OpenShift Container Platform 3.10

リリースノート

Last Updated: 2019-03-25

法律上の通知

概要

目次

第1章 概要

以下の OpenShift Container Platform 3.10 リリースノートでは、新機能のすべて、以前のバージョンからの主な修正、一般公開バージョンの既知の問題についてまとめています。

1.1. バージョン管理ポリシー

OpenShift Container Platform では、サポートされているすべての API の厳密な後方互換対応を保証しています。ただし、アルファ API (通知なしに変更される可能性がある) およびベータ API (後方互換性の対応なしに変更されることがある) は例外となります。

OpenShift Container Platform のバージョンは、マスターとノードホストの間で一致している必要があります。ただし、クラスターのアップグレード時にバージョンが一時的に一致しなくなる場合を除きます。たとえば、3.10 クラスターでは、すべてのマスターが 3.10 であり、すべてのノードが 3.10 である必要があります。ただし、OpenShift Container Platform は、引き続き新規サーバーで以前のバージョンの **oc** クライアントをサポートします。たとえば、3.5 の **oc** は 3.4、3.5 および 3.6 のサーバーで機能します。

セキュリティーとは関連性のない理由で API が変更された場合には、古いバージョンの **oc** が更新されるように 2 つ以上のマイナーリリース (例: 3.4、3.5、3.6) 間での更新が行われます。新機能を使用するには新規バージョンの **oc** が必要です。3.2 サーバーにはバージョン 3.1 の **oc**で使用できない機能が追加されている場合や、バージョン 3.2 の **oc** には 3.1 サーバーでサポートされていない追加機能が含まれる場合があります。

表1.1 互換性に関する表

	X.Y (**oc** クライアント)	**X.Y+N** [a] (**oc** クライアント)
X.Y (サーバー)	1	3
X.Y+N [a] (サーバー)	2	1

[a] **N** は 2 以上

1 完全に互換性があります。

2 **oc** クライアントはサーバー機能にアクセスできない場合があります。

3 **oc** クライアントは、アクセスされるサーバーと互換性のないオプションや機能を提供する可能性があります。

第2章 OPENSHIFT CONTAINER PLATFORM 3.10 リリースノート

2.1. 概要

Red Hat OpenShift Container Platform では、設定や管理のオーバーヘッドを最小限に抑えながら、セキュアでスケーラブルなリソースに新規および既存のアプリケーションをデプロイするハイブリッドクラウドアプリケーションプラットフォームを開発者や IT 組織に提供します。OpenShift Container Platform は、Java、Javascript、Python、Ruby および PHP など、幅広いプログラミング言語およびフレームワークをサポートしています。

Red Hat Enterprise Linux および Kubernetes にビルドされる OpenShift Container Platform は、最新のエンタープライズレベルのアプリケーションにセキュアでスケーラブルなマルチテナント対応のオペレーティングシステムを提供するだけでなく、統合アプリケーションランタイムやライブラリーを提供します。OpenShift Container Platform を使用することで、組織はセキュリティー、プライバシー、コンプライアンス、ガバナンスの各種の要件を満たすことができます。

2.2. 本リリースについて

Red Hat OpenShift Container Platform バージョン 3.10 (RHBA-2018:1816) をご利用いただけるようになりました。このリリースは、OpenShift Origin 3.10 をベースとしており、Kubernetes 1.10 を使用します。以下では、OpenShift Container Platform 3.10 に関連する新機能、変更点、バグ修正および既知の問題について説明します。

OpenShift Container Platform 3.10 は RHEL 7.4 および 7.5 上で、Docker 1.13 を含む Extras の最新パッケージでサポートされます。また、Atomic Host 7.4.5 以降のバージョンでもサポートされます。**docker-latest** パッケージは非推奨となりました。

TLSV1.2 は、OpenShift Container Platform バージョン 3.4 以降でサポートされる唯一のセキュリティーバージョンです。TLSV1.0 または TLSV1.1 をお使いの場合は、更新する必要があります。

初回のインストールについては、『Installing Clusters』ドキュメントを参照してください。

以前のバージョンから今回のリリースにアップグレードするには、『Upgrading Clusters』ドキュメントを参照してください。

2.3. 新機能および改良された機能

今回のリリースでは、以下のコンポーネントおよび概念に関連する拡張機能が追加されました。

2.3.1. インストール

2.3.1.1. 重要な変更点

OpenShift Container Platform 3.10 のインストールプロセスに数多くの重要な変更点が加えられました。

- コントロールプレーンのコンポーネント (etcd、API サーバー、およびコントローラー) はマスター上の kubelet により静的 Podとして実行されるようになりました。現時点では、コントロールプレーンのコンポーネントを実行する方法としてこの方法のみがサポートされています。アップグレードによりこの方法への変更も自動的に加えられることになります。コント

ロールプレーンが表示される際に **oc get pods -n kube-system** を実行すると Pod が表示され、通常の **oc** CLI コマンドを使用してこれらを実行し、ログに記録し、検査することができます。

- OpenShift Container Platform のコンテナー化モード (コンテナーを Docker から直接起動する) はサポートされなくなりました。
- openshift-sdn および openvswitch は daemonset で実行されます。
- **/etc/sysconfig/(origin|atomic-openshift)-(api|controllers)** ファイルは使用されなくなりました。新規の **/etc/origin/master/master.env** ファイルは、静的 Pod の環境変数を設定するために使用できます。環境変数として利用可能な設定のセットは制限されています。OpenShift Container Platform の今後のバージョンでは、コントロールプレーンファイルを優先して使用するためにこの設定を削除します。**master.env** ファイルは最終的な手段として使用するものとし、非推奨とみなしてください。静的 Pod のホストレベルのデバッグについてのギャップを埋めるために、shim のセットが **/usr/local/bin** にインストールされます。
  - **/usr/local/bin/master-restart (etcd|api|controllers)**
    - これにより、kubelet が名前付きコンポーネントの静的 Pod 全体を起動します。
    - これはブロック操作 **ではなく**、複雑なスクリプト化を防ぐために意図的に制限されています。
    - 以下を実行して再起動をトリガーすることもできます。

      ```
      $ oc annotate -n kube-system POD_NAME gen=N --overwrite
      ```

  - **/usr/local/bin/master-logs (etcd etcd|api api|controllers controllers)**
    - 所定コンポーネントの最近のコンテナーログを出力します (コンポーネント名とコンテナー名が一致します)。
    - 追加のオプションがコンテナーランタイムに渡されます。
    - **oc logs -n kube-system POD_NAME** を代わりに使用します。
  - **/usr/local/bin/master-exec (etcd etcd|api api|controllers controllers)**
    - etcd のバックアップをサポートできるようにコマンドをコンテナー内で実行します。これらのコマンドは慎重に使用してください。

詳細については、「コンテナー化インストール方法を削除」および「静的 Pod としてのコントロールプレーン」を参照してください。

デバッグに関するヒント

oc get nodes を使用してノードが準備ができている状態かどうかを確認します。ノードの準備ができていない場合、通常は以下の状況であることが考えられます。

- 証明書の承認を待機しています。**oc get csr** を実行して要求されているすべての証明書を表示します。**oc get csr -o name | xargs oc adm certificate approve** を実行してすべての証明書を承認します。証明書を承認した後には、ノードが準備できている状態にあることを再び報告するまでに数秒の時間がかかる場合があります。
- SDN Pod がノード上で実行されていません。通常は、リセットをトリガーするために所定のノードで SDN または OVS Pod を削除できます。その後にプロセスが続行されます。

- API またはコントローラーコンテナーが実行されていない場合 (**`docker ps | grep api`**)、**`master-logs api`** を使用して理由を確認してください。通常は、失敗した設定に原因があります。

- 1.10 リベースで解決される kubelet が停止状態になるという既知の問題があります。この状態では、kubelet が **system:anonymous cannot access resource foo** などのメッセージを表示します。これは、kubelet が証明書を更新する前にそれらの証明書が期限切れになったことを示します。kubelet を再起動してもこの問題が解決されない場合は、**/etc/origin/node/certificates/** の内容を削除してから kubelet を再起動します。

- 接続されないコンポーネントがある場合、つまり、コンポーネントがクラッシュのループに入る場合は、その Pod のログでバグを報告してください。これは、コンテナーのネットワークが失われても kubelet または SDN がそれを認識しないという最も一般的な openshift-sdn / OVS の問題です。

Atomic Host が非推奨となる
Atomic Host が非推奨となりました。Atomic Host は OpenShift Container Platform 3.11 で引き続きサポートされますが、OpenShift Container Platform 4.0 では削除されます。

2.3.2. OpenShift Automation Broker

2.3.2.1. OpenShift Automation Broker はローカル etcd ではなく CRD を使用する

OpenShift Automation Broker はローカル etcd インスタンスの代わりにカスタムリソース定義 (CRD) を使用します。

`openshift-ansible` について etcd から CRD への移行パスが用意されています。

2.3.2.2. mediawiki-abp サンプルが更新される

mediawiki-apb Ansible Playbook Bundle (APB) のサンプルがバージョン 1.27 を使用するように更新されました。

2.3.3. ストレージ

2.3.3.1. OpenStack Manila を使用した永続ボリューム (PV) のプロビジョニング (テクノロジープレビュー)

OpenStack Manila を使用した永続ボリューム (PV) のプロビジョニングは現在テクノロジープレビューとして提供されており、実稼働環境のワークロードには適していません。

OpenShift Container Platform は、OpenStack Manila 共有ファイルシステムサービスを使用して PV をプロビジョニングできます。

詳細は、「Persistent Storage Using OpenStack Manila」を参照してください。

2.3.3.2. PV のサイズ変更 (テクノロジープレビュー)

永続ボリューム (PV) のサイズ変更は現在テクノロジープレビューとして提供されており、実稼働環境のワークロードには適していません。

glusterFS について、OpenShift Container Platform からオンラインで Persistent Volume Claim (永続ボリューム要求、PVC) を拡張できます。

1. **`allowVolumeExpansion=true`** を指定してストレージクラスを作成します。
2. PVC はこのストレージクラスを使用して、要求を送信します。
3. PVC は拡張したサイズを新たに指定します。
4. 基礎となる PV のサイズが変更されます。

詳細は、「Expanding Persistent Volumes」を参照してください。

2.3.3.3. コンテナーストレージインターフェース (テクノロジープレビュー)

コンテナーストレージインターフェース (CSI) は現在テクノロジープレビューとして提供されており、実稼働環境のワークロードには適していません。

CSI により、OpenShift Container Platform は CSI インターフェースを永続ストレージとして実装するストレージバックエンドからストレージを消費できます。

詳細は、「Persistent Storage Using Container Storage Interface (CSI)」を参照してください。

2.3.3.4. ローカル一時ストレージの保護 (テクノロジープレビュー)

ローカル一時ストレージの保護は現在テクノロジープレビューとして提供されており、実稼働環境のワークロードには適していません。

ユーザーがノードのローカルストレージをそれらの Pod や同じノード上にある他の Pod で使い尽くさないようにするために、ノード上でローカル一時ストレージ機能を制御できるようになりました。

この機能はデフォルトで無効にされています。有効にされる場合、OpenShift Container Platform クラスターは一時ストレージを使用して、クラスターを破棄した後に永続させる必要のない情報を保存します。

詳細は、「Configuring Ephemeral Storage」を参照してください。

2.3.3.5. テナント駆動型のストレージのスナップショット (テクノロジープレビュー)

テナント駆動型のストレージのスナップショット機能は現在テクノロジープレビュー として提供されており、実稼働環境のワークロードには適していません。

テナントは、アプリケーションデータのスナップショットを作成するために割り当てられる、永続ボリューム (PV) をサポートする基礎のストレージ技術を活用できるようになりました。またテナントは、以前のアプリケーションから現在のアプリケーションに指定のスナップショットを復元できるようになりました。

外部プロビジョナーは、EBS、GCE pDisk および HostPath にアクセスするために使用されます。このテクノロジープレビュー機能は、EBS および HostPath についてテスト済みです。テナントは Pod を手動で停止して、起動する必要があります。

1. 管理者はクラスターの外部プロビジョナーを実行します。それらは Red Hat Container Catalog のイメージになります。
2. テナントは PVC を作成して、サポートされるストレージソリューションのいずれかの PV を所有します。管理者は、以下を設定して、クラスターに新しい **`StorageClass`** を作成する必要があります。

```
kind: StorageClass
```

```
apiVersion: storage.k8s.io/v1
metadata:
  name: snapshot-promoter
provisioner: volumesnapshot.external-storage.k8s.io/snapshot-
promoter
```

3. テナントは **gce-pvc** という名前の PVC のスナップショットを作成し、作成されるスナップショットの名前は **snapshot-demo** になります。

```
$ oc create -f snapshot.yaml

apiVersion: volumesnapshot.external-storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: snapshot-demo
  namespace: myns
spec:
  persistentVolumeClaimName: gce-pvc
```

4. これで、それらの Pod をそのスナップショットに復元できます。

```
$ oc create -f restore.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: snapshot-pv-provisioning-demo
  annotations:
    snapshot.alpha.kubernetes.io/snapshot: snapshot-demo
spec:
  storageClassName: snapshot-promoter
```

2.3.4. スケーリング

2.3.4.1. クラスターの制限

OpenShift Container Platform 3.10 の クラスター制限 に関するガイダンスが更新され、ご利用いただけるようになりました。

2.3.4.2. デバイスプラグイン

テクノロジープレビューであったデバイスプラグインは、OpenShift Container Platform 3.10 で一般に利用できるようになりました。OpenShift Container Platform はデバイスプラグイン API をサポートしますが、デバイスプラグインコンテナーは個別のベンダーによりサポートされます。

デバイスプラグインを使用すると、カスタムコードを作成せずに特定のデバイスタイプ (GPU、InfiniBand、またはベンダー固有の初期化およびセットアップを必要とする他の同様のコンピューティングリソース) を OpenShift Container Platform Pod で使用できます。デバイスプラグインは、クラスター全体でハードウェアデバイスを使用するための一貫性のある移植可能なソリューションを提供します。デバイスプラグインはこれらのデバイスのサポートを拡張メカニズムによって提供します。このメカニズムによって、これらのデバイスはコンテナーで利用可能となり、デバイスのヘルスチェックやそれらの共有におけるセキュリティー保護が可能になります。

デバイスプラグインはノード上で実行される gRPC サービスであり (**atomic-openshift-node.service** の外部にある)、これは特定のハードウェアリソースを管理します。

デバイスプラグインの概念に関する詳細は、『開発者ガイド』を参照してください。

2.3.4.3. CPU マネージャー

テクノロジープレビューであった CPU マネージャーは OpenShift Container Platform 3.10 で一般に利用できるようになりました。

CPU マネージャーは CPU グループを管理して、ワークロードを固有の CPU に制限します。

CPU マネージャーは、以下のような属性を持つワークロードに役立ちます。

- 可能な限り長い CPU 時間を必要とする
- プロセッサーのキャッシュミスの影響を受ける
- レイテンシーが低いネットワークアプリケーションである
- 他のプロセスと連携し、単一のプロセッサーキャッシュを共有することによる利点がある

詳細は、「CPU マネージャーの使用」を参照してください。

2.3.4.4. デバイスマネージャー

テクノロジープレビューであったデバイスマネージャーは OpenShift Container Platform 3.10 で一般に利用できるようになりました。OpenShift Container Platform はデバイスプラグイン API をサポートしますが、デバイスプラグインコンテナーは個別のベンダーによりサポートされます。

一部のユーザーは、Pod 定義内でハードウェアデバイスのリソース制限を設定する必要があり、スケジューラーを使ってこれらのリソースに関連してクラスター内のノードを検出する必要があります。この場合、Kubernetes では、kubernetes 内でコアコードを変更せずにハードウェアベンダーが kubelet にリソースを公開できる方法が必要でした。

kubelet には、プラグインで拡張可能なデバイスマネージャーが組み込まれました。ドライバーサポートはノードレベルで読み込むことができます。その後に、ユーザーまたはベンダーがドライバーで表示される要求済みハードウェアリソースの停止/開始/アタッチ/割り当て要求をリッスンするプラグインを作成します。このプラグインは daemonSet を使用してすべてのノードにデプロイされます。

詳細は、「Using Device Manager」を参照してください。

2.3.4.5. Huge Page

テクノロジープレビューであった Huge Page は OpenShift Container Platform 3.10 で一般に利用できるようになりました。

メモリーは Page と呼ばれるブロックで管理されます。ほとんどのシステムでは 1 ページは 4Ki になります。メモリー 1Mi は 256 Page に、メモリー 1Gi は 256,000 Page に相当します。CPU には、ハードウェアのこれらの Page の一覧を管理する、組み込まれたメモリー管理ユニットがあります。Translation Lookaside Buffer (TLB) は、仮想から物理へのページマッピングの小規模なハードウェアキャッシュです。ハードウェアの指示で渡された仮想アドレスが TLB にあると、マッピングはすぐに決定されます。これがない場合には TLB ミスが発生し、システムは速度が遅い、ソフトウェアベースのアドレス変換にフォールバックし、パフォーマンスの問題が発生します。TLB のサイズは固定されているので、TLB ミスの発生率を減らすには Page サイズを大きくする必要があります。

Huge Page とは 4Ki を超えるメモリーページのことです。x86_64 アーキテクチャーでは、2Mi と 1Gi の 2 つが一般的な Huge Page のサイズになります。他のアーキテクチャーではサイズは異なります。Huge Page を使用するには、アプリケーションが認識できるようにコードを作成する必要があります。

Transparent Huge Page (THP) はアプリケーションの情報なしに Huge Page 管理の自動化を試みますが、それらには制限があります。とくにページサイズは 2Mi に制限されています。THP では、THP のデフラグの試行によりメモリー使用率の上昇や断片化が生じてパフォーマンスが低下し、メモリーページがロックされてしまう可能性があります。このような理由から、一部のアプリケーションは THP ではなく、事前に割り当て済みの Huge Page を使用するように設計されているか、この使用が推奨されている場合があります。

OpenShift Container Platform では、Pod のアプリケーションが事前に割り当てられた Huge Page を割り当て、消費することができます。

詳細は、「Managing Huge Pages」を参照してください。

2.3.5. ネットワーク

2.3.5.1. ルートアノテーションによる同時接続の制限

ルートアノテーション **`haproxy.router.openshift.io/pod-concurrent-connections`** は同時接続を制限します。

詳細は、「Route-specific Annotations」を参照してください。

2.3.5.2. Kubernetes Ingress オブジェクトのサポート

Kubernetes Ingress オブジェクトは受信接続が内部サービスに到達する方法を判別する設定オブジェクトです。OpenShift Container Platform では OpenShift Container Platform 3.10 より、Ingress コントローラー設定ファイルを使用したこれらのオブジェクトのサポートがあります。

詳細は、「Support for Kubernetes ingress objects」を参照してください。

2.3.5.3. IP フェイルオーバー管理が VIP アドレスの 254 グループに制限される

IP フェイルオーバー管理は VIP アドレスの 254 グループに制限されています。デフォルトでは、OpenShift Container Platform は各グループに 1 つの IP アドレスを割り当てます。**`virtual-ip-groups`** オプションを使用するとこれを変更でき、IP フェイルオーバーの設定時に複数の IP アドレスをそれぞれのグループに配置し、各 VRRPインスタンスに利用できる VIP グループの数を定義できます。

詳細は、「High Availability」を参照してください。

2.3.5.4. Egress ルーターの DNS 名を許可

Egress ルーターを、不安定な IP アドレスを使用し、ホスト名で外部サービスを参照できるように設定できます。

詳細は、「Deploying an Egress Router DNS Proxy Pod」を参照してください。

2.3.5.5. serviceNetwork の拡張

サービスネットワークアドレス範囲は、複数ノード環境で大規模なアドレス空間に拡張できます。これは異なる範囲への移行ではなく、既存の範囲を拡張することが関係します。

詳細は、「Expanding the Service Network」を参照してください。

2.3.5.6. OpenShift Container Platform と Red Hat OpenStack の Kuryr との統合を改善 (テクノロジープレビュー)

この機能は現在 テクノロジープレビュー として提供されており、実稼働環境のワークロードには適していません。

OpenShift Container Platform と Red Hat OpenStack の統合におけるベストプラクティスについては、「Kuryr SDN Administration」および「Configuring Kuryr SDN」を参照してください。

2.3.6. マスター

2.3.6.1. Descheduler (テクノロジープレビュー)

Descheduler は現在テクノロジープレビュー として提供されており、実稼働環境のワークロードには適していません。

Descheduler は適切でないノードから新規ノードへと Pod を移行します。Pod の移行は、以下をはじめとする様々な理由によって実行されます。

- 一部のノードの使用率が低くなっているか、または高くなっている。
- テイントまたはラベルがノードに追加またはノードから削除され、Pod/ノードのアフィニティーの要件が満たされなくなったため、元のスケジューリングにおける決定が当てはまらない。
- 一部のノードが失敗し、それらの Pod が別のノードに移行している。
- 新規ノードがクラスターに追加されている。

詳細は、「Descheduling」を参照してください。

2.3.6.2. Node Problem Detector (テクノロジープレビュー)

Node Problem Detector は現在テクノロジープレビュー として提供されており、実稼働環境のワークロードには適していません。

Node Problem Detector (ノード問題検出機能) は特定の問題を検出し、それらの問題を外部コントローラーがアクションを実行する外部 API サーバーに報告することで、ノードの正常性をモニターします。Node Problem Detector は、各ノードで daemonSet として実行されるデーモンです。このデーモンはクラスターに、ノードをスケジュール対象外とするノードレベルの障害を認識させようとします。Node Problem Detector を起動する際に、これに対して、検出する問題を配信するポートを指定します。Node Problem Detector により、データ収集を行うためにサブデーモンをロードすることができます。現時点は、3 つのサブデーモンを使用できます。問題デーモンによって検出される問題は、`NodeCondition` として分類できます。

問題デーモン:

- カーネルモニター: journald でカーネルをモニターし、正規表現パターンに基づいて問題を報告します。
- AbrtAdaptor: ノードでカーネルの問題、および journald でアプリケーションのクラッシュをモニターします。
- CustomerPluginMonitor: 条件の有無をテストできるようにし、条件が満たされない場合には `0` または `1` で終了します。

詳細は、「Node Problem Detector」を参照してください。

2.3.6.3. システムサービスが Pod でホストされる

システムサービスのそれぞれ、API、コントローラーおよび etcd はマスター上でシステムサービスとして実行されていました。これらのサービスはクラスター内の静的 Pod で実行されるようになりました。その結果として、これらのサービスを再起動するために、**`master-restart api`**、**`master-restart controllers`**、および **`master-restart etcd`** の新規コマンドを使用できるようになりました。これらのサービスのログ情報を表示するには、**`master-logs api api`**、**`master-logs controllers controllers`**、および **`master-logs etcd etcd`** を使用します。

詳細は、「重要な変更点」を参照してください。

2.3.6.4. 新規ノード設定プロセス

既存ノードの変更は、**node-config.yaml** ではなく設定マップを使用して実行できます。インストールは、**node-config-master**、**node-config-infra**、および **node-config-compute** の 3 つのノード設定グループを作成し、各グループの設定マップを作成します。同期 Pod はこれらの設定マップに対する変更の有無を監視します。変更が検出されると、同期 Pod はすべてのノードの **node-config.yaml** ファイルを更新します。

2.3.6.5. LDAP グループのプルーニング

グループのレコードを外部プロバイダーからプルーニングするために、管理者は以下のコマンドを実行できます。

```
$ oc adm prune groups --sync-config=path/to/sync/config [<options>]
```

詳細は、「Pruning groups」を参照してください。

2.3.6.6. Podman (テクノロジープレビュー)

Podman は現在テクノロジープレビューとして提供されており、実稼働環境のワークロードには適していません。

Podman は、OCI コンテナーおよび Pod を実行し、管理し、デバッグするためのデーモンなしの CLI/API です。以下の特徴があります。

- 高速かつ軽量である。
- runC を利用する。
- コンテナーで使用する構文を提供する。
- Varlink 経由のリモート管理 API がある。
- systemd 統合および高度な namespace の分離を行う。

詳細は、「Crictl Vs Podman」を参照してください。

2.3.7. メトリクスとロギング

2.3.7.1. Prometheus (テクノロジープレビュー)

Prometheus は依然としてテクノロジープレビューとして提供されており、実稼働環境のワークロードには適していません。Prometheus、AlertManager および AlertBuffer のバージョンは更新され、node-exporter も追加されています。

- prometheus 2.2.1
- Alertmanager 0.14.0
- AlertBuffer 0.2
- node_exporter 0.15.2

OpenShift Container Platform クラスターに Prometheus をデプロイし、Kubernetes およびインフラストラクチャーのメトリクスを収集して、アラートを取得できます。Prometheus web ダッシュボードでメトリクスおよびアラートを表示し、照会できます。または、独自の Grafana を用意して、Prometheus に連携させることも可能です。

詳細は、「Prometheus on OpenShift」を参照してください。

2.3.7.2. fluentd 向けの syslog 出力プラグイン (テクノロジープレビュー)

fluentd 向けの syslog 出力プラグインは現在テクノロジープレビュー として提供されており、実稼働環境のワークロードには適していません。

システムおよびコンテナーログは OpenShift Container Platform ノードから外部のエンドポイントに syslog プロトコルを使用して送信できます。fluentd syslog 出力プラグインはこの機能をサポートします。

重要

syslog 経由で送信されたログは暗号化されないので、安全ではありません。

詳細情報は、「Sending Logs to an External Syslog Server」を参照してください。

2.3.8. 開発者のエクスペリエンス

2.3.8.1. サービスカタログのコマンドラインインターフェース (CLI)

サービスカタログのコマンドインターフェース (CLI) を使用すると、コマンドラインでサービスをプロビジョニングし、バインドすることができます。コマンドの完全なセットを使用し、一覧表示、記述、プロビジョニング、プロビジョニング解除、バインド、およびバインド解除を実行できます。

`svcat` というサービスカタログ CLI ユーティリティーは、サービスカタログリソースとの対話を容易にするために使用できます。`svcat` は、OpenShift クラスターの集約された API エンドポイントを使用してサービス API と通信します。

詳細は、「Service catalog command-line interface (CLI)」を参照してください。

2.3.8.2. 新規 ignore-volume-az 設定オプション

新規設定オプションの `ignore-volume-az` は Red Hat OpenStack の **cloud.conf** ファイルで利用できるようになりました。これは、OpenShift Container Platform で永続ボリュームのゾーンと共にラベルが作成されないようにするために追加されました。OpenStack Cinder および OpenStack Nova には異なるトポロジーゾーンがあります。OpenShift Container Platform は Nova ゾーンと排他的に機能し、

Cinder トポロジーを無視します。そのためゾーンが Nova ゾーンと異なる場合があるため、Cinder ゾーン名のラベルを PV に設定しても意味がありません。この PV を使用する Pod は OpenShift Container Platform でスケジュール対象外となります。クラスター管理者は Cinder PV のラベリングをオフにし、それらの Pod をスケジュール対象とすることができます (**BZ#1500776**)。

2.3.8.3. CLI プラグイン (テクノロジープレビュー)

CLI プラグインは現在テクノロジープレビュー として提供されており、実稼働環境のワークロードには適していません。

通常、**プラグイン** または **バイナリー拡張** と呼ばれるこの機能は、利用可能なデフォルトの **oc** コマンドセットを拡張するので、新たなタスクの実行を可能にします。

CLI の拡張のインストールおよび作成方法に関する詳細は、「Extending the CLI」を参照してください。

2.3.8.4. Jenkins の更新

ビルドジョブの同期削除が有効になり、古くなったジョブのクリーンナップが可能になりました。

Jenkins は 2.107.3-1.1 に更新され、Jenkins ビルドエージェント (スレーブ) イメージが更新されました。

- Node.js 8
- Maven 3.5

以下のイメージは OpenShift Container Platform 3.10 で非推奨となりました。

```
jenkins-slave-maven-*
jenkins-slave-nodejs-*
```

アプリケーションを新しいイメージに移行できるまで、これらのイメージはその間そのまま存在します。

```
jenkins-agent-maven-*
jenkins-agent-nodejs-*
```

詳細は、「Jenkins Agents」を参照してください。

2.3.9. レジストリー

2.3.9.1. OpenShift 認証によるレジストリーメトリクスの公開

OpenShift Container Platform 3.10 レジストリーメトリクスエンドポイントはビルトインの OpenShift Container Platform 認証で保護されるようになりました。ClusterRole を使用してレジストリーメトリクスにアクセスできます。

詳細は、「Accessing Registry Metrics」を参照してください。

2.3.10. Web コンソール

2.3.10.1. サービスカタログ検索の強化

サービスカタログ UI の検索アルゴリズムが強化されました。重み付けは一致が検出される場所に基づいて行われ、ファクターにはタイトル、説明、およびタグ付けが含まれます。

2.3.10.2. アプリケーションのルートの表示および選択方法の改善

アプリケーションのルートを表示し、選択する方法が改善されました。複数のルートが利用可能であることを示す機能を利用できるようになりました。プライマリーとして使用するルートにはアノテーションを付けます。

```
console.alpha.openshift.io/overview-app-route: ‘true’
```

2.3.10.3. 汎用シークレットの作成

Web コンソールで汎用シークレット (キー/値のペアを持つシークレット) を作成できます。すでにシークレットを作成している場合でも、不透明なシークレットを作成できます。この動作は ConfigMaps を作成する動作に似ています。

2.3.10.4. その他の変更点

- Pod 端末の **xterm.js** の依存関係を更新する際のパフォーマンスが大幅に改善されています。
- イメージプルのシークレットをイメージのデプロイ (deploy image) ダイアログから直接作成できるようになりました。

2.3.11. セキュリティー

2.3.11.1. etcd での TLS 暗号化スイートの指定

TLS 暗号化スイートを etcd で使用するように設定し、セキュリティーポリシーを満たせるようにできます。

詳細は、「Specifying TLS ciphers for etcd」を参照してください。

2.3.11.2. コンテナー間での PID Namespace 共有の制御 (テクノロジープレビュー)

コンテナー間での PID Namespace 共有の制御は、現在テクノロジープレビュー として提供されており、実稼働環境のワークロードには適していません。

この機能を使用して、ログハンドラーサイドカーコンテナーなどの Pod 内の関連コンテナーを設定したり、シェルのようなデバッグユーティリティーを含まないコンテナーイメージのトラブルシューティングを行ったりできます。

- 機能ゲート **PodShareProcessNamespace** はデフォルトで **false** に設定されます。
- API サーバー、コントローラーおよび kubelet に **feature-gates=PodShareProcessNamespace=true** を設定します。
- API サーバー、コントローラー、およびノードサービスを再起動します。
- Pod を **shareProcessNamespace: true** を指定して作成します。
- **oc create -f <pod spec file>** を実行します。

注意事項

PID namespace をコンテナー間で共有する際に、以下の点に注意してください。

- サイドカーコンテナーは分離できない。
- 環境変数はその他すべてのプロセスに表示されます。
- プロセス内で使用されるすべての **kill all** セマンティクスが機能しなくなります。
- 他のコンテナーからのすべての **exec** プロセスが表示されます。

詳細は、「Expanding Persistent Volumes」を参照してください。

2.3.11.3. ルーターサービスアカウントでサービスへのアクセスが不要になる

ルーターサービスアカウントではすべてのシークレット読み取るパーミッションが不要になります。これにより、セキュリティーが強化されます。以前のバージョンでは、ルーターのセキュリティーが危険にさらされると、クラスター内のすべての機密データを読み取る可能性がありました。

現時点では ingress オブジェクトの作成時に、対応するルートオブジェクトが作成されます。ingress オブジェクトが変更される場合、変更されるシークレットがすぐに有効になります。ingress オブジェクトが削除される場合、そのために作成されたルートは削除されます。

2.3.12. ドキュメンテーション

2.3.12.1. クイックインストールを削除

OpenShift Container Platform 3.10 では、クイックインストール方式およびこれに対応するドキュメントが削除されています。

2.3.12.2. 手動アップグレードを削除

OpenShift Container Platform 3.10 では、手動アップグレード方法およびこれに対応するドキュメントが削除されています。

2.3.12.3. 『インストールと設定』ガイドが分割される

読みやすさを強化するために、『インストールと設定』ガイドが『クラスターのインストール』および『クラスターの設定』に分割されました。

2.4. 主な技術上の変更点

OpenShift Container Platform 3.10 では、主に以下のような技術的な変更点が加えられています。

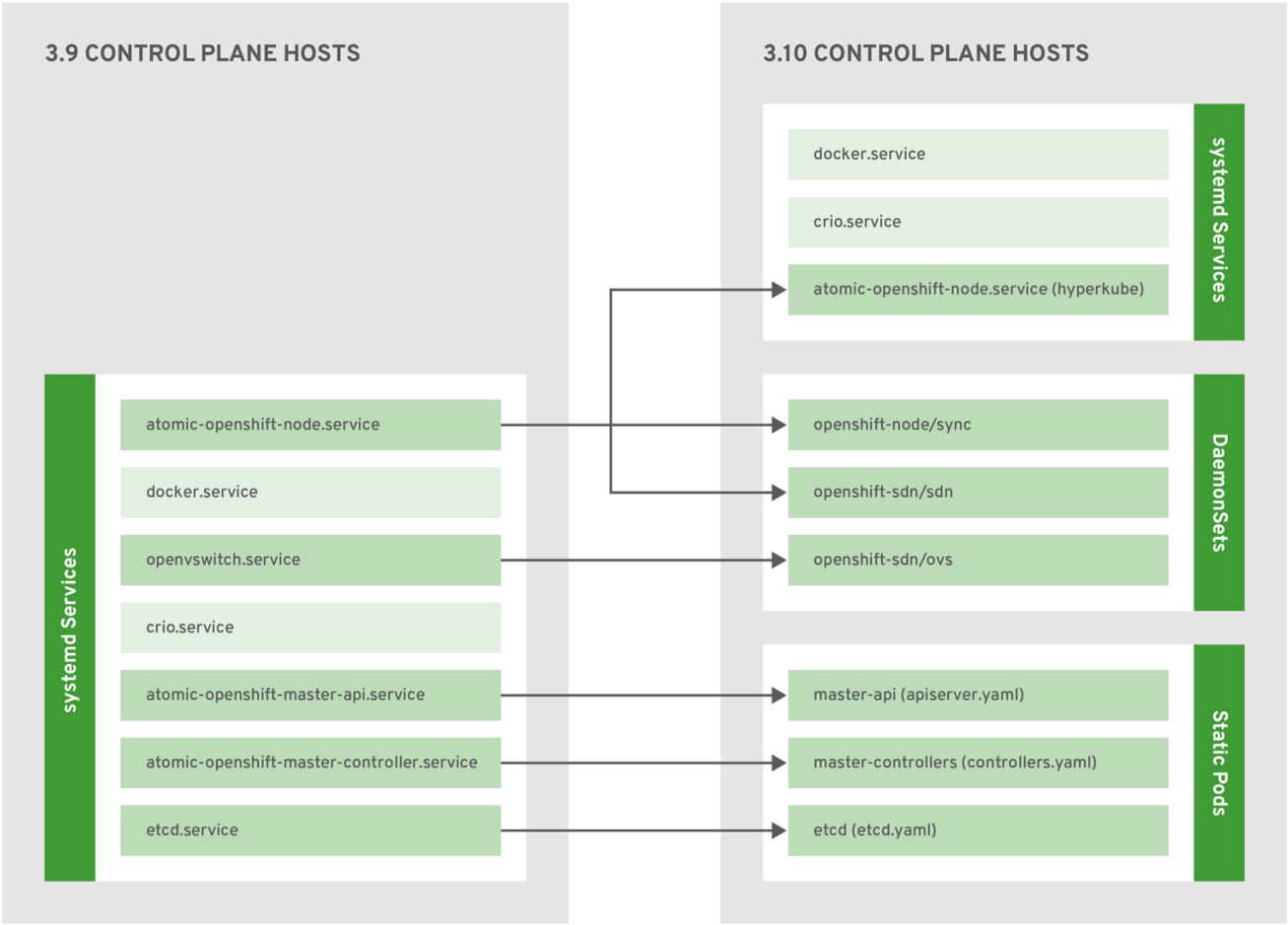
クラスターアーキテクチャーに対する主要な変更点
OpenShift Container Platform 3.10 では、コントロールプレーンとノードコンポーネントのデプロイ方法についてのアーキテクチャー上の主要な変更が導入されました。OpenShift Container Platform 3.9 からの新規インストールおよびアップグレードがこの影響を受けます。

以下のセクションでは、最も大きな変更点について重点的に扱います。詳細については、『Architecture Guide』で説明されています。

静的 Pod としてのコントロールプレーン
以前のバージョンでは、**systemd** サービスまたはシステムコンテナーとして実行されていたコントロールプレーンの各種コンポーネント (apiserver、コントローラーおよびマスターと同じ場所に置かれ

ている場合の etcd) は、マスターホストの kubelet によって静的 Pod として実行されるようになりました。ノードコンポーネントの **openshift-sdn** および **openvswitch** は、**systemd** サービスの代わりに DaemonSet を使用して実行されるようになりました。

図2.1 コントロールパネルのホストアーキテクチャーにおける変更点



OPENSHIFT_473421_0718

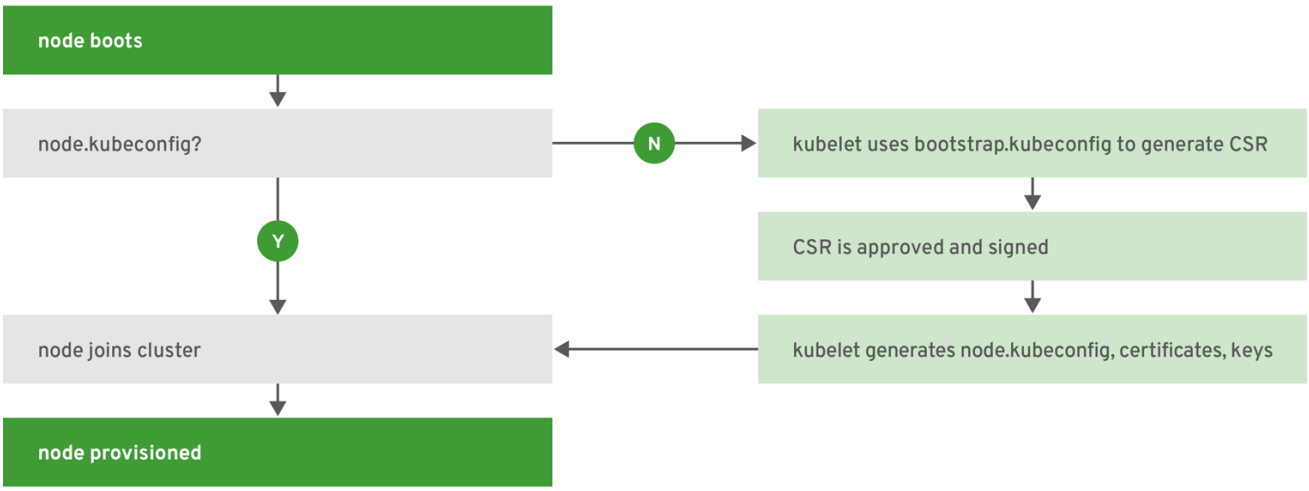
現時点で、これらのコンポーネントを実行する方法としてサポートされているのは上記の方法のみになります。システムコンテナーは (kublet なしで) サポートされなくなりました。ただし、ノードサービスの RHEL Atomic Host はこの例外となります。アップグレードにより新規アーキテクチャーへの変更が自動的に導入されることになります。コントロールプレーンの各種コンポーネントは引き続き、**/etc/origin/master/** および **/etc/etcd/** ディレクトリーから設定を読み取ります。

コントロールプレーンの起動後に **`oc get pods -n kube-system`** を使用して Pod を表示でき、通常の **oc** CLI コマンドを使用してこれらに対して **`exec`**、**`log`**、**`inspect`** を実行できます。

変更の理由
静的 Pod は特定のノード上の kubelet デーモンによって直接管理され、API サーバーはこれを監視する必要がありません。この単純なアーキテクチャーでは、マスターおよびノードの静的 Pod にはレプリケーションコントローラーが関連付けられておらず、kubelet デーモン自体がクラッシュの発生時にそれらを監視し、再起動します。静的 Pod は常に 1 つの kubelet デーモンにバインドされ、常にそのデーモンが設定された同じノードで実行されます。

ノードのマスターからのブートストラップ
ノードはデフォルトでマスターからブートストラップされるようになりました。これは、ノードがマスターから事前に定義された設定、クライアントおよびサーバー証明書をプルすることを意味します。3.10 アップグレードにより、お使いのノードがこの新規モードを使用するように自動的に変換されます。

図2.2 ノードブートストラップのワークフローの概要



変更の理由
ブートストラップの目的は、ノード間の差異を減らし、より多くの設定を集中管理し、クラスターを望ましい状態で接続することでノードの起動を高速化することにあります。これにより、デフォルトで証明書のローテーションおよび証明書の一元管理が可能になります (保留中の証明書を表示するには、**`oc get csr`** を使用します)。

コンテナー化インストール方法を削除
OpenShift Container Platform のコンテナー化インストール方法 (コンポーネントが標準コンテナーイメージとして実行される) は削除され、3.10 以降でサポートされなくなりました。3.10 アップグレードでは RHEL サーバーホストを RPM ベースの方法に移行し、RHEL Atomic Host をシステムコンテナーベースの方法 (ノードサービスの場合のみ) に移行します。これらは RHEL バリアントについてサポートされる唯一の方法になります。OpenShift Container Platform 3.9 から 3.10 にアップグレードし、スタンドアロンの etcd が RHEL 上でコンテナー化として実行されている場合、そのインストールはアップグレード後もコンテナー化されたままになります。

変更の理由
これにより、インストールおよびアップグレードパスの数を削減でき、今後のリリースに導入される各種機能との連携が強化されます。

設定ファイル
OpenShift Container Platform 3.9 から 3.10 にアップグレードするには、以前のマスターおよびノード設定を新規の ConfigMap の使用にマップする設定ファイルを最初に作成し、クラスターアップグレードの実行時にそのマッピングを指定する必要があります。これにより、アップグレードが重要な情報なしに開始されることを防ぎ、アップグレード後もホストが以前の形式のデプロイメントを使用しているという状態を避けることができます。

さらに、**/etc/sysconfig/(origin|atomic-openshift)-(api|controllers)** ファイルは使用されなくなりました。新規の **/etc/origin/master/master.env** ファイルを静的 Pod の環境変数を設定するために使用できます。環境変数として利用可能な設定のセットは制限されています (プロキシーおよびログレベル)。OpenShift Container Platform の今後のバージョンでは、コントロールプレーンファイルを優先的に使用するためにこの設定を削除するため、**master.env** ファイルの使用を最後の手段および非推奨とみなしてください。

静的 **Pod** イメージへの更新
以下のイメージが削除されました。

```
openshift3/ose-*
openshift3/container-engine-*
```

```
openshift3/node-*
openshift3/openvswitch-*
```

これらのイメージは以下に置き換わります。

```
openshift3/ose-node-*
openshift3/ose-control-plane-*
```

イメージ **openshift3/metrics-schema-installer-container** も追加されました。

イメージ **openshift3/ose-sti-builder** は既存の **openshift3/ose-docker-builder** に置き換わります。

詳細は、「Syncing Images」を参照してください。

oc port-forward の Pod フラグを削除

oc port-forward の非推奨となっていた **-p <POD>** フラグが削除されました。代わりに **oc port-forward pod/<POD>** を使用してください。

API プレフィックスを使用しない API グループおよびバージョンの指定

kubernetesMasterConfig.apiServerArguments の **--runtime-config** フラグで APIグループを有効または無効にする場合、API プレフィックスなしで **<group>/<version>** を指定します。今後のリリースでは、API プレフィックスの使用は許可されなくなります。以下は例になります。

```
kubernetesMasterConfig:
  apiServerArguments:
    runtime-config:
    - apps.k8s.io/v1beta1=false
    - apps.k8s.io/v1beta2=false
...
```

-o name の出力に API グループが含まれる

-o name の出力に API グループおよび単一の kind が含まれるようになりました。以下は例になります。

```
$ oc get imagestream/my-image-stream -o name
imagestream.image.openshift.io/my-image-stream
```

Internet Explorer 11 の非推奨の Web コンソールサポート

Internet Explorer (IE) 11 の Web コンソールのサポートが非推奨となりました。これは、今後のバージョンの OpenShift Container Platform で削除されます。Microsoft Edge は現在でもサポートされているブラウザーです。

ローカルプロビジョナー設定の変更点

新規デバイスの追加は半自動で実行されます。プロビジョナーは設定されたディレクトリーで新規マウントについて定期的にチェックします。管理者はそこに新規ディレクトリーを作成し、デバイスをマウントし、SELinux ラベルを適用して Pod がデバイスを使用できるようにする必要があります。

詳細は、「Configuring for Local Volume」を参照してください。

OpenStack 設定の更新

Red Hat OpenStack クラウドプロバイダーの設定時に、ノードのホスト名が OpenStack のインスタンス名に一致しており、登録されている名前が DNS-1123 仕様に準拠していることを確認する必要があります。

非推奨の **openshift-namespace** フラグを削除
非推奨となっていた **openshift-namespace** フラグが **oc adm create-bootstrap-policy-file** コマンドから削除されました。

openshift_set_node_ip and openshift_ip の使用がサポート対象外となる
OpenShift Container Platform 3.10 では、**openshift_set_node_ip** および **openshift_ip** の使用がサポートされなくなりました。

dnsIP が設定不可能になる
以前は **openshift_dns_ip** で設定されていたノードの **dnsIP** 値を設定することができなくなりました。

非推奨の **openshift_hostname** 変数
openshift_hostname 変数が削除されました。

非推奨の **openshift_docker_additional_registries** の使用
openshift_docker_additional_registries を使用しない、またはこれに依存しないようにしてください。

システムコンポーネント用に予約される **openshift-infra**
openshift-infra namespace はシステムコンポーネント用に予約されます。これは、Kubernetesリソース用の OpenShift Container Platform 受付プラグインを実行しません。SCC 受付は **openshift-infra** namespace の Pod について実行されません。これにより、Pod が Persistent Volume Claim (永続ボリューム要求) を使用し、SCC で割り当てられた **uid**/**fsGroup**/**supplementalGroup**/**seLinux** 設定を使用している場合などには Pod が失敗する可能性があります。

oc edit での **Kube_EDITOR** の使用
oc edit コマンドでは **KUBE_EDITOR** が使用されるようになりました。**OC_EDITOR** サポートは今後のリリースで削除されるため、**KUBE_EDITOR** に切り換えることをお勧めします。

batch/v2alpha1 API バージョンのデフォルトでの提供が終了
batch/v2alpha1 API バージョンはデフォルトで提供されなくなりました。必要な場合は、これを以下の設定を使い、**master-config.yaml** ファイルで再度有効にすることができます。

```
kubernetesMasterConfig:
  apiServerArguments:
    ...
    runtime-config:
    - apis/batch/v2alpha1=true
```

新規の **openshift_additional_ca** オプション
ロードバランサーを含むファイルを参照する OpenShift Ansible インストーラーの新規オプション **openshift_additional_ca** を使用できるようになりました。マスターノード用にインストーラーで生成されるものとは異なる CA を要求するロードバランサーをクラスターで使用している場合、ユーザーはこの追加の CA 証明書を **/etc/origin/master/ca-bundle.crt** ファイルに追加する必要があります。これにより、この CA 証明書がクラスター内の Pod で利用できるようになります。

namespace スコープの要求
subjectaccessreviews.authorization.openshift.io および **resourceaccessreviews.authorization.openshift.io** は、今後のリリースではクラスタースコープの場合にのみ使用されます。namespace スコープの要求が必要な場合には、**localsubjectaccessreviews.authorization.openshift.io** および **localresourceaccessreviews.authorization.openshift.io** を使用します。

デフォルトのイメージストリームによるプルスルーの使用

デフォルトのイメージストリームはプルスルーを使用できるようになりました。これは、内部レジストリーがユーザーの代わりにイメージをプルできるという意味です。イメージストリームのイメージのアップストリームの場所を変更する場合、レジストリーはその場所からプルします。これは、レジストリーがアップストリームの場所を信頼できる必要があることを意味しています。アップストリームの場所が標準的なシステム信頼ストアの一部ではない場合、プルは失敗します。適切な証明書を指定するために適切な信頼ストアを docker-registry Pod にマウントする必要があります (この場合は **/etc/tls** ディレクトリーパスを使用)。

イメージのインポートプロセスは Pod (apiserver Pod) 内で実行されるようになりました。イメージのインポートではインポート元のレジストリーを信頼する必要があります。ソースレジストリーが標準的なシステムストアにある CA で署名されていない証明書を使用する場合、適切な信頼ストア情報を apiserver Pod に提供する必要があります。これは、内容を Pod の **/etc/tls** ディレクトリーにマウントして実行できます。

サーバーへの接続を防ぐためのローカルフラグの使用

今後のリリースでは、**oc** コマンドをローカルファイルに対して起動する場合、クライアントがサーバーに接続できないようにするには **`--local`** フラグを使用する必要があります。

非推奨の **GitLab** バージョン

v11.1.0 よりも前のバージョンで GitLの自己ホスト型のバージョンの使用が非推奨となりました。自己ホスト型のバージョンを使用している場合は、すぐに GitLab インストールをアップグレードする必要があります。gitlab.com のホスト型バージョンが使用される場合は、最新バージョンが常に環境で実行されるためアクションは不要になります。

Flexvolume プラグインの更新

`attach`/**`detach`** を実行するための flexvolume を使用している場合、flex バイナリーでは外部の依存関係を使用できず、自己完結型の機能として使用される必要があります。Atomic Host の Flexvolume プラグインのパスは **/etc/origin/kubelet-plugins** に変更されており、これはマスターおよびコンピュートノードの両方に適用されます。

非推奨の **oc rollout latest ... --output=revision**

OpenShift Container Platform 3.10 では、**`oc rollout latest … --output=revision`** が非推奨となりました。代わりに **`oc rollout latest … --output jsonpath={.status.latestVersion}`** または **`oc rollout latest … --output go-template={{.status.latestVersion}`** を使用してください。

CNS の名称が **Red Hat OpenShift Container Storage (RHOCS)** になる

Container Native Storage (CNS) の名称が Red Hat OpenShift Container Storage (RHOCS) になりました。以前のバージョンでは、CNS と CRS という混乱を招きかねない用語に使用されていました。

ビルダーイメージの置き換え

OpenShift Container Platform 3.10 では、**Atomic OpenShift Docker Builder** の**`registry.access.redhat.com/openshift3/ose-docker-builder`** が **Atomic OpenShift S2I Builder** の **`registry.access.redhat.com/openshift3/ose-sti-builder`** に置き換わりました。

以前のバージョンでは、**Atomic OpenShift Docker Builder** は Docker イメージビルドを実行していましたが、Source-to-Image (S2I) イメージビルドも実行できるようになりました。

2.5. バグ修正

今回のリリースでは、以下のコンポーネントのバグが修正されました。

ビルド

- 一部のビルドコンテナー環境変数は、コンテナーログで編集される場合に変更されていました。その結果、URL プロキシー設定 (HTTP/S プロキシーなど) が変更され、これらの設定が壊れてしまう可能性がありました。これらの環境変数のコピーは、ログの編集前に作成されるようになりました (BZ#11571349)。

- ビルドログのストリームは、ビルド Pod が起動するのを待機するサーバー側のタイムアウトのために失敗していました。そのため、**`oc start-build`** は、**`--wait`** および **`--follow`** フラグが設定されている場合にハングする可能性がありました。このバグ修正により、以下が実行されるようになりました。

  - ビルド Pod が起動するためのサーバー側のタイムアウトは 10 秒から 30 秒に引き上げられました。

  - **`--follow`** フラグが指定され、ログストリームが失敗した場合、エラーメッセージがユーザーに返されます。

  - **`--follow`** および **`--wait`** が指定される場合、ログストリームを再試行します。
    上記の結果として、以下のようになります。

  - ビルド Pod による待機のタイムアウトによるログストリームの失敗が生じる可能性は低くなりました。

  - **`--follow`** が失敗する場合、ユーザーにはメッセージ **Failed to stream the build logs - to view the logs, run oc logs build/<build-name>** が表示されます。

  - **`--follow`** および **`--wait`** フラグが設定される場合、**`oc start-build`** は成功するまでビルドログのフェッチを再試行します。
    (BZ#1575990)

- **openshift jenkins sync** プラグインで保持されてきたビルドの監視は、他の API オブジェクトタイプの監視が依然として機能する場合でも機能しなくなることがありました。ビルドの検出には、デフォルトで 5 分間隔で実行されるバックグラウンドビルド一覧のスレッドが使用されました。今回のバグ修正により、**openshift jenkins sync** プラグインの監視が予期せずに終了する場合時により適切なロギングを追加でき、終了する場合の再接続機能や、一覧表示の間隔を設定できる機能が追加されました。これで、お客様は Pipleline ストラテジーのビルドの起動のために 5 分待機する必要がなくなりました (BZ#1554902)。

- ビルドコントローラーは、時刻が複数のマスター間で正確に同期されていない場合に誤って失敗したビルドの影響を受けていました。コントローラーのロジックが強化されることにより、コントローラーが時刻の同期に依存しなくなりました (BZ#1547551)。

- Webhook ペイロードにはコミットの空の配列が含まれている可能性があり、API サーバーで処理される場合に配列のインデックスエラーが発生する可能性がありました。この場合、API サーバーはクラッシュします。インデックスの試行前に、空のアレイがないかどうかを確認してください。今回のバグ修正では、API サーバーがクラッシュすることなく、空のコミットのペイロードを処理できるようになりました (BZ#1585663)。

コンテナー

- Docker エンジンの無効な SELinux コンテキストにより **`docker exec`** が機能できませんでした。今回のバグ修正により、この問題は解決されています (BZ#1517212)。

イメージ

- Jenkins は **`BEGIN CERTIFICATE`** 行の前にある **Bag Attributes** により証明書の解析に失敗しており、**openshift jenkins** イメージがこれらの証明書を Kubernetes クラウド設定に追加するために起動に失敗しました。今回のバグ修正で、Pod にマウントされる証明書の **`BEGIN`**

CERTIFICATE 行の前にあった **Bag Attributes** が削除され、通常は証明書の形式が適切であるかどうかが検証されます。Jenkins はこれらの証明書が導入されている場合でも起動できるようになりました (BZ#1548619)。

- **Reference** フィールドの新規の値は変更とはみなされませんでした。そのため、ステータスフィールドは更新されませんでした。今回のバグ修正により、変更の検出が更新されるようになりました。**Reference: true** を設定し、イメージストリームタグでイメージ参照を取得できるようになりました (BZ#1555149)。

- 証明書名についての追加の制約により、有効な証明書の処理が妨げられていました。これにより「tls: failed to parse certificate from server: x509: unhandled critical extension」のエラーが生じました。その結果、有効な証明書が使用不可となっていました。この制約について修正されている新規の golang ライブラリーへの移行により、以前に失敗していた証明書が使用できるようになりました (BZ#1518583)。

- 以前のバージョンでは、PhantomJS は **jenkins-slave-base-rhel7** イメージではインストールできませんでした。PhantomJS が **tar.bz2** アーカイブとしてパッケージ化されており、**jenkins-slave-base-rhel7** には bzip2 バイナリーが含まれていなかったためです。OpenShift Container Platform バージョン 3.10 には、bzip2 バイナリーを含む新規の Jenkins イメージが含まれるようになりました (BZ#1544693)。

インストーラー

- 以前のバージョンの互換性の問題により、**networkPluginName** エントリーは **node-config.yaml** に 2 回一覧表示されました。重複したエントリーは不要になり、削除されました (BZ#1567970)。

- インストーラーの変更により、デフォルト以外のレジストリーからイメージを使用している場合は、すべてのコンポーネントおよびイメージについて、**/etc/ansible/hosts** ファイルの **oreg_url** パラメーターを使用してレジストリーを設定する必要があります。以前のバージョンでは、**oreg_url**、**openshift_docker_additional_registries**、および **openshift_docker_insecure_registries** パラメーターを設定する必要がありました (BZ#1516534)。

- Azure クラウドプロバイダーが有効にされている環境では、Azure ストレージで使用するデフォルトのストレージクラスをプロビジョニングできるようになりました (BZ#1537479)。

- **openshift-ansible-serivce-broker** プロジェクトがない場合は、Ansible Playbook を使用してサービスカタログをアンインストールできるようになりました。以前のバージョンでは、プロジェクトがない場合に Playbook のアンインストールは失敗していました (BZ#1561485)。

- NFS ストレージは OpenShift レジストリー、ロギング、およびメトリクスコンポーネントで必要なファイルシステム機能を提供しないため、これらのコンポーネントに NFS ストレージを許可しないチェックがインストーラーに追加されました。これらのコンポーネントに NFS ストレージを使用するには、クラスター変数の **openshift_enable_unsupported_configurations** を **true** に設定してオプトインを選択する必要があります。レジストリー、メトリクス、およびロギングコンポーネント用に NFS ストレージを使用することは、概念実証用の環境でのみサポートされており、実稼働環境ではサポートされていません (BZ#1416639)。

- Ansible Playbook の実行には長い時間がかかり、実行されているタスクと関連性のないホストから証明書のエラーが発生していました。Playbook は関連性のあるホストのみをチェックするように変更されています (BZ#1516526)。

- Ansible インストーラー Playbook はストレージクラスの作成前に永続ボリュームを作成したため、Playbook が 2 回実行されていました。Playbook は永続ボリュームが作成される前にストレージクラスを作成するように変更されました (**BZ#1564170**)

- OpenShift のプレフィックスおよびバージョンがコンソールに設定される方法により、コンソールで報告されるバージョンは他のコンポーネントで表示されるバージョンとは異なっていました。コントロールプレーンのアップグレードにより、コンソールのバージョンが他のコントロールプレーンのコンポーネントのバージョンと一致するようになりました (**BZ#1540427**)。

- Ansible インストール Playbook では、PVC を作成するためにインストール後にストレージクラスを手動で設定する必要があります。インベントリーファイルに以下のパラメーターを設定してストレージクラスを設定できるようになりました。

```
openshift_storageclass_name=test-1
openshift_storageclass_provisioner=rbd
openshift_storageclass_parameters={'fstype': 'ext4', 'iopsPerGB':
'10', 'foo': 'bar'}
```

  (**BZ#1471718**)

- 証明書の有効期限切れ Playbook **easy-mode.yaml** は、期限切れの情報についてすべての証明書ファイルをチェックしていませんでした。その結果として、期限切れのファイルは検出されず、エラーが発生する可能性がありました。Ansible Playbook の更新が行われています (**BZ#1520971**)。

- 以前のバージョンでは、dnsmasq は、ノードサービスがその DNS サービスを実行する **`127.0.0.1:53`** にバインドされることを避けるために特定の IP アドレスでリッスンするように設定されました。今回の更新により、dnsmasq は loopback を除くすべてのインターフェースにバインドするように設定され、これにより dnsmasq がホストの複数インターフェースと適切に機能するようになりました (**BZ#1481366**)。

- ルーターまたはレジストリーの **`registryurl`** 変数は、最初の **`master registry_url`** 値以外の値に設定しなければならない場合がまれにあります。今回の修正により、**`openshift_hosted_router_registryurl`** および **`openshift_hosted_registry_registryurl`** 変数をインベントリーに設定できるようになりました (**BZ#1509853**)。

- SELinux ポリシーにおける最近の変更により、CFME を含む systemd で Pod を実行する際に追加の SEBoolean を設定する必要があります(**BZ#1587825**)。

ロギング

- **`kube-`** および **`openshift-`** プレフィックスは内部のユースケース用に予約されます。名前の競合を避けるために、予約されたプレフィックスをデフォルトのロギングプロジェクトとして使用することが適切です。今回の修正では、予約されたプレフィックスをデフォルトのロギングプロジェクトとして使用します。これは他のインフラストラクチャーアプリケーションで使用されるパターンに適しており、インフラストラクチャーが既知のパターンで namespace にデプロイされていることを前提とする他のサービスと共に EFK スタックの参加を可能にします (**`openshift-`** など)。(**BZ#1535300**)。

- ユーティリティー **logging-dump.sh** は ElasticSearch ログをトラブルシューティングに役立つ情報の一部としてダンプします。OpenShift Container Platform 3.10 では、ElasticSearch のログの場所は **/elasticsearch/logging-es[-ops]/logs** から **/elasticsearch/persistent/logging-es[-ops]/logs** に移行しました。**logging-dump.sh** は ElasticSearch ログのダンプ時に **Unable to get ES logs from pod <ES_POD_NAME>** のエラーを出して失敗しまし

た。**/elasticsearch/logging-es[-ops]/logs** のほかにも、新規パス **/elasticsearch/persistent/logging-es[-ops]/logs** でログファイルをチェックしてください。今回のバグ修正により、**logging-dump.sh** は ElasticSearch ログを正常にダンプできるようになりました (**BZ#1588416**)。

Web コンソール

- 以前のバージョンでは、Pod が準備できるまでに 5 分を超える時間がかかる場合、デプロイメント設定で指定される **`timeoutSeconds`** の値を問わず、Web コンソールが警告を出していました。一部のアプリケーションの場合、この間隔は短すぎるものでした。今回の修正により、この警告は Web コンソールから削除されました (**BZ#1550138**)。

- このリリースの前のバージョンでは、Web コンソールコンテナー端末のコピーアンドペースト操作は Firefox および Internet Explorer で適切に機能しませんでした。今回の修正により、**`xterm.js`** は **`v3.1.0`** に更新されています。コピーアンドペーストはコンテキストメニューまたはキーボードのショートカットを使用して実行できるようになりました (**BZ#1278733**)。

- 「No results match」の結果がコンソールまたはカタログページに表示される際に、検索キーをクリアするの「Clear Filters」および「Clear All Filters」の 2 つのリンクがありました。今回の修正により、「Clear Filters」のすべての出現箇所が「Clear All Filters」に変更され、フィルターをクリアする 1 つのオプションが用意されました (**BZ#1549450**)。

- 複数の異なる BuildConfig Webhook URL が CLI および Web コンソールで取得されていました。これにより、CLI は正しい **build.openshift.io** API グループを使用するものの、Web コンソールは API グループを使用しませんでした。今回の修正により、Webhook フィルターが Web コンソールの正しい **build.openshift.io** API グループを使用するように更新され、その結果として BuildConfig Webhook の正しい URL が指定されるようになりました (**BZ#1559325**)。

- **`/console/project/pro1/browse/images/non-existent-image`** などの存在しないイメージの URL を手動で入力すると、プロセスが終了し、「The image stream details could not be loaded」というアラートが表示される場合でもロード画面がフリーズしました。今回の修正により、ロードされたスコープ変数が、イメージが読み込まれているかどうかにかかわらず設定され、ロード画面を非表示にするビューで使用されるようになりました。結果として、イメージデータのロードの試行後、画面はロード時にフリーズされなくなりました (**BZ#1550797**)。

- 以前のバージョンでは、Web コンソールは、**Deploy Image** ページでのプライベートリポジトリーイメージを使用したアプリケーションのデプロイをサポートしませんでした。これについては修正され、ユーザーはプライベートリポジトリーイメージでアプリをデプロイできるようになりました (**BZ#1489374**)。

マスター

- 以前のバージョンでは、DaemonSet ノードはプロジェクトのデフォルトノードセレクターで制限され、これにより DaemonSet Pod の作成および削除がそれらのノード上でループに陥る可能性がありました。今回の修正により、アップストリームの DaemonSet ロジックがプロジェクトのデフォルトノードセレクターを認識できるようにパッチが適用されました。結果として、プロジェクトのデフォルトのノードセレクターで制限されていたノード上の DaemonSet Pod の作成および削除ループが解決されました (**BZ#1501514**)。

- 以前のバージョンでは、クライアントは完全検出を読み取ることができず、一時的に利用不可能となる最初の集計サーバーで停止しました。これにより、利用可能なすべてのリソースについての適切な情報を取得することができませんでした。今回の修正により、検出アクションについてのデフォルトのタイムアウトが導入されました。その結果、集計サーバーで失敗が生じた場合に、クライアントは引き続き他のサーバーのリソースを検出し、ユーザーは利用可能なリソースを使用できるようになりました (**BZ#1525014**)。

- 以前のバージョンでは、再作成ストラテジーで DeploymentConfigs を使用した Pod がエビクトされる場合に、新規 Pod はタイムアウトの期間が経過するまでオンラインになりませんでした。再作成ストラテジーはエビクトされた Pod が存在している場合でも新規 Pod を作成できるようになりました (BZ#1549931)。

メトリクス

- 以前のバージョンでは、**auto_snapshot** パラメーターは **cassandra.yaml** ファイルで **true** に設定され、OpenShift Container Platform 3.7 で導入された Hawkular Metrics への変更によって数多くのスナップショットが生成され、ディスクが一杯になる可能性が生じました。**auto_snapshot** がデフォルトで無効にされるようになったため、スナップショットは、Ansible インベントリーファイルで **openshift_metrics_cassandra_take_snapshot** プロパティーを**true** に設定する場合にのみ生成されるようになりました (BZ#1567251)。

- 以前のバージョンでは、複数の CA 証明書をクラスターの Pod に分散することはできませんでした。この制限により、マスターノード用にインストーラーで生成されたものとは異なる CA 証明書を必要とするロードバランサーの設定に関連する問題がありました。今回の修正により、インストール時に **openshift_additional_ca** パラメーターでロードバランサーの証明書の場所を定義できるようになりました。証明書は **/etc/origin/master/ca-bundle.crt** ファイルに追加され、これはクラスター内の Pod で利用可能になります (BZ#1535585)。

- バージョン 3.9 では、Prometheus サービスアカウントには、ルーターのメトリクスエンドポイントにアクセスするために必要なパーミッションがありませんでした。そのため、Prometheus はルーターのメトリクスを取得できませんでした。Prometheus サービスアカウントにはメトリクスエンドポイントにアクセスするためのロールが追加され、ルーターからメトリクスを取得できるようになりました (BZ#1565095)。

ネットワーク

- 以前のバージョンでは、サービスコントローラーはサービスの作成時に要求をクラウドプロバイダーに常に送信しました。この要求は、クラウドプロバイダーが、LoadBalancer 以外のサービスについてもサービスのロードバランサーがあるかどうかを検査しました。多くのサービスが作成されるクラスターでは、要求の追加により一部のクラウドプロバイダー API が大幅に使用されました。LoadBalancer 以外のサービスが作成される場合には、サービスコントローラーはこの要求をクラウドプロバイダーに送信しなくなり、クラウドプロバイダー API の使用は軽減されます (BZ#1571940)。

- 以前のバージョンでは、egress ルーター設定は egress ルーター Pod が、それらをホストするノードのパブリック IP アドレスに接続することを防いでいました。egress Pod がそのノードを **/etc/resolv.conf** ファイルのネームサーバーとして使用するように設定されている場合、DNS 解決は失敗しました。しかし、egress ルーター Pod からそのノードへのトラフィックは egress インターフェースではなく SDN トンネル経由でルーティングされるようになりました。egress ルーターはそれらのノードの IP に接続できるようになり、egress ルーター DNS は機能します (BZ#1552738)。

- 2 つのノードを再起動する際にそれらが IP アドレスをスワップする場合、他のノードはそれらのノードのいずれかまたは両方の Pod にトラフィックを送信できないことがありました。今回のバージョンでは、OVS フローがノード IP アドレスの再割り当てを適切に管理し、ノードが IP アドレスをスワップする場合でも Pod 間のトラフィックが継続されるようになりました (BZ#1538220)。

- 以前のバージョンでは、既存の EgressIP がアクティブな状態でNetNamespace の EgressIP を変更すると、重複した EgressIP が同じ HostSubnets の NetNamespaces に割り当てられ、egress IP があるプロジェクトまたはノードから別のプロジェクトまたはノードに移行すると egress IP は機能しなくなりました。また、同じ egress IP が 2 つの異なるプロジェクトまたはノードに割り当てられる場合も、重複された割り当てが削除された後に適切に機能しなくなる

可能性があります。NetNamespace の EgressIPs フィールドは、egress IP がアクティブな場合に変更するように修正されています。これにより、静的なプロジェクトごとの egress IP がより安全に機能できるようになりました (BZ#1551028)。

- OpenShift ノードプロセスの kube-proxy および kubelet の部分には iptables と対話する方法について記述する設定オプションの異なるデフォルト値が指定されていました。これにより、OpenShift では正しくない iptables ルールが定期的に追加され、これにより一部のプロジェクトごとの静的 egress IP が正しくないルールが取り除かれるまで一定期間使用できなくなりました。正しくないルールが存在する場合、それらのプロジェクトからのトラフィックは egress IP 自体ではなく、egress IP をホストするノードのノード IP アドレスを使用しました。この整合性のない設定は解決され、正しくない iptables ルールは追加できなくなり、プロジェクトはそれらの静的 egress IP を一貫して使用できるようになりました (BZ#1552869)。

- 以前のバージョンでは、OpenShift のデフォルトネットワークプラグインには Kubernetes のアップストリームに導入された最新の NetworkPolicy 機能が含まれていませんでした。これらには、Pod または namespace ではなく IP アドレスに基づいて egress およびポリシーを制御するためのポリシーが含まれていました。つまり、バージョン 3.9 では、**`ipBlock`** スタンザで NetworkPolicy を作成するとノードがクラッシュし、「egress」ルールのみを含む NetworkPolicy を作成すると ingress トラフィックがブロックされました。今回のバージョンでは、OpenShift Container Platform は実装していない段階の、サポートされていない NetworkPolicy 機能を認識でき、NetworkPolicy に **`ipBlock`** ルールが含まれる場合はそれらのルールは無視されるようになりました。これにより、**`ipBlock`** ルールがポリシー内の唯一のルールである場合に、ポリシーは「denay all」として処理される可能性があります。NetworkPolicy に「egress」ルールのみが含まれる場合、これは完全に無視され、ingress には影響がなくなりました (BZ#1583255)。

- Pod の削除時に、一部の IP ファイルは意図される通りに削除されませんでした。これは、無効なコンテナーを取得するガベージコレクションによって引き起こされていました。kubelet は再起動が必要な場合に、1 つ以上のコンテナーからの情報を保持します。今回のバグ修正により、適切なクリーンアップがネットワークプラグインが成功を返す場合にのみ実行されるようになり、他のエラーはその後のランタイム (例: dockershim または CRI-O) が kubelet に戻る前に発生します (BZ#1532965)。

- 以前のバージョンでは、**dnsmasq** サービスがランダムにフリーズし、手動での再起動が必要でした。これにより、OpenShift Container Platform ノードホストで **dnsmasq** サービスについてのログがキャプチャーされませんでした。これは **dnsmasq** に接続するインターフェースがリリース間で変更し、サービスがオーバーロードすることによって生じていました。**`dns-forward-max`** および **`cache-size`** オプションの制限は 10000 に引き上げられ、このサービスは予想通りに機能するようになりました (BZ#1560489)。

- 更新された egress ポリシーは送信トラフィックをブロックし、OVS フローにパッチを適用し、トラフィックを再度有効にする必要がありましたが、DNS 名の OVS フロー生成に時間がかかっていました。そのため、egress トラフィックが数秒間ダウンしました。今回のバグ修正により、egress ポリシー処理が、送信トラフィックをブロックする前に新規 OVS フローを事前に生成するように更新されており、これにより egress ポリシーの更新時のダウンタイムが減少します (BZ#1558484)。

- 内部状態の正しくないクリーンアップにより、1 つのプロジェクトから「プロジェクトごとの静的な egress IP」を削除してからその IP を別のプロジェクトで再利用しようとする場合に、新規プロジェクトの OVS ルールが誤って作成されました。egress IP は新規プロジェクトに作成されず、古いプロジェクトからのトラフィックで再度使用される可能性がありました。egress IP を削除する際に、内部状態のクリーンアップは適切に実行されるようになり、egress トラフィックが予想通りに機能するようになりました (BZ#1543786)。

- namespace ごとの静的 egress IP を使用する場合に、外部トラフィックすべてが egress IP 経由でルーティングされます。**外部** とは別の Pod にダイレクトされないすべてのトラフィックを

意味し、これには Pod から Pod のノードへのトラフィックも含まれます。Pod が DNS にノードの IP アドレスを使用するように指示され、Pod が静的な egress IP を使用している場合、DNS トラフィックはまず egress ノードにルーティングされてから、元のノードに戻ってきます。ここで他のホストからの DNS 要求を許可しないように設定されていると、Pod が DNS を解決できない可能性があります。Pod からノードへの DNS 要求は egress IP をバイパスして直接ノードに到達できるようになり、DNS は正常に機能するようになりました (**BZ#1557924**)。

Pod

- 以前のバージョンでは、**`oc describe`** コマンドのエラーおよび警告は明確ではありませんでした。今回のバージョンで、この問題は解決されています (**BZ#1523778**)。
- 以前のバージョンでは、ガベージコレクターは停止したコンテナーが使用されているイメージの削除を試行しました。変更が OpenShift Container Platform バージョン 3.10 に加えられ、これにより、ガベージコレクターが停止したコンテナーで使用されているイメージの削除を試行できなくなりました (**BZ#1577739**)。
- **`cpu-cfs-quota`** は、**`node-config.yaml`** ファイルで **`cpu-cfs-quota`** が **`false`** に設定されている場合でも適用されていました。これは、**`cfs`** クォータのコンテナー cgroup がバインド解除されているのに Pod レベルの cgroup がバインドされているために生じました。この問題は解決され、Pod レベルの cgroup がバインド解除されたままになるように変更が加えられました。**`cpu-cfs-quota`** が **`false`** に設定される場合、いずれの制限も施行されなくなります (**BZ#1581409**)。
- Web コンソールは、スケールターゲットの実際のグループの種類を問わず、HPA リソースの作成時に **`extensions/v1beta1`** を API バージョンとして誤って割り当てていました。この問題は修正されています (**BZ#1543043**)。

ルーティング

- 以前のバージョンでは、HAProxy 設定は読み込みに失敗すると、ルーターがルートを提供できなくなるという障害が生じました。これは、ヘッドレスサービスに **`service.Spec.ClusterIP=None`** フィールドが設定され、これがアイドル状態からの復帰の一環として無視されないために生じました。今回の修正により、HAProxy 設定は、アイドル状態からの復帰時にヘッドレスサービスを無視し、ルーターが予想通りにルートを提供できるようになりました (**BZ#1567532**)。
- パスベースのルートは、TLS シナリオが混在する場合に予想通りに機能しませんでした。ルートタイプを別々のマップファイルに分割することにより問題が生じました。これにより、haproxy が正しくないルートに一致する結果になりました。今回のバージョンで、マップは自動的にマージされるようになり、受信要求を対応するバックエンドに一致させるようそれらを適切に検索できるようになりました (**BZ#1534816**)。
- HAProxy Docker イメージのアップグレード時に、デフォルトでは要求のロギングは実行されません。ロギングが **`httplog`** オプションを使用して要求される場合、このオプションは TCP のみの接続では利用できないために警告メッセージが表示されました。この状態になる場合、HAProxy は代わりに **`tcplog`** オプションを使用するようにフォールバックします。この警告メッセージには有害な影響はありませんが、削除されています (**BZ#1533346**)。

サービスブローカー

- **`type: openshift`** レジストリーアダプターは APB イメージの検出をサポートしていません。これは、このレジストリーアダプターのユーザーはブートストラップするイメージの一覧を手動で組み込む必要があることを意味します。今回の機能拡張により、新規レジストリーア

ダプターの **type: partner_rhcc** が導入され、これは https://registry.connect.redhat.com で機能し、この手動の要件なしにイメージの検出をサポートするようになりました (**BZ#1576881**)。

- サービスインスタンスのプロビジョニング解除を試行する際に、プロセス中に無効な応答本体と共にエラーが出され、プロビジョニング解除プロセスが失敗しました。今回変更が新たに実装され、操作キーで適切な応答本体を返し、プロビジョニング解除ワークフローの全体的な堅牢性を強化でき、プロビジョニング解除が成功する可能性が高くなりました (**BZ#1562732**)。

- Open Source Broker (OSB) API ドキュメントによると、バインディングが存在する場合、ステータス **200 OK** がバインディングの呼び出しから返されることになっていました。しかし、正しくない応答コード (**201**) が返される問題がありました。非同期バインディングのサポートの導入により、この問題は解決されています (**BZ#1563560**)。

サービスカタログ

- サービスクラスがサービスブローカーのカタログのプロビジョニングされたサービスから削除される場合、サービスカタログはクラスに **removedFromBrokerCatalog: true** のマークを付けます。これにより、このクラスを新規サービスプランまたはインスタンスで使用することができませんでした。サービスクラスがブローカーカタログに再度追加される場合にステータスを **removedFromBrokerCatalog: false** にリセットできず、削除されたクラスを再び使用することができないことが問題となっていましたが、この問題は解決されています (**BZ#1548122**)。

- 以前のバージョンでは、Prometheus コンソールは、バックエンドで curl でのみ使用できたサービスカタログメトリクスへのアクセスを許可しませんでした。サービスカタログコントローラーは Prometheus のメトリクスを取得できるよう公開することになり、サービスカタログをモニターできるようになりました (**BZ#1549021**)。

ストレージ

- ローカル永続ストレージボリューム (PV) の容量は、**df** ユーティリティーによって報告されるものとは異なって報告される場合がありました。これは、新規にマウントされたデバイスの Pod への伝播が行われず、追加の PV が設定済みのディレクトリーに作成されることによって生じました。この新たに作成された PV の容量はルーターデバイスと等しくなっていました。この伝播の問題は解決されました (**BZ#1490722**)。

- AWS クラウドの API 呼び出しクォータに達すると、特定の AWS API 呼び出しはエラーを返しました。これらのエラーは、AWS 永続ボリュームの割り当てを解除し、一部の AWS ボリュームが、それらを使用する Pod がないにもかかわらずノードに割り当てられたままの状態の場合に適切に処理されませんでした。これらのボリュームの割り当ては手動で解除する必要がありました。そうでないと、それらのボリュームは停止した状態が永久に続くことになりました。今回のバグ修正により、動的ボリューム割り当て解除についての AWS API 呼び出しエラーの処理が更新されました。その結果として、AWS API 呼び出しクォータに達すると、割り当て/割り当て解除コントローラーは成功するまでボリュームの割り当て解除を試行し、割り当てを解除する必要のあるボリュームが実際に解除されことを確認できるようになりました (**BZ#1537236**)。

テスト

- 以前のバージョンでは、**masterConfig.ImagePolicyConfig.ExternalRegistryHostname** が **master-config.yaml** に追加され、API およびコントローラーサービスが再起動すると、API Pod は再作成されましたが、コントローラー Pod は **CrashLoopBackOff** のエラーを出しました。AWS での **m3.large** インスタンスの使用よりこの問題が解決されています (**BZ#1593635**)。

アップグレード

- ノードのアップグレードプロセス中に任意のタスクを実行するフックセットを定義できるようになりました。これらのフックを実装するには、**openshift_node_upgrade_pre_hook**、**openshift_node_upgrade_hook** または **openshift_node_upgrade_post_hook** を、実行するタスクファイルのパスに設定します。**openshift_node_upgrade_pre_hook** フックは、ノードのドレイン (解放) 後およびアップグレード前に実行します。**openshift_node_upgrade_hook** は、ノードのドレイン (解放) およびパッケージの更新後のスケジュール可能とマークされる前の段階で実行されます。また、**openshift_node_upgrade_post_hook** フックは、ノードがスケジュール可能とマークされた後で、他のノードに移行する直前に実行されます (**BZ#1559143**)。

2.6. テクノロジープレビュー機能

現在、今回のリリースに含まれる機能にはテクノロジープレビューのものがあります。これらの実験的機能は、実稼働環境での使用を目的としていません。これらの機能に関しては、Red Hat カスタマーポータルの以下のサポート範囲を参照してください。

テクノロジープレビュー機能のサポート範囲

以下の表では、**TP** とマークが付いた機能は **テクノロジープレビュー**、**GA** とマークが付いた機能は **一般公開** 機能です。

表2.1 テクノロジープレビュートラッカー

機能	OCP 3.7	OCP 3.9	OCP 3.10
Prometheus のクラスターのモニタリング	TP	TP	TP
ローカルストレージ永続ボリューム	TP	TP	TP
ランタイム Pod の CRI-O	TP	GA* [a]	GA
テナント駆動型のスナップショット	TP	TP	TP
oc CLI プラグイン	TP	TP	TP
サービスカタログ	GA	GA	GA
Template Service Broker	GA	GA	GA
OpenShift Automation Broker	GA	GA	GA
ネットワークポリシー	GA	GA	GA

機能	OCP 3.7	OCP 3.9	OCP 3.10
サービスカタログの初回エクスペリエンス	GA	GA	GA
プロジェクト追加に関する新しいフロー	GA	GA	GA
検索カタログ	GA	GA	GA
CFME インストーラー	GA	GA	GA
Cron ジョブ	TP	GA	GA
Kubernetes デプロイメント	TP	GA	GA
StatefulSets	TP	GA	GA
明示的なクォータ	TP	GA	GA
マウントオプション	TP	GA	GA
docker のシステムコンテナー、CRI-O	TP	廃止	-
システムコンテナーからのインストール	TP	GA	GA
Hawkular エージェント	廃止	-	-
Pod の PreSet	廃止	-	-
experimental-qos-reserved	TP	TP	TP
Pod sysctls	TP	TP	TP
中央監査	TP	GA	GA
外部プロジェクトトラフィックの静的 IP	TP	GA	GA
テンプレート完了の検出	TP	GA	GA
`replicaSet`	TP	GA	GA
Mux	TP	TP	TP

機能	OCP 3.7	OCP 3.9	OCP 3.10
クラスター化された MongoDB テンプレート	コミュニティー	-	-
クラスター化された MySQL テンプレート	コミュニティー	-	-
Kubernetes リソースでのイメージストリームの使用	TP	GA	GA
デバイスマネージャー	-	TP	GA
永続ボリュームのサイズ調整	-	TP	TP
Huge Page	-	TP	GA
CPU マネージャー	-	TP	GA
デバイスプラグイン	-	TP	GA
fluentd 向けの syslog 出力プラグイン	-	TP	TP
コンテナーストレージインターフェース (CSI)	-	-	TP
OpenStack Manila を使用した永続ボリューム (PV) のプロビジョニング	-	-	TP
Node Problem Detector	-	-	TP
ローカルー時ストレージの保護	-	-	TP
Descheduler	-	-	TP
Podman	-	-	TP
Kuryr CNI プラグイン	-	-	TP
PID Namespace 共有の制御	-	-	TP

機能	OCP 3.7	OCP 3.9	OCP 3.10

[a] * のマークが付いている機能は、z ストリームパッチで提供されることを指します。

2.7. 既知の問題

- 1.10 リベースで解決される kubelet が停止状態になるという既知の問題があります。この状態では、kubelet が **system:anonymous cannot access resource foo** などのメッセージを表示します。これは、kubelet が証明書を更新する前にそれらの証明書が期限切れになったことを示します。kubelet を再起動してもこの問題が解決されない場合は、**/etc/origin/node/certificates/** の内容を削除してから kubelet を再起動します。

- 「Upgrading Clusters」で説明されている blue-green 方法というノードのデプロイメント方法は、OpenShift Container Platform 3.9 から 3.10 への初回アップグレードのパスでのみ使用する必要があります。これは 非同期 OpenShift Container Platform 3.10.z 更新の初回リリース時にさらに更新されます。

- GA リリース版の『Downgrading OpenShift』で。etcd を復元する手順についての問題が発見されました。このドキュメントの更新により、この問題はなくなりました。

- OpenShift Container Platform 3.10 は、APB の複数のインスタンスが同じ namespace で起動できる機能を追加しています。この新規機能では、各インスタンスについてグローバルー意識別子 (GUID) を使用することが必要になります。3.9 バージョンの APB でデプロイされたインスタンスには GUID がないものの、3.10 の APB ではこれが必要になります。
  3.10 の APB では 3.9 でデプロイされたサービスを管理することができません。そのサービスには新たに必要とされる GUID がないためです。これにより、3.9 から 3.10 にアップグレードされたクラスターの場合に、3.9 で以前にデプロイされたアプリケーションが 3.10 の APB からプロビジョニング解除される場合にエラーが生じます。

  現時点で、この問題には 2 つの回避策があります。

  - クラスターの 3.10 へのアップグレードが完了した後に、アプリケーションの namespace を削除し、これを再作成します。これは APB のバージョン 3.10 を使用し、予想通りに機能します。

  - OpenShift Ansible Broker の設定を変更し、APB の 3.9 バージョンをそのまま使用できるようにします。これは推奨される方法ではありません。APB が古い 3.9 バージョンを使用する間に Broker が 3.10 のコードを使用するという不利な点があります。

    a. 「Modifying the OpenShift Ansible Broker Configuration」の手順に従って、ラベルを **`v3.9`** に変更します。

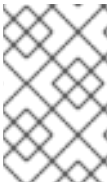
    b. **`apb bootstrap`** コマンドを実行し、Broker をブートストラップし、カタログを再度一覧表示します。

  (**BZ#1586108**)

2.8. エラータの非同期更新

OpenShift Container Platform 3.10 のセキュリティー、バグ修正、拡張機能の更新は、Red Hat Network 経由で非同期エラータとして発表されます。OpenShift Container Platform 3.10 の全エラータは Red Hat カスタマーポータル から入手できます。非同期エラータについてはOpenShift Container Platform ライフサイクル を参照してください。

Red Hat カスタマーポータルのユーザーは、Red Hat サブスクリプション管理 (RHSM) のアカウント設定でエラータの通知を有効にすることができます。エラータの通知を有効にすると、登録しているシステムに関連するエラータが新たに発表されるたびに、メールで通知が送信されます。



注記

OpenShift Container Platform のエラータ通知メールを生成させるには、Red Hat カスタマーポータルのユーザーアカウントでシステムが登録されており、OpenShift Container Platform エンタイトルメントを使用している必要があります。

以下のセクションは、これからも継続して更新され、今後の OpenShift Container Platform 3.10 バージョンの非同期リリースで発表されるエラータの拡張機能およびバグ修正に関する情報を追加していきます。たとえば、OpenShift Container Platform 3.10.z などのバージョン付けされた非同期リリースについてはサブセクションで説明します。さらに、エラーター情報がアドバイザリーで指定されたスペースに収まらないリリースについては、詳細についてその後のサブセクションで説明します。

重要

OpenShift Container Platform のいずれのバージョンについても、『クラスターのアップグレード』についての指示には必ず目を通してください。

2.8.1. RHBA-2018:2376: OpenShift Container Platform 3.10.34 バグ修正および機能拡張の更新

発行日: 2018-08-28

OpenShift Container Platform リリース 3.10.34 が公開されました。この更新に含まれるパッケージおよびバグ修正は、RHBA-2018:2376 アドバイザリーにまとめられています。この更新に含まれるコンテナーイメージは、RHBA-2018:2377 アドバイザリーで提供されています。

アドバイザリーでは、このリリースすべてのバグ修正および機能拡張に関する説明は除外されています。アップグレードについての注記およびこのリリースに含まれるバグ修正および機能拡張の詳細については、以下のセクションを参照してください。

2.8.1.1. バグ修正

- Mux が設定され、ログが属するプロジェクトまたは namespace を検出できない場合に、ログは **`project.mux-undefined`** にインデックス化されました (**`mux-undefined`** は Mux のデフォルト namespace です)。同時に、fluentd (Mux 設定なし) がこれらのログを **.orphaned.YYYY.MM.DD** インデックスに配置します。今回のバグ修正では、このような孤立したログも Mux ケースの **.orphaned.YYYY.MM.DD** インデックスにインデックス化されるようになりました (**BZ#1538560**)。

- インストーラーは CPU およびメモリーの正しくない **spec** 属性を使用しており、CPU 制限の変更を許可しませんでした。そのため、それらの値は無視されていました。**`cpu_limit`** にはパッチを条件的に適用し (定義されている場合)、CPU およびメモリー要求を指定するために使用される属性名を修正する必要がありました。今回のバグ修正では、これらの値は予想通りに使用されるようになりました (**BZ#1575546**)。

- Ansible テンプレートでは、セレクターで値を引用符で囲まないため、無効な JSON を生成しました。セレクターの値は引用符で囲まれ、PVC がセレクターで作成できるようになりました (**BZ#1597282**)。

- 9100 ポートはデフォルトですべてのノードでブロックされました。Prometheus は、ポート

9100 でリッスンする他のノードで実行される **node_exporter** サービスを取り出すことができませんでした。ファイアウォール設定が 9000-1000 のポート範囲の受信 TCP トラフィックを許可するように変更され、Prometheus は **node_exporter** サービスの取り出しを実行できるようになりました (BZ#1600562)。

- 最近、クラウドプロバイダーからノードアドレスを継続的にフェッチする **cloudResourceSyncManager** が導入されました。Kubelet はノードアドレスを **cloudResourceSyncManager** から受信します。ノードの登録または kubelet の起動時に、kubelet は **cloudResourceSyncManager** からのブロッキングループでノードアドレスをフェッチします。これまでは kubelet が最初にノードアドレスをフェッチする前に **cloudResourceSyncManager** が起動していないという問題があり、kubelet はブロッキングループで停止し、返されませんでした。ネットワークレベルでノードが失敗し、ノードを登録することもできませんでした。また、kubelet が早期にブロックされるため、**cloudResourceSyncManager** の起動する機会がありませんでした。しかし、**CloudResourceSyncManager** は kubelet の起動プロセスの早期に起動されるようになり、kubelet がその上でブロックされることはなくなり、**cloudResourceSyncManager** が常に起動するようになりました (BZ#1603611)。

- ノードセレクターが **true** の値として指定される場合、それはブール値として解釈され、daemonset のデプロイメントが失敗しました。daemonset を作成するためのテンプレートは指定された値を引用符で囲むように更新され、値が文字列として解釈されるようになりました (BZ#1609027)。

- ユーザーに関連付けられたグループは、テンプレートで作成されたオブジェクトの readiness (準備状態) を確認するためにアクセスチェックを実行する際にチェックされませんでした。グループのメンバーシップに基づいてのみユーザーがアクセスできるオブジェクトについては、オブジェクトはテンプレートで作成されても、その readiness については確認できないため、テンプレートのインスタンスレベルで readines にパスしない場合がありました。オブジェクトの作成時のみではなく、readiness チェック操作を実行する際にユーザーのグループを渡す必要があります。ユーザーのグループメンバーシップがチェックを許可する限り、オブジェクトの readiness が正常にチェックされるようになりました。(BZ#1610994)

- tar ストリームの展開から出力をパイプする際に競合状態が見られました。多数のファイルを含むバイナリービルドは無期限にハングする可能性がありました。tar ストリームのロジックは、競合状態のない以前のメカニズムを使用するように元に戻され、多数のファイルを含むバイナリービルドが正常に完了するようになりました (BZ#1614493)。

- デフォルトでは、古いバージョンの dnsmasq は送信 DNS クエリーの特権付きの低い番号が付けられたソースポートを使用できました。しかし、ファイアウォールルールが予約されたポートから受信されるクエリーをドロップする場合を含め、送信 DNS クエリーはドロップされる可能性がありました。dnsmasq は **min-port** 設定を使用して設定され、送信クエリーの最小ポート番号は **1024** に設定されるようになり、DNS クエリーはドロップされなくなりました (BZ#1614984)。

- Ansible 2.6.0 では、**|bool** が **false** として設定されている状態では未定義の変数は評価されず、**logging_elasticsearch_rollout_override** について **| default(false)** を定義する必要があります。今回のバグ修正により、Playbook は正常に実行されるようになっています (BZ#1615194)。

2.8.1.2. 機能拡張

- デフォルトの fluentd メモリーは **756m** に引き上げられました。パフォーマンスおよびスケーリングのテストにより、メモリー不足による障害およびコンテナーの再起動を避けるため、機能が改善された fluentd にはより多くのメモリーが必要になることが明らかになりました。fluentd のメモリー不足になる可能性は低くなりました (BZ#1600258)。

- アップグレード時に、ノードが CRI-O をシステムコンテナーとして実行しているかどうかを確認するためにチェックが実行されます。該当する場合には CRI-O システムコンテナーがアンインストールされ、CRI-O RPM がインストールされました。CRI-O をシステムコンテナーとしての実行することはサポートされていません。OpenShift Container Platform 3.9 のインストール時に、ノードは CRI-O をシステムコンテナーとして設定した状態で誤ってインストールされる可能性があります。OpenShift Container Platform 3.9 から 3.10 にアップグレードされたノードについては、RPM から実行される CRI-O が設定されたサポートされる設定に変換されます (**BZ#1618425**)。

2.8.1.3. アップグレード

既存の OpenShift Container Platform 3.9 または 3.10 クラスターをこの最新リリースにアップグレードするには、自動アップグレード Playbook を使用します。説明は、「Performing Automated In-place Cluster Upgrades」を参照してください。

第3章 XPAAS リリースノート

xPaaS ドキュメントのリリースノートは、Red Hat カスタマーポータル 上の XPaaS 専用のブックに移行されています。

第4章 OPENSHIFT ENTERPRISE 2 との比較

4.1. 概要

OpenShift Container Platform 3 は OpenShift バージョン 3 (v3) のアーキテクチャーをベースにしており、OpenShift バージョン 2 (v2) とは非常に異なる製品です。OpenShift v2 と同じ用語が数多く v3 でも使用されており、同じ機能が実行されますが、異なる用語が使用される場合もあり、背景で実行される内容は非常に異なる場合がありますが、OpenShift 自体は依然としてアプリケーションプラットフォームとして機能します。

このトピックでは、バージョンの違いを詳しく説明し、OpenShift のユーザーの OpenShift v2 から OpenShift v3 への移行を支援します。

4.2. アーキテクチャーの変更

ギア vs コンテナー

ギアは OpenShift v2 のコアコンポーネントです。カーネルの namespace、cGroups および SELinux などのテクノロジーで、スケーラビリティーが高く、セキュアなコンテナーアプリケーションプラットフォームを OpenShift ユーザーに提供します。ギア自体はコンテナーテクノロジーの 1 つの形態です。

OpenShift v3 は、ギアのアイデアを次のレベルに進化させ、v2 コンテナーテクノロジーの進化したバージョンとして Docker を使用します。このコンテナーアーキテクチャーは OpenShift v3 の中核となります。

Kubernetes

OpenShift v2 のアプリケーションは通常は複数のギアを使用するのに対し、OpenShift v3 のアプリケーションは複数のコンテナーを使用します。OpenShift v2 では、ギアのオーケストレーション、スケジューリングおよび配置は、OpenShift Broker ホストが処理していました。OpenShift v3 では、マスターホストに Kubernetes を統合してコンテナーのオーケストレーションを駆動します。

4.3. アプリケーション

アプリケーションは依然として OpenShift の中心的な要素です。OpenShift v2 では、アプリケーションが単一のユニットで、カートリッジタイプ 1 つに対して 1 つの Web フレームワークで構成されていました。たとえば、アプリケーションに PHP 1 つと MySQL 1 つを含めることはできますが、Ruby 1 つ、PHP 1 つ、および MySQL 2 つを含めることはできませんでした。また、MySQL などのデータベースカートリッジだけで機能させることもできませんでした。

アプリケーションの範囲を制限することで、OpenShift が環境変数を使用してアプリケーション内のすべてのコンポーネントをシームレスにリンクすることができます。たとえば、Web フレームワークはすべて **`OPENSHIFT_MYSQL_DB_HOST`** および **`OPENSHIFT_MYSQL_DB_PORT`** 変数を使用して MySQL に接続する方法を把握しますが、このリンクはアプリケーション内に限られており、連携して機能するように設計されたカートリッジ内でしか機能しませんでした。2 つのアプリケーション間で MySQL インスタンスを共有するなど、アプリケーションのコンポーネント間でリンクする方法はありませんでした。

他の PaaSes の大半は、Web フレームワークだけに制限され、他の種類のコンポーネントについては外部サービスに依存しますが、OpenShift v3 ではさらに多くのアプリケーショントポロジーや管理が可能です。

OpenShift v3 では、サービスをリンクする概念として「アプリケーション」という用語を使用しています。コンポーネントは必要な数だけいくつでも持つことができ、プロジェクト 内でコンテナー化した

り、柔軟にリンクしたり、オプションで、グループ化や構造化のためにラベルを付けたりできます。この更新されたモデルは、MySQL インスタンスをスタンドアロンで使用したり、JBoss コンポーネント間で共有したりすることを可能にします。

柔軟にリンクするとは、任意の 2 つのコンポーネントをリンクできることを意味します。コンポーネントの 1 つが環境変数をエクスポートでき、他方がこれらの環境変数の値を使用できる限り (また環境変数名が変換される可能性があります)、ベースにするイメージを変更せずに 2 つのコンポーネントをリンクすることができます。そのため、両方をフォークし、互換性を持たせるように設定を変更せずに、任意のデータベースおよび Web フレームワークの最適にコンテナー化された実装を直接使用できるようになります。

これは OpenShift 上には何でもビルドできることを意味し、OpenShift の主要な目的と合致しています。OpenShift の主要な目的とは、反復可能なライフサイクルに基づいてアプリケーション全体をビルドするためにコンテナーベースのプラットフォームとして機能することにあります。

4.4. カートリッジ VS イメージ

OpenShift v2 のカートリッジの概念の代わりに、OpenShift v3 ではイメージ が使用されるようになりました。

OpenShift v2 のカートリッジはアプリケーションのビルドにおける中心的な要素です。各カートリッジは、必要なライブラリー、ソースコード、ビルドメカニズム、接続ロジック、ルーティングロジックを事前定義済みの環境と共に提供し、アプリケーションの各種コンポーネントを実行していました。

ただし、カートリッジには欠点があり、開発者のコンテンツとカートリッジのコンテンツの違いが明確でないことや、ユーザーにはアプリケーションの各ギアにあるホームディレクトリーの所有権がないことなどの不利な点がありました。また、カートリッジは大規模なバイナリーの配信メカニズムとして最適ではありませんでした。カートリッジ内から外部の依存関係を使用することもできましたが、これをあえて実行することはカプセル化の利点が活かすことにはなりませんでした。

パッケージ化の観点では、イメージはカートリッジよりも多くのタスクを実行し、より優れたカプセル化機能や柔軟性を提供します。ただし、カートリッジには、イメージには含まれないビルド、デプロイ、ルーティングのロジックが含まれています。OpenShift v3 では、これらのニーズは「Source-to-Image (S2I)」および「テンプレートの設定」で対応しています。

依存関係

OpenShift v2 では、カートリッジの依存関係はカートリッジマニフェストの **`Configure-Order`** または **`Requires`** で定義されていました。OpenShift v3 では、Pod が自らを事前定義された状態にする宣言型モデルを使用します。インストール時の順序だけでなく、適用される明示的な依存関係がランタイム時に実行されます。

たとえば、開始前に別のサービスを利用可能な状態にしておく必要がある場合があります。このような依存関係チェックは、2 つのコンポーネントの作成時のみではなく常に適用できます。そのため、依存関係チェックをランタイムにプッシュすることで、システムを長期にわたって正常な状態に保つことができます。

コレクション

OpenShift v2 のカートリッジはギア内に共同で配置されますが、OpenShift v3 の イメージ は コンテナー と 1 対 1 でマッピングされます。コンテナーは、共同配置のメカニズムとして Pod を使用します。

ソースコード

OpenShift v2 では、アプリケーションには 1 つ以上の Web フレームワークと 1 つの git リポジトリー

必要でした。OpenShift v3 では、ソースからビルドするイメージを選択でき、そのソースは OpenShift 外にある場合もあります。ソースはイメージから切断されているので、イメージとソースの選択は異なる操作となります (ソースの設定はオプションです)。

ビルド

OpenShift v2 では、ビルドはアプリケーションギアで行われていたので、リソースに制約によりスケーリングできないアプリケーションの場合にはダウンタイムが発生していました。v3 では、個別のコンテナーで ビルド が行われます。また OpenShift v2 のビルド結果では、rsync を使用してギアを同期していました。v3 では、ビルド結果はまずイミュータブルな不変イメージとしてコミットされ、内部レジストリーに公開されます。その後、このイメージはクラスターのいずれかのノードで起動できるか、または後にロールバックに利用できるようになります。

ルーティング

OpenShift v2 では、アプリケーションに拡張性を持たせるかやアプリケーションのルーティング層が高可用性 (HA) を確保するために有効にするかどうかを最初に選択する必要がありました。OpenShift v3 では、単純にアプリケーションコンポーネントを 2 つ以上のレプリカにスケールアップすることで、ルート を HA 対応のファーストクラスのオブジェクトとして使用することができます。アプリケーションを再作成したり、DNS エントリーを変更したりする必要はありません。

ルート自体はイメージから切断されています。以前のバージョンでは、カートリッジによりデフォルトのルートセットが定義され、アプリケーションにエイリアスを追加できました。OpenShift v3 では、テンプレートを使用してイメージに任意の数のルートを設定できます。これらのルートを使用すると、システムルートとユーザーエイリアスを区別することなく、公開するスキーム、ホスト、パスを任意に変更できます。

4.5. ブローカー VS マスター

OpenShift v3 の マスター は、OpenShift v2 のブローカーホストとよく似ていますが、通常 **etcd** が各マスターホストにインストールされるので、OpenShift v2 のブローカーが使用する MongoDB および ActiveMQ の階層は不要になりました。

4.6. ドメイン VS プロジェクト

プロジェクト は基本的には v2 のドメインに相当します。