Open Networking Conference Japan 2019

アプリケーションとネットワークの融合

サービスメッシュが注目される背景、課題と展望

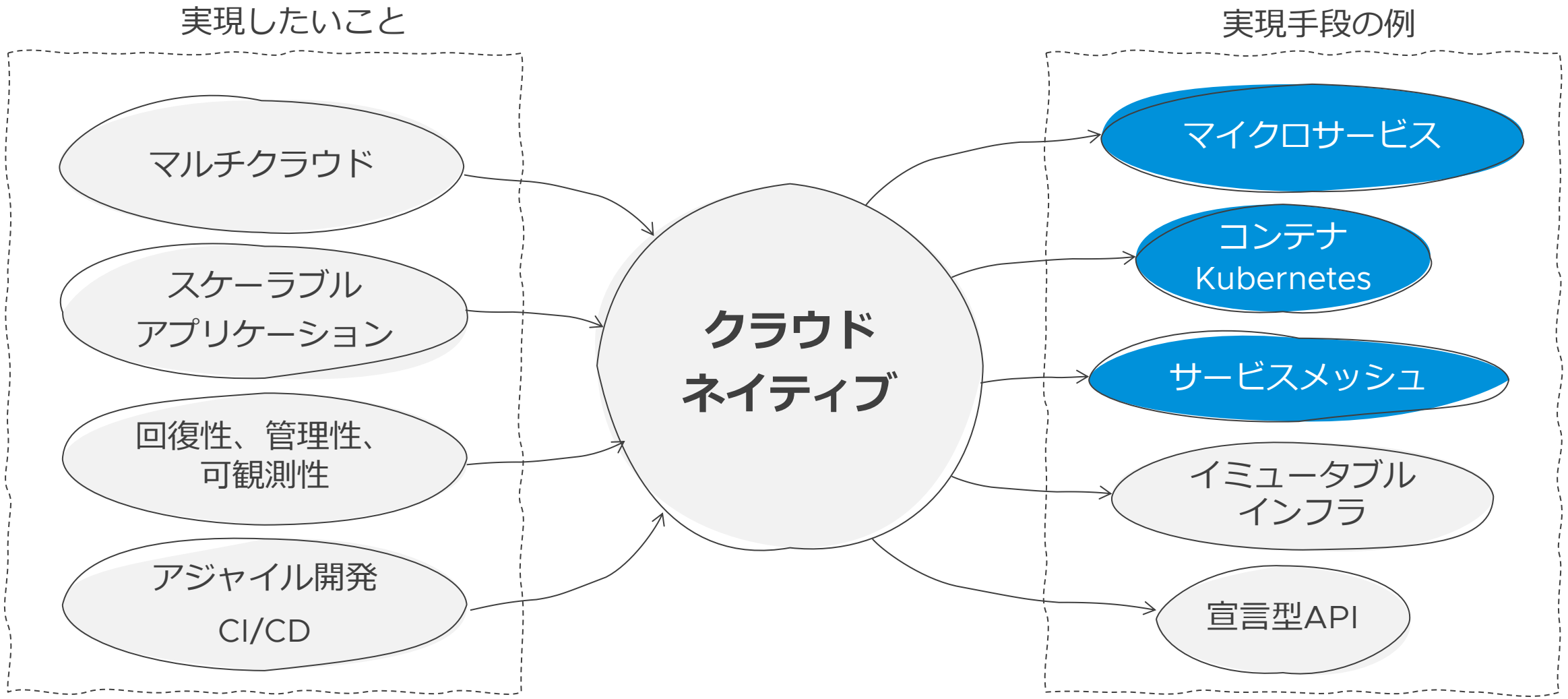
中奥洋志彦
シニアシステムズエンジニア、CTO Ambassador
ヴイエムウェア株式会社

2019年11月

vmware

クラウドネイティブを実現するための手段

マイクロサービス、コンテナ、サービスメッシュ



https://github.com/cncf/toc/blob/master/DEFINITION.md

アプリケーションの変革

Application Transformation

複雑性 – 成長にともない複雑さが増大し、全体の理解が困難になる

俊敏性を損なう – 長いリリースサイクルと長いチェンジウィンドウ

可用性の低下 – 一つのバグがシステム全体に影響を及ぼす可能性

技術革新の遅れ – 実行環境のスタックに対する長期間のコミットメントが必要になる

マイクロサービスとは？

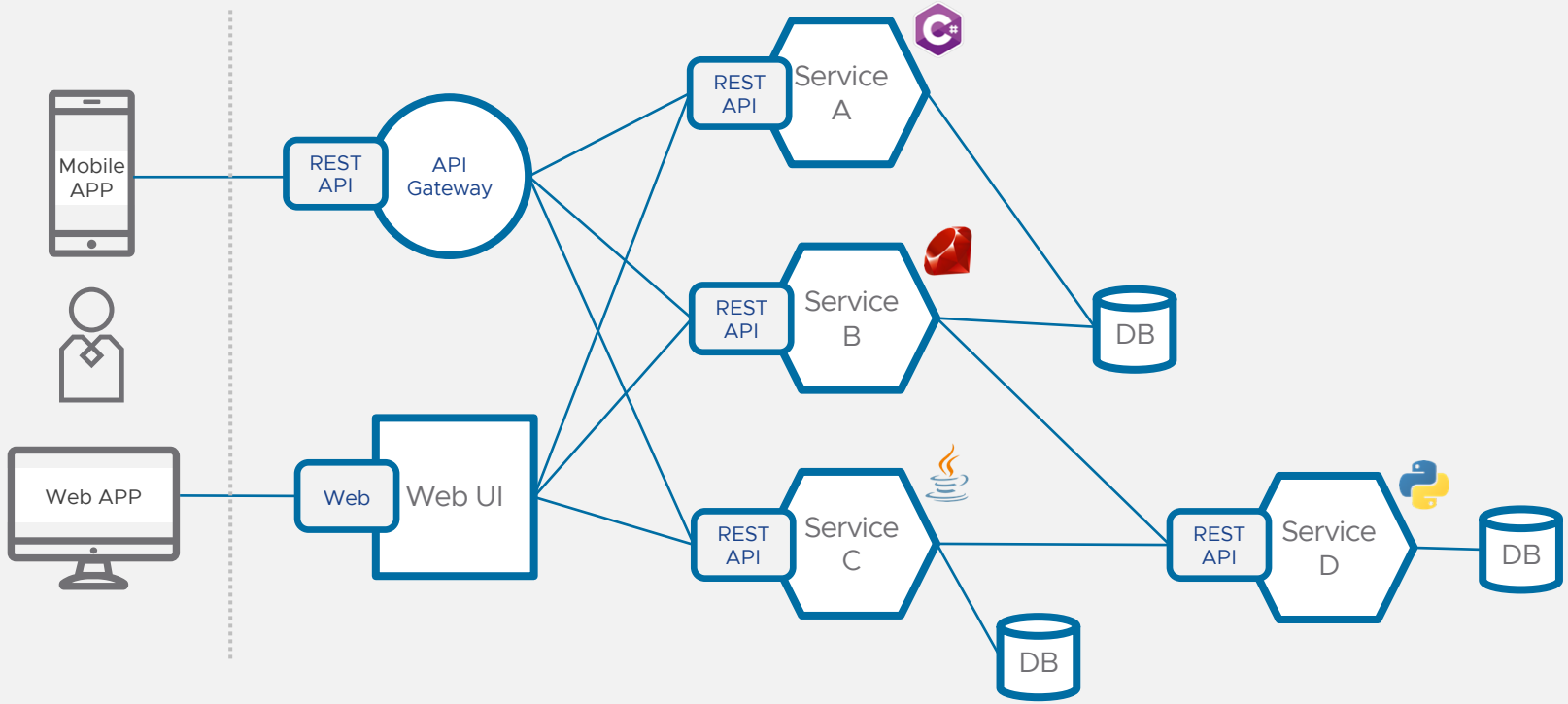
マイクロサービス != コンテナ

アプリケーションの機能を、異なるチーム、異なる言語で開発された小さなソフトウェアの単位に分割するという考え方

マイクロサービスの間では、言語非依存の API を用いて通信を行う

マイクロサービスを提供するホストは VM でも構わないが、より小さなフットプリントで実現できるコンテナの方が適している

マイクロサービス アーキテクチャの利点



シンプル – 各サービスが独立しており、個別のアップグレードも可能

柔軟性 – 水平展開が容易で、多様なプラットフォームへの展開も可能

回復性 – 故障影響範囲を最小化し易い

革新性 – 新しい技術を迅速に展開できる、新しいフレームワークや開発言語への適合

サービスメッシュは
何を解決するか

マイクロサービス: 数多くの利点と、新しい課題

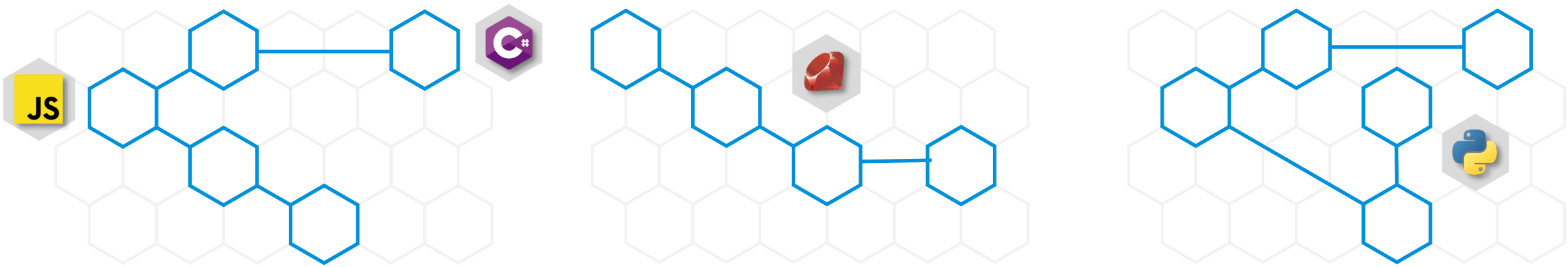
いかにして安定的にマイクロサービスを監視、制御し、セキュリティを担保するか

多言語・多フレームワークへの習熟

セキュアな接続性とトラフィック制御

可観測性 - マイクロサービスの正常性の監視

プラットフォームをまたがる視覚化と監査



Enterprise PKS

マイクロサービス: アーキテクチャの変革

コード、ライブラリ、
フレームワーク

API ゲートウェイ

サービスメッシュと
サイドカー

アプリ開発者

高い依存性、複雑さ

コーディング

長いリリースサイクル

DevOps

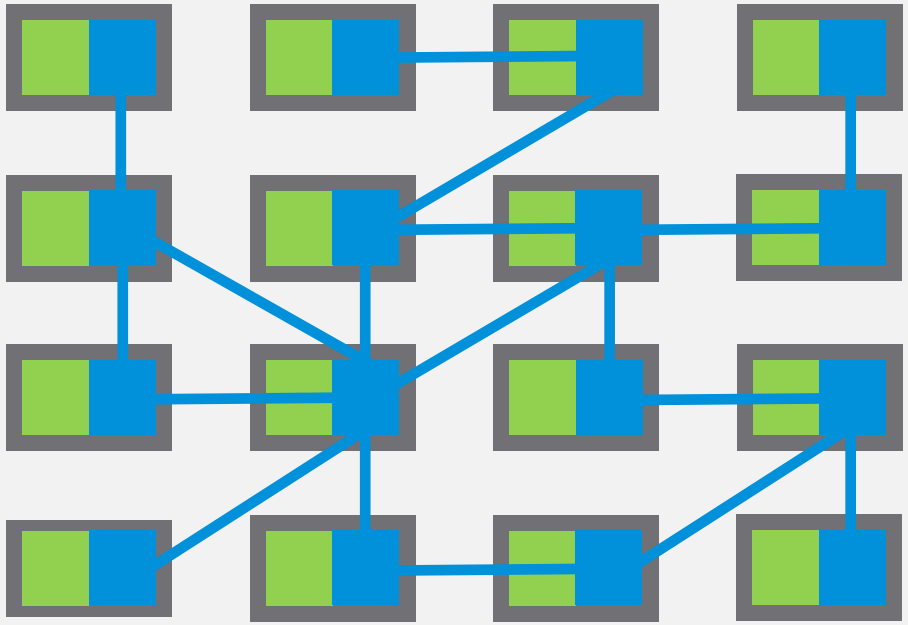
高い抽象度、シンプル

自動化とオーケストレーション

継続的開発 (CI/CD)

サービスメッシュ

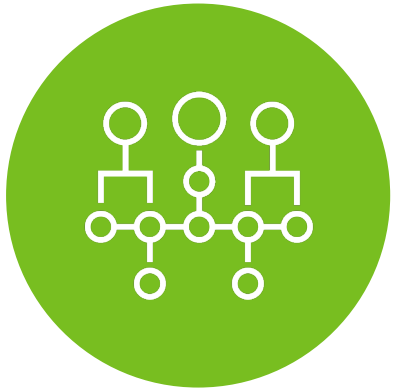
インフラ・ネットワークに依存せずにマイクロサービスの課題を解決



サイドカー プロキシー

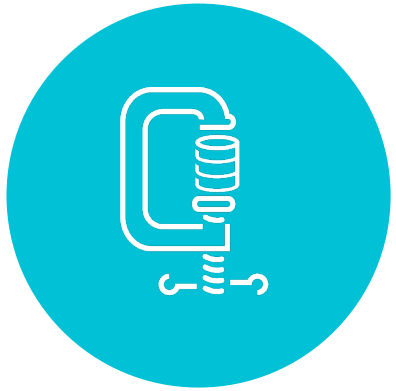
- 共用インフラから切り離された専用の通信レイヤー
- サービス間通信を扱う
- 複雑なサービス間トポロジーを管理
- 軽量なネットワークプロキシーの配列
- アプリケーションプロセスと一緒に展開される
- アプリケーション側で認識する必要がない

サービスメッシュの利点



サービスの検出

サービスは互いを
見つけることができ
る



回復性

ビルトインの堅牢な
フレームワーク、
ロードバランシング
とテスト機能

設定の柔軟さ

サービスの実行時に
動的に設定できる

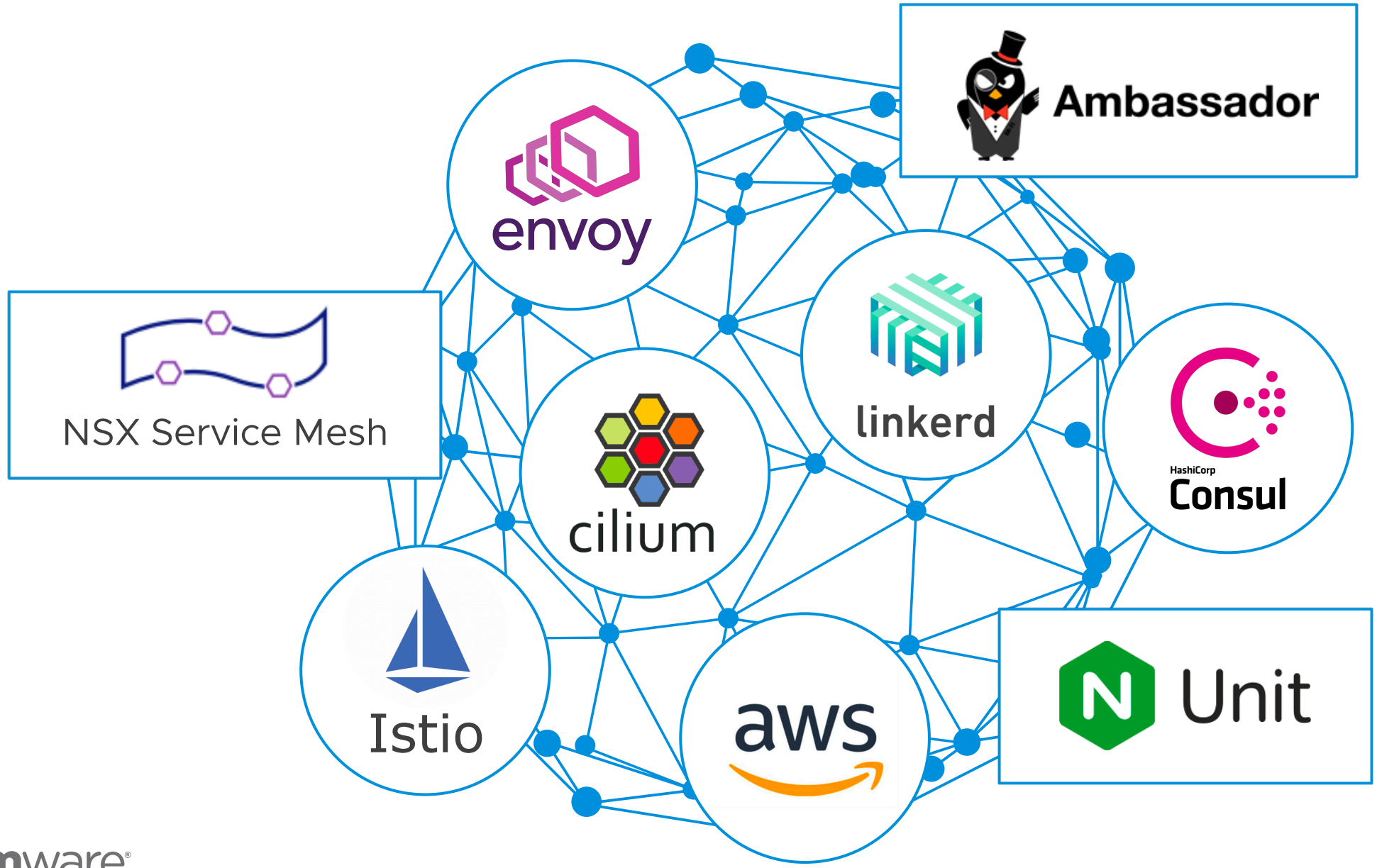
可観測性

標準的なメトリック、
ロギング、監視と
分散トレーシング

セキュリティ

サービス間通信を
暗号化して保護

サービスメッシュの世界



Istio とは

istio / istio

Watch 845 | Star 15,816 | Fork 2,512

Code | Issues 834 | Pull requests 166 | Wiki | Insights

Connect, secure, control, and observe services. https://istio.io

microservices | service-mesh | lyft-envoy | kubernetes | api-management | circuit-breaker | polyglot-microservices | enforce-policies | proxies

microservice | envoy | consul | nomad | request-routing | resiliency | fault-injection

7,069 commits | 26 branches | 45 releases | 333 contributors | Apache-2.0

コントロールプレーン

Google

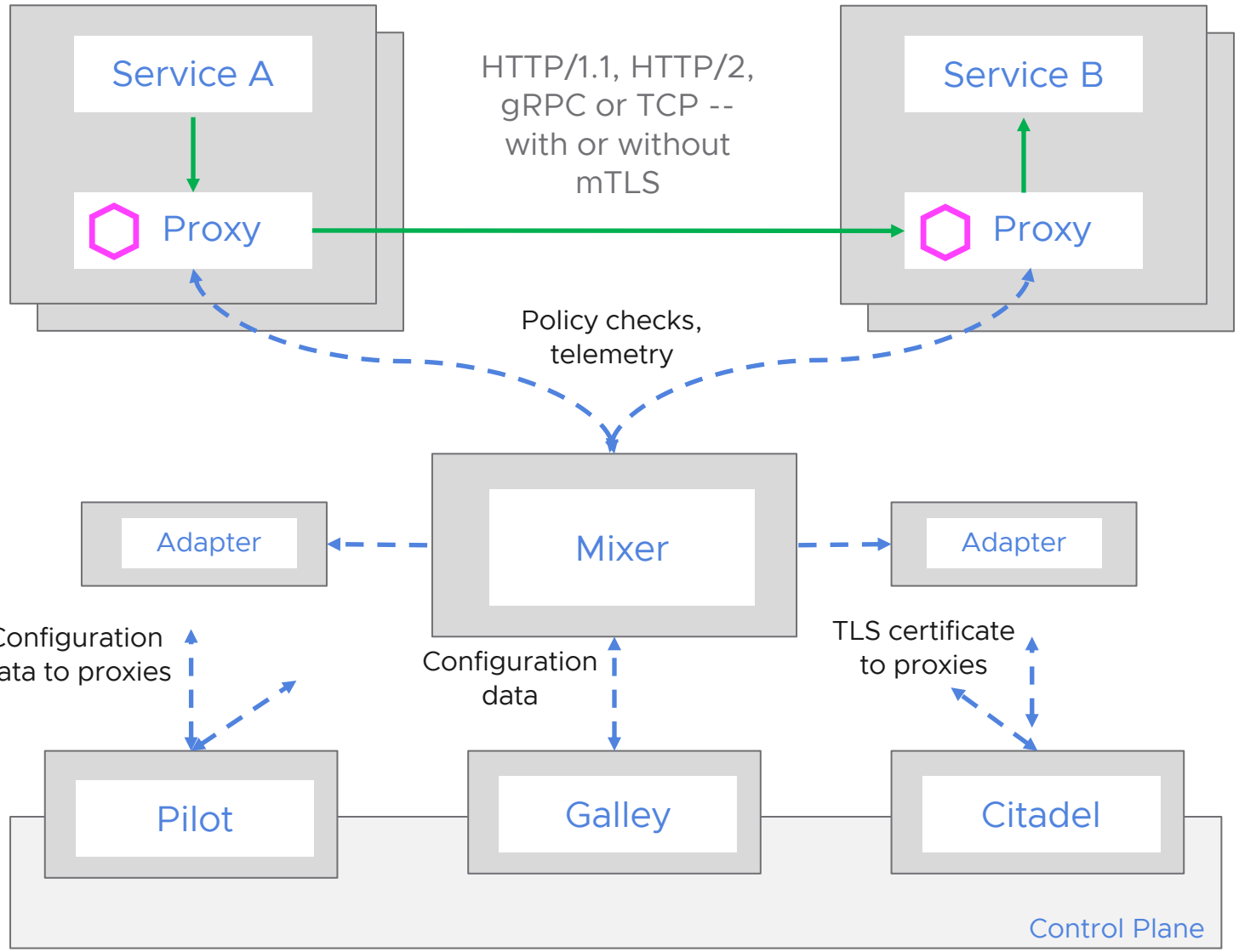
IBM

データプレーン

lyft

2017年5月にプロジェクト開始　　2018年7月に 1.0 に到達

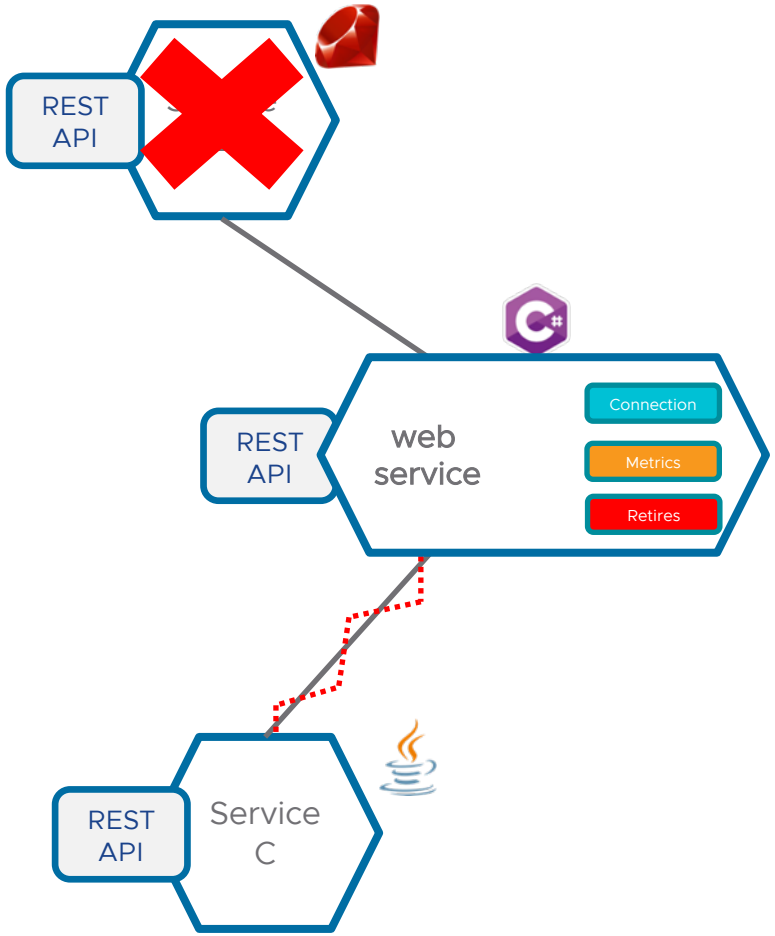
Istio アーキテクチャ



Source: https://istio.io/docs/concepts/what-is-istio/

サービス間コネクション管理のオフロード

サービス検出、暗号化、エラー検出とモニタリング



他のサービスとの連携

- サービスの検出
- コネクション情報 (通信の秘匿性、暗号化)
- 異なる言語のサポート
- エラーハンドリング

ビジネスロジック – データを取得して Web に表示する

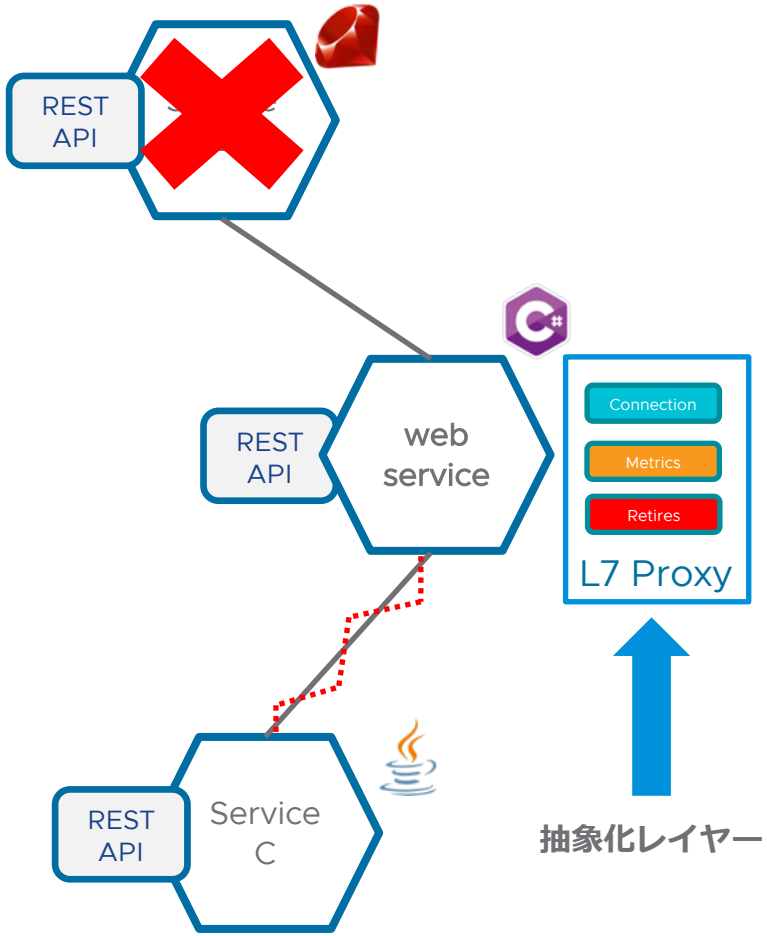
遅延の検出と対応

- メトリックの収集
- メトリックの送信
- セルフヒーリング

エラー検出とハンドリング

サービス間コネクション管理のオフロード

サービス検出、暗号化、エラー検出とモニタリング



他のサービスとの連携

- サービスの検出
- コネクション情報 (通信の秘匿性、暗号化)
- 異なる言語のサポート
- エラーハンドリング

ビジネスロジック – データを取得して Web に表示する

遅延の検出と対応

- メトリックの収集
- メトリックの送信
- セルフヒーリング

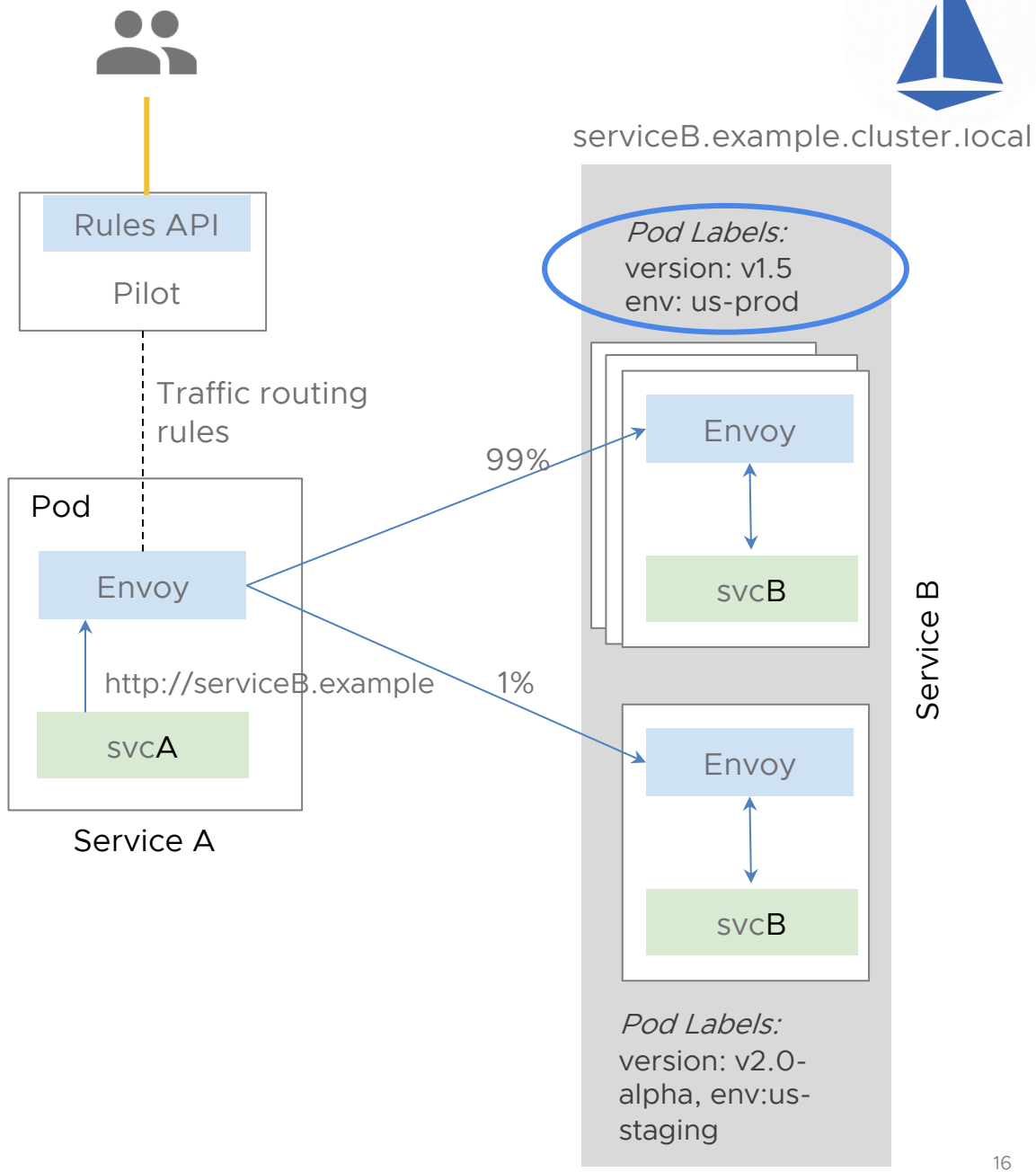
エラー検出とハンドリング

トラフィック スプリッティング

```
// A simple traffic splitting rule

destination: serviceB.example.cluster.local
match:
  source: serviceA.example.cluster.local
route:
- tags:
    version: v1.5
    env: us-prod
  weight: 99
- tags:
    version: v2.0-alpha
    env: us-staging
  weight: 1
```

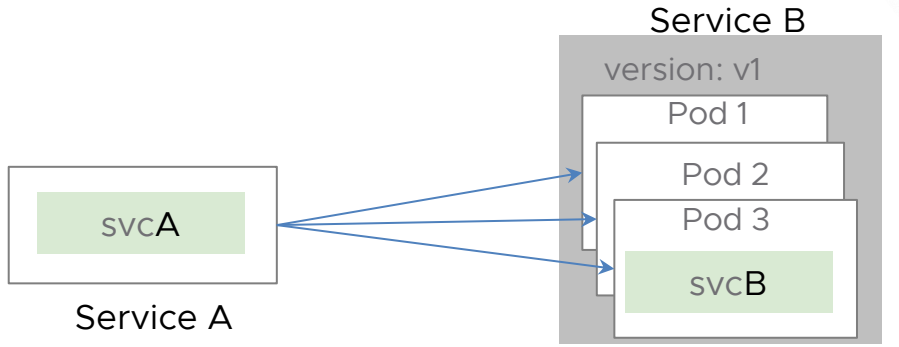
インフラから切り離された
トラフィック制御

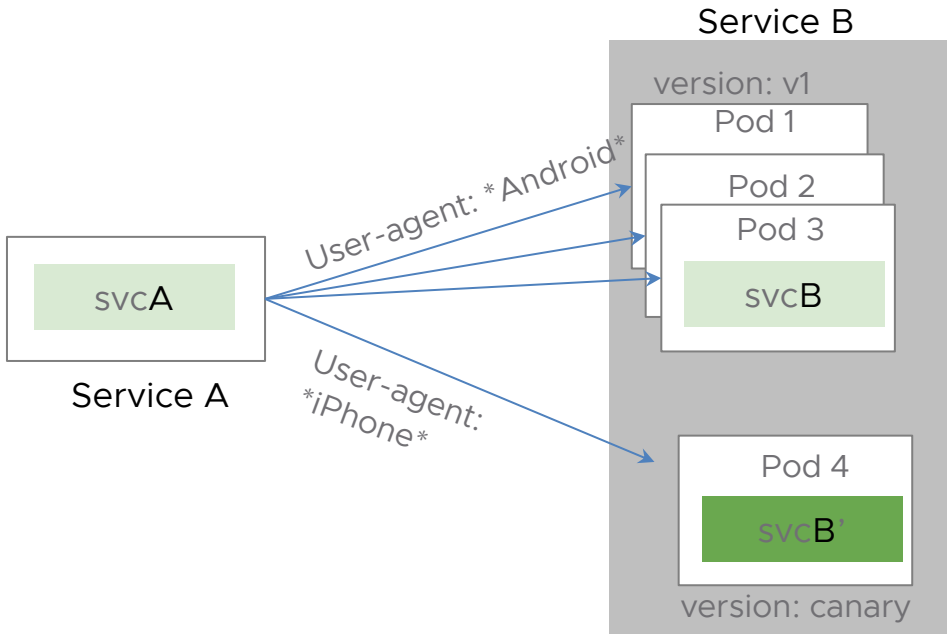


トラフィック ステアリング

```
// Content-based traffic steering rule

destination: serviceB.example.cluster.local
match:
  httpHeaders:
    user-agent:
      regex: ^(.*?;)?(iPhone)(;.*)?$
precedence: 2
route:
- tags:
    version: canary
```





コンテンツに基づく
トラフィックステアリング

サービスメッシュの展望

理想のサービスメッシュに向けて

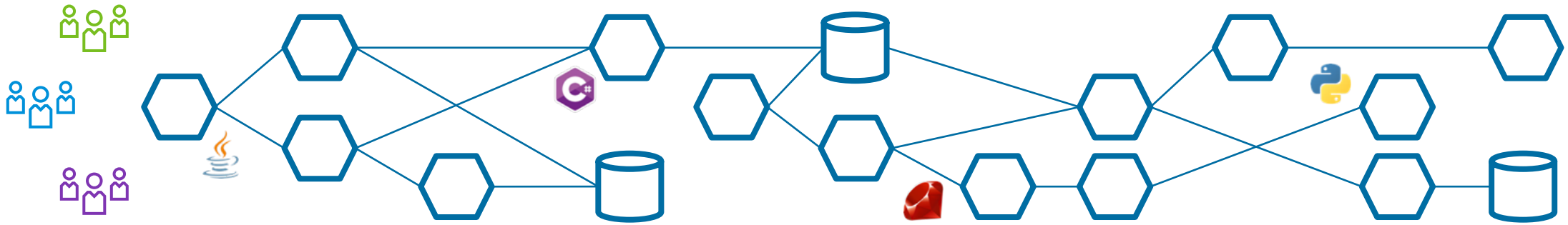
安定したサービス検出、視覚化、制御とセキュリティ

マルチプラットフォーム、
マルチクラウドの
フェデレーション

中央集中型の
可視化と監視、
セキュリティ

ユーザ、サービス、
データのグローバルな
ポリシー管理

コンテナ以外の
プラットフォーム
への対応



Enterprise PKS

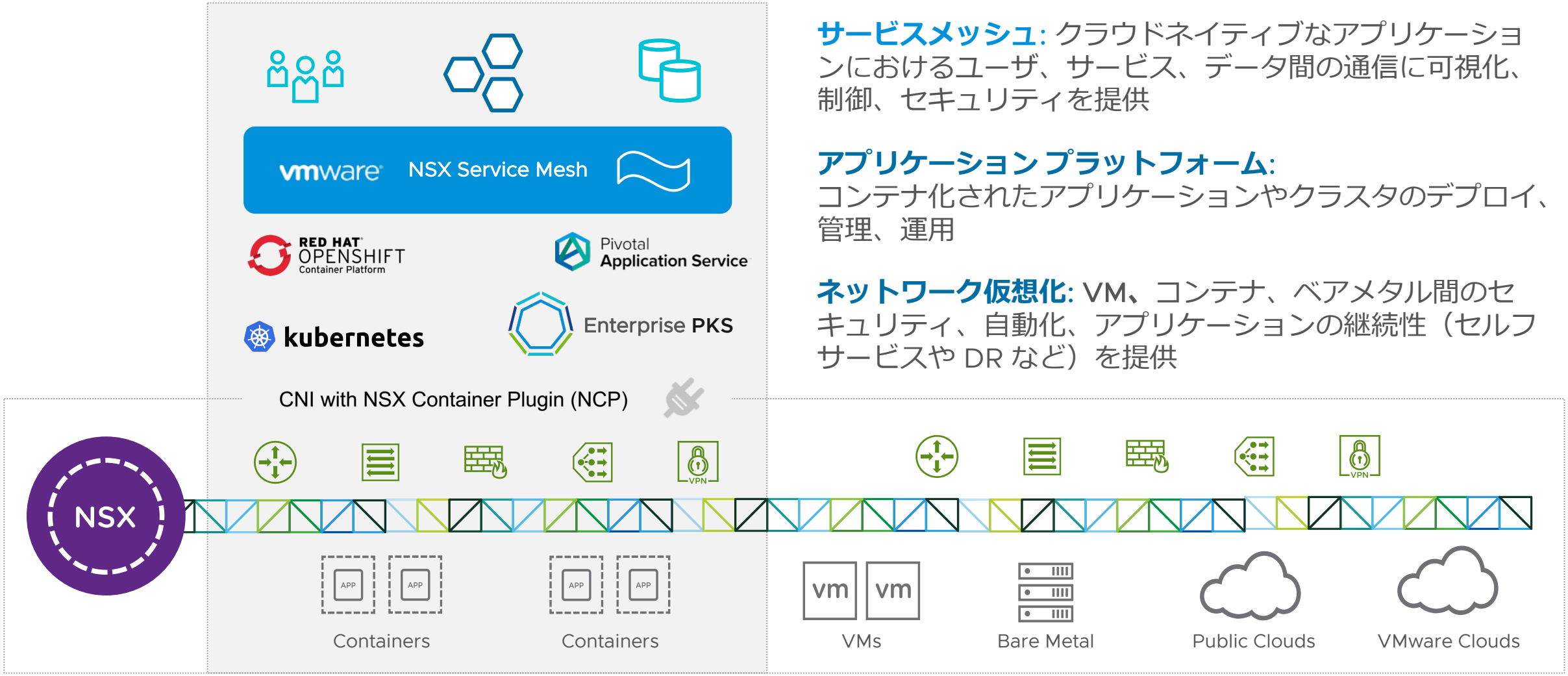
Pivotal Cloud Foundry

OPENSHIFT by Red Hat

Google KE

Amazon EKS

VMware NSX にサービスメッシュを拡張

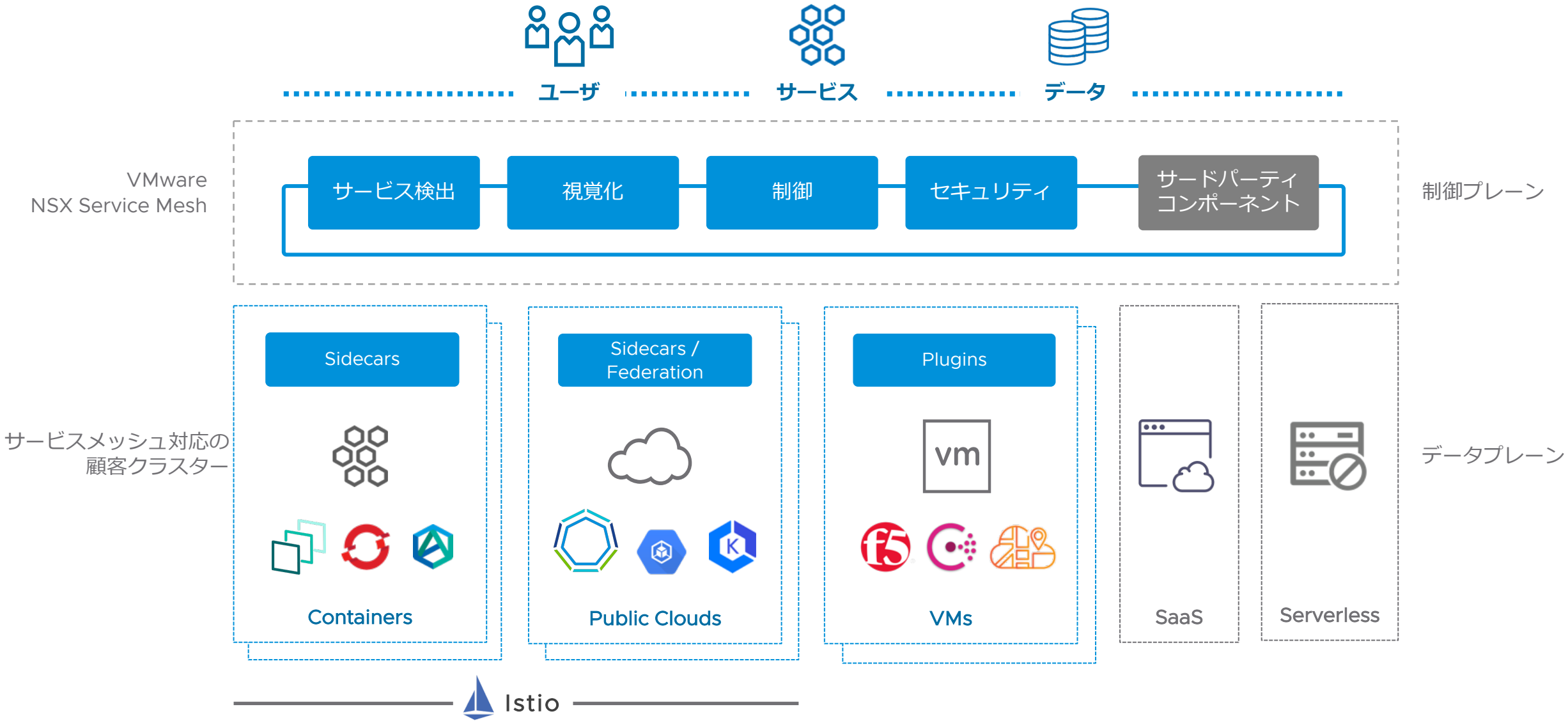


サービスメッシュ: クラウドネイティブなアプリケーションにおけるユーザ、サービス、データ間の通信に可視化、制御、セキュリティを提供

アプリケーション プラットフォーム: コンテナ化されたアプリケーションやクラスタのデプロイ、管理、運用

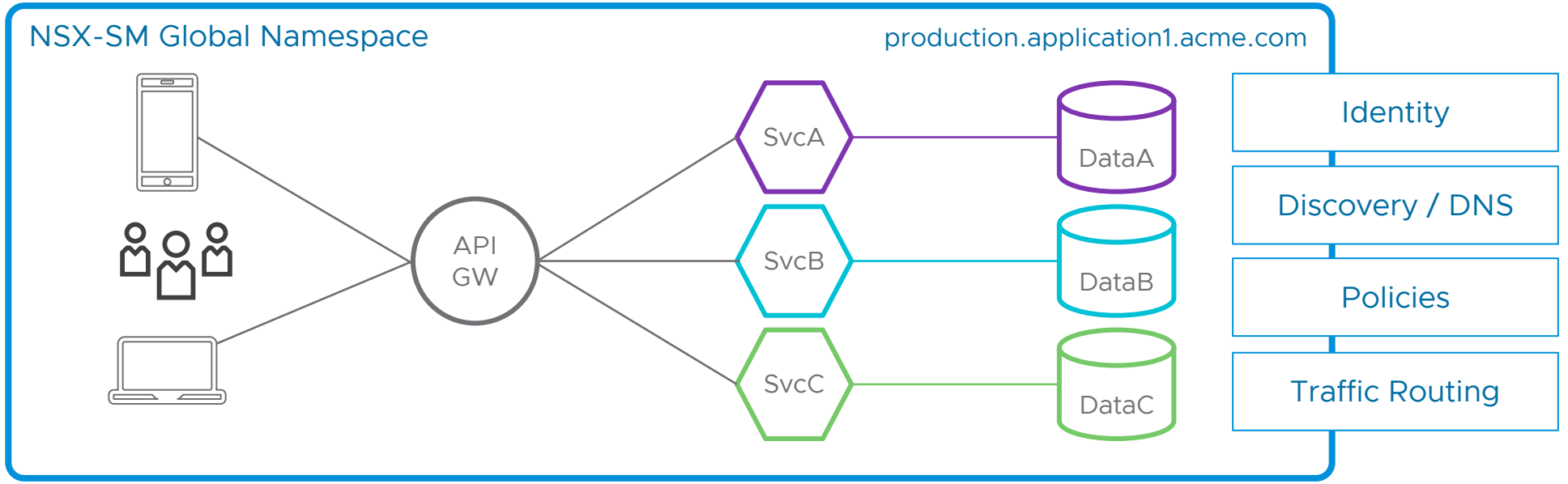
ネットワーク仮想化: VM、コンテナ、ベアメタル間のセキュリティ、自動化、アプリケーションの継続性（セルフサービスや DR など）を提供

VMware NSX Service Mesh のビジョン

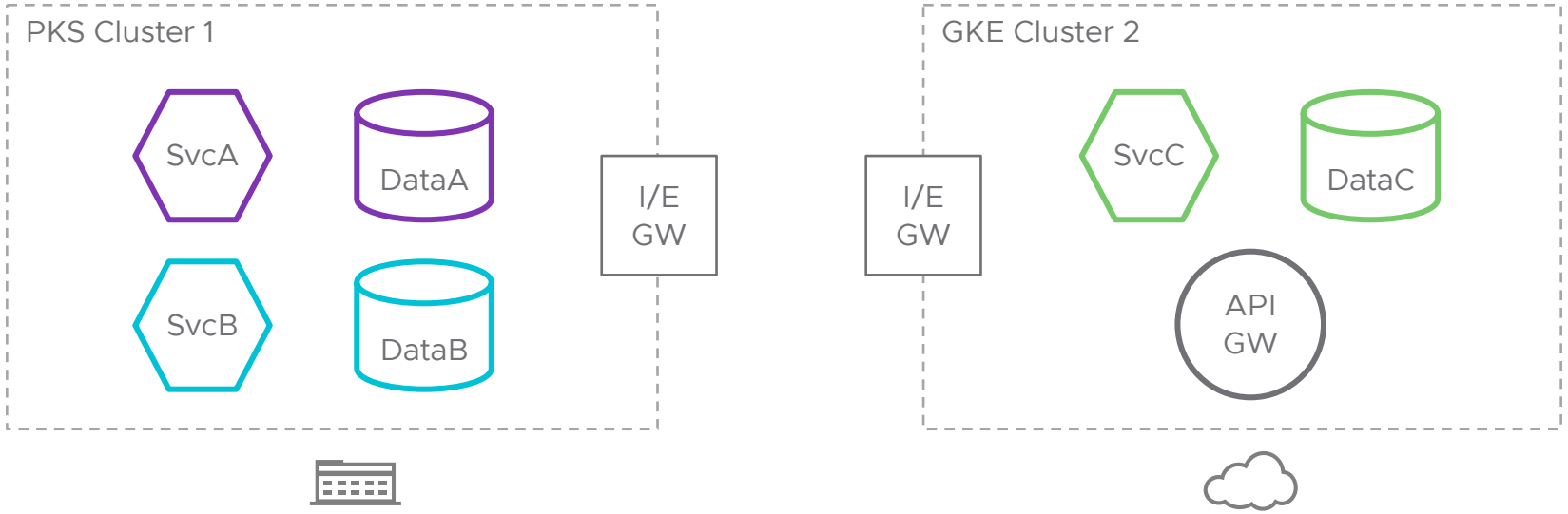


NSX Service Mesh Global Namespaces

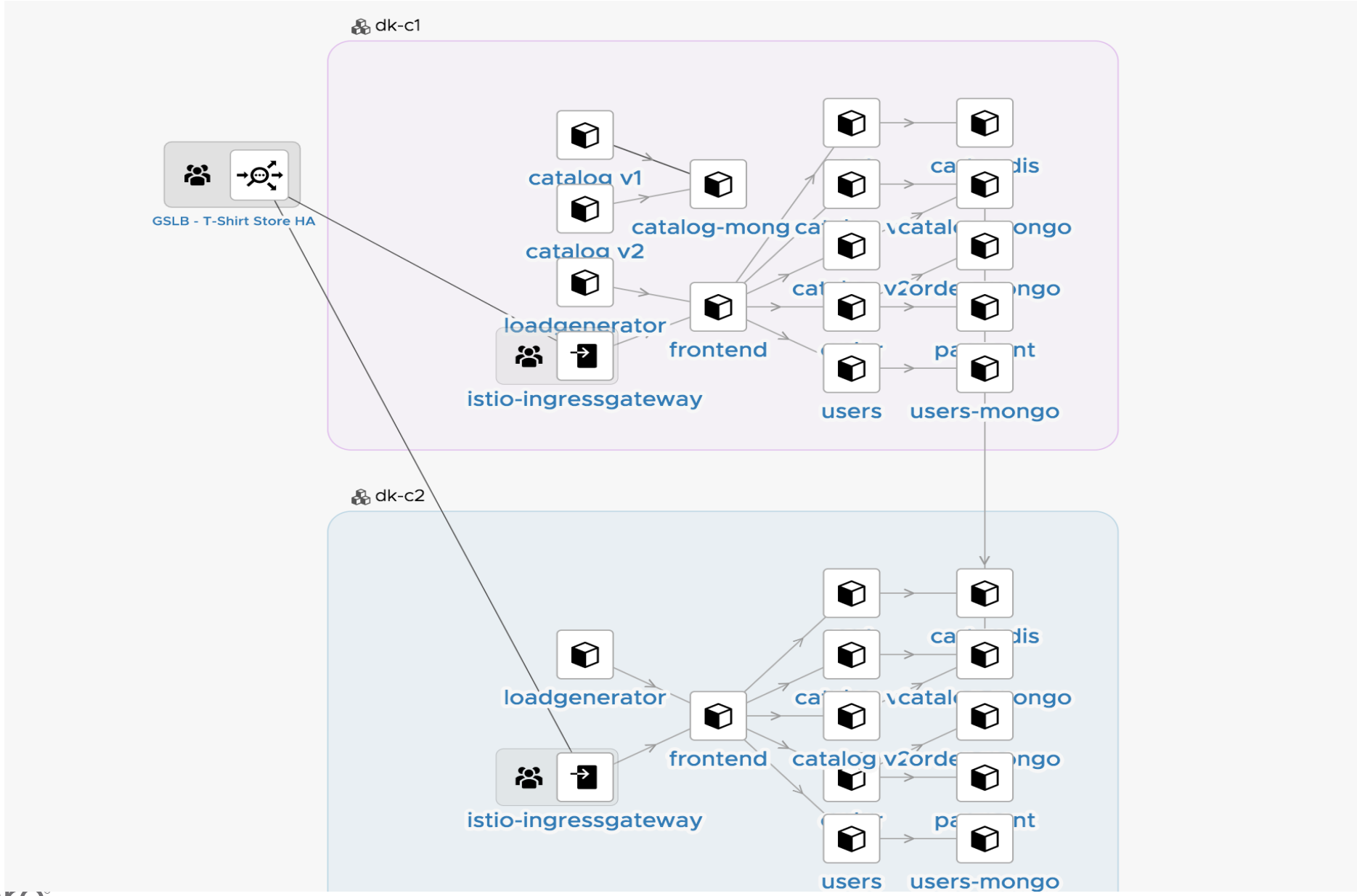
論理ビュー



物理ビュー



NSX Service Mesh Global Namespaces



まとめ

サービスメッシュのこれまでとこれから

サービスメッシュによる解決

アプリケーションから通信の機能を分離

- 異なる言語やフレームワークでの異なる実装が不要に
- mTLS を利用したセキュアな接続性
- カナリアリリース等を意識した、細やかなトラフィック制御
- 可観測性 - マイクロサービスで発生しているエラーや遅延の把握は容易に

サービスメッシュの将来

マルチプラットフォーム、マルチクラウド

集中型の可視化と監視

ユーザ、サービス、データのポリシー管理

集中型のセキュリティとコンプライアンス

コンテナ以外のワークロード

Thank You

vmware®