ﾊﾞｯｸｴﾝﾄﾞのはなし

tmatsumor

発表の目的：バックエンドの開発業務について、どんな事してんのかなーってところを、浅くざーっとお話します。

@フロント会 241210

バックエンドとは

- サーバーサイドのこと。
- フロント・バックエンドでの分業は主に Webアプリケーションにおいて用いられるため、ここでは端的にサーバーサイドを指すとする。
- サーバーとは Serveする：提供する 意味。
- サービスやコンテンツを提供する。
- 種類としてWebサーバー、ファイルサーバー、メールサーバー、DB、DNS、等。
- フロント・クライアント側から「呼び出される側」を広く作るのが、バックエンドの仕事。

すごく単純なサーバー構成例

↑これ3B+ですね

物理マシン：ラズベリーパイ4

OS：Linux（Raspberry Pi OS Lite）

永続化層：ファイルシステム

Webサーバー：Node.js + express

↑↓

Webソケットクライアント（C#アプリ）

※ 上記に加え無線モジュールとのシリアル通信も実装。

Linux とは

- リーナスさんが作った UNIX 系 カーネル。
- カーネルとは OS のコア。
- カーネルを OS として動くようにしたものがディストリビューション。

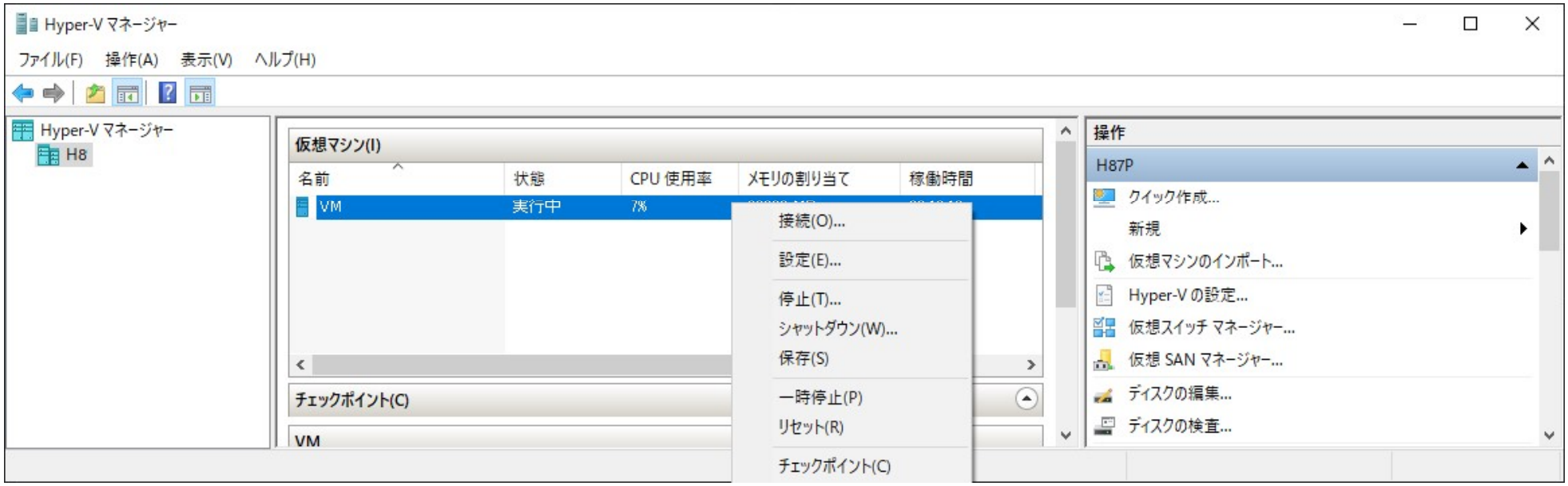
Git 作ったお方

- ディストリビューションには良く見かけるものに RedHat系と、Debian系がある。前者は商用に強く、また CentOS を含む。後者は Ubuntu を含む。

Raspberry Pi OS はDebian系（旧名Raspbian）

仮想化：物理からひっぺがす

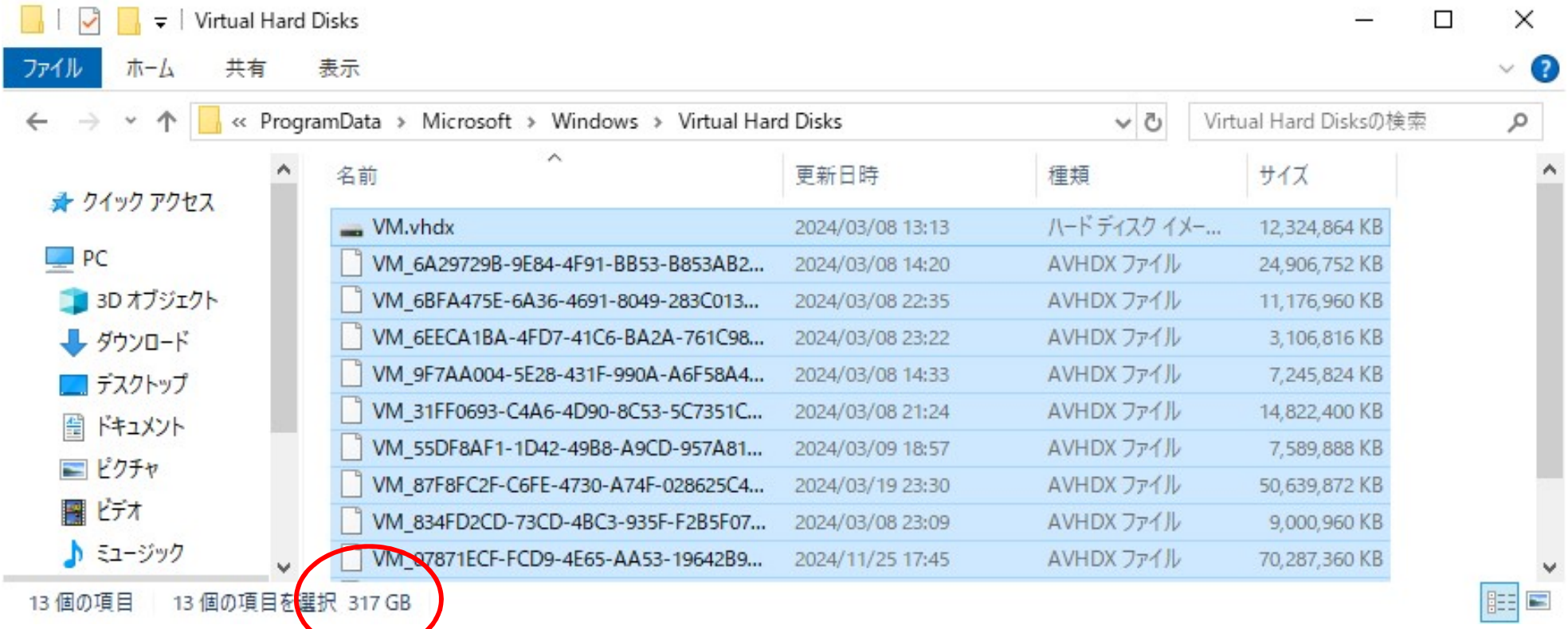
- 身近なもの：VM。2000年ぐらいから。
- VMware, VirtualBox, Hyper-V 等。
- 物理からシステムを引っ剥がし、持ち運べるように。スナップショットも取れる。



Hyper-V は Win Pro 以上で使用可能

VM の問題点

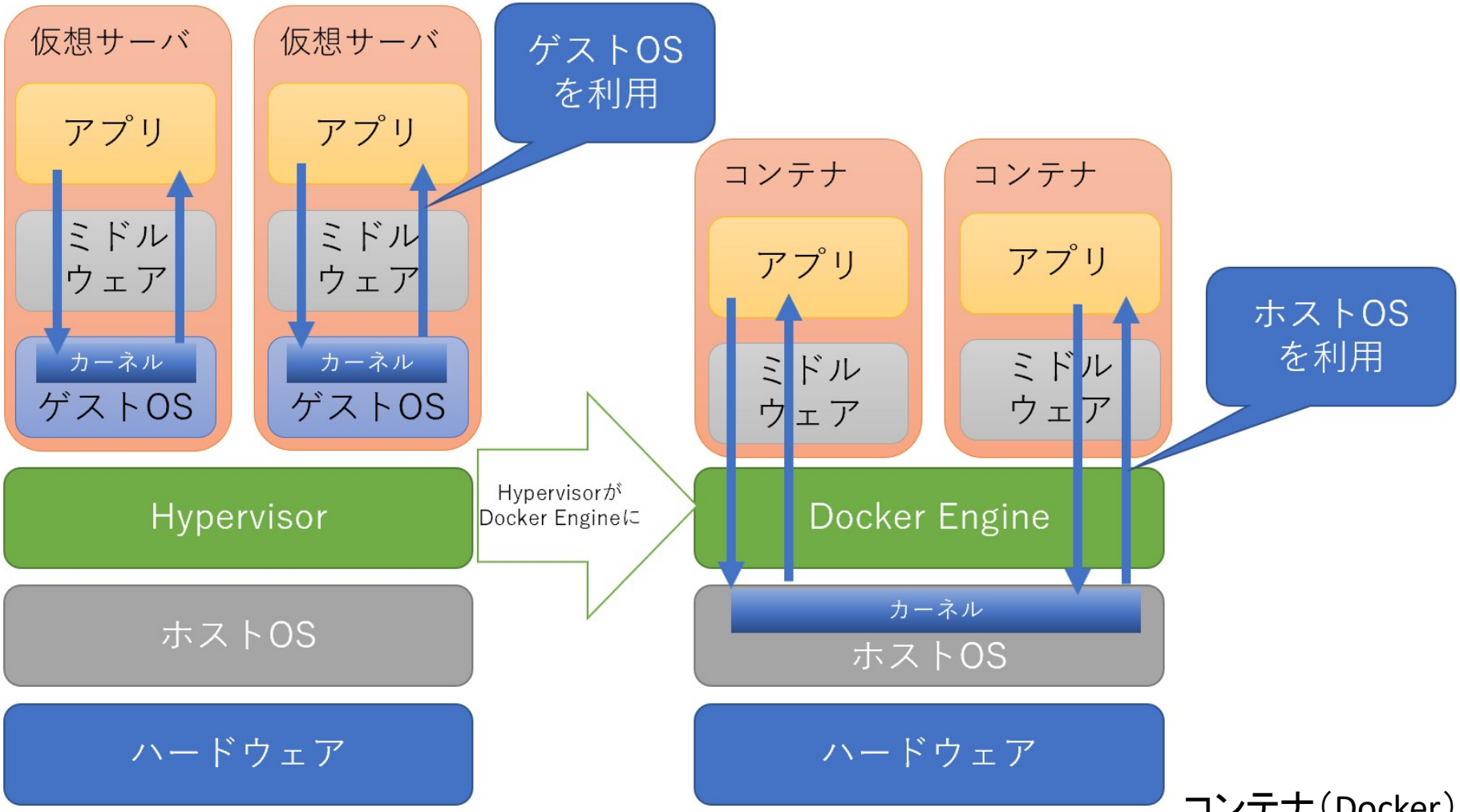
- VM はすっごく容量食う。ゲストの OS が丸っと入っているので。
- メモリも食う。ゲストの OS が必要とするので。



これはスナップショットが大きいんだけれど

コンテナ化

- ゲストOS を使わない。ホストのカーネルを利用。



VM型

コンテナ（Docker）型

https://mtaketani113.github.io/images/it/container/DockerStructure.png

コンテナ技術により

- アプリの実行環境をパッケージ化できた。

→ DEVからSTG、PRD環境にそのまま移動可能。

- 処理が軽量で、最小限のCPU、メモリのみ。

→ 複数のコンテナを同時に実行可能。

- アプリの起動が高速で、開発サイクル向上。

→ DevOps、CI/CD、アジャイルと相性が良い。

Docker

- DockerHub にシステムの雛形となるイメージがある。それを元にコンテナを起動する。
- Dockerfileという命令書でイメージをカスタムする。雛形への変更をレイヤーで重ね合わせる。
- docker-compose
  複数のコンテナをまとめて連携させて起動する。
- コマンド例 :

docker ps	コンテナー覧
docker images	イメージー覧
docker exec -it (コンテナ名) bash	コンテナ入る
docker compose up	複数立ち上げ

たとえば

- MySQL DB のコンテナと、phpMyAdmin、そしてアプリのコンテナの3つを立てる。
- それぞれ Dockerfile を作り、docker-compose.yaml で順指定して起動。
- compose up するだけで、手元に開発環境を準備することができる。
- DB接続設定は yaml にて環境変数経由で指定（secrets を使うのがベター）。

ちょっと待って

- その通り！だがインフラさんは暇ではない。
- yaml を触るのは普通にバックエンドの仕事。
- CI/CD, DevOpsで運用に関わる議論もよく出る。
- どこまでやりゃいいの?
- ﾁｭｰﾆﾝｸﾞ、監視、運用以外は基本的にバックエンドの守備範囲ｶﾅ。

ステートレスと永続化

- コンテナはスクラップ&ビルドされるものなので、ステートレスが望ましい（状態を持たない）。
- 一方で、永続化も必要。落としても、残す。
- ホストのフォルダをマウントして、ファイルやDBを永続化する。また、クラウドに預ける。
- ログはコンテナに持てないため処置を考える。

- コンテナが廃棄容易である事は、サービスのスケーラビリティ、負荷分散、可用性に関わる。また、環境に依存せず可搬性を確保できる。

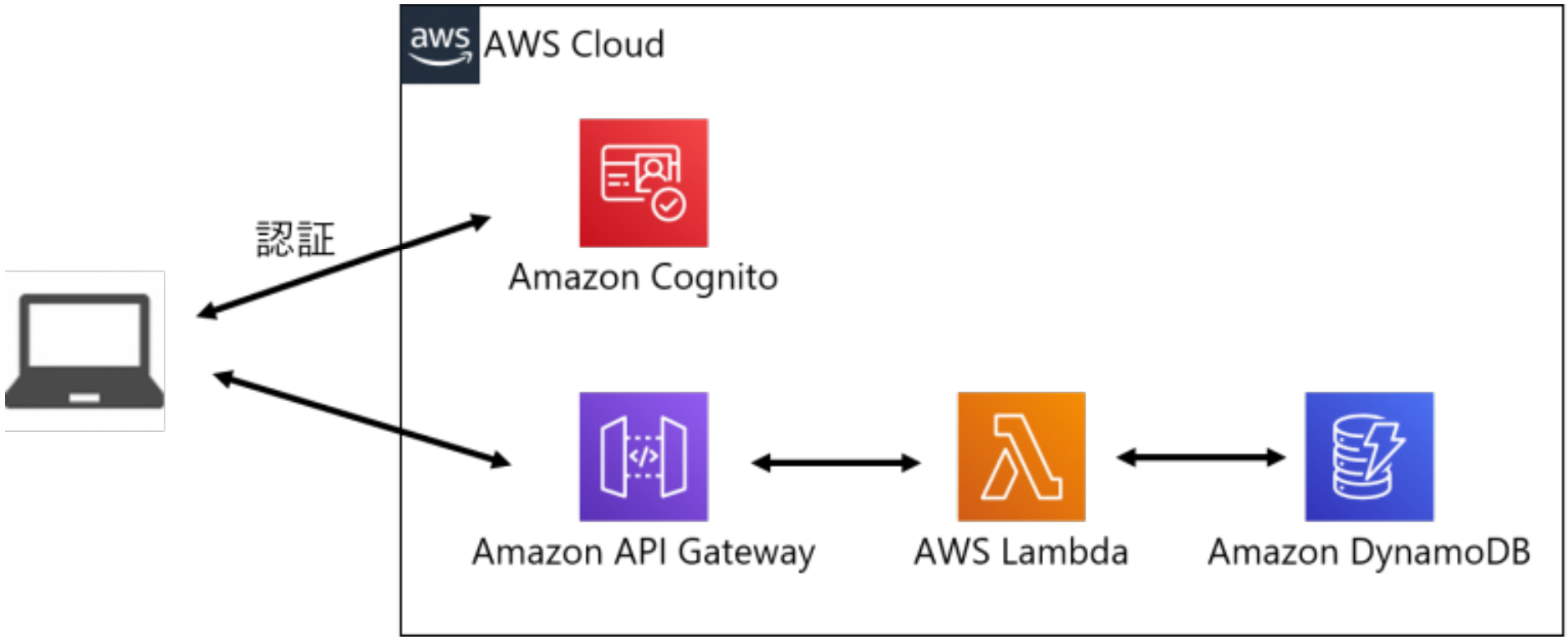
クラウドサービス

- AWS、GCP、Azure が代表的。
- インフラを貸してくれるもの : 自前で物理的に用意しなくても良い。
- サービスの種類が多く、従量課金。
- 運用負担の軽減。高セキュリティで冗長化可。
- Dockerイメージをそのまま持っていく事も可能。

- AWS には 240種類以上のサービスがある。
- IAMユーザーに、IAMポリシーを適用して、サービス利用の許可・禁止を設定する。

IAMロール
もあるよ

使用例（RestAPI）



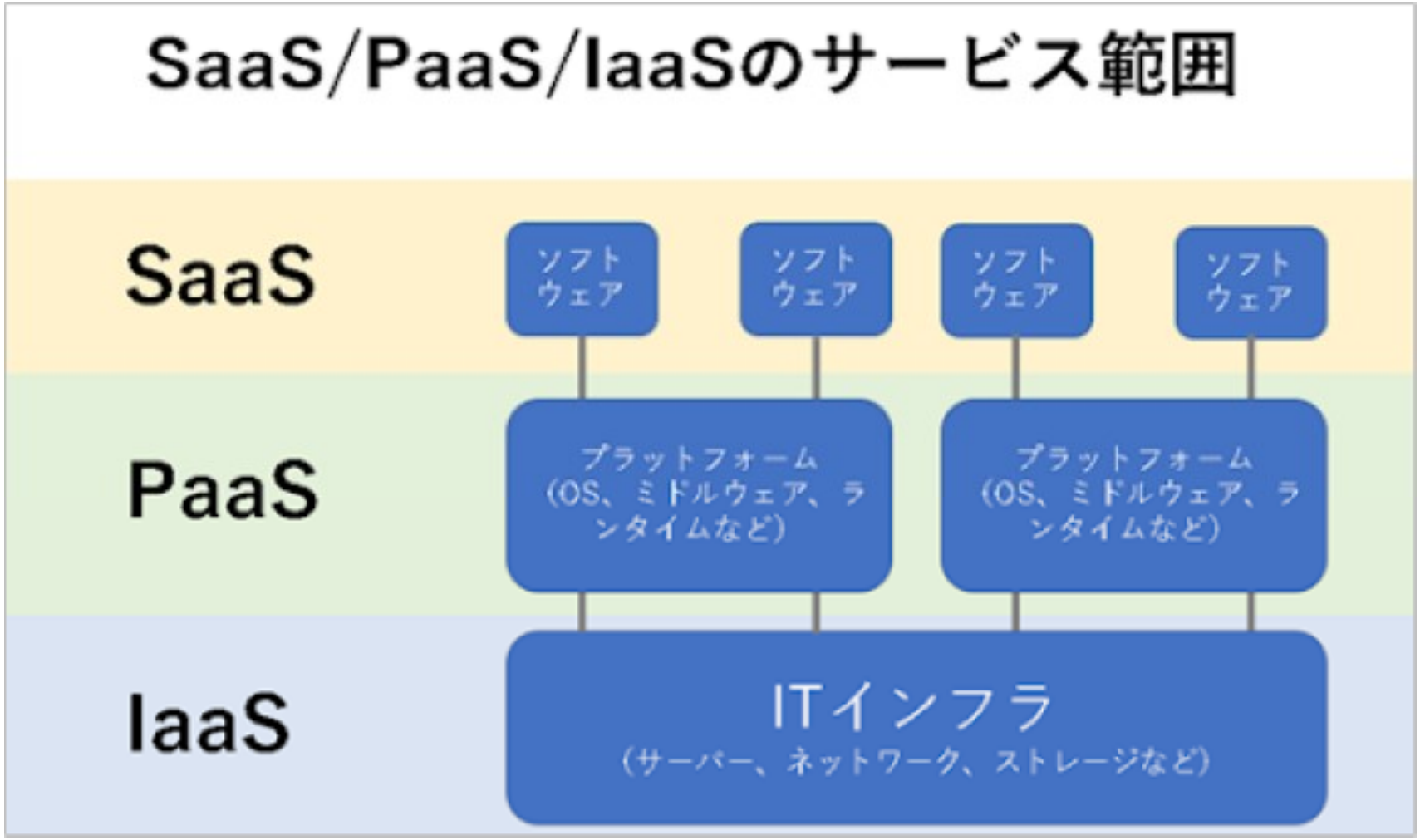
https://www.tdi.co.jp/miso/wp-content/uploads/2019/06/AmazonCognito_Lambda_01-768x309.png

（そのほか）

- EC2：仮想サーバー。ﾚﾝﾀﾙｻｰﾊﾞｰみたいなもの
- S3：Simple Storage Service。データ保存。ﾊﾞｹｯ
- SQS：アプリ間のﾒｯｾｰｼﾞｷｭｰｲﾝｸﾞ　取る側が好きなﾀｲﾐﾝｸﾞで取れる

SaaS, PaaS, IaaS

- エンジニアが使うのはPaaS（のAPI呼び出す）



Gmail
Datadog

Lambda
Cognito

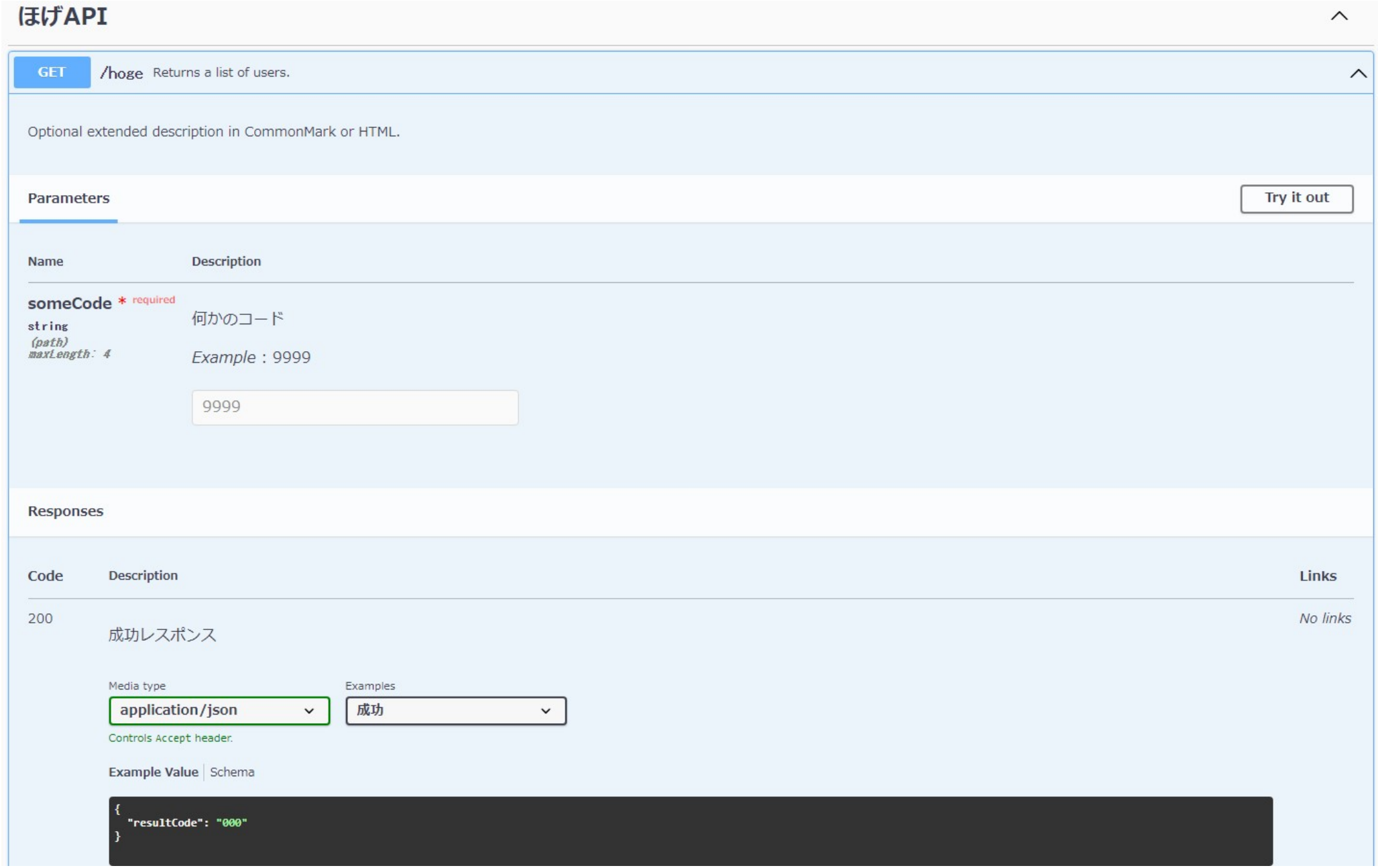
EC2

https://sogyotecho.jp/wp-content/uploads/2021/05/unnamed-2.png

Web APIサーバー

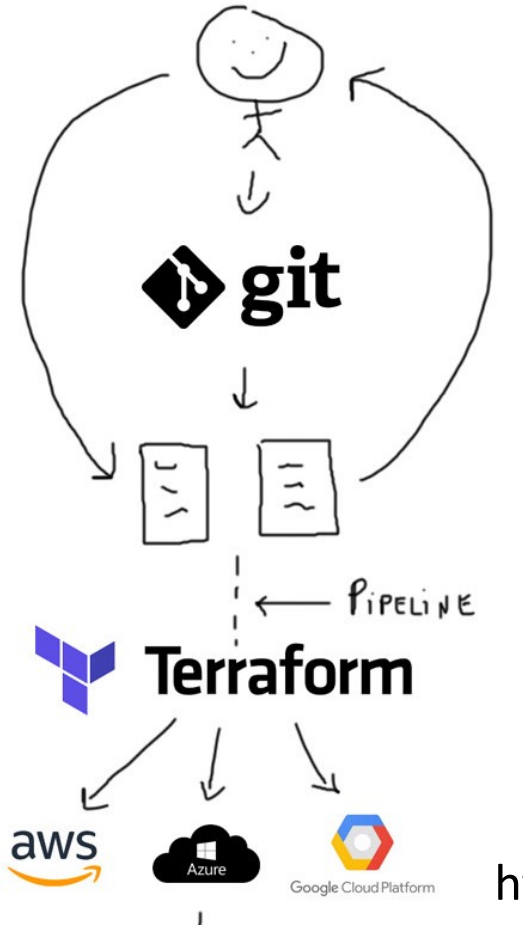
- REST(ful) API：ﾒｿｯﾄﾞ、ﾚｽﾎﾟﾝｽｺｰﾄﾞ、ｽﾃｰﾄﾚｽ
- 仕様書を OpenAPI（Swagger）でyamlに定義してそこからハンドラをコードジェネレートする。（定義と実装のズレを避ける）
- oapi-codegen -generate server でサーバーのスタブが出力されるので、それを元にサーバーを実装しハンドラを紐づければAPIできる。
- oapi-codegen -generate types でリクエストやレスポンスの型定義が出力される。

OpenAPI spec（仕様書）サンプル



IaC（Infrastructure as Code）

- TerraformはHashiCorpによるBSL: Business Source License
- インフラの構成をコードで宣言的に記述可能。
- Git 等でインフラをバージョン管理できる。



- main.tf に書いて terraform apply。
- 手順（AWS）
  1. AWS CLI, Terraform インスコ
  2. terraform init
  3. AWSの認証情報のコピペ
  4. main.tf の実装
  5. terraform apply

IaCにはAWS SAM もあるよ

https://www.lac.co.jp/lacwatch/service/20200903_002270.html

コンテナオーケストレーション

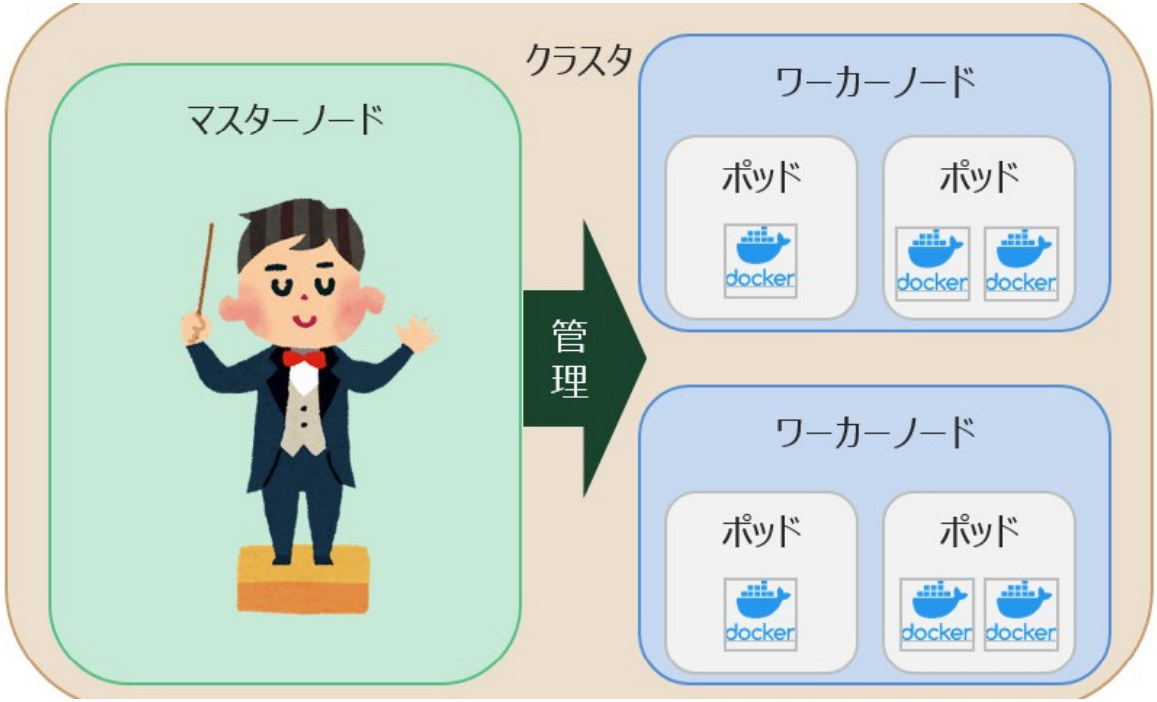
- コケないシステムのため。複数のコンテナを統合的に管理。可用性とスケーリング。

  例）この種類のコンテナは複数のサーバに分散して最低3つ以上起動した状態を維持し、死活監視をして応答がなかったらコンテナを破棄する、コンテナごとの平均の処理負荷が70%を超えていたらコンテナの数を自動的に増やし、30%を切っていたらコンテナの数を自動的に減らす、というような実現してほしい状態を記述して、それにより自動運用…
  https://www.hulft.com/column/glossary-43

- コンテナが落ちた時の自動復旧、負荷に応じたスケール、コンテナへのロードバランシング。

クバネテス：Kubernetes（k8s）

- 2014年にGoogleが発表。同社のBorgがﾍﾞｰｽ。OSS



https://qiita.com/kisama2000/items/2f0bd776022ba257d419

- Pod は IPアドレスを持ちServiceへのアクセスを負荷分散してPodに分配する。
- それらを宣言的にyamlで記述可能。

- Amazon は従来ｺﾝﾃﾅｵｰｹｽﾄﾚｰｼｮﾝとして ECSを提供していた。しかし、k8sに対応したEKSも提供した。ﾏｽﾀｰﾉｰﾄﾞをﾏﾈｰｼﾞﾄﾞにする。

Argo Family

- Argoとはｸﾊﾞﾈ ﾈｲﾃｨﾌﾞのﾂｰﾙｾｯﾄ。OSS

① Argo Workflows：依存関係を持つ等、単純でないバッチを簡単にGUIで扱える。

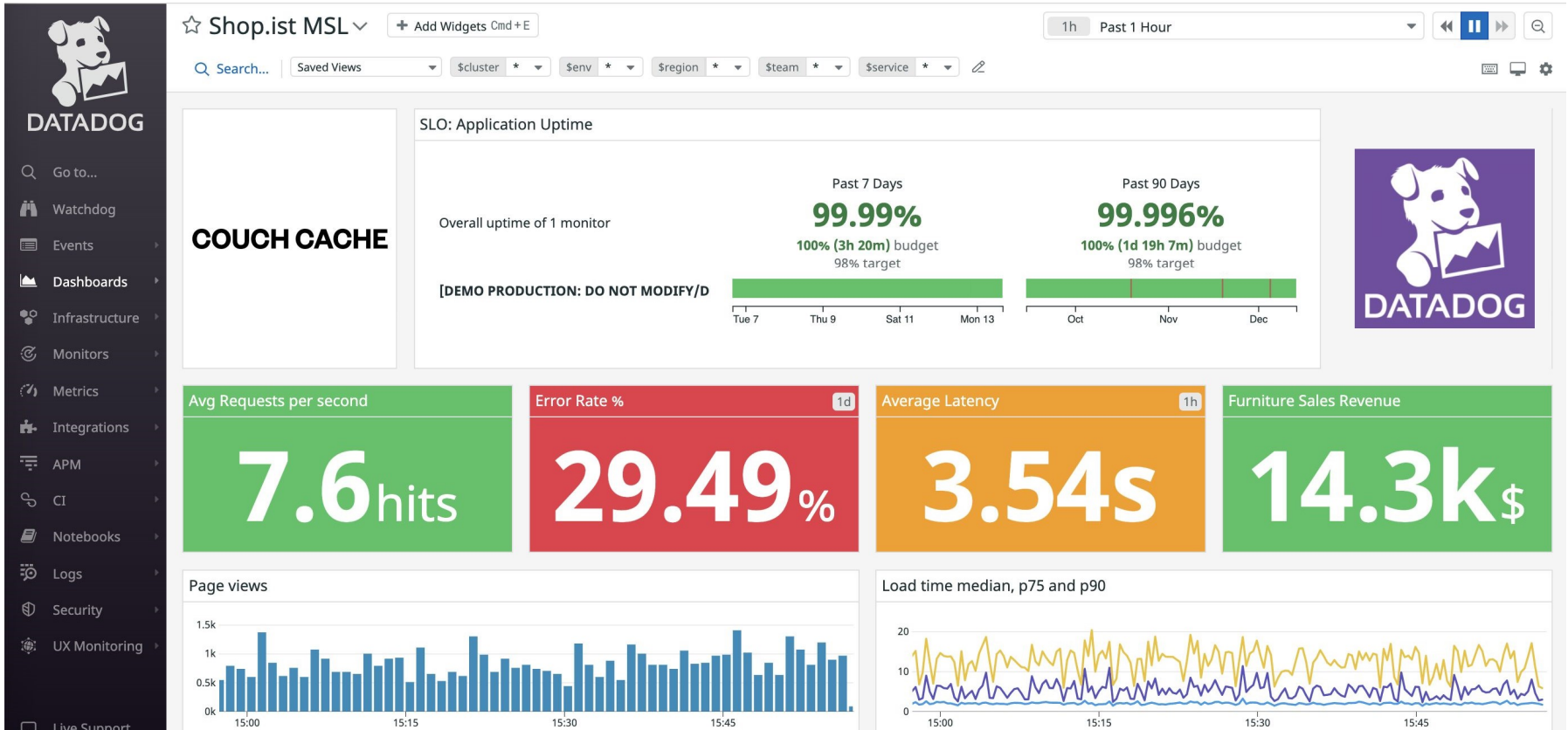
② Argo CD： GitOpsの実現のため。k8s上にArgo CDを置きｺﾝﾌｨｸﾞﾘﾎﾟｼﾞﾄﾘをﾌﾟﾙしてﾃﾞﾌﾟﾛｲ。

③ Argo Rollout はブルーグリーンデプロイメント、またカナリアリリース等を機能提供。

- ブルー（旧）グリーン（新）の２系統を用意し、ロードバランサで接続先切り替え。障害対応。
- カナリアは新しいバージョンを一部の Pod でリリースし、新旧を Pod で段階的に切替える。

Datadog

- SaaS で提供されるﾓﾆﾀﾘﾝｸﾞｻｰﾋﾞｽ。
- 導入が簡単。k8sに Datadog Agent をインストールすれば、とりあえず使える。
- リッチなダッシュボード↓

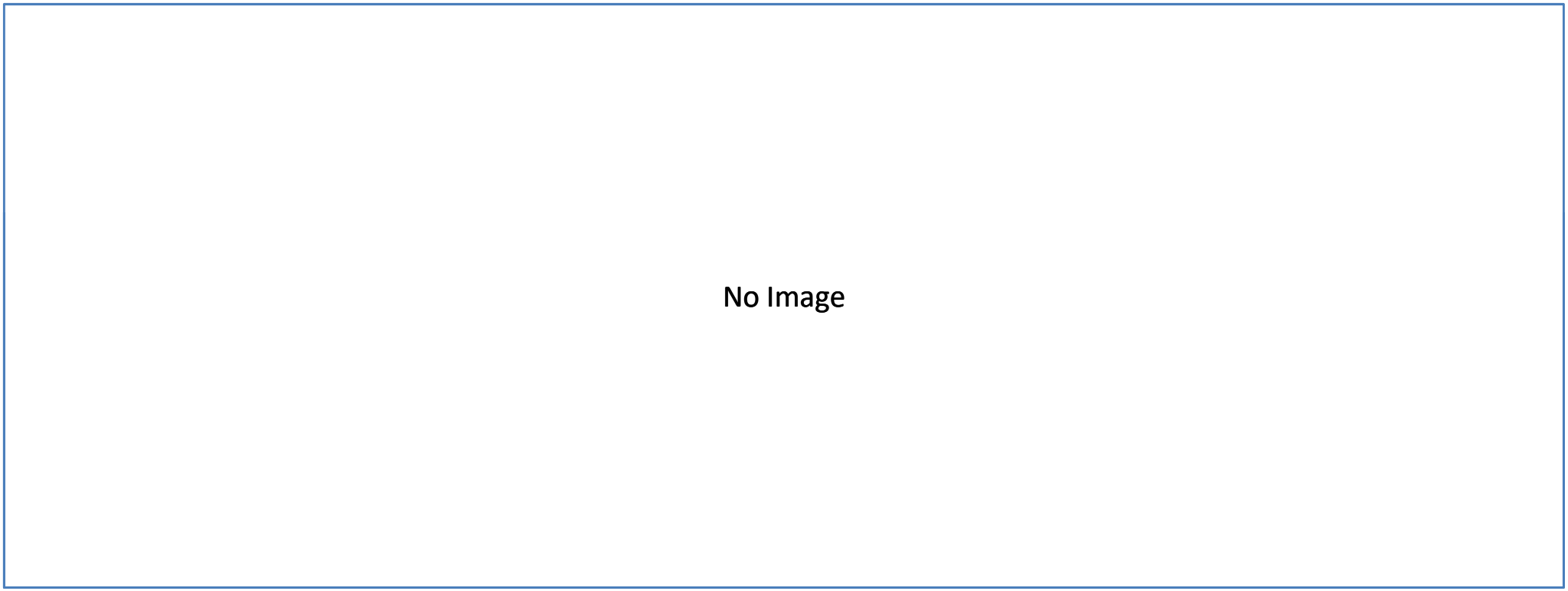


特徴

- コンテナの標準出力はログ取得容易。
- メトリクスをサーバーから取得。定量化。
- 従量課金。（アーカイブされているログの利用は別途お金がかかったハズ）⇒ リハイドレート
- ログパイプラインで流れてくるログを絞り込む。
- ダッシュボードの監視画面は自由に作れる。
- 豊富なアラート。メール、Slack 等自由自在。
- リアルタイム監視
- スケーラブル。複雑・大規模なインフラでも可。
- Watchdog: 機械学習による異常検知機能

まだ色々あると思うのだけれど

- CI/CD, DevOps, GitOps
- アーキテクチャ（オニオン、クリーン）と DI
- チャンネル「#dev-architecture」作りましたので、そちらでお話下さい。

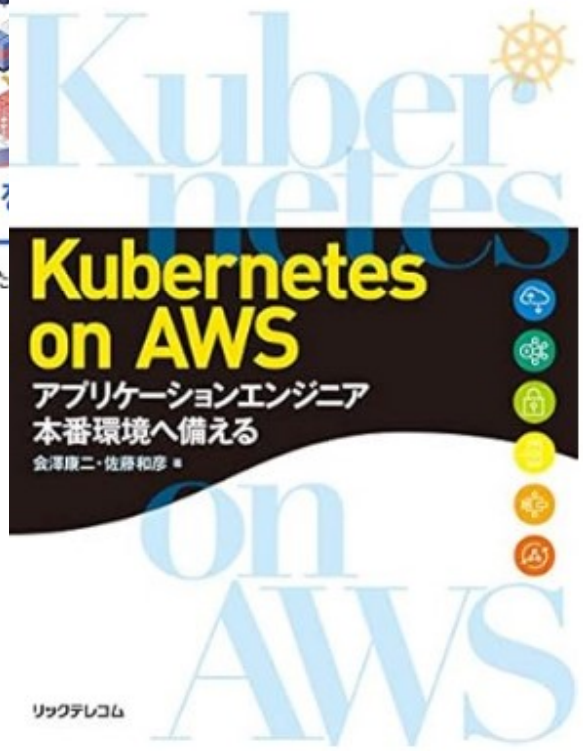


参考図書


図解即戦力
豊富な図解と丁寧な解説で、知識0でもわかりやすい！
Amazon Web Services
AWS
の
しくみと技術が
しっかりわかる
教科書
小笠原種高
これ1冊で
Amazon EC2 / Amazon S3 / Amazon VPC / Amazon RDS /
ネットワーク / サーバー / クラウド / オンプレミス / 仮想化技術 / 分散技術 /
リージョン / アベイラビリティーゾーン / インスタンス / Amazon EBS /
AMI / Amazon Route53 / AWS Lambda / コンテナ…
Amazon Web Servicesのしくみがキーワードベースでわか
AWSの
一番やさしい本で
ネットワーク&クラウドの
AWSの各サービスまで
非エンジニアにもわかりや


インプレスR&D【NextPublishing】
技術の泉
E-Book/Print
実践Terraform
野村
AWSにおけるシステム設計とベストプラクティス
Terraform職人を目指す
すべてのエンジニアのために!
泉
技術の泉
K&D


現場での基礎から本番運用まで
Kubernetes
Kubernetes の Roadmap
知識地図
青山真也
小竹智士
長谷川誠
川部勝也
岩井佑樹
杉浦智基
あふれる資料から厳選した
「情報のハブ」としての入門書
技術評論社
信頼性・可用性・保守性を
コンテナオーケストレー
第一線のエンジニアによる、最短で学ぶた


Kubernetes
on AWS
アプリケーションエンジニア
本番環境へ備える
金澤康二・佐藤和彦 著
リックテレコム