# Continuous Deployment with **Go** on **AWS ECS**

Kazuhisa Togo

# Kazuhisa Togo

a backend developer

works at **LIG INC.**

kazuhisa.togo

# Kazuhisa Togo

freelancer, etc.

▼

some web dev company in Sydney

▼

some British financial company

▼

LIG INC.

LIG ID

Facebookでログイン

もしくは

メールアドレス

パスワード

メールアドレスでログイン

▶ 新規登録する

▶ パスワードを忘れた方

LIG ID

LIG ID

Facebookでログイン

もしくは

メールアドレス

パスワード

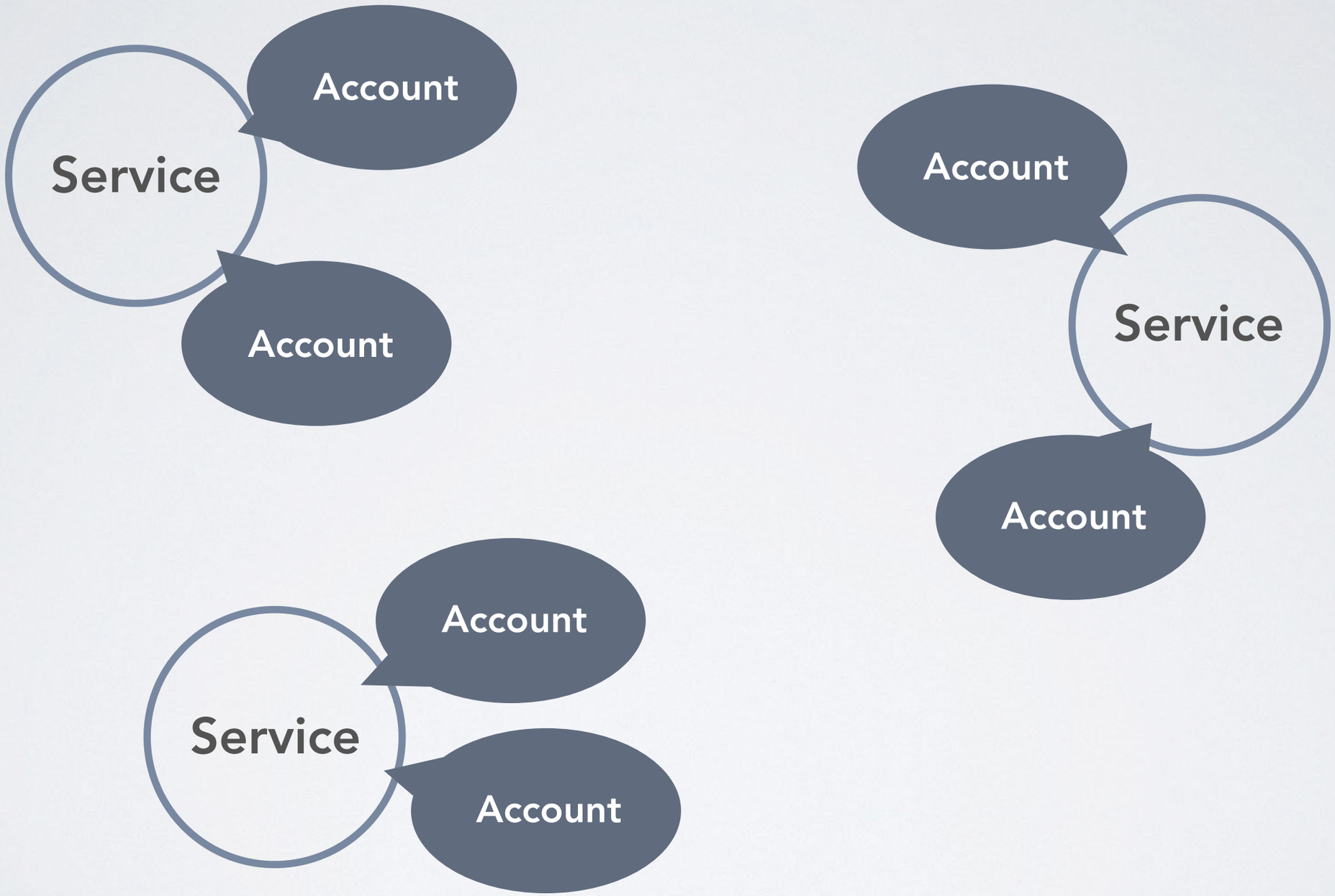
メールアドレスでログイン

▶ 新規登録する

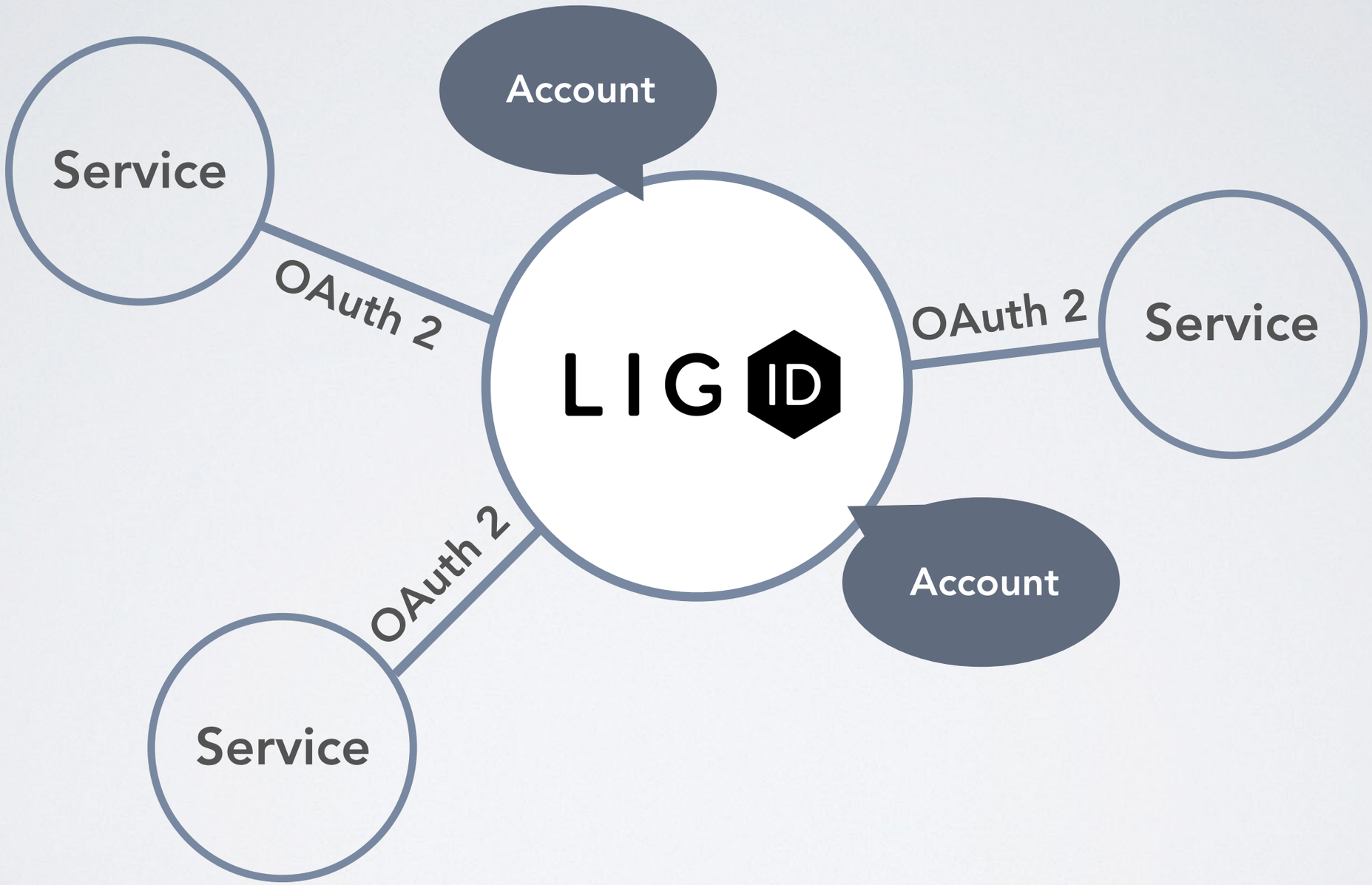
▶ パスワードを忘れた方

# 複数のマイクロサービスに統一のアカウントを提供

# LIG ID がないとき


Service
Account
Account
Account
Service
Account
Service
Account
Account

# LIG ID があるとき


Account
Service
OAuth 2
LIG ID
OAuth 2
Service
OAuth 2
Service
Account


Front end → Angular

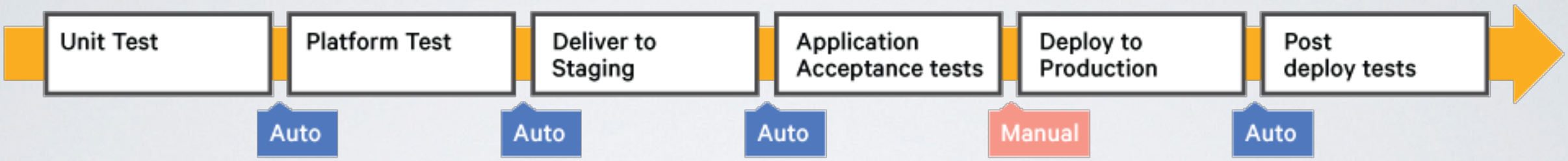
Backend → PHP

API/Core → **Go**

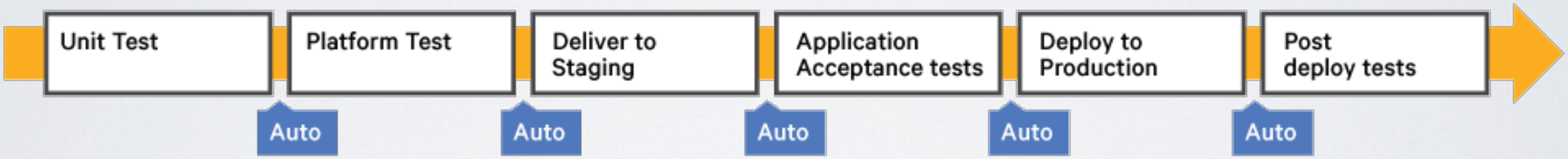
# 今日のお話

# Continuous Deployment

## Continuous Delivery




## Continuous Deployment




https://puppet.com/blog/continuous-delivery-vs-continuous-deployment-what-s-diff

# 最初のデプロイ手順

# ローカルでテスト

ローカルでテスト

ssh

ローカルでテスト

ssh

go get

ローカルでテスト

ssh

go get

go build

ローカルでテスト



ssh



go get



go build



restart

ローカルでテスト

▼ 手動

ssh

▼ 手動

go get

▼ 手動

go build

▼ 手動

restart

いけてない

ローカルでテスト → 時間かかる

ssh

go get

go build

restart

ローカルでテスト → 時間かかる

ssh → 危険

go get

go build

restart

ローカルでテスト → 時間かかる

ssh → 危険

go get → サーバーにソース置く？

go build

restart

ローカルでテスト → 時間かかる

▼

ssh → 危険

▼

go get → サーバーにソース置く？

▼

go build → サーバーでやる？

▼

restart

ローカルでテスト → 時間かかる

▼

ssh → 危険

▼

go get → サーバーにソース置く？

▼

go build → サーバーでやる？

▼

restart → 恐怖

# そうだ自動化しよう

circleci

# CIでテスト自動化

✔ ~~ローカルでテスト~~

▼

ssh

▼

go get

▼

go build

▼

restart

circleci

ついでにビルド

✔ ~~ローカルでテスト~~

▼

ssh

▼

✔ ~~go get~~

▼

✔ ~~go build~~

▼

restart

CodeDeployと組み合わせて

Continuous Deployment


S3

CodeDeploy

EC2

おおまかな流れ





リポジトリに *git push*



S3

CodeD

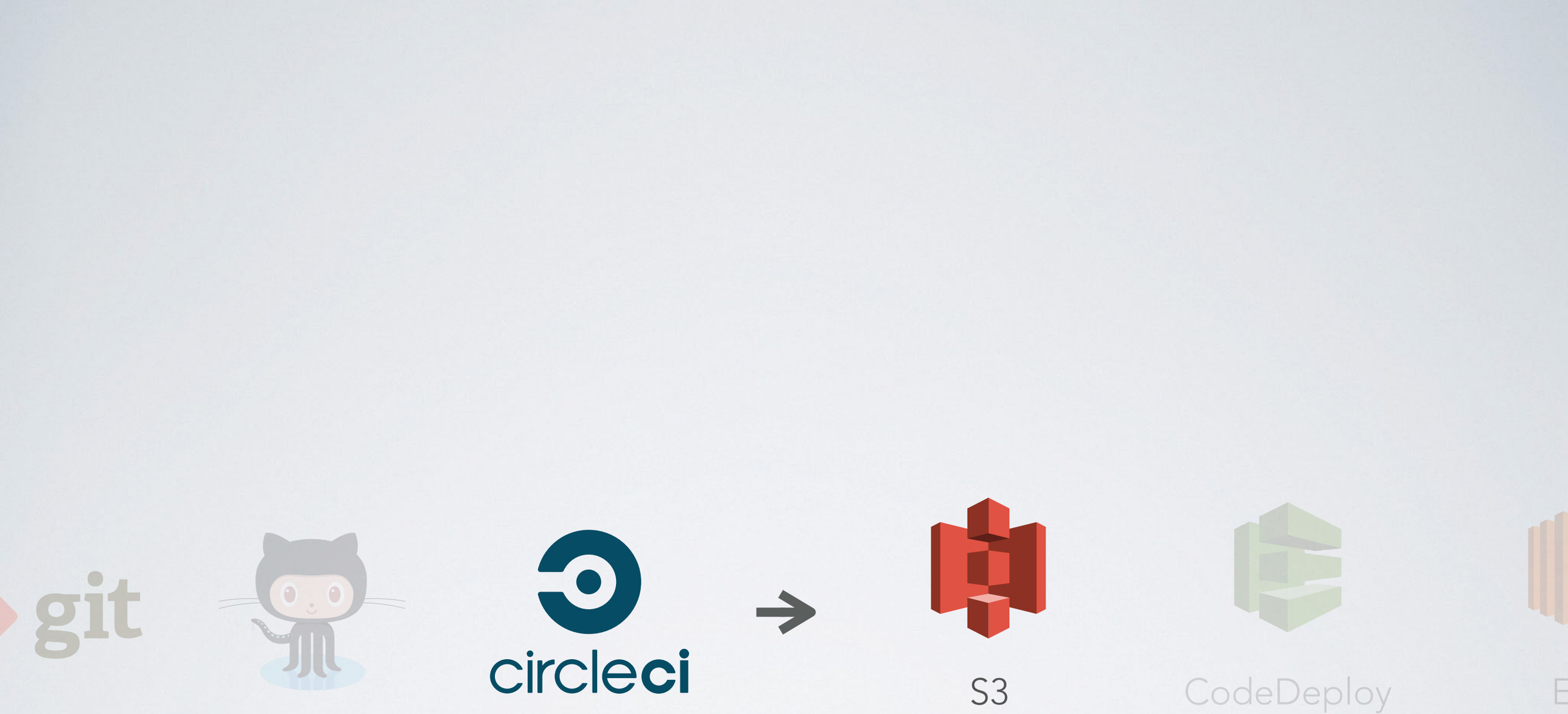
GitHub Hook







ブランチのデータを取得
ビルド＆テスト




デプロイ用のファイル一式をアーカイブ

S3に転送



準備ができたらCodeDeploy呼び出し


S3

CodeDeploy

EC2

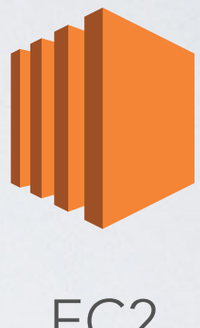
S3からアーカイブをダウンロード

展開・インストール








アプリケーションの再起動


S3

CodeDeploy

EC2

*git push* でデプロイまで実行

✔ ~~ローカルでテスト~~

▼

✔ ~~ssh~~

▼

✔ ~~go get~~

▼

✔ ~~go build~~

▼

? restart

✔ ~~ローカルでテスト~~

▼

✔ ~~ssh~~

▼

✔ ~~go get~~

▼

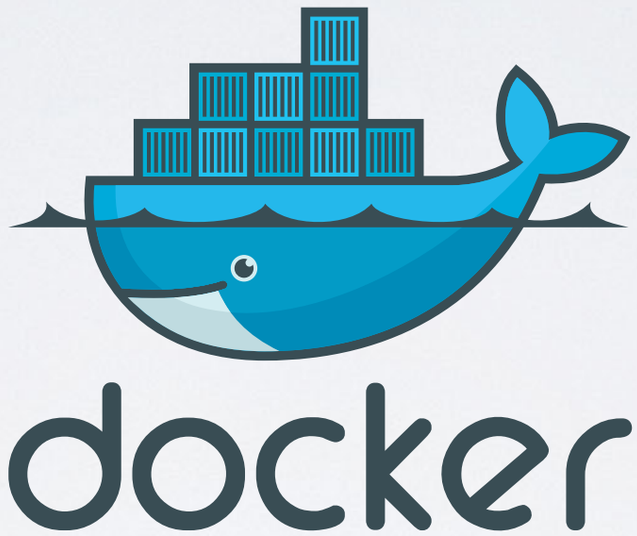
✔ ~~go build~~

▼

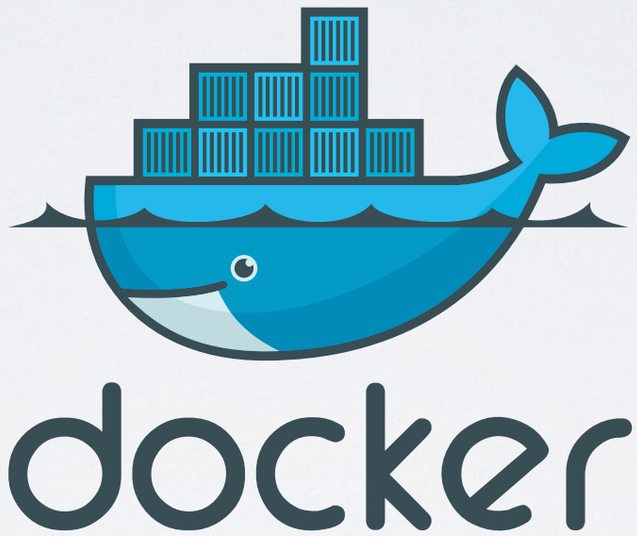
? restart → まだ恐怖

# 本番≠ローカル環境

本番をローカルで再現したい




コンテナによる仮想化



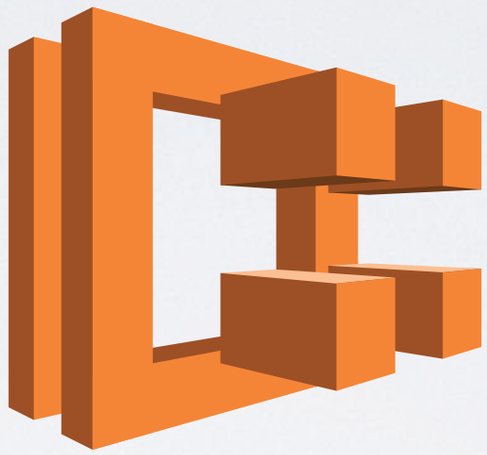

本番とサーバーで同じ環境

しかし…


AWS

GCPじゃない

なにか代わりになるものは…？



EC2 Container Service



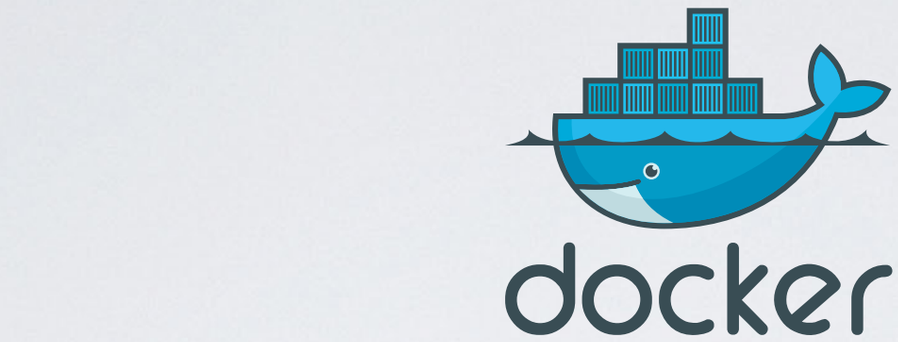
# ECSの仕組み


母艦マシン

Dockerコンテナを走らせる土台

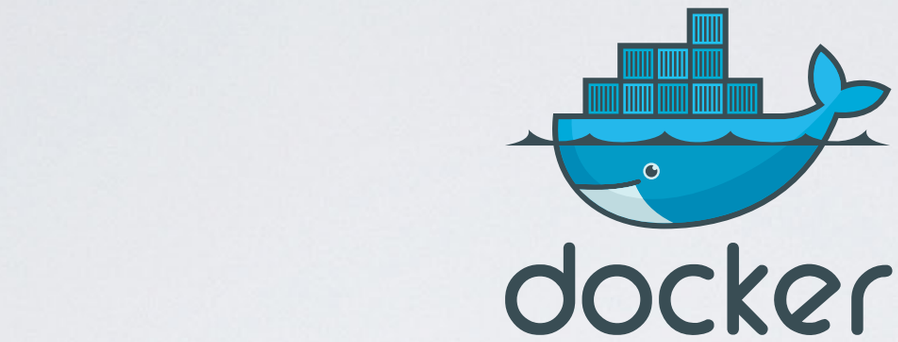

Dockerコンテナのマネジメント

死活監視など





Dockerイメージのプライベート貯蔵庫





3つ合わせてDockerコンテナ管理

これで戦える…！？

しかし現実は甘かった

Write
Test
Run

Build
Test

Run

Write
Test
Run

Build
Test

Run

Write
Test
Run

Build
Test

Run

Write
Test
Run

Build
Test

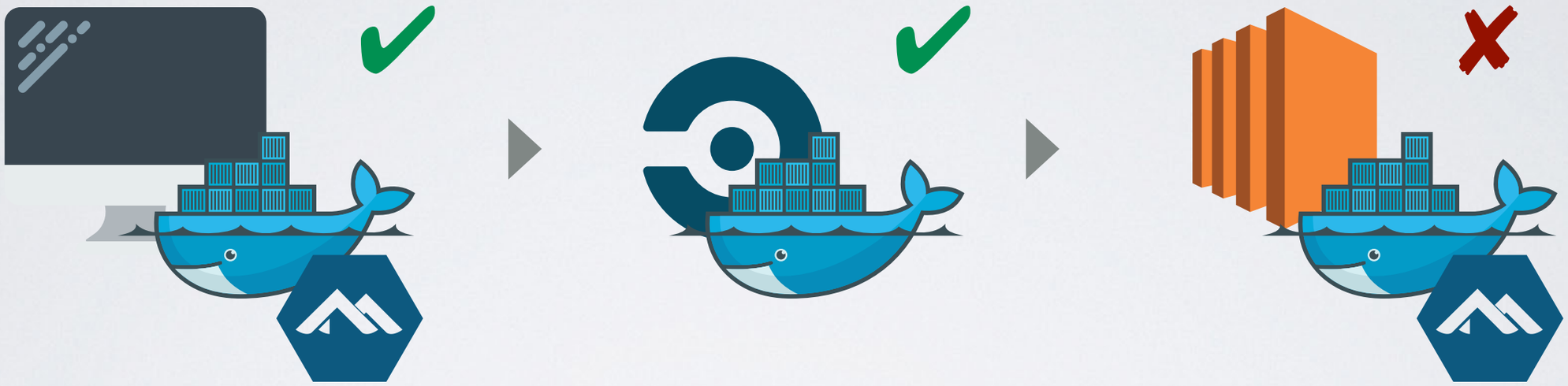
Run

**FAIL**

本番とサーバー **<u>は</u>** 同じ環境

本番とサーバー **は** 同じ環境

Alpine Linux

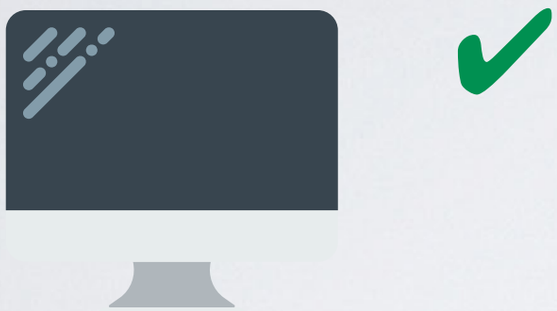


CircleCIも実はDockerベース

ただしUbuntu

**FAIL**

Mac上のDockerでも動かない



macOS

PASS

だがMacでビルドしたファイルはOK

PASS

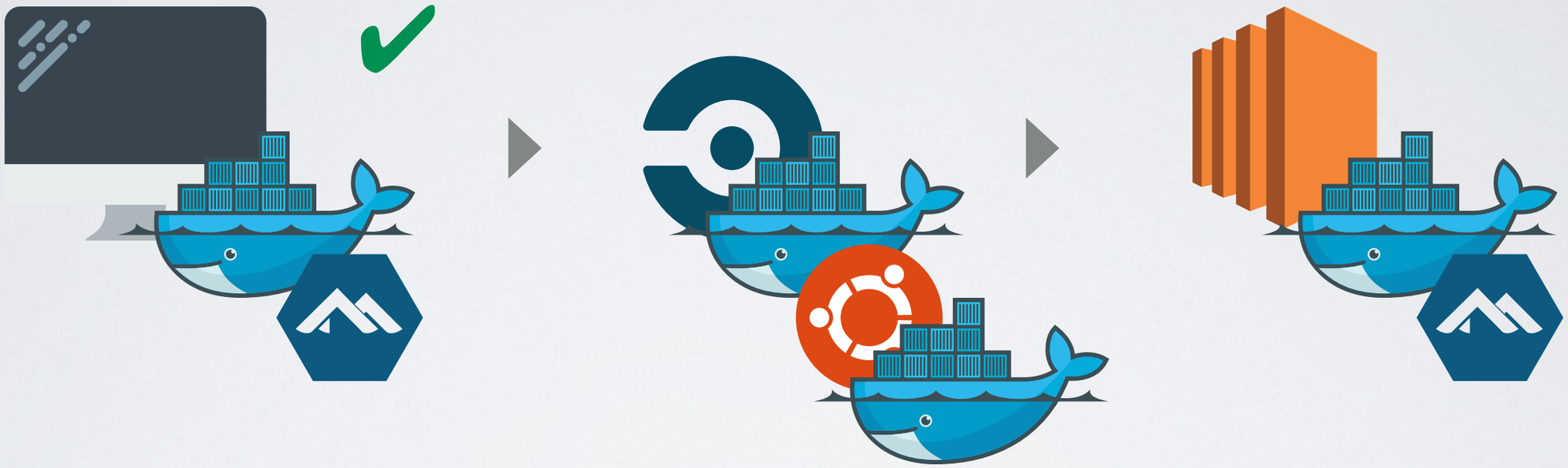
Ubuntuなら動く

原因はこいつ

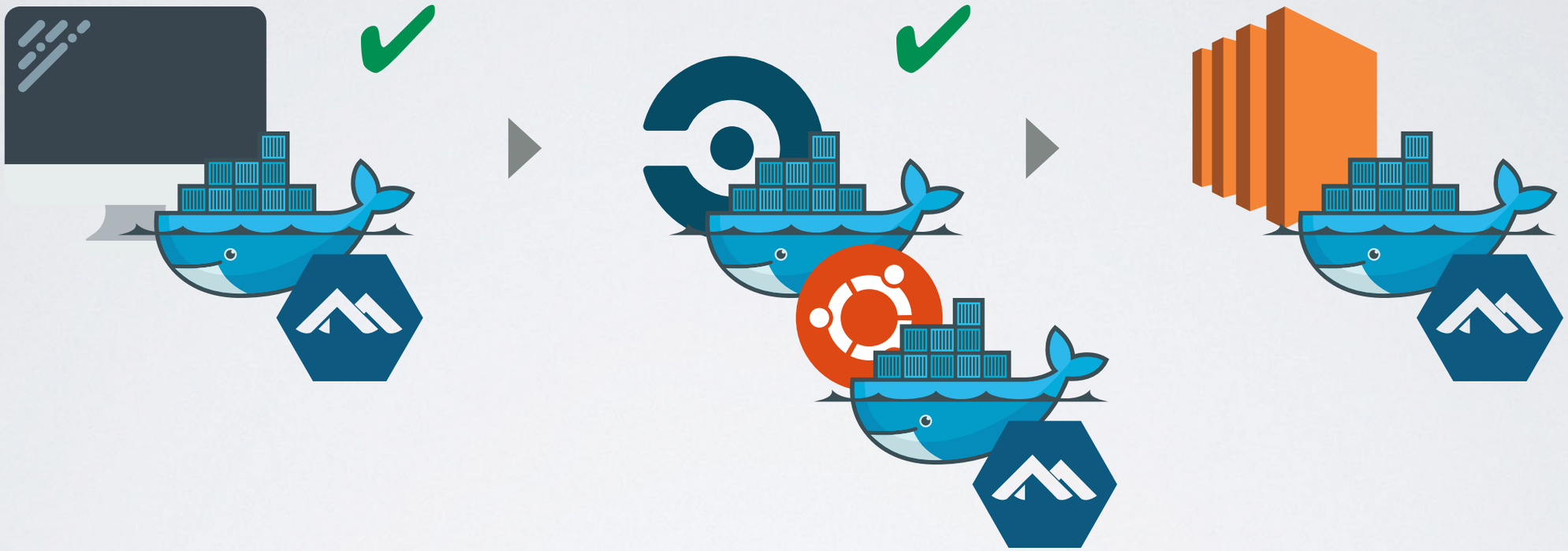
環境依存

じゃあ同じ環境にしよう

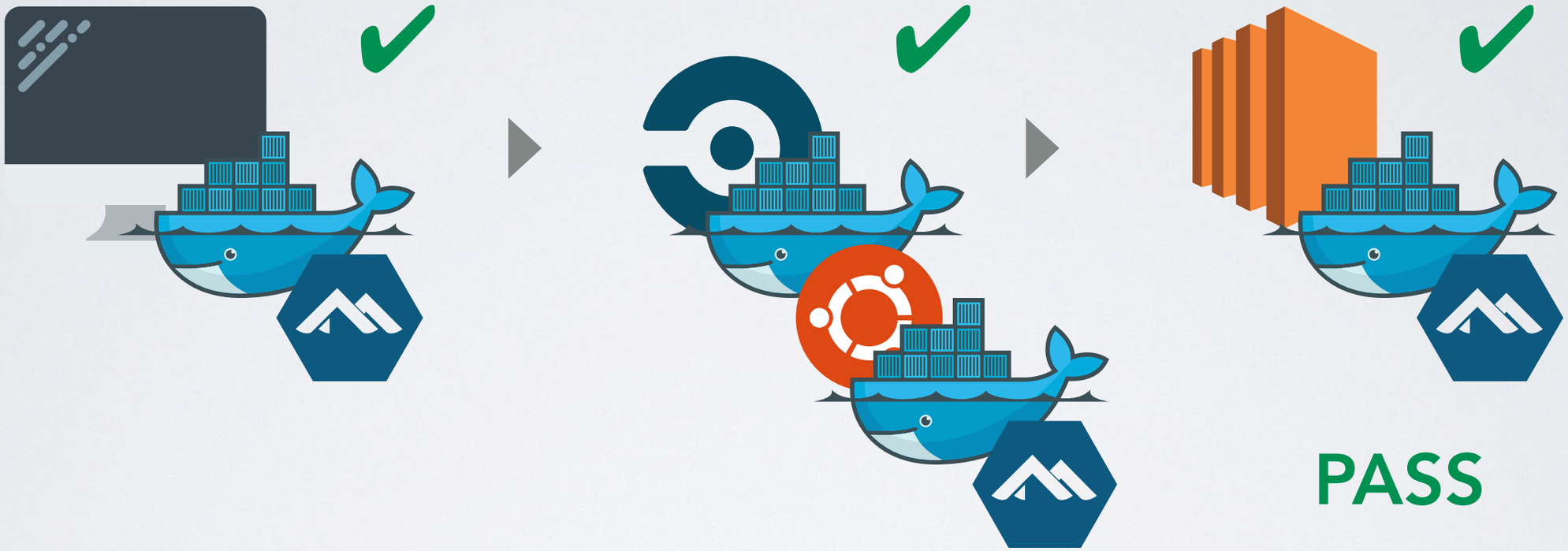
でもCircleCIのコンテナはUbuntuしかない



# Docker on Docker



環境を完全統一




動いた！

✔ ~~ローカルでテスト~~

▼

✔ ~~ssh~~

▼

✔ ~~go get~~

▼

✔ ~~go build~~

▼

✔ ~~restart~~

# デプロイはもう怖くない！

自動化と環境統一で

無駄のないアジャイル生活を

Enjoy

Fearless Deployment!