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

Last Updated: 2021-09-22

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

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

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

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も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 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。

- 接続されていないクラスターの更新を完了すると、イメージの新しいバージョンでイメージレジストリーを更新する必要があります。そうでない場合には、クラスターの更新に失敗します。たとえば、3.1.153 から **3.11.153** から **3.11.157** に更新する場合は **v3.11.157** イメージタグが存在することを確認します。

- 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。

- クラスターが現在の 前提条件を満たしていることを確認 します。そうでない場合は、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 ストレージ用に保存された API オブジェクトを更新します。

```
$ 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となる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 リポジトリーを無効にして、3.11 リポジトリーを有効にします。**rhel-7-server-ansible-2.9-rpms** リポジトリー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-rhscl-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-rhscl-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。

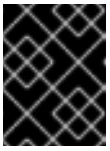
a. (初期インストールまたは最近のクラスターのアップグレードなどで) Ansible Playbook を最後に実行した後にマスターまたはノード設定ファイルに設定変更を手動で加えた場合は、それらの変更がインベントリーファイルに反映されていることを確認します。手動で加えた変更に関連する変数については、アップグレードの実行前に同等の変更をインベントリーファイルに適用してください。これを実行しないと、手動による変更がアップグレード時にデフォルト値で上書きされ、Pod が適切に実行されなくなるか、または他のクラスターの安定性に問題が生じる可能性があります。

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、証明書が 180 日間有効とする場合に、**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が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ー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す。

- 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	• 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、そのタスクは **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	• 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、そのタスクは **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、そのタスクは **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	• 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、そのタスクは **delegate_to** または **local_action** を使用する必要があります。
openshift_node_upgrade_hook	• 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、そのタスクは **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	• 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、そのタスクは **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、インベントリーファイルで **openshift_pkg_version** および **openshift_image_tag** パラメーターの両方に値を設定します。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同じバージョンの OpenShift Container Platform 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 など、メジャーアップデート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にはこれらの値を設定する必要があります。

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
```

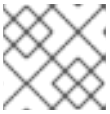
2.4. 最新 OPENSHIFT CONTAINER PLATFORM リリースへのアップグレード

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および完全なシステムの再起動を実行するには、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。これを設定しないと、サービスはマスターで起動しますが、システムは再起動されません。

   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.yaml** 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.yaml** 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とすべてのノードホストを手動で再起動してください。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。

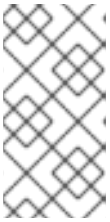
2.5. コンテナー化された GLUSTERFS を使用する場合の OPENSHIFT CONTAINER PLATFORM のアップグレード

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。

コンテナー化された 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)。

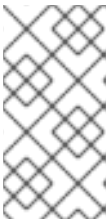
2. 標準 **infra** ノード (ルーター、レジストリー、ロギング、およびメトリクス) をアップグレードします。



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. 以下のパラメーターを、**/etc/ansible/hosts** のインベントリーファイルに追加します。

```
openshift_hosted_registry_storage_kind=glusterfs
openshift_storage_glusterfs_heketi_image=registry.access.redhat.com/rhgs3/rhgs-volmanager-rhel7:<your_cns_vesion>(1)
openshift_storage_glusterfs_image=registry.access.redhat.com/rhgs3/rhgs-server-rhel7:<your_cns_vesion>(2)
openshift_storage_glusterfs_block_image=registry.access.redhat.com/rhgs3/rhgs-gluster-block-prov-rhel7:<your_cns_vesion>(3)
openshift_storage_glusterfs_s3_image=registry.access.redhat.com/rhgs3/rhgs-s3-server-rhel7:<your_cns_vesion>(4)
```

   (1)(2)(3)(4)**v3.9** または **v3.11.3** などの CNS バージョンを指定します。

   b. 以下のリソースでは、イメージタグが存在する場合には **latest** から **<your_cns_version>** に更新します (**v3.9** または **v3.11.3** など)。

```
$ oc edit -n <glusterfs_namespace> ds glusterfs-<name>
$ oc edit -n <glusterfs_namespace> dc heketi-<name>
$ oc edit -n <glusterfs_namespace> dc glusterblock-<name>-provisioner-dc
$ oc edit -n <glusterfs_namespace> dc gluster-<name>-<account>-s3
```

   c. 一度に 1 つのノードだけがアップグレードされるように、アップグレードするノードにラベルを追加します。

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

   d. 再起動する GlusterFS Pod は終了しないでください。

   e. 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 は終了されないはずです。

f. GlusterFS Pod が再生成され、表示されるまで待機します。

g. 各 Pod の再起動後に基本的なヘルスチェックを実行し、ヘルスチェックに合格していることを確認します。

h. **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。

i. 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。

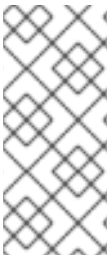
```
$ oc label node <node_name> type-
```

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

アップグレードで **logging-fluentd** と **logging-curator** ConfigMaps が置き換えられます。現在の **logging-fluentd** と **logging-curator** ConfigMaps 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 の **openshift_logging_fluentd_replace_configmap** と **openshift_logging_curator_replace_configmap** のパラメーターを **no** に設定します。

EFK 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 の変更に関する重要な情報については、Elasticsearch breaking changes を確認する必要があります。

Elasticsearch 5 では、インデックス構造に大きな変更がある点に留意することが重要です。以前のバージョンでは、Elasticsearch はフィールド名のドット文字 **.** を許可していました。バージョン 5 では、Elasticsearch は Elasticsearch フィールド名のすべてのドットをネストされた構造として解釈します。フィールドにドットが含まれる場合は、ドットの後の文字列はフィールドの型として解釈され、アップグレード中にマッピングの競合が発生します。

競合の可能性を特定できるように、OpenShift Container Platform は Elasticsearch フィールドを検査して、フィールド名にドットが含まれるかどうかを判別します。

たとえば、以下のフィールドは Elasticsearch 2 では使用できていました。

```
{
  "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になり、**name** 文字列は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
  }
}
```

この場合にアップグレードすると、**field** フィールドには 2 つの異なるタイプが指定されてしまいます。

競合するインデックスを保持する必要がある場合は、データを再インデックスし、競合するデータ構造を取り除くように、ドキュメントを変更してください。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、インデックス名にドットが付いたフィールドが含まれるかどうかを判別できます。

注記

以下のコマンドは、**jq** JSON プロセッサーを使用して必要なデータを直接取得します。バージョンによっては、Red Hat Enterprise Linux (RHEL) では **jq** のパッケージが提供されない場合があります。外部のソースからインストール、またはサポート対象外の場所から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あるかないかによって異なります。

2.6.1.2. フィールド名にドットがある場合のアップグレード

上記のスクリプト で、インデックスから、名前にドットが含まれるフィールドがあることが分かると、以下の手順を使用してこの問題を修正し、アップグレードしてください。

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のあるその他の openshift_logging_* 変数を更新します。**openshift_logging_elasticsearch_replace_configmap** パラメーターを **true** に設定すると、**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場合があります。デフォルトは **false** に設定されます。詳細は、ロギング Ansible 変数を指定するパラメーター を参照してください。

注記

EFK コンポーネントの Fluentd **DeploymentConfig** オブジェクトおよび **DaemonSet** オブジェクトがすでに以下で設定されている場合:

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

最新バージョン **<image_name>** は、Pod の再デプロイ先のノードにローカルに保存される **<image_name:vX.Y>** と同じ名前のイメージが存在する場合にプルされないことがあります。イメージのバージョンが v3.11 であり、Playbook を使用して最新バージョンにアップグレードする必要がある場合には、**openshift_image_tag=v3.11.<Z>** または **oreg_url=registry.access.redhat.com/openshift3/ose-${component}:v3.11.<Z>** Ansible パラメーターを設定します。

3. データの取得を停止する Fluentd Pod のスケジュールを解除して、クラスターの状態が変更しないようにします。
たとえば、Fluentd Pod のノードセレクターを、いずれのノードにも一致しないものに切り換えることができます。

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

4. 関連するすべてのインデックスで Elasticsearch インデックスのフラッシュ を実行します。フラッシュプロセスでは、すべてのログをメモリーからディスクに永続化し、アップグレード時に Elasticsearch がシャットダウンしてもログが失われないようにできます。

5. 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。

   - 特定の Elasticsearch クラスターの 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。

6. ドット文字を含むファイル名については、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。

   - インデックスを削除します。この方法は、アップグレード時にマッピングの競合回避に適しています。

   - データを再インデックス化して、ドキュメントを変更し、競合するデータ構造を削除します。このメソッドはデータを保持します。マッピングの競合の可能性は、Elasticsearch ドキュメントの Mappging changes を参照してください。

7. オンラインまたはオフラインでバックアップを繰り返します。

8. 「**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 で、インデックスから名前にドットが含まれるフィールドがあることが分かると、以下の手順を使用してアップグレードを行います。

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のあるその他の openshift_logging_* 変数を更新します。**openshift_logging_elasticsearch_replace_configmap** パラメーターを **true** に設定すると、**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場合があります。デフォルトは **false** に設定されます。詳細は、ロギング Ansible 変数を指定するパラメーター を参照してください。

注記

EFK コンポーネントの Fluentd **DeploymentConfig** オブジェクトおよび **DaemonSet** オブジェクトがすでに以下で設定されている場合:

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

最新バージョン <image_name> は、Pod の再デプロイ先のノードにローカルに保存される **<image_name:vX.Y>** と同じ名前のイメージが存在する場合にプルされないことがあります。イメージのバージョンが v3.11 であり、Playbook を使用して最新バージョンにアップグレードする必要がある場合には、**openshift_image_tag=v3.11.<Z>** または **oreg_url=registry.access.redhat.com/openshift3/ose-${component}:v3.11.<Z>** Ansible パラメーターを設定します。

3. オプションとして、Fluentd Pod のスケジュールを解除して Elasticsearch Pod をスケールダウンし、データの処理を停止して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 のノードセレクターを、いずれのノードにも一致しないものに切り換えることができます。

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

4. 必要に応じて、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。
   - 特定の Elasticsearch クラスターの 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。

5. 「**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。

6. オプションで、Elasticsearch restore module 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のあるその他の 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。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。

2.7. 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。

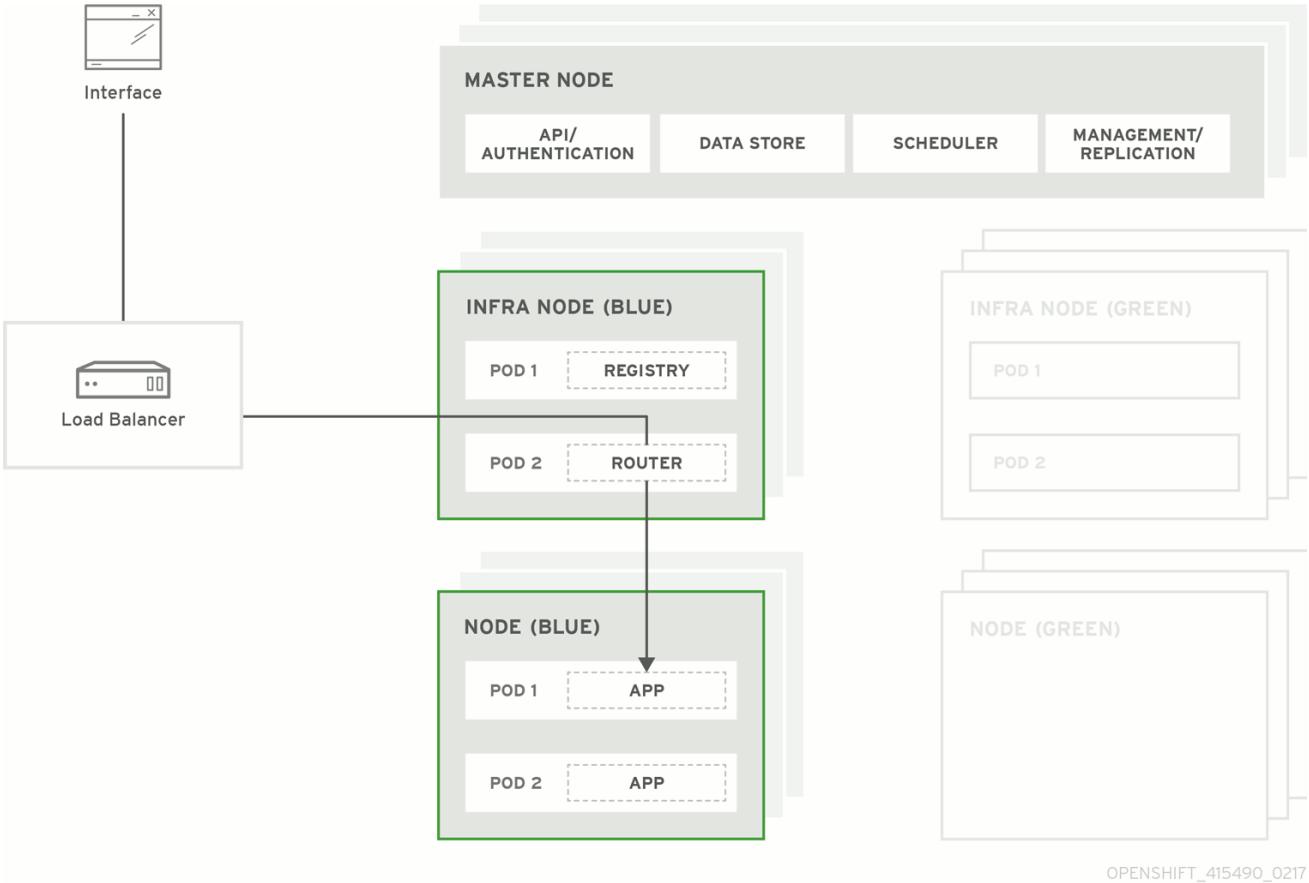
この方式では、管理者は新規デプロイメントの検証後、古いノードホストセット (例: 「**Blue**」デプロイメント) から新規セット (例: 「**Green** デプロイメント) にトラフィックを切り換えることができます。問題が検出される場合も、古いデプロイメントへのロールバックを簡単かつすぐに実行できます。

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



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。

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 (コンピュートノード) 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/hostna
me=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、必要なコンテナー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、環境の容量が十分に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タ更新) 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-b7xbn   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
```

4. 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。

5. 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が、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。

6. 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ら、以下の手順に従います。

1. 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>
```

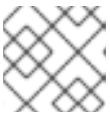
4.1.1. OpenShift Container Storage を実行するノードのアップグレード

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. まず、OpenShift Container Storage がデプロイされた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および Pod セレクターを確認します。

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



注記

出力に Pod セレクターを含めるには **-o wide** を使用します。

これらのセレクターは、それぞれ **NODE-SELECTOR** および **SELECTOR** で参照できます。以下のコマンド例では、それぞれ **glusterfs=storage-host** と **glusterfs=storage-pod** を使用します。

3. daemonset のノードセレクターを使用して、どのホストにラベルが指定されているかを確認し、DeamonSet から Pod が実行されていることを確認します。

```
$ 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た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、ノードが OS をアップグレードできるようになりました。

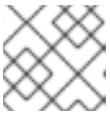
1. ノードで OpenShift Container Storage Pod を再起動するには、demonset ラベルでノードのラベルを変更します。

```
$ 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 セレクターでは、OS をアップグレードしたノードで実行されている Pod を検索し、新しく起動された Pod の名前を判別します。

```
$ 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。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生じ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ません。そのため、ダウングレードはしないでください。

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.yaml.<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 のリポジトリーを無効にし、3.10 のリポジトリー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. 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/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 ノードの設定マップ を更新し、**atomic-openshift-node** サービスを有効にして再起動します。

```
# 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

To verify the downgrade:

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.
```