Alibaba Cloud
云服#器 ECS

デプロイとメンテナンス

Document Version20200224

目次

1 クラウドアシスタント.....1
1.1 クラウドアシスタント..... 1
1.2 クラウドアシスタントの使用.....6
1.2.1 コマンドの作成.....6
1.2.2 コマンドの実行.....9
1.2.3 実行結果とステータスの照会.....11
1.2.4 コマンドの管理..... 15
1.3 DevOps 練習..... 17
1.4 インスタンスの自動管理.....17
2 モニタリング..... 23
2.1 システムイベント..... 23
3 インスタンス状態のモニタリング..... 28
4 操作エラーのトラブルシューティング.....29
4.1 コンソール出力とスクリーンショット..... 29

1 クラウドアシスタント

1.1 クラウドアシスタント

クラウドアシスタント クラインアントにより、お使いの **ECS** インスタンスを安全なセキュリティ保護された状態で管理することが可能になります。 特に、クラウドアシスタントにより、自動的に複数の日常のメンテナンスコマンドを実行でき、自動化 **O&M** スクリプト、処理のポーリング、ソフトウェアのインストールまたはアンインストール、アプリケーションの更新、パッチのインストールなどが実行できます。

課金方法

クラウドアシスタント クライアントは無料で利用可能です。 ただし、お使いのインスタンスの実行には料金が発生します。 詳しくは、『料金の概要』をご参照ください。

詳細

ECS インスタンスに クラウド アシスタントクライアントをインストールすることによって、**ECS** インスタンスまたは **APIundefined**を介して、**Bat** または **PowerShell** スクリプト **(Windows** インスタンス)、あるいはシェルスクリプト **(Linux** インスタンス) を複数の実行中インスタンス (`[Running]`) で実行できます。 選択されたインスタンスはお互いに影響を与えません。 時間による呼び出しを設定した場合、**ECS** インスタンスを特定のステータスに維持し、モニタリングおよびログ情報を取得し、デーモンプロセスを実行できます。 クラウドアシスタントクライアントは、どの操作も起動させることはありません。 つまり、ユーザーだけが必要に応じて特定の操作を実行します。

クラウドアシスタントクライアントに関する概念の解説は、以下の表のようになります。

表 1-1 : クラウドアシスタントクライアントの概念

用語	一般的な名称	説明
クラウドアシスタント	クラウドアシスタント	**Alibaba Cloud ECS** により提供される日常のメンテナンスタスクの自動化バッチ呼び出しに関する便利な機能です。
クラウドアシスタントクライアント	クライアント	**ECS** インスタンスにインストールされたクラインアントプログラムです。 **Alibaba Cloud Service** に含まれるタスク処理です。 **ECS** インスタンスのすべての操作は、クライアントを使用して実行されます。

用語	一般的な名称	説明
コマンド	コマンド	**Shell** スクリプトなどの **ECS** インスタンスで呼び出される指定されたのコマンド。
1 回限りの呼び出し	呼び出し	**1** 回のみ**1** つまたは複数のインスタンスでコマンドが実行される (つまり、呼び出される) こと (`Invocation`)。
定期的な呼び出し	時間による呼び出し	**1** つまたは複数のインスタンス上でコマンドを呼び出す際、呼び出しシーケンス(つまり、時間周期)を指定することができます。
呼び出しステータス	呼び出しステータス	コマンド呼び出しステータス間の関係性 詳しくは、「コマンド呼び出しのライフサイクル」をご参照ください。

制限

クラウドアシスタントクライアントには、以下のような制限があります。

- 管理者として、クラウドアシスタントクライアントをインストールおよび管理する必要があります。 具体的には、**Linux** インスタンス管理者は **"root"**、 **Windows** インスタンス管理者は **"administrator"** です。
- それぞれの **Alibaba Cloud** リージョンで、最大 **100** のクラウドアシスタントクライアントコマンドを作成できます。
- それぞれの **Alibaba Cloud** リージョンで、**1** 日に最大 **100** 個のクラウドアシスタントクライアントコマンドを実行できます。
- クラウドアシスタントサービスでの **ECS** インスタンスは、すべての **Alibaba Cloud** リージョンで利用可能です。
- 時間による呼び出しコマンドに関して、`Timed` の間隔は **10** 未満には設定できません。
- **Base64** でエンコードされた **Bat** または **PowerSell** スクリプトのサイズ、またはシェルスクリプトのサイズは、**16 KB** 超えることはできません。
- 対象となる **ECS** インスタンスは **[Running]** (`Running`) ステータスである必要があります。
- 現在、クラウドアシスタントクライアントは以下のオペレーティングシステムからのみサポートされます**: Windows Server 2008/2012/2016**、**Ubuntu 12/14/16**、**CentOS 5/6/7**、**Debian 7/8/9**、**RedHat 5/6/7**、**SUSE Linux Enterprise Server 11/12**、**OpenSUSE**、**Alibaba Cloud Linux** および **CoreOS**。

クラウドアシスタントの使用方法

クライアントの使用前に、お使いの **ECS** インスタンスにクラウドアシスタントクライアントをインストールする必要があります。

API を介して使用できます。 詳しくは、『自動的なインスタンスの管理』をご参照ください。

コマンド呼び出しのライフサイクル

インスタンスの実行時、コマンドは以下のようなステータスになります。

表 1-2 : 1 つのインスタンスで実行されたコマンドのステータス

コマンドステータス	API ステータス	説明
Being executed	`Running`	コマンドが実行中です。
Stopped	`Stopped`	実行中にコマンドが停止されました。
Execution finished	`Finished`	コマンドの呼び出しが終了しました。 ただし、呼び出しが成功したことを示しているわけではありません。 コマンド処理の実際の出力 (`Output`) を確認することで、呼び出しが成功したかどうかを確認できます。
Execution failed	`Failed`	タイムアウト時間 (`Timeout`) に達した際に、コマンドの呼び出しが終了しませんでした。

一括または定期的な実行の管理を簡単に行うために、全体の呼び出しステータス、インスタンスの呼び出しステータス および 呼び出し記録ステータスの観点からコマンド実行のライフサイクルを管理することができます。 異なるレベル間の関係は、以下の図のようになります。

図 1-1 : 呼び出しステータス間の関係

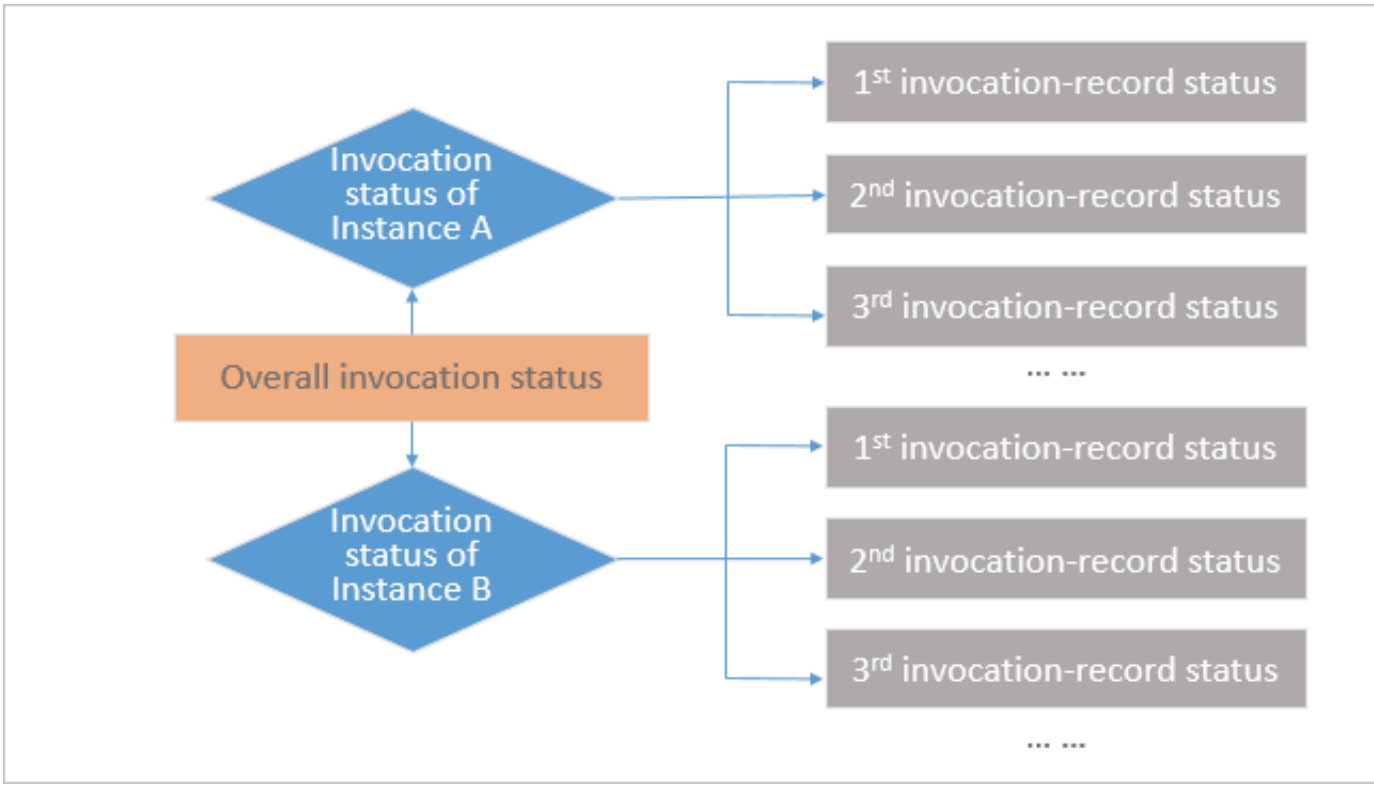


表 1-3 : 1 回限りのバッチ実行のライフサイクル

ステータス	コマンド呼び出し	表示されるステータス
全体の呼び出しステータス	すべてのインスタンスの呼び出しステータスが **[Finished]** (`Finished`)。	**Finished**
	いくつかのインスタンスの呼び出しステータスが **[Finished]** (`Finished`) で、 他のインスタンスの呼び出しステータスが **[Stopped]** (`Stopped`)。	
	すべてのインスタンスのステータスが **[Failed]** (`Failed`)。	**Failed**
	すべてのインスタンスの呼び出しステータスが **[Stopped]** (`Stopped`)。	**Stopped**

ステータス	コマンド呼び出し	表示されるステータス
	すべてのインスタンスの呼び出しステータスが **[Running]** (`Running`)、またはいくつかのインスタンスのステータスが `Running` (`Running`)。	**Running**
	いくつのインスタンスの呼び出しステータスが **[Failed]** (`Failed`)。	**Partially failed**
インスタンス呼び出しステータス	**1** 回限りのバッチ化された実行は単発の操作のため、インスタンス呼び出しステータスは、呼び出し記録ステータスと同一のものになります。	
呼び出し記録ステータス	「インスタンスで実行されたコマンドのステータス」表をご参照ください。	

3 つの **ECS** インスタンスを例にとります。 複数のインスタンスで **1** 回限りの実行中の全体の呼び出しステータスとインスタンス呼び出しステータスの間の関係は以下の図のようになります。

図 1-2 : 1 回限りのバッチ化実行のライフサイクル

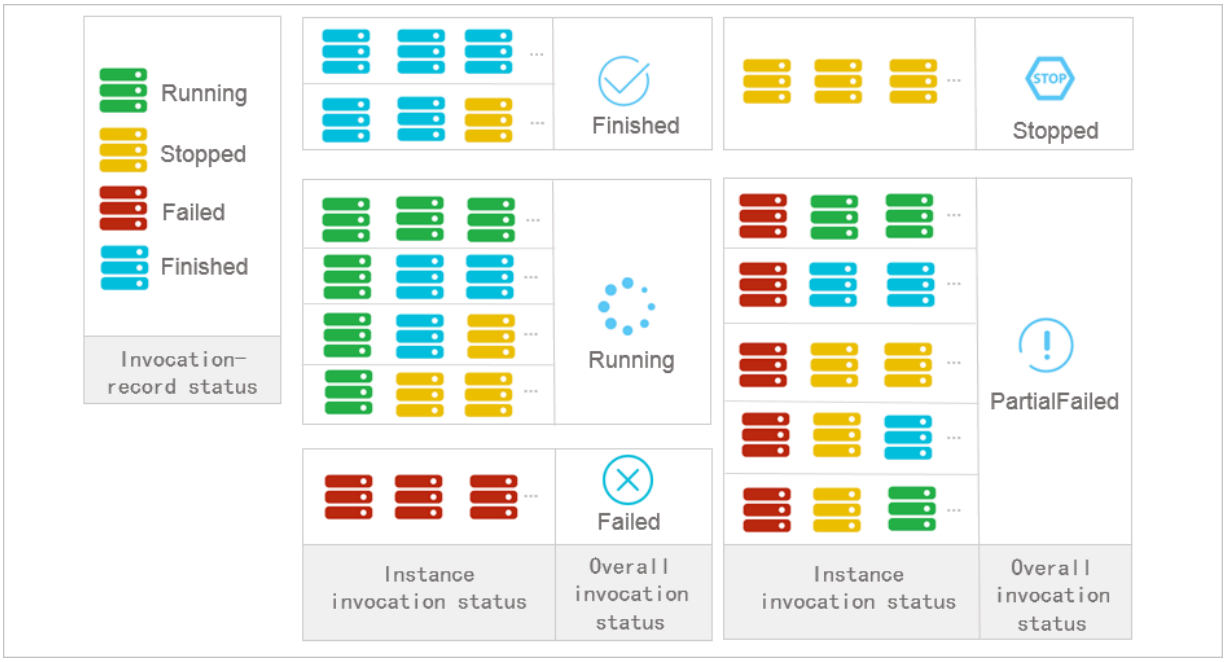


表 1-4 : 定期的な実行のライフサイクル

ステータス	説明
全体の呼び出しステータス	すべてのインスタンスの呼び出しを停止していなくても、全体の呼び出しステータスが **[Running]** (`Running`) のままの場合。

ステータス	説明
インスタンス呼び出しステータス	呼び出しを停止していなくても、インスタンス呼び出しステータスが **[Running]** (`Running`) のままの場合。
呼び出し記録ステータス	「インスタンスで実行されたコマンドのステータス」表をご参照ください。

参考資料

クラウドアシスタントクライアント

コンソール操作

- コマンドの作成
- コマンドの実行
- 呼び出し結果および呼び出しステータスの紹介
- コマンドの管理

API 操作

- 『*CreateCommand*』: コマンドの作成
- 『*InvokeCommnad*』: 対象となる **ECS** インスタンスでの作成したコマンドの実行
- 『*DescribeInvocations*』: コマンドの実行ステータスの照会
- 『*DescribeInvocationResults*』: コマンドの実行結果、つまり、特定の **ECS** インスタンスでの実際の出力情報 (`Output`) の照会
- 『*StopInvocation*』: 実行中のコマンド処理の停止
- 『*ModifyCommand*』: 作成したコマンド内容の変更
- 『*DescribeCommands*』: 作成したコマンドの照会
- 『*DeleteCommand*』: 作成したコマンドの削除

1.2 クラウドアシスタントの使用

1.2.1 コマンドの作成

クラウドアシスタンスコマンドを使って、**ECS** インスタンスのルーティンタスクを実行できます。たとえば、自動メンテナンススクリプト、ポーリング処理、ユーザーパスワードの再設定、ソフトウェアのインストールと削除、アプリケーションの更新、修正プログラムのインストールなどです。 コマンドのタイプは **Windows** の場合は **PowerShell**、バッチファイル**(bat)**、**Liinux** の場合は **Shell**です。

制限

- 各 **Alibaba Cloud** リージョンでサポートできるクラウドアシスタントコマンドは**100**までです。
- **Base64** で エンコードされたスクリプトは 最大**16KB** です。

コマンドの作成

ECS コンソールにコマンドを作成するには、以下の手順に従います。

1. ログインします。
2. 左側のナビゲーションウィンドウで、**[Cloud Assistant]** を選択します。
3. ターゲットリージョンを選択します。
4. **[**コマンドを作成**]** をクリックし、以下の操作を行います。
   a. **[**コマンド名**]** を入力する。たとえば、「**HelloECS**」。
   b. **[**コマンドの説明**]** を入力する。例えば「**UserGuide**」。
   c. アイコンをクリックし、ドロップダウンリストからコマンドタイプを選択します。

      Windows インスタンスの場合、**Bat** か **PowerShell** を選びます。 **Linux** インスタンスの場合、**Shell** を選びます。
   d. 次のように、コマンド内容を入力するか貼り付けます

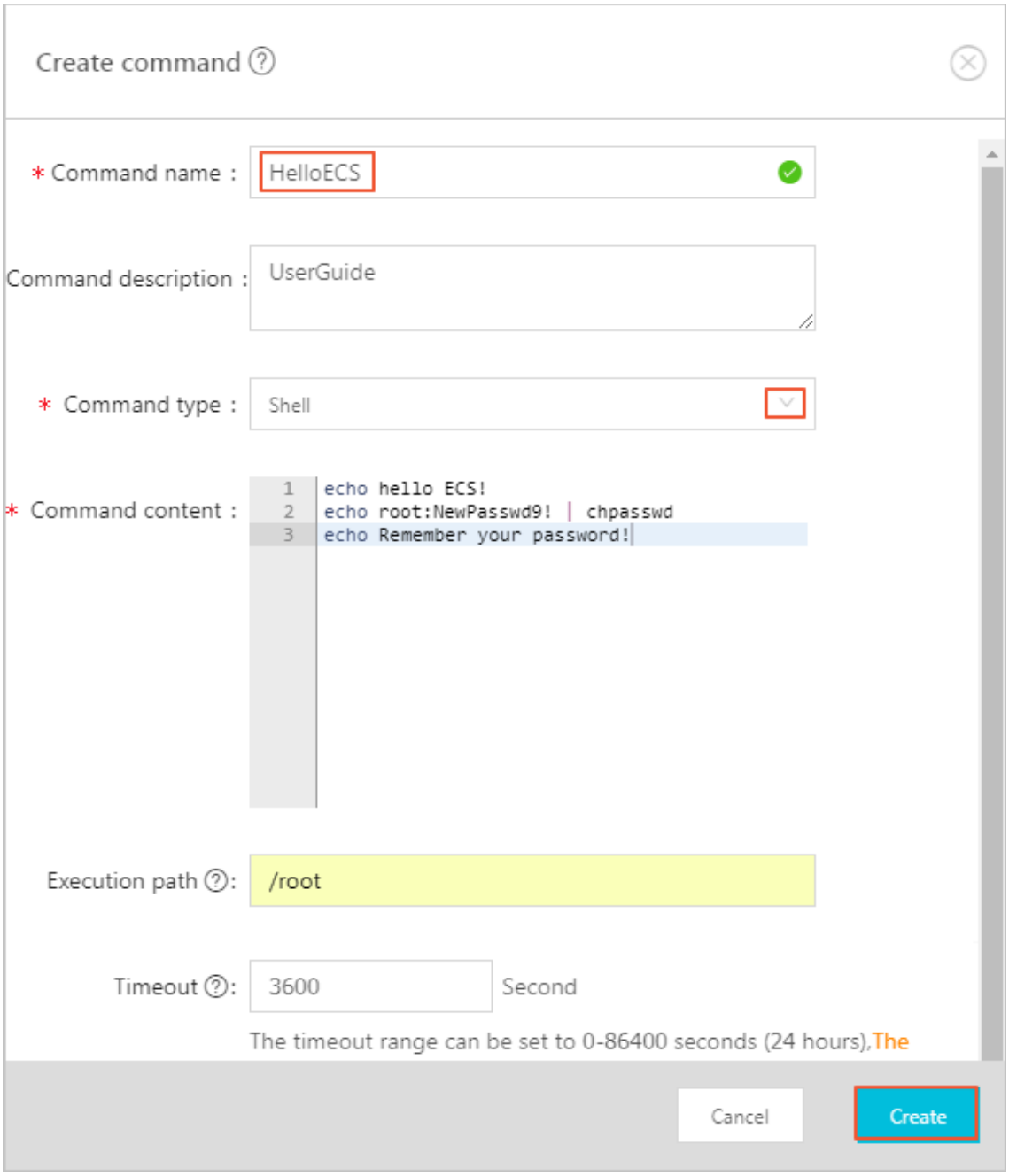
      ```
      echo hello ECS!
      echo root:NewPasswd9! | chpasswd
      echo Remmember your password!
      ```

   e. コマンドの **[**実行パス**]** を決めます。 **Bat** と **PowerShell** コマンドでは、デフォルトでクラウドアシスタントクライアントが保存されているディレクトリに設定されます。たとえ

ば、`C:\ProgramData\aliyun\assist\$(version)`。 さらにシェルコマンドはデフォルトで直接 `/root` ディレクトリに設定されます。

f. コマンドの最大タイムアウト時間 (秒) を設定します。 デフォルト値は **3,600** です。 コマンドがタイムアウトになると、コマンド処理は強制的に終了します。

g. コマンドの詳細を確認し、**[作成]** をクリックします。



ESC API *[CreateCommand]* を使ってクラウドアシスタントコマンドを作成することもできます。

次のステップ

コマンドの呼び出し

1.2.2 コマンドの実行

一つ以上のインスタンスでクラウドアシスタントコマンドを実行できます。 それぞれのインスタンスで実行されるクラウドアシスタントコマンドの実行ステータスと対応する結果は他のインスタンスの実行結果に影響を与えることはありません。 また、必要に応じてそれぞれのクラウドアシスタントコマンドでの実行間隔を設定することもできます。

制限

- 各 **Alibaba Cloud** リージョンでは **1** 日あたり最大 **500** のクラウドアシスタントコマンドを実行することができます。
- 一度に最大 **50** のインスタンスでコマンドを実行することができます。
- 対象となるインスタンスのステータスは、[実行中]でなければなりません。
- 対象となるインスタンスには、クラウドアシスタントクライアント がインストールされていなければなりません。
- 対象となるインスタンスのネットワークタイプは、*VPC* 接続でなくてはなりません。
- クラウドアシスタントコマンドの実行時間は **10** 秒以上となります。
- コマンド実行の時刻設定は、**ECS** インスタンスから取得したシステム時刻に基づき中国の標準時（**UTC +08:00**) で行います。 **ECS** インスタンスの時刻や時間帯が要求に合致しているかどうか確認します。

コマンドの実行

ECS コンソール上でクラウドアシスタントコマンドを実行するには、以下の手順に従います。

1. ログインします。
2. 左側のナビゲーションウィンドウで、[クラウドアシスタント] を選択します。
3. ターゲットリージョンを選択します。
4. 実行するクラウドアシスタントコマンドを検索し、[操作] 列の [実行] を選択します。 表示されたポップアップウィンドウで、次のパラメーターを構成します。

   a. [コマンドを表示] をクリックし、コマンドの内容を確認します。

   b. [インスタンスを選択] をクリックし以下の操作を実行します。

      A. 一つ、または複数のインスタンスを選択します。

      B. 選択したいインスタンスをクリック » して加えます。

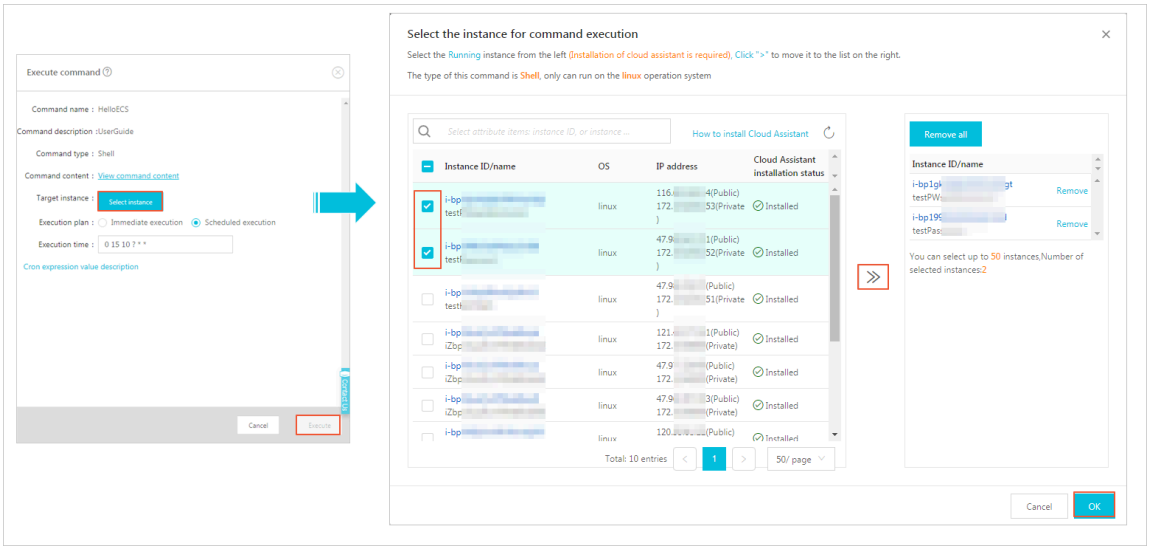
> 注：
> **Windows** インスタンスの場合、**Bat** か **PowerShell** コマンドを選択します。 **Linux** インスタンスの場合、**Shell** コマンドのみ選択できます。 全てのインスタンスには、ク

ラウドアシスタントクライアントが [インストール] されている必要があります。 そうでなければ、選択できません。

C. **[OK]** をクリックします。

c. **[**すぐに実行**]** か **[**スケジュールで実行**]** を選択します。

- **[**すぐに実行**]:** クラウドアシスタントはインスタンス上ですぐに実行します。
- **[**スケジュールで実行**]: cron**式表示が使用され、定期的にコマンドが実行されます。 必要な**[**実行時間**]** を入力します。 詳細は、「*cron*式表示値の説明」をご参照ください。



5. **[**実行**]** をクリックします。

また、**ECS API** *[InvokeCommand]* を用いてクラウドアシスタントコマンドを実行することができます。

コマンド実行の停止

前提条件**:** コマンドは定期的なものか **[**実行中**]**のステータス でなければなりません。

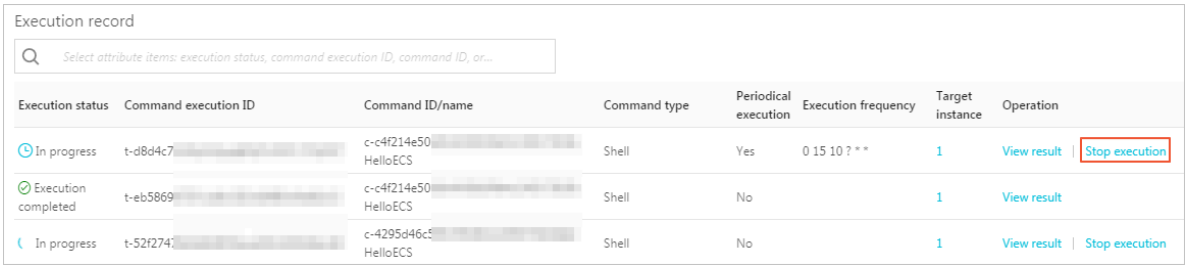
ECS コンソールでコマンドを停止するには、以下の手順に従います。

1. ログインします。

2. 左側のナビゲーションウィンドウで、**[**クラウドアシスタント**]** を選択します。

3. ターゲットリージョンを選択します。

4. [実行記録] エリアで、停止したいコマンドを検索し、[操作] 列から [コマンドを停止] を選択します。



次のステップ

実行結果とステータスを問い合わせる。

1.2.3 実行結果とステータスの照会

コマンドを実行した後、対象の作業が適切に完了したことを確認することを推奨します。 なお、クラウドアシスタントコマンドをコンソールで実行することと、インスタンスにログインしてコマンドを実行することに違いはありません。

前提条件

コマンドは、最低一回は実行します。

コマンド実行結果の表示

コマンド実行結果をECS コンソールに表示するには、以下の手順に従います。

1. 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、[クラウドアシスタント] を選択します。
3. ターゲットリージョンを選択します。
4. [実行記録] エリアで対象としたコマンド実行記録を検索し、[操作] 列から [結果を表示] を選択します。
5. 表示されたポップアップウィンドウで実行記録を選択し、コマンド実施記録をクリック して拡大します。

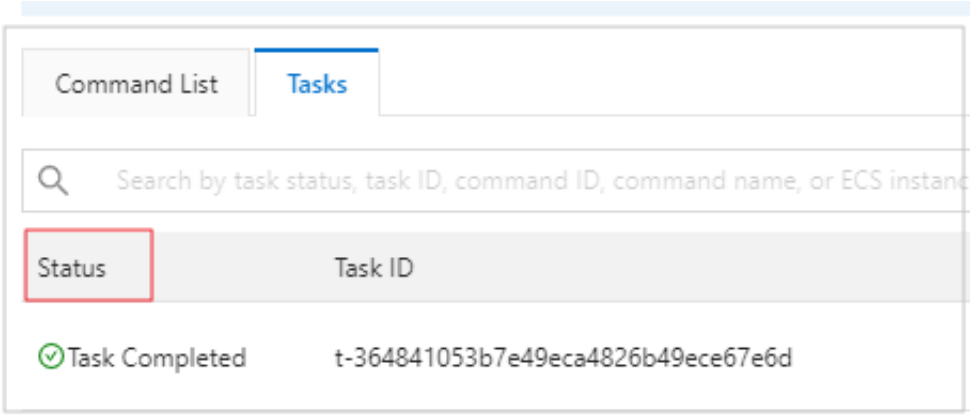
ECS API *DescribeInvocationResults*を使ってコマンドの結果を表示することもできます。

コマンド実行結果の表示

コマンド実行ステータスを ECS コンソールに表示させるには、以下の手順に従います。

1. 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、[クラウドアシスタント] を選択します。

3. ターゲット リージョンを選択します。
4. [実行記録] エリアで、対象となるコマンド実行記録を検索し、[実行ステータス] 列にコマンド実行ステータスを表示させます。



ECS API *DescribeInvocations*を使ってコマンドの実行ステータスを表示することもできます。

コマンド呼び出しのライフサイクル

インスタンスの実行時、コマンドは以下のようなステータスになります。

表 1-5 : 1 つのインスタンスで実行されたコマンドのステータス

コマンドステータス	API ステータス	説明
Being executed	`Running`	コマンドが実行中です。
Stopped	`Stopped`	実行中にコマンドが停止されました。
Execution finished	`Finished`	コマンドの呼び出しが終了しました。 ただし、呼び出しが成功したことを示しているわけではありません。 コマンド処理の実際の出力 (`Output`) を確認することで、呼び出しが成功したかどうかを確認できます。
Execution failed	`Failed`	タイムアウト時間 (`Timeout`) に達した際に、コマンドの呼び出しが終了しませんでした。

一括または定期的な実行の管理を簡単に行うために、全体の呼び出しステータス、インスタンスの呼び出しステータス および 呼び出し記録ステータスの観点からコマンド実行のライフサイクルを管理することができます。 異なるレベル間の関係は、以下の図のようになります。

図 1-3 : 呼び出しステータス間の関係

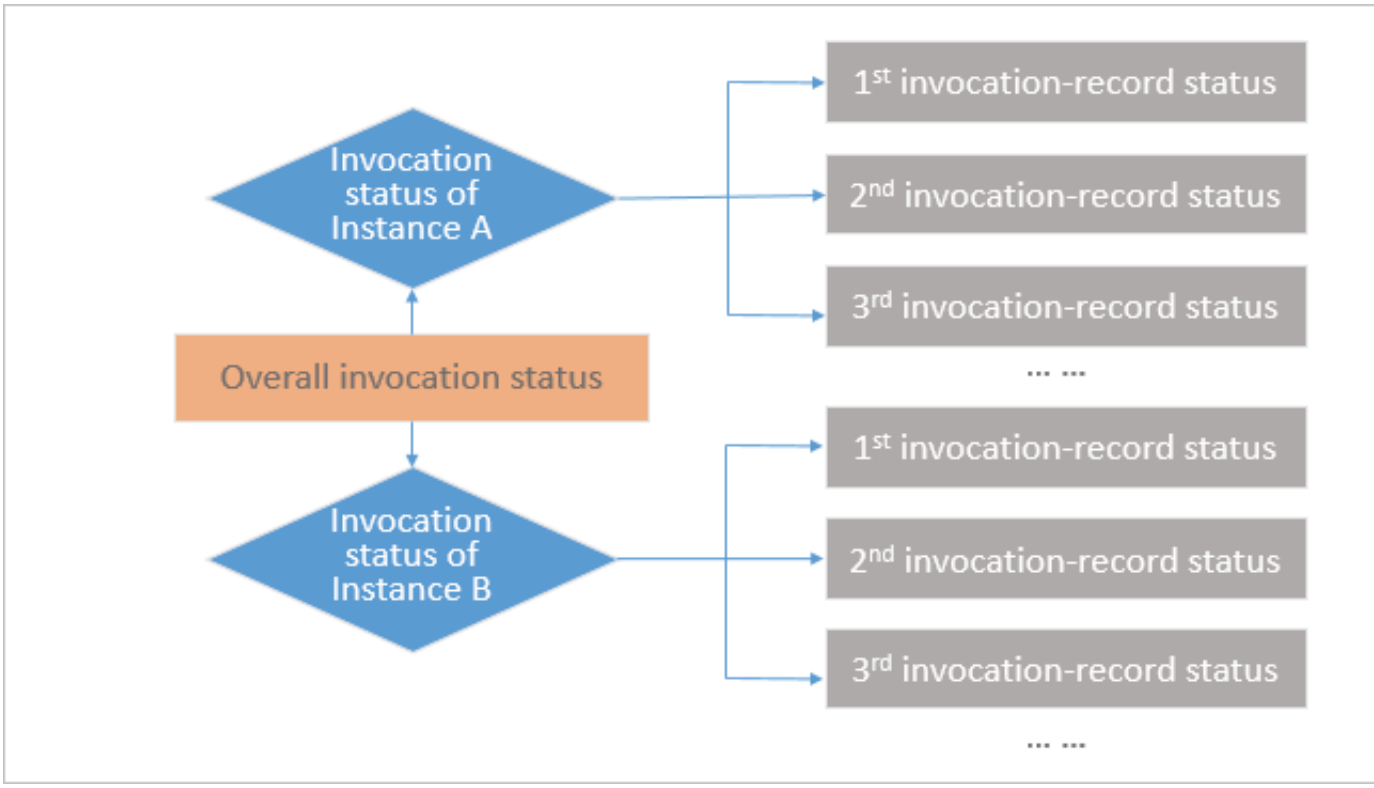


表 1-6 : 1 回限りのバッチ実行のライフサイクル

ステータス	コマンド呼び出し	表示されるステータス
全体の呼び出しステータス	すべてのインスタンスの呼び出しステータスが **[Finished]** (`Finished`)。	**Finished**
	いくつかのインスタンスの呼び出しステータスが **[Finished]** (`Finished`) で、 他のインスタンスの呼び出しステータスが **[Stopped]** (`Stopped`)。	
	すべてのインスタンスのステータスが **[Failed]** (`Failed`)。	**Failed**
	すべてのインスタンスの呼び出しステータスが **[Stopped]** (`Stopped`)。	**Stopped**

ステータス	コマンド呼び出し	表示されるステータス
	すべてのインスタンスの呼び出しステータスが **[Running]** (`Running`)、またはいくつかのインスタンスのステータスが `Running` (`Running`)。	**Running**
	いくつのインスタンスの呼び出しステータスが **[Failed]** (`Failed`)。	**Partially failed**
インスタンス呼び出しステータス	**1** 回限りのバッチ化された実行は単発の操作のため、インスタンス呼び出しステータスは、呼び出し記録ステータスと同一のものになります。	
呼び出し記録ステータス	「インスタンスで実行されたコマンドのステータス」表をご参照ください。	

3 つの **ECS** インスタンスを例にとります。 複数のインスタンスで **1** 回限りの実行中の全体の呼び出しステータスとインスタンス呼び出しステータスの間の関係は以下の図のようになります。

図 1-4 : 1 回限りのバッチ化実行のライフサイクル

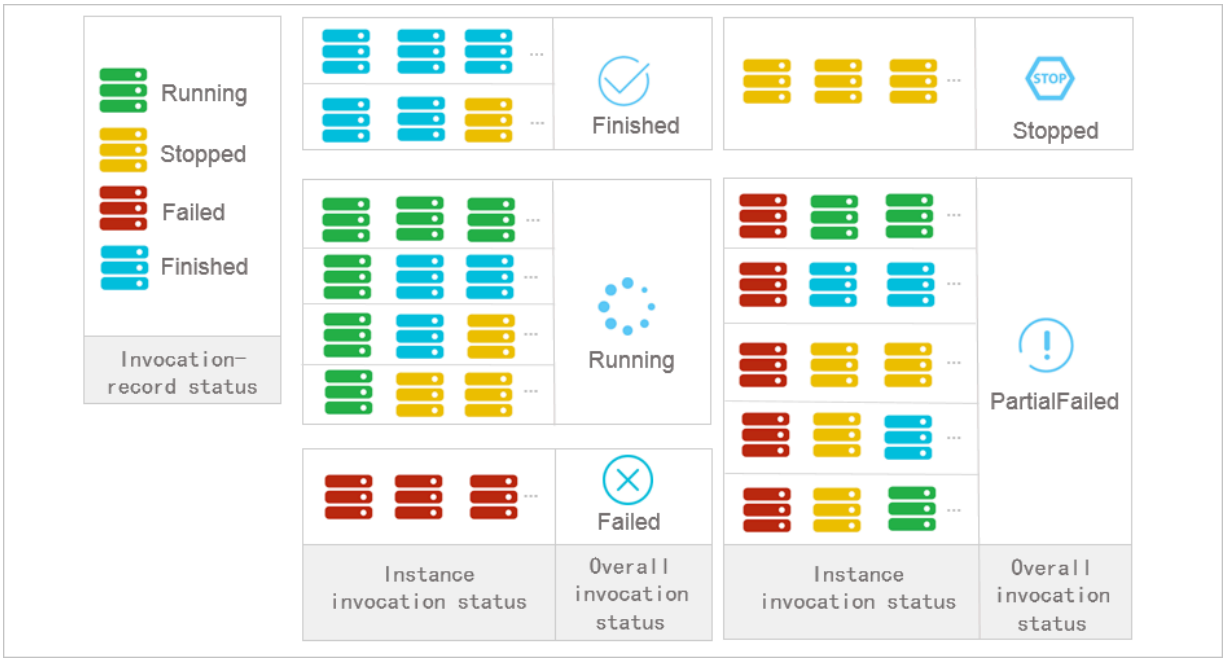


表 1-7 : 定期的な実行のライフサイクル

ステータス	説明
全体の呼び出しステータス	すべてのインスタンスの呼び出しを停止していなくても、全体の呼び出しステータスが **[Running]** (`Running`) のままの場合。

ステータス	説明
インスタンス呼び出しステータス	呼び出しを停止していなくても、インスタンス呼び出しステータスが **[Running]** (`Running`) のままの場合。
呼び出し記録ステータス	「インスタンスで実行されたコマンドのステータス」表をご参照ください。

1.2.4 コマンドの管理

クラウドアシスタントコマンドの作成後は、コマンド名や説明の設定、コマンドの複製、不要なコマンドの削除をすることができます。

コマンドの名前と説明の変更

クラウドアシスタントコマンドに名前や説明の設定、または変更をするには、以下の手順に従います。

1. ログインします。
2. 左側のナビゲーションウィンドウで、**[クラウドアシスタント]** を選択します。
3. ターゲットリージョンを選択します。
4. マウスをターゲットコマンドに合わせ、表示される アイコンをクリックします。 必要に応じて以下を編集します。
   - **[コマンド名]:** 新しいコマンド名を入力します。
   - **[コマンド説明]:** 新しいコマンドの説明を入力します。
5. **[OK]** をクリックします。

ECS API の *[ModifyCommand]* を用いてコマンド情報を変更することもできます。

コマンドの複製

複製コマンドとは、既存クラウドアシスタントコマンドの新しいバージョンを追加することと同じです。 複製コマンドに対してオリジナルのコマンドのすべての情報を維持する、または新しい名前、説明、タイプ、内容、実行パス、タイムアウト時間を設定することもできます。 コマンドを複製するには、以下の手順に従います。

1. ログインする。
2. 左側のナビゲーションウィンドウで、**[クラウドアシスタント]** を選択します。
3. ターゲットリージョンを選択します。
4. ターゲットとなるクラウドアシスタントコマンドを検索し、**[操作]** リストから **[複製]** をクリックします。

5. [コマンドを複製] のダイアログボックスで、必要に応じ次の手順に従います。

   a. 新しい [コマンド名] を入力します。

   b. 新しい [コマンドの説明] を入力します。

   c. アイコンをクリックし、ドロップダウンリストからコマンドタイプを置き換えます。

      Windows インスタンスの場合、**Bat** か **Power Shell** を選択します。 **Linux** インスタンスの場合は、**Shell** のみ選択可能です。

   d. 新しいコマンド内容を入力するか貼り付けます。

   e. 新しいコマンドの [実行パス] を決めます。 **Bat** や **PowerShell** コマンドのデフォルトの実行パスは、クラウドアシスタントクライアントがインストールされているディレクトリになります。たとえば、`『C:\ProgramData\aliyun\assist\$(version)』` です。**Shell** コマンドのデフォルト実行パスは、`/root` ディレクトリになります。

   f. タイムアウト時間を秒単位で構成します。 デフォルト値は **3,600** です。 作成したコマンドがこのパラメーターで設定された時間内に実行できない場合、タイムアウトとなります。タイムアウトになると、コマンド処理は強制的に終了します。

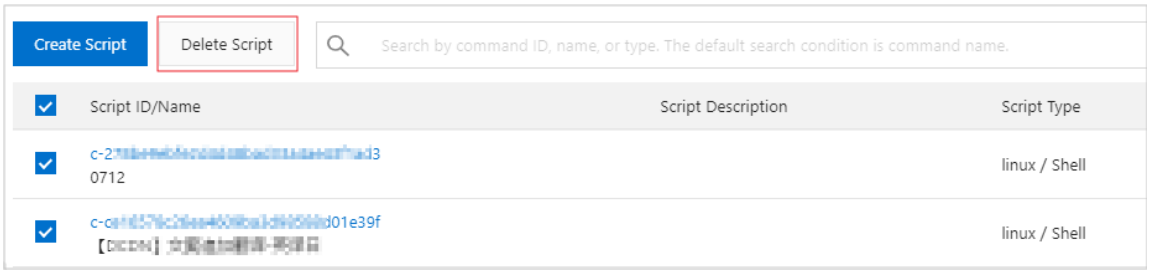
   g. 修正設定を確認し、[作成] をクリックします。

コマンドの削除

Alibaba Cloud リージョン内では、最大 **100** のクラウドアシスタントコマンドを作成することができます。 不要、かつ削除可能なコマンドがあるか定期的にレビューすることを推奨します。**ECS** コンソールでコマンドを削除するには、以下の手順に従います。

1. ログインします。
2. 左側のナビゲーションウィンドウで、[クラウドアシスタント] を選択します。
3. ターゲットリージョンを選択します。
4. 条件によって、以下の操作を実行します。

   - コマンドを一つだけ削除するには、[操作] リストで [削除] を選択します。
   - 複数のコマンドを削除するには、対象となるインスタンスを選んで、[コマンドを削除] をクリックします。



5. [コマンドを削除] のダイアログボックスで、[OK] をクリックします。

また、**ECS API** の *[DeleteCommand]* を用いてコマンドを削除することができます。

1.3 DevOps 練習

1.4 インスタンスの自動管理

ECS インスタンスのメンテナンスは、**ECS** インスタンスを最良の状態に保ち、トラブルシューティングの効率を保証することを目的としています。 しかし、手動によるメンテナンスには膨大な時間と労力がかかります。 この問題に対処するために、 **Cloud Assistant**を使用して、日常のメンテナンス作業の自動化とバッチ処理を行うことができます。 ここでは、**ECS** インスタンスで **Cloud Assistant** コマンドを呼び出して、**ECS** インスタンスを自動的に維持する方法を説明します。

Cloud Assistant は、次の**3**つのコマンドタイプをサポートします。

コマンドタイプ	パラメーター	説明
シェルスクリプト	**RunShellScript**	実行中の **Linux** インスタンス上で実行されているシェルスクリプト。
PowerShell スクリプト	**RunPowerShellScript**	実行中の **Windows** インスタンス上で実行されている **PowerShell** スクリプト。
Bat スクリプト	**RunBatScript**	実行中の **Windows** インスタンス上で実行されている **Bat** スクリプト。

前提条件

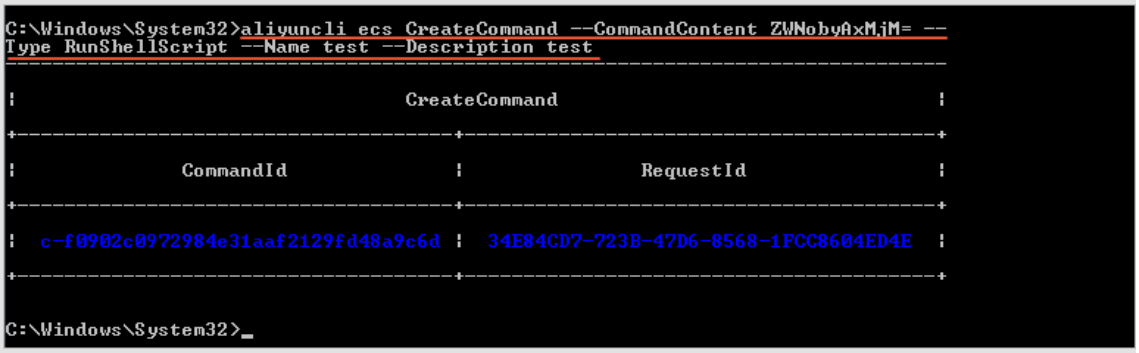
- ターゲット **ECS** インスタンスのネットワーク種別が *VPC* であることを確認する必要があります。
- ターゲット **ECS** インスタンスのステータスは、**[**実行中**]** (`Running`) でなければなりません。
- ターゲット **ECS** インスタンスには、事前に **Cloud Assistant** クライアントがインストールされている必要があります。 詳細は、「*#unique_4*」をご参照ください。
- **PowerShell** コマンドを実行するには、ターゲットの **Windows** インスタンスに **PowerShell** 機能が設定されていることを確認する必要があります。
- 最新バージョンの **Alibaba Cloud CLI** は *GitHub* から入手できます。
- *Alibaba Cloud CLI (Command-Line Interface)* がインストールされていることを確認する必要があります。
- *SDK がアップグレードされている*必要があります。

次の例は、**Cloud Assistant** を使用するために **Alibaba Cloud CLI** で **API** を使用する方法を示しています。 たとえば、**Linux** インスタンス上で `echo 123` コマンドを実行します。

1. ローカルコンピューターの **CMD**、**PowerShell**、または **Shell** で、`aliyuncli ecs CreateCommand --CommandContent ZWNobyAxMjM= --Type RunShellScript --Name test --Description test` を実行して、シェルスクリプトを作成します (`CreateCommand`)。 正常に作成されたら、コマンド **ID** 情報が返されます。

   注：

   - `CommandContent` の `ZWNobyAxMjM =` は、`echo 123` コマンドの **Base 64** コードです。 **Base 64** のエンコードまたはデコードの詳細については、『*Wikipedia - Base64*』をご参照ください。
   - ターゲット **ECS** インスタンスのオペレーティングシステムが **Windows** の場合は、`type` を `RunBatScript` または `RunPowershellScript` に変更します。
   - スクリプトが作成されたら、 `CommandId` が返されます。

2. `aliyuncl ecs InvokeCommand --InstanceId.1 your-vm-instance-id1 --InstanceId.2 your-vm-instance-id2 --CommandId your-command-id --Timed false`を実行して、コマンド *(InvokeCommand)* を実行します。

   注：

   - `InstanceIds` は、**ECS** インスタンス **ID** を示します。 毎回最大 **100** 個の **ECS** インスタンスがサポートされます。
   - `Timed` は、タスクが定期的なものかどうかを示します。 `--Timed True` は、タスクが定期的なものであることを示し、`--Timed False` はその反対を示します。
   - タスクが定期的なもので、 `Timed` パラメーター値が `True` である場合、`Frequency` パラメーターに間隔値を指定する必要があります。 たとえば、 `0 */20 * * * *` は、間隔値が **20** 分であることを示します。 詳細は、「式」をご参照ください。

- 共有の InvokeId は、すべてのターゲット ECS インスタンスに対して返されます。InvokeId を使用して、コマンドの呼び出し状況を確認することができます。

3. 任意。 aliyuncli ecs DescribeInvocations --InstanceId your-vm-instance-id --InvokeId your-invoke-id を実行して、呼び出し状況 *(DescribeInvocations)* を照会します。 具体的には、InvokeId は、ECS インスタンスに対するコマンド呼び出し中に手順 *2* で返された呼び出し ID です。

   返された InvokeStatus 値が Finished である場合は、コマンド処理が完了したことを示しますが、必ずしも期待したほど効果的ではありません。 *DescribeInvocationResults* の Output パラメーターを確認して、特定の呼び出し結果を取得する必要があります。

4. (任意)。 aliyuncli ecs DescribeInvocationResults --InstanceId your-vm-instance-id --InvokeId your-invoke-id を実行して、呼び出し結果 *(DescribeInvocationResults)* を確認します。 具体的には、InvokeId は、ECS インスタンスに対するコマンド呼び出し中に手順 *2* で返された呼び出し ID です。

コマンド *(CreateCommand)* を作成する場合、コマンドに次のリクエストのパラメーターを設定できます。

コマンドのプロパティ	パラメーター	説明
実行ディレクトリ	**WorkingDir**	コマンドが実行される ECS インスタンス内のパスを指定します。 デフォルト値: ・ Linux インスタンスの場合: /root 。 ・ Windows インスタンスの場合: C:\ProgramData\aliyun\assist\$(version)など、Cloud Assistant クライアントプロセスが置かれているパス。

コマンドのプロパティ	パラメーター	説明
タイムアウト期間	**TimeOut**	**ECS** インスタンス上のコマンドの呼び出しタイムアウト値を変更します。 単位は秒です。 何らかの理由でコマンドが失敗した場合、呼び出しはタイムアウトする可能性があり、**Cloud Assistant** クライアントにより、その後コマンドプロセスは強制的に終了します。 パラメーター値は **60** 以上でなければなりません。 値が **60** より小さい場合、タイムアウト値はデフォルトで **60** 秒となります。 デフォルト値**: 3600** ・ **1** 回限りの呼び出し**:** - 呼び出しタイムアウト後は、指定された **ECS** インスタンスのコマンド呼び出しステータス (*DescribeInvocationResults*) は **Failed** となります。 ・ 定期的な呼び出し**:** - 定期的な呼び出しのタイムアウト値は、すべての呼び出しレコードに対して有効です。 - **1** 回の呼び出し操作がタイムアウトすると、呼び出しレコード (*DescribeInvocationResults*) のステータスは **Failed** となります。 - 最後の呼び出しのタイムアウトステータスは、次の呼び出しに影響しません。

クラウドアシスタントを使用するための **Python SDK** のサンプル

Alibaba Cloud SDK を使用して **Cloud Assistant** を使用することもできます。 **Alibaba Cloud SDK** の設定方法の詳細については、*Alibaba Cloud* ユーザー向けの「」をご参照ください。 以下は、**Cloud Assistant** を使用するための **Python SDK** コードです。

```
# coding=utf-8
# if the python sdk is not install using 'sudo pip install aliyun-
python-sdk-ecs'
# if the python sdk is install using 'sudo pip install --upgrade
aliyun-python-sdk-ecs'
# make sure the sdk version is 2.1.2, you can use command 'pip show
aliyun-python-sdk-ecs' to check
import json
import logging
import os
import time
```

```
import datetime
import base64
from aliyunsdkcore import client

from aliyunsdkecs.request.v20140526. CreateCommandRequest import
CreateCommandRequest
from aliyunsdkecs.request.v20140526. InvokeCommandRequest import
InvokeCommandRequest
from aliyunsdkecs.request.v20140526. DescribeInvocationResultsRequest
import DescribeInvocationResultsRequest

# configuration the log output formatter, if you want to save the
output to file,
# append ",filename='ecs_invoke.log'" after datefmt.
logging.basicConfig(level=logging.INFO,
                    format='%(asctime)s %(filename)s[line:%(lineno)d]
 %(levelname)s %(message)s',
                    datefmt='%a, %d %b %Y %H:%M:%S',filename='
aliyun_assist_openapi_test.log', filemode='w')
#access_key = 'Your Access Key Id'
#acess_key_secrect = 'Your Access Key Secrect'
#region_name = 'cn-shanghai'
#zone_id = 'cn-shanghai-b'

access_key = 'LTAIXXXXXXXXXXX'
acess_key_secrect = '4dZXXXXXXXXXXXXXXXXXXXXXXX'
region_name = 'cn-hangzhou'
zone_id = 'cn-hangzhou-f'

clt = client.AcsClient(access_key, acess_key_secrect, region_name)

def create_command(command_content, type, name, description):
    request = CreateCommandRequest()
    request.set_CommandContent(command_content)
    request.set_Type(type)
    request.set_Name(name)
    request.set_Description(description)
    response = _send_request(request)
    if response is None:
        return None
    command_id = response.get('CommandId')
    return command_id;

def invoke_command(instance_id, command_id, timed, cronat):
    request = InvokeCommandRequest()
    request.set_Timed(timed)
    InstanceIds = [instance_id]
    request.set_InstanceIds(InstanceIds)
    request.set_CommandId(command_id)
    request.set_Frequency(cronat)
    response = _send_request(request)
    invoke_id = response.get('InvokeId')
    return invoke_id;

def get_task_output_by_id(instance_id, invoke_id):
    logging.info("Check instance %s invoke_id is %s", instance_id,
invoke_id)
    request = DescribeInvocationResultsRequest()
    request.set_InstanceId(instance_id)
    request.set_InvokeId(invoke_id)
    response = _send_request(request)
    invoke_detail = None
    output = None
    if response is not None:
```

```
        result_list = response.get('Invocation').get('Invocation
Results').get('InvocationResult')
        for item in result_list:
            invoke_detail = item
            output = base64.b64decode(item.get('Output'))
            break;
        return output;

def execute_command(instance_id):
    command_str = 'yum check-update'
    command_id = create_command(base64.b64encode(command_str), '
RunShellScript', 'test', 'test')
    if(command_id is None):
        logging.info('create command failed')
        return

    invoke_id = invoke_command(instance_id, command_id, 'false', '')
    if(invoke_id is None):
        logging.info('invoke command failed')
        return

    time.sleep(15)

    output = get_task_output_by_id(instance_id, invoke_id)
    if(output is None):
        logging.info('get result failed')
        return

    logging.info("output: %s is \n", output)

# send open api request
def _send_request(request):
    request.set_accept_format('json')
    try:
        response_str = clt.do_action(request)
        logging.info(response_str)
        response_detail = json.loads(response_str)
        return response_detail
    except Exception as e:
        logging.error(e)

if __name__ == '__main__':
    execute_command('i-bp17zhpbXXXXXXXXXXXXX')
```

参考文献

上記の例では、**Alibaba Cloud CLI** と**Cloud Assistant API** *CreateCommand*、*InvokeCommand*、*DescribeInvocations*、および*DescribeInvocationResults*を使用して、**ECS** インスタンスのメンテナンスを自動管理する方法を示しています。 **Cloud Assistant** の他の **API** も使用できます。

- *StopInvocation*: スケジュールされたコマンド処理の中止。
- *ModifyCommand*: コマンドの内容の変更。
- *DescribeCommands*: 使用可能なコマンドの照会。
- *DeleteCommand*: コマンドの削除。

2 モニタリング

2.1 システムイベント

システムイベントとは、予定され記録される ECS リソースのメンテナンスイベントです。 ECS インスタンスで、セキュリティアップデート、無効な操作、サブスクリプションの期限切れ、料金滞納、予期しないエラーなどが検知された時に、システムイベントが発生します。 システムイベントが発生すると、インスタンスが起動、再起動、停止、またはリリースされます。

定期メンテナンスとシステムイベント

ECS インスタンスは、アプリケーションを構築するためのコアコンポーネントです。 ECS インスタンスを選択し起動した後に、設定を開始、アプリケーションをデプロイします。ECS インスタンスが正常に動くことが、ビジネスにとってきわめて重要です。 バックエンドのパフォーマンスと ECS インスタンスのセキュリティを保証するために、Alibaba Cloud は物理サーバーの定期メンテナンスを実施します。 Alibaba Cloud が、ハードウェアとソフトウェアのエラーや物理サーバーの潜在的リスクを見つけるためにインスタンスをスキャンする際、例外が検知され場合にはインスタンスはリアルタイムで健全なサーバーに移行されます。 しかしながら、システムイベントとは異なり、通知は送付されません。 定期メンテナンスが進行中でも、インスタンスには影響ありません。

システムイベントが発生すると、インスタンス上でデフォルトアクションとそのアクションを実行するようにスケジュールされた時間が通知されます。 計画的なシステムイベントの場合、イベントがインスタンスに与える影響と想定される実行ポイントなどの情報が事前に送付されます。 ビジネスへの影響を避けるため、システムイベントを処理する前にデータをバックアップし、着信トラフィックを分散することを推奨します。 不良とされた診断や修復をさらに分析するには、システムイベントが解決された後、過去 1 週間のシステムイベント履歴を照会します。

制限

sn2、sn1、t1、s1、s2、s3、m1、m2、c1、c2、c4、ce4、cm4、n1、n2、e3 など (これらを含むが限定されない) の段階的に廃止されたインスタンスタイプは、システムイベントをサポートしません。 詳細は、「インスタンスタイプファミリー」をご参照ください 。

イベントタイプ

次の表は、ECS システムイベントのタイプを示しています。

カテゴリー	イベントタイプ	パラメーター
スケジュールされた再起動	計画されたシステムメンテナンスまたはセキュリティ更新後にインスタンスが再起動されます。	`SystemMaintenance.Reboot`
予期せぬ再起動	予期していないシステム障害が発生した後でインスタンスが再起動します。	`SystemFailure.Reboot`
	予期しないインスタンスの障害発生後、インスタンスが再起動します。	`InstanceFailure.Reboot`
インスタンスの停止	有効期限が切れると、サブスクリプションインスタンスは停止します。	`InstanceExpiration.Stop`
	支払期限が過ぎると、従量課金インスタンスは停止します。	`AccountUnbalanced.Stop`
インスタンスのリリース	サブスクリプションインスタンスは、期限切れの数日後にリリースされます。	`InstanceExpiration.Delete`
	従量課金インスタンスは、料金滞納の数日後にリリースされます。	`AccountUnbalanced.Delete`

イベントのステータス

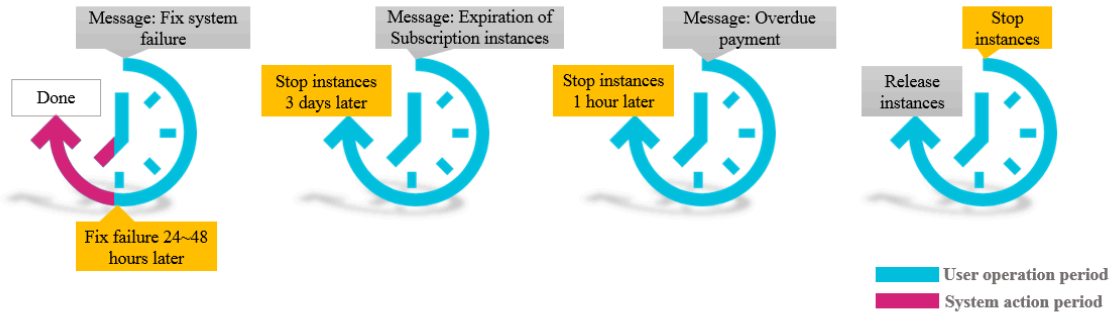
次の表は、ライフサイクル中のシステムイベントのステータスを示しています。

ステータス	ステータス属性	説明
スケジュール済み	中間ステータス	システムイベントはスケジュールされていますが、実行されません。
回避	安定したステータス	「ユーザー操作期間内」 に推奨アクションを事前に実行しました。
実行中	中間ステータス	システムイベントのレスポンス計画が実行されています。
実行済み	安定したステータス	システムイベントが修正されました。
キャンセル	安定したステータス	**ECS** が予定されたシステムイベントをキャンセルしました。
失敗	安定したステータス	システムイベントは修正されていません。

システムイベント期間

システムイベントは、次の 2 つの期間を監視します。

- ユーザー操作期間: システムイベントの開始時刻と予定時刻の間の期間。 通常、システム障害イベントが修正される 24～48 時間前、サブスクリプションインスタンスが停止する 3 日前から、および従量課金インスタンスが停止する 1 時間前に通知が届きます。 アカウントの更新や追加の支払いがなされない場合、15 日後にインスタンスはリリースされます。 この期間に、システムイベントを事前に処理するために推奨される方法を選択します。 また、デフォルトのアクションがトリガーされるまで待つこともできます。
- システムアクション期間: 通常、デフォルトアクションがトリガーされるまで待つと、システムアクション期間が予定された時刻に開始されてから 6 時間以内にシステムイベントが自動的に修正されます。 その後、システムイベントレポートが届きます。



> 注:
> 予定されたシステムイベントのみユーザー操作期間があります。 緊急障害または無効な操作に起因する予期しないシステムイベントには、ユーザー操作期間はありません。 予期しないシステムイベントが発生すると、通知が届きますが、何もすることはありません。 ただし、システムのイベント履歴で、障害診断、原因分析、またはデータ復旧について照会できます。

システムイベントを表示する

システムイベントがスケジュールされている場合、ECS コンソールの [未解決タスク] ボタンは、イベントを確認できるようにハイライトされたタグを表示します。

1. にログインします。
2. <li class="li">左側のナビゲーションウィンドウで [概要] を選択します。
3. 概要ページの右側のナビゲーションウィンドウで、[未解決タスク] を選択します。

4. [未解決タスク] ページには、インスタンス ID、リージョン、実行ステータス、システムタスク、推奨されるユーザー操作、および操作用のボタンのリストが表示されます。 あるいは、[操作] 列で推奨されるユーザー操作を選択して、システムイベントを処理します。

API 操作: *[DescribeInstancesFullStatus]* を呼び出して、システムイベントを表示します。

システムイベントを表示する

すべてのイベントページで、前週のシステムイベントの履歴で、障害の診断や障害の再現を照会します。

1.
2. 左側のナビゲーションウィンドウで、[概要] を選択します。
3. 概要 ページの右側にあるナビゲーションウィンドウから [未解決のタスク] を選択します。
4. [全てのタスク] をクリックし、全てのタスク ページで [システムタスク] > [インスタンス] をクリックすると、インスタンス ID 、イベントタイプ、リージョン、タスクステータスのリストを確認します。

API 操作: システムのイベント履歴を表示するには、*DescribeInstanceHistoryEvents*を呼び出します。

システムイベントの提案

システムイベントで **Alibaba Cloud ECS** の基盤となるコンポーネントを把握します。 システムイベントに基づいてインスタンスの **O&M** を最適化できます。 システムイベントに適切に対処するには以下の手順を実行することを推奨します。

イベントタイプ	パラメーター	推奨法
保留中のシステムメンテナンス後にインスタンスが再起動する。	**SystemMaintenance.Reboot**	ユーザー操作期間内に以下のうちのいずれかの方法を使います。 ・ **ECS**コンソールで 「インスタンスの再起動」 ・ **API** 「*RebootInstance*」 の呼び出し 注： インスタンスまたはインスタンスリストから実行されたインスタンスの再起動は、このタイプのシステムイベントには影響しません。 データを安全にバックアップするために、接続されているディスクの 「スナップショット」「*CreateSnapshot*」 を作成することを推奨します。

イベントタイプ	パラメーター	推奨法
予期しないシステム障害が発生した後でインスタンスが再起動する。	**SystemFailure.Reboot**	通知を受け取ると、インスタンスが再起動されます。 イベント後にインスタンスやアプリケーションの回復を確認することを推奨します。
予期しないインスタンスの障害が起きると、インスタンスが再起動する。	**InstanceFailure.Reboot**	通知を受け取ると、インスタンスが再起動されます。 次のことを推奨します。 ・ インスタンスとアプリケーションの回復を確認します。 ・ インスタンスクラッシュの原因を分析して、今後同様のイベントが発生しないようにします。
サブスクリプションインスタンスは、有効期限が切れると停止する。	**InstanceExpiration.Stop**	*[インスタンスを更新]* するか、インスタンスが停止するのを待ちます。
従量課金のインスタンスは、支払期限が切れると停止する。	**AccountUnbalanced.Stop**	アカウントに追加入金するか、インスタンスが停止するまで待ちます。
サブスクリプションインスタンスは、有効期限が切れたためにリリースされる。	**InstanceExpiration.Delete**	「インスタンスを更新」 するか、インスタンスがリリースされるのを待ちます。
支払期限が過ぎると、従量課金インスタンスがリリースされる。	**AccountUnbalanced.Delete**	アカウントに追加入金するか、インスタンスがリリースされるまで待ちます。

3 インスタンス状態のモニタリング

4 操作エラーのトラブルシューティング

4.1 コンソール出力とスクリーンショット

ECS インスタンスは、仮想化されたクラウドベースのサービスで、ディスプレイデバイスに接続することも、モバイルスナップショットを作成することもできません。 しかしながら、インスタンスのコンソール出力は最後の立上げ時、再起動時、シャットダウンイベントの時にキャッシュされます。 しかも、リアルタイムでインスタンスのスクリーンショットも取得します。 このような機能を用いて、オペレーティングシステムの例外診断、異常な再起動、インスタンスに接続できないなどのインスタンスの障害を分析、トラブルシューティングすることを推奨します。

制限

- **Windows Server** のイメージを実行しているインスタンスでは、コンソール出力を取得することはできません。
- 「段階的に廃止されたインスタンス」では、コンソール出力やスナップショットを取得することはできません。
- **2018** 年 **1** 月 **1** 日以前に作成されたインスタンスのコンソール出力やスクリーンショットを取得することはできません。

前提条件

インスタンスは、実行中 ステータスでなければなりません。 詳細については、「概要」をご参照ください。

手順

インスタンスの詳細 ページ、インスタンス のリストページ、または **API** を呼び出すことで、インスタンスのコンソール出力やスクリーンショットを表示します。

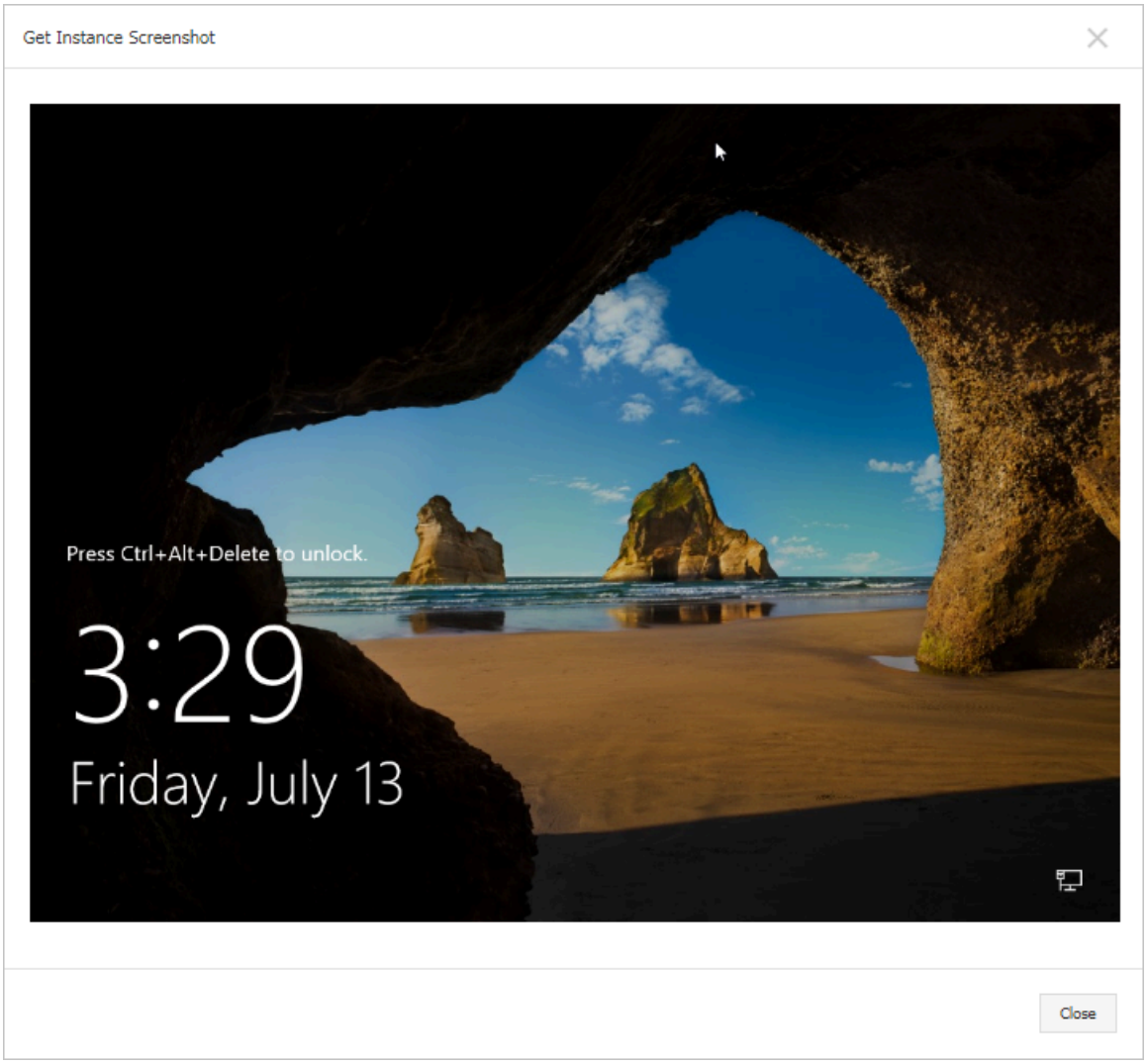
インスタンス詳細ページでの操作

1. *ECS* 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、 **[**インスタンス**]** をクリックします。
3. ターゲットリージョンを選択します。
4. トラブルシューティングを行うターゲットインスタンスのインスタンス **ID** をクリックして、インスタンス詳細 ページに移動します。

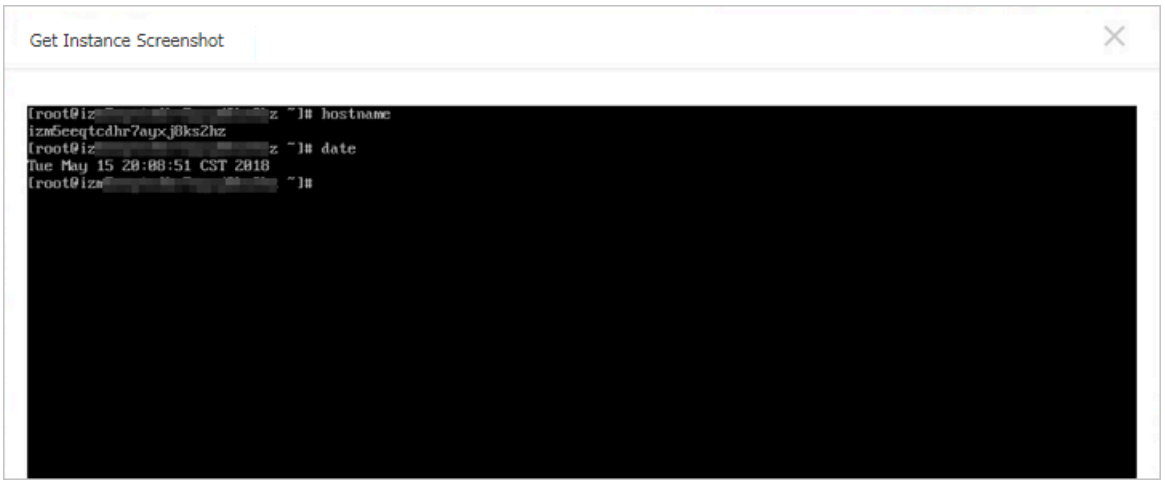
5. [詳細] > [インスタンスのスクリーンショットを取得] の順にクリックし、スクリーンショットを表示します。 または、[詳細] > [インスタンスコンソール出力を取得] の順にクリックし、ルートコンソールをモニタリングします。

6. インスタンスのスクリーンショットやコンソール出力を確認します。

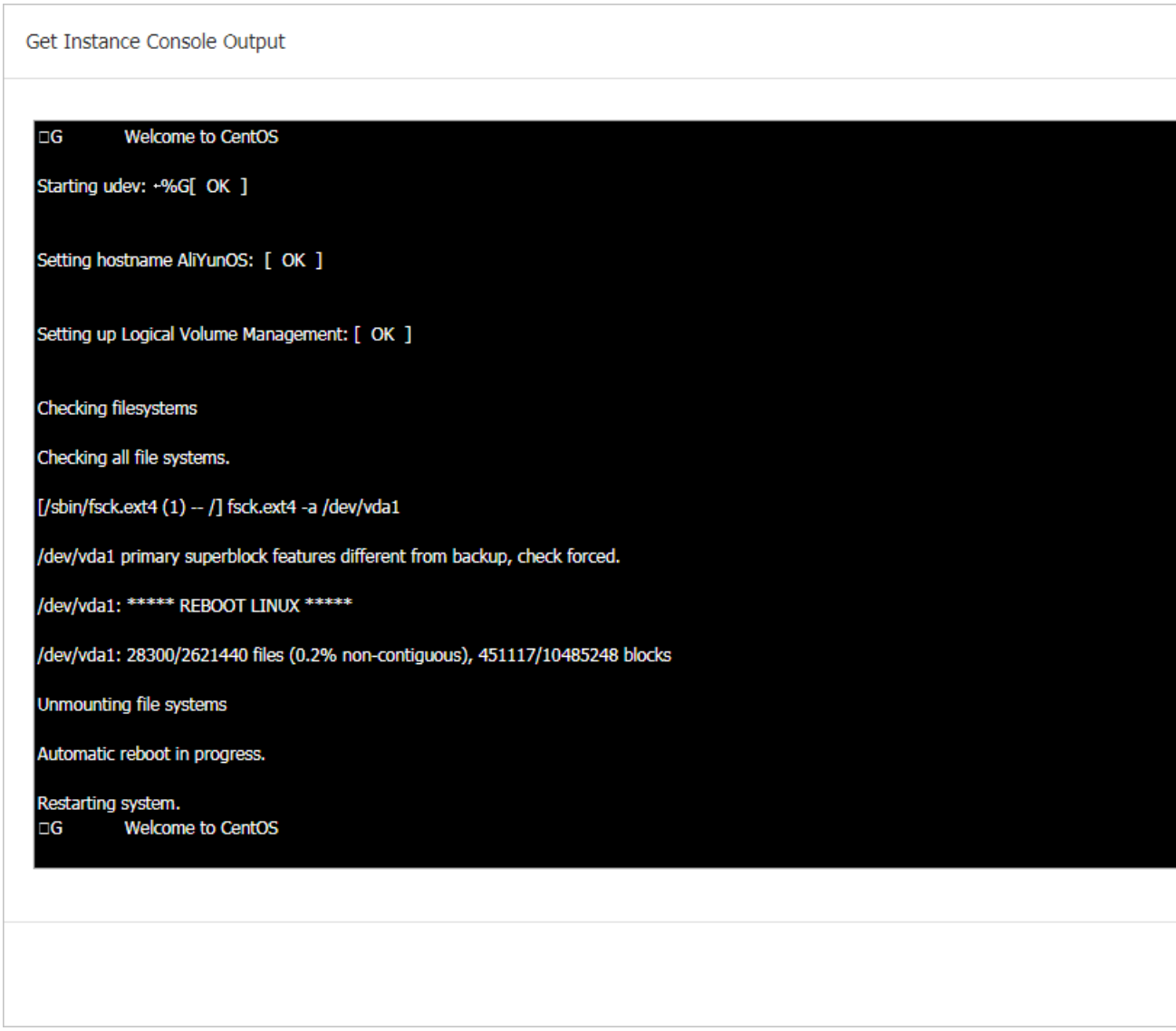
- **Windows** インスタンスのスクリーンショット例



- **Linux** インスタンスのスクリーンショット例



- **Linux** インスタンスのコンソール出力例



インスタンスのリストページでの操作

1. *[ECS 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、[インスタンス] をクリックします。
3. ターゲットリージョンを選択します。
4. トラブルシューティングを行うターゲットインスタンスを検索します。
5. [操作] 列で、[詳細] > [操作とトラブルシューティング] > [インスタンスのスクリーンショットを取得] の順でクリックし、スクリーンショットを表示します。 または、[詳細] > [操作とトラブルシューティング] > [インスタンスコンソール出力の取得] の順にクリックし、ルートコンソールをモニタリングします。
6. インスタンスのスクリーンショットやコンソール出力を確認します。

API 操作

・ インスタンスのスクリーンショット: *GetInstanceScreenshot*

・ インスタンスのコンソール出力: *GetInstanceConsoleOutput*

次のステップ

他のトラブルシューティング方法については、「*ping* パケット損失あるいは *ping* エラーのリンクテストツール」、をご参照ください。