



アプリを保護します Astra Control Center

NetApp
November 21, 2023

目次

アプリを保護します	1
保護の概要	1
Snapshot とバックアップでアプリケーションを保護	1
アプリケーションのリストア	5
アプリケーションのクローン作成と移行	7
アプリケーション実行フックを管理します	9

アプリを保護します

保護の概要

Astra Control Center を使用して、アプリケーションのバックアップ、クローン、スナップショット、および保護ポリシーを作成できます。アプリケーションをバックアップすることで、サービスや関連データを可能な限り利用できるようになります。災害時にバックアップからリストアすることで、アプリケーションと関連データを最小限の中断で完全にリカバリできます。バックアップ、クローン、Snapshot を使用すると、ランサムウェアや偶発的なデータ損失、環境障害などの一般的な脅威からデータを保護できます。"[Astra Control Center で使用可能なデータ保護の種類と、それらを使用するタイミングについて説明します](#)"。

アプリケーション保護のワークフロー

次のワークフロー例を使用して、アプリケーションの保護を開始できます。

[1つ] すべてのアプリケーションをバックアップ

アプリケーションをすぐに保護するには、次の手順を実行します。"[すべてのアプリケーションの手動バックアップを作成する](#)"。

[2つ] 各アプリケーションの保護ポリシーを設定します

将来のバックアップとスナップショットを自動化するには、"[各アプリケーションの保護ポリシーを設定します](#)"。たとえば、週単位のバックアップと日単位の Snapshot をそれぞれ 1 カ月ずつ保持して開始できます。手動バックアップやスナップショットよりも、保護ポリシーを使用してバックアップとスナップショットを自動化することを強く推奨します。

[3つ] オプション：保護ポリシーを調整します

アプリとその使用パターンが変化したら、必要に応じて保護ポリシーを調整して、最適な保護を実現します。

[4.] 災害が発生した場合は、アプリケーションをリストアします

データ損失が発生した場合は、を使用してリカバリできます "[最新のバックアップをリストアしています](#)" まず、各アプリケーションについて説明します。その後、最新の Snapshot をリストアできます（使用可能な場合）。

Snapshot とバックアップでアプリケーションを保護

自動保護ポリシーまたはアドホックベースを使用してスナップショットやバックアップを作成することで、アプリケーションを保護します。Astra の UI またはを使用できます "[Astra Control API](#)" アプリを保護します。



Helm を使用してアプリケーションを展開する場合、Astra Control Center には Helm バージョン 3 が必要です。Helm 3（または Helm 2 から Helm 3 にアップグレード）を使用して展開されたアプリケーションの管理とクローニングが完全にサポートされています。Helm 2 で展開されたアプリケーションはサポートされていません。



OpenShift クラスターでアプリケーションをホストするプロジェクトを作成すると、プロジェクト（または Kubernetes ネームスペース）に SecurityContext UID が割り当てられます。Astra Control Center でアプリケーションを保護し、OpenShift でそのアプリケーションを別のクラスターまたはプロジェクトに移動できるようにするには、アプリケーションを任意の UID として実行できるようにポリシーを追加する必要があります。たとえば、次の OpenShift CLI コマンドは、WordPress アプリケーションに適切なポリシーを付与します。

```
OC new-project ワードプレス `OC adm policy add -scc to -group anyuid system:serviceaccounts:wordpress `OC adm policy add -scc to -user Privileged -z default-n wordpress
```

保護ポリシーを設定します

保護ポリシーは、定義されたスケジュールでスナップショット、バックアップ、またはその両方を作成することでアプリケーションを保護します。Snapshot とバックアップを毎時、日次、週次、および月単位で作成し、保持するコピーの数を指定できます。たとえば、保護ポリシーでは、週単位のバックアップと日単位の Snapshot が作成され、1 カ月間はバックアップと Snapshot が保持されます。スナップショットやバックアップを作成する頻度と、それらを保持する期間は、組織のニーズによって異なります。

手順

1. 「* アプリケーション」を選択し、アプリケーションの名前を選択します。
2. [* データ保護 *] を選択します。
3. [保護ポリシーの設定] を選択します。
4. 毎時、日次、週次、および月単位で保持する Snapshot とバックアップの数を選択して、保護スケジュールを定義します。

スケジュールは、毎時、毎日、毎週、および毎月の各スケジュールで同時に定義できます。保持レベルを設定するまで、スケジュールはアクティブになりません。

次の例では、Snapshot とバックアップの保護スケジュールとして、毎時、毎日、毎週、毎月の 4 つを設定します。

Configure protection policy

STEP 1/2: DETAILS

✕

PROTECTION SCHEDULE

Hourly

Every hour on the 0th minute, keep the last 4 snapshots

Daily

Daily at 02:00 (UTC), keep the last 15 snapshots

Weekly

Weekly on Mondays at 02:00 (UTC), keep the last 26 snapshots

Monthly

Every 1st of the month at 02:00 (UTC), keep the last 12 backups

● Hourly

● Daily

● **Weekly**

● Monthly

Select Weekday(s) (optional)

Monday X

Time (UTC) (optional)

02:00

– Snapshots to keep +

26

– Backups to keep +

0

BACKUP DESTINATION

Bucket

ntp-nautilus-bucket-10 - ntp-nautilus-bucket-10

OVERVIEW

Schedule and retention

Define a policy to continuously protect your application on a schedule and configure a retention count to get started.

For select stateful applications, expect I/O to pause for a short time during a backup or snapshot operation.

Read more in [Protection policies](#)

Application

cattle-logging

Namespace

cattle-logging

Cluster

se-openlab-astra-enterprise-05-se-openlab-astra-enterprise-05-mstr-1

Cancel

Review →

5. [* Review (レビュー)] を選択します
6. [* 保護ポリシーの設定 *] を選択します

結果

Astra Control Center は、定義したスケジュールと保持ポリシーを使用して、スナップショットとバックアップを作成し、保持することによって、データ保護ポリシーを実装します。

Snapshot を作成します

オンデマンド Snapshot はいつでも作成できます。

手順

1. 「 * アプリケーション * 」を選択します。
2. 目的のアプリケーションの * アクション * 列のオプションメニューから、 * スナップショット * を選択します。
3. スナップショットの名前をカスタマイズし、 * Review * を選択します。
4. Snapshot の概要を確認し、「 * Snapshot * 」を選択します。

結果

スナップショットプロセスが開始されます。スナップショットは、ステータスが「 * 使用可能 * 」である場合に成功します。この場合、「 * データ保護 * > * スナップショット * 」ページの「 * アクション * 」列に表示されます。

バックアップを作成します

アプリケーションはいつでもバックアップできます。



Astra Control Center の S3 バケットは、使用可能容量を報告しません。Astra Control Center で管理されているアプリケーションのバックアップまたはクローニングを行う前に、ONTAP または StorageGRID 管理システムでバケット情報を確認します。

手順

1. 「* アプリケーション *」を選択します。
2. 目的のアプリケーションの * アクション * 列のオプションメニューから、* バックアップ * を選択します。
3. バックアップ名をカスタマイズする。
4. 既存のスナップショットからアプリケーションをバックアップするかどうかを選択します。このオプションを選択すると、既存の Snapshot のリストから選択できます。
5. ストレージバケットのリストから選択して、バックアップのデスティネーションを選択します。
6. [* Review (レビュー)] を選択します
7. バックアップの概要を確認し、「* Backup *」を選択します。

結果

Astra Control Center は、アプリケーションのバックアップを作成します。



ネットワークに障害が発生している場合や、処理速度が異常に遅い場合は、バックアップ処理がタイムアウトする可能性があります。その結果、バックアップは失敗します。



実行中のバックアップを停止する方法はありません。バックアップを削除する必要がある場合は、完了するまで待ってから、の手順を実行してください [\[バックアップを削除します\]](#)。失敗したバックアップを削除するには、"[Astra Control API を使用](#)"。



データ保護処理（クローン、バックアップ、リストア）が完了して永続ボリュームのサイズを変更したあと、新しいボリュームのサイズが UI に表示されるまでに最大 20 分かかります。データ保護処理にかかる時間は数分です。また、ストレージバックエンドの管理ソフトウェアを使用してボリュームサイズの変更を確認できます。

Snapshot とバックアップを表示します

アプリケーションのスナップショットとバックアップは、[データ保護 (Data Protection)] タブで表示できます。

手順

1. 「* アプリケーション」を選択し、アプリケーションの名前を選択します。
2. [* データ保護 *] を選択します。

デフォルトでは、Snapshot が表示されます。

3. バックアップのリストを表示するには、「* Backups *」を選択します。

Snapshot を削除します

不要になったスケジュール済みまたはオンデマンドの Snapshot を削除します。

手順

1. 「* アプリケーション」を選択し、アプリケーションの名前を選択します。
2. [* データ保護 *] を選択します。
3. 目的のスナップショットの * アクション * 列のオプションメニューから、* スナップショットの削除 * を選択します。
4. 削除を確認するために「delete」と入力し、「* はい、Snapshot を削除します *」を選択します。

結果

Astra Control Center がスナップショットを削除します。

バックアップを削除します

不要になったスケジュール済みまたはオンデマンドのバックアップを削除します。



実行中のバックアップを停止する方法はありません。バックアップを削除する必要がある場合は、完了するまで待ってから、以下の手順を実行してください。失敗したバックアップを削除するには、["Astra Control API を使用"](#)。

1. 「* アプリケーション」を選択し、アプリケーションの名前を選択します。
2. [* データ保護 *] を選択します。
3. 「* Backups *」を選択します。
4. 目的のバックアップの [* アクション *] 列の [オプション] メニューから、[* バックアップの削除 *] を選択します。
5. 削除を確認するために「delete」と入力し、「* はい、バックアップを削除 *」を選択します。

結果

Astra Control Center はバックアップを削除します。

アプリケーションのリストア

Astra Control を使用すると、スナップショットまたはバックアップからアプリケーションをリストアできます。同じクラスタにアプリケーションをリストアする場合、既存の Snapshot からのリストアは高速です。Astra Control UI またはを使用できます ["Astra Control API"](#) アプリを復元するには、

このタスクについて

- アプリケーションをリストアする前に、アプリケーションのスナップショットを作成するか、バックアップすることを強くお勧めします。リストアに失敗した場合に、Snapshot またはバックアップからクローニングできます。

- Helm を使用してアプリケーションを展開する場合、Astra Control Center には Helm バージョン 3 が必要です。Helm 3（または Helm 2 から Helm 3 にアップグレード）を使用して展開されたアプリケーションの管理とクローニングが完全にサポートされています。Helm 2 で展開されたアプリケーションはサポートされていません。
- 別のクラスタにリストアする場合は、同じ永続的ボリュームアクセスモード（ReadWriteMany など）をクラスタで使用していることを確認してください。デスティネーションの永続ボリュームアクセスモードが異なると、リストア処理は失敗します。
- 名前空間の名前 / ID または名前空間ラベルによる名前空間の制約を持つメンバーユーザーは、同じクラスタ上の新しい名前空間、または組織のアカウント内の他のクラスタに対して、アプリケーションのクローンまたはリストアを実行できます。ただし、同じユーザが、クローニングまたはリストアされたアプリケーションに新しいネームスペースからアクセスすることはできません。クローンまたはリストア処理によって新しいネームスペースが作成されると、アカウントの管理者 / 所有者はメンバーユーザアカウントを編集し、該当するユーザに新しいネームスペースへのアクセスを許可するロールの制限を更新できます。
- OpenShift クラスタでアプリケーションをホストするプロジェクトを作成すると、プロジェクト（または Kubernetes ネームスペース）に SecurityContext UID が割り当てられます。Astra Control Center でアプリケーションを保護し、OpenShift でそのアプリケーションを別のクラスタまたはプロジェクトに移動できるようにするには、アプリケーションを任意の UID として実行できるようにポリシーを追加する必要があります。たとえば、次の OpenShift CLI コマンドは、WordPress アプリケーションに適切なポリシーを付与します。

```
OC new-project ワードプレス `OC adm policy add -scc to -group anyuid system:serviceaccounts:wordpress `OC adm policy add -scc to -user Privileged -z default-n wordpress
```

手順

1. 「* アプリケーション」を選択し、アプリケーションの名前を選択します。
2. 「* データ保護 *」を選択します。
3. Snapshot からリストアする場合は、* Snapshots * アイコンを選択したままにします。それ以外の場合は、「* Backups *」アイコンを選択してバックアップからリストアします。
4. リストア元のスナップショットまたはバックアップの[* アクション *]列の[オプション]メニューから、[* アプリケーションのリストア *]を選択します。
5. * リストアの詳細 * : リストアされたアプリの詳細を指定します。デフォルトでは、現在のクラスタとネームスペースが表示されます。これらの値をそのままにしておくと、アプリがインプレースで復元され、アプリは以前のバージョンのに戻ります。別のクラスタまたはネームスペースにリストアする場合は、これらの値を変更してください。
 - アプリケーションの名前と名前空間を入力します。
 - アプリケーションのデスティネーションクラスタを選択します。
 - [* Review（レビュー）]を選択します



以前に削除したネームスペースにリストアすると、同じ名前の新しいネームスペースがリストアプロセスで作成されます。以前に削除したネームスペースでアプリケーションを管理する権限を持つユーザは、新しく作成したネームスペースに手動で権限を復元する必要があります。

6. * リストアの概要 * : リストア操作の詳細を確認し、「restore」と入力して、* Restore * を選択します。

結果

Astra Control Center は、指定した情報に基づいてアプリケーションを復元します。アプリケーションをインプレースでリストアした場合、既存の永続ボリュームの内容が、リストアしたアプリケーションの永続ボリュームの内容に置き換えられます。



データ保護処理（クローン、バックアップ、リストア）が完了して永続ボリュームのサイズが変更されたあと、Web UIに新しいボリュームサイズが表示されるまでに最大20分かかります。データ保護処理にかかる時間は数分です。また、ストレージバックエンドの管理ソフトウェアを使用してボリュームサイズの変更を確認できます。

アプリケーションのクローン作成と移行

既存のアプリケーションをクローニングして、同じ Kubernetes クラスタまたは別のクラスタに重複するアプリケーションを作成する。Astra Control Center は、アプリケーションのクローンを作成するときに、アプリケーション構成と永続的ストレージのクローンを作成します。

Kubernetes クラスタ間でアプリケーションとストレージを移動する必要がある場合は、クローニングが役立ちます。たとえば、CI / CD パイプラインや Kubernetes ネームスペース間でワークロードを移動できます。Astra の UI またはを使用できます ["Astra Control API"](#) アプリケーションのクローン作成と移行を実行します。

必要なもの

アプリケーションを別のクラスタにクローニングするには、デフォルトのバケットが必要です。最初のバケットを追加した時点でデフォルトのバケットになります。

このタスクについて

- StorageClass が明示的に設定されたアプリケーションを展開し、そのアプリケーションをクローニングする必要がある場合、ターゲットクラスタには元の StorageClass が指定されている必要があります。明示的に StorageClass を設定したアプリケーションを、同じストレージクラスを使用しないクラスタにクローニングすると、失敗します。
- オペレータが配置した Jenkins CI のインスタンスをクローニングする場合は、永続データを手動でリストアする必要があります。これは、アプリケーションの展開モデルの制限事項です。
- Astra Control Center の S3 バケットは、使用可能容量を報告しません。Astra Control Center で管理されているアプリケーションのバックアップまたはクローニングを行う前に、ONTAP または StorageGRID 管理システムでバケット情報を確認します。
- アプリケーションのバックアップやリストア時に、バケット ID を必要に応じて指定することができます。ただし、アプリケーションのクローニング処理では、定義済みのデフォルトバケットが常に使用されます。クローンのバケットを変更するオプションはありません。どのバケットを使用するかを制御する必要がある場合は、どちらかを選択できます ["バケットのデフォルト設定を変更する"](#) または、を実行します ["バックアップ"](#) その後を押します ["リストア"](#) 個別。
- 名前空間の名前 / ID または名前空間ラベルによる名前空間の制約を持つメンバーユーザーは、同じクラスタ上の新しい名前空間、または組織のアカウント内の他のクラスタに対して、アプリケーションのクローンまたはリストアを実行できます。ただし、同じユーザが、クローニングまたはリストアされたアプリケーションに新しいネームスペースからアクセスすることはできません。クローンまたはリストア処理によって新しいネームスペースが作成されると、アカウントの管理者 / 所有者はメンバーユーザアカウントを編集し、該当するユーザに新しいネームスペースへのアクセスを許可するロールの制限を更新できます。

OpenShift に関する考慮事項

- クラスタ間でアプリケーションをクローニングする場合、ソースクラスタとデスティネーションクラスタは OpenShift の同じディストリビューションである必要があります。たとえば、OpenShift 4.7 クラスタからアプリケーションをクローニングする場合は、OpenShift 4.7 でもあるデスティネーションクラスタを使用します。
- OpenShift クラスタでアプリケーションをホストするプロジェクトを作成すると、プロジェクト（または Kubernetes ネームスペース）に SecurityContext UID が割り当てられます。Astra Control Center でアプリケーションを保護し、OpenShift でそのアプリケーションを別のクラスタまたはプロジェクトに移動できるようにするには、アプリケーションを任意の UID として実行できるようにポリシーを追加する必要があります。たとえば、次の OpenShift CLI コマンドは、WordPress アプリケーションに適切なポリシーを付与します。

```
OC new-project ワードプレス ` OC adm policy add -scc to -group anyuid system:
serviceaccounts:wordpress `OC adm policy add -scc to -user Privileged -z default-n wordpress
```

手順

1. 「* アプリケーション *」を選択します。
2. 次のいずれかを実行します。
 - 目的のアプリケーションの [* アクション * （* Actions *）] 列で [オプション（Options）] メニューを選択します。
 - 目的のアプリケーションの名前を選択し、ページの右上にあるステータスドロップダウンリストを選択します。
3. 「* Clone *」を選択します。
4. * クローンの詳細 *：クローンの詳細を指定します。
 - 名前を入力します。
 - クローンのネームスペースを入力します。
 - クローンのデスティネーションクラスタを選択してください。
 - 既存の Snapshot からクローンを作成するかバックアップを作成するかを選択します。このオプションを選択しない場合、Astra Control Center はアプリケーションの現在の状態からクローンを作成します。
5. * 出典 *：既存のスナップショットまたはバックアップからクローンを作成する場合は、使用するスナップショットまたはバックアップを選択します。
6. [* Review （レビュー）] を選択します
7. * Clone Summary *：クローンの詳細を確認し、* Clone * を選択します。

結果

Astra Control Center では、入力した情報に基づいてアプリケーションのクローンを作成します。新しいアプリケーション・クローンが [* Applications] ページの [Available （使用可能）] 状態になっている場合、クローン操作は正常に実行されます



データ保護処理（クローン、バックアップ、リストア）が完了して永続ボリュームのサイズを変更したあと、新しいボリュームのサイズが UI に表示されるまでに最大 20 分かかります。データ保護処理にかかる時間は数分です。また、ストレージバックエンドの管理ソフトウェアを使用してボリュームサイズの変更を確認できます。

アプリケーション実行フックを管理します

実行フックは、管理対象アプリケーションのスナップショットの前または後に実行できるカスタムスクリプトです。たとえば、データベースアプリケーションがある場合、実行フックを使用して、スナップショットの前にすべてのデータベーストランザクションを一時停止し、スナップショットの完了後にトランザクションを再開できます。これにより、アプリケーションと整合性のある Snapshot を作成できます。

デフォルトの実行フックと正規表現

一部のアプリケーションでは、ネットアップが提供するデフォルトの実行フックが Astra Control に付属しており、スナップショットの前後にフリーズや再開の操作を処理します。Astra Control では、正規表現を使用して、アプリケーションのコンテナイメージを次のアプリケーションに照合します。

- MariaDB
 - 正規表現 `\bmariadb\b` に一致しています
- MySQL
 - 正規表現 `\bmysql\b` に一致しています
- PostgreSQL
 - 正規表現 `\bpostgresql\b` と一致します

一致した場合は、そのアプリケーションのデフォルトの実行フックがアプリケーションのアクティブな実行フックのリストに表示され、そのアプリケーションのスナップショットが作成されると、それらのフックが自動的に実行されます。カスタムアプリケーションの 1 つに、正規表現の 1 つと一致するように表示されるイメージ名が似ている場合（デフォルトの実行フックを使用しない場合）、イメージ名を変更することができます。または、そのアプリケーションのデフォルト実行フックを無効にして、代わりにカスタムフックを使用します。

デフォルトの実行フックを削除または変更することはできません。

カスタム実行フックに関する重要な注意事項

アプリケーションの実行フックを計画するときは、次の点を考慮してください。

- Astra Control では、実行フックを実行可能なシェルスクリプトの形式で記述する必要があります。
- スクリプトサイズは 128KB に制限されています。
- Astra Control は、実行フックの設定と一致条件を使用して、スナップショットに適用できるフックを決定します。
- 実行フックの障害はすべてソフトな障害です他のフックとスナップショットは 'フックが失敗しても試行されます' ただし、フックが失敗すると、* アクティビティ * ページイベントログに警告イベントが記録されます。
- 実行フックを作成、編集、または削除するには、Owner、Admin、または Member 権限を持つユーザーである必要があります。
- 実行フックの実行に 25 分以上かかる場合 'フックは失敗し' 戻りコードが N/A のイベント・ログ・エントリが作成されます該当する Snapshot はタイムアウトして失敗とマークされ、タイムアウトを通知するイベントログエントリが生成されます。



実行フックは、実行中のアプリケーションの機能を低下させるか、完全に無効にすることが多いため、カスタム実行フックの実行時間を最小限に抑えるようにしてください。

スナップショットが実行されると、実行フックイベントが次の順序で実行されます。

1. ネットアップが提供するデフォルトの Snapshot 前実行フックは、該当するコンテナで実行されます。
2. 適用可能なカスタムスナップショット前実行フックは、適切なコンテナで実行されます。必要な数のカスタムスナップショット前フックを作成して実行できますが、スナップショットの実行順序は保証も構成もされていません
3. スナップショットが実行されます。
4. 適用可能なカスタムスナップショット後実行フックは、適切なコンテナで実行されます。必要な数のカスタムスナップショット後フックを作成して実行できますが、スナップショット後のこれらのフックの実行順序は保証されておらず、構成もできません
5. ネットアップが提供するデフォルトのポスト Snapshot 実行フックは、該当するコンテナで実行されます。



本番環境で実行スクリプトを有効にする前に、必ず実行フックスクリプトをテストしてください。'kubectl exec' コマンドを使用すると、スクリプトを簡単にテストできます。本番環境で実行フックを有効にしたら、作成されたスナップショットの整合性をテストします。これを行うには、アプリケーションを一時ネームスペースにクローニングし、スナップショットをリストアしてから、アプリケーションをテストします。

既存の実行フックを表示します

既存のカスタム実行フックまたはネットアップが提供するアプリケーションのデフォルト実行フックを表示できます。

手順

1. 「* アプリケーション」に移動し、管理アプリの名前を選択します。
2. [実行フック*] タブを選択します。

有効または無効になっているすべての実行フックを結果リストに表示できます。フックのステータス 'ソース' および実行時 (スナップショット前またはスナップショット後) を表示できます。実行フックに関連するイベントログを表示するには、左側のナビゲーション領域の * アクティビティ * ページに移動します。

カスタム実行フックを作成します

アプリケーションのカスタム実行フックを作成できます。を参照してください ["実行フックの例"](#) フックの例を参照してください。実行フックを作成するには、Owner、Admin、または Member のいずれかの権限が必要です。



実行フックとして使用するカスタムシェルスクリプトを作成する場合は、Linux コマンドを実行しているか、実行可能ファイルへの完全パスを提供している場合を除き、ファイルの先頭に適切なシェルを指定するようにしてください。

手順

1. 「* アプリケーション」を選択し、管理アプリの名前を選択します。
2. [実行フック*] タブを選択します。
3. [* 新しいフックを追加*] を選択します。
4. フックの詳細* 領域で、フックを実行するタイミングに応じて、「* Pre-Snapshot*」または「* Post-Snapshot*」を選択します。
5. フックの一意の名前を入力します。
6. (オプション) 実行中にフックに渡す引数を入力し、各引数を入力した後で Enter キーを押して、それぞれを記録します。
7. [* Container Images* (コンテナイメージ*)] 領域で、アプリケーションに含まれるすべてのコンテナイメージに対してフックを実行する必要がある場合は、[* Apply to all container images* (すべてのコンテナイメージに適用*)] チェックボックスを有効にします。代わりに、フックが 1 つ以上の指定されたコンテナイメージに対してのみ機能する場合は、* Container image names to match* フィールドにコンテナイメージ名を入力します。
8. [* スクリプト* (* Script*)] 領域で、次のいずれかを実行します。
 - カスタムスクリプトをアップロードする。
 - i. [ファイルのアップロード (Upload file)] オプションを選択します。
 - ii. ファイルを参照してアップロードします。
 - iii. スクリプトに一意の名前を付けます。
 - iv. (オプション) 他の管理者がスクリプトについて知っておく必要があるメモを入力します。
 - クリップボードからカスタムスクリプトを貼り付けます。
 - i. クリップボードから貼り付け* オプションを選択します。
 - ii. テキストフィールドを選択し、スクリプトテキストをフィールドに貼り付けます。
 - iii. スクリプトに一意の名前を付けます。
 - iv. (オプション) 他の管理者がスクリプトについて知っておく必要があるメモを入力します。
9. [* フックを追加*] を選択します。

実行フックを無効にします

アプリケーションのスナップショットの前または後に実行を一時的に禁止する場合は、実行フックを無効にできます。実行フックを無効にするには、Owner、Admin、または Member のいずれかの権限が必要です。

手順

1. 「* アプリケーション」を選択し、管理アプリの名前を選択します。
2. [実行フック*] タブを選択します。
3. 無効にするフックの* アクション* 列のオプションメニューを選択します。
4. [Disable] を選択します。

実行フックを削除します

不要になった実行フックは完全に削除できます。実行フックを削除するには、Owner、Admin、または

Member のいずれかの権限が必要です。

手順

1. 「* アプリケーション」を選択し、管理アプリの名前を選択します。
2. [実行フック*] タブを選択します。
3. 削除するフックの * アクション * 列のオプションメニューを選択します。
4. 「* 削除」を選択します。

実行フックの例

次の例を使用して、実行フックの構造を確認してください。これらのフックは、テンプレートまたはテストスクリプトとして使用できます。

シンプルな成功例

次に、成功し、標準出力および標準エラーにメッセージを書き込む単純フックの例を示します。

```
#!/bin/sh

# success_sample.sh
#
# A simple noop success hook script for testing purposes.
#
# args: None
#

#
# Writes the given message to standard output
#
# $* - The message to write
#
msg() {
    echo "$*"
}

#
# Writes the given information message to standard output
#
# $* - The message to write
#
info() {
    msg "INFO: $*"
}
```

```

#
# Writes the given error message to standard error
#
# $* - The message to write
#
error() {
    msg "ERROR: $" 1>&2
}

#
# main
#

# log something to stdout
info "running success_sample.sh"

# exit with 0 to indicate success
info "exit 0"
exit 0

```

シンプルな成功の例（**bash** バージョン）

次に、bash 用に書かれた標準出力と標準エラーにメッセージを書き込む単純なフックの例を示します。

```

#!/bin/bash

# success_sample.bash
#
# A simple noop success hook script for testing purposes.
#
# args: None

#
# Writes the given message to standard output
#
# $* - The message to write
#
msg() {
    echo "$*"
}

#

```

```

# Writes the given information message to standard output
#
# $* - The message to write
#
info() {
    msg "INFO: $*"
}

#
# Writes the given error message to standard error
#
# $* - The message to write
#
error() {
    msg "ERROR: $*" 1>&2
}

#
# main
#

# log something to stdout
info "running success_sample.bash"

# exit with 0 to indicate success
info "exit 0"
exit 0

```

単純な成功例（**zsh** バージョン）

これは、成功した単純なフックの例であり、標準出力と標準エラーに Z シェル用に記述されたメッセージを書き込みます。

```

#!/bin/zsh

# success_sample.zsh
#
# A simple noop success hook script for testing purposes.
#
# args: None
#

#
# Writes the given message to standard output

```



```

#
# $* - The message to write
#
msg() {
    echo "$*"
}

#
# Writes the given information message to standard output
#
# $* - The message to write
#
info() {
    msg "INFO: $*"
}

#
# Writes the given error message to standard error
#
# $* - The message to write
#
error() {
    msg "ERROR: $*" 1>&2
}

#
# main
#

# log something to stdout
info "running success_sample.zsh"

# exit with 0 to indicate success
info "exit 0"
exit 0

```

引数を指定した成功の例

次の例は、フックで args を使用する方法を示しています。

```

#!/bin/sh

# success_sample_args.sh
#
# A simple success hook script with args for testing purposes.

```

```

#
# args: Up to two optional args that are echoed to stdout

#
# Writes the given message to standard output
#
# $* - The message to write
#
msg() {
    echo "$*"
}

#
# Writes the given information message to standard output
#
# $* - The message to write
#
info() {
    msg "INFO: $*"
}

#
# Writes the given error message to standard error
#
# $* - The message to write
#
error() {
    msg "ERROR: $*" 1>&2
}

#
# main
#

# log something to stdout
info "running success_sample_args.sh"

# collect args
arg1=$1
arg2=$2

# output args and arg count to stdout
info "number of args: $#"
```

```

info "arg1 ${arg1}"
```

```
info "arg2 ${arg2}"

# exit with 0 to indicate success
info "exit 0"
exit 0
```

スナップショット前 / スナップショット後のフックの例

次の例は、Snapshot 前フックと Snapshot 後フックの両方に同じスクリプトを使用する方法を示しています。

```
#!/bin/sh

# success_sample_pre_post.sh
#
# A simple success hook script example with an arg for testing purposes
# to demonstrate how the same script can be used for both a prehook and
# posthook
#
# args: [pre|post]

# unique error codes for every error case
ebase=100
eusage=$((ebase+1))
ebadstage=$((ebase+2))
epre=$((ebase+3))
epost=$((ebase+4))

#
# Writes the given message to standard output
#
# $* - The message to write
#
msg() {
    echo "$*"
}

#
# Writes the given information message to standard output
#
# $* - The message to write
#
info() {
    msg "INFO: $*"
}
```

```

}

#
# Writes the given error message to standard error
#
# $* - The message to write
#
error() {
    msg "ERROR: $*" 1>&2
}

#
# Would run prehook steps here
#
prehook() {
    info "Running noop prehook"
    return 0
}

#
# Would run posthook steps here
#
posthook() {
    info "Running noop posthook"
    return 0
}

#
# main
#

# check arg
stage=$1
if [ -z "${stage}" ]; then
    echo "Usage: $0 <pre|post>"
    exit ${eusage}
fi

if [ "${stage}" != "pre" ] && [ "${stage}" != "post" ]; then
    echo "Invalid arg: ${stage}"
    exit ${ebadstage}
fi

# log something to stdout

```

```

info "running success_sample_pre_post.sh"

if [ "${stage}" = "pre" ]; then
    prehook
    rc=$?
    if [ ${rc} -ne 0 ]; then
        error "Error during prehook"
    fi
fi

if [ "${stage}" = "post" ]; then
    posthook
    rc=$?
    if [ ${rc} -ne 0 ]; then
        error "Error during posthook"
    fi
fi

exit ${rc}

```

失敗の例

次の例は、フックで障害を処理する方法を示しています。

```

#!/bin/sh

# failure_sample_arg_exit_code.sh
#
# A simple failure hook script for testing purposes.
#
# args: [the exit code to return]
#

#
# Writes the given message to standard output
#
# $* - The message to write
#
msg() {
    echo "$*"
}

#
# Writes the given information message to standard output

```

```

#
# $* - The message to write
#
info() {
    msg "INFO: $*"
}

#
# Writes the given error message to standard error
#
# $* - The message to write
#
error() {
    msg "ERROR: $*" 1>&2
}

#
# main
#

# log something to stdout
info "running failure_sample_arg_exit_code.sh"

argexitcode=$1

# log to stderr
error "script failed, returning exit code ${argexitcode}"

# exit with specified exit code
exit ${argexitcode}

```

詳細なエラーの例

次の例では 'フック' の失敗をより詳細なロギングで処理する方法を示します

```

#!/bin/sh

# failure_sample_verbose.sh
#
# A simple failure hook script with args for testing purposes.
#
# args: [The number of lines to output to stdout]

#

```

```

# Writes the given message to standard output
#
# $* - The message to write
#
msg() {
    echo "$*"
}

#
# Writes the given information message to standard output
#
# $* - The message to write
#
info() {
    msg "INFO: $*"
}

#
# Writes the given error message to standard error
#
# $* - The message to write
#
error() {
    msg "ERROR: $*" 1>&2
}

#
# main
#

# log something to stdout
info "running failure_sample_verbose.sh"

# output arg value to stdout
linecount=$1
info "line count ${linecount}"

# write out a line to stdout based on line count arg
i=1
while [ "$i" -le ${linecount} ]; do
    info "This is line ${i} from failure_sample_verbose.sh"
    i=$(( i + 1 ))
done

```

```
error "exiting with error code 8"  
exit 8
```

終了コード例を使用した失敗

次の例は、終了コードを使用したフックの失敗を示しています。

```
#!/bin/sh  
  
# failure_sample_arg_exit_code.sh  
#  
# A simple failure hook script for testing purposes.  
#  
# args: [the exit code to return]  
#  
  
#  
# Writes the given message to standard output  
#  
# $* - The message to write  
#  
msg() {  
    echo "$*"  
}  
  
#  
# Writes the given information message to standard output  
#  
# $* - The message to write  
#  
info() {  
    msg "INFO: $*"  
}  
  
#  
# Writes the given error message to standard error  
#  
# $* - The message to write  
#  
error() {  
    msg "ERROR: $*" 1>&2  
}
```



```
#
# main
#

# log something to stdout
info "running failure_sample_arg_exit_code.sh"

argexitcode=$1

# log to stderr
error "script failed, returning exit code ${argexitcode}"

# exit with specified exit code
exit ${argexitcode}
```

失敗後の成功の例

次の例では、最初の実行時にフックが失敗していますが、2回目の実行後に成功しています。

```
#!/bin/sh

# failure_then_success_sample.sh
#
# A hook script that fails on initial run but succeeds on second run for
# testing purposes.
#
# Helpful for testing retry logic for post hooks.
#
# args: None
#

#
# Writes the given message to standard output
#
# $* - The message to write
#
msg() {
    echo "$*"
}

#
# Writes the given information message to standard output
#
# $* - The message to write
#
```

```

info() {
    msg "INFO: $*"
}

#
# Writes the given error message to standard error
#
# $* - The message to write
#
error() {
    msg "ERROR: $*" 1>&2
}

#
# main
#

# log something to stdout
info "running failure_success sample.sh"

if [ -e /tmp/hook-test.junk ] ; then
    info "File does exist. Removing /tmp/hook-test.junk"
    rm /tmp/hook-test.junk
    info "Second run so returning exit code 0"
    exit 0
else
    info "File does not exist. Creating /tmp/hook-test.junk"
    echo "test" > /tmp/hook-test.junk
    error "Failed first run, returning exit code 5"
    exit 5
fi

```

著作権に関する情報

Copyright © 2023 NetApp, Inc. All Rights Reserved. Printed in the U.S. このドキュメントは著作権によって保護されています。著作権所有者の書面による事前承諾がある場合を除き、画像媒体、電子媒体、および写真複写、記録媒体、テープ媒体、電子検索システムへの組み込みを含む機械媒体など、いかなる形式および方法による複製も禁止します。

ネットアップの著作物から派生したソフトウェアは、次に示す使用許諾条項および免責条項の対象となります。

このソフトウェアは、ネットアップによって「現状のまま」提供されています。ネットアップは明示的な保証、または商品性および特定目的に対する適合性の暗示的保証を含み、かつこれに限定されないいかなる暗示的な保証も行いません。ネットアップは、代替品または代替サービスの調達、使用不能、データ損失、利益損失、業務中断を含み、かつこれに限定されない、このソフトウェアの使用により生じたすべての直接的損害、間接的損害、偶発的損害、特別損害、懲罰的損害、必然的損害の発生に対して、損失の発生の可能性が通知されていたとしても、その発生理由、根拠とする責任論、契約の有無、厳格責任、不法行為（過失またはそうでない場合を含む）にかかわらず、一切の責任を負いません。

ネットアップは、ここに記載されているすべての製品に対する変更を随時、予告なく行う権利を保有します。ネットアップによる明示的な書面による合意がある場合を除き、ここに記載されている製品の使用により生じる責任および義務に対して、ネットアップは責任を負いません。この製品の使用または購入は、ネットアップの特許権、商標権、または他の知的所有権に基づくライセンスの供与とはみなされません。

このマニュアルに記載されている製品は、1つ以上の米国特許、その他の国の特許、および出願中の特許によって保護されている場合があります。

権利の制限について：政府による使用、複製、開示は、DFARS 252.227-7013（2014年2月）およびFAR 5252.227-19（2007年12月）のRights in Technical Data -Noncommercial Items（技術データ - 非商用品目に関する諸権利）条項の(b)(3)項、に規定された制限が適用されます。

本書に含まれるデータは商用製品および / または商用サービス（FAR 2.101の定義に基づく）に関係し、データの所有権はNetApp, Inc.にあります。本契約に基づき提供されるすべてのネットアップの技術データおよびコンピュータ ソフトウェアは、商用目的であり、私費のみで開発されたものです。米国政府は本データに対し、非独占的かつ移転およびサブライセンス不可で、全世界を対象とする取り消し不能の制限付き使用权を有し、本データの提供の根拠となった米国政府契約に関連し、当該契約の裏付けとする場合にのみ本データを使用できます。前述の場合を除き、NetApp, Inc.の書面による許可を事前に得ることなく、本データを使用、開示、転載、改変するほか、上演または展示することはできません。国防総省にかかる米国政府のデータ使用权については、DFARS 252.227-7015(b)項（2014年2月）で定められた権利のみが認められます。

商標に関する情報

NetApp、NetAppのロゴ、<http://www.netapp.com/TM>に記載されているマークは、NetApp, Inc.の商標です。その他の会社名と製品名は、それを所有する各社の商標である場合があります。