



OpenShift Container Platform 3.11

クラスタのアップグレード

OpenShift Container Platform 3.11 クラスタのアップグレード

OpenShift Container Platform 3.11 クラスターのアップグレード

OpenShift Container Platform 3.11 クラスターのアップグレード

法律上の通知

Copyright © 2020 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

本書を活用した OpenShift Container Platform 3.11 クラスターのアップグレード

目次

第1章 アップグレード方式およびストラテジー	3
1.1. アップグレードストラテジー	3
1.1.1. インプレースアップグレード	3
1.1.2. Blue-Green デプロイメント	3
第2章 クラスターの自動インプレースアップグレードの実行	4
2.1. アップグレードのワークフロー	4
2.2. 前提条件	4
2.3. アップグレードの準備	5
2.3.1. ポリシー定義の更新	8
2.3.2. アップグレードフェーズ	9
2.3.3. ノードのアップグレードパラメーター	9
2.3.4. アップグレード用の Ansible Hook	10
2.3.4.1. 制限	10
2.3.4.2. フックの使用	11
2.3.4.3. 利用可能なアップグレードフック	11
2.3.5. OpenShift Container Platform のアップグレードにおける特別な考慮事項	13
2.3.5.1. 大規模なアップグレードに関する特別な考慮事項	13
2.3.5.2. gcePD を使用する場合の特別な考慮事項	13
2.4. 最新 OPENSIFT CONTAINER PLATFORM リリースへのアップグレード	14
2.5. コンテナ化された GLUSTERFS を使用する場合の OPENSIFT CONTAINER PLATFORM のアップグレード	15
2.6. オプションコンポーネントのアップグレード	16
2.6.1. EFK ログスタックのアップグレード	17
2.6.1.1. フィールドのフィールド名にドットがあるかどうかの判別	18
2.6.1.2. フィールドのフィールド名にドットが含まれる場合のアップグレード	18
2.6.1.3. フィールドにドットが含まれない場合のアップグレード	19
2.6.2. クラスターメトリクスのアップグレード	20
2.7. アップグレードの検証	20
第3章 BLUE-GREEN クラスターアップグレードの実行	22
3.1. BLUE-GREEN アップグレードの準備	23
3.1.1. ソフトウェアエンタイトルメントの共有	23
3.1.2. Blue ノードのラベリング	23
3.1.3. Green ノードの作成およびラベリング	23
3.1.4. Green ノードの検証	24
3.2. GREEN ノードの準備	25
3.3. BLUE ノードの退避および使用停止	26
第4章 オペレーティングシステムの更新	28
4.1. ホストでのオペレーティングシステムの更新	28
4.1.1. OpenShift Container Storage を実行するノードのアップグレード	28
第5章 クラスターのダウングレード	30
5.1. バックアップの検証	30
5.2. クラスターのシャットダウン	30
5.3. RPM と静的 POD の削除	31
5.4. RPM の再インストール	31
5.5. OPENSIFT CONTAINER PLATFORM サービスの再オンライン化 手順	32
5.6. ダウングレードの検証	32

第1章 アップグレード方式およびストラテジー

OpenShift Container Platform の新規バージョンがリリースされると、既存のクラスターをアップグレードして最新の拡張機能およびバグ修正を適用できます。これには、リリース 3.10 から 3.11 などの以前のマイナーバージョンからのアップグレードや、マイナーバージョン (3.11.z release) 内での非同期エラータ更新の適用が含まれます。最新の変更点を確認するには、『[OpenShift Container Platform 3.11 リリースノート](#)』を参照してください。



注記

メジャーバージョン間のコアとなるアーキテクチャーの変更により、OpenShift Enterprise 2 環境は OpenShift Container Platform 3 にアップグレードできず、新規インストールが必要になります。

OpenShift Container Platform のアップグレードプロセスでは、Ansible Playbook を使用して、クラスターのアップグレードに必要なタスクを自動化します。初期インストール時や前回の正常なアップグレードで使用した「インベントリーファイル」を使用して、アップグレード Playbook を実行する必要があります。この方法を使用する場合には、アップグレードストラテジー (インプレースアップグレード、または Blue-Green デプロイメントのいずれか) を選択できます。

特に記述がない限り、メジャーバージョンに含まれるノードおよびマスターには、[1つのマイナーバージョンに対する](#)前方および後方互換性があるので、クラスターのアップグレードはスムーズに実行できます。ただし、クラスター全体をアップグレードするのに、一致しないバージョンを必要以上の時間をかけて実行することはできません。

アップグレードの前に、OpenShift Container Platform サービスがすべて実行されていることを確認します。コントロールプレーンのアップグレードが失敗した場合は、マスターのバージョンをチェックし、すべてのバージョンが同じであることを確認します。マスターがすべて同じバージョンである場合、アップグレードを再実行します。バージョンが異なる場合は、マスターをバージョン付けされた低いバージョンのマスターに一致するようにダウングレードしてからアップグレードを再実行します。

1.1. アップグレードストラテジー

OpenShift Container Platform クラスターのアップグレードでは、インプレースアップグレードまたは Blue-Green デプロイメントの2つのストラテジーを使用できます。

1.1.1. インプレースアップグレード

インプレースアップグレードでは、クラスターのアップグレードは実行中の単一クラスターのすべてのホストで実行されます。これはまずマスターで実行され、次にノードで実行されます。Pod はノードのアップグレードの開始前にそのノードから退避し、他の実行中のノードで再作成されます。これは、ユーザーアプリケーションのダウンタイムを短縮するのに役立ちます。

1.1.2. Blue-Green デプロイメント

[Blue-Green デプロイメント](#) アップグレード方式はインプレース方式と同様のフローに従います。まずマスターおよび etcd サーバーがアップグレードされます。ただしこの方式では、新規ノードをインプレースでアップグレードするのではなく、新規ノード用の並列環境が作成されます。

この方式では、新規デプロイメントの検証後、管理者は古いノードセット (例: 「Blue」 デプロイメント) から新規セット (例: 「Green」 デプロイメント) にトラフィックを切り換えることができます。問題が検出される場合も、古いデプロイメントへのロールバックを簡単かつすぐに実行できます。

第2章 クラスターの自動インプレースアップグレードの実行

標準の[クラスターインストール](#)プロセスを使用してインストールしており、使用したインベントリーファイルが利用できる場合には、アップグレード Playbook を使用してクラスターのアップグレードプロセスを自動化できます。

OpenShift Container Platform をアップグレードするには、インストール時に使用したのと同じインベントリーファイルを使用して Ansible Playbook を実行します。同じ `v3_11` アップグレード Playbook を実行して、以下を実行します。

- 既存の OpenShift Container Platform バージョン 3.10 クラスターのバージョン 3.11 へのアップグレード。
- OpenShift Container Platform version 3.11 クラスターの最新の[非同期エラータ更新](#)へのアップグレード。



重要

Ansible Playbook を `--tags` または `--check` オプションを使用して実行することを、Red Hat ではサポートしていません。

2.1. アップグレードのワークフロー

3.10 から 3.11 へのコントロールプレーンのアップグレードでは、以下の手順が実行されます。

- リカバリー用にすべての etcd データのバックアップを実行する。
- API およびコントローラーを 3.10 から 3.11 に更新する。
- 内部データ構造を 3.11 に更新する。
- デフォルトルーター (ある場合) を 3.10 から 3.11 に更新する。
- デフォルトレジストリー (ある場合) を 3.10 から 3.11 に更新する。
- デフォルトのイメージストリームと InstantApp テンプレートを更新する。

3.10 から 3.11 へのノードのアップグレードではノードのローリングアップデートを実行し、以下が行われます。

- ノードのサブセットにスケジュール対象外 (unschedulable) のマークを付け、それらのノードを Pod からドレイン (解放) する。
- ノードコンポーネントを 3.10 から 3.11 に更新する。
- それらのノードをサービスに返す。

2.2. 前提条件

クラスターをアップグレードする前に、以下を実行してください。

- 『[OpenShift Container Platform 3.11 リリースノート](#)』を確認してください。リリースノートには、OpenShift Container Platform およびその機能に対する変更点についての重要な通知が含まれています。

- 10以上のワーカーノードおよび数千のプロジェクトおよびPodを含む大規模なアップグレードを実行している場合は、アップグレードの失敗を防ぐために「[大規模なアップグレードに関する特別な考慮事項](#)」を確認してください。
- 非接続のクラスター更新を実行している場合、イメージの新しいバージョンでイメージレジストリを更新する必要があります。そうしないと、クラスターの更新が失敗します。たとえば、**3.11.153** から **3.11.157** に更新する場合には、**v3.11.157** イメージタグが存在することを確認してください。
- クラスターを [バージョン 3.10 の最新の非同期リリース](#) にアップグレードします。3.10 よりも前のバージョンを使用している場合は、まず段階的なアップグレードを実行する必要があります。たとえば、3.7 から 3.9 にアップグレードし (3.8 バージョンは [省略](#) されています)、次に 3.9 から 3.10 にアップグレードします。
- [環境ヘルスチェック](#) を実行してクラスターの健全性を確認します。このプロセスでは、ノードが **Ready** 状態で、予想される開始バージョンを実行していることが確認され、診断エラーまたは警告がないことを確認できます。
- クラスターが現行の [前提条件](#) を満たしていることを確認します。そうでないと、アップグレードが失敗する可能性があります。
- アップグレードの前日に、OpenShift Container Platform ストレージの移行を検証して、サービスを停止する前に考えられる問題を解決しておくようにしてください。以下のコマンドは etcd ストレージの API オブジェクトの保存されたバージョンを更新します。

```
$ oc adm migrate storage --include=* --loglevel=2 --config
/etc/origin/master/admin.kubeconfig
```

2.3. アップグレードの準備

前提条件を満たしていることを確認したら、自動アップグレードを準備します。

1. 最新のサブスクリプションデータを Red Hat Subscription Manager からプルします。

```
# subscription-manager refresh
```

2. OpenShift Container Platform 3.10 から 3.11 にアップグレードしている場合は、以下を実行します。
 - a. OpenShift Container Platform 3.10 にダウングレードする場合に必要なファイルをバックアップします。
 - i. マスターホストで、以下のファイルをバックアップします。

```
/etc/origin/master/master-config.yaml
/etc/origin/master/master.env
/etc/origin/master/scheduler.json
```

- ii. マスターを含むノードホストで、以下のファイルをバックアップします。

```
/etc/origin/node/node-config.yaml
```

- iii. etcd ホストで (etcd が同一の場所に配置されているマスターを含む)、以下のファイルをバックアップします。

```
/etc/etcd/etcd.conf
```

- b. アップグレードプロセスでは、リカバリー目的ですべての etcd データのバックアップを作成しますが、`/backup/etcd-xxxxxx/backup.db` に最新の etcd のバックアップがあることを確認してからこのタスクを進めてください。etcd の手動でのバックアップ手順は、『[Day 2 操作ガイド](#)』に記載されています。



注記

OpenShift Container Platform をアップグレードする場合、etcd 設定は変更されません。etcd をマスターホストの静的 Pod として、またはマスターホストの別のサービスまたは別のホストとして実行するいずれの場合でも、etcd はアップグレード後に変更されません。

- c. 手作業で、各マスターおよびノードホストの 3.10 リポジトリを無効にして、3.11 リポジトリを有効にします。rhel-7-server-ansible-2.9-rpms リポジトリをまだ有効にしていない場合には、有効にしてください。

- x86_64 サーバーでのクラウドインストールおよびオンプレミスインストールの場合には、以下のコマンドを実行します。

```
# subscription-manager repos \
  --disable="rhel-7-server-ose-3.10-rpms" \
  --disable="rhel-7-server-ansible-2.4-rpms" \
  --enable="rhel-7-server-ose-3.11-rpms" \
  --enable="rhel-7-server-rpms" \
  --enable="rhel-7-server-extras-rpms" \
  --enable="rhel-7-server-ansible-2.9-rpms"
# yum clean all
```

- IBM POWER8 サーバーでのオンプレミスインストールの場合には、以下のコマンドを実行します。

```
# subscription-manager repos \
  --disable="rhel-7-for-power-le-ose-3.10-rpms" \
  --enable="rhel-7-for-power-le-rpms" \
  --enable="rhel-7-for-power-le-extras-rpms" \
  --enable="rhel-7-for-power-le-optional-rpms" \
  --enable="rhel-7-server-ansible-2.9-for-power-le-rpms" \
  --enable="rhel-7-server-for-power-le-rhsc-rpms" \
  --enable="rhel-7-for-power-le-ose-3.11-rpms"
# yum clean all
```

- IBM POWER9 サーバーでのオンプレミスインストールの場合には、以下のコマンドを実行します。

```
# subscription-manager repos \
  --disable="rhel-7-for-power-le-ose-3.10-rpms" \
  --enable="rhel-7-for-power-9-rpms" \
  --enable="rhel-7-for-power-9-extras-rpms" \
  --enable="rhel-7-for-power-9-optional-rpms" \
  --enable="rhel-7-server-ansible-2.9-for-power-9-rpms" \
```

```
--enable="rhel-7-server-for-power-9-rhsc-rpms" \
--enable="rhel-7-for-power-9-ose-3.11-rpms"
# yum clean all
```

- d. アップグレード Playbook を実行するホストで、最新版の **openshift-ansible** パッケージが設定されていることを確認してください。

```
# yum update -y openshift-ansible
```

- e. Cluster Monitoring Operator の準備をします。バージョン 3.11 では、Cluster Monitoring Operator はデフォルトでインフラストラクチャーノードにインストールされます。クラスターがインフラストラクチャーノードを使用しない場合は、以下を実行します。

- インフラストラクチャーノードをクラスターに追加します。
- **openshift_cluster_monitoring_operator_install=false** をインベントリーファイルに追加して Cluster Monitoring Operator を無効にします。
- Cluster Monitoring Operator をインストールするノードに **openshift_cluster_monitoring_operator_node_selector** のマークを付けて、ノードを指定します。

- f. 標準の OpenShift Container Platform レジストリーを使用する場合、**registry.access.redhat.com** から **registry.redhat.io** への変更に対して準備してください。「[Accessing and Configuring the Red Hat Registry](#)」に記載されている設定の手順を実行してください。

3. インベントリーファイルを確認し、これを更新します。

- a. (初期インストールまたは最近のクラスターのアップグレードなどで) Ansible Playbook を最後に実行した後にマスターまたはノード設定ファイルに設定変更を手動で加えた場合は、それらの変更がインベントリーファイルに反映されていることを確認します。手動で加えた変更に関連する変数については、アップグレードの実行前に同等の変更をインベントリーファイルに適用してください。これを実行しないと、手動による変更がアップグレード時にデフォルト値で上書きされ、Pod が適切に実行されなくなるか、または他のクラスターの安定性に問題が生じる可能性があります。
- b. デフォルトで、インストーラーは証明書が1年以内に期限切れになるかどうかや、その期間内に期限切れになる場合に失敗するかどうかを確認します。証明書の有効日数を変更するには、**openshift_certificate_expiry_warning_days** パラメーターに新規の値を指定します。たとえば、証明書の有効期間を180日間とする場合、**openshift_certificate_expiry_warning_days=180** を指定します。
- c. 証明書が期限切れになるかどうかのチェックを省略するには、**openshift_certificate_expiry_fail_on_warn=False** を設定します。
- d. マスター設定ファイルの **admissionConfig** 設定に変更を加えた場合は、「[インベントリーファイルの設定](#)」で **openshift_master_admission_plugin_config** 変数を確認してください。これを確認しない場合、「[マスターでのオーバーコミットの設定](#)」に説明されているように **ClusterResourceOverride** 設定を事前に手動で設定している場合には、Pod が **Pending** 状態のままになる可能性があります。
- e. 3.10 よりも前のバージョンの OpenShift Container Platform で **openshift_hostname** パラメーターを使用していた場合、**openshift_kubelet_name_override** パラメーターがまだインベントリーファイルにあり、以前のバージョンで使用した **openshift_hostname** の値に設定されていることを確認してください。



重要

アップグレード後は、**openshift_kubelet_name_override** パラメーターをインベントリーファイルから削除できません。

- f. クラスターの `/etc/origin/master/htpasswd` ファイルを手動で管理する場合、**openshift_master_manage_htpasswd=false** をインベントリーファイルに追加して、アップグレードプロセスで `htpasswd` ファイルが上書きされないようにします。

2.3.1. ポリシー定義の更新

クラスターのアップグレード時に、また任意のマスターの再起動時は常に、[デフォルトのクラスターロール](#)が欠落しているパーミッションを復元するために自動的に調整されます。

1. デフォルトクラスターロールをカスタマイズしており、ロールの調整によってそれらに変更されないようにするには、以下のように各ロールを調整から保護します。

```
$ oc annotate clusterrole.rbac <role_name> --overwrite
rbac.authorization.kubernetes.io/autoupdate=false
```



警告

この設定を含むロールがアップグレード後に新規または必須のパーミッションを組み込むように手動で更新する必要があります。

2. デフォルトのブートストラップポリシーテンプレートを生成します。

```
$ oc adm create-bootstrap-policy-file --filename=policy.json
```



注記

ファイルの内容は OpenShift Container Platform バージョンによって異なりますが、ファイルにはデフォルトポリシーのみが含まれます。

3. **policy.json** ファイルを、クラスターロールのカスタマイズを組み込むように更新します。
4. ポリシーを使用し、調整から保護されていないロールおよびロールバインディングを自動的に調整します。

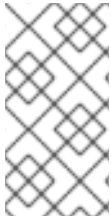
```
$ oc auth reconcile -f policy.json
```

5. SCC (Security Context Constraints) を調整します。

```
# oc adm policy reconcile-sccs \
--additive-only=true \
--confirm
```

2.3.2. アップグレードフェーズ

OpenShift Container Platform クラスターは1つまたは複数のフェーズでアップグレードできます。単一の Ansible Playbook を実行してすべてのホストを1つのフェーズでアップグレードすることも、別々の Playbook を使用して **コントロールプレーン** または **マスターコンポーネント** と **ノード** を複数のフェーズでアップグレードすることもできます。



注記

OpenShift Container Platform クラスターが GlusterFS Pod を使用する場合、複数のフェーズでアップグレードを実行する必要があります。GlusterFS に関するアップグレード方法についての詳細は、「[コンテナ化された GlusterFS を使用する場合の特別な考慮事項](#)」を参照してください。

複数のフェーズでアップグレードする場合、コントロールプレーンのフェーズには、以下のアップグレードが含まれます。

- マスターコンポーネント
- マスターで実行されるノードサービス
- マスターで実行される Docker または CRI-O
- スタンドアロン etcd ホストで実行される Docker または CRI-O

ノードのみをアップグレードする場合、まずコントロールプレーンをアップグレードする必要があります。ノードのフェーズには、以下のアップグレードが含まれます。

- スタンドアロンノードで実行されるノードサービス
- スタンドアロンノードで実行される Docker または CRI-O

マスターコンポーネントを実行するノードは、コントロールプレーンのアップグレードフェーズにのみアップグレードされます。これは、マスターのノードサービスおよびコンテナエンジンのアップグレードが2回(1回目はコントロールプレーンのフェーズ、2回目はノードのフェーズ)実行されないようにします。

2.3.3. ノードのアップグレードパラメーター

アップグレードを単一または複数フェーズのいずれかで実行する場合でも、**-e** オプションを使って特定の Ansible 変数をアップグレード Playbook に渡すことで、アップグレードのノード部分の処理方法をカスタマイズできます。

- **openshift_upgrade_nodes_serial** 変数を整数またはパーセンテージに設定して同時にアップグレードできるノードホストの数を制御できます。デフォルトは **1** であり、この場合ノードは一度に1つずつアップグレードされます。
たとえば、一度に検出されるノードの全体数の 20 パーセントをアップグレードするには、以下を実行します。

```
$ ansible-playbook -i <path/to/inventory/file> \
  </path/to/upgrade/playbook> \
  -e openshift_upgrade_nodes_serial="20%"
```

- 特定のラベルを持つノードのみのアップグレードを指定するには **openshift_upgrade_nodes_label** を設定します。

たとえば、`group1` リージョンのノードのみを一度に2つアップグレードするには、以下を実行します。

```
$ ansible-playbook -i <path/to/inventory/file> \
  </path/to/upgrade/playbook> \
  -e openshift_upgrade_nodes_serial="2" \
  -e openshift_upgrade_nodes_label="region=group1"
```



注記

ノードラベルについての詳細は、「[ノードの管理](#)」を参照してください。

- アップグレードの各バッチで失敗するノードの数を指定するには **`openshift_upgrade_nodes_max_fail_percentage`** 変数を設定します。失敗したノードのパーセンテージが指定した値を上回る場合、Playbook はアップグレードプロセスを停止します。
- ノードに失敗のマークを付ける前の待機時間の長さを指定するには **`openshift_upgrade_nodes_drain_timeout`** 変数を設定します。以下の例では、一度に10ノードをアップグレードして、20パーセントを超えるノードが失敗する場合にはアップグレードが停止します。また、ノードをドレイン(解放)するまでに600秒を超える時間がかかる場合、ノードには失敗のマークが付けられます。

```
$ ansible-playbook -i <path/to/inventory/file> \
  </path/to/upgrade/playbook> \
  -e openshift_upgrade_nodes_serial=10 \
  -e openshift_upgrade_nodes_max_fail_percentage=20 \
  -e openshift_upgrade_nodes_drain_timeout=600
```

2.3.4. アップグレード用の Ansible Hook

OpenShift Container Platform のアップグレード時の特定の操作の実行中に、**フック** というシステムでカスタムタスクを実行できます。フックを使うと、クラスター管理者はアップグレードの特定分野の前後に実行するタスクを定義するファイルを指定できます。また、フックを使用して OpenShift Container Platform のアップグレード時にカスタムインフラストラクチャーを検証し、変更することができます。

フックが失敗すると操作も失敗します。フックはべき等性があるか、または複数回実行でき、同じ結果を出せるように設計します。

2.3.4.1. 制限

- フックには定義され、バージョン付けされたインターフェースがありません。フックは内部の **`openshift-ansible`** 変数を使用できますが、それらの変数が今後のリリースでもそのまま残される保証はありません。また、フックは今後バージョン付けされる可能性があり、フックが最新の **`openshift-ansible`** と連動するように更新する必要があることを示す警告が事前に出されません。
- フックにはエラー処理機能がなく、フックにエラーが生じるとアップグレードプロセスは中止します。エラーが出た場合は、問題に対応してからアップグレードを再び開始する必要があります。
- ノードのアップグレードフックはマスターではなく、ノードのみに実行できます。マスターでフックを実行するには、それらのノードにマスターフックを指定する必要があります。

2.3.4.2. フックの使用

フックは **ホスト** インベントリーファイルの **OSEv3:vars** セクションに定義されます。

各フックは Ansible タスクを定義する YAML ファイルをポイントする必要があります。このファイルは **include** として使用されます。つまり、このファイルは Playbook にすることはできず、タスクのセットである必要があります。あいまいさを避けるためにフックファイルへの絶対パスを使用することがベストプラクティスとして推奨されます。

インベントリーファイルのフック定義の例

```
[OSEv3:vars]
openshift_master_upgrade_pre_hook=/usr/share/custom/pre_master.yml
openshift_master_upgrade_hook=/usr/share/custom/master.yml
openshift_master_upgrade_post_hook=/usr/share/custom/post_master.yml

openshift_node_upgrade_pre_hook=/usr/share/custom/pre_node.yml
openshift_node_upgrade_hook=/usr/share/custom/node.yml
openshift_node_upgrade_post_hook=/usr/share/custom/post_node.yml
```

pre_master.yml タスクの例

```
---
# Trivial example forcing an operator to ack the start of an upgrade
# file=/usr/share/custom/pre_master.yml

- name: note the start of a master upgrade
  debug:
    msg: "Master upgrade of {{ inventory_hostname }} is about to start"

- name: require an operator agree to start an upgrade
  pause:
    prompt: "Hit enter to start the master upgrade"
```

2.3.4.3. 利用可能なアップグレードフック

表2.1 マスターのアップグレードフック

フック名	説明
openshift_master_upgrade_pre_hook	<ul style="list-style-type: none"> 各マスターのアップグレード 前 に実行します。 この Hook は 各マスター に対して連続して実行されます。 タスクが異なるホストに対して実行される必要がある場合、該当するタスクは delegate_to または local_action を使用する必要があります。

フック名	説明
openshift_master_upgrade_hook	<ul style="list-style-type: none"> 各マスターのアップグレード 後 に実行しますが、そのサービスまたはシステムの再起動 前 に実行します。 この Hook は 各マスター に対して連続して実行されます。 タスクが異なるホストに対して実行される必要がある場合、該当するタスクは delegate_to または local_action を使用する必要があります。
openshift_master_upgrade_post_hook	<ul style="list-style-type: none"> 各マスターをアップグレードし、そのサービスまたはシステムが再起動した 後 に実行します。 この Hook は 各マスター に対して連続して実行されます。 タスクが異なるホストに対して実行される必要がある場合、該当するタスクは delegate_to または local_action を使用する必要があります。

表2.2 ノードのアップグレードフック

フック名	説明
openshift_node_upgrade_pre_hook	<ul style="list-style-type: none"> 各ノードのアップグレード 前 に実行します。 このフックは ノードごと に順に実行されます。 タスクが異なるホストに対して実行される必要がある場合、該当するタスクは delegate_to または local_action を使用する必要があります。
openshift_node_upgrade_hook	<ul style="list-style-type: none"> 各ノードのアップグレード 後、かつスケジュール対象として再びマークされる 前 に実行します。 このフックは ノードごと に順に実行されます。 タスクが異なるホストに対して実行される必要がある場合、該当するタスクは delegate_to または local_action を使用する必要があります。

フック名	説明
openshift_node_upgrade_post_hook	<ul style="list-style-type: none"> ● 各ノードのアップグレード後に実行します。これは、最後のノードアップグレードのアクションとなります。 ● このフックはノードごとに順に実行されます。 ● タスクが異なるホストに対して実行される必要がある場合、該当するタスクは delegate_to または local_action を使用する必要があります。

2.3.5. OpenShift Container Platform のアップグレードにおける特別な考慮事項

OpenShift Container Platform クラスターが混在環境または gcePD ストレージを使用する場合、アップグレードの前に追加の手順を実行する必要があります。

Red Hat Enterprise Linux (RHEL) および RHEL Atomic Host が設定された環境などの混在環境をアップグレードする前に、**openshift_pkg_version** と **openshift_image_tag** パラメーターの両方についてインベントリーファイルに値を設定します。これらの値を設定すると、クラスター内のすべてのノードが OpenShift Container Platform の同一のバージョンを実行するようになります。これは OpenShift Container Platform 2 から OpenShift Container Platform 3 のメジャー更新の場合のベストプラクティスですが、マイナーバージョンのアップグレードの場合、これらの値を設定することは必須となっています。

たとえば、OpenShift Container Platform 3.9 から OpenShift Container Platform 3.10 にアップグレードするには、以下のパラメーターおよび値を設定します。

```
openshift_pkg_version=-3.10.16
openshift_image_tag=v3.10.16
```



注記

他の混在していない環境では、これらのパラメーターも表示される場合があります。

2.3.5.1. 大規模なアップグレードに関する特別な考慮事項

10 以上のワーカーノードおよび数千のプロジェクトおよび Pod を含む大規模なアップグレードの場合、API オブジェクトストレージの移行は、アップグレード Playbook の実行前と、アップグレードが正常に実行された後に再度実行する必要があります。そうしないと、アップグレードプロセスは失敗します。

詳細の説明については、「[Recommendations for large-scale OpenShift upgrades](#)」の「[Running the pre- and post- API server model object migration outside of the upgrade window](#)」セクションを参照してください。

2.3.5.2. gcePD を使用する場合の特別な考慮事項

デフォルトの gcePD ストレージプロバイダーは RWO (Read-Write Only) アクセスモードを使用するため、レジストリーでローリングアップグレードを実行したり、レジストリーを複数の Pod に拡張した

りすることはできません。そのため、OpenShift Container Platform のアップグレード時に以下の環境変数を Ansible インベントリーファイルに指定する必要があります。

```
[OSEv3:vars]
```

```
openshift_hosted_registry_storage_provider=gcs
openshift_hosted_registry_storage_gcs_bucket=bucket01
openshift_hosted_registry_storage_gcs_keyfile=test.key
openshift_hosted_registry_storage_gcs_rootdirectory=/registry
```

2.4. 最新 OPENSIFT CONTAINER PLATFORM リリースへのアップグレード

既存の OpenShift Container Platform 3.10 または 3.11 クラスターを最新の 3.11 リリースにアップグレードするには、以下を実行します。

1. **アップグレードの準備**を行い、最新のアップグレード Playbook を使用していることを確認します。
2. インベントリーファイルの **openshift_deployment_type** パラメーターが **openshift-enterprise** に設定されていることを確認します。
3. ホストのローリング再起動および完全なシステムの再起動を有効にするには、インベントリーファイルの **openshift_rolling_restart_mode** パラメーターを **system** に設定します。これを設定しない場合、サービスはマスターで再起動されますが、システムは再起動しません。



注記

openshift_rolling_restart_mode はマスターホストでのみ機能します。

詳細については、「[Configuring Cluster Variables](#)」を参照してください。

4. クラスターイメージレジストリーの場所を変更するために **oreg_url** パラメーターを変更した場合、**imageconfig** Playbook を実行してイメージの場所を更新する必要があります。

```
$ cd /usr/share/ansible/openshift-ansible
$ ansible-playbook -i </path/to/inventory/file> \
  playbooks/openshift-node/imageconfig.yml
```

5. ノードをアップグレードします。
インベントリーファイルがデフォルトの **/etc/ansible/hosts** 以外の場所に置かれている場合、**-i** フラグを追加してその場所を指定します。以前に **atomic-openshift-installer** コマンドを使用してインストールを実行したことがある場合は、**~/.config/openshift/hosts** で最後に使用されたインベントリーファイルを確認できます。

- コントロールプレーンおよび複数のノードを単一フェーズでアップグレードするには、**upgrade.yml** Playbook を実行します。

```
$ cd /usr/share/ansible/openshift-ansible
$ ansible-playbook -i </path/to/inventory/file> \
  playbooks/byo/openshift-cluster/upgrades/v3_11/upgrade.yml
```

- コントロールプレーンおよび複数のノードを別々のフェーズでアップグレードするには、以下を実行します。
 - a. コントロールプレーンを実行するには、`upgrade_control_plane.yml` Playbook を実行します。

```
$ cd /usr/share/ansible/openshift-ansible
$ ansible-playbook -i </path/to/inventory/file> \
  playbooks/byo/openshift-cluster/upgrades/v3_11/upgrade_control_plane.yml
```

- b. ノードを実行するには、`upgrade_nodes.yml` Playbook を実行します。

```
$ cd /usr/share/ansible/openshift-ansible
$ ansible-playbook -i </path/to/inventory/file> \
  playbooks/byo/openshift-cluster/upgrades/v3_11/upgrade_nodes.yml \
  [-e <customized_node_upgrade_variables>] 1
```

- 1** 必要な `<customized_node_upgrade_variables>` については、「[ノードアップグレードのカスタマイズ](#)」を参照してください。

「[ノードアップグレードのカスタマイズ](#)」で説明されているようにノードをグループでアップグレードしている場合、すべてのノードがアップグレードされるまで `upgrade_nodes.yml` Playbook を実行し続けます。

6. この手順のステップ 3 で `openshift_rolling_restart variable=system` を使用してマスターホストの自動化された再起動を有効にしなかった場合は、アップグレードの完了後にすべてのマスターホストをすべてのノードホストと共に手動で再起動できます。ホストの再起動はオプションです。
7. 集計ロギングを使用する場合は、[EFK ロギングスタックをアップグレード](#)します。
8. クラスターメトリクスを使用する場合は、[クラスターメトリクスをアップグレード](#)します。
9. [アップグレードを検証](#)します。

2.5. コンテナ化された GLUSTERFS を使用する場合は OPENSIFT CONTAINER PLATFORM のアップグレード

OpenShift Container Platform をアップグレードする際に、GlusterFS Pod が実行されるノードのセットをアップグレードする必要があります。ただし、これらの Pod は `daemonset` の一部として実行されるため、GlusterFS Pod を終了したり、退避したりするために `drain` または `unschedule` コマンドを使用することはできません。データ可用性およびクラスターの破損の問題を避けるには、GlusterFS Pod ホストするノードを一度に1つずつアップグレードして、アップグレードプロセスが GlusterFS を実行するノードで完了してから次のノードでのアップグレードを開始するようにしてください。

コンテナ化された GlusterFS を使用する場合に OpenShift Container Platform をアップグレードするには、以下を実行します。

1. [コントロールプレーンをアップグレード](#)します (マスターノードおよび `etcd` ノード)。
2. 標準 `infra` ノード (ルーター、レジストリー、ロギング、およびメトリクス) をアップグレードします。



注記

これらのグループのノードのいずれかが GlusterFS を実行している場合、この手順のステップ 4 を同時に実行します。GlusterFS ノードは (**app** や **infra** などの) クラス内の他のノードと共に一度に1つずつアップグレードする必要があります。

3. アプリケーションコンテナを実行する標準ノードをアップグレードします。



注記

これらのグループのノードのいずれかが GlusterFS を実行している場合、この手順のステップ 4 を同時に実行します。GlusterFS ノードは (**app** や **infra** などの) クラス内の他のノードと共に一度に1つずつアップグレードする必要があります。

4. GlusterFS を実行する OpenShift Container Platform ノードを一度に1つずつアップグレードします。
 - a. ラベルをアップグレードする必要のあるノードに追加して、一度に1つのノードのみをアップグレードするようにします。

```
$ oc label node <node_name> type=upgrade
```

- b. 再起動する必要のある GlusterFS Pod を終了しないでください。
- c. 単一 GlusterFS ノードでアップグレード Playbook を実行するには、**-e openshift_upgrade_nodes_label="type=upgrade"** を使用します。



注記

GlusterFS Pod は終了しないでください。

- d. GlusterFS Pod が再生成され、表示されるまで待機します。
- e. 基本的なヘルスチェックは、各 Pod の再起動後に実行され、パスすることが確認されます。
- f. **oc rsh** を Pod に実行し、すべてのボリュームが自動修復されていることを確認します。

```
$ oc rsh <GlusterFS_pod_name>
$ for vol in `gluster volume list`; do gluster volume heal $vol info; done
```

すべてのボリュームが自動修復され、未処理のタスクがないことを確認します。**heal info** コマンドは所定ボリュームの自動修復プロセスのすべての未処理エントリを一覧表示します。ボリュームは、ボリュームの **Number of entries** が **0** の場合に自動修正済みとみなされます。

- g. アップグレードラベルを削除し、次の GlusterFS ノードに移動します。

```
$ oc label node <node_name> type-
```

2.6. オプションコンポーネントのアップグレード

EFK ロギングスタックまたはクラスターメトリクスをインストールしている場合、コンポーネントを別々にアップグレードする必要があります。

2.6.1. EFK ロギングスタックのアップグレード

既存の EFK ロギングスタックデプロイメントをアップグレードするには、パラメーターを確認してから `openshift-logging/config.yml` Playbook を実行します。



注記

アップグレードにより、**logging-fluentd** および **logging-curator** ConfigMap を置き換えることができます。現在の **logging-fluentd** および **logging-curator** ConfigMap を保持する必要がある場合は、[インベントリーホストファイル](#)で、**openshift_logging_fluentd_replace_configmap** および **openshift_logging_curator_replace_configmap** パラメーターを **no** に設定します。

EFK のアップグレードも Elasticsearch をバージョン 2 からバージョン 5 にアップグレードします。Elasticsearch 5 での変更点についての重要な情報については、「[Elasticsearch breaking changes](#)」を参照してください。

Elasticsearch 5 では、インデックス構造に大幅な変更が加われました。これまでは、Elasticsearch はドット文字の . の使用がフィールド名で許可されていました。バージョン 5 では、Elasticsearch は Elasticsearch フィールド名のドットをネストされた構造として解釈します。ドットのあるフィールドがある場合、ドットの後に文字列がフィールドのタイプとして解釈され、これはアップグレード時のマッピングの競合を生じさせます。

潜在的な競合を特定しやすくするために、OpenShift Container Platform では Elasticsearch フィールドを検査し、名前にドットが含まれるフィールドがあるかどうかを判別するためのスクリプトを提供しています。

たとえば、以下のフィールドは Elasticsearch 2 で許可されていました。

```
{
  "field": 123 // "field" is of type number
}

// Any dot in field name is treated as any other valid character in the field name.
// It is just part of the field name.
{
  "field.name": "Bob" // "field.name" is of type String
```

Elasticsearch 5 以降では、**field** 文字列がフィールドとなり、**name** 文字列がフィールドのタイプになります。

```
{
  "field": 123 // "field" is of type number
}

// Any dot in field name is always interpreted as nested structure.
{
  "field" : { // "field" is of type Object
    "name": "Bob" // "name" is of type String
  }
}
```

この場合、アップグレードにより **field** フィールドに 2 つの異なるタイプが組み込まれますが、これは許可されません。

これらの競合するインデックスを保持する必要がある場合、データのインデックスを再作成し、競合するデータ構造を取り除くように記述を変更する必要があります。詳細は、「[Upgrading fields with dots to 5.x](#)」を参照してください。

2.6.1.1. フィールドのフィールド名にドットがあるかどうかの判別

以下のスクリプトを実行して、インデックスに名前にドットが含まれるフィールドが含まれるかどうかを判別することができます。



注記

以下のコマンドは **jq** JSON プロセッサを使用して必要なデータで直接取得します。Red Hat Enterprise Linux (RHEL) のバージョンによっては、**jq** のパッケージを提供しない可能性があります。その場合、外部ソースからこれをインストールする必要がある場合があります。

```
oc exec -c elasticsearch -n $LOGGING_NS $pod -- es_util --query='_mapping?
pretty&filter_path=**.mappings*.properties' \
| jq '.[].mappings[].properties | keys' \
| jq .[] \
| egrep -e "\."
```

アップグレードパスは、インデックスにドットのあるフィールドが含まれるか、またはドットのあるフィールドが含まれないかどうかによって異なります。

2.6.1.2. フィールドのフィールド名にドットが含まれる場合のアップグレード

上記のスクリプトが、インデックスに名前にドットが含まれるフィールドが含まれることを示す場合、以下の手順で問題を修正し、アップグレードを実行します。

EFK スタックをアップグレードするには、以下を実行します。

1. [ロギング Ansible 変数を指定する](#) 方法を確認し、少なくとも **[OSEv3:vars]** セクション内で以下の必要な変数を設定するように Ansible インベントリーファイルを更新します。

```
[OSEv3:vars]
```

```
openshift_logging_install_logging=true 1
```

- 1** ロギングスタックをアップグレードする機能を有効にします。

2. 「[Specifying Logging Ansible Variables](#)」で説明されているように、デフォルトの値を上書きするために指定する必要があるその他の **openshift_logging_*** 変数を更新します。
openshift_logging_elasticsearch_replace_configmap パラメーターを **true** に設定し、**logging-elasticsearch** ConfigMap を現在のデフォルト値に置き換えることができます。場合によっては、古い ConfigMap を使用すると、アップグレードが失敗することがあります。デフォルトは **false** に設定されます。詳細は、「[specify logging Ansible variables](#)」のパラメーターを参照してください。

- Fluentd Pod のスケジュールを解除し、データの取り込みを停止し、クラスターの状態が変更されないようにします。
たとえば、Fluentd Pod のノードセクターをいずれのノードにも一致しないセクターに変更することができます。

```
oc patch daemonset logging-fluentd -p '{"spec": {"template": {"spec": {"nodeSelector": {"non-existing": "true"}}}}}'
```

- リンクを実行します (すべての関連するインデックスでの [Elasticsearch インデックスのフラッシュ](#))。フラッシュプロセスはメモリーからディスクまでのすべてのログを維持します。これにより、アップグレード時の Elasticsearch がシャットダウンしてログが失われるのを防ぐことができます。
- オンラインまたはオフラインのバックアップを実行します。
 - クラスター全体で特定の Elasticsearch インデックスの [オンラインバックアップ](#) を実行します。
 - オフラインバックアップを実行します。
 - すべての Elasticsearch DeploymentConfig を **0** にスケールダウンします。

```
$ oc scale dc <name> -n openshift-logging --replicas=0
```
 - 組織に適した方法を使用して、外部永続ボリュームをバックアップします。
- ドット文字を含むファイル名の場合、アップグレード前に以下のいずれかのアクションを実行する必要があります。
 - インデックスの削除。これは、アップグレード時のマッピングの競合を避けるための効果的な方法です。
 - データのインデックスの再作成、および競合するデータ構造を取り除くための記述の変更。この方法ではデータを保持します。マッピングの競合についての詳細は、Elasticsearch ドキュメントの「[Mapping changes](#)」を参照してください。
- オンラインまたはオフラインバックアップを繰り返します。
- 「[EFK スタックのデプロイ](#)」の説明に従って `openshift-logging/config.yml` Playbook を実行し、ロギングのアップグレードを完了します。ロギングデプロイメントをアップグレードするには、新規 OpenShift Container Platform バージョンのインストール Playbook を実行します。

2.6.1.3. フィールドにドットが含まれない場合のアップグレード

[上記のスクリプト](#)が、インデックスに名前にドットが含まれるフィールドが含まれていないことを示す場合、以下の手順でアップグレードを実行します。

- オプションとして、Fluentd Pod のスケジュールを解除し、Elasticsearch Pod をスケールダウンしてデータの取り込みを停止し、クラスターの状態が変更されないようにします。
たとえば、Fluentd Pod のノードセクターをいずれのノードにも一致しないセクターに変更することができます。

```
oc patch daemonset logging-fluentd -p '{"spec": {"template": {"spec": {"nodeSelector": {"non-existing": "true"}}}}}'
```

- オプションで、オンラインまたはオフラインバックアップを実行します。

- クラスター全体で特定の Elasticsearch インデックスの[オンラインバックアップ](#)を実行します。
- オフラインバックアップを実行します。
 - a. すべての Elasticsearch DeploymentConfig を **0** にスケールダウンします。


```
$ oc scale dc <name> -n openshift-logging --replicas=0
```
 - b. 組織に適した方法を使用して、外部永続ボリュームをバックアップします。
- 3. 「[EFK スタックのデプロイ](#)」の説明に従って `openshift-logging/config.yml` Playbook を実行し、ロギングのアップグレードを完了します。ロギングデプロイメントをアップグレードするには、新規 OpenShift Container Platform バージョンのインストール Playbook を実行します。
- 4. オプションで、[Elasticsearch 復元モジュール](#)を使用して、スナップショットから Elasticsearch インデックスを復元します。

2.6.2. クラスターメトリクスのアップグレード

既存のクラスターメトリクスのデプロイメントをアップグレードするには、パラメーターを確認してから `openshift-metrics/config.yml` Playbook を実行します。

1. [メトリクス Ansible 変数を指定する](#)方法を確認し、少なくとも `[OSEv3:vars]` セクション内で以下の必要な変数を設定するように Ansible インベントリーファイルを更新します。

```
[OSEv3:vars]
```

```
openshift_metrics_install_metrics=true 1
openshift_metrics_hawkular_hostname=<fqdn> 2
openshift_metrics_cassandra_storage_type=(emptydir|pv|dynamic) 3
```

- 1** メトリクスデプロイメントをアップグレードする機能を有効にします。
 - 2** Hawkular Metrics ルートに使用します。完全修飾ドメイン名を指定します。
 - 3** 以前のデプロイメントと同じタイプを指定します。
2. 「[Specifying Logging Ansible Variables](#)」で説明されているように、デフォルトの値を上書きするために指定する必要があるその他の `openshift_metrics_*` 変数を更新します。
 3. 「[メトリクスデプロイメントのデプロイ](#)」の説明に従って `openshift-metrics/config.yml` Playbook を実行し、メトリクスのアップグレードを完了します。メトリクスデプロイメントをアップグレードするには、新規 OpenShift Container Platform バージョンのインストール Playbook を実行します。

2.7. アップグレードの検証

以下の点を確認してください。

- クラスターが正常である。
- マスター、ノード、etcd サービスまたは静的 Pod が正常に実行されている。

- OpenShift Container Platform、**docker-registry**、およびルーターバージョンが正しい。
- 元のアプリケーションが依存として利用可能な状態で、新規アプリケーションの作成が可能である。
- **oc adm diagnostics** を実行してもエラーが生じない。

アップグレードを検証するには、以下を確認します。

1. すべてのノードに **Ready** のマークが付けられていることを確認します。

```
# oc get nodes
NAME                STATUS  ROLES    AGE   VERSION
master1.example.com Ready   master   47d   v1.11.0+d4cacc0
master2.example.com Ready   master   47d   v1.11.0+d4cacc0
master3.example.com Ready   master   47d   v1.11.0+d4cacc0
infra-node1.example.com Ready   infra    47d   v1.11.0+d4cacc0
infra-node2.example.com Ready   infra    47d   v1.11.0+d4cacc0
node1.example.com   Ready   compute  47d   v1.11.0+d4cacc0
node2.example.com   Ready   compute  47d   v1.11.0+d4cacc0
```

2. コントロールプレーンの静的 Pod が実行されていることを確認します。

```
# oc get pods -n kube-system
NAME                                READY  STATUS   RESTARTS  AGE
master-api-master1.example.com      1/1    Running  4         1h
master-controllers-master1.example.com 1/1    Running  3         1h
master-etcd-master1.example.com     1/1    Running  6         5d
[...]
```

3. 予想されるバージョンの **docker-registry** および **router** イメージを実行していることを確認します (デプロイされている場合)。

```
# oc get -n default dc/docker-registry -o json | grep "image\"
  "image": "openshift3/ose-docker-registry:v3.11",
# oc get -n default dc/router -o json | grep "image\"
  "image": "openshift3/ose-haproxy-router:v3.11",
```

4. マスター上で診断ツールを使用し、一般的な問題を検索します。

```
# oc adm diagnostics
...
[Note] Summary of diagnostics execution:
[Note] Completed with no errors or warnings seen.
```

第3章 BLUE-GREEN クラスターアップグレードの実行



注記

このトピックでは、インプレースアップグレード方法に代わるノードホストのアップグレード方法について説明します。

Blue-Green デプロイメントのアップグレード方式はインプレース方式と同様のフローに基づいて実行されます。この場合もマスターおよび etcd サーバーが最初にアップグレードされます。ただし、インプレースアップグレードを実行する代わりに、新規ノードホストの並列環境が作成されます。

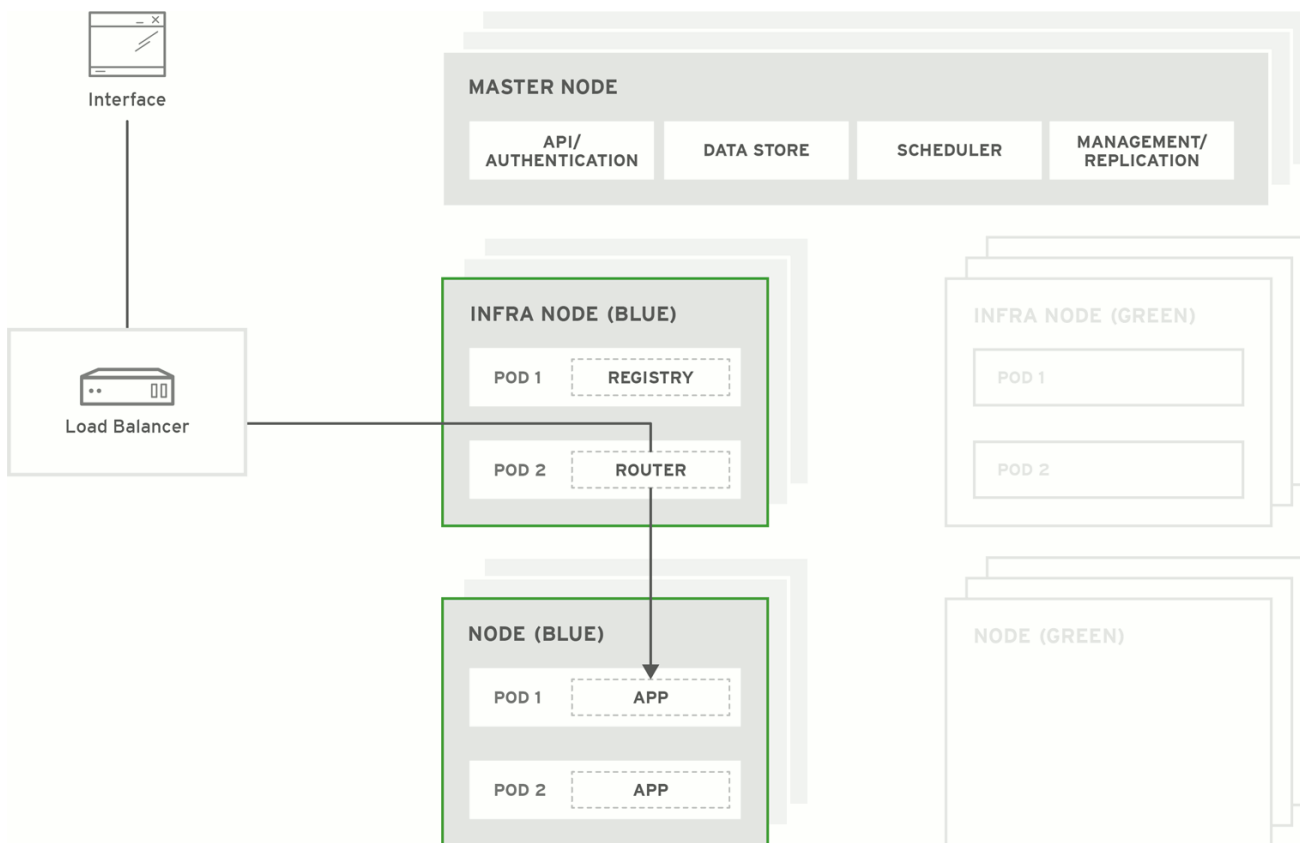
この方式では、管理者は新規デプロイメントの検証後、古いノードホストセット (例: 「Blue」 デプロイメント) から新規セット (例: 「Green デプロイメント) にトラフィックを切り換えることができます。問題が検出される場合も、古いデプロイメントへのロールバックを簡単かつすぐに実行できます。

Blue-Green はソフトウェアについての証明済みの有効なデプロイストラテジーであるものの、トレードオフも常にあります。すべての環境が Blue-Green デプロイメントの適切な実行に必要な同じアップタイム要件を満たす訳ではなく、必要なリソースがある訳でもありません。

OpenShift Container Platform 環境では、Blue-Green デプロイメントに最も適しているのはノードホストです。すべてのユーザープロセスはこれらのシステムで実行され、OpenShift Container Platform インフラストラクチャーの重要な部分もこれらのリソースで自己ホストされます。アップタイムはこれらのワークロードで最も重要であり、Blue-Green デプロイメントによって複雑性が増すとしてもこれを確保できる場合には問題になりません。

この方法の実際の実装は要件に応じて異なります。通常、主な課題は、この方法を容易に実行するための追加の容量を確保することにあります。

図3.1 Blue-Green デプロイメント



OPENSIFT_415490_0217

3.1. BLUE-GREEN アップグレードの準備

「[インプレースアップグレード](#)」で説明されている方法を使用してマスターと etcd ホストをアップグレードした後、以下のセクションを参照して残りのノードホストの Blue-Green アップグレードの環境を準備します。

3.1.1. ソフトウェアエンタイトルメントの共有

管理者は Blue-Green デプロイメント間で Red Hat ソフトウェアエンタイトルメントを一時的に共有するか、または Red Hat Satellite などのシステムでインストールコンテンツへのアクセスを提供する必要があります。これは、以前のノードホストからのコンシューマー ID を共有して実行できます。

1. アップグレードされるそれぞれの古いノードホストで、コンシューマー ID であるその **system identity** 値を書き留めておいてください。

```
# subscription-manager identity | grep system
system identity: 6699375b-06db-48c4-941e-689efd6ce3aa
```

2. 古いノードホストに置き換わるそれぞれの新規 RHEL 7 または RHEL Atomic Host 7 システムで、直前の手順のそれぞれのコンシューマー ID を使用して登録を行います。

```
# subscription-manager register --consumerid=6699375b-06db-48c4-941e-689efd6ce3aa
```

3.1.2. Blue ノードのラベリング

実稼働環境の現在のノードホストに **blue** または **green** のいずれかのラベルが付けられていることを確認する必要があります。以下の例では、現在の実稼働環境は **blue** となり、新規の環境は **green** になります。

1. クラスターに認識されるノード名の現在の一覧を取得します。

```
$ oc get nodes
```

2. 現在の実稼働環境内にあるマスター以外のノードホスト (コンピューターノード) および専用のインフラストラクチャーノードに、**color=blue** のラベルを付けます。

```
$ oc label node --selector=node-role.kubernetes.io/compute=true color=blue
```

```
$ oc label node --selector=node-role.kubernetes.io/infra=true color=blue
```

上記のコマンドでは、**--selector** フラグを使用して、関連するノードラベルでクラスターのサブセットと一致させるように設定され、すべての一致項目には **color=blue** のラベルが付けられます。

3.1.3. Green ノードの作成およびラベリング

等しい数の新規ノードホストを既存のクラスターに追加して Green 環境を作成します。

1. 「[Adding Hosts to an Existing Cluster](#)」に説明されている手順を使用して、新規ノードホストを追加します。この手順にある **[new_nodes]** グループでインベントリーファイルを更新する際に、これらの変数が設定されていることを確認します。

- ノードが **健全**であるとみなされるまでワークロードのスケジューリングを遅らせるために (健全性については後の手順で検証します)、それぞれの新規ノードホストに

openshift_schedulable=false 変数を設定し、これらが初期の時点でスケジュール対象外となるようにします。

2. 新規ノードをデプロイしてから、新規のノードごとに **color=green** ラベルを適用します。

```
$ oc label node <node_name> color=green
```

3.1.4. Green ノードの検証

新規 Green ノードが健全な状態にあることを確認します。

1. 新規ノードがクラスター内で検出され、**Ready,SchedulingDisabled** 状態にあることを確認します。

```
$ oc get nodes
```

NAME	STATUS	ROLES	AGE
node4.example.com	Ready,SchedulingDisabled	compute	1d

2. Green ノードに適切なラベルがあることを確認します。

```
$ oc get nodes --show-labels
```

NAME	STATUS	ROLES	AGE	LABELS
node4.example.com	Ready,SchedulingDisabled	compute	1d	beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,color=green,kubernetes.io/hostname=m01.example.com,node-role.kubernetes.io/compute=true

3. クラスターの診断チェックを実行します。

```
$ oc adm diagnostics
```

```
[Note] Determining if client configuration exists for client/cluster diagnostics
Info: Successfully read a client config file at '/root/.kube/config'
Info: Using context for cluster-admin access: 'default/m01-example-com:8443/system:admin'
[Note] Performing systemd discovery

[Note] Running diagnostic: ConfigContexts[default/m01-example-com:8443/system:admin]
Description: Validate client config context is complete and has connectivity
...
[Note] Running diagnostic: CheckExternalNetwork
Description: Check that external network is accessible within a pod

[Note] Running diagnostic: CheckNodeNetwork
Description: Check that pods in the cluster can access its own node.

[Note] Running diagnostic: CheckPodNetwork
Description: Check pod to pod communication in the cluster. In case of ovs-subnet
network plugin, all pods
should be able to communicate with each other and in case of multitenant network plugin,
pods in non-global projects
should be isolated and pods in global projects should be able to access any pod in the cluster
and vice versa.
```

[Note] Running diagnostic: CheckServiceNetwork

Description: Check pod to service communication in the cluster. In case of ovs-subnet network plugin, all pods should be able to communicate with all services and in case of multitenant network plugin, services in non-global projects should be isolated and pods in global projects should be able to access any service in the cluster.

...

3.2. GREEN ノードの準備

Pod を Blue 環境から Green に移行するには、必要なコンテナイメージをプルする必要があります。

ネットワークのレイテンシーやレジストリーへの負荷により、環境の容量が十分でない場合に遅延が生じる可能性があります。新規 Pod デプロイメントを新規ノードに対してトリガーするために新規イメージストリームをインポートすることにより、実行中のシステムへの影響を最小限に抑えることができます。

メジャーリリースの OpenShift Container Platform (および一部の非同期エラー更新) では、Source-to-Image (S2I) のユーザー向けにビルダーイメージの新規イメージストリームを導入しています。インポート時に、**イメージ変更トリガー**で設定されるビルドおよびデプロイメントが自動的に作成されます。

ビルドをトリガーする別の利点として、各種のビルダーイメージ、Pod インフラストラクチャーイメージおよびデプロイヤーなどの多数の補助イメージをすべてのノードホストに効果的に取り込める点があります。Green ノードは予想される負荷の増加に対応できるように準備され、その他の残りはノードの退避時に迅速に移行します。

アップグレードプロセスに進む準備ができたなら、以下の手順に従って Green ノードを準備します。

1. Green ノードをスケジュール対象 (Schedulable) に設定し、新規 Pod がそれらのノードにデプロイさせるようにします。

```
$ oc adm manage-node --schedulable=true --selector=color=green
```

2. Blue ノードをスケジュール対象外 (Unschedulable) に設定し、新規 Pod がそれらのノードで実行されないようにします。

```
$ oc adm manage-node --schedulable=false --selector=color=blue
```

3. **node-role.kubernetes.io/infra=true** ラベルを使用するように、レジストリーとルーターデプロイメント設定のノードセレクターを更新します。この変更により、レジストリーおよびルーター Pod を新規のインフラストラクチャーノードに配置する新規のデプロイメントが起動します。

- a. **docker-registry** デプロイメント設定を編集します。

```
$ oc edit -n default dc/docker-registry
```

- b. **nodeSelector** パラメーターを以下の値を使用するように更新し (引用符で囲まれた **"true"** を使用します)、変更を保存します。

```
nodeSelector:
  node-role.kubernetes.io/infra: "true"
```

- c. **router** デプロイメント設定を編集します。

```
$ oc edit -n default dc/router
```

- d. **nodeSelector** パラメーターを以下の値を使用するように更新し (引用符で囲まれた **"true"** を使用します)、変更を保存します。

```
nodeSelector:
  node-role.kubernetes.io/infra: "true"
```

- e. 新規インフラストラクチャーノードで **docker-registry** および **router** Pod が実行中で、Ready の状態であることを確認します。

```
$ oc get pods -n default -o wide
```

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
docker-registry-2-b7xnb	1/1	Running	0	18m	10.128.0.188	infra-node3.example.com
router-2-mvq6p	1/1	Running	0	6m	192.168.122.184	infra-node4.example.com

- デフォルトのイメージストリームとテンプレートを更新します。
- 最新のイメージをインポートします。このプロセスで多数のビルドがトリガーされる可能性があります、ビルドは Green ノードで実行されるため、これが Blue デプロイメントのトラフィックに影響を与えることはありません。
- クラスター内のすべての namespace (プロジェクト) でのビルドプロセスをモニターするには、以下を実行します。

```
$ oc get events -w --all-namespaces
```

大規模な環境では、ビルドはほとんど停止することがありません。しかしながら、管理上のイメージのインポートによって大きな増減が生じます。

3.3. BLUE ノードの退避および使用停止

大規模デプロイメントでは、退避の調整方法を定めるのに役立つ他のラベルを使用できます。ダウンタイムを防ぐための最も保守的な方法として、一度に1つのノードホストを退避する方法があります。

サービスがゾーンの非アフィニティーを使用して複数の Pod で構成されている場合、ゾーン全体を一度に退避できます。使用されるストレージボリュームが新規ゾーンで利用可能であることを確認する必要があります。クラウドプロバイダーのドキュメントでその方法を確認してください。

ノードホストの退避はノードサービスの停止時に常にトリガーされます。ノードのラベリングは非常に重要であり、ノードに誤ったラベルが付けられていたり、コマンドが汎用化されたラベルの付いたノードで実行される場合は問題が生じる可能性があります。マスターホストにも **color=blue** のラベルが付けられている場合には注意が必要です。

アップグレードプロセスに進む準備ができたなら、以下の手順に従います。

- 以下のコマンドで、Blue ノードをすべて退避して削除します。

```
$ oc adm manage-node --selector=color=blue --evacuate  
$ oc delete node --selector=color=blue
```

2. Blue ノードホストから Pod がなくなり、Blue ノードホストが OpenShift Container Platform から削除され後は、電源をオフにしても問題ありません。安全上の措置として、アップグレードに問題がないことを確認してからホストの電源をオフにしてください。
 - a. それぞれの古いホストの登録を解除します。

```
# subscription-manager clean
```

- b. ホストに保存されている役立つスクリプトまたは必要なファイルをバックアップします。
- c. アップグレードが正常に実行されたことを確認したら、それらのホストを削除します。

第4章 オペレーティングシステムの更新

メジャーリリース間でのアップグレードや、マイナーリリースのソフトウェアの更新のいずれかによってホストでオペレーティングシステム (OS) を更新すると、それらのマシンで実行されている OpenShift Container Platform ソフトウェアに影響が及びます。とくに、これらの更新は、OpenShift Container Platform で動作する必要のある **iptables** ルールまたは **ovs** フローに影響を与えます。

4.1. ホストでのオペレーティングシステムの更新

ホストで OS を安全にアップグレードするには、以下を実行します。

1. メンテナンスの準備のためにノードをドレイン (解放) します。

```
$ oc adm drain <node_name> --force --delete-local-data --ignore-daemonsets
```

2. 更新される必要のない重要なパッケージを保護するには、ホストに除外ルールを適用します。

```
# atomic-openshift-docker-excluder exclude
# atomic-openshift-excluder exclude
```

再起動により、ホストで最新バージョンが実行されていることを確認できます。これは、**container engine** および OpenShift Container Platform プロセスが再起動されていることを意味し、これにより、他のサービスのすべてのルールが正しいことの確認が強制的に実行されます。

```
# yum update
# reboot
```

ただし、ノードホストを再起動する代わりに、影響を受けるサービスを再起動するか、または **iptables** 状態を保持することができます。どちらのプロセスについても、「[OpenShift Container Platform iptables](#)」のトピックで説明されています。**ovs** フロールールは保存される必要がありませんが、OpenShift Container Platform ノードソフトウェアを再起動するとフロールールが固定されます。

3. ホストを再びスケジュール対象 (Schedulable) に設定するには、以下を実行します。

```
$ oc adm uncordon <node_name>
```

4.1.1. OpenShift Container Storage を実行するノードのアップグレード

OpenShift Container Storage を使用している場合、OpenShift Container Storage を実行する OpenShift Container Platform ノードを一度に1つずつアップグレードします。

1. 開始するには、OpenShift Container Storage がデプロイされているプロジェクトを再度呼び出します。
2. サービスの daemonset に設定されたノードと Pod セレクターを確認します。

```
$ oc get daemonset -n <project_name> -o wide
```



注記

-o wide を使用して Pod セレクターを出力に追加します。

これらのセクターは **NODE-SELECTOR** と **SELECTOR** にそれぞれ表示されます。以下のサンプルコマンドは **glusterfs=storage-host** と **glusterfs=storage-pod** をそれぞれ使用します。

3. daemonset のノードセクターが指定されている場合、ラベルを持つために、daemonset から Pod を実行しているホストを確認します。

```
$ oc get nodes --selector=glusterfs=storage-host
```

オペレーティングシステムがアップグレードされているノードを選択します。

4. daemonset ラベルをノードから削除します。

```
$ oc label node <node_name> glusterfs-
```

これにより、OpenShift Container Storage Pod はそのノードで終了します。

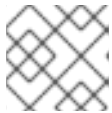
上記のように、ノードで OS をアップグレードできるようになりました。

1. ノードで OpenShift Container Storage Pod を再起動するには、daemonset ラベルでノードに再度ラベルを付けます。

```
$ oc label node <node_name> glusterfs=storage-host
```

2. OpenShift Container Storage Pod が再生成され、表示されるまで待機します。
3. daemonset の Pod セクターが指定される場合、新たに生成された Pod の名前を判別するには、アップグレードした OS のあるノードで実行されている Pod を検索します。

```
$ oc get pod -n <project_name> --selector=glusterfs=storage-pod -o wide
```



注記

-o wide を使用して、出力に Pod が実行されているホストを追加します。

4. **oc rsh** を Pod に実行し、Gluster Pod でボリュームの修復を確認します。

```
$ oc rsh <pod_name>
$ for vol in `gluster volume list`; do gluster volume heal $vol info; done
$ exit
```

すべてのボリュームが自動修復され、未処理のタスクがないことを確認します。**heal info** コマンドは所定ボリュームの自動修復プロセスのすべての未処理エントリを一覧表示します。ボリュームは、ボリュームの **Number of entries** が **0** の場合に自動修正済みとみなされます。ボリュームの追加の詳細情報については **gluster volume status <volume_name>** を使用します。**Online** 状態の場合は、すべてのブリックについて **Y** のマークを付ける必要があります。

第5章 クラスターのダウングレード

OpenShift Container Platform のアップグレードの後に、クラスターを前のバージョンにダウングレードする必要がある可能性があります。OpenShift Container Platform バージョン 3.11 からバージョン 3.10 へのダウングレードが可能です。



警告

OpenShift Container Platform バージョン 3.11 の初期リリースでは、ダウングレードしてもクラスターをバージョン 3.10 に完全に復元することができなくて、ダウングレードしないでください。

ダウングレードする必要がある場合には、Red Hat サポートに問い合わせ、最も適切なアクションを確認してください。



重要

クラスターのバージョン 3.10 へのダウングレードは、OpenShift Container Platform の [RPM ベースのインストール](#) についてのみサポートされています。ダウングレードを実行する際には、クラスター全体をオフラインにする必要があります。

5.1. バックアップの検証

1. `master-config.yaml` ファイル、`scheduler.json` ファイル、および `etcd` データディレクトリーのバックアップがマスターに存在していることを確認します。

```
/etc/origin/master/master-config.<timestamp>
/etc/origin/master/master.env
/etc/origin/master/scheduler.json
/var/lib/etcd/openshift-backup-xxxx
```

これらのファイルをアップグレードプロセス時に保存します。

2. アップグレードの準備をした時に作成した以下のファイルのコピーの場所を確認します。ノードとマスターホスト:

```
/etc/origin/node/node-config.yaml
```

`etcd` ホスト (`etcd` が同一の場所に配置されているマスターを含む)

```
/etc/etcd/etcd.conf
```

5.2. クラスターのシャットダウン

1. すべてのマスターおよびノードホストで、Pod 定義を削除し、ホストを再起動してマスターおよびノードサービスを停止します。

```
# mkdir -p /etc/origin/node/pods-stopped
# mv /etc/origin/node/pods/* /etc/origin/node/pods-stopped/
# reboot
```

5.3. RPM と静的 POD の削除

1. すべてのマスター、ノードおよび etcd メンバー (専用 etcd クラスターを使用している場合) で、以下のパッケージを削除します。

```
# yum remove atomic-openshift \
atomic-openshift-excluder \
atomic-openshift-hyperkube \
atomic-openshift-node \
atomic-openshift-docker-excluder \
atomic-openshift-clients
```

2. パッケージが正常に削除されたことを確認します。

```
# rpm -qa | grep atomic-openshift
```

3. コントロールプレーンホスト (マスターおよび etcd ホスト) で静的な Pod 定義を移動します。

```
# mkdir /etc/origin/node/pods-backup
# mv /etc/origin/node/pods/* /etc/origin/node/pods-backup/
```

4. 各ホストを再起動します。

```
# reboot
```

5.4. RPM の再インストール

1. OpenShift Container Platform 3.11 のリポジトリを無効にし、3.10 のリポジトリを再び有効にします。

```
# subscription-manager repos \
--disable=rhel-7-server-ose-3.11-rpms \
--enable=rhel-7-server-ose-3.10-rpms
```

2. 各マスターおよびノードホストで、以下のパッケージをインストールします。

```
# yum install atomic-openshift \
atomic-openshift-node \
atomic-openshift-docker-excluder \
atomic-openshift-excluder \
atomic-openshift-clients \
atomic-openshift-hyperkube
```

3. 各ホストでパッケージが正常にインストールされたことを確認します。

```
# rpm -qa | grep atomic-openshift
```

5.5. OPENSIFT CONTAINER PLATFORM サービスの再オンライン化

変更を終了した後に、OpenShift Container Platform をオンラインに戻します。

手順

1. それぞれの OpenShift Container Platform マスターで、バックアップからマスターおよびノード設定を復元し、すべての関連するサービスを有効にしてから再起動します。

```
# cp ${MYBACKUPDIR}/etc/origin/node/pods/* /etc/origin/node/pods/
# cp ${MYBACKUPDIR}/etc/origin/master/master.env /etc/origin/master/master.env
# cp ${MYBACKUPDIR}/etc/origin/master/master-config.yaml.<timestamp>
/etc/origin/master/master-config.yaml
# cp ${MYBACKUPDIR}/etc/origin/node/node-config.yaml.<timestamp>
/etc/origin/node/node-config.yaml
# cp ${MYBACKUPDIR}/etc/origin/master/scheduler.json.<timestamp>
/etc/origin/master/scheduler.json
# master-restart api
# master-restart controllers
```

2. 各 OpenShift Container Platform ノードで、必要に応じてノード設定マップを更新し、`atomic-openshift-node` サービスを有効化して再起動します。

```
# cp /etc/origin/node/node-config.yaml.<timestamp> /etc/origin/node/node-config.yaml
# systemctl enable atomic-openshift-node
# systemctl start atomic-openshift-node
```

5.6. ダウングレードの検証

ダウングレードを検証するには、以下を実行します。

1. すべてのノードに `Ready` のマークが付けられていることを確認します。

```
# oc get nodes
NAME                STATUS              AGE
master.example.com  Ready,SchedulingDisabled  165d
node1.example.com   Ready                165d
node2.example.com   Ready                165d
```

2. デプロイされている場合には、レジストリーおよびルーターが正常にダウングレードされたことを確認します。
 - a. **v3.10** バージョンの `docker-registry` と `router` イメージを実行していることを確認します。

```
# oc get -n default dc/docker-registry -o json | grep "\"image\""
"image": "openshift3/ose-docker-registry:v3.10",
# oc get -n default dc/router -o json | grep "\"image\""
"image": "openshift3/ose-haproxy-router:v3.10",
```

- b. `docker-registry` と `router` pod が実行中で、`Ready` の状態であることを確認します。

```
# oc get pods -n default
```

```
NAME                READY  STATUS  RESTARTS  AGE
docker-registry-2-b7xbn  1/1    Running  0          18m
router-2-mvq6p        1/1    Running  0          6m
```

3. [診断ツール](#)をマスターで使用し、一般的な問題を検索し、提案される方法を確認します。

```
# oc adm diagnostics
```

```
...
```

```
[Note] Summary of diagnostics execution:
```

```
[Note] Completed with no errors or warnings seen.
```