クラシックモード用コンテナ ガイド

JobCenter

R16.3

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

→ <クラシック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

→ <クラシックモード用インストールガイド>を参照してください。

■JobCenter をコンテナ環境で構築、運用をする場合

→ <クラシックモード用コンテナガイド>を参照してください。

■JobCenter を初めて利用される場合

→ <クラシックモード用クイックスタート編>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

→ <クラシックモード用基本操作ガイド>を参照してください。

■環境の構築や各種機能の設定を理解したい場合

→ <クラシックモード用環境構築ガイド>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

→ <クラシックモード用コマンドリファレンス>を参照してください。

■JobCenter の運用方法を理解したい場合

→ <クラシックモード用運用・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

→ <クラシックモード用移行ガイド>を参照してください。

■クラスタ環境で運用中のJobCenter をバージョンアップする場合

→ <クラシックモード用クラスタ環境でのバージョンアップ・パッチ適用ガイド>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/SV	JobCenter MG/SVの略称
CL/Web	JobCenter CL/Webの略称
MG/SVコンテナ	JobCenter MG/JobCenter SV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
CL/Webコンテナ	JobCenter CL/Web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
システムコンテナ	複数のアプリケーションが1つのコンテナ内に同居し、Systemdによってサービスが管理されてい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SV	コンテナ上で動作しているJobCenter SVを指します
コンテナCL/Web	コンテナ上で動作しているJobCenter CL/Web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/SV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。
CR	KubernetesのリソースであるCustomResourceのことを指します。
OpenShift管理者	OpenShiftのユーザ権限として、ClusterRoleのcluster-admin相当の権限を所持しているユーザを指します。

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<クラシックモード用リリースメモ>の「3.1.7 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。

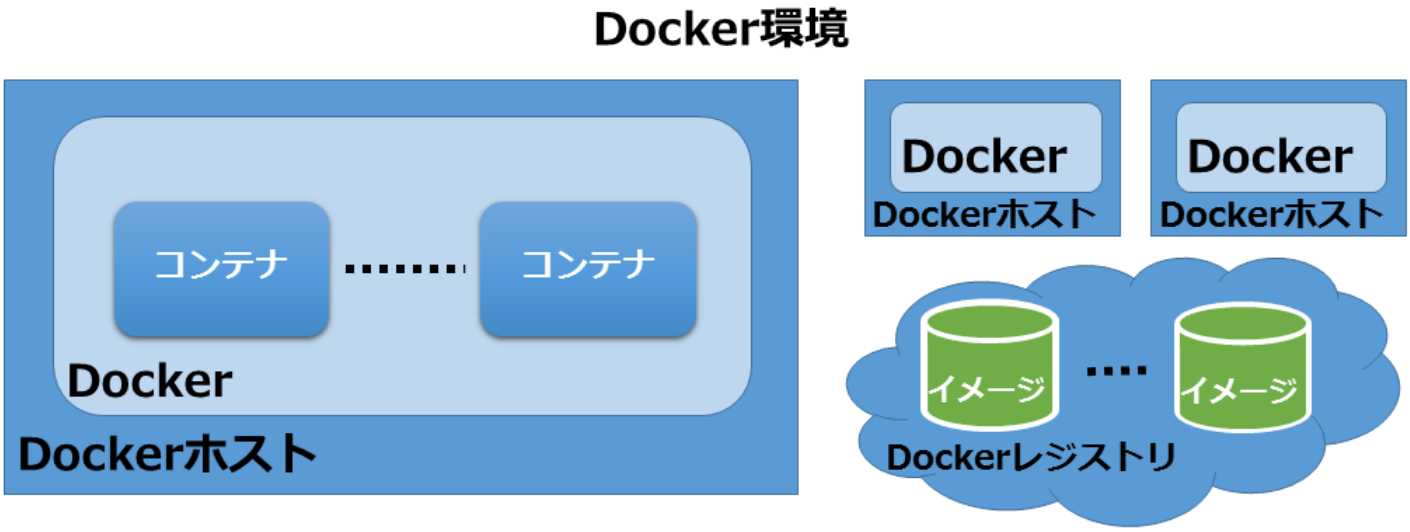

図2.1 Docker環境

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。

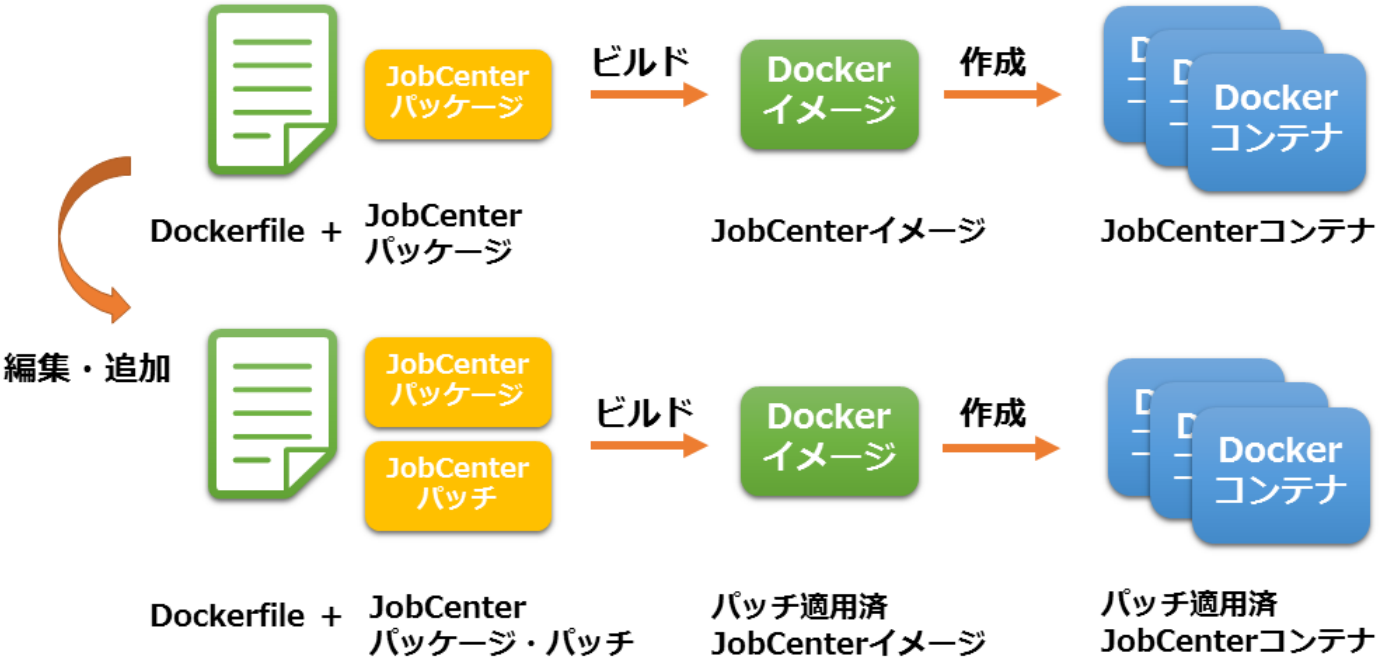

図2.2 イメージのライフサイクル

■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「2.4.3 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。

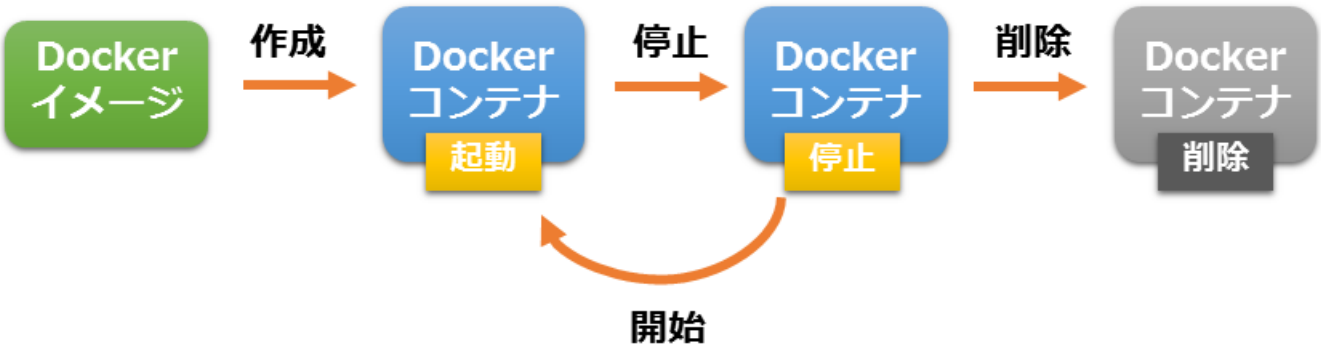

図2.3 コンテナのライフサイクル

■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「2.4.3 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をデプロイする方法によって、次の2種類のコンテナを作成できます。

■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/SVのアプリケーションコンテナをMG/SVコンテナ、CL/Webのアプリケーションコンテナを CL/Webコンテナと呼称します。

■システムコンテナ

コンテナ内に複数のアプリケーションが存在するコンテナです。それぞれのアプリケーションは、Systemd によって管理します。JobCenterとほかのアプリケーションを1つのコンテナ内に同居して運用することで、仮想マシンの軽量版としてコンテナを使用できます。

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。

コンテナMGは、コンテナSVと連携できます。またコンテナSVだけではなく、物理環境のMG/SVとも連携できます。

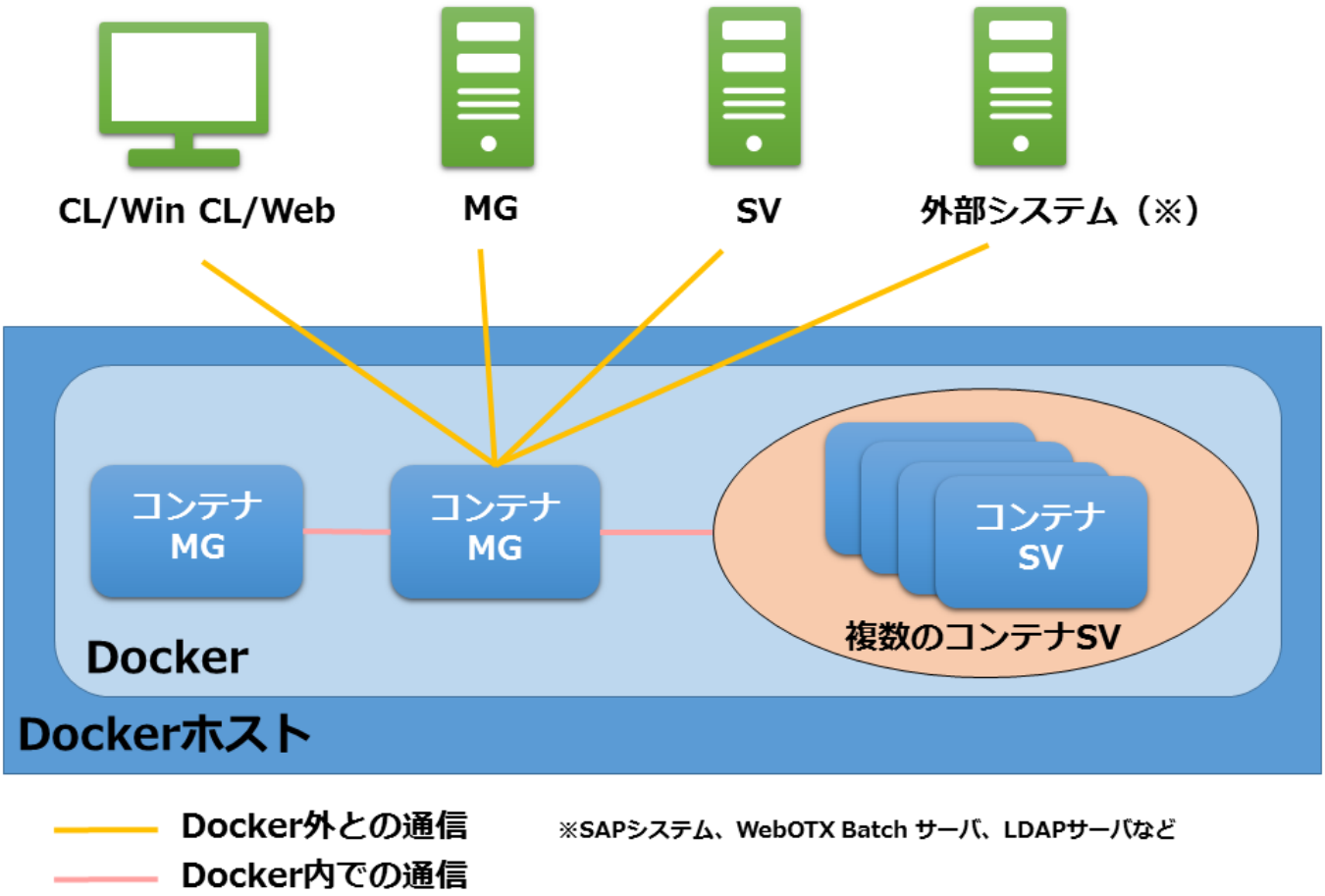

図2.4 コンテナMGを中心とした構成図

2.2.1.2. SVの構成

コンテナSVを中心としたJobCenterの構成図は図2.5「コンテナSVを中心とした構成図」のとおりです。

コンテナSVは、コンテナMGおよび物理環境のMGと連携できます。コンテナSVは1台のDockerホストで複数起動できるため、仮想マシンを利用したサーバ仮想化よりも多くのSVを集約できます。

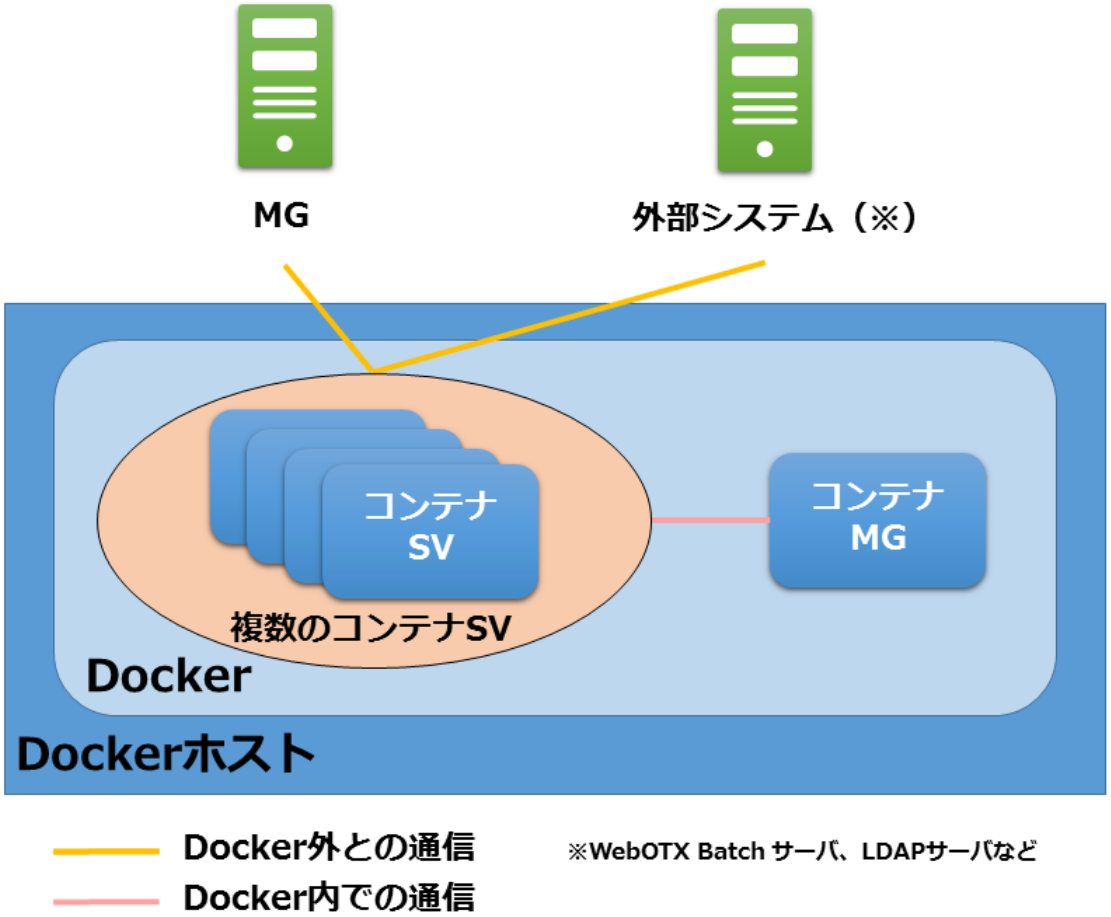

図2.5 コンテナSVを中心とした構成図

2.2.1.3. CL/Webの構成

コンテナCL/Webは、コンテナMG/SVおよび物理環境のMG/SVに対して接続できます。

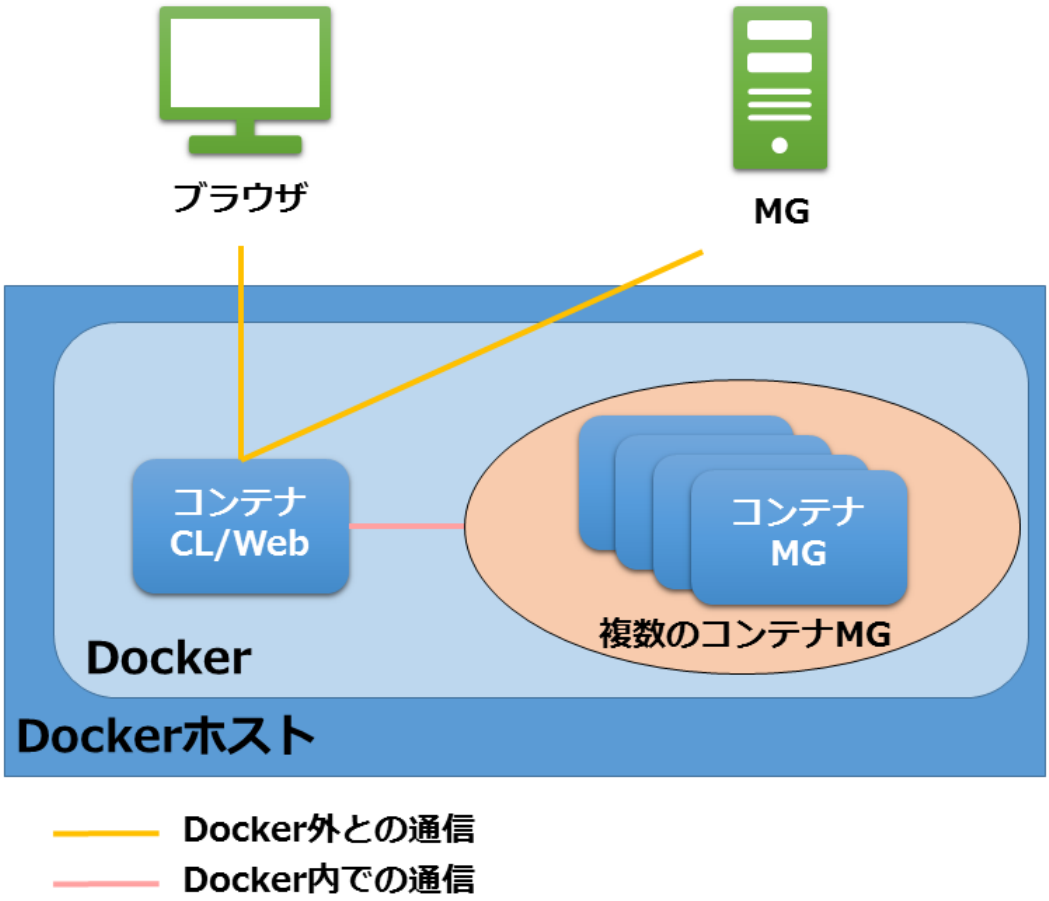

図2.6 コンテナCL/Webを中心とした構成図

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。

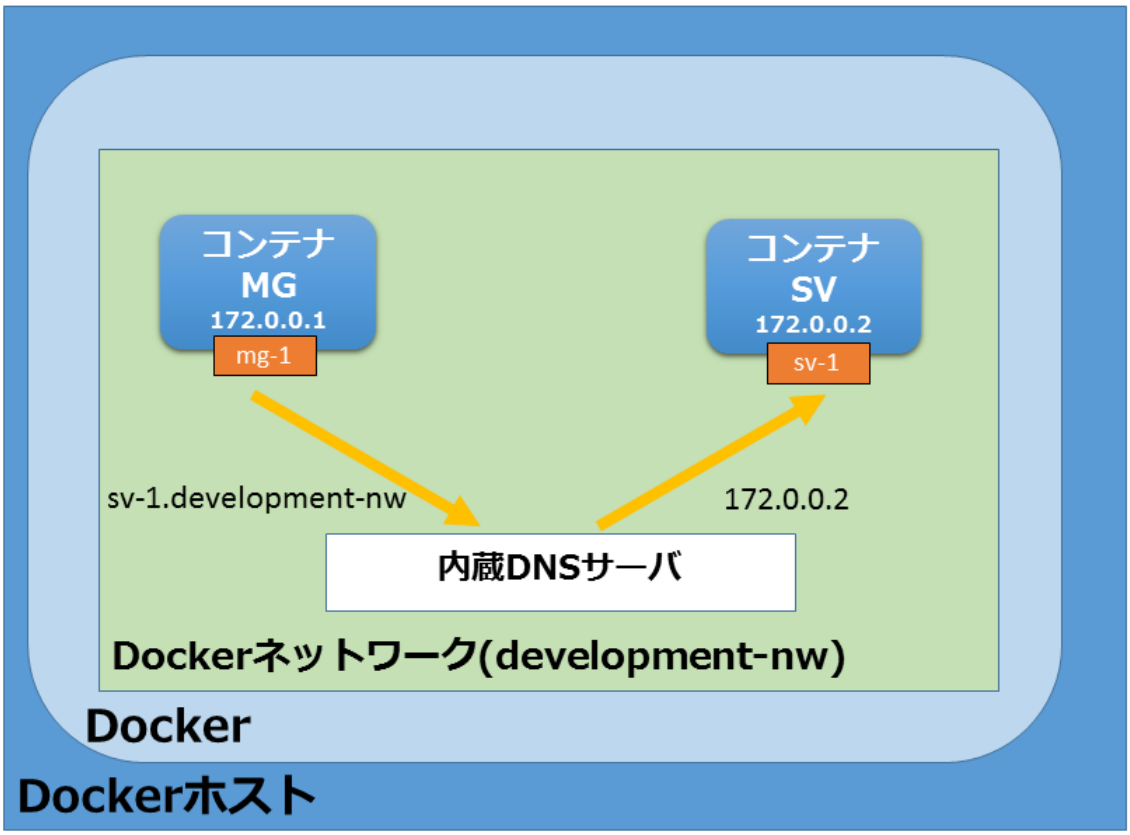

図2.7 Docker内部での通信

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。

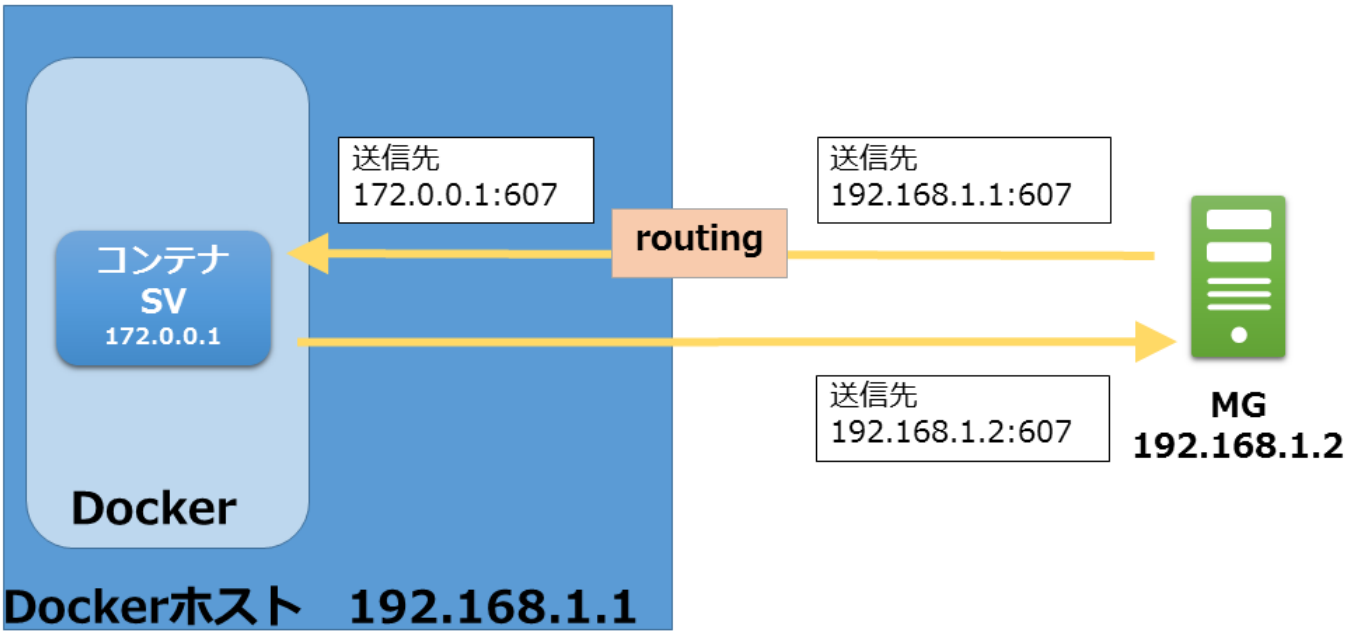

図2.8 Docker外部との通信

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。

Dockerホスト上の同一のIPアドレスに対して、同じポート番号をbindすることはできないため、複数のコンテナSVを立ち上げてMGと通信を行う場合、コンテナSVにつきDockerホスト上のIPアドレスが1つ必要になります。

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/SV	CL/Web
イメージ作成時	「2.2.2.1.1 ipcheck=OFFの設定」	○	×
コンテナ作成前	「2.2.2.1.2 Dockerネットワークの作成」	○	○
コンテナ作成時	「2.2.2.1.3 コンテナの設定」	○	○

（凡例）○：作業が必要　×：作業が不要

2.2.2.1.1. ipcheck=OFFの設定

コンテナMGがコンテナSVとマシン連携を行う場合、コンテナSVのdaemon.confに、ipcheck=OFFのパラメータを設定する必要があります。

「2.3.1.1 Dockerfileの作成」の「コンテナ作成時に、JobCenter起動時の設定を適用させたい場合」を参照して、あらかじめdaemon.confにipcheck=OFFが設定されたイメージを作成してください。

daemon.confおよびipcheck=OFFの詳細は、<クラシックモード用環境構築ガイド>の5章「JobCenter起動時の設定を変更する」 を参照してください。

2.2.2.1.2. 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、「2.4.1 Dockerネットワークの作成」を参照してください。

2.2.2.1.3. 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/SVコンテナの場合は「2.4.3.1.11 Docker内での通信の設定」を、

CL/Webコンテナの場合は「2.4.3.2.7 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/SV	CL/Web
コンテナ作成時	「2.2.2.2.1 コンテナの設定」	○	○
コンテナ作成後	「2.2.2.2.2 ipcheck=OFF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/SVコンテナの場合は「2.4.3.1.12 Docker外との通信の設定」を、

CL/Webコンテナの場合は「2.4.3.2.8 Docker外との通信の設定」を参照してください。

2.2.2.2.2. ipcheck=OFFの設定

コンテナMGが物理環境のSVとマシン連携を行う場合、SVのdaemon.confに、ipcheck=OFFのパラメータを設定する必要があります。

連携するSVのdaemon.confファイルを編集して、ipcheck=OFFを記載してください。

daemon.confおよびipcheck=OFFの詳細は、<クラシックモード用環境構築ガイド>の5章「JobCenter起動時の設定を変更する」 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/SVコンテナを起動して、Docker外のMG/SVとマシン連携を行うような運用をする場合、「2.4.3.1.12.1 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/SV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SVで通信を行う場合、Docker内での通信を利用してください。「2.4.3.1.12.1 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、マシン連携に失敗します。

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、「2.4.2 マウント元データの作成」を行います。
4. 「2.4.3 コンテナの作成/起動」を行います。

   MG/SVコンテナの場合は「2.4.3.1.9 JobCenterのデータの永続化」を、

   CL/Webコンテナの場合は「2.4.3.2.5 CL/Web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/SVで永続化できるデータ」、表2.2「CL/Webで永続化できるデータ」を参照してください。

表2.1 MG/SV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、<クラシックモード用リリースメモ>の「3.2.3.1 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、<クラシックモード用環境構築ガイド>の5章 「JobCenter起動時の設定を変更する」 を参照してください。
マシン連携時の設定	/etc/hosts.equiv	ファイル	リモートシェルの実行権を与える対象を設定するために利用するファイルです。ユーザ名によるユーザマッピングを利用する場合は設定が必要です。 本ファイルの設定に関する詳細は、<クラシックモード用NQS機能利用の手引き>の「6.5.1.2 ユーザに関するネットワーク環境」を参照してください。
	/home/<ユーザ名>/.rhosts	ファイル	
	/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。
	/etc/profile	ファイル	リモートジョブ投入を行った際に、MGからSV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、<クラシック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	
	/usr/lib/nqs/codecnv.cnf	ファイル	文字コードがEUCとShift-JISのマシン同士が通信を行う際に、文字コードの変換を行うためのファイルです。 本ファイルの設定に関する詳細は、<クラシックモード用環境構築ガイド>の「9.2.1.1 SJIS側のUNIX版JobCenterの文字コード変換を設定する」を参照してください。

永続化によって引き継げるもの	ファイル名	種類	説明
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、<クラシック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、<クラシックモード用コマンドリファレンス>の「3.30 jcres JobCenter MG/SV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、<クラシックモード用環境構築ガイド>の「5.7 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、<クラシック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、<クラシックモード用環境構築ガイド>の「5.2 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、<クラシック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、<クラシックモード用環境構築ガイド>の「5.2 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、<クラシックモード用環境構築ガイド>の「5.7 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、<クラシックモード用環境構築ガイド>の「5.7 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

永続化によって引き継げるもの	ファイル名	種類	説明
			ラメータファイルを設定する」を参照してください。

destconf.fを永続化する際、sapnwrfc.iniファイルも同時に永続化したい場合、/usr/spool/nqsディレクトリの永続化を行います。「2.4.2 マウント元データの作成」の際に、sapnwrfc.iniファイルを配置したディレクトリをマウントしてください。

表2.2 CL/Web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
CL/Webサーバの設定	/usr/local/jcclweb/config/clweb.conf	ファイル	CL/Webの環境設定ファイルです。 本ファイルを永続化することで、CL/Webの環境設定を、引き継ぐことができます。 clweb.confについては、R16.2の<クラシックモード用Web機能利用の手引き>の「3.1.1 CL/Web環境設定ファイル(clweb.conf)」を参照してください。
	/usr/local/jcclweb/config/ssl_cert	ファイル	CL/Webサーバの証明書ファイルです。 本ファイルを永続化することで、R16.2の<クラシックモード用Web機能利用の手引き>の「3.1.3 SSL署名証明書の設定」で作成したCL/Webサーバの証明書を、引き継ぐことができます。
	/usr/local/jcclweb/config/ssl_key		
CL/Webの設定	/usr/local/jcclweb/config/mypage	ディレクトリ	マイページの情報が保存されているディレクトリです。 本ディレクトリを永続化することで、R16.2の<クラシックモード用Web機能利用の手引き>の「5.1.6 マイページ機能」で設定したマイページの設定を、引き継ぐことができます。
	/usr/local/jcclweb/config/mail	ディレクトリ	メールテンプレートファイルを保存するディレクトリです。 本ファイルを永続化することで、R16.2の<クラシックモード用Web機能利用の手引き>の「7.1.4 メールテンプレート設定」で作成したメールテンプレートファイルを、引き継ぐことができます。
APIの設定	/usr/local/jcclweb/config/clwebkeys.db	ファイル	APIアクセスキーの情報が保存されているファイルです。 本ファイルを永続化することで、APIアクセスキーの値を、引き継ぐことができます。 APIアクセスキーについては、R16.2の<クラシックモード用Web機能利用の手引き>の「6.4.1 APIアクセスキー認証」を参照してください。
ログデータ	/usr/local/jcclweb/log	ディレクトリ	ログファイルが保存されているディレクトリです。 本ディレクトリを永続化することで、R16.2の<クラシックモード用Web機能利用の手引き>の「7.5 証跡ログ機能」で保存されるログファイルを、引き継ぐことができます。

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/SVで永続化できるデータ」表2.2「CL/Web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。

■MG/SVコンテナのイメージを作成する場合は、「2.3.1　MG/SVコンテナのイメージの作成」を参照してください。

■システムコンテナのイメージを作成する場合は、「2.3.3　システムコンテナのイメージの作成」を参照してください。

■CL/Webコンテナのイメージを作成する場合は、「2.3.2　CL/Webコンテナのイメージの作成」を参照してください。

2.3.1. MG/SV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/SVのインストールを行うため、MG/SV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。

2.3.1.1. 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（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daemon.conf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。

以下の手順に従って、daemon.confの準備とDockerfileの編集を行ってください。

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、daemon.confのテンプレートファイルを用意します。

2. daemon.confのテンプレートファイルに、適用したい設定を記載します。

作成したdaemon.confファイルは、「2.3.1.2 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。

3. 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 daemon.conf /usr/lib/nqs/rc/daemon.conf
```

daemon.confについては、<クラシックモード用環境構築ガイド>の5章　「JobCenter起動時の設定を変更する」 を参照してください。

2.3.1.2. 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/SVのパッケージ

■LicenseManagerのパッケージ

■（パッチを適用する場合）MG/SVのパッチ

■（daemon.confの設定を固定化する場合）daemon.conf

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。

2.3.2. CL/Web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CL/Webのインストールを行うため、CL/Web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。

■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,CLWEBPKG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 CLWEBPKG=<CL/Webパッケージ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。

■Javaのバージョンを変更したい場合

JobCenterが提供しているDockerfileでは、Red　Hatが提供しているOpenJDK　8をインストールしています。Javaのバージョンを11でCL/Webを動作させたい場合は、Dockerfileを編集してRed　Hatが提供しているOpenJDK 11またはOracle Java SE 11をインストールしてください。

■コンテナ作成時に、CL/Webサーバの設定を適用させたい場合

CL/Web環境設定ファイル（clweb.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clweb.conf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CL/Webサーバを再起動する手間を省けます。

以下の手順に従って、clweb.confの準備とDockerfileの編集を行ってください。

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、clweb.confのテンプレートファイルを用意します。

2. clweb.confのテンプレートファイルに、適用したい設定を記載します。

作成したclweb.conf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。

3. DockerfileのWORKDIR / 行からCMD ["/usr/local/jcclweb/run/clweb_ctrl.sh", "run"]行の間に、以下を記載します。

```
COPY clweb.conf /usr/local/jcclweb/config/clweb.conf
```

clweb.confについては、R16.2の<クラシックモード用Web機能利用の手引き>の「3.1.1 CL/Web環境設定ファイル(clweb.conf)」を参照してください。

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

■CL/Webのパッケージ

■LicenseManagerのパッケージ

■（clweb.confの設定を固定化する場合）clweb.conf

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. システムコンテナのイメージの作成

コンテナのrootプロセスを/sbin/initにした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/SV、CL/Webのインストールを行うため、インストールを行う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。

2.3.3.1. Dockerfileの作成

コンテナのrootプロセスを/sbin/initにしたDockerfileを作成する必要があります。

JobCenterが提供しているDockerfileをベースに作成する場合と、すでにあるDockerfileにJobCenterの必要な処理を記載して作成する場合の手順を説明します。

■JobCenterが提供しているDockerfileから作成する場合

1. MG/SVがインストールされているシステムコンテナを作成する場合は、MG/SVのDockerfileを用意します。CL/Webがインストールされているシステムコンテナを作成する場合は、CL/WebのDockerfileを用意します。

2. ENTRYPOINTは不要なため、以下の2行を削除します。

```
COPY entrypoint.sh /
ENTRYPOINT ["/entrypoint.sh"]
```

3. CMDで実行するプロセスを、/sbin/initに変更します。

```
CMD ["/sbin/init"]
```

4. 以下を記載して、docker container stop時に送信するシグナルをSIGRTMIN+3に変更します。

```
STOPSIGNAL SIGRTMIN+3
```

5. ほかのアプリケーションをシステムコンテナに同居させる場合、そのアプリケーションのインストールの処理を追記します。

■すでにあるDockerfileを利用して作成する場合

1. DockerfileのCMDで実行されているプロセスが、/sbin/initであることを確認します。
2. 以下を記載して、docker container stop時に送信するシグナルをSIGRTMIN+3に変更します。

```
STOPSIGNAL SIGRTMIN+3
```

3. JobCenterが提供しているDockerfileのうち、LABELが設定されている行から、WORKDIR /行までの内容を、Dockerfileに追記します。

2.3.3.2. イメージのビルド

Dockerfileからイメージのビルドを行います。

ビルドの手順については、MG/SVコンテナおよびCL/Webコンテナと同じです。

MG/SVの場合は「2.3.1.2 イメージのビルド」を、

CL/Webの場合は「2.3.2.2 イメージのビルド」を参照してください。

2.3.4. イメージの作成における注意事項

■MG/SV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/SV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：<クラシックモード用インストールガイド>の「2.3.2 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/SV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/SVのパッケージが存在しない
- CL/Webがインストールされたイメージを作成する際、指定したLicenseManager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やclweb.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

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。

1. Docker内のコンテナ間で通信を行う場合は、「2.4.1 Dockerネットワークの作成」を行います。

2. bind mountの方式でデータの永続化を行う場合は、「2.4.2 マウント元データの作成」を行います。

3. 「2.4.3 コンテナの作成/起動」を行います。

2.4.1. 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
```

2.4.2. 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/SV）」、表2.4「マウント元データの作成方法（CL/Web）」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/SV）

永続化するデータ	マウント元データの作成方法
/etc/hosts.equiv	<クラシックモード用NQS機能利用の手引き>の「6.5.1.2 ユーザに関するネットワーク環境」を参照して、ファイルを作成してください。
/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	Dockerホスト上に存在する/etc/profileファイルに、コンテナ内で使用したい設定を記載して、ファイルを作成してください。
/home/<ユーザ名>/.nsifrc	<クラシックモード用環境構築ガイド>の「15.1 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。
/home/<ユーザ名>/.rhosts	<クラシックモード用NQS機能利用の手引き>の「6.5.1.2 ユーザに関するネットワーク環境」を参照して、ファイルを作成してください。
/usr/lib/nqs/codecnv.cnf	<クラシックモード用環境構築ガイド>の「9.2.1.1 SJIS側のUNIX版JobCenterの文字コード変換を設定する」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	<クラシック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	<クラシックモード用環境構築ガイド>の5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	<クラシックモード用コマンドリファレンス>の「3.30 jcres JobCenter MG/SV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	<クラシックモード用環境構築ガイド>の「5.7 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/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	<クラシック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、<クラシックモード用環境構築ガイド>の「5.2 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	<クラシック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、<クラシックモード用環境構築ガイド>の「5.2 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	<クラシックモード用環境構築ガイド>の「5.7 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	<クラシックモード用環境構築ガイド>の「5.7 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/lib/nqs/sap/destconf.f	<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、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。

表2.4 マウント元データの作成方法（CL/Web）

永続化するデータ	マウント元データの作成方法
/usr/local/jcclweb/config/clweb.conf	R16.2の<クラシックモード用Web機能利用の手引き>の「3.1.1 CL/Web環境設定ファイル(clweb.conf)」を参照して、ファイルを作成してください。 clweb.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。
/usr/local/jcclweb/config/clwebkeys.db	空ファイルを作成してください。
/usr/local/jcclweb/config/mail	空ディレクトリを作成してください。作成したディレクトリ内に、R16.2の<クラシックモード用Web機能利用の手引き>の「7.1.4 メールテンプレート設定」で作成したファイルを配置してください。
/usr/local/jcclweb/config/mypage	空ディレクトリを作成してください。
/usr/local/jcclweb/config/ssl_cert	R16.2の<クラシックモード用Web機能利用の手引き>の「3.1.3 SSL署名証明書の設定」を参照して、ファイルを作成してください。
/usr/local/jcclweb/config/ssl_key	
/usr/local/jcclweb/log	空ディレクトリを作成してください。

2.4.3. 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/SVコンテナおよび、MG/SVをインストールしているシステムコンテナについては「2.4.3.1 MG/SVコンテナの作成/起動」を、

CL/Webコンテナおよび、CL/Webをインストールしているシステムコンテナについては「2.4.3.2 CL/Webコンテナの作成/起動」を参照してください。

2.4.3.1. MG/SV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2.5「MG/SV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/SV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>
```

■システムコンテナ

```
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 <イメージ名/イメージID>
```

表2.5 MG/SVのコンテナの設定

設定	MG/SVコンテナ	システムコンテナ
「2.4.3.1.1 バックグラウンドで起動」	◎	◎
「2.4.3.1.2 コンテナホスト名の設定」	◎	◎
「2.4.3.1.3 ライセンスの登録」	◎	◎
「2.4.3.1.4 initプロセスの設定」	◎	×
「2.4.3.1.5 JobCenter管理者ユーザのパスワードの設定」	◎	×
「2.4.3.1.6 JobCenterのセットアップ時の設定」	○	×
「2.4.3.1.7 JobCenterが使用するプロトコルのポート番号の変更」	○	×
「2.4.3.1.8 一般ユーザの作成」	○	×
「2.4.3.1.9 JobCenterのデータの永続化」	○	○
「2.4.3.1.10 永続化したデータの引き継ぎ」	○	○
「2.4.3.1.11 Docker内での通信の設定」	○	○
「2.4.3.1.12 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

2.4.3.1.1. 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/SV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。

2.4.3.1.2. 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「2.4.3.1.11 Docker内での通信の設定」を参照してください。

2.4.3.1.3. 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/SV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<クラシックモード用インストールガイド>の「2.3　コードワードを登録する」を参照してください。

システムコンテナにおいては、コンテナ作成後にコンテナ内の.lockinfoファイルに直接コードワードを記載してライセンス登録を行っても構いません。

2.4.3.1.4. 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>
   ```

2.4.3.1.5. 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2.6「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「2.4.3.4.2 パスワードの暗号化」を参照してください。

表2.6 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

2.4.3.1.6. 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2.7「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2.7 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

2.4.3.1.7. 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2.8「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2.8 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_NQS_PORT	nqsのポート番号	607	1 ～ 65535
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

各プロトコルの詳細については、<クラシック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。

2.4.3.1.8. 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。

■「2.4.3.1.8.1 パスワードファイルの作成」

■「2.4.3.1.8.2 パスワードファイルのマウント」

■「2.4.3.1.8.3 環境変数でマウント先のパスを指定」

2.4.3.1.8.1. 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「2.4.3.4.2 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。

■<クラシックモード用リリースメモ>の「6.1.1　使用不可ユーザ名について」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。JobCenterとして登録可能なユーザ数については<クラシックモード用環境構築ガイド>の「3.3.8 登録可能なユーザ数」を参照してください。

■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。「2.4.3.1.5　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/SV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。

2.4.3.1.8.2. 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。

2.4.3.1.8.3. 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、「2.4.3.1.8.2 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2.9「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2.9 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

2.4.3.1.9. 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/SV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。

2.4.3.1.10. 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/SV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。

「2.4.3.1.9 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。

2.4.3.1.11. 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。

■「2.4.3.1.11.1 ネットワークの指定」

■「2.4.3.1.11.2 コンテナ名の設定」

■「2.4.3.1.11.3 コンテナホスト名の設定」

2.4.3.1.11.1. 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。

「2.4.1 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。

2.4.3.1.11.2. 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。

2.4.3.1.11.3. 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

2.4.3.1.12. 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。

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

■「2.4.3.1.12.2 名前解決方法の設定」

2.4.3.1.12.1. 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/SVが使用するポートは以下です。

ポート番号	説明
607	nqsが使用するポート番号
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

「2.4.3.1.7 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。

ホスト側とコンテナ側で開放するポート番号は一致させる必要があります。違うポート番号を指定した場合、マシン連携の際にマシン一覧への登録に失敗します。

2.4.3.1.12.2. 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、「2.4.3.1.12.1 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、「2.4.3.1.2　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。

2.4.3.2. CL/Web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2.10「CL/Web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。

■CL/Webコンテナ、システムコンテナ

```
docker container run --detach --volume <.lockinfoファイルのパス>:/etc/opt/wsnlesd/.lockinfo <イメージ名/イメージID>
```

表2.10 CL/Webのコンテナの設定

設定	CL/Webコンテナ	システムコンテナ
「2.4.3.2.1 バックグラウンドで起動」	◎	◎
「2.4.3.2.2 ライセンスの登録」	◎	◎
「2.4.3.2.3 CL/Webの環境設定」	○	×
「2.4.3.2.4 Javaのパラメータの設定」	○	×
「2.4.3.2.5 CL/Webのデータの永続化」	○	○
「2.4.3.2.6 永続化したデータの引き継ぎ」	○	○
「2.4.3.2.7 Docker内での通信の設定」	○	○
「2.4.3.2.8 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

2.4.3.2.1. 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。CL/Web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。

2.4.3.2.2. 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CL/Web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<クラシックモード用インストールガイド>の「2.3　コードワードを登録する」を参照してください。

システムコンテナにおいては、コンテナ作成後にコンテナ内の.lockinfoファイルに直接コードワードを記載してライセンス登録を行っても構いません。

2.4.3.2.3. CL/Webの環境設定

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、CL/Webの環境設定を行います。

環境変数名はclweb.confで設定できる各パラメータに対応しています。指定する環境変数は表2.11「CL/Webの環境設定を行う環境変数」を参照してください。

各パラメータに設定できる値については、R16.2の<クラシックモード用Web機能利用の手引き>の「3.1.1 CL/Web環境設定ファイル(clweb.conf)」を参照してください。

本設定は、clweb.confファイルの設定よりも優先されます。そのため、コンテナ内のclweb.confファイルを編集してコンテナを再起動しても、変更した設定は反映されません。設定を変更したい場合は、コンテナを削除した後再作成を行い、本設定で適切な環境変数を設定してください。

表2.11 CL/Webの環境設定を行う環境変数

環境変数名	環境変数値
JOBCENTER_CLWEB_PORT	clweb.confの $port パラメータで設定する値
JOBCENTER_CLWEB_BIND	clweb.confの $bind パラメータで設定する値
JOBCENTER_CLWEB_SSL_DISABLE	clweb.confの $ssl_disable パラメータで設定する値
JOBCENTER_CLWEB_TRACKER_AUTO_REFRESH	clweb.confの $tracker_auto_refresh パラメータで設定する値
JOBCENTER_CLWEB_JCCOMBASE	clweb.confの $jccombase パラメータで設定する値
JOBCENTER_CLWEB_ALLOW_SSL	clweb.confの $allow_ssl パラメータで設定する値
JOBCENTER_CLWEB_ACCESS_LOG_RETENTION_PERIOD	clweb.confの $access_log_retention_period パラメータで設定する値
JOBCENTER_CLWEB_MAIL_SERVER	clweb.confの $mail_server パラメータで設定する値
JOBCENTER_CLWEB_MAIL_PORT	clweb.confの $mail_port パラメータで設定する値
JOBCENTER_CLWEB_MAIL_DOMAIN	clweb.confの $mail_domain パラメータで設定する値
JOBCENTER_CLWEB_MAIL_AUTHENTICATION	clweb.confの $mail_authentication パラメータで設定する値
JOBCENTER_CLWEB_MAIL_USERNAME	clweb.confの $mail_username パラメータで設定する値
JOBCENTER_CLWEB_MAIL_PASSWORD	clweb.confの $mail_password パラメータで設定する値
JOBCENTER_CLWEB_MAIL_FROM	clweb.confの $mail_from パラメータで設定する値
JOBCENTER_CLWEB_MAIL_CHARSET_UTF8	clweb.confの $mail_charset_utf8 パラメータで設定する値
JOBCENTER_CLWEB_RELATIVE_URL_ROOT	clweb.confの $relative_url_root パラメータで設定する値
JOBCENTER_CLWEB_TAB_ORDER	clweb.confの $tab_order パラメータで設定する値

2.4.3.2.4. Javaのパラメータの設定

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、Javaのパラメータの設定を行います。詳細については、R16.2の<クラシックモード用Web機能利用の手引き>の「3.1.5 Javaのパラメータの設定」を参照してください。

指定する環境変数は表2.12「Javaのパラメータの設定を行う環境変数」を参照してください。

表2.12 Javaのパラメータ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_CLWEB_JAVA_OPTS_XMX	Javaの最大使用可能メモリ	1024m	1024m以上
JOBCENTER_CLWEB_JAVA_OPTS_XSS	各スレッドのスタックサイズ	8192k	8192k以上

本環境変数で設定する値には、コンテナが使用するメモリサイズの上限値よりも小さい値を指定してください。大きい値を指定した場合、コンテナが起動しない、起動したコンテナCL/Webに接続できない、などの状態になります。

2.4.3.2.5. CL/Web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

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。

■マウント先

マウント先のファイルのパスは、表2.1「MG/SV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。

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。

2.4.3.2.6. 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

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。

■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/SV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。

「2.4.3.2.5 CL/Webのデータの永続化」によって永続化したファイルを引き継いでコンテナを作成できます。

2.4.3.2.7. 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。

■「2.4.3.2.7.1 ネットワークの指定」

2.4.3.2.7.1. 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。

「2.4.1 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。

2.4.3.2.8. 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。

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

■「2.4.3.2.8.2 コンテナホスト名の設定」

■「2.4.3.2.8.3 名前解決方法の設定」

2.4.3.2.8.1. 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。

CL/Webが使用するポートは以下です。

ポート番号	説明
443	CL/WebサーバとWebブラウザ間の通信に使用するポート番号

「2.4.3.2.3　CL/Webの環境設定」でCL/Webサーバが使用する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。

2.4.3.2.8.2. 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>

Docker外部からコンテナへの名前解決を行うための、コンテナホスト名を指定します。

2.4.3.2.8.3. 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、「2.4.3.2.8.1　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、「2.4.3.2.8.2 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。

2.4.3.3. コンテナ作成後の作業

システムコンテナの場合、コンテナを作成/起動後にコンテナ内でセットアップやサービス起動の設定を行う必要があります。

MG/SVをインストールしたシステムコンテナの場合は、「2.4.3.3.1　MG/SVをインストールしたシステムコンテナ」を、

CL/Webをインストールしたシステムコンテナの場合は、「2.4.3.3.2　CL/Webをインストールしたシステムコンテナ」を参照して作業を行ってください。

2.4.3.3.1. MG/SVをインストールしたシステムコンテナ

作成したコンテナにbashで接続し、コンテナ内で以下の作業を行ってください。

コンテナにbashで接続する方法については「2.5.1 コンテナ内で操作」を参照してください。

■JobCenter管理者ユーザのパスワードの設定

passwdコマンドを実行し、JobCenter管理者ユーザ（nsumsmgr）のパスワードを設定します。

■JobCenterのセットアップ

/usr/local/netshep/nssetupコマンドを実行し、JobCenterのセットアップを行います。

「2.4.3.1.10　永続化したデータの引き継ぎ」を利用して、/usr/spool/nqsディレクトリを引き継いだ場合、JobCenterのセットアップは不要です。

「2.4.3.1.9 JobCenterのデータの永続化」を利用して/usr/spool/nqsディレクトリを永続化した場合、セットアップ時の以下のメッセージには y を選択してください。

```
Do you use the old spool directory? [y/n](default: n)
```

■サービスの自動起動の有効化

JobCenterサービスの自動起動を行いたい場合、以下のコマンドで自動起動の有効化を行います。

```
systemctl enable nqs
```

```
systemctl enable nqs.pre
```

■JobCenterサービスの起動

以下のコマンドを実行し、JobCenterサービスを起動します。

```
systemctl start nqs
```

2.4.3.3.2. CL/Webをインストールしたシステムコンテナ

作成したコンテナにbashで接続し、コンテナ内で以下の作業を行ってください。

コンテナにbashで接続する方法については「2.5.1 コンテナ内で操作」を参照してください。

■サービスの自動起動の有効化

CL/Webサービスの自動起動を行いたい場合、以下のコマンドで自動起動の有効化を行います。

```
systemctl enable jcclweb
```

■CL/Webサービスの起動

以下のコマンドを実行し、CL/Webサービスを起動します。

```
systemctl start jcclweb
```

2.4.3.4. 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。

2.4.3.4.1. 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。

2.4.3.4.2. パスワードの暗号化

「2.4.3.1.5 JobCenter管理者ユーザのパスワードの設定」および「2.4.3.1.8 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。

2.4.3.4.3. 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>
```

本節での手順でコンテナMGとコンテナSVの通信を行う場合、コンテナMGとコンテナSVの両方のdaemon.confにipcheck=OFFを記載する必要があります。

2.4.4. 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、<クラシック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<クラシックモード用環境構築ガイド>の23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。

■「2.4.3.1.10 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

- 「2.4.3.1.6 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。
- 「2.4.3.1.6　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。
- 「2.4.3.1.2 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/SVやCL/Web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/SVやCL/Web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/SV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、<クラシックモード用環境構築ガイド>の「5.2 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/SVの停止、起動

コンテナを起動させたまま、コンテナ内のMG/SV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/SV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、<クラシック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。

■MG/SV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<クラシック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。

■システムコンテナの場合

systemctlコマンドを利用します。

停止する場合はsystemctl stop nqsを、起動する場合はsystemctl start nqsを実行してください。

システムコンテナでのMG/SVの停止、起動にnqsstop、nqsstartコマンドを使用しないでください。

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
- システムコンテナで停止処理に時間のかかるアプリケーションを同居させている場合

■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）、システムコンテナの場合はSIGRTMIN+3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/SVおよびCL/Web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/SVコンテナおよび、MG/SVがインストールされたシステムコンテナについては「2.6.1.1 MG/SVコンテナにおける障害」を、

CL/Webコンテナおよび、CL/Webがインストールされたシステムコンテナについては「2.6.1.2 CL/Web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/SVコンテナにおける障害

表2.13「MG/SVコンテナにおける障害」を参照してください。

表2.13 MG/SV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/SV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/SV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/SVのパッケージ名> (/<MG/SV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/SV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/SV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。 「2.4.3.1.2 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。 「2.4.3.1.3 ライセンスの登録」を参照して、コンテナ内のMG/SV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。 「2.4.3.1.5 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。 「2.4.3.1.6 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。 「2.4.3.1.6 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。 「2.4.3.1.6 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_NQS_PORT ■JOBCENTER_JCCOMBASE_PORT ■JOBCENTER_JCCOMBASE_OVER_SSL_PORT ■JOBCENTER_JCEVENT_PORT ■JOBCENTER_JCWEBSERVER_PORT 「2.4.3.1.7 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_NQS_PORT ■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 「2.4.3.1.7 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。 「2.4.3.1.8.1 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。 「2.4.3.1.10 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/SV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/SV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、<クラシック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。

■全てのプロセスが起動していない場合

システムコンテナにおいては、コンテナ作成後にコンテナ内でJobCenterのセットアップおよび起動を行う必要があります。

「2.4.3.3 コンテナ作成後の作業」を参照して、作業を行ってください。

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, 607/tcp, 611/tcp, 23116/tcp, 23180/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、「2.4.3.1.12.1 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。

「2.4.3.1.7 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、「2.4.3.4.3 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「2.4.1 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/SV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/SV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/SV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、「2.4.3.4.3　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「2.4.1　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/SV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、「2.4.3.1.12.2　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. daemon.confにipcheck=OFFが記載されているか確認する

   コンテナMGがSVとマシン連携を行う場合、SV側のマシンのdaemon.confにipcheck=OFFを記載して、JobCenterを再起動する必要があります。

   また、「2.4.3.4.3　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」を行っている場合は、MG側のマシンのdaemon.confにもipcheck=OFFを記載する必要があります。

2. 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, 607/tcp, 611/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、「2.4.3.1.12.1 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。

「2.4.3.1.7 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. リモート投入したジョブがSUBMIT状態のまま進行しない

SV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。

「2.4.3.1.11.3　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<クラシック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。

2.6.1.2. CL/Webコンテナにおける障害

表2.14「CL/Webコンテナにおける障害」を参照してください。

表2.14 CL/Web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	「2.6.1.2.1 イメージの作成に失敗する」
コンテナ環境の構築時	「2.6.1.2.2 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	「2.6.1.2.3 ブラウザからCL/Webへ接続できない」
コンテナ環境の運用時	「2.6.1.2.4 MG/SVへのログインに失敗する」

2.6.1.2.1. 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。
+ '[' -e NECJCpkg-clweb.zip ']' + curl -OL NECJCpkg-clweb.zip curl: Remote file name has no length! curl: try 'curl --help' or 'curl --manual' for more information	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。

2.6.1.2.2. コンテナが起動しない/起動後停止する

以下の点を確認して、対処を行ってください。

1. 有効なライセンスのコードワードが記載された.lockinfoファイルが存在しているか確認する

   コンテナを起動するためには、コンテナ内でライセンス登録を行う必要があります。

   「2.4.3.2.2 ライセンスの登録」を参照して、コンテナ内のCL/Webのライセンス登録を行ってください。

2. イメージの作成時に、LicenseManagerのパッケージが正しく配置されていたか確認する

   docker image build実行時に以下のメッセージが出力されている場合、LicenseManagerのインストールが行われていません。

   ```
   No package NECWSLM-1.11-1.i386.rpm available.
   ```

   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、LicenseManager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LMPKGに正しいURLが指定されていること

2.6.1.2.3. ブラウザからCL/Webへ接続できない

以下の点を確認して、対処を行ってください。

1. CL/Webの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>
```

■clweb_servが起動している場合

コンテナを起動してからCL/Webサーバに接続できるようになるまで30秒ほどかかります。

時間をおいて再度接続を試みてください。

■clweb_servが起動していない場合

システムコンテナにおいては、コンテナ作成後にコンテナ内でCL/Webの起動を行う必要があります。

「2.4.3.3 コンテナ作成後の作業」を参照して、作業を行ってください。

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>
```

443/tcpが、Dockerホスト上のIPアドレスのポート番号とマッピングされていない場合、「2.4.3.2.8.1　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。

clweb.confの設定で、CL/Webが使用するポート番号を変更した場合、変更したポート番号を開放してください。

3. 接続元からコンテナCL/Webへの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、CL/Webコンテナが名前解決されるよう設定してください。

2.6.1.2.4. MG/SV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。

■コンテナ環境のMG/SVにログインする場合（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、「2.4.3.4.3　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「2.4.1　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/SVにログインする場合（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、「2.4.3.2.8.3 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。

■MG/SVが使用するjccombaseのポート番号を変更している場合

MG/SVのマシンの/etc/servicesファイルを確認し、jccombaseが使用するポート番号を確認します。

デフォルトで使用する611から変更されている場合、clweb.confの$jccombaseパラメータに変更したポート番号を設定する必要があります。

clweb.confの設定を変更する場合、「2.4.3.2.3 CL/Webの環境設定」を利用するか、コンテナ内のclweb.confファイルを編集してコンテナを再起動してください。

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。

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/SV

com.nec.name=JobCenter MG/SV

■CL/Web

com.nec.name=JobCenter CL/Web

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。

また、同一のラベル名を複数設定することはできないため、MG/SVおよびCL/Webが両方インストールされているシステムコンテナにおいては、ラベルはどちらかしか設定できません。設定されてない方の情報採取は、手動での情報採取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/SVの情報採取にはjc_getinfoコマンドを、CL/Webの情報採取にはclweb_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/SV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>
```

■CL/Web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、clweb_getinfoを実行する

```
docker container exec -u root <コンテナ名/コンテナID> /usr/local/jcclweb/script/clweb_getinfo/clweb_getinfo.sh
```

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

```
Output to "/clwebinfo_<YYYYMMDDhhmmss>.zip"
```

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およびclweb_getinfoについては、<クラシックモード用コマンドリファレンス>の「7.1 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」、「7.2 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2.15「コピーするファイル一覧」を参照してください。

表2.15 コピーするファイル一覧

コピーするファイル	MG/SV	CL/Web
/etc/hosts	○	○
/etc/resolv.conf	○	○
/etc/nsswitch.conf	○	○
/etc/services	○	×
/usr/lib/nqs/rc	○	×
/usr/spool/nqs/log	○	×
/usr/local/jcclweb/config	×	○
/usr/local/jcclweb/run	×	○
/usr/local/jcclweb/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/SV、CL/Webのコンテナの情報を採取します。なお、MG/SVおよびCL/Webのコンテナは、以下のラベルによって判断します。

■MG/SV

com.nec.name=JobCenter MG/SV

■CL/Web

com.nec.name=JobCenter CL/Web

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。

■同一のラベル名を重複して指定することができません。そのため、MG/SVおよびCL/Webが両方インストールされているシステムコンテナにおいては、どちらかのラベルしか設定できません。その場合、ラベルが設定されてない方の製品の情報については、「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/SVコンテナからファイルのコピーを開始します。

メッセージ	説明
Start to copy clweb files from container(<コンテナID>).	CL/Webの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 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

3.1. 概要

3.1.1. AWS EKS上でのコンテナ運用概要

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/SVおよびCL/Webコンテナをデプロイと管理できます。

■イメージ作成

MG/SVコンテナ用のイメージの作成は「3.2.1 MG/SVのイメージの作成」を参照してください。

CL/Webコンテナ用のイメージの作成は「3.2.2 CL/Webのイメージの作成」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と編集

  MG/SVのマニフェストファイルの作成は「3.3.1　MG/SVのマニフェストファイルの作成」を参照してください。マニフェストファイルの編集は「3.4.1　MG/SVのマニフェストファイルの編集」を参照してください。

  CL/Webのマニフェストファイルの作成は「3.3.2　CL/Webのマニフェストファイルの作成」を参照してください。マニフェストファイルの編集は「3.4.2　CL/Webのマニフェストファイルの編集」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「3.5 リソースの作成」を参照してください。

■AWS EKS環境の運用

- EKS上のMG/SVへの運用

  EKS上のMG/SVへの通常接続は「3.6.1 EKS上のMG/SVへの接続（通常の接続）」を参照してください。

  EKS上のMG/SVへの保護された接続は「3.6.2　EKS上のMG/SVへの接続（保護された接続）」を参照してください。

  EKS上のMG/SVへのWebAPIの発行は「3.6.3 EKS上のMG/SVへのWebAPIの発行」を参照してください。

- EKS上のCL/Webへの接続

  詳細は「3.6.4 EKS上のCL/Webへの接続」を参照してください。

- デバッグ方法

  詳細は「3.6.5 デバッグのためMG/SVおよび、CL/Webのコンテナへの接続」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「3.7.1 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「3.7.2 エラーメッセージ一覧」を参照してください。

3.2. 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。

3.2.1. MG/SVのイメージの作成

DockerfileからMG/SVのコンテナイメージを作成する手順を説明します。

1. Dockerfileおよびentrypoint.shを用意します。

JobCenterのパッケージの以下のディレクトリに同梱されています。

■CONTAINER/DOCKER/DOCKERFILE/MGSV

2. Dockerfileに対して、以下の修正を行います。

■FROMで指定するベースイメージを、RedHat 8のUBIイメージに修正します。

latestのイメージを使用する場合の指定例

```
FROM registry.access.redhat.com/ubi8/ubi:latest
```

■ENTRYPOINTを以下に修正します。

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

■tiniアプリケーション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。

■yumでインストールするアプリケーションにprocsを追加します。

```
RUN yum -y install procps
```

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、「2.3.1.2　イメージのビルド」を参照してください。

3.2.2. CL/Webのイメージの作成

DockerfileからCL/Webのコンテナイメージを作成する手順を説明します。

1. Dockerfileおよびentrypoint.shを用意します。

JobCenterのパッケージの以下のディレクトリに同梱されています。

■CONTAINER/DOCKER/DOCKERFILE/CLWEB

2. Dockerfileに対して、以下の修正を行います。

■FROMで指定するベースイメージを、RedHat 8のUBIイメージに修正します。

latestのイメージを使用する場合の指定例

```
FROM registry.access.redhat.com/ubi8/ubi:latest
```

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。

3.3. マニフェストファイルの作成

EKS上にKubernetesのリソースを作成するためのマニフェストファイルを作成します。

3.3.1. MG/SVのマニフェストファイルの作成

MG/SVコンテナをEKS上で作成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: 607
          protocol: TCP
        - containerPort: 611
          protocol: TCP
        - containerPort: 10012
          protocol: TCP
        - containerPort: 23116
          protocol: TCP
        - containerPort: 23180
          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/SV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。「3.2.1 MG/SV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PersitentVolumeのオブジェクトの名前です。PersistentVolumeClaim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: nqs
    port: 607
    protocol: TCP
  - 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
  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からコンテナMG/SVに接続を行う場合、または、JobCenter　MG/SVのWebAPIをコンテナMG/SVで実行する場合には、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
  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/SVのWebAPIをコンテナMG/SVで実行しない場合には、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。

■PersistentVolumeClaim 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/SV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。

3.3.2. CL/Webのマニフェストファイルの作成

CL/WebコンテナをEKS上で作成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: RollingUpdate
  template:
    metadata:
      labels:
        app: jobcenter
        name: <name>
      name: jobcenter-clweb
    spec:
      containers:
      - name: jobcenter-clweb
        imagePullPolicy: IfNotPresent
        image: <image>
        ports:
        - containerPort: 443
          protocol: TCP
        volumeMounts:
          - name: lockinfo
            mountPath: /etc/opt/wsnlesd/.lockinfo
            subPath: lockinfo
          - name: log-pvc
            mountPath: /usr/local/jcclweb/log
            subPath: log-pvc
      volumes:
        - name: log-pvc
          persistentVolumeClaim:
            claimName: <pvc name>-log
        - 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.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。

- <image>

  「3.2.2 CL/Web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PersitentVolumeのオブジェクトの名前です。PersistentVolumeClaim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: clweb
    port: 443
    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

- <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。

■PersistentVolumeClaim のマニフェストファイル

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <name>-log
  labels:
    app: jobcenter
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
```

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CL/Web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。

3.4. 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。

3.4.1. MG/SV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MG/SV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	「3.4.1.1 管理者ユーザのパスワードの設定」
マシンIDの設定	「3.4.1.2 マシンIDの設定」
daemon.confの設定	「3.4.1.3 daemon.confの設定」
一般ユーザの設定	「3.4.1.4 一般ユーザの作成」
ポート番号の変更	「3.4.1.5 ポート番号の変更」
セットアップ言語の設定	「3.4.1.6 セットアップ言語の設定」
EKS外部のマシンとの連携	「3.4.1.7 EKS外部のマシンとの連携」
jcwebserver.confの設定	「3.4.1.8 jcwebserver.confの設定」
証明書・秘密鍵ファイルの設定	「3.4.1.9 証明書・秘密鍵ファイルの設定」

3.4.1.1. 管理者ユーザのパスワードの設定

MG/SV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. 「2.4.3.4.2 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>
```

3.4.1.2. マシンIDの設定

コンテナMG/SV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。

3.4.1.3. daemon.confの設定

コンテナ内にdaemon.confを設定することで、コンテナMG/SVの環境設定を行えます。

daemon.confについては、<クラシックモード用環境構築ガイド>の5章　「JobCenter起動時の設定を変更する」 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、daemon.conf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: daemon-conf
data:
  daemon-conf: |
    ipcheck=OFF
```

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: daemon-conf
          configMap:
            name: daemon-conf
            defaultMode: 420
```

■spec.template.spec.containers.volumeMounts

```
        volumeMounts:
          - name: daemon-conf
            mountPath: /usr/lib/nqs/rc/daemon.conf
            subPath: daemon-conf
```

EKS上のMG/SV間でマシン連携を行う場合、daemon.confにipcheck=OFFのパラメータを設定してください。

3.4.1.4. 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. 「2.4.3.1.8.1 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>
```

3.4.1.5. ポート番号の変更

コンテナMG/SVのプロセスが使用するport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_NQS_PORT
          value: <設定したいnqsのポート番号>
        - 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>
```

3.4.1.6. セットアップ言語の設定

コンテナMG/SV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

3.4.1.7. EKS外部のマシンとの連携

EKS外部のサーバに存在するMG/SVとマシン連携を行うために、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/SV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/SVコンテナが使用するが利用するポート(デフォルトは607/611/10012/23116/23180)を許可する設定にします。

4. コンテナ内のMG/SVおよび、EKS外のサーバ上のMG/SV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/SVコンテナしか作成できません。複数のMG/SV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。

3.4.1.8. jcwebserver.confの設定

コンテナ内にjcwebserver.confを設定することで、コンテナMG/SVのjcwebserverの環境設定を行えます。

jcwebserver.confについては、<クラシックモード用環境構築ガイド>の「5.7　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. 以下をベースにして、jcwebserver.conf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: 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: jcwebserver-conf
          configMap:
            name: jcwebserver-conf
            defaultMode: 420
```

■spec.template.spec.containers.volumeMounts

```
        volumeMounts:
```

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

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。

3.4.1.9. 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、<クラシック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、<クラシックモード用環境構築ガイド>の「5.2　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、<クラシックモード用環境構築ガイド>の「5.7　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。

3.4.2. CL/Web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MG/SVの以下の機能を利用できます。

機能	章
clweb.confの設定	「3.4.2.1 clweb.confの設定」
SSL署名証明書の設定	「3.4.2.2 SSL署名証明書の設定」
マイページ設定の永続化	「3.4.2.3 マイページ設定の永続化」
メールテンプレートの永続化	「3.4.2.4 メールテンプレートの永続化」

3.4.2.1. clweb.confの設定

コンテナ内にclweb.confを設定することで、コンテナCL/Webの環境設定を行えます。

clweb.confについては、R16.2の<クラシックモード用Web機能利用の手引き>の「3.1.1 CL/Web環境設定ファイル(clweb.conf)」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のパラメータ追加とConfigMapの作成を行ってください。

1. 以下をベースにして、clweb.conf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: clweb-conf
data:
  clweb-conf: |
    $port = 443
    $bind = "0.0.0.0"
    $ssl_disable = false
    $tracker_auto_refresh = false
    $jccombase = 611
    $allow_ssl = false
    $access_log_retention_period = 365
    $mail_server = "smtpserver"
    $mail_port = 25
    $mail_domain = ""
    $mail_authentication = "plain"
    $mail_username = "username"
    $mail_password = "password"
    $mail_from = ""
    $mail_charset_utf8 = true
    $relative_url_root = "/clweb"
    $tab_order = "6 1 2 3 7 4 5"
```

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: conf
          configMap:
            name: clweb-conf
            defaultMode: 420
```

■spec.template.spec.containers.voluumeMounts

```
        volumeMounts:
          - name: conf
            mountPath: /usr/local/jcclweb/config/clweb.conf
            subPath: clweb-conf
```

3.4.2.2. SSL署名証明書の設定

コンテナ内に証明書ファイルを設定することで、コンテナCL/Webの証明書の設定を行えます。

SSL署名証明書については、R16.2の<クラシックモード用Web機能利用の手引き>の「3.1.3　SSL署名証明書の設定」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のパラメータ追加とConfigMapの作成を行ってください。

1. CL/Webサーバに設定する証明書ファイルと秘密鍵ファイルを用意します。

2. 証明書ファイルと秘密鍵ファイルからConfigMapを作成します。

```
$ kubectl create configmap clweb-sslkey --from-file <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.volumes

```
      volumes:
        - name: sslcert
          configMap:
            name: clweb-sslcert
            defaultMode: 420
        - name: sslkey
          configMap:
            name: clweb-sslkey
            defaultMode: 420
```

■spec.template.spec.containers.voluumeMounts

```
        volumeMounts:
          - name: sslcert
            mountPath: /usr/local/jcclweb/config/ssl_cert
            subPath: <証明書ファイル名>
          - name: sslkey
            mountPath: /usr/local/jcclweb/config/ssl_key
            subPath: <秘密鍵ファイル名>
```

3.4.2.3. マイページ設定の永続化

コンテナCL/Webのマイページの設定を永続化することで、コンテナが更新などによって再作成された場合に、マイページの設定を引き継ぐことができます。

マイページの設定については、R16.2の<クラシックモード用Web機能利用の手引き>の「5.1.6　マイページ機能」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に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.volumes

```
      volumes:
        - name: mypage-pvc
          persistentVolumeClaim:
            claimName: jobcenter-clweb-mypage
```

■spec.template.spec.containers.voluumeMounts

```
        volumeMounts:
          - name: mypage-pvc
            mountPath: /usr/local/jcclweb/config/mypage
            subPath: mypage-pvc
```

3.4.2.4. メールテンプレートの永続化

コンテナCL/Webのメールテンプレートを永続化することで、コンテナが更新などによって再作成された場合に、作成したメールテンプレートを引き継ぐことができます。

メールテンプレートについては、R16.2の<クラシックモード用Web機能利用の手引き>の「7.1.4 メールテンプレート設定」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に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.volumes

```
      volumes:
        - name: mail-pvc
          persistentVolumeClaim:
            claimName: jobcenter-clweb-mail
```

■spec.template.spec.containers.voluumeMounts

```
        volumeMounts:
          - name: mail-pvc
            mountPath: /usr/local/jcclweb/config/mail
            subPath: mail-pvc
```

3.5. リソースの作成

「3.3 マニフェストファイルの作成」および「3.4 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
```

3.6. 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。

3.6.1. EKS上のMG/SVへの接続（通常の接続）

CL/Win（保護された接続を使用しない）およびCL/Webから、EKS上のコンテナMG/SV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>        607/TCP,611/TCP,10012/
TCP,23116/TCP,23180/TCP   129m
jobcenter-manager-nodeport    NodePort    10.100.28.37     <none>        611:32755/
TCP,23116:31014/TCP,23180:31609/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
```

「3.4.1.7 EKS外部のマシンとの連携」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、「3.4.1.7　EKS外部のマシンとの連携」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。

3.6.2. EKS上のMG/SV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/SV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>        607/TCP,611/TCP,10012/
TCP,23116/TCP,23180/TCP   129m
jobcenter-manager-nodeport    NodePort    10.100.28.37     <none>        611:32755/
TCP,23116:31014/TCP,23180:31609/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
```

「3.4.1.7 EKS外部のマシンとの連携」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、「3.4.1.7　EKS外部のマシンとの連携」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。

3.6.3. EKS上のMG/SVへのWebAPIの発行

ブラウザから、EKS上のコンテナMG/SVへJobCenter MG/SVのWebAPIを発行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>        607/TCP,611/TCP,10012/
TCP,23116/TCP,23180/TCP   129m
jobcenter-manager-nodeport    NodePort    10.100.28.37     <none>        611:32755/
TCP,23116:31014/TCP,23180:31609/TCP   129m
```

2. JobCenter MG/SVのWebAPIを発行する時のURLの<MG/SVサーバのホスト名またはIPアドレス>には、<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
```

「3.4.1.7 EKS外部のマシンとの連携」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、「3.4.1.7　EKS外部のマシンとの連携」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/SVのWebAPIを発行する時のURLの<MG/SV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。

3.6.4. EKS上のCL/Webへの接続

ブラウザからEKS上のコンテナCL/Web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、443port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-clweb   NodePort    10.100.119.119   <none>        443:30824/TCP    10s
```

2. R16.2の<クラシックモード用Web機能利用の手引き>の「5.2　ブラウザでCL/Webサーバにアクセスする」を参照して、ブラウザからコンテナCL/Webに接続します。

   URLの<CL/Webサーバのホスト名またはIPアドレス>には、<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。

3.6.5. デバッグのためMG/SVおよび、CL/Webのコンテナへの接続

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
```

3.7. AWS EKS環境のトラブルシューティング

3.7.1. 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>
```

3.7.2. 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/SVコンテナにおける障害」を参照してください。

4. OpenShift環境での運用

本章ではRed Hat OpenShift Container Platform上でJobCenterコンテナを運用する方法について説明します。

4.1. 概要

4.1.1. OpenShiftの概要

Red Hat OpenShift Container Platform(以下、OpenShift)とは、エンタープライズ向けのKubernetesコンテナプラットフォームです。OpenShiftを用いることで、コンテナのデプロイや障害時の復旧など、コンテナアプリケーションを運用する上で必要な操作を自動化することができます。

OpenShift上でコンテナアプリケーションを運用するためには、yaml形式のマニフェストファイルを用いて、リソースを作成する必要があります。リソースにはコンテナやロードバランサなどのさまざまな種類があり、コンテナアプリケーションを運用するために必要なリソースの作成やパラメータの設定を適切に行う必要があります。

Kubernetes　Operator(以下、Operator)を利用することで、JobCenterのコンテナをOpenShift上で作成して運用するために必要なマニフェストファイルの設定やリソースの作成などを自動化できます。

4.1.2. Operatorの概要

OperatorはKubernetesのリソースの作成、管理を行うアプリケーションです。Operatorは CustomResource(以下、CR)と呼ばれる独自のKubernetesリソースによって管理されます。

JobCenterでは、JobCenter　MG/SVのOpenShift上での運用を自動化するためのJobCenterService　OperatorとCRを提供しています。マニフェストファイルにパラメータを設定してCRをデプロイすることで、JobCenter MG/SVのコンテナの作成やデータのバックアップ、複数のコンテナ間におけるマシン連携などをOperatorによって自動化できます。

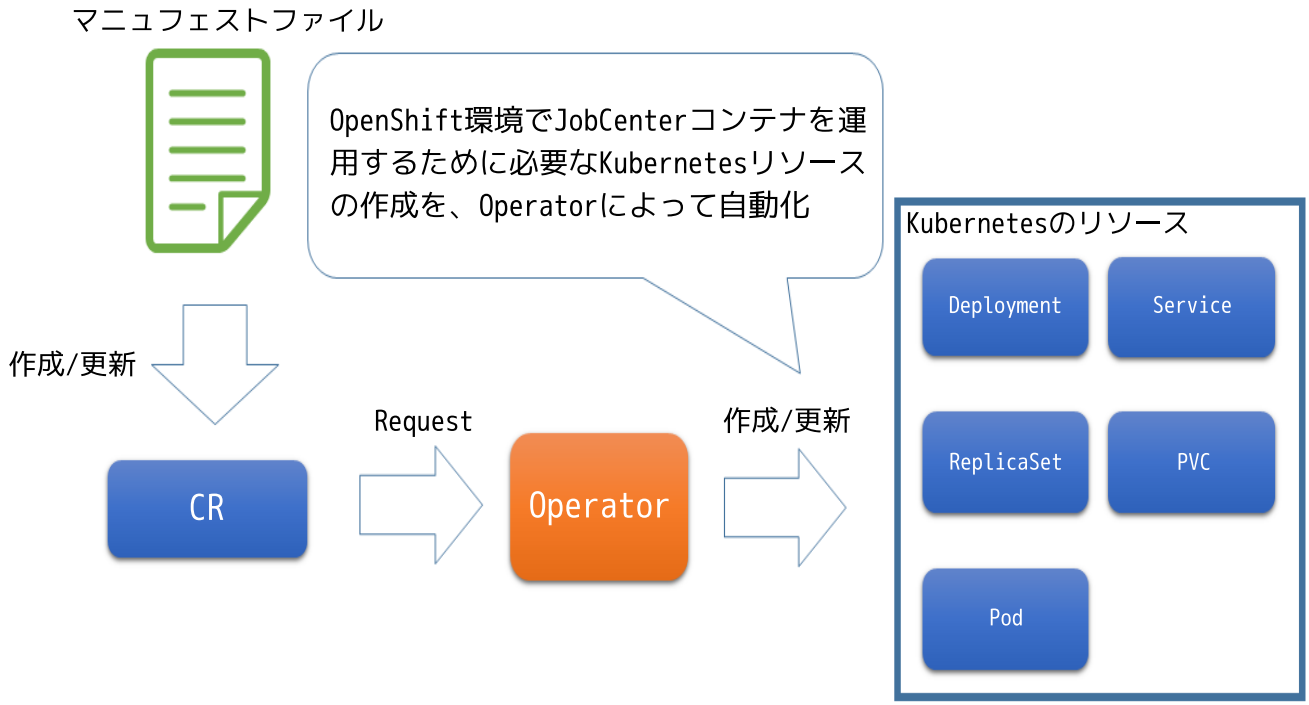

図4.1 Operatorを用いたOpenShift上でのコンテナ運用

4.2. Operatorを用いたJobCenter環境の設計

OpenShift環境におけるOperatorを用いたJobCenter環境の構成を設計します。

4.2.1. JobCenter環境の構成

Operatorを用いたJobCenter環境の構成について説明します。

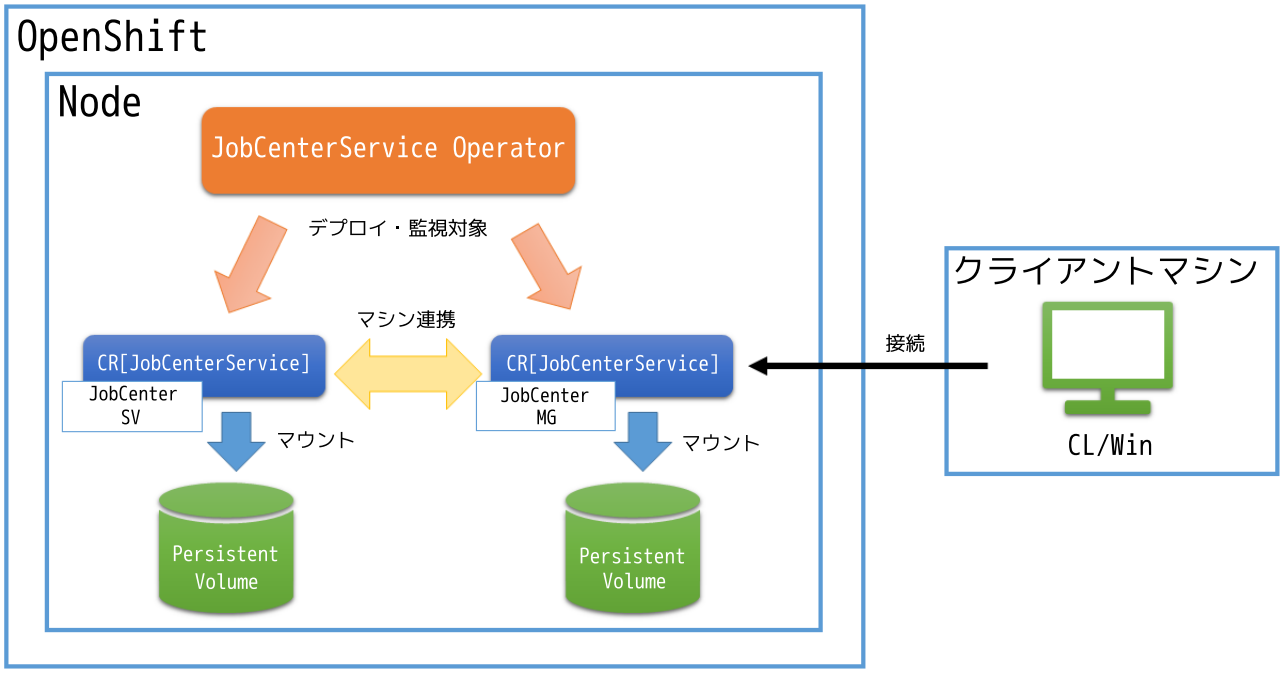

図4.2 JobCenter Operatorを用いた構成図

■JobCenterService Operatorは、CRを作成することでMG/SVコンテナを1つ作成します。CRを複数作成することで、複数のMG/SVコンテナをOpenShift環境にデプロイできます。Operatorが作成したMG/SVコンテナ同士は、Operatorによって自動でマシン連携の設定を行えます。

■OpenShift外部に存在するマシンにインストールしているCL/Winから、MG/SVコンテナに対して接続を行えます。（参照：「4.2.2 ネットワーク環境構築」）

■コンテナ内のデータは、OpenShiftのPersistentVolumeで永続化できます。(参照：「4.2.3 コンテナのデータ管理」)

4.2.2. ネットワーク環境構築

OpenShift外部のマシンから、OpenShift環境のコンテナMG/SVに対して接続する方法について説明します。

4.2.2.1. CL/Winから接続

OpenShift外部に存在するマシン上にインストールしたCL/Winから、OpenShift環境のコンテナMG/SVに対して接続する手順は以下のとおりです。

1. CR JobCenterServiceの、「4.4.2.1.3 OpenShift外部との通信設定」にて、typeパラメータにclientを設定してCRを作成します。
2. 以下のocコマンドを実行して、EXPOSEフィールドに表示されるポート番号を確認します。

```
$ oc get jobcenterservice
NAME      EXPOSE              BACKUPSTATE   LASTBACKUPRESULT   AGE
manager   <MASTER_IP>:30611   UnScheduled                      28s
```

3. CL/Winからサーバへ接続する際、サーバ名に<OpenShiftのマスターサーバ名>:<上記手順で確認したポート番号>を入力して、接続します。

4.2.3. コンテナのデータ管理

コンテナ内のデータは、コンテナが削除や再作成された際に破棄されます。コンテナ内のデータを引き継ぎたい場合、MG/SVコンテナ内のディレクトリをOpenShiftのボリュームにマウントすることで永続化します。

JobCenterService Operatorでは、以下のディレクトリを永続化できます。

ディレクトリ	説明
/usr/spool/nqs	定義ファイルや実行中のジョブの実行結果などを保存するスプールディレクトリです。詳細は<クラシックモード用リリースメモ>の「3.2.3.1 スプールディレクトリ」を参照してください。
/usr/spool/backup	「4.4.2.1.6 バックアップの設定」により、バックアップを行った定義データや構成情報を保存するディレクトリです。

動作検証や設定確認などで一時的にコンテナを作成したい場合ではなく、本番環境でコンテナMG/SVを運用する場合、必ずスプールディレクトリの永続化を行ってください。

データを永続化する手順は以下のとおりです。

1. 永続化用のストレージを用意します。

JobCenterの永続化ストレージとして使用できるものは以下のとおりです。

■AccessModeがReadWriteOnce な ブロックストレージ バックエンド

■AccessModeがReadWriteOnce な NFS バックエンド

AccessModeについては、以下のKubernetesドキュメントを参照してください。

`https://kubernetes.io/docs/concepts/storage/persistent-volumes/#access-modes`

2. OpenShiftの「永続ボリューム」の設定に従って、PersistentVolumeを作成します。

3. CR JobCenterServiceを作成する際、永続化に関するパラメータを設定します。詳細は「4.4.2.1.5 コンテナ内のデータの永続化の設定」を参照してください。

NFSを用いて永続化する場合、以下の設定を行う必要があります。

■NFSサーバの/etc/exportsで、no_root_squashを設定する

■PersisteneVolume作成時、以下のパラメータを設定する

```
mountOptions:
- rw,context="system_u:object_r:container_file_t:s0"
```

■NFSv4でマウントされるように設定する

4.3. Operatorの展開

本章では、OpenShift環境にOperatorを展開する方法について説明します。

OperatorはOpenShiftのプロジェクト単位でデプロイを行います。そのため、複数のプロジェクトでOperatorを利用したい場合、プロジェクトごとに本章に記載されている手順を実施してください。

本章で記載されている手順を行う前に、以下のことが実施されているか確認してください。

■JobCenterパッケージの以下のディレクトリに同梱されている、OpenShift環境構築用ファイル類の入手

- CONTAINER/OPENSHIFT

■コンテナイメージを一時的に格納するDockerレジストリの準備

■cluster-admin相当の権限を有するOpenShiftのユーザ（OpenShift管理者）の作成

■Operatorを展開するプロジェクトの作成

4.3.1. イメージの準備

OpenShift環境でコンテナイメージを利用する場合、OpenShiftのコンテナレジストリにイメージをpushし、プロジェクトごとにImageSteramを作成します。

本章では、Operatorが利用するコンテナイメージをOpenShiftのレジストリにpushし、ImageSteramを作成する手順を説明します。

OpenShift外部のマシンからOpenShiftのレジストリに対してイメージのpushを行うため、OpenShiftの「レジストリの公開」の手順に従って、事前にレジストリの公開を行ってください。

4.3.1.1. JobCenterService Operatorのコンテナイメージの準備

JobCenterService Operatorのコンテナイメージから ImageStreamを作成する手順は以下のとおりです。

本章ではdockerコマンドを使用した手順を記載しています。podmanコマンドを使用する場合は、dockerコマンドの「docker」を「podman」に読み替えてください。

1. JobCenterのパッケージに同梱されているOperatorのコンテナイメージを解凍します。

```
$ gunzip jobcenterservice_operator.tar.gz
```

2. コンテナイメージをDockerレジストリに展開します。

```
$ docker load -i jobcenterservice_operator.tar
```

3. 展開したOperatorのコンテナイメージに、タグで別名を設定します。

```
$ docker tag jobcenterservice-operator:1.0.0 <イメージ名>
```

イメージ名は、以下の形式で指定します。

```
<OpenShiftのレジストリ>/<プロジェクト名>/<ImageStream名>
```

■OpenShiftのレジストリ

OpenShiftのレジストリのアドレスを指定します。

■プロジェクト名

ImageStreamを作成するプロジェクト名を指定します。

■ImageStream名

作成するImageStreamのオブジェクト名を指定します。<オブジェクト名>:<タグ名>の形式で指定してください。

4. OpenShift管理者ユーザでOpenShiftにログインします。

```
$ oc login -u <OpenShift管理者ユーザ名> -p <パスワード> --server=<OpenShiftのサーバのアドレス>
```

5. OpenShiftのイメージレジストリにログインします。

```
$ docker login -u <OpenShift管理者のユーザ名> -p $(oc whoami -t)
```

6. OpenShiftのイメージレジストリにイメージをpushします

```
$ docker push <手順3で設定したイメージ名>
```

7. プロジェクトにImageStreamが作成されていることを確認します。

```
$ oc get imagestream -n <プロジェクト名>
```

作成したImageStreamは、Operatorのマニフェストファイルのimageパラメータに指定します。詳細は「4.3.2 JobCenerService Operatorのデプロイ」を参照してください。

4.3.1.2. MG/SVのコンテナイメージの準備

JobCenter MG/SVのコンテナイメージからImageStreamを作成する手順は以下のとおりです。

本章ではdockerコマンドを使用した手順を記載しています。podmanコマンドを使用する場合は、dockerコマンドの「docker」を「podman」に読み替えてください。

1. Dockerfileから、MG/SVのコンテナイメージを作成します。手順については、「2.3.1.2　イメージのビルド」を参照してください。

DockerfileおよびentrypointはJobCenterのパッケージに同梱されている以下のディレクトリのものを使用してください。

■CONTAINER/OPENSHIFT/DOCKERFILE/MGSV

2. 作成したMG/SVのコンテナイメージに、タグで別名を設定します。

```
$ docker tag <MG/SVのコンテナイメージ名:タグ名> <イメージ名>
```

イメージ名は、以下の形式で指定します

```
<OpenShiftのレジストリ>/<プロジェクト名>/<ImageStream名>
```

■OpenShiftのレジストリ

OpenShiftのレジストリのアドレスを指定します。

■プロジェクト名

ImageStreamを作成するプロジェクト名を指定します。

■ImageStream名

作成するImageStreamのオブジェクト名を指定します。<オブジェクト名>:<タグ名>の形式で指定してください。

3. OpenShift管理者ユーザでOpenShiftにログインします。

```
$ oc login -u <OpenShift管理者ユーザ名> -p <パスワード> --server=<OpenShiftのサーバのアドレス>
```

4. OpenShiftのイメージレジストリにログインします。

```
$ docker login -u <OpenShift管理者のユーザ名> -p $(oc whoami -t)
```

5. OpenShiftのイメージレジストリにイメージをpushします。

```
$ docker push <手順2で設定したイメージ名>
```

6. プロジェクトにImageStreamが作成されていることを確認します。

```
$ oc get imagestream -n <プロジェクト名>
```

作成したImageStreamは、CR JobCenterServiceのマニフェストファイルのimageパラメータに指定します。詳細は「4.4.2.1.2 JobCenter MG/SVのコンテナイメージの設定」を参照してください。

4.3.2. JobCenerService Operatorのデプロイ

JobCenterService Operatorをデプロイして起動する手順について説明します。

本手順はOpenShift管理者ユーザが実施してください。

1. JobCenterのパッケージに同梱されている以下のディレクトリを、任意の場所に配置します。

   ■CONTAINER/OPENSHIFT/MANIFEST/OPENSHIFT_ADMIN

2. OpenShift管理者ユーザでOpenShiftにログインします。

```
$ oc login -u <OpenShift管理者ユーザ名> -p <パスワード> --server=<OpenShiftのサーバのアドレス>
```

3. CR JobCenterServiceを利用するためのCRD / ClusterRole を作成します。

```
$ oc apply -f OPENSHIFT_ADMIN/CUSTOM_RESOURCE_DEFINITION
```

CRDおよびClusterRoleは、OpenShiftクラスター全体に影響するリソースです。そのため、一度作成を行った場合、以後作成する必要はありません。

作成されているかどうかは、以下のocコマンドで確認できます。

```
$ oc get customresourcedefinition jobcenterservices.jobcenter.nec.com
```

```
$ oc get clusterrole jobcenter-cr-edit jobcenter-cr-view
```

またCRDの適用時、OpenShiftのバージョンによって以下のwarningが出力される可能性があります。この場合も上記のコマンドでCRDが作成されていることが確認できれば問題はありません。

```
apiextensions.k8s.io/v1beta1 CustomResourceDefinition is deprecated in v1.16+,
 unavailable in v1.22+; use apiextensions.k8s.io/v1 CustomResourceDefinition
```

4. JobCenterService Operatorをデプロイしたいプロジェクトにログインします。

```
$ oc project <プロジェクト名>
```

5. JobCenterService Operatorに使用するServiceAccountをデプロイします。

```
$ oc apply -f OPENSHIFT_ADMIN/JOBCENTERSERVICE_OPERATOR/service_account.yaml
```

6. 以下のディレクトリ配下に存在するマニフェストファイルoperator.yamlのimageフィールドに、「4.3.1.1 JobCenterService Operatorのコンテナイメージの準備」で作成したImageSteramの<オブジェクト名>:<タグ名>を設定します。

   ■OPENSHIFT_ADMIN/JOBCENTERSERVICE_OPERATOR

```
containers:
  - name: jobcenterservice-operator
    # Replace this with the built image name
    image: REPLACE_OPERATOR_IMAGE
```

7. JobCenterService Operatorおよび関連リソースをデプロイします。

```
$ oc apply -f OPENSHIFT_ADMIN/JOBCENTERSERVICE_OPERATOR
```

8. Operatorが起動していることを確認します。STATUSフィールドの値がRunningになっていることを確認してください。

```
$ oc get pod
NAME                                        READY   STATUS    RESTARTS   AGE
jobcenterservice-operator-59f4dfd86f-npxpv   1/1     Running   1          16d
```

9. ServiceAccountにanyuid権限を付与します。

```
$ oc adm policy add-scc-to-user anyuid -z jobcenterservice --as system:admin
```

4.3.3. MG/SVの設定を行うリソースのデプロイ

JobCenterService Operatorが作成するコンテナMG/SVの設定を行う場合、複数のMG/SV間で設定を共通化したり、パスワードなどの機密性の高い情報を秘匿化したりするため、ConfigMapやSecretを作成します。

詳細は以下の各章の内容を参照してください。

必須	章	マニフェストファイル
○	「4.3.3.1 管理者ユーザ/コードワードの設定」	setup_secret.yaml
×	「4.3.3.2 管理者ユーザID/マシンIDの設定」	setup_configmap.yaml
×	「4.3.3.3 一般ユーザのパスワードの設定」	user_password_secret.yaml
×	「4.3.3.4 daemon.confの設定」	daemonconf_configmap.yaml

（凡例）○：JobCenterService Operatorを利用する場合、作成が必須　×：作成は任意

作成に必要なマニフェストファイルは、JobCenterのパッケージに同梱されています。

■CONTAINER/OPENSHIFT/MANIFEST/OPERATOR_USER

4.3.3.1. 管理者ユーザ/コードワードの設定

コンテナ内のJobCenter管理者ユーザ(nsumsmgr)のパスワードと、JobCenterのコードワードを設定するSecretの作成手順を説明します。

1. マニフェストファイル setup_secret.yamlを用意します。
2. マニフェストファイルのdataフィールドの各パラメータの値に、base64エンコードした値を設定します。

必須	パラメータ	値
○	ADMINUSER_PASS	JobCenter管理者ユーザのパスワード
○	CODE_WORD	JobCenterのライセンスを示すコードワード

パスワードに以下の文字は使用できません。

■:

■改行コード

パスワードに以下を設定すると、CL/WinからMG/SVへのログイン時に使用できないため、設定しないでくだい。

■末尾に奇数個の「\」

■対応が取れていない波括弧（例:a{b）

3. 以下のocコマンドを実行し、Secretを作成します。

```
$ oc apply -f setup_secret.yaml
```

作成したSecretのオブジェクト名を、CRのマニフェストファイルで設定することで、MG/SVコンテナに設定が行われます。マニフェストファイルでの設定方法は「4.4.2.1.1 JobCenter MG/SVセットアップ時の設定」を参照してください。

4.3.3.2. 管理者ユーザID/マシンIDの設定

コンテナ内のJobCenter管理者ユーザ(nsumsmgr)のユーザIDと、JobCenterのマシンIDを設定するConfigMapの作成手順を説明します。

1. マニフェストファイル setup_configmap.yamlを用意します。

2. マニフェストファイルのdataフィールドの各パラメータの値を設定します。

必須	パラメータ	値	値の範囲	デフォルト値
×	MIN_MACHINE_ID	Operatorが採番するマシンIDの最小値	1 - 2000000000	300000
×	ADMINUSER_ID	JobCenter管理者ユーザのユーザID	1000 - 60000	2000

Operatorが作成するMG/SVのマシンIDは、MIN_MACHINE_IDパラメータで指定した値以上の値が採番されます。

3. 以下のocコマンドを実行し、ConfigMapを作成します。

```
$ oc apply -f setup_configmap.yaml
```

作成したConfigMapのオブジェクト名を、CRのマニフェストファイルで設定することで、MG/SVコンテナに設定が行われます。マニフェストファイルでの設定方法は「4.4.2.1.1 JobCenter MG/SVセットアップ時の設定」を参照してください。

4.3.3.3. 一般ユーザのパスワードの設定

コンテナ内の一般ユーザのパスワードを設定するSecretの作成手順を説明します。

1. マニフェストファイル user_password_secret.yamlを用意します。
2. マニフェストファイルのdataフィールドの各パラメータの値に、base64エンコードした値を設定します。

必須	パラメータ	値
○	PASSWORD	作成するユーザのパスワード

パスワードに以下の文字は使用できません。

■:

■改行コード

パスワードに以下を設定すると、CL/WinからMG/SV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。

■末尾に奇数個の「\」

■対応が取れていない波括弧（例:a{b）

3. 以下のocコマンドを実行し、Secretを作成します。

```
$ oc apply -f user_password_secret.yaml
```

作成したSecretのオブジェクト名を、CRのマニフェストファイルで設定することで、MG/SVコンテナに設定が行われます。マニフェストファイルでの設定方法は「4.4.2.1.11　一般ユーザの設定」を参照してください。

4.3.3.4. daemon.confの設定

コンテナ内のJobCenter MG/SVのdaemon.confを設定するConfigMapの作成手順を説明します。

1. マニフェストファイル daemonconf_configmap.yamlを用意します。
2. マニフェストファイルのdataフィールドの各パラメータの値を設定します。

必須	パラメータ	値
○	daemon.conf	JobCenter MG/SVに設定したいdaemon.confのパラメータ

daemon.confで設定するパラメータについては、<クラシックモード用環境構築ガイド>の「5.2 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。

3. 以下のocコマンドを実行し、ConfigMapを作成します。

```
$ oc apply -f daemonconf_configmap.yaml
```

作成したConfigMapのオブジェクト名を、CRのマニフェストファイルで設定することで、MG/SVコンテナに設定が行われます。マニフェストファイルでの設定方法は「4.4.2.1.4 JobCenter MG/SVの動作の設定」を参照してください。

daemon.confに以下のパラメータを設定しないでください。設定した場合、コンテナ内のMG/SVが正常に動作しません。

■ipaddress=<IPアドレス>

■bindmode=ipv6

■local_daemon=OFF

4.4. Operatorの運用

OperatorはCRを用いて操作します。CRを作成すると、OperatorはCRのパラメータに従ってOpenShiftのリソースの作成や管理などを行います。

Operatorと、操作を行うCRは以下のとおりです。

Operator	CR	機能
JobCenterService Operator	JobCenterService	JobCenter MG/SVコンテナの作成/管理

4.4.1. CRのライフサイクル

CRのライフサイクルについて説明します。

■CRの作成

マニフェストファイルにパラメータを設定して、CRを作成します。Operatorは作成されたCRのパラメータに従ってKubernetesのリソースを作成し、作成後マシン連携の設定や定期的なバックアップを行います。

■CRの更新

マニフェストファイルにパラメータを設定して、CRを更新します。Operatorは更新されたCRのパラメータに従ってKubernetesのリソースを更新します。MG/SVコンテナの設定に関するCRのパラメータを更新した場合、MG/SVコンテナは一度削除され、更新後のパラメータで再作成されます。

■CRの削除

存在しているCRのオブジェクトを削除します。Operatorはマシン連携の設定の削除などの終了処理を行った後、作成したKubernetesのリソースを削除します。

4.4.2. CRの作成

CRを作成すると、Operatorは作成したCRのパラメータに従って、リソースの作成や設定を行います。

マニフェストファイルにパラメータを設定し、以下のocコマンドを実行して、CRを作成します。

```
$ oc apply -f <マニフェストファイル>
```

マニフェストファイルに設定するパラメータについては、以下の章の内容を参照してください。

4.4.2.1. JobCenterService

CR JobCenterServiceの機能について説明します。

CR JobCenterServiceはMG/SVコンテナの作成/管理を行うCRです。CRを作成すると、MG/SVコンテナが作成されます。また、パラメータを設定することでJobCenterデータの永続化やバックアップ、複数のJobCenterコンテナ間におけるマシン連携などを自動で行います。

マニフェストファイルのspecフィールドに設定できるパラメータおよび機能については、以下のとおりです。

必須	フィールド	章
○	setup	「4.4.2.1.1 JobCenter MG/SVセットアップ時の設定」
○	image	「4.4.2.1.2 JobCenter MG/SVのコンテナイメージの設定」
×	expose	「4.4.2.1.3 OpenShift外部との通信設定」
×	config	「4.4.2.1.4 JobCenter MG/SV の動作の設定」
×	volume	「4.4.2.1.5 コンテナ内のデータの永続化の設定」
×	backup	「4.4.2.1.6 バックアップの設定」
×	timezone	「4.4.2.1.7 タイムゾーンの設定」
×	cluster	「4.4.2.1.8 マシン連携の設定」
×	livenessProbe	「4.4.2.1.9 コンテナのヘルスチェックの設定」
×	safeMode	「4.4.2.1.10 セーフモードの設定」
×	users	「4.4.2.1.11 一般ユーザの設定」

（凡例）○：specフィールドを指定する場合、パラメータの指定が必須　×：指定は任意

4.4.2.1.1. JobCenter MG/SVセットアップ時の設定

Operatorが作成するJobCenter MG/SVコンテナのセットアップ時の設定を行います。

setupフィールドに設定できるパラメータは以下のとおりです。

必須	パラメータ	説明
○	secretName	「4.3.3.1 管理者ユーザ/コードワードの設定」で作成したSecretのオブジェクト名
×	configMapName	「4.3.3.2 管理者ユーザID/マシンIDの設定」で作成したConfigMapのオブジェクト名

（凡例）○：setupフィールドを指定する場合、指定が必須　×：指定は任意

マニフェストファイル設定例

```
spec:
  setup:
    secretName: jobcenterservice-setup
    configMapName: jobcenterservice-setup
```

■setup.secretName

- [型]

  string

- [値の範囲]

  253文字以下

  半角英数字(大文字は不可)、「.」、「-」（先頭/末尾は英数のみ）

- [説明]

  JobCenterの管理者ユーザパスワードとコードワードの設定を行うパラメータです。「4.3.3.1 管理者ユーザ/コードワードの設定」で作成したSecretのオブジェクト名を指定してください

■setup.configMapName

- [型]

  string

- [値の範囲]

  253文字以下

  半角英数字(大文字は不可)、「.」、「-」（先頭/末尾は英数のみ）

- [説明]

  JobCenterの管理者ユーザIDとマシンIDの設定を行うパラメータです。「4.3.3.2　管理者ユーザID/マシンIDの設定」で作成したSecretのオブジェクト名を指定してください。本パラメータが未指定の場合、管理者ユーザIDおよびマシンIDは、「4.3.3.2　管理者ユーザID/マシンIDの設定」に記載されているデフォルト値が設定されます。

4.4.2.1.2. JobCenter MG/SVのコンテナイメージの設定

Operatorが作成するJobCenter MG/SVのコンテナイメージを設定します。

パラメータに設定する値は以下のとおりです。

必須	パラメータ	説明
○	image	JobCenter MG/SVのコンテナイメージリポジトリ

（凡例）○：パラメータの指定が必須　×：指定は任意

マニフェストファイル設定例

```
spec:
  image: openshift-image-registry/develop/jobcenter_mgsv:15.4
```

■image

- [型]

  string

- [値の範囲]

  なし

- [説明]

  Operatorが作成するMG/SVコンテナのイメージを設定するパラメータです。「4.3.1.2 MG/SVのコンテナイメージの準備」で用意したImageStream上に存在するJobCenter MG/SVのコンテナイメージのリポジトリ名を指定します。

4.4.2.1.3. OpenShift外部との通信設定

OpenShifft上のMG/SVコンテナが、OpenShift外部に存在するCL/Winと連携する際の接続方法を設定します。

exposeパラメータに設定する値は以下のとおりです。

必須	パラメータ	説明
○	type	接続方式
×	port	CL/WinからコンテナのMG/SV に対して接続する際のポート番号

（凡例）○：exposeフィールドを指定する場合、指定が必須　×：指定は任意

マニフェストファイル設定例

```
spec:
  expose:
    type: client
    port: 30611
```

■expose.type

- [型]

  string

- [値の範囲]

  client

- [説明]

  OpenShift外部からコンテナMG/SVへ接続する際の方式を設定するパラメータです。CL/Winから接続を行う場合、clientを指定します。

■expose.port

- [型]

  integer

- [値の範囲]

  1 - 65535

- [説明]

  CL/WinからコンテナMG/SVへ接続する際に使用する、comagentのポート番号を設定するパラメータです。typeパラメータが指定されていて、本パラメータが未指定の場合、Kubernetesが確保しているサービス用のポート番号（デフォルトは 30000-32767）から任意のポートが設定されます。

4.4.2.1.4. JobCenter MG/SV の動作の設定

Operatorが作成するMG/SVコンテナのdaemon.confの設定を行います。

configフィールドに設定できるパラメータは以下のとおりです。

必須	パラメータ		説明
○	daemonConf	configMapName	「4.3.3.4 daemon.confの設定」で作成したConfigMapのオブジェクト名
×		subPath	作成したConfigMapのSubPath

（凡例）○：daemonConfフィールドを指定する場合、指定が必須　×：指定は任意

マニフェストファイル設定例

```
spec:
  config:
    daemonConf:
      configMapName: jobcenterservice-daemonconf
      subPath: daemon.conf
```

■config.daemonConf.configMapName

- [型]

  string

- [値の範囲]

  253文字以下

  半角英数字(大文字は不可)、「.」、「-」（先頭/末尾は英数のみ）

- [説明]

  JobCenter MG/SV のdaemon.confの設定を行うパラメータです。「4.3.3.4 daemon.confの設定」で作成したConfigMapのオブジェクト名を指定してください

■config.daemonConf.subPath

- [型]

  string

- [値の範囲]

  253文字以下

  半角英数字、「.」、「-」、「_」

- [デフォルト値]

  daemon.conf

- [説明]

  ConfigMapのSubPathを設定するパラメータです。作成したConfigMapのうち、daemon.confの設定を行ったSubPathを指定してください。

CRを一度作成した後にdaemon.confの設定を変更したい場合、configMapNameまたはsubPathパラメータの値が現在の設定値とは異なる値になるよう、「4.3.3.4 daemon.confの設定」でConfigMapの再作成を行った後、CRのパラメータの更新を行ってください。

4.4.2.1.5. コンテナ内のデータの永続化の設定

MG/SVコンテナ内のデータをOpenShift上のボリュームで永続化する設定をします。コンテナ内のデータをOpenShift上のボリュームにて永続化することで、更新や削除などでコンテナが再作成された際に、データを引き継ぐことができます。

本機能を利用する場合、事前に「4.2.3　コンテナのデータ管理」に従ってPersistentVolumeを作成してください。

volumeフィールドに設定できるパラメータは以下のとおりです。

必須	パラメータ	説明
×	data	スプールディレクトリの永続化に関する設定を行うパラメータです。詳細は「4.4.2.1.5.1 JobCenter MG/SV のデータの永続化の設定」を参照してください。
×	backup	バックアップファイルの永続化に関する設定を行うパラメータです。詳細は「4.4.2.1.5.2 バックアップファイルの永続化の設定」を参照してください。

（凡例）○：volumeフィールドを指定する場合、指定が必須　×：指定は任意

4.4.2.1.5.1. JobCenter MG/SV のデータの永続化の設定

JobCenterのスプールディレクトリの永続化に関する設定を行います。本パラメータを設定することで、定義ファイルや実行中のジョブの実行結果などを永続化できます。

dataフィールドに設定できるパラメータは以下のとおりです。

必須	パラメータ	説明
○	storageSize	使用するPresistentVolumeのストレージサイズ
×	storageClassName	使用するPresistentVolumeのストレージクラス

（凡例）○：dataフィールドを指定する場合、指定が必須　×：指定は任意

マニフェストファイル設定例

```
spec:
  volume:
    data:
      storageSize: 20Gi
      storageClassName: gp2
```

■volume.data.storageSize

- [型]

  string

- [値の範囲]

  数字 + 単位

  

  指定方法の詳細は、以下のKubernetesドキュメントを参照してください

  https://kubernetes.io/docs/concepts/configuration/manage-compute-resources-container/#meaning-of-memory

- [説明]

使用するPresistentVolumeのストレージサイズを設定するパラメータです。永続化ボリュームとして利用したいストレージサイズを指定してください。

■volume.data.storageClassName

- [型]

  string

- [値の範囲]

  253文字以下

  半角英数字(大文字は不可)、「.」、「-」（先頭/末尾は英数のみ）

- [説明]

  使用するPresistentVolumeのストレージクラス名を設定するパラメータです。作成したPresistentVolumeで設定したストレージクラス名を指定してください。storageSizeパラメータが指定されていて、本パラメータが未指定の場合、Kubernetesのデフォルトのストレージクラスを使用します。

4.4.2.1.5.2. バックアップファイルの永続化の設定

JobCenterのバックアップファイルの永続化に関する設定を行います。本パラメータを設定することで、「4.4.2.1.6 バックアップの設定」にてバックアップしたファイルを永続化できます。

必須	パラメータ	説明
○	storageSize	使用するPresistentVolumeのストレージサイズ
×	storageClassName	使用するPresistentVolumeのストレージクラス

（凡例）○：backupフィールドを指定する場合、指定が必須　×：指定は任意

マニフェストファイル設定例

```
spec:
  volume:
    backup:
      storageSize: 20Gi
      storageClassName: gp2
```

■volume.backup.storageSize

- [型]

  string

- [値の範囲]

  数字 + 単位

  

  指定方法の詳細は、以下のKubernetesドキュメントを参照してください

  https://kubernetes.io/docs/concepts/configuration/manage-compute-resources-container/#meaning-of-memory

- [説明]

  使用するPresistentVolumeのストレージサイズを設定するパラメータです。永続化ボリュームとして利用したいストレージサイズを指定してください。

■volume.backup.storageClassName

- [型]

  string

- [値の範囲]

  253文字以下

  半角英数字(大文字は不可)、「.」、「-」（先頭/末尾は英数のみ）

- [説明]

  使用するPresistentVolumeのストレージクラス名を設定するパラメータです。作成したPresistentVolumeで設定したストレージクラス名を指定してください。storageSizeパラメータが指定されていて、本パラメータが未指定の場合、Kubernetesのデフォルトのストレージクラスを使用します。

4.4.2.1.6. バックアップの設定

コンテナ内のJobCenter MG/SVのバックアップの設定を行います。

JobCenter MG/SVの定義情報と構成情報を定期的に取得して保存します。

backupフィールドに設定できるパラメータは以下のとおりです。

必須	パラメータ		説明
○	schedule	schedule	バックアップを実行するcronの設定
×		target	バックアップ対象
×	retention	period	バックアップしたデータの保存期間(日)
×		sweepAt	バックアップデータの保存期間のチェック時刻

（凡例）○：backupフィールドを指定する場合、指定が必須　×：指定は任意

マニフェストファイル設定例

```
spec:
  backup:
    schedule:
      schedule: "0 0 * * *"
      target: all
    retention:
      period: 90
      sweepAt: 00:00
```

■schedule.schedule

- [型]

  string

- [値の範囲]

  以下のcronのフォーマットが指定できます

  <分> <時> <日> <月> <曜日>

  また、cronの書式以外に、以下の値を指定することもできます

値	説明	同等のcronの書式
@yearly	年に1回実行します	0 0 1 1 *
@monthly	月に1回実行します	0 0 1 * *
@weekly	週に1回実行します	0 0 * * 0
@daily	1日に1回実行します	0 0 * * *
@hourly	1時間に1回実行します	0 * * * *

- [説明]

  バックアップを実行するスケジュールを設定するパラメータです。cronの書式を用いて、バックアップを実行するスケジュールを指定してください。

■schedule.target

- [型]

string

- [値の範囲]

  all/definition/configuration

- [デフォルト値]

  all

- [説明]

  バックアップ対象を設定するパラメータです。定義データをバックアップする場合はdefinitionを、構成情報をバックアップする場合はconfigrationを、両方バックアップする場合はallを指定してください。

■retention.period

- [型]

  integer

- [値の範囲]

  1 - 365

- [デフォルト値]

  90

- [説明]

  バックアップしたファイルの保持期間（単位：日）を設定するパラメータです。

■retention.sweepAt

- [型]

  string

- [値の範囲]

  00:00 - 23:59

- [デフォルト値]

  00:00

- [説明]

  バックアップの保持期間のチェック時刻を設定するパラメータです。指定した時刻にバックアップファイルのチェックを行い、保持期間を過ぎたバックアップファイルを削除します。

4.4.2.1.7. タイムゾーンの設定

Operatorが作成するJobCenter MG/SVのコンテナが使用するタイムゾーンの設定を行います。

タイムゾーンを変更すると、「4.4.2.1.6 バックアップの設定」で設定するパラメータに影響を与えます。

パラメータに設定する値は以下のとおりです。

必須	パラメータ	説明
×	timezone	コンテナが使用するタイムゾーン

（凡例）○：パラメータの指定が必須　×：指定は任意

マニフェストファイル設定例

```
spec:
  timezone: Asia/Tokyo
```

■timezone

- [型]

  string

- [値の範囲]

  tz databaseで設定できるタイムゾーン名

- [デフォルト値]

  UTC

- [説明]

  コンテナが使用するタイムゾーンを設定するパラメータです。

4.4.2.1.8. マシン連携の設定

マシン連携の設定を行います。groupの値が同じで、modeが異なるCRから作成されたJobCenter　MG/SVコンテナを連携対象と見なし、お互いのコンテナMG/SVに対して以下の設定を自動で行います。

■マシン一覧へのマシン追加

■JobCenter管理者ユーザ(nsumsmgr)同士のユーザマッピング

clusterフィールドに設定する値は以下のとおりです。

必須	パラメータ	説明
○	group	JobCenterクラスタのグループ名
○	mode	JobCenterクラスタ内での役割

（凡例）○：clusterフィールドを指定する場合、指定が必須　×：指定は任意

マニフェストファイル設定例

```
spec:
  cluster:
    group: jobcenter-cluster
    mode: manager
```

■cluster.group

- [型]

  string

- [値の範囲]

  半角英数字、「.」、「-」、「_」（先頭/末尾は英数のみ）

- [説明]

  JobCenterクラスタのグループ名を設定するパラメータです。本パラメータの値が同じCR同士は、同一のJobCenterクラスタとみなされます。

■cluster.mode

- [型]

  string

- [値の範囲]

  manager/server

- [説明]

  JobCenterクラスタ内でのJobCenterの役割を設定するパラメータです。同一のJobCenterクラスタ内におけるManagerとServerは、Operatorによって自動でマシン連携が行われます。

マシン連携対象のCRがセーフモード中だった場合、マシン連携の処理が行われません。セーフモード終了時にマシン連携が自動で行われます。

以下のCR同士は、マシン連携が自動で行われます。

■groupが同じ かつ modeが異なるもの

group: develop mode:manager

group: develop mode:server

以下のCR同士は、マシン連携が自動で行われません。

■groupが同じ かつ modeが同じもの

group: develop mode:manager

group: develop mode:manager

■groupが異なるもの

group: develop mode:manager

group: production mode:server

4.4.2.1.9. コンテナのヘルスチェックの設定

Operatorが作成するMG/SVコンテナのヘルスチェックの設定を行います。

コンテナ作成後、パラメータで指定された時間までにコンテナ内のMG/SVが起動しない場合、コンテナの再作成を行います。

livenessProbeフィールドに設定する値は以下のとおりです。

必須	パラメータ	説明
×	initialDelaySeconds	MG/SVが起動するまでの待機時間

（凡例）○：パラメータの指定が必須　×：指定は任意

■マニフェストファイル設定例

```
spec:
  livenessProbe:
    initialDelaySeconds: 180
```

■livenessProbe.initialDelaySeconds

- [型]

  integer

- [値の範囲]

  1 - 2147483647

- [デフォルト値]

  180

- [説明]

  MG/SVコンテナが起動するまでの待機時間を設定するパラメータです。

4.4.2.1.10. セーフモードの設定

JobCenter MG/SVコンテナが障害などにより起動しない場合のセーフモードの設定を行います。

セーフモードで起動することにより、コンテナ内のJobCenterを起動せずにコンテナを作成するため、障害時の調査や情報採取を行えます。

パラメータに設定する値は以下のとおりです。

必須	パラメータ	説明
×	safeMode	セーフモードの設定

（凡例）○：パラメータの指定が必須　×：指定は任意

■マニフェストファイル設定例

```
spec:
  safeMode: "true"
```

■safeMode

- [型]

  string

- [値の範囲]

  true/false

- [デフォルト値]

  false

- [説明]

  セーフモードを設定するパラメータです。MG/SVコンテナをセーフモードで起動したい場合、trueを設定してください。

4.4.2.1.11. 一般ユーザの設定

コンテナ内で作成したいユーザの設定を行います。

usersフィールドに設定できるパラメータは以下のとおりです。

必須	パラメータ		説明
○	<ユーザ名>	password	「4.3.3.3 一般ユーザのパスワードの設定」で作成したSecretのオブジェクト名
○		uid	作成するユーザのuid

（凡例）○：usersフィールドを指定する場合、指定が必須　×：指定は任意

■マニフェストファイル設定例

```
spec:
  users:
    jobcenter-developer:
      password: jobcenter-developer-password
      uid: 2000
```

■<ユーザ名>

- [型]

  string

- [値の範囲]

  ユーザ名の制限については、下記注意事項を参照してください。

- [説明]

  作成するユーザ名を設定するパラメータです。

■<ユーザ名>.password

- [型]

  string

- [値の範囲]

  253文字以下

  半角英数字(大文字は不可)、「.」、「-」（先頭/末尾は英数のみ）

- [説明]

  ユーザのパスワードを設定するパラメータです。「4.3.3.3　一般ユーザのパスワードの設定」で作成したSecretのオブジェクト名を指定してください。

■<ユーザ名>.uid

- [型]

  integer

- [値の範囲]

  1000 - 60000

- [説明]

  ユーザのuidを設定するパラメータです。

ユーザに関する注意事項

■同一のユーザ名のユーザを複数作成しないでください。

■同一のユーザIDを持つユーザを複数作成しないでください。

■作成できるユーザ数については、<クラシックモード用環境構築ガイド>の「3.3.8　登録可能なユーザ数」を参照してください。

■ユーザ名には以下の制限があります

- <クラシックモード用リリースメモ>の「6.1.1　使用不可ユーザ名について」に記載されているユーザ名のユーザは、JobCenterのユーザとしては利用できません。
- ユーザ名がroot および nsumsmgr のユーザは、本機能では作成できません。
- 以下の制限を満たしていないユーザ名を指定した場合、ユーザの作成に失敗します。
  - 32文字以下
  - 英数 + 「.」「-」「_」(先頭は英数「.」「_」のみ)

一度作成したユーザのUIDを変更する場合、<クラシックモード用環境構築ガイド>の「14.4　ユーザのUIDを変更する」に記載されている内容を実施してください。

4.4.3. CRの確認

プロジェクト内に存在するCRの確認を行います。

CRを作成した後、以下のocコマンドを実行して、CRの確認を行います。

```
$ oc get jobcenterservice
NAME    EXPOSE              BACKUPSTATE           LASTBACKUPRESULT   AGE
manager <MASTER_IP>:30611   2020-04-01T12:34:56Z  Success            78h
```

-o wideオプションを指定することで、表示されるパラメータ情報が増えます。

各フィールドに表示される内容については、以下のとおりです。

フィールド名	説明	表示
NAME	オブジェクト名	常に表示
EXPOSE	コンテナMG/SVに対する接続方法	常に表示
PORT(S)	MG/SVが使用しているポート番号	-o wide指定時のみ
MACHINE-ID	MG/SVのマシンID	-o wide指定時のみ
BACKUPSTATE	バックアップの状態	常に表示
LASTBACKUPRESULT	前回行ったバックアップの結果	常に表示
LASTBACKUPTIME	前回バックアップを行った時間	-o wide指定時のみ
AGE	CRを作成してからの経過時間	常に表示

■NAME

CRのオブジェクト名を表示します。

■EXPOSE

OpenShift外部からMG/SVコンテナへの接続方法を表示します。「4.4.2.1.3　OpenShift外部との通信設定」での設定によって、以下の値を表示します。

- OpenShift外部との接続設定を行っていない場合

  <none>

- type: clientとしてOpenShift外部との接続設定を行った場合

  <MASTER_IP>:<ポート番号>

■PORT(S)

MG/SVが使用しているポート番号を表示します。

■MACHINE-ID

MG/SVに設定したマシンIDを表示します。

■BACKUPSTATE

バックアップの設定を表示します。「4.4.2.1.6 バックアップの設定」での設定によって、以下の値を表示します。

- バックアップの設定をしていない場合

UnSchedules

- バックアップの設定が誤っている場合

  Failed

- バックアップが正しく設定されている場合

  <次にバックアップを行う時刻>

■LASTBACKUPRESULTE

前回実行したバックアップの結果を表示します。

- バックアップに成功した場合

  Success

- バックアップに失敗した場合

  Failed

- 一部のバックアップに失敗した場合

  ParticalSuccess

■LASTBACKUPTIME

前回バックアップを行った時刻を表示します。

■AGE

CRが作成されてからの経過時間を表示します。

4.4.4. CRの更新

作成したCRに対してパラメータの更新を行うと、Operatorは更新後のパラメータに従ってリソースや設定の更新を行います。

マニフェストファイルに設定したパラメータの値を変更したあと、以下のocコマンドを実行して、CRを更新します。

```
$ oc apply -f <マニフェストファイル>
```

CRの更新は部分更新ではなく上書き更新になります。変更したいパラメータのみ設定するのではなく、更新後の設定をすべてマニフェストファイルに記載した上でocコマンドを実行してください。

CRの更新によってOperatorが設定の更新を行うパラメータは、以下の表中の「更新に対応」欄に○が記載されているもののみになります。×が記載されているフィールドのパラメータを更新しても、Operatorによって設定の更新は行われません。

フィールド	更新に対応	章
setup	×	「4.4.2.1.1 JobCenter MG/SVセットアップ時の設定」
image	○	「4.4.2.1.2 JobCenter MG/SVのコンテナイメージの設定」
expose	○	「4.4.2.1.3 OpenShift外部との通信設定」
config	○	「4.4.2.1.4 JobCenter MG/SV の動作の設定」
volume	×	「4.4.2.1.5 コンテナ内のデータの永続化の設定」
backup	○	「4.4.2.1.6 バックアップの設定」
timezone	○	「4.4.2.1.7 タイムゾーンの設定」
cluster	○	「4.4.2.1.8 マシン連携の設定」
livenessProbe	○	「4.4.2.1.9 コンテナのヘルスチェックの設定」
safeMode	○	「4.4.2.1.10 セーフモードの設定」
users	○	「4.4.2.1.11 一般ユーザの設定」

4.4.5. CRの削除

作成したCRを削除すると、Operatorは作成したリソースを削除します。

作成したマニフェストファイルがある場合、以下のocコマンドを実行して、CRを削除します。

```
$ oc delete -f <マニフェストファイル>
```

存在するCRのオブジェクトを直接削除する場合は、以下のocコマンドを実行します。

```
$ oc delete jobcenterservice <削除したいオブジェクト名>
```

CRを削除した後に削除したCRと同名のCRを作成する場合、Operatorが作成したKubernetesリソースのオブジェクトの削除処理がすべて完了するまで待機してください。削除処理中に同名のCRを作成した場合、Kubernetesのリソースの作成が正常に行われない場合があります。

4.4.5.1. Operatorが起動していない場合

Operatorが正常に起動していない状態でCRを削除したい場合は、以下の手順に従ってCRの削除を行ってください。

1. 削除したいCRに対してoc editコマンドを実行します

   ```
   $ oc edit jobcenterservice <削除したいオブジェクト名>
   ```

2. metadataフィールドに存在するfinalizersフィールドのパラメータをすべて削除して、パラメータの編集を終了します

   以下はoc editコマンドを実行したときの表示画面の例です。★がついた行を削除してください。

   ```
   # Please edit the object below. Lines beginning with a '#' will be ignored,
   # and an empty file will abort the edit. If an error occurs while saving this file will be
   # reopened with the relevant failures.
   #
   apiVersion: jobcenter.nec.com/v1
   kind: JobCenterService
   metadata:
     annotations:
       kubectl.kubernetes.io/last-applied-configuration: |
         {"apiVersion":"jobcenter.nec.com/v1","kind":"JobCenterService","metadata":{"annotations":
   {},"name":"manager","namespace":"develop"},"spec":{"image":"openshift-image-registry/develop/
   jobcenter_mgsv:15.4","setup":{"expose":{"type":"client"},"secretName":"jobcenterservice-
   setup"}}}
     creationTimestamp: "2020-04-01T12:34:56Z"
     finalizers:                              ★
     - finalizer.jobcenterservice.nec.com     ★
     generation: 1
                           .
                           .
                           .
   ```

3. CRを削除します

   ```
   $ oc delete jobcenterservice <削除したいオブジェクト名>
   ```

4.4.6. CRに関する注意事項

■Operatorが作成したKubernetesのリソースに対して、パラメータを変更して更新したり、オブジェクトを削除しないでください。

■Operatorが作成するKubernetesのリソースのオブジェクトと、同名のオブジェクトをプロジェクト内に作成しないでください。

Operatorが作成するKubernetesのリソースのオブジェクト名は、以下のとおりです。

- Deployment

  <CRのオブジェクト名>

- Service

  jc-<CRのオブジェクト名>

  jc-<CRのオブジェクト名>-client

- PersistentVolumeClaim

  <CRのオブジェクト名>-data

  <CRのオブジェクト名>-backup

- Secret

  jobcenterservice-<CRのオブジェクト名>-userlist

■コンテナ内のMG/SVを、nqsstopコマンドなどによって停止状態にはできません。停止を行うとコンテナが起動していないと見なされ、コンテナの再作成が行われます。コンテナ内のMG/SVを停止した状態でコンテナを起動したい場合、「4.4.2.1.10 セーフモードの設定」でセーフモード状態にしてください。

4.5. Operatorのトラブルシューティング

Operatorによる運用時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。

4.5.1. 障害発生時の対応方法

障害発生時の対応方法を説明します。障害内容ごとに各章の内容を参照してください。

■「4.5.1.1 MG/SVのPodが正常に起動しない場合」

■「4.5.1.2 バックアップに失敗した場合」

4.5.1.1. MG/SVのPodが正常に起動しない場合

CRを作成した後、JobCenterService　Operatorが作成したMG/SVのPodが正常に起動しない場合、以下の手順に従ってエラー原因を確認してください。

1. 以下のocコマンドを実行して、Deployment / ReplicaSet / Pod が作成されているかを確認します。

```
$ oc get all
```

以下はmanagerという名前でCRを作成したときの表示例です

```
NAME                                              READY   STATUS    RESTARTS   AGE
pod/jobcenterservice-operator-6bbd9cd99c-pb67l    1/1     Running   0          22d
pod/manager-556bdf84fd-hvf8q                      1/1     Running   0          16d

NAME                                         READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/jobcenterservice-operator    1/1     1            1           22d
deployment.apps/manager                      1/1     1            1           16d

NAME                                                    DESIRED   CURRENT   READY   AGE
replicaset.apps/jobcenterservice-operator-6bbd9cd99c    1         1         1       22d
replicaset.apps/manager-556bdf84fd                      1         1         1       16d
```

■いずれも作成されていない場合

「4.5.1.1.1 Operatorのログの確認」を行ってください。

■Podが作成されていない場合

「4.5.1.1.2 ReplicaSetのEventログの確認」を行ってください。

2. 以下のocコマンドを実行して、PodのStatusを確認します。

```
$ oc get pod
```

■StatusがCrashLoopBackOFF / Error の場合

「4.5.1.1.3 Podのログの確認」を行ってください。

■StatusがCreateContainerError / CreateContainerConfigError / Pending の場合

「4.5.1.1.4 PodのEventログの確認」を行ってください。

エラー原因を解消した後、CRを一度削除してから再作成し、MG/SVのPodが正常に起動することを確認してください。

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

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、「4.5.2 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。

4.5.1.1.1. Operatorのログの確認

Operatorのログを確認する手順は以下のとおりです。

1. OperatorのPodのオブジェクト名を確認します。

```
$ oc get pod
```

2. OperatorのPodのログを確認します。

```
$ oc logs <オブジェクト名>
```

3. msgフィールドに表示されているメッセージを確認します。

   表示されるエラー内容と対処方法については、以下の表を参照してください。

エラーメッセージ	考えられるエラー原因と対処方法
Failed to find resource. [Secret/<オブジェクト名>]	<オブジェクト名>のSecretが作成されていません。 以下を参照して、<オブジェクト名>のSecretの作成を行ってください。 ■「4.3.3 MG/SVの設定を行うリソースのデプロイ」
Failed to find resource. [ConfigMap/<オブジェクト名>]	<オブジェクト名>のConfigMapが作成されていません。 以下を参照して、<オブジェクト名>のConfigMapの作成を行ってください。 ■「4.3.3 MG/SVの設定を行うリソースのデプロイ」 ■「4.3.2 JobCenerService Operatorのデプロイ」
Failed to create resource. [Service/<オブジェクト名>]	CR JobCenterServiceのexpose.portパラメータの値が不適切です。 「4.4.2.1.3 OpenShift外部との通信設定」を参照し、expose.portパラメータを適切な値に修正した後、CRを更新してください。
Failed to update resource. [Service/<オブジェクト名>]	
Invalid parameter(spec.timezone). [JobCenterService/<オブジェクト名>]	CR JobCenterServiceのtimezoneパラメータの値が不適切です。 「4.4.2.1.7 タイムゾーンの設定」を参照し、timezoneパラメータを適切な値に修正した後、CRを更新してください。
Invalid parameter(spec.backup.retention.sweepAt). [JobCenterService/<オブジェクト名>]	CR JobCenterServiceのbackup.sweepAtパラメータの値が不適切です。 「4.4.2.1.6 バックアップの設定」を参照し、backup.sweepAtパラメータを適切な値に修正した後、CRを更新してください。
Invalid parameter(spec.volume.backup). [JobCenterService/<オブジェクト名>]	CR JobCenterServiceのvolume.backupパラメータの値が不適切です。 「4.4.2.1.5.2 バックアップファイルの永続化の設定」を参照し、volume.backupパラメータを適切な値に修正した後、CRを更新してください。
Invalid parameter(spec.volume.data). [JobCenterService/<オブジェクト名>]	CR JobCenterServiceのvolume.dataパラメータの値が不適切です。

エラーメッセージ	考えられるエラー原因と対処方法
	「4.4.2.1.5.1 JobCenter MG/SV のデータの永続化の設定」を参照し、volume.dataパラメータを適切な値に修正した後、CRを更新してください。
Invalid parameter(spec.users.<ユーザー名>). [JobCenterService/<オブジェクト名>]	CR JobCenterServiceのusers.<ユーザ名>パラメータの値が不適切です。 「4.4.2.1.11 一般ユーザの設定」を参照し、usersパラメータを適切な値に修正した後、CRを更新してください。
Invalid parameter(data.MIN_MACHINE_ID). [ConfigMap/<オブジェクト名>]	ConfigMapのdata.MIN_MACHINE_IDパラメータの値が不適切です。 「4.3.3.2 管理者ユーザID/マシンIDの設定」で作成したConfigMapのマニフェストファイルを確認し、MIN_MACHINE_IDの値を適切な値に修正した後、ConfigMapを更新してください。
Invalid parameter(data.ADMINUSER_ID). [ConfigMap/<オブジェクト名>]	ConfigMapのdata.ADMINUSER_IDパラメータの値が不適切です。 「4.3.3.2 管理者ユーザID/マシンIDの設定」で作成したConfigMapのマニフェストファイルを確認し、ADMINUSER_IDの値を適切な値に修正した後、ConfigMapを更新してください。
Invalid parameter(data). [Secret/<オブジェクト名>]	Secretのdata.ADMINUSER_PASSパラメータの値が不適切です。 「4.3.3.1 管理者ユーザ/コードワードの設定」で作成したSecretのマニフェストファイルを確認し、ADMINUSER_PASSの値を適切な値に修正した後、Secretを更新してください。
Invalid parameter(data.PASSWORD). [Secret/<オブジェクト名>]	Secretのdata.PASSWORDパラメータの値が不適切です。 「4.3.3.3 一般ユーザのパスワードの設定」で作成したSecretのマニフェストファイルを確認し、PASSWORDの値を適切な値に修正した後、Secretを更新してください。

4.5.1.1.2. ReplicaSetのEventログの確認

ReplicaSetのEventログを確認する手順は以下のとおりです。

1. 対象のReplicaSetのオブジェクト名を確認します。

```
$ oc get replicaset
```

2. ReplicaSetの詳細を確認します。

```
$ oc describe replicaset <オブジェクト名>
```

3. Eventsフィールドに表示されているメッセージを確認します。

   表示されるエラー内容と対処方法については、以下の表を参照してください。

エラーメッセージ	考えられるエラー原因と対処方法
serviceaccount "jobcenterservice" not found	Podの作成に必要なServiceAccountが作成されていません。 「4.3.2 JobCenerService Operatorのデプロイ」を参照して、ServiceAccountを作成してください。

4.5.1.1.3. Podのログの確認

Podのログを確認する手順は以下のとおりです。

1. 対象のPodのオブジェクト名を確認します。

```
$ oc get pod
```

2. Podのログを確認します。

```
$ oc logs <オブジェクト名>
```

3. 表示されているメッセージを確認します。

   表示されるエラー内容と対処方法については、以下の表を参照してください。

エラーメッセージ	考えられるエラー原因と対処方法
JobCenter setup is failed.	daemon.confの設定が不適切です。 「4.3.3.4 daemon.confの設定」で作成したConfigMapのマニフェストファイルを確認し、dataフィールドに設定してあるdaemon.confのパラメータを適切な値に修正してください。
ENV JOBCENTER_NSUMSMGR_PASS: the user's password is not applied correctly. [nsumsmgr]	nsumsmgrユーザのパスワードが不適切です。 「4.3.3.1 管理者ユーザ/コードワードの設定」で作成したSecretのマニフェストファイルを確認し、ADMINUSER_PASSの値を適切な値に修正してください。
	Podの作成に必要なServiceAccountにanyuid権限が付与されていません。 「4.3.2 JobCenerService Operatorのデプロイ」を参照して、ServiceAccountにanyuidの権限を付与してください。
Error: No valid license is registered. nqsdaemon start process will exit.	JobCenterのコードワードが不適切です。 「4.3.3.1 管理者ユーザ/コードワードの設定」で作成したSecretのマニフェストファイルを確認し、CODE_WORDの値を適切な値に修正してください。

4.5.1.1.4. PodのEventログの確認

PodのEventログを確認する手順は以下のとおりです。

1. 対象のPodのオブジェクト名を確認します。

```
$ oc get pod
```

2. Podの詳細を確認します。

```
$ oc describe pod <オブジェクト名>
```

3. Eventsフィールドに表示されているメッセージを確認します。

   表示されるエラー内容と対処方法については、以下の表を参照してください。

エラーメッセージ	考えられるエラー原因と対処方法
Error: Couldn't find key ADMINUSER_PASS in Secret	SecretにADMINUSER_PASSパラメータが設定されていません。 「4.3.3.1 管理者ユーザ/コードワードの設定」を参照して、ADMINUSER_PASSパラメータを設定した後、Secretを更新してください。
/etc/opt/wsnlesd/.lockinfo\\\" caused \\\"not a directory\\\"\""	SecretにCODE_WORDパラメータが設定されていません。 「4.3.3.1 管理者ユーザ/コードワードの設定」を参照して、CODE_WORDパラメータを設定した後、Secretを更新してください。
/usr/local/netshep/lib/nqs/rc/daemon.conf\\\" caused \\\"not a directory\\\"\"	ConfigMapにCRのパラメータで指定したSubPathが存在しません。 「4.4.2.1.4 JobCenter MG/SV の動作の設定」を参照して、subPathパラメータを適切な値に修正した後、ConfigMapを更新してください。
pod has unbound immediate PersistentVolumeClaims	PVCのパラメータが不適切です。 「4.4.2.1.5 コンテナ内のデータの永続化の設定」を参照して、volumeパラメータを適切な値に修正した後、CRを更新してください。

4.5.1.2. バックアップに失敗した場合

バックアップに失敗した場合、以下の手順に従ってエラー原因を確認してください。

1. 以下のocコマンドを実行して、対象のCRのBACKUPSTATEおよびLASTBACKUPRESULTフィールドの値を確認します。

```
$ oc get jobcenterservice
```

■BACKUPSTATEがFailedの場合

「4.5.1.2.1 Operatorのログの確認」を行ってください。

■LASTBACKUPRESULTがFailed/PartialSuccessの場合

「4.5.1.2.2 CRの詳細の確認」行ってください。

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、「4.5.2 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。

4.5.1.2.1. Operatorのログの確認

Operatorのログを確認する手順は以下のとおりです。

1. OperatorのPodのオブジェクト名を確認します。

```
$ oc get pod
```

2. OperatorのPodのログを確認します。

```
$ oc logs <オブジェクト名>
```

3. msgフィールドに表示されているメッセージを確認します。

   表示されるエラー内容と対処方法については、以下の表を参照してください。

エラーメッセージ	考えられるエラー原因と対処方法
Failed to create backup schedule. [JobCenterService/<オブジェクト名>]	CR JobCenterServiceのschedule/パラメータが不適切です。 「4.4.2.1.6 バックアップの設定」を参照して、schedule/パラメータを適切な値に修正した後、CRを更新してください。
Failed to update backup schedule. [JobCenterService/<オブジェクト名>]	

4.5.1.2.2. CRの詳細の確認

CRの詳細を確認する手順は以下のとおりです。

1. 対象のCRのオブジェクト名を確認します。

```
$ oc get jobcenterservice
```

2. CRの詳細を確認します。

```
$ oc describe jobcenterservice <オブジェクト名>
```

3. StatusフィールドのLastFailedReasonに表示されているメッセージを確認します。

   表示されるエラー内容と対処方法については、以下の表を参照してください。

エラーメッセージ	考えられるエラー原因と対処方法
Failed to wait until pod status is running.	MG/SVのPodがRunning状態ではありません。 「4.5.1.1 MG/SVのPodが正常に起動しない場合」を参照して、MG/SVのPodが正常に起動することを確認してください。
pods \"<MG/SVのPod名>\" not found	
unknown time zone <timezone/パラメータに指定した値>	CR JobCenterServiceのtimezone/パラメータが不適切です。 「4.4.2.1.7 タイムゾーンの設定」を参照して、timezone/パラメータに適切な値を設定して、CRを更新してください。

4.5.2. 障害発生時の情報採取方法

OpenShift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. 「4.5.2.1 OpenShiftの情報採取」
2. 「4.5.2.2 コンテナ内の情報採取」

4.5.2.1. OpenShiftの情報採取

OpenShiftの情報採取にはjc_ocp_getinfo.shを利用します。

jc_ocp_getinfo.shについては、「4.5.3 jc_ocp_getinfo.sh」を参照してください。

以下の手順を実施して、情報採取を行ってください。

1. JobCenterのパッケージの以下のディレクトリに同梱されているjc_ocp_getinfo.shファイルを入手します。

   ■CONTAINER/OPENSHIFT/SCRIPT

2. jc_ocp_getinfo.sh に実行権を付与します。

```
$ chmod a+x jc_ocp_getinfo.sh
```

3. cluster-admin相当の権限を持ったOpenShiftのユーザでログインします。
4. 情報採取したいプロジェクトで、jc_ocp_getinfo.sh を実行します。

ERRORメッセージが表示されていなければ、情報は正常に取得できています。

5. 以下のデータファイルが作成されていることを確認します。

```
jc_ocp_data_<MMDDhhss>.tar.gz
```

4.5.2.2. コンテナ内の情報採取

障害が発生したMG/SVコンテナの情報採取には、jc_getinfoコマンドを利用します。

jc_getinfoコマンドについては、<クラシックモード用コマンドリファレンス>の「7.1　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」を参照してください。

以下の手順を実施して、情報採取を行ってください。

1. 情報採取を行うPod名を確認します。

```
$ oc get pod
```

2. 取得するPodのStatusがRunningではない場合、セーフモードで起動します。CRのマニフェストファイルに以下の設定を行い、CRを更新します。

```
spec:
  safeMode: "true"
```

セーフモードについては、「4.4.2.1.10 セーフモードの設定」を参照してください。

3. 以下のocコマンドを実行して、コンテナ内でjc_getinfoを実行します。

```
$ oc exec <Pod名> -it -- sh -c 'mkdir /tmp/jcdata ; /usr/lib/nqs/check/jc_getinfo -d /tmp/
jcdata'
```

4. コンテナ内で作成された採取データのファイル名を確認します。jc_getinfoの実行結果の以下のメッセージを確認してください。

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

5. 採取したデータを、ホストマシンにコピーします。

```
$ oc rsync <Pod名>:/tmp/jcdata <コピー先のパス>
```

4.5.3. jc_ocp_getinfo.sh

```
jc_ocp_getinfo.sh [<CRのリソース名>/<オブジェクト名>...]
```

4.5.3.1. 機能説明

■OpenShift環境で障害発生した際、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を自動的に採取します。

■本スクリプトファイルは、JobCenterのパッケージの以下のディレクトリに同梱されています。

- CONTAINER/OPENSHIFT/SCRIPT

■採取した情報は、スクリプトを実行したディレクトリの直下に"jc_ocp_data_<MMDDhhmm>.tar.gz"のファイル名で格納されます。

■本コマンドを実行する前に、以下のことを確認してください。

- cluster-admin相当の権限を所持したユーザでログインしていること

```
$ oc whoami
```

- 情報採取を行いたいプロジェクトにログインしていること

```
$ oc project
```

4.5.3.2. オプション

<CRのリソース名>/<オブジェクト名>

取得するCRのオブジェクトを指定します。本パラメータは複数指定できます。

省略した場合、プロジェクトに存在するすべてのCRのオブジェクトを取得します。

4.5.3.3. 主要メッセージ

■正常系メッセージ

メッセージ	説明
Start to check commands.	スクリプトの実行に必要なコマンドの存在チェックを行います
Finished checking commands.	スクリプトの実行に必要なコマンドの存在チェックが完了しました
Start to check params.	引数で指定されたパラメータチェックを行います
Finished checking params.	引数で指定されたパラメータチェックが完了しました
Start to compress data to <ファイル名>.	採取したデータを圧縮します
Finished compressing data to <ファイル名>.	採取したデータの圧縮が完了しました
Start to get data.	OpenShift環境の情報を取得します
Finished getting data.	OpenShift環境の情報の取得が完了しました
Start to get crs data.	CRに関する情報を取得します
Finished getting crs data.	CRに関する情報の取得が完了しました
Start to get jcs data.	CR JobCenterServiceの情報を取得します

メッセージ	説明
Finished getting jcs data.	CR JobCenterServiceの情報の取得が完了しました
Start to get data of <リソース名>/<オブジェクト名>.	<リソース名>の<オブジェクト名>の情報を取得します
Finished getting data of <リソース名>/<オブジェクト名>.	<リソース名>の<オブジェクト名>の情報を取得が完了しました
Start to get crs common data.	CRに関する共通リソースの情報を取得します
Finished getting crs common data.	CRに関する共通リソースの情報の取得が完了しました
Start to get operators data.	Operatorに関連するリソースの情報を取得します
Finished getting operators data.	Operatorに関連するリソースの情報の取得が完了しました
Start to get cluster data.	OpenShiftクラスターの情報を取得します
Finished getting cluster data.	OpenShiftクラスターの情報の取得が完了しました

■エラーメッセージ

メッセージ	説明
WARNING: Failed to get <取得対象>.	<取得対象>の情報取得に失敗しました
ERROR: Failed to compress data to <ファイル名>.	採取したデータの圧縮に失敗しました
ERROR: Failed to create directory <ディレクトリ名>.	ディレクトリの作成に失敗しました
ERROR: Failed to remove directory <ディレクトリ名>.	ディレクトリの削除に失敗しました
ERROR: Invalid parameter kind <パラメータ名>.	引数で指定したパラメータが不正です
ERROR: Not found command <コマンド名>.	スクリプトの実行に必要なコマンドが存在しません

4.6. Operatorの削除

本章では、OpenShift環境からJobCenterService Operatorを削除する方法を説明します。

以下の手順に従って、削除を行ってください。

1. JobCenterService Operatorをデプロイしたすべてのプロジェクトで、「4.6.1 プロジェクト内のリソースの削除」を実施します。
2. 「4.6.2 OpenShiftクラスタ内のリソースの削除」を実施します。

特定のプロジェクトからJobCenterService Operatorを削除したい場合、「4.6.1 プロジェクト内のリソースの削除」のみを実施してください。

4.6.1. プロジェクト内のリソースの削除

プロジェクト内に存在する、JobCenterService Operatorのリソースの削除方法を説明します。

4.6.1.1. CR JobCenterService の削除

プロジェクト内のCR JobCenterServiceに関連するリソースの削除手順は以下のとおりです。

1. プロジェクト内に存在するCRをすべて削除します。

■オブジェクトの確認

```
$ oc get jobcenterservice
```

■オブジェクトの削除

```
$ oc delete jobcenterservice <オブジェクト名>
```

2. 「4.3.3 MG/SVの設定を行うリソースのデプロイ」で作成したSecret / ConfigMapを削除します。

■オブジェクトの確認

```
$ oc get secret
```

```
$ oc get configmap
```

■オブジェクトの削除

```
$ oc delete secret <オブジェクト名>
```

```
$ oc delete configmap <オブジェクト名>
```

4.6.1.2. JobCenterService Operatorの削除

プロジェクト内のJobCenterService Operatorに関連するリソースの削除手順は以下のとおりです。

本手順はOpenShift管理者で実施してください。

1. ServiceAccountを削除します。

■オブジェクトの確認

```
$ oc get serviceaccount jobcenterservice jobcenterservice-operator
```

■オブジェクトの削除

```
$ oc delete serviceaccount jobcenterservice jobcenterservice-operator
```

2. ServiceAccountに付与した、anyuid権限を削除します。

```
$ oc adm policy remove-scc-from-user anyuid -z jobcenterservice --as system:admin
```

3. Roleを削除します。

■オブジェクトの確認

```
$ oc get role jobcenterservice-operator
```

■オブジェクトの削除

```
$ oc delete role jobcenterservice-operator
```

4. RoleBindingを削除します。

■オブジェクトの確認

```
$ oc get rolebinding jobcenterservice-operator
```

■オブジェクトの削除

```
$ oc delete rolebinding jobcenterservice-operator
```

5. Deploymentを削除します。

■オブジェクトの確認

```
$ oc get deployment jobcenterservice-operator
```

■オブジェクトの削除

```
$ oc delete deployment jobcenterservice-operator
```

6. ConfigMapを削除します。

■オブジェクトの確認

```
$ oc get configmap jobcenterservice-healthy jobcenterservice-setmachinelinkage
```

■オブジェクトの削除

```
$ oc delete configmap jobcenterservice-healthy jobcenterservice-setmachinelinkage
```

7. 「4.3.1 イメージの準備」で作成したImageStreamを削除します。

■オブジェクトの確認

```
$ oc get imagestream
```

■オブジェクトの削除

```
$ oc delete imagestream <オブジェクト名>
```

4.6.2. OpenShiftクラスタ内のリソースの削除

OpenShiftクラスタ内のリソースの削除手順は以下のとおりです。

本手順はOpenShift管理者で実施してください。

1. OpenShiftクラスタ内のすべてのプロジェクトからCRが削除されていることを確認します。

```
$ oc get jobcenterservices --all-namespaces
```

未削除のCRが存在する場合、CRが存在するプロジェクトで「4.6.1 プロジェクト内のリソースの削除」を実施してください。

2. CustomResourceDefinitionを削除します。

■オブジェクトの確認

```
$ oc get customresourcedefinition jobcenterservices.jobcenter.nec.com
```

■オブジェクトの削除

```
$ oc delete customresourcedefinition jobcenterservices.jobcenter.nec.com
```

3. ClusterRoleを削除します。

■オブジェクトの確認

```
$ oc get clusterrole jobcenter-cr-edit jobcenter-cr-view
```

■オブジェクトの削除

```
$ oc delete clusterrole jobcenter-cr-edit jobcenter-cr-view
```

4. 「4.2.3 コンテナのデータ管理」で、PresistentVolumeを作成した場合、削除します。

■オブジェクトの確認

```
$ oc get presistentvolume
```

■オブジェクトの削除

```
$ oc delete presistentvolume <オブジェクト名>
```

発行年月 Apr 2024