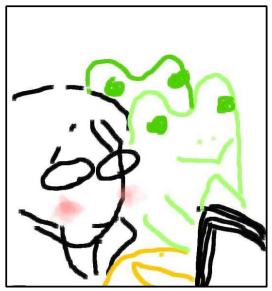
公開版

SIerにおけるKubernetes活用

株式会社スタイルズ
矢野　哲朗

2018年11月7日

自己紹介



矢野　哲朗

tetsurow.yano

株式会社スタイルズ

□経歴　：システム運用 10年・ネットワーク 6年・SI 8年
　近頃はRancher/Nextcloudを担当

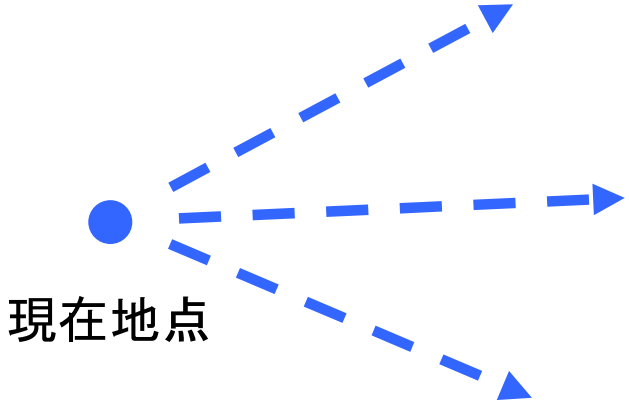
□Rancherの好きな機能：PROJECT LONGHORN

□その他　：全く上達しないRubyist
　一番最初のPCは、OKI if-800 でした…。

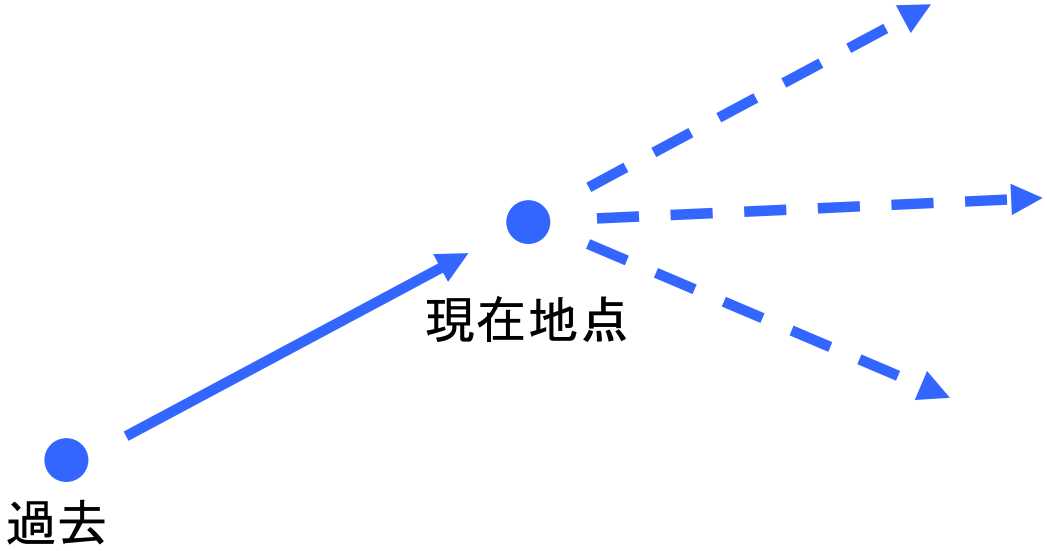
今回説明すること

現在地点

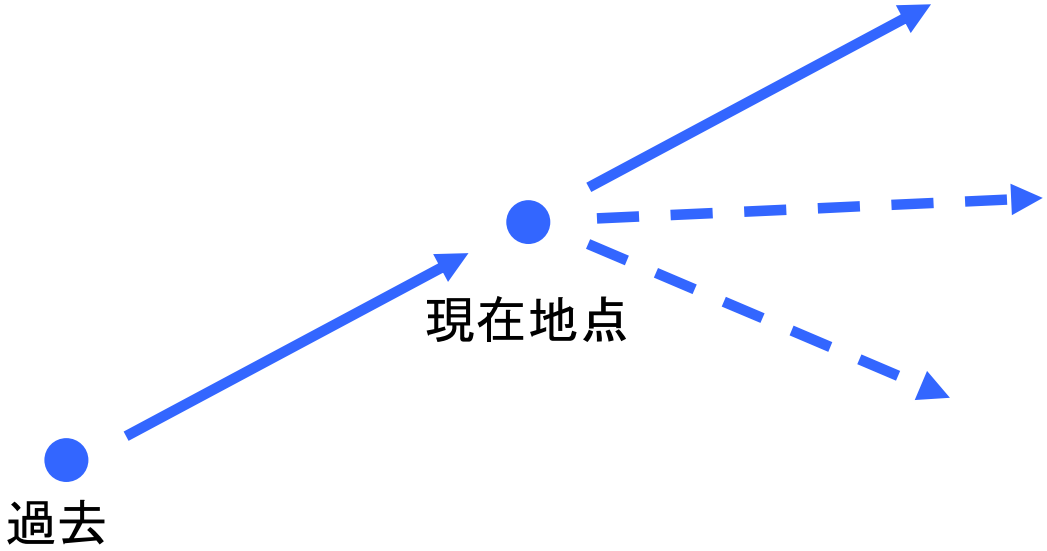
未来を知る



過去を知る



過去を知る＝未来を知る



本日のお話の構成

- コンテナーのメリット
- Kubernetesとはなんなのか
- KubernetesはGoogleがAWSに仕掛けたゲームチェンジ
- Kubernetesによって変わるSIerの役割

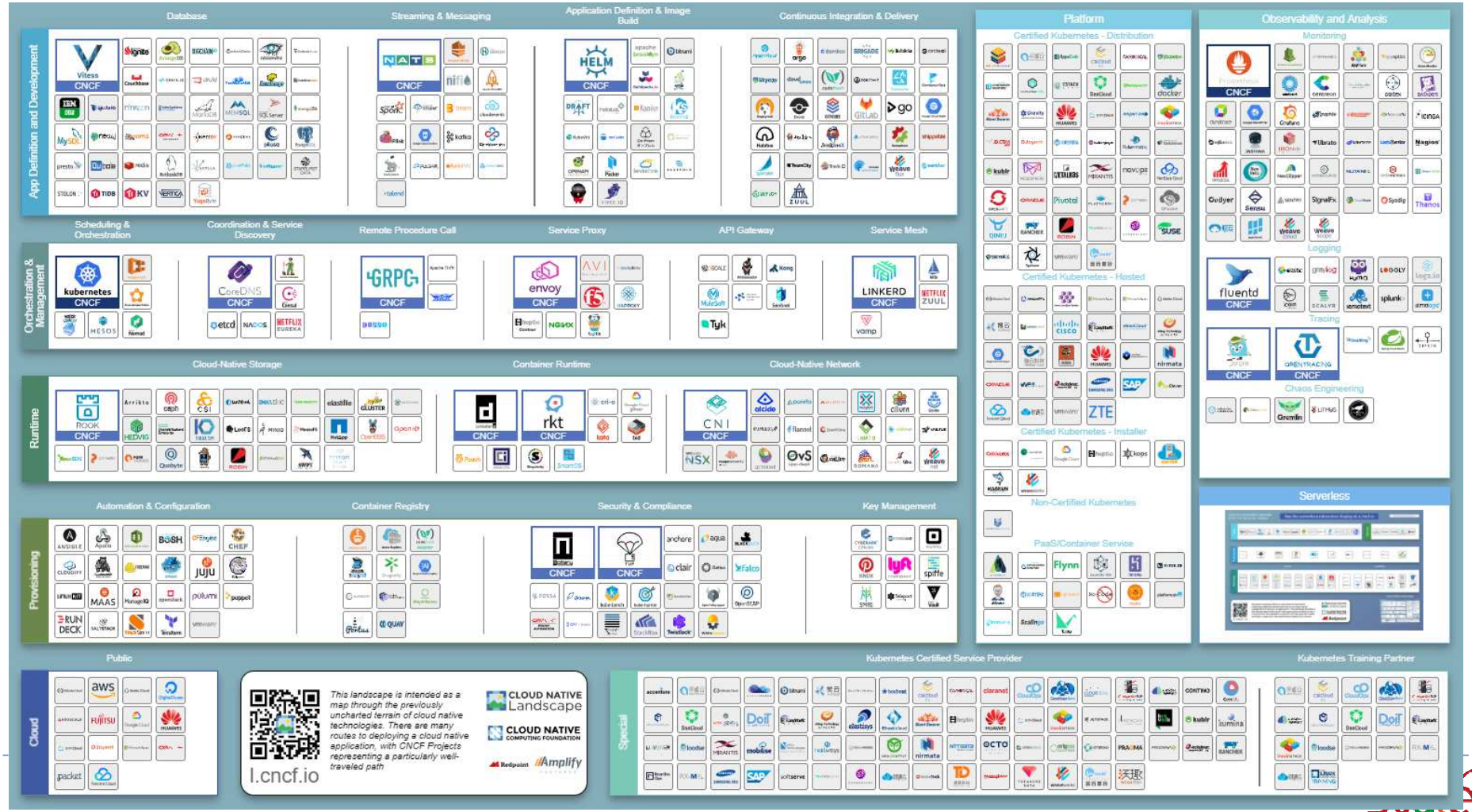
コンテナーといえば、Dockerですよね

Dockerではダメなんですか？

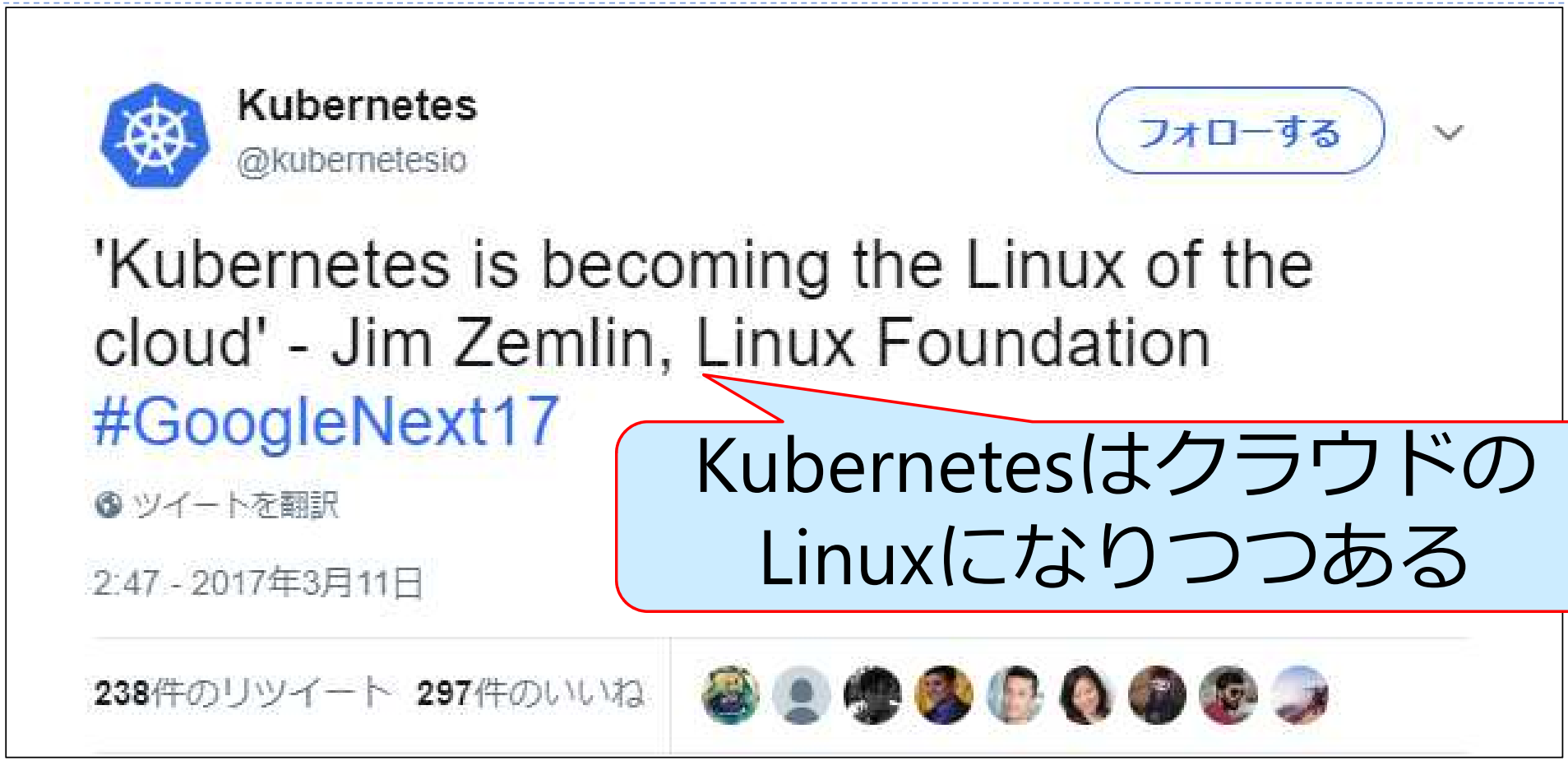
Dockerもダメじゃありません

しかし、海外では
Kubernetesを中心とした
エコシステム
が既にできあがっています

世界でKubernetesを前提として開発中 2018年11月現在



Kubernetesは



Dockerもダメじゃありませんが

この流れを変えるのは無理

Kubernetesを勉強したい人のための書籍

Kubernetes完全ガイド

https://book.impress.co.jp/books/1118101055

Docker/Kubernetes 実践コンテナ開発入門

https://gihyo.jp/book/2018/978-4-297-10033-9

コンテナ・ベース・オーケストレーション

https://www.shoeisha.co.jp/book/detail/9784798155371

Kubernetesは黒船

コンテナーというと

Safmarine【】　　MSKU 959829(1)
(愛媛/今治港FT)　07,10,01　(タイプ45G1)
片妻一方開　　40ft背高型、ドライコンテナ

ドッカーというと

docker

港の労働者

クーバネーティスはなんでしょう

kubernetes

Kubernetesはギリシャ語

Entry | Discussion | Citations

κυβερνήτης

Ancient Greek [edit]

Etymology [edit]

κῠβερνάω (*kubernáō*, "to steer") + -της (*-tēs*)

Pronunciation [edit]

- IPA(key): /ky.ber.nɛ̌ː.tɛːs/ → /ky.βer'ni.

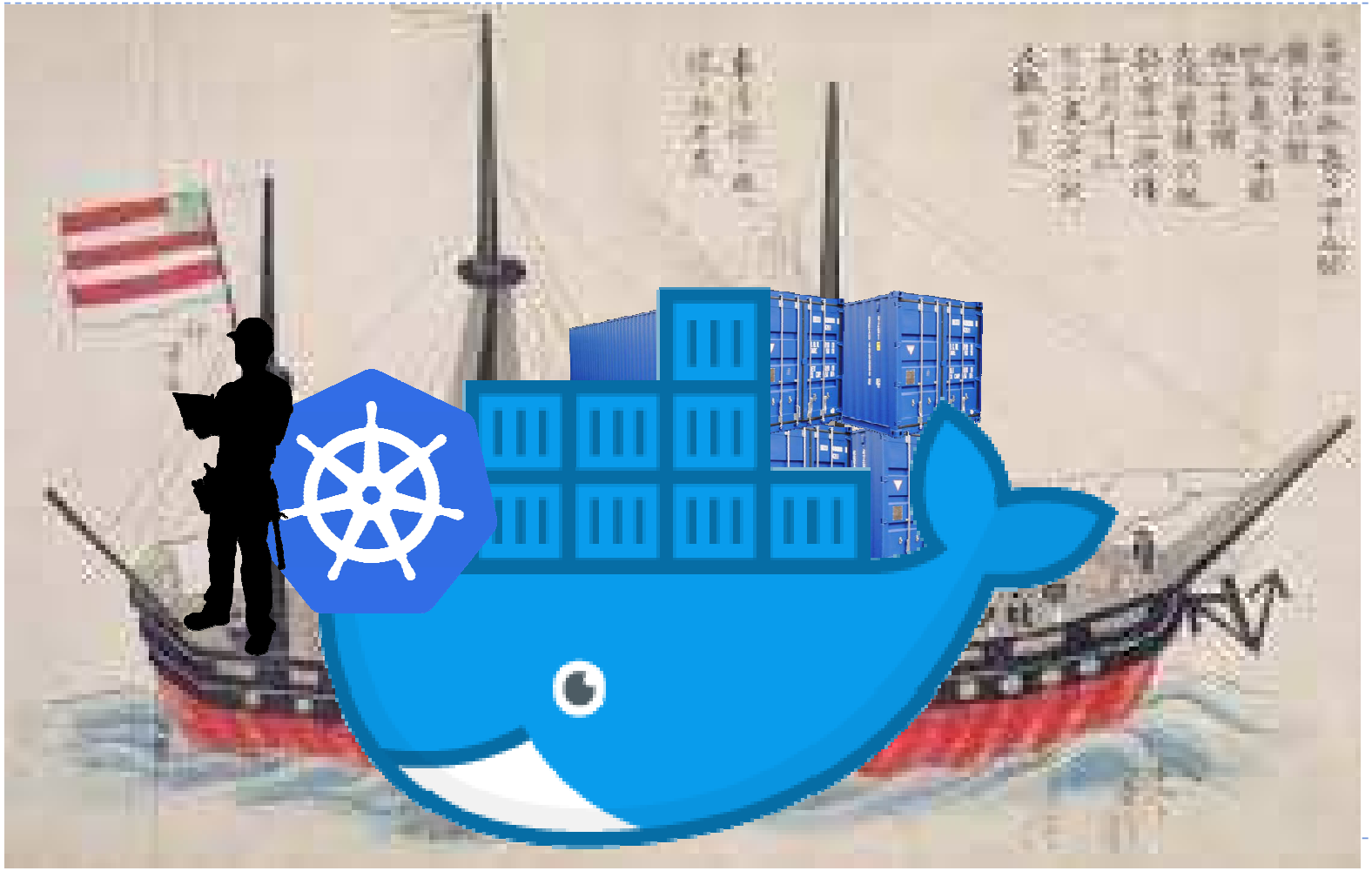
Noun [edit]

κῠβερνήτης • (kubernē´tēs) *m* (*genitiv* **κυβερνήτου**); *first declension* (*Epic*, *Attic*, *Ionic*, *Koine*)

1. a captain, a steersman, a pilot, a navigator
2. (*figuratively*) a guide

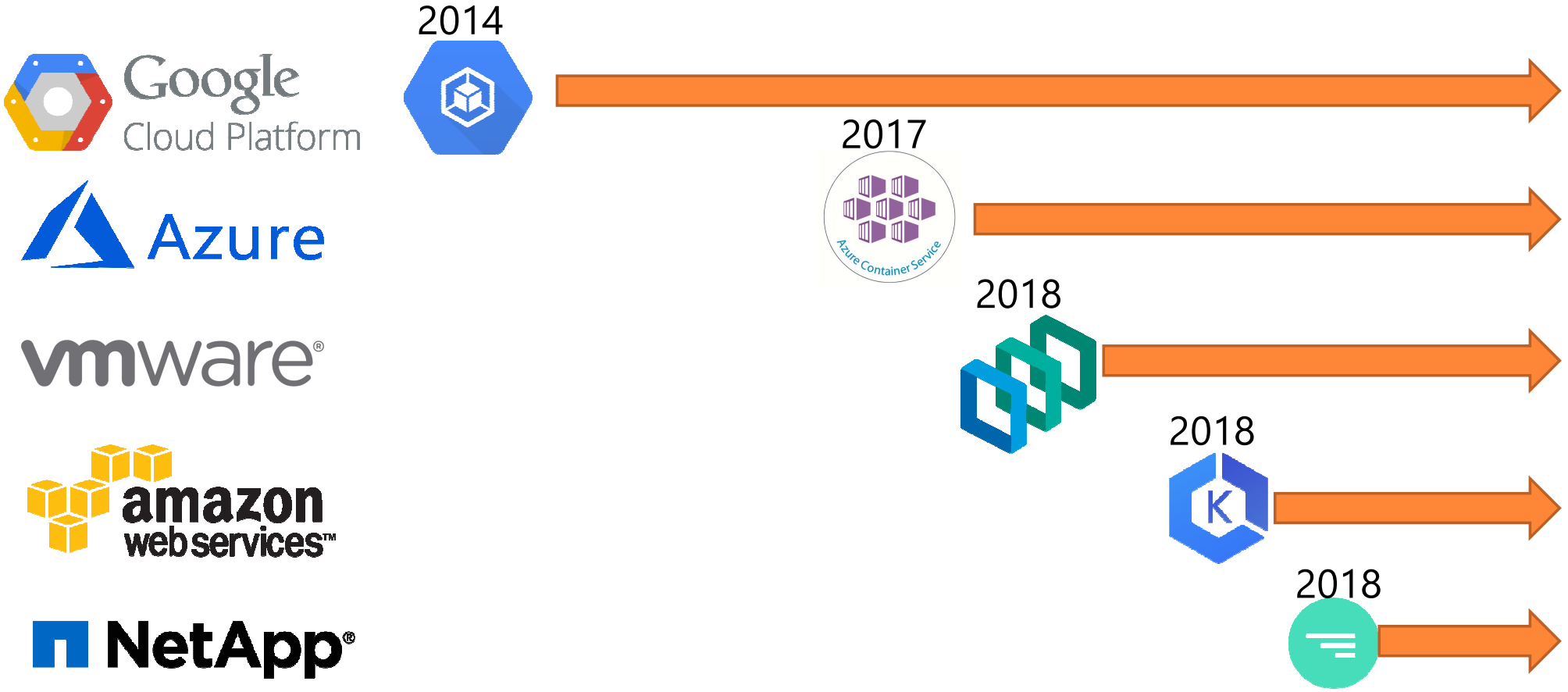
船長、総舵手、パイロット、航海手、ガイド

Kubernetesは黒船



押し寄せる“KaaS” の波

kubernetes as a Service



Stylez3

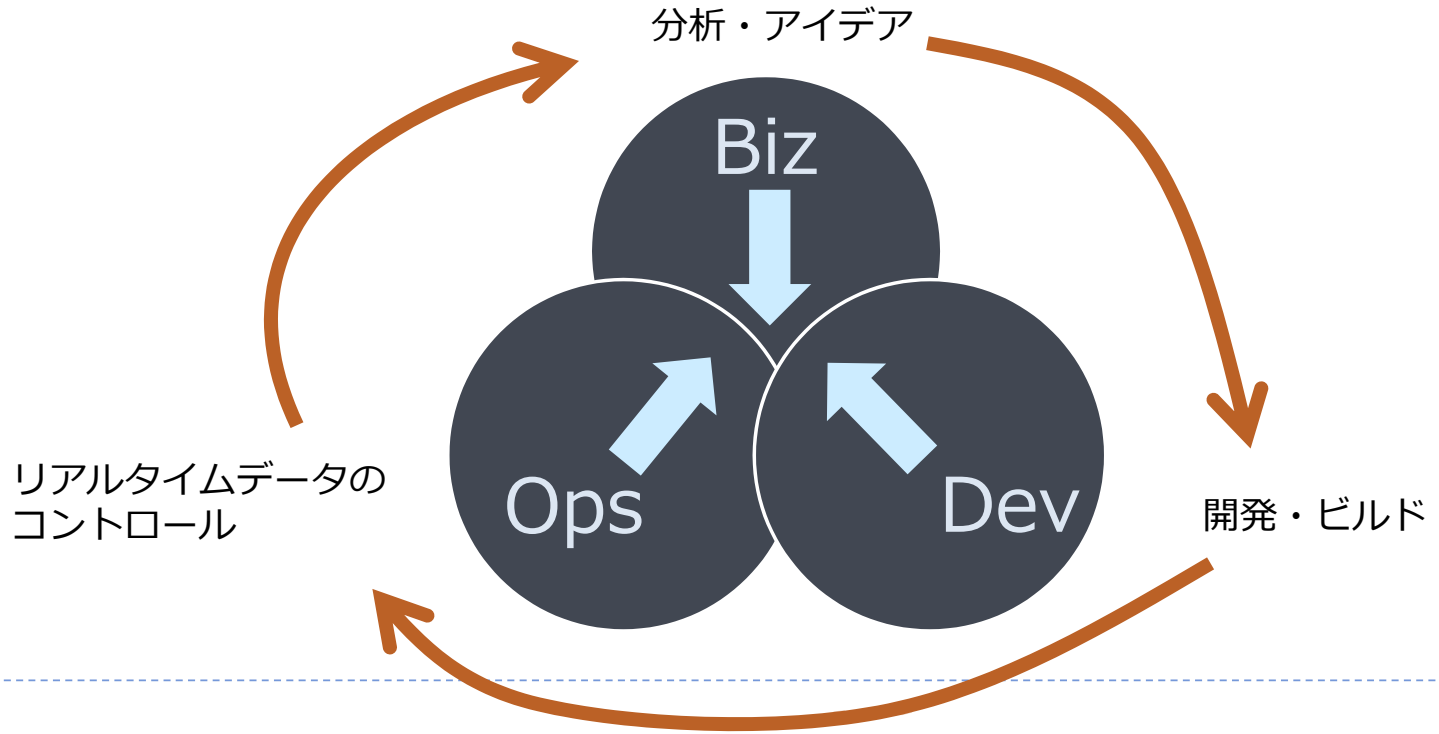
コンテナーのメリット

本日のお話の構成

- コンテナーのメリット
- Kubernetesとはなんなのか
- KubernetesはGoogleがAWSに仕掛けたゲームチェンジ
- Kubernetesによって変わるSIerの役割

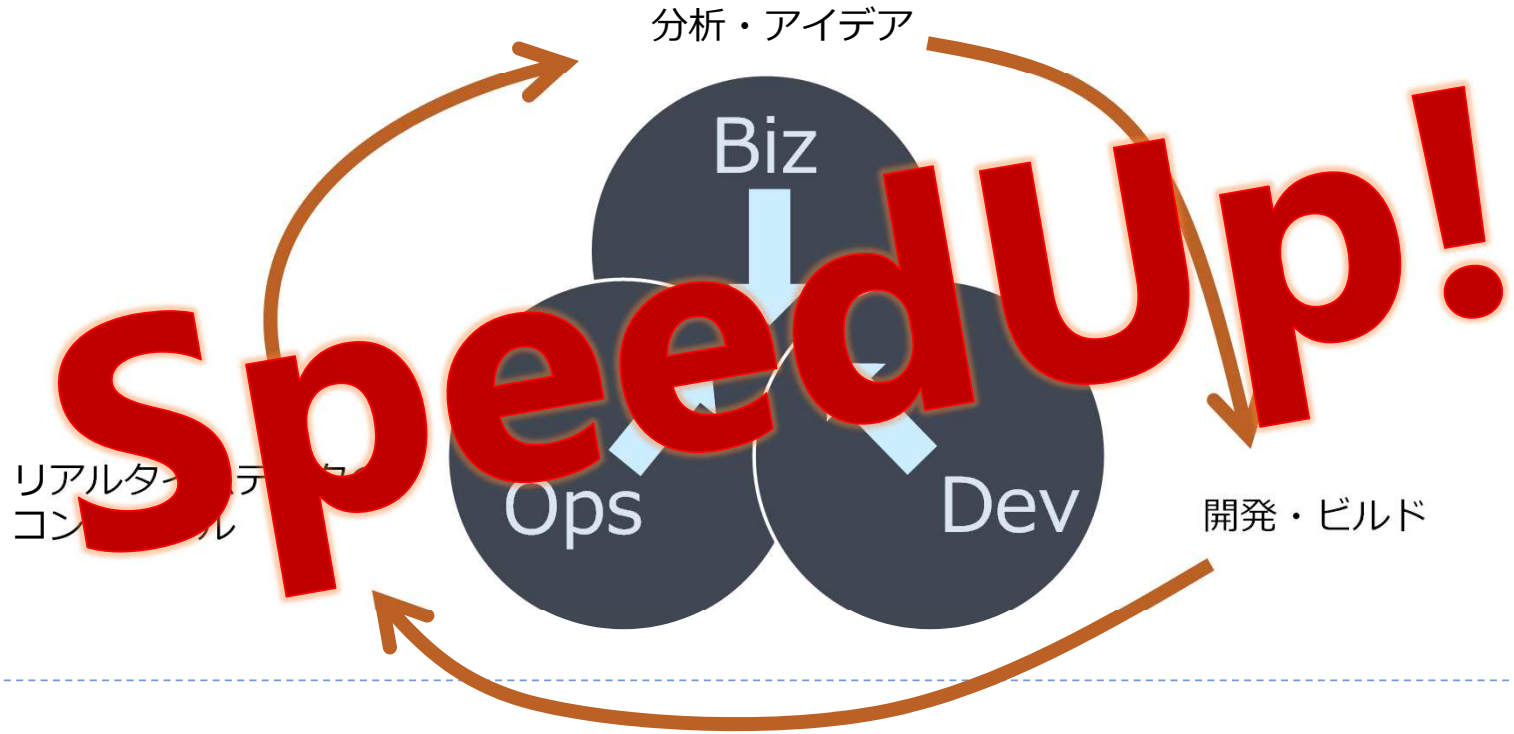
デジタル時代に求められるビジネスモデル

- リアルタイム性のある情報をコントロール
- それに対応した機能のリリースによるユーザ体験の徹底的な訴求
- Biz+Dev+Opsが連携
- スピード感のある絶え間ないシステムの変更への柔軟な対応



デジタル時代に求められるビジネスモデル

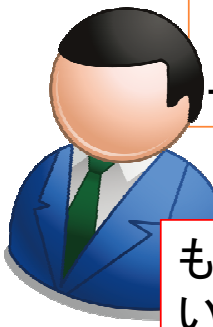
- リアルタイム性のある情報をコントロール
- それに対応した機能のリリースによるユーザ体験の徹底的な訴求
- Biz+Dev+Opsが連携
- スピード感のある絶え間ないシステムの変更への柔軟な対応



よく聞かれるコンテナー利用の声

□ 経営層の声
・コンテナー何それ？
・そんなの無くてもいいいよね。
・何のメリットがあるか分からない。
・DevOpsなんて必要ないでしょ。
・コスト下がるの？

□ 開発現場の声
・コンテナーすごく便利もっと使いたい。
・属人化も避けられるし、リリースに関する秘伝のタレを避けられる。
・アプリケーションのライブラリーミドルウェアの依存関係によるアプリのデプロイの不具合を避けられる。
・自分の環境で動いたアプリが他の人の環境でうごかないという問題がなくなる。
・テスターさんにテストしてもらう環境を簡単に渡すことができる。



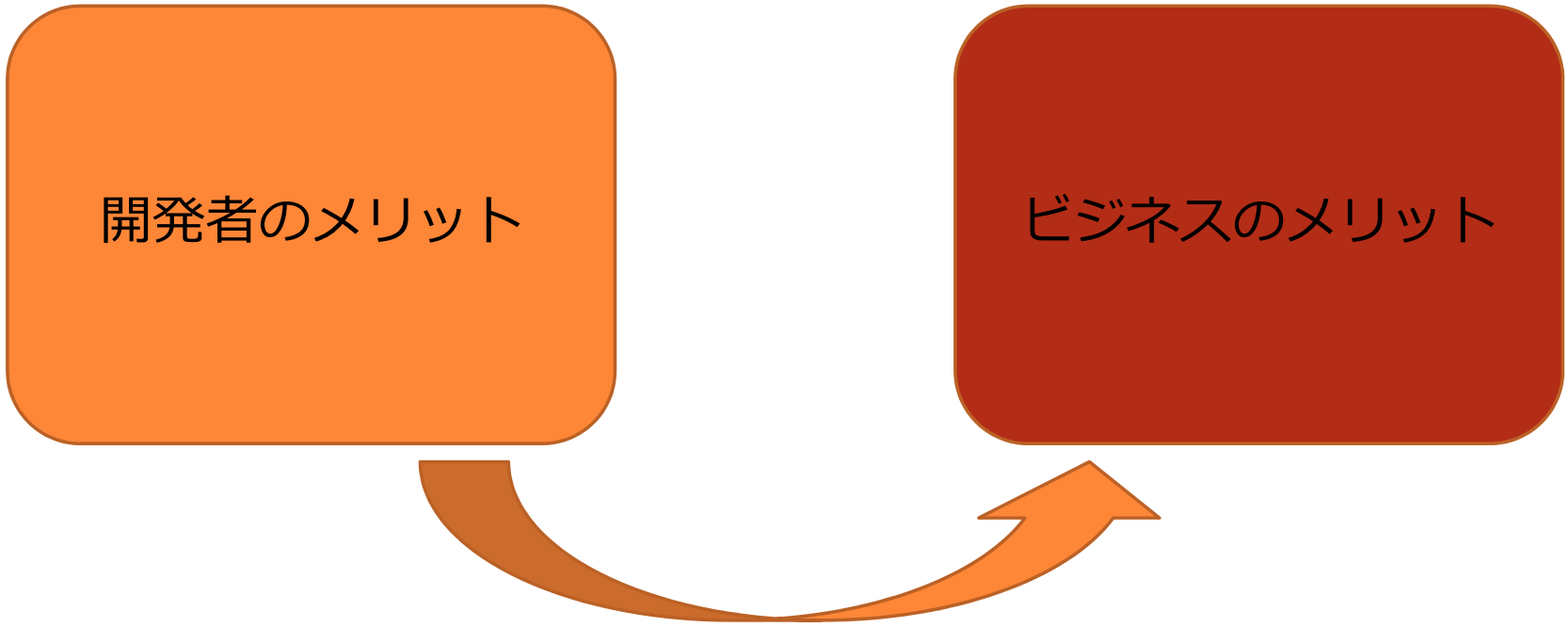
激しい認識のギャップ

もう正直、コンテナー環境以外では作業したくないのになんで分かってくれないの？？？？

コンテナー (Docker) のメリット

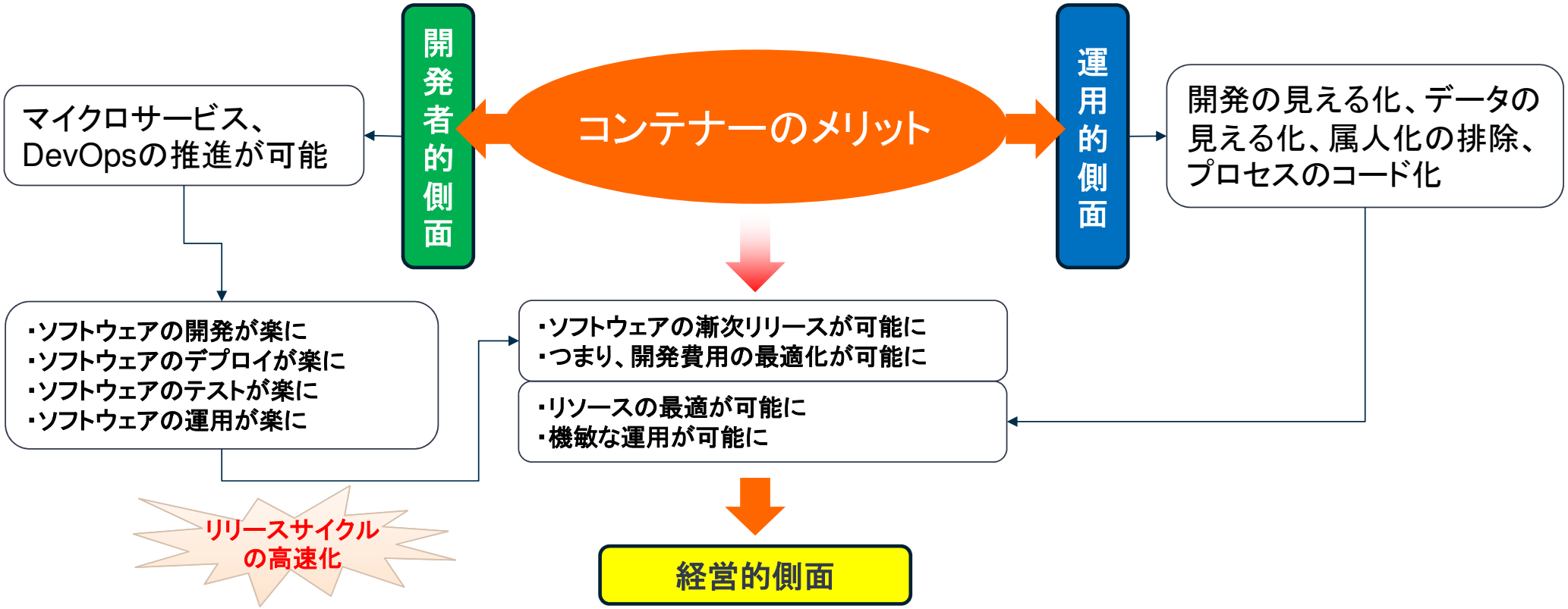
元々は開発者が楽になるようにと考えられたものでした

ビジネス面でのメリットも同様に生かすことができる



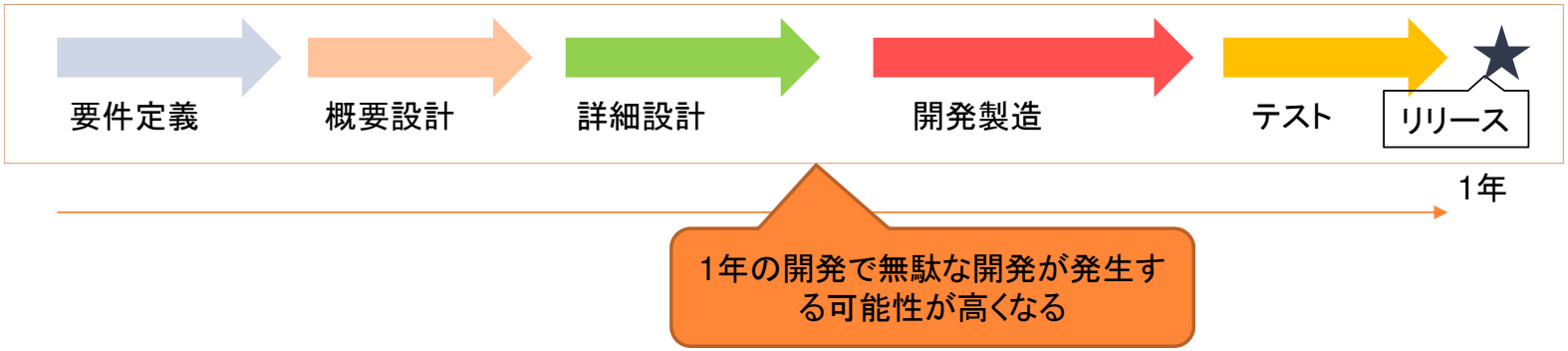
コンテナーは利益に直結するわけではありません

コンテナーには経営的な利益に直結するものではありません。コンテナーには多面的なメリットがありそれによって経営的なメリットが生まれます。



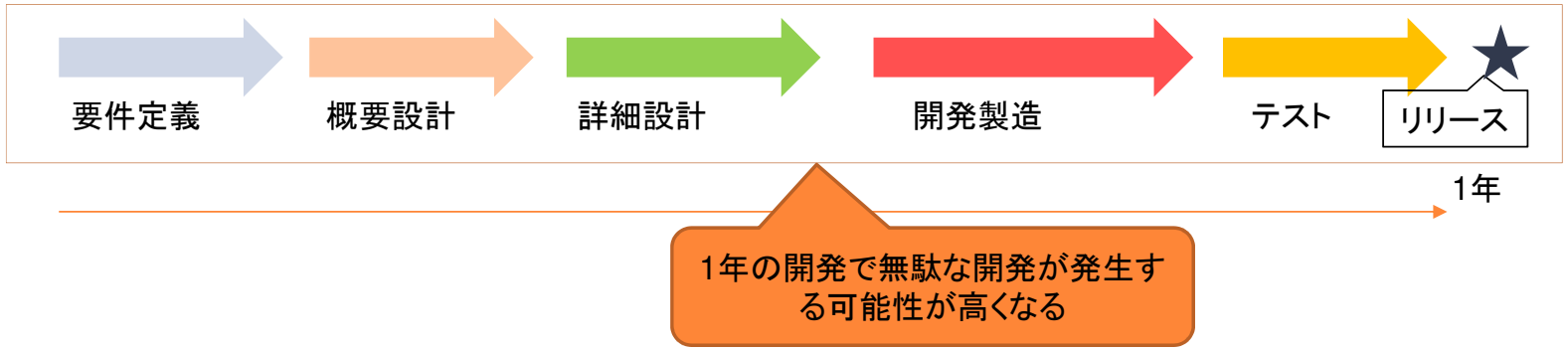
リリースサイクル短縮化の効用

- これまでのアプリケーション開発スパン・サイクル

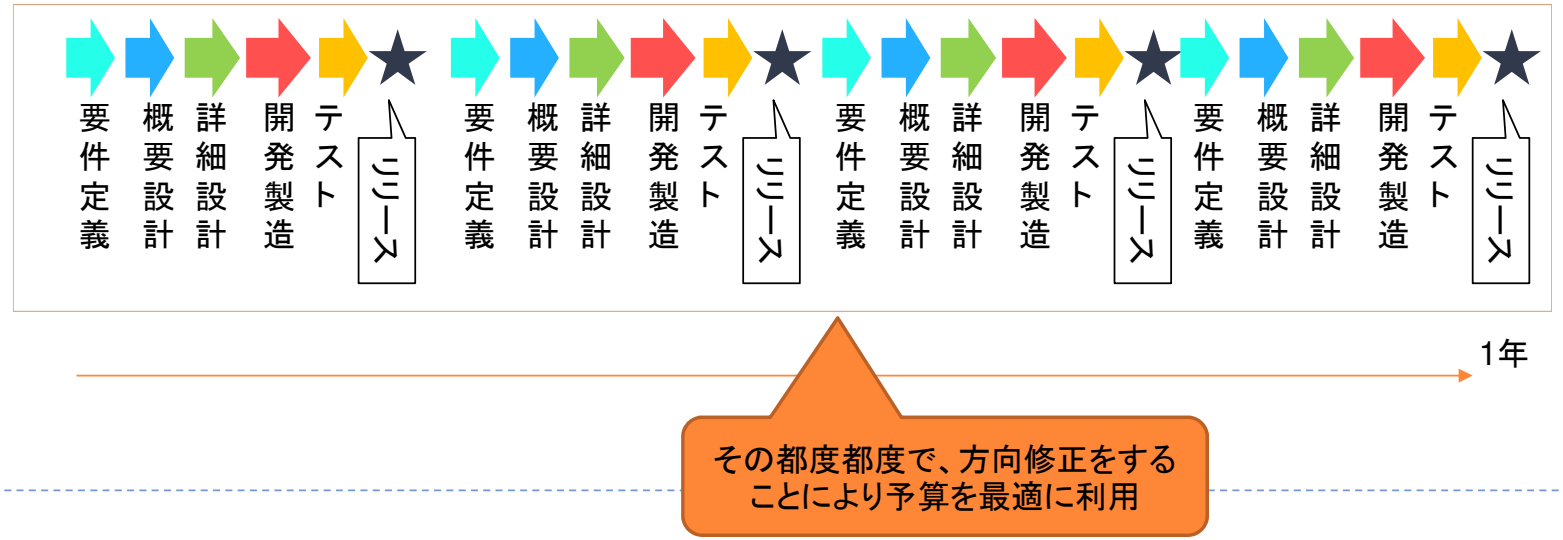


リリースサイクル短縮化の効用

- これまでのアプリケーション開発スパン・サイクル



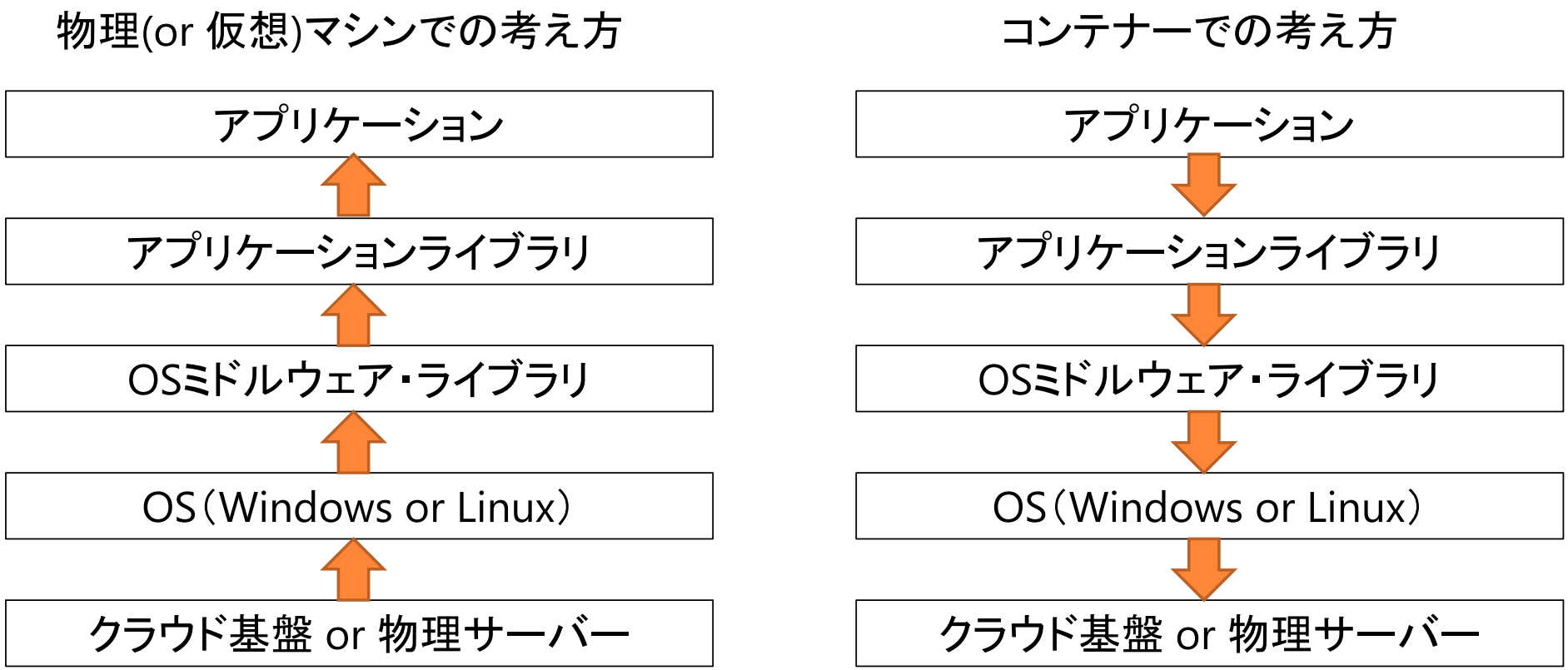
コンテナーを利用してのアプリケーション開発スパン



まずはコンテナーの仕組みを理解しましょう

物理（or仮想）マシンとコンテナー

仮想マシン(クラウド含む)とコンテナーは全く正反対の考え方でできています

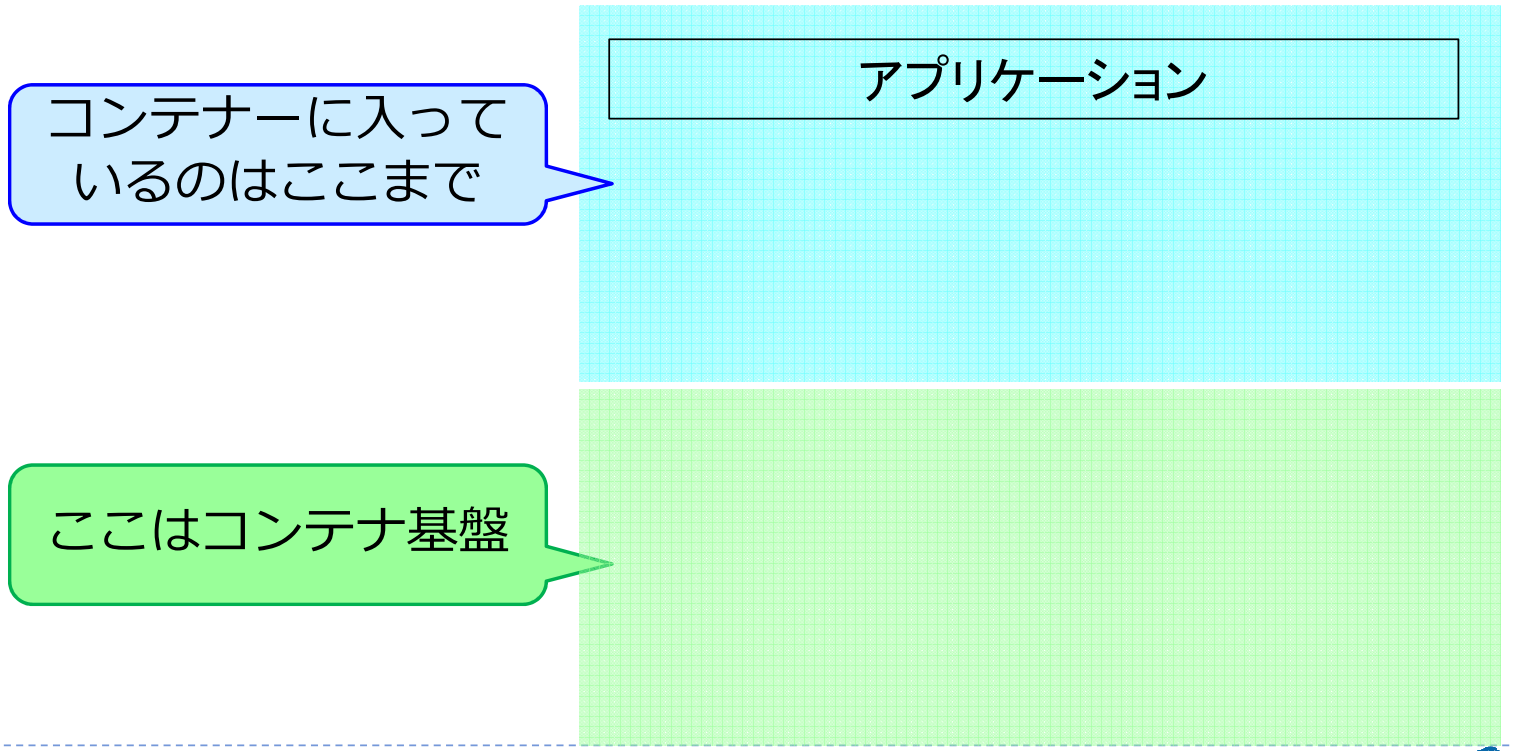


コンテナーの考え方

仮想マシン(クラウド含む)とコンテナーは全く正反対の考え方でできています

物理(or 仮想)マシンでの考え方

コンテナーでの考え方

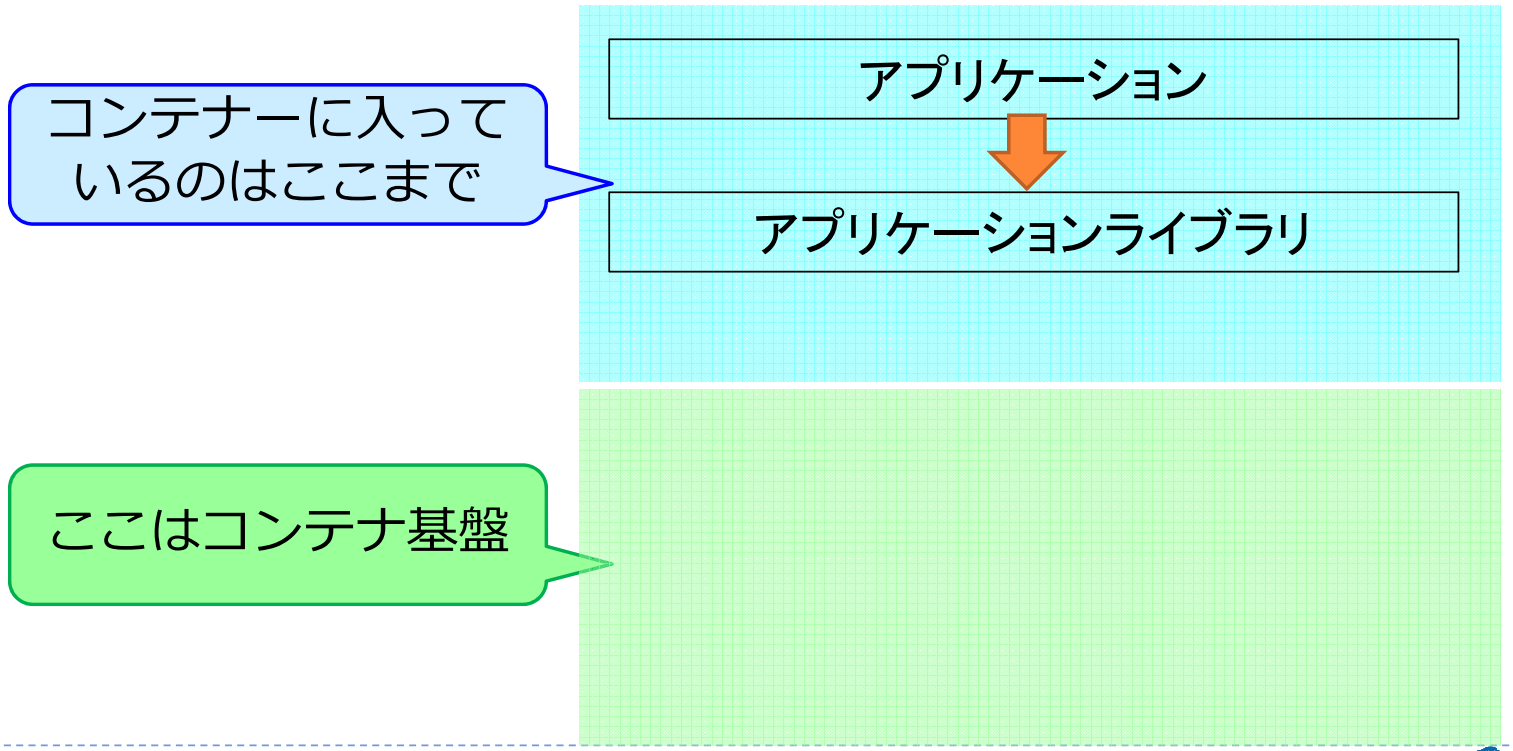


コンテナーの考え方

仮想マシン(クラウド含む)とコンテナーは全く正反対の考え方でできています

物理(or 仮想)マシンでの考え方

コンテナーでの考え方

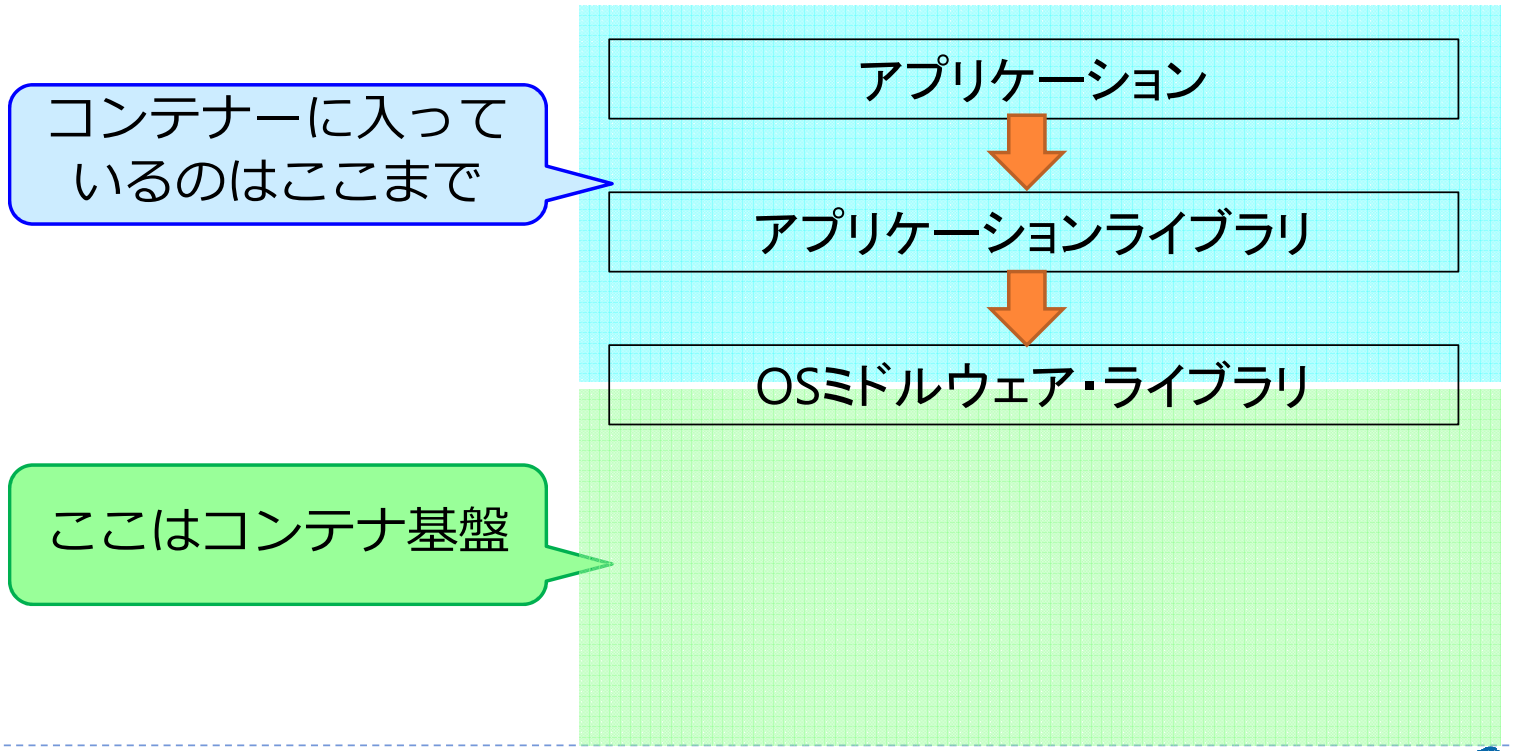


コンテナーの考え方

仮想マシン(クラウド含む)とコンテナーは全く正反対の考え方でできています

物理(or 仮想)マシンでの考え方

コンテナーでの考え方

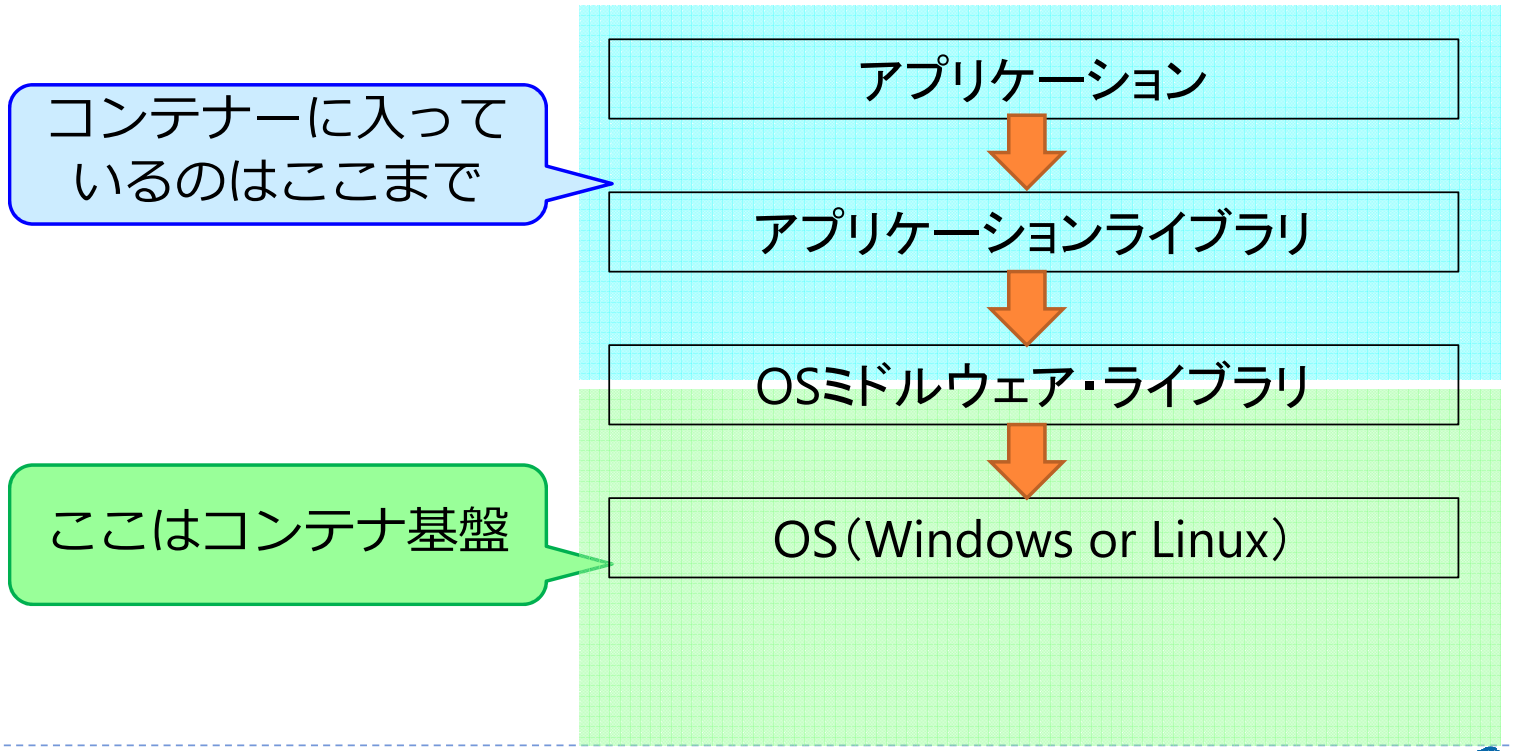


コンテナーの考え方

仮想マシン(クラウド含む)とコンテナーは全く正反対の考え方でできています

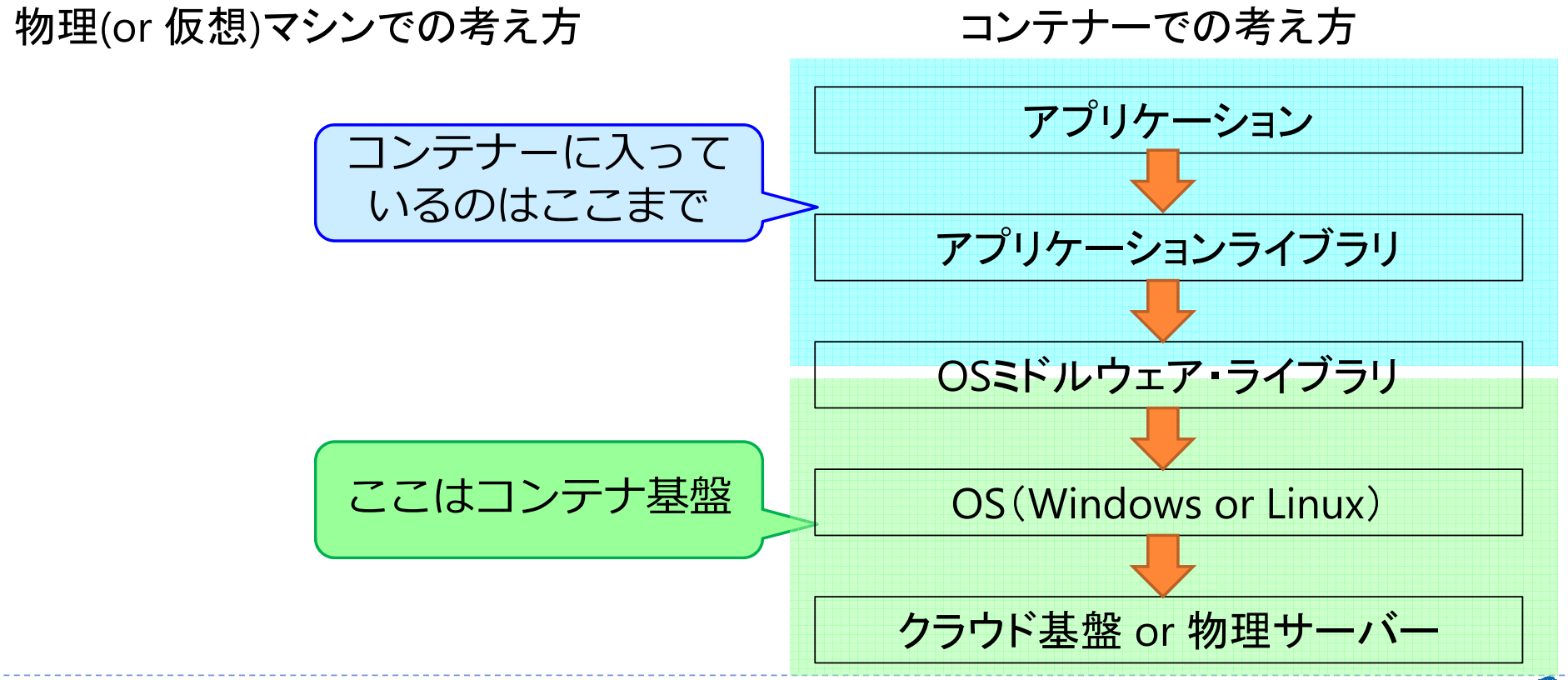
物理(or 仮想)マシンでの考え方

コンテナーでの考え方



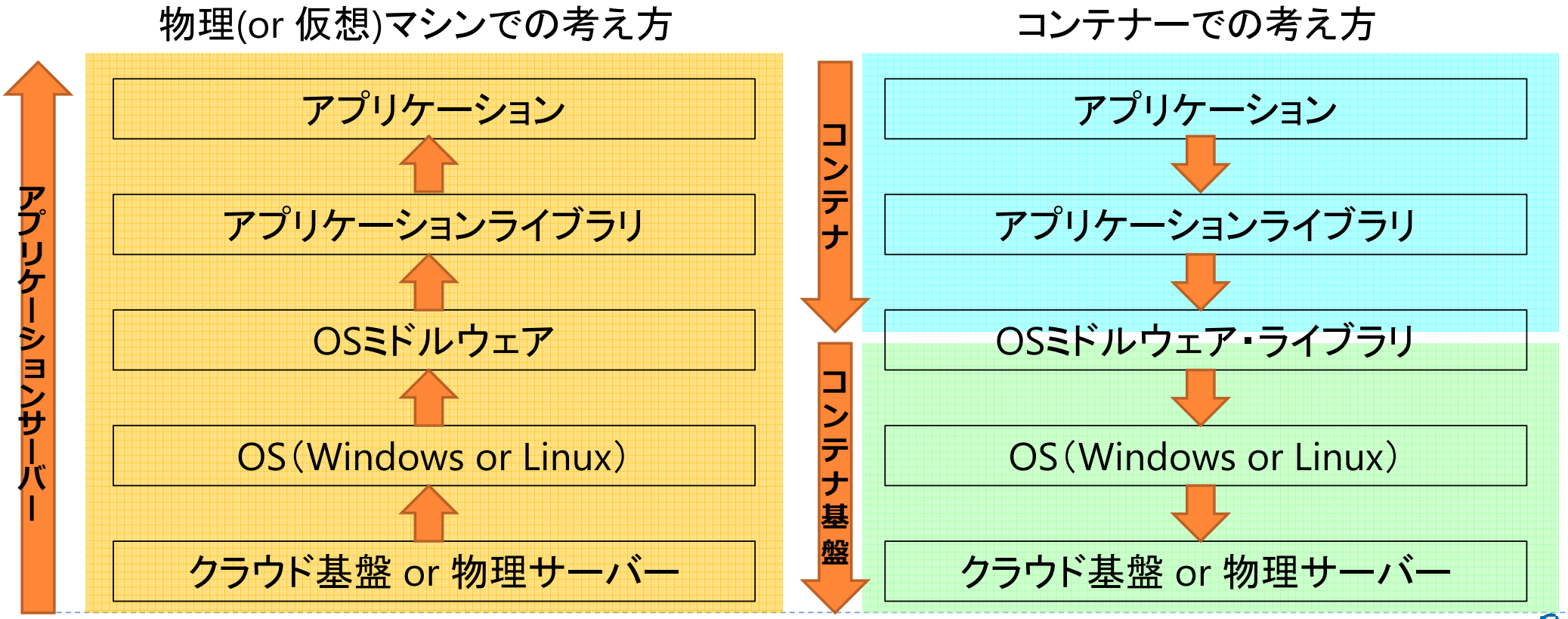
コンテナーの考え方

仮想マシン(クラウド含む)とコンテナーは全く正反対の考え方でできています



物理（or仮想）マシンとコンテナー

仮想マシン(クラウド含む)とコンテナーは全く正反対の考え方でできています

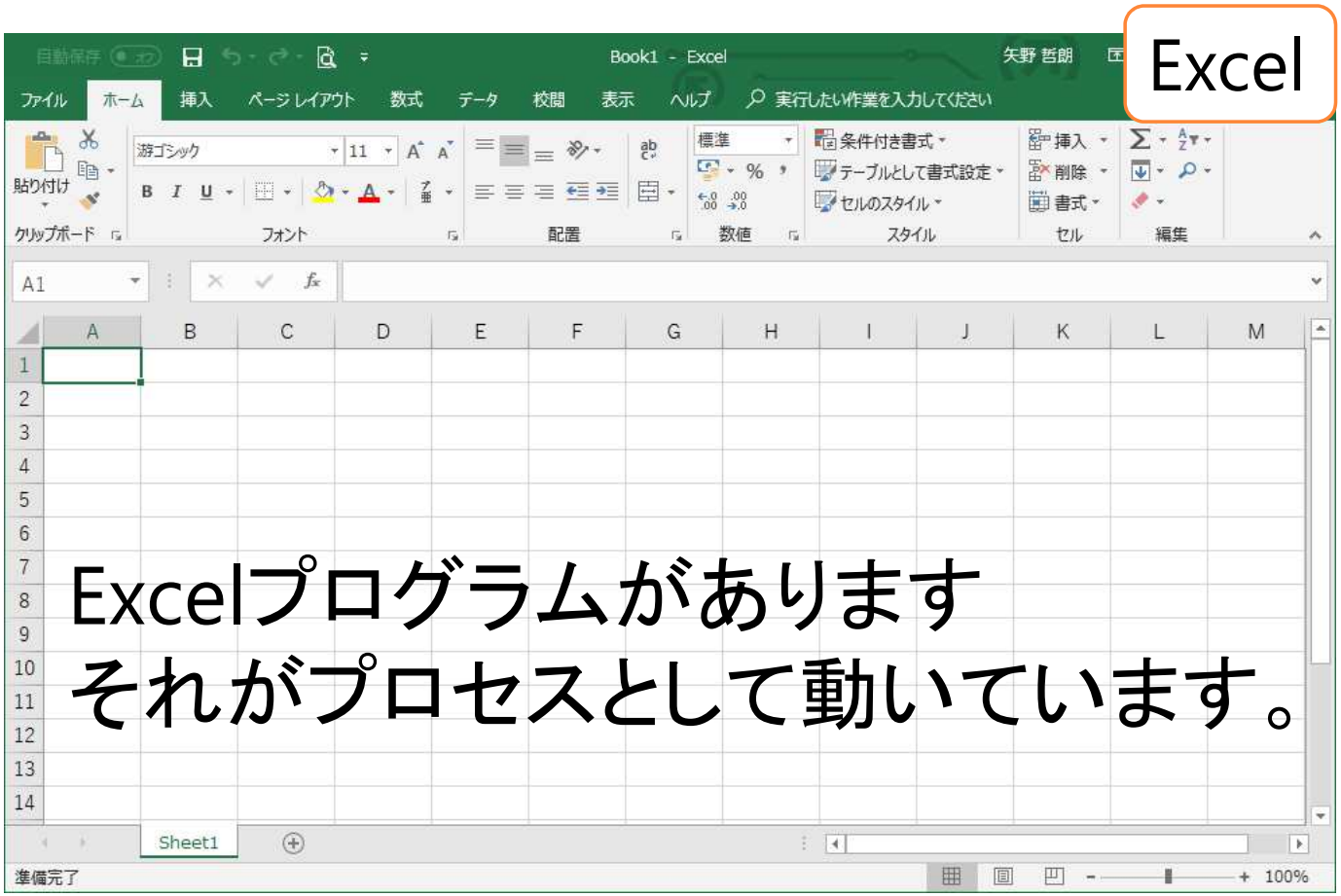


具体的な例で考えてみましょう

※注意：こういうものがあるわけではありません
例えです

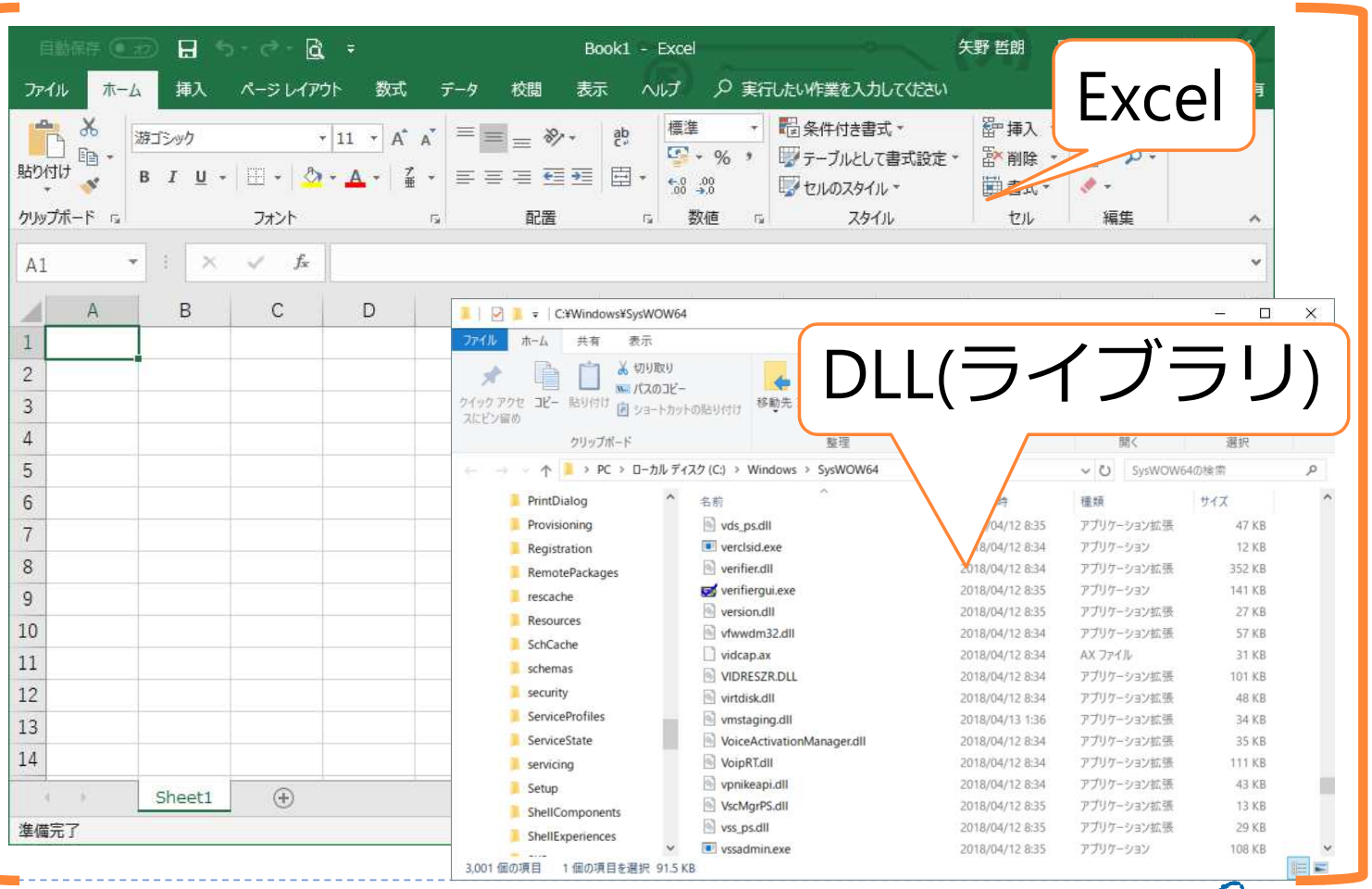
例えば、Excelアプリが
コンテナー化されていたら？

コンテナーとは　実行プロセスです

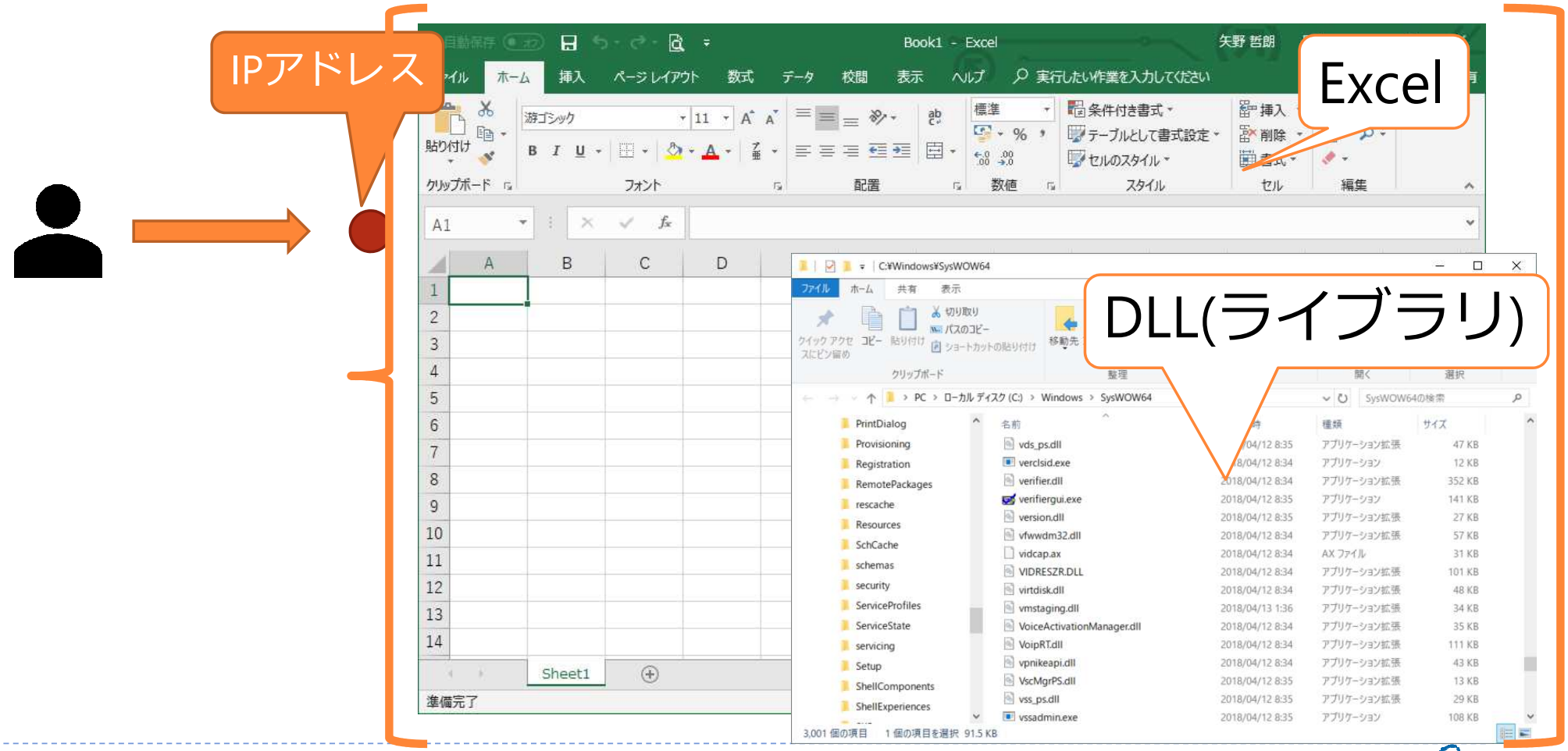


プロセスを実行するためDLL（ライブラリ）がついている

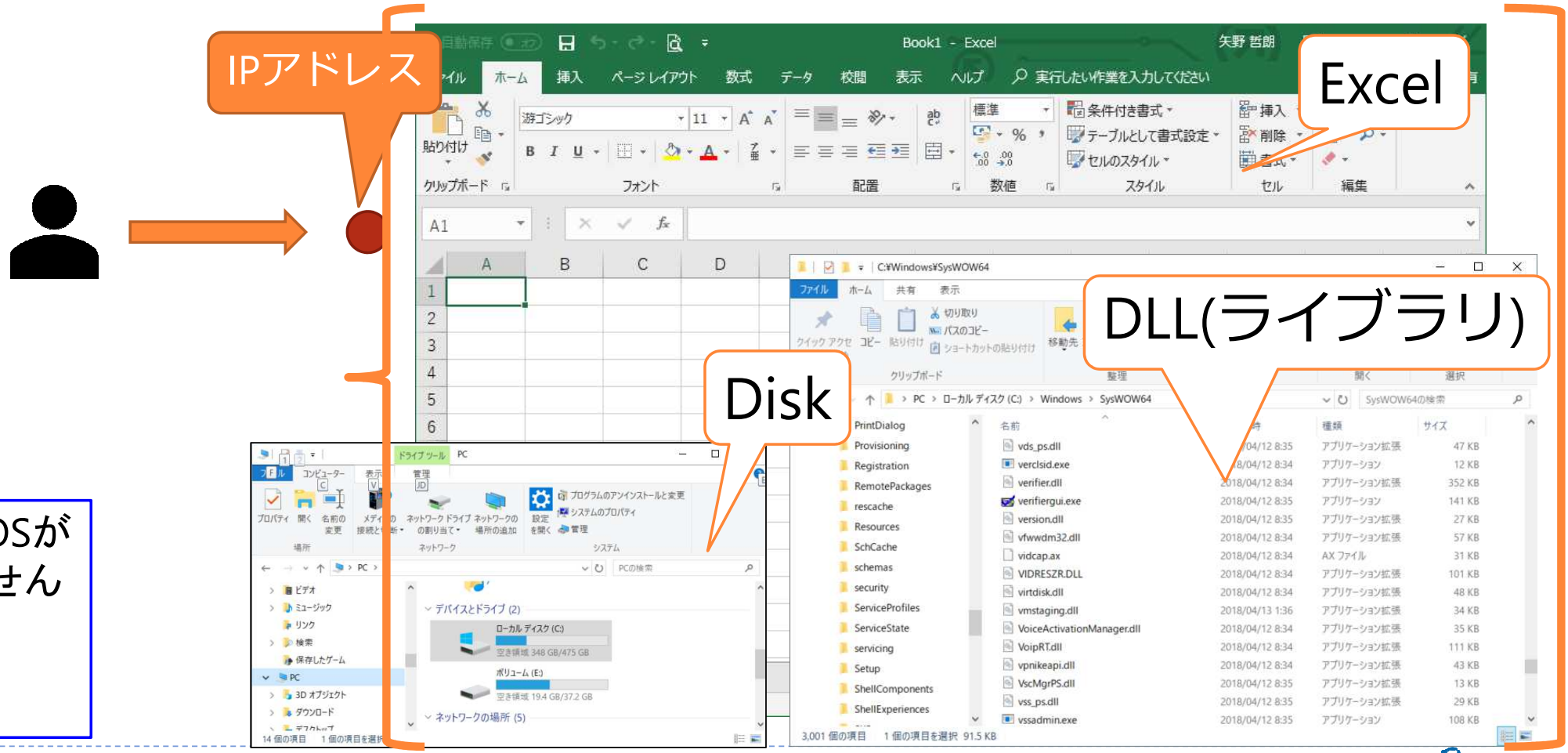
ここまでが、
コンテナー化されている部分



このプロセスにはIPアドレスがついています

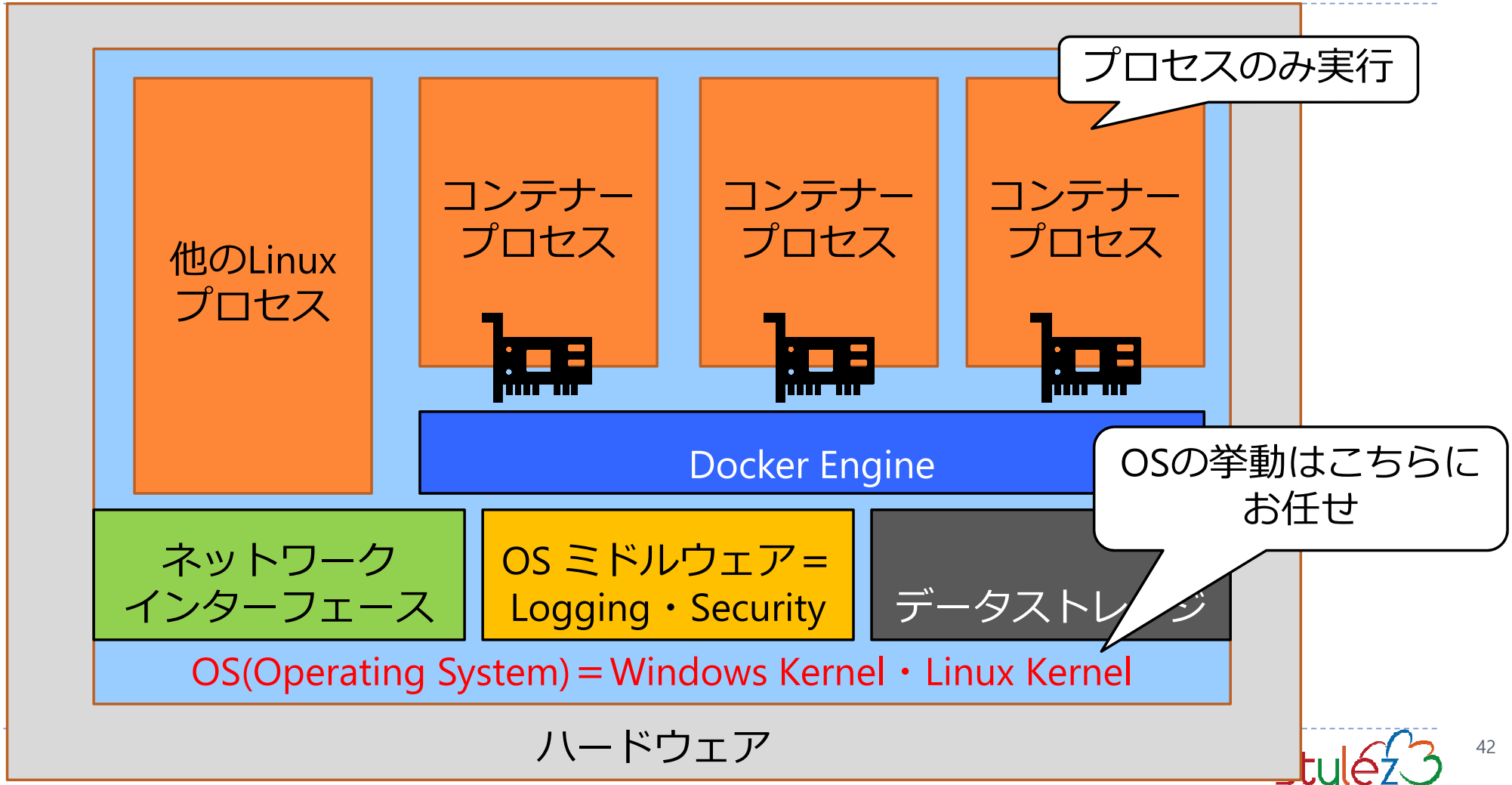


このプロセスにはDiskがついている



ここにはOSが出てきません

コンテナーのプロセス

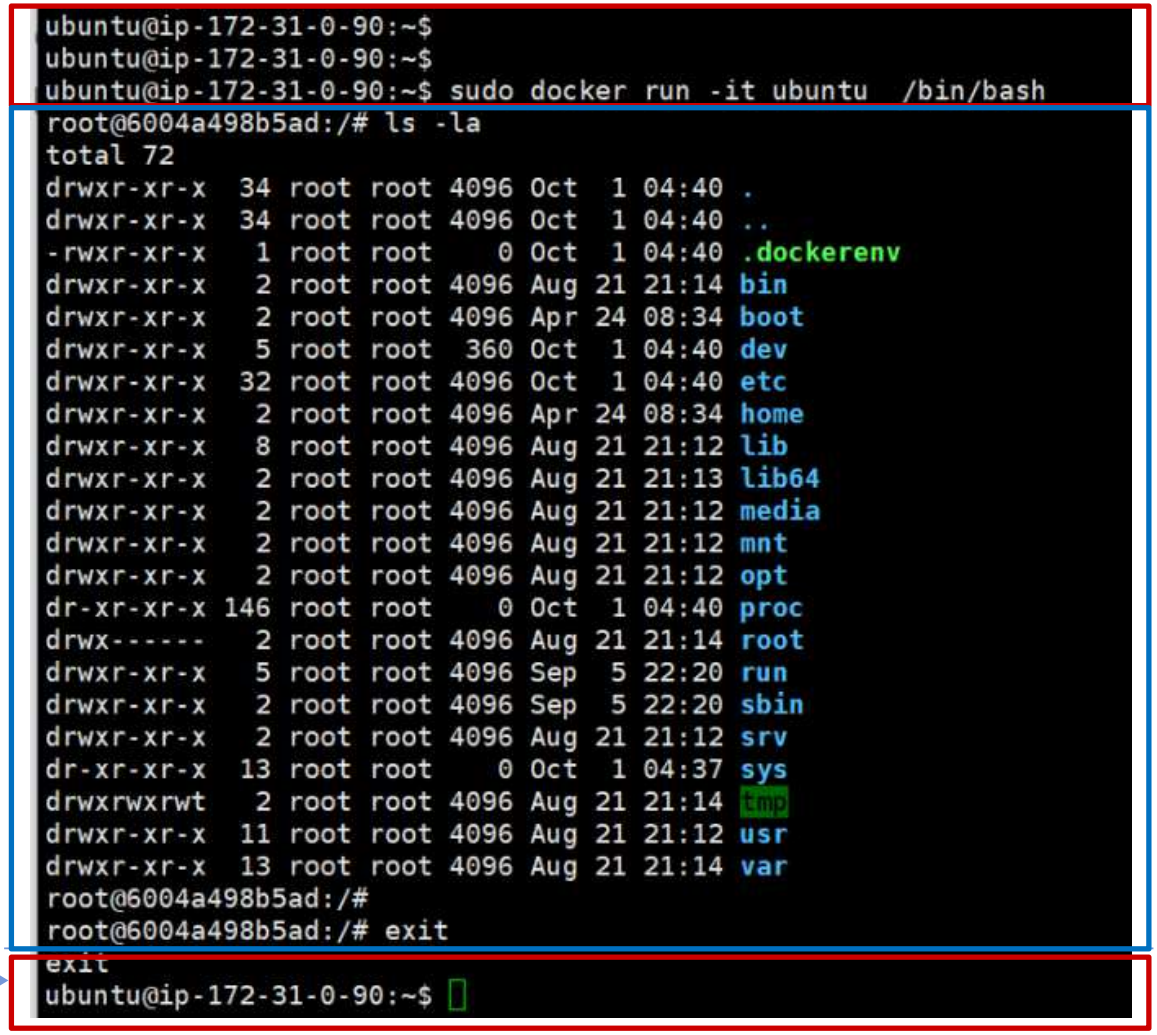

プロセスのみ実行
他のLinux
プロセス
コンテナー
プロセス
コンテナー
プロセス
コンテナー
プロセス
Docker Engine
OSの挙動はこちらに
お任せ
ネットワーク
インターフェース
OS ミドルウェア＝
Logging・Security
データストレージ
OS(Operating System)＝Windows Kernel・Linux Kernel
ハードウェア

結局、コンテナーとは？

OS部分をコンテナー基盤に任せることにより
プロセスのみを独立して動かすことができる
ようにした

そのプロセスに必要なアプリケーションや
ライブラリを一つのパッケージ(塊)にして、
どこでも動かせるようにした

コンテナを動かしてみたらどんな感じ？



コンテナーを起動

コンテナーを実行中

コンテナーを終了

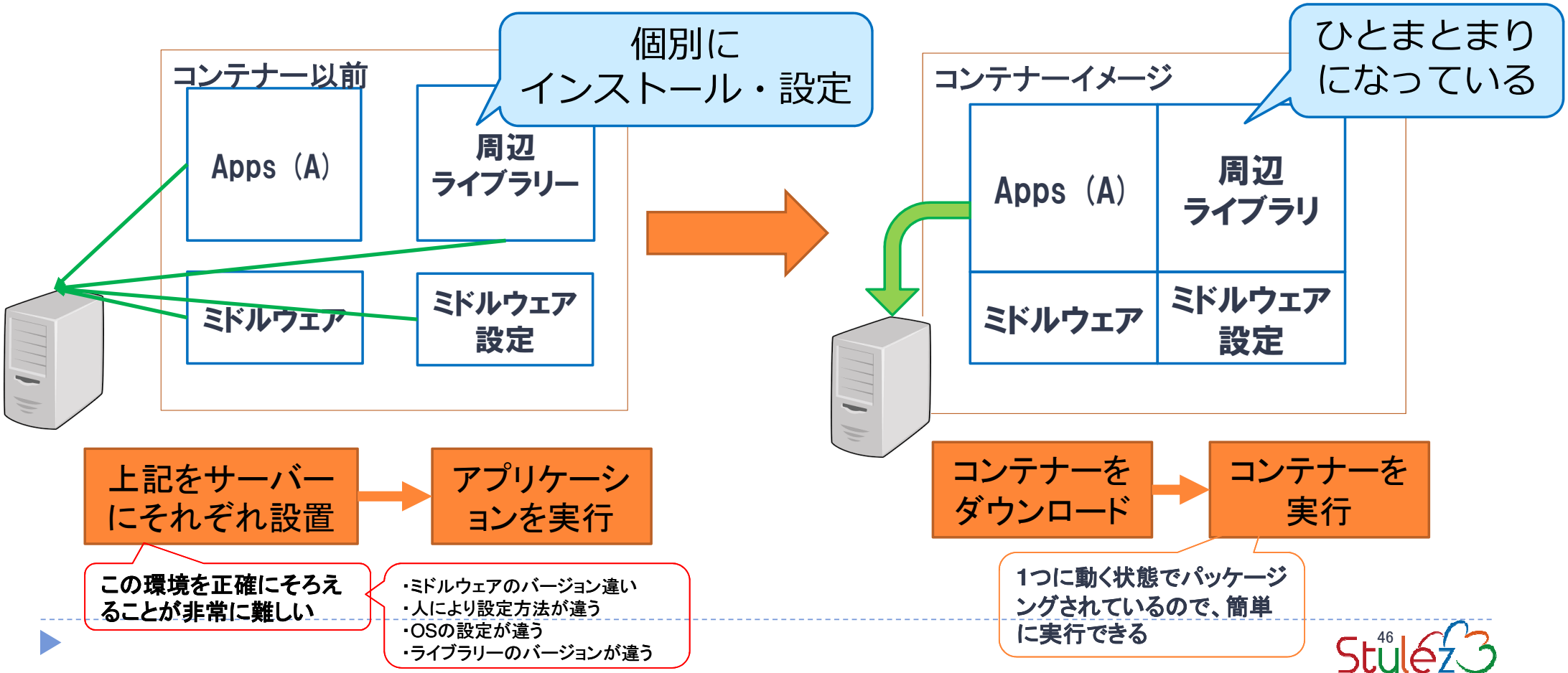
元のプロンプト

コンテナーの最も重要なこと

アプリケーションがパッケージに
まとめられていること

パッケージング

- 必要なアプリケーションとライブラリーが実行可能な状況でパッケージングできる



アプリケーションのポータビリティーの効果

それによってもたらされる数多くのメリット

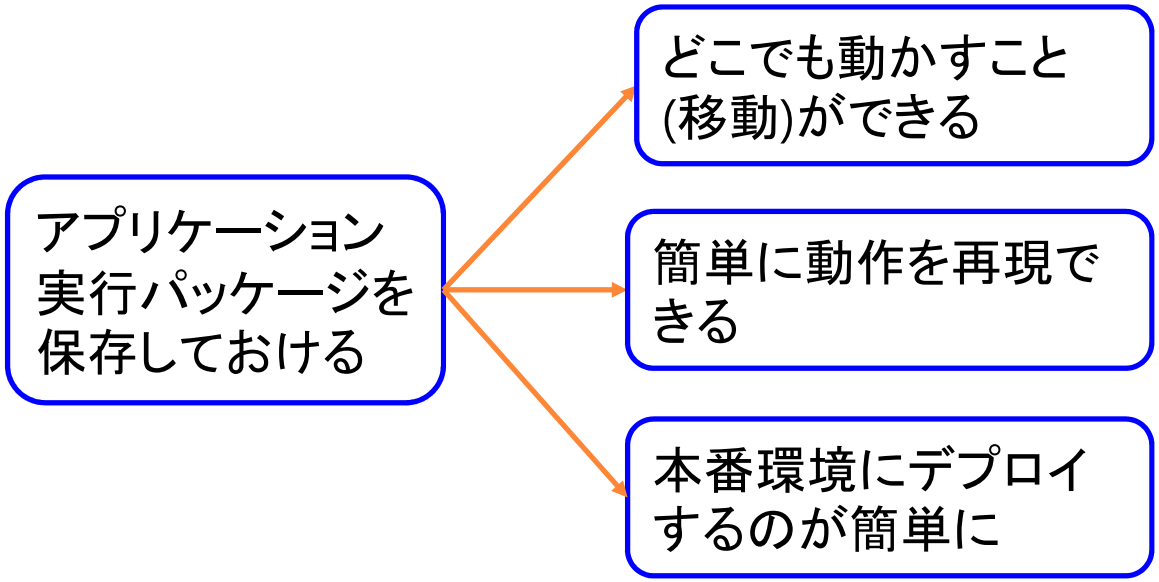
コンテナーの最も重要なこと

アプリケーションをパッケージできる

アプリケーション
実行パッケージを
保存しておける

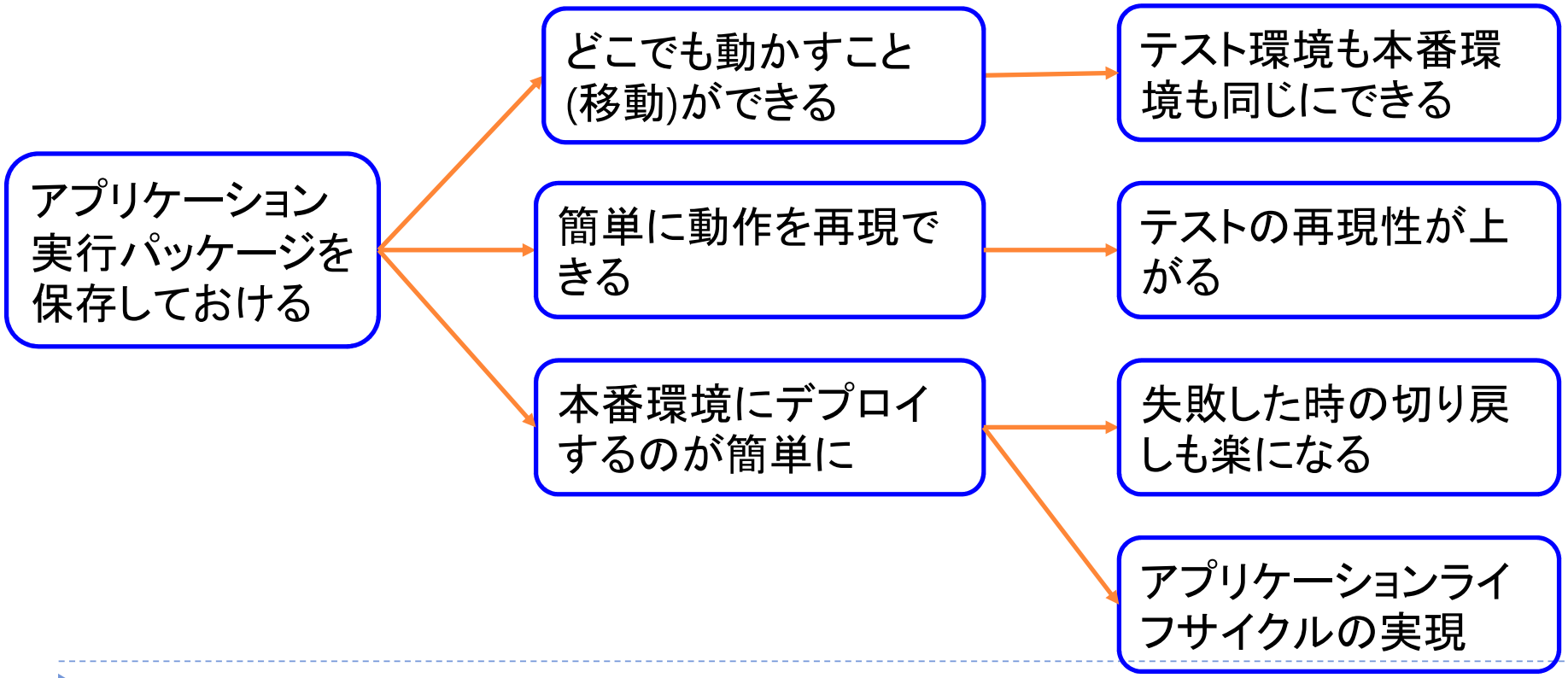
コンテナーの最も重要なこと

ということは、次のメリットが生まれる



コンテナーの最も重要なこと

ということは、次のメリットが生まれる



アプリケーションポータビリティーの効果

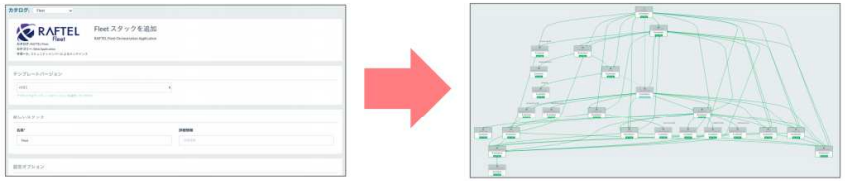
コンテナーを運用しているリクルートテクノロジーズの藤原 涼馬さんのお話

環境を全面的に作り直す

① カタログによる環境構成のコード化 (2/2)

ポイント

短時間で確実に同一環境を構築することができるようになった
<開発面を増やす時でも所要時間は合計15分程度(待ち時間含む)>



全面的に環境をつくりなおすという
思い切った判断も軸に含まれるようになった※

※ただしアプリケーションの永続化データなどの保護は考慮した上で

RECRUIT リクルートテクノロジーズ

17

デプロイと同じ手順で切り戻し

② ビルド・デプロイフローの定型化 (2/3) 切り戻しの定型化

- 切り戻しが短時間かつ容易に実施可能できる※
  ※ただしDBのスキーマなどが変更されていない場合に限る

デプロイ時と
同じ手順で切り戻し

何か誤りがあっても用意かつ確実に切り戻しできることで
エンジニアの学習コストとストレスを軽減できた

RECRUIT リクルートテクノロジーズ

19

https://www.slideshare.net/recruitcojp/rancherrancher

その結果どうなったか？

② ビルド・デプロイフローの定型化 (3/3) 結果としてどうなったか

ポイント

1. Docker CLIの知識がなくてもビルド・デプロイが可能
2. 切り戻しが確実にできる

上記2つを確実に実施できるようになった結果、

開発環境では積極的にデプロイ&テストが行われるように

チーム内の全員が様々なコンポーネントの
デプロイを容易に実施可能になりました。

開発環境へのデプロイの頻度も多い日は10回/日を軽く超えています。

RECRUIT リクルートテクノロジーズ

20

デプロイに関する負荷軽減

↓

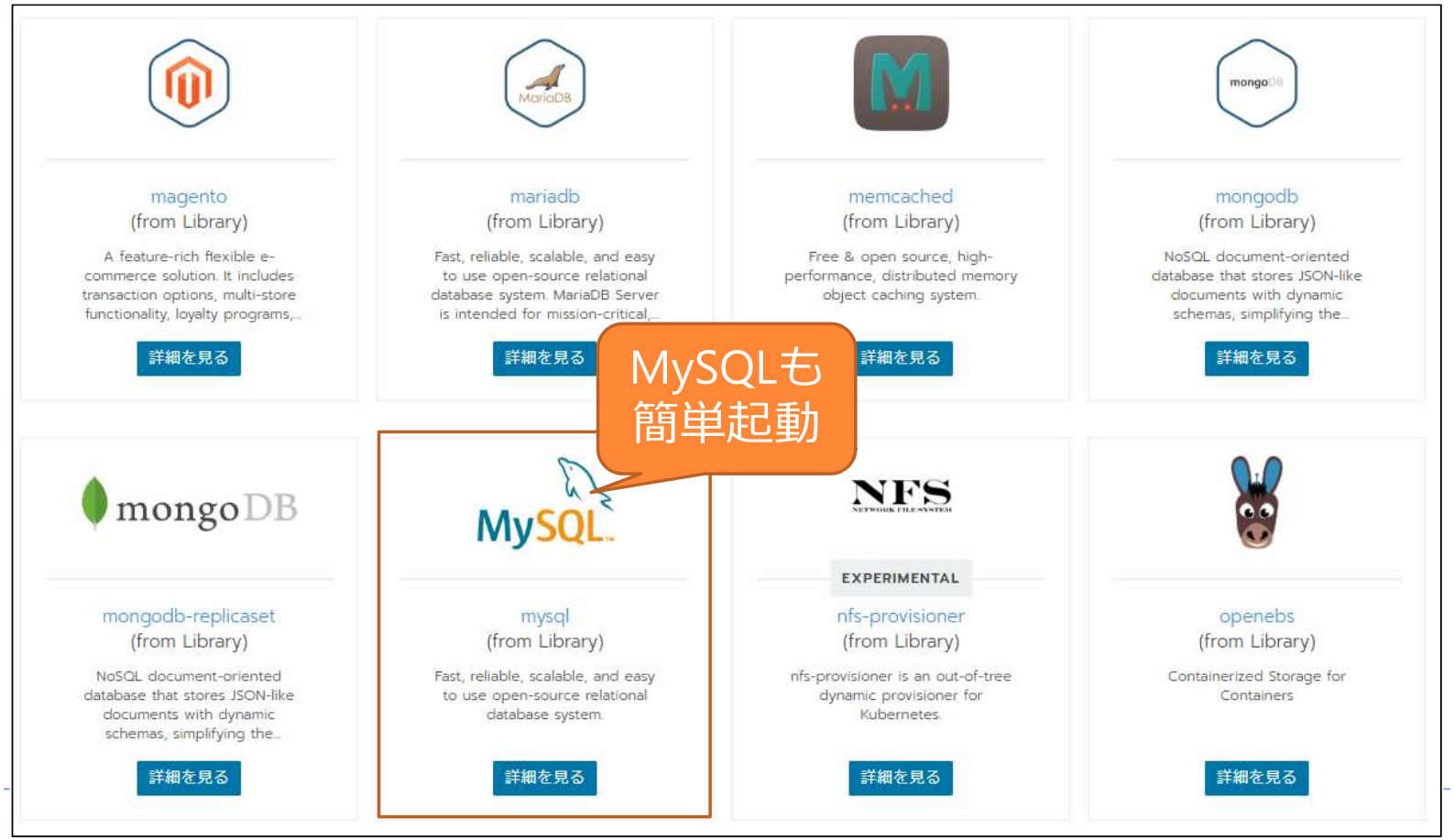
頻繁なリリース 10回/日

↓

市場のニーズの変化に柔軟に対応できる

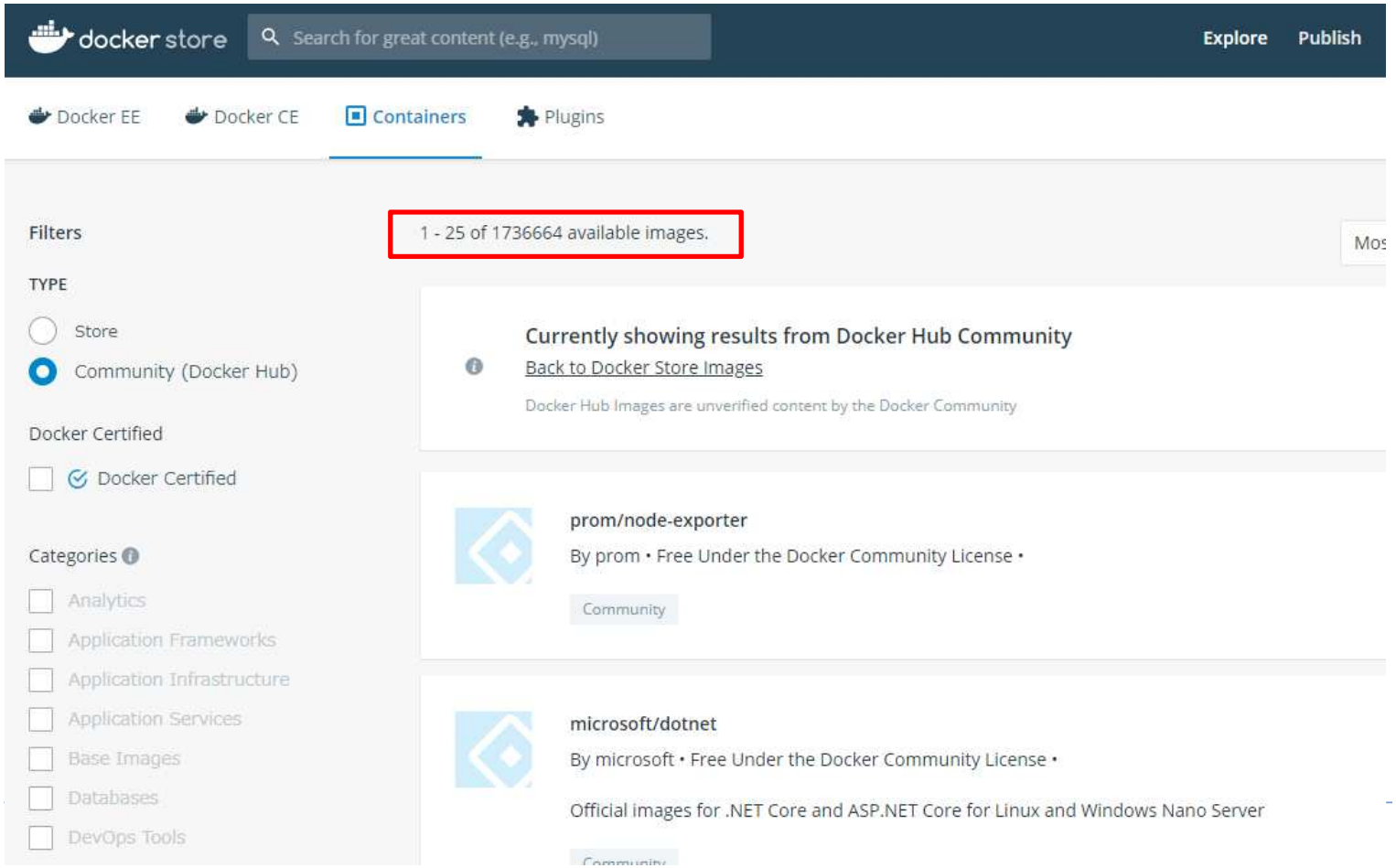
アプリケーションがパッケージ化された

- これにより、アプリケーションの構築がとても簡単になりました。



docker storeに登録されているイメージ数

- docker storeには、173万のコンテナーイメージが登録されています



Kuberntesで何ができるか

本日のお話の構成

- コンテナーのメリット
- Kubernetesとはなんなのか
- KubernetesはGoogleがAWSに仕掛けたゲームチェンジ
- Kubernetesによって変わるSIerの役割

Kubernetesとは何か？

- Kubernetesとは自動復旧ツールである
- Kubernetesとはリソース管理ツールである
- Kubernetesとは設定管理ツールである
- Kubernetesとはログ集約システムである
- Kubernetesとは権限管理ツールである
- Kubernetesとは冗長構成ツールである
- Kubernetesとはロードバランサーである
- Kubernetesとはアプリケーションデプロイツールである
- Kubernetesとはロールバックツールである
- Kubernetesとはオートスケーリングツールである
- Kubernetesとは死活監視ツールである
- Kubernetesとはジョブ実行ツールである

Kubernetesとは何か？

- Kubernetesとは自動復旧ツールである
- Kubernetesとはリソース管理ツールである
- Kubernetesとは設定管理ツールである
- Kubernetesとはログ集約システムである
- Kubernetesとは権限管理ツールである
- Kubernetesとは冗長構成ツールである
- Kubernetesとはロードバランサーである
- Kubernetesとはアプリケーションデプロイツールである
- Kubernetesとはロールバックツールである
- Kubernetesとはオートスケーリングツールである
- Kubernetesとは死活監視ツールである
- Kubernetesとはジョブ実行ツールである

Kubernetesとは、
アプリケーションを実行するために個別に構成していたツールをまとめて利用できるようにしたもの

Kubernetesの特徴

3行で

- Immutable Infrastructure
  （変更を積み重ねず、都度作る/作り直す）

- 宣言的設定
  （あるべき姿を宣言し、その姿に収束させる）

- 自己修復
  （どこかが壊れても、人を介さずに修復する）

Kubernetesのしくみ やさしく学ぶ 内部構造とアーキテクチャー より
https://www.slideshare.net/ToruMakabe/kubernetes-120907020

Kubernetesの特徴

3行で

- Immutable Infrastructure
  （変更を積み重ねず、都度作る/作り直す）
- 宣言的設定
  （あるべき姿を宣言し、その姿に収束させる）
- 自己修復
  （どこかが壊れても、人を介さずに修復する）

つまりこういうこと

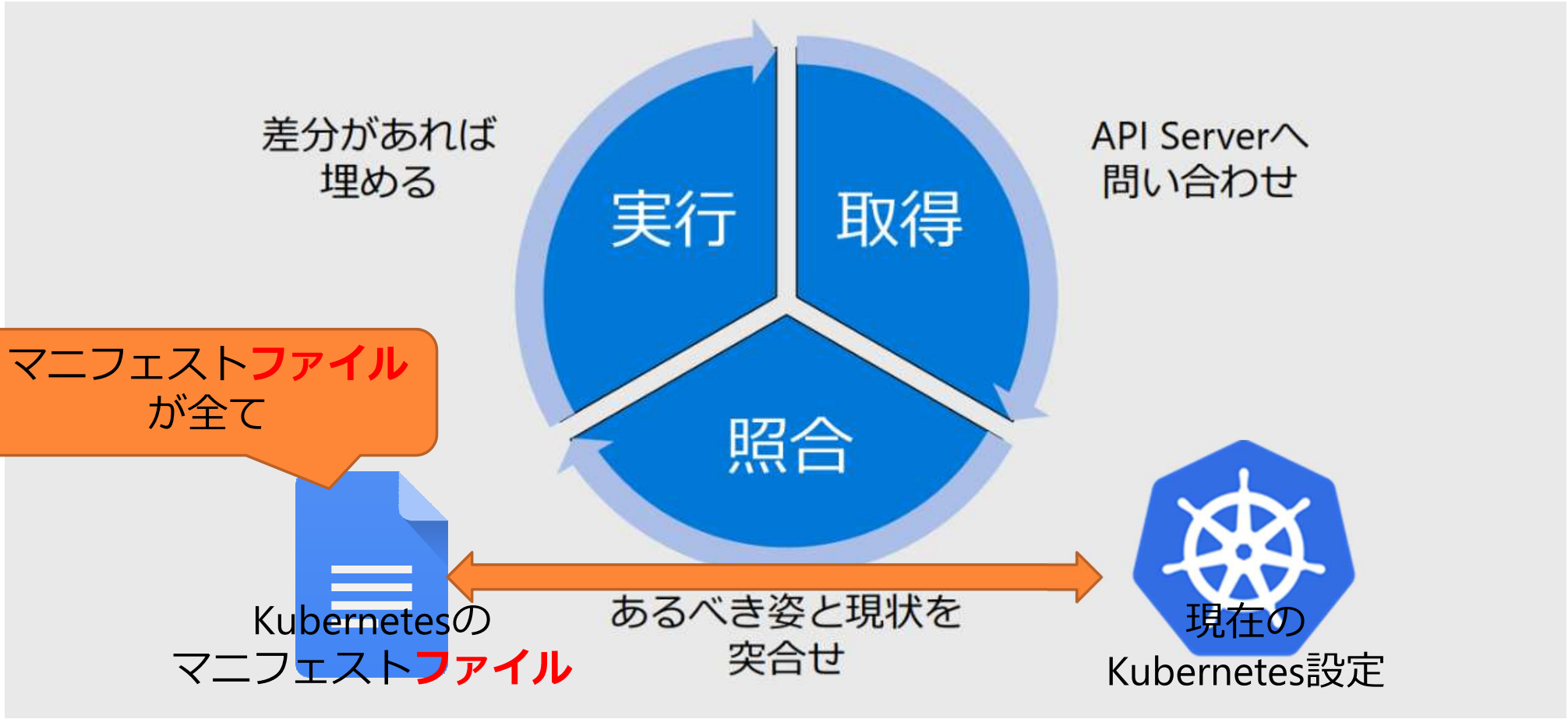
「手順書よ、さようなら」

「インフラの全てをコード化する」

「Kubernetesが全てをコントロール」

Kubernetesのしくみ やさしく学ぶ 内部構造とアーキテクチャー より
https://www.slideshare.net/ToruMakabe/kubernetes-120907020

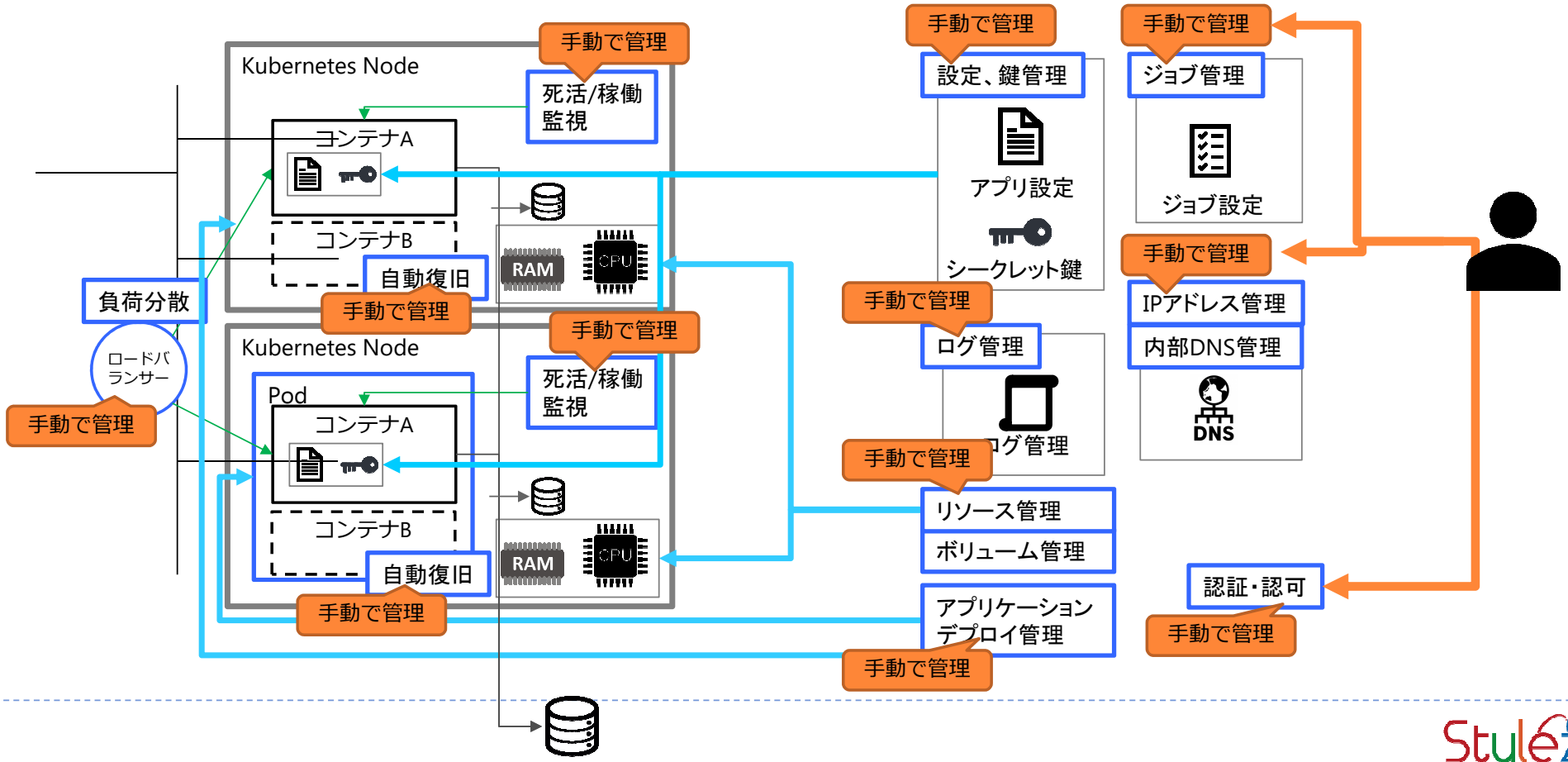
Reconciliation Loop（突き合わせループ）



Kubernetesのしくみ やさしく学ぶ 内部構造とアーキテクチャー より
https://www.slideshare.net/ToruMakabe/kubernetes-120907020

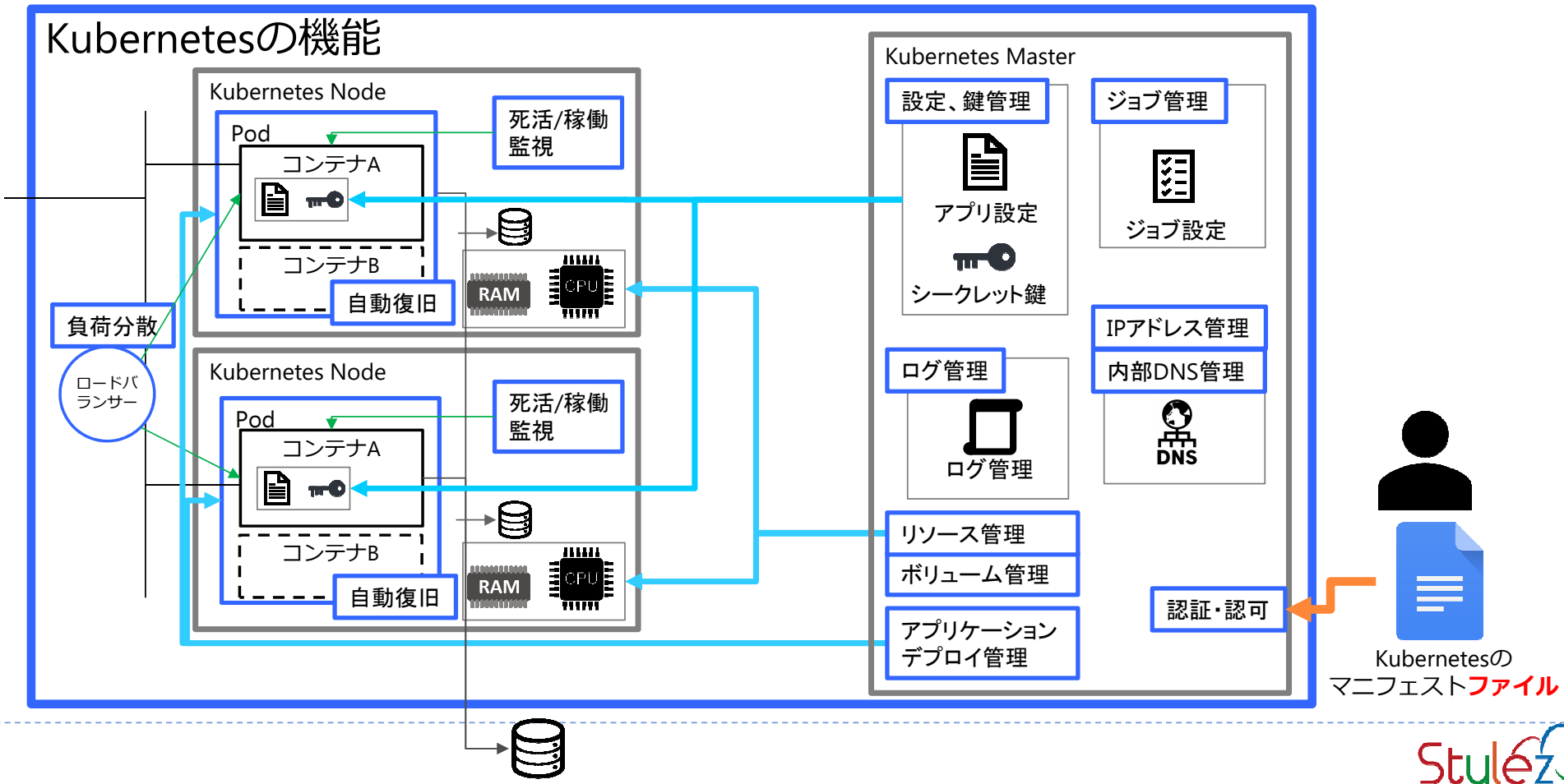
Kubernetes以前

Kubernetesがないとそれぞれの状態をそれぞれのインターフェースと設定方法で設定しなければいけなかった



Kubernetesによる変化

Kubernetesにより統一したインターフェースとファイルで全てを設定・管理・自動かできるようになる

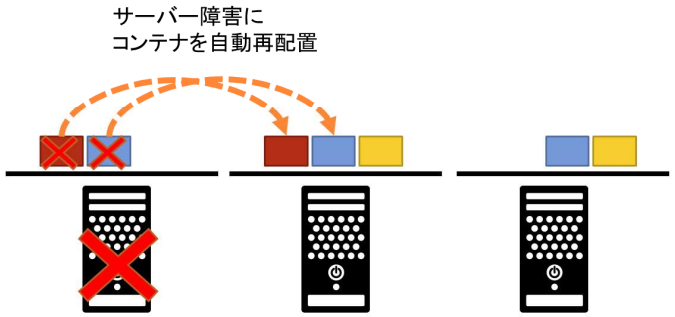


Kubernetesのメリット

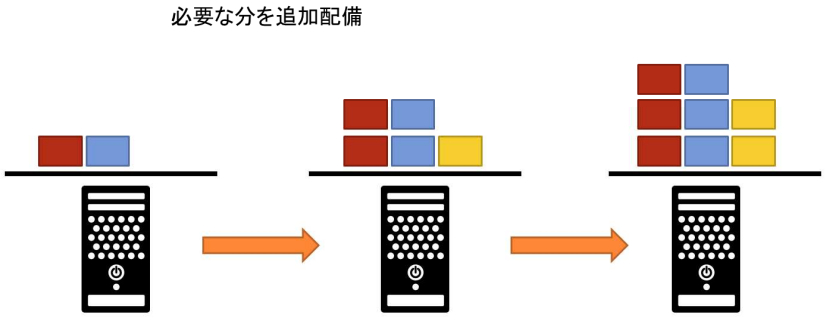
Kubernetesにより
構成をまとめて
管理できるようになった

Kubernetesの機能（1）

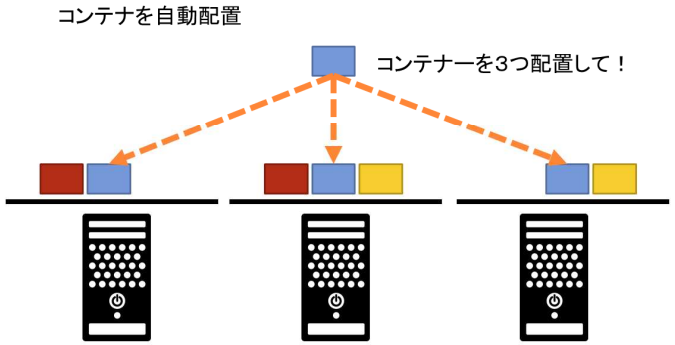
セルフヒーリング（自動リカバリー）



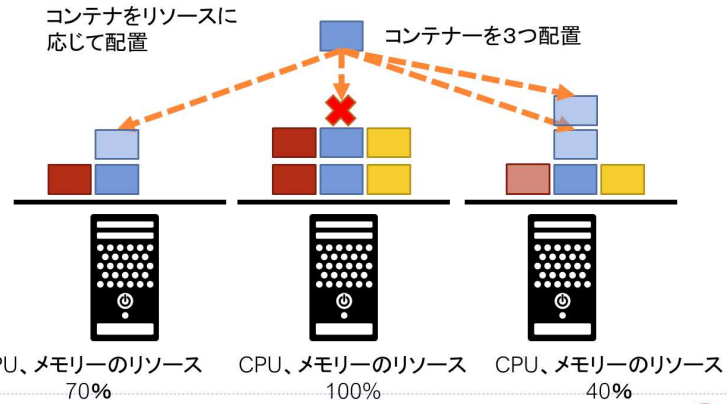
スケーリング／オートスケーリング



スケジューリング



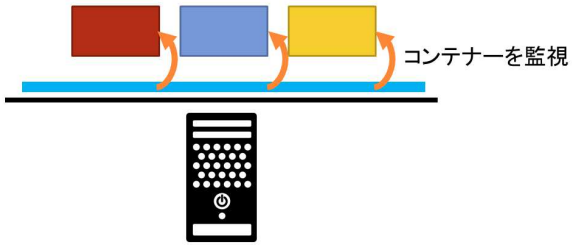
リソース管理



Kubernetesの機能（2）

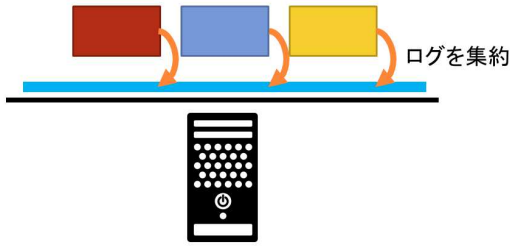
サービス監視

コンテナーが配置されたら
自動的に監視



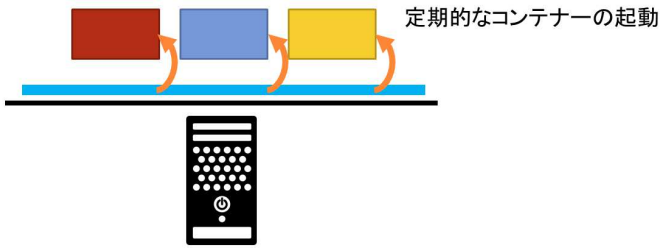
ログの管理

コンテナーが配置されたら
自動的にログ収集



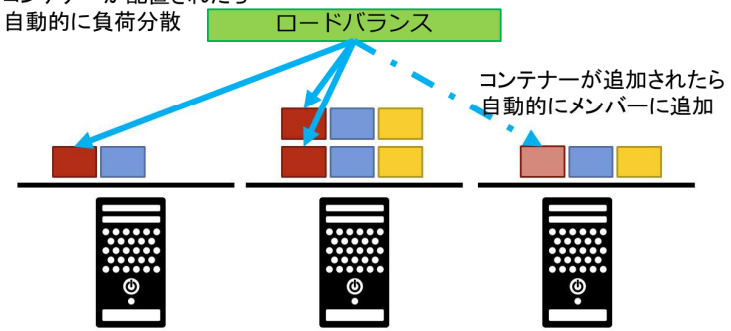
ジョブ実行

ジョブスケジュールに沿って
自動的にコンテナー起動



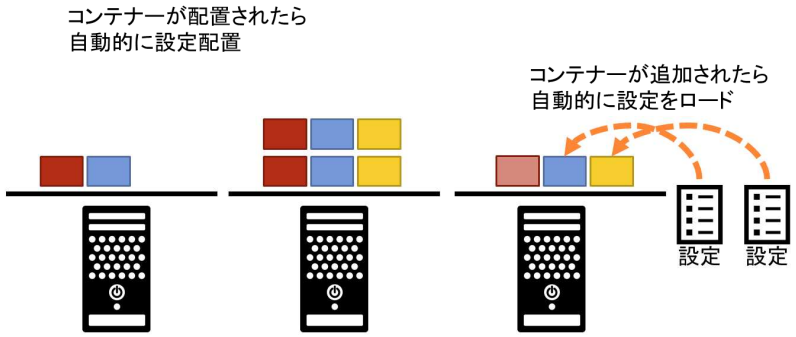
サービスロードバランス

コンテナーが配置されたら
自動的に負荷分散

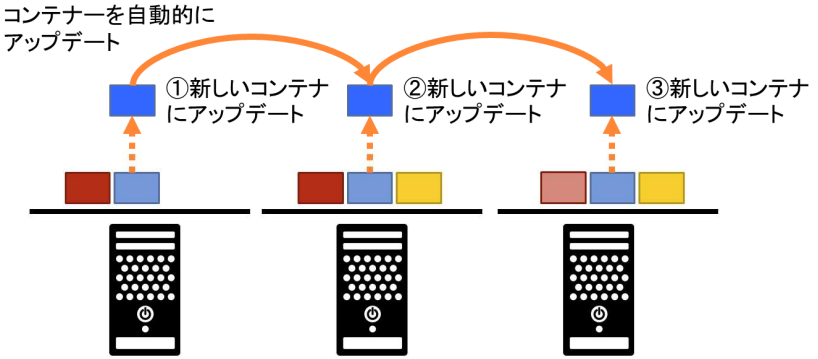


Kubernetesの機能（3）

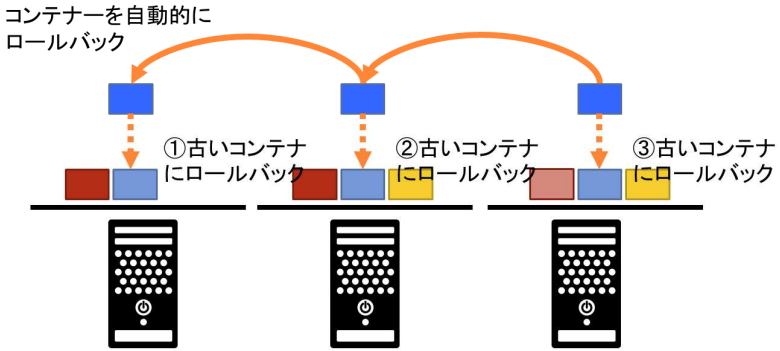
設定の自動ロード



コンテナーローリングアップデート



コンテナーロールバック



Kubernetesとはズバリ！

複数のサーバーをまとめて
一つの基盤として管理する
アプリケーション実行基盤

なぜ、GoogleはKubernetesを開発したのか？

本日のお話の構成

- コンテナーのメリット
- Kubernetesとはなんなのか
- KubernetesはGoogleがAWSに仕掛けたゲームチェンジ
- Kubernetesによって変わるSIerの役割

クラウドの歴史

2006

Stylez3

クラウドの歴史

2006

2008

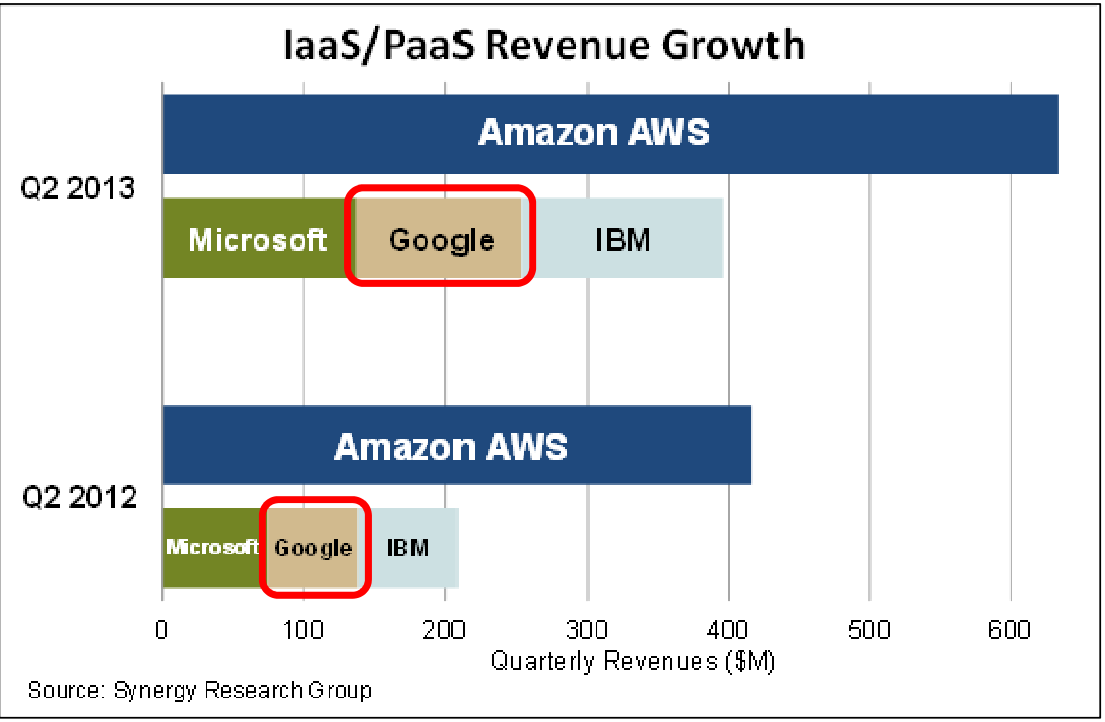
クラウドの歴史

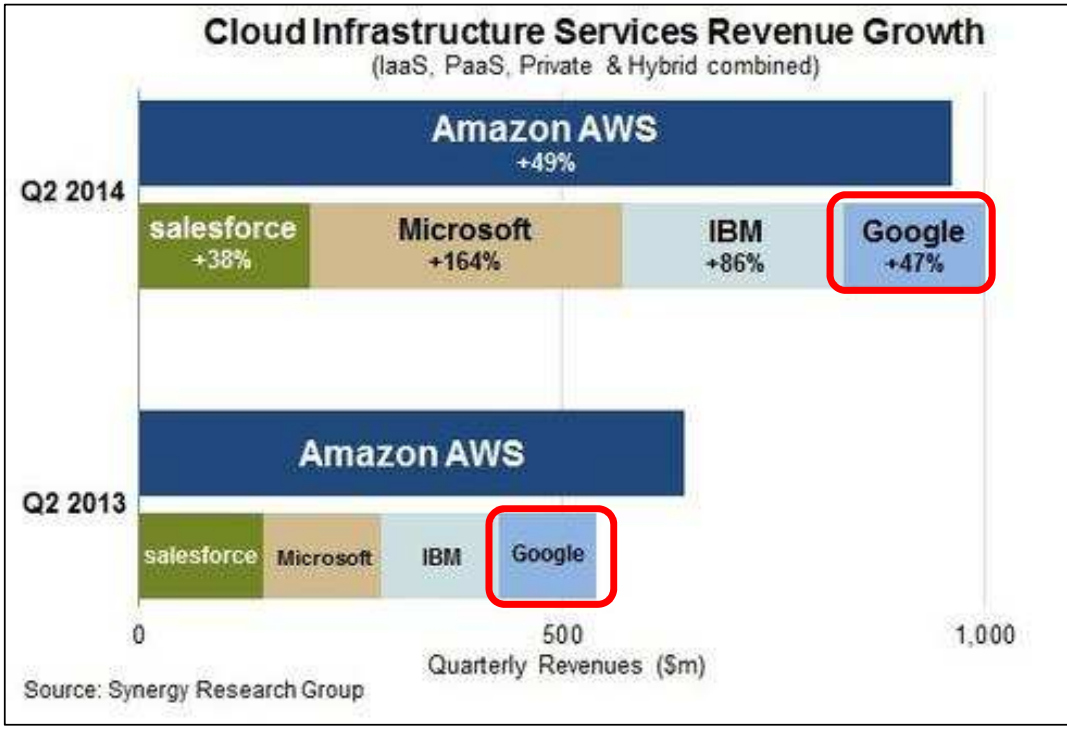
2006

2008

2012

2012～2014年のAWSとGoogleのシェア





2014～2015年のサービスリスト比較

2014年1月 AWSサービスリスト

- Amazon EC2
- Auto Scaling
- Elastic Load Balancing
- Amazon WorkSpaces
- Amazon VPC
- Amazon Route 53
- AWS Direct Connect
- Amazon S3
- Amazon Glacier
- Amazon Elastic Block Storage
- AWS Storage Gateway
- AWS Import/Export
- Amazon CloudFront
- Amazon RDS
- Amazon DynamoDB
- Amazon ElastiCache
- Amazon Redshift
- Amazon EMR
- Amazon Kinesis
- AWS Data Pipeline
- Amazon AppStream
- Amazon SQS
- Amazon SNS
- Amazon SWF
- Amazon SES
- Amazon CloudSearch
- Amazon Elastic Transcoder
- AWS Identity and Access Management
- AWS CloudTrail
- Amazon CloudWatch
- AWS Elastic Beanstalk
- AWS CloudFormation
- AWS OpsWorks
- AWS CloudHSM

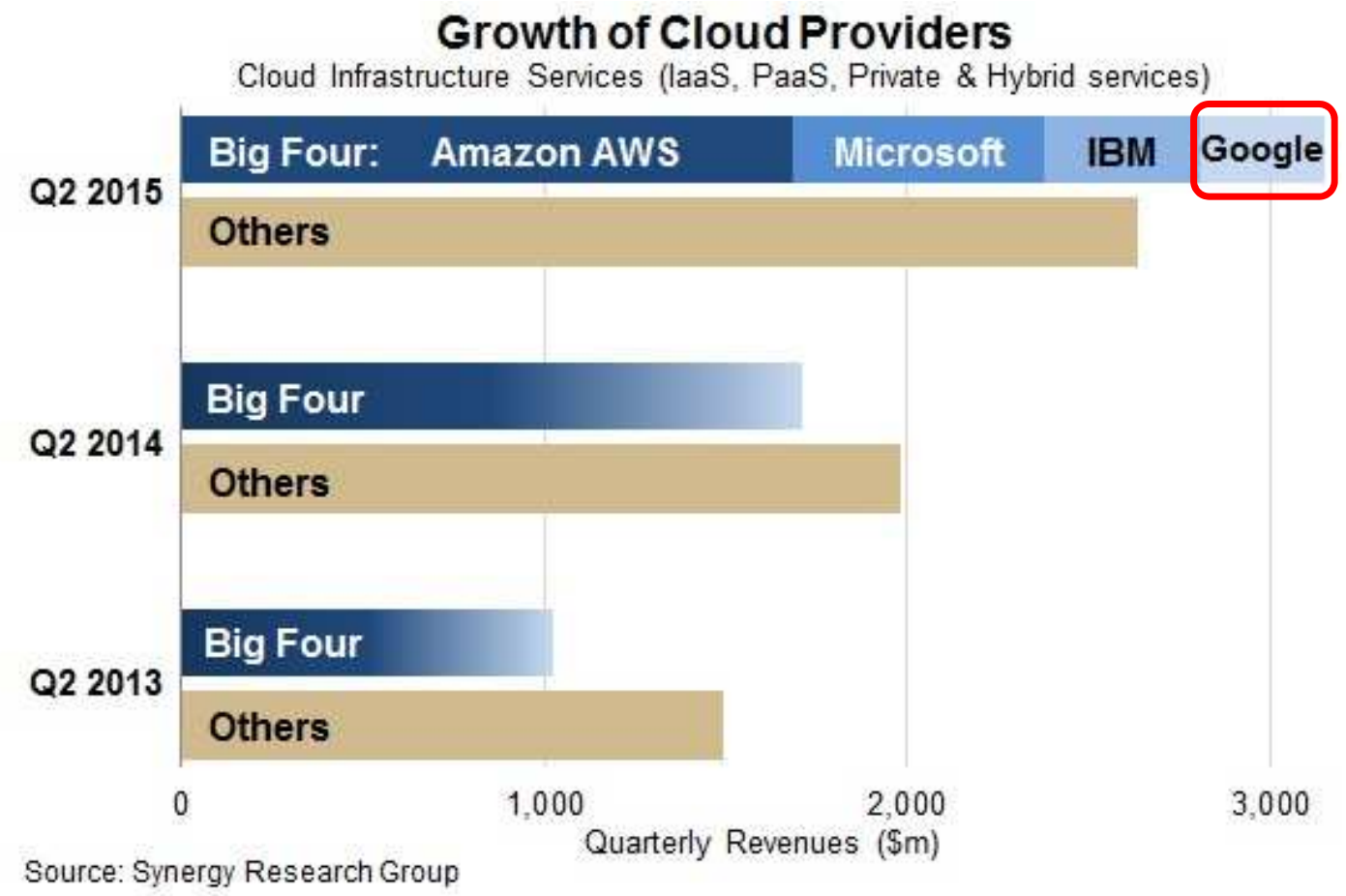
2015年1月 GCPサービスリスト

- Google App Engine
- Google Cloud Storage
- Google BigQuery and Prediction API
- Google Cloud SQL
- Google Compute Engine
- Cloud Dataflow
- Cloud Datastore
- Persistent Disk
- Cloud Endpoints
- Cloud Caching
- Cloud Queues
- Cloud DNS
- Cloud Deployment Manager
- Cloud Pub/Sub
- Cloud Messaging
- Authentication
- Managed VM
- Container Engine
- Google Cloud Monitoring

https://media.amazonwebservices.com/AWS_Overview.pdf

https://www.researchgate.net/profile/Krishnan_S_P_T2/publication/300149417_Getting_Started_with_Google_Cloud_Platform/links/5991dbe9458515a8a24bbeaa/Getting-Started-with-Google-Cloud-Platform.pdf?origin=publication_detail

先行者に追いつけていないGoogle



ゲームチェンジするために必要なもの

AWSの サービス	AWSの サービス	AWSの サービス	AWSの サービス
オペレーション			
認証 システム	ネットワーク 管理システム	リソース管理 システム	
AWSのハードウェア、ネットワーク			

AWSのサービス基盤

ゲームチェンジするために必要なもの

AWSの サービス	AWSの サービス	AWSの サービス	AWSの サービス
オペレーション			
認証 システム	ネットワーク 管理システム	リソース管理 システム	
AWSのハードウェア、ネットワーク			

AWSのサービス基盤

GCPの サービス	GCPの サービス	Googleが 足りてない部分	
オペレーション			
認証 システム	ネットワーク 管理システム	リソース管理 システム	
GCPのハードウェア、ネットワーク			

GCPのサービス基盤

ゲームチェンジするために必要なもの

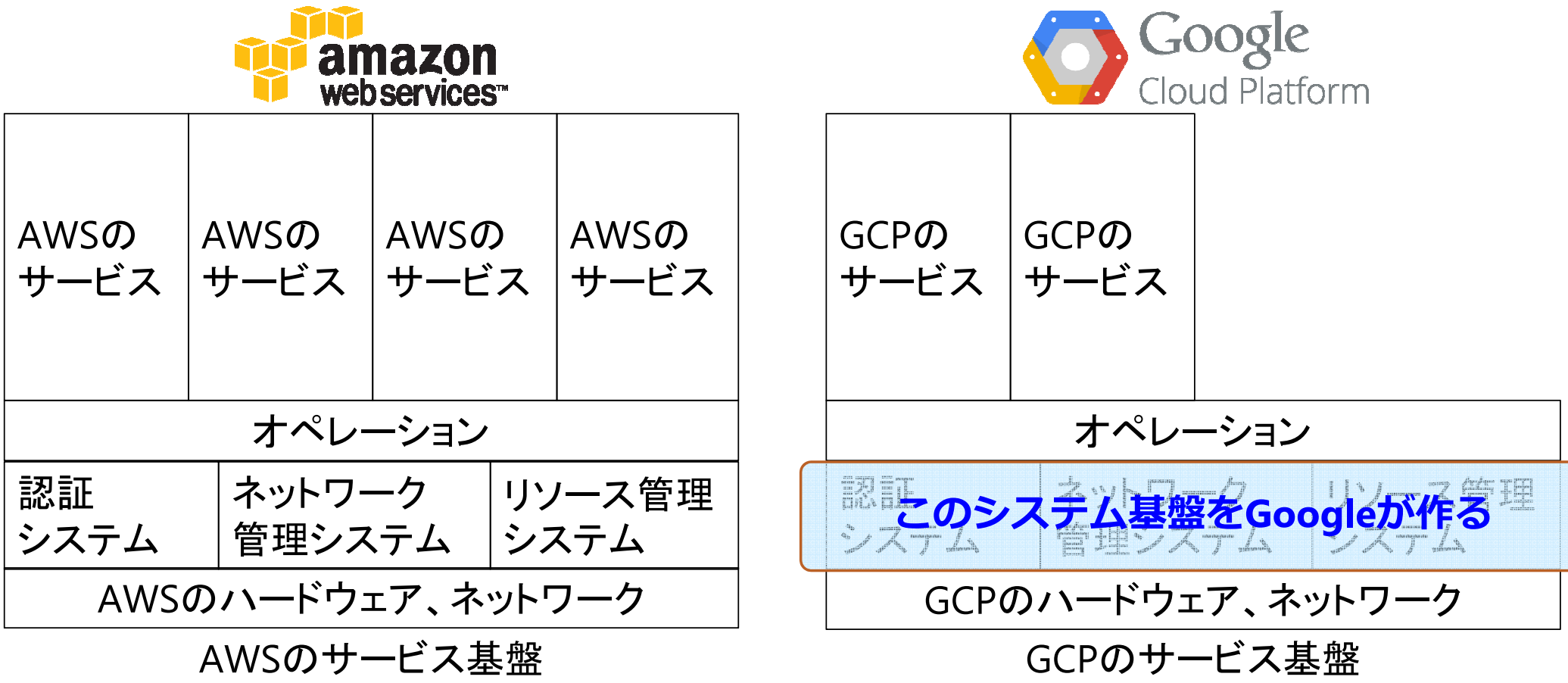
AWSの サービス	AWSの サービス	AWSの サービス	AWSの サービス
オペレーション			
認証 システム	ネットワーク 管理システム	リソース管理 システム	
AWSのハードウェア、ネットワーク			

AWSのサービス基盤

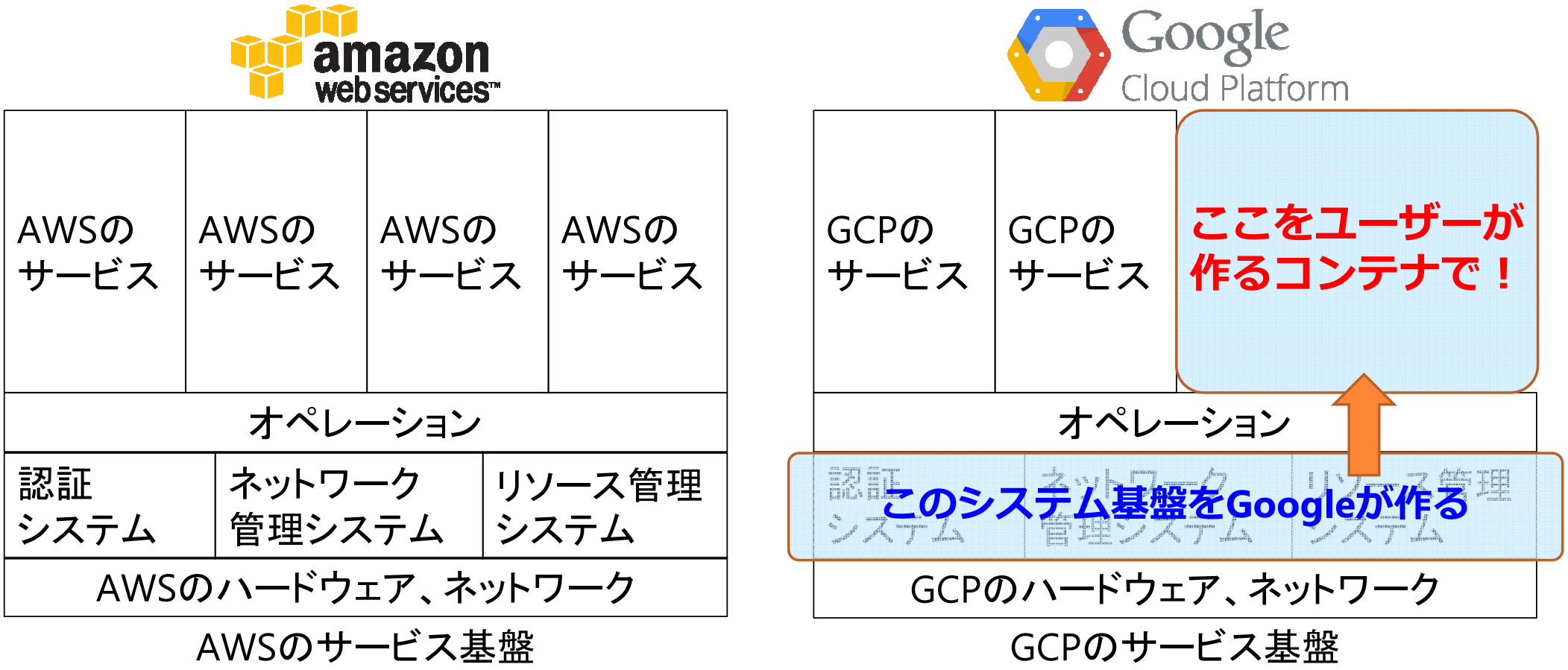
GCPの サービス	GCPの サービス	**ここをユーザーに 作って欲しい**
オペレーション		
認証 システム	ネットワーク 管理システム	リソース管理 システム
GCPのハードウェア、ネットワーク		

GCPのサービス基盤

ゲームチェンジするために必要なもの



ゲームチェンジするために必要なもの



Googleの狙い（3段論法）

矢野の想像です

- 1. Googleがクラウドのような自動管理される
  サービスプラットフォーム環境を作る
- 2. プラットフォームで、サービス部分は
  ユーザー自身に作ってもらうことができる
- 3. サービスをGoogleが作る必要がなくなる

コンテナーが十分に信頼性の高い環境で動かせるなら、
それはもうサービスと同じことでは？
コンテナーならそれが可能！

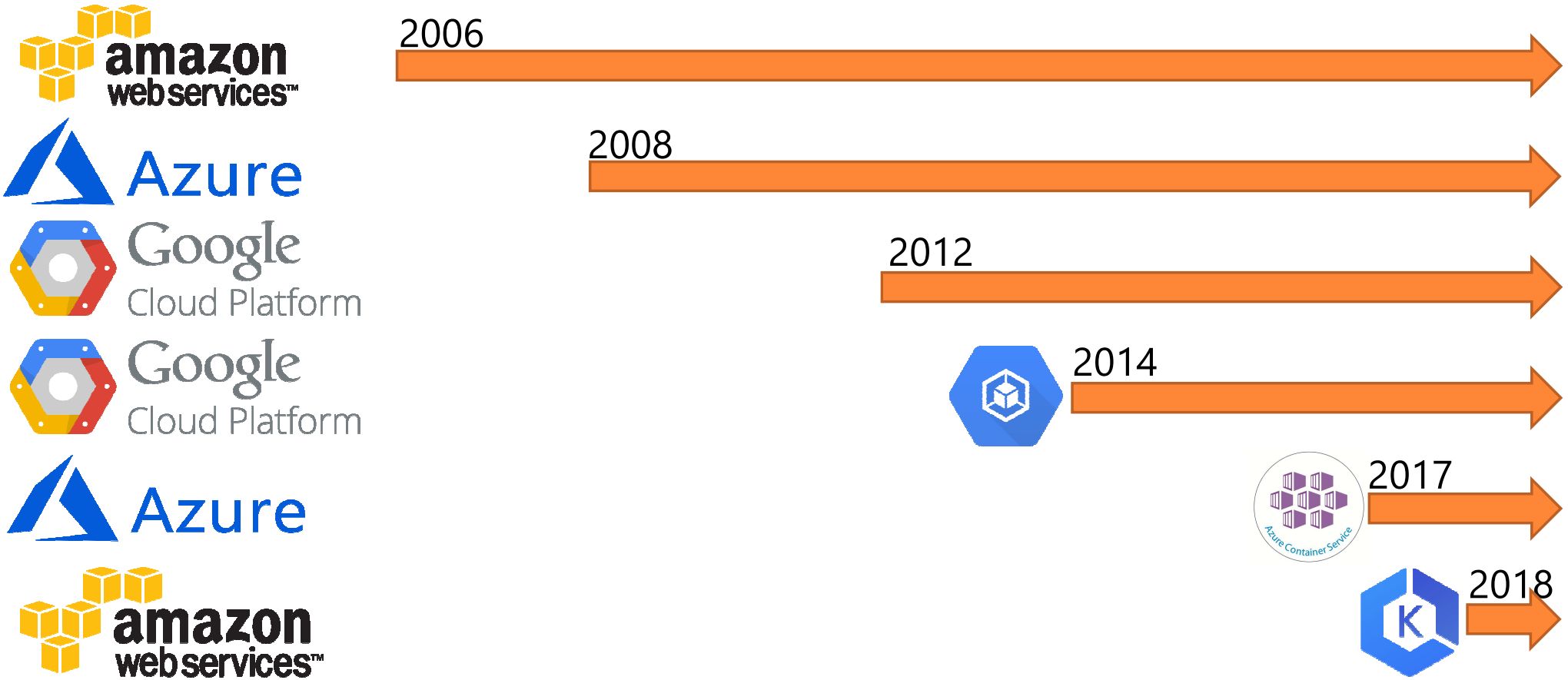
クラウドの歴史

2006

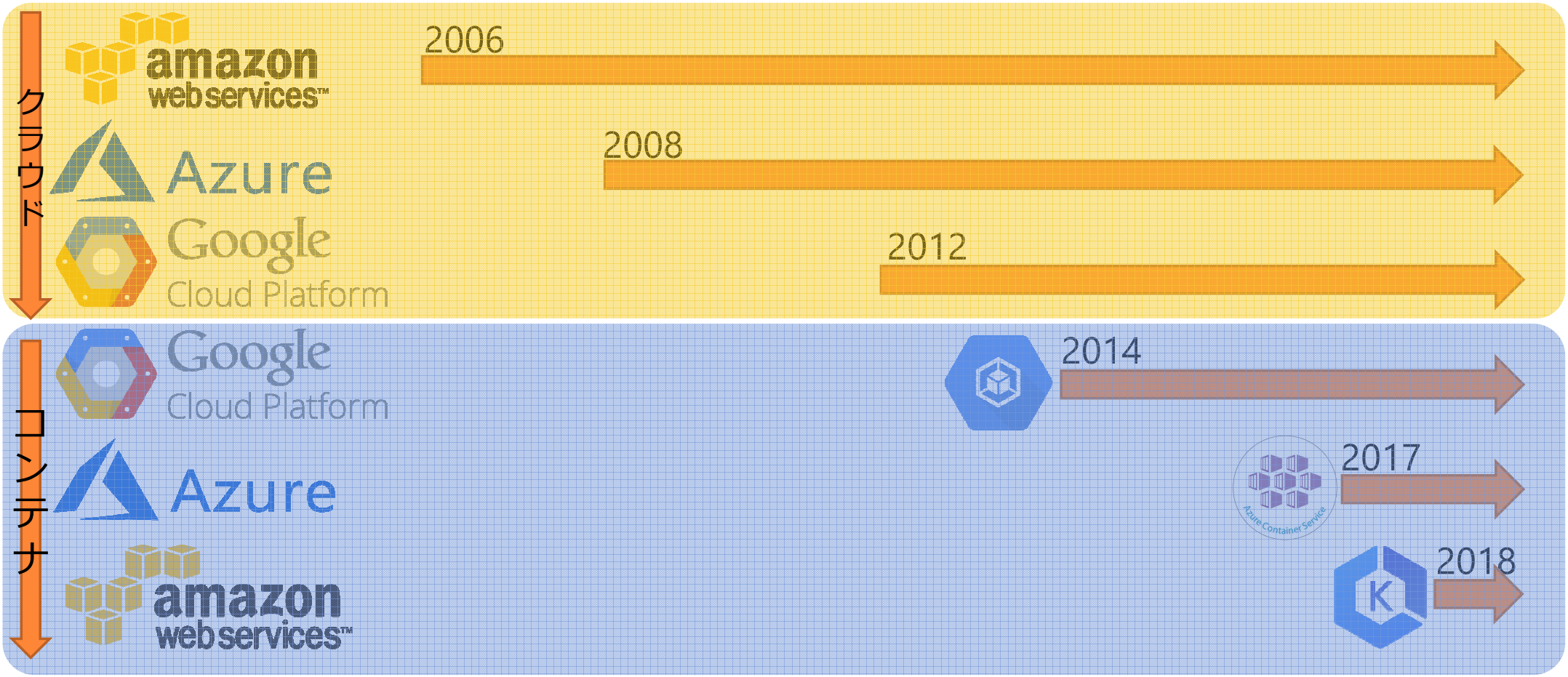
2008

2012

クラウドの歴史→コンテナー (KaaS) の歴史



クラウドの歴史→コンテナー (KaaS) の歴史



Googleのゲームチェンジが成功しつつある

SIerが考えなければならない影響範囲

本日のお話の構成

- コンテナーのメリット
- Kubernetesとはなんなのか
- KubernetesはGoogleがAWSに仕掛けたゲームチェンジ
- Kubernetesによって変わるSIerの役割

何が変わる？

- OSが変わる
- 開発が変わる
- デバッグが変わる
- デプロイが変わる
- リリース方法が変わる
- インフラが変わる
- アプリ設定方法が変わる
- 冗長構成方法が変わる
- ネットワークが変わる
- セキュリティが変わる
- ジョブ実行が変わる
- テストが変わる
- ログ収集が変わる
- 監視・モニタリングが変わる
- バックアップが変わる

ゲームチェンジ

全てをKubernetes中心に考える

ゲームチェンジ

SIerにとってのメリット

SIerにとってのメリット

AWSに奪われた市場を取り戻す
最後のチャンス

IBMとMicrosoftはそう考えてKubernetesを
一生懸命やっていると思います

何を勉強すればよいか？3つのレイヤー

gitの基礎

コンテナの基礎

CIビルドの基礎

YAMLファイル形式

Kubernetesネットワーク

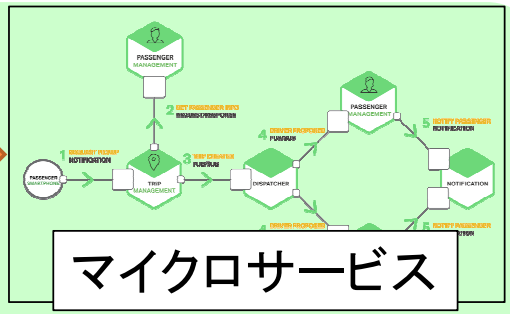
Kubernetesストレージ

Kubernetesパッケージング

クラウドデザインパターン

THE TWELVE-FACTOR APP

12Factor App



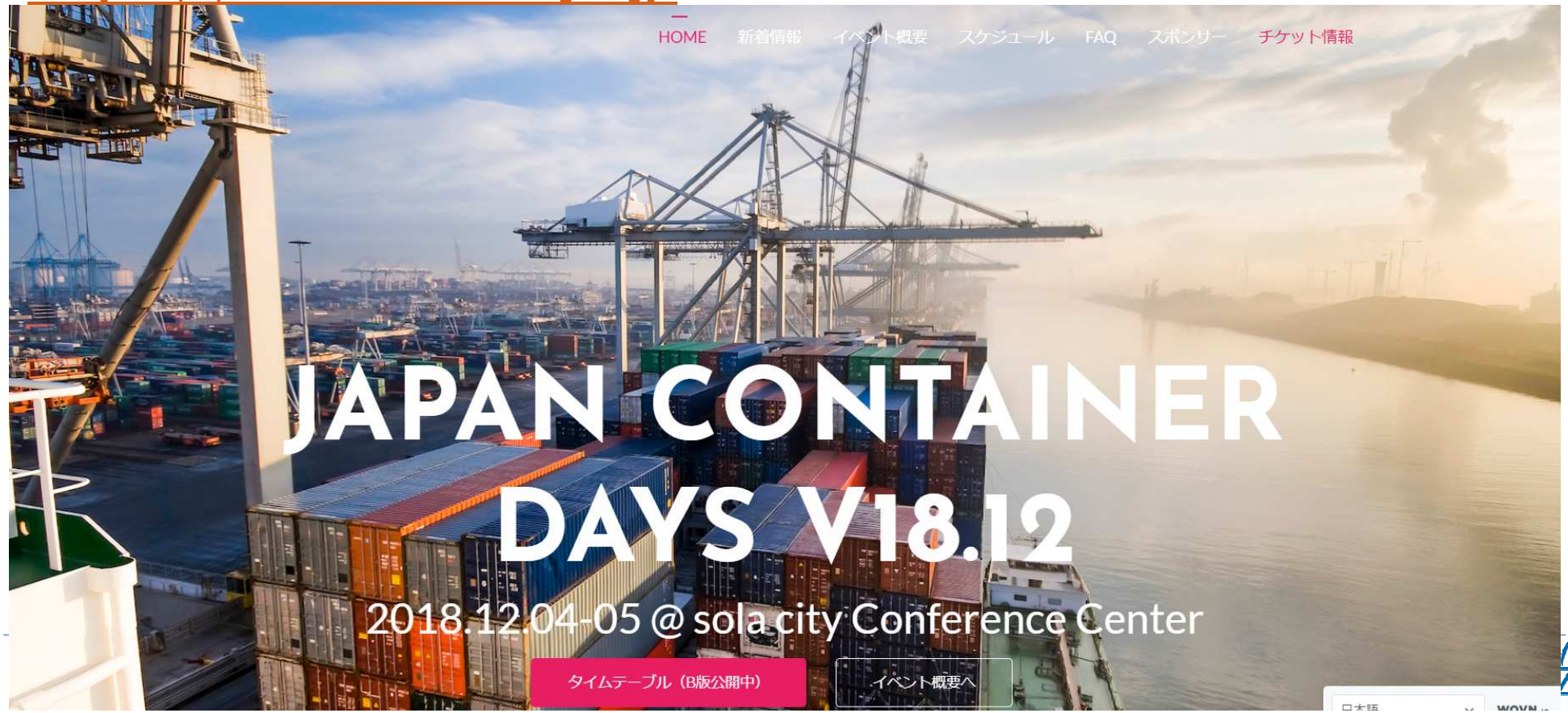
マイクロサービス


結構大変なことが一杯
しかし、Kubernetesは
必ずやってきます
今から始めましょう

Japan Container Days v18.12 (12月4日、5日)

- Kubernetesの先達の貴重な話が聞けるイベントです

https://containerdays.jp



ご清聴
ありがとうございました。