Table of Contents

Introduction

The present document describes the current architecture of the ERA ecosystem, both at a logical and physical level. The architecture includes the web application that allows users to submit studies, the component responsible for data transformation, the layer responsible for storing data in various formats, the API, and finally, the layer for user and third-party system recognition and authentication. The current architecture has been designed with a focus on cloud technologies. Additionally, the document presents the CI/CD practices that are necessary for the smooth operation, maintenance, and expansion of the system.

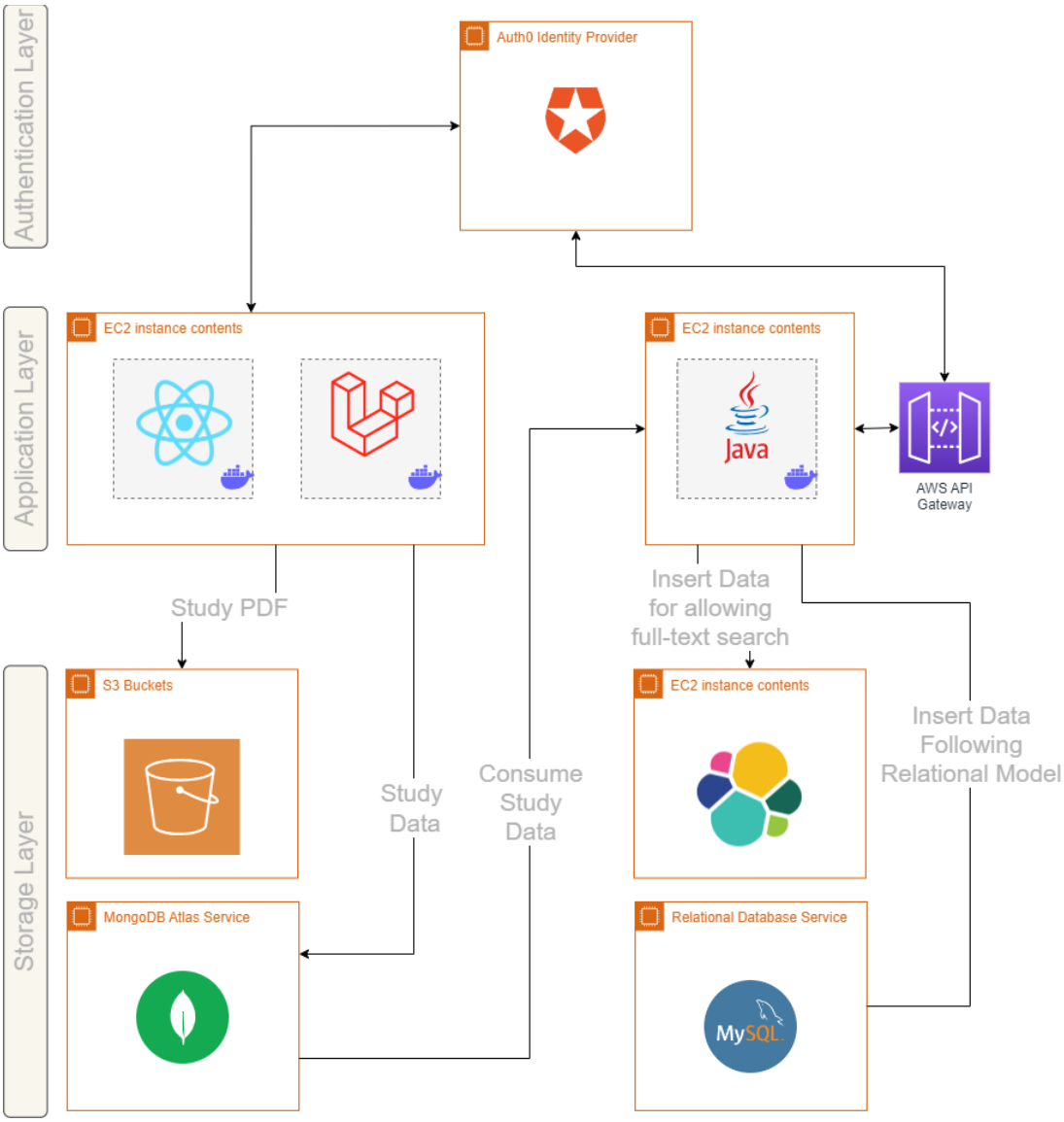
Cloud Architecture

ERA Ecosystem is designed and developed to follow cloud best practices to gain the following advantages and benefits:

1. **Scalability**: Cloud architecture allows for easy scalability, enabling the ecosystem to quickly adjust their resources based on demand. It provides the flexibility to scale up or down infrastructure, storage, and computing power as needed without significant upfront costs.
2. **Reliability and High Availability**: Cloud architecture offers built-in redundancy and high availability features. Service providers typically have multiple data centers and robust infrastructure, ensuring that applications and data are accessible even in the event of hardware failures or outages.
3. **Cost Efficiency**: Cloud architecture follows a pay-as-you-go model, allowing the ecosystem to pay only for the resources they use. This eliminates the need for upfront investments in hardware and infrastructure. Additionally, cloud services often offer cost optimization features, such as auto-scaling and resource optimization, helping minimize expenses.
4. **Flexibility and Agility**: Cloud architecture enables the ecosystem to quickly deploy, test, and iterate on applications and services. It provides developers with tools, APIs, and pre-built services that streamline the development process and enable faster time to market.
5. **Global Accessibility**: Cloud architecture allows users to access applications and data from anywhere in the world, as long as they have an internet connection. This global accessibility facilitates collaboration among teams, remote work, and the ability to serve systems or users in different geographical locations.
6. **Security and Data Protection**: Cloud service providers prioritize security and invest heavily in advanced security measures, such as data encryption, access controls, and regular security audits. They also offer data backup and disaster recovery options, ensuring data resilience and protection.
7. **Integration and Interoperability**: Cloud architecture supports integration with various systems, applications, and third-party services through APIs and standardized protocols. This facilitates interoperability and enables organizations to leverage existing systems while incorporating cloud-based services.

Moreover, several components across the different layers are based on Managed Services; Using Managed Services we gain:

1. **Reduced Operational Burden**: Managed services offload the operational burden from the ecosystem, as the service provider takes care of tasks such as infrastructure management, software updates, security patches, and system monitoring.
2. **Increased Reliability and Availability**: Managed services often come with service level agreements (SLAs) that guarantee a certain level of reliability and availability. Service providers have robust infrastructure, redundant systems, and disaster recovery mechanisms in place to ensure high uptime and minimal service disruptions. This can significantly improve the reliability and availability of critical business systems.
3. **Cost Savings**: Managed services can offer cost savings compared to building and maintaining in-house infrastructure and expertise. By opting for a managed service, organizations can avoid upfront capital expenditures, reduce hardware and software costs, and minimize ongoing maintenance expenses. They can also benefit from predictable monthly or usage-based pricing models, allowing for better budget planning and cost control.



1. ERA Ecosystem

Authentication Layer

The authentication layer is responsible for verifying and validating the identity of users or entities accessing a system or application. Its primary function is to ensure that only authorized individuals or systems can gain access to protected resources. The main tasks of the ERA authentication layer are:

1. **User Identification**: The authentication layer collects and verifies the identity of users attempting to access the system.
2. **Credential Verification**: The authentication layer validates the provided credentials against stored and centralized authentication data.

3. **Access Control**: Once the user's identity is authenticated, the authentication layer collaborates with the access control mechanisms to determine the user's authorized level of access. This ensures that users only have access to the resources and functionalities permitted for their role or privileges.
4. **Integration with Identity Providers**: the authentication layer integrates with external identity providers to enable single sign-on (SSO) capabilities. In the case of ERA, the Identity Provider is Auth0 (https://auth0.com/)

The authentication layer of ERA is based on Auth0 (https://auth0.com/). Auth0 is a managed service that provides authentication and authorization services for applications and APIs. It offers a comprehensive set of tools and capabilities to handle the authentication layer of an application, making it easier for developers to implement secure user authentication.

Auth0 pricing policy includes a free plan offering up to 7000 users and 1000 Machine to machine Authentication.

Application Layer

The application layer is responsible for managing the interaction between the users or 3rd-party systems with the underlying system infrastructure. The basic components of the application layer are the **Web Application** and the **Back-office.** The web application is responsible for presenting information to users, managing user interactions and submitting data that will be eventually consumed by the back-office component. The back-office is responsible for consuming the submitted data and inserting them to both **MySQL** (https://www.mysql.com/) and **Elasticsearch** (https://www.elastic.co/), for allowing increased capacity for serving different use cases. Moreover, the Back-office, allows the connection of 3rd-party systems to the ERA ecosystem, by exposing the relevant API endpoints.

The web application is built on **React/Laravel** (https://react.dev/)(https://laravel.com/) stack and is served through an Nginx web server (https://www.nginx.com/), while the back-office is written in **Java** (https://www.java.com/en/) using the Apache Camel framework (https://camel.apache.org/). Both components are hosted to a respective EC2 Virtual Machine.

All components are containerized using Docker (https://www.docker.com/) and can be deployed to any infrastructure.

The application layer also includes the **API Gateway.** This is a service offered by AWS **without cost** and allows:

API Routing and Aggregation: The API Gateway handles routing and redirection of client requests to the appropriate backend services or APIs. It acts as a traffic director, forwarding requests to the corresponding endpoints based on the requested resource or functionality. It also allows for the aggregation of multiple API calls into a single request, reducing the number of round trips between the client and backend services.

Request and Response Transformation: The API Gateway can perform data transformation and protocol conversion to ensure compatibility between clients and backend services. It can modify request and response payloads, reformat data, or translate between different data formats or protocols.

Security and Authentication: The API Gateway provides security mechanisms to protect APIs and services from unauthorized access. It can enforce authentication and authorization checks, validate access tokens or API keys, and enforce security policies such as rate limiting, throttling, or IP whitelisting.

Traffic Management and Load Balancing: The API Gateway helps manage and distribute incoming traffic across multiple backend services or instances. It can perform load balancing to ensure optimal resource utilization and scalability. It can also handle traffic spikes or high-demand situations by applying rate limiting or queuing mechanisms.

Caching and Performance Optimization: The API Gateway can cache responses from backend services, reducing the load on those services and improving response times for subsequent requests. It can cache static or frequently accessed data, serving it directly to clients without involving backend services.

Analytics and Monitoring: The API Gateway collects metrics, logs, and monitoring data related to API usage, performance, and errors. It provides insights into traffic patterns, response times, error rates, and other key metrics. This information can be used for troubleshooting, capacity planning, or business intelligence purposes.

Application Layer Resources

Component	Service	Vendor	Item	On Demand Cost Estimation (USD)
Web Application	EC2	AWS	t4g.medium	35/Month
Back Office	EC2	AWS	t4g.medium	35/Month
-	API Gateway	AWS	-	-

Storage Layer

The storage layer is a component of a system or architecture that is responsible for managing the storage and retrieval of data. It provides a means for applications to store, access, and manipulate data in a structured manner. The EPA storage layer includes an Object Storage service (S3), two NOSQL databases (Mongodb & Elasticsearch) and finally MySQL to manage relations amongst data. The storage layer provides:

1. **Data Persistence**: The storage layer ensures that data is persisted and preserved even when the application or system is not actively using it. It manages the storage of data in a reliable and durable manner, preventing data loss or corruption.
2. **Data Organization**: The storage layer determines how data is organized and structured. It defines the data schema, data models, and relationships between different data entities. It provides mechanisms for creating, updating, and deleting data entities and managing their properties.
3. **Data Retrieval**: The storage layer enables applications to retrieve data based on specific criteria or queries. It provides querying capabilities that allow applications to search, filter, and sort data efficiently. This may involve the use of query languages, indexing techniques, or search algorithms.
4. **Data Integrity and Consistency:** The storage layer ensures the integrity and consistency of data by enforcing rules and constraints on data operations. It may include mechanisms such as transactions, concurrency control, and data validation to maintain the accuracy and reliability of stored data.
5. **Scalability and Performance**: The storage layer addresses the scalability and performance requirements of the system. It includes strategies for managing large volumes of data, distributing data across multiple storage nodes or partitions, and optimizing data access and retrieval for faster performance.

6. **Backup and Recovery**: The storage layer often includes mechanisms for backing up data and recovering it in the event of data loss, system failures, or disasters. It may involve regular data backups, replication, snapshots, or point-in-time recovery capabilities.
7. **Integration with Other Layers**: The storage layer integrates with other layers of the system, such as the application layer or the middleware layer, to provide data access and persistence services. It may expose APIs or interfaces that allow applications to interact with the stored data.

ERA Ecosystem is using S3 (https://aws.amazon.com/s3/), to store and retrieve binary files like PDFs from the different studies. It ensures that all files are stored in a central point and can be retrieved by different systems, with minimum costs. S3 follows a pay-as-you-go model, and the generated costs are quite low, in comparison with other storage technologies.

The next storage component is Mongodb, a NOSQL database, which is used to store the submitted data of the studies.

Due to its NOSQL nature MongoDB allows for:

1. **Flexible and Scalable Data Model**: MongoDB uses a flexible document model, allowing you to store data in a schema-less format called BSON (Binary JSON). This flexibility enables you to easily adapt and evolve your data model as your application requirements change. It also supports nested and hierarchical data structures, making it suitable for handling complex and dynamic data.
2. **High Performance**: MongoDB's architecture is designed for high performance and scalability. It employs a distributed and horizontally scalable approach, allowing you to scale your database across multiple servers or clusters to handle large amounts of data and high traffic loads. Additionally, MongoDB provides features like indexing, sharding, and in-memory caching to optimize query performance.
3. **Rich Querying Capabilities**: MongoDB offers a powerful query language with support for a wide range of query types, including ad-hoc queries, range queries, text searches, geospatial queries, and aggregations. It also supports indexing on any field, enabling fast and efficient query execution.

In the context of ERA, MongoDB is used as a serverless Managed Service, offered by MongoDB itself. The cost model follows a pay-as-you-go policy. The main cost units are the Read Processing Unit, Write Processing Unit and Storage. Due to the architecture of ERA ecosystems, all these dimensions have been minimized, hence the overall price is quite low.

An important role in the whole ecosystem plays the relational database. In the context of ERA, MySQL is the relational database served through the Relational Database Service of AWS (https://aws.amazon.com/rds/). RDS simplifies the process of setting up, operating, and scaling relational databases in the cloud. More specifically, RDS allows for:

1. **Managed Service**: Amazon RDS takes care of routine database tasks such as database setup, patching, backups, and software updates. It handles administrative tasks, allowing you to focus on your application development rather than database management.
2. **Scalability and High Availability**: Amazon RDS offers built-in scalability and high availability features. You can easily scale your database resources (compute and storage) up or down to meet the changing demands of your applications. Additionally, it provides automated backups, automated software

patching, and Multi-AZ (Availability Zone) deployment options for enhanced availability and data durability.

3. **Cost-Effective Pricing Model**: Amazon RDS offers a pay-as-you-go pricing model, allowing you to pay only for the resources you consume. It eliminates the need for upfront hardware investments and provides cost-effective options for database provisioning and scaling.

Finally, the last storage solution that ERA ecosystem uses is Elasticsearch, an open-source, distributed search and analytics engine. It is designed to provide fast and scalable full-text search, as well as real-time analytics capabilities, on large volumes of data. Elasticsearch allows for:

1. Full-Text Search: Elasticsearch excels at full-text search, enabling ERA to perform complex search operations on structured and unstructured data. It uses the inverted index data structure to index and search text, providing fast and relevant search results even on large datasets.
2. Real-Time Analytics: Elasticsearch supports real-time analytics and aggregations on ERA data. It can process and analyze data on-the-fly, allowing you to derive insights and perform complex aggregations, filtering, and statistical calculations in real-time.
3. Schemaless JSON Documents: Elasticsearch stores data as JSON (JavaScript Object Notation) documents. It is schemaless, meaning that you can index and search documents without specifying a predefined schema. This flexibility allows for easy data modeling and adaptability to changing data structures.

In ERA, Elasticsearch is hosted in an EC2 Virtual Machine.

Storage Layer Resources

Component	Service	Vendor	Item	On Demand Cost Estimation (USD)
Binary Storage	S3	AWS		12/Month
MongoDB	Atlas	MongoDB	Serverless	20/Month
MySQL	RDS	AWS	db.t3.small	50/month
Elasticsearch	EC2	AWS	t4g.large	70/Month

Cost Summary

Component	Service	Vendor	Item	On Demand Cost Estimation (USD)
Web Application	EC2	AWS	t4g.medium	35/Month
Back Office	EC2	AWS	t4g.medium	35/Month
-	API Gateway	AWS	-	-
Binary Storage	S3	AWS		12/Month
MongoDB	Atlas	MongoDB	Serverless	20/Month
MySQL	RDS	AWS	db.t3.small	50/month
Elasticsearch	EC2	AWS	t4g.large	70/Month

Continuous Integration/Continuous Development

The ERA Ecosystem is supported by a set of practices and processes used in software development to automate the building, testing, and deployment of applications. These practices ensure, frequent and reliable software releases, ensuring that new features, bug fixes, and updates can be delivered to users quickly and with high quality. The key used practices are:

- Version Control: Developers commit their code changes to a central version control system, enabling collaboration and traceability.
- Automated Build: The CI system automatically builds the application using build scripts or configuration files, ensuring that all necessary dependencies are included.
- Automated Testing: Automated tests, including unit tests, integration tests, and sometimes even user interface tests, are executed to verify that the application functions correctly and remains stable after each code change.
- Code Quality Checks: Code quality tools and static analysis are used to identify issues, enforce coding standards, and ensure consistency across the codebase.
- Continuous Feedback: CI provides immediate feedback to developers by reporting build failures, test failures, and other issues. This helps catch and fix problems early in the development cycle.

Moreover, ERA Ecosystem is supported by:

- Automated Deployment: The deployment process is automated using deployment scripts or tools. This ensures consistency and reduces the chances of human error.
- Environment Parity: The different environments (e.g., development, staging, production) are kept as similar as possible to minimize deployment-related issues caused by environmental differences.
- Rollback and Recovery: Quickly roll back to a previous version in case of issues or failures during deployment, ensuring minimal downtime and impact on users.

To support these practices, the following set of tools is required:

A Git installation, to host the source code of the different components. Each repository should have three branches. Development, Staging and Production. Each branch is connected to a different Virtual Machine, avoiding mixing experimental code with the production environment.

A CI/CD server: Currently ERA ecosystem is supported by Jenkins (https://www.jenkins.io/). It allows the automation of building, testing, and deploying the available source code.

Containerization: Each part of the system should use Docker containers to be packaged along with its dependencies, libraries, and configuration files. Each container runs as an isolated process, utilizing the host system's operating system kernel. Containers are lightweight, portable, and can be easily moved between different environments without compatibility issues.