Engineer

Kubernetes

多数のコンテナーの稼働を維持するシステム

どのようなものか

Kubernetesは、K8sとも書くオープンソースのソフトウェアです。クラスターと呼ぶ複数のマシンで、コンテナー化したアプリケーションを動かし、与えた定義に従わせます。各アプリのイメージ、コピー数、各コピーが使えるメモリーとCPU、ほかのプログラムからの接続方法を定めます。プロジェクトの文書は、「コンテナー化されたワークロードとサービスを管理する、移植性と拡張性を持つオープンソースのプラットフォーム」と呼びます。名前はギリシャ語の操舵手や水先案内人を意味します。K8sは、Kとsを残し、その間の八文字を数に置き換えます。Googleが2014年にオープンソースとして公開しました。

クラスターには、二つの部分があります。コントロールプレーンが定義を保持します。APIサーバーは、すべての変更が通る唯一のHTTPインターフェースで、etcdが指示された内容を保存し、スケジューラーが新しい各Podを適切なノードに割り当て、コントローラーマネージャーが、定義に基づく制御ループを動かします。各ワーカーノードでは、kubeletが、割り当てられたPodの稼働を保ち、kube-proxyが、Serviceを支えるネットワーク規則を保ち、containerdやCRI-Oなどのコンテナーランタイムが、コンテナーを起動します。Docker Engineは、Kubernetes 1.24で組み込みランタイムから外れました。Dockerで作ったイメージは、標準のOCIイメージなので、そのまま動きます。

定義は、通常YAMLで書く、オブジェクトの集合です。Podは、Kubernetesが配置する最小単位で、一台のマシン上でネットワークアドレスとストレージを共有する、一つ以上のコンテナーです。Deploymentは、同一のPodをいくつ置くかを示し、テンプレートが変わると、段階的に置き換えます。Serviceは、変わり続けるPodの集合に、安定した一つの名前とアドレスを与え、ラベルでメンバーを選びます。小さなウェブアプリケーションは、次のようになります。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web
spec:
  replicas: 3
  selector:
    matchLabels:
      app: web
  template:
    metadata:
      labels:
        app: web
    spec:
      containers:
        - name: web
          image: registry.example.com/web:1.4.2
          resources:
            requests:
              cpu: 250m
              memory: 256Mi
            limits:
              memory: 256Mi
          readinessProbe:
            httpGet:
              path: /healthz
              port: 8080
```

通常は、より小さな三つの選択肢と比較します。Docker Composeは、一台のマシン上のコンテナーを記述し、そのマシンがなくなれば、コンテナーもなくなります。systemdなどのサービスマネージャーは、異常終了したコンテナーを再起動し、起動時にも立ち上げます。はるかに少ない仕組みで、多くの要求を満たしますが、別のマシンへ仕事を移す機能はありません。Google Cloud RunやAzure Container Appsなどのホステッド基盤は、運用すべきコントロールプレーンを利用者に渡さず、設定項目を減らす代わりに、コンテナーを動かします。文書は、Kubernetesが何ではないかも明示します。あらゆる機能を含む従来型の、サービスとしてのプラットフォーム（Platform as a Service）ではありません。アプリをビルドせず、独自のデータベースやメッセージキューも付属しません。

概念を判断に生かす

「Kubernetesとは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 ノートパソコン、テストサーバー、本番環境で同じ動作になるようにまとめ、忙しい週には処理能力を増やせるようにします。

サービスを見る お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 Google Cloud・AWSでDockerアプリ20個を運用 — 20 デプロイし監視したコンテナアプリケーション

Google Cloud・AWSでDockerアプリ20個を運用 お問い合わせ →

なぜ重要なのか

中心にある考えは、一度だけ命令の一覧を実行するのでなく、記述した状態へ近づくよう、クラスターが継続的に動作することです。上のファイルを適用しても、何かを直接起動するのではなく、望む状態を記録します。コントローラーは、クラスターを監視し、現状を望む状態へ近づける制御ループです。マシンが故障して、三つのコピーが二つになると、コントローラーが不足を見つけ、新しいPodを作り、スケジューラーが余裕のあるマシンに配置します。定義はファイルの集合なので、リポジトリーに置き、その定義で動かすコードと同じレビューを受けられます。

リリースの仕組みも、基盤に付属します。既定はローリング更新です。新しいPodを起動し、それが準備できるにつれて古いPodを止めます。どの時点でも、追加は25%、不足は25%という、既定の上限内で行います。Podテンプレートの各変更をリビジョンとして保ち、既定は十件です。`kubectl rollout undo` で一つ前へ戻せます。以前なら、サーバーの処理を退避し、バイナリーを交換して戻すスクリプトで行ったことが、組み込みの動作になります。

スケジューラーは、サーバー群の経済性も変えます。各Podが要求するCPUとメモリーを基準に置くため、多くの小さなサービスが、一つのマシン群を共有します。それぞれに、最大負荷時に合わせた大きさのサーバーを与える必要がありません。

代償も確かにあります。クラスター自体が分散システムであり、予定を立てて更新するバージョン、ネットワーク層、ストレージ層、証明書、アクセス制御を持ちます。問題が起きれば、すべてを誰かが理解しなければなりません。Google Kubernetes Engine、Amazon EKS、Azure Kubernetes Serviceなどのマネージド製品は、コントロールプレーンを運用しますが、全マニフェストと、通常はワーカーノードも、利用者側の仕事です。一、二台のマシンで少数のサービスを動かすなら、単純なサービスマネージャーの方が、少ない構成要素で済むこともよくあります。

実際の活用例

会社の技術構成ページは、方針を明示しています。使う意義のあるワークロードにはDockerやKubernetesを使い、そうでなければ、単純なsystemdユニットを使います。Kubernetesは既定ではなく、仕事の規模と形に応じて、ワークロードごとに選びます。

ある案件では、導入に見合うワークロードがありました。二十のDockerアプリケーションを、Google CloudとAWSでデプロイし、保守しました。Rを使う分析アプリも含み、どのアプリも特別扱いにならないよう、設定と監視を統一しました。トラブルシューティングにはPodmanを、オーケストレーションの調査にはKubernetesを使いました。同ページは、問題をより速く診断でき、デプロイが安定し、再現可能だったと報告しています。

方針のもう半分を示すのが、会社自身の基盤です。PodmanとQuadletで構築したコンテナー層は、各コンテナーを、OS自身のスーパーバイザーのサービスとして動かし、ホスト上のほかのものと同じ宣言的な形式で定義します。コンテナーはホストネットワークを使い、コンテナー側のポート公開設定は使わないため、強化ロールが導入するファイアウォール規則が、引き続き制御を担います。KubernetesならAPIで新しいイメージを展開するところを、root権限のないデプロイ用ユーザーが、イメージをビルドし、サービスを再起動する、一つの固定スクリプトを実行します。一台のホストでは、各コンテナーに定義を与え、ホストですでに信頼して使っているスーパーバイザーを使えるため、第二のプロセスマネージャーは要りません。ないのは、複数マシンへの配置と、故障ノードの置き換えです。その領域で、Kubernetesの運用の負担に見合う効果が出るようになります。

問題が起きるところ

Kubernetesの問題の大半は、マニフェストが求めるものと、クラスターが報告するものの差から生まれます。

デプロイジョブは成功と報告するのに、展開は途中で止まります。 誤ったイメージタグ、起動時の異常終了、失敗する準備確認（readiness）プローブ、割当量の不足で、新しいPodが準備できなければ、Deploymentの進行が止まります。進行期限、既定の600秒を過ぎると、理由を `ProgressDeadlineExceeded` と記録します。文書は、Kubernetesはそれ以上の処置をしないと述べています。残った古いPodが配信を続け、自動ではロールバックしません。 `kubectl rollout status` は、期限後に非ゼロの終了コードを返すため、デプロイのステップで待ち、リリースを取り消せます。

```
kubectl apply -f web.yaml
kubectl rollout status deployment/web --timeout=10m \
  || kubectl rollout undo deployment/web
```

`OOMKilled` **でコンテナーが再起動するか、CPUに余裕のあるノードで遅くなります。** 上限はカーネルが強制し、二つの資源で動作が異なります。メモリー上限を超えるコンテナーは、終了させられ、再起動することがあります。CPU上限に達したものは、抑制されるため、停止せず遅くなり、アプリの問題に見えます。スケジューラーが基準にするのは、実際の使用量ではなく要求量です。要求を低くすると、ノードに詰め込みすぎ、高くすると、マシンが遊びます。要求量を指定せず上限だけ与えると、その値が要求量になります。メモリーは計測した使用量から設定し、CPU上限は、意識して判断します。

起動の遅いアプリが、起動完了前に終了させられます。 生存確認（liveness）プローブが失敗すると、kubeletはコンテナーを終了させ、再起動します。そのため、読み込みに一分かかるアプリを十秒後に検査すると、再起動を繰り返します。起動確認（startup）プローブは、アプリが立ち上がるまで、ほかの検査を待たせます。readinessプローブは、再起動せず、PodをServiceから外すため、処理中や、キャッシュの準備をしているPodに向きます。データベースを呼ぶlivenessプローブは、データベースが遅いと全レプリカを同時に再起動し、依存先の遅さを障害に変えてしまいます。

手で起動したPodが消え、手で行った修正が戻ります。 コントローラーを介さず単独で作ったPodは、終了、削除、ノードからの退去、ノード故障まで、そのノードに残り、コントローラーが再作成することはありません。文書は、一つのコピーでもDeploymentかJobを勧めています。 `kubectl edit` の変更は、クラスター内だけに残り、次のリポジトリーからの適用が古い値へ戻すため、緊急修正は、緊急事態が終わり次第、ファイルにも反映します。

参考資料

- Kubernetes概要 — Kubernetesとは何か、何を提供し、何を含めないかについて、プロジェクト自身が説明しています。
- Pod — 配置の最小単位、ライフサイクル、手で作ることが少ない理由です。
- Deployment — ローリング更新、ロールバック、リビジョン履歴、進行期限です。
- Service — 変わり続けるPod集合の前に置く、安定した名前とアドレスです。

- Podとコンテナーの資源管理 — 要求、上限、カーネルによる各資源の制御です。
- 生存確認・準備確認・起動確認プローブの設定 — 三つのプローブと、起動の遅いコンテナーを保護する方法です。

用語集

実際の制約に合わせて選ぶ

デプロイの流れ、運用上のリスク、緊急対応の担当者をお知らせください。検査、復旧手段、明確な責任の所在を伴う変更の範囲を定められます。

料金を見る お問い合わせ →

用語を実務に結び付ける

「Kubernetesとは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く

連絡先カード

心を込めて丁寧に制作： D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/kubernetes/