[Technical Architecture Overview](https://wiki.eisgroup.com/display/GRC/Technical+Architecture+Overview)

# Advantages of the EIS Suite V20 Architecture

There are many advantages to the new application architecture that elevate EIS Suite V20 to be cutting edge.

#### >  It follows the [Reactive Manifesto](https://www.reactivemanifesto.org/).

Our system is flexible, loosely coupled, scalable, and fault-tolerant.

#### >  It is responsive.

Core components are loosely coupled, they scale independently, and they do not have central bottlenecks.

#### >  It is resilient.

Core components use industry-leading replication, containment, isolation, and delegation patterns. Built-in fail and recovery mechanisms ensure high availability and disaster recovery.

#### >  It is cloud-native.

The application is designed to be built and run in a public, private, or hybrid cloud.

#### >  It is stateless.

Each process or application is understandable in isolation without reference to past transactions.

#### >  It aligns fully with microservices characteristics.

#### >  It is multi-region.

This fault-tolerant application has easy failover to a backup region.

#### >  It uses a blue/green strategy.

Alternating production and staging servers are swapped to ensure rigorous testing.

# Technologies on Each Level

Different technologies may be used by each primary component, or level.

#### UI SPA

* React
* Typescript
* AntD
* RxJS

#### DXP Gateway

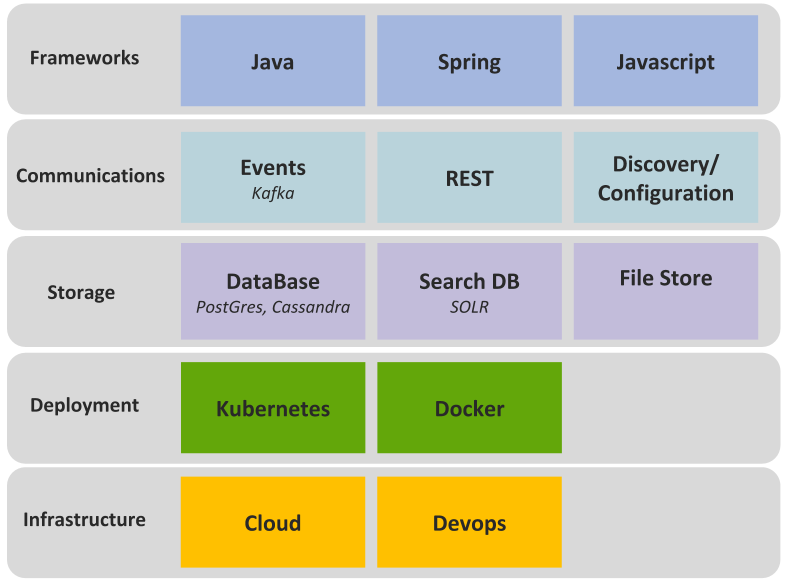
* [Play!](https://www.playframework.com/) framework

#### Microservices Backend

* Java (RxJava) Services Tier
* Event/Stream processing Kafka
* Persistence - Cassandra, PostgreSQL, MS SQL
* Search – SOLR
* Rules – [OpenL Tablets](https://wiki.eisgroup.com/display/GRC/OpenL+Technical+Documentation), [Kraken](https://wiki.eisgroup.com/display/GRC/Kraken)

#### Cloud

* MS Azure
* AWS
* Google Cloud
* private clouds



# Physical Deployment Diagram

A simple deployment of EIS Suite V20 may be represented by the diagram shown below.

Graphical user interface, diagram, application, Word

Description automatically generated

# Key Features of the EIS Suite V20 Architecture

## **Domain**

The [domain](https://wiki.eisgroup.com/display/GRC/Domain) framework is part of the platform microservices. It provides capabilities for entity management, DSL modeling, persistence, and validation.

* Dynamic modelling (JSON).
* Module is factory-based.
* Product evolution is supported.
* Product elements are moduled as business entities (Policy, Risk Item, Coverage, etc.) and assigned a business meaning from the start.

## **User Interface**

The UI layer is based on client-side javascript. This minimizes transitions to the server side and only accesses a stateless back end to load data and store changes using RESTful API.

* TypeScript-based non-blocking UI (React, strict unidirectional data flow, RxJS).
* Minimized non-blocking interaction between server and client.
* Typesafe automatically generated interfaces for client/server communication.
* Unified visual design ([Ant Design](https://ant.design/)).

## **Backend**

* Services exposed through REST API (HTTP,  WebSockets, JSON, Product schema)
* Stateless Services
* Message/Event-driven: Write [Commands](https://medium.com/event-driven-utopia/using-commands-events-and-queries-in-microservices-communication-3573f1fcfafe) and batch jobs are processed asynchronously using event streams (Kafka):
  + Eventually consistent
  + Scalable and resilient
  + Durable partitioned event streams vs. traditional messaging (Kafka)
  + Event stream processors responsible for failure recovery, retry of command execution, and event offset management.
  + Non-blocking response after command execution (callback streams).
* DB read and write separation (CQS pattern) to maintain responsive reads.
* Non-blocking reads (RxJava, Netflix Observers)
  + Resources are not held while awaiting a response (threads, sockets, etc.).
* Update idempotent DB writes
  + May be applied once or multiple times with the same result.

## **Storage**

* Horizontally scalable database with almost limitless capacity (Cassandra).
* Resilience using [ring network](https://en.wikipedia.org/wiki/Ring_network) instead of a typical [master-slave](https://en.wikipedia.org/wiki/Master/slave_(technology)) approach.
* On-demand data upgrade: No DDL impact, zero downtime on upgrade.
* Multi-data center support.

## **Deployment**

* System components packaged as containers (Docker).
* Containers deployed and managed by Cluster Manager (Docker Swarm, Apache Mesos, Google Kubernetes, or native cloud cluster managers).
* Auto scaling and dynamic reconfiguration of containers
* Auto-discovery-based configuration
* Cluster monitoring tools
* Support for Cloud Compute environments (AWS, Google Cloud, Microsoft Azure).
* Zero downtime deployment for maximized cost-efficiency.