JobCenter

R16.3

スタンダードモード用コンテナガイド

本マニュアルでは、製品名およびサービス名を次のように略称表記しています。

略称	製品名・サービス名
Office	Microsoft Office
Excel	Microsoft Excel
Azure	Microsoft Azure
Internet Explorer	Internet Explorer 11
Firefox	Mozilla Firefox
AWS	Amazon Web Services
EC2	Amazon Elastic Compute Cloud
EBS	Amazon Elastic Block Store
S3	Amazon Simple Storage Service
ELB	Elastic Load Balancing
CloudFormation, CF	AWS CloudFormation
CloudWatch, CW	Amazon CloudWatch
RDS	Amazon Relational Database Service
Glue	AWS Glue
Lambda	AWS Lambda
EKS	Amazon Elastic Kubernetes Service
ECS	Amazon Elastic Container Service
STS	AWS Security Token Service
CloudWatch Logs	Amazon CloudWatch Logs
SNS	Amazon Simple Notification Service

輸出する際の注意事項

本製品（ソフトウエア）は、外国為替令に定める提供を規制される技術に該当いたしますので、日本国外へ持ち出す際には日本国政府の役務取引許可申請等必要な手続きをお取りください。許可手続き等にあたり特別な資料等が必要な場合には、お買い上げの販売店またはお近くの当社営業拠点にご相談ください。

はじめに

本書は、コンテナ環境におけるJobCenterの構築、運用方法について説明します。

本書の内容は将来、予告なしに変更する場合があります。あらかじめご了承下さい。

1. マニュアルの読み方

■本バージョンにおける新規機能や変更事項を理解したい場合

→ <スタンダードモード用リリースメモ>を参照してください。

■JobCenterを新規にインストール、またはバージョンアップされる場合

→ <セットアップガイド>を参照してください。

■JobCenterを初めて利用される場合

→ <セットアップガイド>を参照してください。

■JobCenterの基本的な操作方法を理解したい場合

→ <スタンダードモード用基本操作ガイド>を参照してください。

■JobCenterの動作を制御する設定やネットワーク関連の設定を理解したい場合

→　ジョブ管理マネージャ(MG)機能の設定については<スタンダードモード用環境構築ガイド>を参照してください。

→　ジョブ実行エージェント(AG)機能の設定については<スタンダードモード用ジョブ実行エージェント構築ガイド>を参照してください。

■JobCenterの操作をコマンドラインから行う場合

→ <スタンダードモード用コマンドリファレンス>を参照してください。

■JobCenterのイベントログ出力方法など監視に関連した機能を理解したい場合

→ <スタンダードモード用環境構築ガイド>を参照してください。

■運用中のJobCenterを新環境に移行する場合

→ <スタンダードモード用移行ガイド>を参照してください。

■JobCenterのクラスタ環境を構築したい場合

→ <スタンダードモード用クラスタ機能利用の手引き>を参照してください。

■ブラウザを用いたJobCenterの監視やWebAPIでの制御などWebに関連した機能を理解したい場合

→ <スタンダードモード用Web機能利用の手引き>を参照してください。

■その他機能についてお知りになりたい場合

→ 関連マニュアルの内容をお読みいただき、目的のマニュアルを参照してください。

2. 凡例

本書内での凡例を紹介します。

	気をつけて読んでいただきたい内容です。
	本文中の補足説明
	本文中のヒントとなる説明
注	本文中につけた注の説明
__	UNIX版のインストール画面の説明では、__部分(下線部分)はキーボードからの入力を示します。

3. 関連マニュアル

JobCenter に関するマニュアルです。JobCenter メディア内に格納されています。

最新のマニュアルは、JobCenter 製品サイトのダウンロードのページを参照してください。

https://jpn.nec.com/websam/jobcenter/download.html

【スタンダードモードのマニュアル】

資料名	概要
JobCenter セットアップガイド	JobCenterを新規にインストール、またはバージョンアップする場合の方法について説明しています。
JobCenter 基本操作ガイド	JobCenterの基本機能、操作方法について説明しています。
JobCenter 環境構築ガイド	JobCenterを利用するために必要なジョブ実行マネージャ環境の構築方法や設定方法の詳細、マネージャ環境の運用に役立つ機能について説明しています。
JobCenter ジョブ実行エージェント構築ガイド	JobCenterを利用するために必要なジョブ実行エージェント環境の構築方法や設定方法の詳細について説明しています。
JobCenter コマンドリファレンス	GUIと同様にジョブネットワークの投入、実行状況の参照などをコマンドラインから行うために、JobCenterで用意されているコマンドについて説明しています。
JobCenter クラスタ機能利用の手引き	クラスタシステムでJobCenterを操作するための連携方法について説明しています。
JobCenter Web機能利用の手引き	Webブラウザ上でジョブ監視を行うことができるWebコンソール機能、ジョブネットワークやトラッカ等の情報を参照、制御をHTTPプロトコルで行えるWebAPI機能について説明しています。
JobCenter 移行ガイド	運用中のJobCenterを別の新環境に移行する手順について横断的に説明しています。
JobCenter R16.3 リリースメモ	バージョン固有の情報を記載しています。

【クラシックモードのマニュアル】

資料名	概要
JobCenter インストールガイド	JobCenterを新規にインストール、またはバージョンアップする場合の方法について説明しています。
JobCenter クイックスタート編	初めてJobCenterをお使いになる方を対象に、JobCenterの基本的な機能と一通りの操作を説明しています。
JobCenter 基本操作ガイド	JobCenterの基本機能、操作方法について説明しています。
JobCenter 環境構築ガイド	JobCenterを利用するために必要な環境の構築、環境の移行や他製品との連携などの各種設定方法について説明しています。
JobCenter NQS機能利用の手引き	JobCenterの基盤であるNQSの機能をJobCenterから利用する方法について説明しています。
JobCenter コマンドリファレンス	GUIと同様にジョブネットワークの投入、実行状況の参照などをコマンドラインから行うために、JobCenterで用意されているコマンドについて説明しています。
JobCenter クラスタ機能利用の手引き	クラスタシステムでJobCenterを操作するための連携方法について説明しています。
JobCenter SAP機能利用の手引き	JobCenterをSAPと連携させるための方法について説明しています。
JobCenter WebOTX Batch Server連携機能利用の手引き	JobCenterをWebOTX Batch Serverと連携させるための方法について説明しています。

資料名	概要
JobCenter Web機能利用の手引き	Webブラウザ上でジョブ監視を行うことができるWebコンソール機能、ジョブネットワークやトラッカ等の情報を参照、制御をHTTPプロトコルで行えるWebAPI機能について説明しています。CL/Webについては以下のR16.2のWeb機能利用の手引きを参照してください。 https://jpn.nec.com/websam/jobcenter/download/manual/16_2/JB_CLS_WEB.pdf
JobCenter　クラスタ環境でのバージョンアップ・パッチ適用ガイド	クラスタ環境で運用しているJobCenterのアップデート、パッチ適用手順を説明しています。
JobCenter 運用・構築ガイド	JobCenterの設計、構築、開発、運用について横断的に説明しています。
JobCenter 移行ガイド	運用中のJobCenterを別の新環境に移行する手順について横断的に説明しています。
JobCenter コンテナガイド	JobCenterをコンテナ環境で構築・運用する方法について説明しています。
JobCenter R16.3 リリースメモ	バージョン固有の情報を記載しています。

【共通のマニュアル】

資料名	概要
JobCenter 操作・実行ログ機能利用の手引き	JobCenter　CL/Winからの操作ログ、ジョブネットワーク実行ログ取得機能および設定方法について説明しています。
JobCenter Helper機能利用の手引き	Excelを用いたJobCenterの効率的な運用をサポートするJobCenter　Definition　Helper　(定義情報のメンテナンス)、JobCenter　Report　Helper　(帳票作成)、JobCenter　Analysis　Helper　(性能分析)の3つの機能について説明しています。
JobCenter テキスト定義機能の利用手引き	JobCenterの定義情報をテキストファイルで定義する方法について説明しています。
JobCenter 拡張カスタムジョブ部品利用の手引き	拡張カスタムジョブとして提供される各部品の利用方法について説明しています。

4. 改版履歴

版数	変更日付	項目	形式	変更内容
1	2024/04/19	新規作成	ー	第1版

目次

表の一覧

1. 用語集

本マニュアルで使用されるコンテナに関連する用語を次に説明します。

用語	説明
MG	JobCenter MGの略称です。
AG	JobCenter AGの略称です。
MGコンテナ	JobCenter MGのアプリケーションのみが存在するコンテナを指します。
コンテナMG	コンテナ上で動作しているJobCenter MGを指します。
Docker	イメージのビルドやコンテナの作成など、コンテナ型仮想化を実行するためのコンテナエンジンを指します。
Dockerホスト	Dockerを動作させているホストマシンを指します。
Dockerレジストリ	Dockerイメージを格納しておく場所を指します。
Docker環境	Dockerを用いてJobCenterを運用するために必要な、複数のDockerホストやDockerレジストリを含めた総称を指します。
スプールディレクトリ	MGの定義データやジョブの実行結果などが格納されている、/usr/spool/nqsディレクトリのことを指します。
AWS	Amazon Web Servicesの略称です。
CR	KubernetesのリソースであるCustomResourceのことを指します。

2. コンテナ環境での運用

本章ではコンテナソフトウェアを用いてJobCenterコンテナ環境を運用する方法について説明します。

JobCenterがサポートするコンテナソフトウェアはDockerおよびPodmanです。詳細は<スタンダードモード用リリースメモ>の「3.1.6 対応コンテナ詳細」を参照してください。

本章ではDockerでの説明および使用方法を記載しています。Podmanを使用する場合は適宜読み替えてください。

podmanコマンドはdockerコマンドと互換性があります。podmanコマンドを使用する場合は、本章で説明しているdockerコマンドの「docker」を「podman」に読み替えてください。

2.1. 概要

コンテナとは、アプリケーションを実行環境ごとパッケージングしたものです。コンテナ単体で1つのサーバのように使用できます。コンテナを用いてアプリケーションを実行する方式を、コンテナ型仮想化と呼びます。

コンテナ型仮想化は、ホストOS型やハイパーバイザー型などの仮想化方式と比べて、次のような特徴があります。

■低リソース、高速

コンテナ型仮想化においては、ゲストOSを起動しません。コンテナはホストOS上では1つのプロセスとして実行されます。そのため、仮想マシンのサーバよりもオーバヘッドが少なく、軽量で高速に動作します。

■環境の固定/分離

コンテナではアプリケーションと実行環境がセットなため、アプリケーションごとにライブラリのバージョンなどの実行環境を固定できます。各コンテナはホストOSに論理的な区画を作成するため、ホストやほかのコンテナに影響を与えない、隔離された環境を構築できます。

JobCenterはコンテナ型の仮想化技術の1つであるDockerに対応しています。Dockerのアプリケーションがインストールされている環境であれば、JobCenterの提供しているDockerfileを利用することで、コンテナを利用したJobCenterの運用を始められます。

2.1.1. Dockerの概要

Dockerには次のような特徴があります。

■アプリケーションの実行に必要な環境をDockerイメージにまとめ、そのDockerイメージからコンテナの作成を行います。Dockerイメージの配布/共有を行うことで、Docker環境があるホスト間における環境移行が容易に行えます。

■Dockerfileと呼ばれる環境などの設定が記載されたスクリプトファイルから、Dockerイメージを作成します。同一のDockerfileがあれば、同一のDockerイメージを作成できます。

Docker環境の構成図は図2.1「Docker環境」の通りです。各用語の意味については、1章 「用語集」 を参照してください。

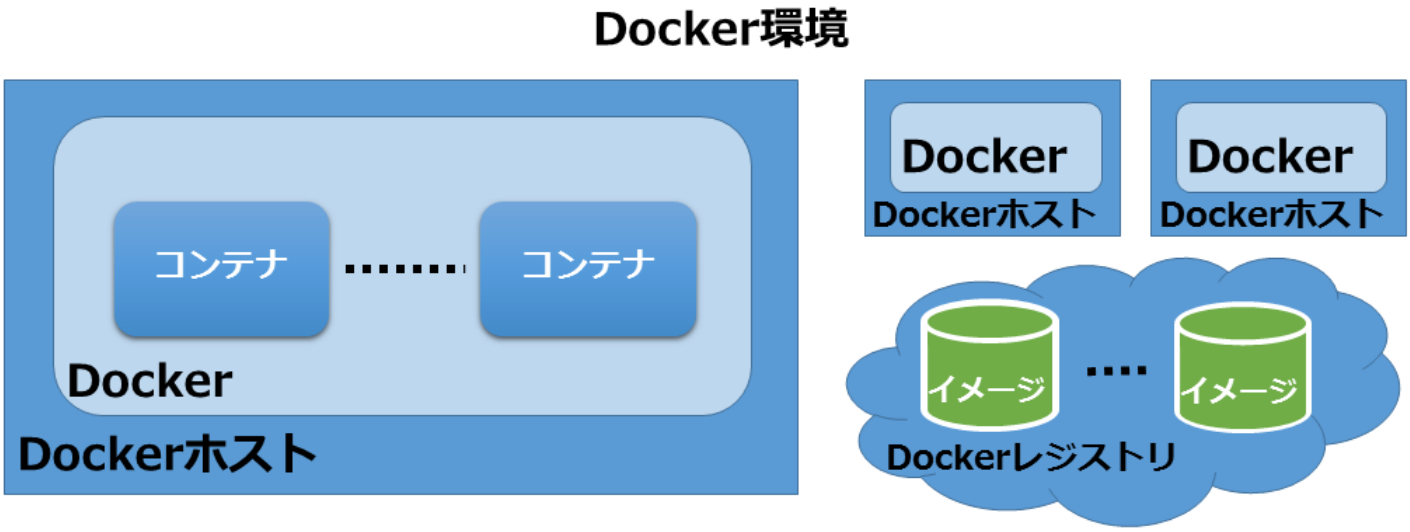

図2.1 Docker環境

2.1.2. イメージのライフサイクル

イメージのライフサイクルについて説明します。

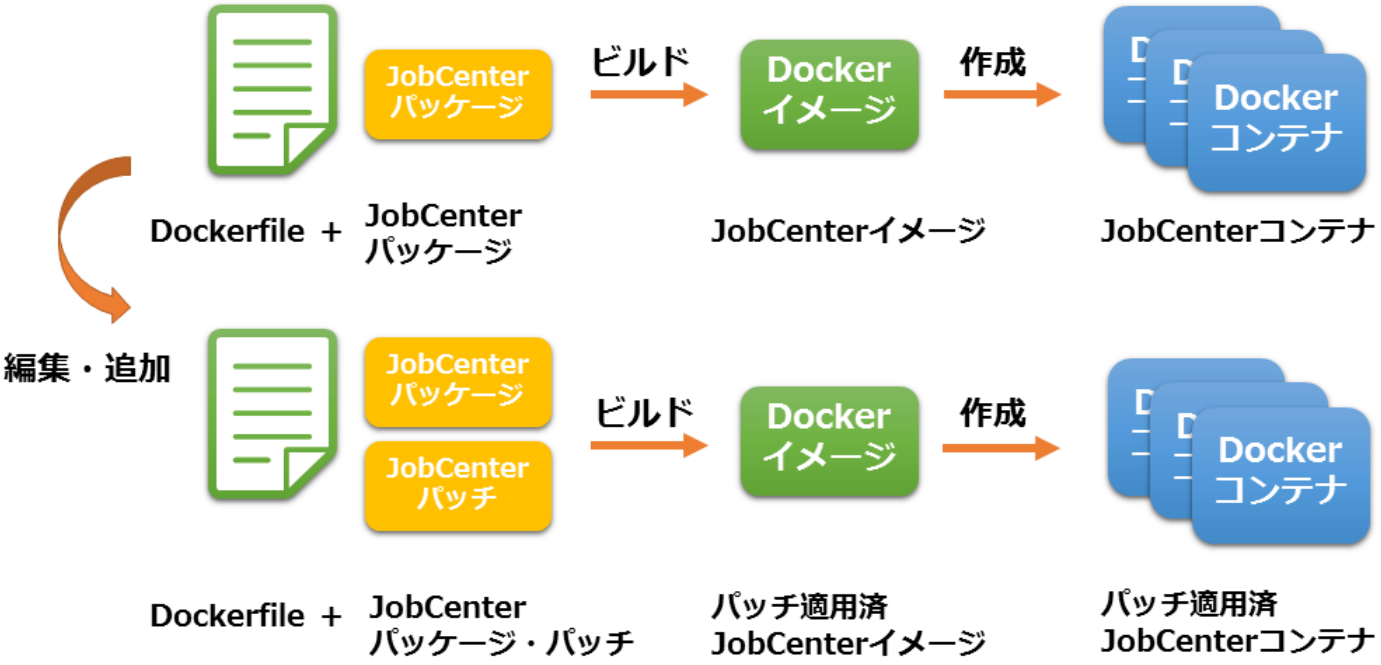

図2.2 イメージのライフサイクル

■イメージのビルド

DockerfileおよびJobCenterのパッケージからDockerイメージをビルドします。同一のDockerfile、JobCenterパッケージからは同一のイメージが作成されます。

詳細は「2.3 イメージの作成」を参照してください。

■コンテナの作成

Dockerイメージからコンテナの作成を行います。1つのイメージからコンテナは複数作成できます。作成時にオプションを指定することで、コンテナの設定ができます。

詳細は「2.4.4 コンテナの作成/起動」を参照してください。

■Dockerfileの編集

Dockerfileに記載したアプリケーションや環境の設定は、Dockerイメージ内で固定されます。環境設定の変更や、使用しているJobCenterパッケージのバージョンアップなどを行いたい場合は、Dockerfileを編集して再ビルドを行い、新しいイメージを作成します。

2.1.3. コンテナのライフサイクル

コンテナのライフサイクルについて説明します。

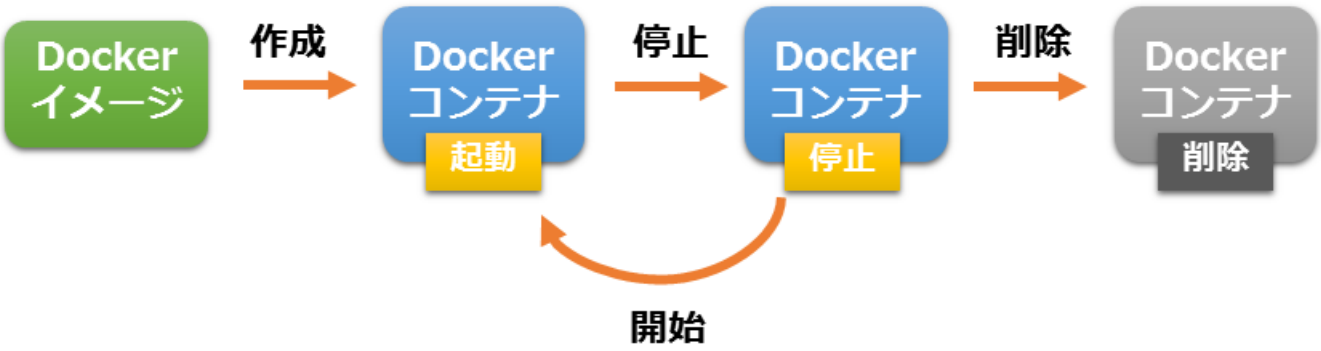

図2.3 コンテナのライフサイクル

■コンテナの作成

Dockerイメージからコンテナの作成を行います。作成したコンテナは起動状態になります。

詳細は「2.4.4 コンテナの作成/起動」を参照してください。

■コンテナの停止

起動中のコンテナを停止状態にします。

詳細は「2.5.2 コンテナの管理」を参照してください。

■コンテナの開始

停止中のコンテナを起動状態にします。

詳細は「2.5.2 コンテナの管理」を参照してください。

■コンテナの削除

停止中のコンテナを削除します。

詳細は「2.5.2 コンテナの管理」を参照してください。

2.1.4. JobCenterのコンテナ

JobCenterは、コンテナを作成するためのDockerfileおよびスクリプトファイルを提供しています。それらのファイルとJobCenterのパッケージを用いることで、JobCenterのアプリケーションが存在するコンテナを作成できます。

JobCenterにより、以下のコンテナを作成することができます。

■アプリケーションコンテナ

コンテナ内に1つのアプリケーションのみが存在するコンテナです。Dockerfileおよびイメージを共有することで、同一の環境をデプロイできるため、JobCenter環境の構築を容易に行えます。

本マニュアルにおいては、MGのアプリケーションコンテナをMGコンテナと呼称します。

2.2. コンテナ環境の設計

コンテナを用いたJobCenterの構成を設計します。

2.2.1. コンテナを用いたJobCenterの構成

Dockerコンテナを用いたJobCenterの構成について説明します。

2.2.1.1. MGの構成

コンテナMGを中心としたJobCenterの構成図は図2.4「コンテナMGを中心とした構成図」のとおりです。

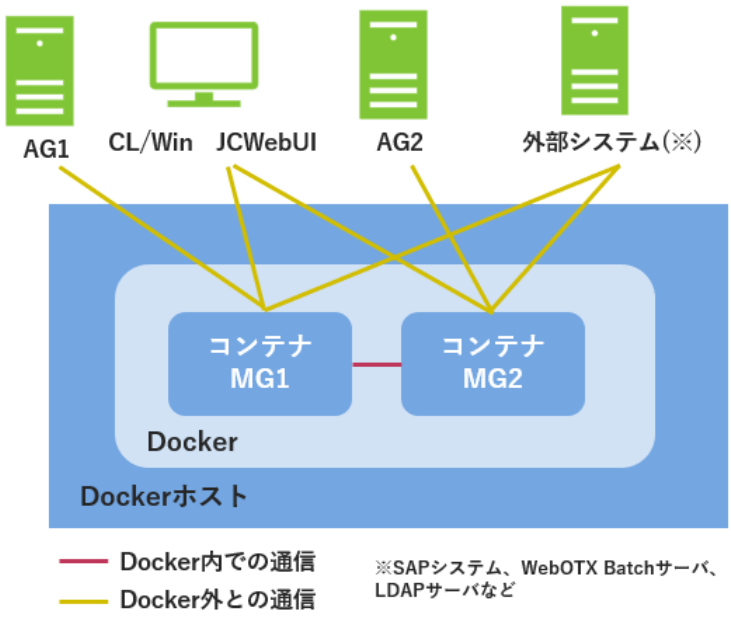

図2.4 コンテナMGを中心とした構成図

2.2.2. コンテナの通信

コンテナの通信は、Docker内の通信とDocker外の通信の2つに分けられます。

■Docker内の通信

Docker内のコンテナ同士の通信のことを指します。Dockerネットワークを利用して通信を行い、Dockerの内蔵DNSサーバ（embedded DNS server）を利用して名前解決を行います。

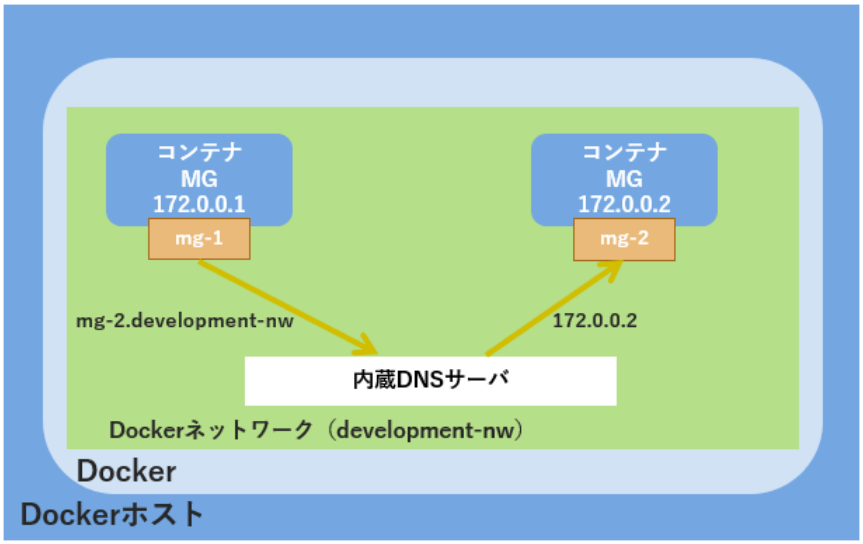

図2.5 Docker内部での通信

Dockerネットワークを作成し、通信するコンテナ同士を同じDockerネットワークに接続することで、Docker内のコンテナ同士の通信を行います。作成したDockerネットワーク（user defined networks）には内蔵DNSサーバが存在しているため、コンテナ同士の名前解決が行えます。

Dockerの内蔵DNSサーバでは、コンテナ名をショートネーム、Dockerネットワーク名をドメイン名としたFQDNによって、コンテナのIPアドレスを名前解決します。そのため、JobCenterのサイト名を<コンテナ名.Dockerネットワーク名>と設定する必要があります。

■Docker外の通信

コンテナとDocker外のマシンとの通信のことを指します。DockerホストのIPアドレスを利用して通信を行い、Docker外のDNSサーバやコンテナ内のhostsファイルを利用して名前解決を行います。

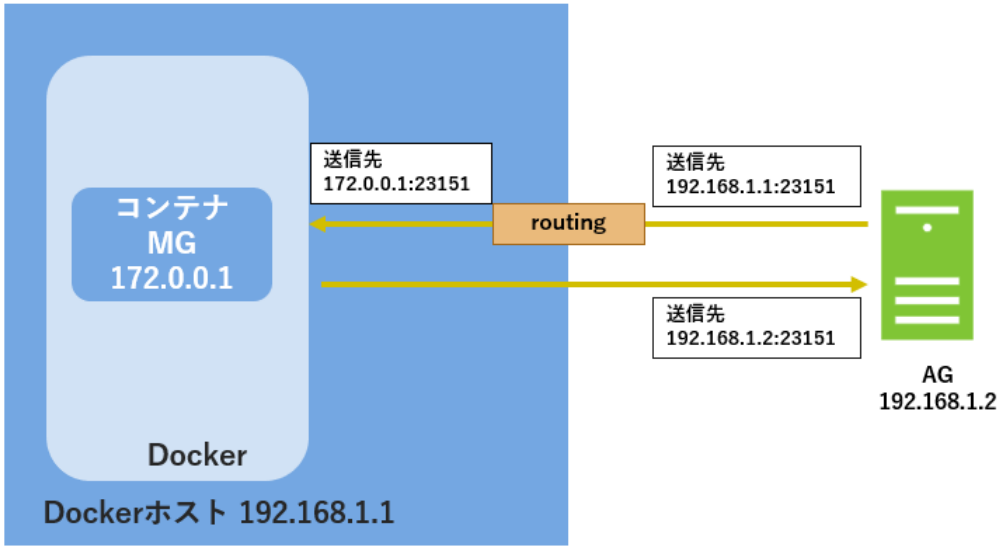

図2.6 Docker外部との通信

コンテナが使用するポート番号を、Dockerホスト上のポート番号として開放することで、DockerホストのIPアドレスを使用してDocker外との通信を行います。これにより、外部からはDockerホストのIPアドレスに対して接続を行うことで、コンテナと通信が行えます。

2.2.2.1. Docker内での通信に必要な作業

Docker内での通信を行う際に必要な作業は次のとおりです。

フェーズ	必要な作業	MG
コンテナ作成前	「2.2.2.1.1 Dockerネットワークの作成」	○
コンテナ作成時	「2.2.2.1.2 コンテナの設定」	○

（凡例）○：作業が必要　×：作業が不要

2.2.2.1.1. Dockerネットワークの作成

コンテナが使用するDockerネットワークを作成します。Dockerネットワークの作成手順については、「2.4.2 Dockerネットワークの作成」を参照してください。

2.2.2.1.2. コンテナの設定

作成したDockerネットワークに、コンテナを接続します。また、名前解決を行うための設定を行います。

MGコンテナの場合は「2.4.4.1.11 Docker内での通信の設定」を参照してください。

2.2.2.2. Docker外との通信に必要な作業

Docker外との通信を行う際に必要な作業は次のとおりです。

フェーズ	必要な作業	MG
コンテナ作成時	「2.2.2.2.1 コンテナの設定」	○

（凡例）○：作業が必要　×：作業が不要

2.2.2.2.1. コンテナの設定

コンテナが利用するポート番号を、Dockerホスト上のIPアドレスと紐づけて公開します。また、名前解決を行うための設定を行います。

MGコンテナの場合は「2.4.4.1.12 Docker外との通信の設定」を参照してください。

2.2.2.3. コンテナの通信における注意事項

■同時に複数のMGコンテナを起動して、Docker外のMGからマシンの追加行うような運用をする場合、「2.4.4.1.12.1 コンテナのポートをDockerホストに割り当て」によってコンテナに割り当てるDockerホストのIPアドレスは、それぞれ別のものを使用する必要があります。同時に起動するコンテナMGの数と同じ数のIPアドレスを、Dockerホストに用意してください。

■コンテナMGとコンテナMGで通信を行う場合、Docker内での通信を利用してください。「2.4.4.1.12.1 コンテナのポートをDockerホストに割り当て」によって各コンテナにDockerホストのIPアドレスを割り当て、Docker外部を経由した通信を行うと、接続に失敗します。

2.2.3. コンテナのデータ管理

コンテナ内のデータは、コンテナが削除されると同時に消失します。Dockerホスト上の領域をコンテナ内にマウントして、コンテナ内のデータをDockerホスト上で管理することで、必要なデータを永続化できます。

また、永続化したデータを用いてコンテナを作成することで、別のコンテナにデータを引き継ぐことができます。

JobCenterにおいては、ログやトラッカ、設定に関連したファイルを永続化できます。これにより、各種設定などを引き継いでコンテナを作成できます。

データの永続化を行うために必要な手順は、次のとおりです。

1. 「2.2.3.1 永続化できるデータ」から、永続化を行うJobCenterのデータを選択します。
2. 「2.2.3.2 永続化方法」を参照して、どの永続化方法を利用するか選択します。
3. bind mountの方式で永続化を行う場合、「2.4.3 マウント元データの作成」を行います。
4. 「2.4.4 コンテナの作成/起動」を行います。

   MGコンテナの場合は「2.4.4.1.9 JobCenterのデータの永続化」を参照してください。

2.2.3.1. 永続化できるデータ

永続化できるJobCenterのデータについては、表2.1「MGで永続化できるデータ」を参照してください。

表2.1 MGで永続化できるデータ

永続化によって引き継げるもの	ファイル名	種類	説明
定義情報やトラッカデータ	/usr/spool/nqs	ディレクトリ	JobCenterの定義等のデータが格納されたスプールディレクトリです。 本ディレクトリを永続化することで、定義ファイルや実行中のジョブの実行結果を、引き継ぐことができます。 スプールディレクトリについては、<スタンダードモード用リリースメモ>の「3.2.4.1 スプールディレクトリ」を参照してください。
JobCenter起動時の設定	/usr/lib/nqs/rc/daemon.conf	ファイル	JobCenterの起動時の設定を行うデーモン設定ファイルです。 本ファイルの設定に関する詳細は、<スタンダードモード用環境構築ガイド>の6章 「JobCenter起動時の設定を変更する」 を参照してください。
	/usr/lib/nqs/rc/jcwebserver.conf	ファイル	jcwebserverの設定ファイルです。 本ファイルの設定に関する詳細は、<スタンダードモード用環境構築ガイド>の「6.7 jcwebserverの動作設定について」を参照してください。
	/usr/lib/nqs/rc/jcexecutor_manager.yaml	ファイル	jcexecutor_managerデーモン独自の動作の設定を行うファイルです。 本ファイルの設定に関する詳細は、<スタンダードモード用環境構築ガイド>の「6.8 jcexecutor_managerデーモンの動作設定について」を参照してください。
	/usr/lib/nqs/rc/jcexecutor_webserver.yaml	ファイル	jcexecutor_webserverデーモン独自の動作の設定を行うファイルです。 本ファイルの設定に関する詳細は、<スタンダードモード用環境構築ガイド>の「6.9 jcexecutor_webserverデーモンの動作設定について」を参照してください。
	/usr/lib/nqs/rc/nats_start.yaml	ファイル	nats-server監視デーモン独自の動作の設定を行うファイルです。 本ファイルの設定に関する詳細は、<スタンダードモード用環境構築ガイド>の「6.10 nats-server監視デーモンの動作設定について」を参照してください。
ジョブ実行ユーザとしてのLDAP連携設定	/etc/nsswitch.conf	ファイル	名前解決の優先度の設定のために利用するファイルです。 本ファイルを永続化することで、名前解決の優先度の設定を、引き継ぐことができます。

永続化によって引き継げるもの	ファイル名	種類	説明
環境変数の引き継ぐ設定	/etc/profile	ファイル	MGでジョブ投入を行った際に、MGからAGへ引き継ぐ環境変数を設定するために利用するファイルです。 本ファイルを永続化することで、<スタンダードモード用環境構築ガイド>の「14.1 UNIX版JobCenterの環境変数」で設定した環境変数の設定を、引き継ぐことができます。
	/home/<ユーザ名>/.nsifrc	ファイル	
JobCenterコマンドの設定	/usr/lib/nqs/gui/bin/jnwschprt.f	ファイル	jnwschprtコマンドのデフォルトのオプションを設定するコンフィグレーションファイルです。 本ファイルの設定に関する詳細は、<スタンダードモード用コマンドリファレンス>の「3.2 jnwschprt ジョブネットワークのカレンダやスケジュール情報を表示」を参照してください。
	/usr/lib/nqs/rc/jcres.conf	ファイル	jcresのデフォルトの動作を変更するための設定ファイルです。 本ファイルの設定に関する詳細は、<スタンダードモード用コマンドリファレンス>の「3.27 jcres JobCenter MG専用のHTTPデーモン」を参照してください。
証明書・秘密鍵ファイルの設定	/usr/spool/nqs/ssl_cert または daemon.confのCOMAGENT_SSLCERTパラメータに設定したファイル	ファイル	CL/Winの「保護された接続」の機能で使用する証明書ファイルです。 本ファイルに関する詳細は、<スタンダードモード用基本操作ガイド>の「2.3 サーバへ接続する」および、<スタンダードモード用環境構築ガイド>の「6.2 デーモン設定ファイルの使用可能パラメータ」を参照してください。
	/usr/spool/nqs/ssl_key または daemon.confのCOMAGENT_SSLKEYパラメータに設定したファイル	ファイル	CL/Winの「保護された接続」の機能で使用する秘密鍵ファイルです。 本ファイルに関する詳細は、<スタンダードモード用基本操作ガイド>の「2.3 サーバへ接続する」および、<スタンダードモード用環境構築ガイド>の「6.2 デーモン設定ファイルの使用可能パラメータ」を参照してください。
	jcwebserver.confのtls.certificateパラメータに設定したファイル	ファイル	jcwebserverの証明書ファイルです。 本ファイルに関する詳細は、<スタンダードモード用環境構築ガイド>の「6.7 jcwebserverの動作設定について」を参照してください。
	jcwebserver.confのtls.privateKeyパラメータに設定したファイル	ファイル	jcwebserverの秘密鍵ファイルです。 本ファイルに関する詳細は、<スタンダードモード用環境構築ガイド>の「6.7 jcwebserverの動作設定について」を参照してください。

2.2.3.2. 永続化方法

JobCenterのコンテナは、bind mountとvolumeによる永続化方法をサポートしています。

それぞれの永続化方法には、次のような特徴があります。

■bind mount

Dockerホスト上に存在するファイルやディレクトリを、コンテナ内にマウントします。ファイルやディレクトリはDockerホスト上か、外部ストレージ上で、お客様ご自身で適切に管理する必要があります。

外部ストレージにはNFSを使用することができます。NFSのストレージをDockerホスト上にマウントし、その領域をコンテナ内にマウントすることで永続化を行います。

■volume

Dockerが管理するvolume内にあるディレクトリを、コンテナ内にマウントします。volumeによって永続化されたデータは、Dockerホスト上の/var/lib/docker/volumes/配下に作成されますが、これらのデータはDockerが管理を行います。

volumeはファイル単位での永続化には利用できません。そのため、表2.1「MGで永続化できるデータ」の種類がディレクトリであるものが、本方式で永続化できる対象のデータになります。

2.2.3.3. コンテナのデータ管理における注意事項

■永続化を行ったJobCenterのデータを、複数のコンテナに対して同時にマウントして使用しないでください。

■Dockerホスト上に存在しているマウント中のファイルは、削除しないでください。コンテナの再起動時にマウント元に指定しているファイルがない場合、コンテナの起動に失敗します。

■NFSのストレージを利用してスプールディレクトリを永続化する場合、NFSの/etc/exportsの設定を、no_root_squashにしてください。

■NFSのストレージを利用してスプールディレクトリを永続化する場合、NFSv4でマウントするようにしてください。

2.3. イメージの作成

DockerfileとJobCenterのパッケージから、イメージのビルドを行います。

Dockerfileは、NECサポートポータルのダウンロード、またはNECカスタマーサポートセンターから入手してください。

本章では、JobCenterのコンテナを作成するための、イメージの作成手順について説明します。

2.3.1. 概要

以下の手順に従い、イメージをビルドすることが可能です。

■事前準備

イメージビルド用のパッケージとDockerfileを事前に用意します。必要なパッケージとDockerfileの詳細は「2.3.2.2 イメージのビルド」を参照してください。

■Dockerfileの作成

JobCenterが提供しているDockerfileをベースに作成します。詳細手順は「2.3.2.1 Dockerfileの作成」を参照してください。

■イメージのビルド

イメージをビルドします。詳細手順は「2.3.2.2 イメージのビルド」を参照してください。

2.3.2. MGコンテナのイメージの作成

JobCenterが提供しているDockerfileをベースにDockerfileを作成し、作成したDockerfileからイメージをビルドします。

イメージのビルド時にMGのインストールを行うため、MGおよびLicenseManagerのパッケージが必要になります。パッケージはイメージのビルドを行うマシン上に配置するか、ファイルサーバ等のリポジトリに配置して利用します。

2.3.2.1. Dockerfileの作成

JobCenterが提供しているDockerfileをそのまま利用できます。

ただし、次のような場合においては、Dockerfileの編集が必要になります。

■パッチのインストールを行いたい場合

DockerfileのARGで指定されているJCPATCHパラメータに、使用するパッチ名を記載します。

```
ARG JCPATCH=<パッチ名>
```

■リポジトリにあるパッケージを利用したい場合

DockerfileのARGで指定されているLMPKG,JCPKG,JCPATCHパラメータに、リポジトリのURLを記載します。

```
ARG LMPKG=<LicenseManagerが配置されているリポジトリのURL>
ARG JCPKG=<JobCenterパッケージが配置されているリポジトリのURL>
ARG JCPATCH=<JobCenterパッチが配置されているリポジトリのURL>
```

URLはhttp（s）で指定してください。

■コンテナ作成時に、JobCenter起動時の設定を適用させたい場合

デーモン設定ファイル（例：daemon.conf）の内容をイメージ内で固定化します。これにより、コンテナを作成/起動した時点でデーモン設定ファイルの設定が適用されるため、　コンテナ作成後にファイルを編集してJobCenterを再起動する手間を省けます。

固定化可能なデーモン設定ファイルは表2.2「固定化可能なデーモン設定ファイル」を参照してください。

表2.2 固定化可能なデーモン設定ファイル

ファイル名	格納場所	説明
daemon.conf	/usr/lib/nqs/rc/daemon.conf	JobCenterの起動時の設定を行うデーモン設定ファイルです。 本ファイルの設定に関する詳細は、<スタンダードモード用環境構築ガイド>の6章 「JobCenter起動時の設定を変更する」 を参照してください。
jcwebserver.conf	/usr/lib/nqs/rc/ jcwebserver.conf	jcwebserverの設定ファイルです。 本ファイルの設定に関する詳細は、<スタンダードモード用環境構築ガイド>の「6.7　jcwebserverの

ファイル名	格納場所	説明
		動作設定について」を参照してください。
jcexecutor_manager.yaml	/usr/lib/nqs/rc/jcexecutor_manager.yaml	jcexecutor_managerデーモン独自の動作の設定を行うファイルです。 本ファイルの設定に関する詳細は、<スタンダードモード用環境構築ガイド>の「6.8 jcexecutor_managerデーモンの動作設定について」を参照してください。
jcexecutor_webserver.yaml	/usr/lib/nqs/rc/jcexecutor_webserver.yaml	jcexecutor_webserverデーモン独自の動作の設定を行うファイルです。 本ファイルの設定に関する詳細は、<スタンダードモード用環境構築ガイド>の「6.9 jcexecutor_webserverデーモンの動作設定について」を参照してください。
nats_start.yaml	/usr/lib/nqs/rc/nats_start.yaml	nats-server監視デーモン独自の動作の設定を行うファイルです。 本ファイルの設定に関する詳細は、<スタンダードモード用環境構築ガイド>の「6.10 nats-server監視デーモンの動作設定について」を参照してください。

以下の手順に従って、デーモン設定ファイルの準備とDockerfileの編集を行ってください。

1. <スタンダードモード用環境構築ガイド>の6章 「JobCenter起動時の設定を変更する」 を参照して、適用したい設定を記載し、固定化するデーモン設定ファイルを作成します。

作成したデーモン設定ファイルは、「2.3.2.2 イメージのビルド」時に、Dockerfileと同じディレクトリ内に配置してください。

2. DockerfileのWORKDIR / 行からCMD ["/usr/lib/nqs/cluster/cjcpw", "-local"] 行の間に、以下を記載します。

```
COPY <デーモン設定ファイル> <デーモン設定ファイルの格納場所>
```

2.3.2.2. イメージのビルド

以下の手順に従って、Dockerfileからイメージのビルドを行います。

1. ディレクトリを作成し、以下の物をディレクトリ内に配置します。

■Dockerfile

■entrypoint.sh

■MGのパッケージ

■LicenseManagerのパッケージ

■（パッチを適用する場合）MGのパッチ

■（デーモン設定ファイルの設定を固定化する場合）デーモン設定ファイル

リポジトリ上のパッケージを利用する場合は、ディレクトリ内にパッケージを配置する必要はありません。

2. ディレクトリ内で以下のコマンドを実行し、イメージのビルドを行います。

```
docker image build -t <作成するイメージ名> .
```

3. 以下のコマンドを実行し、イメージの一覧を表示します。

```
docker image ls
```

<作成するイメージ名>で指定した名前のイメージが存在することを確認してください。

2.3.3. イメージの作成における注意事項

■MGのセットアップの処理を、Dockerfileに記載しないでください。イメージのビルド時にMGのセットアップを行うと、先頭に数字を持つホスト名のコンテナでビルドが行われた場合、セットアップに失敗します。

■JobCenterが提供しているDockerfileにおいて、COPY行はデフォルトのビルドコンテキストで実行されることを前提にしています。docker image buildコマンドの-fオプションを指定してビルドコンテキストを変更した場合、コピー元のパスを適切に修正する必要があります。

■DockerfileのベースイメージのRed Hat Enterprise Linux 7では、デフォルトではsyslogがインストールされていません。そのため、LicenseManagerインストール後に出力されるメッセージがsyslogに記載されません。必要な場合はDockerfileやコンテナ内で、syslogのインストールを行ってください。

参照：<セットアップガイド>の「2.4.2 LicenseManagerインストール後に出力されるメッセージ」

■以下の場合は、docker image buildによってイメージのビルドには成功しますが、作成したイメージからコンテナを起動できません。ビルド時に表示されるメッセージを確認し、各パッケージが正しくインストールされているか確認してください。

- MGがインストールされたイメージを作成する際、指定したMGのパッケージが存在しない

■コンテナの作成/起動時に、--volumeオプションでdaemon.confをマウントした場合、イメージ内で固定化した設定は上書きされます。

2.4. コンテナ環境の構築

本章では、JobCenterのコンテナを作成するための、コンテナ環境の構築手順について説明します。

2.4.1. 概要

コンテナ環境の構築を行います。構築するための手順は次のとおりです。

■ネットワーク準備

Docker内のコンテナ間で通信を行う場合は、「2.4.2 Dockerネットワークの作成」を行います。

■データ永続化のための準備

bind mountの方式でデータの永続化を行う場合は、「2.4.3 マウント元データの作成」を行います。

■コンテナの作成

「2.4.4 コンテナの作成/起動」を参照して、コンテナを作成します。

2.4.2. Dockerネットワークの作成

Docker内でコンテナ間の通信を行う場合、Dockerネットワークを作成し、通信を行うコンテナを作成したネットワークに接続する必要があります。

コンテナの作成/起動を行う前に、以下のコマンドを用いてDockerネットワークを作成してください。

```
docker network create <作成するネットワーク名>
```

作成したDockerネットワークは、以下のコマンドで確認できます。

```
docker network ls
```

2.4.3. マウント元データの作成

「2.2.3.1 永続化できるデータ」で選択したデータを、bind mountの方式で永続化する場合、Dockerホスト上にマウント元のデータを用意する必要があります。

コンテナの作成/起動を行う前に、表2.3「マウント元データの作成方法（MG）」を参照して、マウント元データを作成してください。

表2.3 マウント元データの作成方法（MG）

永続化するデータ	マウント元データの作成方法
/etc/nsswitch.conf	man 5 nsswitch.confを参照して、ファイルを作成してください。
/etc/profile	コンテナ内で使用したい環境変数設定を記載して、ファイルを作成してください。
/home/<ユーザ名>/.nsifrc	<スタンダードモード用環境構築ガイド>の「14.1 UNIX版JobCenterの環境変数」を参照して、ファイルを作成してください。
/usr/lib/nqs/gui/bin/jnwschprt.f	<スタンダードモード用コマンドリファレンス>の「3.2 jnwschprt ジョブネットワークのカレンダやスケジュール情報を表示」を参照して、ファイルを作成してください。
/usr/lib/nqs/rc/daemon.conf	<スタンダードモード用環境構築ガイド>の6章 「JobCenter起動時の設定を変更する」 を参照して、ファイルを作成してください。 daemon.confのテンプレートファイルは、NECサポートポータルのダウンロード、またはNECカスタマーサポートセンターから入手できます。

永続化するデータ	マウント元データの作成方法
/usr/lib/nqs/rc/jcres.conf	<スタンダードモード用コマンドリファレンス>の「3.27 jcres JobCenter MG専用のHTTPデーモン」を参照して、ファイルを作成してください。
/usr/lib/nqs/rc/jcwebserver.conf	<スタンダードモード用環境構築ガイド>の「6.7 jcwebserverの動作設定について」を参照して、ファイルを作成してください。
/usr/lib/nqs/rc/jcexecutor_manager.yaml	<スタンダードモード用環境構築ガイド>の「6.8 jcexecutor_managerデーモンの動作設定について」を参照して、ファイルを作成してください。
/usr/lib/nqs/rc/jcexecutor_webserver.yaml	<スタンダードモード用環境構築ガイド>の「6.9 jcexecutor_webserverデーモンの動作設定について」を参照して、ファイルを作成してください。
/usr/lib/nqs/rc/nats_start.yaml	<スタンダードモード用環境構築ガイド>の「6.10 nats-server監視デーモンの動作設定について」参照して、ファイルを作成してください。
/usr/spool/nqs/ssl_cert または daemon.confのCOMAGENT_SSLCERTパラメータに設定したファイル	<スタンダードモード用基本操作ガイド>の「2.3 サーバへ接続する」および、<スタンダードモード用環境構築ガイド>の「6.2 デーモン設定ファイルの使用可能パラメータ」を参照して、証明書ファイルを作成してください。
/usr/spool/nqs/ssl_key または daemon.confのCOMAGENT_SSLKEYパラメータに設定したファイル	<スタンダードモード用基本操作ガイド>の「2.3 サーバへ接続する」および、<スタンダードモード用環境構築ガイド>の「6.2 デーモン設定ファイルの使用可能パラメータ」を参照して、秘密鍵ファイルを作成してください。
jcwebserver.confのtls.certificateパラメータに設定したファイル	<スタンダードモード用環境構築ガイド>の「6.7 jcwebserverの動作設定について」を参照して、証明書ファイルを作成してください。
jcwebserver.confのtls.privateKeyパラメータに設定したファイル	<スタンダードモード用環境構築ガイド>の「6.7 jcwebserverの動作設定について」を参照して、秘密鍵ファイルを作成してください。
/usr/spool/nqs	空ディレクトリを作成してください。

2.4.4. コンテナの作成/起動

作成したイメージから、コンテナの作成/起動を行います。

MGコンテナについては「2.4.4.1 MGコンテナの作成/起動」を参照してください。

2.4.4.1. MGコンテナの作成/起動

docker container runコマンドによって、コンテナの作成/起動を行います。

オプションを指定することで、コンテナに対して表2.4「MGのコンテナの設定」に記載されている設定を行います。表中に◎が記載されているものは、コンテナを正常に起動させるために必ず設定する必要があります。

コンテナを起動させるために必要なオプションを付与したdocker container runコマンドは以下です。

■MGコンテナ

```
docker container run --detach --hostname <コンテナホスト名> --volume <.lockinfoファイルのパス>:/etc/opt/wsnlesd/.lockinfo --init --env JOBCENTER_NSUMSMGR_PASS=<パスワード> <イメージ名/イメージID>
```

表2.4 MGのコンテナの設定

設定	MGコンテナ
「2.4.4.1.1 バックグラウンドで起動」	◎
「2.4.4.1.2 コンテナホスト名の設定」	◎
「2.4.4.1.3 ライセンスの登録」	◎
「2.4.4.1.4 initプロセスの設定」	◎
「2.4.4.1.5 JobCenter管理者ユーザのパスワードの設定」	◎
「2.4.4.1.6 JobCenterのセットアップ時の設定」	○
「2.4.4.1.7 JobCenterが使用するプロトコルのポート番号の変更」	○
「2.4.4.1.8 一般ユーザの作成」	○
「2.4.4.1.9 JobCenterのデータの永続化」	○
「2.4.4.1.10 永続化したデータの引き継ぎ」	○
「2.4.4.1.11 Docker内での通信の設定」	○
「2.4.4.1.12 Docker外との通信の設定」	○

（凡例）◎：必ず利用する設定　○：必要に応じて利用する設定　×：利用できない設定

2.4.4.1.1. バックグラウンドで起動

利用するdocker container runのオプション	--detach

コンテナをバックグラウンドで起動します。MGは常駐アプリケーションなため、コンテナはバックグラウンドで動作させます。

2.4.4.1.2. コンテナホスト名の設定

利用するdocker container runのオプション	--hostname <コンテナホスト名>

コンテナホスト名を指定します。コンテナホスト名はJobCenterのサイト名として登録されます。

指定するコンテナホスト名には以下の制限があります。

■64文字以内

■マルチバイト文字の使用は禁止

■先頭の文字に数字の使用は禁止

Docker内での通信を行う場合は、コンテナホスト名にはコンテナ名.ネットワーク名を指定する必要があります。

詳細は「2.4.4.1.11 Docker内での通信の設定」を参照してください。

2.4.4.1.3. ライセンスの登録

利用するdocker container runのオプション	--volume <マウント元>:<マウント先>

■マウント元

Dockerホスト上に存在する.lockinfoファイルのパスを指定します。

■マウント先

/etc/opt/wsnlesd/.lockinfoを指定します。

コンテナを起動させるためには、コードワードが登録された.lockinfoファイルが必要です。

Dockerホスト上でコードワードが記載された.lockinfoファイルを用意し、そのファイルをコンテナ内の/etc/opt/wsnlesd/.lockinfoにマウントすることで、コンテナ内のMGのライセンス登録を行います。

コードワードの登録に関しては<セットアップガイド>の「2.4　コードワードを登録する」を参照してください。

2.4.4.1.4. initプロセスの設定

利用するdocker container runのオプション	--init

コンテナのrootプロセスを/dev/initにします。--init を指定していない場合、コンテナ内で実行したプロセスの終了後、OSによって回収（reap）されない場合があります。

コンテナソフトウェアにPodmanを使用している場合は、以下の手順を実施してください。

1. krallin / tini のサイトから最新のtiniの実行ファイルを入手します。

   ```
   https://github.com/krallin/tini
   ```

2. tiniの実行ファイルをpodmanコマンドを実行するマシンの任意の場所に格納します。

3. --initオプションと--init-pathオプションを両方指定します。--init-pathオプションにはtiniを格納したパスを指定してください。

   ```
   --init --init-path <tiniを格納したパス>
   ```

2.4.4.1.5. JobCenter管理者ユーザのパスワードの設定

利用するdocker container runのオプション	--env <環境変数名>=<環境変数値>

環境変数を利用して、JobCenter管理者ユーザ（nsumsmgr）のパスワードを設定します。パスワードは平文または暗号化されたものを使用できます。

指定する環境変数は表2.5「JobCenter管理者ユーザのパスワードの設定を行う環境変数」を参照してください。

パスワードの暗号化については「2.4.4.2.2 パスワードの暗号化」を参照してください。

表2.5 JobCenter管理者ユーザのパスワードの設定を行う環境変数

環境変数名	環境変数値
JOBCENTER_NSUMSMGR_PASS	JobCenter管理者ユーザの平文のパスワード
JOBCENTER_NSUMSMGR_CRYPTED_PASS	JobCenter管理者ユーザの暗号化されたパスワード

2.4.4.1.6. JobCenterのセットアップ時の設定

利用するdocker container runのオプション	--env <環境変数名>=<環境変数値>

環境変数を利用して、JobCenterのマシンIDおよび使用する言語を設定します。

指定する環境変数は表2.6「JobCenterのセットアップ時の設定を行う環境変数」を参照してください。

表2.6 JobCenterのセットアップ時の設定を行う環境変数

環境変数名	環境変数値	デフォルト値	範囲
JOBCENTER_LANGUAGE_CODE	セットアップ時に指定する文字コード	UTF-8	English EUC Shift-JIS Chinese UTF-8
JOBCENTER_MACHINE_ID	セットアップ時に指定するマシンID	1	1 ～ 2147483647

2.4.4.1.7. JobCenterが使用するプロトコルのポート番号の変更

利用するdocker container runのオプション	--env <環境変数名>=<環境変数値>

環境変数を利用して、JobCenterが使用するプロトコルのポート番号を変更します。

指定する環境変数は表2.7「JobCenterが使用するプロトコルのポート番号の変更を行う環境変数」を参照してください。

表2.7 JobCenterが使用するプロトコルのポート番号の変更を行う環境変数

環境変数名	環境変数値	デフォルト値	範囲
JOBCENTER_JCCOMBASE_PORT	jccombaseのポート番号	611	1 ～ 65535
JOBCENTER_JCCOMBASE_OVER_SSL_PORT	jccombase over sslのポート番号	23116	1 ～ 65535
JOBCENTER_JCEVENT_PORT	jceventのポート番号	10012	1 ～ 65535
JOBCENTER_JCWEBSERVER_PORT	jcwebserverのポート番号	23180	1 ～ 65535
JOBCENTER_JCNATS_PORT	jcnatsのポート番号	23141	1 ～ 65535
JOBCENTER_JCEXECUTOR_PORT	jcexecutorのポート番号	23151	1 ～ 65535

各プロトコルの詳細については、<スタンダードモード用リリースメモ>の「3.4　使用するネットワークポート」を参照してください。

2.4.4.1.8. 一般ユーザの作成

作成するユーザ名とパスワードが記載されたファイル（以下、パスワードファイル）を利用して、JobCenterで利用するための一般ユーザを作成します。

必要な手順は次のとおりです。

■「2.4.4.1.8.1 パスワードファイルの作成」

■「2.4.4.1.8.2 パスワードファイルのマウント」

■「2.4.4.1.8.3 環境変数でマウント先のパスを指定」

2.4.4.1.8.1. パスワードファイルの作成

Dockerホスト上にパスワードファイルを用意します。パスワードファイルには<ユーザ名>:<パスワード>の形式で、作成したいユーザ名とパスワードを記載します。

記載例）

```
user1:password1
user2:password2
```

パスワードは平文のものと暗号化されたものが使用できます。

パスワードの暗号化については「2.4.4.2.2 パスワードの暗号化」を参照してください。

パスワードファイルは、コンテナ内にマウントする必要があります。そのため、コンテナ作成後もDockerホスト上およびコンテナ内に、パスワードファイルが残り続けます。一般ユーザからパスワードファイルを参照されたくない場合は、chmodコマンドなどを使用して、パスワードファイルに適切な権限を与えて管理してください。

パスワードファイルに関する注意事項

■パスワードファイルのサイズは512KBまでです。512KBを超えるファイルサイズのパスワードファイルは使用できません。

■作成するユーザ名は、OSの制限を守ってください。OSで使用できないユーザ名を指定した場合、コンテナの起動に失敗します。

■<スタンダードモード用リリースメモ>の「6.2.1　制限事項」に該当するユーザ名を使用しないでください。

■作成できるユーザ数に制限はありません。

■パスワードファイル内に同一ユーザ名が複数回記載されていた場合、一番後ろで指定されているパスワードの値で設定されます。

■JobCenter管理者ユーザのパスワードは設定できません。「2.4.4.1.5　JobCenter管理者ユーザのパスワードの設定」で指定した値が、パスワードとして設定されます。

■以下のパスワードは、CL/WinからMGへのログイン時に使用できないため、設定しないでください。

- 末尾の奇数個の\
- 対応が取れていない波括弧（例:a{b）

■1つのパスワードファイル内で、平文のパスワードと暗号化されたパスワードを併用することはできません。

2.4.4.1.8.2. パスワードファイルのマウント

利用するdocker container runのオプション	--volume <マウント元>:<マウント先>

■マウント元

Dockerホスト上に存在するパスワードファイルのパスを指定します。

■マウント先

任意のパスを指定します。

Dockerホスト上で作成したパスワードファイルを、コンテナ内の任意の箇所にマウントします。

2.4.4.1.8.3. 環境変数でマウント先のパスを指定

利用するdocker container runのオプション	--env <環境変数名>=<環境変数値>

環境変数の値に、「2.4.4.1.8.2 パスワードファイルのマウント」でマウント先に指定したパスを指定します。

パスワードファイルに使用したパスワードの種類によって、設定する環境変数名が変わります。指定する環境変数は表2.8「パスワードファイルのマウント先のパスを指定する環境変数」を参照してください。

表2.8 パスワードファイルのマウント先のパスを指定する環境変数

環境変数名	環境変数値
JOBCENTER_USERS_PASS_FILE	コンテナ内に存在する、パスワードが平文で記載されているパスワードファイルのパス
JOBCENTER_USERS_CRYPTED_PASS_FILE	コンテナ内に存在する、パスワードが暗号化されて記載されているパスワードファイルのパス

2.4.4.1.9. JobCenterのデータの永続化

利用するdocker container runのオプション	--volume <マウント元>:<マウント先>

■マウント元

永続化方式によって、指定するものが異なります。

bind　mount方式の場合、Dockerホスト上に存在する、マウント元のファイルのパスを、/から始まる絶対パスの形式で指定します。

volume方式の場合、使用するvolume名を指定します。指定したvolumeが存在しない場合、新しくvolumeが作成されます。

■マウント先

コンテナ内のマウント先のファイルのパスを、/から始まる絶対パスの形式で指定します。

マウント先のファイルのパスは、表2.1「MGで永続化できるデータ」の、ファイル名に記載されています。

ホスト側がSELinux有効環境の場合は「--volume　<マウント元>:<マウント先>:Z」のようにして、永続化対象ディレクトリのセキュリティコンテキストにcontainer_file_tラベルを付与してください。

コンテナ内のファイルやディレクトリを、Dockerホスト上からマウントすることで、JobCenterのデータを永続化します。

永続化については、「2.2.3 コンテナのデータ管理」を参照してください。

2.4.4.1.10. 永続化したデータの引き継ぎ

利用するdocker container runのオプション	--volume <マウント元>:<マウント先>

■マウント元

永続化方式によって、指定するものが異なります。

bind　mount方式の場合、Dockerホスト上に存在する、マウント元のファイルのパスを、/から始まる絶対パスの形式で指定します。

volume方式の場合、使用するvolume名を指定します。

■マウント先

コンテナ内のマウント先のファイルのパスを、/から始まる絶対パスの形式で指定します。

マウント先のファイルのパスは、表2.1「MGで永続化できるデータ」の、ファイル名に記載されています。

ホスト側がSELinux有効環境の場合は「--volume　<マウント元>:<マウント先>:Z」のようにして、永続化対象ディレクトリのセキュリティコンテキストにcontainer_file_tラベルを付与してください。

「2.4.4.1.9　JobCenterのデータの永続化」によって永続化したデータを引き継いで、コンテナを作成できます。

2.4.4.1.11. Docker内での通信の設定

Docker内で他のコンテナと通信を行う場合、以下の設定を行います。

■「2.4.4.1.11.1 ネットワークの指定」

■「2.4.4.1.11.2 コンテナ名の設定」

■「2.4.4.1.11.3 コンテナホスト名の設定」

2.4.4.1.11.1. ネットワークの指定

利用するdocker container runのオプション	--network <Dockerネットワーク名>

コンテナをDockerネットワークに接続します。

「2.4.2 Dockerネットワークの作成」で作成したDockerネットワーク名を指定してください。

2.4.4.1.11.2. コンテナ名の設定

利用するdocker container runのオプション	--name <コンテナ名>

コンテナ名を指定します。

重複するコンテナ名を指定することはできません。そのため、接続するDockerネットワークを別にしてコンテナホスト名のショート名は同じにする、というような構成にすることはできません。

2.4.4.1.11.3. コンテナホスト名の設定

利用するdocker container runのオプション	--hostname <コンテナホスト名>

■コンテナホスト名

<コンテナ名>.<ネットワーク名>を指定します。

コンテナホスト名を指定します。

Dockerの内蔵DNSサーバでは、コンテナのIPアドレスは、コンテナ名をショートネーム、ネットワーク名をドメインネームとしたFQDNで名前解決が行われます。そのため、コンテナホスト名にはコンテナ名.ネットワーク名を指定する必要があります。

指定するコンテナホスト名には以下の制限があります。

■64文字以内

■マルチバイト文字の使用は禁止

■先頭の文字に数字の使用は禁止

2.4.4.1.12. Docker外との通信の設定

Docker外との通信を行う場合、以下の設定を行います。

■「2.4.4.1.12.1 コンテナのポートをDockerホストに割り当て」

■「2.4.4.1.12.2 名前解決方法の設定」

2.4.4.1.12.1. コンテナのポートをDockerホストに割り当て

利用するdocker container runのオプション	--publish <DockerホストのIPアドレス>:<ホスト側のポート番号>:<コンテナ側のポート番号>

コンテナが利用するポート番号を、DockerホストのIPアドレスと紐づいたポート番号として開放します。これにより、DockerホストマシンのIPアドレスを、コンテナのIPアドレスとして利用できます。

MGが使用するポートは以下です。

ポート番号	説明
611	jccombaseが使用するポート番号
23116	jccombase over sslが使用するポート番号
10012	jceventが使用するポート番号
23180	jcwebserverが使用するポート番号
50080	jcresが使用するポート番号
23141	jcnatsのポート番号
23151	jcexecutorのポート番号

「2.4.4.1.7 JobCenterが使用するプロトコルのポート番号の変更」によってポート番号を変更した場合は、変更したポート番号を開放してください。

2.4.4.1.12.2. 名前解決方法の設定

利用するdocker container runのオプション	--dns <DNSサーバのIPアドレス>

コンテナはデフォルトではDockerホストの/etc/resolv.confで設定されているDNSサーバを利用します。

Dockerホストの設定とは異なるDNSサーバを利用したい場合、使用したいDNSサーバのIPアドレスを指定します。

コンテナへの名前解決には、「2.4.4.1.12.1 コンテナのポートをDockerホストに割り当て」で指定したDockerホストのIPアドレスと、「2.4.4.1.2　コンテナホスト名の設定」で指定したコンテナホスト名で行われるように、DNSサーバで設定してください。

2.4.4.2. 参考情報

コンテナの起動/作成における参考情報について記載します。

2.4.4.2.1. 環境変数をファイルで渡す

```
--env-file <環境変数を記載したファイルのパス>
```

--env-fileオプションを使用することで、環境変数を1つのファイルにまとめて設定できます。

環境変数をファイルで設定することにより、以下のようなメリットがあります。

■環境変数を利用する設定が多いため、一括で管理することでコマンド実行時の入力ミスを減らせます。

■環境変数値を秘匿化できます。環境変数値にパスワードを指定する場合、--env-fileオプションではファイルのパスを指定するため、docker container runコマンド実行時のプロセスにパスワードが表示されることを防げます。

--env-fileオプションおよび環境変数を記載するファイルの詳細は、Docker社が提供している公式ドキュメントを参照してください。

2.4.4.2.2. パスワードの暗号化

「2.4.4.1.5 JobCenter管理者ユーザのパスワードの設定」および「2.4.4.1.8 一般ユーザの作成」では、環境変数値やファイルにユーザのパスワードを記載しますが、これらのパスワードは暗号化することもできます。

/etc/shadowファイルの第二フィールドの形式に従って、パスワードの暗号化を行ってください。

pythonおよびrubyを使用した、パスワードの暗号化方法例は次のとおりです。

■python

```
python -c 'import crypt; print(crypt.crypt("</パスワード>", salt="$<ID>$<salt>"))'
```

■ruby

```
ruby -e 'puts "</パスワード>".crypt("$<ID>$<salt>")'
```

</パスワード>:暗号化したいパスワードを指定します。

<ID>:暗号化方式を指定します。詳細はman 3 cryptを参照してください。

<salt>:16文字以下の任意の文字列を指定します。

2.4.4.2.3. 異なるDockerネットワークに接続しているコンテナ同士の通信

```
docker network connect <Dockerネットワーク名> <コンテナ名/コンテナID>
```

docker network connectコマンドを用いることで、起動中のコンテナをDockerネットワークに接続できます。

これにより、異なるDockerネットワークに接続しているコンテナ同士でも、コンテナの再作成を行うことなく通信を行えます。

docker network connectコマンドをそれぞれのコンテナに対して実行し、通信をしたいコンテナ同士を、お互いのDockerネットワークに接続してください。

例）ネットワークAに接続しているコンテナAと、ネットワークBに接続しているコンテナBで通信を行いたい場合

■コンテナAを、ネットワークBに接続する

```
docker network connect <ネットワークB> <コンテナA>
```

■コンテナBを、ネットワークAに接続する

```
docker network connect <ネットワークA> <コンテナB>
```

2.4.5. コンテナ環境の構築における注意事項

■コンテナが使用するメモリの上限に制限をかける場合は、<スタンダードモード用リリースメモ>の「3.2.1 必要メモリ量・ディスク容量」および<スタンダードモード用環境構築ガイド>の22章　「システム利用資源」 を参照して、適切に設定してください。

■「2.4.4.1.10 永続化したデータの引き継ぎ」で、スプールディレクトリを引き継ぐ時の注意事項

- 「2.4.4.1.6 JobCenterのセットアップ時の設定」で環境変数JOBCENTER_MACHINE_IDを指定しても、マシンIDは変更されません。スプールディレクトリで設定されているマシンIDで動作します。
- 「2.4.4.1.6　JobCenterのセットアップ時の設定」で指定する環境変数JOBCENTER_LANGUAGE_CODEには、スプールディレクトリで設定されている文字コードと同じものを指定してください。違う値を指定した場合、JobCenterは正常に動作しません。
- 「2.4.4.1.2 コンテナホスト名の設定」で指定するコンテナホスト名は、スプールディレクトリで設定されているJobCenterサイト名と同じものを指定してください。違う値を指定した場合、コンテナの作成/起動に失敗します。

■コンテナの起動直後は、MGの起動が完了していません。コンテナの起動後30秒ほど待ってから、MGに接続してください。

■JobCenterが提供しているDockerfileに記載してあるラベルは、理由がない限り削除しないでください。ラベルが設定されていない場合、jc_docker_getinfo.shによる情報採取が行われないため、情報採取の際に手動で情報を採取する必要があります。

■コンテナ環境のMGにおいて、ファイル待ち合わせ部品のclose_check機能を利用する場合、以下が必要になります。

- fuserコマンドのインストール

  ```
  yum install psmisc
  ```

- capabilityの付与

  docker container run時のオプションで--cap-add=SYS_PTRACEを指定する

close_check機能については、<スタンダードモード用環境構築ガイド>の「6.2 デーモン設定ファイルの使用可能パラメータ」の3. JNWENGINE_OPTの-Fオプションを参照してください。

2.5. コンテナ環境の運用

JobCenterをコンテナ環境で運用する際の操作について説明します。

2.5.1. コンテナ内で操作

起動中のコンテナにbashで接続する場合は、以下のコマンドを使用します。

```
docker container exec -it <コンテナ名/コンテナID> bash
```

JobCenterのコマンドの実行などを行いたい場合は、コンテナにbashで接続し、コンテナ内からコマンドを実行してください。

2.5.1.1. コンテナ内のMGの停止、起動

コンテナを起動させたまま、コンテナ内のMGの停止を行いたい場合、コンテナ内でコマンドを実行します。

コンテナ内でMGを停止させる場合、JobCenterのすべてのプロセスの起動が完了したことを確認してから実行してください。JobCenterのプロセスについては、<スタンダードモード用環境構築ガイド>の表19.3「JobCenter常駐プロセス一覧（UNIX）」を参照してください。

■MGコンテナの場合

JobCenterのコマンドを利用します。

停止する場合は/usr/lib/nqs/nqsstopを、起動する場合は/usr/lib/nqs/nqsstartを実行してください。

各コマンドの詳細は<スタンダードモード用コマンドリファレンス>の「3.11 nqsstop デーモンプロセスを停止」、「3.10 nqsstart デーモンプロセスを起動」を参照してください。

2.5.2. コンテナの管理

コンテナを管理する上で利用するdockerコマンドについて説明します。

■コンテナの停止

```
docker container stop <コンテナ名/コンテナID>
```

指定したコンテナを停止状態にします。

■コンテナの起動

```
docker container start <コンテナ名/コンテナID>
```

指定したコンテナを起動状態にします。

■コンテナの再起動

```
docker container restart <コンテナ名/コンテナID>
```

指定したコンテナを再起動します。

■コンテナー覧の表示

```
docker container ls
```

起動中のコンテナの一覧を表示します。停止中のコンテナも表示したい場合は、-aオプションを指定してください。

■コンテナの情報を表示

```
docker container inspect <コンテナ名/コンテナID>
```

指定したコンテナの情報を表示します。

■コンテナのログを表示

```
docker container logs <コンテナ名/コンテナID>
```

指定したコンテナのログを表示します。作成したコンテナが起動しない場合や、起動中のコンテナが停止した場合は、本コマンドを実行して表示されるメッセージを確認してください。

■コンテナの削除

```
docker container rm <コンテナ名/コンテナID>
```

指定したコンテナを削除します。削除するコンテナは、あらかじめ停止させる必要があります。

コンテナの管理における注意事項

■コンテナの停止に、docker container killコマンドを用いないでください。

docker container killコマンドでコンテナを停止すると、コンテナのrootプロセスにSIGKILLが送信されて強制終了します。JobCenterが強制終了した場合、データの不整合などが発生し、正常に動作しない可能性があります。

■コンテナの停止に10秒以上かかる場合、docker container stopコマンドの--timeオプションで、タイムアウト時間を調節してください。

```
docker container stop --time <秒数> <コンテナ名/コンテナID>
```

タイムアウト時間を超過した場合、docker container stopコマンドはコンテナ内のすべてのプロセスを強制終了します。以下のような環境の場合はコンテナの停止に時間がかかる可能性がありますのでご注意ください。

- Dockerホスト側で負荷が高い場合

■コンテナの停止時に送信されるシグナルを、本マニュアルで記載されている内容以外で変更しないでください。

アプリケーションコンテナの場合はSIGTERM（デフォルト）である必要があります。ほかのシグナルに変更した場合、コンテナ停止時にMGを正常に停止できません。

2.5.3. イメージの管理

イメージを管理する上で利用するdockerコマンドについて説明します。

■イメージ一覧の表示

```
docker image ls
```

作成されているイメージの一覧を表示します。

■イメージの情報を表示

```
docker image inspect <イメージ名/イメージID>
```

指定したイメージの情報を表示します。

■イメージの削除

```
docker image rm <イメージ名/イメージID>
```

指定したイメージを削除します。指定したイメージから作成されているコンテナが存在する場合、イメージは削除できません。

2.5.4. ボリュームの管理

ボリュームを管理する上で利用するdockerコマンドについて説明します。

■ボリューム一覧の表示

```
docker volume ls
```

作成されているボリュームの一覧を表示します。

■ボリュームの情報を表示

```
docker volume inspect <ボリューム名>
```

指定したボリュームの情報を表示します。

■ボリュームの削除

```
docker volume rm <ボリューム名>
```

指定したボリュームを削除します。コンテナにマウントされているボリュームは削除できません。

2.6. コンテナ環境のトラブルシューティング

コンテナ環境の構築および運用中に障害が発生した場合の、トラブルシューティングに関する情報について説明します。

2.6.1. 障害発生時の対応方法

コンテナ環境の構築および運用の各フェーズで障害が発生した場合の、確認観点と対処方法について説明します。

MGコンテナについては「2.6.1.1 MGコンテナにおける障害」を参照してください。

記載されている対処を行っても障害が解消されない場合、「2.6.2　障害発生時の情報採取方法」の手順に従って情報採取を行い、サポート窓口へお問い合わせください。

2.6.1.1. MGコンテナにおける障害

表2.9「MGコンテナにおける障害」を参照してください。

表2.9 MGコンテナにおける障害

フェーズ	障害内容
イメージの作成時	「2.6.1.1.1 イメージの作成に失敗する」
コンテナ環境の構築時	「2.6.1.1.2 コンテナが起動しない/起動後停止する」
コンテナ環境の運用時	「2.6.1.1.3 MGへのログインに失敗する」
コンテナ環境の運用時	「2.6.1.1.4 マシン一覧へMGが追加できない」
コンテナ環境の運用時	「2.6.1.1.5 リモート（AG）投入したジョブがエラーになる」
コンテナ環境の運用時	「2.6.1.1.6 リモート(AG) 投入したジョブがSUBMIT状態のまま進行しない」
コンテナ環境の運用時	「2.6.1.1.7 フローが進行しない」

2.6.1.1.1. イメージの作成に失敗する

docker image buildコマンド実行時に表示されたメッセージを確認し、対処を行ってください。

エラーメッセージ内容	考えられるエラーの原因と対処方法
lstat entrypoint.sh: no such file or directory	entrypoint.shが存在していません。 Dockerfileと同じディレクトリ内にentrypoint.shが配置されているか確認してください。
Error: Package: <MGのパッケージ名> (/<MGのパッケージ名>) Requires: NECWSLM	License Managerのパッケージが存在していません。 Dockerfileが配置されているディレクトリ内に、License Managerのパッケージが配置されているか確認してください。 リポジトリにあるパッケージを利用する場合は、DockerfileのLMPKGに指定したURLが正しいか確認してください。

2.6.1.1.2. コンテナが起動しない/起動後停止する

起動に失敗したコンテナに対して以下のコマンドを実行し、表示されたメッセージを確認して対処を行ってください。

```
docker container logs <コンテナ名/コンテナID>
```

エラーメッセージ内容	考えられるエラーの原因と対処方法
/entrypoint.sh: line 82: /usr/local/netshep/nssetup: No such file or directory	イメージ内にMGのパッケージがインストールされていません。 以下の点を確認した上で、イメージを再作成してください。 ■Dockerfileが配置されているディレクトリ内に、MGのパッケージが配置されていること ■リポジトリにあるパッケージを利用する場合、DockerfileのJCPKGに正しいURLが指定されていること
Hostname is invalid. [<コンテナホスト名>]	--hostnameオプションが指定されていない、または不適切な値が指定されています。 「2.4.4.1.2 コンテナホスト名の設定」を参照して、コンテナホスト名に適切な値を設定してください。

エラーメッセージ内容	考えられるエラーの原因と対処方法
Error: No valid license is registered. nqsdaemon start process will exit.	有効なライセンスが登録されていません。 「2.4.4.1.3 ライセンスの登録」を参照して、コンテナ内のMGのライセンス登録を行ってください。
exec -- failed: No such file or directory	--initオプションが利用できません。 Dockerのリリース番号が86以降のもの（docker-1.13.1-86.git07f3374.el7）を利用してください。
At least one of the JOBCENTER_NSUMSMGR_PASS or JOBCENTER_NSUMSMGR_CRYPTED_PASS must be specified.	--envオプションで指定する環境変数 JOBCENTER_NSUMSMGR_PASS および JOBCENTER_NSUMSMGR_CRYPTED_PASS が、どちらも指定されていません。 「2.4.4.1.5 JobCenter管理者ユーザのパスワードの設定」を参照して、JobCenter管理者ユーザのパスワードの設定を行ってください。
Language code is invalid value. [<環境変数値>]	--envオプションで指定する環境変数 JOBCENTER_LANGUAGE_CODE に、不適切な値が指定されています。 「2.4.4.1.6 JobCenterのセットアップ時の設定」を参照して、適切な値を指定してください。
Machine ID is positive integer required. [<環境変数値>]	--envオプションで指定する環境変数 JOBCENTER_MACHINE_ID に、整数値以外が指定されています。 「2.4.4.1.6 JobCenterのセットアップ時の設定」を参照して、適切な値を指定してください。
Machine ID is in the invalid range. [<環境変数値>]	--envオプションで指定する環境変数 JOBCENTER_MACHINE_ID に、範囲外の値が指定されています。 「2.4.4.1.6 JobCenterのセットアップ時の設定」を参照して、適切な値を指定してください。
JobCenter port is positive integer required. [<環境変数名>=<環境変数値>]	--envオプションで指定する以下の環境変数に整数値以外が指定されています。 ■JOBCENTER_JCCOMBASE_PORT ■JOBCENTER_JCCOMBASE_OVER_SSL_PORT ■JOBCENTER_JCEVENT_PORT ■JOBCENTER_JCWEBSERVER_PORT ■JOBCENTER_JCNATS_PORT ■JOBCENTER_JCEXECUTOR_PORT 「2.4.4.1.7 JobCenterが使用するプロトコルのポート番号の変更」を参照して、適切な値を指定してください。
JobCenter port is in the invalid range. [<環境変数名>=<環境変数値>]	--envオプションで指定する以下の環境変数に範囲外の値が指定されています。 ■JOBCENTER_JCCOMBASE_PORT ■JOBCENTER_JCCOMBASE_OVER_SSL_PORT

エラーメッセージ内容	考えられるエラーの原因と対処方法
	■JOBCENTER_JCEVENT_PORT ■JOBCENTER_JCWEBSERVER_PORT ■JOBCENTER_JCNATS_PORT ■JOBCENTER_JCEXECUTOR_PORT 「2.4.4.1.7 JobCenterが使用するプロトコルのポート番号の変更」を参照して、適切な値を指定してください。
Password file is not exists. [<環境変数名>=<環境変数値>]	--envオプションで指定する環境変数 JOBCENTER_USERS_PASS_FILE および JOBCENTER_USERS_CRYPTED_PASS_FILE で指定したパスに、ファイルが存在しません。 --volumeオプションで指定した、パスワードファイルのマウント先のパスを、環境変数値に指定してください。
Password file is not a regular file. [<環境変数名>=<環境変数値>]	--envオプションで指定する環境変数 JOBCENTER_USERS_PASS_FILE および JOBCENTER_USERS_CRYPTED_PASS_FILE で指定したパスに存在するファイルが、レギュラーファイルではありません。 --volumeオプションで指定するマウント元のパスに、Dockerホスト上に存在するパスワードファイルのパスを、正しく指定してください。
ENV <環境変数名> line <行数>: the format is not valid, the correct separator must be ":".	パスワードファイルの<行数>行目のフォーマットが不正です。 「2.4.4.1.8.1 パスワードファイルの作成」を参照して、正しいフォーマットでパスワードファイルを作成してください。
ENV <環境変数名> line <行数>: the user is not created correctly. [<ユーザ名>]	パスワードファイルの<行数>行目のユーザ名が不正です。 ユーザ名はOSの制限を守ってください。
ENV <環境変数名>: the file size [<ファイルサイズ> bytes] is larger than maximum size [524288 bytes].	パスワードファイルのサイズが、512KBを超えています。 512KB以内のサイズでパスワードファイルを作成してください。
Site names do not match.	■コンテナ作成後、すぐに停止した場合 スプールディレクトリで設定されているJobCenterのサイト名と、--hostnameオプションで指定したコンテナホスト名が一致していません。 「2.4.4.1.10 永続化したデータの引き継ぎ」でスプールディレクトリを引き継いでコンテナを作成した場合、引き継いだスプールディレクトリで設定されているサイト名と同じ値を、コンテナホスト名に指定してください。 ■コンテナ作成後、2分ほど経過してから停止した場合 MGが名前解決に失敗しています。 Docker内での通信を行う場合、--hostnameで指定するコンテナホスト名に、<コンテナ名>.<ネットワーク名>を指定してください。

2.6.1.1.3. MGへのログインに失敗する

以下の点を確認して、対処を行ってください。

1. JobCenterプロセスが起動しているか確認する

   コンテナに対して以下のコマンドを実行すると、コンテナで実行中のプロセス一覧が表示されます。

   ```
   docker container top <コンテナ名/コンテナID>
   ```

   

   JobCenterのプロセス一覧は、<スタンダードモード用環境構築ガイド>の表19.3「JobCenter常駐プロセス一覧（UNIX）」を参照してください。

   ■一部のプロセスが起動していない場合

   コンテナの起動直後は、JobCenterのプロセスの起動が完了していません。

   時間をおいて再度プロセスを確認し、全てのプロセスが起動していることを確認したのち、ログインを試みてください。

2. ポートが開放されているか確認する

   コンテナに対して以下のコマンドを実行すると、コンテナのポートマッピングが表示されます。

   ```
   docker container port <コンテナ名/コンテナID>
   ```

   10012/tcp, 611/tcp, 23116/tcp, 23180/tcp、23141/tcp, 23151/tcpが、Dockerホスト上のIPアドレスの同じポート番号とマッピングされていない場合、「2.4.4.1.12.1 コンテナのポートをDockerホストに割り当て」を参照して、ポートの開放を行ってください。

   

   「2.4.4.1.7 JobCenterが使用するプロトコルのポート番号の変更」でポート番号を変更した場合は、変更したポート番号を開放してください。

3. Docker内の通信を使って接続している場合、コンテナ同士が同一のDockerネットワークに接続しているか確認する

   以下のコマンドを実行すると、コンテナが接続しているDockerネットワーク名が表示されます。

   ```
   docker container inspect --format='{{range $p, $conf := .NetworkSettings.Networks}}{{$p}}
    {{end}}' <コンテナ名/コンテナID>
   ```

   コンテナ同士が同じDockerネットワークに接続していない場合、「2.4.4.2.3 異なるDockerネットワークに接続しているコンテナ同士の通信」を参照して、それぞれのコンテナを同じDockerネットワークに接続してください。

   

   Dockerネットワークは「2.4.2 Dockerネットワークの作成」で作成したものを使用する必要があります。デフォルトで使用されるbridgeネットワークでは、Dockerの内蔵DNSサーバを利用できないため、名前解決が行えません。

4. Docker外との通信を使って接続している場合、名前解決の設定を確認する

   コンテナに対する名前解決の設定を確認します。

   DockerホストのIPアドレスと、コンテナホスト名で、MGコンテナが名前解決されるよう設定してください。

2.6.1.1.4. マシン一覧へMGが追加できない

追加するマシンがコンテナ環境か物理環境かで確認観点が異なります。以下の点を確認して、対処を行ってください。

■コンテナ環境のMGを追加する場合（Docker内の通信を利用している場合）

コンテナ同士が同じDockerネットワークに接続しているか確認します。

以下のコマンドを実行すると、コンテナが接続しているDockerネットワーク名が表示されます。

```
docker container inspect --format='{{range $p, $conf := .NetworkSettings.Networks}}{{$p}}
 {{end}}' <コンテナ名/コンテナID>
```

コンテナ同士が同じDockerネットワークに接続していない場合、「2.4.4.2.3　異なるDockerネットワークに接続しているコンテナ同士の通信」を参照して、それぞれのコンテナを同じDockerネットワークに接続してください。

Dockerネットワークは「2.4.2　Dockerネットワークの作成」で作成したものを使用する必要があります。デフォルトで使用されるbridgeネットワークでは、Dockerの内蔵DNSサーバを利用できないため、名前解決が行えません。

■物理環境のMGを追加する場合（Docker外の通信を利用している場合）

名前解決の設定を確認します。

以下のコマンドを実行すると、コンテナが使用するDNSサーバのアドレスが表示されます。

```
docker container inspect --format="{{ .HostConfig.Dns }}" <コンテナ名/コンテナID>
```

DNSサーバのアドレスが間違っている場合、「2.4.4.1.12.2　名前解決方法の設定」でDNSサーバのアドレスを正しく指定してコンテナを作成しなおしてください。

コマンドを実行して何も表示されていない場合は、Dockerホストの/etc/resolv.confで設定されているDNSサーバを使用します。

また、以下のコマンドを実行すると、コンテナ内のhostsファイルの設定が表示されます。

```
docker container inspect --format='{{.HostConfig.ExtraHosts}}' <コンテナ名/コンテナID>
```

誤った設定が記載されている場合は、正しい設定を指定してコンテナを作成しなおしてください。

2.6.1.1.5. リモート（AG）投入したジョブがエラーになる

以下の点を確認して、対処を行ってください。

1. コンテナとDockerホストで、同じポート番号がマッピングされているか確認する

   コンテナに対して以下のコマンドを実行すると、コンテナのポートマッピングが表示されます。

   ```
   docker container port <コンテナ名/コンテナID>
   ```

   10012/tcp,　611/tcp,　23151/tcpが、Dockerホスト上のIPアドレスの同じポート番号とマッピングされていない場合、「2.4.4.1.12.1 コンテナのポートをDockerホストに割り当て」を参照して、ポートの開放を行ってください。

「2.4.4.1.7 JobCenterが使用するプロトコルのポート番号の変更」でポート番号を変更した場合は、変更したポート番号を開放してください。

2.6.1.1.6. リモート(AG) 投入したジョブがSUBMIT状態のまま進行しない

AGからコンテナMGへの名前解決が正しく行われているか確認します。

「2.4.4.1.11.3　コンテナホスト名の設定」で指定したコンテナホスト名で、名前解決の正引き逆引きが行われるよう設定してください。

2.6.1.1.7. フローが進行しない

以下の点を確認して、対処を行ってください。

1. キューが停止していないか確認する

   CL/Winのマネージャールームのキュー一覧画面を確認し、使用するキューが停止状態になっている場合、開始状態にします。

2. プロセスが実行中のままかどうか確認する

   単位ジョブが想定の実行時間を超えても実行中のままの場合、単位ジョブから起動したアプリケーションのプロセスが終了しているか確認してください。

   コンテナに対して以下のコマンドを実行すると、コンテナで実行中のプロセス一覧が表示されます。

   ```
   docker container top <コンテナ名/コンテナID>
   ```

   単位ジョブから起動したプロセスが実行中の場合、プロセスの終了を待つか、単位ジョブのスキップ、強制停止、コントロール解除の操作を行ってください。単位ジョブ操作の詳細については<スタンダードモード用基本操作ガイド>の「8.17.1 単位ジョブトラッカアイコンの操作」を参照してください。

2.6.2. 障害発生時の情報採取方法

コンテナ環境で障害が発生した場合、原因究明に必要な一次情報を漏れなく採取するために、コマンドを使用します。

次の手順を実施して、採取したデータをサポート窓口へ送付してください。

1. 「2.6.2.1 Dockerホストの情報採取」
2. 「2.6.2.2 停止しているコンテナの起動」
3. 「2.6.2.3 コンテナ内の情報採取」

2.6.2.1. Dockerホストの情報採取

Dockerホストの情報採取にはjc_docker_getinfo.shを利用します。

jc_docker_getinfo.shを利用するため、podmanの場合に最新版を利用してください。

Dockerホスト上で以下の手順を実施して、情報採取を行ってください。

1. NECサポートポータルのダウンロード、またはNECカスタマーサポートセンターから、jc_docker_getinfo.shファイルを入手する
2. jc_docker_getinfo.shに実行権を付与する

```
chmod a+x jc_docker_getinfo.sh
```

3. Dockerホストのrootユーザで、jc_docker_getinfo.shを実行する

4. 以下のデータファイルが作成されていることを確認する

```
jc_docker_data_<MMDDhhmm>_<Dockerホストのホスト名>.tar.gz
```

jc_docker_getinfo.shで情報採取を行うためには、コンテナに以下のラベルが設定されている必要があります。

■MG

com.nec.name=JobCenter MG/SV

ラベルが設定されていないJobCenterのコンテナで情報採取を行う場合、「2.6.2.4 Dockerホストの情報を手動で採取する」の手順を実施してください。

2.6.2.2. 停止しているコンテナの起動

コンテナ内の情報採取を行うためには、コンテナが起動状態である必要があります。障害によってコンテナが停止し、起動できない状態になった場合は、以下の手順を実施して起動状態のコンテナを作成します。

コンテナが起動している場合は、本手順は省略して「2.6.2.3 コンテナ内の情報採取」を行ってください。

1. 停止しているコンテナに対して以下のコマンドを実行し、コンテナをイメージ化する

```
docker container commit <コンテナ名/コンテナID> <作成するイメージ名>
```

2. 作成したイメージに対して以下のコマンドを実行し、コンテナを作成/起動する

```
docker container run --entrypoint=bash -it <作成したイメージ名>
```

2.6.2.3. コンテナ内の情報採取

MGの情報採取にはjc_getinfoコマンドを利用します。

Dockerホスト上で以下の手順を実施して、情報採取を行ってください。

■MGの場合

1. 情報採取を行うコンテナに対して以下のコマンドを実行し、jc_getinfoを実行する

```
docker container exec -u root <コンテナ名/コンテナID> /usr/lib/nqs/check/jc_getinfo
```

2. 実行時に表示された以下のメッセージを確認し、採取したデータのファイル名を確認する

```
Create "jcdata_<MMDDhhmmss>_<コンテナホスト名>.tar.gz"
```

3. 以下のコマンドを実行し、コンテナ内に存在する採取したデータを、Dockerホストにコピーする

```
docker container cp <コンテナ名/コンテナID>:/<採取したファイル名> <データをコピーするDockerホストのパス>
```

jc_getinfoについては、<スタンダードモード用コマンドリファレンス>の「8.1 jc_getinfo JobCenterの障害発生時、原因究明に必要な1次情報を漏れなく採取」、「8.2 clweb_getinfo CL/Webサーバの障害発生時、原因究明に必要な1次情報を漏れなく採取」を参照してください。

2.6.2.4. Dockerホストの情報を手動で採取する

jc_docker_getinfo.shによる情報採取が行えない場合、手動でデータを採取する必要があります。

採取したデータを格納するためのディレクトリを作成した後、以下の手順を実施して、データの採取を行ってください。

■「2.6.2.4.1 コンテナ情報の採取」

■「2.6.2.4.2 コンテナ内のファイルのコピー」（コンテナが停止している場合のみ実施）

■「2.6.2.4.3 Dockerネットワーク情報の採取」

■「2.6.2.4.4 マウント情報の採取」

■「2.6.2.4.5 マウントしているディレクトリ内のファイルリストの採取」

2.6.2.4.1. コンテナ情報の採取

情報を採取するコンテナに対して以下のコマンドを実行し、コンテナの情報をcontainer.infoに出力します。

```
docker container inspect <コンテナ名/コンテナID> >> <データ格納先のディレクトリ>/container.info
```

```
docker container logs <コンテナ名/コンテナID> >> <データ格納先のディレクトリ>/container.info
```

```
docker container top <コンテナ名/コンテナID> >> <データ格納先のディレクトリ>/container.info
```

```
docker container stats --no-stream <コンテナ名/コンテナID> >> <データ格納先のディレクトリ>/
container.info
```

```
docker events --filter container=<コンテナ名/コンテナID> --until `date -u "+%FT%TZ"` >> <データ格納
先のディレクトリ>/container.info
```

2.6.2.4.2. コンテナ内のファイルのコピー

コンテナが停止している場合、以下のコマンドを実行してコンテナ内のファイルをコピーします。

```
docker container cp <コンテナ名/コンテナID>:<コピーするファイル> <データ格納先のディレクトリ>
```

コピーするファイルは表2.10「コピーするファイル一覧」を参照してください。

表2.10 コピーするファイル一覧

コピーするファイル	MG
/etc/hosts	○
/etc/resolv.conf	○
/etc/nsswitch.conf	○
/etc/services	○
/usr/lib/nqs/rc	○
/usr/spool/nqs/log	○

（凡例）○:コピーが必要　×:コピーが不要

2.6.2.4.3. Dockerネットワーク情報の採取

情報を採取するコンテナに対して以下のコマンドを実行し、Dockerネットワークの情報をnetwork.infoに出力します。

```
docker network inspect <ネットワークID> > <データ格納先のディレクトリ>/network_<ネットワークID>.info
```

ネットワークIDは以下のコマンドで確認できます。

```
docker container inspect -f "{{range .NetworkSettings.Networks}}{{.NetworkID}} {{end}}" <コンテナ名/コンテナID>
```

コンテナが複数のDockerネットワークに接続している場合、<ネットワークID> <ネットワークID>という形式で複数表示されます。全てのネットワークに対してdocker network inspectコマンドを実施して、情報採取を行ってください。

2.6.2.4.4. マウント情報の採取

情報を採取するコンテナに対して以下のコマンドを実行し、コンテナのマウント情報をmount.infoに出力します。

```
docker container inspect -f "{{range .Mounts}}{{.Source}}:{{.Destination}}$(echo -en "\n\b") {{end}}" <コンテナ名/コンテナID> >> <データ格納先のディレクトリ>/mount.info
```

2.6.2.4.5. マウントしているディレクトリ内のファイルリストの採取

情報を採取するコンテナに対して以下のコマンドを実行し、コンテナのマウント情報を表示します。

```
docker container inspect -f "{{range .Mounts}}{{.Source}}:{{.Destination}}$(echo -en "\n\b") {{end}}" <コンテナ名/コンテナID>
```

マウント情報は、<マウント元のパス>:<マウント先のパス>の形式で表示されます。

マウント元にディレクトリを指定している場合、以下のコマンドを実行してディレクトリ内のファイル一覧の情報をls_<ディレクトリパス>_infoに出力します。

```
ls -alR <マウント元のパス> > <データ格納先のディレクトリ>/ls_<ディレクトリパス>.info
```

ディレクトリパスには、パスの区切り文字にアンダースコア（_）を利用したパス名を指定してください。

例）マウント元のディレクトリが/usr/spool/nqsの場合、出力先のファイル名はls_usr_spool_nqs.infoになります。

2.6.2.5. jc_docker_getinfo.sh

コンテナの情報採取のために利用する、jc_docker_getinfo.shについて説明します。

本スクリプトファイルは、NECサポートポータルのダウンロードまたはNECカスタマーサポートセンタより入手してください。

2.6.2.5.1. 機能説明

コンテナ環境のJobCenterの障害発生時、本スクリプトを実行することによって、原因究明に必要な情報が自動的に採取されます。

Dockerホストの情報およびMGのコンテナの情報を採取します。なお、MGのコンテナは、以下のラベルによって判断します。

■MG

com.nec.name=JobCenter MG/SV

スクリプトを実行すると、実行したディレクトリの直下に"jc_docker_data_<MMDDhhmm>_<Dockerホストのホスト名>.tar.gz"が作成されます。

2.6.2.5.2. 注意事項

■本スクリプトファイルを実行するにはdockerコマンドが必要です。

■本スクリプトファイルはrootユーザで実行してください。また、スクリプトファイルが存在するディレクトリ内で実行してください。

■com.nec.nameにJobCenter製品名のラベルが設定されていないコンテナについては、本スクリプトで情報採取を行いません。ラベルが設定されていないコンテナの情報採取を行う場合、「2.6.2.4　Dockerホストの情報を手動で採取する」を実施してください。

2.6.2.5.3. 主要メッセージ

スクリプト実行時に出力されるメッセージについては、以下を参照してください。

■情報採取に関するメッセージ

メッセージ	説明
Start to get docker info.	Dockerデーモンの情報収集を開始します。
Start to copy system files.	システムファイルのコピーを開始します。
Start to get net info.	Dockerホストのネットワーク情報の収集を開始します。
Start to get system info.	Dockerホストのシステム情報の収集を開始します。
Start to get container(<コンテナID>) info.	コンテナの情報収集を開始します。
Start to copy mgsv files from container(<コンテナID>).	MGコンテナからファイルのコピーを開始します。
Start to get container(<コンテナID>) network info.	コンテナのDockerネットワーク情報の収集を開始します。
Start to get container(<コンテナID>) mount info.	コンテナのマウント情報の収集を開始します。
Start to compress.	収集したデータの圧縮を開始します。
The Info file "<収集データの圧縮ファイル名>" was created successfully.	データの収集に成功しました。

■エラーメッセージ

エラーメッセージ	考えられるエラーの原因と対処方法
jc_docker_getinfo.sh: Only root user can execute this script.	root以外のユーザが本スクリプトを実行しています。 本スクリプトはrootユーザで実行してください。
docker version error. Get docker info error! Stop.	docker versionコマンドが異常終了しました。 dockerコマンドのパスが通っていること、Dockerデーモンが起動していることを確認してください。
docker info error. Get docker info error! Stop.	docker infoコマンドが異常終了しました。 dockerコマンドのパスが通っていること、Dockerデーモンが起動していることを確認してください。
tar error. Failed to compress the files! Stop.	tarコマンドによる収集データの作成が異常終了しました。 tarコマンドのパスが通っていること、収集データの作成先のディスク容量およびクォータを確認してください。
gzip error. Failed to compress the files! Stop.	gzipコマンドによる収集データの圧縮が異常終了しました。 gzipコマンドのパスが通っていること、収集データの作成先のディスク容量およびクォータを確認してください。

tarコマンドおよびgzipコマンドに失敗した場合、作成された jc_docker_data_<MMDDhhmm>_<Dockerホストのホスト名>ディレクトリ内のデータを採取してください。採取後、ディレクトリは削除しても構いません。

3. AWS ECS on Fargate環境での運用

AWS Elastic Container Service on AWS Fargate(以下、ECS on Fargateと記します)は、AWSが提供するサーバレスコンピューティングサービスです。 このサービスにより、開発者はサーバのセットアップや管理に時間を費やすことなく、コンテナ化されたJobCenterを簡単にデプロイし、運用することができます。

本章ではECS on Fargate上でJobCenter MGコンテナを運用する方法について説明します。

3.1. 概要

3.1.1. ECS on Fargate上でMGコンテナ運用概要

ECS on Fargate上でMGコンテナを運用する場合、以下のステップで構築・運用を行います。

■事前準備

ECS on Fargateを利用してMGコンテナを運用する事前準備として、環境設計と必要なAWSリソースの作成を行います。

詳細説明は「3.1.2 事前準備」を参照してください。

■イメージの作成

ECS on Fargate上のMGコンテナ用のイメージを作成します。

イメージ作成詳細は「3.2 イメージの作成」を参照してください。

■ECS on Fargate上のMGコンテナ環境構築

- タスク定義作成

  ECS on Fargate上でMGコンテナを運用するにあたり、コンテナMGの設定および要件を明確に指定するため、設定ファイルである「タスク定義」の作成を行います。

  タスク定義の作成詳細は「3.3.2 タスク定義を作成」を参照してください。

- サービス設定と起動

  ECS on Fargate上でコンテナMGを運用するため、タスク/サービスを起動します。

  サービス設定と起動の詳細は「3.3.3 サービスの起動」を参照してください。

■ECS on Fargate環境の運用

- サービスの管理

  ECS on Fargate上のコンテナMGのサービスを更新/削除をします。

  サービスの更新と削除詳細は「3.4.1 サービスの管理」を参照してください。

- デバッグ方法

  MGコンテナをデバッグするため、MGコンテナを接続します。

  デバッグ方法の詳細は「3.4.2 コンテナ内でのコマンド実行」を参照してください。

■ECS on Fargateコンテナ環境のトラブルシューティング

ECS on Fargate上のMGコンテナのトラブルシューティングを実施します。

詳細は「3.5 ECS on Fargateコンテナ環境のトラブルシューティング」を参照してください。

3.1.2. 事前準備

ECS on Fargate上でMGコンテナを運用するために、必要な事前準備について説明します。

■環境設計

ECS on Fargate上で動作させるMGコンテナと、他コンポーネント(AGやCL/Win)との通信について整理して必要な通信が行えるよう環境設計を行います。他コンポーネントをVPC内に配置するかAWS外部に配置するかに応じて必要な設定は異なります。

また、MGのスプール領域の永続化に利用するストレージが必要となります。ECS on Fargate上のMGコンテナから利用できるストレージとしてはEFSのみとなります。

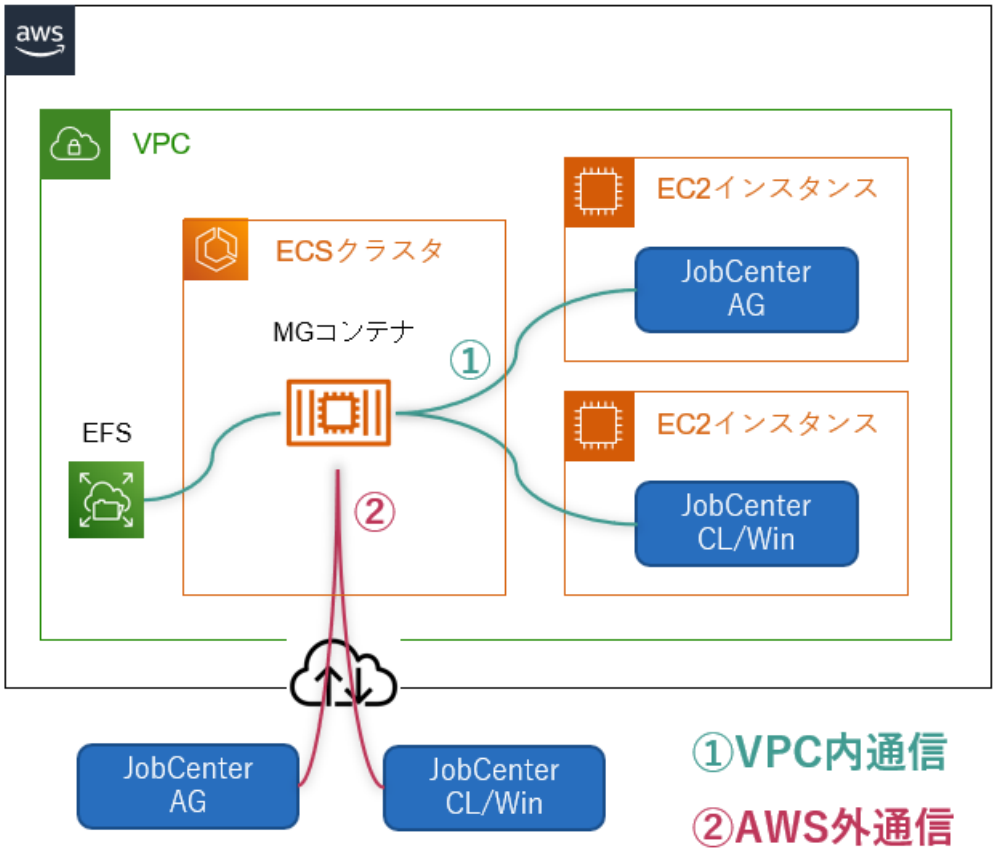

図3.1 MGコンテナとの通信

図ではVPC以外を記載していませんが、必要に応じて必要なネットワークリソースを配置してください。

■VPCおよび関連リソースの作成

環境設計にもとづいて、コンテナとの通信に利用するためのネットワーク設定や永続化用のストレージ設定を事前に実施してください。VPC以外に必要なネットワークリソース(NATゲートウェイやVPCエンドポイントなど)はネットワーク構成に応じて適宜作成してください。

本書ではVPCおよびその他のリソースの作成や設定については解説しないため、詳細はAWS公式ドキュメントを参照してください。

■ECSクラスターの作成

AWS公式ドキュメントを参照してECSクラスターを適宜作成してください。

3.2. イメージの作成

ECS on Fargate上で実行するMGコンテナのイメージを作成する手順について説明します。

3.2.1. 概要

以下の手順に従い、イメージをビルドすることが可能です。

■事前準備

イメージビルド用のパッケージとDockerfileを事前に用意します。必要なパケージとDockerfile詳細は「3.2.3 MGコンテナのイメージの作成」を参照してください。

■Dockerfileの作成

JobCenterが提供しているDockerfileをベースにDockerfileを作成します。 詳細手順は「3.2.2 Dockerfileの作成」を参照してください。

■イメージのビルド

イメージをビルドします。詳細手順は「3.2.3 MGコンテナのイメージの作成」を参照してください。

3.2.2. Dockerfileの作成

Dockerfileに対して、以下の修正を行います。

■パッケージ/パッチの設定、デーモン設定ファイルの固定化

「2.3.2.1 Dockerfileの作成」を参照して、設定してください。

■コンテナ内へのMGのライセンス登録

- コードワードが記載された.lockinfoファイルを用意します。

  コードワードの登録に関しては<セットアップガイド>の「2.4　コードワードを登録する」を参照してください。

- .lockinfoを/etc/opt/wsnlesd/.lockinfoのディレクトリにCOPYします。

  DockerfileのWORKDIR / 行からCMD ["/usr/lib/nqs/cluster/cjcpw", "-local"] 行の間に、以下を記載します。

```
COPY <コードワードが登録された.lockinfoファイル> </etc/opt/wsnlesd/.lockinfo>
```

■パスワードファイルの登録

一般ユーザを作成する場合、パスワードファイルを登録します。

- パスワードファイルを作成します。

  パスワードファイルの作成に関しては「2.4.4.1.8.1 パスワードファイルの作成」 を参照してください。

- パスワードファイルをコンテナ内部の任意ディレクトリにCOPYします。

  DockerfileのWORKDIR / 行からCMD ["/usr/lib/nqs/cluster/cjcpw", "-local"] 行の間に、以下を記載します。

```
COPY <パスワードファイル> <任意ディレクトリ>
```

3.2.3. MGコンテナのイメージの作成

以下の手順に従って、Dockerfileからイメージのビルドを行います。

1. 作業ディレクトリを作成し、以下のファイルをディレクトリ内に配置します。

   ■Dockerfile

   ■entrypoint_ecs_awsvpc.sh (entrypoint.sh に名前を変更してください)

   ■MGのパッケージ

   ■LicenseManagerのパッケージ

   ■コードワードが登録された.lockinfoファイル

   ■（パッチを適用する場合）MGのパッチ

   ■（一般ユーザ作成する場合）パスワードファイル

   ■（デーモン設定ファイルの設定を固定化する場合）デーモン設定ファイル

   

   リポジトリ上のパッケージを利用する場合は、ディレクトリ内にパッケージを配置する必要はありません。

2. ディレクトリ内で以下のコマンドを実行し、イメージのビルドを行います。

```
docker image build -t <作成するイメージ名> .
```

3. 以下のコマンドを実行し、イメージの一覧を表示します。

```
docker image ls
```

   <作成するイメージ名>で指定した名前のイメージが存在することを確認してください。

4. 作成したイメージは、Amazon Elastic Container Registry（以下はECRと表します）などのECSから参照できるリポジトリへ格納してください。

イメージの作成における注意事項は「2.3.3 イメージの作成における注意事項」を参照してください。

3.3. ECS on Fargateコンテナ環境の構築

3.3.1. 概要

以下の手順に従い、MGコンテナを構築することが可能です。

■タスク定義のJSONファイルの作成

タスク定義パラメータ説明は表3.1「MGコンテナで利用するタスク定義パラメーター一覧」を参照してください。 タスク定義の作成例は「3.3.2.3 タスク定義の登録と管理」を参照してください。

■タスク定義の登録

タスク定義をAWS上に登録します。 詳細手順は「3.3.2.3 タスク定義の登録と管理」を参照してください。

■サービス起動

サービスを起動します。詳細手順は「3.3.3 サービスの起動」を参照してください。

3.3.2. タスク定義を作成

タスク定義とは、Amazon　ECSでコンテナ化されたアプリケーションを実行するための設定を記述するテンプレートです。　タスク定義には、アプリケーションを構成するコンテナのセットアップ情報が含まれており、使用するコンテナイメージ、コンテナへのCPUやメモリの割り当て、　ネットワーク設定、ボリュームのマウント、環境変数、実行ロールなど、タスク（コンテナのグループ）を実行するために必要なすべての詳細を指定します。

本節ではMGコンテナを実行するためのタスク定義について説明します。

3.3.2.1. タスク定義のパラメータ

タスク定義で設定するパラメータを紹介します。タスク定義の作成方法はJSONを利用した方式を前提としています。

パラメータの詳細についてはAWS公式ドキュメントを参照してください。

表3.1 MGコンテナで利用するタスク定義パラメーター一覧

パラメータ分類	パラメータ名	型	必須	説明
ファミリー	family	文字列	◎	タスク定義の名前を指定します。 「family": "名前"」の形式で指定します。
起動タイプ	requiresCompatibilities	文字列	◎	起動タイプを「FARGATE」に指定してください。 「"requiresCompatibilities": ["FARGATE"]」の形式で指定します。
タスクロール	taskRoleArn	文字列	○	コンテナ内のアプリケーションからAWSのサービスを利用する場合に必要な権限を持ったロールを指定します。 「"taskRoleArn": "タスクロールのARN"」の形式で指定します。
タスク実行ロール	executionRoleArn	文字列	○	タスクを起動時に利用するIAMロールを指定します。例えば、ECRの読み込み権限やAmazon CloudWatch Logsへの書き込み権限を与えます。 「"executionRoleArn": "タスク実行ロールのARN"」の形式で指定します。

パラメータ分類	パラメータ名	型	必須	説明
ネットワークモード	networkMode	文字列	◎	Fargateではawsvpcモードのみがサポートされていますので「awsvpc」を指定してください。 「"networkMode": "awsvpc"」の形式で指定します。
ランタイムプラットフォーム	operatingSystemFamily	文字列	◎	「LINUX」を指定してください。 「"operatingSystemFamily": "LINUX"」の形式で指定します。
	cpuArchitecture	文字列	◎（	「X86_64」を指定してください。 「"cpuArchitecture": "X86_64"」の形式で指定します。
タスクサイズ	cpu	文字列	◎	タスクに割り当てるCPU単位を指定してください。 「"cpu": "割り当てるCPU単位"」の形式で指定します。
	memory	文字列	◎	タスクに割り当てるメモリ単位を指定してください。 「"memory": "割り当てるメモリ単位"」の形式で指定します。
コンテナ定義	name	文字列	◎	コンテナの名前を指定します。 "name": "コンテナ名"」の形式で指定します。
	image	文字列	◎	コンテナの開始に使用するイメージを指定します。作成したMGコンテナのイメージを指定してください。 「"image": "registry/repository:tag"」の形式で指定します。
	memory	整数	○	コンテナに適用されるメモリの量 (MiB 単位)を指定します。JobCenter MGの動作要件を満たすよう指定してください。 「"memory": メモリの量」の方法で指定します。
	portMappings	オブジェクト配列	◎	MG通信用のポートの開放設定を記載します。 以下ように配列形式で指定します: `"portMappings": [ { "name": "名称", "containerPort": ポート番号, "hostPort": containerPortと同じポート番号, "protocol": "tcp" }, ... ]`

パラメータ分類	パラメータ名	型	必須	説明

	environment	オブジェクト配列	◎	環境変数を指定します。
				
				指定可能な環境変数は表2.6「JobCenterのセットアップ時の設定を行う環境変数」を参照してください。"
				
				 
				必ず環境変数「JOBCENTER_HOSTNAME」にMGコンテナのホスト名を指定してください。
				
				以下のように配列形式で指定します:
				
				`"environment" :`
				`  [`
				`  {`
				`  "name" : "環境変数名",`
				`  "value" : "環境変数値"`
				`  }, ...`
				`  ]`

	mountPoints	オブジェクト配列	◎	コンテナでのデータボリュームのマウントポイントを指定します。
				
				以下の形式で指定します:
				
				`"mountPoints":`
				`  [`
				`  {`
				`  "sourceVolume": "volumesで指定したボリューム名",`
				`  "containerPath": "spoolディレクトリのパス"`
				`  }`
				`  ]`

	volumes	オブジェクト配列	◎	EFSのファイルシステムボリュームを使用するための設定です。
				
				以下の形式で指定してください：
				
				`"volumes":`
				`  [`
				`  {`
				`  "name": "ボリュームの名前",`
				`  "efsVolumeConfiguration":`
				`  {`
				`  "fileSystemId": "EFSのファイルシステムID",`
				`  "rootDirectory": "ホスト内にルートディレクトリとしてマウントするEFS内のディレクトリ",`

パラメータ分類	パラメータ名	型	必須	説明
				"transitEncryption": "ENABLED または DISABLED（データ暗号化するかどうかを指定します)" } }]

	logConfiguration	オブジェクト配列	○	コンテナに対するログ構成の仕様を設定します。
				
				ECSサポートされているログドライバーはいくつありますがここでは「awslogs」を例として指定方法を説明します。 サポートされているログドライバーのオプションの詳細についてAWS公式ドキュメントを参照してください。
				
				awslogsログドライバーを利用する場合、以下の形式で指定します:
				"logConfiguration":
				{
				"logDriver": "awslogs",
				"options":
				{
				"awslogs-create-group": "true または falseを指定します。trueの場合、指定したロググループが存在しない場合に自動的に作成されます。",
				"awslogs-group": "ログを送信するCloudWatch Logsのロググループ名を指定します",
				"awslogs-region": "ログを送信するリージョンを指定します",
				"awslogs-stream-prefix": "ログストリームの名前にプレフィックスを付けるために使用します"
				}
				}
				]

Linux パラメータ	initProcessEnabled	ブール	◎	trueを指定します。 「"linuxParameters": { "initProcessEnabled": true }」の形式で指定します。

（凡例）◎：必ず利用する設定, ○：必要に応じて利用する設定

3.3.2.2. タスク定義の作成例

以下はタスク定義の例です。ECR上のリポジトリに登録したコンテナイメージを指定しています。

```
{
  "family": "mg-task-definition",
  "containerDefinitions": [
      {
          "name": "mg-container",
          "image": "XXX.dkr.ecr.ap-northeast-1.amazonaws.com/jobcenter-mg:latest",
```

```
        "cpu": 2048,
        "memory": 4096,
        "portMappings": [
            {
                "name": "jccombase-port",
                "containerPort": 611,
                "hostPort": 611,
                "protocol": "tcp"
            },
            {
                "name": "jccombase-ssl-port",
                "containerPort": 23116,
                "hostPort": 23116,
                "protocol": "tcp"
            },
            {
                "name": "jcevevt-port",
                "containerPort": 10012,
                "hostPort": 10012,
                "protocol": "tcp"
            },
            {
                "name": "jcwebserver-port",
                "containerPort": 23180,
                "hostPort": 23180,
                "protocol": "tcp"
            },
            {
                "name": "jcnats-port",
                "containerPort": 23141,
                "hostPort": 23141,
                "protocol": "tcp"
            },
            {
                "name": "jcexecutor-port",
                "containerPort": 23151,
                "hostPort": 23151,
                "protocol": "tcp"
            }
        ],
        "environment": [
            {
                "name": "JOBCENTER_NSUMSMGR_PASS",
                "value": "testpwd"
            },
            {
                "name": "JOBCENTER_HOSTNAME",
                "value": "test-mg"
            },
            {
                "name": "JOBCENTER_USERS_PASS_FILE",
                "value": "/home/passwdFile"
            },
            {
                "name": "JOBCENTER_MACHINE_ID",
                "value": "1000"
```

```
            },
            {
                "name": "JOBCENTER_LANGUAGE_CODE",
                "value": "UTF-8"
            }
        ],
        "mountPoints": [
            {
                "sourceVolume": "jcData",
                "containerPath": "/usr/spool/nqs"
            }
        ],
        "volumesFrom": [],
        "linuxParameters": {
            "initProcessEnabled": true
        },
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-create-group": "true",
                "awslogs-group": "/ecs/mgContainerGroup",
                "awslogs-region": "ap-northeast-1",
                "awslogs-stream-prefix": "ecs"
            }
        }
    }
  ],
  "taskRoleArn": "arn:aws:iam::XXX:role/ecsTaskRole",
  "executionRoleArn": "arn:aws:iam::XXX:role/ecsTaskExecutionRole",
  "networkMode": "awsvpc",
  "volumes": [
      {
          "name": "jcData",
          "efsVolumeConfiguration": {
              "fileSystemId": "fs-XXX",
              "rootDirectory": "/jcdata",
              "transitEncryption": "ENABLED"
          }
      }
  ],
  "requiresCompatibilities": [
      "FARGATE"
  ],
  "cpu": "4096",
  "memory": "16384",
  "runtimePlatform": {
      "cpuArchitecture": "X86_64",
      "operatingSystemFamily": "LINUX"
  }
}
```

3.3.2.3. タスク定義の登録と管理

タスク定義の管理にはAWSコンソールまたはAWSコマンドラインインタフェースを利用する方法があります。
以下ではAWSのコマンドラインインタフェースを利用したタスク定義管理のコマンドライン例を説明します。

3.3.2.3.1. タスク定義の登録

以下のコマンドを実行して、タスク定義を登録します。

```
aws ecs register-task-definition --cli-input-json file://<タスク定義のJSONファイルのパス>
```

3.3.2.3.2. タスク定義をリスト表示します。

登録したタスク定義を以下のコマンドで、リスト表示しますす。

```
aws ecs list-task-definitions
```

3.3.2.3.3. タスク定義の更新

タスク定義のJSONファイルを編集して、以下のコマンドを実行して、タスク定義を更新します。

```
aws ecs register-task-definition --cli-input-json file://<タスク定義のJSONファイルのパス>
```

3.3.2.3.4. タスク定義の削除

タスク定義の特定のリビジョンの削除は、以下のコマンドを実行して、実現できます。

```
aws ecs deregister-task-definition --task-definition <タスク定義の特定のリビジョンのARN>
```

3.3.3. サービスの起動

登録したタスク定義を利用してサービスを起動するためにはサービスを作成する必要があります。 AWSコマンドラインインタフェース(AWS CLI)でサービスを作成して起動する場合、以下のように指定します。

```
aws ecs create-service \
  --cluster 事前に作成したECSのクラスター名 \
  --task-definition タスク定義名 \
  --enable-execute-command \
  --service-name 任意サービス名 \
  --desired-count  サービスが起動するタスクの数 \
  --launch-type FARGATE \
  --network-configuration "VPCとサブネットを指定"
```

以下はコマンドの実行例です。

```
aws ecs create-service \
    --cluster MGCluster \
    --task-definition mg-task-definition \
    --enable-execute-command \
    --service-name MGService \
    --desired-count 1 \
    --launch-type FARGATE \
```

```
    --network-configuration "awsvpcConfiguration={subnets=[subnet-XXX],securityGroups=[sg-
XXX],assignPublicIp=DISABLED}"
```

3.4. ECS on Fargateコンテナ環境の運用

3.4.1. サービスの管理

ECSではサービスに登録されたタスク定義にもとづいてコンテナが実行されます。

本節では作成したサービスの更新および削除手順について説明します。

3.4.1.1. サービスの更新

実行するコンテナイメージや各種設定を変更する場合はサービスのタスク定義を更新する必要があります。

サービスのタスク定義を更新する場合、以下のコマンドを実行します。

```
aws ecs update-service --cluster <ECSクラスター名> --service <サービス名> --task-definition <タスク定義名>
```

3.4.1.2. サービスの削除

コンテナが不要となった場合はサービスを削除するとコンテナも合わせて停止・削除されます。

サービスを削除する場合は、以下のコマンドを実行します。

```
aws ecs delete-service --cluster <ECSクラスター名> --service <サービス名> --force
```

3.4.2. コンテナ内でのコマンド実行

MGコンテナ内でコマンドを実行する場合、AWS CLI(aws ecs execute-command)を利用して実行してください。

コマンド実行を行う場合、サービス作成時に「--enable-execute-command」オプションを付けてコマンド実行機能を有効化する必要があります。　サービス作成については「3.3.3　サービスの起動」を参照してください。

以下は対話シェルを実行する場合のコマンド形式です。

```
aws ecs execute-command --cluster ECSのクラスター名 \
  --task タスクID \
  --container コンテナ名 \
  --interactive \
  --command "/bin/sh"
```

以下はコマンドの実行例です。

```
aws ecs execute-command --cluster MGCluster \
  --task arn:aws:ecs:ap-northeast-1XXX:task/MGCluster/1c89XXX \
  --container mg-container \
  --interactive \
  --command "/bin/sh"
```

3.5. ECS on Fargateコンテナ環境のトラブルシューティング

3.5.1. 情報採取

以下の手順に従って、コンテナ情報の採取を行います。

■MGコンテナを接続

「3.4.2 コンテナ内でのコマンド実行」を参照して、MGコンテナを接続します。

■接続したコンテナ上に情報採取コマンドjc_getinfoを実行

<スタンダードモード用コマンドリファレンス>の8章 「情報採取コマンド」 を参照して、情報採取コマンドjc_getinfoを実行します。

3.5.2. エラーメッセージ

エラーメッセージーの詳細は「2.6.1.1 MGコンテナにおける障害」を参照してください。

4. Amazon EKS環境での運用

本章ではAmazon Elastic Kubernetes Service(以下、EKSと記します)上でJobCenterコンテナを運用する方法について説明します。

本章の内容を実施する前に、EKSにて以下の作業を実施してください。

■EKSクラスターの作成

■EKSクラスター上にNodeGroupとNodeの作成

■Nodeが所属しているセキュリティーグループのインバウンドルール設定で、Kubernetesが利用するポート(デフォルトは30000 - 32767)の許可

4.1. 概要

4.1.1. AWS EKS上でJobCenterコンテナ運用概要

AWS EKSを利用してJobCenterコンテナを運用するため、以下のステップに従って、JobCenter MGコンテナをデプロイと管理できます。

■イメージ作成

AWS EKS上のMGコンテナ用のイメージを作成します。

イメージ作成詳細は「4.2 コンテナイメージの作成」を参照してください。

■環境構築

- マニフェストファイルの作成

  詳細は「4.3.1 MGのマニフェストファイルの作成」を参照してください。

- EKSリソースの作成とデプロイ

  詳細は「4.5 リソースの作成」を参照してください。

■AWS EKS環境の運用

- CL/Winとの接続

  詳細は「4.6.1 EKS上のMGへの接続（通常の接続）」を参照してください。

- AGとの接続

  詳細は「4.4.1.6 EKS外部のマシンとの接続」を参照してください。

- デバッグ方法

  詳細は「4.6.4 デバッグのためMGコンテナへの接続」を参照してください。

■AWS EKS環境のトラブルシューティング

- ログ収集

  詳細は「4.7.1 ログ収集」を参照してください。

- エラーメッセージの参照

  詳細は「4.7.2 エラーメッセージ一覧」を参照してください。

4.2. コンテナイメージの作成

EKSで利用するコンテナのイメージを作成します。

作成したイメージは、Amazon Elastic Container RegistryなどのEKSのノード上から参照できるリポジトリへ格納してください。

4.2.1. MGのイメージの作成

DockerfileからJobCenter MGコンテナのイメージを作成する手順を説明します。

1. ディレクトリを作成し、以下のファイルをディレクトリ内に配置します。

   ■Dockerfile

   ■entrypoint.sh

   ■JobCenter MGのパッケージ

   ■LicenseManagerのパッケージ

   ■（パッチを適用する場合）JobCenter MGのパッチ

   ■（一般ユーザ作成する場合）パスワードファイル

2. Dockerfileに対して、以下の修正を行います。

   ■ENTRYPOINTを以下の通り修正します。

   ```
   ENTRYPOINT ["/usr/bin/tini", "--", "/entrypoint.sh"]
   ```

   ■Dockerfileを修正しtiniをインストールする処理を加えます。

   ```
   # Use tini as subreaper in Docker container to reap zombie processes
   ARG TINI_VERSION=v0.18.0
   ARG TINI_NAME=tini-static
   ARG TINI_PATH=/usr/bin/tini
   ARG TINI_SHA=eadb9d6e2dc960655481d78a92d2c8bc021861045987ccd3e27c7eae5af0cf33
   ARG TINI_ASC_SHA=48223d0afcb4423ab0724589363aa00075d98bf2f82b2ede338619ff1df7b48d
   ADD https://github.com/krallin/tini/releases/download/${TINI_VERSION}/${TINI_NAME}
    ${TINI_PATH}
   ADD https://github.com/krallin/tini/releases/download/${TINI_VERSION}/${TINI_NAME}.asc
    ${TINI_PATH}.asc
   RUN echo "${TINI_SHA} ${TINI_PATH}" | sha256sum -c \
    && echo "${TINI_ASC_SHA} ${TINI_PATH}.asc" | sha256sum -c \
    && gpg --batch --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys
    595E85A6B1B4779EA4DAAEC70B588DFF0527A9B7 \
    && gpg --batch --verify ${TINI_PATH}.asc ${TINI_PATH} \
    && chmod +x ${TINI_PATH} \
    && rm -rf ${TINI_PATH}.asc
   ```

   

   tiniはコンテナ上でinitプロセスとして動作するアプリケーションです。JobCenterはinitプロセスが存在することを前提としたアプリケーションのため、追加が必要です。

3. Dockerfileからコンテナイメージをビルドします。手順については、「2.3.2.2　イメージのビルド」を参照してください。

4.3. マニフェストファイルの作成

Kubernetesのオブジェクトを作成するためのマニフェストファイルを作成します。

以下のオブジェクトごとにマニフェストファイルを作成します。

■Deployment

JobCenter　MGコンテナをデプロイするためのDeploymentを作成するためにマニフェストファイルを作成します。

■Service

JobCenter　MGコンテナが実行されるPodとの疎通のためのServiceを作成するためにマニフェストファイルを作成します。

■ConfigMap

JobCenter　MGコンテナのライセンス情報や設定ファイルを登録するためのConfigMapを作成するためにマニフェストファイルを作成します。

■Persistenct Volume Claim

JobCenter MGコンテナのデータを永続化するためのPersistenct Volume Claim(永続ボリューム要求, PVC)を設定するためにマニフェストファイルを作成します。

4.3.1. MGのマニフェストファイルの作成

4.3.1.1. マニフェストファイルの作成

JobCenter　MGコンテナをEKS上で実行するためのマニフェストファイルは、以下をベースに作成してください。

■Deployment のマニフェストファイル

```
apiVersion: apps/v1
  kind: Deployment
metadata:
  name: <name>
  labels:
    app: jobcenter
spec:
  replicas: 1
  selector:
    matchLabels:
      name: <name>
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: jobcenter
        name: <name>
      name: <name>
    spec:
      containers:
      - name: jobcenter
        imagePullPolicy: IfNotPresent
        image: <image>
        env:
        - name: JOBCENTER_MACHINE_ID
          value: "1000"
        - name: JOBCENTER_LANGUAGE_CODE
          value: "UTF-8"
        - name: JOBCENTER_NSUMSMGR_PASS
          value: "password"
        ports:
        - containerPort: 611
          protocol: TCP
        - containerPort: 10012
          protocol: TCP
        - containerPort: 23116
          protocol: TCP
        - containerPort: 23180
          protocol: TCP
        - containerPort: 23141
          protocol: TCP
        - containerPort: 23151
          protocol: TCP
        volumeMounts:
          - name: lockinfo
            mountPath: /etc/opt/wsnlesd/.lockinfo
            subPath: lockinfo
```

```
        - name: spool-pvc
          mountPath: /usr/spool/nqs
          subPath: <name>
    hostname: <hostname>
    volumes:
      - name: spool-pvc
        persistentVolumeClaim:
          claimName: <pvc name>-spool
      - name: lockinfo
        configMap:
          name: lockinfo
          defaultMode: 420
```

マニフェストファイルの以下のパラメータの値は変更しないでください。

- spec.replicas
- spec.strategy.type
- spec.template.spec.containers.ports

- <name>

  作成するオブジェクトの名前です。リソースのオブジェクト名に設定したい名前を設定してください。

  

  <name>の箇所には、すべて同一の値を設定してください。

- <hostname>

  作成するコンテナのホスト名です。MGのマシン名に設定したい名前を設定してください。

- <image>

  コンテナが使用するイメージです。「4.2.1　MGのイメージの作成」で作成したImageが配置されているリポジトリを設定してください。

- <pvc name>

  利用するPersitent Volumeのオブジェクトの名前です。Persistent Volume Claimのマニフェストファイルの<name>と同じ値を設定してください。

■Service のマニフェストファイル

```
apiVersion: v1
kind: Service
metadata:
  name: <name>
  labels:
    app: jobcenter
spec:
  ports:
  - name: jccombase
    port: 611
    protocol: TCP
  - name: jcevent
    port: 10012
    protocol: TCP
  - name: jccombase-over-ssl
    port: 23116
    protocol: TCP
  - name: jcwebserver
    port: 23180
    protocol: TCP
  - name: jcnats
    port: 23141
    protocol: TCP
  - name: jcexecutor
    port: 23151
    protocol: TCP
  type: ClusterIP
  selector:
    name: <deployment name>
```

マニフェストファイルの以下のパラメータの値は変更しないでください。

- spec.ports
- spec.type

- <name>

  作成するオブジェクトの名前です。リソースのオブジェクト名に設定したい名前を設定してください。

- <deployment name>

  Serviceを関連付けるオブジェクトの名前です。Deploymentのマニフェストファイルの<name>と同じ値を設定してください。

CL/Winの接続、WebコンソールやWebAPIへの接続、JobCenterエージェントの接続を行えるよう、以下のNodePortのServiceを作成するマニフェストファイルも作成してください。

```
apiVersion: v1
kind: Service
metadata:
  name: <name>-nodeport
  labels:
    app: jobcenter
spec:
  ports:
  - name: jccombase
    port: 611
    protocol: TCP
  - name: jccombase-over-ssl
    port: 23116
    protocol: TCP
  - name: jcwebserver
    port: 23180
    protocol: TCP
  - name: jcexecutor
    port: 23151
    protocol: TCP
  type: NodePort
  selector:
    name: <deployment name>
```

- マニフェストファイルの以下のパラメータの値は変更しないでください。
  - spec.ports
  - spec.type
- CL/Winの接続時に「保護された接続」の機能を使用しない場合には、上記のポート:23116　の設定は必要ありません。
- JobCenter　　MGのWebコンソール機能やWebAPI機能を利用しない場合には、上記のポート:23180 の設定は必要ありません。

- <name>

  作成するオブジェクトの名前です。リソースのオブジェクト名に設定したい名前を設定してください。

- <deployment name>

  Serviceを関連付けるオブジェクトの名前です。Deploymentのマニフェストの<name>と同じ値を設定してください。

■Persistent Volume Claim のマニフェストファイル

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <name>-spool
  labels:
    app: jobcenter
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```

マニフェストファイルの以下のパラメータの値は変更しないでください。

- spec.accessModes

- <name>

  作成するオブジェクトの名前です。リソースのオブジェクト名に設定したい名前を設定してください。

■ConfigMap のマニフェストファイル

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: lockinfo
data:
  lockinfo: |
    UL1256-A10 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
    UL1256-A00 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
    UL1256-A02 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

- data.lockinfo

  コンテナ内のMGのライセンスを解除するためのコードワードを設定してください。

上記マニフェストファイルで設定されているコードワードはサンプルです。ライセンス購入時に付与されたコードワードに置き換えてください。

4.4. マニフェストファイルの編集

マニフェストファイルのパラメータを編集することで、コンテナの設定を行います。

4.4.1. MGのマニフェストファイルの編集

マニフェストファイルへのパラメータの追加やConfigMapを作成することで、コンテナの以下の機能を利用できます。

機能	章
管理者ユーザのパスワード設定	「4.4.1.1 管理者ユーザのパスワードの設定」
マシンIDの設定	「4.4.1.2 マシンIDの設定」
一般ユーザの設定	「4.4.1.3 一般ユーザの作成」
ポート番号の変更	「4.4.1.4 ポート番号の変更」
セットアップ言語の設定	「4.4.1.5 セットアップ言語の設定」
EKS外部のマシンとの接続	「4.4.1.6 EKS外部のマシンとの接続」
起動時の設定ファイルの設定	「4.4.1.7 起動時の設定ファイルの設定」
証明書・秘密鍵ファイルの設定	「4.4.1.8 証明書・秘密鍵ファイルの設定」

4.4.1.1. 管理者ユーザのパスワードの設定

MGコンテナ内のJobCenter管理者ユーザ(nsumsmgr)のパスワードを設定できます。

以下の手順に従って、マニフェストファイルへパラメータを追加してください。

1. Deploymentのマニフェストファイルに以下を追加します。

■spec.template.spec.containers.env

```
        env:
        - name: JOBCENTER_NSUMSMGR_PASS
          value: <設定したいパスワード>
```

暗号化したパスワードを使用する場合、以下の手順に従って設定してください。

1. 「2.4.4.2.2 パスワードの暗号化」を参照して、設定したいパスワードを暗号化します。

2. Deploymentのマニフェストファイルに以下を追加します。

■spec.template.spec.containers.env

```
        env:
        - name: JOBCENTER_NSUMSMGR_CRYPTED_PASS
          value: <暗号化したパスワード>
```

4.4.1.2. マシンIDの設定

コンテナMGののマシンIDを設定できます。

以下の手順に従って、マニフェストファイルへパラメータを追加してください。

1. Deploymentのマニフェストファイルに以下を追加します。

■spec.template.spec.containers.env

```
        env:
        - name: JOBCENTER_MACHINE_ID
          value: <設定したいマシンID>
```

マシンIDには1 ～ 2147483647の値が設定できます。

4.4.1.3. 一般ユーザの作成

コンテナ内にJobCenterユーザとして利用する一般ユーザを作成できます。

以下の手順に従って、マニフェストファイルへパラメータを追加してください。

1. 「2.4.4.1.8.1 パスワードファイルの作成」を参照してパスワードファイルを作成します。
2. 作成したパスワードファイルからConfigMapを作成します。

```
$ kubectl create configmap userlist --from-file <パスワードファイルのファイル名>
```

3. Deploymentのマニフェストファイルに以下を追加します。

■spec.template.spec.containers.env

```
        env:
        - name: JOBCENTER_USERS_PASS_FILE
          value: /password_file
```

パスワードファイルで暗号化したパスワードを使用した場合、nameには
JOBCENTER_USERS_CRYPTED_PASS_FILEを指定してください。

■spec.template.spec.volumes

```
      volumes:
        - name: userlist
          configMap:
            name: userlist
            defaultMode: 420
```

■spec.template.spec.containers.volumeMounts

```
        volumeMounts:
          - name: userlist
            mountPath: /password_file
            subPath: <パスワードファイルのファイル名>
```

4.4.1.4. ポート番号の変更

コンテナMGのプロセスが使用するポート番号を変更できます。

以下の手順に従って、マニフェストファイルへパラメータを追加してください。

1. Deploymentのマニフェストファイルに以下を追加します。

■spec.template.spec.containers.env

```
        env:
        - name: JOBCENTER_JCCOMBASE_PORT
          value: <設定したいjccombaseのポート番号>
        - name: JOBCENTER_JCCOMBASE_OVER_SSL_PORT
          value: <設定したいjccombase over sslのポート番号>
        - name: JOBCENTER_JCEVENT_PORT
          value: <設定したいjceventのポート番号>
        - name: JOBCENTER_JCWEBSERVER_PORT
          value: <設定したいjcwebserverのポート番号>
        - name: JOBCENTER_JCNATS_PORT
          value: <設定したいjcnatsのポート番号>
        - name: JOBCENTER_JCEXECUTOR_PORT
          value: <設定したいjcexecutorのポート番号>
```

4.4.1.5. セットアップ言語の設定

コンテナMGのセットアップ言語を設定できます。

以下の手順に従って、マニフェストファイルへパラメータを追加してください。

1. Deploymentのマニフェストファイルに以下を追加します。

■spec.template.spec.containers.env

```
      env:
      - name: JOBCENTER_LANGUAGE_CODE
        value: <設定したい文字コード>
```

文字コードは以下を設定できます。

- English
- EUC
- Shift-JIS
- Chinese
- UTF-8

4.4.1.6. EKS外部のマシンとの接続

EKS外部のサーバに存在するMGと接続を行うために、ServiceにExternal IPを付与します。

以下の手順に従って、マニフェストファイルへパラメータの追加や設定を行ってください。

1. 以下のkubectlコマンドを実行して、NodeのInternal IPとExternal IPを確認します。

```
$ kubectl get nodes -o wide
```

2. Serviceのマニフェストファイルに以下を追加します。

■spec.externalIPs

```
externalIPs:
- <External IP>
- <Internal IP>
```

本設定を行う場合、CL/WinからコンテナMGへ接続を行うためにNodePortのサービスを作成する必要はありません。

3. Nodeが所属しているセキュリティーグループのインバウンドルール設定で、MGコンテナが使用するが利用するポート(デフォルトは611/10012/23116/23180/23141/23151)を許可する設定にします。

4. コンテナ内のMGおよび、EKS外のサーバ上のMGが、互いに名前解決できる状態にします。

EKS外部のマシンと連携する場合、EKSのNode1つにつき1つのMGコンテナしか作成できません。複数のMGコンテナをEKS上に作成して、EKS外部のマシンと連携する場合、Nodeを複数用意してください。

4.4.1.7. 起動時の設定ファイルの設定

コンテナ内に起動時の設定ファイルを設定することで、コンテナMG環境設定を行えます。

設定可能な起動時の設定ファイルは以下になります。

■daemon.conf

mountPath: /usr/lib/nqs/rc/daemon.conf

設定内容の詳細は<スタンダードモード用環境構築ガイド>の6章 「JobCenter起動時の設定を変更する」 を参照してください。

■jcwebserver.conf

mountPath: /usr/lib/nqs/rc/jcwebserver.conf

設定内容の詳細は<スタンダードモード用環境構築ガイド>の「6.7　jcwebserverの動作設定について」を参照してください。

■jcexecutor_manager.yaml

mountPath: /usr/lib/nqs/rc/jcexecutor_manager.yaml

設定内容の詳細は<スタンダードモード用環境構築ガイド>の「6.8　jcexecutor_managerデーモンの動作設定について」を参照してください。

■jcexecutor_webserver.yaml

mountPath: /usr/lib/nqs/rc/jcexecutor_webserver.yaml

設定内容の詳細は<スタンダードモード用環境構築ガイド>の「6.9 jcexecutor_webserverデーモンの動作設定について」を参照してください。

■nats_start.yaml

mountPath: /usr/lib/nqs/rc/nats_start.yaml

設定内容の詳細は<スタンダードモード用環境構築ガイド>の「6.10 nats-server監視デーモンの動作設定について」を参照してください。

以下の手順に従って、マニフェストファイルへパラメータを追加してください。

1. 以下をベースにして、起動時の設定ファイルの設定を記載したConfigMapのマニフェストファイルを作成します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: <設定したいConfigMapリソース名>
data:
  <設定したConfigMapリソース名>: |
    <起動時の設定ファイルの設定内容>
```

以下はjcwebserver.confの設定を記載したConfigMapのマニフェストファイルの例です。

```
apiVersion: v1
kind: ConfigMap
metadata:
```

```
  name: jcwebserver-conf
data:
  jcwebserver-conf: |
    debug: false
    timeout:
      apiExecution: 300
      readRequest: 20
      readRequestHeader: 5
      writeResponse: 10
      keepAlive: 5
    # tls:
    #   http2: true
    #   certificate: "/usr/spool/nqs/ssl_cert"
    #   privateKey: "/usr/spool/nqs/ssl_key"
    log:
      serverLog:
        maxSize: 4
        maxBackups: 10
      accessLog:
        maxSize: 4
        maxBackups: 10
      errorLog:
        maxSize: 4
        maxBackups: 10
```

2. 作成したマニフェストファイルからConfigMapを作成します。

```
$ kubectl apply -f <作成したマニフェストファイル>
```

3. Deploymentのマニフェストファイルに以下を追加します。

■spec.template.spec.volumes

```
      volumes:
        - name: <設定したConfigMapリソース名>
          configMap:
            name: <設定したConfigMapリソース名>
            defaultMode: 420
```

以下はjcwebserverのConfigMapリソースの設定例です。

```
      volumes:
        - name: jcwebserver-conf
          configMap:
            name: jcwebserver-conf
            defaultMode: 420
```

■spec.template.spec.containers.volumeMounts

```
        volumeMounts:
          - name: <設定したConfigMapリソース名>
            mountPath: <起動時の設定ファイルがMGコンテナ内のマントパス>
            subPath: <設定したConfigMapリソース名>
```

以下はjcwebserverのConfigMapリソースの設定例です。

```
        volumeMounts:
          - name: jcwebserver-conf
```

```
            mountPath: /usr/lib/nqs/rc/jcwebserver.conf
            subPath: jcwebserver-conf
```

上記の例のようにjcwebserverで証明書と秘密鍵を使用する場合には別途、証明書ファイルと秘密鍵ファイルの設定をおこなってください。

4.4.1.8. 証明書・秘密鍵ファイルの設定

証明書および秘密鍵ファイルを設定することで、CL/Win接続時の「保護された接続」の機能やjcwebserverのHTTPS通信の機能が使用出来る様になります。

CL/Win接続時の「保護された接続」の機能については、<スタンダードモード用基本操作ガイド>の「2.3 サーバへ接続する」および、<スタンダードモード用環境構築ガイド>の「6.2 デーモン設定ファイルの使用可能パラメータ」のCOMAGENT_SSLCERT、COMAGENT_SSLKEYパラメータを参照してください。

jcwebserver.confについては、<スタンダードモード用環境構築ガイド>の「6.7　jcwebserverの動作設定について」を参照してください。

以下の手順に従って、マニフェストファイルへパラメータを追加してください。

1. 設定する証明書ファイル及び秘密鍵ファイルを用意します。
2. 証明書ファイルからSecretを作成します。

```
kubectl create secret generic jc-ssl-cert --from-file <証明書ファイル名>
```

3. 秘密鍵ファイルからSecretを作成します。

```
kubectl create secret generic jc-ssl-key --from-file <秘密鍵ファイル名>
```

4. Deploymentのマニフェストファイルに以下を追加します。

■spec.template.spec.volumes

```
      volumes:
        - name: jc-ssl-cert
          secret:
            secretName: jc-ssl-cert
            defaultMode: 420
        - name: jc-ssl-key
          secret:
            secretName: jc-ssl-key
            defaultMode: 420
```

■spec.template.spec.containers.volumeMounts

```
          volumeMounts:
            - name: jc-ssl-cert
              mountPath: /usr/spool/nqs/ssl_cert
              subPath: <証明書ファイル名>
            - name: jc-ssl-key
              mountPath: /usr/spool/nqs/ssl_key
              subPath: <秘密鍵ファイル名>
```

CL/Win接続時の「保護された接続」の機能とjcwebserverのHTTPS通信の機能で使用する証明書・秘密鍵ファイルが異なる場合は、上記手順に沿ってそれぞれの証明書・秘密鍵ファイルの設定をおこなってください。

4.5. リソースの作成

「4.3 マニフェストファイルの作成」および「4.4 マニフェストファイルの編集」を行ったマニフェストファイルを用いて、Kubernetesのリソースを作成します。

以下のkubectlコマンドを実行してください。

```
$ kubectl apply -f <マニフェストファイル>
```

作成後、以下のkubectlコマンドを実行して、Deploymentが作成されていることを確認してください。

```
$ kubectl get deployment
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
jobcenter-manager   1/1     1            1           3m30s
```

4.6. EKS環境の運用

EKS上のJobCenterを運用する際の操作について説明します。

4.6.1. EKS上のMGへの接続（通常の接続）

CL/Win（保護された接続を使用しない）から、EKS上のコンテナMGへ接続する手順について説明します。

1. 以下のkubectlコマンドを実行して、611portの変換先のport番号を確認します。

```
$ kubectl get service
NAME                          TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)
    AGE
jobcenter-manager             ClusterIP   10.100.175.119   <none>        611/TCP,10012/TCP,23116/
TCP,23180/TCP,23141/TCP,23151/TCP  129m
jobcenter-manager-nodeport    NodePort    10.100.28.37     <none>        611:32755/
TCP,23116:31014/TCP,23180:31609/TCP,23151:31527/TCP   129m
```

2. CL/Winの接続画面で、サーバ名に<EKSのノードのIPアドレス>:<上記手順で確認したport>を指定して接続します。

```
指定例）192.0.2.0:32755
```

「4.4.1.6 EKS外部のマシンとの接続」の設定を行っている場合は、以下の手順で接続を行います。

1. 以下のkubectlコマンドを実行して、ServiceリソースのEXTERNAL-IPに、「4.4.1.6　EKS外部のマシンとの接続」で設定したNodeのEXTERNAL-IPが設定されていることを確認します。

```
$ kubectl get service
```

2. CL/Winの接続画面で、サーバ名に<External IP>を指定して接続します。

4.6.2. EKS上のMGへの接続（保護された接続）

CL/Win（保護された接続を使用する）から、EKS上のコンテナMGへ接続する手順について説明します。

1. 以下のkubectlコマンドを実行して、23116portの変換先のport番号を確認します。

```
$ kubectl get service
NAME                          TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)
    AGE
jobcenter-manager             ClusterIP   10.100.175.119   <none>        611/TCP,10012/TCP,23116/
TCP,23180/TCP,23141/TCP,23151/TCP   129m
jobcenter-manager-nodeport    NodePort    10.100.28.37     <none>        23116:31014/
TCP,23180:31609/TCP,23151:31527/TCP   129m
```

2. CL/Winの接続画面で、保護された接続をチェックして、サーバ名に<EKSのノードのIPアドレス>:<上記手順で確認したport>を指定して接続します。

```
指定例）192.0.2.0:31014
```

「4.4.1.6 EKS外部のマシンとの接続」の設定を行っている場合は、以下の手順で接続を行います。

1. 以下のkubectlコマンドを実行して、ServiceリソースのEXTERNAL-IPに、「4.4.1.6　EKS外部のマシンとの接続」で設定したNodeのEXTERNAL-IPが設定されていることを確認します。

```
$ kubectl get service
```

2. CL/Winの接続画面で、保護された接続をチェックして、サーバ名に<External IP>を指定して接続します。

4.6.3. EKS上のMGのWebコンソール機能、WebAPIの利用

EKS上のコンテナMGのWebコンソール機能、WebAPI機能を利用する手順について説明します。

1. 以下のkubectlコマンドを実行して、23180portの変換先のport番号を確認します。

```
$ kubectl get service
NAME                          TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)
    AGE
jobcenter-manager             ClusterIP   10.100.175.119   <none>        611/TCP,10012/TCP,23116/
TCP,23180/TCP,23141/TCP,23151/TCP   129m
jobcenter-manager-nodeport    NodePort    10.100.28.37     <none>        611:32755/
TCP,23116:31014/TCP,23180:31609/TCP,23151:31527/TCP   129m
```

2. WebコンソールまたはWebAPIのURLとして<EKSのノードのIPアドレス>:<上記手順で確認したport>を指定してください。

```
指定例）192.0.2.0:31609
```

「4.4.1.6 EKS外部のマシンとの接続」の設定を行っている場合は、以下の手順で接続を行います。

1. 以下のkubectlコマンドを実行して、ServiceリソースのEXTERNAL-IPに、「4.4.1.6　EKS外部のマシンとの接続」で設定したNodeのEXTERNAL-IPが設定されていることを確認します。

```
$ kubectl get service
```

2. JobCenter　　MGのWebAPIを発行する時のURLの<MGサーバのホスト名またはIPアドレス>に<External IP>:23180 を指定して接続します。

4.6.4. デバッグのためMGコンテナへの接続

MGコンテナへの接続は以下のステップで、実現できます。

■pod名を確認します。

```
$ kubectl get pod
```

■確認したpod名を使って、コンテナに接続をします。

```
$ kubectl exec -it <pod名> -- /bin/sh
```

4.7. AWS EKS環境のトラブルシューティング

4.7.1. ログ収集

以下のコマンドを実行して、コンテナのログを参照することができます。

■pod名を確認します。

```
$ kubectl get pod
```

■確認したpod名を使って、コンテナのログを確認できます。

```
$ kubectl logs <pod名>
```

4.7.2. エラーメッセージ一覧

エラーメッセージーの詳細は「2.6.1.1 MGコンテナにおける障害」を参照してください。

発行年月 Apr 2024