



AWS Managed Container Service ECS or EKS or ROSA ?? + ROSA HCP 紹介

Red Hat K.K.

2023.12

Agenda

- ECS/EKS/ROSAの概要とライフサイクル
- EKS/ROSAサンプルユースケース その1
- EKS/ROSAサンプルユースケース その2
- ROSA Hosted Control Plane サービス仕様
- AWSサービスとの連携
- サポート体制
- Appendix:
AWSに持ち込むRed Hat製品 Bring your Own Subscription (BYOS)

ECS/EKS/ROSAの 概要とライフサイクル

AWS上でのコンテナサービスの選択肢

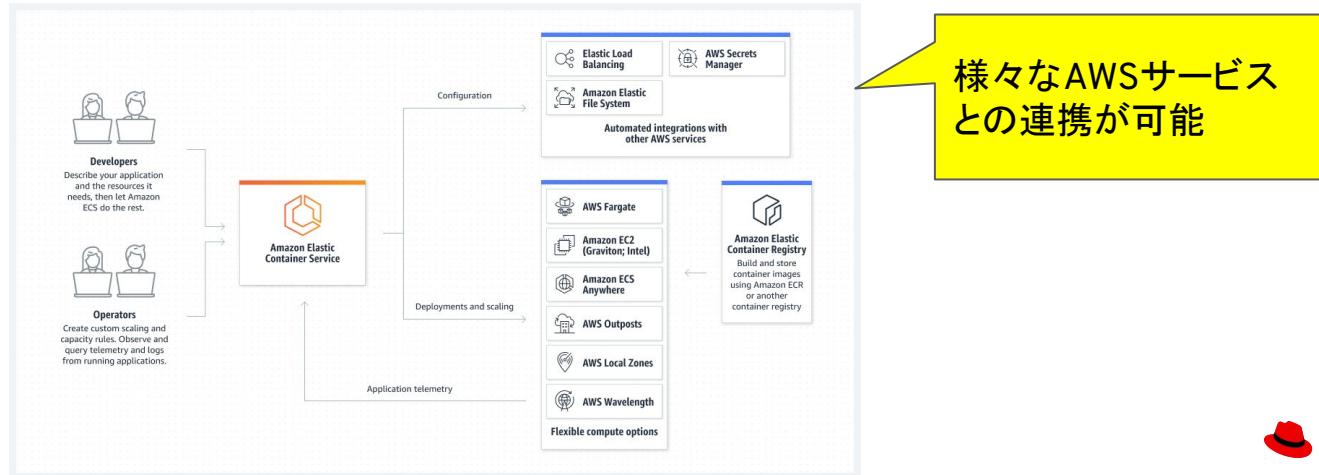
<https://aws.amazon.com/jp/getting-started/decision-guides/containers-on-aws-how-to-choose/>

コンテナ	使用するタイミングは?	どのような用途に最適化されていますか?	関連するコンテナサービスまたはツール
容量 ▾	セルフマネージド型の AWS 仮想マシンや AWS マネージドコンピューティングでコンテナを実行する場合に使用します。	AWS コンピューティングでコンテナを実行するのに最適化されています。	AWS Fargate ▾ Amazon EC2 ▾
オーケストレーション ▾	最大数千のコンテナをデプロイして管理する必要がある場合に使用します。	AWS でコンテナ化されたアプリケーションをデプロイ、管理、スケーリングするのに最適化されています。	Amazon ECS ▾ Amazon Elastic Kubernetes Service ▾ Red Hat OpenShift Service on AWS (ROSA) ▾
プロビジョニング ▾	ユーザーや自身のチームがコンテナやインフラストラクチャを使用した経験があまりない場合に使用します。	使いやすさを考慮して最適化されています。	AWS App Runner ▾ Amazon Lightsail ▾ AWS Elastic Beanstalk ▾
ツール ▾	コンテナレジストリを提供するだけでなく、既存のアプリケーションをコンテナ化および移行するツールが必要な場合に使用します。	コンテナ運用のサポートに最適化されています。	Amazon Elastic Container Registry ▾
オンプレミス ▾	使い慣れたコントロールプレーンを実行する必要がある場合に使用すると、コンテナベースのアプリケーションがどこで実行されていても一貫したエクスペリエンスを実現できます。	コンテナベースのアプリケーションを実行する場所に柔軟性を持たせるように最適化されています。	Amazon Elastic Container Service (ECS) Anywhere ▾ Amazon EKS Anywhere ▾ Amazon EKS Distro ▾

コンテナオーケストラ
トレーニングサービス
にフォーカス

Amazon ECS

- マネージドのコンテナオーケストレーションサービス
- AWSネイティブな方法で、Dockerフォーマットのコンテナの大規模実行が可能
 - 様々なAWSサービスとの連携を、迅速に設定して利用可能
 - コンテナクラスター利用料金が無いことによる、コスト最適化が可能
 - コントロールプレーンは完全に管理され、利用者によるアップグレードなどの対応は不要
- **AWS上でのアプリケーションを開発/実行するユースケースを想定**
 - CNCF(後述)のプロジェクトを利用したい Kubernetesユーザーには非推奨



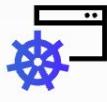
Cloud Native Computing Foundation (CNCF)

<https://www.cncf.io/about/who-we-are/>

- CNCFはクラウドネイティブコンピューティング技術を推進する非営利団体であり、Linux Foundationプロジェクトの1つ
- 2015年のGoogleによるKubernetes v1発表と同時に、CNCFも発表
 - <https://cloudplatform.googleblog.com/2015/07/Kubernetes-V1-Released.html>
- AWSとRed Hatを含む多くの企業が参画しており、業界標準ソフトウェアとなっているKubernetesの発展と密接にリンク
- Amazon EKSの提供は、AWS顧客のKubernetesサービス需要によるもの
 - <https://aws.amazon.com/jp/blogsopensource/cloud-native-computing/>

Kubernetes and other CNCF projects have quickly gained adoption and secured diverse community support, becoming some of the highest velocity projects in the history of open source.

2023年7月時点のCNCF



162

CNCF Projects



205K+

CNCF Project
Contributors



811

CNCF
Members



59K+

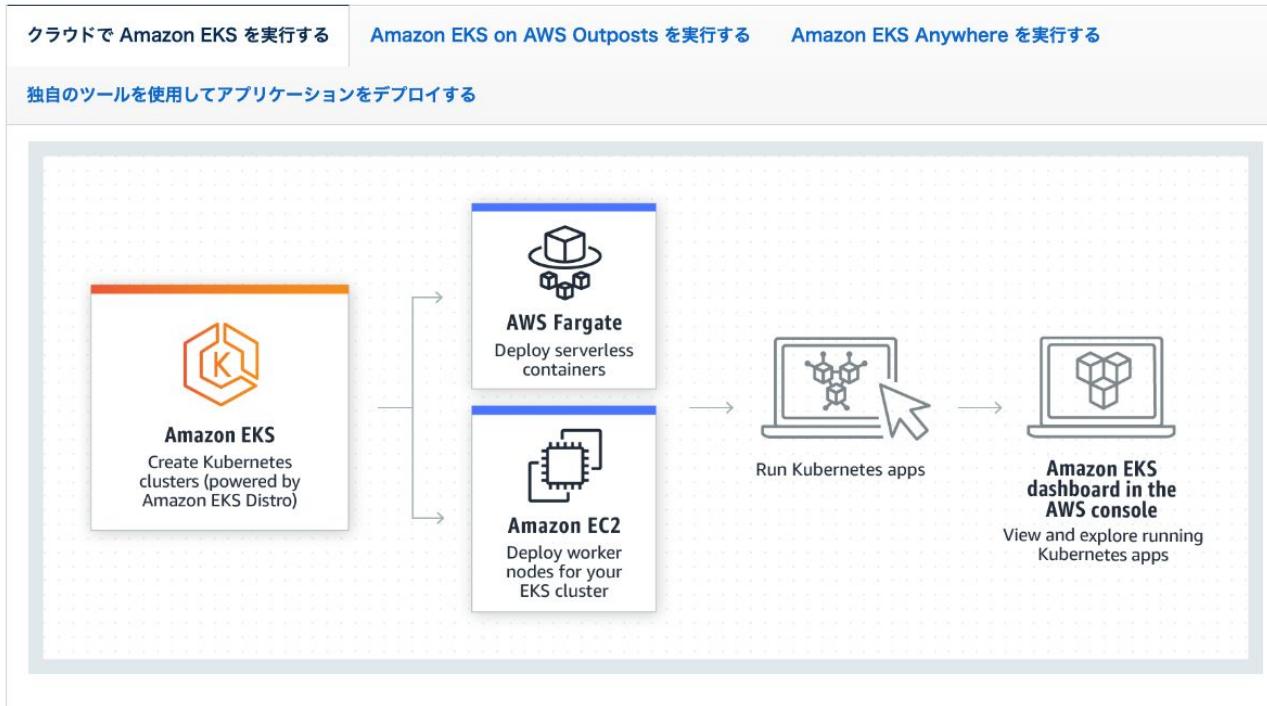
Cloud Native Community
Members



Amazon EKS

<https://aws.amazon.com/jp/eks/>

- AWS クラウドおよびオンプレで Kubernetes を実行するためのマネージド Kubernetes サービス



Red Hat OpenShift と Red Hat OpenShift Service on AWS (ROSA)

<https://www.redhat.com/ia/technologies/cloud-computing/openshift/red-hat-openshift-kubernetes>

- Red Hat OpenShift
 - Red Hat Enterprise Linux (RHEL) と並ぶ主力製品の1つ
 - RHELと統合されており、RHELをベースとしたコンテナ専用OSの上で動作
 - Kubernetesがベース
 - エンタープライズシステムで必要となる、様々な開発/運用関連の機能が統合
 - 前述のCNCFプロジェクトの機能を多数搭載
- ROSA
 - AWS上のマネージドOpenShiftサービス
 - Red HatとAWSによるサポートを提供



サービス基盤の機能実装に必要なAWSサービス

ROSAにより、基盤構築に必要な時間の短縮が可能

要件に応じたAWSサービスの取捨選択と、EKSからそのサービスを利用する場合、EKSクラスター内での、AWS IAMによる権限設定が別途必要

Amazon EKSを使う場合

- Dashboard
- Developer IDE/tools
- Build Automation
- Pipeline (CI)
- Deployment Automation (CD)
- Serverless
- Service Mesh
- Monitoring
- Logging
- Registry
- Orchestration
- Infrastructure

- Amazon EKS Console
- AWS Cloud9 / Amazon CodeCatalyst
- AWS CodeBuild
- AWS CodePipeline
- AWS Lambda
- AWS AppMesh / AWS X-Ray
- Amazon Managed Service for Prometheus
- Amazon CloudWatch
- Amazon ECR
- Amazon EKS
- AWS Compute Resources

ROSAを使う場合

ROSA 標準機能

- Built in Console
- OpenShift Dev Spaces
- S2i (Source-to-Image) Build
- OpenShift Pipelines
- OpenShift GitOps
- OpenShift Serverless
- OpenShift Service Mesh
- OpenShift Monitoring
- OpenShift Logging
- OpenShift Internal Registry
- Kubernetes

Kubernetesによる
システム作り込み時に
必要な主要機能

ROSA提供の
OpenShift標準機能
を利用可能

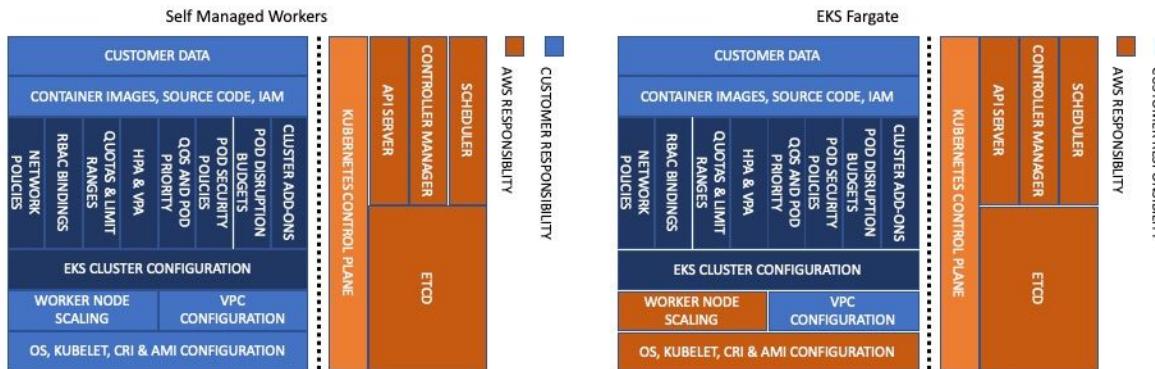
OpenShiftでは、統合された認証/認可機能により、各標準機能を利用するための権限設定がデフォルトで適用済み

Amazon EKS Shared Responsibility Model

<https://aws.github.io/aws-eks-best-practices/security/docs/>

- AWSが管理/サポート
 - Amazon EKS コントロールプレーン
 - Amazon EKS ワーカーノード (Fargate, Managed Node Groups)
 - Amazon EKS アドオン (CoreDNS, Kube-proxyなど)
- ユーザーが管理※
 - Amazon EKS ワーカーノード (Self Managed Workers)
 - EKSに含まれない、CNCFなどが提供するKubernetes (K8s) 用アドオン
 - K8sのセキュリティ設定 (Pod, Image, Network, RBAC, Multi-tenancyなど)
 - ユーザーアプリやデータ、および、それらの実行基盤となるコンテナ上のミドルウェア

EKSは、Kubernetesクラスターの管理を、一部委任したいという希望者向けのサービス
(クラスターの作成/削除/更新をセルフサービスで実施可能)



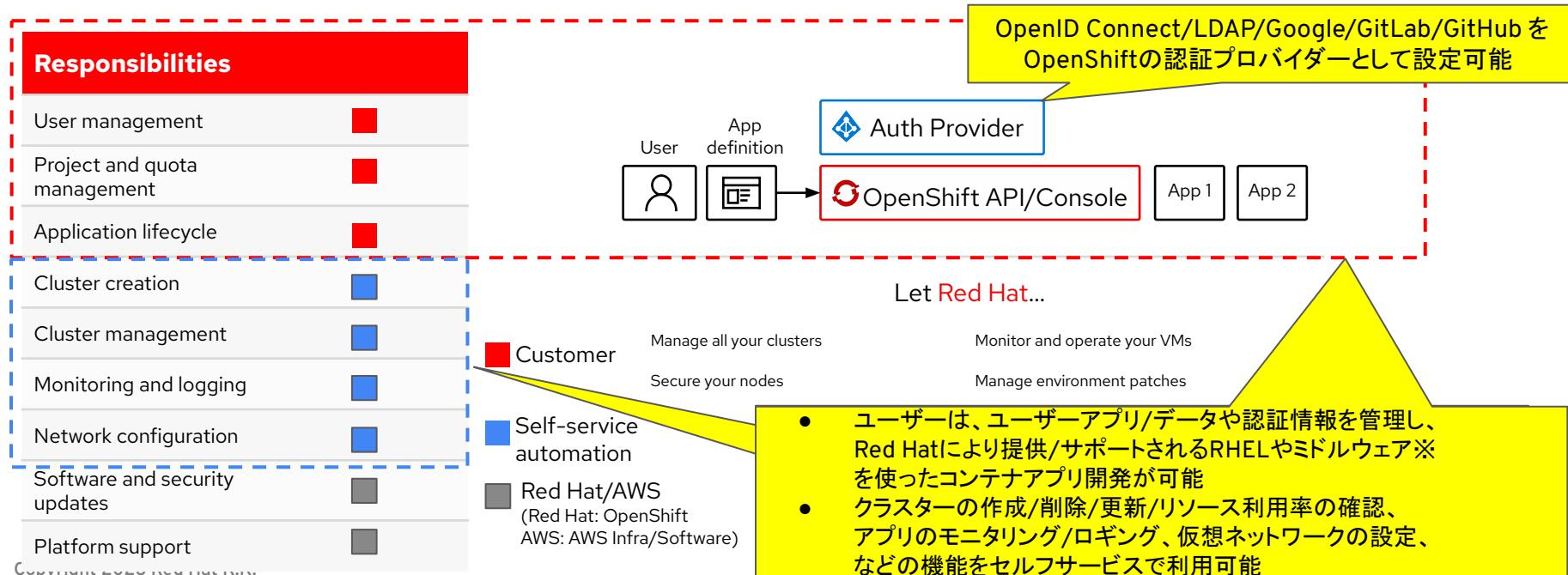
ROSA Shared Responsibility Model

https://docs.openshift.com/rosa/roса_architecture/roса_policy_service_definition/roса-policy-responsibility-matrix.html

- Red HatとAWSが管理/サポート

- コントロールプレーン/ワーカーノード、K8s Network/DNS関連機能、ロードバランサーを含めたクラスター管理
- OpenShiftは予めセキュリティが強化されたK8sとして設定され、Red Hatがその利用をサポート
(Pod, Image, Network, RBAC, Multi-tenancy, ホストOSなど)

ROSAは、KubernetesからコンテナのOS、ミドルウェアまでの統合サポートの希望者向けのサービス



EKS/ROSA ライフサイクル (2024年3月時点)

EKS: 最低14ヶ月※、ROSA: 16ヶ月のサポートを提供

Kubernetes Version	Upstream release	EKS release	EKS End of standard support	EKS End of extended support (注: クラスター料金が 6倍になります)
1.29	2023年12月13日	2024年1月23日	2025年3月23日	2026年3月23日
1.28	2023年8月15日	2023年9月26日	2024年11月26日	2025年11月26日
1.27	2023年4月11日	2023年5月24日	2024年7月24日	2025年7月24日
1.26	2022年12月9日	2023年4月11日	2024年6月11日	2025年6月11日
1.25	2022年8月23日	2023年2月22日	2024年5月1日	2025年5月1日
1.24	2022年5月3日	2022年11月15日	2024年1月31日	2025年1月31日
1.23	2021年12月7日	2022年8月11日	2023年10月11日	2024年10月11日

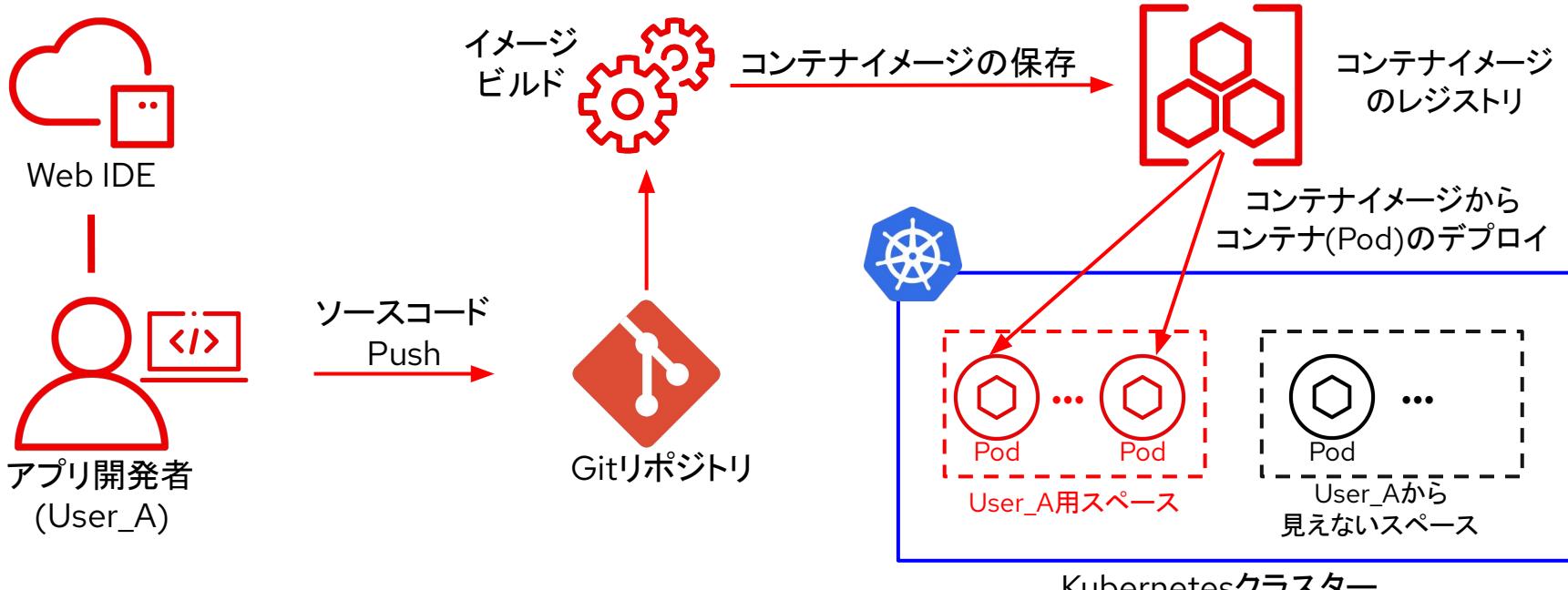
Kubernetes version	ROSA Version	ROSA release	ROSA End of Life
1.27	4.14	2023年10月31日	2025年2月28日
1.26	4.13	2023年5月17日	2024年9月17日
1.25	4.12	2023年1月17日	2024年5月17日

※ EKSでは、最低4つのKubernetesバージョンをサポートするように定義しており、Upstreamのリリース状況に応じて、14ヶ月以上のサポートを提供する場合があります。

EKS/ROSA サンプルユースケース その1

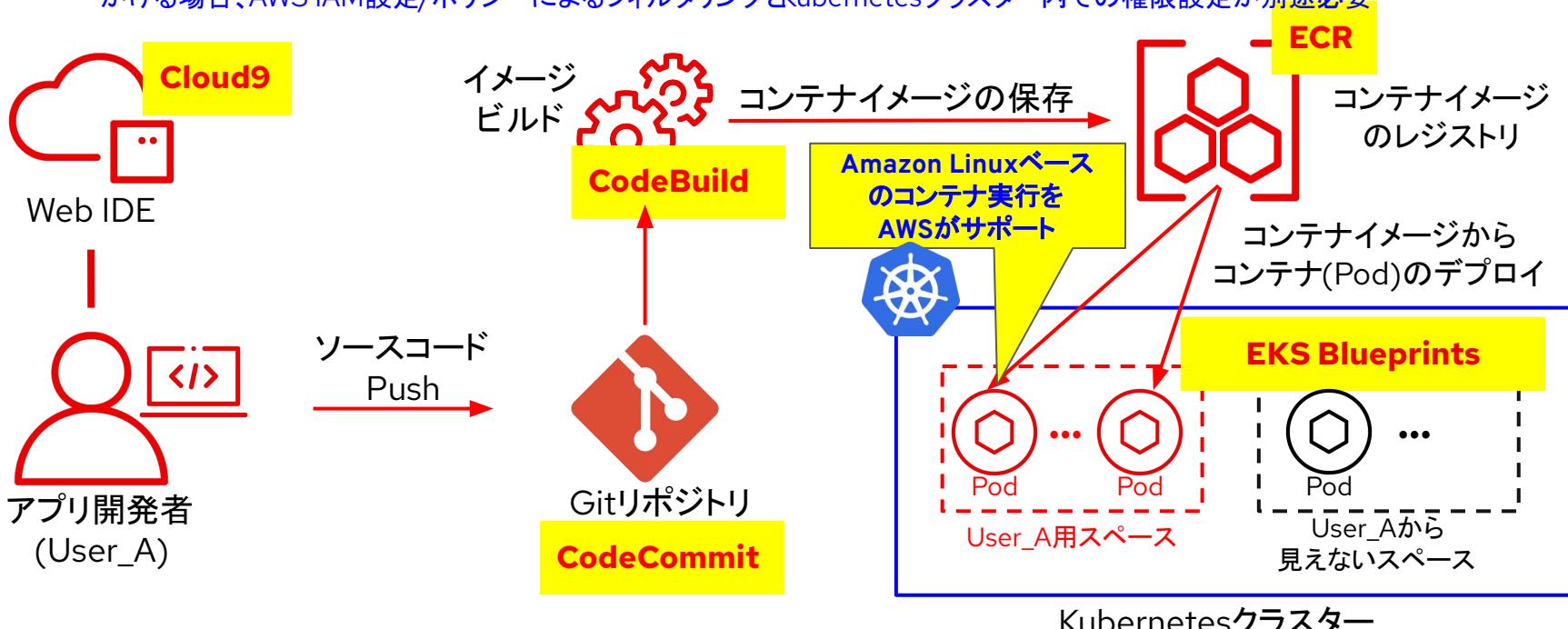
EKS/ROSA のサンプルユースケース その1

- Web IDEを使ったコンテナアプリの開発とデプロイ
- 開発者は、自分の開発/デプロイしたコンテナアプリしか見えない状態を想定



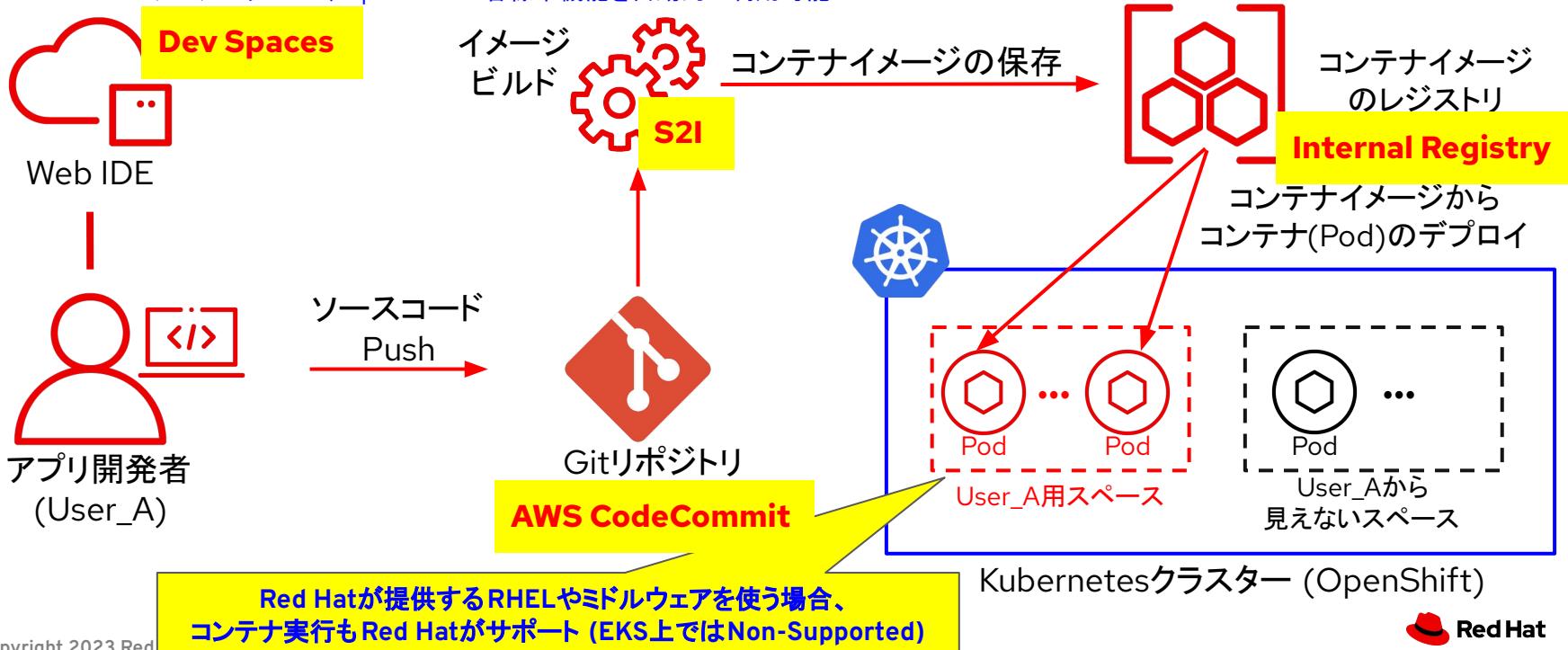
EKSを使う場合のイメージ

- 他のAWSサービス(AWS Cloud9/CodeCommit/CodeBuild/ECR)を合わせて利用
- EKSのマルチテナント化には、EKS Blueprints※というオープンソースプロジェクトを利用可能
- 各AWSサービスを利用するためのAWS IAM設定(認証/認可)と、ユーザーごとに参照可能なECRレジストリの制限をかける場合、AWS IAM設定/ポリシーによるfiltreringとKubernetesクラスター内での権限設定が別途必要



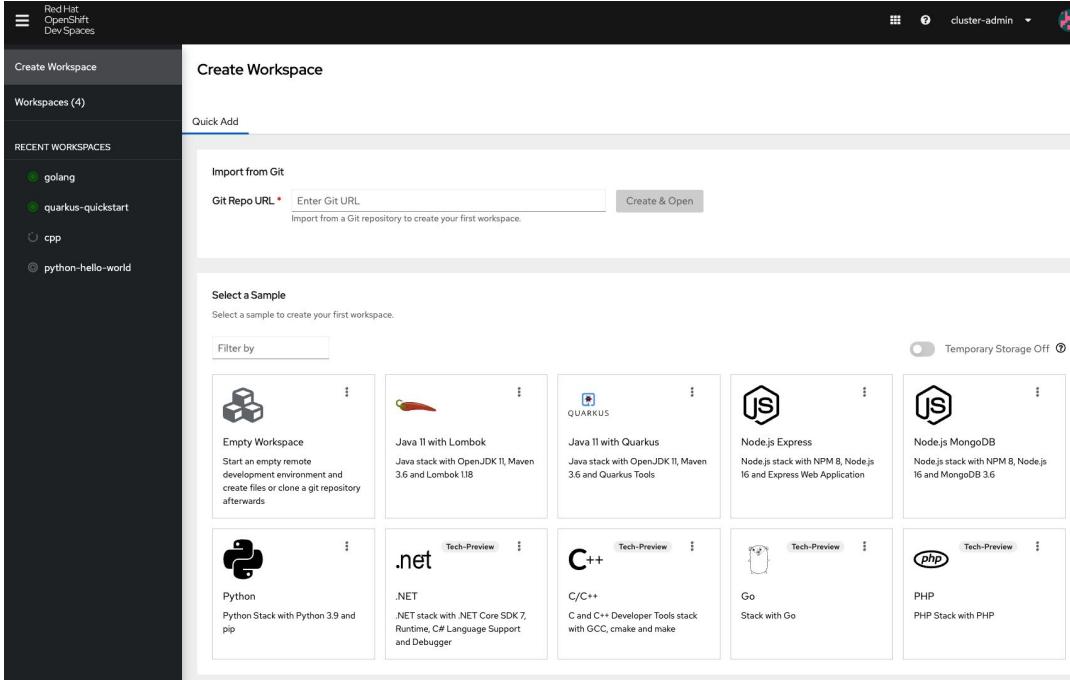
ROSAを使う場合のイメージ

- ROSAにあるOpenShift標準機能 (OpenShift Dev Spaces/S2I/Internal Registry)を利用
- Gitリポジトリには、AWS CodeCommitを利用 (AWS IAM設定が必要)
- OpenShiftではマルチテナント化を考慮しており、ユーザーごとのスペースやレジストリがデフォルトで隔離済み
- OpenShiftに統合された認証/認可機能により、OpenShiftクラスターと連携した認証プロバイダー(OpenID Connectなど)のユーザー アカウントで、OpenShiftの各標準機能を自動的に利用可能



OpenShift Dev Spaces用のテンプレート

- 標準テンプレートの他に、カスタマイズされたWeb IDEも利用可能
- カスタマイズには専用のテンプレート(Devfile)※を利用



※ Devfile: Web IDEを定義するフォーマット(YAML)であり、ベースコンテナイメージやカスタムコマンドを定義。

OpenShift Dev Spacesの画面

(Visual Studio Code的な Web IDE)

The screenshot shows the OpenShift Dev Spaces interface, which is a web-based IDE designed to look like Visual Studio Code. On the left is a sidebar with various icons and toolbars for AWS services like Lambda, CloudWatch, and CloudFormation. The main area is a code editor with a dark theme, currently displaying a Java file named GreetingResource.java. The code contains annotations for GET methods and produces plain text. A yellow callout box highlights the use of the Amazon CodeWhisperer plugin for code completion, specifically pointing to the @Path annotation at line 30. The status bar at the bottom provides information about the current session, including the number of spaces (4), the builder ID, and the fact that CodeWhisperer is active.

```
21
22     @GET
23     @Produces(MediaType.TEXT_PLAIN)
24     public String hello() {
25         return "hello";
26     }
27
28     // return bye
29     @GET
30     @Path("/bye")
31     @Produces(MediaType.TEXT_PLAIN)
32     public String bye() {
```

Visual Studio Code プラグインの、
Amazon CodeWhisperer によるコード補完も利用可能

OpenShift S2I に利用可能なカタログ

Project: test-project01 ▾

開発者カタログ > ビルダーイメージ

ビルダーイメージ

特定の言語またはフレームワークをサポートするコンテナイメージについて参照します。クラスター管理者は、カタログで利用可能にされるコンテンツをカスタマイズできます。

すべての項目 Languages Middleware その他

すべての項目 キーワードでフィル... A-Z 12 項目

 ビルダーイメージ .NET Red Hat による提供 Build and run .NET 7 applications on UBI 8. For more information about using this builder image,...	 ビルダーイメージ Apache HTTP Server (httpd) Build and serve static content via Apache HTTP Server (httpd) 2.4 on RHEL 7. For more information about using this builder image, includin...	 ビルダーイメージ Go Build and run Go applications on UBI 7. For more information about using this builder image, includin...	 ビルダーイメージ JBoss EAP XP 3.0 with OpenJDK 11 Red Hat による提供 JBoss EAP expansion pack 3.0 image for OpenShift to build and run Microservices applications o...
 ビルダーイメージ JBoss EAP XP 4.0 with OpenJDK 11 Red Hat による提供 JBoss EAP expansion pack 4.0 image for OpenShift to build and run Microservices applications o...	 ビルダーイメージ Nginx HTTP server and a reverse proxy (nginx) Build and serve static content via Nginx HTTP server and a reverse proxy (nginx) on RHEL 7. For...	 ビルダーイメージ Node.js Build and run Node.js 16 applications on UBI 8. For more information about using this...	 ビルダーイメージ Perl Build and run Perl 5.32 applications on UBI 8. For more information about using this...
 ビルダーイメージ PHP Build and run PHP 8.0 applications on UBI 8. For more information about using this...	 ビルダーイメージ Python Build and run Python 3.9 applications on UBI 8. For more information about using this...	 ビルダーイメージ Red Hat OpenJDK Red Hat, Inc. による提供 Build and run Java applications using Maven and OpenJDK 17.	 ビルダーイメージ Ruby Build and run Ruby 3.0 applications on UBI 7. For more information about using this...

OpenShift S2I の画面

The screenshot shows the Red Hat OpenShift Service on AWS interface for creating a Source-to-Image (S2I) application. The left sidebar lists builder image versions: IST 3.9-ubi8, IST 3.9-ubi8, IST 3.9-ubi9, IST 3.8-ubi7, IST 3.8-ubi8, IST 3.6-ubi8, IST 2.7-ubi7, IST 2.7-ubi8, and IST latest. The main panel is titled "Source-to-Image (S2I) アプリケーションの作成". It includes fields for "Builder Image Version" (set to IST 3.9-ubi8), "Python 3.9 (UBI 8)" (selected as BUILDER PYTHON), "Git Repository URL" (set to https://github.com/sclorg/django-ex.git), and a "Sample" button. A yellow callout box highlights the "Builder Image Version" field with the text "コンテナイメージビルト用のコンテナの選択". Another yellow callout box highlights the "Git Repository URL" field with the text "ソースコード置き場となる GitリポジトリのURL". A third yellow callout box covers the entire main panel with the text "「作成」クリックにより、アプリ作成と公開を自動実行". On the right, a "General" configuration panel shows the application name as "sample-python" and the resource type as "Deployment". It also features a "Route" checkbox under "Advanced Options" which is checked and highlighted with a red dashed box, along with the text "アプリ公開用URLの作成と公開の自動実行を指定するオプション". A fourth yellow callout box highlights this section with the text "アプリ公開用URLの作成と公開の自動実行を指定するオプション". At the bottom right are "Create" and "Cancel" buttons.

一般

アプリケーション

sample-python

このコンポーネントをグループ化するアプリケーションを選択します。

名前 *

sample-python01

コンポーネントに指定する一意名で、関連リソースに名前を付けるのに使用します。

リソースタイプ

Deployment

Resource type to generate. The default can be set in [User Preferences](#).

詳細オプション

ターゲットポート

8080

トラフィックのターゲットポート。

route を作成する
パブリック URL でコンポーネントを公開します

詳細なルーティングオプションを表示する

名前をクリックして、ヘルスチェック、ビルト設定、Deployment、スケーリング、リソース制限、and ラベル の詳細オプションにアクセスします。

作成 キャンセル

一般

「作成」クリックにより、
アプリ作成と公開を自動実行

コンテナイメージビルト用の
コンテナの選択

ソースコード置き場となる
GitリポジトリのURL

アプリ公開用URLの作成と公開の
自動実行を指定するオプション

Project: test-project01 アプリケーション: すべてのアプリケーション

ビルドイメージのバージョン

IST 3.9-ubi8

IST 3.9-ubi8

IST 3.9-ubi9

IST 3.8-ubi7

IST 3.8-ubi8

IST 3.6-ubi8

IST 2.7-ubi7

IST 2.7-ubi8

IST latest

サンプルを試す ↗

詳細の Git オプションの表示

Copyright 2023 Red Hat K.K.

20

OpenShift上のアプリケーショントポロジー (ユーザーは、作成したアプリケーションの様々な情報を確認可能)

The image displays three side-by-side screenshots of the OpenShift web interface, illustrating the application topology and monitoring features.

Screenshot 1: Application Details (Left)

This screenshot shows the detailed view for the application "django-ex". It includes:

- Pod Summary:** Shows 1 Pod.
- General Information:** Name: django-ex, Namespace: test-project01.
- Labels:** app=django-ex, app.kubernetes.io/...=django-ex, etc.
- Pod Selector:** app=django-ex.
- Node Selector:** セレクターなし (No selector).

Screenshot 2: Resource Monitoring (Middle)

This screenshot shows the resource monitoring section for the "django-ex" application. It includes:

- Pod:** django-ex-85fcccc7c4-wtr5f, Status: Running.
- Build:** Build #1 completed 20 minutes ago.
- Service:** django-ex, Service port: 8080-tcp → Pod port: 8080.
- Route:** 自動作成されたアプリ公開用 URL (Route53に自動登録されたドメイン名を利用)
場所: https://django-ex-test-project01.apps.rosa.hcp-cluster01.6tzy.p3.openshiftapps.com

A yellow callout box highlights the automatically generated public URL for the application.

Screenshot 3: Metrics (Right)

This screenshot shows the metrics monitoring section for the "django-ex" application. It includes:

- Metrics:** CPU 使用量 (CPU Usage) chart from 9:30 to 9:45. The chart shows a significant spike in usage around 9:35.
- Memory Usage:** メモリー使用量 (Memory Usage) chart showing usage between 100M and 200M.

EKS/ROSA 利用時の月額(730時間)料金イメージ その1

※ ROSAのHosted Control Plane利用を想定。2024年4月時点では、東京と大阪リージョンで利用できます。

- AWSの東京リージョンのMulti-AZ構成を想定 (ワーカーノードが3台構成)
- EKSではManaged Node Group (EC2インスタンス)の利用を想定
- 計算の簡単化のために、EKS/ROSA共に利用が想定される NAT Gateway, AWS Load Balancer, Route53のサービス利用料金や、AWS内外でのデータ転送利用料金などを除外
- AWS CodeCommit は無料利用枠の利用を想定

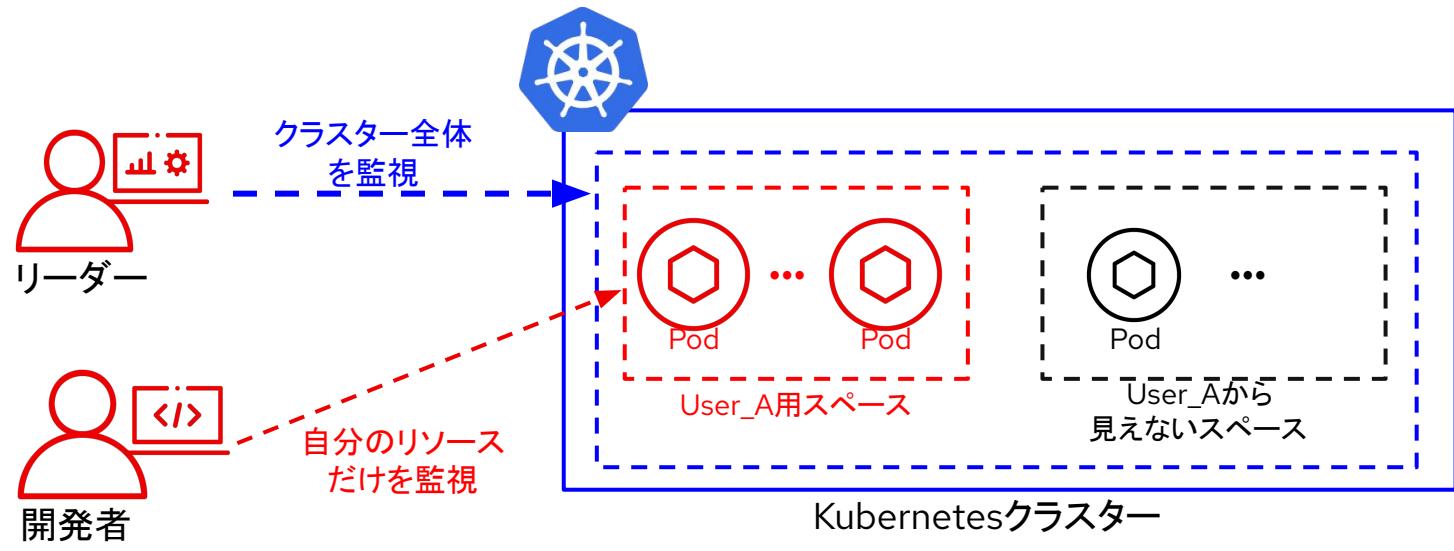
AWSサービス	利用料金 (USD)
EKS (0.10USD/hour/cluster)	73 (0.10 x 730)
EC2 インスタンス (Worker) (m5.xlarge x3, 0.248USD/hour EBS(gp3): 300GB, 0.08USD/GB/month)	615.12 (0.248 x 3 x 730 + 300 x 3 x 0.08)
Cloud9用 EC2インスタンス (共有環境 m5.large(2vCPU, メモリ8GB) x3, EBS(gp3): 100GB)	295.56 (0.124 x 3 x 730 + 100 x 3 x 0.08)
CodeBuild (16時間/monthのビルドを想定 ビルド用インスタンス general.medium (4vCPU, 7GB), 0.60USD/hour)	9.6 (0.6 x 16)
ECR (計100GBのイメージ保存を想定 保存料金: 0.10USD/GB/month)	10 (0.1 x 100)
EKS利用時の想定合計金額(USD): 1003.28	

AWSサービス	利用料金 (USD)
ROSA サービス料金 (Cluster fee: 0.25USD/hour/cluster Worker Node service fee: 0.171USD/4vCPU/hour)	557 (0.25 x 730 + 0.171 x 3 x 730)
EC2 インスタンス (Worker) (m5.xlarge x3, 0.248USD/hour EBS(gp3): 300GB, 0.08USD/GB/month)	615.12 (0.248 x 3 x 730 + 300 x 3 x 0.08)
S3 (Internal Registryのバックエンド) (計100GBのイメージ保存を想定 保存料金: 0.025USD/GB/month)	2.5 (0.025 x 100)
ROSA利用時の想定合計金額(USD): 1174.62	

EKS/ROSA サンプルユースケース その2

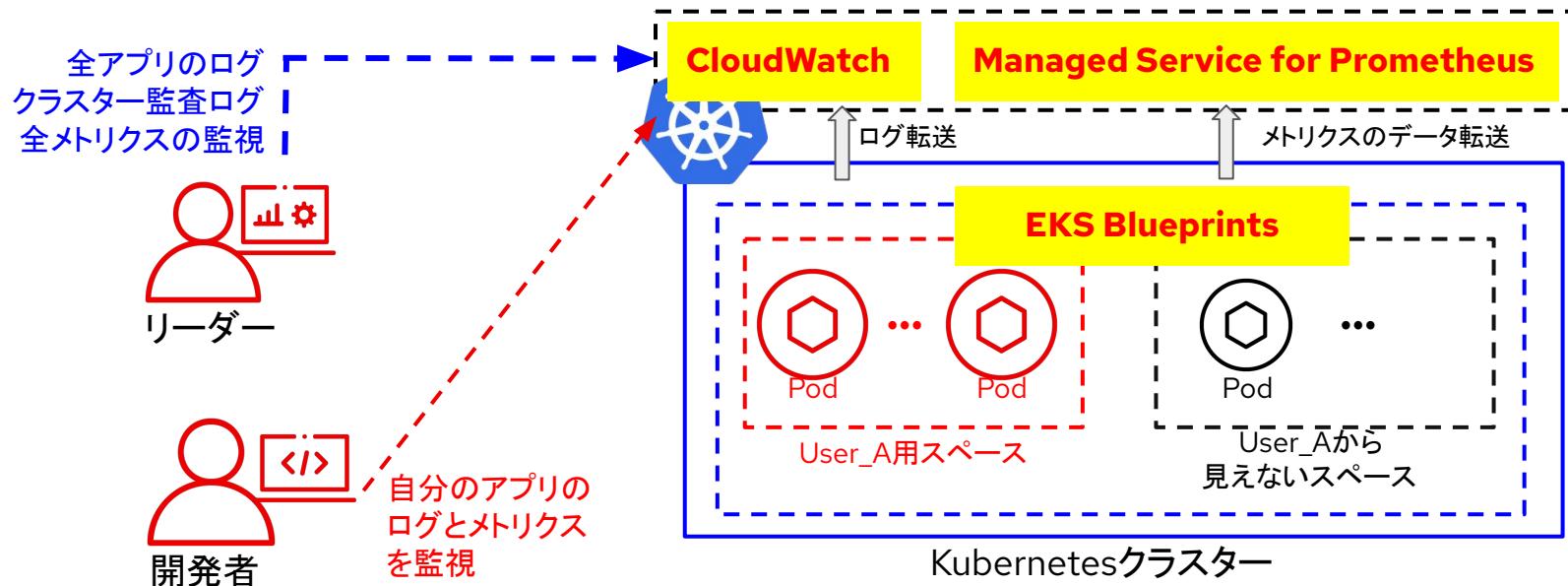
EKS/ROSA のサンプルユースケース その2

- Kubernetesクラスターのロギングとモニタリングを設定 /利用
- 開発部門のリーダーは、クラスター全体ログの集約や、アプリを実行するワーカーノードを中心としたクラスター全体のリソース (CPU/メモリなど) 利用率を監視
- 開発者は、自分のスペースにあるアプリログやリソース利用率を監視



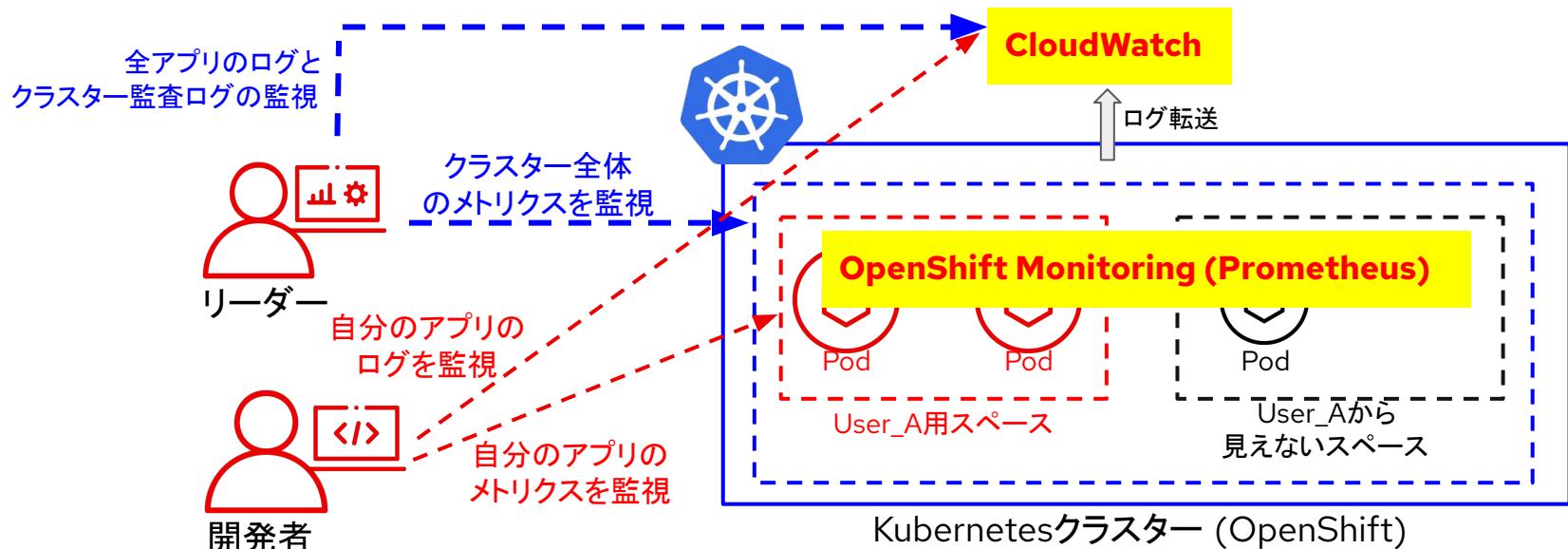
EKSを使う場合のイメージ

- ロギングには、Amazon CloudWatchを利用する (AWS CLIによるログのダウンロードも可能)
- モニタリングには、Amazon Managed Service for Prometheus (AMP) を利用
- CloudWatch/AMPを利用するための、AWS IAM設定/ポリシーによるfiltrationや、データ転送のための Kubernetesクラスター内の権限設定が別途必要



ROSAを使う場合のイメージ

- ロギングには、Amazon CloudWatch を利用※
- モニタリングには、ROSAのデフォルトで有効化されているPrometheusを利用
- Amazon CloudWatchを利用するためのAWS IAM設定と、IAMポリシーによるフィルタリングが別途必要
 - OpenShift Logging Operatorによる、AWS IAM 認証情報を利用したログ転送を設定
- OpenShiftクラスターと連携した認証プロバイダーのユーザー アカウントが、管理するリソース利用率だけが見えるような設定が、デフォルトで適用済み



※ OpenShift Logging (Loki) も利用できますが、Lokiのサポートに必要な要件が、vCPU:36以上/メモリ:63GB以上、と大きなサイズとなりますので、ご注意ください。

EKS/ROSAでのCloudWatchによるログ集約のイメージ

EKS/ROSA 共通: ログタイプごとのロググループ作成(アプリケーション/監査/インフラストラクチャー)
ROSAの場合、Red Hatのサポートケース経由で、監査ログのリクエストも可能です。

The screenshot shows the AWS CloudWatch Log Groups interface. On the left, a sidebar lists log groups under 'ロググループ (4)'. The 'rosa-sample-cluster01.test-project01' group is selected and highlighted with a dashed border. The main pane displays the details for this group, including 'データ保護' (Data Protection), '機密データの数' (Number of sensitive data), '保持' (Retention), 'メトリクスフィルター' (Metrics Filter), and '寄稿者のインサート' (Authorizer's Insert). A yellow callout box points to the 'ロググループを作成' (Create Log Group) button at the top right of the main pane.

ROSAのOpenShift Logging Operatorによるログ転送では、ユーザーのスペース(この例では「test-project01」を指定)ごとにロググループを作成するような設定が可能
(EKSでもFluent-Bitによるログ転送時のフィルタリングルールを利用した、同等の設定が可能)

CloudWatch > ロググループ > rosa-sample-cluster01.test-project01 > kubernetes.var.log.pods.test-project01_django-psql-persistent-1-rgzqj_7c44cd17-5799-47cc-83db-3a9cc9f70d28.djangoproject-psql-persistent.0.log

ログイベント

メッセージ

ユーザーアプリのログを参照可能

Copyright 2023 Red Hat K.K.

OpenShiftのコンソールでのログ確認

Project: test-project01

Pod > Pod の詳細

P django-ex-85fcccc7c4-wtr5f Running

詳細 Metrics YAML 環境 ログ イベント ターミナル

ログのストリーミング中です... C django-ex 現在のログ Search 行の折り返し | Raw | ダウンロード | 拡張

36 行

```
Operations to perform:
2 Apply all migrations: admin, auth, contenttypes, sessions, welcome
3 Running migrations:
4 Applying contenttypes.0001_initial... OK
5 Applying auth.0001_initial... OK
6 Applying admin.0001_initial... OK
7 Applying admin.0002_remove_auto_add... OK
8 Applying contenttypes.0002_remove_content_type_name... OK
9 Applying auth.0002_alter_permission_name_max_length... OK
10 Applying auth.0003_alter_user_email_max_length... OK
11 Applying auth.0004_alter_user_username_opts... OK
12 Applying auth.0005_alter_user_last_login_null... OK
13 Applying auth.0006_require_contenttypes_0002... OK
14 Applying auth.0007_alter_validators add error messages... OK
15 Applying auth.0008_alter_user_username_max_length... OK
16 Applying sessions.0001_initial... OK
17 Applying welcome.0001_initial... OK
18 --> Serving application with gunicorn (wsgi) with default settings
[2023-07-07 00:26:58 +0000] [1] [INFO] Starting gunicorn 19.5.0
[2023-07-07 00:26:58 +0000] [1] [INFO] Listening at: http://0.0.0.0
[2023-07-07 00:26:58 +0000] [1] [INFO] Using worker: sync
[usr@lib64/python3.9/os.py:1023: RuntimeWarning: line buffering (bu
return io.open(fd, *args, **kwargs)
[2023-07-07 00:26:58 +0000] [21] [INFO] Booting worker with pid: 21
[2023-07-07 00:26:58 +0000] [22] [INFO] Booting worker with pid: 22
[2023-07-07 00:26:58 +0000] [23] [INFO] Booting worker with pid: 23
[2023-07-07 00:26:58 +0000] [24] [INFO] Booting worker with pid: 24
[2023-07-07 00:26:58 +0000] [25] [INFO] Booting worker with pid: 25
[2023-07-07 00:26:59 +0000] [26] [INFO] Booting worker with pid: 26
[2023-07-07 00:26:59 +0000] [27] [INFO] Booting worker with pid: 27
[2023-07-07 00:26:59 +0000] [28] [INFO] Booting worker with pid: 28
```

Podのログ

アクション

ログのストリーミング中です... C django-ex 現在のログ Search 行の折り返し | Raw | ダウンロード | 拡張

36 行

```
Operations to perform:
2 Apply all migrations: admin, auth, contenttypes, sessions, welcome
3 Running migrations:
4 Applying contenttypes.0001_initial... OK
5 Applying auth.0001_initial... OK
6 Applying admin.0001_initial... OK
7 Applying admin.0002_remove_auto_add... OK
8 Applying contenttypes.0002_remove_content_type_name... OK
9 Applying auth.0002_alter_permission_name_max_length... OK
10 Applying auth.0003_alter_user_email_max_length... OK
11 Applying auth.0004_alter_user_username_opts... OK
12 Applying auth.0005_alter_user_last_login_null... OK
13 Applying auth.0006_require_contenttypes_0002... OK
14 Applying auth.0007_alter_validators add error messages... OK
15 Applying auth.0008_alter_user_username_max_length... OK
16 Applying sessions.0001_initial... OK
17 Applying welcome.0001_initial... OK
18 --> Serving application with gunicorn (wsgi) with default settings
[2023-07-07 00:26:58 +0000] [1] [INFO] Starting gunicorn 19.5.0
[2023-07-07 00:26:58 +0000] [1] [INFO] Listening at: http://0.0.0.0
[2023-07-07 00:26:58 +0000] [1] [INFO] Using worker: sync
[usr@lib64/python3.9/os.py:1023: RuntimeWarning: line buffering (bu
return io.open(fd, *args, **kwargs)
[2023-07-07 00:26:58 +0000] [21] [INFO] Booting worker with pid: 21
[2023-07-07 00:26:58 +0000] [22] [INFO] Booting worker with pid: 22
[2023-07-07 00:26:58 +0000] [23] [INFO] Booting worker with pid: 23
[2023-07-07 00:26:58 +0000] [24] [INFO] Booting worker with pid: 24
[2023-07-07 00:26:58 +0000] [25] [INFO] Booting worker with pid: 25
[2023-07-07 00:26:59 +0000] [26] [INFO] Booting worker with pid: 26
[2023-07-07 00:26:59 +0000] [27] [INFO] Booting worker with pid: 27
[2023-07-07 00:26:59 +0000] [28] [INFO] Booting worker with pid: 28
```

Node > Node の詳細

N ip-10-0-1-62.us-east-2.compute.internal Ready

概要 詳細 YAML Pod ログ イベント ターミナル

ノードのログ
(RHELのjournaldによるログを表示)

The log is abridged due to length.

To view unabridged log content, open the raw file in another window.

journal Filter by unit

検索

アクション

行の折り返し

```
1 Jul 07 00:42:49.534725 ip-10-0-1-62 systemd[1]: run-runc-fb5f5ae5b28c0d5499400d19d2b83275f7f64c2fc52299c3e0f8f44e21cf0-runc.S9VHzn.mount: Succeeded.
2 Jul 07 00:42:56.832053 ip-10-0-1-62 ovs-vswitchd[1120]: ovs[57729] [connmgr] INFO |br-<->>unix#366758: 2 flow_mods in the last 0 s (2 adds)
3 Jul 07 00:43:11.830292 ip-10-0-1-62 ovs-vswitchd[1120]: ovs[57730] [connmgr] INFO |br-<->>unix#366767: 2 flow_mods in the last 0 s (2 adds)
4 Jul 07 00:43:25.947747 ip-10-0-1-62 kubenswrapper[2115]: I0707 00:43:25.947639 2115 kubelet_getters.go:182] "Pod status updated" pod="kube-system/kube-apiserver-proxy-ip-10-0-1-62.us-east-2.com"
5 Jul 07 00:43:26.833547 ip-10-0-1-62 ovs-vswitchd[1120]: ovs[57731] [connmgr] INFO |br-<->>unix#366771: 2 flow_mods in the last 0 s (2 adds)
6 Jul 07 00:43:41.828364 ip-10-0-1-62 ovs-vswitchd[1120]: ovs[57732] [connmgr] INFO |br-<->>unix#366788: 2 flow_mods in the last 0 s (2 adds)
7 Jul 07 00:43:56.829261 ip-10-0-1-62 ovs-vswitchd[1120]: ovs[57733] [connmgr] INFO |br-<->>unix#366784: 2 flow_mods in the last 0 s (2 adds)
8 Jul 07 00:44:11.828987 ip-10-0-1-62 ovs-vswitchd[1120]: ovs[57734] [connmgr] INFO |br-<->>unix#366793: 2 flow_mods in the last 0 s (2 adds)
9 Jul 07 00:44:25.947820 ip-10-0-1-62 kubenswrapper[2115]: I0707 00:44:25.947795 2115 kubelet_getters.go:182] "Pod status updated" pod="kube-system/kube-apiserver-proxy-ip-10-0-1-62.us-east-2.com"
10 Jul 07 00:44:26.828184 ip-10-0-1-62 ovs-vswitchd[1120]: ovs[57735] [connmgr] INFO |br-<->>unix#366797: 2 flow_mods in the last 0 s (2 adds)
11 Jul 07 00:44:41.829108 ip-10-0-1-62 ovs-vswitchd[1120]: ovs[57736] [connmgr] INFO |br-<->>unix#366806: 2 flow_mods in the last 0 s (2 adds)
12 Jul 07 00:44:56.830494 ip-10-0-1-62 ovs-vswitchd[1120]: ovs[57737] [connmgr] INFO |br-<->>unix#366810: 2 flow_mods in the last 0 s (2 adds)
13 Jul 07 00:45:11.827005 ip-10-0-1-62 ovs-vswitchd[1120]: ovs[57738] [connmgr] INFO |br-<->>unix#366819: 2 flow_mods in the last 0 s (2 adds)
14 Jul 07 00:45:18.544443 ip-10-0-1-62 systemd[1]: run-runc-53bb6238bf864f577a4b27799e87037d08ad4cad00e599b05cc29c4419a4fc5-runc.xIRGLz.mount: Succeeded.
15 Jul 07 00:45:25.948013 ip-10-0-1-62 kubenswrapper[2115]: I0707 00:45:25.947985 2115 kubelet_getters.go:182] "Pod status updated" pod="kube-system/kube-apiserver-proxy-ip-10-0-1-62.us-east-2.com"
16 Jul 07 00:45:26.828281 ip-10-0-1-62 ovs-vswitchd[1120]: ovs[57739] [connmgr] INFO |br-<->>unix#366823: 2 flow_mods in the last 0 s (2 adds)
17 Jul 07 00:45:31.248004 ip-10-0-1-62 crio[2088]: time=2023-07-07 00:45:31.2398631047" level=info msg="Checking image status: quay.io/openshift/release-dev/ocp-v4.0-art-dev@sha256:e63c4cff31ad
18 Jul 07 00:45:31.247277 ip-10-0-1-62 crio[2088]: time=2023-07-07 00:45:31.2401168522" level=info msg="Image status: &ImageStatusResponse{Image:&Image{Id:e0da58ff44217a0fc0857f38ec7db81d41286
19 Jul 07 00:45:41.826000 ip-10-0-1-62 ovs-vswitchd[1120]: ovs[57740] [connmgr] INFO |br-<->>unix#366832: 2 flow_mods in the last 0 s (2 adds)
20 Jul 07 00:45:56.829795 ip-10-0-1-62 ovs-vswitchd[1120]: ovs[57741] [connmgr] INFO |br-<->>unix#366836: 2 flow_mods in the last 0 s (2 adds)
21 Jul 07 00:46:11.829947 ip-10-0-1-62 ovs-vswitchd[1120]: ovs[57742] [connmgr] INFO |br-<->>unix#366846: 2 flow_mods in the last 0 s (2 adds)
22 Jul 07 00:46:25.949016 ip-10-0-1-62 kubenswrapper[2115]: I0707 00:46:25.948909 2115 kubelet_getters.go:182] "Pod status updated" pod="kube-system/kube-apiserver-proxy-ip-10-0-1-62.us-east-2.com"
23 Jul 07 00:46:26.834576 ip-10-0-1-62 ovs-vswitchd[1120]: ovs[57743] [connmgr] INFO |br-<->>unix#366850: 2 flow_mods in the last 0 s (2 adds)
24 Jul 07 00:46:38.546894 ip-10-0-1-62 systemd[1]: run-runc-53bb6238bf864f577a4b27799e87037d08ad4cad00e599b05cc29c4419a4fc5-runc.vZ4H05.mount: Succeeded.
25 Jul 07 00:46:41.832282 ip-10-0-1-62 ovs-vswitchd[1120]: ovs[57744] [connmgr] INFO |br-<->>unix#366859: 2 flow_mods in the last 0 s (2 adds)
```

OpenShiftのモニタリング (Prometheus)



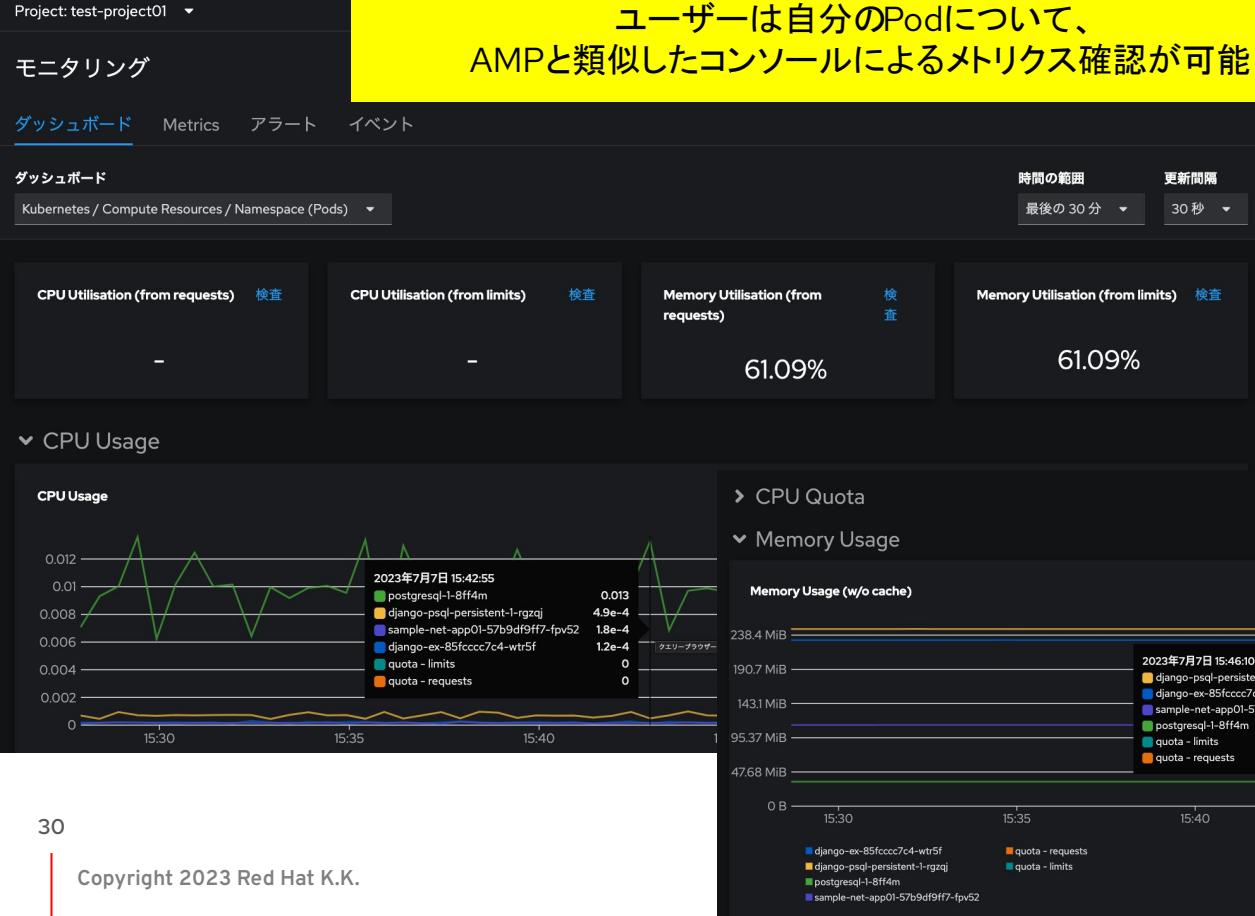
クラスター全体や各ノード/Podのリソース利用情報

名前	ステータス	Role	Pod	メモリー	CPU	ファイルシ...	作成
ip-10-0-1-62.us-east-2.compute.internal	Ready	worker	41	6.9 GiB / 15.35 GiB	0.387 コア / 4 コア	21.58 GiB / 299.8 GiB	2023年6月27日 16:30
ip-10-0-1-169.us-east-2.compute.internal	Ready	worker	42	5.54 GiB / 15.35 GiB	0.380 コア / 4 コア	26.98 GiB / 299.8 GiB	2023年6月27日 16:29

OpenShiftのモニタリング (Prometheus)

※ ユーザーは自分のアプリに関する情報しか見えない状態

ユーザーは自分のPodについて、
AMPと類似したコンソールによるメトリクス確認が可能



EKS/ROSA 利用時の月額(730時間)料金イメージ その2

※ ROSAのHosted Control Plane利用を想定。2024年4月時点では、東京と大阪リージョンで利用できます。

- 基本的には、「サンプルユースケース その1」と同じ利用条件と計算過程を想定
- EKS/ROSA以外に利用するAWSサービスは、CloudWatchとManaged Service for Prometheus (AMP)を想定
- CloudWatchでは、ログの収集と保存のみを利用すると想定
- AMPでは、下記の「Example 1 - EKS on EC2 and Kubernetes」のシナリオを想定
 - <https://aws.amazon.com/jp/prometheus/pricing/>

AWSサービス	利用料金 (USD)
EKS (0.10USD/hour/cluster)	73 (0.10 x 730)
EC2 インスタンス (Worker) (m5.xlarge x3, 0.248USD/hour EBS(gp3): 300GB, 0.08USD/GB/month)	615.12 (0.248 x 3 x 730 + 300 x 3 x 0.08)
CloudWatch (10GB/month のログサイズを想定 ログ収集: 0.76USD/GB/month ログ保存: 0.033USD/GB/month)	7.933 (0.76 x 10 + 0.033 x 10)
AMP (メトリクスのストレージは、3.34GB)	81.75
EKS利用時の想定合計金額(USD): 777.803	

AWSサービス	利用料金 (USD)
ROSA サービス料金 (Cluster fee: 0.25USD/hour/cluster Worker Node service fee: 0.171USD/4vCPU/hour)	557 (0.25 x 730 + 0.171 x 3 x 730)
EC2 インスタンス (Worker) (m5.xlarge x3, 0.248USD/hour EBS(gp3): 300GB, 0.08USD/GB/month)	615.12 (0.248 x 3 x 730 + 300 x 3 x 0.08)
CloudWatch (10GB/month のログサイズを想定 ログ収集: 0.76USD/GB/month ログ保存: 0.033USD/GB/month)	7.933 (0.76 x 10 + 0.033 x 10)
ROSA利用時の想定合計金額(USD): 1,180.053	

ここまでまとめ: ECS/EKS/ROSAのどれを使うべきか??

- アプリ開発や実行のために、様々なAWSネイティブのサービスを活用したい
- Kubernetes (K8s) や関連するCNCFプロジェクトは使わない

ECS

- 最小限のインフラコストでKubernetes環境を利用したい
- K8sのセキュリティ設定に関するノウハウがある
- 選択したOSS、K8sアドオンやAWSサービスを合わせた運用体制を構築済み
 - 各サービスを組み合わせて商用レベルの開発・運用環境を構築可能
 - AWS IAMの設定やクラスター内の権限設定を実現可能
 - AWSによるIAM標準ポリシーのアップデートにも対応可能な体制がある
- コンテナを実行するためのミドルウェアのサポートが不要
 - 脆弱性対策やトラブルシューティングなどを自力で実施可能

EKS

- K8sレイヤやコンテナ開発/運用に必要なコンポーネントがパッケージングされた製品を活用したい
 - 基盤構築に必要な時間を短縮したい
 - セキュリティが強化されたK8sを利用したい
 - AWS IAMの設定やクラスター内の権限設定の手間を、最小限に抑えたい
 - 他サービスと統合されたコンソールを利用したい
- Red HatによるコンテナのOS(RHEL)やミドルウェアのサポートが必要
 - 脆弱性対策やトラブルシューティングなどの支援を依頼したい

ROSA

ROSA Hosted Control Plane サービス仕様

ROSAの契約形態

ROSAは、AWSの請求書にまとめたいお客様を想定した、従量課金のサービス

ROSA	
処理主体 (Transacted)	AWS
請求元 (Billed by) ※1	AWS
契約条件 (Contract terms)	AWS
マネージド管理主体 (Managed by) ※2	AWS + Red Hat
サポート作業主体 (Supported by) ※3	AWS + Red Hat

※1: 請求元が発行する請求書には、OpenShiftサービスの利用料金や、仮想マシンやストレージなどのクラウドサービスの利用料金が記載されます。

※2: ロギング、モニタリング、プラットフォームのアップグレード、セキュリティ、などを担当する主体者。

※3: インストール、利用方法、設定、問題診断、バグ解決などに関して、チャット、電話、メールなどでの問い合わせ対応を実施する主体者。

ROSA overview of responsibility assignment matrix

- AWSがIaaS基盤、Red HatがOpenShiftを管理
- お客様は、アプリ、アプリの{データ、ロギング、ネットワークポリシー}、VPN/VPC接続、クラスターのプライベートネットワークなどを管理
- 詳細: https://docs.openshift.com/roса/roса_architecture/roса_policy_service_definition/roса-policy-responsibility-matrix.html

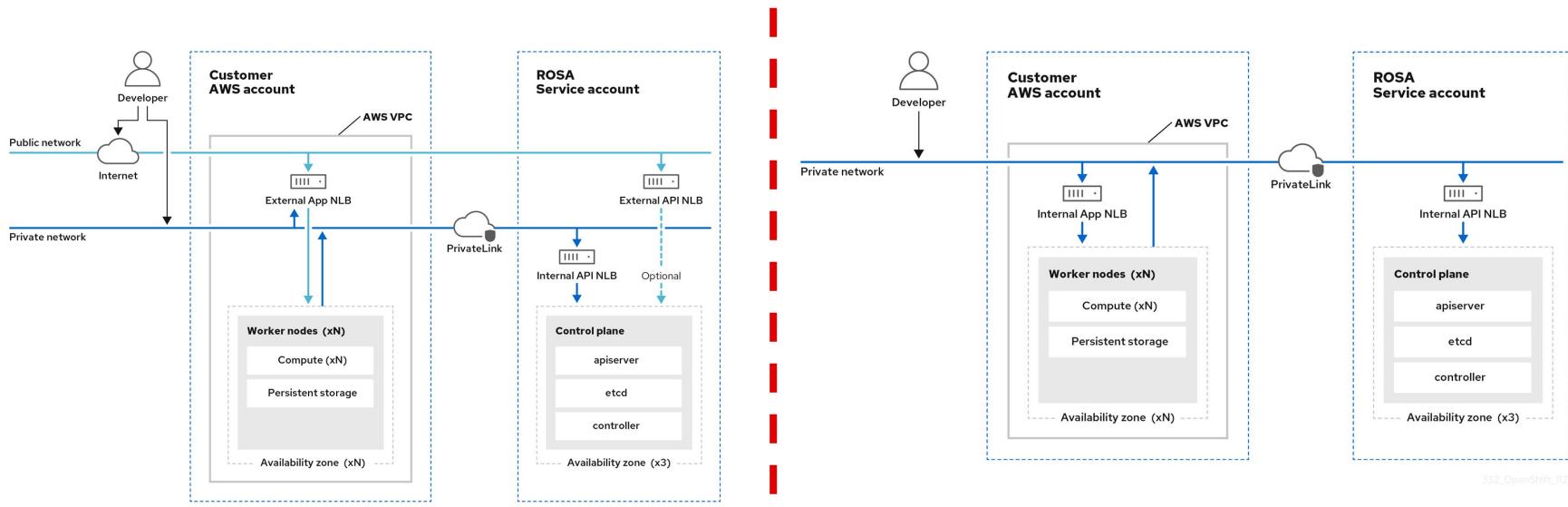
Resource	Incident and operations management	Change management	Identity and access management	Security and regulation compliance	Disaster recovery					
Customer Data	Customer									
Customer applications	Customer									
Developer services	Customer									
Platform monitoring	Red Hat									
Logging	Red Hat	Shared			Red Hat					
Application networking	Shared			Red Hat						
Cluster networking	Red Hat	Shared		Red Hat						
Virtual networking	Shared									
Master and infrastructure nodes	Red Hat									
Worker nodes	Red Hat									
Cluster version	Red Hat	Shared	Red Hat							
Capacity management	Red Hat	Shared	Red Hat							
Virtual storage	Red Hat									
Physical infrastructure and security	Red Hat									

ROSA Hosted Control Plane (HCP) の主なサービス仕様

Specification	ROSA
デフォルトの アーキテクチャ (SingleAZ/MultiAZ共通)	Worker Node: m5.xlarge (4vCPU/RAM 16GiB) x2 (最小は2台)
Max Worker Nodes	51
対応リージョン	us-east-1, us-east-2, us-west-2, eu-west-1, eu-central-1, ap-southeast-3 (2023年12月時点で、東京/大阪リージョンでは利用不可)
プライベートクラスター	対応 (ROSA) AWS VPC peering, AWS VPN, AWS Direct Connect が利用可能
IDプロバイダー (認証)	GitHub, GitHub Enterprise, GitLab, Google, LDAP, OpenID Connect, HTPasswd (HTPasswdはROSAクラスター管理者用途としてのみサポート)
AWSサービスとの連携	Amazon EBS/EFS(ストレージ機能), CloudWatch (ログ転送) AWS Controllers for Kubernetes (ACK) によるAWS SDKを利用したサービスデプロイが可能
アップデート作業	Red HatによるOpenShiftの自動アップデート (お客様による、アップデート時間の設定が可能)
SLA	99.95%

ROSA HCPクラスターのアーキテクチャ

- EKSと同様に、ユーザー アカウントにはワーカーノードしか見えない状態のクラスター
- ROSA Service Accountは、Red Hat SREチームが管理
- AWS ファイアウォールによる送信トラフィックの制御 が可能



AWSサービスとの連携

ROSAクラスターでのAmazon EBSの利用

- ROSAクラスターでは、EBSを利用するためのストレージクラスが予め設定されており、追加設定することなく、Pod用の永続ボリュームとして利用可能

The screenshot shows two views of the Red Hat OpenShift Service on AWS web interface.

Storage Classes View: The top part shows a list of storage classes under the "Storage" menu. A red box highlights the "gp3 - デフォルト" entry, which is selected. The table columns are: Name, Provisioner, and Recycle Policy. The "gp3 - デフォルト" row has "ebs.csi.aws.com" listed under both Provisioner and Recycle Policy.

名前	プロビジョナー	回収ポリシー
gp2	kubernetes.io/aws-ebs	Delete
gp2-csi	ebs.csi.aws.com	Delete
gp3 - デフォルト	ebs.csi.aws.com	Delete
gp3-csi	ebs.csi.aws.com	Delete

Persistent Volume Claims View: The bottom part shows a list of PVCs under the "Storage" menu. A red box highlights the "pvc-63062e46-d76a-45bb-8f12-828ee7faf0a" entry, which is bound to a volume. The table columns are: Name, Status, Persistent Volume, Capacity, Usage, and Storage Class. The "pvc-63062e46-d76a-45bb-8f12-828ee7faf0a" row has "gp3" listed under Storage Class.

名前	ステータス	永続ポリューム	容量	使用済み	ストレージクラス
pvc-63062e46-d76a-45bb-8f12-828ee7faf0a	Bound	pvc-63062e46-d76a-45bb-8f12-828ee7faf0a	1GiB	50.33 MiB	gp3

ROSAクラスターのAmazon EFSの利用

- ROSAクラスターの複数のワーカーノードから同時に読み書き可能な永続ボリュームとして、EFSを利用するように設定可能
- EFSを利用するための、ファイルシステムとアクセスポイントを事前作成
- ROSAクラスターにある「AWS EFS CSI Driver Operator」を利用して、EFSを利用するためのストレージドライバーをインストール
- インストールされたEFS用のストレージドライバーを利用して、永続ボリュームリクエスト (PVC)を作成し、Podが利用するイメージ (NFS v4プロトコルによるアクセス)

The screenshot shows the Red Hat OpenShift Service on AWS interface. The left sidebar has a 'Operator' dropdown set to 'OperatorHub', which is highlighted. The main area displays the 'OperatorHub' page with a search bar containing 'EFS CSI'. A single result is shown: 'AWS EFS CSI Driver Operator' by Red Hat, described as 'Install and configure AWS EFS CSI driver.'

ROSAクラスターのロギング

- Amazon CloudWatchへの、下記3種類のログ転送が可能
 - ユーザーアプリケーションのログ
 - インフラストラクチャー関連のログ
 - openshift-*, kube-*などのプロジェクトにあるログ
 - セキュリティ監査に関するログ(audit.log)
 - 監査ログはRed HatのSREチームによって、別途収集され、ユーザー リクエストに応じて随時提供するため、必ずしも収集する必要はありません。
- ログ転送の設定方法の概要※
 - CloudWatchを利用する専用のAWS IAMユーザーを作成
 - ROSAクラスターでOpenShift Logging Operatorをインストールして、 vectorによるロギングとCloudWatchへのログ転送用のインスタンスを作成
 - ログ転送に、上記IAMユーザーの認証情報を利用

※ STSを利用した設定方法もありますが、ここでは割愛します。

ROSAクラスターのロギング

The screenshot shows the Red Hat OpenShift Service on AWS console. The left sidebar navigation includes: Administrator, ホーム, Operator, OperatorHub (selected), インストール済みの Operator, ワークロード, ネットワーク, ストレージ, ビルド, 監視, コンピュート, ユーザー管理, 管理. The main content area shows the "openshift-logging" project details for the "Red Hat OpenShift Logging" operator version 5.5.2. It lists "提供される API": Cluster Log Forwarder (description: "ClusterLogForwarder is an API to configure forwarding logs. You configure forwarding by specifying a list of pipelines, which forward from a set of named inputs to a set of named outputs. There are built-in input names for common log categories, and...") and Cluster Logging (description: "A Red Hat OpenShift Logging instance. ClusterLogging is the Schema for the clusterloggings API"). Buttons for "インスタンスの作成" are shown for both. Below this, there's a "説明" section for "Red Hat OpenShift Logging" (description: "The Red Hat OpenShift Logging Operator orchestrates and manages the aggregated logging stack as a cluster-wide service.") and "Features" (Create/Destroy, Simplified Configuration). The "Prerequisites and Requirements" section notes that the Red Hat OpenShift Logging Namespace must be explicitly created by a cluster administrator. At the bottom, it links to "Deploying cluster logging in the OpenShift product documentation".

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
  name: instance
  namespace: openshift-logging
spec:
  collection:
    type: vector
managementState: Managed
```

```
apiVersion: "logging.openshift.io/v1"
kind: ClusterLogForwarder
metadata:
  name: instance
  namespace: openshift-logging
spec:
  outputs:
    - name: cw
      type: cloudwatch
      cloudwatch:
        region: ap-northeast-1
      secret:
        name: cw-secret
  pipelines:
    - name: all-logs
      inputRefs:
        - application
        - infrastructure
        - audit
      outputRefs:
        - CW
```

転送先のCloudWatchと
リージョン/認証情報を指定

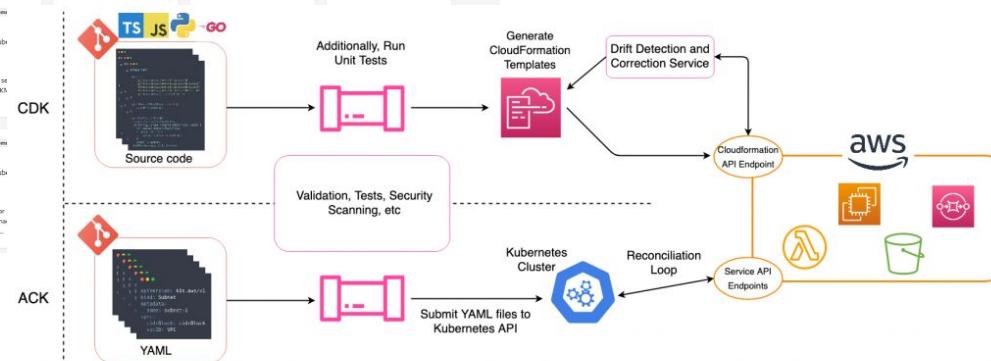
転送ログの種類を指定

ACKによるAWSサービスのデプロイ

- さまざまなAWSサービスの作成/削除が可能

The screenshot shows the Red Hat OpenShift Service on AWS OperatorHub interface. On the left, there's a sidebar with navigation links for Administrator, Operator, OperatorHub, and various project categories like Home, Operator, Workload, Storage, Build, Monitor, Compute, User Management, and Management. The main area displays a search results page for 'aws'. It lists several AWS Controller Operators, each with a brief description and the provider (Red Hat or Community). The results include:

- AWS Controllers for Kubernetes - Amazon API Gateway v2 controller
- AWS Application Auto Scaling
- AWS IAM controller for managing API Gateway v2 resources
- AWS Controllers for Kubernetes - Amazon ElastiCache
- AWS IAM controller for managing ElastiCache resources
- AWS Controllers for Kubernetes - Amazon RDS
- AWS S3 controller for managing RDS resources in Kubernetes
- AWS Controllers for Kubernetes - Amazon DynamoDB
- AWS ECR controller for managing EC2 resources in Kubernetes
- AWS Controllers for Kubernetes - Amazon ECR
- AWS EKS controller for managing EKS resources in Kubernetes

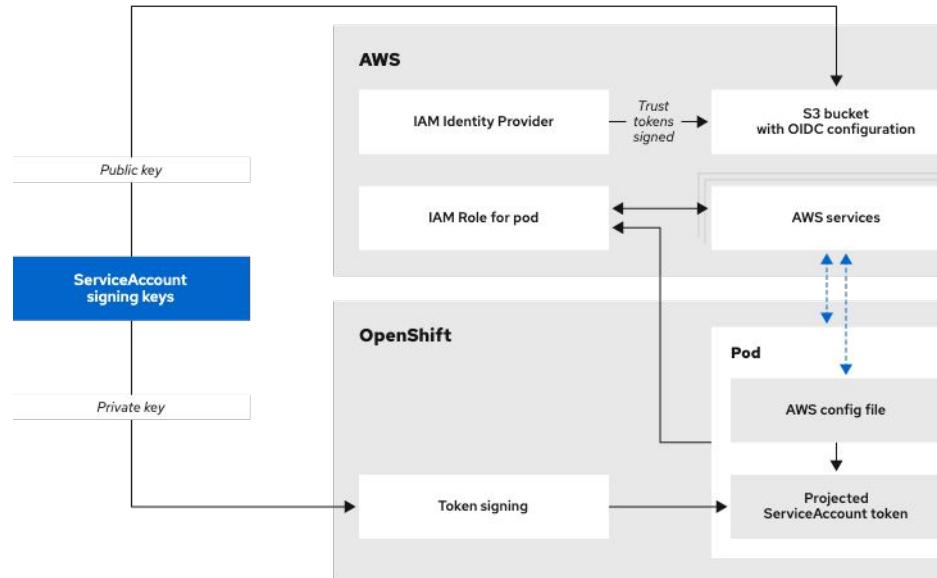


引用元: [Comparing AWS Cloud Development Kit and AWS Controllers for Kubernetes](#)

ROSA上のアプリケーションとAWSサービスとの連携

<https://aws.amazon.com/jp/blogs/containers/fine-grained-iam-roles-for-red-hat-openshift-service-on-aws-rosa-workloads-with-sts/>

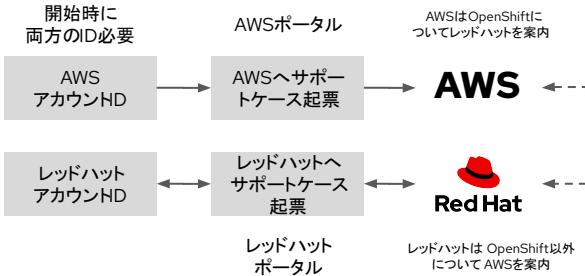
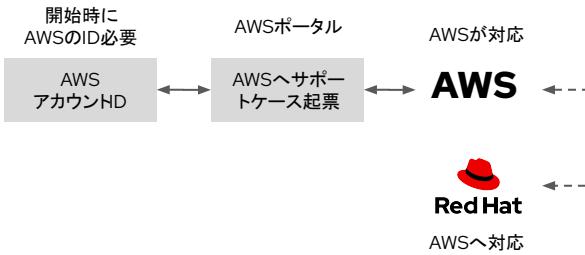
- AWS STSを使った、一時的な認証設定が可能
- OpenShiftの各Podは、サービスアカウントのトークン(システムコンポーネント間で利用される認証情報)により、AWSサービスを呼び出すことが可能
 - 参考: [サービスアカウントのIAMロール](#) (AWS公式ドキュメント)



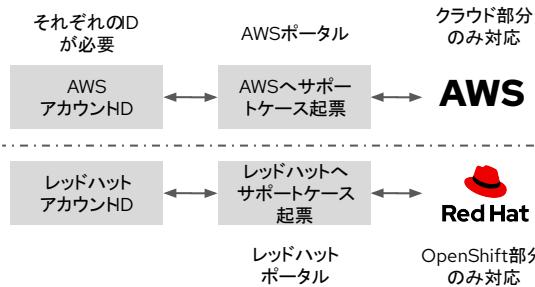
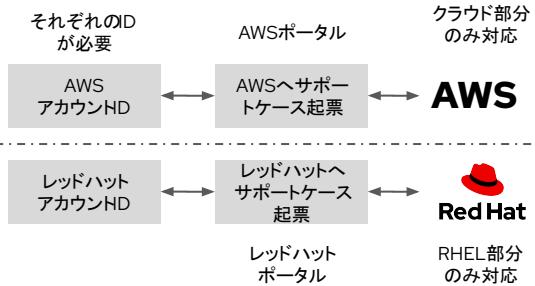
サポート体制

PAYGとBYOSのサポート体制

PAYG



BYOS



Appendix: AWSに持ち込むRed Hat製品 Bring your Own Subscription (BYOS)

RHEL/OpenShiftのサブスクリプション情報

- RHELのサブスクリプションガイド
 - AWSに持ち込む場合、実行する仮想インスタンス数を2で割った数が、
RHELのサブスクリプションの必要本数 (vCPUの数は関係ありません)
 - RHELのゴールドイメージ(BYOS用のEC2プライベートイメージとしてRed Hatが提供)を利用する場合、Cloud Accessという登録手続きが必要
<https://access.redhat.com/ja/articles/5855161>
 - 「RHEL サブスクリプションガイド」で検索
<https://www.redhat.com/ja/resources/red-hat-enterprise-linux-subscription-guide>
- OpenShiftのサブスクリプションガイド
 - AWSに持ち込む場合、OpenShiftワーカーノードのvCPU数の総数を4で割った数が、
OpenShiftのサブスクリプションの必要本数 (2コアまたは4vCPU単位)
 - 「OpenShift サブスクリプションガイド」で検索
<https://www.redhat.com/ja/resources/self-managed-openshift-sizing-subscription-guide>
- BYOSとPAYG (Pay-As-You-Go)については、相互変換ができません。
それぞれ独立してご購入いただくことになります。
RHELについては、PAYG RHELへのデータ移行ガイドを下記でご提示しています。
https://docs.aws.amazon.com/ja_jp/prescriptive-guidance/latest/patterns/migrate-rhel-byol-systems-to-aws-license-included-instances-by-using-aws-mgn.html



Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.



[linkedin.com/company/red-hat](https://www.linkedin.com/company/red-hat)



[facebook.com/redhatinc](https://www.facebook.com/redhatinc)



[youtube.com/user/RedHatVideos](https://www.youtube.com/user/RedHatVideos)



twitter.com/RedHat