Google Cloud

コンテナに苦手意識ないですか？
今さら聞きづらくないですか？
初心者向けに１から説明します

2022.03.01

本セッションの概要

コンテナに苦手意識ないですか？　今さら聞きづらくないですか？

ソースコードからコンテナをデプロイするまでの流れをだれでもできるように詳しく説明してみようと思います。

デモを中心に進めますので、画面をみながら一緒に作業をすすめることもできます。

コンテナ苦手意識を一緒に克服しましょう！

初心者向けセッションのため、どんな疑問でもカジュアルに質問してください！

アジェンダ

コンテナを理解してみる	01
Google Cloud にデプロイしてみる	02
まとめ	03

01

コンテナを
理解してみる

物理サーバー vs 仮想マシン vs コンテナ

Deployment ~months
Not portable
Low utilization

Deployment ~days (mins)
Hypervisor specific
Low isolation, Tied to OS

Deployment ~mins (sec)
Portable
Very Efficient

コンテナを支える主な Linux カーネルの機能

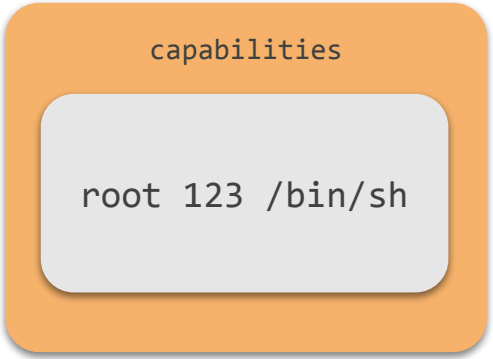
- Capabilities
- cgroup
- namespace
- Overlayfs

```
root 123 /bin/sh
```

capabilities

**root 権限を細かく分け、必要な権限のみを
プロセスやファイルに付与する。**

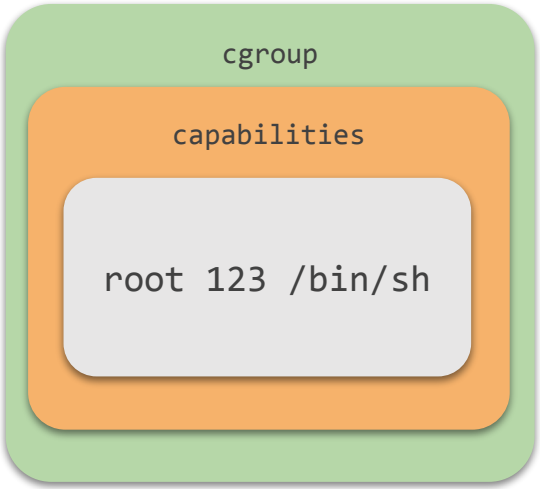
例えばコンテナで raw ソケット（ネットワークプロトコルのヘッダー情報までアクセス可能なソケット）を使って Pingコマンドを実行したり、TCP/UDP ポート番号 1024 以下に対してプロセスをバインドしたりするのは、capabilities によって実現している。



cgroup

プロセスをグルーピングした上で、CPU やメモリ、ネットワーク帯域などのハードウェアリソースを割り当て制限する。

仮想マシンと同様、「CPU を 1、メモリーを 2GB」といった形で、ハードウェアリソースをコンテナ上のグルーピングされたプロセスに割り当てる。



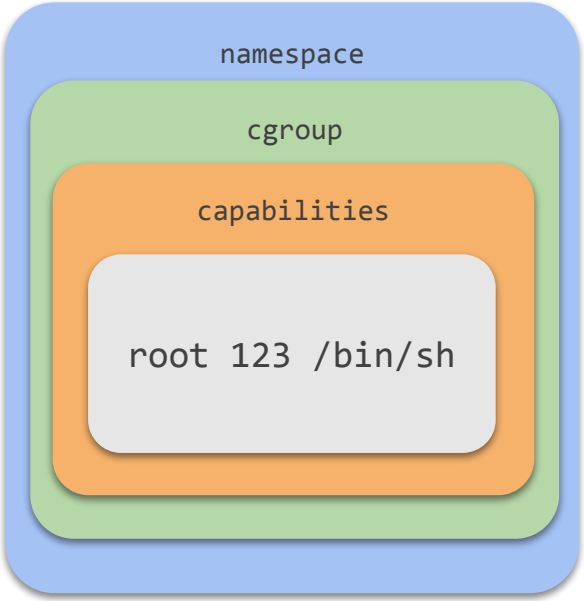
namespace

プロセスから見えるシステムリソースを
他のプロセスから分離する。

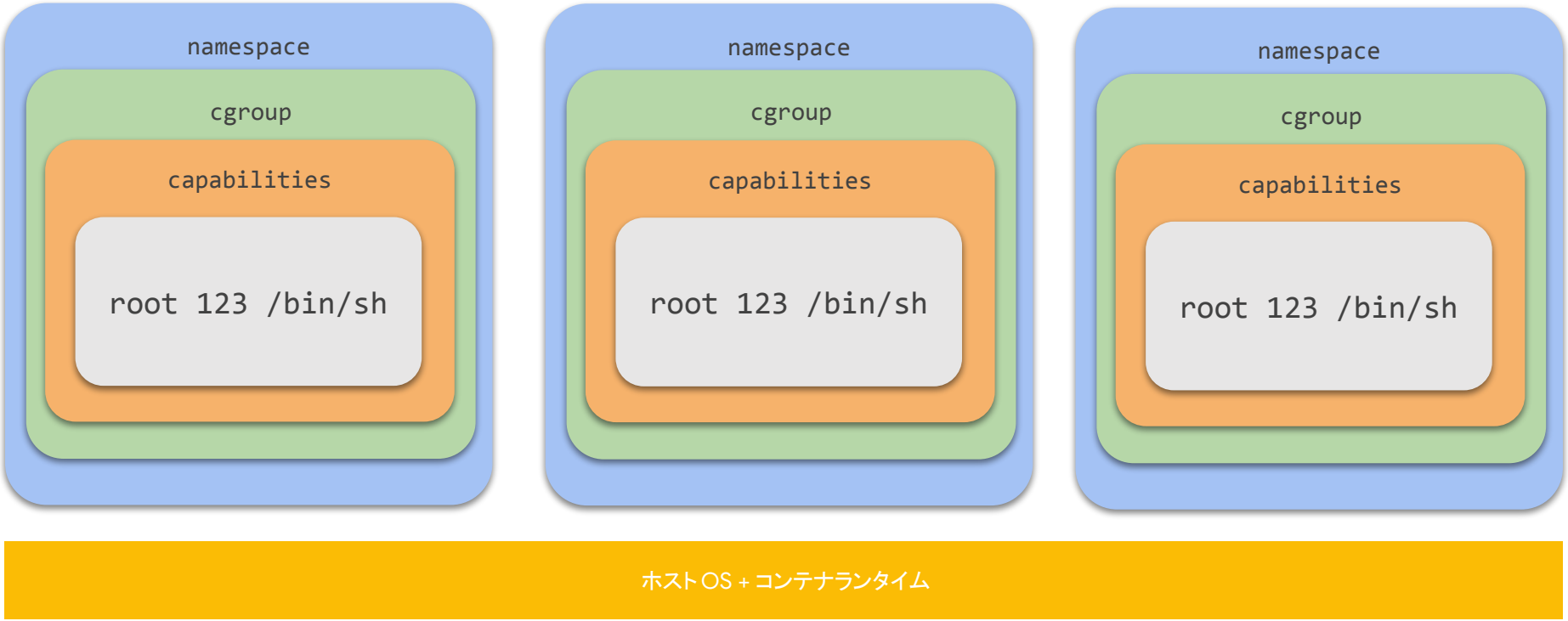
Namespace によって分離できる
主なシステムリソース

- マウントポイント
- プロセス ID
- ネットワーク
- ユーザー ID

例えばコンテナは同一ホスト OS 上で動作していても、
それぞれ異なるネットワークインターフェースや
IP アドレスを持つ。これらは、ネットワークの Namespace により実現している。



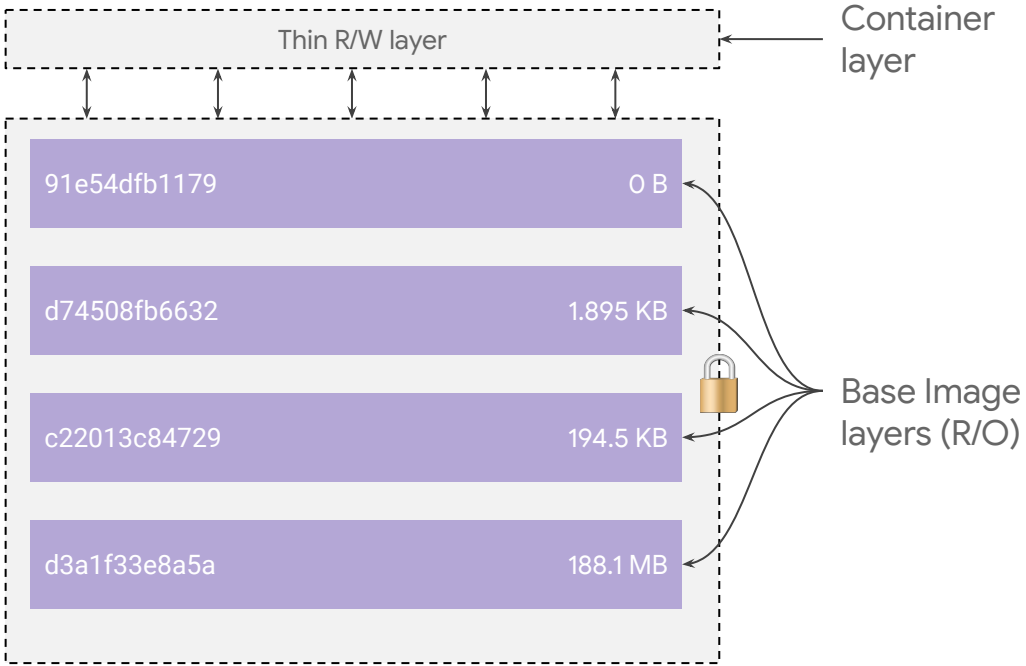
コンテナとして Isolation された状態



Overlayfs

複数のイメージレイヤー（ディレクトリやファイル）を重ね合わせて 1 つのファイルシステムとして見せる。

同一ホスト上で動くコンテナ同士でイメージレイヤーを共有するため、全体のコンテナのデータサイズ（イメージサイズ）を小さく出来る。



どんなワークロードに向いているか

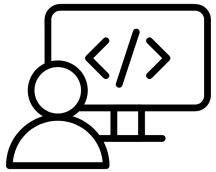
- ステートフル よりも ステートレス (API サーバー etc.)
- 多機能より単機能 (マイクロサービス アーキテクチャ)
- 高速なスケールアウトが必要
- 新規開発 (開発 / 運用要件が変わるので)

※もちろんこれ以外にも利用可能です

02

Google Cloud にデプロイしてみる

仮想マシンを使ったデプロイまでの流れ (例)



1. アプリ開発 (コーディング、ビルド、テスト)

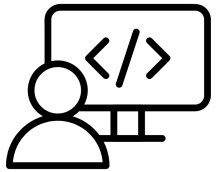
2. 仮想マシンの作成
Vagrant / Terraform

3. OS 周りの設定、アプリ、依存ライブラリ等のインストール
Ansible / Chef / Puppet

4. ゴールデンイメージの作成
Packer

5. ゴールデンイメージからサーバー作成
Terraform / Deployment Manager 他

コンテナを使ったデプロイまでの流れ (例)



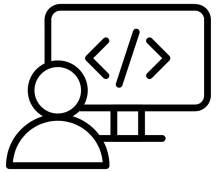
1. アプリ開発 (コーディング、ビルド、テスト)

2. コンテナイメージの作成
Docker

3. コンテナイメージのデプロイ
Docker, Kubernetes 他

#コンテナをホストする仮想マシン等が最初からある前提

実際にこの流れを Google Cloud でやってみます



1. アプリ開発（コーディング、ビルド、テスト）

2. コンテナイメージの作成
Docker

3. コンテナイメージのデプロイ
Docker, Kubernetes 他

Cloud Run によってコンテナをシンプルに運用できる

Cloud Run は**コマンド 1 つ**でサービスのデプロイから外部公 開まで可能。VPC や Load Balancer など下回りのインフラを気にする必要なし。
開発者はアプリケーション開発により集中可能。

サービスをデプロイした瞬間から
メトリクス、ロギングの収集を自動的に開始。

Cloud Monitoring / Logging とネイティブに連携しているので設定を別途行う必要なし。

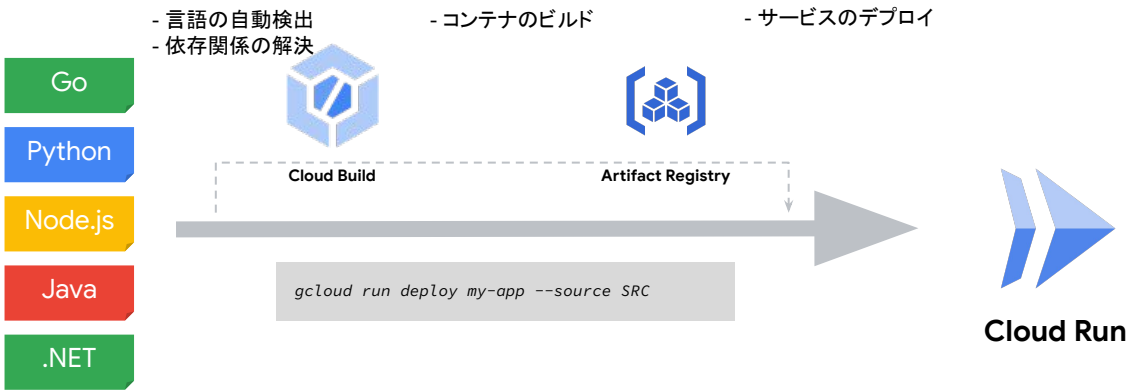
```
# Cloud Run
$ gcloud run deploy...
```

Cloud Operations

Logs / Metrics etc.

Google Cloud Buildpacks によってビルドをシンプルにできる

Buildpacks による言語自動検出と自動ビルドによって、Dockerfile を書くこと無くソースコードから直接 Cloud Run サービスのデプロイが可能に。
Google が CVE 基準に沿った脆弱性スキャンを定期的に行う、マネージドイメージをベースとすることでイメージの安全性も可能な限り確保。



https://github.com/GoogleCloudPlatform/buildpacks

実際に動かしてみよう

03

まとめ

まとめ

今回のセッションはコンテナの初級編になります。

- コンテナの仕組みがどうなっているか
- コンテナをどうデプロイしたらよいか

この 2 点の理解が進んでいると幸いです。

さらに詳しくコンテナの運用を知りたい方は
こちらのセッションもオススメです