



AWS Managed Container Service ECS or EKS or ROSA ??

Red Hat K.K.

2023.07

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/>

← ページの内容

[はじめに](#)

[理解](#)

[検討事項](#)

[選択](#)

[使用](#)

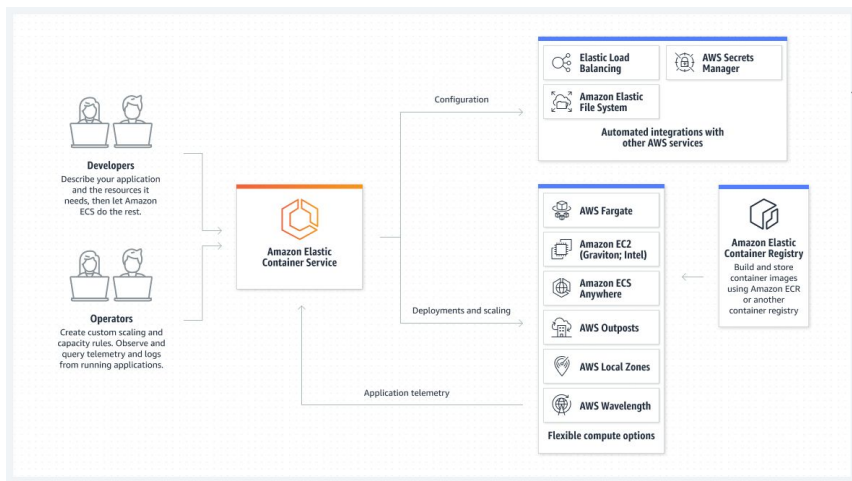
[詳しく見る](#)

コンテナオーケストレーションサービスにフォーカス

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、スケール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 ▼

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



様々なAWSサービス
との連携が可能

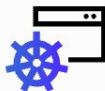
Cloud Native Computing Foundation (CNCF)

<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/blogs/opensource/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/>

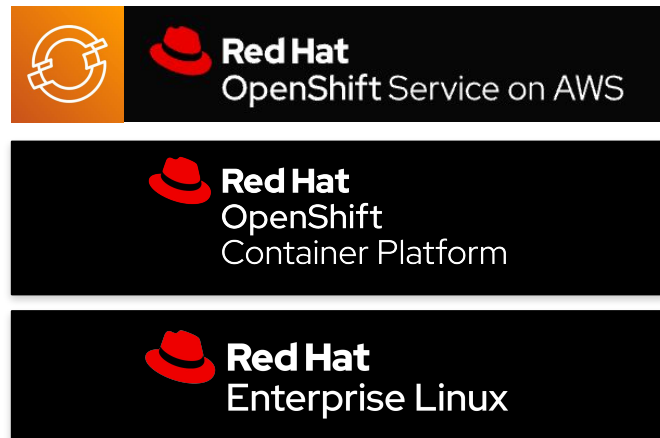
- AWS クラウドおよびオンプレで Kubernetes を実行するためのマネージド Kubernetes サービス



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

<https://www.redhat.com/ja/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を利用する設定が別途必要

Amazon EKSを使う場合

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

AWS Compute Resources

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

Kubernetesによる
システム作り込み時に
必要な主要機能

Dashboard

Developer IDE/tools

Build Automation

Pipeline (CI)

Deployment
Automation (CD)

Serverless

Service Mesh

Monitoring

Logging

Registry

Orchestration

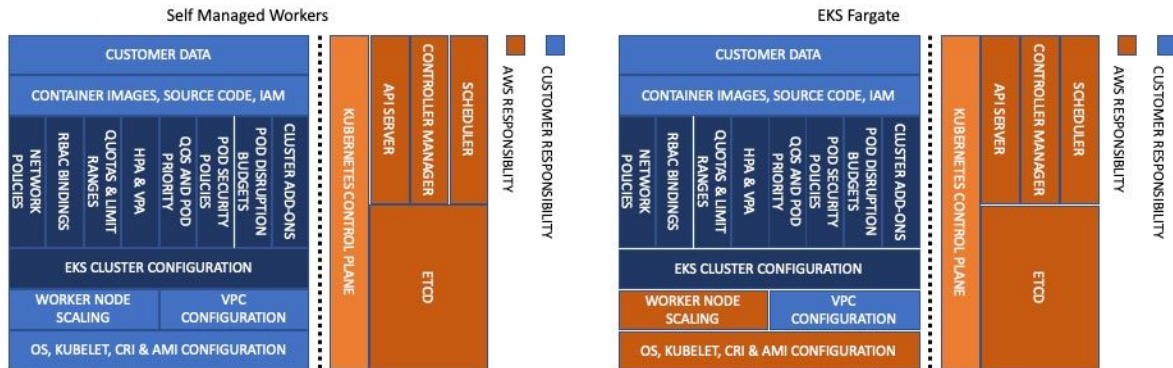
Infrastructure

Amazon EKS Shared Responsibility Model

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

**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)**

- 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



※ 各AWSサービスの利用支援や障害復旧はAWSがサポートしますが、EKSへの統合や更新は、ユーザーが実施する必要があります。



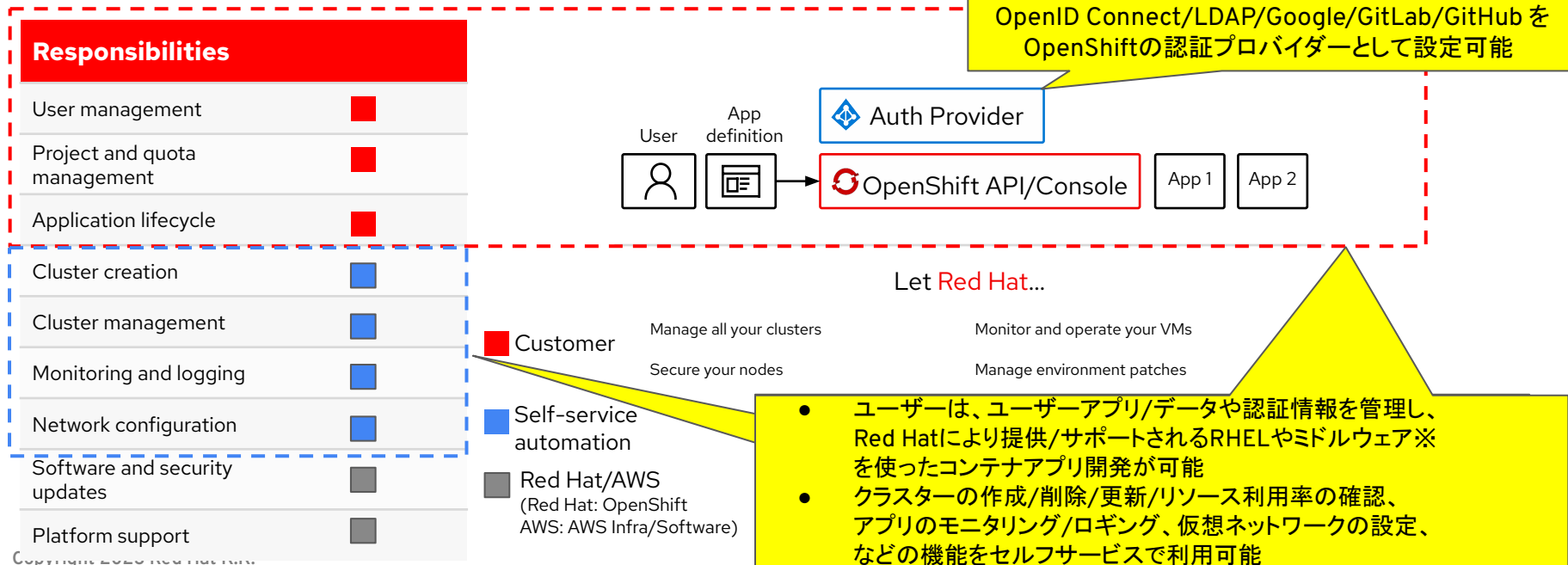
ROSA Shared Responsibility Model

https://docs.openshift.com/rosa/rosa_architecture/rosa_policy_service_definition/rosa-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



※ 一部のRed Hat提供のミドルウェア利用については、Red Hatからサブスクリプションを別途購入する必要があります。

EKS/ROSA ライフサイクル (2023年7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 Life
1.27	2023年4月11日	2023年5月24日	2024年7月
1.26	2022年12月9日	2023年4月11日	2024年6月
1.25	2022年8月23日	2023年2月22日	2024年5月
1.24	2022年5月3日	2022年11月15日	2024年1月
1.23	2021年12月7日	2022年8月11日	2023年10月11日

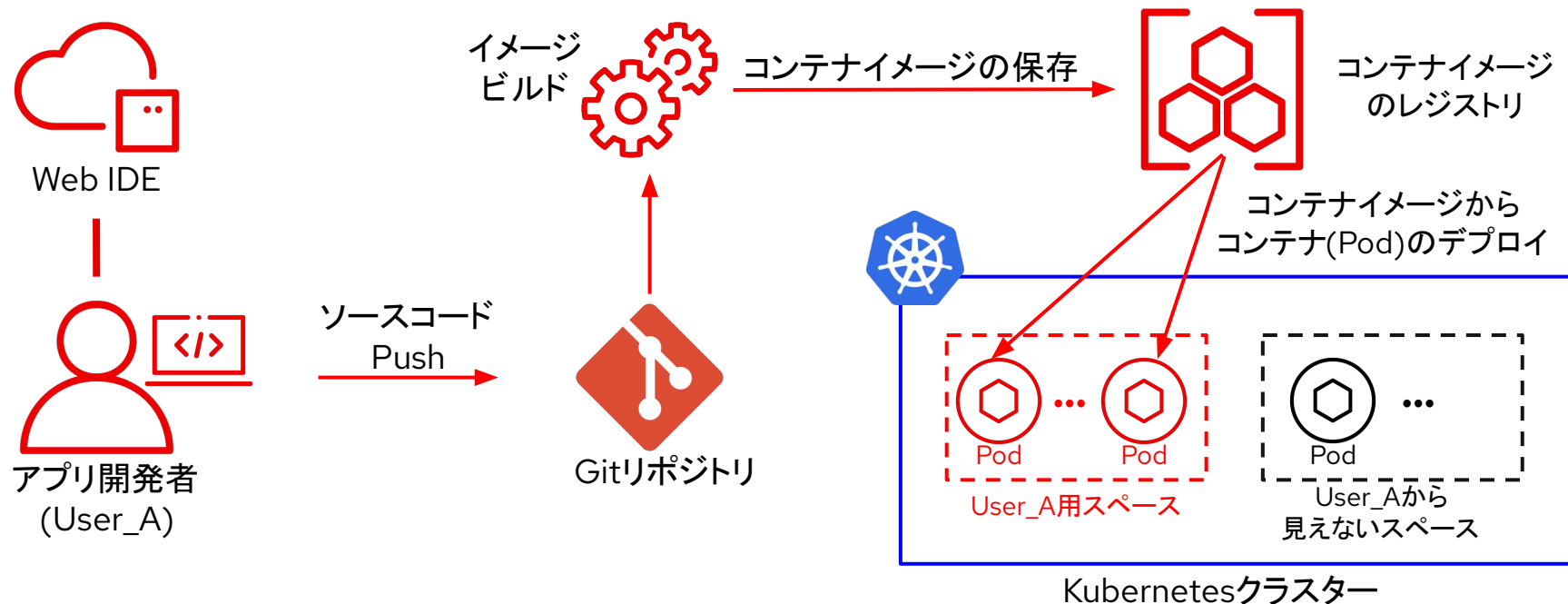
Kubernetes version	ROSA Version	ROSA release	ROSA End of Life
1.26	4.13	2023年5月17日	2024年9月17日
1.25	4.12	2023年1月17日	2024年5月17日
1.24	4.11	2022年8月10日	2023年12月10日
1.23	4.10	2022年3月10日	2023年9月10日

※ 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

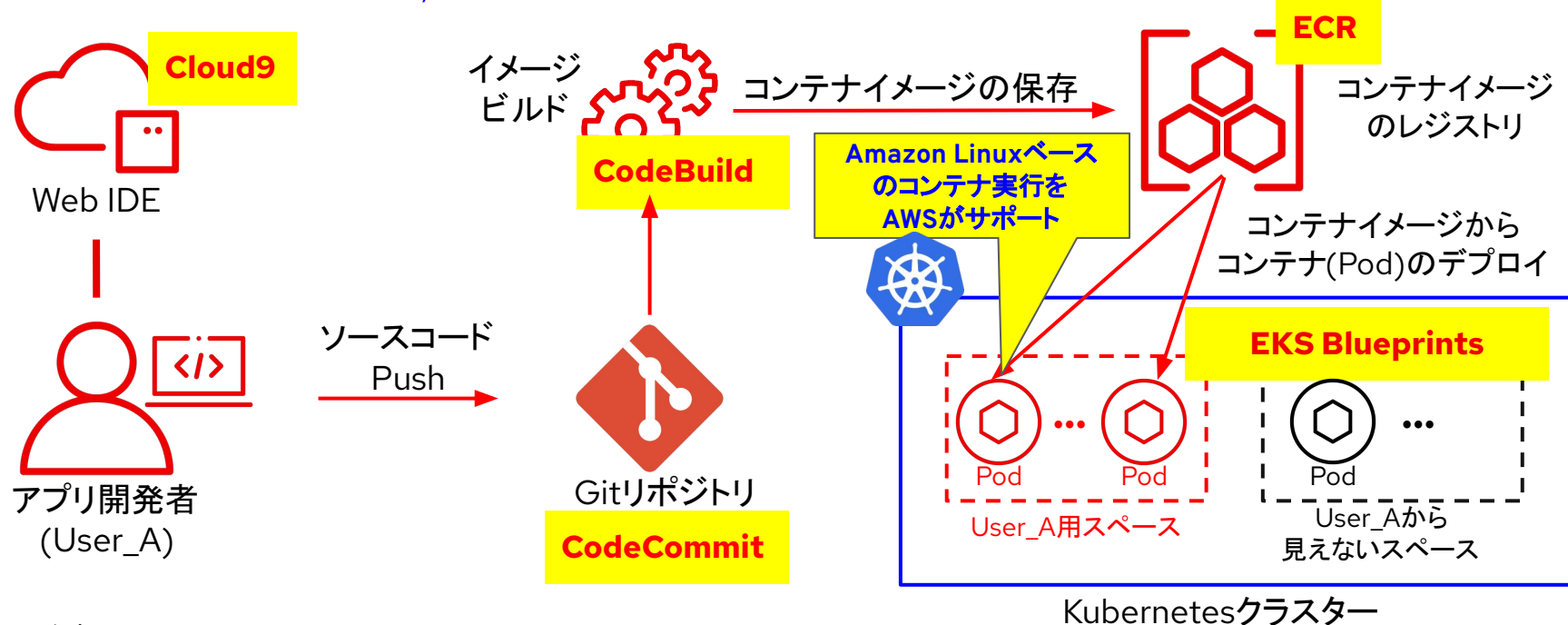
EKS/ROSA のサンプルユースケース その1

- 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



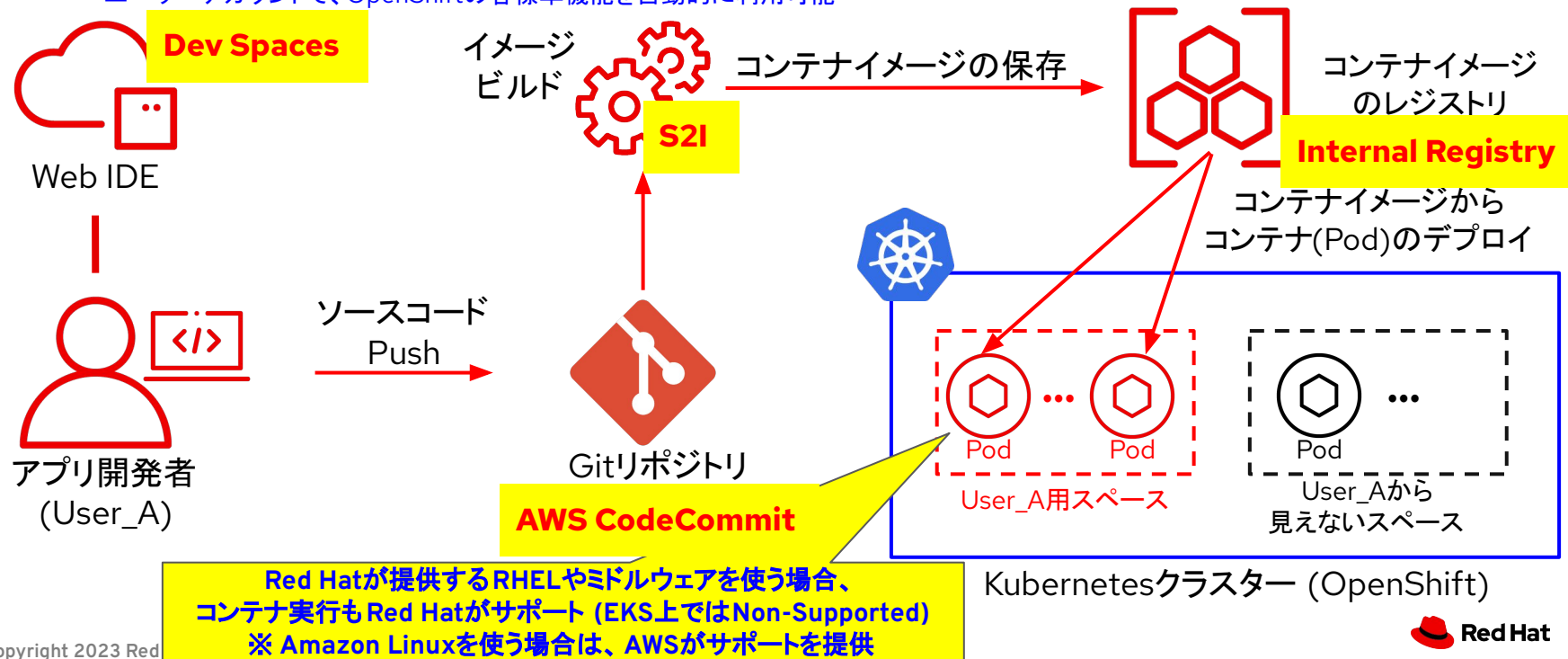
EKSを使う場合のイメージ

- 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/ポリシーによるフィルタリングと、Kubernetesクラスター内のIAM利用設定が別途必要



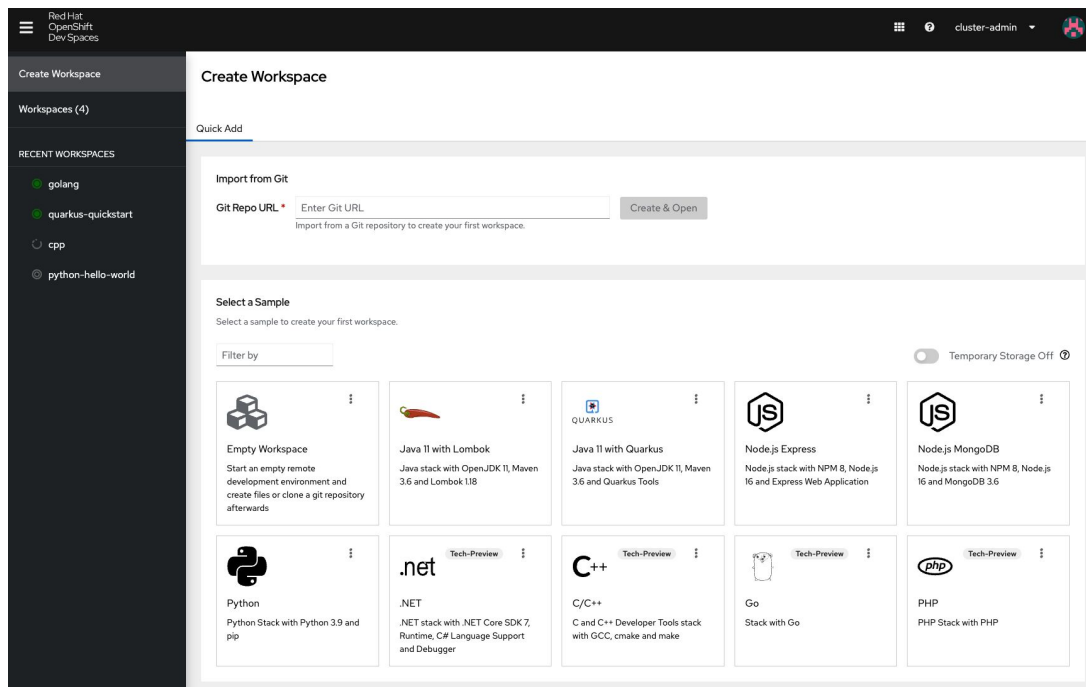
ROSAを使う場合のイメージ

- 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

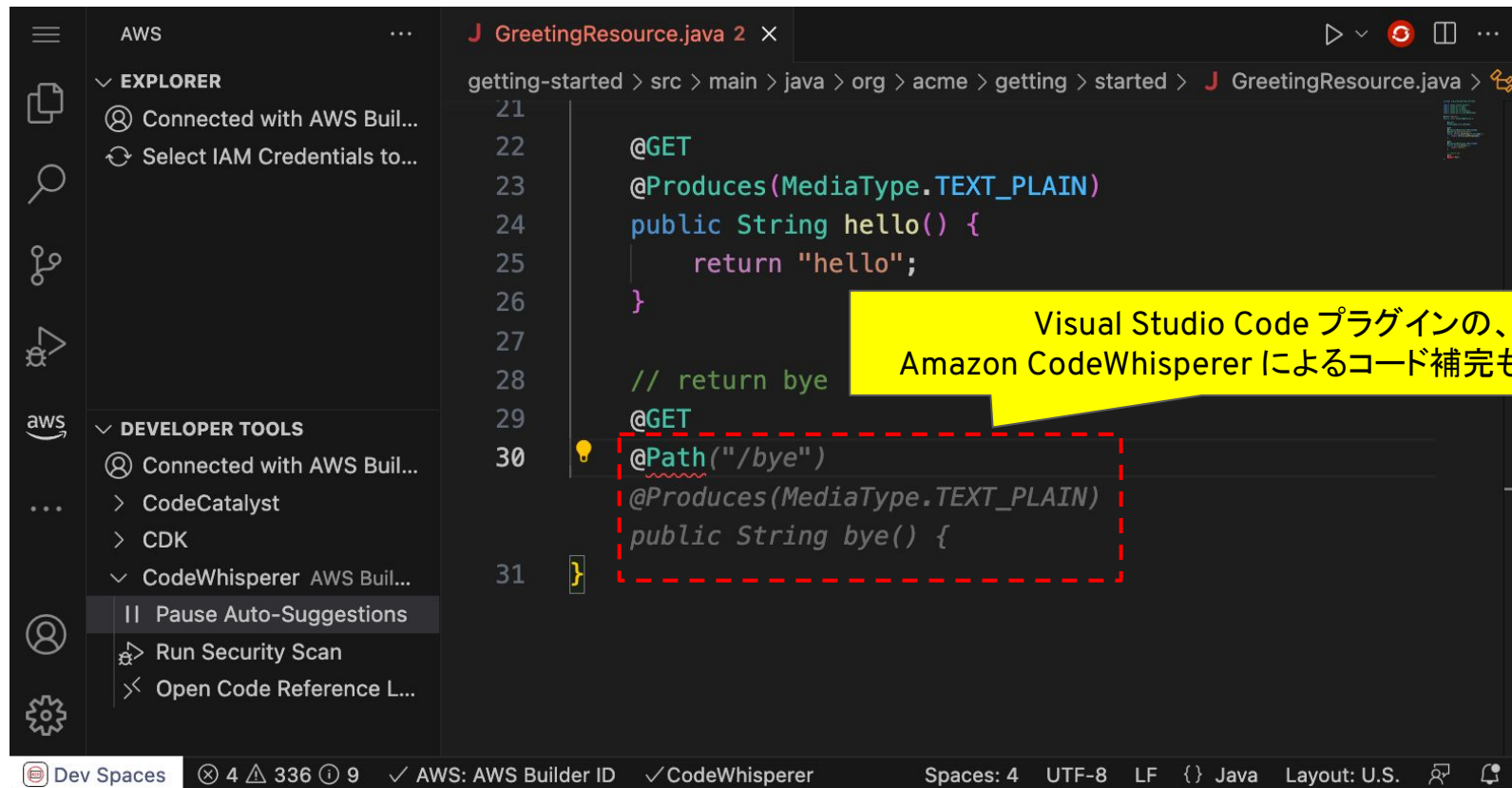
- 標準テンプレートの他に、カスタマイズされたWeb IDEも利用可能
- カスタマイズには専用のテンプレート(Devfile)※を利用



※ 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)



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

12 項目

Keywords: キーワードでフィル...

Sort: A-Z ▾




.NET

ビルダーイメージ

Red Hat による提供


Build and run .NET 7 applications on UBI 8. For more information about using this builder image...



Apache HTTP Server (httpd)

ビルダーイメージ


Build and serve static content via Apache HTTP Server (httpd) 2.4 on RHEL 7. For more information...



Go

ビルダーイメージ

Build and run Go applications on UBI 7. For more information about using this builder image, including...




JBoss EAP XP 3.0 with OpenJDK 11

ビルダーイメージ

Red Hat による提供

JBoss EAP expansion pack 3.0 image for OpenShift to build and run Microservices applications o...




JBoss EAP XP 4.0 with OpenJDK 11

ビルダーイメージ

Red Hat による提供


JBoss EAP expansion pack 4.0 image for OpenShift to build and run Microservices applications o...



Nginx

ビルダーイメージ


Build and serve static content via Nginx HTTP server and a reverse proxy (nginx) on RHEL 7. For...



Node.js

ビルダーイメージ


Build and run Node.js 16 applications on UBI 8. For more information about using this...



Perl

ビルダーイメージ


Build and run Perl 5.32 applications on UBI 8. For more information about using this...



PHP

ビルダーイメージ


Build and run PHP 8.0 applications on UBI 8. For more information about using this...



Python

ビルダーイメージ

Build and run Python 3.9 applications on UBI 8. For more information about using this...




Red Hat OpenJDK

ビルダーイメージ

Red Hat, Inc. による提供

Build and run Java applications using Maven and OpenJDK 17.



Ruby

ビルダーイメージ

Build and run Ruby 3.0 applications on UBI 7. For more information about using this...

OpenShift S2I の画面

The screenshot displays the OpenShift S2I (Source-to-Image) application creation interface. The interface is divided into two main panels: the left panel for configuration and the right panel for details.

Left Panel: Source-to-Image (S2I) アプリケーションの作成

- ビルドイメージのバージョン (Build Image Version):** A dropdown menu showing various Python 3.9 and 3.8 UBI images. A yellow callout points to this section, stating: **コンテナイメージビルド用のコンテナの選択** (Selection of container for container image build).
- Python 3.9 (UBI 8):** The selected build image, with a description: "Build and run Python 3.9 applications on UBI 8. For more information about using this builder image, including OpenShift considerations, see <https://github.com/sclorg/s2i-python-container/blob/master/3.9/README.md>. サンプルリポジトリ: <https://github.com/sclorg/django-ex.git>".
- Git:** A section for the source code repository. A yellow callout points to the "Git リポジトリ URL" field, stating: **ソースコード置き場となるGitリポジトリのURL** (URL of the Git repository where the source code is stored).
- Git リポジトリ URL:** A text input field containing `https://github.com/sclorg/django-ex.git`.
- コードをビルドおよびデプロイするためのリポジトリ URL:** A text input field for the repository URL used for building and deploying the code.
- サンプルを試す (Try Sample):** A button to try the sample application.
- 詳細の Git オプションの表示 (Show Detailed Git Options):** A link to view detailed Git options.

Right Panel: 一般 (General)

- アプリケーション (Application):** A dropdown menu showing "sample-python". Below it, a note says: "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。" (Select an application to group this component).
- 名前 (Name):** A text input field containing "sample-python01". Below it, a note says: "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。" (A unique name to specify the component, used to name related resources).
- リソースタイプ (Resource Type):** A dropdown menu showing "Deployment". Below it, a note says: "Resource type to generate. The default can be set in [User Preferences](#)."
- 詳細オプション (Detailed Options):** A section for advanced configuration. A yellow callout points to this section, stating: **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** (Options to specify automatic execution of application creation and publication).
 - ターゲットポート (Target Port):** A text input field containing "8080". Below it, a note says: "トラフィックのターゲットポート。" (Target port for traffic).
 - route を作成する (Create route):** A checkbox that is checked. Below it, a note says: "パブリック URL でコンポーネントを公開します" (Publish the component with a public URL).
 - 詳細なルーティングオプションを表示する (Show detailed routing options):** A link to show detailed routing options.
- 名前をクリックして、ヘルスチェック、ビルド設定、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。 (Click the name to access detailed options for health checks, build configuration, Deployment, scaling, resource limits, and labels.)**
- 作成 (Create):** A button to create the application.
- キャンセル (Cancel):** A button to cancel the operation.

Bottom Panel: 一般 (General)

- 「作成」クリックにより、アプリ作成と公開を自動実行 (Clicking "Create" will automatically execute application creation and publication)**

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)



D django-ex アクション

詳細 リソース モニタリング

1 Pod

名前
django-ex

更新ストラテジー
RollingUpdate

Namespace
test-project01

利用できない Pod の最大値
25%/1 Pod

ラベル 編集

Pod 増分の最大値
1 Pod を 25% 超える

進行の期限 (秒)
600 秒

最小の準備状態 (秒)
未設定

PodDisruptionBudget
PodDisruptionBudgetがありません

Pod セレクター
app=django-ex

Node セレクター
セレクターなし



D django-ex アクション

詳細 リソース モニタリング

Pod

django-ex-85fcccc7c4-wtr5f Running ログの表示

Build

BC django-ex ビルドの開始

ビルド #1 は完了しました (20 分前) ログの表示

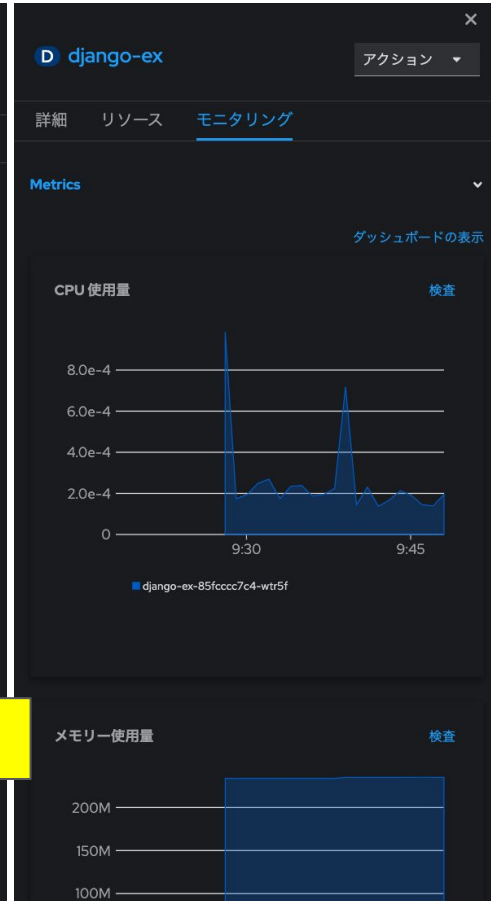
Service

S django-ex
サービスポート: 8080-tcp → Pod ポート: 8080

Route

RT django-ex
場所:
https://django-ex-test-project01.apps.rosa.hcp-cluster01.6tzy.p3.openshiftapps.com

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)



D django-ex アクション

詳細 リソース モニタリング

Metrics

ダッシュボードの表示

CPU 使用量 検査

メモリ使用量 検査

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では、ユーザーのAWSアカウント上にあるInfra/ControlのEC2インスタンスが無くなり、料金請求も発生しなくなります。

- 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 general1.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	

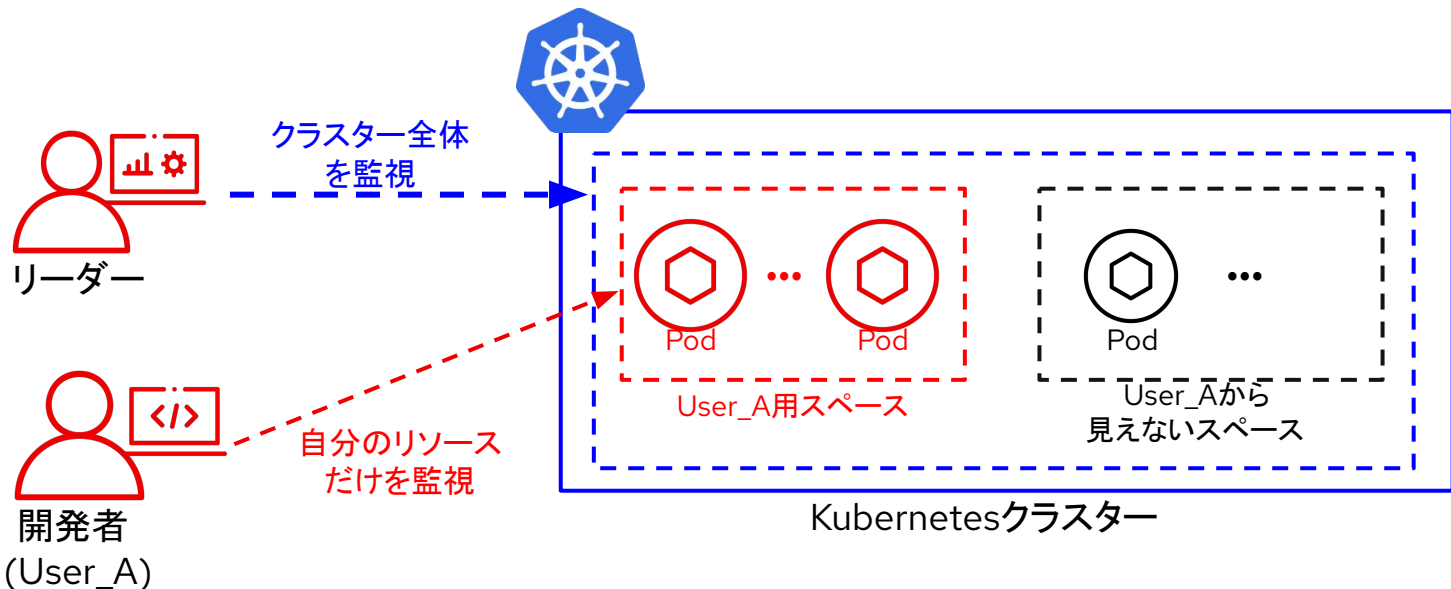
AWSサービス	利用料金 (USD)
ROSA サービス料金 (Cluster fee: 0.03USD/hour/cluster Worker Node service fee: 0.171USD/4vCPU/hour)	396.39 (0.03 x 730 + 0.171 x 3 x 730)
EC2 インスタンス (Worker) (m5.xlarge x3, 0.248USD/hour EBS(gp3): 300GB, 0.08USD/GB/month)	615.12 (0.248 x 3 x 730 + 300 x 3 x 0.08)
EC2 インスタンス (Infra)※ (r5.xlarge x3, 0.304USD/hour EBS(gp3): 300GB, 0.08USD/GB/month)	737.76 (0.304 x 3 x 730 + 300 x 3 x 0.08)
EC2 インスタンス (Control)※ (m5.2xlarge x3, 0.496USD/hour EBS(Provisioned io1): 350GB, 0.142USD/GB/month)	1235.34 (0.496 x 3 x 730 + 350 x 3 x 0.142)
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): 2987.11	

EKS/ROSA

サンプルユースケース その2

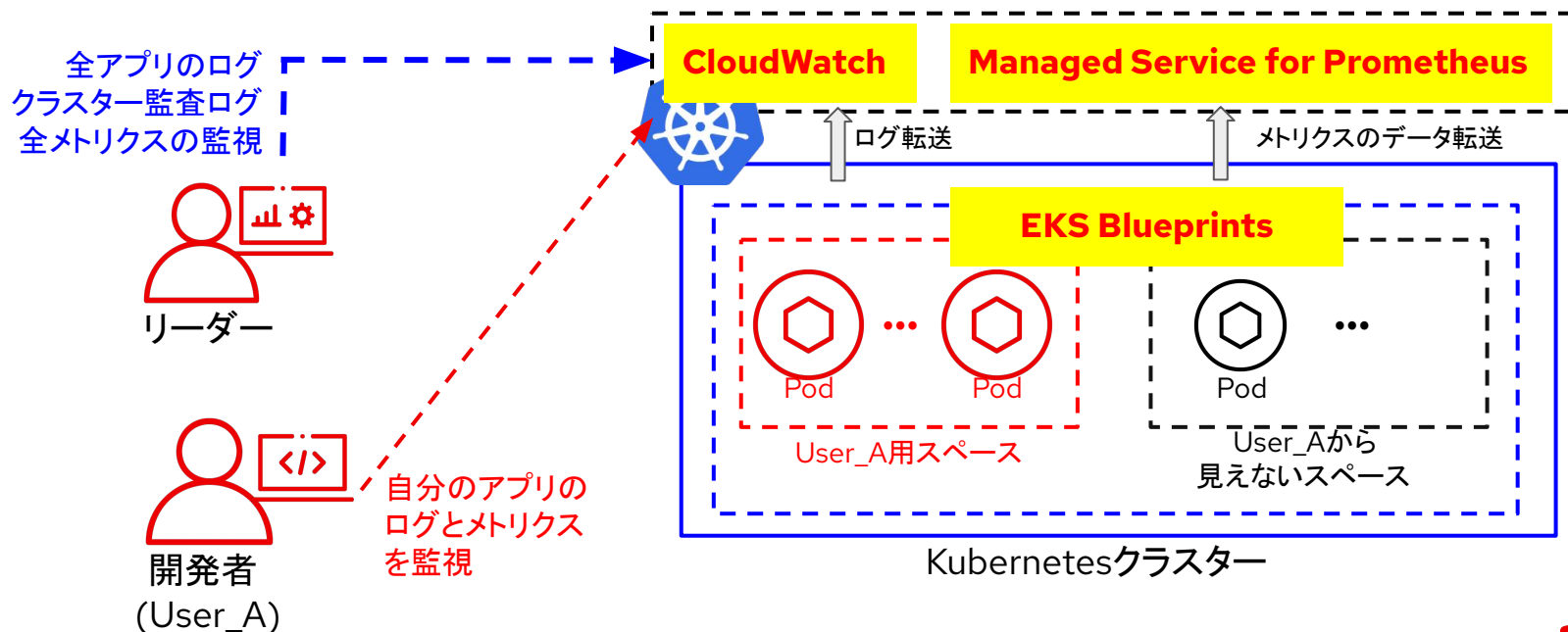
EKS/ROSA のサンプルユースケース その2

- 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



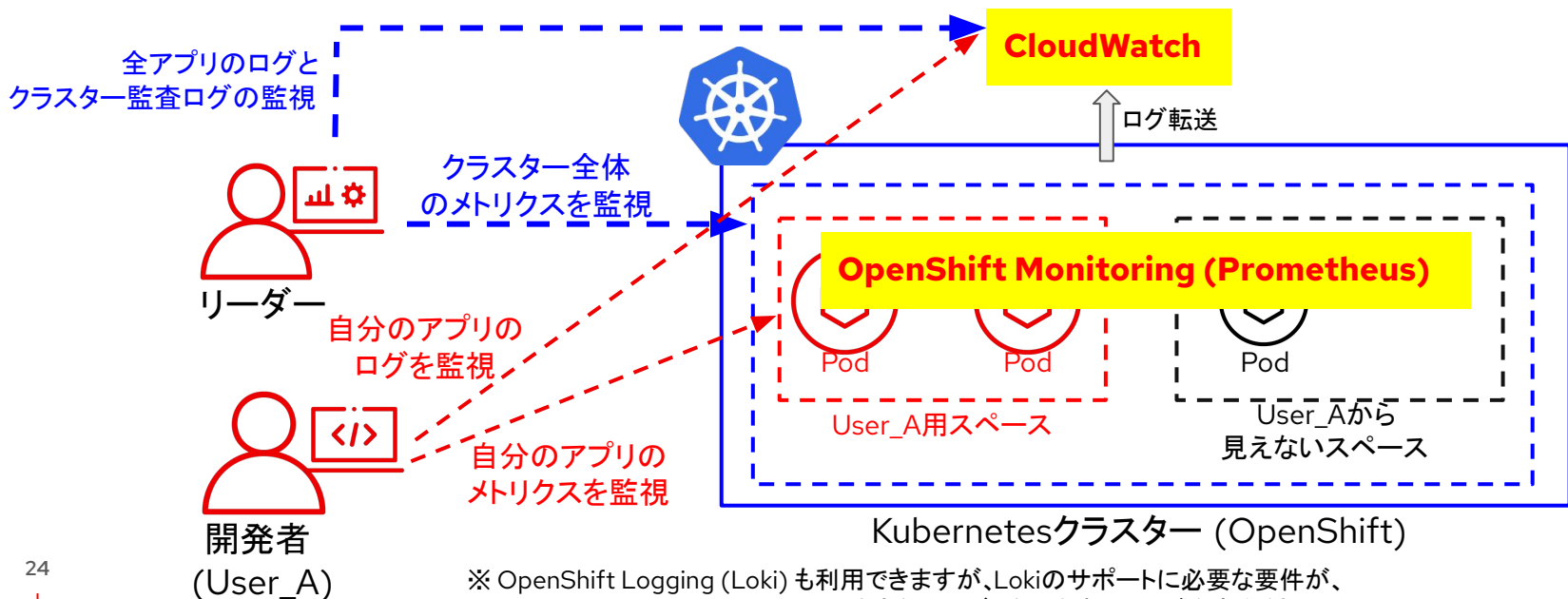
EKSを使う場合のイメージ

- ロギングには、Amazon CloudWatchを利用 (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/ポリシーによるフィルタリングや、データ転送のための Kubernetesクラスター内での IAM利用設定が別途必要



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、インシデント調査のための監査ログリクエストも可能

CloudWatch > ロググループ

ロググループ (4)

デフォルトでは、最大 10000 個のロググループのみをロー

🔍 ロググループをフィルタリングするか、検索を試す

☐ 完全一致

<input type="checkbox"/>	ロググループ	データ保護	機密データの数	保持	メトリクスフィ...	寄稿者のインサ
<input type="checkbox"/>	rosa-sample-cluster01.application	-	-	失効しない	-	-
<input type="checkbox"/>	rosa-sample-cluster01.audit	-	-	-	-	-
<input type="checkbox"/>	rosa-sample-cluster01.infrastructure	-	-	-	-	-
<input type="checkbox"/>	rosa-sample-cluster01.test-project01	-	-	-	-	-

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.django-psql-persistent.0.log

ログイベント

下のフィルターバーを使用して、ログイベント内の用語、語句、値の検索や照合ができます。 [フィルターパターンの詳細](#)

アクション ▼ テーリングを開始 メトリクスフィルターを作成

🔍 イベントをフィルター

クリア 1m 30m 1h 12h カスタム 表示 ▼

▶ タイムスタンプ メッセージ

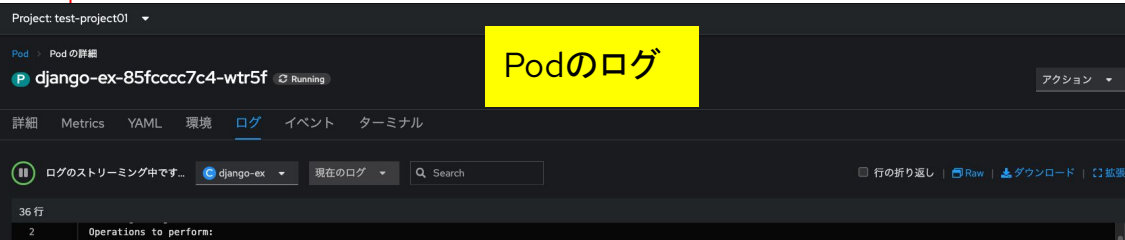
ロードする古いイベントがあります。 [さらにロードします。](#)

```
2023-07-07T15:10:28.123+09:00 {"@timestamp": "2023-07-07T06:10:18.025299232Z", "group_name": "rosa-sample-cluster01.test-project01", "hostname": "ip-10-0-1-169.us-east-2.compute.internal", "kubernetes": {"annotations": {"k8s.ovn.org/pod-networks": "{\"default\": {\"ip_addresses\": [\"10.128.1.30/23\"], \"mac_address\": \"0a:58:0a:80:01:1e\", \"gateway_ips\": [\"10.128.0.1\"]}, \"ip_address\": \"10.128.1.30/23\", \"gateway_ip\": \"10.128.0.1\"}}, \"k8s.v1.cni.cncf.io/network-status\": \"{\\n  \\\"name\\\": \\\"ovn-kubernetes\\\",\\n  \\\"interface\\\": \\\"eth0\\\",\\n  \\\"ip_addresses\\\": [\\n    \\\"10.128.1.30\\\",\\n    \\\"mac\\\": \\\"0a:58:0a:80:01:1e\\\",\\n    \\\"default\\\": true,\\n    \\\"dns\\\": {\\n\\n}}\", \"k8s.v1.cni.cncf.io/networks-status\": \"{\\n  \\\"name\\\": \\\"ovn-kubernetes\\\",\\n  \\\"interface\\\": \\\"eth0\\\",\\n  \\\"ip_addresses\\\": [\\n    \\\"10.128.1.30\\\",\\n    \\\"mac\\\": \\\"0a:58:0a:80:01:1e\\\",\\n    \\\"default\\\": true,\\n    \\\"dns\\\": {\\n\\n}}\", \"openshift.io/deployment-config.latest-version\": \"1\", \"openshift.io/deployment-config.name\": \"django-psql-persistent\", \"openshift.io/deployment.name\": \"django-psql-persistent-1\", \"openshift.io/scc\": \"restricted-v2\", \"seccomp.security.alpha.kubernetes.io/pod\": \"runtime/default\"\"}, {\"@timestamp\": \"2023-07-07T06:10:18.025299232Z\", \"group_name\": \"rosa-sample-cluster01.test-project01\", \"hostname\": \"ip-10-0-1-169.us-east-2.compute.internal\", \"kubernetes\": {\"annotations\": {\"k8s.ovn.org/pod-networks\": \"{\\n  \\\"default\\\": {\\n    \\\"ip_addresses\\\": [\\n      \\\"10.128.1.30/23\\\",\\n    \\\"mac_address\\\": \\\"0a:58:0a:80:01:1e\\\",\\n    \\\"gateway_ips\\\": [\\n        \\\"10.128.0.1\\\"\\n      ],\\n    \\\"ip_address\\\": \\\"10.128.1.30/23\\\",\\n    \\\"gateway_ip\\\": \\\"10.128.0.1\\\"\\n    }\\n  },\\n  \\\"k8s.v1.cni.cncf.io/network-status\\\": \"{\\n    \\\"name\\\": \\\"ovn-kubernetes\\\",\\n    \\\"interface\\\": \\\"eth0\\\",\\n    \\\"ip_addresses\\\": [\\n      \\\"10.128.1.30\\\",\\n      \\\"mac\\\": \\\"0a:58:0a:80:01:1e\\\",\\n      \\\"default\\\": true,\\n      \\\"dns\\\": {\\n\\n    }\\n  },\\n  \\\"k8s.v1.cni.cncf.io/networks-status\\\": \"{\\n    \\\"name\\\": \\\"ovn-kubernetes\\\",\\n    \\\"interface\\\": \\\"eth0\\\",\\n    \\\"ip_addresses\\\": [\\n      \\\"10.128.1.30\\\",\\n      \\\"mac\\\": \\\"0a:58:0a:80:01:1e\\\",\\n      \\\"default\\\": true,\\n      \\\"dns\\\": {\\n\\n    }\\n  },\\n  \\\"openshift.io/deployment-config.latest-version\\\": \\\"1\\\",\\n  \\\"openshift.io/deployment-config.name\\\": \\\"django-psql-persistent\\\",\\n  \\\"openshift.io/deployment.name\\\": \\\"django-psql-persistent-1\\\",\\n  \\\"openshift.io/scc\\\": \\\"restricted-v2\\\",\\n  \\\"seccomp.security.alpha.kubernetes.io/pod\\\": \\\"runtime/default\\\"\"\"}
```

ユーザーアプリのログを参照可能

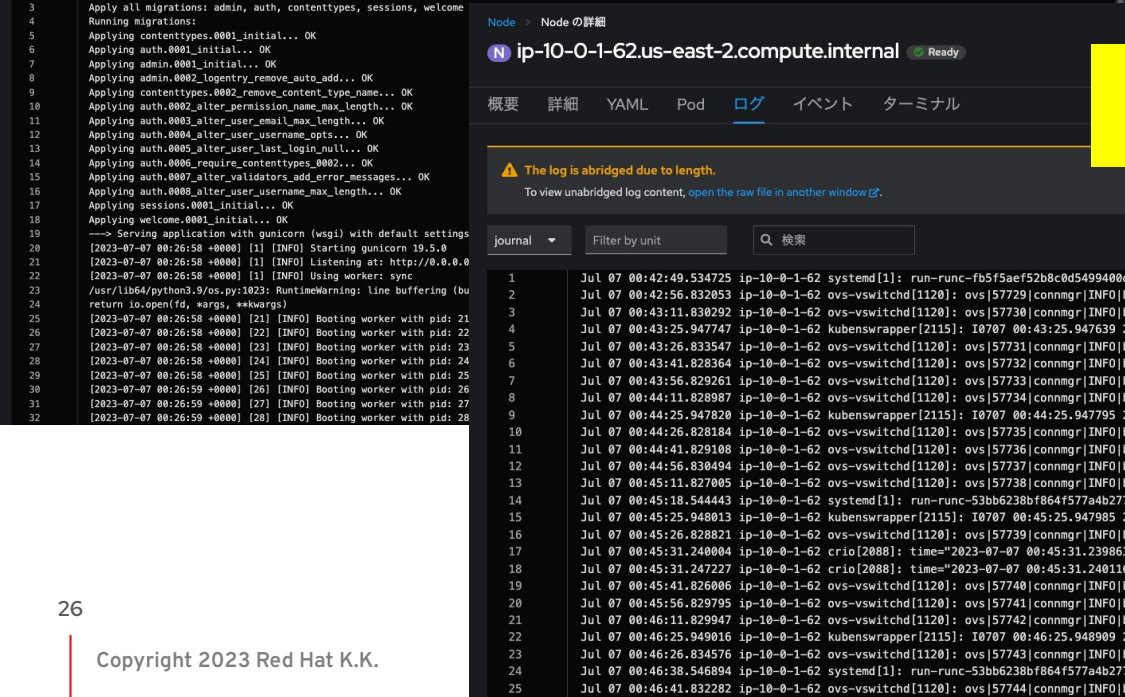
コピー

OpenShiftのコンソールでのログ確認



Podのログ

アクション



Nodeの詳細

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

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

行の折り返し

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

Project: test-project01

Pod

Pod の作成

フィルター 名前 名前を検索...

名前	ステータス	準備完了	再起動回数	オーナー	メモリー	CPU	作成
django-ex-85fcccc7c4-wtr5f	Running	1/1	0	RS django-ex-85fcccc7c4	231.9 MiB	0.000 コア	2023年7月7日 9:26
django-psql-persistent-l-rqzqj	Running	1/1	0	RC django-psql-persistent-l	247.5 MiB	0.001 コア	2023年7月2日 13:37
postgresql-l-8ff4m	Running	1/1	0	RC postgresql-l	34.6 MiB	0.010 コア	2023年7月2日 13:36
sample-net-app01-57b9df9ff7-fpv52	Running	1/1	0	RS sample-net-app01-57b9df9ff7	113.7 MiB	0.000 コア	2023年7月2日 13:42

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

メトリクスの時間範囲や
更新間隔も適宜設定可能

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では、ユーザーのAWSアカウント上にあるInfra/ControlのEC2インスタンスが無くなり、料金請求も発生しなくなります。

- 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 (P.20 参照)
- 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: 0.03USD/hour/cluster Worker Node service fee: 0.171USD/4vCPU/hour)	396.39 (0.03 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)
EC2 インスタンス (Infra)※ (r5.xlarge x3, 0.304USD/hour EBS(gp3): 300GB, 0.08USD/GB/month)	737.76 (0.304 x 3 x 730 + 300 x 3 x 0.08)
EC2 インスタンス (Control)※ (m5.2xlarge x3, 0.496USD/hour EBS(Provisioned io1): 350GB, 0.142USD/GB/month)	1235.34 (0.496 x 3 x 730 + 350 x 3 x 0.142)
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): 2992.543	

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IAM利用設定を実現可能
 - 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IAM利用設定の手間を、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

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