Migration Discussion

Presented by:

IBM

IBM Cloud

Agenda

- Why would you Migrate or Renovate Enterprise Apps?
- What are the issues making Java Apps cloud-ready?
- What Apps should be migrated?
- What Cloud services to choose while renovating?

Why Renovate?

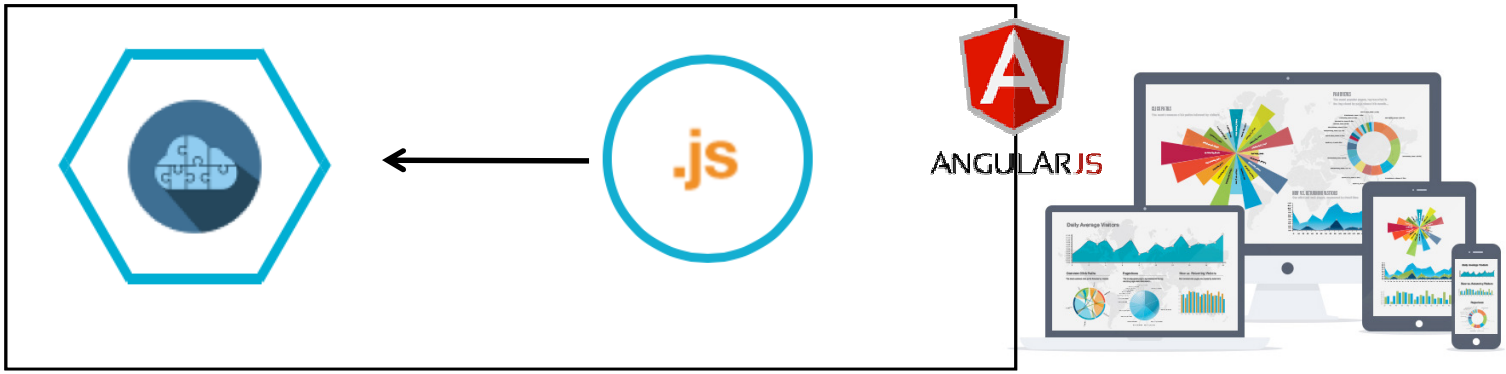
Why renovate or migrate?

Three reasons :

- "Front-end renovation" : Building a new UI.
- Lot of options in Bluemix for front-end development.
- "Full renovation" : Taking advantage of Modern programming, Scaling & Services in Bluemix.

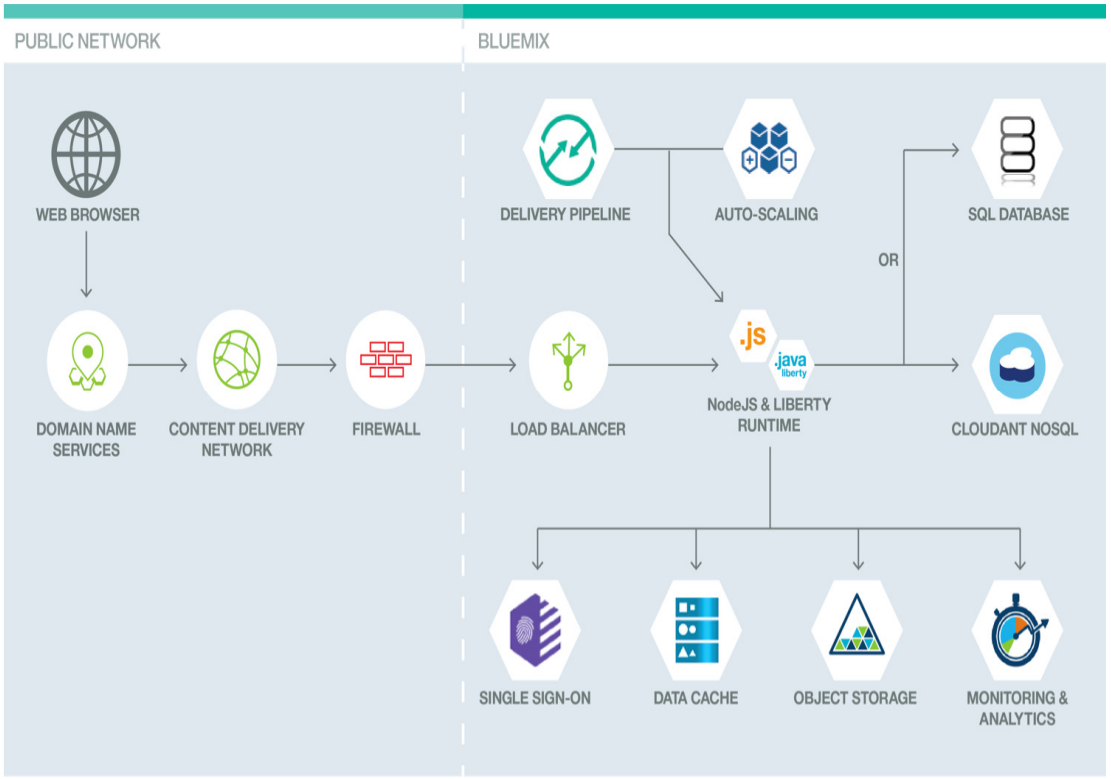
Front-End Renovation

- Building a Mobile App (Hybrid or Native) using Mobile features
- Building a modern Web App using Angular and Node
- Exposing a public API using Bluemix for external clients
- How ? Use Design Thinking : don't take existing web page



Full Renovation

- Benefits:
  - Migrating to newer JEE features
  - Modernizing caching infrastructure
  - Modernizing database infrastructure
  - Modernizing monitoring infrastructure
  - Using APIs instead of directly connecting to back-end services
  - Taking advantage of a DevOps Pipeline



https://developer.ibm.com/bluemix/docs/actionable-architecture-building-web-application-hosting-cloud-foundry/

Considerations for Cloud Readiness

What does moving to Cloud mean?

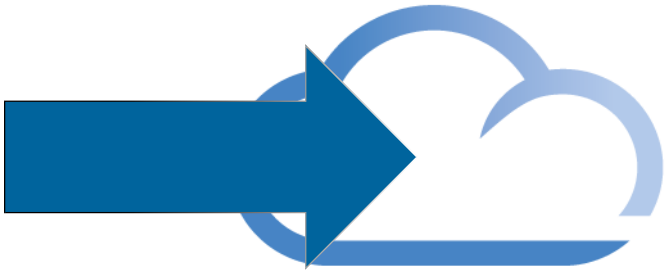
- **Cloud-ready** App can be deployed into public or private cloud
  - Certain architectural decisions can make Apps not cloud-ready

- **Cloud-centric** Apps are written to run on cloud
  - Built using tools and runtimes different than traditional apps
  - For example, instead of relational database use NoSQL database
  - Different infrastructure dependencies

The decision on whether you need an IaaS or PaaS is application dependent

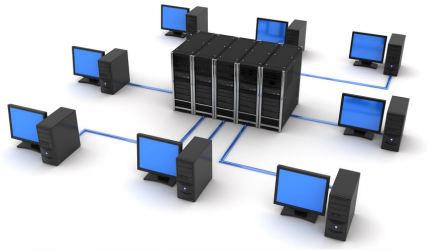
Common characteristics of Enterprise Applications

- Not built with Cloud compatibility in mind
  - Trying to catch up to latest versions of libraries, etc.
- May be built on WebSphere Stack products
  - Portals with WebSphere Portal Server
  - Departmental ESB's built with DataPower and Integration Bus
- Deep ties into internal IT
  - Update systems of record
  - Update multiple systems not specified as API's
  - Deep ties into Corporate security systems (LDAP, ISAM, Siteminder) and standards
- High and well-specified QoS attributes
  - Expectations on how often they are available and how they perform



You have to consider all of the business, architectural and design decisions of an application when moving to Cloud.

Differing Assumptions



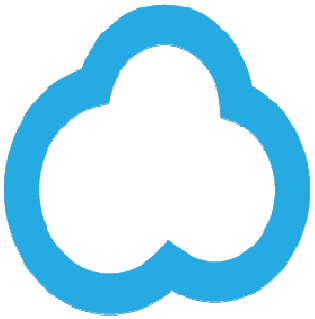
- Enterprise (Cloud Ready) Assumptions
  - Infrastructure is stable.
  - Components of my application are co-located.
  - Operations team controls production servers.
  - If a disaster happens, it's someone's responsibility.

- Cloud Centric Assumptions
  - Infrastructure is constantly changing (elastic).
  - Components of my application may be globally distributed.
  - As a Dev/Ops team member, I control production servers.
  - If a disaster happens, it's my responsibility to fix it

Choosing one or the other has an effect on your team composition and roles

Making Applications Cloud Ready

Criteria for cloud-ready applications

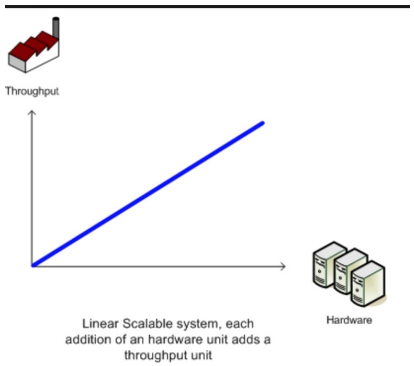


1. Application's design is topology-agnostic
   – Ex: Clustering is supported, no specific cluster size needed
2. Application's management is infrastructure-agnostic
   – Ex: Doesn't depend on IP addresses, hostnames, or VLANs
3. Application doesn't use infrastructure-specific APIs
4. Application doesn't use OS-specific features
5. Application doesn't use local file system
6. Application logs to persistent storage, not the file system
7. Application keeps session state only in the interaction layer
8. Application components connect via standard protocols
9. Application's installation and configuration is scripted
   – Ex: Deployment is easily repeatable

Additional considerations to port apps to Bluemix

Application Considerations:

- On a recent, supported level of the runtime
- Have linear, horizontal scaling
- Shouldn't hog bandwidth
- Not need data at rest to be encrypted
  - Limits options for data sources

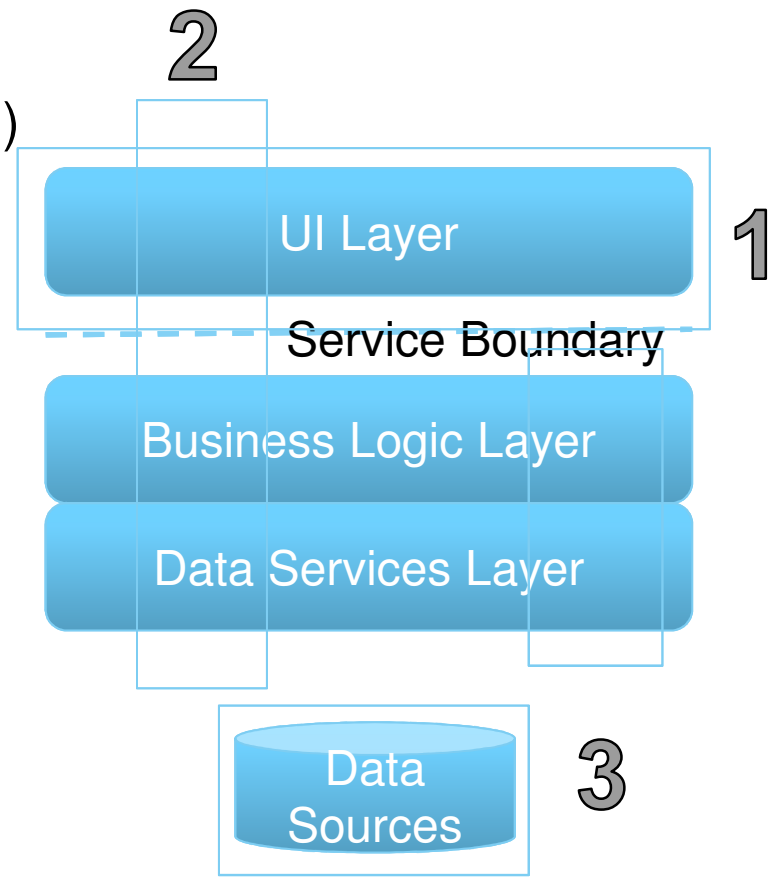


Infrastructure Modernization for Java applications

- Plan on moving to WAS Liberty
- Code refactoring
  - –Moving to EJB 3.0 or JEE Web Service Profile
  - –Understanding dependencies on scripts, internal messaging, telecom classes etc.
- Formulate Liberty migration strategy
  - –Use Migration toolkit
  - –Classify and evaluate applications for migration
  - –Classification criteria (# users, uptime, #data sources, Tech stack)

Changing your applications

- App has to change in cloud migration
- Need a good set of
  - Application tests (preferably automated)
  - Application performance baselines
  - Process to access test data
- Application Refactoring
  - Breaking down applications by layer
  - Breaking down applications along functional boundaries
  - Breaking larger apps into smaller ones
  - Breaking out services
    - Developing shared services architecture



Paying Technical Debt

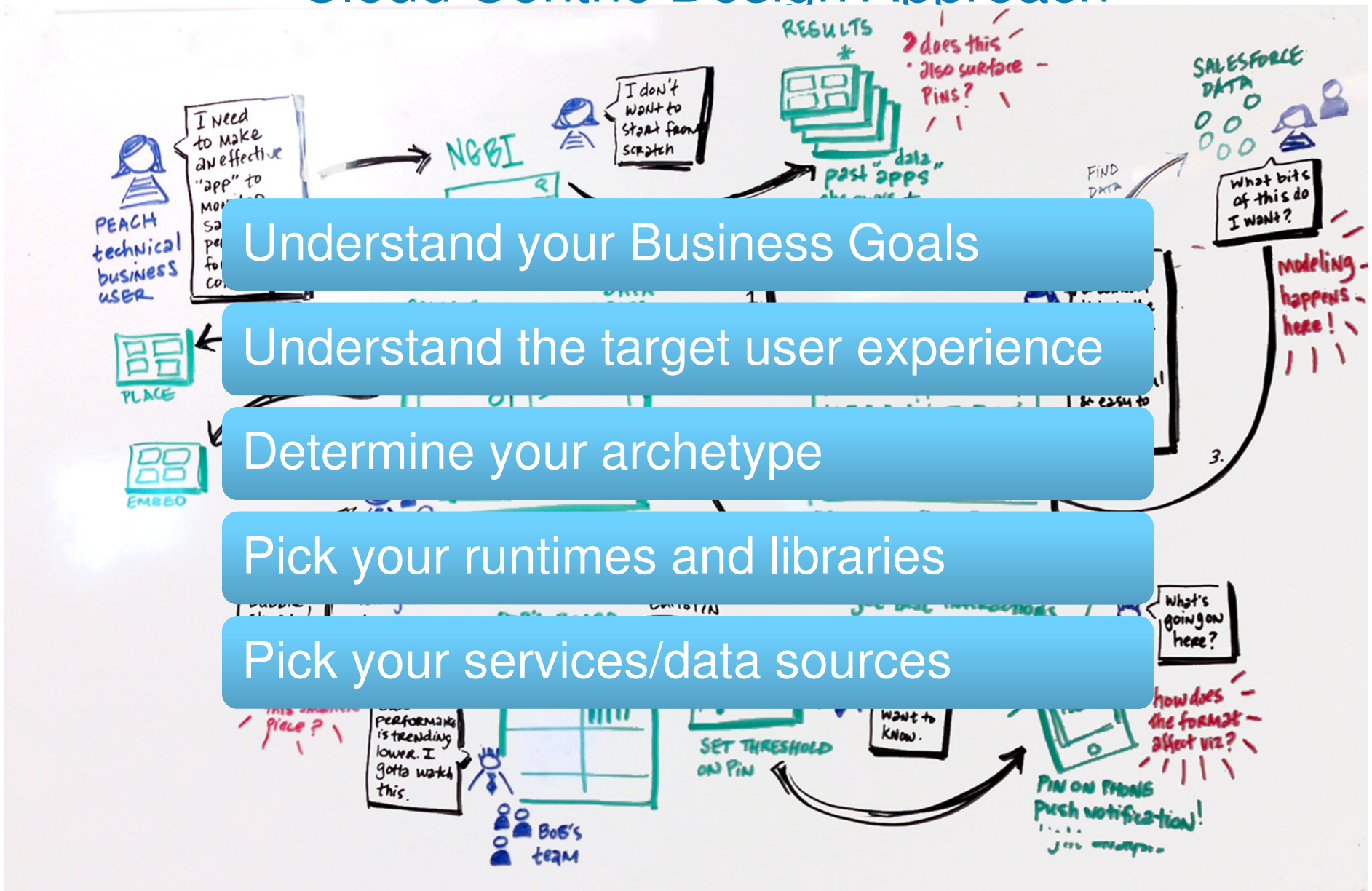
- Fully automated functional and performance test suite
- Fully automated code and configuration deployment sch
- Clean up unneeded internal dependencies
- Refactor to classical SOA or *Microservices*
- Modernize your services infrastructure
  - Add caching, monitoring, etc…
- Modernize your user interface
  - Come up to date on framework versions
  - Move to Web 2.0/3.0 Hybrid model or Native mobile apps

Rebuilding Apps for Cloud Centricity

Design Principles for Bluemix Applications

- Bluemix applications are “Born on the Cloud” :
  - –Built using **Microservices** architectures
  - –Follow DevOps principles
  - –Built in accordance with the 12-factor.net rules
- Born on the Cloud applications
  - –Have shorter lifetime– new versions rolled out very quickly
  - –Built using principles of polyglot programming and polyglot persistence

Cloud Centric Design Approach



Understand the target user experience

- Use IBM Design Thinking
  - Use IBM Design team to conduct Design Thinking Workshops
- Core concepts (Hills, Sponsor Users, Playbacks) provides overall approach
- Artifacts like Empathy Maps, Scenario Maps and Wireframes provide *important information*
  - *Leads to implementation* in Cloud environment

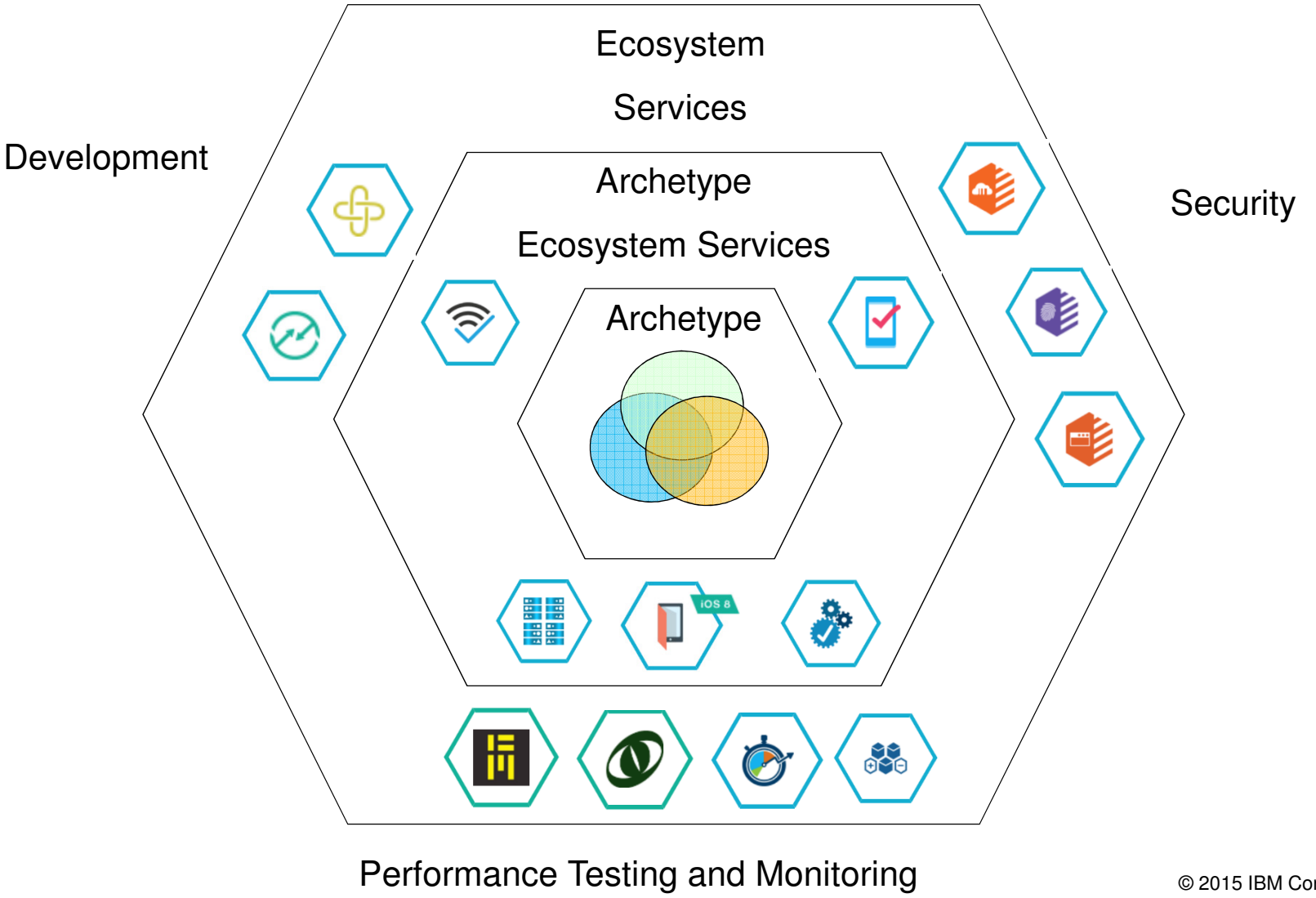
How do you get started?

- Three different **archetypes** :
  - Systems of Record
  - Systems of Interaction
  - Systems of Insight
- [brackets] indicates 0-N of these may be needed
  - System of Interaction Archetype
    - [Gateway] + Runtime + Data Store + [Data Movement]
  - System of Insight Archetype
    - [Visualization] + Runtime + Data Store + [Analysis Tool]
  - System of Record Archetype
    - [Gateway] + Runtime + Data Store
- Two different layers of “Ecosystem Services” ca be added for functional and non-functional requirements

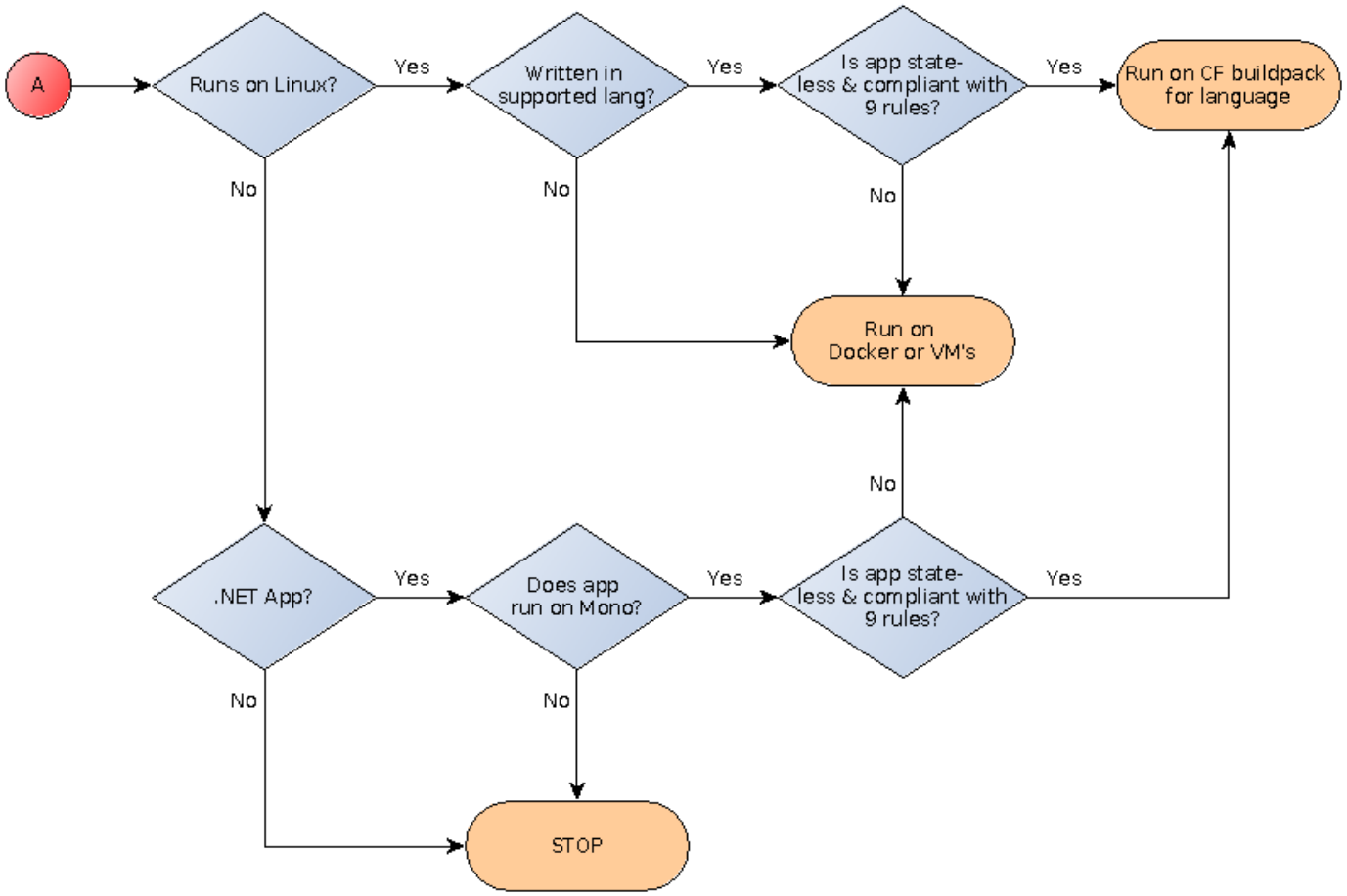
Core Services Categorization

Data Storage	Gateway	Visualization	Analysis	Data Movement, Augmentation and Validation
SQL Options	IOT	Embeddable Reporting	Analytics for Hadoop	Dataworks
NoSQL Options	Twilio	Cognitive Graph	Apache Spark	Pitney Bowes Geocoding
	Secure Gateway	IOT Realtime Insights	BigInsights for Hadoop	Watson Natural Language Translation
	API Management	Document Generation	DashDB	
	SendGrid		Geospacial Analytics	
			Streaming Analytics	
			Predictive Modeling	

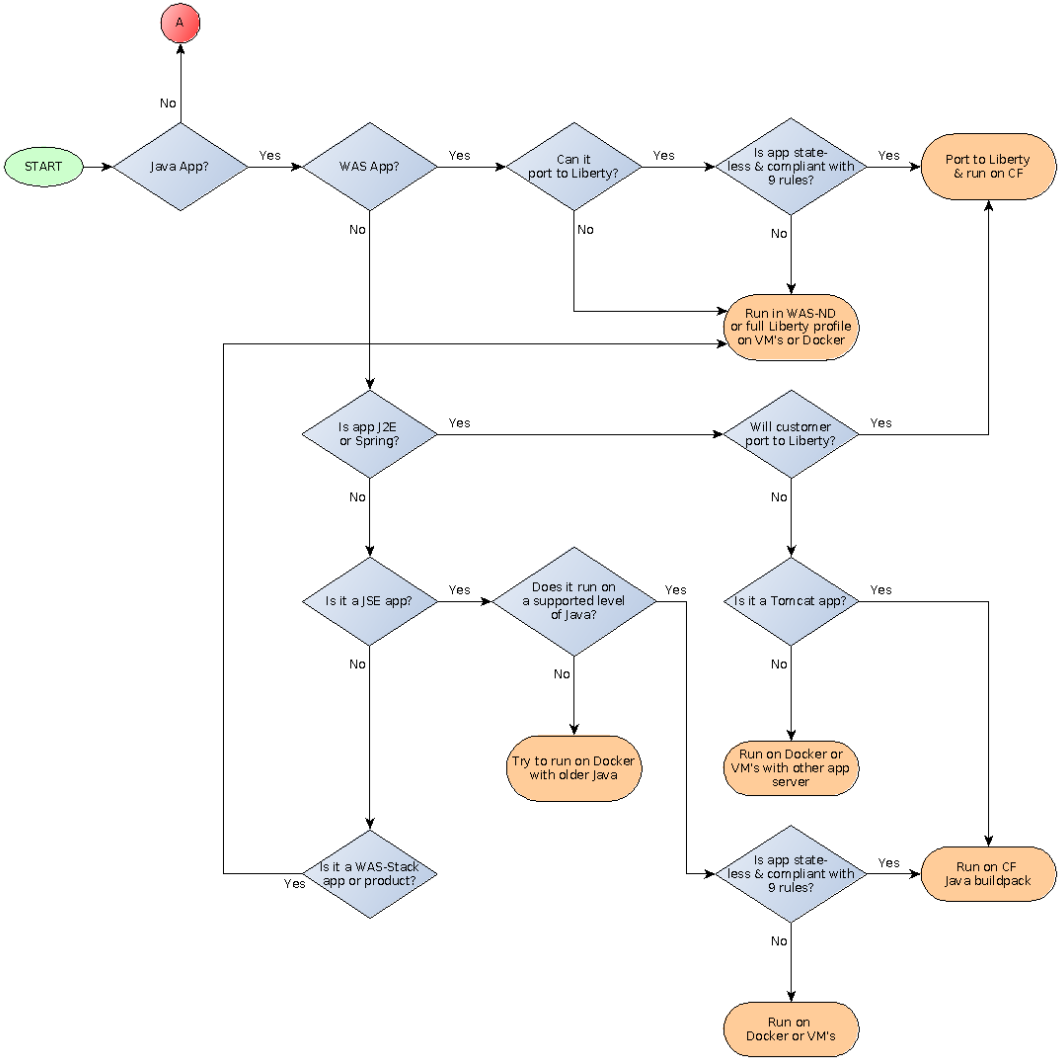
Layers of a Bluemix Application



Simplified Migration Flowchart



More complete Migration Options



	Bluemix Cloud Foundry Runtimes	IBM Container Service	HEAT Patterns or PureApp Services Patterns
Optimal System Type	Systems of Engagement	Either	Systems of Record
Optimal Application Type	Cloud Centric	Either, but most commonly Cloud Centric	Cloud Ready
Scaling	Horizontally scaled single-process solutions (Vertical scaling limited by 4GB limit on DEAs)	High density vertically scaled single-process solutions	Horizontally scaled complex multi-process assemblies,
Composition Model	Service Composition model	Runtime assembly model compatible with Service composition	Runtime assembly model for complex multi-runtime systems
Cloud solution type	Public or managed private cloud solution	Public, Private or managed cloud solutions and local development	Private cloud solutions
DevOps Integration	Integrated with DevOps Pipeline	Integrated with DevOps Pipeline (Bluemix public) and Integrated with UCD	Integrated with UCD
Security	Limited security options	Not yet explored	Corporate Grade security options
Middleware coverage	Limited traditional middleware functionality	Limited but growing WAS and stack options	Full set of WAS stack products and partner products available

Choosing Core Services

Polyglot Persistence

- A basic assumption is that your application will probably have to store some data persistently (e.g. for longer than the length of a single user session)
- At one time, it was a simple assumption that Web programs would run on a relational database
  - Multiple relational database options exist for Bluemix
- However the assumption of using a SQL database (DB2, MySQL, etc.) should be challenged by the set of open-source data options collectively called “NoSQL”
  - These data stores are characterized into four different basic types that are each specifically qualified for different purposes

NoSQL Options

- Cloudant – High Performance JSON document Database built on Apache CouchDB. Built in support for map/reduce. Useful in nearly any NoSQL situation.
- Redis by Compose – BSD-licensed Key-value store. Especially useful for leaderboards and document ranking.
- Mongodb by Compose – open-source document DB (documents similar to JSON objects). Often used for log data, product data management and content management.
- RedisCloud and MongoLab also provide third-party Redis and Mongo hosting services.
- Graph Data Store - built on Apache Tinkerpop and is especially useful for modeling social networks.

SQL Database Options

The SQL options for storing data vary by functionality and the amount of scaling and multi-tenancy that they support

- The SQL Database service provides a DB2 Database (not a DB2 Instance) that is unique to a space and suitable for low-volume applications
- The DB2 on Cloud Service provides a DB2 Instance that is unique to a space and can scale up to Enterprise sizes

Both DB2 options are fully compatible with existing DB2 drivers and SQL syntax

- PostgreSQL on Compose and Elephant SQL (third party) provide implementations of Postgres.
- ClearDB and the (experimental) MySQL Service both provide implementations of MySQL

In the end, the decision is usually based on what database the team (and their DBA's) are most comfortable with.

Storing Data Temporarily

For performance reasons (or for scalability of session-type data) you may find the need to store data for the duration of a user session

There's also the need to cache commonly accessed data rather than pull it from a database, especially a relational database.

- The Session Cache service is specific to the Liberty buildpack and implements the JEE Session API.
- The Data Cache service implements a highly available key-value cache with API's for Java, Node and REST
- The Memcached service hosts a distributed key-value cache using the widely available Memcached API's

These should not be used as systems of record – they are explicitly unreliable and may suffer from any of the standard problems (staleness, etc) that distributed caches suffer from. A Cache is not a database!

Communicating between services asynchronously

Sometimes there is the need to add asynchronicity between stages of a business application. For instance, commonly a web site will need immediate response, while processing an order could be an operation that takes hours or days.

Asynchronous messaging systems allow different services to run at different speeds

- MQLight is a fully-featured AMQP-based messaging engine with clients for Java, Node, Ruby and Python
- CloudAMQP hosts the open-source RabbitMQ AMQP-based messaging engine
- There is also an experimental RabbitMQ Service available from Bluemix Labs.

None of these solutions is capable of being an Enterprise messaging solution or tying to those – you still need to use MQ Series and the Secure Gateway for that.

Communicate with the world outside Bluemix (Gateways)

- A Gateway allows you to bridge between code running inside a Bluemix runtime and people and services that exist outside the Bluemix environment.
  - Twilio allows you to communicate via Voice and SMS
  - SendGrid allows you to communicate via Email
  - IOT allows you to send and receive MQTT messages through the Internet Of Things Foundation
  - API Management allows you to create Bluemix services from existing API's in your enterprise and to manage those API's
  - Secure Gateway allows you to connect to resources behind your corporate firewall
- While Twilio and SendGrid normally act as *Data Sinks* the Secure Gateway may support limited cases of bidirectional communication

Validating, Cleansing and Converting Data

- The Pitney-Bowes Geocoding and Reverse Geocoding services map GPS coordinates to postal addresses, and vice versa
- DataWorks allows you to move data from one database to another and also validates postal addresses
- Watson Language Translation converts text between several natural languages
- Presence Insights helps you Geolocate a user's device within a venue to provide them with location-specific information
- Geospacial analytics gives you MQTT notification when a device moves within a bounded area

Think of these as pipes or filters in a data pathway. Either you're moving data from one place to another, or changing, converting or augmenting that data