# Alibaba Cloud
# ACS 容器服#

# FAQ

**Document Version20191101**

# 目次

# 1 よくある質問

## 1.1 Container Service とは何ですか。

**Container Service** は高性能かつ拡張可能なコンテナーアプリケーション管理サービスを提供します。これにより、**Docker** や **Kubernetes** を用いてコンテナー化されたアプリケーションのライフサイクルの管理が 可能になります。 **Container Service** は複数のアプリケーションリリース方法および継続的なデリバリー機能を提供し、マイクロサービスアーキテクチャをサポートします。 コンテナー管理クラスターの設定を簡素化し、**Alibaba Cloud** の仮想化、ストレージ、ネットワークおよびセキュリティの機能を統合することで、**Container Service** によりコンテナーの理想的なクラウド実行環境を構築できます。

## 1.2 Container Service がサポートするコンテナータイプは何ですか。

**Container Service** は現在、 **Docker** および **Kubernetes** をサポートしています。

## 1.3 Container Service でのアプリケーションおよびサービスはどのようなものですか。

**1** つのアプリケーションをさまざまなマイクロサービスに分割できます。 それぞれのマイクロサービスが同じイメージと設定のコンテナー群で構成され、アトミック機能を持ち、互いに接続しています。

前述のマイクロサービスは **Container Service** ではサービスとして知られています。
**Container Service** では **1** つ以上のサービスがアプリケーションを構成しています。

コンテナーを利用してアプリケーションを管理するために、最初にイメージまたはオーケストレーションテンプレートによりアプリケーションを作成します。 イメージを使って作成されたアプリケーションは **1** つのサービスを構成します。 オーケストレーションテンプレートを使って作成されたアプリケーションは、オーケストレーションテンプレートにより指定された **1** つ以上のサービスを構成します。

## 1.4 Container Service は有料ですか。

1. 現在、**Container Service** では、**ECS (Elastic Compute Service)** インスタンスおよび **Server Load Balancer** インスタンスなどのお使いのリソースの料金以外は請求されません。
2. **Container Service** では、自動または手動で作成された **ECS** インスタンスや **Server Load Balancer** インスタンスは、それぞれ **ECS** および **Server Load Balancer** の料金に応じて課金されます。 詳しくは、「*ECS* 課金方法」および課金方法をご参照ください。

## 1.5 コンテナー起動するときにアプリケーションの作成が必要なのはなぜですか。

アプリケーションは **Container Service** によりユーザーに提示された管理概念であり、イメージまたはオーケストレーションテンプレートを使用して作成されます。

複雑なアプリケーションを異なるコンポーネントに分割することを推奨します。また、**Container Service** はコンポーネントのプロパティやコンポーネント間の接続に有用です。

各コンポーネントが同じイメージと設定を持つコンテナ群で構成され、**Container Service** ではサービスと称されています。

## 1.6 Container Service は外部の Docker イメージをサポートしていますか。

**Container Service** では、サーバーのセキュリティポリシーにより許可されている **Docker** イメージソースに関しては制限を設けていません。

# 2 Swarm に関するよくある質問

## 2.1 オーケストレーションテンプレートでクラスターの各ノードにどのようにサービスをデプロイしますか。

アプリケーションの作成に使用されるオーケストレーションテンプレートに応じて、それぞれのノードにテンプレートでサービスをデプロイする方法が 2 つあります。

**Compose V1/V2: Alibaba Cloud Container Service** により提供される拡張機能ラベル **"global"** を使用します。

Compose V1/V2

**Alibaba Cloud Container Service** により提供される拡張機能のラベル `global` を使用することでグローバルサービスとしてサービスを設定できます。

サービスが `aliyun.global: true` として設定された場合、 このサービスはクラスターのそれぞれのノードにデプロイされます。 コンテナーは自動的に新しくクラスターに追加されたノードにデプロイされます。

> 注:
> `global` ラベルについて詳しくは、*#unique_11*をご参照ください。

例

```
monitor:
  image: sample
  labels:
    aliyun.global: true
```

## 2.2 再デプロイと再スケジューリングの違いはなんですか。

**再デプロイ**

再デプロイは、アプリケーションイメージを使用してアプリケーションを再デプロイすることです。

次のような場合に再デプロイできます。

- アプリケーションのデプロイ後にアプリケーションイメージを更新し、更新したイメージを使ってアプリケーションをデプロイする場合。

- 停止または削除したコンテナーを起動、あるいは再作成する場合。 アプリケーションを再デプロイする場合、**Container Service** は停止したコンテナーを起動し、削除されたコンテナー再作成します。

**再スケジューリング**

再スケジューリングは、それぞれのノードで実行中のコンテナー数を調整します。高負荷のノードから新しく追加されたノードまたは低負荷のノードへのコンテナーを移動することで、クラスターの負荷が調整されます。

> 注：
> 再スケジューリングはノードでのコンテナーの分配のみを変更し、アプリケーションの再デプロイのためにイメージをプルしません。

詳しくは、*#unique_13*をご参照ください。

## 2.3 アプリケーションの再デプロイが有効にならなかったらどうなりますか。

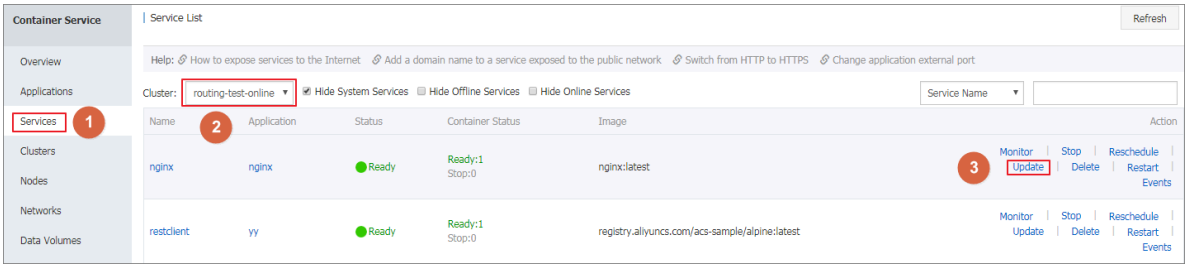
1. イメージ **sha265** を参照して、再デプロイ後のイメージが最新のものであるかを確認します。以下の手順に従い、イメージ **sha256** を確認します: *Container Service コンソール*にログインします。 左側のナビゲーションウィンドウから [アプリケーション] をクリックします。 アプリケーションの属するクラスターを [クラスター] リストから選択します。 アプリケーション名をクリックします。 [コンテナー] タブをクリックします。 イメージを確認します。 イメージが最新のものである場合、再デプロイが成功しています。
2. データボリュームをホストにマウントしているかどうかを確認します。 再デプロイによりデータボリュームは更新されず、ホスト上の古いデータボリュームが使用されます。 そのため、新しいイメージに加えたデータボリューム設定の変更は、 アプリケーションの再デプロイ後には反映されません。

> 注：
> 再デプロイについて詳しくは、*#unique_15*をご参照ください。

## 2.4 どのようにアプリケーションの外部ポートを公開または変更できますか。

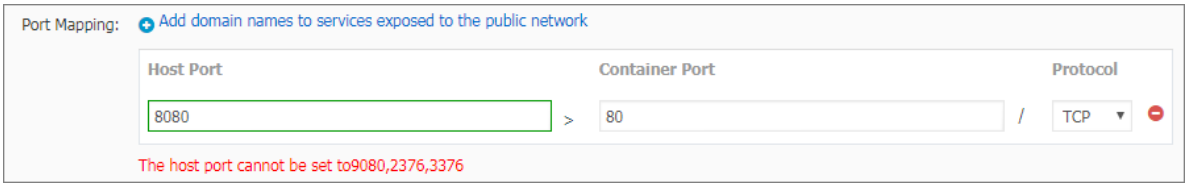
1. *Container Service コンソール*にログインします。
2. 左側のナビゲーションウィンドウから [サービス] をクリックします。

3. [クラスター] リストからクラスターを選択します。
4. サービス (この例では **"nginx"**) の右側にある [更新] をクリックします。



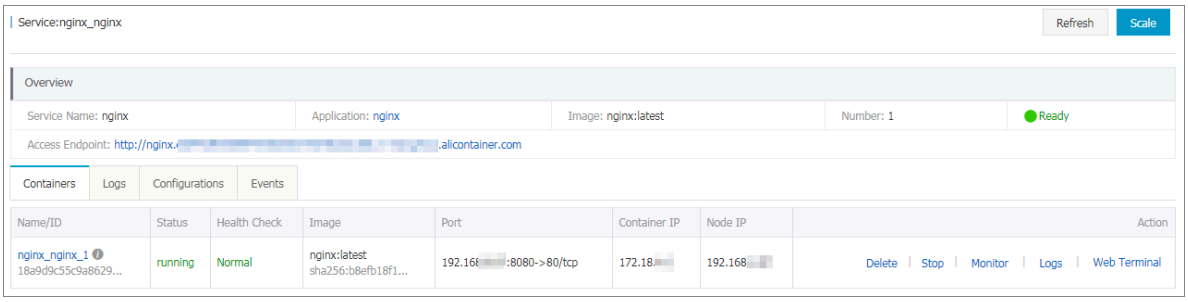

5. [ポートマッピング] でマッピングするホストポートを入力します。

   複数のポートを公開するには、"+" アイコンをクリックして、複数のホストポートとコンテナーポートを入力します。




6. [更新] をクリックします。
7. サービス名 (この例では **"nginx"**) をクリックします。

   ホストポートがコンテナーポートにマッピングされ、**Telnet** を利用したポート接続が成功します。




## 2.5 Container Service のネットワークモデルはどのようにホストコンテナー間で通信していますか。

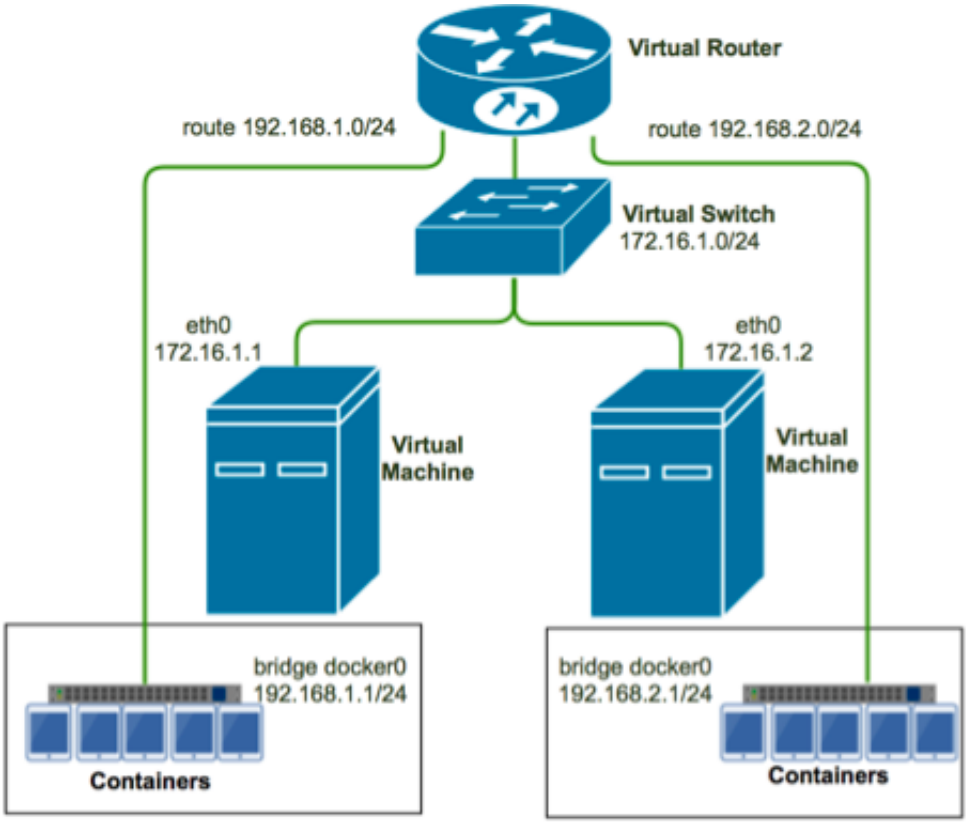
### コンテナー間の相互作用

**Container Service** では、クラスター内の各コンテナーに独立した **IP** アドレスが割り振られ、クラスター内で通信します。 コンテナーは、**NAT (Network Address Translation)** により公開されるホストポートではなく、個別の **IP** アドレスを使用して相互に通信できます。 そのため、ホストの **IP** アドレスに依存しなくなり、 **NAT** 設定時に複数のコンテナー間でポートが競合

することがありません。 以下では、異なるネットワークモデルのホストコンテナー同士を通信させる方法について解説します。

**VPC (Virtual Private Cloud)** では

**VPC** は **Alibaba Cloud** を基に、切り離されされたネットワーク環境の構築に有用です。 空き **IP** アドレスの範囲から、**CIDR (Classless Inter-Domain Routing)** ブロックの分割、ルートテーブルやゲートウェイの設定まで、 仮想ネットワークのすべてを管理できます。 **VPC** ルートテーブルを設定することで、**Container Service** はコンテナー **IP** アドレス範囲に該当する **ECS (Elastic Compute Service)** インスタンスにコンテナー間アクセス要求を転送します。 以下をご参照ください。




クラスターノード **(172.16.1.1)** で **Docker daemon** を起動し、ブリッジネットワークのデフォルトの **IP** アドレス範囲を **"192.168.1.0/24"** に設定します。 もう一方のノード **(172.16.1.2)** で **Docker daemon** を起動し、ブリッジネットワークのデフォルトの **IP** アドレス範囲を **"192.168.2.0/24"** に設定します。 **VPC** 内の **VRouter** ルートテーブルで該当するルーティングルールを設定し、 **"192.168.1.0/24"** からのアクセスリクエストをノード **"172.16.1.1"** に転送します。 もう 1 つのノードに対しても同様のルーティングルールを設定します。

ノード **1** の **IP** アドレス **"192.168.1.2"** を持つコンテナーが、ノード **2** の **IP** アドレス **"192.168.2.2"** を持つコンテナーにアクセスした場合、 アクセスリクエストがルートテーブルによって該当するマシンへ転送されます。 アクセスリクエストはその後、 **Docker** により作成されたルー

ティングルールを基に **Docker0** のブリッジに転送されます。 最終的に、リクエストは **IP** アドレス **"192.168.2.2"** のコンテナーに転送されます。

なお、**Container Service** は **VPC** 内のコンテナーに対して個別の **CIDR** ブロックおよびルートエントリーを割り当てます。 これにより、元の **VSwitch CIDR** ブロック、ルートテーブルエントリーやマシンの **IP** ルートテーブルとの競合を避けることができます。 そうでなければ、アクセスリクエストは正しくコンテナーに転送されません。

クラシックネットワークにおいて

**Docker 1.9** およびそれ以降のバージョンでは、ネイティブの**VXLAN** プロトコルに基づく『ホスト間コンテナーネットワーク』をサポートしています。 クラシックネットワークでは、**Container Service** は、 **Docker Overlay Network** に基づいて、クラスター内のコンテナー間通信が実現するネットワーク環境を構築します。 **Docker Overlay Network** を仮想化したマルチホストコンテナーネットワークは、仮想化サブネットが同じため、異なるホストのコンテナー同士が通信できます。

### ノード間リンク

複数のコンテナーアプリケーションでは、しばしば『リンク』を使ってコンテナー間の依存関係が説明されます。 たとえば、**WordPress Web** サービスは **MySQL** データベースサービスに依存しています。 そのため、**WordPress** コンテナーが起動すると、リンクを使用して、データベース接続のための **IP** アドレスやポートを含む一連の **MySQL** コンテナーのパラメーターが取得されます。

しかし、**Docker** リンクは同じホストノードでのコンテナー接続のみサポートしています。一方で、**Container Service** は異なるノード間のコンテナー接続をサポートしています。 コンテナー **IP** アドレスが変更された場合、リンクのコンテナーエイリアスも変更されます。 ノード単独でリンクを使った場合も まったく同じように動作します。

### コンテナーから仮想マシンへのアクセス

**Container Service** のコンテナーは外部ネットワークアクセスのためのルートを保持しています。 そのため、コンテナーがサービスまたは仮想マシンの **IP** アドレスにアクセスする必要がある場合、**IP** アドレスや仮想マシンドメイン名がそのまま使用されます。

### 参考資料

- *Alibaba Cloud VPC* サービス
- マルチホストネットワークの 始め方
- *Docker* コンテナーネットワークを 理解する
- *Docker* コンテナーリンク

## 2.6 ルーティングサービスに関するよくある質問

**Q: Container Service** で、**Web** コンテナーを簡単に素早くデプロイし、接続する方法を教えてください。

**A:** 簡単に素早くアプリケーションをデプロイする方法については、*#unique_20*および*#unique_21*をご参照ください。

**Q:** どのようにルーティングサービスを使用しますか。

**A:**「*シンプルルーティング - HTTP および HTTPS のサポート*」をご参照ください。

**Q:** どのようにルーティングラベルを使用しますか。

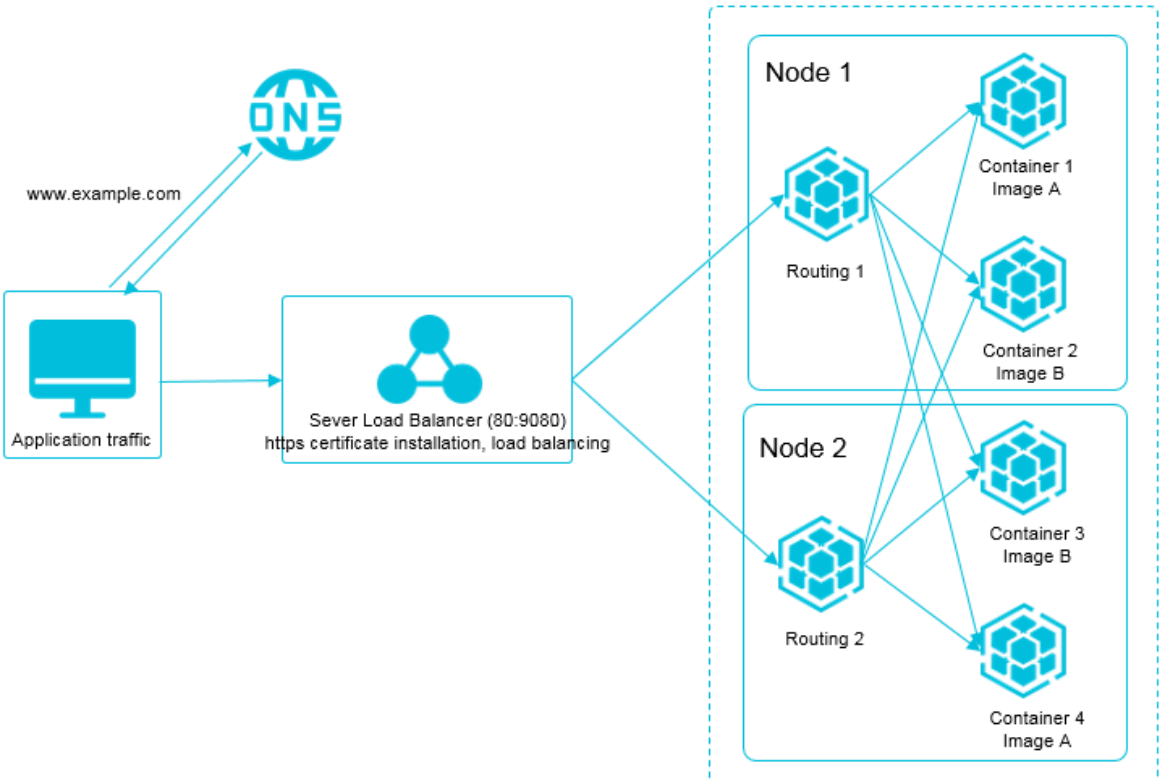
**A:** サービスオーケストレーションドキュメント*#unique_11*をご参照ください。

**Q:** どのようにデプロイ済みの **Web** コンテナーのコンテンツにアクセスしますか。

**A:** コンテナーはプロセスやプロセスの集まりです。 プロセスがポートを公開している場合にのみ、コンテナーのコンテンツにアクセスできます。 コンテナーはコンテナーポートを抜き出し、コンテナーポートをホストポートにマッピングします。 コンテナーポートにマッピングされたホストポートを介してコンテナーにアクセスできます。

**Q:** 高可用性を目的として、同じ機能を持つ複数のコンテナーに対して **1** つのアクセスエンドポイントのみ提供されます。 **Container Service** ではこれがどのように実装されていますか。

**A:** 以下の図に示すように、ルーティングサービスが提供されています。ルーティングサービスは、 サービス > 更新 > **Web** ルーティングで設定します。 デフォルトでは、ルーティングサービスはクラスター内のそれぞれのノードにルーティングコンテナーをデプロイします (クラスターが作成され、**acsrouting** アプリケーションに属すると、ルーティングコンテナーはクラスターリストに 表示されます)。 クラスターの作成後、**Server Load Balancer** インスタンスはデフォルトで作成されます。 すべてのアクセスリクエストは、クラスター **Server Load Balancer** のフロントエンドポート**80** > ノードのポート **9080** > ルーティングコンテナーのポート **80** を経由します。 ルーティングコンテナーは、**Nginx** に似たサーバー負荷分散ソフトウェアの **HAProxy** を搭載しており、 負荷を分散します。 ルーティングコンテナーはアクセスリクエストを、**HTTP** で **"HOST"** ヘッダーにより指定されるドメイン名に基づいた他のコンテナーバックエンドに転送します (クラスター内のコンテナーは相互に作用しています)。 ルーティング設定の際は、**Server Load Balancer** ポート、ノード仮想マシン **(VM)** ポートおよびコンテナーポートの間の 違いと関係に注意を払います。




**Q:** パブリックネットワークに公開されているサービスにドメイン名を追加し、 サービスを **HTTP** に対応させる方法を教えてください。

**A:**『シンプルルーティング - ドメイン名の設定』をご参照ください。

**Q: HTTP** から **HTTPS** にどのようにプロトコルを変更しますか。

**A:**『シンプルルーティング - *HTTP* から *HTTPS* への変更』をご参照ください。

**Q:** どのように**HTTP** 要求を **HTTPS** 要求にリダイレクトしますか。

**A:** 要求プロトコルが **HTTP** か**HTTPS** であるかを判別します。 プロトコルが **HTTP** である場合、要求は **301** または **302** により **HTTPS** にリダイレクトされます。 ただし、**HTTP** 要求および **HTTPS** 要求が それぞれ**Server Load Balancer** の ポート **80** および ポート **443** を使って同じバックエンドポート (たとえば、ポート **9080** ) に転送される場合、 バックエンドコンテナーは要求プロトコルが **HTTP** か **HTTPS** なのか 判別できません。 この問題を解決するには、新たにポート (たとえば、ポート **8080** ) を公開して、**HTTP** 要求が次の経路を辿るようにします**: Server Load Balancer** のフロントエンドのポート **80** > ノードホスト **VM** の ポート **8080** > リダイレクト専用のコンテナー、 たとえば、**Nginx** ポート **80** > **302** により **HTTPS** へリダイレクト。

**Q:** なぜ **Server Load Balancer** の **HTTP** ポートで例外が報告されるのですか。

**A: Server Load Balancer** の **HTTP** ポートでのヘルスチェックが失敗した場合、例外が報告されます。 ヘルスチェックは原則、 **GET**要求に似た**HTTP HEAD** 要求を送り、 レスポンスヘッダーのみが返ってくることを求めます。 ドメイン名は **HTTP** 要求に対して設定される必要があり、デフォルト値は **IP** アドレスです。 **Server Load Balancer** は、要求に対してステータスコード **200** が返されると、 ヘルスチェックが成功したとみなします。 **Server Load Balancer** を使わず、直接ホストノードで*curl*コマンドを実行し、要求に対してステータスコード **200** が返ってくることを確認します。

**Q:** なぜ **Server Load Balancer** の **HTTPS** ポートで例外が報告されるのですか。

**A: Server Load Balancer** の **HTTPS** ポートでのヘルスチェックが失敗した場合、例外が報告されます。 ヘルスチェックは原則、 **GET**要求に似た**HTTP HEAD** 要求を送り、 レスポンスヘッダーのみが返ってくることを求めます。 ドメイン名は **HTTP** 要求に対して設定される必要があり、デフォルト値は **IP** アドレスです。 **Server Load Balancer** は、要求に対してステータスコード **200** が返されるとヘルスチェックが成功したとみなします。 **HTTPS** ポートでは、 ヘルスチェックの設定時に要求ドメイン名が設定されている必要があります。 設定されていない場合、ヘルスチェックがデフォルトで失敗します。 (デフォルト **IP** アドレスの要求はルーティングコンテナーに転送されますが、ルーティングコンテナーはどのバックエンドに対して要求を転送すればいいのかわからないため、エラー **503** を返します)。 **Server Load Balancer** を使わずに、直接ホストノードで *curl*コマンドを実行し、要求に対しステータスコード **200** が返ってくるかを確認します。 ステータスコード **200** が返されない場合、アプリケーションの有効性を確認します。

**Q:** クラスターはイントラネット **Server Load Balancer** インスタンスのバインドまたはバインド解除をサポートしていますか。

**A:** クラスターは **1** つの **Server Load Balancer** インスタンスしかバインドできません。クラスターから **Server Load Balancer** インスタンスからバインド解除することはできます。 詳しくは、*#unique_25*をご参照ください。

**Q:** クラスターは複数のクラスターレベル **Server Load Balancer** インスタンスをバインドすることができますか。

**A:** 現在は対応していません。 手動で **Server Load Balancer** を作成してから、クラスターノードの ポート **9080** にバインドできます。 ノードが拡張された場合、バックエンドサーバーの数を増やしたり減らしたりして、ご自身で作成した **Server Load Balancer** インスタンスのバックエンドサーバーを管理する必要があります。

**Q:** クラスター内のコンテナー間通信はどのように行われますか。

**A:** コンテナーが、同じクラスターにある他のコンテナーにアクセスする場合、コンテナー名を内部ドメイン名として使用できます。

**Q:** クラスター内のコンテナー同士のサービス検出や**Server Load Balancer** はどのように実装されていますか。

**A:** 転送や検出にルーティングサービスプロキシが使用されます。「クラスター内サービス間のルーティングと *Server Load Balancer*」をご参照ください。

**Q:** ルーティングサービスの接続に関するトラブルシューティングを教えてください。

**A:**どのようにリンク接続問題のトラブルシューティングを行いますか。をご参照ください。

**Q:** オーケストレーションテンプレートで `aliyun.routing.port_$container_port` を使用するか、 サービス設定を更新することで **Web** ルーティングルールを設定できます。 この **2** つの方法の違いはなんですか。

**A: 2** つの方法は本質的には同じです。 オーケストレーションテンプレートで `aliyun.routing.port_$container_port` を使用することで設定された **Web** ルーティングルールは、 **[**サービスの更新**]** ページでの **Web** ルーティングルールで反映されます。 しかし、サービス設定で **Web** ルーティングルールを変更した場合、オーケストレーションテンプレートには反映されません。 **Web** ルーティングルールを **[**サービスの更新**]** で設定することで、コンソールでの操作、問題のトラブルシューティング、およびエラーの確認が簡単になります。 この **Web** ルーティングルール設定フォームはオーケストレーションテンプレートのラベルに変換され、 サービス設定の更新に使用されます。

**Q:** デフォルトのルーティングサービスが事例に合わない場合はどうしたらいいですか。

**A:** カスタムプロキシイメージ*registry.aliyuncs.com/acs/proxy*がよい解決法になります。 **HAProxy** に基づいたイメージであり、**HAProxy** を定義するパラメーター設定をサポートしています。 このイメージはダイナミックサービス検出にも対応しており、サービスのヘルスステータスを基に正常なコンテナーにサービスが転送されます。

**Q:** シンプルルーティングを使用後、クライアントの実 **IP** アドレスをどのように取得できますか。

**A:** シンプルルーティングを使用するすべての要求に対して、**Container Service** は要求ヘッダーに **"x-forwarded-for"** 情報を追加します。

```
x-forwarded-for: <Client IP address>
x-forwarded-for: <Proxy server IP>
```

注：

ヘッダーは複数の行を含むことがあります。 クライアントの実 IP アドレスは 最初の行の"x-forwarded-for" から取得できます。

## 2.7 カスタム Server Load Balancer に関するよくある質問

**Q:** カスタム **Server Load Balancer** のシナリオはどのようなものですか。

**A:** カスタム **Server Load Balancer** は以下のようなシナリオで使用できます。

- レイヤー 7 プロトコル **Server Load Balancer** で、それぞれのサービスにルートがカスタマイズされている場合。 従来のアーキテクチャがコンテナーアーキテクチャに移行されると、非コンテナークラスターのサービスがコンテナークラスターのコンテナーサービスにアクセスします。
- レイヤー 4 プロトコル **Server Load Balancer** で、それぞれのサービスにルートがカスタマイズされている場合。 従来のアーキテクチャがコンテナーアーキテクチャに移行されると、非コンテナークラスターのサービスがコンテナークラスターのコンテナーサービスにアクセスします。
- イントラネット **Server Load Balancer** インスタンスが **Container Service** での通信に使用されます。

**Q:** どのようにカスタム **Server Load Balancer** を使用しますか。

**A:** 「*Server Load Balancer ルーティング*」をご参照ください。

**Q:** どのようにカスタム **Server Load Balancer** のラベルを使用しますか。

**A:** サービスオーケストレーションドキュメント*#unique_11* の **lb** をご参照ください。

**Q: Server Load Balancer** をサポートする **ECS** の設定方法を教えてください。

**A:** 原則として、**Server Load Balancer** インスタンスバックエンドに追加された **ECS (Elastic Compute Service)** インスタンスに対して特別な設定は必要ありません。 レイヤー 4 プロトコル **(TCP) Server Load Balancer** と関連する **Linux** の **ECS** インスタンスと 正常に 通信できない場合、システム構成ファイル `/etc/sysctl.conf` で以下の 3 つのパラメータの値が "0" であることを確認します。

```
net.ipv4.conf.default.rp_filter = 0
net.ipv4.conf.all.rp_filter = 0
net.ipv4.conf.eth0.rp_filter = 0
```

同じイントラネットセグメントでデプロイされた **ECS** インスタンスがお互いに通信できない場合、以下のパラメーターが正しく設定されているかを確認します。

```
net.ipv4.conf.default.arp_announce =2
```

```
net.ipv4.conf.all.arp_announce =2
```

sysctl -p コマンドを実行してパラメーター設定を更新します。

**Q:** カスタム **Server Load Balancer** のメリットは何ですか。

**A:** カスタム **Server Load Balancer** は、サービス設定の更新中やコンテナーが停止またはデプロイに失敗したときに、稼働していないバックエンドコンテナーのルートを自動的に削除します。 **Server Load Balancer** のその他の設定はご自身で管理する必要があります。

**Q:** カスタム **Server Load Balancer** に制限はありますか。

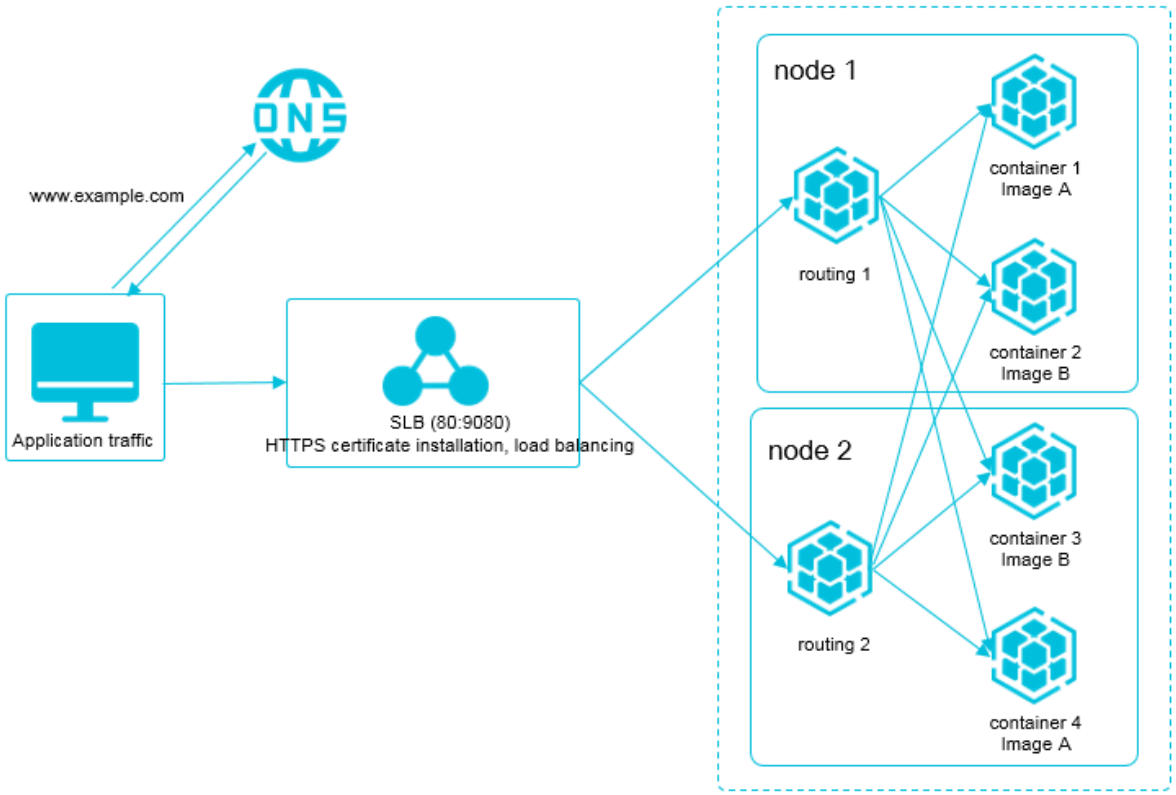
**A:** 現在、カスタム **Server Load Balancer** の制限は以下のとおりです。

- **Server Load Balancer** インスタンスを作成し、名前を付けて、対応するリスニングポートを作成します。 次に、拡張ラベルを使用して、**Server Load Balancer** インスタンス名を`$slb_name` または `$slb_id`にし 、公開するポート、使用するプロトコル `$scheme` (使用できる値は **tcp**、**http**、**https** および **udp** を含みます)、 マッピングされたコンテナーポート `$container_port`を設定し 、 フロントエンドポート `$front_port` を指定します。
- 公開するサービスポートの **hots: container** ポートマッピングを指定してから、標準 **Dockerfile** ラベル `ports` を使用して ポートマッピングを指定しなければなりません。 ホストポートを指定する必要があり、このポートは他のサービスによりマッピングされたホストポートと競合しないようにします。 **Server Load Balancer** はホストポートをバックエンド **ECS** インスタンスのバインドに使用します。
- サービスポートを公開するには、 **1** つのサービスに対し **1** つ以上の **Server Load Balancer** インスタンスを使わなくてはなりません。 サービスは同じ **Server Load Balancer** インスタンスを共有できません。これは、異なる **ECS** インスタンスバックエンドで分配されているためです。
- デプロイされた **Server Load Balancer NAT** マッピングを持つホストは同じ **host: container** ポートマッピングを使用します。 そのため、これらのサービスはそれぞれの **ECS** インスタンスで **1** つのインスタンスのみを持ちます。
- **Server Load Balancer** で使用できるプロトコル `$scheme` は **tcp**、**http**、**https** および **udp** を含みます。
- **Alibaba Cloud Server Load Balancer** コンソールでご自身でリスニングポートを作成します。
- **Server Load Balancer** コンソールにログインし、帯域幅制限などの、**Container Service** で使用される **Server Load Balancer** インスタンスに関する設定をご自身で変更します。

- `lb` ラベルの値は、**Server Load Balancer** のバックエンド **ECS** インスタンスをバインドすることなく、 対応するラベルを設定した後に、自動的にバックエンドがバインドされます。 そのため、**Server Load Balancer** バックンドのバインドを除いて、**Alibaba Cloud Server Load Balancer** コンソールで **Server Load Balancer** をご自身で設定および変更します。
- **Container Service** は、**RAM (Resource Access Management)** ユーザーの生成に有用です (**RAM** を有効にしておく必要があります)。 このアカウントはいくつかの **Server Load Balancer** の権限を持ちますが、**Server Load Balancer** インスタンスの作成および削除の権限はありません。 **Container Service** での **Server Load Balancer** インスタンスの管理、たとえば、クラスター内のいくつかのノードをサーバーのバックエンドにバインドするなどにこのアカウントをご使用ください。

## 2.8 どのようにリンク接続問題のトラブルシューティングを行いますか。

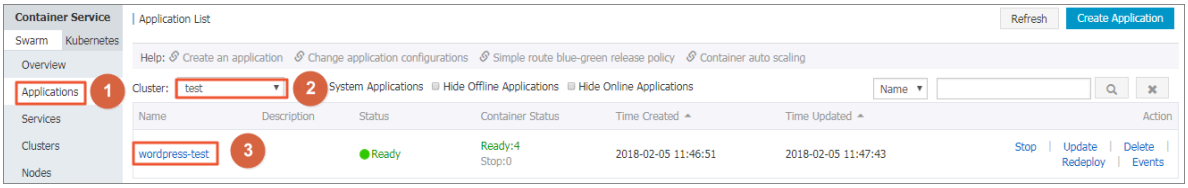
**Web** コンテナーが **Container Service** で構築され、このサーバーへのリクエスト転送にルーティングが使用されている場合、リクエストリンクは次のとおりです。 クライアント > **DNS** レゾルーション> **Container Service** > クラスターの **acsrouting** コンテナー > **Web** コンテナーへの転送。 これは以下の図のように示されます。



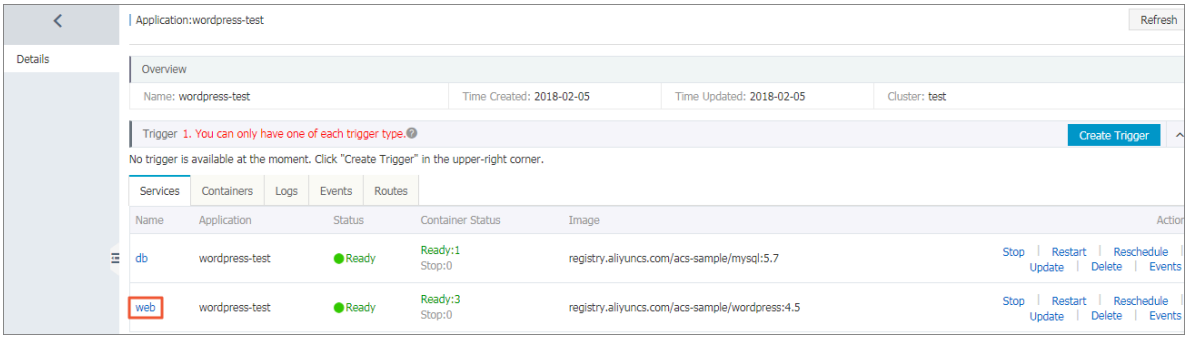
このプロセスのどこかのステージで問題が起こった場合、ユーザーリクエストは正しく **Web** コンテナーに転送されない可能性があります。 リンク接続に関する問題のトラブルシューティングは以下のようになります。問題が常に存在する 開発者の **Web** コンテナーに対するヘルスチェックから始めます。

1. コンテナーが実行中かを確認します。

   *Container Service* コンソールにログインします。 左側のナビゲーションウィンドウから [アプリケーション] をクリックします。 [クラスター] リストからクラスターを選択します。 アプリケーション名をクリックします (この例では、**"wordpress-test"** です)。

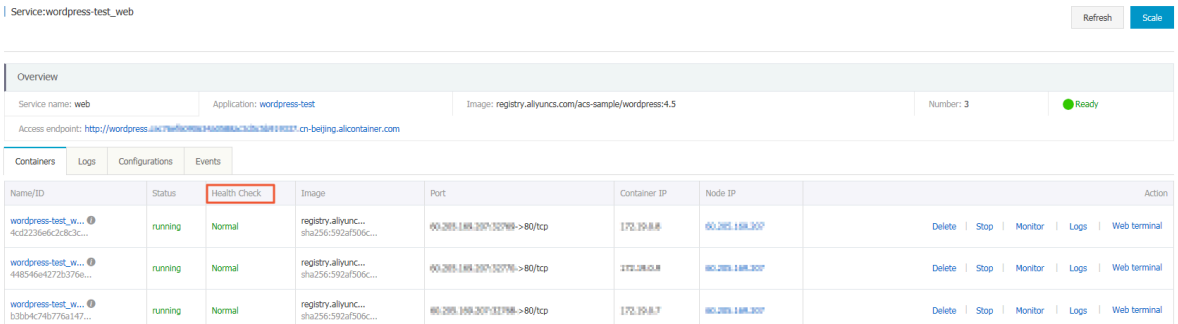



2. **Web** コンテナーを提供しているサービス名 (この例では、**"web"**) をクリックします。




3. **Web** サービスを提供するコンテナーのヘルスチェックステータスを確認します。

   [コンテナー] タブで、すべてのコンテナーが [正常] と [ヘルスチェック] で表示されているかを確認します。 そうでない場合、[ログ] をクリックしてエラーメッセージを確認し、 [イベント] タブをクリックしてデプロイに例外が発生していないかを確認します。 『ヘルスチェック』がアプリケーションに対して設定されている場合、ヘルスチェックページがステータスコード **200** を返しているかを確認し、ヘルスチェックステータスが正常であることを確認する必要があります。 以下の図をご参照ください。



4. **Web** コンテナーページが正常に応答しているかを確認します。

   コンテナーのヘルスチェックステータスが正常の場合、ルーティングサービスをバイパスして、**Web** コンテナーのアクセシビリティを直接確認する必要があります。 上の図で示したように、**Web** コンテナーのコンテナー **IP** を参照できます。 クラスターのマシンのルーティングコンテナーにログインし、コンテナー **IP** を使用して **Web** コンテナーページにリクエストします。 返された **HTTP** ステータスコードが **400** 未満の場合、 **Web** コンテナーのページは正常です。 以下の例では、`docker exec -it f171110f2fe2 sh` が使用されます。 ここで、`f171110f2fe2` はコンテナー **"acsrouting_routing_1"** のコンテナー **ID** で、 `curl -v 172.19.0.7` の **IP** アドレス `172.19.0.7` は **Web** サービスのコンテナー **IP** アドレスです。 リクエストがステータスコード **302** を返した場合、 **Web** コンテナーが正常にアクセスされています。

```
 root@c68a460635b8c405e83c052b7c2057c7b-node2:~# docker ps
 CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
 b403ea045fa1 registry.aliyuncs.com/acs-sample/wordpress:4.5 "/
entrypoint.sh apach" 13 seconds ago Up 11 seconds 0.0.0.0:32768->80/
tcp w_web_2
 025f7967cec3 registry.aliyuncs.com/acs-sample/mysql:5.7 "/
entrypoint.sh mysql" About a minute ago Up About a minute 3306/tcp
w_db_1
 2f247b8a76e5 registry.aliyuncs.com/acs/ilogtail:0.9.9 "/bin/sh -c '
sh /usr/" 31 minutes ago Up 31 minutes acslogging_logtail_1
 42b75bee6cd8 registry.aliyuncs.com/acs/monitoring-agent:latest "acs
-mon-run.sh --hel" 31 minutes ago Up 31 minutes acsmonitoring_acs-
monitoring-agent_2
 0a9afa527f03 registry.aliyuncs.com/acs/volume-driver:0.7-252cb09
 "acs-agent volume_exe" 31 minutes ago Up 31 minutes acsvolumed
river_volumedriver_2
 3c1440fd114c registry.aliyuncs.com/acs/logspout:0.1-41e0e21 "/bin/
logspout" 32 minutes ago Up 32 minutes acslogging_logspout_1
 f171110f2fe2 registry.aliyuncs.com/acs/routing:0.7-staging "/opt/
run.sh" 32 minutes ago Up 32 minutes 127.0.0.1:1936->1936/tcp, 0.0.0
.0:9080->80/tcp acsrouting_routing_1
 0bdeb8464c14 registry.aliyuncs.com/acs/agent:0.7-bfe8bdf "acs-agent
 join --nod" 33 minutes ago Up 33 minutes acs-agent
 ba32a0e9e7fe registry.aliyuncs.com/acs/tunnel-agent:0.21 "/acs/
agent -config=c" 33 minutes ago Up 33 minutes tunnel-agent
 root@c68a460635b8c405e83c052b7c2057c7b-node2:~# docker exec -it
f171110f2fe2 sh
 / # curl -v 172.19.0.7
 * Rebuilt URL to: 172.19.0.7/
 * Trying 172.19.0.7...
 * Connected to 172.19.0.7 (172.19.0.7) port 80 (#0)
   > GET / HTTP/1.1
   > Host: 172.19.0.7
   > User-Agent: curl/7.47.0
   > Accept: */*
   >
   < HTTP/1.1 302 Found
   < Date: Mon, 09 May 2016 03:19:47 GMT
   < Server: Apache/2.4.10 (Debian) PHP/5.6.21
   < X-Powered-By: PHP/5.6.21
   < Expires: Wed, 11 Jan 1984 05:00:00 GMT
   < Cache-Control: no-cache, must-revalidate, max-age=0
```

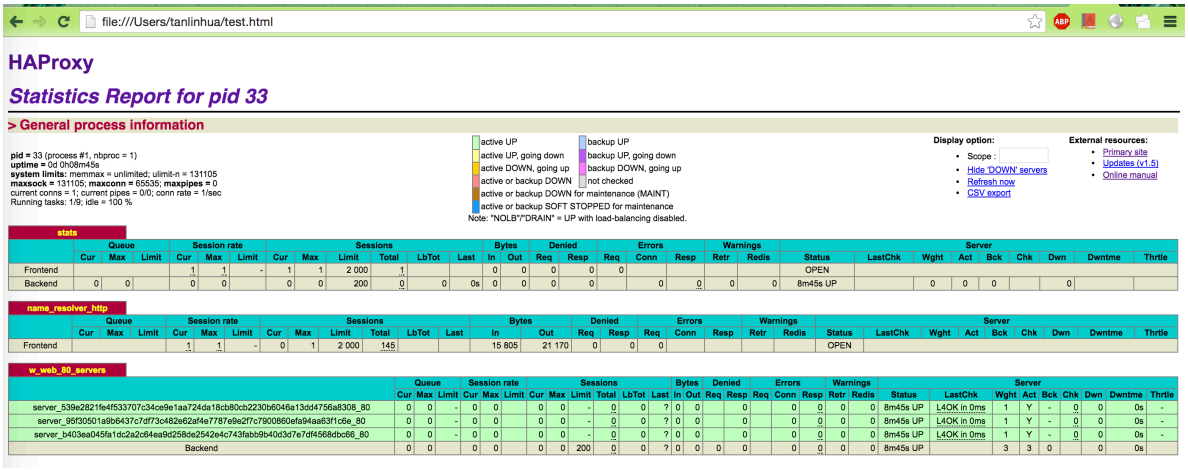
```
    < Pragma: no-cache
    < Location: http://172.19.0.7/wp-admin/install.php
    < Content-Length: 0
    < Content-Type: text/html; charset=UTF-8
    <
  * Connection #0 to host 172.19.0.7 left intact
```

5. **Acsrouting** の有効性を検証します。

   最新バージョンにルーティングを アップグレード します。 クラスターのそれぞれのマシンにログインして (どのマシンにアプリケーションコンテナーがデプロイされたとしても、いずれかのマシンがリクエストを受信します)、 ルーティングのヘルスチェックページをリクエストします。

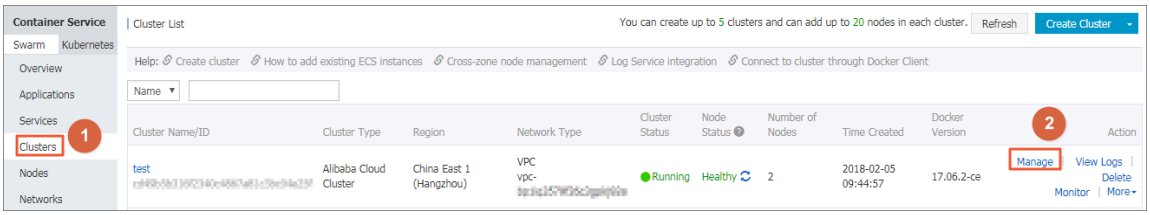
```
root@c68a460635b8c405e83c052b7c2057c7b-node2:~# curl -Ss -u admin:
admin 'http://127.0.0.1:1936/haproxy?stats' &> test.html
```

   ブラウザでページ *test.html* をマシンにコピーして、ブラウザを使用して、 ローカルファイル *test.html* を開きます。 該当する **Web** サービスおよびコンテナーバックエンドを確認します。 最初の部分は統計情報で、 ルーティングの統計を提供しています。 **2** 番目の部分はフロントエンドの統計です。 **3** 番目の部分は、 バックエンド情報を提供しており、確認がもっとも重要な部分です。 ここで、**w_web_80_servers** はアプリケーション **"wordpress-test"** のサービス **"web"** のバックエンドポート **80** の情報を示しています。 合計で **3** つのバックエンドサーバーが存在しており、つまり**Web** サービスを提供する コンテナーが **3** つバックエンドにあります。 緑は、 ルーティングコンテナーが **3** つのコンテナーに接続できており、システムが正常に作動していることを示しています。 他の色はいずれも例外を示しています。



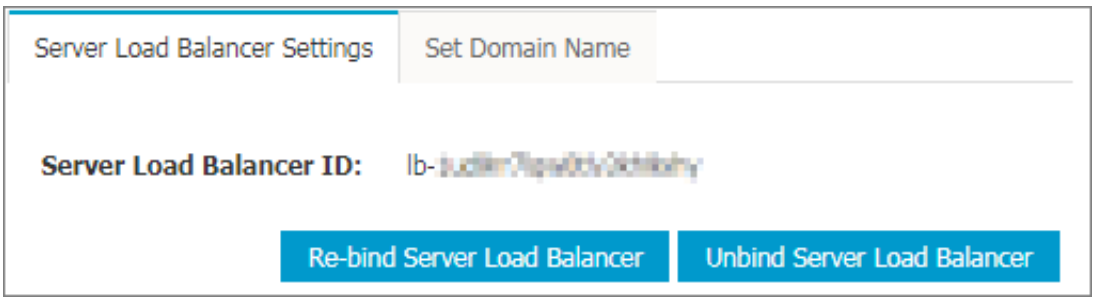
6. **Server Load Balancer VIP** がデータを正しく転送しており、ヘルスチェックステータスが正常であることを確認します。

   a) クラスターの **Server Load Balancer VIP** を探します。 左側のナビゲーションウィンドウから **[クラスター]** をクリックします。 *Container Service コンソール*にログインします。

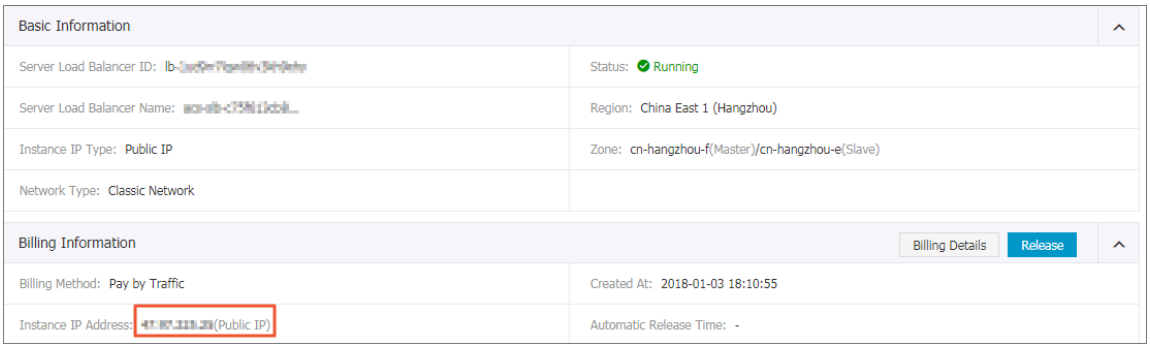



   b) クラスター (この例では、**"test"**) の右側にある **[管理]** をクリックします。 左側のナビゲーションウィンドウから **[ロードバランサーの設定]** をクリックします。 **Server Load Balancer ID** を参照し、コピーします。 **[プロダクト] > [Server Load Balancer]** をク

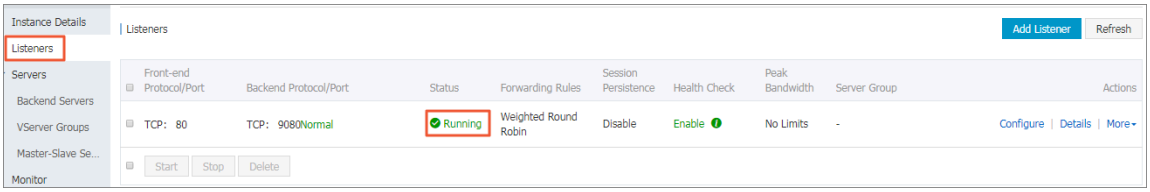
リックして、*Server Load Balancer コンソール*に移動します。 **Server Load Balancer** インスタンスの右側にある [管理] をクリックし、インスタンスの詳細ページを開きます。



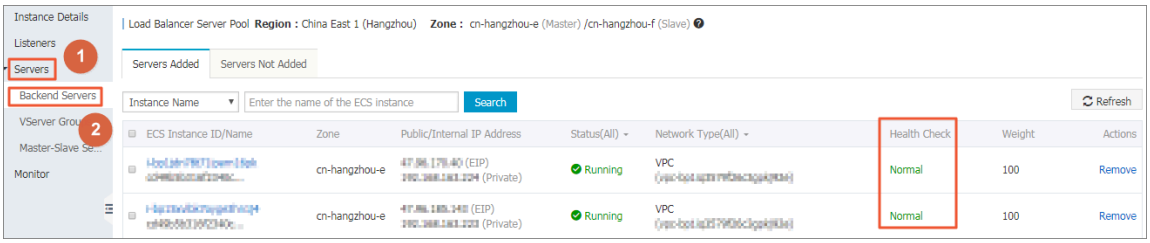

c) **Server Load Balancer** インスタンスの **IP** アドレスを参照します。




d) **Server Load Balancer** ポートのヘルスステータスを参照します。 左側のナビゲーションウィンドウから [リスナー] をクリックします。 "実行中" [ステータス] は ポートが正常に動作していることを示します。




e) **Server Load Balancer** にマウントされているバックエンドサーバーのステータスを確認します。 左側のナビゲーションウィンドウから [サーバー] > > [バックエンドサーバー] をクリックします。[ヘルスチェック] ステータスが "正常" であることを確認します。




7. ドメイン名が正しく **Server Load Balancer VIP** に名前解決されているかを確認します。 たとえば、`ping` または `dig` コマンドを使用して、 名前解決の結果を参照します。 ドメイン名

が解決され、前の手順で検索した **Server Load Balancer VIP** のアドレスに送信されている必要があります。

```
$ ping www.example-domain.com
```

```
$ dig www.example-domain.com
```

## 2.9 Container Service は他のユーザーのコンテナーとどのように分離していますか。

**Container Service** は、お客様の権限に基づいて **ECS (Elastic Compute Service)** インスタンスを生成し、管理します。 お客様のコンテナーは、お客様の **ECS** インスタンス上でのみ稼働させることができます。

クラスターがクラシックネットワークである場合、他のユーザーのクラスター間のアクセスはセキュリティグループにより分離されています。

クラスターのネットワークが **VPC (Virtual Private Cloud)** の場合、他のユーザーのクラスター間のアクセスは **VPC** により分離されます。

お客様のクラスターのセキュリティグループや **VPC** アクセス権限を変更することができます。

## 2.10 アプリケーション作成し、カスタム Server Load Balancer インスタンスを追加すると、 "Invalid input for user ram ak or ak secret" が表示されます。

1. **RAM** サービスが起動しているかどうかを確認します。 起動している場合は、手順 **2** に進みます。 起動していない場合は、サービスを起動し、再度試します。
2. **RAM** アカウント数が上限に達しているかどうかを確認します。 達している場合、アカウントを **1**つ 削除して再度試します。
3. 上記の問題のどちらも存在しない場合、 **RAM** 権限情報を更新します (*Container Service* コンソールにログインします。 左側のナビゲーションウィンドウから **[**クラスター**]** をクリックします。 クラスターの右側にある **[**詳細**]** をクリックします。 ドロップダウンリストから **[RAM** 権限情報の更新**]** を選択します。 表示されたダイアログボックスで **[**確認**]** をクリックします)。

## 2.11 アプリケーション設定の変更に関するよくある質問

デフォルトでは、アプリケーションの設定を変更した場合、**Container Service** は現行マシン上にコンテナーを再起動または再作成します。 そうすることで、現行マシン上のサービスコンテ

ナーのローカルデータボリュームが失われないようにしています。 そのため、設定変更時に、コンテナーを別のマシンにスケジュールするように設定した場合、**Container Service** は設定されたスケジュールを無視します。

サービスにローカルデータボリュームがないことが確実な場合やローカルデータボリュームのコンテナーデータが失われてもよい場合、[強制再スケジュール] チェックボックスをオンにします。 そうすると、 **Container Service** は、[テンプレート] のスケジュール設定に基づいて別のマシンにコンテナーをスケジュールします。

 注:
[強制再スケジューリング] チェックボックスをオンにしてコンテナーを別のマシンにスケジュールした場合、現行マシンのローカルデータボリュームのコンテナーデータは失われます。 そのため注意して進めます。

**例**

コンテナーが **node1** にデプロイされているとします。

アプリケーション設定の変更時、次のようにコンテナーを **node2** (`constraint:aliyun.node_index==2`) にスケジューリングするように設定します。

```
web:
    image: 'nginx:latest'
    restart: always
    environment:
        - 'constraint:aliyun.node_index==2'
    ports:
        - 80
    labels:
        aliyun.scale: 1
```

この状況では、

- [強制再スケジュール] チェックボックスがオフの場合、**Container Service** はスケジューリング設定を無視し、コンテナーを **node1** にデプロイします。
- [強制再スケジュール] チェックボックスがオンの場合、**Container Service** はコンテナーを **node2** にスケジュールします。 **node1** のローカルデータボリュームのコンテナーデータは失われます。

## 2.12 クラスターの作成失敗につながるよくあるエラー

**Container Service** でクラスター作成する際、エラーのために失敗することがあります。 よくあるエラーと解決法について以下をご参照ください。

- **Server Load Balancer** エラー。 **(RequestID** をサポートスタッフに お伝え下さい。**)**

```
Config cs AcessRouting failed : Failed to CreateLoadBalancer:Aliyun
 API Error: RequestId: 47AE1BFC-AFEA-469C-82F5-1E3BD81897F2 Status
```

```
 Code: 500 Code: InternalError Message: The request processing has
failed due to some unknown error, exception or failure.
```

- **Docker** 設定がタイムアウトしました。 **(daemon** ログを 確認します**)**

```
Adding tags map[provider:aliyunacs acsversion:1.0 acsclusterid:xxx
acsclustername:xxx] to instance xx
```

- 現在、購入いただくことのできないゾーンです。

```
Code: Zone.NotOnSale Message: The specified zone is not on sale.
```

- クラスターマスターの作成に 失敗しました。

```
Creating Master Region Controller: 500 Internal Server Error
```

- セキュリティグループが上限を超えています。

```
Config cs ClusterNetwork failed : Aliyun API Error: RequestId
: 264E5ADC-0571-44C0-8508-C037096856C7 Status Code: 403 Code:
QuotaExceed.SecurityGroup Message: The security group quota exceeds.
```

- リソースが不足しています。

```
Failed to create instance: Aliyun API Error: RequestId: CC2FF296-
D29E-484E-B095-8905CDA016BA Status Code: 403 Code: OperationDenied
Message: The resource is out of usage.
```

- **Server Load Balancer** インスタンス数が上限を超えています。

```
Config cs AcessRouting failed : Failed to CreateLoadBalancer:Aliyun
 API Error: RequestId: A1E5D644-C31A-4142-B722-CBD6EF57A3A7 Status
Code: 400 Code: OverQuota Message: The Total is over the quota.
```

- 共有イメージ数が上限を超えています。

```
Fail shared image: Aliyun API Error: RequestId: 693F8B6C-9349-4277
-B109-9841BFF1F76C Status Code: 404 Code: InvalidAccount.Forbbiden
Message: The specified Account does not yourself.
```

- **EIP (Elastic IP)** が上限を 超えています。

```
QuotaExceeded.Eip Message: Elastic IP address quota exceeded
```

- 従量課金 **ECS (Elastic Compute Service)** インスタンスが 上限を超えています。

```
QuotaExceed.AfterpayInstance Message: Living afterpay instances
quota exceeded
```

- パラメーターが見つかりません。 **(**基本的に **API** 呼び出し時**)**

```
c7904aa1b9e3642fa8fbf440f48dfedb8 | Fail shared image: Aliyun API
Error: RequestId: D2518C32-0C2B-4EAD-9F88-43A5C2AAF0C1 Status Code:
```

```
404 Code: InvalidAccount.NotFound Message: The specified parameter "
AddAccount.n" or "RemoveAccount.n" does not exist.
```

- インスタンスタイプが一致しません。(このエラーは主に **API** ユーザーから 書き出されます。)

```
InvalidInstanceType.ValueUnauthorized Message: The specified
InstanceType is not authorized
```

- **VPC (Virtual Private Cloud)** 環境で設定されたコンテナー **CIDR (ClasslessInter-Domain Routing)** ブロックは、現行のルーティングテーブル **CIDR** ブロックまたは、 **VPC** 下の **VSwitch** の **CIDR** と競合しています。

```
InvalidCIDRBlock.Duplicate Message: Specified CIDR block is already
exists
```

## 2.13 コンテナーでのDNS オプションを設定し、DNS 名前解決を最適化する

### コンテナーでの DNS オプションの設定

`dns` や`dns_options` を **Container Service** のオーケストレーションテンプレートで指定し、コンテナーの **DNS** サーバーおよび **DNS** オプションを指定できます。

たとえば、

```
testdns:
  image: nginx
  dns:


  dns_options:
    - use-vc
    - no-tld-query
```

上記の例では、**DNS** サーバーおよび **DNS** クエリオプションをサービスコンテナーに対して設定しています。

 注:

**Docker** では、サービスの検出のために各コンテナーに **DNS** サーバーが埋め込まれています。コンテナーの */etc/resolv.conf* ファイルにおける **DNS** サーバーは **Docker** のビルトイン **DNS** サーバー **"127.0.0.11"** です。 **Docker** はビルトインサーバーの **DNS** リクエストをリッスンし、**DNS** リクエストを `dns` により設定されたサーバーに転送します。

### DNS 名前解決の最適化

ドメイン名がリクエストされた際、**DNS** 名前解決がタイムアウトまたはエラーになり、**Web** サイトにアクセスできなくなることがあります。 オペレーティングシステムは、通常、`nscd` サービスを有効にし、**DNS** キャッシュすることで **DNS** 名前解決のエラーを回避しています。 しかし、`nscd` サービスは通常コンテナーイメージでは設定されていません。 **DNS** 名前解決を頻繁に行うコンテナーに `nscd` サービスをインストールすることで、コンテナーでの **DNS** 名前解決を最適化できます。

`nscd` ソフトウェアパッケージをインストールします。 次に、クラスターの再起動時に、まず `nscd` サービスを起動し、 その後にプロセスを開始します。

```
FROM registry.aliyuncs.com/acs/ubuntu
RUN apt-get update && apt-get install -y nscd && rm -rf /var/lib/apt/
lists/*
CMD service nscd start; bash
```

## 2.14 Container Service Docker でデータディスクを追加する方法を教えてください。

**Docker** データは **Union File System** を使ってディスクに保存されます。 マシンで実行されるコンテナー数やイメージ数が継続的に増加する場合、ディスクサイズが要件を満たさないことがあります。 このような場合は、データディスクを増やして、 **Docker** データディレクトリ用のストレージ領域を拡張します。

### Docker データディレクトリ

**Docker** では、コンテナーデータおよびイメージデータはデフォルトで */var/lib/docker* ディレクトリに保存されます。 du コマンドを実行して、このディレクトリの占有ディスクサイズを確認できます。 例

```
du -h --max-depth=0 /var/lib/docker
7.9G /var/lib/docker
```

### Docker データディスクの変更

多くの **Docker** イメージは大きいものです。 そのため、数個のイメージがディスクを占有し、ディスク領域に不足が生じます。 **Docker** データディレクトリ用のデータディスクを増やすことで、 継続的に増加するイメージまたはコンテナーの要求を満たすことができます。

### ECS データディスクの購入し、拡張が必要なマシンへマウントする

1. *ECS (Elastic Compute Service) コンソール*にログインして、対応する設定のクラウドディスクを購入します。

2. 左側のナビゲーションウィンドウから [インスタンス] をクリックします。 リージョンを選択して、インスタンス名をクリックするか、インスタンスの右側にある [管理] をクリックします。 > 左側のナビゲーションウィンドウから [インスタンスのディスク] をクリックします。 > 右上の [ディスクのアタッチ] をクリックします。購入したディスクを選択して、マウントポイント `/dev/xvd*` または `/dev/vd*` を登録します。 cd コマンドを実行してマウントポイントを決定します。 I/O 最適化インスタンスのマウントポイントは `/dev/vd*` となります。

マシンへのログインし、マウントされたディスクを初期化する

1. `ls -l /dev/xvd*` または `ls -l /dev/vd*` をマシンで実行して、ディスク ID と 先に登録したものと一致することを確認します。
2. fdisk コマンドを実行してディスクを分割します。 次に、`mkfs.ext4` によりディスクを初期化します。 詳しくは、*#unique_37*をご参照ください。 例

```
root@iZbp16h1ijt5er5wempg4sZ:~# ls -l /dev/vd*
 brw-rw---- 1 root disk 253, 0 Jan 5 17:44 /dev/vda
 brw-rw---- 1 root disk 253, 1 Jan 5 17:44 /dev/vda1
 brw-rw---- 1 root disk 253, 16 Jan 5 17:55 /dev/vdb
 root@iZbp16h1ijt5er5wempg4sZ:~# fdisk -S 56 /dev/vdb
 Welcome to fdisk (util-linux 2.27.1).
 Changes will remain in memory only, until you decide to write them.
 Be careful before using the write command.
 Device does not contain a recognized partition table.
 Created a new DOS disklabel with disk identifier 0x44e128c4.
 Command (m for help): n
 Partition type
    p primary (0 primary, 0 extended, 4 free)
    e extended (container for logical partitions)
 Select (default p): p
 Partition number (1-4, default 1): 1
 First sector (2048-41943039, default 2048):
 Last sector, +sectors or +size{K,M,G,T,P} (2048-41943039, default
41943039):
 Created a new partition 1 of type 'Linux' and of size 20 GiB.
 Command (m for help): wq
 The partition table has been altered.
 Calling ioctl() to re-read partition table.
 Syncing disks.
 root@iZbp16h1ijt5er5wempg4sZ:~# ll /dev/vd*
 brw-rw---- 1 root disk 253, 0 Jan 5 17:44 /dev/vda
 brw-rw---- 1 root disk 253, 1 Jan 5 17:44 /dev/vda1
 brw-rw---- 1 root disk 253, 16 Jan 5 17:58 /dev/vdb
 brw-rw---- 1 root disk 253, 17 Jan 5 17:58 /dev/vdb1 ##Add
partition
 root@iZbp16h1ijt5er5wempg4sZ:~# mkfs.ext4 /dev/vdb1 ##Format
 mke2fs 1.42.13 (17-May-2015)
 Creating filesystem with 5242624 4k blocks and 1310720 inodes
 Filesystem UUID: cef1625c-7533-4308-bc44-511580e3edc8
 Superblock backups stored on blocks:
         32768, 98304, 163840, 229376, 294912, 819200, 884736,
1605632, 2654208,
         4096000
 Allocating group tables: done
 Writing inode tables: done
 Creating journal (32768 blocks): done
```

```
Writing superblocks and filesystem accounting information: done
```

**Docker** データを新しいディスクへ移行する

1. まず **Docker daemon** を停止し、**Docker** データの移行過程でのデータの整合性を確保します。 `service docker stop` コマンドを実行して、 **Docker daemon** を停止します。
2. **Docker** デフォルトデータディレクトリのデータをバックアップディレクトリに移します。 たとえば、バックアップディレクトリが */var/lib/docker_data* の場合、 コマンド `mv /var/lib/docker /var/lib/docker_data` を実行します。
3. 新しく初期化されたディスクを */var/lib/docker* ディレクトリにマウントします。 コマンド `echo "/dev/vdb1 /var/lib/docker ext4 defaults 0 0" >>/etc/fstab && mkdir /var/lib/docker && mount -a` を実行します。
4. バックアップされた **Docker** データを新しいディスクに移します。 コマンド `mv /var/lib/docker_data/* /var/lib/docker/` を実行します。

**Docker daemon** を起動し、データの格納場所を確認する

1. **Docker daemon** を起動して、コマンド `service docker start` を実行します。
2. コマンド `df` を実行します。 */var/lib/docker* が新しいディスクにマウントされていることが確認できます。

```
root@iZbp16h1ijt5er5wempg4sZ:/# df -h
Filesystem Size Used Avail Use% Mounted on
udev 2.0G 0 2.0G 0% /dev
tmpfs 396M 7.1M 389M 2% /run
/dev/vda1 40G 2.7G 35G 8% /
tmpfs 2.0G 476K 2.0G 1% /dev/shm
tmpfs 5.0M 0 5.0M 0% /run/lock
tmpfs 2.0G 0 2.0G 0% /sys/fs/cgroup
tmpfs 396M 0 396M 0% /run/user/0
/dev/vdb1 20G 2.1G 17G 12% /var/lib/docker ##This directory is
mounted to the new disk.
```

3. コマンド `docker ps` を実行してコンテナーが失われていないかを確認します。 必要に応じて、関連するコンテナー、たとえば、 `restart:always` のラベルが設定されていないコンテナーなどの関連コンテナーを再起動します。

## 2.15 どのようにログの問題のトラブルシューティングを行いますか。

アプリケーションに拡張ラベル `label aliyun.log_store_xxx: xxx` が追加されていても、**Log Service** にログが取得されない場合、以下の手順に従い問題のトラブルシューティングを行います。

 注：

トラブルシューティングの手順をスキップしないでください。

1. Logstore **が正常に作成されているかの確認**

   **Logstore** が作成されていない場合、アプリケーションが正常にデプロイされていません。 デプロイに関するエラーメッセージがアプリケーションイベントにあるかを確認します。

2. ilogtail **バージョンの確認**

   コマンド `docker ps|grep ilogtail` をマシンで実行して、出力により **ilogtail** イメージのバージョンを決定します。 バージョンが **0.11.6** の場合、 システムサービスを最新バージョンにアップグレードします (現在、最新バージョンは **0.13.4** です)。 アップグレード後、アプリケーションが新しいログを生成した後に**Log Service** コンソールでログを照会します。

3. ilogtail **ログの確認**

   コマンド `docker exec -it <ilogtail container ID> cat /usr/local/ilogtail/ilogtail.LOG` を実行して、**ilogtail** ログにより問題が何であるかを特定します。 よくある理由は次のとおりです。

   - ネットワークに接続されていない。 次のコマンドを実行し、ネットワークの接続の有無を確認します。

     ```
     VPC (Virtual Private Cloud):
       telnet logtail.cn-<region>-vpc.log.aliyuncs.com 80
       Internet:
       telnet logtail.cn-<region>.log.aliyuncs.com 80
     ```

   - **Access Key** が設定されていない。

     `プライマリアカウントが Access Key が設定しない場合、ログにUnauthorized ErrorMessage:no authority, denied by ACL` と表示されます。 まずプライマリアカウント用に **Access Key** を作成します。 **"Unauthorized ErrorMessage:no authority, denied by ACL"** がログに書き出されているかどうかにかかわらず、 プライマリアカウントが **Access Key** を指定しているかどうかを確認します。

4. **マシンの** IP **が** Log Service **マシングループに属しているかどうかの確認**

   1. *Log Service コンソール*にログインします。
   2. 該当するクラスターの **Log Service** プロジェクト名を確認します。 プロジェクトの命名規則は **"acslog-project-**<クラスター **ID** の頭 **10** 文字>" となります。
   3. 左側のナビゲーションウィンドウから **[logtail** マシングループ] をクリックします。

4. マシングループの右側にある [マシンステータス] をクリックして、現行マシンの IP アドレスが IP リストに記載されているかを確認します。

5. **プライマリアカウントが** Access Key **を指定しているかどうかの確認**

   プライマリアカウントが少なくとも 1 つの有効な **Access Key** を持つことを確認します。

6. **ログファイルに内容があるかの確認**

   業務アプリケーションコンテナーに入り、実際にログが生成されているかを確認します。 **stdout** ログでは、**"docker logs"** コマンドを直接使用します。

## 2.16 Container Service で Nginx + FPM を使用する

**Container Service** で **Nginx** + **FPM** を使用するには、ベースイメージとし **Nginx** と **FRM** の両方を含 む『*https://github.com/ngineered/nginx-php-fpm*』を使用することを推奨します。

このイメージは **Nginx** と **PHP-FPM** 用のコンテナーの作成に使用できます。 作成されたコンテナーは **Git** から **Web** サイトコードをプルしたり、コードの変更を **Git** にプッシュ またはプルできます。 コンテナーは、**Docker** に渡される変数を使用してオーケストレーションファイルを更新し、コードや設定を変更できます。

このイメージは、**Let's Encrypt SSL** 設定、**Nginx** 設定のカスタマイズ、**Nginx** や **PHP** 設定の変更、**X-Forwarded-For** ヘッダーおよび **UID** マッピング (ローカルデータボリューム対応) にも対応します。

## 2.17 ノードの例外

ノードのステータスが [例外] の場合、**Container Service** はノードに接続できません。

**原因分析**

ノードの例外は主に重いノード負荷により発生します。これには、**CPU** 使用率、メモリー使用率、ネットワークトラフィックおよびノードの **I/O** が含まれます。

**Swarm** クラスター

ノードのモニタリングデータを、**Container Service** コンソールまたは **Alibaba Cloud CloudMonitor** コンソールのどちらかで確認できます。

- **Container Service** コンソールでノードのモニタリングデータを確認

  1. *Container Service コンソール*にログインします。
  2. 左側のナビゲーションウィンドウから **[Swarm]** > **[クラスター]** をクリックします。
  3. クラスター名をクリックします。
  4. 該当するノードの右側にある **[モニター]** をクリックします。

- **Alibaba Cloud CloudMonitor** コンソールでノードのモニタリングデータを確認

  1. *CloudMonitor コンソール*にログインします。
  2. 左側のナビゲーションウィンドウから **[クラウドサービスモニタリング]** > **[Container Service]** をクリックします。
  3. 該当ノードが属するクラスターの右側にある **[ノードモニタリング]** をクリックします。
  4. ノードの右側の **[モニタリングチャート]** をクリックして、選択したノードのモニタリングデータを確認します。

  > 注：
  > リアルタイムでノードの負荷を監視するために、ノードにアラームルールを作成します。ページの右上の **[アラームルールの作成]** をクリックします。

**Kubernetes** クラスター

ノードのモニタリングデータを、**Container Service** コンソールまたは **Kubernetes** アプリケーショングループのどちらかで確認できます。

- **Container Service** コンソールでノードのモニタリングデータを確認

  1. *Container Service コンソール*にログインします。
  2. 左側のナビゲーションウィンドウから **[Kubernetes]** > **[クラスター]** > > **[ノード]** をクリックします。
  3. **[クラスター]** ドロップダウンリストからクラスターを選択します。 該当ノードの右側にある **[モニター]** をクリックします。

- **CloudMonitor** コンソールの **Kubernetes** アプリケーショングループでノードのモニタリングデータを確認

  1. *Container Service コンソール*にログインします。
  2. 左側のナビゲーションウィンドウから **[Kubernetes]** > **[クラスター]** をクリックします。
  3. クラスターの右側にある **[詳細]** をクリックして、 **[モニタリングサービスのアップグレード]** を > 選択します。表示されるダイアログボックスで **[OK]** をクリックします。
  4. *CloudMonitor コンソール*にログインします。
  5. 左側のナビゲーションウィンドウから **[アプリケーショングループ]** をクリックします。

**解決法**

以下の方法でノードの例外を解決できます。

- ノードにデプロイされているコンテナー数を削減する。
- コンテナーが使用するリソースを制限する。 **Swarm** クラスターについては*#unique_41*をご参照ください。
- ノードが正常になるように負荷を削減する。
- ノードまたはクラスターを拡張する。
- クラスターのグループリソースに対してモニタリングチャートを追加し、アラームルールを作成する。これによりノードのオーバーロードを回避します。

# 2.18 Container Service のオペレーティングシステムとカーネルに関するよくある質問

**コンテナーの削除または更新時にエラーが発生する**

コンテナーの削除または更新時に発生するエラーは以下に類似しています。

```
failed to remove root filesystem for xxx: device or resource busy
```

通常、このエラーはコンテナーが属するノードのカーネルのバージョンが低いために発生します。 コンテナーが属するノードにログインして、コマンド `uname -a` を実行してカーネルのバージョンを 表示します。 以下の場合にエラーが発生します。

- **Ubuntu 14.04** で、カーネルバージョンが **3.13** 以前の場合。
- **CentOS 7** で、カーネルバージョンが **3.10.0-514** 以前の場合。

解決法

コンテナーが属するノードのカーネルをアップグレードして、この問題を解決できます。

1. スケジューリング制約機能を利用して、アプリケーションをこのノードからスケジューリングします。 詳しくは、*#unique_43*をご参照ください。
2. ノードのカーネルバージョンをアップグレードします。

   **Ubuntu 14.04**

   ```
   apt-get update && apt-get install -y linux-generic-lts-xenial
   ```

   **CentOS 7**

   ```
   yum update -y kernel
   ```
3. カーネルのアップグレード後にノードを再起動し、新しいバージョンのカーネルを有効にします。
4. スケジューリング制約機能を利用して、アプリケーションをこのノードに戻すようにスケジュールします。

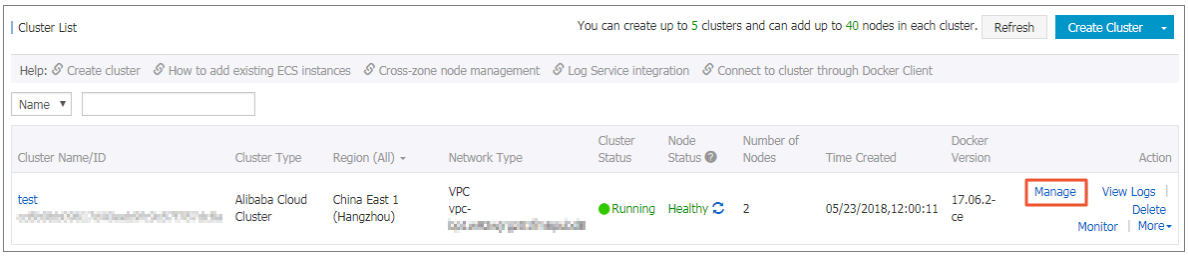
### コンテナーの時間に対して NTP 同期が設定されているかどうか

**Linux** の時間はカーネルインターフェイスにより取得されます。そのカーネルは同じノード上の各コンテナーにより共有されます。 そのため、時間が一致しています。 通常、ノードで **NTP** 時間の同期が設定されています。 **NTP** 同期のためにコンテナーに追加設定する必要はありません。

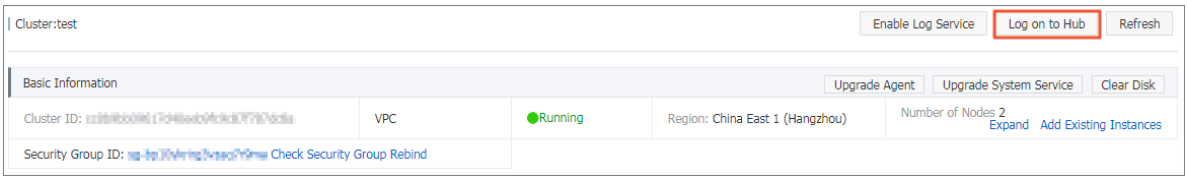
## 2.19 イメージのプルに失敗したらどうしたらいいですか。

イメージのプルに失敗した場合、以下の手順で再度イメージリポジトリにログインします。

1. **[クラスターリスト]** ページで、アプリケーションがデプロイされているクラスターの右側にある **[管理]** を クリックします。

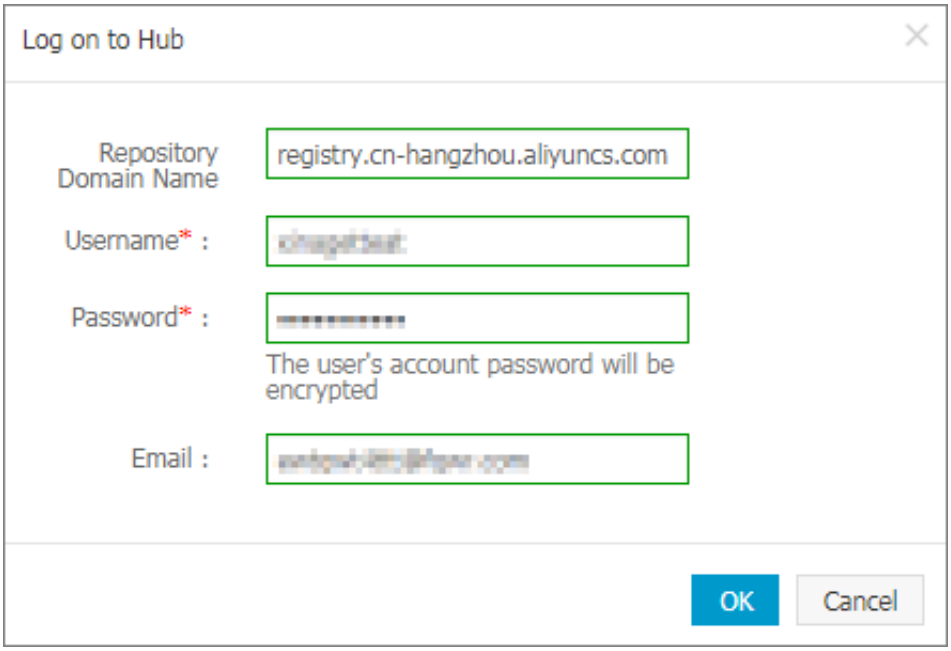



2. **[Hub へのログイン]** をクリックします。



3. 表示されたダイアログボックスで、ログイン情報を入力して **[OK]** をクリックします。

   **Alibaba Cloud** イメージリポジトリを使用するには**Alibaba Cloud** イメージリポジトリのドメイン名 (たとえば、**"registry.cn-hangzhou.aliyuncs.com"** ) を [リポジトリドメイン名] フィールドに、**Alibaba Cloud** ユーザー名を [ユーザー名] フィールドに、[パスワード] フィールドにリポジトリにログインするために使用する個別のパスワードを入力します。




## 2.20 Container Service は RAM コンソールでのサブアカウントへの権限付与をサポートしていますか。

現在、**Container Service** では **RAM (Resource Access Management)** コンソールでのサブアカウントに対する権限の付与をサポートしていません **(RAM** をサポートするクラウドプロダクトについては、*RAM* に対応しているクラウドサービスのリストをご参照ください)。 ただし、*Container Service* コンソールでサブアカウントに権限を付与できます。

**Container Service** コンソールでのサブアカウントに対する権限付与の方法については、*#unique_46*をご参照ください。