[AIG USA 2](#_Toc15419164)

[Inshoring SOX functions 2](#_Toc15419165)

[CLEARWATER 2](#_Toc15419166)

[IDR 2](#_Toc15419167)

[RPA 2](#_Toc15419168)

[SAP HANA 2](#_Toc15419169)

[TREASURY 2](#_Toc15419170)

[OL/OLE 2](#_Toc15419171)

[JAVAH 2](#_Toc15419172)

[SACM 2](#_Toc15419173)

[SONIC 2](#_Toc15419174)

[SAILPOINT 2](#_Toc15419175)

[SPLUNK 2](#_Toc15419176)

[SKILLS 2](#_Toc15419177)

[AGILE AUDIT 2](#_Toc15419178)

[CI/CD 2](#_Toc15419179)

[AZURE TFS 2](#_Toc15419180)

[JENKINS 2](#_Toc15419181)

[TRAVIS CI 2](#_Toc15419182)

[TDD 2](#_Toc15419183)

[DEVOPS 2](#_Toc15419184)

[HADOOPP 2](#_Toc15419185)

[MONGODB 2](#_Toc15419186)

[SPARK 2](#_Toc15419187)

[Core 2](#_Toc15419188)

[SQL 2](#_Toc15419189)

[Stream 2](#_Toc15419190)

[STREAM 2](#_Toc15419191)

[AWS 2](#_Toc15419192)

[DJANGO 2](#_Toc15419193)

[BIG DATA SQL 2](#_Toc15419194)

[API 2](#_Toc15419195)

[12 FACTORS 2](#_Toc15419196)

[REST 2](#_Toc15419197)

[DOCKER 2](#_Toc15419198)

[KUBERNETES 3](#_Toc15419199)

[PYTHON 3](#_Toc15419200)

[LUIGI 3](#_Toc15419201)

[DASK 3](#_Toc15419202)

[PANDAS 3](#_Toc15419203)

[SQL ALCHEMY 3](#_Toc15419204)

[R 3](#_Toc15419205)

[JAVASCRIPT 3](#_Toc15419206)

[SCIKIT-LEARN 3](#_Toc15419207)

[TENSORFLOW 3](#_Toc15419208)

[PYTORCH 3](#_Toc15419209)

[MACHINE LEARNING 3](#_Toc15419210)

[Regression 3](#_Toc15419211)

[Boost 3](#_Toc15419212)

[Deep Learning 3](#_Toc15419213)

[POWER BI 3](#_Toc15419214)

[ORACLE 3](#_Toc15419215)

[SQL SERVER 4](#_Toc15419216)

[EXHIBIT 5](#_Toc15419217)

[AGILE AUDIT 5](#_Toc15419218)

[NOVA Target State 7](#_Toc15419219)

[IDR Data Governance 8](#_Toc15419220)

[IDR Data Quality Lifecycle 8](#_Toc15419221)

[IDR State Architecture 10](#_Toc15419222)

# AIG USA

## Inshoring SOX functions

## CLEARWATER

## IDR

## RPA

⬩Initiated from Accounts Payable (NJ), DBA <Tax, FIS Billing, FP&A Planning&Analysis, Comptrollers> ⬩Consultant: GENPACT ⬩Process 1: Batch creation + Monies moving ⬩Process 2: VOID/STOP Payment (Reversal) ⬩Systems AWD (Automated Work Distributor Imaging & Workflow), OASYS PrC (Fixed annuity Admin) ⬩ RPA: OPENSPAN PEGASYSTEMS

## SAP HANA

## TREASURY

#### OL/OLE

#### JAVAH

#### SACM

## SONIC

## SAILPOINT

## SPLUNK

# SKILLS

## AGILE AUDIT

## CI/CD

#### AZURE TFS

#### JENKINS

#### TRAVIS CI

## TDD

## DEVOPS

## HADOOPP

## MONGODB

## SPARK

#### Core

#### SQL

#### Stream

## STREAM

## AWS

## DJANGO

## BIG DATA SQL

## API

#### 12 FACTORS

#### REST

#### DOCKER

Docker is a platform for packaging, distributing, and running applications. As we’ve already stated, it allows you to package your application together with its whole environment.

This can be either a few libraries that the app requires or even all the files that are usually available on the filesystem of an installed operating system. Docker makes it possible to transfer this package to a central repository from which it can then be transferred to any computer running Docker and executed there (for the most part, but not always, as we’ll soon explain). Three main concepts in Docker comprise this scenario:

 Images—A Docker-based container image is something you package your application

and its environment into. It contains the filesystem that will be available

to the application and other metadata, such as the path to the executable that

should be executed when the image is run.

 Registries—A Docker Registry is a repository that stores your Docker images and

facilitates easy sharing of those images between different people and computers.

When you build your image, you can either run it on the computer you’ve

built it on, or you can push (upload) the image to a registry and then pull

(download) it on another computer and run it there. Certain registries are public,

allowing anyone to pull images from it, while others are private, only accessible

to certain people or machines.

 Containers—A Docker-based container is a regular Linux container created from

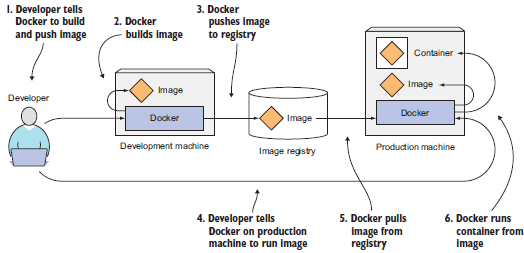
a Docker-based container image. A running container is a process running on

the host running Docker, but it’s completely isolated from both the host and all

other processes running on it. The process is also resource-constrained, meaning

it can only access and use the amount of resources (CPU, RAM, and so on)

that are allocated to it.



#### KUBERNETES

Kubernetes is a software system that allows you to easily deploy and manage containerized

applications on top of it. It relies on the features of Linux containers to run heterogeneous

applications without having to know any internal details of these applications and without having to manually deploy these applications on each host. Because these apps run in containers, they don’t affect other apps running on the same server, which is critical when you run applications for completely different organizations

on the same hardware. This is of paramount importance for cloud providers,

because they strive for the best possible utilization of their hardware while still

having to maintain complete isolation of hosted applications.

Kubernetes enables you to run your software applications on thousands of computer

nodes as if all those nodes were a single, enormous computer. It abstracts away

the underlying infrastructure and, by doing so, simplifies development, deployment,

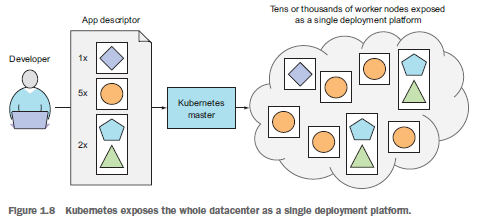
and management for both development and the operations teams.

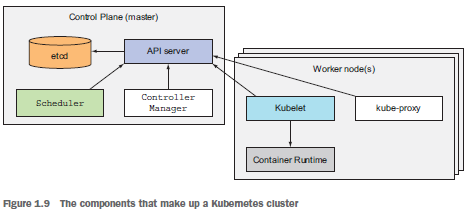
Deploying applications through Kubernetes is always the same, whether your cluster

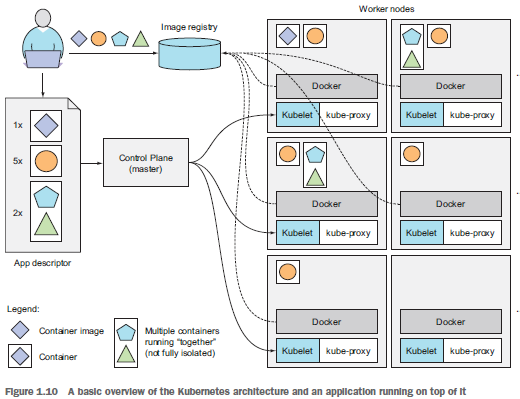
contains only a couple of nodes or thousands of them. The size of the cluster

makes no difference at all. Additional cluster nodes simply represent an additional

amount of resources available to deployed apps.







## PYTHON

#### LUIGI

#### DASK

#### PANDAS

#### SQL ALCHEMY

## R

## JAVASCRIPT

## SCIKIT-LEARN

## TENSORFLOW

## PYTORCH

## MACHINE LEARNING

#### Regression

#### Boost

#### Deep Learning

## POWER BI

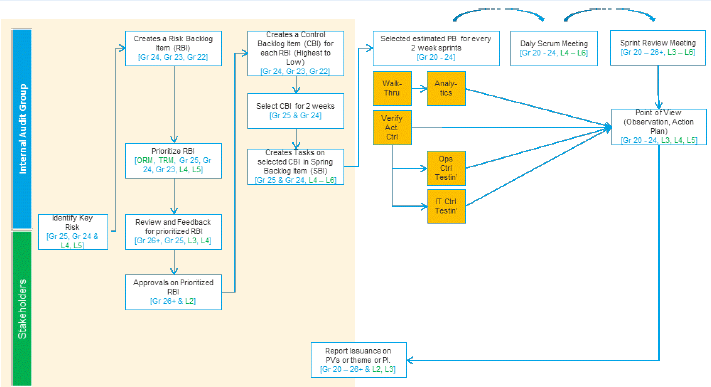
## ORACLE

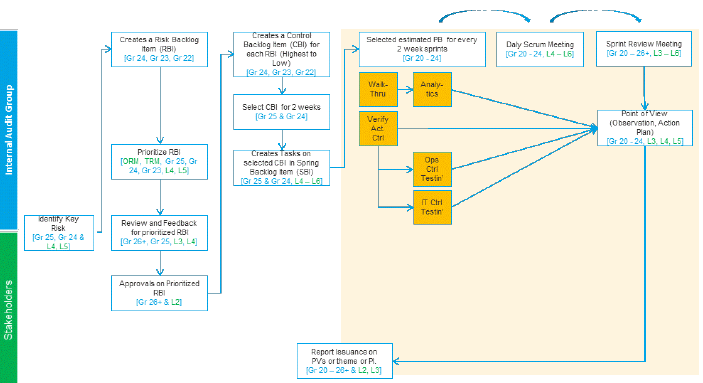
## SQL SERVER

# EXHIBIT

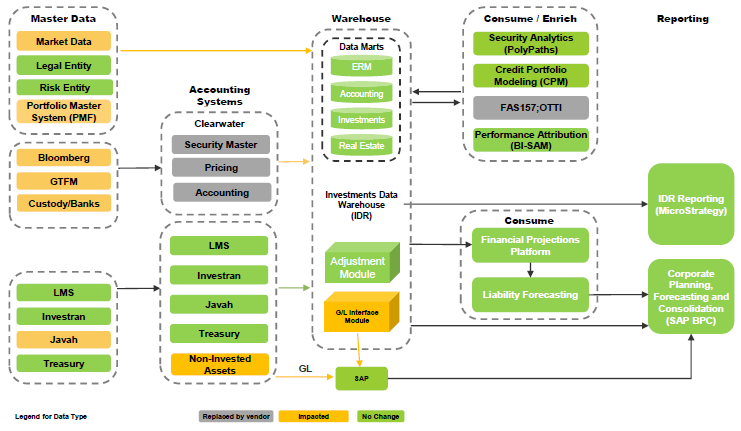
## AGILE AUDIT

Agile concepts: •Audit Increment planning” build a backlog of key risks and controls •Execute each sprint (2 week intervals) •After each sprint have a sprint review meeting with L4 to discuss results and initiate. After each sprint have tollgate to discuss stopping or continuing with audit •After each sprint and before next Sprint have Lessons learned session to discuss went well in sprint and what needs enhancements from next sprint •Holding daily scrum meetings (10 minutes) to discuss progress from yesterday, plan for current day and if any escalation is required





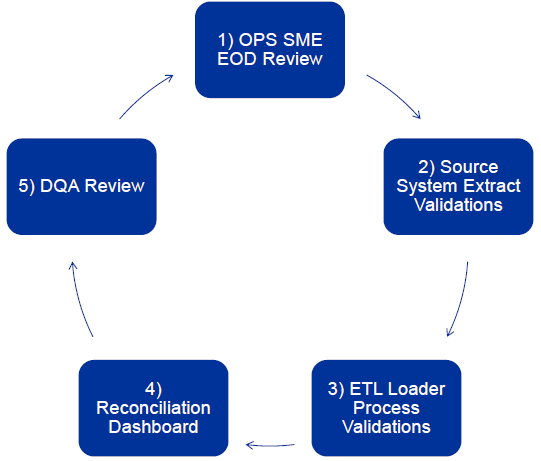
## NOVA Target State

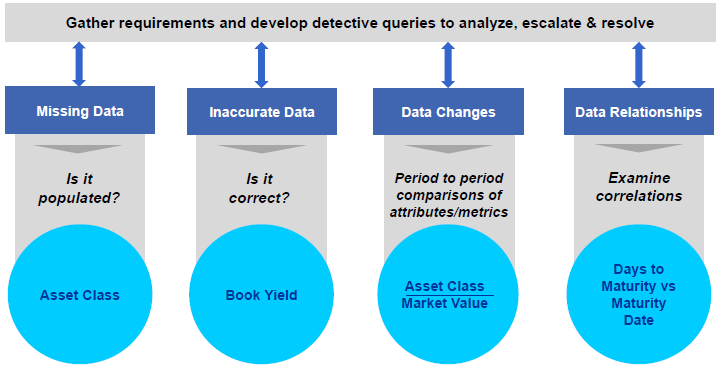


## IDR Data Governance



## IDR Data Quality Lifecycle





## IDR State Architecture

