



# Smart Data Platform™

Technical Architecture



## Smart App Builder™

# Objectives



ChainSys' Smart Data Platform enables the business to achieve these critical needs.

1. Empower the organization to be data-driven
2. All your data management problems solved
3. World class innovation at an accessible price



## Subash Chandar Elango

Chief Product Officer  
ChainSys Corporation

Subash's expertise in the data management sphere is unparalleled. As the creative & technical brain behind ChainSys' products, no problem is too big for Subash, and he has been part of hundreds of data projects worldwide.

# Introduction

This document describes the Technical Architecture of the Chainsys Platform



## Purpose

The purpose of this Technical Architecture is to define the technologies, products, and techniques necessary to develop and support the system and to ensure that the system components are compatible and comply with the enterprise-wide standards and direction defined by the Agency.



## Scope

The document's scope is to identify and explain the advantages and risks inherent in this Technical Architecture.

This document is not intended to address the installation and configuration details of the actual implementation. Installation and configuration details are provided in technology guides produced during the project.



## Audience

The intended audience for this document is Project Stakeholders, technical architects, and deployment architects























**Smart Data Platform™**

# Architecture Goals

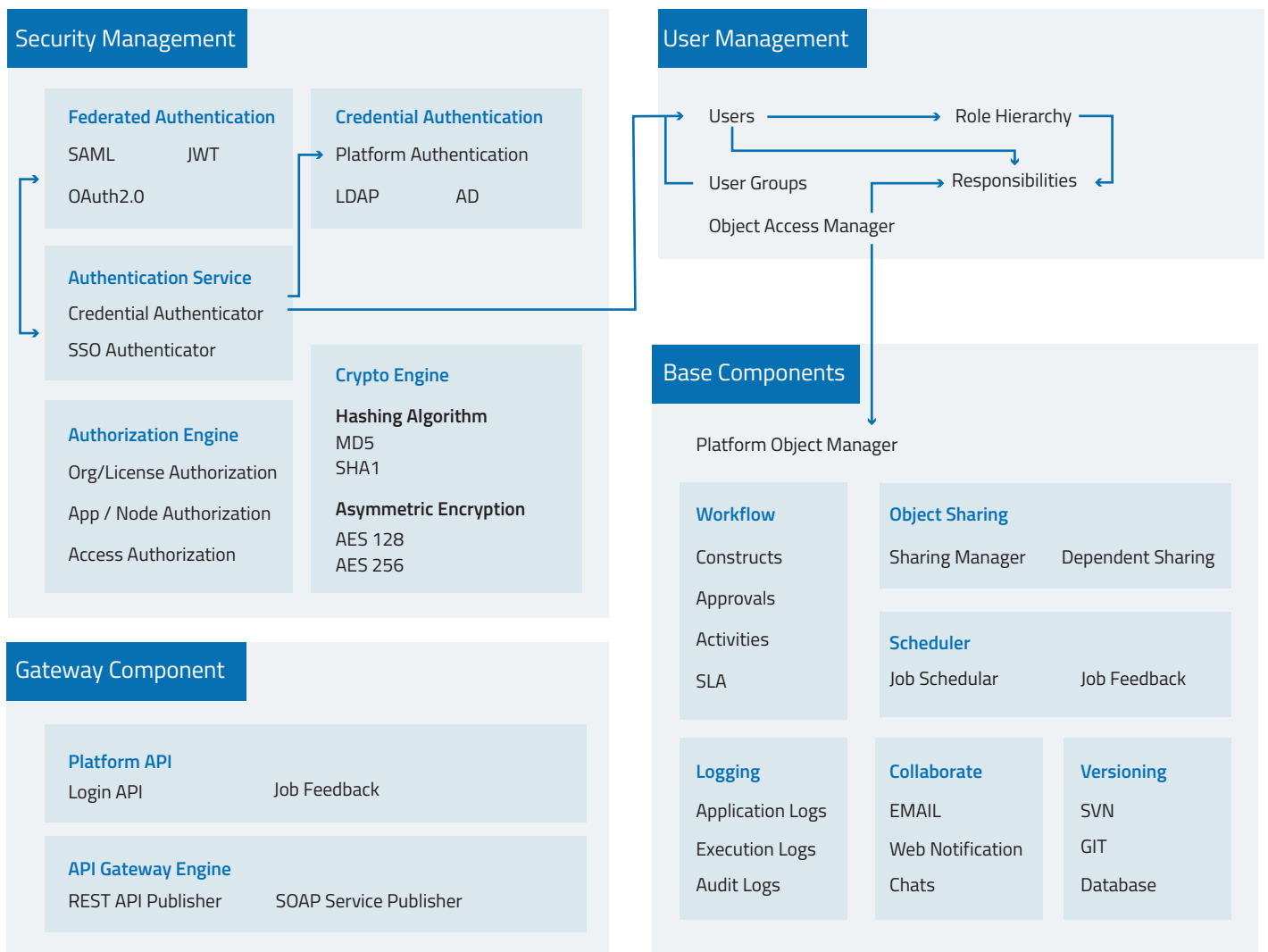
- The system's overall architecture goals are to provide a highly available, scalable, & flexible data management platform
- A key Architectural goal is to leverage industry best practices to design and develop a scalable, enterprise-wide J2EE application and follow the industry-standard development guidelines.
- All aspects of Security must be developed and built within the application and be based on Best Practices.

## Platform Component Definition

Foundation	Smart data platform			Smart Business Platform	
<b>Security</b> Authentication / Authorization / Crypto	 <ul style="list-style-type: none"> <li>• Data Migration</li> <li>• Setup Migration</li> <li>• Test data Prep</li> <li>• Big Data Ingestion</li> <li>• Data Archival</li> <li>• Data Reconciliation</li> <li>• Data Integration</li> </ul>	 <ul style="list-style-type: none"> <li>• Data Quality Management</li> <li>• Master Data Governance</li> <li>• Analytical MDM (Customer 360, Supplier360, Product 360)</li> </ul>	 <ul style="list-style-type: none"> <li>• Data Masking</li> <li>• Data Compliance (PII, GDPR, CCPA, OIOO)</li> <li>• Data Cataloging</li> <li>• Data Analytics</li> <li>• Data Visualizat</li> </ul>	 <ul style="list-style-type: none"> <li>• Used for Autonomous Regression Testing</li> <li>• Used for Load and Performance Testing</li> </ul>	 <ul style="list-style-type: none"> <li>• Rapid Application Development (RAD) Framework</li> <li>• Visual Development Approach</li> <li>• Drag &amp; Drop Design Tools</li> <li>• Functional Components into Visual Workflow</li> </ul>
<b>User Management</b> User / Groups Roles / Responsibility Access Manager	<div>      </div>				
<b>Base Component</b> Workflow Versioning Notification Logging Scheduler Object Manager	<div>          </div>				
<b>Gateway Component</b> API Gateway					

The Platform Foundation forms the base on which the entire Platform is built. The major components that create the Platform are described in brief.

## Platform Foundation



# User Management

The component manages all the Roles, Responsibilities, Hierarchy, Users, & User Groups.



## Responsibilities

The Platform comes with the preconfigured Responsibilities for dataZap, dataZen, and dataZense. Organizations can customize Responsibilities and are assigned to the platform objects with additional privileges.



## Roles

The Platform comes with the predefined Roles for dataZap, dataZen, and dataZense. Organizations can create their Roles. The Role-based hierarchy is also configured for the user level hierarchy. The roles are assigned with the default responsibilities.



## Users

The users will be assigned these applications that are necessary for them. The User will be given a Role. The hierarchy formed using the role hierarchy setup where a manager from the next role is assigned. The responsibility against these roles is set by default for the users. The User can be given more responsibilities or revoked an existing responsibility against a role. Users gain access to the objects based on the privileges assigned for the responsibility.



The security management component takes care of the following

## Security Management

### SSL

The Platform is SSL / HTTPS enabled on the transport layer with TLS 1.2 support. The SSL is applied to the nodes exposed to the users like the DMZ nodes and Web nodes and the nodes exposed to the third-party applications like the API Gateway nodes.

### Authentication Engine

The Platform offers a Credential based authentication handled by the Platform and also Single Sign-On based federated authentication. Both SSO and Credential authentication can co-exist for an organization.

#### Credential Authentication

User authentication on the Platform happens with the supplied credentials. All the successful sessions are logged, and failed attempts are tracked at the user level for locking the user account. A user can have only one web session at any given point in time. Password policy, including expiry, is configured at the Organization level, applicable for all users. Enforced password complexity like.

- Min length
- Max length
- Usage of Numbers, Cases, and Special Characters can be set.
- No of unsuccessful attempts are also configurable

#### Single Sign-On

SSO can be set up with federated services like SAML, OAuth 2.0, or JWT (Java Web Tokens). Setup for an IDP would be configured against the organization profile, and authentication would happen in the IDP. This can either be IDP initiated or SP (Chainsys Smart Data Platform) initiated. The organization users with SSO would get a different context to login.



## Authorization Engine

- The first level of authorization would be the Organization License. The Licensing engine would be used to setup the organization for the authentications too
- The next level of authentication would be the Applications assigned to the Organization and the respective User. The individual application nodes would be given to the organization as per the service agreement to handle load balancing and high availability
- Authorization of pages happens with the responsibilities assigned to the users
- Authorization of a record happens concerning sharing the records to a group or individual users
- Responsibility and sharing will have the respective privileges to the pages and records
- On conflict, the Principle of least privilege is used to determine the access

## Authorization Engine

- The Crypto Engine handles both asymmetric encryption and hashing algorithms
- AES 128 is the default encryption algorithm but also supports 256 bits
- The keys are managed within the Platform at the organization level. The usage of keys maintained at the application level determines how they are used for encryption and decryption.
- All the internal passwords are being stored by default with MD5 hashing
- Encryption of the stored data can be done at the Database layer as needed

## Authorization Engine

The workflow engine is created to manage the orchestration of the flow of activities. The workflow engine is part of the platform foundation extended by the applications to add application-specific activities.

## Version Management

This component helps in handling the version of objects and records eligible for versioning. The foundation has the API to version the objects and its records and can be extended by the applications to add specific functionalities. Currently, the Platform supports SVN as default and also supports database-level version management. Support for GIT is on the roadmap.

## Notification Engine

The notification engine is the component that will do all the notifications to the User in the system. The feature helps notify the users on the page when online in the application. The other notifications like Mail notification and Chat Notification are also part of this component.

## Logging Engine

All activity logs, both foundation, and application are handled to understand and help in the debugging.



## Scheduler Creation

It enables you to schedule a job once or regularly. In terms of recurring jobs are planned minutely, hourly, weekly, and monthly.

## Scheduler Execution

The scheduler execution engine uses the configuration and fires the job in the respective application. The next job would be scheduled at the end of each job as per the configuration.

## Job Monitoring

The scheduled jobs are monitored and keep track of the progress and status at any stage. If there is any delay in the expected job or unexpected errors, the responsible users are notified accordingly for actions.

The API Gateway forms the foundation for publishing and consuming services with the Platform. All the eligible jobs or actions can be published for external applications to access. The following are the components that would form the publishing node.

## API Gateway Engine

## Login Service

The login service is the one that authenticates if the requested consumer has the proper authentication or credentials to invoke the job or action. The publisher engine has two methods of authentication.

- Inline authentication - where all the requests will have the Credential for authentication and access control
- Session Authentication - This service is explicitly invoked to get the token and gather the other published services using this token to authorize the request.

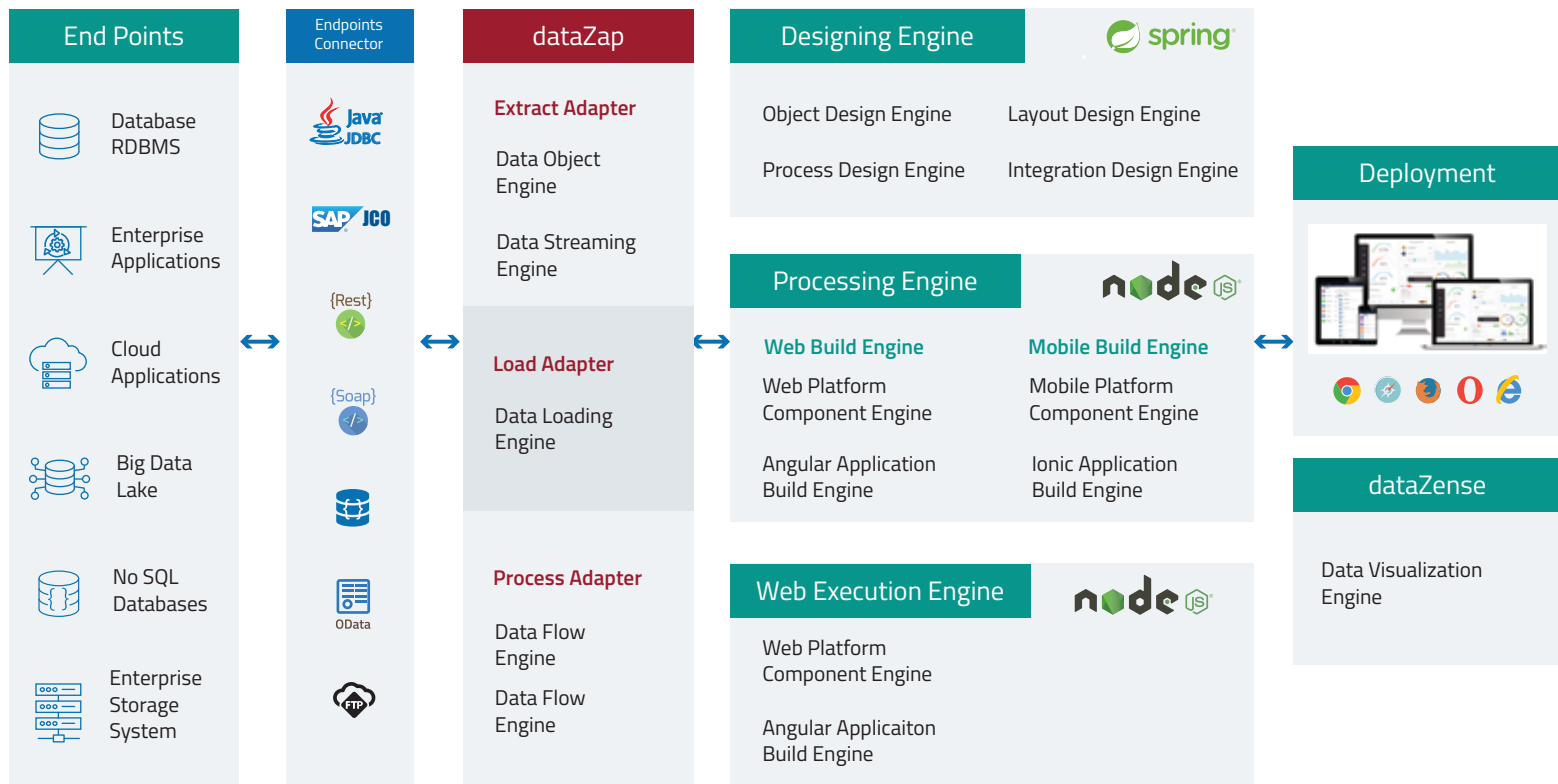
## SOAP Service

The eligible jobs or actions can be published using the Simple Object Access Protocol (SOAP). SOAP is a messaging protocol that allows programs that run on disparate operating systems to communicate using Hypertext Transfer Protocol (HTTP) and its Extensible Markup Language (XML).

## REST Service

The eligible jobs or actions can be published using the Representational State Transfer Protocol (REST). REST communicates using the HTTP like SOAP and can have messages in multiple formats. In dataZap, we will publish in the XML format or the JSON (JavaScript Object Notation) format.

# Smart App Builder



Object configuration helps you define the object with a collection of attributes (Example: Single line, Number, Date, Date & Time, etc. An object is a data layer for your application.

## Object Design Engine

Object configuration has the following features



File management



Calendar



Field tracking



Multi-language



Notifications

## Layout Design Engine

In Smart App Builder, with 30+ field types, customizable themes, and templates, it helps you create just about any form you need. You can use the Smart App Builder web interface to configure your layouts, and it will be synced into a web and mobile container based on the assignment after successful login. Whenever an object is created, a full set of navigable layouts will be generated for the basic View, Edit & New feature based on its relationships.



# Developer Template

Smart App Builder is a No-Code/ Low-Code platform. We can customize the layouts using Angular and upload our layouts into Smart App Builder through developer layout configuration for the customer requirement.

## Action configuration

Action configuration helps create multiple actions for your custom application, such as SMS action, Phone call action, Web-service call, and navigation to different configured layouts.

## Process Design Engine

This component overrides the workflow component in the foundation. The process flow engine is specific to the application business process workflow. The feature has all the orchestration capabilities and human intervention capabilities like Approval, User Confirmation, and Receive Input.

## Integration Engine

It uses DataZap to establish a connection with all the supported enterprise endpoints for the data.

## Processing Engine

The processing engine is the NodeJs environment. It will generate the configured layouts using an Angular and ionic framework.

## App configuration

App configuration helps to group multiple layouts and name it as your own to categories your layouts. You can assign this application to individual users, user groups, or role- based users.

## Web/Mobile Execution Engine

The web execution engine is the NodeJs environment. It handles all the web and mobile application requests.

## Deployment

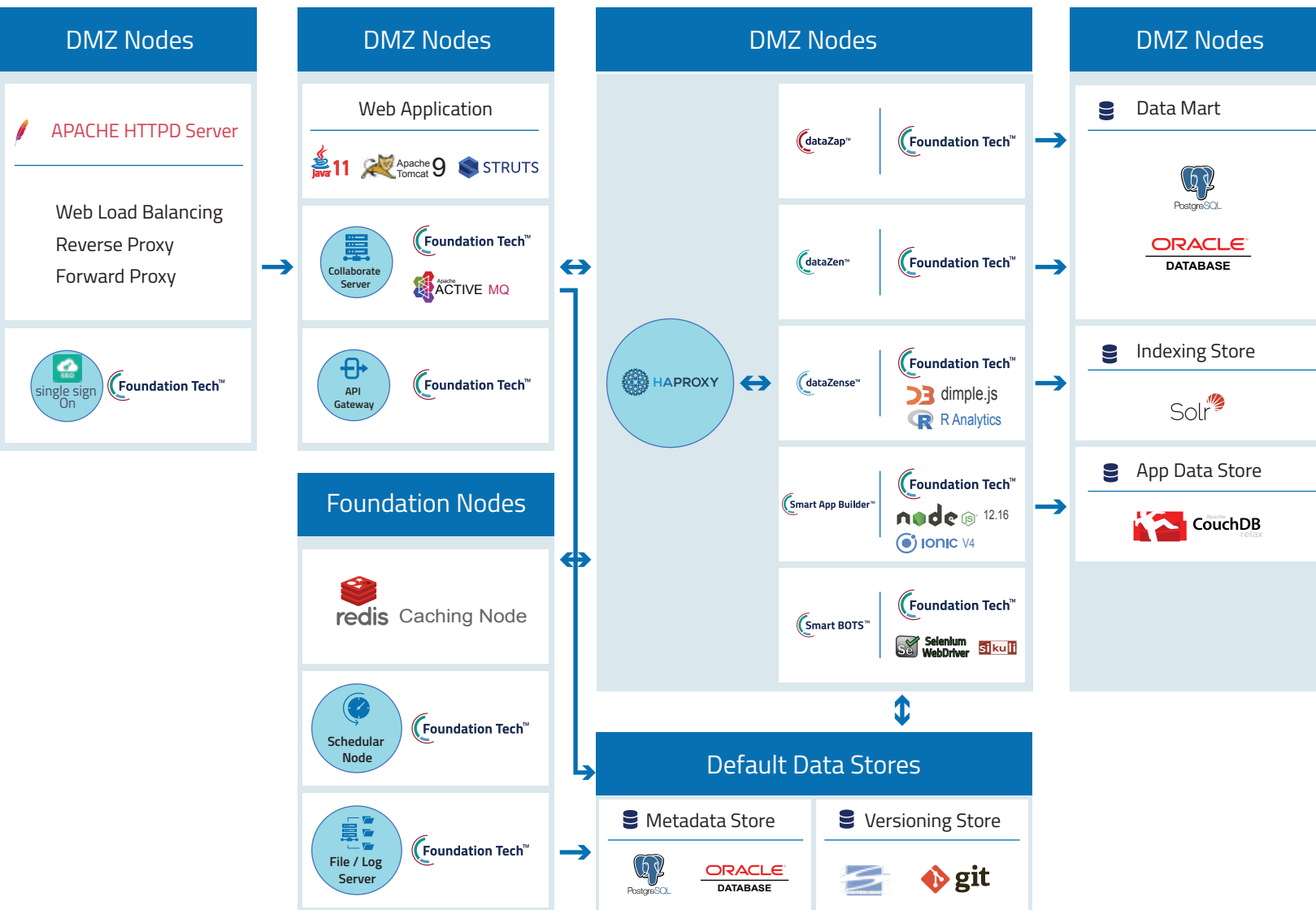
Quick deployment for mobile/web. With our mobile apps for iOS and Android, you can access all your mobile apps on the go.

Single Data Management Platform, solves all your data management needs.

- Distributed computing helps in scaling both horizontal and vertical.
- Battle tested with several Fortune 500 Organizations.

Many Fortune 500 Organizations are well poised to select ChainSys for their data projects.

## System Technology Landscape



## DMZ Nodes

These nodes are generally the only nodes exposed to the external world outside the enterprise network. The two nodes in this layer are the Apache HTTPD server and the "Single Sign O" Node.

## Apache HTTPD

The Apache HTTPD server is used to route the calls to the Web nodes. The server also handles the load balancing for both the Web Server Nodes and the API gateway Nodes. The following features are used in the Apache HTTPD

- Highly scalable
- Forward / Reverse proxy with caching
- Multiple load balancing mechanisms
- Fault tolerance and Failover with automatic recovery
- WebSocket support with caching
- Fine-grained authentication and authorization access control
- Loadable Dynamic Modules like ModSecurity for WAF etc.
- TLS/SSL with SNI and OCSP stapling support



## Web Nodes

This layer consists of the nodes exposed to the users for invoking the actions through frontend or a third-party application as API's. The nodes available in this layer would be the Web Server to render the web pages, API Gateway for other applications to interact with the application, and the collaborate node for notifications.

## Single Sign-On

This Node is built on the Spring Boot application with Tomcat as the Servlet container.

Organizations opting to have a single sign-on would have a separate SSO node with a particular context. The default context will take them to the platform-based authentication.

# Web Server

The web application server hosts all the web pages of the chainsys platform.

- Apache Tomcat 9.x is used as the servlet container.
- JDK 11 is the JRE used for the application. The Platform works on OpenJDK / Azul Zulu / AWS Corretto and Oracle JDK.
- Struts 1.3 platforms are used as the controllers
- Integration between the webserver to the application nodes is handled with Microservices based on the SpringBoot
- The presentation layer uses HTML 5 / CSS 3 components and uses many scripting frameworks like JQuery, d3js, etc.
- The web server can be clustered to n- nodes as per the number of concurrent users and requests.

# Gateway Node

This Node uses all the default application services.

- This Node uses the service of Jetty to publish the API as SOAP or REST API.
- The API Gateway can be clustered based on the number of concurrent API calls from the external systems.
- The Denial of Service (DoS) is accomplished in both JAX-WS and JAX-RS to prevent illegal attacks.

# Collaborate

This Node is used to handle all different kinds of notifications to the users like front end notifications, emails, push notifications (in the roadmap). This Node also has the chat services enabled that can be used by the applications as needed

- The notification engine uses netty APIs for sending notification from the Platform.
- Apache Active MQ is used for messaging the notification from application nodes.

## Application Nodes

- The application nodes are spring boot applications for communicating between the other application nodes and web servers.
- JDK 11 is the JRE used for the application. The Platform works on OpenJDK / AzulZulu / AWS Corretto and Oracle JDK.
- Load Balancing is handled by the HAProxy based on the number of nodes instantiated for each application.



The application uses only the default services that are mentioned above.



The application uses only the default services that are mentioned above.



The application uses all the default services that are mentioned above. In addition to this, it also uses R analytics for Machine Learning algorithms. It also uses D3 and Dimple JS for the visual layer.







The application uses all the default services that are mentioned above. In addition to this, it also uses the Selenium API for web-based automation and Sikuli.



The application uses all the default services that are mentioned above. These services are used to configure the custom applications and to generate dynamic web applications as configured.

The mobile applications' service would need NodeJS 12.16, which would use the IonicFramework V4 to build the web and mobile apps for the configured custom applications.



## Scheduler Node

This Node uses only the default application node services.

- This Node can be clustered only as failover nodes.
- When the primary Node is down, the HAProxy makes the secondary Node the primary Node
- The secondary Node handles notifications, automatic rescheduling of the jobs. It calls each of the application objects that are schedulable to take all the possible exception scenarios to be addressed.
- Once the Node is up and running, this will become the secondary Node.

## Data Storage Nodes

### Database

Chainsys Platform supports both PostgreSQL 9.6 or higher and Oracle 11g or higher databases for both

- Metadata of the setups and configurations of the applications
- Data marting for the temporary storage of the data.

The Platform uses PostgreSQL for the Metadata in the cloud. PostgreSQL is a highly scalable database.

Designed to scale vertically by running on more significant and faster servers when you need more performance

1

Can be configured to do horizontal scaling, Postgres has useful streaming replication features so you can create multiple replicas that can be used for reading Data

2

It can be easily configured for High Availability based on the above.

3

- Password Storage Encryption
- Encryption For Specific Columns
- Data Partition Encryption
- Encrypting Passwords Across A Network
- Encrypting Data Across A Network
- SSL Host Authentication
- Client-Side Encryption

PostgreSQL offers encryption at several levels and provides flexibility in protecting data from disclosure due to database server theft, unscrupulous administrators, and insecure networks. Encryption might also be required to secure sensitive data.

Multi-tenant database architecture has been designed based on the following

- Separate Databases approach for each tenant
- Trusted Database connections for each tenant
- Secure Database tables for each tenant
- Easily extensible Custom columns
- Scalability is handled on Single Tenant scaleout

## Cache Server

Redis cache is used for caching the platform configuration objects and execution progress information.

• This helps to avoid network latency across the database and thus increases the performance of the application.

• When the durability of Data is not needed, the in-memory nature of Redis allows it to perform well compared to database systems that write every change to disk before considering a transaction committed.

• The component is set up as a distributed cache service to enable better performance during data access.

• Redis cache can be made HA enabled clusters. Redis supports master-replica replication



## File Log Server

This component is used for centralized logging, which handles the application logs, execution logs, and error logs in the platform applications' common server. Log4J is used for distributed logging. These logs can be downloaded for monitoring and auditing purposes. A small Http service gets executed, which allows the users to download the file from this component—implemented with the Single Tenant scaleout approach.

## Subversion (SVN) Server

Apache Subversion (abbreviated as SVN) is a software versioning and revision control system distributed as open-source under the Apache License. The Platform uses SVN to version all the metadata configurations to revert in the same instance or move the configurations to multiple instances for different milestones. All the applications in the Platform use the foundation APIs to version their objects as needed.



Loader Adapters,  
Data Objects,  
Data Extracts,  
Data Flows,  
Process Flows,  
Migrations Flows,  
Reconciliations



Data Model,  
Rules,  
Augmentations,  
Workflow



Data Set,  
Views,  
Dashboards,  
Ad-hoc Reports



Object Model,  
Layouts,  
Workflow

ChainSys Platform uses SOLR for the data cataloging needs as an indexing and search engine.

Solr is an open-source enterprise-search platform. Its major features include full-text search, hit highlighting, faceted search, real-time indexing, dynamic clustering, database integration, NoSQL features, and rich document handling.

Apache Solr was used over the others for the following reasons.

# Apache SOLR

## Real-Time, Massive Read, and Write Scalability

Solr supports large-scale, distributed indexing, search, and aggregation/statistics operations, enabling it to handle large and small applications. Solr also supports real-time updates and can take millions of writes per second.

## SQL and Streaming Expressions/Aggregations

Streaming expressions and aggregations provide the basis for running traditional data warehouse workloads on a search engine with the added enhancement of basing those workloads on much more complex matching and ranking criteria.

## Security Out of the Box

With Solr, Security is built-in, integrating with systems like Kerberos, SSL, and LDAP to secure the design and the content inside of it.

## Fully distributed sharding model

Solr moved from a master-replica model to a fully distributed sharding model in Solr 4 to focus on consistency and accuracy of results over other distributed approaches.

## Cross-Data Center Replication Support

Solr supports active-passive CDCR, enabling applications to synchronize indexing operations across data centers located across regions without third-party systems.

## Solr is highly Big Data enabled

Users can store Solr's data in HDFS. Solr integrates nicely with Hadoop's authentication approaches, and Solr leverages Zookeeper to simplify fault tolerance infrastructure.

## Documentation and Support

Solr has an extensive reference guide that covers the functional and operational aspects of Solr for every version.

## Solr and Machine Learning

Solr is actively adding capabilities to make LTR an out of the box functionality.

# Apache CouchDB

Chainsys Platform uses CouchDB for mobile applications in the Application Builder module. PostgreSQL would be the initial entry point for the Dynamic Web Applications. The data in the PostgreSQL will sync with CouchDB if mobile applications are enabled. In contrast, the initial entry point for the Dynamic Mobile Applications would be in the PouchDB. CouchDB syncs with the PouchDB in the mobile devices, which then syncs with PostgreSQL. The main feature for having CouchDB are

- CouchDB throws the HTTP and REST as its primary means of communication out the window to talk to the database directly from the client apps.
- The Couch Replication Protocol lets your Data flow seamlessly between server clusters to mobile phones and web browsers, enabling a compelling offline-first user-experience while maintaining high performance and strong reliability.
- Another unique feature of CouchDB is that it was designed from the bottom-up to enable easy synchronization between different databases.
- CouchDB has JSON as its data format.



## Distributed Mode

Chainsys Smart Data Platform is a highly distributed application and with a highly scalable environment. Most of the nodes are horizontally and vertically scalable.

## Deployment at Customer

### DMZ Services VM



APACHE HTTPD Server



single sign on

### Web Container Cluster

Web Page Services



Node1 ..... Node n

Collaborate Services



Node1

API Gateway

Node1 ..... Node n

### Foundation Services Cluster

Foundation Services Cluster



Caching Node

File/Log Services

Node1

Scheduling Services

Primary Node ..... Secondary Node

### Smart Data Platform Cluster



Web Page Services

Node1 ..... Node n



Web Page Services

Node1 ..... Node n



Web Page Services

Node1 ..... Node n

Web Page Services

Node1 ..... Node n



Node1 ..... Node n



Design & Process

Node1 ..... Node n

Layout Build

Node1



Layout Rendering

Node1 ..... Node n

### Database Layer

Versioning VM



Database Cluster



ORACLE  
DATABASE

Primary Node

- Metadata
- Datamart

Secondary Node

- Metadata
- Datamart

SOLR Cluster



Master Node

- Core 1
- Core 2

Slave node

- Core 1
- Core 2

CouchDB Cluster



RELAX

Node 1

- Doc 2
- Doc 2

Node 1

- Doc 2
- Doc 2

HAPROXY

Load  
Balancer

## DMZ Nodes

- Apache HTTPD would be needed in a distributed environment as a load balancer. This would also be used as a reverse proxy for access outside the network. This would be a mandatory node to be available.
- SSO Node would be needed only if there is a need for the Single-Sign-On capability with any federated services.

## Web Cluster

- Chainsys recommends having a minimum of two web node clusters to handle high availability and Load balanced for better performance. This is a mandatory node to be deployed for the Chainsys Platform.
- The number of nodes is not restricted to two and can be scaled as per the application pages' concurrent usage.
- The Collaborate node generally is a single node, but the Node can be configured for High Availability if needed.

## Gateway Cluster

- The API Gateway Nodes are not mandatory to be deployed. It would be required only when there is a need to expose the application APIs outside the Platform.
- When deployed, Chainsys would recommend having a two-node cluster to handle high availability and load balancing in high API call expectations.
- The number of nodes in the clustered can be determined based on API calls' volume and is not restricted to two.

## Application Cluster

- The HAProxy or Apache HTTPD acts as the load balancer. All the calls within the application nodes are handled based on the node configuration. If the Apache HTTPD is used in the DMZ for Reverse Proxy, it is recommended to have HAProxy for internal routing or a separate Apache HTTPD.
- The number of nodes in the cluster is not restricted to two. Individual application nodes can be scaled horizontally for load balancing as per the processing and mission-critical needs.
- Integration Cluster is a mandatory node that will be deployed in the Platform. All the other applications depend on this application for all the integration needs.
- Visualization Cluster is also a mandatory node that will be deployed in the Platform. All the other applications depend on this application for all the dashboard report needs.



- The visualization uses the R Studio Server for Machine Learning capabilities. It is needed only when the Machine Learning algorithms are to be used.
- When deploying the MDM, the "Smart Application Builder" node would be needed for the dynamic layout generation and augmentation. The vice versa doesn't apply as "Smart Application Builder" is not dependent on the MDM nodes.
- NodeJS would be needed only when mobile applications are to be dynamically generated. The Apache HTTPD server will handle load balancing.
- The Scheduler cluster would be needed even if one of the applications use the scheduling capability. The cluster would only be a High Availability (Failover) and not load balanced. The number of nodes is restricted to two.

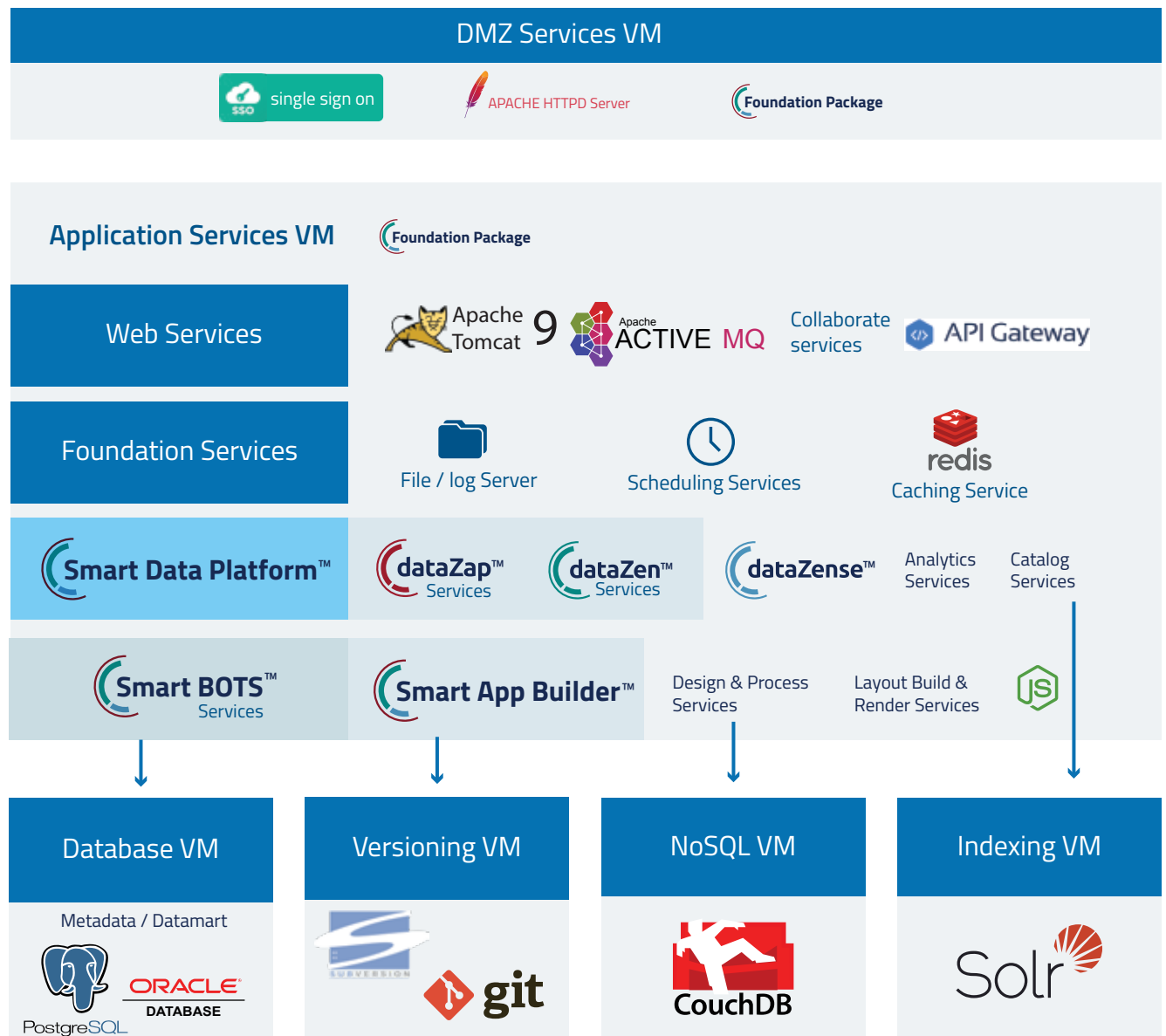
## Data Storage Nodes

- Generally, the PostgreSQL database would be configured for High Availability as an Active - Passive instance. Depending on the number of read-write operations, it can be Load balanced too. This can be replaced by Oracle 11g or more significant if the client wants to use the existing database license.
- File Server would be needed only if there is no NAS or SAN availability to mount the same disk space into the clusters to handle the distributed logging. The NFS operations for distributed logging would require this Node.
- SVN server would be mandatory to store all the configuration objects in the repository for porting from one instance to the other. Generally, it would be a single node as the operation on this would not be too high.
- REDIS is used as a cache engine. It is mandatory for distributed deployment. This can be configured for high availability using the master-slave replication.
- SOLR would be needed only if data cataloging is implemented, and search capability is enabled. This can be configured for High Availability. SOLR sharding can be done when the Data is too large for one Node or distributed to increase performance/throughput.
- CouchDB would be needed only if dynamic mobile applications are to be generated. CouchDB can be configured for high availability. For better performance, Chainsys recommends having individual instances of CouchDB for each active application.



# Single Node

Single Node does not mean that literally. Here we would say that all application services produced by the ChainSys Platform are deployed in a Single Node or Server. The rest of the data storage nodes are separate servers or nodes. This type of installation would generally be for a patching environment where there are not too many operations. These would also be recommended for non-mission critical development activities where high availability and scalability are not a determining factor.



## DMZ Nodes

- Apache HTTPD would be needed only if a reverse proxy is required for access outside the network. This is not a mandatory node for a Single Node installation.
- SSO Node would be needed only if there is a need for the Single-Sign-On capability with any federated services.

## Application Server

- There will be just one Apache Tomcat as the web application service and will not be configured for high availability.
- Collaborate service will have the Apache ActiveMQ and the spring integration service.
- The API Gateway would be required only if the objects are to be published as a REST API or SOAP Service. This service can be shut down if not needed.
- The Integration Service, Visualization Service, and Scheduler Service would be mandatory services running.
- The rest of the applications would be running or shut down depending on the license and need.

## Data Storage Nodes

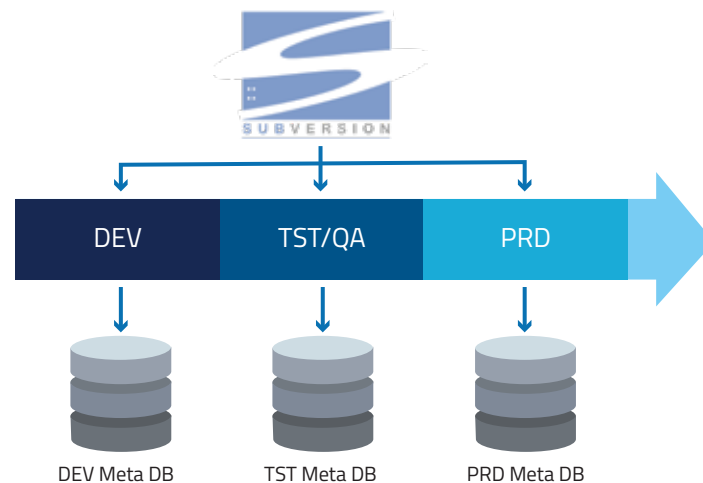
- PostgreSQL would be in a separate node. Chainsys does not recommend having the applications and the Databases on the same machine.
- SVN server would be mandatory to store all the configuration objects in the repository for porting from one instance to the other.
- SOLR would be needed only if data cataloging is implemented, and search capability is enabled.
- CouchDB would be needed only if dynamic mobile applications are to be generated as a separate node.



## Instance Strategy

Built-in Configuration management approaches for check-in and check-out without leaving ChainSys Platform.

- Gives a great Software development lifecycle process for your projects.
- All your work is protected in a secure location and backed up regularly.

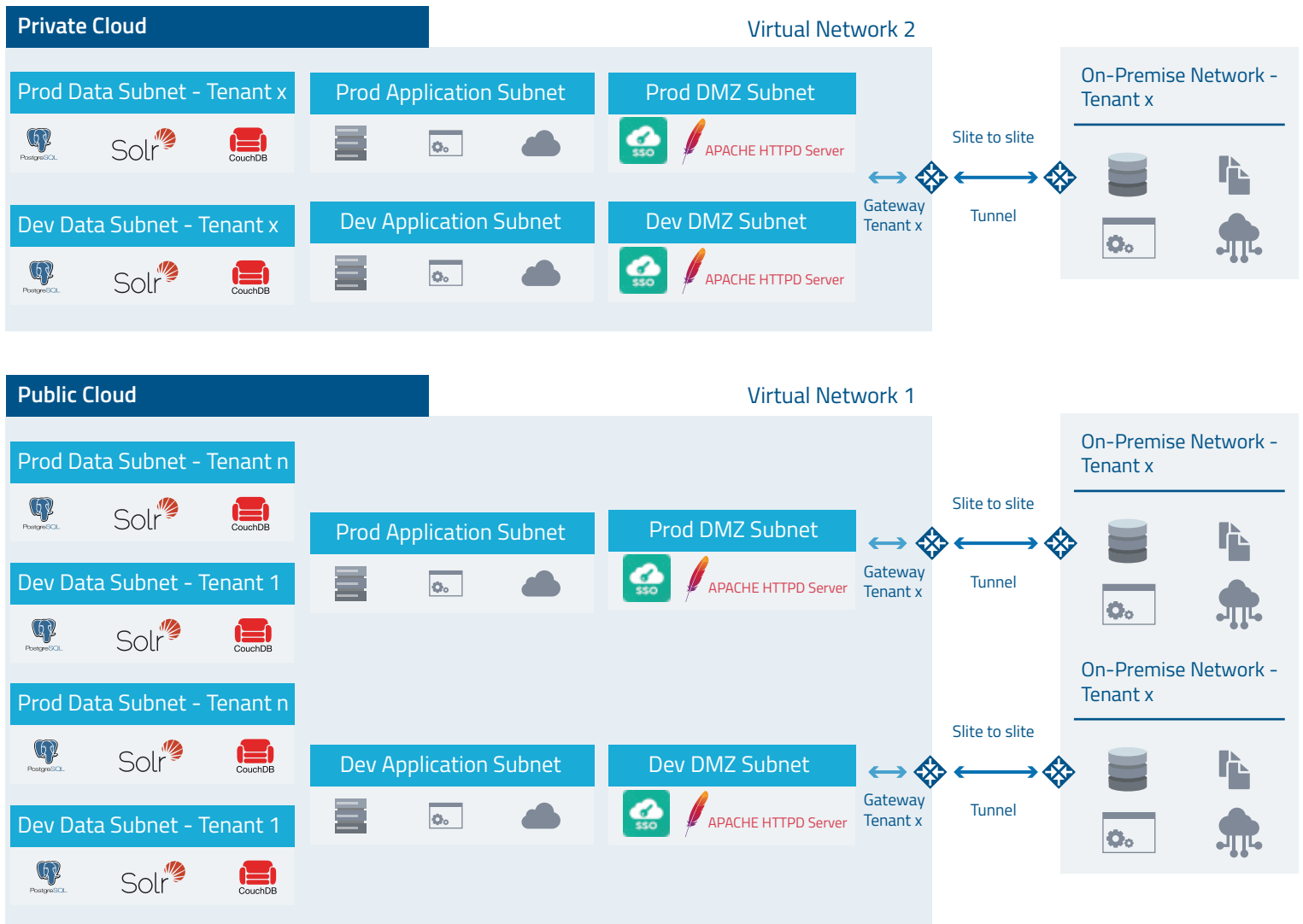


Generally, the above instance propagation strategy is recommended. Depending on the applications in use and the Load, it could be determined to go with a single node deployment or a distributed model deployment. Generally, it is recommended to have a distributed deployment for Production instances. The adapters are forward propagated using the SVN repository.

All the instances need not follow the same deployment model. For the reverse propagation of the example from Production to Non-Production instances, we can clone the application and the data storage layer and have the node configurations re-configured to the lower instances.

ChainSys Platform is available on the cloud.  
The Platform has been hosted as a  
Public Cloud and also has the  
Private Cloud options.

## Pure Cloud Deployment



## Public Cloud

- The Site would handle connectivity to the Customer Data Center to Site tunneling between the Tenants Data Center and the ChainSys Data Center. Individual Gateway Routers can be provisioned per tenant.
- Tenants will share the same application and DMZ node clusters except the data storage nodes.
- If a tenant needs to be assigned a separate application node for the higher workloads, we can have the particular application node-set only for that specific tenant.
- As mentioned earlier in the Database section, Multi-Tenancy is handled at the database level. Tenants will have separate database instances
- The databases would be provisioned based on the license and the subscription.
- Depending on the workload on the nodes, each Node can be clustered to balance the Load.



## Private Cloud

Customers (Tenants) will have all applications, DMZ nodes, and data storage nodes assigned to the specific tenant and are not shared.

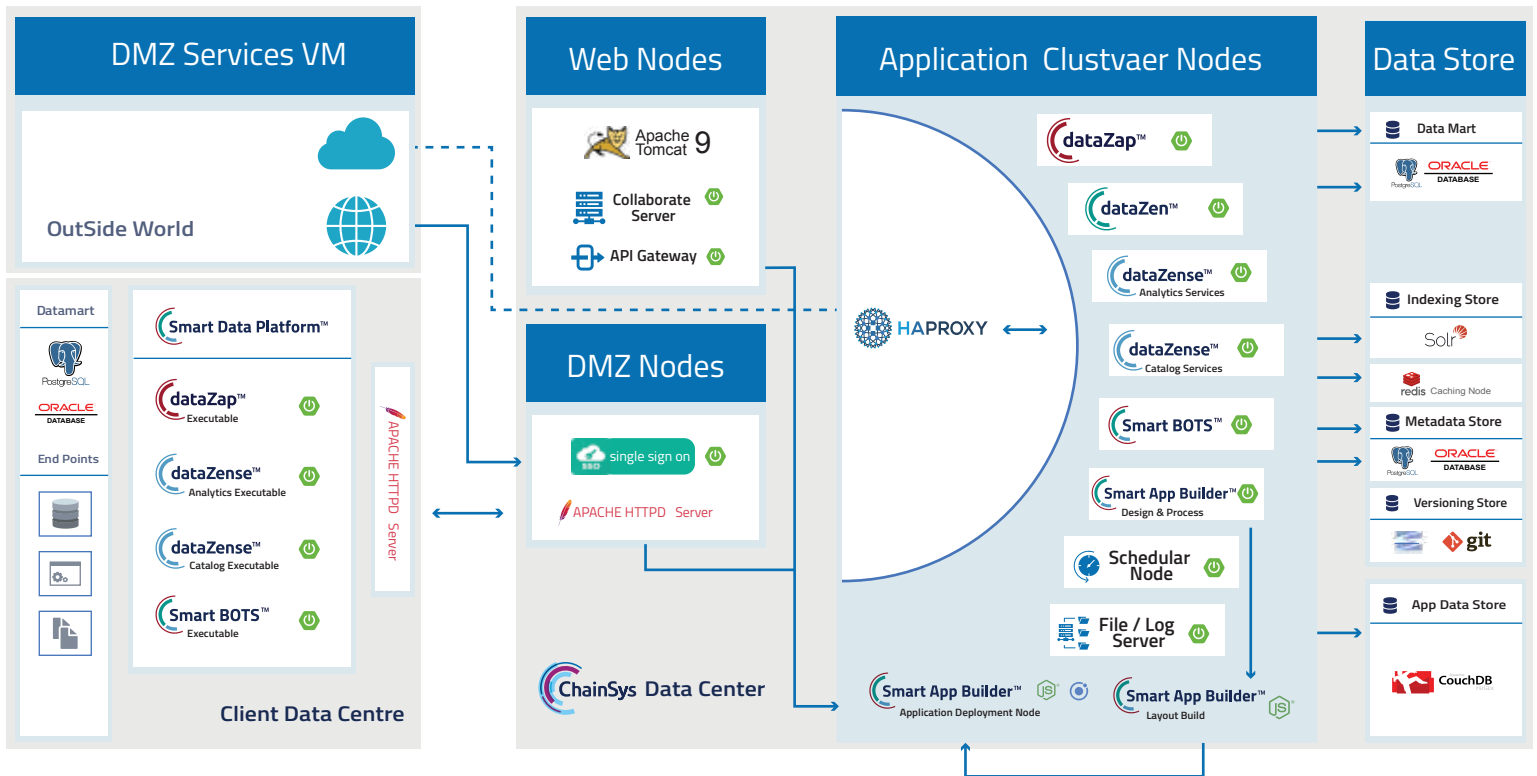
Depending on the workload on the nodes, each Node can be clustered to balance the Load.

The application nodes and databases would be provisioned based on the license and subscription.



# Hybrid Cloud

## Hybrid Cloud Deployment

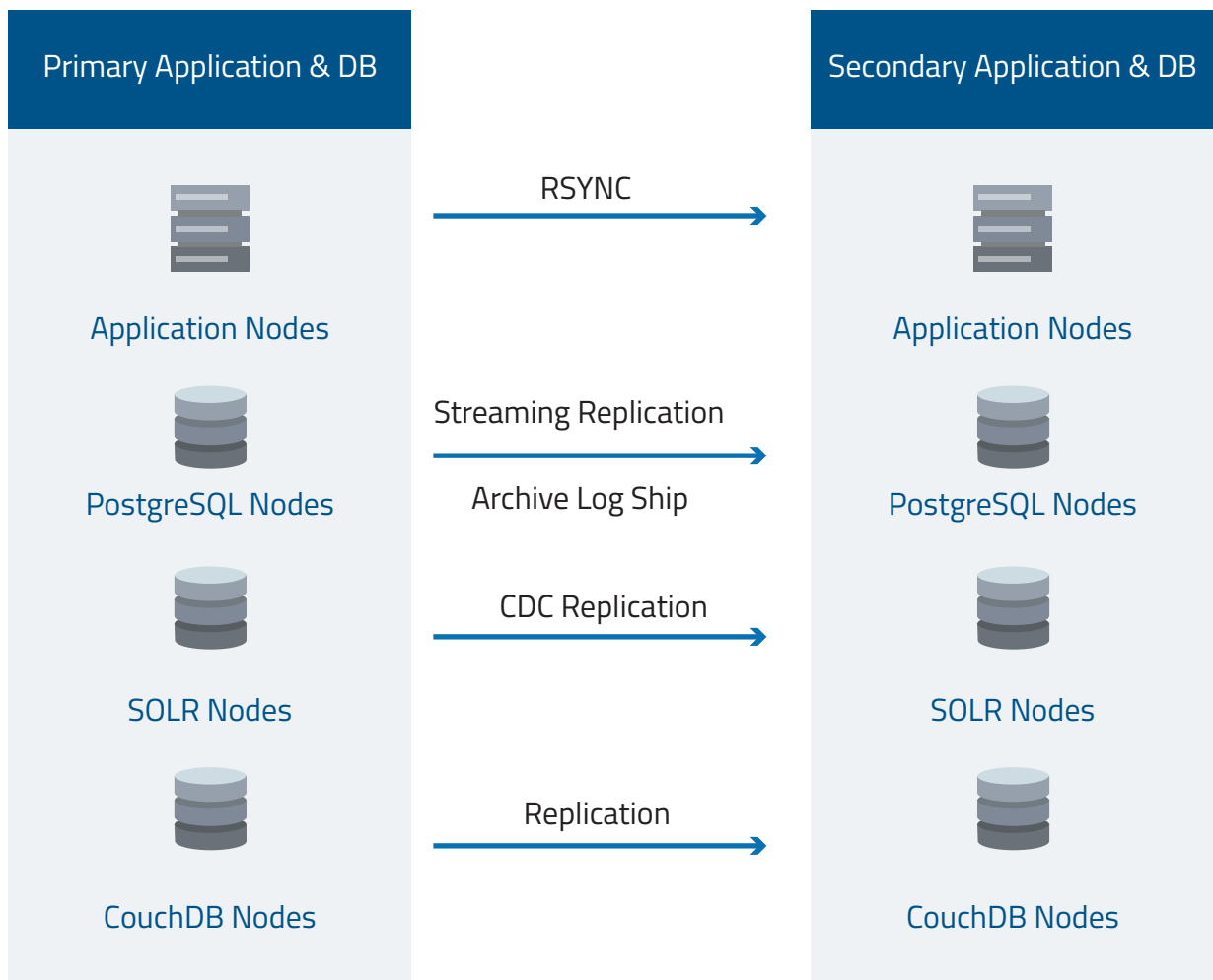


This can be associated along with both the Private or Public cloud. An Agent would be deployed in the client organization's premises or data center to access the endpoints. This would avoid creating the Site to Site tunnel between the Client Data Center and the ChainSys Cloud Data Center.

There is a proxy (Apache HTTPD Server) on both sides, the ChainSys Data Center and the Client Data Center. All the back and forth communications between the ChainSys Data Center and the Agent are routed through the proxy only. The ChainSys cloud sends instructions to the Agent to start a Task along with the Task information. The Agent executes the Task and sends back the response to the cloud with the Task's status.

- The Agents (for dataZap, dataZense, and Smart BOTS) would be deployed.
- For dataZap, we can use the existing database (either PostgreSQL or Oracle) for the staging process. The Agent executes all integration and migration tasks by connecting directly to the source and target systems, validating and transforming data, and transferring data between them.
- For dataZense and Smart App Builder, data would be streamed to the Chainsys Platform to manipulate the data.

# Disaster Recovery



- All the application nodes and the web nodes would be replicated using the RSYNC. The specific install directory and any other log directories would be synced to the secondary replication nodes.
- For PostgreSQL, the Streaming replication feature would be used, which used the archive log shipping.
- SOLR comes up with the in-built CDCR (Cross Data Center Replication) feature, which can be used for disaster recovery.
- CouchDB has an outstanding replication architecture, which will replicate the primary database to the secondary database.
- The RPO can be set to as per the needs individually for both Applications and Databases
- The RTO for the DR would be approximately an hour.



ChainSys uses third party monitoring open-source tools such as Zabbix and Jenkins to monitor all the nodes. Zabbix supports tracking the Servers' availability and performance, Virtual Machines, Applications (like Apache, Tomcat, ActiveMQ, and Java), and Databases like PostgreSQL, Redis, etc.) that are used in the Platform. Using Zabbix, the following are achieved

## Application Monitoring

### Third-Party Monitoring Tools

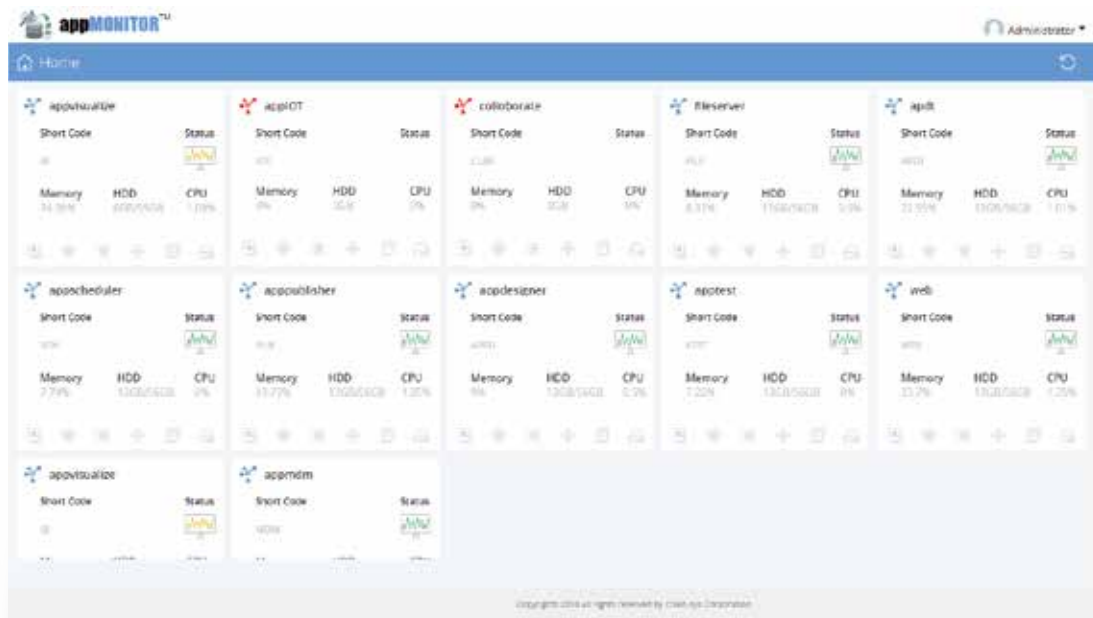


- Various data collection methods and protocols
- Start to monitor all metrics instantly by using out-of-the-box templates
- Flexible trigger expressions and Trigger dependencies
- Proactive network monitoring
- Remote command execution
- Flexible notifications
- Integration with external applications using Zabbix API

We can also use the individual application monitoring systems for more in-depth analysis but having an integrated approach to looking into the problems helps us be proactive & faster.

# In-Built Monitoring System

ChainSys is working on its Application Monitoring tool that monitors the necessary parameters like the CPU / Memory. This tool is also planned to help monitor individual threads within the application. It is also intended to do most maintenance activities like patching, cloning, and database maintenance from one single toolset. This will be integrated with Zabbix for monitoring and alerting systems.



## Supported Endpoints ( Partial )

Oracle Sales Cloud, Oracle Marketing Cloud, Oracle Engagement Cloud, Oracle CRM On Demand, SAP C/4HANA, SAP S/4HANA, SAP BW, SAP Concur, SAP SuccessFactors, Salesforce, Microsoft Dynamics 365, Workday, Infor Cloud, Procore, Planview Enterprise One

Cloud  
Applications

Oracle E-Business Suite, Oracle ERP Cloud, Oracle JD Edwards, Oracle PeopleSoft, SAP S/4HANA, SAP ECC, IBM Maximo, Workday, Microsoft Dynamics, Microsoft Dynamics GP, Microsoft Dynamics Nav, Microsoft Dynamics Ax, Smart ERP, Infor, BaaN, Mapics, BPICS

Enterprise  
Applications

Windchill PTC, Oracle Agile PLM, Oracle PLM Cloud, Teamcenter, SAP PLM, SAP Hybris, SAP C/4HANA, Enovia, Proficy, Honeywell OptiVision, Salesforce Sales, Salesforce Marketing, Salesforce CPQ, Salesforce Service, Oracle Engagement Cloud, Oracle Sales Cloud, Oracle CPQ Cloud, Oracle Service Cloud, Oracle Marketing Cloud, Microsoft Dynamics CRM

PLM, MES &  
CRM

Oracle HCM Cloud, SAP SuccessFactors, Workday, ICON, SAP APO and IBP, Oracle Taleo, Oracle Demantra, Oracle ASCP, Steelwedge

HCM & Supply  
Chain Planning

Oracle Primavera, Oracle Unifier, SAP PM, Procore, Ecosys, Oracle EAM Cloud, Oracle Maintenance Cloud, JD Edwards EAM, IBM Maximo

Project Management  
& EAM

OneDrive, Box, SharePoint, File Transfer Protocol (FTP), Oracle Webcenter, Amazon S3

Enterprise Storage  
Systems

HIVE, Apache Impala, Apache Hbase, Snowflake, mongoDB, Elasticsearch, SAP HANA, Hadoop, Teradata, Oracle Database, Redshift, BigQuery

Big Data

mangoDB, Solr, CouchDB, Elasticsearch

No SQL Databases

PostgreSQL, Oracle Database, SAP HANA, SYBASE, DB2, SQL Server, MySQL, memsql

Databases

IBM MQ, Active MQ

Message Broker

Java, .Net, Oracle PaaS, Force.com, IBM, ChainSys Platform

Development  
Platform

# One Platform for your

**End to End** Data Management needs



Data Migration  
Data Reconciliation  
Data Integration



Data Quality Management  
Data Governance  
Analytical MDM



Data Analytics  
Data Catalog  
Data Security & Compliance

[www.chainsys.com](http://www.chainsys.com)