-hosted onto the CSP can also expose SOAP interfaces these will be properly exposed and routed through the API Gateway and API Management layers.
- Orchestration: ESB will not perform orchestration logic of core app services in dynamic way. It will simply pass requests to either internal or external/public.
- Software as a Service (SaaS): The API Management portal component can be provided through a SaaS solution.
- Regional Reporting: The use of API call reports by region within the API management solution may be managed by offering specific API subscription packages for each region
- Deprecation: Deprecating APIs is a matter of managing release plans, source code, schedules, and deployment activities.

4.3. Physical Architecture

Several commercial-off-the-shelf (COTS) and Open Source API Management portal solutions are available. As a result, API Management portals differ across Federal government web sites (e.g. data.api.gov, data.nasa.gov). However, few options offer a hybrid deployment model for API

Management functionality. For the CSP, the selection of an API Management portal and gateway focuses on two options. The first option is an Open Source solution that is used on data.api.gov and with support from GSA. The second option other is a leading commercial solution used by large organizations, such as Coca-Cola and Sabre.

4.3.1. Portal and Gateway

API Umbrella is an Open Source solution released under the MIT license. The solution was originally developed for the National Renewable Energy Laboratory (NREL) to expose and manage agency web services as APIs. It has also been used on data.gov to provide a means for developers to subscribe to and use APIs from Federal agencies. As an Open Source solution, there is an available GSA Git branch of the codebase. Table 10 outlines the various dependencies, strengths, and weaknesses of the API Umbrella option.

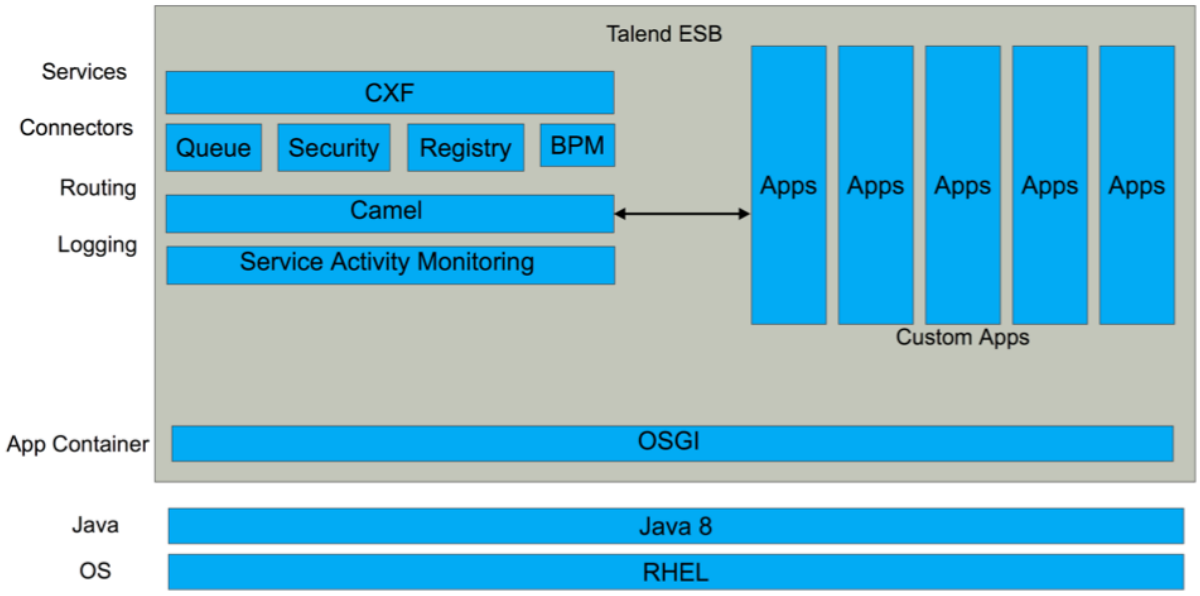
Some of the key advantages of Umbrella are:

- Access to source code as an Open Source solution
- Used on existing Federal government sites (e.g. api.data.gov, developer.nrel.gov)
- No licensing/subscription costs
- Available for major Linux distributions (e.g. Red Hat Enterprise Linux, CentOS, Ubuntu)

4.3.2. Enterprise Service Bus

Figure 25 presents the various components of the Apache ESB stack supported by Talend. The applications deployed within the ESB shown in the figure are not Core Applications, but rather custom applications. The ESB, through the OSGI application container, allows to extend its behavior, and deploy custom applications and code within the ESB to provide those capabilities.

Figure 25. API Management High-Level ESB



Due to the fact that the ESB is stateless, its states are kept in queues that must be replicated to maintain state. Different availability options for the queue are being explored.

4.4. Use Cases

Table 9, below provides a brief summary of all API Management-related use cases.

Table 9. API Management Use Case Listing

ID	Use Case Name	Description	Actors	Pre-condition	Post-condition
API-1	Publishing an API	Steps and Decisions for publishing an API	Developer	API is unavailable	API is published and available for use

4.4.1. API-1 Publishing an API

As shown in Figure 26, publishing an API requires that an underlying service is available and can be made accessible to other systems through the API gateway. As the first step, the API request (i.e. input) configuration needs to have available service endpoints defined. In the case of services exposed by Core Applications, these endpoints will be located on the ESB. Required and optional service parameters must also be defined in the API request configuration. After setting the request configuration, the response (i.e. output) configuration should be set to indicate the format and types of data being returned, including any custom error information. API security options available within a typical API gateway include requiring the use of an API key (e.g. subscription package) and/or OAuth-based authorization. Other API publication configuration settings cover cross-domain usage, caching, and access controls that define who can manage, change, and subscribe to the API.

Figure 26. API Management Use Case Publishing an API

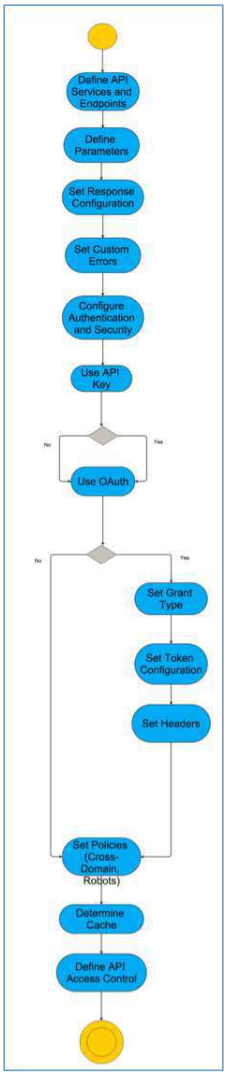


Figure 27. API Management Use Case High-Level ESB Logging Architecture

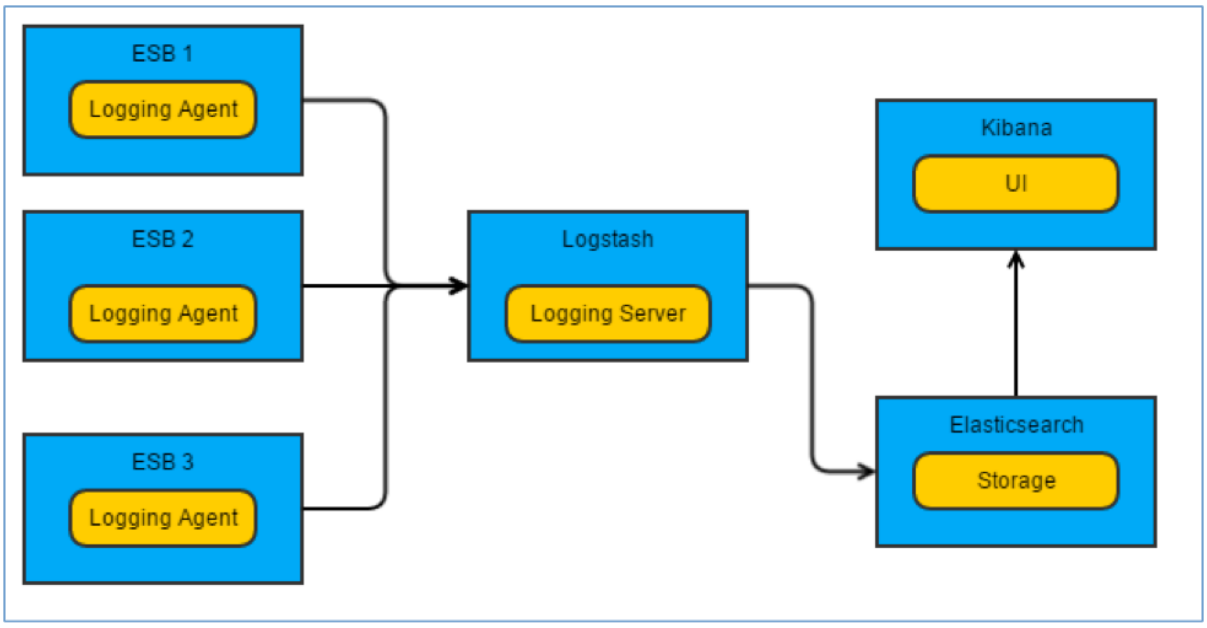


Figure 27 shows that the ESB will contribute data to Logstash, the CSP's centralized logging tool that aggregates all logging data. Logstash connects to Elasticsearch, where the aggregated log data is stored and available to be searched at any given time. Kibana is a data visualization tool for data collected by Logstash. Kibana is used to monitor the logging information that pertains to the Common Services Platform.

Figure 28. API Management Use Case Queue High Availability Configuration

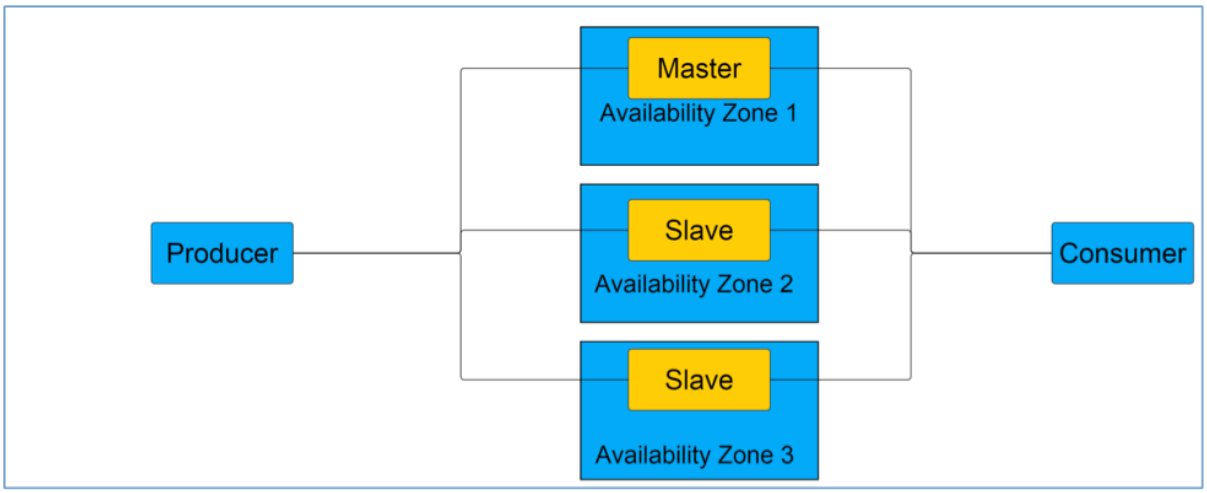


Figure 28 shows that ESB manages its states through the use of queues that must be replicated to maintain state. In this example, the master queue is replicating the slaves. The ESB has the address of each queue and is able to resolve to the next queue upon one failing over without the loss of information.

5. Data and Search

Data ingest, processing, and management along with efficient search is an integral part of the Common Services Platform (CSP). The purpose of this section is to provide a drill down view into the data and search work streams, and how they ultimately connect and operate together. Utilizing our previous experiences and best practices; the core component of the data work stream will be implemented using industry recognized tools that integrate together to create a Data Lake. The Data Lake is centralized repository that integrates data of all types and formats into a single, logical namespace. This Data Lake will centralize all data of the CSP and will act as a highly available and fault tolerant data warehouse that will store both Application data and data from current IAE legacy databases. It will also provide a platform for advanced analytics, where users will be able to create and discover analytics, and track data and CSP metrics through business intelligence tools.

The Applications that interact with the data work stream will directly interface with two data stores: a relational data store and a document-oriented data store. The former is used to store and manage relational and transactional data, while the latter stores data sets that are structured as documents. These data stores will consistently sync data to the Data Lake for centralized storage and analytics. The data stores will have their own security mechanisms (managed by the IAM services) such as role based access control and any sensitive data (such as personally identifiable information) will be secured appropriately using cell-level security.

Search will be powered by the Elastic Search query engine, which provides a distributed and fast searching compatibility driven by RESTful APIs. The search work stream will very closely integrate with the data work stream as Elastic Search will store all data copies in distributed indices for high availability and rapid response time search. The search work stream will support various types of search options such as full text, Boolean search, and suggestive (Google-like) search.

5.1. Site Map

Figure 29. Data and Search Data Lake Sitemap

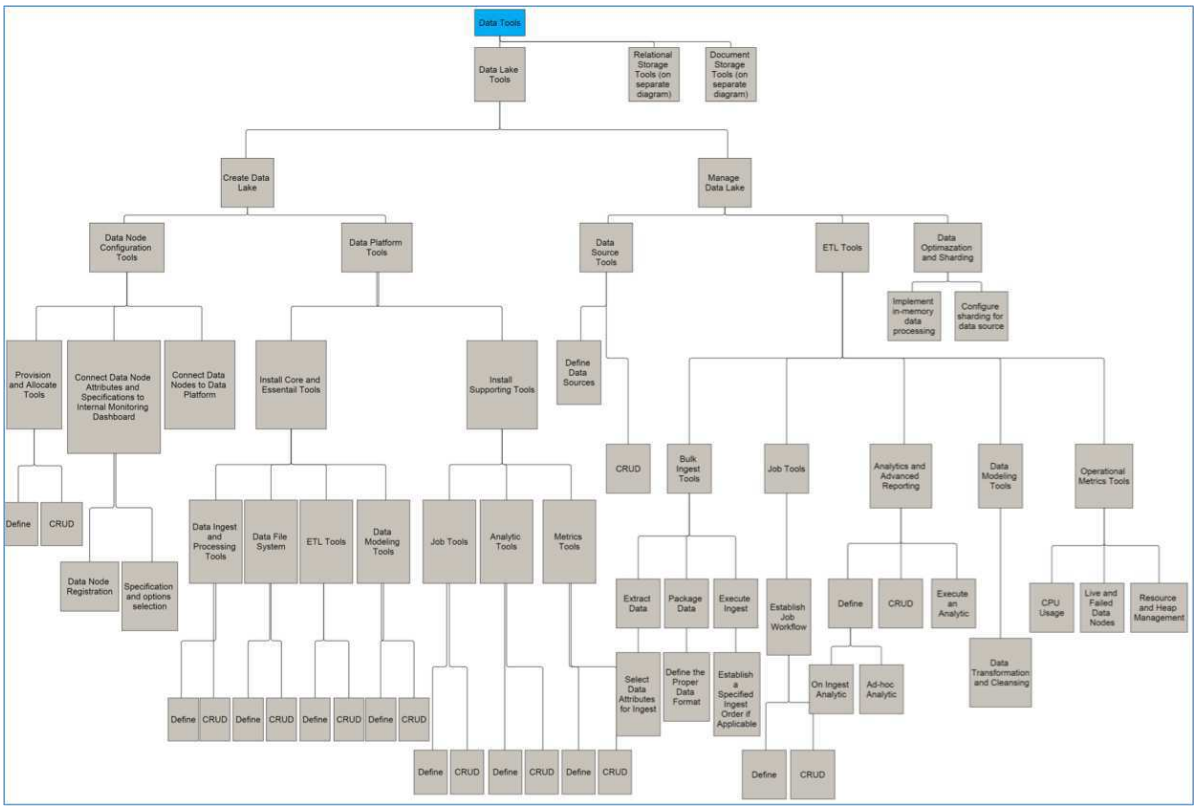


Figure 29, above, provides a structured breakdown of the tasks and features needed to create and manage a Data Lake. The Sitemap decomposes each task into subtasks recursively until it reaches the most basic subtask needed. This approach provides an overall view into what's needed for completing the instantiation of a Data Lake and what the typical required tasks are to manage a Data Lake once it's implemented successfully.

Figure 30. Data and Search Relational and Document Storage Tools Sitemap

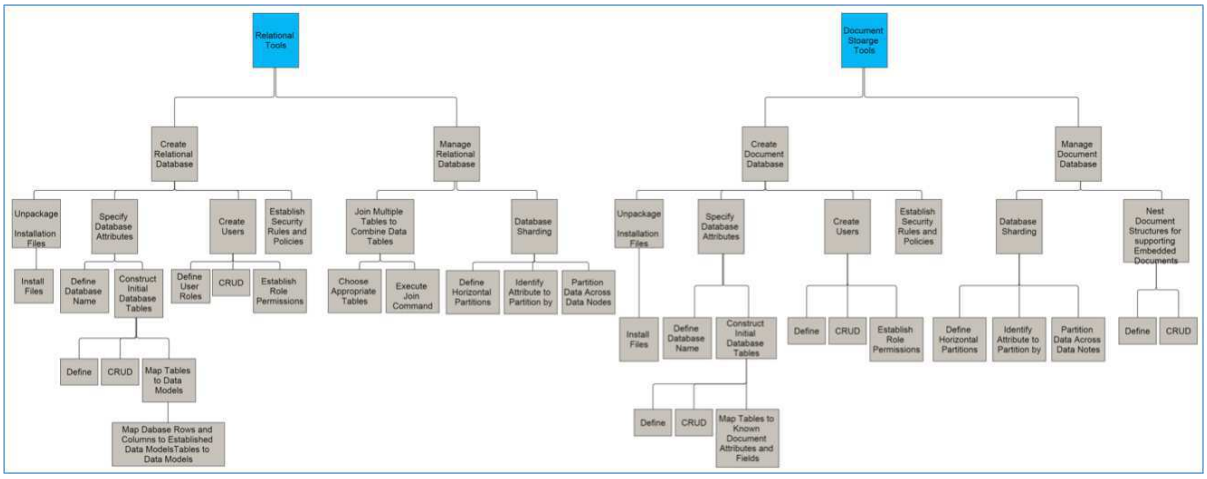


Figure 30, above, provides a structured breakdown of the tasks and features needed to create and manage a relational data store and document-oriented data store. The Sitemap decomposes each task

into subtasks recursively until it reaches the most basic subtask needed. This approach provides an overall view into what's needed for completing the installation of each data store and tracks what the typical tasks are for managing the two data stores. Even though each data store will include similar tasks, there are subtle differences that are captured in the Site Map.

Figure 31. Data and Search Search Tools Sitemap

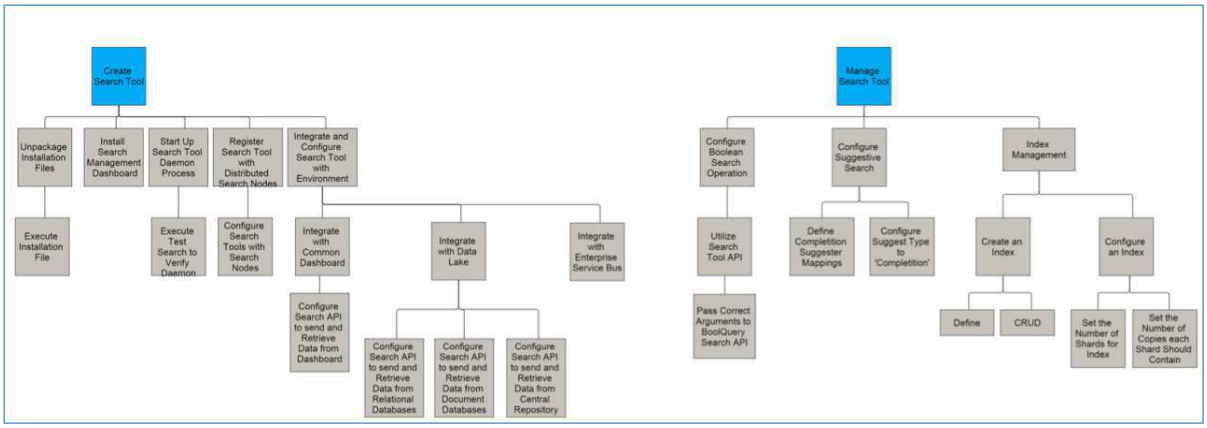
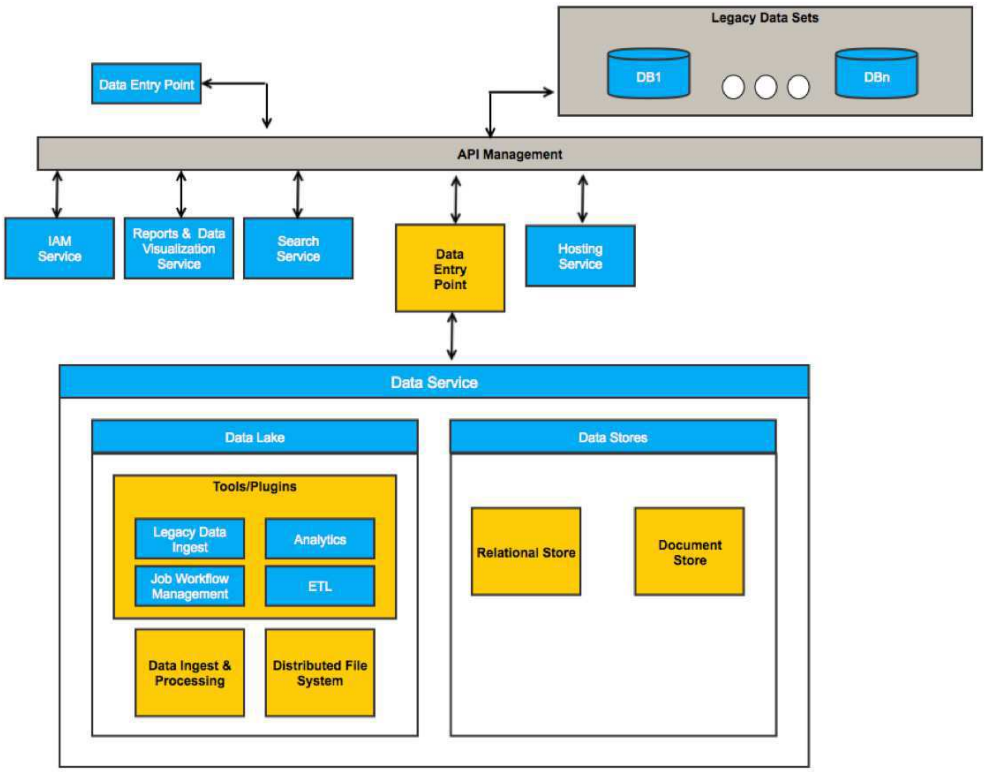


Figure 31, above, provides a structured breakdown of the tasks and features needed to create and manage the search tool that will enable the ability to search through the entire CSP. The Sitemap decomposes each task into subtasks recursively until it reaches the most basic subtask needed. This approach provides an overall view into what's needed for completing the installation of the search tool and what the typical tasks are for the managing the search tool effectively.

5.2. Logical Architecture

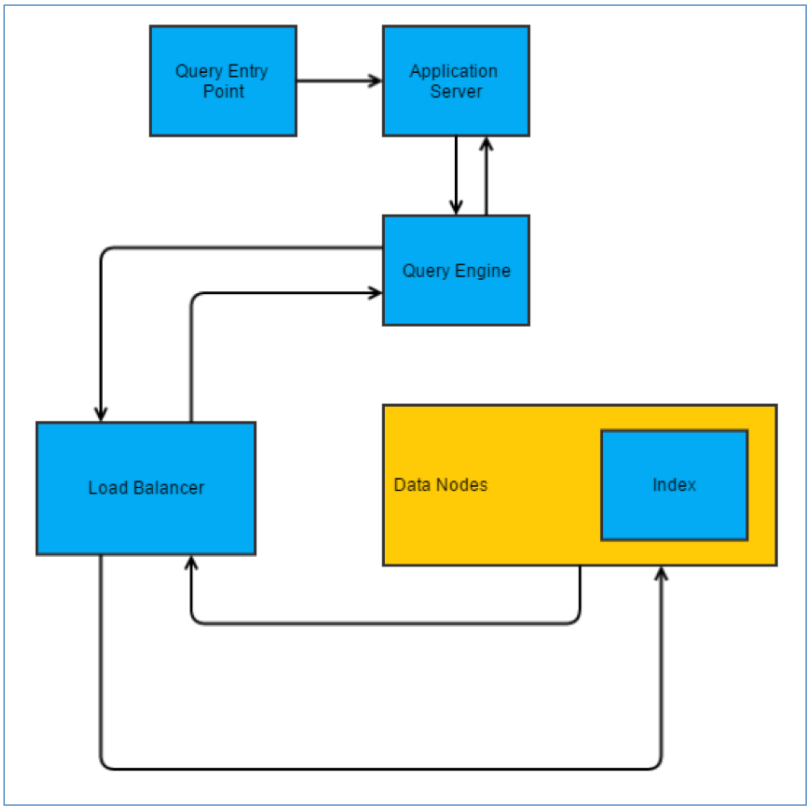
5.2.1. Data Logical Architecture

Figure 32. Data and Search Logical Architecture



The legacy database will feed data into the Data Lake for centralized storage and analytics. Within the Data Lake, there will be various tools and plugins. At the core of the Data Lake will be the data ingest and processing framework (which will Map Reduce or a comparable framework) and the distributed file system. The data stores that the Applications interact with will transmit data into the Data Lake and data can be sent back to the data stores as needed.

Figure 33. Data and Search Search Logical Architecture



The Applications will interact directly with the search query engine. This query engine will communicate with the load balancer that will be orchestrating each data node. A data node will contain a specified index and will be responsible for storing and maintaining the index (e.g. make sure the index is up to date, run auto indexing tasks, etc.), which will contain the data and search results.

5.2.2. Assumptions and Constraints

Assumptions

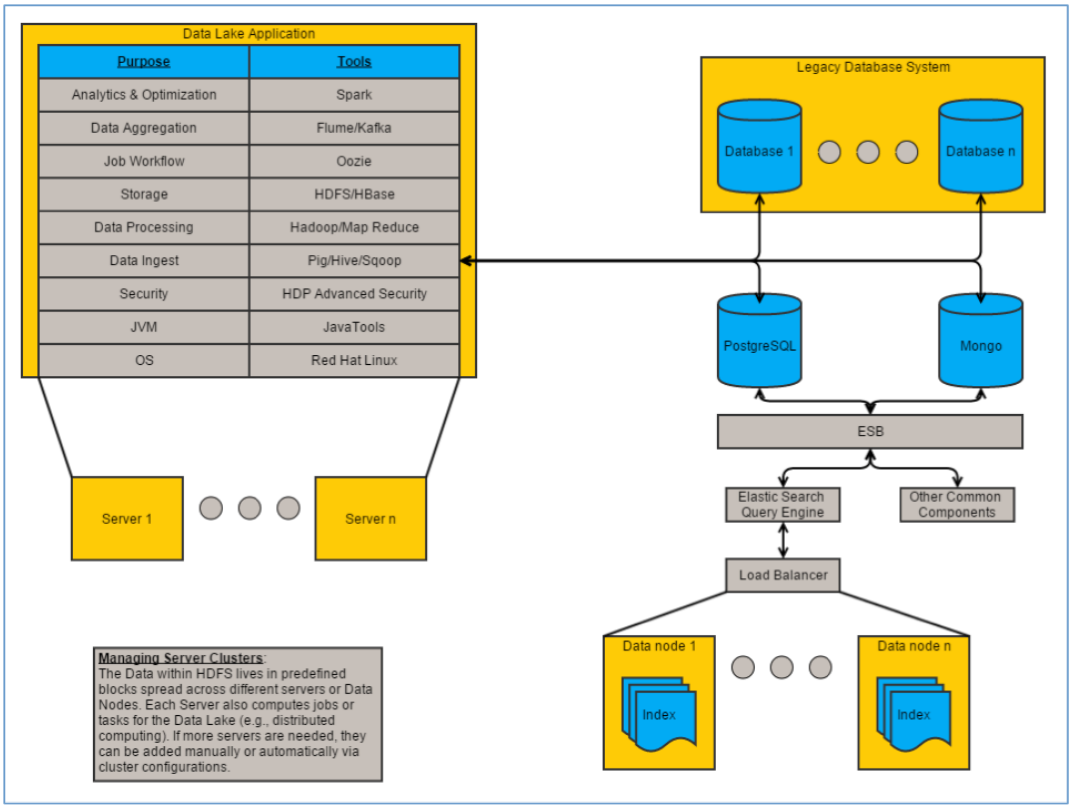
- The legacy databases will not interface with Sqoop for data transfer.
- Sqoop will only be used to transmit data into the Data Lake from the native CSP data stores.
- The data flow will be from the applications to the data stores (relational and document) to the Data Lake for centralized storage and analytics.

Constraints

- The Core Applications be able to access data directly in the Data Lake through Business Intelligence (BI) tools to view metrics and analytics.

5.3. Physical Architecture

Figure 34. Data and Search Physical Data and Search Architecture



Data will be transmitted from the applications to the PostgreSQL and MongoDB (or comparable) data stores. Data will then be transmitted to the Data Lake via Sqoop for centralized storage and analytics. Data within the Data Lake is stored within Hadoop's file system HDFS. Data in HDFS is organized by blocks (by a predetermined block size via Hadoop configurations) and stored across various servers (data nodes). Similarly to how data is stored in HDFS, data stored in the Elastic Search indices will be spread across various servers.

Figure 35. Data and Search Search Indexing and High Availability

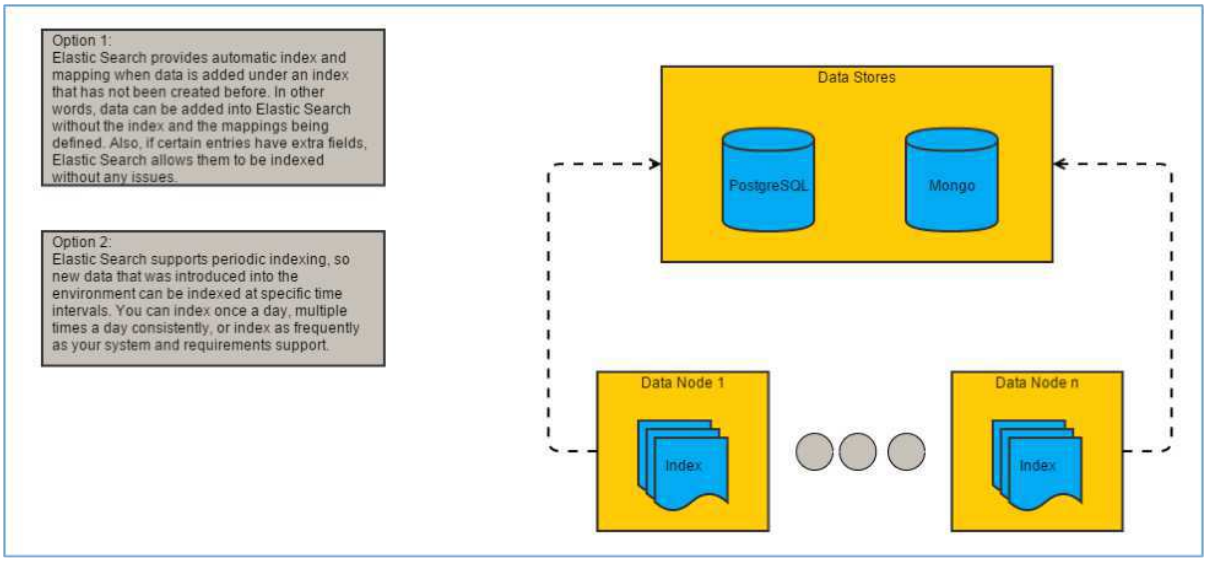
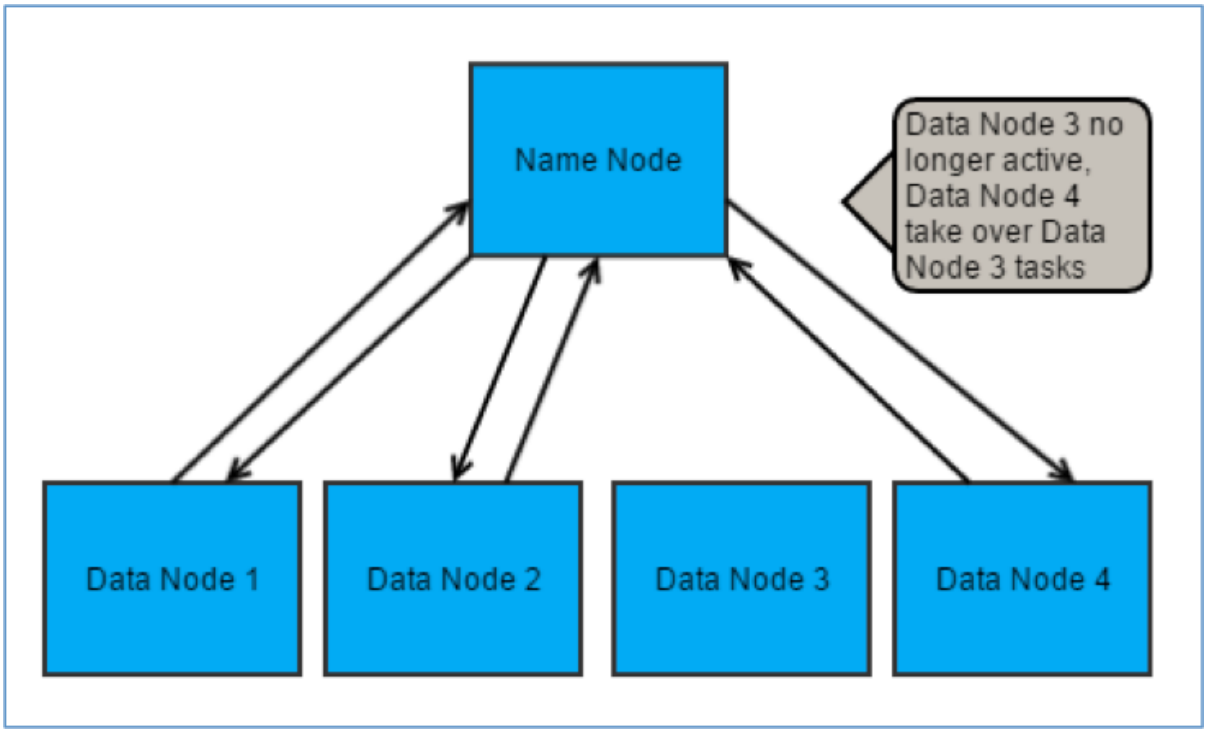
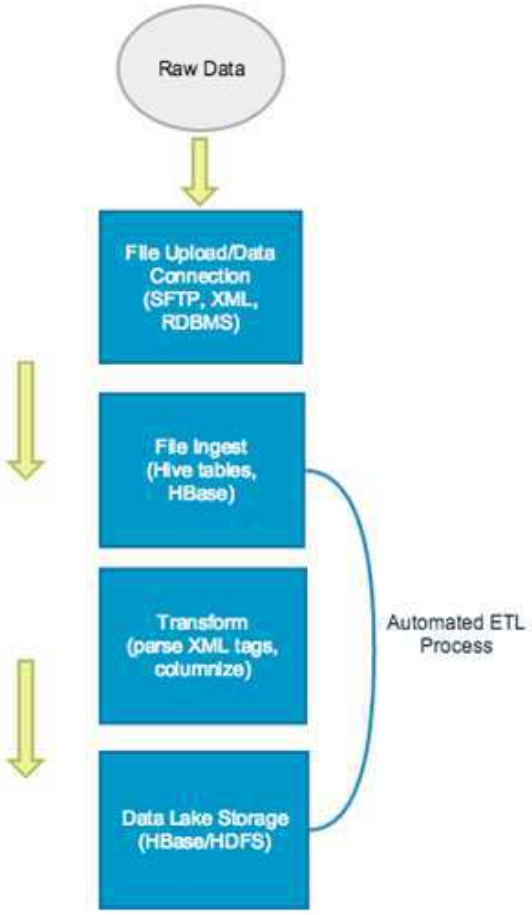


Figure 36. Data and Search Data Lake High Availability



The Data Lake uses Hadoop as its distributed data ingest and processing framework. Data that gets processed by Hadoop is stored on many different data nodes, and each data node is maintained by a master node called the Name Node. The Name Node monitors the life of each data node through constant heart beat communication. When it recognizes that a data node has failed or has become inactive, it orchestrates the remaining data nodes to compensate for the inactive node. This achieves high availability and provides fault tolerance for the Data Lake.

Figure 37. Data and Search Automated ETL Process



When data is ingested into the Data Lake (via applications that use a data store), as shown in Figure 37 above, it will go through an ETL process before being stored. If the data is represented in XML, then the ETL process will be automated so the core application developer does not have to write boilerplate XML processing tasks before getting the data into the environment. This ETL framework provides major benefits for groups such as WDOL, FPDS, and CFDA as all data sets across those groups are in XML format. The automated ETL process can be analyzed in four steps:

- File Upload / Data Connection – Data needs to be introduced to the Data Lake through some mechanism. Uploading could occur through the marketplace UI (such as the WDOL drill down) or a connection to a data store or data feed (e.g. FPDS ATOM feed).
- File Ingest – The data will be ingested into a ETL tool such as Hive to be prepared for transformation. At this stage, the data has entered the Data Lake environment.
- Transform – The data will be parsed of XML tags (via Java libraries like JAX-P for DOM manipulation) so only the raw data is operated on. The relationships amongst the data attributes

will be maintained and the data sets will be columnized so they can be successfully stored in HBase tables.

- Data Lake Storage – After the transformation step is complete, the data will be stored in one or more HBase tables and automatically stored on HDFS (HDFS is the file system HBase uses to store its contents).

5.4. Use Cases

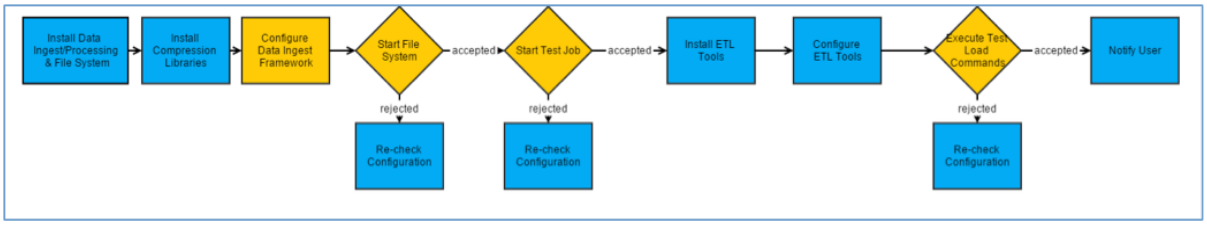
Table 10. Data and Search Use Case Listing

ID	Use Case Name	Description	Actors	Pre-condition	Post-condition
DAT-1	Install Core & Essential Data Platform Tools	Install the minimum essential Data Lake tools to activate the Data Lake	CSP Data Developers	No data platform tools installed	Basic Data Lake infrastructure completed that allows data processing and ETL
DAT-2	Create Relational Database	Install the relational data store	CSP Data Developers	No relational data store installed	Relational data store installed with user access permissions configured
DAT-3	Data Import from Data Stores to Data Lake	Transmit data from the relational and document data stores into the Data Lake	Application Developers	Data only resides in the data stores	Data within the data stores can now be transmitted into the Data Lake.
DAT-4	Data Export from Data Lake to Data Stores	Transmit data from the Data Lake into the relational and document data stores	Application Developers	Data in the Data Lake does not get ingested into the data stores	As needed, data can transmit from the Data Lake into the data stores
DAT-5	Enable Boolean Search Capability	Enable the Boolean search capability to add search capabilities	CSP Data Developers	Boolean search is not currently enabled; only other search operations are supported.	Boolean search is enabled and provides a new search option for users
DAT-6	Search Specific Index via Elastic Search API	Search for a data set given a specific index and ID number	Application Developers	Users will search for data through the	Users can now search for particular data sets via

ID	Use Case Name	Description	Actors	Pre-condition	Post-condition
		that maps to the specific data set.		appropriate search interface	RESTful URLs.
DAT-7	Elastic Search Auto Indexing	Indexing job is executed when new data is stored into the environment.	CSP Data Developers	Only manual indexing occurs	New data will be indexed after being ingested and stored into the environment.
DAT-8	Migration of Redacted data from Prod to Non-Prod Container	Redact and copy data from the data stores in the Prod IAE CS Container to the data stores in the Prod-like container	Application Developers	Data is not transmitted between containers	Data can be transmitted between data stores across containers.
DAT-9	Relational Data Continuous Integration / Deployment 1	Process for an Application developer to create a database schema or manage their database through the CSP Portal	Application Developers	Database management cannot occur through the CSP Portal	A database schema or task can be created through the CSP Portal, be properly controlled in Git, and verified through the CI process
DAT-10	Relational Data Continuous Integration / Deployment 2	Show the different options that exist for multiple Applications to maintain their database schemas	Application Developers	Application Developers only have a single environment for maintaining their database schemas or may not have maintenance flexibility	Application Developers can maintain different schemas across various data management options

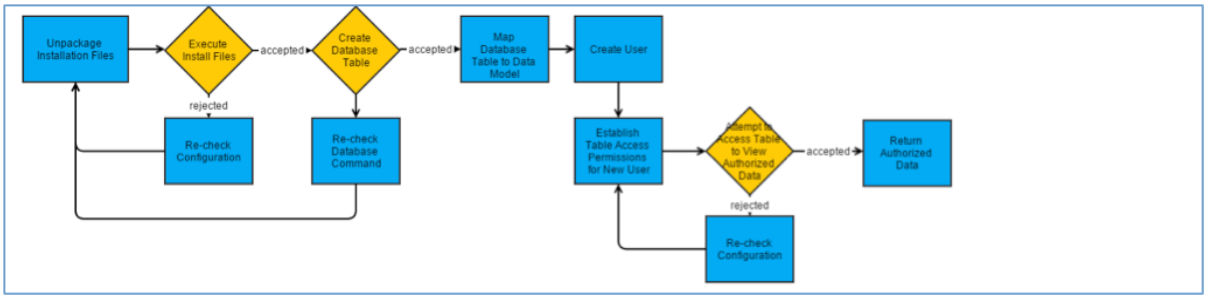
5.4.1. DAT-1 Install Core & Essential Data Platform Tools

Figure 38. Data and Search Use Case Install Core and Essential Data Platform Tools



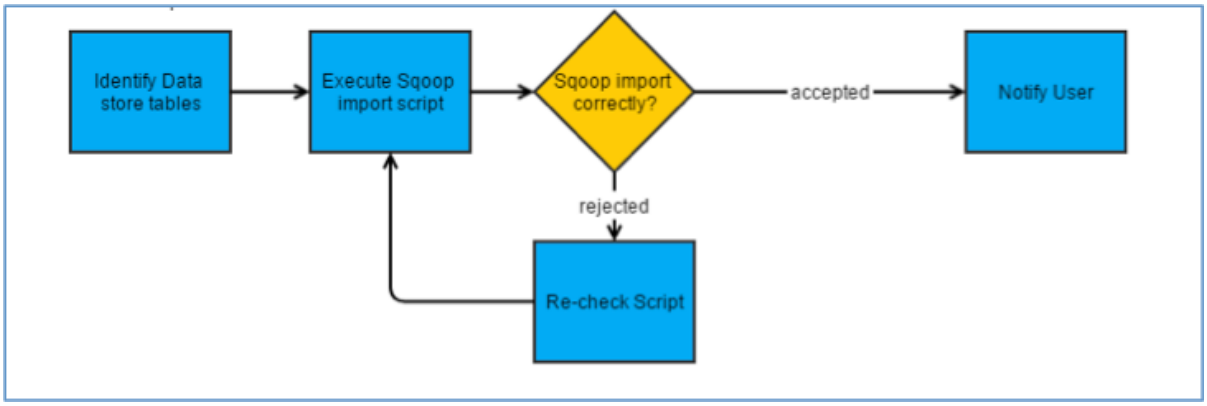
This use case, Figure 38, above, walks through the steps needed in order to install the basic toolsets for the Data Lake. The first step is to install the data ingest framework and file system so data can be ingested and store successfully. After that step is completed and verified, ETL tools will be installed to manage and transform data.

Figure 39. Data and Search Use Case Create Relational Database



This use case, Figure 39, above, walks through the steps needed in order to install a relational database, create the necessary database tables, and assign user permissions to the data stored in the table.

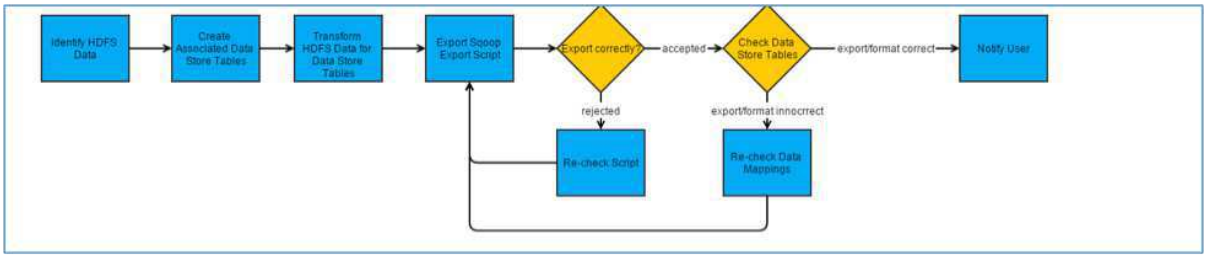
Figure 40. Data and Search Data Import from Data Stores to Data Lake



This use case, Figure 40, above, walks through the steps needed in order to transmit data from the relational and document data stores into the Data Lake.

5.4.2. DAT-4 Data Export from Data Lake to Data Stores

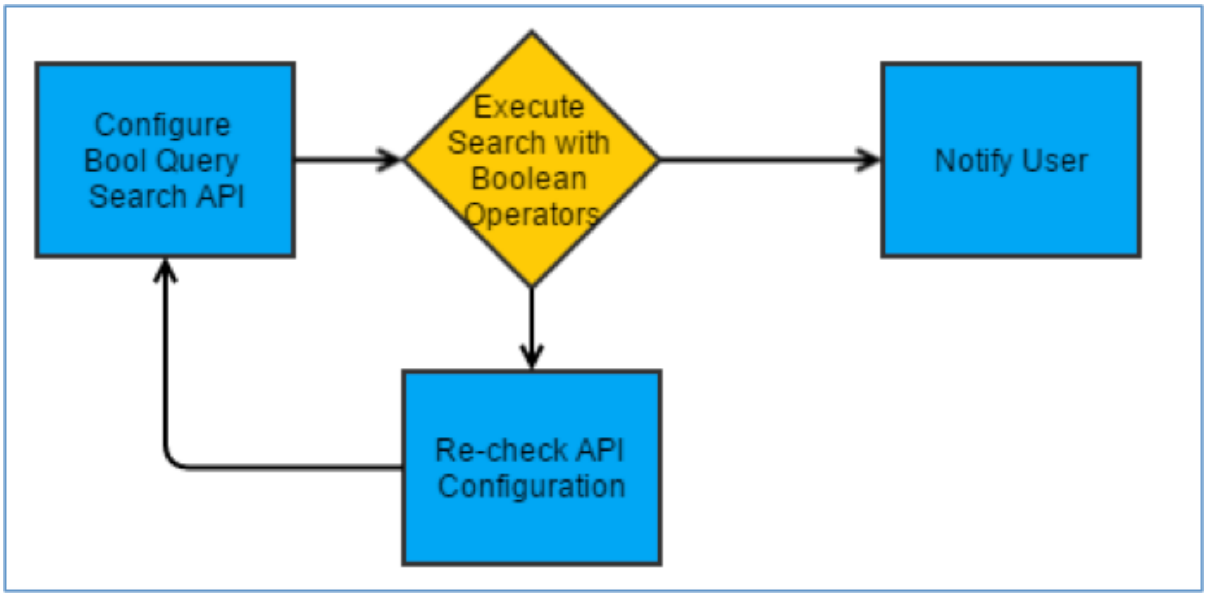
Figure 41. Data and Search Use Case Data Export from Data Lake to Data Stores



This use case, Figure 41, above, walks through the steps needed in order to transmit data from the Data Lake into the relational and document data stores. This use case differs from the ingress use case above, as data transformations are required to correctly map data from HDFS into the database tables.

5.4.3. DAT-5 Enable Boolean Search Capability

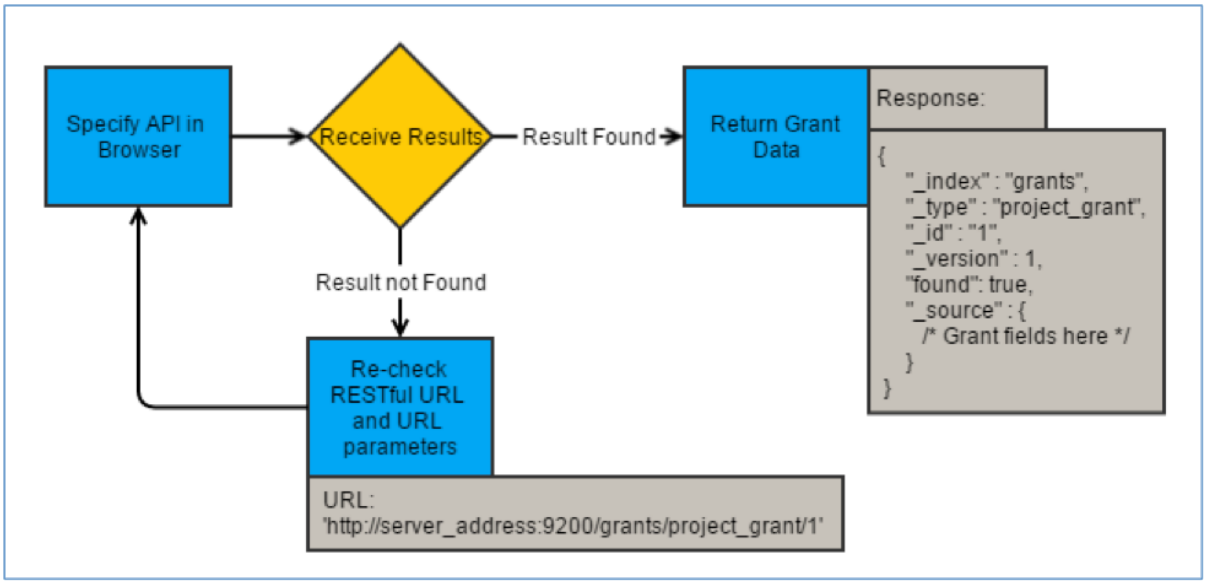
Figure 42. Data and Search Enable Boolean Search Capability



This use case, Figure 42, above, walks through the steps needed in order to configure Boolean search so users can specify Boolean operators in their search criteria.

5.4.4. DAT-6 Search Specific Index via Elastic Search API

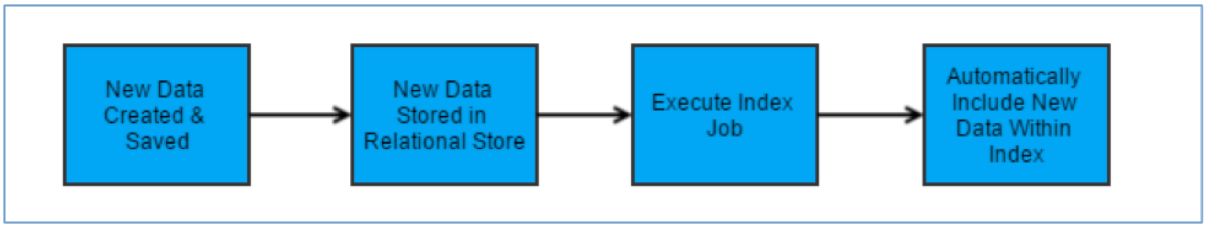
Figure 43. Data and Search Search Specific Index via Elastic Search API



This use case, Figure 43, above, walks through the steps needed in order to search for data via a specified index and ID number that corresponds to a particular data set. Through Elastic Search's RESTful API, the user is searching for a project grant within the grant index that is mapped to the ID '1'. Once that grant is found, it is returned in JSON format back to the user.

5.4.5. DAT-7 Elastic Search Auto Indexing

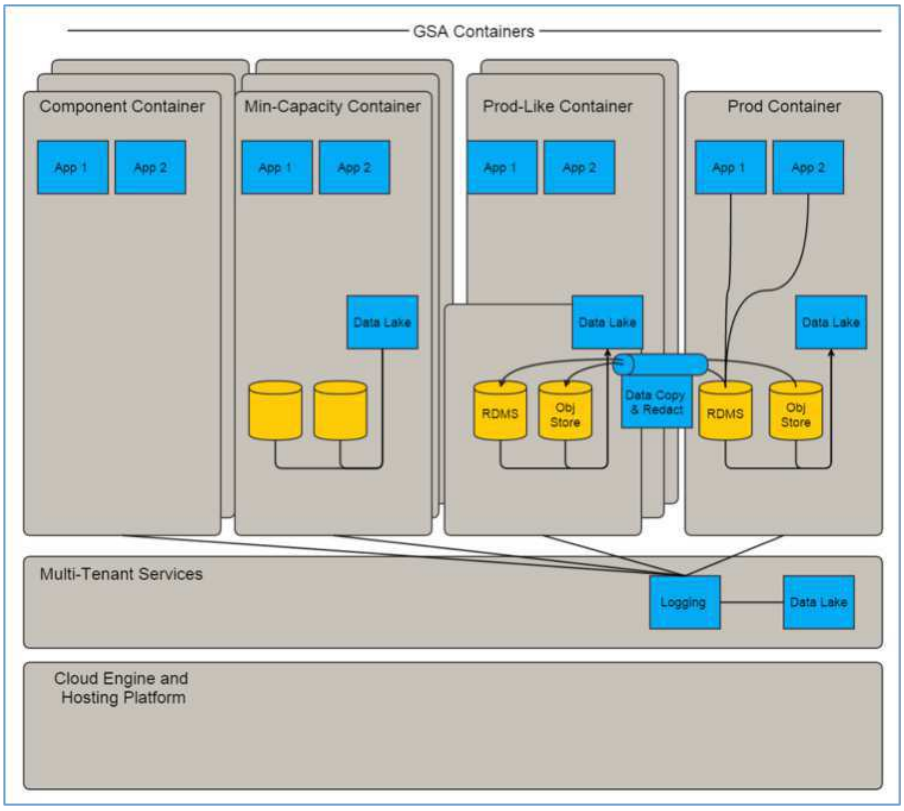
Figure 44. Data and Search Elastic Search Auto Indexing



This use case, Figure 44, above, walks through the steps that occur when auto indexing is enabled. Auto indexing can be configured, which enables the Elastic Search index to take a snapshot of the databases whenever new data is ingested and stored into the data stores. Using this snapshot, the new data set is then properly indexed for quick data retrieval.

5.4.6. DAT-8 Migration of Redacted data from Prod to Non-Prod Container

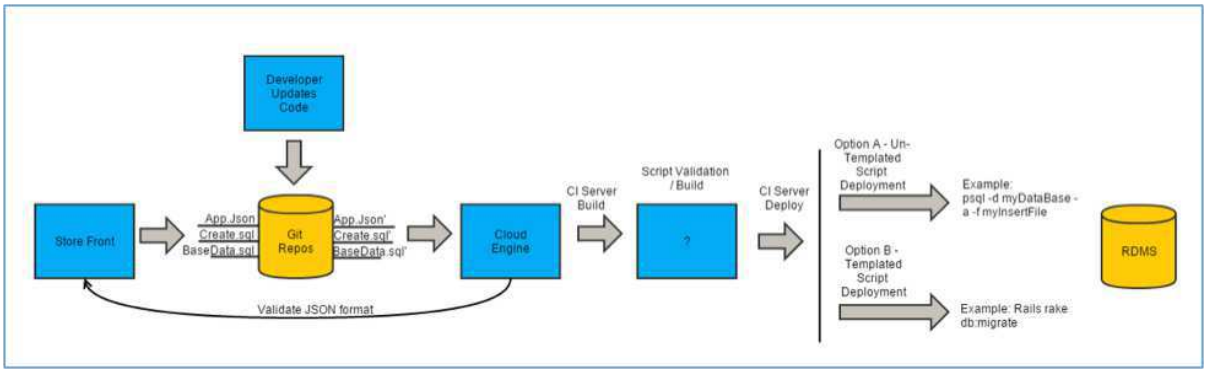
Figure 45. Data and Search Use Case Database Migration from Prod to Prod-Like



Applications in production container solely interact with the relational and document data stores. A process is established to allow data from production to be copied and redacted into the prod-like data stores. After that process successfully completes, the data within the prod-like container is re-ingested into the Data Lake. The details of the copy/redaction process are to be determined.

5.4.7. DAT-9 Relational Data Continuous Integration / Deployment 1

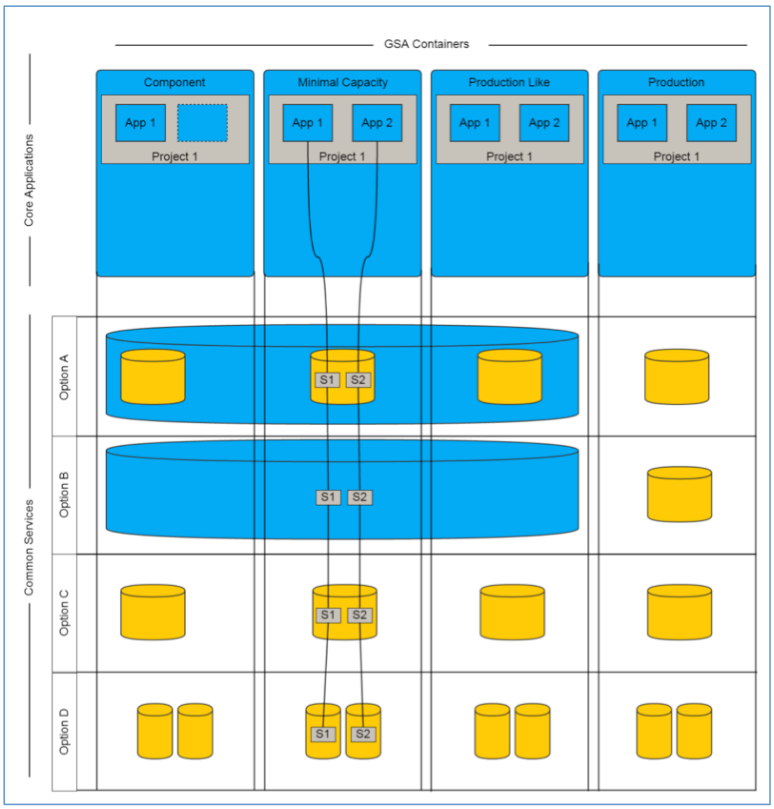
Figure 46. Data and Search Use Case Relational Data Continuous Integration Deployment 1



The Application Developer creates an application in the CSP Portal which provisions Git with the App.json specification and sample Create.sql / BaseData.sql files. The Application Developer then commits any updates to the Git repositories. Cloud Engine will then validate the App.json file and initiate the CI build process (E.g. Docker Image including application files). After this validation, a final validation check will occur to verify that the database scripts are properly formatted and correct. The CI deploy process will then deploy core application infrastructure and apply the database scripts to the relational data service. These scripts provide options for easily maintaining database schema migrations and updates.

5.4.8. DAT-10 Relational Data Continuous Integration / Deployment 2

Figure 47. Data and Search Use Case Relational Data Continuous Integration Deployment 2



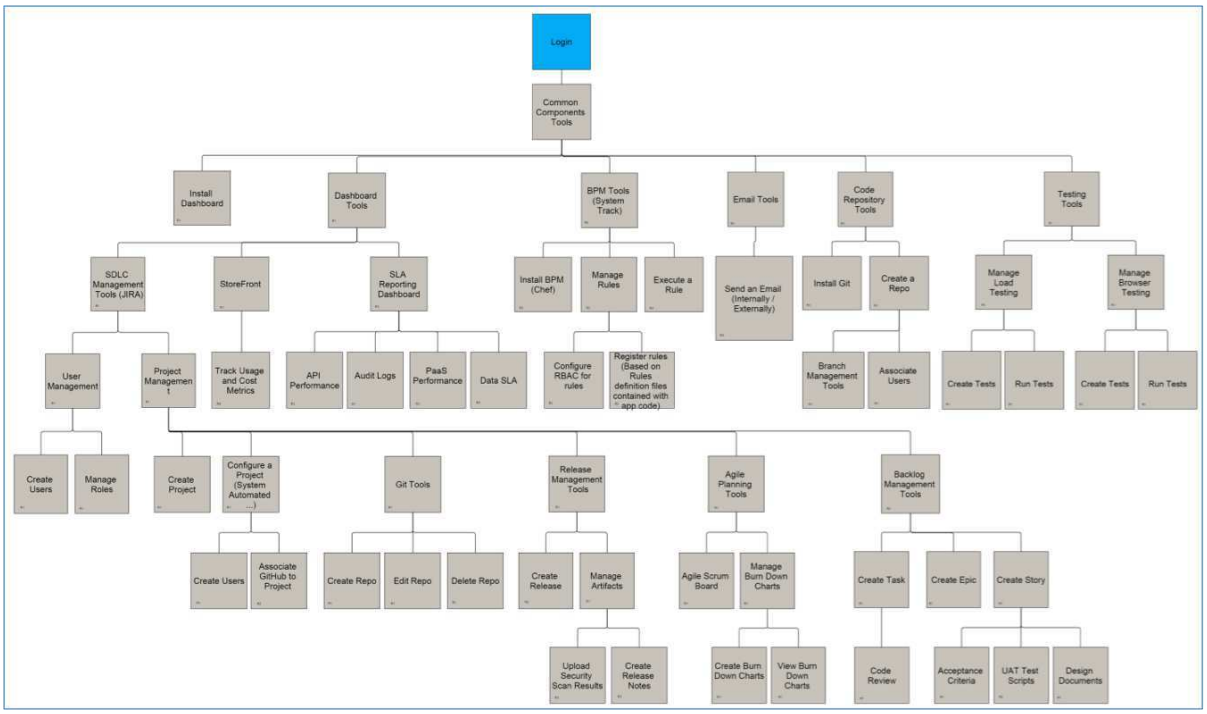
This use case, Figure 47, above, portrays various options for different Applications to manage their respective schemas. Options C (every schema is maintained through a single database) and D (every schema is mapped to a different database) are the more likely candidates as the data stores don't span multiple containers.

6. Common Components

Common Components will encompass several key areas. The CSP Portal will provide various dashboards that will be customized based on a user's roles and entitlements. One such dashboard may include project management tools such as JIRA that will be available to users with the corresponding roles and privileges. Other users may wish to see performance metrics and operational statuses in order to assist them with software/architecture optimization processes. Managing notifications will also be available in order to keep essential users informed of business process changes and system status updates. To ensure software stability and compliance with security policies, a continuous integration process will be in place using open-source tools. Lastly, software revision control systems such as Git will be used to support and maintain parallel development efforts.

6.1. Site Map

Figure 48. Common Components Sitemap

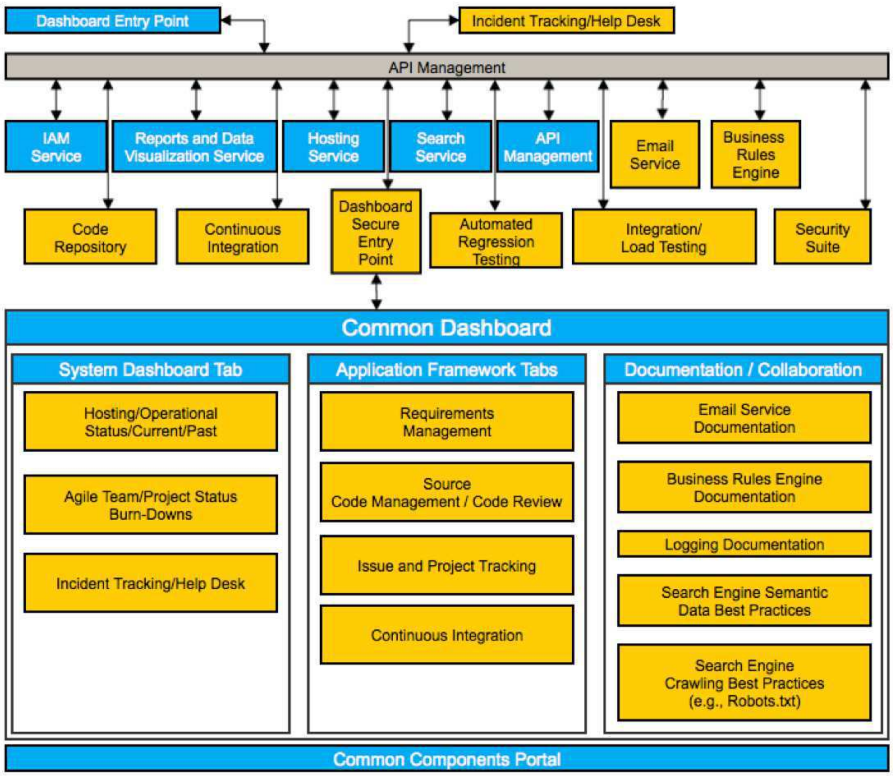


An overview of all the tools for the common components is shown above. After successful authentication, users may access the various tools. Tool availability will change based on a user's permissions and roles (which will be provided by the IAM). Some components that offer management capabilities will provide enhanced features to authorized users

6.2. Logical Architecture

6.2.1. Logical Architecture

Figure 49. Common Components Logical Architecture



Using a secure entry point, users will be able to access the Common Dashboard. From this dashboard, users will be able to retrieve documentation regarding the various components and have access to the services connected via the ESB. Multiple tabs with different capabilities will be available to the user. One example would be the ability to manage user stories using the JIRA Agile tool.

6.2.2. Marketplace Logical Architecture

Marketplace Features

The Marketplace offers the ability to access different functionality based on user role. For example, PaaS admins may have a different set of entitlements to functionalities within their dashboard vs. a project admin. Providing personalized views based on role creates a more streamlined and focused workflow for the end user. Ultimately, this helps to increase productivity.

Another key Marketplace capability is providing Git functionality to support future development efforts. Developers are able to customize and enhance the code by forking from the main Marketplace repository. Working in separate code branches enables parallel development which in turn leads to more frequent release cycles.

6.2.3. Assumptions and Constraints

Assumptions

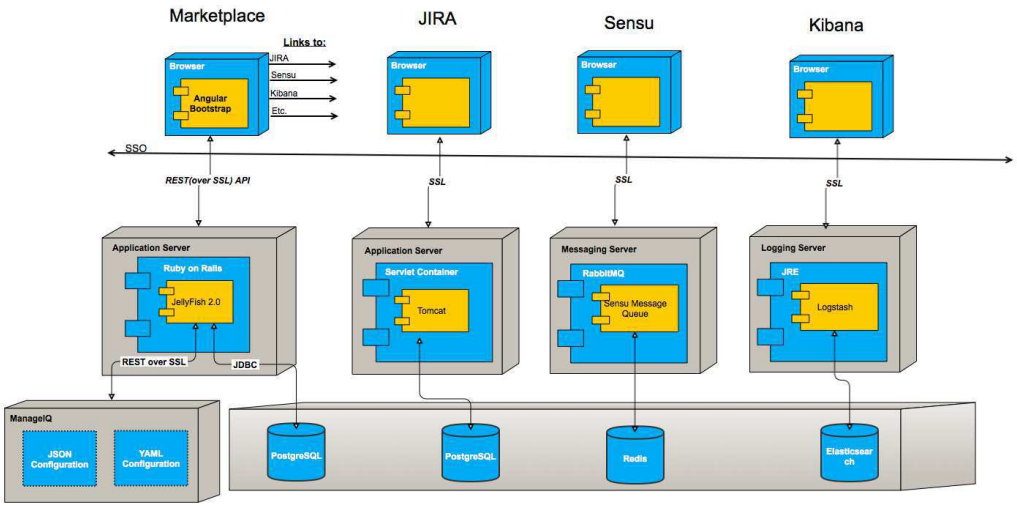
- CSP Marketplace UI will be a single-point-of entry for any UX-based system offered by CSP that will need to be consumed by Core App Developers or Users.
- JIRA will not be re-created in the CSP Portal.
- CSP Marketplace UI will be unique per user role.
- CSP Marketplace UI Dashboard will be supported in Firefox, Chrome, Internet Explorer and only GA and government version will be supported.

Constraints

- Existing applications may need to be redeveloped in order to comply with the new architecture
- System performance will vary depending on the hosting services provided

6.3. Physical Architecture

Figure 50. Common Components Physical Architecture



The application developers will use their browsers to access the marketplace and other applications available to support their development activities.

Different server tiers will manage the various capabilities. The Marketplace operates on top of the Ruby on Rails framework while JIRA operates on the Tomcat Java servlet container. PostgreSQL is used as the datastore. RabbitMQ is used by Sensu for monitoring along with Redis. The logging servers is provided by Logstash and made accessible to developers through an interface (using Kibana) to view and filter through the logging messages. Elasticsearch will be used as the datastore for logs. Lastly, CloudForms options can be customized by updating the JSON and YAML configurations.

6.4. Use Cases

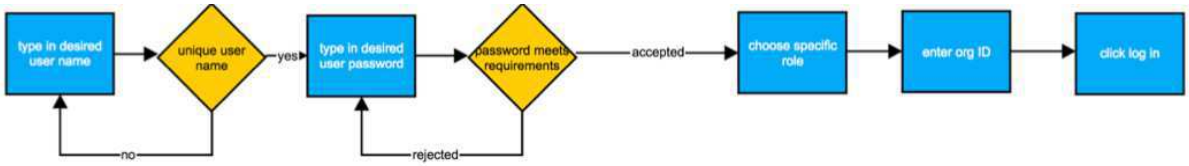
Table 11. Common Components Use Case Listing

ID	Use Case Name	Description	Actors	Pre-condition
CC-1	Create JIRA Account	Process to create a JIRA user account	Administrator	No JIRA account exists for user
CC-2	Log-in to CSP Portal	User authorization via the CSP Portal interface	User	User is not authorized
CC-3	Create/Manage Dashboard	Create a dashboard and set preferences	User	No user dashboard exist
CC-4	Gadget Customization	Add gadgets to dashboard	User	Gadget missing from user dashboard
CC-5	Project Metrics	View Sensu project metrics	User	Project metrics not available for user
CC-6	Search	Execute a search from within the CSP Portal	User	Empty query conditions
CC-7	Order Placement	Place an order from the CSP Portal	Core Application Developer/ Architect	Product/server is not provisioned
CC-8	Create CSP Portal	Steps to create a new CSP Portal	CSP Application Developer	Wireframes are designed
CC-9	User Account Management	Add user accounts and their related projects	PaaS Admin	User does not exist
CC-10	Project Status Report	Generate chart to show project status	Project Admin	A project has enough data to generate a burn down chart
CC-11	CSP Portal (User Portal)	Shows user path to the various CSP Portal tools	User	CSP Portal has at least one tool available
CC-12	Continuous Integration	Shows the role of continuous integration during development	Developer	Project is ready to send through the continuous integration process
CC-13	Load Testing	Process to run load tests	Core Application Developer/ Architect	Application is not load tested

ID	Use Case Name	Description	Actors	Pre-condition
CC-14	Browser Testing	Process to run browser tests	Core Application Developer/ Architect	Application is not browser tested
CC-15	Kibana UI Search	Perform search through the Kibana UI	Developer,Architect,System Administrator	Systemquery results
CC-16	Configure Kibana	Setup Kibana on server	System Administrator	Kibana source files are available but not installed
CC-17	Configure Logstash	Setup Logstash on server	System Administrator	Logstash source files are available but not installed
CC-18	Configure Logstash Forwarder	Setup Logstash Forwarder on server	System Administrator	Logstash Forwarder source files are available but not installed

6.4.1. CC-1: Create JIRA Account

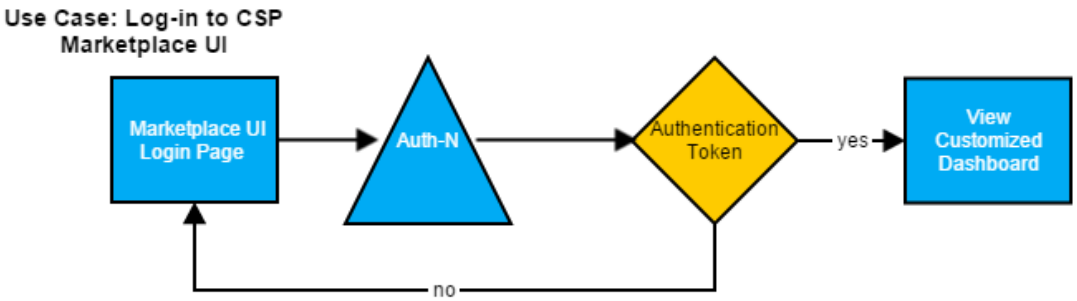
Figure 51. Common Components Use Case Create JIRA Account



User will enter in the desired user name. If the user name already exists, a different user name will have to be provided. Once a user name is created, a password (that meets the system's security criteria) will be created. Afterwards, roles and the user's organizational ID will be specified. Now, the user may log in.

6.4.2. CC-2: Log-in to CSP Marketplace UI

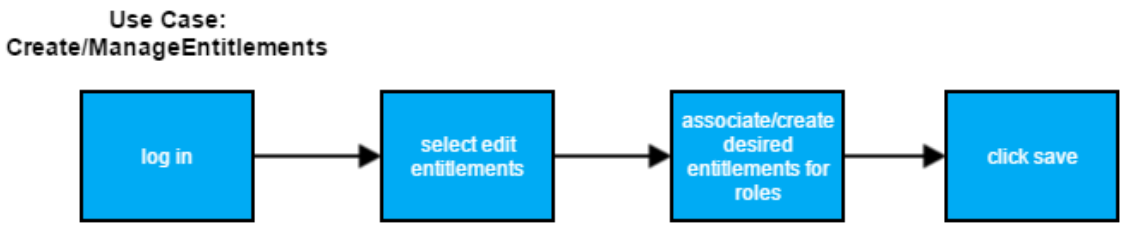
Figure 52. Common Components Use Case Login to CSP Marketplace UI



A user will enter through the CSP Marketplace UI login page and proceed through authentication. If authenticated, a user will receive an authentication token and be forwarded to their customized dashboard. If authentication fails, the user will be redirected to CSP Marketplace UI login page informing the user of the failed authentication.

6.4.3. CC-3: Create/Manage Entitlements

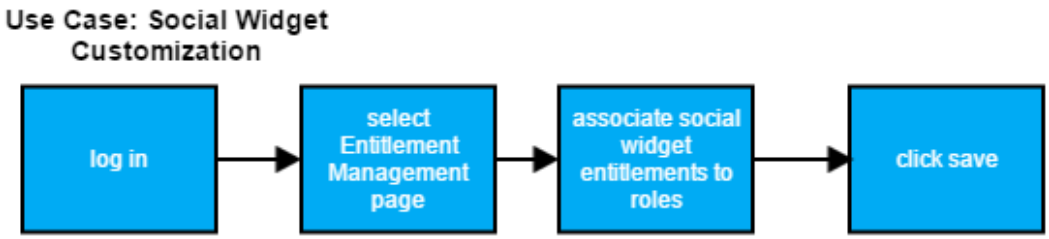
Figure 53. Common Components Use Case Create Manage Entitlement



After successful login, a privileged user will select the Entitlement Management option. From there, the user will edit entitlements based on roles. The user's entitlements can be stored permanently by clicking the Save button.

6.4.4. CC-4: Social Widget Customizations

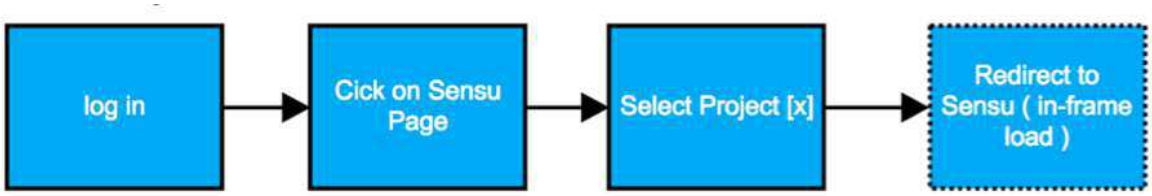
Figure 54. Common Components Use Case Widget Customization



A privileged user will login and select Entitlement Management page. From here, user will associate desired social widget entitlements (Facebook, Twitter…) with roles. Once selected, the social widgets will be added to the dashboard next time user logs in. Once finished, clicking Save will permanently store the changes.

6.4.5. CC-5: Project Metrics

Figure 55. Common Components Use Case Project Metrics



User will login and click on the Sensu page. Afterwards, a project can be selected which will redirect the user to the page with the Sensu metrics data.

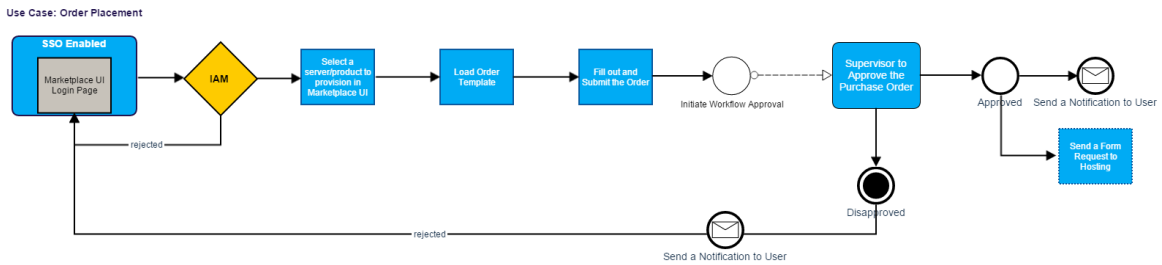
6.4.6. CC-6: Search

Figure 56. Common Components Use Case Search

User will type in a query into the search bar. If any additional search criteria is required, the user may select those options. From there, clicking the search button will perform the query and the results will be displayed to the user.

6.4.7. CC-7: Order Placement

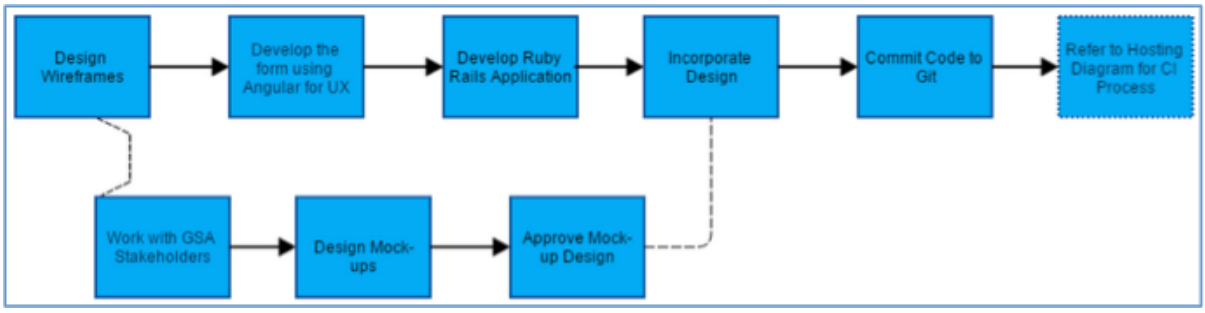
Figure 57. Common Components Use Case Order Placement



Developer/architect authenticates and proceeds to the marketplace. If user is not authenticated, then the user is redirected back to the CSP Marketplace UI login page. Once inside of the Marketplace UI, user selects A product/server to provision. Once the order template is filled out, it is submitted. The order goes through an approval process. If approved, the request is forwarded to Hosting and the user is notified. If not, then the user is redirected back to the CSP Marketplace UI. A notification of the failure is sent to the user.

6.4.8. CC-8: Create CSP Portal

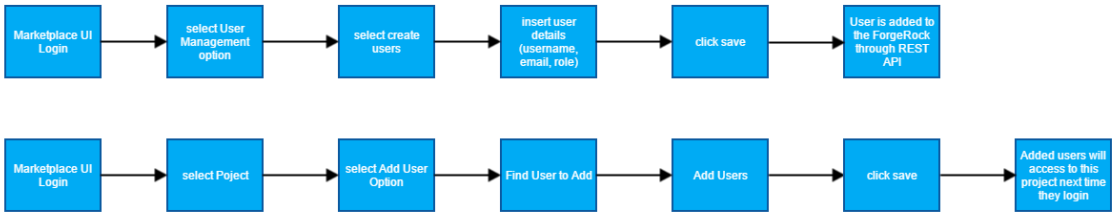
Figure 58. Common Components Use Case Create CSP Portal



Using the available wireframes, the developer creates the form components using Angular. The Ruby on Rails framework is coded to provide the necessary endpoints. Design elements are worked out with GSA stakeholders and the design is incorporated into the application. All of the CSP Portal code is committed to a Git repository. From there, the code is now included as part of the continuous integration process.

6.4.9. CC-9: User Account Management

Figure 59. Common Components Use Case User Account Management



Once logged in, the admin selects the User Management option. The create users option is now selected. User details are inserted. The admin clicks save to complete the process. The user will be saved to the ForgeRock database through REST API.

After logged in, the admin selects the project to associate users with. Click on Add User option. Find the users in the User Management page. Add users to the project. Click Save to complete the process. The added users will see project next time they log into the Marketplace.

6.4.10. CC-10: Project Status Report

Figure 60. Common Components Use Case Project Status Report 1

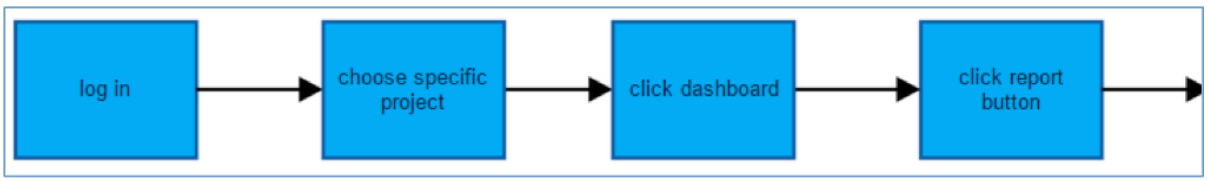
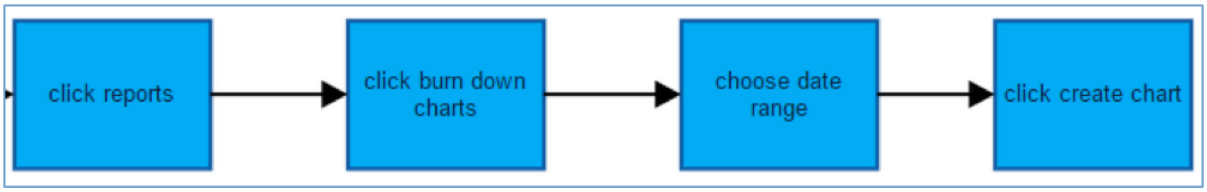


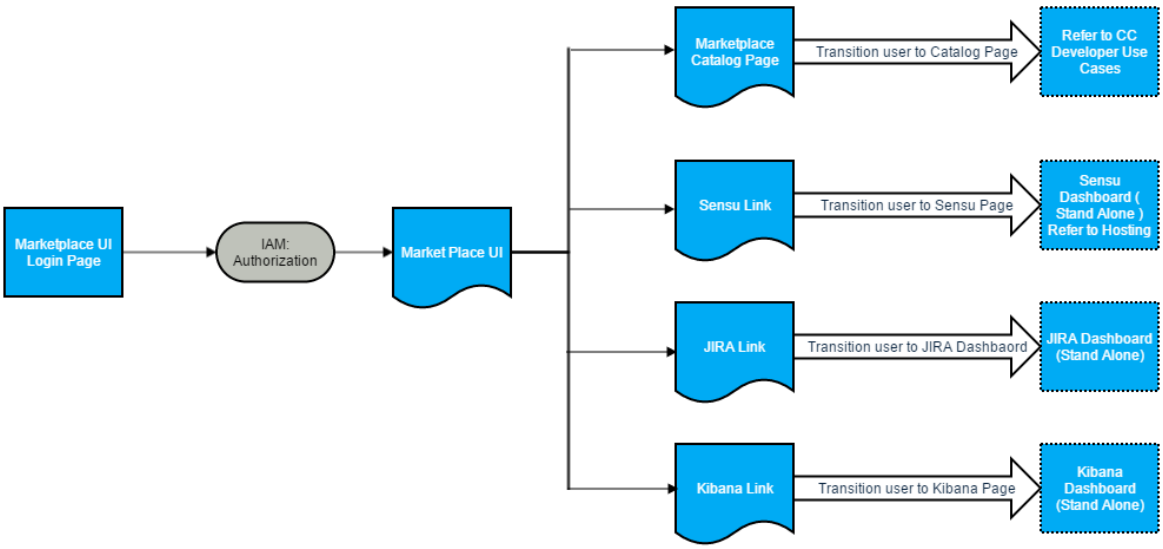
Figure 61. Common Components Use Case Project Status Report 2



Project admin logs in and selects a specific project. Then, the admin proceeds to the dashboard. The report button is clicked and then the burn down chart option is chosen. A date range is specified and then the create chart button is clicked which will generate the chart.

6.4.11. CC-11: CSP Marketplace UI (User Dashboard)

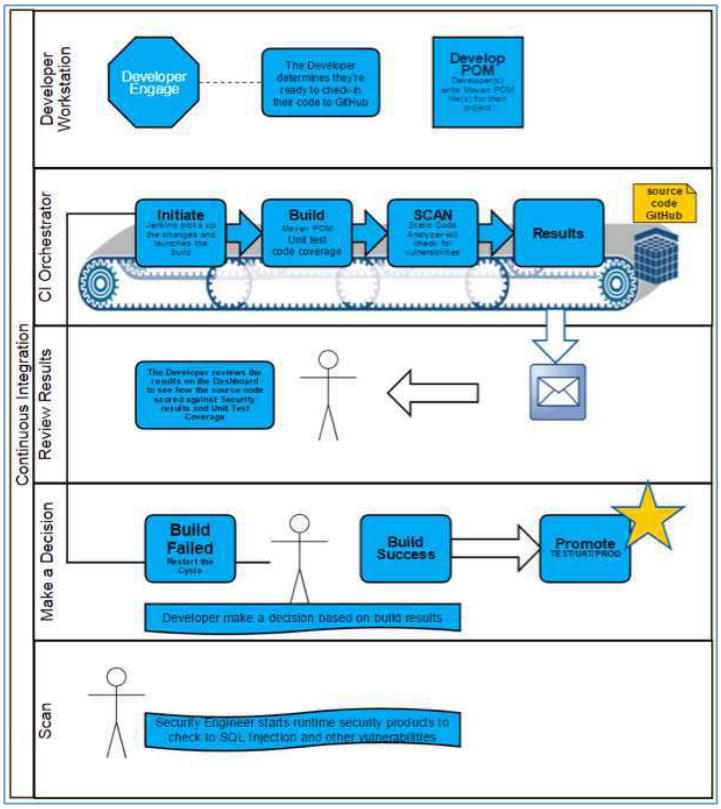
Figure 62. Common Components Use Case CSP Marketplace UI User Dashboard



Multiple tools are available from the CSP Marketplace UI view. Before any further page can be displayed, the IAM will verify each user for their roles and permissions. From the Marketplace Page, a user is transitioned to the Catalog Page. The Sensu link transitions a user to the Sensu Dashboard. The JIRA link transitions a user to the JIRA Dashboard. Finally, the Kibana link transitions a user to the Kibana Dashboard.

6.4.12. CC-12: Continuous Integration

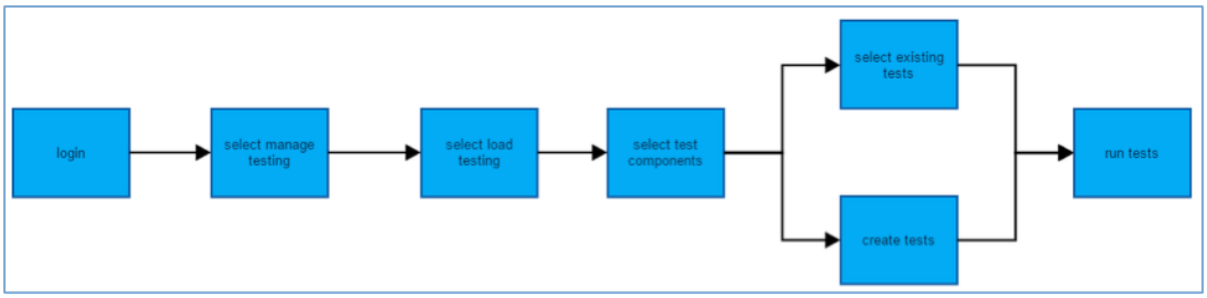
Figure 63. Common Components Use Case Continuous Integration



The continuous integration process encourages developers to remain actively engaged with the software build process. Project code, along with the associated POM files, is committed to the Git repository. Jenkins picks up the changes and launches the build process. Unit tests are run and the code is checked for vulnerabilities. The results are collected and notifications are sent out to the developer. Developer reviews over the results and determines whether the build was successful or not. If the build failed, then the developer will repeat the CI process. If the build was successful, further testing is performed. Once UAT and other testing are completed, the code is pushed to the production environment. Once in the production environment, the code is continually tested against existing security policies.

6.4.13. CC-13: Load Testing

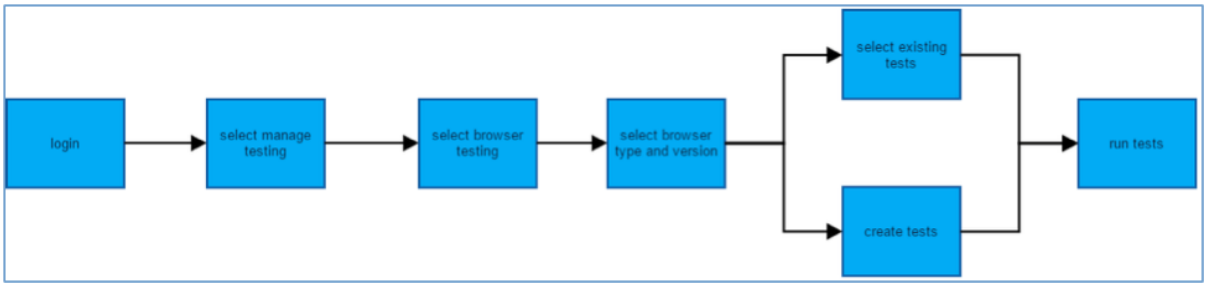
Figure 64. Common Components Use Case Load Testing



User logs in and selects manage testing. From there, the load testing option is selected. The components to test are chosen. Afterwards, the tests are to be run against those components are selected. At this point, the user can specify any tests that do not exist at the time. Once all the tests are defined, the user proceeds to run them.

6.4.14. CC-14: Browser Testing

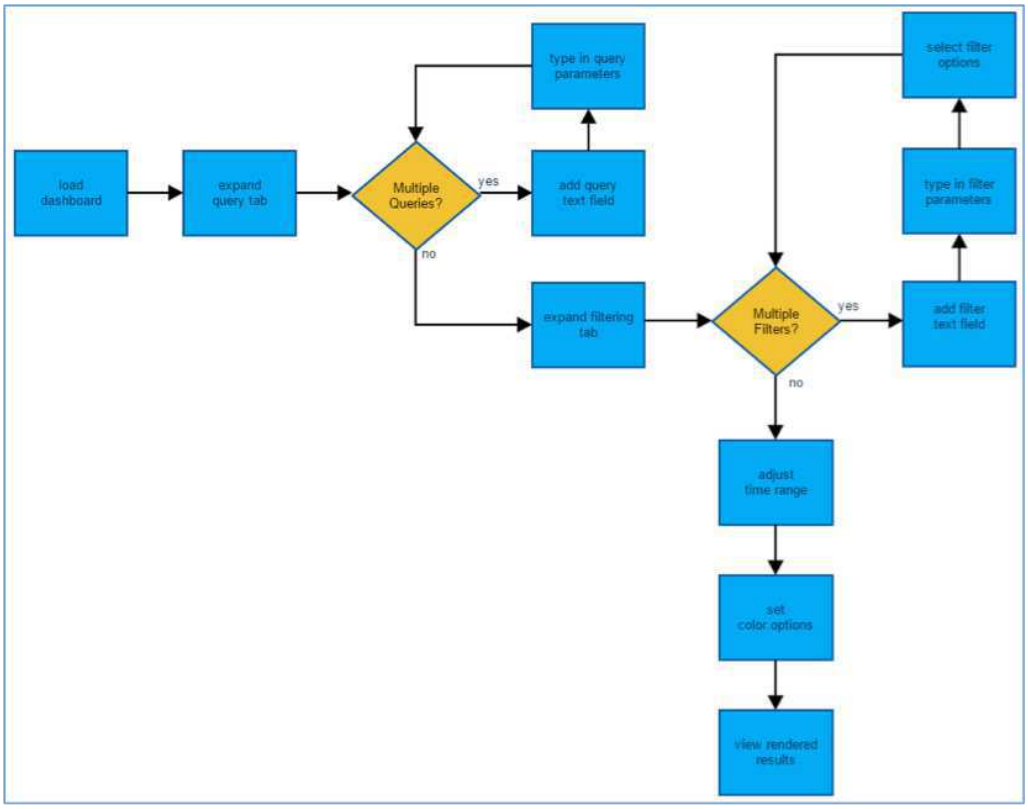
Figure 65. Common Components Use Case Browser Testing



User logs in and selects manage testing. From here, the user selects browser testing. A browser type and version must be specified for the tests to run against. Afterwards, the user can use existing test cases or create new ones. Finally, the tests are run against the defined browser.

6.4.15. CC-15: Kibana UI Search

Figure 66. Common Components Use Case Kibana UI Search

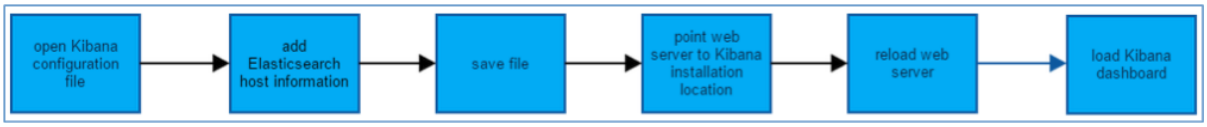


Once the Kibana dashboard is loaded, the user expands the query tab. If multiple queries exist, the user can add a new text field. After typing in all the query parameters for each text field, the filtering

tab is expanded. If multiple filters, the user can append additional filter text fields. Next, if any filtering options are desired, the user should select them. The time range for the search can be adjusted along with the report color options. The display will update to reflect any options/parameters specified by the user.

6.4.16. CC-16: Configure Kibana

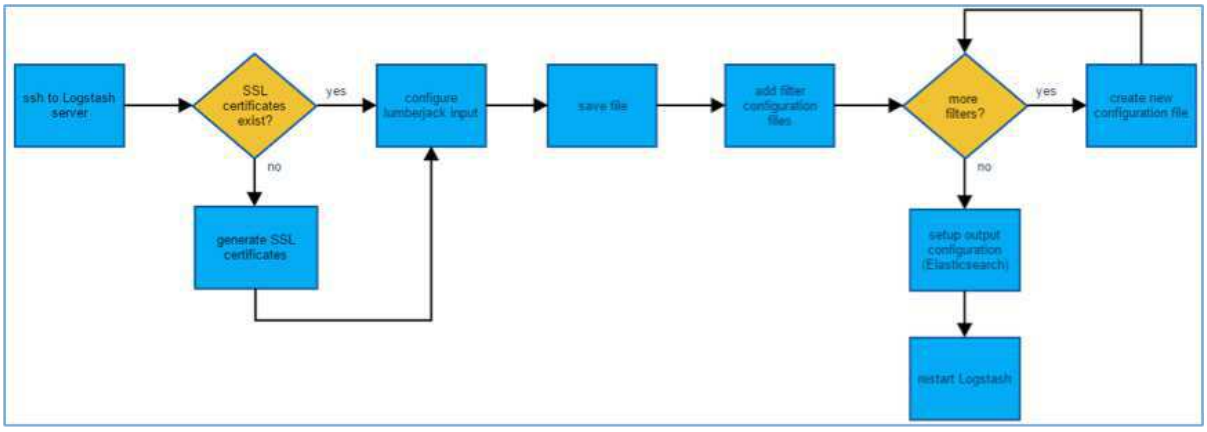
Figure 67. Common Components Use Case Configure Kibana



After opening the Kibana configuration file, add the datastore (Elasticsearch) host information. Save the file and point the web server to the Kibana install location. Reload the web server. The Kibana dashboard is now installed and ready for use.

6.4.17. CC-17: Configure Logstash

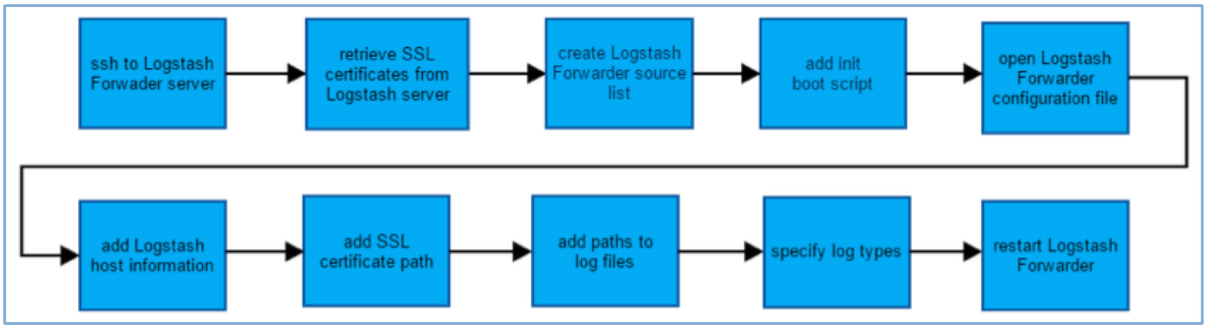
Figure 68. Common Components Use Case Configure Logstash



Using a terminal window, SSH to the Logstash server. If SSL certificates do not exist, generate them. If SSL certificates exist, proceed to configure the lumberjack input. Save the file and add filter configuration files. If more than one filter exists, create a new configuration file for each new filter. Specify the output (Elasticsearch) configuration and restart the Logstash process. Logstash is now ready for use.

6.4.18. CC-18: Configure Logstash Forwarder

Figure 69. Common Components Use Case Configure Logstash Forwarder



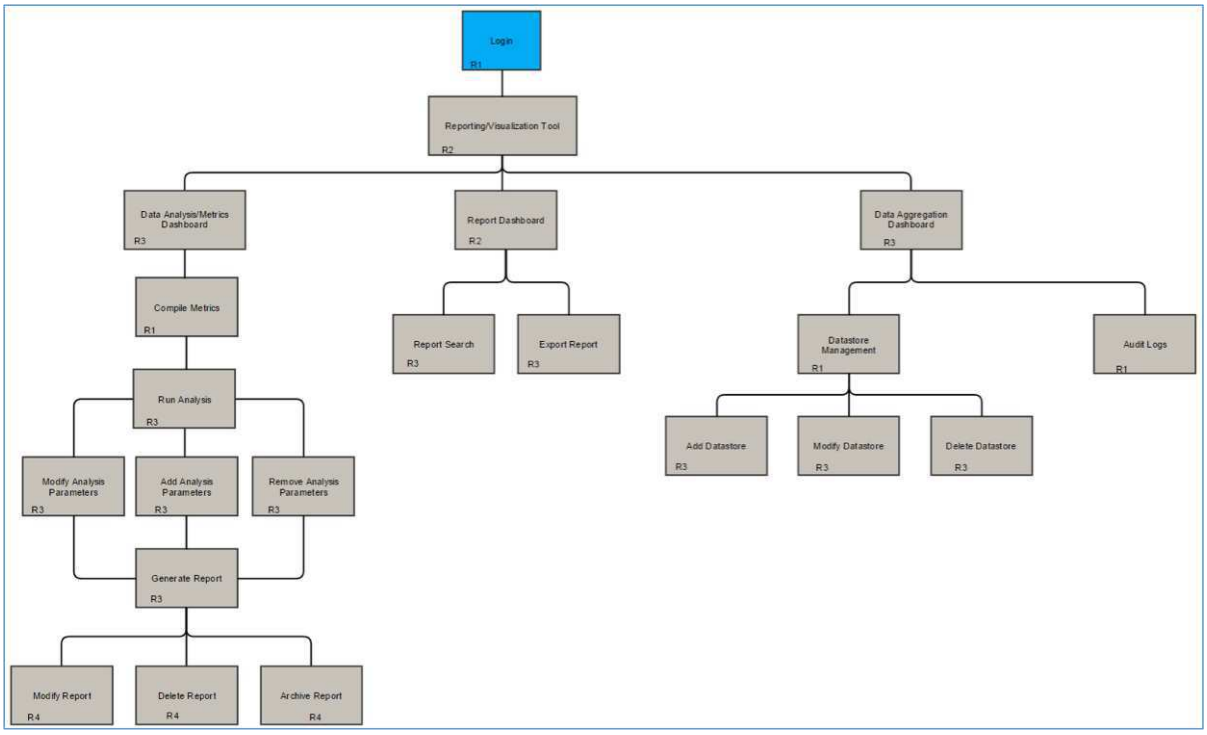
SSH to the Logstash Forwarder server. From the Logstash server, copy over the SSL certificate. Create the Logstash Forwarder source list and add the boot initialization script. Open the Logstash configuration file. Add the Logstash host information, SSL certificate path, path of log files and specify the log types. Restart the Logstash Forwarder process. Logstash Forwarder is now ready for use.

7. Reporting and Visualization

The Reports and Visualization component will provide access to aggregated metrics and their relevance to a project/system. Reports may include performance analysis and project statuses. The unstructured data can be filtered and sorted to produce meaningful results. These results can then be translated to increase productivity and streamline workflows.

7.1. Site Map

Figure 70. Reporting and Visualization Sitemap

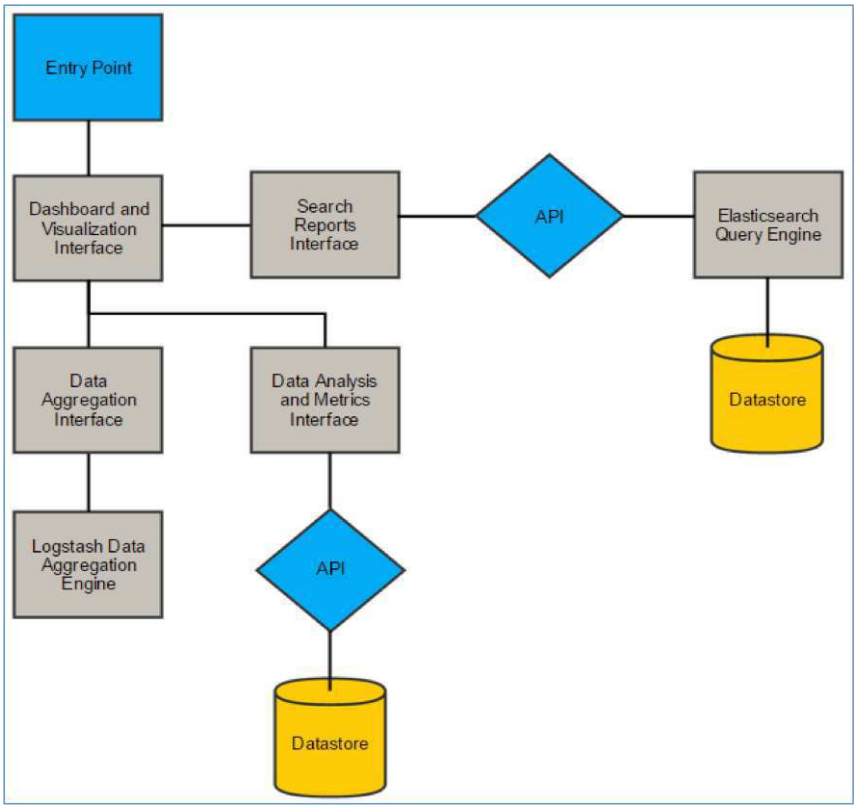


An overview of the reporting and visualization tool is shown in Figure 70, above. After successful authentication, users may access the various features of the tool. Availability of specific features will vary based on a user's permissions and roles (which will be provided by the IAM). Some components that offer management capabilities will provide enhanced features to authorized users.

7.2. Logical Architecture

7.2.1. Logical Architecture

Figure 71. Reporting and Visualization Logical Architecture



The Reports and Visualization component will be secured behind the entry point. Once users are authenticated, they will be able to access the dashboard and visualization interface. From here, multiple interfaces will be available such as reporting and data analysis/metrics. Logstash will be used to assist in aggregating the data. Reports can then be generated by processing the data through an Elasticsearch engine.

7.2.2. Assumptions and Constraints

Assumptions

- CSP Portal will be a single-point-of entry for any User Experience (UX)-based system offered by CSP that will need to be consumed by Core App Developers or Users. As a result, users will be authenticated and authorized at point of entry to CSP Portal.
- Tableau Desktop licensing will be provided through the CSP contract similar to other tools such as JIRA. Tableau Desktop usage will be limited, as most users will leverage the Tableau Server for presentation.
- CSP Portal Dashboard will only allow access to the Reports and Visualization tool if the user has the appropriate roles and permissions

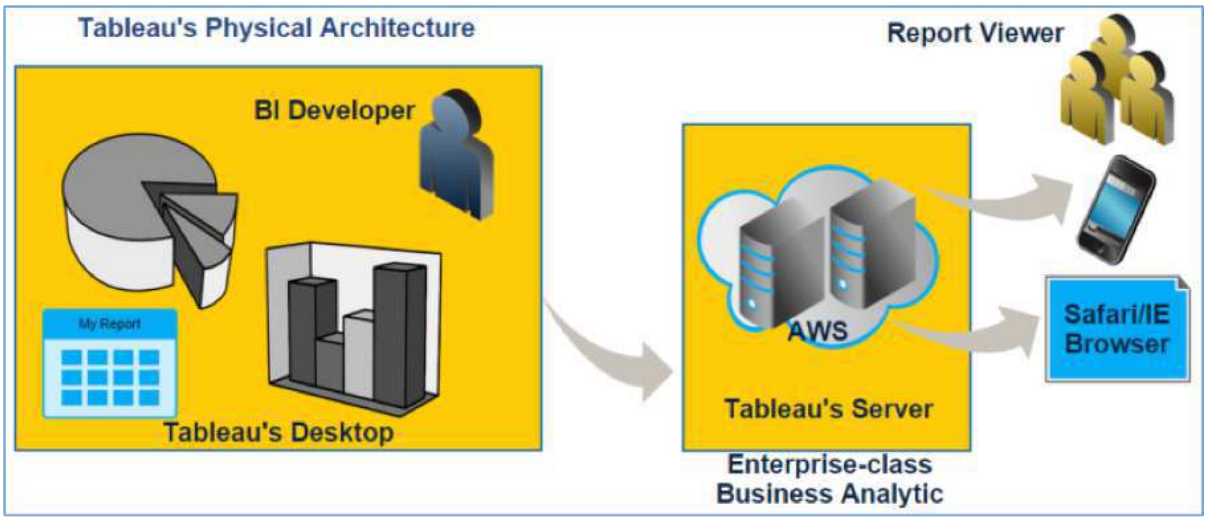
Constraints

- Existing applications may need to be updated to provide the necessary data points for the new reporting interfaces
- System performance will vary depending on the hosting services provided

7.3. Physical Architecture

7.3.1. Physical Architecture

Figure 72. Reporting and Visualization Physical Architecture



BI developers will use Tableau to create the reporting tools. Tableau Desktop can be installed locally on end user/developer machines. Once installed, Tableau Desktop will connect with the enterprise-class Tableau Server. AWS will be used as the basis for the server infrastructure. The server will perform business analytics and the results will then be viewable to the client. The reports may be viewed on either desktop or mobile platforms.

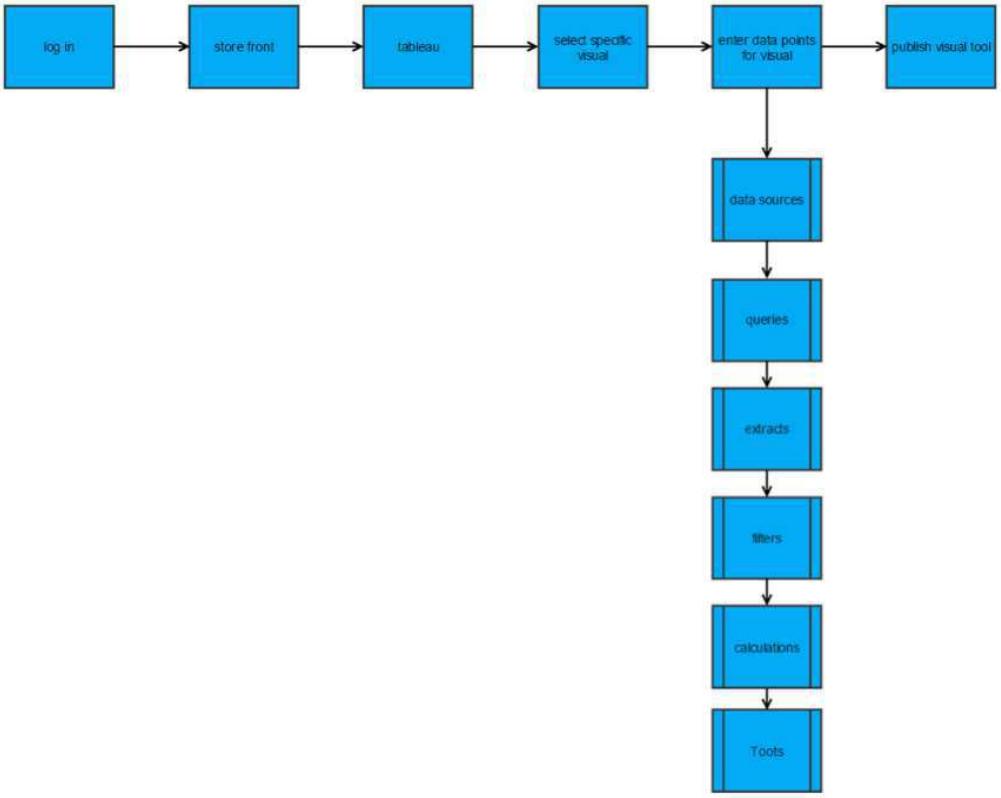
7.4. Use Cases

Table 12. Reporting and Visualization Use Case Listing

ID	Use Case Name	Description	Actors	Pre-condition	Post-condition
RV-1	Developer	Create a new visualization	Developer	Desired visualization does not exist	New visualization added
RV-2	User	View an existing visualization	User	At least one visualization is available	Visualization is viewable or downloadable

7.4.1. RV-1 Reporting and Visualization Developer Use Case

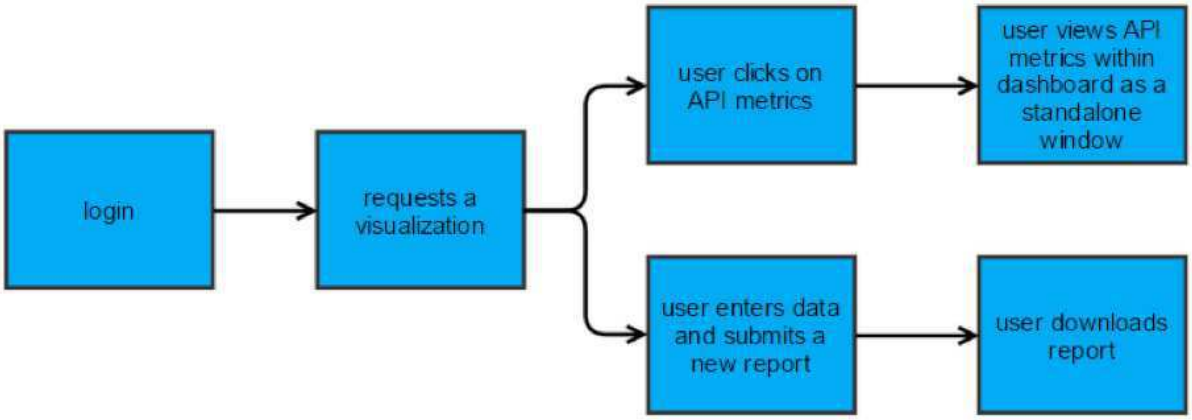
Figure 73. Reporting and Visualization Use Case Developer Use of Reporting and Visualization



Developer will log in and proceed to the CSP Portal. Afterwards, the Tableau application will be opened. A specific visual will be selected and data points for that visual will be defined. Additional criteria and conditions can be added to the visual. These include data sources, queries, extracts, filters and calculations. Once all the input parameters have been accounted for, the visualization tool is created.

7.4.2. RV-2 Reporting and Visualization User Use Case

Figure 74. Reporting and Visualization Use Case User Use of Reporting and Visualization



8. Appendices

8.1. Glossary of Acronyms

Table 13. Glossary of Acronyms Listing

Term	Definition
AAA	Authentication, Authorization, and Auditing
AD	Active Directory
API	Application Programming Interface
AWS	Amazon Web Services
BI	Business Intelligence
BPM	Business Process Management
CAC	Common Access Card
CB	Cloud Bees
CC	Common Components
CE	Cloud Engine
CI	Continuous Integration
CLI	Command Line Interface
CMDB	Configuration Management Database
COTS	Commercial Off The Shelf
CPU	Central Processing Unit
CRUD	Create, Read, Update, Delete
CSP	Common Services Platform
CSP	Cloud Services Provider
CSS	Cascading Style Sheets
CSV	Comma Separated Values
CXF	Celtix and XFire
DB	Database
DBA	Database Administrator
DDOS	Distributed Denial of Service Attack
DL	Data Lake
DMZ	Demilitarized Zone
DNS	Domain Name Service
EAP	Enterprise Application Platform
EBS	Elastic Block Store
EC2	Elastic Compute Cloud

Term	Definition
EIP	Enterprise Integration Patterns
ELB	Elastic Load Balancer
ESB	Enterprise Service Bus
eSRS	Electronic Subcontracting Reporting System
ETL	Extract, Transform, Load
GSA	General Services Administration
HA	High Availability
HDFS	Hadoop File System
HDP	Hadoop Data Platform
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IAE	Integrated Award Environment
IAM	Identity and Access Management
IDM	Identity Management
IDP	Identity Provider
IDS	Intrusion Detection System
IE	Internet Explorer
IPS	Intrusion Prevention System
J2E	Java Enterprise Edition
JAAS	Java Authentication and Authorization Service
JBPM	Java Business Process Management
JCR	Java Content Repository
JDBC	Java Database Connectivity
JMS	Java Message Service
JS	JavaScript
JSON	JavaScript Object Notation
LDAP	Lightweight Directory Access Protocol
MBI	Minimal Background Investigation
MC	Minimum Capacity
MD5	Message Digest 5
NIDS	Network Intrusion Detection System
NIPS	Network Intrusion Prevention System
NIST	National Institute of Standards and Technology
NVDA	Non-Visual Desktop Access
OLAP	Online Analytical Processing

Term	Definition
OS	Open Source
OS	Operating System
OSGI	Open Service Gateway Initiative
OTP	One Time Password Token
PaaS	Platform as a Service
PDF	Portable Document Format
PIM	Personal Information Manager
PIV	Personal Identity Verification
POC	Point of Contact
POM	Project Object Model
RAML	RESTful API Modeling Language
RBAC	Role Base Access Control
RDBMS	Relational Database Management System
RDMS	Relational Database Management System
REST	Representational State Transfer
RHEL	Red Hat Enterprise Linux
S3	Simple Storage Service
SaaS	Software as a Service
SAM	Security Access Manager
SAML	Security Assertion Markup Language
SCAP	Security Content Automation Protocol
SCIM	System for Cross-domain Identity Management
SDK	Software Development Kit
SDLC	Software Development Life Cycle
SES	Simple Email Service
SG	Security Group
SLA	Service Level Agreement
SLO	Single Logout
SMTP	Simple Mail Transfer Protocol
SQL	Structured Query Language
SSH	Secure Shell
SSL	Secure Socket Layers
SSO	Single Sign On
TCO	Total Cost of Ownership
TCP/IP	Transfer Control Protocol/Internet Protocol

Term	Definition
TLS	Transport Layer Security
TTL	Time to Live
UAT	User Acceptance Testing
UI	User Interface
URL	Uniform Resource Locator
UTM	Unified Threat Management
UX	User Experience
VM	Virtual Machine
VPC	Virtual Private Cloud
VPN	Virtual Private Network
WSDL	Web Services Description Language
XML	Extensible Markup Language
YAML	YAML Ain't Markup Language

8.2. Term Catalog

Table 14. Term Catalog Listing

Layer	Term	Definition
API Management	Migrated Application	Rewritten applications on the PaaS
API Management	Legacy Application	Forklifted application residing on the platform but not integrated
Common Components	REST API	Defines a uniform application interface that abstracts the implementation details.
Common Components	Continuous Integration	Process to maintain software by continually running tests against the code and other policies (such as security) that may exist.
Common Components	Legacy Applications	Software that was developed out of CSP scope.
Common Components	Jellyfish	Booz Allen Open Source Cloud Broker Marketplace
IAM	Organization	An Organization is Considered a Development Team
IAM	Application	An Application is a single deployable unit with one or more dependencies

Layer	Term	Definition
Reports and Visualization	Data Analysis/Analytics	Involves processing and modeling data to produce results that will support making business decisions
Reports and Visualization	Visualization	A report that is generated from a combination of various data points, filters, extracts, calculations, and queries