



公開版

# SlerにおけるKubernetes活用



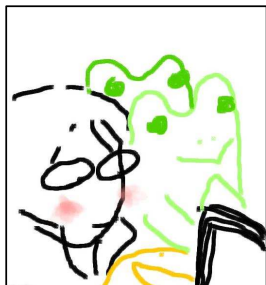
株式会社スタイルズ  
矢野 哲朗



2018年11月7日

# 自己紹介

---



矢野 哲朗

 [tetsurow.yano](https://www.facebook.com/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

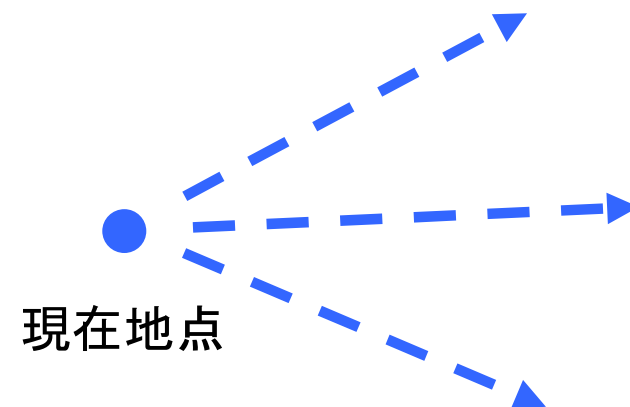
---



現在地点

# 未来を知る

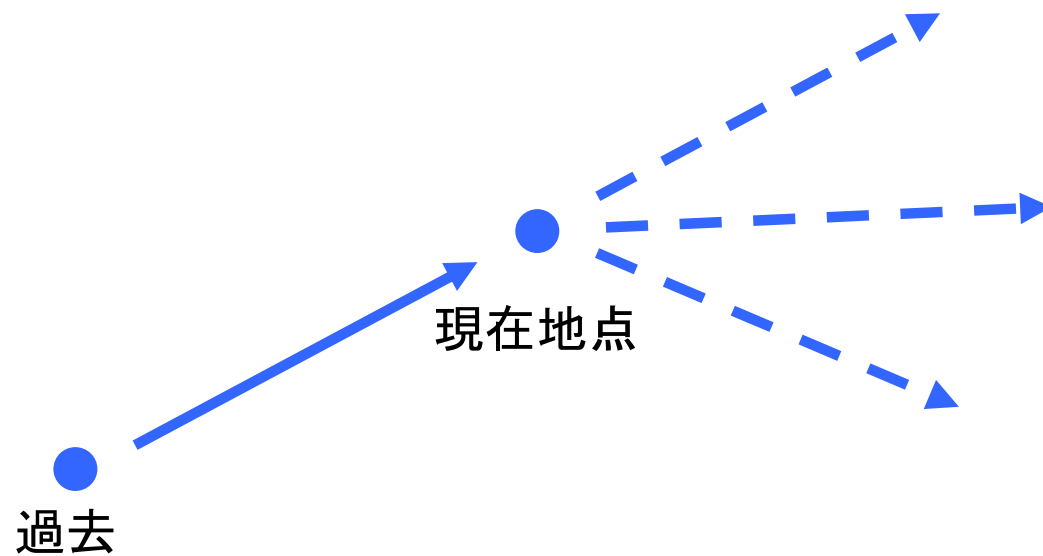
---





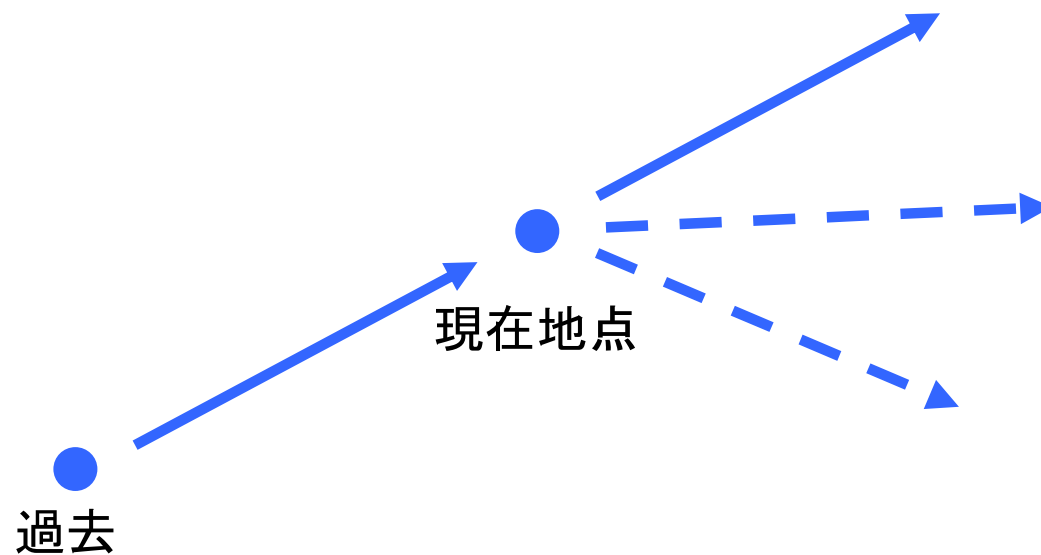
# 過去を知る

---



# 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  
Sler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？



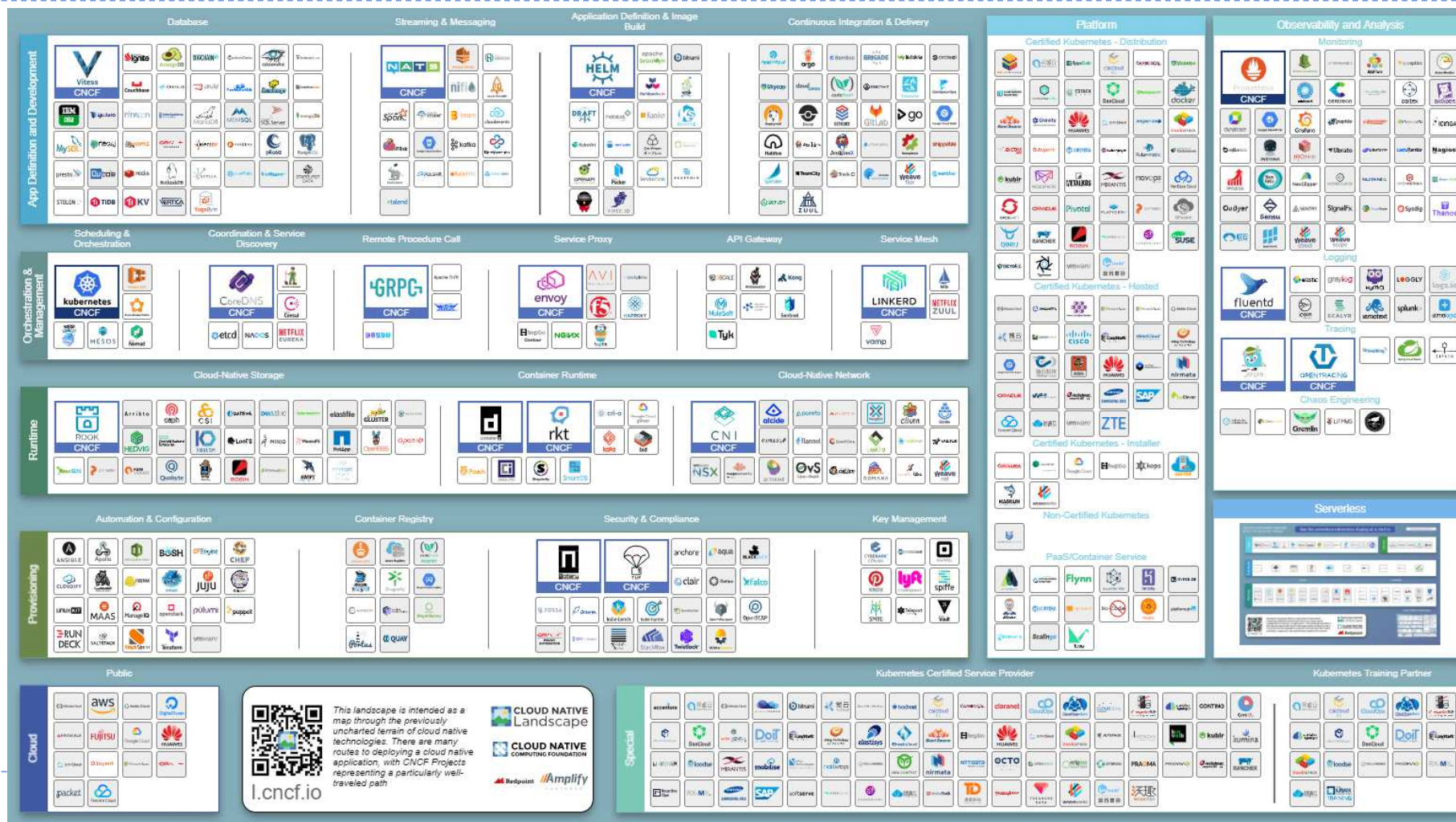
Dockerもダメじゃありません

---

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



**Kubernetes**

@kubernetesio

フォローする



'Kubernetes is becoming the Linux of the cloud' - Jim Zemlin, Linux Foundation  
[#GoogleNext17](#)

🌐 ツイートを翻訳

2:47 - 2017年3月11日

238件のリツイート 297件のいいね



KubernetesはクラウドのLinuxになりつつある

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



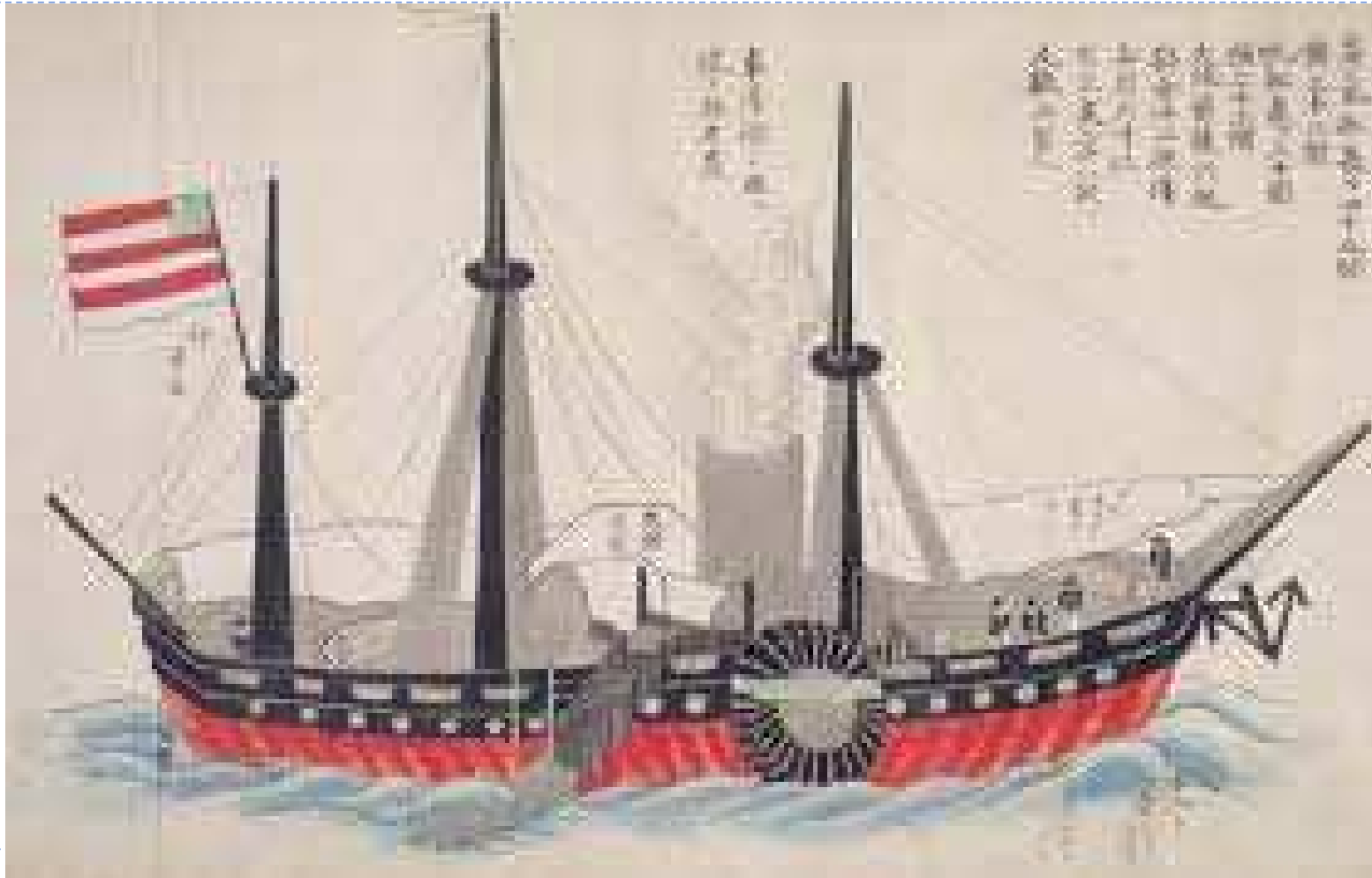
<https://gihyo.jp/book/2018/978-4-297-10033-9>

## 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

# クーバー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](#) ]

κυβερνάω (*kubernáō*, “to steer”) + -της (*-tēs*)

**Pronunciation** [ [edit](#) ]

- IPA(<sup>key</sup>): /ky.ber.nɛi.tɛɪs/ → /ky.berˈniː/

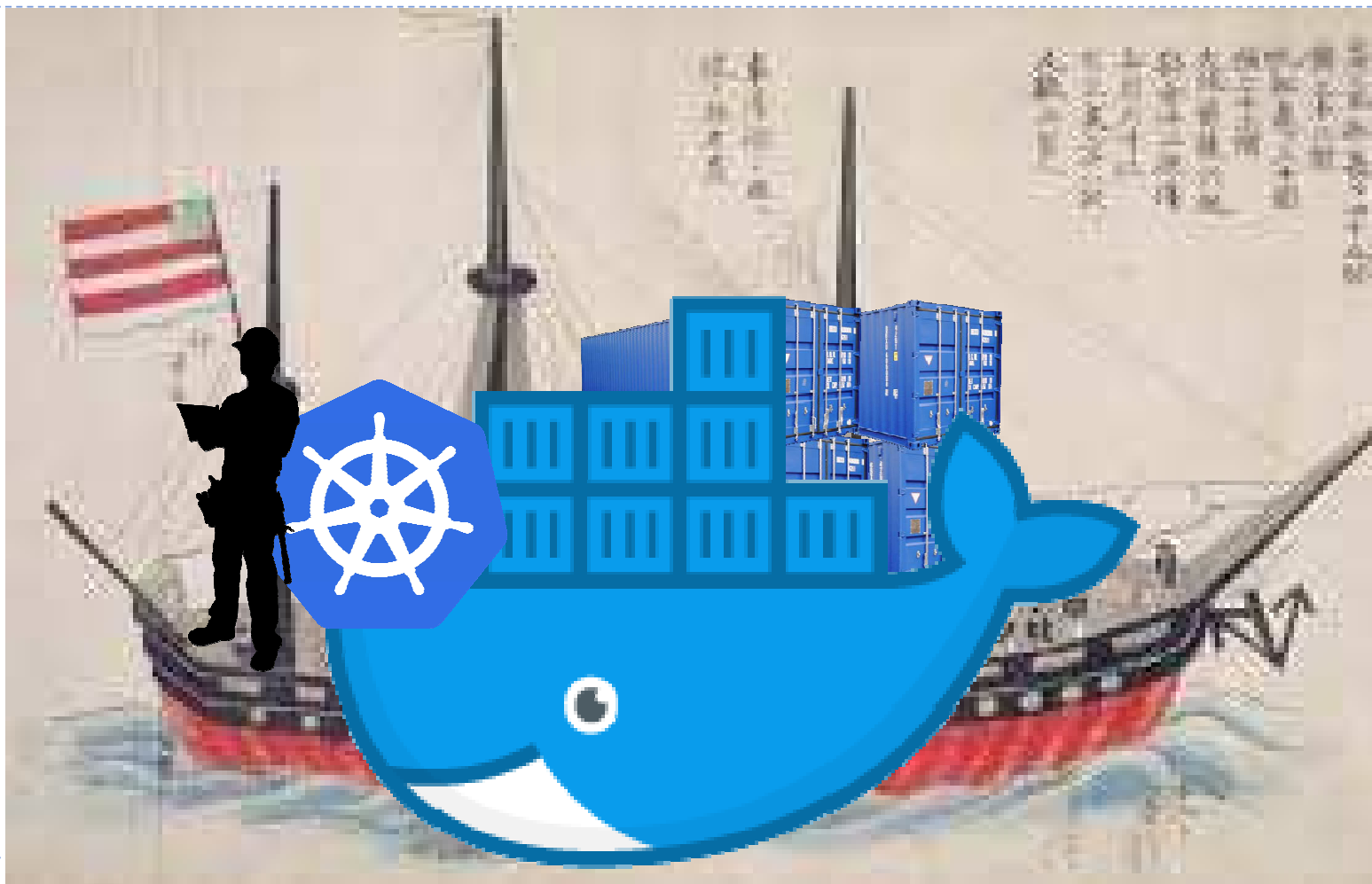
**Noun** [ [edit](#) ]

κυβερνήτης • (kubernḗtēs) *m* (genitive: κυβερνήτου); *first declension* (*Epic, Attic, Ionic, Koine*)

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

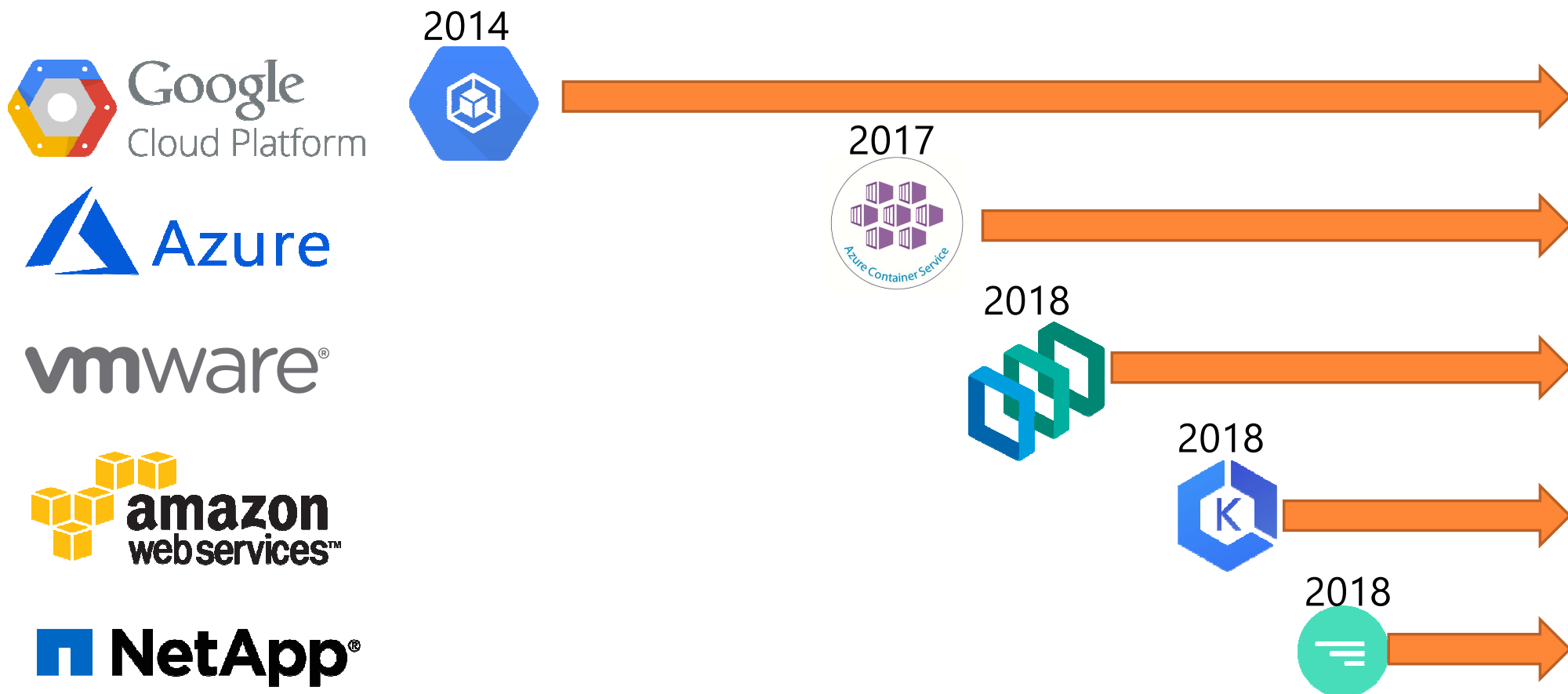
船長、総舵手、パイロット、航海手、ガイド

# Kubernetesは黒船



# 押し寄せる“KaaS”の波

  
kubernetes **as a Service**





---

## コンテナのメリット

# 本日のお話の構成

---

□コンテナのメリット

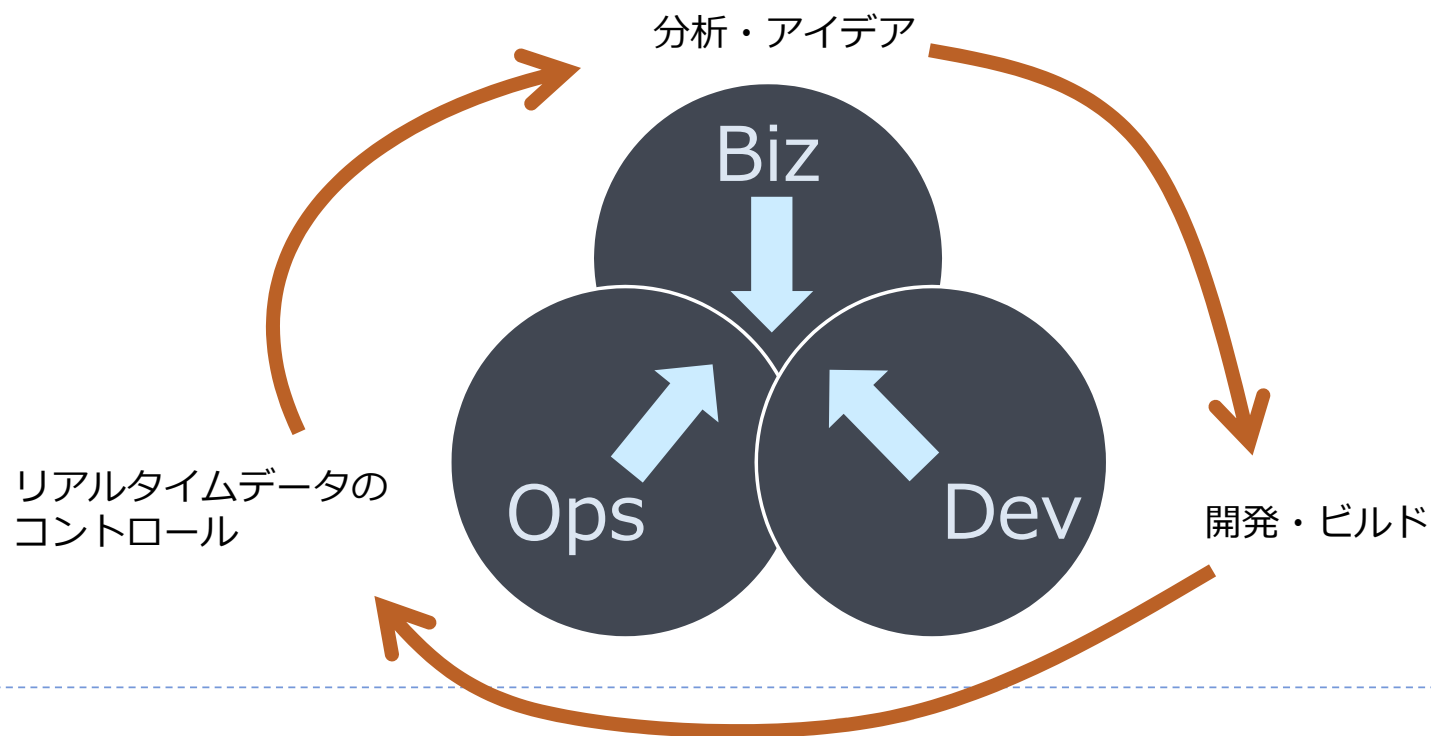
□Kubernetesとはなんなのか

□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  
Slerの役割

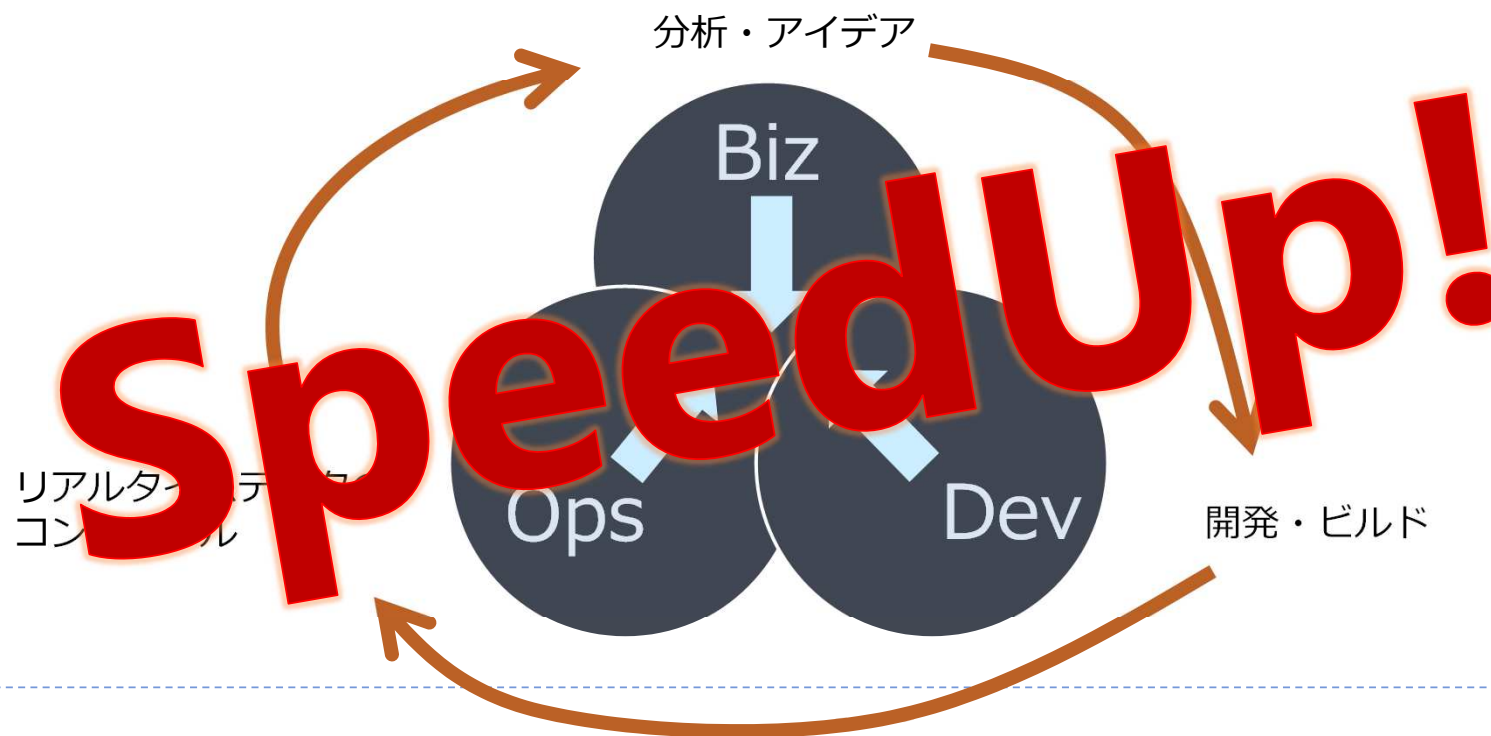
# デジタル時代に求められるビジネスモデル

- 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



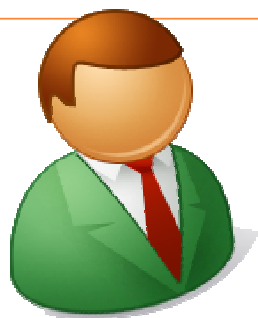
# よく聞かれるコンテナ利用の声

## □ 経営層の声

- ・コンテナ何それ？
- ・そんなの無くてもいいよね。
- ・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 (Docker) のメリット

---

元々は開発者が楽になるように  
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

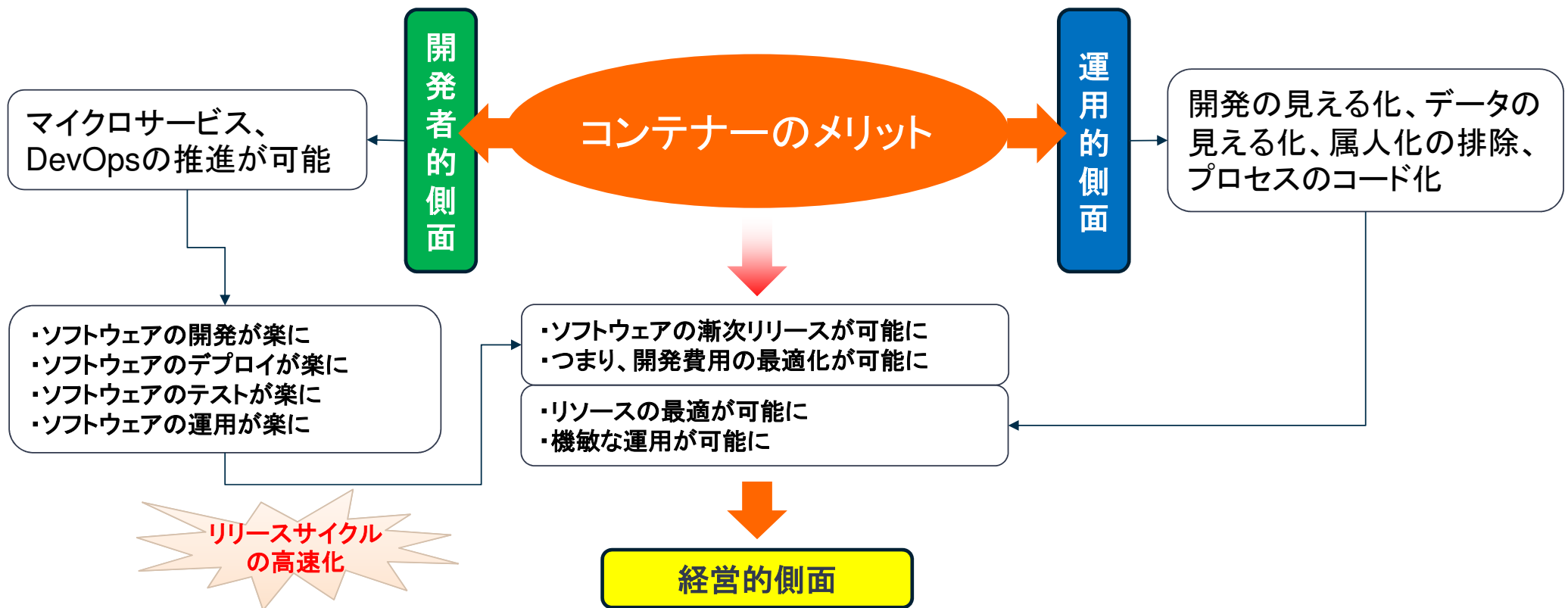
開発者のメリット

ビジネスのメリット

とても深く関連している

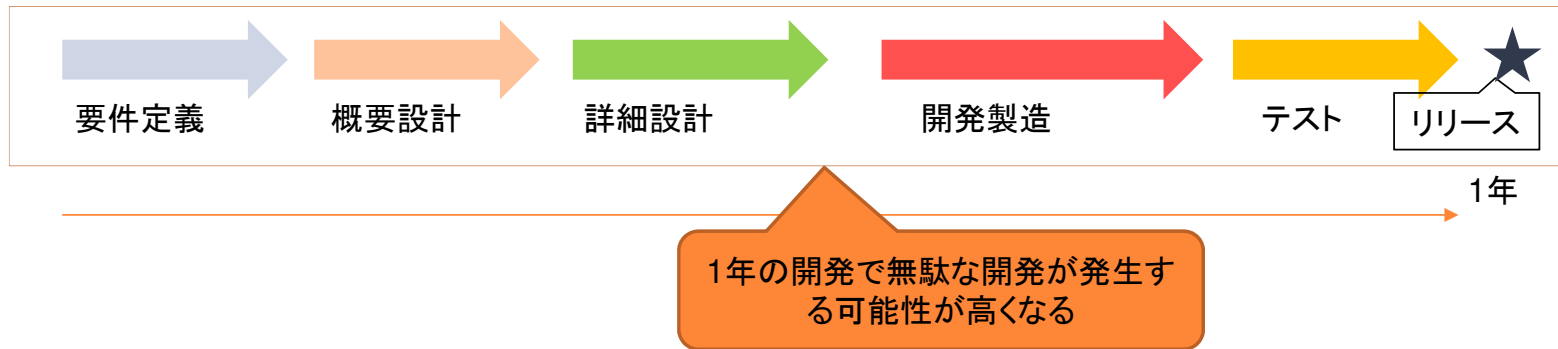
## コンテナは利益に直結するわけではありません

コンテナ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

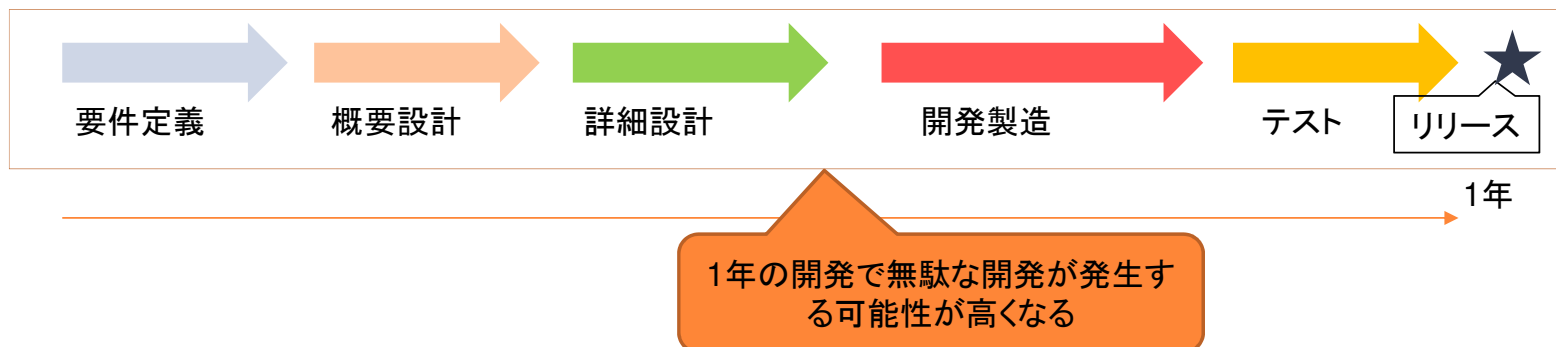
## □ これまでのアプリケーション開発スパン・サイクル



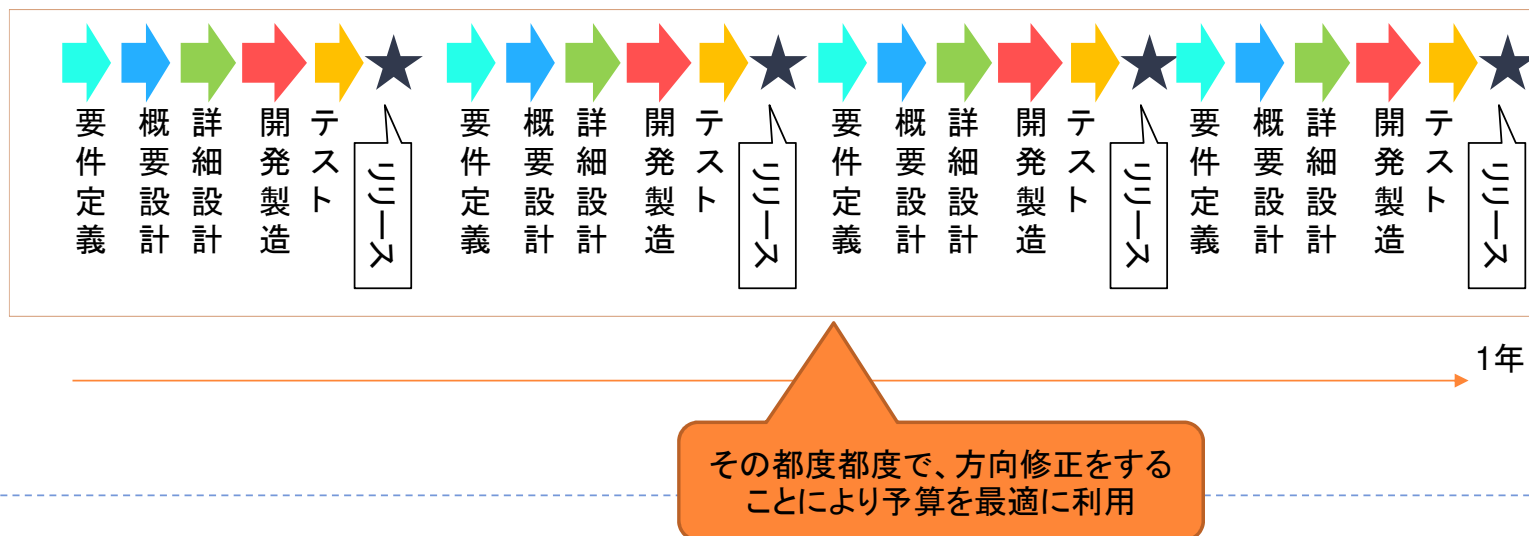


# リリースサイクル短縮化の効用

## □ これまでのアプリケーション開発スパン・サイクル



## コンテナを利用したアプリケーション開発スパン



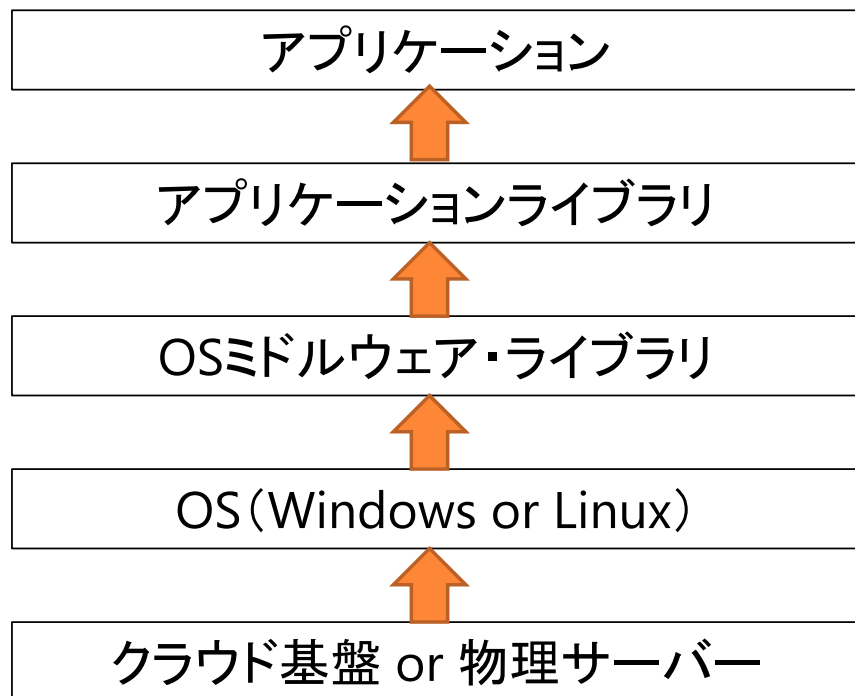
---

**まずはコンテナの仕組みを理解しましょう**

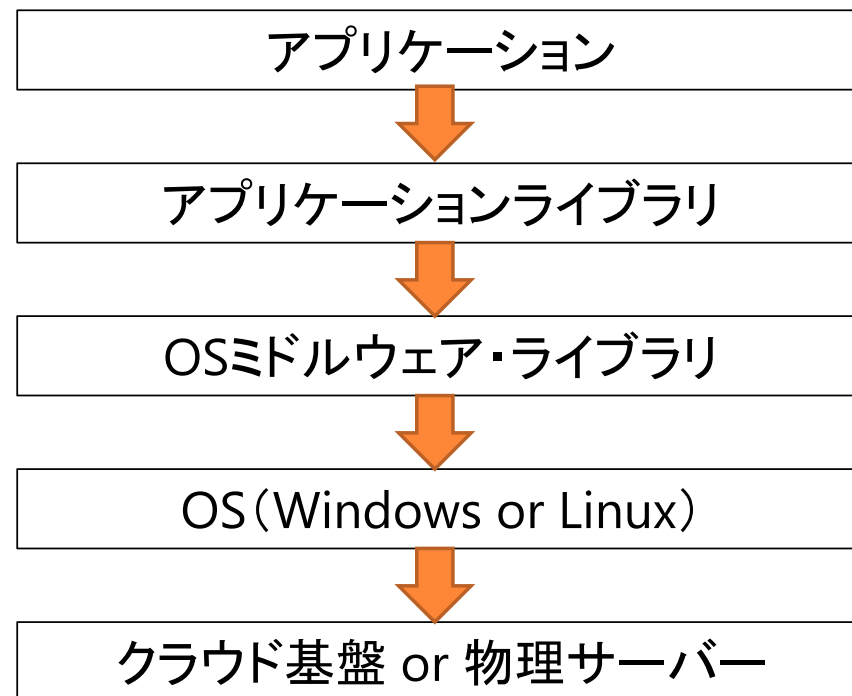
# 物理 (or 仮想) マシンとコンテナ

仮想マシン(クラウド含む)とコンテナ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



コンテナでの考え方



# コンテナの考え方

仮想マシン(クラウド含む)とコンテナ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

コンテナに入っているのはここまで

ここはコンテナ基盤

コンテナでの考え方

アプリケーション

# コンテナの考え方

仮想マシン(クラウド含む)とコンテナ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

コンテナに入っているのはここまで

ここはコンテナ基盤

コンテナでの考え方

アプリケーション



アプリケーションライブラリ

# コンテナの考え方

仮想マシン(クラウド含む)とコンテナ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

コンテナに入っているのはここまで

ここはコンテナ基盤

コンテナでの考え方

アプリケーション

アプリケーションライブラリ

OSミドルウェア・ライブラリ

# コンテナの考え方

仮想マシン(クラウド含む)とコンテナ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

コンテナに入っているのはここまで

ここはコンテナ基盤

コンテナでの考え方

アプリケーション

アプリケーションライブラリ

OSミドルウェア・ライブラリ

OS (Windows or Linux)

# コンテナの考え方

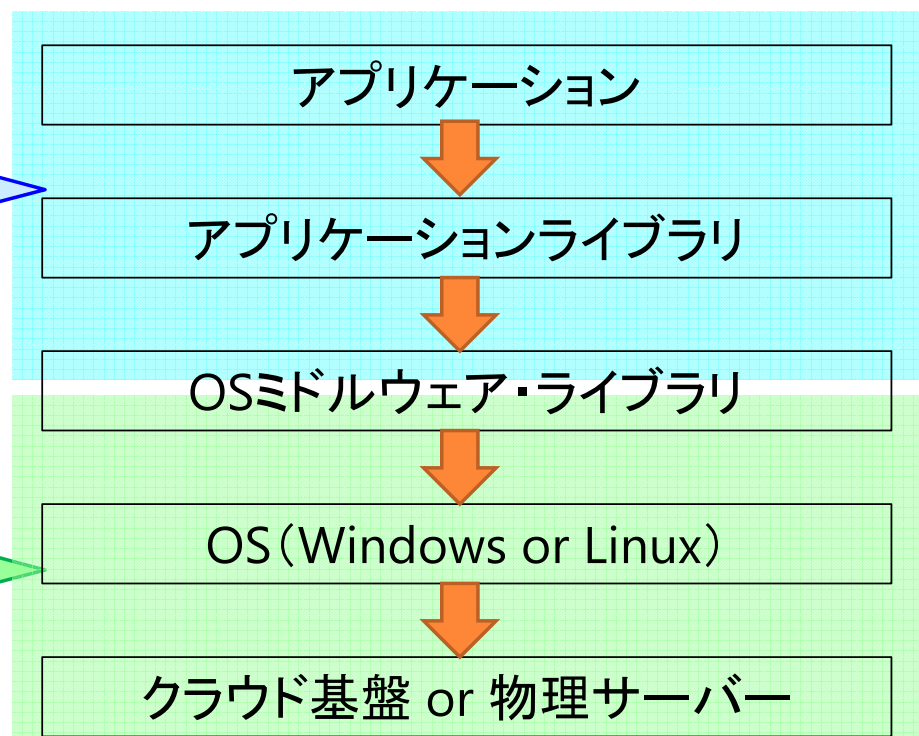
仮想マシン(クラウド含む)とコンテナは全く正反対の考え方でできています

物理(or 仮想)マシンでの考え方

コンテナに入っているのはここまで

ここはコンテナ基盤

コンテナでの考え方

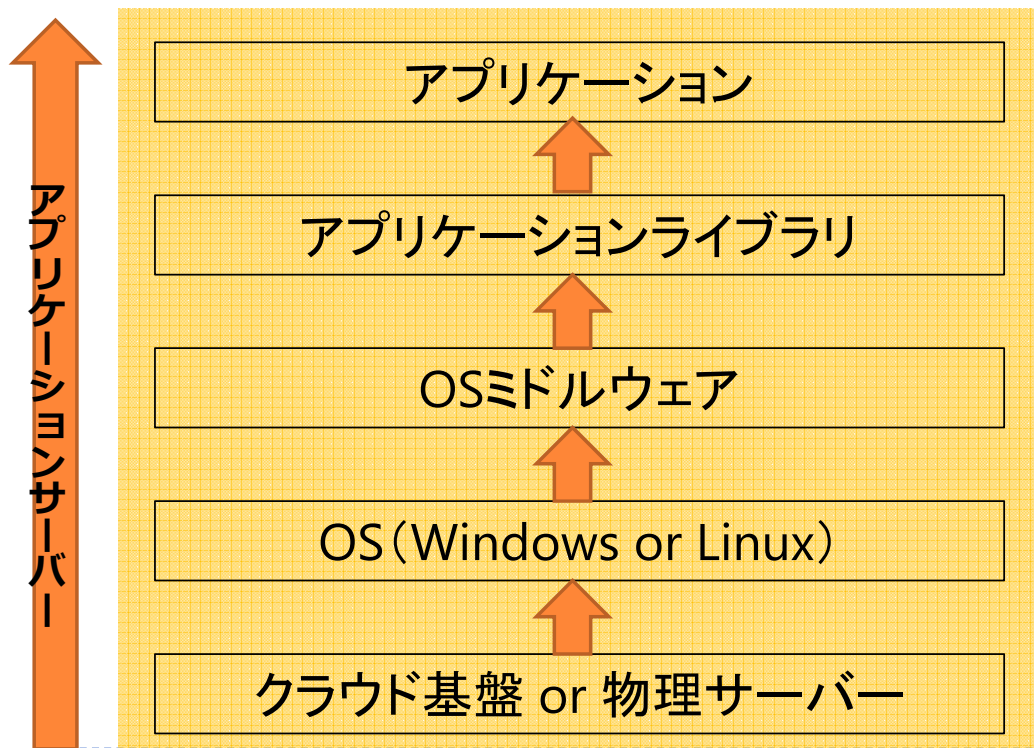




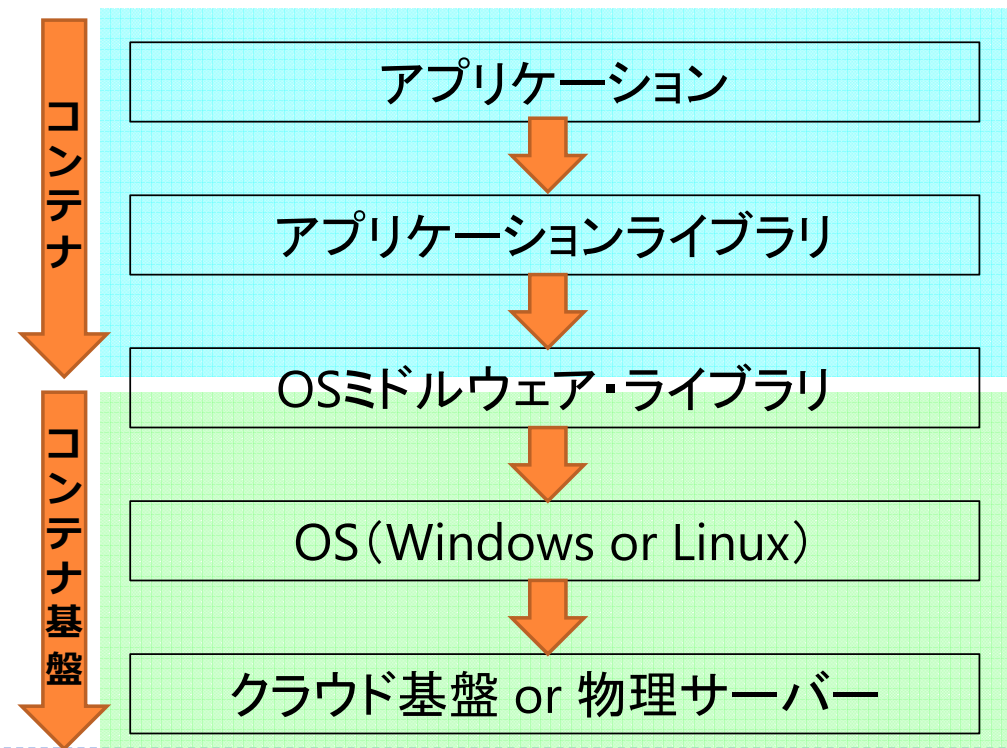
# 物理 (or 仮想) マシンとコンテナ

仮想マシン(クラウド含む)とコンテナ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



コンテナでの考え方



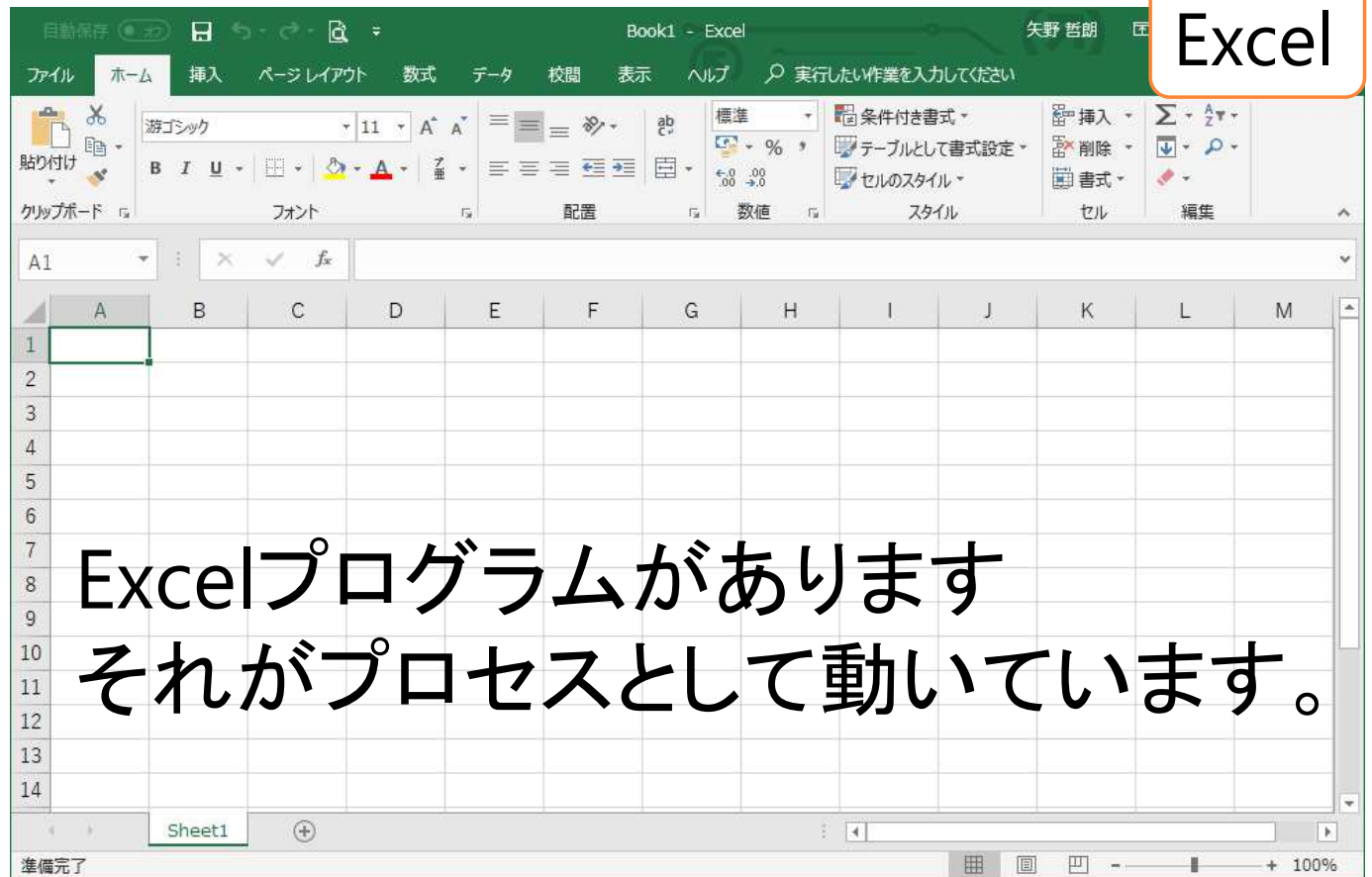
## 具体的な例で考えてみましょう

---

※注意: こういうものがあるわけではありません  
例えです

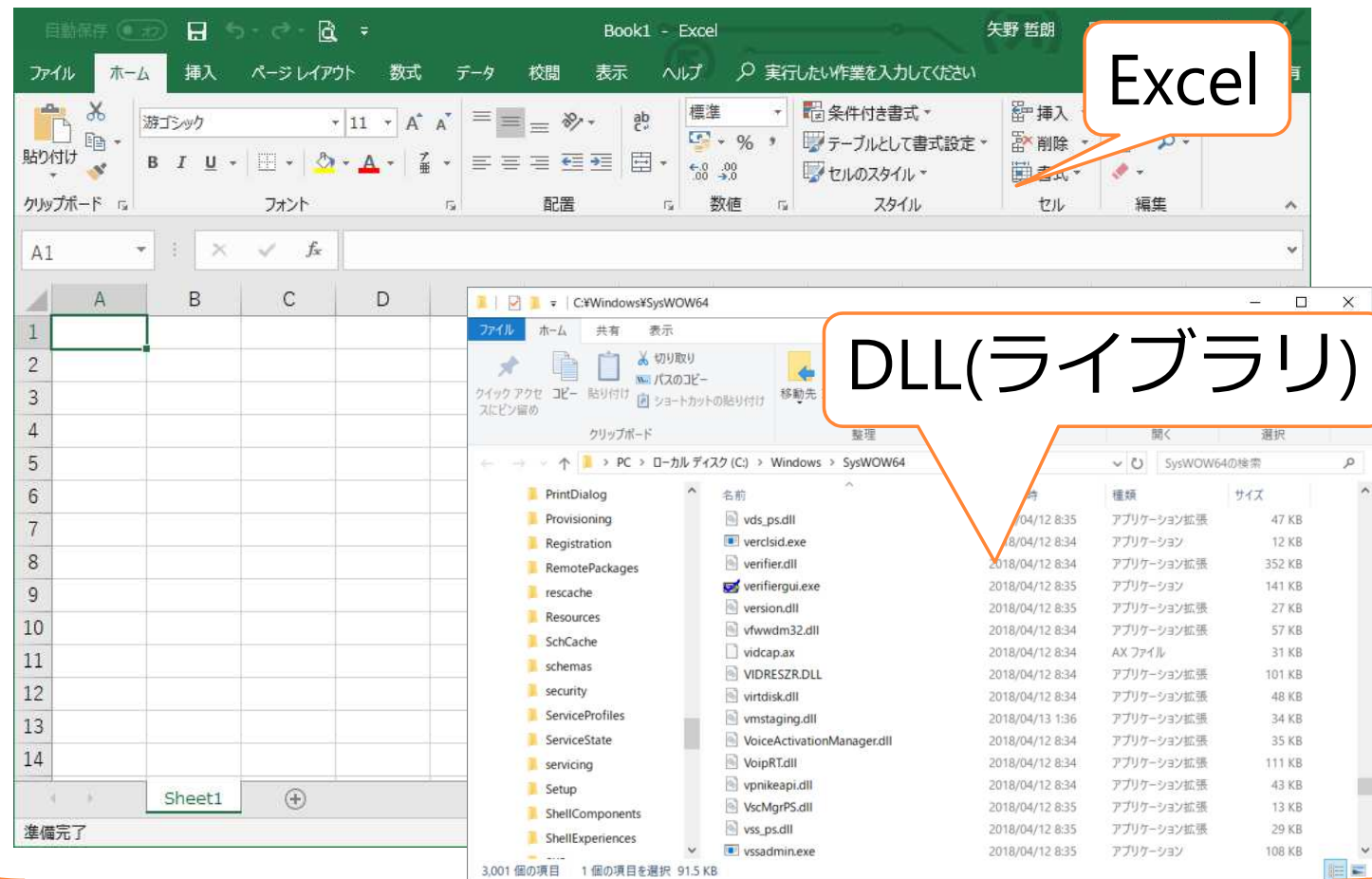
例えば、Excelアプリが  
コンテナー化されてい  
たら？

# 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

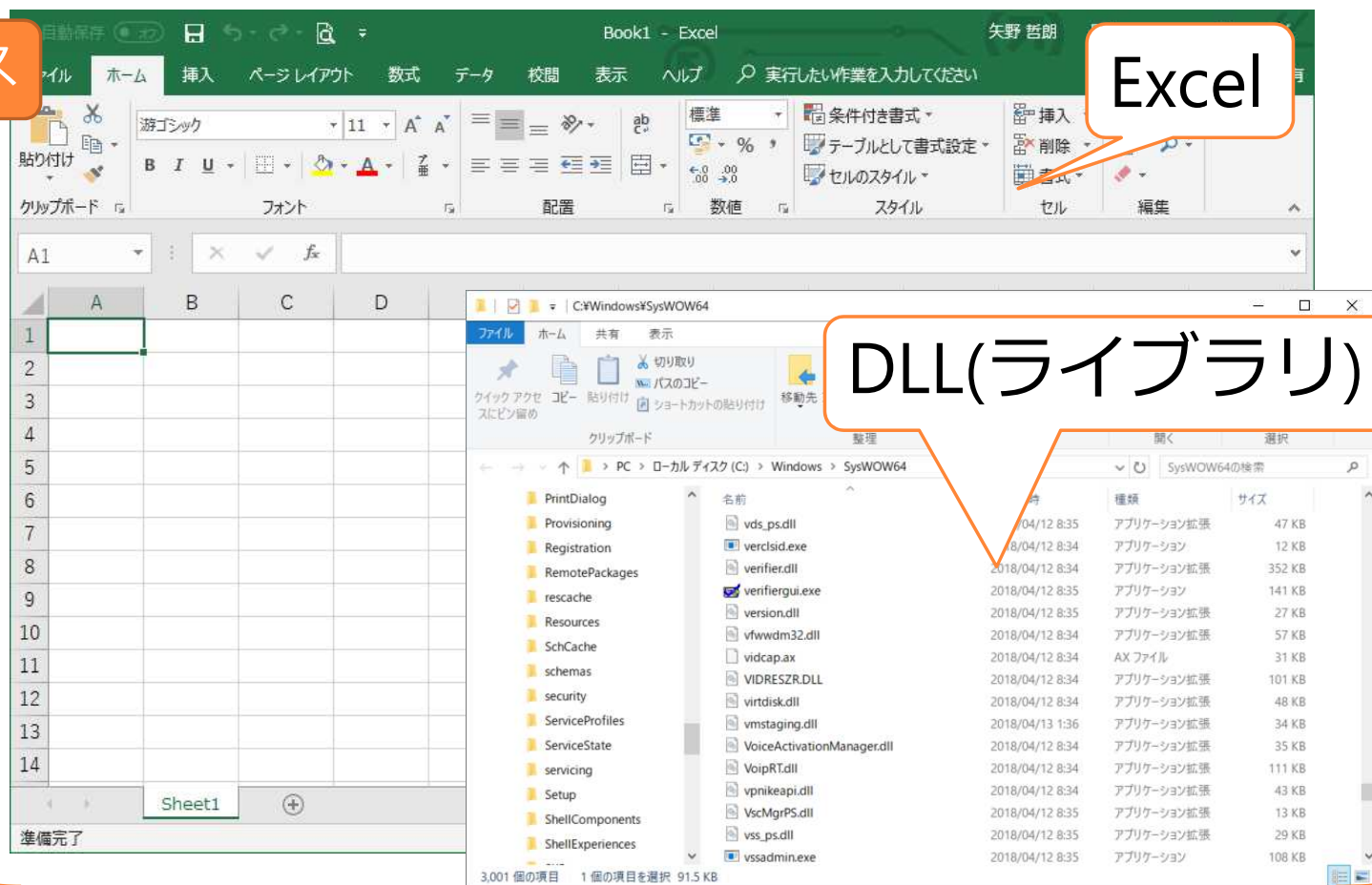


# このプロセスにはIPアドレスがついています



IPアドレス

Excel



The image shows two overlapping screenshots. The background is a screenshot of the Microsoft Excel application, titled 'Book1 - Excel' by '矢野 哲朗'. The ribbon is set to 'ホーム' (Home), and the '数値' (Number) group is visible. The active cell is A1. Overlaid on the bottom right is a screenshot of a Windows File Explorer window showing the directory 'C:\Windows\SysWOW64'. The file list includes various system files and executables. A table of files is shown below the list:

| 名前                         | 種類         | サイズ    |
|----------------------------|------------|--------|
| vds_ps.dll                 | アプリケーション拡張 | 47 KB  |
| verclsid.exe               | アプリケーション   | 12 KB  |
| verifier.dll               | アプリケーション拡張 | 352 KB |
| verifiergui.exe            | アプリケーション   | 141 KB |
| version.dll                | アプリケーション拡張 | 27 KB  |
| vfwvdm32.dll               | アプリケーション拡張 | 57 KB  |
| vidcap.ax                  | AX ファイル    | 31 KB  |
| VIDRESZR.DLL               | アプリケーション拡張 | 101 KB |
| virtdisk.dll               | アプリケーション拡張 | 48 KB  |
| vmstaging.dll              | アプリケーション拡張 | 34 KB  |
| VoiceActivationManager.dll | アプリケーション拡張 | 35 KB  |
| VoipRT.dll                 | アプリケーション拡張 | 111 KB |
| vpniapi.dll                | アプリケーション拡張 | 43 KB  |
| VscMgrPS.dll               | アプリケーション拡張 | 13 KB  |
| vss_ps.dll                 | アプリケーション拡張 | 29 KB  |
| vssadmin.exe               | アプリケーション   | 108 KB |

DLL(ライブラリ)

## このプロセスにはDiskがついている



# IPアドレス

# Excel

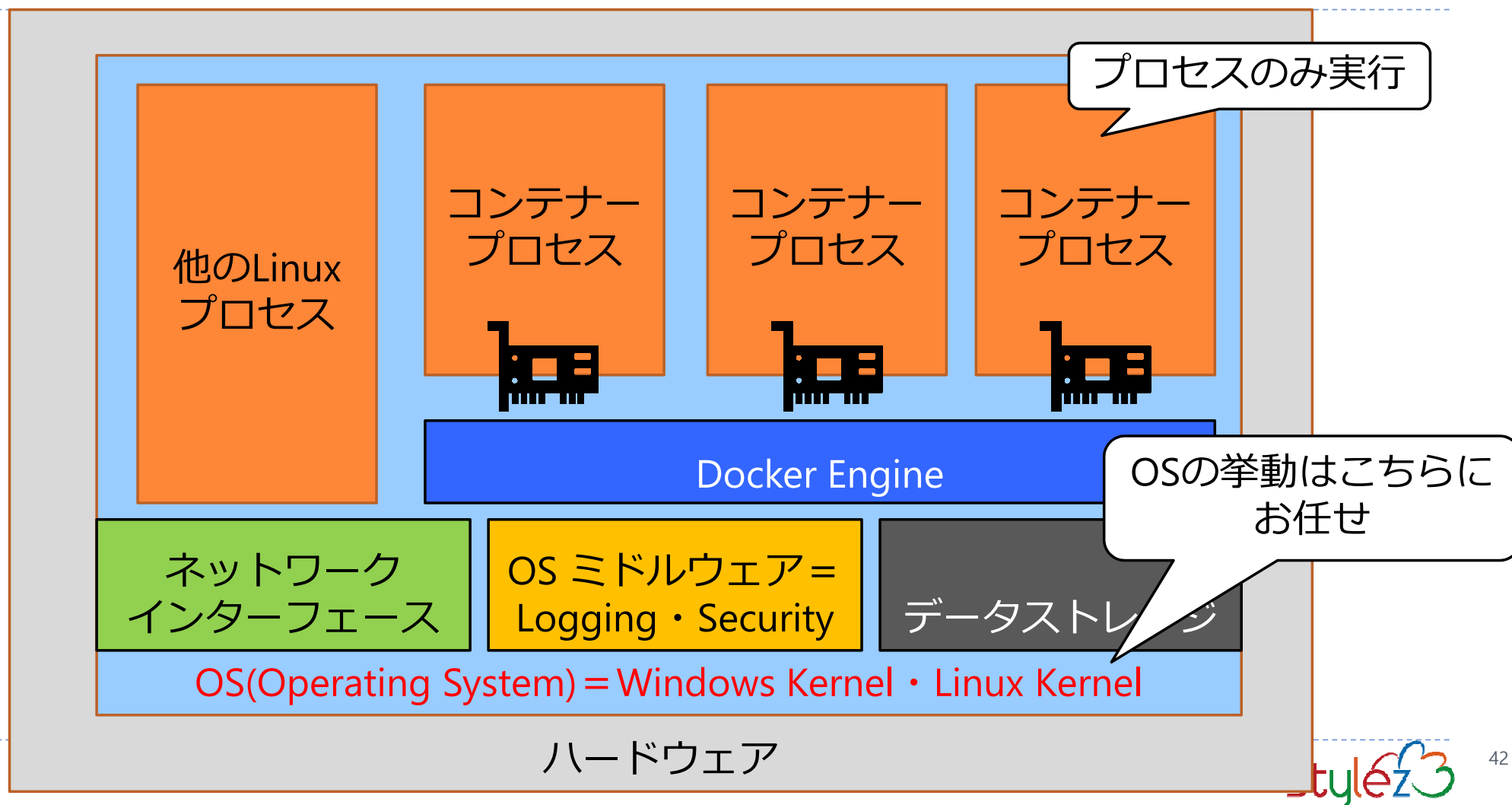
## DLL(ライブラリ)

# Disk

ここにはOSが  
出てきません



# コンテナのプロセス



## 結局、コンテナとは？

---

OS部分をコンテナ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？

```
ubuntu@ip-172-31-0-90:~$  
ubuntu@ip-172-31-0-90:~$  
ubuntu@ip-172-31-0-90:~$ sudo docker run -it ubuntu /bin/bash
```

```
root@6004a498b5ad:/# ls -la  
total 72  
drwxr-xr-x 34 root root 4096 Oct  1 04:40 .  
drwxr-xr-x 34 root root 4096 Oct  1 04:40 ..  
-rwxr-xr-x  1 root root    0 Oct  1 04:40 .dockerenv  
drwxr-xr-x  2 root root 4096 Aug 21 21:14 bin  
drwxr-xr-x  2 root root 4096 Apr 24 08:34 boot  
drwxr-xr-x  5 root root 360 Oct  1 04:40 dev  
drwxr-xr-x 32 root root 4096 Oct  1 04:40 etc  
drwxr-xr-x  2 root root 4096 Apr 24 08:34 home  
drwxr-xr-x  8 root root 4096 Aug 21 21:12 lib  
drwxr-xr-x  2 root root 4096 Aug 21 21:13 lib64  
drwxr-xr-x  2 root root 4096 Aug 21 21:12 media  
drwxr-xr-x  2 root root 4096 Aug 21 21:12 mnt  
drwxr-xr-x  2 root root 4096 Aug 21 21:12 opt  
dr-xr-xr-x 146 root root    0 Oct  1 04:40 proc  
drwx-----  2 root root 4096 Aug 21 21:14 root  
drwxr-xr-x  5 root root 4096 Sep  5 22:20 run  
drwxr-xr-x  2 root root 4096 Sep  5 22:20 sbin  
drwxr-xr-x  2 root root 4096 Aug 21 21:12 srv  
dr-xr-xr-x 13 root root    0 Oct  1 04:37 sys  
drwxrwxrwt  2 root root 4096 Aug 21 21:14 tmp  
drwxr-xr-x 11 root root 4096 Aug 21 21:12 usr  
drwxr-xr-x 13 root root 4096 Aug 21 21:14 var  
root@6004a498b5ad:/#  
root@6004a498b5ad:/# exit
```

```
exit  
ubuntu@ip-172-31-0-90:~$ █
```

← コンテナを起動

← コンテナを実行中

← コンテナを終了

← 元のプロンプト

# コンテナの最も重要なこと

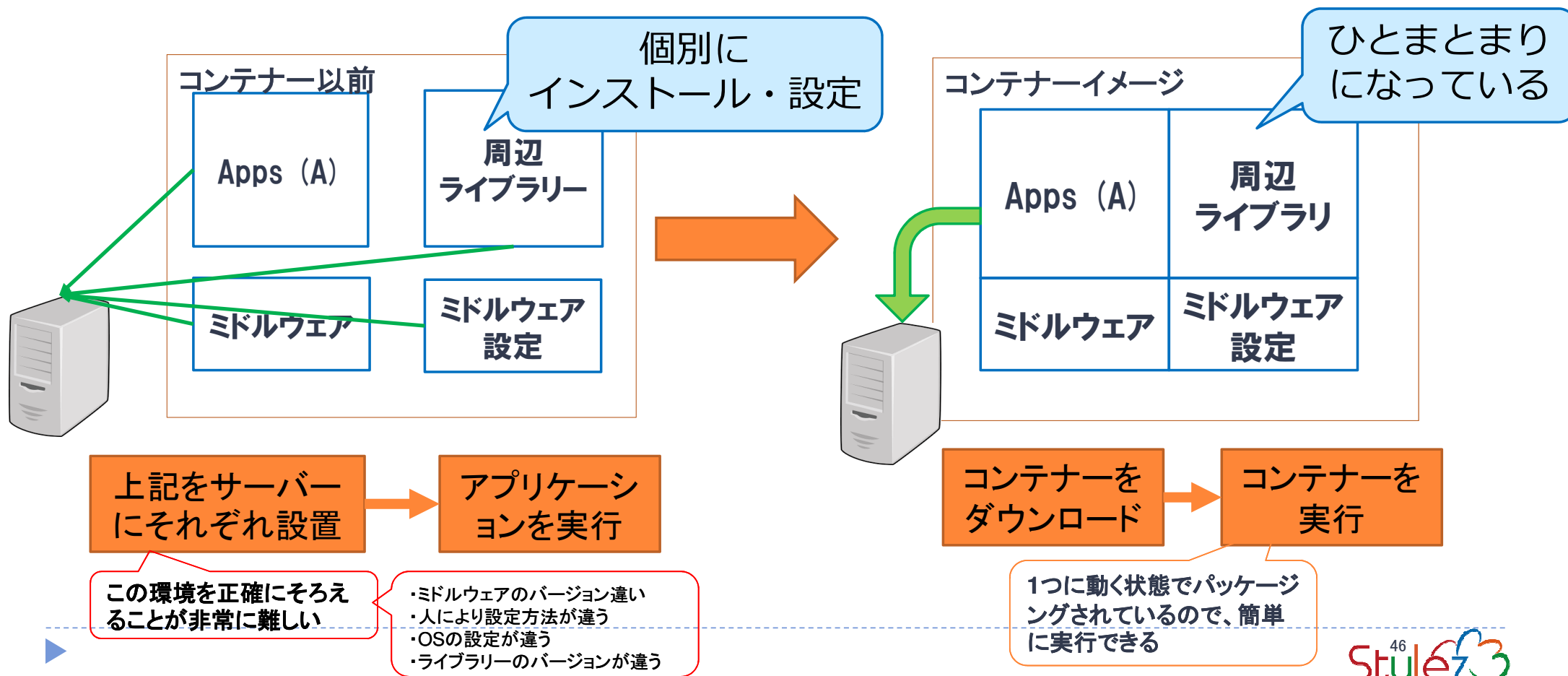
---

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態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

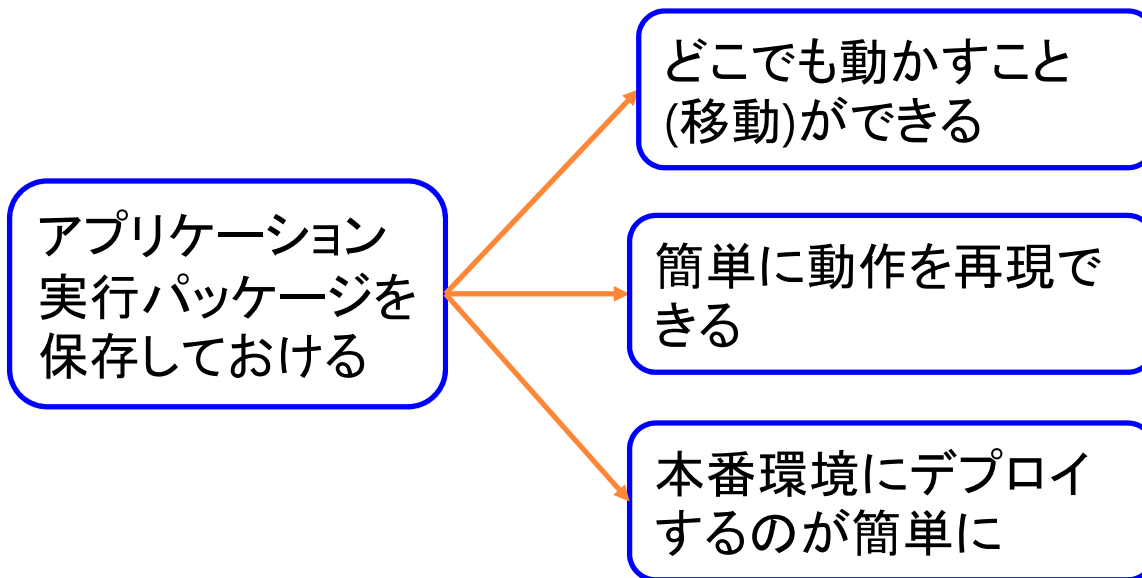
アプリケーション  
実行パッケージを  
保存しておける



# コンテナの最も重要なこと

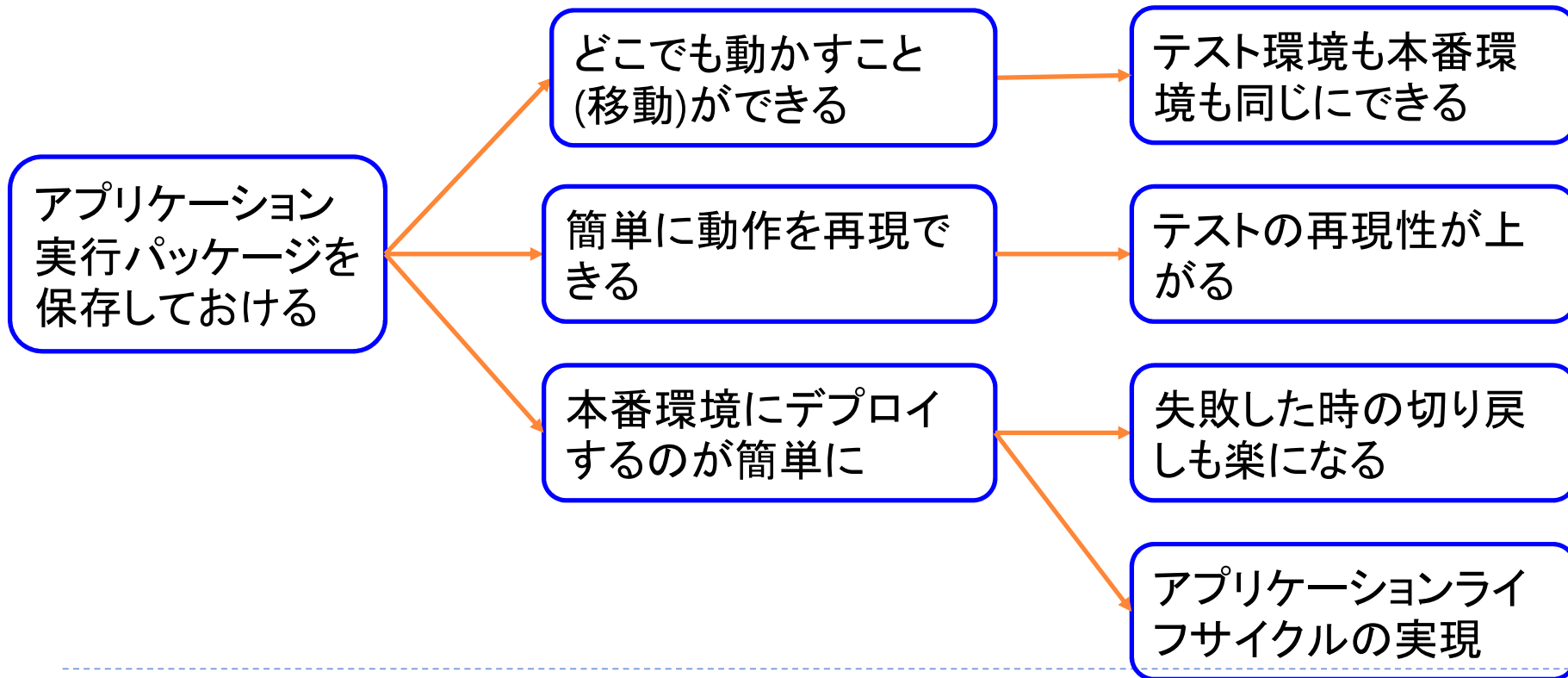
---

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



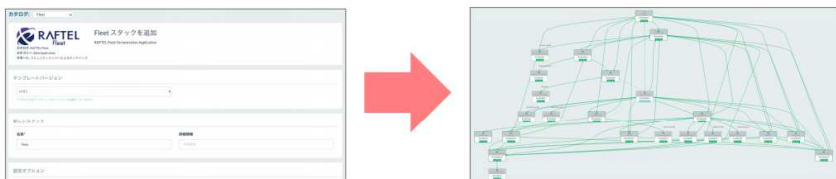
# アプリケーションポータビリティの効果

コンテナ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)

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

## 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

<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。

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。

|                                                                                                                                                                                                                                                        |                                                                                                                                                                                                                                                                |                                                                                                                                                                                                                                  |                                                                                                                                                                                                                                      |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <br><b>magento</b><br>(from Library)<br>A feature-rich flexible e-commerce solution. It includes transaction options, multi-store functionality, loyalty programs,... | <br><b>mariadb</b><br>(from Library)<br>Fast, reliable, scalable, and easy to use open-source relational database system. MariaDB Server is intended for mission-critical,... | <br><b>memcached</b><br>(from Library)<br>Free & open source, high-performance, distributed memory object caching system.                     | <br><b>mongodb</b><br>(from Library)<br>NoSQL document-oriented database that stores JSON-like documents with dynamic schemas, simplifying the... |
| <br><b>mongodb-replicaset</b><br>(from Library)<br>NoSQL document-oriented database that stores JSON-like documents with dynamic schemas, simplifying the...         | <br><b>mysql</b><br>(from Library)<br>Fast, reliable, scalable, and easy to use open-source relational database system.                                                      | <br><b>EXPERIMENTAL</b><br><b>nfs-provisioner</b><br>(from Library)<br>nfs-provisioner is an out-of-tree dynamic provisioner for Kubernetes. | <br><b>openebs</b><br>(from Library)<br>Containerized Storage for Containers                                                                     |

# docker storeに登録されているイメージ数

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

The screenshot shows the Docker Store interface. At the top, there is a search bar with the text "Search for great content (e.g., mysql)". Below the search bar, there are tabs for "Docker EE", "Docker CE", "Containers", and "Plugins". The "Containers" tab is selected. On the left side, there is a "Filters" section with the following options:

- TYPE**
  - ☐ Store
  - ☒ Community (Docker Hub)
- Docker Certified**
  - ☐ Docker Certified
- Categories**
  - ☐ Analytics
  - ☐ Application Frameworks
  - ☐ Application Infrastructure
  - ☐ Application Services
  - ☐ Base Images
  - ☐ Databases
  - ☐ DevOps Tools

In the main content area, a red box highlights the text "1 - 25 of 173664 available images." Below this, there is a message: "Currently showing results from Docker Hub Community" with a link "Back to Docker Store Images". Below this message, there are two image listings:

- prom/node-exporter**  
By prom • Free Under the Docker Community License •  
Community
- microsoft/dotnet**  
By microsoft • Free Under the Docker Community License •  
Official images for .NET Core and ASP.NET Core for Linux and Windows Nano Server  
Community

---

## Kuberntesで何ができるか

# 本日のお話の構成

---

□コンテナのメリット

□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  
Sler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の特徴

3行で

## つまりこういうこと

### □ Immutable Infrastructure

(変更を積み重ねず、都度作る/作り直す)

「手順書よ、さようなら」

### □ 宣言的設定

(あるべき姿を宣言し、その姿に収束させる)

「インフラの全てをコード化する」

### □ 自己修復

(どこかが壊れても、自動的に修復する)

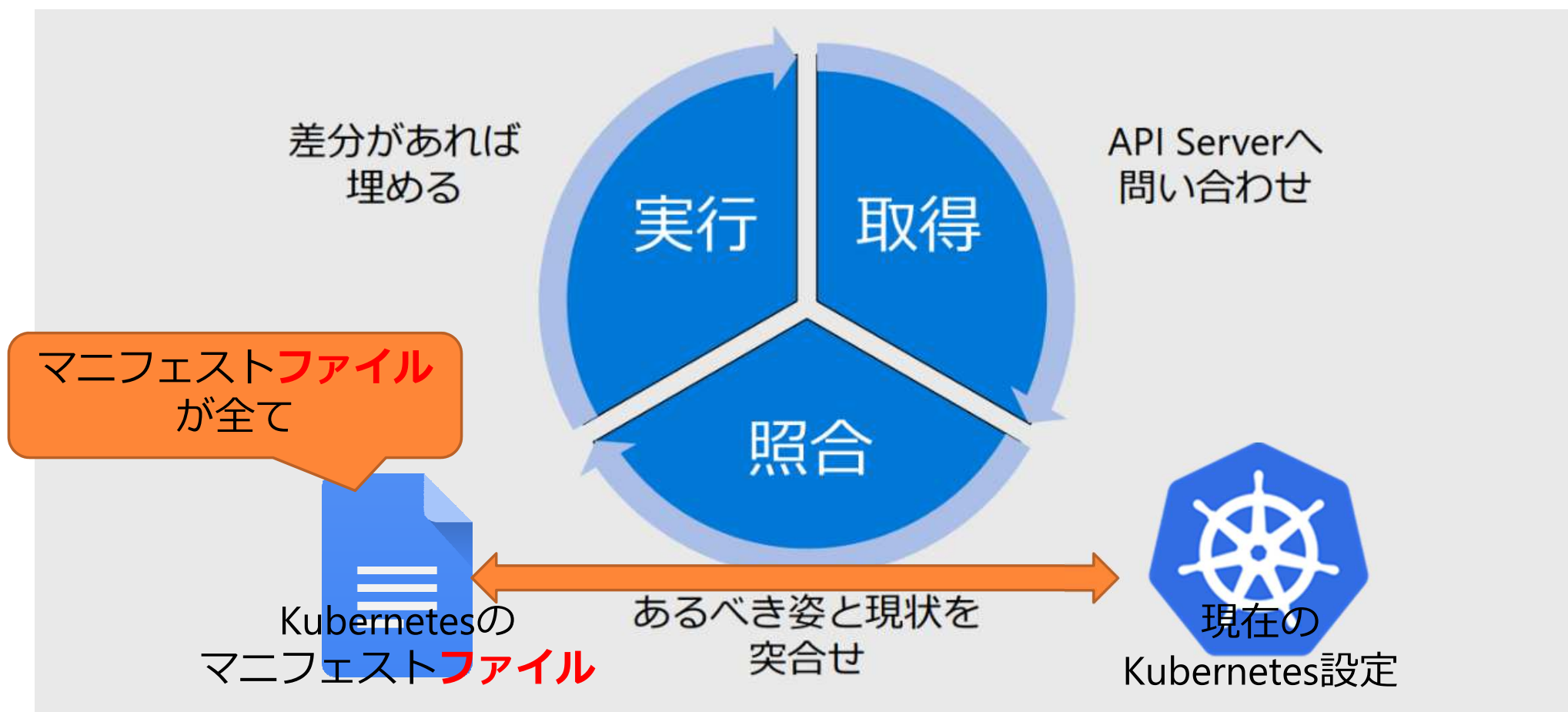
「Kubernetesが全てをコントロール」

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>

Stylez3

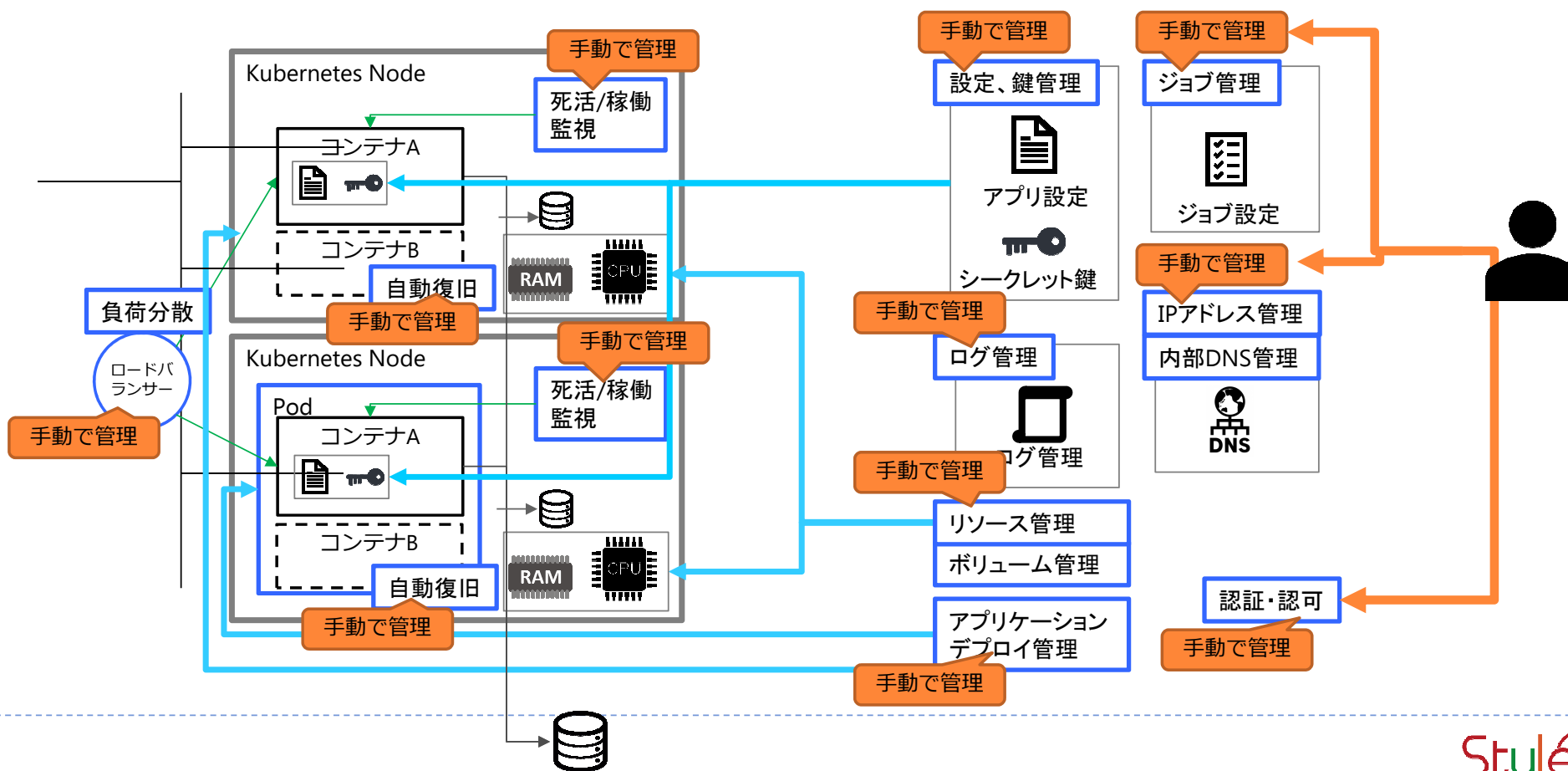
# Reconciliation Loop (突き合わせループ)



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>

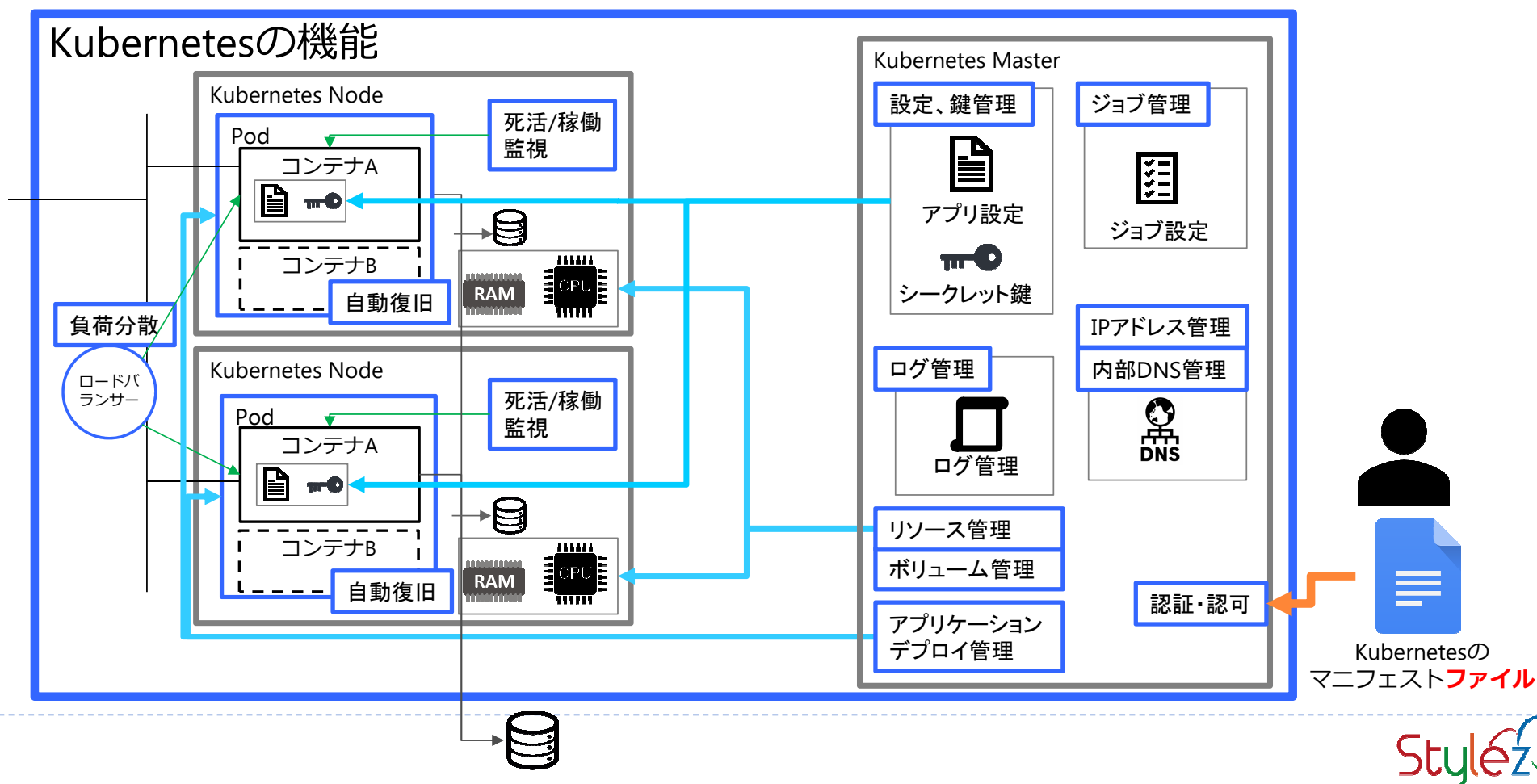
# Kubernetes以前

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

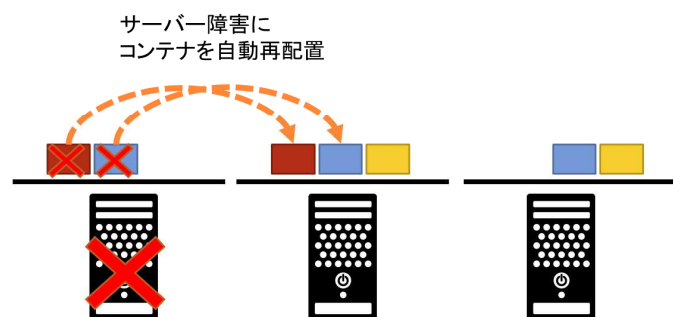
---

**Kubernetesにより  
構成をまとめて  
管理できるようになった**

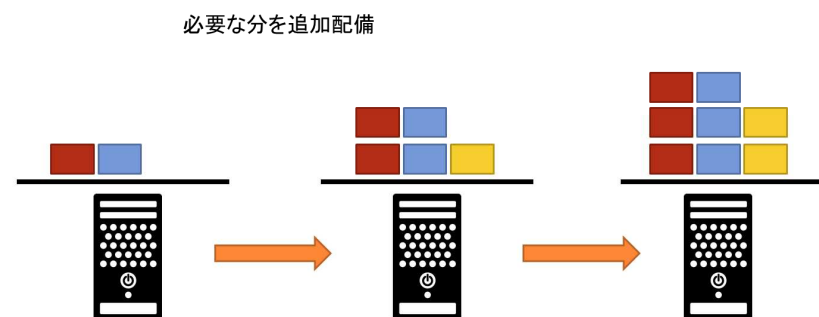


# Kubernetesの機能（1）

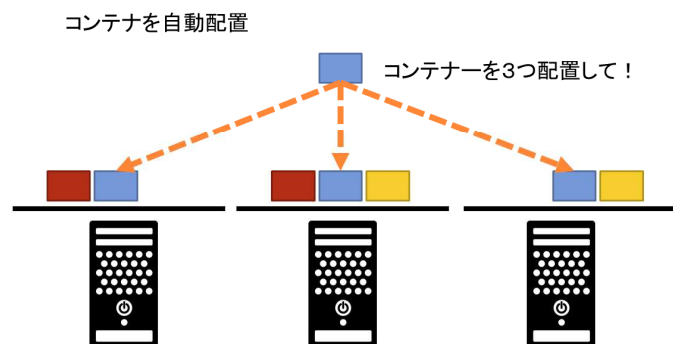
## セルフヒーリング（自動リカバリー）



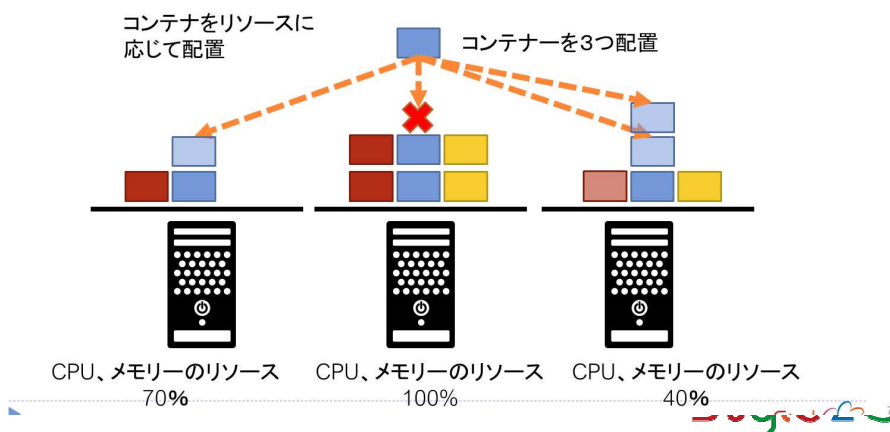
## スケーリング／オートスケーリング



## スケジューリング



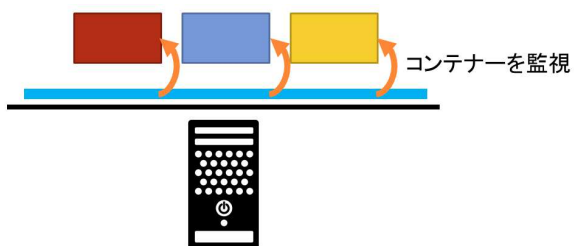
## リソース管理



# Kubernetesの機能（2）

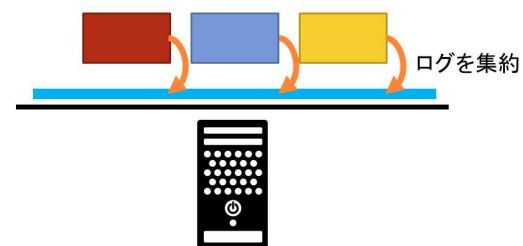
## サービス監視

コンテナが配置されたら  
自動的に監視



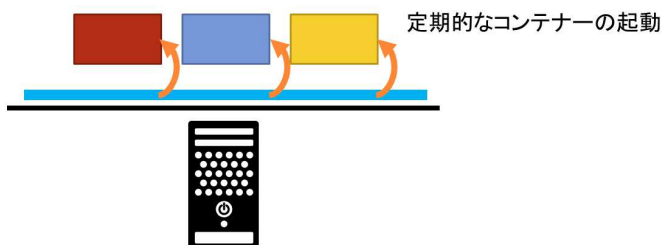
## ログの管理

コンテナが配置されたら  
自動的にログ収集



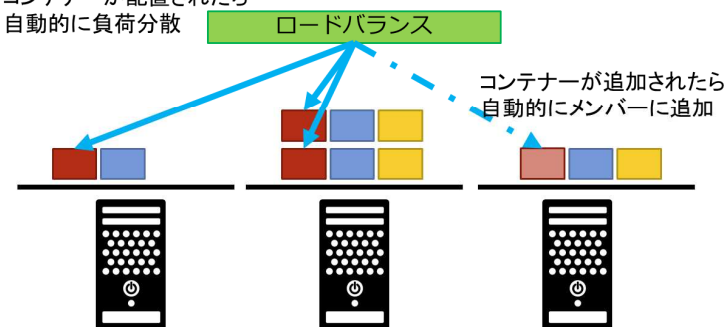
## ジョブ実行

ジョブスケジュールに沿って  
自動的にコンテナ起動



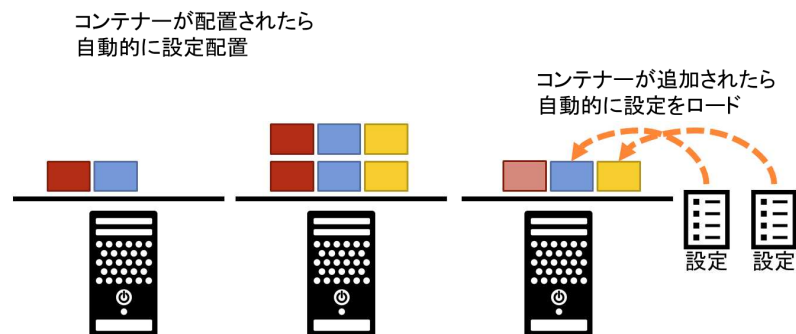
## サービスロードバランス

コンテナが配置されたら  
自動的に負荷分散

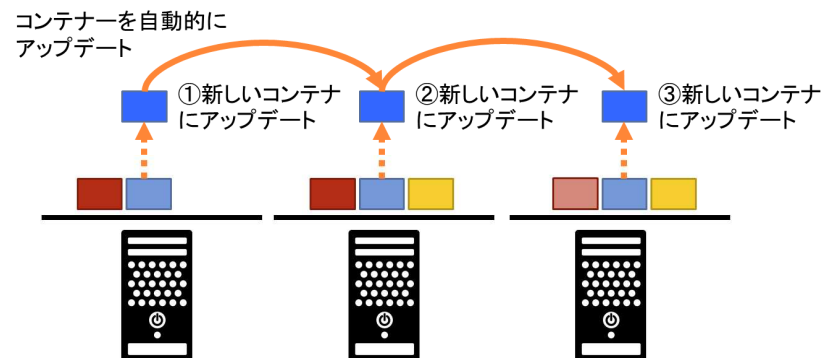


# Kubernetesの機能（3）

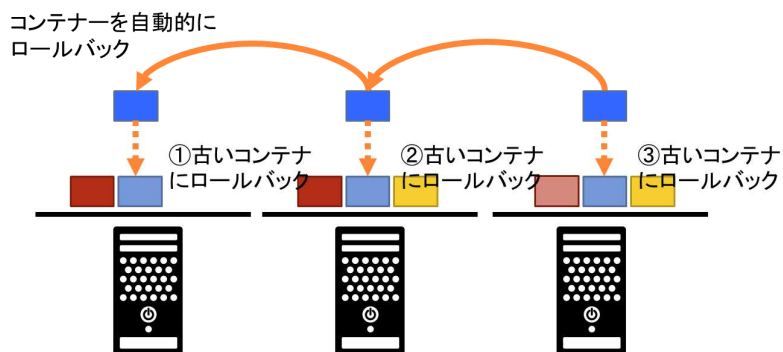
## 設定の自動ロード



## コンテナローリングアップデート



## コンテナロールバック





# 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  
Slerの役割

# クラウドの歴史

---



2006



# クラウドの歴史

---



2006



2008



# クラウドの歴史

---



2006



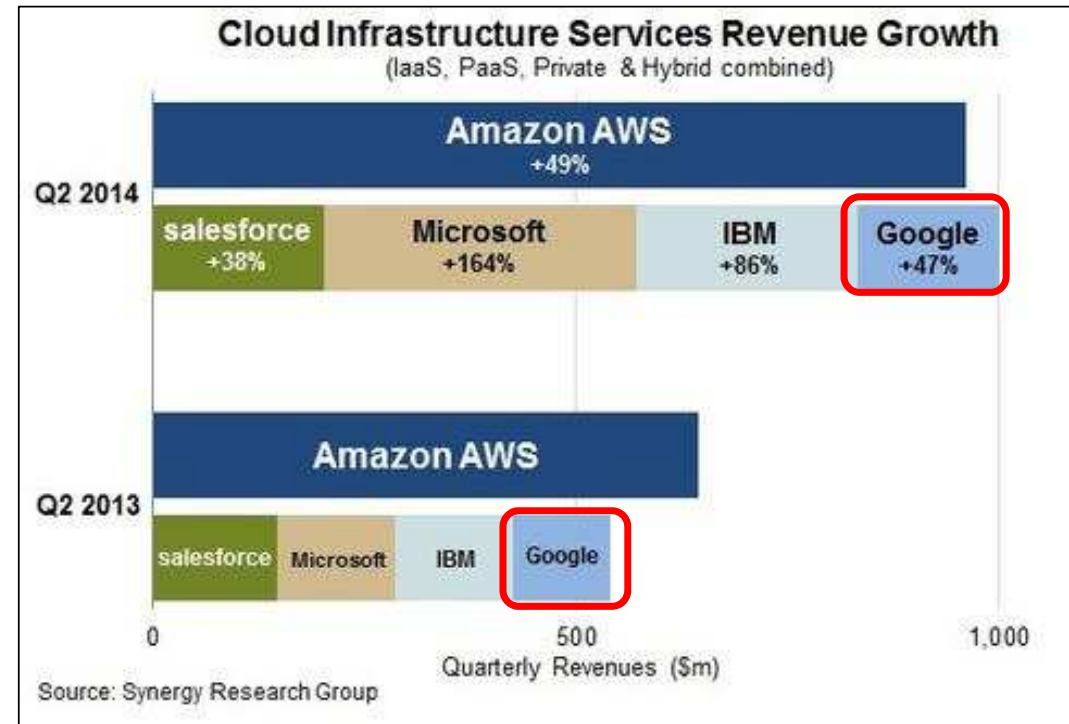
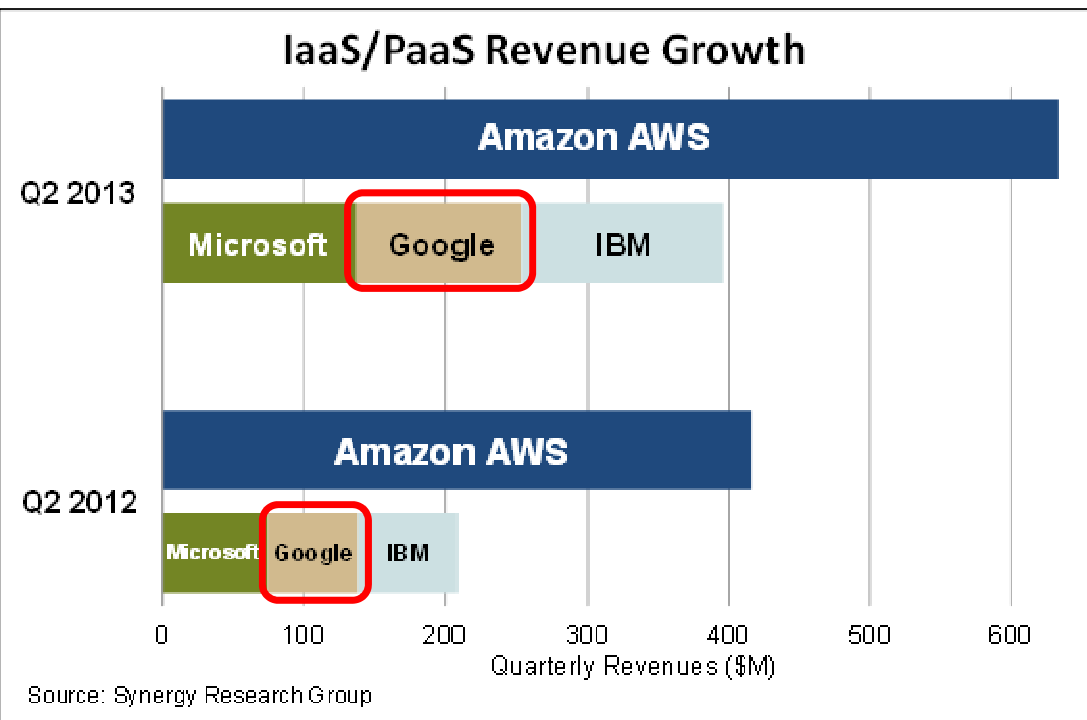
2008



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

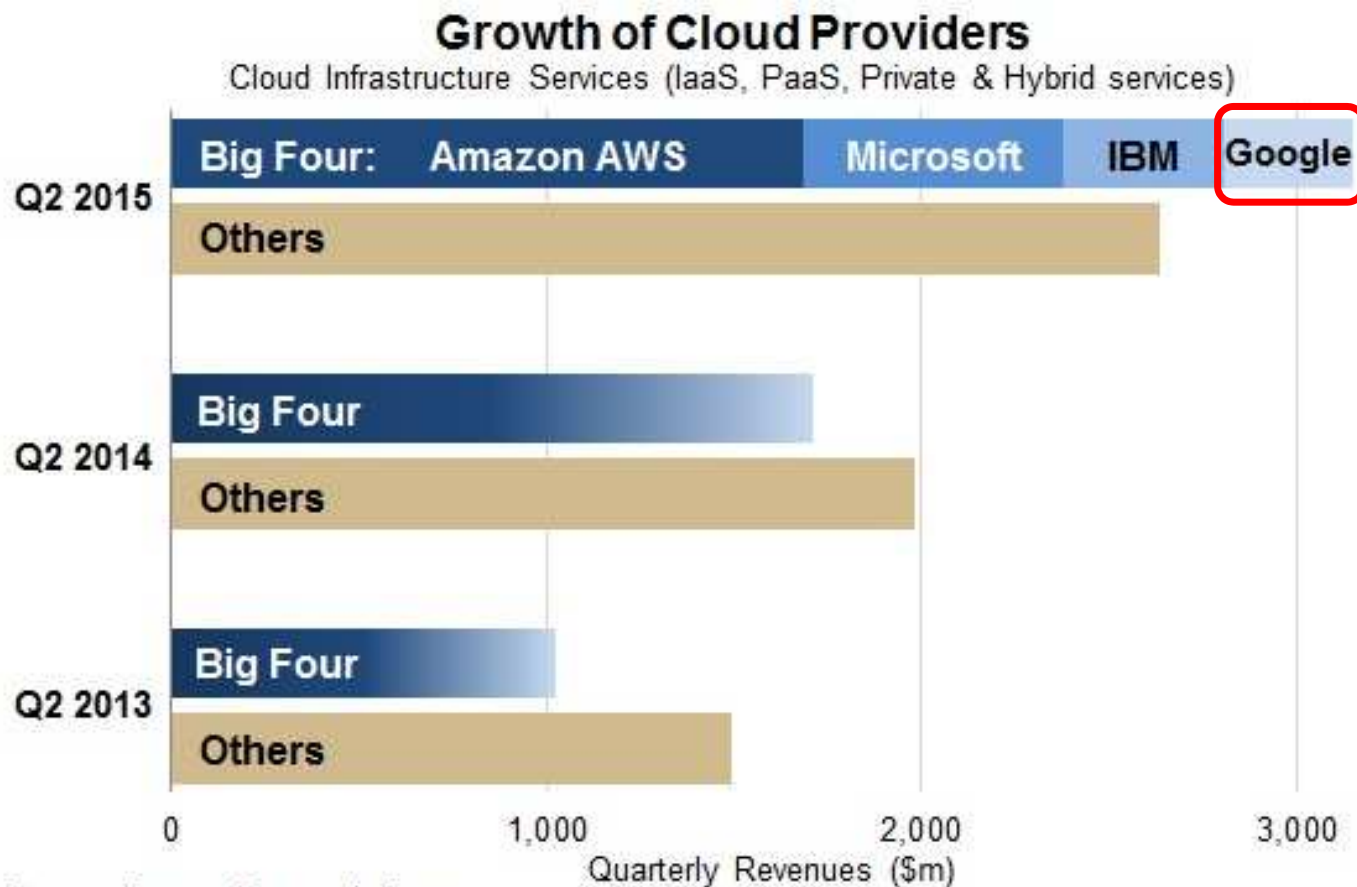
- Amazon EC2
- Auto Scaling
- Elastic Load Balancing
- Amazon WorkSpaces
- Amazon VPC
- Amazon Route 53
- AWS Direct Connect
- Amazon S3
- Amazon Glacier
- Amazon Elastic Block Storage
- AWS Storage Gateway
- AWS Import/Export
- Amazon CloudFront
- Amazon RDS
- Amazon DynamoDB
- Amazon ElastiCache
- 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



# 先行者に追いつけていないGoogle



Source: Synergy Research Group

# 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<br>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<br>管理システム | リソース管理<br>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<br>サービス       | GCPの<br>サービス |
| オペレーション            |              |
| このシステム基盤をGoogleが作る |              |
| GCPのハードウェア、ネットワーク  |              |

このシステム基盤をGoogleが作る

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 | ここをユーザーが作るコンテナで！ |
| オペレーション            |          |                  |
| このシステム基盤をGoogleが作る |          |                  |
| GCPのハードウェア、ネットワーク  |          |                  |

## 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

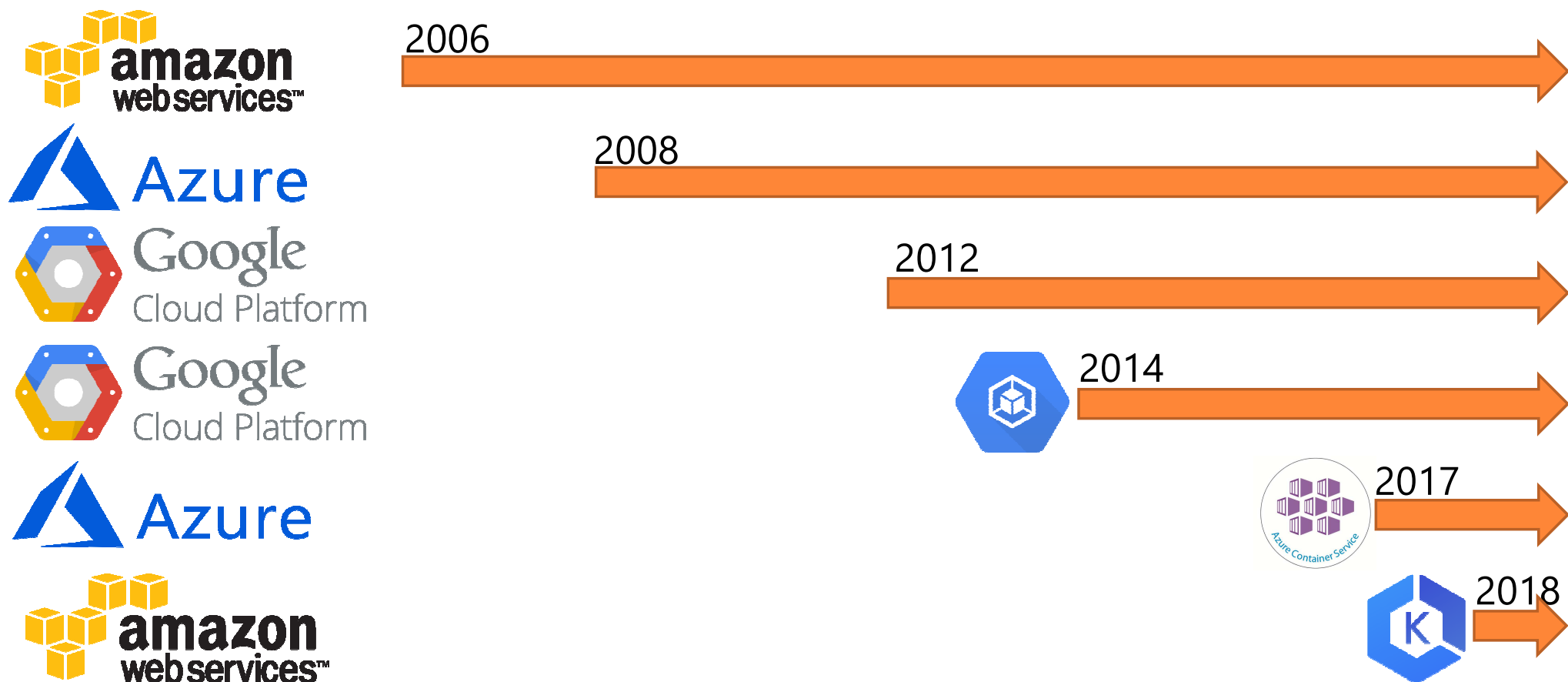


2012

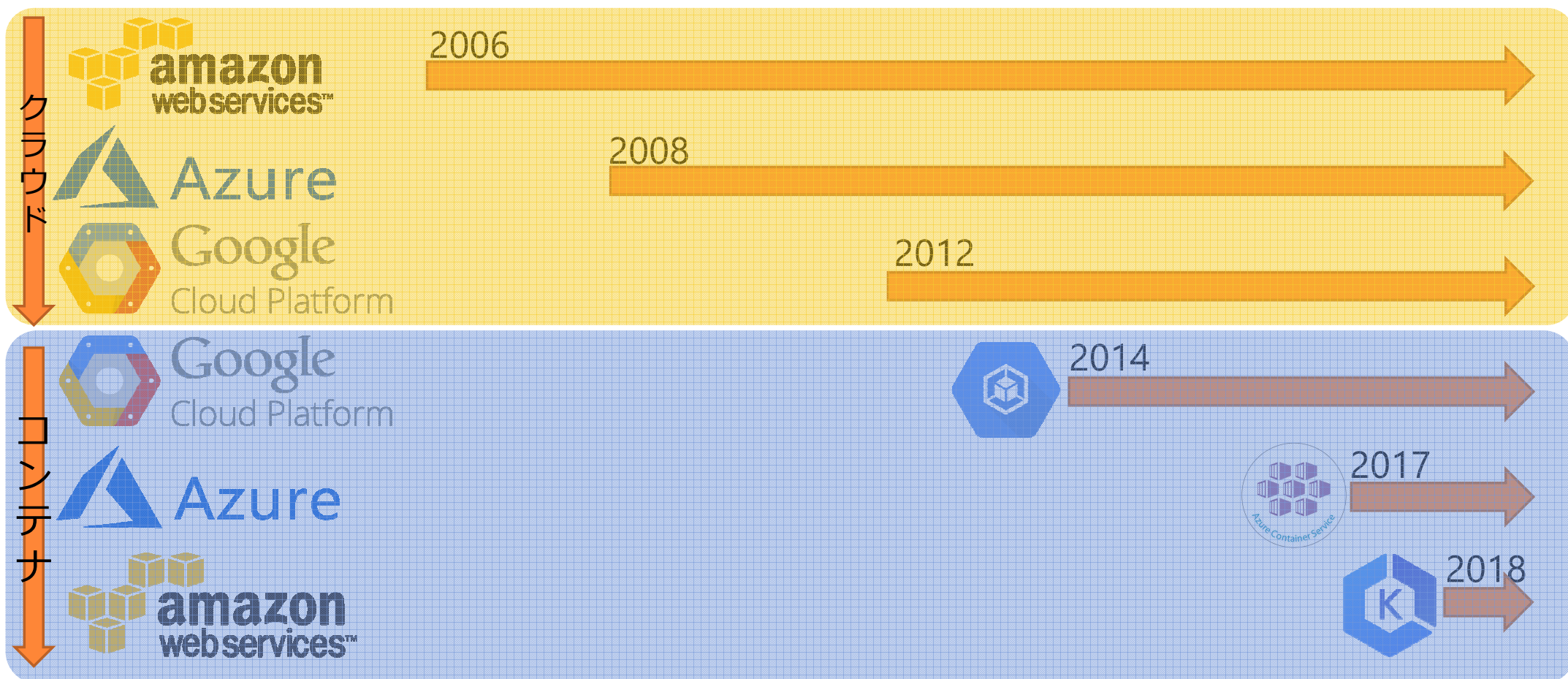




# クラウドの歴史→コンテナ (KaaS) の歴史



# クラウドの歴史→コンテナ (KaaS) の歴史



## ゲームチェンジ

---

Googleのゲームチェンジ  
が成功しつつある



---

**Sler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  
Sler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

---

# Slerにとってのメリット





# Sler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



Kubernetesネットワーク



Kubernetesストレージ

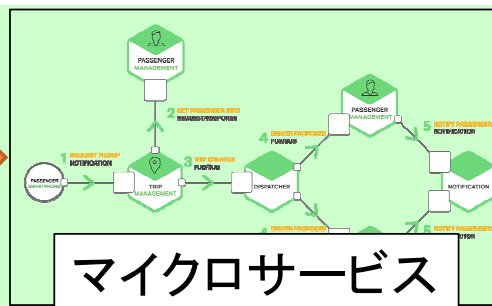


Kubernetesパッケージング



Cloud Design Patterns

クラウドデザインパターン



マイクロサービス





**結構大変なことが一杯**

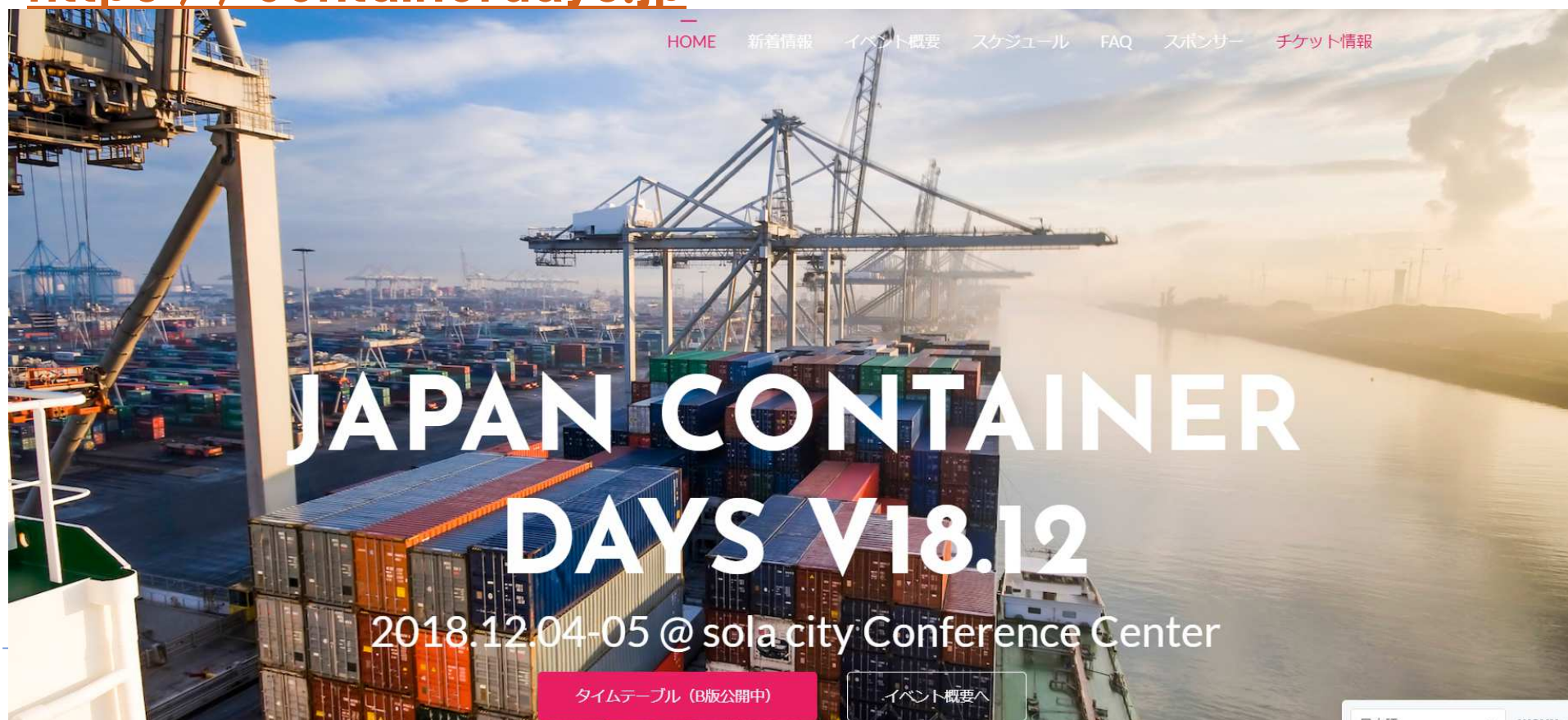
**しかし、Kubernetesは  
必ずやってきます  
今から始めましょう**



# 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。**