Red Hat Fuse 7.3

Fuse の管理

Howtio および Prometheus を使用した Fuse アプリケーションの管理

Last Updated: 2021-02-10

Red Hat Fuse 7.3 Fuse の管理

Howtio および Prometheus を使用した Fuse アプリケーションの管理

法律上の通知

概要

Fuse アプリケーションをデプロイする場合、Fuse Console (Hawtio) と Prometheus の両方を使用すると、Red Hat Fuse インテグレーションを監視し、対話することができます。

目次

前書き

Red Hat Fuse は、Fuse インテグレーションを表示および管理する、以下の 2 つのエンタープライズ管理ツールを提供します。

- Fuse Console は、ブラウザーからアクセスする web ベースのコンソールで、実行中の Fuse コンテナーを監視および管理します。Fuse Console は Hawtio オープンソースソフトウェア (http://hawt.io/) をベースにしています。
- Prometheus (https://prometheus.io/docs/introduction/overview/) は、Fuse ディストリビューションのシステムおよびインテグレーションレベルのメトリクスを保存します。Grafana などのグラフィカル分析インターフェースを使用して、保存された履歴データを表示および分析できます。

本ガイドの対象読者は Red Hat Fuse の管理者です。本ガイドの読者は、Red Hat Fuse プラットフォーム、Apache Camel、および所属組織の処理要件をよく理解していることを前提としています。

Fuse アプリケーションをデプロイする場合、Fuse Console (Hawtio) と Prometheus の両方を使用すると、Red Hat Fuse インテグレーションを監視し、対話することができます。

Fuse Console

Fuse Console は、デプロイされた 1 つ以上の Fuse コンテナーの詳細を確認および管理する中央インターフェースを提供します。また、Red Hat Fuse およびシステムリソースの監視、更新の実行、およびサービスの開始と停止を行うこともできます。

Fuse Console は、Red Hat Fuse スタンドアロンをインストールしたり Fuse on OpenShift の使用すると利用することができます。Fuse Console で表示および管理できるインテグレーションは、実行されているプラグインによって異なります。使用できるプラグインには以下が含まれます。

- Camel
- JMX
- Spring Boot
- OSGI
- Runtime
- Logs

Fuse Console は web ベースのコンソールです。サポートされるブラウザーの一覧は https://access.redhat.com/articles/310603 を参照してください。

Prometheus

Prometheus は、履歴データを保存し、Fuse on OpenShift がコンポーネントの 1 つである大型でスケーラブルなシステムを監視するために構築されたコンテナーネイティブなソフトウェアです。現在実行中のセッションだけでなく、長時間にわたってデータを収集します。

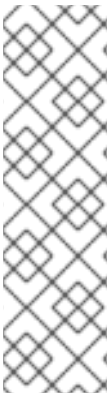
第1章 FUSE CONSOLE へのアクセス

Fuse Console にアクセスする方法は Fuse ディストリビューションによって異なります。

- Fuse on OpenShift
- Fuse スタンドアロン:
  - Spring Boot
  - Red Hat JBoss EAP
  - Apache Karaf

1.1. OPENSHIFT 上の FUSE CONSOLE へのアクセス

OpenShift Console またはコマンドラインから Fuse Console をデプロイできます。



注記

Fuse Console のセキュリティーおよびユーザー管理は、OpenShift によって処理されます。

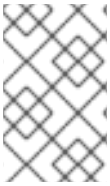
Fuse Console のテンプレートは、デフォルトでエンドツーエンド暗号化を設定するため、Fuse Console のリクエストはブラウザーからクラスター内のサービスまでエンドツーエンドでセキュア化されます。

ロールベースアクセス制御 (デプロイ後に Fuse Console にアクセスするユーザーの場合) は現在 Fuse on OpenShift では使用できません。

1.1.1. 始める前の準備 (クラスターモードの設定)

Fuse Console をクラスターモードで OpenShift Container Platform 環境にデプロイする場合、クラスター管理者ロールとクラスターモードテンプレートが必要です。以下のコマンドを実行します。

```
oc adm policy add-cluster-role-to-user cluster-admin system:serviceaccount:openshift-infra:template-instance-controller
```



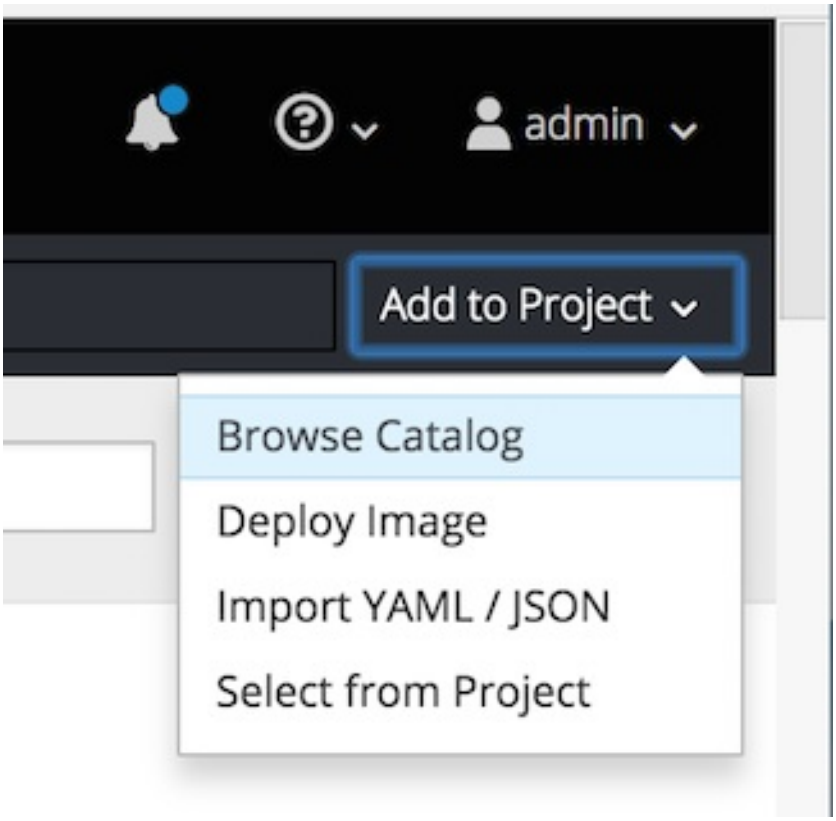
注記

クラスターモードテンプレートは、デフォルトでは OpenShift Container Platform の最新バージョンでのみ利用できます。OpenShift Online のデフォルトカタログでは提供されません。

1.1.2. OpenShift Console からの Fuse Console のデプロイ

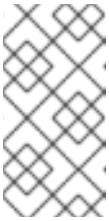
OpenShift クラスターで Fuse Console をデプロイするには、以下の手順にしたがいます。

1. OpenShift コンソールで既存のプロジェクトを開くか、新しいプロジェクトを作成します。
2. Fuse Console を OpenShift プロジェクトに追加します。
   a. **Add to Project → Browse Catalog** と選択します。



Select an item to add to the current projectページが開きます。

b. **Search** フィールドで **Fuse Console** を入力します。
Red Hat Fuse 7.x Console項目が検索結果として表示されます。



注記

Red Hat Fuse Console項目が検索結果として表示されない場合や、表示される項目が最新バージョンでない場合は、『Fuse on OpenShift Guide』の「Prepare the OpenShift server」にある説明にしたがって、Fuse Console テンプレートを手作業でインストールします。

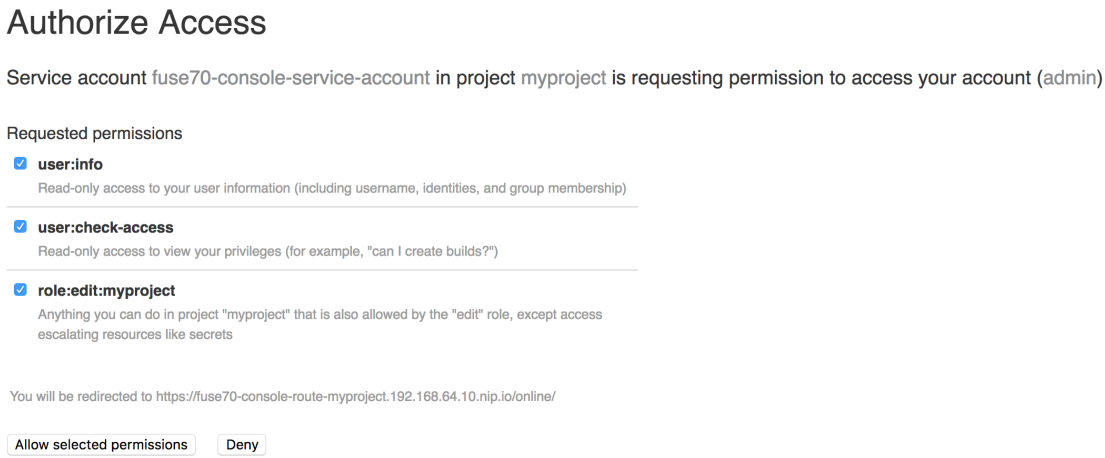
c. **Red Hat Fuse Console**項目をクリックします。
Red Hat Fuse Consoleウィザードが開きます。

d. **Next** をクリックします。ウィザードの **Configuration** ページが開きます。
任意で、設定パラメーターのデフォルト値を変更できます。

3. **Create** をクリックします。
ウィザードの **Results** ページに Red Hat Fuse Console が作成されたことが表示されます。

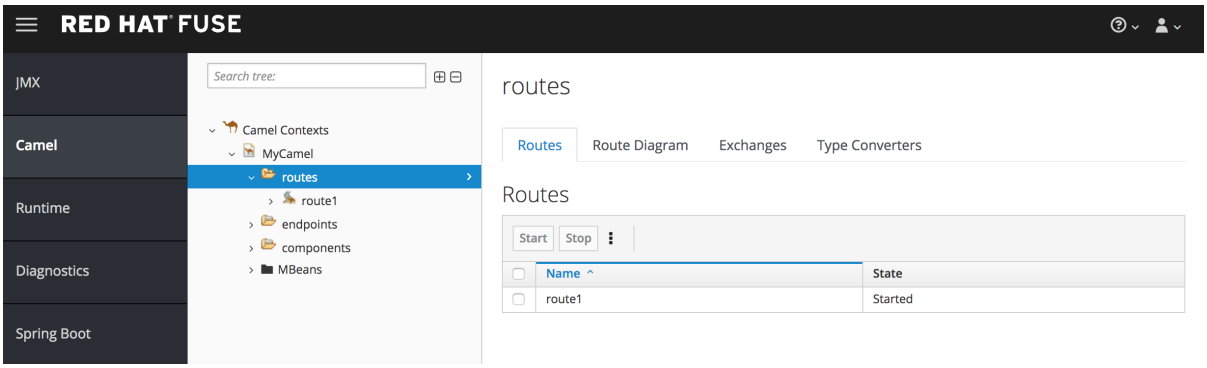
4. **Continue to the project overview**をクリックし、Fuse Console アプリケーションがプロジェクトに追加されたことを確認します。

5. Fuse Console を開き、提供された URL をクリックした後にログインします。
ブラウザーに **Authorize Access** ページが表示され、必要なパーミッションが表示されます。



6. **Allow selected permissons** をクリックします。
ブラウザーで Fuse Console が開かれ、プロジェクトで実行されている Fuse の Pod が表示されます。

7. 表示するアプリケーションの **Connect** をクリックします。
新しいブラウザーウィンドウが開かれ、Fuse Console にアプリケーションが表示されます。



1.1.3. コマンドラインからの Fuse Console のデプロイ

表1.1「Fuse Console テンプレート」 は、Fuse アプリケーションのデプロイメントのタイプに応じて、コマンドラインから Fuse Console へのアクセスに使用できる 2 つの OpenShift テンプレートについて説明しています。

表1.1 Fuse Console テンプレート

タイプ	説明
cluster	クラスター管理者ロールの作成を必要とする OAuth クライアントを使用します。Fuse Console は、複数の namespace またはプロジェクトにまたがってデプロイされた Fuse アプリケーションを検出し、接続することができます。

タイプ	説明
namespace	プロジェクトでの管理者ロールの作成のみを必要とする、サービスアカウントを OAuth クライアントとして使用します。これにより、Fuse Console のこの単一プロジェクトへのアクセスが制限されるため、単一テナントのデプロイメントとして動作します。

任意で、以下のコマンドを実行するとテンプレートパラメーターの一覧を表示できます。

```
oc process --parameters -f https://raw.githubusercontent.com/jboss-fuse/application-templates/application-templates-2.1.fuse-730065-redhat-00002/fis-console-namespace-template.json
```

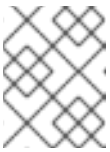
コマンドラインから Fuse Console をデプロイするには、以下を行います。

1. 以下のコマンドの 1 つを実行して、Fuse Console テンプレートをベースとした新しいアプリケーションを作成します。コマンドの **myproject** はプロジェクトの名前に置き換えます。
   - Fuse Console の **cluster** テンプレートの場合は、以下のようになります。**myhost** は Fuse Console にアクセスするホストの名前になります。

     ```
     oc new-app -n myproject -f https://raw.githubusercontent.com/jboss-fuse/application-templates/application-templates-2.1.fuse-730065-redhat-00002/fis-console-cluster-template.json -p ROUTE_HOSTNAME=myhost
     ```

   - Fuse Console の **namespace** テンプレートの場合は以下のようになります。

     ```
     oc new-app -n myproject -f https://raw.githubusercontent.com/jboss-fuse/application-templates/application-templates-2.1.fuse-730065-redhat-00002/fis-console-namespace-template.json
     ```

     注記

     namespace テンプレートの route_hostname パラメーターは OpenShift によって自動的に生成されるため、省略することが可能です。

2. 以下のコマンドを実行し、デプロイメントの状態を取得します。

   ```
   oc status
   ```

3. ブラウザーから Fuse Console にアクセスするには、提供される URL (例: https://fuse-console.192.168.64.12.nip.io) を使用します。

1.2. SPRING BOOT スタンドアロンの FUSE CONSOLE へのアクセス

スタンドアロン Fuse Spring Boot ディストリビューションの Fuse Console にアクセスするには、以下を実行します。

1. **hawtio-springboot** を Fuse アプリケーションの **pom.xml** ファイル依存関係に追加します。
   -

```
<dependency>
  <groupId>io.hawt</groupId>
  <artifactId>hawtio-springboot</artifactId>
</dependency>
```

バージョンは Maven BOM によって提供されるため、指定する必要はありません。

2. **src/main/resources/application.properties** ファイルを編集します。

   a. 以下のプロパティーを false に設定します。

      - **endpoints.jolokia.sensitive**
      - **endpoints.hawtio.sensitive**
      - **hawtio.authenticationEnabled**

   b. 以下のプロパティーを true に設定します。

      - **endpoints.hawtio.enabled**
      - **endpoints.jolokia.enabled**
        application.properties の設定は以下の例のようになるはずです。

```
# ports

server.port=8080

management.port=10001


# enable management endpoints for healthchecks and hawtio

endpoints.enabled = false

endpoints.hawtio.enabled = true

endpoints.jolokia.enabled = true

endpoints.health.enabled = true

management.health.defaults.enabled=false

camel.health.enabled=false

camel.health.indicator.enabled=true

endpoints.jolokia.sensitive=false

endpoints.hawtio.sensitive=false

hawtio.authenticationEnabled=false
```

注記

デフォルトでは、Spring Boot の Fuse Console の認証は無効になっています。任意で、Fuse Console ディストリビューションに固有するコードを作成すると、認証を有効にすることができます。例は次のとおりです。

https://github.com/hawtio/hawtio/tree/master/examples/springboot-authentication

3. Fuse アプリケーションを実行します。

```
mvn spring-boot:run
```

4. Fuse Console の URL のポート番号を特定するには、**src/main/resources/application.properties** ファイルで **management.port** の設定値を見つけます。以下に例を示します。

```
management.port   = 10001
```

5. ブラウザーで Fuse Console を開くには、以下の URL 構文を使用します。**nnnnn** は **management.port** プロパティーの値に置き換えます。
http://localhost:nnnnn/hawtio/index.html

   例: **http://localhost:10001/hawtio/index.html**

1.3. JBOSS EAP スタンドアロンの FUSE CONSOLE へのアクセス

Red Hat JBoss Enterprise Application Platform の Fuse Console にアクセスする前に、JBoss EAP コンテナーに Fuse をインストールする必要があります。ステップごとの手順は、『Installing on JBoss EAP』を参照してください。

スタンドアロン JBoss EAP ディストリビューションの Fuse Console にアクセスするには、以下を実行します。

1. Red Hat Fuse スタンドアロンを以下のコマンドで起動します。
Linux/Mac OS の場合: **./bin/standalone.sh**

   Windows の場合: **./bin/standalone.bat**

2. Web ブラウザーで URL を入力し、Fuse Console に接続します。例:
http://localhost:8080/hawtio

3. ログインページでユーザー名とパスワードを入力し、**Log In** をクリックします。

デフォルトでは、Fuse Console に Home ページが表示されます。左側のナビゲーションタブには実行中のプラグインが表示されます。

1.3.1. Fuse Console のロード遅延の解決

Fuse Console のメインページがブラウザーに表示されるまで時間がかかる場合、ログファイルの数とサイズを削減する必要がある場合があります。ログファイルの最大サイズ (rotate-size) に到達し、ファイル数を維持する (max-backup-index) 場合は、**periodic-size-rotating-file-handler** を使用してファイ

ルのローテーションを行うことができます。このハンドラーの使用方法に関する詳細は、Red Hat JBoss Enterprise Application Platform の製品ドキュメントを参照してください。

1.4. KARAF スタンドアロンの FUSE CONSOLE へのアクセス

Apache Karaf スタンドアロンの Fuse Console へアクセスするには、以下を行います。

1. Karaf コンテナーに Fuse をインストールします。ステップごとの手順は『Installing on Apache Karaf』を参照してください。
2. コマンドラインで、Red Hat Fuse をインストールしたディレクトリーに移動し、以下のコマンドを実行して Fuse スタンドアロンを起動します。

   ```
   ./bin/fuse
   ```

   Karaf コンソールが起動し、バージョン情報、デフォルトの Fuse Console URL、および一般的なコマンドのリストが表示されます。
3. ブラウザーで URL を入力し、Fuse Console に接続します。例: **http://localhost:8181/hawtio**
4. ログインページでユーザー名とパスワードを入力し、**Log In** をクリックします。

デフォルトでは、Fuse Console に Home ページが表示されます。左側のナビゲーションタブには実行中のプラグインが表示されます。

第2章 FUSE CONSOLE のセキュア化

以下のセキュリティー機能の一部またはすべてを実装して、スタンドアロンデプロイメントの Fuse コンテナーをセキュアにすることができます。

- 「必要なプロトコルとして HTTPS を設定」
- 「公開鍵を使用して応答をセキュアにする」
- 「SSL/TLS セキュリティーの有効化 (Karaf のみ)」
- 「ユーザーアクセスの制御 (Karaf のみ)」

2.1. 必要なプロトコルとして HTTPS を設定

hawtio.http.strictTransportSecurity プロパティーを使用すると、Web ブラウザーでセキュア HTTPS プロトコルを使用して Fuse Console にアクセスするよう要求できます。このプロパティーでは、HTTP を使用して Fuse Console へのアクセスを試みる Web ブラウザーは、要求を自動的に変換して HTTPS を使用する必要があることが指定されます。

プロパティーの構文の説明は、HTTP Strict Transport Security (HSTS) 応答ヘッダーの説明ページを参照してください。

hawtio.http.strictTransportSecurity プロパティーを以下の例のように設定します。

- Karaf スタンドアロンの場合
  以下の例のように、このプロパティーを **$KARAF_HOME/etc/system.properties** ファイルに設定します。

```
hawtio.http.strictTransportSecurity = max-age=31536000; includeSubDomains; preload
```

- JBoss EAP スタンドアロンの場合
  以下の例のように、**$EAP_HOME/standalone/configuration/standalone*.xml** ファイルの **system-properties** セクションでこのプロパティーを設定します。

```
<property name="hawtio.http.strictTransportSecurity" value="max-age=31536000;
includeSubDomains; preload"/>
```

- Spring Boot スタンドアロンの場合
  このプロパティーを以下の例のように設定します。

```
public static void main(String[] args) {
    System.setProperty("hawtio.http.strictTransportSecurity", "max-age=31536000;
includeSubDomains; preload");
    SpringApplication.run(YourSpringBootApplication.class, args);
}
```

2.2. 公開鍵を使用して応答をセキュアにする

特定の暗号の公開鍵を Fuse Console に関連付けし、偽造された証明書を使用した「中間者攻撃」のリスクを軽減するよう Web ブラウザーに要求すると、**hawtio.http.publicKeyPins** プロパティーを使用して HTTPS プロトコルをセキュアにすることができます。

プロパティーの構文や、Base64 でエンコードされた公開鍵の抽出方法の説明は、HTTP Public Key Pinning 応答ヘッダーの説明ページを参照してください。

hawtio.http.publicKeyPins プロパティーを以下の例のように設定します。

- Karaf スタンドアロンの場合
  以下の例のように、このプロパティーを **$KARAF_HOME/etc/system.properties** ファイルに設定します。

```
hawtio.http.publicKeyPins = pin-sha256="cUPcTAZWKaASuYWhhneDttWpY3oBAkE3h2+soZS7sWs"; max-age=5184000; includeSubDomains
```

- JBoss EAP スタンドアロンの場合
  以下の例のように、**$EAP_HOME/standalone/configuration/standalone*.xml** ファイルの **system-properties** セクションでこのプロパティーを設定します。

```
<property name="hawtio.http.publicKeyPins" value="pin-sha256=cUPcTAZWKaASuYWhhneDttWpY3oBAkE3h2+soZS7sWs"; max-age=5184000; includeSubDomains"/>
```

- Spring Boot スタンドアロンの場合
  このプロパティーを以下の例のように設定します。

```
public static void main(String[] args) {
    System.setProperty("hawtio.http.publicKeyPins", "pin-sha256=cUPcTAZWKaASuYWhhneDttWpY3oBAkE3h2+soZS7sWs"; max-age=5184000; includeSubDomains");
    SpringApplication.run(YourSpringBootApplication.class, args);
}
```

2.3. SSL/TLS セキュリティーの有効化 (KARAF のみ)

SSL/TLS セキュリティーは、Fuse Console ではデフォルトで有効になっていません。Fuse Console で SSL/TLS セキュリティーを有効にして、ユーザー名およびパスワードのクレデンシャルをスヌーピングから保護することが推奨されます。SSL/TLS セキュリティーを有効にする方法の詳細は、『Apache Karaf Security Guide』の「Enabling SSL/TLS for Undertow in an Apache Karaf container」を参照してください。

2.4. RED HAT SINGLE SIGN ON (KARAF)

Red Hat Single Sign-On で Fuse Console をセキュアにする方法については、『**Red Hat Single Sign-on Securing Applications and Services Guide**』の「Securing the Hawtio Administration Console」セクションを参照してください。

2.5. ユーザーアクセスの制御 (KARAF のみ)

注記

本リリースでは、Fuse Console のロールベースアクセス制御は Fuse on Karaf スタンドアロンのみで有効になっています。

表2.1「Karaf スタンドアロンでのロールベースアクセス」 に記載されているとおり、認証されたユーザーが実行できる操作はそのユーザーに割り当てられたロールによって異なります。

実行する必要のある Fuse Console 操作を実行するため、ユーザーに必要なユーザーロールの権限があることを確認してください。

ユーザーロールを設定するには、以下を行います。

1. エディターで Red Hat Fuse **etc/users.properties** ファイルを開きます。
2. ユーザー名、パスワード、およびロールのエントリーを追加します。
   たとえば、**etc/users.properties** ファイルの以下のエントリーは管理ユーザーを定義し、管理ロールを割り当てます。

   ```
   admin = secretpass,admin
   ```

3. ファイルを保存します。

表2.1 Karaf スタンドアロンでのロールベースアクセス

操作	admin	manager (マネージャー)	viewer
ログイン/ログアウト	可能	可能	可能
ヘルプトピックの表示	可能	可能	可能
ユーザー設定の指定	可能	可能	可能
接続			
リモートインテグレーションの検出および接続	可能	可能	可能
ローカルインテグレーションの検出および接続	可能	可能	可能
Camel			
実行中の Camel アプリケーションをすべて表示	可能	可能	可能
Camel コンテキストの開始、一時停止、再開、および削除	可能	可能	
メッセージの送信	可能	可能	
エンドポイントの追加	可能	可能	
ルート、ルート図、およびランタイム統計の表示	可能	可能	可能

操作	admin	manager (マネージャー)	viewer
ルートの起動と停止	可能	可能	
ルートの削除	可能	可能	
JMX			
属性値の変更	可能	可能	
タイムベースチャートで属性を選択および表示	可能	可能	可能
操作の表示	可能	可能	可能
OSGI			
バンドル、機能、パッケージ、サービス、サーバー、フレームワーク、および設定を表示	可能	可能	可能
バンドルの追加および削除	可能	可能	
設定の追加	可能	可能	
機能のインストールおよびアンインストール	可能		
Runtime			
システムプロパティー、メトリクス、およびスレッドの表示	可能	可能	可能
Logs			
ログの表示	可能	可能	可能

ロールベースアクセス制御の詳細は、『Deploying into Apache Karaf』を参照してください。

第3章 FUSE CONSOLE の無効化

Fuse Console を無効にする方法は Fuse ディストリビューションによって異なります。

3.1. OPENSHIFT

OpenShift では、Fuse Console はデフォルトでは有効になっていません。

3.2. SPRING BOOT スタンドアロン

Spring Boot では、Fuse Console はデフォルトでは有効になっていません。

3.3. JBOSS EAP スタンドアロン

JBoss EAP の Fuse Console を無効にするには、以下のいずれかを実行します。

- **$EAP_HOME/standalone/deployments/hawtio-wildfly-xxxxx.war** デプロイメントファイルを削除します。
- JBoss EAP 管理コンソールまたはコマンドラインインターフェースを使用して Fuse Console をアンデプロイします。

3.4. KARAF スタンドアロン

Karaf の Fuse Console を無効にし、他のコンポーネントに影響を与えずにすべてのユーザーがアクセスできないようにするには、以下の手順にしたがいます。

1. hawtio-web バンドル ID を特定するには、以下のコマンドを使用して Fuse Console が使用する Fuse バンドルをリストします。
   osgi:list | grep hawtio
2. バンドルを停止するには、**osgi:stop** コマンドを使用します。たとえば、**hawtio :: Web console** バンドルの ID が 246 の場合、以下のコマンドを入力します。
   osgi:stop 246

バンドルが解決状態になり、Fuse Console にアクセスできないようになります。

バンドルの管理に関する詳細は『Deploying into Apache Karaf』の「Lifecycle Management」の章を参照してください。

第4章 FUSE CONSOLE からの FUSE アプリケーションの表示および管理

以下のセクションの説明どおりに Fuse アプリケーションを表示および管理できます。

- 「コンテナーおよびアプリケーションの表示 (Fuse on OpenShift)」
- 「リモート Fuse インテグレーションへの接続 (スタンドアロンディストリビューション)」
- 「Apache Camel アプリケーションの表示および管理」
- 「JMX ドメインおよび MBean の表示および管理」
- 「OSGi 環境の表示および管理 (Karaf スタンドアロン)」
- 「スレッド状態の表示および監視」
- 「ログエントリーの表示」
- 「Fuse Console のブランディングの変更」
- 「Fuse Console でデータが正しく表示されるよう確認」

4.1. コンテナーおよびアプリケーションの表示 (FUSE ON OPENSHIFT)

OpenShift の Fuse Console にログインすると、Fuse Console のホームページに利用可能なコンテナーが表示されます。

コンテナーを管理 (作成、編集、または削除) するには、OpenShift コンソールを使用します。

OpenShift クラスターで Fuse アプリケーションを表示するには、**Online** タブをクリックします。

4.2. リモート FUSE インテグレーションへの接続 (スタンドアロンディストリビューション)

Fuse Console は Jolokia を使用します。Jolokia は、クライアントに追加のソフトウェア (エージェント) をインストールする必要がある Java Management Extensions (JMX) にエージェントベースで対応します。Red Hat Fuse には jolokia エージェントがデフォルトで含まれています。

スタンドアロン Fuse Console ディストリビューションでは、内部ですでに jolokia エージェント (https://jolokia.org/) が実行されているリモートインテグレーションに接続することができます。接続するプロセス内に jolokia エージェントがない場合は、jolokia のドキュメント (http://jolokia.org/agent.html) を参照してください。

4.2.1. Fuse Console のアンロック

デフォルトでは、Fuse 7 スタンドアロンの Jolokia (Apache Karaf および JBoss EAP 上) はロックされており、Fuse Console はリモートでアクセスできません。

locahost や **127.0.0.1** 以外のホスト名や IP アドレスの Fuse Console をアンロックするには、以下の手順にしたがいます。

1. エディターで **jolokia-access.xml** ファイルを開きます。
   Karaf では、XML ファイルは **$KARAF_HOME/etc** フォルダーにあります。

JBoss EAP では、**$EAP_HOME/standalone/configuration** フォルダーにあります。

2. Fuse Console でアクセスする Fuse インテグレーションのホスト名または IP アドレスを登録するため、**<cors>** セクションにホスト名または IP アドレスを追加します。
たとえば、Fuse Console からホスト名 **0.0.0.3** にアクセスするために追加する行は次のとおりです。

```
*<allow-origin>http://0.0.0.3:*</allow-origin>*
```

この行を次のように追加します。

```
<!--

 Cross-Origin Resource Sharing (CORS) restrictions

 By default, only CORS access within localhost is allowed for maximum security.

 You can add trusted hostnames in the <cors> section to unlock CORS access from them.

 -->

 <cors>

  <!-- Allow cross origin access only within localhost -->

  <allow-origin>http*://localhost:*</allow-origin>

  <allow-origin>http*://127.0.0.1:*</allow-origin>

  <allow-origin>http://0.0.0.3:*</allow-origin>

  <!-- Whitelist the hostname patterns as <allow-origin> -->

  <!--

  <allow-origin>http*://*.example.com</allow-origin>

  <allow-origin>http*://*.example.com:*</allow-origin>

  -->

  <!-- Check for the proper origin on the server side to protect against CSRF -->

  <strict-checking />

 </cors>
```

3. ファイルを保存します。

4.2.2. リモートアクセスの制限

任意設定として、特定のホストおよび IP アドレスの Fuse Console へのリモートアクセスを制限できます。

HTTP クライアントの IP アドレスを基にして全体的なアクセス権限を割り当てることができます。これらの制限を指定するには、以下を行います。

jolokia-access.xml ファイルで、**<host>** 要素が 1 つ以上含まれる **<remote>** セクションを追加または編集します。**<host>** 要素に対して IP アドレス、ホスト名、または CIDR 形式のネットマスク (例: 10.0 ネットワークからのすべてのクライアントは **10.0.0.0/16**) を指定できます。

以下の例は、ローカルホストと IP アドレスが **10.0** で始まるすべてのクライアントからのアクセスを許可します。他の IPアドレスの場合はアクセスが拒否されます。

```
<remote>
  <host>localhost</host>
  <host>10.0.0.0/16</host>
</remote>
```

詳細は Jolokia のセキュリティーに関するドキュメント (https://jolokia.org/reference/html/security.html) を参照してください。

4.2.3. リモート Fuse インスタンスへのコネクションの許可

Fuse Console のプロキシーサーブレットはホワイトリストを使ってホストを保護し、Fuse Console はデフォルトではローカルホストのみに接続できます。Fuse Console を他のリモート Fuse インスタンスに接続する場合は、ホワイトリストを以下のように設定する必要があります。

- Apache Karaf では、**etc/system.properties** ファイルで設定を以下のように変更します。

```
hawtio.proxyWhitelist = localhost, 127.0.0.1, myhost1, myhost2, myhost3
```

- JBoss EAP では、**standalone/configuration/standalone-*.xml** ファイルで以下の設定変更を行います。

```
 <property name=hawtio.proxyWhitelist" value="localhost, 127.0.0.1, myhost1, myhost2, myhost3"/>
```

- Spring Boot では、Spring Boot アプリケーションの **main()** メソッドで **hawtio.proxyWhitelist** システムプロパティーを以下のように設定します。

```
System.setProperty("hawtio.proxyWhitelist", "localhost, 127.0.0.1, myhost1, myhost2, myhost3");
```

4.2.4. リモート Jolokia エージェントへの接続

作業を開始する前に、リモート Jolokia エージェントのコネクション詳細 (ホスト名、ポート、およびパス) を知っておく必要があります。

Fuse ディストリビューション別の Jolokia エージェントのデフォルトコネクション URL は次のとおりです。

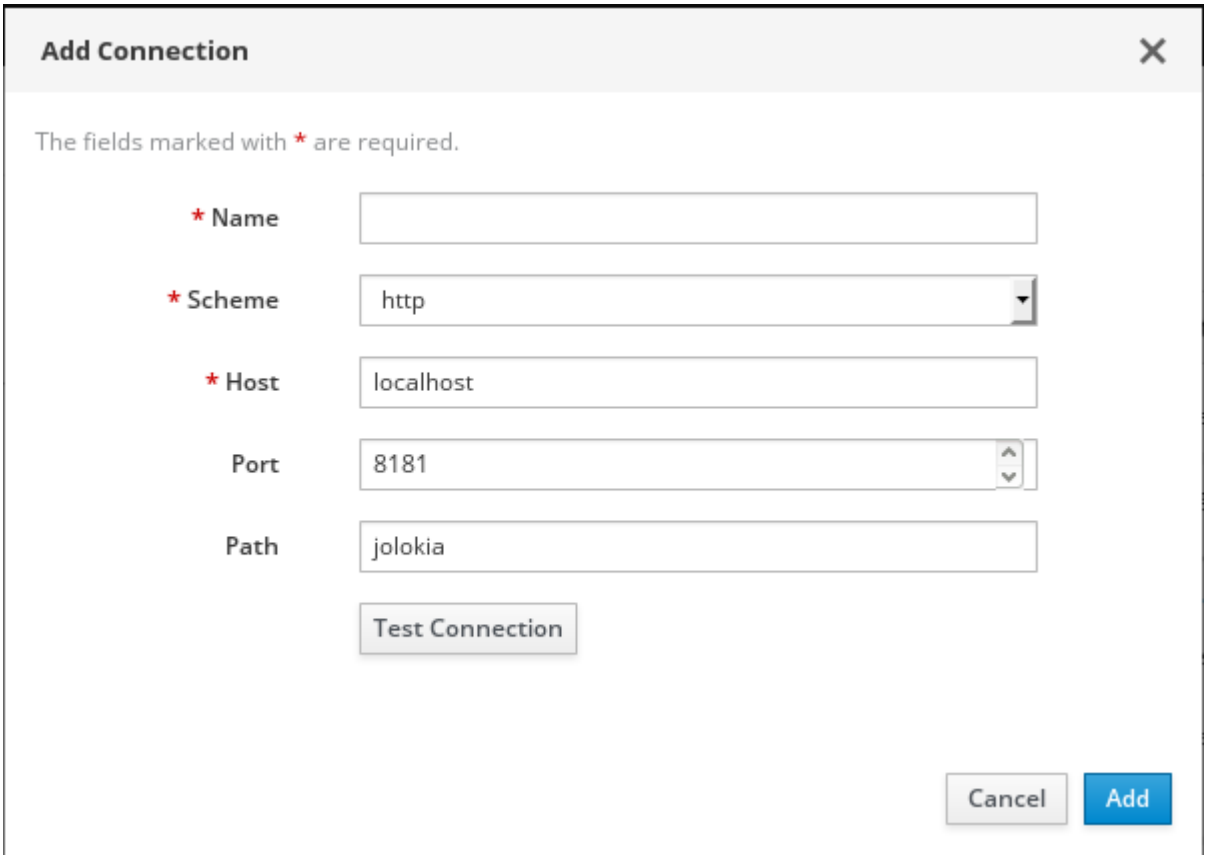
- Spring Boot: **http://<host>:8080/jolokia**
- Red Hat JBoss EAP: **http://<host>:8080/hawtio/jolokia**
- Fuse Karaf: **http://<host>:8181/hawtio/jolokia**

システム管理者はデフォルト設定を変更することができます。

通常、Jolokia エージェントにリモートで接続する URL は、Fuse Console を開く URL に **/jolokia** を追加したものです。たとえば、Fuse Console を開く URL が **http://<host>:1234/hawtio** の場合、リモート接続の URL は **http://<host>:1234/hawtio/jolokia** になります。

JVM を確認するためにリモート Jolokia インスタンスに接続するには、以下を行います。

1. **Connect** タブをクリックします。
2. **Remote** タブをクリックし、**Add connection** をクリックします。



3. **Name**、**Scheme** (HTTP または HTTPS)、および **hostname** を入力します。
4. **Test Connection** をクリックします。
5. **Add** をクリックします。

注記

Fuse Console は自動的にローカルホストと 127.0.0.1 以外のローカルネットワークインターフェースをプローブし、ホワイトリストに追加します。そのため、ローカルマシンのアドレスを手作業でホワイトリストに登録する必要はありません。

4.2.5. データ移動設定の指定

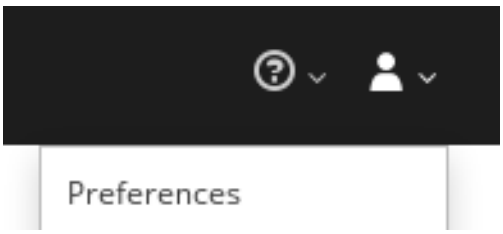
Fuse Console に表示されるデータをより頻繁にリフレッシュする場合などに、以下の Jolokia 設定を変更することができます。データの更新を頻繁に行うと、ネットワークトラフィックに影響し、サーバーに対するリクエストの数が増加するため注意してください。

- **Update rate** - JMX データを取得するため Jolokia へポーリングを行う間隔 (デフォルトは 5 秒)。

- **Maximum depth** - 戻る前にサーバー側で Jolokia がオブジェクトを JSON にマーシャルするレベル数 (デフォルトは 7)。
- **Maximum collection size** - 応答で Jolokia がマーシャルするアレイの最大要素数 (デフォルトは 50,000)。

これらの設定の値を変更するには、以下を行います。

1. Fuse Console の右上にあるユーザーアイコンをクリックして、**Preferences** をクリックします。



2. オプションを編集して **Close** をクリックします。

4.2.6. JVM ランタイム情報の表示

システムプロパティー、メトリクス、スレッドなどの JVM のランタイム情報を表示するには、**Runtime** タブをクリックします。

4.3. APACHE CAMEL アプリケーションの表示および管理

Fuse Console の **Camel** タブで Apache Camel のコンテキスト、ルート、および依存関係を表示および管理します。

4.3.1. 概要

Camel タブは、1 つ以上の Camel ルートを使用するコンテナーに接続する場合のみ使用できます。

次の詳細を表示できます。

- 実行中の Camel コンテキストすべてのリスト。
- Camel バージョン番号やランタイム統計など、各 Camel コンテキストの詳細情報。
- 各 Camel アプリケーションの全ルートおよびランタイム統計のリスト。
- 実行中のルートとリアルタイムのメトリクスのグラフィック表示。

また、以下を行うと Camel アプリケーションと対話もできます。

- コンテキストの開始および一時停止。
- 再起動、停止、一時停止、再開などを実行できるよう、すべての Camel アプリケーションとそれらのルートのライフサイクルを管理。
- 実行中のルートのライブトレースおよびデバッグ。
- Camel エンドポイントへのメッセージの閲覧および送信。

4.3.2. Camel アプリケーションとの対話。

コンテキストを開始、一時停止、または削除するには、以下を行います。

1. **Camel** タブのツリービューで、**Camel Contexts** をクリックします。
2. リストのコンテキストの横にあるボックスにチェックマークを入れます。
3. **Start** または **Suspend** をクリックします。
4. コンテキストを最初に停止してから削除する必要があります。停止したら楕円のアイコンをクリックし、ドロップダウンメニューで **Delete** を選択します。

注記

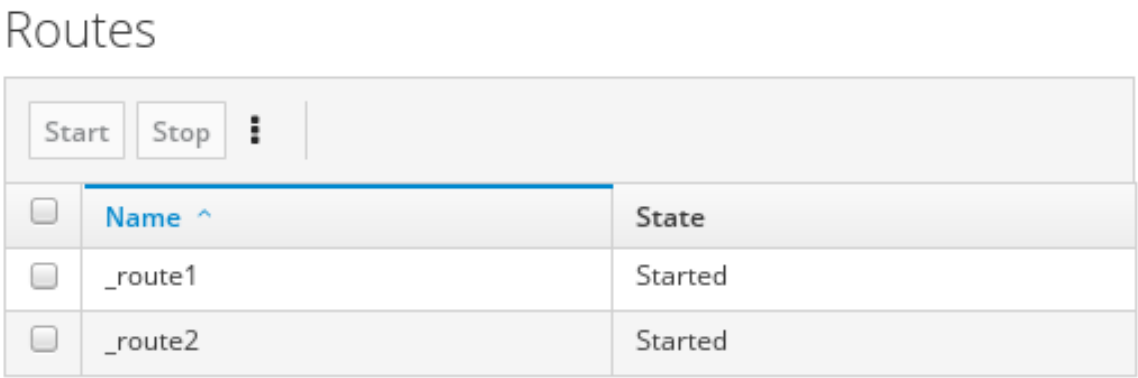
コンテキストを削除する場合、デプロイされたアプリケーションから削除します。

Camel アプリケーションの詳細を表示するには、以下を行います。

1. **Camel** タブのツリービューで、Camel アプリケーションをクリックします。
2. アプリケーションの属性と値のリストを表示するには、**Attributes** をクリックします。
3. アプリケーション属性をグラフィック表示するには、以下を行います。
   a. **Chart** をクリックします。
   b. **Edit** をクリックし、チャートに表示する属性を選択します。
4. inflight exchange および blocked exchange を表示するには、**Exchanges** をクリックします。
5. アプリケーションエンドポイントを表示するには、**Endpoints** をクリックします。リストは **URL**、**Route ID**、および **direction** で絞り込むことができます。
6. **Type Converters** をクリックし、Camel の組み込みタイプ変換メカニズムに関連する統計を表示、有効化、および無効化します。このメカニズムはメッセージ本文とメッセージヘッダーを別のタイプに変換するために使用されます。
7. **Operations** をクリックして、JMX 操作 (XML からのルートの追加または更新、クラスパスで利用できる Camel コンポーネントの検索など) を表示および実行します。

Camel ルートと対話するには、以下を行います。

1. **Camel** タブのツリービューでアプリケーションのルートフォルダーをクリックし、ルートのリストを表示します。

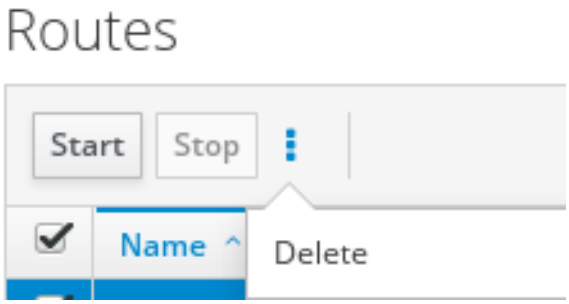


2. 1 つまたは複数のルートを開始、停止、または削除するには、以下を行います。

a. リストのルートの横にあるボックスにチェックマークを入れます。

b. **Start** または **Stop** をクリックします。

c. 最初にルートを停止してから削除する必要があります。停止したら楕円のアイコンをクリックし、ドロップダウンメニューで **Delete** を選択します。



注記

- ルートを削除する場合、デプロイされたアプリケーションから削除します。
- ツリービューで特定のルートを選択し、右上のメニューをクリックして開始、停止、または削除することもできます。

3. ルートのグラフィックな図を表示するには、**Route Diagram** をクリックします。

4. inflight exchange および blocked exchange を表示するには、**Exchanges** をクリックします。

5. エンドポイントを表示するには、**Endpoints** をクリックします。URL、Route ID、および方向でリストを絞り込むことができます。

6. **Type Converters** をクリックし、Camel の組み込みタイプ変換メカニズムに関連する統計を表示、有効化、および無効化します。このメカニズムはメッセージ本文とメッセージヘッダーを別のタイプに変換するために使用されます。

特定のルートと対話するには、以下を行います。

1. **Camel** タブのツリービューで、ルートを選択します。

2. ルート属性と値のリストを表示するには、**Attributes** をクリックします。

3. ルート属性をグラフィックに表示するには、**Chart** をクリックします。**Edit** をクリックするとチャートに表示する属性を選択することができます。

4. inflight exchange および blocked exchange を表示するには、**Exchanges** をクリックします。

5. **Operations** をクリックして JMX 操作 (ルートを XML としてダンプ、ルートの Camel ID 値の取得など) を表示および実行できます。

ルートを介してメッセージをトレースするには、以下を実行します。

1. **Camel** タブのツリービューで、ルートを選択します。

2. **Trace** を選択し、**Start tracing** をクリックします。

3. メッセージをルートに送信するには、以下を行います。

a. **Camel** タブのツリービューでコンテキストのエンドポイントフォルダーを開き、エンドポイントを選択します。

b. **Send** サブタブをクリックします。

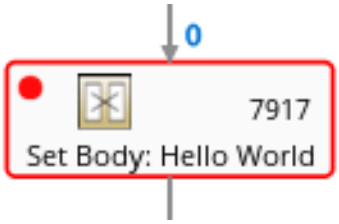
c. JSON または XML 形式のメッセージを設定します。

d. **Send** をクリックします。

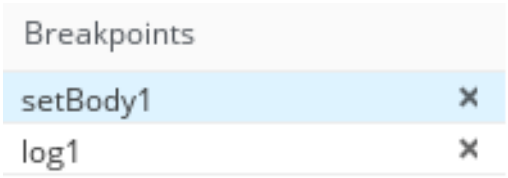
4. ルートの **Trace** タブに戻り、ルートを介したメッセージのフローを確認します。

ルートをデバッグするには、以下を実行します。

1. **Camel** タブのツリービューで、ルートを選択します。
2. **Debug** を選択し、**Start debugging** をクリックします。
3. ブレークポイントを追加するには、図のノードを選択し、**Add breakpoint** をクリックします。ノードに赤い点が表示されます。



ノードがブレークポイントのリストに追加されます。



4. 下矢印をクリックして次のノードに移動するか、**Play** ボタンをクリックしてルートの実行を再開します。
5. **Pause** ボタンをクリックして、ルートのすべてのスレッドを一時停止します。
6. 終了したら **Stop debugging** をクリックします。すべてのブレークポイントが消去されます。

4.4. JMX ドメインおよび MBEAN の表示および管理

JMX (Java Management Extensions) は、実行時にリソース (サービス、デバイス、およびアプリケーション) を動的に管理できる Java 技術です。リソースは MBean (Managed Bean) と呼ばれるオブジェクトで表現されます。リソースが作成、実装、またはインストールされると即座に管理することができます。

Fuse Console で JMX プラグインを使用すると、JMX ドメインと MBean を表示および管理できます。MBean 属性の表示、コマンドの実行、および MBean の統計を示すチャートの作成を行うことができます。

JMX タブは、フォルダーに整理されたアクティブな JMX ドメインと MBean のツリービューを提供します。詳細を確認し、MBean でコマンドを実行できます。

MBean 属性を表示および編集するには、以下を行います。

1. ツリービューで MBean を選択します。
2. **Attributes** タブをクリックします。
3. 属性をクリックしてその詳細を表示します。

操作を実行するには、以下を行います。

1. ツリービューで MBean を選択します。
2. **Operations** タブをクリックし、リストにある操作の 1 つを展開します。
3. **Execute** をクリックし、操作を実行します。

チャートを表示するには、以下を行います。

1. ツリービューで項目を選択します。
2. **Chart** タブをクリックします。

4.5. OSGI 環境の表示および管理 (KARAF スタンドアロン)

Apache Karaf スタンドアロンディストリビューションでは、Red Hat Fuse OSGi 環境を表示および管理できます。コンテナーバンドル、機能、および設定のほか、Java パッケージや OSGi サービスも表示および管理できます。

OSGi タブには、各コンテナーコンポーネントのオプションが含まれる複数のサブタブが含まれています。

Bundles
: インストールされたバンドルのリストです。バンドルのインストールおよびアンインストール、バンドルの開始および停止、およびバンドルプロパティーの編集を行うことができます。さらに、リストの絞り込みやリストとグリッドビューの切り替えを行うこともできます。

Features
: 使用できる機能のリスト。機能や機能リポジトリーをインストールおよびアンインストールでき、機能の詳細を表示できます。

Packages
: Java パッケージのリスト。パッケージバージョンと関連するバンドルを表示できます。

Services
: 実行中のサービスのリスト。サービス ID、関連するバンドル、およびオブジェクトクラスを表示できます。

Declarative Services
: 宣言的 OSGi サービスのリスト。サービスの状態を表示し、サービスの詳細を表示することができます。また、サービスをアクティベートおよび非アクティベートすることもできます。

Server
: 読み取り専用モードのローカルまたはリモートホストに関する詳細情報。

Framework
: コンテナー OSGi フレームワークの設定オプション。フレームワーク開始レベルと初期バンドル開始レベルを設定できます。

Configuration

設定オブジェクトのリスト。各オブジェクトの状態を表示し、オブジェクトの詳細を表示または編集できます。また、新しい設定オブジェクトを作成することもできます。

4.6. スレッド状態の表示および監視

スレッドの状態を表示し、監視するには、以下を行います。

1. **Runtime** タブをクリックし、**Threads** サブタブをクリックします。**Threads** ページには、アクティブなスレッドと各スレッドのスタックトレースの詳細が表示されます。デフォルトでは、スレッドリストにはすべてのスレッドが ID 値が大きい順に表示されます。
2. ID 値が小さい順に表示するには、**ID** 列ラベルをクリックします。
3. 任意で、スレッドの状態 (例: **Blocked**) やスレッド名でリストを絞り込むことができます。
4. ロッククラス名やスレッドのフルスタックトレースなど、特定スレッドの詳細情報を表示するには、**Actions** 列で **More** をクリックします。

4.7. ログエントリーの表示

Logs タブで Red Hat Fuse のログエントリーを表示できます。 **Logs** タブは、Java アプリケーションに Log MBean がある場合に使用できます (Karaf の Fuse スタンドアロンおよび JBoss EAP の Fuse スタンドアロン)。

ログのリストを絞り込んで特定のログタイプを表示し、各ログエントリーの詳細情報を表示することができます。

Logs タブには以下のセクションが含まれます。

Action Bar

テキスト文字列またはログレベルに応じてログエントリーを絞り込むオプション。

Log Entries

ログエントリーのリストビュー。デフォルトでは、リストのログエントリーは昇順で表示されます。デフォルトの並び替えは **User → Preferences → Server Logs** で変更できます。ログエントリーのリンクをクリックし、バンドル名、スレッド、フルメッセージテキストなどのログエントリーの詳細を表示します。

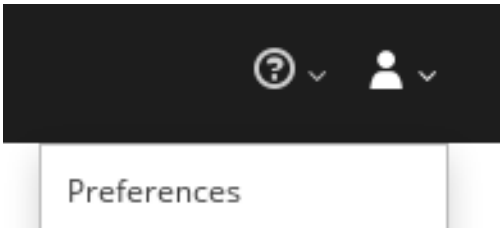
4.7.1. Fuse Console ログ属性の設定

ログメッセージの保存に関し、以下の Fuse Console ブラウザー設定をカスタマイズすることができます。

- Fuse Console に保持するログステートメントの数 (デフォルトは 100)。
- グローバルログレベル: **INFO** (デフォルト)、OFF、ERROR、WARN、および DEBUG
- hawtio-oauth や hawtio-core-utils など、含める子レベルの Howtio メッセージ。

デフォルトを変更するには、以下を行います。

1. Fuse Console の右上隅にあるユーザーアイコンをクリックして、ドロップダウンメニューの **Preferences** をクリックします。



2. オプションを編集して **Close** をクリックします。

Console Logs 設定をデフォルト値にリセットするには、**Reset → Reset settings** とクリックします。

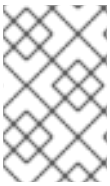
4.8. FUSE CONSOLE のブランディングの変更

Fuse Console のブランディング名およびイメージを変更するには、以下のファイルを変更します。

- Fuse Console の war ファイル (**karaf-install-dir/system/io/hawt/hawtio-war/version/hawtio-war-version.war**) の **hawtconfig.json** ファイルを編集します。

```
{
  "branding": {
    "appName": "Hawtio Management Console",
    "appLogoUrl": "img/hawtio-logo.svg",
    "companyLogoUrl": "img/hawtio-logo.svg"
  },
  "about": {
    "title": "Hawtio Management Console",
    "productInfo": [],
    "additionalInfo": "",
    "copyright": "",
    "imgSrc": "img/hawtio-logo.svg"
  },
  "disabledRoutes": []
}
```

- war ファイルのスタイルシート (**.css**) ファイルで Fuse Console UI のその他の内容を変更します。



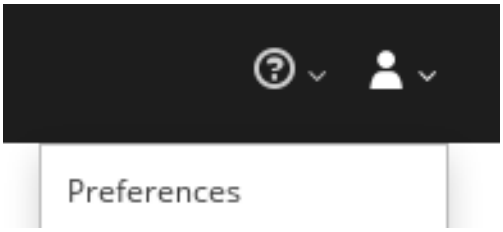
注記

Web ブラウザーで Fuse Console をすでに実行した場合、ブランディングはブラウザーのローカルストレージに保存されます。新しいブランディング設定を使用するには、ブラウザーのローカルストレージをクリアする必要があります。

4.9. FUSE CONSOLE でデータが正しく表示されるよう確認

Fuse Console のキューおよびコネクションの表示で、不足しているキューやコネクションがあったり、一貫性のないアイコンが表示される場合、Jolokia が応答でマーシャルするアレイの要素の最大数を指定する、Jolokia コレクションサイズパラメーターを調節します。

1. Fuse Console の右上隅にあるユーザーアイコンをクリックして、**Preferences** をクリックします。



2. **Maximum collection size** オプションの値を大きくします (デフォルトは 50,000)。
3. **Close** をクリックします。

第5章 PROMETHEUS を使用した FUSE アプリケーションの監視

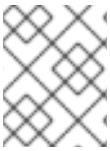
Prometheus を使用して Fuse on OpenShift のデータを監視および保存するには、エンドポイントを Fuse アプリケーションのデータとともに Prometheus 形式で公開します。Prometheus はデータを保存するため、Grafana などのグラフィカルツールを使用してデータを視覚化し、クエリーを実行することができます。

Prometheus を使用して、Minishift や Red Hat Container Development Kit などのオンプレミス OpenShift クラスターや単一ノードのクラスターで実行している Fuse アプリケーションを監視できます。

Red Hat Fuse on OpenShift でインストールおよび開発を行うための情報は、Fuse on OpenShift Guide を参照してください。

以下のセクションでは、Prometheus へのアクセス方法を説明します。

- 「Prometheus の設定」
- 「Prometheus の設定」



注記

Prometheus を使用して OpenShift Online 上の Fuse アプリケーションを監視することはサポートされません。

5.1. PROMETHEUS の設定

Prometheus を設定するには、クラスターに Prometheus Operator のカスタムリソース定義をインストールし、Prometheus を Fuse アプリケーションが含まれる OpenShift プロジェクトに追加します。

前提条件

- OpenShift クラスターへのシステム管理者アクセス権限が必要です。
- 『Fuse on OpenShift Guide』の説明にしたがって、Fuse on OpenShift イメージおよびテンプレートをインストールして OpenShift クラスターが準備されている必要があります。
- クラスターで OpenShift プロジェクトを作成し、Fuse アプリケーションをそのプロジェクトに追加されている必要があります。

手順

1. 管理者権限で OpenShift にログインします。

```
$ oc login -u system:admin
```

2. Prometheus Operator の実行に必要なカスタムリソース定義をインストールします。ここで、**{$templates-base-url}** は Fuse on OpenShift テンプレートファイルの場所になります。

```
$ oc create -f {$templates-base-url}\fuse-prometheus-crd.yml
```

Prometheus Operator がクラスターのすべての namespace で利用できるようになります。

3. 以下のコマンド構文を使用して、Prometheus Operator を namespace にインストールします。

```
$ oc process -f {$templates-base-url}/fuse-prometheus-operator.yml -p NAMESPACE=
<YOUR NAMESPACE> | oc create -f -
```

たとえば、**myproject** という名前のプロジェクト (namespace) には、以下のコマンドを使用します。

```
oc process -f {$templates-base-url}/fuse-prometheus-operator.yml -p
NAMESPACE=myproject | oc create -f -
```

注記

Prometheus Operator を namespace に初めてインストールする場合、Prometheus リソース Pod が起動するまで数分かかる場合があります。その後、Prometheus Operator をクラスターの他の namespace にインストールすると、Prometheus リソース Pod の起動ははるかに速くなります。

4. 以下のコマンド構文を使用して、Prometheus Operator を指示し、プロジェクトの Fuse アプリケーションを監視します。

```
$ oc process -f {$templates-base-url}/fuse-servicemonitor.yml -p NAMESPACE=<YOUR
NAMESPACE> FUSE_SERVICE_NAME=<YOUR FUSE SERVICE> | oc apply -f -
```

たとえば、**myfuseapp** という名前の Fuse アプリケーションが含まれる **myproject** という名前の OpenShift プロジェクト (namespace) には、次のコマンドを使用します。

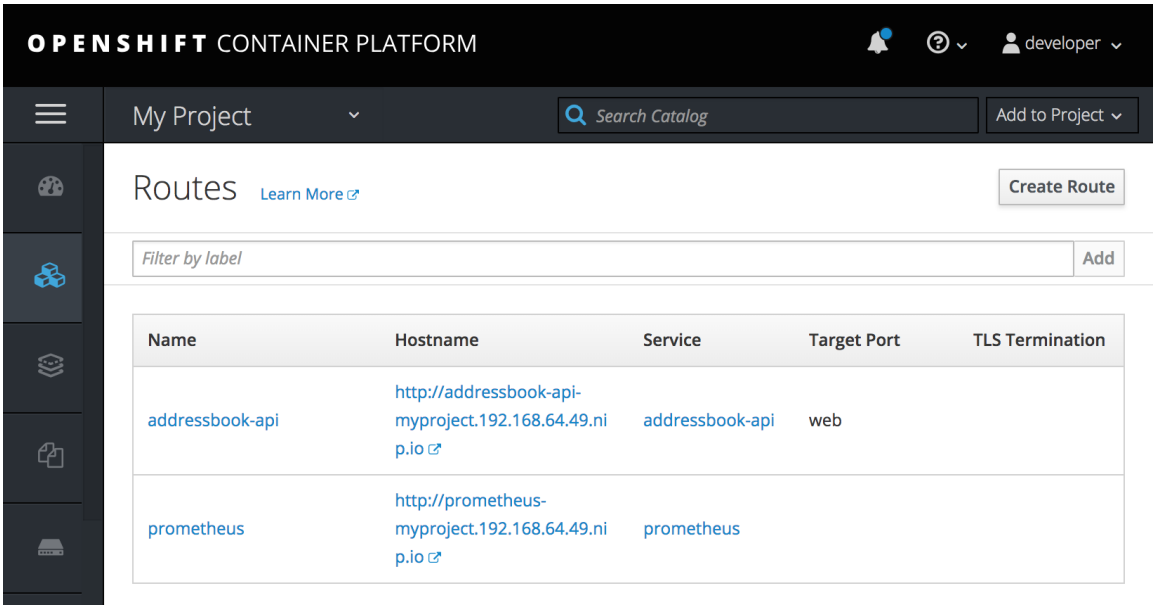
```
oc process -f {$templates-base-url}/fuse-servicemonitor.yml -p NAMESPACE=myproject
FUSE_SERVICE_NAME=myfuseapp | oc apply -f -
```

5. Prometheus ダッシュボードを開くには、以下を実行します。

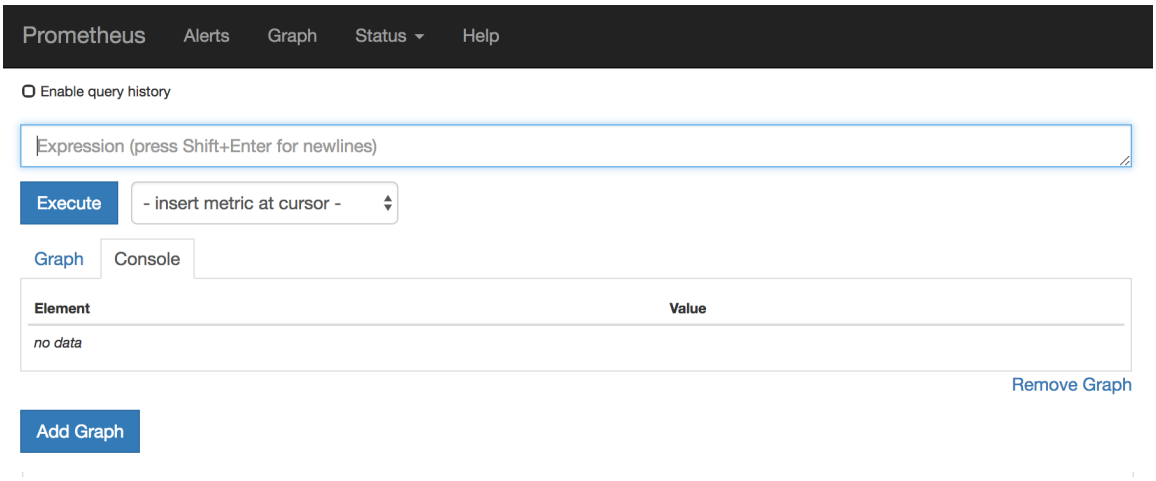
   a. OpenShift コンソールにログインします。

   b. Prometheus を追加したプロジェクトを開きます。

   c. 左側のペインで、**Applications → Routes** と選択します。



d. Prometheus Hostname URL をクリックし、新しいブラウザータブまたはウインドウで Prometheus ダッシュボードを開きます。



e. Prometheus を初めて使用する場合は、「https://prometheus.io/docs/prometheus/latest/getting_started/」を参照してください。

5.2. PROMETHEUS の設定

以下のセクションでは、Prometheus の設定方法を説明します。

- 「OpenShift 環境変数の設定」
- 「Prometheus が監視および収集するメトリクスの制御」
- 「アラートの生成」
- 「Fuse Online インテグレーションアプリケーションを監視するよう外部 Prometheus インスタンスを設定」

5.2.1. OpenShift 環境変数の設定

アプリケーションの Prometheus インスタンスを設定する場合は、以下の環境変数を OpenShift に設定できます。

表5.1 Prometheus 環境変数

環境変数	説明	デフォルト
AB_PROMETHEUS_HOST	バインドするホストアドレス。	**0.0.0.0**
AB_PROMETHEUS_OFF	設定されている場合、Prometheus のアクティベートを無効にします (空の値をエコーします)。	Prometheus が有効。
AB_PROMETHEUS_PORT	使用するポート。	**9779**
AB_JMX_EXPORTER_CONFIG	ファイル (パスを含む) を Prometheus 設定ファイルとして使用します。	Camel メトリクスが含まれる /opt/prometheus/prometheus-config.yml ファイル。
AB_JMX_EXPORTER_OPTS	JMX エクスポーター設定に追加する追加オプション。	該当なし

Pod の環境変数の設定に関する詳細は、**『OpenShift 開発者ガイド』** (https://access.redhat.com/documentation/ja-jp/openshift_container_platform/3.11/html/developer_guide/index) を参照してください。

5.2.2. Prometheus が監視および収集するメトリクスの制御

デフォルトでは、Prometheus は Camel によって公開される可能性のあるすべてのメトリクスが含まれる設定ファイル (**https://raw.githubusercontent.com/jboss-fuse/application-templates/master/prometheus/prometheus-config.yml**) を使用します。

Prometheus が監視および収集するアプリケーション内にカスタムメトリクスがある場合 (アプリケーションプロセスを実行するオーダーの数など)、独自の設定ファイルを使用できます。識別できるメトリクスは、JMX で提供されるメトリクスに限定されるため注意してください。

手順

カスタム設定ファイルを使用して、デフォルトの Prometheus 設定では対応されない JMX Bean を公開するには、以下の手順に従います。

1. カスタム Prometheus 設定ファイルを作成します。デフォルトファイルのコンテンツ (**prometheus-config.yml**https://raw.githubusercontent.com/jboss-fuse/application-templates/master/prometheus/prometheus-config.yml) を形式の目安として使用できます。カスタム設定ファイルには任意の名前を使用できます (例: **my-prometheus-config.yml** など)。

2. prometheus 設定ファイル (例: **my-prometheus-config.yml**) をアプリケーションの **src/main/fabric8-includes** ディレクトリーに追加します。

3. アプリケーション内に **src/main/fabric8/deployment.xml** ファイルを作成し、設定ファイルに設定された値で **AB_JMX_EXPORTER_CONFIG** 環境変数のエントリーを追加します。以下に例を示します。

```
spec:
  template:
```

```
spec:
  containers:
    -
      resources:
        requests:
          cpu: "0.2"
        limits:
          cpu: "1.0"
      env:
      - name: SPRING_APPLICATION_JSON
        value: '{"server":{"tomcat":{"max-threads":1}}}'
      - name: AB_JMX_EXPORTER_CONFIG
        value: "my-prometheus-config.yml"
```

この環境変数は、Pod レベルでアプリケーションに適用されます。

4. アプリケーションの再構築およびデプロイ

5.2.3. アラートの生成

OpenShift に Prometheus を使用してアラートを生成する例は、Red Hat Cloud Forms の『**モニタリング、アラート、およびレポーティング**』を参照してください。

https://access.redhat.com/documentation/en-us/red_hat_cloudforms/4.6/html/monitoring_alerts_and_reporting/integrating_prometheus_alerts

5.2.4. Fuse Online インテグレーションアプリケーションを監視するよう外部 Prometheus インスタンスを設定

OpenShift Container Platform に Fuse Online をインストールする場合、デフォルトで **syndesis-prometheus** が含まれます。Fuse Online を OpenShift Container Platform にインストールする方法の詳細は、『Integrating Applications with Fuse Online』を参照してください。

外部の Prometheus インスタンスがすでに存在する場合、その外部のインスタンスを設定して、OpenShift Container Platform にデプロイされた Fuse Online インテグレーションアプリケーションを監視することもできます。

手順は、Prometheus Operator を使用して外部の Prometheus インスタンスをインストールしたかどうかによって異なります。

5.2.4.1. Prometheus インスタンスの設定 (Prometheus Operator を使用)

Prometheus Operator を使用して Prometheus インスタンスをインストールした場合、外部 Prometheus 設定を更新して Fuse Online インテグレーションを監視するのに、サービスモニターの追加および編集が必要になります。

前提条件

「Prometheus の設定」 の説明どおりに Prometheus がインストールされている必要があります。

手順

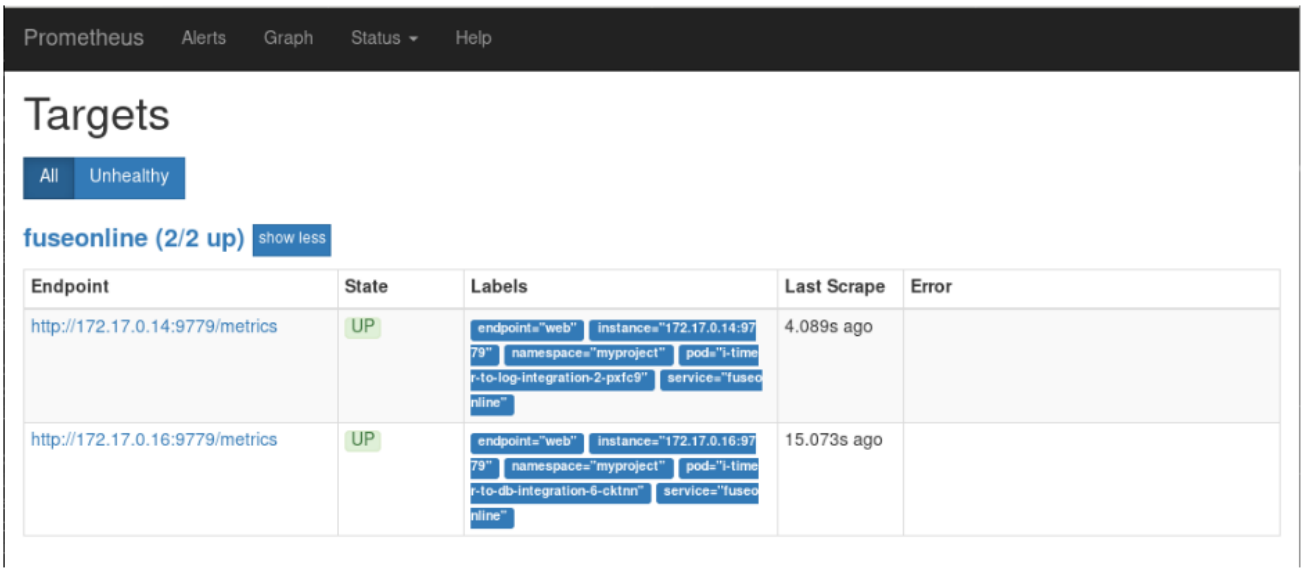
1. ターミナルウインドウで、サービスモニターを Prometheus がインストールされた namespace (プロジェクト) に追加します。

```
oc process -f  {$templates-base-url}/fuse-servicemonitor.yml -p NAMESPACE=<YOUR-NAMESPACE> -p FUSE_SERVICE_NAME=fuseonline | oc create -f -
```

2. OpenShift コンソールでプロジェクトを開き、**Applications → Services** と選択します。
3. **fuseonline** サービスをクリックし、**Actions → Edit YAML** と選択します。
4. エディターで、**app: fuseonline** を **syndesis.io/type: integration** に置き換えて YAML ファイルのセレクターセクションを変更します。
5. **Save** をクリックします。

Prometheus Operator が設定を更新し、Fuse Online インテグレーションをすべて監視します。

これで、Fuse Online インテグレーションを Prometheus インスタンスで表示できるようになります。以下の例は、2 つのインテグレーションを監視する Prometheus インスタンスを示しています。



5.2.4.2. Prometheus インスタンスの設定 (Prometheus Operator を不使用)

Prometheus Operator を使用せずに Prometheus インスタンスをインストールした場合、設定を更新して Fuse Online インテグレーションを監視するのに、Prometheus 設定ファイルを編集し、Prometheus Pod 設定を更新する必要があります。

前提条件

Prometheus 設定ファイル (**prometheus-config.yml**) への書き込み権限が必要です。OpenShift web コンソールでは、設定ファイルは **Resources → Config Maps** にあります。

手順

1. Prometheus 設定ファイル (**prometheus-config.yml**) を以下のように編集します。
   a. スクレープ間隔を 5 秒に設定します。

```
global:
      scrape_interval:     5s
      evaluation_interval: 5s
```

b. 通常は **syndesis** namespace である **${OPENSHIFT_PROJECT}** で Pod をスクレープするよう設定する Kubernetes サービス検出設定とともに **integration-pod** という名前のスクレープ設定ジョブを追加します。

```
- job_name: integration-pods

    kubernetes_sd_configs:
    - role: pod
      namespaces:
        names:
          - ${OPENSHIFT_PROJECT}
```

c. 以下のような **relabel_configs** セクションを追加します。

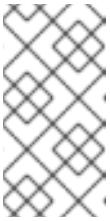
- **prometheus.io/scrape** ラベルが **true** に設定されたインテグレーション Pod のみをスクレープする。
- **prometheus.io/path** および **prometheus.io/port** ラベルから値を使用して、インテグレーション Pod で JMX エクスポーターをスクレープするのに使用される **metrics_path** および **address** ラベルを設定する。
- Pod ラベルおよびアノテーションを Prometheus レベルとして追加する。
- **kubernetes_namespace** および **kubernetes_pod_name** ラベルを作成する。

以下は、**relabel_configs** セクションの例です。

```
relabel_configs:
        - source_labels:
[__meta_kubernetes_pod_annotation_prometheus_io_scrape]
          action: keep
          regex: true
        - source_labels: [__meta_kubernetes_pod_annotation_prometheus_io_path]
          action: replace
          target_label: __metrics_path__
          regex: (.+)
        - source_labels: [__address__,
__meta_kubernetes_pod_annotation_prometheus_io_port]
          action: replace
          regex: ([^:]+)(?::\d+)?;(\d+)
          replacement: $1:$2
          target_label: __address__
        - action: labelmap
          regex: __meta_kubernetes_pod_label_(.+)
        - action: labelmap
          regex: __meta_kubernetes_pod_annotation_(syndesis.+)
        - source_labels: [__meta_kubernetes_namespace]
          action: replace
          target_label: kubernetes_namespace
        - source_labels: [__meta_kubernetes_pod_name]
          action: replace
          target_label: kubernetes_pod_name
```

d. Fuse Online インテグレーションは、JVM、Camel、および CXF から多数のメトリクスを公開します。メトリクスに必要なストレージの量を削減するには、以下の **metric_relabel_configs** セクションを追加し、Fuse Online コンソールに表示されないメトリクスを除外します。

```
 metric_relabel_configs:
        - source_labels: [__name__]
          regex: jmx_(.+)
          action: drop
        - source_labels: [__name__]
          regex: jvm_(.+)
          action: drop
        - source_labels: [__name__]
          regex: process_(.+)
          action: drop
        - source_labels: [type, __name__]
          separator: ':'
          regex: context:
(org_apache_camel_ExchangesTotal|org_apache_camel_ExchangesFailed|io_syndesis_ca
mel_StartTimestamp|io_syndesis_camel_LastExchangeCompletedTimestamp|io_syndesis_
camel_LastExchangeFailureTimestamp)
          action: keep
```



注記

最後の設定行は、Fuse Online web コンソールに表示される統計に不可欠な Prometheus メトリクスストアに追加するメトリクスを明示的にリストしています。Prometheus インスタンスは、他のメトリクスが除外された場合にこれらのメトリクスの収集を明示的に許可する**必要があります**。

2. 以下のように、Prometheus Pod 設定が 30 日分のメトリクスを保存するよう更新します。

```
args:
        - '--config.file=/etc/prometheus/prometheus.yml'
        - '--storage.tsdb.retention=30d'
```