アプリコードだけではアプリは動かない、JavaだったらJVMが必要

インタプリンタ型であったらそれを解釈できるランタイムが必要

アプリケーションに外部のライブラリを組み込んでいる

複数の環境にわたってアプリを動かす時に一個一個動かすためにコード書き換えていたらオペレーションの負荷が高いし、ミスオペレーションを誘発させてしまう可能性もあるので、多くの場合は設定という形で外だしする

そういった四つの要素が合わさってアプリは動くという仕組み

例えば三つの環境があり（ローカルラップトップ、ステージング、本番）とあり

ローカルではv6.0で本番ではv4.0という状況で、ローカルでは動いたけど本番では動かないという現象が発生する

こういった環境ごとに差異が出てしまうことへの解決策へのひとつとしてコンテナがある（図一枚目）

ランタイムやライブラリ、アプリコードなどバラバラだったものを全てまとめてコンテナの中に一つにまとめてあげる

一つのコンテナにまとめる事で環境に依存せず機能する

※設定をコンテナの中に入れない理由として、設定は環境の差異を埋めるためのもので、コンテナの中にいれてしまうとコンテナ別に分ける必要がでてくるので意味がなくなるので実行時に渡すという仕組みになっている

コンテナの代表的なサービスはDocker、各所にドッカーエンジンを各サーバーに導入しどこでもドッカーが使用できるようになる

つまり、バージョンの差異による実行できないトラブルを回避できる

仮想マシンとコンテナの比較

起動時間

仮想はマシンは１～２分

コンテナは数秒で起動する

仮想マシンとコンテナは似ている、例えば

EC2インスタンスを起動する際 Ami（マシンイメージ）を使用して起動するのに対して

コンテナも同じよういコンテナイメージを指定して実行するとコンテナが起動する

変更を加える時

仮想マシンに何か変更を加えるとき、SSH接続してコマンドをたたく

コンテナもSSH接続ではないが、中に入りyumのようなコマンドをたたく

特徴としてコンテナと仮想マシンはよく似ていて、軽量な仮想マシンと思い込んでしまいがち

実際は違う（図２

簡潔にまとめると仮想マシンはマシンでコンテナはプロセス

Dockerを利用した基本的ワークフロー

１コンテナイメージを作る（必要なものをコンテナイメージに突っ込む

コンテナイメージの作り方は２つある

一つはDockerFileというdslにどういうイメージを作りたいかというコマンド等を書いていく

そのコマンドをドッカービルドというコマンドを実行するとコンテナイメージができる（図３

プロダクションでコンテナイメージを使う場合、多くの場合この方法が取られる

理由はコンテナイメージの作成が再現可能だから

もう一つの方法

まずコンテナを実行するコンテナの中に入り、コマンドを実行して必要なイメージを作って、そのあとにドッカーコミットというコマンドでイメージを固めることができる

図３

イメージができたらイメージレジストリ（githubのようなもの）

イメージレジストリにプッシュしてあげる

これらが基本的なイメージ作成のワークフロー

作成したイメージを実行する

まずサーバーにsshしてdocker pullというコマンドを実行

docker pullはイメージレジストリからイメージをダウンロードしてくるコマンド

ダウンロードしたらdocker runというコマンドでコンテナを実行する

※複数サーバーを持ってる場合は複数のサーバーで同じことをする

ドッカーの責務の範囲は単一のサーバー上でのコンテナのライフサイクル管理

ここまでは以外に手作業でここから自動化してくれるサービスが

コンテナオーケストレーション

コンテナオーケストレーションはdocker pull や docker runなどのコマンドを代わりに発行してくれる仕組み

従来の方法だと何百台とあるサーバーから手作業でコマンドを実行してきて、それだとミスオペや違ったイメージをダウンロードしてしまったなどのトラブルがあったのに対してECSというサービスをリリースした

要は手作業だったものをECSが代わりに実行してくれるもの

例えば

６台のインスタンスの上でコンテナを実行したい時、かつその上でこのコンテナイメージを３つのAZに分散した上で、かつ同時に10個実行する  
実行したあとにELBに繋ぐ

これを自動化するためのソフトがコンテナオーケストレーション

その一つがAmazon ECS

ECSの特徴

AWSの他のサービス（ELBなど）と自動的に連携したり、高度なインテグレーションが実装されている

スケーラビリティもかなり大きな規模感

非常にシンプル（タスクとサービスを押さえておけば利用できる）

Linux/Windows をサポート

kubernetes

EKS

EKSの特徴

・EKSはkubernetesのコードに一切を手を付けていない、アップストリームのコードをそのまあAWS上の環境にデプロイして、それをお客様に提供している