

Unit #10

MIS5203

System Development

Agenda

- Distributed Systems
 - File Server Architecture
 - Client/Server Architecture
 - N-Tier Architecture
 - Thick versus Thin clients
 - Cloud Architecture
 - Service Oriented Architecture (SOA)
- Example Cloud-based N-Tier SOA Application Development System
- Control Stages and Objectives

Distributed Systems

Systems for Local Area Networks (LANs)

LAN: the cabling, hardware, and software used to connect workstations, computers, and file servers located in a confined geographical area

- Typically within one building or campus

Systems that operate in a local-area networked environment are generally called client/server systems

Some computers in the network are servers, providing some kind of service (such as data) to the clients

The system's human users are typically sitting at client computers, and share the common serviced provided by server computers

LAN systems usually are either based on either:

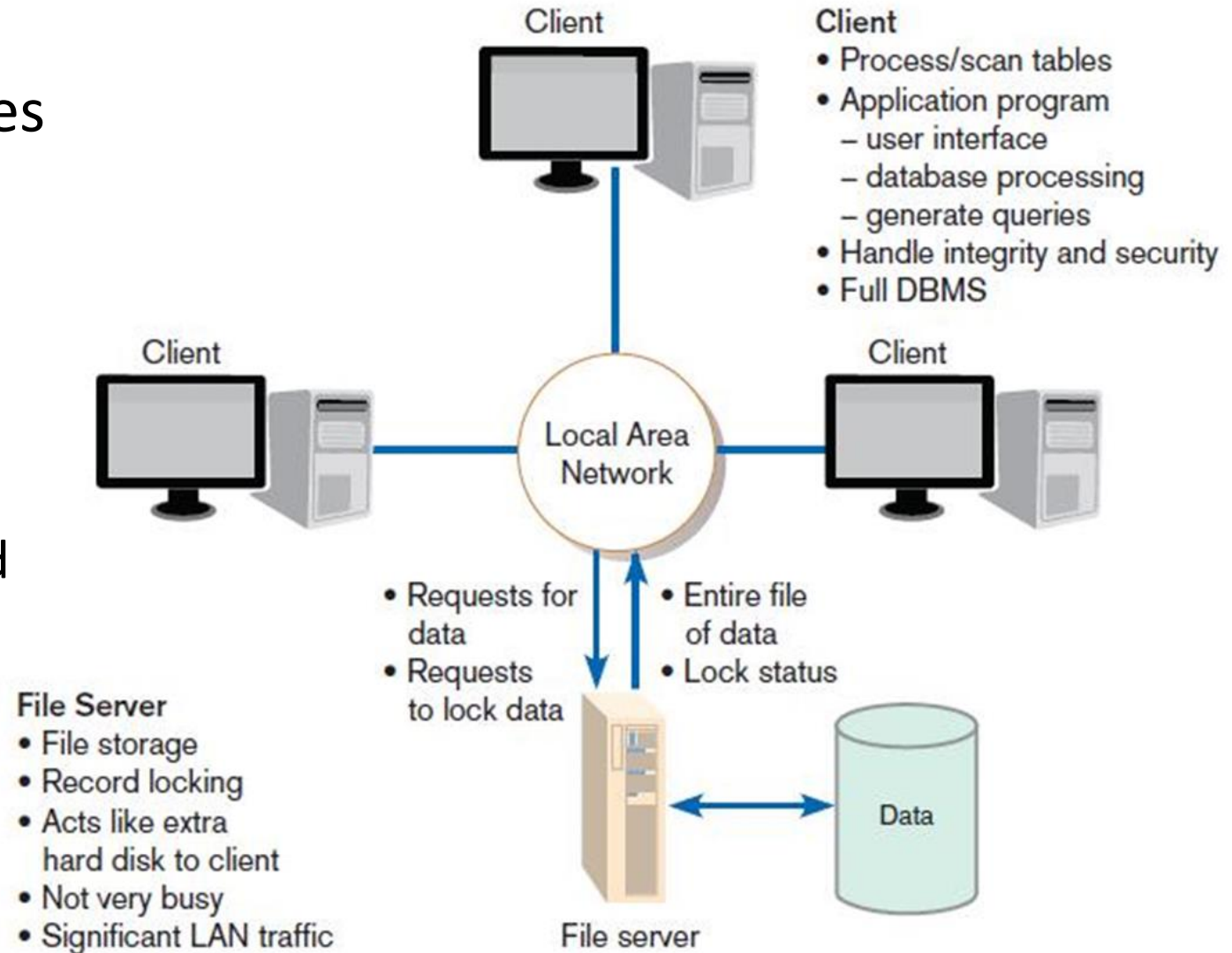
- LAN-based file server architecture
- Client/server architecture

Note: we are not talking about the Web, but a local area network (LAN) such as that used by a single department in a company

University computer labs often configured as a LAN. In an environment like this, there will often be a computer that provides file and print resources for the other computers in the network

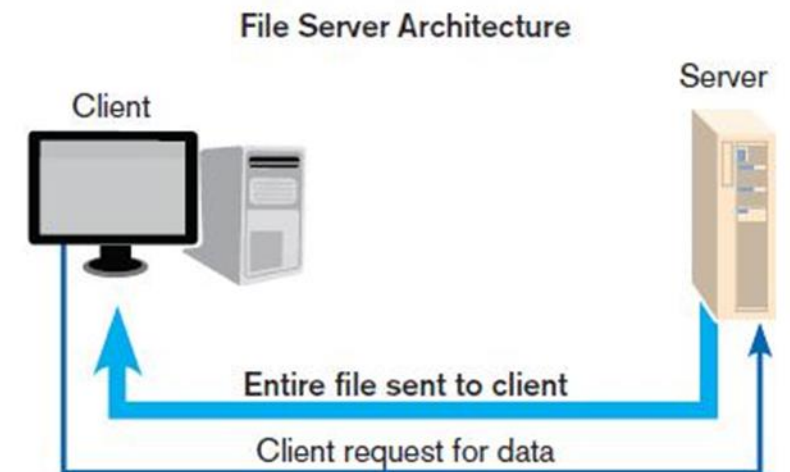
File Server architecture

- **File server:** a device that manages file operations and is shared by each client PC attached to a LAN
- The simplest configuration
 - Applications and data control take place on the client computers.
 - The file server simply holds shared data



Limitations of File Servers

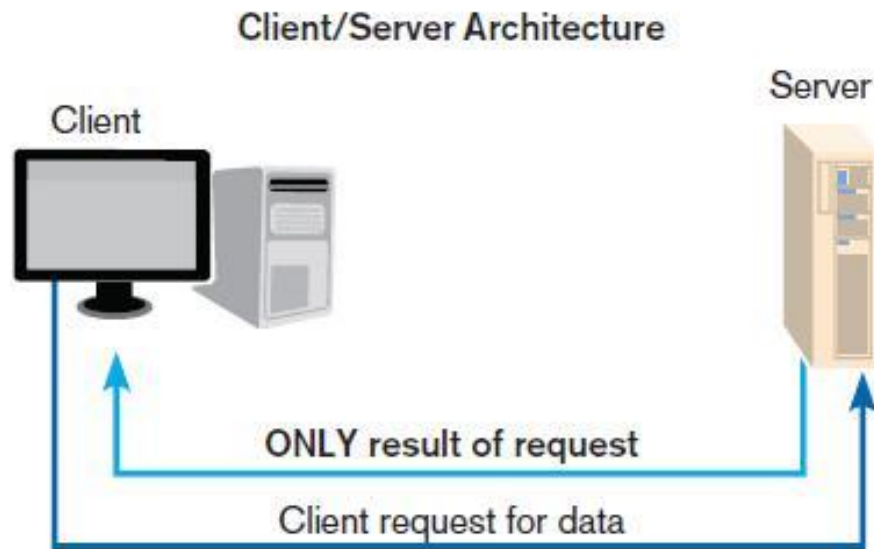
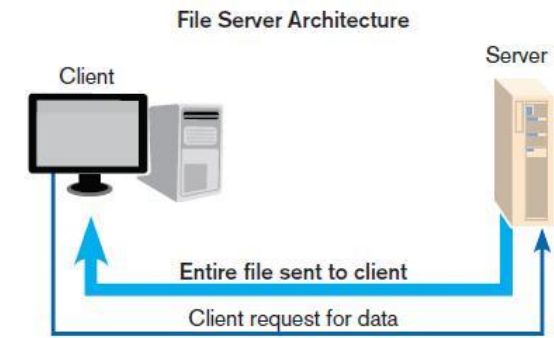
- Excessive data movement
 - Entire dataset must be transferred, instead of individual data records
- Need for powerful client workstations
 - Each client workstation must devote memory and computational resources to run a complete standalone application
- Decentralized data control
 - Data file concurrency control, recovery, and security are complicated



Client-Server Architecture

LAN-based computing environment in which

- A central database server or engine performs all database commands sent to it from client workstations
- Application programs on each client concentrate on user interface functions



Increased efficiency and control over File server

- Server only sends specific data, not entire files, which saves on network bandwidth
- Computing load is carried out by the server
 - Increasing security
 - Decreasing computing demand on the clients

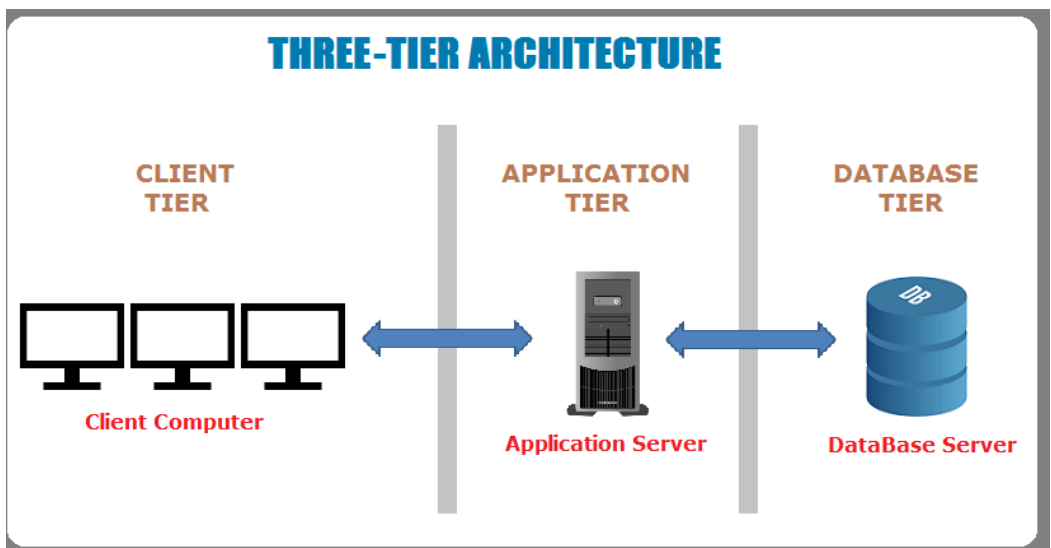
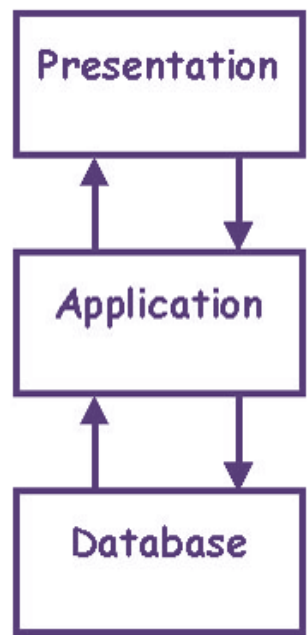
Some differences between File Server and Client/Server Architectures

Characteristic	File Server	Client/Server
Processing	Client only	Both client and server
Concurrent Data Access	Low—managed by each client	High—managed by server
Network Usage	Large file and data transfers	Efficient data transfers
Database Security and Integrity	Low—managed by each client	High—managed by server
Software Maintenance	Low—software changes just on server	Mixed—some new parts must be delivered to each client
Hardware and System Software Flexibility	Client and server decoupled and can be mixed	Need for greater coordination between client and server

Client/Server System Architecture

- Application processing is divided between client and server
 - Client manages the user interface
 - Database server is responsible for data storage and query processing

N-Tier Architecture



Presentation tier

The top-most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.



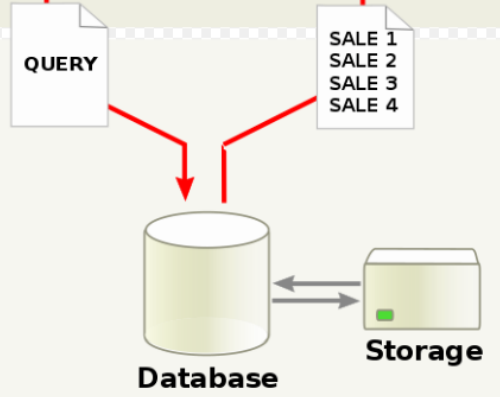
Logic tier

This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.

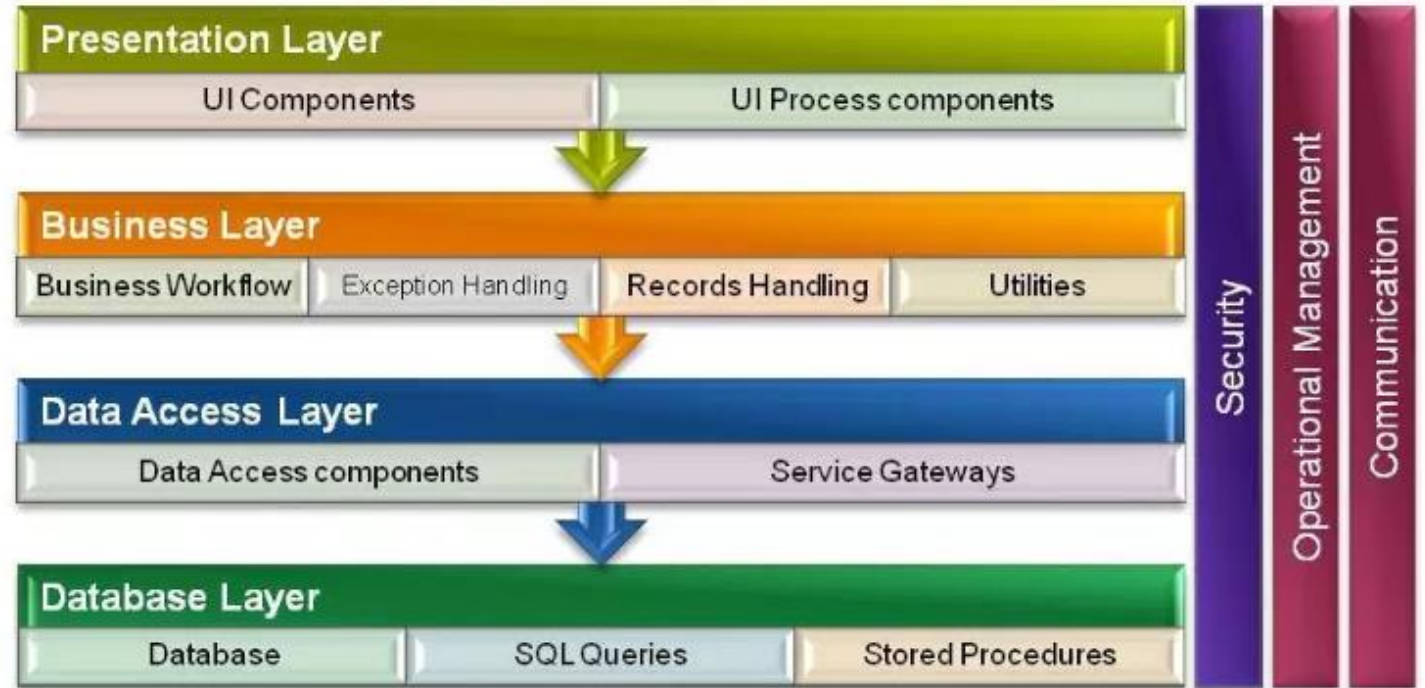
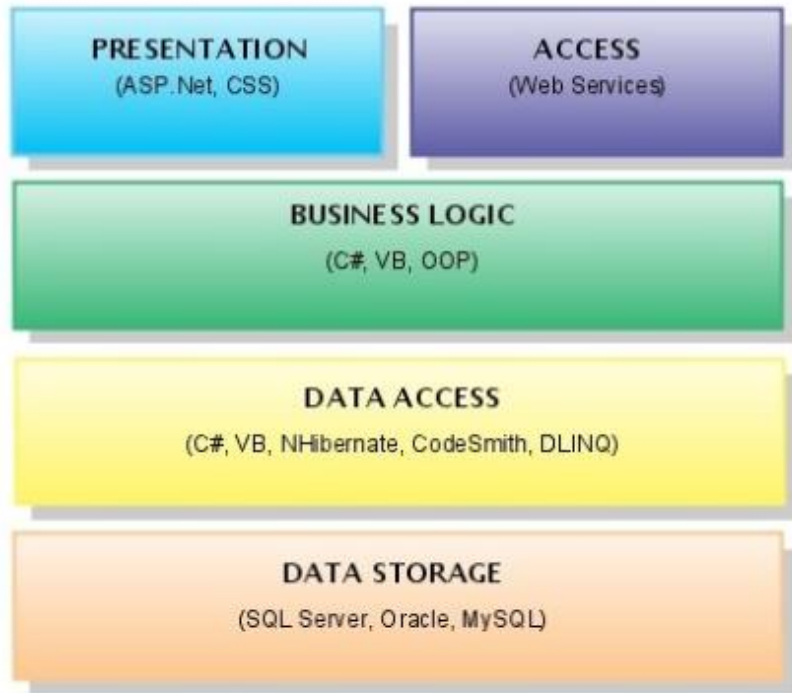


Data tier

Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.

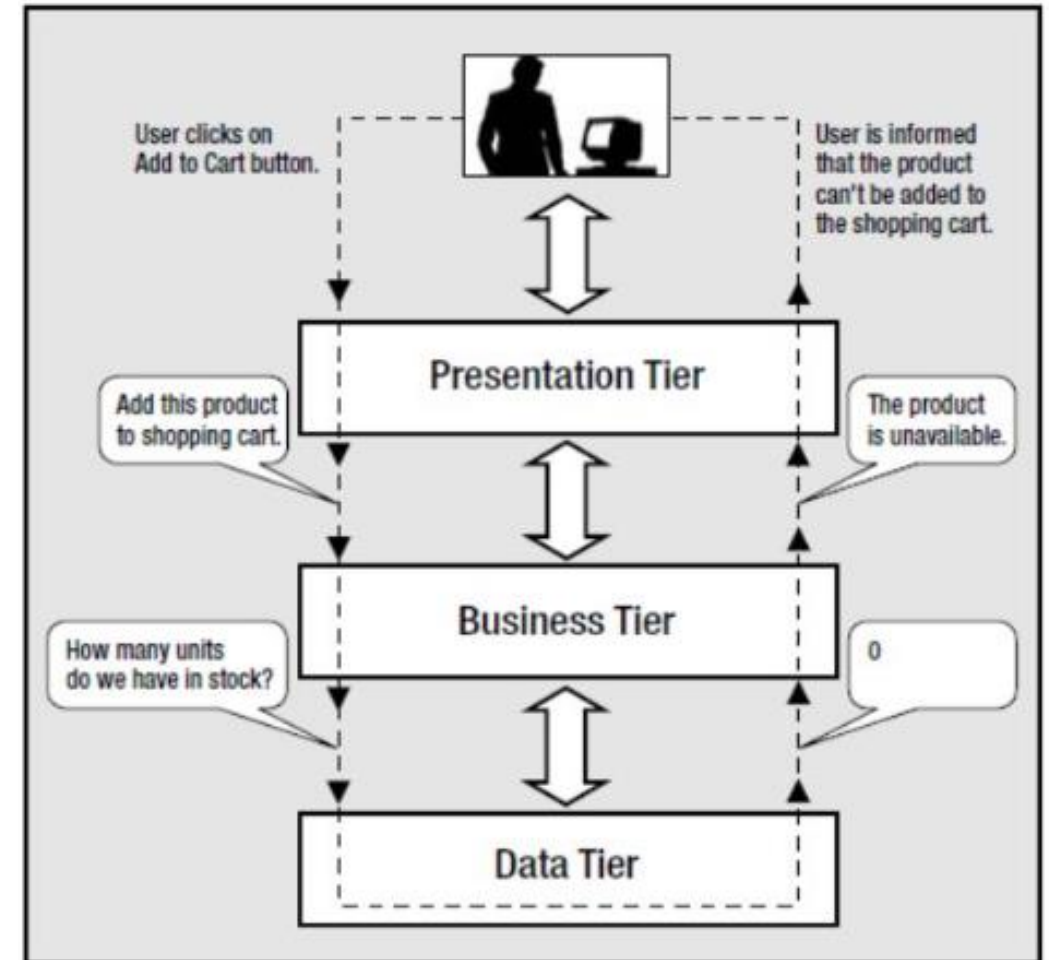
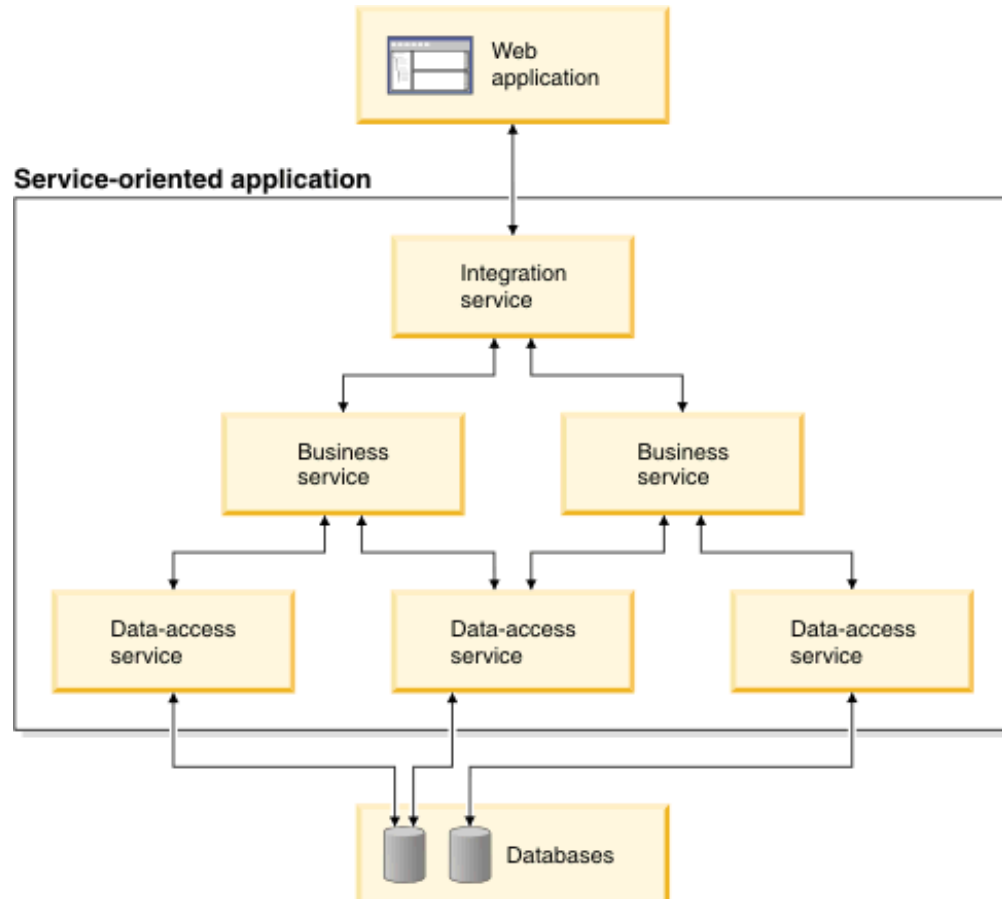


N-Tier Applications



Where's the programming code?

N-Tier Applications



“Thin” versus “Thick” Client

The degree to which application processing takes place at the client vs. the server

- Web based systems are often thought of as having thin clients
 - Because the only processing is what the browser does
 - Presentation and user interface, with business logic
 - Data processing taking place at the server
- But many feature-rich application programs run within the client, making periodic data requests to the server
- These clients are “fatter” *It's all a matter of degree*

Cloud Computing

- Provision of applications over the Internet
- Customers do not have to invest in the hardware and software resources needed to run and maintain the applications, but are charged on a per-use basis
- Amazon Example:
 - Amazon Web Services (AWS)
 - Simple Storage Service (S3)
 - Elastic Compute Cloud (EC2)

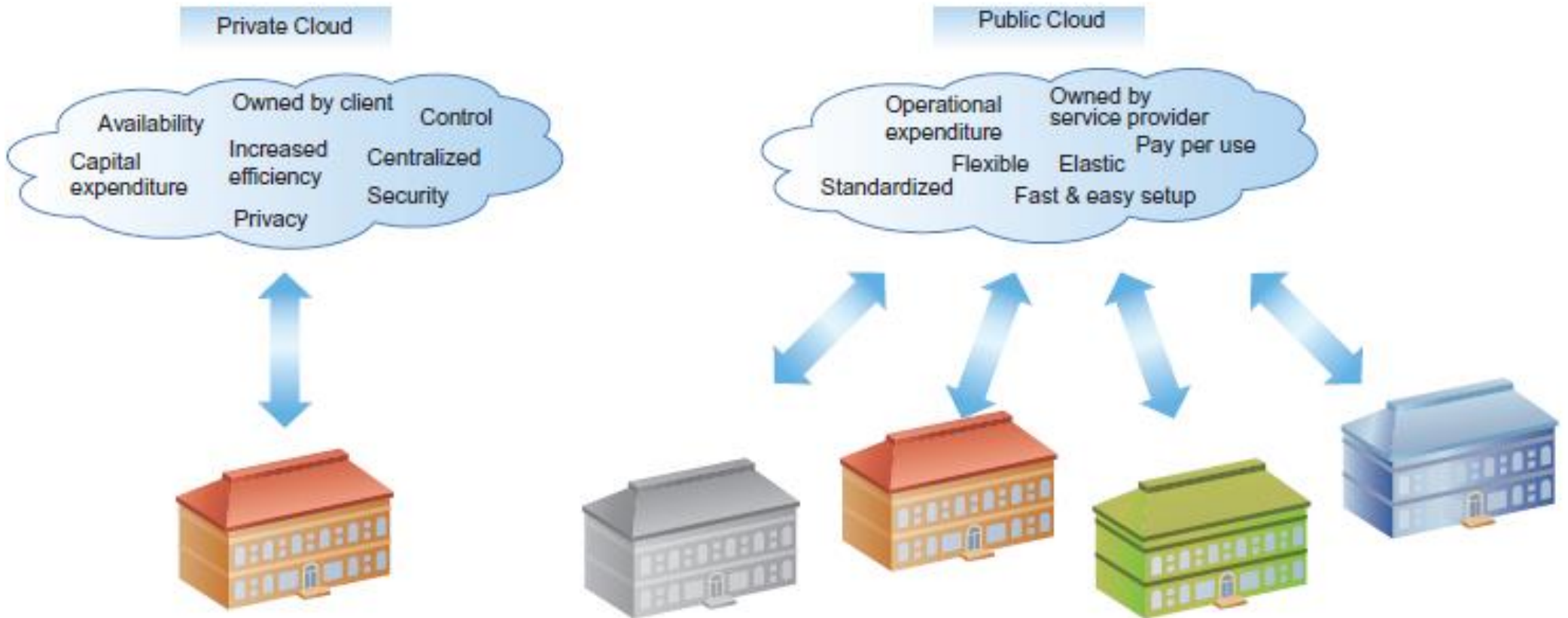
Cloud Computing – Service Models

- **Infrastructure as a Service (IaaS):**
provides basic processing, storage, and network capabilities
- **Platform as a Service (PaaS):**
customers run their own applications, using tools provided by the service provider.
- **Software as a Service (SaaS):**
applications are provided by the service provider



- IaaS is most basic service model
- PaaS will include infrastructure along with platform
- SaaS will typically include both the platform and infrastructure on which the application runs

Private versus Public Clouds



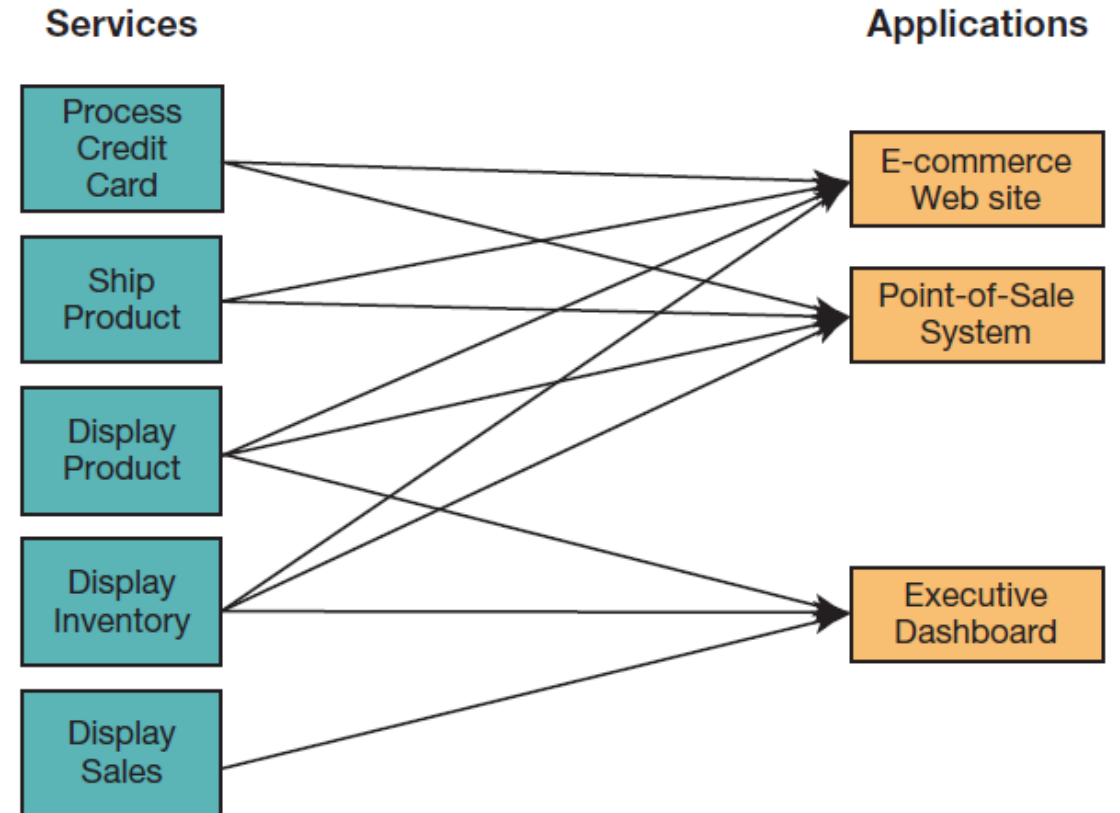
Service Oriented Architecture (SOA)

A **software** architecture

- Business processes broken down into individual components (services)
- Designed to achieve desired results for the service **consumer**
 - Application
 - Another service
 - Person (user)

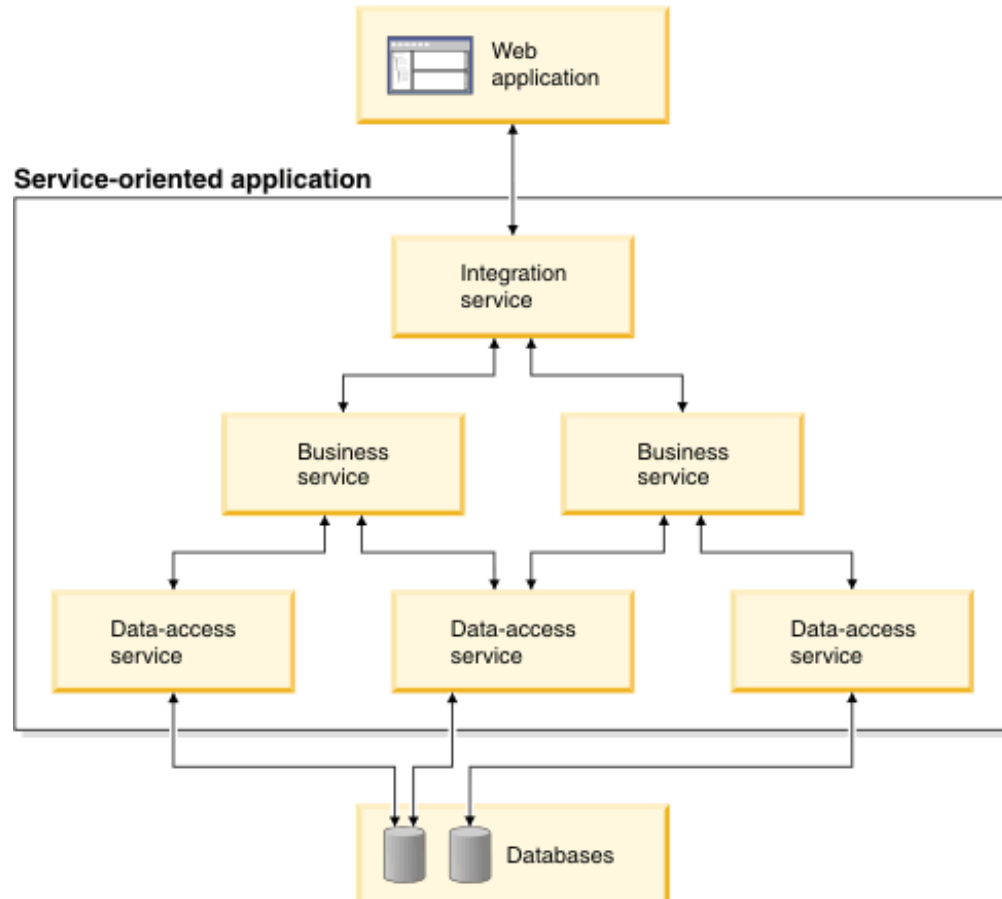
Principles:

- Reusability
- Interoperability
- Componentization



Using SOA, multiple applications can invoke multiple services

N-Tier Applications using SOA in the cloud



Agenda

✓ Distributed Systems

- ✓ File Server Architecture

- ✓ Client/Server Architecture

- ✓ N-Tier Architecture

- ✓ Thick versus Thin clients

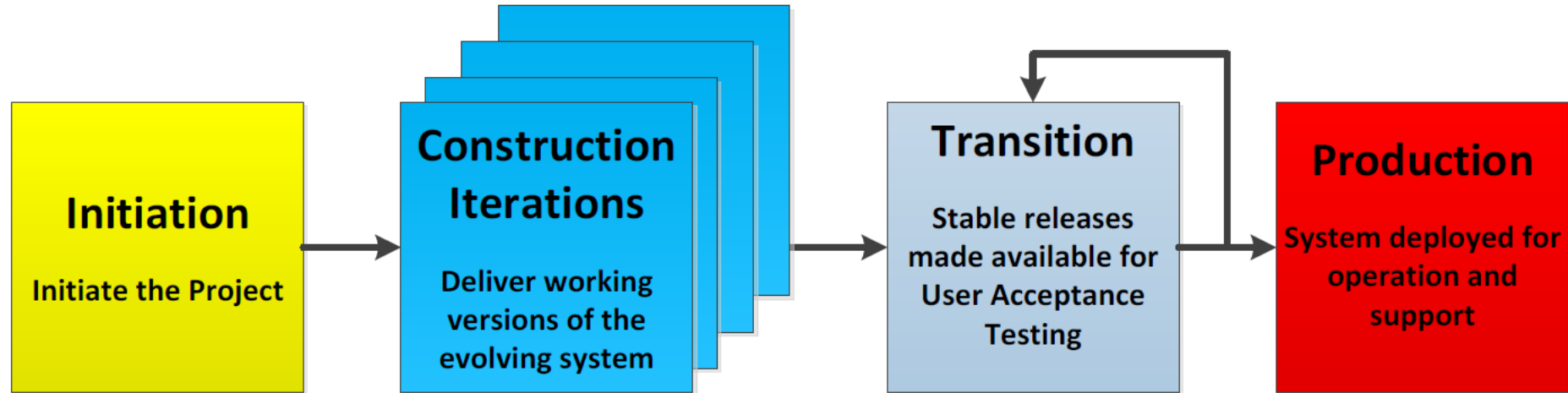
- ✓ Cloud Architecture

- ✓ Service Oriented Architecture (SOA)

- Example Cloud-based N-Tier SOA Application Development System

- Control Stages and Objectives

Development Phases – An example...

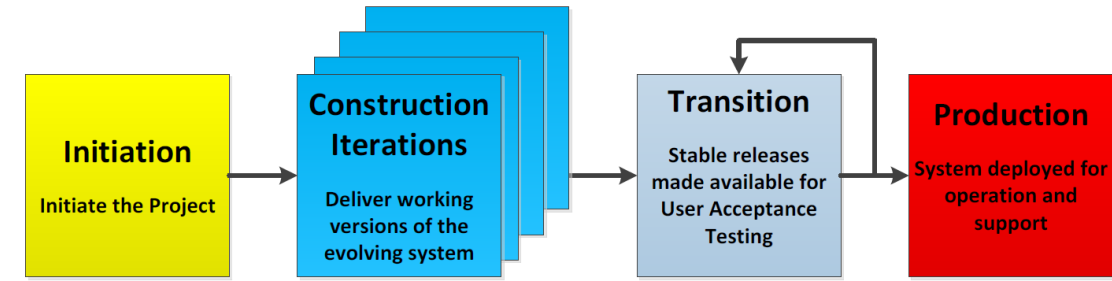


Four RAD / Agile methodology phases for developing applications:

1. Initiation – UX Design and Technical Infrastructure Setup
2. Construction Iterations – Develop and assure the quality of the system
3. Transition – User Acceptance Testing
4. Production – Deploy the application

Development Phases

An example...



Example milestones scheduling notable development accomplishments included in the plan and distributed among the phases:

1. Initiation

- Milestone 0 – Development Infrastructure in Place
- Milestone 1 – Functional Specifications Complete, Design Document and Application Configuration Guide Drafted



2. Construction Iterations

- Milestone 2 – Landing Page, Queries and Results Page
- Milestone 3 – Simple Query Drill-down Implemented and Test Plan Complete
- Milestone 4 – Data Pre-Processor and Dynamic Display Implemented
- Milestone 5 – Entity Detail Page and Data Stewardship Implemented
- Milestone 6 – Standalone Interactive Viewer Implemented

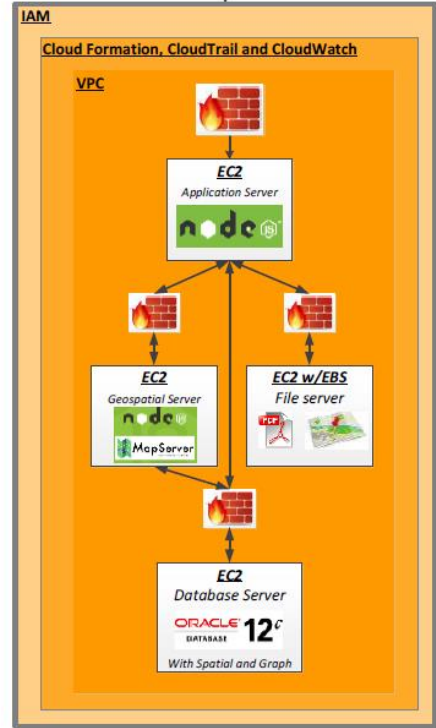
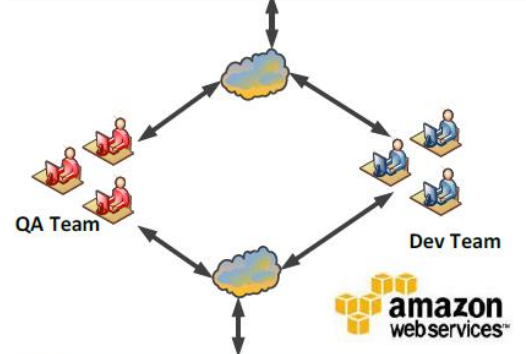
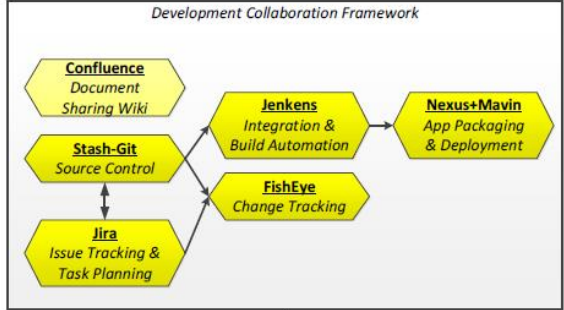
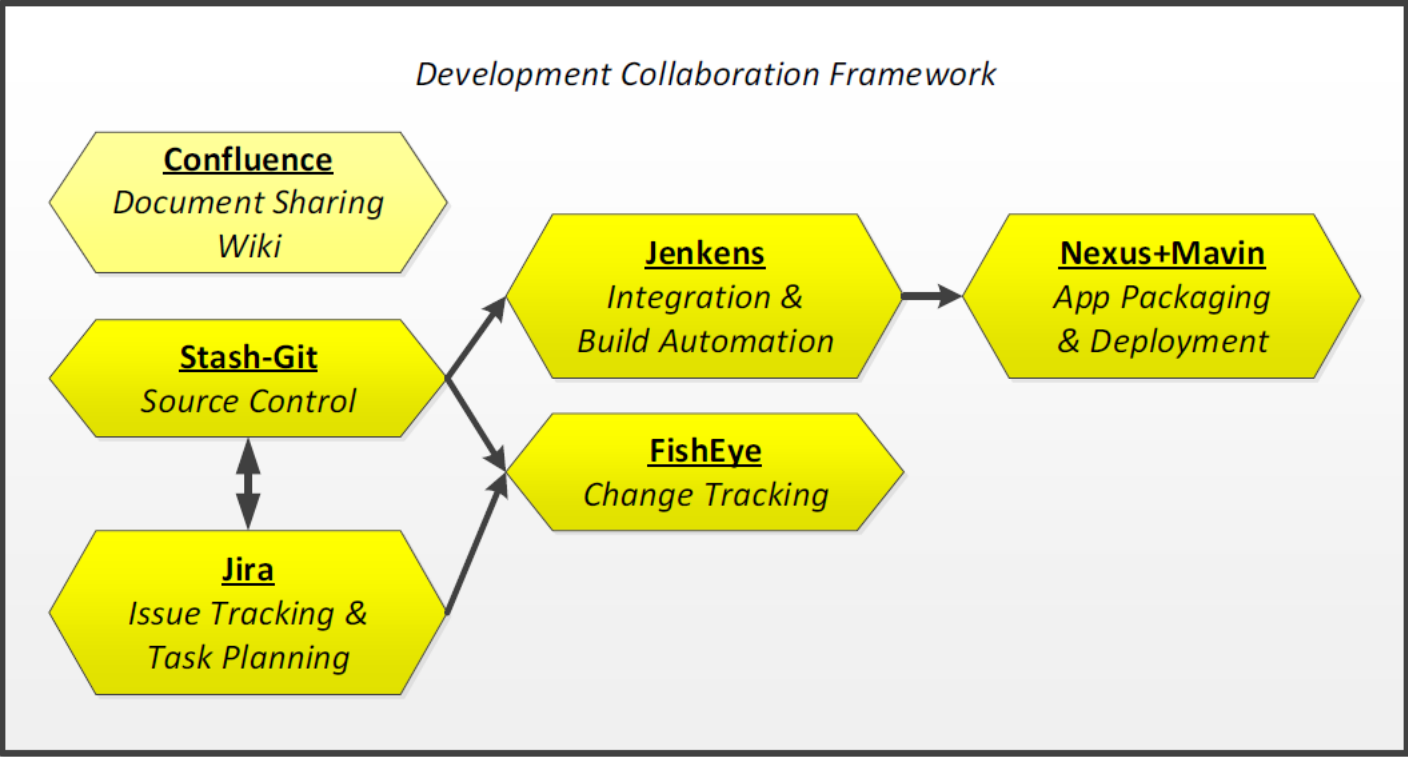
3. Transition

- Milestone 7 – Code Complete, User Acceptance Testing, Test Final Delivery Package

4. Production

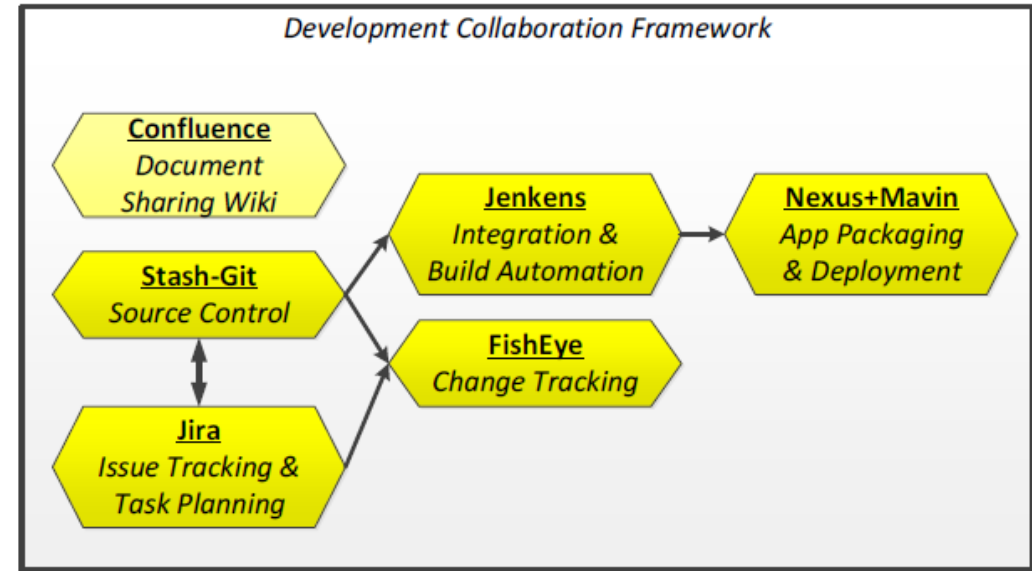
- Milestone 8 – Application Approved and Delivered

Development Infrastructure Example...



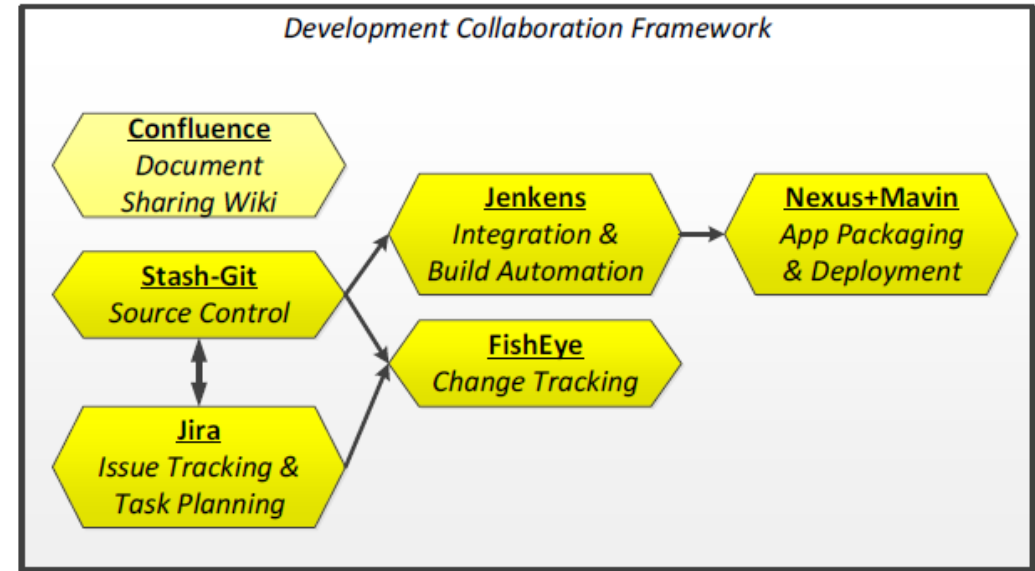
Document Sharing Wiki

- Central content management website for organizing, sharing, accessing and providing feedback on documents, meeting notes, requirements and design documents, and project plans and artifacts
- Confluence will be maintained by the development team managers to provide a common place for finding and accessing the current set of project requirements, designs and plans.
- Stash or the more recent Bitbucket is used for serving as a repository for tracking revision history of documents,
- *“Confluence can be linked to Jira to provide project management with reports that provide insight into the status of development work and issue creation”*



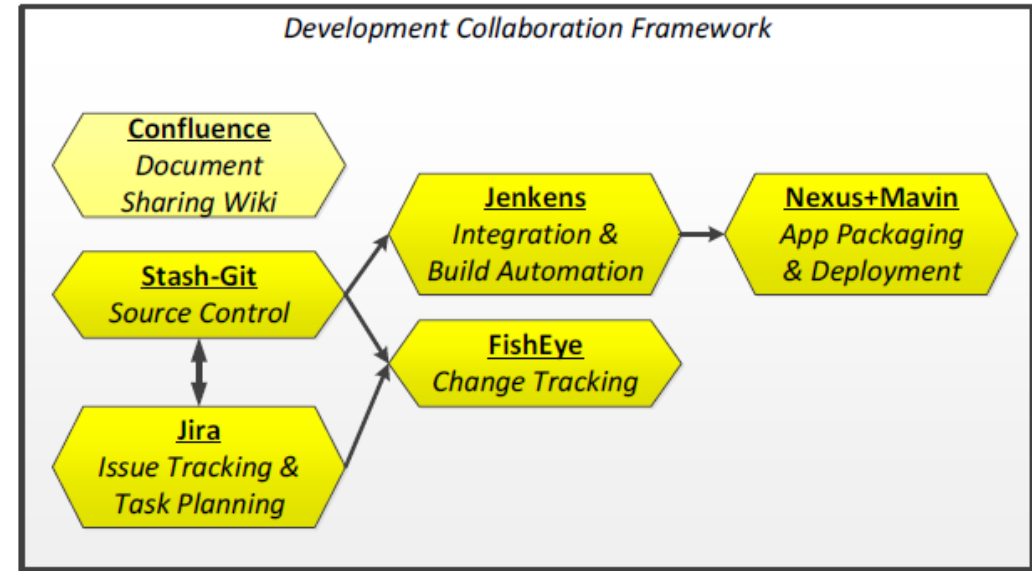
Source Control

- A commercial off the shelf (COTS) proprietary web-based hosting repository service for distributed access and version control of programming code
- Enables maintaining versioned shareable software code and design artifacts with check-in/check-out and maintenance capabilities



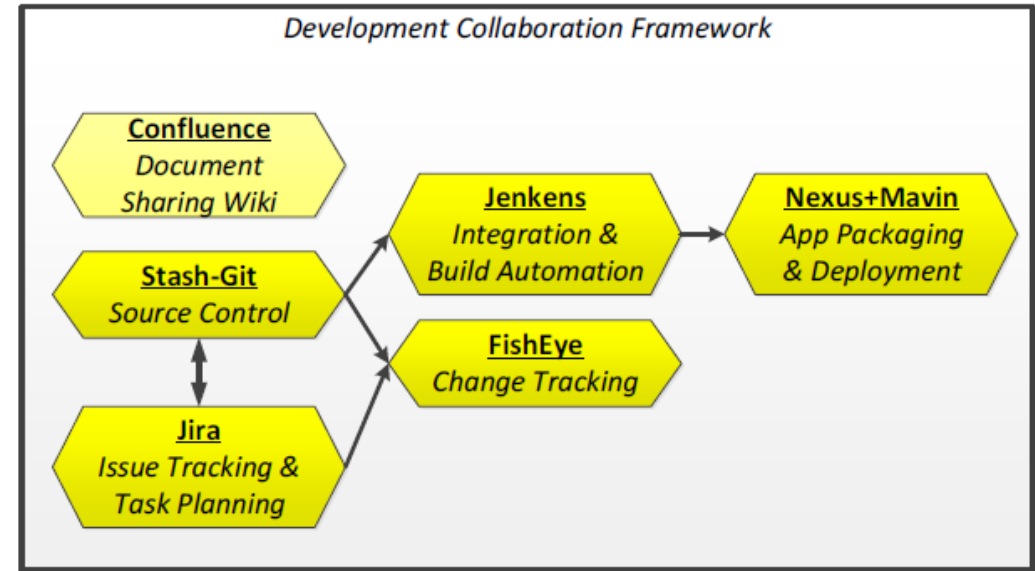
Issue Tracking System

- Enables organization, prioritization, triage, planning and tracking resolution of issues and project tasks
- Provides visibility of issues and tasks on “To Do”, “In Progress”, “For Review”, and “Done lists”
- *Integration of Issue Tracking and Source Control Systems enables end-to end traceability of issue and tasks through resolution to source code implementation and issue to source code resolution*
- Change tracking and control enables visualizing and reporting on revisions and changes made to source code and documents by project teammates
- *Enables linking software issues documented in Jira to differences, sets of changes, full source code and provides a visual audit trail of changes over time in Stash-Git*



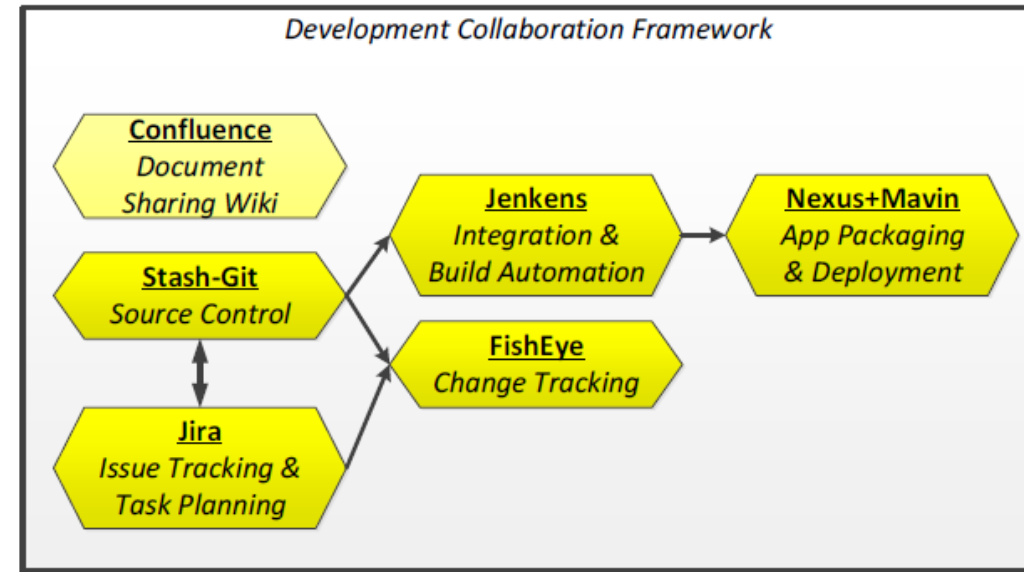
Issue Tracking System

- Enables organization, prioritization, triage, planning and tracking resolution of issues and project tasks
- Provides visibility of issues and tasks on “To Do”, “In Progress”, “For Review”, and “Done lists”
- *Integration of Issue Tracking and Source Control Systems enables end-to-end traceability of issue and tasks through resolution to source code implementation and issue to source code resolution*
- Change tracking and control enables visualizing and reporting on revisions and changes made to source code and documents by project teammates
- *Enables linking software issues documented in Jira to differences, sets of changes, full source code and provides a visual audit trail of changes over time in Stash-Git*

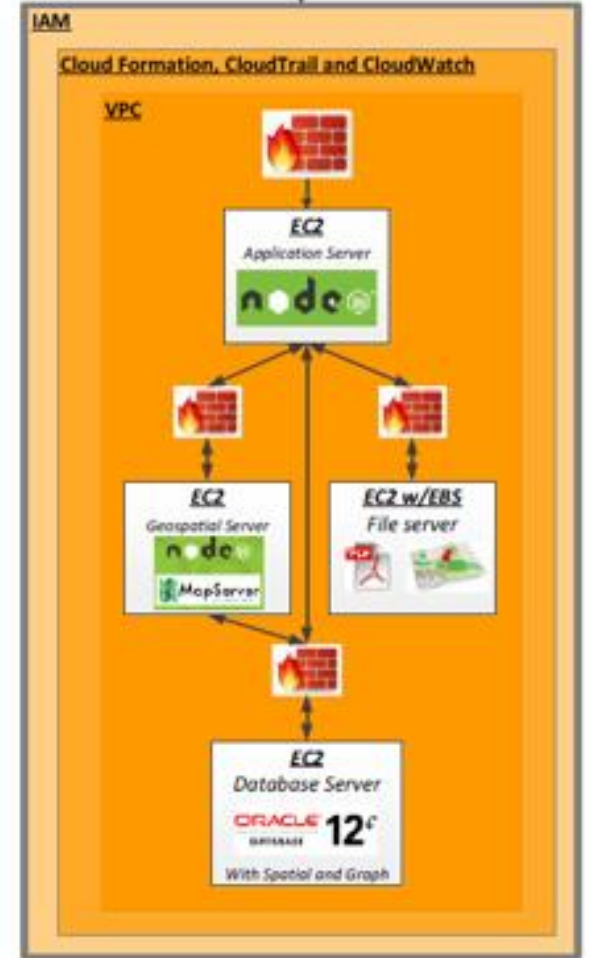
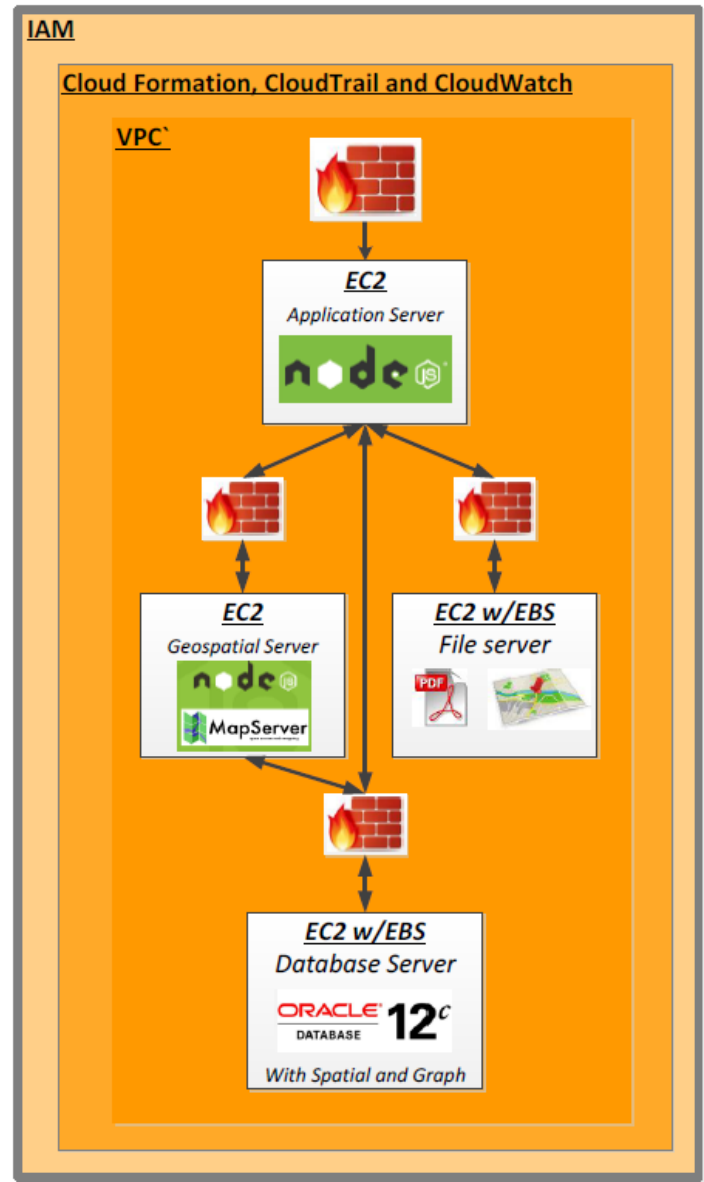
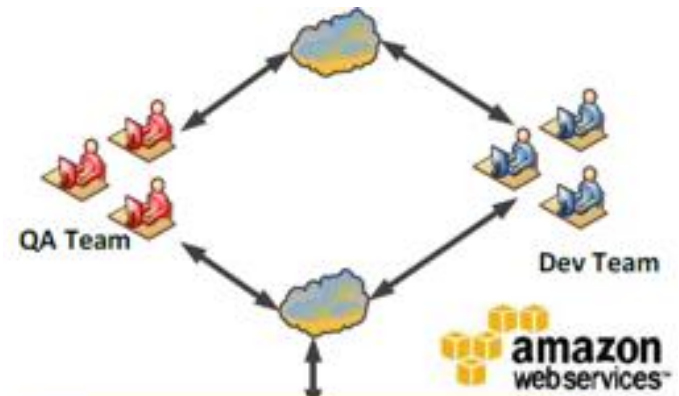


Continuous Integration and Build Automation

- Helps development team make system builds, triggered by either
 - A commit of updated source code to the Stash-Git version control system
 - Scheduling directive
 - A dependency on the completion of another component's build
 - Developer kicking off the build using a URL to make the request



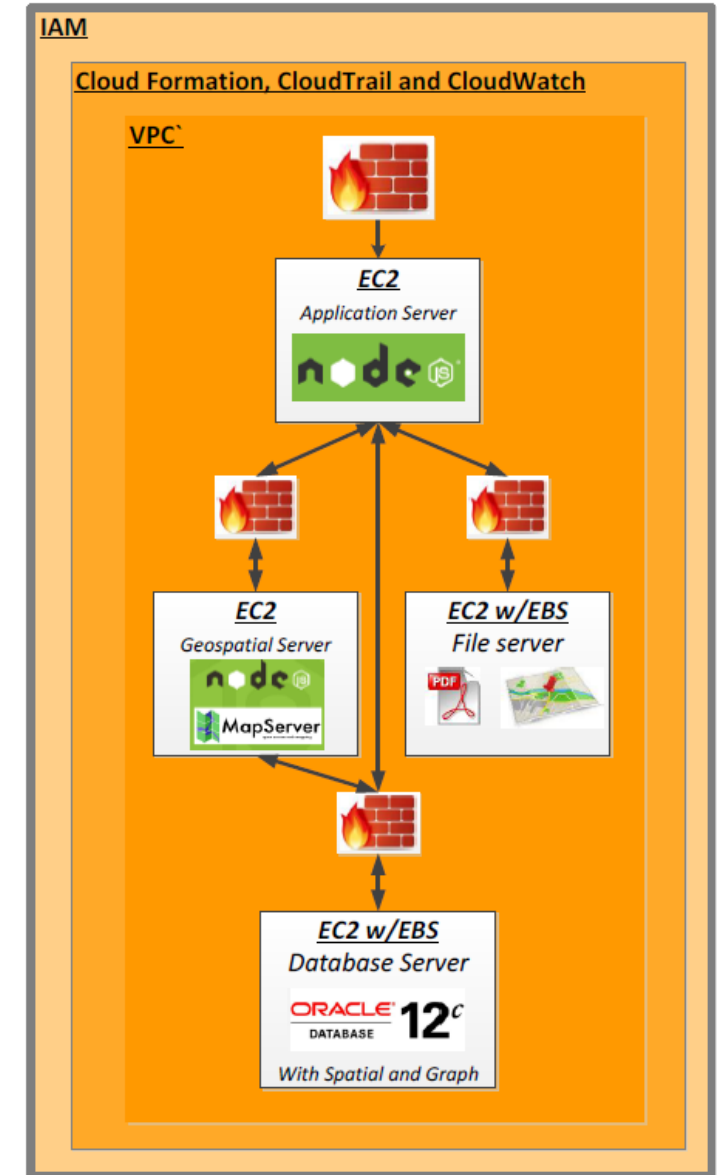
Development Infrastructure Example...



Development Infrastructure Example...

Amazon Elastic Compute Cloud (Amazon EC2)

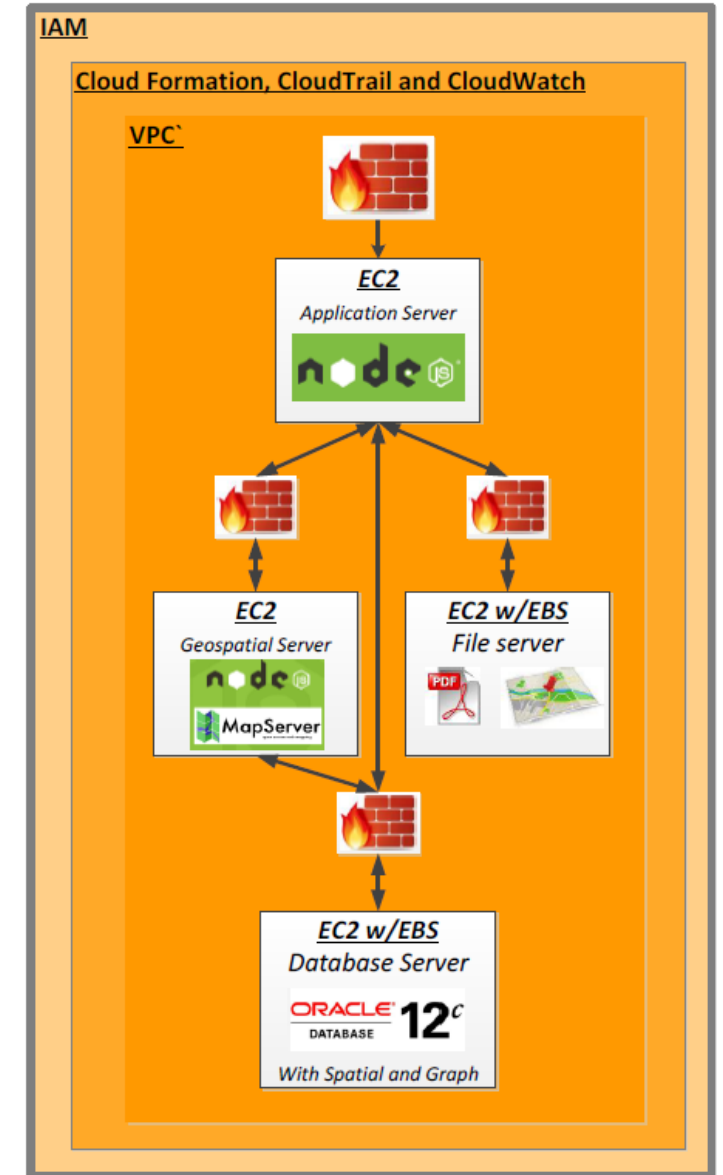
- Provides cloud-based computer servers for hosting web application and geospatial capabilities services
- EC2 provides flexibility in selecting among multiple server instance types, including multiple Linux and Windows server operating systems and supporting virtual hardware configurations of CPU, memory and disk storage
- It also will provide flexibility to scale the n-tier architecture with additional server resources to meet increased node.js application server capacity needs if and as they emerge. Amazon EC2 will be put to work in conjunction with the other AWS resources to support the application development project



Development Infrastructure Example...

Amazon Elastic Block Store (Amazon EBS)

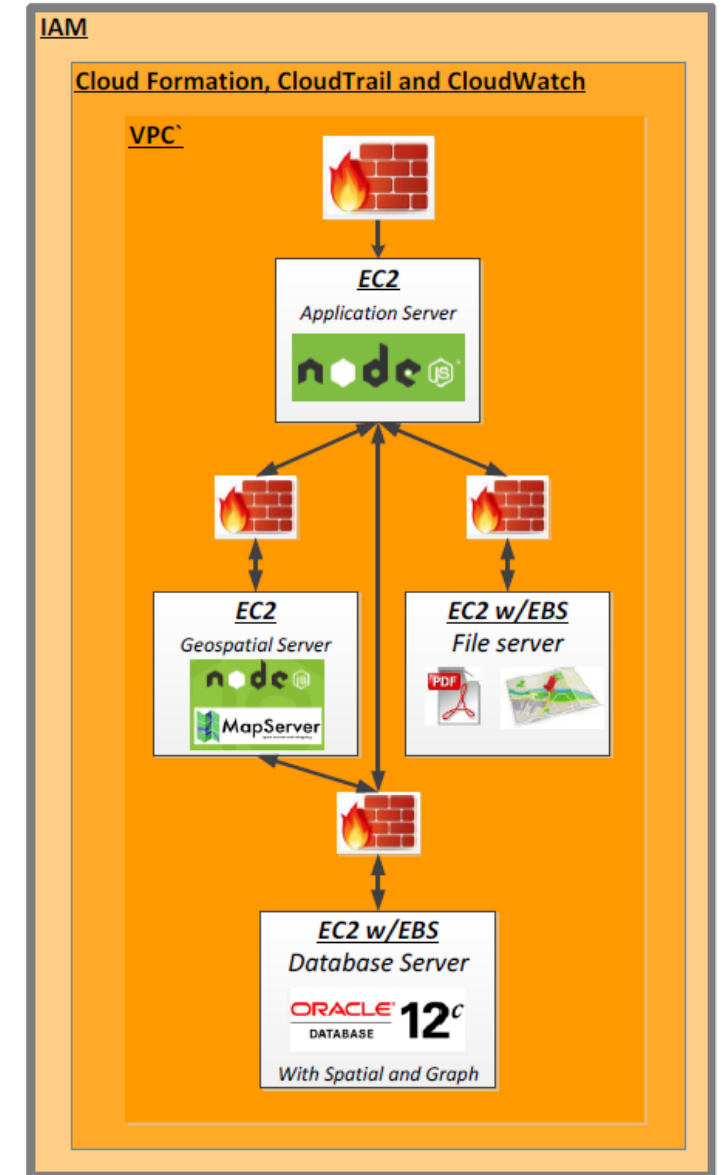
- Will combine with EC2 and serve as a file server to provide a responsive, secure, scalable place to store media (pictures and images) and pre-rendered static drill-down maps that are accessible over the web from pages of the client application
- Amazon EBS will also combine with EC2 for implement Oracle 12c Database with Spatial and Graph to serve as a secure, scalable database server to store:
 - Critical infrastructure data
 - User account and role-based authorization permission
 - Information on organizations and their data stewards



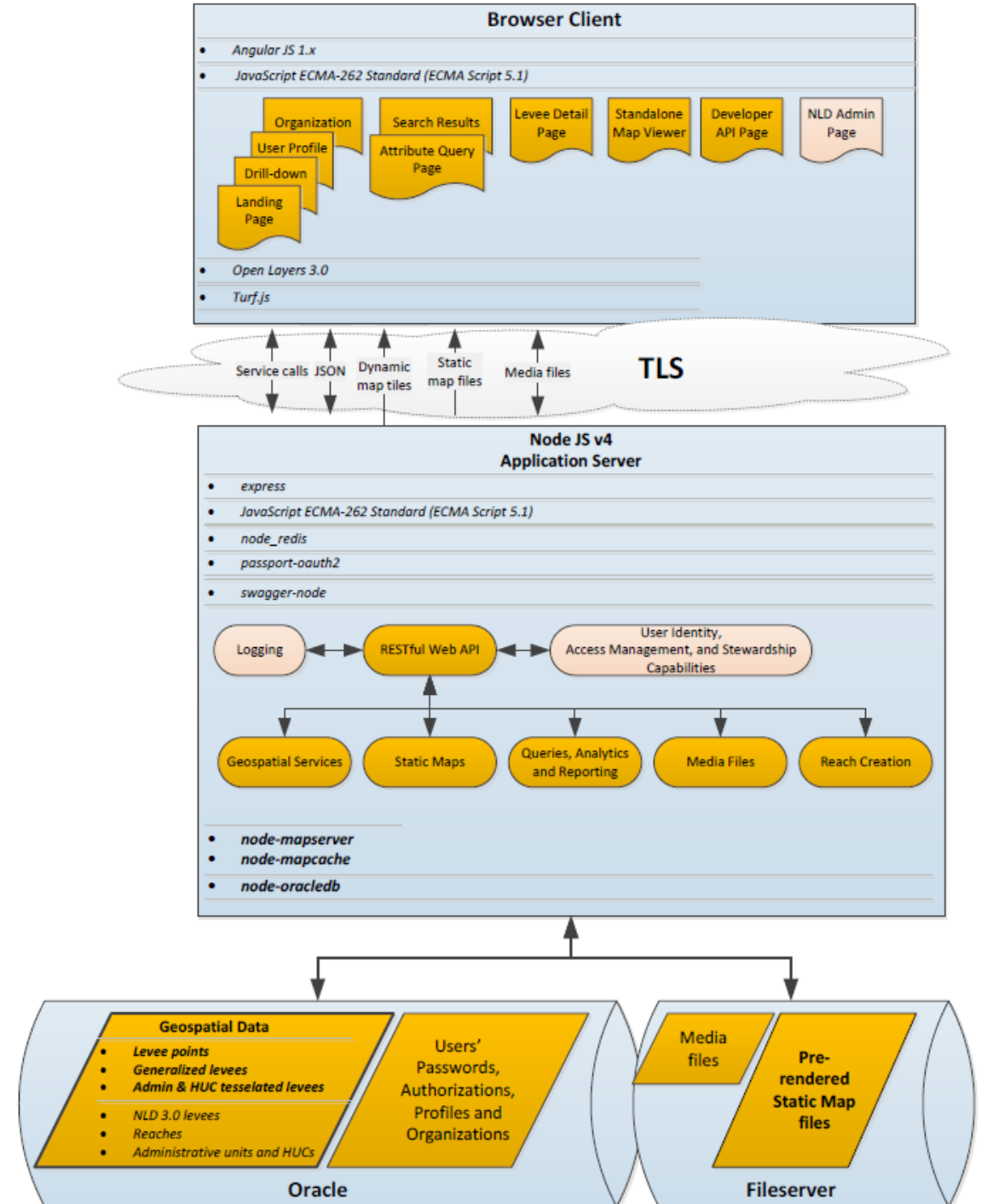
Development Infrastructure Example...

Amazon Elastic Block Store (Amazon EBS)

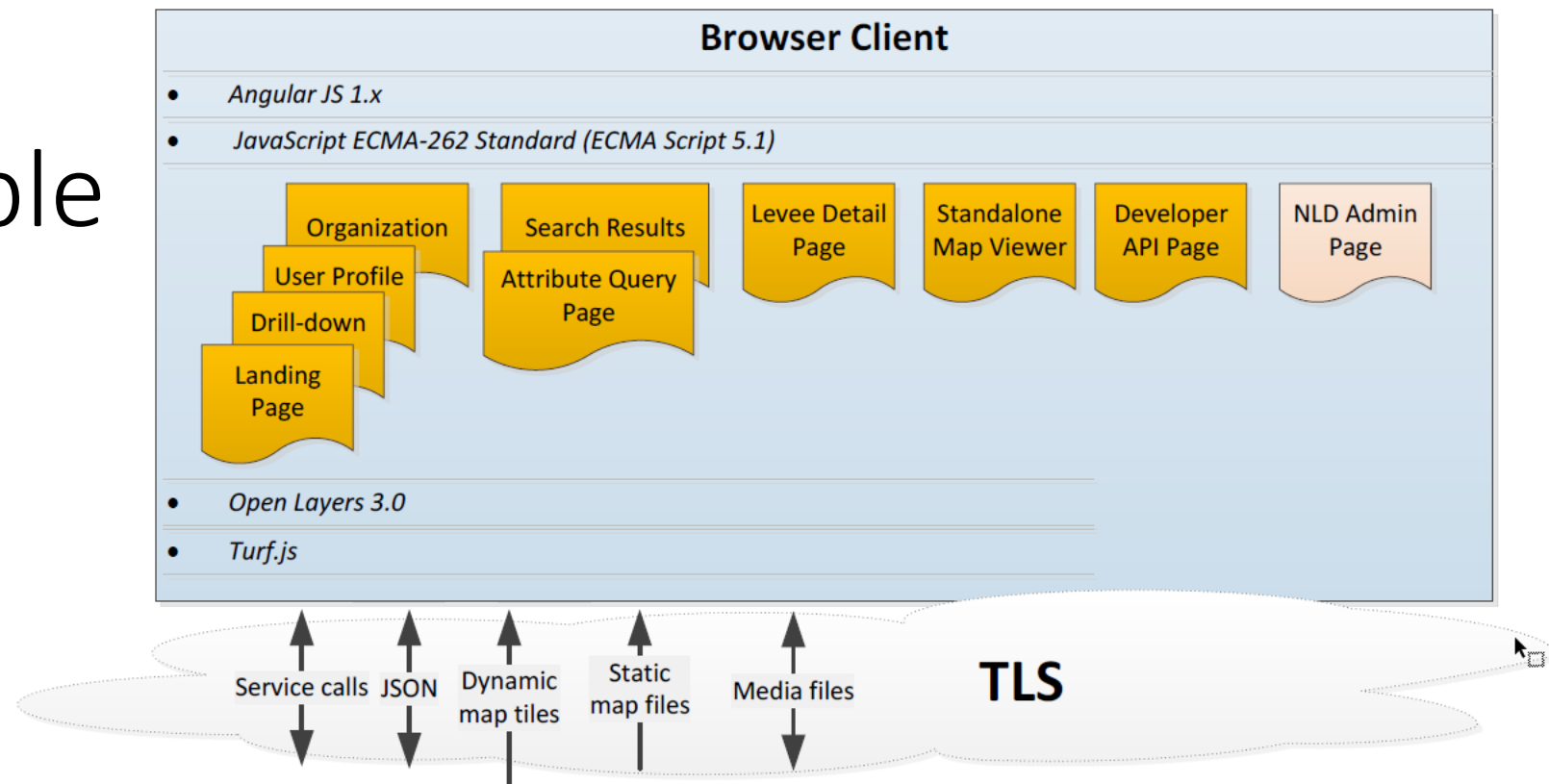
- Combines with EC2 and serve as a file server to provide a responsive, secure, scalable place to store media (pictures and images) and pre-rendered static drill-down maps that are accessible over the web from pages of the client application
- Amazon EBS will also combine with EC2 for implement Oracle 12c Database with Spatial and Graph to serve as a secure, scalable database server to store:
 - Critical infrastructure data
 - User account and role-based authorization permission
 - Information on organizations and their data stewards



Application 3+ Tier Architecture example



Application 3+ Tier Architecture example



The Presentation Tier runs in the client browser developed using the latest stable production release versions of:

- **Angular.js** open-source web client application development framework
- **Open Layers** open-source dynamic map display client library
- **Turf.js** web geographic information system
- **JavaScript** programming language

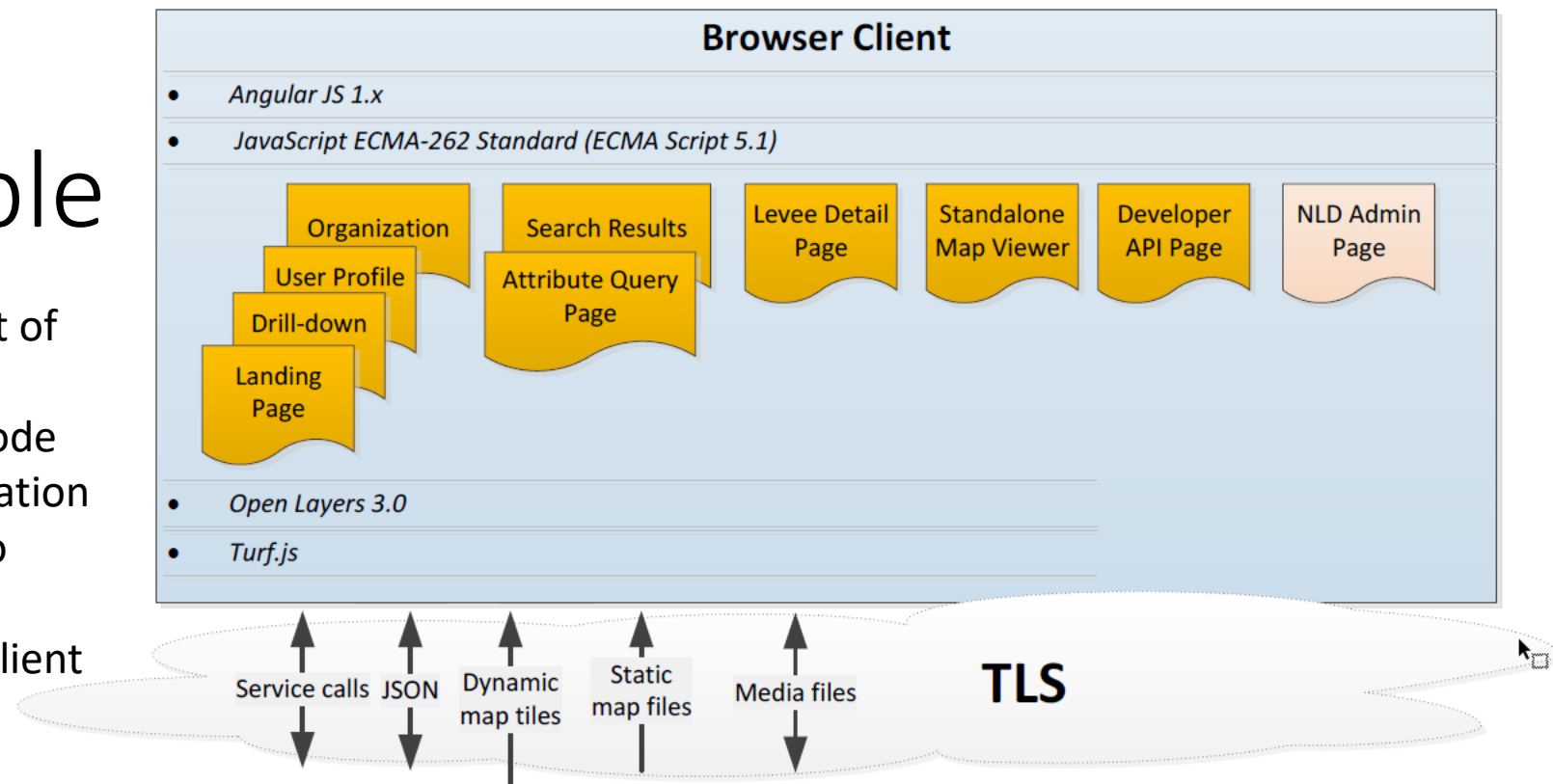
Application 3+ Tier Architecture example

Angular JS framework enables development of rich internet client application pages which support event-driven user interfaces, and code organization separating display and presentation logic from data objects and business logic to facilitate review and understanding

- Angular JS adds intelligence to the web client (“Fat Client”) which helps provide responsiveness to the user and reduces processing burden on the server

- **OpenLayers 3.0** open source JavaScript library enables displaying maps in web browsers and its API can be used to build rich web-based geographic applications similar to Google Maps and Bing Maps

- **Turf.js** provides native support for GeoJSON which enables exchange of data objects with the server



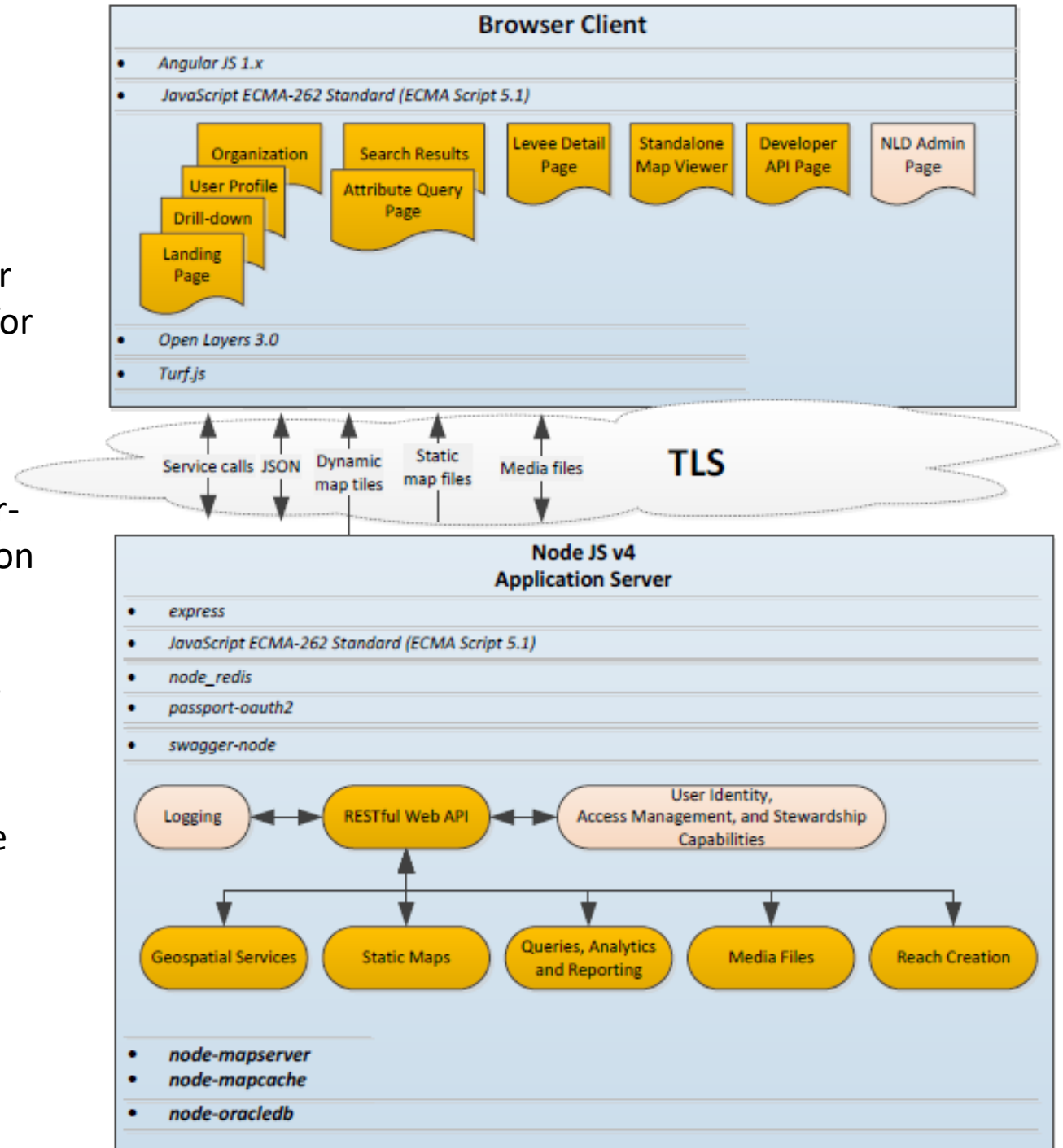
Application 3+ Tier Architecture example

Application Tier occupies the middle portion or “logic” tier and will be implemented in Node.js and provide support for the application’s functionality with detailed processing

Node.js provides an event-driven architecture and open-source runtime environment to support developing server-side web applications with an API that optimizes application throughput and scalability for real-time applications.

Node.js can be built with the TLS module (Transport Layer Security) to provide encrypted communication with the browser client

- This will require a public key encryption certificate signed by a Certificate Authority approved by the customer

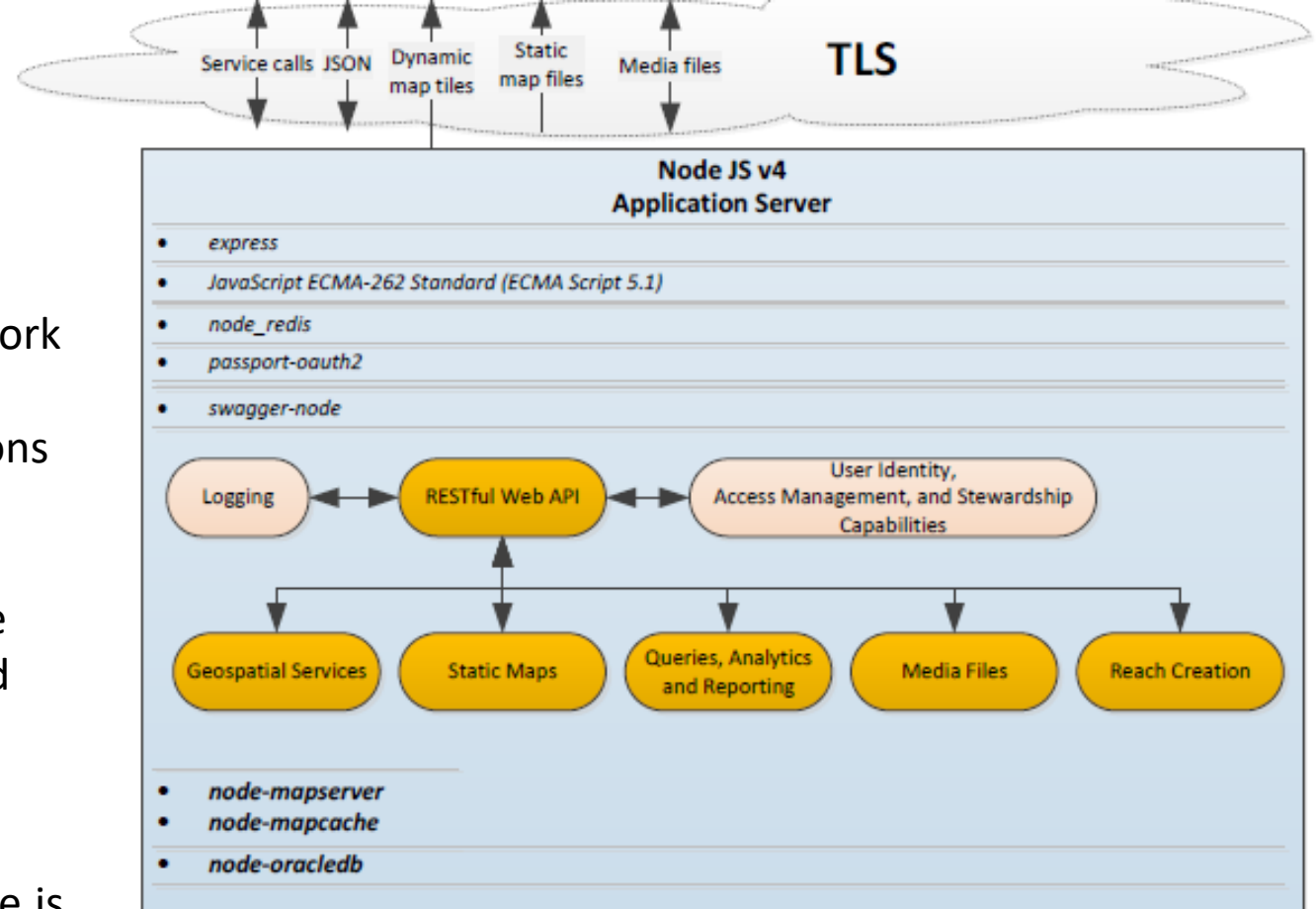


Application 3+ Tier Architecture example

Express.js will provide a light-weight application framework for handling connection requests to Node JS from the presentation tier and manage memory storage for sessions

Node_redis provides a fast, efficient, and flexible in-memory key-value store that behaves as a data structure server with keys containing strings, lists, sets, hashes and other data structures that can efficiently support the workings of a public facing application server

Passport.js is a flexible authentication capability for Node.js that provides authentication and session management functions for integration within web services exposed on the server-side and Angular pages and their GUI components on the client-side



Passport's OAuth2 strategy can provide token and cookie management capabilities for tracking application use and provide time out capabilities and control for keeping the application alive as long as specified

Application 3+ Tier Architecture example

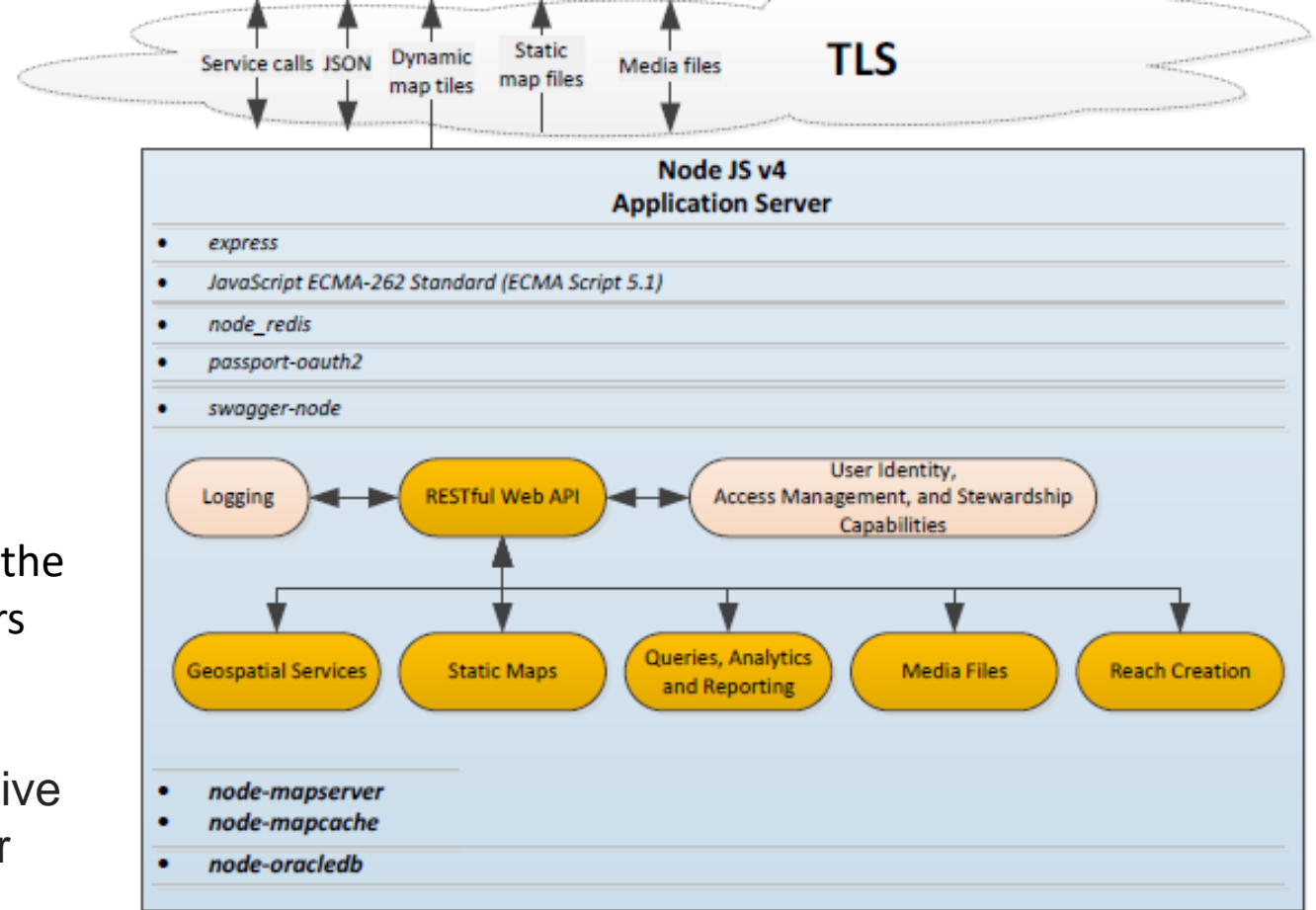
Swagger-node will provide a framework, documentation and test environment for the RESTful web service API

- It includes a front end user interface that will enable external developers to work with the application's API and gain a clear picture of how the API responds to requests with various parameters and options

node-mapserver library provides access to interactive mapping functionality of the open source MapServer common gateway interface through a JavaScript programming interface

node-mapcache provides a rapid map tile caching solution to provide an intermediary between MapServer (when needed) and Open Layers working in the presentation tier

node-oracledb add-on will serve as a node.js driver for connecting to and interacting with the Oracle database



Agenda

- ✓ Distributed Systems
 - ✓ File Server Architecture
 - ✓ Client/Server Architecture
 - ✓ N-Tier Architecture
 - ✓ Thick versus Thin clients
 - ✓ Cloud Architecture
 - ✓ Service Oriented Architecture (SOA)
- ✓ Example Cloud-based N-Tier SOA Application Development System
- **Control Stages and Objectives**


Information System Development Control Stages

Control over applications is conducted at every stage and begins at the start of the development of the information system

This takes 2 basic forms:

1. Control over the development process itself
2. Ensuring adequate business controls are built into the finished product

Major control stages would include:

- System design
- System development 
- System operation
- System utilization

Control Objectives for Business Information Systems

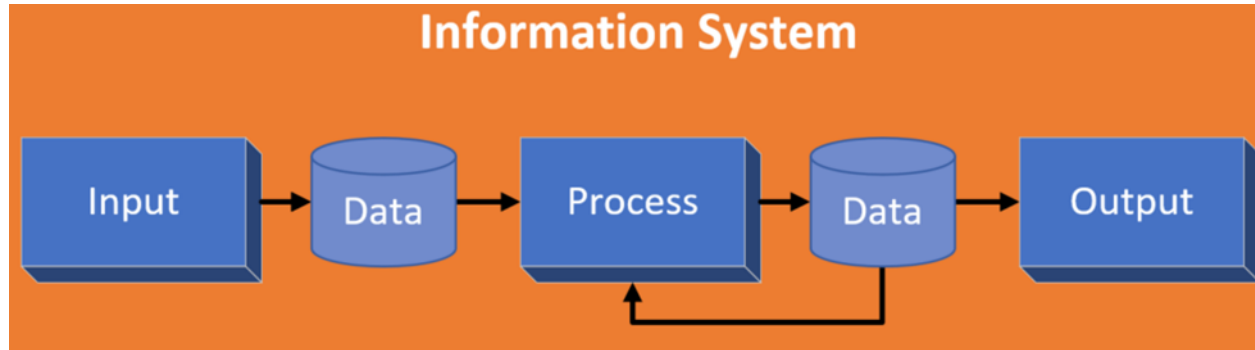
- Confidentiality
- Integrity
 - Accuracy
 - Completeness
 - Validity
- Availability

Control Objectives for Business Information Systems

Different system types may have additional control objectives and differing priorities within general control objectives, e.g.

- Order processing
- Invoicing
- Inventory control
- Accounts receivable
- Accounts payable
- Purchasing
- Shipping
- Receiving Payroll
- General ledger
- Specialized systems
- Banking systems
- Retail systems
- Manufacturing systems
- Electronic Data Interchange (EDI)

Control Objectives for Business Information Systems

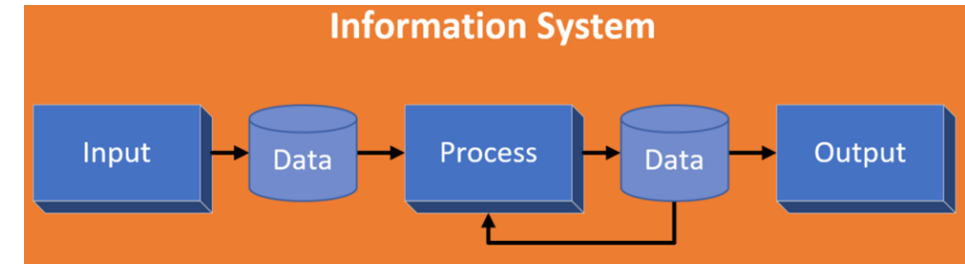


- Input control objectives
- Processing control objectives
- Output control objectives

Control Objectives for Business Information Systems

Input control objectives

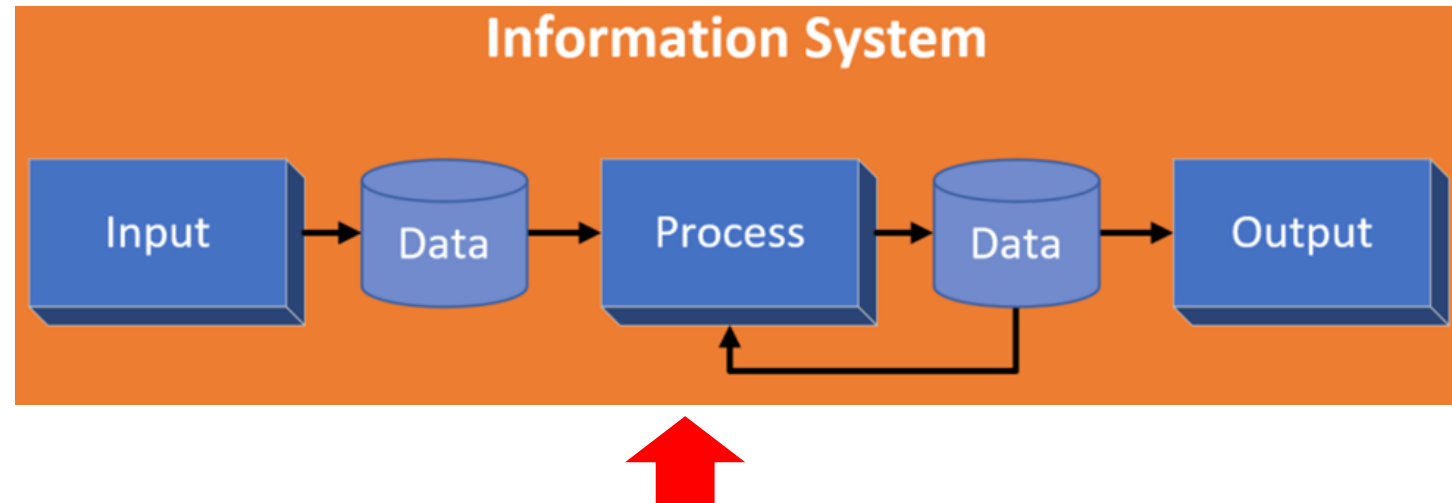
- All transactions are
 - initially and completely recorded
 - completely and accurately entered into the system
 - entered only once
- Controls in this area may include:
 - Pre-numbered documents
 - Control total reconciliation
 - Data validation
 - Activity logging
 - Document scanning and retention for checking
 - Access authorization
 - Document cancellation (e.g. after entry)



Control Objectives for Business Information Systems

Processing control objectives

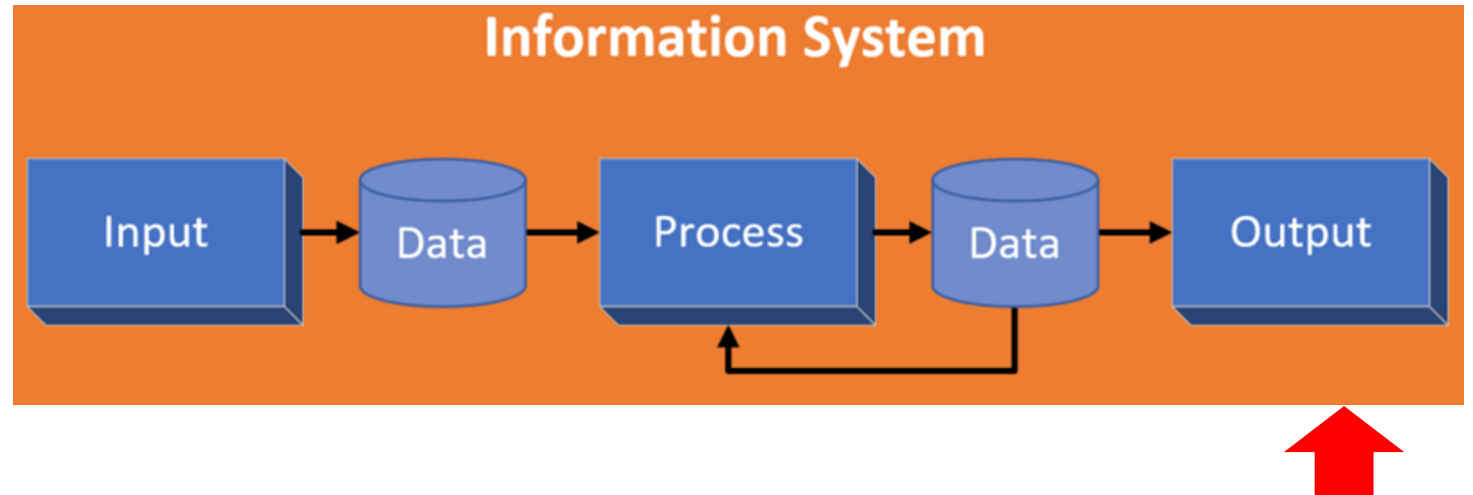
- Approved transactions are accepted by the system and processed
 - All rejected transactions are reported, corrected, and re-input
 - All accepted transactions are processed only once
 - All transactions are accurately processed
 - All transactions are completely processed
- Controls over processing may include:
 - Control totals
 - Programmed balancing
 - Segregation of duties
 - Restricted access
 - File labels
 - Exception reports
 - Error logs
 - Reasonableness tests
 - Concurrent update control



Control Objectives for Business Information Systems

Output control objectives focus on

- Hardcopy
- File outputs and output record sets stored in tables
- Online query files and outputs stored in tables
- Controls over output may include:
 - Assurance that the results of input and processing are output
 - Output is available to only authorized personnel
 - Complete audit trail
 - Output distribution logs



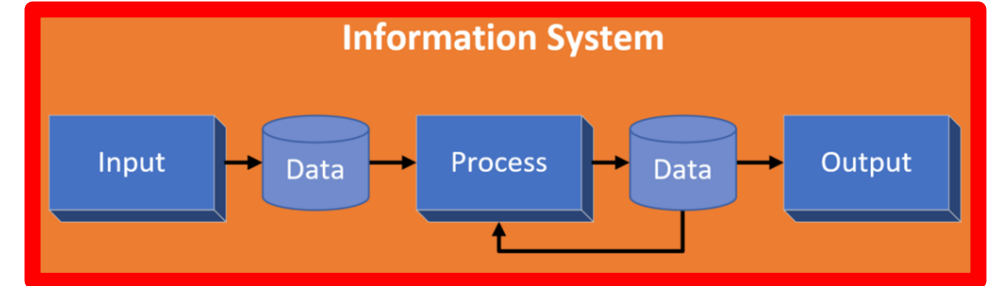
Control Objectives for Business Information Systems

Computer program control objectives focus on

- Integrity of programs and processing
- Prevention of unwanted changes

Typical computer program controls include:

- Ensuring adequate design and development
- Ensuring adequate testing
- Controlled transfer of programs (among machines, from version control, ...)
- Ongoing maintainability of systems
- Use of formal SDLC
- User involvement
- Adequate documentation
- Formalized testing plan
- Planned conversion
- Use of post-implementation reviews (see CISA chapter)
- Establishment of a quality assurance (QA) function
- Involvement of internal auditors



Testing of these controls require auditors to seek evidence regarding their adequacy and effectiveness....

Agenda

- ✓ Distributed Systems
 - ✓ File Server Architecture
 - ✓ Client/Server Architecture
 - ✓ N-Tier Architecture
 - ✓ Thick versus Thin clients
 - ✓ Cloud Architecture
 - ✓ Service Oriented Architecture (SOA)
- ✓ Example Cloud-based N-Tier SOA Application Development System
- ✓ Control Stages and Objectives