SONY

クラウドサービス基盤における
IaaSからPaaSへの転換と効率性の追求

June 2015

玉井　久視

会社紹介

- 商号
  - ソニー株式会社（Sony Corporation）
- 設立
  - 1946年（昭和21年）5月7日
- 本社所在地
  - 東京都港区港南1-7-1
- 主要営業品目
  - テレビ　液晶テレビ
  - デジタルイメージング　レンズ交換式一眼カメラ、コンパクトデジタルカメラ、ビデオカメラ
  - オーディオ・ビデオ　家庭用オーディオ、ブルーレイディスクプレーヤー/レコーダー、メモリ内蔵型携帯オーディオ
  - 半導体　イメージセンサー、その他の半導体
  - コンポーネント　電池、記録メディア、データ記録システム
  - プロフェッショナル・ソリューション　放送用・業務用機器
  - メディカル　メディカル関連機器
  - 他

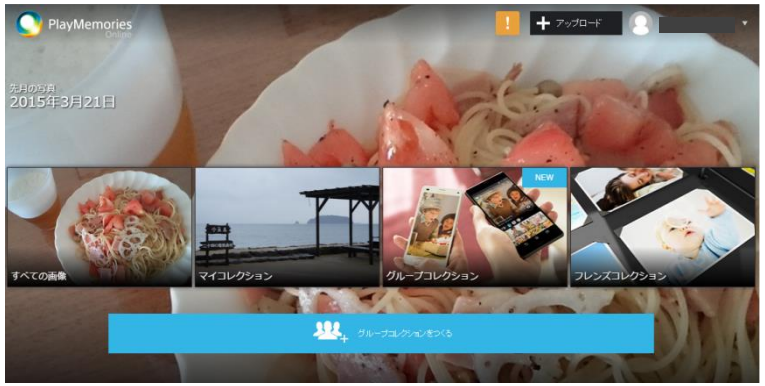
自己紹介

- 玉井　久視（Hisashi Tamai）
  - UXﾌﾟﾗｯﾄﾌｫｰﾑ UX･ﾏｰｹﾃｨﾝｸﾞ本部 ｸﾗｳﾄﾞ＆ｻｰﾋﾞｽｱﾌﾟﾘ開発運用部門 部門長
  - 1986年入社（51歳）
  - 略歴　　TV設計、Video設計、携帯設計に従事後
    サービスインフラ運用開発設計を担当

担当しているクラウドサービスと製品

AV系

PlayMemories Online



TV SideView

IoT系

Smart B-Trainer

Smart Tennis Sensor

Media系

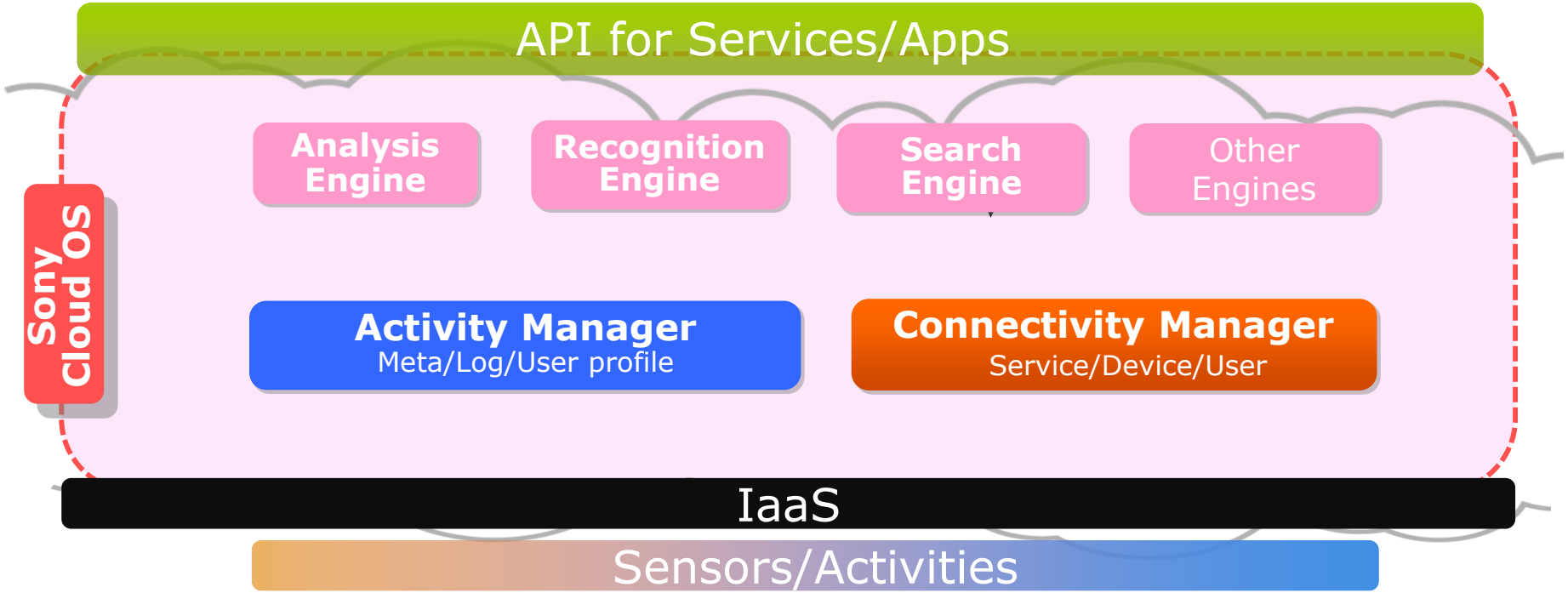
Socialife

見たい情報をセレクトして、１アプリで高速チェックできる、"自分専用"ニュースリーダー。

FacebookやTwitterの投稿、YouTube動画、ニュースやブログの記事をまとめてチェックできる、Android/Windows8アプリケーションです。

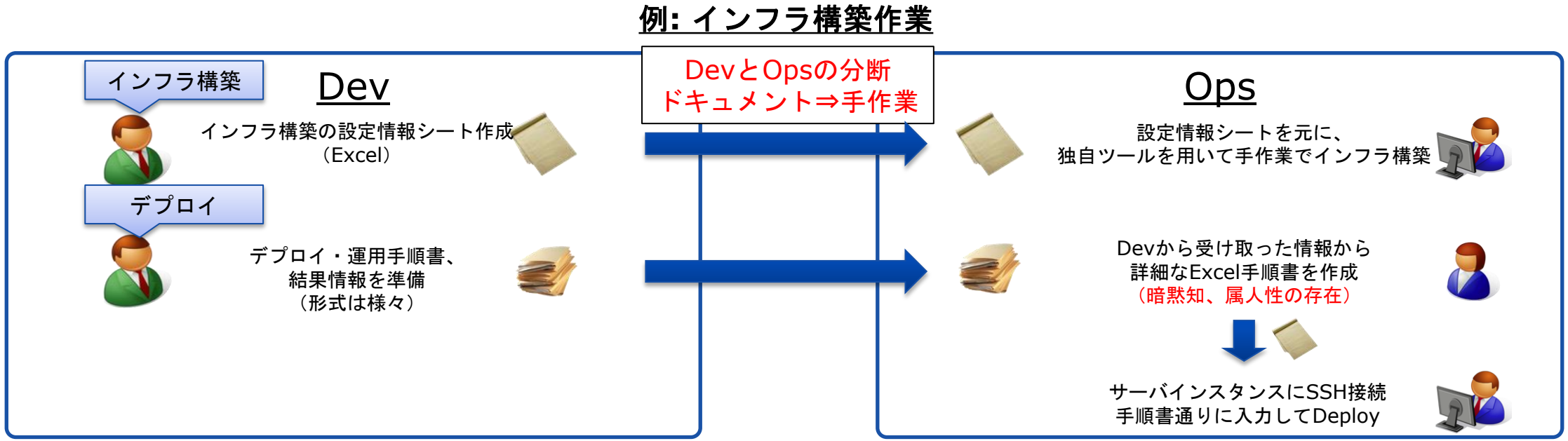
以前のシステム設計思想: OS的な考え方

- クラウド側の共通基盤にユーザプロファイル/分析機能/各種エンジンを集約
- 個別サービスはクラウド側の共通機能の組み合わせで実現



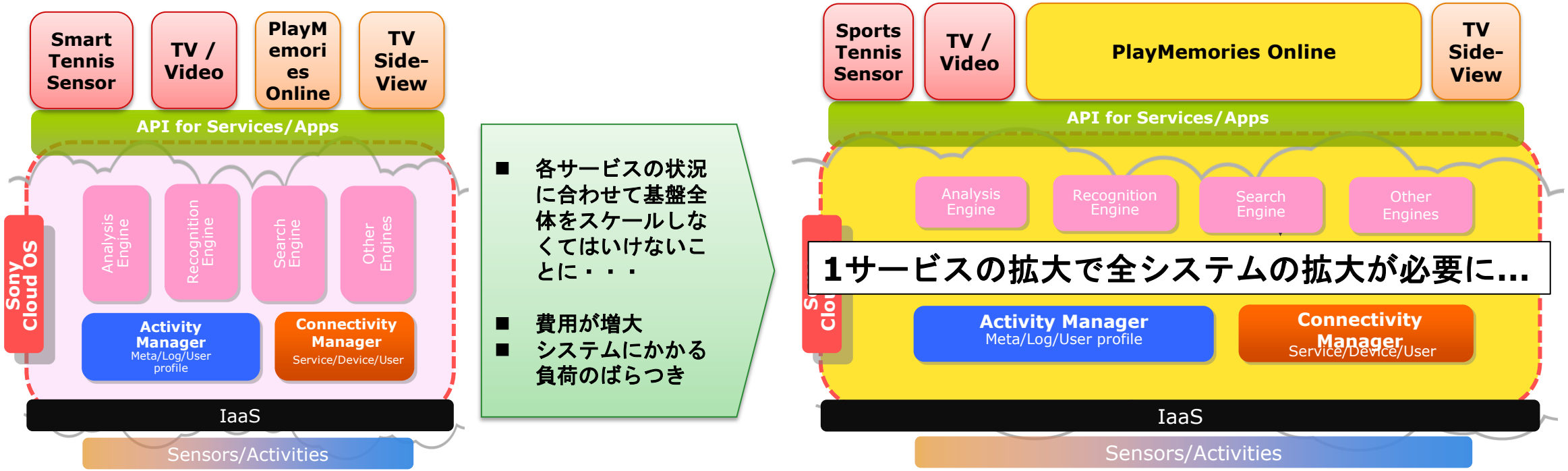
膨大な開発/運用工数

- 自社開発の、様々な仕組みを持った意欲的なシステム
- 独自システムゆえ、すべてを実現するには相当な工数が必要
  - DB等重要M/Wの一部内製 – 設計漏れ/問題
  - サービス特有の個別機能も基盤機能化 - 一般化により肥大化/複雑化
  - 構成管理等運用ツールも内製 – 自動化の仕組みも内製化が必須/世の中の自動化の流れに乗れない
  - サービスからの要求に対して、過去資産流用での対応の限界

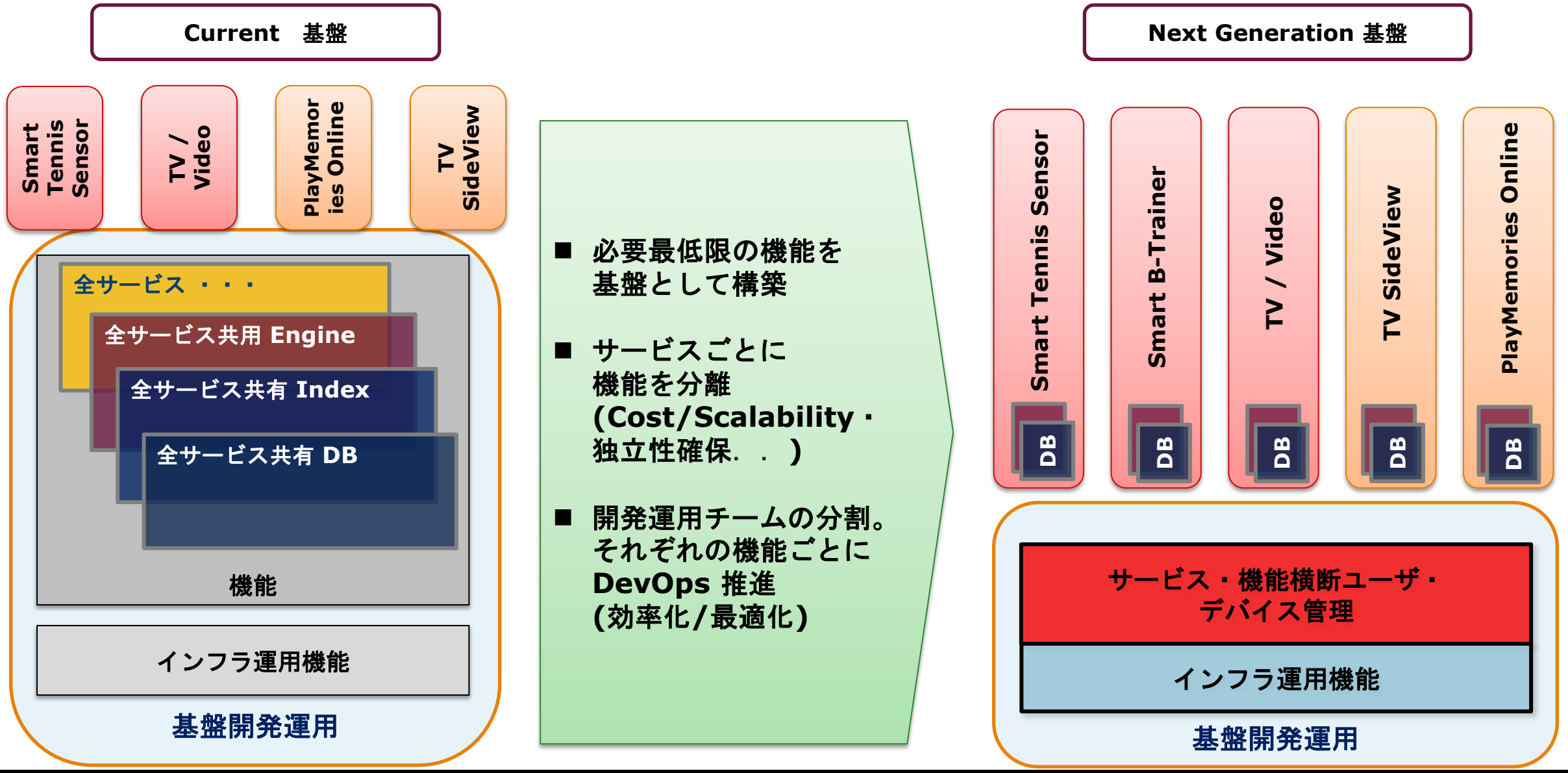


ユーザ数/データ量の増大による課題顕在化

- 各サービスの規模が拡大
  - 全サービスの事情に合わせてシステム規模を模索
  - スケールを変更するためにシステム停止が必要となり、頻繁な停止
  - 費用の増大（予算に入らない）
- DBの冗長化機能等、負荷増大で独自実装部分の問題が顕在化
  - 開発に手が回らず、メンテナンス、問題解決で忙殺



システム再構築の方針



考え方

- 共通基盤の再定義/再実装
  - サービス特有な機能を分離
  - 最小限の機能を提供
  - ユーザ認証/共通ユーザプロフィールに限定
- モダンなアーキテクチャ
  - 薄い共通コア（マイクロサービス連携）
  - 適切な言語（Go、JSなど）によるソフトウェアの開発効率化
  - Shared nothing architectureによるスケーラビリティ
- 少人数チームでのDevOps
  - 高品質なサーバアプリを短期間で内製
  - 最新のツール・外部サービス（Github.com, Docker Hub等)活用による、自動化と開発/運用効率化

目標

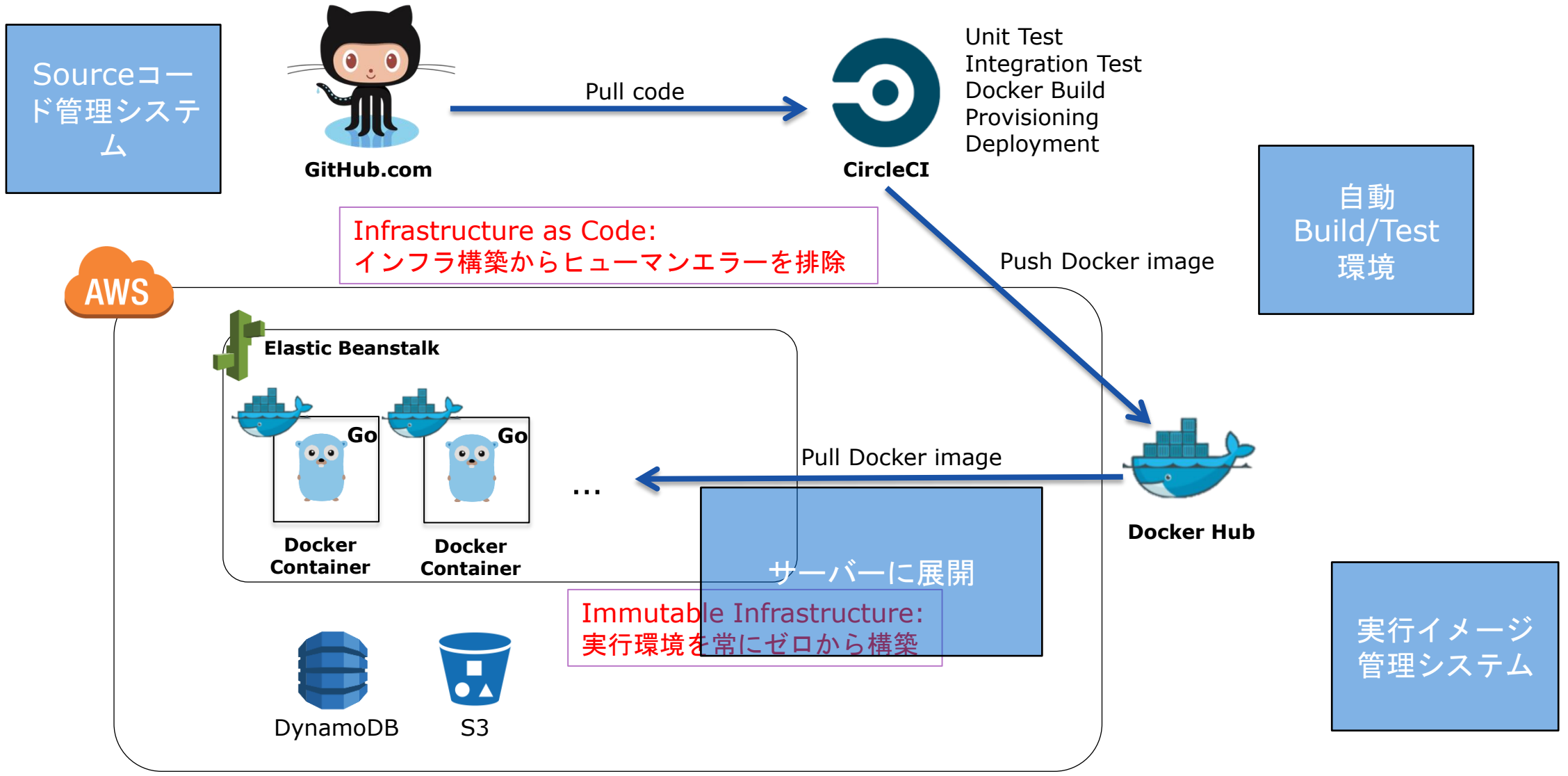
① 最先端SaaS/PaaSの徹底活用

② ローカル環境で開発が完結

③ テスト/品質チェックの徹底

④ 運用の自動化と安定化

① 最先端SaaS/PaaSの徹底活用



②

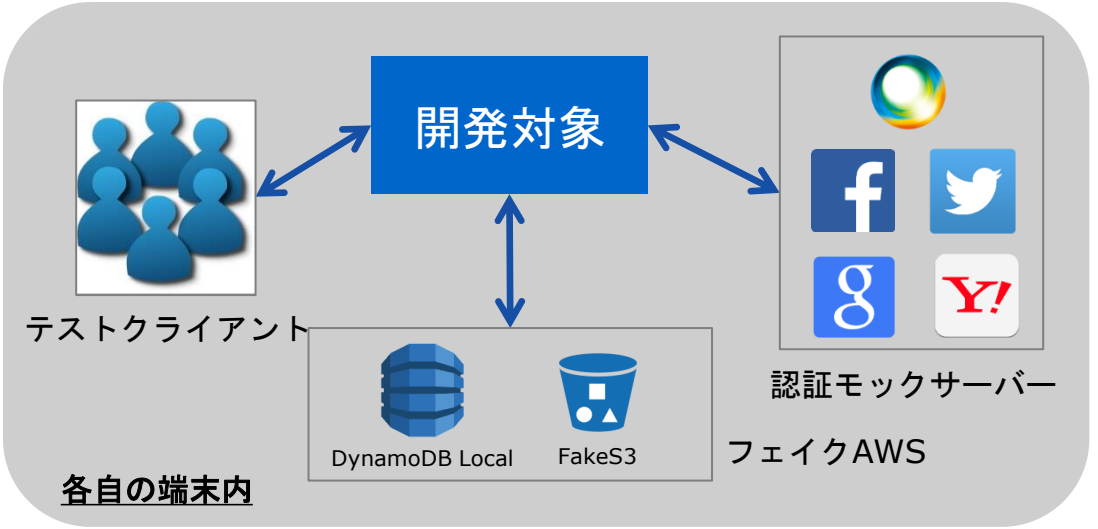
ローカル環境で開発が完結

これまで

- 開発効率が悪い
  - クラウド上の開発環境を共有
  - 実験的な作業をしにくい
  - 待ち時間、初期化等で非効率
- プロダクション環境で問題が出る
  - 開発環境ごとに構成が異なる
  - 開発環境内のシステム更新経緯により、差分が発生
  - 開発環境とプロダクション環境で挙動が異なる

やったこと

- モックサーバでローカルに開発環境を実現



- Infrastructure as codeとして環境をバージョン管理、開発とプロダクション環境の差分がほぼゼロ

③

テスト・品質チェックの徹底

- 急がば回れ
  - テストコード開発、テスト環境開発に総工数の半分以上を投入
  - 自動化の徹底（ユニットテスト、結合テスト、システムテスト、フォーマッタ、リンタ、コードメトリクス、コードカバレッジ, etc.）

- 作り直すことで、テストコード不在という積み上がった問題を解消
  - 開発の当初からテストを自動化して継続（本体コードより多いテストコード）
  - 開発者自身が品質を作り込み、テストにも責任を負う
  - テストも開発者が実施するため、よりテストが容易な設計に

メンテが容易になり、デグレしにくくなった
障害を恐れず、どんどん新機能開発が可能に

④

運用の自動化と安定化

データ保全

- 自動データ整合性チェック
- 自動DBバックアップ

過負荷対策

- 内部サービスに対する流入トラフィックの制御
- 外部サービスに対するサーキットブレーカー
- オートスケール

メンテナンス範囲の局所化

- サービス単位、API単位の閉塞/再開

エスカレーション、運用工数の大幅な減少

結果

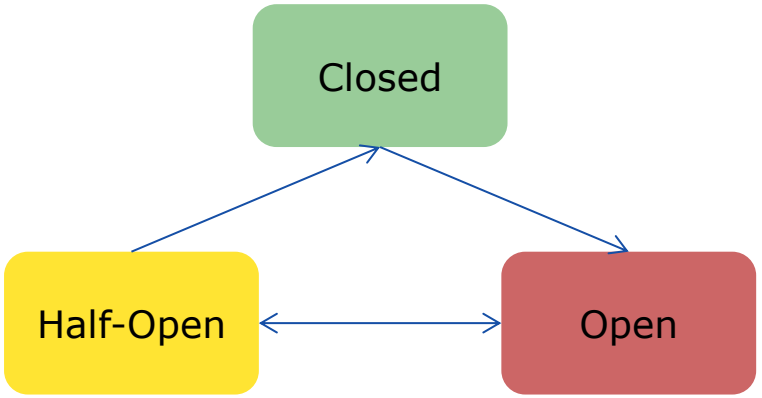
目標	結果
① 最先端SaaS/PaaSの徹底活用	デプロイ時間が 数日から30分に短縮
② ローカル環境で開発が完結	開発が並列化され 開発効率は6倍以上
③ テスト/品質チェックの徹底	10+回のリリースを 経ても停止0
④ 運用の自動化と安定化	更新・メンテ工数1/3 運用工数ほぼ0

ライブラリの公開

Gobreaker

- Circuit breaker patternのGo実装
- 標準的な実装に対して、
  エラーカウンターの世代管理を強化

Circuit breaker pattern



Sonyflake

- 分散型ID発行ライブラリ
- Goで実装
- TwitterのSnowflakeに着想
- AWSのVPCにおけるIPアドレスのユニーク性を利用し、Machine ID発行サーバを不要にしたもの
- Twitterほどのトランザクションがない、一般的なサービスに最適化

Sonyflake ID

39 bits for time in units of 10 msec
8 bits for a sequence number
16 bits for a machine id

github.com/sonyで公開中！

まとめ

手間をかけない

- SaaS/PaaSの利用拡大
- 自動化の追求

急がば回れ

- 開発環境への投資
- 開発ツールの開発

役に立つ

- ライブラリを公開
- Github.com/sony

最後に

ドキュメントが非常に充実していて、助かりました。

ソリューションアーキテクト、営業、**BizDev**、
現AWS社員（元ソニー社員）などの皆様のサポートが手厚く、助かりました。

エンタープライズサポート契約してよかったです。

ご清聴ありがとうございました。