Pivotal

企业现存应用的云原生改造之路
—— 规划、方法与实践

高松
AppTX 亚太区项目经理
2019年9月

企业中存在着数量众多的遗留软件系统
支撑着重要的业务

传统/老旧的技术架构和平台
无法满足新时代的业务和IT运维需求

迁移到现代化的云平台（PaaS）
重构为云原生（微服务）架构

高松 - Program Lead，AppTX APJ

Adapting & Developing Tools

Transform Existing Business Critical Applications to the Cloud

Pivotal App Transformation

Transforming where the world runs software

5+ Years	80+ Transformation Engagements	~1,000 of Transformation Recipes	Recognizable Customers

Pivotal

日程

- 如何制定企业云原生改造路线图
  - 如何确定应用改造的业务价值
  - 如何确定应用改造技术复杂度（成本）
  - 综合价值和成本，制定改造策略和优先级

- 如何快速拆解一个单体应用为微服务
  - 使用Event Storming快速梳理业务
  - 形成初步的微服务架构设计图
  - 形成微服务描述文档

如何制定企业云原生改造路线图

Pivotal

从分析每个应用改造带来的业务价值开始

花太长时间决定
该先做哪个应用

如何评估
改造带来的价值

决定太过主观

大量的会议、
效率低下

太多的业务负责人

太多的讨论回合

不同目标的相互冲突

我们需要一个快速评估应用业务优先级的方法

和企业的战略目标一致

可以为应用在每个业务目标维度打分

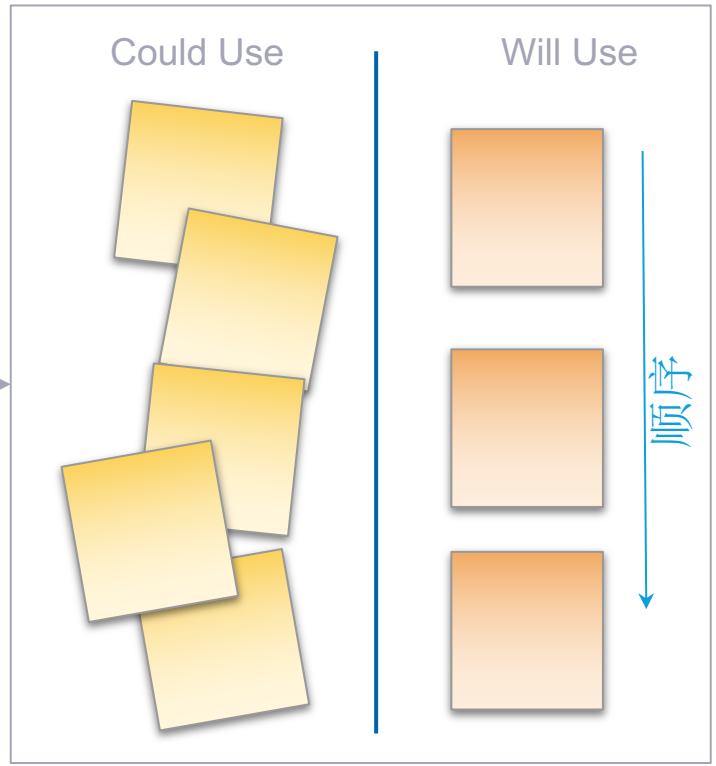
建立一个“漏斗”来筛选应用

这些是我们的战略目标
提高客户体验
提高敏捷性 ...

头脑风暴
分组
定义

业务指标

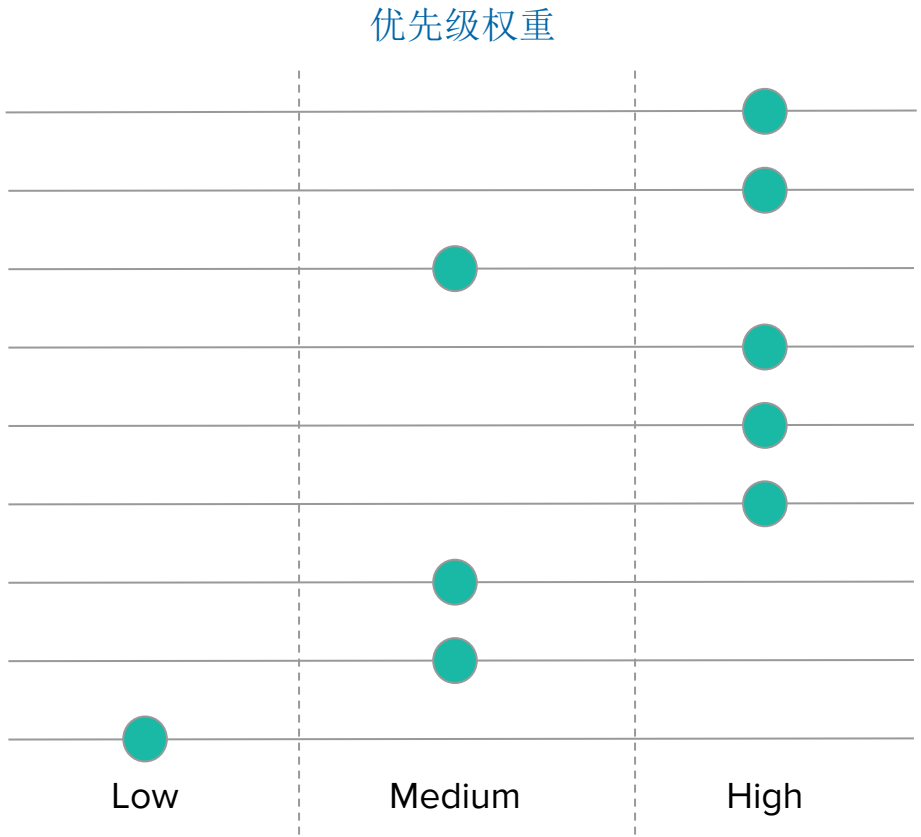
头脑风暴 决定



“指标的相对重要性有变化怎么办？”

“这套指标是否可以重复使用？”

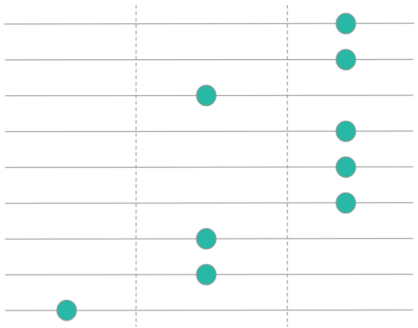
- 客户体验
- 业务不断变化
- 节约成本
- 迁移带来的财务风险低
- 市场竞争力
- 预算
- 对其他应用的价值
- 有利于降低复杂性
- 合规需求

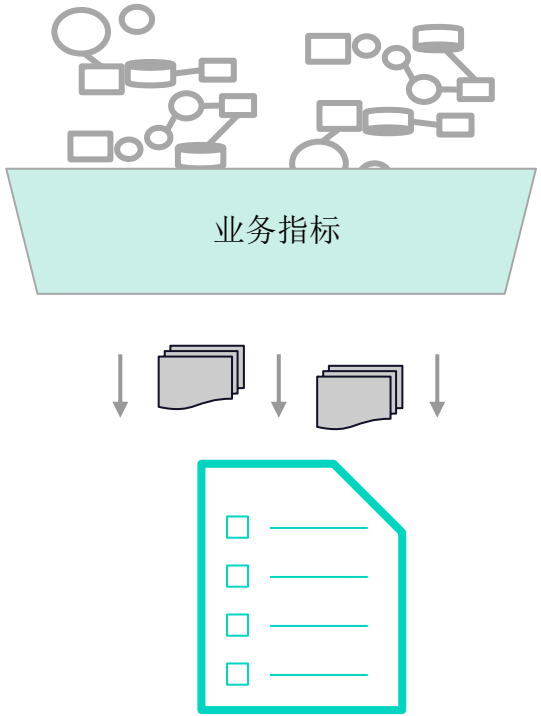


使用加权业务指标对应用评估和打分

	客户体验 (3)	业务变化 (3)	节省成本 (2)	降低财务风险 (2)	...	分数
App A	相关 (2)	非常相关 (3)	相关 (2)	非常相关 (3)	...	33
App B	非常相关 (3)	相关 (2)	非常相关(3)	相关 (2)	...	30
App C	无关 (1)	非常相关 (3)	无关 (1)	相关 (2)	...	26
App D	无关 (1)	相关 (2)	无关(1)	相关 (2)	...	26
App E	相关 (2)	相关 (2)	非常相关(3)	非常相关(3)	...	18

漏斗





怎么评估应用改造的技术难度？

SNAP分析时间太长

要分析的应用太多

缺乏宏观的技术视角

评估方法是否很好的扩展性

	Group	Characteristic	Large	Medium
19	Integration	**ESB, Gateway**	Home Grown, heavily customized COTS	
20	Integration	**File System Access**	File system as a core function - say for CMS	File system used
21	Integration	**Distributed Transactions**	Yes	
22	Integration	**Home Grown Frameworks/Libraries**	No source / Point of Contact. Polyglot source compiled into native libraries.	Source Available, SME's availal
23	Integration	**Legacy Communication Protocols**	Corba/RMI etc, Custom TCP protocols established over socket connections	TCP based to the app instance: TCP comunication between ap instances
24	Integration	**COTS Products**	Yes	
25	Integration	**Native Calls**	Yes. The native libs are 32 bit at uncompatible architecture with the stemcell	Native libs are 64 bit at compa architecture
26	Security	**JAAS, Spring Security, OAuth2, SAML, LDAP, SSO(Siteminder)**	Home Grown, JAAS	SAML
27	Others	**Batch Processing, ETL**	File System Access. Any one-off servers - singleton services , Loads entire data into memory	Database, Custom Batch processing frameworks
28	Core	**Build Tools**	Ant with migration, Home Grown. Is the source code organized in a few mono-repos	Ant/Ivy as is, Custom Maven, Gradle Plugins
29	CI	**Environments**	> 5	4-5
30	CI	**Test Coverage**		< 50%
31	Runtime	**Logging**	Heavy dependence on logging to file	Application writing to syslog
32	Runtime	**Startup time**	> 10 mins	5 - 10 mins
33	Runtime	**Configuration**	Credentials specified in an opaque fashion. Credentials checked into the source code. Custom credential management for backing resources	Snowflake - Each environment things a little different Highly customized Spring Prop Placeholder logic
34	Runtime	**JVM Memory settings**	> 10 G	
35	Runtime	**Multiple Ports**	> 1	
36	Runtime	**Telemetry**	No monitoring. No production view into the health of the app . No KPIs. No SLIs or SLOs	one-off custom dashboards. Porting of home grown monitc framework. Customized librari that need to be replaced with Spring cloud compatible ones.

Pivotal

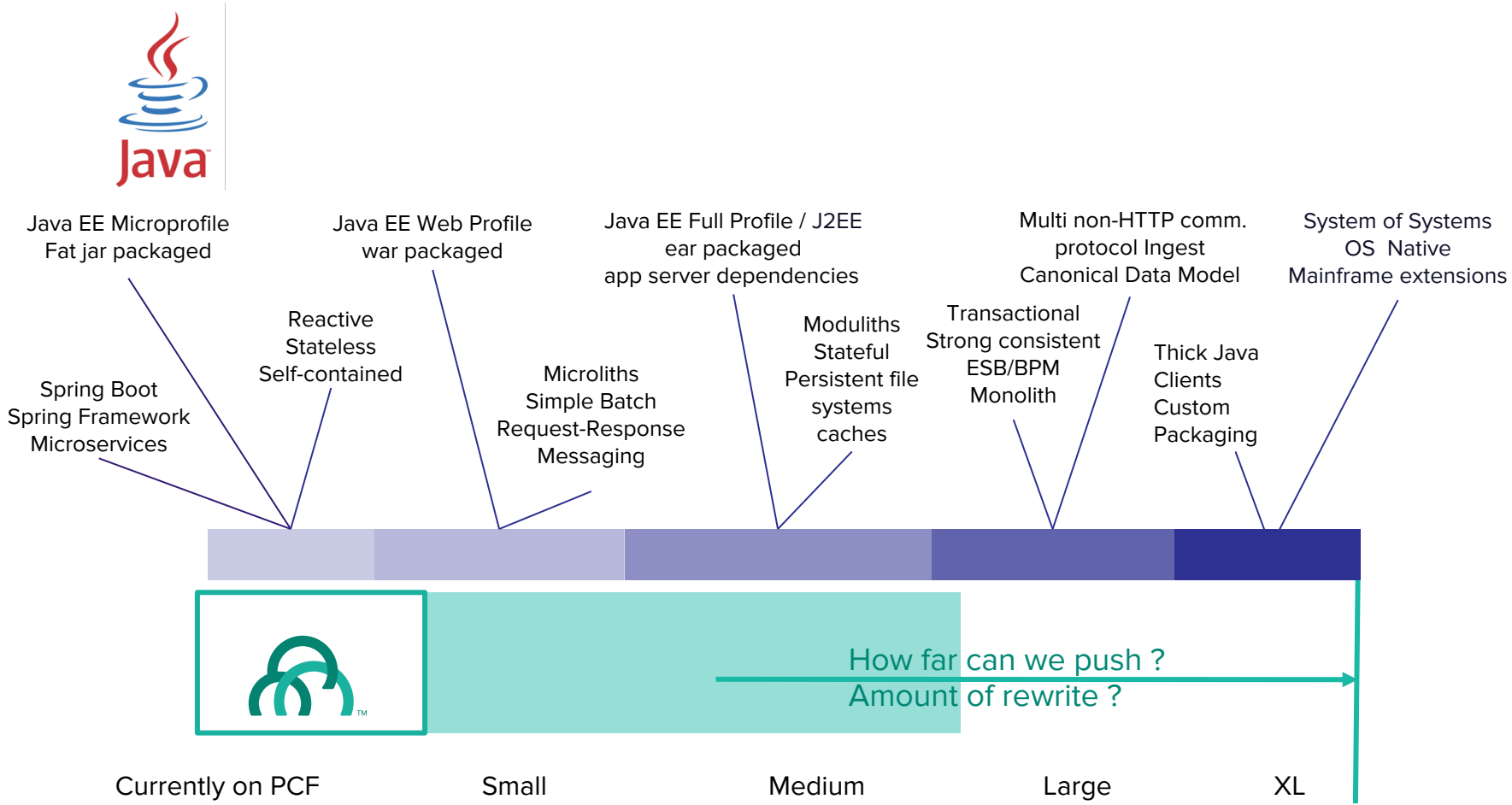
“我们需要一个快速的方法来为应用迁移的技术可行性打分”

“把应用分组”

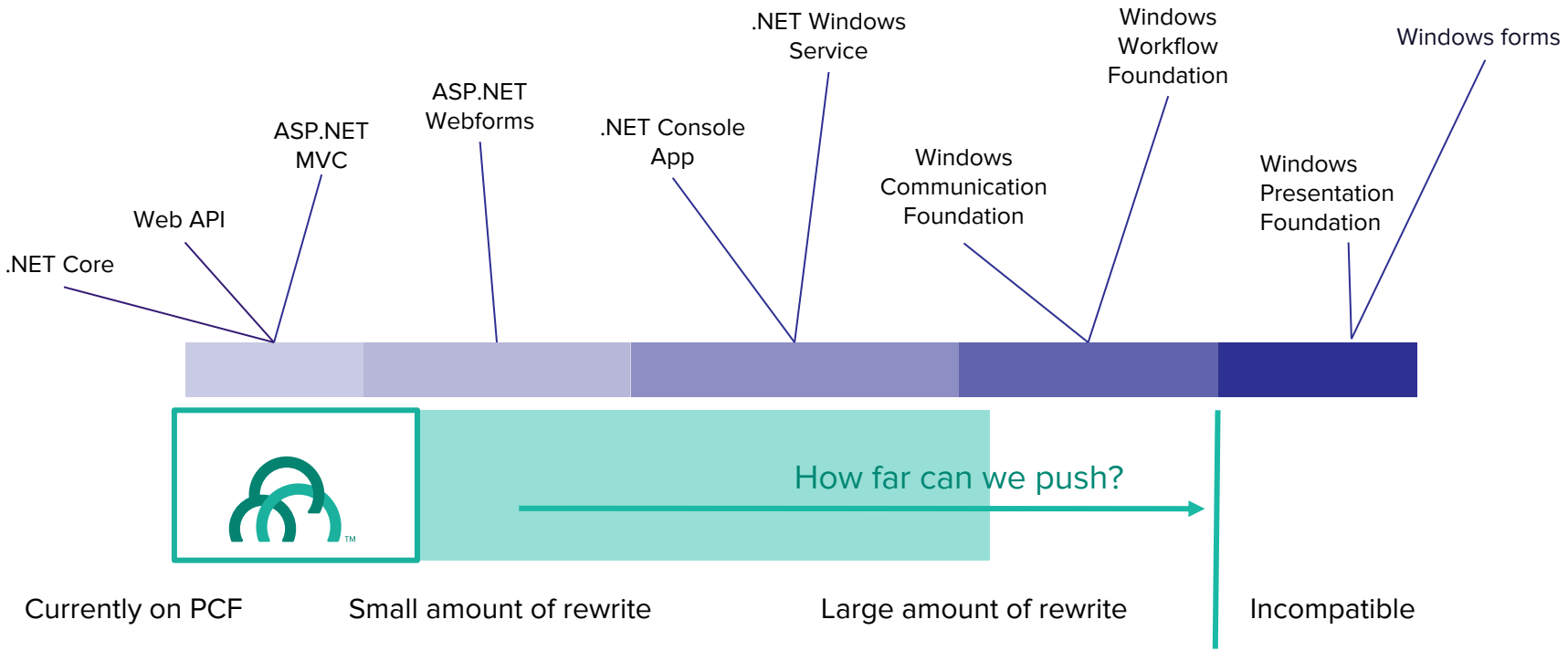
“为每个组打分，而不是为单个应用打分”

“是否可以推断出每个应用的分数”

“是否可以按应用的技术栈分类”


Java EE Microprofile
Fat jar packaged
Java EE Web Profile
war packaged
Java EE Full Profile / J2EE
ear packaged
app server dependencies
Multi non-HTTP comm.
protocol Ingest
Canonical Data Model
System of Systems
OS Native
Mainframe extensions
Reactive
Stateless
Self-contained
Moduliths
Stateful
Persistent file
systems
caches
Transactional
Strong consistent
ESB/BPM
Monolith
Thick Java
Clients
Custom
Packaging
Spring Boot
Spring Framework
Microservices
Microliths
Simple Batch
Request-Response
Messaging
How far can we push ?
Amount of rewrite ?
Currently on PCF
Small
Medium
Large
XL

Microsoft® .NET

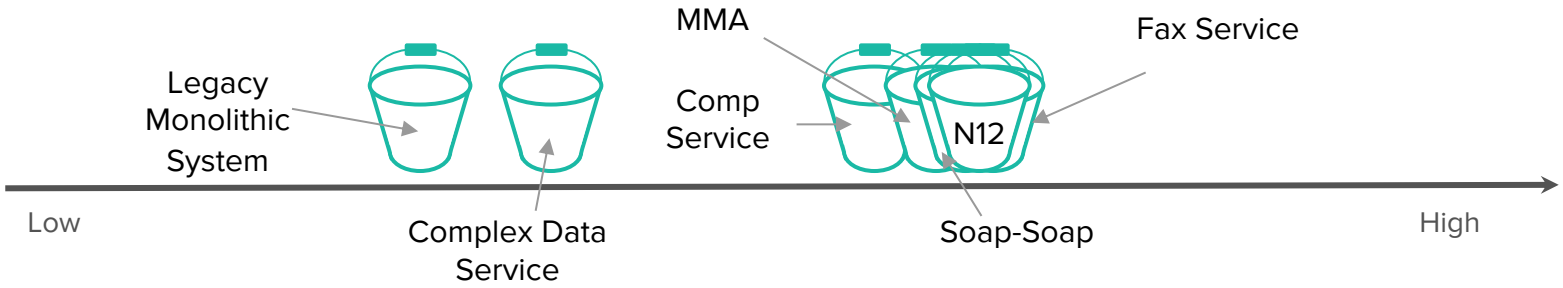


Pivotal

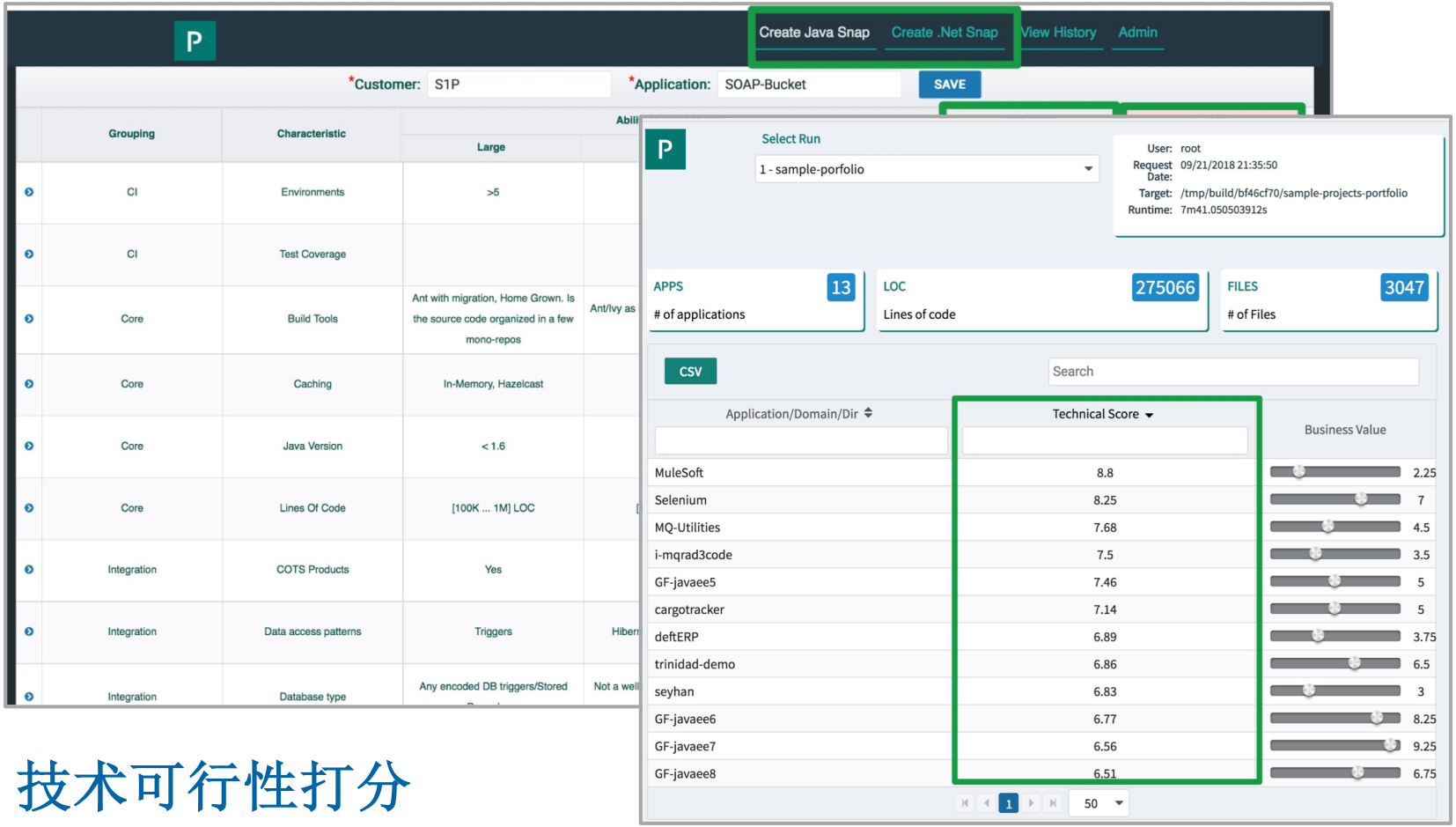
根据应用的技术架构相似性来确定应用的分类 -- “篮子”

根据需要识别出新的篮子

用SNAP对每个篮子进行评估



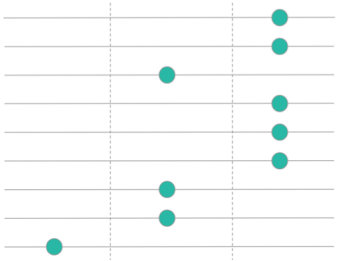
技术可行性

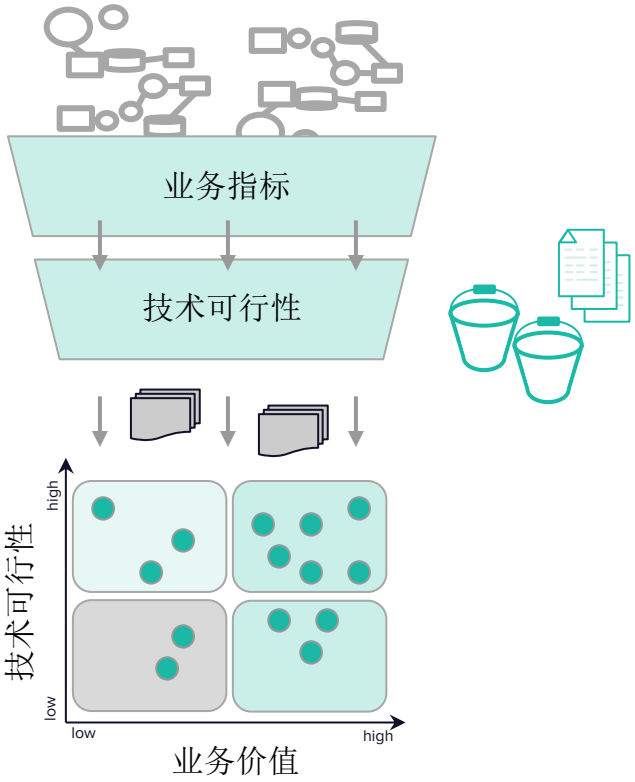


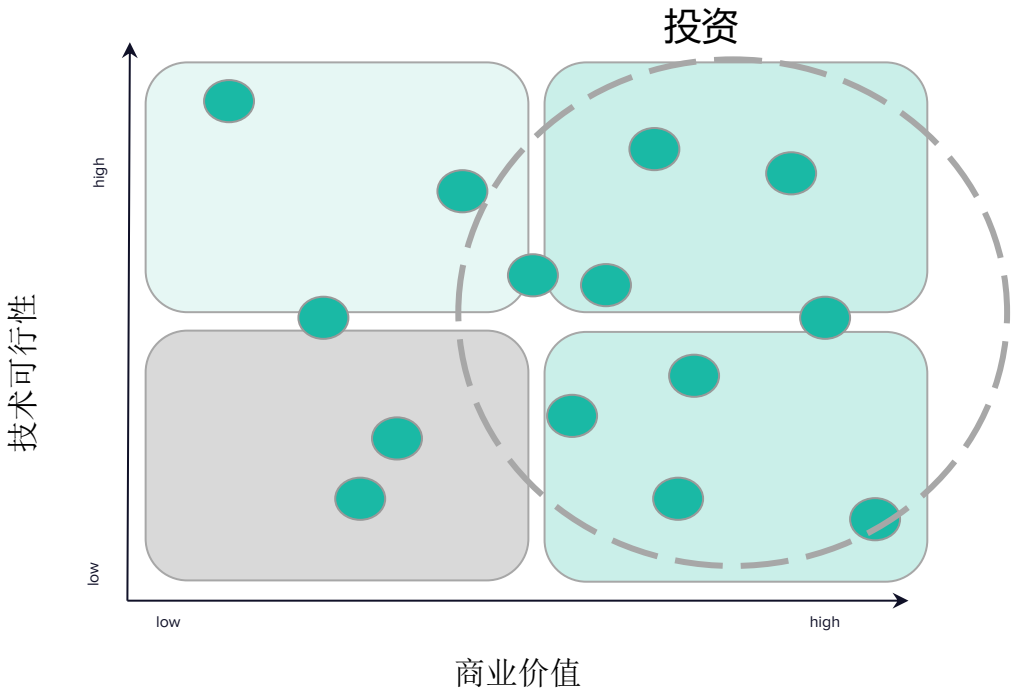
技术可行性打分

Pivotal

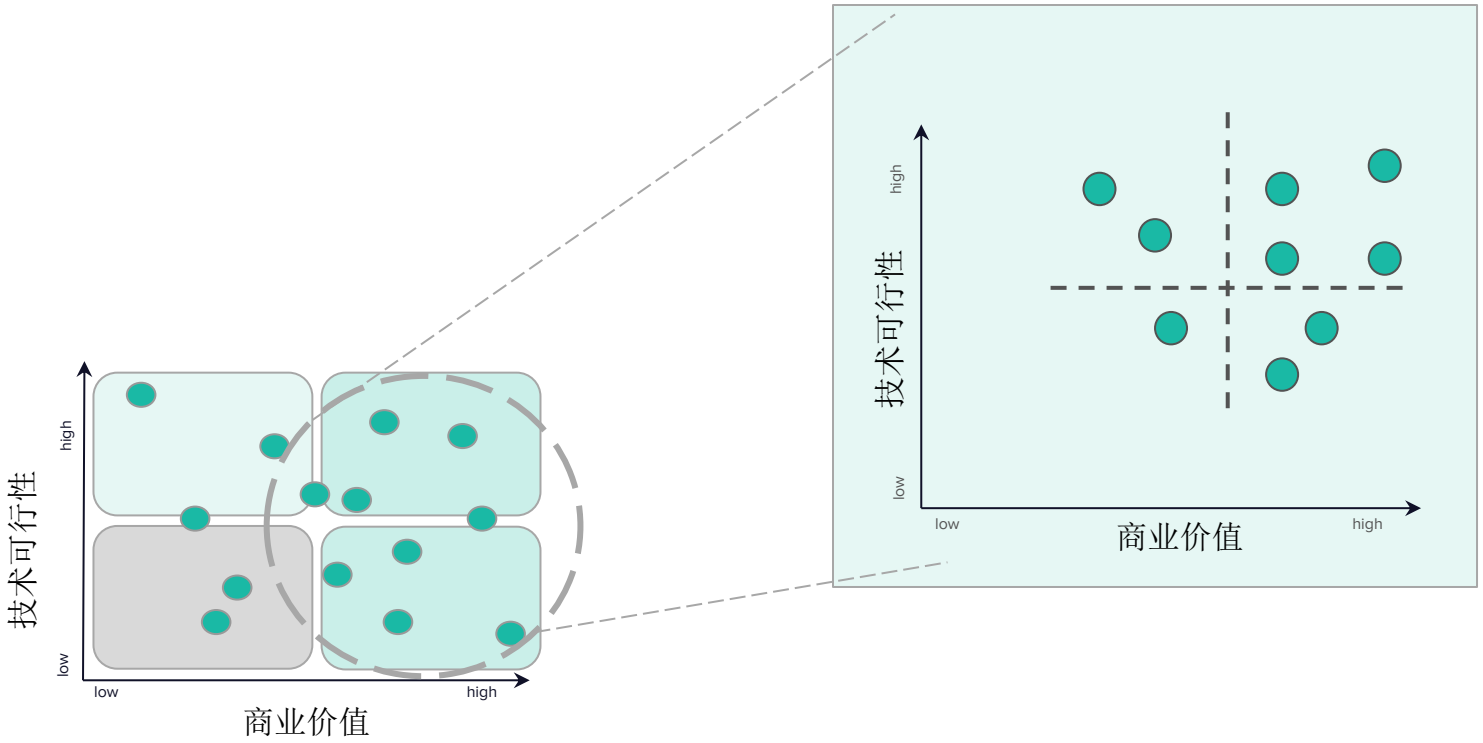
漏斗





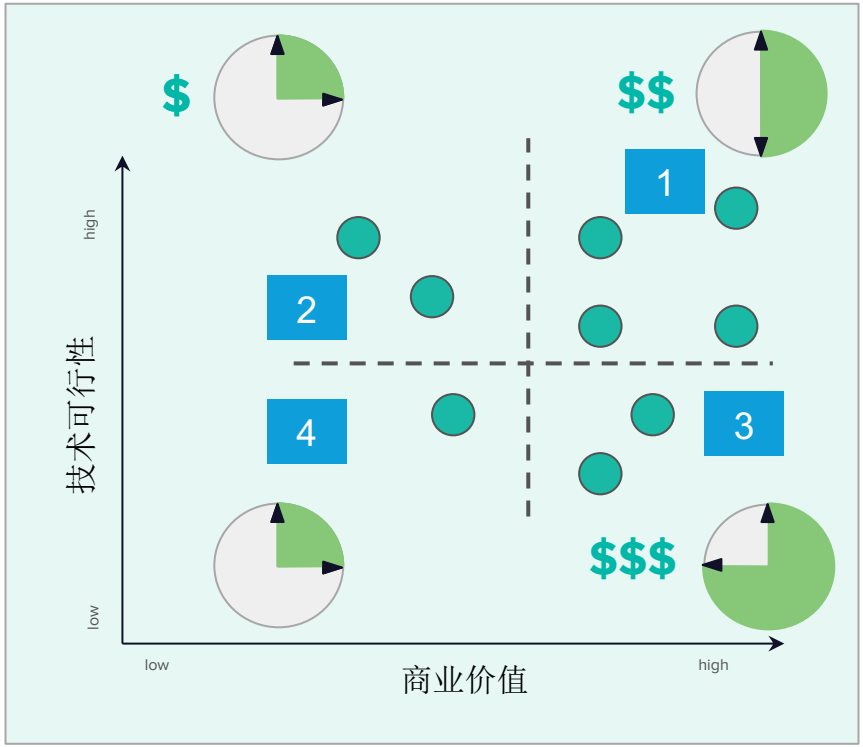

投资
技术可行性
high
low
low
high
商业价值

优先级矩阵



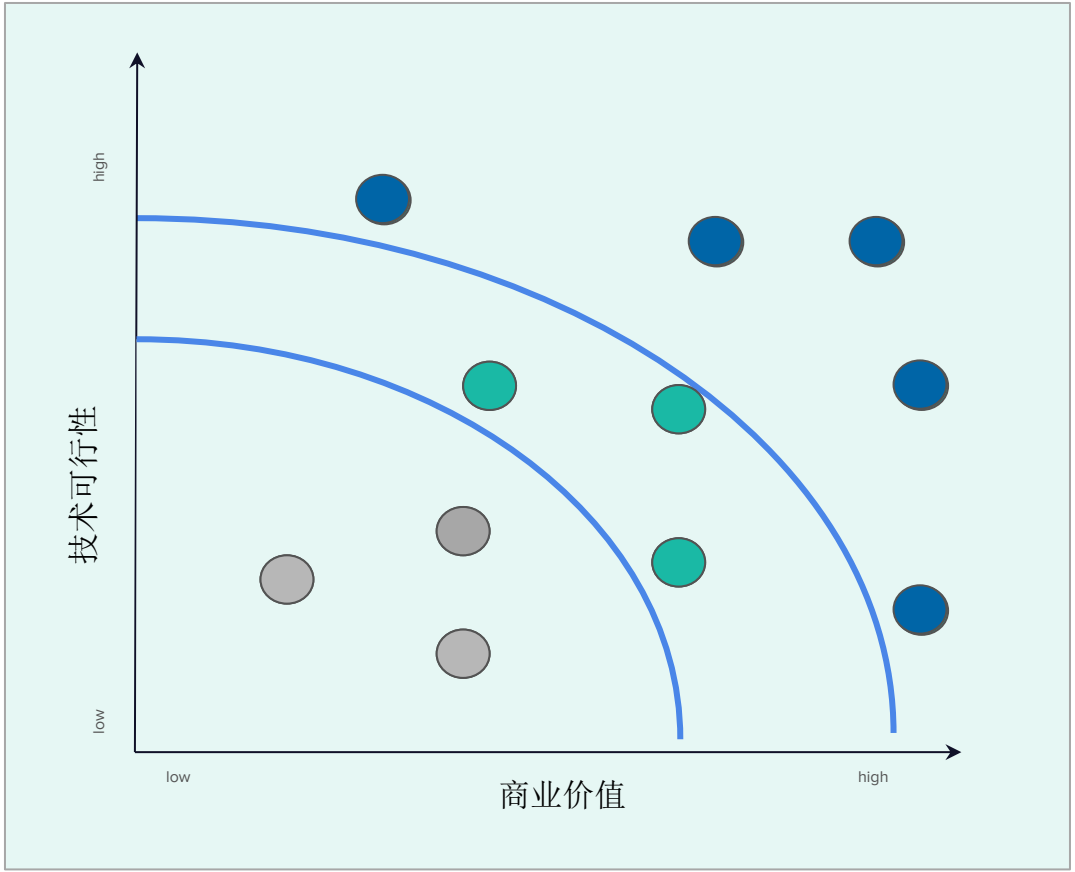
Pivotal

应用改造
优先级矩阵



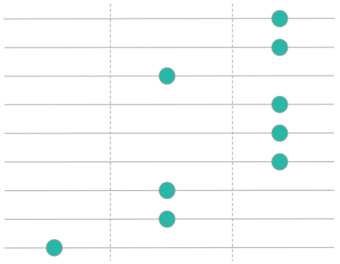
Pivotal

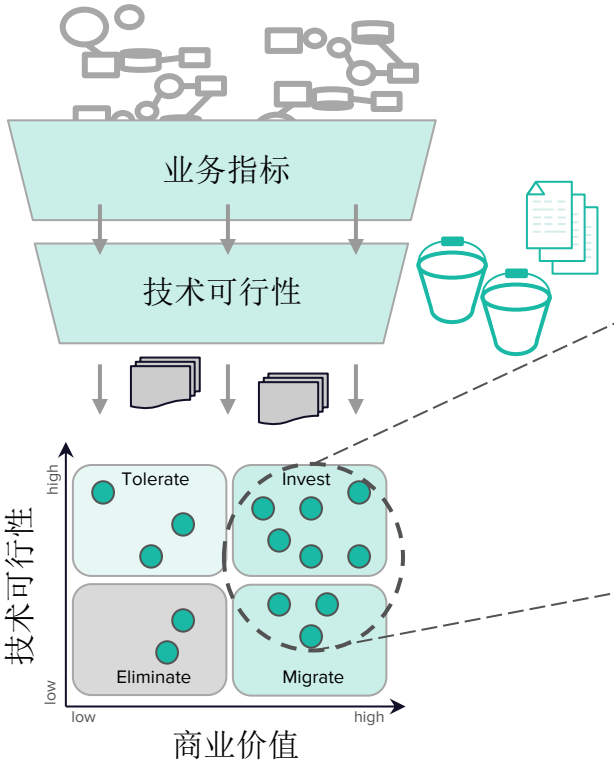
应用改造
优先级矩阵



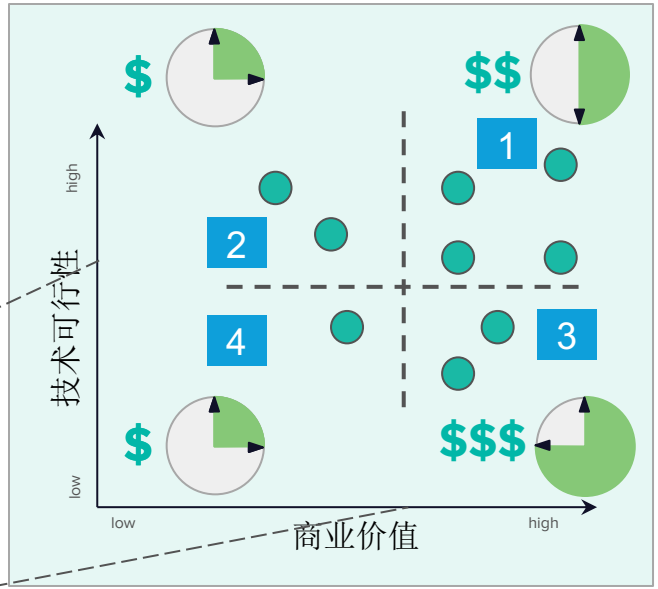
Pivotal

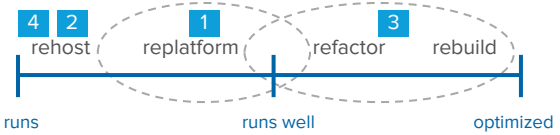
漏斗





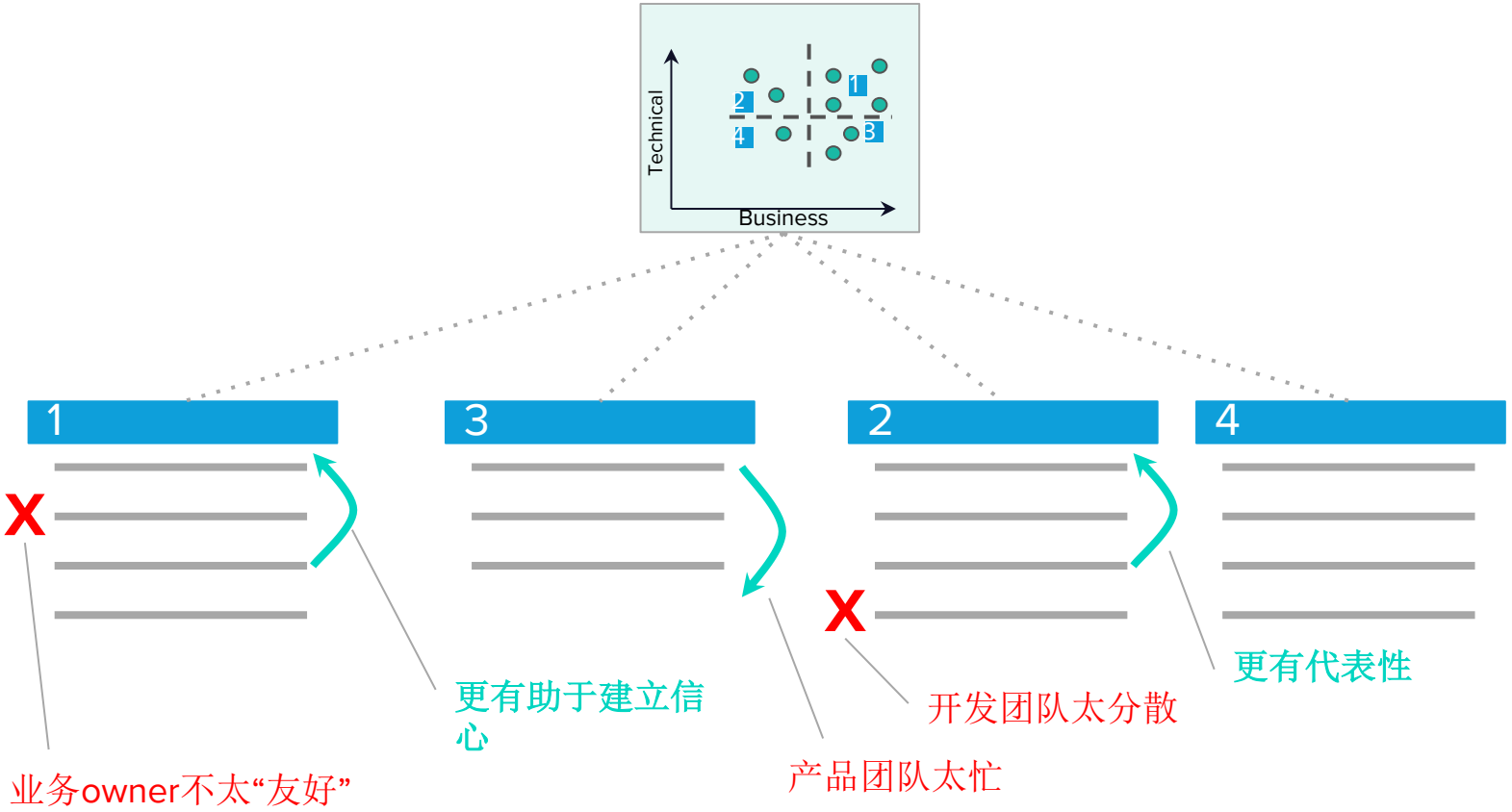
优先级矩阵





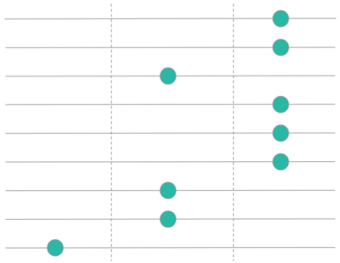
改造策略

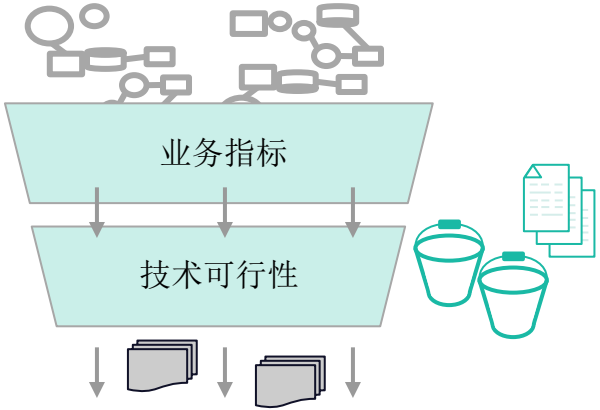
Pivotal

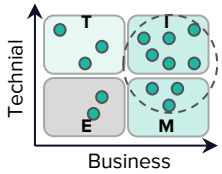

Technical
Business
1
2
3
4
1
3
2
4
X
X
业务owner不太“友好”
更有助于建立信心
产品团队太忙
开发团队太分散
更有代表性

Pivotal

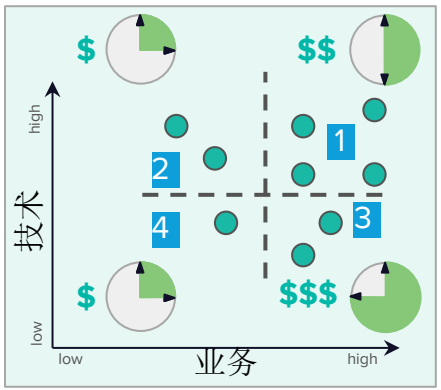
漏斗

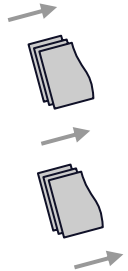


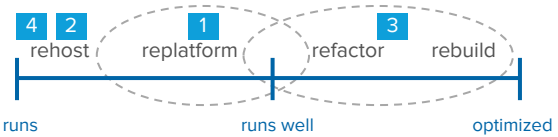


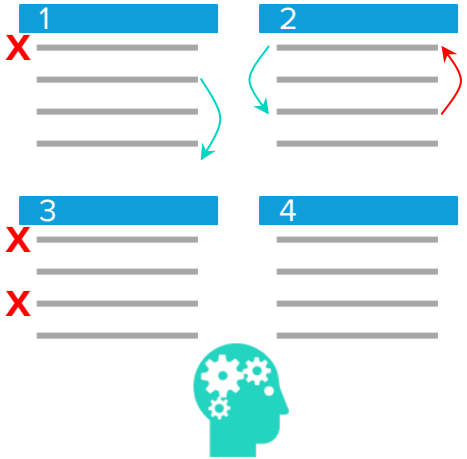


优先级矩阵





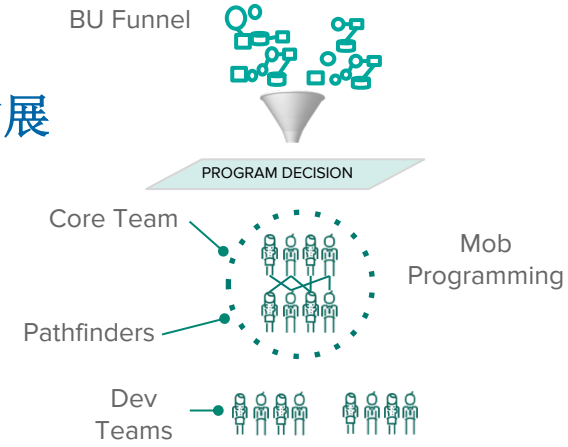


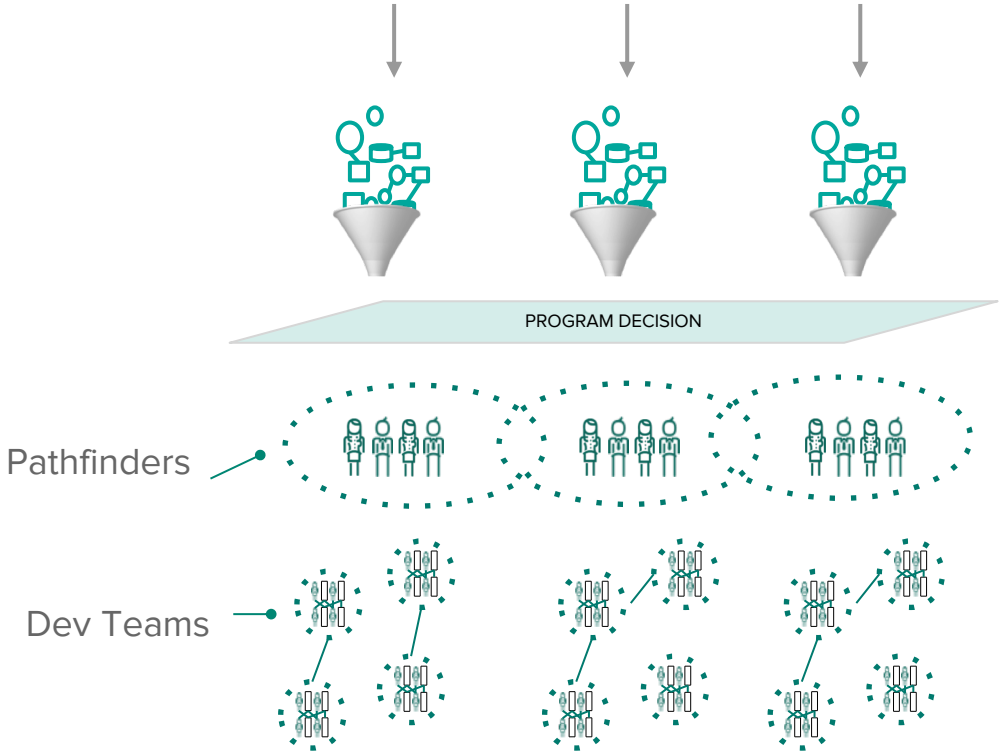


“Think”
优化

Pivotal

扩展


BU Funnel
PROGRAM DECISION
Core Team
Mob
Programming
Pathfinders
Dev
Teams


PROGRAM DECISION
Pathfinders
Dev Teams

AppTx Journey Markers

怎么度量？

生产环境上的应用数量

发布频率

业务改变的反应时间

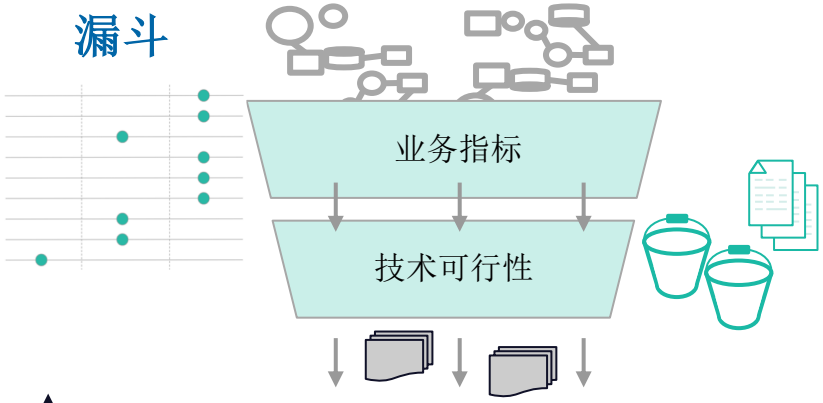
专门的转型顾问团队

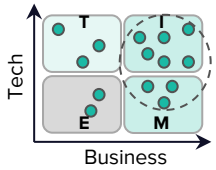
开发者的参与度

云原生改造技术积累

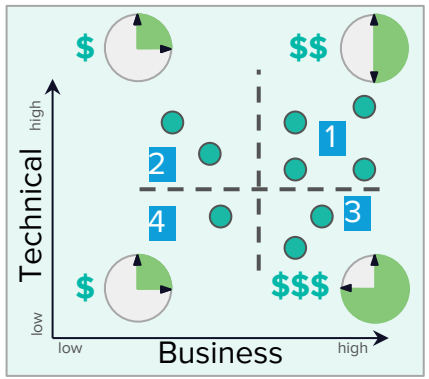
云原生培训

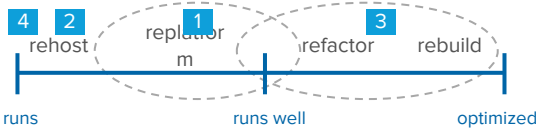
漏斗

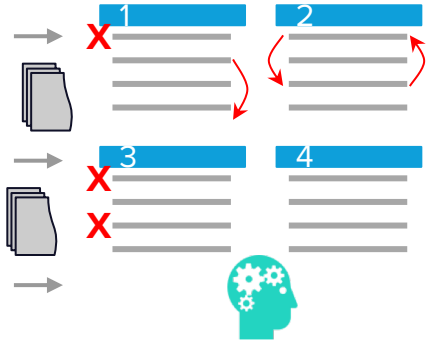




优先级矩阵

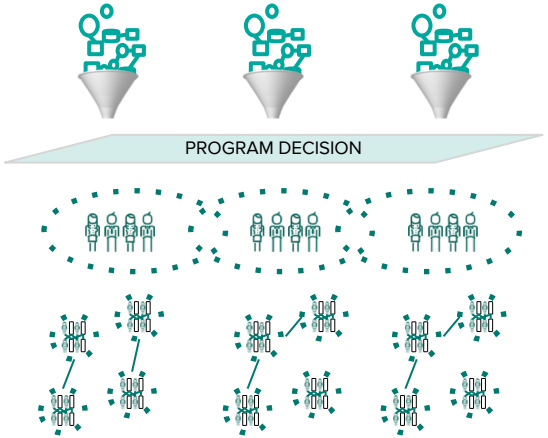


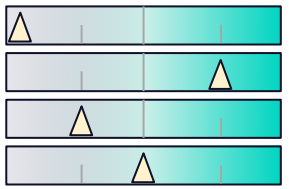




“Think”
优化

扩展

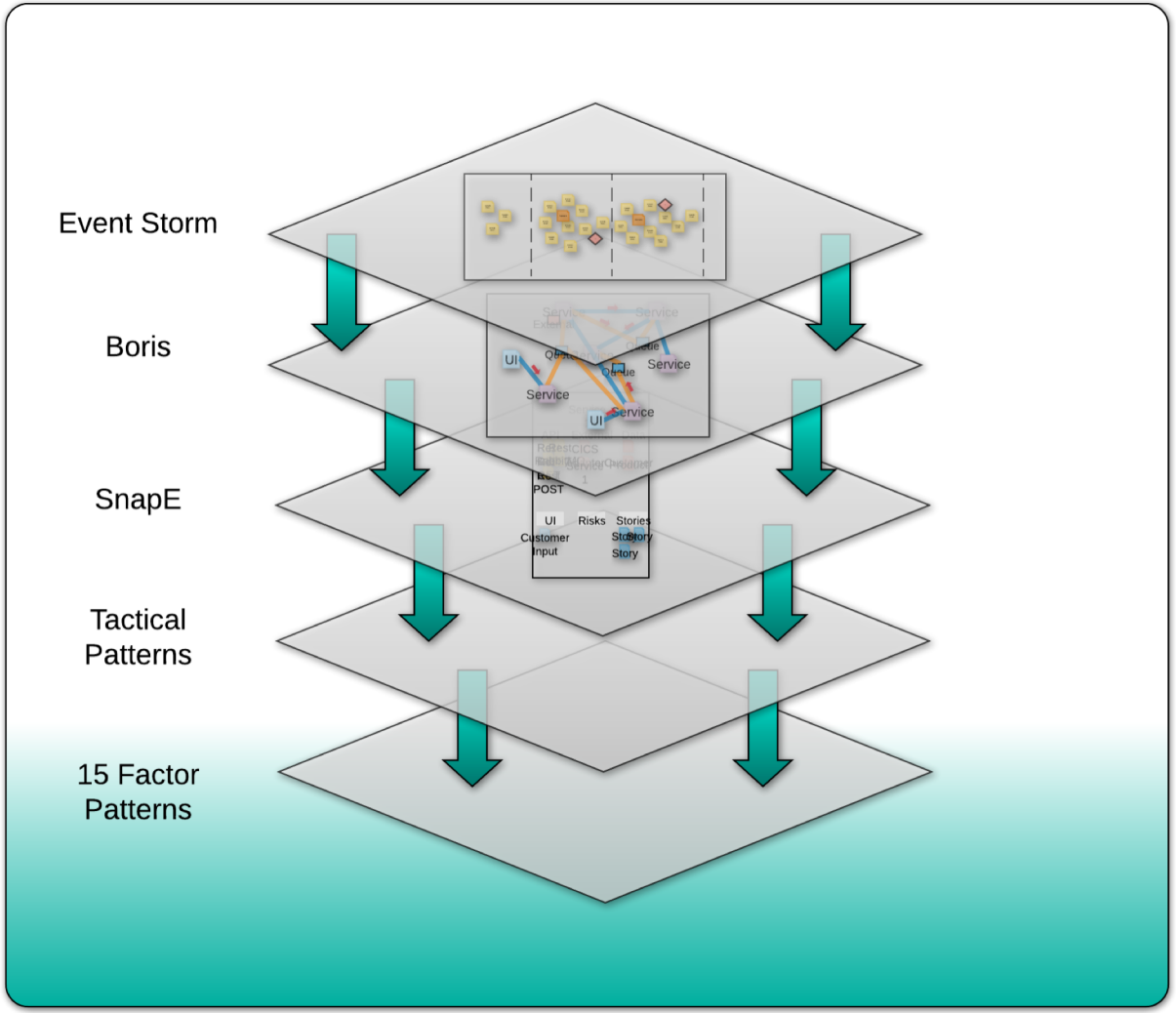




改造之路的
度量

Pivotal

如何快速拆解一个单体应用为微服务


Event Storm
Boris
SnapE
Tactical Patterns
15 Factor Patterns
UI
Service
Queue
Service
UI
Service
POST
UI
Risks
Stories
Customer Input
Story
Story

Pivotal

事件风暴

目的

- 快速发掘和梳理业务流程
- 形成业务边界、初步的微服务划分
- 形成统一的业务语言

参与者

- 熟悉领域业务的人员
- 架构师等技术人员

形式

- Workshop
- 白纸、便利贴、笔

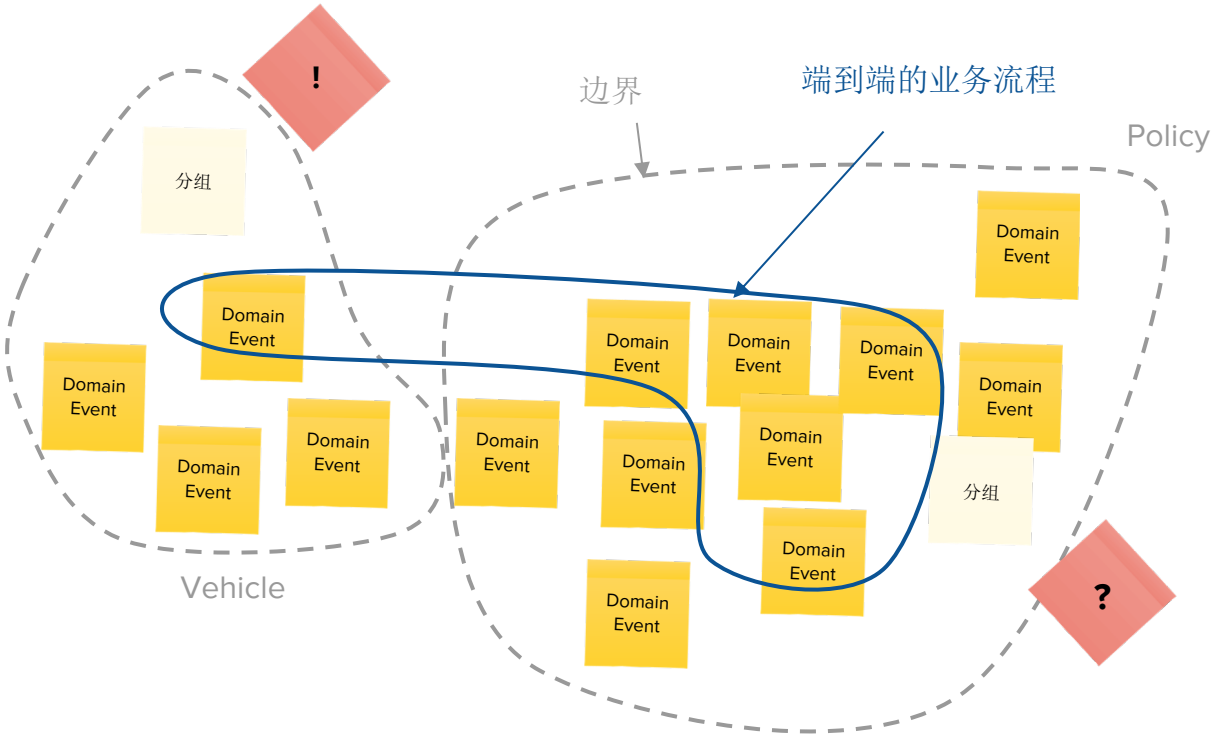
事件风暴

核心概念

- 领域事件 - 系统中发生的业务事件

步骤

1. 头脑风暴领域事件
2. 按先后顺序排列事件
3. 按业务相关性分组
4. 选择一个端到端的业务流程作为开始（在系统较大较复杂时）



Boris图

目的

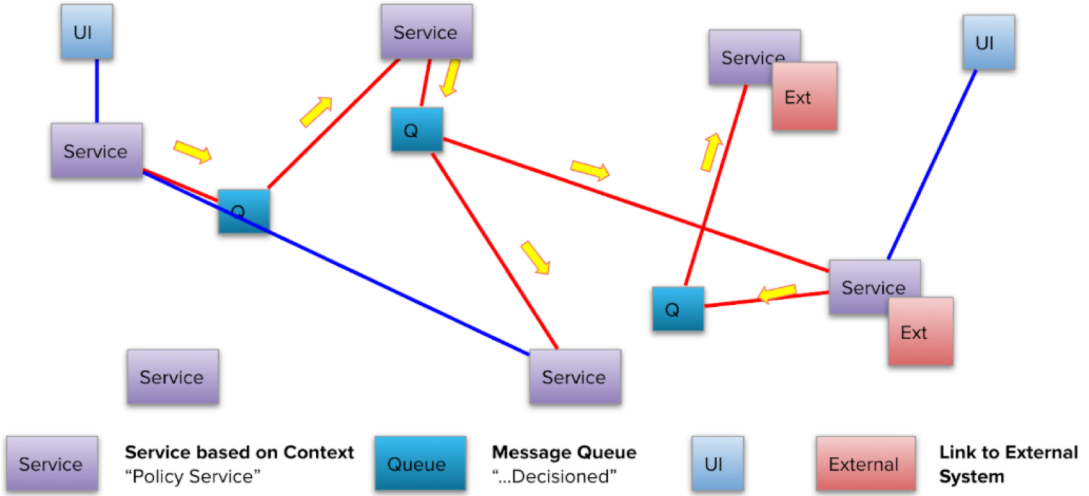
- 由业务过渡到技术
- 形成系统微服务架构雏形
- 为定义微服务做准备

参与者

- 架构师等技术人员为主
- 熟悉领域业务的人员

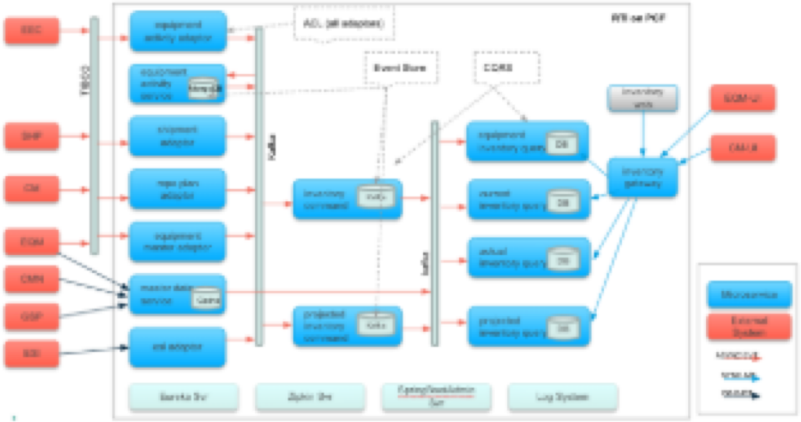
形式

- Workshop
- 白纸、便利贴、笔



步骤

1. 选择一个端到端的业务流程
2. 串联起相关服务和外围系统
3. 确定连接是同步还是异步
4. 填上连接涉及到的数据
5. UI、Queue、Risk等



Pivotal

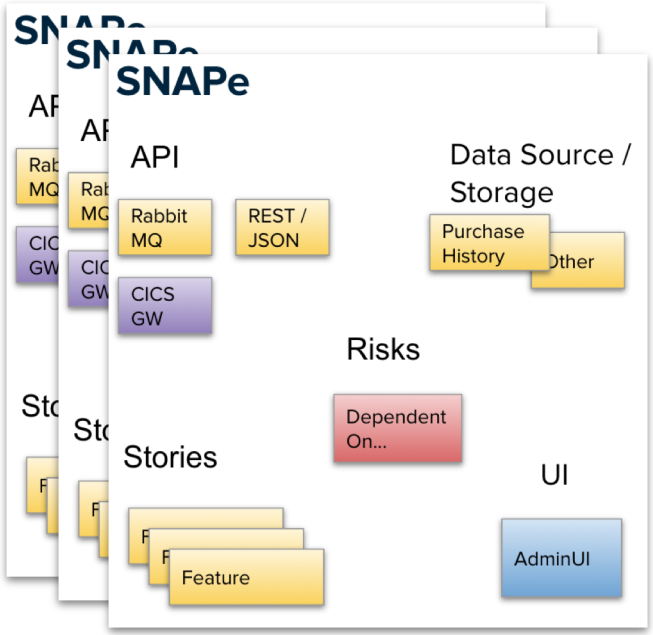
SNAP-E

目的

- 极简微服定义文档

内容/步骤

- API - 服务API定义
  - 来自Boris中的连接线
  - 同步/异步 (Pub-Sub)
- Data
  - 来自连接中收发的数据
- Stories - 服务支持的业务功能
- UI - 服务涉及到的UI简单描述
- Risks - 服务实现的技术/业务风险等



APP ARCHITECTURE

API First Development

API Gateway Pattern

Batch Microservices with Spring Batch

Batch Scheduler

Setting up Spring Config Server

OBSERVABILITY

SECURITY

TESTING

REPLATFORMING

CONFIGURATION

Application Transformation Recipes

Application Transformation (or "AppTx") encompasses a set of strategies for migration of existing software into the cloud. Pivotal has helped hundreds of customers through AppTx initiatives and has captured this guidance here. This cookbook includes a growing set of proven patterns and techniques that will help accelerate your journey to the cloud.

LATEST RECIPES

API First Development

API Gateway Pattern

Batch Microservices with Spring Batch

Batch Scheduler

Setting up Spring Config Server

相关链接

- https://pivotal.io/application-transformation
- https://pivotal.io/application-transformation-recipes


Pivotal®
Transforming How The World Builds Software