ChainSys

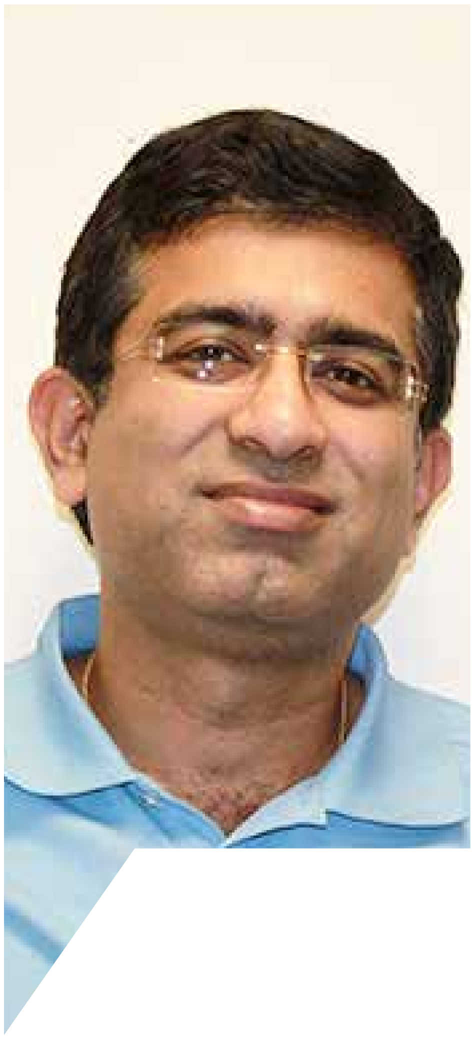
Smart Data Platform™

Technical Architecture

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.

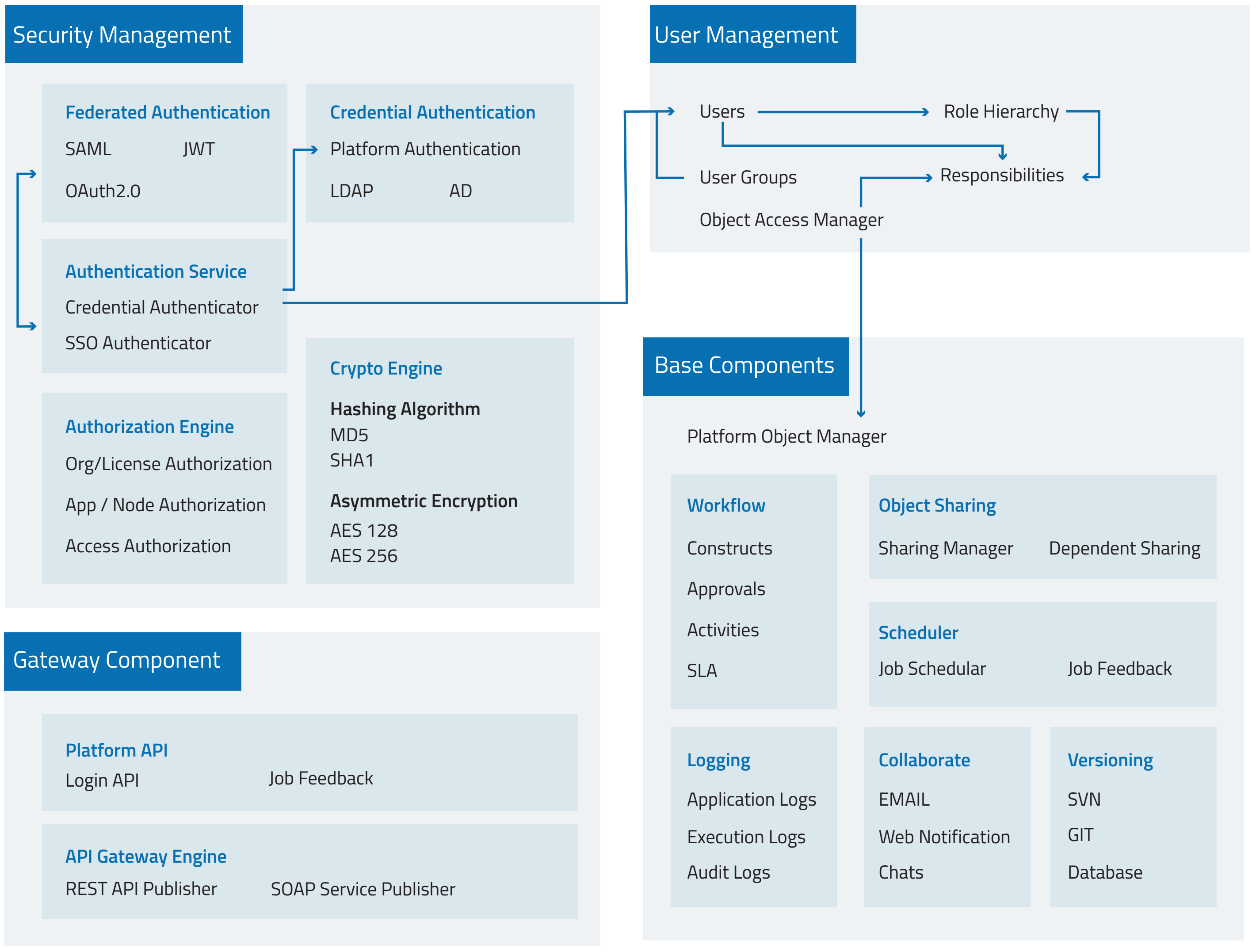
Platform Component Definition

Foundation	Smart data platform			Smart Business Platform	
	dataZap™	dataZen™	dataZense™	Smart BOTS™	Smart App Builder™
Security Authentication / Authorization / Crypto **User Management** User / Groups Roles / Responsibility Access Manager **Base Component** Workflow Versioning Notification Logging Scheduler Object Manager **Gateway Component** API Gateway	• Data Migration • Setup Migration • Test data Prep • Big Data Ingestion • Data Archival • Data Reconciliation • Data Integration	• Data Quality Management • Master Data Governance • Analytical MDM (Customer 360, Supplier360, Product 360)	• Data Masking • Data Compliance (PII, GDPR, CCPA, OIOO) • Data Cataloging Data Analytics Data Visualizat	• Used for Autonomous Regression Testing • Used for Load and Performance Testing	• Rapid Application Develiopment (RAD) Framework • Visual Development Approach • Drag & Drop Design Tools • Functional Components into Visual Workflow

dataZap™

Platform Foundation

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



Smart Data Platform™

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.

dataZap™

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.tv
- 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.

API Gateway Engine

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.

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

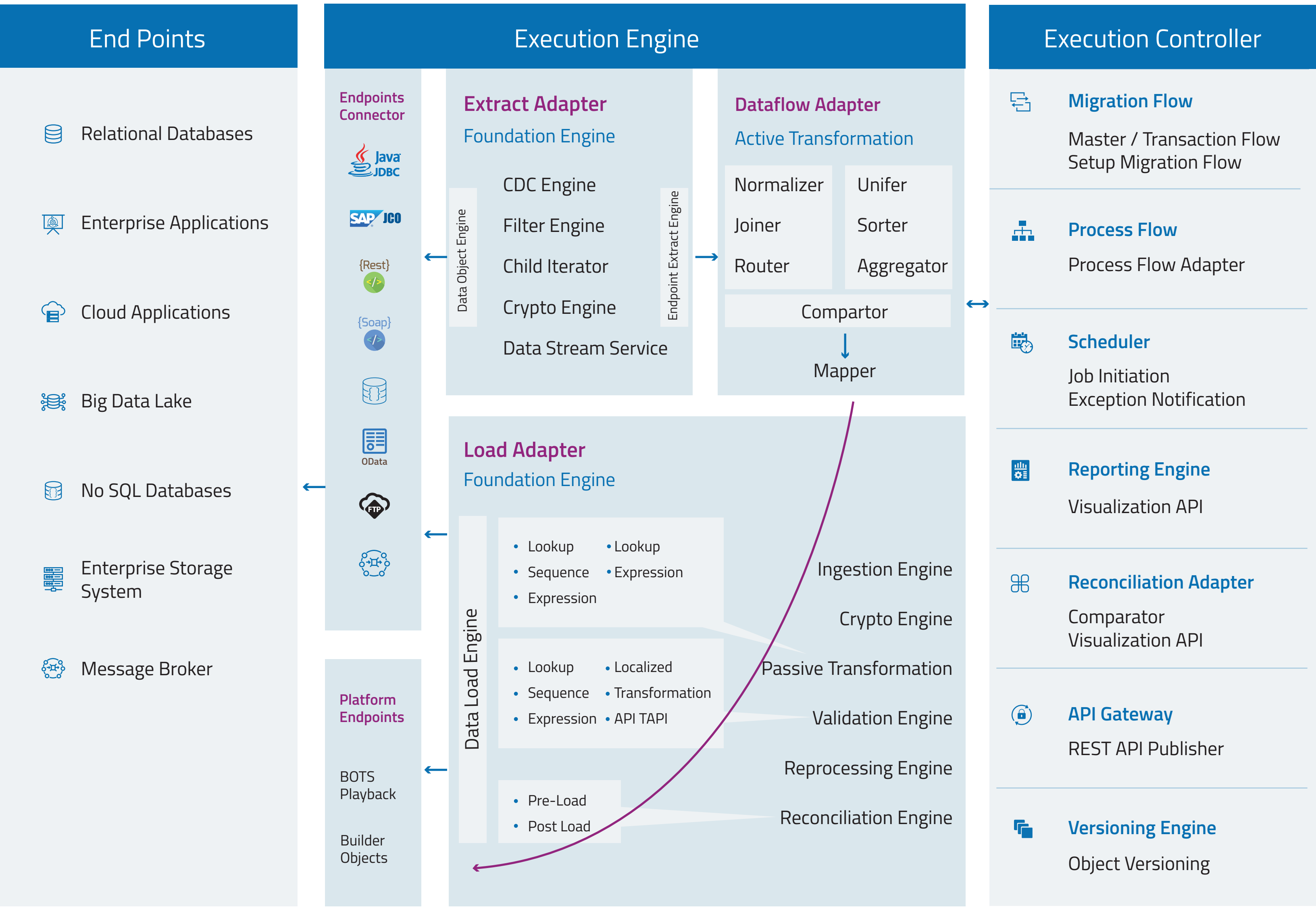
The eligible jobs or actions can be published using the Simple Object Access Protocol (SOAP). SOAP is a messaging protocol that allows programs that run on disparate operating systems to communicate using Hypertext Transfer Protocol (HTTP) and its Extensible Markup Language (XML).

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.

dataZap Component

The Execution Handler will be available on the client-side and at the cloud to handle the pure cloud environment and manipulate data in the cloud for less Load at the client end. The Execution controller will be available in the cloud to direct the execution handler in every step.



Endpoint Connectors

The component has all the base connectors used to connect to most of the endpoint applications. The base connectors would include

- JDBC - For all the RDBMS connections
- SAP JCo - For connecting to the SAP Systems
- SOAP - Connects to applications enabled with SOAP APIs.
- REST - Connects to applications enabled with REST APIs
- OData - Connects to applications enables with OData APIs
- FTP - To connect and extract data from files in the FTP sites.
- NoSQL - To connect to databases with NoSQL like Mongo
- Message Broker - To connect to different messaging services like ActiveMQ, IBM MQ

The connections can be made secure based on the endpoint configuration by Secured Layer through all of the above base connectors.

The specific connectors for Enterprise applications are wrappers built over these base connectors with specific Security and governance applied as per the application needs. The diagram shows a few of the existing wrapper endpoints created for the enterprise applications in the market.

We can use the base connectors for applications that do not have specific connectors if they do not have any particular authentication methods other than the base level authentications provided. ChainSys would build the applications specific connectors if not already exists.

Load Adapter

The component that handles the loading of data into multiple systems or endpoints.

Ingestion Engine

Initial Data marting happens in this Engine and then manipulate the data.

Crypto Engine

The Crypto Engine enables us to encrypt the data during the data marting process to ensure Data is protected in all formats. It also has the decryption to be applied before loading the data to the final target endpoint.

Transformation

Passive transformation

This transformation just changes the values of the columns from one form to another.
The different transformations types like lookup transformation, sequence transformation, and expression-based transformation can be performed.

TAPI

TAPI helps create reusable transformations (API) in multiple objects to make changes in one place rather than in numerous areas.

Reconciliation Engine

The reconciliation engine handles the technical reconciliation of the data in two stages.
The reconciliation is done at the end of the pre-validation stage to determine the differences between the raw data and the transformed data and further after the loading complete to decide the differences between the loaded and the raw or changed data.

Reprocessing Engine

This component helps to correct the errored data both at the pre-validation level and the post-load level. Data fix can be handled both online and offline. Users will be able to download the error data as an excel and upload the corrected Data as a bulk update process. In addition to the data error correction, we can also enhance or construct data that will pass through the validation step for quality.

Loading Engine

The loading engine is where the application understands the endpoint type and uses one of the loading engines to load the data into the target application. The loading engine also has special adapters to use the Playback Adapters of the Smart BOTS and Smart App Builder in the business platform.

The extract adapter component retrieves data from multiple different types of endpoints and processes the data to give the data in the expected format.

Extract Adapter

Data Object Engine

This Engine handles almost all different kinds of systems and formats to retrieve data. It can work with SQL / Flat files / SOAP and REST service.

Filters

This is to reduce the number of rows from the raw data extracted from the source.

Changed Data Capture

This is the component that gets the changed data from the source. Two modes can achieve this

- The recommended option is by assigning the date field to be used for bringing the changed data.
- There is also an option to bring in data by comparing the records and is supposed to be resource-intensive and is not recommended until there is no date field to compare.

Child Iterator

This component handles the master child relationship between the data extracts so that the filter applied on the master can get down to all the child levels.

Crypto Engine

The Crypto Engine is to read or extract the data with encryption applied over the selected fields for extraction. This helps in encrypting the data from being accessed from the front end or the back end.

Data Streaming Service

Here the data extracted from the data object or the extract adapter are streamed to the applications calling the service to pull the data from the endpoint.

dataZap™

Dataflow Adapter

The dataflow adapter helps in transforming and mapping the data from multiple sources to multiple target systems.

Active transformation

This transformation where the number of rows is getting affected. The possible active transformations available are the Normalizer, Joiner, Router, Unifier, Aggregator, and Sorter. The active transformation engines convert the data structures from the source to the target.

It also has the rules engine (Router) to move the data as per the rules to different endpoints.

It also can compare the data between the two systems and determine action before moving the data.

Migration Flow

This component overrides the workflow component in the foundation. The migration flow engine is specific to the migrating Master or Transaction Data. The feature has orchestration capabilities and human intervention capabilities like Approval, User Confirmation, and Receive Input.

Process flow Engine

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

Scheduler

These components provide the job's execution agents specific to dataZap that needs to be executed by the base scheduling engine. These are wrappers for the data movement components like Load Adapters, Extract Adapters, Dataflow Adapters, and Process Flow.

API Gateway

These are execution agents for the publisher in the foundation. These form the wrapper to the jobs that need to be executed in the data movement components like Load Adapters, Extract Adapters, Dataflow Adapters, and Process Flow.

API Gateway

These are execution agents for the publisher in the foundation. These form the wrapper to the jobs that need to be executed in the data movement components like Load Adapters, Extract Adapters, Dataflow Adapters, and Process Flow.

Reconciliation Adapter

The reconciliation adapter generates the query to compare the data and produce the Visualization API result to create the necessary reconciliation dashboard.

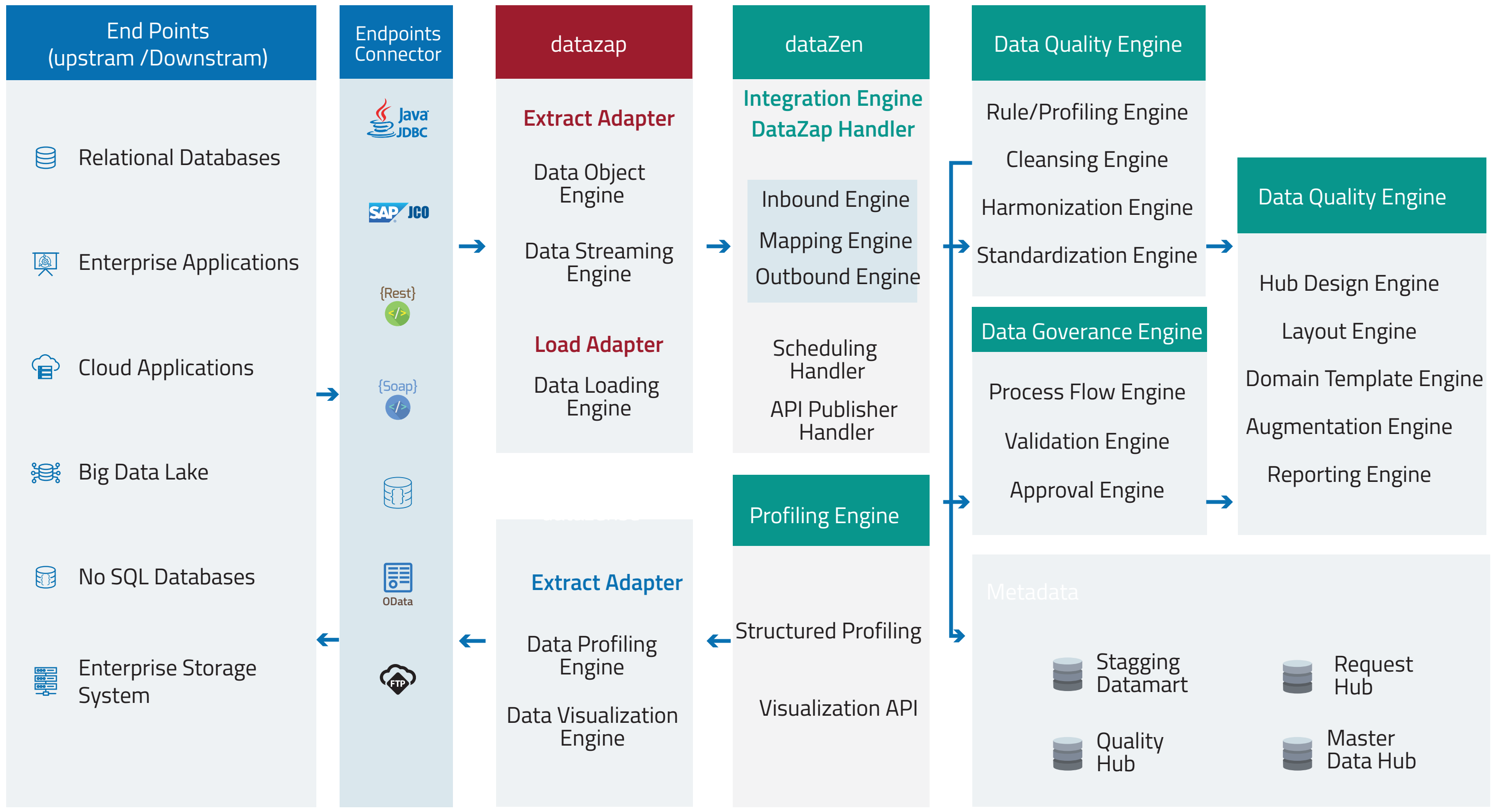
Reporting Engine

The reporting engine generates reports on the various adapters' execution and produces dashboards to understand the actions taken and to be taken.

dataZap™

dataZen Component

dataZen is a Multi-Domain
Master data management platform.
We offer a unique templated approach to
MDM to give you the flexibility
and innovation.



dataZen™

This is the component that holds the requested, validated, and approved master data.

Master Data Hub

Data Hub Engine

This Engine creates the data model to be used for data management activities like data cleansing and governance. The request for new or update of records would be in the Request hub. All the cleansing activities will happen in the intermediate hub, and all the validated data would be moved on to the Master Hub.

Data Hub Engine

The component comes with pre-built data model adapters for multiple enterprise applications and verticals. This Engine helps to build the data hub model using the pre-built model adapters.

Layout Engine

This Engine helps in creating the page layout for the built data hub model. There can be multiple layouts for the same data model as per the User's requirement.

Augmentation Engine

This Engine creates rules for validating data in the pages built for the data hub to help data governance.

Reporting Engine

This Engine uses the visualization layer to create dashboards for the rule execution and other reports on the data hub.

Smart Data Platform™

Data Quality

The component is to hold all the requests for manipulating the master data. This hub contains all the approved and unapproved requests.

Rule / Profile Engine

This component helps in identifying the corrupt, inaccurate, or irrelevant data in the existing systems. The profiling engine helps identify the data problems on a higher level and creates rules to identify data issues to the depth.

Cleansing Engine

This component helps resolve the corrupt, inaccurate, or irrelevant data before moving it to the master hub. The transformation rules can be created to handle data errors and automate data correction.

Harmonization Engine

This component has the data deduplication engine using multiple algorithms like Fuzzy, Edit Distance, and NLP. The Engine would also create cross-references for helping on further cleansing of transactional data.

Standardization Engine

This component helps standardize using in-built standards and external services for attributes like address and other possible attributes.

dataZen™

This component handles all the
Data Quality management.
It internally uses the BRE engine to
validate and cleanse the data.

Data Governance

Process Flow Engine

This Engine helps in orchestrating the flow of data from the Data being requested through to the master data hub with all validations and approval.

Validation / Enrichment Engine

This Engine validates the data and prevents the data from moving to the master data hub.
This Engine uses all the components of the data quality engine to validate and enrich the data with the defined rules.

Approval Engine

This Engine helps in the approval process of the requested data. The approval workflow can be well-orchestrated by parallel approval, hierarchical approval, and rule-based user approval.

Profiling Engine

The profiling engine helps to profile the Data for the data architects to understand the current data anomalies and form the required rules and checks for clean data. The Engine also has the visualization APIs to represent the data in a nice visual layer.

Interfacing Engine

This component is used to interface the data from master hub to the downstream systems and interface the upstream systems to the request-hub. This Engine creates the following Data Movement Adapters to move data from the Upstream Systems to the data hub and from the hub to the downstream systems.

Data Extract Adapters for Upstream systems and Data Hub

These adapters bring in bulk or incremental data to the request-hub to further process the data based on the configurations on the data quality engine or the data governance engine. This can connect to any number of endpoints defined in the data movement engine.

Data loader adapters for the downstream systems

These adapters help move the clean data into the other downstream systems. Again, the adapters can transfer data to all the endpoints defined in the data movement engine.

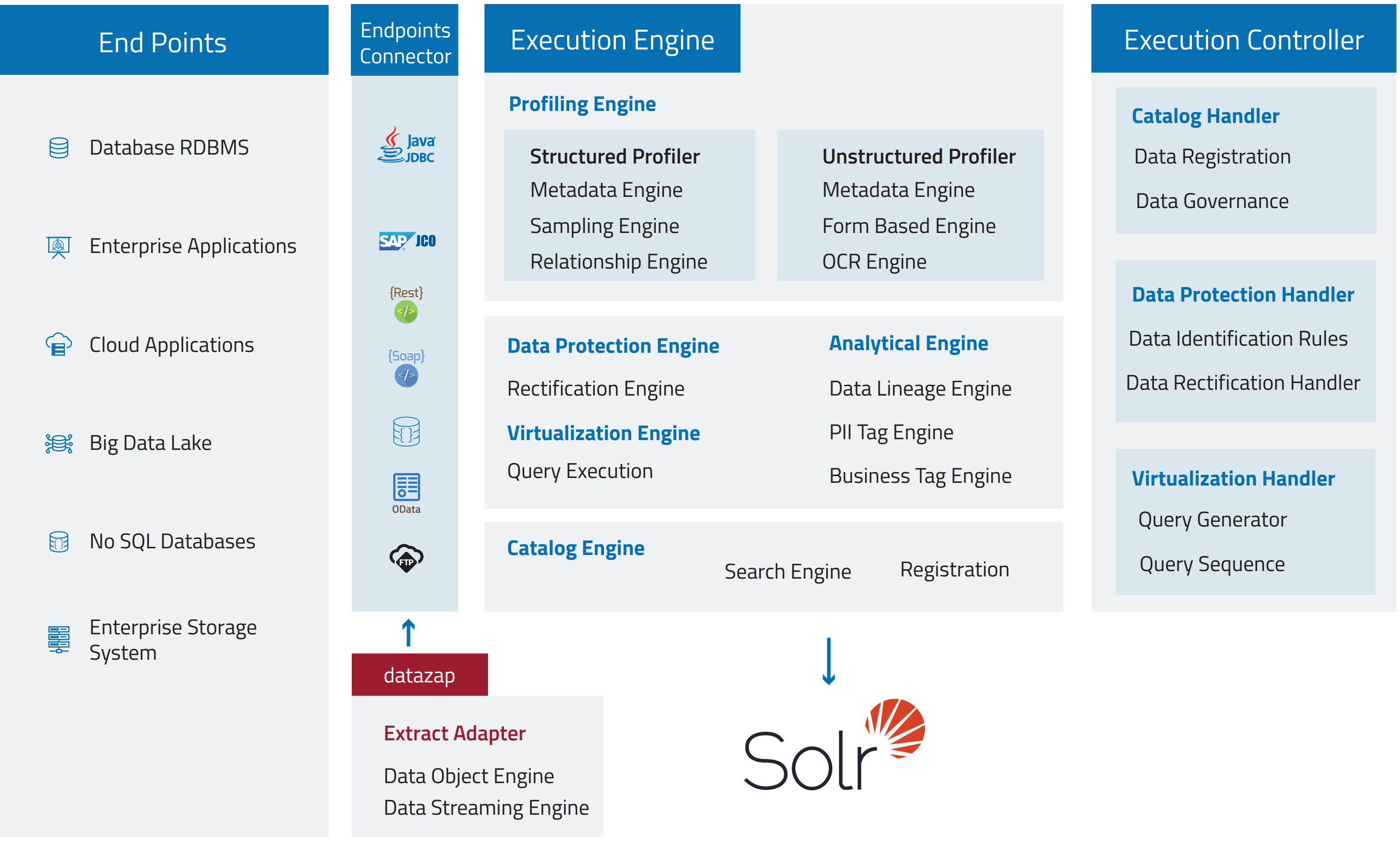
Data flow Engine to map from hub to the downstream systems

These adapters help map and move the data from the master data hub into the downstream systems using the data movement layer's help. The mapping between the hub and the systems are predefined when using the domain objects' adapters.

Smart Data Platform™

dataZense - Data Catalog Component

dataZense offers full trust to your enterprise data. Find below the components of dataZense.



dataZense™

Profiling Engine

This component handles the profiling of the systems. It can address the profiling of both structured and unstructured data sources.

Structured Profiling

In the structure data sources, it can handle all the endpoints supported by the Platform. We can not only profile relational databases and Big data sources it can also handle but also the cloud data sources using the data connectors we produce.

Technical Metadata Engine

This Engine catalogs all the metadata information of the source. These are automated to have most of the metadata information. The metadata attributes are configurable to add more business terms as needed by the organization. The relationship between the entities is also suggested. The data citizens would be able to approve and comment on all the suggestions put forward by the Engine.

Unstructured Profiling

In the unstructured data profiling, we would be able to read and process all the text-based data files from the different connectors that have been provided. The unstructured profile would identify Tet Content, Table Formats, form-based data, and images. We will be able to create a rule-based reading algorithm to categorize and identify data.

Cataloging Engine

This component creates a catalog of all the profiled Data and helps create a whole meaning and purpose.

Advanced Search Engine

The search engine provided searches for the Metadata from the catalog and searched all the data indexed across all sources to give a unified search engine. The search results would also pick the data from the respective sources to look at the data.

Data Lineage

Data lineage traces your Data's origins, movements, and joins to provide insight into its quality. The data lineage also considers the Data that are indexed to give more viable suggestions for the lineage. The data lineage also offers options to indicate the forward and backward propagation of the lineage pipeline's data.

Data Security

This Engine helps make sure that row level and column-level Security ensure only relevant data are discovered in the catalog. It provides granular access management controls and supports secure and role-based access levels for individuals like the data engineer, data steward, data analyst, and scientist. Access notifications are provided to managers with the complete picture and approval of the request to view the data.

Data Citizenship

There is a high level of flexibility for the data citizens to view the Metadata and comment and approve on all of the results of the above Engine. This Engine also helps to build a data visualization for all the cataloged data.

dataZense™

dataZense protects your personal data breach by identifying and remediating the PII data. It supports both structured and unstructured data.

Data Protect Engine

Personal Identifier Linked Engine

This Engine uses the profiling engine results to determine any piece of personal information that can be used to identify sensitive data like Full name, Home address, Phone Number, Email address, National Identification Number, GovernmentID's, Credit Card Number, etc.

Personal Identifier Linkable Engine

This Engine uses the profiling results to determine the information that may not identify a person on its own. When combined with another piece of information, it could locate, trace, or locate a person—some examples like Gender, Ethnicity, Job Position, etc.

Identification Rule Engine

This Engine helps create rules to identify the PII data not given in the standard list of rules from the application. You would also be able to modify the default rules to the needs as per the organizational requirements.

Data Lake or Virtualization

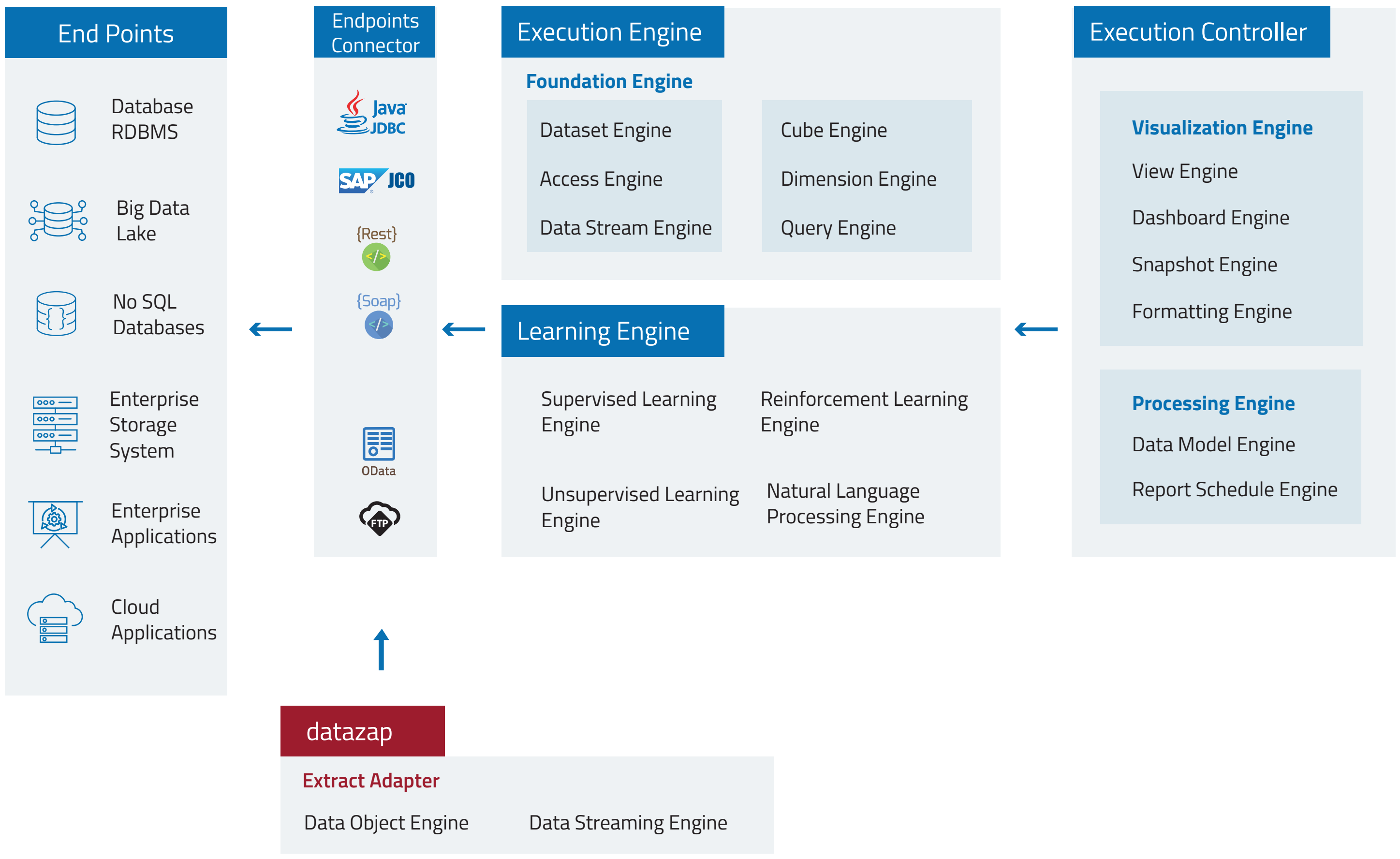
This is the data movement layer of the Platform. Here data from multiple different sources with different structures are brought into a data lake or virtualization environment. This can be used to catalog the data for creating lineages and provision the Data for users.

Data Provisioning

This is where the Data is being provisioned to the users to view the data in a unified manner across all the data sources. The application building layer in the Platform is used to have this enabled for the users.

dataZense - Analytics & Visualization Component

The Execution Engine will be available on the client-side and at the cloud to handle the pure cloud environment and manipulate data in the cloud for less Load at the client end. The Execution controller will be available in the cloud to direct the execution handler in every step.



dataZense™

This is the foundation component of all the activities and engines in the dataZense execution Engine.

Foundation Engine

Dataset Engine

This Engine executes the data model defined that would be required for the analyticsof the data. This is the data holder for the reports to convert into the individual cubes and dimensions. Data can be directly fetched from the endpoints using the endpoint connector, or for complicated API's data would be fetched through the extract adapter in the dataZen application.

Access / Filter Engine

This Engine creates filters based on users or views. This can be used to filter records needed for a particular Chart and can be used to filter data to be processed based on the User's access or privileges set in the dataset model.

Data Streaming Engine

This Engine streams the manipulated data to the controller to produce the visual effect on the data.

Analytics Engine

This is the Engine that does all the analytics on the dataset to produce the results. The main components of this Engine are the

Cube Engine

This Engine generates the cubes from the data in the dataset to run the analytics on the data based on the dimensions and KPI's configured in the dataset.

Query Engine

Generates query on the cube to fetch the needed report from the cube. The Data is streamed to the controller to form the visual effects on the data.

Visualization Engine

This is the foundation component of all the activities and engines in the dataZenseexecution Engine.

View Engine

Multiple chart types can be created in the visualization component. The User can change the compatible charts based on the dimensions and KPI's in the runtime. The data used for the views can be downloaded in CSV or Excel format.

Dashboard Engine

These multiple views are assigned and arranged for a dashboard. We can bring in various views from various datasets into a dashboard. There are options to drill down based on the filters created from one filter to impact or not to impact the other dataset views. The dashboard can be made fluid to change the views at runtime as per the User's choice.

Snapshot Engine

The snapshot engine generates the snapshot of a dashboard, not just which is in the browser's view but the entire dashboard even if not visible.

Formatting Engine

This component helps to create and handle the formatting of the fonts and colors of the charts. This also addresses the conditional formatting based on the data.

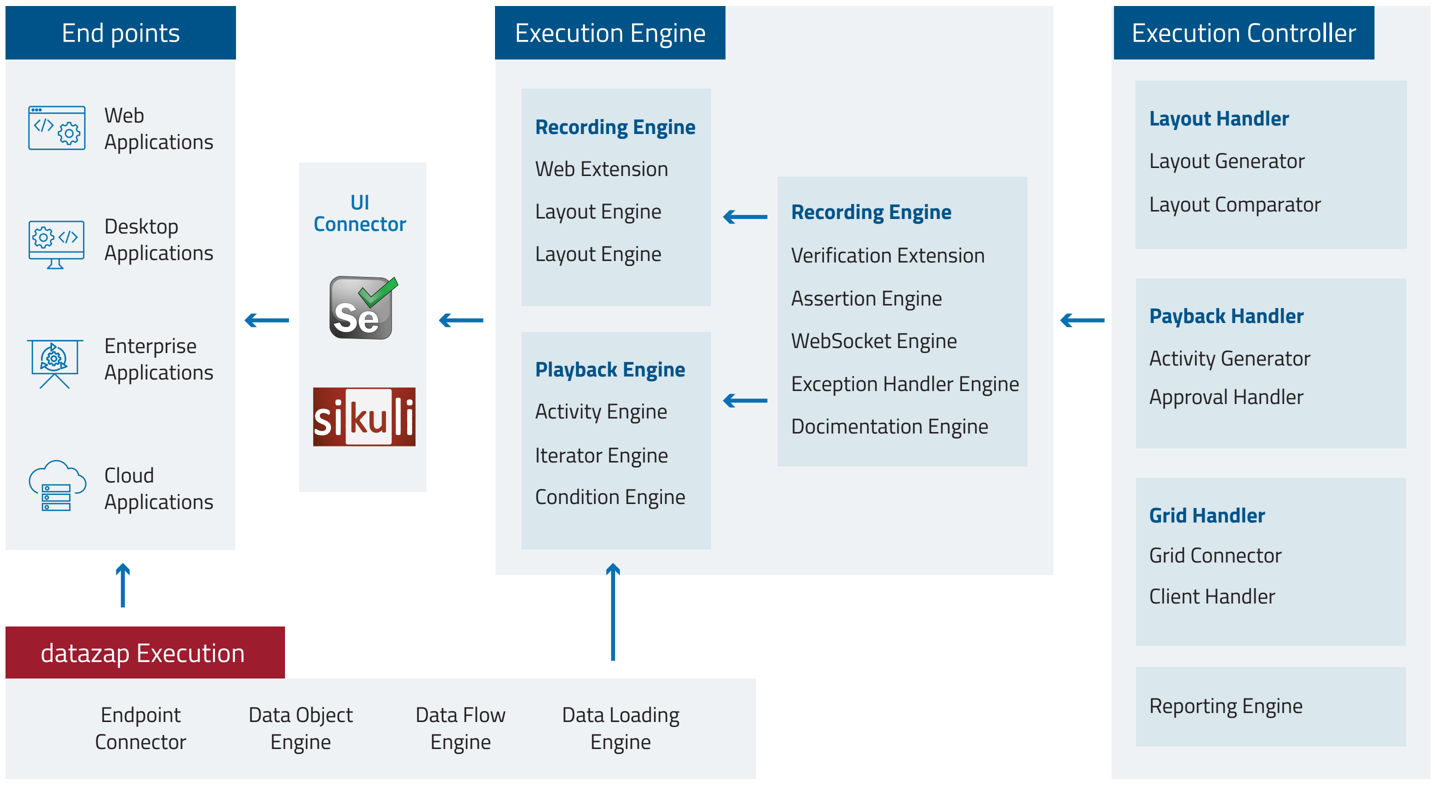
Processing Engine

Report Scheduling Engine

This Engine processes the data and generates the reports/dashboards at a scheduled time. These dashboards and reports can also be sent as attachments to emails.

Smart BOTS

The Execution Handler will be available on the client-side and at the cloud to handle the pure cloud environment and manipulate data in the cloud for less Load at the client end. The Execution controller will be available in the cloud to direct the execution handler in every step.



Smart BOTS™

UI Connector

The UI connector helps in connecting the application frontend to run the automation process. The connectors are used to access the pages in the Webbrowser or as a desktop application. The connectors would be called to record the playbacks of the pages and execute the automation process.

Recording Engine

The Engine helps to record the multiple activities performed on a page or a flow of pages.
The major components in this are

Web Extension

The component has a web extension available that can be installed to your browser, where the web-based recordings can be performed.

Layout Engine

The component generates the layouts for the pages based on the activities performed. The layouts are used to define the components in each page and its characteristics. These layouts would help us to quickly modify a component to reflect in the multiple playbacks for the same page with different scenarios.

Activity Engine

This is the Engine that creates a playback with the activities sequenced as per the recording.
The completed activities can be overridden if necessary.

Playback Engine

This Engine is used to execute the playbacks with the static or dynamic data produced by the handler. The major components are

Activity Engine

Here the Engine executes the playback based on the sequences that were recorded. The data to this Engine can be statically assigned during the recording or supplied by the handler from sources like Excel or CSV. The other option would be when the dataZap invokes the playback using the loader.

Iterator Engine

This is the data iterator for supplying data to the playback activities. This is an extension of the iterator component in the workflow.

Condition Engine

The condition engine handles the condition based routing of the activities. This is an extension of the base workflow component.

Smart BOTS™

This Engine contains the base activities that would be required to process any kind of automation.

Foundation Engine

Verification Engine

This activity verifies the activity&#39;s success based on the conditions and calls the exception handler and document engine to record the action if the state fails. Option to document all the activities despite the status can also be achievable. In case of failure, the document is handled and proceeded to the next step.

Assertion Engine

This activity is very similar to the verification activity. The only difference is that the playback would not move the next set of actions and stop and exit the playback.

Websocket Engine

This Engine handles the requests that come from the server (Execution controller) to initiate playback.

Exception Handler

This component helps in determining the action in case of an unexpected exception during the execution. This can trigger another playback or email notification, or any other activity can be handled.

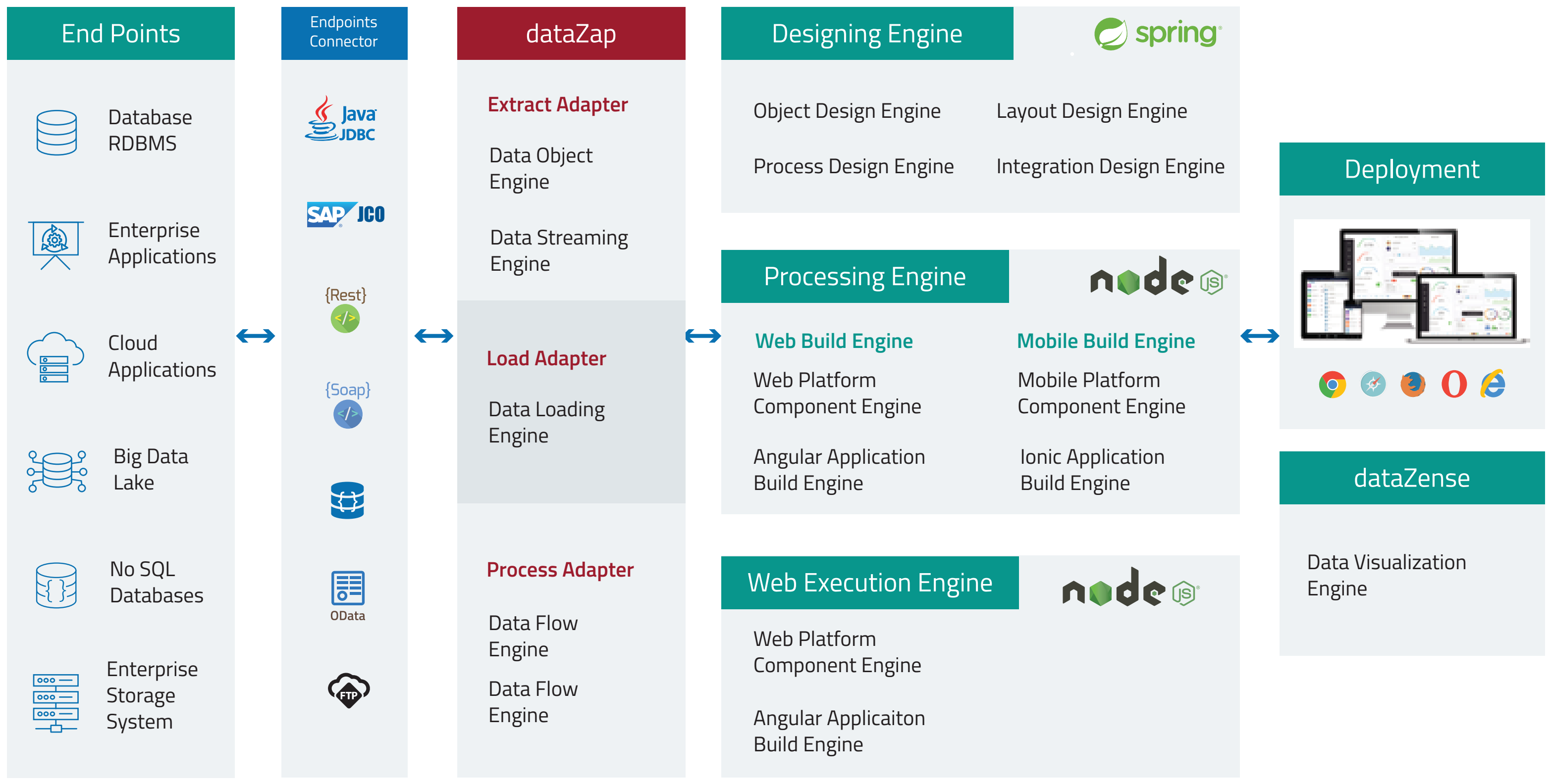
Documentation Engine

This Engine handles the creation of documents during the execution of a playback. As discussed, documentation can be configured to capture all activities or just the validation failure and exceptions during the performance.

Grid Handler

This component is to create grids of clients to run multiple sessions on the same or different laybacks. The grids can run with the same profile or other profiles that can be handled by the client handler. This is used to run the load test on applications that allow such activities to be performed.

Smart App Builder



Smart App Builder™

Object Design Engine

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 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.

Smart App Builder™

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

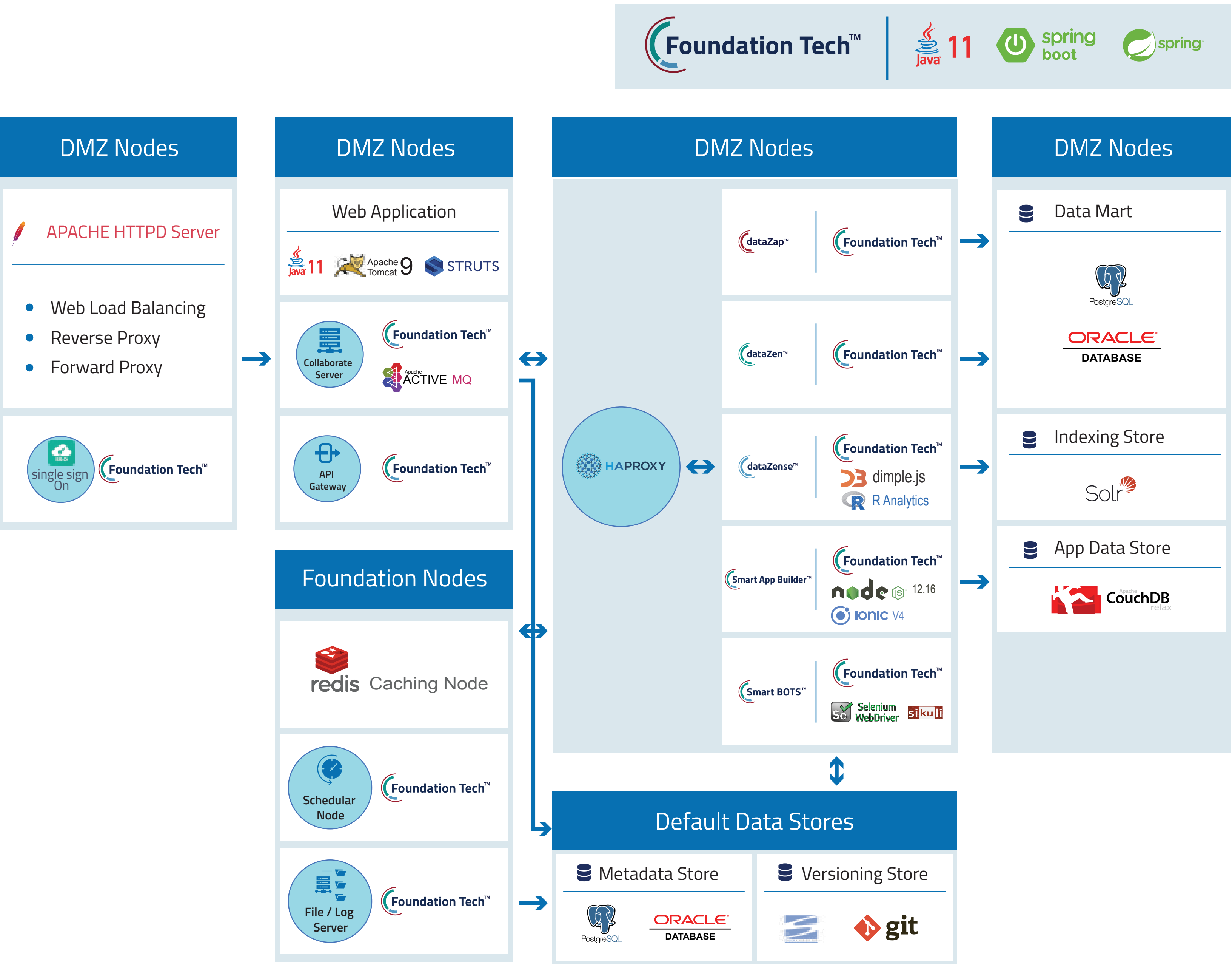
The web execution engine is the NodeJs environment. It handles all the web and mobile application requests.

Deployment

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

System Technology Landscape

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.



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 throughfrontend or a third-party application asAPI's. The nodes available in this layer would be theWeb 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 theother 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.

dataZap™ Node

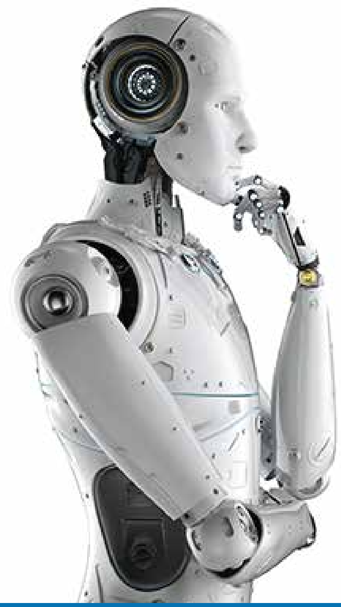
The application uses only the default services that are mentioned above.

dataZen™ Node

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

dataZense™ Node
(Analytical Services / Catalog Services)

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.



Smart BOTS™

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.

Smart App Builder™

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&#39; 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

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 s ensitive data.

- 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

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

Smart Data Platform™

File Log Server

This component is used for centralized logging, which handles the application logs, execution logs, and error logs in the platform applications&#39; 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.

dataZap™	dataZen™	dataZense™	Smart App Builder™
Loader Adapters,	Data Model,	Data Set,	Object Model,
Data Objects,	Rules,	Views,	Layouts,
Data Extracts,	Augmentations,	Dashboards,	Workflow
Data Flows,	Workflow	Ad-hoc Reports	
Process Flows,			
Migrations Flows,			
Reconciliations			

Apache SOLR

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.

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 storeSolr's data in HDFS. Solr integrates nicely with Hadoop's authentication approaches, and Solr leverages Zookeeperto 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 PostgreSQLwill sync with CouchDB if mobile applications are enabled. In contrast, the initial ntry 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

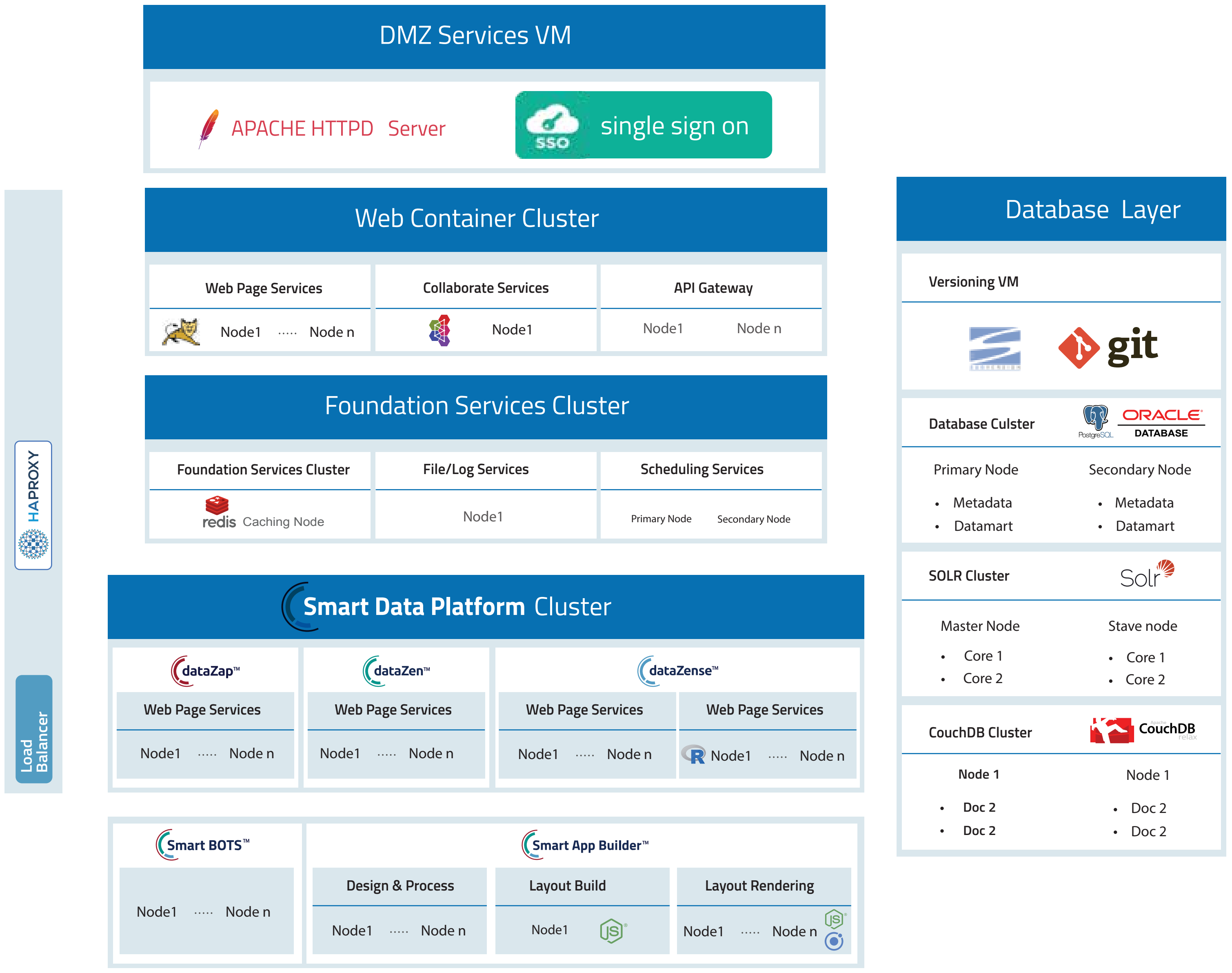
- CouchDB throws the HTTP and REST as its primary means of communication out the window to talk to the database directly from the client apps.
- The Couch Replication Protocol lets your Data flow seamlessly between server clusters to mobile phones and web browsers, enabling a compelling offline-first 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.

Smart Data Platform™

Deployment at Customer

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.



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.

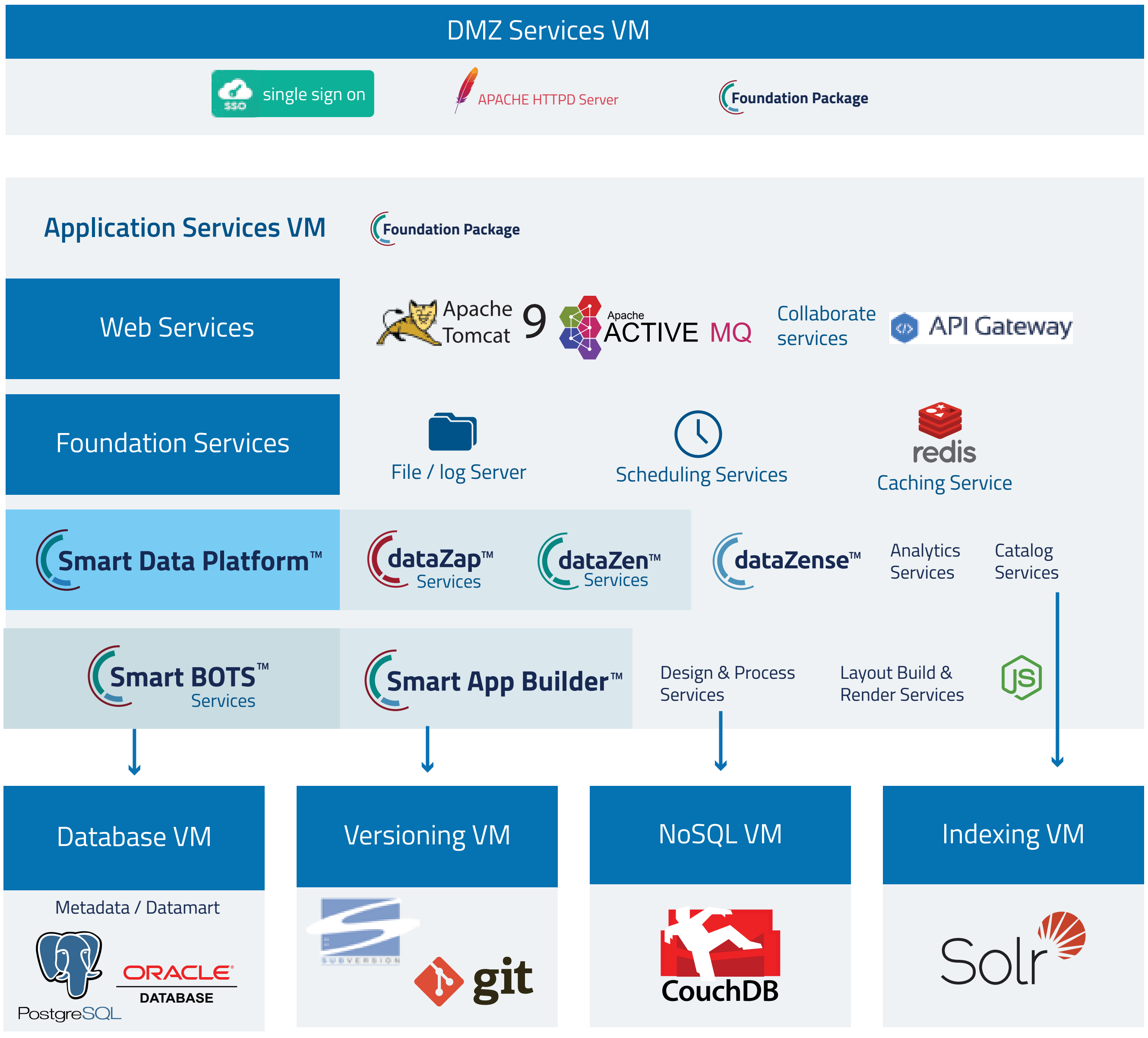
- REDIS is used as a cache engine. It is mandatory for distributed deployment. This can be configured for high availability using the master-slave replication.

- 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.

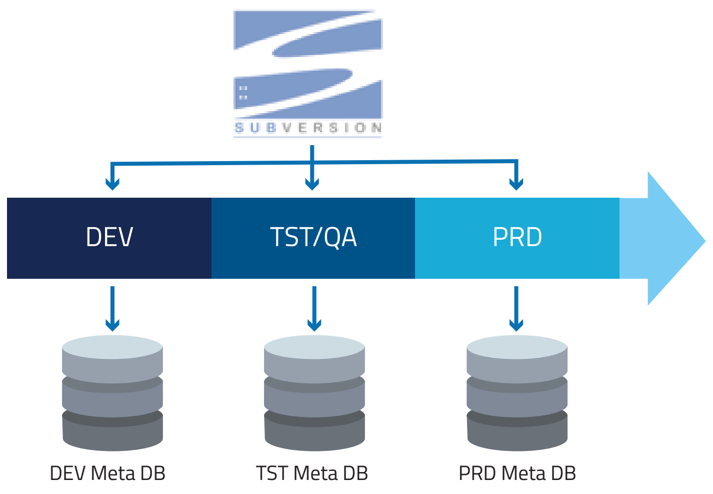
SVN server would be mandatory to store all the configuration objects in the repository for porting from one instance to the other.

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.

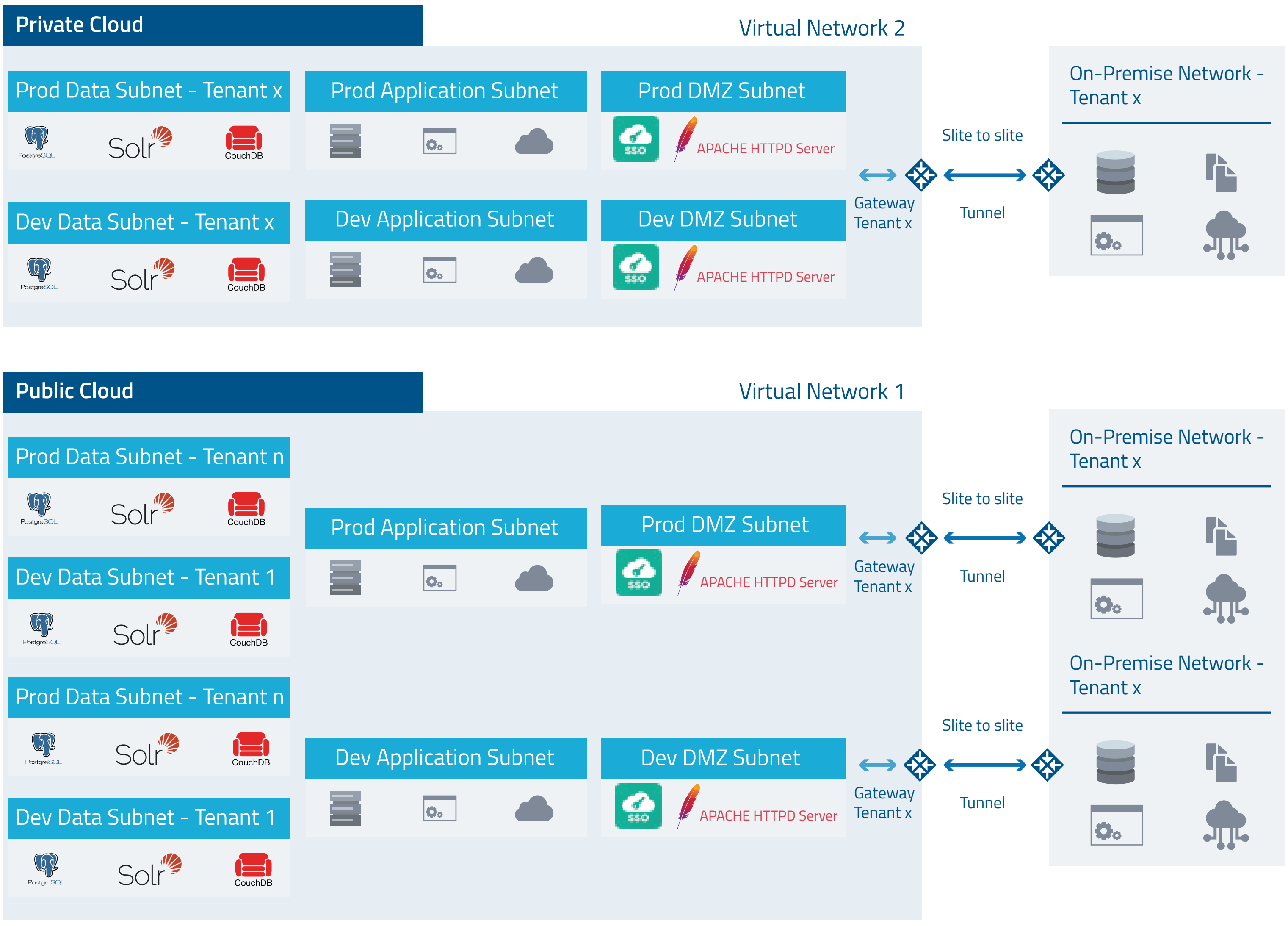


Generally, the above instance propagation strategy is recommended. Depending on the applications in use and the Load, it could be determined to go with a single node deployment or a distributed model deployment. Generally, it is recommended to have a distributed deployment for Production instances. The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

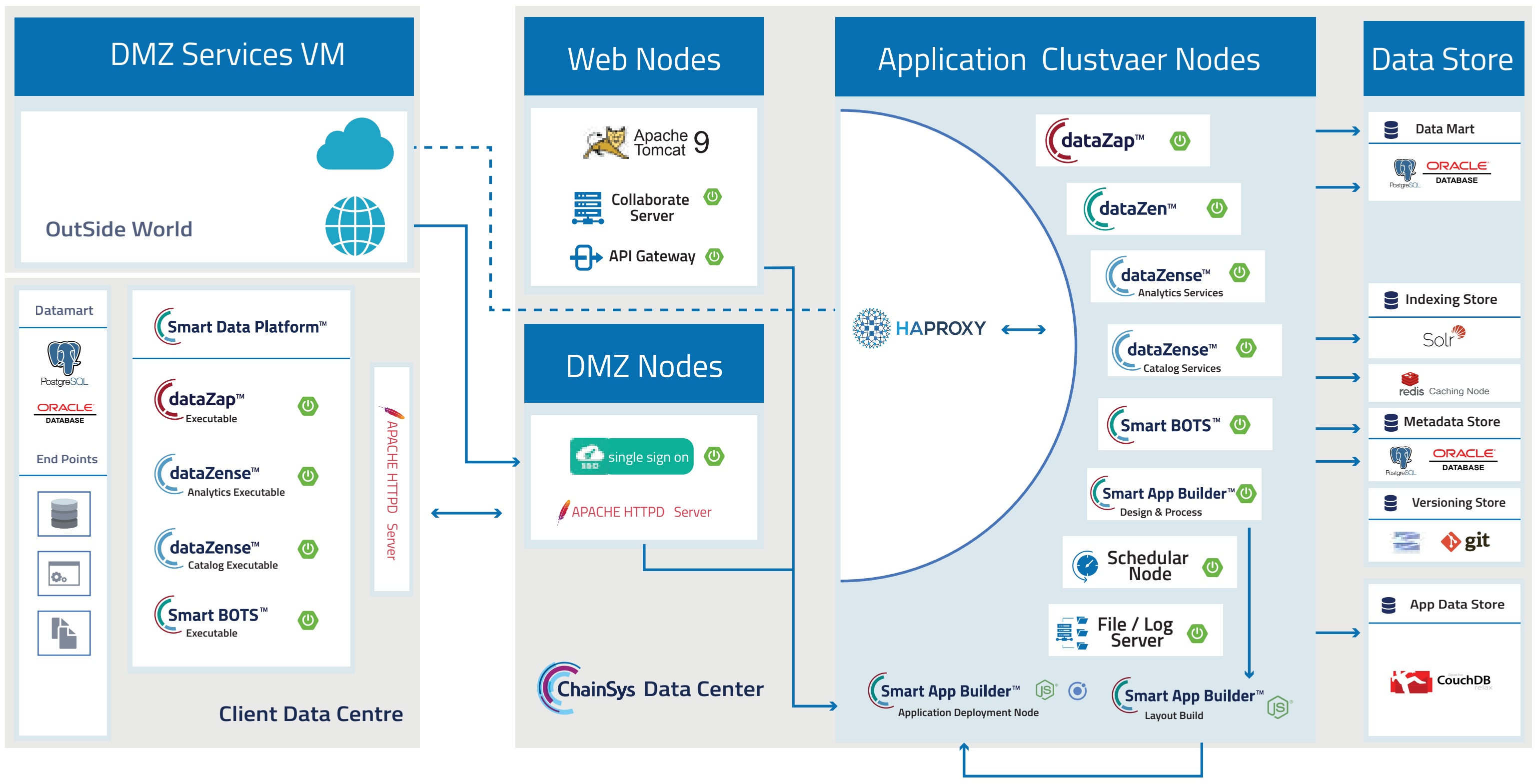
Depending on the workload on the nodes, each Node can be clustered to balance the Load.

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

Smart Data Platform™

Hybrid Cloud

Hybrid Cloud Deployment

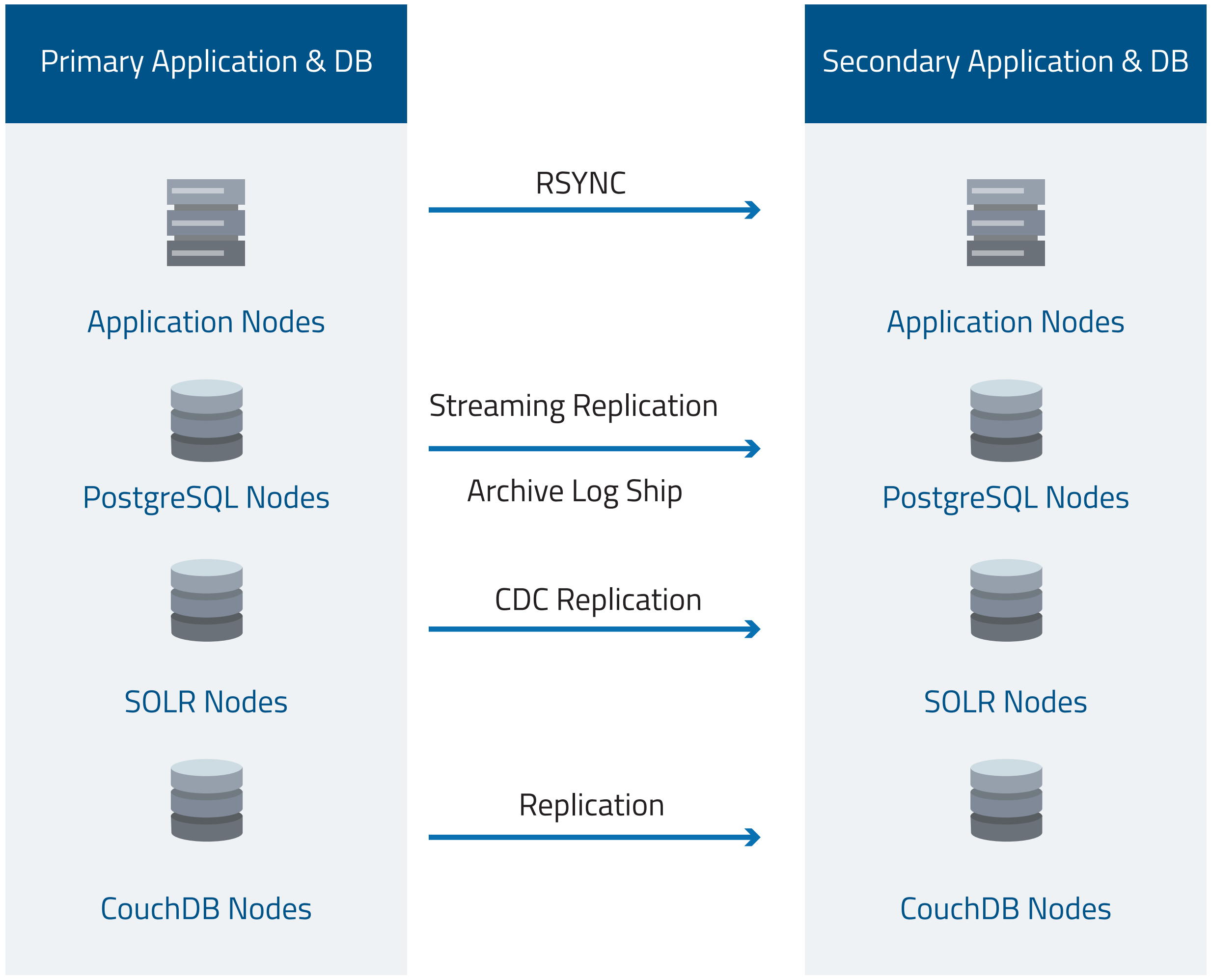


This can be associated along with both the Private or Public cloud. An Agent would be deployed in the client organization's premises or data center to access the endpoints. This would avoid creating the Site to Site tunnel between the Client Data Center and the ChainSys Cloud Data Center.

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 dataZen and Smart App Builder, data would be streamed to the Chainsys Platform to manipulate the data.

Disaster Recovery

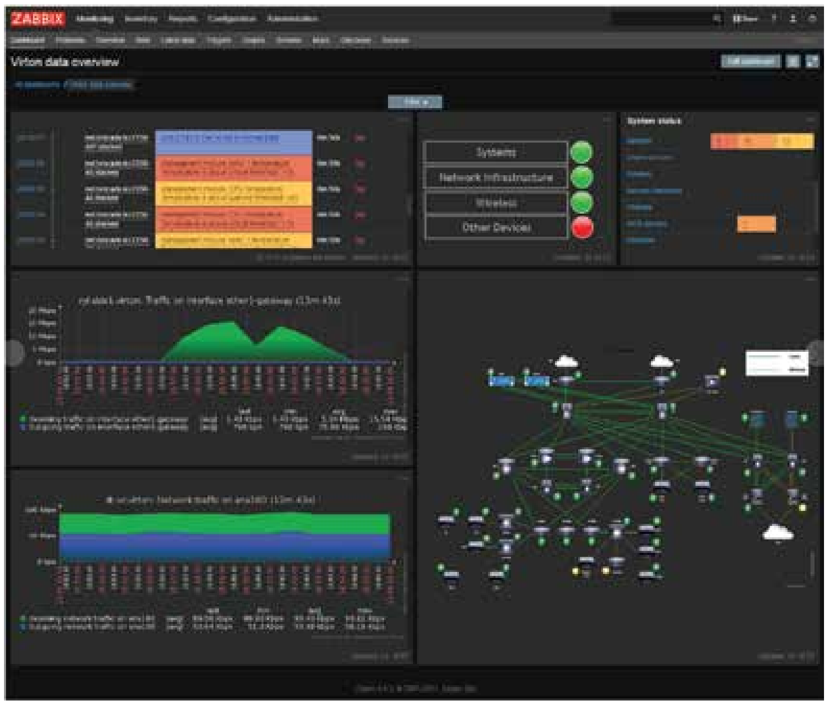


- All the application nodes and the web nodes would be replicated using the RSYNC. The specific install directory and any other log directories would be synced to the secondary replication nodes.
- For PostgreSQL, the Streaming replication feature would be used, which used the archive log shipping.
- 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 uch 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

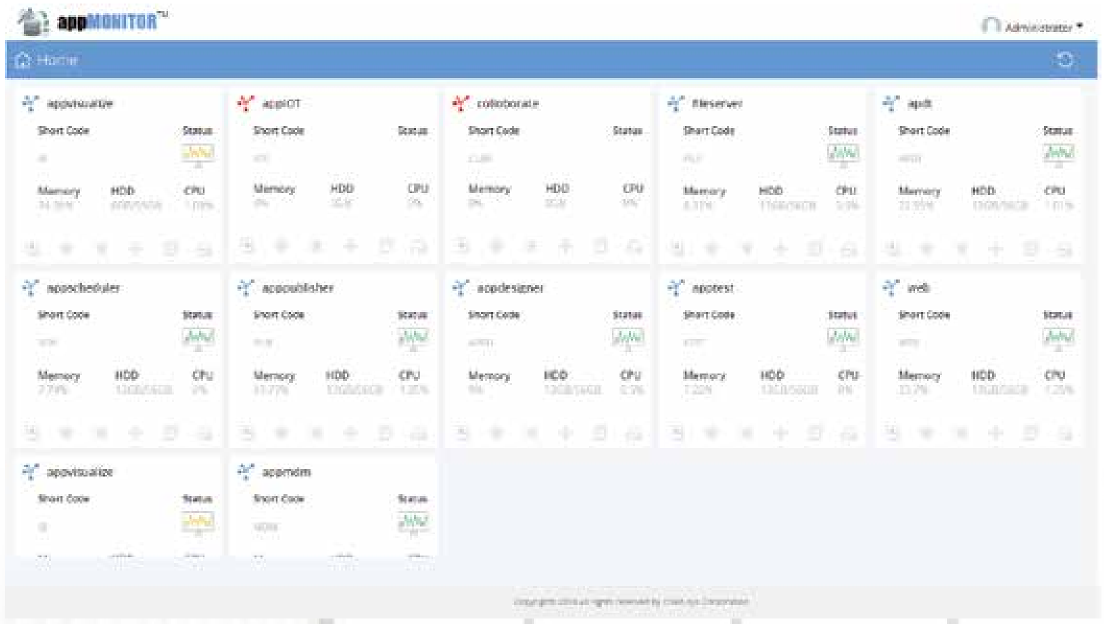


- Various data collection methods and protocols
- Start to monitor all metrics instantly by using out-of-the-box templates
- Flexible trigger expressions and Trigger dependencies
- Proactive network monitoring
- Remote command execution
- Flexible notifications
- Integration with external applications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

Smart Data Platform™

dataZap™

Data Migration
Data Reconciliation
Data Integaration

dataZen™

Data Quality Management
Data Governance
Analytical MDM

dataZense™

Data Analytics
Data Catalog
Data Security & Compliance

www.chainsys.com