Replatform Workflow

Platform Architecture

Pivotal

Who this deck for?
This deck is aimed at sales delivery to help them better understand replatforming.

This deck contains a lot of material that can be reused for client facing conversations.

Replatform Overview

Platform Architecture

Pivotal

Topics

Replatform Overview

- Why replatforming is important
- How we help customers migrate applications
- What we do in an engagement

Running Apps Drives Consumption

There Are Two Paths Here

BUILD NEW APPS

CO-LOCATE YOUR PEOPLE AT LABS; BUILD NEW APPS WHILE LEARNING MODERN SOFTWARE PRACTICES ON-THE-JOB

MIGRATE EXISTING APPS

GET MIGRATION GOING BY PAIRING OUR EXPERTS WITH CUSTOMER PEOPLE AS WE MOVE LEGACY APPS TO PCF

EXISTING, INTEGRATED SYSTEMS

Pivotal **Cloud Foundry**

YOUR IAAS CLOUD(s)

Pivotal

Most Existing Custom Apps Will Run on PCF

But, Why? Why Migrate to Pivotal Cloud Foundry?

FREEDOM
To Run Your App on Any Cloud

AUTOMATION
To Lower Your Ops Cost and Increase Your Velocity

RESILIENCY
Auto-Scale, Blue/green Deployments and Health Mgmt.

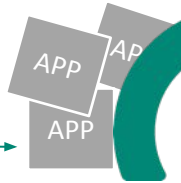
GOVERNANCE
Common Set of Developer-Self-Serviced Frameworks

VISIBILITY
End to End Logging, Monitoring and Auditability

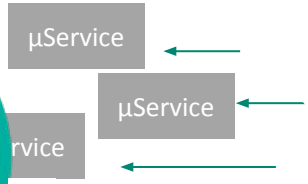
SPEED
Move Cycle Time from Months to Weeks to Days or Hours

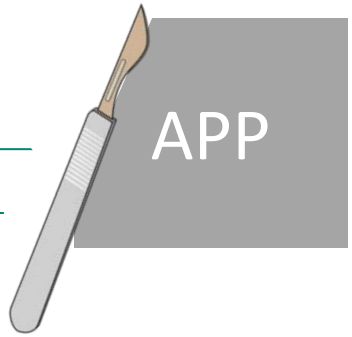
Replatform and Modernize

APPLICATION REPLATFORMING



APPLICATION MODERNIZATION





The "Push" Approach

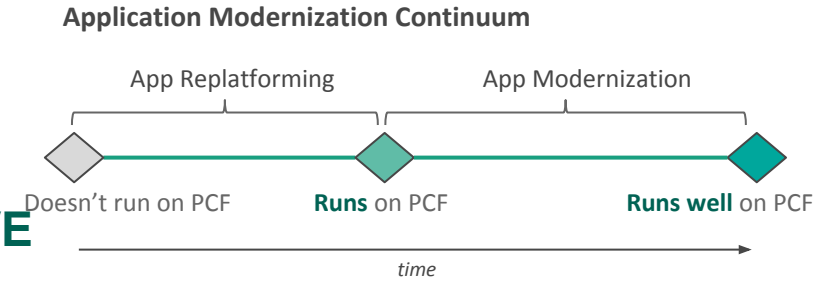
1. Push the app
2. See what fails
3. Write a test
4. Get the tests to pass
5. Write down your recipes

The "Pull" Approach

1. Build new features as microservices
2. Find the seams
3. Write tests
4. Refactor code around seams
5. Get your tests to pass

Pivotal

Cloud Native Terms



GREENFIELD - NEW BUSINESS INITIATIVE

- **New App development** Driven by new features
  - New code base with -- modern software (Spring Boot), modern architecture (Microservices), deployed to modern platform (Pivotal Cloud Foundry), Sm-Lrg Arch

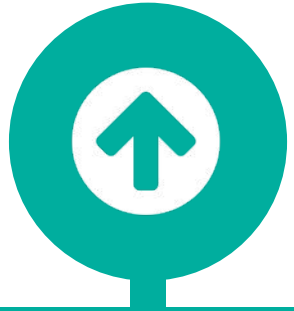
BROWNFIELD - EXISTING APP RUNS ON LEGACY IT

- **Application Replatforming**
  - Modify an existing application to run on Pivotal Cloud Foundry **(minimum required changes)**
  - Typically includes **minor config and packaging changes**; little to no code changes
- **Application Modernization**
  - Modify an existing application to run well on Pivotal Cloud Foundry
  - Typically includes **major architecture, code** and packaging **changes**; may include feature work
- **Application Rewrite** (Similar to New App Development) Driven by technical debt
  - New code base with -- modern software (Spring Boot, Spring Cloud Services), modern architecture (Microservices), deployed to modern platform (PCF), Lrg Arch

Pivotal


What We Believe
• You Should Move Apps to PCF ☺
• Plan Just Enough to Start
• Start With “One Thing”
• Break Big Things into Small Things
• Automate Everything You Can
• Build Skills by Pairing and Doing
• Let Real Work Inform Strategy
• Monoliths Aren’t Inherently Bad
Pivotal

The Road to Cloud Native



LAUNCH

Deploy PCF and get an app running from end-to-end while automating as much as you can

SCALE

Build new apps (ideally, with Pivotal Labs!) while aggressively moving existing ones to Pivotal Cloud Foundry to maximize operational efficiency

TRANSFORM

Organize and operate like a high velocity software startup by applying Lean and Agile models to your organization

Pivotal

And How We Help You Succeed

LAUNCH

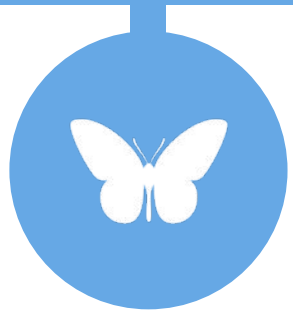
- DESIGNATED ARCHITECT
- PLATFORM OPS SERVICE
- PLATFORM DOJO

SCALE

- LABS PRODUCT DEV
- APP REPLATFORMING
- DOJO – BOSH / PLATFORM
- EDUCATION & ENABLEMENT
- BUSINESS CRITICAL SUPPORT

TRANSFORM

- EXECUTIVE WORKSHOPS
- ADVISORY SERVICES
- LABS ENABLEMENT



Pivotal

What to Expect

Pivotal Cloud Foundry Application Replatforming

Use Real Work to Inform Your Strategy by Moving ~ 10 (or more) Apps in 10 Weeks to PCF While Learning On-the-Job

6-10 WEEKS

TEAM OF 4.25

TYPICAL SCOPE

- Backlog Planning & Prioritization
- Application Suitability Reviews
- Migrate Apps in Developer Pairs
- Build PCF Extensions
- Automated Testing Pipelines
- Transition Apps Quickly; Increase Velocity Over Time
- Cookbook of Replatform Patterns
- Invaluable OTJ Knowledge Transfer

OUTCOMES

- Strategy Informed by Real Work
- Legacy Apps Running on PCF
- Sunset of Legacy Systems and Licensing
- New PCF Operational Efficiencies
- Established Patterns for Subsequent Replatform Work
- Next Steps Guidance
- Valuable On-The-Job Training and Enablement

Pivotal

What to Expect

Pivotal Cloud Foundry Application Modernization

Use Real Work to Inform Your Strategy by Moving ~ 1 App in 10 Weeks to PCF While Learning On-the-Job

10 WEEKS

TEAM OF 4.25

TYPICAL SCOPE

- Backlog Planning & Prioritization
- Application Suitability Reviews
- Decompose monolith in Developer Pairs
- Build PCF Extensions
- Automated Testing Pipelines
- Pattern cookbook
- Invaluable OTJ Knowledge Transfer

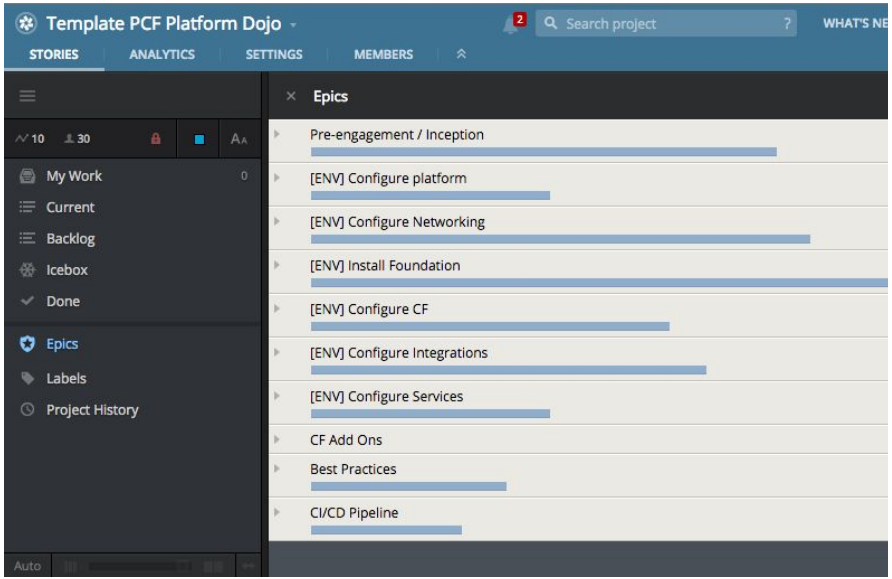
OUTCOMES

- Strategy Informed by Real Work
- Mission critical App Running on PCF
- Sunset of Legacy System and Licensing
- New PCF Operational Efficiencies
- Established Patterns for Modernization Work
- Next Steps Guidance
- Valuable On-The-Job Training and Enablement

Pivotal

How We Do What We Do

Pairing With Developers and Operators to Enable Awesome



We Use The Pivotal Methodology

- Pre-kickoff planning
- Day One Inception prioritizes the scope
- Backlog managed to meet evolving needs

A Toolkit That Maximizes Customer Touch

- Pivotal Tracker for backlog management
- GitHub for source code collaboration
- Slack for messaging and knowledge sharing

Our Job is Software Success

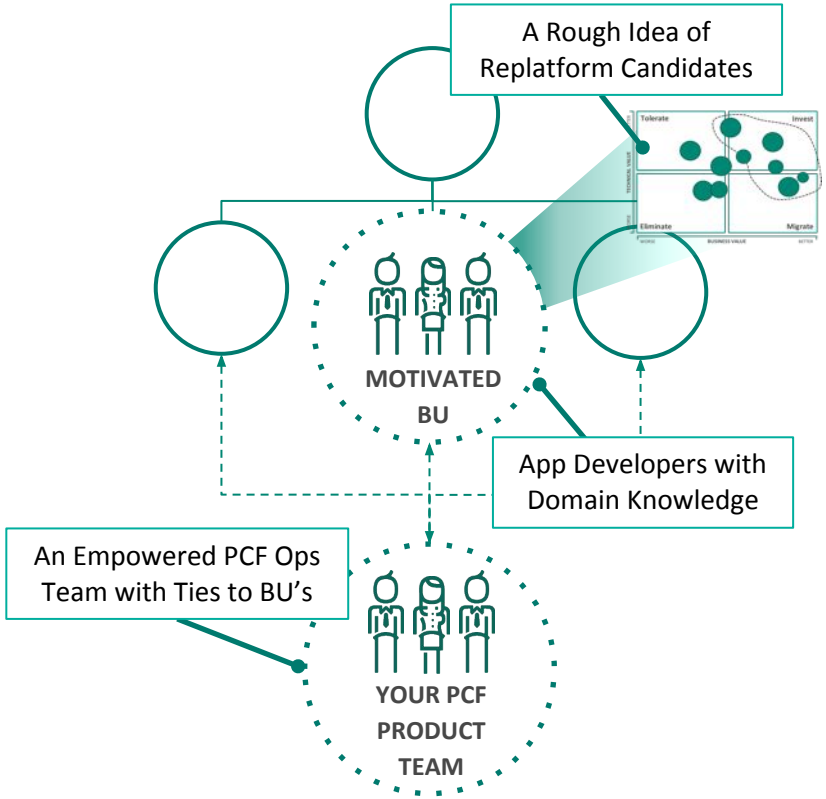
- Best practice re-platform recipies
- Skills and process enablement

WE ACCELERATE SUCCESS BY HELPING CUSTOMERS LEARN BY DOING

I DO > WE DO > YOU DO

Pivotal

How to Start – What You'll Need



ORGANIZATIONAL COMMITMENT

A motivated business unit with leadership committed to Cloud and a willingness to invest time and money in transformation.

LIST OF APPLICATION CANDIDATES

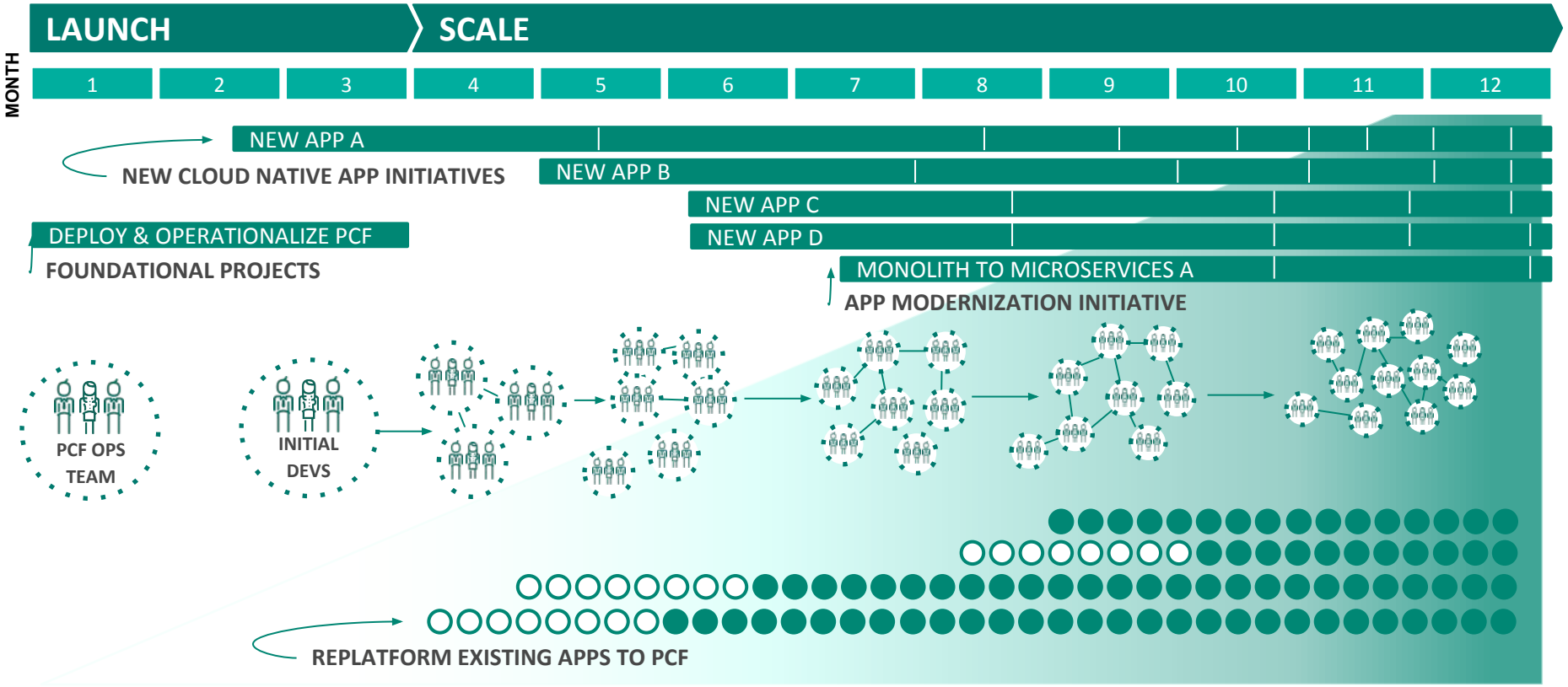
Start with apps built <7 years ago using tech supported by PCF (ex. JEE) that are actively being used, updated and maintained

GOOD PEOPLE

A small team of people that understand candidate applications and made available to work on the initiative in a dedicated way

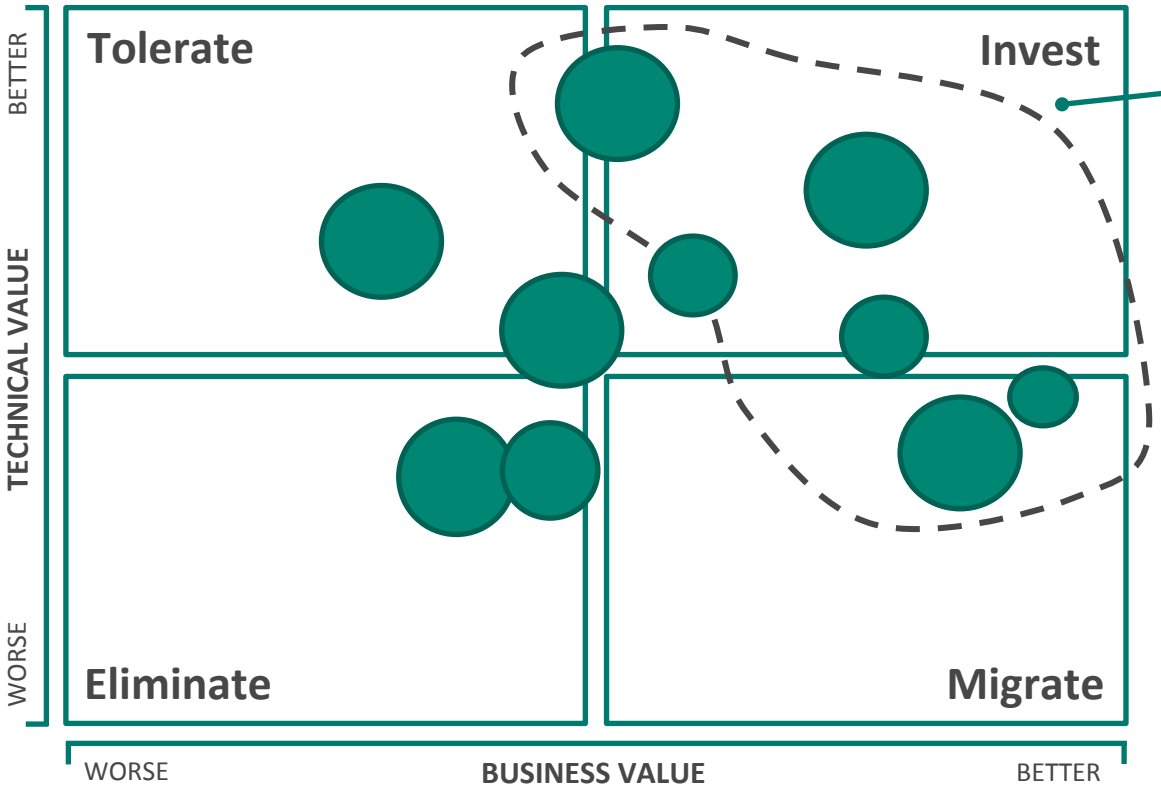
Pivotal

What All of This Might Look Like



Pivotal

Ex: Reference Application Portfolio



Out of the 76 Websphere applications that will be migrated to PCF. Focus here and start with the first 10 most impactful apps

Realize Immediate ROI Through:

Auto-Scaling & Provisioning
Standardize on devstack & platform services
Automated patching & upgrading
Application Modernization
Operational efficiencies by reducing headcount
Runtime consolidations & reduction in proprietary License Spend

* Gartner's TIME methodology for Application Portfolio Rationalization

Pivotal

Replatforming in Summary

How to Get a Replatforming Initiative Started

- **Understand the Value of Replatforming to PCF**
  - Operational Efficiency – Health Management
  - Time-to-Market – Accelerate SDLC with CI/CD
  - Retire Licenses – Remove Dependencies as You Move
- **Identify First Movers**
  - “Cast a Wide Net”
  - Participate in Re-platform Workshops
- **Our Approach**
  - Move “10s” of Apps in 10 weeks
  - Create Cookbook as We Do the Work Together
  - Use Initial People to Seed Next Teams

Replatform Deep Dive

Platform Architecture

Pivotal

Topics

Replatform Deep Dive

- Cloud native journey level set
- Application down select process
- Replatform best practices
- App Modernization best practices

Legacy Portfolio Realities

Things we hear:

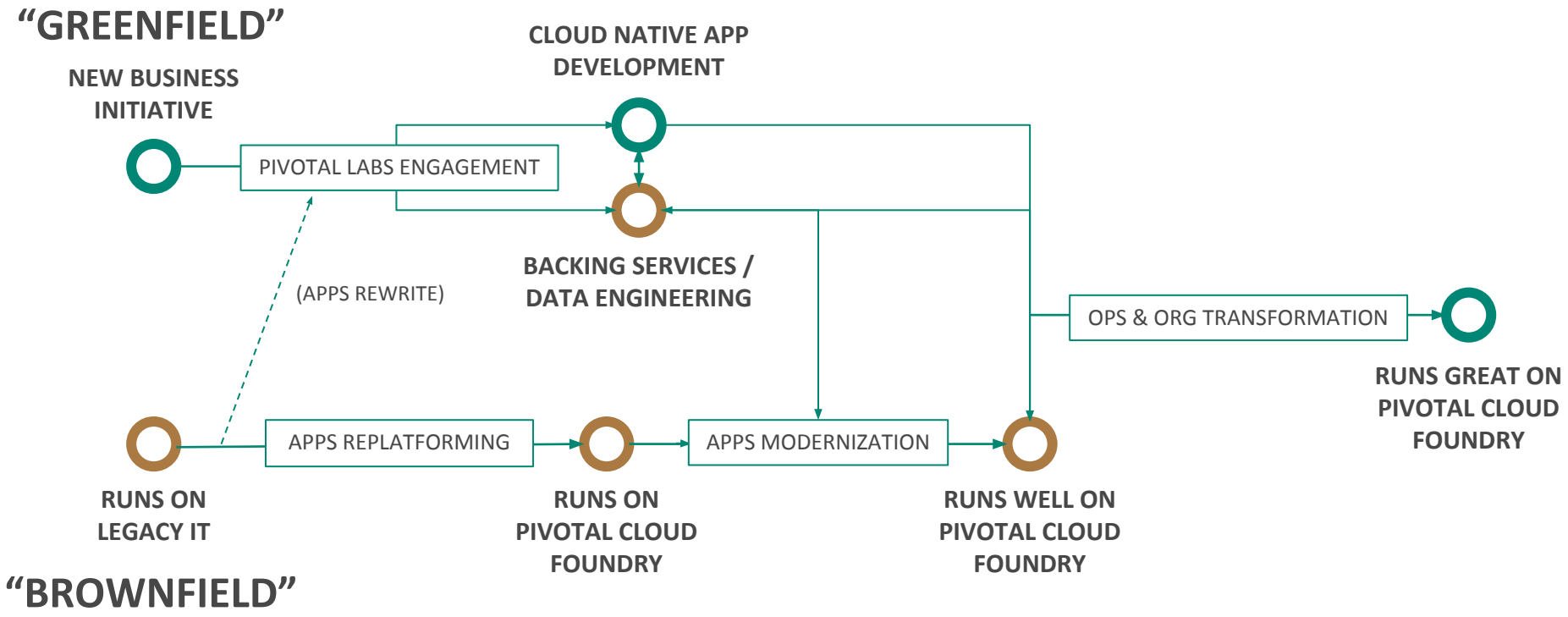
- Our portfolio is **complicated**, **documentation is sparse** and it's a mix of many things
- Architecture is often **tightly coupled** code and **complex dependencies**
- Many people spend their days **working with legacy technology**; they lack new skills

What we believe:

- Pivotal Cloud Foundry is the **comprehensive Cloud Native platform** to build the future
- You **get value** by **building apps on it**
- You **see more value** by moving existing apps

Pivotal

Simplified Journeys to Cloud Native Agility

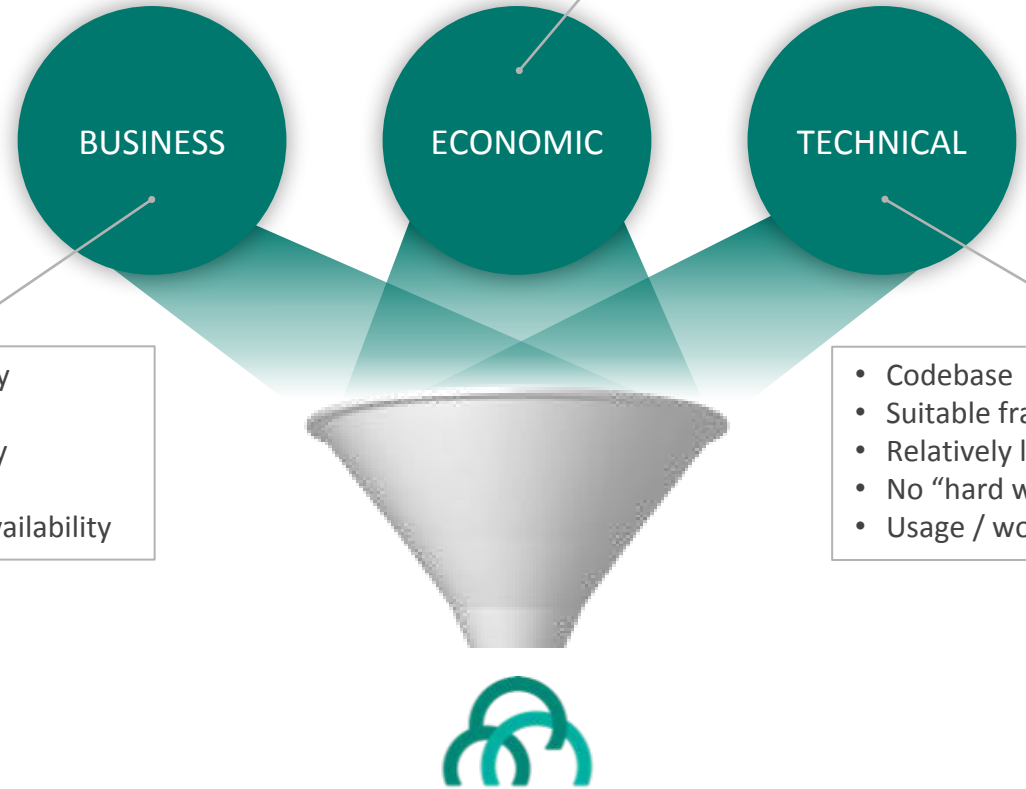


Pivotal

Brownfield Journey

Choosing Candidate Applications

- Licensing costs (HW / SW)
- Time-to-market implications
- Revenue impact-fullness

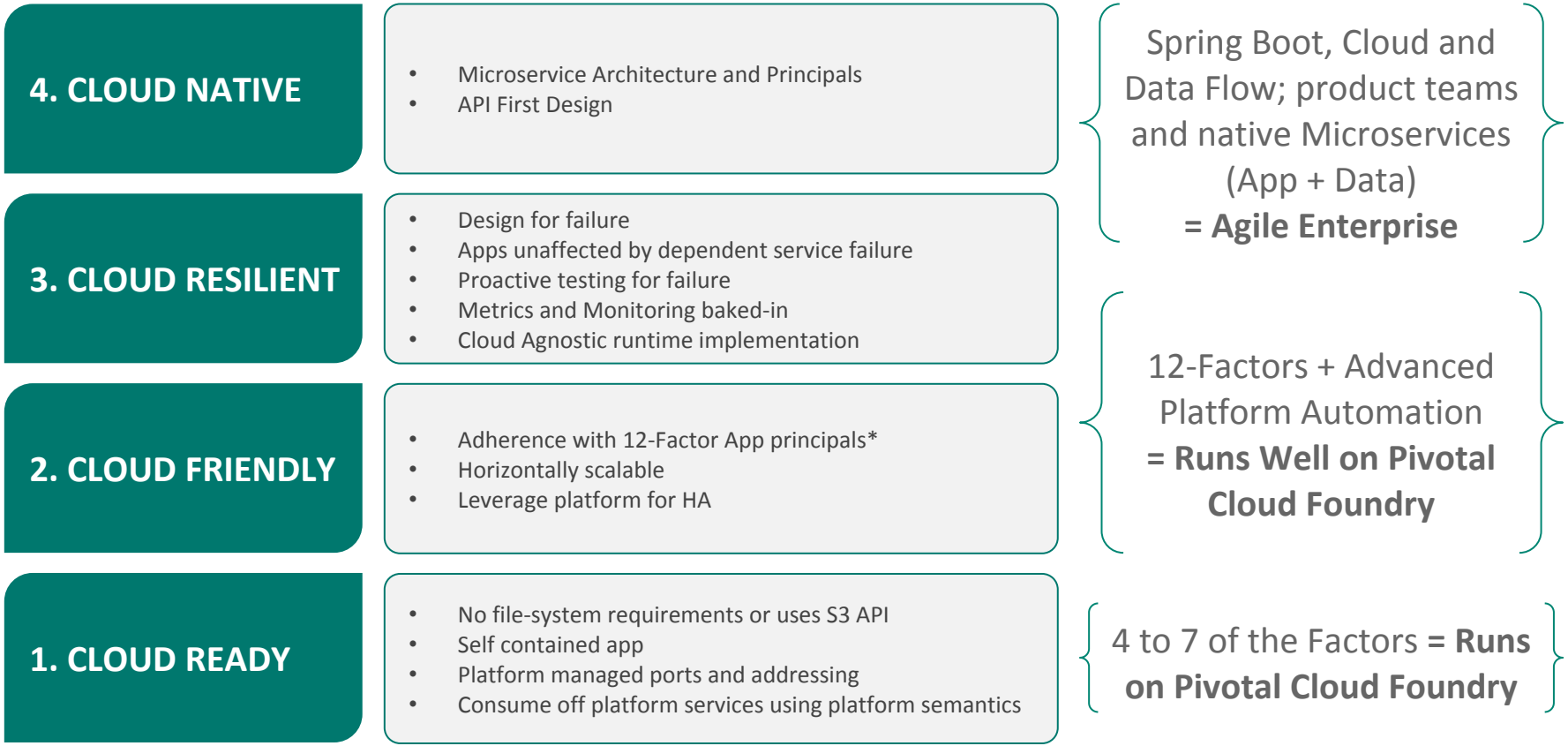


- Business criticality
- Risk tolerance
- Change frequency
- Lifecycle stage
- Domain expert availability

- Codebase
- Suitable framework / runtime
- Relatively lightweight footprint
- No “hard wire” file system dependencies
- Usage / workload

Pivotal

A Brownfield Maturity Model



* We believe there are more like 15 “factors” that exemplify a true, “Cloud Native” application… more later

Pivotal

Why Bother?

Why Migrate Monoliths to Pivotal Cloud Foundry?

- **Cloud Portability** – free to run on any Cloud
- **Operational Efficiency** – reduce headcount, automate “almost” everything
- **Auto-scaling** - efficiencies for spiky workloads
- **Standardization -** on dev stacks and platform-provided services
- **Runtime Consolidation -** reduction of software vendors
- **Constant Innovation -** cloud native platform advancements included in subscription
- **Resilience -** 4 levels of HA with built in management and monitoring

Pivotal

Siri ? How To Start ?

Pivotal

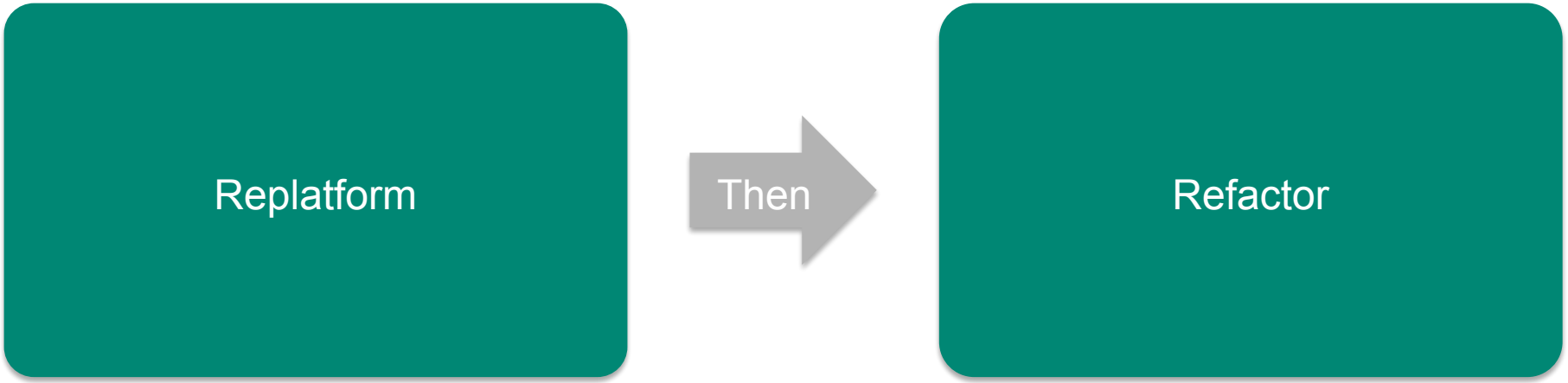
Traditional Approach

Of Tackling Modernization

- Upfront Study in a Multi-Phased Program
- Tools and/or Surveys to Gather Information
  - Consultants who cast a wide net over everything
  - Delivery of an expensive report and phased roadmap definition
- Long projects with big budgets and large batches of work
  - Failure is slow; it takes time and a lot of money to see problems
  - Value is slow; measured returns often take years

Migrating The Monolith

Always …



How to Start

Don’t Plan Everything; Start Small and Let Your Work Inform the Strategy

- **Scope**
  - Get the right people in a room
  - Define the business value to drive
  - Identify candidate apps
- **Discover & Frame**
  - Assess from a 15-factor standpoint
  - Decide on replatform flows
  - Flag risks and assumptions
- **Begin With a Time-Bound Pilot**
  - Designate the right people
  - Work for 6 to 10 weeks
  - Translate lessons into patterns

Pivotal

Getting To Cloud Native

Pivotal

Those 12-Factors

http://12factor.net/

A “manifesto” of sorts published in 2012 by a team at Heroku

The goal of these 12-factors is to teach developers how to build cloud-ready applications using **declarative formats for automating setup**, had a **clean contract with underlying operating system** and were prepared for **dynamic scaling**

A Generalization About…

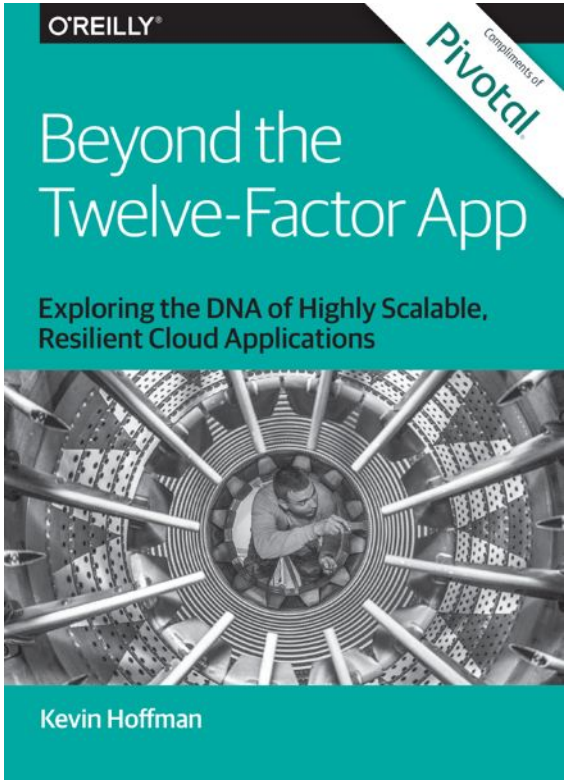
… In Order of Importance and Colored for Level of Effort

VALUE & APPROACH

I. One Codebase, One App*	= **Time to Market**; find the seams; use good SDLC practices
II. Dependency Management*	= **Dev Productivity**; standardize & remove surprises
V. Build, Release, Run*	= **Release Mgmt Hygiene**; use CI/CD automation /w PCF
III. Configuration*	= **Release Mgmt Hygiene**; move to environment vars
XI. Logs*	= **Real-Time Metrics**; use PCF features; stdout / stderr
IX. Disposability	= **Auto-Scale**; move slow processes to backing services
IV. Backing Services	= **Resiliency / Agility**; use circuit breaker; loose binding
X. Environmental Parity*	= **Reliability**; use well architected PCF, get parity
XII. Administrative Process	= **Reliability**; move to backing service(s), expose as REST
VII. Port Binding*	= **Ops Efficiency**; use PCF features like routing, scaling, etc.
VI. Process	= **Cloud Compatibility**; move state to backing service(s)
VIII. Concurrency	= **Auto-Scale, ZDD**; design for cloud, use PCF features

Pivotal

Looking Beyond 12-Factors



- **12-Factor Published in 2012**
  - In context of Heroku
  - A LOT has changed
- **New Guidance**
  - Emphasis on Enterprise Java & PCF
  - 3 new “factors”
    - API First
    - Telemetry – APM, Logs, Domain-Specific
    - Authn / Authz – Security First Design
- **Must Read for Application Architects**

Pivotal

Replatform Suitability

Red (and Yellow) Flags That Should be Evaluated and Planned For

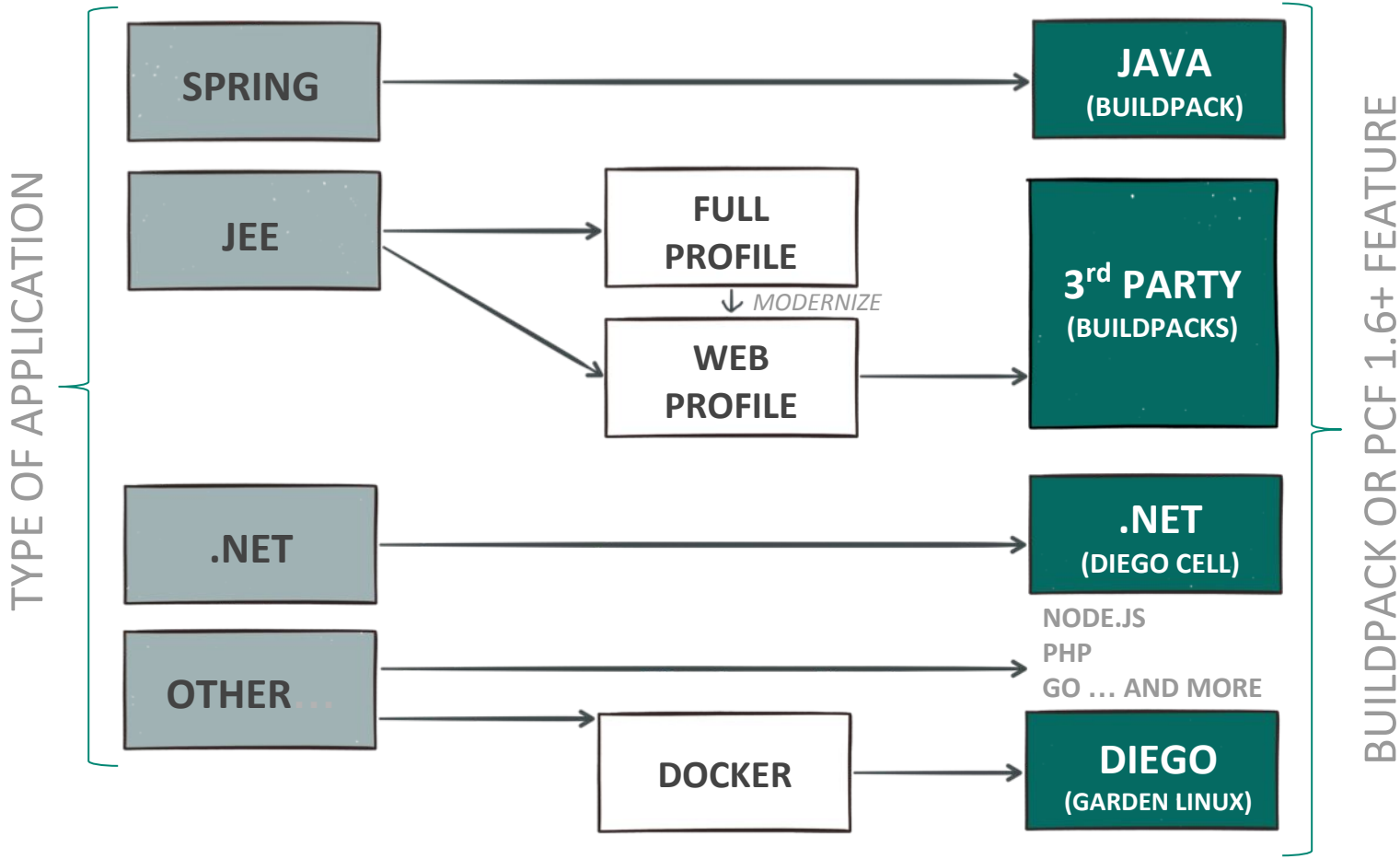
- Accepts inbound connections for non-HTTP protocol
- Application Container Hosted Clustering
- Stateful Process
- Filesystem I/O (Reading/Writing files & NFS mounts)
- Logging other than **STDOUT** or **STDERR**

Migration Suitability Continued

Red (and Yellow) Flags That Should be Evaluated and Planned For

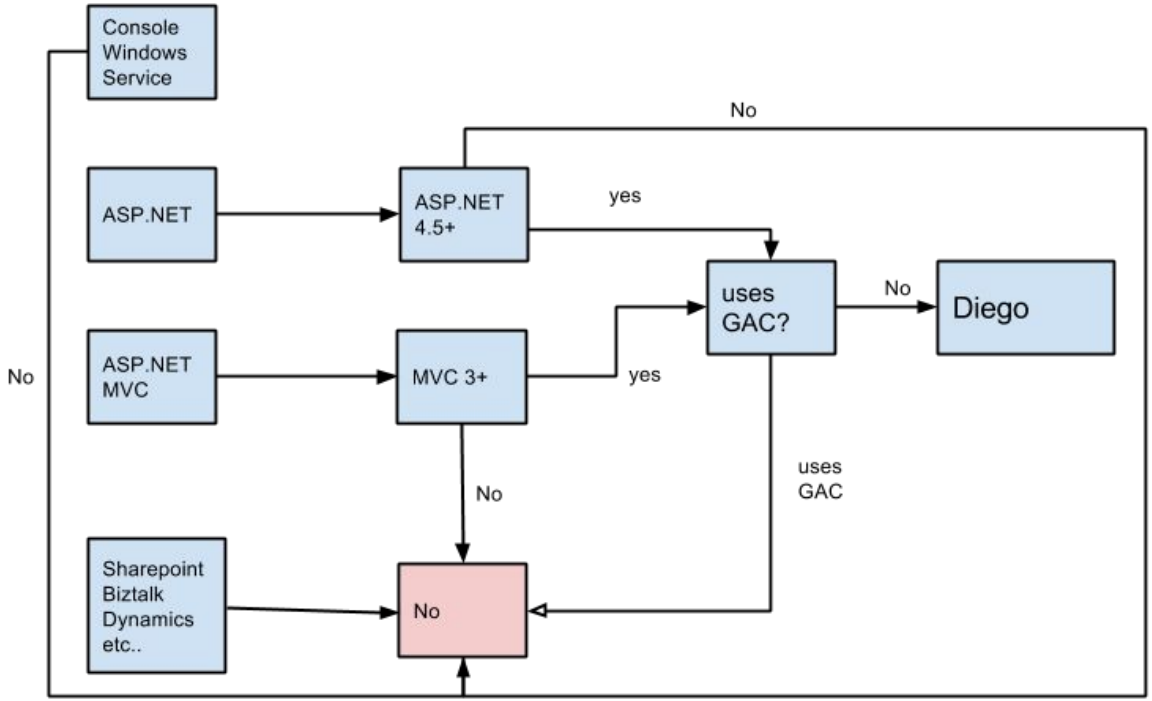
- Distributed transactions, including **XA**
- Multi-minute application startup/shutdown time
- Java properties files/.NET web.config files
- Hardcoded configuration
- Nonstandard security
- Batch Processing - not all batch workloads are problematic
- OS level dependencies
  - Ex: Windows Domain Auth, IBM DB2 driver for .NET

Replatforming Java apps



Pivotal

Replatforming .NET Applications



Pivotal

Start by Replatforming Suitable Applications

Work with One Group; Move “10s” of Apps in 10 Weeks

TYPICAL ACTIVITIES

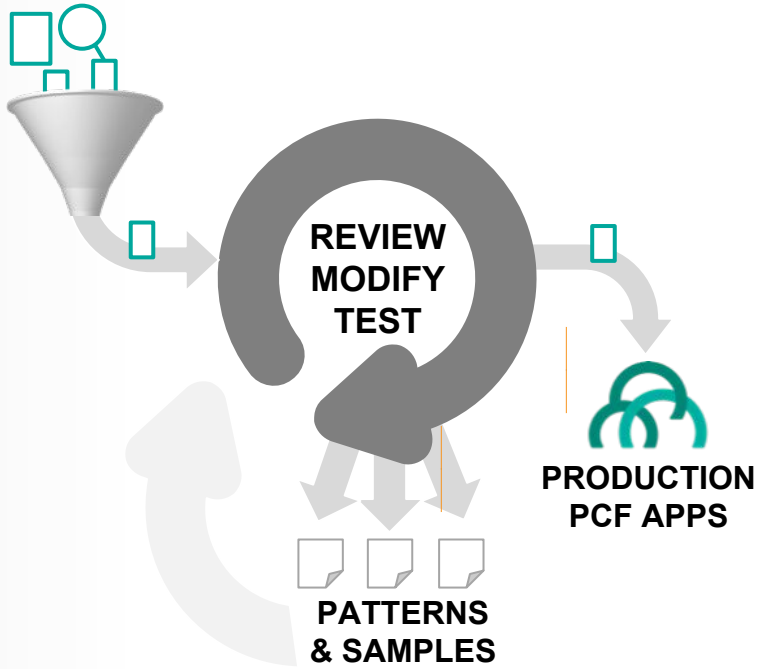
Discovery & Framing
- Suitability Workshops; Technology Planning
- Backlog Development, Grooming & Prioritization

Platform Extensibility
- Buildpack Engineering; Apps Configuration

Test Automation
- Build CI/CD Pipelines; Automate Testing for App & Backing Services (e.g. Connectivity, Perf.)

Refactoring
- Backing Service Location / Configuration
- File System Usage, Messaging, etc.

Process and Documentation
- CI Everything (Including Docs)
- Reference Patterns Informed by Work



Pivotal

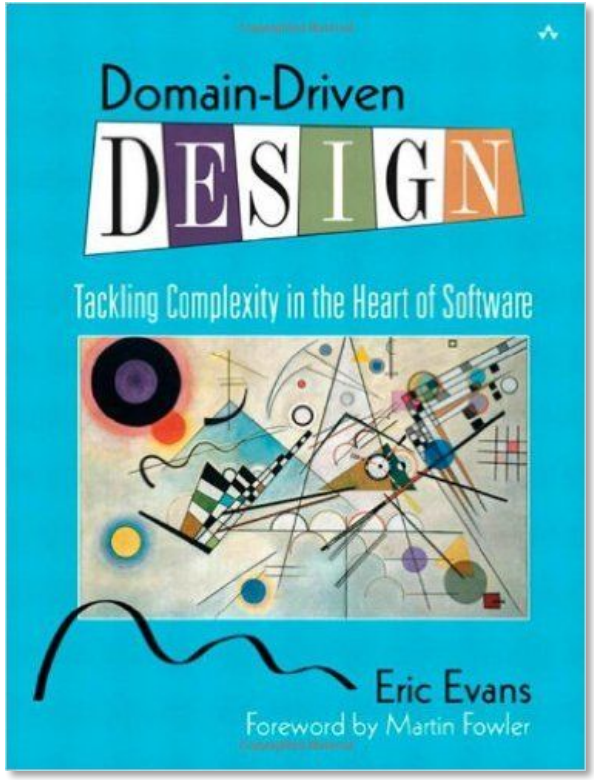
Replatforming - Spring Way

- **Spring Boot** - Modernize dependencies/management
- **Spring Cloud Services -** read configuration settings from VCAP_SERVICES
- **Lowest Common Denominator -** should run on standalone tomcat
- **Executable JAR -** can run in embedded server
- **Clean Profiles -** separate profile for every environment
- **Tune your CI / CD pipeline -** to take full advantage of Spring Boot

Now For Some Theory

Domain Driven Design

- An approach for designing software systems that model the complexity of the real world
- Published in **2004** by Eric Evans
- Practical micro service design methodology



Pivotal

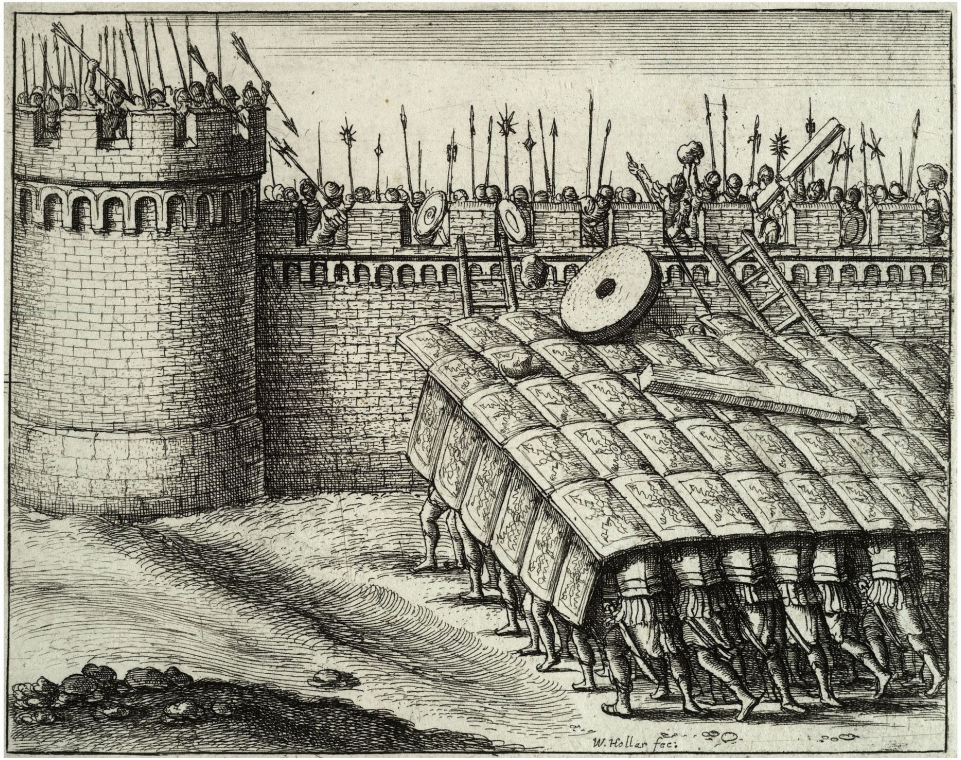
Strategic Domain Driven Design

- Domain
- Subdomain
- Bounded Context
- Context Map
- Ubiquitous Language
- Anti corruption layer
- Shared Kernel
- Open Host Service

Pivotal

Tactical Domain Driven Design

- Entities
- Value Objects
- Aggregates
- Services
- Modules
- Factories
- Repositories



Pivotal

Domain / Subdomain / Bounded Context

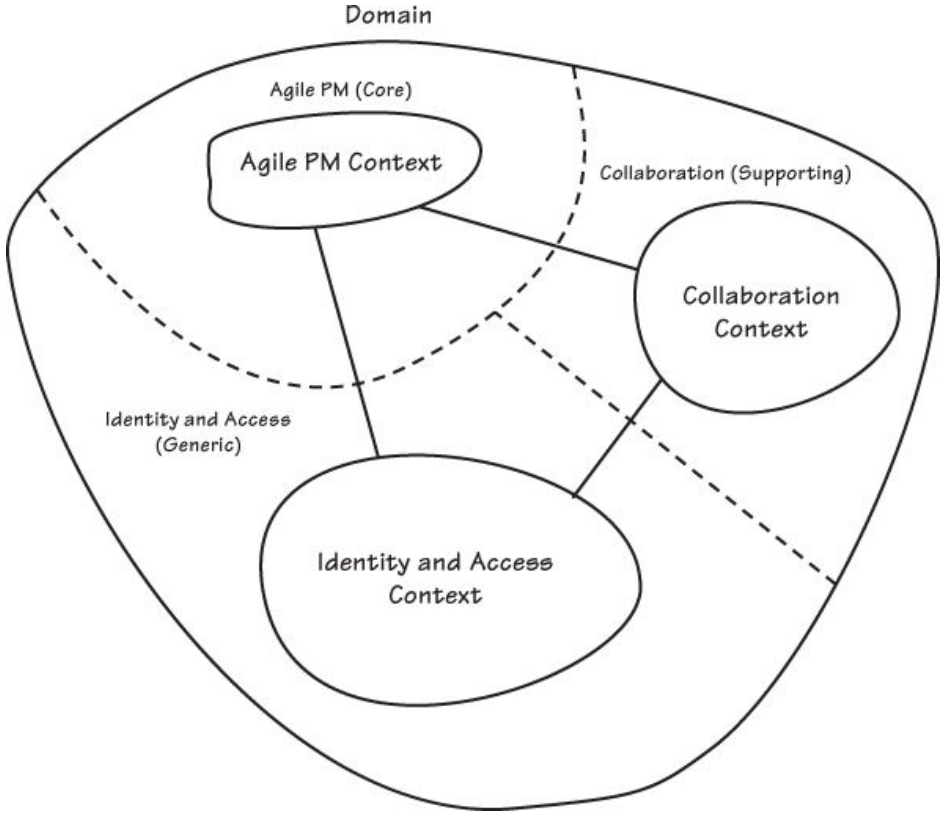
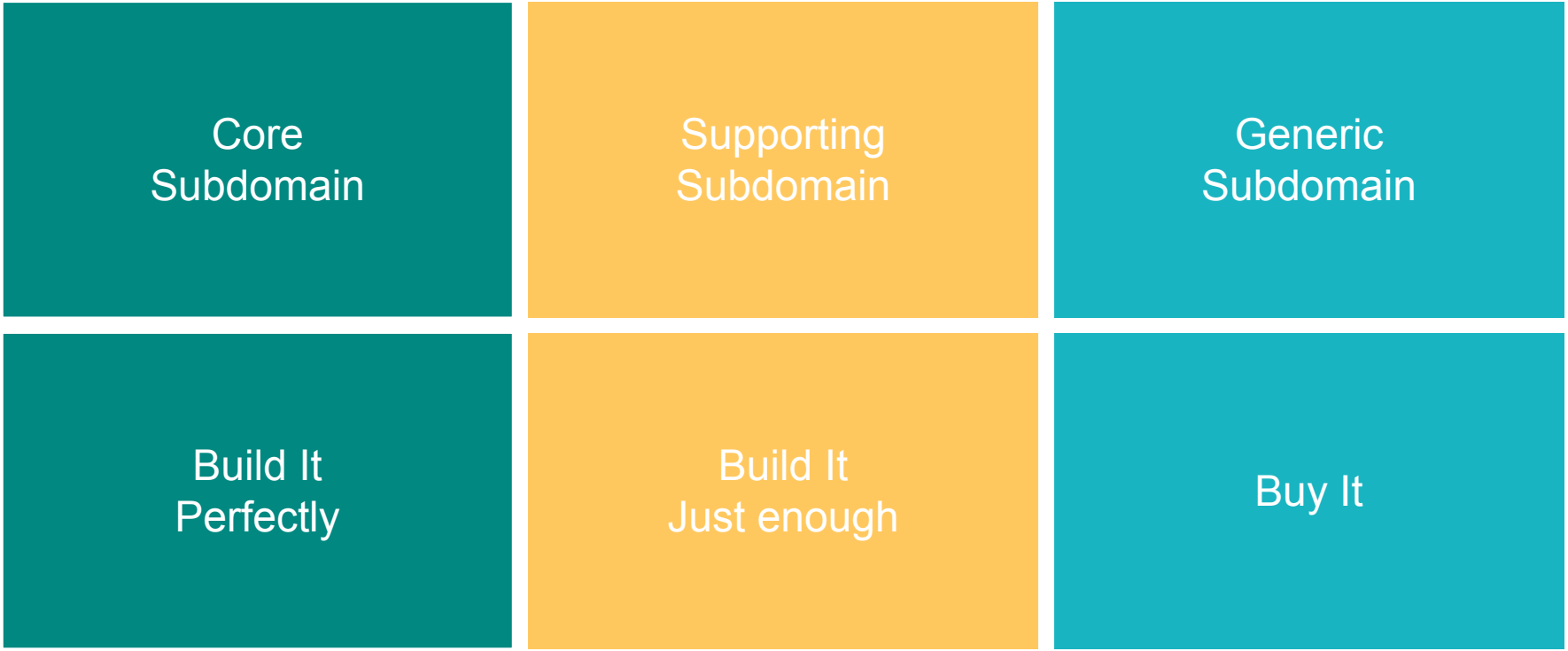


Figure 2.7 from Implementing Domain Driven Design

Pivotal

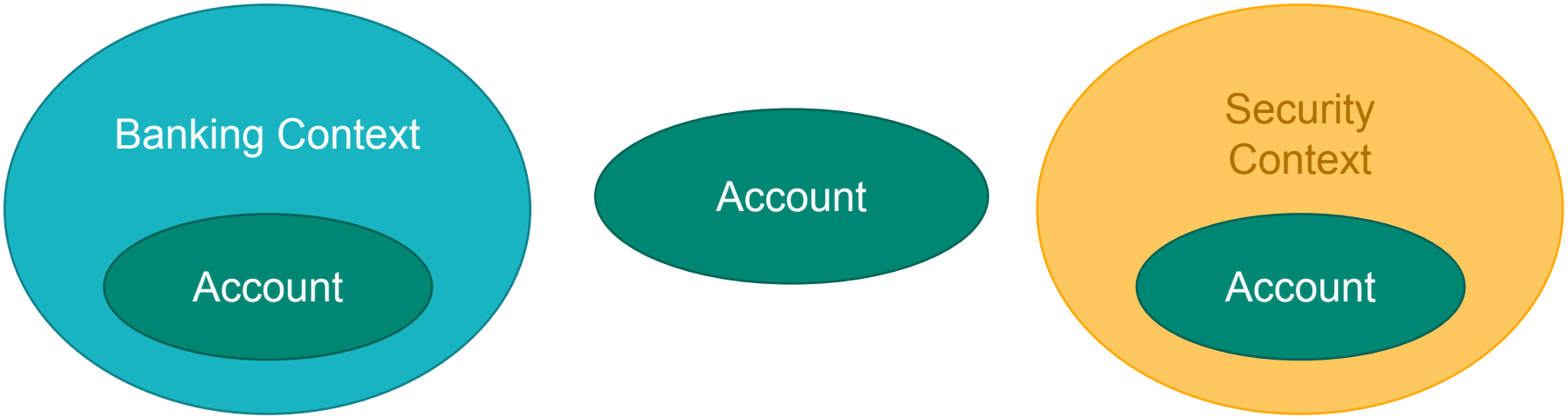
Subdomains

A subdomain is a subset of a domain

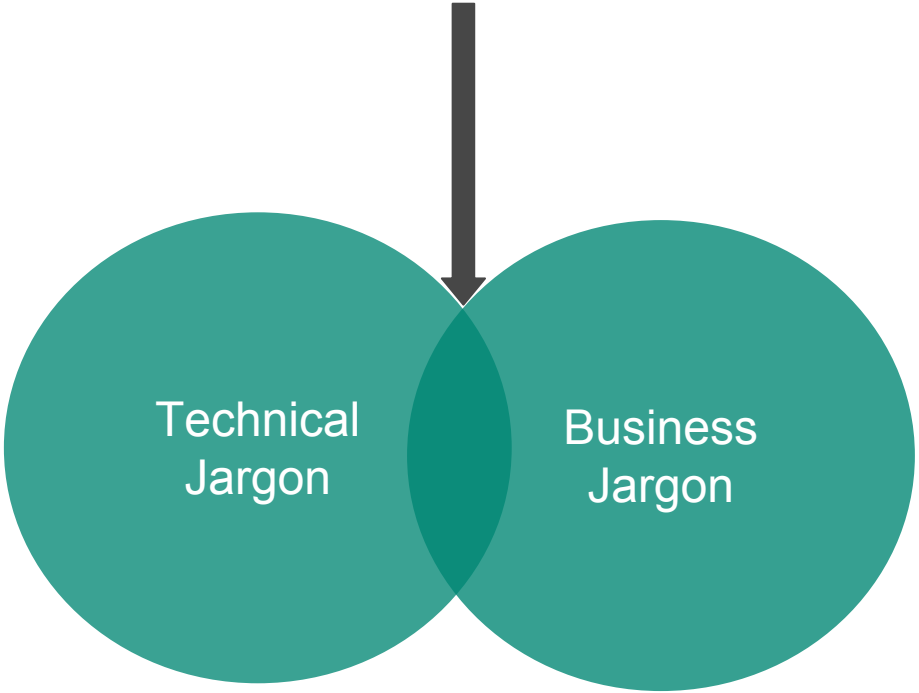


Bounded Context

A boundary inside which the meaning of a model is contained



Ubiquitous Language



Road To Microservices

Pivotal

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

Pivotal

Generic Bounded Context Refactoring Recipe

SpringBoot a Hexagonal Architecture Application - Part 1

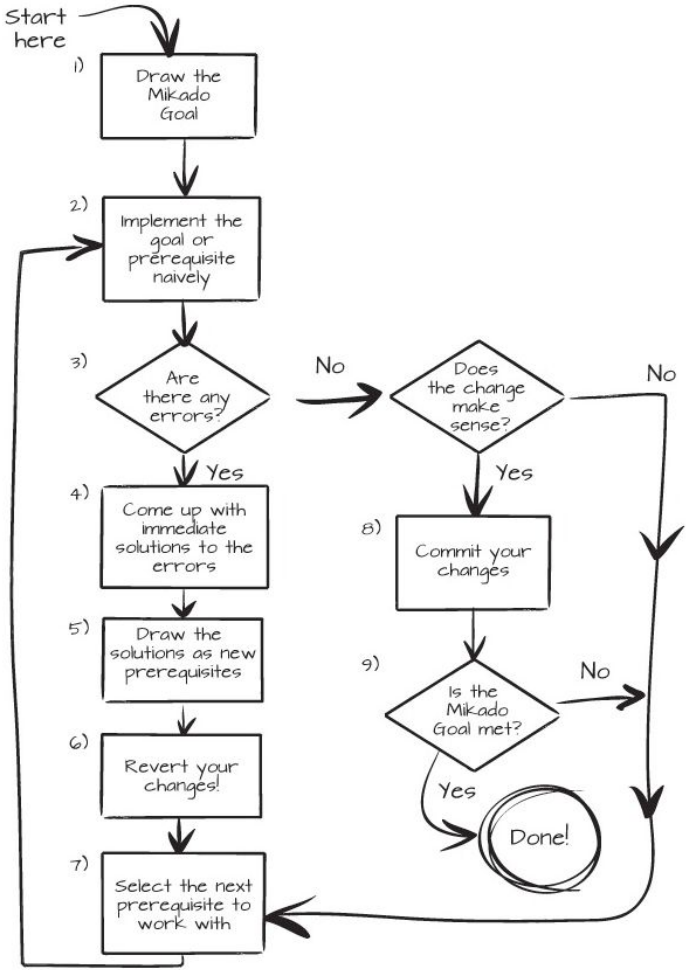
- Search for all the call sites into the bounded context
- Analyze the current interfaces exposed by the bounded context
- Define the required ports for the bounded context
- Analyze external dependencies used by the bounded contexts
- Define the required adapters for the bounded context
- Analyze how the bounded context will stay in sync with the rest of the system

Generic Bounded Context Refactoring Recipe

SpringBoot a Hexagonal Architecture Application - Part 2

- Define what domain events the bounded context will emit
- Define what events the bounded context will listen for
- Copy and paste the code from the old project s
- Create a new spring boot project for hosting the refactored code
- Add the newly configured project to the CI / CD Pipeline
- Write unit and integration tests
- Cut and paste code and refactor it
- Iterate until done

Mikado Method

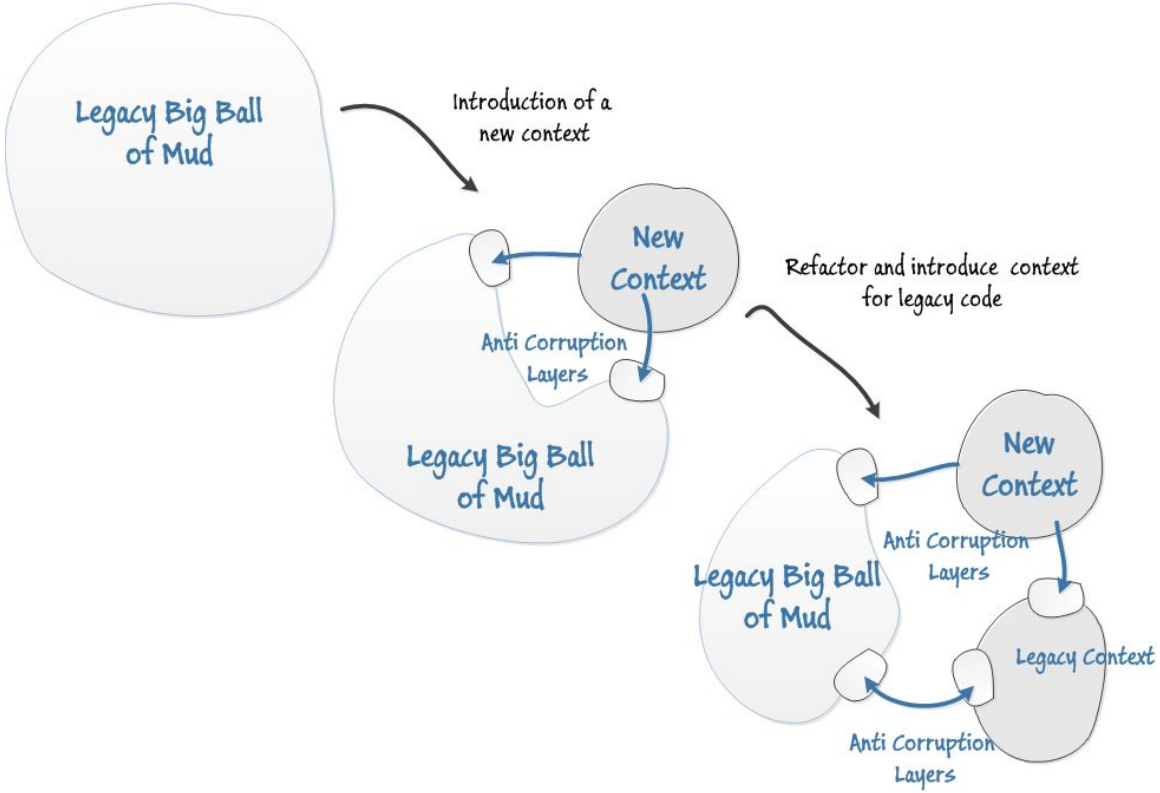


https://www.manning.com/books/the-mikado-method

Pivotal

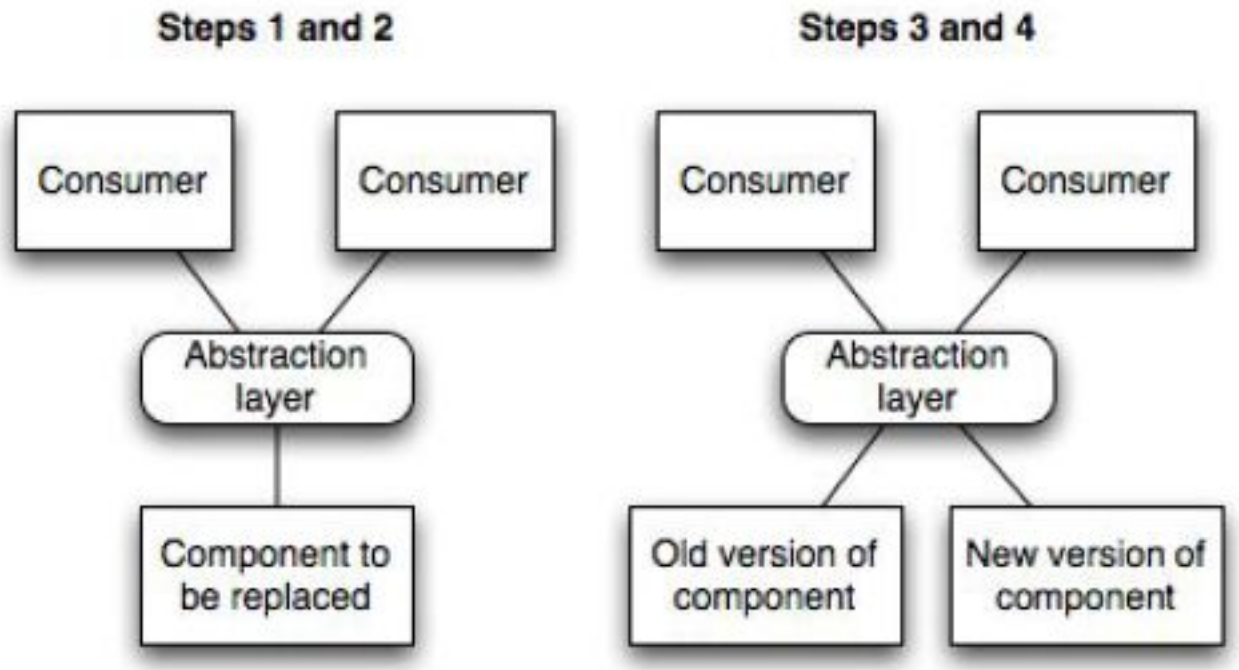
Monolith Decomposition Patterns

Anti-Corruption Layer



https://leanpub.com/Practicing-DDD

STRANGLING THE MONOLITH



Inverse Conway Maneuver

- Application designs will mirror communication structures
- Leads to unintended friction points
- **Evolve team and organizational structure to promote your desired architecture**
- **Break down silos to foster collaboration**

Smart Routing

- Canary Testing
- Dynamic Routing
- Service Migration
- Load Shedding
- Active/Active Traffic Management

Dark Launching/Feature Flags

Wrapping software features in a way that let you turn them on or off

- **Why?**
  - Private beta release
  - Commit code in logical chunks
  - Release a new feature to all users at a specific date
  - Not confident in how stable or how scalable a new feature is
- **How?**
  - **Boolean** – Feature will be on or off
  - **Percentage** – Certain % of Users, Cookie, Random, Group
  - **List** – User ID, Group ID, Organization ID, …
  - **Identity** – Always on! and cannot be turned off.
  - **Nil** - Always off! and cannot be turned on.
- **Cloud Foundry Constructs**
  - cf scale, Configuration Server, Route Services

Migrating Data

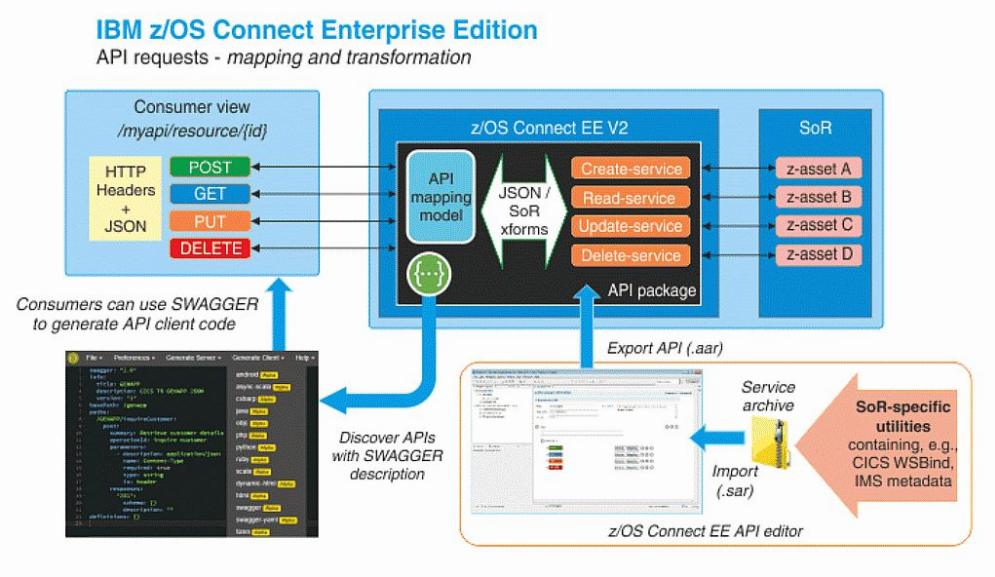
- **Tools – SchemaSpy**
  - Generate graphical table schema
- **Tools - Liquibase, Flyway, jooQ**
  - Auto apply bundles of database refactorings
- Run both the original and new schemas in production during transition
- Expose a Facade service to encapsulate DB changes
- Move logic and constraints to the edge aka services
- Implement retry and compensations
- Read “Refactoring Databases” by Ambler/Sadalage

SOA => Microservices

- **Phased approach to replatform from ESB**
- **Business logic should reside in Java apps**
  - only fundamental ESB functions like legacy adapters and pure transformation and mediation should be handled by the ESB
- **Do not use ESB for greenfield microservice development**
- **5 step evolution of the ESB to the cloud**
  1. Co-exist
  2. Lift & Shift
  3. Refactor
  4. Replace
  5. Transform

Modernizing Batch

- Address Concurrent Batch and Online
- z/OS v2 Connect
- Eliminate needless Data Movement
- Eliminate file transfer and unnecessary app integration
- Scheduling and Job Management
- Technical Solutions
- Leverage Distributed Batch



Pivotal

Refactoring Recipes

Pivotal

Application Level Eventual Consistency

Persist operation state in the microservice and track to success or compensate

- Databases SQL or NoSQL GemFire
- Queues (RabbitMQ, JMS, WebSphereMQ, Tibco .. etc)
- Spring State Machine
- Java 8 Completeable Futures
- “Distributed transactions in Spring, with and without XA” from Dave Syer http://www.javaworld.com/article/2077963/open-source-tools/distributed-transactions-in-spring--with-and-without-xa.html

Foreign Keys Constraints

How are Foreign Key Constraints Validated Across Table is Different Bounded Contexts

- Enforcing Foreign Key Constraints between microservices application level problem not a database concern
- Usage of Immutable Stable URI's to identify Foreign keys can be helpful

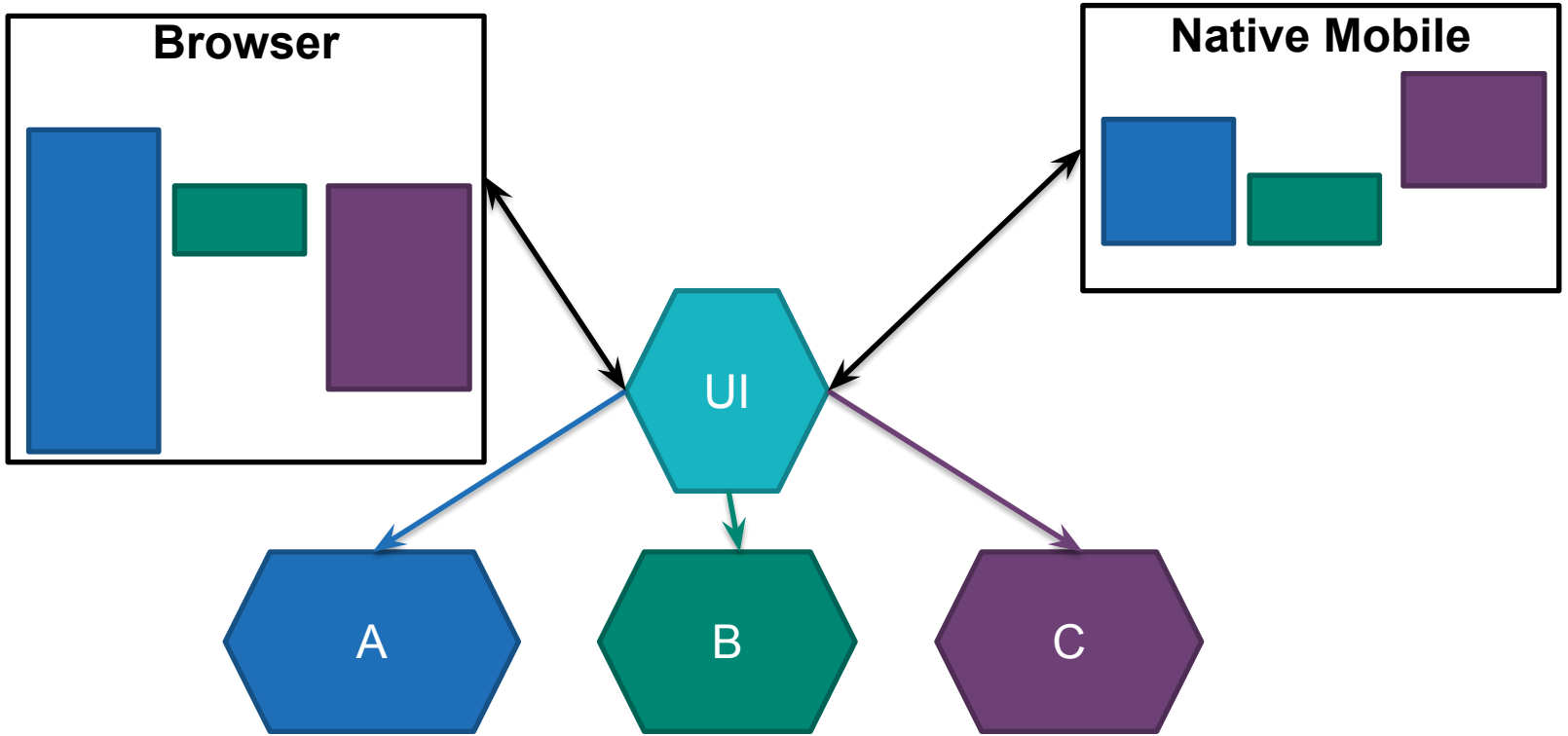
Shared Static Data Becomes Code

Turn Static Shared Data into Code Accessible via dependency manager

```
public enum CurrencyCode
{
    CAD("CAD"), USD("USD"), EUR("EUR");

    private final String isoCode;
    public final Currency currency;
    public final MathContext displayContext;
    public final RoundingMode ROUNDING_MODE = RoundingMode.HALF_EVEN;
    public final MathContext computeContext = new MathContext(6, ROUNDING_MODE);

    private CurrencyCode(String isoCode)

    public String getCode()

    public static CurrencyCode parse(String isoCode)

    public BigDecimal round(BigDecimal amount)
}
```

Monolithic Edge UI Gateway

UI Microservice exposed to end users and have it serve up the UI?

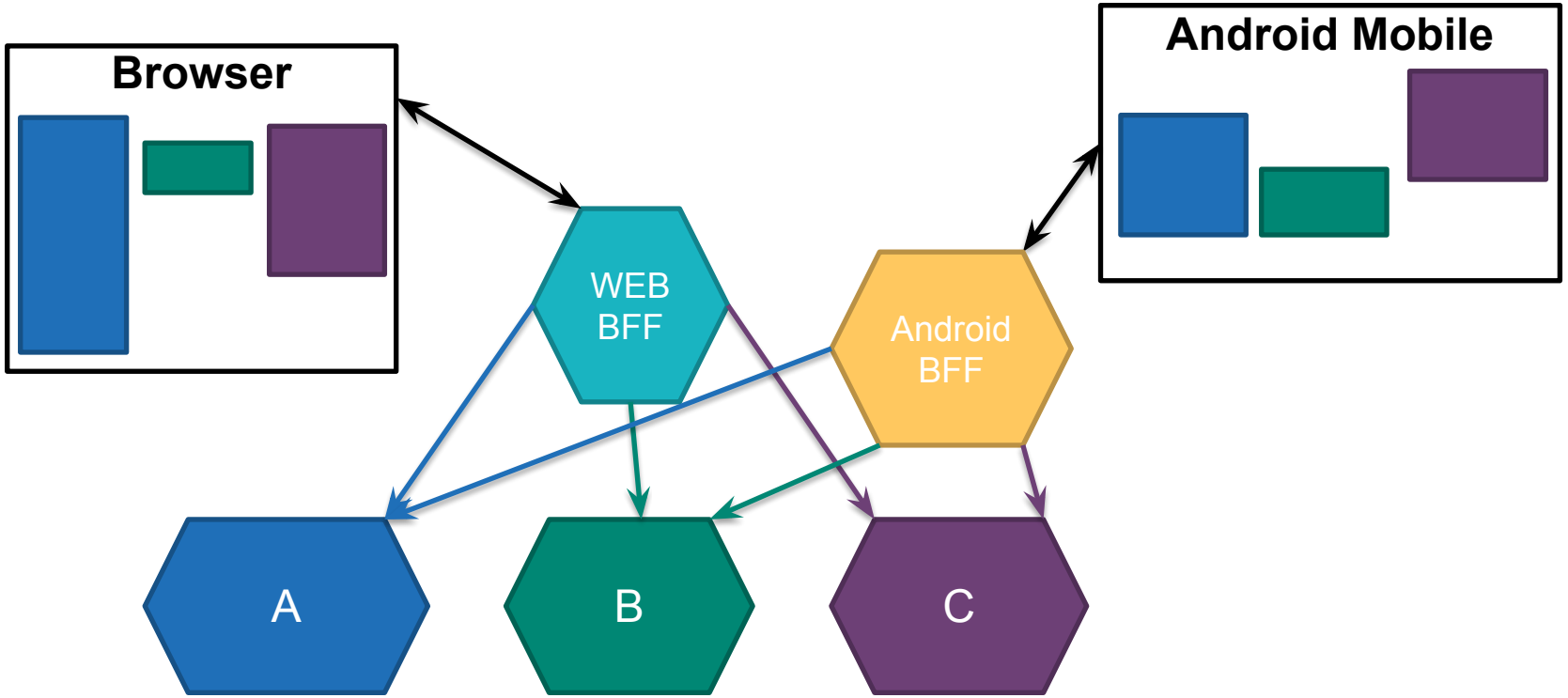


Pivotal

Back End For Front End

Extend each UI experience with a dedicated backend component for UI

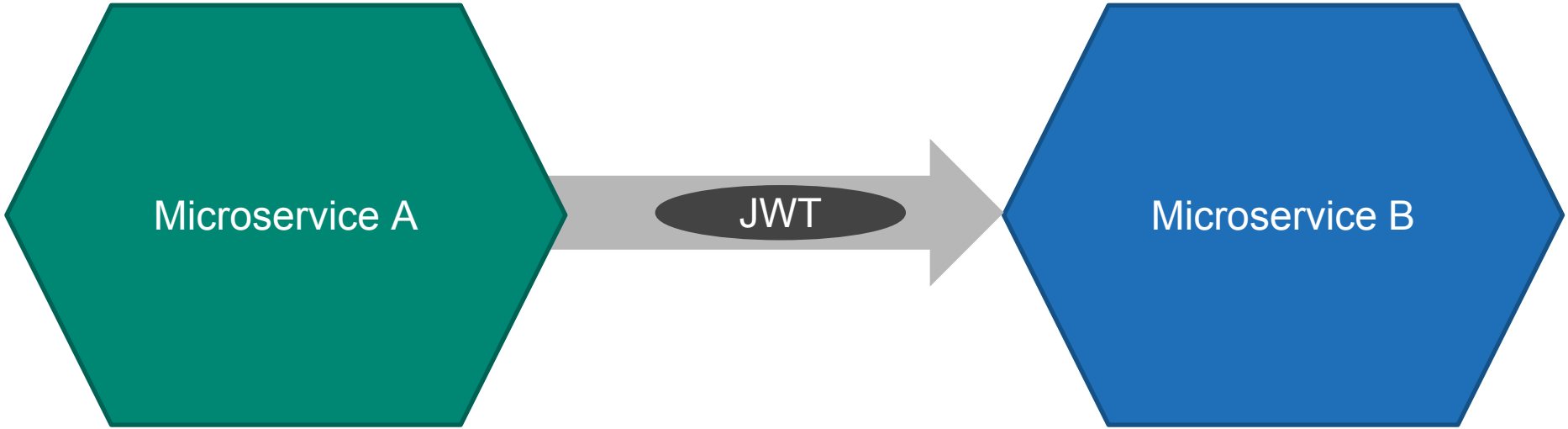
http://samnewman.io/patterns/architectural/bff/



Pivotal

Use JWT Tokens

Use JSON Web Token to pass user info between microservices

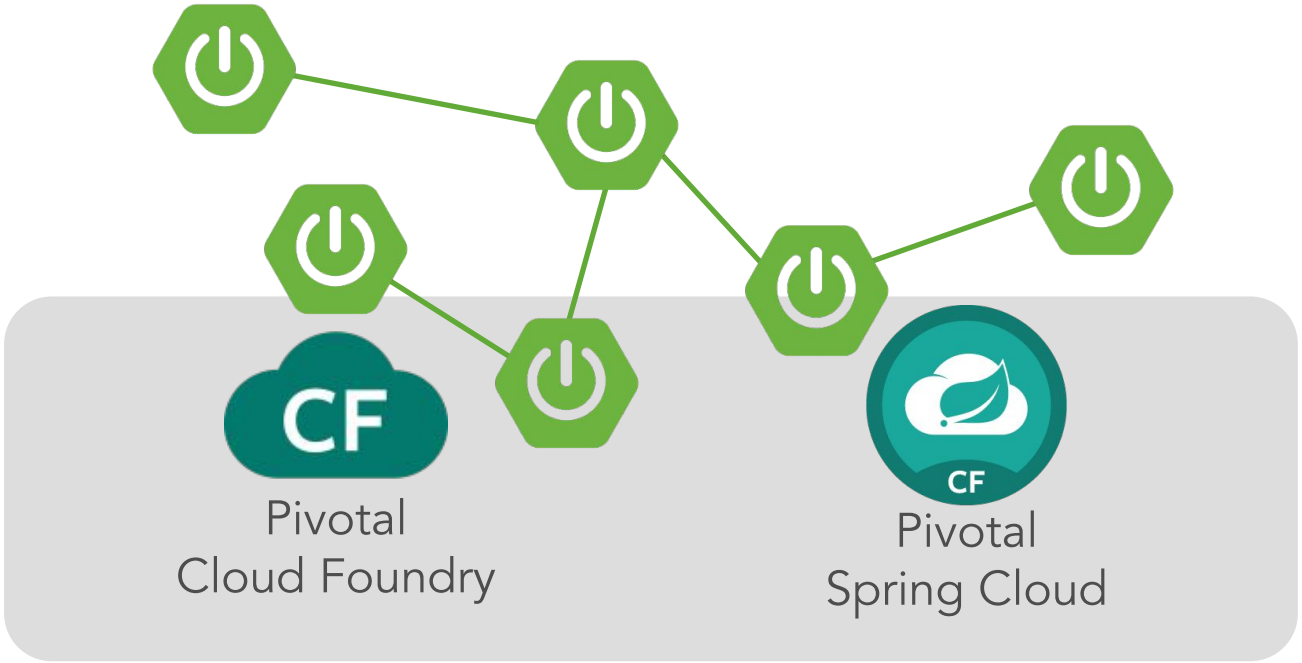


JSON Web Tokens are an open, industry standard RFC 7519 method for representing claims securely between two parties.

Pivotal

Distributed Systems are Hard!

Pivotal has got you covered!

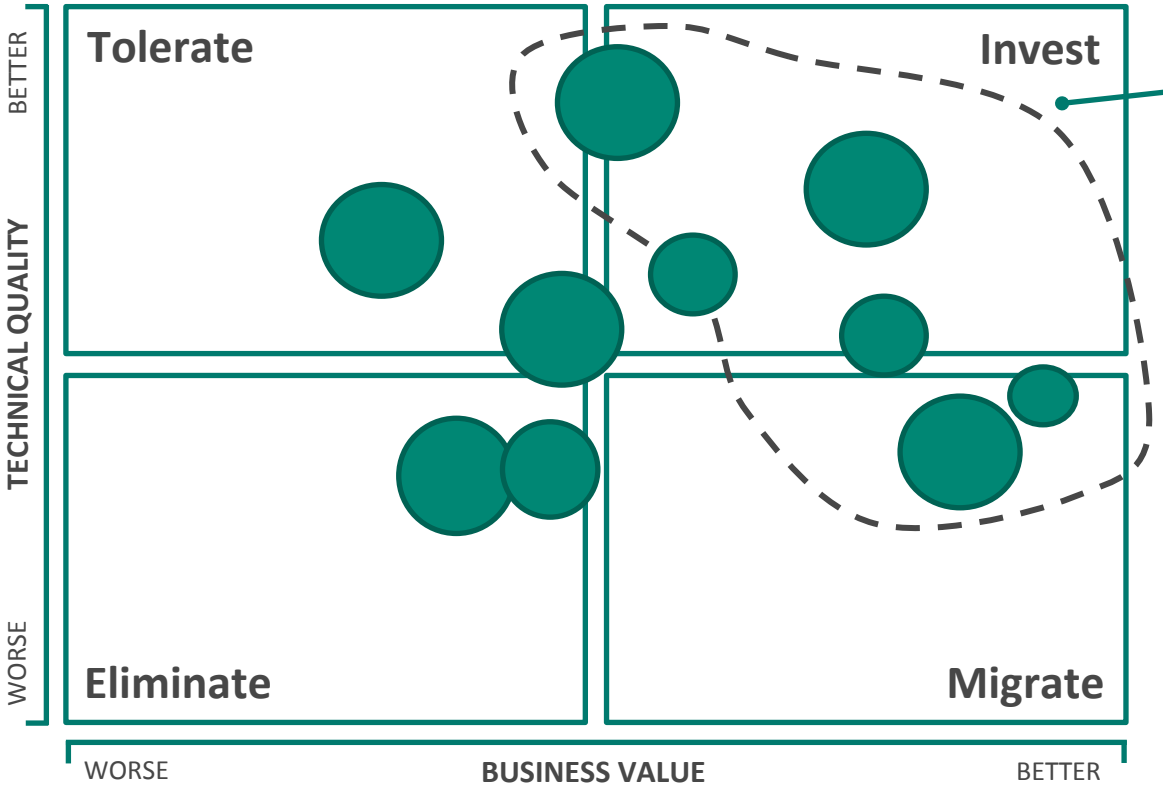


Replatform Workshops

Platform Architecture

Pivotal

TIME Methodology



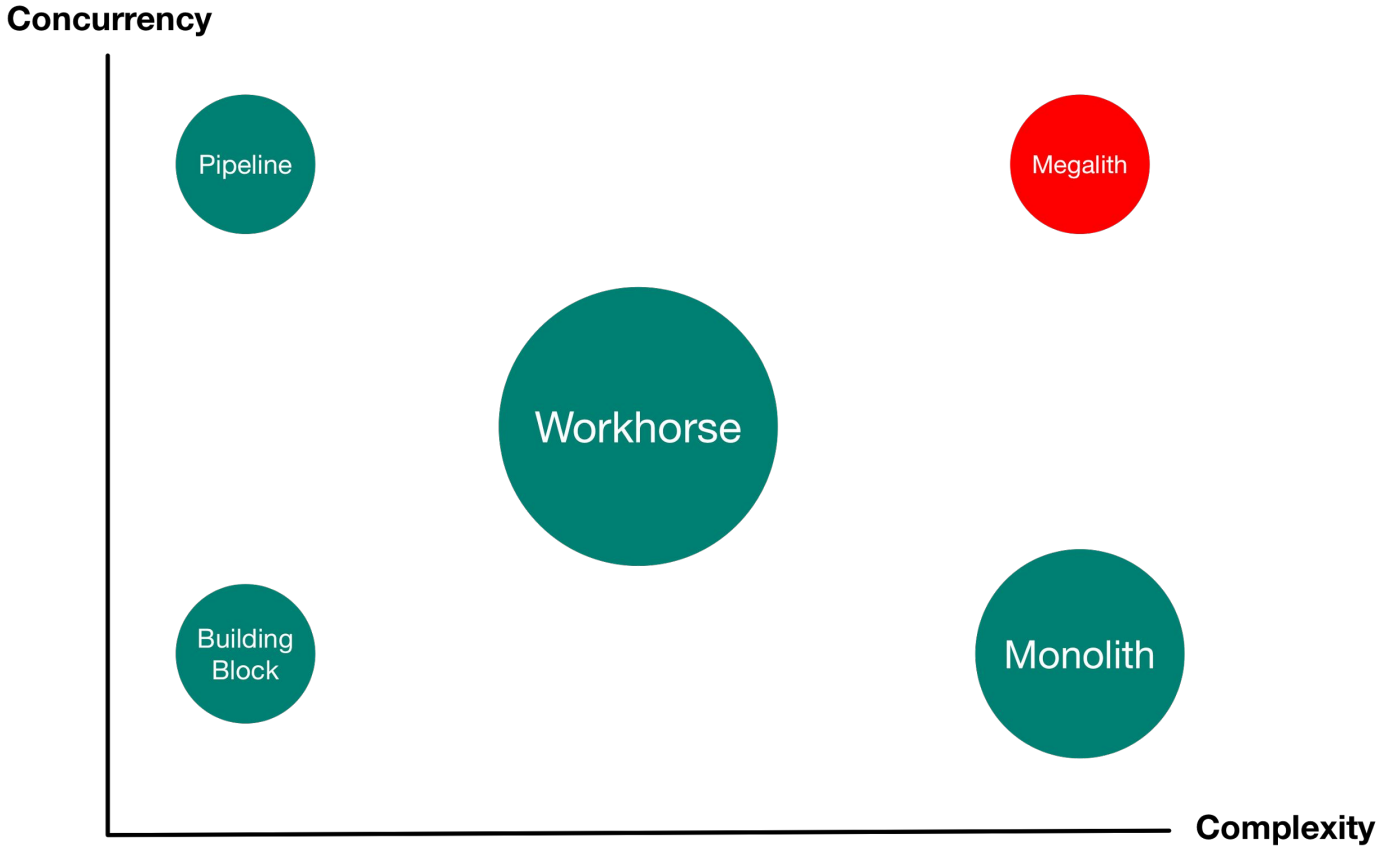
Identify top 10s list

TECHNICAL QUALITY - Technical Debt Level
BUSINESS VALUE - Revenue / Cost Impact

* Gartner's TIME methodology for Application Portfolio Rationalization

Pivotal

Workload Categories

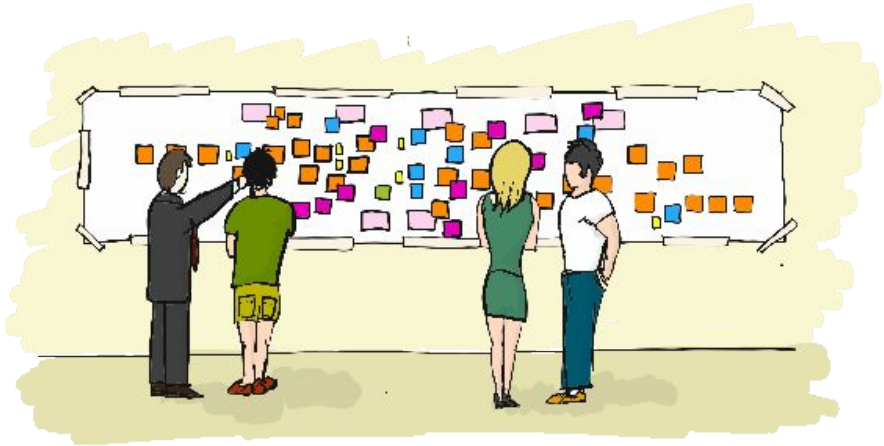


Pivotal

Event Storming

1. Invite right people
2. Large modeling space
3. Explore Domain Events
4. Look for Aggregates

Event Based Modeling Workshop



Replatform Questionnaire

- **3 Modes**
  - Quick, Standard, Detailed
- **5 Red Zone Questions**
  - Fast qualify in/out
- **Polyglot**
  - Java, .NET, Other
- **45 Questions Total**
  - We don’t need to answer all of them
  - Each app should take 15-30 minutes
  - Pivotal will share findings and suggest next steps for each app

Architecture Reviews

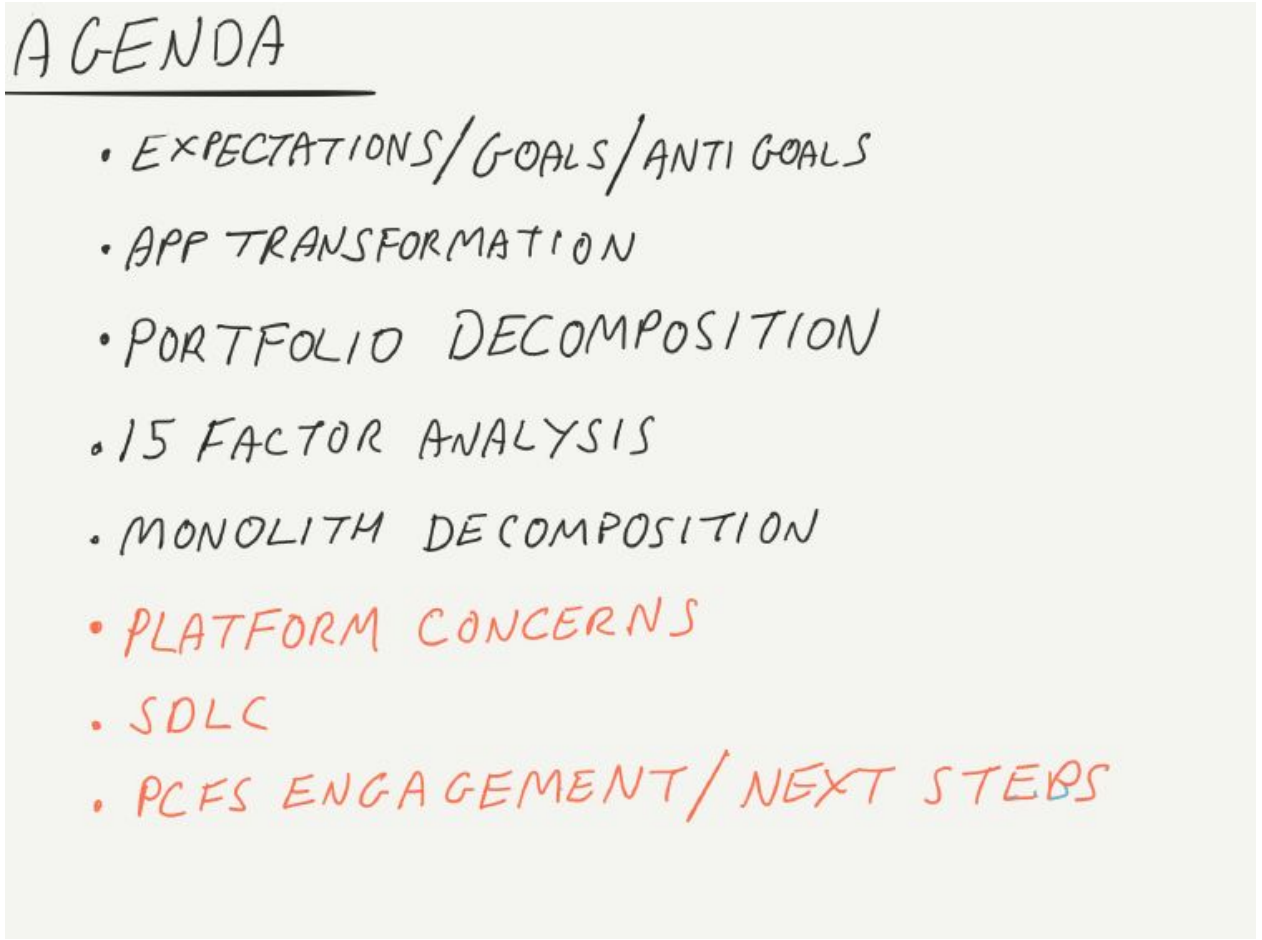
- **Enterprise Architecture Review**
  - Build System
  - Dependency Tree
  - Key Patterns & Frameworks
  - User Flows & Transaction Flows
- **Static Code Analysis**
  - Sonar, Ephemerol, FindBugs
  - Unlikely but useful
- **Outcome**
  - Pivotal will provide an Application Architecture Blueprint PDF
  - Customer Time Investment: 1-8 hours

Scoping Checklist

Customer

1. **Conduct pre-scoping workshop**
   a. Explain scoping checklist
   b. Provide scoping overview
2. **Create candidate “goal” list with customer**
   a. Verify current state / future state suggestions
   b. Refine goals with customer prior to scoping
3. **Secure attendance commitment from**
   a. Application Development
   b. Application & Platform Operations
   c. Data & Integration(DBA, ESB…)
   d. Line of Business

Agenda


AGENDA
• EXPECTATIONS/GOALS/ANTI GOALS
• APP TRANSFORMATION
• PORTFOLIO DECOMPOSITION
• 15 FACTOR ANALYSIS
• MONOLITH DECOMPOSITION
• PLATFORM CONCERNS
• SDLC
• PCFS ENGAGEMENT/NEXT STEPS

Expectations

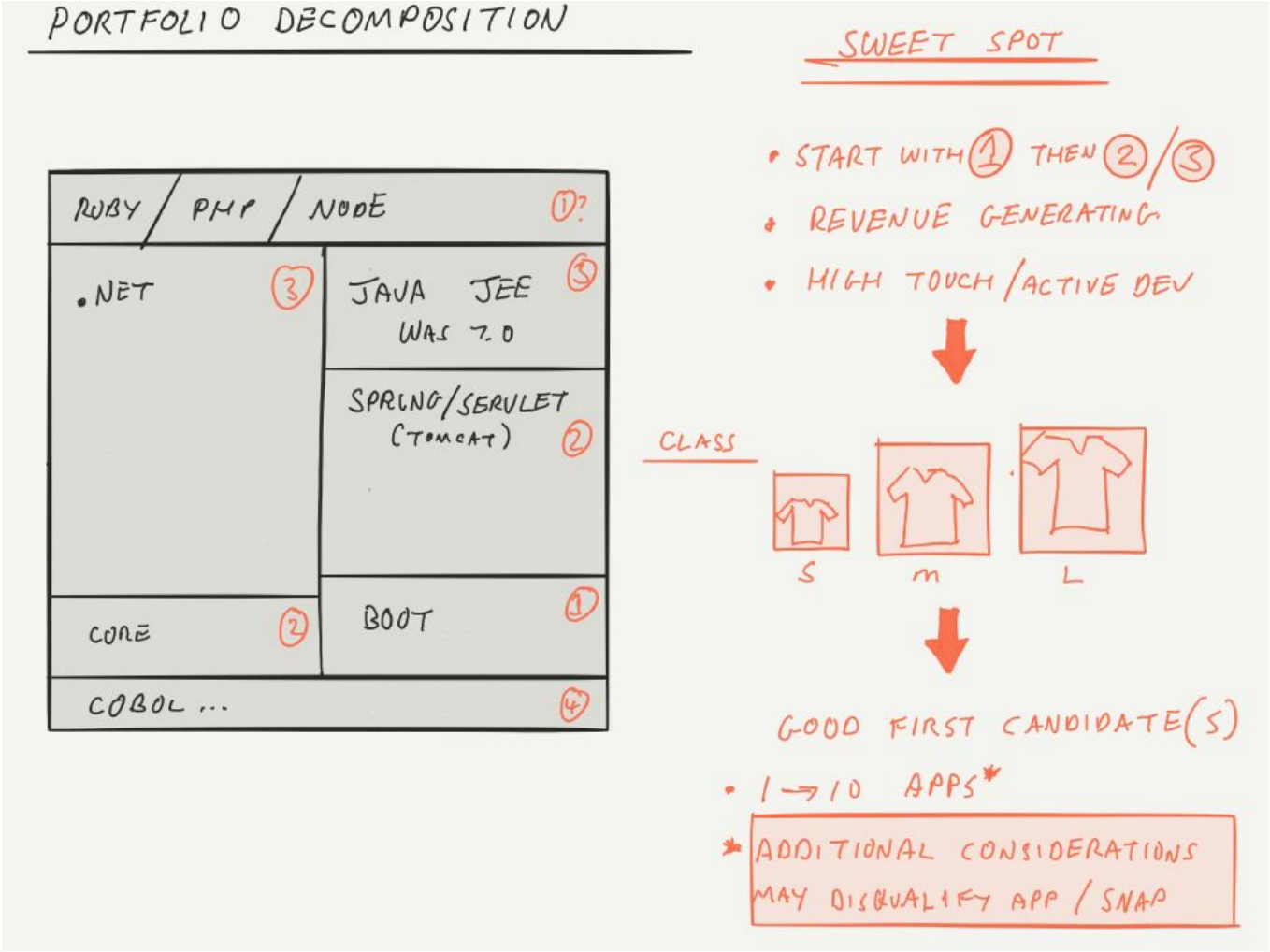
EXPECTATIONS

- PURPOSE OF SCOPING / 3HRS
- PARTICIPANTS (AGREE UPFRONT): APP DEV / APP OPERATOR
  - LOB
  - SERVICE TIER
  - PLAT OPERATOR
  - PLAT CHAMPION
  - PLAT DEV

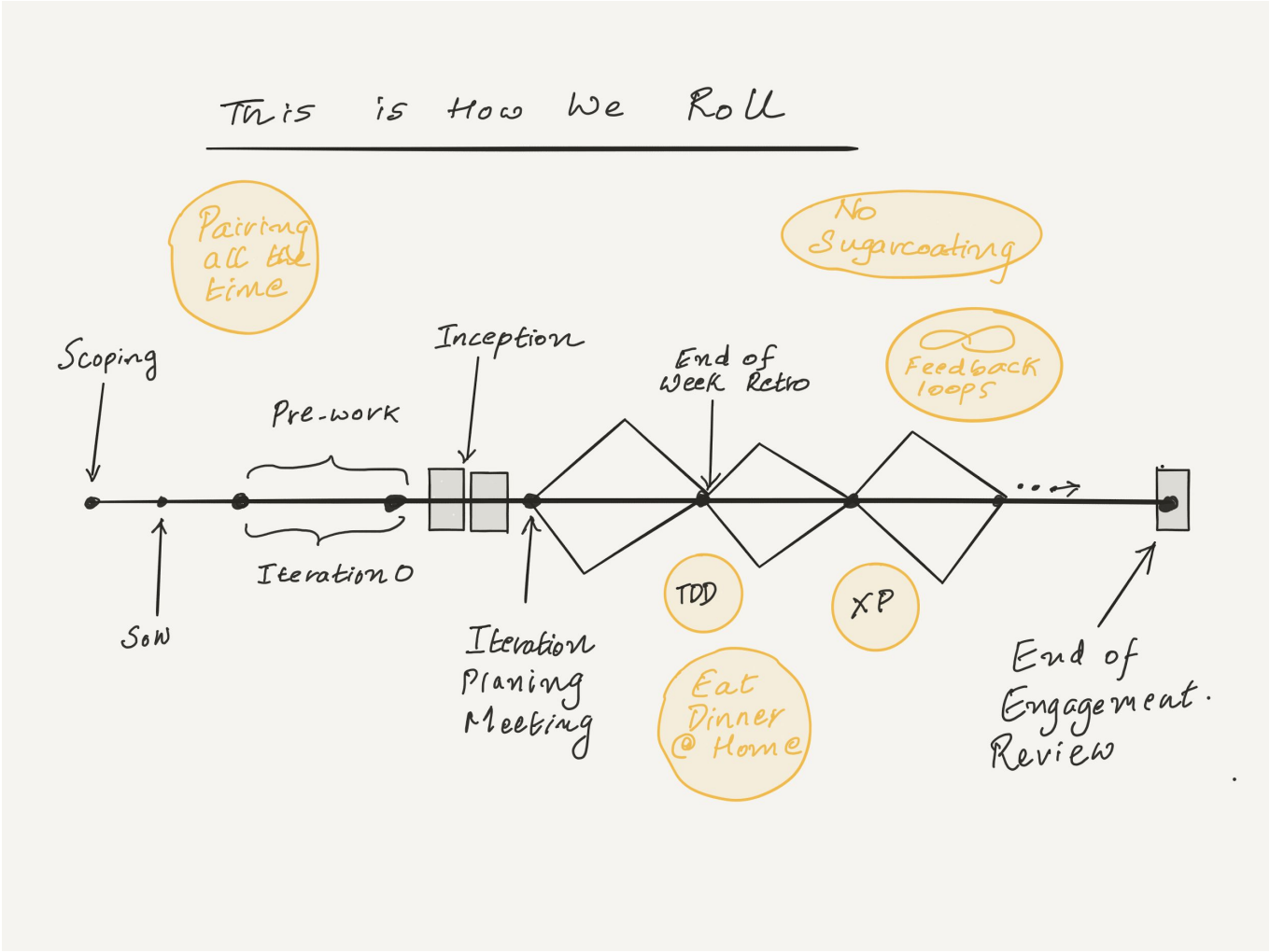
APP DEV	APP OPERATOR
PLATFORM DEV	PLATFORM OPERATOR

- GOALS / 3 MAIN CUSTOMER OBJECTIVES:
  - MIGRATE 1 APP TO PROD
  - ASSESS PORTFOLIO
  - TRAIN TEAM
  - ESTABLISH COE
- ANTI GOALS / OFF THE TABLE
- VOCAB (OOD) RUNS / RUNS WELL / CLOUD NATIVE
- ROI SELECTION > MIGRATION COST / MEAN TIME TO RECOVERY / ONGOING MAINT

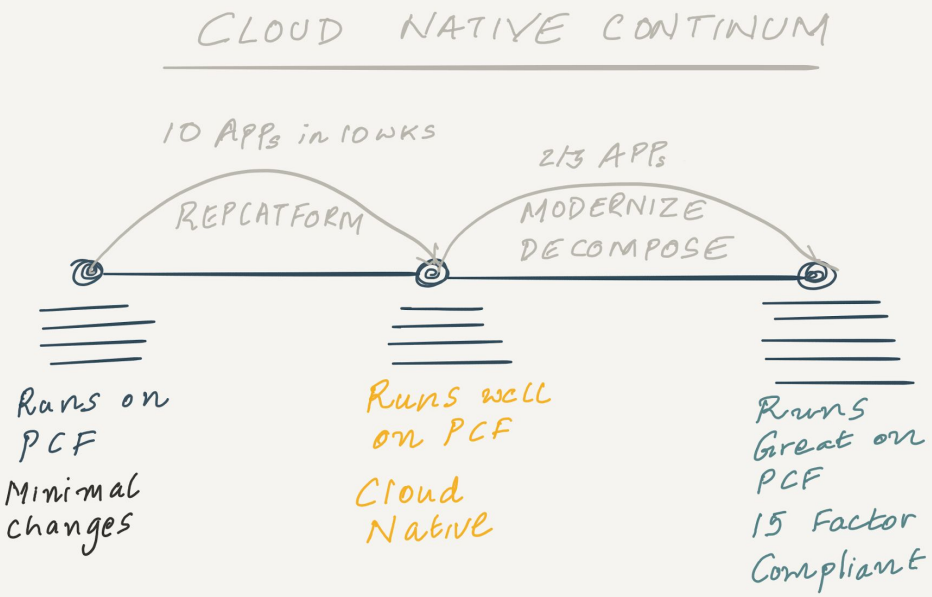
Portfolio Decomposition



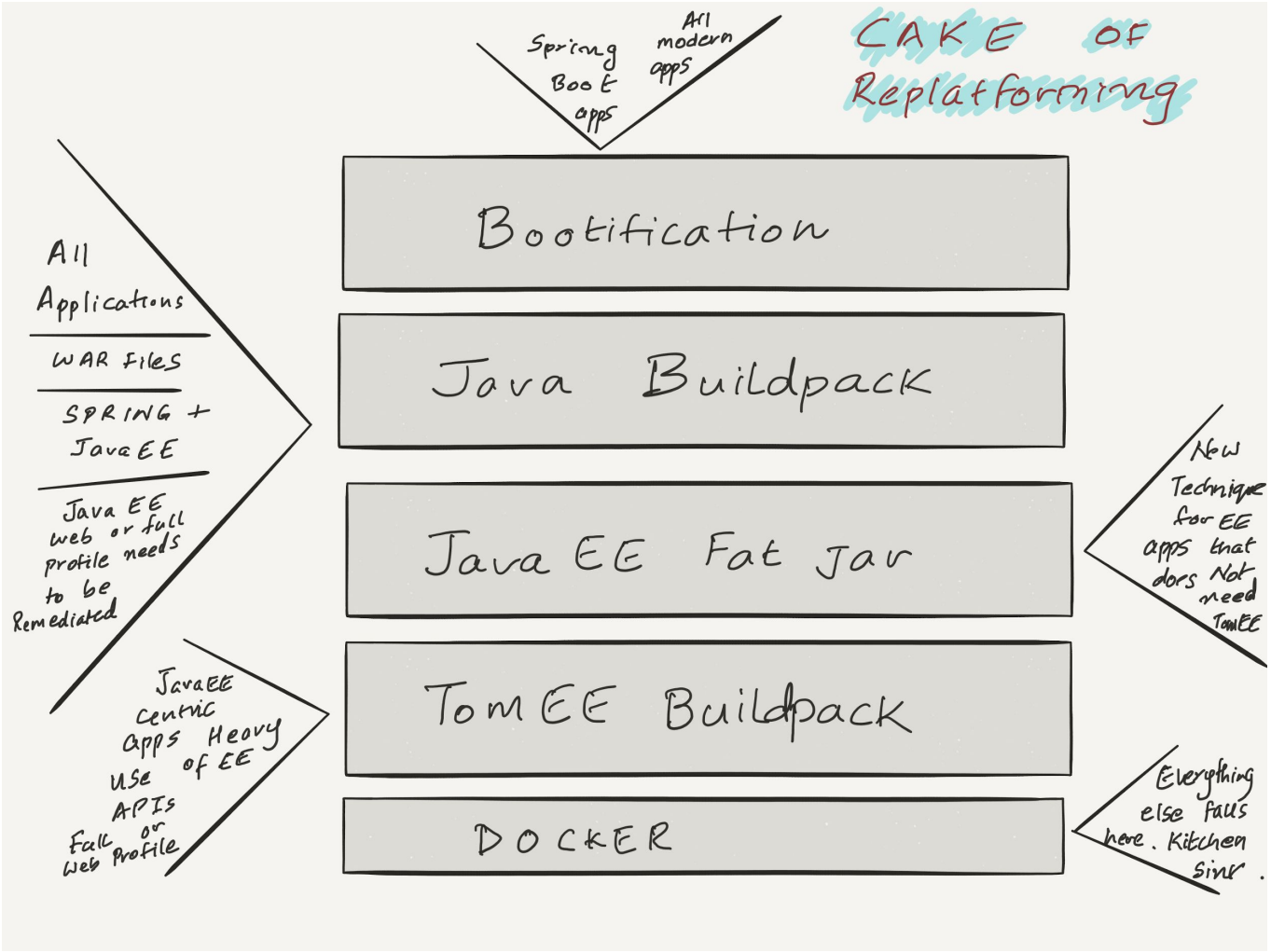
Iteration Timeline



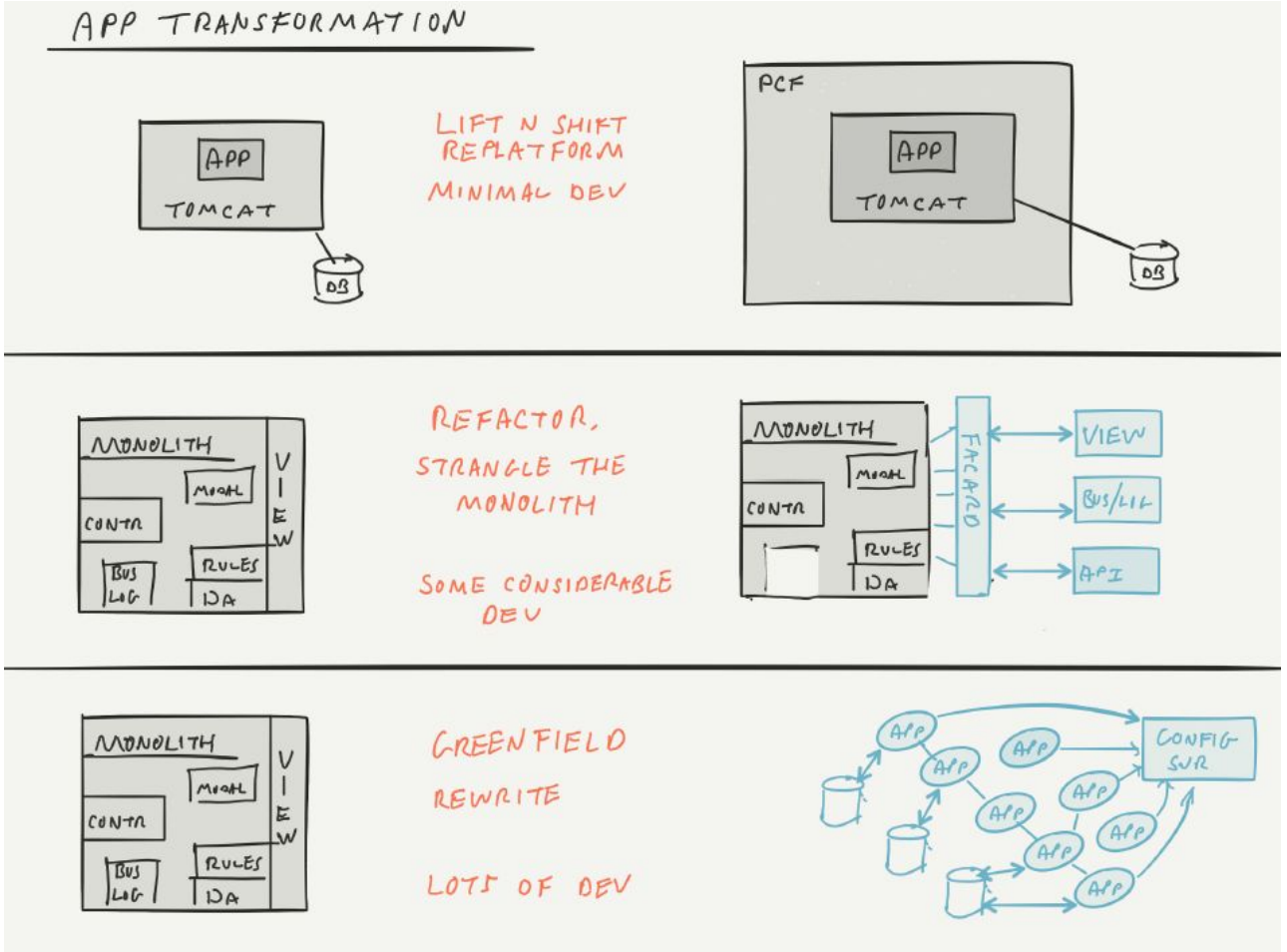
Cloud Native Continuum



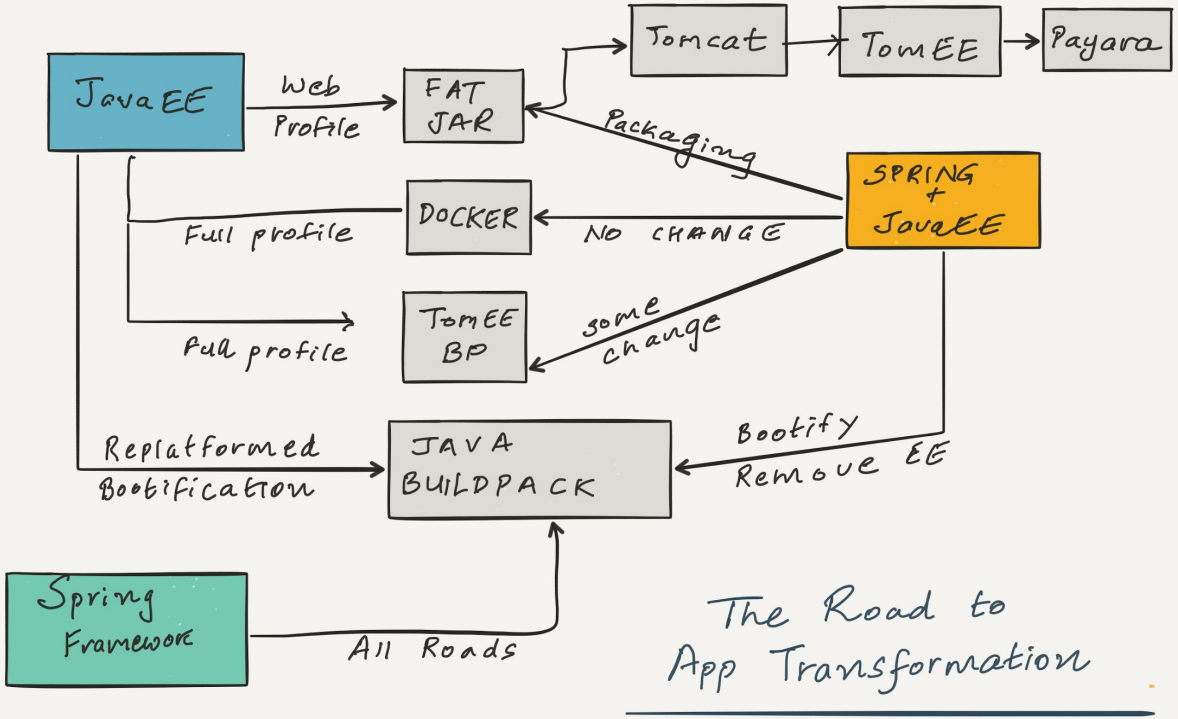
Java Replatform Options



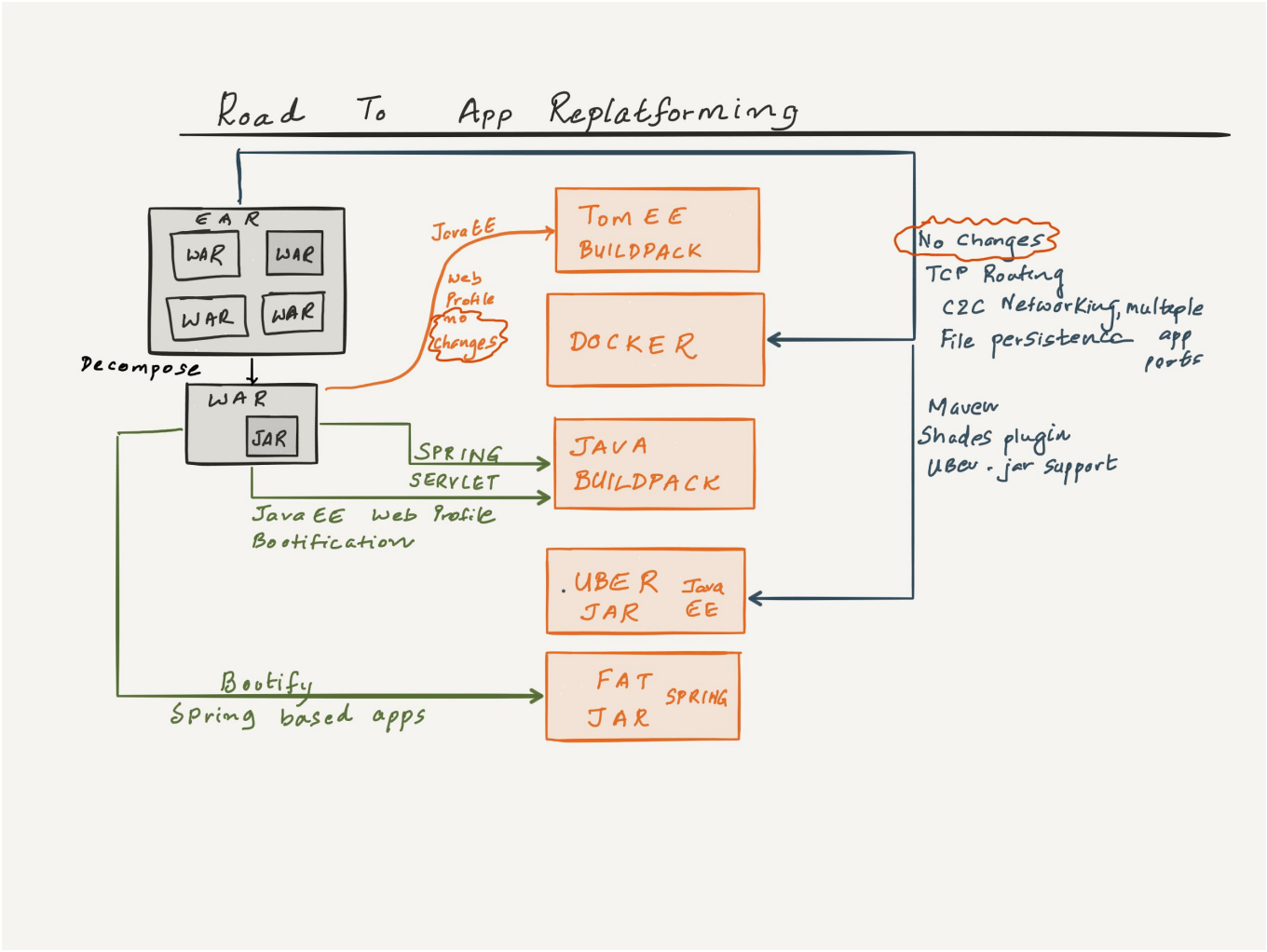
App Transformation



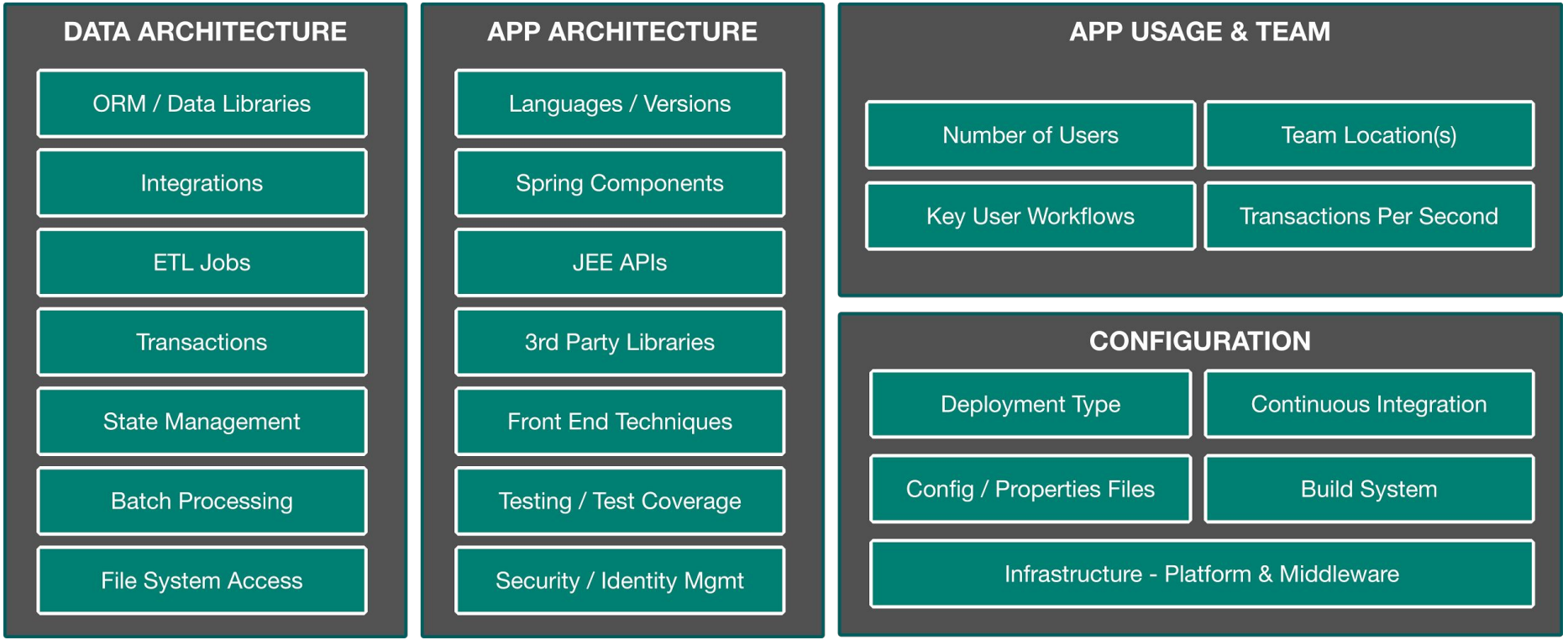
Java Decision Tree



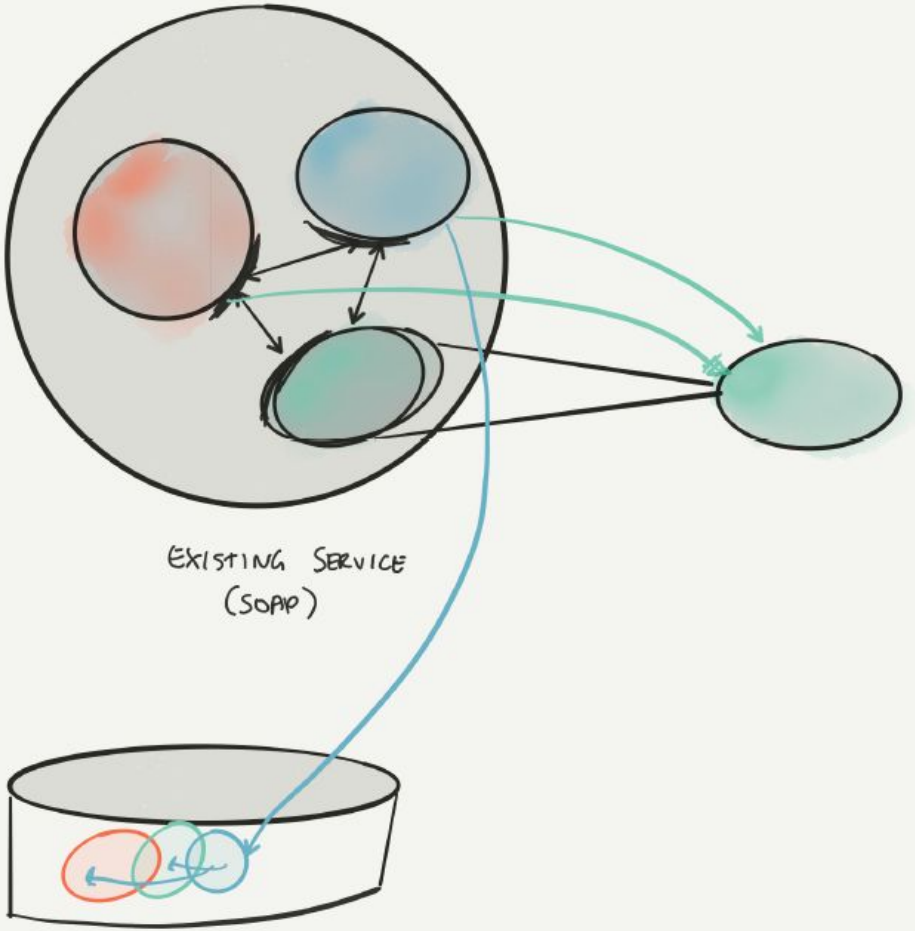
Java Monolith Decomposition



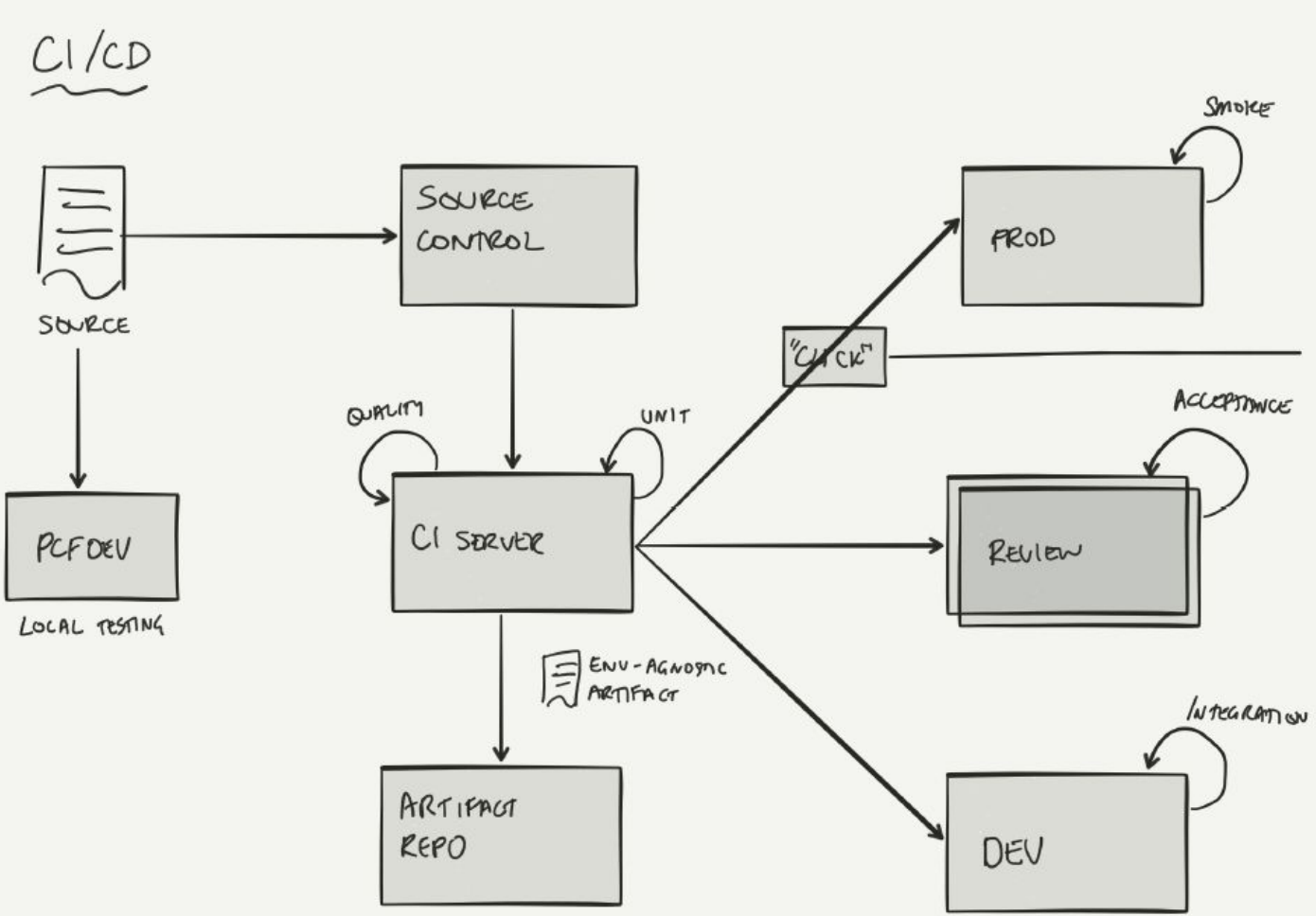
Snap Analysis



External Services



CI
CD


CI/CD
SOURCE
SOURCE CONTROL
PCF DEV
LOCAL TESTING
QUALITY
UNIT
CI SERVER
ENV-AGNOSTIC ARTIFACT
ARTIFACT REPO
SMOKE
PROD
"CI CK"
ACCEPTANCE
REVIEW
INTEGRATION
DEV