Cloud-Native Day
2020 Korea
LIVE!

마이크로서비스 전환에 필요한 2가지 측면의 접근법

Jung In Lee

vmware

Cloud-Native Day
2020 Korea
LIVE!

Today's Sessions

1. 마이크로서비스 디자인
2. 마이크로서비스 구축
3. 마이크로서비스 런타임 & 네트워크
4. 마이크로서비스 모니터링 및 관리

vmware

2가지 측면의 고려 사항

"How do we move toward

modern apps

with greater velocity?"

"How do we operate

modern infrastructure

with more stability at scale?"

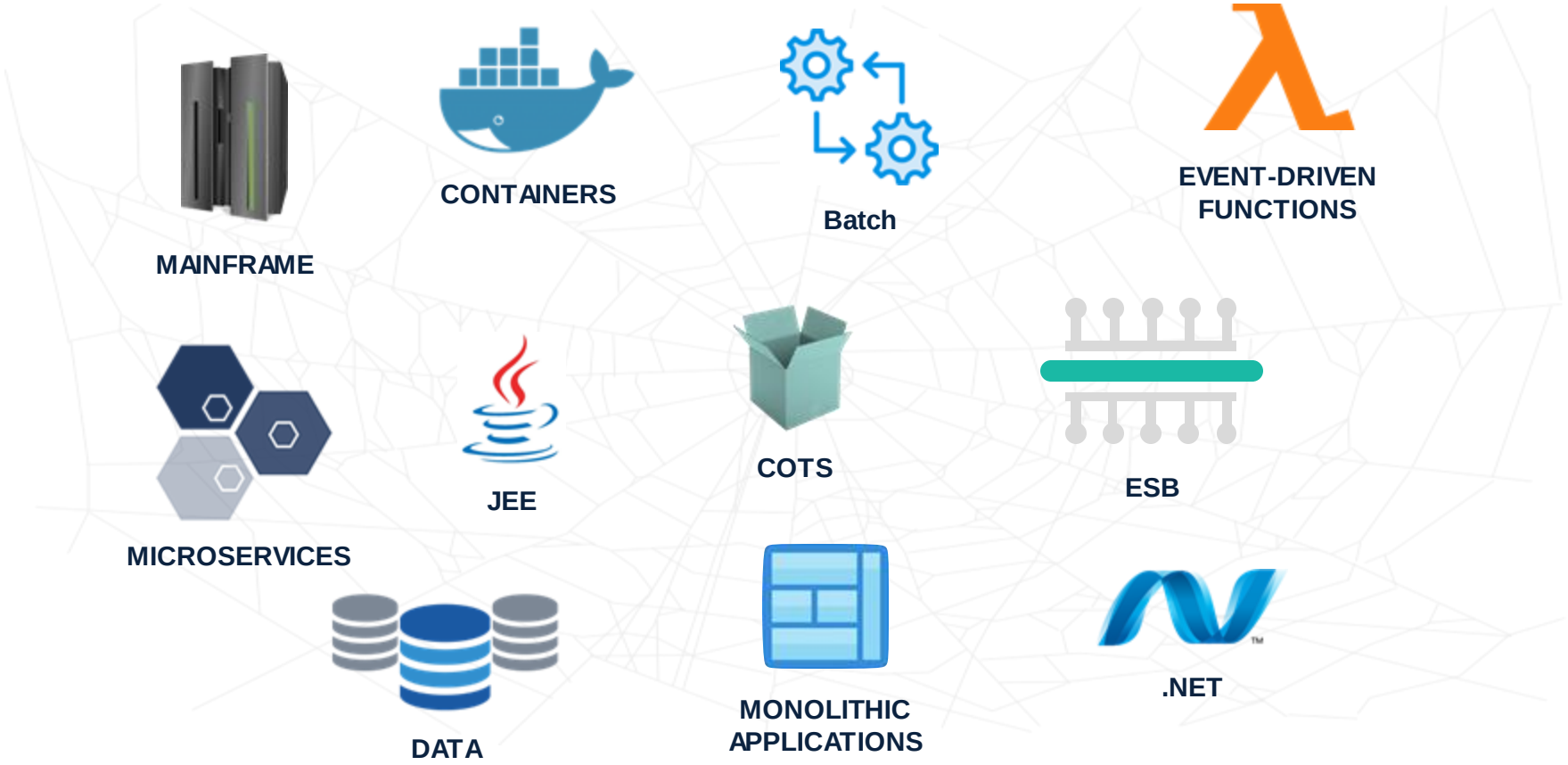
Application 측면의 접근법

7 Key Questions

1. 모든 Application이 마이크로 서비스에 적합한 것인가?
2. 우리가 얻으려고 하는 마이크로 서비스의 Value는 무엇인가?
3. 어떠한 Criteria에 가장 우선순위를 두고 아키텍처를 설계할 것인가?
4. 전환의 방법은 Big Bang 인가? 혹은 점진적 소멸(Strangler) 방식인가?
5. 어떤 Implementation 패턴을 이용하여 마이크로 서비스를 구현할 것인가?
6. 마이크로 서비스 운영에 필요한 툴에 따라 코딩 패턴의 Dependency는 생기는가?
7. 성공적인 MSA 경험을 다른 개발 조직으로 어떻게 전파할 수 있을까?

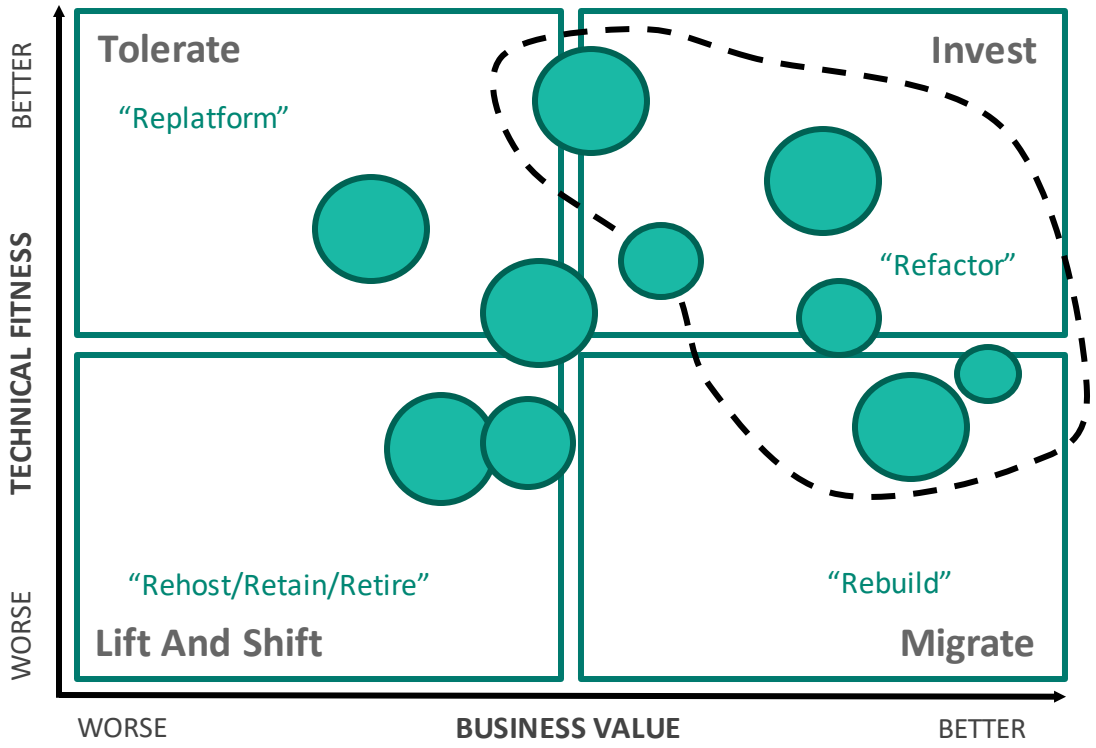
Application 측면의 접근법

1. 모든 Application이 마이크로 서비스에 적합한 것인가?



Application 측면의 접근법

1. 모든 Application이 마이크로 서비스에 적합한 것인가?



Upload binary or map to source control

Parsers for most common languages (Java, .NET, PHP, Ruby, etc.)

Extensible rules (100s available currently)

Understand technical effort to move your app

UI dashboard to help triage decisioning

Application 측면의 접근법

2. 우리가 얻으려고 하는 마이크로 서비스의 Value는 무엇인가?

개발	배포	운영
☐ 서비스 고유의 Best-Fit 기술 활용	☐ 독립적인 서비스 배포 사이클 확보	☐ 서비스의 독립적인 확장성
☐ 모듈 재사용성 극대화	☐ CI/CD 배포 자동화	☐ 서비스 장애 고립
☐ 모던 프레임워크 전환	☐ Immutable 배포 통한 에러 감소	☐ MTTR 감소
☐ Small Team 구성과 애자일 문화	☐ A/B 테스트	☐ Failure Rate 감소
⋮	⋮	⋮

Application 측면의 접근법

3. 어떠한 Criteria에 가장 우선순위를 두고 아키텍처를 설계할 것인가?

- Organization Ownership : 서비스의 담당 현업팀이 분리 되어있는가?
- Release Cycle : 서비스의 업데이트 주기를 독립적으로 가져가야 하는가?
- Scaling : 서비스 마다 별도의 확장성을 확보해야 하는가?
- Data and Transaction : 데이터가 분리되어야 하는가?
- SPOF : 하나의 서비스에 너무 많은 의존성이 존재하지 않는가?
- Maintenance Effort : 소수의 개발/운영 조직이 운영을 감당할 수 있는가?

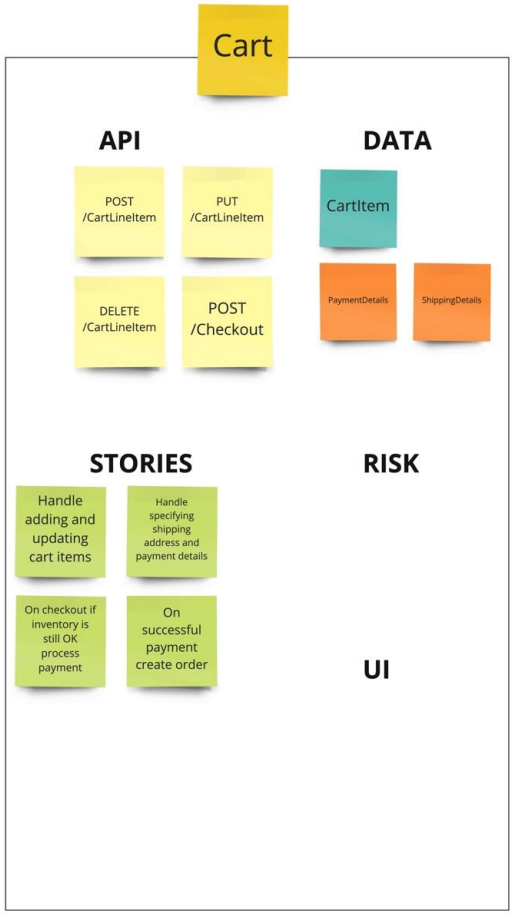
Application 측면의 접근법

3. 어떠한 Criteria에 가장 우선순위를 두고 아키텍처를 설계할 것인가?

Domain Driven Design

Boris Diagram

SNAP-E



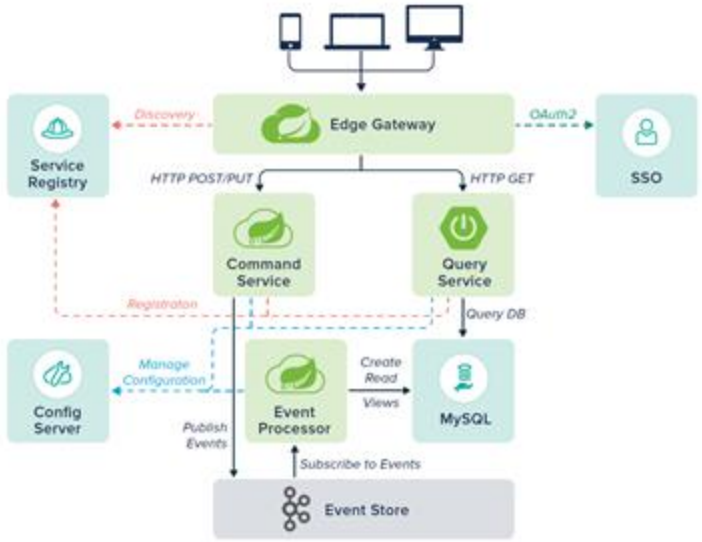
Application 측면의 접근법

4. 전환의 방법은 Big Bang 인가? 혹은 점진적 소멸(Strangler) 방식인가?

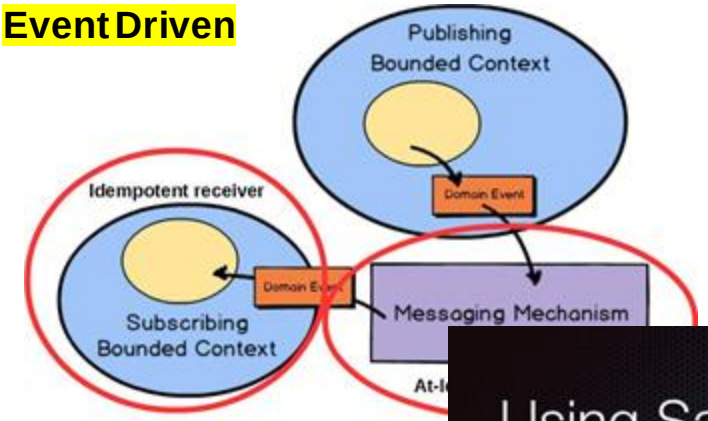
Application 측면의 접근법

5. 어떤 Implementation 패턴을 이용하여 마이크로 서비스를 구현할 것인가?

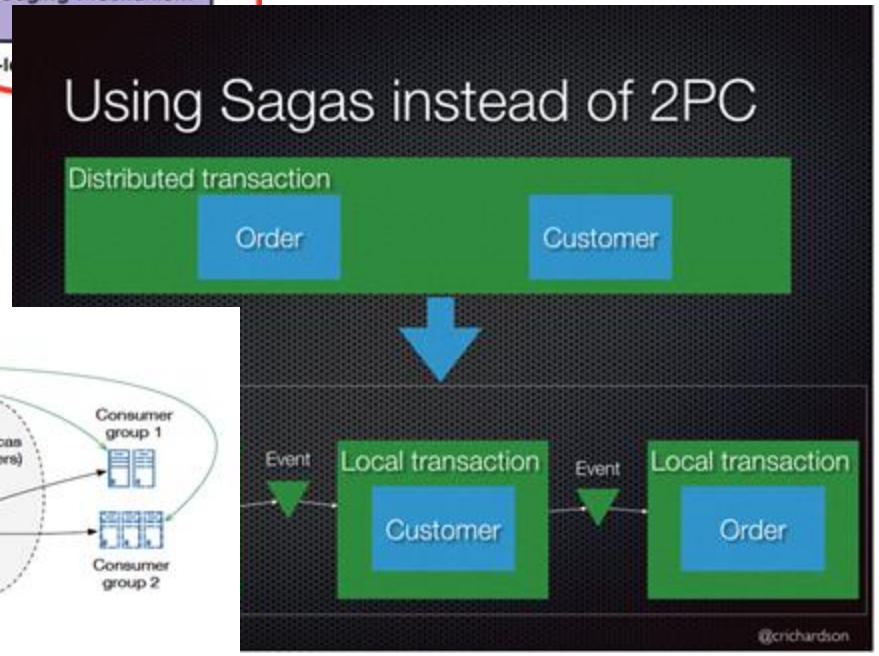
CQRS



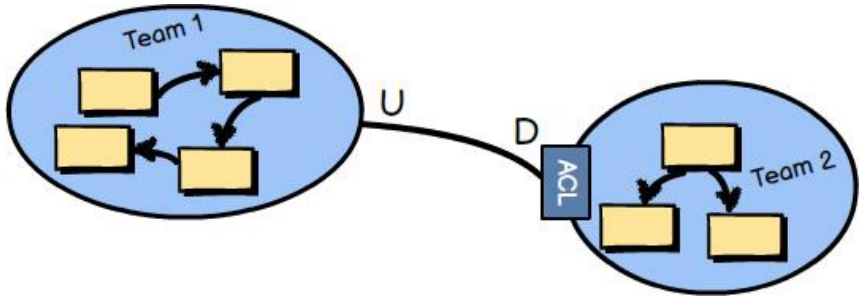
Event Driven



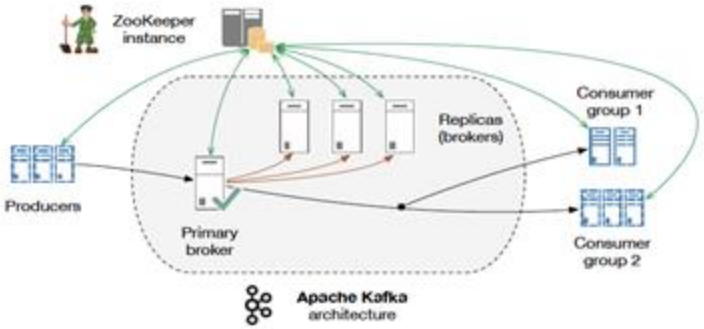
Saga for distributed txns



Anti Corruption Layer



Messaging Queue



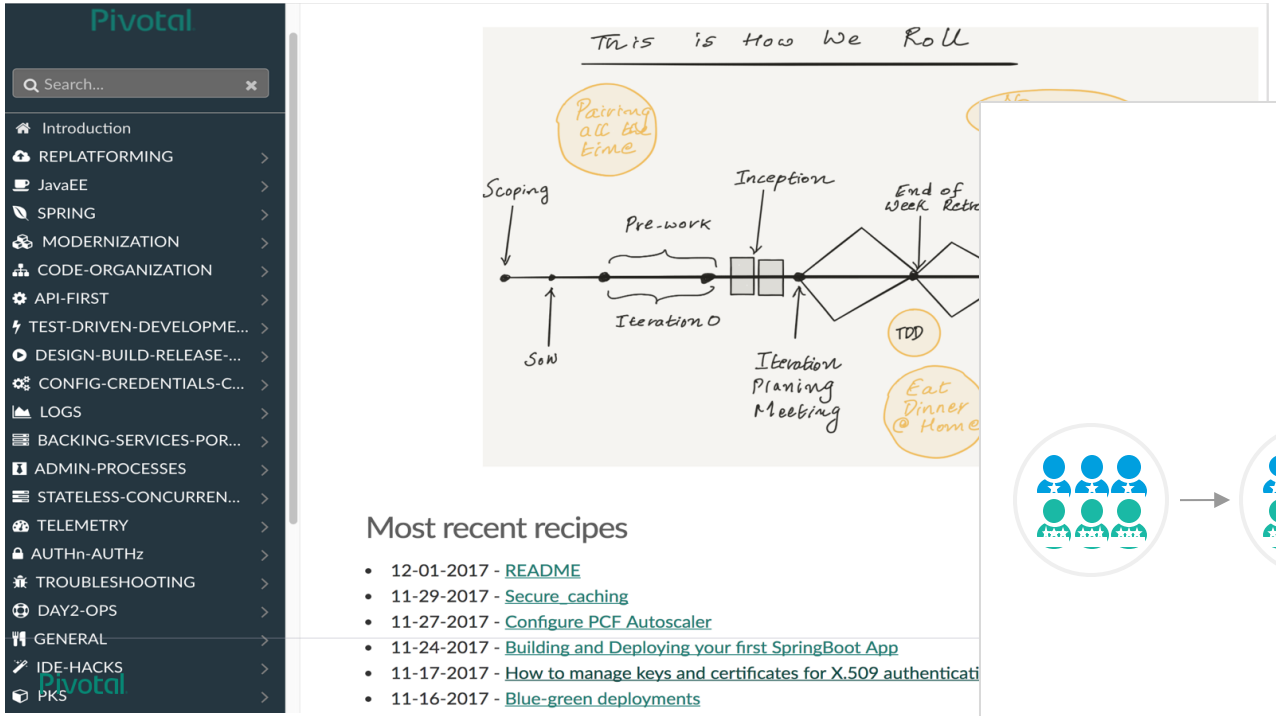
Application 측면의 접근법

6. 마이크로 서비스 운영에 필요한 툴에 따라 코딩 패턴의 Dependency는 생기는가?

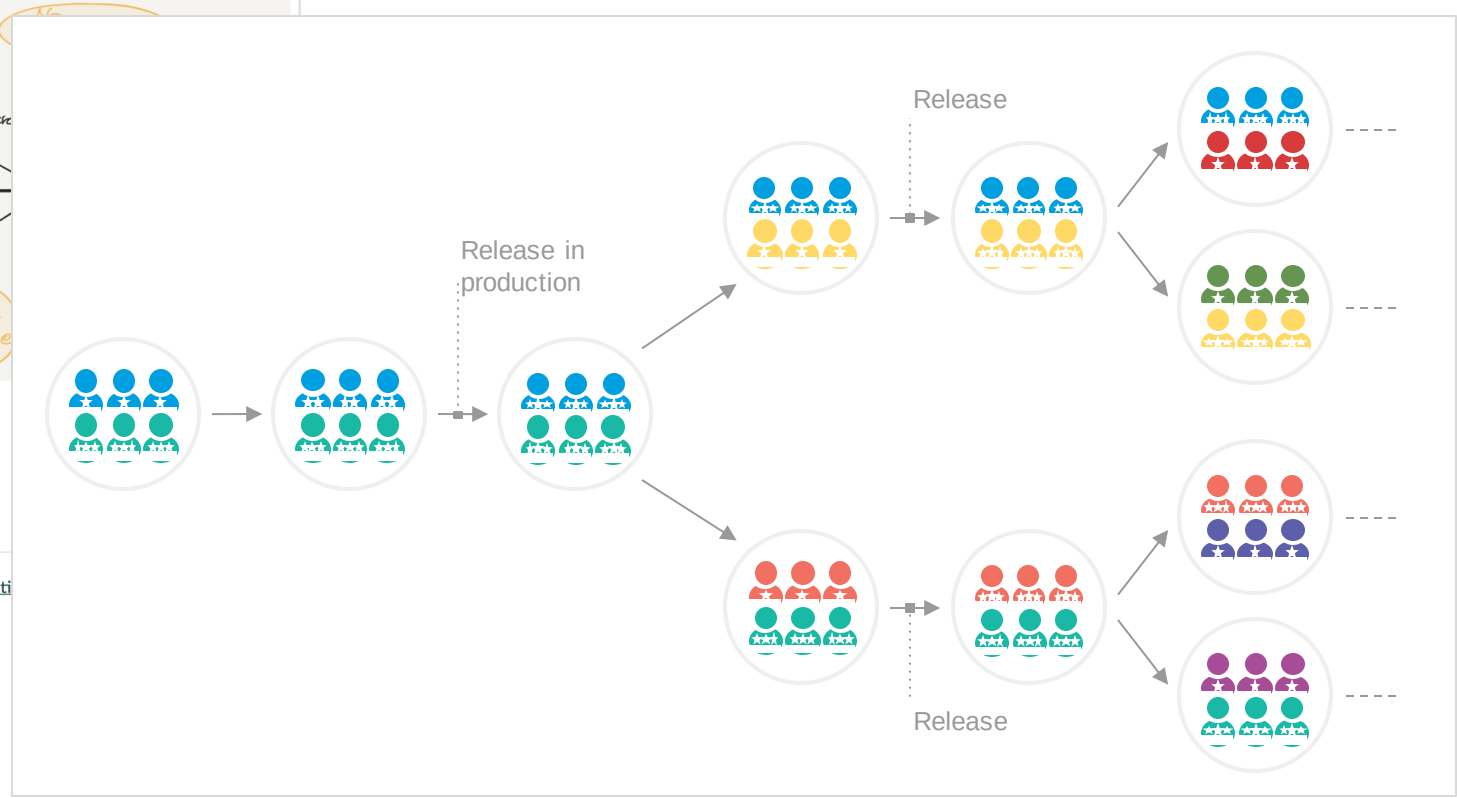
Application 측면의 접근법

7. 성공적인 MSA 경험을 다른 개발 조직으로 어떻게 전파할 수 있을까?

Cookbook and Recipes



Pollination Mode Scaling



Application 측면의 접근법

7 Key Questions

1. 모든 Application이 마이크로 서비스에 적합한 것인가?
2. 우리가 얻으려고 하는 마이크로 서비스의 Value는 무엇인가?
3. 어떠한 Value를 가장 우선순위를 두어, 아키텍처를 설계할 것인가?
4. 전환의 방법은 Big Bang 인가? 혹은 점진적 소멸(Strangler) 방식인가?
5. 어떤 Implementation 패턴을 이용하여 마이크로 서비스를 구현할 것인가?
6. 마이크로 서비스 운영에 필요한 툴에 따라 코딩 패턴의 Dependency는 생기는가?
7. 성공적인 MSA 경험을 다른 개발 조직으로 어떻게 전파할 수 있을까?

1. VMware **Journey Navigator** Program

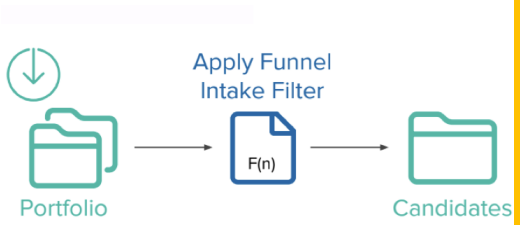
2. VMware **App Navigator** Program

3. VMware **Modernization** Program

App Tx 프로세스

Application Transformation Process

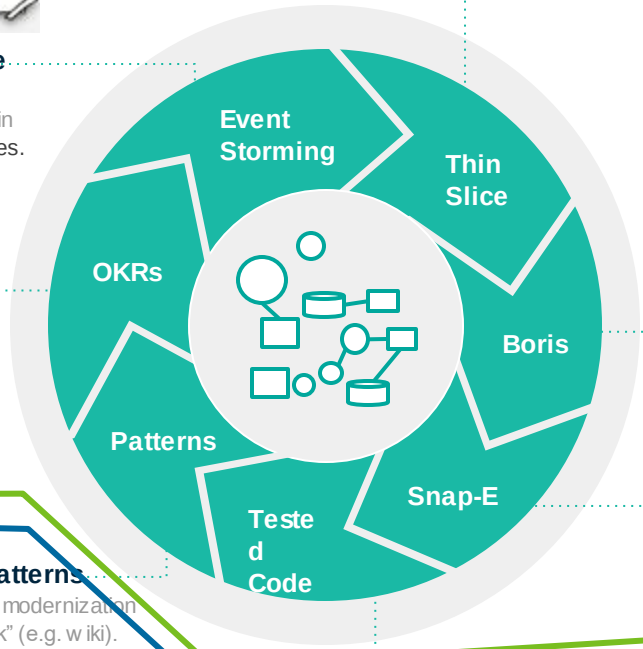
1. VMware **Journey Navigator** Program



2. VMware **App Navigator** Program

2. Event Storm the legacy space
Highly collaborative modeling tornado!
Make sense of a complex business domain with a common language for different tribes.
Identify trouble spots & starting points.

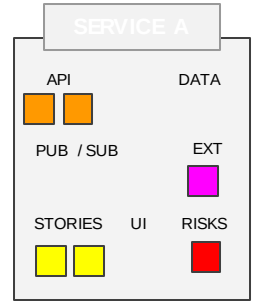
1. Define Objectives & Key Results
Decide on direction for outcomes & approach.
Agree on objectives and refine key results to measure those.



3. Select "Thin Slice(s)"
Cut the monolithic business cake and pick a sweet piece to start the modernization work.

4. "Boris" crafts the desired notional architecture
Model the API
ıe business capabilities.

5. Snap-E
Fill the backlog with actionable user stories.



7. Cloud Modernization Patterns
Reuse and document the learned modernization patterns as recipes in a "cookbook" (e.g. wiki). Consolidate knowledge and accelerate team velocity.

6. Produce tested and working code
Take low-level design decisions, select technical patterns to use, and implement user stories. Time-bounded set of 1-week iterations.

3. VMware **Modernization** Program

Platform 측면의 접근법

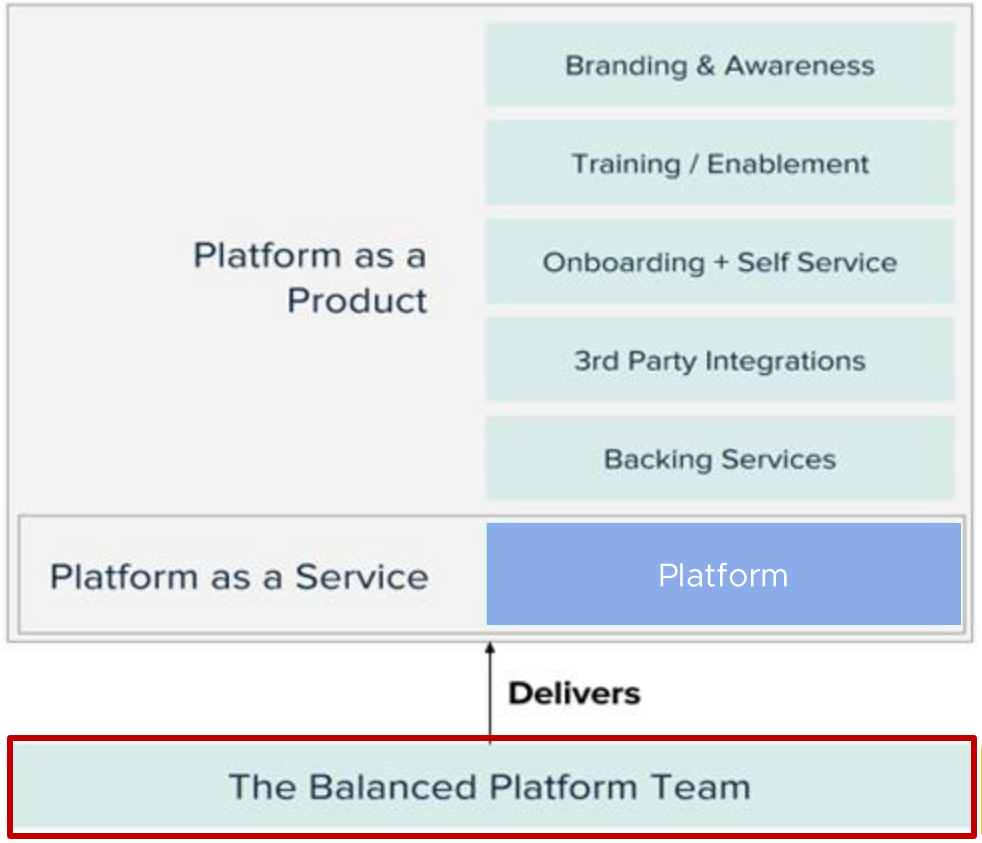
4 Key Questions

1. Platform as a Product 사상을 이해하고 있는 팀인가?
2. 플랫폼 Team의 목표에 따른 역할과 그에 따른 역량 강화는 어떻게 할 것인가?
3. K8S 기반의 플랫폼 위에 어떤 Tooling을 선택하고 적용할 것인가?
4. 더 많은 애플리케이션을 Hosting 하기 위한 플랫폼 팀의 전략은 무엇인가?

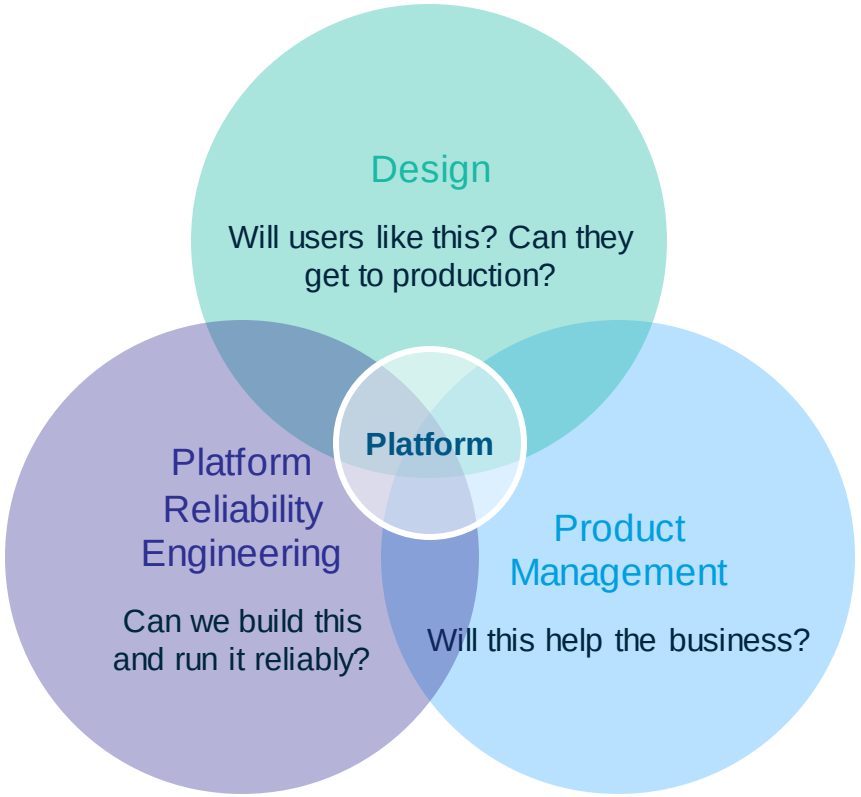
Platform 측면의 접근법

1. Platform as a Product 사상을 이해하고 있는 팀인가?
2. 플랫폼 Team의 목표에 따른 역할과 그에 따른 역량 강화는 어떻게 할 것인가?

Platform as a Product

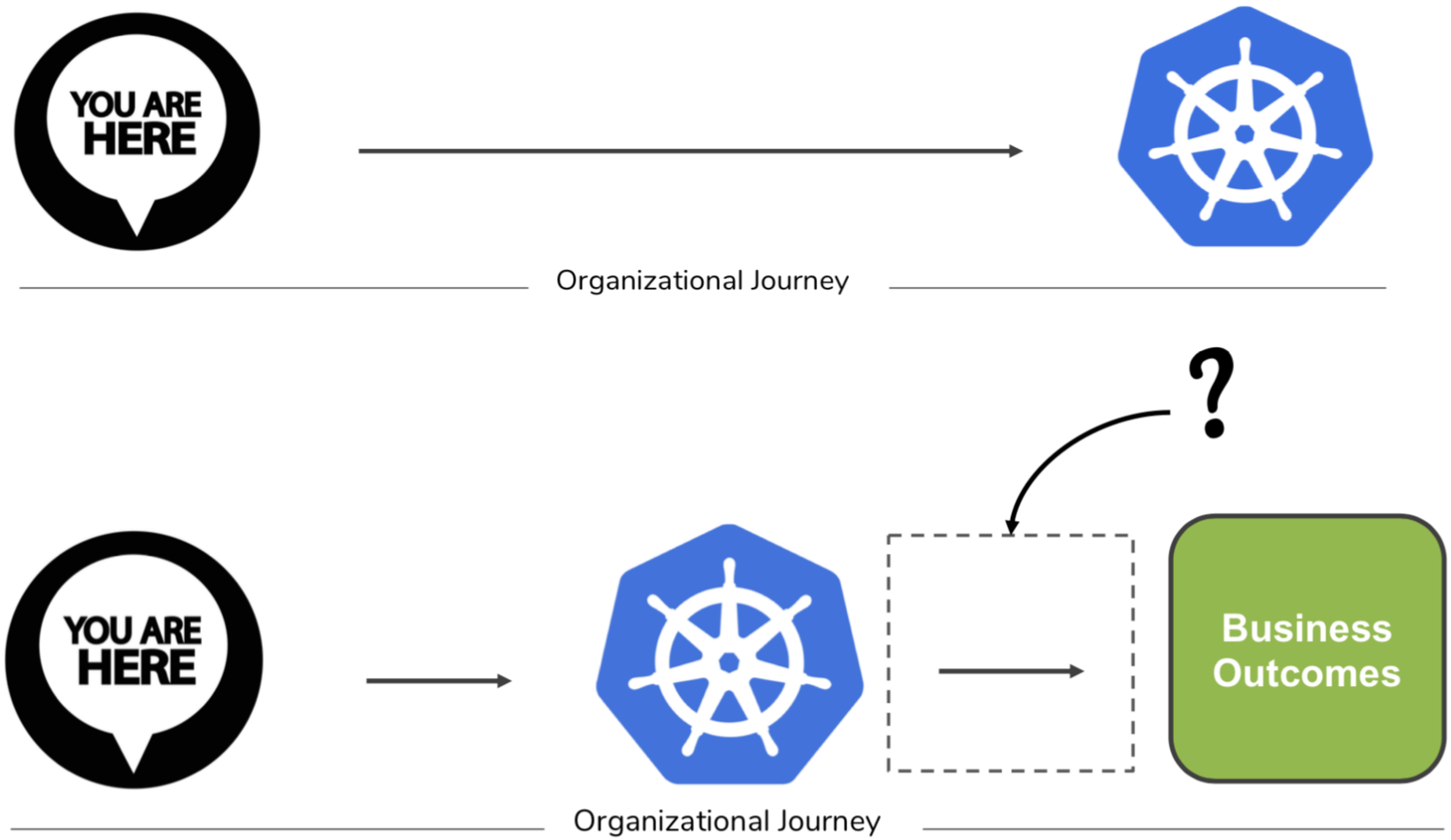


Balanced Platform Team



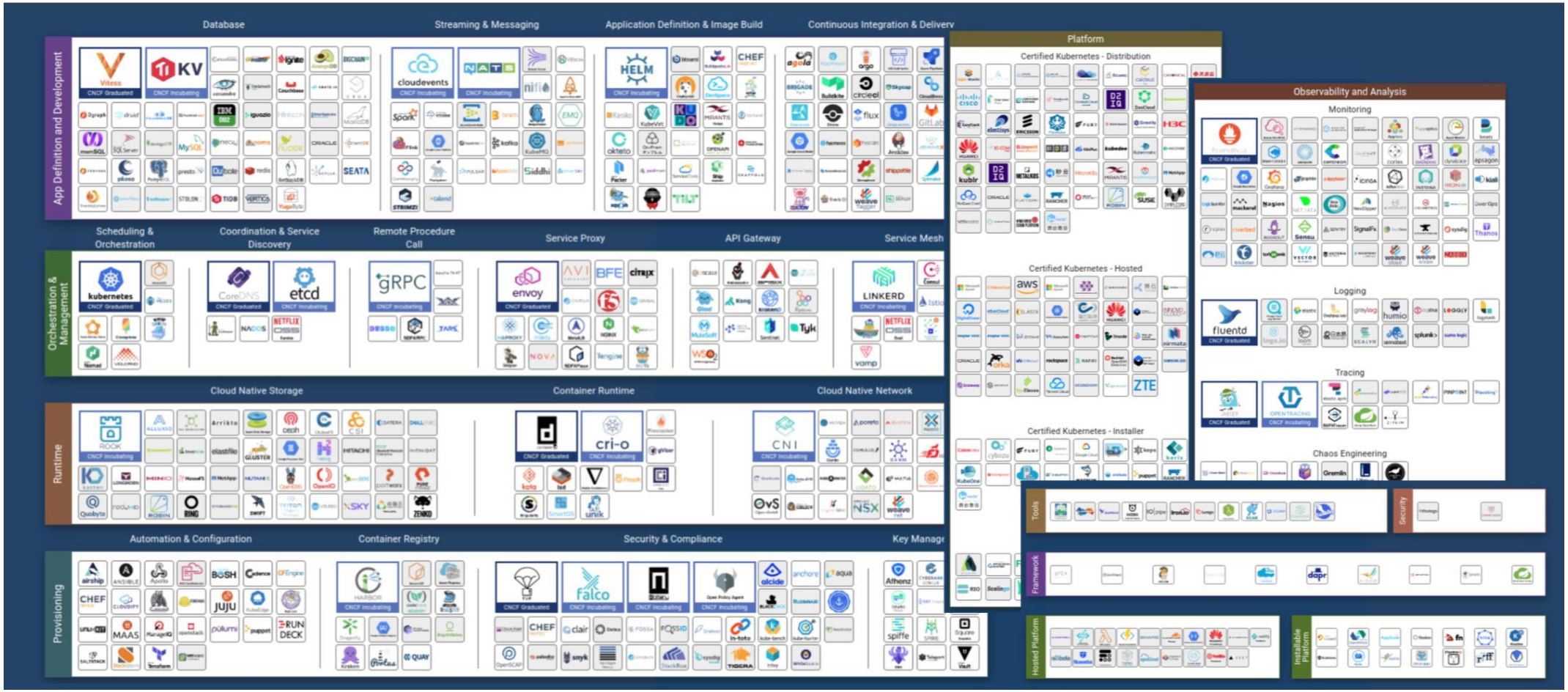
Platform 측면의 접근법

3. K8S 기반의 플랫폼 위에 어떤 Tooling을 선택하고 적용할 것인가?
4. 더 많은 애플리케이션을 Hosting 하기 위한 플랫폼 팀의 전략은 무엇인가?



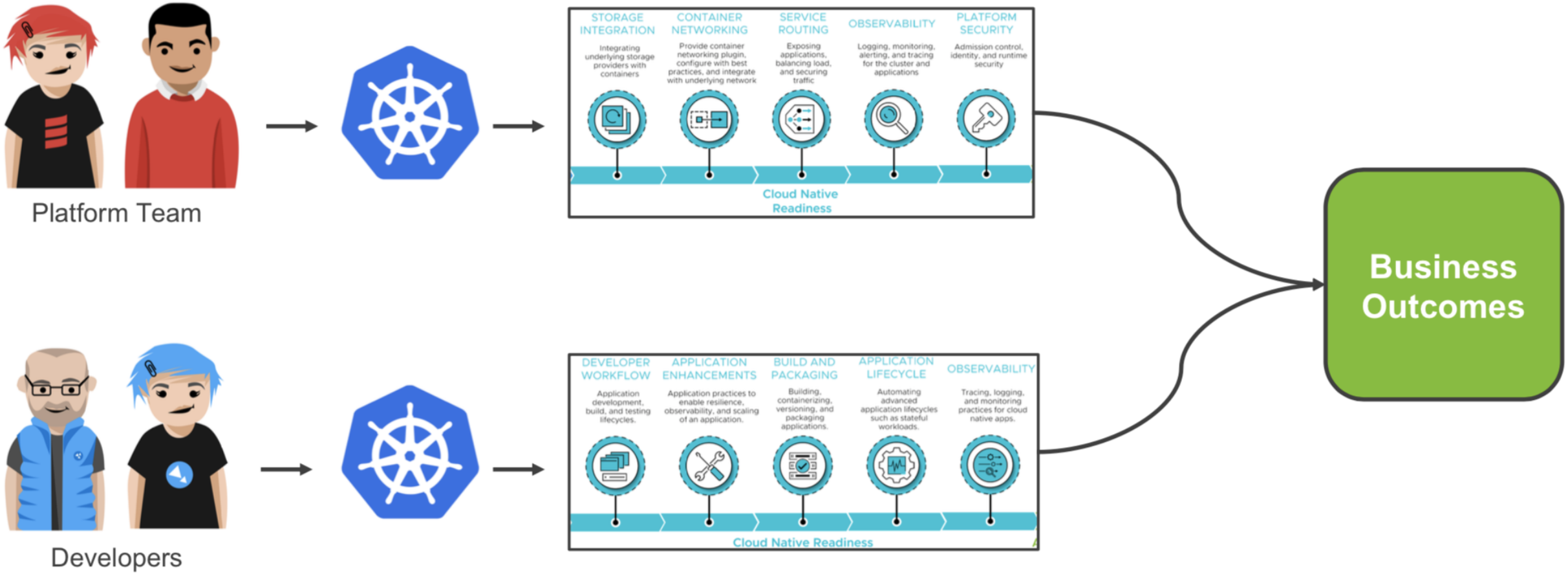
Platform 측면의 접근법

3. K8S 기반의 플랫폼 위에 어떤 Tooling을 선택하고 적용할 것인가?
4. 더 많은 애플리케이션을 Hosting 하기 위한 플랫폼 팀의 전략은 무엇인가?



Platform 측면의 접근법

3. K8S 기반의 플랫폼 위에 어떤 Tooling을 선택하고 적용할 것인가?
4. 더 많은 애플리케이션을 Hosting 하기 위한 플랫폼 팀의 전략은 무엇인가?



Platform 측면의 접근법

4 Key Questions

1. Platform as a Product 사상을 이해하고 있는 팀인가?
2. 플랫폼 Team의 목표에 따른 역할과 그에 따른 역량 강화는 어떻게 할 것인가?
3. K8S 기반의 플랫폼 위에 어떤 Tooling을 선택하고 적용할 것인가?
4. 더 많은 애플리케이션을 Hosting 하기 위한 플랫폼 팀의 전략은 무엇인가?

Platform Dojo

K8S Service

Platform 측면의 접근법

Platform Dojo & K8S Services

1. Platform Dojo

VMWare Tanzu TEAM		Your TEAM
Product Manager	⟷	Product Manager
PCFS Architect / PRE	⟷	Platform Reliability Engineer
PCFS Architect / PRE	⟷	Platform Reliability Engineer

플랫폼 팀은 플랫폼의 공통서비스 개발, 브랜드, 프로모션, 교육, 채용 등에 걸친 포괄적인 역할 정의가 필요합니다.

Platform as a Product

제품(Product)는 항상 "유지"만 되어서는 안되고, 요구에 따라 "발전" 되고 "업데이트" 되어야 합니다.

SRE

오퍼레이션을 "소프트웨어" 방식으로 보며, Toil을 최소화 하며, 어플리케이션의 속도와 안정성을 함께 추구하는 팀입니다.

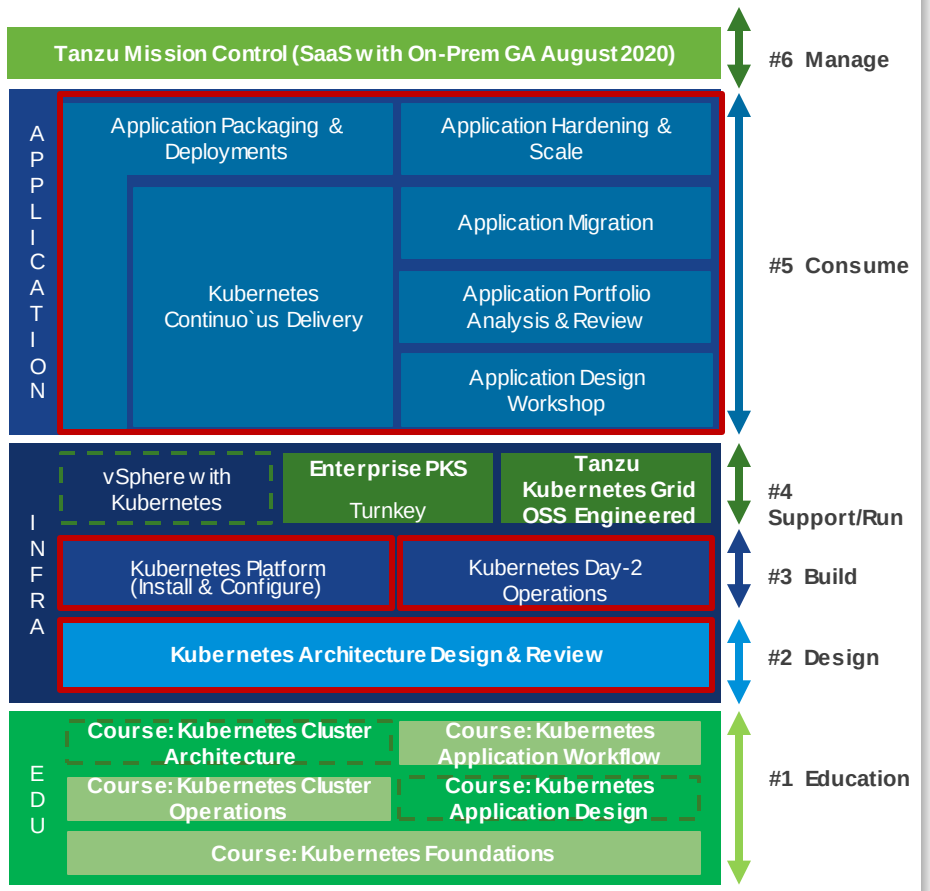
Dedicated Balanced Team

작은 규모의 플랫폼 팀은 수천 개의 앱과 "수천 명의 개발자"의 요구를 받아들일 수 있어야 합니다.

Platform as a Product

플랫폼이 Production으로 갈 수 있는 유일한 통로로서 역할을 해야 합니다.

2. K8S Services



2가지 측면의 고려 사항

"How do we move toward

modern apps

with greater velocity?"

"How do we operate

modern infrastructure

with more stability at scale?"

Cloud-Native Day
2020 Korea
LIVE!

Thank you

Jung In Lee (ljungin@vmware.com)

vmware®