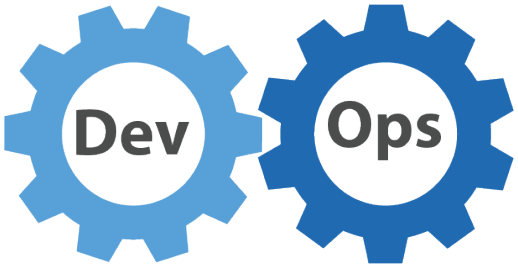


ユニリタのDevOpsに対する取り組み

～Docker使ってみた～

株式会社ユニリタ

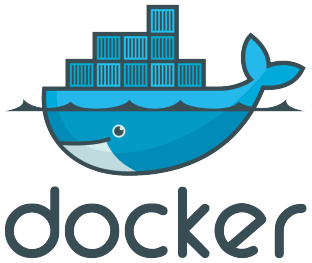
Be.Cloud

真木 卓爾

2016.06.10

Agenda

起(Why)　・・・・7分
　なぜDevOps、なぜDocker？



承(YWT)　・・・・5分
　ソリューション、手法、ツール

転(YWT)　・・・・6分
　理想と現実

結(YWT)　・・・・2分
　今後の取り組み

UNIRITA

起(Why)

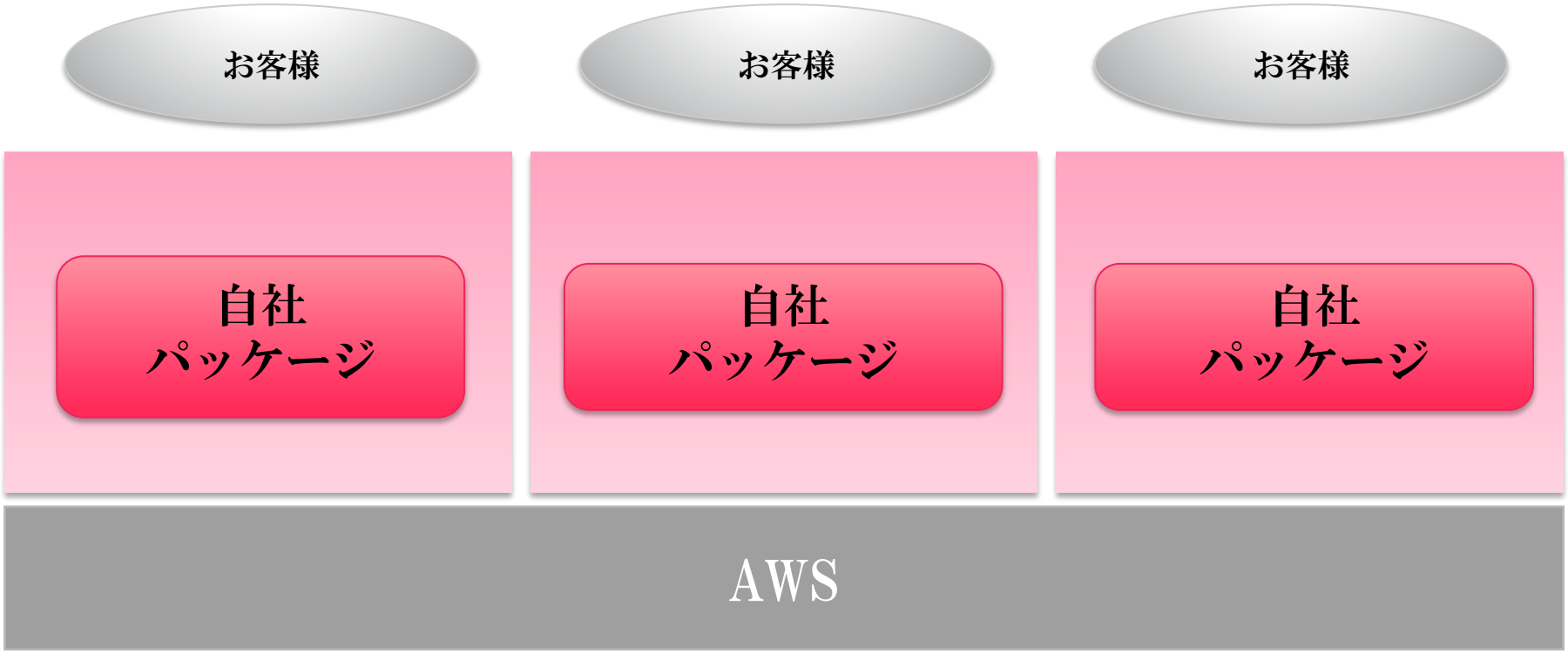
なぜDevOps(Docker)？

ビジネスモデル1

- オンプレミス
- ライセンス、技術支援
- 役務型
- 直販営業が売る

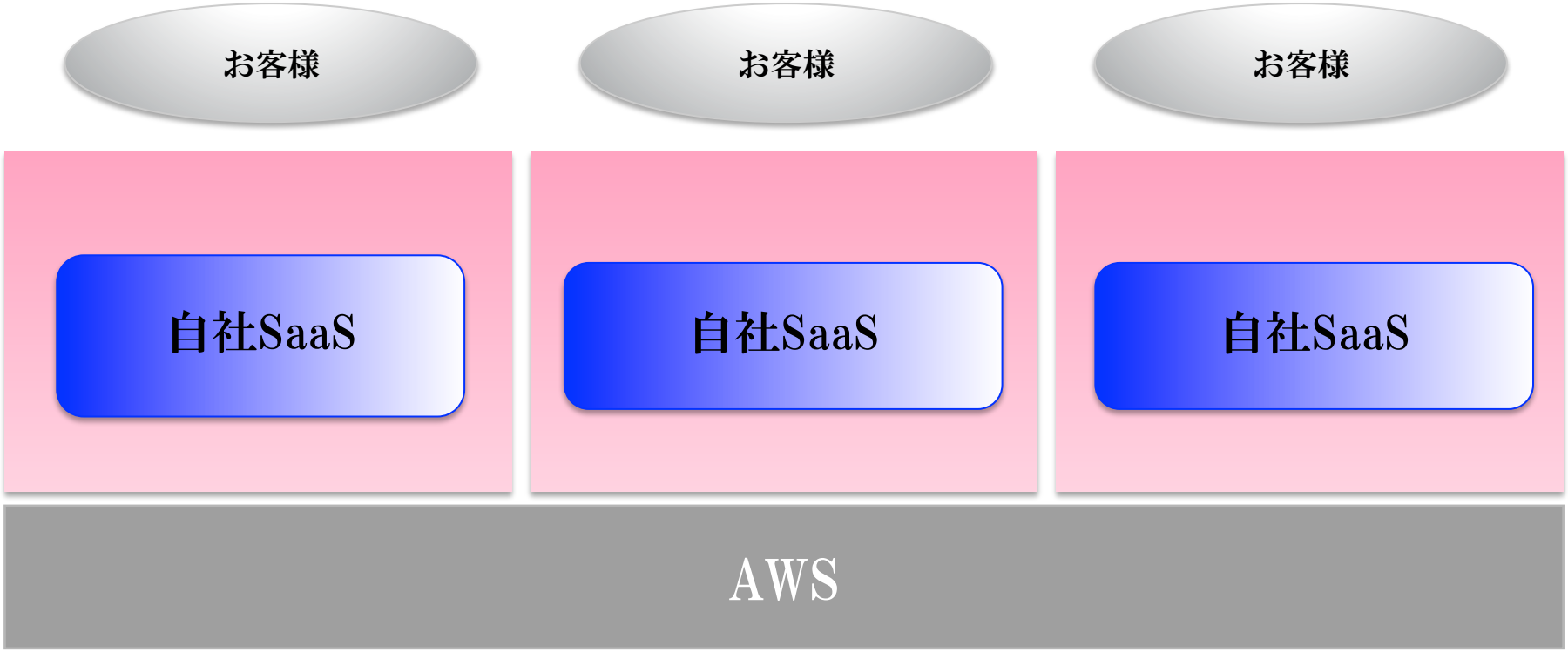
ビジネスモデル2

- マネージドサービス(いちおうクラウド)
- 月額利用料、技術支援
- 役務型
- 直販営業が売る



ビジネスモデル3

- SaaS(いちおうクラウド)
- 月額利用料
- サービス型
- 直販営業が売る



ビジネスモデルまとめ

	ビジネスモデル1	ビジネスモデル2	ビジネスモデル3
インフラは？	オンプレミス	クラウド	クラウド
何でお金をいただく？	ライセンス 技術支援	月額利用料 技術支援	月額利用料
何を売る？	自社パッケージ	自社パッケージ	SaaS
誰が売る？	直販営業	直販営業	直販営業

スケールしない

UNIRITA

スケールするためには？

- すぐに始められる
  →マルチテナント
  →トライアルサイト

- 口コミで広まる(バイラルマーケティング)
  →マーケティング、インサイドセールスが売る
  →事業部はコンテンツを提供する

- 良いサービスがどんどん生み出される
  →他部門や他社との協業モデル

私たちは、これらを実現できる
プラットフォームを持っていない

理想的なスケールサイクル

新ビジネスモデル

	ビジネスモデル1	ビジネスモデル2	ビジネスモデル3	新ビジネスモデル
インフラは？	オンプレミス	クラウド	クラウド	クラウド プラットフォーム
何でお金をいただく？	ライセンス 技術支援	月額利用料 技術支援	月額利用料	月額利用料
何を売る？	自社パッケージ	自社パッケージ	SaaS	自部門SaaS 他部門SaaS 他社SaaS
誰が売る？	直販営業	直販営業	直販営業	マーケティング インサイドセールス 事業部、他社 口コミ

CloudGearコンセプト

Be.Cloudサービス	ユニリタサービス	協業サービス
自部門SaaS	他部門SaaS	他社SaaS
自部門SaaS	他部門SaaS	他社SaaS

CloudGear

サインアップ/イン、課金、SSO、ID管理、モバイル、セキュリティ、パーソナライズ

AWS (Elastic Container Service)

CloudGear
アカウント

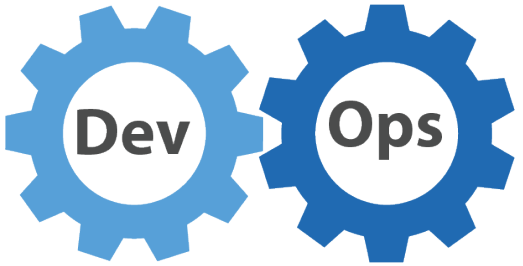
どうやって作るか？

・１年後にできました、はいリリースします
　・・・ではもう遅い

市場投入までの時間をいかに短くするか？

・サービスは先に出来上がっていく
　運用しながら開発しなければならない

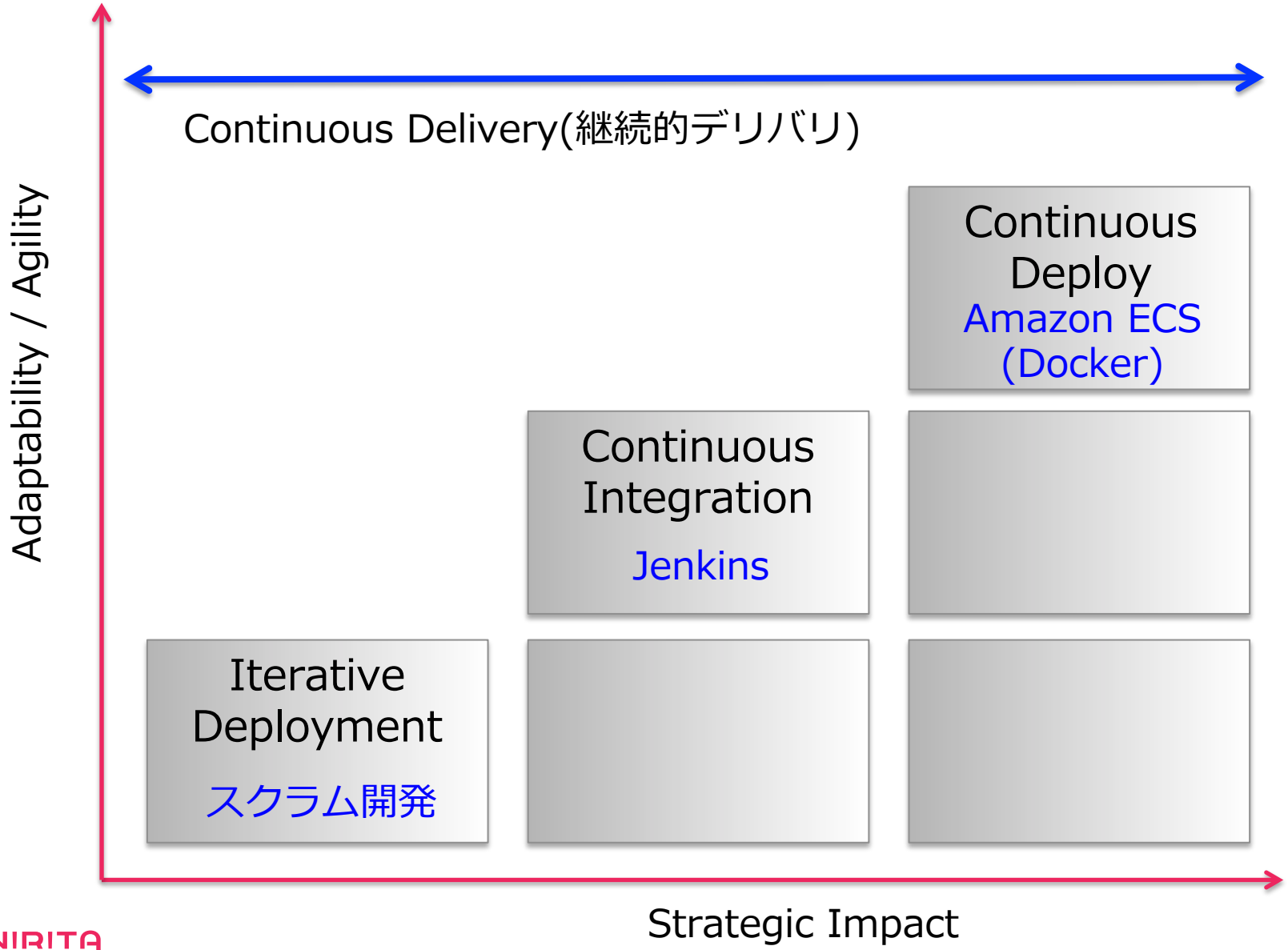
運用しながら、プラットフォームの変更を
いかにスピーディに反映させ、
いかに差別化するか



承(YWT)

ソリューション、手法、ツール

ソリューション



手法(スクラム開発)

- ２週間に１度のスプリントレビューを実施

プロダクトオーナー
→プロダクトバックログに責任を持つ
→機能や改善の優先度を決める
→誰のために、何のために、それが必要なのか、の合意と納得を伝える

スクラムマスター(優秀なメンバー)
→スプリントバックログに責任を持つ

スクラムメンバー(優秀なメンバー２名)
→デイリースクラムを実施

手法(スクラム開発)

スプリントスケジュール

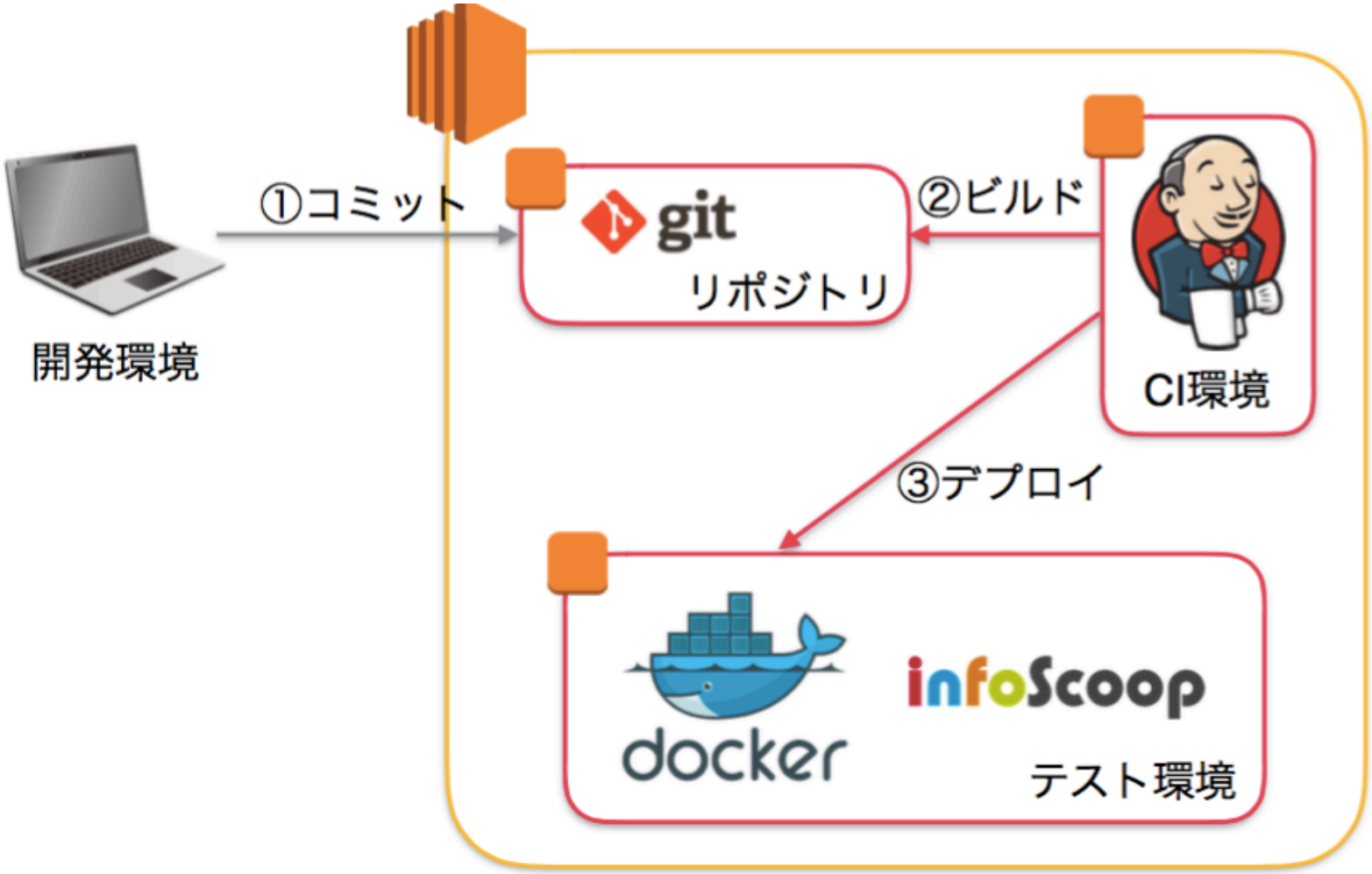
実作業時間は、「（スプリント日数）ー3日」しかない！

ツール(Jenkins)

・自動コンパイル、自動テストを実現

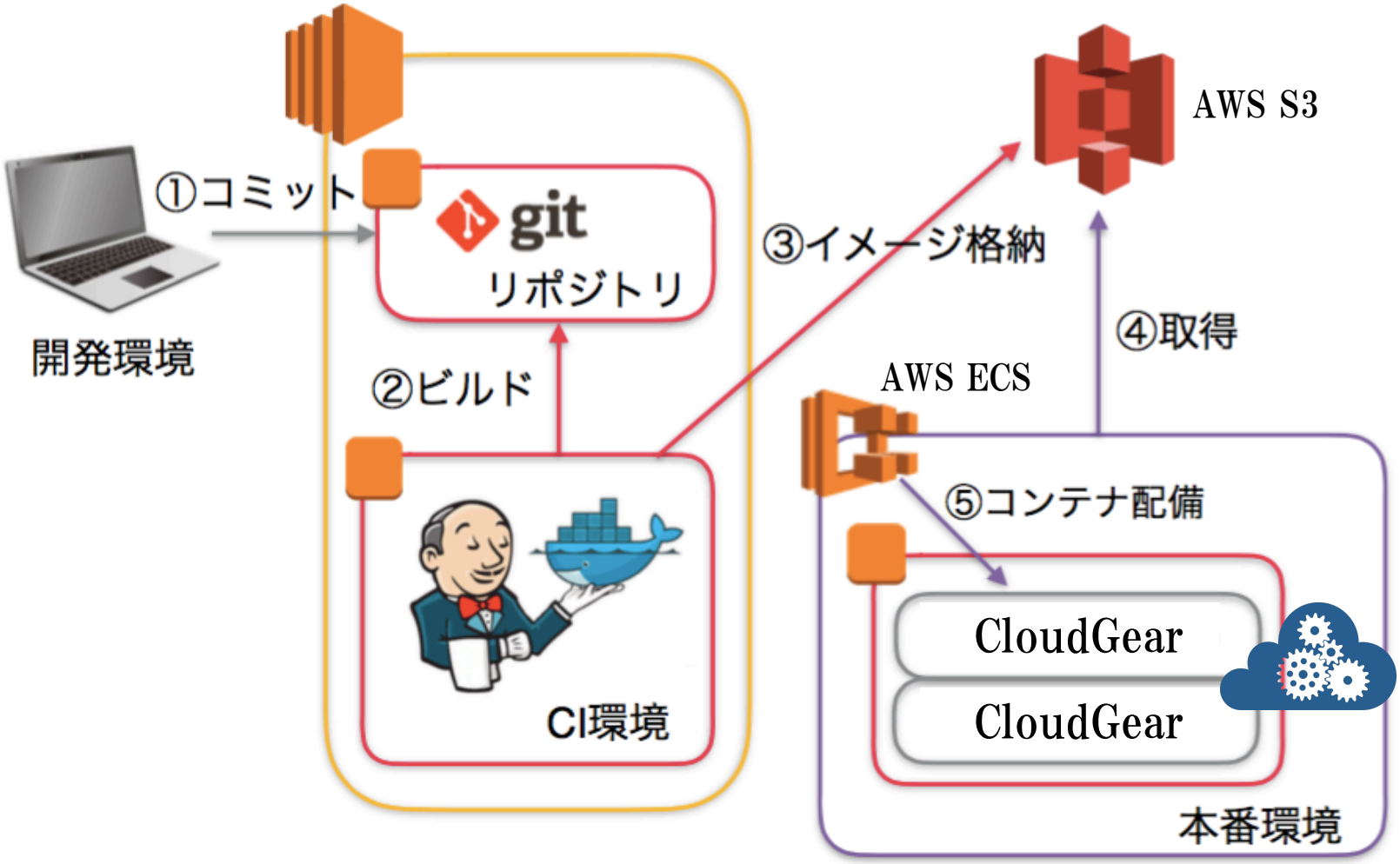
ツール(Docker)

・テスト環境



ツール(AWS ECS)

・CloudGear本番環境

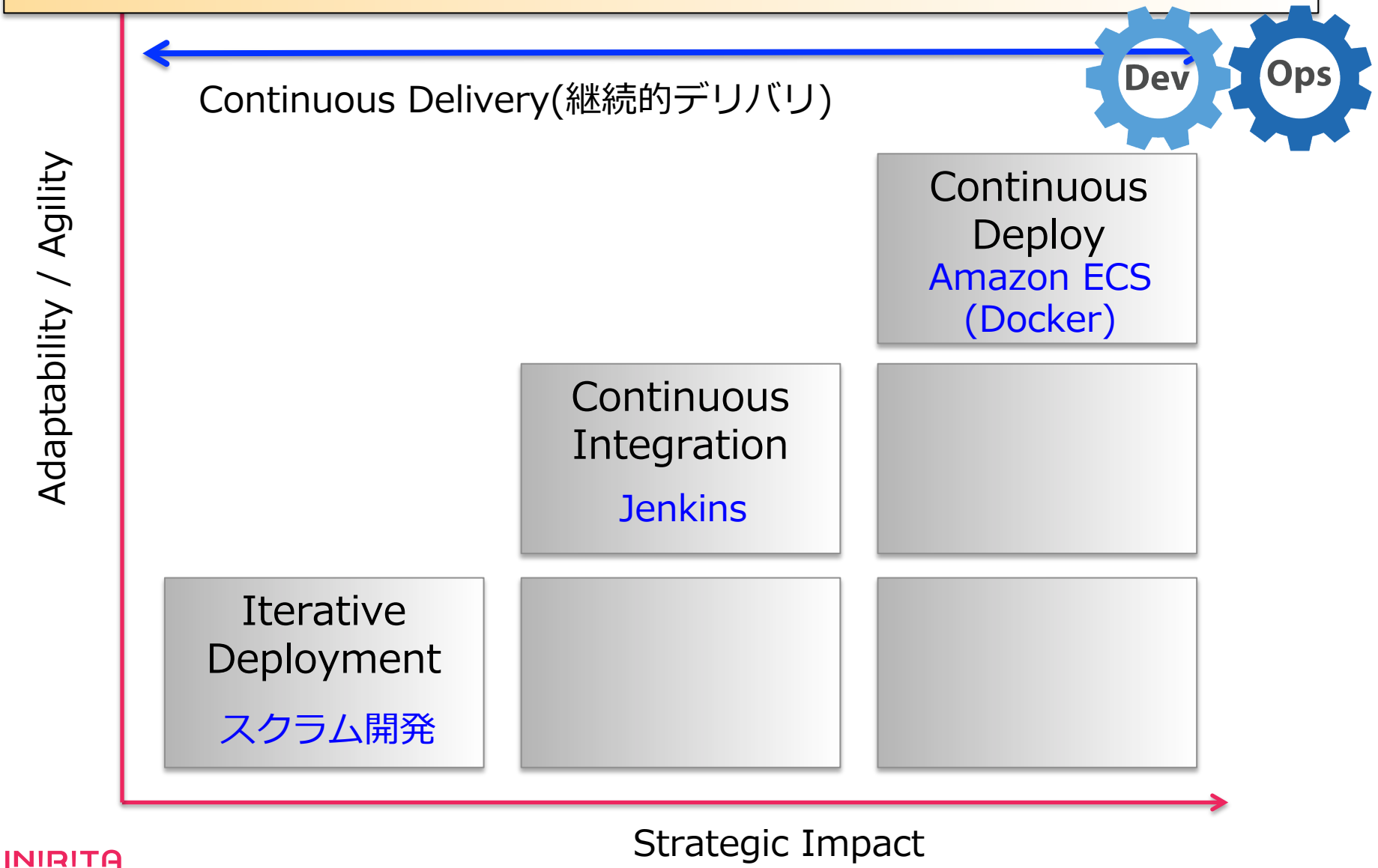


DockerとAWS ECSの関係

- AWS ECSはDockerをラップしたAWS EC2の機能
- DockerコンテナのEC2インスタンスへの配備

- インスタンス内ではDockerクライアントが動作している。
- ECSコンソールからの操作は、インスタンス内のecs-agentコンテナを介して行われる。

・サービスの市場投入までの時間をいかに短くするか
・サービスの変更をスピーディに反映させ、いかに差別化するか



転(YWT)

理想と現実

AWS ECSのメリット、デメリット

【メリット】

・少ない工数で本番移行可能
・ECS管理によりオーケストレーションを簡易化
・10秒程度で本番リリース可能
・ポータビリティが良い(スケーラビリティより)

【デメリット・・・というか手間】

・EC2インスタンスを終了した際、自動的にECSクラスタから外れない
・ECSありきの設計を考えておかないと難易度高

理想と現実

- 勘違いをとく
- 円滑な運営のために
- チーム管理

理想と現実

- 勘違いをとく
- 円滑な運営のために
- チーム管理

勘違いをとく

スクラム開発は

WFで起きうる問題を解決できる

解決しません

勘違いをとく

スクラム開発は

見積り精度が良くなる

良くなりません

多少のブレがあっても問題ないだけ

勘違いをとく

スクラム開発は

仕様変更に強い

強くありません

プロダクトオーナーの要求との乖離が少なくなるだけ

勘違いをとく

スクラム開発は

追加要求にすぐ対応できる

できません

トレードオフという概念があるだけ

理想と現実

- 勘違いをとく
- 円滑な運営のために
- チーム管理

円滑な運営のために

スクラム開発では、プロダクトオーナーと**必ず**約束しなければならないことがある

約束＝合意と納得

1. スプリント内で実現できること
2. トレードオフの概念
3. ベロシティに気をつける
4. ユーザーストーリーと成果物

これらの約束が守れない(守ってくれない)場合、スクラム開発は必ず失敗する

UNIRITA

円滑な運営のために

1. スプリント内で実現できることの約束

・必要なのはガントチャート(進捗管理)ではない
ひとつのスプリントで「何が」できるか
それ以上でもそれ以下でもない

2. トレードオフの概念の約束

・あれをやるならこれはやらない
・追加要求は２スプリント後に再度検討すべき

時間が経つとプロダクトオーナーは往々にしてその要求を忘れている

円滑な運営のために

すごいダメな例

3. ベロシティに気をつけるの約束

・頑張った分だけベロシティは高い
　頑張るほどにチームの首を絞める
・消化不可能な量のユーザーストーリー
　を持ち込まないこと

4. ユーザーストーリーと成果物の約束

・成果物は動くものであり、それがすべて
・成果物に不満があるなら、それはユーザー
　ストーリーの約束ができていなかったということ

円滑な運営のために

プロダクトオーナーとスクラムチームの約束

1. スプリント内で実現できること
2. トレードオフの概念
3. ベロシティに気をつける
4. ユーザーストーリーと成果物

私たちは勘違いにうすうす気づきながらも、
少しずつ約束を守れるようになってきた

つまり、ペースをつかんできた

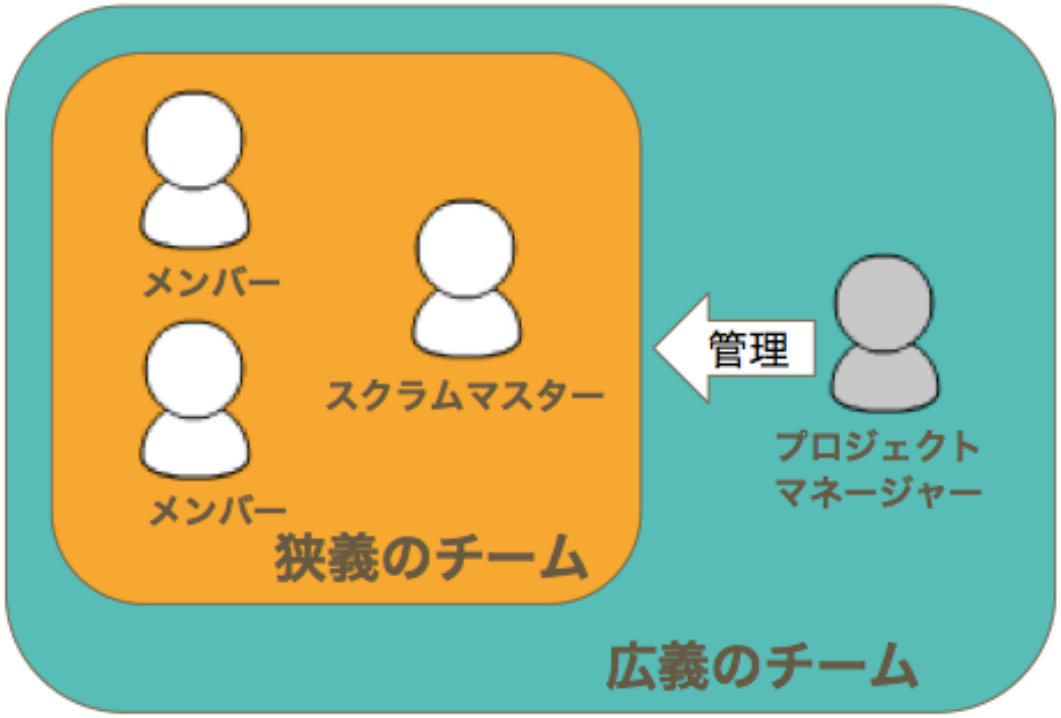
UNIRITA

理想と現実

- 勘違いをとく
- 円滑な運営のために
- チーム管理

チーム管理

理想的なチーム構成



プロダクトオーナー

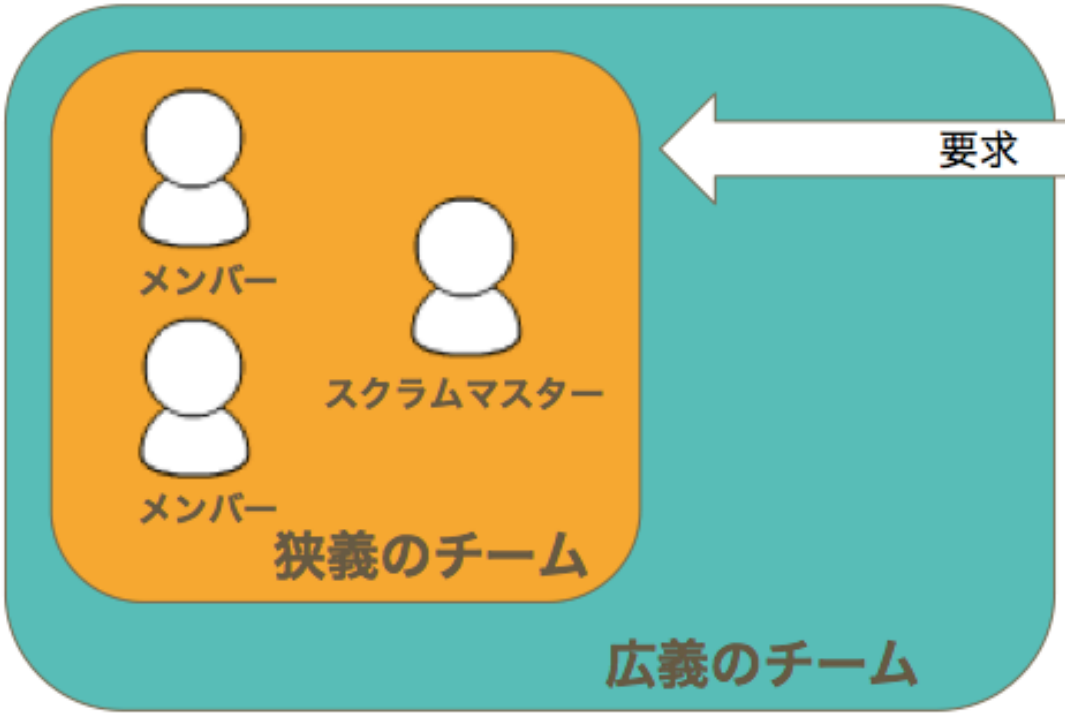
ＰＯ：要求者。成果物の受け入れ条件を決める。

ＰＭ：スケジューリング、バックログ管理を行う。**実装は行わない。**ＰＯの要求を遵守するよう仕向ける。

ＳＭ：活動の円滑化を図る。チームの生存を最優先する。要求を棄却できる。

チーム管理

実際のチーム構成



プロダクトオーナー

マネージメントする役割がいない！！

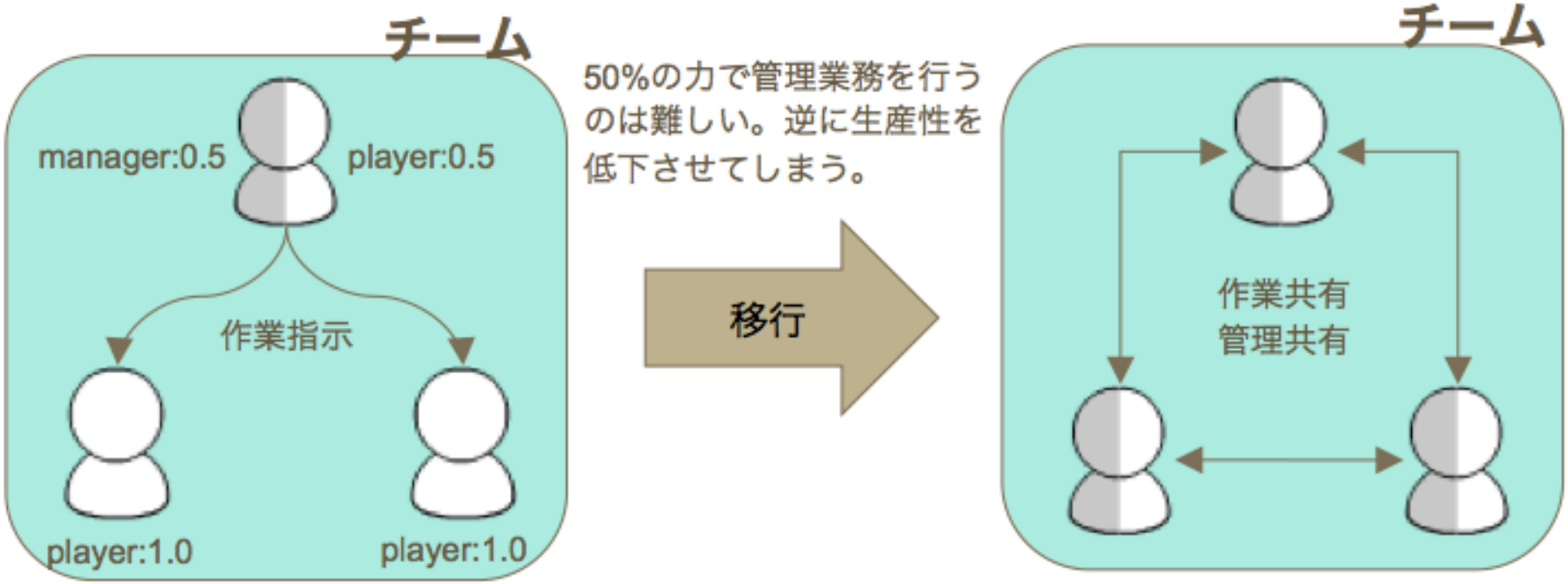
ＰＯ：要求者。成果物の受け入れ条件を決める。

ＰＭ：スケジューリング、バックログ管理を行う。**実装は行わない。**ＰＯの要求を遵守するよう仕向ける。

ＳＭ：活動の円滑化を図る。チームの生存を最優先する。要求を棄却できる。

チーム管理

全員がプレイングマネージャー



個人のプレイング能力を最大限に活かすため、
開発作業だけでなく管理業務も共有する。
進捗は個々の責任のもと監視しなければならない。

UNIRITA

チーム管理

ヒエラルキー型管理からホラクラシー型管理へ

活動の円滑化とプロダクトオーナーとの約束の両方を死守するため、ホラクラシー型管理を模索し、互いが互いを管理する手法を導入。

初期の管理手法

移行

新しい管理手法

理想と現実

勘違いにうすうす気づきながらも、プロダクトオーナーとスクラムチームがお互いに、少しずつ約束(合意と納得)を守れるようになってきた

スクラム開発のペースをつかんできた

さらに、ホラクラシー型管理

次はペース配分をどう高めるかの議論が始まる

結(YWT)

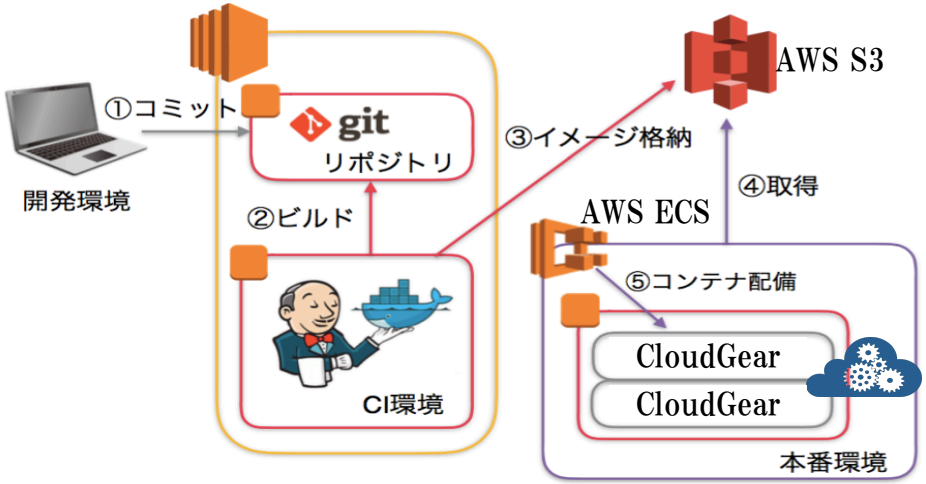
今後の取り組み

今後の取り組み

- できる限り手動を自動に

AWS Lambda

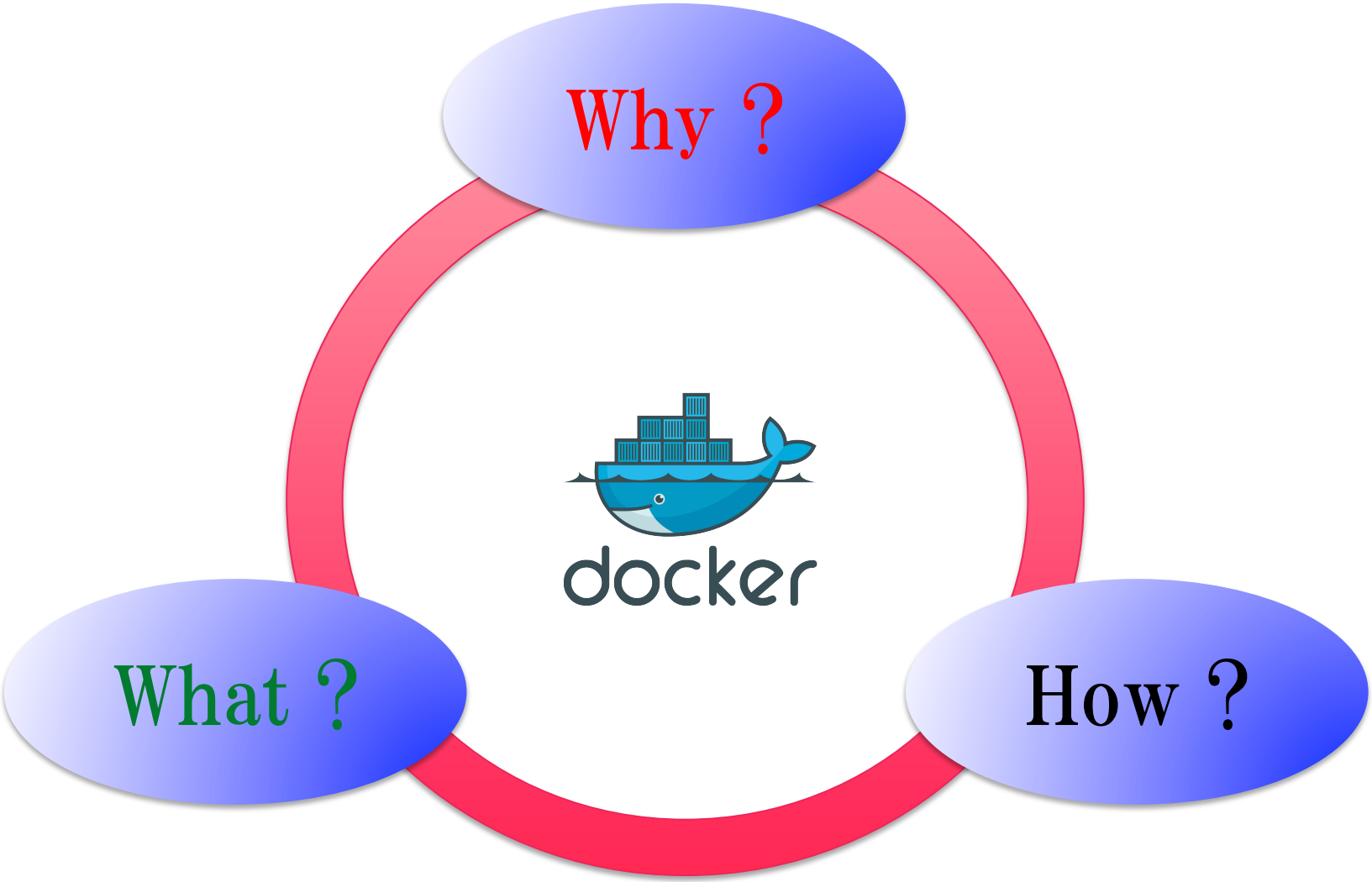
- ECSのスケールアップ/アウト



自分たちがどれだけやれるかのペースが分かってきた
↓
ペース配分をどう高めるかの議論が始まった

今後の取り組み

最後に・・・



ご清聴、ありがとうございました

UNIRITA

ユニリタのDevOpsに対する取り組み

～Docker使ってみた～