スタンダードモード用コンテナガイド

JobCenter

R17.1

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 R17.1 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 R17.1 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	2025/05/27	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を動作させるコンテナを指します。
AGコンテナ	JobCenter AGを動作させ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。
コンテナAG	コンテナ上で動作しているJobCenter AG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JobCenter利用

本章ではコンテナソフトウェアを用いてJobCenterコンテナを利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Podmanでの説明および使用方法を記載しています。Docker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。dockerコマンドを使用する場合は、本章で説明しているpodmanコマンドの「podman」を「docker」に読み替えてください。

2.1. 概要

スタンダードモードでは、JobCenterの以下の機能のコンテナ化をサポートしています。

■JobCenter MG

■JobCenter AG

上記2種類のコンテナイメージを作成する方法と、作成したコンテナイメージを利用する方法について以降の節で説明します。

本書ではコンテナで実行する機能に応じて「MGコンテナ」「AGコンテナ」と記述して区別します。JobCenter MGとJobCenter AGの機能をひとつのコンテナとして作成・実行することはできませんのでご注意ください。

コンテナ実行環境としてはPodmanを前提として記述していますので、Dockerを利用する場合は利用コマンドを適宜読み替えてください。また、コンテナのベースイメージとしてはRed Hat Universal Base Image (UBI)を前提としています。

2.2. コンテナイメージの作成

JobCenterのコンテナイメージを作成する手順を説明します。

コンテナイメージを作成するためのDockerfile等のファイルはインストールメディアに同梱しておりますが、最新版についてはNECサポートポータルまたはNECカスタマーサポートセンターから入手してください。

2.2.1. MGコンテナイメージの作成

JobCenter MGコンテナを作成する手順について説明します。

事前に以下のファイルを準備して作業用のディレクトリに配置してください。

■Dockerfile, entrypoint.sh

コンテナイメージをビルドするために実行する処理を記述したDockerfileと、コンテナ起動時に実行するスクリプトであるentrypoint.shを配置してください。

それぞれインストールメディアの以下のディレクトリに同梱されています。また、最新盤はNECサポートポータルまたはカスタマサポートセンターから入手可能です。

```
CONTAINER/DOCKER/DOCKERFILE/MGSV/STANDARD
```

Dockerfileについては利用するベースイメージのバージョンがファイル名の末尾に記載されているので、利用したいバージョンのファイルをコピーしてください。

AWS ECS(Fargate)環境で利用するコンテナイメージを作成する場合は entrypoint.sh の代わりに entrypoint_ecs_awsvpc.sh を entrypoint.sh にリネームして利用してください。

■NECWSLM-x.xx-1.x86_64.rpm

LicenseManagerのインストールパッケージをコピーしてください。インストールメディアの以下のディレクトリに格納されています。

```
PACKAGE/LM/LINUX
```

■NECJCpkg-Standard-xx.x-1.elN.x86-64rpm

JobCenter　MGのスタンダードモード用のインストールパッケージをコピーしてください。インストールメディアの以下のディレクトリに格納されています。

```
PACKAGE/JB/LINUX/MGSV
```

el7，el8，el9の3種類のファイルがありますが、ベースイメージのバージョンに合わせて適切なファイルを選択してください。

また、パッチを適用する必要がある場合はパッチパッケージも合わせてコピーしてください。

必要なファイルを配置したらDockerfileの編集を行います。

2.2.1.1. Dockerfileの編集

提供するDockerfileはひな形となりますので、必要に応じて編集が必要となります。

以下が編集のポイントとなりますので、利用環境に応じて編集してください。

■インストールパッケージの確認・修正

事前準備したインストールパッケージのファイル名と一致させてください。

```
ARG LMPKG=<LicenseManagerパッケージファイル名>
ARG JCPKG=<JobCenter MGパッケージファイル名>
```

また、JobCenter MGのパッチを合わせてインストールする場合はいかについてもパッケージファイル名を記述してください。パッチをインストールしない場合は空欄のままで問題ありません。

```
ARG JCPATCH=<パッチパッケージファイル名>
```

■JobCenter MG起動時の設定ファイルの追加(任意)

コンテナ起動時に適用する設定ファイルを必要に応じて追加します。追加する設定ファイルは作業用ディレクトリに適宜作成してください。

表2.1 追加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 MGの起動時の設定を記述する設定ファイルです。 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	JobCenter MGのWeb機能(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。
jcexecutor_manager.yaml	/usr/lib/nqs/rc/jcexecutor_manager.yaml	jcexecutor_managerデーモン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「6.8 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

ファイル名	格納場所	説明
		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。

<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た設定ファイルを作成して作業用ディレクトリへ格納します。

DockerfileのWORKDIR / 行からCMD ["/usr/lib/nqs/cluster/cjcpw", "-local"] 行の間に、設定ファイルごとに以下の行を挿入します。

```
COPY <設定ファイル> <設定ファイルの格納場所>
```

■コードワードファイルの追加(任意)

JobCenter MGのライセンス解除のためのコードワードファイルを追加します。 コードワードファイルについてはコンテナイメージに追加せず、コンテナ作成時にマウントする形でも問題ありません。

コンテナ環境でJobCenter利用する場合においては60日間のお試し期間は存在しません。必ずコードワードファイルによるライセンス解除が必要となります。

AWS ECS(Fargate)環境で利用する場合はコードワードファイルをコンテナイメージに追加してご利用ください。

それ以外の場合については、コンテナ作成時にコードワードファイルをマウントする方法を推奨します。

- コードワードが記載された.lockinfoファイルを用意します。

  .lockinfoファイルについては<セットアップガイド>の「2.5 コードワードを登録する」を参照してください。

- .lockinfoファイルを/etc/opt/wsnlesd/.lockinfoへCOPYします。

  DockerfileのWORKDIR / 行からCMD ["/usr/lib/nqs/cluster/cjcpw", "-local"] 行の間に、設定ファイルごとに以下の行を挿入します。

  ```
  COPY <.lockinfoファイル> /etc/opt/wsnlesd/.lockinfo
  ```

■パスワードファイルの追加(任意)

JobCenterを利用する一般ユーザを追加する場合、パスワードファイルを利用してコンテナ起動時にOSユーザの追加を行います。 パスワードファイルはユーザ名とパスワードが記載されたファイルです。

パスワードファイルはコンテナイメージに追加せずに、コンテナ作成時にマウントする方法を推奨します。

JobCenter管理者ユーザ(nsumsmgr)ユーザをパスワードファイルに記載しないでください。nsumsmgrユーザのパスワードはコンテナ作成時に環境変数で指定します。

- 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形式については「2.3.1.9.1 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任意のパスへCOPYします。

  DockerfileのWORKDIR / 行からCMD ["/usr/lib/nqs/cluster/cjcpw", "-local"] 行の間に、設定ファイルごとに以下の行を挿入します。

```
COPY <パスワードファイル> <任意のパス>
```

パスワードファイルのパスはコンテナ作成時に環境変数JOBCENTER_USERS_PASS_FILEまたはJOBCETNER_USERS_CRYPTED_PASS_FILEで指定します。

環境変数でパスワードファイルのパスを指定しない場合はコンテナ起動時にOSユーザは作成されません。

2.2.1.2. イメージのビルド

以下の手順に従ってイメージをビルドします。

1. 作業用のディレクトリへコピー・修正した以下のファイルを配置します。

   ■Dockerfile

   ■entrypoint.sh

   ■JobCenter MGパッケージファイル

   ■LicenseManagerパッケージファイル

   ■（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

   ■その他任意でコンテナイメージへ追加するファイル

2. 作業ディレクトリ内で以下のコマンドでイメージのビルドを行います。

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

3. ビルド後に以下のコマンドでイメージの一覧を表示します。

```
podman image ls
```

   ビルド時に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。

2.2.2. AGコンテナイメージの作成

JobCenter AGコンテナを作成する手順について説明します。

事前に以下のファイルを準備して作業用のディレクトリに配置してください。

■Dockerfile, entrypoint.sh

コンテナイメージをビルドするために実行する処理を記述したDockerfileと、コンテナ起動時に実行するスクリプトであるentrypoint.shを配置してください。

それぞれインストールメディアの以下のディレクトリに同梱されています。また、最新盤はNECサポートポータルまたはカスタマサポートセンターから入手可能です。

```
CONTAINER/DOCKER/DOCKERFILE/AG/STANDARD
```

Dockerfileについては利用するベースイメージのバージョンがファイル名の末尾に記載されているので、利用したいバージョンのファイルをコピーしてください。

■NECWSLM-x.xx-1.x86_64.rpm

LicenseManagerのインストールパッケージをコピーしてください。インストールメディアの以下のディレクトリに格納されています。

```
PACKAGE/LM/LINUX
```

■NECJCAGpkg-xx.x-1.x86-64.rpm

JobCenter AGのインストールパッケージをコピーしてください。インストールメディアの以下のディレクトリに格納されています。

```
PACKAGE/JB/LINUX/AG
```

2.2.2.1. Dockerfileの編集

提供するDockerfileはひな形となりますので、必要に応じて編集が必要となります。

以下が編集のポイントとなりますので、利用環境に応じて編集してください。

■インストールパッケージの確認・修正

事前準備したインストールパッケージのファイル名と一致させてください。

```
ARG LMPKG=<LicenseManagerパッケージファイル名>
ARG JCPKG=<JobCenter AGパッケージファイル名>
```

■コードワードファイルの追加(任意)

JobCenter AGのライセンス解除のためのコードワードファイルを追加します。 コードワードファイルについてはコンテナイメージに追加せず、コンテナ作成時にマウントする形でも問題ありません。

コンテナ環境でJobCenter利用する場合においては60日間のお試し期間は存在しません。必ずコードワードファイルによるライセンス解除が必要となります。

AWS ECS(Fargate)環境で利用する場合はコードワードファイルをコンテナイメージに追加してご利用ください。

それ以外の場合については、コンテナ作成時にコードワードファイルをマウントする方法を推奨します。

- コードワードが記載された.lockinfoファイルを用意します。

  .lockinfoファイルについては<セットアップガイド>の「2.5　コードワードを登録する」を参照してください。

- .lockinfoファイルを/etc/opt/wsnlesd/.lockinfoへCOPYします。

  DockerfileのENTRYPOINT ["/entrypoint.sh"] 行の前に以下の行を挿入します。

```
COPY <.lockinfoファイル> /etc/opt/wsnlesd/.lockinfo
```

■秘密鍵ファイルの追加(任意)

JobCenter　AGの認証のための秘密鍵ファイルを追加します。　秘密鍵ファイルについてはコンテナイメージに追加せず、コンテナ作成時にマウントする形でも問題ありません。

- あらかじめAG認証に利用する鍵ペアを作成しします。

  鍵ペアについては<スタンダードモード用環境構築ガイド>の「4.1.1　エージェントの登録」の記述も参照してください。

- 秘密鍵ファイルをコンテナ内の任意のパスへCOPYします。

  DockerfileのENTRYPOINT ["/entrypoint.sh"] 行の前に以下の行を挿入します。

```
COPY <秘密鍵ファイル> <秘密鍵ファイルのパス>
```

■ユーザの追加(任意)

ジョブ実行ユーザとしてrootユーザ以外のユーザを利用する場合は以下のようにRUN命令でuseraddコマンドを実行して適宜ユーザを追加してください。

```
RUN useradd -m -s /bin/bash -u <UID> <ユーザ>
```

追加したユーザでジョブを実行する場合は、コンテナ作成時に環境変数JOBCENTER_AGENT_USER_LIST にユーザを指定する必要があります。

2.2.2.2. イメージのビルド

以下の手順に従ってイメージをビルドします。

1. 作業用のディレクトリへコピー・修正した以下のファイルを配置します。

   ■Dockerfile

   ■entrypoint.sh

   ■LicenseManagerのインストールパッケージ(NECWSLM-x.xx-1.i386.rpm)

   ■JobCenter AGのインストールパッケージ(NECJCAGpkg-xx-x-1.x86-64.rpm)

2. 作業ディレクトリ内で以下のコマンドでイメージのビルドを行います。

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

3. ビルド後に以下のコマンドでイメージの一覧を表示します。

```
docker image ls
```

ビルド自に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。

2.2.3. 作成したイメージで利用可能な環境変数について

作成したコンテナイメージを利用してコンテナを作成する際に環境変数を指定することで設定を行うことができます。

MGコンテナイメージとAGコンテナイメージそれぞれで利用可能な環境変数について以下に記載します。

2.2.3.1. MGコンテナイメージで利用可能な環境変数

MGコンテナイメージでは以下の環境変数を利用可能です。

表2.2 MGコンテナイメージで利用可能な環境変数

環境変数名	環境変数値	デフォルト値	範囲
JOBCENTER_HOSTNAME	MGホスト名  本環境変数はイメージ作成時にentrypoins.shの代わりにentrypoint_ecs_awsvpc.shを利用して、Amazon ECS(Fargate)用に作成したイメージでのみ利用可能です。		ホスト名として有効な文字列
JOBCENTER_NSUMSMGR_PASS	JobCenter管理者ユーザの平文のパスワード		空欄または平文のパスワード
JOBCENTER_NSUMSMGR_CRYPTED_PASS	JobCenter管理者ユーザの暗号化されたパスワード		空欄または暗号化されたパスワード
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_USERS_PASS_FILE	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		空欄またはコンテナ内から参照可能なファイルパス
JOBCENTER_USERS_CRYPTED_PASS_FILE	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		空欄またはコンテナ内から参照可能なファイルパス
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

環境変数名	環境変数値	デフォルト値	範囲
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

2.2.3.2. AGコンテナイメージで利用可能な環境変数

AGコンテナイメージでは以下の環境変数を利用可能です。

表2.3 AGコンテナイメージで利用可能な環境変数

環境変数名	環境変数値	デフォルト値	範囲
JOBCENTER_AGENT_NAME	JobCenter MGに登録したエージェント名		エージェント名として有効な文字列
JOBCENTER_INSTANCE_NAME	コンテナ内でエージェントを識別するエージェント名		エージェント名として有効な文字列
JOBCENTER_ENDPOINT_URL	接続先JobCenter MGのURL		URL文字列
JOBCENTER_LANG	エージェントの言語設定	utf-8	■utf-8 ■euc ■sjis ■english ■gb18030
JOBCENTER_PRIVKEY_PATH	エージェント認証用の秘密鍵ファイルのパス		コンテナ内から参照可能なファイルパス
JOBCENTER_AGENT_CREATE_OPTS	エージェント作成時の追加オプション		空欄または`jcagctrl create`コマンドに指定するオプション
JOBCENTER_AGENT_USER_LIST	エージェント登録するユーザのリスト		空欄またはコンテナ内に存在するユーザをカンマ区切りで記載

2.2.4. イメージ作成に関する注意事項

■entrypoint.shに記述されているセットアップ処理(MGコンテナのnssetupや、AGコンテナのjcagctrl create コマンド)を、Dockerfileに直接記述しないでください。

■提供しているDockerfile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としています。podman 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する場合、COPY行のコピー元のパスを適切に修正してください。

■ベースイメージとしているUBI(Red Hat Universal Base Images)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は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「2.5.2 LicenseManagerインストール後に出力されるメッセージ」

2.3. コンテナの作成/起動

JobCenterのコンテナを作成/起動する手順を説明します。

起動するコンテナイメージの作成手順については「2.2 コンテナイメージの作成」を参照してください。

2.3.1. MGコンテナの作成/起動

本節ではMGコンテナを作成/起動する際の設定やコマンド例を説明します。

最低限のオプションを指定してMGコンテナを起動するコマンド例は以下の通りです。

```
podman container run --detach --hostname <ホスト名> --volume <.lockinfoのファイルパス>:/etc/opt/
wsnlesd/.lockinfo --init --env JOBCENTER_NSUMSMGR_PASS=<パスワード> <MGコンテナイメージ名>
```

MGコンテナを作成/起動する際の設定について項目ごとに以下で説明します。

説明中に記載されているpodman container run(あるいはdocker container run)コマンドの各種オプションについては、PodmanやDockerのドキュメントを確認して適切に指定してください。

設定項目	必須
「2.3.1.1 ホスト名の設定」	◎
「2.3.1.2 ライセンスの登録」	◎
「2.3.1.3 initプロセスの設定」	◎
「2.3.1.4 JobCenter管理者ユーザのパスワード設定」	◎
「2.3.1.5 JobCenter MGのデータ永続化」	○
「2.3.1.6 外部ポートの公開」	○
「2.3.1.7 JobCenter MGの言語設定、マシンID設定」	○
「2.3.1.8 設定ファイルのマウント」	○
「2.3.1.9 JobCenter MGのユーザ追加」	○

（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

2.3.1.1. ホスト名の設定

MGコンテナのホスト名を指定します。指定したホスト名はJobCenter MGのマシン名として利用されます。

podman container runコマンドの以下のオプションで指定します。

```
--hostname <ホスト名>
```

指定するホスト名について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

2.3.1.2. ライセンスの登録

MGコンテナを起動するにはコードワードが登録された.lockinfoファイルが必要となります。

コードワードが記載された.lockinfoファイルを用意し、以下のようにコンテナの/etc/opt/wsnlesd/.lockinfoにマウントすることでコンテナMGのライセンス登録を行います。

```
--volume <マウント元>:/etc/opt/wsnlesd/.lockinfo
```

コンテナイメージを作成する際に.lockinfoファイルを登録している場合は本設定は不要です。

2.3.1.3. initプロセスの設定

MGコンテナの終了処理を正常に行うため、--initオプションを指定してください。

指定しない場合、コンテナ内で実行したプロセスがOSによって回収されない場合があります。

2.3.1.4. JobCenter管理者ユーザのパスワード設定

環境変数でJobCenter管理者ユーザ(nsumsmgr)のパスワードを設定します。

パスワードを平文で記述する場合は以下のように指定します。

```
--env JOBCENTER_NSUMSMGR_PASS=<パスワード>
```

暗号化したパスワードを用いる場合は以下のように指定します。

```
--env JOBCENTER_NSUMSMGR_CRYPTED_PASS=<暗号化されたパスワード>
```

パスワードの暗号化については「2.3.1.10 パスワードの暗号化」 を参照してください。

2.3.1.5. JobCenter MGのデータ永続化

コンテナを再作成した場合に永続化していないデータは失われます。

MGコンテナではジョブネットワーク、スケジュール、カレンダ等の定義ファイルや実行中のジョブの定義データや実行結果(ジョブの標準出力、標準エラー出力)をスプールディレクトリ(/usr/spool/nqs)配下に格納しているため、コンテナ再作成後に元の動作を継続するにはこのディレクトリを永続化する必要があります。

以下のように永続化領域をスプールディレクトリにマウントしてください。

```
--volume <マウント元>:/usr/spoon/nqs
```

2.3.1.6. 外部ポートの公開

MGコンテナへ外部から接続するためにポートを公開する必要があります。

JobCenter　MGが利用するポートには以下があります。外部公開が必要なポートについて--publishオプションで公開してください。

ポート番号	説明	外部公開の必要性
611	jccombaseが使用するポート番号	CL/Winで接続する際に必要
23116	jccombase over sslが使用するポート番号	CL/WinでSSL接続する際に必要
10012	jceventが使用するポート番号	MGコンテナへイベント送信する場合に必要

ポート番号	説明	外部公開の必要性
23180	jcwebserverが使用するポート番号	Webコンソールへのアクセス、またはWeb　API利用時に必要
23141	jcnatsのポート番号	内部連携用のポートです。外部公開する必要はありません。
23151	jcexecutorのポート番号	JobCenter AGが接続するポートです。外部からAGを接続する場合に必要です。
50080	jcresが使用するポート番号	SystemManager Gとの連携等で利用する場合に必要

通常実施する必要はありませんが、以下の環境変数を指定することでMGコンテナが利用するポート番号を変更することができます。

ポート番号を変更した場合、外部公開するポートの指定もそれに合わせて変更してください。

表2.4 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

2.3.1.7. JobCenter MGの言語設定、マシンID設定

JobCenter MGの言語設定およびマシンIDについてデフォルト設定から変更する場合は環境変数での指定が必要です。

それぞれ以下の環境変数で指定可能です。

表2.5 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

環境変数名	環境変数値	デフォルト値	範囲
			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

2.3.1.8. 設定ファイルのマウント

JobCenter MGの動作をカスタマイズするために設定ファイルを変更する必要がある場合は必要に応じてマウントします。

コンテナ作成/起動時にマウント可能な設定ファイルは以下の通りです。

表2.6 JobCenter MG設定ファイル

カテゴリ	ファイル名	種類	説明
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。
	/etc/profile	ファイル	リモート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「15.1 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。
jcwebserverの設定	/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。
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

カテゴリ	ファイル名	種類	説明
			>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。
SAPの接続先の設定	/usr/lib/nqs/sap/destconf.f	ファイル	SAP ERP連携機能において、接続先のパラメータを設定するファイルです。 本ファイルの設定に関する詳細は、<クラシックモード用SAP機能利用の手引き>の「1.1.2 接続パラメータファイルを設定する」を参照してください。

2.3.1.9. JobCenter MGのユーザ追加

JobCenterで利用する一般ユーザを追加するための設定を行います。

設定には次の手順が必要です。

1. パスワードファイルの作成
2. パスワードファイルのマウント
3. 環境変数でマウント先パスの指定

2.3.1.9.1. パスワードファイルの作成

追加したユーザごとにユーザ名:パスワード形式で記載したファイルを作成します。

記載例）

```
user1:password1
user2:passowrd2
```

パスワード部分は平文で記述するか、暗号化したしたものを記述することができます。パスワードを暗号化する方法については「2.3.1.10 パスワードの暗号化」を参照してください。

2.3.1.9.2. パスワードファイルのマウント

作成したパスワードファイルをコンテナ上の任意のパスにマウントします。

```
--volume <パスワードファイル><任意のパス>
```

2.3.1.9.3. 環境変数でマウント先パスの指定

マウントしたパスワードファイルのコンテナ上ののでのパスを環境変数に指定します。

パスワードファイルに記述したパスワードが平文の場合は環境変数JOBCENTER_USERS_PASS_FILEに指定します。

```
--env JOBCENTER_USERS_PASS_FILE=<パスワードファイルのパス>
```

パスワードファイルに記述したパスワードが暗号化したものの場合は環境変数JOBCETNER_USERS_CRYPTED_PASS_FILE に指定します。

```
--env JOBCENTER_USERS_CRYPTED_PASS_FILE=<パスワードファイルのパス>
```

2.3.1.10. パスワードの暗号化

「2.3.1.4　JobCenter管理者ユーザのパスワード設定」および「2.3.1.9　JobCenter　MGのユーザ追加」 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。

2.3.2. AGコンテナの作成/起動

本節ではAGコンテナのイメージを利用してコンテナを作成/起動する際の設定やコマンド例を説明します。

最低限のオプションを指定してAGコンテナを起動するコマンド例は以下の通りです。

```
podman container run --detach --volume <.lockinfoのファイルパス>:/etc/opt/wsnlesd/.lockinfo --volume <秘密鍵のファイルパス>:/tmp/privkey.pem --env JOBCENTER_AGENT_NAME=<MGに登録したエージェント名> --env JOBCENTER_INSTANCE_NAME=<エージェント名> --env JOBCENTER_PRIVKEY_PATH=/tmp/privkey.pem --env JOBCENTER_ENDPOINT_URL=<http://MGホスト名:23151> <AGコンテナイメージ名>
```

AGコンテナを作成/起動する際の設定について項目ごとに以下で説明します。

説明中に記載されているpodman container run(あるいはdocker container run)コマンドの各種オプションについては、PodmanやDockerのドキュメントを確認して適切に指定してください。

設定項目	必須
「2.3.2.1 エージェント名とインスタンス名」	◎
「2.3.2.2 接続先JobCenter MGのURL」	◎
「2.3.2.3 秘密鍵」	◎
「2.3.2.4 データ領域の永続化」	○
「2.3.2.5 言語設定」	○
「2.3.2.6 実行ユーザの一覧」	○
「2.3.2.7 その他オプション」	○

（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

2.3.2.1. エージェント名とインスタンス名

JobCenter MGに登録したエージェントと名前を環境変数JOBCENTER_AGENT_NAMEで指定します。

コンテナ内でのエージェントの管理用にインスタンス名を環境変数JOBCENTER_INSTANCE_NAME で指定する必要がありますが、これはエージェント名を同じ名前で問題ありません。

```
--env JOBCENTER_AGENT_NAME=<エージェント名> --env JOBCENTER_INSTANCE_NAME=<エージェント名>
```

2.3.2.2. 接続先JobCenter MGのURL

接続先JobCenter MGのURLを環境変数JOBCENTER_ENDPOINT_URLで指定します。

http, httpsのどちらも利用できるのでJobCenter MGの設定に合わせて適切なプロトコルを指定してください。

```
--env JOBCENTER_ENDPOINT_URL=<http://MGホスト名:23151>
```

2.3.2.3. 秘密鍵

接続先JobCenter MGにエージェントを登録する際に生成または設定した鍵ペアのうち、秘密鍵ファイルをAGコンテナに配置してそのファイルパスを環境変数 JOBCENTER_PRIVKEY_PATHで指定します。

```
--volume <秘密鍵ファイルのパス>:<コンテナ内のファイルパス> --env JOBCENTER_PRIVKEY_PATH=<コンテナ内のファイルパス>
```

2.3.2.4. データ領域の永続化

JobCenter AGが動作中の一時データやログファイルを格納する領域を永続化するため、コンテナ内の/jcagディレクトリに永続化領域をマウントします。

データ領域を永続化しなかった場合、AGコンテナを再作成した場合に実行中のジョブの状態やログファイルが失われます。

```
--volume <データ領域のディレクトリパス>:</jcag>
```

2.3.2.5. 言語設定

JobCenter AGの言語設定をデフォルトのutf-8から変更する場合は環境変JOBCENTER_LANGで指定します。

```
--env JOBCENTER_LANG=<利用する言語>
```

利用する言語として指定可能な値は utf-8, euc, sjis, english, gb18030 のいずれかです。

2.3.2.6. 実行ユーザの一覧

JobCenter AGでジョブを実行するユーザとして登録するユーザの一覧を環境変数JOBCENTER_AGENT_USER_LISTで指定します。

ユーザが複数いる場合はカンマ区切りで記述します。

指定しない場合はrootユーザのみが指定されているものとして扱います。

```
--env JOBCENTER_AGENT_USER_LIST=root,user1,user2
```

2.3.2.7. その他オプション

JobCenter AGのインスタンスを登録する際のオプションを追加で指定したい場合に環境変数JOBCENTER_AGENT_CREATE_OPTS で指定します。

例えばJobCenter MGへの接続にプロキシサーバを利用する場合は以下のように指定します。

```
--env JOBCENTER_AGENT_CREATE_OPTS="--proxy http://プロキシサーバホスト名:ポート番号"
```

利用可能なオプションは<スタンダードモード用コマンドリファレンス>の「4.1.3.1　create」を参照してください。

2.4. 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。

2.4.1. 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「2.4.1.1 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、「2.4.2　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。

2.4.1.1. MGコンテナにおける障害

表2.7「MGコンテナにおける障害」を参照してください。

表2.7 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	「2.4.1.1.1 イメージの作成に失敗する」
コンテナ環境の構築時	「2.4.1.1.2 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	「2.4.1.1.3 MGへのログインに失敗する」
コンテナ環境の運用時	「2.4.1.1.4 マシン一覧へMGが追加できない」
コンテナ環境の運用時	「2.4.1.1.5 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	「2.4.1.1.6 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	「2.4.1.1.7 フローが進行しない」

2.4.1.1.1. 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。

2.4.1.1.2. 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。 「2.3.1.1 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。 「2.3.1.2 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。 「2.3.1.4 JobCenter管理者ユーザのパスワード設定」 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。 「2.3.1.7 JobCenter MGの言語設定、マシンID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。 「2.3.1.7 JobCenter MGの言語設定、マシンID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。 「2.3.1.7 JobCenter MGの言語設定、マシンID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 表2.4「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、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 表2.4「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、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。 「2.3.1.9.1 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。 「2.3.1.5 JobCenter MGのデータ永続化」 で永続化した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。

2.4.1.1.3. 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20.3「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、「2.3.1.6 外部ポートの公開」 を参照して、ポートの開放を行ってください。

   

   表2.4「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」 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、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。

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。

2.4.1.1.4. 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、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。

■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、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。

2.4.1.1.5. 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、「2.3.1.6 外部ポートの公開」 を参照して、ポートの開放を行ってください。

表2.4「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」 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。

2.4.1.1.6. 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。

「2.3.2.2 接続先JobCenter MGのURL」 で指定したURLのホスト名について名前解決の正引きが行われるよう設定してください。

2.4.1.1.7. 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のマネージャーレ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。

2.4.2. 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. 「2.4.2.1 Dockerホストの情報採取」
2. 「2.4.2.2 停止しているコンテナの起動」
3. 「2.4.2.3 コンテナ内の情報採取」

2.4.2.1. 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、「2.4.2.4 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。

2.4.2.2. 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「2.4.2.3 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>
```

2.4.2.3. 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。

2.4.2.4. 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。

■「2.4.2.4.1 コンテナ情報の採取」

■「2.4.2.4.2 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）

■「2.4.2.4.3 Dockerネットワーク情報の採取」

■「2.4.2.4.4 マウント情報の採取」

■「2.4.2.4.5 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」

2.4.2.4.1. 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
```

2.4.2.4.2. 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2.8「コピーするファイル一覧」を参照してください。

表2.8 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

2.4.2.4.3. 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。

2.4.2.4.4. 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
```

2.4.2.4.5. 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。

2.4.2.5. 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。

2.4.2.5.1. 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。

2.4.2.5.2. 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、「2.4.2.4　Dockerホストの情報を手動で採取する」を実施してください。

2.4.2.5.3. 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. Amazon ECS (Fargarte)環境での利用

本章ではAmazon ECS (Fargate)環境におけるJobCenterコンテナの利用について説明します。

3.1. 概要

Amazon　ECS(Fargate)環境でMGコンテナおよびAGコンテナを実行するための環境構築について例を用いて説明します。

ここでは図3.1「Amazon　　ECS(Fargate)上のJobCenter環境の構成例」で示した構成でJobCenter　　MGとJobCenter AGをデプロイする例を示します。

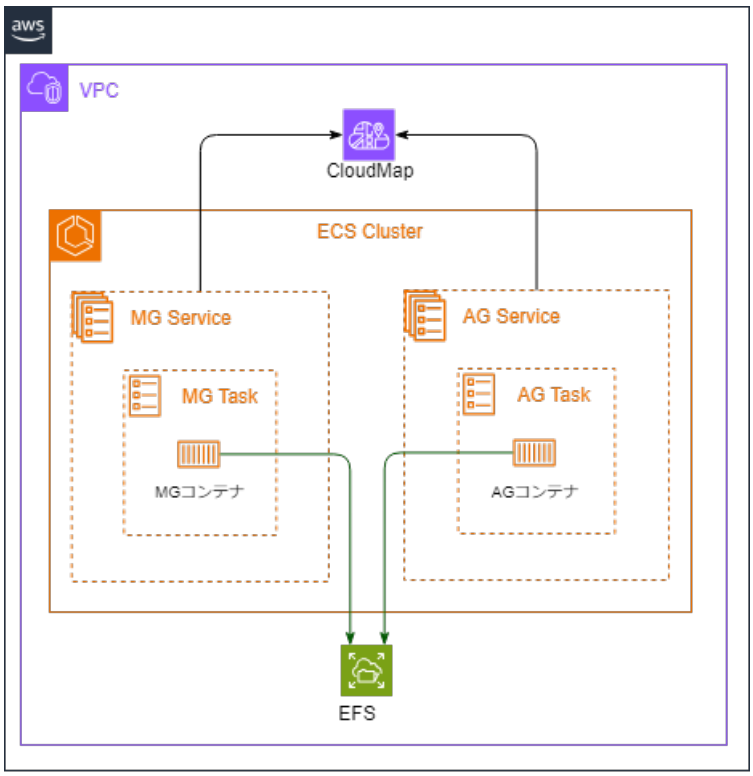

図3.1 Amazon ECS(Fargate)上のJobCenter環境の構成例

VPCおよび関連リソース、EFSファイルシステム、ECSクラスターの作成方法について本書では解説しないため、AWS公式ドキュメントを参照して適宜作成してください。

3.2. 事前準備

3.2.1. コンテナイメージの作成

コンテナイメージの作成手順については「2.2 コンテナイメージの作成」を参照してください。

MGコンテナイメージについてはAmazon ECS(Fargate)上で動作させる場合は以下が必要となりますので、手順に記載されている注意書きと合わせて確認して対応してください。

■entrypoint_ecs_awsvpc.shの利用

■コンテナイメージへのコードワードファイルの追加

AGコンテナイメージについてはAmazon ECS(Fargate)上で動作させる場合は以下が必要となりますので、手順に記載されている注意書きと合わせて確認して対応してください。

■コンテナイメージへのコードワードファイルの追加

作成したコンテナイメージはECRから参照可能な任意のコンテナレジストリへ登録してください。本章では作成したコンテナイメージはAmazon ECRへ登録することを想定しています。

3.2.2. サービス検出に必要な設定

Amazon ECSのサービス検出を利用してECSのサービス間で名前解決が行えるようにネームスペーの作成とサービス登録を行います。

ECSサービスを作成する際に「サービス検出を有効化」を選択することで、タスクのIPアドレスが登録され、他サービスがDNSで「サービス名.ネームスペース名」で名前解決できるようになります。

本章ではMGコンテナの名前解決が行えるように登録します。

■ネームスペースの作成

以下のコマンドでネームスペースを作成します。

```
$ aws servicediscovery create-private-dns-namespace \
--name ネームスペース名 \
--vpc VPCID
```

コマンドを実行すると以下のようにオペレーションIDが出力がされます。

```
{
  "OperationId": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx-xxxxxxxx"
}
```

■ネームスペースのID確認

以下のコマンドを実行してネームスペースのIDを確認します。

```
$ aws servicediscovery get-operation \
--operation-id オペレーションID
```

コマンドを実行すると以下のようにネームスペースのIDを含む出力がされます。

```
{
    "Operation": {
        "Id": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx-xxxxxxxx",
        "Type": "CREATE_NAMESPACE",
        "Status": "SUCCESS",
        "CreateDate": 1519777852.502,
        "UpdateDate": 1519777856.086,
        "Targets": {
           "NAMESPACE": "ns-xxxxxxxxxxxxxxxx"
        }
    }
}
```

ns-xxxxxxxxxxxxxxxx部分がネームスペースのIDとなります。

■サービス登録

以下のコマンドでサービス登録を行います。サービス名にはMGコンテナへアクセスする際に利用したいホスト名を指定してください。

```
$ aws servicediscovery create-service \
--name サービス名 \
--dns-config "NamespaceId="ネームスペースのID",DnsRecords=[{Type="A",TTL="300"}]"
```

コマンドを実行すると以下のような出力がされます。

```
{
```

```
  "Service": {
      "Id": "srv-xxxxxxxxxxxxxxxx",
      "Arn": "arn:aws:servicediscovery:region:aws_account_id:service/srv-xxxxxxxxxxxxxxxx",
      "Name": "サービス名",
      "DnsConfig": {
          "NamespaceId": "ns-xxxxxxxxxxxxxxxx",
          "DnsRecords": [
              {
                  "Type": "A",
                  "TTL": 300
              }
          ]
      },
      "HealthCheckCustomConfig": {
          "FailureThreshold": 1
      },
      "CreatorRequestId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
  }
}
```

3.3. コンテナの作成/起動

3.3.1. MGコンテナの作成/起動

Amazon ECS(Fargate)上でMGコンテナを作成/起動する際の設定を説明します。

3.3.1.1. タスク定義の作成

AWS　ECSでコンテナを起動するためにはタスク定義を作成する必要があります。タスク定義はアプリケーションを構成するパラメータや実行するコンテナについてJSON形式で記述します。

3.3.1.1.1. 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について説明します。

各パラメータの詳細についてはAWSの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。
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。

パラメータ分類	パラメータ名	型	必須	説明
				「"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" }, ... ]`
	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2.2「MGコンテナイメージで利用可能な環境変数」 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。 サービス検出を利用する場合は「3.2.2 サービス検出に必要な設定」 で登録した値に合わせて「サービス名.ネームスペース名」で指定してください。 環境変数は以下のように配列形式で指定します:

パラメータ分類	パラメータ名	型	必須	説明
				"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",
				"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":

パラメータ分類	パラメータ名	型	必須	説明

				<pre>{
				"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"
				}
				}
				]</pre>

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

3.3.1.1.2. 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": "mg.test"
            },
            {
                "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"
  }
}
```

3.3.1.2. タスク定義の登録とサービスの作成

作成したJSON形式のタスク定義を以下のコマンドで登録します。

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

登録したタスク定義にもとづいてECSクラスター内でタスクを実行するためにサービスを以下のコマンドで作成します。

```
$ aws ecs create-service \
--cluster 事前に作成したECSのクラスター名 \
--task-definition 登録したタスク定義名 \
--enable-execute-command \
--service-name 任意のサービス名 \
--desired-count 1 \
--launch-type FARGATE \
--platform-version 1.4.0 \
--service-registries "ネームスペースに登録したサービスのARN"
--network-configuration "タスクに関連付けられるVPCサブネットとセキュリティグループ"
```

サービス検出を利用しない場合は`--service-registries`オプションは不要です。

以下はサービス作成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 \
--platform-version 1.4.0 \
--service-registries "[{\"registryArn\":\"arn:aws:servicediscovery:region:aws_account_id:service/
srv-xxxxxxxxxxxxxxxx\"}]"
--network-configuration "awsvpcConfiguration={subnets=[subnet-XXX],securityGroups=[sg-
XXX],assignPublicIp=DISABLED}"
```

3.3.2. AGコンテナの作成/起動

Amazon ECS(Fargate)上でAGコンテナを作成/起動する際の設定を説明します。

3.3.2.1. タスク定義の作成

AWS　ECSでコンテナを起動するためにはタスク定義を作成する必要があります。タスク定義はアプリケーションを構成するパラメータや実行するコンテナについてJSON形式で記述します。

3.3.2.1.1. 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について説明します。

各パラメータの詳細についてはAWSのドキュメントを参照してください。

表3.2 AGコンテナで利用するタスク定義パラメーター覧

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。
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。

パラメータ分類	パラメータ名	型	必須	説明
				「"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AGコンテナ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。
	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2.3「AGコンテナイメージで利用可能な環境変数」 を参照してください。  以下の環境変数は必ず指定してください。 JOBCENTER_AGENT_NAME JOBCENTER_INSTANCE_NAME JOBCENTER_PRIVKEY_PATH JOBCENTER_ENDPOINT_URL 環境変数は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" }`

パラメータ分類	パラメータ名	型	必須	説明
				] }

	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",
				"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"
				}
				}
				]

パラメータ分類	パラメータ名	型	必須	説明
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

3.3.2.1.2. 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。

また、コードワードファイルや秘密鍵ファイルはコンテナイメージに追加されている想定とします。

```
{
    "family": "ag",
    "containerDefinitions": [
        {
            "name": "ag-container",
            "image": "XXX.dkr.ecr.ap-northeast-1.amazonaws.com/jobcenter-ag:latest",
            "cpu": 2048,
            "memory": 4096,
            "portMappings": [],
            "essential": true,
            "environment": [
                {
                    "name": "JOBCENTER_AGENT_NAME",
                    "value": "testag"
                },
                {
                    "name": "JOBCENTER_INSTANCE_NAME",
                    "value": "testag"
                },
                {
                    "name": "JOBCENTER_PRIVKEY_PATH",
                    "value": "秘密鍵ファイルのパス"
                },
                {
                    "name": "JOBCENTER_ENDPOINT_URL",
                    "value": "http://MGサービス名.ネームスペース名:23151"
                },
                {
                    "name": "JOBCENTER_AGENT_USER_LIST",
                    "value": "root,test"
                }
            ],
            "mountPoints": [
                {
                    "sourceVolume": "efs-volume",
                    "containerPath": "/jcag",
                    "readOnly": false
                }
            ],
            "volumesFrom": [],
            "linuxParameters": {
                "initProcessEnabled": true
            },
```

```
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/FargateTestGroup",
                    "awslogs-create-group": "true",
                    "awslogs-region": "ap-northeast-1",
                    "awslogs-stream-prefix": "ecs"
                }
            },
            "systemControls": []
        }
    ],
    "taskRoleArn": "タスクロール",
    "executionRoleArn": "タスク実行ロール",
    "networkMode": "awsvpc",
    "volumes": [
        {
            "name": "efs-volume",
            "efsVolumeConfiguration": {
                "fileSystemId": "EFSファイルシステムID",
                "rootDirectory": "/"
            }
        }
    ],
    "placementConstraints": [],
    "requiresCompatibilities": [
        "FARGATE"
    ],
    "cpu": "2048",
    "memory": "4096",
    "runtimePlatform": {
        "cpuArchitecture": "X86_64",
        "operatingSystemFamily": "LINUX"
    }
}
```

3.3.2.2. タスク定義の登録とサービスの作成

作成したJSON形式のタスク定義を以下のコマンドで登録します。

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

登録したタスク定義にもとづいてECSクラスター内でタスクを実行するためにサービスを以下のコマンドで作成します。

```
$ aws ecs create-service \
--cluster 事前に作成したECSのクラスター名 \
--task-definition 登録したAGタスク定義名 \
--enable-execute-command \
--service-name 任意のAGサービス名 \
--desired-count 1 \
--launch-type FARGATE \
--platform-version 1.4.0 \
--network-configuration "タスクに関連付けられるVPCサブネットとセキュリティグループ"
```

以下はサービス作成コマンドの実行例です。

```
$ aws ecs create-service \
```

```
--cluster MGCluster \
--task-definition ag-task-definition \
--enable-execute-command \
--service-name AGService \
--desired-count 1 \
--launch-type FARGATE \
--platform-version 1.4.0 \
--network-configuration "awsvpcConfiguration={subnets=[subnet-XXX],securityGroups=[sg-
XXX],assignPublicIp=DISABLED}"
```

3.3.2.3. エージェントの設定ファイル修正

本作業は必要に応じて任意で実施してください。

コンテナAGの動作を一部動作を変更するには、永続化されたデータ領域に格納される設定ファイル(jcexecutor_agent.yaml)を修正する必要があります。

特に、Amazon　ECS(Fargate)上で実行しているコンテナAGから拡張カスタムジョブのAWS連携部品についてIAMRoleを利用して実行する場合は request.environment.agentEnvironment/パラメータをtrue に設定する必要があるため注意してください。

設定ファイルの詳細は<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>の「3.1　エージェントの動作設定変更」を参照してください。

設定ファイルはコンテナ内の以下のパスに作成されるので初回起動後に編集してください。

```
/jcag/spool_${JOBCENTER_INSTANCE_NAME}/jcexecutor/agent/jcexecutor_agent.yaml
```

起動中のAGコンテナへ接続して直接編集する場合は以下のように対話シェルを実行して作業を行ってください。

```
$ 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"
```

設定ファイルの修正を反映するにはJobCenter AGを再起動する必要がありますので、設定ファイル修正後はECSタスクの再作成を行うようにしてください。

3.4. Amazon ECS(Fargate)環境のトラブルシューティング

3.4.1. エラーメッセージ一覧

エラーメッセージの詳細は「2.4.1.1 MGコンテナにおける障害」を参照してください。

4. Amazon EKS (Fargarte)環境での利用

本章ではAmazon EKS (Fargate)環境におけるJobCenterコンテナの利用について説明します。

4.1. 概要

Amazon　EKS(Fargate)環境でMGコンテナおよびAGコンテナを実行するための環境構築について例を用いて説明します。

ここでは図4.1「Amazon　　EKS(Fargate)上のJobCenter環境の構成例」で示した構成でJobCenter　　MGとJobCenter AGをデプロイする例を示します。

永続化領域としてEFSファイルシステムを利用し、EKSクラスタ外部からJobCenter　MGへアクセスできるようNLBを設定します。

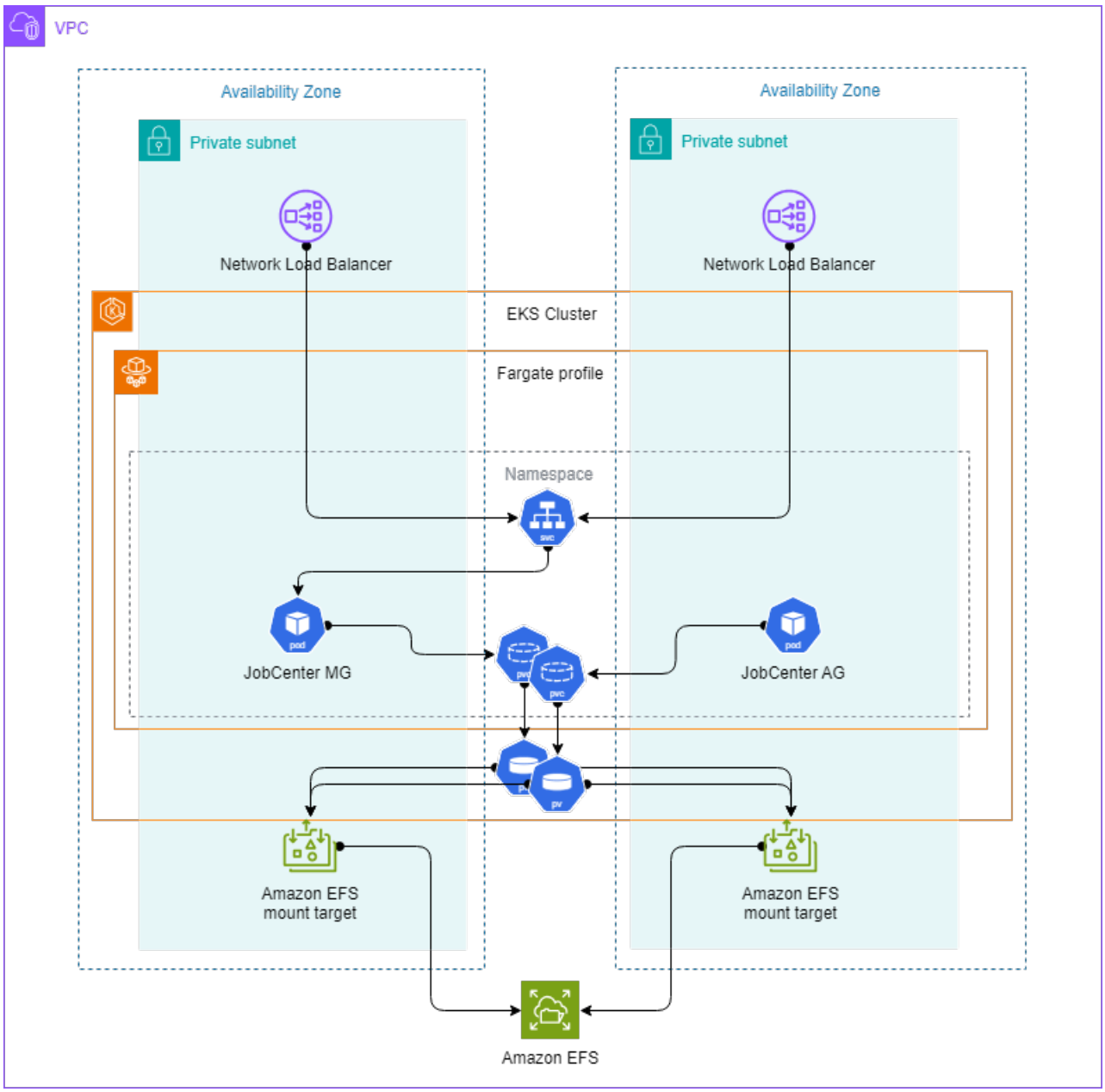

図4.1 Amazon EKS(Fargate)上のJobCenter環境の構成例

本書ではEKSクラスタを新規作成する手順を示しますが、すでにEKSクラスタを作成済みの場合は本節の記述を参考に必要な設定を行うようにしてください。

4.2. EKS環境の構築

構築作業のために以下のツールを利用するので事前に導入してください。

■AWS CLI

■eksctl

■kubectl

■helm

例示したコマンド例については環境に合わせて適宜修正してください。

以降は記載された順番で実施することを想定して記述しています。

4.2.1. EKSクラスタの作成

eksctl create clusterコマンドでEKSクラスタを作成します。--fargate オプションを指定してFargateをサポートするEKSクラスタを作成します。

```
eksctl create cluster --name ${CLUSTER_NAME} --region ${REGION_NAME} --fargate
```

上記コマンドは新規にVPC等のリソースも合わせて作成します。既存のVPCを利用する場合はeksctlコマンドのドキュメントを参照して適宜オプションを指定してください。

また、作成したEKSクラスタをkubectlコマンドで操作できるよう設定してください。

4.2.2. IAM OIDCプロバイダーの作成

作成したEKSクラスタのサービスアカウントからAWSリソースを制御するため、IAM OIDCプロバイダーの設定を行います。 下記のURLに記載された手順に従って設定を行ってください。

https://docs.aws.amazon.com/eks/latest/userguide/enable-iam-roles-for-service-accounts.html

4.2.3. Amazon EFS CSI Driverの導入

EKS(Fargate)環境では永続化ボリュームとしてEFSのみ利用することができます。 また、EFSを永続化ボリュームとして利用するには、事前に下記のURLに記載された手順に従いAmazon EFS CSI Driverを導入する必要があります。

https://docs.aws.amazon.com/eks/latest/userguide/efs-csi.html

実施する手順は以下の2つです。

■Creating an IAM role

■Installing the Amazon EFS CSI Driver

前者はeksctlとawsコマンドを利用して以下のように実施します。

```
$ ROLE_NAME=AmazonEKS_EFS_CSI_DriverRole
$ eksctl create iamserviceaccount \
    --name efs-csi-controller-sa \
    --namespace kube-system \
    --cluster ${CLUSTER_NAME} \
    --role-name ${ROLE_NAME} \
    --role-only \
    --attach-policy-arn arn:aws:iam::aws:policy/service-role/AmazonEFSCSIDriverPolicy \
    --approve
$ TRUST_POLICY=$(aws iam get-role --role-name ${ROLE_NAME} --query 'Role.AssumeRolePolicyDocument'
 | \
    sed -e 's/efs-csi-controller-sa/efs-csi-*/' -e 's/StringEquals/StringLike/')
$ aws iam update-assume-role-policy --role-name ${ROLE_NAME} --policy-document "${TRUST_POLICY}"
```

後者についてはEKSアドオンとして登録する方法が容易です。 以下のコマンドで導入可能なバージョンを確認することができます。

```
$ KUBERNETES_VERSION=1.29
$ eksctl utils describe-addon-versions --kubernetes-version ${KUBERNETES_VERSION} --name aws-efs-
csi-driver | grep AddonVersion
```

導入対象のバージョンを決定して以下のコマンドでEKSアドオンとして登録することができます。

```
$ DRIVER_VERSION=v2.0.1-eksbuild.1
```

```
$ eksctl create addon --cluster ${CLUSTER_NAME} --name aws-efs-csi-driver --version
 ${DRIVER_VERSION} --service-account-role-arn arn:aws:iam::${ACCOUNT_ID}:role/
AmazonEKS_EFS_CSI_DriverRole --force
```

以下のコマンドでefs-csi-controllerがデプロイされていることを確認できれば導入は完了です。

```
$ kubectl get deployment -n kube-system efs-csi-controller
```

4.2.4. AWS Load Balancer Controllerの導入

EKSクラスタ外のJobCenter AGやCL/WinからEKSクラスタ内のJobCenter MGへアクセスするにはNLBまたはALBを利用する必要があります。 そのため、EKSクラスタへAWS Load Balancer Controllerを導入する必要があります。 導入方法については下記のURLに記載されている手順に従ってください。

https://docs.aws.amazon.com/eks/latest/userguide/aws-load-balancer-controller.html

ここではHelmを利用した手順を示します。

```
# IAMポリシーの定義をダウンロード
$ curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.7.2/
docs/install/iam_policy.json

# IAMポリシーを作成
$ POLICY_NAME=AWSLoadBalancerControllerIAMPolicy
$ aws iam create-policy \
    --policy-name ${POLICY_NAME} \
    --policy-document file://iam_policy.json

# IAMロールを作成
$ ROLE_NAME=AmazonEKSLoadBalancerControllerRole
$ eksctl create iamserviceaccount \
  --cluster=${CLUSTER_NAME} \
  --namespace=kube-system \
  --name=aws-load-balancer-controller \
  --role-name ${ROLE_NAME} \
  --attach-policy-arn=arn:aws:iam::${ACCOUNT_ID}:policy/${POLICY_NAME} \
  --approve

# Helmを利用してAWS Loadbalancer Controllerを導入
$ helm repo add eks https://aws.github.io/eks-charts
$ helm repo update eks
$ helm install aws-load-balancer-controller eks/aws-load-balancer-controller \
  --set clusterName=${CLUSTER_NAME} \
  --set serviceAccount.create=false \
  --set region=${REGION_NAME} \
  --set vpcId=${VPC_ID} \
  --set serviceAccount.name=aws-load-balancer-controller \
  -n kube-system
```

以下のコマンドでaws-load-balancer-controllerがデプロイされていることを確認できれば導入は完了です。

```
$ kubectl get deployment -n kube-system aws-load-balancer-controller
```

4.3. JobCenter環境の構築

図4.1「Amazon EKS(Fargate)上のJobCenter環境の構成例」で示した構成でJobCenter MGとJobCenter AGをデプロイするための手順を示します。

4.3.1. コンテナイメージの準備

MGコンテナイメージとAGコンテナイメージを事前に作成してAmazon ECR等に登録してください。

MGコンテナイメージの作成については「5.2.1 MGコンテナイメージの作成」を参照して実施してください。

AGコンテナイメージの作成については「2.2.2 AGコンテナイメージの作成」を参照して実施してください。

4.3.2. EFSファイルシステムの作成

JobCenter　MGとJobCenter　AG(1台)の永続化領域としてEFSファイルシステムの作成を行います。ここではEFSファイルシステムを一つ作成してJobCenter　MG用とJobCenter　AG用のアクセスポイントを作成します。EFSの設定方法についてはAWSのドキュメントを適宜参照してください。

以下のようなコマンドラインでEFSファイルシステムを作成します。

```
$ CREATION_TOKEN=`uuidgen`
$ aws efs create-file-system --creation-token ${CREATION_TOKEN} --performance-mode generalPurpose
 --throughput-mode bursting --encrypted
```

マウントターゲットに割り当てるセキュリティグループを作成します。

```
$ SECURITY_GROUP_NAME=eksefssecgroup
$ aws ec2 create-security-group --group-name ${SECURITY_GROUP_NAME} --description "Allow EKS
 Cluster to access EFS File System" --vpc-id ${VPC_ID} --tag-specifications "ResourceType=security-
group,Tags=[{Key=Name,Value=${SECURITY_GROUP_NAME}}]"
```

各サブネットのインバウンドルールを追加します。サブネットが複数ある場合はサブネット分だけ実施します。

```
$ SECURITY_GROUP_ID=sg-xxxxxxxxxxxxxxxx
$ CIDR_IP=xxx.xxx.xxx.xxx/xx
$ aws ec2 authorize-security-group-ingress --group-id ${SECURITY_GROUP_ID} --ip-permissions
 IpProtocol=tcp,FromPort=2049,ToPort=2049,IpRanges=[{CidrIp=${CIDR_IP}}]
```

各サブネットごとにマウントターゲットを作成します。サブネットが複数ある場合はサブネット分だけ実施します。

```
$ FILE_SYSTEM_ID=fs-xxxxxxxxxxxxxxxx
$ SUBNET_ID=subnet-xxxxxxxxxxxxxxxx
$ SECURITY_GROUP_ID=sg-xxxxxxxxxxxxxxxx
$ aws efs create-mount-target --file-system-id ${FILE_SYSTEM_ID} --subnet-id ${SUBNET_ID} --
security-group ${SECURITY_GROUP_ID}
```

JobCenter MG用、JobCenter AG用それぞれのアクアスポイントを作成します。

```
$ FILE_SYSTEM_ID=fs-xxxxxxxxxxxxxxxx
$ aws efs create-access-point --file-system-id ${FILE_SYSTEM_ID} --root-directory "Path=/
mg,CreationInfo={OwnerUid=0,OwnerGid=11,Permissions=775}"
$ aws efs create-access-point --file-system-id ${FILE_SYSTEM_ID} --root-directory "Path=/
ag,CreationInfo={OwnerUid=0,OwnerGid=11,Permissions=775}"
```

4.3.3. ストレージクラスの作成

EFSファイルシステムを永続化ボリュームとして利用するため、ストレージクラスを作成します。　設定の詳細についてはAWS EFS CSI Driverのドキュメントを参照してください。

以下のようにマニフェストを作成します。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: efs-sc
provisioner: efs.csi.aws.com
```

作成したマニフェストを適用します。

```
$ kubectl apply -f sc.yaml
```

以下のコマンドでefs-scが作成されたことを確認できればストレージクラスの作成は完了です。

```
$ kubectl get sc
```

4.3.4. 永続化ボリュームの作成

作成したEFSファイルシステムのアクセスポイントと対応する形で永続化ボリュームを作成します。　以下のようにマニフェストを作成します。EFSファイルシステムとアクセスポイントのIDを適切に指定してください。

```
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mg-pv
spec:
  capacity:
    storage: 10Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  storageClassName: efs-sc
  csi:
    driver: efs.csi.aws.com
    volumeHandle: [FILE_SYSTEM_ID]::[ACCESS_POINT_ID_MG]
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: ag-pv
spec:
  capacity:
    storage: 10Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  storageClassName: efs-sc
  csi:
    driver: efs.csi.aws.com
    volumeHandle: [FILE_SYSTEM_ID]::[ACCESS_POINT_ID_AG]
```

作成したマニフェストを適用します。

```
$ kubectl apply -f pv.yaml
```

4.3.5. namespaceとFargate profileの作成

JobCenter MGおよびJobCenter AGの関連リソースを登録するnamespaceと、namespaceに対応するFargate profileを作成します。

```
$ NAMESPACE=jobcenter
$ kubectl create namespace ${NAMESPACE}
$ eksctl create fargateprofile --namespace ${NAMESPACE} --cluster ${CLUSTER_NAME} --name fp-
${NAMESPACE}
```

4.3.6. JobCenter MGの構築

MG用に以下のオブジェクトを作成します。

■コンフィグマップ(ライセンス情報)

■永続ボリューム要求

■デプロイメント

■サービス(NLBでCL/Win・エージェント接続用ポートを公開)

以下のようにマニフェストを作成します。ここでは一つのYAMLファイルとしていますが、分割しても問題ありません。 [...]の部分やライセンス情報は環境に合わせて書き換えてください。 また、annotationに記述するNLBの設定についてはAWS Load Balancer Controllerのドキュメントを参照してください。

```
---
# ライセンス情報
apiVersion: v1
kind: ConfigMap
metadata:
  name: mg-lockinfo-configmap
data:
  lockinfo: |
    UL1256-P00-I XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
---
# spool領域の永続ボリューム要求
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mg-spool-claim
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: efs-sc
  resources:
    requests:
      storage: 10Gi
  volumeName: mg-pv
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mg-deployment
  labels:
```

```
    app: jobcenter-mg
spec:
  replicas: 1
  selector:
    matchLabels:
      app: jobcenter-mg
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: jobcenter-mg
    spec:
      hostname: mg
      containers:
      - name: mg
        imagePullPolicy: IfNotPresent
        image: [IMAGE]
        env:
        - name: JOBCENTER_MACHINE_ID
          value: "1001"
        - 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:
        - mountPath: /etc/opt/wsnlesd/.lockinfo
          name: mg-lockinfo-volume
          subPath: lockinfo
        - mountPath: /usr/spool/nqs
          name: mg-spool-volume
          subPath: nqs
      volumes:
      - name: mg-spool-volume
        persistentVolumeClaim:
          claimName: mg-spool-claim
      - name: mg-lockinfo-volume
        configMap:
          name: mg-lockinfo-configmap
          defaultMode: 420
---
# CL/Win, AG接続用ポートの公開
apiVersion: v1
```

```
kind: Service
metadata:
  name: mg-service
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-type: "external"
    service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: "ip"
    service.beta.kubernetes.io/aws-load-balancer-healthcheck-interval: "300"
spec:
  type: LoadBalancer
  selector:
    app: jobcenter-mg
  ports:
    - name: clwin
      protocol: TCP
      port: 611
      targetPort: 611
    - name: agent
      protocol: TCP
      port: 23151
      targetPort: 23151
```

作成したマニフェストを適用します。

```
$ kubectl apply -f mg.yaml -n ${NAMESPACE}
```

4.3.7. CL/Winの接続

構築したJobCenter MGにCL/Winで接続するには以下のどちらかの方法を取ります。

■NLB経由での接続

■kubectl port-forwardを利用

前者はEKSクラスタが利用するVPCのサブネットに接続するEC2インスタンスを作成し、EC2インスタンス上にCL/Winをインストールしてリモートデスクトップで利用する形となります。

後者はCL/Winをインストールしたローカル環境から接続可能なホスト上で以下のようにポートフォワードの設定を行って接続します。

```
$ MG_POD_NAME=mg-deployment-xxxxxxxxxx-xxxxx
$ kubectl port-forward --address 0.0.0.0 -n jobcenter ${MG_POD_NAME} 10611:611
```

4.3.8. JobCenter AGの構築

EKSクラスタ上にAGを構築するために以下のオブジェクトを作成します。　AG名や認証用の秘密鍵を指定する必要があるので、事前にMGへAGを登録しておく必要があります。

■コンフィグマップ(ライセンス情報とAG認証用の秘密鍵)

■永続ボリューム要求

■デプロイメント

以下のようにマニフェストを作成します。ここでは一つのYAMLファイルとしていますが、分割しても問題ありません。 [...]の部分やライセンス情報、秘密鍵情報は環境に合わせて書き換えてください。

```
---
# ライセンス情報
apiVersion: v1
kind: ConfigMap
metadata:
```

```
  name: ag-lockinfo-configmap
data:
  lockinfo: |
    UL1256-P01-I XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
---
# AG認証用の秘密鍵
apiVersion: v1
kind: ConfigMap
metadata:
  name: ag-privkey-configmap
data:
  privkey: |
    -----BEGIN RSA PRIVATE KEY-----
    MIIEpgIBAAKCAQEA2OmVDC2alyDyOtI/H8ENMziwnNfZenfbmUsNXtp6g8INnv/J
    W4vxlSEI/YUep83hUy7DAhiOk4Ak/FdMCiNrZ9MOl3SeJU+90qCLY//qfioxTVpZ
    z7PTp4xC9jrHiJAumPN6slo3CDzb/IIbDPaWxAFpRLyZFVsNIobaF4dEIZEGK7o0
    A6T0YWoUt7Z++WUlin+LmkiGOdOOpYPM+/42+Zq8rpS03dHDkSI5FOysszQbU4TX
    LwJPWh6Sb3BbHTvM01LSSwJHr32wE4rcS4FQBrRfCq2xrcDNEixmSu3KMNFywBZP
    LUsxp+JJ3WUkcbby1Y52lwncL4XnLKUy82uekwIDAQABAoIBAQCguqS4+2nbpowX
    Tgd8Th6r38vuzHrYJsxQEK79pEK4MyjIspzP+yit6maxBN0sVoAqlTHm8a9kWMYP
    PdiYgpppupjLXZx55VaxJkljdhSaHEWgHd0C7Stesa5EqlWEOeJtb61xTywMLJyN
    xRKqTmZ1It6QEngUvP1EokwktxSjEWdoohcFjMZmyG9w/yu35szultSefvQ6balP
    SZmQlFrQMO/EUk7Z1M06jh03R3mkGkBAOJ79SlIfAg3u2Mzq1k3B4C3hHLCLcvgJ
    jzzoQ0LPLxqHe1JTTCS6cG26r9CPuVynWe8sFD2K+7FVPCmXSDRTB1WzfjHxank3
    KsgtOk0xAoGBAPrqCjlt7JfmARb6lOvY4d1PmLxDy4+Uc1J1G2WHCUWnY665yYB1
    PUzc8+mXHtL47z+Euj5r3jxdhOLx2oyIYI74zs8/AIoTBfvA/LS8lJexgRn5wyaw
    gKzJw4Aub3qvYwZrOZbcYFmBDz7WuKkVfprQl1CNXEFnjDA4y8ABhN/FAoGBAN1P
    HJC45LeDvB+Qf5sfp/UnF5ZIDoT2MLfbKTw1Jk8p217xUGmDlT0BEMe4D/OfvigT
    MPRaCgZkH5GmA7taQDCZjHVUIUb6ZlT0zFi2o98LkR3U56pnoQ9z6mEaSB9TU4PO
    t7iXEzdh1AsCuQNvGqVIv+XzpLLVEWQzX1HcONJ3AoGBAIRuTmMtT3iEnwKf00X/
    vIZwYm7LYX8wnQViZclaUlx/Eq+/w0iXmHkfDWW2q0AyvyPJS/tyN2OpAGA/wlf2
    CvCrBYyA9inqDHhpzXcfKVY/C6LIHmYnNmL8QsE+wPwB4cQbh9PGENxsr708203y
    2FWhzptnYM/sdSNmdrOqmZT5AoGBAIuC7b0F8ZkGgLddPjQqFpVi2WmxbwW+tSOS
    E0KTNt08w7/sXAbIHnJgk5VzoQqxrwxQXNINWLiCsav9fejDkieRBIyKRpgaDhyM
    Xa2nJVLcVdT6fDheL5gwOJaSHNkyjDlRPVwuULze/O8b/kn95kw/uklZ+H+L9fS2
    teZC78ctAoGBALHoYmkKqWGLbmAGMUwrJa+3323N5eAwqc6QgAiAP9qM0cHWSSMK
    HtQx9kPy05Wq9gwZWs+ZNZQTbqJPPUZ725+lzya0tK8iRqAUFpRHD6Vz3r9m+Oqh
    x7aiDNL3ZAcIBA9ZGdlj7vVvuWYRNBw0cfF4R+ghxzMg8O2X4uPal9X+
    -----END RSA PRIVATE KEY-----
---
# AGデータ領域用の永続ボリューム要求
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: ag-data-claim
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: efs-sc
  resources:
    requests:
      storage: 10Gi
  volumeName: ag-pv
---
apiVersion: apps/v1
kind: Deployment
```

```
metadata:
  name: ag-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: jobcenter-ag
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: jobcenter-ag
    spec:
      hostname: [AG_NAME]
      containers:
      - name: ag
        imagePullPolicy: IfNotPresent
        image: <image>
        env:
        - name: JOBCENTER_AGENT_NAME
          value: "[AG_NAME]"
        - name: JOBCENTER_INSTANCE_NAME
          value: "[AG_NAME]"
        - name: JOBCENTER_LANGUAGE_CODE
          value: "UTF-8"
        - name: JOBCENTER_ENDPOINT_URL
          value: "http://mg-service:23151"
        - name: JOBCENTER_PRIVKEY_PATH
          value: "/[AG_NAME].pem"
        volumeMounts:
        - mountPath: /etc/opt/wsnlesd/.lockinfo
          name: ag-lockinfo-volume
          subPath: lockinfo
        - mountPath: /[AG_NAME].pem
          name: ag-privkey-volume
          subPath: privkey
        - mountPath: /jcag
          name: ag-data-volume
          subPath: jcag
      volumes:
      - name: ag-data-volume
        persistentVolumeClaim:
          claimName: ag-data-claim
      - name: ag-lockinfo-volume
        configMap:
          name: ag-lockinfo-configmap
          defaultMode: 420
      - name: ag-privkey-volume
        configMap:
          name: ag-privkey-configmap
          defaultMode: 420
```

作成したマニフェストを適用します。

```
$ kubectl apply -f ag.yaml -n ${NAMESPACE}
```

4.3.9. ALBを利用したJobCenter AGの接続

ここまでに示した例ではJobCenter MGのCL/Win接続用ポートやAG接続用ポートをNLB経由でアクセスできるように設定しましたが、AG接続用ポートについてはALBでの公開も可能です。 ALBで公開する場合は以下のようなマニフェストを作成して適用します。 annotationに記述するNLBの設定についてはAWS Load Balancer Controllerのドキュメントを参照してください。

```
---
apiVersion: v1
kind: Service
metadata:
  name: mg-service
spec:
  type: NodePort
  selector:
    app: jobcenter-mg
  ports:
    - name: agent
      protocol: TCP
      port: 23151
      targetPort: 23151
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: mg-ingress
  annotations:
    alb.ingress.kubernetes.io/scheme: internal
    alb.ingress.kubernetes.io/target-type: ip
spec:
  ingressClassName: alb
  rules:
    - http:
        paths:
        - path: /
          pathType: Prefix
          backend:
            service:
              name: mg-service
              port:
                number: 23151
```

4.3.10. エージェントの設定ファイル修正

本作業は必要に応じて任意で実施してください。

コンテナAGの動作を一部動作を変更するには、永続化されたデータ領域に格納される設定ファイル(jcexecutor_agent.yaml)を修正する必要があります。

特に、Amazon EKS(Fargate)上で実行しているコンテナAGから拡張カスタムジョブのAWS連携部品についてIAMRoleを利用して実行する場合は `request.environment.agentEnvironment`パラメータを`true` に設定する必要があるため注意してください。

設定ファイルの詳細は<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>の「3.1 エージェントの動作設定変更」を参照してください。

設定ファイルはコンテナ内の以下のパスに作成されるので初回起動後に編集してください。

```
/jcag/spool_${JOBCENTER_INSTANCE_NAME}/jcexecutor/agent/jcexecutor_agent.yaml
```

起動中のAGコンテナへ接続して直接編集する場合は以下のように対話シェルを実行して作業を行ってください。

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

設定ファイルの修正を反映するにはJobCenter AGを再起動する必要がありますので、設定ファイル修正後はDeploymentの再作成を行うようにしてください。

4.4. AWS EKS(Fargate)環境のトラブルシューティング

4.4.1. エラーメッセージ一覧

エラーメッセージの詳細は「2.4.1.1 MGコンテナにおける障害」を参照してください。

5. Amazon EKS環境での利用

本章ではAmazon EKS環境におけるJobCenterコンテナの利用方法について説明します。

5.1. 概要

Amazon EKS環境でMGコンテナを実行するための環境構築について例を用いて説明します。

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

5.2. 事前準備

5.2.1. MGコンテナイメージの作成

EKSで利用するMG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 ECR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。

MGコンテナイメージの作成手順については「2.2.1 MGコンテナイメージの作成」を参照してください。

ただし、コンテナイメージをビルドする前にDockerfileに対して以下の修正を実施してください。

■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。

5.3. 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。

5.3.1. MGのマニフェストファイルの作成

5.3.1.1. 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。「5.2.1 MGコンテナイメージの作成」 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。

5.4. 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。

5.4.1. 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	「5.4.1.1 管理者ユーザのパスワードの設定」
マシンIDの設定	「5.4.1.2 マシンIDの設定」
一般ユーザの設定	「5.4.1.3 一般ユーザの作成」
ポート番号の変更	「5.4.1.4 ポート番号の変更」
セットアップ言語の設定	「5.4.1.5 セットアップ言語の設定」
EKS外部のマシンとの接続	「5.4.1.6 EKS外部のマシンとの接続」
起動時の設定ファイルの設定	「5.4.1.7 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	「5.4.1.8 証明書・秘密鍵ファイルの設定」

5.4.1.1. 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. 「2.3.1.10 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>
```

5.4.1.2. 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。

5.4.1.3. 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. 「2.3.1.9.1 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>
```

5.4.1.4. 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>
```

5.4.1.5. 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

5.4.1.6. 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。

5.4.1.7. 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。

5.4.1.8. 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。

5.5. リソースの作成

「5.3 マニフェストファイルの作成」および「5.4 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
```

5.6. 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。

5.6.1. 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
```

「5.4.1.6 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、「5.4.1.6　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。

5.6.2. 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
```

「5.4.1.6 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、「5.4.1.6　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。

5.6.3. 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
```

「5.4.1.6 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、「5.4.1.6　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。

5.6.4. 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
```

5.7. AWS EKS環境のトラブルシューティング

5.7.1. エラーメッセージ一覧

エラーメッセージの詳細は「2.4.1.1 MGコンテナにおける障害」を参照してください。

発行年月 May 2025