Alibaba Cloud
Elastic Compute Service

ベストプラクティス

Document Version20190813

目次

1 セキュリティ

1.1 セキュリティグループのベストプラクティス (パート 2)

ここでは、次の点を紹介します。

- セキュリティグループのAuthorizeSecurityGroupeとRevokeSecurityGroup
- セキュリティグループのJoinSecurityGroupeとLeaveSecurityGroup

Alibaba Cloud は、クラシックネットワークと VPC ネットワークという 2 種類のネットワークを提供します。 それらは異なるセキュリティグループルールをサポートしています。

- クラシックネットワークの場合: イントラネットインバウンド、イントラネットアウトバウンド、インターネットインバウンド、およびインターネットアウトバウンドのルールを設定します。
- VPC ネットワークの場合: イントラネットインバウンドとイントラネットアウトバウンドのルールを設定します。

セキュリティグループに関するイントラネット通信の基礎知識

セキュリティグループのイントラネット通信に関する次の点を解説します。

- デフォルトでは、同一セキュリティグループ内の ECS インスタンスだけが互いにアクセスできます。 つまり、異なるセキュリティグループ内の同一アカウントのインスタンスは、イントラネット上で互いにアクセスできません。 これは、クラシックネットワークと VPC ネットワークの両方に当てはまります。 したがって、クラシックネットワークの ECS インスタンスはイントラネット上で安全が保証されます。
- 異なるセキュリティグループに 2 つの ECS インスタンスがあり、それらがイントラネット経由で互いにアクセスできないようにしたいが実際にはアクセスできてしまう場合は、セキュリティグループのイントラネットルール設定を確認する必要があります。 イントラネットのルールに次の項目が含まれる場合は、再設定することを推奨します。
  - すべてのポートを許可
  - 許可されたオブジェクトが、CIDR セグメント (SourceCidrIp) 。`0.0.0.0/0` または `10.0.0.0/8`。 クラシックネットワークの場合、上記のルールによってイントラネットが外部アクセスにさらされる可能性があります。

- 異なるセキュリティグループのリソース間でネットワーク相互通信を実装する場合は、セキュリティグループ権限を採用する必要があります。 イントラネットアクセスの場合は、CIDR セグメント権限ではなく、送信元セキュリティグループ権限を採用することを推奨します。

セキュリティルールの属性

セキュリティルールは、次の属性を持つさまざまなアクセス権限を主に記述します。

- Policy: 権限ポリシー。 パラメーター値は *accept* または *drop* です。
- Priority: 優先順位。 優先順位は作成時間の降順で並べ替えられています。 ルールの優先度範囲は 1～100 です。 デフォルト値は 1 で、これが最も高い優先度です。 値が大きいほど、優先度が低くなります。
- NicType: ネットワークタイプ。 セキュリティグループ権限では (つまり、SourceGroupId を指定して、SourceCidrIp を指定しない) 、NicType を *intranet* と指定する必要があります。
- 説明:
  - IpProtocol: IP プロトコル。 値: *tcp*、*udp*、*icmp*、*gre* または *all*。 値 "all" はすべてのプロトコルを指します。
  - PortRange: IP プロトコルに関連するポート番号の範囲。
    - IpProtocol の値が *tcp* または *udp* の場合は、ポート番号の範囲は 1～65535 です。 形式は "開始ポート番号/終了ポート番号" とする必要があります。 たとえば、 "1/200" はポート範囲が 1～200 であることを示しています。 入力値を "200/1" とすると、インターフェイスを呼び出したときにエラーが報告されます。
    - IpProtocol の値が *icmp*、*gre*、*all* の場合は、ポート番号範囲は "-1/-1" であり、ポート番号に制限がないことを示しています。
  - セキュリティグループ権限を採用する場合は、SourceGroupId (つまり、送信元セキュリティグループ ID) を指定する必要があります。 この場合は、この権限がクロスアカウント権限であるかどうかに基づいて SourceGroupOwnerAccount の設定を選択可能です。 SourceGroupOwnerAccount は、送信元セキュリティグループが属するアカウントを示しています。
  - CIDR 権限を採用する場合は、SourceCidrIp を指定する必要があります。 SourceCidrIp は送信元 IP アドレスセグメントであり、CIDR 形式にする必要があります。

インバウンドリクエストを許可するルールの作成

コンソールまたは API を使用してセキュリティグループを作成する場合、デフォルトのインバウンドルールは *deny all* です。つまり、デフォルトではインバウンドリクエストはすべて拒否さ

れます。 これはすべての状況に当てはまるわけではないため、適切にインバウンドルールを設定する必要があります。

インターネットのポート 80 を有効にして外部アプリケーションに HTTP サービスを提供する必要がある場合は、インバウンドリクエストをすべて許可するために、IP ネットワークセグメントに制限をかけるのではなく `0.0.0.0/0` と設定します。 このために、コンソールパラメーターがかっこの外側にあり、OpenAPI パラメーターがかっこの内側にある以下のプロパティを参照します (パラメーターが両方とも同じであっても違いは生じません) 。

- NIC タイプ (NicType): インターネット (internet) 。 VPC の場合は、「intranet」と入力して、EIP を介したインターネットアクセスを実装します。
- アクション (Policy): 許可 (accept)
- ルールの方向 (NicType): インバウンド
- プロトコル種別 (IpProtocol): TCP (tcp)
- ポート範囲 (PortRange): 80/80
- 許可オブジェクト (SourceCidrIp): 0.0.0.0/0
- 優先度 (Priority): 1

 注 :
これらの推奨値はインターネットにのみ適用されます。 イントラネットリクエストの場合、CIDR ネットワークセグメントの使用は推奨しません。 「クラシックネットワークのイントラネットセキュリティグループルールでは CIDR または IP 許可は不使用」をご参照ください。

インバウンドリクエストを拒否するルールの作成

インバウンドリクエストを拒否するには、優先度の低い拒否ポリシーを設定するだけです。 このようにして、必要に応じてより高い優先度を持つ別のルールを設定して、このルールを上書きします。 たとえば、ポート 6379 へのアクセスを拒否する方法を以下に説明します。

- NIC タイプ (NicType): イントラネット (intranet)
- アクション (Policy): 禁止 (drop)
- ルールの方向 (NicType): インバウンド
- プロトコル種別 (IpProtocol): TCP (tcp)
- ポート範囲 (PortRange): 6379/6379
- 許可オブジェクト (SourceCidrIp): 0.0.0.0/0
- 優先度 (Priority): 100

クラシックネットワークのイントラネットセキュリティグループルールでは CIDR または IP 権限は使用しません。

クラシックネットワークの ECS インスタンスの場合、デフォルトではイントラネットのインバウンドルールは有効になっていません。 イントラネット権限には常に注意します。

> 注 :
> セキュリティ上の理由から、CIDR ネットワークセグメントに基づく権限を有効にすることは推奨しません。

エラスティックコンピューティングの場合、イントラネット IP アドレスは頻繁に変わり、IP アドレスがマッピングされるネットワークセグメントが動的に変わります。 このため、クラシックネットワークではセキュリティグループを介してイントラネットアクセスを許可することのみ推奨します。

たとえば、sg-redis セキュリティグループ内に Redis クラスターを構築し、特定のコンピューター (sg-web 内のコンピューターなど) にだけ、この Redis クラスターのサーバーへのアクセスを許可する場合、CIDR を設定する必要はありません。 代わりに、インバウンドルールを追加して、関連するセキュリティグループ ID を指定します。

- NIC タイプ (NicType): イントラネット (intranet)
- アクション (Policy): 許可 (accept)
- ルールの方向 (NicType): インバウンド
- プロトコル種別 (IpProtocol): TCP (tcp)
- ポート範囲 (PortRange): 6379/6379
- 許可オブジェクト (SourceGroupId): sg-web
- 優先度 (Priority): 1

VPC のインスタンスの場合、複数の VSwitch を使用して IP アドレス範囲の計画を立てている場合は、セキュリティグループのインバウンドルールとして CIDR 設定を使用します。 ただし、VPC ネットワークセグメントがはっきりしない場合は、インバウンドルールのセキュリティグループに優先度付けすることを推奨します。

相互通信が必要な ECS インスタンスの同一セキュリティグループへの追加

単一 ECS インスタンスは最大 5 つのセキュリティグループに参加し、同一セキュリティグループ内の ECS インスタンスはイントラネットを介して相互通信します。 計画中に複数のセキュリティグループを作成し、複数のセキュリティルールを直接設定するのが複雑すぎる場合は、セキュリティグループを作成し、イントラネット通信が必要なインスタンスを追加します。

セキュリティグループが異なれば、ネットワークの種類も異なります。 さらに重要なことに、クラシックネットワークの ECS インスタンスは、クラシックネットワーク用に作成したセキュリティグループにしか参加できません。 VPC 内の ECS インスタンスは、同一 VPC 用に作成したセキュリティグループにのみ参加可能です。

さらに、セキュリティグループルールの設定が非常に面倒になるため、ECS インスタンスをすべて同一セキュリティグループに追加することは推奨しません。 大規模または中規模のアプリケーションの場合、各サーバーグループには異なるロールがあり、インバウンドおよびアウトバウンドリクエストを合理的な方法で計画することが重要です。

コンソールで、 セキュリティグループに参加の説明に従ってセキュリティグループにインスタンスを追加します。

Alibaba Cloud OpenAPI に馴染みがある場合は、OpenAPI によってバッチ操作を実行します。 詳細は、「OpenAPI を使用した ECS インスタンスのエラスティックな管理」をご参照ください。 対応する Python スニペットは次のとおりです。

```
 def   join_sg ( sg_id ,  instance_i  d ):
     request  =  JoinSecuri  tyGroupReq  uest ()
     request . set_Instan  ceId ( instance_i  d )
     request . set_Securi  tyGroupId ( sg_id )
     response  =  _send_requ  est ( request )
     return   response
#  send   open   api   request
 def   _send_requ  est ( request ):
     request . set_accept  _format (' json ')
     try :
         response_s  tr  =  clt . do_action ( request )
         logging . info ( response_s  tr )
         response_d  etail  =  json . loads ( response_s  tr )
         return   response_d  etail
     except   Exception   as   e :
         logging . error ( e )
```

セキュリティグループからの ECS インスタンスの削除

ECS インスタンスを不適切なセキュリティグループに追加すると、サービスが公開されたりブロックされたりする可能性があります。 この場合、セキュリティグループから ECS インスタンスを削除します。 ただし、削除前に、ECS インスタンスが別のセキュリティグループに追加されていることを確認する必要があります。

 注：

インスタンスと現在のセキュリティグループ内の他のインスタンスとの間で相互通信障害が発生する可能性があるため、削除前に十分なテストを実行することを推奨します。

対応する Python スニペットは次のとおりです。

```
 def   leave_sg ( sg_id ,  instance_i  d ):
     request  =  LeaveSecur  ityGroupRe  quest ()
     request . set_Instan  ceId ( instance_i  d )
     request . set_Securi  tyGroupId ( sg_id )
     response  =  _send_requ  est ( request )
     return   response
#  send   open   api   request
 def   _send_requ  est ( request ):
     request . set_accept  _format (' json ')
     try :
         response_s  tr  =  clt . do_action ( request )
         logging . info ( response_s  tr )
         response_d  etail  =  json . loads ( response_s  tr )
         return   response_d  etail
     except   Exception   as   e :
         logging . error ( e )
```

セキュリティグループの適切な名前とタグの定義

セキュリティグループの合理的な名前と説明は、複雑なルールの組み合わせの意味をすばやく識別するのに役立ちます。 セキュリティグループの名前と説明を必要に応じて変更します。

また、セキュリティグループにタグを設定することも可能です。 タグでグループ化して自身のセキュリティグループを管理します。 タグを設定するには、コンソールまたは API を使用してタグを直接設定します。

不要なセキュリティグループの削除

セキュリティグループのセキュリティルールは、ホワイトリストおよびブラックリスト項目と似ています。 したがって、不要なセキュリティグループに 誤って ECS インスタンスを追加することによる予期しない問題の発生を防ぐため、不要なセキュリティグループを削除することを推奨します。

1.2 セキュリティグループのベストプラクティス (パート 3)

実際には、すべてのインスタンスを同一セキュリティグループに配置できるため、初期の設定作業負荷は軽減されます。 しかし、長期的にはビジネスシステムの相互作用は複雑になり、制御不能になります。 セキュリティグループを変更すると、ルールの追加または削除による影響範囲を明確に特定できなくなります。

セキュリティグループの合理的な計画と差別化により、システムの調整、アプリケーションによって提供されるサービスの整理、およびさまざまなレイヤーでアプリケーションの配置が容易になります。 異なるセキュリティグループを計画し、さまざまな業務に対して異なるセキュリティグループルールを設定することを推奨します。

異なるセキュリティグループの区別

- インターネット上とイントラネット上で、ECS インスタンスに対する異なるセキュリティグループの使用

  インターネットサービスを提供する ECS インスタンスは、外部アクセス用の一部のポートの公開 (80 や 443 など) 、またはポート転送ルールの提供 (インターネット IP アドレス、EIP アドレス、NAT ポートに対する転送ルールによって設定されたインスタンスなど) のいずれかによって、アプリケーションをインターネットに公開します。

  上記の 2 つのシナリオの場合、関連するセキュリティグループは最も厳密なルールを採用する必要があります。 インターネットへの接続を最初は拒否することを推奨します。 具体的には、80 や 443 などの外部サービスの提供に必要なポートを除き、デフォルトでポートとプロトコルをすべて無効にする必要があります。 セキュリティグループにはインターネットアクセスを提供する ECS インスタンスしか含まれていないため、セキュリティグループルールを調整する方が簡単です。

  インターネットアクセスを提供する ECS インスタンスのグループの場合、それらの責任を明確化、単純化して、同一インスタンスで他の外部サービスが提供されないようにする必要があります。 たとえば、MySQL、Redis などの場合は、インターネットアクセスを無効にする ECS インスタンスにそのようなサービスをインストールし、セキュリティグループ権限付与によってサービスへのアクセスを有効にすることを推奨します。

  他のアプリケーションのインスタンスとしてセキュリティグループ SG_CURRENT にある、インターネットアクセスを提供する ECS インスタンスがあると仮定します。 以下の手順を実行して変更を加えます。

  1. 80 や 443 など、現在のインターネットサービスで公開されているポートとプロトコルを整理します。
  2. SG_WEB などの新しいセキュリティグループを作成し、対応するポートとルールを追加します。

     注 :

アクション: 許可; プロトコル種別: すべて; ポート範囲: 80/80; 権限付与オブジェクト: 0.0.0.0/0; アクション: 許可; プロトコル種別: すべて; ポート範囲: 443/443; 権限付与オブジェクト: 0.0.0.0/0。

3. セキュリティグループ SG_CURRENT を選択し、セキュリティグループ権限付与のルールを追加します。つまり、SG_WEB 内のリソースが SG_CURRENT 内のリソースにアクセスすることを許可します。

    注 :
   アクション: 許可; プロトコル種別: すべて; ポート範囲: -1/-1; 権限付与オブジェクト: SG_WEB; 優先度: 実際の状況に応じて [1 ～ 100] から選択します。

4. ECS_WEB_1 を新しいセキュリティグループに追加します。 これはセキュリティグループを切り替える必要のあるインスタンスです。

   a. ECS コンソールで、[セキュリティグループ] をクリックします。
   b. [SG_WEB] > [インスタンスの管理] > [インスタンスの追加] をクリックします。 インスタンス ECS_WEB_1 を新しいセキュリティグループ SG_WEB に追加します。 ECS_WEB_1 が正常に機能することを確認します。

5. 元のセキュリティグループからインスタンス ECS_WEB_1 を削除します。

   a. ECS コンソールで、[セキュリティグループ] をクリックします。
   b. [SG_WEB] > [インスタンスの管理] > [インスタンスの追加] をクリックします。 ECS_WEB_1 を選択して SG_CURRENT から削除します。 トラフィックとネットワークが正常であることを確認します。
   c. エラーが発生した場合は、ECS_WEB_1 を元のセキュリティグループ SG_CURRENT に追加します。 SG_WEB のポートが予想どおりに公開されているかどうかを確認し、それに応じて調整します。

6. セキュリティグループに他の変更を加えます。

- 異なるアプリケーションに対する異なるセキュリティグループの使用

  本番環境では、異なるオペレーティングシステムは一般的に、同一アプリケーショングループに属して負荷分散サービスを提供することはありません。 異なるサービスを提供することは、公開されたポートが拒否されたポートとは異なることを意味します。 したがって、異なる

オペレーティングシステムを持つインスタンスを異なるセキュリティグループに属させることを推奨します。

たとえば、TCP ポート 22 は Linux で SSH を実装するために公開され、TCP ポート 3389 は Windows でリモートデスクトップ接続を実装するために公開されます。

さらに、同じ種類のイメージを持つが異なるサービスを提供するインスタンスについては、イントラネット経由で互いにアクセスする必要がない場合は、それらを異なるセキュリティグループに入れることを推奨します。 これにより、ルールをできるだけ単純にできるので、セキュリティグループルールの分離と将来の変更が容易になります。

新しいアプリケーションを計画して追加するときは、異なる VSwitch を分割してサブネットを設定するのとは別に、セキュリティグループを合理的に編成する必要があります。 ネットワークセグメントとセキュリティグループを使用して、サービスプロバイダーまたはコンシューマーとして自身を区別します。

具体的な変更手順については、上記の操作をご参照ください。

- 本番環境とテスト環境用の異なるセキュリティグループの使用

  システムをよりよく分離するには、実際の開発中に複数のテスト環境と 1 つのオンライン環境を構築します。 ネットワークの分離をよくするには、異なる環境に異なるセキュリティポリシーを設定し、テスト環境への変更をオンライン環境に同期しないようにする必要があります。オンラインサービスの安定性に影響を与える可能性があるためです。

  異なるセキュリティグループを作成することで、アプリケーションのアクセスドメインを制限し、本番環境とテスト環境の間で相互運用がされないようにします。 また、異なるテスト環境に対して異なるセキュリティグループを作成することで、テスト環境間で干渉されなくなり、開発効率が向上します。

インターネットアクセスを必要とするサブネットまたはインスタンスのみへのインターネットアドレスの割り当て

クラシックネットワークか VPC かにかかわらず、インターネットアドレスの合理的な割り当てにより、システムのインターネット管理が容易になり、攻撃のリスクが減ります。 VPC の場合、VSwitch を作成するときに、インターネットアクセスが必要なインスタンスの IP セグメントを、複数の専用 VSwitch (サブネット CIDR) に配置することを推奨します。 これにより、監査と差別化が容易になり、インターネットに偶発的にアクセスしないようになります。

ほとんどの分散アプリケーションには、異なるレイヤーとグループがあります。 インターネットアクセスを提供しない ECS インスタンスの場合は、インターネットアドレスを提供しないようにします。 インターネットアクセスを提供するインスタンスが複数ある場合は、 Server Load

Balancer を設定して、インターネットサービスのトラフィックを分散させることを推奨します。それにより、システムの可用性が向上し、障害点が 1 つも発生しないようになります。

インターネットアクセスを必要としない ECS インスタンスの場合は、インターネットアドレスを割り当てないようにします。 VPC では、ECS インスタンスがインターネットにアクセスする必要がある場合は、 NAT ゲートウェイ を使用し、VPC でインターネットアドレスなしで、ECS インスタンスのインターネットプロキシサービスを提供します。 対応する SNAT ルールを設定するだけで、具体的な CIDR セグメントまたはサブネットがインターネットにアクセスできるようにします。 具体的な設定については、「SNAT」をご参照ください。 このようにして、アウトバウンドアクセスのみが必要な場合に EIP (Elastic IP) アドレスが割り当てられた後のインターネットへのサービスの公開を回避します。

最小限の原則

セキュリティグループはホワイトリストとして機能する必要があります。 したがって、開いて公開するポートをできるだけ少なくし、割り当てるインターネットアドレスをできるだけ少なくするようにします。 インターネットアドレスを割り当てたり EIP をバインドしたりすることで、オンラインインスタンスにアクセスしてトラブルシューティングを行うのは簡単になりますが、結果的にインスタンス全体をインターネットにさらすことになります。 より安全なポリシーは、Jump Server を使用して IP アドレスを管理することです。

Jump Server の使用

Jump Server ははるかに強い権限を持っているので、関連する操作をツールによって適切に記録し、監査する必要があります。 さらに、VPC で Jump Server 専用 VSwitch を選択し、対応する EIP または NAT ポート転送テーブルを提供することを推奨します。

まず、Linux の TCP 22 や Windows の RDP 3389 など、対応するポートを有効にして、専用のセキュリティグループ SG_BRIDGE を作成します。 インバウンドアクセスを制限するには、会社のインターネット出口ポートへの権限付与オブジェクトを制限して、スキャンおよびアクセスされる可能性を低くします。

その後、このセキュリティグループに Jumper Server インスタンスを追加します。 この Jumper Server が他の適切なインスタンスにアクセスするために、適切なグループ権限付与を設定します。 たとえば、SG_CURRENT のルールを追加して、SG_BRIDGE が特定のポートとプロトコルにアクセスできるようにします。

SSH 通信に Jumper Server を使用する場合は、 パスワードの代わりにログイン用 SSH キーペアを使用します。

要約すると、セキュリティグループを適切に計画することで、アプリケーションの拡張が簡単になり、システムがより安全になります。

1.3 ECS データセキュリティのベストプラクティス

ここでは、O&M の観点から ECS インスタンス用のデータセキュリティを実装する方法を紹介します。

対象ユーザー

この内容は、Alibaba Cloud が初めての個人や企業に適用されます。

目次

- データの定期的なバックアップ
- セキュリティドメインの適切な設計
- セキュリティグループのルールの設定
- ログインパスワードの設定
- サーバーポートセキュリティ
- アプリケーション脆弱性の保護
- セキュリティ情報収集

データの定期的なバックアップ

耐障害性の基盤として、データのバックアップは、システム障害、操作エラー、およびセキュリティ問題によるデータ損失のリスクを減らすことを目的としています。 ECS インスタンスは、スナップショットバックアップ機能を備えています。 スナップショット機能を正しく使用することにより、ほとんどのユーザーのデータバックアップ要件が満たされます。 実際の業務ニーズに応じて独自のバックアップポリシーをカスタマイズすることを推奨します。 [スナップショットの作成] または [自動スナップショットポリシーの作成] を選択し、ポリシーを特定のディスクに適用します。 自動スナップショットを毎日取り、少なくとも 7 日間保存することを推奨します。 よいバックアップ習慣は、迅速なデータ回復と、障害発生時の損失の最小化に役立ちます。

セキュリティドメインの適切な設計

SDN (ソフトウェア定義ネットワーク) テクノロジーに基づいて開発された VPC を使用して、企業内でセキュリティレベルが異なるサーバーを分離するプライベートネットワークを構築し、サーバーが相互接続ネットワークを介して相互に影響を与えるのを防ぐことが可能です。

VPC を作成し、IP アドレス範囲、ネットワークセグメント、ルートテーブル、およびゲートウェイを設定することを推奨します。 インターネットから完全に分離されているイントラネットに重

要なデータを格納します。 EIP (Elastic IP) アドレスまたは Jumper Server を使用して、日々の O&M でデータを管理します。

セキュリティグループのルールの設定

セキュリティ分離の重要な手段として、セキュリティグループを使用して、1 つ以上の ECS インスタンスに対してネットワークアクセス制御を設定します。 セキュリティグループを使用すると、インスタンスレベルでファイアウォールポリシーを設定し、ネットワーク層でインスタンスのアクティブアクセスとパッシブアクセスをフィルタリングします。 具体的には、ポート上のインバウンドとアウトバウンドのアクセスを制限し、IP アドレスへのアクセスを許可し、攻撃を減らし、インスタンスのセキュリティを強化します。

たとえば、Linux のデフォルトではリモートポートは 22 であり、インターネットに直接開放してはいけません。 セキュリティグループを設定して、インスタンスにアクセスするための固定 IP アドレスの許可など、ECS インスタンスへのインターネットアクセスを制御します。 セキュリティグループの詳細については、「アプリケーション事例」をご参照ください。 より高い要件がある場合は、サードパーティ製 VPN プロダクトを使用してログインデータを暗号化することも可能です。 その他のソフトウェアについては、「 Alibaba Cloud Market」をご参照ください。

ログインパスワードの設定

脆弱なパスワードは最も一般的な脆弱性の 1 つであり、容易に悪用される可能性が高いため、データ漏えいの主な原因となっています。 サーバーのパスワードは 8 文字以上とし、大文字と小文字、数字、特殊文字を含めて複雑にすることを推奨します。 また、パスワードは定期的に変更する必要があります。

サーバーポートセキュリティ

サーバーがインターネットサービスを提供している限り、対応するポートはインターネットに公開されます。 セキュリティ管理の観点からは、ポートをより多く開けるほど、システムリスクがより高くなることを意味します。 インターネットに必要な数のポートのみ開くことを推奨します。 共通ポートをカスタマイズポート (ポート 30000 以上) に変更し、アクセス制御をサービスポートに実装する必要があります。

たとえば、データベースサービスをイントラネットに限定し、インターネットからのアクセスを防ぐことを推奨します。 インターネットからデータベースに直接アクセスする必要がある場合は、接続ポートを 3306 からそれより大きなポートに変更し、業務ニーズに応じて関連する IP アドレスを許可する必要があります。

アプリケーション脆弱性の保護

アプリケーションの脆弱性は、Web アプリケーション、キャッシュ、データベース、およびストレージのデータに不正にアクセスするためにハッカーが悪用する可能性のある、セキュリティ上の欠陥です。 一般的なアプリケーションの脆弱性には、SQL インジェクション、XSS 攻撃、Web シェル、バックドア、コマンドインジェクション、不正な HTTP リクエスト、一般的な Web サーバーの脆弱性攻撃、コアファイルへの不正アクセス、パストラバーサルなどがあります。 これらの脆弱性はシステムの脆弱性とは異なり、修正が困難です。 初期設計時にアプリケーションのセキュリティが保証されない場合、このような脆弱性によりサーバーが攻撃される可能性があります。 そのため、 WAF (Web Application Firewall) をインストールして、さまざまな攻撃を防ぎ、Web サイトのセキュリティと可用性を確保することを推奨します。

1.4 クラシックネットワーク内のインスタンス同士のアクセスを設定する方法

セキュリティグループはインスタンスレベルのファイアウォールです。 インスタンスのセキュリティを確保するために、セキュリティグループのルール設定に関して最低限の権限付与原則を守る必要があります。 ここでは、インスタンスのイントラネット相互通信を可能にする安全な方法を 4 つ紹介します。

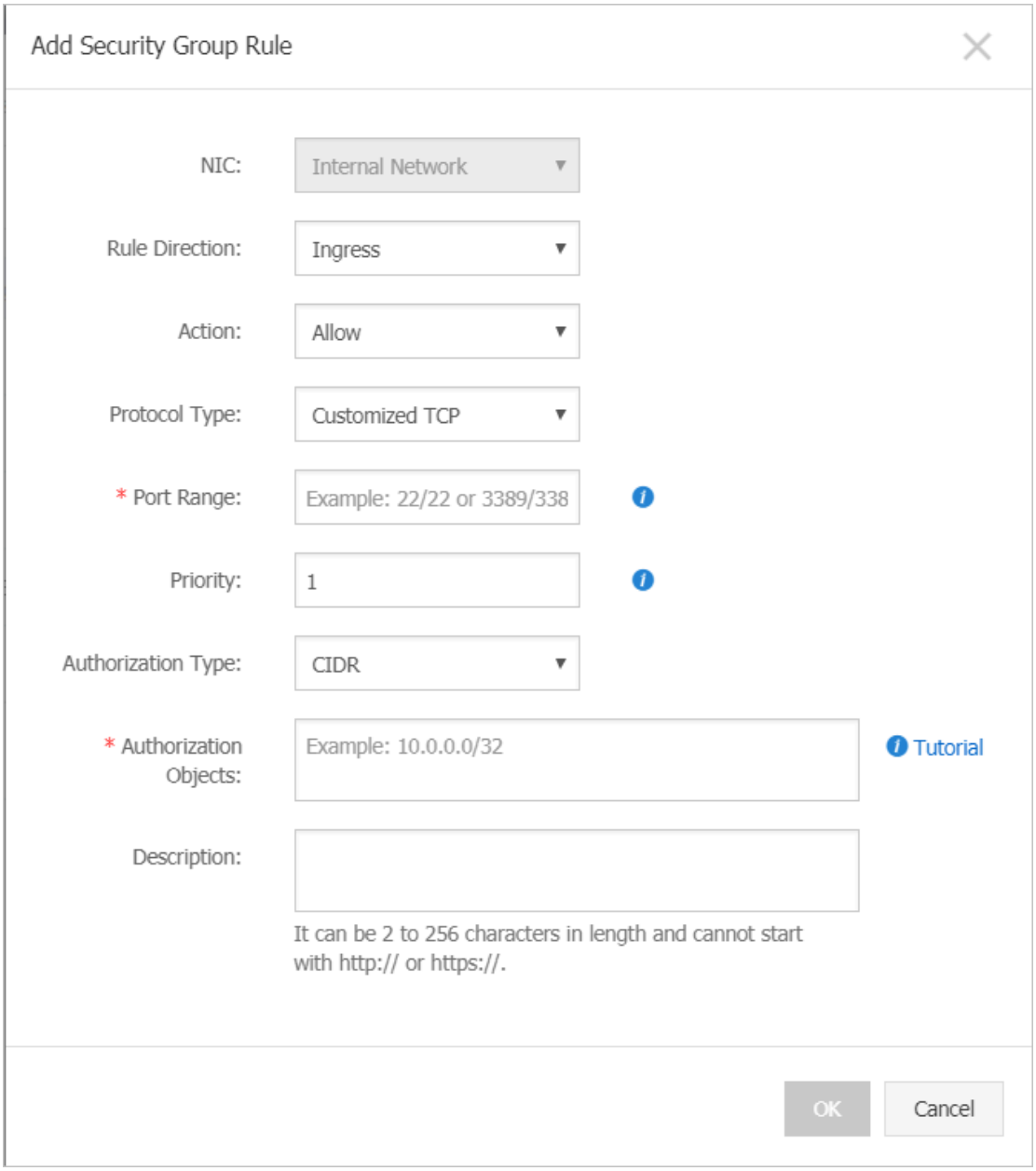
方法 1. 単一の IP アドレスへのアクセス権限付与

- アプリケーションシナリオ: イントラネットを介した少数のインスタンスの相互通信。
- 長所: IP アドレスへのアクセス権限を付与することで、セキュリティグループのルールが明確になり、理解しやすくなります。
- 短所: イントラネットを介して多数のインスタンスが相互にアクセスする必要がある場合は、セキュリティグループのルールの割り当てが 100 に制限されます。 さらに、メンテナンスの作業負荷も高くなります。

- 設定:

  1. 相互通信を必要とするインスタンスを選択し、[セキュリティグループ] をクリックします。
  2. 該当するセキュリティグループを選択し、[ルールの追加] をクリックします。
  3. [Ingress] をクリックし、[セキュリティグループのルールの追加] をクリックします。
  4. 以下の説明に従ってセキュリティグループのルールを追加します。

     - 操作: 許可します。
     - プロトコルの種類: 必要に応じてプロトコルの種類を選択します。
     - ポートの範囲: 必要に応じてポートの範囲を設定します。 形式は “start port number/end port number” です。
     - 権限付与の種類: CIDR.
     - 権限付与オブジェクト: イントラネット相互通信向けの、予想されるイントラネット IP アドレスを入力します。 形式は *a.b.c.d/32* である必要があります。 サブネットマスクは */32* である必要があります。



方法 2. 同じセキュリティグループへの参加

- **アプリケーションシナリオ: アプリケーションのアーキテクチャが比較的単純である場合は、すべてのインスタンスを同じセキュリティグループに追加できます。 このようなインスタンスは、デフォルトでイントラネットを介して互いにアクセスできるため、特別なルールは必要ありません。**
- **長所: セキュリティグループのルールが明確になり、理解しやすくなります。**
- **短所: 単純なアプリケーションネットワークアーキテクチャにしか適用できません。 ネットワークアーキテクチャを調整した場合、権限付与方法もそれに応じて修正する必要があります。**

方法 3. 相互通信専用に作成されたセキュリティグループへのインスタンスのバインド

- アプリケーションシナリオ: 該当するインスタンスを相互通信専用のセキュリティグループにバインドすることができます。 この方法は、複数のアプリケーション層を持つネットワークアーキテクチャに適用可能です。
- 長所: この方法は容易に実装でき、インスタンス間の相互通信を迅速に確立できます。 これは、複雑なネットワークアーキテクチャに適用可能です。
- 短所: インスタンスは複数のセキュリティグループにバインドされる必要があり、セキュリティグループのルールが理解しにくくなります。
- 設定:
  1. “相互通信用のセキュリティグループ” という名前の新しいセキュリティグループを作成します。 新しいセキュリティグループにはルールは必要ありません。
  2. 該当するインスタンスを、新しく作成された “相互通信用のセキュリティグループ” に追加します。 同じセキュリティグループのインスタンスのデフォルトの機能であるため、インスタンスはイントラネットを介して相互接続されます。

方法4. セキュリティグループの権限付与

- アプリケーションシナリオ: ネットワークアーキテクチャが複雑で、異なるインスタンスに展開されたアプリケーションが異なるサービスの役割を持っている場合、セキュリティグループの権限付与を選択できます。
- 長所: セキュリティグループのルールが明確になり、理解しやすくなります。 さらに、相互通信はアカウントを越えて実装することができます。
- 短所: セキュリティグループのルールをたくさん設定する必要があります。

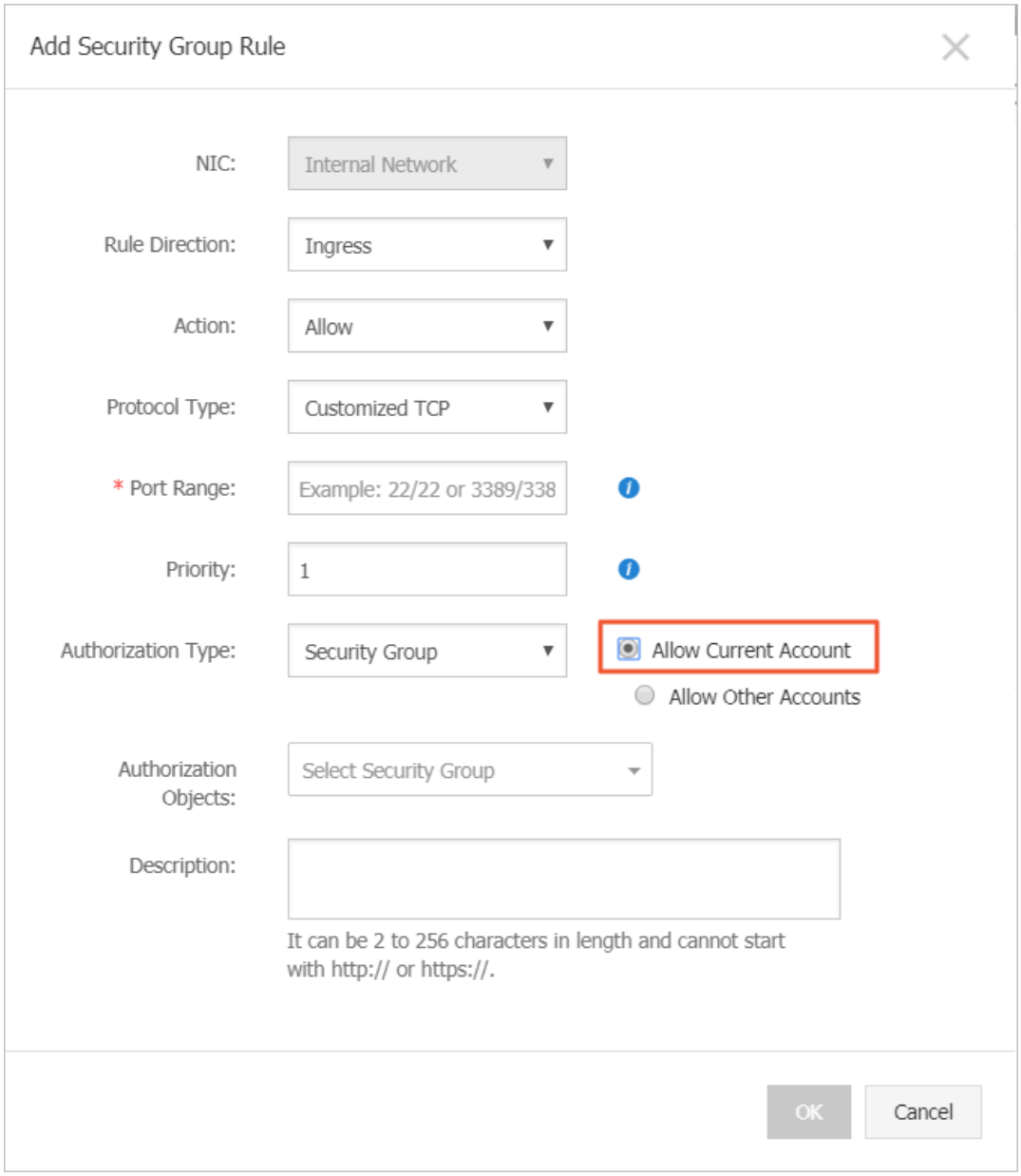
- 設定:

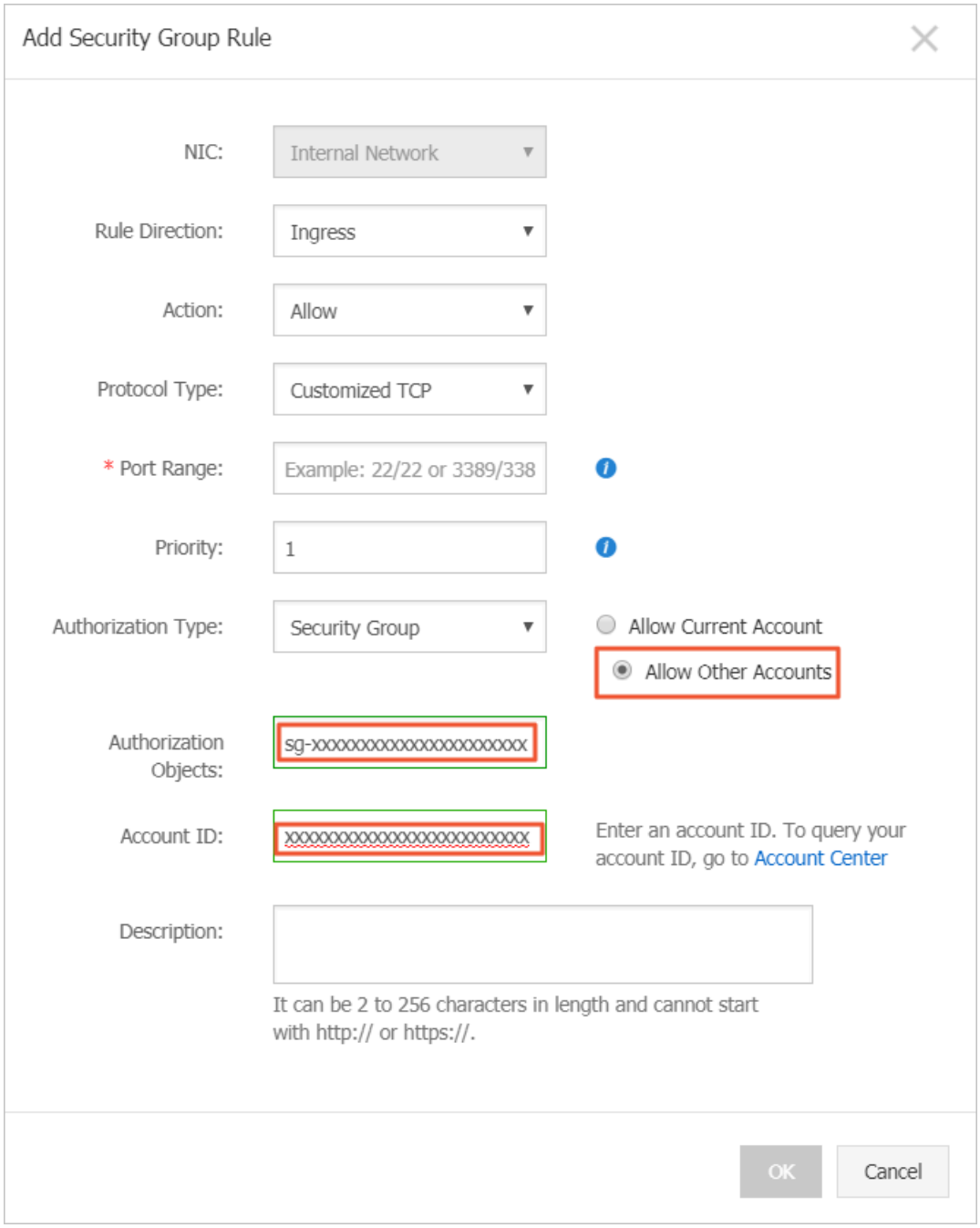
  1. 該当するインスタンスを選択し、[セキュリティグループ] ページに入ります。
  2. 該当するセキュリティグループを選択し、[ルールの追加] をクリックします。
  3. [Ingress] をクリックし、[セキュリティグループのルールの追加] をクリックします。
  4. 以下の説明に従って、セキュリティグループのルールを追加します。

     - 操作: 許可します。
     - プロトコルの種類: 必要に応じてプロトコルの種類を選択します。
     - ポートの範囲: 必要に応じて設定します。
     - 権限付与の種類: セキュリティグループ。
     - 権限付与オブジェクト:

       ■ 現在のアカウントを許可する: ネットワークの要件に基づいて、[権限付与されたオブジェクト]の中から、イントラネット相互通信用のピアインスタンスのセキュリティグループ ID を選択します。

       ■ 他のアカウントを許可する: [許可されたオブジェクト] の中のピアインスタンスのセキュリティグループ ID を入力します。 [アカウント ID] の中のピアアカウント IDを入力します。 [アカウント管理] > [セキュリティ設定] で照会できます。


Add Security Group Rule
NIC:
Internal Network
Rule Direction:
Ingress
Action:
Allow
Protocol Type:
Customized TCP
* Port Range:
Example: 22/22 or 3389/338
Priority:
1
Authorization Type:
Security Group
Allow Current Account
Allow Other Accounts
Authorization Objects:
Select Security Group
Description:
It can be 2 to 256 characters in length and cannot start with http:// or https://.
OK
Cancel



提案

早い段階でセキュリティグループによって付与されるアクセスが多すぎる場合、次の手順で権限付与範囲を狭めることを推奨します。

この図で、 0.0.0.0 を削除するとは、*0.0.0.0/0* アドレスセグメントからのインバウンドアクセスを許可する、元のセキュリティグループを削除するということです。

セキュリティグループが不適切に変更された場合、インスタンス間の通信が影響を受ける可能性があります。 相互通信の問題が発生したときにタイムリーに回復できるように、設定変更対象のセキュリティグループのルールをバックアップしてください。

セキュリティグループは、アプリケーションアーキテクチャ全体におけるインスタンスの役割をマップします。 アプリケーションアーキテクチャに基づいて、ファイアウォールのルールを計画することを推奨します。 たとえば、一般的な 3 層 Web アプリケーションアーキテクチャでは、3 つのセキュリティグループを計画し、それぞれアプリケーションまたはデータベースとともに展開されたインスタンスにバインドできます。

- Web 層のセキュリティグループ: ポート 80 を開く。
- アプリケーション層のセキュリティグループ: ポート 8080 を開く。
- DB 層のセキュリティグループ: ポート 3306 を開く。

1.5 既定のリモートアクセスポートの変更

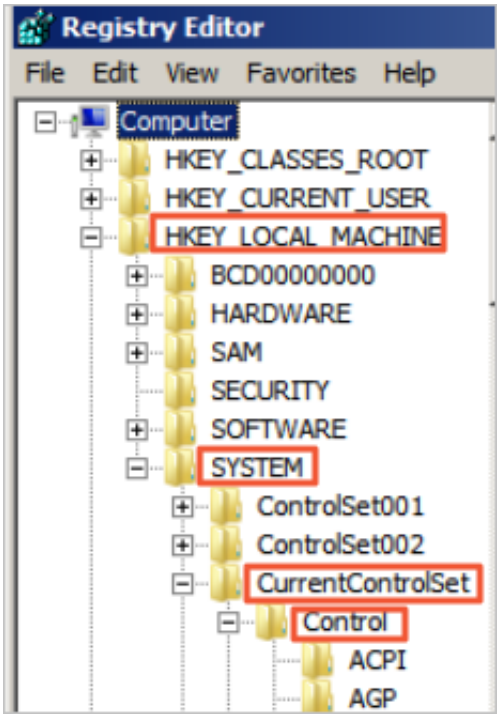
ここでは、Windows または Linux インスタンスのリモートポートを変更する方法について説明します。

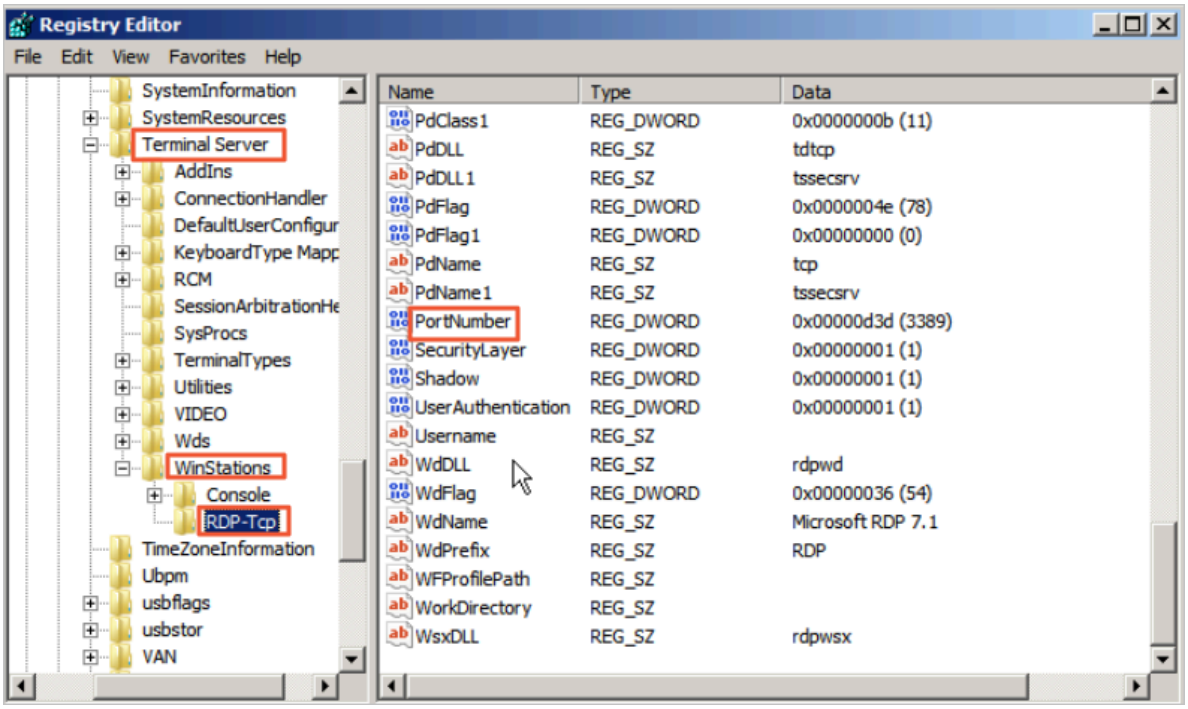
Windows インスタンスの既定のリモートポートの変更

このセクションでは、Windows Server 2008 を実行している Windows インスタンスのリモートポートを変更する方法について説明します。

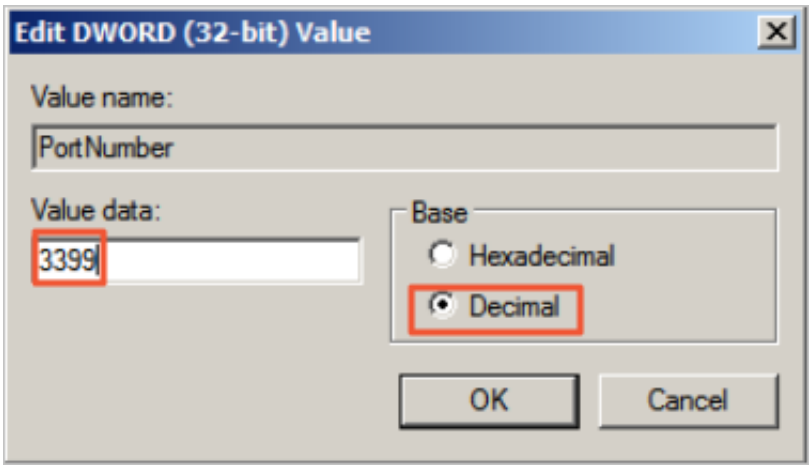
1. Windows インスタンスに接続します。
2. *regedit.exe* を実行してレジストリエディタを開きます。

3. レジストリエディタの左側のナビゲーションウィンドウで、" *HKEY_LOCAL_MACHINE \ System \ CurrentControlSet \ Control \ Terminal Server \ WinStations \ RDP-Tcp \ PortNumber* "を探します。

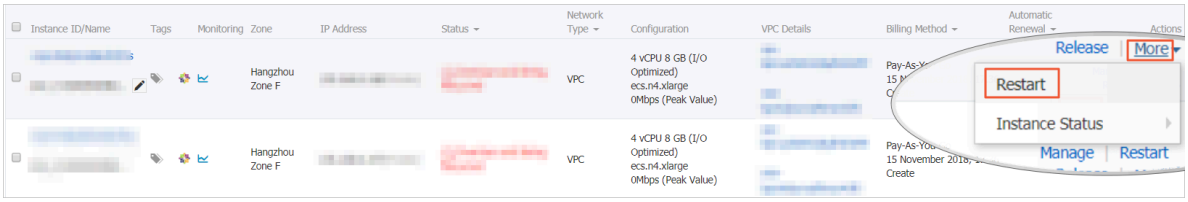




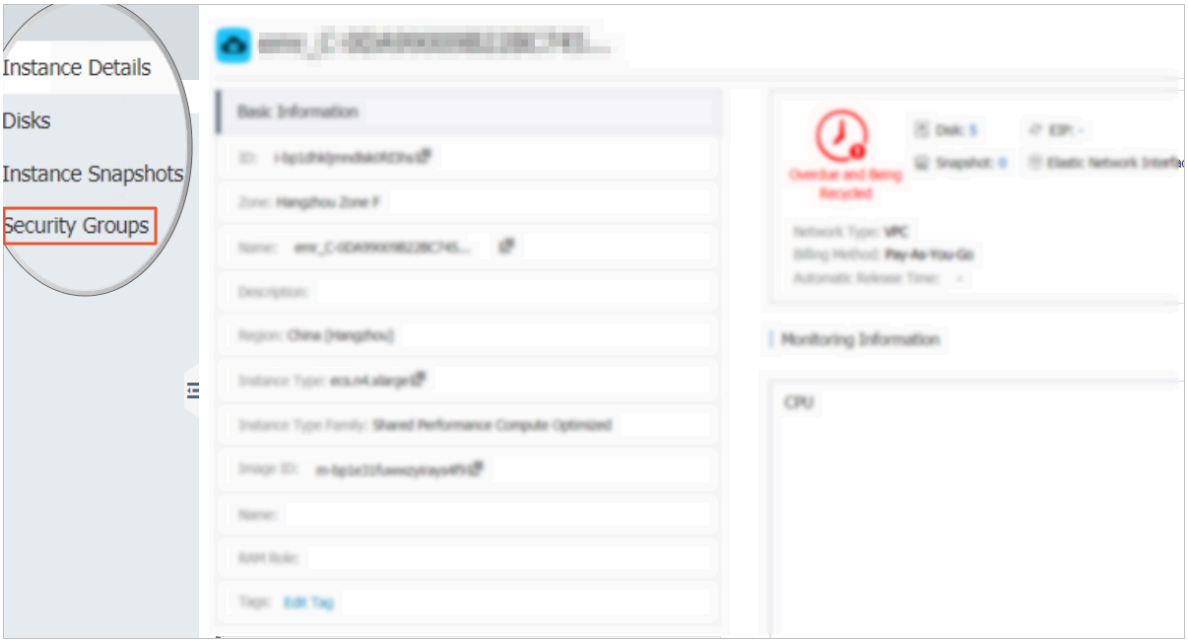
4. ダイアログボックスで、[10 進数] をオンにし、[値のデータ] フィールドに、新しいリモートポート番号として数字を入力します。この例では"3399"です。 [OK] をクリックします。



5. (オプション) ファイアウォールを有効にしている場合は、ファイアウォールの新しいポートを開きます。
6. ECS console にログインし、インスタンスを見つけ、[詳細] > [再起動] をクリックします。

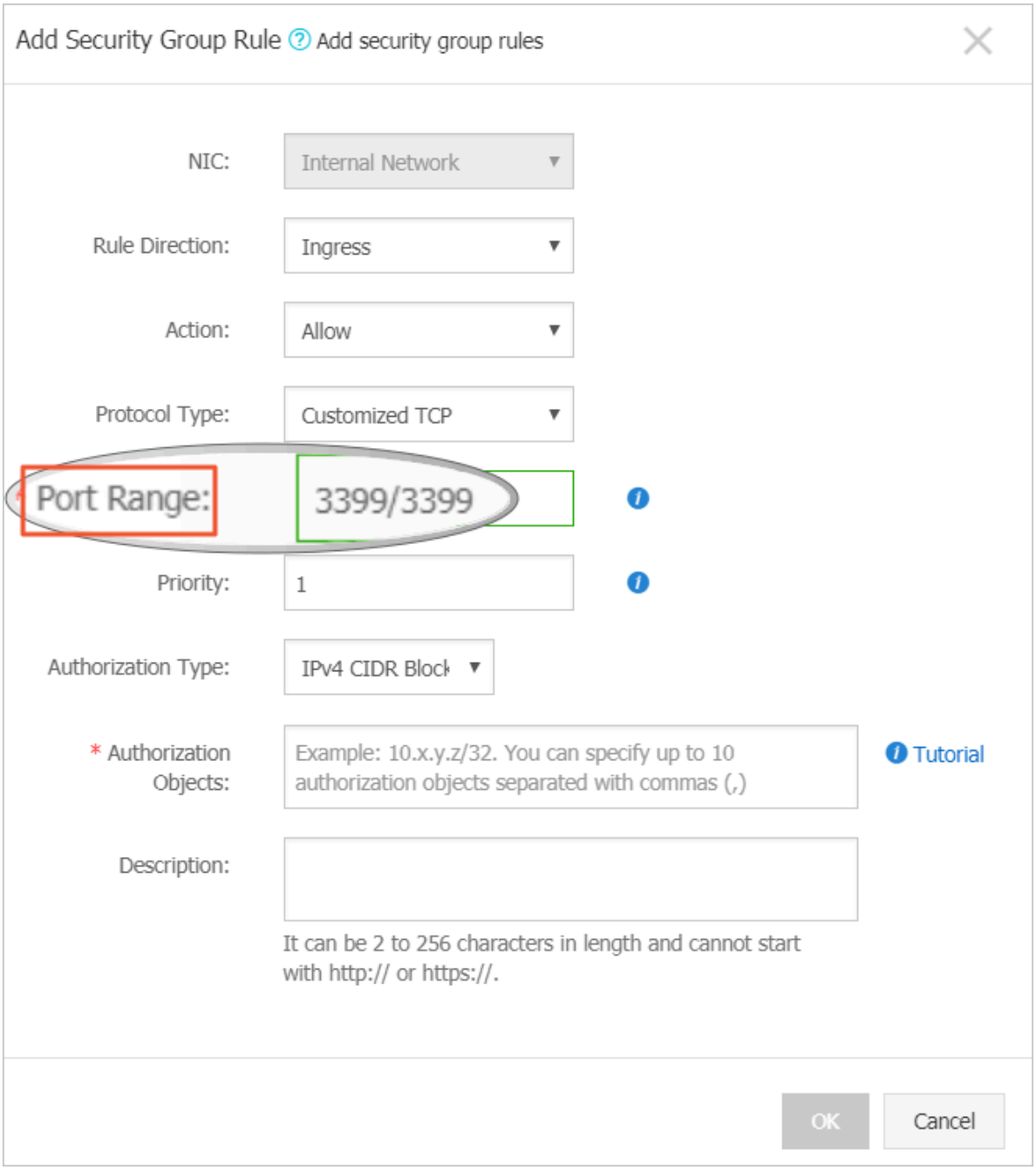


7. インスタンスを再起動した後、インスタンスの [管理] をクリックして [インスタンスの詳細] ページに入ります。 [セキュリティグループ] をクリックします。

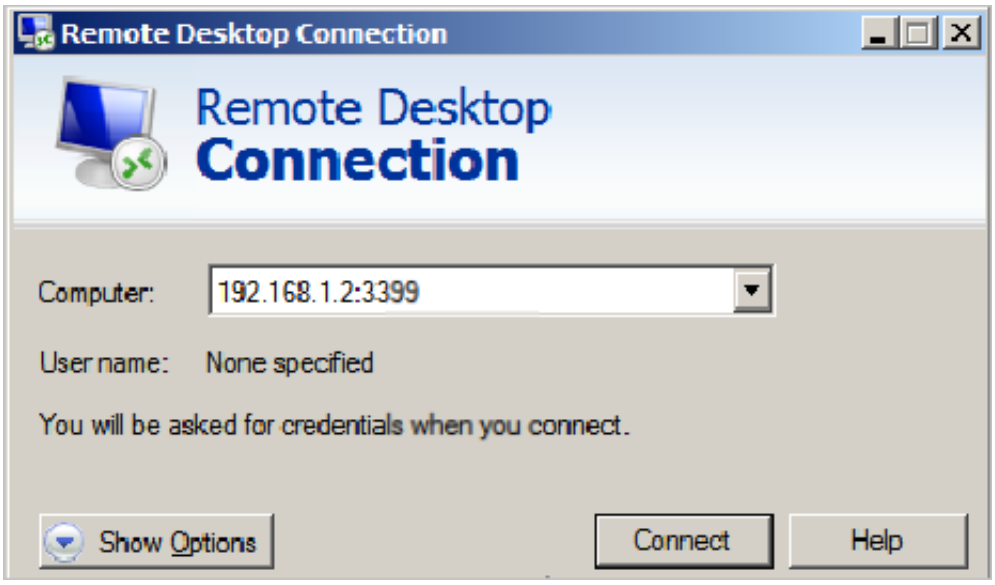


8. [セキュリティグループ] ページで、[ルールの追加] をクリックします。

9. [セキュリティグループのルール] ページで、[セキュリティグループのルールの追加] をクリックします。 新しいセキュリティグループのルールを追加して、新しいリモートポートへのアクセスを許可します。 セキュリティグループのルールの追加の詳細については、「セキュリティグループのルールの追加」をご参照ください。



10.末尾に新しいポート番号が付いた IP アドレスにアクセスして、インスタンスに接続します。
たとえば、この例では"192.168.1.2:3399" です。



注：
Mac のリモートデスクトップユーザーがアクセスに使用できるのは、既定のポート 3389 だけです。

Linux インスタンスの既定のリモートポートの変更

このセクションでは、CentOS 6.8 を実行している Linux インスタンスのリモートポートを変更する方法について説明します。

注：
ポート 22 を直接変更せずに、最初に新しい既定のリモートポートを追加します。 最初に 2 つのポートを設定し、テストが成功したら 1 つを削除します。 新しいポートを介してインスタンスに接続できない場合は、ポート 22 を使用して問題をデバッグできるようにします。

1. Linux インスタンスに接続します。
2. `vim / etc / ssh / sshd_confi g` を実行します。
3. キーボードの I キーを押して編集モードに入ります。 新しいリモートサービスポートを追加します (例えば、ポート 1022)。 `### 22` の下の`### 1022` に入ります。
4. Esc キーを押して「: `wq` 」と入力し、編集を終了します。

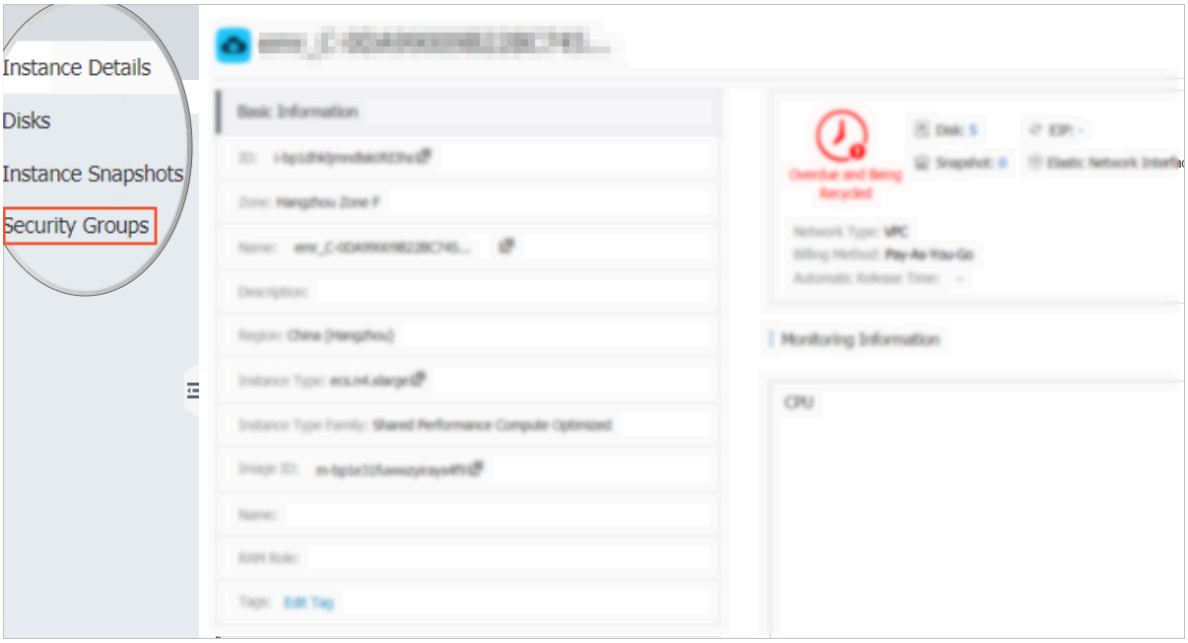
5. 次のコマンドを実行してインスタンスを再起動します。 これで、ポート 22 とポート 1022 を介して Linux インスタンスにログインできるようになります。

```
/ etc / init . d / ssh   restart
```

6. (オプション) ファイアウォールを設定します。 CentOS 7 より前の Linux バージョンを使用していて、ファイアウォール iptables を有効にしている場合、既定では iptables がアクセスを遮断しないことに注意します。 iptables のルールを設定した場合は、 `iptables - A INPUT - p tcp -- dport 1022 - j ACCEPT` を実行してファイアウォールを設定します。 次に、 `service iptables restart` を実行してファイアウォールを再起動します。

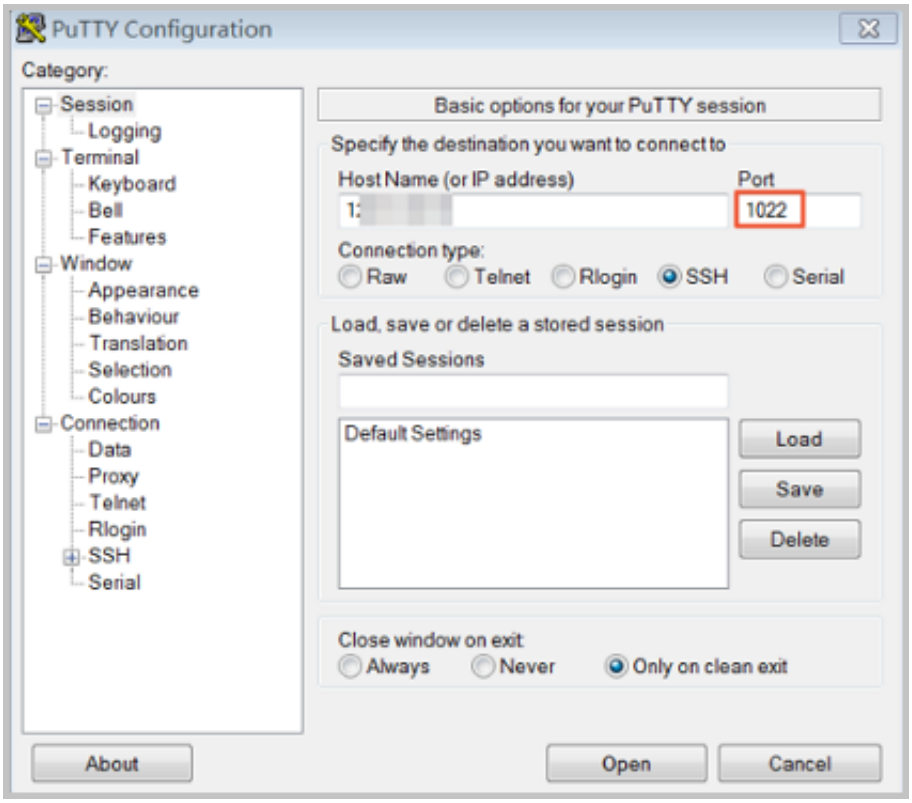
> 注 :
> 既定では、ファイアウォールは CentOS 7 以降のバージョンにインストールされています。 firewalld.service を有効にしている場合は、コマンド firewall-cmd --add-port=1022/tcp --permanent を実行して、TCP ポート 1022 を開きます。 成功が返されれば、TCP ポート 1022 が開きます。

7. ECS console にログインし、インスタンスを見つけ、[管理] を選択します。
8. [インスタンスの詳細] ページに入ります。 [セキュリティグループ] をクリックします。



9. [セキュリティグループ] ページで、[ルールの追加] をクリックします。
10. [セキュリティグループのルール] ページで、[セキュリティグループのルールの追加] をクリックします。 新しいセキュリティグループのルールを追加して、新しいリモートポートへのアクセスを許可します。 セキュリティグループのルールの追加の詳細については、「セキュリティグループのルールの追加」をご参照ください。

11.SSH ツールを使用して新しいポートに接続し、既定のリモートポートが正常に変更されたかどうかをテストします。 インスタンスにログインする際に、[ポート] に新しいポート番号を入力します。この例では "1022" です。



12.ポート 1022 経由でインスタンスに正常に接続したら、 `vim / etc / ssh / sshd_confi g` を再び実行してポート 22 を削除します。

13.`/ etc / init . d / sshd` を実行してインスタンスを再起動します。既定のリモートポートが正常に変更されます。 末尾に新しいポート番号が付いた IP アドレスにアクセスして、インスタンスに接続します。

1.6 Windows インスタンスでのログの使用

ログは、システム内のハードウェアとソフトウェアの記録、およびシステムエラー情報です。 それらの情報は、システムイベントの監視にも使用されます。 サーバー侵入またはシステム (アプリケーション) エラー発生時、管理者はログを使用して問題をすばやく特定し、問題を迅速に解決することができ、作業効率とサーバーのセキュリティが大幅に向上します。 Windows ログは、システムログ、アプリケーションログ、セキュリティログ、およびアプリケーションとサービスログの 4 つのカテゴリに分類ができます。 この例では、Windows Server 2008 R2 を使用して 4 つのカテゴリのログの使用方法および分析方法を紹介します。

イベントビューアー を開く

次の手順に従って イベントビューアー を開きます。[ファイル名を指定して実行] ウィンドウを開き、「 eventvwr 」と入力し、[OK]をクリックして [イベントビューアー] を開きます。

次に、[イベントビューアー] で、次の 4 つのカテゴリのログを表示します。

注 :

Microsoft サポート技術情報で、これらのログで確認することができるエラーイベント ID に対するソリューションを検索できます。

- システムログ

  システムログには、Windows システムコンポーネントによって記録されたイベントが含まれます。 たとえば、システムログは、起動時にドライバーや他のシステムコンポーネントをロードする際に発生した障害を記録します。

  システムコンポーネントによって記録されるイベントの種類は、Windows によって事前に決められています。
- アプリケーションログ

  アプリケーションログには、アプリケーションまたはプログラムによって記録されたイベントが含まれます。 たとえば、データベースアプリケーションはファイルエラーをアプリケーションログに記録します。

  記録されるイベントの種類は開発者によって決定されます。
- セキュリティログ

  セキュリティログには、有効および無効なログインの試行などのイベント、ファイルや他のオブジェクトを作成する、開く、または削除するなどのリソース使用量に関するイベントが含まれます。

  管理者はセキュリティログに記録されるイベントの種類を指定します。 たとえば、ログインが監査対象に設定されている場合、ログインの試行がセキュリティログに記録されます。
- アプリケーションとサービスログ

  アプリケーションとサービスログは、新しい種類のイベントログです。 これらのログは、グローバルシステムに影響を与える可能性があるイベントではなく、単一のアプリケーションまたはコンポーネントからのイベントを格納します。

ログのパス変更とバックアップ

ログは既定ではシステムディスクに保存されます。 ログサイズは既定では最大 20 MB で、20 MB を超えると最も早いイベントが上書きされます。 必要に応じて最大ログサイズを変更します。

次の手順に従って、ログのパスを変更し、ログをバックアップします。

1. [イベントビューアー] の左側のナビゲーションウィンドウで、[Windows ログ] をクリックします。
2. [アプリケーション] などのログ名を右クリックし、プロパティをクリックします。
3. [ログのプロパティ] ダイアログボックスでは、次の設定を変更します。
   - ログのパス
   - 最大ログサイズ
   - イベントログサイズが最大値に達したときに実行される操作

1.7 セキュリティが強化された Windows ファイアウォールの概要とベストプラクティス

ここでは、WFAS (セキュリティが強化された Windows ファイアウォール)、そのアプリケーションシナリオ、および共通操作について説明します。

概要

階層型セキュリティモデルの重要な部分として、WFAS は Microsoft から Windows NT6.0 以降に発売されました。 WFAS は、現在の接続状態に基づいて双方向のフィルタリングを提供することで、ローカルコンピュータに出入りする不正なトラフィックをブロックします。 WFAS はまた、NLA (Network Location Awareness) を使用し、現在の接続状態に基づいて、対応するファイアウォールプロファイルをコンピューターに適用します。 Windows ファイアウォールおよび IPsec (インターネットプロトコルセキュリティ) のセキュリティルールは、MMC (Microsoft 管理コンソール) スナップインで構成されており、WFAS もネットワークの分離ポリシーの重要な部分です。

アプリケーションシナリオ

サーバーが攻撃され、パスワードが破られたと報告する O&M の担当者がますます増えています。これはほとんどの場合、"侵入者"に開かれた"バックドア"が原因です。 侵入者は、コンピュータの開いているポートをスキャンし、脆弱なポート、たとえば Windows のリモートポート 3389 や Linux のリモートポート 22 を通過します。 問題がどこにあるかがわかったので、効果的な対策が講じられます。 具体的には、既定のリモートポートを変更してリモートアクセスを

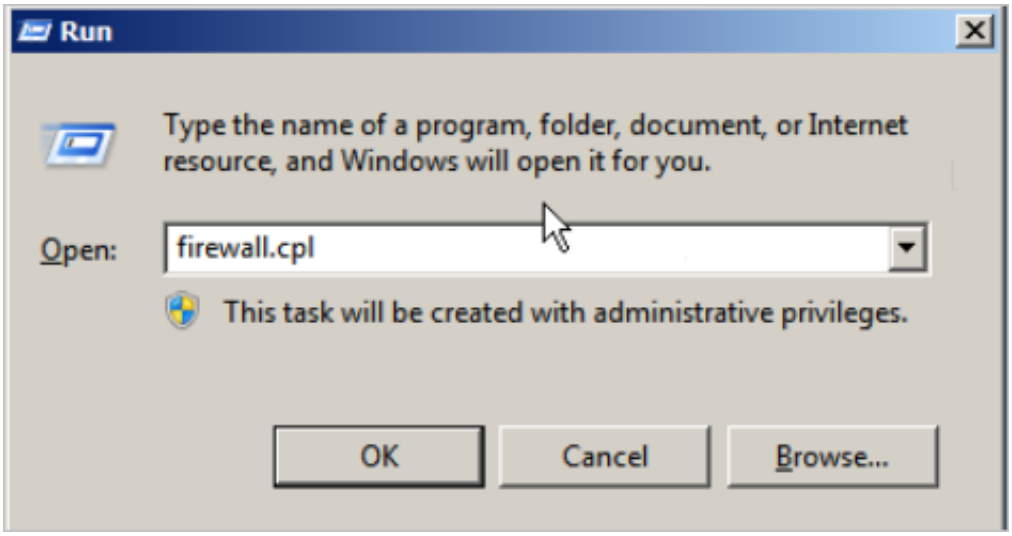
制限することで、これらの"バックドア"を閉じます。 それでは、どのようにリモートアクセスを制限したらよいでしょうか。 たとえば、ECS インスタンス (Windows Server 2008 R2) を使用してリモートデスクトップ接続を制限する方法をここで説明します。

手順

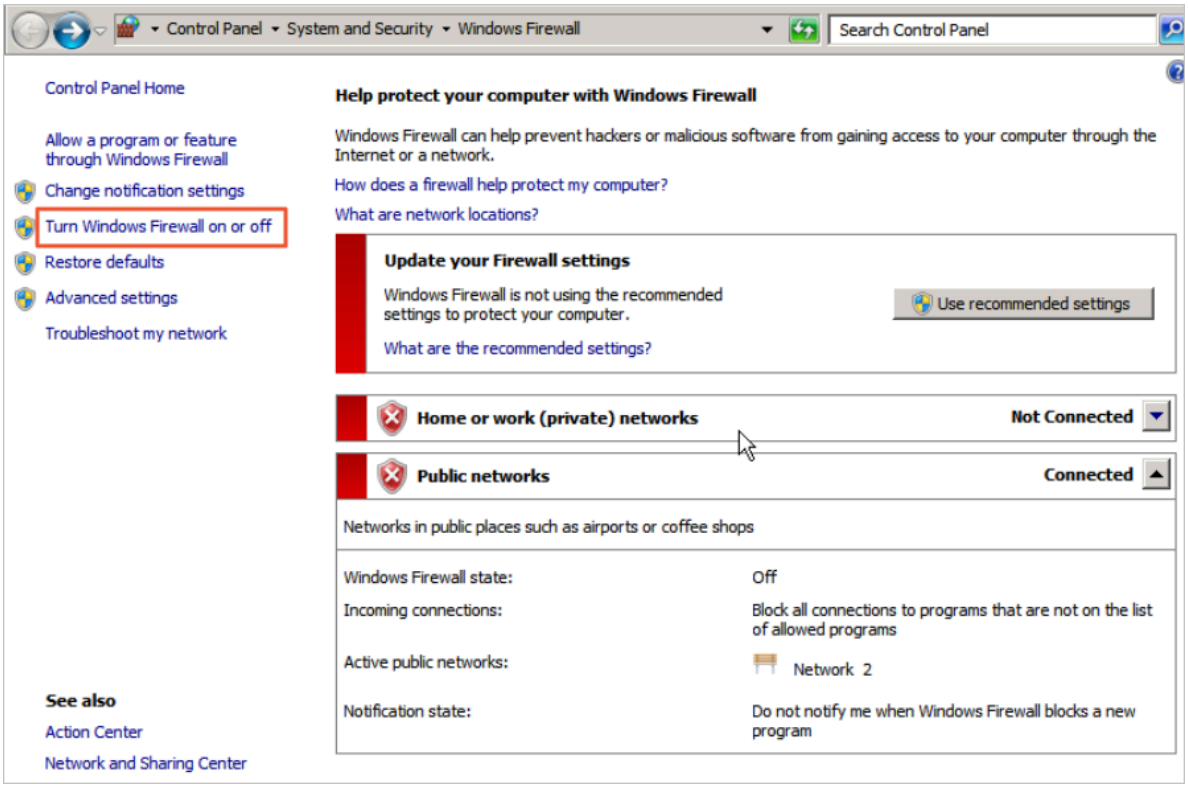
1. Windows ファイアウォールの状態の表示

   ECS インスタンスの Windows ファイアウォールは、既定では無効になっています。
   Win キーを押しながら R キーを押して [ファイル名を指定して実行] ウィンドウを開き、

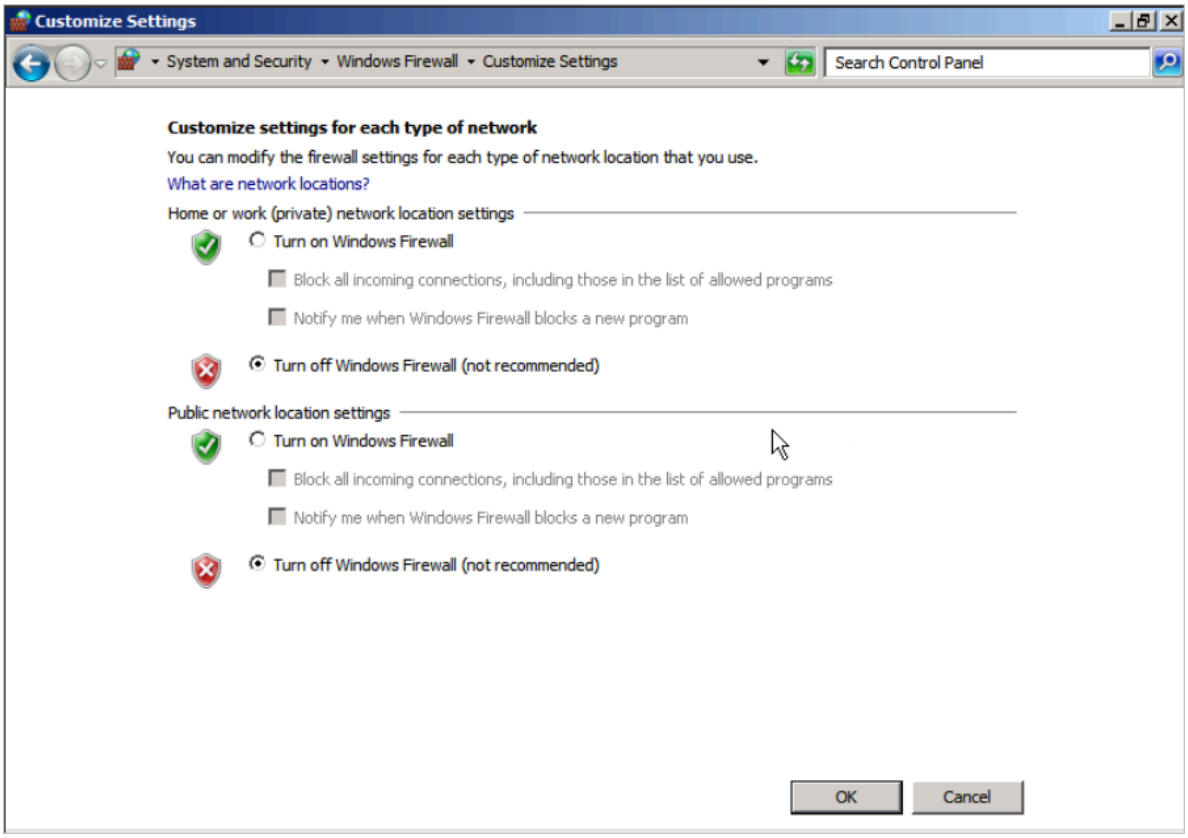
「*firewall.cpl* 」と入力し、Enter キーを押すと、以下に示すように Windows ファイアウォールコンソールが開きます。



Windows ファイアウォールを有効または無効にします。

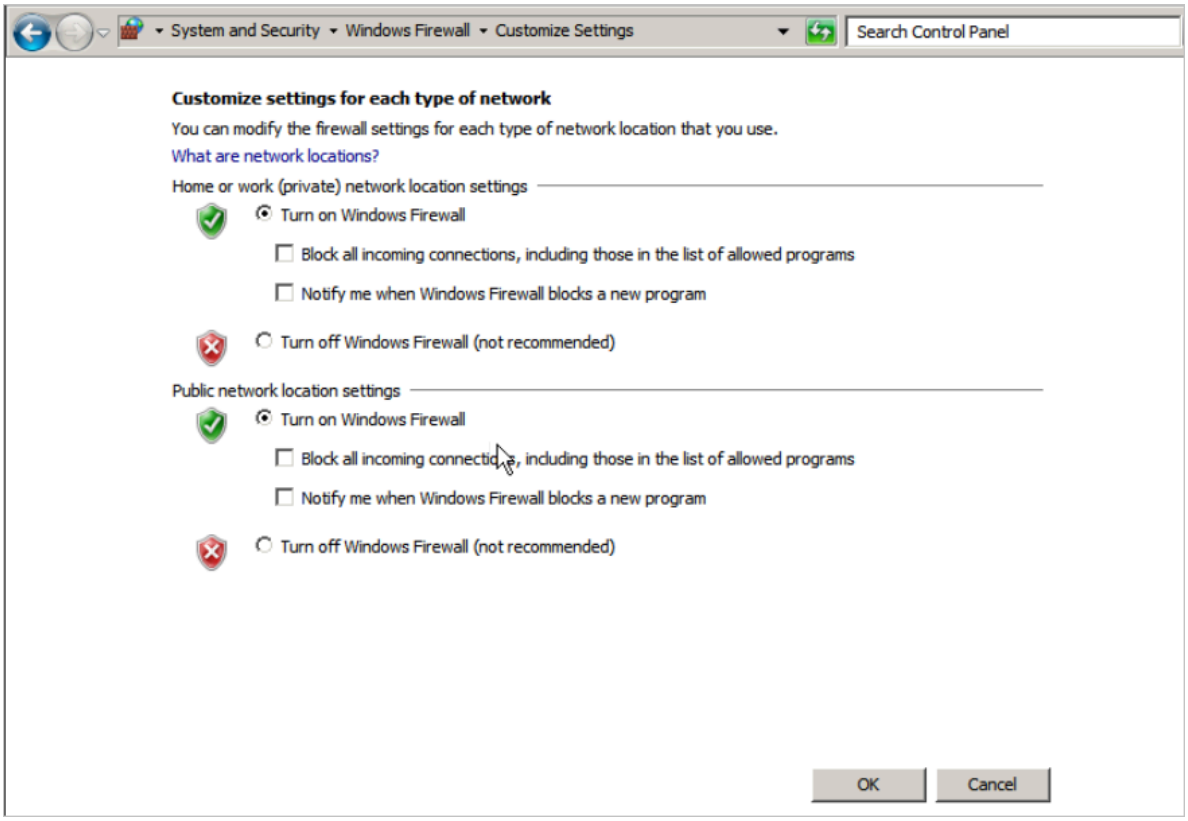


以下に示すように、Windows ファイアウォールは既定では無効になっています。


Customize Settings
System and Security
Windows Firewall
Customize Settings
Search Control Panel
Customize settings for each type of network
You can modify the firewall settings for each type of network location that you use.
What are network locations?
Home or work (private) network location settings
Turn on Windows Firewall
Block all incoming connections, including those in the list of allowed programs
Notify me when Windows Firewall blocks a new program
Turn off Windows Firewall (not recommended)
Public network location settings
Turn on Windows Firewall
Block all incoming connections, including those in the list of allowed programs
Notify me when Windows Firewall blocks a new program
Turn off Windows Firewall (not recommended)
OK
Cancel

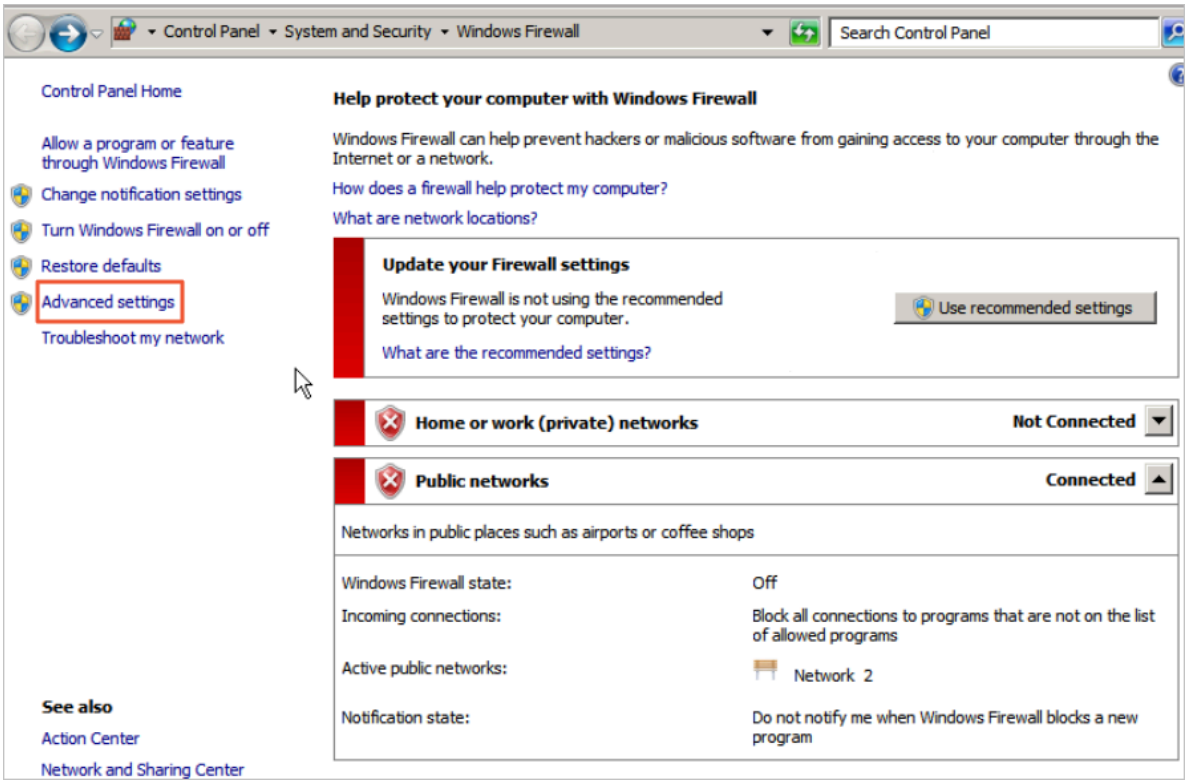
2. Windows ファイアウォールの有効化

以下に示すように、前の手順により Windows ファイアウォールを有効にします。

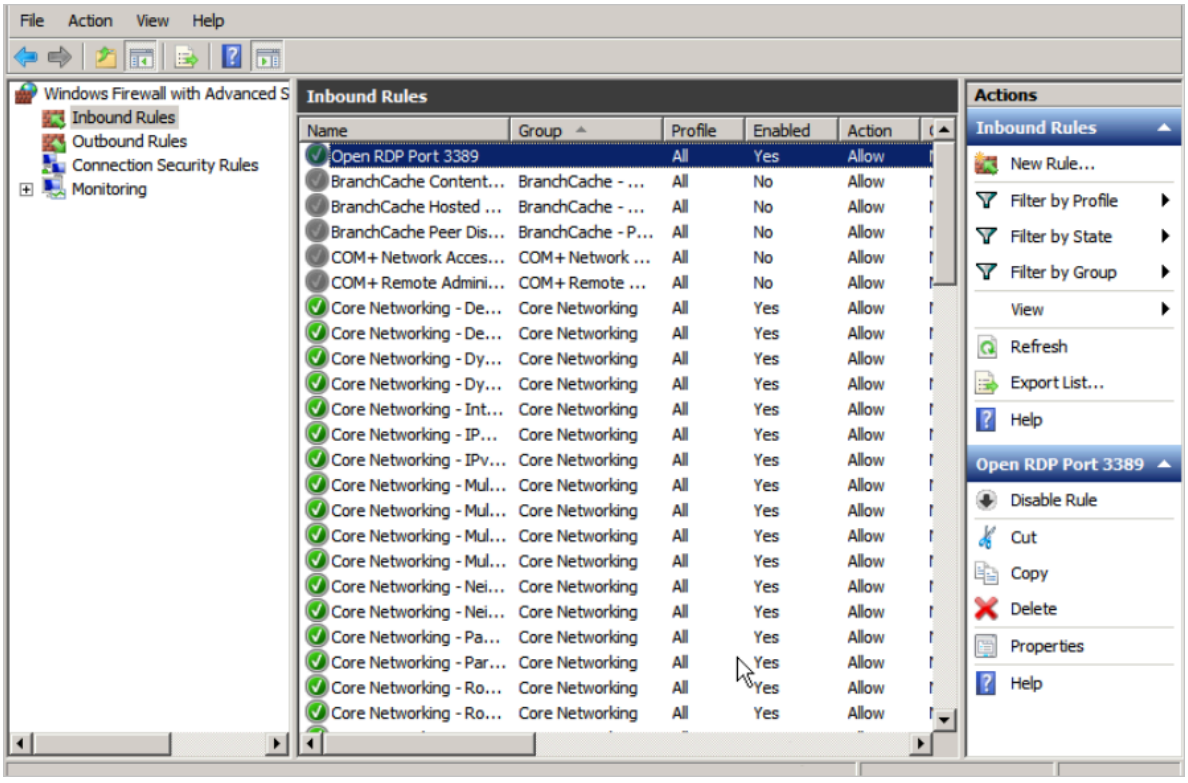


Windows ファイアウォールを有効にする前に、インバウンドルールでリモートポートが開いていることを確認します。開いていないと、自分自身でもリモート接続を確立できません。

ただし、WFAS は既定ではインバウンドルールで RDP ポート 3389 を開きます。 [詳細設定] をクリックします。

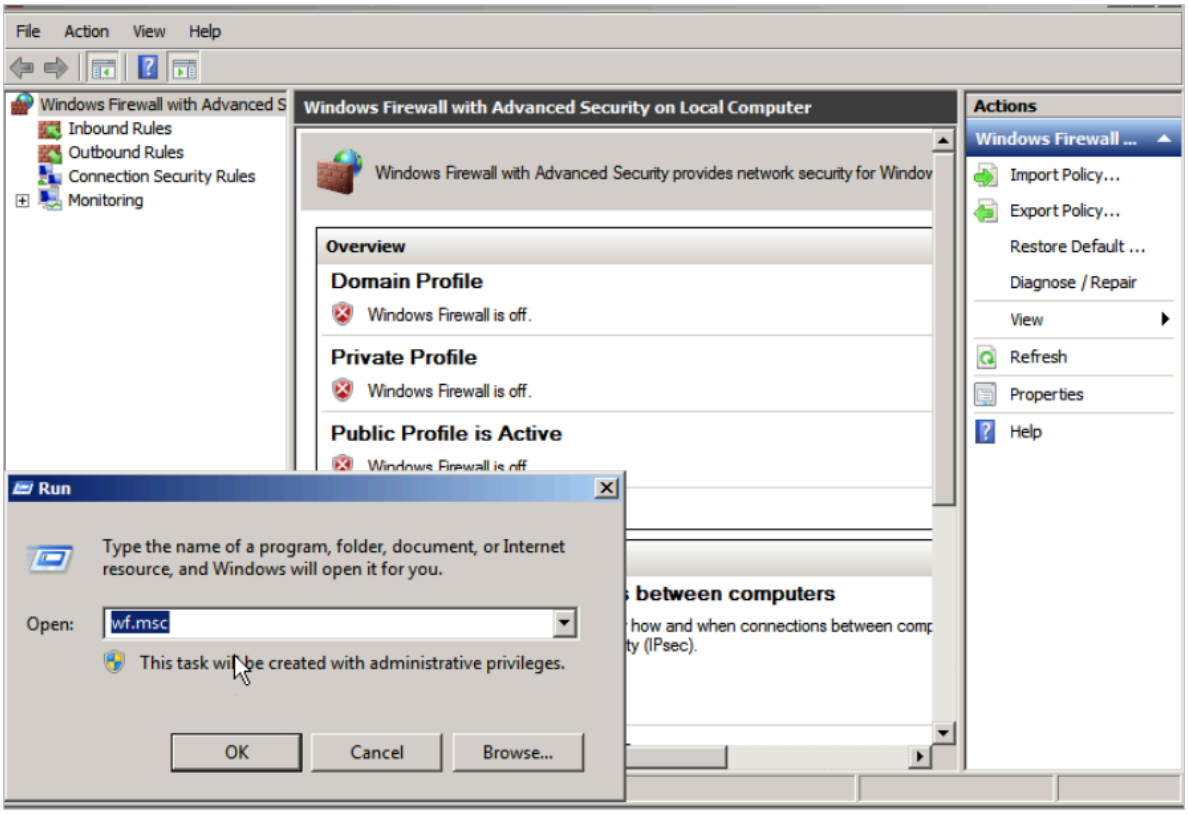


[インバウンドルール] をクリックします。 既定では、"Open RDP Port 3389"のルールが有効になっていることがわかります。

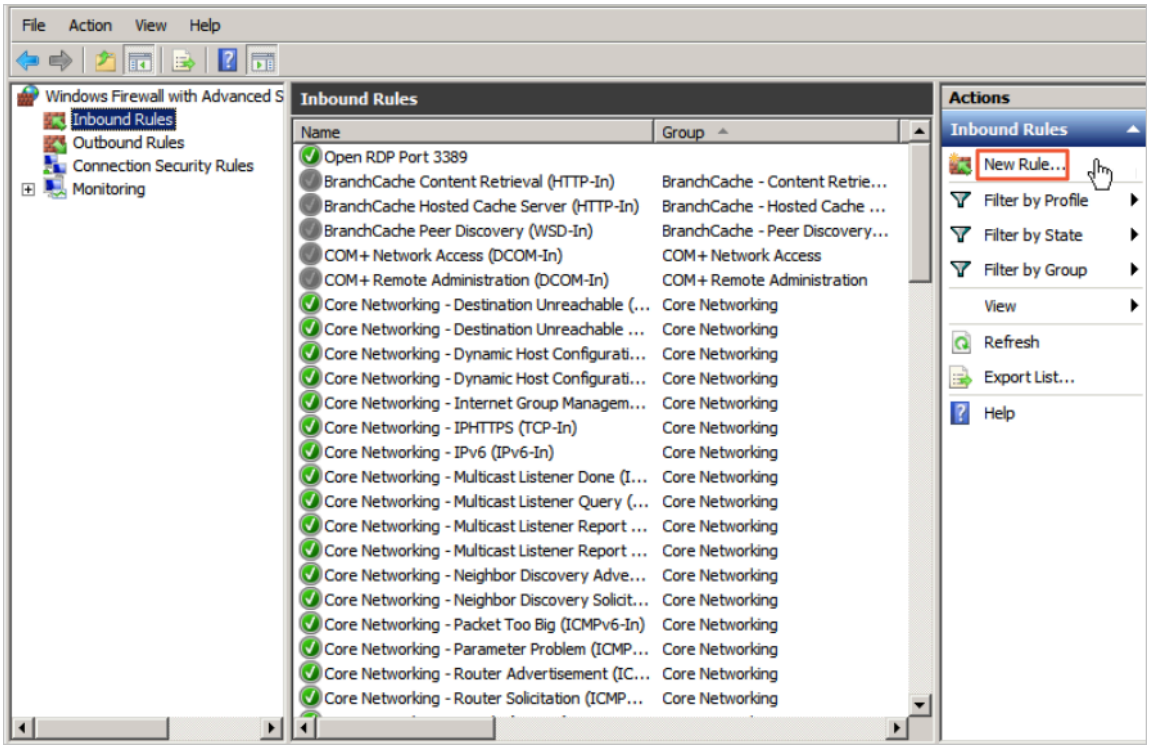


3. WFAS の設定

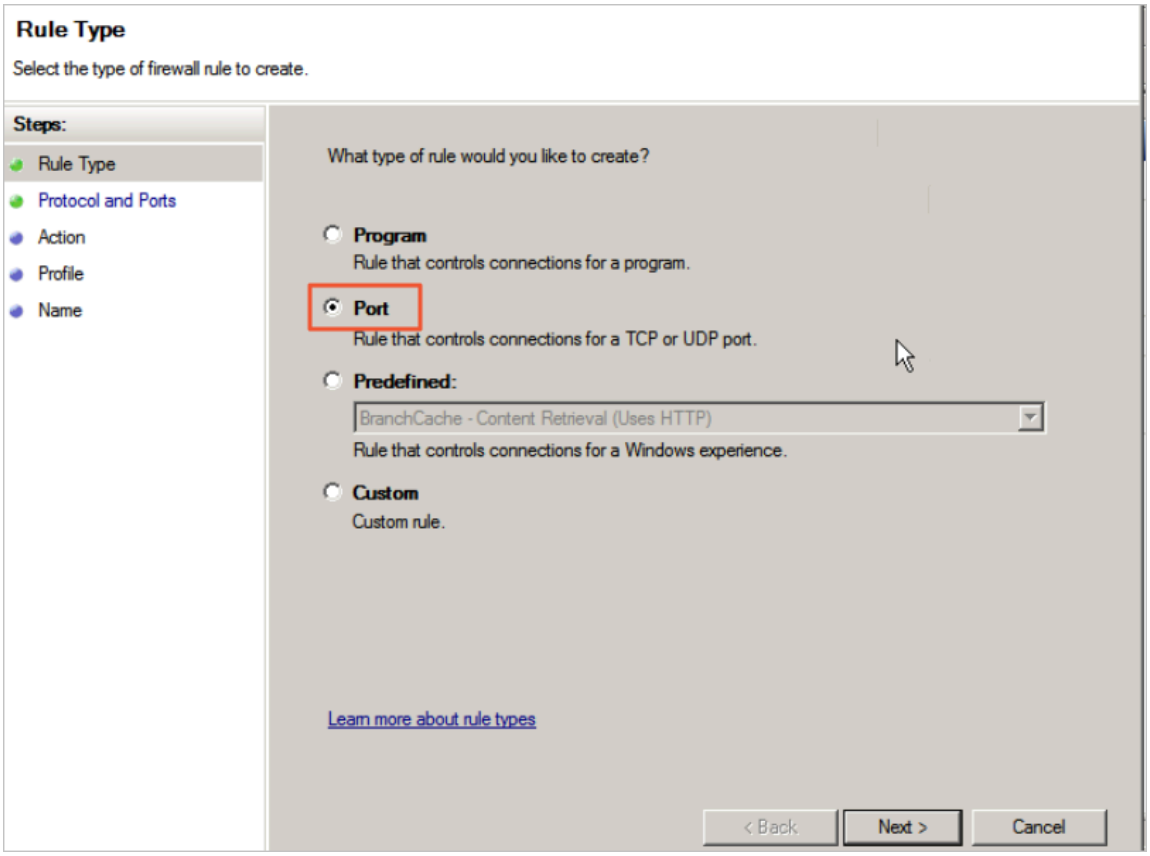
Win キーを押しながら R キーを押して [ファイル名を指定して実行] ウィンドウを開き、「*wf.msc*」と入力し、Enter キーを押すと、以下に示すように WFAS ウィンドウが開きます。



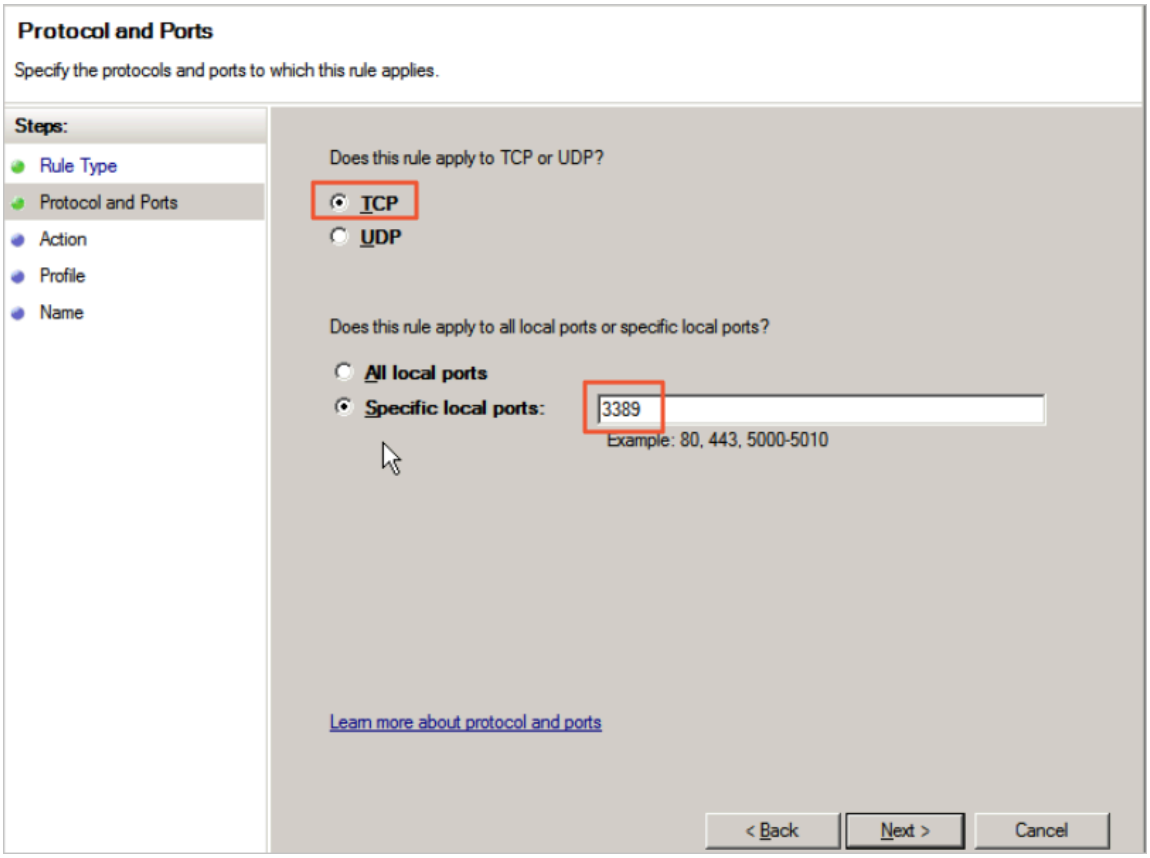
a. インバウンドルールの手動作成



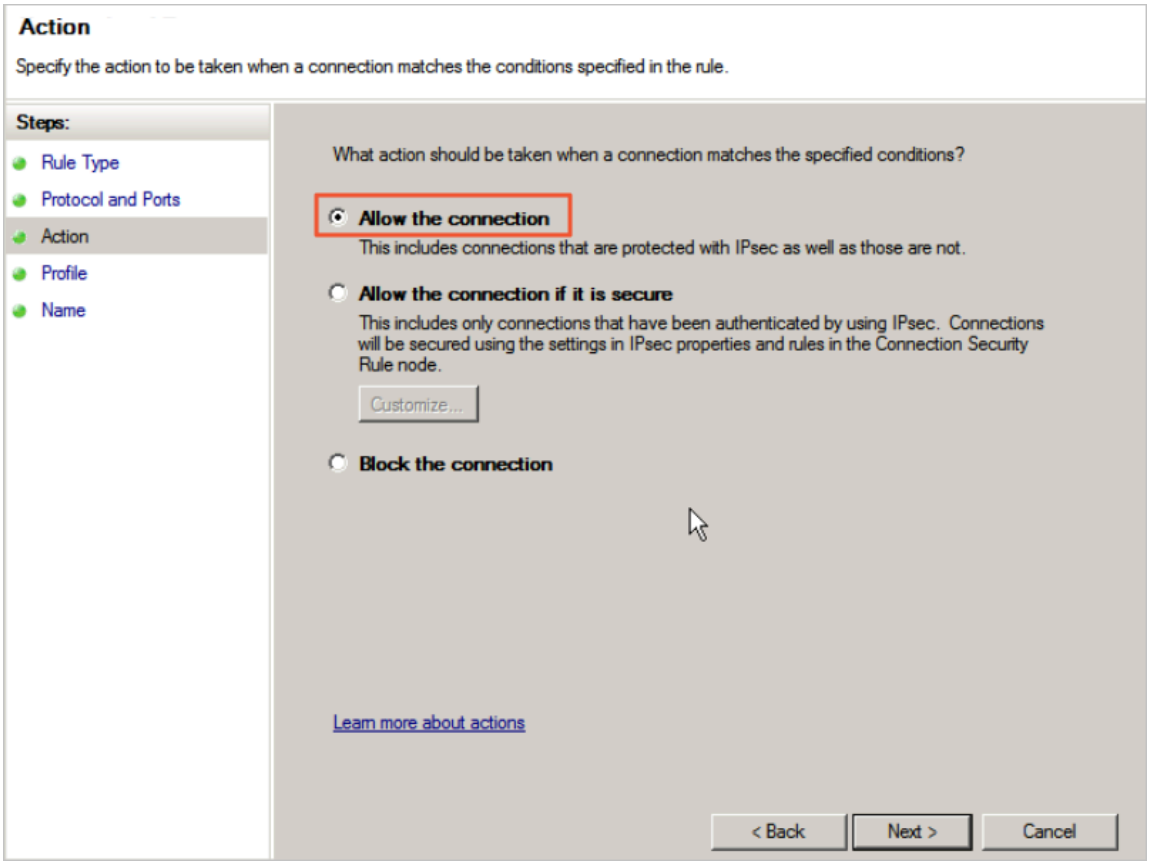
[新規のインバウンドルールウィザード] ウィンドウで、[ポート] をオンにして [次へ] をクリックします。



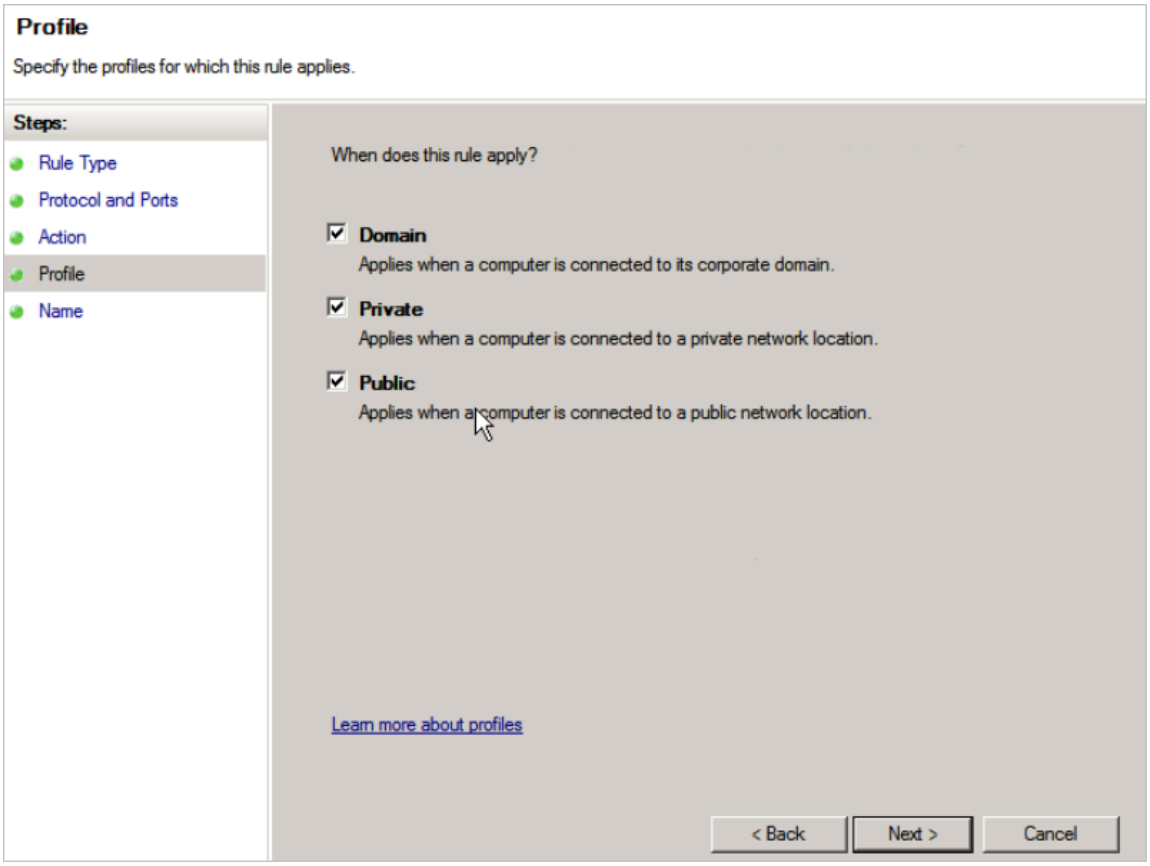
[TCP] をオンにして [特定のローカルポート] を"3389"に設定します。



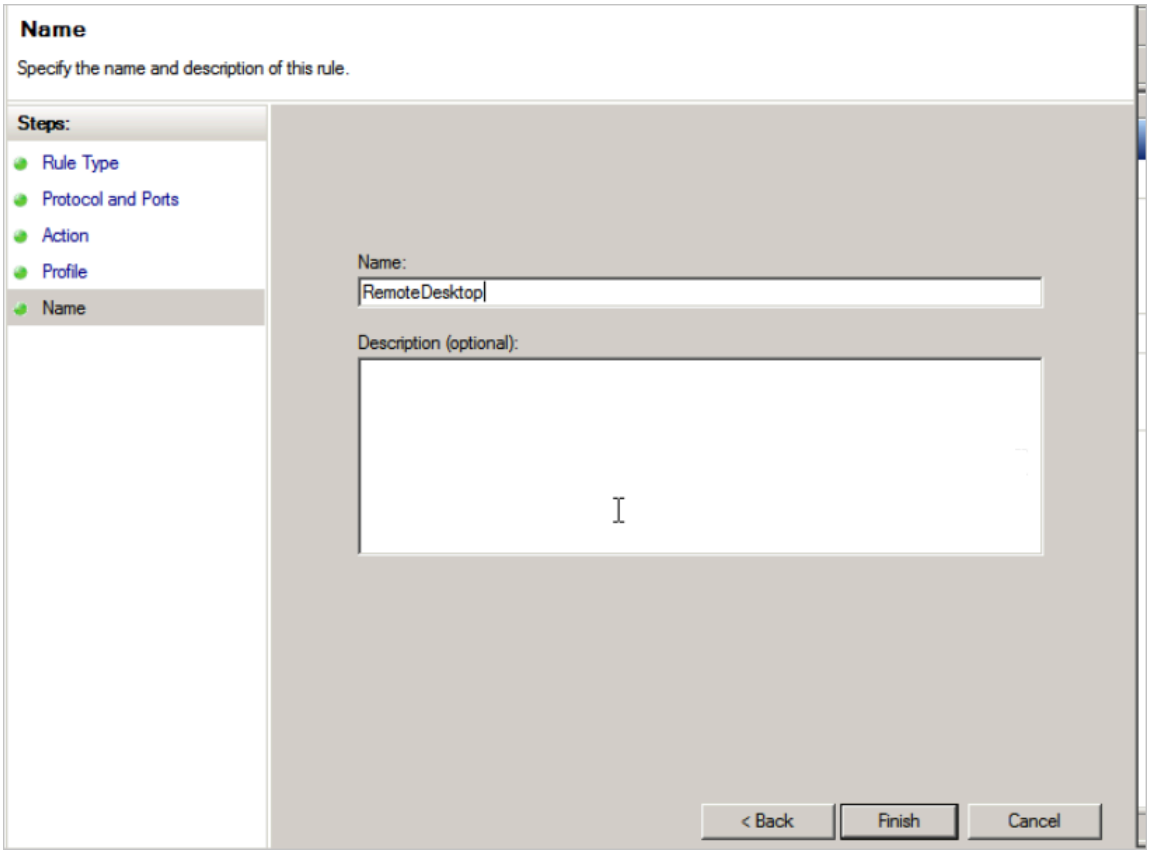
[次へ] をクリックして [接続を許可する] をオンにします。



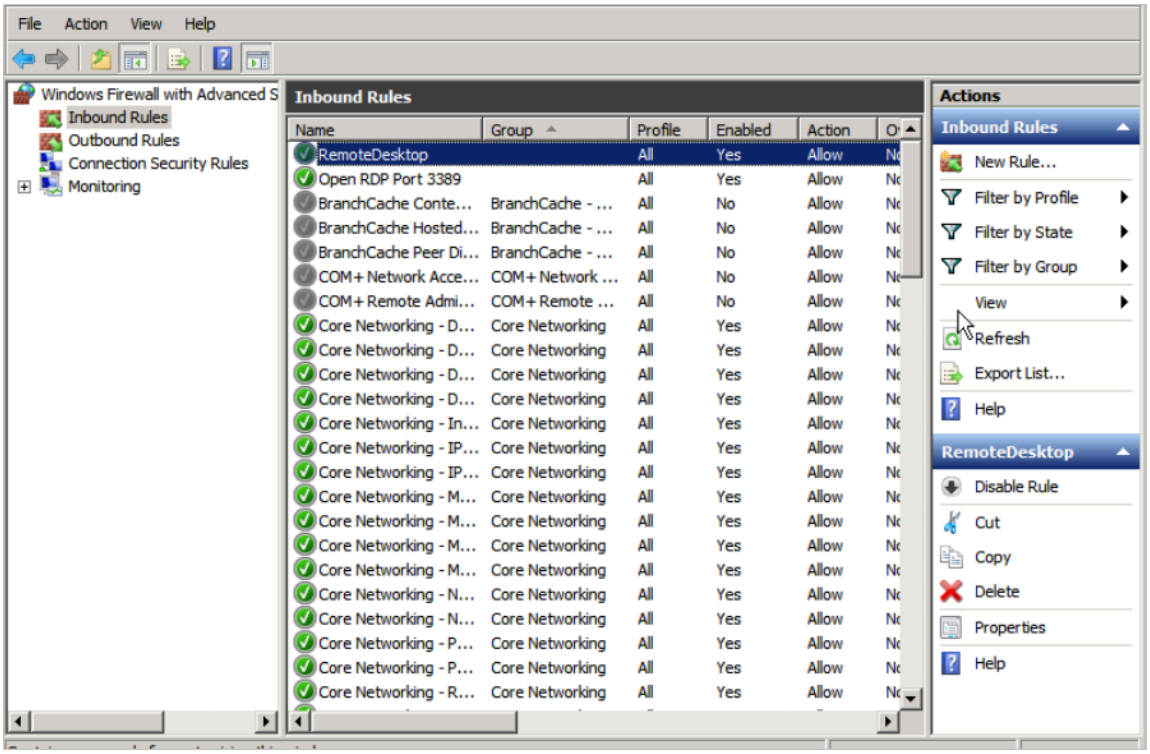
[次へ] をクリックして既定の設定をそのまま使用します。



[次へ] をクリックしてルールの名前 ("RemoteDesktop"など) を入力し、[終了] をクリックします。



新しいルールが [インバウンドルール] リストに表示されます。

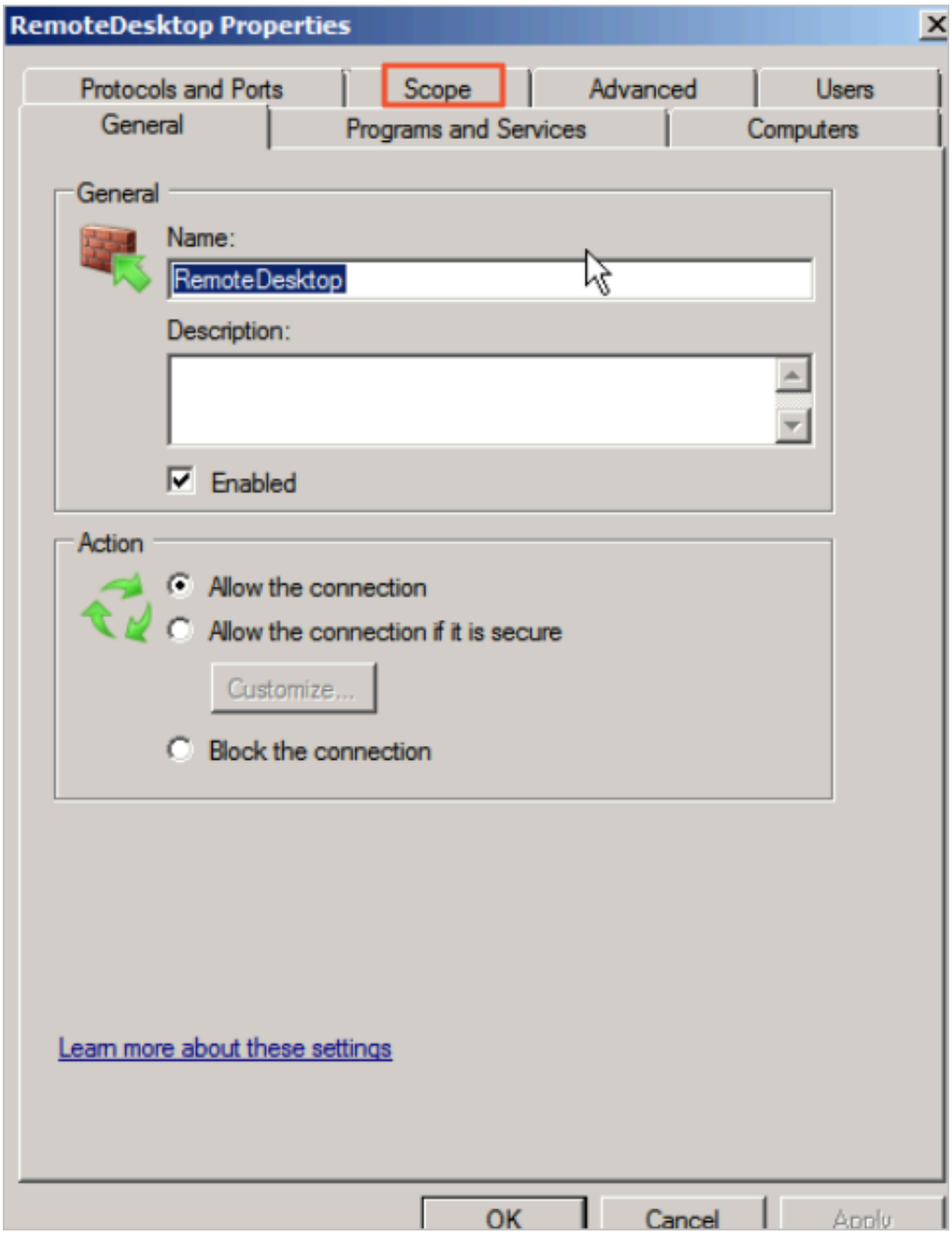


上記の手順で、リモートポートが WFAS に追加されますが、アクセス制限はまだ実装されていません。 これからそれを実装します。

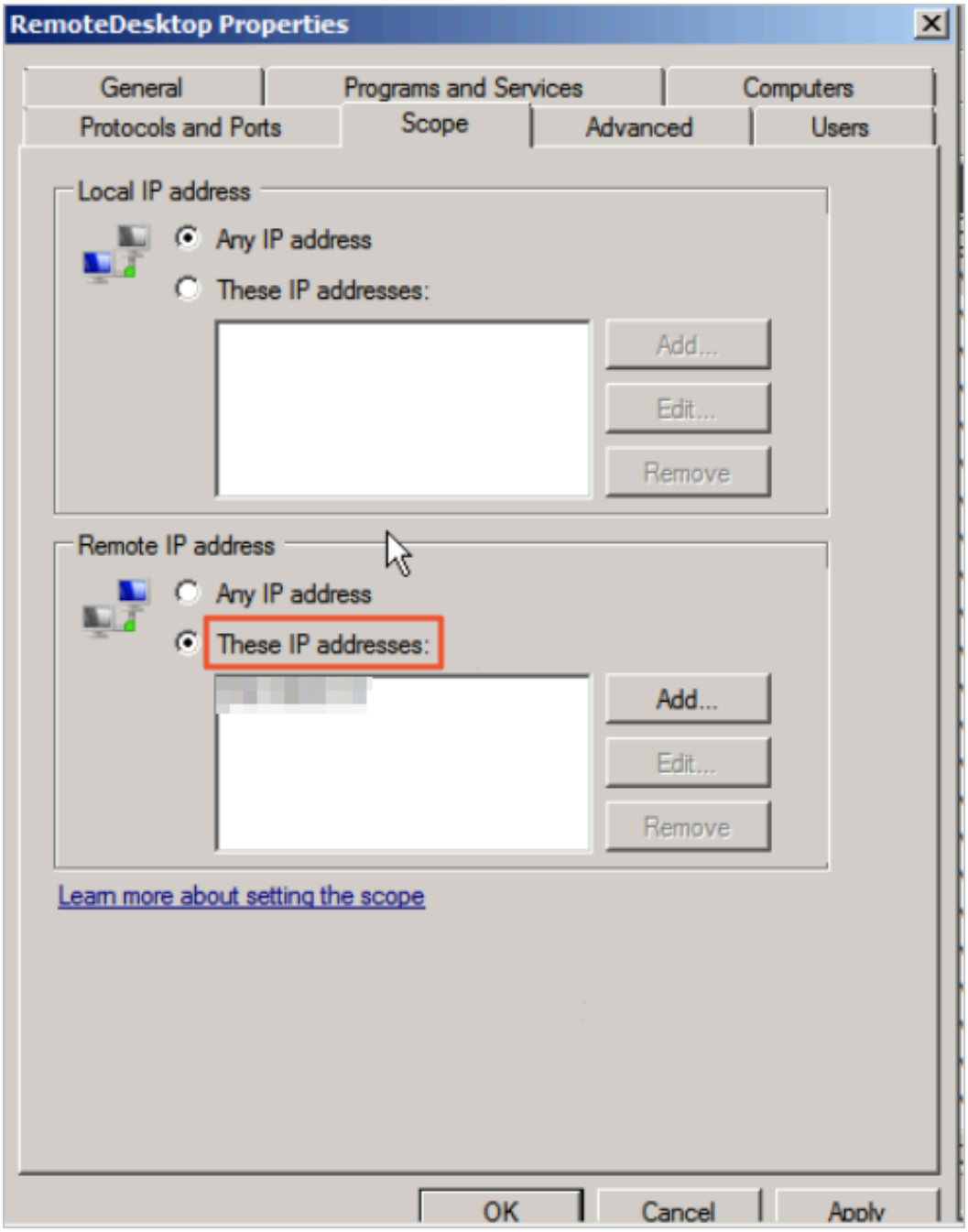
b. IP アドレスのスコープの設定

作成した[インバウンドルール]を右クリックして、コンテキストメニューの [プロパティ] をクリックします。 表示されたダイアログボックスで、[スコープ] タブをクリックします。 次に、この ECS インスタンスにアクセスできるリモート IP アドレスを追加します。

ここで IP アドレス設定を有効にすると、他の IP アドレスはこの ECS インスタンスにアクセスできなくなります。

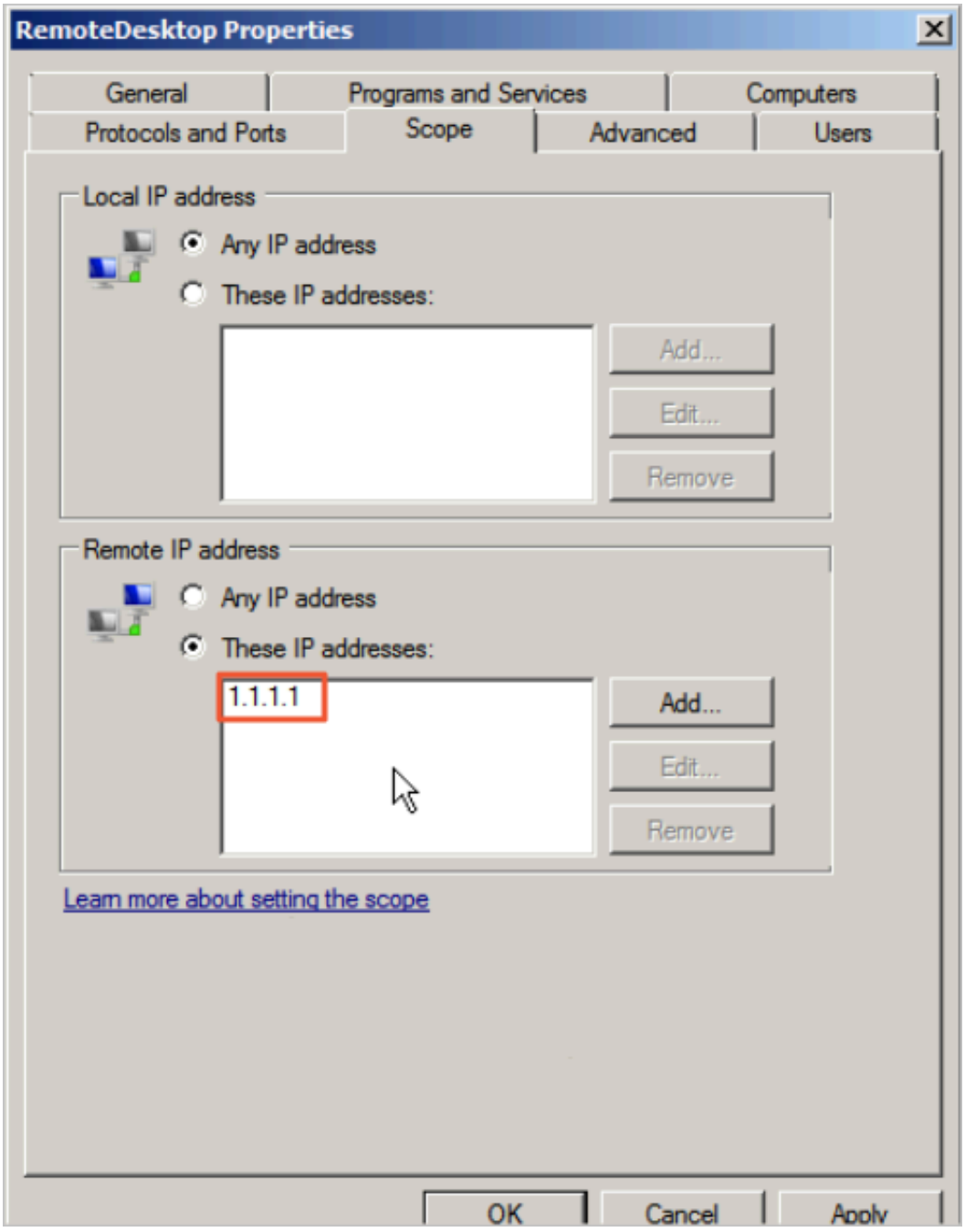


リモート IP アドレスを追加します。

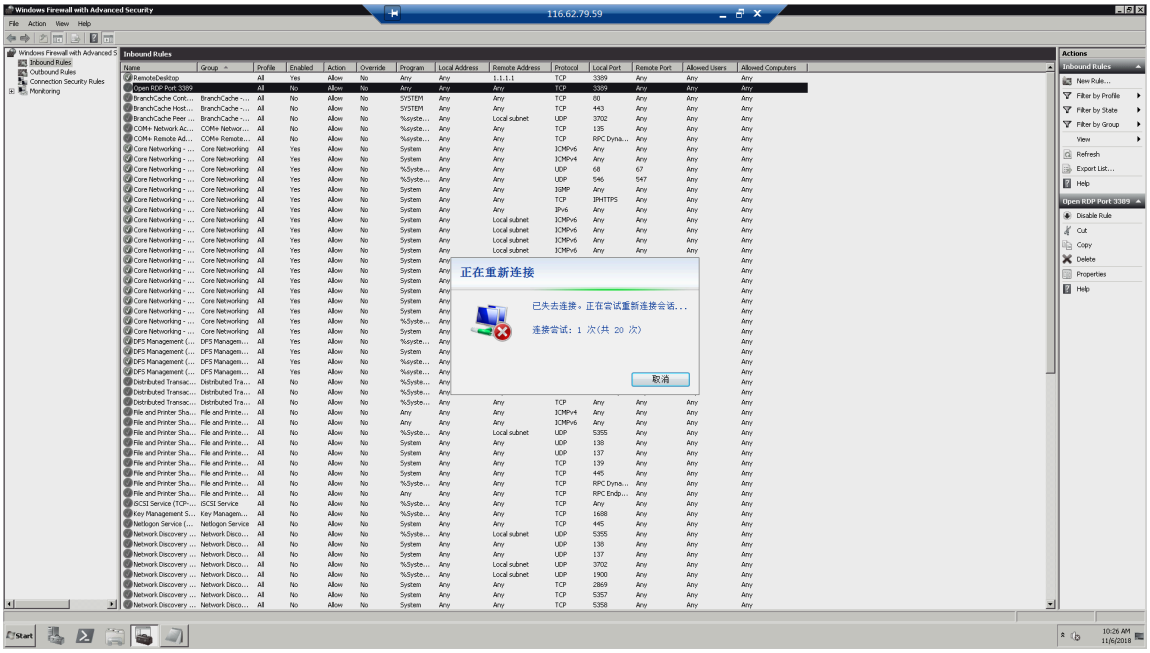


c. IP アドレスのスコープの検証

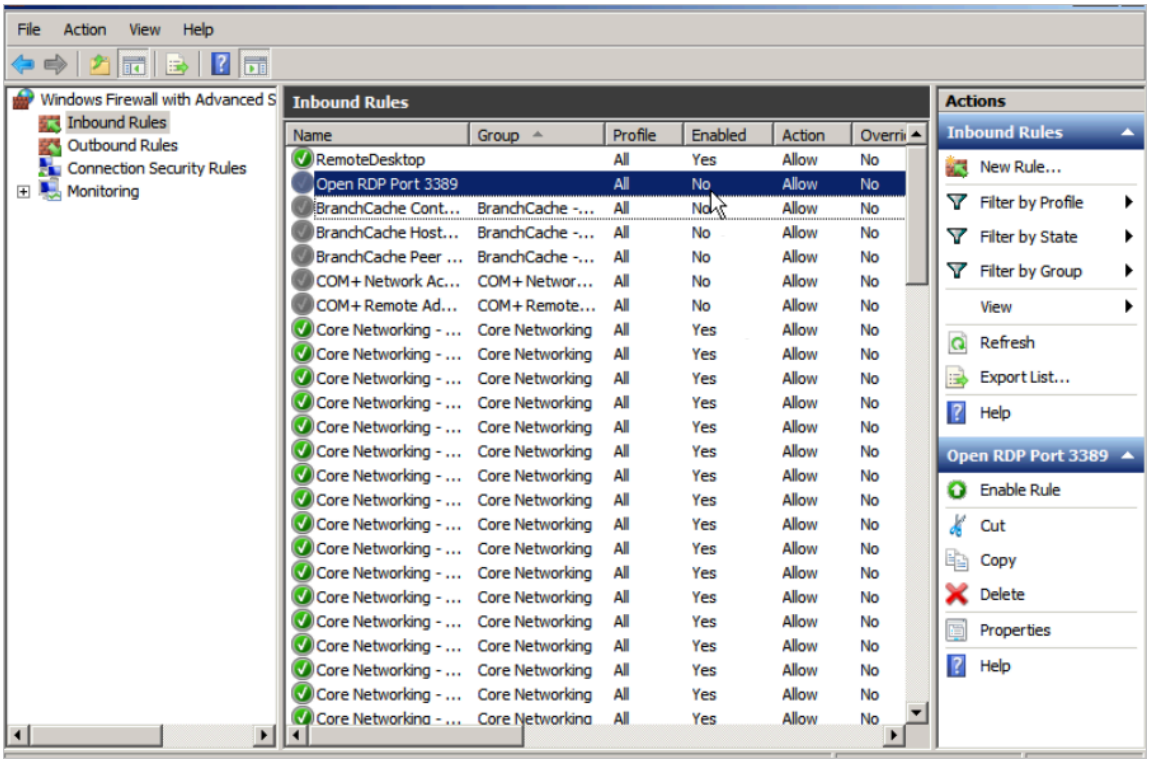
リモート IP アドレスボックスに IP アドレスを任意に追加して、リモート接続に何が起きるのかを見てみます。



リモート接続がダウンしています。



それでもリモート接続が確立されている場合は、"Open RDP Port 3389"のルールを無効にするだけです。

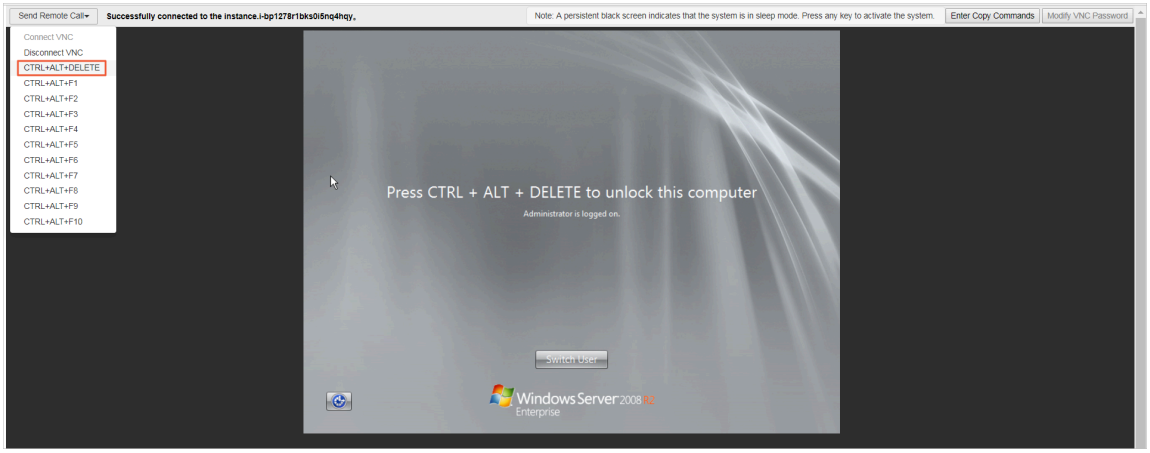


リモート接続がダウンしている場合は、IP アドレスのスコープが有効になっていることを意味します。 ただし、現在 ECS インスタンスに接続することはできません。 どうすべきでしょうか。 ここで ECS コンソールを見てみます。 ECS コンソールにログインし、[スコープ] タブで既に設定したリモート IP アドレスを自身のアドレスに置き換えます (もし

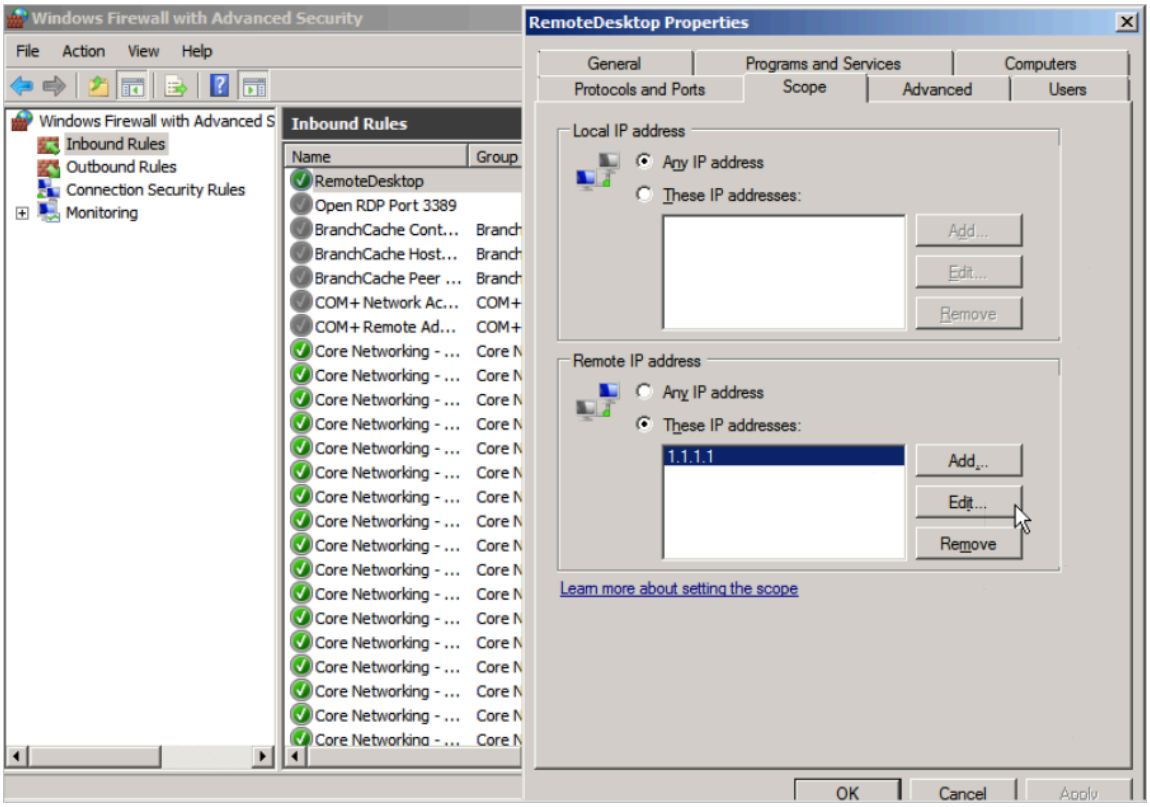
仕事環境が Alibaba Cloud に接続されていなければ、インターネットアドレスを入力します)。 これで ECS インスタンスに再度接続します。

ECS コンソールに入り、対応するインスタンスを見つけて接続します。

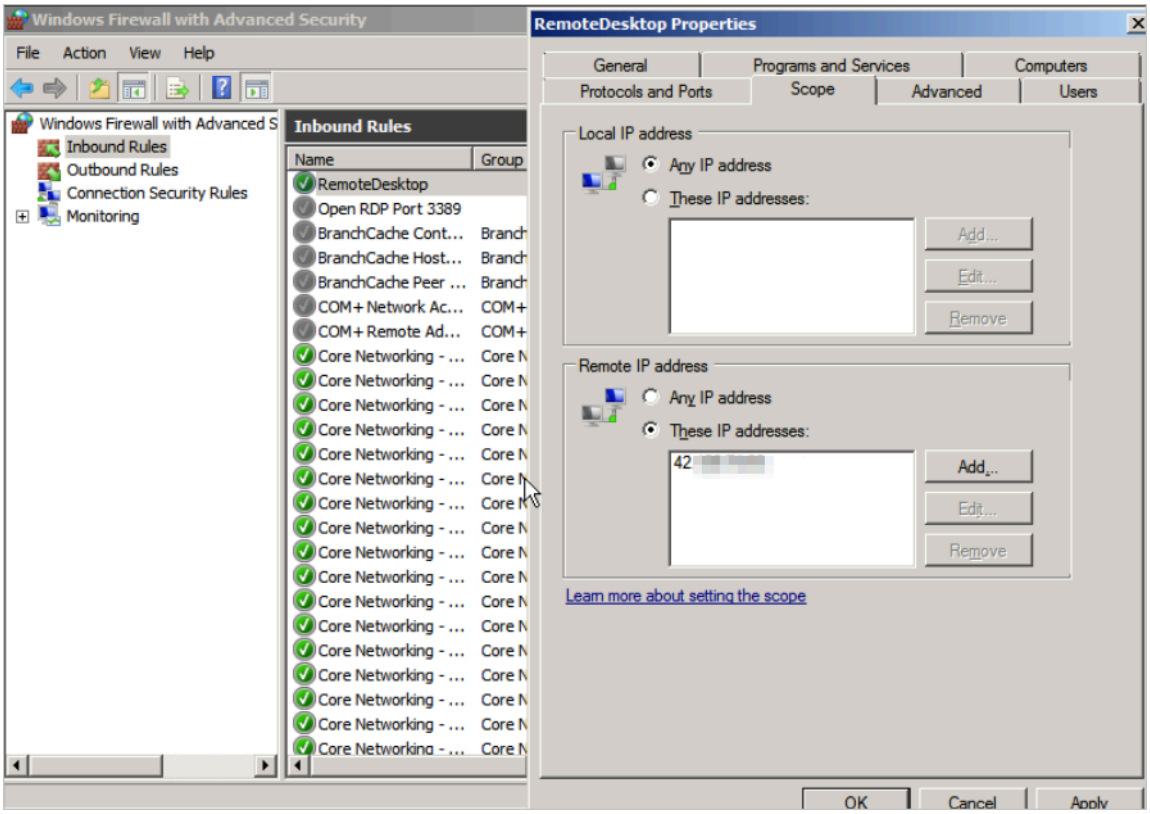
ECS インスタンスにログインします。



同様に、RemoteDesktop のルールの [スコープ] タブでリモート IP アドレスを変更します。 具体的には、"1.1.1.1"を自身の IP アドレスに置き換えます。



これで、IP アドレスを追加した後、正常に ECS インスタンスに接続できます。 インターネットアドレスがわからない場合は、ここをクリックして表示します。



上記の手順は、WFAS を介して、ECS インスタンスに対するリモートアクセス制限を実装します。 他のサービスやポートについても同様に制限を実装可能です。たとえば、あまり使用されないポート 135、137、138、および 445 を無効にしたり、FTP および関連サービスへのアクセスを制限したりします。さらには、ECS インスタンスの保護を最大化します。

コマンドライン操作

1. ファイアウォール設定をファイルにエクスポートします。

```
netsh advfirewal l export c :\ adv . pol
```

2. ファイアウォール設定ファイルをシステムにインポートします。

```
netsh advfirewal l import c :\ adv . pol
```

3. 既定のファイアウォール設定を復元します。

```
Netsh advfirewal l reset
```

4. ファイアウォールを無効にします。

```
netsh advfirewal l set allprofile s state off
```

5. ファイアウォールを有効にします。

```
netsh advfirewal l set allprofile s state on
```

6. すべての設定ファイルで、既定では、インバウンドトラフィックをブロックし、アウトバウンドトラフィックを許可するように設定します。

```
netsh advfirewal l set allprofile s firewallpo licy
blockinbou nd , allowoutbo und
```

7. “ftp” という名前の規則を削除します。

```
netsh advfirewal l firewall delete rule name = ftp
```

8. ローカルポート 80 のすべてのインバウンドルールを削除します。

```
netsh advfirewal l firewall delete rule name = all
protocol = tcp localport = 80
```

9. RemoteDesktop のルールを追加して、ポート 3389 を許可します。

```
netsh advfirewal l firewall add rule name = RemoteDesk
top ( TCP - In - 3389 ) protocol = TCP dir = in localport =
3389 action = allow
```

リファレンス

Windows 2008 または 2012 のファイアウォールを使用して、ポート、IP アドレス、アプリケーションのアクセスを制限する方法

『Alibaba Cloud Marketplace』で、より多くのオープンソースソフトウェアが入手可能です。

1.8 セキュリティグループ内のインスタンスの分離

セキュリティグループは、SPI (Stateful Packet Inspection) とパケットフィルタリングを提供する仮想ファイアウォールです。 これには、同じセキュリティ要件と相互信頼を持つ同じリージョン内のインスタンスが含まれています。 Alibaba Cloud は、セキュリティグループ内のインスタンスを分離できるようにさまざまなアクセス制御ポリシーを提供します。

グループ内の分離ルール

- セキュリティグループ内のネットワーク分離は、インスタンス間ではなくネットワークインターフェイス間で実装されます。 複数の ENI (Elastic Network Interface) がインスタンスにバインドされている場合は、各 ENI に対して分離ルールを設定する必要があります。
- セキュリティグループ内のインスタンス は、既定では互いにアクセスできますが、分離ルールによって変更されることはありません。

  グループ内分離ルールは、ユーザー定義のアクセス制御ポリシーであり、既定のセキュリティグループおよび新しいセキュリティグループには無効です。 セキュリティグループの既定のアクセス制御ポリシーは次のとおりです。同じセキュリティグループ内のインスタンスはイントラネットを介して互いにアクセスできますが、異なるセキュリティグループ内のインスタンスはアクセスできません。
- グループ内分離ルールの優先順位が最も低くなります。

  セキュリティグループ内のインスタンスを分離するには、分離ルール以外に相互通信ルールが適用されないようにします。 次の場合、グループ内分離ルールが設定されていても、インスタンスは相互にアクセスできます。

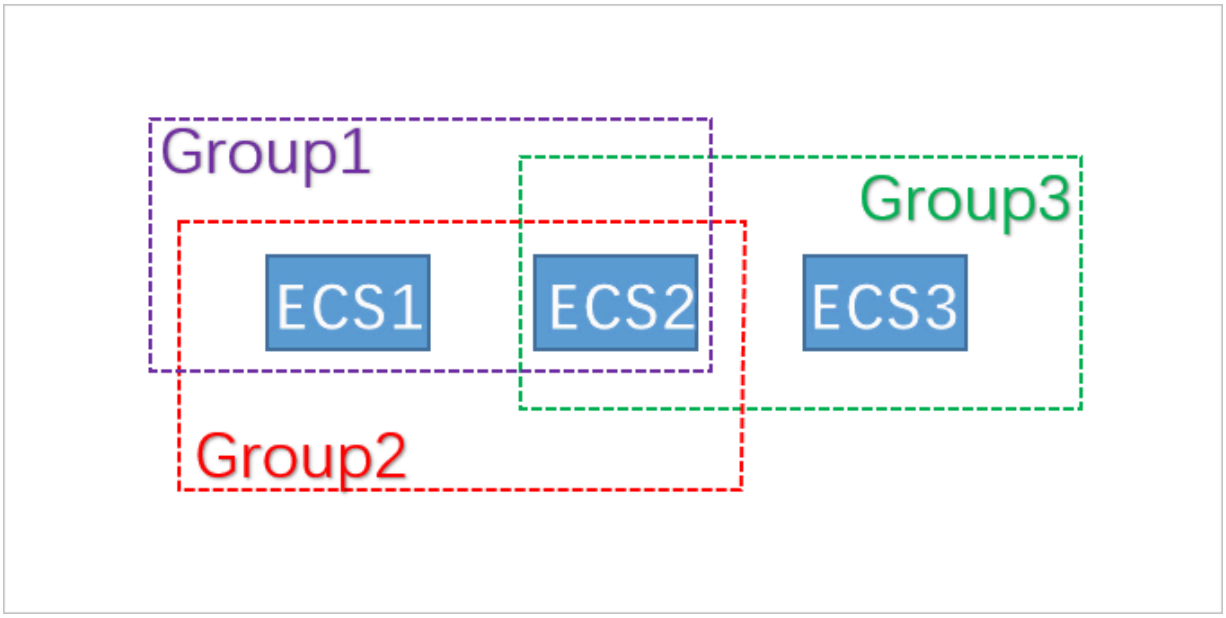
  - グループ内分離ルールはセキュリティグループに設定され、インスタンス間のグループ内通信を許可する アクセス制御リスト (ACL) が同時に設定されます。
  - グループ内分離ルールはセキュリティグループに設定され、グループ内相互通信は同時に設定されます。
- グループ内分離ルールは、現在のセキュリティグループ内のインスタンスにのみ適用されます。

アクセス制御ポリシーの変更

ModifySecurityGroupPolicy インターフェイスを使用して、セキュリティグループ内のアクセス制御ポリシーを変更します。

ケース分析

次の図は、3 つのインスタンスとそれらのセキュリティグループの関係を示しています。



この例では、Group1、Group2、および Group3 は 3 つの異なるセキュリティグループです。 ECS1、ECS2、および ECS3 は 3 つの異なる ECS インスタンスです。 ECS1 と ECS2 は、Group1 と Group2 に属します。 ECS2 と ECS3 は、Group3 に属します。

3 つのセキュリティグループのグループ内相互通信ポリシーは次のとおりです。

セキュリティグループ	グループ内相互通信ポリシー	含まれるインスタンス
Group1	分離	ECS1 と ECS2
Group2	相互接続	ECS1 と ECS2
Group3	相互接続	ECS2 と ECS3

インスタンス間の通信状態は次のとおりです。

インスタンス	相互接続か分離か	理由
ECS1 と ECS2	相互接続	ECS1 と ECS2 は Group1 とGroup2 の両方に属します。 Group1 のポリシーは "分離" で、Group2 のポリシーは "相互接続" です。 グループ内分離の優先順位が最も低いため、ECS1 と ECS2 は相互接続されています。
ECS2 と ECS3	相互接続	ECS2 と ECS3 の両方が Group3 に属します。 Group3 のポリシーは ” 相互接続” であるため、ECS2 と ECS3 は相互接続されています。

インスタンス	相互接続か分離か	理由
ECS1 と ECS3	分離	ECS1 と ECS3 は異なるセキュリティグループに属します。 異なるセキュリティグループ内のインスタンスは、既定では相互接続されていません。 2 つのセキュリティグループ内のインスタンス間のアクセスを許可するために、セキュリティグループルールを通じてセキュリティグループに権限付与します。

1.9 セキュリティグループの 5 つのルール

セキュリティグループは、1 つ以上の ECS インスタンスにネットワークアクセス制御を設定するために使用されます。 セキュリティの分離の重要な手段として、セキュリティグループを使用すると、クラウド上のセキュリティ領域を分割できます。 セキュリティグループの 5 つのルールを使用すると、送信元 IP アドレス、送信元ポート、宛先 IP アドレス、宛先ポート、トランスポート層プロトコルという 5 つのパラメーターを正確に制御できます。

背景情報

以前から、セキュリティグループのルールには、以下の特徴があります。

- Ingress ルールは、送信元 IP アドレス、宛先ポート、およびトランスポート層プロトコルの設定のみをサポートします。
- Egress ルールは、宛先 IP アドレス、宛先ポート、およびトランスポート層プロトコルの設定のみをサポートします。

ほとんどの場合、これらの種類のセキュリティグループのルールは設定プロセスを簡略化しますが、以下の欠点があります。

- Ingress ルールの送信元ポート範囲が制限されません。 つまり、すべての送信元ポートがデフォルトで許可されます。
- Ingress ルールの宛先 IP アドレスが制限されません。 つまり、セキュリティグループ内のすべての IP アドレスがデフォルトで許可されます。
- Egress ルールの送信元ポート範囲が制限されません。 つまり、すべての送信元ポートがデフォルトで許可されます。
- Egress ルールの送信元 IP アドレスが制限されません。 つまり、セキュリティグループ内のすべての IP アドレスがデフォルトで許可されます。

5 つのルールの定義

5 つのルールには、送信元 IP アドレス、送信元ポート、宛先 IP アドレス、宛先ポート、トランスポート層プロトコルというパラメーターが含まれます。

5 つのルールは、既存のセキュリティグループのルールと完全に互換性を保ちながら、前述の 5 つのパラメーターをよりきめ細かく制御できるように設計されています。

以下に、5 つのルールの例を示します。

```
Source   IP   address :  172 . 16 . 1 . 0 / 32
Source   port :  22
Destinatio  n   IP   address :  10 . 0 . 0 . 1 / 32
Destinatio  n   port :  no   restrictio  n
Transport   layer   protocol :  TCP
Action :  Drop
```

Egress ルールの例では、TCP を介してポート 22 から 10.0.0.1/32 にアクセスする際に、172.16.1.0/32 が禁止されていることを示しています。

シナリオ

- 一部のプラットフォームプロダクトは、サードパーティベンダーのソリューションに接続してネットワークサービスを提供しています。 これらのプロダクトがユーザーの ECS インスタンスに不正にアクセスしないようにするには、セキュリティグループに 5 つのルールを設定して、インバウンドおよびアウトバウンドのトラフィックをより正確に制御する必要があります。
- インスタンスが設定によりセキュリティグループ内で分離されていて、グループ内の複数の ECS インスタンス間のアクセスを正確に制御する場合は、ニーズに合わせてセキュリティグループの 5 つのルールを設定できます。

5 つのルールを設定する方法

OpenAPI を使用して 5 つのルールを設定できます。

- セキュリティグループの Ingress ルールを追加するには、「AuthorizeSecurityGroup」をご参照ください。
- セキュリティグループの Egress ルールを追加するには、「AuthorizeSecurityGroupEgress」をご参照ください
- セキュリティグループの Ingress ルールを削除するには、「RevokeSecurityGroup」をご参照ください。
- セキュリティグループの Egress ルールを削除するには、「RevokeSecurityGroupEgress」をご参照ください。

パラメーター

以下の表でパラメーターについて説明します。

パラメーター	Ingress ルールの意味	Egress ルールの意味
SecurityGroupId	現在の Ingress ルールが属するセキュリティグループの ID (つまり、宛先セキュリティグループの ID)。	現在の Egress ルールが属するセキュリティグループの ID (つまり、送信元セキュリティグループの ID)。
DestCidrIp	宛先 IP アドレス範囲 (任意)。 ・ DestCidrIp が指定されている場合は、Ingress ルールの宛先 IP アドレス範囲をより正確に制御できます。 ・ DestCidrIp が指定されていない場合、Ingress ルールの IP アドレス範囲には、SecurityGroupId で示されるセキュリティグループのすべての IP アドレスが含まれます。	宛先 IP アドレス。 DestGroupId と DestCidrIp のいずれかを指定する必要があります。 両方とも指定されている場合は、DestCidrIp が優先されます。
PortRange	宛先ポート範囲 (必須)。	宛先ポート範囲 (必須)。
DestGroupId	入力不可。 宛先セキュリティグループ ID は SecurityGroupId である必要があります。	宛先セキュリティグループ ID。 DestGroupId と DestCidrIp のいずれかを指定する必要があります。 両方とも指定されている場合は、DestCidrIp が優先されます。
SourceGroupId	送信元セキュリティグループ ID。 SourceGroupId と SourceCidrIp のどちらかを指定する必要があります。 両方とも指定した場合は、SourceCidrIp が優先されます。	入力不可。 Egress ルールの送信元セキュリティグループ ID は SecurityGroupId である必要があります。
SourceCidrIp	送信元 IP アドレス範囲。 SourceGroupId と SourceCidrIp のどちらかを指定する必要があります。 両方とも指定した場合は、SourceCidrIp がより優先されます。	送信元 IP アドレス範囲 (任意)。 ・ SourceCidrIp が指定されている場合は、Egress ルールの送信元 IP アドレス範囲をより正確に制御できます。 ・ SourceCidrIp が指定されていない場合、Egress ルールの送信元 IP アドレスには、SecurityGroupId で示されるセキュリティグループのすべての IP アドレスが含まれます。

パラメーター	Ingress ルールの意味	Egress ルールの意味
SourcePort Range	送信元ポート範囲 (任意)。 指定しない場合、送信元ポートは制限されません。	送信元ポート範囲 (任意)。 指定しない場合、送信元ポートは制限されません。

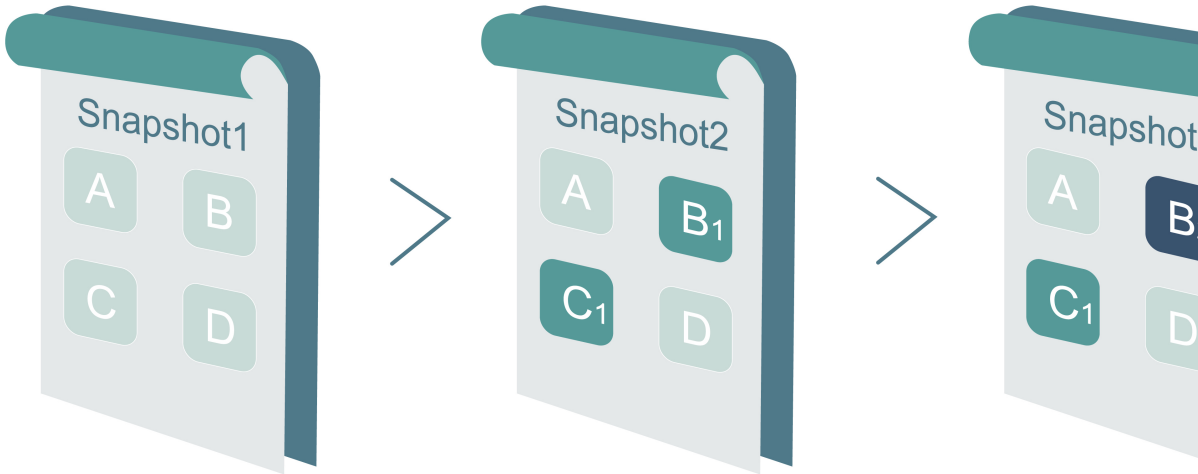
2 ディザスタリカバリソリューション

ディザスタリカバリソリューションは、IT システムの安定性とデータセキュリティを保証するのに役立ちます。 具体的には、ソリューションに、システムおよびアプリケーションのデータバックアップとディザスタリカバリが組み込まれています。 Alibaba Cloud ECS を使用すると、データのバックアップにスナップショットとイメージを使用できます。

ディザスタリカバリ方法

・ スナップショットバックアップ

Alibaba Cloud ECS を使用すると、システムディスクとデータディスクをスナップショットでバックアップできます。 現在、Alibaba Cloud は Snapshot 2.0 サービスを提供しています。これは以前のスナップショットサービスよりも高いスナップショットクォータとより柔軟な自動タスク戦略を特徴とし、業務の入出力への影響を減らすのに役立ちます。スナップショットをデータバックアップに使用する場合、最初のバックアップはフルバックアップで、続いて増分バックアップになります。 バックアップ期間は、バックアップするデータ量によって異なります。



上の図に示すように、スナップショット 1、スナップショット 2、およびスナップショット 3 は、ディスクの 1 番目、2 番目、および 3 番目のスナップショットです。 ファイルシステムはディスクデータをブロック単位でチェックします。 スナップショットが作成されると、データが変更されたブロックのみがスナップショットにコピーされます。 Alibaba Cloud ECS を使用すると、手動または自動のスナップショットバックアップを設定できます。 自動バックアップでは、スナップショット作成の時刻 (24 時間オプション、毎正時)、曜日 (月曜日から日曜日)、および保存期間を指定できます。 保存期間はカスタマイズ可能で、1 ～ 65,536 日の値を設定することも、スナップショットの永続保存を選択することもできます。

- スナップショットロールバック

  システムで例外が発生し、ディスクを以前の状態にロールバックする必要がある場合は、対応するスナップショットが作成されている限り、ディスクをロールバックすることができます。
  注記:

  - ディスクをロールバックすると、元に戻せません。 ディスクのロールバックが完了したら、データを復元することはできません。 このアクションを実行するときはご注意ください。
  - ディスクをロールバックすると、スナップショットの作成時から現在までの間のデータは失われ回復できません。

- イメージバックアップ

  イメージファイルは、1 つ以上のディスク (システムディスク、またはシステムディスクとデータディスクの両方) からのすべてのデータを含むレプリカファイルと同じです。 すべてのイメージバックアップはフルバックアップであり、手動でのみ起動できます。

- イメージリカバリ

  スナップショットからカスタムイメージを作成して、オペレーティングシステムとデータ環境をイメージに含めることができます。 カスタムイメージを使用して、同じオペレーティングシステムとデータ環境で複数のインスタンスを作成できます。 スナップショットとイメージの設定については、「スナップショット」と「イメージ」をご参照ください。

  > 注 :
  > 複数のリージョン間でカスタムイメージを使用することはできません。

技術メトリクス

RTO と RPO: 通常は 1 時間ごとのレベルでデータ量に関連します。

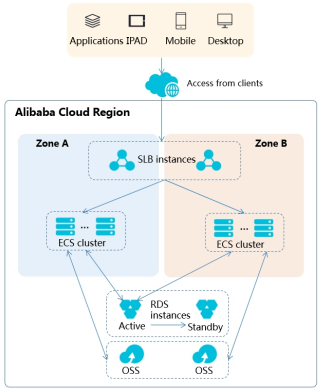
シナリオ

- バックアップと復元

  Alibaba Cloud ECS を使用すると、システムディスクとデータディスクをスナップショットとイメージでバックアップできます。 アプリケーションエラーに起因するデータエラー、または悪意のあるアクセスのためにアプリケーションの脆弱性を悪用するハッカーが原因で、誤ったデータがディスクに保存された場合は、スナップショットサービスを使用してディスクを本来の状態に復元できます。 さらに、Alibaba Cloud ECS を使用すると、イメージ付きのディスクを再初期化したり、カスタムイメージ付きの新しい ECS インスタンスを購入したりすることができます。

・ ディザスタリカバリアプリケーション

Alibaba Cloud ECS はディザスタリカバリアーキテクチャの実装をサポートします。 たとえば、アプリケーションのフロントエンドで SLB (Server Load Balancer) を購入して使用し、同じアプリケーションのバックエンドで少なくとも 2 つの ECS インスタンスをデプロイできます。 または、ECS リソースの使用方法を定義することにより、Alibaba Cloud が提供するオートスケーリングテクノロジーを使用して Auto Scaling ソリューションを実装することができます。 このようにして、ECS インスタンスの 1 つに障害が発生したり、リソースが過剰に使用されたりしても、業務が中断されることはなく、ディザスタリカバリが実現します。 たとえば、同じ都市の 2 つの IDC (インターネットデータセンター) に ECS インスタンスを展開するとします。



- ECS インスタンスのクラスターが両方の IDC にデプロイされています。 アクセス側では、2 つの IDC 間の負荷分散に SLB が使用されます。
- 両方の IDC の Region Master ノードは同一で、アクティブおよびスタンバイモードで動作します。 1 つのノードに障害が発生したからといって、ECS 制御機能が影響を受けることはありません。
- IDC に障害が発生した場合に ECS インスタンスの制御ノードを切り替えるために、ミドルウェアドメイン名はクラスターの制御に使用されるので、新しく関連付けられます。 制御ノードの IDC に問題が発生した場合は、ミドルウェアドメイン名を他の IDC 制御ノードに関連付ける必要があります。

3 データリカバリ

3.1 誤って削除したデータを復元する方法

ここでは、CentOS 7 を例として、誤って削除したデータをすばやく復元するためのオープンソースツールである Extundelete の使い方を紹介します。

概要

作業中に、データを誤って削除してしまうことがあります。 この場合、データをすばやく効果的に復元するにはどうしたらよいでしょうか。 Alibaba Cloud には、以下に示す例のように、データを復元する方法がいくつかあります。

- ECS コンソールからスナップショットまたは カスタムイメージをロールバックします。
- 負荷分散とサービスの高可用性を実装するために、ECS インスタンスをいくつか購入します。
- OSS (Object Storage Service)を使用して、Web ページ、画像、動画など、大量のデータを保存します。

debugfs、R-Linux、ext3grep、Extundelete など、Linux 用のさまざまなオープンソースデータリカバリツールがあります。 その中で、ext3grep と Extundelete が一般的に使用されています。 どちらのツールも同じリカバリ手法を採用していますが、Extundelete の方が強力です。

Extundelete は、Linux ベースのオープンソースのデータリカバリソフトウェアです。 Linux インスタンスを使用する場合、Linux にはごみ箱がないため、誤って削除したデータをすばやく復元するためにこのツールを手軽にインストールできます。

Extundelete は、inode 情報とログを組み合わせることによって inode ブロックの位置を特定し、目的のデータを検索して復元できます。 この強力なツールは、ext3 および ext4 デュアルフォーマットパーティションのディスク全体の復元をサポートします。

誤ってデータを削除してしまった場合は、まず削除したデータを含むディスクまたはディスクパーティションのマウントを解除する必要があります。 これは、ファイルが削除された後、実際のファイルはまだディスクに保存されているのに、そのファイルの inode ポインタだけがゼロに設定されるためです。 ディスクが読み書きモードでマウントされている場合、削除されたファイルのデータブロックは、オペレーティングシステムによって再割り当てされる可能性があります。 データブロックが新しいデータで上書きされると、元のデータは完全に失われ、決して復元することはできません。 したがって、ディスクを読み取り専用モードでマウントすると、データ

がブロック単位で上書きされるリスクを減らすことができるため、データを正常に復元する可能性が高くなります。

 注 :
オンライン復元プロセス中に、削除されたファイルがあるディスクに Extundelete をインストールしないでください。 インストールすると、復元するデータが上書きされる可能性があります。 操作の前にスナップショットを取ってディスクをバックアップすることを忘れないでください。

対象ユーザー

- 誤ってディスク上のファイルを削除し、削除後にディスク上で書き込み操作を行っていないユーザー。
- Web サイトのトラフィックが少なく、ECS インスタンスをほとんど持っていないユーザー。

手順

ソフトウェアリリース: e2fsprogs-devel e2fsprogs gcc-c++ make (コンパイラおよびそれ以上) Extundelete-0.2.4。

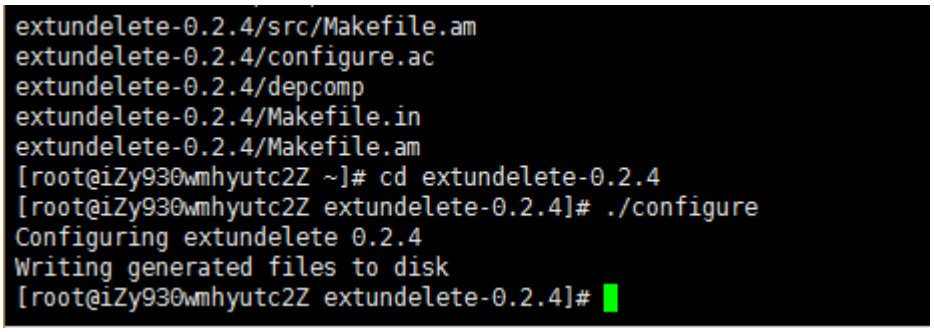
 注 :
Extundelete の通常の操作には、libext2fs 1.39 以上が必要です。 ただし、ext4 をサポートするには、e2fsprogs 1.41 以上が備わっていることを確認します (コマンド `dumpe2fs` を実行して、バージョン出力を確認することができます)。

このページが書かれている時点で利用可能なのは、上記のリリースです。 お客様の手元には、違うものがダウンロードされているかもしれません。

- Extundelete のデプロイ

```
wget    http :// zy - res . oss - cn - hangzhou . aliyuncs . com /
server / extundelet  e - 0 . 2 . 4 . tar . bz2
yum  - y   install    bzip2    e2fsprogs - devel    e2fsprogs
gcc - c ++   make     # Install   related   dependenci  es   and
libraries
tar  - xvjf   extundelet  e - 0 . 2 . 4 . tar . bz2
cd   extundelet  e - 0 . 2 . 4                                  #
Enter   the   program   directory
```

```
./ configure                                                    # Installed
 successful  ly   as   shown   below
```



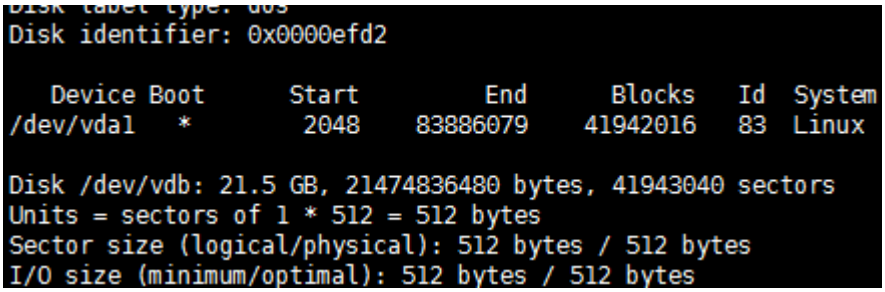

```
 make  &&  make   install
```

この時点で、src ディレクトリが表示されます。 Extundelete 実行可能ファイルと対応するパスが含まれています。 以下のように、デフォルトのファイルが " *usr / local / bin* " にインストールされ、次のデモは " *usr / local / bin* " ディレクトリに作成されます。

- ファイルを削除し、Extundelete を使って復元します。

1. ECS インスタンスの使用可能なディスクとパーティションを確認し、次に /dev/vdb パーティションをフォーマットしてパーティション分割します。 フォーマットとパーティションの詳細は、「データディスクの形式とマウント」をご参照ください。

   ```
   fdisk  - l
   ```

2. パーティションに分割されたディスクを "/zhuyun" ディレクトリ配下にマウントしてから、" *hello* " という名前のファイルを作成します。

   ```
   mkdir  / zhuyun                                  # Create   the
   zhuyun   directory .
   mount  / dev / vdb1  / zhuyun                        # Mount   the
     disk   under   the   zhuyun   directory .
   ```

```
echo  test  >  hello                              # Create  a
test  file .
```

3. **md5sum コマンドを実行してファイルの MD5 値を生成し、書き留めます。 削除前と削除後のファイルの MD5 値 を比較して、ファイルの整合性を確認できます。**

```
md5sum  hello
```

```
[root@iZbp13micdqsi2364umm8aZ zhuyun]# md5sum hello
d8e8fca2dc0f896fd7cb4cb0031ba249  hello
```

4. **" *hello* " ファイルを削除します。**

```
rm  - rf  hello
cd  ~
fuser  - k  / zhuyun                        # Terminate  the
process  tree  that  uses  a  certain  partition  ( skip
  this  if  you  are  sure  that  no  resources  are
occupied ).
```

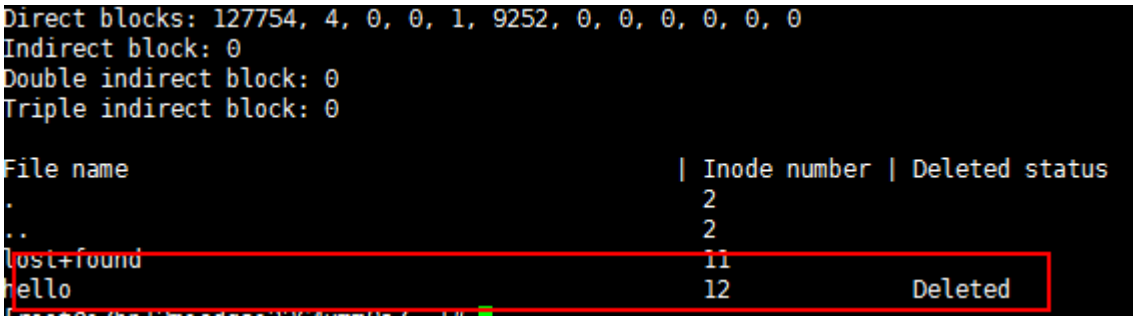
5. **データディスクのマウントを解除します**

```
umount  / dev / vdb1                         # Before  using  any
  file  restoratio  n  tool ,  unmount  or  mount  the
partitions  to  be  restored  in  read - only  mode  to
prevent  their  data  from  being  overwritte  n .
```

6. **Extundelete を使用してファイルを復元します。**

```
extundelet  e  -- inode  2  / dev / vdb1     # Query  the
contents  in  a  certain  inode . " 2 " を使用するとは、パーティション全体を検索するということです。 ディレクトリを検索するには、 inode と
```

ディレクトリを指定するだけです。 これで、削除したファイルと inode を確認できます。




```
/ usr / local / bin / extundelet e -- restore - inode 12 /
 dev / vdb1 # Restore the deleted file .
```

この時点で、RECOVERED_FILES ディレクトリが、コマンドが実行されたディレクトリ配下に表示されます。 ファイルが復元されたかどうかを確認します。

```
[root@iZbp13micdqsi2364umm8aZ /]# ll RECOVERED_FILES/
total 4
-rw-r--r-- 1 root root 5 Mar  8 14:20 hello
```

削除前と削除後のファイルの MD5 値を確認します。 MD5 値が同じであれば、復元は成功です。

```
-- restore - inode 12 # -- restore - inode
 Restore by the specified inode .
-- extundelet e -- restore - all # -- restore - all
 Restore all .
```

3.2 Linux インスタンスでのデータ復元

ディスクに関連する問題を解決する際、データディスクのパーティションを失うことがよくあるかもしれません。 ここでは、よくあるデータパーティション損失の問題とそれに対応する Linux のソリューションについて説明します。また、クラウドディスクのデータ損失のリスクを回避するために、よくある間違いとベストプラクティスを紹介します。

データを復元する前に、パーティションを失うデータディスクのスナップショットを作成する必要があります。 復元プロセス中に問題が発生した場合は、データディスクを復元前の状態にロールバックすることができます。

前提条件

データを復元する前に、パーティションを失うデータディスクのスナップショットを作成する必要があります。 復元プロセス中に問題が発生した場合は、データディスクを復元前の状態にロールバックすることができます。

ディスク管理ツールの紹介

次のツールのうち一つを選択して、ディスクパーティションを修正し、Linux インスタンスのデータを復元することができます。

- fdisk : Linux インスタンスにインストールされているデフォルトのパーティション分割ツールです。
- testdisk : Linux システムのディスクパーティションやデータを復元するために主に使用されます。 このツールはデフォルトでは Linux にインストールされていません。 ご自身でインストールする必要があります。 たとえば、CentOS システムでは、`[ yum install -y testdisk ]` コマンドを実行して、オンラインでインストールすることができます。
- partprobe : Linux システムにインストールされているデフォルトのツールです。 これは、システムを再起動せずにカーネルがパーティションを再読み込みできるようにするために主に使用されます。

Linux でのデータディスクのパーティションの損失とデータの復元処理

Linux インスタンスを再起動した後、データディスクのパーティションの損失またはデータ損失の問題が起こる可能性があります。 これは、 *etc / fstab* ファイルでインスタンスの起動時にパーティションが自動的にマウントされるように設定されていないためです。 この場合、最初にデータディスクのパーティションを手動でマウントできます。 データディスクを手動でマウントする際にシステムがパーティションテーブル損失を引き起こした場合、次の3つの方法、fdisk を使用したパーティションの復元、testdisk を使用したパーティションの復元、 testdisk を使用したデータの復元によって問題の解決を試みることができます。

- fdisk を使用したパーティションの復元

  データディスクをパーティション分割する場合、デフォルト値は通常、パーティションの開始セクタと終了セクタに適用されます。 その後、fdisk を使ってパーティションを復元できま

す。 このツールの詳細については、「Linux データディスクのフォーマットとマウント」をご参照ください。

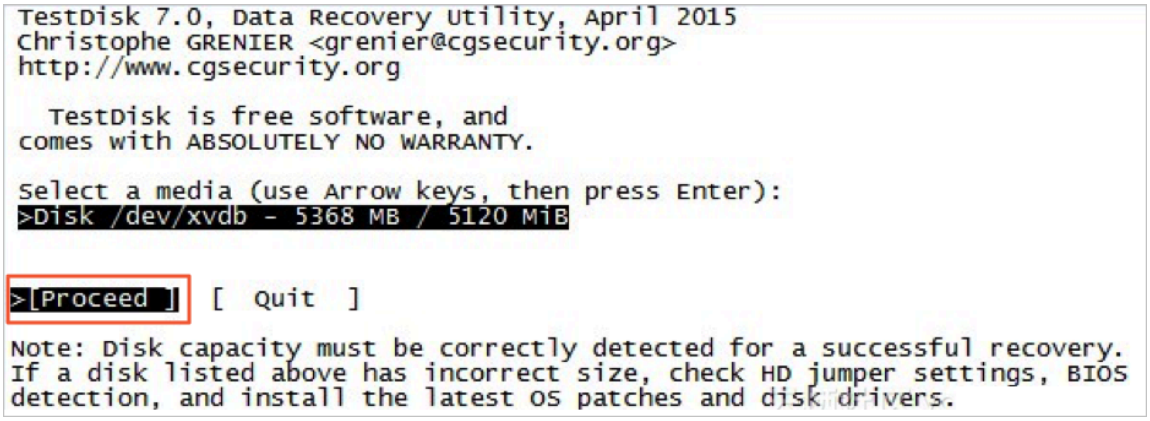
```
[root@Aliyun ~]# fdisk /dev/xvdb
Welcome to fdisk (util-linux 2.23.2).

Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.


Command (m for help): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-10485759, default 2048):
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-10485759, default 10485759):
Using default value 10485759
Partition 1 of type Linux and of size 5 GiB is set

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
[root@Aliyun ~]# mount /dev/xvd
xvda   xvda1  xvdb   xvdb1
[root@Aliyun ~]# mount /dev/xvdb
xvdb   xvdb1
[root@Aliyun ~]# mount /dev/xvdb1 /mnt/
[root@Aliyun ~]# ls /mnt/
123.sh  configclient  data  diamond  install_edsd.sh  install.sh  ip.qz
```

上記の操作で問題が解決しない場合は、testdisk で復元を試すことができます。

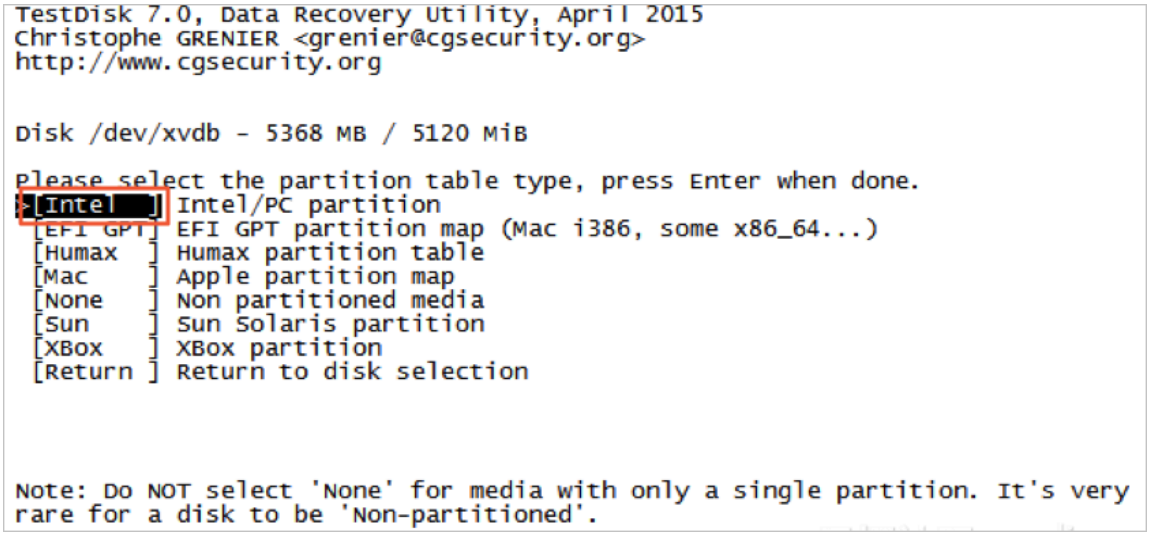
- **testdisk を使用したパーティションの復元**

  ここでは、クラウドディスクデバイスの名前が */ dev / xvdb* **であるとします。 次の手順に従って、testdisk を使用してパーティションを復元します。**

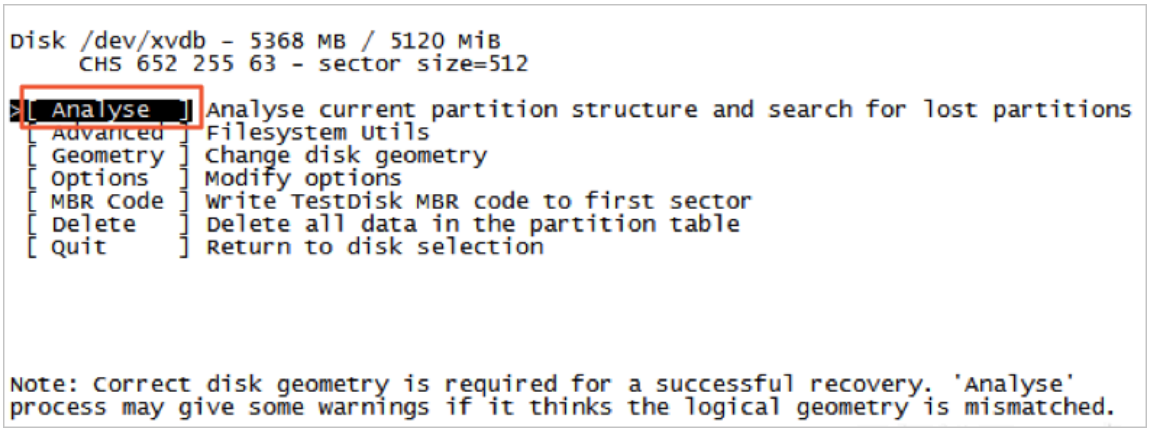
  1. **[** `testdisk / dev / xvdb` **] を実行 (必要に応じてデバイス名を変更) し、[** `Proceed` **] (デフォルト値) を選択して、Enter キーを押します。**



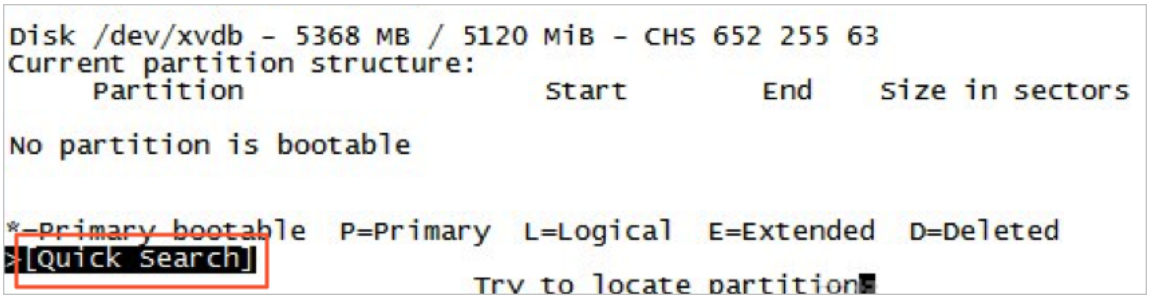
  2. **スキャン対象のパーティションテーブルの種類を選択します。デフォルトは** *Intel* **です。データディスクが GPT フォーマットを使用している場合は、** *EFI GPT* **を選択します。**



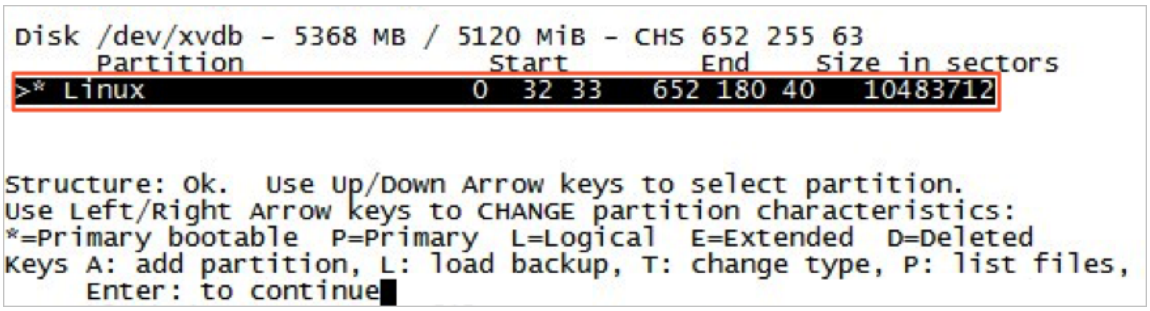
  3. *Analyse* **を選択し、Enter キーを押します。**



4. パーティションが見えない場合は、*Quick Search* を選択し、Enter キーを押すとクイック検索ができます。



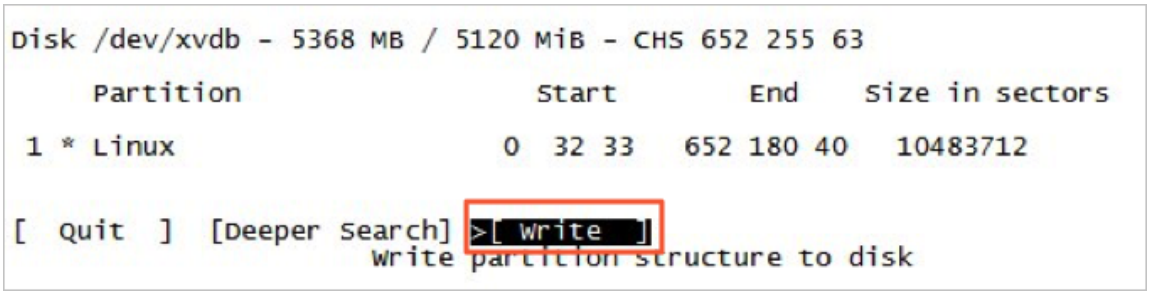
次の図に示すように、パーティション情報が返された結果に表示されます。



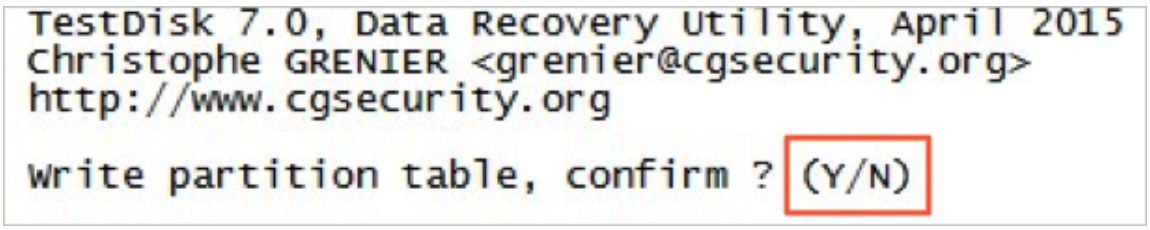
5. パーティションを選択して Enter キーを押します。
6. *Write*を選択してパーティションを保存します。

 注：

*Deeper Search*を選択して、期待するパーティションがリストされていない場合は検索を続行します。



7. Y キーを押してパーティションを保存します。

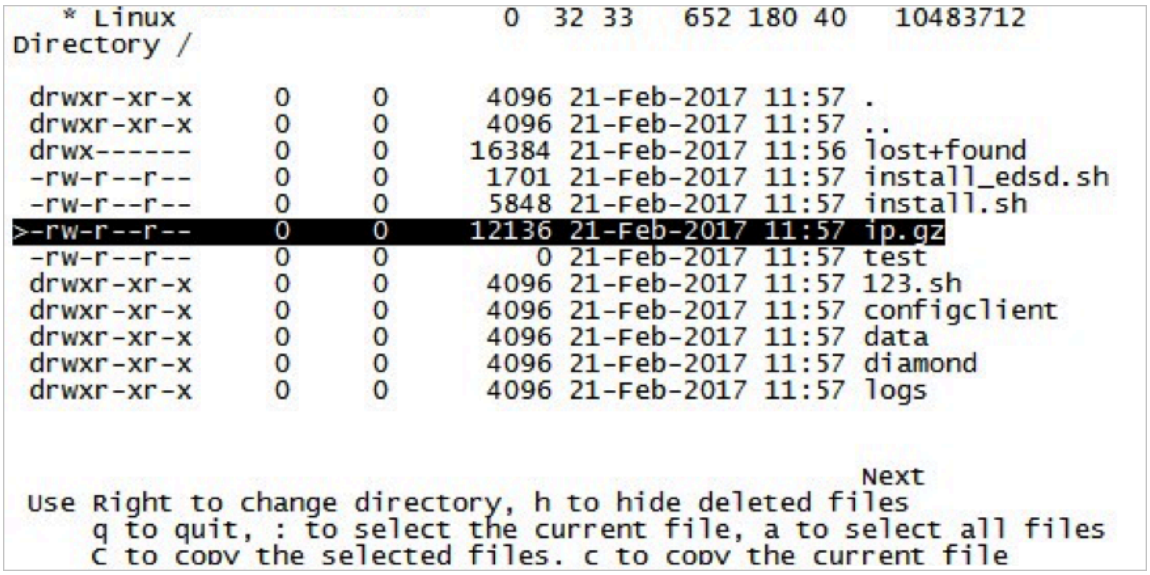


8. partprobe / dev / xvdb を実行 (必要に応じてデバイス名を変更) して、パーティションテーブルを手動で更新します。
9. パーティションを再度マウントして、データディスクのデータを確認します。

- testdisk を使用したデータを復元

  場合によっては、testdisk を使用してディスクパーティションをスキャンして見つけることができますが、パーティションの保存はできません。 この場合、ファイルを直接復元することを試めすことができます。 次の手順を実行します。

  1. 「testdisk を使用したデータを復元」で説明したステップ 1 ～ステップ 4 に従ってパーティションを検索します。
  2. P キーを押してファイルを一覧表示します。 返された結果を次の図に示します。



  3. 復元するファイルを選択して C キーを押します。
  4. ディレクトリを選択します。 この例では、ファイルが復元されて / *home* ディレクトリにコピーされます。

```
Please select a destination where /ip.gz will be copied.
Keys: Arrow keys to select another directory
      C when the destination is correct
      Q to quit
Directory /
 drwxr-xr-x     0     0      4096 11-Jan-2017 09:32 .
 drwxr-xr-x     0     0      4096 11-Jan-2017 09:32 ..
 dr-xr-xr-x     0     0      4096 25-Jul-2016 16:23 boot
 drwxr-xr-x     0     0      2940 21-Feb-2017 12:30 dev
 drwxr-xr-x     0     0      4096 21-Feb-2017 12:12 etc
>drwxr-xr-x     0     0      4096 16-Feb-2017 11:48 home
 drwx------     0     0     16384 12-May-2016 19:58 lost+found
 drwxr-xr-x     0     0      4096 12-Aug-2015 22:22 media
 drwxr-xr-x     0     0      4096 21-Feb-2017 11:57 mnt
 drwxr-xr-x     0     0      4096 12-Aug-2015 22:22 opt
 dr-xr-xr-x     0     0         0 16-Feb-2017 21:35 proc
 dr-xr-x---     0     0      4096 21-Feb-2017 11:57 root
 drwxr-xr-x     0     0       560 21-Feb-2017 12:12 run
 drwxr-xr-x     0     0      4096 12-Aug-2015 22:22 srv
 dr-xr-xr-x     0     0         0 16-Feb-2017 21:35 sys
 drwxrwxrwt     0     0      4096 21-Feb-2017 12:34 tmp
 drwxr-xr-x     0     0      4096 16-Feb-2017 11:48 usr
 drwxr-xr-x     0     0      4096 16-Feb-2017 21:35 var
 lrwxrwxrwx     0     0         7  3-May-2016 13:48 bin
 lrwxrwxrwx     0     0         7  3-May-2016 13:48 lib
 lrwxrwxrwx     0     0         9  3-May-2016 13:48 lib64
 lrwxrwxrwx     0     0         8  3-May-2016 13:48 sbin
```

`Copy done ! 1 ok , 0 failed` が表示されれば、次の図に示すように、コピーが成功したことを示します。

```
   * Linux                          0  32 33   652 180 40   10483712
Directory /
Copy done! 1 ok, 0 failed
 drwxr-xr-x     0     0      4096 21-Feb-2017 11:57 .
 drwxr-xr-x     0     0      4096 21-Feb-2017 11:57 ..
 drwx------     0     0     16384 21-Feb-2017 11:56 lost+found
 -rw-r--r--     0     0      1701 21-Feb-2017 11:57 install_edsd.sh
 -rw-r--r--     0     0      5848 21-Feb-2017 11:57 install.sh
>-rw-r--r--     0     0     12136 21-Feb-2017 11:57 ip.gz
 -rw-r--r--     0     0         0 21-Feb-2017 11:57 test
 drwxr-xr-x     0     0      4096 21-Feb-2017 11:57 123.sh
 drwxr-xr-x     0     0      4096 21-Feb-2017 11:57 configclient
 drwxr-xr-x     0     0      4096 21-Feb-2017 11:57 data
 drwxr-xr-x     0     0      4096 21-Feb-2017 11:57 diamond
 drwxr-xr-x     0     0      4096 21-Feb-2017 11:57 logs
```

5. */ home* ディレクトリに切り替えて詳細を表示します。ファイルが表示されている場合は、ファイルが正常に復元されたことを示しています。

```
[root@Aliyun /]# ls /home/
admin  ip.gz
[root@Aliyun /]#
```

よくある間違いとベストプラクティス

データはユーザーのコア資産です。 多くのユーザーが ECS 上に Web サイトとデータベース (MYSQL や MongoDB や Redis) を確立しています。 データが失われると、ユーザーのサービスに対する大きなリスクが発生する可能性があります。 よくある間違いとベストプラクティスは、次のようにまとめられています。

- よくある間違い

  Alibaba Cloud ブロックレベルストレージの最下層は、三重化技術に基づいています。 したがって、一部のユーザーは、オペレーティングシステムでデータが失われる危険性はないと考えています。 実際にはそれは誤解です。 最下層に格納されている 3 つのデータコピーは、データディスクの物理層を保護します。 ただし、ウイルス、偶発的なデータ削除、ファイルシステムの損傷など、システム内のクラウドディスクロジックに問題が発生した場合でも、データは失われる可能性があります。 データセキュリティを確保するためには、スナップショットやバックアップなどの技術を使用する必要があります。
- ベストプラクティス

  データディスクのパーティションの復元とデータの復元はデータ損失の問題を解決するための最終的な解決策ですが、それは保証されるものではありません。 ベストプラクティスに従ってデータの自動または手動スナップショットを実行し、データセキュリティを最大限に高めるためにさまざまなバックアップスキームを実行することを強く推奨します。

  - 自動スナップショットの有効化

    実際のサービス状態に基づいて、システムディスクとデータディスクに対して自動スナップショットが有効になります。 留意すべき点は、システムディスクが変更されたとき、イ

ンスタンスが期限切れになったとき、またはディスクが手動で解放されたときに、自動スナップショットが解放されることがあることです。

ECS コンソールにログインして ディスクの属性を変更し、[ディスクのスナップショットリリース] を有効にします。 スナップショットを保持する場合は、ディスクのスナップショットリリースを無効にします。

詳細は、「自動スナップショットに関するよくある質問」をご参照ください。

- 手動スナップショットの作成

  次のような重要な操作またはリスクを伴う操作の前に、スナップショットを手動で作成します。

  ■ カーネルのアップグレード
  ■ アプリケーションのアップグレードまたは変更
  ■ ディスクデータの復元

  復元する前に、ディスクのスナップショットを作成する必要があります。 スナップショットが完成したら、他の操作を実行できます。

- OSS、オフライン、またはオフサイトバックアップ

  重要なデータは、実際の状況に基づいて、OSS 、オフライン、またはオフサイトバックアップによってバックアップできます。

3.3 Windows インスタンスでのデータ復元

ディスクに関連する問題を解決する際、データディスクのパーティションを失うことがよくあるかもしれません。 ここでは、よくあるデータパーティション損失の問題とそれに対応する Windows のソリューションについて説明します。また、クラウドディスクのデータ損失のリスクを回避するために、よくある間違いとベストプラクティスを紹介します。

前提条件

データを復元する前に、パーティションを失うデータディスクのスナップショットを作成する必要があります。 復元プロセス中に問題が発生した場合は、データディスクを復元前の状態にロールバックすることができます。

ディスク管理ツールの紹介

Windows インスタンスでは、次のツールのうちいずれかを選択して、データディスクデータを復元することができます。

- ディスク管理: ディスクのパーティション分割とフォーマット用に Windows が提供するツールです。
- データ復元ソフトウェア: 一般的に、市販ソフトウェアであり、プロバイダーの公式 Web サイトからダウンロードできます。 これらは主に、異常なファイルシステムのデータを復元するために使用されます。

ディスクのステータスが Foreign で、パーティションが表示されません

Windows の [ディスクの管理] で、ディスクのステータスは [Foreign] で、パーティションは表示されません。

解決策:

[Foreign] ディスクを右クリックし、[Foreign ディスクのインポート] を選択して、[OK] をクリックします。

ディスクのステータスがオフラインで、パーティションが表示されません

Windows の [ディスクの管理] で、ディスクのステータスは [オフライン] で、パーティションは表示されません。

解決策:

[オフライン] ディスク (たとえば [ディスク 1]) を右クリックし、[オンライン] を選択して [OK] をクリックします。

ドライブ文字が割り当てられていません

Windows の [ディスクの管理] で、データディスク情報を表示できますが、データディスクにドライブ文字が割り当てられていません。

解決策:

ディスクのプライマリパーティション (たとえば、[ディスク 1]) を右クリックし、[ドライブ文字とパスの変更] をクリックして、プロンプトで操作を完了します。

ストレージの列挙中にエラーが発生しました

Windows の [ディスクの管理] で、データディスクを表示できません。 ストレージの列挙中に発生したエラーはシステムログに報告されます。

> 注 :
> 一部のバージョンでは、ボリュームの列挙中にエラーが発生したと報告されることがあります。それらは同じです。

解決策:

1. Windows PowerShell を起動します。
2. `winrm quickconfig -q` を実行して復元します。"これらの変更を加えますか [y / n] ?" と インターフェイスに表示されたら、「*y*」と入力してコマンドを実行する必要があります。

復元後、[ディスクの管理] にデータディスクを入れることができます。

データディスクの形式が RAW です

特別な状況下では、Windows のディスクは RAW 形式です。

ディスクのファイルシステムが Windows に認識されない場合は、RAW ディスクとして表示されます。 これは通常、ファイルシステムの種類や場所を記録しているパーティションテーブルやブートセクターが失われたり破損したりした場合に発生します。 一般的な原因は次のとおりです。

- 外付けディスクを取り外す際に、[ハードウェアの安全な取り外し] を使用していません。
- ディスクの問題が、停電や予期しないシャットダウンに起因しています。
- ハードウェア層の障害によって、ディスクパーティションの情報が失われる可能性もあります。
- 最下層ドライバーまたはディスク関連アプリケーション。 たとえば、DiskProbe を使用してディスクテーブル構造を直接変更できます。
- コンピューターウイルス。

これらの問題を解決する方法の詳細は、『Dskprobe Overview』をご参照ください。

さらに、Windows には、失われたデータを復元するためのさまざまな無料または市販のデータ復元ソフトウェアも含まれています。 たとえば、Disk Genius を使用し、目的の文書をスキャンして復元することができます。

よくある間違いとベストプラクティス

データはユーザーのコア資産です。 多くのユーザーが ECS 上に Web サイトとデータベース (MYSQL/MongoDB/Redis) を確立しています。 データが失われると、ユーザーのサービスに対して大きなリスクが発生する可能性があります。 よくある間違いとベストプラクティスは、次のようにまとめられています。

- よくある間違い

  Alibaba Cloud ブロックレベルストレージの最下層は、トリプリケートテクノロジーに基づいています。 したがって、一部のユーザーは、オペレーティングシステムでデータが失われる危険性はないと考えています。 それは実際には誤解です。 最下層に格納されている 3 つの

データコピーは、データディスクの物理層を保護します。 ただし、ウイルス、偶発的なデータ削除、ファイルシステムの損傷など、システム内のクラウドディスクロジックに問題が発生した場合でも、データは失われる可能性があります。 データセキュリティを確保するためには、スナップショットやバックアップなどの技術を使用する必要があります。

- ベストプラクティス

  データディスクパーティションの復元とデータの復元は、データ損失の問題を解決するための最終的な解決策ですが、その解決は保証されるものではありません。 ベストプラクティスに従ってデータの自動または手動スナップショットを実行し、データセキュリティを最大限に高めるためにさまざまなバックアップスキームを実行することを強く推奨します。

  - 自動スナップショットの有効化

    実際のサービス状態に基づいて、システムディスクとデータディスクに対して自動スナップショットが有効になります。 留意すべき点は、システムディスクが変更されたとき、インスタンスが期限切れになったとき、またはディスクが手動で解放されたときに、自動スナップショットが解放されることがあることです。

    ECS コンソールにログインしてディスクの属性を変更 し、[ディスクのスナップショットリリースの有効化] を行います。 スナップショットを保持する場合は、ディスクのスナップショットリリースを無効にします。

    詳細は、「自動スナップショットに関するよくある質問」をご参照ください。

  - 手動スナップショットの作成

    次のような重要または危険な操作の前に、手動でスナップショットを作成します。

    ■ カーネルのアップグレード
    ■ アプリケーションのアップグレードまたは変更
    ■ ディスクデータの復元

    復元する前に、ディスクのスナップショットを作成する必要があります。 スナップショットが完了したら、他の操作を実行できます。

  - OSS、オフライン、またはオフサイトバックアップ

    重要なデータは、実際の状況に基づいて、OSS、オフライン、またはオフサイトバックアップによってバックアップできます。

4 設定の優先度

4.1 複数インスタンスの言語の設定方法

このチュートリアルでは、例としてドイツ語を取りあげます。 ドイツ語パッケージは、Windows Update からダウンロードします。 ドイツ語とドイツ語のキーボード設定を使用するカスタムイメージが作成されます。 その後、カスタムイメージを使用して、必要な数のインスタンスを作成できます。

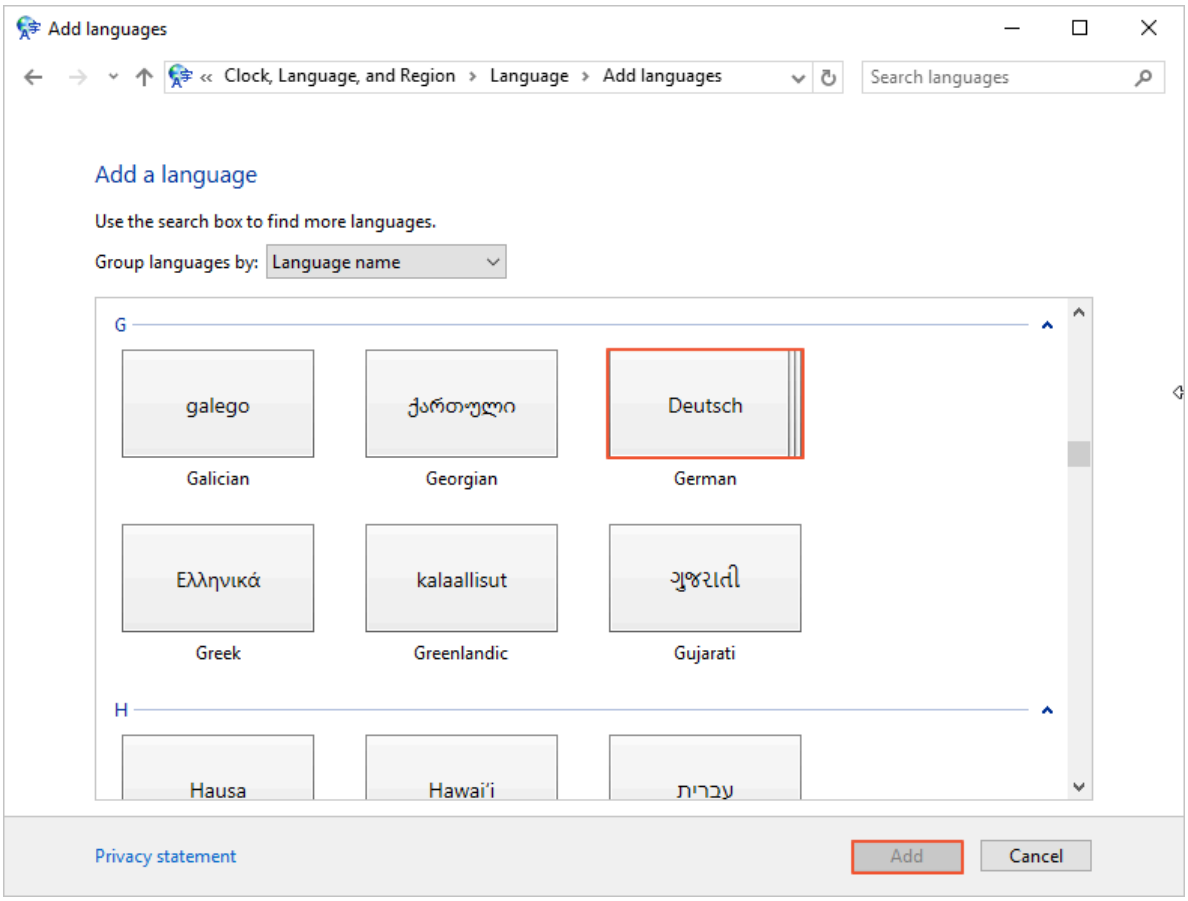
現在、Alibaba Cloud ECS は Windows Server イメージの中国語版と英語版のみを提供しています。 アラビア語、ドイツ語、ロシア語など、その他の言語版を使用する場合は、このチュートリアルに従って ECS インスタンスの設定およびデプロイができます。

1. Windows インスタンスに接続します。
2. PowerShell モジュールを開きます。
3. 次のコマンドを実行し、 WSUS を一時的に無効にします。

```
Set - ItemProper  ty  - Path  ' HKLM :\ SOFTWARE \ Policies \
Microsoft \ Windows \ WindowsUpd  ate \ AU ' - Name  UseWUServe  r
 - Value  0
Restart - Service  - Name  wuauserv
```

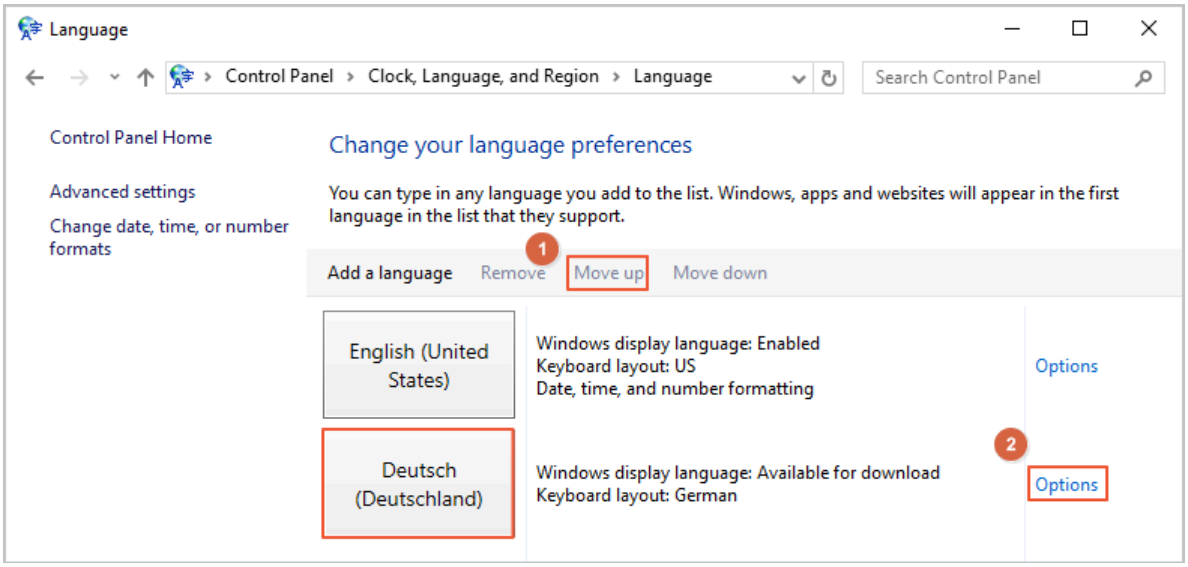
4. [コントロールパネル] を開き、[時計、言語、および地域] > [言語] > [言語を追加] の順にクリックします。

5. [言語を追加] ダイアログボックスで、たとえば、[ドイツ語 (ドイツ)] > [ドイツ語 (ドイツ)] の順に選択し、[追加] をクリックします。

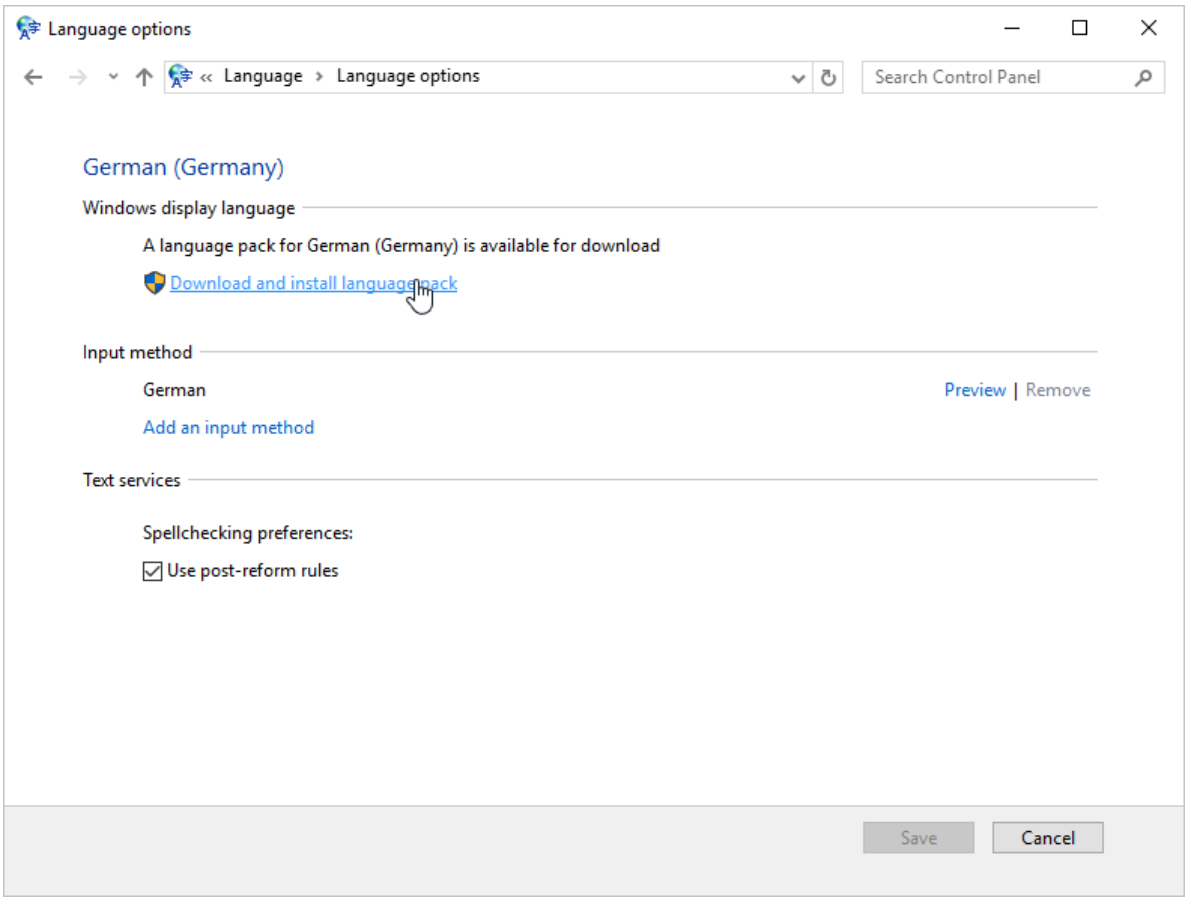


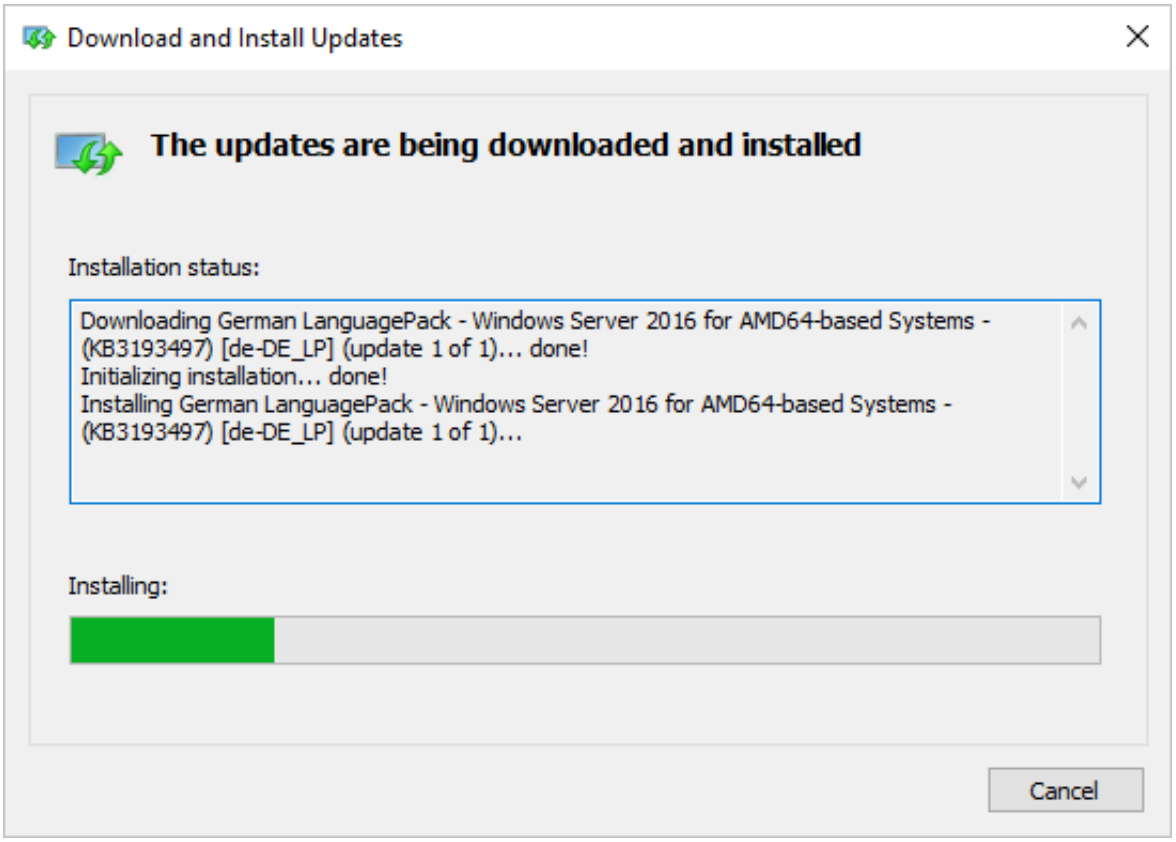
6. ドイツ語 (ドイツ) の言語を選択し、 [上に移動] をクリックして言語の優先順位を変更します。

7. 選択した言語の横の [オプション] をクリックし、言語の更新をオンラインで確認します。



8. インスタンスが更新を確認するまで、約 3 分待ちます。 更新がダウンロードできるようになったら、[言語パックのダウンロードとインストール] をクリックし、インストールが完了するまで待ちます。





9. インスタンスを再起動すると、次回のログイン時に表示言語が変更されます。
10. 再び、Windows インスタンスに接続します。 表示言語は、ドイツ語 (ドイツ) になります。
11. PowerShell ISE モジュールを開き、次のコマンドを実行して WSUS を再び有効にします。

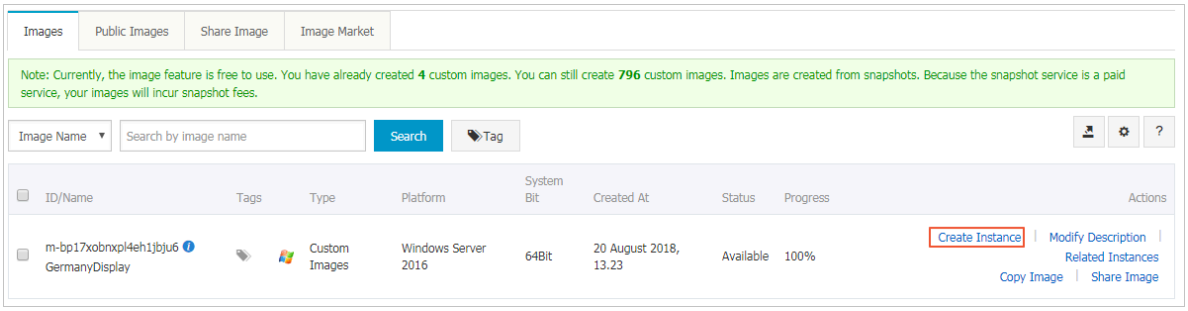
```
Set - ItemProper  ty  - Path  ' HKLM :\ SOFTWARE \ Policies \
Microsoft \ Windows \ WindowsUpd  ate \ AU ' - Name  UseWUServe  r
 - Value  1
Restart - Service  - Name  wuauserv
```

12. [Windows Update] を開き、セキュリティ更新プログラムを確認し、言語設定の前に既に実行されているセキュリティ更新プログラムをすべて再インストールします。

同じ言語設定での複数のインスタンスの作成

1. ECS console にログインします。
2. そして新しい表示言語で Windows インスタンスを使用してカスタムイメージを作成します。

3. カスタムイメージから指定した数のインスタンスを作成します。

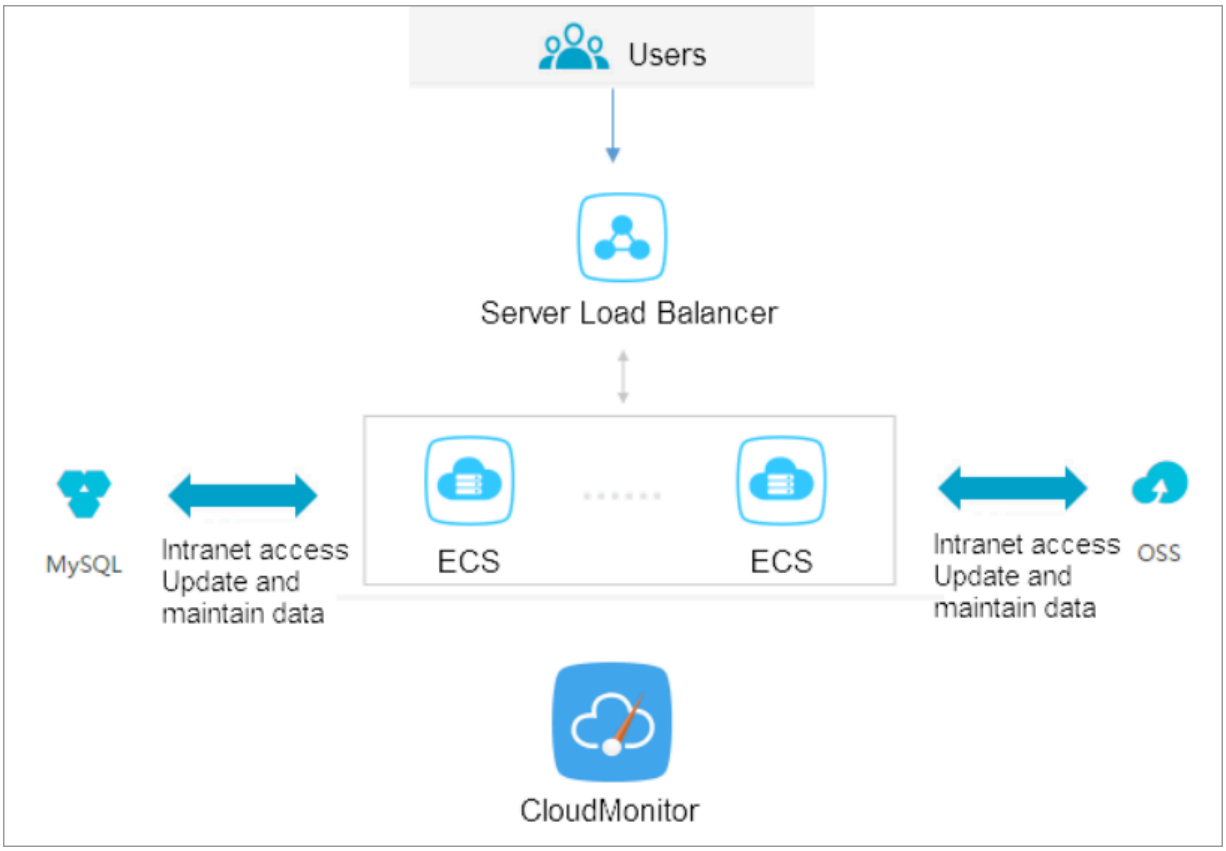


5 モニター

5.1 CloudMonitor を使用した ECS インスタンスのモニター

多くの企業は、費用対効果が高く、顧客の負担が軽減されるため、クラウドコンピューティングに移行しつつあります。 このことが、モニタリングの活用に大きく起因している可能性があります。 モニタリングサービスは、リスクを事前に特定し、潜在的な損失を回避し、可能な限り迅速にトラブルシューティングするためのリアルタイムの運用データを提供します。

ここでは、CloudMonitor の設定方法を説明するために例として Web サイトを取り上げます (Web サイトのアーキテクチャは以下のとおりです)。 この例でWeb サイトは、ECS、RDS、OSS、Server Load Balancer などの Alibaba Cloud サービスを使用しています。



前提条件

始める前に、以下の操作を完了する必要があります。

- ECS モニタリングエージェントがメトリックデータを収集するために機能していることを確認します。 それ以外の場合は、エージェントを手動でインストールする必要があります。 詳細については、「CloudMonitor エージェントのインストール方法」をご参照ください。
- アラーム連絡先と連絡先グループを追加します。 モニタリングアラームに対するリアルタイムの応答を確実にするために、少なくとも 2 つの連絡先を追加することを推奨します。 メトリクスの詳細については、「クラウドサービスの概要とアラームの概要」をご参照ください。
- CloudMonitor ダッシュボードを使用すると、リソース使用率と運用状態についてシステム全体の状況を把握することができます。 メトリクスディメンションを選択できます。 インスタンスが複数ある場合のみ、インスタンスごとのメトリクスディメンションを選択できます。

  それ以外の場合は、ECS グループディメンションまたはユーザーディメンションを選択して、平均値を選択できます。

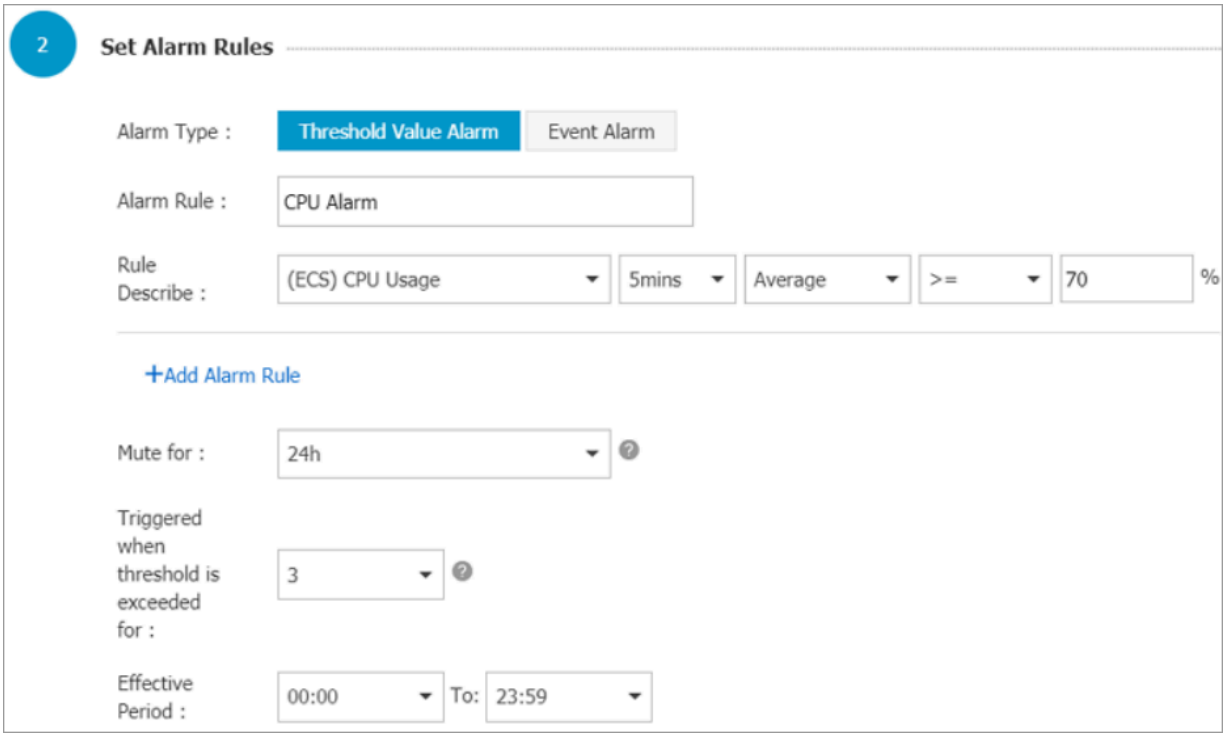
アラームしきい値の設定

業務の状況に応じて、アラームしきい値を設定することを推奨します。 しきい値が低すぎると頻繁にアラームが作動し、モニタリングが無意味になる可能性がありますが、しきい値が高すぎると、主要なイベントに応答する時間がなくなる可能性があります。

アラームのルールの設定

例として CPU 使用率を取り上げます。 次の図に示すように、正常な機能を保証するためにある程度の処理能力を確保する必要があるため、しきい値を 70% に設定し、しきい値を 3 回連続で超えた場合にアラームが作動するようにできます。

他のメトリクスにアラームのルールを設定する必要がある場合は、[アラームのルールの追加] をクリックします。



プロセスモニタリングの設定

Web アプリケーションの場合、 プロセスのモニタリングを追加することができます。 詳細については、「プロセスのモニタリング」をご参照ください。

サイトのモニタリングの設定

サイトのモニタリングは、可用性をテストするためにネットワークアクセス層で行われます。

RDS モニタリングの設定

RDS CPU 使用率アラームしきい値を 70% に設定し、しきい値を 3 回連続で超えた場合にアラームが作動するようにすることを推奨します。 必要に応じて、ディスク使用率、IOPS 使用率、合計接続数などのメトリクスを設定できます。

Server Load Balancer モニタリングの設定

始める前に、Server Load Balancer インスタンスのヘルスチェックを有効にしていることを確認してください。

必要なメトリクスがカバーされていない場合は、カスタマイズモニタリングメトリクスを使用できます。

6 インスタンス RAM ロールによる他のクラウドプロダクト API へのアクセス

以前は、ECS インスタンスにデプロイされたアプリケーションは通常、他の Alibaba Cloud プロダクトの API にアクセスするために、AccessKey ID と AK (AccessKey Secret) を使用する必要がありました。 AK は Alibaba Cloud API にアクセスするためのキーであり、対応するアカウントのすべての権限を持っています。 アプリケーションが AK を管理できるようにするには、AK をアプリケーション設定ファイルに保存するか、他の方法を使用して ECS インスタンスに保存する必要があり、これにより、AK の管理がより複雑になり、機密性が低下します。 さらに、複数のリージョンにわたって同時デプロイが必要な場合は、AK が、イメージまたはイメージによって作成されたインスタンスと共に拡散されるため、AK を変更するときにインスタンスとイメージを1つずつ更新および再度デプロイする必要があります。

インスタンス RAM ロールを利用して、ECS インスタンスに RAM ロールを割り当てます。 インスタンス上のアプリケーションは、STS 資格情報を使用して他のクラウドプロダクトの API にアクセスできます。 STS 資格情報はシステムによって自動的に生成および更新され、アプリケーションは指定されたメタデータ URL を使用し、特別な管理なしに STS 資格情報を取得します。 その間、RAM ロールと権限付与ポリシーを変更して、インスタンスに対する異なるまたは同一の権限を、異なる Alibaba Cloud プロダクトに付与します。

ここでは、RAM ロールを果たす ECS インスタンスを作成する方法と、ECS インスタンス上のアプリケーションを、 STS 資格情報を使用して他の Alibaba Cloud プロダクトにアクセスする方法を紹介します。

> 注 :
> ここでの例を簡単に始めるために、文書内のすべての操作は OpenAPI Explorer で行います。OpenAPI Explorer は、記録されたユーザー情報を介して現在のアカウントの一時的な AK を取得し、現在のアカウントに対してオンラインリソース操作を開始します。 操作は慎重に行ってください。 インスタンスを作成すると料金が発生します。 操作が完了したらすぐにインスタンスをリリースしてください。

手順

インスタンス RAM ロールを使用して、インスタンス上の Python が同じアカウントで OSS バケットにアクセスできるようにするには、次の手順を実行します。

手順 1. RAM ロール を作成して権限付与ポリシーにアタッチします。

手順 2 . 作成する RAM のロールを果たす ECS インスタンスを作成します。

手順 3 . インスタンス内で、メタデータ URL にアクセスして STS 資格情報を取得します。

手順 4 . STS 資格情報を使用し、Python を使用して OSS にアクセスし ます。

手順 1. RAM **ロール の作成および権限付与ポリシーへのアタッチ**

CreateRole API を使用して

1. RAM ロールを作成します。 必要なリクエストのパラメーターは以下のとおりです。
   - RoleName: ロールの名前を指定します。 この例では *EcsRamRoleTest* が使用されています。
   - AssumeRolePolicyDocument: 次のようにポリシーを指定します。これは、作成されるロールがサービスロールであり、Alibaba Cloud プロダクト (この例では ECS) がこのロールを果たすように割り当てられていることを示します。

```
{
" Statement ": [
{
" Action ":" sts : AssumeRole ",
" Effect ":" Allow ",
" Principal ": {
" Service ":[
  & quot ; ecs . aliyuncs . com & quot ;
]
}
}
],
" Version ":" 1 "
}
```

2. CreatePolicy APIを使用して、権限付与ポリシーを作成します。 必要なリクエストのパラメーターは以下のとおりです。
   - PolicyName: 権限付与ポリシーの名前を指定します。 この例では *EcsRamRolePolicyTest* が使用されています。
   - PolicyDocument: 次のようにポリシーを指定します。これは、ロールが OSS 読み取り専用権限を持っていることを示します。

```
{
" Statement ": [
{
" Action ": [
" oss : Get *",
" oss : List *"
],
" Effect ":" Allow ",
" Resource ":"*"
}
],
" Version ":" 1 "
```

```
}
```

3. AttachPolicyToRole API を使用して、権限付与ポリシーをロールにアタッチします。 必要なリクエストのパラメーターは以下のとおりです。
   - PolicyType: カスタムに設定します。
   - PolicyName: 手順 2 で指定したポリシー名を 使用します。 この例では *EcsRamRole PolicyTes* を使用します。
   - RoleName: 手順 1 で指定したロール名を使用します。 この例では *EcsRamRoleTest* を使用します。

手順 2. RAM **ロールを再生する** ECS **インスタンスを作成する**

どちらの方法でも、RAM ロールを再生する ECS インスタンスを作成できます。

- 既存の VPC 接続 ECS インスタンスにRAM ロールをアタッチする
- RAM ロールを持つ VPC 接続 ECS インスタンスの作成

既存の VPC 接続 ECS インスタンスにRAM ロールをアタッチする

AttachInstanceRamRole API を使用して、既存の VPC 接続 ECS インスタンスに RAM ロールをアタッチします。 パラメーターは以下のとおりです。

- RegionId: インスタンスが置かれているリージョンの ID。
- RamRoleName: RAM ロールの名前。 この例では、EcsRamRoleTest が使用されています。 この例では、 *EcsRamRoleTest* です。
- InstanceIds: RAM ロールをアタッチする VPC 接続 ECS インスタンスの ID。1 つのインスタンスの場合は ["i-bXXXXXXX"]、複数のインスタンスの場合は ["i-bXXXXX"、"i-cXXXXX"、["i-bXXXXXXXX"] の形式になります。

RAM ロールを持つ VPC 接続 ECS インスタンスの作成の作成

RAM ロールを持つ ECS インスタンスを作成する前に、VPC ネットワークが必要です。

1. RAM ロールを持つ VPC 接続 ECS インスタンスを作成するには、次の手順に従います。CreateInstance API を使用して ECS インスタンスを作成します。 必要なリクエストのパラメーターは以下のとおりです。

   - RegionId: インスタンスのリージョン。 この例では、cn-hangzhou が使用されています。 この例では、*cn-hangzhou* が使用されています。
   - ImageId: インスタンスのイメージ。 この例では、centos_7_03_64_40G_alibase_20170503.vhd が使用されています。 この例では、*centos_7_03_64_40G_alibase_20170503.vhd* が使用されています。
   - InstanceType: インスタンスのタイプ。 この例では、*ecs.xn4.small* が使用されています。
   - VSwitchId: インスタンスが置かれている VPC ネットワークの仮想スイッチ。 インスタンス RAM のロールは VPC ネットワークのみをサポートするため、VSwitchId が必要です。
   - RamRoleName: RAM ロールの名前。 この例では、*EcsRamRoleTest* が使用されています。

   ECS インスタンスを作成する権限に加えて、サブアカウントを承認して指定された RAM のロールを果たす ECS インスタンスを作成する場合は、サブアカウントに PassRole 権限が必要です。 したがって、権限付与ポリシーを次のようにカスタマイズしてサブアカウントにアタッチする必要があります。 アクションが ECS インスタンスのみを作成している場合は、[ECS RAM アクション] を" `ecs : CreateInst ance` "に設定します。 サブアカウントに対するすべての ECS アクション権限を付与する場合は、[ECS RAM Action] を" `ecs :*` " に設定します。

   ```
   {
     " Statement ": [
       {
         " ecs : [ ECS   RAM   Action ]",
         " Resource ":"*",
         " Effect ":" Allow "
       },
    {
         " Action ":" ram : PassRole ",
         " Resource ":"*",
         " Effect ":" Allow "
     ],
     " Version ":" 1 "
   }
   ```

2. パスワードを設定してインスタンスを起動します。
3. ECS インスタンスを設定して、API を使用するか ECS コンソールでインターネットにアクセスします。

手順 3: インスタンス内のメタデータ URL にアクセスして STS 資格情報を取得

インスタンスの STS 資格情報を取得するには、次の手順を実行します。

注：

現在の STS 資格情報が期限切れになる 30 分前に、新しい ものが生成されます。 この期間中は、両方の STS 資格情報を使用できます。

1. インスタンスに接続します。
2. `http :// 100 . 100 . 100 . 200 / latest / meta - data / ram / security - credential s / EcsRamRole Test` にアクセスして STS 資格情報を取得します。URLの最後の部分は RAM ロール名です。作成する名前と置き換える必要があります。 パスの最後の部分は RAM ロール名です。作成する名前と置き換える必要があります。

注：

この例では、curly コマンドを使用して、上記の `curl` にアクセスします。この例では、URL にアクセスするために curl コマンドを実行します。 Windows ECS インスタンスを使用している場合は、ECS ユーザーガイドの「インスタンスのメタデータの使用」を参照して、STS 資格情報を取得してください。

リターンパラメーターは以下のとおりです。

```
[ root @ local ~]# curl http :// 100 . 100 . 100 . 200 / latest
 / meta - data / ram / security - credential s / EcsRamRole Test
{
" AccessKeyI d ":" XXXXXXXXX ",
" AccessKeyS ecret ":" XXXXXXXXX ",
" Expiration ":" 2017 - 06 - 09T09 : 17 : 19Z ",
" SecurityTo ken ":" CAIXXXXXXX XXXXwmBkle CTkyI +",
" LastUpdate d ":" 2017 - 10 - 31T23 : 20 : 01Z ",
" Code ":" Success "
}
```

手順 4: Python SDK を使用して STS 資格情報を使って OSS にアクセス

この例では、STS 資格情報を使用して、インスタンスと同じリージョンにある OSS バケット内の 10 個のファイルをリストアップするために Python を使用します。

前提条件

ECS インスタンスにリモート接続しています。

Python が ECS インスタンスにインストールされています。 Linux ECS インスタンスを使用している場合は、pip をインストールする必要があります。

インスタンスのリージョンにバケットが作成され、バケット名とエンドポイントが取得されています。この例では、バケット名は `ramroletes t`、エンドポイントは `oss - cn - hangzhou . aliyuncs . com` です。

手順

Python を使用して OSS バケットにアクセスするには、次の手順に従います。

1. コマンド `pip install oss2` を実行して、OSS Python SDK をインストールします。
2. 次のコマンドを実行してテストします。
   - `The three parameters in oss2 . StsAuth` は、上記の URL が返す AccessKeyId、AccessKeySecret、SecurityToken にそれぞれ対応します。
   - `The last two parameters in oss2 . Bucket` は、bucketcodeph 名とエンドポイントです。

```
import oss2
from itertools import islice
auth = oss2 . StsAuth (< AccessKeyI d >, < AccessKeyS ecret
>, < SecurityTo ken >)
bucket = oss2 . bucket = oss2 . Bucket ( auth , < your
Endpoint >, < your Bucket name >)
for b in islice ( oss2 . ObjectIter ator ( bucket ), 10
):
    print ( b . key )
```

出力結果は以下のとおりです。

```
[ root @ local ~]# python
Python 2 . 7 . 5 ( default , Nov 6 2016 , 00 : 28 : 07 )
[ GCC 4 . 8 . 5 20150623 ( Red Hat 4 . 8 . 5 - 11 )] on
linux2
Type " help "," copyright "," credits " or " license " for more
informatio n .
>>> import oss2
>>> from itertools import islice
>>> auth = oss2 . StsAuth (" STS . J8XXXXXXXX XX4 ","
9PjfXXXXXX XXXBf2XAW "," CAIXXXXXXX XXXXwmBkle CTkyI +")
>>> bucket = oss2 . Bucket ( auth ," oss - cn - hangzhou .
aliyuncs . com "," ramroletes t ")
>>> for b in islice ( oss2 . ObjectIter ator ( bucket ),
10 ):
...     print ( b . key )
...
ramroletes t . txt
test . sh
```

7 GPU インスタンス

7.1 gn5 インスタンスへの NGC のデプロイ

NVIDIA のディープラーニングエコシステムとして、NGC (NVIDIA GPU CLOUD) を使用すると、開発者はディープラーニングソフトウェアスタックに無料でアクセスでき、ディープラーニング開発環境を作るのに適しています。

現在、NGC は gn5 インスタンスに完全にデプロイされています。 さらに、イメージマーケットは、NVIDIA Pascal GPU 用に最適化された NGC コンテナー画像も提供しています。 イメージマーケットから NGC コンテナー画像をデプロイすることで、開発者は NGC コンテナー環境を手軽に構築することができ、最適化されたディープラーニングフレームワークに即座にアクセスできるため、製品開発および業務デプロイ時間が大幅に短縮されます。 その他の利点として、開発環境の事前インストール、最適化されたアルゴリズムフレームワークのサポート、継続的な更新などがあります。

NGC Web サイトは、現在の主流のディープラーニングフレームワーク (Caffe、Caffe2、CNTK、MxNet、TensorFlow、Theano、Torch など) のさまざまなバージョンのイメージを提供します。 環境を構築するために希望のイメージを選択することができます。 例として TensorFlow ディープラーニングフレームワークを挙げ、 gn5 インスタンス上で NGC 環境を構築する方法について説明します。

TensorFlow 環境を構築する前に、以下を行う必要があります。

- Alibaba Cloud にサインアップして、本名の登録を終了します。
- NGC Web サイトにログインし、NGC アカウントを作成します。
- NGC Web サイトにログインし、NGC API キーを取得してローカルに保存します。 NGC コンテナー環境にログインすると、NGC API キーが検証されます。

手順

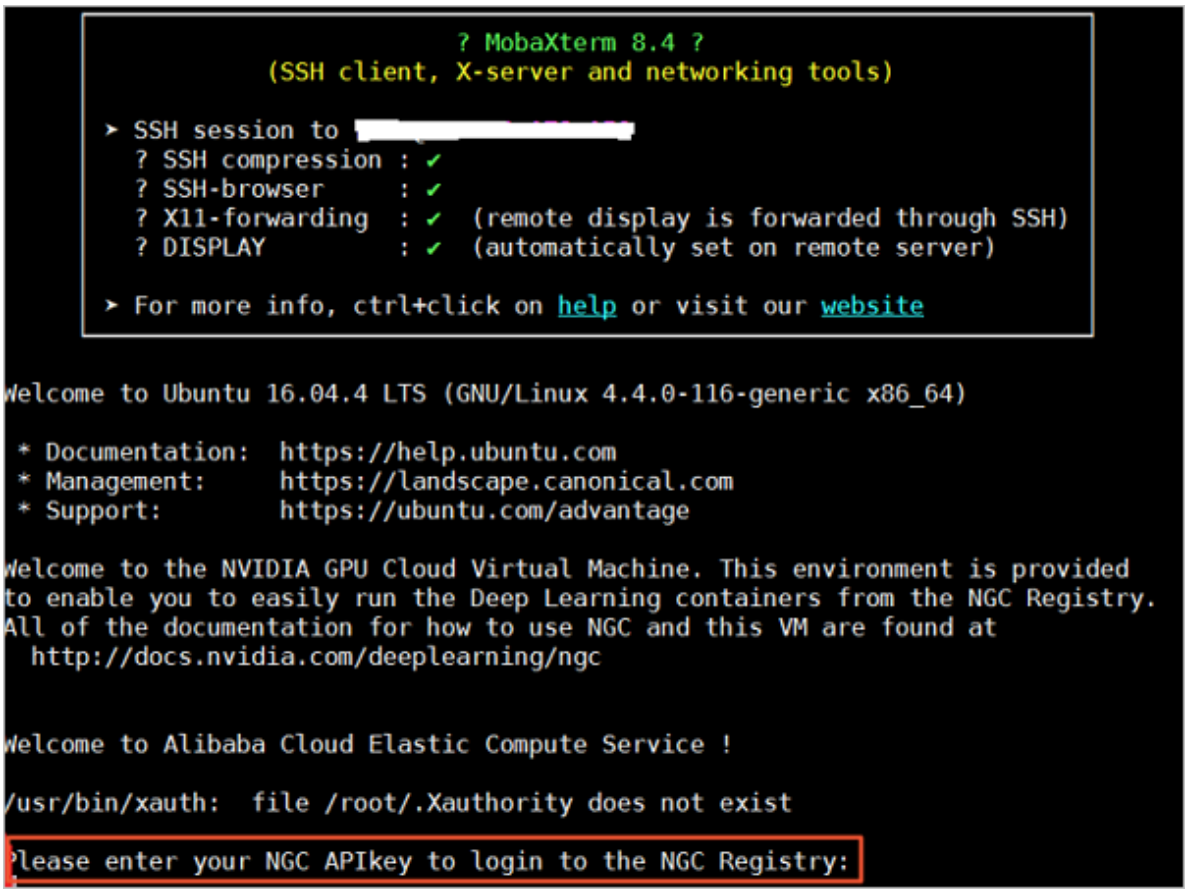
1. 「ECS インスタンスの作成」を参照して、gn5 インスタンスを作成します。 以下の設定に注意します。

   - リージョン: 中国 (青島)、中国 (北京)、中国 (フフホト)、中国 (杭州)、中国 (上海)、中国 (深セン)、中国 (香港)、シンガポール、オーストラリア (シドニー)、米国 (シリコンバレー)、米国 (バージニア)、ドイツ (フランクフルト) のみが利用できます。
   - インスタンス: gn5 インスタンスタイプを選択します。
   - イメージ: [イメージマーケット] を選択します。 表示されたダイアログボックスで、[NVIDIA GPU Cloud VM イメージ] を検索し、[続行] をクリックします。
   - ネットワーク請求方法: [パブリック IP の割り当て] を選択します。

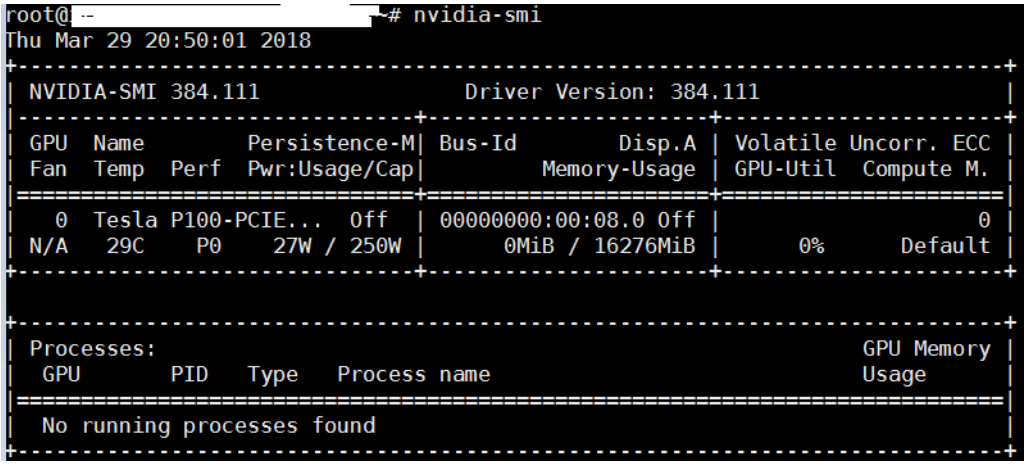
     
     注 :
     ここでパブリックIPアドレスを割り当てない場合は、インスタンスが正常に作成された後に EIP アドレスをバインドできます。
   - セキュリティグループ: セキュリティグループを選択します。 セキュリティグループで、TCP ポート 22 へのアクセスを許可する必要があります。 インスタンスが HTTPS または DIGITS 6 をサポートする必要がある場合、TCP ポート 443 (HTTPS 用) または TCP ポート5000 (DIGITS 6 用) へのアクセスを許可する必要があります。

   ECS インスタンスが正常に作成された後、ECS コンソールにログインし、インスタンスのパブリック IP アドレスを書き留めます。
2. ECS インスタンスに接続します。インスタンス作成中に選択したログオン認証情報に基づいて、パスワードを使用して ECS インスタンスに接続するか、または SSH キーペアを使用して ECS インスタンスに接続することができます。

3. NGC Web サイトから取得した NGC API キーを入力し、Enter キーを押して NGC コンテナー環境にログインします。

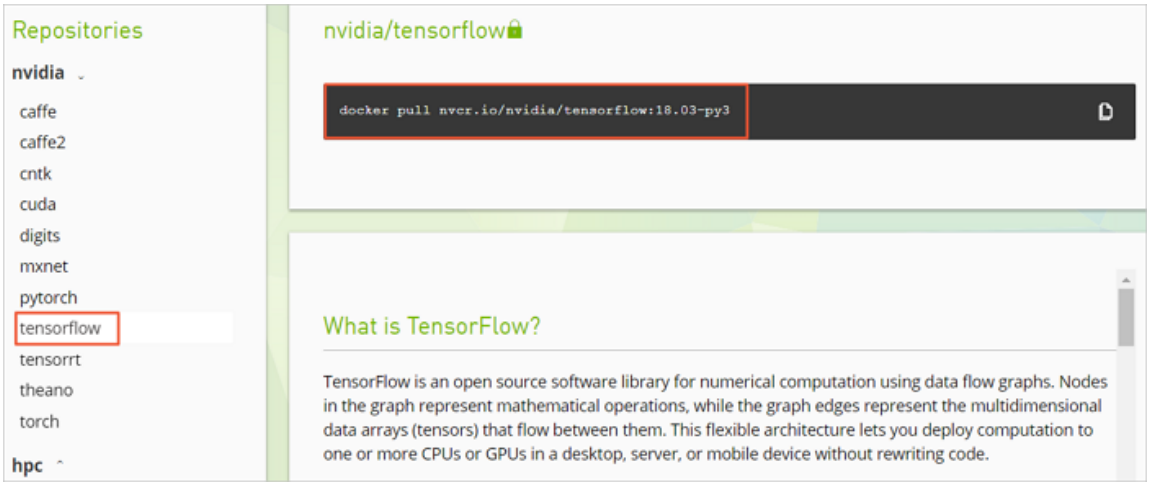


4. `nvidia - smi` を実行します。以下に示すように、GPU モデル、ドライバーのバージョンなど、現在の GPU に関する情報を表示できます。



5. 以下の手順に従って TensorFlow 環境を構築します。

   a. NGC Web サイトにログインし、TensorFlow イメージページに移動し、`docker pull` コマンドを取得します。



   b. TensorFlow イメージをダウンロードします。

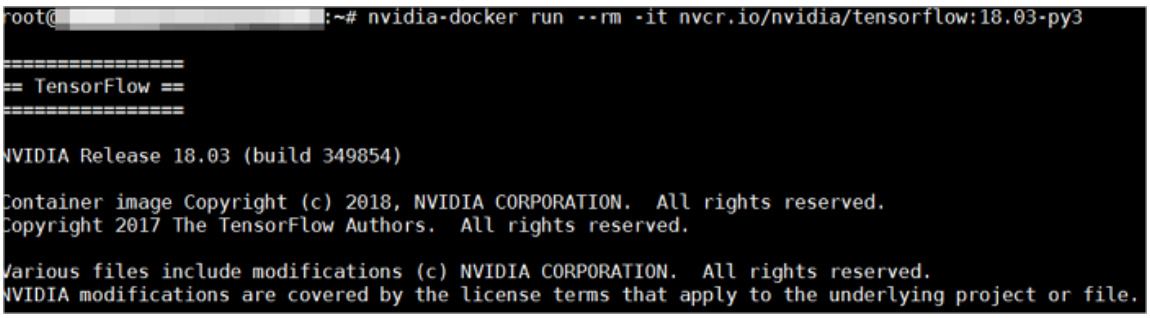
```
docker pull nvcr.io/nvidia/tensorflow:18.03-py3
```

   c. ダウンロードしたイメージを表示します。

```
docker image ls
```

   d. コンテナーを実行して TensorFlow 開発環境をデプロイします。

```
nvidia-docker run --rm -it nvcr.io/nvidia/tensorflow:18.03-py3
```



6. 以下のいずれかの方法を使用して TensorFlow をテストします。

   - TensorFlow の簡易テスト。

```
$ python
```

```
>>> import tensorflow as tf
```

```
 >>>  hello  =  tf . constant (' Hello ,  TensorFlow !')
>>>  sess  =  tf . Session ()
>>>  sess . run ( hello )
```

TensorFlow が GPU デバイスを正しくロードすると、結果は以下のようになります。

```
root@                    # python
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
>>> hello = tf.constant('Hello, TensorFlow!')
>>> sess = tf.Session()
2018-03-30 03:37:53.682157: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:892] s
be at least one NUMA node, so returning NUMA node zero
2018-03-30 03:37:53.682544: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1030] Foun
name: Tesla P100-PCIE-16GB major: 6 minor: 0 memoryClockRate(GHz): 1.3285
pciBusID: 0000:00:08.0
totalMemory: 15.89GiB freeMemory: 15.60GiB
2018-03-30 03:37:53.682583: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1120] Crea
16GB, pci bus id: 0000:00:08.0, compute capability: 6.0)
>>> sess.run(hello)
b'Hello, TensorFlow!'
>>>
```

- TensorFlow モデルをダウンロードして TensorFlow をテストします。

```
git   clone   https :// github . com / tensorflow / models . git
cd   models / tutorials / image / alexnet
python   alexnet_be  nchmark . py  -- batch_size   128  --
num_batche  s   100
```

実行状態は以下のとおりです。

```
conv1   [128, 56, 56, 64]
pool1   [128, 27, 27, 64]
conv2   [128, 27, 27, 192]
pool2   [128, 13, 13, 192]
conv3   [128, 13, 13, 384]
conv4   [128, 13, 13, 256]
conv5   [128, 13, 13, 256]
pool5   [128, 6, 6, 256]
2018-03-30 03:40:13.357785: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:892] successful NUMA node read from SysFS
be at least one NUMA node, so returning NUMA node zero
2018-03-30 03:40:13.358207: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1030] Found device 0 with properties:
name: Tesla P100-PCIE-16GB major: 6 minor: 0 memoryClockRate(GHz): 1.3285
pciBusID: 0000:00:08.0
totalMemory: 15.89GiB freeMemory: 15.60GiB
2018-03-30 03:40:13.358245: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1120] Creating TensorFlow device (/device:GPU:
16GB, pci bus id: 0000:00:08.0, compute capability: 6.0)
2018-03-30 03:40:15.916471: step 0, duration = 0.038
2018-03-30 03:40:16.299169: step 10, duration = 0.038
2018-03-30 03:40:16.682881: step 20, duration = 0.038
2018-03-30 03:40:17.065379: step 30, duration = 0.038
2018-03-30 03:40:17.448118: step 40, duration = 0.038
2018-03-30 03:40:17.830372: step 50, duration = 0.038
2018-03-30 03:40:18.213018: step 60, duration = 0.038
2018-03-30 03:40:18.595734: step 70, duration = 0.038
2018-03-30 03:40:18.978311: step 80, duration = 0.038
2018-03-30 03:40:19.361063: step 90, duration = 0.038
2018-03-30 03:40:19.705396: Forward across 100 steps, 0.038 +/- 0.000 sec / batch
2018-03-30 03:40:21.164735: step 0, duration = 0.090
2018-03-30 03:40:22.062778: step 10, duration = 0.090
2018-03-30 03:40:22.962202: step 20, duration = 0.090
2018-03-30 03:40:23.860856: step 30, duration = 0.090
2018-03-30 03:40:24.758891: step 40, duration = 0.090
2018-03-30 03:40:25.657170: step 50, duration = 0.090
2018-03-30 03:40:26.555194: step 60, duration = 0.090
2018-03-30 03:40:27.452843: step 70, duration = 0.090
2018-03-30 03:40:28.351092: step 80, duration = 0.090
2018-03-30 03:40:29.249606: step 90, duration = 0.090
2018-03-30 03:40:30.058089: Forward-backward across 100 steps, 0.090 +/- 0.000 sec / batch
```

7. TensorFlow イメージに加えた変更を保存します。 保存しないと、次回ログイン時に設定が失われます。

8 FaaS インスタンスのベストプラクティス

8.1 f1 インスタンスでの RTL コンパイラの使用

ここでは、f1 インスタンスで RTL (Register Transfer Level) コンパイラを使用する方法について説明します。

> 注：
>
> - ここに記載されているすべての操作は、同じリージョンの 1 つのアカウントで実行する必要があります。
> - f1 インスタンスを RAM ユーザーとして使用することを強く推奨します。 不要な操作を避けるには、RAM ユーザーに必要な操作のみを実行する権限を与える必要があります。 RAM ユーザーのロールを作成し、OSS バケットにアクセスするため、そのロールに一時的な権限を付与する必要があります。 IP アドレスを暗号化する場合は、RAM ユーザーが KMS (Key Management Service) を使用できるようにします。 RAM ユーザーが権限を確認するためには、アカウントのリソースを表示する権限を RAM ユーザーに与えます。

前提条件

- f1 インスタンスを作成し、セキュリティグループのルールを追加して、インスタンスの SSH ポート 22 へのインターネットアクセスを許可します。

  > 注：
  > 共有しているイメージのみが、f1 インスタンスで使用できます。 詳しくは「f1 インスタンスの作成」をご参照ください。

- ECS コンソールにログインして、インスタンス ID を取得します。
- OSS を有効にして OSS バケットを作成し、ファイルをアップロードします。 OSS バケットと f1 インスタンスは、1 つのアカウントで所有され、同じリージョンで運用される必要があります。
- 暗号化するには、KMS (Key Management Service) を有効にします。
- FPGA を RAM ユーザーとして操作するには、事前に次の手順を実行します。
  - RAM を作成して、権限を付与します。
  - RAM を作成して、権限を付与します。
  - 認証を完了するために AccessKey を使用してください。

手順

f1 インスタンスで RTL コンパイラを使用するには、次の手順を実行します。

手順 1. f1 インスタンスへの接続

f1 インスタンスに接続します。

手順 2. 基本環境の設定

スクリプトを実行して、基本環境を設定します。

```
source / opt / dcp1_1 / script / f1_env_set . sh
```

手順 3. プロジェクトのコンパイル

次のコマンドを実行して、プロジェクトをコンパイルします。

```
cd / opt / dcp1_1 / hw / samples / dma_afu
afu_synth_ setup -- source hw / rtl / filelist . txt
build_synt h
cd build_synt h /
run . sh
```

注：

プロジェクトのコンパイルには長い時間がかかります。

手順 4. イメージの作成

イメージを作成するには、次の手順を実行します。

1. 次のコマンドを実行して `faascmd` を初期化します。

```
# If needed , add the environmen t variable and grant
  permission to run the commands .
 export PATH =$ PATH :/ opt / dcp1_1 / script /
 chmod + x / opt / dcp1_1 / script / faascmd
# Replace hereIsMySe cretId with your AccessKey ID .
 Replace hereIsMySe cretKey with your AccessKey Secret .
 faascmd config -- id = hereIsMySe cretId -- key = hereIsMySe
 cretKey
 faascmd config -- id = hereIsYour SecretId -- key =
 hereIsYour SecretKey
# Replace hereIsYour Bucket with the OSS bucket name
 in the China East 1 region .
```

```
faascmd auth -- bucket = hereIsYour Bucket
```

2. "*/ opt / dcp1_0 / hw / samples / dma_afu* " ディレクトリに入っていることを確認し、コマンドを実行して gbs ファイルをアップロードします。

```
faascmd upload_obj ect -- object = dma_afu . gbs -- file =
dma_afu . gbs
```

3. コマンドを実行してイメージを作成します。

```
# Replace hereIsYour ImageName with your image name .
faascmd create_ima ge -- object = dma_afu . gbs -- fpgatype
= intel -- name = hereIsYour ImageName -- tags = hereIsYour
ImageTag -- encrypted = false -- shell = V1 . 1
```

手順 5. イメージのダウンロード

イメージをダウンロードするには、次の手順を実行します。

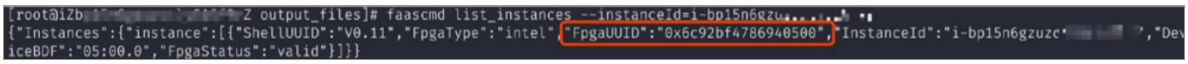
1. `faascmd list_image s` コマンドを実行して、イメージが作成されたかどうかを確認します。

   返された結果に `" State ":" success "` がある場合、イメージが作成されたことを意味します。 FpgaImageUUID を記録します。 FpgaImageUUID を記録します。

2. コマンドを実行して、FPGA ID を取得します。

```
# Replace hereIsYour InstanceId with your f1 instance
ID .
faascmd list_insta nces -- instanceId = hereIsYour InstanceId
```

   返された結果に FpgaUUID を記録します。



3. コマンドを実行して、イメージを f1 インスタンスにダウンロードします。

```
# hereIsYour InstanceID を f1 インスタンス ID に置き換えます。
Replace hereIsFpga UUID with your FpgaUUID . Replace
hereIsImag eUUID with your FpgaImageU UID .
faascmd download_i mage -- instanceId = hereIsYour InstanceID
-- fpgauuid = hereIsFpga UUID -- fpgatype = intel -- imageuuid =
hereIsImag eUUID -- imagetype = afu -- shell = V0 . 11
```

4. コマンドを実行して、イメージがダウンロードされているかどうかを確認します。

```
# Replace hereIsYour InstanceID with your f1 instance
ID . Replace hereIsFpga UUID with your FpgaUUID .
```

```
faascmd  fpga_statu  s  -- instanceId = hereIsYour  InstanceID  --
fpgauuid = hereIsFpga  UUID
```

返された結果に " TaskStatus ":" operating " があり、表示された FpgaImageUUID が、記録された FpgaImageUUID と同じ場合、イメージはダウンロードされています。

手順 6. テスト

テストを行うためにコマンドを 1 つずつ実行します。

```
cd  / opt / dcp1_1 / hw / samples / dma_afu / sw
make
sudo  LD_LIBRARY  _PATH =/ opt / dcp1_1 / hw / samples / dma_afu /
sw :$ LD_LIBRARY  _PATH  ./ fpga_dma_t  est  0
```

次の結果が返された場合、テストは完了です。

```
[root@iZ Z sw]# ./fpga_dma_test use_ase=0
Running test in HW mode
Buffer Verification Success!
Buffer Verification Success!
Running DDR sweep test
Allocated test buffer
Fill test buffer
DDR Sweep Host to FPGA
Measured bandwidth = 5726.623061 Megabytes/sec
Clear buffer
DDR Sweep FPGA to Host
Measured bandwidth = 4473.924267 Megabytes/sec
Verifying buffer..
Buffer Verification Success!
```

注：

大量ページ機能が有効になっていない場合は、次のコマンドを実行して有効にします。

```
sudo bash -c "echo 20 > /sys/kernel/mm/hugepages/
hugepages-2048kB/nr_hugepages"
```

8.2 f1 インスタンスでの OpenCL の使用

ここでは、OpenCL (Open Computing Language) を使用してイメージファイルを作成し、FPGA チップにイメージをダウンロードする方法を紹介します。

注:

- ここに記載されているすべての操作は、同じリージョンの 1 つのアカウントで実行する必要があります。
- f1 インスタンスを RAM ユーザーとして使用することを強く推奨します。 不要な操作を避けるには、RAM ユーザーに必要な操作のみを実行する権限を与える必要があります。 RAM ユーザーのロールを作成し、OSS バケットにアクセスするため、ロールに一時的な権限を付与する必要があります。 IP アドレスを暗号化する場合は、RAM ユーザーが KMS (Key Management Service) を使用できるようにします。 RAMユーザーが権限を確認するには、アカウントのリソースを表示する権限を与えます。 始める前に、以下を完了してください。

前提条件

- f1 インスタンスを作成し、セキュリティグループのルールを追加して、インスタンスの SSH ポート 22 へのインターネットアクセスを許可します。

  注:
  共有しているイメージのみが、f1 インスタンスで使用できます。 詳しくは「f1 インスタンスの作成」をご参照ください。
- ECS コンソールにログインして、インスタンス ID を取得します。
- OSS バケットを作成して、カスタムビットストリームファイルをアップロードします。 OSS バケットと f1 インスタンスは、同じリージョンの 1 つのアカウントで所有している必要があります。
- ビットストリームを暗号化するには、KMS (Key Management Service) を有効にします。

- f1 インスタンスを RAM ユーザーとして操作するには、次の操作を実行する必要があります。
  - RAM ユーザーを作成し、許可を付与します。
  - RAM ロールを作成し、許可を付与します。
  - AccessKey を作成します。

手順

FPGA Server Example の環境を設定するには、次の手順を実行します。

手順 1. f1 インスタンスへの接続

Linux インスタンスに接続します。

手順 2. 基本環境のインストール

次のスクリプトを実行して、基本環境をインストールします。

```
source  / opt / dcp1_0 / script / f1_env_set . sh
```

手順 3. OpenCL サンプルのダウンロード

以下の手順に従って、公式 opencl のサンプルをダウンロードします。

1. "*/ opt / tmp* " ディレクトリを作成し、現在のディレクトリをそのディレクトリに変更します。

   ```
   mkdir  - p  / opt / tmp
   cd  / opt / tmp
   ```

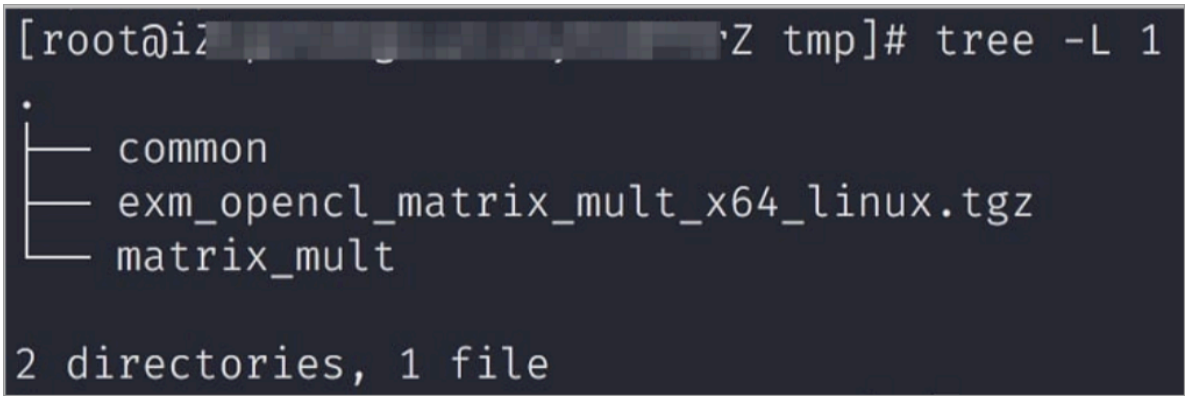
   現在、*/ opt / tmp* ディレクトリに入っています。

2. コマンドを1つずつ実行して、OpenCL Example のファイルをダウンロードして解凍します。

   ```
   wget   https :// www . altera . com / content / dam / altera -
   www / global / en_US / others / support / examples / download /
   exm_opencl  _matrix_mu  lt_x64_lin  ux . tgz
   tar  - zxvf   exm_opencl  _matrix_mu  lt_x64_lin  ux . tgz
   ```

次の図は、解凍後のディレクトリを示しています。



3. 現在のディレクトリを" *matrix_mul t* "ディレクトリに変更し、コンパイルのコマンドを実行します。

```
cd matrix_mul t
aoc - v - g -- report ./ device / matrix_mul t . cl
```

コンパイルのプロセスには数時間かかります。新しいコンソールを開き、`top` コマンドを実行すると、インスタンス上のプロセスやシステムリソースの使用状況をモニターし、コンパイルプロセスのステータスを見ることができます。

手順 4. OSS バケットへの設定ファイルのアップロード

以下の手順に従って、設定ファイルをアップロードします。

1. コマンドを実行して faascmd を初期化します。

```
# If needed , add the environmen t variable and grant
the permission to run the commands
export PATH =$ PATH :/ opt / dcp1_1 / script /
chmod + x / opt / dcp1_1 / script / faascmd
# Replace hereIsYour SecretId with your AccessKey ID .
Replace hereIsYour SecretKey with your AccessKey Secret
faascmd config -- id = hereIsYour SecretId -- key =
hereIsYour SecretKey
# Replace hereIsYour Bucket with the bucket name of
your OSS in the Region China East 1 .
faascmd auth -- bucket = hereIsYour Bucket
```

2. 現在のディレクトリを" *matrix_mul t / output_fil es* "ディレクトリに変更し、設定ファイルをアップロードします。

```
cd matrix_mul t / output_fil es # Now you are accessing
/ opt / tmp / matrix_mul t / matrix_mul t / output_fil es
faascmd upload_obj ect -- object = afu_fit . gbs -- file =
afu_fit . gbs
```

3. gbs を使用して FPGA イメージを作成します。

```
# Replace hereIsYour ImageName with your image name .
Replace hereIsYour ImageTag with your image tag .
faascmd create_ima ge -- object = dma_afu . gbs -- fpgatype
= intel -- name = hereIsYour ImageName -- tags = hereIsYour
ImageTag -- encrypted = false -- shell = V1 . 1
```

4. `faascmd list_image s` コマンドを実行して、イメージが作成されたかどうかを確認します。返された結果で `" State ":" success "` が表示されている場合は、イメージが作成されたことを意味します。FpgaImageUUID を記録します。

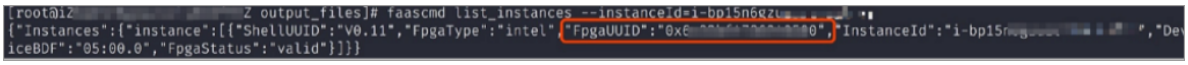
手順 5. f1 インスタンスへのイメージのダウンロード

f1 インスタンスにイメージをダウンロードするには、次の手順を実行します。

1. コマンドを実行して、FPGA ID を取得します。

```
# Replace hereIsYour InstanceId with your f1 instance
ID .
faascmd list_insta nces -- instanceId = hereIsYour InstanceId
```

返された結果のサンプル: 返された結果に FpgaUUID を記録します。



2. コマンドを実行して、イメージを f1 インスタンスにダウンロードします。

```
# Replace hereIsYour InstanceID with your f1 instance
ID . Replace hereIsFpga UUID with your FPGA UUID .
Replace hereIsImag eUUID with your image UUID .
faascmd download_i mage -- instanceId = hereIsYour InstanceID
-- fpgauuid = hereIsFpga UUID -- fpgatype = intel -- imageuuid
= hereIsImag eUUID -- imagetype = afu -- shell = V0 . 11
```

3. コマンドを実行して、イメージがダウンロードされているかどうかを確認します。

```
# Replace hereIsYour InstanceID with your f1 instance
ID . Replace hereIsFpga UUID with your FPGA UUID .
faascmd fpga_statu s -- fpgauuid = hereIsFpga UUID --
instanceId = hereIsYour InstanceID
```

返された結果に "TaskStatus": "operating" が存在する場合、イメージがダウンロードされたことを意味します。

手順 6. FPGA チップへの FPGA イメージのダウンロード

FPGA イメージを FPGA チップにダウンロードするには、次の手順を実行します。

1. 手順 1 でコンソールを開きます。 閉じている場合、手順 1 を繰り返します。
2. OpenCL のランタイム環境を設定するには、次のコマンドを実行します。

```
sh  / opt / dcp1_1 / opencl / opencl_bsp / linux64 / libexec /
setup_perm  issions . sh
```

3. コマンドを実行して親ディレクトリに戻ります。

```
cd  .. /.. #  Now ,  you  are  at  the  / opt / tmp /
matrix_mul  t  directory
```

4. コマンドを実行してコンパイルします。

```
make
#  Output  the  environmen  t  configurat  ion
export  CL_CONTEXT  _COMPILER_  MODE_ALTER  A = 3
cp  matrix_mul  t . aocx  ./ bin / matrix_mul  t . aocx
cd  bin
host  matrix_mul  t . aocx
```

次の結果が返された場合は、設定が成功したことを意味します。 最後の行は `Verificati on : PASS` でなければならないことにご注意ください。

```
[ root @ iZbpXXXXXZ  bin ]# ./ host  matrix_mul  t . aocx
Matrix  sizes :
  A :  2048  x  1024
  B :  1024  x  1024
  C :  2048  x  1024
Initializi  ng  OpenCL
Platform :  Intel ( R )  FPGA  SDK  for  OpenCL ( TM )
Using  1  device ( s )
  skx_fpga_d  cp_ddr  :  SKX  DCP  FPGA  OpenCL  BSP  ( acl0 )
Using  AOCX :  matrix_mul  t . aocx
Generating  input  matrices
Launching  for  device  0  ( global  size :  1024 ,  2048 )
Time :  40 . 415  ms
Kernel  time  ( device  0 ):  40 . 355  ms
Throughput :  106 . 27  GFLOPS
Computing  reference  output
```

```
Verifying
Verificati  on :  PASS
```

8.3 f3 インスタンスでの OpenCL のベストプラクティス

本ページでは、OpenCL (Open Computing Language) を使用してイメージを作成し、f3 インスタンスの FPGA チップにイメージをダウンロードする方法を紹介します。

注 :

- 本ページに記載されているすべての操作は、同じリージョンの 1 つのアカウントで実行する必要があります。
- f3 インスタンスを RAM ユーザーとして使用することを推奨します。 RAM ユーザーのロールを作成し、OSS バケットにアクセスするため、ロールに一時的な権限を付与する必要があります。

前提条件

- f3 インスタンスの作成します。

  注 :

  - 共有しているイメージのみ、 f3 インスタンスで使用できます。
  - インスタンスをインターネットにアクセスできるように、インスタンスの作成時に [パブリック IP の割り当て] を選択します。
  - f3 インスタンスのセキュリティグループは、ルールを追加して、SSH ポート 22 へのアクセスを許可しました。
- ECS コンソールにログインして、f3 インスタンスのインスタンス ID を取得します。
- 同じアカウントを使用して、f3 インスタンスと同じリージョンに OSS バケットを作成します。 詳しくは「OSS への登録」および「バケットの作成」をご参照ください。
- FPGA を RAM ユーザーとして操作するには、事前に次の手順を実行します。
  - RAM ユーザーを作成し、許可を与えます。
  - RAM ロールを作成し、許可を与えます。
  - AccessKey ID と AccessKey Secret を取得します。

手順

f3 インスタンスで OpenCL を使用してイメージを作成し、FPGA チップにダウンロードするには、次の手順を実行します。

手順 1. 環境設定

環境を設定するには、次の手順を実行します。

1. f3 インスタンスに接続します。

   注：

   その後のコンパイル処理には数時間かかる場合があります。 SSH タイムアウトによって強制的にログアウトされないようにするために、screen または nohub を介してログインすることを推奨します。

2. コマンドを実行して Screen をインストールします。

   ```
   yum   install   screen  - y
   ```

3. コマンドを実行して Screen に入ります。

   ```
   screen  - S   f3opencl
   ```

4. コマンドを実行して、環境を設定します。

   ```
   source  / root / xbinst_oem / f3_env_set  up . sh   xocl   #  Run
     the   command   each   time   you   open   a   new   terminal
   window
   ```

   注：

   - 環境の設定には、xocl ドライバーのインストール、vivado 環境変数の設定、vivado ライセンスの確認、aliyun-f3 sdaccel プラットフォームの検出、2018.2 ランタイムの設定、および faascmd バージョンの検出が含まれます。
   - sdaccel のエミュレーションを実行する場合は、上記のコマンドを実行して環境を設定しないでください。 代わりに、vivado の環境変数を個別に設定するだけで済みます。
   - エミュレーションには Makefile を使用することを推奨します。

手順 2. バイナリファイルのコンパイル

- **例 1: vadd**

  バイナリファイルをコンパイルするには、次の手順を実行します。

  1. **"** `example` **"ディレクトリをコピーします。**

     ```
     cp  - rf  / opt / Xilinx / SDx / 2018 . 2 / examples  . /
     ```

  2. **"** `vadd` **" ディレクトリを入力します。**

     ```
     cd  examples / vadd /
     ```

  3. `cat  sdaccel . mk  |  grep  " XDEVICE ="` **を実行して、** `XDEVICE` **の値を表示します。 その設定が** `XDEVICE  =  xilinx_ali  yun - f3_dynamic  _5_0` **であることを確認してください。**

  4. **次の手順に従って、"** `common . mk` **" ファイルを変更します。**

     a. `vim  ../ common / common . mk` **コマンドを実行してファイルを開きます。**

     b. **コードの 61 行目の末尾に、コンパイルパラメーター "** `-- xp  param : compiler . accelerato  rBinaryCon  tent = dcp` **" を追加します (パラメーターは、ファイルによっては 60 ~ 62 行になる場合があります)。 修正されたコードは次のとおりです。**

     ```
     CLCC_OPT  += $( CLCC_OPT_L  EVEL ) ${ DEVICE_REP  O_OPT }
     -- platform  ${ XDEVICE } - o  ${ XCLBIN } ${ KERNEL_DEF  S
     } ${ KERNEL_INC  S } -- xp  param : compiler . accelerato
     rBinaryCon  tent = dcp
     ```

     注 :

     DCP ファイルをコンパイルサーバに送信する必要があることを考えると、Xilinx® OpenCL™ Compiler (xocc) により、配置とルーティングが完了した後に (ビットファイルではなく) DCPファイルが生成されるように、パラメーター "`-- xp`

`param : compiler . accelerato rBinaryCon tent = dcp "` を追加する必要があります。

5. コマンドを実行してプログラムをコンパイルします。

```
make - f sdaccel . mk xbin_hw
```

次の情報が表示された場合、バイナリファイルのコンパイルは開始されています。 処理には数時間かかる場合があります。

- 例 2: kernel_global_bandwidth

次の手順に従って、"kernel_global_bandwidth" バイナリファイルをコンパイルします。

1. `xilinx 2018 . 2 example` のクローンを作成します。

```
git clone https :// github . com / Xilinx / SDAccel_Ex
amples . git
cd SDAccel_Ex amples /
git checkout 2018 . 2
```

> 注:
> git ブランチは 2018.2 バージョンである必要があります。

2. `cd getting_st arted / kernel_to_ gmem / kernel_glo bal_bandwi dth /` コマンドを実行して、ディレクトリに入ります。
3. 次の手順に従って、`" Makefile "` ファイルを変更します。
   a. `vim Makefile` コマンドを実行して、ファイルを開きます。
   b. `DEVICES = xilinx_ali yun - f3_dynamic _5_0` を設定します。
   c. コードの 33 行目に、コンパイルパラメーター `"-- xp param : compiler . accelerato rBinaryCon tent = dcp "` を追加します。 修正されたコードは次のとおりです。

```
CLFLAGS +=-- xp " param : compiler . accelerato rBinaryCon
tent = dcp " -- xp " param : compiler . preserveHl sOutput
= 1 " -- xp " param : compiler . generateEx traRunData =
```

```
true " -- max_memory _ports bandwidth - DNDDR_BANK S =$(
ddr_banks )
```

4. コマンドを実行してプログラムをコンパイルします。

```
make TARGET = hw
```

次の情報が表示された場合、バイナリファイルのコンパイルは開始されています。 処理には数時間かかる場合があります。

手順 3. パッケージングスクリプトの確認

コマンドを実行して、パッケージングスクリプトが存在するかどうかを確認します。

```
file / root / xbinst_oem / sdaccel_pa ckage . sh
```

返されたメッセージに `cannot open ( No such file or directory )` が含まれる場合、ファイルは存在しません。 次のコマンドを実行してスクリプトをダウンロードします。

```
wget http :// fpga - tools . oss - cn - shanghai . aliyuncs . com /
sdaccel_pa ckage . sh
```

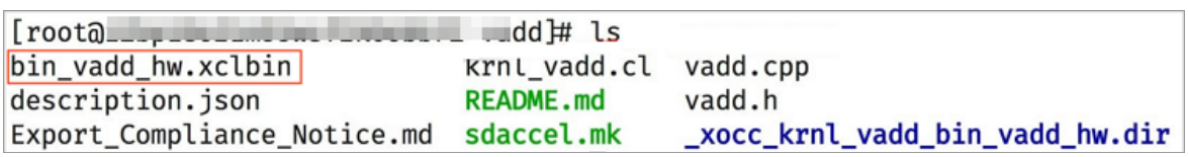
手順 4. イメージの作成

イメージを作成するには、次の手順を実行します。

1. コマンドを実行して OSS 環境を設定します。

```
faascmd config -- id = hereIsMySe cretId -- key = hereIsMySe
cretKey # Replace hereIsMySe cretId , hereIsMySe cretKey
with your AccessKeyI D , AccessKeyS ecret
faascmd auth -- bucket = hereIsMyBu cket # Replace
hereIsMyBu cket with your bucket name
```

2. `ls` コマンドを実行して、接尾辞が `. xclbin` のファイルを取得します。



3. コマンドを実行してバイナリファイルをパッケージ化します。

```
/ root / xbinst_oem / sdaccel_pa  ckage . sh  - xclbin =/ opt /
 Xilinx / SDx / 2017 . 4 . op / examples / vadd / bin_vadd_h  w .
 xclbin
```

パッケージ化が完了すると、次の図に示すように、同じディレクトリにパッケージファイルが見つかります。

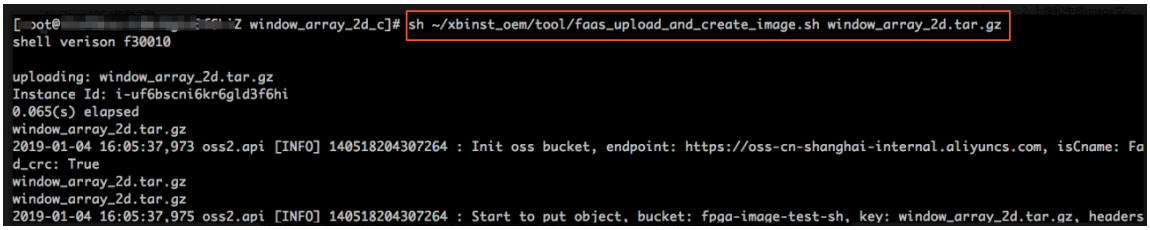
手順 5. イメージのダウンロード

スクリプト化されたプロセスまたは段階的プロセスを使用してパッケージファイルをアップロードし、FPGA イメージをダウンロードできます。

・ スクリプトプロセス: 1 つの FPGA チップを持つ f3 インスタンスのみに適用。

1. 以下のコマンドを実行してパッケージをアップロードし、イメージファイルを生成します。

```
sh  / root / xbinst_oem / tool / faas_uploa  d_and_crea  te_image
. sh  < bit . tar . gz  -  the  package  to  upload >
```



2. イメージファイルをダウンロードします。

```
sh  / root / xbinst_oem / tool / faas_downl  oad_image . sh  <
bit . tar . gz  -  package  name > < 0 / 1 > #  The  last
```

```
number < 0 / 1 > stands for the FPGA serial No . in
  the instance
```

0 は f3 インスタンスの最初の FPGA を示します。 シングル FPGA インスタンスの場合、FPGA シリアル番号は常に 0 です。 4 つの FPGA を持つインスタンスなど、複数の FPGA を持つインスタンスの場合、シリアル番号は 0、1、2、3 です。

同じイメージを複数の FPGA にダウンロードするには、末尾にシリアル番号を追加します。 たとえば、同じイメージを 4 つの FPGA チップにダウンロードするには、次のようなコマンドを実行します。

```
sh / root / xbinst_oem / tool / faas_downl oad_image . sh <
bit . tar . gz - package name > 0
sh / root / xbinst_oem / tool / faas_downl oad_image . sh <
bit . tar . gz - package name > 1
sh / root / xbinst_oem / tool / faas_downl oad_image . sh <
bit . tar . gz - package name > 2
sh / root / xbinst_oem / tool / faas_downl oad_image . sh <
bit . tar . gz - package name > 3
```

- 段階的プロセス: faascmd ツールを使用して、操作を行います。

  1. コマンドを実行して、パッケージを自身の OSS バケットにアップロードします。 次に、自身の OSS バケットの gbs を、FaaS 管理ユニットの OSS バケットにアップロードします。

```
faascmd upload_obj ect -- object = bit . tar . gz -- file =
bit . tar . gz
faascmd create_ima ge -- object = bit . tar . gz --
fpgatype = xilinx -- name = hereIsFPGA ImageName -- tags
= hereIsFPGA ImageTag -- encrypted = false -- shell =
hereIsShel lVersionOf FPGA
```

```
[root@iZ                    Z ~]# faascmd upload_object --object=rion.zj_test_SDAccel_Kernel.tar.gz --file=18_05_03-222718_SDAccel_Kernel.tar
.gz
rion.zj_test_SDAccel_Kernel.tar.gz
18_05_03-222718_SDAccel_Kernel.tar.gz
4.735(s) elapsed
```

```
[root@iZ                    Z ~]# faascmd create_image --object=rion.zj_test_SDAccel_Kernel.tar.gz --fpgatype=xilinx --name=rion.zj_xilinx_f3
_test --tags=hereIsFPGAImageTag --encrypted=false --shell=f30001
{"Name":"rion.zj_xilinx_f3_test","CreateTie":"Fri May 04 2018 20:24:21 GMT+0800 (CST)","ShellUUID":"f30001","Description":"None","FpgaImageUU
ID":"xilinx1                                5","State":"queued"}
0.221(s) elapsed
```

2. コマンドを実行して、FPGA イメージがダウンロード可能かどうかを確認します。

```
faascmd list_image s
```

返されたメッセージに `State : compiling` が表示された場合は、FPGA イメージはコンパイル中です。 返されたメッセージに `State : success` が表示されたら、FPGA イメージのダウンロード準備が整いました。 FpgaImageUUID を探して書き留めます。

```
[root@                    ~]# faascmd list_images
{
  "FpgaImages": {
    "fpgaImage": [
      {
        "CreateTime": "Fri Jan 04 2019 16:05:43 GMT+0800 (CST)",
        "Description": "None",
        "Encrypted": "false",
        "FpgaImageUUID": "xilinx8858a3c1-                                ",
        "Name": "window_array_2d.tar.gz",
        "ShellUUID": "f30010",
        "State": "compiling",
        "Tags": "hereIsFPGAImageTag",
        "UpdateTime": "Fri Jan 04 2019 16:05:44 GMT+0800 (CST)"
      },
      {
        "CreateTime": "Thu Jan 03 2019 15:58:58 GMT+0800 (CST)",
        "Description": "None",
        "Encrypted": "false",
        "FpgaImageUUID": "xilinx6cbd48c1-                                ",
        "Name": "vadd.tar.gz",
        "ShellUUID": "f30010",
        "State": "success",
        "Tags": "hereIsFPGAImageTag",
        "UpdateTime": "Thu Jan 03 2019 16:32:32 GMT+0800 (CST)"
      },
```

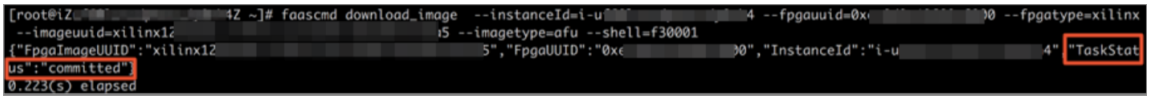
3. 次のコマンドを実行します。 返されたメッセージで、FpgaUUID を探して書き留めます。

```
faascmd list_insta nces -- instanceId = hereIsYour
InstanceId # Replace hereIsYour InstanceId with the f3
  instance ID
```

4. コマンドを実行して FPGA イメージをダウンロードします。

```
faascmd download_i mage -- instanceId = hereIsYour
InstanceId -- fpgauuid = hereIsFpga UUID -- fpgatype = xilinx
```

```
  -- imageuuid = hereIsImag  eUUID  -- imagetype = afu  -- shell =
 hereIsShel  lVersionOf  Fpga
#  Replace  hereIsYour  InstanceId  with  the  f3  instance
   ID ,  hereIsFpga  UUID  with  the  FpgaUUID ,  and
 hereIsImag  eUUID  with  the  FpgaImageU  UID
```



5. コマンドを実行して、イメージが正常にダウンロードされたかどうかを表示します。

```
faascmd  fpga_statu  s  -- fpgauuid = hereIsFpga  UUID  --
instanceId = hereIsYour  InstanceId  #  Replace  hereIsFpga
UUID  with  the  obtained  FpgaUUID ,  and  hereIsYour
InstanceId  with  the  f3  instance  ID
```

以下は返されるメッセージの例です。 メッセージ内の FpgaImageUUID と、書き留めた FpgaImageUUID が同じで、メッセージに `" TaskStatus ":" valid "` と表示されている場合、イメージは正常にダウンロードされています。

手順 6: Host プログラムの実行

Host プログラムを実行するには、次の手順を実行します。

1. 以下のコマンドを実行して環境を設定します。

```
source  / root / xbinst_oem / f3_env_set  up . sh  xocl  #  Run
  the  command  each  time  you  open  a  new  terminal
window
```

2. sdaccel.ini ファイルを設定します。

   Host バイナリファイルが置かれたディレクトリで、`vim  sdaccel . ini` コマンドを実行して、"sdaccel.ini" ファイルを作成して次の内容を入力します。

```
[ Debug ]
 profile = true
[ Runtime ]
 runtime_lo  g  = " run . log "
 hal_log  =  hal . log
 ert = false
 kds = false
```

3. Host を実行します。

   - vadd については、次のコマンドを実行します。

```
make  - f  sdaccel . mk  host
```

```
./ vadd   bin_vadd_h  w . xclbin
```

- kernel_global_bandwidth については、次のコマンドを実行します。

```
./ kernel_glo  bal
```

`Test  Passed` が返されたら、テストは成功です。

その他の一般的なコマンド

このセクションでは、FPGA インスタンスの一般的なコマンドをいくつか紹介します。

タスク	コマンド
ヘルプ文書の表示	`make  - f  ./ sdaccel . mk  help`
ソフトウェアシミュレーションの実行	`make  - f  ./ sdaccel . mk  run_cpu_em`
ハードウェアシミュレーションの実行	`make  - f  ./ sdaccel . mk  run_hw_em`
ホストコードのみのコンパイル	`make  - f  ./ sdaccel . mk  host`
ダウンロード用ファイルのコンパイルおよび生成	`make  - f  sdaccel . mk  xbin_hw`
作業ディレクトリの消去	`make  - f  sdaccel . mk  clean`
作業ディレクトリの強制消去	`make  - f  sdaccel . mk  cleanall`

注：

- エミュレーション中は、Xilinx エミュレーションプロセスに従います。 f3_env_setup 環境を設定する必要はありません。
- SDAccel ランタイムと SDAccel 開発プラットフォームは、Alibaba Cloud が提供する公式 f3 イメージで入手できます。 ダウンロードは、SDAccel ランタイムと SDAccel 開発プラットフォームで行うこともできます。

8.4 f3 インスタンスでの RTL コンパイラの使用

ここでは、f3 インスタンスで RTL (Register Transfer Level) コンパイラを使用する方法について説明します。

注：

- ここに記載されているすべての操作は、同じリージョンの 1 つのアカウントで実行する必要があります。
- f3 インスタンスを RAM ユーザーとして使用することを強く推奨します。 不要な操作を避けるために、RAM ユーザーに必要な操作のみを実行する権限を与える必要があります。 RAM ユーザーのロールを作成し、OSS バケットにアクセスするため、ロールに一時的な権限を付与する必要があります。 IP アドレスを暗号化する場合は、RAM ユーザーに KMS (Key Management Service) を使用する権限を与えるようにします。 RAM ユーザーが許可を確認するには、RAM ユーザーにアカウントのリソースを表示する権限を与えます。

前提条件

- f3 インスタンスを作成し、セキュリティグループのルールを追加して、インスタンスの SSH ポート 22 へのインターネットアクセスを許可します。
- ECS コンソールにログインして、f3 インスタンスの詳細ページでインスタンス ID を取得します。
- FaaS サービスについては、中国 (上海) に OSS バケットを作成します。

  注：
  バケットは FaaS 管理アカウントへの読み書きアクセスを提供します。 FaaS に関連しないオブジェクトを保存しないことを推奨します。
- f3 インスタンスを RAM ユーザーとして操作するには、次の手順を実行します。
  - RAM ユーザーを作成し、権限を付与します。
  - RAM ユーザーを作成し、権限を付与します。
  - AccessKey を作成します。

手順

1. f3 インスタンスに接続します。

   注：
   プロジェクトのコンパイルには 2 ～ 3 時間かかります。 予期せず切断することがないようにするため、インスタンスの接続には nohup または VNC を使用することを推奨します。
2. RTL リファレンスデザインをダウンロードします。
3. ファイルを解凍します。
4. f3 環境を設定します。

```
source /root/xbinst_oem/F3_env_setup.sh xdma
```

```
```

注：

このコマンドは、新しいターミナルウィンドウを開くたびに実行します。

5. OSS バケットを指定します。

```
faascmd  config  -- id = hereIsYour  SecretId  -- key =
hereIsYour  SecretKey  #  Replace  hereIsYour  SecretId  and
hereIsYour  SecretKey  with  your  RAM  AK  informatio  n
faascmd  auth  -- bucket = hereIsYour  Bucket  #  Replace
hereIsYour  Bucket  with  the  name  of  your  OSS  Bucket
```

6. 次のコマンドを実行して、RTL プロジェクトをコンパイルします。

```
cd  < decompress  ed  directory >/ hw / #  Enter  the
decompress  ed  hw  directory
sh  compiling . sh
```

注：

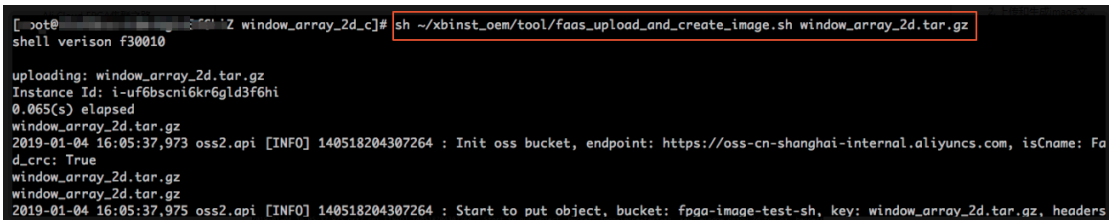
プロジェクトのコンパイルには 2 ～ 3 時間かかります。

7. "Netlist" ファイルをアップロードして FPGA イメージをダウンロードします。 スクリプト化されたプロセスまたは段階的なプロセスを使用してこのタスクを終了できます。

   - スクリプト化されたプロセス: 1 つの FPGA チップを持つ f3 インスタンスに適用可能。

     a. 以下のコマンドを実行してパッケージをアップロードし、イメージファイルを生成します。

```
sh  / root / xbinst_oem / tool / faas_uploa  d_and_crea
te_image . sh  < bit . tar . gz  -  the  package  to
upload >
```



     b. イメージファイルをダウンロードします。

```
sh  / root / xbinst_oem / tool / faas_downl  oad_image . sh
 < bit . tar . gz  -  the  package  filename >  0  #  The
```

```
last number stands for the FPGA serial No . of
  the instance
```

0 は f3 インスタンスの最初の FPGA を示します。 1 つの FPGA インスタンスの場合、FPGA シリアル番号は常に 0 です。 4つの FPGA を持つインスタンスなど、複数の FPGA を持つインスタンスの場合、シリアル番号は 0、1、2、3 です。

同じイメージを複数の FPGA にダウンロードするには、末尾にシリアル番号を追加します。

```
sh faas_downl oad_image . sh bit . tar . gz 0 1 2
```

- 段階的プロセス:
  a. 次のコマンドを実行してパッケージを OSS バケットにアップロードしてから、OSS バケットの gbs を FaaS ユニットの OSS バケットにアップロードします。

```
faascmd upload_obj ect -- object = bit . tar . gz -- file
= bit . tar . gz
faascmd create_ima ge -- object = bit . tar . gz --
fpgatype = xilinx -- name = hereIsFPGA ImageName -- tags =
hereIsFPGA ImageTag -- encrypted = false -- shell = f30001
```

```
[root@iZ                    Z ~]# faascmd upload_object --object=rion.zj_test_SDAccel_Kernel.tar.gz --file=18_05_03-222718_SDAccel_Kernel.tar
.gz
rion.zj_test_SDAccel_Kernel.tar.gz
18_05_03-222718_SDAccel_Kernel.tar.gz
4.735(s) elapsed
```

```
[root@iZ                    Z ~]# faascmd create_image --object=rion.zj_test_SDAccel_Kernel.tar.gz --fpgatype=xilinx --name=rion.zj_xilinx_f3
_test --tags=hereIsFPGAImageTag --encrypted=false --shell=f30001
{"Name":"rion.zj_xilinx_f3_test","CreateTie":"Fri May 04 2018 20:24:21 GMT+0800 (CST)","ShellUUID":"f30001","Description":"None","FpgaImageUU
ID":"xilinx1                                5","State":"queued"}
0.221(s) elapsed
```

b. 次のコマンドを実行して、FPGA イメージをダウンロードする準備ができているかどうかを確認します。

```
faascmd   list_image  s
```

返されたメッセージに " State ":" success " と表示されたら、FPGA イメージはダウンロード可能です。 FpgaImageUUID を探して書き留めます

```
[root@                    ~]# faascmd list_images
{
  "FpgaImages": {
    "fpgaImage": [
      {
        "CreateTime": "Fri Jan 04 2019 16:05:43 GMT+0800 (CST)",
        "Description": "None",
        "Encrypted": "false",
        "FpgaImageUUID": "xilinx8858a3c1-                                ",
        "Name": "window_array_2d.tar.gz",
        "ShellUUID": "f30010",
        "State": "compiling",
        "Tags": "hereIsFPGAImageTag",
        "UpdateTime": "Fri Jan 04 2019 16:05:44 GMT+0800 (CST)"
      },
      {
        "CreateTime": "Thu Jan 03 2019 15:58:58 GMT+0800 (CST)",
        "Description": "None",
        "Encrypted": "false",
        "FpgaImageUUID": "xilinx6cbd48c1-                                ",
        "Name": "vadd.tar.gz",
        "ShellUUID": "f30010",
        "State": "success",
        "Tags": "hereIsFPGAImageTag",
        "UpdateTime": "Thu Jan 03 2019 16:32:32 GMT+0800 (CST)"
      },
```

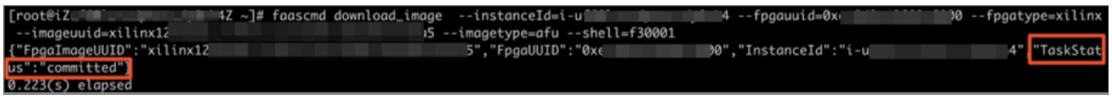
c. 次のコマンドを実行してから、返されたメッセージの FpgaUUID を書き留めます。

```
faascmd   list_insta  nces  -- instanceId = hereIsYour
InstanceId  #  Replace   hereIsYour  InstanceId   with   your
  f3   instance   ID
```

d. 次のコマンドを実行して FPGA イメージをダウンロードします。

```
faascmd   download_i  mage  -- instanceId = hereIsYour
InstanceId  -- fpgauuid = hereIsFpga  UUID  -- fpgatype =
xilinx  -- imageuuid = hereIsImag  eUUID  -- imagetype = afu
-- shell = f30001
```

```
# Replace hereIsYour InstanceId with f3 instance ID
, hereIsFpga UUID with the obtained FpgaUUID , and
hereIsImag eUUID with the obtained FpgaImageU UID
```



e. 次のコマンドを実行して、イメージが正常にダウンロードされたかどうかを確認します。

```
faascmd fpga_statu s -- fpgauuid = hereIsFpga UUID --
instanceId = hereIsYour InstanceId # Replace hereIsFpga
UUID with the obtained FpgaUUID , and hereIsYour
InstanceId with f3 instance ID .
```

以下は返されるメッセージの例です。 メッセージ内の FPGA イメージの UUID と、書き留めた FPGA イメージの UUID が同じで、メッセージに `" TaskStatus ":" valid "` と表示されている場合、イメージは正常にダウンロードされました。

よくある質問

イメージのアップロード中に発生したエラーの詳細を表示する方法について。

プロジェクトがイメージのアップロード中にエラー (コンパイルエラーなど) を報告した場合は、次の 2 つの方法のいずれかでエラーの詳細を表示できます。

- Check faas_compiling.log. アップロードスクリプト faas_upload_and_create_image.sh を使用すると、コンパイルが失敗した場合に faas_compiling.log が自動的にダウンロードされて端末に出力されます。
- 次のコマンドを実行して、ログファイルを表示します: `sh / root / xbinst_oem / tool / faas_check log . sh < bit . tar . gz - package uploaded previously >`。

イメージを再度読み込む方法について。

イメージを再度読み込むには、次の手順を実行します。

1. インスタンス上でコマンドを実行してドライバーをアンインストールします。

```
sudo rmmod xdma
sudo rmmod xocl
```

```
```

2. 次の 2 つの方法のいずれかでイメージをダウンロードします。

   - スクリプトを使用します。 最後の番号は、インスタンスの FPGA シリアル番号を表します:

     ```
     sh  faas_downl  oad_image . sh  bit . tar . gz  0
     ```

   - faascmd の使用: `faascmd  download_i  mage  -- instanceId =`

     ```
     hereIsYour  InstanceId  -- fpgauuid = hereIsFpga  UUID  -- fpgatype
     = xilinx  -- imageuuid = hereIsImag  eUUID  -- imagetype = afu  --
     shell = f30001
     ```

3. ドライバーをインストールします。

```
sudo  depmod
sudo  modprobe  xdma
```

8.5 faascmd ツール

8.5.1 faascmd の概要

faascmd は Alibaba Cloud FPGA クラウドサーバーが提供するコマンドラインツールです。 Python SDK に基づいて開発されたスクリプトです。

faascmd を使って次のことができます。

- 権限付与および関連操作の実行
- FPGA イメージの管理および操作
- オブジェクトの表示およびアップロード
- FPGA インスタンスに関する情報の取得

8.5.2 faascmd のインストール

ここでは、faascmd をダウンロードしてインストールする方法について説明します。

準備

- faascmd を実行するインスタンスで以下の手順を実行します。

  1. 次のコマンドを実行して、Python のバージョンが 2.7.x であることを確認します。

     ```
     python  - V
     ```

     ```
     [root@testhost script]# python -V
     Python 2.7.5
     ```

  2. 以下のコマンドを実行して Python モジュールをインストールします。

     ```
     pip  - q   install   oss2
     pip  - q   install   aliyun - python - sdk - core
     pip  - q   install   aliyun - python - sdk - faas
     pip  - q   install   aliyun - python - sdk - ram
     ```

  3. 次のコマンドを実行して、aliyun-python-sdk-core のバージョンが 2.11.0 以降であることを確認します。

     ```
     cat  / usr / lib / python2 . 7 / sitepackag  es / aliyunsdkc  ore
     / __init__ . py
     ```

     ```
     [root@testhost python2.7]# cat /usr/lib/python2.7/site-packages/aliyunsdkcore/__init__.py
     __version__ = "2.11.0"[root@testhost python2.7]#
     ```

     > 注：
     > バージョンが2.11.0より前の場合は、`pip  install  -- upgrade  aliyun - python - sdk - core` を実行し、aliyun-python-sdk-core を最新バージョンにアップグレードします。

- RAM ユーザーの AccessKeyID と AccessKeySecret を取得します。

手順

1. インスタンスにログインし、現在または他のディレクトリで `wget  http :// fpga - tools . oss - cn - shanghai . aliyuncs . com / faascmd` を実行して faascmd をダウンロードします。

   > 注：
   > faascmd の設定を実行する場合は、faascmd がインストールされているディレクトリの絶対パスを PATH 変数に追加する必要があります。

2. 次のコマンドを実行して、faascmd に実行可能なアクセス許可を追加します。

```
chmod + x faascmd
```

8.5.3 faascmd の設定

faascmd を使用する前に、関連する環境変数と RAM ユーザーの AccessKey を設定する必要があります。

1. インスタンスにログインし、次のコマンドを実行して PATH 環境変数を設定します。

```
export PATH =$ PATH :< path where faascmd is located >
```

2. 次のコマンドを実行して、AccessKey (つまり、AccessKeyId と AccessKeySecret) を設定します。

```
faascmd config -- id =< yourAccess KeyID > -- key =<
yourAccess KeySecret >
```

```
[root@testhost script]# faascmd config --id=                --key=
Your configuration is saved into /root/.faascredentials .
[root@testhost script]#
```

8.5.4 faascmd の使用

ここでは、faascmd コマンドの使用方法について説明します。

前提条件

使用する前に faascmd を設定しておきます。

構文の説明

- faascmd が提供するコマンドとパラメーターはすべて、大文字と小文字を区別します。
- faascmd コマンドのパラメーターでは、等号 (=) の前後にスペースを入れないでください。

ユーザーの権限付与

" `faascmd auth` "コマンドを使用して、faas の管理者ユーザーにユーザーの OSS バケットへのアクセス権限を付与します。

前提条件

1. 最初にコンパイルされた DCP ファイルをアップロードするために、FaaS 用の OSS バケットを作成しておきます。
2. FaaS OSS バケットに"compiling_logs" という名前のフォルダーを作成しておきます。

コマンド形式

```
faascmd auth -- bucket =< yourFaasOS SBucketNam e >
```

コード例

```
[root@testhost script]# faascmd auth --bucket=juliabucket
faasRole has existed!
RAMSECTION has existed!
OSSSECTION has existed!
RoleArn: acs:ram::[redacted]:role/faasrole
Create role success
faasPolicy has not existed! Create it Now!
Create policy success
Attach policy to role success
0.459(s) elapsed
```

注：

Alibaba Cloud アカウントに複数の RAM ユーザーアカウントがある場合は、権限付与ポリシーが繰り返し変更または上書きされないように、RAM ユーザーアカウントが OSS バケットを共有することを推奨します。

権限付与ポリシーの表示

`faascmd list_polic y` **コマンドを使用して、指定された OSS バケットが、対応する権限付与ポリシー (faasPolicy) に追加されているかどうかを表示します。**

コマンド形式

```
faascmd list_polic y
```

コード例

```
[root@testhost script]# faascmd list_policy
VersionId : v1   CreateTime : 2018-11-09T03:22:01Z   IsDefaultVersion : True
{
  "Statement": [
    {
      "Action": "ecs:DescribeInstances",
      "Effect": "Allow",
      "Resource": "acs:ecs:*:*:*"
    },
```

注：

自身の OSS バケットと OSS バケットまたは compiling_log がポリシー情報に表示されているかどうかを確認する必要があります。

権限付与ポリシーの削除

`faascmd delete_pol icy` コマンドを使用して、権限付与ポリシー (faasPolicy) を削除します。

コマンド形式

```
faascmd delete_pol icy
```

コード例

```
[root@testhost script]# faascmd delete_policy
Detach faasPolicy from faasRole successfully!!!
Delete the faasPolicy successfully!!!
0.306(s) elapsed
```

注：
Alibaba Cloud アカウントに複数の RAM ユーザーアカウントがある場合は、誤って権限付与ポリシーを削除しないように、RAM コンソールでターゲットポリシーを削除することを推奨します。

OSS バケットの下にあるすべてのオブジェクトの表示

`faascmd list_objec ts command` コマンドを使用して、OSS バケットの下にあるオブジェクトをすべて表示します。

コマンド形式

```
faascmd list_objec ts
```

コード例

```
[root@testhost script]# faascmd list_objects
compiling_logs/
juliabucket
juliafile
0.081(s) elapsed
[root@testhost script]# faascmd list_objects |grep "julia"
0.082(s) elapsed
juliabucket
juliafile
```

注：

grep コマンドと一緒にこのコマンドを使用して、目的のファイル (たとえば `faascmd list_objec ts | grep " xxx`) をフィルタリングします。

元のコンパイルファイルのアップロード

`faascmd upload_obj ect` コマンドを使用して、ローカル PC 上でコンパイルされた元のファイルを、指定された OSS バケットにアップロードします。

コマンド形式

```
faascmd upload_obj ect -- object =< newFileNam einOSSBuck et >
-- file = < your_file_ path >/ fileNameYo uWantToUpl oad
```

コード例

```
[root@testhost script]# faascmd upload_object --object=juliaOSSFile1 --file=julia_test.tar
juliaOSSFile1
julia_test.tar
0.091(s) elapsed
[root@testhost script]# faascmd upload_object --object=juliaOSSFile2 --file=/opt/dcp1_0/testfile.tar
juliaOSSFile2
/opt/dcp1_0/testfile.tar
0.089(s) elapsed
```

注:

- ターゲットファイルが現在のディレクトリに格納されている場合、パスは必要ありません。
- Intel FPGA が提供するローカルでコンパイルされた元のファイルは .gbs 形式で、Xilinx FPGA が提供するものは、スクリプト処理後に .tar 形式のパッケージとして圧縮されています。

OSS バケットからのオブジェクトのダウンロード

`faascmd get_object` コマンドを使用して、指定されたオブジェクトをOSS バケットからダウンロードします。

コマンド形式

```
faascmd get_object -- obejct =< yourObject Name > -- file =<
your_local _path >/< yourFileNa me >
```

コード例

```
[root@ ~]# faascmd get_object --object=juliaOSSFile3 --file=vivado1.log
2018-12-04 10:09:47,342 oss2.api [INFO] 140410558318400 : Init oss bucket, endpoint: https://oss-cn-hangzhou-internal.aliyuncs.com, isCname: False, connect_timeout: None, app_name: , enabled_crc: True
juliaOSSFile3
vivado1.log
2018-12-04 10:09:47,344 oss2.api [INFO] 140410558318400 : Start to get object to file, bucket: juliabucket, key: juliaOSSFile3, file path: vivado1.log
2018-12-04 10:09:47,344 oss2.api [INFO] 140410558318400 : Start to get object, bucket: juliabucket, key: juliaOSSFile3, range: , headers: {}, params: {}
2018-12-04 10:09:47,456 oss2.api [INFO] 140410558318400 : Get object done, req_id: 5C05E1EB874F5A9B75E1728B, status_code: 200
0.117(s) elapsed
```

注:

パスが指定されていない場合、オブジェクトはデフォルトで現在のフォルダーにダウンロードされます。

FPGA イメージの作成

`faascmd create_image` コマンドを使用して、FPGA イメージ作成リクエストを送信します。 リクエストが成功すると、"fpga imageuuid" と返されます。

コマンド形式

```
 faascmd   create_ima  ge  -- object =< yourObject  Name >
-- fpgatype =< intel / xilinx >  -- encrypted =< true / false >
-- kmskey =< key / mandatory   if   encrypted   is   true >
-- shell =< Shell   Version / mandatory > -- name =< name / optional
 >
-- descriptio  n =< descriptio  n / optional > -- tags =< tags /
 optional >
```

コード例

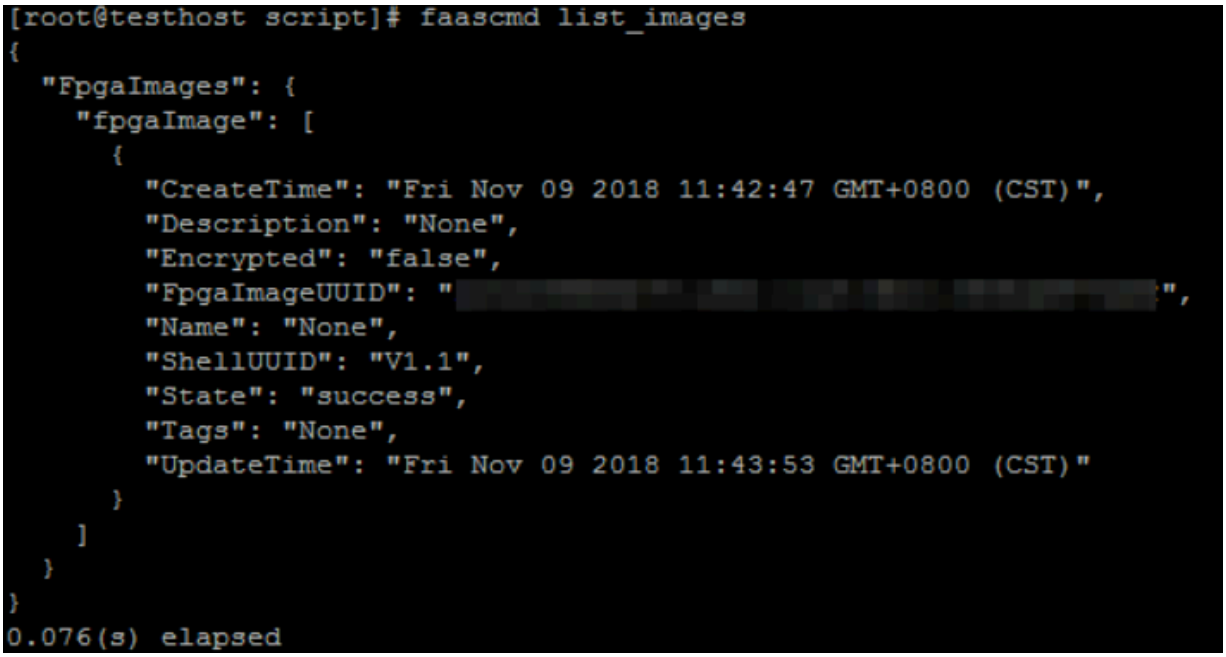
FPGA イメージの表示

`faascmd list_image s` コマンドを使用して、作成したすべての FPGA イメージに関する情報を表示します。

コマンド形式

```
 faascmd   list_image  s
```

コード例



> 注：
> RAM ユーザーアカウントごとに最大 10 個の FPGA イメージを予約します。

FPGA イメージの削除

`faascmd delete_image` コマンドを使用して、FPGA イメージを削除します。

コマンド形式

```
faascmd delete_image --imageuuid =< yourImageuuid >
```

コード例

```
[root@testhost script]# faascmd delete_image --imageuuid=
{"Status":200,"FpgaImageUUID":"                                    ","Message":"delete succeed!"}
0.143(s) elapsed
```

FPGA イメージのダウンロード

`faascmd download_image` コマンドを使用して、FPGA イメージのダウンロードリクエストを送信します。

コマンド形式

```
faascmd download_image --instanceId =< yourInstance Id >
-- fpgauuid =< yourfpgauuid > -- fpgatype =< intel / xilinx >
-- imageuuid =< yourImageuuid > -- imagetype =< afu >
-- shell =< yourImageShellVersion >
```

コード例

```
faascmd download_image --instanceId = XXXXX --fpgauuid = XXXX
 --fpgatype = intel --imageuuid = XXXX
```

FPGA イメージのダウンロードステータスの表示

`faascmd fpga_status` コマンドを使用して、現在の FPGA ボードカードのステータスと FPGA イメージのダウンロードステータスを表示します。

コマンド形式

```
faascmd fpga_status --fpgauuid =< fpgauuid > -- instanceId =<
instanceId >
```

コード例

```
[root@testhost script]# faascmd fpga_status --fpgauuid=                    --instanceId=
{"shellUUID":"V1.0","FpgaImageUUID":"                                    ","FpgaUUID":"                    ",
askStatus":"invalid","Encrypted":"false"}
0.310(s) elapsed
```

FPGA イメージの発行

`faascmd publish_image` コマンドを使用して、FPGA イメージ発行リクエストを送信します。

コマンド形式

```
faascmd publish_image -- imageuuid =< yourImageuuid > --imageid =< yourFPGAImageid >
```

> **注：**
>
> - `imageuuid` は、クラウドマーケットプレースに発行しようとしているイメージの ID です。`faascmd list_images` コマンドを実行して、イメージ ID を表示します。
> - `imageid` は FPGA イメージ ID です。ECS コンソールのインスタンスの詳細ページで ID を確認します。

FPGA インスタンス情報の表示

`faascmd list_instances` コマンドを使用して、インスタンス ID、FPGA ボードカード情報、シェルバージョンなど、FPGA インスタンスに関する基本情報を取得します。

コマンド形式

```
faascmd list_instances --instanceId =< yourInstanceId >
```

コード例

```
[root@testhost script]# faascmd list_instances --instanceId=
{
  "Instances": {
    "instance": [
      {
        "DeviceBDF": "05:00.0",
        "FpgaStatus": "invalid",
        "FpgaType": "intel",
        "FpgaUUID": "",
        "InstanceId": "",
        "ShellUUID": "V1.1"
      }
    ]
  }
}
0.275(s) elapsed
```

8.5.5 よくある質問

ここでは、faascmd ツールに関する共通のよくある質問をリストアップし、対応する解決策を提供します。

よくある質問

- "Name Error:global name'ID' is not defined."というエラーが報告された場合はどうしたらいいですか?

  原因: faascmd が AccessKeyId または AccessKeySecret を取得できません。

  解決策: `faascmd config` コマンドを実行します。 そして、入力した AccessKeyId と AccessKeySecret に関する情報が"`/ root /. faascreden tials`"ファイルに保存されます。

- "HTTP Status:403 Error:RoleAccessError. You have no right to assume this role."というエラーが報告された場合はどうしたらいいですか?

  原因: faascmd がロール ARN に関する情報を取得できないか、取得した ARN が既存の AccessKeyId および AccessKeySecret と同じアカウントに属していません。

  解決策:"`/ root /. faascreden tials`" ファイルに 以下の情報が含まれているかどうかを確認します。

```
[ FaaSCreden tials ]
 accessid = xxxxxxxxxx
 accesskey = xxxxxxxxxx xxxxxxxxxx xxx
[ Role ]
 role = acs : ram :: 1234567890 123456 : role / xxxxxx
[ OSS ]
 bucket = xxxx
```

注:

- 上記の情報が既に存在する場合は、ロール ARN と AccessKeyId および AccessKeySecret が同じアカウントに属しているかどうかを確認します。
- 上記の情報が存在しない場合は、`faascmd auth bucket = xxxx` を実行して、権限を付与します。

- "HTTP Status: 404 Error: EntityNotExist. Role Error. The specified Role not exists." というエラーが報告された場合はどうしたらいいですか?

  原因: アカウントに faasrole ロールがありません。

  解決策: RAM コンソールにログインして、faasrole ロールが存在するかどうかを確認します。

  - faasrole ロールが存在しない場合は、" `faascmd config` "コマンドおよび" `faascmd auth` "コマンドを実行して、そのロールを作成して、それに権限を付与します。
  - faasrole ロールがすでに存在する場合は、チケットを起票し、サポートセンターへお問い合わせください。
- "SDK.InvalidRegionId. Can not find endpoint to access."というエラーが報告された場合は どうしたらいいですか?

  原因: faascmd が FaaS のエンドポイントアドレスを取得できません。

  解決策: 次の手順を実行して、faascmd 設定が指定の要件を満たしているかどうかを確認します。

  - `python -V` コマンドを実行して、Pythonのバージョンが 2.7.x かどうかを確認します。
  - `which python` コマンドを実行して、Python のデフォルトのインストールパスが `/usr/bin/python` かどうかを確認します。
  - `cat /usr/lib/python2.7/site-packages/aliyunsdkcore/__init__.py` コマンドを実行して、aliyunsdkcore のバージョンが 2.11.0 以降であるかどうかを確認します。

     注:
    aliyunsdkcore のバージョンが 2.11.0 より前の場合は、`pip install --upgrade aliyun-python-sdk-core` コマンドを実行して、aliyunsdkcore を最新バージョンにアップグレードする必要があります。

- イメージをダウンロードしたときに "HTTP Status:404 Error:SHELL NOT MATCH The image Shell is not match with fpga Shell! Request ID:D7D1AB1E-8682-4091-8129-C17D54FD10D4" と返された場合はどうしたらいいですか?

  原因: ターゲット FPGA イメージと指定された FPGA のシェルバージョンが一致していません。

  解決策: 次の手順を実行します。

  - `faascmd list_instances --instance xxx` コマンドを実行して、現在の FPGA のシェルバージョンを確認します。
  - `faascmd list_images` コマンドを実行して、指定された FPGA イメージのシェルバージョンを確認します。

    > 注:
    > ■ 2 つのシェルバージョンが異なる場合は、シェルバージョンが FPGA のシェルバージョンと同じ FPGA イメージを作成してから、イメージをダウンロードする必要があります。
    > ■ 2 つのシェルバージョンが一致している場合は、チケットを起票し、サポートセンターへお問い合わせください。

- イメージをダウンロードしたときに "HTTP Status:503 Error:ANOTHER TASK RUNNING. Another task is running,user is allowed to take this task half an hour Request ID:5FCB6F75-8572-4840-9BDC-87C57174F26D" と返された場合はどうしたらいいですか?

  原因: 予期しない失敗、または送信したダウンロードリクエストの中断により、FPGA が動作状態のままになっています。

  解決策: ダウンロードタスクが終了するまで 10 分間待ってから、イメージのダウンロードリクエストを再送信します。

  > 注:
  > 問題が解決しない場合は、チケットを開起票し、サポートセンターへお問い合わせください。

- faascmd list_images コマンドを実行したときに、イメージのステータスが Failed になっている場合はどうしたらいいですか?

  解決策: 次のコマンドを実行して、トラブルシューティング用のコンパイルログを取得します。

  ```
  faascmd list_objects | grep vivado
  ```

```
faascmd   get_object  -- obejct =< yourObject  Name > -- file =<
your_local  _path >/ vivado . log   # The   path   is   optional
```

. コンパイルログは、既定では現在のフォルダーにダウンロードされます。

共通エラーコード

faascmd コマンド	API 名	エラーメッセージ	エラーの説明	エラーコード
Applicable to all commands	すべての API に適用可能	PARAMETER INVALIDATE	入力パラメーターが正しくありません。	400
Applicable to all commands	すべての API に適用可能	InternalError	内部エラーがあります。 チケットを起票し、サポートセンターへお問い合わせください。	500
auth	auth	NoPermisson	特定のオープン API にアクセスする権限がありません。	403
create_image	CreateFpgaImage	IMAGE NUMBER EXCEED	イメージリストには、10 を超えるイメージを含めることはできません。 不要なイメージを削除して、もう一度やり直してください。	401
		FREQUENCY ERROR	イメージリクエストを送信する間隔は 30 分です。	503
		SHELL NOT SUPPORT	入力シェルバージョンはサポートされていません。 シェルのバージョンが正しいことを確認します。	404
		EntityNotExist.RoleError	現在のアカウントには faasrole ロールはありません。	404
		RoleAccessError	ロール ARN が空であるか、ロール ARN と AccessKeyId または AccessKeySecret が同じアカウントに属していません。	403
		InvalidAccessKeyIdError	AccessKeyId または AccessKeySecret が無効です。	401
		Forbidden.KeyNotFoundError	指定された KMS キーが見つかりません。 KMS コンソールにログインし、入力 KeyId が存在するかどうかを確認します。	503
		AccessDeniedError	faas 管理者アカウントには、現在のバケットにアクセスする権限がありません。	

faascmd コマンド	API 名	エラーメッセージ	エラーの説明	エラーコード
		OSS OBJECT NOT FOUND	指定された OSS バケットおよびオブジェクトが存在しないか、またはアクセスできません。	404
delete_image	DeleteFpgaImage	IMAGE NOT FOUND	指定された FPGA イメージが見つかりません。	400
list_instances	DescribeFpgaInstances	NOT AUTHORIZED	指定されたインスタンスが存在しないか、現在のアカウントに属していません。	401
		RoleAccess Error	ロール ARN が空であるか、ロール ARN と AccessKeyId または AccessKeySecret が同じアカウントに属していません。	403
		INSTANCE INVALIDATE	指定されたインスタンスは FPGA インスタンスではありません。 指定したインスタンスが FPGA インスタンスの場合は、チケットを起票し、サポートセンターへお問い合わせください。	404
fpga_status	DescribeLoadTaskStatus	NOT AUTHORIZED	指定された instanceId が見つかりません。 入力パラメーターを確認してください。	401
		FPGA NOT FOUND	指定された fpgauuid が見つかりません。 入力パラメーターを確認してください。	404
download_image	LoadFpgaImage	ANOTHER TASK RUNNING	送信したイメージダウンロードタスクはまだ動作状態です。	503
		IMAGE ACCESS ERROR	指定されたイメージは現在のアカウントに属していません。	401
		YOU HAVE NO ACCESS TO THIS INSTANCE	指定されたインスタンスは現在のアカウントに属していません。	401
		IMAGE NOT FOUND	指定された FPGA イメージが見つかりません。	404
		FPGA NOT FOUND	指定された FPGA が見つかりません。	404

faascmd コマンド	API 名	エラーメッセージ	エラーの説明	エラーコード
		SHELL NOT MATCH	イメージと指定された FPGA がシェルバージョンで一致しません。	404
		RoleAccess Error	ロール ARN が空である、またはロール ARN と AccessKeyId または AccessKeySecret が同じクラウドアカウントに属していません。	403
		Image not in success state	指定されたイメージは成功状態ではありません。 成功状態のイメージのみダウンロードできます。	404
publish_image	PublishFpgaImage	FPGA IMAGE STATE ERROR	指定されたイメージは成功状態ではありません。	404
		FPGA IMAGE NOT FOUND	指定されたイメージが見つからない、または現在のアカウントに属していません。	404

9 ディスクの圧縮

現在、ECS (Elastic Compute Service) は、システムディスクやデータディスクの圧縮をサポートしていません。 ディスクボリュームを圧縮する場合は、代わりに Alibaba Cloud Migration Tool を試します。

Cloud Migration Tool は Alibaba Cloud ユーザーのクラウドベースとオフラインのワークロードのバランスをとるように設計されていますが、それを使用して ECS ディスクボリュームを圧縮できます。

このツールは、ECS インスタンスに基づいてカスタマイズイメージを作成します。 このプロセスの間に、ディスクのサイズを再指定し、圧縮します。 ターゲットオブジェクトを ECS インスタンスで置き換えることを除けば、クラウド移行とディスクボリューム圧縮のためのツールは、操作と制限の両方の点で 同じです。 ECS インスタンスは既に仮想化されているため、使用するにはより便利で、エラーを報告する可能性が少なくなります。

ただし、このツールを使用すると、ECS インスタンスの一部の属性が変わる可能性があります。 たとえば、インスタンス ID (`InstanceId`) やパブリック IP アドレスです。 インスタンスが VPC 接続インスタンスの場合、パブリック IP アドレスを EIP アドレスに変換することにより、パブリック IP アドレスを保存します。 Alibaba Cloud EIP (Elastic IP) を使用するユーザーとパブリック IP への依存度が低いユーザーは、このアプローチを使用してディスクを圧縮することを推奨します。

前提条件

- ディスクが Linux インスタンスにマウントされたら、まずリモートデータ同期ツールである rsync をインストールする必要があります。
  - CentOS インスタンス: `yum install rsync -y` を実行します。
  - Ubuntu インスタンス: `apt-get install rsync -y` を実行します。
  - Debian インスタンス: `apt-get install rsync -y` を実行します。
  - その他の配布: 公式 Web サイトにアクセスして関連するインストール文書を探します。
- 最初にコンソールで AccessKey を作成する必要があり、これを使用して設定ファイル "user_config.json" に出力します。

> 注 :
> AccessKey に対する過剰なアクセス許可によるデータ漏洩を防ぐために、RAM サブアカウントを作成し、そのアカウントを使用して AccessKey を作成することを推奨します。

- 他の前提条件と制限については、「Cloud Migration Tool を使用した Alibaba Cloud への移行」をご参照ください。

手順

1. 管理者またはルートアカウントを使用してターゲット ECS インスタンスに接続します。
2. Alibaba Cloud Migration Tool の zip ファイルをダウンロードします。
3. Cloud Migration Tool を解凍します。 対応するオペレーティングシステムとクライアントファイルディレクトリのバージョンを入力して、設定ファイル "user_config.json" を見つけます。
4. カスタマイズされた "user_config.json" を参照して、設定を完了します。

   Linux インスタンスの設定ファイルについては、次の図をご参照ください。

   ディスクボリュームを圧縮するために設定する最も重要なパラメーターは次のとおりです。

   - `system_dis k_size`：このパラメーターを、期待するシステムディスクサイズ (GB 単位) に設定します。 値はシステムディスクの実際のサイズより小さくすることはできません。
   - `data_disks`：このパラメーターを、期待するデータディスクサイズ (GB 単位) に設定します。 値はデータディスクの実際のサイズより小さくすることはできません。

   > 注：
   >
   > - Linux インスタンスにデータディスクが付属しているときは、データディスクのボリュームを圧縮したくない場合でも "`data_disks`" パラメーターが必要です。 それが設定されていない場合は、Cloud Migration Tool はデフォルトでデータディスクからシステムディスクにデータをコピーします。
   > - Windows インスタンスにデータディスクが付属しているときは、データディスクのサイズを圧縮しない場合、"`data_disks`" パラメーターは省略可能になります。

5. 以下のように、go2aliyun_client.exe プログラムを実行します。
   - Windows インスタンス: "go2aliyun_client.exe" を右クリックし、[管理者として実行] を選択します。
   - Linux インスタンス:
     a. `chmod + x go2aliyun_ client` を実行して、クライアントに実行可能権限を与えます。
     b. `./ go2aliyun_ client` を実行して、クライアントを実行します。
6. 実行結果を待ちます。
   - `Goto Aliyun Finished !` と表示されたら、ECS コンソールにアクセスし、圧縮後のカスタマイズイメージを確認します。 カスタマイズイメージが作成されている場合は、元のインスタンスをリリースし、カスタマイズイメージを使用して ECS インスタンスを作成します。 新しいインスタンスを作成したら、ディスクボリュームの圧縮プロセスは完了です。
   - `Goto Aliyun Not Finished !` と表示された場合は、トラブルシューティング用の同じディレクトリ内のログファイルを確認します。 問題を解決したら、Cloud Migration Tool を再度実行してボリュームの圧縮を再開します。 このツールは最新の移行の進行状況を継続し、最初からやり直すことはありません。

参照

- Cloud Migration Tool の導入詳細については、「Alibaba Cloud Migration Tool の概要」をご参照ください。
- Cloud Migration Tool の使用方法については、「Cloud Migration Tool を使用した Alibaba Cloud への移行 」をご参照ください。

10 ECS ステータス変更イベントの処理

このトピックでは、CloudMonitor が MNS メッセージキューを使用して、ECS ステータス変更イベントを自動的に処理する方法について説明します。

概要

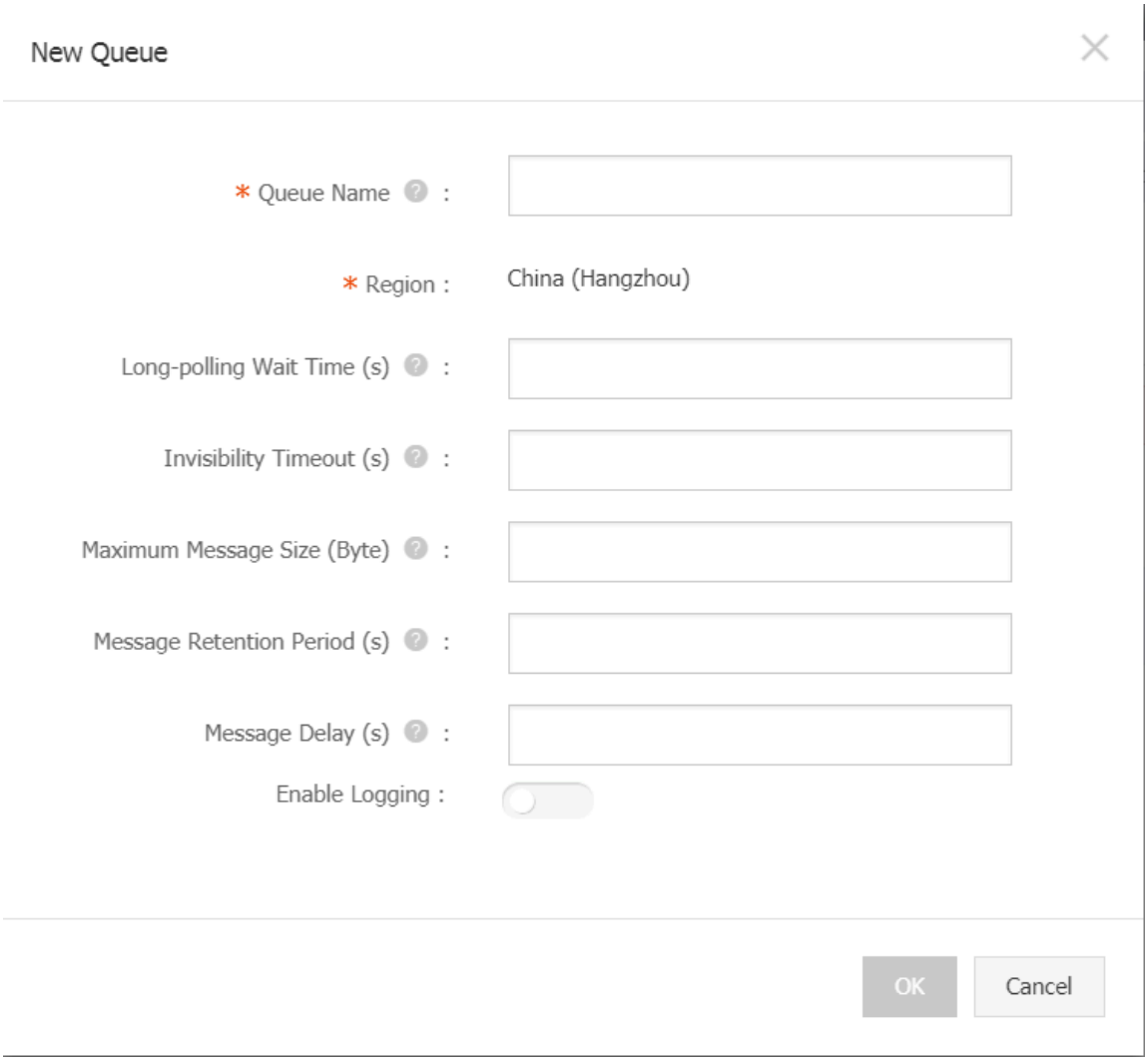
インスタンスステータスが変更されると、ECS インスタンスステータス変更イベントがトリガーされます。 具体的には、ステータス変更イベントは、コンソール上の操作、API または SDK の使用、自動スケーリング、料金滞納の検出、システム例外などに起因する変更です。

ECS ステータス変更イベントの処理を自動化するために、CloudMonitor では、関数計算式と MNS メッセージキューの 2 つの方法を提供しています。 このトピックでは、MNS メッセージキューを使用する 3 つのベストプラクティスについて説明します。

準備

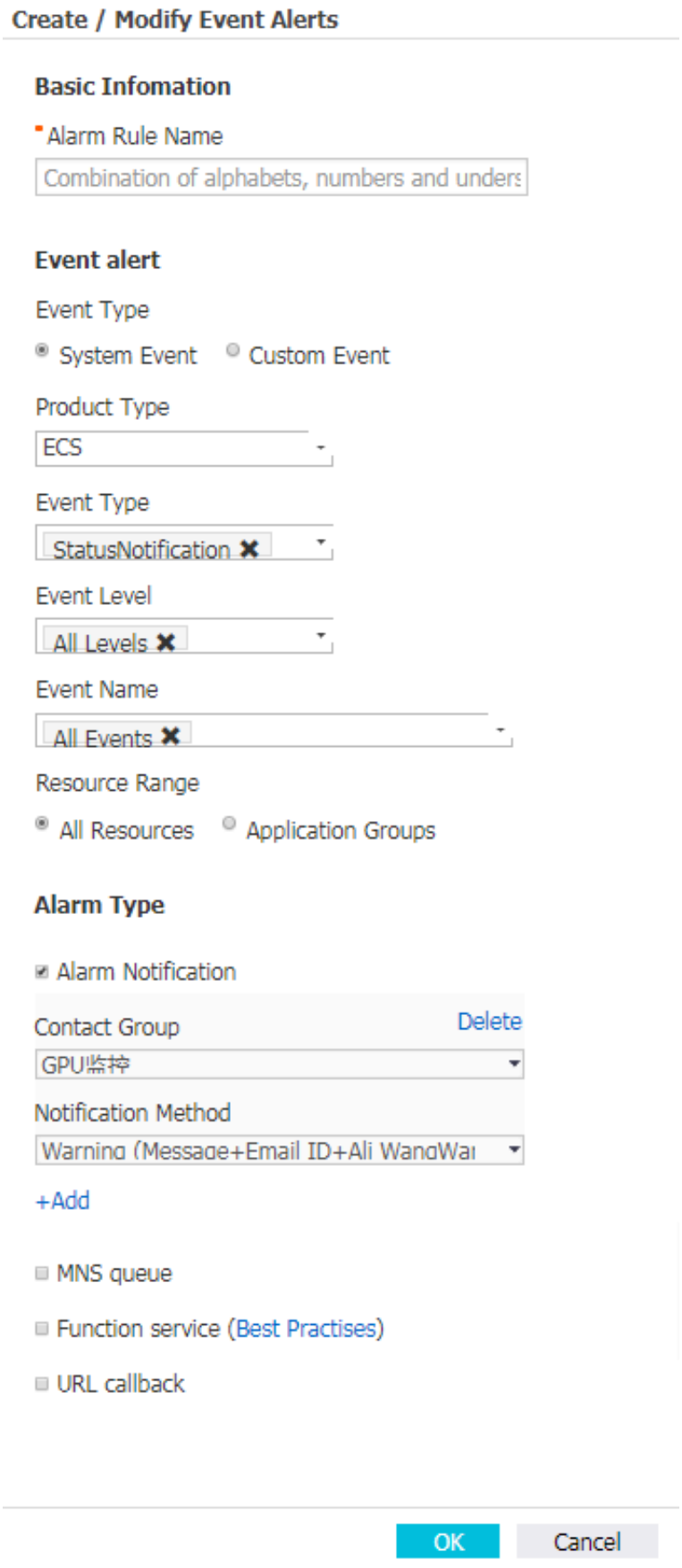
・ メッセージキューを作成します。

1. MNS コンソールにログインします。
2. [キューリスト] ページで、対象のリージョンを選択し、右上隅の [キューの作成] をクリックします。



3. [新しいキュー] ダイアログボックスで、[キュー名] (例：ecs-cms-event) およびその他の必要な情報を入力し、[OK] をクリックします。

- ステータス変更イベント用のアラームルールを作成します。
  1. Cloud Monitor 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で、[イベントモニタリング] をクリックします。
  3. [アラームルール] タブページに移動し、 [イベントアラーム作成] をクリックします。



4. [基本情報] エリアで、アラームルール名称 (たとえば、ecs-test-rule) を入力します。
5. [イベントアラーム] エリアで、以下のとおりパラメーターを設定します。

- [イベントタイプ] を [システムイベント] に設定します。
- [製品タイプ] を [ECS] に、[イベントタイプ] を [StatusNotiifcation] に設定し、その他のパラメーターを必要に応じて設定します。
- [リソース範囲] が [全リソース] に設定されている場合、任意のリソースの変更イベントが、通知をトリガーします。 [リソース範囲] が [アプリケーショングループ] に設定されている場合、指定されたグループ内のリソースの変更イベントのみが通知をトリガーします。

6. [アラームタイプ] エリアで [MNS queue] を選択し、[リージョン] と [queue] (たとえば、ecs-cms-event) を指定します。
7. [OK] をクリックします。

・ Python の依存関係をインストールします。

次のコードは Python 3.6 でテストされています。 必要に応じて、Java などの他のプログラミング言語を使用できます。

PyPi を使用して、以下の Python 依存関係をインストールします。

- aliyun-python-sdk-core-v3 of 2.12.1 以降
- aliyun-python-sdk-ecs of 4.16.0 以降
- aliyun-mns of 1.1.5 以降

手順

CloudMonitor は、ECS インスタンスのすべてのステータス変更イベントを MNS に送信します。 その後、MNS から通知を取得し、コードを実行してそれらを処理できます。 次の演習セクションでは、上に述べた方法のチュートリアルを説明します。

演習1：すべての ECS 作成およびリリースイベントの記録

現在、リリースされたインスタンスを ECS コンソールで照会することはできません。 これらの照会を実行する必要がある場合は、すべての ECS インスタンスのライフサイクルを独自のデータベースに記録するか、ECS ステータス変更イベントを使用して、ログに記録する必要があります。 具体的には、ECS インスタンスが作成されるたびに Pending イベントが送信され、ECS インスタンスがリリースされるたびに Deleted イベントが送信されます。 次の手順を実行して、これら 2 つのイベントを記録できます。

1. *conf* ファイルを作成します。ファイルには、MNS エンドポイント、Alibaba Cloud アカウントの AccessKeyId と AccessKeySecret、リージョン ID (たとえば、cn-beijing)、および MNS キュー名が含まれている必要があります。

注:

MNS エンドポイントを表示するには、MNS コンソールにログインし、[キューリスト] ページで、[エンドポイントの取得] をクリックします。

```
class   Conf :
    endpoint  = ' http ://< id >. mns .< region >. aliyuncs . com
/'
    access_key  = '< access_key >'
    access_key  _secret  = '< access_key  _secrect >'
    region_id  = ' cn - beijing '
    queue_name  = ' test '
    vsever_gro  up_id  = '< your_vserv  er_group_i  d >'
```

2. MNS SDK を使用して、MNS メッセージを受信するように MNS クライアントをコンパイルします。

```
# -*-  coding :  utf - 8  -*-
import   json
from   mns . mns_except  ion   import   MNSExcepti  onBase
import   logging
from   mns . account   import   Account
from  .  import   Conf


class   MNSClient ( object ):
    def   __init__ ( self ):
        self . account  =   Account ( Conf . endpoint ,  Conf .
access_key ,  Conf . access_key  _secret )
        self . queue_name  =  Conf . queue_name
        self . listeners  =  dict ()

    def   regist_lis  tener ( self ,  listener ,  eventname ='
Instance : StateChang  e '):
        if   eventname   in   self . listeners . keys ():
            self . listeners . get ( eventname ). append ( listener
)
        else :
            self . listeners [ eventname ] = [ listener ]

    def   run ( self ):
        queue  =  self . account . get_queue ( self . queue_name )
        while   True :
            try :
                message  =  queue . receive_me  ssage ( wait_secon
ds = 5 )
                event  =  json . loads ( message . message_bo  dy )
                if   event [' name ']  in   self . listeners :
                    for   listener   in   self . listeners . get (
event [' name ']):
                        listener . process ( event )
                queue . delete_mes  sage ( receipt_ha  ndle =
message . receipt_ha  ndle )
            except   MNSExcepti  onBase   as   e :
```

```
                if   e . type  == ' QueueNotEx  ist ':
                    logging . error (' Queue  % s   not   exist ,
please   create   queue   before   receive   message .',  self .
queue_name )
                else :
                    logging . error (' No   Message ,  continue
waiting ')


class   BasicListe  ner ( object ):
    def   process ( self ,  event ):
        pass
```

上記のコードは、MNS メッセージを取得し、リスナー消費メッセージが呼び出された後にメッセージを削除するためにのみ使用されます。

3. 指定されたイベントを使用するようにリスナーを登録します。 このリスナーは、Pending または Deleted イベントを受信したと判断すると、ログファイルに行を出力します。

```
# -*-  coding :  utf - 8  -*-
import   logging
from  . mns_client   import   BasicListe  ner


class   ListenerLo  g ( BasicListe  ner ):
    def   process ( self ,  event ):
        state  =  event [' content '][' state ']
        resource_i  d  =  event [' content '][' resourceId ']
        if   state  == ' Panding ':
            logging . info ( f ' The   instance  { resource_i  d }
state   is  { state }')
        elif   state  == ' Deleted ':
            logging . info ( f ' The   instance  { resource_i  d }
state   is  { state }')
```

以下の Main 関数も使用できます。

```
mns_client  =  MNSClient ()

mns_client . regist_lis  tener ( ListenerLo  g ())

mns_client . run ()
```

実際のシナリオでは、イベントをデータベースに格納するか SLS を使用して、後ほど行われる検索および監査タスクを容易にすることができます。

演習 2：ECS サーバーの自動再起動

シナリオによっては、ECS サーバーが突然シャットダウンすることがあります。 この場合、サーバーの自動再起動を設定する必要があります。

演習 1 で MNS クライアントを使用して、新しいリスナーを作成します。 リスナーが Stopped イベントを受け取ると、リスナーはターゲット ECS サーバーで、 `Start` コマンドを実行します。

```
# -*-  coding :  utf - 8  -*-
 import   logging
 from   aliyunsdke  cs . request . v20140526   import   StartInsta
 nceRequest
 from   aliyunsdkc  ore . client   import   AcsClient
 from  . mns_client   import   BasicListe  ner
 from  . config   import   Conf


 class   ECSClient ( object ):
     def   __init__ ( self ,  acs_client ):
         self . client  =  acs_client

    # Start   the   ECS   instance
     def   start_inst  ance ( self ,  instance_i  d ):
         logging . info ( f ' Start   instance  { instance_i  d
 } ...')
         request  =  StartInsta  nceRequest . StartInsta  nceRequest
 ()
         request . set_accept  _format (' json ')
         request . set_Instan  ceId ( instance_i  d )
         self . client . do_action_  with_excep  tion ( request )


 class   ListenerSt  art ( BasicListe  ner ):
     def   __init__ ( self ):
         acs_client  =  AcsClient ( Conf . access_key ,  Conf .
 access_key  _secret ,  Conf . region_id )
         self . ecs_client  =  ECSClient ( acs_client )

     def   process ( self ,  event ):
         detail  =  event [' content ']
         instance_i  d  =  detail [' resourceId ']
         if   detail [' state '] == ' Stopped ':
             self . ecs_client . start_inst  ance ( instance_i  d )
```

実際のシナリオでは、 `Start` コマンドが実行されると、Starting、Running、または Stopped イベント通知を受け取ります。 この場合、タイマーとカウンターを使用した、より詳細な O&M のためのコマンド実行時に、継続して手順を実行できます。

演習 3：リリースされる前の SLB からのプリエンプティブルインスタンスの自動削除

プリエンプティブルインスタンスがリリースされる 5 分前に、リリースアラームイベントが送信されます。 この 5 分間に、サービスを中断することなくいくつかのプロセスを実行できます。 たとえば、バックエンド SLB サーバーからターゲットのプリエンプティブルインスタンスを手動で削除できます。

演習 1 で MNS クライアントを使用して、新しいリスナーを作成します。 リスナーはプリエンプティブルインスタンスのリリースアラームを受信すると、SLB SDK を呼び出します。

```
# -*-  coding :  utf - 8  -*-
 from   aliyunsdkc  ore . client   import   AcsClient
 from   aliyunsdkc  ore . request   import   CommonRequ  est
 from  . mns_client   import   BasicListe  ner
 from  . config   import   Conf


 class   SLBClient ( object ):
     def   __init__ ( self ):
         self . client  =  AcsClient ( Conf . access_key ,  Conf .
 access_key  _secret ,  Conf . region_id )
         self . request  =  CommonRequ  est ()
         self . request . set_method (' POST ')
         self . request . set_accept  _format (' json ')
         self . request . set_versio  n (' 2014 - 05 - 15 ')
         self . request . set_domain (' slb . aliyuncs . com ')
         self . request . add_query_  param (' RegionId ',  Conf .
 region_id )

     def   remove_vse  rver_group  _backend_s  ervers ( self ,
 vserver_gr  oup_id ,  instance_i  d ):
         self . request . set_action  _name (' RemoveVSer  verGroupBa
 ckendServe  rs ')
         self . request . add_query_  param (' VServerGro  upId ',
 vserver_gr  oup_id )
         self . request . add_query_  param (' BackendSer  vers ',
                                    "[{' ServerId ':'" +  instance_i
 d  + "',' Port ':' 80 ',' Weight ':' 100 '}]")
         response  =  self . client . do_action_  with_excep  tion (
 self . request )
         return   str ( response ,  encoding =' utf - 8 ')


 class   ListenerSL  B ( BasicListe  ner ):
     def   __init__ ( self ,  vsever_gro  up_id ):
         self . slb_caller  =  SLBClient  ()
         self . vsever_gro  up_id  =  Conf . vsever_gro  up_id

     def   process ( self ,  event ):
         detail  =  event [' content ']
         instance_i  d  =  detail [' instanceId ']
         if   detail [' action '] == ' delete ':
             self . slb_caller . remove_vse  rver_group  _backend_s
 ervers ( self . vsever_gro  up_id ,  instance_i  d )
```

:

プリエンプティブルインスタンスのリリースアラームのイベント名は、"Instance:PreemptibleInstanceInterruption"、mns_client.regist_listener(ListenerSLB(Conf.vsever_group_id)、'Instance:PreemptibleInstanceInterruption') です。

実際のシナリオでは、サービスを正常に実行できるようにするために、新しいプリエンプティブルインスタンスを申請して、SLB にアタッチする必要があります。