Technical Architecture Industry Day
Support Documentation

4/7/2014

Introduction

This document provides an introduction to key concepts within the IAE To-Be Technical Architecture. It provides a point-in-time view of the architecture and may not represent the final architecture of the IAE system.

Architecture Overview

The legacy IAE environment consists of systems that were procured and are operated as independent technical entities. Shared components of the application stack for these systems are limited to colocation services that cover some systems and a consolidated end-user help desk that supports a different subset of the systems. One of the objectives of the IAE To-Be Architecture is to provide a technical foundation for consolidating all these systems into a single environment.

The IAE Team have created three key architecture products to date; the overall architecture, the architecture principles that guide the overall technical environment (described later) and a conceptual architecture exploring the key Common Services components of Hosting, Search, IAM, Data Store, Data Visualization and API Management. The Common Services contractor will translate this architecture into a design and then implement that design.

The foundational architectural components, described in the Conceptual Architecture, are *Nodes, Containers, Entry Points, Use Cases, and Services*. Each of these concepts is described below.

Containers are logical entities that define the characteristics that a physical deployment of the system must satisfy. *Nodes* are the smallest units of deployable functionality. Containers and Nodes are combined by a *Deployment Manager* to form *Node Instances*, which are deployed units of functionality. The IAE Conceptual Architecture makes no assumptions concerning the implementation of Node Instances and generalizes the architecture to include multiple possible physical architectures. As an example, the Conceptual Architecture describes Hybrid and Community Clouds that may not be required by the implementation proposed by the Common Services contractor.

The Conceptual Architecture also provides the flexibility at the physical level. For example, in one possible scenario a Node could be implemented using Java and expose its services as RESTful APIs. How this node is deployed depends on the container that will be used. When deployed in a Production Container, the underlying physical infrastructure may be owned by the Common Services contractor but when deployed in a Development Container, the infrastructure may be virtual machines running on a developer laptop.

Nodes

A Node is the the smallest unit of deployable functionality within IAE and encapsulates one or more capabilities. All nodes present their interface as a service, mediated through the Common Services API Management Service. Some nodes are foundational, for example the node that provides data storage may be required by a node that provides CRUD functionality for an entity.

An effective low level architecture and design will identify the units of deployable functionality and partition them into nodes. These nodes will be reflected throughout the implementation, including in partitioning responsibilities between individual development teams.

The functionality assigned to a Node may expand through its lifetime and this might eventually result in a node being split into two. For example a service bus node might start out as a simple proxy server, but gradually, through multiple iterations add key management, dead letter queue, throttling and other services. Eventually the Node may be split to provide more granular control of deployment.

Another significant use of nodes is to allow automated deployment across multiple environments. A node is the smallest unit of deployability and as such is the building block for deployment automation, continuous integration and other activities. As an example for search, the nodes might be Basic Search, Advanced Search, Search Admin, Production Search Index, Test Search Index. With this breakdown

- In a test environment, the a test team might deploy the following nodes: Basic Search, Advanced Search, Search Admin and the Test Search Index
- Alternatively, a production-like capability could be deployed by substituting the Production Search Index for the Test Search Index
- The separation of Search Admin into its own node allows separate development, deployment and testing independent of the rest of the search components.

User Interface components are also considered Nodes. An appropriate design will allow the substitution of alternate user interfaces to support mobile devices (as a native app rather than a responsive web app) or third party providers of IAE services.

Nodes are not tied to specific technical implementations, but rather a functional capability. So a web server might be part of the infrastructure that serves some user-facing capability, but would not in itself be a node. It is the collection of functional capabilities that represent the node. In the case of a web server, a VM with webserver software installed (IIS, apache, nginx etc) could be one possible implementation of a node, but the same functionality could also be provided through html files (perhaps with client-side javascript populating data through standard IAE APIs) that are served directly from Akamai NetStorage or AWS S3, without the direct need for a web-server. The idea of node is to encapsulate this capability and then allow development both of the external functionality and the internal structure and capabilities as the overall environment matures.

Container Templates and Containers

Containers are the logical unit that determine the structure and capabilities of a physical environment. When a functioning system is deployed, a Container is combined with one of more Nodes to create Node Instances which collectively implement a running system capable of servicing users and client systems. Additional information on Containers provided below.

Entry Points

All Nodes present their capabilities through Entry Points. The entry point is generalized interface for discovery and use of the services that Nodes provide.

Use Cases

A use case provides an example of how a system, user or node interacts with a service provided by a Node through an Entry Point.

Services

Services are the appropriate abstraction layer over our capabilities. These services need not be of any unified protocol, but would allow IAE to manage access, integrity, and measure usage of the system. We expect that services will be layered with low-level services available internally and business-level services available widely.

Container Description

The Common Services environment is a collection of largely independent technical environments that support production deployment as well as development and test activities for IAE. The table below outlines the attributes of these environments, called Containers, and provides and differentiates between the various container types that the Government believes will be required. The Common Services contractor will determine an appropriate solution for supporting the likely deployment scenarios. The contractor should propose an architecture that provides an ability to stand up and tear down these containers on demand.

Attribute	Attribute Definition	Production	Production-Like	Minimal Capacity	Component
Template Description	A narrative description of the	A container that is capable	A container that is capable of	A container that supports the full	A minimal environment that

	template	of supporting the full availability, use and data for IAE in a production environment	supporting the full availability, use and data for a production environment without support for long term sustainability	range of IAE capabilities at a minimal level	provides the basic infrastructure of the environment where individual technical and functional capabilities may be deployed
Usage Scenarios	How the program might use VMs and other resources deployed into a container built on this template	Production	Load Testing, UAT, Penetration Testing	Demonstrations, Functional Testing, Integration Testing, Localized Development Team Usage	Development
Number of concurrently active containers based on this template	How many containers will be deployed simultaneously	One	One, perhaps two depending on work activity	Three to four	Multiple, perhaps as many as one per developer
Life Expectancy	A typical length of time that an environment based on this container template would last	Permanent	Short term, to meet specific activities such as Load Testing, User Acceptance Testing	Minimally configured environments will be in continuous use, but individual environments may be torn-down and replaced with an updated version on a daily or weekly basis	Minimally configured environments will be in continuous use, but individual environments may be torn-down and replaced with an updated version on a daily or weekly basis
Service Demand	The level of usage, number of users, API calls and other measures	Full production capacity	For load testing, demand will be sustained and above-peak load, including load that generates failures; for other activities, minimal usage	Individual development team members, IV&V and customer usage, CI testing which may generate considerable load depending on the level of concurrency achieved during testing	Developer use, internal development team use
Attribute	**Attribute Definition**	**Production**	**Production-Like**	**Minimal Capacity**	**Component**

Demand Profile	The daily, weekly or long term demand profile; for example, not used at night or over weekends	Production demand profile	Sustained usage under load testing, other activities Business Hours only (can be turned off at night and weekends)	Business Hours usage (can be turned off at night and weekends)	Business Hours usage (can be turned off at night and weekends)
Demand Projection	How demand will grow over the long term	Production demand projections	No long term demand growth projected	No long term demand growth projected	No long term demand growth projected
Recovery Time Objective	A container characteristic that measures the requirement for the Recovery Time Objective (RTO) of the container	Production RTO	No objective set	No objective set	No objective set
Recovery Point Objective (RPO)	Standard RPO definition for data within the environment	Production RPO	No objective set	No objective set	No objective set
Availability	A container characteristic that measures the requirement for the system availability (available to the end user) of the container	99.99%	99.99%	No objective set	No objective set
FISMA Classification and Level	Implementation of NIST security standards and guidelines	Moderate	Moderate	Low (assumes appropriately designed test data)	Low (assumes appropriately designed test data)
Geographical Location	Restrictions on geographical location	CONUS	CONUS	CONUS	CONUS
Storage of Sensitive Data	Whether PII or other information may be stored or processed within the environment	Yes	Yes	No	No