Red Hat OpenShift Red Hat OpenShift 入門



■本冊子について

本冊子は、『OpenShift徹底入門』(翔泳社)の内容の一部(3部構成の第1部)を著者、出版社の了解のもと抜粋したものです。



書籍『OpenShift徹底入門』

刊行: 翔泳社

・著者:株式会社リアルグローブ・オートメーティッド、青島 秀治、レッドハット株式会社、森 真也、 清水 護、織 学、宇都宮 卓也、斎藤 和史、野間 亮志、荒木 俊博
・監修:須江 信洋

https://www.amazon.co.jp/dp/4798172553/

.....

発売日:2022年01月19日

.....

定価:4,180円(本体3,800円+税10%)

仕様:B5変・648ページ

ISBN 978-4-7981-7255-2



株式会社リアルグローブ・オートメーティッド 青島 秀治、 レッドハット株式会社 森 真也、清水 護、織 学、宇都宮 卓也、 斎藤 和史、野間 亮志、荒木 俊博 /著 レッドハット株式会社 須江 信洋 / 監修



はじめに

本書は、エンタープライズ向け Kubernetes ディストリビューションの1つである Red Hat OpenShift Container Platform (以下、OpenShift) に関する技術解説書です。OpenShiftは CNCF (Cloud Native Computing Foundation)の Kubernetes 互換認定を取得しており、エンタープライズシステム向けの Kubernetes として利用できます。また、エンタープライズ環境で扱いやすいように機能を追加するな ど、様々な工夫を凝らしています。それらの諸機能を理解してうまく使いこなしていくため、本書で は実際のユースケースを考慮し、製品ドキュメントとは違った切り口でまとまった情報を提供してい ます。限られた紙面の中でではありますが、実際の操作画面やコマンドラインの例、YAMLファイル のサンプル、ログの出力例など、現場で役立つ情報を多数掲載しています。

本書は紙幅が限られているため、OpenShiftのコアである Kubernetes 自体についての解説は必要 最小限にとどめています。Kubernetesの解説書は和書・洋書ともに多数の良書が出版されています。 Kubernetes および Linux コンテナに関する詳細情報についてはそれらをあわせてご参照ください。

本書の構成は以下のようになっています。はじめて OpenShift に触れる方はもちろん、ある程度 OpenShift に慣れてきて実践的な知識やノウハウを求める方にも役立つ内容がまとまっています。

第1部 入門編

OpenShiftの概要を理解し、AWS上での実際のインストールの手順、およびインストール後の基本 的な使い方を一通り体験することができます。これにより、独力で OpenShiftを導入して環境を準備 することが可能になります。これまで OpenShiftを使ったことがない方は、ここから始めることをお すすめします。

第2部 インフラ実践編

OpenShiftのアーキテクチャを理解し、複雑なインフラ要件に合わせて OpenShift のインフラ設計 および導入を行うために検討しておくべきポイントを把握できます。特にネットワークやストレージ、 セキュリティに関して詳しく解説しています。また、導入後の運用やトラブルシューティングにおいて 役立つ情報もまとめられています。

第3部 アプリ実践編

OpenShiftにおけるアプリケーションのビルドやデプロイの手順や利用可能なCI/CDツール、および設計指針について解説しています。これにより、OpenShift上で稼働するアプリケーションの開発 やビルド・デプロイおよびCI/CDが実践できるようになります。既存のアプリケーションを単にコン テナ化するだけでなく、OpenShiftを活用してクラウドネイティブなアプリケーションを開発・運用す るためのノウハウについてもまとめられています。 本書では特に断りがない限り、執筆時点での最新バージョンである OpenShift 4.9を対象としています。実際に OpenShiftを利用する場合は、バージョンアップに伴う変更等によって本書の記載内容と異なる可能性があることにご注意ください。ご使用になる際は、都度最新バージョンのドキュメントを確認されることをおすすめします。次のサイトを参照するようにしてください^[1]。

Product Documentation for OpenShift Container Platform 4.9

https://access.redhat.com/documentation/ja-jp/openshift_container_platform/4.9

最後に、本書はコンサルタントやサポート、プリセールスなど、複数の立場からOpenShiftに携 わっているメンバーから有志を募って執筆しました。そのため、記述内容は概説的なものにとどまら ず、日々の業務で培われた実践的かつ実用性の高いものが詰まっています。OpenShiftは、その基盤 となるKubernetesだけでなく、関連する多数のOSS(オープンソースソフトウェア)を含んだプロダ クトであり、高頻度でアップデートされる内容をキャッチアップして整理するのはそれなりに難度の高 い作業でした。執筆陣の努力により、本書は比較的普遍性の高い内容に仕上がっていると思います。 本書で学んだ内容がすぐに古くなってしまうということはありませんのでご安心ください。

本書の執筆にあたっては、内容の検討やレビューで以下の方々にご協力いただきました。伊藤拓 也、暮林達也、林智史、木村貴由、福地大輔、今井大亮の皆さん(敬称略)。この場を借りて感謝の 意を表します。

> 執筆者にかえて 須江 信洋

 ⁽¹⁾ 英語ですが、https://docs.openshift.com/container-platform/4.9/にもドキュメントがあります。こちらのサイトでは、すべてのブック を串刺し検索できるので便利です。

本書の読み方

■概要と対象読者

本書は、エンタープライズ向け Kubernetes ディストリビューションの1つである Red Hat OpenShift Container Platform (以下、OpenShift) に関する技術解説書です。エキスパートの経験に基づき、 OpenShiftを実際に使うにあたって有用な情報をインフラとアプリケーションの両面から集大成して います。また、環境構築の章では自分で OpenShiftをインストールして利用するための手順も解説し ており、これから OpenShift に触れる方にとっても役立つ内容になっています。 本書の想定する対象読者は以下のとおりです。

本書の想定する対象

読者は以下のとおりです

- これからOpenShiftを使う方
- OpenShiftの基盤設計や運用に関わる方
- OpenShift上で稼働するアプリケーションの設計・開発・運用に関わる方

なお、紙面の制約上、本書ではOpenShiftのコアであるKubernetes自体についての解説は必要最 小限にとどめています。KubernetesやLinuxコンテナに関する詳細情報については本書以外の解説 書をあわせてご参照ください。

サンプルについて

本書に掲載しているコードや設定ファイルのサンプルはGitHubから入手できます。

https://github.com/team-ohc-jp-place/openshift-tettei-nyumon

■動作確認環境

本書に掲載しているコードや手順は、可能な限り執筆時点(2021年11月)の最新バージョンであ る OpenShift 4.9で動作を確認していますが、完全な動作を保証するものではないことにご注意くだ さい。また、各種制約により、異なるバージョンでの動作を前提とする箇所はその旨を明記していま す。

- OpenShiftクラスタへのアクセスには、Windows/macOS/Linuxが利用可能です。Webコンソー ルを利用するには、Webブラウザ(ChromeやFirefoxなど)が必要です。
- OpenShiftのインストールを実行するにはmacOSかLinuxが必要ですが、本書内ではクラウド環境のLinux仮想マシンを利用する手順も紹介しています。

■本書の構成

第1部 入門編

OpenShiftの概要が理解でき、実際にインストールや基本的な操作ができるようになります。これ までOpenShiftを使ったことがない方は、ここから始めることをおすすめします。

- 第1章 OpenShiftの概要:クラウドネイティブ技術の市場動向を踏まえ、OpenShiftの特徴や コンセプト、利用形態などを解説します。
- 第2章 環境構築:AWS上でOpenShiftをインストールする手順を説明します。
- 第3章 基本的な操作方法: OpenShiftのWebコンソールとコマンドラインインターフェース (CLI)の利用方法を説明します。
- 第4章 ユーザー管理:ユーザーと権限の管理方法について説明します。
- 第5章 アプリケーションの実行:サンプルアプリケーションのデプロイ方法を説明します。

第2部 インフラ実践編

OpenShiftのアーキテクチャが理解でき、複雑なインフラ要件に合わせてOpenShiftの設計や導入 を行う際の検討ポイントを把握できます。また、導入後の運用やトラブルシューティングにおいて役 立つ情報もまとめられています。

- 第6章 OCPのインストール方法:設計とインストールにおける考慮点の説明と、最も汎用性の高いAny Platform インストールの手順を紹介します。
- 第7章 Operator:アプリケーションやOpenShiftクラスタの運用を自律化するOperatorの概要およびOperator Frameworkなどについて説明します。
- 第8章 ネットワーク: OpenShift におけるネットワーク関連の話題、特に Router や Egress といった OpenShift 特有の機能や、NetworkPolicy や Multus といった発展的な内容について解説します。
- 第9章 ストレージ: OpenShiftにおけるストレージの存在意義と概念、取り扱いについて説明 します。
- 第10章 セキュリティ:コンテナアプリケーションにおけるセキュリティの考え方やOpenShift 関連製品が提供するセキュリティ機能について説明します。
- 第11章 Day 2オペレーション:インストール後の運用やトラブルシューティングについて解説します。
- 第12章 OpenShiftとマルチクラウド: OpenShift によって実現する世界観としての Open Hybrid Cloudと、マルチクラウドに関連する OpenShiftの機能や関連する製品ポートフォリオを 紹介します。

第3部 アプリ実践編

OpenShift上で稼働するアプリケーションの開発やビルド・デプロイおよびCI/CDが実践できるようになります。既存のアプリケーションを単にコンテナ化するだけでなく、OpenShiftを活用してクラウドネイティブなアプリケーションを開発・運用するためのノウハウについてもまとめられています。

- 第13章 アプリケーションのコンテナ化とモダナイズ:アプリケーションをコンテナ化する際の 前提知識となる、コンテナの特徴、コンテナデザインパターン、アプリケーションのモダナイズ のアプローチなどについて説明します。
- 第14章 コンテナイメージのビルド: OpenShift でのコンテナイメージのビルドに必要な前提知 識について説明します。
- 第15章 アプリケーションのデプロイ: OpenShift でのアプリケーションデプロイ方法について 説明します。
- 第16章 アプリケーション開発におけるライフサイクル:アプリケーションのライフサイクルに
 関してメトリクス・ロギング・トレーシングの3つの視点からの紹介と、OpenShift上での実現方
 法について解説します。
- 第17章 クラウドネイティブ開発: クラウドネイティブとは何かの説明から、サービスメッシュ やサーバーレスなどのクラウドネイティブ開発に関する技術と OpenShift との関係について説明 します。
- 第18章 DevOps: OpenShiftで実践する DevOpsについて解説します。OpenShiftで利用できる、DevOpsツールの特徴や使い方に加え、効果的な DevOpsを実践するうえでのポイントについて紹介します。

目次

はじめに	ii
本書の読み方	iv

Part 1 入門編

1.1	クラウ	パネイティブ技術の市場動向	
	1.1.1	CNCF Projects	
	1.1.2	CI/CD — 継続的インテグレーションと継続的デリバリー	
	1.1.3	Observability and Analysis――オブザーバビリティとアナリシス	
1.2	Kuber	rnetesとOpenShiftの違い	
	1.2.1	クラスタ構成	
	1.2.2	クラスタの構築	
	1.2.3	拡張コンポーネント	
	1.2.4	モニタリング	
	1.2.5	ロギング	
	1.2.6	ネットワーク	
	1.2.7	ストレージ	
	1.2.8	クラスタアップグレード	
	1.2.9	CI/CD	
	1.2.10	その他	
1.3	Open	Shift利用形態とサブスクリプション	
	1.3.1	マネージドOpenShiftサービス	
	1.3.2	セルフマネージドOpenShiftとサブスクリプション	

2 環境構築

2	1	
∠		

1

2.1	IPIイン	ノストール	22
	2.1.1	アカウント作成	23
	2.1.2	AWSアカウントの設定	23
	2.1.3	クラスタ作成前の準備	30
	2.1.4	クラスタ作成	37
	2.1.5	補足 クラスタ削除	38
2.2	サーバ	証明書の設置	38
	2.2.1	acme.shでサーバー証明書を発行	39
	2.2.2	Routerに証明書を設置	41
2.3	Red H	lat CodeReady Containers環境	42
	2.3.1	システム要件	42

2.3.2	インストール手順
2.3.3	アップグレード手順
2.3.4	使い方

ログイン 52

3 基本的な操作方法

3.1.1

3.1.2

3.2.1

3.2.2	_{oc} コマンドでログイン	57
3.2.3	ocコマンドの使い方	58

3.1

3.2

4 ユーザー管理

4.1	認証と	:認可	. 62
	4.1.1	認証 (Authentication)	62
	4.1.2	認可(Authorization)	63
4.2	ユーサ	「一作成	. 64
	4.2.1	.htpasswdファイルを作成	64
	4.2.2	, .htpasswdファイルのSecretを作成	65
	4.2.3	Identity Providerを登録	65
	4.2.4	新規作成したユーザーでログイン	67
	4.2.5	ユーザーに権限を付与	67
4.3	ユーサ	「一削除	. 68
	4.3.1	kubeadminの削除	68
	4.3.2	kubeadmin以外のユーザーの削除	69

5 アプリケーションの実行

5.1	プロシ	ジェクト	72
	5.1.1	プロジェクトの作成	. 72
5.2	プロシ	ジェクトの操作権限をユーザーに付与	73
5.3	アプリ	ターションのライフサイクル管理	75
	5.3.1	アプリケーションのデプロイ方法	. 75
	5.3.2	トポロジー画面	. 79

71

51

Part 2 インフラ実践編

83

83

6 OCPのインストール方法

4 1	<i></i>	,トフォーム とインフトール方法の選択	0 /
0.1	611		. 04
	0.1.1 6.1.0	IPI (Installer-Provisioned Infrastructure)	04
	0.1.Z	UPI (User-Provisioned Infrastructure)	00
	0.1.3	リホートするノブツトノオームとイノストール方法	80
	6.1.4		86
	6.1.5	RHCOSのコンボーネント	87
	6.1.6	Ignition による自動構成	87
	6.1.7	Any Platform インストール	88
6.2	クラス	く夕の設計	. 89
	6.2.1	設計のポイント	89
	6.2.2	ノード	89
	6.2.3	ノードのサイジング	93
	6.2.4	ネットワーク	96
	6.2.5	ストレージ	99
6.3	UPI 1	ンストール	. 99
	6.3.1	Any Platform インストールの流れ	99
	6.3.2	, インストール環境の整備	101
	6.3.3	OpenShiftインストーラー、コマンドラインツール、RHCOSの入手	105
	6.3.4	Ignition Configファイルの作成	107
	6.3.5	Bootstrapノード/Masterノードの作成	111
	6.3.6	Workerノード/Infraノードの作成	112
	6.3.7	クラスタ作成の確認	114
	6.3.8	クラスタ完成後の作業	116

7 Operator

7.1	Opero	ator概要	120
	7.1.1	Operator とは	. 120
	7.1.2	CustomResourceDefinitionとカスタムリソース	. 121
	7.1.3	Operatorの仕組み	. 122
	7.1.4	Operator 成熟度モデル	. 123
7.2	Open	Shiftにおける Operator	124
	7.2.1	Cluster Operator	. 124
	7.2.2	追加インストール可能なOperator	. 127
7.3	Operc	ator Framework	128
	7.3.1	Operator Frameworkとは	. 128
	7.3.2	Öperator SDK	. 129
	7.3.3	Operator Lifecycle Manager	. 138
	7.3.4	OperatorHub.io	. 139

7.4	Open	Shift上での GPU ノードの利用	141
	7.4.1	NVIDIA GPU Operatorとは	141
	7.4.2	Node Feature Discovery	142
	7.4.3	NVIDIA GPU DriverとNode Feature Discoveryの関係性	143
	7.4.4	GPU 搭載の Worker ノード追加	144
	7.4.5	cluster-wide entitlementの設定	144
	7.4.6	Node Feature Discoveryのインストール	148
	7.4.7	NVIDIA GPU Operatorのインストール	151
	7.4.8	GPUノードへのスケジューリング	154
	7.4.9	GPU ノードの監視	157

8 ネットワーク

8.1	CNI		160
8.2	Oper	Shift SDN	161
	8.2.1	Cluster Network Operator	164
	8.2.2	OpenShift SDNを構成するコンポーネント	166
	8.2.3	仮想スイッチ	167
	8.2.4	Podと仮想スイッチの接続	171
	8.2.5	ノードをまたいだPod間の通信	172
	8.2.6	PodからClusterIP Serviceへの通信	178
	8.2.7	外部から NodePort Service 経由の通信	180
	8.2.8	Podから外部への通信	180
8.3	Route	Lingress	181
	8.3.1	Routerによるロードバランス	182
	8.3.2	クラスタの外部から Router への接続	183
	8.3.3	Routerの高度な機能	187
	8.3.4	Ingress	191
	8.3.5	サードパーティ製のIngress Controller	192
8.4	Netw	orkPolicy	192
	8.4.1	すべての通信を拒否する NetworkPolicy 設定	193
	8.4.2	同じプロジェクト内の通信を許可する NetworkPolicy 設定	194
	8.4.3	指定したプロジェクトからの通信を許可する NetworkPolicy 設定	194
	8.4.4	指定したPodからの通信を許可するNetworkPolicy設定	196
	8.4.5	指定したIPアドレスからの通信を許可する NetworkPorlicy 設定	197
	8.4.6	新規プロジェクト作成時のNetworkPolicyの自動適用	197
8.5	Egres	S	199
	8.5.1	Egress IP	200
	8.5.2	Egressルーター	203
	8.5.3	Egress Firewall	220
8.6	Multu	JS	222
	8.6.1	Multus による複数ネットワークへの接続	222
	8.6.2	bridge プラグイン+ static プラグイン	223
	8.6.3	macvlan プラグイン+dhcp プラグイン	229

3.6.4	ipvlan プラグイン+ whereabouts プラグイン	233
8.6.5	Multusの課題と応用例	237
DVN-k	Kubernetes	238
8.7.1	OVNとは	238
3.7.2	OpenShiftにおける OVN-Kubernetes	240
3.7.3	OVSブリッジの構成	241
8.7.4	ロジカルネットワークのトポロジー	241
8.7.5	OVN-Kubernetes に関連するコマンド	242
8.7.6	OVN-KubernetesとOpenShift SDNの比較	244
8.7.7	OVN-Kubernetesの通信概要	244
	.6.4 .6.5)VN-I .7.1 .7.2 .7.3 .7.4 .7.5 .7.6 .7.7	.6.4 ipvlan プラグイン + whereabouts プラグイン .6.5 Multus の課題と応用例



9.1	コンラ	ーナ環境におけるストレージの基礎知識	. 256
	9.1.1	永続ストレージの必要性	256
	9.1.2	OpenShiftにおける永続ストレージの概念	257
9.2	永続ス	ストレージの選択肢	. 267
	9.2.1	ストレージの種類の違い	267
	9.2.2	ストレージプロビジョナーの違い	270
	9.2.3	構成形態の違い	272
	9.2.4	OpenShiftがサポートするストレージ	273
9.3	Red H	lat OpenShift Data Foundation	. 274
	9.3.1	Red Hat OpenShift Data Foundationの特徴	275

10 セキュリティ

10.1	コンテ	⁻ ナアプリケーションにおけるセキュリティ	280
	10.1.1	NIST SP 800-190とは	280
	10.1.2	OpenShiftのNIST SP 800-190対応状況	
10.2	イメー	-ジの対策	284
	10.2.1	イメージの脆弱性	
	10.2.2	イメージ設定の不具合	287
	10.2.3	埋め込まれたマルウェア	287
	10.2.4	埋め込まれた平文の秘密情報	287
	10.2.5	信頼できないイメージの利用	287
10.3	レジス	くトリの対策	291
	10.3.1	レジストリへのセキュアではない接続	
	10.3.2	レジストリ内の古いイメージ	
	10.3.3	不十分な認証・認可制限	292
10.4	オーケ	マストレーターの対策	293
	10.4.1	制限のない管理者アクセス	293
	10.4.2	不正アクセス	294
	10.4.3	コンテナ間ネットワークトラフィックの不十分な分離	295
	10.4.4	ワークロードの機微性レベルの混合	295

255

279

.....

	10.4.5	オーケストレーターのノードの信頼	295
10.5	コンテ	ナの対策	296
	10.5.1	ランタイムソフトウェア内の脆弱性	296
	10.5.2	コンテナからの無制限ネットワークアクセス	296
	10.5.3	セキュアでないコンテナランタイムの設定	297
	10.5.4	アプリケーションの脆弱性	297
	10.5.5	未承認コンテナ	298
10.6	ホスト	·OSの対策	298
	10.6.1	大きなアタックサーフェス	298
	10.6.2	共有カーネル	299
	10.6.3	ホストOSコンポーネントの脆弱性	299
	10.6.4	不適切なユーザーアクセス権	300
	10.6.5	ホストファイルシステムの改ざん	300

11 Day 2オペレーション

11.1	Mach	ine管理	302
	11.1.1	Machine 管理に関連するオブジェクト	. 302
	11.1.2	MachineConfigOperatorのサブコンポーネント	. 310
	11.1.3	スケーリング	. 311
	11.1.4	MachineSetの作成	. 317
	11.1.5	MachineSetを使用しないWorkerノードの追加	. 323
	11.1.6	MachineConfig による設定変更	. 324
11.2	クラス	.タモニタリング	329
	11.2.1	クラスタモニタリングスタック	. 329
	11.2.2	メトリクス	. 333
	11.2.3	アラート	. 333
	11.2.4	クラスタモニタリングの Web UI	. 334
	11.2.5	モニタリングスタックの設定変更	. 336
	11.2.6	PV (永続ボリューム) の設定	. 337
11.3	クラス	タロギング	338
	11.3.1	クラスタロギングスタック	. 338
	11.3.2	クラスタロギングの導入手順	. 340
	11.3.3	Kibana コンソールの使用	. 345
	11.3.4	クラスタロギングカスタムリソースの設定	. 348
	11.3.5	ログの外部への転送	. 351
	11.3.6	サイジング	. 353
11.4	メンテ	ナンス	354
	11.4.1	スケジューラー	. 354
	11.4.2	Prune	. 356
	11.4.3	ガベージコレクション	. 357
	11.4.4	バックアップ	. 358
11.5	クラス	タのアップグレード	360
	11.5.1	アップグレードの前提	. 360

	11.5.2	Webコンソールによるアップグレード	363
	11.5.3	CLIによるアップグレード	366
11.6	トラブ	ルシューティング	369
	11.6.1	よく発生するトラブルへの対処	369
	11.6.2	ログ	375
	11.6.3	イベント	376

12 OpenShift とマルチクラウド

12.1	マルチ	クラウド戦略	381
	12.1.1	ディザスターリカバリや可用性の向上	. 383
	12.1.2	コストの最適化	. 384
	12.1.3	スケーラビリティの確保	. 385
12.2	マルチ	クラウド計画の検討ポイント	386
	12.2.1	OpenShiftをマルチクラウドの基盤として考える	. 387
	12.2.2	 OpenShiftのマルチクラウドアプローチの限界	. 387
10.0	~ ~		001
12.3	Opens	htt のよび関連製品による美境力式の症条	391
12.3	OpenS 12.3.1	Initt あよび関連設品による美境力式の佐条 Red Hatのマルチクラウド関連製品ポートフォリオ	. 391 . 391
12.3	OpenS 12.3.1 12.3.2	Initt あよび関連製品による美妃力丸の提条 Red Hatのマルチクラウド関連製品ポートフォリオ Advanced Cluster Management for Kubernetes	. 391 . 391 . 393
12.3	OpenS 12.3.1 12.3.2 12.3.3	httt あよび関連製品による美境力丸の提条 Red Hatのマルチクラウド関連製品ポートフォリオ Advanced Cluster Management for Kubernetes Advanced Cluster Security for Kubernetes	. 391 . 391 . 393 . 394
12.3	OpenS 12.3.1 12.3.2 12.3.3 12.3.4	httt あよび関連製品による美境力丸の提条 Red Hatのマルチクラウド関連製品ポートフォリオ Advanced Cluster Management for Kubernetes Advanced Cluster Security for Kubernetes Red Hat Quay	. 391 . 391 . 393 . 394 . 395
12.3	OpenS 12.3.1 12.3.2 12.3.3 12.3.4 12.3.5	httt あよび関連製品による美境力丸の提条 Red Hatのマルチクラウド関連製品ポートフォリオ Advanced Cluster Management for Kubernetes Advanced Cluster Security for Kubernetes Red Hat Quay AMQ Streams	. 391 . 391 . 393 . 394 . 395 . 395

Part 3 アプリ実践編

5	<u> </u>	リケー	・ジョンのコンテナ化とモタテイス	401
	13.1	アプリ	リケーションのコンテナ化の基礎知識	402
		13.1.1	コンテナの特徴	402
		13.1.2	データの永続化	403
		13.1.3	設定情報の外部化	404
		13.1.4	サービスディスカバリーの実現	405
		13.1.5	ステートフルなアプリケーションの扱い	406
	13.2	コンテ	-ナデザインパターン	407
		13.2.1	単一関心の原則	408
		13.2.2	高可観測性の原則	408
		13.2.3	ライフサイクル適合の原則	409
		13.2.4	イメージ不変性の原則	
		13.2.5	プロセス廃棄性の原則	
		13.2.6	自己完結性の原則	

401

13.3 アプリケーションのモダナイズ 412 13.3.1 モダナイズのアプローチ 412 13.3.2 マイグレーションツール 416

14 コンテナイメージのビルド

14.1	ベース	イメージの取得	420
	14.1.1	Red Hat Ecosystem Catalog	. 420
	14.1.2	UBI (Universal Base Image)	. 423
14.2	アプリ	ケーションコンテナイメージのビルド	425
	14.2.1	コンテナイメージのビルド	. 425
	14.2.2	BuildConfig	. 426
	14.2.3	ImageStream	. 431
	14.2.4	BuildConfigとビルドツールとの組み合わせ	. 433

15 アプリケーションのデプロイ

15.1	マニフ	'ェストファイルによるアプリケーションのデプロイ	438
	15.1.1	ReplicaSet	. 438
	15.1.2	Deployment	. 439
	15.1.3	DeploymentConfig	. 440
	15.1.4	DaemonSet	. 442
	15.1.5	StatefulSet	. 443
	15.1.6	Job, CronJob	. 445
15.2	Opera	itorによるアプリケーションのデプロイ	445
	15.2.1	OperatorHub	. 445
	15.2.2	OperatorHubによるOperatorのインストール	. 446
	15.2.3	Operator インストールの流れ(GUI)	. 447
	15.2.4	Operator インストールの流れ(CLI)	. 450
	15.2.5	Operator でアプリケーションをデプロイする例	. 452
15.3	Helm	チャートによるアプリケーションのデプロイ	456
	15.3.1	Helm	. 456
	15.3.1 15.3.2	Helm CliによるHelmチャートのインストール	. 456 . 457
	15.3.1 15.3.2 15.3.3	Helm CIIによるHelmチャートのインストール GUIによるHelmチャートのインストール	. 456 . 457 . 458
	15.3.1 15.3.2 15.3.3 15.3.4	Helm CIIによるHelmチャートのインストール GUIによるHelmチャートのインストール カスタムHelmリポジトリの追加	. 456 . 457 . 458 . 460
15.4	15.3.1 15.3.2 15.3.3 15.3.4 Templ	Helm CliによるHelmチャートのインストール GUIによるHelmチャートのインストール カスタムHelmリポジトリの追加 ateによるアプリケーションのデプロイ	. 456 . 457 . 458 . 460 461
15.4	15.3.1 15.3.2 15.3.3 15.3.4 Templ 15.4.1	Helm CIIによるHelmチャートのインストール GUIによるHelmチャートのインストール カスタムHelmリポジトリの追加 ate によるアプリケーションのデプロイ Template	. 456 . 457 . 458 . 460 461 . 461
15.4	15.3.1 15.3.2 15.3.3 15.3.4 Templ 15.4.1 15.4.2	Helm CUIによるHelm チャートのインストール GUIによるHelm チャートのインストール カスタムHelm リポジトリの追加 ate によるアプリケーションのデプロイ Template Template によるアプリケーションのデプロイの流れ	. 456 . 457 . 458 . 460 461 . 461 . 463
15.4	15.3.1 15.3.2 15.3.3 15.3.4 Templ 15.4.1 15.4.2 アプリ	Helm CUによるHelm チャートのインストール GUIによるHelm チャートのインストール カスタムHelm リポジトリの追加 ate によるアプリケーションのデプロイ Template Template によるアプリケーションのデプロイの流れ ケーションのリリース戦略	. 456 . 457 . 458 . 460 461 . 461 . 463 464
15.4 15.5	15.3.1 15.3.2 15.3.3 15.3.4 Templ 15.4.1 15.4.2 アプリ 15.5.1	Helm CUIによるHelm チャートのインストール GUIによるHelm チャートのインストール カスタムHelm リポジトリの追加 ate によるアプリケーションのデプロイ Template Template によるアプリケーションのデプロイの流れ ケーションのリリース戦略 Route によるリクエストの振り分け	. 456 . 457 . 458 . 460 461 . 461 . 463 464 . 464
15.4 15.5	15.3.1 15.3.2 15.3.3 15.3.4 Templ 15.4.1 15.4.2 アプリ 15.5.1 15.5.1	Helm Cll による Helm チャートのインストール GUI による Helm チャートのインストール カスタム Helm リポジトリの追加 ate によるアプリケーションのデプロイ Template Template によるアプリケーションのデプロイの流れ ケーションのリリース戦略 Route によるリクエストの振り分け Blue-Green デプロイメント	. 456 . 457 . 458 . 460 461 . 461 . 463 464 . 464 . 465
15.4 15.5	15.3.1 15.3.2 15.3.3 15.3.4 Templ 15.4.1 15.4.2 77 J 15.5.1 15.5.2 15.5.3	Helm Cll による Helm チャートのインストール GUI による Helm チャートのインストール カスタム Helm リポジトリの追加 ate によるアプリケーションのデプロイ Template Template によるアプリケーションのデプロイの流れ ウーションのリリース戦略 Route によるリクエストの振り分け Blue-Green デプロイメント カナリアリリース	. 456 . 457 . 458 . 460 461 . 461 . 463 464 . 464 . 465 . 465

419

16 アプリケーション開発におけるライフサイクル

16.1	Open	Shiftにおけるアプリケーションの監視	470
16.2	アプリ	ケーションモニタリング	471
	16.2.1	ユーザー定義プロジェクトの監視の仕組み	. 471
	16.2.2	ユーザー定義プロジェクトの監視の有効化	. 473
	16.2.3	ユーザーへの権限付与	474
	16.2.4	アプリケーションとエクスポーターのデプロイ	. 475
	16.2.5	メトリクスの収集	. 479
	16.2.6	アラート設定	480
	16.2.7	通知設定	482
16.3	アプリ	ケーションロギング	483
	16.3.1	コンテナアプリケーションのロギング	484
	16.3.2	OpenShift Logging に収集されたログのデータ構造	485
	16.3.3	OpenShift Loggingを用いたログ検索の例	486
16.4	トレー	·シング	489
	16.4.1	Jaegerのアーキテクチャ	490
	16.4.2	Red Hat OpenShift Jaegerのインストール	491
	16.4.3	Jaegerインスタンスの起動	. 492
	16.4.4	jaeger-agentのサイドカーとしての起動	496
	16.4.5	サンプルアプリケーションでのトレーシング体験	498

17 クラウドネイティブ開発

クラウ	ドネイティブとは	504
マイク	ロサービス	505
17.2.1	マイクロサービスとは	. 505
17.2.2	ドメイン駆動設計 (DDD) とマイクロサービス	. 506
17.2.3	マイクロサービスにおけるデータの扱い	. 508
17.2.4	マイクロサービスにおける代表的なデザインパターン	. 509
サービ	`スメッシュ	515
17.3.1	サービスメッシュとは	. 515
17.3.2	lstio	. 517
17.3.3	Red Hat OpenShift Service Mesh	. 518
サーバ	ーレス	525
17.4.1	サーバーレスとは	. 525
17.4.2	Knative	. 525
17.4.3	Red Hat OpenShift Serverless	. 529
データ	連携のためのソフトウェア	539
17.5.1	Red Hat AMQ Streams	. 540
17.5.2	Red Hat Data Grid	. 546
17.5.3	Red Hat Single Sign-On	. 550
	クラウ マイク 17.2.1 17.2.2 17.2.3 17.2.4 サービ 17.3.1 17.3.2 17.3.3 サーバ 17.4.1 17.4.2 17.4.3 データ 17.5.1 17.5.2 17.5.3	クラウドネイティブとは マイクロサービス 17.2.1 マイクロサービスとは 17.2.2 ドメイン駆動設計 (DDD) とマイクロサービス 17.2.3 マイクロサービスにおけるデータの扱い 17.2.4 マイクロサービスにおける代表的なデザインパターン サービスメッシュ 17.3.1 サービスメッシュとは 17.3.2 Istio 17.3.3 Red Hat OpenShift Service Mesh サーバーレス 17.4.1 サーバーレスとは 17.4.2 Knative 17.4.3 Red Hat OpenShift Serverless データ連携のためのソフトウェア 17.5.1 Red Hat AMQ Streams 17.5.2 Red Hat Data Grid 17.5.3 Red Hat Single Sign-On

469

18 DevOps

18.1 18.2	Open Jenkir	Shiftを活用したDevOpsの概略 ns	556 560			
	18.2.1	Jenkins on OpenShiftの特徴	. 560			
	18.2.2	はしのとのjenkins	JOZ			
	18.2.3	JEIKINS (インジーンジース) マルチクラスタ対応	573			
	18.2.5	lenkinsのカスタマイズ				
	18.2.6	Jenkins agentのカスタマイズ	. 581			
18.3	Open	Shift Pipelines	582			
	18.3.1	Tekton	. 583			
	18.3.2	OpenShift Pipelines	. 583			
	18.3.3	Tekton Pipelines	. 585			
	18.3.4	Tekton Pipelinesを用いた実践的なパイプライン	. 595			
	18.3.5	Tekton Triggers	. 601			
	18.3.6	Tekton Triggersを用いたGitHubとの連携	. 602			
18.4	Open	Shift GitOps	607			
	18.4.1	GitOpsという考え方	. 607			
	18.4.2	OpenShift GitOps	. 609			
	18.4.3	Argo CDの基礎	. 612			
	18.4.4	CIツールとの組み合わせ	. 616			
索引			618			
著者紹介						

コラム目次

Red HatとOpenShiftに期待されていること	. 16
サブスクリプションの考え方	. 20
OpenShiftの学習コンテンツ	. 49
クラウドプロバイダーとは?	. 85
過去のバージョンの OpenShift クラスタを構築するには	107
エアギャップ環境への対応	117
Container Object Storage Interface (COSI)	268
Red Hat Advanced Cluster Security for Kubernetes	282
$\ln fra \mathcal{I} - \mathcal{K}$	317
OpenShift管理者が使用するツール	359
Red Hatへのカスタマーサポート問い合わせで送付する情報	376
Buildah	436
Red Hat CodeReady Workspaces	501
Quarkus	554



Chapter

1

OpenShiftの概要



Kubernetes^[1]をパッケージングし、商用サポートを提供する Kubernetes ディストリビューション^[2] は、2021年12月時点で61個あります。Red Hat OpenShift Container Platform(以下、OpenShift) は、世界で一番利用されている Kubernetesの商用ディストリビューションの1つです。

「OpenShiftは多数のOperatorの集合体」と言うことができます。OperatorとはKubernetesを拡張 する手法の1つであり、運用を自律化するためのカスタムコントローラーとCustomResource Definition (CRD)を指します(詳細は7.1節で解説します)。またOpenShiftは、クラスタアップグ レードをはじめ、モニタリングやロギング、その他クラスタを構成するOperator群、高機能GUIの OperatorHub^[3]から導入可能なサービスメッシュやCI/CD、ストレージクラスタなど、もともとの Kubernetesにはない機能が多数追加され、ありとあらゆる要素がOperatorによってインストールで き、自律的に運用できるというコンセプトを持っています。

本章では、クラウドネイティブ技術の市場動向を踏まえ、OpenShiftの特徴やコンセプト、利用形 態などについて解説します。

(クラウドネイティブ技術の市場動向

Cloud Native Computing Foundation (以下、CNCF) がKubernetesのホスティングを開始したの は2014年です。それから約7年の歳月が経過し、Kubernetes (2021年11月現在のバージョンは v1.22) はコンテナオーケストレーションツールのデファクトスタンダードになり、世界中にインパク トを与えたテクノロジーの1つと言えるほどに熱い視線を浴び続けています。CNCF Case Studies^[4] では、金融や通信、Eコマースなど、多種多様な業界・業種におけるクラウドネイティブ技術の活用 事例が2021年11月現在91件紹介されています。また、2000以上の組織がビジネス上の課題を解決 するためにOpenShiftを採用しています。

画期的な技術としてKubernetesが世界中から耳目を集めていることは事実ですが、Kubernetesは あくまでコンテナオーケストレーションツールです。コンテナオーケストレーションツールだけでは プロダクション環境を安定稼働させることは困難であり、Kubernetes単品で使用されることはほとん どありません。

CNCF TOC (Technical Oversight Committee) では、クラウドネイティブ技術を次のように説明しています ^[5]。

[4] https://www.cncf.io/case-studies/

Part 1 > 入門編

CHAPTER

1

1.1

) OpenShiftの概要

^{【1】} Kubernetesはしばしば「K8s」と略されることがあります。

^[2] Certified Kubernetes Distribution

https://landscape.cncf.io/card-mode?category=certified-kubernetes-distribution&grouping=category

^[3] https://operatorhub.io/

^[5] https://github.com/cncf/toc/blob/main/DEFINITION.md

Part

1

入門編

1

クラウドネイティブ技術は、パブリッククラウド、プライベートクラウド、ハイブリッドクラウド などの近代的でダイナミックな環境において、スケーラブルなアプリケーションを構築および実 行するための能力を組織にもたらします。このアプローチの代表例に、コンテナ、サービスメッ シュ、マイクロサービス、イミュータブルインフラストラクチャ、および宣言型APIがあります。 これらの手法により、回復性、管理力、および可観測性 [オブザーバビリティ]のある疎結合 システムが実現します。これらを堅牢な自動化と組み合わせることで、エンジニアはインパクト のある変更を最小限の労力で頻繁かつ予測どおりに行うことができます。(※[]は引用者追 記)

この説明にあるように、Kubernetesはクラウドネイティブ技術の1つであると解釈できます。ユー ザーによって、使用するテクノロジーや重視する要素、アプローチは異なりますが、共通して言える のは、Kubernetesだけで完結する話ではなく、あらゆるクラウドネイティブ技術を駆使して取り組む ことが必然となるということです。

1.1.1 CNCF Projects

CNCFは、Kubernetesをはじめとする多数のプロジェクト^[6]をホスティングし、プロジェクトの活性化を支援しています。

ホスティングしているプロジェクトは、基本的には成熟度別(成熟度が高い順)に「Graduated」「Incubating」「Sandbox」の3段階に分類されます(図1.1)。2021年11月時点では、KubernetesやPrometheusをはじめとするGraduatedプロジェクトは16個あります。Incubatingは25個、Sandboxは



図1.1 成熟度レベル(https://www.cncf.io/projects/)

^[6] CNCF Projects https://www.cncf.io/projects/



65個あります。また、「Archived」というカテゴリもあり、現時点ではコンテナランタイムの「rkt」が ここに分類されています。各成熟度に分類される際の指標など詳細については、CNCF公式ドキュ メント^[7]を参照してください。

「CNCF Cloud Native Landscape」(図1.2) は、未知の領域であるクラウドネイティブ技術を歩んで いく開発者や企業などのユーザーを支援するために提供されているリソースマップのような存在で す。各社が提供する Certified Kubernetes ディストリビューションや、CI/CD ツール、セキュリティ、 コンテナレジストリ、データベース、クラウドネイティブストレージなど、カテゴリ別に多種多様なプ ロジェクトが掲げられています。オープンソースソフトウェア (Open Source Software : OSS) はもち ろんのこと、ベンダーがサポートを提供する商用プロジェクトも記載されています。



図1.2 CNCF Cloud Native Landscape (https://landscape.cncf.io/)

Kubernetes に触れたことがない方は、「なぜ Kubernetes 周辺ツールが100 個近くあるんだ」と思う かもしれません。たしかに特段の意識をせずとも Kubernetes を使うことで、コンテナのスケーリング や宣言的な運用など一般的なコントロールはできます。

一方で、コンテナの実行速度や起動速度などのパフォーマンス面やセキュリティ面を考慮して内 部的に動作するコンテナランタイムを選択したり、Kubernetesクラスタを構成する環境のネットワー ク特性を考慮して適切なネットワークを選択したり、既存および今後使いたいストレージに対応させ たり、監視・オブザーバビリティ用のツールや、CI/CDツール選定など、目的や用途に合わせた自由

Part

1

入門編

1

選択ができるように多数のプロジェクトがラインナップされています。

また、Kubernetesの重要なコンセプトとして、多種多様なインターフェースを通じて疎結合できる ように開発されていることが挙げられます。ネットワークであれば CNI (Container Network Interface)、ストレージであれば CSI (Container Storage Interface)のように、インターフェースを通 じて疎結合できるように開発されています。このため、任意のネットワークやストレージなどを選択 して利用するのも Kubernetes であれば容易です。

コンテナオーケストレーション以外に必要になる代表的な要素を、クラウドネイティブを実現する ロードマップ「Cloud Native Trail Map」(図1.3)にならって2つ取り上げます。



図 1.3 Cloud Native Trail Map (https://github.com/cncf/landscape/blob/master/README.md#trail-map)



- Cloud Native Trail Map : 2. CI/CD
- Cloud Native Trail Map : 4. Observability and Analysis

Cloud Native Trail Map は、OSS やクラウドネイティブ技術活用のための道しるべとして、技術カテ ゴリ別にステップに分けて解説およびプロジェクトがマッピングされています。

CI/CDやオブザーバビリティは、従来からコンテナに限らず既存のアプリケーションでも実施して きたように、アプリケーションを各環境にデプロイしたり、継続的に状態を監視するなど、当たり前に 実施されてきたことでもあります。

1.1.2 CI/CD — 継続的インテグレーションと 継続的デリバリー

CI/CDは日本語では、継続的インテグレーション/継続的デリバリーと訳されます。CIは Continuous Integrationの略語で、CDはContinuous Delivery/Deploymentの略語です。CI/CDは、 ソフトウェアの変更を常にテストし、ステージング環境や本番環境に自動的に正しくリリースできる 状態を整えておくソフトウェア開発手法の1つです。

コンテナ/Kubernetesの文脈では、次のような使い方が想定されています。

アプリケーションコードの変更をトリガーにして、新しいコンテナイメージをビルドし、コンテナ レジストリに格納する。ユニットテストや統合テスト、脆弱性検査など必要なテストをパスした 後に任意のKubernetesクラスタにコンテナをデプロイする。

CI/CDを使えば、バグやセキュリティ的な問題をいち早く発見して解決することが可能になります。 さらに、品質を高め、変更に応じて自動的にリリースすることでアジリティのある開発ができるよう になります。

また、アプリケーションコードによらず、各種のKubernetesマニフェストをGitHubなどのGitリポ ジトリをSingle Source of Truth (唯一信頼できる情報源)として管理し、リポジトリ上のコード、構成 ファイルの変更をトリガーに、Kubernetesクラスタ上のコンテナの構成やクラスタ設定を変更し、シ ステム全体をあるべき状態 (Desired State) に近づける「GitOps」^[8]という手法もCI/CDのプラク ティスの1つとして注目を浴びています。

1つのコンテナのみを扱い、動作環境も限定的である場合は、CI/CDは必須ではないかもしれません。一方で、コンテナオーケストレーションツールであるKubernetesを用い、多数のコンテナを扱う 場合はどうでしょうか。Kubernetesクラスタ上で膨大な数のコンテナが動作し、盛んに新機能開発 が行われるクラウドネイティブな環境を想像してみてください。都度、人間の手によってコンテナイ

Part

1

入門編

П

メージを作成し、テストを行い、クラスタに合わせた設定を動的に行った後にリリースするのは、時 間的コストだけでなく、オペミス発生の温床にもなりかねないことは容易に想像できるでしょう。

CI/CDは、従来のソフトウェア開発においても実践できるに越したことはない開発手法でしたが、 Kubernetesをはじめとするクラウドネイティブ技術の活用においては「実践することが必須条件」と も言えます。

クラウドネイティブ以前から古くから利用者の多いJenkinsや、KubernetesネイティブなCIツール としてのTekton、CDツールとしてのArgo CDなどが注目を浴びています。他にもソフトウェア、 SaaSなど多数のプロジェクトがあるので、興味のある方はCNCF Cloud Native Landscapeの技術カ テゴリ「Continuous Integration & Delivery」を眺めてみてください。

0bservability and Analysis — オブザーバビリティとアナリシス

オブザーバビリティ(Observability:可観測性)は、「システムがいかに観測されやすい状態であ るか」を意味します。

分散システムでは、すべてのマシンが完全な状態で動作していることは期待できず、常にどこかが 壊れている可能性があります。これは、コンテナ/Kubernetesを活用したクラウドネイティブな環境 でも同じことが言えます。従来の監視では、ピンポイントにアプリケーションやネットワーク、データ ベース、ストレージなどを指定して問題が起きていないかを把握し、アラートを発行することで、「壊 れてしまっている対象物」を認識することは可能です。

一方で常にどこかが壊れている可能性のある分散システムでは、従来の監視だけでなく「なぜ壊れてしまっているのか」「何がどう劣化しているのか」を理解できることが求められます。サービス利用者であるエンドユーザーから見て、「対象サービスが利用できない状態」はサービスダウンを意味しますが、対象サービスを構成する分散システムの一部が壊れていることは直接関係がなく、サービスが不満なくいつもどおり利用できることが重要です。そのため、「何が原因で部分的に壊れているのかを観測できるシステムのオブザーバビリティ」が非常に大切になります^[9]。

Cloud Native Trail Mapの「4. Observability and Analysis」では、オブザーバビリティを高めるた め、モニタリング用途に「Prometheus」を、ロギング用途に「Fluentd」を、トレーシング用途に 「Jaeger」を使うことを検討するよう記載されています。これら3つのプロジェクトは、CNCF Cloud Native Landscapeの技術カテゴリ「Observability and Analysis」に分類されており、成熟度レベル (Maturity Level)はGraduatedに位置し、開発コミュニティが活発で、非常に多くのユーザーに使用 されています。興味のある方は、各プロジェクトのWebサイトやコードリポジトリなど確認してみてく

^{【9】} オブザーバビリティと監視の違いや、Kubernetesをメトリクス中心のアプローチによってオブザーバビリティを高めるヒントについては、 「Kubernetesで実践するクラウドネイティブDevOps」(John Arundel、Justin Domingus 著、オライリー・ジャパン、2020年) などの 解説書を参考にしてください。

ださい。

Kubernetesをはじめとするクラウドネイティブ技術は、DevOpsと可観測性に非常に相性が良いと されています。通常は企業組織にDevOpsを推進するなど大きな変革には膨大な時間を要しますが、 このような組織変革でも、現在のKubernetesの盛り上がりのムーブメントに乗って、クラウドネイ ティブ技術を積極的に取り入れ、CI/CDという開発手法やオブザーバビリティを導入していけば、テ クノロジードリブンなアプローチで道を切り拓いていくことも期待できます。



KubernetesとOpenShiftの違い

OpenShift は、Kubernetesの商用ディストリビューションです。ここでは、アップストリーム版の Kubernetes^[10]に対して、OpenShiftが提供する付加価値について説明します。

OpenShiftは、CNCFの適合性プログラムに準拠^[11]しています。これによって、アップストリーム 版のKubernetesとAPIレベルでの互換性があり、ユーザーから見た相互運用性が保証されていま す。そのため、OpenShiftを「エンタープライズ版Kubernetes」として利用する企業も多く存在しま す。たとえば、Kubernetesの機能の範囲内であれば、ocコマンド(OpenShift固有のCLI)ではなく、 kubectlコマンドを利用して他のKubernetes運用との互換性を担保できます。

OpenShiftは、各クラウドプロバイダー上で提供されるマネージドOpenShiftサービスだけでなく、 自身で物理サーバーや仮想マシン、プライベートクラウド、パブリッククラウド、Edgeなど、あらゆ るプラットフォーム上に構築し利用することができます。エンタープライズ向けの製品としては当然 ですが、サブスクリプションを契約することで、製品サポートを受けられます^[12]。

OpenShiftは、エンタープライズ利用を想定したKubernetesとして多種多様な機能群を備え、 Kubernetesを利用しやすく、運用しやすくするための配慮が行われています。OpenShiftがカバーす る機能の全体イメージを図1.4に示します。図中の「Kubernetes (orchestration)」の部分が、アップ ストリーム版のKubernetesがカバーしている範囲です。それに対し、OpenShiftは、Kubernetesクラ スタを構成するノード(物理、仮想環境などにOSが入ったノード)のレイヤーを含む、Kubernetes クラスタ上で動作する多種多様なコンポーネントも含む製品となっています。

^{【10】} CNCF が配布する Kubernetes を「アップストリーム (上流)版の Kubernetes」と呼びます。

^[11] CNCF Certified Kubernetes - Distribution (OpenShift)

https://landscape.cncf.io/card-mode?category=certified-kubernetes-distribution&grouping=category&selected=redhat-open-shift

^{【12】}提供形態やサブスクリプションについては、本章の「1.3 OpenShift利用形態とサブスクリプション」を参照してください。

Part

1

入門編



図1.4 OpenShiftがカバーする機能の全体イメージ

たとえば、クラスタモニタリングやロギング、クラスタアップグレードなど、アプリケーションや Kubernetesクラスタの運用を下支えする機能群や、CI/CD、ServiceMesh、ServerlessなどDevOpsを はじめとする開発生産性の向上や、マイクロサービスのオブザーバビリティを向上させる仕組みなど を必要に応じて容易に追加できるように構成されています。

他にも、内部コンテナレジストリや、コンテナアプリケーションへのトラフィックを管理する Router、充実したGUI (Graphical User Interface)が標準機能として用意されているため、運用者が 別のツールを選択したり、導入に時間を割いたりする必要はありません。

OpenShiftはアップストリーム版のKubernetesを包含しており、Masterノード(Control Plane)で 稼働する「etcd」の可用性(Quorum)を担保するために3台、Workerノードはノードレベルの障害 の影響を回避するために2台以上のクラスタ構成が推奨されています^[13]。

また、Operatorというテクノロジーを活用することによって、OpenShiftクラスタのインストール、 構成変更、アップグレード、機能追加などを容易にしています。OpenShiftの根幹を成す部分と言っ ても過言ではないのが、Operatorによる自律運用です。Kubernetes APIを拡張することによって、 Kubernetesが管理できないリソースに対しても、Kubernetesによって宣言的な管理を可能にします。 Operatorによって、クラスタがあるべき状態を維持するように管理され、クラスタ上への機能追加お

^[13] 基本的には、Master ノード (Control Plane) はK8s クラスタおよび各リソースの管理を行い、Worker ノードでは実働アプリケーションが 動作します。詳細については、第6章の「6.2.2 ノード」を参照してください。

よび追加された機能のライフサイクル管理もサポートされます [14]。

さらに、1台のクラスタのみならず企業で利用される複数のクラスタの管理や、マルチクラウド上 に存在する多数のクラスタなどを統合管理するための機能や、高度なセキュリティ機能、大規模利用 を想定したレジストリなど、用途に合わせたオプションも用意されています。

これらの特徴は一例にすぎませんが、企業がKubernetesを利用するにあたって、Kubernetesだけ では不足している運用機能や開発生産性を高められる機能、組織など複数人での使い勝手を便利に する権限設定などの機能をアップストリーム版のKubernetesに追加し、OpenShiftとして提供してい ます。



Part 1 > 入門編

CHAPTER 1) OpenShiftの概要

OpenShiftのサポート期間

OpenShift は、最新リリースを含む直近3マイナーバージョンまでがサポート対象です。た とえば「4.7」が最新リリースである場合は「4.5」までがサポートされ、「4.4」はEOL (End Of Life:サポート終了)になります。



OpenShift では、アップストリーム版の Kubernetes と同様に、Master ノードと Worker ノードに よってクラスタを構成します。さらに、OpenShift では Infra ノードを追加し、モニタリングサービス やコンテナイメージレジストリやルーティングをサポートする機能などを Master ノードや Worker ノードではなく、Infra ノードに委譲させることもできます。

OpenShiftクラスタの設計や構成コンポーネント、システム要件などについては、第6章の「6.2 クラスタの設計」を参照してください。

1.2.2 クラスタの構築

ユーザーはクラウドプロバイダーが提供するマネージド OpenShift サービスを利用できますが、自 身でオンプレミス環境やクラウドリソースを使用して OpenShiftを構築することもできます。構築方 法には、IPIと UPIの2つの方法があります。

IPI (Installer-Provisioned Infrastructure)

IPIは、インストーラーを用いて動的に仮想マシンやロードバランサー、ストレージなどを準備し、 OpenShiftクラスタをインストールする方法です。IPIの詳細およびIPIによるインストール手順につ

^{【14】} Operator の詳細については、第7章「Operator」を参照してください。

Part

1

入門編

1

いては、第2章の「2.1 IPIインストール」を参照してください。

UPI (User-Provisioned Infrastructure)

UPIは、事前に仮想マシンなどのリソースを準備し、OpenShiftクラスタをインストールする方法で す。オンプレ環境やインターネット接続に制限のある環境や、既存のIT資産を活用する場合に使わ れます。UPIの詳細およびUPIによるインストール手順については、第6章の「6.3 UPIインストール」 を参照してください。

それぞれのインストール方法が対応するクラウドプロバイダーやプラットフォームは、以下のとおりです(最新の対応環境は公式サイト^[15]で確認してください)。

IPI

- Amazon Web Services (AWS)
- Google Cloud Platform (GCP)
- Microsoft Azure (Azure)
- Red Hat OpenStack Platform (RHOSP) version 13 and 16
- Red Hat Virtualization (RHV)
- VMware vSphere
- VMware Cloud (VMC) on AWS
- ベアメタル

UPI

- AWS
- GCP
- Azure
- RHOSP
- RHV
- VMware vSphere
- VMware Cloud (VMC) on AWS
- IBM Z or LinuxONE
- IBM Power Systems
- ベアメタル



1.2.3 拡張コンポーネント

OpenShiftのMasterノードには、Kubernetesのクラスタ制御サービスが起動しています。Master ノードで稼働するKubernetesのサービスとして以下のものがあります。

- Kubernetes API (kube-apiserver)
- etcd (etcd-member)
- Kubernetes Controller Manager (kube-controller-manager)
- Kubernetes Scheduler (kube-scheduler)

さらにOpenShiftでは、Kubernetes コンポーネントを補う拡張コンポーネントが提供されています。 たとえば、Kubernetesの中核となるKubernetes API(kube-apiserver)に対して、機能を拡張した OpenShift API(openshift-apiserver)がOperator 経由で展開されています。OpenShift APIはあくま でKubernetes APIを拡張しているだけにすぎず、アップストリーム版のKubernetesと同様に 「kubect1」コマンドを利用できるように構成されています。しかし、OpenShift専用の管理コマンド の「oc」を利用することで、拡張機能が容易に利用できるようにしています。たとえば、「oc」コマン ドを利用してコンテナアプリケーションのビルドやクラスタ管理操作を直観的に行うことができます。

OpenShiftのWorkerノードでは、Kubernetesのコアサービス以外にもノード監視を行うための Prometheusのエージェント(Node Exporter)やクラスタ間のネットワークを経由するCNIプラグイ ンなどが存在します。これらは、OpenShiftを構成するうえで欠かせない機能であるため、Master ノード上でも同様のコンテナが稼働しています。他にもOpenShiftでは、「MachineSet」というリソー スオブジェクトがノードを制御しています。これは、KubernetesにおけるPodとReplicaSetの関係と 似た機能です。つまり、アプリケーションのワークロードに応じてMachineSetオブジェクトのreplicas フィールドを変更することで、動的なノードの追加や縮退が制御できます。ユーザーがWorkerノー ド上のサービスの制御を積極的に行うのではなく、OpenShiftがアプリケーションワークロードに応 じてWorkerノードの管理を自動的に行う仕組みに委ねることで、インフラの管理コストの削減に役 立ちます。



OpenShift上のモニタリングは、クラスタモニタリングとアプリケーションモニタリングの2つに分かれており、Prometheusを使用しています。クラスタの稼働メトリクス情報は「openshiftmonitoring」というプロジェクト内のOperatorが管理する、Prometheus (Prometheus Operator)に よって取得されています。また、取得したメトリクス情報はGrafanaで可視化されます。可視化に関

Part

1

入門編

1

しては、Grafana以外にもOpenShiftのコンソール(GUI)上から確認することも可能です。

このコンテナのメトリクス取得には、「kube-state-metrics」が利用されています。kube-statemetricsは、Kubernetes APIを監視し、OpenShiftが提供する Podや Deployment、DaemonSet などの オブジェクトに対して、メトリクス情報を取得します。また、Prometheus上の CPU やメモリ使用率な どのリソースをメトリクスに変換し公開することで、HPA(Horizontal Pod Autoscaling)にも活用さ れます。特別な設定を行わなくても OpenShift上で監視できます。ただし、固有アプリケーションの メトリクスを監視する場合は「kube-state-metrics」では取得ができないため、別途設定が必要です。

Prometheusの運用そのものはPrometheus Operator が行いますが、取得するメトリクスの指定は ユーザー自身が決めて対応します。また、Prometheus Operator では、Alertmanagerも管理できます。 Alertmanagerを使うことによって、Prometheusが取得したメトリクスをトリガーに、Webhookを呼び 出してアラートを上げることも可能です。

クラスタモニタリングの機能紹介については第11章の「11.2 クラスタモニタリング」を、アプリ ケーション (ユーザーワークロード)のモニタリングについては第16章の「16.2 アプリケーションモ ニタリング」を参照してください。

1.2.5 ロギング

OpenShiftでは、ログ監視のため、EFK (Elasticsearch、Fluentd、Kibana) スタックを活用した Cluster Logging (Red Hat OpenShift Logging) の仕組みがあります。

ログコレクターとしてFluentdが利用され、各Workerノード上にDaemonSetとして展開されます。 これによってすべてのノードのログを収集し、Elasticsearchに蓄積して分析を行います。また、開発 者やクラスタ管理者が集計されたデータを分析するため、Kibanaによって可視化しています。ただ し、Elasticsearchは比較的多くのリソースを必要とするため、クラスタインストールの初期状態では、 Cluster Logging は稼働しないようになっています。必要に応じて管理者が、Cluster Logging Operator と Elasticsearch Operator の双方を準備し、EFK スタックを構築します。

Cluster Logging Operator は、ログの収集(Collection)、保存(LogStore)、可視化(Visualization) といった、EFKスタックの設定を行うインターフェースです。Cluster Logging Operatorを使えば、 Elasticsearch と Kibana は OpenShift との認証連携とアクセス制御ができます。「一般の開発者は所属 するプロジェクトのログのみ参照できる」といった制限も設定できます。

Cluster Logging は比較的処理に負荷のかかるコンポーネントであるため、アプリケーションワーク ロードが動作する Worker ノードと切り離してスケジューリングしたほうが安定稼働が期待できます。 この場合、Worker ノードの負荷を軽減するために EFK スタックを Infra ノードに導入することもでき ます。本書では扱いませんが、OpenShift のサブスクリプション対象が Worker ノードであることから、 Infra ノードにインフラ機能として構築する選択も考えられます。



Cluster Loggingの概要やインストール方法については、第11章の「11.2 クラスタモニタリング」を 参照してください。アプリケーションのロギング方法や、Kibana ダッシュボードを用いたログ検索方 法などについては、第16章の「16.2 アプリケーションモニタリング」で解説しています。



Kubernetes におけるコンテナ間のネットワークは、Container Network Interface (CNI) を実装す るネットワークプラグインに依存します。CNIは、コンテナのネットワークインターフェースを設定す るための仕様です。

ネットワークプラグインがオーバーレイネットワークを作ることで、各 Worker ノードに配置される コンテナ同士が疎通できます。OpenShift が同梱する CNI プラグインは、OpenShift SDN、OVN-Kubernetes、Kuryr の3つです。この他にも様々なネットワークプラグインをサポートしています。 OpenShift 4.9のデフォルトでは「OpenShift SDN」という Open vSwitch (OVS) ベースの CNI プラ グインを使用し、オーバーレイネットワークを構築します。

こうした、ネットワークプラグインもCluster Network Operator (CNO) によって管理されていま す。CNOは、OpenShift SDN等のCNIプラグインを管理したり、ユーザーによる設定変更の内容が 悪影響を及ぼさないか事前に検査したりします。

OpenShiftにおけるネットワークの詳細については、第8章「ネットワーク」を参照してください。

1.2.7 ストレージ

Kubernetes上でデータベースのようなステートフルなアプリケーションを扱うためには、データを 保存するための永続化を実現する方法とストレージが必要です。ビルド済みコンテナイメージをレジ ストリに格納しておくことで、常に同一のコンテナを動作させることができますが、コンテナは揮発 性のあるプロセスであり、起動・再起動、再作成を短いサイクルで繰り返します。つまり、コンテナ内 に永続データを保存することはできません。アプリケーションに関連するデータだけでなく、前述の クラスタモニタリングやロギング、内部レジストリなど、様々な用途で必要となる永続データが存在 します。

コンテナのストレージインターフェースを設定するための仕様として、Container Storage Interface (CSI) があります。プラグインのCSI driver を用いれば、Kubernetes とバックエンドストレージが通信 できるようになります。

また、Red Hat OpenShift Data Foundation (ODF) と呼ばれる OpenShift クラスタのための専用 SDS (Software-defined Storage) が用意されています。ODF は、Rookをコアテクノロジーとして活

Part

1

入門編

1

用しており、クラスタ内部にストレージを構成するため基盤を問わず、Operatorによってインストー ルからストレージクラスタの構成や変更などの作業を制御できます。さらに、OpenShiftに組み込ま れたPrometheusによって利用状況やメトリクスの取得も可能です。

OpenShiftにおけるストレージの詳細については、第9章「ストレージ」を参照してください。

1.2.8 クラスタアップグレード

Kubernetesは、アップデートサイクルが短く、約4か月ごとに新しいバージョンがリリースされています。ユーザーはこのアップデートサイクルに追従する必要があります。

OpenShiftは、OpenShiftクラスタのアップグレードを支援する仕組みであるOver-The-Air (OTA) アップグレード機能を用意しています。マネージドKubernetesを利用しているかのようにコンソール 上あるいはCLIからクラスタバージョンをアップグレードできます。ここでも内部的には、Operator が活躍しています。Cluster Version Operator (CVO)というOperatorが、現行クラスタで動作する 多数のOperator 群を管理しており、クラスタのアップグレードパス(アップグレード可能なバージョ ンの確認、複数バージョンをまたぐ場合の経路など)を確認し、アップグレードを実行する役割を 担っています。

OpenShiftのアップグレードの仕組みや手順については、第11章の「11.5 クラスタのアップグレード」を参照してください。

1.2.9 CI/CD

OpenShiftは、Kubernetesの利用において必要不可欠であるCI/CDを実現するための機能として「OpenShift Pipelines」と「OpenShift GitOps」を提供しています。これらはOperatorを通じて OpenShiftに組み込まれます。

OpenShift Pipelines は、内部的には「Tekton」を使用しています。Tektonは、CIパイプラインの構成・設定をKubernetesリソース(マニフェスト)として管理ができ、KubernetesネイティブなCIツールと呼ばれるOSSです。OpenShift Pipelinesでは、CIパイプラインの開発・利用をサポートする機能 群をコミュニティ版 Tekton に加えて統合しています。また CI/CD のシーンでよく使われてきた Jenkins については、Red Hat がビルドした Jenkins イメージを引き続き提供しています。

OpenShift GitOpsは、内部的には「Argo CD」を使用しています。Argo CDは、CD(継続的デリ バリー)に特化した CDパイプラインツールであり、Kubernetesネイティブな CD ツールと呼ばれる OSS です。GitOpsは、Gitリポジトリを Single Source of Truthと捉えることで、リポジトリ上のコード や設定変更をトリガーにしてクラスタに対して変更を反映させるアプローチを採用しています。これ Part 1 》入門編 CHAPTER 1 OpenShiftの概要

> はKubernetesのControl Loop/リコンシリエーションループ(Reconciliation Loop)と同様に、宣 言的な構成管理と自動化によってあるべき状態(Desired State)の維持を実現します。OpenShift GitOpsもOperatorを通じてArgo CDのインストール、構成管理が行われます。

> OpenShift Pipelines やGitOpsの概要や使い方、そしてJenkinsとの比較については、第18章 「DevOps」を参照してください。

1.2.10 その他

これまで見てきたもの以外にも、エンタープライズ用途でKubernetesを最大限活用できるよう、多数の機能がOpenShiftには組み込まれています。たとえば、セキュリティ、コンテナイメージのビルド、アプリケーションのデプロイ、クラウドネイティブ開発、マルチクラウド対応などです。以降の章では、それぞれ詳しく解説していきます。



皆さんは Kubernetes とうまく付き合えていますか?

ご存じのとおり、Kubernetesを利用することで、コンテナのスケーリングやセルフヒーリング、宣言的記述(マニフェスト)による制御を行うことができますよね。一方で、コンテナを動作させるまでの準備や、コンテナ動作後の運用、クラスタ自体の運用など、Kubernetesだけではカバーできない部分があります。一例を挙げると、コンテナイメージのビルド、コンテナイメージの格納・管理、Kubernetesクラスタへのデプロイ、起動したコンテナのモニタリング、コンテナ間(連携するアプリケーションプロセス間)のトレーシング、さらにKubernetesクラスタ自体の管理や、セキュアな状態の確保などです。必ず、Kubernetes周辺の技術要素について検討し、何かしらの手だてを整えなければなりません。

そこで、筆者が把握している範囲で、Red HatおよびOpenShiftに対するユーザーからの期待の声の代表例を以下に挙げてみます。

- KubernetesのRed Hatによる商用サポート
- OpenShift に組み込まれる OSS テクノロジー選定における先進性と安定性の両立
- Kubernetes へのアプリケーションデプロイの容易性
- Kubernetes上でのアプリケーション開発の柔軟性
- クラスタインストールの容易性、俊敏性
- クラスタアップグレード、他運用面の負担軽減
- ハイブリッド、マルチクラウド対応
- セキュリティ

ユーザーによって求める部分は異なりますが、共通して言えるのは「Kubernetesの運用をいかに手間な く行えるか、そしてKubernetesの能力をいかに即座に利用できるか」に対するRed HatおよびOpenShift の動向への期待と思われます。テクノロジーの進化とそれを使いこなすことは終わりなき戦いとも言えるか もしれませんが、いちKubernetes好きとしては、技術を負債の要因とせず、最大限活用することを楽しん でいただきたいと思います。



ユーザーは、2種類の利用形態を選ぶことができます。

- マネージドOpenShiftサービス (Hosted OpenShift): クラウドプロバイダーが提供したものを利用する (図1.5)
- セルフマネージドOpenShift (Self-Hosted OpenShift):オンプレミスなどに自身で構築して利用する



図1.5 マネージドOpenShiftサービス

1.3.1 マネージドOpenShiftサービス

様々なクラウドプロバイダーがマネージドOpenShiftを提供し、運用を行っています。具体的には、 クラスタの管理やノードのセキュリティ確保、パッチの適用管理、クラスタアップグレード、メトリク スログの収集など、クラスタ維持に伴う運用作業を行います。ユーザーは、各クラウドプロバイダー のマネージドOpenShiftサービスのプランを契約することで使用できます。以下のようなプランがあ ります。



- Red Hat OpenShift Dedicated (OSD)
- Red Hat OpenShift Service on AWS (ROSA)
- Azure Red Hat OpenShift (ARO)
- Red Hat OpenShift on IBM Cloud (ROKS)

各種マネージドサービスにおいて、Red Hatのサポートチームも各社と連携したサポートを提供します(表1.1)。

表 1.1 マネージド OpenShift サービス提供主体

提供サービス	Red Hat OpenShift Dedicated (OSD on AWS/GCP)	Red Hat OpenShift Service on AWS (ROSA)	Azure Red Hat OpenShift (ARO)	Red Hat OpenShift on IBM Cloud
Infrastructure	AWS or Google Cloud	AWS	Azure	IBM Cloud
Billed by	Red Hat/AWS or Google Cloud	AWS	Microsoft	IBM
Managed by	Red Hat	AWS & Red Hat	Microsoft & Red Hat	IBM
Supported by	Red Hat	AWS & Red Hat	Microsoft & Red Hat	IBM & Red Hat

※執筆時点(2021年11月)の情報です。最新情報は以下を確認してください。 https://cloud.redhat.com/products

1.3.2 セルフマネージド Open Shift と サブスクリプション

ユーザーは、オンプレミスやパブリッククラウドリソースを使用して OpenShiftを構築できます。この利用形態を「セルフマネージド」と呼び、すべての運用をユーザー自身で行います。

Red Hat 社からサブスクリプションを購入することで、OpenShiftを使用し、サポートを受けられま す。以下の3種類のサブスクリプションが用意されています(表1.2)。

- Red Hat OpenShift Platform Plus ^[16]
- Red Hat OpenShift Container Platform
- Red Hat OpenShift Kubernetes Engine

^[16] https://www.openshift.com/products/platform-plus
		サブスクリプション		
同梱製品	Red Hat OpenShift Platform Plus	Red Hat OpenShift Container Platform	Red Hat OpenShift Kubernetes Engine	備考
Red Hat OpenShift Container Platform	0	0	○ (機能制限有)	Kubernetes 運用および開発を支 援する拡張機能を提供
Red Hat Advanced Cluster Management for Kubernetes	0	×	×	オンプレミス、パブリッククラウ ドを問わず、複数のKubernetes クラスタの統合管理機能を提供
Red Hat Advanced Cluster Security for Kubernetes	0	×	×	ビルド、デプロイ、実行中などあ らゆるタイミングで適用可能なセ キュリティポリシーを多数提供
Red Hat Quay	0	×	×	セキュアかつグローバルに利用可 能な大規模レジストリ機能を提供

表1.2 セルフマネージドOpenShiftの機能の違い





COLUMN

サブスクリプションの考え方

OpenShift クラスタは、Master ノード、Worker ノード、Infra ノード(オプション)を独立して用意する ことで構成されます。OpenShift のサブスクリプションは、基本的にはノードに搭載された CPU コア数に 対して課金されます。CPU コア数をカウントする対象ノードは、Worker ノードだけです。Master ノードや Infra ノードは考慮する必要がありません。図 **1.6** に例を挙げます。



図1.6 必要サブスクリプションコア数の例

例外的にMasterの役割とInfraの役割をまとめて1つのノードとして構成することがあります。さらに Edge など特殊なケースでは、Master/Worker などすべての機能を1台のノードに集約する構成例もあり ます。基本的な考え方は通常構成時と同様に、Workerノードを構成するノードのCPUコア数に相当するサ ブスクリプション数が必要となります。

また、物理サーバー上にクラスタを構成する場合は、CPUソケット数に相当するサブスクリプションの選択も可能です。サブスクリプションやサイジングの最新情報および詳細については、以下のガイドを参照してください。

• OpenShift sizing and subscription guide for enterprise Kubernetes https://www.redhat.com/en/resources/openshift-subscription-sizing-guide





オンプレミス環境やクラウド環境にOpenShiftをインストールする方法は、IPIとUPIの2種類があ ります。本章ではより簡単にOCPクラスタを作成できるIPIインストールについて解説します。IPIイ ンストールとUPIインストールの違いについては、第6章の「6.1 プラットフォームとインストール方 法の選択」を参照してください。



本節では、OpenShift 4のインストール方法の一例として、AWS 環境にIPIで OpenShift 4.9をイン ストールする手順を説明します。ここで紹介する構成は、実際に本番環境でも使えるよう、Master ノード・Worker ノードともに冗長性があり、複数管理者による管理を意識した構成になっています。



本構成の場合、クラウド費用は1日あたり5,000円程度かかります。そのため、個人で手軽 にOpenShiftを試してみるには少々オーバースペックであると感じるかもしれません。より手 軽にOpenShiftを試してみたい方は、本章の「2.3 Red Hat CodeReady Containers」を参 照してください。

インストールは、次の手順で進めていきます。

- 1. アカウント作成
 - a. Red Hat Customer Portal アカウントの作成
 - **b.** AWSアカウントの作成
- 2. AWSアカウントの設定
 - a. IAMユーザーの作成と権限付与
 - b. Amazon Route 53 でパブリックホストゾーンを作成
 - **c.** サービスクォータの確認
- 3. クラスタ作成前の準備
 - a. Amazon EC2で踏み台サーバーを作成
 - **b.** AWS CLI のインストール
 - c. openshift-installのインストール
 - d. Pull-Secretの取得
 - e. SSH 鍵ペアの作成
 - f. クラスタ構成ファイル (install-config.yaml) の作成
- 4. クラスタ作成
 - a. openshift-installでクラスタを作成

入門編

2

- 5. 補足 クラスタ削除
 - a. openshift-install でクラスタを削除

2.1.1 アカウント作成

OpenShift 4のインストールを行うには、Red Hat Customer PortalのアカウントとAWSアカウント が必要です。アカウントを持っていない方は、以下を参考にアカウントを作成してください。

Red Hat Customer Portalのアカウント作成

Red Hat Customer Portalのアカウントは、OpenShiftのインストーラーやOpenShift CLI、Pull-Secret (クレデンシャル情報) などを取得するときに使用します。次のURLからアカウントを作成してください。

 Red Hat Customer Portal https://access.redhat.com/

■AWSのアカウント作成

今回はAWS上にOpenShiftをインストールするため、AWSリソースを作成するためのAWSアカウントが必要です。次のサイトを参考にアカウントを作成してください。

 AWSアカウント作成の流れ https://aws.amazon.com/jp/register-flow/

2.1.2 AWSアカウントの設定

OpenShiftをインストールする前には、AWSアカウントで次の準備が必要です。

■IAMユーザーの作成と権限付与

インストール作業用にAdministratorAccessポリシーがアタッチされたIAMユーザーを作成しま す。AWSアカウント作成時にrootユーザーが作成されるため、このrootユーザーを使ってインス トール作業を行うこともできます。しかし、rootユーザーには強い権限が与えられているため、root ユーザーをそのまま使い続けるのはセキュリティの観点から好ましくありません。必要最小限の権限 を付与したインストール作業専用のIAMユーザーを作成して、そのIAMユーザーでインストール作 業を行うことが推奨されています。



IAMユーザーの作成手順は、次のとおりです。

1. AWSマネジメントコンソールの「IAM ダッシュボード」にログインします (図2.1)。左側のメ ニューで [アクセス管理] → [ユーザー] と遷移してから [ユーザーを追加] ボタンをクリックし ます。

aws # #-22 Q #-22.	特徴、プログ、およびドキュメントなどを放業 [オプション+5] 🖉 ダローバル・ 🦛	demo 🔻
Identity and Access × Management (IAM)	WM > 3-4-	0
Q. IAM の依潔	ユーザー (2) 物理 MAI ユーザーは、アカウントで AVIS を操作するために見解的な認証機能を許つアイデンティティです。	
ダッシュボード	Q. ユーザー名またはアクセスキーでユーザーを検索 < 1 > (④	
▼ アクセス管理 雪のユーサーグループ	ユーザー名 ▽ グループ ▽ 最後のアクティビティ ▽ MFA ▽ パスワードが作成されてから証拠した期間 ▽ アクティブなキーが作成されてから証拠した期間 ▽	
	販示するリソースがありません	
ポリシー		
ID プロパイダ		
アカウント設定		
▼ アクセスレポート アクセスアナライザー アーカイブルール		
アナライザー		
設定		
認証情報レポート		
組織アクティビティ		
サービスコントロールポリシー (SCP)		

図2.1 IAMダッシュボード:ユーザーを追加

- 2. ユーザー情報を入力していきます (図2.2)。
 - ▶ ユーザー名:任意 (ここでは [ocp-installer] と入力)
 - ▶ **アクセスの**種類:[プログラムによるアクセス]をチェック

[次のステップ:アクセス権限] ボタンをクリックします。

aws, サービス ▼	Q サービス、標準、マーケットプレイスの製品、ドキュメントを検索 [Option+S]	
	ユーザーを追加	5
	ユーザー詳細の設定 同じアクセスの種類とアクセス種類を使用して複数のユーザーを一変に追加できます。詳細はこちら (ユーザー名・ opp restaller O 別のユーザーの追加	
	AWS アクセスの電频を選択 これらのユーザーから AMR ビアクセス7も方法を選択します。アクセスキーと自動生成パスワードは約のステップで提供されています。 評制はこく アクセスの機能 ジ 70 ウンスムネククセス AMR/ CLL SXX を20開発リールの アクセスキー D と シークレットアクセス	56
	キー を初始にします。 → WWS マネジメントコンソールへのアクセス ユーザーに XWN マネジメントコンソールへのサインインを許可するための パスワード を有効にします。	
	*必須 キャンセル 次のステップ: アクセス#	IR)

図2.2 IAMダッシュボード:「ユーザー名」と「アクセスの種類」を設定

入門編

2

 [アクセス許可の設定]では[既存のポリシーを直接アタッチ]を選択します(図2.3)。その下に 表示される既存のポリシー一覧の中から [AdministratorAccess] をチェックします。[次のステッ プ:タグ] ボタンをクリックします。

aws, サービス ▼	Q サービス、僭能、マーケットプレイスの製品、ドキュメント	を検索 [Option+S]
	ユーザーを追加	1 2 3 4 5
	▼ アクセス許可の設定	
	2 ユーザーをグループに追 2 アクセス種類を既存の コーザーからコピー	既存のポリシーを直接ア タッチ
	ポリシーの作成	0
	ポリシーのフィルタ ~ Q 検索	665 件の結果を表示中
	ポリシー名 マ	タイプ 次として使用
	AdministratorAccess	ジョブ機能 なし
	AdministratorAccess-Amplify	AWS による管理 なし
	AdministratorAccess-AWSElasticBeanstalk	AWSによる管理 なし
	AlexaForBusinessDeviceSetup	AWS による管理 なし
	AlexaForBusinessFulAccess	AWS による管理 なし
	AlexaForBusinessGatewayExecution	AWS による管理 なし
	AlexaForBusinessLifesizeDelegatedAccessPolicy	AWS による管理 なし
	AlexaForBusinessPolyDelegatedAccessPolicy	AWS による管理 なし
	▶ アクセス権限の境界の設定	
		キャンセル 戻る 次のステップ: タグ

図2.3 IAM ダッシュボード:アクセス許可の設定

4. [タグの追加(オプション)] では任意のタグを設定できます。この例では何も設定せずに、[次の ステップ:確認] ボタンをクリックします(図2.4)。

aws サ−ビス ▼	Q サービス、機能	慾、マーケットプレイスの製品、ドキュメントを検索 (Option-	+\$]
	ユーザーを追加		1 2 3 4 5
	タグの追加 (オプション)		
	IAM タグは、ユーザー に追加できるキ できます。タグを使用して、この ユー	ーと値のペアです。タグには、Eメールアドレスなどのユーザ ザー のアクセスを整理、追踪、制御できます。 詳細はこちら	一情報を含めるか、役職などの説明文とすることが
	*-	値 (オプション)	HEIRR
	新しいキーを追加		
	さらに 50 個のタグを追加できます。		
			TT2 C/0 200 (XOX7777:812

図2.4 IAMダッシュボード:IAMタグの設定



5. 確認を求める画面が表示されたら内容を確認し、[ユーザーの作成] ボタンをクリックします(図 2.5)。

aws サービス▼	٩	サービス、鍵症、マーケットプレイスの製品、ドキュメントを検索 [Option+S]	
	ユーザーを	自加	1 2 3 4 5
	確認		
	選択内容を確認します。	ユーザーを作成した後で、自動生成パスワードとアクセスキーを確認してダウンロー	-ドできます。
	ユーザー詳細		
	AW/0 1	ユーザー名 ocp-installer	
	アクセ	ス権限の境界 アクセス権限の境界が設定されていません	
	マクセス施岡の編	25	
	次のポリシー例は、上	∝ 2のユーザーにアタッチされます。	
	タイプ	名前	
	管理ポリシー	AdministratorAccess	
	タグ		
	追加されたタグはあり	ません。	
		*	ヤンセル 戻る ユーザーの作成

図2.5 IAM ダッシュボード:確認画面

 ユーザーの作成が完了すると、アクセスキーIDとシークレットアクセスキーが表示されます(図
 2.6)。これ以降、シークレットアクセスキーを確認することはできないため、必ずメモをとるか CSVファイルをダウンロードしてください。

aws #−ピス ▼	Q サービス、鑚総、マーケットプレイスの製品、	ドキュメントを検索 [Option+S]	
	ユーザーを追加	1 2 3 4	6
	成功 以下に示すユーザーを正常に作成しました。ユーザーのセキ リールへのサイソイン手続きをメールマユーザーに送着する 会でを、たじ、私いび経営機構のいでやられできます。 ANG マネジメントコンソールへのアクセス優を持つユーザー できます。	a.リティ際証明報を補額してダウンロードできます。AWS マネジメン こともできます。今回が、これらの部証明報をダウンロードできる要求 は ¹ https://7007307366448.signin.aws.amazon.com/console ₃ でサイン	トコン 長の機 シイン
	≛.csv のダウンロード		
	ユーザー	アクセスキー ID シークレットアクセ	エスキー
	▶ Ø ocp-installer	AKIA2JDDPHTIHFSAPWQI 包	
			閉じる

図2.6 IAMダッシュボード:ユーザー追加の成功画面

入門編

2

Amazon Route 53でパブリックホストゾーンを作成

Amazon Route 53 (以下、Route 53) サービスでパブリックホストゾーンを作成します。ここで、クラスタのベースドメインをRoute 53 に登録します。

Creating a public hosted zone

https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/CreatingHostedZone.html

使用するドメインを取得する方法はいくつかあります。今回は、会社や個人で所有しているドメインから OCP 用のサブドメインを作成し、そのサブドメインの管理を先ほど作成した IAM ユーザーが 操作できる Route 53 に委任する方法を紹介します。親ドメインは別のアカウントの Route 53 で管理 されていても、別のレジストラ(例: Google Domains)で管理されていても問題はありません。

ここでは例として、親ドメイン aoshima.me から OCP 用に ocpdemo.aoshima.me というサブドメインを作成し、これをパブリックホストゾーンに登録します。

まずは、Route 53のダッシュボードから [ホストゾーン] へ遷移し、[ホストゾーンの作成] をクリックします (図2.7)。

• Route 53 ダッシュボード

https://console.aws.amazon.com/route53/v2/home#Dashboard

aws	サ−ビス ▼	Q サービス、標態、マーケットプレイスの製品、ドキュメントを検索 [Option+5]	
≡⊚	新しい Route 53 コンソールの紹介 使いやすさにこだわり、Route 53 のコンソールを設計から見直しました。 <u>ご意見を</u> クしてください。	<u>お聞かせください</u> 。当社では、皆様のフィードバックに基づいて、ユーザーエクスペリエ	× ンスを戦勢的に改善しています。お楽しみに1以前のコンソールの使用をご希望の場合は、 <u>ごちら</u> をクリッ
	Route 53 〉 ホストゾーン		
	ホストゾーン(0) Automatic モードは最適なフィルク結果に最適化された現在の検索動作です。モードを変更す	るには、(設定)に移動します。	C 詳細を表示 編集 削除 ホストゾーンの作成
	Q プロバティまたは値でホストゾーンをフィルタリングする		< 1 > ©
	ドメイン名 マ タイプ	▽ 作成者 ▽ レコード数	▽ 説明 マ ホストゾーン ID マ
		ホストゾーンがありません	
		このアカウント用に作成されたホストゾーンはありません。	
		ポストソーンの作家	

図2.7 Route 53のダッシュボードで [ホストゾーンの作成] を選択

「ホストゾーンの作成」画面に遷移したら、必要な情報を入力していきます(図2.8)。ここでは、次のように入力しました。

- ドメイン名:使用するドメイン名(ここでは「ocpdemo.aoshima.me」と入力)
- 説明:ドメインの説明を入力(未入力も可)
- **タイプ**: [パブリックホストゾーン] を選択
- タグ:任意



図2.8 「ホストゾーンの作成」 画面

Part 1 > 入門編

CHAPTER 2 環境構築

必要な情報を入力してから[ホストゾーンの作成]ボタンをクリックします。ホストゾーンの作成が 完了すると次の画面に遷移し、ここでNSレコードが確認できます(図2.9)。このNSレコードを親ド メイン側で設定します。

② cycles saakbaus 052 812/hd 278 01/hd 2010 >>> >>> >>> >>> >>>> >>>> >>>>> >>>>> >>>>> >>>>>>>>>>>>>>>>>>>>>>>>>>>>	vs		Q、サービス、機能、マーケットプレイスの製品、ドキュメントを検索 [Option+S]	
Note 53 〉 ホストゾーン 〉 opdema.aoshima.me ゾーンを照解する レコードを見たしたをなえたする グェブログの配立 ・ ホストゾーンの詳細 ・ ホストゾーンの詳細 ・ ホストゾーンの詳細 ・ ホストゾーンを見照する ・ ホストゾーンを見照する レコード(2) Info Automits モード目標目なったりまたに思った時間を発展された記念の情報報告です。モードを支充するとは、(102)に移動します。 ・ ロードを目面 ・ ワーンを見照する ・ ロードを目面 レコード(2) Info Automits モード目標目なったりまたに思った時間を発展された記念の情報目やです。モードを支充するとは、(102)に移動します。 ・ ロードを目面 ・ ロードを目面 ・ レコードを目 ・ ク	⊘ •q ≍	pdemo.aoshima.me が正常に作成されました。 れで、ホストゾーンにレコードを作成して、Route 53 でドメインのト		
ocpdemo.aoshima.me hek ゾーンを照解する レコードを見入まする クェンロクの認定 ・ ホストゾーンの詳細 ホストゾーンの詳細 ホストゾーンの書類 レコード(1) Dissset 著名 ホストゾーンのクタグ(0) レコード(2) Ins エストゾーンの学校 (2) プロパゲィ 老太は思さいたここのであったいたこのであったいのファイルをマンボート レコードを見合いたしたこのであったいのファイルをマンボート (2) プロパゲィ 老太は思さいしコードをマイルタリンググター (2) (2) プロパゲックスカルーチャー 新潟県、 (4) (2) プロパゲックスカルーチャー (2) (2) (2) プロパゲックスカルーチャー (2) (2) (2) プロパゲックスカルーチャー (2) (2) (2) (2) (2) (2) (2) (2) (2) (2) (2) (2) (2) (2) (2) (2) (2) (2) (2) (2) (2) (Ro	ute 53 > ホストゾーン > ocpdemo.aoshima.me		
・れストゾーンの評細 ホストゾーンの評細 レコードロ DMSSEC 電気 ホストゾーンのクグ(0) レコードロ DMSSEC 電気 ホストゾーンのクグ(0) レコードロ C レコード電源 ゾーンファイルをインボート レコードロ DMSSEC 電気 ホストゾーンのクグ(0) レコードロ DMSSEC 電気 ホストゾーンのクグ(0) レコードロ DMSSEC 電気 ホストゾーンのクグ(0) レコードロ DMSSEC 電気 アメーン・マーンのクグ(0) ロ ロード電気 ダー・マーン・ロ ター・フィン・マ ロ レコード電 ダー・マ ター・フィン・マ エイリアスマ ロ レコード電 ダー・マ ロ ロード電気 マ ロ レコード電 ダー・マ ロ ロ ロ ロ ロ レコード電 ダー・マ ロ ロ ロ ロ ロ ロ ロ ダー・マ ロ	0	cpdemo.aoshima.me Info		ソーンを削除する レコードをテストする クエリログの設定
Lコード(3) DM35KC電査 ホストゾーンのタグ(8) Lコード(3) DM35KC電査 ホストゾーンのタグ(8) Lコード(3) DM35KC電査 ホストゾーンのタグ(8) Lコード(3) DM35KC電査 たされた気化の構築的やす。モードを変更するには、成功に早楽します。 Q プロパケィネカム型ボンククリングする Q プロパケィネカム型ボンククリングする Q プロパケィネカム型ボンククリングする Q プロパケィネカム型ボンククリングする Q プロパケィネカム型ボンククリン Q プロパケィネカム型ボンククリン Q プロパケィネカム型ボンククリングする Q プロパケィネカム型ボンククリングする Q プロパケィネカム型ボンククリング Q プロパケィネカム型ボンククリング Q プログレード Q プロパケィネカム型ボンククリング Q プログレッ Q グログレッ Q プログレッ Q グログレッ Q グログ Q グログ		▶ ホストゾーンの詳細		ホストゾーンを編集
レコード名 マ タ マ ルーア マ 観知 マ 観/トラブィックのルーディング発 マ ccpdeme.asoh N5 シンプル ・ ccpdeme.asoh		レコード(2) DMSSEC署名 ホストゾーンのタグ(0) レコード(2) units Commute モードは事項なフィルラ結果に最近された現在の検索動作です。1 の、プロパイナーまたは雪にレコードをフィルタリングする	- ドを変更するには、彼均にが発します。 - アイクマーマー「ルーティン」、マー・スイリアスマー」	C レコードを削除 ゾーンファイルをインボート レコードを作成 (11) (0)
レコード名 マ タ マ ルーテ マ 奥胆 マ 側/トラフィククのルーディング先 マ copdemo.aodh NS シンブル ・ no-1147.awardrs-15.org. no-1731.awardrs-25.org. no-453.awardrs-25.org. マ no-1147.awardrs-15.org. no-453.awardrs-25.org. マ copdemo.aodh NS シンブル ・ no-1147.awardrs-15.org. no-453.awardrs-25.org. ref ref <th></th> <th>C JUIG FALLAGE COL-FESTAD SSSS 6</th> <th></th> <th></th>		C JUIG FALLAGE COL-FESTAD SSSS 6		
CopdemoLassh N5 シンプル ・		□ レコード名 ▽ タ ▽ ルーテ ▽ ま	2別 ▽ 値/トラフィックのルーティング先	
 ocpdemo.aoshi SOA シンブル - ns-1147.awsdns-15.org. awsdns-hostmaster.amazon.com. 1 7200 900 1209600 86400 		□ ocpdemo.aoshl NS シンプル -	ns-1147.awsdns-15.org. ns-1731.awsdns-24.co.uk. ns-679.awsdns-20.net. ns-460.awsdns-57.com.	
		ocpdemo.aoshi SOA シンプル -	ns-1147.awsdns-15.org. awsdns-hostmaster.amazon.com. 1 7200 900 1209600 86400	

図2.9 パブリックホストゾーンの作成が正常完了した画面

これで、パブリックホストゾーンの作成は完了です。

1

入門編

2

■サービスクォータの確認

AWS アカウントは、作成可能なコンポーネント数の上限がサービスごとに設定されています。 OpenShift Container Platform(以下、OCP)は多数のAWSコンポーネントを使用するため、クラス タの数やサイズによっては制限に引っかかってインストールに失敗することがあります。そのため、ク ラスタの作成前には現在のサービスクォータの設定とクラスタが必要とするコンポーネント数を確 認しておく必要があります。

必要なコンポーネント数がサービスクォータの設定値を上回っていた場合は、AWS コンソールか らクォータの引き上げリクエストを送ります。また、サービスクォータはリージョンごとに存在するの で、OCPをインストールするリージョンそれぞれについて設定を確認するようにしておきます。

OCPクラスタのインストールおよび実行機能に影響を与える可能性のある AWS コンポーネントの 制限は、以下のオンラインドキュメントにまとめられています。

• AWSアカウントの制限

https://access.redhat.com/documentation/ja-jp/openshift_container_platform/4.9/html/ installing/installation-aws-limits_installing-aws-account

サービスクォータの設定状況は、AWSコンソールの「Service Quotasのダッシュボード」で確認できます(図2.10)。

• Service Quotasのダッシュボード(東京リージョン)

https://ap-northeast-1.console.aws.amazon.com/servicequotas/home

aws #−ビス ▼	Q、サービス、機能、マーケットプレイスの製品、ドキュメントを検索 [Option+S]	🖸 🔶 ocp-demo 🔻
Service Quotas × ダッシュポード	ダッシュボード	ダッシュボードカードを変更する
AwvSのサービス クォータリクエスト履歴 ▼ 組織	Amazon Athena	Amazon Elastic Block Store (Amazon EBS) 合計クォーク: 37
クォータリクエストテンプレート	Amazon Elastic Compute Cloud (Amazon EC2) 金計やフーク: 84	▲ Amazon Virtual Private Cloud (Amazon VPC) 合計クォーク: 23
	AWS CloudFormation 合計クォータ: 24 AWS Key Management Service (AWS KMS) 合計クォーク: 52	図 AWS Lambda 合計クォータ: 22
	保留中のサービスクォータリクエスト (0) 最近解決したサービスクォ く 1 >	ータのリクエスト (0) < 1 >
	クォータの名称 マ ステータス マ 最終更新日 ◆ クォータの名称 マ リクエストが見つかりません <t< th=""><th> マ ステータス ▼ 最終更新日 マ リクエストが見つかりません </th></t<>	 マ ステータス ▼ 最終更新日 マ リクエストが見つかりません

図2.10 Service Quotasのダッシュボード

なお、本書執筆時点(2021年11月)では、デフォルトのサービスクォータ設定でも、デフォルトサ イズのOCPクラスタ1台(Masterノード3台+Workerノード3台)をインストールできます。



2.1.3 クラスタ作成前の準備

■Amazon EC2 で踏み台サーバーを作成

本章では、Amazon Elastic Compute Cloud (以下、Amazon EC2) で踏み台サーバーを作成し、踏 み台サーバーから openshift-install (OpenShiftインストーラー)を実行する方法を紹介します (図2.11)。

openshift-installは、LinuxとmacOSで実行可能なため、踏み台サーバーを使わずに手元の PCにopenshift-installをダウンロードして実行することも可能です(図2.12)。ただし、複数人で クラスタを管理する場合は、踏み台サーバーを1つ用意しておくと、次のような便利な点があります。

- 1. openshift-installの実行環境を管理者がそれぞれ用意する必要がない。
- 2. インストール設定ファイルやtfstateファイル^[1]、インストールログファイルなどを開発者間で 共有する仕組みを別途用意しなくてもいい。

本章では踏み台サーバーがある前提で説明をしますが、踏み台サーバーではなく自分の手元のPC から操作する場合は、今後踏み台サーバーで行うセットアップ作業を手元のPC上で実施してください。



図2.11 本章で紹介する構成(特に管理者が複数人いる場合は、こちらがおすすめ)



図2.12 踏み台サーバーを使わない構成(個人で管理する場合はこの構成も可能)

^{【1】} インストール時に生成される terraform.tfstate ファイル (2.1.5項) です。

1

入門編

2

本書の検証では、AMIは「Red Hat Enterprise Linux 8 (HVM), SSD Volume Type - ami-0bccc42bb a4dedac1 (64ビットx86)」を使用し、インスタンスは「t2.micro」、そして、SSH接続できるようなセキュリティグループを設定しました。



踏み台サーバーは、必ずしも Red Hat Enterprise Linux である必要はありません。 openshift-installが実行可能であればいいので、他のLinux やmacOS でも代替可能です。 また、インターネットに疎通できて AWSの APIを利用できるのであれば、必ずしも AWS上に 作成する必要もありません。

■AWS CLIのインストール

踏み台サーバーにAWS Command Line Interface (CLI)をインストールします。踏み台サーバーがRHEL以外の場合は次のサイトを参照してください。

 AWS CLIバージョン2のインストール、更新、アンインストール https://docs.aws.amazon.com/ja_jp/cli/latest/userguide/install-cliv2.html

unzipコマンドが必要なため、インストールされていない場合は事前にインストールしてください (sudo yum install unzipを実行します)。

```
$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
$ unzip awscliv2.zip
$ sudo ./aws/install
You can now run: /usr/local/bin/aws --version
```

AWS CLIがインストールできたことをaws --versionコマンドで確認します。以下のようにAWS CLIのバージョンが表示されれば成功です。

\$ aws --version
aws-cli/2.3.4 Python/3.8.8 Linux/4.18.0-305.el8.x86_64 exe/x86_64.rhel.8 prompt/off

次に、先ほど作成したIAMユーザー (ocp-installer) の情報をaws configure コマンドで登録 します。

▶ 入力例



認証情報ファイルが正常に認識されているかどうかは、次のコマンドで確認することができます。

```
$ aws sts get-caller-identity
{
    "UserId": "AIDASN7WUQMYMXTYPHVM3",
    "Account": "167484097328",
    "Arn": "arn:aws:iam::167484097328:user/ocp-installer"
}
```

■openshift-installのインストール

openshift-install (OpenShiftインストーラー) は、下記のRed Hat OpenShift Cluster Manager サイトからダウンロードしてください (図2.13)。openshift-installのバージョンによって、インス トールされる OpenShiftクラスタのバージョンが決まります。

Red Hat OpenShift Cluster Manager サイトで配布されている openshift-install は最新バージョンなので、旧バージョンの OpenShift クラスタをインストールしたい場合は、対応する openshift-installを下記 URL からダウンロードしてください。

 最新安定バージョンのOpenShift 4をインストールしたい場合: Red Hat OpenShift Cluster Managerサイト

https://cloud.redhat.com/openshift/install/aws/installer-provisioned

 特定のバージョンのOpenShift 4をインストールしたい場合: openshift-installバイナリー覧 https://mirror.openshift.com/pub/openshift-v4/clients/ocp/

😑 🍓 Red Hat	¢ 0 🔤	◎治 青島 👻
OpenShift Cluster Clu Manager Ins	Inster > Create > Ansator Web Services > Instate-provisioned Infrastructure sstall OpenShift on AWS with Installer-provisioned Infrastructure	
Clusters		
Subscriptions	1 What you need to get started	
Overview	OpenShift installer Deveload and extract the install program for your operating system and place the file in the	
Support Cases	directory where you will store the installation configuration files. Note: The OpenShift install program is only available for Linux and macOS at this time.	
Cluster Manager Feedback	Linux	
Red Hat Marketplace		
Documentation 🗹	Puil secret Download or copy your puil secret. The install program will prompt you for your puil secret during installation.	
	Download pull secret	
	Command line interface	
	Download the OpenShift command-line tools and add them to your PATE.	
	Linux Download command-line tools	
	When the installer is complete you will use the console URL and credentials for accessing your new cluster. A kakesense if ig file will also be generated for you to use with the or CLI tools you downloaded.	
	2 Follow the documentation to configure your AWS account and run the installer	ଡ

図2.13 Red Hat OpenShift Cluster Managerサイト

入門編

2

今回は例としてhttps://mirror.openshift.com/pub/openshift-v4/clients/ocp/から執筆時 (2021年11月)の最新安定バージョンの「4.9.4」をインストールします。

まずは、openshift-installをダウンロードします。

\$ curl -0 https://mirror.openshift.com/pub/openshift-v4/clients/ocp/stable/openshift-install-linu
\$ x.tar.gz

圧縮ファイルを展開します。

\$ tar xvf openshift-install-linux.tar.gz

パスが通っているディレクトリ、たとえば「/usr/local/bin」などに openshift-installを移動し ます。

\$ sudo mv ./openshift-install /usr/local/bin

最後に、パスが通っているかどうかを確認します。次のような出力が返ってくれば、正常にパスが 通っています。

```
$ openshift-install version
openshift-install 4.9.5
built from commit 8223216bdcef5de56b52240ab7160ca909a9e56c
release image quay.io/openshift-release-dev/ocp-release@sha256:386f4e08c48d01e0c73d294a88bb64fac3 
284d1d16a5b8938deb3b8699825a88
release architecture amd64
```

■ Pull-Secretの取得

Pull-Secret (クレデンシャル情報) は、openshift-install でクラスタをインストールするときに 入力を要求されます。事前に取得し、安全な場所で保管しておいてください。Pull-Secret は、open shift-install と同じく、Red Hat OpenShift Cluster Manager サイトで取得できます。

 Red Hat OpenShift Cluster Managerサイト https://cloud.redhat.com/openshift/install/aws/installer-provisioned

■SSH鍵ペア作成

MasterノードにSSH接続するためのSSH鍵ペアを作成します。クラスタ作成時(次ページの「構成ファイルの作成」)に、ここで作成したSSH鍵の公開鍵を登録します。ここで登録したSSH鍵は、

Part 1 > 入門編 CHAPTER 2 環境構築

> インストール失敗時にBootstrapノード^[2]にSSH接続して原因調査をしたり、クラスタインストール 後も障害が発生したときにMasterノードにSSH接続して対応する場合などに使用することがあるた め、クラスタインストール後も大切に保管してください。

\$ ssh-keygen -t ed25519 -N '<パスフレーズ>' -f <パス>/<ファイル名>

▶ 登録例

\$ ssh-keygen -t ed25519 -N 'passphrase' -f ~/.ssh/id_ed25519

次に、パスフレーズ入力をスキップできるように、秘密鍵とパスフレーズをssh-agentに登録しま す。パスフレーズを空にした場合は、この作業は必要ありません。

まずは、ssh-agent プロセスをバックグラウンドタスクとして起動します。

\$ eval "\$(ssh-agent -s)"

そして、ssh-agentに公開鍵を登録します。

\$ ssh-add <パス>/<ファイル名>

▶ 登録例

\$ ssh-add ~/.ssh/id_ed25519
Enter passphrase for /home/ec2-user/.ssh/id_ed25519:
Identity added: /home/ec2-user/.ssh/id_ed25519 (ec2-user@ip-172-31-2-15.ap-northeast-1.compute.
internal)

鍵が登録されているかどうかを確認します。

\$ ssh-add -L

ssh-ed25519 AAAAC3NzaC11ZDI1NTE5AAAAIBSGpbxzp7b/jRyJCMzQDB7kZyH19LBziyV1R9pKYfj/ ec2-user@ip-172 -31-2-15.ap-northeast-1.compute.internal

■構成ファイルの作成

構成ファイル (install-config.yaml) を作成します。ここで、どのような構成でクラスタをインス トールするか決めます。

最初に、作業用ディレクトリを作成し、移動します(作業ディレクトリ名は任意でかまいません)。

^{【2】} Bootstrap ノードとは、クラスタインストール時に一時的に作成される VM (Virtual Machine:仮想マシン)のことです。

\$ mkdir openshift
\$ cd openshift

まずは、openshift-install create install-configを実行してinstall-config.yamlファイルを作成します。次の情報を対話形式で入力するので、事前に用意しておきましょう。

- SSH Public Key: <「2.1.3 クラスタ作成前の準備」で作成したSSH公開鍵のパス>
- Platform : AWS
- Region:<クラスタを作成するリージョン>(対応リージョンは以下) https://access.redhat.com/documentation/ja-jp/openshift_container_platform/4.9/ht ml/installing/installation-aws-regions_installing-aws-account
- Base Domain: < [2.1.2 AWSアカウントの設定」でRoute53に登録したドメイン>
- Cluster Name:<クラスタ名>
- Pull Secret: <「2.1.3 クラスタ作成前の準備」で取得した Pull-Secret>

▶ install-config.yamlファイルを作成

```
$ openshift-install create install-config
? SSH Public Key /home/ec2-user/.ssh/id_ed25519.pub
? Platform aws
INFO Credentials loaded from the "default" profile in file "/home/ec2-user/.aws/credentials"
? Region ap-northeast-1
? Base Domain ocpdemo.aoshima.me
? Cluster Name democluster
? Pull Secret [? for help]
```

次のような install-config.yaml ファイルが作成されます。

```
$ cat ./install-config.yaml
apiVersion: v1
baseDomain: ocpdemo.aoshima.me
compute:
- architecture: amd64
hyperthreading: Enabled
name: worker
platform: {}
replicas: 3
controlPlane:
architecture: amd64
hyperthreading: Enabled
name: master
platform: {}
replicas: 3
metadata:
creationTimestamp: null
```

Part

1

入門編

2

```
name: democluster
networking:
clusterNetwork:
- cidr: 10.128.0.0/14
hostPrefix: 23
machineNetwork:
- cidr: 10.0.0.0/16
networkType: OpenShiftSDN
serviceNetwork:
- 172.30.0.0/16
platform:
aws:
region: ap-northeast-1
publish: External
pullSecret: '{"auths":{"cloud.openshift.com":{"auth":"XXXXXXXXXX","email":"XXXXXXXXXX"}}}'
sshKey: |
ssh-ed25519 AAAAC3NzaC11ZDI1NTE5AAAAINZhOov1nefrj5jo4NdPzqHu5hg0kAHNotcN04PwE27B ec2-user@ip-172- 🔊
31-32-111.ap-northeast-1.compute.internal
```

このinstall-config.yamlをそのまま使ってクラスタを作成することもできますが、今回は少し構成を変更してみます。

デフォルトではm4.xlargeのMasterノード3台とm4.largeのWorkerノード3台の構成となっていますが、WorkerノードのEC2インスタンスタイプをもう少し大きなm5.xlargeに変更します。次のように、install-config.yamlの「compute」部分(Workerノードの構成部分)を書き換えます。

```
...略...
compute:
- architecture: amd64
hyperthreading: Enabled
name: worker
platform:
   aws:
      type: m5.xlarge
replicas: 3
...略...
```

Part 1 > 入門編

CHAPTER 2 環境構築

openshift-install create cluster (OpenShiftクラスタを構築するコマンド)を実行すると、作 成した install-config.yaml は自動的に削除されます。そのため、インストール前にバックアップを とっておきます。

\$ cp install-config.yaml install-config.yaml.bak

入門編

2



■openshift-installでクラスタを作成

クラスタを構築するコマンドを実行します。完了まで30~40分ほどかかります。

<pre>\$ openshift-install create cluster INFO Credentials loaded from the "default" profile in file "/home/ec2-user/.aws/credentials" INFO Consuming Install Config from target directory</pre>
INFO Creating infrastructure resources
INFO Waiting up to 20m0s for the Kubernetes API at https://api.democluster.ocpdemo.aoshima.me:644 🔁 3
INFO API v1.22.0-rc.0+a44d0f0 up
INFO Waiting up to 30m0s for bootstrapping to complete
INFO Destroying the bootstrap resources
INFO Waiting up to 40m0s for the cluster at https://api.democluster.ocpdemo.aoshima.me:6443 to in 🕑
itialize
INFO Waiting up to 10m0s for the openshift-console route to be created
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export KUBECONFIG=/home 🕑
/ec2-user/openshift/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-console.apps.democluster.oc 🕑
pdemo.aoshima.me
INFO Login to the console with user: "kubeadmin", and password: "IoAzZ-5QrAY-LJtqr-3KqGe" INFO Time elapsed: 40m47s



以上で、インストール作業は完了です。AWS上には図2.14のようなリソースが構成されます^[3]。

図2.14 インストール直後のAWS構成概要(Amazon EC2とELBのみ)

【3】 EC2とELB (Elastic Load Balancing) 以外にも多くのリソースが作成されますが、図2.14では省略しています。

意図したバージョンのクラスタが正常にインストールができているかどうかは、カスタムリソース のclusterversionを見ることでも確認することができます。詳細は、次章の「3.2.3 ocコマンドの 使い方」を参照してください。

また、詳細なインストールログは、コマンド実行ディレクトリの.openshift_install.logファイル に保存されます。インストールに失敗したときは、このファイルを調査してください。

インストール直後はkubeadminというクラスタ管理者権限を持つユーザーが作成されています。 このkubeadminユーザーのパスワードやWebコンソールのURLは、標準出力に記載されているので メモしておきましょう^{【4】}。ログイン方法の詳細については次章で紹介します。

2.1.5 補足 クラスタ削除

■ openshift-install でクラスタを削除

クラスタの削除には、インストール時に生成された terraform.tfstateファイルが必要です。イン ストール作業をしたディレクトリに terraform.tfstateが作成されるため、同じディレクトリで次の コマンドを実行してください。

\$ cd \$HOME/openshift
\$ openshift-install destroy cluster



Part 1 > 入門編

CHAPTER 2 環境構築

サーバー証明書の設置

OpenShiftには、コンテナアプリケーションをクラスタ外部に公開する「Router」という機能が備わっています。ユーザーがOpenShiftにデプロイしたコンテナアプリケーションだけではなく、 OpenShiftのWebコンソールなどもRouterを介して外部公開しています。

このRouterに対してサーバー証明書を設置できますが、インストール直後は自己署名証明書を利 用しているため、OpenShift上のアプリケーションにアクセスするとブラウザに警告が表示されます (図2.15)。自己署名証明書でも暗号化通信はできますが、実在性証明ができません。必ず、信頼でき る認証局が発行した証明書を設置するようにしてください。

1

入門編



図2.15 自己署名証明書を利用している場合の警告

本節では、誰でも無料で使える「ZeroSSL」でSSLサーバー証明書を発行し、OpenShiftに設置す るまでの手順を紹介します。他の認証局を使用する場合でも、証明書発行以外の手順は同じです。 まずは、証明書の発行手順から説明していきます。

2.2.1 acme.shでサーバー証明書を発行

acme.shはACME^[5]プロトコルのクライアントツールです。SSL証明書を簡単に発行することができます。acme.shは複数の認証局に対応していますが、本例ではデフォルト認証局であるZeroSSLを利用します。また、以下の手順で作成するZeroSSLのSSL証明書は無料です。クレジットカードの登録なども必要ありません^[6]。

acme.sh

https://github.com/acmesh-official/acme.sh



本項の作業は、作業端末(踏み台サーバー)からOpenShiftクラスタにcluster-admin権限 を持つユーザーでCLIログインした状態で実行してください。インストール時に自動作成され たkubeadminユーザーは、cluster-admin権限を持っています。 CLIでのログイン方法については、第3章の「3.2 CLI(ocコマンド)」を参照してください。

^{【5】} ACMEとは Automatic Certificate Management Environment の略で、証明書発行プロトコルのことです。

^[6] https://zerossl.com/features/acme/



まずは acme.shをインストールします。コマンドの email の部分は自分のものに書き換えてください。後で ZeroSSL の登録に使用します。

\$ curl https://get.acme.sh | sh -s email=my@example.com

インストールログに「OK, Close and reopen your terminal to start using acme.sh」と表示 されるので一度ターミナルにログインし直します。

続いて次の4つの環境変数をセットします。

- AWS_ACCESS_KEY_ID: AWSのアクセスキーID (OCPクラスタのインストールに使用したIAM ユーザー openshift-installerのID^[7])
- AWS_SECRET_ACCESS_KEY: AWSのシークレットキー
- LE_API:完全修飾ドメイン
- LE_WILDCARD: ワイルドカードドメイン

\$ export AWS_ACCESS_KEY_ID="..."

\$ export AWS_SECRET_ACCESS_KEY="..."

```
$ export LE_API=$(oc whoami --show-server | cut -f 2 -d ':' | cut -f 3 -d '/' | sed 's/-api././')
$ echo $LE_API
api.democluster.ocpdemo.aoshima.me
```

```
$ export LE_WILDCARD=$(oc get ingresscontroller default -n openshift-ingress-operator -o jsonpath 
='{.status.domain}')
$ echo $LE_WILDCARD
apps.democluster.ocpdemo.aoshima.me
```

必要な環境変数をセットできたら証明書を発行します。

\$ acme.sh --issue --dns dns_aws -d \${LE_API} -d *.\${LE_WILDCARD}

もしここで「AWS Route53 rate exceeded」エラーが発生した場合は、環境変数AWS_DNS_SLOWRATE でAPIのリクエスト間隔を調整することができます^[8]。

証明書の発行に成功すると acme.sh のデフォルトパス (~/.acme.sh) 以下に証明書ファイルが生成されます。管理しやすくするために次のコマンドで別ディレクトリにも出力します。

^{【7】} OCPクラスタのインストールに使用したIAMユーザーでなくても、次のリンクの操作権限を持つIAMユーザーであれば可能です。 https://github.com/acmesh-official/acme.sh/wiki/How-to-use-Amazon-Route53-API

^[8] https://github.com/acmesh-official/acme.sh/wiki/dnsapi#10-use-amazon-route53-domain-api

入門編

```
$ export CERTDIR=$HOME/certificates
$ mkdir -p ${CERTDIR}
$ acme.sh --install-cert -d ${LE_API} -d *.${LE_WILDCARD} --cert-file ${CERTDIR}/cert.pem --key-f
ile ${CERTDIR}/key.pem --fullchain-file ${CERTDIR}/fullchain.pem --ca-file ${CERTDIR}/ca.cer
```

2.2.2 Routerに証明書を設置

前項で取得したサーバー証明書(または別途用意したサーバー証明書)をRouterに設置していきます。



本項の作業は、作業端末(踏み台サーバー)からOpenShiftクラスタにcluster-admin権限 を持つユーザーでCLIログインした状態で実行してください。インストール時に自動作成され たkubeadminユーザーは、cluster-admin権限を持っています。CLIでのログイン方法につい ては、「3.2 CLI (ocコマンド)」を参照してください。

まずは証明書からSecretを作成します。openshift-ingress プロジェクトに router-certs という 名前でSecretを作成します。

\$ oc create secret tls router-certs --cert=\${CERTDIR}/fullchain.pem --key=\${CERTDIR}/key.pem -n
openshift-ingress

次に、router-certsをデフォルトの証明書として使用するように Router の設定を書き換えます。 Router の設定は、openshift-ingress-operator プロジェクトにある default という名前の Ingress Controller オブジェクトで行います。

\$ oc patch ingresscontroller default -n openshift-ingress-operator --type=merge --patch='{"spec": { "defaultCertificate": { "name": "router-certs" }}}'

以上で、設定作業は終了です。

IngressControllerオブジェクトを更新すると、Routerは自動で再デプロイされて新しい証明書の設定が反映されます。この設定により、Webコンソールなどに適切な証明書を使って、外部からアクセスできるようになります。

Routerの再デプロイが完了するまで数分程度かかるので、少し待ってからWebコンソールをブラ ウザで開いて証明書が適切に設置されているか確認してみてください。





CHAPTER 2 環境構築

Part 1 > 入門

Red Hat CodeReady Containers環境

Red Hat CodeReady Containers (以下、CRC) は、ローカルマシンの仮想マシン (VM) 上で動く ミニマルな OpenShift 4環境です。

 Red Hat CodeReady Containers https://developers.redhat.com/products/codeready-containers/overview

通常のOpenShiftと比較すると、次のような違いがあります。

- シングルノードで動作する。つまり Master ノードも Worker ノードも同じ1つの VM 上で動く
- アップグレードはできない。最新のバージョンを使いたい場合は既存のCRCを削除して、新しいCRCを再インストールする(後述の「2.3.3 アップグレード手順」を参照)
- デフォルトでは Monitoring オペレーターが無効になっている。このため、Web コンソールの「Monitoring」機能も使えない
- 一部の設定が変更できない
 - ▶ クラスタのベースドメイン。crc.testingで固定
 - ▶ クラスタの内部通信に利用するアドレス範囲

CRCはあくまでもテストや開発向けの環境です。本番環境として利用する場合や複数人で利用する場合は通常のOpenShiftを用意してください。

2.3.1 システム要件

システム要件はバージョンによって異なるため、インストールしたいCRCのドキュメントを確認し てください。以下はバージョン1.34の例です。

 Red Hat CodeReady Containers 1.34のシステム要件 https://access.redhat.com/documentation/en-us/red_hat_codeready_containers/1.34/ html/getting_started_guide/installation_gsg#minimum-system-requirements_gsg

■ハードウェア要件

最低でも以下のリソースが必要です(バージョン1.34の場合)。CRC上で動かすワークロードに よってはさらに多くのリソースが必要になります。

- CPU : 4 physical CPU cores
- メモリ:9GB
- ストレージ: 35 GB

■OS要件

次のいずれかのOSが必要です。また、CRCのバージョンごとにOSの対応バージョンも異なりま す。詳細は対応するCRCのバージョンのシステム要件を参照してください。

- Windows (対応OSバージョンはCRCの各バージョンごとに要確認)
- macOS (対応OSバージョンはCRCの各バージョンごとに要確認)
- Linux
 - Red Hat Enterprise Linux、CentOS、Fedoraは公式対応(対応OSバージョンはCRCの各バージョンごとに要確認)
 - ▶ Ubuntuと Debian は公式対応はしていないため一部手動でセットアップが必要なこともある

2.3.2 インストール手順

まずはCRCのバイナリまたはインストーラーをダウンロードしてください。

- 最新バージョンのCRCをインストールする場合: Create an OpenShift cluster (local) https://cloud.redhat.com/openshift/create/local
- 特定のバージョンのCRCをインストールする場合: Index of /pub/openshift-v4/clients/crc https://mirror.openshift.com/pub/openshift-v4/clients/crc/

特に特定のバージョンのCRCをインストールする場合は、バイナリやインストーラーと一緒に配布 されている release-info.jsonファイルを確認して、対象のCRCがどのバージョンのOpenShiftに 対応しているのか確認しておくとよいでしょう(図2.16)。 Part 1

入門編

2



図2.16 release-info.jsonの例(CRCと対応しているOpenShiftのバージョンが確認できる)

ちなみにCRCのインストール後は、crc versionコマンドでCRC自身のバージョンと、対応して いるOpenShiftのバージョンを確認することができます。

本書ではmacOSに最新のCRCをインストールする方法を紹介します。

まずは、https://console.redhat.com/openshift/create/localからmacOS向けのCRCインス トーラーをダウンロードします(図2.17)。

E C Red Hat Hybrid Cloud Consol	e All apps and services 👻	¢Ø
OpenShift	Clusters > Create	
Clusters	Create an OpenShift cluster	
Overview	Cloud E Datacenter	🖵 Local
Releases	Red Hat CodeReady Containers (local sandbox)	
Downloads	Create a minimal custer on your desktop/aprop for local development and testing. Note: Your CodeReady installation won't appear in the OpenShift Cluster Manager unless you enable cluster monitoring and telemetry.	
Insights		
Subscriptions >	1 Download what you need to get started	
Cost Management >	CodeReady Containers Download and open the CodeReady Containers file. Opening the file will automatically start a step-by-	
Support Cases 🗹	step installation guide.	
Cluster Manager Feedback 앱	MacOS	
Red Hat Marketplace 12	Pull secret Download or copy your pull secret. You'll be prompted for this information during installation.	
Documentation &	Download pull secret	
	Follow the documentation to install CodeReady containers Run Crc setup to set up your host operating system for the CodeReady Containers virtual machine. Then, run Crc start to create a minimal OpenShift 4 cluster on your computer. View the CodeReady Containers Getting started guide [2]	

図2.17 インストーラーとPull secretの配布ページ

インストーラー (pkgファイル) のダウンロードが完了したら、ファイルをダブルクリックしてインス トーラーを起動します (図2.18)。

Part 1 > 入門編

CHAPTER 2 環境構築

	🌍 CodeReady Containers のインストール	A
	ようこそ CodeReady Containers インストーラヘ	
 はじめに 使用許諾契約 インストール先 インストールの種類 インストール 概要 	This will install CodeReady Containers 1.34.0 on your computer. You will be guided through the steps necessary to install this software. Click "Continue" to continue the setup	
🐣 Red Hat	戻る 続ける	

図2.18 CodeReady Containers インストーラーの起動画面

あとはインストーラーの指示に従ってインストール作業を進めます。 インストールが完了したら、インストールしたCRCのバージョンを確認します。

```
$ crc version
CodeReady Containers version: 1.34.0+34c31851
OpenShift version: 4.9.0 (bundle installed at /Applications/CodeReady Containers.app/Contents/Res 
ources/crc_hyperkit_4.9.0.crcbundle)
```

次に初期設定を行います。

```
$ crc setup
CodeReady Containers is constantly improving and we would like to know more about usage (more det 3
ails at https://developers.redhat.com/article/tool-data-collection)
Your preference can be changed manually if desired using 'crc config set consent-telemetry <yes/n Đ
0>'
Would you like to contribute anonymous usage statistics? [y/N]: y
Thanks for helping us! You can disable telemetry with the command 'crc config set consent-telemet 🔊
ry no'.
INFO Checking if running as non-root
INFO Checking if crc-admin-helper executable is cached
INFO Checking for obsolete admin-helper executable
INFO Checking if running on a supported CPU architecture
INFO Checking minimum RAM requirements
INFO Checking if running emulated on a M1 CPU
INFO Checking if HyperKit is installed
INFO Checking if gcow-tool is installed
```

Part 1

入門編



INFO Checking if crc-driver-hyperkit is installed INFO Checking if CodeReady Containers daemon is running INFO Checking if launchd configuration for tray exists INFO Check if CodeReady Containers tray is running INFO Starting CodeReady Containers trav INFO Checking if CRC bundle is extracted in '\$HOME/.crc' INFO Checking if /Applications/CodeReady Containers.app/Contents/Resources/crc_hyperkit_4.9.0.crc 🔊 bundle exists INFO Extracting bundle from the CRC executable INFO Ensuring directory /Applications/CodeReady Containers.app/Contents/Resources exists INFO Uncompressing crc_hyperkit_4.9.0.crcbundle crc.gcow2: 11.50 GiB / 11.50 GiB [---------- 🗗 -----] 100.00% Your system is correctly setup for using CodeReady Containers, you can now run 'crc start' to sta 🔁 rt the OpenShift cluster

以上で、CRCのインストールと初期セットアップは完了です。



CRCのアップグレードは以下の手順で手動で実施します。まずはCRCが稼働している仮想マシン を削除します。この操作で古いVMに保存されていたデータは削除されます。

\$ crc delete

次に前項と同じ手順で新しいCRCをインストールしてください。この操作で古いCRCが新しい CRCに置き換えられます。crc setupコマンドまで忘れずに実施してください。



CRCのクラスタを稼働させるVMの起動・停止・削除方法およびクラスタへの接続方法を説明します。

■VMの起動方法

次のコマンドでVMを起動します。このとき、Pull-Secret (クレデンシャル情報)を尋ねられるの で、事前に用意しておきましょう。クレデンシャル情報は、CRCのバイナリをダウンロードしたとき と同じ下記URLで取得できます。 • Create an OpenShift cluster (local)

https://cloud.redhat.com/openshift/create/local

```
$ crc start
...
Started the OpenShift cluster.
The server is accessible via web console at:
https://console-openshift-console.apps-crc.testing
Log in as administrator:
Username: kubeadmin
Password: vtAoY-DcVx7-PEjMY-bCXZI
Log in as user:
Username: developer
Password: developer
Password: developer
Use the 'oc' command line interface:
$ eval $(crc oc-env)
$ oc login -u developer https://api.crc.testing:6443
```

VMの起動が完了すると、ログイン情報が標準出力に表示されます。上記の出力例を見てわかるように、kubeadminとdeveloperの2つのアカウントが作成されています。この内容に従ってクラスタ にログインしてください。

■クラスタへの接続方法

ログイン方法は、通常のOpenShiftの場合と同じです。第3章「基本的な操作方法」の内容を参考 に、クラスタ起動時に出力されたログイン情報を使ってログインしてください。

CRCには、ocバイナリが同梱されています。ocがインストールされていない環境や、CRCのクラス タバージョンに対してインストール済みのocのバージョンが古くて使えない場合などは、次のように CodeReady Containers 付属のocにパスを通して使用してください。

\$ eval \$(crc oc-env)

CRC付属のocにパスが通っているかどうかは、次のようにして確認することができます。

\$ which oc
/Users/aoshima/.crc/bin/oc/oc

Part

1

入門編



また、Web コンソールからログインする場合は、crc console コマンドを実行してください。自身のPCのデフォルトブラウザでWeb コンソールが開きます。kubeadmin か developer アカウントを使ってログインしてください^[9] (図2.19)。

アカウントにログイン ^{Open} a terminal and run 'orc consoleonedentiad' to get your oredentias.	Red Hat OpenShift Container Platform
<u>⊐-∀</u> -& •	ようこそ: Red Hat OpenShift Container Platform
1(X)-k ·	
ログイン	

図2.19 CodeReady ContainersのWebコンソール(ログイン画面)

■VMの停止方法

CRCのクラスタを停止させるには、次のコマンドで該当のVMを停止させます。停止しても、VM 上のデータは削除されません。

\$ crc stop

■VMの削除方法

次のコマンドで、CRCのVMを削除することができます。この操作により、VM上のデータはすべ て削除されます。ただし、~/.crc以下に残っているキャッシュデータなどは消えないため、キャッ シュデータも含めて完全に削除する場合は、このディレクトリも削除します。

\$ crc delete

^[9] このログイン場所の違いは、アイデンティティブロバイダーの違いによるものです。アイデンティティプロバイダーについては、第4章「ユー ザー管理」で説明します。



Red Hat Developer (https://developers.redhat.com/) では、ハンズオン形式でOpenShiftを学ぶこ とができる学習コンテンツや、OpenShiftのサンドボックス環境などを無料で提供しています。OpenShift 環境を自前で用意しなくても、手軽にOpenShiftやその関連プロダクトに触れることができるところがうれ しいです。

- OpenShiftの学習コンテンツ https://developers.redhat.com/learn
- OpenShiftのサンドボックス環境 https://developers.redhat.com/developer-sandbox





Chapter



基本的な操作方法





本章では、OpenShift Container Platform(以下、OCP)の基本的な使い方を紹介します。OCPを 利用するには、Webコンソールかコマンドラインインターフェース(Command Line Interface: CLI) を使います。



Webコンソールは、Webブラウザからアクセスできるユーザーインターフェースです。コンテンツ を視覚的に把握・管理することができます。

Web コンソールのURLは、インストールログの最後にユーザー名、パスワードとともに次のように 表示されています。それぞれ太字で表記しています。

```
$ openshift-install create cluster
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export KUBECONFIG=/home 
/ec2-user/openshift/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-console.apps.democluster.oc 
pdemo.aoshima.me
INFO Login to the console with user: "kubeadmin", and password: "SPFiM-bINKC-8HRSd-VAApM"
INFO Time elapsed: 34m43s
```

既存のクラスタを利用するなど、インストールログが確認できない場合は、ログインに必要な情報 を管理者等から入手してください。



ブラウザでWebコンソールのURLを開くと、図3.1のようなログイン画面が表示されます。ユー ザー名とパスワードを入力し、[ログイン] ボタンをクリックしてログインします。

アカウントにログイン ュー ッ - & -	CopenShift Container Platform ようこそ: Red Hat OpenShift Container Platform
バスフード・ 	

図3.1 ログイン画面

ログインに成功すると、図3.2のような画面が表示されます。

Red Hat OpenShift Container Platf	orm						1 O	0	kube:admin v
🕫 Administrator			一時的な管理ユーザー	としてログインします。 <u>クラスターの OAuth 設定</u> を更新し、他のユーザーがログ	インできるようにします	f.			
ホーム		概要							
Operator		クラスター							
ワークロード									
ネットワーク		スタートアップリソース ⑦							1
ストレージ		2 クラスターの設定 推測される設定でクラスターの設定を完了します。		・ ガイド付きドキュメントを使用したビルド ガイド付きドキュメントに従って、アプリケーションをビルドし、主要な機能に	■ 新たな管理機能 Admin パースペクラ	能の確認 ティブの新機能および	リソースを	紹介します。	
Слк		アイデンティティープロバイダーの追加。		ついて学習しましょう。 Monitor your sample application →	API Explorer →				
確認		79-F09-A-08029		Get started with Quarkus using a Helm Chart ->	Operatornuo 🤿				
コンピュート		ドキュメントのすべての手順の表示は		すべてのクィックスタートの表示	OpenShift 4.9 න1	新機能の確認。			
ユーザー管理		詳細 股定の表示	ステータス		アラートの表示	アクティビテ			イベントの表示
管理		クラスター API アドレス	. 4574-	A Insights		進行中			
		https://api.demock/ster.ocpdemo.aoshima.me:6443	0 ////	1件の開	重が見つかりました	進行中のアクラ	ティビティ	ーはありませ	th.
		97cb3fab-3173-4eb9-b5de-e71c5ee5c7ef OpenSNHt Cluster Manager g* プロバイダー AWS OpenSNHt パージョン	A 2021年11月 Alerts are timely fash configure	9E 13:01 not configured to be sent to a notification system, meaning that you may not be no ion when important failures occur. Check the OpenShift documentation to learn h notifications with Alertmanager.	tified in a ow to	最近のイベント - A	Treadyz Inon long	=true g-running rec	■一時停止 → quest(s →
		4.9.5 更新チャネル stable-4.9	クラスターの	史用率 ノードタイプでの絞り込み ・ 使用 15.00 16.5 16.30	1時間 •	- A	Preadyz Ni pending HTTP Serv	•true requests priver has stopp	> ocessed > ped liste >

図3.2 Web コンソール (Administrator 画面)



左側のサイドバー(図3.3)から確認・操作したい項目を選ぶと、その画面に移ります。

¢ € Administrator	-
ホーム	>
Operator	>
ワークロード	>
ネットワーク	>
ストレージ	>
ビルド	>
確認	>
コンピュート	>
ユーザー管理	>
管理	>

図3.3 Administrator画面のサイドバー

Web コンソールは、Administrator用(前掲図3.2)とDeveloper用(図3.4)の2つの視点(パース ペクティブ)を提供しています。サイドバー上部の選択欄で「Administrator」または「Developer」を 選ぶことで、パースペクティブを切り替えられます(図3.5)。



図 3.4 Web コンソール (Developer 画面)
Part 1

入門編

🌣 Administrator	
✿ Administrator	
Developer	

図3.5 AdministratorとDeveloperの切り替え



OCPのCLIは「oc」というコマンドです。ocを使ってOCPの管理、プロジェクトやアプリケーションの作成・管理を行うことができます。OCPのインストーラーはLinuxとmacOSにしかインストールできませんでしたが、OCPのCLIはLinux、macOS、Windowsにインストールすることができます。

3.2.1 ocコマンドのインストール

ocコマンドは、次のいずれかの方法でインストールできます。

- バイナリをダウンロードしてインストール(●~●のいずれか)
 - 稼働中のOCPのWebコンソールからバイナリをダウンロード
 - Red Hat OpenShift Cluster Managerサイトからバイナリをダウンロード https://cloud.redhat.com/openshift/install/aws/installer-provisioned
 - https://mirror.openshift.comからバイナリをダウンロード 過去のバージョンのバイナリを取得可

https://mirror.openshift.com/pub/openshift-v4/clients/ocp/

• RPMでインストール

https://access.redhat.com/documentation/ja-jp/openshift_container_platform/4.9/ht
ml/cli_tools/_openshift-cli-oc#cli-installing-cli-rpm_cli-developer-commands

ocコマンドは後方互換性があるので、操作対象のOpenShiftクラスタのバージョンと同じか、それ より新しいものを使用すれば正常に動作します。バージョン間の互換性の詳細はOCPのオンラインド キュメントを参照してください。

 OpenShift Container Platformのバージョン管理ポリシー https://access.redhat.com/documentation/ja-jp/openshift_container_platform/4.9/ht ml/release_notes/ocp-versioning-policy 本書では、「② Red Hat OpenShift Cluster Manager サイトからバイナリをダウンロード」してインストールする方法を説明します。バイナリファイルの入手方法が違う場合も、バイナリ取得後の作業手順は同じです。他のインストール方法については、以下のオンラインドキュメントを参考にしてください。

OpenShift CLIの使用を開始する

CHAPTER 3 基本的な操作方法

Part 1 > 入門編

https://access.redhat.com/documentation/ja-jp/openshift_container_platform/4.9/ht
ml/cli_tools/_openshift-cli-oc#cli-getting-started

■ バイナリをダウンロードしてインストール

それではまず、Red Hat OpenShift Cluster Manager サイト (図 3.6) にアクセスします。「Command line interface」でインストール先のOSを選択し、[Download command-line tools] ボタンを押して ダウンロードします。

 Red Hat OpenShift Cluster Managerサイト https://cloud.redhat.com/openshift/install/aws/installer-provisioned

なお、踏み台サーバーに SSH でログインしてこの作業を行うときなど、CLI 環境で操作していてブ ラウザを開けない場合は、この [Download command-line tools] ボタンのリンクをコピーして、次の ように curl などを使ってダウンロードしてください。

\$ curl -0 https://mirror.openshift.com/pub/openshift-v4/clients/ocp/stable/openshift-client-linux .tar.gz

E CRed Hat	o or service 🔹	¢ 0	🧟 秀治青島 🔸		
OpenShift Clusters	> Creste > Amazon Web Services > Installer-provisioned infrastructure				
Clusters	instali Openshin ton Aws with installer-provisioned innastructure				
Overview	What you need to get started				
Releases	OpenShift installer				
INSIGHTS	Download and extract the install program for your operating system and place the file in the				
Subscriptions >	directory where you will store the installation configuration files. Note: The OpenShirt Install program is only available for Linux and macOS at this time.				
Cost Management >	Linux Download installer				
Support Cases 12					
Cluster Manager Feedback 년	dback 2* Download or copy your pull secret. You'll be promoted for this information during				
Red Hat Marketplace 🖒	installation.				
Documentation 🖄	Download pull secret				
	Command line Interface Download the OpenShift command-line tools and add them to your PATH. Liux Download command-line tools				
	When the installer is complete you will see the console URL and credentials for accessing your new cluster. A kubeconfig file will also be generated for you to use with the oc CLI tools you downloaded.				
	Follow the documentation to configure your AWS account and run the installer		ଡ		

図 3.6 Red Hat OpenShift Cluster Managerサイト

Part

1

入門編

ダウンロードしたアーカイブを展開します。Linuxでは次のように展開します。

\$ tar xvf openshift-client-linux.tar.gz

展開されたocファイルを実行可能なパスに移動します。Linuxであれば、/usr/local/bin/の中が 実行可能であることが多いです。

\$ sudo mv oc /usr/local/bin

これで、ocコマンドを利用できるようになります。 試しに次のコマンドでインストールが成功しているか確認してみてください。

\$ oc version
Client Version: 4.9.5

上記のversionのように、ocの次に入力した単語を「サブコマンド」と言います。

3.2.2 oc コマンドでログイン

CLIでも、OCPの操作を始める前にログインする必要があります。ログインは、loginサブコマンド で行います。

```
▶ ログインの入力例
```





3.2.3 oc コマンドの使い方

ocをサブコマンドなしで実行すると、使い方の簡単な説明が表示されます。

\$ oc OpenShift Client

This client helps you develop, build, deploy, and run your applications on any OpenShift or Kubernetes cluster. It also includes the administrative commands for managing a cluster under the 'adm' subcommand.

To familiarize yourself with OpenShift, login to your cluster and try creating a sample application:

oc login mycluster.mycompany.com oc new-project my-example oc new-app django-psql-example oc logs -f bc/django-psql-exampl

. . .

使えるコマンドは、helpサブコマンドで表示できます。

サブコマンドの使い方を調べたいときは、「oc help login」のようにhelpサブコマンドに続けて 調べたいサブコマンドを入力するか、もしくは「oc login -h」のようにサブコマンドに続けて-hを入 力します。

▶ helpでloginサブコマンドの使い方を調べる

\$ oc help login Log in to your server and save login for subsequent use First-time users of the client should run this command to connect to a server, establish an auth enticated session, and save connection to the configuration file. The default configuration will be saved to your home directory under

".kube/config".

...

▶ -hでサブコマンドloginの使い方を調べる

\$ oc login -h Log in to your server and save login for subsequent use

First-time users of the client should run this command to connect to a server, establish an auth $\ensuremath{\textcircled{o}}$ enticated session, and

save connection to the configuration file. The default configuration will be saved to your home D directory under

Part 1

入門編

3

```
".kube/config".
```

それでは、実際にいくつかコマンドを使ってみましょう。

たとえば、意図したバージョンのクラスタが正常にインストールされていることを確認したい場合 は、oc get clusterversionコマンドを使います。

<pre>\$ oc get clusterversion</pre>					
NAME	VERSION	AVAILABLE	PROGRESSING	SINCE	STATUS
version	4.9.5	True	False	62m	Cluster version is 4.9.5

上記の出力例は、バージョン4.9.5のクラスタが正常にインストールされていることを示しています。

また、ノードのステータスを確認したい場合は、oc get nodeコマンドを使ってください。

\$ oc get node				
NAME	STATUS	ROLES	AGE	VERSION
ip-10-0-132-54.ap-northeast-1.compute.internal	Ready	master	90m	v1.22.0-rc.0+a44d0f0
ip-10-0-156-248.ap-northeast-1.compute.internal	Ready	worker	80m	v1.22.0-rc.0+a44d0f0
ip-10-0-164-88.ap-northeast-1.compute.internal	Ready	master	91m	v1.22.0-rc.0+a44d0f0
<pre>ip-10-0-182-235.ap-northeast-1.compute.internal</pre>	Ready	worker	77m	v1.22.0-rc.0+a44d0f0
<pre>ip-10-0-196-225.ap-northeast-1.compute.internal</pre>	Ready	worker	81m	v1.22.0-rc.0+a44d0f0
ip-10-0-199-89.ap-northeast-1.compute.internal	Ready	master	91m	v1.22.0-rc.0+a44d0f0

上記の出力例は、全ノードのステータスが「Ready」であることを示しています。

■ocコマンドチートシート

運用管理でよく使うコマンドの例を表3.1に挙げます。チートシートとしてご利用ください。

コマンド	説明
oc login -u USER -n PROJ	カレントプロジェクトをPROJとしてUSERでログイン
oc cluster-info	クラスタ情報を出力
oc status	カレントプロジェクトの状態を出力
oc whoami	ログイン中のユーザーを表示
oc projects	プロジェクトの一覧を出力
oc project PROJ	カレントプロジェクトを <i>PR0J</i> に変更
oc get all	Pod、Service、Deployment、ReplicaSetなどの一覧を出力
oc get podshow-labels	Pod一覧にラベル情報も付加して出力
oc get pod -1 KEY=VALUE	条件に合致するPod一覧を出力
oc get events -w	プロジェクトで起こるイベントを継続的に出力

表31	運田管理で、	よく使う	or コマンド
10.1	建用自住し	のへにノ	



コマンド	説明
oc get pvc -n <i>PROJ</i>	<i>PROJ</i> のPersistentVolumeClaimの一覧を出力
oc get deployall-namespaces	すべてのプロジェクトのDeploymentの一覧を出力
oc get node -o wide	Nodeの一覧をより詳しく表形式で出力
oc get svc SVC -o yaml json	<i>SVC</i> の情報をyaml json形式で出力
oc get csv	ClusterServiceVersion情報の出力
oc describe rc RC	RCの詳細情報を出力
oc new-project PROJ	新規プロジェクトとして PROJ を作成
oc create -f MANIFEST	MANIFEST ファイル (yam1 json) で定義されたリソースを新規作成
oc apply -f MANIFEST	MANIFEST ファイル (yaml json) で定義されたリソースが存在しない場合は 新規作成、存在する場合は更新を適用
oc process TEMPLATE oc apply -f	TEMPLATE ファイル (yam1 json)をリソースリストに変換して app1y コマン ドにパイプで渡す
oc edit dc <i>DC</i>	DCの設定を変更(vi形式)
oc patch dc DC -p PATCH	<i>DC</i> に <i>PATCH</i> 文字列 (yaml json)の内容に基づくパッチをあててオブジェクト を更新する
oc set env dc/DC KEY=VALUE	DCに環境変数を追加
oc delete pod <i>POD</i>	POD を削除
oc exec <i>POD</i> bash -it	PODのファーストコンテナで対話的なbashを実行
oc exec POD date -c CONTAINER	<i>POD</i> の <i>CONTAINER</i> でdateコマンドを実行(リモートコマンド)
oc rsh POD	PODのファーストコンテナでシェルセッションを開く
oc debug dc/DC	<i>DC</i> のデバッグ用Podの立ち上げ
oc port-forward svc/SVC 8888:http	ローカルの 8888 ポートへのアクセスを SVC の http ポートにポートフォワード する
oc explain RESOURCE	RESOURCE (たとえば pods、nodes) についての説明とフィールドの表示
oc logs POD -f	POD のログを継続的に出力
oc logs POD -c CONTAINER	<i>POD</i> のCONTAINERのログを出力
oc rollout undo dc/DC	DCを1つ前のリビジョンにロールバック
oc config view	CLIの設定情報を出力
oc adm policy add-cluster-role-to- user <i>ROLE USER</i>	USER に ROLE のクラスタロールを付与
oc adm top node	Nodeのメトリクスを出力
oc adm drain NODE	NODE からすべての Pod を退避
oc adm cordon NODE	NODEをPodのスケジュール不可にする設定
oc adm uncordon NODE	NODEをPodのスケジュール可能にする設定





本章では、ユーザーの登録と削除、権限設定などユーザーの管理方法について具体例を通して説 明します。



ユーザーがOpenShiftを操作するためには、まずクラスタに対して「認証(Authentication)」を行う必要があります。認証とは、リクエストを送ってきたのが誰なのか確認することです。

OpenShiftにはOAuthサーバーが含まれており、ユーザーがOpenShiftを操作したいときは、最初 にこのOAuthサーバーにアクセストークンを要求します。すると、OAuthサーバーは連携している Identity Provider にそのユーザーが登録されているか確認を行い、確認ができたらそのユーザーにア クセストークンを発行します。ユーザーはこのアクセストークンを使ってAPIにリクエストを送ります。 以上がユーザーを識別する認証のプロセスになります。

■Identity Provider一覧

Identity Provider (IdP) は、ユーザー認証をサービスとし提供します。OpenShift 4で使用できる Identity Provider は、以下のとおりです。

- HTPasswd: .htpasswdファイルを利用
- Keystone: OpenStack Keystone v3 サーバーを利用
- LDAP: LDAPv3サーバーを利用
- Basic 認証: HTTPのBasic 認証を利用
- Request Header: X-Remote-User などのリクエストヘッダー値を利用
- GitHub/GitHub Enterprise: GitHub/GitHub EnterpriseのOAuthを利用
- GitLab: GitLabのOAuthまたはOpenID Connectを利用
- Google: GoogleのOpenID Connectを利用
- OpenID Connect: OpenID Connectの認可コードフローを利用

これら Identity Provider については、以下のオンラインドキュメントも参照してください。

• サポートされるアイデンティティープロバイダー

https://access.redhat.com/documentation/ja-jp/openshift_container_platform/4.9/ht
ml/authentication_and_authorization/supported-identity-providers

4.1.2 認可(Authorization)

「認可」では、ユーザーが行おうとしている操作に対して、そのユーザーに実行権限があるのかどうかを確認します。OpenShiftでは、RBAC (Role Based Access Control) によって、この実行権限を管理しています。

どのような操作が許可されるか定義されたRoleをUserに対してRoleBindingで紐づけます。認可で使用されるオブジェクトを以下に整理します(図4.1)。

- Role:プロジェクトスコープのリソース(たとえばPod、Serviceなど)に対して許可される操作の一覧。Role自身は特定のプロジェクトに紐づく。
- ClusterRole: プロジェクトスコープのリソース + Cluster レベルのリソース (たとえば Node、 PersistentVolume、プロジェクトなど) に対して許可される操作の一覧。ClusterRoleは、特定のプロジェクトに紐づかない。
- RoleBinding: RoleとUserもしくはClusterRoleとUserを紐づける。RoleBindingは特定のプロジェクトに紐づく。
- ClusterRoleBinding: ClusterRoleとUserを紐づける。ClusterRoleBindingは、特定のプロ ジェクトに紐づかない。



図4.1 権限設定に関するリソース間の関係

ClusterRoleはClusterRoleBindingでUserに紐づける場合と、RoleBindingでUserに紐づける場合があります。

ClusterRoleBindingで紐づけた場合は全プロジェクトに対してClusterRoleで定義された権限 が付与されますが、RoleBindingで紐づけた場合はRoleBindingを作成したプロジェクト内でのみ、 その権限が付与されます。 Part

1

入門編



プロジェクトスコープのリソースに関する権限設定がしたい場合でも、複数プロジェクトで使用す るような権限設定であれば、RoleではなくClusterRoleで管理しておくことで全プロジェクトから参 照できるようになるため、同じ内容のRoleをプロジェクトごとに管理する手間が省けて便利です。



Identity Providerを追加して新規ユーザーを登録する方法を説明します。今回は、一番手軽な HTPasswdを利用することにします。HTPasswdは、ユーザー名とパスワードを格納する.htpasswd ファイルを使用してユーザー認証を行う Identity Provider です。 次の手順で作業します。

- 1. .htpasswdファイルを作成
- 2. .htpasswdファイルのSecretを作成
- 3. OAuthカスタムリソースを編集して Identity Provider を登録
- 4. 新規ユーザーに権限を付与

1.と**2.**で Identity Provider の設定を行い、**3.** で OpenShift に Identity Provider を登録します。ここ までが「認証」部分の設定になります。最後に、**4.**でユーザーに権限を与える「認可」部分の設定を 行います。

本節の以降の作業では、cluster-admin権限を持つユーザーで実施してください(kubeadmin ユーザーはcluster-admin権限を持っているので、クラスタインストール直後はkubeadminを使用してください)。

4.2.1 .htpasswdファイルを作成

まずは、踏み台サーバー上で、ユーザー名とパスワードを含む.htpasswdファイルを新規作成して ください。今回は admin01、user01、user02という3人のユーザーを登録することにします^[1]。 以下のコマンドを実行すると、プロンプトでパスワード入力を求められます。

```
$ mkdir ~/htpasswd
$ cd ~/htpasswd
```

\$ htpasswd -cB ./users.htpasswd admin01

htpasswd コマンドがインストールされていない場合は、インストールしてください。RHEL 8 (Red Hat Enterprise Linux 8)の場合は、 sudo yum install httpd-toolsでインストールできます。

\$ htpasswd -B ./users.htpasswd user01
\$ htpasswd -B ./users.htpasswd user02

※-cは新規ファイル作成、-Bはbcryptでパスワード作成のオプション。

4.2.2 .htpasswdファイルの Secret を作成

.htpasswdファイルの内容を、OpenShiftのopenshift-config プロジェクト内にSecretとして登録してください。ここでは、users.htpasswdファイルを、htpass-secretという名前のSecretで登録することにします。

\$ oc create secret generic htpass-secret --from-file=htpasswd=./users.htpasswd -n openshift-config

4.2.3 Identity Providerを登録

次にOAuthサーバーが今作成したSecretを利用するように設定します。OAuthサーバーの設定 は、OAuthというカスタムリソースに記載されています。オブジェクトの名前はclusterです。 CLIから設定する場合は、まず次のようにしてOAuthカスタムリソースにパッチをあてます。

\$ oc patch oauth cluster --type=merge --patch='{"spec":{"identityProviders":[{"htpasswd":{"file D
Data": {"name":"htpass-secret"}},"mappingMethod":"claim","name":"my_htpasswd_provider","type":"H D
TPasswd"}]}'

もちろん、認証サーバーを複数登録することも可能です。その際は、spec.identityProvidersに 複数の認証サーバー設定を記載してください。

Web コンソールから設定する場合は、Administrator 画面で左側のメニューから [管理] → [クラ スター設定] をクリックします。そのあと、右側のパネルでクラスター設定の [設定] タブの設定リ ソース一覧の中から [OAuth] を選択すると、OAuthの設定画面(図4.2) に移動します。この画面 で Identity Provider の設定をすることができます。 Part



図 4.2 Identity Provider を Web コンソールで設定

Part 1 > 入門編

図4.3は、Identity Providerを追加した後、Web コンソールでログインしたときの画面です。my_ htpasswd_providerという Identity Provider が新規登録されています。

ログイン: kube:admin my_htpasswd_provider	Red Hat OpenShift Container Platform

図 4.3 Web コンソールのログイン画面(Identity Provider 追加後)

4.2.4 新規作成したユーザーでログイン

認証サーバーに登録したユーザーで一度ログインを行うと、OpenShiftにUserとIdentityリソースが作成されます。

\$ oc login https://api.democluster.ocpdemo.aoshima.me:6443 -u admin01 \$ oc login https://api.democluster.ocpdemo.aoshima.me:6443 -u user01 \$ oc login https://api.democluster.ocpdemo.aoshima.me:6443 -u user02

Webコンソールからログインした場合もUserとIdentityが作成されるので、ocコマンドで再度ロ グインする必要はありません。

oc getコマンドでUserとIdentityの一覧を表示してみます。

> oc get	user		
NAME	UID	FULL NAME	IDENTITIES
admin01	234e5e5d-7a9e-42f9-a6a3-dfdf3747b34c		my_htpasswd_provider:admin01
user01	2f565a39-6929-4f63-a6b5-87e97fd33d76		my_htpasswd_provider:user01
user02	cd3754e1-38a5-4ab6-be38-cd30a0b1ef8a		<pre>my_htpasswd_provider:user02</pre>

<pre>\$ oc get identity NAME</pre>	IDP NAME	IDP USER NAME	USER NAME	USER UID
<pre>my_htpasswd_provider:admin01 Sources_uscloser.admin01</pre>	<pre>my_htpasswd_provider</pre>	admin01	admin01	234e5e5d-7a9e-42 Đ
r9-aba3-drdr3/4/D34C my_htpasswd_provider:user01 63-a6b5-87e97fd33d76	my_htpasswd_provider	user01	user01	2f565a39-6929-4f €
<pre>my_htpasswd_provider:user02 b6-be38-cd30a0b1ef8a</pre>	<pre>my_htpasswd_provider</pre>	user02	user02	cd3754e1-38a5-4a €

各ユーザーが、どのIdentity Provider と紐づけられているのかを確認することができます^[2]。

4.2.5 ユーザーに権限を付与

admin01ユーザーをクラスタ管理者として使用するには、cluster-adminというClusterRoleを admin01ユーザーにバインドします。

\$ oc adm policy add-cluster-role-to-user cluster-admin admin01

Part

^{【2】} 確認はkubeadminユーザーで行ってください。ここで新規作成したユーザーはまだ権限を付与されていないため、UserやIdentityの一覧情報を取得することができません。

Part 1 入門編 CHAPTER 4 ユーザー管理

> なお、cluster-adminはデフォルトでクラスタに登録されている ClusterRole なので、自分で作成 する必要はありません。このように、よく使用されそうな ClusterRole は、デフォルトで登録されて います。

- admin:プロジェクトマネージャ。RoleBindingで使用される場合、adminには、プロジェクト内のすべてのリソースの表示と、プロジェクト内のQuota以外のすべてのリソースを変更する権限があります。
- basic-user:プロジェクトおよびユーザーについての基本的な情報を取得できるユーザーです。
- cluster-admin: すべてのプロジェクトですべてのアクションを実行できるスーパーユーザーです。RoleBindingでユーザーにバインドされる場合、Quotaに対する完全な制御およびプロジェクト内のすべてのリソースに対するすべてのアクションを実行できます。
- cluster-status:基本的なクラスタのステータス情報を取得できるユーザーです。
- edit:プロジェクトのほとんどのオブジェクトを変更できるが、ロールまたはバインディングを表示したり、変更したりする機能を持たないユーザーです。
- self-provisioner:独自のプロジェクトを作成できるユーザーです。
- view:プロジェクト内でほとんどのオブジェクトを確認できるユーザーです。オブジェクトを変 更することはできません。また、ロールとバインディングは表示も変更もできません。

これらデフォルトで登録されている ClusterRole については、以下のオンラインドキュメントも参照 してください。

デフォルトのクラスタロール

4.3.1

https://access.redhat.com/documentation/ja-jp/openshift_container_platform/4.9/ht
ml/authentication_and_authorization/using-rbac#default-roles_using-rbac



) kubeadminの削除

OpenShiftは、インストール時にクラスタ管理者権限(cluster-admin)を持つkubeadminユー ザーが作成されます。このkubeadminユーザーは非常に強い権限を持っていますが、そのパスワー ドはインストールログに平文で保存されているため、このユーザーをそのまま使い続けるべきではあ りません。クラスタのセキュリティを強化するために、新たにcluster-admin権限を持つユーザーを 作成し、kubeadminは削除してください。 しかし、cluster-adminロールを付与されたユーザーが作成される前にkubeadminユーザーを削除してしまうと、クラスタ管理者権限を持つユーザーが1人もいなくなってしまいます。そうなると、 クラスタの再インストールが必要になるので十分注意してください^[3]。

それでは、kubeadminを削除しましょう。まずは、kubeadmin以外のcluster-adminユーザーでク ラスタにログインしておきます。つまり今回は、admin01ユーザーでログインします。

admin01ユーザーでログインしたら、kubeadmin Secretを削除します。

\$ oc delete secrets kubeadmin -n kube-system

以上で、kubeadminユーザーの削除は完了です。

4.3.2 kubeadmin以外のユーザーの削除

次にkubeadmin以外のユーザーを削除してみましょう。ここでは例としてuser02を削除してみま す。

まず、Identity Provider 上のユーザーエントリを削除します。Identity Provider に HTPasswd を使用 している場合は、すでに登録している.htpasswdのSecretリソース(htpass--secret)の内容を書 き換えます。

上記のSecretリソースから.htpasswdファイルの中身を抜き出し、users.htpasswdという名前で 作業端末に保存します。

\$ oc get secret htpass-secret -o jsonpath={.data.htpasswd} -n openshift-config | base64 -d > use rs.htpasswd

次に、このusers.htpasswdファイルからuser02のユーザーエントリを削除します。

\$ htpasswd -D users.htpasswd user02

書き換えた users. htpasswd ファイルをもとに Secret オブジェクトを更新します。

\$ oc create secret generic htpass-secret --from-file=htpasswd=users.htpasswd --dry-run=client -o yaml -n openshift-config | oc replace -f -

以上で、Identity Providerの編集が終わりました。

【3】 上記の手順どおりに admin01 ユーザーを作成し、admin01 ユーザーに cluster-admin権限を付与できていれば問題ありません。



続いて、OpenShift上のUserオブジェクトとIdentityオブジェクトを削除します。これらのオブ ジェクトを削除する権限を持ったユーザー、つまり今回でいえばadmin01ユーザーでログインして、 次の作業を実施してください。

まずは、Userリソースを削除します。

\$ oc delete user user02

次に、Identityリソースを削除します。

\$ oc delete identity my_htpasswd_provider:user02

これでuser02の削除が完了しました。



Chapter



アプリケーションの実行



Part 1 > 入門編 CHAPTER 5 アプリケーションの実行

本章では、具体例を通してサンプルアプリケーションのデプロイ方法を説明します。



まずは、アプリケーションをデプロイするためのプロジェクトを用意します。OpenShiftでは、プロ ジェクトという単位でリソースを分離しています。Kubernetes でいう Namespace のように使いま す^[1]。プロジェクト単位でのリソース制限やユーザーの権限制御、プロジェクトをまたいだコンテナ 同士の通信制御などマルチテナント構成を想定した機能も豊富です。

リソースには、プロジェクトに属するもの(例:Pod、Serviceなど)とプロジェクトに属さないもの (例:Node、PersistentVolumeなど)があるため、慣れないうちは注意が必要です。リソースがプロ ジェクトに属すものなのかそうでないのかは、次のコマンドで確認することができます。

▶ プロジェクトに属するリソース一覧の表示

<pre>\$ oc api-resourcesnamespaced=true</pre>					
NAME	SHORTNAMES	APIVERSION	NAMESPACED	KIND	
bindings		v1	true	Binding	
configmaps	CM	v1	true	ConfigMap	
endpoints	ер	v1	true	Endpoints	
events	ev	v1	true	Event	

▶ プロジェクトに属さないリソース一覧の表示

<pre>\$ oc api-resourcesnames</pre>	oaced=false			
NAME	SHORTNAMES	APIVERSION	NAMESPACED	KIND
componentstatuses	CS	v1	false	ComponentStatus
namespaces	ns	v1	false	Namespace
nodes	no	v1	false	Node
persistentvolumes	pv	v1	false	PersistentVolume

5.1.1 プロジェクトの作成

Webコンソールからプロジェクトを作成します。以下の操作は、プロジェクト作成権限を持つユー ザーで実施してください。本書では、前章で作成した cluster-adminの admin01 ユーザーで実施し ます。

【1】 実態としては、Kubernetesのnamespaceにいくつかのannotationが追加されたものです。

■Webコンソール (Administrator画面) から作成

Web コンソールの Administrator 画面で、左側のメニューから [ホーム] → [プロジェクト] を選択 してから、画面右上の [プロジェクトの作成] ボタンをクリックします。図5.1のようなダイアログボッ クスが表示されるので、必要な情報を入力してから [作成] ボタンをクリックしてください。

E Container Pla									
☆ Administrator ホーム	÷	プロジェクト		プロジェクトの作成 ^{名前・} ①				70919	'⊦o#ø
概要		▼ フィルター ・ 名前	 名前で検索 	my-project 表示名		_			
プロジェクト		名前	表示名 💷	My Project		1	CPU I	作成済み 1	
検索		CR default	表示名なし	說明				3 2021年11月9日 12:41	1
API Explorer		🛞 kube-node-lease	表示名なし	This is an example project				2021年11月9日 12:41	1
1.321		PB kube-public	表示名なし					3 2021年11月9日 12:41	1
Operator		🛞 kube-system	表示名なし		キャンセル	(frat		3 2021年11月9日 12:41	1
ワークロード		(PB) openshift	表示名なし	S Active	リクエスターなし	_		3 2021年11月9日 12:48	1
ネットワーク		PR openshift-apiserver	表示名なし	C Active	リクエスターなし	657.3 MiB	0.073 ⊐ 7*	❸ 2021年11月9日 12:46	1
ストレージ		openshift-apiserver- operator	表示名なし	Active	リクエスターなし	105.9 MiB	0.009 37		1
ビルド		PR openshift-authentication	表示名なし	C Active	リクエスターなし	46.4 MiB	0.011 🗆 77	❹ 2021年11月9日 12:46	1
確認		openshift- authentication-operator	表示名なし	Active	リクエスターなし	126.9 MIB	0.023 ⊐ 7⁄2	2021年11月9日 12:42	:
コンピュート		er openshift-cloud- controller-manager	表示名なし	C Active	リクエスターなし			3 2021年11月9日 12:42	1
ユーザー管理		controller-manager- operator	表示名なし	O Active	リクエスターなし	69.4 MiB	0.001 ⊐7	● 2021年11月9日 12:42	I
管理 クラスター設定		Credential-operator	表示名なし	C Active	リクエスターなし	192.5 MB	0.002 ⊐77	❸ 2021年11月9日 12:42	1
namespace		CR openshift-cluster-csi-	表示名なし	C Active	リクエスターなし	874.5 MiB	0.011 37	3 2021年11月9日 12:42	1

図 5.1 Administrator 画面でプロジェクトを作成

なお、プロジェクトの作成は、プロジェクトの作成権限を持っているユーザーであれば、Webコン ソールのDeveloper 画面から作成することも、CLIから作成することもできます。詳しくは、以下の オンラインドキュメントを参照してください。

プロジェクトの使用

https://access.redhat.com/documentation/ja-jp/openshift_container_platform/4.9/ht
ml/building_applications/_projects#working-with-projects



開発者ユーザーにも、my-project プロジェクトへのアクセス権限を与えます。Developer 画面で、 Developer 画面の [プロジェクト] で操作したいプロジェクト (my-project) を選択して [プロジェク トアクセス] タブに移動します。そこで図5.2のように、

名前:「user01」と入力

Part 1



ロール: [Admin] を選択

最後に[保存]ボタンをクリックします。これにより、user01もmy-project内では管理者として振る舞うことができるようになります。

E CopenShift Container Platform			\$ 1	0 0	admin01 v
Developer •	プロジェクト:my-project 👻				
+追加 + 地口 > I	my-project setime				アクション ・
Demo	概要 詳細 プロジェクトアクセス				
²⁸⁸⁶⁰ 検索	プロジェクトのアクセス制限のことでのユーザーのアクセスを追加または制度できます。ロールペースのアクセス制限のより詳細な管理については、Roles および Role Bindings ロールペースのアクセス制限についてのドキュメント 皮を発展してください。	に表示されます。】	単細は		
ピルド	名前 ロール				
Helm	admin01 🗸 🗢				
プロジェクト	user01 Admin 🔹 🗭				
設定マップ	◎ アクセスの追加				
シークレット	② このベージに変更を放えました。 依存をクリックして変更を依存するか、また出リロードモクリックして変更をキャンセルします。				
	XAL JUL-R				

図5.2 プロジェクトのアクセス権限をユーザーに与える



上記の操作により、内部的には admin というプリセットのクラスタロールを user 01 に紐づけるロールバインディングが my-project 内に作成されています。

プリセットの権限設定ではなく、より細かい権限設定をしたい場合は、自分でロールまたは クラスタロールを作成してユーザーに紐づけることもできます。詳しくは、以下のオンライン ドキュメントを参照してください。

 RBACの使用によるパーミッションの定義および適用 https://access.redhat.com/documentation/ja-jp/openshift_container_platform/4.9/ html/authentication_and_authorization/using-rbac



アプリケーションをデプロイする方法は複数ありますが、ここでは一例としてWebコンソールから 既存のコンテナイメージを利用してサンプルアプリケーション(workshop-terminal)をデプロイする 方法を紹介します。

workshop-terminal

https://github.com/openshift-homeroom/workshop-terminal

5.3.1 アプリケーションのデプロイ方法

まずは、user01でWebコンソールにログインして、Developer画面を開きます。

次に、左側のメニューから[+追加]を選択し、プロジェクトは「my-project」を選択します(図 5.3)。

■ ^{Sed} Hat OpenShift Container Platform							÷	Ø	user01 -
♦ Developer	プロジェクト: my-project 🔹								
+追加	追加								
トポロジー	オプションのいずれかからアプリケーション、コ	ンポーネント、またはサービスをf	*成する方法を選	択します。				~C) 詳紙の表示
確認									
検索	スタートアップリソース ⑦								:
ビルド	鮒 サンプルを使用したアプリケーションの作	Fat .	🛃 ガイド付	きドキュメントを使用したビルド		■ Developer 新機能の紹介			
Helm	コードサンブルを選択して、アブリケーションの作成を開始します。 Back Quarkus a			キュメントに従って、アフリゲーションをビルドし、 ましょう。	Developer パースペクティブの機動販売よりリソースを紹介します。 原定された Helm チャートの検出 ◆ トボロジー内でのアプリケーションの迅速なビルドを開始する →				
プロジェクト	Basic Spring Boot →		Get started with Quarkus using s2i → Get started with Spring →						
設定マップ	すべてのサンプルを表示		すべてのクィ	ックスタートの表示		OpenShift の新機能 C			
Su-Allin L									
	開発者カタログ	Git リポジトリー		⊜ コンテナーイメージ	場所:口	ーカルマシン	旦 サンプル		
	舗 すべてのサービス	� Git からのインポート		イメージストリームまたはイメージ レジストリータグからの現存イメー	🛙 YAML	. のインポート	コードサンプルか ンを作成します。	らアプリケー	-ショ
	カタログを参照してサービスを検出 し、デプロイし、これらに接続しま	ビルドおよびデプロイされる ポジトリーからのコードのイ	される Git リ ジのデプロイ YAI		YAML また ソースの作	ML または JSON 定義からのリ ースの作成			
	Τ.	F							
	■ データベース				F SAR	///////////////////////////////////////			
	カタログを参照して、アプリケー ションに追加するデータペースサー ビスを検出します。				ローカルラ OpenShift プロード	『スクトップから : への JAR ファイルのアッ			
	♥ Operator 提供								
	カタログを参照して Operator の管 理対象サービスを検出し、デプロイ								

図5.3 [+追加]を選択

ここで [コンテナーイメージ]を選択すると、設定値の入力画面に遷移します (図5.4・図5.5)。

Part



プロジェクト: my-project ▼ アプリケーション: すべてのアプリケーション ▼ イメージのデプロイ イメージ イメージストリームまたはイメージレジストリーから既存のイメージをデプロイします。 ④ 外部レジストリーからのイメージ名 quay.io/openshifthomeroom/workshop-terminal:3.4.3 検証済み プライベートリポジトリーからイメージをデプロイするには、イメージレジストリーの認証情報を使用して □ 非セキュアなレジストリーからのイメージの許可 内部レジストリーからのイメージストリームタグ ランタイムアイコン 🖸 openshift アイコンは Topology ビューでイメージを表します。また、ラベルはアイコンを定義するリソースに追加されます。 一般 アプリケーション名 workshop-terminal-app リソースにラベルを付けるためにアプリケーションのグループに付けられた一意の名前。 名前 * workshop-terminal -関連するリソースに名前を付けるために使用されるコンポーネントに指定される一意の名前。 リソース 生成するリソースタイプの選択 デプロイメント apps/Deployment Deployment は、Pod および ReplicaSet の宣言的な更新を有効にします。 〇 デプロイメント設定 apps.openshift.io/DeploymentConfig DeploymentConfig は Pod のテンプレートを定義し、新規イメージまたは設定変更のデプロイを管理します。

図5.4 コンテナイメージデプロイの入力値(「イメージのデプロイ」画面・前半)

プロジェクト:my-project * アプリクーション: すべてのアプリケーション *
Final イジョン Final クジョン Final クジョン Final アブリケーションのシルートの作成 Kブリック URL でのアブリケーションのシス開 Final Automation Fi
C アプリケーションへのルートの特徴 Kプリック URL でのアプリケーションの公開 W 課価 なルーライングオプションを身表示にする AL W m なれのp-terminal-my-projectapps democluster ocpdemo a oshima me ルートのパプリック RA ト 名。 設定 されない場合は、ホスト 名が生成されます。 K C / ーク・パト クライック やサービス にルーティングするために意味するパス。 P・グリトボート To Pot トラフィック やサービスにルーティングするために意味するパス。 P・グリトボート To Pot トラフィック の デリットボート。 E ge E ge C
★ 「オリック ルル、 での アブリッ ーシュンの公開 ★ 詳細な ルーティング オブションを 非表示 にする ★ 詳細な ルーティング オブションを 非表示 にする ★ 非細な ルーティング オブションを 非表示 にする ★ かくれんの、 ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・
w 学 課題なルーティングオプションを非表示にする tx L workshor terminal-my-project.app.democluter.ocpdemo.aoshima.me
 ・ 詳細なルーティングオブションを発表示にする ホスト名
Nachap-terminal-my-projectappademocluter.opdemo.aoshima.me ルートのパブリックホスト&、酸定されない場合は、ホスト&が生成されます。 パス / ルークーがトラフィックをサービスにルーティングするために監視するパス。 ターグットホート 10080 トラフィククのプリボート。 ピオコクズのルート レークロペナジアイート。 ビオコクズのルート レークロペナジャード、 ビオコクズのルート トラレインタンク・ボート。 Edge
workshop-terminal-my-project.apps.democluster.ocpdemo.aoshima.me ルートのパブリックホスト&、施定されない場合は、ホスト&が生成されます。 パズ / ルークーがトラフィックをサービスにルーティングするために監視するパス。 クーグットボート 10080 トラフィックのケーグットボート。 セキュフタルート レークロゼキュンティーム、証明書を提供するために複数のTLS接続タイプを使用して保護できます。 TLS接端 Edge B セキュアなルート レートのセキュンティーム、証明書を提供するために複数のTLS接続タイプを使用して保護できます。 TLS接端 Edge B セキュアないトライ TLT などのセキュンディームのドラフィックのポリシー。 Edge HTTP などのセキュンアではないスキームでのトラフィックのポリシー。 Edge **** **** #*** #**** ****
ルートのパプリックホスト名、開始されない場合は、ホスト名が生成されます。 // / ルーターがトラフィックをサービスにルーティングするために監視するパス。 ターグットボート 10080 トラフィックのターゲットボート。 はキュソティー 営 セキュアなルート ルートのセキュリティーは、証明書を提供するために複数の TLS 接端タイプを使用して保護できます。 TLS接端 Edge Redirect 4TTP などのセキュアではないスキームでのトラフィックのポリシー。 証明書 1tdge および rk-encrypt 終端の TLS 証明書。無定されない場合は、ルーターのデフォルト証明書が使用されます。 Emfle
パレー・ワーゲトラフィックをサービスにルーティングするために監視するバス。 ターグットボート 10080 トラマックのターグックトボート。 セロックティー 20 せネュアのターグックトボート。 セロックティー 20 せネュアのレート レートのセキュリティーは、証明確を提供するために複数の工にS 統保タイプを使用して保護できます。 F5 / 10000 たのないトラフィック Redirect ・パTTP などのセキュアではないスキームでのトラフィックのボリシー。 証明書 edge および re-encrypt 終戦の TLS 証明書、推定されない場合は、ルーターのデフォルト 証明書が使用されます。 Efge *EM 形式の証明書、ドラッグアンドドロッズ、選択、またはクリップボードから貼り付けてファイルをアップロードします。
パペ / ルーターがトラフィックをサービスにルーティングするために監視するパス。 ターグットボート 10080 トラフィックのターゲットボート。 セキュフタルート ルートのセキュリティーム 図 セキュアなルート ルートのセキュリティーム 図 セキュアなルート レートのセキュリティーム Edge 全なないトラフィック Redirect ************************************
/ ルーターがトラフィックをサービスにルーティングするために監視するバス、 クーグットボート 10080 トラフィックのターグットボート。 セキュリティー 図 セキュフなルート ルートのセキュリティーは、証明書を提供するために複数のTLS 装端タイプを使用して保護できます。 TLS装備 Edge をなないトラフィック Redirect TTTP などのセキュアではないスキームでのトラフィックのポリシー。 証明書 rtdge および re-encrypt 終端の TLS 証明書。指定されない場合は、ルーターのデフォルト証明書が使用されます。 Efge *EM 形式の証明書。ドラッグアンドドロッズ、選択、またはクリップボードから貼り付けてファイルをアップロードします。
ルーターがトラフィックなサービスにルーティングするために監視するパス。 ターグットボート 10080 トラフィックのターゲットボート。 セキュリティー 図 セキュフなルート ルートのセキュリティーは、証明書を提供するために複数のTLS終端タイプを使用して保護できます。 TS基礎 Edge Edge Redirect 1TTP などのセキュアではないスキームでのトラフィックのポリシー。 証明書 edge および re-encrypt 終端の TLS 証明書。指定されない場合は、ルーターのデフォルト証明書が使用されます。 E時書 PEM 形式の証明書。ドラッグアンドドロッズ、選択、またはクリップボードから貼り付けてファイルをアップロードします。
ウイツトボート 10080 トラフィックのターグットボート。 セキュワタレート レートのセキュリティーは、証明書を提供するために複数のTLS 接端タイプを使用して保護できます。 TLS接端 Edge をなないトラフィック Redirect HTTP などのセキュアではないスキームでのトラフィックのポリシー。 証明書 edge および re-encrypt 終端の TLS 証明書。指定されない場合は、ルーターのデフォルト証明書が使用されます。 Emplane PEM 形式の証明書。ドラッグアンドドロップ、選択、またはクリップボードから貼り付けてファイルをアップロードします。
10080 トラフィックのターグットボート。 セキュリティーム 2 セキュアなルート ルートのセキュリティーは、証明書を提供するために複数のTLS 統領タイプを使用して保護できます。 TLS 結場 Edge をなないトラフィック Redirect HTTP などのセキュアではないスキームでのトラフィックのポリシー。 証明書 edge および re-encrypt 終端の TLS 証明書。指定されない場合は、ルーターのデフォルト証明書が使用されます。 Emminiation PEM 形式の証明書。ドラッグアンドドロッズ、選択、またはクリップボードから貼り付けてファイルをアップロードします。
トラフィックのターグットボート。 セキュリティー ② セキュアなルート ルートのセキュリティーは、証明書を提供するために複数のTLS終端タイプを使用して保護できます。 TLS 終端 Edge Edge を全なないトラフィック Redirect HTTP などのセキュアではないスキームでのトラフィックのボリシー。 証明書 edge および re-encrypt 終端のTLS 証明書。指定されない場合は、ルーターのデフォルト証明書が使用されます。 E明書 =PEM 形式の 証明書。ドラッグアンドドロップ、選択、またはクリップボードから貼り付けてファイルをアップロードします。
はキュリティー 図 セキュアなルート ルートのセキュリティーは、証明書を提供するために複数の115 統領タイプを使用して保護できます。 T.5装備 Edge を会でないトラフィック Redirect HTTP などのセキュアではないスキームでのトラフィックのポリシー。 証明書 edge および re-encrypt 鉄端の TLS 証明書。指定されない場合は、ルーターのデフォルト証明書が使用されます。 Eff書 PEM 形式の証明書。ドラッグアンドドロッズ、選択、またはクリップボードから貼り付けてファイルをアップロードします。
セキュアネルート ルートのセキュリティーは、証明書を提供するために複数のTLS 終端タイプを使用して保護できます。 TLS 終端 Edge Edge Redirect HTTP などのセキュアではないスキームでのトラフィックのポリシー。 証明書 edge および re-encrypt 終端の TLS 証明書。指定されない場合は、ルーターのデフォルト証明書が使用されます。 EFM 形式の証明書。ドラッグアンドドロッズ、選択、またはクリップボードから貼り付けてファイルをアップロードします。
図 セキュアなルート ルートのセキュリティーは、証明書を提供するために複数のTLS終端タイプを使用して保護できます。 TLS 経営 Edge をなないトラフィック Redirect HTTP などのセキュアではないスキームでのトラフィックのポリシー。 証明書 tdge および re-encrypt 終端の TLS 証明書。指定されない場合は、ルーターのデフォルト証明書が使用されます。 EF明書 PEM 形式の証明書。ドラッグアンドドロッズ、選択、またはクリップボードから貼り付けてファイルをアップロードします。
ルートのセキュリティーは、証明書を提供するために複数のTLS 終端タイプを使用して保護できます。 TLS 終端 Edge Edge Redirect ITTP などのセキュアではないスキームでのトラフィックのポリシー。 証明書 edge および re-encrypt 終端の TLS 証明書。指定されない場合は、ルーターのデフォルト証明書が使用されます。 E明書 PEM 形式の証明書。ドラッグアンドドロッズ、選択、またはクリップポードから貼り付けてファイルをアップロードします。
TLS 終端 Edge Eddrect HTTP などのセキュアではないスキームでのトラフィックのポリシー。 証明書 edge およびre-encrypt 終端の TLS 証明書。激定されない場合は、ルーターのデフォルト証明書が使用されます。 Emp書 PEM 形式の証明書。ドラッグアンドドロップ、選択、またはクリップポードから貼り付けてファイルをアップロードします。
TLS 終端 Edge Edge State Redirect HTTP などのセキュアではないスキームでのトラフィックのポリシー。 証明書 edge および re-encrypt 終端の TLS 証明書。類定されない場合は、ルーターのデフォルト証明書が使用されます。 E明書 PEM 形式の証明書。ドラッグアンドドロップ、選択、またはクリップポードから貼り付けてファイルをアップロードします。
Edge gge takur トラフィック Redirect MTTP などのセキュアではないスキームでのトラフィックのポリシー。 証明書 edge および re-encrypt 終端の TLS 証明書。指定されない場合は、ルーターのデフォルト証明書が使用されます。 E明書 PEM 形式の証明書。ドラッグアンドドロップ、選択、またはクリップポードから貼り付けてファイルをアップロードします。
安全でないトラフィック Redirect MTTP などのセキュアではないスキームでのトラフィックのポリシー。 証明書 edge および re-encrypt 終端の TLS 証明書。指定されない場合は、ルーターのデフォルト証明書が使用されます。 E明書 PEM 形式の証明書。ドラッグアンドドロップ、選択、またはクリップポードから貼り付けてファイルをアップロードします。
Redirect HTTP などのセキュアではないスキームでのトラフィックのポリシー。 証明書 edge および re-encrypt 終端の TLS 証明書。指定されない場合は、ルーターのデフォルト証明書が使用されます。 E明書 PEM 形式の証明書。ドラッグアンドドロップ、選択、またはクリップポードから貼り付けてファイルをアップロードします。
Redirect HTTP などのセキュアではないスキームでのトラフィックのポリシー。 証明書 edge および re-encrypt 終端の TLS 証明書。指定されない場合は、ルーターのデフォルト証明書が使用されます。 E明書 PEM 形式の証明書。ドラッグアンドドロップ、選択、またはクリップポードから貼り付けてファイルをアップロードします。
HTTP などのセキュアではないスキームでのトラフィックのボリシー。 証明書 edge および re-encrypt 終端の TLS 証明書。指定されない場合は、ルーターのデフォルト証明書が使用されます。 証明書 PEM 形式の証明書。ドラッグアンドドロップ、選択、またはクリップポードから貼り付けてファイルをアップロードします。
証明書 edge およびre-encrypt 終端の TLS 証明書。指定されない場合は、ルーターのデフォルト証明書が使用されます。 E明書 PEM 形式の証明書。ドラッグアンドドロップ、選択、またはクリップポードから貼り付けてファイルをアップロードします。
edge および re-encrypt 終端の TLS 証明書。指定されない場合は、ルーターのデフォルト証明書が使用されます。 証明書 PEM 形式の証明書。ドラッグアンドドロップ、選択、またはクリップポードから貼り付けてファイルをアップロードします。
edge および re-encrypt 終端の TLS 証明書。指定されない場合は、ルーターのデフォルト証明書が使用されます。 証明書 PEM 形式の証明書。ドラッグアンドドロップ、選択、またはクリップポードから貼り付けてファイルをアップロードします。
EBJE および refersing R R M FLS 単分量。 M E E R R R R R R R R R R R R R R R R R
証明書 PEM 形式の証明書。ドラッグアンドドロップ、選択、またはクリップボードから貼り付けてファイルをアップロードします。
PEM 形式の証明書。ドラッグアンドドロップ、選択、またはクリップボードから貼り付けてファイルをアップロードします。
PEM 形式の証明書。ドラッグアンドドロップ、選択、またはクリップボードから貼り付けてファイルをアップロードします。
作成キャンセル

図5.5 コンテナイメージデプロイの入力値(「イメージのデプロイ」画面・後半)

さらに、「イメージのデプロイ」画面で**表5.1**のように必要な情報を入力してから [作成] ボタンを クリックします。 Part 1

入門編



表5.1 「イメージのデプロイ」 画面の設定

設定項目		設定値				
イメージ		[外部レジストリーからのイメージ名] を選択し、「quay.io/openshifthomeroom /workshop-terminal:3.4.3」と入力する				
	ランタイムアイコン	[openshift] を選択する				
	アプリケーション名	workshop-terminal-app				
	名前	workshop-terminal				
リソース						
		[アプリケーションへのルートの作成] にチェックを入れる				
詳細オプション		「名前をクリックして、詳細オプションにアクセスします。 ルーティング, ヘル スチェック, デプロイメント, スケーリング, リソース制限 and ラベル.」の中の 「ルーティング」という文字列をクリックすると、図5.5のようにルーティング関 連の設定欄が表示される				
ホスト名		workshop-terminal-my-project.apps.democluster.ocpdemo.aoshima.me				
	パス	/				
ルーティング	ターゲットポート	10080(公開したい Service を選択する。 クリックすると Pod に紐づいた Service が表示されるので、 それを選択する)				
	セキュリティー	[セキュアなルート] にチェックを入れる				
	TLS終端	[Edge] を選択する				
	安全でないトラフィック	[Redirect] を選択する				
	証明書	空欄のまま				

入力内容について説明します。

- イメージ:外部レジストリからサンプルアプリとして動くコンテナイメージを取得するように指定しています。「ランタイムアイコン」は、トポロジー画面(図5.6)で表示されるアイコンなので何を指定してもかまいませんが、アプリケーションの中身がわかりやすいものにしておいたほうがよいでしょう。
- 一般:作成されるリソース(Deployment、Service、Routeなど)のラベルと名前を指定しています。[アプリケーション名]がリソースのラベルで、[名前]がリソース名です。
- リソース:アプリケーションを Deployment と Deployment Config のどちらでデプロイするのか 指定しています。なお、Deployment Config は OpenShift 固有のリソースなので、Kubernetes に は存在しません。Deployment と Deployment Config はとても似た機能なので、今回のサンプル アプリケーションではどちらを選択してもかまいません^[2]。また、普段アプリケーションをデプロ イするときは、Kubernetes との移植性を考慮して、Deployment Config 固有の機能がどうしても 必要なとき以外は Deployment を選択したほうがよいでしょう。

 ^{【2】} 機能の違いについては、以下のオンラインドキュメントを参照してください。
 Comparing Deployment and DeploymentConfig objects
 https://docs.openshift.com/container-platform/4.9/applications/deployments/what-deployments-are.html#deployment
 ts-comparing-deploymentconfigs_what-deployments-are

●詳細オプション、ルーティング:ルートの設定をしています。ルートはクラスタ外部にコンテナア プリケーションを公開するために必要です。通信を暗号化する場合は、TLS終端をどこにするか や使用するサーバー証明書などを設定します。今回は[TLS終端]に[Edge]を選択しているの で、暗号化はRouterまでで終了し、Routerからコンテナへの通信は暗号化されていないHTTP になります。また、公開するサービスに独自の証明書が必要な場合は、[証明書]で設置する証 明書の設定をしてください。[証明書]で何も設定しなかった場合は、第2章の「2.2 サーバー証 明書の設置」で設定したデフォルト証明書が使用されます。今回はデフォルト証明書を使うので 何も設定していません。

5.3.2 トポロジー画面

Developer 画面の左側のメニューからトポロジー画面に遷移することができます。トポロジー画面 (図5.6)では、リソースの状態や設定内容また、リソース同士の関係などを視覚的にわかりやすく確 認することができます。また、リソースの設定値の変更なども行うことができます。

トポロジー画面で各リソース名をクリックすると、画面右側にリソースの情報が表示されます。た とえば、図5.6のように [D workshop-terminal] (DはDeploymentの略) \rightarrow [リソース] と進むと、 Deploymentの workshop-terminal に紐づいたリソースが表示されます。

E Container Platform		O O user01 -
♦ Developer	プロジェクト: my-project ▼ アプリケーション: すべてのアプリケーション ▼	● ショートカットの表示 📰
+追加	↓ オブションの表示 ▼ リソースでフィルター ▼ 名前で検照	
(+ж¤У-		workshop-terminal
確認		4 + 1175-114
検索		マリンテナー workshop-terminal には、アプリケーションが正常に実行されていることを確認するためのヘルスチェックはありません。ヘルスチェックの
ย่นห		38,00
Helm		詳細 (リソース) 確認
プロジェクト		Pod
設定マップ		workshop-terminal- ⑦ Running ログの表示
シークレット		6b54bb77f5-jcgqt
		サービス
	worksh_minal (6) worksh_artigop	workshop-terminal サービスポート: 10080-tcp → Pod ポート: 10080
		$\mu - \mu$ $\mu - \mu \sigma URL$
		CD workshop-terminal
		https://workshop-terminal-my- project.apps.democluster.ocpdemo.aoshima.me/ @
	Q , Q , X C ×1 ×2	

図5.6 トポロジー画面

Part 1

入門編



図5.6のリソース一覧([リソース])の[ルート]からルートのURLをクリックし、デプロイしたター ミナルアプリケーションが外部公開されていることを確認します。図5.7のように表示されれば成功 です。



図5.7 ターミナルアプリケーション

■CLIでリソースを確認

続いて、作成されたリソースをCLIで確認してみましょう。

まずは、対象のプロジェクト (my-porject) のリソースの閲覧権限を持つユーザーでクラスタにロ グインします。

\$ oc login https://api.democluster.ocpdemo.aoshima.me:6443 -u user01

現在使用中のプロジェクトは、oc project コマンドで確認できます。

\$ oc project Using project "my-project" on server "https://api.democluster.ocpdemo.aoshima.me:6443".

複数プロジェクトの操作権限を持つユーザーを使っている場合は、my-project以外のプロジェクトがアクティブになっているかもしれません。その場合はoc projectコマンドでmy-projectをアクティブにしてください。

\$ oc project my-project

これで、my-project プロジェクト内のリソースが確認できるようになりました。oc get コマンド で、Deployment、Replicaset、Pod、Service、Route を確認してみましょう。

<pre>\$ oc get deployment NAME workshop-terminal</pre>	READY 1/1	UP-TO-[1	DATE AVA 1	ILABLE	AGE 28m
<pre>\$ oc get replicaset NAME workshop-terminal-6b54bb</pre>	o77f5	DESIRED 1	CURRENT 1	READY 1	AGE 28m

\$ oc get pod NAME READY AGE STATUS RESTARTS workshop-terminal-6b54bb77f5-g4cjq 1/1 Running 0 29m \$ oc get service TYPE CLUSTER-IP NAME EXTERNAL-IP PORT(S) AGE workshop-terminal ClusterIP 172.30.57.108 <none> 10080/TCP 29m \$ oc get route NAME HOST/PORT Ð PATH SERVICES TERMINATION WILDCARD PORT workshop-terminal workshop-terminal-my-project.apps.democluster.ocpdemo.aoshima.me 🔊 workshop-terminal 10080-tcp edge/Redirect None

CLIからもリソース情報を確認することができました。



Chapter



OCPのインストール方法



Part 2 > インフラ実践編 CHAPTER 6 OCPのインストール方法

第2章「環境構築」では、IPIでOpenShift Container Platform(以下、OCP)のクラスタをAWS環境にインストールする手順を紹介しました。OpenShift は他にも様々なプラットフォームに導入可能で、複数のインストール方法をサポートしています。

OCPでOpenShiftクラスタをインストールする手順は、おおまかに以下の3つのステップに分けられます。

1. プラットフォームとインストール方法の選択

2. クラスタの設計

3. インストールの実施: UPI (User Provisioned Infrastructure) インストール

本章では、各ステップで考慮して決定すべきことを説明し、最後に最も汎用性の高いインストール 方法 (Any Platform インストール) でクラスタを構築する手順を紹介します。

6.1 プラットフォームとインストール方法の選択

はじめに、クラスタを構築するプラットフォームを選択します。このとき、インストール方法も一緒 に選択します。OCPがサポートするプラットフォームは、パブリッククラウドからオンプレミスまで幅 広いのですが、インストール方法によって利用できるプラットフォームが異なります。また、既存のイ ンフラ環境を利用する場合は、環境の制約によってインストール方法が限定されてしまうこともあり ます。このように、プラットフォームとインストール方法は互いに影響を与えるため、同時に選択す ることになります。

OCPのインストール方法は、IPIとUPIの2つに大別されます。ここではそれぞれのインストール 方法の特徴を説明します。

6.1.1 IPI (Installer-Provisioned Infrastructure)

IPIは、クラウドプロバイダーが提供する基盤構築用のAPIを使用してインストールする方法です。 ノードの作成、ロードバランサーの設定、DNSの設定など、OpenShiftクラスタに必要となるコン ポーネントを自動で作成し、構築します。

IPIを選択した場合の事前準備として、基盤構築用のAPIを使用するためクラウドプロバイダーの アカウントが必要になります。さらに、OpenShiftで使用するドメインを取得し、そのドメインをプラッ トフォームのDNSに登録しておく必要があります。

IPIでインストーラーを実行すると、自動で指定した数のノードをデプロイし、DNSサーバーも構成します。このため、ほぼ全自動で簡単にクラスタを構築できます。

その一方で、IPIでは構成は固定的で、細かなカスタマイズはできません。また、IPIが利用できな いプラットフォームもあります。この場合は、次に説明するUPIによるOpenShiftのインストールを行 うことになります。

6.1.2 UPI (User-Provisioned Infrastructure)

UPIは、ユーザー自身がクラスタの構築に必要となるインフラを準備し、その上にソフトウェアとしてOCPをインストールする方法です。UPIでは、ユーザーが物理サーバーもしくは仮想サーバーを 用意してOSをインストールし、さらにDNSの設定やロードバランサーなどの環境を設定します。



UPIでもIPIと同じく、プラットフォームによってはクラウドプロバイダーのユーザーアカウントが求められますが、これはインストール完了後のクラスタでアカウントの認証情報を利用することでクラスタの管理をしやすくするためで、クラウドプロバイダーのユーザーアカウントが必須というわけではありません

UPIには、クラスタを構築する環境に即してカスタマイズできるメリットがあります。たとえば、既存のインフラ環境でクラスタを構築したいものの、ネットワークに制約があってIPIではインストールできない場合でも、UPIでインストールすることができます。ただし、ユーザーが事前に環境を整備する必要があるため、IPIと比べて手間はかかります。

COLUMN

クラウドプロバイダーとは?

「クラウドプロバイダー」と聞くと、AWSやMicrosoftなどのパブリッククラウドサービスの事業者を 思い浮かべる方が多いかもしれません。しかし OpenShiftでは、「クラスタの基盤を構築するのための API を提供する laaS プラットフォーム」と捉えるのが適切です。したがってオンプレミスの環境であっても、 OpenStack はインフラ構築用 APIを提供するクラウドプロバイダーであり、VMware も vCenter Server がインフラ構築用 APIを提供するためクラウドプロバイダーとして捉えられます。

また、インストール時に使用するクラウドプロバイダーのアカウントの認証情報は、構築後のクラスタで 稼働する「Cloud Credential Operator」によって管理され、プラットフォームに関係するリソースを扱う 際に活用されます。



6.1.3 サポートするプラットフォームとインストール方法

執筆時点の最新版である OCP 4.9 では、表6.1 に挙げている各プラットフォームにインストールす ことができます。プラットフォームのバージョンによって条件があることもあるため、必ず OpenShift Container Platform のオンラインドキュメントで、選択するプラットフォームでのインストールの前提 条件を確認するようにしてください。

 OpenShift Container Platform 4.9 Documentation https://docs.openshift.com/container-platform/4.9/welcome/index.html

プラットフォーム	IPI	UPI
Amazon Web Services (AWS)	0	0
Google Cloud Platform (GCP)	0	0
Microsoft Azure	0	0
Microsoft Azure Stack Hub	_	0
Red Hat OpenStack Platform (RHOSP)	0	0
Red Hat Virtualization (RHV)	0	0
VMware vSphere	0	0
VMware Cloud (VMC) on AWS	0	0
IBM Z or LinuxONE	_	0
IBM Power Systems	_	0
Bare metal	0	0

表 6.1 OpenShift Container Platform 4.9 がサポートするプラットフォーム



OpenShift クラスタのすべてのノードでOSが稼働します。サポートしているOSは次の2つです^[1]。

- Red Hat Enterprise Linux CoreOS (RHCOS)
- Red Hat Enterprise Linux (RHEL) ^[2]

クラスタ構築後、Windows Containerを利用する場合は、Windows Serverが稼働するノードを使います。Windows Serverはクラスタインストール時には使用できないため、ここでは省略しています。

^[2] RHEL 8のWorkerノードがサポートされます。RHEL 7のWorkerノードは非推奨です。 参考: OpenShift Container Platform 4.9 release notes; Red Hat Enterprise Linux (RHEL) 8 now supported for compute machines https://docs.openshift.com/container-platform/4.9/release_notes/ocp-4-9-release-notes.html#ocp-4-9-machine-api -rhel8 OpenShift Container Platform 4.9 release notes; Deprecated and removed features https://docs.openshift.com/container-platform/4.9/release_notes/ocp-4-9-release-notes.html#ocp-4-9-machine-api -rhel8

https://docs.openshift.com/container-platform/4.9/release_notes/ocp-4-9-release-notes.html#ocp-4-9-deprecatedremoved-features

Master ノードと Bootstrap ノードの OS は RHCOS であることが必須で、RHEL はサポートしていません。

Worker ノードと Infra ノードは、サーバーに接続される外部機器が独自のドライバーを必要とするなど、RHCOSでは対応できない場合は RHELを使うことができます。

このようにOSの第1候補はRHCOSであり、特別な事情がない限りすべてのノードでRHCOSを使うことを推奨します。

6.1.5 RHCOSのコンポーネント

RHCOSはOpenShiftクラスタでのみ使える、OpenShift環境専用のOSです。RHCOSはRHELを ベースに作られているため、RHCOSのコンポーネントはRHELとほぼ同じもので構成され、RHEL と同様に管理されます。たとえば、RHCOS内部で稼働するソフトウェアはRPMパッケージで配布 されているものと同じで、これらのソフトウェアはRHELカーネルと systemdで管理される一連の サービスとともに稼働します^[3]。また、サポートする品質やセキュリティや管理の基準はRHELと同 じであるため、同等の信頼性を持つと言えます。セキュリティについては、第10章「セキュリティ」も 参照してください。

6.1.6 Ignition による自動構成

RHCOSには「Ignition」というユーティリティがあり、RHCOSのインストールや起動後の構成作 業を自動で実行できます。構成作業の内容は「Ignition Config」と呼ばれるファイルに記述して、こ れをRHCOSの起動時に読み込ませることによって自動構成を実現します。

UPIではOpenShiftインストーラーを使ってMasterノード、Workerノード、Bootstrapノードのそ れぞれ向けのIgnition Configを作成し、各ノードでRHCOSを起動する際に投入します。

Machine Config Operatorによる管理

RHCOSはOpenShiftクラスタのMachine Config Operatorによってリモートで管理されます。たと えばRHCOSのアップグレードは、OpenShiftクラスタのアップデートに合わせて Machine Config Operatorが行います。管理者がRHCOSを単独でアップグレードすることはありません。

運用中のRHCOSはMachine Config Operatorで管理されるため、管理者はOpenShiftクラスタの み管理するだけで済みます。OpenShiftクラスタとRHCOSの両者を管理する必要はありません。 Part

2

インフラ実践編

^{【3】} ただし、RHCOSはコンテナの実行に特化しているため、任意のRPM形式のソフトウェアをインストールすることはできないので注意しましょう。





サポートするプラットフォームを示した表6.1の最下行に、「Bare metal」があります。Bare metalは 物理サーバーの意味ですが、このBare metalインストールの方法は、物理サーバー環境だけでなく 他のあらゆるプラットフォームでも適用できる、最も汎用性の高い "Any Platform" なインストール 方法です。

OCPのオンラインドキュメントにも「Installing on any platform」というインストール方法^[4]が記載されていますが、これはBare metal UPIインストールの手順とほぼ同じです。ここでは便宜上、この方法を「Any Platform インストール」と呼ぶことにします。

なお、本章第3節では、このAny Platformインストールの方法でOpenShiftクラスタを構築する手順を紹介します。

■Any Platform インストールの特徴

Bare metalを除く他のプラットフォームのインストール方法では、クラウドプロバイダーが必要となります。しかし Bare metal および Any Platform インストールでは必要ありません。つまり、物理サーバー環境のように、クラウドプロバイダーが存在しない、もしくはクラウドプロバイダーの API が利用できない環境であっても、OpenShift クラスタを構築できる方法が、Any Platform インストールです。

たとえば、VMwareの仮想基盤でクラスタを構築する場合、vCenter Server にアクセスするユー ザーアカウントがないとvSphere IPI/UPIでインストールはできません。また、vCenter Serverのユー ザーアカウントがあったとしても、vSphere IPI/UPIが必要とするユーザー権限が与えられていない 場合でも同様です。このような場合は、Any Platform インストールでクラスタを構築できます。

■Any Platformインストールの制約

前述のように、Any Platform インストールではクラウドプロバイダーの APIを使用しません。した がって、インストール完了後のクラスタの運用管理でもクラウドプロバイダーの APIを使うことはで きません。そのため、運用保守を行う場合でも、プラットフォームに関係する便利な機能を使うこと ができなくなります。

たとえば、MachineSetでノードを簡単に増減させたり、ノード障害時に自動復旧するようにしたり はできないため、手作業でのノード管理が必要となります。また、永続ストレージについても、プラッ トフォームが持つストレージサービスを利用することができないため、管理者が手作業でバックエン ドストレージを設定しなくてはなりません。

 ^[4] https://docs.openshift.com/container-platform/4.9/installing/installing_platform_agnostic/installing-platform-agnostic. html

このように、Any Platform インストールはクラウドプロバイダーを利用しない方法と比べて、運用 フェーズに入ってからの手作業が多く残ってしまうことがあります。



6.2.1 設計のポイント

プラットフォームとインストール方法が決まったら、次は構築するOpenShiftクラスタを設計しま す。OpenShiftクラスタは様々な要素が含まれています。代表的なコンポーネントとして、「ノード」 「ネットワーク」「ストレージ」の3つがあります。

本章ではこの3つと、他に必要となるサービスの設計について説明します。



ノードはOpenShiftクラスタを構成する主要コンポーネントで、OCPのソフトウェアが稼働する物 理サーバーや仮想マシン、クラウドインスタンスを指します。

ノードは「Masterノード」と「Workerノード」の2つに大別されます。それぞれOpenShiftクラス タの中で担う役割や必要となる数が異なります。

■Masterノード

MasterノードはOpenShiftクラスタに必須のノードで、クラスタ全体にわたってノードやPodをは じめとするリソースを管理する役割を持ちます。Masterノードの主な機能として以下のものがありま す。

API Server

OpenShiftクラスタの操作は、すべて API によってコントロールされます。その API を公開し ているのが API Server です。API はクラスタ内だけでなく、クラスタ外部にも公開されていて、 OpenShift Web コンソールや CLI (oc コマンド) などによるクラスタ外部からの操作も可能とし ています。

• OAuth Server/OAuth API Server

クラスタ外部からのAPIリクエストを受ける際には、リクエストを送るユーザーが何者かを確認

Part 2

インフラ実践編

(CHAPTER () 6) OCPのインストール方法

する必要があり、さらにそのユーザーが適切な権限を持つかを確認する必要があります。OAuth Server と OAuth API Server はこの認証と認可の機能を提供しています。

etcd

Part 2 > インフラ実践

OpenShiftクラスタでは、クラスタの構成や設定、各種リソース、アカウント/ロールなどのほ ぼすべての情報を、「etcd」という分散型のKVS(Key-Value Store)に保管します。etcdは複数 のMasterノードで稼働し、ノード間で常時データを複製して保管しています。etcdでは、分散し たetcdインスタンスの間で合意をとってクラスタ情報の変更を管理しています。この合意形成の ために etcd は奇数個のインスタンスが必要とされ、これが後述する Master ノードの台数の要件 に影響を与えます。なお、偶数個のインスタンスの場合は、インスタンス間で通信できなくなっ た際に「スプリットブレイン」という何が正しいかを判断できない状態に陥る可能性があるため、 避けるべきです。

Scheduler

Schedulerは、Podを稼働させるノードを決定する役割を持ちます。クラスタ内で新しく作成される Podを監視し、Podが稼働するために最適なノードを選択します。どのように最適なノードを 決めるかは、第11章「Day 2オペレーション」の「11.4 メンテナンス」で詳述しています。

Controller Manager

OpenShiftには様々なリソースがありますが、各リソースには自らの状態を監視する Controller というプロセスがあります。Controller は各リソースに問題が発生するとそれを検知し、継続し て稼働するにおいて好ましい状態になるよう対応を行います。この各 Controller をまとめて単 ープロセスとして実行しているのが Controller Manager です。

Masterノードのシステム要件

Masterノードとなるサーバーには、表6.2のような最小システム要件があります。Masterノードは etcdを構成するため3台必要になります。Masterノードの台数は、3台より多くても少なくてもサポー トされません。ちょうど3台がサポートされる台数であるため、注意してください。

表 6.2 Master ノードの最小システム要件

台数 ^{※1}	OS	CPU (コア) ^{※2}	メモリ (GB)	ストレージ (GB)
3	RHCOS	4	16	120

※1 技術的には1台でも稼働しますが、サポート要件上複数台が求められます。

※2 SMT (同時マルチスレッディング)が利用できる CPU の場合は、1 スレッド = 1 コアと数えてかまいません。

■Workerノード

Worker ノードは OpenShift クラスタに必須のノードで、アプリケーションの Pod を稼働させるという OpenShift の中核をなす役割を持ちます。Worker ノードは、アプリケーション Pod に CPU やメモ
リといったコンピュートリソースを提供するため、Workerノードの数とリソースの量はクラスタ全体の規模を反映します。それゆえ Workerノードはクラスタ設計において重要な項目です。

また、WorkerノードではOpenShiftが提供するコンポーネント (OpenShift Logging、OpenShift Monitoring など) の Pod も稼働します。

Workerノードのシステム要件

Workerノードとなるサーバーには、表6.3のような最小システム要件があります。Workerノードは、アプリケーションの単一障害点とならないために2台以上が必要です。

表 6.3 Worker ノードの最小システム要件

台数 ** 1	OS	CPU (コア) ^{※2}	メモリ (GB)	ストレージ (GB)
2以上	RHCOS RHEL	2	8	120

※1 技術的には1台でも稼働しますが、サポート要件上複数台が求められます。

※2 SMT (同時マルチスレッディング) が利用できる CPU の場合は、1 スレッド = 1 コアと数えてかまいません。



Masterノードでのユーザーアプリケーションの稼働

一般的に、Master ノードではユーザーアプリケーションのPodをスケジュールしないよう にします。しかし、エッジコンピューティングなどサーバーの台数や空間に制約があるユース ケースでは、ノードの数を抑える目的で Master ノードにユーザーアプリケーションのPodを スケジュールするように設定できます。その場合、ノードは Master と Worker の両方の機能を 果たすことになるため、コンピュートリソースを豊富に搭載するようにしてください。

■特殊なノード

Infraノード

Infra ノードは Infrastructure ノードの略で、Worker ノードの一種です。Worker ノードでは OpenShiftが提供するコンポーネントも稼働しますが、これらのコンポーネントを専用で稼働させる ノードが Infra ノードです。具体的には、次の4つを Infra ノードで稼働させることができます。

- OpenShift Logging
- OpenShift Monitoring
- OpenShift Registry
- Ingress Router

Infraノードを用意しておくと、Workerノードでこれらのコンポーネントにコンピュートリソースを 取られて、本来稼働させたいユーザーアプリケーションに割り当てるリソースが圧迫されるのを回避



できます。

可能であればInfraノードを用意することをおすすめしますが、必須ではありません。Infraノードのためのサーバーを用意することが難しいケースも考えられるので、ケースバイケースで検討してください。



図 6.1 各ノードの役割分担

Bootstrapノード

Bootstrapノードは、OpenShiftクラスタをインストールする際に必須となるノードです。クラスタ インストールは、Masterノードを展開して最小限のクラスタを作成し、次にWorkerノードを展開して クラスタに追加する手順を踏みます。Bootstrapノードは、Masterノードを展開するためのクラスタの 設定情報を提供したり、Masterノードにコントロールプレーン^[5]を作成する役割を持ち、Master ノードよりも先に展開されます。Bootstrapノードの詳細な動作についてはGitHubの「OpenShift Installer Overview^[6]」に記載されています。

Bootstrapノードはクラスタの一部ではなく、あくまでMasterノードが展開されてクラスタが作成 されるまでの一時的なコンポーネントです。そのため、一度Masterノードで最小限のクラスタが作成 されるとBootstrapノードは必要なくなるので、削除することができます。

[5] OpenShift Container Platform 4.9 Documentation; The OpenShift Container Platform control plane https://docs.openshift.com/container-platform/4.9/architecture/control-plane.html

[6] OpenShift Installer Overview https://github.com/openshift/installer/blob/release-4.9/docs/user/overview.md



図 6.2 OpenShift クラスタのインストールの流れ

Bootstrap ノードとなるサーバーには、表6.4のような最小システム要件があります。

表 6.4 Bootstrap ノードの最小システム要件

台数	OS	CPU (コア) [※]	メモリ (GB)	ストレージ (GB)
1	RHCOS	4	16	120

※ SMT (同時マルチスレッディング) が利用できる CPU の場合は、1 スレッド = 1 コアと数えてかまいません。



■Masterノード

Master ノードのハードウェアリソースは、管理対象のWorker ノードの数によって変わります。 Master ノードはスケールアウトしたり、CPUやメモリのサイズを変更することができないため、あら かじめクラスタに配備するWorker ノードの最大数を想定してスペックを決めるようにしてください。 参考になる記述がOCPのオンラインドキュメントにあります^[7]。「Control plane node sizing」のセ クションには、表6.5のような表が記載されています(オリジナルは英語表記です)。これはWorker ノードとクラスタのNamespaceの数に対する、Master ノードに搭載する CPUとメモリの対応表で す。これを参考にサイジングしてもよいでしょう。なお、それぞれのNamespace で稼働するワーク ロード(OpenShiftリソース)の前提についてはオンラインドキュメントを参照してください。

^[7] OpenShift Container Platform 4.9 Documentation: [Recommended host practices]内の[Control plane node sizing] https://docs.openshift.com/container-platform/4.9/scalability_and_performance/recommended-host-practices.html# master-node-sizing_



表6.5 Master ノードのリソースサイジング

クラスタ内の Workerノード数	クラスタ内の Namespace数	->	Masterノードの CPU ⊐ア [※]	Master ノードの メモリ (GB)
25ノード	500	\rightarrow	4	16
100ノード	1000	\rightarrow	8	32
250ノード	4000	\rightarrow	16	96

※ SMT (同時マルチスレッディング) が利用できる CPU の場合は、1 スレッド = 1 コアと数えてかまいません。

■Workerノード

Worker ノードのハードウェアリソースは、稼働する Podが求めるリソースに依存します。あらかじ め稼働するアプリケーションがすべてわかっている場合は、アプリケーションが必要とするリソース からクラスタ全体で必要となるリソースをサイジングができます。しかし OpenShift でこういったケー スは稀で、稼働するアプリケーションをすべて事前に特定することが難しい場合が多いでしょう。

この場合は厳密なサイジングは困難なため、様々な仮定を置くことになります。これが確実という 方法論はありませんが、次のようなアプローチで臨むことで、ある程度精度の高いサイジングができ るでしょう。

1. クラスタ全体で稼働するPodの数を仮定する

まず稼働する Podの数を仮定します。たとえば、クラスタ全体で1200 Pod稼働するとします。 OpenShift では1台の Worker ノードで稼働できる Pod は最大 500 です。そのため、「1200÷500 = 2.4」から切り上げて少なくとも3台の Worker ノードが必要とわかります。

実際には、Workerノードに障害が発生する際に別のWorkerノードにPodを再スケジュールでき るようにもっと余裕を持たせます。たとえば、1 Workerノードで稼働させるPodを最大250として、 「1200 ÷ 250 = 4.8」、これを切り上げて「5台」などと見積もります。

2. Podが必要とするリソースを仮定する

Podに必要となるリソースはアプリケーションごとに異なりますが、稼働するアプリケーションが わからない場合はこれも仮定します。

たとえば、1 Podあたり平均して0.25コアのCPUと、0.5 GBのメモリを使うと仮定します。クラス タ全体で1200 Pod稼働するとすれば、すべてのWorkerノードで必要となるリソースは次のように計 算できます。

- CPU: 0.25 × 1200 = 300 コア
- メモリ: 0.5 × 1200 = 600 GB

3. Workerノードの数とリソースを計算して調整する

先に計算した CPU とメモリを5台の Worker ノードで満たすためには、1 Worker ノードに求められるリソースは次のように計算できます。

• CPU: $300 \div 5 = 60 \exists 7$

• メモリ: 600 ÷ 5 = 120 GB

計算の結果が現実的でない数値の場合、それはWorkerノードの数が少ないことを意味します。現 実的なWorkerノードのリソースになるようにWorkerノードの数を調整します。

たとえばWorkerノードの数を20台とすれば、1Workerノードに求められるリソースは次のように なり、先ほどよりは用意しやすいスペックとなります。

- CPU: $300 \div 20 = 15 \exists 7$
- メモリ: 600 ÷ 20 = 30 GB

以上のようなPodの数とリソースを仮定してサイジングする方法は、単純ですが概算としては十分 なレベルの結果を得られるでしょう。

なお、OpenShiftではオーバーコミットが許容されています。上の例ではオーバーコミットを一切 考慮していないので、安全ですがかなり余裕を持たせたサイジングと言えます。この場合、リスクと のバランスを考慮したうえで適切な割合のオーバーコミットを織り込むことで、より現実的な値のリ ソースで構成することが可能です。

ただし、メモリのオーバーコミットは極力避けるべきです。Workerノードでのメモリ不足はOOM Killer (Out of Memory Killer) による Pod の停止につながり、多くの場合致命的な事態になります。 CPUはオーバーコミットしても、メモリはオーバーコミットしないようにすることをおすすめします。



Worker ノードのサイジングのヒントになる情報がOCPのオンラインドキュメントに記載されています。参考にしてください。

OpenShift Container Platform 4.9 Documentation : Planning your environment according to object maximums

https://docs.openshift.com/container-platform/4.9/scalability_and_performance/ planning-your-environment-according-to-object-maximums.html

.....

■Infraノード

InfraノードのハードウェアリソースはMasterノードと同じく、Workerノードの数によります。また こちらもWorkerノードの数に対してInfraノードに搭載するCPUとメモリの参考数値がOCPのオン Part 2

インフラ実践編

6

ラインドキュメント^[8]に記載されています(表6.6、オリジナルは英語表記です)。

クラスタ内の Workerノード数	\rightarrow	Infraノードの CPU コア [※]	Infraノードの メモリ (GB)	Infra ノードの CPU コア with Logging	メモリ (GB) with Logging
25ノード	\rightarrow	4	16	4	64
100ノード	\rightarrow	8	32	8	128
250ノード	\rightarrow	16	128	16	128
500ノード		32	128	32	192

表 6.6 Infra ノードのリソースサイジング

※ SMT (同時マルチスレッディング) が利用できる CPU の場合は、1 スレッド = 1 コアと数えてかまいません。

なおこの表では、OpenShift Loggingだけは別にリソースの確保が求められている点に注意してく ださい。

左から2列目と3列目のInfraノードの「CPUコア」と「メモリ(GB)」は、Monitoring、Registry、 Routerの3つ分だけのリソースです。これらのコンポーネントはクラスタインストールと同時に自動 でインストールされていますが、OpenShift Loggingはクラスタ構築後に管理者が任意でインストー ルするものです。そのため、Loggingだけは別枠で分かれています。

たとえば、Worker ノードが25台以下のクラスタで Monitoring、Registry、Router、LoggingのすべてをInfra ノードで稼働させる場合は、CPUは4+4=8コアを、メモリは16+64=80 GBをそれぞれInfra ノードで確保しておくとよいでしょう。

6.2.4 ネットワーク

次に、OpenShiftクラスタを構築するときに必要となるネットワークについて説明します。ここで は、OpenShiftのネットワークで使われる技術要素の概要のみ取り上げます。ネットワークの詳細に ついては第8章「ネットワーク」を参照してください。

■仮想ネットワーク

クラスタを構成するすべてのノードは、1つのNICから単一のネットワーク(Host Network)に接続することが一般的です。各ノード上ではSDN (Software Defined Networking)のPodが稼働して、 全ノードを横断する仮想ネットワークが作られます。OpenShiftクラスタ内のノード間の通信は、おおむねこの仮想ネットワークを使って行われます。

^[8] OpenShift Container Platform 4.9 Documentation: [Recommended host practices] 内の [Infrastructure node sizing] https://docs.openshift.com/container-platform/4.9/scalability_and_performance/recommended-host-practices.html# infrastructure-node-sizing_

クラスタのインストール時に定義する仮想ネットワークとして、「Cluster Network」と「Service Network」の2つがあります。これらのネットワークのサブネットは、他のネットワークと衝突していない限りは任意です。しかし Pod や Service はクラスタ全体で大量に作られることもあるので、広いレンジにするように注意してください。以下にそれぞれのネットワークの役割と、デフォルトの CIDR (Classless Inter-Domain Routing)を示します。

Cluster Network

Cluster Networkは、クラスタで稼働する Pod にインターフェースを提供し、クラスタ内部の Pod 間 通信を実現するためのネットワークです。Pod には Cluster Networkのレンジにある IP アドレスがダ イナミックに割り当てられます^[9]。

Cluster Network $\mathcal{O}\mathcal{T}\mathcal{T}\mathcal{H}$ CIDR \mathcal{L} [10.128.0.0/14] $\mathcal{C}\mathcal{T}$.

Service Network

Service Networkは、Serviceリソースにインターフェースを提供し、Pod内で稼働するアプリケーションをクラスタ内部/外部に公開する際に使用するネットワークです。ServiceにはService NetworkのレンジにあるIPアドレスがダイナミックに割り当てられます。

Service NetworkのデフォルトCIDRは「172.30.0.0/16」です。

OpenShiftでは、クラスタ内外に公開されるアプリケーションやAPIサーバーのURLのドメイン名 を解決するためのDNSが必要です。そのため、クラスタインストール時にはDNSサーバーが必要と なります。クラスタを構築する環境で使用できる既存のDNSサーバーがあれば、新規で用意しなく ても問題ありません。

DNSサーバーでは、**表6.7**に挙げている6つのドメイン名を解決します。cluster_nameとbase_ domain、およびMasterノードとWorkerノードのホスト名については任意です。

表6.7 DNSサーバーが解決する6つのドメイン名

ドメイン名	レコード	コンポーネント
api. <cluster_name>.<base_domain>.</base_domain></cluster_name>	A/AAAA または CNAME PTR	Kubernetes API
api-int. <cluster_name>.<base_domain>.</base_domain></cluster_name>	A/AAAA または CNAME PTR	Kubernetes API
*.app. <cluster_name>.<base_domain>.</base_domain></cluster_name>	A/AAAA または CNAME	Route
<pre>bootstrap.<cluster_name>.<base_domain>.</base_domain></cluster_name></pre>	A/AAAA または CNAME PTR	Bootstrap
<master><n>.<cluster_name>.<base_domain>.</base_domain></cluster_name></n></master>	A/AAAA または CNAME PTR	Master
<pre><worker><n>.<cluster_name>.<base_domain>.</base_domain></cluster_name></n></worker></pre>	A/AAAA または CNAME PTR	Worker

[9] PodにHost NetworkのIPアドレスを割り当てて直接Host Networkで通信することも可能ですが、これは特殊なケースです。アプリケーション上どうしても必要となる場合を除いて、Podは Cluster Networkを使うことが一般的です。

6



■ロードバランサー

OpenShiftでは、クラスタ内外のアプリケーションやAPI通信のためのロードバランサーが必要で す。そのためクラスタインストール時にはロードバランサーが必要となります。ロードバランサーは、 クラスタを構築する環境で使用できる既存のロードバランサーがあれば、新規で用意する必要はあ りません。

ロードバランサーは、クラスタ内部のノード間通信の負荷分散用と、クラスタ外部から来るアクセスの負荷分散の2種類を必要とします。別々で用意しても、1つにまとめてもどちらでも問題ありません。表6.8のようにポートと対象ノードのエントリをロードバランサーで設定してください。

ポート	対象ノード	クラスタ 内部向け	クラスタ 外部向け	負荷分散の対象
6443	Bootstrap Master	0	0	API Server
22623	Bootstrap Master	0	—	Machine Config Server
443	Worker or Infra (Ingress Routerが稼働するノード)	0	0	Ingress RouterのHTTPSアクセス
80	Worker or Infra (Ingress Routerが稼働するノード)	0	0	Ingress RouterのHTTPアクセス

DHCP

OpenShiftクラスタのノードはすべて Host Network に接続されるため、それぞれに Host Network の IP アドレスを付与する必要があります。このときに DHCP で IP アドレスが割り振られます。クラウドプロバイダーによっては、プラットフォーム 側に DHCP のサービスが存在するものもあるので、具体的な設定方法については、OCP のオンラインドキュメントの各クラウドプロバイダーごとのインストール方法^[10]を参照してください。

次節では「Any Platform インストールの手順」を記載していますが、ここではノードのセットアッ プで iPXE を使用するために DHCP を用意しています。Any Platform インストールでは厳密には DHCP は必須ではなく、IPXEを使わずに ISO イメージからノードをセットアップして静的に IP アドレ スを設定する方法もあります。しかし、非常に手間がかかり間違いも起きやすくなるため、DHCPを使 うことを推奨します。

[10] OpenShift Container Platform 4.9 Documentation: Cluster installer activities https://docs.openshift.com/container-platform/4.9/welcome/index.html#cluster-installer-activities



OpenShiftでは、ステートフルなユーザーアプリケーションには永続ストレージが必要となります が、OpenShiftの基盤コンポーネントである、OpenShift LoggingやOpenShift Registryの構成にも永 続ストレージを必要とします。OpenShift Monitoringは永続ストレージがなくても使うことはできま すが、実際の運用ではメトリクスデータを永続的に保管するために永続ストレージが必要となること がよくあります。

OpenShiftのアーキテクチャにおいて、ストレージは疎結合なコンポーネントであり、OpenShiftク ラスタをインストールする段階では必要ではありません。クラスタインストール完了後に、ストレージ をセットアップするのが一般的です。

ストレージの設計は使用するストレージシステムに依存するところが大きいため、本書では説明は 省略します。OpenShiftの永続ストレージで使われる技術要素やストレージの選択肢については、第 9章「ストレージ」を参照してください。



第2章「環境構築」では、AWS環境にIPIでOpenShiftクラスタをインストールする方法を紹介しましたが、本節ではUPIインストールの方法を紹介します。

すでに何度か述べたように、UPIではノードやDNSなどのインストールに必要なインフラをユー ザー自身が用意する必要があります。UPIは様々なプラットフォームに対応していますが、インフラ を準備するところが異なるだけで、本質的な手順はいずれも変わりません。

そこで本節では、本章第1節で説明した Any Platform インストールを用いて、OpenShift クラスタ を構築する方法を紹介します。この方法は汎用性が高く、インストールに必要なコンポーネントや手 順を理解するのに適しています。

6.3.1 Any Platform インストールの流れ

Any Platform インストールでは、次のような流れでクラスタインストールを行います。

- 1. インストール環境の整備
 - ▶ 作業用マシンの作成
 - ▶ iPXE/HTTPサーバーの作成
 - ▶ DNS/ロードバランサー/DHCPの作成

Part 2

インフラ実践編

6



- ▶ ノードとなるマシンの用意
- 2. OpenShiftインストーラー、コマンドラインツール、RHCOSの入手
- 3. Ignition Configファイルの作成
- 4. Bootstrapノード/Masterノードの作成
- 5. Worker ノード/Infra ノードの作成
- 6. クラスタ作成の確認

以降は、Any Platform インストールの構成および手順を追いながら、おおまかにその説明を行いま す。省略している部分については、GitHub (OpenShift Architecture Design Patterns) で同様の内容 が公開されているので、そちらを参照してください。

• OpenShift Architecture Design Pattern

https://github.com/team-ohc-jp-place/OpenShift-ADP/tree/4.9/BaremetalUPI

本書では図6.3に示している環境を使って、3台のMasterノード、3台のWorkerノード、3台の Infraノードで構成されるクラスタを構築します。



図 6.3 Any Platform インストールで構築するクラスタ環境のシステム構成図



■作業用マシンの作成

まずは、インストール作業を行うマシンを作成します。作業用マシンでは、OpenShiftインストー ラーを実行したり、コマンドラインツールでクラスタに対して oc コマンドを実行したりします。

作業用マシンのスペックに特に指定はありませんが、OpenShiftインストーラーはLinux、macOSの み、コマンドラインツールはLinux、macOS、Windowsのみ使用できます。また、作業用マシンはク ラスタインストール後にノードの追加などの運用管理に利用することもできます。可能であれば、イ ンストール先の環境内にLinuxの作業用マシンを作成しておくことをおすすめします。

■ iPXE/HTTPサーバーの作成

今回の環境では、作業用サーバーにiPXEサーバーとHTTPサーバーをセットアップしています。 クラウドプロバイダーが存在するプラットフォームでのクラスタインストールでは、RHCOSの仮想 マシンイメージを利用してノードでRHCOSを起動できます。しかし、Bare metalへのインストールお よび Any Platform インストールでは仮想マシンイメージが利用できないため、PXE ブートで起動しま す。RHCOSはISO イメージからでもインストールできますが、クラスタインストールに失敗した際に はノードの再インストールが必要となるため、PXE ブートできる環境を整えておくことをおすすめし ます。

また、HTTPサーバーには、RHCOSのインストールイメージと各ノードのIgnition Configファイルを配置します。各ノードがiPXEで起動する際にこれらのファイルを読み込めるように公開しています。

iPXEサーバーとHTTPサーバーのセットアップ手順は省略します。

■DNS/ロードバランサー/DHCPの作成

DNSとロードバランサーはOpenShiftのクラスタが内外で通信するのに不可欠なシステムです。 新規で作成する場合は、冗長構成で作成するようにしてください。すでに環境内に冗長化された DNSとロードバランサーのシステムがあるなら、それらを活用しても問題ありません。

なお、パブリッククラウドの環境でUPIインストールする場合は、クラウドサービスのDNSとロー ドバランサーを利用できるので、作成する必要はありません。

DNSサーバー

今回の環境では既存のDNSサーバーが使えないため、新規に作成したRHEL8サーバーでBIND を使って作成しています。ここではBINDのセットアップおよび冗長化の手順は省略します。セット アップ後には、前節の表6.7に記載したDNSのエントリを登録しておきます。

101

Part 2

インフラ実践編

6

参考までに、前述のシステム構成図に準ずるレコードのエントリは以下となります。

```
$ORIGIN example.localdomain.
$TTL 3600
0
               SOA
                       ns1.example.localdomain. root.example.localdomain.(
       ΤN
       2000091802
                       : Serial
       3600
                       ; Refresh
       900
                       ; Retry
       3600000
                       ; Expire
       3600)
                       : Minimum
;
       ΤN
               NS
                       ns1.example.localdomain.
api-int.ocp49.example.localdomain.
                                    IN A 172.16.0.110
api.ocp49.example.localdomain.
                                    IN A 172.16.0.111
*.apps.ocp49.example.localdomain.
                                    IN A 172.16.0.111
bs.ocp49.example.localdomain.
                                    IN A 172.16.0.11
m1.ocp49.example.localdomain.
                                    IN A 172.16.0.21
m2.ocp49.example.localdomain.
                                    IN A 172.16.0.22
m3.ocp49.example.localdomain.
                                    IN A 172.16.0.23
w1.ocp49.example.localdomain.
                                    IN A 172.16.0.31
w2.ocp49.example.localdomain.
                                    IN A 172.16.0.32
w3.ocp49.example.localdomain.
                                    IN A 172.16.0.33
i1.ocp49.example.localdomain.
                                    IN A 172.16.0.41
i2.ocp49.example.localdomain.
                                    IN A 172.16.0.42
i3.ocp49.example.localdomain.
                                    IN A 172.16.0.43
```

▶ 正引き用設定ファイルの例:/var/named/ocp4.9.example.localdomain.db

▶ 逆引き用設定ファイルの例:/var/named/0.16.172.in-addr.arpa.db

```
$TTL 3600
0
                SOA
                        ns1.example.localdomain. root.example.localdomain.(
        ΤN
        2000091802
                        ; Serial
        3600
                        ; Refresh
        900
                        ; Retry
        3600000
                        ; Expire
        3600)
                        ; Minimum
;
        IN
                NS
                        ns1.example.localdomain.
;
11
     IN PTR bs.ocp49.example.localdomain.
21
         PTR m1.ocp49.example.localdomain.
     ΤN
22
     IN PTR m2.ocp49.example.localdomain.
     IN PTR m3.ocp49.example.localdomain.
23
31
     IN PTR w1.ocp49.example.localdomain.
32
     IN PTR w2.ocp49.example.localdomain.
33
     IN PTR w3.ocp49.example.localdomain.
41
     IN PTR i1.ocp49.example.localdomain.
42
     IN PTR i2.ocp49.example.localdomain.
     IN PTR i3.ocp49.example.localdomain.
43
```

ロードバランサー

ロードバランサーは、クラスタ内部のノード間通信の負荷分散用と、クラスタ外部からのアクセス の負荷分散の2種類が必要になります。これらが別々のシステムに分かれている必要はありません。 図6.3のシステム構成図のように、内部用と外部用を1つの冗長化されたシステム上で別々のIPをア サインする形も可能です。

今回の環境では既存のロードバランサーが使えないため、新規に作成した RHEL8 サーバーで HAProxyを使って作成しています。ここでは HAProxyのセットアップおよび冗長化の手順は省略し ます。セットアップ後には、前節の表6.8 に記載したロードバランサーのエントリを登録しておきます。 参考までに、前述のシステム構成図に準ずるエントリは以下となります。

▶ HA proxy構成ファイルの例:/etc/haproxy/haproxy.cfg

```
# main frontend which proxys to the backends
#------
frontend kubeapi
   default_backend kube_api
   mode tcp
   bind *:6443
frontend machineconfig
   default_backend machine_config
   mode tcp
   bind *:22623
frontend workerhttp
   default_backend worker_http
   mode tcp
   bind *:80
frontend workerhttps
   default_backend worker_https
   mode tcp
   bind *:443
#_____
# round robin balancing between the various backends
backend kube_api
   mode tcp
           roundrobin
   balance
   option
             ssl-hello-chk
   server bs bs.ocp49.example.localdomain:6443 check
   server m1 m1.ocp49.example.localdomain:6443 check
   server m2 m2.ocp49.example.localdomain:6443 check
   server m3 m3.ocp49.example.localdomain:6443 check
```

Part

2

インフラ実践編



```
backend machine_config
    mode tcp
    balance
               roundrobin
    server bs bs.ocp49.example.localdomain:22623 check
    server m1 m1.ocp49.example.localdomain:22623 check
    server m2 m2.ocp49.example.localdomain:22623 check
    server m3 m3.ocp49.example.localdomain:22623 check
backend worker_http
    mode tcp
    balance
               source
    server w1 w1.ocp49.example.localdomain:80 check
    server w2 w2.ocp49.example.localdomain:80 check
    server w3 w3.ocp49.example.localdomain:80 check
    server i1 i1.ocp49.example.localdomain:80 check
    server i2 i2.ocp49.example.localdomain:80 check
    server i3 i3.ocp49.example.localdomain:80 check
backend worker_https
    mode tcp
    balance
               source
    server w1 w1.ocp49.example.localdomain:443 check
    server w2 w2.ocp49.example.localdomain:443 check
    server w3 w3.ocp49.example.localdomain:443 check
    server i1 i1.ocp49.example.localdomain:443 check
    server i2 i2.ocp49.example.localdomain:443 check
    server i3 i3.ocp49.example.localdomain:443 check
```

DHCPサーバー

今回の環境では1台のDNSサーバー上にdhcpdを使って作成しています。DHCPサーバーは冗長 化しても問題ありませんが、DHCPサーバーはクラスタインストール時に必要なだけで、その後はほ ぼ参照される機会がないため、本書ではシングル構成としています。

ここでは dhcpd のセットアップおよび冗長化の手順は省略します。

■ノードとなるマシンの用意

Bootstrapノード、Masterノード、Workerノードのそれぞれに対応するマシンを作成します。OSが インストールされていないドライブを持つ物理マシンもしくは仮想マシンを必要な数だけ準備してく ださい。前述のシステム構成図では、Bootstrapノードが1台、Masterノードが3台、Workerノード が3台、Infraノードが3台で、合わせて10台となります。

Master ノードで最小限のクラスタが作成された後に、Bootstrap ノードとして使っていたマシンを Worker ノードに回すことで、台数を節約することも可能です。

6.3.3 OpenShiftインストーラー、コマンドライン ツール、RHCOSの入手

作業用マシンで、OpenShiftインストーラー、コマンドラインツール、RHCOSのファイルをそれぞ れダウンロードします。

Red Hat OpenShift Cluster Managerの「Create an OpenShift cluster」のページで、「Datacenter」 のタブの下部にある「Platform agnostic (x86_64)」を選びます。

• Red Hat OpenShift Cluster Manager : Create an OpenShift cluster https://console.redhat.com/openshift/create

> Create					
te an OpenShift cluster					
Cloud	Datacenter	묘 Local			
sisted Installer Technology Previ	ew				
ate a cluster on bare metal following	a guided installation process to connect	ur hardware.			
reate cluster					
her datacenter options					
ate clusters on supported infrastruct	ure using our extensive documentation a	are Metal: Select an installation type			
nfrastructure provider	Installation options	(Technology Prev	(ew)	* Recommended	
Bare Metal (x86_64)	Full stack automation and pre-ex			~	
BM Z	Pre-existing infrastructure	((•))	/		
Power	Pre-existing infrastructure	Assisted Bare Meta Install OpenShift on infrastructure with ste guidance.	al Installer your own :p-by-step	Installer-provisioned infrastru Deploy an OpenShift cluster of infrastructure that the installat program provisions and the clu maintains	on ion ister
Red Hat OpenStack	Full stack automation and pre-ex				
Red Hat Virtualization	Full stack automation and pre-ex				
vSphere	Full stack automation and pre-ex		È		
				ad infrastructure	

図 6.4 OpenShift インストーラーとコマンドラインツールのダウンロードページへのアクセス

表示されたページの、「① What you need to get started」にリストされているもののうち、以下の 6つを作業用マシンにダウンロードします。

OpenShift installer (作業用マシンのOSに合わせる)



- Pull Secret
- Command line interface (作業用マシンのOSに合わせる)
- RHCOS kernel
- RHCOS initramfs
- RHCOS rootfs

なお、これらの6つのファイルはダウンロードする時点で最新のバージョンのものがダウンロード されます。これにより、その時点で最新のOpenShiftクラスタがインストールされます。

1	What you need to get started
	OpenShift installer
	Download and extract the install program for your operating system and place the file in the directory where you will store the installation configuration files. Note: The OpenShift install program is only available for Linux and macOS at this time.
	Linux • x86_64 • Download installer
	Appeveloper Preview Download pre-release builds
	Pull secret
	Download or copy your pull secret. You'll be prompted for this information during installation.
	Download pull secret
	Command line interface
	Download the OpenShift command-line tools and add them to your PATH.
	Linux • x86_64 • Download command-line tools
	When the installer is complete you will see the console URL and credentials for accessing your new cluster. A kubeconf ig file will also be generated for you to use with the oc CLI tools you downloaded.
	Red Hat Enterprise Linux CoreOS (RHCOS)
	Download RHCOS to create machines for your cluster to use during installation. Download the installer ISO image, or the kernel, initramfs, and rootfs. Learn more \mathbb{C}^3 .
	Download RHCOS ISO
	Download RHCOS kernel
	Download RHCOS initramfs
	Download RHCOS rootfs

図 6.5 OpenShift インストーラー、コマンドラインツール、RHCOS のダウンロード

ダウンロードした「OpenShift installer」のファイルを解凍すると、openshift-installという実行 ファイルが入手できます。「Command line interface」のファイルを解凍すると、ocとkubectlという 2つの実行ファイルが入手できます。これらのファイルは以後何回も実行するので、パス指定なしで



RHCOSのkernel、initramfs、rootfsの3つのファイルについては、HTTPサーバーで公開するディレクトリに配置しておきます。



6.3.4 Ignition Config ファイルの作成

Bootstrap ノード、Master ノード、Worker ノード用のそれぞれの Ignition Config ファイルを作成し ます。Ignition Config ファイルは、install-config.yaml という YAML から openshift-install コ マンドを使って作成します。

■install-config.yamlの作成

作業用マシンでインストール用ディレクトリを作成し、その中に以下のような install-config. yaml ファイルを作成します。



install-config.yaml

# mkdir -p ~/openshift && cd ~/openshift	
<pre># cat << EOF > ~/openshift/install-config.yaml</pre>	
apiVersion: v1	
baseDomain: example.localdomain	0
compute:	
- hyperthreading: Enabled	
name: worker	
replicas: 0	
controlPlane:	
hyperthreading: Enabled	
name: master	
replicas: 3 —	O
metadata:	
name: ocp49	
networking:	
clusterNetwork:	
- cidr: 10.128.0.0/14	
hostPrefix: 23	
networkType: OpenShiftSDN	Ø
serviceNetwork:	
- 172.30.0.0/16	•••••••••••••••••••••••••••••••••••••••
platform:	
none: {}	•••••••••••••••••••••••••••••••••••••••
fips: false	•••••••••••••••••••••••••••••••••••••••
pullSecret: '{"auths":}'	B
sshKey: 'ssh-ed25519 AAAA'	

EOF

- ベースとなるドメイン名を指定する。
- compute セクションの頭には (ハイフン) が必要だが、controlPlane セクションには (ハイフン) は必要ない。
- SMTを使用する場合はEnableを指定する。
- Worker ノードの数は、この時点では0を指定しておき、あとからWorker ノードを手動で展開する。
- Master ノードの数は3を指定する。
- ⑥ 作成されるクラスタの名前を指定する。●のベースドメインのサブドメイン名として使用される。
- ⑦ Cluster NetworkのCIDRを指定する。
- ③ 各ノードにアサインされるサブネットのPrefixを指定する。
- クラスタで使用するCNI (Container Network Interface)を指定する。デフォルトではOpenShiftSDNが指定される。
- Service NetworkのCIDRを指定する。
- Bare metal インストールおよび Any Platform インストールでは none を指定する。
- D FIPS モードの有効化を指定する。デフォルトは false が指定される。
- 「「…」の部分に、OpenShiftインストーラーと一緒にダウンロードしたPull Secretの中身を転記する。
- 個
 各ノードのRHCOSとSSHで通信するための公開鍵を転記する。

■マニフェストファイルの作成

作成した install-config. yaml から、マニフェストファイルを作成します。

任意の名前のインストール用のディレクトリを作って、先ほど作成した install-config.yamlを 作ったディレクトリにコピーします。以下の例では、installdirという名前のディレクトリを作成し ています。

インストールディレクトリにコピーした install-config.yaml ファイルは、以降の作業で消えてし まうので、オリジナルの install-config.yaml は再利用できるように保管しておきましょう。

mkdir -p ~/openshift/installdir # cp ~/openshift/install-config.yaml ~/openshift/installdir/

次のコマンドでマニフェストファイルが作成されます。

```
# cd ~/openshift
# openshift-install create manifests --dir=installdir
INFO Consuming Install Config from target directory
INFO Manifests created in: installdir/manifests and installdir/openshift
```

インストールディレクトリの下に新しいディレクトリが作成されているのを確認します。このディレ クトリの中にインストール用のマニフェストファイル(YAMLファイル)が生成されているはずです。 確認してみてください。

```
# ls -ltr installdir
total 8
drwxr-x---. 2 root root 4096 Nov 09 15:28 manifests
drwxr-x---. 2 root root 4096 Nov 09 15:28 openshift
# ls -ltr installdir/manifests
total 64
-rw-r-----. 1 root root 169 Nov 09 15:28 openshift-kubevirt-infra-namespace.yaml
-rw-r-----. 1 root root 3833 Nov 09 15:28 openshift-config-secret-pull-secret.yaml
...
# ls -ltr installdir/openshift
total 24
-rw-r-----. 1 root root 1185 Nov 09 15:28 99_openshift-machineconfig_99-master-ssh.yaml
-rw-r-----. 1 root root 2466 Nov 09 15:28 99_openshift-cluster-api_master-user-data-secret.yaml
```

これらのマニフェストファイルを編集することで、OpenShiftクラスタの構成を変更できます。たと えば、デフォルトではMasterノードにはユーザーアプリケーションのPodをスケジュールできません が、installdir/manifests/cluster-scheduler-02-config.ymlを編集することでMasterノードに もユーザーアプリケーションのPodをスケジュールできます。



ここでは特に編集は行わず、デフォルトのまま次に進みます。

■Ignition Configファイルの作成

作成したマニフェストファイルから、Ignition Configファイルを作成します。 次のコマンドでIgnition Configファイルが作成されます。

cd ~/openshift # openshift-install create ignition-configs --dir=installdir INFO Consuming Openshift Manifests from target directory INFO Consuming Worker Machines from target directory INFO Consuming Master Machines from target directory INFO Consuming Common Manifests from target directory INFO Consuming OpenShift Install (Manifests) from target directory INFO Ignition-Configs created in: installdir and installdir/auth

インストールディレクトリの中に、Bootstrapノード、Masterノード、Workerノードのそれぞれの Ignition Configファイル(.ign)が作成されていることがわかります。作成された Ignition Configファ イルは、HTTPサーバーで公開するディレクトリに配置しておきましょう。

<pre># cd ~/openshift</pre>
<pre># tree installdir</pre>
installdir
├─── auth
kubeadmin-password
│ └─── kubeconfig
└─── bootstrap.ign
└─── master.ign
└─── metadata.json
└─── worker.ign
1 directory, 6 files

Ignition Configファイルと同時に、作成する OpenShift クラスタの kubeadmin ユーザーパスワード やコンテキストなど、クラスタの認証に関わるファイルとディレクトリ(auth)も作成されます。 kubeconfigファイルはクラスタが作成されているかocコマンドで確認する際に使われます。この タイミングで export コマンドで KUBECONFIG という環境変数を宣言しておきます。

export KUBECONFIG= openshift/installdir/auth/kubeconfig



Ignition Configファイルの期限切れ

作成されたIgnition Configファイルには、X509形式の証明書が含まれています。この証明 書の期限は24時間になっていて、24時間を超えると期限切れとなり再利用することができま せん。正確に言えば、証明書が期限切れのIgnition Configファイルを使ってノードを起動する ことはできますが、認証が通らないためOpenShiftクラスタを構成できません。 Ignition Configファイルが作成後24時間を超えてしまった場合は、再作成してHTTPサー バーにアップロードすることを忘れないようにしましょう。

6.3.5 Bootstrapノード/Masterノードの作成

それではノードを起動してみましょう。

まずはBootstrap ノードとMaster ノードをすべて起動します。起動の順序やタイミングに制限はないので、Bootstrap ノードと3台のMaster ノードのサーバーを順に起動します。iPXEと Ignition Config ファイルが適切に用意されていれば、サーバーをパワーオンするだけで何もしなくても自動的に起動してきます。

Bootstrap ノードと Master ノードで画面を確認して、ログインコンソールが図 6.6 のように表示さ れていれば起動はできています。エラーメッセージが表示されることがありますが、ここでは無視し て問題ありません。



図 6.6 Bootstrap ノード/Master ノードのログインコンソール

Bootstrap ノードと3台の Master ノードの起動が確認できたら、作業用マシンで次のコマンドを実行して Bootstrap のプロセスをモニターします。このコマンドが終了すれば Bootstrap のプロセスは完了し、Bootsrtap ノードの作成とそれをもとにした Master ノードの作成までができています。

```
# openshift-install wait-for bootstrap-complete --dir=installdir --log-level=debug
DEBUG OpenShift Installer 4.9.5
DEBUG Built from commit 8223216bdcef5de56b52240ab7160ca909a9e56c
INFO Waiting up to 20m0s for the Kubernetes API at https://api.ocp49.example.localdomain:6443...
INFO API v1.22.0-rc.0+a44d0f0 up
INFO Waiting up to 30m0s for bootstrapping to complete...
DEBUG Bootstrap status: complete
INFO It is now safe to remove the bootstrap resources
DEBUG Time elapsed per stage:
DEBUG API: 1m528s
INFO Time elapsed: 13m12s
```

2 インフラ実践編

Part

Part 2 > インフラ実践編 CHAPTER 6 OCPのインストール方法

Bootstrapのプロセスが完了すればBootstrapノードの役目は終了です。コマンド出力に「INFO It is now safe to remove the bootstrap resources」というメッセージが出ていれば、Bootstrap ノードを削除してもかまいません。

6.3.6 Workerノード/Infraノードの作成

次にWorkerノードを起動します。InfraノードはWorkerノードの一種なので、一緒に起動するこ とにします。こちらも先ほどと同様に、Workerノードのサーバーを順にパワーオンすれば自動的に 起動していきます。BootstrapノードやMasterノードと同じように、Workerノードの画面でログイン コンソールが表示されることを確認しましょう。エラーメッセージが表示されることがありますが、 ここでは無視して問題ありません。

すべてのWorkerノードが起動したら、WorkerノードのCSR (Certificate Signing Request) に署名 をしていきます。CSRは証明書のもとになる情報への署名要求で、承認することで署名されて証明書 になります。これらのCSRはWorkerノードに紐づくもので、WorkerノードがOpenShiftクラスタに 参加する際に必要となるものです。

作業用マシンでoc get csr コマンドを実行して、CONDITIONがPendingになっているものを探します。

# oc get csr			
NAME	AGE	SIGNERNAME	REQUESTOR 🔁
		REQUESTEDDURATION CONDITION	N NAME
csr-5pc2p	64m	kubernetes.io/kubelet-serving	system:node:m1 Đ
.ocp49.example.localdomain		<none> Approved</none>	,Issued
csr-b4qrw	64m	kubernetes.io/kubelet-serving	system:node:m2 Đ
.ocp49.example.localdomain		<none> Approved</none>	,Issued
csr-c897d	20m	kubernetes.io/kube-apiserver-client-kubel	et 🛛 system:service 🔁
account:openshift-machine-config-operator:node-b	ootstr	apper <none> Pending</none>	
csr-f8qtn	64m	kubernetes.io/kubelet-serving	system:node:m3 Đ
.ocp49.example.localdomain		<none> Approved</none>	,Issued
csr-gcb44	64m	kubernetes.io/kube-apiserver-client-kubel	et 🛛 system:service Đ
account:openshift-machine-config-operator:node-b	ootstr	apper <none> Approved</none>	,Issued
csr-hgj62	21m	kubernetes.io/kube-apiserver-client-kube]	et system:service €
account:openshift-machine-config-operator:node-b	ootstr	apper <none> Pending</none>	
csr-hx6q4	21m	kubernetes.io/kube-apiserver-client-kube]	et system:service Đ
account:openshift-machine-config-operator:node-b	ootstr	apper <none> Pending</none>	
csr-kw65q	21m	kubernetes.io/kube-apiserver-client-kubel	et system:service Đ
account:openshift-machine-config-operator:node-b	ootstr	apper <none> Pending</none>	
csr-m26ks	64m	kubernetes.io/kube-apiserver-client-kubel	et system:service Đ
account:openshift-machine-config-operator:node-b	ootstr	apper <none> Approved</none>	,Issued
csr-mtt6q	20m	kubernetes.io/kube-apiserver-client-kubel	et system:service Đ
account:openshift-machine-config-operator:node-b	ootstr	rapper <none> Pending</none>	
csr-nffzk	20m	kubernetes.io/kube-apiserver-client-kubel	et system:service €

account:openshift-machine-config-operator:node-bo	ootstr	apper	<none></none>	Pending	
csr-pvcwl	64m	kubernetes.	io/kube-apiserver	-client-kubelet	system:service €
<pre>account:openshift-machine-config-operator:node-bd</pre>	ootstr	apper	<none></none>	Approved,Issu	Jed
<pre>system:openshift:openshift-authenticator-vcv8f</pre>	62m	kubernetes.	io/kube-apiserver	-client	system:service €
account:openshift-authentication-operator:authentication-operator.	ticati	on-operator	<none></none>	Approved,Issu	Jed
<pre>system:openshift:openshift-monitoring-phsxc</pre>	62m	kubernetes.	io/kube-apiserver	-client	system:service €
$\verb+account:openshift-monitoring:cluster-monitoring-cluster-monitoring$	operat	or	<none></none>	Approved,Issu	beu

ここでは6つのCSRがPendingと表示されていますが、3台のWorkerノードと3台のInfraノードの合計6ノードに対応しています。Workerノードの準備が整うと必ず表示されるものなので、もし Pendingが表示されていない場合はしばらく待ってから再度コマンドを実行してみてください。

Pending状態のCSRを承認 (approve) するには、oc adm certificate approve コマンドを実行 します。1つずつ実行してもよいのですが、次のように xargs コマンドを使って一気に承認することも できます。

oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}{end}}
{{end}}' | xargs oc adm certificate approve

certificatesigningrequest.certificates.k8s.io/csr-c897d approved certificatesigningrequest.certificates.k8s.io/csr-hgj62 approved certificatesigningrequest.certificates.k8s.io/csr-hx6q4 approved certificatesigningrequest.certificates.k8s.io/csr-hx6q4 approved certificatesigningrequest.certificates.k8s.io/csr-mtt6q approved certificatesigningrequest.certificates.k8s.io/csr-mtt6q approved certificatesigningrequest.certificates.k8s.io/csr-mfzk approved

承認が完了すると、今度はすぐに別のPendingのCSRが出てきます。1つのWorkerノードに対し て、クライアント・リクエストCSRとサーバー・リクエストCSRの、2つのCSRの承認を要求されます。 Pending表示がなくなるまで、oc adm certificate approveコマンドを繰り返し実行し、承認して いってください。

CSRをすべて承認したら、oc get nodes コマンドで Worker ノードの状態(STATUS)がReady に なっているのを確認できます。

# oc get nodes				
NAME	STATUS	ROLES	AGE	VERSION
i1.ocp49.example.localdomain	Ready	worker	10m	v1.22.0-rc.0+a44d0f0
i2.ocp49.example.localdomain	Ready	worker	10m	v1.22.0-rc.0+a44d0f0
i3.ocp49.example.localdomain	Ready	worker	10m	v1.22.0-rc.0+a44d0f0
m1.ocp49.example.localdomain	Ready	master	75m	v1.22.0-rc.0+a44d0f0
m2.ocp49.example.localdomain	Ready	master	75m	v1.22.0-rc.0+a44d0f0
m3.ocp49.example.localdomain	Ready	master	75m	v1.22.0-rc.0+a44d0f0
w1.ocp49.example.localdomain	Ready	worker	10m	v1.22.0-rc.0+a44d0f0
w2.ocp49.example.localdomain	Ready	worker	10m	v1.22.0-rc.0+a44d0f0
w3.ocp49.example.localdomain	Ready	worker	10m	v1.22.0-rc.0+a44d0f0



6.3.7 クラスタ作成の確認

あとはOpenShiftクラスタが作成されるのを待つだけです。作業用マシンで次のコマンドを実行し てクラスタインストールのプロセスをモニターします。このコマンドが終了すればクラスタ完成です。

openshift-install wait-for install-complete --dir=installdir INFO Waiting up to 40m0s for the cluster at https://api.ocp49.example.localdomain:6443 to initial ize... INFO Waiting up to 10m0s for the openshift-console route to be created... INFO Install complete! INFO To access the cluster as the system:admin user when using 'oc', run 'export KUBECONFIG=/root /openshift/installdir/auth/kubeconfig' INFO Access the OpenShift web-console here: https://console-openshift-console.apps.ocp49.example. localdomain INFO Login to the console with user: "kubeadmin", and password: "NqrXg-pdjy9-2c2er-MYTzr" INFO Time elapsed: 30m24s

クラスタインストールのプロセスでは Cluster Operator のセットアップが行われており、これらの Operator の AVAILABLE がすべて True になるまで待つ必要があります。 Cluster Operator は、oc get clusteroperators コマンドを使って状態を確認することができます。

<pre># oc get clusteroperators</pre>						
NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED	SINCE	MESSAGE
authentication	4.9.5	True	False	False	5m8s	
baremetal	4.9.5	True	False	False	93m	
cloud-controller-manager	4.9.5	True	False	False	97m	
cloud-credential	4.9.5	True	False	False	98m	
cluster-autoscaler	4.9.5	True	False	False	95m	
config-operator	4.9.5	True	False	False	96m	
console	4.9.5	True	False	False	9m32s	
csi-snapshot-controller	4.9.5	True	False	False	95m	
dns	4.9.5	True	False	False	93m	
etcd	4.9.5	True	False	False	94m	
image-registry	4.9.5	True	False	False	87m	
ingress	4.9.5	True	False	False	11m	
insights	4.9.5	True	False	False	88m	
kube-apiserver	4.9.5	True	False	False	90m	
kube-controller-manager	4.9.5	True	False	False	90m	
kube-scheduler	4.9.5	True	False	False	94m	
kube-storage-version-migrator	4.9.5	True	False	False	95m	
machine-api	4.9.5	True	False	False	95m	
machine-approver	4.9.5	True	False	False	94m	
machine-config	4.9.5	True	False	False	94m	
marketplace	4.9.5	True	False	False	95m	
monitoring	4.9.5	True	False	False	10m	
network	4.9.5	True	False	False	95m	

<pre>node-tuning openshift-apiserver openshift-controller-manager openshift-samples operator-lifecycle-manager operator-lifecycle-manager-catalog operator-lifecycle-manager-packageserver service-ca storage</pre>	4.9.5 4.9.5 4.9.5 4.9.5 4.9.5 4.9.5 4.9.5 4.9.5 4.9.5 4.9.5	True True True True True True True True	False False False False False False False False	False False False False False False False False	94m 87m 87m 94m 94m 87m 95m 96m
storage	4.9.5	True	False	False	96m

通常は何もしなくてもすべてのOperatorでAVAILABLEがTrueと表示されるのですが、いつまで 経ってもAVAILABLEがTrueにならないことがあります。この場合、該当するOperatorが管理する Podが稼働していないなどの問題が発生している可能性があります。対処としては、次の実行例のよ うにoc get pod --all-namespacesコマンドを使って、RunningまたはCompletedのステータス以 外のPodを見つけ出し、oc logsコマンドやoc get eventsコマンドなどで原因を調べて対処するよ うにしてください。

# oc get pod NAMESPACE	all-r	amespaces gr	ep -v Runni	ing grep -v Completed NAME	Ð
	READY	STATUS	RESTARTS	AGE	
openshift-co	nsole			console-86bfd57fc9-k8sbm	Ð
	1/1	Terminating	0	3m52s	
openshift-ku	be-apise	rver		kube-apiserver-m1.ocp49.example.local	Ð
	0/4	Init:0/1	0	54s	
	074	11110.071	0	515	

クラスタのインストールが完了したら、クラスタの Web コンソールにアクセスしてみましょう。 kubeadminユーザーでクラスタにログインできれば成功です。

Web コンソールの URL と kubeadmin ユーザーのパスワードは、openshift-install wait-for install-complete コマンドの出力に表示されています。これらがわからなくなってしまったら、再度 openshift-install wait-for install-complete コマンドを実行するか、作業用マシンのインストールディレクトリに保存されるインストールログファイルを確認するとわかります。次の実行例で は、tail コマンドでインストールログファイルを表示しています。出力で太字にしている部分がユーザー名とパスワードです。

```
# tail installdir/.openshift_install.log
time="2021-11-10T02:27:10+09:00" level=info msg="Waiting up to 40m0s for the cluster at https://a >
pi.ocp49.example.localdomain:6443 to initialize..."
time="2021-11-10T02:27:10+09:00" level=debug msg="Cluster is initialized"
time="2021-11-10T02:27:10+09:00" level=info msg="Waiting up to 10m0s for the openshift-console ro >
ute to be created..."
time="2021-11-10T02:27:10+09:00" level=debug msg="Route found in openshift-console namespace: con >
sole"
```

Part 2

インフラ実践編

6



time="2021-11-10T02:27:10+09:00" level=debug msg="OpenShift console route is admitted" time="2021-11-10T02:27:10+09:00" level=info msg="Install complete!" time="2021-11-10T02:27:10+09:00" level=info msg="To access the cluster as the system:admin user w hen using 'oc', run 'export KUBECONFIG=/root/install-bm/auth/kubeconfig'" time="2021-11-10T02:27:10+09:00" level=info msg="Access the OpenShift web-console here: https://c onsole-openshift-console.apps.ocp49.example.localdomain" time="2021-11-10T02:27:10+09:00" level=info msg="Login to the console with user: \"kubeadmin\", a nd password: \"NqrXg-pdjy9-2c2er-MYTzr\"" time="2021-11-10T02:27:10+09:00" level=info msg="Time elapsed: 30m24s"

6.3.8 クラスタ完成後の作業

以上でOpenShiftクラスタのUPIインストールは完成です。この時点でアプリケーションPodを稼働させることも可能です。しかし現実的にコンテナプラットフォームとして使えるようになるには、永 続ストレージを用意したり、クラスタアプリケーションをInfraノードに移動させたり、さらに作業を 行うことになります。

ここではこれらの作業の詳細については触れません。本節の冒頭で紹介したGitHubのサイト(図 6.3の上部)に手順が記載されているので、そちらを参考にして実施してください。



エアギャップ環境への対応

OpenShift クラスタのインストールは、インターネットに接続できる環境で行われることが想定され ています。これは、作業用マシンで各種ファイルをダウンロードするだけでなく、クラスタ内で Cluster Operator が稼働させるアプリケーションのコンテナイメージを、registry.redhat.io や quay.io といった インターネット上のコンテナレジストリから Pull するために必要だからです。

しかし、場合によってはインターネットに接続できない、いわゆるエアギャップ環境(隔離環境)で OpenShift クラスタをインストールすることもあるでしょう。そのような場合は、環境内にローカルレジ ストリを作成してクラスタインストールすることができます。この場合、インターネット上のコンテナレジ ストリからローカルレジストリにミラーするため、ローカルレジストリが稼働するサーバーだけはインター ネットに接続できる必要があるので注意してください。一度ミラーしてしまえばインターネット接続は切断 してもかまいません。

具体的な手順については、OCPのオンラインドキュメントの「Mirroring images for a disconnected installation」のページを参照してください。

• OpenShift Container Platform 4.9 Documentation; Mirroring images for a disconnected installation

https://docs.openshift.com/container-platform/4.9/installing/installing-mirroring-install ation-images.html



企画	レッドハット株式会社
	株式会社リアルグローブ・オートメーティッド、青島 秀治、レッドハット株式会社、
	森 真也、清水 護、織 学、宇都宮 卓也、斎藤 和史、野間 亮志、荒木 俊博
監 修	須江 信洋
URL	https://www.redhat.com/ja/global/japan
問い合わせ	https://www.redhat.com/ja/contact

※本冊子は下記よりPCやタブレットで閲覧、ダウンロードできます。 https://tracks.redhat.com/openshiftmook2022

