OpenShift Container Platform 3.9

クラスターのアップグレード

OpenShift Container Platform 3.9 クラスターのアップグレード

Last Updated: 2018-08-16

OpenShift Container Platform 3.9 クラスターのアップグレード

OpenShift Container Platform 3.9 クラスターのアップグレード

法律上の通知

概要

本書を活用した OpenShift Container Platform 3.9 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

1.1. クラスターのアップグレードについて

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 3.7 から 3.9 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 (3.9.z リリース) 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 3.9 Release Notes』を参照してください。

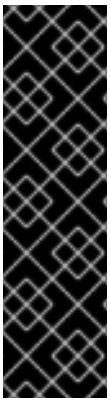
注記

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。

とくに記述がない限り、1 つのメジャーバージョン内のノードおよびマスターには 1 つのマイナーバージョンとの前方互換性および後方互換性があるため、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、すべての OpenShift Container Platform サービスが正常に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。



重要

OpenShift Container Platform 3.9 リリースには、Kubernetes 1.8 および 1.9 のマージされた各種機能および修正が含まれます。そのため、OpenShift Container Platform 3.7 からのアップグレードプロセスにより、クラスターは OpenShift Container Platform 3.9 に完全にアップグレードされた状態になります (3.8 リリースは「省略」されているように見えます)。実際には、OpenShift Container Platform 3.7 クラスターはまず 3.8 バージョンのパッケージにアップグレードされますが、その直後に OpenShift Container Platform 3.9 へのアップグレードプロセスが自動的に継続されます。クラスターのパッケージのバージョンが 3.8 であるのは OpenShift Container Platform 3.9 へのアップグレードが正常に終了するまでの期間になります。

1.2. アップグレード方式

OpenShift Container Platform クラスターのアップグレードを実行するために、自動と手動の 2 種類の方式を利用できます。

1.2.1. 自動方式

自動アップグレード方式は Ansible Playbook を使用して OpenShift Container Platform クラスターを自動化します。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。この方式では、インプレースアップグレードまたは Blue-Green デプロイメントのいずれかのアップグレードストラテジーを選択できます。

1.2.2. 手動方式

手動アップグレード方式は、自動化された Ansible ベースのアップグレード時のプロセスをステップに分類し、手動で実行するのに必要な同等のコマンドを提供します。この方式を使用する場合、インプレースのアップグレードストラテジーが使用されます。

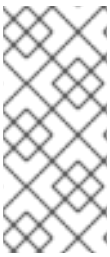
1.3. アップグレードストラテジー

自動アップグレード方式を使用する場合、OpenShift Container Platform クラスターのアップグレードについてインプレースアップグレードまたは Blue-Green デプロイメントという 2 つのストラテジーを使用できます。手動のアップグレード方式を使用する場合は、インプレースアップグレードが使用されます。

1.3.1. 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。

クイックインストールまたは 通常インストール (advanced installation) を使用してインストールしている場合、自動インプレースアップグレードを実行できます。または、手動によるインプレースアップグレードを実行することもできます。



注記

クイックインストールは OpenShift Container Platform 3.9 では非推奨となっており、今後のリリースでは完全に削除されます。

クイックインストールは 3.9 のみをインストールできます。これは 3.7 または 3.8 から 3.9 へのアップグレードには使用できません。

1.3.2. 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

2.1. 概要

通常インストール (advanced installation) を使用してインストールしていて、使用したインベントリーファイルが利用可能である場合、アップグレード Playbook を使用して OpenShift 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 3.9 リリースには、Kubernetes 1.8 および 1.9 のマージされた各種機能および修正が含まれます。そのため、OpenShift Container Platform 3.7 からのアップグレードプロセスにより、クラスターは OpenShift Container Platform 3.9 に完全にアップグレードされた状態になります (3.8 リリースは「省略」されているように見えます)。実際には、OpenShift Container Platform 3.7 クラスターはまず 3.8 バージョンのパッケージにアップグレードされますが、その直後に OpenShift Container Platform 3.9 へのアップグレードプロセスが自動的に継続されます。クラスターのパッケージのバージョンが 3.8 であるのは OpenShift Container Platform 3.9 へのアップグレードが正常に終了するまでの期間になります。

重要

OpenShift Container Platform 3.9 の時点で、クイックインストール方式は非推奨になっています。今後のリリースではこれは完全に削除されます。また、クイックインストーラーを使用したバージョン 3.7 から 3.9 へのアップグレードはサポートされていません。

3.7 から 3.9 へのコントロールプレーンの自動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作成する。
- API およびコントローラーを 3.7 から 3.8 に更新する。
- 内部データ構造を 3.8 に更新する。
- リカバリー用にすべての etcd データの 2 度目のバックアップを実行する。
- API およびコントローラーを 3.8 から 3.9 に更新する。
- 内部データ構造を 3.9 に更新する。
- デフォルトルーター (ある場合) を 3.7 から 3.9 に更新する。
- デフォルトレジストリー (ある場合) を 3.7 から 3.9 に更新する。
- デフォルトイメージストリームおよび InstantApp テンプレートを更新する。

3.7 から 3.9 へのノードの自動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ます。以下が実行されます。

- 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) します。
- 3.7 から3.9 にノードコンポーネントを更新します (openvswitch およびコンテナーランタイムを含む)。
- それらのノードをサービスに返します。

重要

- アップグレードに進む前に、すべての前提条件を満たしていることを確認します。前提条件を満たしていないと、アップグレードが失敗する可能性があります。
- GlusterFS を使用している場合は、「コンテナー化された GlusterFS を使用する場合の特別な考慮事項」を参照してからアップグレードを行ってください。
- GCE 永続ディスク (gcePD) を使用している場合は、「gcePD を使用する場合の特別な考慮事項」を参照してからアップグレードを行ってください。

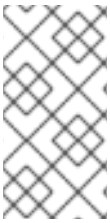
自動化されたアップグレード Playbook は、インベントリーファイルと共に **ansible-playbook** コマンドを使用して Ansible で直接実行されます。これは通常インストール (advanced installation) 方式と同様です。以下のいずれのシナリオでも、同じ **v3_9** アップグレード Playbook を使用することができます。

- 既存の OpenShift Container Platform 3.7 クラスターの 3.9 へのアップグレード
- 既存の OpenShift Container Platform 3.9 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

2.2. 自動アップグレードの準備

重要

アップグレード (クラスターの OpenShift Container Platform 3.9 へのアップグレード) の前に、クラスターはバージョン 3.7 の最新の非同期リリースにアップグレードされている必要があります。クラスターのバージョンが 3.7 よりも前の場合、徐々にアップグレードする必要があります (例: 3.5 から 3.6 にアップグレードしてから 3.6 から 3.7 にアップグレードする)。



注記

アップグレードを試行する前に、「環境ヘルスチェック」のガイダンスに従い、クラスターの健全性を確認します。これにより、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。

自動アップグレードを準備するには、以下を実行します。

1. RHSM から最新のサブスクリプションデータをプルします。

```
# subscription-manager refresh
```

2. OpenShift Container Platform 3.7 から 3.9 にアップグレードしている場合は、以下を実行します。

   a. 手動で 3.7 リポジトリーを無効にし、各マスターおよびノードホストで 3.8 および 3.9 リポジトリーの **両方** を有効にします。また、**rhel-7-server-ansible-2.4-rpms** リポジトリーを有効にする必要もあります。これは OpenShift Container Platform 3.9 以降の新規の要件です。

```
# subscription-manager repos --disable="rhel-7-server-ose-3.7-
```

```
rpms" \
    --enable="rhel-7-server-ose-3.8-rpms" \
    --enable="rhel-7-server-ose-3.9-rpms" \
    --enable="rhel-7-server-rpms" \
    --enable="rhel-7-server-extras-rpms" \
    --enable="rhel-7-server-ansible-2.4-rpms" \
    --enable="rhel-7-fast-datapath-rpms"
# yum clean all
```

b. 以前のバージョンの OpenShift Container Platform では、インストーラーがデフォルトでマスターホストにスケジュール対象外 (Unschedulable) のマークを付けました。これにより、Pod をこれらのホストに配置できませんでした。しかし、OpenShift Container Platform 3.9 より、マスターにはスケジュール対象 (Schedulable) のマークを付ける必要があります。これはアップグレード時に自動的に実行されます。
この要件により、デフォルトのノードセレクターもデフォルトで設定されることが必要になりました。これも、設定されていない場合はアップグレード時に自動的に設定されますが、アップグレード時に設定されない場合は設定されないままになります。どちらのシナリオの場合でも、マスターには **master** ノードロールのラベルが自動的に付けられ、マスターおよびインフラストラクチャー以外のノードには **compute** ノードロールのラベルが付けられます。

この重要な技術上の変更については、『OpenShift Container Platform 3.9 Release Notes』を参照し、これらのクラスターに及ぼす影響について確認してください。

- Masters Marked as Schedulable Nodes by Default (デフォルトでスケジュール対象ノードとマークされるマスター)
- Default Node Selector Set By Default and Automatic Node Labeling (デフォルトで設定されるデフォルトのノードセレクターおよび自動ノードラベリング)

c. OpenShift Container Platform 3.9 にアップグレードする前に、swap メモリーをクラスターで無効にしている必要があります。そうしない場合、アップグレードは失敗します。swap メモリーが Ansible インベントリーファイルの **openshift_disable_swap=false** を使用して有効にされているか、またはホストごとに手動で有効にされているかどうかについては、『クラスター管理ガイド』の「swap メモリーの無効化」を確認し、各ホストで無効にしてください。

3. アップグレードパスについては、各 RHEL 7 システムにある **atomic-openshift-utils** パッケージが最新バージョンであることを常に確認してください。これにより **openshift-ansible-*** パッケージも更新されます。

```
# yum update atomic-openshift-utils
```

4. (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、「Ansible インベントリーファイルの設定」を参照してください。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。
とくに、マスター設定ファイルの **admissionConfig** 設定に変更を加えた場合は、「Ansible 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、**ClusterResourceOverride** 設定を事前に手動で設定している場合には (「マスターでのオーバーコミットの設定」など)、Pod が **Pending** 状態のままになる可能性があります。

上記を確認した後に、以下のセクションでアップグレードプロセスのしくみをさらに確認し、カスタマイズする場合はアップグレードの追加のカスタマイズオプションについて決定します。アップグレードを実行する準備が整ったら、最新の OpenShift Container Platform 3.9 リリースへのアップグレードに進んでください。

2.2.1. コントロールプレーンとノードの別フェーズでのアップグレード

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**(マスターコンポーネント) および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

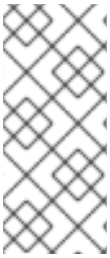
完全なアップグレードプロセスおよびこれらの Playbook を呼び出すタイミングについては、「最新の OpenShift Container Platform 3.9 リリースへのアップグレード」に説明されています。

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
- スタンドアロン etcd ホストで実行される Docker

ノードのみをアップグレードする場合、コントロールプレーンはすでにアップグレードされている必要があります。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

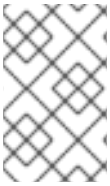


注記

マスターコンポーネントを実行するノードは、ノードサービスや Docker が実行されている場合でもノードのアップグレードフェーズには含まれず、それらはコントロールプレーンのアップグレードフェーズの一環として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 Docker 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。

2.2.2. ノードのアップグレードのカスタマイズ

アップグレードを単一または複数フェーズのいずれ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。



注記

完全なアップグレードプロセスおよびこれらの Playbook を呼び出すタイミングについては、「最新の OpenShift Container Platform 3.7 リリースへのアップグレード」に説明されています。

openshift_upgrade_nodes_serial 変数は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、1 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%"
```

openshift_upgrade_nodes_label 変数を指定すると、特定のラベルを持つノードのみをアップグレードするように指定できます。これは **openshift_upgrade_nodes_serial** 変数と組み合わせて使用することもできます。

たとえば、**group1** リージョンのノードのみを1 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。

openshift_upgrade_nodes_max_fail_percentage 変数を使用すると、各バッチで失敗するノードの数を指定できます。失敗のパーセンテージは Playbook がアップグレードを中止する前に指定した値を上回っている必要があります。

openshift_upgrade_nodes_drain_timeout 変数を使用すると、終了前の待機時間の長さを指定できます。

以下の例では、1 度に 10 ノードがアップグレードし、20 パーセントを超えるノードが失敗する場合にアップグレードが中止し、ノードをドレイン (解放) するまでに 600 秒の待機時間があります。

```
$ 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
```

2.2.3. Ansible Hook を使用したアップグレードのカスタマイズ

OpenShift Container Platform のアップグレード時の特定の操作の実行中に、**Hook** というシステムでカスタムタスクを実行できます。Hook を使うと、管理者はアップグレード時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。これは、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、変更するのに非常に便利です。

Hook が失敗すると操作も失敗することに留意してください。つまり、Hook が適切であると複数回実行でき、同じ結果を出せます。適切な Hook にはべき等性があります。

2.2.3.1. 制限

- Hook 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。Hook 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、Hook は今後バージョン付けされる可能性があります。これは Hook 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となります。
- Hook にはエラー処理機能がなく、Hook 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。その場合、問題に対応してからアップグレードを再実行する必要があります。

2.2.3.2. Hook の使用

Hook 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 Hook 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 Hook 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 Hook 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
```

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"
```

2.2.3.3. Available アップグレード Hook

openshift_master_upgrade_pre_hook

- 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。

openshift_master_upgrade_hook

- 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

- 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。

2.3. 最新の OPENSHIFT CONTAINER PLATFORM 3.9 リリースへのアップグレード

既存の OpenShift Container Platform 3.7 または 3.9 クラスターを最新の 3.9 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. 「自動化アップグレードの準備」のステップを満たしていることを確認した上で、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. OpenShift Container Platform 3.9 より、アップグレード時に OpenShift Container Platform の Web コンソールがアップグレード時にマスターの Pod としてデプロイされ、**openshift_web_console_prefix** がカスタマイズされたイメージのプレフィックスを使用して Web コンソールをデプロイできるように導入されています。**template_service_broker_prefix** は他のコンポーネントに一致するように更新されています。インストールのカスタマイズされた **docker-registry** を **registry.access.redhat.com** の代わりに使用する場合、**openshift_web_console_prefix** および **template_service_broker_prefix** をアップグレード時に適切なイメージのプレフィックスをポイントするように明示的に指定する必要があります。

```
openshift_web_console_prefix=<registry_ip>:<port>/openshift3/ose-
template_service_broker_prefix=<registry_ip>:<port>/openshift3/ose-
```

4. ホストのローリングおよび完全なシステムの再起動を実行する必要がある場合は、インベントリーファイルの **openshift_rolling_restart_mode** パラメーターを **system** に設定できます。これを設定しないと、デフォルト値の **services** が HA マスターのローリングサービスの再起動を実行しますが、システムの再起動は実行しません。詳細については、「クラスター変数の設定」を参照してください。

5. この時点で、アップグレードを単一フェーズで実行するか、または複数フェーズで実行するかを選択できます。各フェーズでアップグレードされるコンポーネントについての詳細は、「コントロールプレーンとノードの別フェーズでのアップグレード」を参照してください。
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、必要に応じて **~/.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。

   - **オプション A:** 単一フェーズでコントロールプレーンおよびノードをアップグレードします。
   upgrade.yml Playbook を実行し、1 つの Playbook を使用して単一フェーズでクラスターのアップグレードを実行します。ここでも、まずコントロールプレーンをアップグレードしてからノードのインプレースアップグレードが実行されます。

     ```
     # ansible-playbook -i </path/to/inventory/file> \
         /usr/share/ansible/openshift-ansible/playbooks/byo/openshift-
     cluster/upgrades/v3_9/upgrade.yml
     ```

   - **オプション B:** 別々のフェーズでコントロールプレーンおよびノードをアップグレードします。

     a. コントロールプレーンのみを実行するには、**upgrade_control_plane.yaml** Playbook を実行します。

     ```
     # ansible-playbook -i </path/to/inventory/file> \
         /usr/share/ansible/openshift-
     ansible/playbooks/byo/openshift-
     cluster/upgrades/v3_9/upgrade_control_plane.yml
     ```

     b. ノードのみを実行するには、**upgrade_nodes.yaml** Playbook を実行します。

     ```
     # ansible-playbook -i </path/to/inventory/file> \
         /usr/share/ansible/openshift-
     ansible/playbooks/byo/openshift-
     cluster/upgrades/v3_9/upgrade_nodes.yml \
         [-e <customized_node_upgrade_variables>] 1
     ```

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

     「ノードの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、すべてのノードが正常にアップグレードされるまで **upgrade_nodes.yml** Playbook の起動を継続します。

6. すべてのマスターおよびノードのアップグレードの完了後に、すべてのホストを再起動します。再起動後に追加の機能が有効にされていない場合は、アップグレードの検証 を実行できます。追加機能が有効にされている場合には、次の手順は以前に有効にした追加機能の内容によって変わります。

機能	次の手順
集計されたロギング	EFK ロギングスタックのアップグレード。

機能	次の手順
クラスターメトリクス	クラスターメトリクスのアップグレード。

2.4. 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、提供されている **/usr/share/ansible/openshift-ansible/playbooks/openshift-logging/config.yml** Ansible Playbook を使用する必要があります。これは既存のクラスターでロギングを最初にデプロイする際に使用する Playbook ですが、既存のロギングデプロイメントをアップグレードする場合にも使用されます。

1. これをまだ実行していない場合は、「コンテナーログの集計」のトピックの「ロギング Ansible 変数の指定」を参照し、Ansible インベントリーファイルを更新して少なくとも **[OSEv3:vars]** セクション内に以下の必要な変数を設定します。

```
[OSEv3:vars]

openshift_logging_install_logging=true 1
openshift_logging_image_version=<tag> 2
```

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 **<tag>** を最新バージョンの **v3.9.31** に置き換えます。

2. 「ロギング Ansible 変数の指定」で説明されているように、デフォルトを上書きするために指定する必要のあるその他の **openshift_logging_*** 変数を追加します。

3. インベントリーファイルの更新が終了したら、「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、ロギングデプロイメントのアップグレードを完了します。

注記

EFK コンポーネントの Fluentd DeploymentConfig および DaemonSet が次のように設定されている場合、以下に注意してください。

```
        image: <image_name>:<vX.Y>
        imagePullPolicy: IfNotPresent
```

最新バージョンの **<image_name>** は、Pod が再デプロイされるノードに同じ **<image_name:vX.Y>** のイメージがローカルに保存されている場合にはプルされない可能性があります。その場合は、DeploymentConfig および DaemonSet を手動で **imagePullPolicy: Always** に設定し、再度プルされるようにします。

2.5. 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、提供されている **/usr/share/ansible/openshift-ansible/playbooks/openshift-metrics/config.yml** Ansible Playbook を使用する必要があります。これは、既存クラスターでメトリクスを最初にデプロイしている場合に使

用する Playbook ですが、既存のメトリクスデプロイメントをアップグレードする場合にも使用されます。

1. これをまだ実行していない場合は、「クラスターメトリクスの有効化」のトピックの「メトリクス Ansible 変数の指定」を参照し、Ansible インベントリーファイルを更新して少なくとも **[OSEv3:vars]** セクション内で以下の必要な変数を設定します。

```
[OSEv3:vars]

openshift_metrics_install_metrics=true (1)
openshift_metrics_image_version=<tag> (2)
openshift_metrics_hawkular_hostname=<fqdn> (3)
openshift_metrics_cassandra_storage_type=(emptydir|pv|dynamic) (4)
```

(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) **<tag>** を最新バージョンの **v3.9.31** に置き換えます。

(3) Hawkular Metrics ルートに使用されます。完全修飾ドメイン名に対応している必要があります。

(4) 以前のデプロイメントと一貫性のあるタイプを選択します。

2. 「メトリクス Ansible 変数の指定」で説明されているように、デフォルトを上書きするために指定する必要のあるその他の **openshift_metrics_*** 変数を追加します。

3. インベントリーファイルの更新が終了したら、「メトリクスコンポーネントのデプロイ」の説明に従って **openshift-metrics/config.yml** Playbook を実行し、メトリクスデプロイメントのアップグレードを完了します。

2.6. 混在環境についての特別な考慮事項

混在環境のアップグレード (Red Hat Enterprise Linux および Red Hat Enterprise Linux Atomic Host を含む環境など) では、**openshift_pkg_version** および **openshift_image_tag** の両方を設定する必要があります。混在環境では、**openshift_pkg_version** のみを指定する場合、その番号が Red Hat Enterprise Linux および Red Hat Enterprise Linux Atomic Host のイメージに使用されます。

2.7. コンテナー化された GLUSTERFS を使用する場合の特別な考慮事項

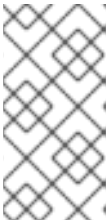
OpenShift Container Platform のアップグレード時に、GlusterFS Pod が実行されているノードのセットをアップグレードする必要があります。

drain および **unschedule** は daemonset の一部として実行されているために GlusterFS Pod を停止せず、退避しません。そのため、これらのノードをアップグレードする際には特別な考慮が必要になります。

また、複数のノードでアップグレードを同時に実行される可能性もあり、これにより複数のノードが GlusterFS Pod をホストしている場合にデータ可用性の問題が生じる可能性があります。

シリアルアップグレードが実行されている場合でも、次のノードの GlusterFS が終了する前に GlusterFS が自動修復操作のすべてを完了するのに必要な時間が確保される保証はありません。そのため、クラスターの状態が正常でなくなるか、または不明な状態になる可能性があります。したがって、以下の手順を実行することをお勧めします。

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ー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. **oc get daemonset** を実行して **NODE-SELECTOR** にあるラベルを検証します。デフォルト値は **storagenode=glusterfs** です。
   b. daemonset ラベルをノードから削除します。

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

   これにより、GlusterFS Pod がこのノード上で終了します。
   c. 追加のラベル (**type=upgrade** など) をアップグレードする必要のあるノードに追加します。
   d. アップグレード Playbook を GlusterFS を終了した単一ノードで実行するには、**-e openshift_upgrade_nodes_label="type=upgrade"** を使用します。
   e. アップグレードの完了時に、deamonset セレクターでノードのラベルを変更します。

```
$ oc label node <node_name> <daemonset_label>
```

   f. GlusterFS Pod が再生成され、表示されるまで待機します。
   g. **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ーを一覧表示します。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。

h. アップグレードラベル (**type=upgrade** など) を削除し、次の GlusterFS ノードに移動します。

2.8. 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 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
```

2.9. アップグレードの検証

以下を確認します。

- クラスターが正常である。
- サービス (マスター、ノードおよび etcd) 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する。

```
# oc get nodes
NAME                 STATUS    ROLES     AGE       VERSION
master.example.com   Ready     master    7h
v1.9.1+a0ce1bc657
node1.example.com    Ready     compute   7h
v1.9.1+a0ce1bc657
node2.example.com    Ready     compute   7h
v1.9.1+a0ce1bc657
```

2. 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)。**<tag>** を最新バージョンの **v3.9.31** に置き換えます。

```
# oc get -n default dc/docker-registry -o json | grep \"image\"
    "image": "openshift3/ose-docker-registry:<tag>",
# oc get -n default dc/router -o json | grep \"image\"
```

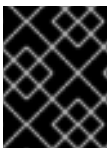
```
    "image": "openshift3/ose-haproxy-router:<tag>",
```

3. 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 クラスターの手動によるインプレースアップグレードの実行

3.1. 概要



重要

OpenShift Container Platform 3.9 の時点で、手動のアップグレードはサポートされていません。今後のリリースで、このプロセスは削除されます。

自動アップグレードに代わる方法として、OpenShift Container Platform クラスターを手動でアップグレードできます。中断なしの手動アップグレードを実行するには、このトピックで説明されているように各コンポーネントをアップグレードすることが重要になります。

アップグレードを開始する前に、手順全体に精通するようにしてください。リリースごとの追加の手動ステップを、標準アップグレードプロセス前やその実行中の主要なポイントで実行する必要がある場合があります。

重要

アップグレードに進む前に、すべての 前提条件を満たしていることを確認します。前提条件を満たしていないと、アップグレードが失敗する可能性があります。

GlusterFS を使用している場合は、「コンテナー化された GlusterFS を使用する場合の特別な考慮事項」を参照してからアップグレードを行ってください。

3.2. 手動アップグレードの準備

注記

クラスターの OpenShift Container Platform 3.9 へのアップグレードの前に、クラスターは バージョン 3.7 の最新の非同期リリースにアップグレードされている必要があります。クラスターのバージョンが 3.7 よりも前の場合、バージョンを徐々にアップグレードする必要があります (例: 3.5 から 3.6 にアップグレードしてから 3.6 から 3.7 にアップグレードする)。

手動アップグレードの準備をするには、以下の手順に従います。

1. 「アップグレードの検証」の手順に従い、クラスターの健全性を確認します (3.7 から 3.9 にアップグレードしている場合は、コマンド出力で指定の 3.9 ではなく、関連する 3.7 バージョンについて確認します)。これにより、ノードが **Ready** 状態にあり、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、診断エラーや警告がないことを確認できます。

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

```
# subscription-manager refresh
```

3. OpenShift Container Platform 3.7 から 3.9 にアップグレードしている場合は、以下を実行します。

   a. 3.7 リポジトリーを手動で無効にし、各マスターおよびノードホストで 3.8 および 3.9 リポジトリーの両方を有効にします。また、**rhel-7-server-ansible-2.4-rpms** リポジトリーを有効にする必要もあります。これは OpenShift Container Platform 3.9 以降の新規の要件で

す。

```
# subscription-manager repos --disable="rhel-7-server-ose-3.7-
rpms" \
    --enable="rhel-7-server-ose-3.8-rpms" \
    --enable="rhel-7-server-ose-3.9-rpms" \
    --enable="rhel-7-server-extras-rpms" \
    --enable="rhel-7-server-ansible-2.4-rpms" \
    --enable="rhel-7-fast-datapath-rpms"
# yum clean all
```

b. OpenShift Container Platform 3.9 にアップグレードする前に、swap メモリーをクラスターで無効にしている必要があります。そうしない場合、アップグレードは失敗します。swap メモリーが Ansible インベントリーファイルの **openshift_disable_swap=false** を使用して有効にされているか、またはホストごとに手動で有効にされているかどうかについては、『クラスター管理ガイド』の「swap メモリーの無効化」を確認し、各ホストで無効にしてください。

4. 各 RHEL 7 システムで利用可能な最新バージョンの **atomic-openshift-utils** パッケージをインストールするか、これに更新します。これは、後のセクションで使用されるファイルを提供します。

```
# yum install atomic-openshift-utils
```

5. 各 RHEL 7 システムで以下の利用可能な最新の ***-excluder** パッケージをインストールするか、またはこれに更新します。これにより、アップグレードを OpenShift Container Platform バージョンに応じて実行しようとしない場合に、システムのバージョンが適切なバージョンの **atomic-openshift** および **docker** パッケージであることを確認できます。

```
# yum install atomic-openshift-excluder atomic-openshift-docker-
excluder
```

これらのパッケージはエントリーをホストの **/etc/yum.conf** ファイルの **exclude** ディレクティブに追加します。

6. 各 etcd ホストで **etcd** バックアップを作成 します。

7. アップグレードパスについては、各 RHEL 7 システムで最新のカーネルを実行していることを確認します。

```
# yum update kernel
```

3.3. マスターコンポーネントのアップグレード

スタンドアロンノードをアップグレードする前に、(クラスターの **コントロールプレーン** を提供する) マスターコンポーネントをアップグレードします。

1. 各マスターで以下のコマンドを実行し、**atomic-openshift** パッケージをホストの yum の **exclude** 一覧から取り除きます。

```
# atomic-openshift-excluder unexclude
```

2. すべてのマスターホストおよび外部 **etcd** ホストで **etcd** をアップグレードします。

a. RPM ベースの方法を使用する RHEL 7 システムの場合:

a. RPM ベースの方法を使用する RHEL 7 システムの場合:

i. **etcd** パッケージをアップグレードします。

```
# yum update etcd
```

ii. **etcd** サービスを再起動し、ログでこれが正常に再起動したことを確認します。

```
# systemctl restart etcd
# journalctl -r -u etcd
```

b. コンテナー化方式を使用する RHEL Atomic Host 7 システムおよび RHEL 7 システムの場合:

i. 最新の **rhel7/etcd** イメージをプルします。

```
# docker pull registry.access.redhat.com/rhel7/etcd
```

ii. **etcd_container** サービスを再起動し、ログでこれが正常に再起動したことを確認します。

```
# systemctl restart etcd_container
# journalctl -r -u etcd_container
```

3. 各マスターホストで、**atomic-openshift** パッケージまたは関連イメージをアップグレードします。

a. RHEL 7 システムで RPM ベースの方法を使用するマスターの場合、インストールされたすべての **atomic-openshift** パッケージおよび **openvswitch** パッケージをアップグレードします。

```
# yum upgrade atomic-openshift\* openvswitch
```

b. RHEL 7 または RHEL Atomic Host 7 システムでコンテナー化方式を使用するマスターの場合、以下のファイルで **IMAGE_VERSION** パラメーターをアップグレードするバージョンに設定します。

- **/etc/sysconfig/atomic-openshift-master-controllers**
- **/etc/sysconfig/atomic-openshift-master-api**
- **/etc/sysconfig/atomic-openshift-node**
- **/etc/sysconfig/atomic-openshift-openvswitch**

例:

```
IMAGE_VERSION=<tag>
```

<tag> を最新バージョンの **v3.9.31** に置き換えます。

4. 以前のバージョンの OpenShift Container Platform では、マスターホストには、インストーラーによってデフォルトでスケジュール対象外 (Unschedulable) のマークが付けられました。これにより、Pod をこれらのホストに配置できませんでした。しかし、OpenShift Container Platform 3.9 より、Web コンソールはマスターサービスから削除され、マスターの Pod として

デプロイされます。つまり、マスターにはスケジュール対象 (Schedulable) のマークを付ける必要があります。これにより、**ノードロール (`master` または `compute`)** が割り当てるラベルを特定ホストに追加することも必要になります。

注記

この重要な技術上の変更のサポートされる自動アップグレードとの関連については、『OpenShift Container Platform 3.9 Release Notes』を参照してください。

- Masters Marked as Schedulable Nodes by Default (デフォルトでスケジュール対象ノードとマークされるマスター)
- Default Node Selector Set By Default and Automatic Node Labeling (デフォルトで設定されるデフォルトのノードセレクターおよび自動ノードラベリング)

OpenShift Container Platform 3.7 から 3.9 にアップグレードしている場合は、以下を実行します。

a. 以下のラベルを各マスターホストに追加します。

```
$ oc label node <hostname> node-role.kubernetes.io/master=true
```

これは、それらに **`master`** ノードロールを付与します。

b. 以下のラベルをマスター以外、インフラストラクチャー以外 (**`region=infra`** のラベル) の各ホストに追加します。

```
$ oc label node <hostname> node-role.kubernetes.io/compute=true
```

これは、それらに **`compute`** ノードロールを付与します。

c. Web コンソール Pod のみがマスターにスケジュールされるようにするには、デフォルトのノードセレクターも設定する必要があります。これを **/etc/origin/master/master-config.yaml** ファイルの **`projectConfig.defaultNodeSelector`** でカスタム値に設定している場合は、この手順を省略できます。まだ設定していない場合は、これを以下のように設定します。

```
projectConfig:
  defaultNodeSelector: node-role.kubernetes.io/compute=true
```

5. 各マスターでマスターサービスを再起動し、ログでそれが正常に再起動することを確認します。

```
# systemctl restart atomic-openshift-master-controllers
# systemctl restart atomic-openshift-master-api
# journalctl -r -u atomic-openshift-master-controllers
# journalctl -r -u atomic-openshift-master-api
```

6. マスターには OpenShift SDN の一部として設定されるようにノードコンポーネントが実行されているため、**atomic-openshift-node** および **openvswitch** サービスを再起動します。

```
# systemctl restart openvswitch
# systemctl restart atomic-openshift-node
```

```
# journalctl -r -u openvswitch
# journalctl -r -u atomic-openshift-node
```

7. OpenShift Container Platform 3.7 から 3.9 にアップグレードしている場合は、各マスターホストに対して以下を実行してそれらにスケジュール対象 (Schedulable) のマークを付けます。

```
$ oc adm manage-node <hostname> --schedulable=true
```

8. Docker をバージョン 1.13 への更新を必要とするクラスターアップグレードを実行している場合、Docker 1.13 に更新されていない場合は以下の手順を実行する必要もあります。

   a. **docker** パッケージをアップグレードします。

      i. RHEL 7 システムの場合:

```
# yum update docker
```

      次に **docker** サービスを再起動し、ログでこれが正常に再起動することを確認します。

```
# systemctl restart docker
# journalctl -r -u docker
```

      ii. RHEL Atomic Host 7 システムの場合、利用可能な場合は最新の Atomic ツリーにアップグレードします。

```
# atomic host upgrade
```

   b. アップグレードが完了し、次の起動の準備ができたら、ホストを再起動し、**docker** サービスが正常に起動することを確認します。

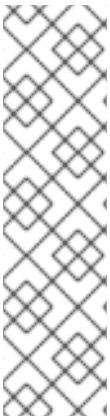
```
# systemctl reboot
# journalctl -r -u docker
```

   c. 必要なくなった以下のパッケージを削除します。

```
# rm /etc/systemd/system/docker.service.d/docker-sdn-ovs.conf
```

9. 各マスターで以下のコマンドを実行し、**atomic-openshift** パッケージをホストの yum の exclude 一覧に戻します。

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



注記

クラスターのアップグレード時に、マスターをローテーションから外すと良い場合あります。それは、一部の DNS クライアントライブラリーがクラスター DNS の他のマスターに適切に接続しないことがあるためです。マスターおよびコントローラーサービスを停止するほかに、EndPoint を Kubernetes サービスの **subsets.addresses** から削除することができます。

```
$ oc edit ep/kubernetes -n default
```

マスターが再起動すると、Kubernetes サービスが自動的に更新されます。

3.4. ポリシー定義の更新

クラスターのアップグレード後に、デフォルトロール デフォルトクラスターロールが自動的に更新されます。すべてのデフォルトがご使用の環境の推奨値に設定されているかどうかを確認するには、以下を実行します。

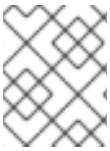
```
# oc adm policy reconcile-cluster-roles
```

警告

デフォルトのクラスターロールをカスタマイズしていて、ロールを調整してもそれらのカスタマイズされたロールが変更されないようにするには、Openshift ポリシーフォーマットの使用時にそれらのカスタマイズされたロールに **openshift.io/reconcile-protect** が **true** に設定されたアノテーションを付けることができます。新規の RBAC ロールの使用時は、**false** に設定された **rbac.authorization.kubernetes.io/autoupdate** を使用します。この場合、アップグレード時にそれらのロールを新規または必須のパーミッションで手動で更新する必要があります。

このコマンドは古いロールのとそれらの提案される新規の値の一覧を出力します。以下は例になります。

```
# oc adm policy reconcile-cluster-roles
apiVersion: v1
items:
- apiVersion: v1
  kind: ClusterRole
  metadata:
    creationTimestamp: null
    name: admin
  rules:
  - attributeRestrictions: null
    resources:
    - builds/custom
...
```



注記

出力は OpenShift バージョンおよびローカルのカスタマイズ内容によって異なります。提案されるポリシーに留意してください。

この出力をすでに行ったローカルポリシーの変更を再適用するように変更するか、または以下のプロセスを使用して新規ポリシーを自動的に適用することができます。

1. クラスターロールを調整します。

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

2. クラスターのロールバインディングを調整します。

```
# oc adm policy reconcile-cluster-role-bindings \
    --exclude-groups=system:authenticated \
    --exclude-groups=system:authenticated:oauth \
    --exclude-groups=system:unauthenticated \
    --exclude-users=system:anonymous \
    --additive-only=true \
    --confirm
```

さらに以下を実行します。

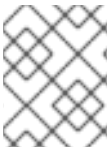
```
# oc adm policy reconcile-cluster-role-bindings \
    system:build-strategy-jenkinspipeline \
    --confirm \
    -o name
```

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

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

3.5. ノードのアップグレード

マスターのアップグレード後に、ノードをアップグレードすることができます。**atomic-openshift-node** サービスを再起動すると、サービスプロキシーが再起動している状態で、実行中の Pod からサービスへの発信ネットワーク接続の短い中断が発生します。この中断の期間は非常に短く、クラスター全体のサービス数に基づいて変動します。



注記

または、この時点で Blue-Green デプロイメント方式を使用して、インプレースアップグレードの代わりに新規ノードの並列環境を作成するともできます。

マスターではない各ノードについて一度に 1 つずつ、スケジューリングを無効にしてその Pod を他のノードに退避してから、パッケージをアップグレードし、サービスを再起動する必要があります。

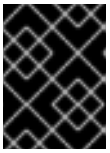
1. 各ノードで以下のコマンドを実行し、**atomic-openshift** パッケージをホストの yum の exclude 一覧から削除します。

```
# atomic-openshift-excluder unexclude
```

2. **cluster-admin** 権限を持つユーザーとして、ノードのスケジューリングを無効にします。

```
# oc adm manage-node <node> --schedulable=false
```

3. ノード上の Pod を他のノードに退避します。



重要

--force オプションは、レプリケーションコントローラーによってサポートされない Pod を削除します。

```
# oc adm drain <node> --force --delete-local-data --ignore-
daemonsets
```

4. ノードコンポーネントパッケージまたは関連イメージをアップグレードします。

   a. RHEL 7 システムで RPM ベースの方法を使用するノードの場合、インストールされたすべての **atomic-openshift** パッケージおよび **openvswitch** パッケージをアップグレードします。

   ```
   # yum upgrade atomic-openshift\* openvswitch
   ```

   b. RHEL 7 または RHEL Atomic Host 7 システムでコンテナー化方式を使用するノードの場合、**/etc/sysconfig/atomic-openshift-node** および **/etc/sysconfig/openvswitch** ファイルで **IMAGE_VERSION** パラメーターをアップグレードするバージョンに設定します。以下は例になります。

   ```
   IMAGE_VERSION=<tag>
   ```

   <tag> を最新バージョンの **v3.9.31** に置き換えます。

5. **atomic-openshift-node** および **openvswitch** サービスを再起動し、ログでこれが正常に再起動することを確認します。

```
# systemctl restart openvswitch
# systemctl restart atomic-openshift-node
# journalctl -r -u atomic-openshift-node
# journalctl -r -u openvswitch
```

6. Docker をバージョン 1.13 への更新を必要とするクラスターアップグレードを実行している場合、Docker 1.13 に更新されていない場合は以下の手順を実行する必要もあります。

   a. **docker** パッケージをアップグレードします。

      i. RHEL 7 システムの場合:

      ```
      # yum update docker
      ```

      次に **docker** サービスを再起動し、ログでこれが正常に再起動することを確認します。

      ```
      # systemctl restart docker
      # journalctl -r -u docker
      ```

      Docker を再起動した後に、**atomic-openshift-node** サービスを再び再起動して、ログでこれが正常に再起動することを確認します。

      ```
      # systemctl restart atomic-openshift-node
      # journalctl -r -u atomic-openshift-node
      ```

ii. RHEL Atomic Host 7 システムの場合、利用可能な場合は最新の Atomic ツリーにアップグレードします。

```
# atomic host upgrade
```

アップグレードが完了し、次の起動の準備ができたら、ホストを再起動し、**docker** サービスが正常に起動することを確認します。

```
# systemctl reboot
# journalctl -r -u docker
```

b. 必要なくなった以下のパッケージを削除します。

```
# rm /etc/systemd/system/docker.service.d/docker-sdn-ovs.conf
```

7. ノードのスケジューリングを再び有効にします。

```
# oc adm manage-node <node> --schedulable
```

8. 各ノードで以下のコマンドを実行し、**atomic-openshift** パッケージをホストの yum の exclude 一覧に戻します。

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

9. 次のノードで直前の手順を繰り返し、すべてのノードがアップグレードされるまでこれらの手順を繰り返します。

10. すべてのノードがアップグレードされた後に、**cluster-admin** 権限を持つユーザーがすべてのノードが **Ready** として表示されていることを確認します。

```
# oc get nodes
NAME                 STATUS    ROLES     AGE       VERSION
master.example.com   Ready     master    7h
v1.9.1+a0ce1bc657
node1.example.com    Ready     compute   7h
v1.9.1+a0ce1bc657
node2.example.com    Ready     compute   7h
v1.9.1+a0ce1bc657
```

3.6. ルーターのアップグレード

ルーターのデプロイを事前に実行している場合、ルーターのデプロイメント設定は、ルーターイメージに含まれている更新を適用できるようにアップグレードされている必要があります。サービスを中断することなくルーターをアップグレードするには、高可用ルーティングサービスを事前にデプロイしている必要があります。

ルーターのデプロイメント設定を編集します。たとえば、デフォルトの **ルーター** 名がある場合は、以下を実行します。

```
# oc edit dc/router
```

以下の変更を適用します。

-

```
...
spec:
 template:
    spec:
      containers:
      - env:
        ...
        image: registry.access.redhat.com/openshift3/ose-haproxy-router:
<tag> (1)
        imagePullPolicy: IfNotPresent
        ...
```

(1) **<tag>** をアップグレードしているバージョンに一致するように調整します (最新バージョンの場合は **v3.9.31** を使用します)。

1 つのルーター Pod が更新され、次の更新が実行されるのを確認できます。

3.7. レジストリーのアップグレード

変更をレジストリーイメージで有効にするには、レジストリーも更新する必要があります。**PersistentVolumeClaim** またはホストマウントポイントを使用している場合には、レジストリーの内容を失うことなくレジストリーを再起動できます。レジストリーの永続ストレージを設定する方法については、「レジストリーのストレージ」を参照してください。

レジストリーのデプロイメント設定を編集します。

```
# oc edit dc/docker-registry
```

以下の変更を適用します。

```
...
spec:
 template:
    spec:
      containers:
      - env:
        ...
        image: registry.access.redhat.com/openshift3/ose-docker-registry:
<tag> (1)
        imagePullPolicy: IfNotPresent
        ...
```

(1) **<tag>** をアップグレードしているバージョンに一致するように調整します (最新バージョンの場合は **v3.9.31** を使用します)。

レジストリーコンソールがデプロイされている場合、そのデプロイメント設定を編集します。

```
# oc edit dc/registry-console
```

以下の変更を適用します。

```
...
```

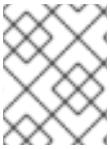
```
spec:
 template:
    spec:
      containers:
      - env:
        ...
        image: registry.access.redhat.com/openshift3/registry-console:v3.9
        imagePullPolicy: IfNotPresent
        ...
```

重要

アップグレード時に内部レジストリーにプッシュされているか、またはそこからプルされているイメージは失敗し、自動的に再起動される必要があります。これによってすでに実行されている Pod は中断されません。

3.8. デフォルトのイメージストリームおよびテンプレートの更新

デフォルトでは、通常インストール (advanced installation) 方式は、すべてのユーザーが閲覧アクセスを持つデフォルトプロジェクトである **openshift** プロジェクトにデフォルトのイメージストリーム、InstantApp テンプレート、およびデータベースサービステンプレートを作成します。これらのオブジェクトは、**/usr/share/ansible/openshift-ansible/roles/openshift_examples/files/examples/** ディレクトリーにある JSON ファイルからインストール時に作成されます。



注記

RHEL Atomic Host 7 は **yum** を使用してパッケージを更新できないので、以下の手順が RHEL 7 システムで実行される必要があります。

サンプルの JSON ファイルを提供するパッケージを更新します。CLI を **cluster-admin** パーミッションを持つユーザーとして実行できるサブスクライブ済みの Red Hat Enterprise Linux 7 システムで、最新バージョンの **atomic-openshift-utils** パッケージのインストールまたは更新を実行します。これにより、**openshift-ansible-** パッケージも更新されるはずです。

```
# yum update atomic-openshift-utils
```

最初のマスターで **/usr/share/openshift/examples/** を永続化するには、以下を実行します。

```
$ scp -R /usr/share/ansible/openshift-
ansible/roles/openshift_examples/files/examples/v3.9/*
user@master1:/usr/share/openshift/examples/
```

すべてのマスターで **/usr/share/openshift/examples/** を永続化するには、以下を実行します。

```
$ mkdir /usr/share/openshift/examples
$ scp -R /usr/share/ansible/openshift-
ansible/roles/openshift_examples/files/examples/v3.9/*
user@masterx:/usr/share/openshift/examples
```

openshift-ansible-roles パッケージは最新のサンプル JSON ファイルを提供します。

1. 手動のアップグレード後に、**openshift-ansible-roles** から最新テンプレートを取得します。

```
# rpm -ql openshift-ansible-roles | grep examples | grep v3.9
```

この例では、**/usr/share/ansible/openshift-ansible/roles/openshift_examples/files/examples/v3.9/image-streams/image-streams-rhel7.json** が最新の **openshift-ansible-roles** パッケージの必要な最新ファイルになります。

/usr/share/openshift/examples/image-streams/image-streams-rhel7.json はパッケージで所有されていませんが、Ansible で更新されます。Ansible 外でアップグレードしている場合、**oc** を実行しているシステムで最新の **.json** ファイルを取得する必要があります。このコマンドはマスターにアクセスできる任意の場所で実行できます。

2. **atomic-openshift-utils** およびその依存関係をインストールして、最新のコンテンツを **/usr/share/ansible/openshift-ansible/roles/openshift_examples/files/examples/v3.9/** にインストールします。

```
$ oc create -n openshift -f  /usr/share/ansible/openshift-
ansible/roles/openshift_examples/files/examples/v3.9/image-
streams/image-streams-rhel7.json
$ oc create -n openshift -f  /usr/share/ansible/openshift-
ansible/roles/openshift_examples/files/examples/v3.9/image-
streams/dotnet_imagestreams.json
$ oc replace -n openshift -f  /usr/share/ansible/openshift-
ansible/roles/openshift_examples/files/examples/v3.9/image-
streams/image-streams-rhel7.json
$ oc replace -n openshift -f  /usr/share/ansible/openshift-
ansible/roles/openshift_examples/files/examples/v3.9/image-
streams/dotnet_imagestreams.json
```

3. テンプレートを更新します。

```
$ oc create -n openshift -f /usr/share/ansible/openshift-
ansible/roles/openshift_examples/files/examples/v3.9/quickstart-
templates/
$ oc create -n openshift -f /usr/share/ansible/openshift-
ansible/roles/openshift_examples/files/examples/v3.9/db-templates/
$ oc create -n openshift -f /usr/share/ansible/openshift-
ansible/roles/openshift_examples/files/examples/v3.9/infrastructure-
templates/
$ oc create -n openshift -f /usr/share/ansible/openshift-
ansible/roles/openshift_examples/files/examples/v3.9/xpaas-
templates/
$ oc create -n openshift -f /usr/share/ansible/openshift-
ansible/roles/openshift_examples/files/examples/v3.9/xpaas-streams/
$ oc replace -n openshift -f /usr/share/ansible/openshift-
ansible/roles/openshift_examples/files/examples/v3.9/quickstart-
templates/
$ oc replace -n openshift -f /usr/share/ansible/openshift-
ansible/roles/openshift_examples/files/examples/v3.9/db-templates/
$ oc replace -n openshift -f /usr/share/ansible/openshift-
ansible/roles/openshift_examples/files/examples/v3.9/infrastructure-
templates/
$ oc replace -n openshift -f /usr/share/ansible/openshift-
ansible/roles/openshift_examples/files/examples/v3.9/xpaas-
```

```
templates/
$ oc replace -n openshift -f /usr/share/ansible/openshift-
ansible/roles/openshift_examples/files/examples/v3.9/xpaas-streams/
```

すでに存在する項目についてエラーが生成されます。これは予期される動作です。

```
# oc create -n openshift -f /usr/share/ansible/openshift-
ansible/roles/openshift_examples/files/examples/v3.9/quickstart-
templates/
Error from server: error when creating
"/usr/share/ansible/openshift-
ansible/roles/openshift_examples/files/examples/v3.9/quickstart-
templates/cakephp-mysql.json": templates "cakephp-mysql-example"
already exists
Error from server: error when creating
"/usr/share/ansible/openshift-
ansible/roles/openshift_examples/files/examples/v3.9/quickstart-
templates/cakephp.json": templates "cakephp-example" already exists
Error from server: error when creating
"/usr/share/ansible/openshift-
ansible/roles/openshift_examples/files/examples/v3.9/quickstart-
templates/dancer-mysql.json": templates "dancer-mysql-example"
already exists
Error from server: error when creating
"/usr/share/ansible/openshift-
ansible/roles/openshift_examples/files/examples/v3.9/quickstart-
templates/dancer.json": templates "dancer-example" already exists
Error from server: error when creating
"/usr/share/ansible/openshift-
ansible/roles/openshift_examples/files/examples/v3.9/quickstart-
templates/django-postgresql.json": templates "django-psql-example"
already exists
```

ここで、コンテンツを更新できます。自動化されたアップグレード Playbook を実行しないと、コンテンツは **/usr/share/openshift/** で更新されません。

3.9. 最新イメージのインポート

デフォルトのイメージストリームの更新の実行後に、それらのストリーム内のイメージも更新されていることを確認する必要がある場合があります。デフォルトの **openshift** プロジェクトの各イメージストリームについて、以下を実行できます。

```
# oc import-image -n openshift <imagestream>
```

たとえば、デフォルトの **openshift** プロジェクトのすべてのイメージストリームの一覧を取得します。

```
# oc get is -n openshift
NAME     DOCKER REPO
TAGS                    UPDATED
mongodb  registry.access.redhat.com/openshift3/mongodb-24-rhel7
2.4,latest,v3.1.1.6    16 hours ago
mysql    registry.access.redhat.com/openshift3/mysql-55-rhel7
5.5,latest,v3.1.1.6    16 hours ago
```

```
nodejs   registry.access.redhat.com/openshift3/nodejs-010-rhel7
0.10,latest,v3.1.1.6   16 hours ago
...
```

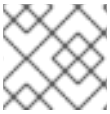
各イメージストリームを一度に1つずつ更新します。

```
# oc import-image -n openshift nodejs
The import completed successfully.

Name:   nodejs
Created:  10 seconds ago
Labels:   <none>
Annotations:  openshift.io/image.dockerRepositoryCheck=2016-07-
05T19:20:30Z
Docker Pull Spec: 172.30.204.22:5000/openshift/nodejs

Tag Spec        Created  PullSpec      Image
latest 4        9 seconds ago registry.access.redhat.com/rhscl/nodejs-4-
rhel7:latest
570ad8ed927fd5c2c9554ef4d9534cef808dfa05df31ec491c0969c3bd372b05
4 registry.access.redhat.com/rhscl/nodejs-4-rhel7:latest  9 seconds ago
<same>
570ad8ed927fd5c2c9554ef4d9534cef808dfa05df31ec491c0969c3bd372b05
0.10 registry.access.redhat.com/openshift3/nodejs-010-rhel7:latest 9
seconds ago <same>
a1ef33be788a28ec2bdd48a9a5d174ebcfbe11c8e986d2996b77f5bccaaa4774
```

重要

S2I ベースのアプリケーションを更新するために、**oc start-build <app-name>** を使用して新規イメージをインポートした後にこれらのアプリケーションの新規ビルドを手動でトリガーする必要があります。

3.10. WEB コンソールのアップグレード



注記

Web コンソールの手動によるアップグレード手順は利用できません。

OpenShift Container Platform 3.9 より、Web コンソールはマスターの一部ではなく、別個のコンポーネントとしてデプロイされます。このコンソールは、**openshift_web_console_install** 変数を **false** に設定しない限り、自動アップグレードの一環としてデフォルトでインストールされます。

または、手動によるアップグレードの一環としてなど、必要に応じて Web コンソールを別個にインストールすることもできます。

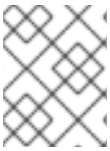
1. 「通常インストール (Advanced Installation)」のトピックのセクションを参照し、インベントリーファイルを随時更新します。
2. 以下の Playbook を実行します。

```
# ansible-playbook -i </path/to/inventory/file> \
    /usr/share/ansible/openshift-ansible/playbooks/openshift-web-
console/config.yml
```

注意

Web コンソールが拡張を読み込む方法は OpenShift Container Platform 3.9 で変更されました。以前のバージョンからアップグレードしている場合、マスターファイルシステムではなく URL で拡張スクリプトおよびスタイルシートをホストする必要があります。詳細は、「拡張スクリプトとスタイルシートの読み込み」のトピックを参照してください。

3.11. サービスカタログのアップグレード



注記

サービスカタログおよびサービスブローカーの手動によるアップグレード手順は利用できません。

サービスカタログをアップグレードするには、以下を実行します。

1. 「通常インストール (Advanced Installation)」のトピックで以下の 3 つのセクションを参照し、インベントリーファイルを随時更新します。
   - サービスカタログの設定
   - OpenShift Ansible Broker の設定
   - テンプレートサービスブローカーの設定
2. 以下の Playbook を実行します。

```
# ansible-playbook -i </path/to/inventory/file> \
    /usr/share/ansible/openshift-ansible/playbooks/openshift-
service-catalog/config.yml
```

3.12. EFK ロギングスタックのアップグレード

注記

ロギングデプロイメントの手動によるアップグレード手順は、OpenShift Container Platform 3.5 以降では利用できません。

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、提供されている **/usr/share/ansible/openshift-ansible/playbooks/openshift-logging/config.yml** Ansible Playbook を使用する必要があります。これは既存のクラスターでロギングを最初にデプロイする際に使用する Playbook ですが、既存のロギングデプロイメントをアップグレードする場合にも使用されます。

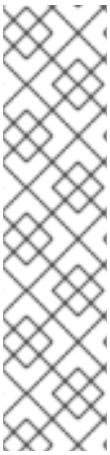
1. これをまだ実行していない場合は、「コンテナーログの集計」のトピックの「ロギング Ansible 変数の指定」を参照し、Ansible インベントリーファイルを更新して少なくとも **[OSEv3:vars]** セクション内に以下の必要な変数を設定します。

```
[OSEv3:vars]

openshift_logging_install_logging=true 1
openshift_logging_image_version=<tag> 2
```

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. **<tag>** を最新バージョンの **v3.9.31** に置き換えます。

2. 「ロギング Ansible 変数の指定」で説明されているように、デフォルトを上書きするために指定する必要のあるその他の **openshift_logging_*** 変数を追加します。

3. インベントリーファイルの更新が終了したら、「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、ロギングデプロイメントのアップグレードを完了します。



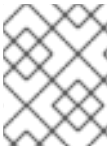
注記

EFK コンポーネントの Fluentd DeploymentConfig および DaemonSet が次のように設定されている場合、以下に注意してください。

```
        image: <image_name>:<vX.Y>
        imagePullPolicy: IfNotPresent
```

最新バージョンの **<image_name>** は、Pod が再デプロイされるノードに同じ **<image_name:vX.Y>** のイメージがローカルに保存されている場合にはプルされない可能性があります。その場合は、DeploymentConfig および DaemonSet を手動で **imagePullPolicy: Always** に設定し、再度プルされるようにします。

3.13. クラスターメトリクスのアップグレード



注記

メトリクスデプロイメントの手動によるアップグレード手順は、OpenShift Container Platform 3.5 以降では利用できません。

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、提供されている **/usr/share/ansible/openshift-ansible/playbooks/openshift-metrics/config.yml** Ansible Playbook を使用する必要があります。これは、既存クラスターでメトリクスを最初にデプロイしている場合に使用する Playbook ですが、既存のメトリクスデプロイメントをアップグレードする場合にも使用されます。

1. これをまだ実行していない場合は、「クラスターメトリクスの有効化」のトピックの「メトリクス Ansible 変数の指定」を参照し、Ansible インベントリーファイルを更新して少なくとも **[OSEv3:vars]** セクション内で以下の必要な変数を設定します。

```
[OSEv3:vars]

openshift_metrics_install_metrics=true 1
openshift_metrics_image_version=<tag> 2
openshift_metrics_hawkular_hostname=<fqdn> 3
openshift_metrics_cassandra_storage_type=(emptydir|pv|dynamic) 4
```

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. **<tag>** を最新バージョンの **v3.9.31** に置き換えます。

3 Hawkular Metrics ルートに使用されます。完全修飾ドメイン名に対応している必要があります。

4 以前のデプロイメントと一貫性のあるタイプを選択します。

2. 「メトリクス Ansible 変数の指定」で説明されているように、デフォルトを上書きするために指定する必要のあるその他の **openshift_metrics_*** 変数を追加します。

3. インベントリーファイルの更新が終了したら、「メトリクスコンポーネントのデプロイ」の説明に従って **openshift-metrics/config.yml** Playbook を実行し、メトリクスデプロイメントのアップグレードを完了します。

3.14. リリースごとの追加の手動ステップ

一部の OpenShift Container Platform リリースには、クラスター全体に更新を完全に適用するために実行する必要のあるリリース固有の追加の指示が含まれている場合があります。このセクションは、OpenShift Container Platform 3.9 の新規の非同期更新がリリースされるたびに更新されます。

最新のリリースノートを参照するには、『OpenShift Container Platform 3.9 Release Notes』を参照してください。

3.15. アップグレードの検証

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する。

```
# oc get nodes
NAME                     STATUS    ROLES     AGE       VERSION
master.example.com       Ready     master    7h
v1.9.1+a0ce1bc657
node1.example.com        Ready     compute   7h
v1.9.1+a0ce1bc657
node2.example.com        Ready     compute   7h
v1.9.1+a0ce1bc657
```

2. 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)。**<tag>** を最新バージョンの **v3.9.31** に置き換えます。

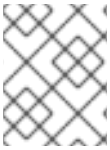
```
# oc get -n default dc/docker-registry -o json | grep \"image\"
    "image": "openshift3/ose-docker-registry:<tag>",
# oc get -n default dc/router -o json | grep \"image\"
    "image": "openshift3/ose-haproxy-router:<tag>",
```

3. 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4章 BLUE-GREEN デプロイメント

4.1. 概要



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。

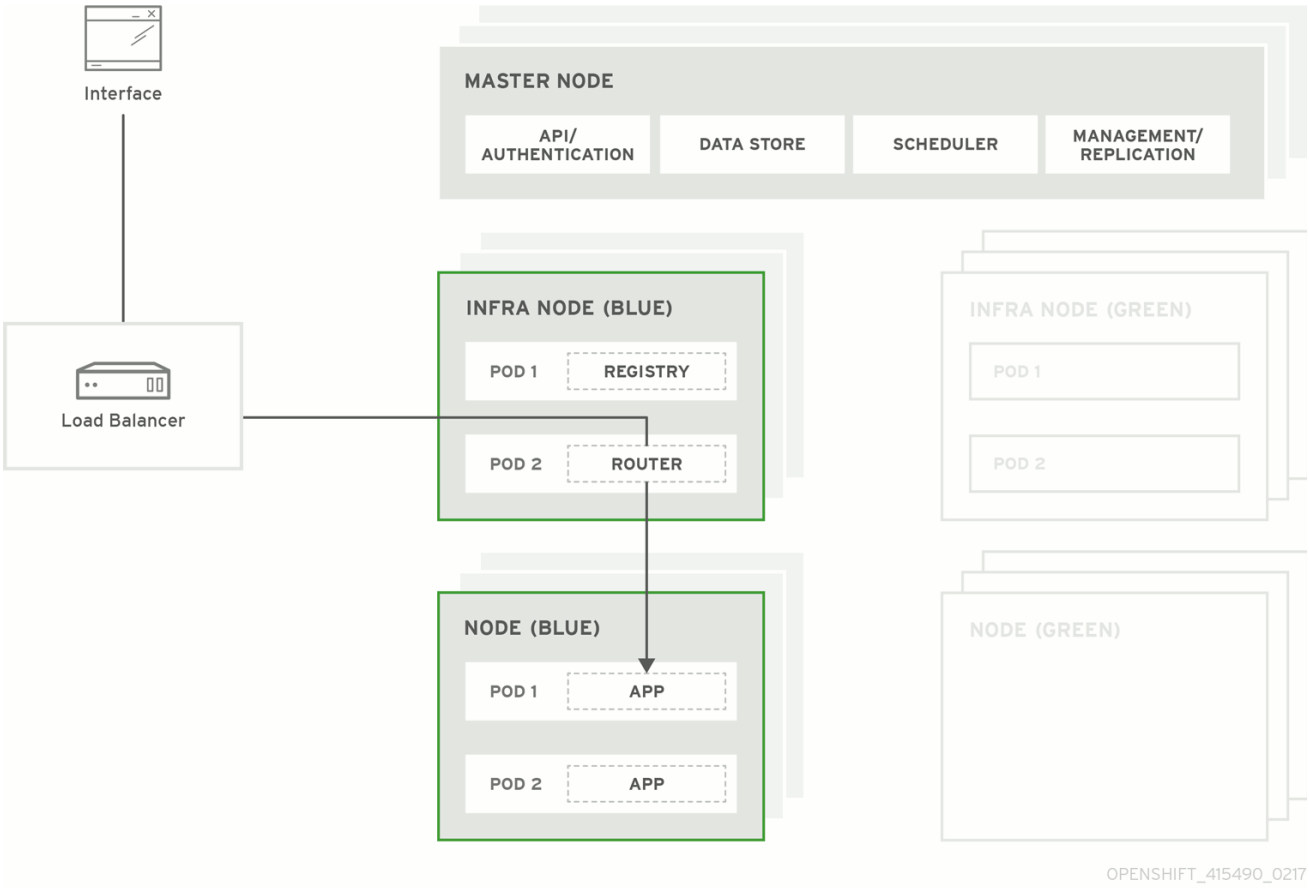
この方式では、管理者は新規デプロイメントの検証後、古いノードホストセット (例:「**Blue**」デプロイメント) から新規セット (例:「**Green** デプロイメント) にトラフィックを切り換えることができます。問題が検出される場合も、古いデプロイメントへのロールバックを簡単かつすぐに実行できます。

Blue-Green がソフトウェアについてのデプロイの証明済みの有効な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している訳でもなく、これに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た候補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4.1 Blue-Green デプロイメント



4.2. 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に、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。

4.2.1. 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 を使用して登録します。

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

重要

正常なデプロイメントの後に、**subscription-manager clean** で古いホストを登録解除し、環境が非コンプライアンスの状態にならないようにします。

4.2.2. Blue ノードのラベリング

実稼働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. すべてのホストに適切なノードラベルがあることを確認します。すべてのマスターはスケジュール対象 (Schedulable) ノードホストとして設定される必要があります (それらが Pod ネットワークに加わり、Web コンソール Pod を実行できるようにするために必要です)。クラスター管理を強化するには、各ホストにタイプ (**type=master** または **type=node** など) を記述するラベルを追加します。
たとえば、マスターでもあるノードホストに **type=master** のラベルを付けるには、それぞれの関連する **<node_name>** について以下を実行します。

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

マスター以外のノードホストに **type=node** のラベルを付けるには、それぞれの関連する **<node_name>** について以下を実行します。

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

または、**type=master** を使用して特定ノードにラベル付けしており、残りのノードすべてに **type=node** のラベルを付ける必要がある場合、**--all** オプションを使用しても、**type=** がすでに設定されたすべてのホストは上書きされません。

```
$ oc label node --all type=node
```

3. 現行の実稼働環境のマスター以外のノードホストに **color=blue** のラベルを付けます。たとえば、直前の手順で記述されたラベルを使用します。

```
$ oc label node -l type=node color=blue
```

上記のコマンドでは、**-l** フラグはセレクター **type=node** を使用して環境のサブセットに一致するように設定され、すべての一致項目には **color=blue** のラベルが付けられます。

4.2.3. Green ノードの作成およびラベリング

置き換えられるノードホストについて、これと同じ数の新規ノードホストを既存クラスターに追加することで新規の Green 環境を作成します。「ホストの既存クラスターへの追加」で説明されている通常インストール (advanced install) 方式を使用できます。

これらの新規ノードの追加時に、以下の Ansible 変数を使用します。

- 各ノードホストに **openshift_node_labels** 変数を設定して、これらのホストのインストール時に **color=green** ラベルを自動的に適用します。また、**oc label node** コマンドを使用して、必要な場合にインストール後にラベルをいつでも調整することができます。
- 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)、各ホストノードに **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。

new_nodes ホストグループの例

以下を既存のインベントリーに追加します。事前にインベントリーに置かれているすべてのものはそのままの状態にする必要があります。

```
[new_nodes]
node4.example.com openshift_node_labels="{'region': 'primary',
'color':'green'}" openshift_schedulable=false
node5.example.com openshift_node_labels="{'region': 'primary',
'color':'green'}" openshift_schedulable=false
node6.example.com openshift_node_labels="{'region': 'primary',
'color':'green'}" openshift_schedulable=false
infra-node3.example.com openshift_node_labels="{'region': 'infra',
'color':'green'}" openshift_schedulable=false
infra-node4.example.com openshift_node_labels="{'region': 'infra',
'color':'green'}" openshift_schedulable=false
```

4.2.4. 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。以下のチェックリストを実行します。

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

```
$ oc get nodes

ip-172-31-49-10.ec2.internal    Ready                         3d
```

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

ip-172-31-49-10.ec2.internal    Ready                         4d
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/instance-
type=m4.large,beta.kubernetes.io/os=linux,color=green,failure-
domain.beta.kubernetes.io/region=us-east-1,failure-
domain.beta.kubernetes.io/zone=us-east-1c,hostname=openshift-
cluster-1d005,kubernetes.io/hostname=ip-172-31-49-
10.ec2.internal,region=us-east-1,type=infra
```

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/internal-
api-upgradetest-openshift-com:443/system:admin'
[Note] Performing systemd discovery

[Note] Running diagnostic: ConfigContexts[default/api-upgradetest-
openshift-com:443/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.
...
```

4.3. レジストリーおよびルーターのカナリアデプロイメント

一般的に、レジストリーおよびルーター Pod はそれらが新規 (Green) インフラストラクチャーノードホストに移行するまでスケーリングされます。これらの Pod については、カナリア デプロイメント方法が一般的に使用されます。

これらの Pod のスケーリングによって、それらの Pod は新規インフラストラクチャーノードですぐにアクティブになります。それらのデプロイメント設定を新規イメージにポイントすると、ローリングアップデートが起動します。ただし、ノードの非アフィニティーおよび Blue ノードがスケジュール対象外 (Unschedulable) のままであることにより、ユーザーアプリケーションの古いノードへのデプロイメントは失敗します。

この時点で、レジストリーおよびルーターのデプロイメントは Pod の元の数に縮小される可能性があります。いつの時点でも、元の数の Pod は依然として利用可能なままであるため、容量を失うことなく、ダウンタイムを防ぐことができます。

4.4. GREEN ノードの準備

Pod を Blue 環境から Green に移行するために、必要なコンテナーイメージをプルする必要があります。レジストリーについてのネットワークの待機時間およびロードにより、環境に十分な容量が組み込まれていない場合は遅延が生じる可能性があります。

多くの場合、実行中のシステムへの影響を最小限に抑えるための最良の方法として、新規ノードに到達する新規 Pod デプロイメントをトリガーすることができます。これは、新規イメージストリームをインポートして実行できます。

メジャーリリースの OpenShift Container Platform (および一部の非同期エラータ更新) では、Source-to-Image (S2I) のユーザーのビルダーイメージについて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 **準備完了** (予想される負荷の増加に対応できる状態にある) とみなされると、他のすべてのものが後の手順のノードの退避で移行するため、結果として処理がより迅速になります。

アップグレードプロセスに進む準備ができたら、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のみ到達するようにします。

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

2. Blue ノードを無効にし、それらをスケジュール対象外 (Unschedulable) に設定して新規 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. 「手動によるインプレースアップグレード」で説明されているようにデフォルトのイメージストリームおよびテンプレートを更新します。

4. 「手動によるインプレースアップグレード」で説明されているように最新イメージをインポートします。
このプロセスでは多数のビルドをトリガーする可能性があることに留意してください。ただし、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。

5. 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。

4.5. 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ことは重要です。これについての詳細はクラウドプロバイダーごとに異なる場合があるためです。

OpenShift Container Platform 3.2 以降では、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ら、以下の手順に従います。

1. 以下のオプションのいずれかに従ってすべての Blue ノードを退避し、削除します。

   a. **オプション A:** 以下のコマンドを使ってすべての**color=blue** ノードを手動で退避してからこれらを削除します。

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

   b. **オプション B: `delete`** コマンドを実行する前にマスターをフィルターにかけます。

      i. 以下のコマンドを実行して削除する Blue ノードホストの一覧を確認します。このコマンドの出力には、**color=blue** ラベルを持つが、**type=master** ラベルを持たないすべてのノードホストの一覧が含まれます。クラスター内のすべてのホストには **color** および **type** ラベルの両方が割り当てられている必要があります。ノードの一覧をさらに制限する必要がある場合は、追加のフィルターを適用するようにコマンドを変更できます。

      ```
      $ oc get nodes -o go-template='{{ range .items }}{{ if (eq
      .metadata.labels.color "blue") and (ne .metadata.labels.type
      "master") }}{{ .metadata.name }}{{ "\n" }}{{end}}{{ end }}'
      ```

      ii. 削除する Blue ノードの一覧を確認したら、このコマンドを実行してノードの一覧を削除します。

      ```
      $ for i in $(oc get nodes -o \
          go-template='{{ range .items }}{{ if (eq
      .metadata.labels.color "blue") and (ne .metadata.labels.type
      "master") }}{{ .metadata.name }}{{ "\n" }}{{end}}{{ end }}');
      \
      do
          oc delete node $i
      done
      ```

2. Blue ノードホストに Pod が含まれなくなり、それらが OpenShift Container Platform から削除されたら、それらの電源をオフにしても問題がありません。安全上の措置として、アップグレードに問題がある場合には、ホストを短時間そのままにしておくと良いでしょう。

3. これらのホストを停止する前に、すべての必要なスクリプトまたはファイルが取得されていることを確認します。指定した期間と容量に問題がないことを確認した後に、これらのホストを削除します。

第5章 オペレーティングシステムの更新

5.1. 目的

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。

5.2. ホストでのオペレーティングシステムの更新

以下を使用してホストで OS を安全にアップグレードします。

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. ホストパッケージを更新し、ホストを再起動します。再起動により、ホストが最新バージョンを実行していることを確認できます。つまり、**docker** および OpenShift Container Platform プロセスが再起動されていることになり、これにより他のサービスのすべてのサービスが正しいことを確認するチェックが強制的に実行されます。
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はありませんが、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。

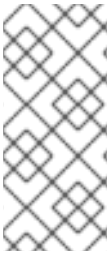
3. ホストを再びスケジュール対象 (Schedulable) に設定するには、以下を実行します。

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

第6章 OPENSHIFT のダウングレード

6.1. 概要

OpenShift Container Platform の アップグレード後に、極端なケースではあるものの、クラスターを直前の状態にダウングレードする必要がある場合があります。以下のセクションでは、OpenShift Container Platform 3.9 から 3.7 へのダウングレードパスなどの、クラスターの各システムでダウングレードを実行するために必要な手順について説明します。



注記

3.9 から 3.7 には直接ダウングレードできますが、etcd バックアップから復元している必要があります。

アップグレードが 3.8 の手順で失敗している場合、同じダウングレード手順が適用されます。

警告

現時点で、これらの手順は OpenShift Container Platform の RPM ベースのインストールでのみサポートされており、クラスター全体のダウンタイムが生じることが前提とされています。

6.2. バックアップの確認

アップグレードプロセスで使用される Ansible Playbook は **master-config.yaml** ファイルと etcd データディレクトリーのバックアップを作成している必要があります。これらがマスターと etcd メンバーに存在していることを確認します。

```
/etc/origin/master/master-config.yaml.<timestamp>
/var/lib/etcd/openshift-backup-pre-upgrade-<timestamp>
/etc/origin/master/scheduler.json.<timestamp>
```

さらに、(ノードコンポーネントを持つマスターを含む) 各ノードでタイムスタンプの付いた **node-config.yaml** ファイルをバックアップします。

```
/etc/origin/node/node-config.yaml.<timestamp>
```

(単一の組み込み etcd ではなく) 外部 etcd クラスターを使用している場合、リカバリー用には 1 つのバックアップのみが必要となりますが、バックアップはすべての etcd メンバーで作成される可能性があります。

以下のファイルの **.rpmsave** バックアップのコピーを保持します。

```
/etc/sysconfig/atomic-openshift-master-api
/etc/sysconfig/atomic-openshift-master-controller
/etc/etcd/etcd.conf
```

注記

アップグレードの実行日に作成されたアップグレード前の最初のバックアップから復元します。

6.3. クラスターのシャットダウン

すべてのマスター、ノードおよび **etcd** メンバー (外部 **etcd** クラスターを使用している場合) で、関連するサービスが停止していることを確認します。

```
# systemctl stop atomic-openshift-master-api atomic-openshift-master-
controllers
```

すべてのマスターおよびノードホストで、以下を実行します。

```
# systemctl stop atomic-openshift-node
```

外部 **etcd** ホストで、以下を実行します。

```
# systemctl stop etcd
```

6.4. RPM の削除

***-excluder** パッケージは、インストール時に、エントリーをホストの**/etc/yum.conf** ファイルの **exclude** ディレクティブに追加します。

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-clients \
    atomic-openshift-node \
    atomic-openshift-master-api \
    atomic-openshift-master-controllers \
    openvswitch \
    atomic-openshift-sdn-ovs \
    tuned-profiles-atomic-openshift-node\
    atomic-openshift-excluder\
    atomic-openshift-docker-excluder
```

2. 外部 **etcd** を使用している場合は、**etcd** パッケージも削除します。

```
# yum remove etcd
```

6.5. DOCKER のダウングレード

OpenShift Container Platform 3.9 には Docker 1.13 が必要で、OpenShift Container Platform 3.7 には Docker 1.12 が必要です。

Docker 1.13 を実行するすべてのホストで、以下の手順で Docker 1.12 にダウングレードします。

1. ホストのすべてのローカルコンテナーおよびイメージを削除します。レプリケーションコントローラーでサポートされるすべての Pod が再作成されます。

警告

以下のコマンドは破壊的動作をするため、注意して使用する必要があります。

すべてのコンテナーを削除します。

```
# docker rm $(docker ps -a -q) -f
```

すべてのイメージを削除します。

```
# docker rmi $(docker images -q)
```

2. (**yum downgrade** の代わりに) **yum swap** を使用して Docker 1.12.6 をインストールします。

```
# yum swap docker-* docker-*1.12.6 -y
# sed -i 's/--storage-opt dm.use_deferred_deletion=true//'
/etc/sysconfig/docker-storage
# systemctl restart docker
```

3. Docker 1.12.6 がインストールされており、ホスト上で実行されているはずです。以下で確認します。

```
# docker version
# systemctl status docker
```

6.6. RPM の再インストール

OpenShift Container Platform 3.8 および 3.9 リポジトリーを無効にし、3.7 リポジトリーを再び有効にします。

```
# subscription-manager repos \
    --disable=rhel-7-server-ose-3.8-rpms \
    --disable=rhel-7-server-ose-3.9-rpms \
    --enable=rhel-7-server-ose-3.7-rpms
```

各マスターで、以下のパッケージをインストールします。

```
# yum install atomic-openshift \
    atomic-openshift-clients \
    atomic-openshift-node \
    atomic-openshift-master-api \
    atomic-openshift-master-controllers \
    openvswitch \
    atomic-openshift-sdn-ovs \
```

```
    tuned-profiles-atomic-openshift-node \
    atomic-openshift-excluder \
    atomic-openshift-docker-excluder
```

各ノードで、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 \
    openvswitch \
    atomic-openshift-sdn-ovs \
    tuned-profiles-atomic-openshift-node \
    atomic-openshift-excluder \
    atomic-openshift-docker-excluder
```

外部 **etcd** クラスターを使用している場合、各 **etcd** メンバーで以下のパッケージをインストールします。

```
# yum install etcd
```

6.7. ETCD の復元

etcd 設定ファイルの復元手順では、適切なファイルを置き換えてからサービスを再起動します。

etcd ホストが破損し、**/etc/etcd/etcd.conf** ファイルが失われる場合は、以下を使用してこれを復元します。

```
$ ssh master-0
# cp /backup/yesterday/master-0-files/etcd.conf /etc/etcd/etcd.conf
# restorecon -Rv /etc/etcd/etcd.conf
# systemctl restart etcd.service
```

この例では、バックアップファイルは **/backup/yesterday/master-0-files/etcd.conf** パスに保存されます。ここでは外部 **NFS** 共有、**S3** バケットまたは他のストレージソリューションとして使用できます。

6.7.1. etcd v2 および v3 データの復元

以下のプロセスでは正常なデータファイルを復元し、**etcd** クラスターを単一ノードとして起動してから、**etcd** クラスターが必要な場合に残りのノードを追加します。

手順

1. すべての **etcd** サービスを停止します。

   ```
   # systemctl stop etcd.service
   ```

2. 適切なバックアップが復元されていることを確認するには、**etcd** ディレクトリーを削除します。

   - ディレクトリーを削除する前に現在の **etcd** データをバックアップするには、以下のコマンドを実行します。

     ```
     # mv /var/lib/etcd /var/lib/etcd.old
     # mkdir /var/lib/etcd
     ```

```
# chown -R etcd.etcd /var/lib/etcd/
# restorecon -Rv /var/lib/etcd/
```

- または、ディレクトリーおよび **etcd**、データを削除するには、以下のコマンドを実行します。

```
# rm -Rf /var/lib/etcd/*
```

注記

オールインワンクラスターの場合、**etcd** データディレクトリーは **/var/lib/origin/openshift.local.etcd** ディレクトリーに置かれます。

3. 正常なバックアップデータファイルをそれぞれの **etcd** ノードに復元します。この手順は、**etcd** と同じ場所に配置されているマスターホストを含むすべての **etcd** ホストで実行します。

```
# cp -R /backup/etcd-xxx/* /var/lib/etcd/
# mv /var/lib/etcd/db /var/lib/etcd/member/snap/db
# chcon -R --reference /backup/etcd-xxx/* /var/lib/etcd/
# chown -R etcd:etcd /var/lib/etcd/R
```

4. 各ホストで **etcd** サービスを実行し、新規クラスターを強制的に実行します。これにより **etcd** サービスのカスタムファイルが作成されます。これにより、**--force-new-cluster** オプションを追加して実行コマンドが上書きされます。

```
# mkdir -p /etc/systemd/system/etcd.service.d/
# echo "[Service]" > /etc/systemd/system/etcd.service.d/temp.conf
# echo "ExecStart=" >> /etc/systemd/system/etcd.service.d/temp.conf
# sed -n '/ExecStart/s/"$/ --force-new-cluster"/p' \
    /usr/lib/systemd/system/etcd.service \
    >> /etc/systemd/system/etcd.service.d/temp.conf

# systemctl daemon-reload
# systemctl restart etcd
```

5. エラーメッセージの有無を確認します。

```
$ journalctl -fu etcd.service
```

6. 健全性のステータスを確認します。

```
# etcdctl2 cluster-health
member 5ee217d17301 is healthy: got healthy result from
https://192.168.55.8:2379
cluster is healthy
```

7. クラスターモードで **etcd** サービスを再起動します。

```
# rm -f /etc/systemd/system/etcd.service.d/temp.conf
# systemctl daemon-reload
# systemctl restart etcd
```

8. 健全性のステータスとメンバーの一覧を確認します。

```
# etcdctl2 cluster-health
member 5ee217d17301 is healthy: got healthy result from
https://192.168.55.8:2379
cluster is healthy

# etcdctl2 member list
5ee217d17301: name=master-0.example.com
peerURLs=http://localhost:2380 clientURLs=https://192.168.55.8:2379
isLeader=true
```

9. 最初のインスタンスの実行後に、残りの **etcd** サーバーを復元できます。

6.7.1.1. peerURLs パラメーターの修正

データの復元および新規クラスターの作成後に、**peerURLs** パラメーターは、 **etcd** がピア通信をリッスンする IP の代わりに **localhost** を示します。

```
# etcdctl2 member list
5ee217d17301: name=master-0.example.com peerURLs=http://*localhost*:2380
clientURLs=https://192.168.55.8:2379 isLeader=true
```

6.7.1.1.1. 手順

1. **etcdctl member list** を使用してメンバー ID を取得します。

```
`etcdctl member list`
```

2. **etcd** がピア通信をリッスンする IP を取得します。

```
$ ss -l4n | grep 2380
```

3. メンバー情報をこの IP で更新します。

```
# etcdctl2 member update 5ee217d17301 https://192.168.55.8:2380
Updated member with ID 5ee217d17301 in cluster
```

4. 検証するには、IP がメンバーの一覧にあることを確認します。

```
$ etcdctl2 member list
5ee217d17301: name=master-0.example.com
peerURLs=https://*192.168.55.8*:2380
clientURLs=https://192.168.55.8:2379 isLeader=true
```

6.7.2. v3 の etcd の復元

v3 データの復元手順は v2 データの復元プロセスと同様です。

スナップショットの整合性については、復元時にオプションで検証できます。スナップショットが **etcdctl snapshot save** を使用して取得される場合、これには **etcdctl snapshot restore** でチェックされる整合性ハッシュが含まれます。スナップショットがデータディレクトリーからコピー

される場合には整合性ハッシュはなく、復元は **--skip-hash-check** を使用して実行されます。

重要

v3 データのみを復元する手順は単一 etcd ホストで実行される必要があります。その後に残りのノードをクラスターに追加することができます。

手順

1. すべての etcd サービスを停止します。

```
# systemctl stop etcd.service
```

2. 古いデータについては、**etcdctl** が復元手順が実行されるノードでその再作成を実行するため、それらすべてをクリアします。

```
# rm -Rf /var/lib/etcd
```

3. **snapshot restore** コマンドを実行し、**/etc/etcd/etcd.conf** ファイルの値を置き換えます。

```
# etcdctl3 snapshot restore /backup/etcd-xxxxxx/backup.db \
  --data-dir /var/lib/etcd \
  --name master-0.example.com \
  --initial-cluster "master-0.example.com=https://192.168.55.8:2380"
\ --initial-cluster-token "etcd-cluster-1" \
  --initial-advertise-peer-urls https://192.168.55.8:2380

2017-10-03 08:55:32.440779 I | mvcc: restore compact to 1041269
2017-10-03 08:55:32.468244 I | etcdserver/membership: added member
40bef1f6c79b3163 [https://192.168.55.8:2380] to cluster
26841ebcf610583c
```

4. パーミッションおよび **selinux** コンテキストを復元ファイルに復元します。

```
# chown -R etcd.etcd /var/lib/etcd/
# restorecon -Rv /var/lib/etcd
```

5. etcd サービスを起動します。

```
# systemctl start etcd
```

6. エラーメッセージの有無を確認します。

```
$ journalctl -fu etcd.service
```

6.8. OPENSHIFT 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/sysconfig/atomic-openshift-master-api
/etc/sysconfig/atomic-openshift-master-api
# cp ${MYBACKUPDIR}/etc/sysconfig/atomic-openshift-master-
controllers /etc/sysconfig/atomic-openshift-master-controllers
# 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
# systemctl enable atomic-openshift-master-api
# systemctl enable atomic-openshift-master-controllers
# systemctl enable atomic-openshift-node
# systemctl start atomic-openshift-master-api
# systemctl start atomic-openshift-master-controllers
# systemctl start atomic-openshift-node
```

2. それぞれの OpenShift Container Platform ノードで、バックアップから **node-config.yaml** ファイルを復元し、**atomic-openshift-node** サービスを有効にし、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
```

6.9. ダウングレードの検証

1. ダウングレードを検証するには、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. 次に、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:v3.7.23",
# oc get -n default dc/router -o json | grep \"image\"
    "image": "openshift3/ose-haproxy-router:v3.7.23",
```

3. 診断ツールをマスターで使用し、共通の問題を検索し、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.
```

第7章 既知の問題

7.1. 概要

OpenShift Container Platform クラスターのアップグレード時に、2 つの主なデータ移行が実行されます。最初は、現行バージョンの OpenShift Container Platform の実行中に生じ、2 つ目はアップグレードの完了後に生じます。

アップグレード前の移行に失敗した場合、アップグレードは中止し、クラスター管理者はアップグレードを再度実行する前にデータの不整合についての問題を解決する必要があります。アップグレード後の移行に失敗する場合には、アップグレードが完了しているため、クラスター管理者はデータ移行の問題をできるだけ早く解決する必要があります。ここでは、既知のデータの不整合の問題とその解決方法について説明します。

7.2. サービス定義の重複するポート

Service オブジェクトにはより厳格な検証ルールがあります。その 1 つは重複した**Ports** エントリーを許可しないルールです (例: **Protocol** と **Port** のペア)。(BZ#1563929)

エラー出力

```
TASK [Upgrade all storage]
*******************************************************************************
*******************************************************************************
************************
fatal: [test.linux.lan]: FAILED! => {
    "changed": true,
    "cmd": [
        "oc",
        "adm",
        "--config=/etc/origin/master/admin.kubeconfig",
        "migrate",
        "storage",
        "--include=*",
        "--confirm"
    ],
    "delta": "0:07:06.574833",
    "end": "2018-04-03 11:22:07.834827",
    "failed_when_result": true,
    "msg": "non-zero return code",
    "rc": 1,
    "start": "2018-04-03 11:15:01.259994",
}

STDOUT

error: Service "my-service" is invalid: spec.ports[5]: Duplicate value:
api.ServicePort{Name:"", Protocol:"TCP", Port:8500,
TargetPort:intstr.IntOrString{Type:0, IntVal:0, StrVal:""}, NodePort:0}
summary: total=37768 errors=1 ignored=0 unchanged=37767 migrated=0
info: to rerun only failing resources, add --include=services
error: 1 resources failed to migrate
```

この問題を解決するには、以下を実行します。

1. クラスター管理者として CLI にログインし、エラーに一覧表示されている **Service** オブジェクトを検査します。

2. 重複したポートを削除するためにオブジェクトの **.spec.ports** を編集します。この際、**.spec.ports[*].name** と **.spec.ports[*].port** は一意である必要があることに留意してください。