SPRINGONE2GX
WASHINGTON, DC

Migrating the Monolith

Rohit Kelapure

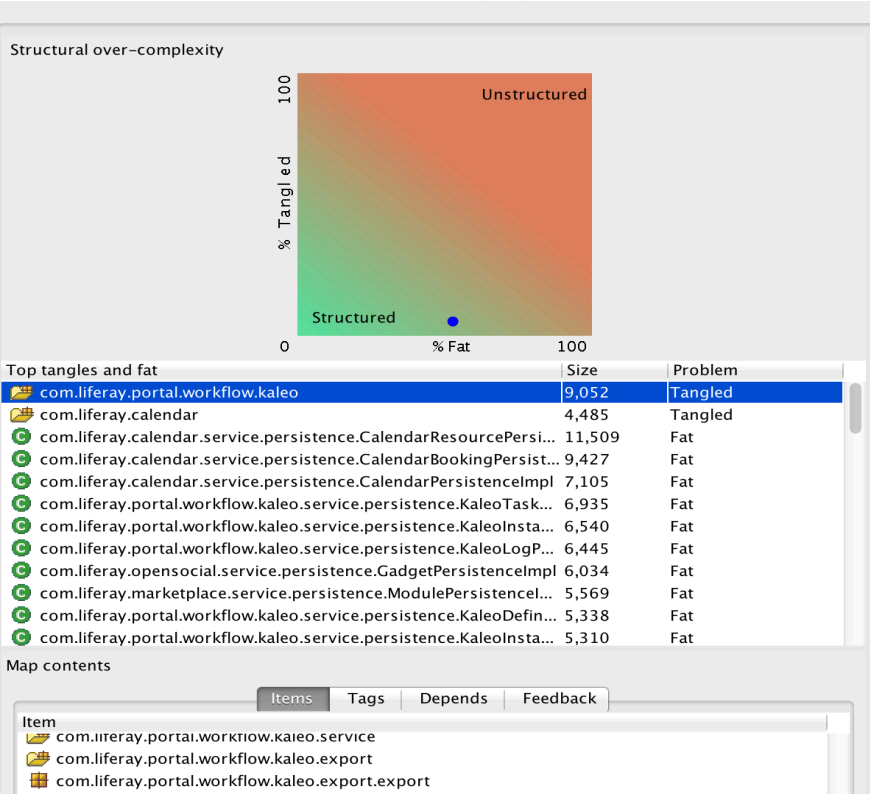
@rkela

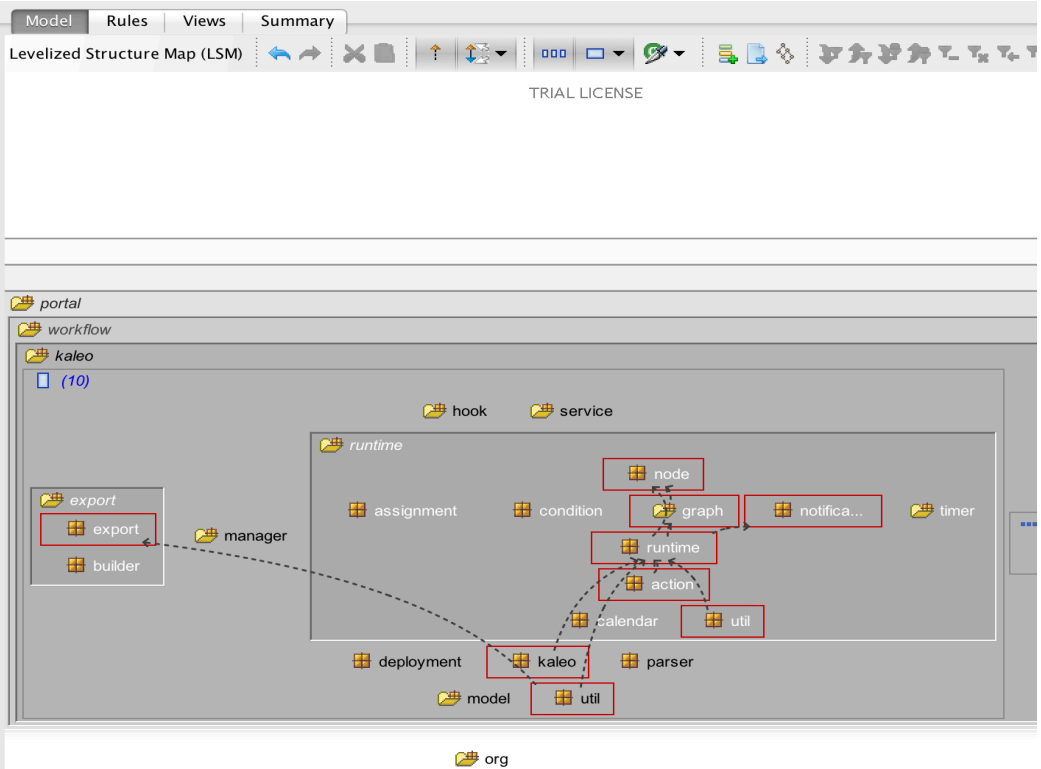


springone 2GX

Monolith – Big Ball Of Mud

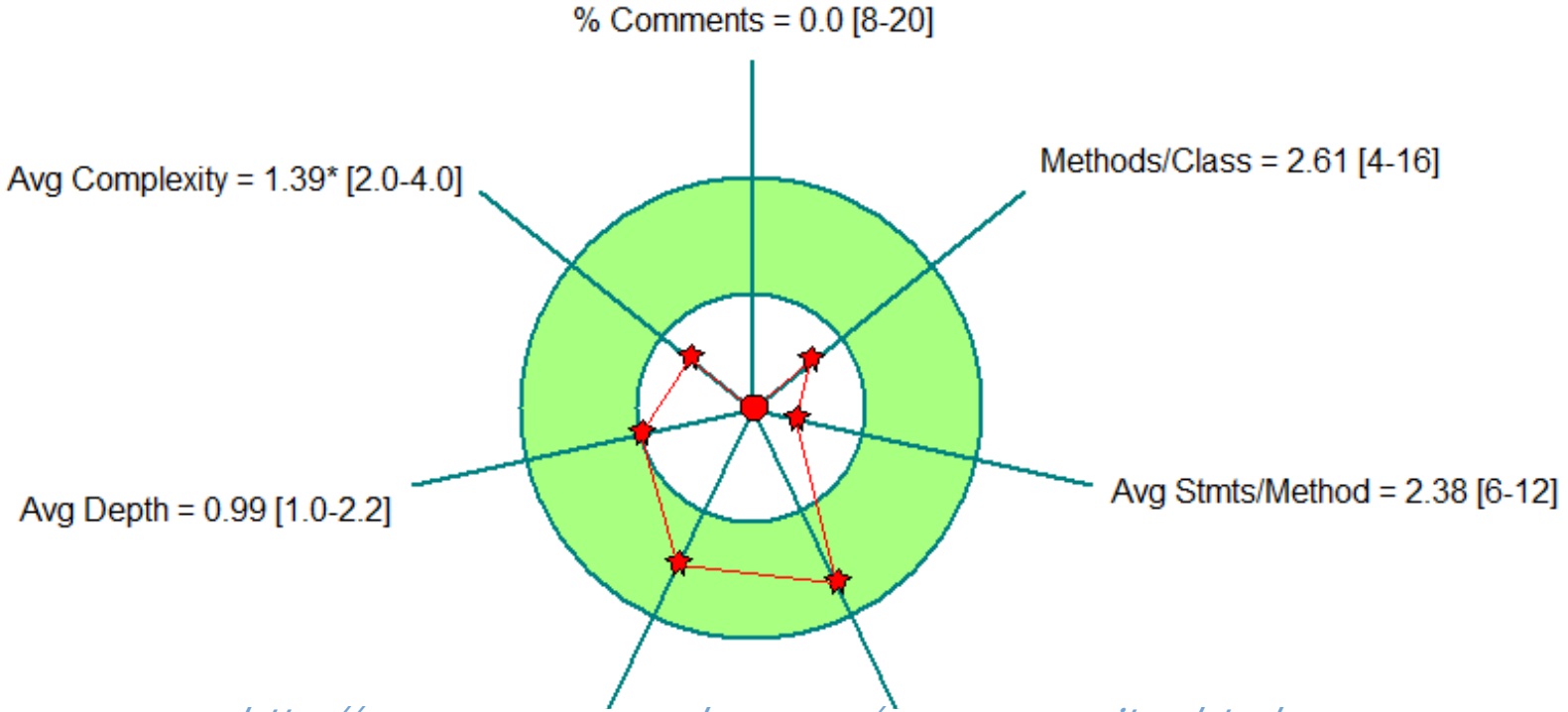
Monoliths in the Wild – Liferay Portal





Kiviat Metrics Graphs

https://github.com/pivotalservices/spring-music



http://www.campwoodsw.com/sourcemonitor.html

How The Hell Did We Get Here ?

THROWAWAY CODE (QUICK HACK, KLEENEX CODE, DISPOSABLE CODE, SCRIPTING, KILLER DEMO, PERMANENT PROTOTYPE, BOOMTOWN)

PIECEMEAL GROWTH (URBAN SPRAWL, ITERATIVE-INCREMENTAL DEVELOPMENT)

KEEP IT WORKING (VITALITY, BABY STEPS, DAILY BUILD, FIRST, DO NO HARM)

SHEARING LAYERS

SWEEPING IT UNDER THE RUG (POTEMKIN VILLAGE, HOUSECLEANING, PRETTY FACE, QUARANTINE, HIDING IT UNDER THE BED, REHABILITATION)

RECONSTRUCTION (TOTAL REWRITE, DEMOLITION, THROWAWAY THE FIRST ONE, START OVER)

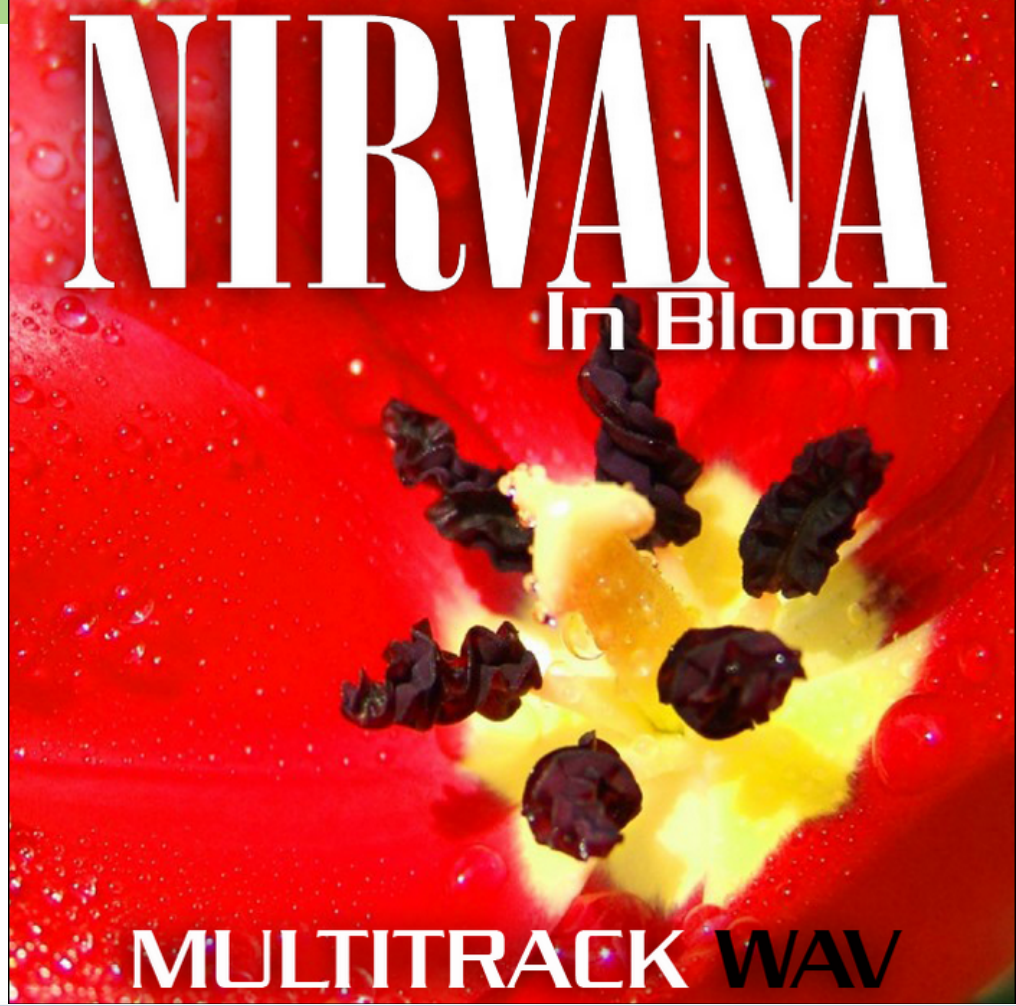
Is there Any Hope ?

Homeostasis and Retrospective feedback - "Homeostasis insulates the system from short-range fluctuations in its environment, while feedback mechanisms respond to long-term discrepancies between a system's actual and desired behavior, and adjust it accordingly."

- *One of mud's most effective enemies is sunshine*
- *Leverage Extreme Programming*
  - *Rely heavily on feedback to keep requirements in sync with code*
  - *Do not engage in extensive up-front planning*
  - *Pair programming and Test Driven Development*

Map to Nirvana

1. Keep the system healthy
2. Conscientiously alternate periods
    1. EXPANSION
    2. CONSOLIDATION
3. Identify Seams
4. Find an inflection point
5. Cover the inflection point
    - Break external dependencies
    - Break internal dependencies
6. Write tests.
7. Make changes
8. Refactor the covered code.



The Power and Promise of Cloud


Yes!
You can get these benefits by building Cloud Native Apps on the Cloud

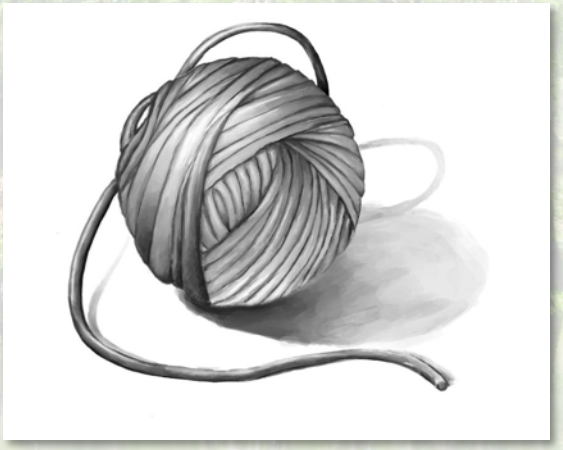
Wait!

You have a portfolio of legacy applications. Can they* be moved? How do we do this?

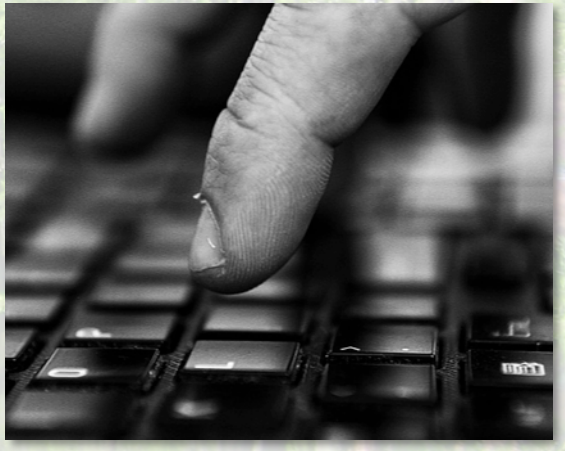
* Okay, if you still have apps running on an IBM System/360, they wouldn't be the first I would move ☺

The Prospect of Legacy Transformation

*Your portfolio is a **complicated** and **poorly documented** mix of many things*



*App architecture is generally **tightly coupled** code and **complex dependencies***



*Your people spend their days **working with legacy technology**; they lack new skills*

Migrating the Monolith

OLD SCHOOL 	NEW SCHOOL 
Measure Twice, Cut Once	*Repeated smaller experiments*
Detailed Application Assessment	*Push app to production*
Magic Quadrant Application Suitability	*Test Driven Migration*
Static Analysis Rule Sets	*Replatforming*
Questionnaires	*Refactoring*
Analysis - Paralysis	*Extreme Refactoring*

Transform to Cloud Native Applications

TRANSFORMATION

APPLICATION REPLATFORMING

Move 10's of Existing Apps to Pivotal Cloud Foundry With Minimal Rework

- Find Suitable Apps
- Make Minimal Code Changes
- Enable Continuous Delivery
- Move Tested Apps to PCF
- Establish Repeatable Process
- Enable Cloud Native Skills

APPLICATION MODERNIZATION

Modernize a Complex Legacy System Into Cloud Native Architecture

- Plan Data & App Architecture
- Reshape Legacy Code
- Build Microservices
- Enable Continuous Delivery
- Use Pivotal Methodology
- Enable Cloud Native Skills

Traditional Ways of Transforming

Discovery

Gather your data—what applications you have, who uses them, how often, when they were written and in what language.

- Discover/inventory applications
- Capture metadata

Migration and Testing

Skilled professionals migrate applications at scale with automated tools and standard processes.

- Remediate and/or migrate applications
- Document code changes and configurations
- Migrate databases, interfaces and dependent applications
- Validate and accept

Migration Planning

Create a practical roadmap based on application assessment. Focus on early value and momentum.

- Plan, estimate and prioritize migration
- Update business case
- Conduct pilots
- Design future architecture
- Define migration roadmap

Assessment

Using our migration framework, make key decisions on the path and destination for each application.

- Establish assessment framework
- Determine application disposition
  Lift and shift, remediate, re-platform, decommission
- Determine cloud deployment and service
  IaaS, PaaS, SaaS, private, public, hybrid
- Determine target cloud vendor platform

Deployment

Go live in a more powerful and efficient place.

- Transition to support
- Post-migration review
- Update service catalog

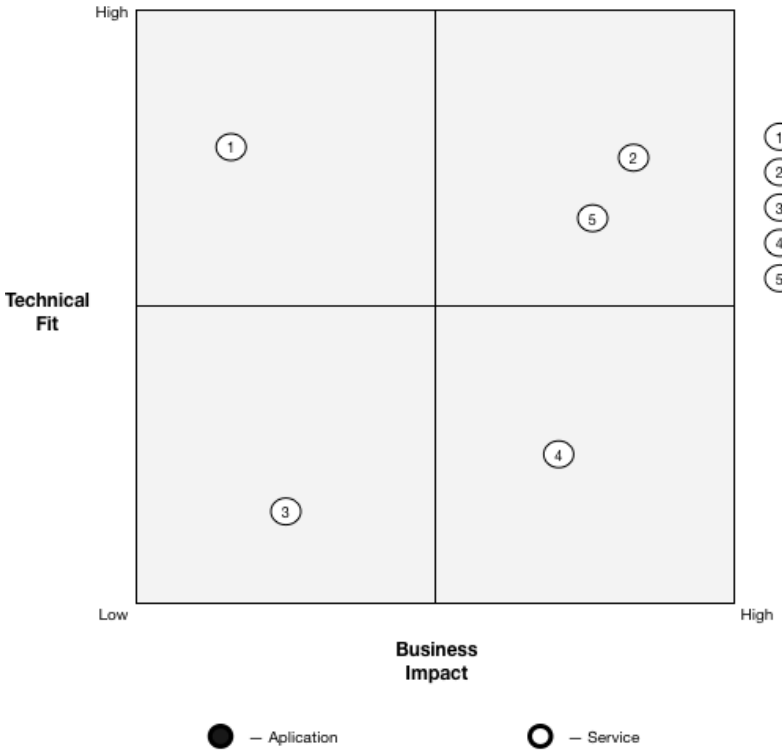
Traditional Ways of Transforming

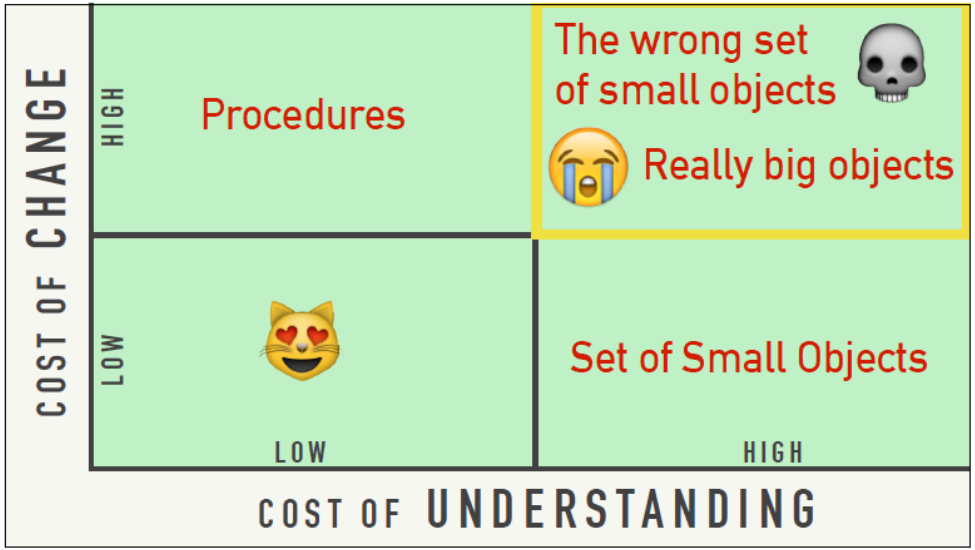
- Led by a Large SI on a Multi-Phase Program
- Always Starts with Analysis using Tools and/or Surveys
- Long projects with big budgets and large batches of work
- Results : powerpoint assessments NO running code

Application Suitability Assessment

	Inbound Network Traffic				HTTP Session involved?		Platform dependent?		Development language/architecture?								Run in a tomcat container?		Rely on Java EE APIs (e.g. EJB, JTA, JCA, JMS)?		Writes to local file system?		3rd party integrations?			Service integrations?										Dependence on properties/config files?		Peer-peet communications?		Specific CPU configuration?	External dependencies?				Hardcoded dependencies to network resources?		Require proxy by any specific component?		Business Impact	
Logical Component	HTTP via port 80	HTTPS via port 443	Non-HTTP or not port 80/443	No	Yes	No	Yes	No	Java	Go	PHP	Python	Ruby	Node.js	.NET	Mainframe	Other	Yes	No	Yes	No	Yes	No	Webservices (SOAP/REST)	JMS Integrations	Other	Oracle	MySQL	Postgres	MongoDB	Redis	SMTP	AMQP	Other	Yes	No	Via direct IP/Multicast	Via messaging platform	Yes	Hardware Dependencies	Software Dependencies	Runtime Dependencies	Filesystem	Yes	No	Yes	No	Future Lifespan of App in Years	Strategic & Central (1..5) 1 is most critical
Application Sample #1	x	x			x			x	x						x			x		x		x	x		x								x		x		x				x		x	x			
Application Sample #2	X				X		X								X				X		X	X		X										X							X		X		X			
Application Sample #3	X				X		X								X				X		X	X		X									X	X							X		X		X			
Application Sample #4	X	X			X			X							X				X		X	X		X		X								X	X							X		X		X			

App Suitability Magic Quadrants



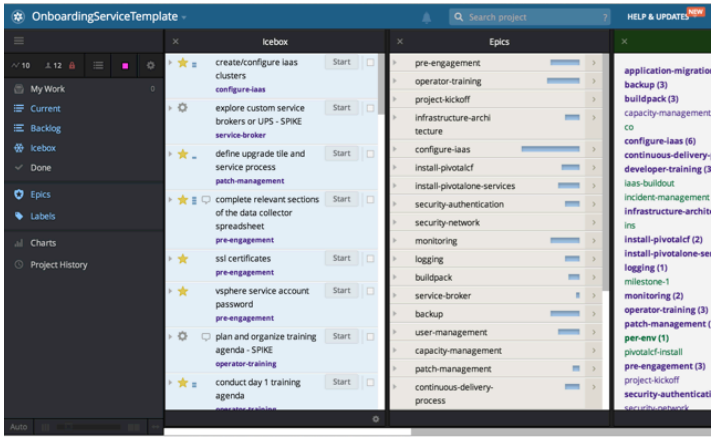


What We Believe

Extreme Refactoring Process

Time Bound 10 weeks, 4 pairs

Scoping → Inception → *Implementation*

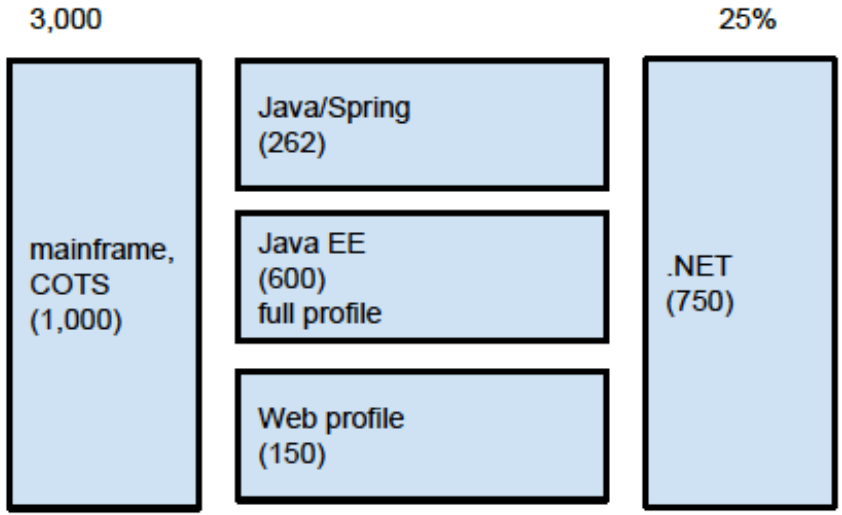


Running applications

CLOUD FOUNDRY™

Application Portfolio Scoping

- *Discovery*
  - *What is the problem we are solving*
  - *Understand App and Service composition*
  - *Broad App Technology Choices*
- *Framing*
  - *How to Solve the Problem*
  - *Migration Flows for Java & .NET apps*
- *Explain Baseline assumptions*
  - *App push to production*
  - *CI/CD Pipelines*
  - *Extreme Programming*



Tools for Scoping

- *Contextual Enquiries*
- *List All the APIs of an application –optional*

```
java jdepend.textui.Jdepend spring-music-master/build |
sed -n '/Summary/,$p' > <report.log>
```

- *Static Analysis Tools – optional – write your own*
  1. *SonarQube*
  2. *Coverity*
  3. *FindBugs*
  4. *Checkstyle*
  5. *PMD*
- *Spring Migration Analyzer - optional- bit dated – Needs a DSL*

Inception

1. Goals
2. Non Goals
3. Risks
4. Roles & Personas
5. Story Mapping
6. Estimation
7. Prioritization
8. Risks (revisited)
9. Next Steps
10. Retrospective

Post Inception Backlog

Anchor generates project backlog from App Replatforming template
Anchor transmits background learning material to CPO
Customer prepares basic portfolio information
Inception leaders prepares for Inception meeting
Inception leader conducts Inception Mtg.
Anchor populates backlog from Inception results
Teams evaluates existing CI/CD toolchain for suitability
Team configures customer CI/CD pipeline (optional)
Team configures configures opinionated Concourse CI/CD pipeline
<App Name> Pipeline executes smoke tests & acceptance to confirm operatic
<App Name> Pipeline deploys to environment
<App Name> is refactored to run in PCF
<App Name> has app modernization assessment
<App Name> has architectural assessment
Team performs transition to FE
Team performs transition to Labs
Team performs transition to Account Exec.

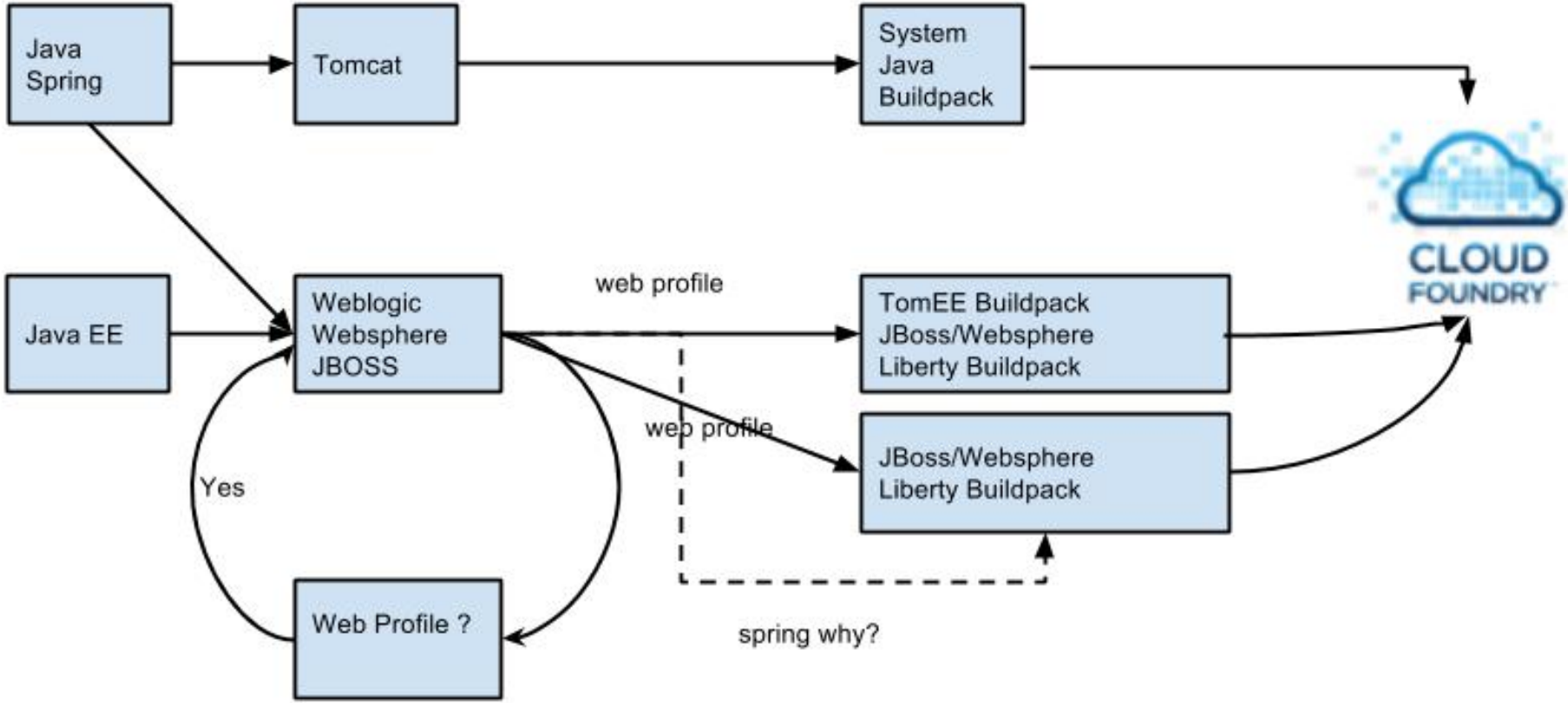
Cloud Foundry

Sssh!! Let me Tell you a secret

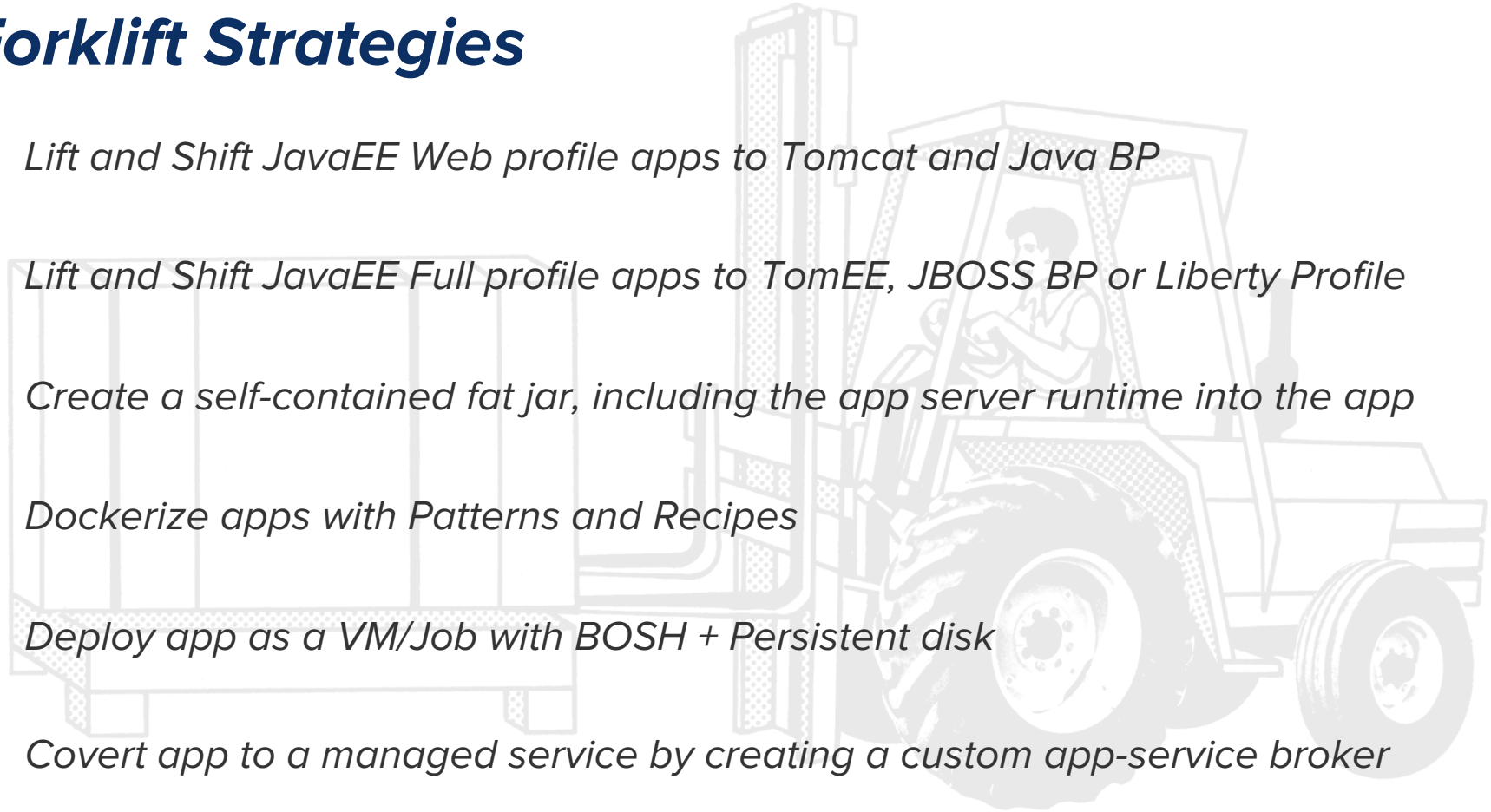
loves cloud native and **non-cloud native apps**

is a polyglot platform (♥ Ruby, Groovy, Go, .NET, etc.)

Java App Replatforming

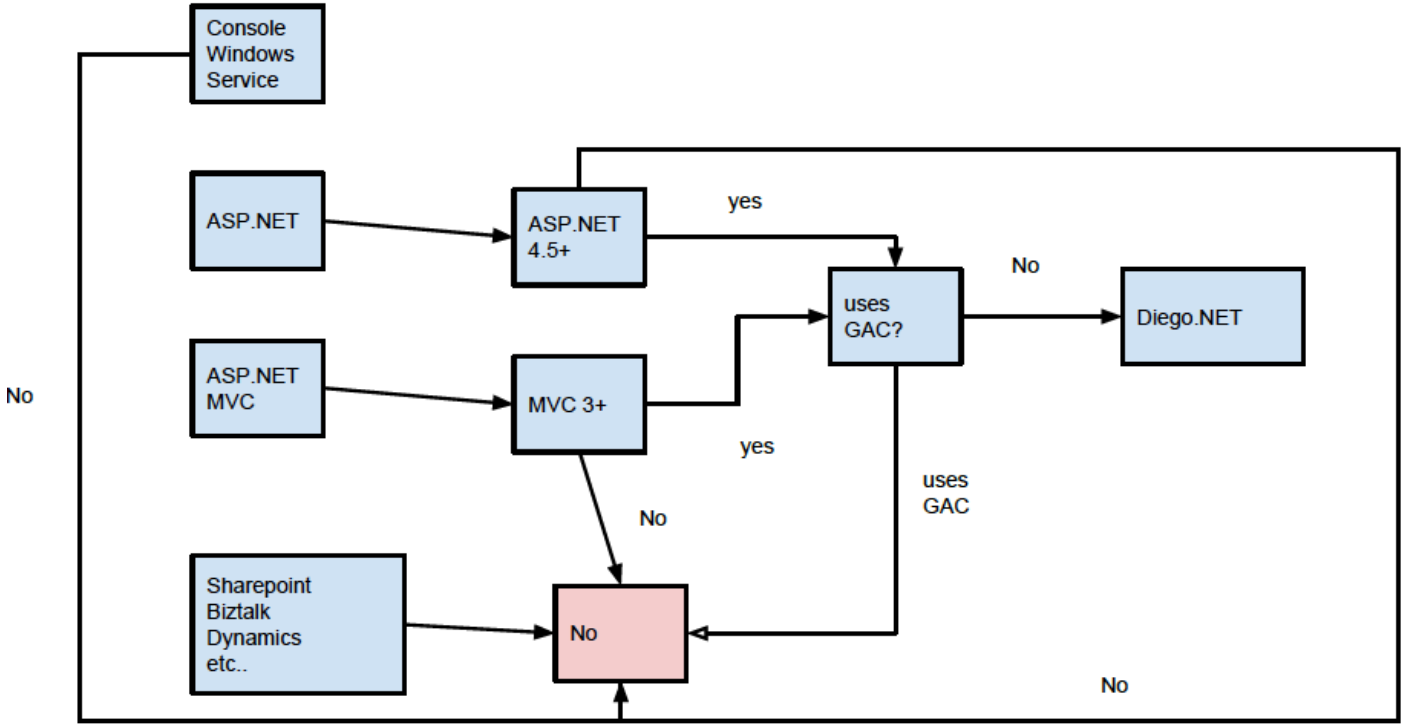


Forklift Strategies

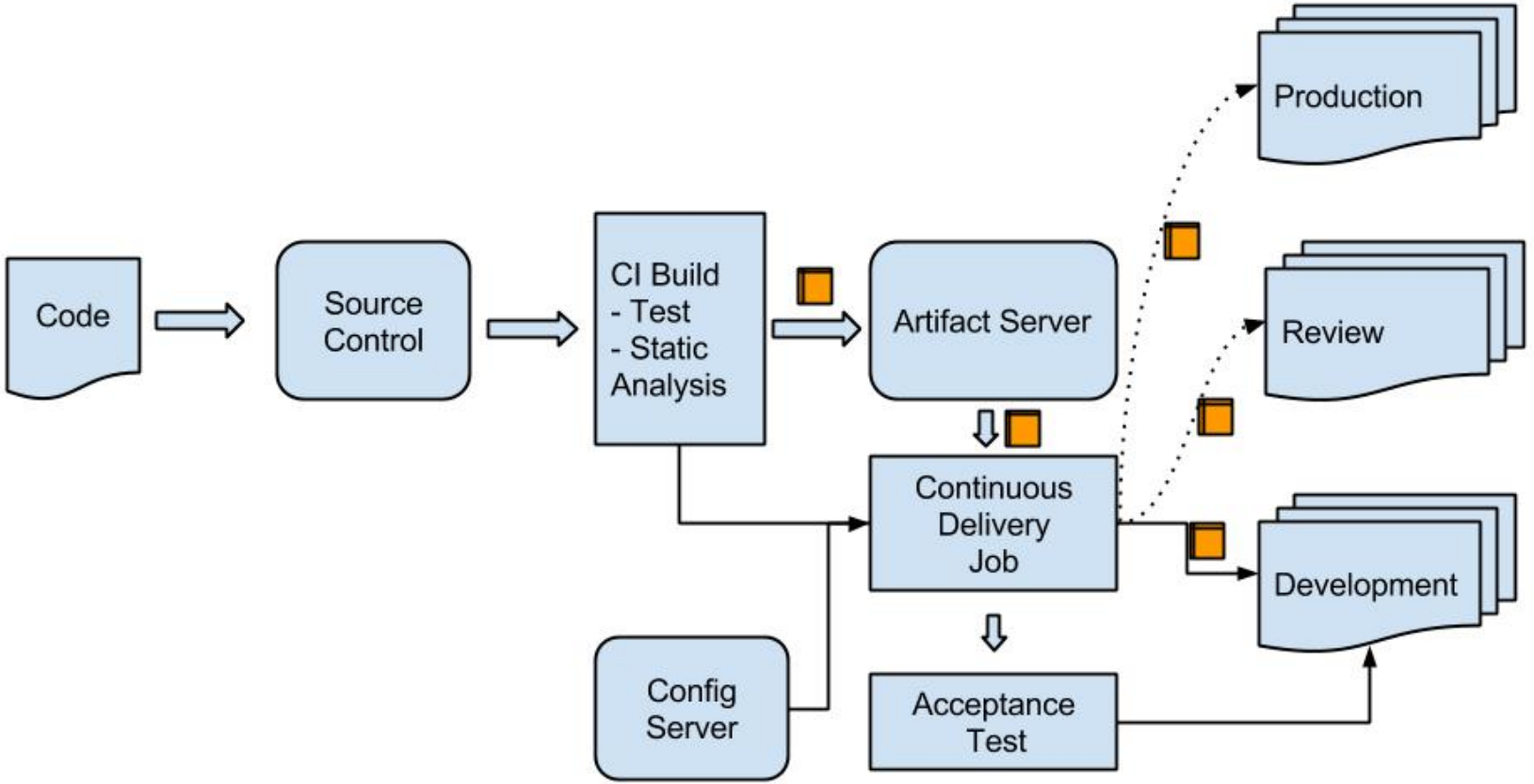


- *Lift and Shift JavaEE Web profile apps to Tomcat and Java BP*
- *Lift and Shift JavaEE Full profile apps to TomEE, JBOSS BP or Liberty Profile*
- *Create a self-contained fat jar, including the app server runtime into the app*
- *Dockerize apps with Patterns and Recipes*
- *Deploy app as a VM/Job with BOSH + Persistent disk*
- *Covert app to a managed service by creating a custom app-service broker*

Dot Net App Replatforming



CI Pipeline Deploys to Production



Continous Integration Tool Set

Source Control

- git
- svn
- cvs
- TFS/VSS
- clearcase
- mercurial

CI Server

- Jenkins
- Concourse
- TeamCIty
- GoCD
- Bamboo
- Travis
- Wercker
- TFS

Artifact Server

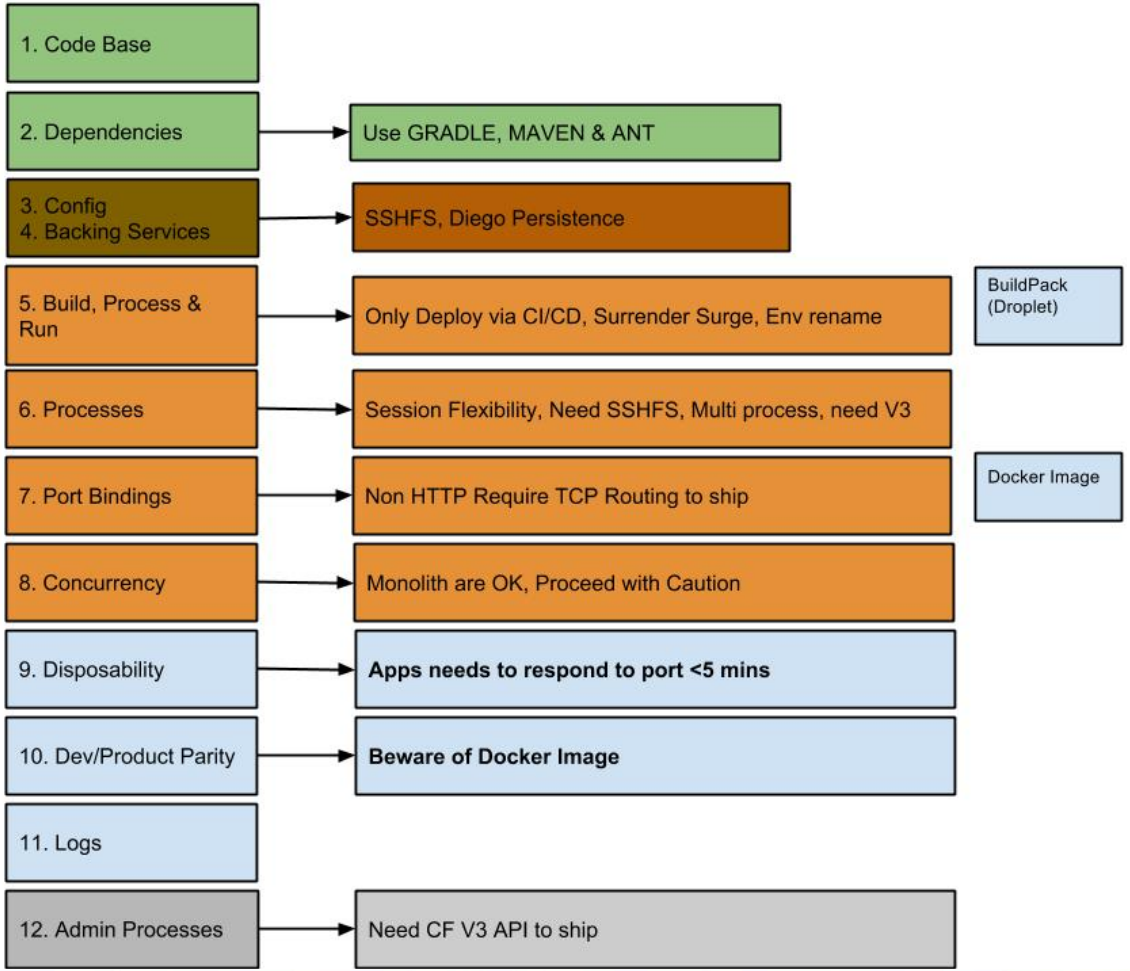
- Artifactory
- Nexus Pro
- Apache Ivy

Requirements

- Pivotal Tracker
- Rally
- JIRA
- TFS
- Clearcase

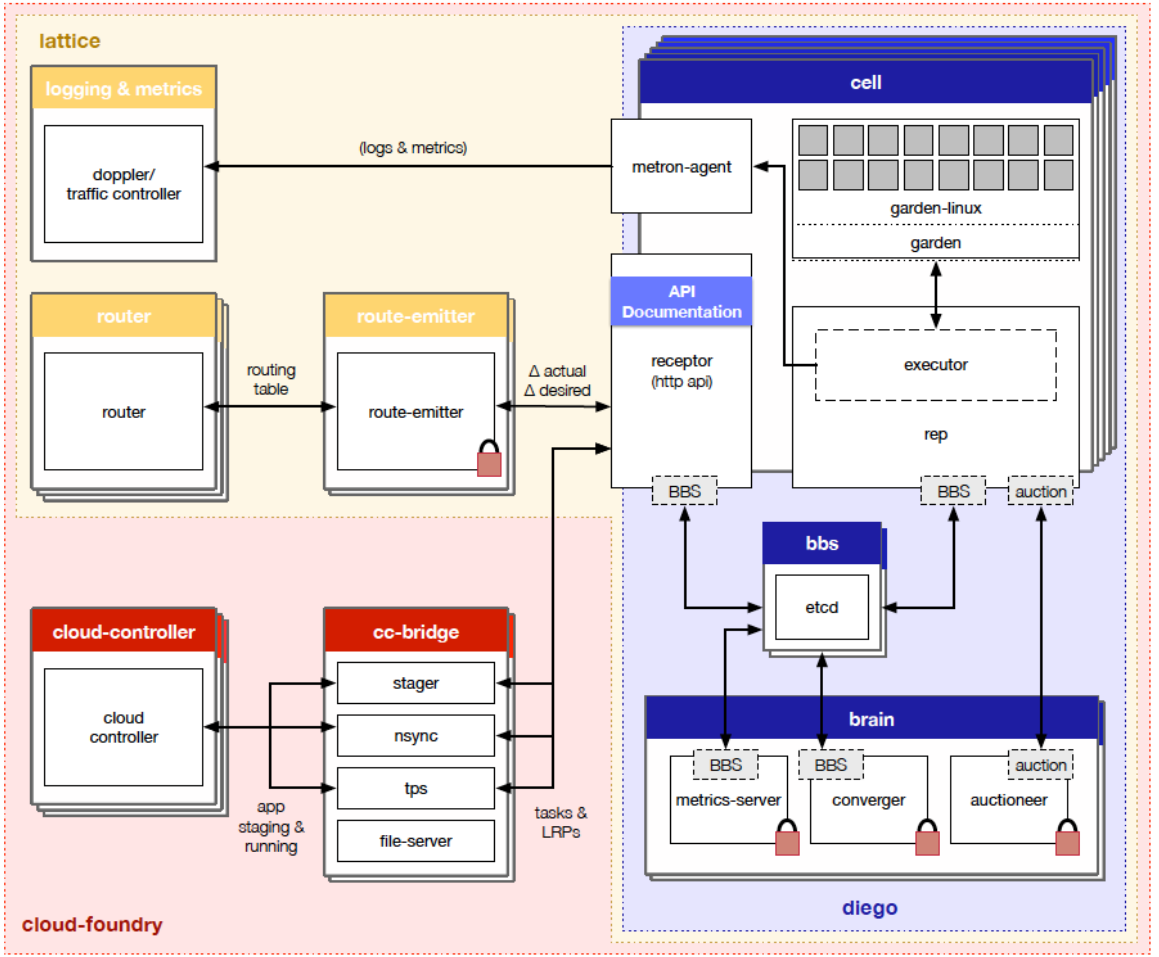
Surrender 12 Factors

12 Factor is the Path to Cloud Native



Critical Features

1. Project Deigo
2. CF v3 API
3. TCP Routing
4. Persistence Support in Deigo
5. Production Grade Persistence
6. File System As A Service
7. Context Path Routing
8. Routing service



PaaS Impediments & Remediations

Anti-Cloud Pattern	*Remediation*
Multiple Inbound protocols like RMI, JMX, Custom-TCP	*TCP Routing, Tunneling*
Persistence in local VMs like Caches, Transaction logs, CMS	*External managed data services, Persistent volume support, SSHFs service, EBS & S3 blob store, mount external NFS*
Distributed Transactions - JTA & 2-pc commit	*Use standalone transaction managers like Atomikos & Bitronix. Introduce eventual consistency patterns*
Configuration	*Externalize, plugin config via Config Server , init script in .profile.d directory that injects the config. as environment variables*
Long Running Batch Processes	*CF v3 API - Different process types*

Let Me Bring You Back From the Evil Empire

1. *Get a lay of the land. Identify ALL the open source and proprietary APIs used by the app.*
   - *Make a list of all WebSphere/WebLogic/JBOSS Proprietary APIs used by your app,*
   - *Make a list of all the Java EE APIs used by your app,*
   - *Make a list of all Spring Framework APIs used the app,*
   - *Make a list of all 3rd party open source frameworks used by the app*
2. *Look at all the XML files bundled within the app.*
   - *Replace vendor specific deployment descriptors with TomEE descriptors or Spring Beans in Tomcat.*
3. *Server scoped shared libraries will need to be bundled in the app when pushed to CF.*

Migrating from Enterprise App Servers to JBP

4. *Disaggregate the EAR file into multiple war files and tie them together as one logical app.*
5. *Take advantage of Java EE6 and collapse EJBs and Servlets in a single war file.*
6. *Do not concern yourself with anything that deals with clustering, workload management, HA, smart routing or proxying.*
7. *Check if there is any native code and ensure that the native library is included with the app and built on the linuxfs32 file system.*
8. *Decide which messaging engine to use to exchange messages asynchronously. CF provides multiple JMS & AMQP messaging providers.*
9. *Auto-configure connections to back-end services or use spring-cloud-connectors to explictly control the JDBC connection to the persistence tier in the cloud.*

Finally Home...

10. *Port JVM arguments, class path, and system properties to the buildpack and cf push manifests in Cloud Foundry.*
11. *Understand the external factors that affect migration like interaction with external CRM/ERP systems via XML or legacy web-service stacks.*
12. *If app relies on session persistence then configure session replication in CF by binding the app to a DB like Redis/Gemfire session replication on Cloud Foundry*
13. *Liberty Tech Preview tool helps move WebSphere Application Server full profile applications to Liberty profile.*
    - *Look for migration tools and guides to Tomcat.*
14. *Follow the Java App Migration Flow Chart to push the app to CF*

How To Refactor ?

Push

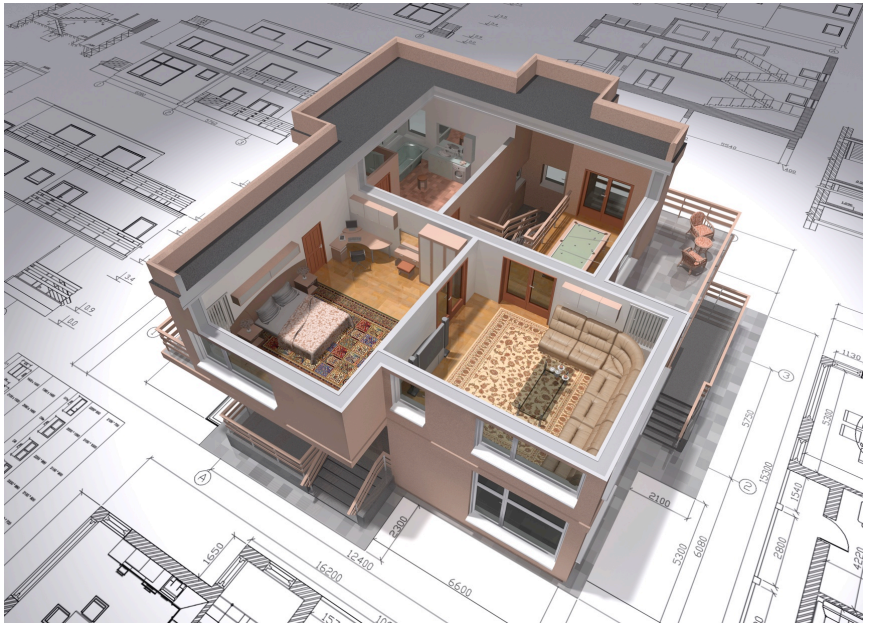
1. Push the app
2. See what fails
3. Write a test to capture the failing assert
4. Refactor app to get the tests to pass
   - Sacrifice some of the 12 factors
   - Create a story to track future refactoring
5. Modified app passes the test
6. Scrub & Repeat

Pull

1. Understand Bounded Contexts by whiteboarding of activity & component diagrams
2. Evaluate for 12 factor compliance.
3. Chose which components we want to tackle.
4. Rely heavily on integration tests, refactoring tools and interfaces
5. Leverage Proxy/Façade/Smart Routing pattern to create an alternate implementation
6. Run tests again pointing to the alternate endpoint
7. If tests pass get rid of the original implementation

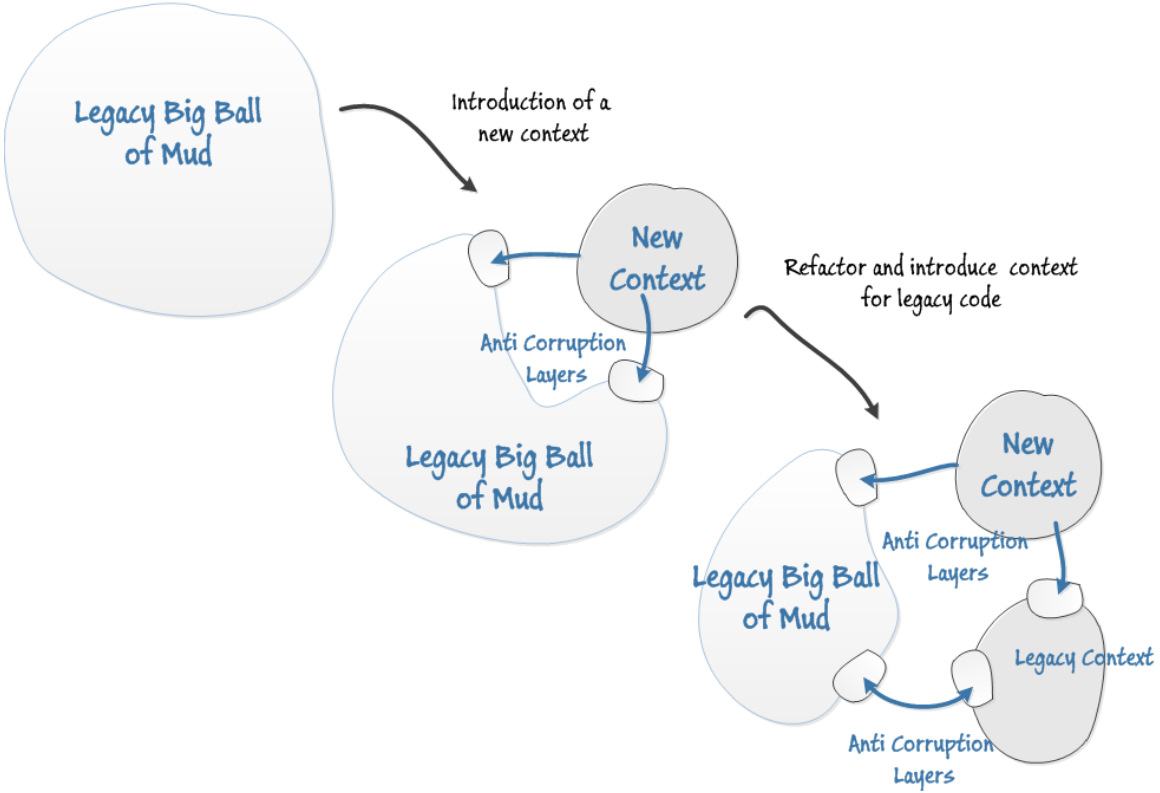
Breaking the Monolith – Picking Seams

- *Stability and Point of Evolution*
- *Inbound and Outbound Coupling*
- *Tools - Xray, JDepend, Structure101*
- *Databases & Data Stores*
- *Transaction Boundaries*
- *Modes of Communication*
- *Team Organization and Structure*
- *Use Cases/User Journeys*
- *Business Processes*
- *Verbs & Operations*
- *Nouns & Resources*
- *Separated models for reading and writing*



Monolith Decomposition Patterns

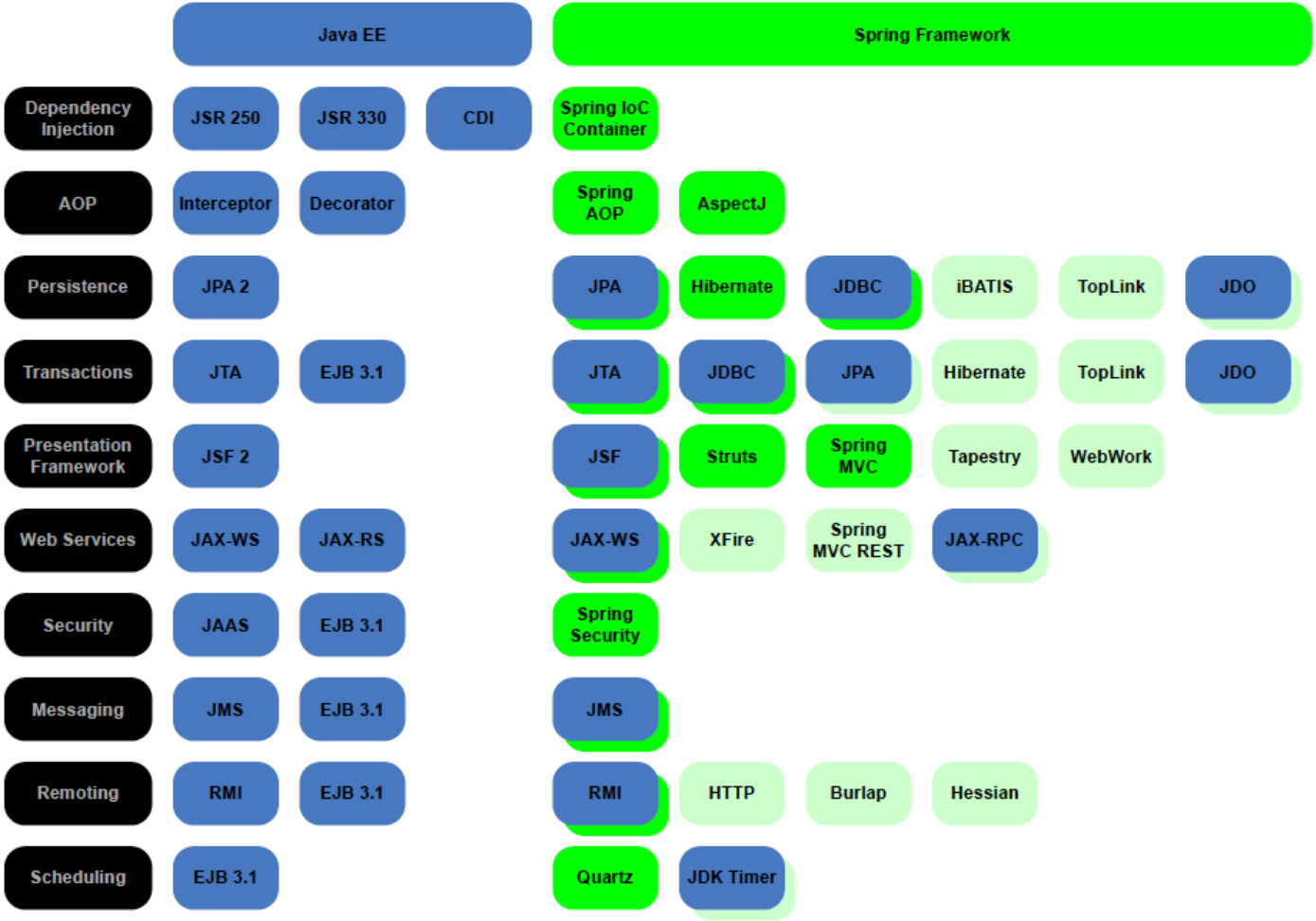
- *Anti-Corruption Layer*
- *Strangling the Monolith*
- *Inverse Conway Maneuver*
- *Proxy Pattern*
- *Façade Pattern*
- *Dark Launching*
- *Feature Flags*
- *Smart Routing*



Migrating Data

- **Tools – SchemaSpy**
  - Generate graphical representations of the relationships between tables.
- **Tools - Liquibase, Flyway, jooQ**
  - Auto apply bundles of database refactorings
- Require a transition period during which both the original and new schemas exist in production
- Expose a Facade service to allow DB to change on the backend
- Move logic and constraints to the service level
- Implement retry and compensations
- Database Transformation Patterns cataloged in "Refactoring Databases" seminal book by by Scott J Ambler and Pramod J. Sadalage

Migrating JavaEE To Spring Framework



SPRINGONE2GX
WASHINGTON, DC

Learn More. Stay Connected.

 @springcentral

 Spring.io/video

rkelapure@pivotal.io http://cloud.rohitkelapure.com/

eBook: Practical Microservices: http://bit.ly/practicalmicroservices