OpenShift Container Platform 4.2

レジストリー

OpenShift Container Platform 4.2 のレジストリーの設定

Last Updated: 2020-04-15

OpenShift Container Platform 4.2 レジストリー

OpenShift Container Platform 4.2 のレジストリーの設定

法律上の通知

概要

本書では、OpenShift Container Platform 4.2 の内部レジストリーを設定し、管理する方法について説明します。また、OpenShift Container Platform 4.2 に関連付けられたレジストリーの概要も提供します。

目次

第1章 イメージレジストリー

1.1. 統合 OPENSHIFT CONTAINER PLATFORM レジストリー

OpenShift Container Platform は、クラスター上の標準ワークロードとして実行されるコンテナーイメージレジストリーでビルドを提供します。このレジストリーはインフラストラクチャー Operator によって設定され、管理されます。これはユーザーがワークロードを実行するイメージを管理するために追加設定なしで使用できるソリューションを提供し、既存のクラスターインフラストラクチャーの上部で実行されます。このレジストリーは、他のクラスターワークロードのようにスケールアップまたはスケールダウンでき、特定のインフラストラクチャーのプロビジョニングを必要としません。さらに、これはクラスターのユーザー認証および認可システムに統合されるため、イメージを作成し、取得するためのアクセスは、イメージリソースでユーザーのパーミッションを定義することによって制御できることを意味します。

通常、レジストリーはクラスター上にビルドされたイメージの公開ターゲットとして、またクラスター上で実行されるワークロードのイメージのソースとして使用されます。新規イメージがレジストリーにプッシュされると、クラスターにはその新規イメージについて通知され、他のコンポーネントは更新されたイメージに応答し、これを使用できます。

イメージデータは 2 つの場所に保存されます。実際のイメージデータは、クラウドストレージまたはファイルシステムボリュームなどの設定可能なストレージの場所に格納されます。標準のクラスター API によって公開され、アクセス制御を実行するために使用されるイメージメタデータは、標準的な API リソース、とくにイメージおよびイメージストリームとして保存されます。

追加リソース

- OpenShift Container Platform のイメージレジストリー Operator

第2章 OPENSHIFT CONTAINER PLATFORM のイメージレジストリー OPERATOR

イメージレジストリー Operator は、OpenShift Container Platform レジストリーの単一インスタンスをインストールし、レジストリーストレージのセットアップを含む、レジストリーのすべての設定を管理します。

注記

ストレージは、AWS、GCP、Azure または OpenStack にインストーラーでプロビジョニングされるインフラストラクチャークラスターをインストールする場合にのみ自動的に設定されます。

コントロールプレーンのデプロイ後、Operator はクラスターで検出される設定に基づいてデフォルトの **configs.imageregistry.operator.openshift.io** リソースインスタンスを作成します。

完全な **configs.imageregistry.operator.openshift.io** リソースを定義するのに利用できる情報が十分にない場合、その不完全なリソースが定義され、Operator は足りない情報を示す情報を使ってリソースのステータスを更新します。

イメージレジストリー Operator は **openshift-image-registry** namespace で実行され、その場所のレジストリーインスタンスも管理します。レジストリーのすべての設定およびワークロードリソースはその namespace に置かれます。

2.1. イメージレジストリー OPERATOR の設定パラメーター

configs.imageregistry.operator.openshift.io リソースは以下の設定パラメーターを提供します。

パラメーター	説明
ManagementState	**Managed**: Operator は、設定リソースが更新されるとレジストリーを更新します。 **Unmanaged**: Operator は設定リソースへの変更を無視します。 **Removed**: Operator はレジストリーインスタンスを取り除き、Operator がプロビジョニングしたすべてのストレージを削除します。
Logging	レジストリーインスタンスの **loglevel** を設定します。
HTTPSecret	デフォルトで生成されるアップロードのセキュリティーを保護するためにレジストリーで必要な値。
Proxy	マスター API およびアップストリームレジストリーの呼び出し時に使用されるプロキシーを定義します。
Storage	**Storagetype**: レジストリーストレージを設定するための詳細。たとえば、S3 バケットの位置情報 (coordinate) など。 通常はデフォルトで設定されます。
ReadOnly	レジストリーインスタンスが新規イメージのプッシュや既存イメージの削除の試行を拒否するかどうかを示します。

パラメーター	説明
Requests	API 要求の制限の詳細。指定されたレジストリーインスタンスが追加リソースをキューに入れる前に処理する並列要求の数を制御します。
DefaultRoute	外部ルートがデフォルトのホスト名を使用して定義されるかどうかを決定します。これが有効にされている場合、ルートは re-encrypt 暗号を使用します。デフォルトは false に設定されます。
Routes	作成する追加ルートの配列。ルートにホスト名および証明書を指定します。
Replicas	レジストリーのレプリカ数。

2.2. イメージレジストリーのデフォルトルートをカスタムリソース定義 (CRD、CUSTOM RESOURCE DEFINITION) で有効にする

OpenShift Container Platform では、**Registry** Operator はレジストリー機能を制御します。Operator は、**configs.imageregistry.operator.openshift.io** カスタムリソース定義 (CRD) で定義されます。

イメージレジストリーのデフォルトルートを自動的に有効にする必要がある場合には、イメージレジストリー Operator CRD のパッチを適用します。

手順

- イメージレジストリー Operator CRD にパッチを適用します。

```
$ oc patch configs.imageregistry.operator.openshift.io/cluster --type merge -p '{"spec":
{"defaultRoute":true}}'
```

2.3. イメージレジストリー OPERATOR の CONFIGMAP の設定

configs.imageregistry.operator.openshift.io およびシークレットリソースのほかにも、**openshift-image-registry** namespace 内の別の ConfigMap リソースによって設定が Operator に提供されます。

前提条件

- CA は PEM でエンコードされている必要があります。

手順

ConfigMap を **openshift-config** namespace に作成し、その名前を **image.config.openshift.io** リソースの **AdditionalTrustedCA** で使用し、追加の CA を指定することができます。

この CA は外部レジストリーと通信する際に信頼される必要があります。

以下の手順で追加の CA を設定することができます。

1. 追加の CA を設定するには、以下を実行します。

```
$ oc create configmap registry-config --from-file=<external_registry_address>=ca.crt -n
openshift-config
```

```
$ oc edit image.config.openshift.io cluster
spec:
  additionalTrustedCA:
    name: registry-config
```

2. **image-registry** Pod 内のイメージを確認します。

```
$ oc rsh image-registry-xxxxx
sh-4.2
$ ls /etc/pki/ca-trust/source/anchors
<external_registry_address> image-registry.openshift-image-registry.svc..5000 image-registry.openshift-image-registry.svc.cluster.local..5000
```

イメージレジストリー CA の例

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: my-registry-ca
data:
  registry.example.com: |
    -----BEGIN CERTIFICATE-----
    ...
    -----END CERTIFICATE-----
  registry-with-port.example.com..5000: | 1
    -----BEGIN CERTIFICATE-----
    ...
    -----END CERTIFICATE-----
```

1 レジストリーにポートがある場合 (例: **registry-with-port.example.com:5000**)、「**:**」は **..** に置き換える必要があります。

2.4. イメージレジストリー OPERATOR のシークレットの設定

configs.imageregistry.operator.openshift.io および ConfigMap リソースのほかにも、**openshift-image-registry** namespace 内の別のシークレットリソースによって設定が Operator に提供されます。

image-registry-private-configuration-user シークレットは、ストレージのアクセスおよび管理に必要な認証情報を提供します。これは、デフォルト認証情報が見つからない場合に Operator によって使用されるデフォルト認証情報を上書きします。

手順

- 必要なキーが含まれる OpenShift Container Platform シークレットを作成します。

```
$ oc create secret generic image-registry-private-configuration-user --from-file=KEY1=value1 --from-literal=KEY2=value2 --namespace openshift-image-registry
```

2.5. 追加リソース

- AWS のユーザーによってプロビジョニングされるインフラストラクチャーでのレジストリーストレージの設定

- GCP のユーザーによってプロビジョニングされるインフラストラクチャーでのレジストリーストレージの設定
- ベアメタルの場合のレジストリーストレージの設定
- vSphere の場合のレジストリーストレージの設定

第3章 レジストリーストレージの設定

3.1. AWS のユーザーによってプロビジョニングされるインフラストラクチャーでのレジストリーストレージの設定

3.1.1. イメージレジストリー Operator のシークレットの設定

configs.imageregistry.operator.openshift.io および ConfigMap リソースのほかにも、 **openshift-image-registry** namespace 内の別のシークレットリソースによって設定が Operator に提供されます。

image-registry-private-configuration-user シークレットは、ストレージのアクセスおよび管理に必要な認証情報を提供します。これは、デフォルト認証情報が見つからない場合に Operator によって使用されるデフォルト認証情報を上書きします。

AWS ストレージ上の S3 の場合、シークレットには以下のキーが含まれることが予想されます。

- REGISTRY_STORAGE_S3_ACCESSKEY
- REGISTRY_STORAGE_S3_SECRETKEY

手順

- 必要なキーが含まれる OpenShift Container Platform シークレットを作成します。

```
$ oc create secret generic image-registry-private-configuration-user --from-literal=REGISTRY_STORAGE_S3_ACCESSKEY=myaccesskey --from-literal=REGISTRY_STORAGE_S3_SECRETKEY=mysecretkey --namespace openshift-image-registry
```

3.1.2. ユーザーによってプロビジョニングされるインフラストラクチャーで AWS のレジストリーストレージを設定する

インストール時に、S3 バケットを作成するにはクラウド認証情報を使用でき、レジストリー Operator がストレージを自動的に設定します。

レジストリー Operator が S3 バケットを作成できず、ストレージを自動的に設定する場合、以下の手順により S3 バケットを作成し、ストレージを設定することができます。

前提条件

- ユーザーによってプロビジョニングされるインフラストラクチャーでの AWS 上のクラスター
- AWS ストレージの S3 の場合、シークレットには以下のキーが含まれることが予想されます。
  - **REGISTRY_STORAGE_S3_ACCESSKEY**
  - **REGISTRY_STORAGE_S3_SECRETKEY**

手順

レジストリー Operator が S3 バケットを作成できず、ストレージを自動的に設定する場合は、以下の手順を使用してください。

1. バケットライフサイクルポリシーを設定し、1 日以上経過している未完了のマルチパートアップロードを中止します。
2. **configs.imageregistry.operator.openshift.io/cluster** にストレージ設定を入力します。

```
$ oc edit configs.imageregistry.operator.openshift.io/cluster

storage:
  s3:
    bucket: <bucket-name>
    region: <region-name>
```

警告

AWS でレジストリーイメージのセキュリティーを保護するには、S3 バケットに対してパブリックアクセスのブロックを実行します。

3.1.3. AWS S3 のイメージレジストリー Operator 設定パラメーター

以下の設定パラメーターは AWS S3 レジストリーストレージで利用できます。

パラメーター	説明
bucket	バケットは、レジストリーのデータを保存するバケット名です。これはオプションであり、指定されていない場合は生成されます。
region	リージョンはバケットが存在する AWS リージョンです。これはオプションであり、インストール済みの AWS リージョンに基づいて設定されます。
regionEndpoint	RegionEndpoint は、S3 互換のストレージサービスのエンドポイントです。これは、指定されるリージョンに応じてオプションおよびデフォルトになります。
encrypt	encrypt は、イメージが暗号化された形式で保存されるかどうかを指定します。これはオプションであり、デフォルトは false です。
keyID	KeyID は、暗号化に使用する KMS キー ID です。これはオプションです。encrypt は true である必要があります。そうでない場合、このパラメーターは無視されます。
ImageRegistryConfigStorageS3CloudFront	CloudFront は Amazon Cloudfront をレジストリーでストレージミドルウェアとして設定します。これはオプションです。

3.2. GCP のユーザーによってプロビジョニングされるインフラストラクチャーでのレジストリーストレージの設定

3.2.1. イメージレジストリー Operator のシークレットの設定

configs.imageregistry.operator.openshift.io および ConfigMap リソースのほかにも、 **openshift-image-registry** namespace 内の別のシークレットリソースによって設定が Operator に提供されます。

image-registry-private-configuration-user シークレットは、ストレージのアクセスおよび管理に必要な認証情報を提供します。これは、デフォルト認証情報が見つからない場合に Operator によって使用されるデフォルト認証情報を上書きします。

GCP ストレージ上の GCS の場合、シークレットには、GCP が提供する認証情報ファイルの内容に相当するキーが含まれることが予想されます。

- REGISTRY_STORAGE_GCS_KEYFILE

手順

- 必要なキーが含まれる OpenShift Container Platform シークレットを作成します。

```
$ oc create secret generic image-registry-private-configuration-user --from-file=REGISTRY_STORAGE_GCS_KEYFILE=<path_to_keyfile> --namespace openshift-image-registry
```

3.2.2. ユーザーによってプロビジョニングされるインフラストラクチャーでの GCP のレジストリーストレージ

ストレージメディアは手動で設定し、レジストリー CRD で設定を行う必要があります。

前提条件

- ユーザーによってプロビジョニングされるインフラストラクチャーのある GCP 上のクラスター。
- GCP のレジストリーストレージを設定するには、レジストリー Operator クラウド認証情報を指定する必要があります。
- GCP ストレージ上の GCS の場合、シークレットには、GCP が提供する認証情報ファイルの内容に相当するキーが含まれることが予想されます。
  - **REGISTRY_STORAGE_GCS_KEYFILE**

3.2.3. GCP GCS のイメージレジストリー Operator 設定パラメーター

手順

以下の設定パラメーターは、GCP GCS レジストリーストレージに利用できます。

パラメーター	説明
bucket	バケットは、レジストリーのデータを保存するバケット名です。これはオプションであり、指定されていない場合は生成されます。
region	リージョンは、バケットが存在する GCS の場所です。これはオプションであり、インストールされている GCS リージョンに基づいて設定されます。

パラメーター	説明
projectID	ProjectID は、このバケットが関連付けられる必要がある GCP プロジェクトのプロジェクト ID です。これはオプションです。
keyID	KeyID は、暗号化に使用する KMS キー ID です。バケットは GCP でデフォルトで暗号化されているため、これはオプションになります。これにより、カスタム暗号化キーを使用できます。

3.3. ベアメタルの場合のレジストリーストレージの設定

3.3.1. ベアメタルの場合のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- クラスター管理者のパーミッション。
- ベアメタル上のクラスター。
- **ReadWriteMany** アクセスモードのプロビジョニングされた永続ボリューム (PV)(例: Red Hat OpenShift Container Storage)。
- 容量は「100Gi」以上である。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。
2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry
```

注記

ストレージタイプが **emptyDIR** の場合、レプリカ数が **1** を超えることはありません。ストレージタイプが **NFS** で、レジストリー Pod を **replica>1** を設定してスケールアップする必要がある場合、**no_wdelay** マウントオプションを有効にする必要があります。以下は例になります。

```
# cat /etc/exports
/mnt/data *(rw,sync,no_wdelay,no_root_squash,insecure,fsid=0)
sh-4.3# exportfs -rv
exporting *:/mnt/data
```

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io

storage:
  pvc:
    claim:
```

claim フィールドを空のままにし、 **image-registry-storage** PVC の自動作成を可能にします。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

3.3.2. 追加リソース

ベアメタルの場合のレジストリーストレージの設定方法についての詳細は、「Recommended configurable storage technology」を参照してください。

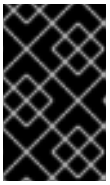
3.4. VSPHERE の場合のレジストリーストレージの設定

3.4.1. VMware vSphere の場合のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- クラスター管理者のパーミッション。
- VMware vSphere 上のクラスター。
- **ReadWriteMany** アクセスモードのプロビジョニングされた永続ボリューム (PV)(例: **NFS**)。



重要

vSphere ボリュームは **ReadWriteMany** アクセスモードをサポートしません。レジストリーストレージを設定するには、**NFS**などの異なるストレージバックエンドを使用する必要があります。

- 容量は「100Gi」以上である。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。
2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry
```

注記

ストレージタイプが **emptyDIR** の場合、レプリカ数が **1** を超えることはありません。ストレージタイプが **NFS** で、レジストリー Pod を **replica>1** を設定してスケールアップする必要がある場合、**no_wdelay** マウントオプションを有効にする必要があります。以下は例になります。

```
# cat /etc/exports
/mnt/data *(rw,sync,no_wdelay,no_root_squash,insecure,fsid=0)
sh-4.3# exportfs -rv
exporting *:/mnt/data
```

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io

storage:
  pvc:
    claim:
```

claim フィールドを空のままにし、 **image-registry-storage** PVC の自動作成を可能にします。

4. オプション: 新しいストレージクラスを PV に追加します。

   a. PV を作成します。

```
$ oc create -f -
```

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: image-registry-pv
spec:
  accessModes:
    ReadWriteMany
    capacity:
      storage: 100Gi
  nfs:
    path: /registry
    server: 172.16.231.181
  persistentVolumeReclaimPolicy: Retain
  storageClassName: nfs01
```

```
$ oc get pv
```

   b. PVC を作成します。

```
$ oc create -n openshift-image-registry -f -
```

```
apiVersion: "v1"
kind: "PersistentVolumeClaim"
metadata:
  name: "image-registry-pvc"
```

```
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: nfs01
  volumeMode: Filesystem
```

```
$ oc get pvc -n openshift-image-registry
```

最後に、PVC の名前を追加します。

```
$ oc edit configs.imageregistry.operator.openshift.io -o yaml
```

```
storage:
  pvc:
    claim: image-registry-pvc (1)
```

(1) カスタム PVC を作成すると、**image-registry-storage** PVC のデフォルトの自動作成の **claim** フィールドを空のままにすることができます。

5. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

3.4.2. 追加リソース

vSphere の場合のレジストリーストレージの設定方法についての詳細は、「Recommended configurable storage technology」を参照してください。

第4章 レジストリーオプション

OpenShift Container Platform はイメージをソースコードからビルドし、それらをデプロイし、それらのライフサイクルを管理できます。これを可能にするため、 OpenShift Container Platform は内部の統合コンテナーイメージレジストリーを提供しています。このレジストリーは OpenShift Container Platform 環境にデプロイでき、ここからイメージをローカルで管理できます。

4.1. 統合 OPENSHIFT CONTAINER PLATFORM レジストリー

OpenShift Container Platform は、クラスター上の標準ワークロードとして実行されるコンテナーイメージレジストリーでビルドを提供します。このレジストリーはインフラストラクチャー Operator によって設定され、管理されます。これはユーザーがワークロードを実行するイメージを管理するために追加設定なしで使用できるソリューションを提供し、既存のクラスターインフラストラクチャーの上部で実行されます。このレジストリーは、他のクラスターワークロードのようにスケールアップまたはスケールダウンでき、特定のインフラストラクチャーのプロビジョニングを必要としません。さらに、これはクラスターのユーザー認証および認可システムに統合されるため、イメージを作成し、取得するためのアクセスは、イメージリソースでユーザーのパーミッションを定義することによって制御できることを意味します。

通常、レジストリーはクラスター上にビルドされたイメージの公開ターゲットとして、またクラスター上で実行されるワークロードのイメージのソースとして使用されます。新規イメージがレジストリーにプッシュされると、クラスターにはその新規イメージについて通知され、他のコンポーネントは更新されたイメージに応答し、これを使用できます。

イメージデータは 2 つの場所に保存されます。実際のイメージデータは、クラウドストレージまたはファイルシステムボリュームなどの設定可能なストレージの場所に格納されます。標準のクラスター API によって公開され、アクセス制御を実行するために使用されるイメージメタデータは、標準的な API リソース、とくにイメージおよびイメージストリームとして保存されます。

4.2. サードパーティーレジストリー

OpenShift Container Platform はサードパーティーレジストリーからのイメージを使用してコンテナーを作成できますが、これらのレジストリーは統合 OpenShift Container Platform レジストリーと同じイメージ通知のサポートを提供する訳ではありません。このため、OpenShift Container Platform はイメージストリームの作成時にリモートレジストリーからタグをフェッチします。

フェッチされたタグの更新は、**oc import-image <stream>** を実行するだけで簡単に実行できます。新規イメージが検出されると、以前に記述されたビルドとデプロイメントの応答が生じます。

4.2.1. 認証

OpenShift Container Platform はユーザーが指定する認証情報を使用してプライベートイメージリポジトリーにアクセスするためにレジストリーと通信できます。これにより、OpenShift Container Platform はイメージのプッシュ/プルをプライベートリポジトリーへ/から実行できます。

4.3. RED HAT QUAY レジストリー

エンタープライズ向けの高品質なコンテナーイメージレジストリーを必要とされる場合、Red Hat Quay をホストされたサービスとして、また独自のデータセンターやクラウド環境にインストールするソフトウェアとしてご利用いただけます。Red Hat Quay の高度なレジストリーには、geo レプリケーション、イメージのスキャニング、およびイメージのロールバック機能が含まれます。

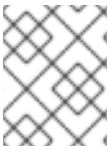
Quay.io サイトにアクセスし、独自のホストされる Quay レジストリーアカウントをセットアップします。その後、Quay チュートリアルに従って Quay レジストリーにログインし、イメージの管理を開始します。

Red Hat Quay レジストリーへのアクセスは、任意のリモートコンテナーイメージレジストリーと同様に OpenShift Container Platform から実行できます。

4.4. 認証で有効にされる RED HAT レジストリー

Red Hat Container Catalog で利用可能なすべてのコンテナーイメージはイメージレジストリーの **registry.redhat.io** でホストされます。

レジストリー **registry.redhat.io** では、イメージおよび OpenShift Container Platform でホストされるコンテンツへのアクセスに認証が必要です。新規レジストリーへの移行後も、既存レジストリーはしばらく利用可能になります。



注記

OpenShift Container Platform はイメージを **registry.redhat.io** からプルするため、これを使用できるようにクラスターを設定する必要があります。

新規レジストリーは、以下の方法を使用して認証に標準の OAuth メカニズムを使用します。

- **認証トークン。**管理者によって生成されるこれらのトークンは、システムにコンテナーイメージレジストリーに対する認証機能を付与するサービスアカウントです。サービスアカウントはユーザーアカウントの変更による影響を受けないため、トークンの認証方法は信頼性があり、回復性があります。これは、実稼働クラスター用にサポートされている唯一の認証オプションです。

- **Web ユーザー名およびパスワード**。これは、**access.redhat.com** などのリソースへのログインに使用する標準的な認証情報のセットです。OpenShift Container Platform でこの認証方法を使用することはできますが、これは実稼働デプロイメントではサポートされません。この認証方法の使用は、OpenShift Container Platform 外のスタンドアロンのプロジェクトに制限されます。

ユーザー名およびパスワード、または認証トークンのいずれかの認証情報を使用して **podman login** を使用し、新規レジストリーのコンテンツにアクセスします。

すべてのイメージストリームは新規レジストリーを参照します。レジストリーにはアクセスするために認証が必要であるため、Samples Operator は **samples-registry-credentials** シークレットを作成します。

認証情報は 2 つの場所に配置する必要があります。

- **OpenShift namespace**。OpenShift namespace のイメージストリームがインポートできるように、認証情報は OpenShift namespace になければなりません。

- **ホスト**。Kubernetes でイメージをプルする際にホストの認証情報を使用するため、認証情報はホスト上になければなりません。

第5章 レジストリーへのアクセス

ログおよびメトリクスの表示やレジストリーのセキュリティー保護および公開などの、レジストリーへのアクセスについての各種の方法について、以下のセクションを参照してください。

レジストリーに直接アクセスし、**podman** コマンドを起動することが可能です。これにより、**podman push** や **podman pull** などの操作で統合レジストリーへ/からイメージを直接プッシュまたはプルすることができます。これを実行するには、**oc login** コマンドを使ってレジストリーにログインしている必要があります。実行できる操作は、以下のセクションで説明されているようにユーザーが持つパーミッションによって異なります。

前提条件

- アイデンティティープロバイダー (IDP) を設定しておく必要があります。
- **podman pull** コマンドを使用する場合などにイメージをプルするには、ユーザーに **registry-viewer** ロールがなければなりません。このロールを追加するには、以下を実行します。

```
$ oc policy add-role-to-user registry-viewer <user_name>
```

- イメージの書き出しやプッシュを実行するには (**podman push** コマンドを使用する場合など)、ユーザーに **registry-editor** ロールが必要です。このロールを追加するには、以下を実行します。

```
$ oc policy add-role-to-user registry-editor <user_name>
```

5.1. クラスターからレジストリーへの直接アクセス

クラスター内からレジストリーにアクセスすることができます。

手順

内部ルートを使用して、クラスターからレジストリーにアクセスします。

1. ノードのアドレスを取得することにより、ノードにアクセスします。

```
$ oc get nodes
$ oc debug nodes/<node_address>
```

2. アクセストークンを使用してコンテナーイメージレジストリーにログインします。

```
$ oc login -u kubeadmin -p <password_from_install_log>
$ podman login -u kubeadmin -p $(oc whoami -t) image-registry.openshift-image-registry.svc:5000
```

以下のようなログインを確認するメッセージが表示されるはずです。

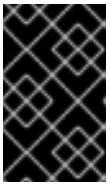
```
Login Succeeded!
```

注記

ユーザー名には任意の値を指定でき、トークンには必要な情報がすべて含まれます。コロンが含まれるユーザー名を指定すると、ログインに失敗します。

イメージレジストリー Operator はルートを作成するため、 **default-route-openshift-image-registry.<cluster_name>** のようになります。

3. レジストリーに対して **podman pull** および **podman push** 操作を実行します。



重要

任意のイメージをプルできますが、**system:registry** ロールを追加している場合は、各自のプロジェクトにあるレジストリーにのみイメージをプッシュすることができます。

次の例では、以下を使用します。

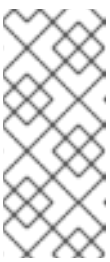
Component	値
<registry_ip>	**172.30.124.220**
<port>	**5000**
<project>	**openshift**
<image>	**image**
<tag>	省略 (デフォルトは **latest**)

a. 任意のイメージをプルします。

```
$ podman pull name.io/image
```

b. 新規イメージに **<registry_ip>:<port>/<project>/<image>** 形式でタグ付けします。プロジェクト名は、イメージを正しくレジストリーに配置し、これに後でアクセスできるようにするために OpenShift Container Platform のプル仕様に表示される必要があります。

```
$ podman tag name.io/image image-registry.openshift-image-
registry.svc:5000/openshift/image
```



注記

指定されたプロジェクトについて **system:image-builder** ロールを持っている必要があります。このロールにより、ユーザーはイメージの書き出しやプッシュを実行できます。このロールが設定されていない場合には次の手順の **podman push** が失敗します。 新規プロジェクトを作成し、イメージをプッシュしてテストできます。

c. 新しくタグ付けされたイメージをレジストリーにプッシュします。

```
$ podman push image-registry.openshift-image-registry.svc:5000/openshift/image
```

5.2. レジストリーの内容の表示

管理者として、レジストリーの内容を確認することができます。

前提条件

- 管理者としてログインします。

手順

1. プロジェクト **openshift-image-registry** の下で Pod を確認します。

```
# oc get pods
NAME READY STATUS RESTARTS AGE
cluster-image-registry-operator-764bd7f846-qqtpb 1/1 Running 0 78m
image-registry-79fb4469f6-llrln 1/1 Running 0 77m
node-ca-hjksc 1/1 Running 0 73m
node-ca-tftj6 1/1 Running 0 77m
node-ca-wb6ht 1/1 Running 0 77m
node-ca-zvt9q 1/1 Running 0 74m
```

5.3. レジストリーログの表示

oc logs コマンドを使用してレジストリーのログを表示することができます。

手順

1. デプロイメントで **oc logs** コマンドを使用して、コンテナーイメージレジストリーのログを表示します。

```
$ oc logs deployments/image-registry
2015-05-01T19:48:36.300593110Z time="2015-05-01T19:48:36Z" level=info
msg="version=v2.0.0+unknown"
2015-05-01T19:48:36.303294724Z time="2015-05-01T19:48:36Z" level=info msg="redis not
configured" instance.id=9ed6c43d-23ee-453f-9a4b-031fea646002
2015-05-01T19:48:36.303422845Z time="2015-05-01T19:48:36Z" level=info msg="using
inmemory layerinfo cache" instance.id=9ed6c43d-23ee-453f-9a4b-031fea646002
2015-05-01T19:48:36.303433991Z time="2015-05-01T19:48:36Z" level=info msg="Using
OpenShift Auth handler"
2015-05-01T19:48:36.303439084Z time="2015-05-01T19:48:36Z" level=info msg="listening
on :5000" instance.id=9ed6c43d-23ee-453f-9a4b-031fea646002
```

5.4. レジストリーメトリクスへのアクセス

OpenShift Container レジストリーは、Prometheus メトリクスのエンドポイントを提供します。Prometheus はスタンドアロンのオープンソースのシステムモニタリングおよびアラートツールキットです。

メトリクスは、レジストリーエンドポイントの **/extensions/v2/metrics** パスに公開されます。

手順

メトリクスクエリーの実行またはクラスターロールの使用という、メトリクスにアクセスするための 2 つの方法を使用できます。

メトリクスクエリー

1. 以下のようにメトリクスクエリーを実行します。

```
$ curl --insecure -s -u <user>:<secret> \ 1
    https://image-registry.openshift-image-registry.svc:5000/extensions/v2/metrics | grep
imageregistry | head -n 20
# HELP imageregistry_build_info A metric with a constant '1' value labeled by major, minor,
git commit & git version from which the image registry was built.
# TYPE imageregistry_build_info gauge
imageregistry_build_info{gitCommit="9f72191",gitVersion="v3.11.0+9f72191-135-
dirty",major="3",minor="11+"} 1
# HELP imageregistry_digest_cache_requests_total Total number of requests without scope
to the digest cache.
# TYPE imageregistry_digest_cache_requests_total counter
imageregistry_digest_cache_requests_total{type="Hit"} 5
imageregistry_digest_cache_requests_total{type="Miss"} 24
# HELP imageregistry_digest_cache_scoped_requests_total Total number of scoped
requests to the digest cache.
# TYPE imageregistry_digest_cache_scoped_requests_total counter
imageregistry_digest_cache_scoped_requests_total{type="Hit"} 33
imageregistry_digest_cache_scoped_requests_total{type="Miss"} 44
# HELP imageregistry_http_in_flight_requests A gauge of requests currently being served by
the registry.
# TYPE imageregistry_http_in_flight_requests gauge
imageregistry_http_in_flight_requests 1
# HELP imageregistry_http_request_duration_seconds A histogram of latencies for requests
to the registry.
# TYPE imageregistry_http_request_duration_seconds summary
imageregistry_http_request_duration_seconds{method="get",quantile="0.5"} 0.01296087
imageregistry_http_request_duration_seconds{method="get",quantile="0.9"} 0.014847248
imageregistry_http_request_duration_seconds{method="get",quantile="0.99"} 0.015981195
imageregistry_http_request_duration_seconds_sum{method="get"} 12.260727916000022
```

1 **<user>** は任意ですが、**<secret>** はレジストリー設定で指定された値と一致していなければなりません。

クラスターロール

1. メトリクスにアクセスするために必要なクラスターロールがない場合、これを作成します。

```
$ cat <<EOF |
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: prometheus-scraper
rules:
- apiGroups:
  - image.openshift.io
  resources:
```

```
  - registry/metrics
  verbs:
  - get
EOF
$ oc create -f -
```

2. このロールをユーザーに追加し、以下のコマンドを実行します。

```
$ oc adm policy add-cluster-role-to-user prometheus-scraper <username>
```

3. クラスターロールを使用してメトリクスにアクセスします。設定ファイルのメトリクスに対応する部分は以下のようになります。

```
openshift:
  version: 1.0
  metrics:
    enabled: true
...
```

追加リソース

- **kubeadmin** は削除されるまでレジストリーにアクセスできます。詳細は、「Removing the kubeadmin user」を参照してください。
- アイデンティティープロバイダーの設定についての詳細は、「Understanding identity provider configuration」を参照してください。

第6章 レジストリーの公開

デフォルトで、OpenShift Container Platform レジストリーのセキュリティーは、TLS 経由でトラフィックを送信できるようにクラスターのインストール時に保護されます。以前のバージョンの OpenShift Container Platform とは異なり、レジストリーはインストール時にクラスター外に公開されません。

6.1. セキュアなレジストリーの手動による公開

クラスター内から OpenShift Container Platform レジストリーにログインするのではなく、外部からレジストリーにアクセスできるように、このレジストリーをルートに公開します。この方法を使うと、ルートアドレスを使ってクラスターの外部からレジストリーにログインし、ルートのホストを使ってイメージにタグ付けしたり、イメージをプッシュしたりできます。

前提条件

- 以下の前提条件は自動的に実行されます。
  - レジストリー Operator をデプロイします。
  - Ingress Operator をデプロイします。

手順

configs.imageregistry.operator.openshift.io リソースで **DefaultRoute** パラメーターを使用するか、またはカスタムルートを使用してルートを公開することができます。

DefaultRoute を使用してレジストリーを公開するには、以下を実行します。

1. **DefaultRoute** を **True** に設定します。

```
$ oc patch configs.imageregistry.operator.openshift.io/cluster --patch '{"spec":
{"defaultRoute":true}}' --type=merge
```

2. Podman でログインします。

```
$ HOST=$(oc get route default-route -n openshift-image-registry --template='{{ .spec.host }}')
$ podman login -u $(oc whoami) -p $(oc whoami -t) --tls-verify=false $HOST (1)
```

(1) **--tls-verify=false** は、ルートのクラスターのデフォルト証明書が信頼されない場合に必要になります。Ingress Operator で、信頼されるカスタム証明書をデフォルト証明書として設定できます。

カスタムルートを使用してレジストリーを公開するには、以下を実行します。

1. ルートの TLS キーでシークレットを作成します。

```
$ oc create secret tls public-route-tls \
    -n image-registry \
    --cert=</path/to/tls.crt> \
    --key=</path/to/tls.key>
```

この手順はオプションです。シークレットを作成しない場合、ルートは Ingress Operator からデフォルトの TLS 設定を使用します。

2. レジストリー Operator では、以下のようになります。

```
spec:
  routes:
    - name: public-routes
      hostname: myregistry.mycorp.organization
      secretName: public-route-tls
...
```

レジストリーのルートのカスタム TLS 設定を指定している場合は **secretName** のみを設定します。