Red Hat Enterprise Linux 7

High Availability Add-On リファレンス

High Availability Add-On の設定および管理のためのリファレンスガイド

Last Updated: 2024-03-02

Red Hat Enterprise Linux 7 High Availability Add-On リファレンス

High Availability Add-On の設定および管理のためのリファレンスガイド

Steven Levine
Red Hat Customer Content Services
slevine@redhat.com

法律上の通知

概要

Red Hat High Availability Add-On リファレンスは、Red Hat Enterprise Linux 7 向けの Red Hat High Availability Add-On をインストール、設定、および管理するための参考情報を提供します。

目次

第1章 RED HAT HIGH AVAILABILITY ADD-ON 設定および管理リファレンスの概要

本章では、Pacemaker を使用する Red Hat High Availability Add-On がサポートするオプションと機能について説明します。ステップごとの基本設定の例は『Red Hat High Availability Add-On の管理 』を参照してください。

pcs 設定インターフェイスまたは **pcsd** GUI インターフェイスを使用して、Red Hat High Availability Add-On クラスターを設定できます。

1.1. 新機能と変更点

本セクションでは、Red Hat Enterprise Linux 7 の初回リリース以降に追加された Red Hat High Availability Add-On の新機能を取り上げます。

1.1.1. Red Hat Enterprise Linux 7.1 の新機能および変更された機能

Red Hat Enterprise Linux 7.1 には、ドキュメントや機能を対象とする以下の更新および変更が含まれています。

- 「クラスターリソースのクリーンアップ」 に記載されているように、**pcs resource cleanup** コマンドが、すべてのリソースの状態と **failcount** をリセットできるようになりました。
- 「リソースを手作業で移動する」 に記載されているように、**pcs resource move** コマンドの **lifetime** パラメーターを指定できます。
- Red Hat Enterprise Linux 7.1 より、**pcs acl** コマンドを使用してローカルユーザーのパーミッションを設定し、アクセス制御リスト(ACL)を使用してクラスター設定への読み取り専用アクセスまたは読み書きアクセスを許可できます。ACL の詳細は、「ユーザーのパーミッション設定」 を参照してください。
- 「順序付けされたリソースセット」 および 「リソースのコロケーション」 が大幅に更新および明確化されました。
- 「リソースの作成」 に、**pcs resource create** コマンドの **disabled** パラメーターが追加され、作成されたリソースが自動的に起動しないことが示されました。
- 「クォーラムオプションの設定」 に、クォーラムの確立時にクラスターがすべてのノードを待たないようにする **cluster quorum unblock** 機能が記載されています。
- 「リソースの作成」 に、リソースグループの順序付けを設定するために使用できる **pcs resource create** コマンドの **before** および **after** パラメーターの説明が追加されました。
- Red Hat Enterprise Linux 7.1 リリースでは、クラスター設定を tarball にバックアップし、**pcs config** コマンドの **backup** および restore オプションを使用して、バックアップからすべてのノードのクラスター設定ファイルを **復元** できます。この機能の詳細は 「クラスター設定のバックアップおよび復元」 を参照してください。
- 内容を明確にするため本書全体に小変更が加えられました。

1.1.2. Red Hat Enterprise Linux 7.2 の新機能および変更された機能

Red Hat Enterprise Linux 7.2 ではドキュメントと機能が以下のように更新/変更されています。

- **pcs resource relocate run** コマンドを使用して、現在のクラスターのステータス、制約、リ

ソースの場所、およびその他の設定によって決定される優先ノードにリソースを移行できるようになりました。このコマンドの詳細は、「リソースの優先ノードへの移動」 を参照してください。

- 「モニタリングのリソースを使ったイベント通知」 外部プログラムを実行してクラスター通知の処理を判断するために **ClusterMon** リソースを設定する方法をより深く説明するため、変更および拡張されました。
- 冗長な電源供給用のフェンスを設定する場合に各デバイスを 1 度のみ設定する必要があり、ノードのフェンシングには両方のデバイスが必要になることを指定する必要があります。冗長な電源供給にフェンスを設定する方法の詳細は、「冗長電源のフェンシング設定」 を参照してください。
- このドキュメントの 「クラスターノードの追加」 に、ノードを既存のクラスターに追加する手順が追加されました。
- 表7.1「簡単な場所の制約オプション」 の説明にあるように、新しい **resource-discovery** の場所制約オプションにより、Pacemaker が指定されたリソースのノードでリソース検出を実行すべきかどうかを指定できます。
- ドキュメント全体にわたり、記載内容の明確化を図り、若干の修正を加えました。

1.1.3. Red Hat Enterprise Linux 7.3 の新機能および変更された機能

Red Hat Enterprise Linux 7.3 ではドキュメントと機能が以下のように更新、変更されています。

- このバージョンでは、「pacemaker_remote サービス」 全体が書き直されました。
- アラートエージェントを使用して Pacemaker アラートを設定できます。アラートエージェントは、リソース設定と操作を処理するためにクラスター呼び出しのリソースエージェントと同様にクラスターが呼び出す外部プログラムです。Pacemaker アラートエージェントの説明は 「Pacemaker アラートエージェント (Red Hat Enterprise Linux 7.3 以降)」 を参照してください。
- このリリースでは新しいクォーラム管理コマンドがサポートされ、クォーラムの状態を表示し、**expected_votes** パラメーターを変更できます。これらのコマンドの説明は 「クォーラム管理コマンド (Red Hat Enterprise Linux 7.3 以降)」 を参照してください。
- 「クォーラムオプションの変更 (Red Hat Enterprise Linux 7.3 以降)」 の説明に従って、**pcs quorum update** コマンドを使用してクラスターの一般的なクォーラムオプションを変更できるようになりました。
- クラスターのサードパーティー判別デバイスとして動作する個別のクォーラムデバイスを設定できます。この機能は主に、標準のクォーラムルールが許可するよりも多くのノード障害をクラスターで維持できるようにするために使用されます。この機能はテクニカルリビューとしてのみ提供されます。クォーラムデバイスの説明は 「クォーラムデバイス」 を参照してください。
- Red Hat Enterprise Linux 7.3 には、Booth クラスターチケットマネージャーを使用して複数のサイトにまたがる高可用性クラスターを設定する機能が提供されます。この機能はテクニカルリビューとしてのみ提供されます。Booth クラスターチケットマネージャーの説明は 14 章*Pacemaker を用いたマルチサイトクラスターの設定* を参照してください。
- **pacemaker_remote** サービスを実行している KVM ゲストノードを設定する場合、グループにゲストノードを含めることができます。これにより、ストレージデバイス、ファイルシステム、および VM をグループ化できます。KVM ゲストノードの設定に関する詳細は 「設定の概要: KVM ゲストノード」 を参照してください。

さらに、ドキュメント全体にわたり記載内容の明確化を図り、若干の修正を加えました。

1.1.4. Red Hat Enterprise Linux 7.4 の新機能および変更された機能

Red Hat Enterprise Linux 7.4 では、ドキュメントと機能が以下のように更新、変更されています。

- Red Hat Enterprise Linux 7.4 には、Booth クラスターチケットマネージャーを使用して複数のサイトにまたがる高可用性クラスターを設定する機能が完全に提供されます。Booth クラスターチケットマネージャーの説明は *14章Pacemaker を用いたマルチサイトクラスターの設定* を参照してください。

- Red Hat Enterprise Linux 7.4 は、個別のクォーラムを設定する機能に完全に対応しています。この機能は主に、標準のクォーラムルールが許可するよりも多くのノード障害をクラスターで維持できるようにするために使用されます。クォーラムデバイスの説明は 「クォーラムデバイス」 を参照してください。

- ノード名で適用した正規表現と、ノード属性とその値によってフェンシングトポロジーでノードを指定できるようになりました。フェンシングレベルの説明は、「フェンスレベルの設定」を参照してください。

- Red Hat Enterprise Linux 7.4 は、**NodeUtilization** リソースエージェントをサポートします。これは、利用可能な CPU、ホストメモリーの可用性、およびハイパーバイザーメモリーの可用性のシステムパラメーターを検出し、これらのパラメーターを CIB に追加します。このリソースエージェントの詳細は、「NodeUtilization リソースエージェント (Red Hat Enterprise Linux 7.4 以降)」 を参照してください。

- Red Hat Enterprise Linux 7.4 では、**クラスターノード add-guest** コマンドおよび **クラスターノード remove-guest** コマンドは、**cluster remote-node add** および **cluster remote-node remove** コマンドを置き換えます。**pcs cluster node add-guest** コマンドはゲストノードの **authkey** をセットアップし、**pcs cluster node add-remote** コマンドはリモートノードの **authkey** を設定します。更新したゲストとリモートノード設定手順は、「リソースとしての仮想ドメインの設定」 を参照してください。

- Red Hat Enterprise Linux 7.4 は、**systemd resource-agents-deps** ターゲットに対応しています。これにより、「Pacemaker で管理されていないリソースの依存関係の起動順の設定 (Red Hat Enterprise Linux 7.4 以降)」 で説明しているように、クラスターにより管理されない依存関係を持つリソースを含むクラスターに適切な起動順序を設定できるようになります。

- マスター/スレーブクローンとしてリソースを作成するコマンドの形式は、このリリースで変更されています。マスター/スレーブクローンの作成の説明は、「多状態のリソース: 複数モードのリソース」 を参照してください。

1.1.5. Red Hat Enterprise Linux 7.5 の新機能および変更された機能

Red Hat Enterprise Linux 7.5 では、ドキュメントと機能が以下のように更新、変更されています。

- Red Hat Enterprise Linux 7.5 では、**pcs_snmp_agent** デーモンを使用して、SNMP でデータについて Pacemaker クラスターを照会できます。SNMP でのクラスター照会は、「SNMP での Pacemaker クラスターを照会 (Red Hat Enterprise Linux 7.5 以降)」 を参照してください。

1.1.6. Red Hat Enterprise Linux 7.8 の新機能および変更された機能

Red Hat Enterprise Linux 7.8 では、ドキュメントと機能が以下のように更新、変更されています。

- Red Hat Enterprise Linux 7.8 以降では、ノードが正常にシャットダウンすると、ノードに接続されているリソースがノードにロックされ、シャットダウンしたノードがクラスターに再度参

加するときに再び起動するまで、他の場所で起動できないように、Pacemaker を設定できます。これにより、ノードのリソースをクラスター内の他のノードにフェイルオーバーせずに、サービスの停止が許容できるメンテナンスウィンドウ中にノードの電源を切ることができます。ノードの正常なシャットダウン時に停止したままになるようにリソースを設定する方法は、「クリーンノードのシャットダウンで停止するようにリソースを設定 (Red Hat Enterprise Linux 7.8 以降)」を参照してください。

1.2. PACEMAKER 設定ツールのインストール

以下の **yum install** コマンドを使用して、Red Hat High Availability Add-On ソフトウェアパッケージと、利用可能なすべてのフェンスエージェントを High Availability チャンネルからインストールできます。

```
# yum install pcs pacemaker fence-agents-all
```

このコマンドの代わりに以下のコマンドを実行すると、Red Hat High Availability Add-On ソフトウェアパッケージと必要なフェンスエージェントのみをインストールできます。

```
# yum install pcs pacemaker fence-agents-model
```

以下のコマンドは、利用できるフェンスエージェントのリストを表示します。

```
# rpm -q -a | grep fence
fence-agents-rhevm-4.0.2-3.el7.x86_64
fence-agents-ilo-mp-4.0.2-3.el7.x86_64
fence-agents-ipmilan-4.0.2-3.el7.x86_64
...
```

lvm2-cluster および **gfs2-utils** パッケージは ResilientStorage チャンネルに含まれます。必要に応じて次のコマンドでインストールを行ってください。

```
# yum install lvm2-cluster gfs2-utils
```

警告

Red Hat High Availability Add-On パッケージをインストールしたら、自動的に何もインストールされないように、ソフトウェア更新設定を行う必要があります。実行中のクラスターにインストールすると、予期しない動作が発生する可能性があります。

1.3. ファイアウォールでクラスターコンポーネントを許可する IPTABLES 設定

注記

クラスターコンポーネントの理想的なファイアウォール設定は、ローカル環境によって異なります。ここでは、ノードに複数のネットワークインターフェイスがあるかどうか、またはオフホストのファイアウォールがあるかどうかを検討しないといけない場合があります。この例では、Pacemaker クラスターで通常必要となるポートを開きますが、ローカル条件に合わせて変更する必要があります。

表1.1「High Availability Add-On で有効にするポート」 では、Red Hat High Availability Add-On で有効にするポートを示し、ポートの使用目的を説明します。以下のコマンドを実行し、**firewalld** デーモンでこのポートをすべて有効にできます。

```
# firewall-cmd --permanent --add-service=high-availability
# firewall-cmd --add-service=high-availability
```

表1.1 High Availability Add-On で有効にするポート

ポート	必要になる場合
TCP 2224	すべてのノードで必須(**pcsd** Web UI で必要で、ノード間通信に必要) 任意のノードの **pcs** が、それ自体も含め、クラスター内のすべてのノードと通信できるように、ポート 2224 を開くことが重要です。Booth クラスターチケットマネージャーまたはクォーラムデバイスを使用する場合は、Booth Arbiter、クォーラムデバイスなどのすべての関連ホストで、ポート 2224 を開く必要があります。
TCP 3121	クラスターに Pacemaker リモートノードがある場合に、すべてのノードで必須です。 完全なクラスターノード上の Pacemaker の **crmd** デーモンは、ポート 3121 で Pacemaker リモートノードの **pacemaker_remoted** デーモンに接続します。クラスター通信に別のインターフェイスを使用する場合は、そのインターフェイスでポートを開くことのみが必要になります。少なくとも、ポートは、Pacemaker リモートノードの全クラスターノードに対して開いている必要があります。ユーザーは完全なノードとリモートノード間でホストを変換する可能性があるか、またはホストのネットワークを使用してコンテナー内でリモートノードを実行する可能性があるため、すべてのノードに対してポートを開くことは役に立ちます。ノード以外のホストにポートを開く必要はありません。
TCP 5403	**corosync-qnetd** でクォーラムデバイスを使用する場合は、クォーラムデバイスホストで必須です。デフォルト値は、**corosync-qnetd** コマンドの **-p** オプションで変更できます。
UDP 5404	corosync がマルチキャスト UDP に設定されている場合には、**corosync** ノードで必須です。
UDP 5405	すべての corosync ノードで必須(**corosync**で必要)
TCP 21064	DLM に必要なリソース(**clvm** や **GFS2**など)がクラスターに含まれる場合に、すべてのノードで必須です。
TCP 9929、UDP 9929	Booth チケットマネージャーを使用してマルチサイトクラスターを確立するときに、すべてのクラスターノード、および同じノードのいずれかからの接続に対して Booth arbitrator ノードで開いている必要があります。

1.4. クラスターと PACEMAKER の設定ファイル

Red Hat High Availability アドオンの設定ファイルは、**corosync.conf** および **cib.xml** です。

corosync.conf ファイルは、Pacemaker が構築されているクラスターマネージャーである **corosync** が使用するクラスターパラメーターを提供します。一般的に、**corosync.conf** を直接編集するのではなく、**pcs** または **pcsd** インターフェイスを使用します。ただし、このファイルを直接編集する必要のある状況も考えられます。corosync.conf ファイルの編集は、Editing the **corosync.conf** file in Red Hat Enterprise Linux 7 を参照してください。

cib.xml ファイルは、クラスターの設定、およびクラスターの全リソースの現在の状態を表す XML ファイルです。このファイルは Pacemaker のクラスター情報ベース (CIB) により使用されます。CIB の内容はクラスター全体で自動的に同期されます。**cib.xml** ファイルは直接編集せず、代わりに **pcs** または **pcsd** インターフェイスを使用してください。

1.5. クラスター設定の注意事項

Red Hat High Availability Add-On クラスターの設定時、以下の注意事項を考慮する必要があります。

- RHEL 7.7 以降、Red Hat はノード数が 32 個を超えるクラスターデプロイメントをサポートしていません。ただし、**pacemaker_remote** サービスを実行しているリモートノードでは、この制限を超えた拡張が可能です。**pacemaker_remote** サービスの説明は 「pacemaker_remote サービス」 を参照してください。

- DHCP (Dynamic Host Configuration Protocol)を使用した **corosync** デーモンで使用されるネットワークインターフェイス上の IP アドレスの取得はサポートされていません。アドレスの更新中、DHCP クライアントは割り当てられたインターフェイスに対して定期的に IP アドレスを削除および再追加することができます。これにより、**corosync** が接続障害を検出し、クラスターの他のノードからのフェンシングアクティビティーがハートビート接続に **corosync** を使用します。

1.6. RED HAT ENTERPRISE LINUX HIGH AVAILABILITY クラスターの更新

RHEL High Availability Add-On および Resilient Storage Add-On を設定するパッケージを、個別または一括で更新するには、以下に示す一般的な方法のいずれかを使用できます。

- **ローリング更新** - サービスからノードを、一度に 1 つずつ削除し、そのソフトウェアを更新してから、そのノードをクラスターに戻します。これにより、各ノードの更新中も、クラスターがサービスの提供とリソースの管理を継続できます。

- **クラスター全体の更新** - クラスター全体を停止し、更新をすべてのノードに適用してから、クラスターのバックアップを開始します。

警告

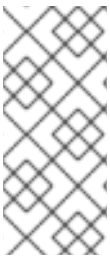
Red Hat Enterprise LInux の High Availability クラスターおよび Resilient Storage クラスターのソフトウェア更新手順を実行する場合は、更新を開始する前に、更新を行うノードがクラスターのアクティブなメンバーではないことを確認する必要があります。

これらの各方法の詳細な説明および更新手順はRecommended Practices for Applying Software Updates to a RHEL High Availability or Resilient Storage Clusterを参照してください。

1.7. RHEL クラスターでの VM のライブ移行についての問題

仮想化クラスターメンバーとの RHEL 高可用性クラスターのサポートポリシーの説明は、Support Policies for RHEL High Availability Clusters - General Conditions with Virtualized Cluster Members を参照してください。前述のように、Red Hat は、ハイパーバイザーまたはホスト全体のアクティブなクラスターノードのライブマイグレーションはサポート対象外です。ライブマイグレーションを実行する必要がある場合は、まず仮想マシンでクラスターサービスを停止して、クラスターからノードを削除し、移行後にクラスターを起動する必要があります。

以下の手順では、クラスターから仮想マシンを削除し、仮想マシンを移行し、クラスターに仮想マシンを復元する手順の概要を説明します。



注記

この手順を実行する前に、クラスターノードを削除するクラスタークォーラム (定足数) への影響を考慮してください。3 つのノードクラスターがあり、1つのノードを削除する場合、クラスターが耐えられるのは、あと1つのノードエラーのみです。3 つのノードクラスターの1つがすでにダウンしている場合は、2 つ目のノードを削除すると、クォーラムが失われます。

1. 移行する仮想マシンで実行しているリソースやソフトウェアの停止または移動を行う前に準備を行う必要がある場合は、以下の手順を実行します。
2. 管理リソースを VM から移動します。リソースの割り当てに関する特定の要件や条件がある場合は、正しいノードにリソースを配置するための新しい場所の制約を作成することを考慮してください。
3. VM をスタンバイモードにして、サービスで考慮されていないことや、残りのリソースが別の場所に再配置され、停止されるようにします。

```
# pcs cluster standby VM
```

4. 仮想マシンで以下のコマンドを実行して、仮想マシン上のクラスターソフトウェアを停止します。

```
# pcs cluster stop
```

5. 仮想マシンのライブマイグレーションを実行します。
6. 仮想マシンでクラスターサービスを起動します。

```
# pcs cluster start
```

7. VM をスタンバイモードから解除します。

```
# pcs cluster unstandby VM
```

8. VM をスタンバイモードにする前に一時的な場所の制約を作成した場合、これらの制約を調整または削除して、リソースが通常の優先場所に戻れるようにします。

第2章 PCSD WEB UI

本章では、**pcsd** Web UI を使用した Red Hat High Availability クラスターの設定の概要を説明します。

2.1. PCSD WEB UI の設定

pcsd Web UI を使用してクラスターを設定するようにシステムを設定するには、以下の手順に従います。

1. 「Pacemaker 設定ツールのインストール」 の説明に従って Pacemaker 設定ツールをインストールします。
2. クラスターの一部である各ノードで、**passwd** コマンドを使用して、各ノードで同じパスワードを使用してユーザー **hacluster** のパスワードを設定します。
3. 各ノードで **pcsd** デーモンを開始して有効にします。

```
# systemctl start pcsd.service
# systemctl enable pcsd.service
```

4. クラスターの 1 つのノードで、以下のコマンドを使用してクラスターを設定するノードを認証します。このコマンドを実行すると、**Username** と **Password** の入力を求められます。**Username** には **hacluster** を指定します。

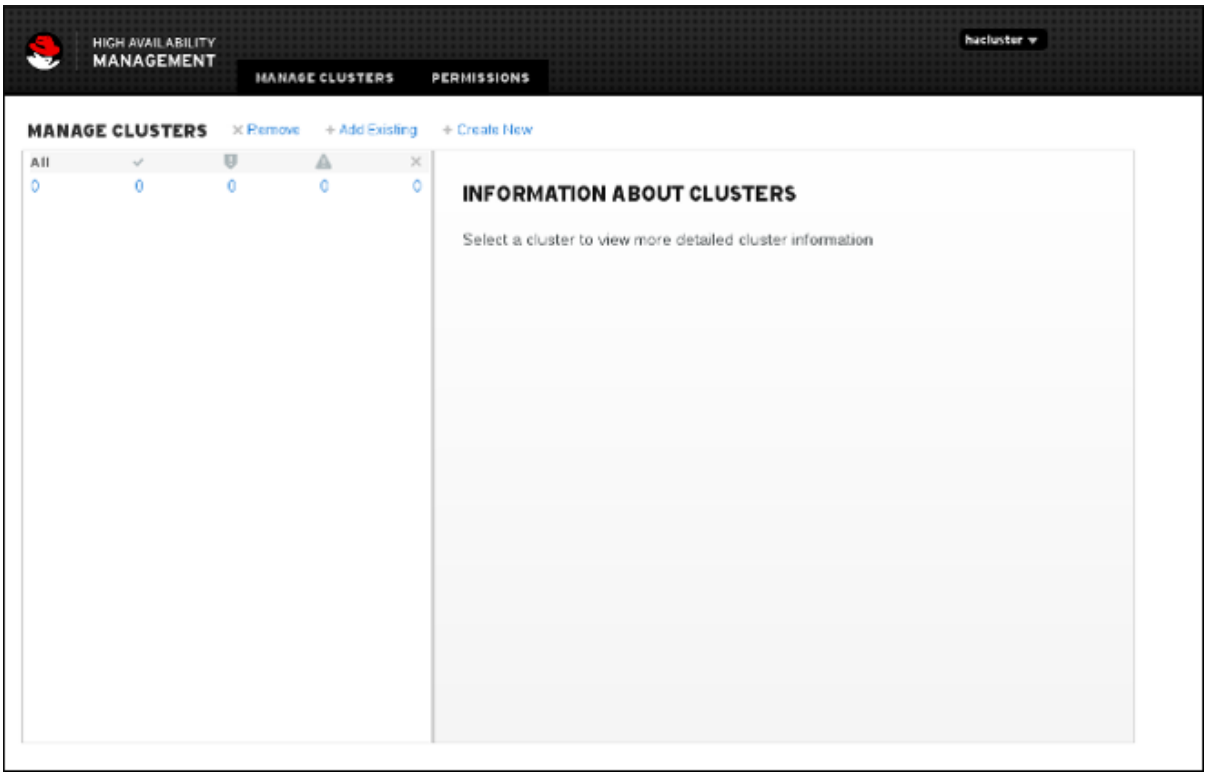
```
# pcs cluster auth node1 node2 ... nodeN
```

5. いずれかのシステムで、次の URL をブラウザーで開き、承認したノードの 1 つを指定します(**https** プロトコルを使用することに注意してください)。これにより、**pcsd** Web UI のログイン画面が表示されます。

```
https://nodename:2224
```

6. ユーザー **hacluster** としてログインします。これにより、図2.1「クラスターの管理ページ」 に示されるように、**クラスターの管理** ページが表示されます。

図2.1 クラスターの管理ページ



[D]

2.2. PCSD WEB UI を用いたクラスターの作成

Manage Clusters ページでは、新規クラスターの作成、既存のクラスターの Web UI への追加、または Web UI からのクラスターの削除を行うことができます。

- クラスターを作成するには、**Create New** をクリックし、作成するクラスターとクラスターを設定するノードの名前を入力します。また、この画面では「高度なクラスター設定オプション」に記載されているクラスター通信のトランスポートメカニズムなどの高度なクラスターオプションを設定することもできます。クラスター情報を入力したら、**Create Cluster** をクリックします。

- 既存のクラスターを Web UI に追加するには、**Add Existing** をクリックし、Web UI で管理するクラスターのノードのホスト名または IP アドレスを入力します。

クラスターを作成または追加すると、**Manage Cluster** ページにクラスター名が表示されます。クラスターを選択すると、クラスターに関する情報が表示されます。

注記

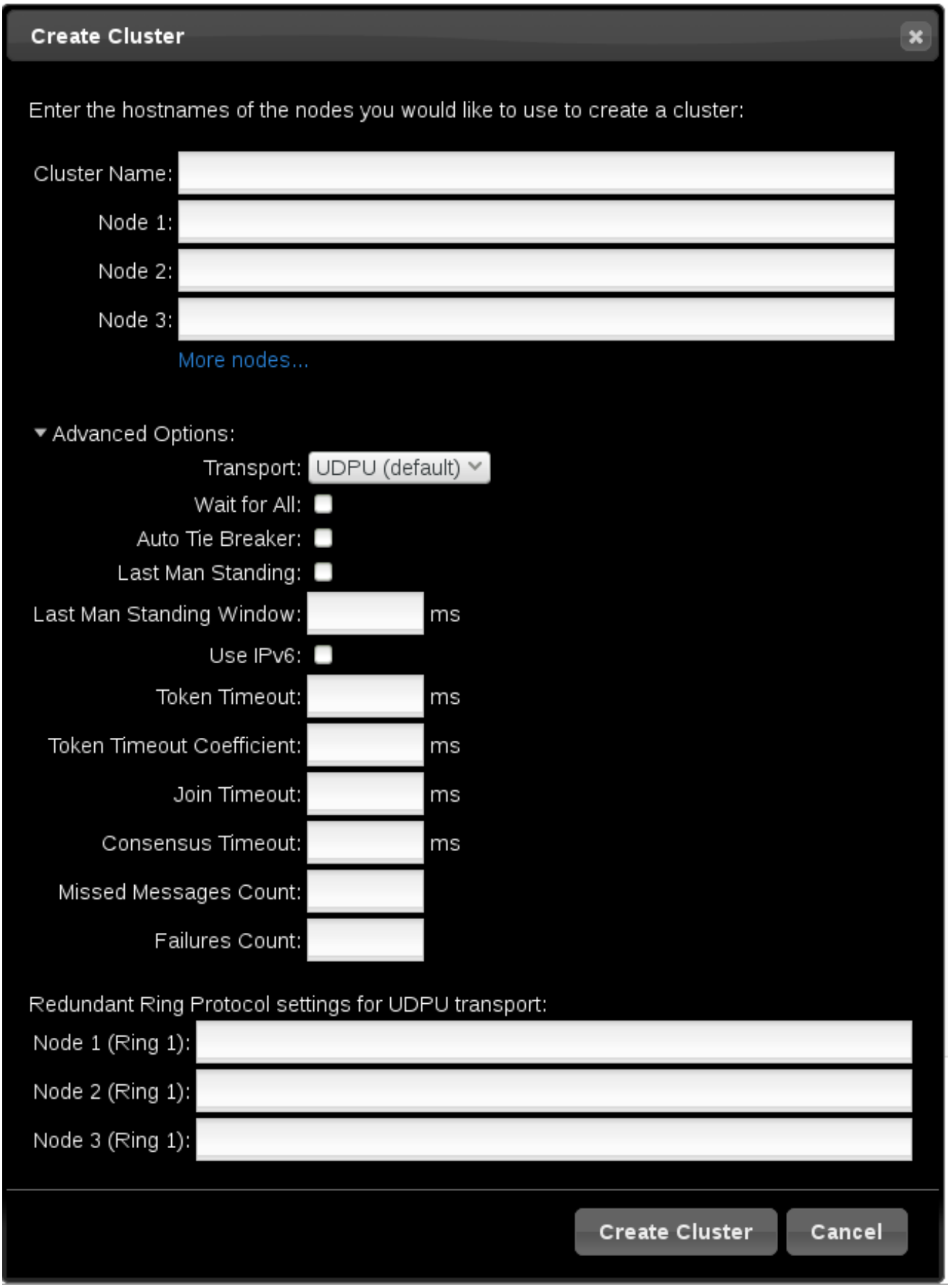
pcsd Web UI を使用してクラスターを設定する場合、多くのオプションを説明するテキストの上にマウスを移動すると、このオプションの詳細な説明が **ツールチップ** 表示として表示できます。

2.2.1. 高度なクラスター設定オプション

クラスターの作成時、図2.2「クラスターページの作成」に示されるように **Advanced Options** をクリックすると追加のクラスターオプションを設定できます。表示されるオプションのテキスト上にマウスを移動すると、そのオプションの情報を確認できます。

各ノードのインターフェイスを指定すると、クラスターに冗長リングプロトコル (Redundant Ring Protocol) を設定できます。クラスターのトランスポートメカニズムとして **UDPU** ではなく **UDP** を選択した場合、Redundant Ring Protocol 設定が表示されます。

図2.2 クラスターページの作成



[D]

2.2.2. クラスター管理パーミッションの設定

ユーザーに付与できるクラスターパーミッションには、以下の 2 つのセットがあります。

- Web UI を使用してクラスターを管理するためのパーミッション。ネットワーク経由でノードに接続する **pcs** コマンドを実行するパーミッションも付与されます。本セクションでは、Web UI でこのパーミッションを設定する方法を説明します。
- ACL を使用し、クラスター設定への読み取り専用アクセスまたは読み書きアクセスをローカルユーザーに許可するパーミッション。Web UI で ACL を設定する方法は、「ACL の設定」 を参照してください。

ユーザーのパーミッションの詳細は、「ユーザーのパーミッション設定」 を参照してください。

グループ **haclient** にユーザーを追加することで、ユーザー **hacluster** 以外の特定のユーザーにパーミッションを付与し、Web UI でクラスターを管理し、ネットワーク経由でノードに接続する **pcs** コマンドを実行できます。次に、**Manage Clusters** ページの **Permissions** タブをクリックし、表示された画面でパーミッションを設定すると、グループ **haclient** の個別のメンバーにパーミッションセットを設定できます。この画面では、グループのパーミッションを設定することもできます。

以下のパーミッションを付与できます。

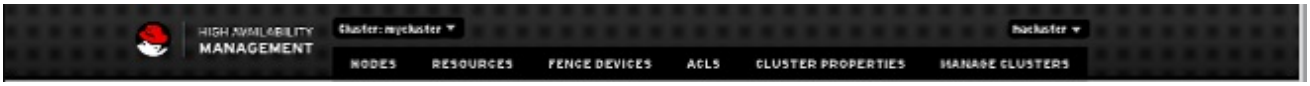
- 読み取りパーミッション (クラスター設定の表示)
- 書き込みパーミッション (パーミッションおよび ACL を除くクラスター設定の変更)
- 付与パーミッション (クラスターパーミッションおよび ACL の変更)
- フルパーミッション (ノードの追加や削除などのクラスターへの無制限アクセス、およびキーや証明書へのアクセス)

2.3. クラスターコンポーネントの設定

クラスターのコンポーネントおよび属性を設定するには、**Manage Clusters** 画面に表示されるクラスターの名前をクリックします。これにより、「クラスターノード」 に示されるように、**Nodes** ページが表示されます。図2.3「クラスターコンポーネントのメニュー」 の説明どおり、このページの上部には以下のエントリーが含まれるメニューが表示されます。

- **nodes** （で説明されている） 「クラスターノード」
- **Resources** （を参照） 「クラスターリソース」
- **フェンスデバイス** （を参照） 「フェンスデバイス」
- **ACL** （を参照） 「ACL の設定」
- **Cluster Properties** （クラスタープロパティー）を参照してください。 「クラスターのプロパティー」

図2.3 クラスターコンポーネントのメニュー



[D]

2.3.1. クラスターノード

クラスター管理ページの上部にあるメニューから ノード オプションを選択すると、現在設定されているノードと、現在選択されているノードのステータス（ノード上で実行されているリソースやリソースの場所の設定など）が表示されます。**このページは、Manage Clusters 画面でクラスターを選択すると表示されるデフォルトページです。**

このページでノードを追加または削除することができ、ノードを起動、停止、および再起動することができます。スタンバイモードの詳細は、「スタンバイモード」 を参照してください。

また、Configure Fencing を選択することで、「フェンスデバイス」 で説明されているように、このページで直接フェンスデバイスを設定することもできます。

2.3.2. クラスターリソース

クラスター管理ページの上部にあるメニューから Resources オプションを選択すると、クラスターに現在設定されているリソースがリソースグループに応じて表示されます。グループまたはリソースを選択すると、そのグループまたはリソースの属性が表示されます。

この画面では、リソースの追加または削除、既存リソースの設定の編集、およびリソースグループの作成を行うことができます。

新しいリソースをクラスターに追加するには、Add をクリックします。Add Resource 画面が表示されます。ドロップダウン タイプ メニューからリソースタイプを選択すると、そのリソースに指定する必要がある引数がメニューに表示されます。Optional Arguments をクリックすると、定義するリソースに指定できる任意の引数を表示できます。作成するリソースのパラメーターを入力したら、Create Resource をクリックします。

リソースの引数を設定する際に、引数の簡単な説明がメニューに表示されます。カーソルをフィールドに移動すると、その引数のヘルプが表示されます。

リソースは、クローンされたリソースまたはマスター/スレーブリソースして定義できます。これらのリソースタイプの詳細は、*9章高度な設定* を参照してください。

少なくとも 1 つのリソースを作成したら、リソースグループを作成できます。リソースグループの詳細は 「リソースグループ」 を参照してください。

リソースグループを作成するには、Resources 画面でグループの一部であるリソースを選択し、Create Group をクリックします。Create Group 画面が表示されます。グループ名を入力し、Create Group をクリックします。これにより、リソースのグループ名を表示する リソース 画面に

戻ります。リソースグループを作成したら、追加のリソースを作成または変更する際に、グループ名をリソースパラメーターとして指定できます。

2.3.3. フェンスデバイス

クラスター管理ページの上部にあるメニューから Fence Devices オプションを選択すると、Fence Devices 画面が表示され、現在設定されているフェンスデバイスが表示されます。

新しいフェンスデバイスをクラスターに追加するには、Add をクリックします。フェンスデバイスの追加 画面が表示されます。ドロップダウンメニューからフェンスデバイスのタイプを選択すると、そのフェンスデバイスに指定する必要がある引数がメニューに表示されます。Optional Arguments をクリックすると、定義するフェンスデバイスに指定できる追加の引数を表示できます。新しいフェンスデバイスのパラメーターを入力したら、Create Fence Instance をクリックします。

Pacemaker を用いたフェンスデバイスの設定の詳細は *5章 フェンス機能: STONITH の設定* を参照してください。

2.3.4. ACL の設定

クラスター管理ページの上部にあるメニューから ACL オプションを選択すると、ローカルユーザーのパーミッションを設定できる画面が表示され、アクセス制御リスト(ACL)を使用してクラスター設定への読み取り専用または読み書きアクセスが可能になります。

ACL パーミッションを割り当てるには、ロールを作成し、そのロールのアクセスパーミッションを定義します。各ロールには、XPath クエリーまたは特定要素の ID のいずれかにパーミッション (読み取り/書き込み/拒否) をいくつでも適用できます。ロールを定義したら、既存のユーザーまたはグループに割り当てることができます。

2.3.5. クラスターのプロパティー

クラスター管理ページの上部にあるメニューから Cluster Properties オプションを選択するとクラスタープロパティーが表示され、このプロパティーをデフォルト値から変更できます。Pacemaker クラスタープロパティーの詳細は *12章Pacemaker クラスターのプロパティー* を参照してください。

2.4. 高可用性 PCSD WEB UI の設定

pcsd Web UI を使用すると、クラスターのノードのいずれかに接続して、クラスター管理ページを表示できます。接続先のノードがダウンするか、使用できなくなった場合は、クラスターの別のノードを指定する URL でブラウザーを開くと、クラスターに再接続できます。ただし、pcsd Web UI 自体を

高可用性向けに設定することもできます。この場合、新しい URL を入力することなく継続してクラスターを管理できます。

高可用性に pcsd Web UI を設定するには、以下の手順を実行します。

1. /etc/sysconfig/pcsd 設定ファイルで PCSD_SSL_CERT_SYNC_ENABLED が true に設定されていることを確認します。これは、RHEL 7 のデフォルト値です。証明書の同期を有効にすると、pcsd はクラスター設定とノードの add コマンドの pcsd 証明書を同期します。

2. pcsd Web UI への接続に使用するフローティング IP アドレスである IPaddr2 クラスターリソースを作成します。物理ノードに関連付けられている IP アドレスは使用できません。IPaddr2 リソースの NIC デバイスが指定されていない場合は、そのノードに静的に割り当てられている IP アドレスの 1 つと同じネットワークにフローティング IP が存在している必要があります。存在していないと、Floating IP アドレスを割り当てる NIC デバイスが適切に検出されません。

3. pcsd で使用するカスタム SSL 証明書を作成し、 pcsd Web UI への接続に使用するノードのアドレスに対して有効であることを確認します。

   a. カスタムの SSL 証明書を作成するには、ワイルドカード証明書を使用するか、SAN (Subject Alternative Name: サブジェクトの別名) 証明書の延長を使用できます。Red Hat 証明書システムに関する詳細は、Red Hat Certificate System Administration Guideを参照してください。

   b. pcs pcsd certkey コマンドを使用して pcsd のカスタム証明書をインストールします。

   c. pcs pcsd sync-certificates コマンドを使用して、pcsd 証明書をクラスター内のすべてのノードに同期します。

4. クラスターリソースとして設定したフローティング IP アドレスを使用して、pcsd Web UI に接続します。

注記

高可用性向けに pcsd Web UI を設定しても、接続先のノードがダウンすると、再度ログインするよう求められます。

第3章 PCS コマンドラインインターフェイス

pcs コマンドラインインターフェイスは、 corosync.conf ファイルおよび cib.xml ファイルにインターフェイスを提供することで、corosync および Pacemaker を制御し、設定します。

pcs コマンドの一般的な形式は以下のとおりです。

```
pcs [-f file] [-h] [commands]...
```

3.1. PCS コマンド

pcs コマンドは以下のとおりです。

- **cluster**

  クラスターオプションおよびノードの設定を行います。pcs cluster コマンドの詳細は *4章 クラスターの作成と管理* を参照してください。

- **resource**

  クラスターリソースの作成と管理を行います。pcs cluster コマンドの詳細は *6章 クラスターリソースの設定*、*8章 クラスターリソースの管理*、*9章 高度な設定* を参照してください。

- **stonith**

  Pacemaker との使用に備えてフェンスデバイスを設定します。pcs stonith コマンドの詳細は、*5章 フェンス機能: STONITH の設定* を参照してください。

- **constraint**

  リソースの制約を管理します。pcs constraint コマンドの詳細は、*7章 リソースの制約* を参照してください。

- プロパティー

  Pacemaker のプロパティーを設定します。pcs property コマンドでプロパティーを設定する方法については 12章*Pacemaker クラスターのプロパティー* を参照してください。

- status

  現在のクラスターとリソースの状態を表示します。pcs status コマンドの詳細は 「状態の表示」 を参照してください。

- config

  ユーザーが理解できる形式でクラスターの全設定を表示します。pcs config コマンドの詳細は 「全クラスター設定の表示」 を参照してください。

3.2. PCS の使用に関するヘルプ表示

pcs の -h オプションを使用すると pcs コマンドのパラメーターとその説明が表示されます。たとえば、以下のコマンドは pcs resource コマンドのパラメーターを表示します。出力の一部だけが表示されます。

```
# pcs resource -h
Usage: pcs resource [commands]...
Manage pacemaker resources
Commands:
    show [resource id] [--all]
        Show all currently configured resources or if a resource is specified
        show the options for the configured resource.  If --all is specified
        resource options will be displayed

    start <resource id>
        Start resource specified by resource_id
...
```

3.3. RAW クラスター設定の表示

クラスター設定ファイルは直接編集する必要はありませんが、pcs cluster cib コマンドを使用すると未加工クラスター 設定を表示できます。

「設定の変更をファイルに保存」 に記載されているように、**pcs cluster cib *filename*コマンドを使用すると raw クラスター 設定を指定のファイルに保存することができます。**

3.4. 設定の変更をファイルに保存

pcs コマンドを使用する場合は、-f オプションを使用して、アクティブな CIB に影響を与えずに、ファイルに設定変更を保存できます。

クラスターを事前に設定していて、アクティブな CIB が存在する場合は、以下のコマンドを実行して、未編集の xml ファイルを保存します。

```
pcs cluster cib filename
```

たとえば、以下のコマンドは CIB の未加工の xml を testfile という名前のファイルに保存します。

```
# pcs cluster cib testfile
```

以下のコマンドは、testfile ファイルにリソースを作成しますが、リソースを現在実行中のクラスター設定には追加しません。

```
# pcs -f testfile resource create VirtualIP ocf:heartbeat:IPaddr2 ip=192.168.0.120 cidr_netmask=24
op monitor interval=30s
```

次のコマンドを使用して、testfile の現在のコンテンツを CIB にプッシュできます。

```
# pcs cluster cib-push testfile
```

3.5. 状態の表示

次のコマンドで、クラスターおよびクラスターリソースのステータスを表示します。

```
pcs status commands
```

***commands* パラメーターを指定しないとクラスターおよびリソースの全情報が表示されます。resources、groups、cluster、nodes、または pcsd を指定すると、特定のクラスター コンポーネントのみの状態を表示します。**

3.6. 全クラスター設定の表示

現在のクラスター設定をすべて表示する場合は、次のコマンドを実行します。

```
pcs config
```

3.7. 現在の PCS バージョンの表示

以下のコマンドは、実行中の **pcs** の現行バージョンを表示します。

```
pcs --version
```

3.8. クラスター設定のバックアップおよび復元

Red Hat Enterprise Linux 7.1 リリース以降では、以下のコマンドを使用してクラスター設定を **tarball** にバックアップできます。ファイル名を指定しないと、標準出力が使用されます。

```
pcs config backup filename
```

以下のコマンドを使用して、バックアップからすべてのクラスターノードのクラスター設定ファイルを復元します。**--local** オプションを指定すると、このコマンドを実行するノードでのみクラスター設定ファイルが復元されます。ファイル名を指定しないと、標準入力が使用されます。

```
pcs config restore [--local] [filename]
```

第4章 クラスターの作成と管理

本章ではクラスターの作成、クラスターコンポーネントの管理、クラスターの状態表示など Pacemaker で行うクラスターの基本的な管理について見ていきます。

4.1. クラスターの作成

クラスターを作成するため次のステップを行って行きます。

1. クラスターの各ノードで pcsd を開始します。
2. クラスターを設定するノードを認証します。
3. クラスターノードの設定と同期を行います。
4. クラスターノードでクラスターサービスを起動します。

次のセクションでは、上記の手順で使用するコマンドについて詳しく見ていきます。

4.1.1. pcsd デーモンの開始

以下のコマンドは pcsd サービスを起動し、システムの起動時に pcsd を有効にします。これらのコマンドはクラスターの各ノードで実行する必要があります。

```
# systemctl start pcsd.service
# systemctl enable pcsd.service
```

4.1.2. クラスターノードの認証

次のコマンドは、クラスター内のノード上の pcs デーモンに対して pcs を認証します。

- すべてのノードで pcs 管理者のユーザー名は hacluster である必要があります。hacluster ユーザーのパスワードは、各ノードで同じにすることが推奨されます。

- username または password を指定しないと、コマンドの実行時にノードごとにこれらのパラメーターの入力を求められます。

- ノードを指定しないと、以前に pcs cluster setup コマンドで指定したノードの pcs が認証されます。

```
pcs cluster auth [node] [...] [-u username] [-p password]
```

たとえば、以下のコマンドは z1.example.com と z2.example.com の両方で設定されるクラスター内の両方のノードに対して、z1.example.com のユーザー hacluster を認証します。このコマンドは、クラスターノードのユーザー hacluster のパスワードを要求します。

```
root@z1 ~]# pcs cluster auth z1.example.com z2.example.com
Username: hacluster
Password:
z1.example.com: Authorized
z2.example.com: Authorized
```

承認トークンは、～/ .pcs/tokens ファイル（または /var/lib/pcsd/tokens ）に保存されます。

4.1.3. クラスターノードの設定と起動

次のコマンドでクラスター設定ファイルの設定、指定ノードに対する設定の同期を行います。

- --start オプションを使用すると指定ノードでクラスターサービスが起動します。必要に応じて、別の pcs cluster start コマンドを使用してクラスターサービスを起動することもできます。

  pcs cluster setup --start コマンドを使用してクラスターを作成する場合、または pcs cluster start コマンドでクラスターサービスを開始する場合、クラスターが稼働するまでにわずかな遅延が生じる可能性があります。クラスターとその設定で後続のアクションを実行する前に、pcs cluster status コマンドを使用してクラスターが稼働していることを確認することが推奨されます。

- --local オプションを指定すると、ローカルノードでのみ変更が実行されます。

```
pcs cluster setup [--start] [--local] --name cluster_ name node1 [node2] [...]
```

次のコマンドは指定ノード (複数指定可) でクラスターサービスを起動します。

- --all オプションを使用するとすべてのノードでクラスターサービスを起動します。
- ノードを指定しないとクラスターサービスはローカルのノードでしか起動されません。

```
pcs cluster start [--all] [node] [...]
```

4.2. クラスターのタイムアウト値の設定

pcs cluster setup コマンドを使用してクラスターを作成する場合、クラスターのタイムアウト値はほとんどのクラスター設定に適したデフォルト値に設定されます。システムに他のタイムアウト値が必要な場合は、に記載されている pcs cluster setup オプションを使用してデフォルト値を変更できます。 表4.1「タイムアウトオプション」

表4.1 タイムアウトオプション

オプション	説明
--token *timeout*	トークンを受信しなかった後にトークンの損失が宣言されるまでの時間をミリ秒単位で設定します (デフォルトは 1000 ms です)。
--join *timeout*	join メッセージの待ち時間をミリ秒単位で設定します (デフォルトは 50 ms です)。
--consensus *timeout*	新しいメンバーシップの設定を開始する前に合意が得られるまでの待ち時間をミリ秒単位で設定します (デフォルトは 1200 ms です)。
--miss_count_const *count*	再送信が行われる前に、トークンの受信時に再送信のメッセージがチェックされる最大回数を設定します。デフォルトは 5 (5 つのメッセージ) です。
--fail_recv_const *failures*	新しい設定の設定前に、受信しなければならないメッセージが発生する可能性がある場合、メッセージを受信せずにトークンをローテーションする回数を指定します (デフォルトの失敗数は 2500 です)。

たとえば、以下のコマンドは new_cluster クラスターを作成し、トークンのタイムアウト値を 10000 ミリ秒(10 秒)に設定し、join タイムアウト値を 100 ミリ秒に設定します。

```
# pcs cluster setup --name new_cluster nodeA nodeB --token 10000 --join 100
```

4.3. 冗長リングプロトコル (RRP) の設定

注記

Red Hat は、 Support Policies for RHEL High Availability Clusters - Cluster Interconnect Network Interfaces の Redundant Ring Protocol (RRP) の項で説明している条件に依存する、クラスターでの冗長リングプロトコルに対応しています。

pcs cluster setup コマンドを使用してクラスターを作成する場合、各ノードに両方のインターフェイスを指定することで冗長リングプロトコルを使用してクラスターを設定できます。デフォルトの udpu トランスポートを使用する場合にクラスターノードを指定するには、リング 0 アドレス、,、リング 1 アドレスの順に指定します。

たとえば、以下のコマンドはノード A とノード B の 2 つのノードを持つ my_rrp_clusterM という名前のクラスターを設定します。ノード A には nodeA -0 と nodeA- 1 の 2 つのインターフェイスがあります。ノード B には、nodeB -0 と nodeB- 1 の 2 つのインターフェイスがあります。RRP を使用してこれらのノードをクラスターとして設定するには、以下のコマンドを実行します。

```
# pcs cluster setup --name my_rrp_cluster nodeA-0,nodeA-1 nodeB-0,nodeB-1
```

udp トランスポートを使用するクラスターで RRP を設定する方法は、pcs cluster setup コマンドのヘルプ画面を参照してください。

4.4. クラスターノードの管理

次のセクションではクラスターサービスの起動や停止、クラスターノードの追加や削除などクラスターノードの管理で使用するコマンドについて説明します。

4.4.1. クラスターサービスの停止

次のコマンドで、指定ノード (複数指定可) のクラスターサービスを停止します。pcs cluster start と同様に、--all オプションを指定すると全ノードのクラスターサービスが停止され、ノードを指定しないとローカルノードでのみクラスターサービスが停止します。

```
pcs cluster stop [--all] [node] [...]
```

次のコマンドで、ローカルノードでクラスターサービスを強制的に停止できます。このコマンドは、kill -9 コマンドを実行します。

```
pcs cluster kill
```

4.4.2. クラスターサービスの有効化および無効化

指定ノード (複数指定可) の起動時にクラスターサービスが実行されるよう設定する場合は次のコマンドを使用します。

- **--all** オプションを使用すると、全ノードでクラスターサービスが有効になります。
- ノードを指定しないと、ローカルノードでのみクラスターサービスが有効になります。

```
pcs cluster enable [--all] [node] [...]
```

指定した 1 つまたは複数のノードの起動時に、クラスターサービスが実行されないよう設定する場合は、次のコマンドを使用します。

- **--all** オプションを使用すると、全ノードのクラスターサービスが無効になります。
- ノードを指定しないと、ローカルノードでのみクラスターサービスが無効になります。

```
pcs cluster disable [--all] [node] [...]
```

4.4.3. クラスターノードの追加

注記

運用保守期間中に、既存のクラスターにノードを追加することが強く推奨されます。これにより、新しいノードとそのフェンシング設定に対して、適切なリソースとデプロイメントのテストを実行できます。

既存クラスターに新しいノードを追加する場合は、以下の手順に従ってください。この例では、既存のクラスターノードは **clusternode-01.example.com**、**clusternode-02.example.com**、および **clusternode-03.example.com** です。新しいノードは **newnode.example.com** になります。

クラスターに追加する新しいノードで、以下の作業を行います。

1. クラスターパッケージをインストールします。クラスターが SBD、Booth チケットマネージャー、またはクォーラムデバイスを使用する場合、新しいノードにそれぞれのパッケージ (sbd、ブートサイト、corosync-qdevice)も手動でインストールする必要があります。

```
[root@newnode ~]# yum install -y pcs fence-agents-all
```

2. firewalld デーモンを実行している場合は、以下のコマンドを実行して Red Hat High Availability Add-On で必要なポートを有効にします。

```
# firewall-cmd --permanent --add-service=high-availability
# firewall-cmd --add-service=high-availability
```

3. ユーザー ID hacluster のパスワードを設定します。クラスターの各ノードで、同じパスワードを使用することが推奨されます。

```
[root@newnode ~]# passwd hacluster
Changing password for user hacluster.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

4. 以下のコマンドを実行して pcsd サービスを開始し、システムの起動時に pcsd を有効にします。

```
# systemctl start pcsd.service
# systemctl enable pcsd.service
```

既存クラスターのノードの 1 つで、以下の作業を行います。

1. 新しいクラスターノードで hacluster ユーザーを認証します。

```
[root@clusternode-01 ~]# pcs cluster auth newnode.example.com
Username: hacluster
Password:
newnode.example.com: Authorized
```

2. 新しいノードを既存のクラスターに追加します。また、このコマンドは corosync.conf クラスター設定ファイルをクラスター内のすべてのノード（追加する新しいノードを含む）に同期します。

```
[root@clusternode-01 ~]# pcs cluster node add newnode.example.com
```

クラスターに追加する新しいノードで、以下の作業を行います。

1. 新しいノードで、クラスターサービスを開始して有効にします。

```
[root@newnode ~]# pcs cluster start
Starting Cluster...
[root@newnode ~]# pcs cluster enable
```

2. 新しいクラスターノードに対して、フェンシングデバイスを設定してテストします。フェンスデバイスの設定は *5章 フェンス機能: STONITH の設定* を参照してください。

4.4.4. クラスターノードの削除

次のコマンドは、指定したノードをシャットダウンし、クラスター内の他のすべてのノードで、クラスター設定ファイル **corosync.conf** からそのノードを削除します。クラスターに関するすべての情報をクラスターノード全体で削除し、クラスターを完全に破棄する方法については、「クラスター設定の削除」 を参照してください。

```
pcs cluster node remove node
```

4.4.5. スタンバイモード

以下のコマンドは、指定ノードをスタンバイモードにします。指定ノードはリソースのホストが行えなくなります。ノードで現在アクティブなリソースは、すべて別のノードに移行されます。**--all** を指定すると、このコマンドはすべてのノードをスタンバイモードにします。

リソースのパッケージを更新する場合にこのコマンドを使用します。また、設定をテストして、ノードを実際にシャットダウンせずに復元のシュミレーションを行う場合にも、このコマンドを使用できます。

```
pcs cluster standby node | --all
```

次のコマンドは、指定したノードをスタンバイモードから外します。このコマンドを実行すると、指定ノードはリソースをホストできるようになります。**--all** を指定すると、このコマンドはすべてのノードをスタンバイモードから外します。

```
pcs cluster unstandby node | --all
```

-

pcs cluster standby コマンドを実行すると、指定したノードでのリソースが実行されないことに注意してください。pcs cluster unstandby コマンドを実行すると、指定したノードでリソースを実行できます。このコマンドを実行しても、リソースが必ずしも指定のノードに戻る訳ではありません。その時点でリソースが実行できる場所は、リソースを最初に設定した方法によって異なります。リソース制約の詳細は *7章 リソースの制約* を参照してください。

4.5. ユーザーのパーミッション設定

ユーザー hacluster 以外の特定のユーザーに、クラスターを管理するパーミッションを付与できます。個々のユーザーに付与できるパーミッションには、以下の 2 つのセットがあります。

- 「ネットワーク上でのノードアクセスのパーミッション設定」 で説明しているように、個々のユーザーが Web UI からクラスターを管理でき、ネットワークからノードに接続できる pcs コマンドを実行可能なパーミッション。ネットワーク経由でノードに接続するコマンドには、クラスターを設定するコマンド、またはクラスターからノードを追加または削除するためのコマンドが含まれます。

- 「ACL を使用したローカルパーミッションの設定」 で説明しているように、クラスター設定への読み込み専用または書き込み専用アクセスをローカルユーザーに許可するパーミッション。ネットワーク経由で接続する必要のないコマンドには、リソースの作成や制約の設定など、クラスター設定を編集するコマンドが含まれます。

両方のパーミッションセットが割り当てられている状況では、ネットワーク経由で接続するコマンドのパーミッションが最初に適用され、次にローカルノードのクラスター設定を編集するパーミッションが適用されます。ほとんどの pcs コマンドは、ネットワークアクセスを必要とせず、ネットワークパーミッションが適用されません。

4.5.1. ネットワーク上でのノードアクセスのパーミッション設定

Web UI でクラスターを管理し、ネットワークからノードに接続する pcs コマンドを実行するパーミッションを特定のユーザーに付与するには、それらのユーザーをグループ haclient に追加します。「クラスター管理パーミッションの設定」 で説明しているように、Web UI を使用することで、これらのユーザーにパーミッションを付与することができます。

4.5.2. ACL を使用したローカルパーミッションの設定

Red Hat Enterprise Linux 7.1 より、pcs acl コマンドを使用してローカルユーザーのパーミッションを設定し、アクセス制御リスト(ACL)を使用してクラスター設定への読み取り専用アクセスまたは読み書きアクセスを許可できます。また、「ACL の設定」 で説明しているように、pcsd Web UI を使用

して **ACL** を設定することも可能です。デフォルトでは、**root** ユーザーと、**haclient** グループのメンバーユーザーは、クラスター設定への完全なローカル読み取り/書き込みアクセスを持ちます。

ローカルユーザーのパーミッションを設定するには、以下の **2** つの手順を実行します。

1. **pcs acl role create...** コマンドを実行して、その *ロール* のパーミッションを定義するロールを作成します。

2. **pcs acl user create** コマンドで作成したロールをユーザーに割り当てます。

以下の例では、**rouser** という名前のローカルユーザーに、クラスター設定に対する読み取り専用アクセスを提供します。

1. この手順では、**rouser** ユーザーがローカルシステムに存在し、**rouser** ユーザーが **haclient** グループのメンバーである必要があります。

```
# adduser rouser
# usermod -a -G haclient rouser
```

2. **enable-acl** クラスタープロパティーで **Pacemaker ACL** を有効にします。

```
# pcs property set enable-acl=true --force
```

3. **cib** に対して読み取り専用権限を持つ **read-only** という名前のロールを作成します。

```
# pcs acl role create read-only description="Read access to cluster" read xpath /cib
```

4. **pcs ACL** システムで **rouser** ユーザーを作成し、そのユーザーに 読み取り専用 ロールを割り当てます。

```
# pcs acl user create rouser read-only
```

5. 現在の **ACL** を表示します。

```
# pcs acl
```

```
User: rouser
  Roles: read-only
Role: read-only
  Description: Read access to cluster
  Permission: read xpath /cib (read-only-read)
```

以下の例では、**wuser** という名前のローカルユーザーにクラスター設定の書き込みアクセスを提供します。

1. この手順では、**wuser** ユーザーがローカルシステムに存在し、**wuser** ユーザーが **haclient** グループのメンバーである必要があります。

```
# adduser wuser
# usermod -a -G haclient wuser
```

2. **enable-acl** クラスタープロパティーで **Pacemaker ACL** を有効にします。

```
# pcs property set enable-acl=true --force
```

3. **cib** に対して書き込みパーミッションを持つ **write-access** という名前のロールを作成します。

```
# pcs acl role create write-access description="Full access" write xpath /cib
```

4. **pcs ACL** システムで **wuser** ユーザーを作成し、そのユーザーに **write-access** ロールを割り当てます。

```
# pcs acl user create wuser write-access
```

5. 現在の **ACL** を表示します。

```
# pcs acl
User: rouser
  Roles: read-only
User: wuser
  Roles: write-access
Role: read-only
  Description: Read access to cluster
  Permission: read xpath /cib (read-only-read)
```

```
Role: write-access
  Description: Full Access
  Permission: write xpath /cib (write-access-write)
```

クラスター **ACL** の詳細は、**pcs acl** コマンドのヘルプ画面を参照してください。

4.6. クラスター設定の削除

クラスター設定ファイルをすべて削除し全クラスターサービスを停止、クラスターを永久的に破棄する場合は次のコマンドを使用します。

警告

作成したクラスター設定をすべて永久に削除します。クラスターを破棄する前に、**pcs cluster stop** を実行することが推奨されます。

```
pcs cluster destroy
```

4.7. クラスターの状態表示

次のコマンドで現在のクラスターとクラスターリソースの状態を表示します。

```
pcs status
```

次のコマンドを使用すると現在のクラスターの状態に関するサブセット情報を表示させることができます。

このコマンドはクラスターの状態を表示しますが、クラスターリソースの状態は表示しません。

```
pcs cluster status
```

クラスターリソースの状態は次のコマンドで表示させます。

pcs status resources

4.8. クラスターメンテナンス

クラスターのノードでメンテナンスを実行するには、そのクラスターで実行しているリソースおよびサービスを停止するか、または移行する必要がある場合があります。または、サービスを変更しない状態で、クラスターソフトウェアの停止が必要になる場合があります。Pacemaker は、システムメンテナンスを実行するための様々な方法を提供します。

- クラスターの別のノードでサービスが継続的に実行している状態で、クラスター内のノードを停止する必要がある場合は、そのクラスターノードをスタンバイモードにすることができます。スタンバイノードのノードは、リソースをホストすることができなくなります。ノードで現在アクティブなリソースは、別のノードに移行するか、(他のノードがそのリソースを実行できない場合は) 停止します。

  スタンバイモードの詳細は、「スタンバイモード」 を参照してください。

- リソースを停止せずに、現在実行しているノードから個別のリソースを移行する必要がある場合は、pcs resource move コマンドを使用してリソースを別のノードに移行できます。pcs resource move コマンドの詳細は、「リソースを手作業で移動する」 を参照してください。

  pcs resource move コマンドを実行すると、現在実行しているノードでリソースが実行されないように、制約がリソースに追加されます。リソースを戻す準備ができたら、pcs resource clear または pcs constraint delete コマンドを実行して制約を削除できます。ただし、このコマンドを実行しても、リソースが必ずしも元のノードに戻る訳ではありません。その時点でリソースが実行できる場所は、リソースを最初に設定した方法によって異なるためです。「現在のノードからリソースを移動」 で説明しているように、pcs resource relocate run コマンドを使用すると、リソースを指定のノードに移動できます。

- 実行中のリソースを完全に停止し、クラスターが再び起動しないようにする必要がある場合は、pcs resource disable コマンドを使用できます。pcs resource disable コマンドの詳細は、「クラスターリソースの有効化、無効化、および禁止」 を参照してください。

- Pacemaker がリソースに対するアクションを実行するのを防ぐ必要がある場合（たとえば、リソースに対するメンテナンスの実行中に復元アクションを無効にする必要がある場合や、/etc/sysconfig/pacemaker 設定をリロードする必要がある場合）、「管理リソース」 の説明にあるように pcs resource unmanage コマンドを使用します。Pacemaker Remote 接続リソースは、非管理モードにしないでください。

- クラスターを、サービスが開始または停止されない状態にする必要がある場合

は、maintenance-mode クラスタープロパティーを設定できます。クラスターをメンテナンスモードにすると、すべてのリソースが自動的に非管理モードになります。クラスターのプロパティーの詳細は 表12.1「クラスターのプロパティー」 を参照してください。

- Pacemaker リモートノードでメンテナンスを実行する必要がある場合、「システムアップグレードおよび pacemaker_remote」 で説明しているように、リモートノードリソースを無効にすることで、ノードをクラスターから削除できます。

第5章 フェンス機能: STONITH の設定

STONITH は Shoot The Other Node In The Head の頭字語で、不安定なノードや同時アクセスによるデータの破損を防ぐことができます。

ノードが無反応だからと言ってデータにアクセスしていないとは限りません。STONITH を使ってノードを排他処理することが唯一 100% 確実にデータの安全を確保する方法になります。排他処理することによりそのノードを確実にオフラインにしてから、別のノードに対してデータへのアクセスを許可することができます。

STONITH はクラスター化したサービスを停止できない場合にも役に立ちます。この場合は、クラスターが STONITH を使用してノード全体を強制的にオフラインにし、その後サービスを別の場所で開始すると安全です。

フェンシングの概要と、Red Hat High Availability クラスターにおけるフェンシングの重要性は**RHEL 高可用性クラスターでフェンシングが重要なのはなぜですか ?**を参照してください。

5.1. STONITH (フェンス) エージェント

以下のコマンドは、利用可能な STONITH エージェントをリスト表示します。フィルターを指定するとフィルターに一致する STONITH エージェントのみを表示します。

```
pcs stonith list [filter]
```

5.2. フェンスデバイスの一般的なプロパティー

クラスターノードは、フェンスリソースが開始しているかどうかに関わらず、フェンスデバイスでその他のクラスターノードをフェンスできます。以下の例外を除き、リソースが開始しているかどうかは、デバイスの定期的なモニターのみを制御するものとなり、使用可能かどうかは制御しません。

- フェンスデバイスを無効にするには、pcs stonith disable *stonith_id* コマンドを実行します。これにより、ノードがそのデバイスを使用できないようにすることができます。
- 特定のノードがフェンスデバイスを使用できないようにするには、pcs constraint location ... avoids コマンドを使用して、フェンシングリソースの場所制約を設定できます。
- stonith-enabled=false を設定すると、フェンシングがすべて無効になります。ただし、実

稼働環境でフェンシングを無効にすることは適していないため、フェンシングが無効になっている場合は、Red Hat ではクラスターがサポートされないことに注意してください。

表5.1「フェンスデバイスの一般的なプロパティー」 は、フェンスデバイスに設定できる一般的なプロパティーを説明します。特定のフェンスデバイスに設定できるフェンスプロパティーについては 「デバイス固有のフェンスオプションの表示」 を参照してください。

注記

より高度なフェンス設定プロパティーについては 「その他のフェンス設定オプション」 を参照してください。

表5.1 フェンスデバイスの一般的なプロパティー

フィールド	タイプ	デフォルト	説明
pcmk_host_map	文字列		ホスト名を、ホスト名に対応していないデバイスのポート番号へマッピングします。たとえば、node1:1 **;node2:2,** 3 の場合は、node1 にはポート 1 を使用し、node2 にはポート 2 と 3 を使用するようにクラスターに指示します。
pcmk_host_list	文字列		このデバイスによって制御されるマシンのリスト(**pcmk_host_check=static-list**以外オプション)。
pcmk_host_check	文字列	dynamic-list	デバイスで制御するマシンを指定します。許可される値： **dynamic-list** （デバイスのクエリー）、 **static-list** (**pcmk_host_list** 属性の確認)、none （すべてのデバイスがすべてのマシンをフェンスできると仮定）

5.3. デバイス固有のフェンスオプションの表示

指定した STONITH エージェントのオプションを表示するには、次のコマンドを使用します。

```
pcs stonith describe stonith_agent
```

次のコマンドでは Telnet または SSH 経由の APC 用フェンスエージェントのオプションを表示します。

```
# pcs stonith describe fence_apc
```

```
Stonith options for: fence_apc
  ipaddr (required): IP Address or Hostname
  login (required): Login Name
  passwd: Login password or passphrase
  passwd_script: Script to retrieve password
  cmd_prompt: Force command prompt
  secure: SSH connection
  port (required): Physical plug number or name of virtual machine
  identity_file: Identity file for ssh
  switch: Physical switch number on device
  inet4_only: Forces agent to use IPv4 addresses only
  inet6_only: Forces agent to use IPv6 addresses only
  ipport: TCP port to use for connection with device
  action (required): Fencing Action
  verbose: Verbose mode
  debug: Write debug information to given file
  version: Display version information and exit
  help: Display help and exit
  separator: Separator for CSV created by operation list
  power_timeout: Test X seconds for status change after ON/OFF
  shell_timeout: Wait X seconds for cmd prompt after issuing command
  login_timeout: Wait X seconds for cmd prompt after login
  power_wait: Wait X seconds after issuing ON/OFF
  delay: Wait X seconds before fencing is started
  retry_on: Count of attempts to retry power on
```

警告

method オプションを提供するフェンスエージェントでは cycle の値がサポートされないため、データの破損が発生する可能性があるため、この値は指定できません。

5.4. フェンスデバイスの作成

次のコマンドで stonith デバイスを作成します。

```
pcs stonith create stonith_id stonith_device_type [stonith_device_options]
```

```
# pcs stonith create MyStonith fence_virt pcmk_host_list=f1 op monitor interval=30s
```

1 つのノードのみをフェンスできるフェンスデバイスや、複数のノードをフェンスできるデバイスもあります。フェンスデバイスの作成時に指定するパラメーターは、フェンスデバイスが対応しているか、必要としているかにより異なります。

- フェンスデバイスの中には、フェンスできるノードを自動的に判断できるものがあります。

- フェンスデバイスの作成時に pcmk_host_list パラメーターを使用すると、フェンスデバイスで制御されるすべてのマシンを指定できます。

- フェンスデバイスによっては、フェンスデバイスが理解する仕様へのホスト名のマッピングが必要となるものがあります。フェンスデバイスの作成時に、pcmk_host_map パラメーターを使用してホスト名をマッピングできます。

pcmk_host_list パラメーターおよび pcmk_host_map パラメーターの詳細は、表5.1「フェンスデバイスの一般的なプロパティー」 を参照してください。

フェンスデバイスを設定したら、デバイスをテストして正しく機能していることを確認してください。フェンスデバイスのテストの詳細は 「フェンスデバイスのテスト」 を参照してください。

5.5. フェンスデバイスの表示

以下のコマンドは現在設定されているフェンスデバイスをすべて表示します。*stonith_id* が指定されていると、指定された stonith デバイスのオプションのみが表示されます。--full オプションを指定すると、設定した stonith オプションがすべて表示されます。

```
pcs stonith show [stonith_id] [--full]
```

5.6. フェンスデバイスの修正と削除

現在設定されているフェンスデバイスのオプションを修正したり、新たにオプションを追加する場合は次のコマンドを使用します。

```
pcs stonith update stonith_id [stonith_device_options]
```

現在の設定からフェンスデバイスを削除する場合は次のコマンドを使用します。

```
pcs stonith delete stonith_id
```

5.7. フェンスデバイスが接続されているノードの管理

次のコマンドで、ノードを手動でフェンスできます。**--off** を指定すると、**stonith** への **off API** 呼び出しが使用され、ノードを再起動する代わりにオフになります。

```
pcs stonith fence node [--off]
```

ノードがアクティブでない場合でも、そのノードを **stonith** デバイスがフェンスできない状況では、そのノードのリソースをクラスターが復旧できない可能性があります。この場合は、ノードの電源が切れたことを手動で確認した後、次のコマンドを入力して、ノードの電源が切れたことをクラスターに確認し、そのリソースを回復のために解放できます。

警告

指定したノードが実際にオフになっていない状態で、クラスターソフトウェア、または通常クラスターが制御するサービスを実行すると、データ破損またはクラスター障害が発生します。

```
pcs stonith confirm node
```

5.8. その他のフェンス設定オプション

フェンスデバイスに設定できるその他のプロパティーは 表5.2「フェンスデバイスの高度なプロパティー」 にまとめられています。これらのオプションは高度な設定を行う場合にのみ使用されます。

表5.2 フェンスデバイスの高度なプロパティー

フィールド	タイプ	デフォルト	説明
pcmk_host_argument	文字列	port	port の代替パラメーターです。デバイスによっては、標準の port パラメーターに対応していない場合や、そのデバイス固有のパラメーターも提供している場合があります。このパラメーターを使用して、フェンスするマシンを示すデバイス固有の代替パラメーターを指定します。クラスターが追加パラメーターを提供しないようにする場合は、**none** 値を使用します。

フィールド	タイプ	デフォルト	説明
pcmk_reboot_action	文字列	reboot	**再起動** の代わりに実行する別のコマンド。標準的なコマンドに対応していないデバイスや、別のコマンドを提供しているデバイスがあります。このパラメーターを使用して、再起動を実行するデバイス固有の代替コマンドを指定します。
pcmk_reboot_timeout	時間	60s	**stonith-timeout** の代わりに、再起動アクションに使用するタイムアウトを指定します。再起動が完了するまでに通常より長い時間を要するデバイスもあれば、通常より短い時間で完了するデバイスもあります。このパラメーターを使用して、再起動にデバイス固有のタイムアウトを指定します。
pcmk_reboot_retries	整数	2	タイムアウト期間内に、**reboot** コマンドを再試行する回数の上限です。複数の接続に対応していないデバイスもあります。デバイスが別のタスクでビジー状態になると操作が失敗する場合があるため、タイムアウトに達していなければ、Pacemaker が操作を自動的に再試行します。Pacemaker による再起動の動作の再試行回数を変更する場合に使用します。
pcmk_off_action	文字列	off	**オフ** の代わりに実行する代替コマンド。標準的なコマンドに対応していないデバイスや、別のコマンドを提供しているデバイスがあります。このような場合は、このパラメーターを使用して、オフ操作を実行するデバイス固有のコマンドを指定します。
pcmk_off_timeout	時間	60s	**stonith-timeout** の代わりに、オフアクションに使用する別のタイムアウトを指定します。デバイスによって、この操作が完了するのにかかる時間が、通常と大きく異なる場合があります。このパラメーターを使用して、オフ操作にデバイス固有のタイムアウトを指定します。
pcmk_off_retries	整数	2	タイムアウト期間内に、off コマンドを再試行する回数の上限です。複数の接続に対応していないデバイスもあります。デバイスが別のタスクでビジー状態になると操作が失敗する場合があるため、タイムアウトに達していなければ、Pacemaker が操作を自動的に再試行します。Pacemaker によるオフ動作の再試行回数を変更する場合に使用します。

フィールド	タイプ	デフォルト	説明
pcmk_list_action	文字列	list	**list** の代わりに実行する別のコマンド。標準的なコマンドに対応していないデバイスや、別のコマンドを提供しているデバイスがあります。このような場合は、このパラメーターを使用して、list 操作を実行するデバイス固有のコマンドを指定します。
pcmk_list_timeout	時間	60s	**stonith-timeout** の代わりに、list 操作に使用する別のタイムアウトを指定します。デバイスによって、この操作が完了するのにかかる時間が、通常と大きく異なる場合があります。このパラメーターを使用して、list 操作にデバイス固有のタイムアウトを指定します。
pcmk_list_retries	整数	2	タイムアウト期間内に、**list** コマンドを再試行する回数の上限です。複数の接続に対応していないデバイスもあります。デバイスが別のタスクでビジー状態になると操作が失敗する場合があるため、タイムアウトに達していなければ、Pacemaker が操作を自動的に再試行します。Pacemaker による list 動作の再試行回数を変更する場合に使用します。
pcmk_monitor_action	文字列	monitor	**monitor** の代わりに実行する代替コマンド。標準的なコマンドに対応していないデバイスや、別のコマンドを提供しているデバイスがあります。このような場合は、このパラメーターを使用して、監視操作を実行するデバイス固有のコマンドを指定します。
pcmk_monitor_timeout	時間	60s	**stonith-timeout** の代わりに、監視アクションに使用する別のタイムアウトを指定します。デバイスによって、この操作が完了するのにかかる時間が、通常と大きく異なる場合があります。このパラメーターを使用して、監視操作にデバイス固有のタイムアウトを指定します。
pcmk_monitor_retries	整数	2	タイムアウト期間内に、**monitor** コマンドを再試行する回数の上限です。複数の接続に対応していないデバイスもあります。デバイスが別のタスクでビジー状態になると操作が失敗する場合があるため、タイムアウトに達していなければ、Pacemaker が操作を自動的に再試行します。Pacemaker による監視操作の再試行回数を変更する場合に使用します。

フィールド	タイプ	デフォルト	説明
pcmk_status_action	文字列	status	**status** の代わりに実行する代替コマンド。標準的なコマンドに対応していないデバイスや、別のコマンドを提供しているデバイスがあります。このような場合は、このパラメーターを使用して、status 操作を実行するデバイス固有のコマンドを指定します。
pcmk_status_timeout	時間	60s	**stonith-timeout** の代わりに、ステータス動作に使用する別のタイムアウトを指定します。デバイスによって、この操作が完了するのにかかる時間が、通常と大きく異なる場合があります。このパラメーターを使用して、status 操作にデバイス固有のタイムアウトを指定します。
pcmk_status_retries	整数	2	タイムアウト期間内に、status コマンドを再試行する回数の上限です。複数の接続に対応していないデバイスもあります。デバイスが別のタスクでビジー状態になると操作が失敗する場合があるため、タイムアウトに達していなければ、Pacemaker が操作を自動的に再試行します。Pacemaker による status 動作の再試行回数を変更する場合に使用します。
pcmk_delay_base	時間	0s	stonith 操作のベース遅延を有効にし、ベース遅延の値を指定します。ノードの数が偶数になるクラスターでは、遅延を設定すると、均等の分割時に同時にノードが互いにフェンシングするのを回避できます。ランダムな遅延は、すべてのノードに同じフェンスデバイスが使用されている場合に役に立つことがあります。また、静的遅延を変更すると、各ノードで異なるデバイスが使用される場合に各フェンシングデバイスで役に立つことがあります。全体の遅延は、合計が最大遅延を下回るように、ランダムな遅延値に静的遅延を加算します。**pcmk_delay_base** を設定し、**pcmk_delay_max** を設定しない場合は、遅延にランダムなコンポーネントがなく、**pcmk_delay_base** の値になります。 個々のフェンスエージェントには delay パラメーターが実装されています。これは、**pcmk_delay_*** プロパティーで設定された遅延とは依存しません。この遅延の両方が設定されている場合は、その両方が一緒に追加されるため、通常は併用されません。

フィールド	タイプ	デフォルト	説明
pcmk_delay_max	時間	0s	stonith 動作のランダムな遅延を有効にし、ランダムな遅延の最大値を指定します。ノードの数が偶数になるクラスターでは、遅延を設定すると、均等の分割時に同時にノードが互いにフェンシングするのを回避できます。ランダムな遅延は、すべてのノードに同じフェンスデバイスが使用されている場合に役に立つことがあります。また、静的遅延を変更すると、各ノードで異なるデバイスが使用される場合に各フェンシングデバイスで役に立つことがあります。全体の遅延は、合計が最大遅延を下回るように、このランダムな遅延値に静的遅延を加算します。**pcmk_delay_max** を設定し、**pcmk_delay_base** を設定しない場合は、静的コンポーネントが遅延に含まれません。 個々のフェンスエージェントには delay パラメーターが実装されています。これは、**pcmk_delay_*** プロパティーで設定された遅延とは依存しません。この遅延の両方が設定されている場合は、その両方が一緒に追加されるため、通常は併用されません。
pcmk_action_limit	整数	1	このデバイスで並行して実行できる操作の上限です。最初に、クラスタープロパティーの **concurrent-fencing=true** を設定する必要があります。値を -1 にすると無制限になります。
pcmk_on_action	文字列	on	高度な使用のみ： **on** の代わりに実行する代替コマンド。標準的なコマンドに対応していないデバイスや、別のコマンドを提供しているデバイスがあります。このパラメーターを使用して、**on** アクションを実装するデバイス固有のコマンドを指定します。
pcmk_on_timeout	時間	60s	高度な使用のみ： **stonith-timeout** の代わりに、**on** 操作に使用する別のタイムアウトを指定します。デバイスによって、この操作が完了するのにかかる時間が、通常と大きく異なる場合があります。このパラメーターを使用して、**on** 操作にデバイス固有のタイムアウトを指定します。
pcmk_on_retries	整数	2	高度な使用のみ：タイムアウト期間内に、**on** コマンドを再試行する最大回数。複数の接続に対応していないデバイスもあります。デバイスが別のタスクでビジー状態になると操作が失敗する場合があるため、タイムアウトに達していなければ、Pacemaker が操作を自動的に再試行します。Pacemaker による **on** 動作の再試行回数を変更する場合に使用します。

フィールド	タイプ	デフォルト	説明

表12.1「クラスターのプロパティー」 の説明にあるように、fence-reaction クラスタープロパティーを設定すると、クラスターノードが独自のフェンシングの通知を受信した場合にどのように反応するかを決定できます。クラスターノードは、フェンシングの設定が間違っている場合に独自のフェンシングの通知を受信するか、ファブリックフェンシングがクラスター通信を遮断しない状態である可能性があります。このプロパティーのデフォルト値は stop (Pacemaker をすぐに停止して停止し続ける) ですが、この値に最も安全な選択肢は panic で、ローカルノードをすぐに再起動しようとします。停止動作を希望する場合は、おそらくファブリックフェンシングと併用する場合は、明示的に指定することが推奨されます。

5.9. フェンスレベルの設定

Pacemaker は、フェンストポロジーと呼ばれる機能を用いて、複数デバイスでのノードのフェンシングに対応します。トポロジーを実装するには、通常の方法で各デバイスを作成し、設定のフェンストポロジーセクションでフェンスレベルを 1 つ以上定義します。

- レベルは、1 から昇順で試行されていきます。
- デバイスに障害が発生すると、現在のレベルの処理が終了します。同レベルのデバイスには試行されず、次のレベルが試行されます。
- すべてのデバイスのフェンシングが正常に完了すると、そのレベルが継承され、他のレベルは試行されなくなります。
- いずれかのレベルで成功するか、すべてのレベルが試行され失敗すると、操作は終了します。

ノードにフェンスレベルを追加する場合は、次のコマンドを使用します。デバイスは、stonith ID をコンマで区切って指定します。stonith ID が、指定したレベルで試行されます。

```
pcs stonith level add level node devices
```

次のコマンドを使用すると現在設定されている全フェンスレベルが表示されます。

```
pcs stonith level
```

以下の例では、ノード **rh7- 2** にフェンスデバイスが **my_ilo** と呼ばれる **ilo** フェンスデバイスと、**my_apc** という **apc** フェンスデバイスが設定されています。このコマンドはフェンスレベルを設定し、デバイス **my_ilo** に障害が発生し、ノードがフェンスできない場合に、**Pacemaker** がデバイス **my_apc** の使用を試みるようにします。この例では、レベル設定後の **pcs stonith level** コマンドの出力も示しています。

```
# pcs stonith level add 1 rh7-2 my_ilo
# pcs stonith level add 2 rh7-2 my_apc
# pcs stonith level
 Node: rh7-2
  Level 1 - my_ilo
  Level 2 - my_apc
```

次のコマンドは、指定したノードおよびデバイスのフェンスレベルを削除します。ノードやデバイスを指定しないと、指定したフェンスレベルがすべてのノードから削除されます。

```
pcs stonith level remove level [node_id] [stonith_id] ... [stonith_id]
```

以下のコマンドを使用すると、指定したノードや **stonith id** のフェンスレベルが削除されます。ノードや **stonith id** を指定しないと、すべてのフェンスレベルが削除されます。

```
pcs stonith level clear [node|stonith_id(s)]
```

複数の **stonith ID** を指定する場合はコンマで区切って指定します。空白は入力しないでください。以下に例を示します。

```
# pcs stonith level clear dev_a,dev_b
```

次のコマンドは、フェンスレベルで指定されたフェンスデバイスとノードがすべて存在することを確認します。

```
pcs stonith level verify
```

Red Hat Enterprise Linux 7.4 では、フェンストポロジーのノードは、ノード名に適用する正規表現

と、ノードの属性 (およびその値) で指定できます。たとえば、以下のコマンドでは、ノード node1、node2、およびnode3 を、フェンスデバイス apc1 およびapc2 を使用するように、ノードnode4、node5、およびnode6 を設定して、フェンスデバイス apc3 およびapc4 を使用します。

```
pcs stonith level add 1 "regexp%node[1-3]" apc1,apc2
pcs stonith level add 1 "regexp%node[4-6]" apc3,apc4
```

次のコマンドでは、ノード属性のマッチングを使用して、同じように設定します。

```
pcs node attribute node1 rack=1
pcs node attribute node2 rack=1
pcs node attribute node3 rack=1
pcs node attribute node4 rack=2
pcs node attribute node5 rack=2
pcs node attribute node6 rack=2
pcs stonith level add 1 attrib%rack=1 apc1,apc2
pcs stonith level add 1 attrib%rack=2 apc3,apc4
```

5.10. 冗長電源のフェンシング設定

冗長電源にフェンシングを設定する場合は、ホストを再起動するときに、クラスターが、最初に両方の電源をオフにしてから、いずれかの電源をオンにするようにする必要があります。

ノードの電源が完全にオフにならないと、ノードがリソースを解放しない場合があります。このとき、解放できなかったリソースに複数のノードが同時にアクセスして、リソースが破損する可能性があります。

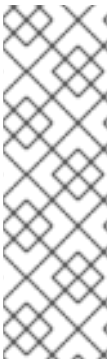
Red Hat Enterprise Linux 7.2 より古いバージョンでは、オンまたはオフのアクションを使用するデバイスで異なるバージョンを明示的に設定する必要がありました。Red Hat Enterprise Linux 7.2 では、以下の例のように各デバイスを 1 度定義して、ノードのフェンシングに両方が必要であることを指定するだけで済むようになりました。

```
# pcs stonith create apc1 fence_apc_snmp ipaddr=apc1.example.com login=user
passwd='7a4D#1j!pz864' pcmk_host_map="node1.example.com:1;node2.example.com:2"

# pcs stonith create apc2 fence_apc_snmp ipaddr=apc2.example.com login=user
passwd='7a4D#1j!pz864' pcmk_host_map="node1.example.com:1;node2.example.com:2"

# pcs stonith level add 1 node1.example.com apc1,apc2
# pcs stonith level add 1 node2.example.com apc1,apc2
```

5.11. 統合フェンスデバイスで使用する ACPI の設定

クラスターが統合フェンスデバイスを使用する場合は、即時かつ完全なフェンシングを実行できるように、ACPI (Advanced Configuration and Power Interface) を設定する必要があります。

クラスターノードが統合フェンスデバイスでフェンシングされるように設定されている場合は、そのノードの ACPI Soft-Off を無効にします。ACPI Soft-Off を無効にすると、統合フェンスデバイスはクリーンシャットダウンを試行するのではなく、ノードを即時かつ完全にオフにできます（例：shutdown -h now）。それ以外の場合は、ACPI Soft-Off が有効になっていると、統合フェンスデバイスがノードをオフにするのに 4 秒以上かかることがあります (以下の注記部分を参照してください)。さらに、ACPI Soft-Off が有効になっていて、ノードがシャットダウン時にパニック状態になるか、フリーズすると、統合フェンスデバイスがノードをオフにできない場合があります。このような状況では、フェンシングが遅延するか、失敗します。したがって、ノードが統合フェンスデバイスでフェンシングされ、ACPI Soft-Off が有効になっている場合は、クラスターが徐々に復元します。または管理者の介入による復旧が必要になります。



注記

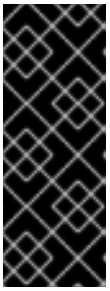
ノードのフェンシングにかかる時間は、使用している統合フェンスデバイスによって異なります。統合フェンスデバイスの中には、電源ボタンを押し続けるのと同じ動作を実行するものもあります。この場合は、ノードがオフになるのに 4 秒から 5 秒かかります。また、電源ボタンを押してすぐ離すのと同等の動作を行い、ノードの電源をオフにする行為をオペレーティングシステムに依存する統合フェンスデバイスもあります。この場合は、ノードがオフになるのにかかる時間は 4~ 5 秒よりも長くなります。

- ACPI Soft-Off を無効にする場合は、BIOS 設定を instant-off、またはこれに類似する設定に変更することが推奨されます。これにより、「BIOS で ACPI Soft-Off を無効化」 で説明しているように、ノードは遅延なくオフになります。

システムによっては、BIOS で ACPI Soft-Off を無効にできません。お使いのクラスターでは、BIOS で ACPI Soft-Off を無効にできない場合に、以下のいずれかの方法で ACPI Soft-Off を無効にできます。

- 「logind.conf ファイルで ACPI Soft-Off の無効化」 の説明に従って、HandlePowerKey=ignore を /etc/systemd/logind.conf ファイルに設定し、ノードがフェンシングされるとすぐにオフになることを確認します。これが、ACPI Soft-Off を無効にする 1 つ目の代替方法です。

- acpi=off で説明されているように、カーネル起動コマンドラインに 「GRUB 2 ファイルでの ACPI の完全な無効化」 を追加します。これは、ACPI Soft-Off を無効にする 2 つ目の代替方法です。この方法の使用が推奨される場合、または 1 つ目の代替方法が利用できない場合に使用してください。



重要

この方法は、ACPI を完全に無効にします。コンピューターの中には、ACPI が完全が無効になってるとシステムが正しく起動しないものもあります。お使いのクラスターに適した方法が他にない場合に 限り、この方法を使用してください。

5.11.1. BIOS で ACPI Soft-Off を無効化

以下の手順で、各クラスターノードの BIOS を設定して、ACPI Soft-Off を無効にできます。

注記

BIOS で ACPI Soft-Off を無効にする手順は、サーバーシステムにより異なる場合があります。この手順は、お使いのハードウェアのドキュメントで確認する必要があります。

1. ノードを再起動して、BIOS CMOS セットアップユーティリティー プログラムを起動します。

2. 電源メニュー（または同等の電源 管理メニュー）に移動します。

3. Power メニューで、PWR-BTN 関数による Soft-Off （またはそれと同等の機能）を Instant-Off に設定します（または、遅延なく電源ボタンでノードをオフにする同等の設定）。例5.1「BIOS CMOS セットアップユーティリティー:PWR-BTTN による Soft-Off を Instant-Offに設定します。」 PWR-BTTN による ACPI Function が Enabled に、Soft-Off が Instant-Off に設定された Power メニューを表示します。

注記

PWR-BTN に よる ACPI 関数、Soft-Off、および Instant- Off に相当するものは、コンピューターごとに異なる場合があります。ただし、この手順の目的は、電源ボタンを使用して遅延なしにコンピューターをオフにするように BIOS を設定することです。

4. BIOS CMOS セットアップユーティリティー プログラムを終了し、BIOS 設定を保存します。

5.

ノードがフェンシングされるとすぐにオフになることを確認します。フェンスデバイスのテストの詳細は 「フェンスデバイスのテスト」 を参照してください。

例5.1 BIOS CMOS セットアップユーティリティー:PWR-BTTN による Soft-Off を Instant-Offに設定します。

```
+---------------------------------------------|------------------+
|   ACPI Function             [Enabled]       |    Item Help     |
|   ACPI Suspend Type         [S1(POS)]       |------------------|
| x Run VGABIOS if S3 Resume   Auto           |   Menu Level   * |
|   Suspend Mode              [Disabled]      |                  |
|   HDD Power Down            [Disabled]      |                  |
|   Soft-Off by PWR-BTTN      [Instant-Off    |                  |
|   CPU THRM-Throttling       [50.0%]         |                  |
|   Wake-Up by PCI card       [Enabled]       |                  |
|   Power On by Ring          [Enabled]       |                  |
|   Wake Up On LAN            [Enabled]       |                  |
| x USB KB Wake-Up From S3     Disabled       |                  |
|   Resume by Alarm           [Disabled]      |                  |
| x  Date(of Month) Alarm       0             |                  |
| x  Time(hh:mm:ss) Alarm       0 :  0 :      |                  |
|   POWER ON Function         [BUTTON ONLY    |                  |
| x KB Power ON Password       Enter          |                  |
| x Hot Key Power ON           Ctrl-F1        |                  |
|                                 |                |
|                                 |                |
+---------------------------------------------|------------------+
```

以下の例は、ACPI Function を Enabled に設定し、PWR-BTTN による Soft-Off を Instant- Off に設定していることを示しています。

5.11.2. logind.conf ファイルで ACPI Soft-Off の無効化

/etc/systemd/logind.conf ファイルで電源キーの処理を無効にするには、以下の手順を行います。

1. /etc/systemd/logind.conf ファイルで以下の設定を定義します。

```
HandlePowerKey=ignore
```

2. systemd 設定をリロードします。

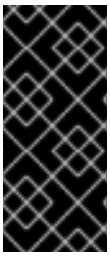
```
# systemctl daemon-reload
```

3.

ノードがフェンシングされるとすぐにオフになることを確認します。フェンスデバイスのテストの詳細は 「フェンスデバイスのテスト」 を参照してください。

5.11.3. GRUB 2 ファイルでの ACPI の完全な無効化

カーネルの GRUB メニューエントリーに acpi=off を追加すると、ACPI Soft-Off を無効にできます。



重要

この方法は、ACPI を完全に無効にします。コンピューターの中には、ACPI が完全が無効になってるとシステムが正しく起動しないものもあります。お使いのクラスターに適した方法が他にない場合に 限り、この方法を使用してください。

以下の手順で、GRUB 2 ファイルで ACPI を無効にします。

1. 以下のように --args オプションを grubby ツールの --update-kernel オプションと組み合わせて使用し、各クラスターノードの grub.cfg ファイルを変更します。

```
# grubby --args=acpi=off --update-kernel=ALL
```

GRUB 2 の概要は、 システム管理者のガイド のGRUB 2 での作業を参照してください。

2. ノードを再起動します。

3. ノードがフェンシングされるとすぐにオフになることを確認します。フェンスデバイスのテストの詳細は 「フェンスデバイスのテスト」 を参照してください。

5.12. フェンスデバイスのテスト

フェンシングは、Red Hat Cluster インフラストラクチャーの基本的な部分を設定しているため、フェンシングが適切に機能していることの確認またはテストを行うことは重要です。

以下の手順で、フェンスデバイスをテストします。

1.

デバイスへの接続に使用する ssh、telnet、HTTP などのリモートプロトコルを使用して、手動でログインしてフェンスデバイスをテストしたり、出力される内容を確認します。たとえば、IPMI 対応デバイスのフェンシングを設定する場合は、ipmitool を使用してリモートでログインしてみてください。手動でログインする際に使用するオプションに注意してください。これらのオプションは、フェンスエージェントを使用する際に必要になる場合があります。

フェンスデバイスにログインできない場合は、そのデバイスが ping 可能であること、ファイアウォール設定フェンスデバイスへのアクセスを妨げていないこと、フェンスエージェントでリモートアクセスが有効になっていること、認証情報が正しいことなどを確認します。

2.

フェンスエージェントスクリプトを使用して、フェンスエージェントを手動で実行します。フェンスエージェントを実行するのに、クラスターサービスが実行している必要はないため、デバイスをクラスターに設定する前にこのステップを完了できます。これにより、先に進む前に、フェンスデバイスが適切に応答することを確認できます。

注記

本セクションの例では、iLO デバイスの fence_ilo フェンスエージェントスクリプトを使用します。実際に使用するフェンスエージェントと、そのエージェントを呼び出すコマンドは、お使いのサーバーハードウェアによって異なります。指定するオプションを確認するには、フェンスエージェントの man ページを参照してください。通常は、フェンスデバイスのログイン、パスワードなどの情報と、その他のフェンスデバイスに関する情報を把握しておく必要があります。

以下の例は、-o status パラメーターを指定して fence_ilo フェンスエージェントスクリプトを実行する場合に使用する形式になります。ノードの再起動を試行する前にデバイスをテストして、動作させることができます。このコマンドを実行する際に、iLO デバイスの電源をオン/オフにするパーミッションを持つ iLO ユーザーの名前およびパスワードを指定します。

```
# fence_ilo -a ipaddress -l username -p password -o status
```

以下の例は、-o reboot パラメーターを指定して fence_ilo フェンスエージェントスクリプトを実行するために使用する形式になります。このコマンドをノードで実行すると、フェンスエージェントを設定した別のノードが再起動します。

```
# fence_ilo -a ipaddress -l username -p password -o reboot
```

フェンスエージェントがステータス、オフ、オン、または再起動の動作を適切に実行しない場合は、ハードウェア、フェンスデバイスの設定、およびコマンドの構文を確認する必要があります。さらに、デバッグ出力を有効にした状態で、フェンスエージェントスクリプトを実行できます。デバッグ出力は、一部のフェンスエージェントで、フェンスデバイスにログインす

る際に、フェンスエージェントスクリプトに問題が発生しているイベントシーケンスの場所を確認するのに役に立ちます。

```
# fence_ilo -a ipaddress -l username -p password -o status -D
/tmp/$(hostname)-fence_agent.debug
```

発生した障害を診断する際に、フェンスデバイスに手動でログインする際に指定したオプションが、フェンスエージェントスクリプトでフェンスエージェントに渡した内容と同一であることを確認する必要があります。

フェンスエージェントが暗号化された接続をサポートする場合は、証明書の検証の失敗によりエラーが表示され、ホストを信頼するか、フェンスエージェントの **ssl-insecure** パラメーターを使用する必要があります。同様に、ターゲットデバイスで **SSL/TLS** を無効にした場合は、フェンスエージェントに **SSL** パラメーターを設定する際に、これを考慮しないといけない場合があります。

注記

テスト中のフェンスエージェントが **fence_drac**、**fence_ilo**、またはその他のシステム管理デバイスのフェンスエージェントで、引き続き失敗する場合は、フォールバックして **fence_ipmilan** を試行します。ほとんどのシステム管理カードは **IPMI** リモートログインに対応しており、フェンシングエージェントとしては **fence_ipmilan** だけに対応しています。

3.

フェンスデバイスを、手動で機能したオプションと同じオプションでクラスターに設定し、クラスターが起動したら、以下の例のように、任意のノードから **pcs stonith fence** コマンドを使用してフェンシングをテストします（または異なるノードから複数回実行します）。**pcs stonith fence** コマンドは、**CIB** からクラスター設定を読み取り、フェンスアクションを実行するように設定されたときにフェンスエージェントを呼び出します。これにより、クラスター設定が正確であることが確認できます。

```
# pcs stonith fence node_name
```

pcs stonith fence コマンドが正しく機能した場合、フェンスイベントの発生時にクラスターのフェンシング設定が機能します。このコマンドが失敗すると、クラスター管理が取得した設定でフェンスデバイスを起動することができません。以下の問題を確認し、必要に応じてクラスター設定を更新します。

- フェンス設定を確認します。たとえば、ホストマップを使用したことがある場合は、指定したホスト名を使用して、システムがノードを見つけられるようにする必要があります。

- デバイスのパスワードおよびユーザー名に、bash シェルが誤って解釈する可能性がある特殊文字が含まれるかどうかを確認します。パスワードとユーザー名を引用符で囲んで入力すると、この問題に対処できます。

- pcs stonith コマンドで IP アドレスまたはホスト名を使用してデバイスに接続できるかどうかを確認します。たとえば、stonith コマンドでホスト名を指定し、IP アドレスを使用して行ったテストは有効ではありません。

- お使いのフェンスデバイスが使用するプロトコルにアクセスできる場合は、そのプロトコルを使用してデバイスへの接続を試行します。たとえば、多くのエージェントが ssh または telnet を使用します。デバイスへの接続は、デバイスの設定時に指定した認証情報を使用して試行する必要があります。これにより、有効なプロンプトを取得し、そのデバイスにログインできるかどうかを確認できます。

すべてのパラメーターが適切であることが確認できたものの、フェンスデバイスには接続できない時に、フェンスデバイスでログ機能が使用できる場合は、ログを確認できます。これにより、ユーザーが接続したかどうかと、ユーザーが実行したコマンドが表示されます。また、/var/log/messages ファイルで stonith やエラーを検索することもできます。これにより、転送しているものの、エージェントによっては追加情報が提供される場合があります。

4. フェンスデバイステストに成功し、クラスターが稼働したら、実際の障害をテストします。このテストでは、クラスターで、トークンの損失を生じさせる動作を実行します。

- ネットワークを停止します。ネットワークの利用方法は、設定により異なります。ただし、多くの場合は、ネットワークケーブルまたは電源ケーブルをホストから物理的に抜くことができます。

注記

ネットワークや電源ケーブルを物理的に切断せずに、ローカルホストのネットワークインターフェイスを無効にすることは、フェンシングのテストとしては推奨されません。実際に発生する障害を正確にシミュレートしていないためです。

- ローカルのファイアウォールを使用して、corosync の受信トラフィックおよび送信トラフィックをブロックします。

以下の例では corosync をブロックします。ここでは、デフォルトの corosync ポートが使用され、firewalld がローカルファイアウォールとして使用し、corosync が使用す

るネットワークインターフェイスがデフォルトのファイアウォールゾーンにあることを前提とします。

```
# firewall-cmd --direct --add-rule ipv4 filter OUTPUT 2 -p udp --dport=5405 -j DROP
# firewall-cmd --add-rich-rule='rule family="ipv4" port port="5405" protocol="udp" drop'
```

- クラッシュをシミュレートし、マシンを **sysrq-trigger** でパニックにします。ただし、カーネルパニックを発生させると、データが損失する可能性があることに注意してください。クラッシュする前に、クラスターリソースを無効にすることが推奨されます。

```
# echo c > /proc/sysrq-trigger
```

第6章 クラスターリソースの設定

本章ではクラスター内にリソースを設定する方法について説明していきます。

6.1. リソースの作成

次のコマンドを使用してクラスターリソースを作成します。

```
pcs resource create resource_id [standard:[provider:]]type [resource_options] [op operation_action operation_options [operation_action operation options]...] [meta meta_options...] [clone [clone_options] | master [master_options] | --group group_name [--before resource_id | --after resource_id] | [bundle bundle_id] [--disabled] [--wait[=n]]
```

--group オプションを指定すると、名前付きのリソースグループにリソースが追加されます。グループが存在しない場合は作成され、そのグループにリソースが追加されます。リソースグループの詳細は「リソースグループ」 を参照してください。

--before および **--after** オプションは、リソースグループに含まれるリソースを基準にして、追加するリソースの位置を指定します。

--disabled オプションは、リソースが自動的に起動しないことを示しています。

次のコマンドは、標準 **ocf**、プロバイダー **heartbeat**、およびタイプ **IPaddr2** という名前の **VirtualIP** という名前のリソースを作成します。このリソースのフローティングアドレスは **192.168.0.120** で、システムはリソースが **30** 秒毎に実行されるかどうかをチェックします。

```
# pcs resource create VirtualIP ocf:heartbeat:IPaddr2 ip=192.168.0.120 cidr_netmask=24 op monitor interval=30s
```

standard と ***provider*** のフィールドを省略して次のようにすることもできます。デフォルトは **ocf** の標準および **ハートビート** のプロバイダーに設定されます。

```
# pcs resource create VirtualIP IPaddr2 ip=192.168.0.120 cidr_netmask=24 op monitor interval=30s
```

設定したリソースを削除する場合は、次のコマンドを実行します。

```
pcs resource delete resource_id
```

たとえば、次のコマンドは、リソース ID が VirtualIPの既存のリソースを削除します。

```
# pcs resource delete VirtualIP
```

- pcs resource create コマンドのフィールド *resource_id*、*standard*、*provider*、および *type* フィールドの詳細は、「リソースのプロパティー」 を参照してください。
- リソースごとにパラメーターを指定する方法は 「リソース固有のパラメーター」 を参照してください。
- リソースの動作をクラスターが決定する場合に使用するリソースのメタオプションを定義する方法は 「リソースのメタオプション」 を参照してください。
- リソースで行う動作を定義する方法は 「リソースの動作」 を参照してください。
- clone オプションを指定すると、クローンリソースが作成されます。master オプションを指定すると、クローンリソースが作成されます。リソースのクローンや、複数モードのリソースに関する詳細は *9章高度な設定* を参照してください。

6.2. リソースのプロパティー

リソースに定義するプロパティーを使ってリソースに使用するスクリプト、スクリプトの格納先、準拠すべき標準をクラスターに指示します。表6.1「リソースのプロパティー」 では、このプロパティーを説明します。

表6.1 リソースのプロパティー

フィールド	説明
resource_id	リソースの名前
standard	スクリプトが準拠する規格。使用できる値は、**ocf**、**service**、**upstart**、**systemd**、**lsb**、**stonith**です。
type	使用するリソースエージェントの名前(**IPaddr** や **Filesystem**など)
provider	OCF 仕様により、複数のベンダーで同じリソースエージェントを指定できます。Red Hat が提供するほとんどのエージェントは、プロバイダーとして **heartbeat** を使用します。

表6.2「リソースプロパティーを表示させるコマンド」 は、利用可能なリソースプロパティーを表示するコマンドの概要を示しています。

表6.2 リソースプロパティーを表示させるコマンド

pcs 表示コマンド	出力
pcs resource list	利用できる全リソースのリストを表示
pcs resource standards	利用できるリソースエージェントの標準を表示
pcs resource providers	利用できるリソースエージェントのプロバイダーを表示
pcs resource list *string*	利用できるリソースを指定文字列でフィルターしたリストを表示。仕様名、プロバイダー名、タイプ名などでフィルターを指定して、リソースを表示できます。

6.3. リソース固有のパラメーター

リソースごとに以下のコマンドを使用すると、そのリソースに設定できるパラメーターが表示されます。

```
# pcs resource describe standard:provider:type|type
```

たとえば、以下のコマンドはタイプ LVM のリソースに設定できるパラメーターを表示します。

```
# pcs resource describe LVM
Resource options for: LVM
  volgrpname (required): The name of volume group.
  exclusive: If set, the volume group will be activated exclusively.
  partial_activation: If set, the volume group will be activated even
  only partial of the physical volumes available. It helps to set to
  true, when you are using mirroring logical volumes.
```

6.4. リソースのメタオプション

リソースには、リソース固有のパラメーターの他に、リソースオプションを設定できます。このような追加オプションは、クラスターがリソースの動作を決定する際に使用されます。表6.3「リソースのメタオプション」 では、このようなオプションを説明します。

表6.3 リソースのメタオプション

フィールド	デフォルト	説明
priority	**0**	すべてのリソースをアクティブにできない場合に、クラスターは優先度の低いリソースを停止して、優先度が高いリソースを実行し続けます。
target-role	**Started**	クラスターが維持するリソースのステータスです。使用できる値は以下のようになります。 * Stopped - リソースの強制停止 * Started - リソースの起動を許可 (多状態リソースの場合マスターには昇格されません) * Master - リソースの起動を許可し、必要に応じて昇格
is-managed	**true**	クラスターによるリソースの起動と停止が可能かどうか。使用できる値は **true**、**false**です。
resource-stickiness	0	リソースを同じ場所に残すための優先度の値です。
requires	Calculated	リソースを起動できる条件を示します。 以下の条件を除き、デフォルトでは **フェンシング** に設定されます。以下の値が使用できます。 * **nothing** - クラスターは常にリソースを起動できます。 * **quorum** - クラスターは、設定されているノードの過半数がアクティブな場合にのみこのリソースを起動できます。**stonith-enabled** が **false** の場合、またはリソースの **standard** が **stonith** の場合は、この値になります。 * **fencing** -設定されているノードの過半数がアクティブで 障害が発生しているノードや不明なノードの電源がすべてオフになっている場合にのみ、クラスターはこのリソースを起動できます。 * ***unfencing*** *- 設定されているノードの過半数がアクティブで 障害が発生しているノードや不明なノードの電源がすべてオフになっている場合にのみ、* **アン** *フェンシングされたノードでのみ、このリソースを起動できます。フェンスデバイスに* ***provides=unfencing stonith*** *メタオプションが設定されている場合、これがデフォルト値です。*

フィールド	デフォルト	説明
migration-threshold	**INFINITY**	指定したリソースが任意のノードで失敗した回数です。この回数を超えると、そのノードには、このリソースのホストとして不適格とするマークが付けられます。値 0 は、この機能が無効になっていることを示します（ノードは不適格としてマークされません）。一方、クラスターは **INFINITY**（デフォルト）が非常に大きいものの、有限数として扱います。このオプションは、失敗した操作に **on-fail=restart**（デフォルト）があり、クラスタープロパティー **start-failure-is-fatal** が **false** の場合にのみ有効になります。**migration-threshold** オプションの設定の詳細は、「障害発生によるリソースの移動」を参照してください。**start-failure-is-fatal** オプションの詳細は、表12.1「クラスターのプロパティー」を参照してください。
failure-timeout	**0**（無効）	**migration-threshold** オプションと併用すると、障害が発生しなかったかのように動作し、障害が発生したノードにリソースを戻せるまで待機する秒数を示します。時間ベースのアクションと同様に、**cluster-recheck-interval** クラスターパラメーターの値よりも頻繁にチェックされる保証はありません。**failure-timeout** オプションの詳細は、「障害発生によるリソースの移動」を参照してください。
multiple-active	**stop_start**	リソースが複数のノードでアクティブであることが検出された場合に、クラスターが実行する動作です。使用できる値は以下のようになります。 * **block** - リソースを unmanaged としてマークします。 * **stop_only** - アクティブなインスタンスをすべて停止してそのままにします。 * **stop_start** - すべてのアクティブなインスタンスを停止し、1つの場所のみでリソースを起動します。

リソースオプションのデフォルト値を変更する場合は、次のコマンドを使用します。

```
pcs resource defaults options
```

たとえば、次のコマンドは、resource-stickiness のデフォルト値を 100 にリセットします。

```
# pcs resource defaults resource-stickiness=100
```

pcs resource defaults で *options* パラメーターを省略すると、現在設定されているリソースオプションのデフォルト値のリストが表示されます。以下の例は、resource-stickiness のデフォルト値を 100 にリセットした後のこのコマンドの出力を示しています。

```
# pcs resource defaults
resource-stickiness:100
```

リソースのメタオプションにおけるデフォルト値のリセットの有無に関わらず、リソースを作成する際に、特定リソースのリソースオプションをデフォルト以外の値に設定できます。以下は、リソースのメタオプションの値を指定するときに使用する **pcs resource create** コマンドの形式です。

```
pcs resource create resource_id standard:provider:type|type [resource options] [meta
meta_options...]
```

たとえば、以下のコマンドは **resource-stickiness** の値を **50** にしてリソースを作成します。

```
# pcs resource create VirtualIP ocf:heartbeat:IPaddr2 ip=192.168.0.120 cidr_netmask=24 meta
resource-stickiness=50
```

また、次のコマンドを使用すると既存のリソース、グループ、クローン作成したリソース、マスターリソースなどのリソースメタオプションの値を作成することもできます。

```
pcs resource meta resource_id | group_id | clone_id | master_id  meta_options
```

以下の例では、**dummy_resource** という名前の既存のリソースがあります。このコマンドは、**failure-timeout** メタオプションを **20** 秒に設定し、同じノードで **20** 秒で再起動を試行できるようにします。

```
# pcs resource meta dummy_resource failure-timeout=20s
```

このコマンドを実行すると、**failure-timeout=20s** が設定されているか確認するためにリソースの値を表示できます。

```
# pcs resource show dummy_resource
 Resource: dummy_resource (class=ocf provider=heartbeat type=Dummy)
  Meta Attrs: failure-timeout=20s
  Operations: start interval=0s timeout=20 (dummy_resource-start-timeout-20)
              stop interval=0s timeout=20 (dummy_resource-stop-timeout-20)
              monitor interval=10 timeout=20 (dummy_resource-monitor-interval-10)
```

リソースの **clone** メタオプションについては 「リソースのクローン」 を参照してください。リソースの **master** メタオプションについては 「多状態のリソース: 複数モードのリソース」 を参照してください。

6.5. リソースグループ

クラスターの最も一般的な設定要素の 1 つがリソースセットです。リソースセットはまとめて配置し、順番に起動し、その逆順で停止する必要があります。この設定を簡単にするため、 Pacemaker はグループの概念をサポートします。

以下のコマンドを使用してリソースグループを作成し、グループに追加するリソースを指定します。グループが存在しない場合は、このコマンドによりグループが作成されます。グループが存在する場合は、このコマンドにより別のリソースがグループに追加されます。リソースは、このコマンドで指定された順序で起動し、その逆順で停止します。

```
pcs resource group add group_name resource_id [resource_id] ... [resource_id]
[--before resource_id | --after resource_id]
```

このコマンドの **--before** オプションおよび **--after** オプションを使用して、追加するリソースの位置を、そのグループにすでに含まれるリソースを基準にして指定できます。

以下のコマンドを使用して、リソースを作成するときに、既存のグループに新しいリソースを追加することもできます。作成するリソースは ***group_name*** というグループに追加されます。

```
pcs resource create resource_id standard:provider:type|type [resource_options] [op
operation_action operation_options] --group group_name
```

以下のコマンドを使用して、グループからリソースを削除します。グループにリソースがない場合、このコマンドはグループ自体を削除します。

```
pcs resource group remove group_name resource_id...
```

以下のコマンドは、現在設定されているリソースグループをリスト表示します。

```
pcs resource group list
```

以下の例では、既存のリソースの **IPaddr** と **Email** が含まれるリソースグループ **shortcut** を作成します。

```
# pcs resource group add shortcut IPaddr Email
```

グループに含まれるリソースの数に制限はありません。グループの基本的なプロパティーは以下のとおりです。

- リソースは、指定した順序で起動します（この例では、最初に IPaddr、次に Email）。

- リソースは、指定した順序と逆の順序で停止します。(電子メール を最初に、次に IPaddr)。

グループ内に実行できないリソースがあると、そのリソースの後に指定されたリソースは実行できません。

- IPaddr を実行できない場合は、電子メール はできません。

- Email を実行できなくても、IPaddr には影響を及ぼしません。

グループが大きくなると、リソースグループ作成の設定作業を軽減することが重要になります。

6.5.1. グループオプション

リソースグループは、含まれるリソースから priority、target-role、is-managed オプションを継承します。リソースオプションの詳細は、表6.3「リソースのメタオプション」 を参照してください。

6.5.2. グループの Stickiness (粘着性)

粘着性は、リソースを現在の場所に留ませる優先度の度合いを示し、グループで加算されます。グループのアクティブなリソースが持つ stickness 値の合計が、グループの合計になります。したがって、デフォルトの resource-stickiness が 100 で、グループに 7 つのメンバーがあり、そのうちの 5 つがアクティブな場合、グループ全体でスコアが 500 の現在の場所が優先されます。

6.6. リソースの動作

リソースの健全性を維持するために、リソースの定義に監視操作を追加することができます。リソースの監視操作を指定しないと、pcs コマンドによりデフォルトで監視動作が作成されます。この間隔は、リソースエージェントにより決定します。リソースエージェントでデフォルトの監視間隔が提供されない場合は、pcs コマンドにより 60 秒間隔の監視動作が作成されます。

表6.4「動作のプロパティー」 に、リソースの監視動作のプロパティーを示します。

表6.4 動作のプロパティー

フィールド	説明
id	動作の一意の名前。システムは、操作を設定する際に、これを割り当てます。
name	実行する動作。一般的な値： **monitor**、**start**、**stop**
interval	値をゼロ以外に設定すると、この周波数で繰り返される反復操作 (秒単位) が作成されます。ゼロ以外の値は、アクション **名** が **monitor** に設定されている場合にのみ有効です。監視の反復アクションは、リソースの起動が完了するとすぐに実行し、その後の監視アクションの開始は、前の監視アクションが完了した時点でスケジュールされます。たとえば、**interval=20s** を持つ monitor アクションが 01:00:00 で実行されている場合、01:00:20 では次の監視アクションは発生しませんが、最初の監視アクションの完了後に 20 秒後に実行されます。 この値を、デフォルト値であるゼロに設定すると、このパラメーターで、クラスターが作成した操作に使用する値を指定できます。たとえば、**interval** をゼロに設定し、操作の **名前** が **start** に設定され、**タイムアウト** 値が 40 に設定されている場合、Pacemaker はこのリソースの起動時に 40 秒のタイムアウトを使用します。**monitor** 操作の間隔がゼロの場合、起動時に Pacemaker が行うプローブの **タイムアウト**/**on-fail**/**enabled** 値を設定し、デフォルトが望ましくない場合に、すべてのリソースの現在のステータスを取得できます。
timeout	このパラメーターで設定された時間内に操作が完了しないと、操作を中止し、失敗したと見なします。デフォルト値は、**pcs resource op defaults** コマンドで設定した場合は **timeout** の値です。設定されていない場合は 20 秒になります。システムが操作(**start**、**stop**、または **monitor**など)の実行を許可する時間よりも長い時間を必要とするリソースがシステムに含まれている場合は、原因を調査し、実行時間が長い場合はこの値を増やすことができます。 **timeout** 値はいずれの遅延でもなく、タイムアウト期間が完了する前に操作が戻ると、クラスターはタイムアウト期間全体を待つことはありません。
on-fail	この動作が失敗した場合に実行する動作。設定できる値は以下のとおりです。 * **ignore** - リソースが失敗しなかったことを追加します。 * **block** - リソースに対してそれ以上の操作を実行しません。 * **stop** - リソースを停止し、別の場所で起動しないようにします 再 **起動** - リソースを停止し、再起動します（別のノードにある可能性があります）。 * **fence** - リソースが失敗したノードである STONITH * **standby** - リソースが失敗したノードから **すべて** のリソースを移動します。 **migrate**: 可能であれば、リソースを別のノードに移行します。これは、**migration-threshold** リソースのメタオプションを 1 に設定するのと同じです。 STONITH が有効な場合は、**stop** 操作のデフォルトは **fence** で、それ以外は **block** となります。その他のすべての操作は、デフォルトで **restart** です。
enabled	**false** の場合、操作は存在しないものとして処理されます。使用できる値は **true**、**false**です。

6.6.1. リソース操作の設定

次のコマンドでリソースを作成すると、監視操作を設定できます。

```
pcs resource create resource_id standard:provider:type|type [resource_options] [op operation_action operation_options [operation_type operation_options]...]
```

たとえば、次のコマンドは、監視操作で **IPaddr2** リソースを作成します。新しいリソースは、**eth2** で **IP** アドレス **192.168.0.99**、ネットマスクが **24** の **VirtualIP** と呼ばれます。監視操作は、**30** 秒ごとに実施されます。

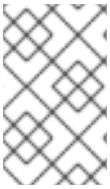
```
# pcs resource create VirtualIP ocf:heartbeat:IPaddr2 ip=192.168.0.99 cidr_netmask=24 nic=eth2 op monitor interval=30s
```

また、次のコマンドで既存のリソースに監視操作を追加することもできます。

```
pcs resource op add resource_id operation_action [operation_properties]
```

設定されているリソース操作を削除する場合は、次のコマンドを使用します。

```
pcs resource op remove resource_id operation_name operation_properties
```



注記

操作プロパティーを正しく指定して、既存の操作を適切に削除する必要があります。

監視オプションの値を変更する場合は、リソースを更新します。たとえば、以下のコマンドで **VirtualIP** を作成できます。

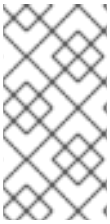
```
# pcs resource create VirtualIP ocf:heartbeat:IPaddr2 ip=192.168.0.99 cidr_netmask=24 nic=eth2
```

デフォルトでは、次の操作が作成されます。

```
Operations: start interval=0s timeout=20s (VirtualIP-start-timeout-20s)
            stop interval=0s timeout=20s (VirtualIP-stop-timeout-20s)
            monitor interval=10s timeout=20s (VirtualIP-monitor-interval-10s)
```

stop の timeout 操作を変更するには、以下のコマンドを実行します。

```
# pcs resource update VirtualIP op stop interval=0s timeout=40s

# pcs resource show VirtualIP
 Resource: VirtualIP (class=ocf provider=heartbeat type=IPaddr2)
  Attributes: ip=192.168.0.99 cidr_netmask=24 nic=eth2
  Operations: start interval=0s timeout=20s (VirtualIP-start-timeout-20s)
              monitor interval=10s timeout=20s (VirtualIP-monitor-interval-10s)
              stop interval=0s timeout=40s (VirtualIP-name-stop-interval-0s-timeout-40s)
```



注記

pcs resource update コマンドでリソースの操作を更新すると、特に呼び出しのないオプションはデフォルト値にリセットされます。

6.6.2. グローバルリソース操作のデフォルトの設定

次のコマンドを使用して、監視操作のグローバルデフォルト値を設定できます。

```
pcs resource op defaults [options]
```

たとえば、次のコマンドは、すべての監視操作に対して、timeout 値のグローバルデフォルトを 240 秒に設定します。

```
# pcs resource op defaults timeout=240s
```

監視操作に現在設定されているデフォルト値を表示するには、オプションを指定せずに pcs resource op defaults コマンドを実行します。

たとえば、以下のコマンドは、タイムアウト 値 240 秒で設定されたクラスターのデフォルトの監視操作値を表示します。

```
# pcs resource op defaults
timeout: 240s
```

クラスターリソース定義でオプションが指定されていない場合に限り、クラスターリソースがグローバルデフォルトを使用することに注意してください。デフォルトでは、リソースエージェントはすべての操作に timeout オプションを定義します。グローバル操作のタイムアウト値を有効にするに

は、**timeout** オプションを明示的に指定せずにクラスターリソースを作成するか、以下のコマンドのように、クラスターリソースを更新して **timeout** オプションを削除する必要があります。

```
# pcs resource update VirtualIP op monitor interval=10s
```

たとえば、すべての監視操作にグローバルなデフォルトの タイムアウト 値を 240 秒に設定し、クラスターリソース **VirtualIP** を更新して **monitor** 操作のタイムアウト値を削除すると、リソース **VirtualIP** には、**start**、**stop**、および **monitor** の操作のタイムアウト値がそれぞれ **20s**、**40s**、および **240s** になります。タイムアウト操作のグローバルデフォルト値は、ここでは **monitor** 操作にのみ適用されます。ここでは、前のコマンドでデフォルトの **timeout** オプションが削除されました。

```
# pcs resource show VirtualIP
 Resource: VirtualIP (class=ocf provider=heartbeat type=IPaddr2)
  Attributes: ip=192.168.0.99 cidr_netmask=24 nic=eth2
  Operations: start interval=0s timeout=20s (VirtualIP-start-timeout-20s)
              monitor interval=10s (VirtualIP-monitor-interval-10s)
              stop interval=0s timeout=40s (VirtualIP-name-stop-interval-0s-timeout-40s)
```

6.7. 設定されているリソースの表示

設定されているリソースのリストを表示する場合は、次のコマンドを使用します。

```
pcs resource show
```

たとえば、システムが **VirtualIP** という名前のリソースと **WebSite** という名前のリソースで設定されていると、**pcs resource show** コマンドを実行すると以下の出力が得られます。

```
# pcs resource show
 VirtualIP (ocf::heartbeat:IPaddr2): Started
 WebSite (ocf::heartbeat:apache): Started
```

リソースに設定されているパラメーターを表示する場合は、次のコマンドを使用します。

```
pcs resource show resource_id
```

たとえば、次のコマンドは、現在設定されているリソース **VirtualIP** 用にパラメーターを表示します。

```
# pcs resource show VirtualIP
 Resource: VirtualIP (type=IPaddr2 class=ocf provider=heartbeat)
  Attributes: ip=192.168.0.120 cidr_netmask=24
  Operations: monitor interval=30s
```

-

6.8. リソースパラメーターの変更

設定されているリソースのパラメーターを変更する場合は、次のコマンドを使用します。

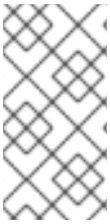
```
pcs resource update resource_id [resource_options]
```

以下のコマンドシーケンスでは、**VirtualIP** リソースに設定したパラメーターの初期値、**ip** パラメーターの値を変更するコマンド、**update** コマンドの後の値を示しています。

```
# pcs resource show VirtualIP
 Resource: VirtualIP (type=IPaddr2 class=ocf provider=heartbeat)
  Attributes: ip=192.168.0.120 cidr_netmask=24
  Operations: monitor interval=30s
# pcs resource update VirtualIP ip=192.169.0.120
# pcs resource show VirtualIP
 Resource: VirtualIP (type=IPaddr2 class=ocf provider=heartbeat)
  Attributes: ip=192.169.0.120 cidr_netmask=24
  Operations: monitor interval=30s
```

6.9. 複数のモニタリング動作

リソースエージェントが対応する範囲で、**1** つのリソースに複数の監視操作を設定できます。これにより、**1** 分ごとに表面的なヘルスチェックを行い、徐々に頻度を上げてより正確なチェックを行うこともできます。



注記

複数の監視操作を設定する場合は、**2** 種類の操作が同じ間隔で実行されないように注意してください。

異なるレベルでの詳細なチェックに対応するリソースの追加監視操作を設定するには、**OCF_CHECK_LEVEL=*n*** オプションを追加します。

たとえば、以下の **IPaddr2** リソースを設定すると、デフォルトで **10** 秒間隔でタイムアウト値が **20** 秒の監視操作が作成されます。

```
# pcs resource create VirtualIP ocf:heartbeat:IPaddr2 ip=192.168.0.99 cidr_netmask=24 nic=eth2
```

仮想 IP が、深さ 10 の様々なチェックに対応する場合は、次のコマンドを実行すると、Pacemaker は、通常の 10 秒間隔の仮想 IP チェックに加えて、60 秒ごとに高度な監視チェックを実行します。なお、上述のとおり、追加の監視操作は 10 秒間隔にしないようにしてください。

```
# pcs resource op add VirtualIP monitor interval=60s OCF_CHECK_LEVEL=10
```

6.10. クラスターリソースの有効化と無効化

以下のコマンドは、**resource_id** で指定されるリソースを有効にします。

```
pcs resource enable resource_id
```

次のコマンドは、**resource_id** で指定されるリソースを無効にします。

```
pcs resource disable resource_id
```

6.11. クラスターリソースのクリーンアップ

リソースに障害が発生すると、クラスターの状態を表示するときに障害メッセージが表示されます。このリソースを解決する場合は、**pcs resource cleanup** コマンドで障害状態を消去できます。このコマンドは、リソースのステータスと **failcount** をリセットし、リソースの操作履歴を取得し、現在の状態を再検出するようにクラスターに指示します。

以下のコマンドは、***resource_id*** によって指定されたリソースをクリーンアップします。

```
pcs resource cleanup resource_id
```

resource_id を指定しないと、このコマンドはすべてのリソースのリソースステータスと **failcount**をリセットします。

Red Hat Enterprise Linux 7.5 では、**pcs resource cleanup** コマンドは、失敗したアクションとして表示されるリソースのみを検証します。全ノードの全リソースを調査するには、次のコマンドを入力します。

```
pcs resource refresh
```

デフォルトでは、**pcs resource refresh** コマンドは、リソースの状態が分かっているノードだけを検証します。ステータスが分からないすべてのリソースを検証するには、以下のコマンドを実行します。

```
pcs resource refresh --full
```

第7章 リソースの制約

リソースの制約を設定して、クラスター内のそのリソースの動作を指定できます。以下の制約のカテゴリーを設定できます。

- 場所 の制約 - リソースを実行できるノードを指定する場所の制約です。場所の制約については 「場所の制約」 で説明しています。
- 順序 の制約 : 順序制約により、リソースが実行される順序が決まります。順序の制約については 「順序の制約」 で説明しています。
- コロケーション の制約 - 他のリソースに対して相対的にリソースの配置先を決定します。コロケーションの制約については 「リソースのコロケーション」 で説明しています。

複数リソースをまとめて配置して、順番に起動するまたは逆順で停止する一連の制約を簡単に設定する方法として、Pacemaker ではリソースグループという概念に対応しています。リソースグループの詳細は 「リソースグループ」 を参照してください。

7.1. 場所の制約

場所の制約は、リソースを実行するノードを指定します。場所の制約を設定することで、たとえば特定のノードで優先してリソースを実行する、または特定のノードではリソースを実行しないことを決定できます。

場所の制約 に加え、リソースが実行されるノードは、そのリソースの resource-stickiness 値の影響を受けます。これは、リソースが現在実行しているノードに留まることをどの程度優先するかを決定します。resource-stickiness 値の設定に関する詳細は、「現在のノードを優先させるリソースの設定」 を参照してください。

7.1.1. 基本的な場所の制約

基本的な場所の制約を設定して、リソースがノードを優先するか回避するかを指定できます。オプションの score 値を使用して、制約の相対的な優先度を指定できます。

以下のコマンドは、リソースの実行を、指定した 1 つまたは複数のノードで優先するように、場所の制約を作成します。1 回のコマンドで、特定のリソースの制約を複数のノードに対して作成できます。

pcs constraint location *rsc* **prefers** *node*[=*score*] [*node*[=*score*]] ...

次のコマンドは、リソースが指定ノードを回避する場所の制約を作成します。

pcs constraint location *rsc* **avoids** *node*[=*score*] [*node*[=*score*]] ...

表7.1「簡単な場所の制約オプション」 では、最も簡単な形式で場所の制約を設定する場合のオプションの意味をまとめています。

表7.1 簡単な場所の制約オプション

フィールド	説明
rsc	リソース名
node	ノード名

フィールド	説明
score	リソースを特定ノードで優先的に実行するか、または実行を回避するかを示す正の整数値。INFINITY は、リソースの場所制約のデフォルトの スコア 値です。 pcs contraint location *rsc* prefers コマンドで score の値の INFINITY を指定すると、そのノードが利用可能な場合は、リソースがそのノードで優先的に実行します。ただし、そのノードが利用できない場合に、別のノードでそのリソースを実行しないようにする訳ではありません。 pcs contraint location *rsc* avoids コマンドの score に INFINITY を指定すると、他のノードが利用できない場合でも、そのリソースはそのノードでは実行されないことを示します。これは、-INFINITY のスコアで pcs constraint location add コマンドを設定するのと同じです。

以下のコマンドは、Webserver リソースが node1 ノードを優先するように指定する場所の制約を作成します。

```
# pcs constraint location Webserver prefers node1
```

Red Hat Enterprise Linux 7.4 では、pcs は、コマンドラインの場所の制約における正規表現に対応しています。この制約は、リソース名に一致する正規表現に基づいて、複数のリソースに適用されます。これにより、1 つのコマンドラインで複数の場所の制約を設定できます。

次のコマンドは、dummy0 から dummy9 までのリソースが node1 を優先するように指定する場所の制約を作成します。

```
# pcs constraint location 'regexp%dummy[0-9]' prefers node1
```

Pacemaker は、

http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap09.html#tag_09_04 で説明しているように、POSIX 拡張正規表現を使用するため、以下のコマンドを実行しても同じ制約を指定できます。

```
# pcs constraint location 'regexp%dummy[[:digit:]]' prefers node1
```

7.1.2. 高度な場所の制約

ノードに場所の制約を設定する場合は、pcs constraint location コマンドの resource-discovery オプションを使用して、指定したリソースに対して Pacemaker がこのノードでリソース検出を実行するかどうかの優先順位を指定できます。物理的にリソースが稼働可能なノードのサブセットでリソース検出を制限すると、ノードが大量に存在する場合にパフォーマンスを大幅に改善できます。pacemaker_remote を使用してノード数を数百のノード数に拡張する場合は、このオプションを考慮する必要があります。

以下のコマンドは、pcs constraint location コマンドで resource-discovery オプションを指定するための形式を示しています。*id* は制約 id であることに注意してください。*rsc*、*node*、*score* の意味は 表7.1「簡単な場所の制約オプション」 で説明しています。このコマンドでは、基本的な場所の制約に対応します。*score* を正の値にすると、リソースが特定のノードで優先的に実行するように設定されます。*score* を負の値にすると、リソースがノードを回避するように設定されます。基本的な場所の制約と同様に、制約にリソースの正規表現を使用することもできます。

```
pcs constraint location add id rsc node score [resource-discovery=option]
```

表7.2「リソース検出の値」 では、resource-discovery オプションに指定できる値を説明しています。

表7.2 リソース検出の値

値	説明
always	このノードに指定したリソースで、リソース検出を常に実行します。これは、リソースの場所の制約のデフォルトの **resource-discovery** 値です。
never	このノードで指定ししたリソースでのリソース検出は実行しません。
exclusive	指定したリソース（および **exclusive**としてマークされている他のノード）でのみ、リソース検出を実行します。異なるノード間で同じリソースの **exclusive** 検出を使用する複数の場所制約により、**resource-discovery** が排他的なノードのサブセットが作成されます。リソースが 1 つ以上のノードの **exclusive** 検出用にマークされている場合、そのリソースは、そのノードのサブセット内にのみ配置できます。

resource-discovery オプションを never または exclusive に設定すると、クラスターが認識しなくても、該当する場所でリソースがアクティブになる可能性があることに注意してください。これにより、サービスがクラスター制御外(systemd または管理者など)以外で起動すると、複数の場所でリ

ソースがアクティブになる可能性があります。これは、クラスターの一部がダウンしたりスプリットブレインが発生しているときに resource-discovery プロパティーが変更された場合や、そのノードでリソースがアクティブになっている間に resource-discovery プロパティーがリソースおよびノードのリソースに対して変更された場合にも発生する可能性があります。そのため、ノードの数が 9 以上で、特定の場所しかリソースを実行できない場合 (必要なソフトウェアが他の場所にインストールされていない場合など) に限り、このオプションは適しています。

7.1.3. ルールを使用したリソースの場所の確定

より複雑な場所の制約は、Pacemaker ルールを使用してリソースの場所を判断することができます。Pacemaker ルールや設定できるパロパティの概要は、11章*Pacemaker ルール* を参照してください。

以下のコマンドを使用して、ルールを使用する Pacemaker 制約を使用します。score を省略すると、デフォルトで INFINITY に設定されます。resource-discovery を省略すると、デフォルトで always に設定されます。resource-discovery オプションの詳細は、「高度な場所の制約」 を参照してください。基本的な場所の制約と同様に、制約にリソースの正規表現を使用することもできます。

ルールを使用して場所の制約を設定する場合は、score の値を正または負の値にすることができます。正の値は "prefers" を示し、負の値は "avoids" を示します。

> **pcs constraint location** *rsc* **rule [resource-discovery=***option***] [role=master|slave] [score=***score* **| score-attribute=***attribute***]** *expression*

expression オプションは、*duration_options* および *date_spec_options* のいずれかに設定できます。使用できる値は、表11.5「日付詳細のプロパティー」 で説明されているように、hours、monthdays、weekdays、yeardays、months、weeks、years、weekyears、moon になります。

- defined|not_defined *attribute*
- *attribute* lt|gt|lte|gte|eq|ne [string|integer|version] *value*
- date gt|lt *date*
- 日付範囲内の *日付*
- date in-range *date* to duration *duration_options* ...

- date-spec *date_spec_options*
- *式* および|or *式*
- (*expression*)

以下の場所の制約は、現在が 2018 年の任意の時点である場合に true の式を設定します。

```
# pcs constraint location Webserver rule score=INFINITY date-spec years=2018
```

以下のコマンドは、月曜日から金曜日までの 9 am から 5 pm までが true となる式を設定します。hours の値 16 には、時間 (hour) の値が一致する 16:59:59 までが含まれます。

```
# pcs constraint location Webserver rule score=INFINITY date-spec hours="9-16" weekdays="1-5"
```

以下のコマンドは、13 日の金曜日が満月になると true になる式を設定します。

```
# pcs constraint location Webserver rule date-spec weekdays=5 monthdays=13 moon=4
```

7.1.4. 場所の制約ストラテジー

「基本的な場所の制約」、「高度な場所の制約」、「ルールを使用したリソースの場所の確定」 で説明している場所の制約のいずれかを使用することで、リソースを実行できるノードを指定するための一般的なストラテジーを設定できます。

- オプトインクラスター - デフォルトでは、すべてのリソースを、どのノードでも実行できません。そして、特定のリソースに対してノードを選択的に許可できるようにクラスターを設定します。オプトインクラスターの設定方法は 「オプトインクラスターの設定」 で説明しています。
- オプトアウトクラスター - デフォルトでは、すべてのリソースをどのノードでも実行できるクラスターを設定してから、リソースを特定のノードで実行しないように、場所の制約を作成します。オプトアウトクラスターの設定方法は 「オプトアウトクラスターの設定」 で説明しています。これは、デフォルトの Pacemaker ストラテジーです。

クラスターでオプトインまたはオプトアウトのどちらを選択するかは、独自に優先する設定やクラ

スターの設定により異なります。ほとんどのリソースをほとんどのノードで実行できるようにする場合は、オプトアウトを使用した方が設定しやすくなる可能性があります。ほとんどのリソースを、一部のノードでのみ実行する場合は、オプトインを使用した方が設定しやすくなる可能性があります。

7.1.4.1. オプトインクラスターの設定

オプトインクラスターを作成するには、クラスタープロパティー **symmetric-cluster** を **false** に設定し、デフォルトでは、いずれの場所でもリソースが実行されないようにします。

```
# pcs property set symmetric-cluster=false
```

個々のリソースでノードを有効にします。以下のコマンドは、場所の制約を設定し、**Webserver** リソースでは **example-1** ノードが優先され、**Database** リソースでは **example-2** ノードが優先され、優先ノードに障害が発生した場合は両方のリソースが **example-3** ノードにフェイルオーバーできるようにします。オプトインクラスターに場所の制約を設定する場合は、スコアをゼロに設定すると、リソースに対してノードの優先や回避を指定せずに、リソースをノードで実行できます。

```
# pcs constraint location Webserver prefers example-1=200
# pcs constraint location Webserver prefers example-3=0
# pcs constraint location Database prefers example-2=200
# pcs constraint location Database prefers example-3=0
```

7.1.4.2. オプトアウトクラスターの設定

オプトアウトクラスターを作成するには、**symmetric-cluster** クラスタープロパティーを **true** に設定して、デフォルトですべてのリソースがどこでも実行できるようにします。

```
# pcs property set symmetric-cluster=true
```

以下のコマンドを実行すると、「オプトインクラスターの設定」 の例と同じ設定になります。すべてのノードのスコアが暗黙的スコアが **0** であるため、優先ノードに障害が発生した場合は、両方のリソースを **example-3** ノードにフェイルオーバーできます。

```
# pcs constraint location Webserver prefers example-1=200
# pcs constraint location Webserver avoids example-2=INFINITY
# pcs constraint location Database avoids example-1=INFINITY
# pcs constraint location Database prefers example-2=200
```

INFINITY は、スコアのデフォルト値であるため、上記コマンドでは、スコアに **INFINITY** を指定する必要はないことに注意してください。

7.1.5. 現在のノードを優先させるリソースの設定

リソースには、「リソースのメタオプション」 で説明されているように、リソースの作成時にメタ属性として設定できる resource-stickiness 値があります。resource-stickiness 値は、現在実行しているノード上にリソースが残す量を決定します。Pacemaker は、他の設定（場所の制約の score 値など）とともに resource-stickiness 値を考慮して、リソースを別のノードに移動するか、そのまま残すかを決定します。

デフォルトでは、resource-stickiness の値が 0 の状態でリソースが作成されます。resource-stickiness が 0 に設定され、場所の制約がない Pacemaker のデフォルト動作では、クラスターノード間で均等に分散されるようにリソースを移動します。この設定では、正常なリソースの移動頻度が想定よりも増える可能性があります。この動作を防ぐには、デフォルトの resource-stickiness 値を 1 に設定します。このデフォルトはクラスター内のすべてのリソースに適用されます。この小さい値は、作成する他の制約で簡単に上書きできますが、Pacemaker がクラスター全体で正常なリソースを不必要に移動しないようにするには十分です。

以下のコマンドは、デフォルトの resource-stickiness 値を 1 に設定します。

```
# pcs resource defaults resource-stickiness=1
```

resource-stickiness 値が設定されている場合、リソースは新たに追加されたノードに移動されません。この時点でリソースバランシングが必要な場合は、resource-stickiness の値を一時的に 0 に設定できます。

場所の制約スコアが resource-stickiness 値よりも高い場合、クラスターは正常なリソースを、場所の制約がポイントするノードに依然として移動する可能性があります。

Pacemaker がリソースの配置先を決定する方法の詳細は、「使用と配置ストラテジー」 を参照してください。

7.2. 順序の制約

順序の制約でリソースを実行する順序を指定します。

次のコマンドを使って順序の制約を設定します。

pcs constraint order [*action*] *resource_id* **then** [*action*] *resource_id* [*options*]

表7.3「順序の制約のプロパティー」 では順序の制約を設定する場合のプロパティーとオプションについて簡単に示します。

表7.3 順序の制約のプロパティー

フィールド	説明
resource_id	動作を行うリソースの名前。
action	リソースで実行する動作。*action* プロパティーでは、以下の値が使用できます。 * start - リソースを開始します。 * stop - リソースを停止します。 * promote - リソースをスレーブリソースからマスターリソースにプロモートします。 * demote - マスターリソースからスレーブリソースにリソースをデモートします。 アクションを指定しないと、デフォルトのアクションは start になります。マスターリソースとスレーブリソースについての詳細は 「多状態のリソース: 複数モードのリソース」 を参照してください。

フィールド	説明
kind オプション	制約の実施方法。kind オプションでは、以下の値を使用できます。 * オプション - 両方のリソースが指定されたアクションを実行する場合にのみ適用されます。オプションの順序は 「勧告的な順序付け」 を参照してください。 * 必須 - 常に（デフォルト値）1 番目に指定したリソースが停止している場合や起動できない場合は、2 番目に指定したリソースが停止します。必須の順序の詳細は、 「強制的な順序付け」 を参照してください。 * シリアル化 - 一連のリソースに対して、2 つの停止/開始アクションが同時に実行されないようにします。
対称 オプション	デフォルトの true に設定されていると逆順でリソースの停止を行ないます。デフォルト値： true

7.2.1. 強制的な順序付け

mandatory 制約では 1 番目に指定しているリソースが実行されない限り 2 番目に指定しているリソースは実行できません。これは、kind オプションのデフォルト値です。デフォルト値のままにして

おくと 1 番目に指定しているリソースの状態が変化した場合、2 番目に指定したリソー スが必ず反応するようになります。

- 1 番目に指定している実行中のリソースを停止すると、2 番目に指定しているリソースも停止されます (実行している場合)。
- 1 番目に指定している実行中のリソースが実行しておらず起動できない場合には、指定したリソースは (実行していれば) 停止します。
- 2 番目に指定しているリソースの実行中に 1 番目に指定しているリソースが再起動されると、2 番目に指定しているリソースが停止され再起動されます。

ただし、クラスターは、それぞれの状態変化に対応することに注意してください。2 番目のリソースで停止操作を開始する前に 1 番目のリソースが再起動し、起動状態にあると、2 番目のリソースを再起動する必要がありません。

7.2.2. 勧告的な順序付け

順序の制約に kind=Optional オプションを指定すると、制約は任意と見なされ、両方のリソースが指定のアクションを実行する場合のみ適用されます。1 番目に指定しているリソースの状態を変更しても、2 番目に指定しているリソースには影響しません。

以下のコマンドは、VirtualIP および dummy_resource という名前のリソースのアドバイザリーの順序制約を設定します。

```
# pcs constraint order VirtualIP then dummy_resource kind=Optional
```

7.2.3. 順序付けされたリソースセット

一般的に、管理者は、複数のリソースの連鎖を作成する際に順序を設定します (例: リソース A が開始してからリソース B を開始し、その後にリソース C を開始)。複数のリソースを作成して同じ場所に配置し (コロケーションを指定)、起動の順序を設定する必要がある場合は、「リソースグループ」 に従って、このようなリソースが含まれるリソースグループを設定できます。ただし、特定の順序で起動する必要があるリソースをリソースグループとして設定することが適切ではない場合があります。

- リソースを順番に起動するように設定する必要があるものの、リソースは必ずしも同じ場所に配置しない場合
-

リソース C の前にリソース A または B のいずれかが起動する必要があるものの、A と B の間には関係が設定されていない場合

- リソース C およびリソース D の前にリソース A およびリソース B の両方が起動している必要があるものの、A と B、または C と D の間には関係が設定されていない場合

このような状況では、pcs constraint order set コマンドを使用して、リソースの 1 つまたは複数のセットに対して順序の制約を作成できます。

pcs constraint order set コマンドを使用すると、以下のオプションをリソースのセットに設定できます。

- sequential。true または false に設定して、リソースセットを相互に相対的に並べる必要があるかどうかを指定します。

  sequential を false に設定すると、セットのメンバーが互いに相対的に順序付けされることなく、順序の制約内の他のセットとの関連で順序付けることができます。そのため、このオプションは、制約に複数のセットが登録されている場合に限り有効です。それ以外の場合は、制約を設定しても効果がありません。

- require-all: 続行する前にセット内のすべてのリソースがアクティブである必要があるかどうかを指定します。true または false に設定できます。require-all を false に設定すると、次のセットに進む前に、セット内の 1 つのリソースのみを起動する必要があります。require-all を false に設定しても、シーケンシャル が false に設定されている順序なしのセットと併用されない限り、効果はありません。

- action: 表7.3「順序の制約のプロパティー」 の説明通りに、start、promote、demote または stop に設定できます。

pcs constraint order set コマンドの setoptions パラメーターの後に、リソースのセットに対する以下の制約オプションを設定できます。

- id: 定義する制約の名前を指定します。

- score: 制約の優先度を示します。このオプションの詳細は 表7.4「コロケーション制約のプロパティー」 を参照してください。

pcs constraint order set *resource1 resource2* [*resourceN*]... [*options*] [**set** *resourceX resourceY* ... [*options*]] [**setoptions** [*constraint_options*]]

D1、D2、および D3 という名前の 3 つのリソースがある場合、次のコマンドはそれらを順序付けされたリソースセットとして設定します。

```
# pcs constraint order set D1 D2 D3
```

7.2.4. 順序の制約からリソースを削除

次のコマンドを使用すると、すべての順序の制約からリソースが削除されます。

pcs constraint order remove *resource1* [*resourceN*]...

7.3. リソースのコロケーション

任意のリソースの場所を別のリソースの場所に依存するよう定義するのがコロケーションの制約です。

2 つのリソース間にコロケーション制約を作成すると、リソースがノードに割り当てられる割り当てる順序に重要な影響を及ぼす点に注意してください。リソース B の場所を把握していない場合は、リソース B に相対的となるようにリソース A を配置することができません。このため、コロケーションの制約を作成する場合は、リソース A をリソース B に対してコロケーションを設定するのか、もしくはリソース B をリソース A に対してコロケーションを設定するのかを考慮する必要があります。

また、コロケーションの制約を作成する際に注意しておきたいもう 1 つの点として、リソース A をリソース B に対してコロケーションすると仮定した場合、クラスターがリソース B に選択するノードを決定する際、リソース A の優先度も考慮に入れます。

次のコマンドはコロケーションの制約を作成します。

pcs constraint colocation add [master|slave] *source_resource* **with [master|slave]** *target_resource* [*score*] [*options*]

マスターリソースとスレーブリソースについての詳細は 「多状態のリソース: 複数モードのリソース」 を参照してください。

表7.4「コロケーション制約のプロパティー」 は、コロケーション制約を設定するのに使用するプロ

パティーおよびオプションをまとめています。

表7.4 コロケーション制約のプロパティー

フィールド	説明
source_resource	コロケーションソース。制約の条件を満たさない場合、クラスターではこのリソースの実行を許可しないことを決定する可能性があります。
target_resource	コロケーションターゲット。クラスターは、このリソースの配置先を決定してから、ソースリソースの配置先を決定します。
score	正の値を指定するとリソースが同じノードで実行します。負の値を指定すると同じノードで実行しなくなります。デフォルト値である +INFINITY を指定すると、*source_resource* は *target_resource* と同じノードで実行する必要があることを示します。-INFINITY の値は、*source_resource* を *target_resource* と同じノードで実行してはならないことを示します。

7.3.1. 強制的な配置

必須配置は、制約のスコアが +INFINITY または -INFINITY であるたびに発生します。制約条件が満たされないと *source_resource* の実行が許可されません。score=INFINITY の場合、これには、*target_resource* がアクティブではないケースが含まれます。

myresource1 を、常に myresource2 と同じマシンで実行する必要がある場合は、次の制約を追加します。

```
# pcs constraint colocation add myresource1 with myresource2 score=INFINITY
```

INFINITY が使用されるため、（何らかの理由で） myresource2 がどのクラスターノードでも実行できない場合、myresource1 の実行は許可されません。

または、逆の設定、つまり myresource1 が myresource2 と同じマシンで実行できないクラスターを設定することもできます。この場合は、score=-INFINITYを使用します。

```
# pcs constraint colocation add myresource1 with myresource2 score=-INFINITY
```

ここでも、-INFINITY を指定することで、制約はバインドされます。したがって、実行したい場所が残っている場所が myresource2 である場合、myresource1 はどこにも実行できません。

7.3.2. 勧告的な配置

必ず実行する (または必ず実行しない) 状況を強制的な配置とするならば、勧告的な配置は、ある状況下で優先される状況を指しています。制約のスコアが -INFINITY より大きく、INFINITY より小さい場合、クラスターはユーザーの希望に対応しようとしますが、クラスターリソースの一部を停止するという選択肢がある場合には無視できます。勧告的なコロケーション制約と設定の他の要素を組み合わせると、強制的であるように動作させることができます。

7.3.3. 複数リソースのコロケート

コロケーションと順序が適用されるリソースのセットを作成する必要がある場合は、「リソースグループ」 に従って、このようなリソースを含むリソースグループを設定できます。ただし、コロケーションを設定する必要があるリソースをリソースグループとして設定することが適切ではない場合もあります。

- リソースのセットにコロケーションを設定する必要があるものの、リソースが必ずしも順番に起動する必要がない場合
- リソース A または B のいずれかに対してコロケーションを設定する必要があるリソース C が起動したものの、A と B の間に関係が設定されていない場合。
- リソース C およびリソース D を、リソース A およびリソース B の両方に対してコロケーションを設定する必要があるものの、A と B の間、または C と D の間に関係が設定されていない場合

このような状況では、pcs constraint colocation set コマンドを使用して、リソースのセットにコロケーション制約を作成できます。

pcs constraint colocation set コマンドを使用すると、以下のオプションをリソースのセットに設

定できます。

- sequential。セットのメンバーが相互にコロケーションする必要があるかどうかを示す true または false に設定できます。

  sequential を false に設定すると、このセットのどのメンバーがアクティブであるかに関係なく、このセットのメンバーを制約の後半に一覧表示されている別のセットと同じ場所に配置できます。そのため、このオプションは制約でこのセットの後に他のセットが指定されている場合に限り有効です。他のセットが指定されていない場合は、制約の効果がありません。

- Role: Stopped、Started、Master、または Slave に設定できます。マルチステートリソースの詳細は 「多状態のリソース: 複数モードのリソース」 を参照してください。

pcs constraint colocation set コマンドの setoptions パラメーターの後に、リソースのセットに対する以下の制約オプションを設定できます。

- kind: 制約の実行方法を示します。このオプションの詳細は 表7.3「順序の制約のプロパティー」 を参照してください。

- symmetrical: リソースを停止する順序を示します。デフォルトの true に設定されていると逆順でリソースの停止を行ないます。デフォルト値： true

- id: 定義する制約の名前を指定します。

セットのメンバーをリストすると、各メンバーは、自身の前のメンバーに対してコロケーションが設定されます。たとえば、set A B は B が A の場所に配置されることを意味します。しかし、複数のセットをリストする場合、各セットはその後のメンバーと同じ場所に配置されます。たとえば、set C D sequential=false set A B は、C と D のセットが、A と B のセットと同じ場所に配置されることを意味します (ただし、C と D には関係がなく、B は A にはコロケーションが設定されています)。

以下のコマンドは、リソースのセットにコロケーションの制約を作成します。

> **pcs constraint colocation set** *resource1 resource2* [*resourceN*]... [*options*] [**set** *resourceX resourceY* ... [*options*]] [**setoptions** [*constraint_options*]]

7.3.4. コロケーション制約の削除

コロケーション制約を削除する場合はコマンドに ***source_resource*** を付けて使用します。

```
pcs constraint colocation remove source_resource target_resource
```

7.4. 制約の表示

設定した制約を表示させるコマンドがいくつかあります。

以下のコマンドは、現在の場所、順序、ロケーションの制約をすべて表示します。

```
pcs constraint list|show
```

以下のコマンドは、現在の場所の制約をリスト表示します。

- **resources** を指定すると、リソースごとに場所の制約が表示されます。これはデフォルトの動作です。
- **nodes** を指定すると、ノードごとに場所の制約が表示されます。
- 特定のリソースまたはノードを指定すると、そのリソースまたはノードの情報のみが表示されます。

```
pcs constraint location [show resources|nodes [specific nodes|resources]] [--full]
```

以下のコマンドは、現在の順序の制約をすべて表示します。**--full** オプションを指定すると、制約の内部 **ID** が表示されます。

```
pcs constraint order show [--full]
```

次のコマンドは、現在のコロケーション制約をリスト表示します。**--full** オプションを指定すると、制約の内部 **ID** が表示されます。

```
pcs constraint colocation show [--full]
```

以下のコマンドは、特定リソースを参照する制約をリスト表示します。

```
pcs constraint ref resource ...
```

第8章 クラスターリソースの管理

本章ではクラスターのリソース管理に使用する各種コマンドについて説明します。次のような手順が含まれます。

- 「リソースを手作業で移動する」
- 「障害発生によるリソースの移動」
- 「クラスターリソースの有効化、無効化、および禁止」
- 「モニター操作の無効化」

8.1. リソースを手作業で移動する

クラスターの設定を無視して、強制的にリソースを現在の場所から移行させることができます。次のような 2 つの状況が考えられます。

- ノードがメンテナンスで、そのノードで実行中の全リソースを別のノードに移行する必要がある
- 個別に指定したリソースを移行する必要がある

ノードで実行中の全リソースを別のノードに移行する場合は、そのノードをスタンバイモードにします。クラスターノードをスタンバイモードにする方法については「スタンバイモード」 を参照してください。

個別に指定したリソースは、以下のいずれかの方法で移行できます。

- 「現在のノードからリソースを移動」 の説明のように、pcs resource move コマンドを使用して現在稼働しているノードからリソースを移動します。
- pcs resource relocate run コマンドを使用して、現在のクラスターのステータス、制約、

リソースの場所、およびその他の設定によって決定される優先ノードにリソースを移動します。このコマンドの詳細は、「リソースの優先ノードへの移動」 を参照してください。

8.1.1. 現在のノードからリソースを移動

現在稼働しているノードからリソースを移動するには以下のコマンドを使用し、リソースの *resource_id* を定義どおりに指定します。移行するリソースを実行するノードを指定する場合は、destination_node を指定します。

```
pcs resource move resource_id [destination_node] [--master] [lifetime=lifetime]
```

注記

pcs resource move コマンドを実行すると、現在実行しているノードでリソースが実行されないように、制約がリソースに追加されます。pcs resource clear コマンドまたは pcs constraint delete コマンドを実行すると、制約を削除できます。このコマンドを実行しても、リソースが必ずしも元のノードに戻る訳ではありません。この時点でリソースが実行できる場所は、リソースの最初の設定方法によって異なります。

pcs resource move コマンドの --master パラメーターを指定すると、制約の範囲がマスターロールに限定され、*resource_id* ではなく *master_id* を指定する必要があります。

任意で pcs resource move コマンドに lifetime パラメーターを設定すると、制約が続く期間を指定できます。ISO 8601 で定義されている形式に従って ライフタイム パラメーターの単位を指定します。この場合、単位は Y （例：年）、M （月用）、W （週）、D （日）、H （時（分）、および S （分）など）として指定する必要があります。

分単位の M と、月単位の M を区別するには、分単位の値の前に PT を指定する必要があります。たとえば、lifetime パラメーターが 5M の場合、間隔は 5 カ月で、有効 期間が PT5M の場合は、間隔が 5 分になります。

lifetime パラメーターは、cluster -recheck-interval クラスター プロパティーで定義される間隔でチェックされます。デフォルト値は 15 分です。このパラメーターをより頻繁に確認する必要がある場合は、次のコマンドを実行して、この値をリセットできます。

```
pcs property set cluster-recheck-interval=value
```

任意で、pcs resource move コマンドに --wait[=*n*] パラメーターを設定し、移行先ノードでリソースが起動するのを待機する秒数を指定できます。この間に、リソースが起動した場合に 0 が返され、リ

ソースが起動していない場合は 1 が返されます。n の値を指定しないと、デフォルトのリソースのタイムアウト値が使用されます。

以下のコマンドは、resource1 リソースを example-node2 ノードに移動し、1 時間 30 分は元のノードに戻らないようにします。

```
pcs resource move resource1 example-node2 lifetime=PT1H30M
```

以下のコマンドは、resource1 リソースを example-node2 ノードに移動し、30 分間実行していたノードに戻らないようにします。

```
pcs resource move resource1 example-node2 lifetime=PT30M
```

リソース制約の詳細は *7章 リソースの制約* を参照してください。

8.1.2. リソースの優先ノードへの移動

フェイルオーバーや管理者の手作業によるノードの移行により、リソースが移行した後、フェイルオーバーの原因となった状況が改善されたとしても、そのリソースが必ずしも元のノードに戻るとは限りません。リソースを優先ノードへ移行するには、以下のコマンドを実行します。優先ノードは、現在のクラスター状態、制約、リソースの場所、およびその他の設定により決定され、時間とともに変更する可能性があります。

```
pcs resource relocate run [resource1] [resource2] ...
```

リソースを指定しないと、すべてのリソースが優先ノードに移行します。

このコマンドは、リソースのスティッキネスを無視し、各リソースの優先ノードを算出します。優先ノードの算出後、リソースを優先ノードに移行する場所の制約を作成します。リソースが移行すると、制約が自動的に削除されます。pcs resource relocate run コマンドによって作成された制約をすべて削除するには、pcs resource relocate clear コマンドを実行します。リソースの現在のステータスと、リソースのスティッキネスを無視した最適なノードを表示するには、pcs resource relocate show コマンドを実行します。

8.2. 障害発生によるリソースの移動

リソースの作成時に、リソースに migration-threshold オプションを設定することで、定義した回数失敗した後にリソースが新しいノードに移動されるように設定できます。このしきい値に一度到達すると、このノードでは、以下が行われるまで、障害が発生したリソースを実行できなくなります。

- 管理者は pcs resource failcount コマンドを使用して、リソースの failcount を手動でリセットします。

- リソースの failure-timeout 値に到達します。

migration-threshold の値は、デフォルトで INFINITY に設定されています。INFINITY は非常に大きいものとして内部的に定義されますが、有限数です。値が 0 の場合は、migration-threshold 機能が無効になります。

注記

リソースの migration-threshold を設定するのと同じことは、リソースが状態を失うことなく別の場所に移動する、移行用のリソースを設定するのと同じです。

以下の例では、dummy_resource という名前のリソースに移行しきい値 10 を追加します。これは、10 回の失敗後にリソースが新しいノードに移動することを示しています。

```
# pcs resource meta dummy_resource migration-threshold=10
```

次のコマンドを使用すると、クラスター全体にデフォルトの移行しきい値を追加できます。

```
# pcs resource defaults migration-threshold=10
```

リソースの現在の障害ステータスと制限を確認するには、pcs resource failcount コマンドを使用します。

移行しきい値の概念には、リソース起動の失敗とリソース停止の失敗の 2 つの例外があります。クラスタープロパティー start-failure-is-fatal が true （デフォルト）に設定されている場合、起動の失敗により failcount が INFINITY に設定され、リソースが常に即座に移動するようになります。start-failure-is-fatal オプションの詳細は、表12.1「クラスターのプロパティー」 を参照してください。

停止時の失敗は、起動時とは若干異なり、極めて重大となります。リソースの停止に失敗し STONITH が有効になっていると、リソースを別のノードで起動できるように、クラスターによるノードのフェンスが行われます。STONITH を有効にしていない場合はクラスターに続行する手段がないため、別のノードでのリソース起動は試行されません。ただし、障害のタイムアウト後に再度停止が試行されます。

8.3. 接続状態変更によるリソースの移動

以下の 2 つのステップに従って、外部の接続が失われた場合にリソースが移動するようにクラスターを設定します。

1. ping リソースをクラスターに追加します。ping リソースは、同じ名前のシステムユーティリティーを使用して、マシン(DNS ホスト名または IPv4/IPv6 アドレスで指定された)のリストにアクセスでき、その結果を使用して pingd と呼ばれるノード属性を維持しているかどうかをテストします。

2. 接続が失われたときに別のノードにリソースを移動させる、リソース場所制約を設定します。

表6.1「リソースのプロパティー」 には、ping リソースに設定できるプロパティーを示します。

表8.1 ping リソースのプロパティー

フィールド	説明
dampen	今後の変更が発生するまでに待機する (弱める) 時間。これにより、クラスターノードが、わずかに異なる時間に接続が失われたことに気が付いたときに、クラスターでリソースがバウンスするのを防ぎます。
multiplier	接続された ping ノードの数は、ノードの数にこの値を掛けて、スコアを取得します。複数の ping ノードが設定された場合に便利です。
host_list	現在の接続状態を判断するために接続するマシン。使用できる値には、解決可能な DNS ホスト名、IPv4 アドレス、および IPv6 アドレスが含まれます。ホストリストのエントリーはスペースで区切られます。

次のコマンド例では、gateway.example.com への接続を検証する ping リソースを作成します。実際には、ネットワークゲートウェイやルーターへの接続を検証します。リソースがすべてのクラスターノードで実行されるように、ping リソースをクローンとして設定します。

```
# pcs resource create ping ocf:pacemaker:ping dampen=5s multiplier=1000
host_list=gateway.example.com clone
```

次の例では、Webserver という名前の既存のリソースに場所の制約ルールを設定します。これにより、Webserver リソースが現在実行しているホストが gateway.example.com に ping できない場合に、Webserver リソースを gateway.example.com に ping できるホストに移動します。

```
# pcs constraint location Webserver rule score=-INFINITY pingd lt 1 or not_defined pingd
```

8.4. クラスターリソースの有効化、無効化、および禁止

「リソースを手作業で移動する」 に説明されている pcs resource move および pcs resource relocate コマンドの他にも、クラスターリソースの動作を制御するのに使用できるコマンドは複数あります。

実行中のリソースを手動で停止し、クラスターが再起動しないようにする場合は、以下のコマンドを使用します。その他の設定 (制約、オプション、失敗など) によっては、リソースが起動した状態のままになる可能性があります。--wait オプションを指定すると、pcs はリソースが停止するまで n 秒間待機し、リソースが停止している場合は 0 を返し、リソースが停止しなかった場合は 1 を返します。n を指定しないと、デフォルトの 60 分に設定されます。

```
pcs resource disable resource_id [--wait[=n]]
```

クラスターがリソースを起動できるようにするには、次のコマンドを使用します。他の設定によっては、リソースが停止したままになることがあります。--wait オプションを指定すると、pcs はリソースが開始するまで最長で n 秒間待機し、リソースが起動した場合は 0、リソースが開始されなかった場合は 1 を返します。n を指定しないと、デフォルトの 60 分に設定されます。

```
pcs resource enable resource_id [--wait[=n]]
```

指定したノードでリソースが実行されないようにする場合は、次のコマンドを使用します。ノードを指定しないと、現在実行中のノードでリソースが実行されないようになります。

```
pcs resource ban resource_id [node] [--master] [lifetime=lifetime] [--wait[=n]]
```

pcs resource ban コマンドを実行すると、場所の制約 -INFINITY がリソースに追加され、リソースが指定のノードで実行されないようにします。pcs resource clear コマンドまたは pcs constraint delete コマンドを実行すると、制約を削除できます。このコマンドを実行しても、リソースが必ずしも指定のノードに戻る訳ではありません。その時点でリソースが実行できる場所は、リソースを最初に設定した方法によって異なります。リソース制約の詳細は *7章 リソースの制約* を参照してください。

pcs resource ban コマンドの --master パラメーターを指定すると、制約の範囲がマスターロールに制限され、*resource_id* ではなく *master_id* を指定する必要があります。

任意で pcs resource ban コマンドに lifetime パラメーターを設定すると、制約が続く期間を指定できます。lifetime パラメーターの単位や、lifetime パラメーターがチェックされる間隔を指定する方法については、「リソースを手作業で移動する」 を参照してください。

任意で、pcs resource ban コマンドに --wait[=*n*] パラメーターを設定し、移行先ノードでリソース

が起動するのを待機する秒数を指定できます。この間に、リソースが起動した場合に 0 が返され、リソースが起動していない場合は 1 が返されます。n の値を指定しないと、デフォルトのリソースのタイムアウト値が使用されます。

指定したリソースを現在のノードで強制的に起動する場合は、pcs resource コマンドの debug-start パラメーターを使用します。これは、クラスターの推奨を無視して、起動しているリソースからの出力を出力します。これは主にデバッグリソースに使用されます。クラスターでのリソースの起動は（ほぼ）常に Pacemaker によって行われ、直接 pcs コマンドでは使用できません。リソースが起動しない原因は、大抵、リソースが誤って設定されているか (システムログでデバッグします)、リソースが起動しないように制約が設定されているか、リソースが無効になっているかのいずれかになります。この場合は、次のコマンドを使用してリソースの設定をテストできます。ただし、通常は、クラスター内でリソースを起動するのに使用しないでください。

debug-start コマンドの形式は以下のようになります。

```
pcs resource debug-start resource_id
```

8.5. モニター操作の無効化

定期的な監視を停止する最も簡単な方法は、監視を削除することです。ただし、一時的に無効にしたい場合もあります。このような場合は、pcs resource update コマンドで enabled="false" を操作の定義に追加します。監視操作を再度有効にするには、操作の定義に enabled="true" を設定します。

pcs resource update コマンドでリソースの操作を更新すると、特に呼び出しのないオプションはデフォルト値にリセットされます。たとえば、カスタムのタイムアウト値 600 を使用して監視操作を設定している場合に以下のコマンドを実行すると、タイムアウト値がデフォルト値の 20 にリセットされます(pcs resource ops default コマンドを使用しても、デフォルト値を設定できます)。

```
# pcs resource update resourceXZY op monitor enabled=false
# pcs resource update resourceXZY op monitor enabled=true
```

このオプションの元の値 600 を維持するために、監視操作に戻す場合は、以下の例のように、その値を指定する必要があります。

```
# pcs resource update resourceXZY op monitor timeout=600 enabled=true
```

8.6. 管理リソース

リソースを unmanaged モードに設定できます。このモードでは、リソースは設定に残りますが Pacemaker はリソースを管理しません。

以下のコマンドは、指定のリソースを **unmanaged** モードに設定します。

```
pcs resource unmanage resource1 [resource2] ...
```

以下のコマンドは、リソースをデフォルトの 管理 モードに設定します。

```
pcs resource manage resource1 [resource2] ...
```

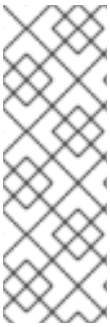
pcs resource manage コマンドまたは **pcs resource unmanage** コマンドを使用して、リソースグループの名前を指定できます。このコマンドは、グループのすべてのリソースで動作するため、1 つのコマンドでグループ内の全リソースをすべて **managed** または **unmanaged** モードに設定し、含まれているリソースを個別に管理できます。

第9章 高度な設定

本章では Pacemaker で対応している高度なリソースタイプと高度な設定機能を説明します。

9.1. リソースのクローン

リソースが複数のノードでアクティブになるよう、リソースをクローンすることができます。たとえば、ノードの分散のために、クローンとなるリソースを使用して、クラスター全体に分散させる IP リソースのインスタンスを複数設定できます。リソースエージェントが対応していれば、任意のリソースのクローンを作成できます。クローンは、1 つのリソースまたは 1 つのリソースグループで構成されます。



注記

同時に複数のノードでアクティブにできるリソースのみがクローンに適しています。たとえば、共有メモリーデバイスから ext4 などの非クラスター化ファイルシステムをマウントする Filesystem リソースのクローンは作成しないでください。ext4 パーティションはクラスターを認識しないため、このファイルシステムは同時に複数のノードから発生する読み取り/書き込み操作には適していません。

9.1.1. クローンリソースの作成と削除

リソースの作成と、そのリソースのクローン作成を同時に行う場合は、次のコマンドを使用します。

```
pcs resource create resource_id standard:provider:type|type [resource options] \
clone [meta clone_options]
```

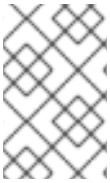
クローンの名前は *resource_id*-clone になります。

1 つのコマンドで、リソースグループの作成と、リソースグループのクローン作成の両方を行うことはできません。

作成済みリソースまたはリソースグループのクローンを作成する場合は、次のコマンドを実行します。

```
pcs resource clone resource_id | group_name [clone_options]...
```

クローンの名前は、*resource_id*-clone または *group_name*-clone になります。



注記

リソース設定の変更が必要なのは、1 つのノードのみです。

注記

制約を設定する場合は、グループ名またはクローン名を必ず使用します。

リソースのクローンを作成すると、そのクローンの名前は、リソース名に -clone を付けた名前になります。次のコマンドは、webfarm という名前のタイプ apache のリソースと、 webfarm -clone という名前のそのリソースのクローンを作成します。

```
# pcs resource create webfarm apache clone
```

注記

別のクローンの後に順序付けされるリソースまたはリソースグループのクローンを作成する場合は、ほとんどの場合 interleave=true オプションを設定する必要があります。これにより、依存されているクローンが同じノードで停止または開始した時に、依存しているクローンのコピーを停止または開始できるようになります。このオプションを設定しない場合は、次のようになります。クローンリソース B がクローンリソース A に依存していると、ノードがクラスターから離れてから戻ってきてから、そのノードでリソース A が起動すると、リソース B の全コピーが、全ノードで再起動します。これは、依存するクローンリソースに interleave オプションが設定されていない場合、そのリソースのすべてのインスタンスは、そのリソースの実行中のインスタンスに依存するためです。

リソースまたはリソースグループのクローンを削除する場合は、次のコマンドを使用します。リソースやリソースグループ自体は削除されません。

pcs resource unclone *resource_id* | *group_name*

リソースオプションの詳細は、「リソースの作成」 を参照してください。

表9.1「リソースのクローンオプション」 には、クローンのリソースに指定できるオプションを示します。

表9.1 リソースのクローンオプション

フィールド	説明
priority, target-role, is-managed	表6.3「リソースのメタオプション」 に従って、クローンされたリソースから継承されるオプション。
clone-max	起動するリソースのコピーの数。デフォルトは、クラスター内のノード数です。
clone-node-max	1 つのノードで起動できるリソースのコピー数。デフォルト値は 1 です。
notify	クローンのコピーを停止したり起動する時に、前もって、およびアクションが成功した時に、他のコピーに通知します。使用できる値は false、true です。デフォルト値は false です。

フィールド	説明
globally-unique	クローンの各コピーで異なる機能を実行させるかどうか。使用できる値は false、trueです。 このオプションの値が false の場合、これらのリソースは実行中のすべての場所で同じように動作するため、マシンごとにアクティブなクローンのコピーは 1 つだけです。 このオプションの値が true の場合、あるマシンで実行しているクローンのコピーは、そのインスタンスが別のノードで実行されているか、同じノード上で実行されているかに関係なく、別のインスタンスと同等ではありません。clone-node-max の値が 1 より大きい場合、デフォルト値は true です。そうでない場合は、デフォルト値は false です。
ordered	コピーを、(並列ではなく) 連続して開始する必要があります。使用できる値は false、true です。デフォルト値は false です。
interleave	2 番目のクローンのすべてのインスタンスが開始または停止するまで待機せずに、2 番目のクローンの同じノードでコピーを開始または停止した直後に最初クローンのコピーが開始または停止できるよう、順序制約の動作を変更します。使用できる値は false、true です。デフォルト値は false です。

フィールド	説明
clone-min	値を指定すると、interleave オプションが true に設定されていても、元のクローンの後に順序付けされたクローンは、元のクローンで指定された数のインスタンスが実行されるまで起動できません。

9.1.2. 制約のクローン作成

ほとんどの場合、アクティブなクラスターノードに対するクローンのコピーは 1 つです。ただし、リソースクローンの clone-max には、クラスター内のノードの合計数より小さい値に設定できます。この場合は、リソースの場所の制約を使用して、クラスターが優先的にコピーを割り当てるノードを指定できます。クローンの ID を使用する点以外、制約は通常のリソースの場合と全く変わらずクローンに記述されます。

次のコマンドは、クラスターがリソースクローン webfarm-clone を node1 に優先的に割り当てる場所の制約を作成します。

```
# pcs constraint location webfarm-clone prefers node1
```

順序制約の動作はクローンでは若干異なります。以下の例では、interleave クローンオプションがデフォルトの false のままであるため、起動する必要がある webfarm-clone のすべてのインスタンスが完了するまで、webfarm-stats のインスタンスは起動しません。webfarm-clone のコピーを開始できない場合にのみ、webfarm-stats がアクティブではなくなります。さらに、webfarm-clone は、webfarm-stats が停止するのを待ってから、それ自体を停止します。

```
# pcs constraint order start webfarm-clone then webfarm-stats
```

通常のリソース (またはリソースグループ) とクローンのコロケーションは、リソースを、クローンのアクティブコピーを持つ任意のマシンで実行できることを意味します。どのコピーが実行しているマシンでそのリソースを実行させるかはクローンが実行している場所とそのリソース自体の場所の優先度に応じて選択されます。

クローン間のコロケーションも可能です。この場合、クローンに対して許可できる場所は、そのクローンが実行中のノード (または実行するノード) に限定されます。割り当ては通常通り行われます。

次のコマンドは、コロケーション制約を作成し、リソース webfarm-stats が webfarm-clone のアクティブなコピーと同じノードで実行されるようにします。

```
# pcs constraint colocation add webfarm-stats with webfarm-clone
```

9.1.3. 粘着性のクローン作成

安定性のある割り当てパターンにするためクローンはデフォルトで若干の粘着性を備えています。**resource-stickiness** の値を指定しないと、クローンは **1** の値を使用します。値を小さくすることで他のリソースのスコア計算への阻害を最小限に抑えながら、**Pacemaker** によるクラスター内の不要なコピーの移動を阻止することができます。

9.2. 多状態のリソース: 複数モードのリソース

多状態リソースはクローンリソースの特殊化です。これにより、インスタンスは **2** つの操作モード(**Master** および **Slave**)のいずれかにすることができます。モードの名前には特別な意味はありませんが、インスタンスの起動時に **Slave** 状態で起動する必要があるという制限があります。

以下のコマンドを実行すると、リソースをマスター/スレーブクローンとして作成できます。

pcs resource create *resource_id standard:provider:type|type* **[***resource options***] master [***master_options***]**

マスター/スレーブクローンの名前は ***resource_id*-master** になります。

注記

Red Hat Enterprise Linux のリリース **7.3** 以前では、マスター/スレーブのクローンの作成は以下の形式で行ってください。

pcs resource create *resource_id standard:provider:type|type* **[***resource options***] -- master [meta** *master_options***]**

また、作成済みのリソースまたはリソースグループからマスター/スレーブリソースを作成することもできます。名前を指定しない場合、マスター/スレーブクローンの名前は ***resource_id*-master** または ***group_name*-master** になります。

pcs resource master *master/slave_name resource_id|group_name* **[***master_options***]**

リソースオプションの詳細は、「リソースの作成」を参照してください。

表9.2「多状態リソースのプロパティー」には、多状態リソースに指定できるオプションが記載されています。

表9.2 多状態リソースのプロパティー

フィールド	説明
id	多状態リソースに付ける名前
priority、target-role、is-managed	表6.3「リソースのメタオプション」を参照してください。
clone-max、clone-node-max、notify、globally-unique、ordered、interleave	表9.1「リソースのクローンオプション」を参照してください。
master-max	master ステータスにプロモートできるリソースのコピー数。デフォルト値は 1 です。

フィールド	説明
master-node-max	master ステータスにプロモートできるリソースのコピー数。デフォルト値は 1 です。

9.2.1. 多状態リソースの監視

マスターリソースのみに監視操作を追加するには、リソースに別の監視操作を追加します。同じリソース上では各監視操作の間隔が異なるようにする必要があります。

以下の例では、ms_resource のマスターリソースに 11 秒の間隔の監視操作を設定します。この監視操作は、10 秒間隔で行われるデフォルトの監視操作とは別に追加されます。

```
# pcs resource op add ms_resource interval=11s role=Master
```

9.2.2. 多状態制約

ほとんどの場合、多状態リソースにはアクティブなクラスターノードごとに 1 つのコピーがあります。そうではない場合は、リソースの場所制約を使用して、クラスターが優先的にコピーを割り当てるノードを指定できます。これらの制約は、通常のリソースと同様に記述されます。

リソースの場所制約については 「場所の制約」 を参照してください。

リソースをマスターにするかスレーブにするかを指定するコロケーション制約を作成することができます。次のコマンドは、リソースのコロケーション制約を作成します。

```
pcs constraint colocation add [master|slave] source_resource with [master|slave]
target_resource [score] [options]
```

コロケーション制約の詳細は 「リソースのコロケーション」 を参照してください。

多状態リソースが含まれる順序制約を設定する場合、リソースに指定できるアクションの 1 つが promote で、リソースがスレーブからマスターへプロモートされることを意味します。さらに、リソー

スを 降格 アクションで指定できます。これは、リソースがマスターからスレーブに降格されることを示します。

順序制約を設定するコマンドは次のようになります。

```
pcs constraint order [action] resource_id then [action] resource_id [options]
```

リソースの順序制約については 「順序の制約」 を参照してください。

9.2.3. 多状態の粘着性 (Stickiness)

安定性のある割り当てパターンを実現するため、多状態リソースはデフォルトで若干の粘着性を備えています。resource-stickiness の値を指定しないと、1 が値として使用されます。値を小さくすることで他のリソースのスコア計算への阻害を最小限に抑えながら、Pacemaker によるクラスター内の不要なコピーの移動を阻止することができます。

9.3. リソースとしての仮想ドメインの設定

pcs resource create コマンドを使用し、VirtualDomain をリソースタイプとして指定して、libvirt 仮想化フレームワークが管理する仮想ドメインをクラスターリソースとして設定できます。

仮想ドメインをリソースとして設定する場合は、以下の点を考慮してください。

- 仮想ドメインは、クラスターリソースとして設定する前に停止する必要があります。
- 仮想ドメインをクラスターリソースにすると、クラスターツールを使用しない限り、起動、停止、または移行を行うことができません。
- クラスターリソースとして設定した仮想ドメインを、ホストの起動時に起動するように設定することはできません。
- すべてのノードは、それぞれの管理される仮想ドメインの必要な設定ファイルおよびストレージデバイスにアクセスできる必要があります。

クラスターが仮想ドメイン内のサービスを管理できるようにする場合は、仮想ドメインをゲストノー

ドとして設定できます。ゲストノードの設定は、「pacemaker_remote サービス」 を参照してください。

仮想ドメインの設定は、仮想化の導入および管理ガイドを参照してください。

表9.3「仮想ドメインリソースのリソースオプション」 は、VirtualDomain リソースに設定できるリソースオプションを説明します。

表9.3 仮想ドメインリソースのリソースオプション

フィールド	デフォルト	説明
config		（必須）この仮想ドメインの **libvirt** 設定ファイルへの絶対パス。
hypervisor	システムに依存	接続先のハイパーバイザーの URI。**virsh --quiet uri** コマンドを実行すると、システムのデフォルト URI を確認できます。
force_stop	**0**	停止時にドメインを常に強制的にシャットダウン (破棄) します。デフォルト動作では、正常なシャットダウンの試行に失敗した後でのみ、強制シャットダウンを実行します。これは、仮想ドメイン（または仮想化バックエンド）が正常なシャットダウンに対応していない場合にのみ **true** に設定する必要があります。
migration_transport	システムに依存	移行中にリモートハイパーバイザーに接続するのに使用されるトランスポート。このパラメーターを省略すると、リソースは **libvirt** のデフォルトトランスポートを使用して、リモートハイパーバイザーに接続します。
migration_network_suffix		専用の移行ネットワークを使用します。移行 URI は、このパラメーターの値をノード名の末尾に追加することにより設定されます。ノード名が完全修飾ドメイン名 (FQDN) の場合は、FQDN の最初のピリオド (.) の直前に接尾辞を挿入します。この設定されたホスト名がローカルで解決可能であり、関連する IP アドレスが優先ネットワークを介して到達可能であることを確認してください。
monitor_scripts		仮想ドメイン内でサービスの監視を追加で実行する場合は、監視するスクリプトのリストとともに、このパラメーターを追加します。**注記**: 監視スクリプトを使用する場合、**start** および **migrate_from** の操作は、すべての監視スクリプトが正常に完了した場合にのみ完了します。この遅延に対応できるように、この操作のタイムアウトを必ず設定してください。
autoset_utilization_cpu	**true**	**true** に設定すると、エージェントは **virsh** から **domainU** の**vCPU**数を検出し、モニターの実行時にリソースの CPU 使用率に置きます。

フィールド	デフォルト	説明
autoset_utilization_hv_memory	**true**	true に設定すると、エージェントは **virsh** から **Max memory** 数を検出し、モニターの実行時にソースの **hv_memory** 使用率にします。
migrateport	ランダムハイポート	このポートは **qemu** の移行 URI で使用されます。これを設定しないと、ポートにはランダムハイポートが使用されます。
snapshot		仮想マシンイメージが保存されるスナップショットディレクトリーへのパス。このパラメーターが設定されている場合、仮想マシンの RAM 状態は、停止時にスナップショットディレクトリーのファイルに保存されます。起動時にドメインのステータスファイルが存在すると、ドメインは、最後に停止する直前の状態に復元されます。このオプションは、**force_stop** オプションと互換性がありません。

VirtualDomain リソースオプションに加えて、allow-migrate メタデータオプションを設定して、リソースの別のノードへのライブマイグレーションを可能にします。このオプションを true に設定すると、状態を失うことなくリソースを移行できます。このオプションがデフォルトの状態である false に設定されている場合、仮想ドメインは最初のノードでシャットダウンされ、2 番目のノードから別のノードに移行するときに 2 番目のノードで再起動されます。

以下の手順に従って VirtualDomain リソースを作成します。

1. **仮想マシンを管理するために VirtualDomain リソースエージェントを作成する場合、Pacemaker では、ディスクのファイルに仮想マシンの xml 設定ファイルをダンプする必要があります。たとえば、guest1 という名前の仮想マシンを作成した場合は、ホストのどこかのファイルに xml をダンプします。任意のファイル名を使用できますが、この例では /etc/pacemaker/guest1.xml を使用します。**

```
# virsh dumpxml guest1 > /etc/pacemaker/guest1.xml
```

2. **ゲストノードが実行している場合はシャットダウンします。Pacemaker は、クラスターで設定されているノードを起動します。**

3. **pcs resource create コマンドを使用して、VirtualDoman リソースを設定します。たとえば、以下のコマンドは、VM という名前の VirtualDomain リソースを設定します。allow-migrate オプションは true に設定されているため、pcs resource move VM *nodeX* コマンドはライブ移行として実行されます。**

-

```
# pcs resource create VM VirtualDomain config=.../vm.xml \
      migration_transport=ssh meta allow-migrate=true
```

9.4. PACEMAKER_REMOTE サービス

pacemaker_remote サービスを使用すると、corosync を実行していないノードをクラスターに統合し、クラスターが実際のクラスターノードであるかのようにリソースを管理できます。

pacemaker_remote サービスが提供する機能には以下が含まれます。

- pacemaker_remote サービスを使用すると、RHEL 7.7 の Red Hat サポート制限である 32 ノードを超えた拡張が可能になります。
- pacemaker_remote サービスを使用すると、仮想環境をクラスターリソースとして管理したり、仮想環境内の個別のサービスをクラスターリソースとして管理したりできます。

pacemaker_remote サービスの記述には、以下の用語が使用されます。

- クラスターノード: 高可用性サービスを実行するノード(pacemaker および corosync)。
- リモートノード: pacemaker_remote を実行するノードで、corosync クラスターのメンバーシップを必要とせずにクラスターにリモートで統合します。リモートノードは、ocf:pacemaker:remote リソースエージェントを使用するクラスターリソースとして設定されます。
- ゲストノード: pacemaker_remote サービスを実行する仮想ゲストノード。仮想ゲストリソースはクラスターにより管理されます。クラスターにより起動し、リモートノードとしてクラスターに統合されます。
- pacemaker_remote: Pacemaker クラスター環境のリモートノードおよびゲストノード (KVM および LXC) 内でリモートアプリケーション管理を実行できるサービスデーモン。このサービスは、corosync を実行していないノード上でリソースをリモートで管理できる Pacemaker のローカルリソース管理デーモン (LRMD) の強化バージョンです。
- l xc - libvirt-lxc Linux コンテナードライバーで定義される Linux コンテナー。

pacemaker_remote サービスを実行している Pacemaker クラスターには、以下の特徴があります。

- リモートノードおよびゲストノードは、pacemaker_remote サービスを実行します（仮想マシン側で必要な設定はほとんどありません）。
- クラスターノードで実行しているクラスタースタック(pacemaker および corosync)はリモートノードで pacemaker_remote サービスに接続し、クラスターに統合できます。
- クラスターノードで実行しているクラスタースタック(pacemaker および corosync)はゲストノードを起動し、ゲストノードで pacemaker_remote サービスに即座に接続して、クラスターに統合できます。

クラスターノードと、クラスターノードが管理するリモートおよびゲストノードの主な違いは、リモートおよびゲストノードはクラスタースタックを実行しないことです。そのため、リモートおよびゲストノードには以下の制限があります。

- クォーラムでは実行されない
- フェンスデバイスのアクションを実行しません
- クラスターの指定コントローラー (DC) として機能できない
- それ自体が完全な pcs コマンドを実行していない

その一方で、リモートノードおよびゲストノードは、クラスタースタックに関連するスケーラビリティーの制限に拘束されません。

このような制限事項以外に、リモートノードとゲストノードは、リソース管理に関してクラスターノードと同様に動作し、リモートノードとゲストノード自体をフェンスすることができます。クラスターは、各リモートノードおよびゲストノードのリソースを完全に管理および監視できます。このようなノードに制約を構築したり、スタンバイ状態にしたり、pcs コマンドを使用してクラスターノードでその他のアクションを実行したりできます。リモートノードおよびゲストノードは、クラスターノードと同様にクラスターステータスの出力に表示されます。

9.4.1. ホストとゲストの認証

クラスターノードと pacemaker_remote の間の接続には、TLS (Transport Layer Security) が使用され、PSK (Pre-Shared Key) の暗号化と TCP 上の認証 (デフォルトで 3121 ポートを使用) でセキュア化されます。つまり、クラスターノードと、pacemaker_remote を実行しているノードは、同じ秘密鍵を共有する必要があります。デフォルトでは、クラスターノードとリモートノードの両方でこのキーを /etc/pacemaker/authkey に配置する必要があります。

Red Hat Enterprise Linux 7.4 以降、pcs cluster node add-guest コマンドはゲストノードの authkey を設定し、pcs cluster node add-remote コマンドはリモートノードの authkey をセットアップします。

9.4.2. ゲストノードリソースのオプション

ゲストノードとして機能するように仮想マシンまたは LXC リソースを設定する場合、仮想マシンを管理する VirtualDomain リソースを作成します。VirtualDomain リソースに設定できるオプションの説明は、表9.3「仮想ドメインリソースのリソースオプション」 を参照してください。

VirtualDomain リソースオプションのほかにも、メタデータオプションはリソースをゲストノードとして定義し、接続パラメーターを定義します。Red Hat Enterprise Linux 7.4 では、pcs cluster node add-guest コマンドを使用してこれらのリソースオプションを設定する必要があります。7.4 以前のリリースでは、リソースを作成する際に、これらのオプションを設定できます。表9.4「KVM/LXC リソースをリモートノードとして設定するメタデータオプション」 では、このようなメタデータオプションについて説明します。

表9.4 KVM/LXC リソースをリモートノードとして設定するメタデータオプション

フィールド	デフォルト	説明
remote-node	<none>	このリソースが定義するゲストノードの名前。リソースをゲストノードとして有効にし、ゲストノードの識別に使用される一意名を定義します。**警告**: この値を、リソースやノードの ID と重複させることはできません。
remote-port	3121	**pacemaker_remote**へのゲスト接続に使用するカスタムポートを設定します
remote-addr	ホスト名として使用される **remote-node** 値	リモートノードの名前がゲストのホスト名ではない場合に接続する IP アドレスまたはホスト名
remote-connect-timeout	60s	保留中のゲスト接続がタイムアウトするまでの時間

9.4.3. リモートノードリソースのオプション

リモートノードは、ocf:pacemaker:remote がリソースエージェントとして指定されたクラスターリソースとして定義されます。Red Hat Enterprise Linux 7.4 では、pcs cluster node add-remote コマンドを使用してこのリソースを作成する必要があります。7.4 以前のリリースでは、pcs resource create コマンドを使用してこのリソースを作成できます。表9.5「リモートノードのリソースオプション」 は、remote リソースに設定できるリソースオプションを説明します。

表9.5 リモートノードのリソースオプション

フィールド	デフォルト	説明
reconnect_interval	0	リモートノードへのアクティブな接続が切断された後、リモートノードへの再接続を試みる前に待機する時間 (秒単位)。この待機期間は繰り返し発生します。待機期間の後に再接続に失敗した場合、待機期間の後に、新しい再接続が試行されます。このオプションが使用されると、Pacemaker は待機期間の後に無限にリモートノードへ接続を試みます。
server		接続するサーバーの場所。IP アドレスまたはホスト名を指定できます。
port		接続する TCP ポート。

9.4.4. ポートのデフォルトの場所の変更

Pacemaker または pacemaker_remote のいずれかのポートのデフォルトの場所を変更する必要がある場合は、これらのデーモンの両方に影響を与える PCMK_remote_port 環境変数を設定できます。この環境変数は、以下のように /etc/sysconfig/pacemaker ファイルに置くことで有効にできます。

```
#==#==# Pacemaker Remote
...
#
# Specify a custom port for Pacemaker Remote connections
PCMK_remote_port=3121
```

特定のゲストノードまたはリモートノードによって使用されるデフォルトのポートを変更する場合は、PCMK_remote_port 変数をそのノードの /etc/sysconfig/pacemaker ファイルに設定する必要があります。また、ゲストノードまたはリモートノードの接続を作成するクラスターリソースを同じポート番号で設定する必要があります（ゲストノードの場合は remote-port メタデータオプション、リモートノードの場合は port オプションを使用します）。

9.4.5. 設定の概要: KVM ゲストノード

本セクションでは、libvirt と KVM 仮想ゲストを使用して、Pacemaker で仮想マシンを起動し、そのマシンをゲストノードとして統合する手順の概要を説明します。

1. 「リソースとしての仮想ドメインの設定」 で説明するように、VirtualDomain リソースを設定します。

2. Red Hat Enterprise Linux 7.3 以前を実行しているシステムでは、以下の手順に従って、すべてのクラスターノードと仮想マシン上で、パス /etc/pacemaker/authkey で同じ暗号鍵を配置します。これにより、リモート通信や認証を保護することができます。

   a. すべてのノードで以下のコマンドセットを入力して、セキュアなパーミッションを持つ authkey ディレクトリーを作成します。

   ```
   # mkdir -p --mode=0750 /etc/pacemaker
   # chgrp haclient /etc/pacemaker
   ```

   b. 以下のコマンドは、暗号化キーを作成する方法の 1 つを示しています。キーは 1 度だけ作成し、すべてのノードにコピーする必要があります。

   ```
   # dd if=/dev/urandom of=/etc/pacemaker/authkey bs=4096 count=1
   ```

3. Red Hat Enterprise Linux 7.4 では、すべての仮想マシンで以下のコマンドを実行し、pacemaker_remote パッケージをインストールし、pcsd サービスを起動し、これを起動時に実行できるようにし、ファイアウォールを介して TCP ポート 3121 を許可します。

   ```
   # yum install pacemaker-remote resource-agents pcs
   # systemctl start pcsd.service
   # systemctl enable pcsd.service
   # firewall-cmd --add-port 3121/tcp --permanent
   # firewall-cmd --add-port 2224/tcp --permanent
   # firewall-cmd --reload
   ```

   Red Hat Enterprise Linux 7.3 以前では、すべての仮想マシンで以下のコマンドを実行し、pacemaker_remote パッケージをインストールし、pacemaker_remote サービスを起動してこれを起動時に実行できるようにし、ファイアウォールを介して TCP ポート 3121 を許可します。

   ```
   # yum install pacemaker-remote resource-agents pcs
   # systemctl start pacemaker_remote.service
   # systemctl enable pacemaker_remote.service
   # firewall-cmd --add-port 3121/tcp --permanent
   # firewall-cmd --add-port 2224/tcp --permanent
   # firewall-cmd --reload
   ```

4.

各仮想マシンに、すべてのノードが認識できる静的ネットワークアドレスと一意なホスト名を割り当てます。ゲスト仮想マシンに静的 IP アドレスを設定する方法については、『仮想化の導入および管理ガイド』を参照してください。

5.

Red Hat Enterprise Linux 7.4 以降では、以下のコマンドを使用して、既存の VirtualDomain リソースをゲストノードに変換します。このコマンドは、追加するゲストノードではなく、クラスターノードで実行する必要があります。リソースを変換する以外にも、このコマンドは /etc/pacemaker/authkey をゲストノードにコピーし、ゲストノードで pacemaker_remote デーモンを起動して有効にします。

```
pcs cluster node add-guest hostname resource_id [options]
```

Red Hat Enterprise Linux 7.3 以前では、以下のコマンドを使用して既存の VirtualDomain リソースをゲストノードに変換します。このコマンドは、追加するゲストノードではなく、クラスターノードで実行する必要があります。

```
pcs cluster remote-node add hostname resource_id [options]
```

6.

VirtualDomain リソースの作成後は、クラスターの他のノードと同じように、ゲストノードを扱うことができます。たとえば、クラスターノードから実行される次のコマンドのように、リソースを作成し、リソースにリソース制約を設定してゲストノードで実行できます。Red Hat Enterprise Linux 7.3 時点では、ゲストノードをグループで含むことができ、ストレージデバイス、ファイルシステム、および VM をグループ化できます。

```
# pcs resource create webserver apache configfile=/etc/httpd/conf/httpd.conf op monitor
interval=30s
# pcs constraint location webserver prefers guest1
```

9.4.6. 設定概要: リモートノード (Red Hat Enterprise Linux 7.4)

本セクションでは、Red Hat Enterprise 7.4 において Pacemaker リモートノードを設定し、そのノードを既存の Pacemaker クラスター環境に統合する手順の概要を説明します。

1.

リモートノードを設定するノードで、ローカルファイアウォールを介してクラスター関連のサービスを許可します。

```
# firewall-cmd --permanent --add-service=high-availability
success
# firewall-cmd --reload
success
```

注記

iptables を直接使用する場合や、firewalld 以外のファイアウォールソリューションを使用する場合は、TCP ポート 2224 および 3121 のポートを開くだけです。

2. **リモートノードに pacemaker_remote デーモンをインストールします。**

```
# yum install -y pacemaker-remote resource-agents pcs
```

3. **リモートノードで pcsd を起動し、有効にします。**

```
# systemctl start pcsd.service
# systemctl enable pcsd.service
```

4. **リモートノードとして追加するノードに pcs を認証していない場合は認証します。**

```
# pcs cluster auth remote1
```

5. **以下のコマンドを使用して、リモートノードリソースをクラスターに追加します。このコマンドは、関連するすべての設定ファイルを新しいノードに同期し、ノードを起動し、起動時に pacemaker_remote を開始するように設定します。このコマンドは、追加するリモートノードではなく、クラスターノードで実行する必要があります。**

```
# pcs cluster node add-remote remote1
```

6. **remote リソースをクラスターに追加した後、リモートノードをクラスター内の他のノードを処理するのと同じように処理できます。たとえば、以下のコマンドをクラスターノードから実行すると、リソースを作成し、そのリソースにリソース制約を配置して、リモートノードで実行できます。**

```
# pcs resource create webserver apache configfile=/etc/httpd/conf/httpd.conf op monitor
interval=30s
# pcs constraint location webserver prefers remote1
```

警告

リソースグループ、コロケーション制約、または順序制約でノード接続リソースを利用しないでください。

7. リモートノードのフェンスリソースを設定します。リモートノードは、クラスターノードと同じ方法でフェンスされます。クラスターノードと同様に、リモートノードで使用するフェンスリソースを設定します。リモートノードはフェンスアクションを開始できないことに注意してください。クラスターノードのみが、実際に別のノードに対してフェンシング操作を実行できます。

9.4.7. 設定概要: リモートノード (Red Hat Enterprise Linux 7.3 以前)

本セクションでは、Red Hat Enterprise 7.3 以前のシステムにおいて Pacemaker リモートノードを設定し、そのノードを既存の Pacemaker クラスター環境に統合する手順の概要を説明します。

1. リモートノードを設定するノードで、ローカルファイアウォールを介してクラスター関連のサービスを許可します。

```
# firewall-cmd --permanent --add-service=high-availability
success
# firewall-cmd --reload
success
```

注記

iptables を直接使用する場合や、firewalld 以外のファイアウォールソリューションを使用する場合は、TCP ポート 2224 および 3121 のポートを開くだけです。

2. リモートノードに pacemaker_remote デーモンをインストールします。

```
# yum install -y pacemaker-remote resource-agents pcs
```

3. 適切に通信するには、すべてのノード (クラスターノードとリモートノードの両方) に同じ認証キーがインストールされている必要があります。既存ノードにすでにキーがある場合、そ

のキーを使用してリモートノードにコピーします。それ以外の場合はリモートノードで新しいキーを作成します。

リモートノードで以下のコマンドを実行し、セキュアなパーミッションを持つ認証キーのディレクトリーを作成します。

```
# mkdir -p --mode=0750 /etc/pacemaker
# chgrp haclient /etc/pacemaker
```

以下のコマンドは、リモートノードで暗号化キーを作成する方法の 1 つを示しています。

```
# dd if=/dev/urandom of=/etc/pacemaker/authkey bs=4096 count=1
```

4. リモートノードで **pacemaker_remote** デーモンを開始して有効にします。

```
# systemctl enable pacemaker_remote.service
# systemctl start pacemaker_remote.service
```

5. クラスターノードにて、リモートノードの認証キーと同じパスを持つ共有された認証キーの保存場所を作成し、そのディレクトリーにキーをコピーします。この例では、キーが作成されたリモートノードからキーがコピーされます。

```
# mkdir -p --mode=0750 /etc/pacemaker
# chgrp haclient /etc/pacemaker
# scp remote1:/etc/pacemaker/authkey /etc/pacemaker/authkey
```

6. クラスターノードから以下のコマンドを入力して、**remote** リソースを作成します。この場合、リモートノードは **remote1** になります。

```
# pcs resource create remote1 ocf:pacemaker:remote
```

7. **remote** リソースの作成後、リモートノードをクラスター内の他のノードを処理するのと同じように処理できます。たとえば、以下のコマンドをクラスターノードから実行すると、リソースを作成し、そのリソースにリソース制約を配置して、リモートノードで実行できます。

```
# pcs resource create webserver apache configfile=/etc/httpd/conf/httpd.conf op monitor
interval=30s
# pcs constraint location webserver prefers remote1
```

警告

リソースグループ、コロケーション制約、または順序制約でノード接続リソースを利用しないでください。

8. リモートノードのフェンスリソースを設定します。リモートノードは、クラスターノードと同じ方法でフェンスされます。クラスターノードと同様に、リモートノードで使用するフェンスリソースを設定します。リモートノードはフェンスアクションを開始できないことに注意してください。クラスターノードのみが、実際に別のノードに対してフェンシング操作を実行できます。

9.4.8. システムアップグレードおよび pacemaker_remote

Red Hat Enterprise Linux 7.3 では、アクティブな Pacemaker リモートノードで pacemaker_remote サービスが停止すると、ノードを停止する前にクラスターがノードからリソースを正常に移行するようになりました。これにより、クラスターからノードを削除せずに、ソフトウェアのアップグレードやその他の定期的なメンテナンスを実行できるようになりました。ただし、pacemaker_remote がシャットダウンすると、クラスターは即座に再接続を試みます。リソースのモニタータイムアウト内に pacemaker_remote が再起動されない場合、クラスターは監視操作が失敗したと見なします。

アクティブな Pacemaker リモートノードで pacemaker_remote サービスが停止したときに監視が失敗しないようにするには、以下の手順に従って、pacemaker_remote を停止する可能性があるシステム管理を実行する前に、ノードをクラスターから外すことができます。

警告

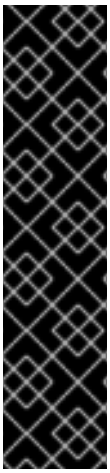
Red Hat Enterprise Linux 7.2 以前のリリースでは、現在クラスターに統合されているノードで pacemaker_remote が停止すると、クラスターはそのノードをフェンスします。yum update のプロセスの一部として自動的に停止した場合、システムが使用不可の状態になる可能性があります（特に pacemaker_remoteと同時にカーネルもアップグレードされる場合）。Red Hat Enterprise Linux 7.2 以前のリリースでは、以下の手順にしたがって、pacemaker_remote を停止する可能性のあるシステム管理を実行する前に、ノードをクラスターから外す必要があります。

1. ノードからすべてのサービスを移動する pcs resource disable *resourcename* を使用して、ノードの接続リソースを停止します。ゲストノードの場合は、仮想マシンも停止するため、メンテナンスを実行するには、クラスターの外部で（たとえば、virshを使用して）VM を起動する必要があります。

2. 必要なメンテナンスを実行します。

3. ノードをクラスターに戻す準備ができたら、pcs resource enable でリソースを再度有効にします。

9.5. DOCKER コンテナーの PACEMAKER サポート (テクノロジープレビュー)



重要

Docker コンテナーの Pacemaker サポートは、テクノロジープレビュー目的でのみ実現されています。テクノロジープレビューの意味については、テクノロジープレビュー機能のサポート範囲 を参照してください。

テクノロジープレビューであるこの機能には 1 つの例外があります。Red Hat Enterprise Linux 7.4 以降、Red Hat は、Red Hat Openstack Platform (RHOSP) デプロイメントで Pacemaker バンドルの使用を完全にサポートします。

Pacemaker は、必要なインフラストラクチャーで Docker コンテナーを起動するための特殊構文 (*bundle*) に対応しています。Pacemaker バンドルを作成したら、バンドルがカプセル化する Pacemaker リソースを作成できます。

- 「Pacemaker バンドルリソースの設定」 では、Pacemaker バンドルを作成するコマンドについて説明し、各バンドルパラメーターに定義できるパラメーターについてまとめた表が記載されています。

- 「バンドルでの Pacemaker リソースの設定」 では、Pacemaker バンドルに含まれているリソースの設定について説明しています。

- 「Pacemaker バンドルの制限」 では、Pacemaker バンドルの制限について説明しています。

- 「Pacemaker バンドル設定の例」 は、Pacemaker バンドルの設定例を説明しています。

9.5.1. Pacemaker バンドルリソースの設定

Docker コンテナーの Pacemaker バンドルを作成するコマンド構文は以下の通りです。このコマンドを使用すると、その他のリソースをカプセル化しないバンドルが作成されます。バンドルでクラスターリソースを作成する方法については、「バンドルでの Pacemaker リソースの設定」 を参照してください。

```
pcs resource bundle create bundle_id container docker [container_options] [network network_options] [port-map port_options]... [storage-map storage_options]... [meta meta_options] [--disabled] [--wait[=n]]
```

必要な ***bundle_id*** パラメーターは、バンドルに対する一意の名前にする必要があります。**--disabled** オプションを指定すると、バンドルは自動的に起動されません。**--wait** オプションを指定すると、Pacemaker は、バンドルが起動するまで最大 **n** 秒待機し、成功時、またはエラー時に 1 を返します。**n** を指定しないと、デフォルトの 60 分に設定されます。

以下のセクションでは、Pacemaker バンドルの各要素に設定できるパラメーターを説明します。

9.5.1.1. Docker パラメーター

表9.6「Docker コンテナーのパラメーター」 は、バンドルに設定できる docker コンテナーオプションを説明します。

注記

Pacemaker で docker バンドルを設定する前に、Docker をインストールし、バンドルの実行が許可されているすべてのノードで完全に設定済みの Docker イメージを提供する必要があります。

表9.6 Docker コンテナーのパラメーター

フィールド	デフォルト	説明
image		Docker イメージタグ (必須)
replicas	正の場合は **promoted-max** の値。そうでない場合は 1 です。	起動するコンテナーインスタンスの数を指定する正の整数

フィールド	デフォルト	説明
replicas-per-host	1	単一ノードで起動できるコンテナーインスタンスの数を指定する正の整数
promoted-max	0	正であれば、非負の整数は、マスターロールにおいてサービスを実行できる多くのレプリカとともに、コンテナー化されたサービスが複数のサービスとして扱われる必要があることを示しています。
network		これが指定されている場合、これは Docker コンテナーのネットワーク設定として **docker run** コマンドに渡されます。
run-command	**/usr/sbin/pacemaker_remoted** バンドルにリソースが含まれる場合、それ以外はなし	このコマンドは、起動する際にコンテナー内で実行されます ("PID 1")。バンドルにリソースが含まれる場合、このコマンドは **pacemaker_remoted** デーモンを起動する必要があります（ただし、他のタスクを実行するスクリプトを使用することもできます）。
options		**docker run** コマンドに渡す、追加のコマンドラインオプション

9.5.1.2. (バンドルネットワークパラメーター

表9.7「バンドルリソースネットワークパラメーター」 では、バンドルに設定できる ネットワークオプションを説明します。

表9.7 バンドルリソースネットワークパラメーター

フィールド	デフォルト	説明
add-host	TRUE	TRUE で **ip-range-start** が使用されている場合、Pacemaker は自動的に、コンテナー内の **/etc/hosts** ファイルに各レプリカ名と割り当てられた IP のエントリーがあることを確認します。
ip-range-start		指定されている場合、Pacemaker は各コンテナーインスタンスに対して暗黙的な **ocf:heartbeat:IPaddr2** リソースを作成します。この IP アドレスは、Docker 要素の **replicas** パラメーターとして指定された数だけ連続アドレスを使用します。これらのアドレスは、ホストのネットワークから使用して、コンテナー内のサービスに到達できます。ただし、コンテナー自身の中では表示されません。現在サポートされているアドレスは、IPv4 のみです。
host-netmask	32	**ip-range-start** が指定されている場合、IP アドレスは、この CIDR ネットマスクで（多数のビットとして）作成されます。

フィールド	デフォルト	説明
host-interface		**ip-range-start** が指定されている場合、IP アドレスはこのホストインターフェイスに作成されます（デフォルトでは、IP アドレスから決定されます）。
control-port	3121	バンドルに Pacemaker リソースが含まれる場合、クラスターは、コンテナー内の Pacemaker Remote との通信に、この整数 の TCP ポートを使用します。コンテナーがデフォルトポートでリッスンできない場合、これはコンテナーが **ip-range-start**（この場合は **replicas-per-host** は 1 である必要があります）ではなくホストのネットワークを使用している場合や、バンドルがデフォルトポートですでにリッスンしている可能性がある場合に発生する可能性があります。ホストまたはコンテナーで設定されている **PCMK_remote_port** 環境変数は、バンドル接続に対して無視されます。 Pacemaker バンドル設定が **control-port** パラメーターを使用する場合、バンドルに独自の IP アドレスがある場合は、corosync を実行しているすべてのクラスターノード、およびその IP アドレスでポートを開く必要があります。代わりに、バンドルが **network="host"** コンテナーパラメーターを設定している場合は、すべてのクラスターノードから各クラスターノードの IP アドレスでポートを開く必要があります。

注記

レプリカは、バンドル ID とダッシュそしてゼロから始まる整数カウンターで名前が付けられます。たとえば、httpd-bundle という名前のバンドルで replicas=2 が設定されている場合、そのコンテナーの名前は httpd-bundle-0 および httpd-bundle-1 になります。

ネットワークパラメーターに加えて、バンドルに port-map パラメーターをオプションで指定できます。表9.8「バンドルリソースポートマップパラメーター」 では、これらの port-map パラメーターを説明します。

表9.8 バンドルリソースポートマップパラメーター

フィールド	デフォルト	説明
id		ポートマッピングの一意の名前 (必須)
port		これが指定されている場合、ホストネットワーク上のこの TCP ポート番号(**ip-range-start** が指定されている場合はコンテナーの割り当てられた IP アドレス上)への接続がコンテナーネットワークに転送されます。**port** または **range** のうち 1 つがポートマッピングで指定される必要があります。

フィールド	デフォルト	説明
internal-port	**port**の値	**port** と **internal-port** が指定されている場合、ホストのネットワーク上の **ポート** への接続は、コンテナーネットワーク上のこのポートに転送されます。
range		**range** が指定されている場合は、ホストネットワーク上 (**ip-range-start** が指定されている場合は、コンテナーの割り当てられた IP アドレス上)上のこれらの TCP ポート番号(*first_port-last_port*)への接続が、コンテナーネットワーク内の同じポートに転送されます。**port** または **range** のうち 1 つだけをポートマッピングに指定する必要があります。

注記

バンドルにリソースが含まれる場合、Pacemaker は自動的に control-port をマッピングします。そのため、ポートマッピングでそのポートを指定する必要はありません。

9.5.1.3. バンドルストレージパラメーター

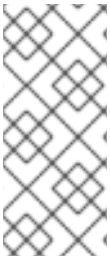
必要に応じて、バンドルの storage-map パラメーターを設定できます。表9.9「バンドルリソースストレージマッピングパラメーター」 では、これらのパラメーターを説明します。

表9.9 バンドルリソースストレージマッピングパラメーター

フィールド	デフォルト	説明
id		ストレージマッピングの一意の名前 (必須)
source-dir		コンテナーにマッピングされるホストファイルシステム上の絶対パス。**storage-map** パラメーターを設定する際には、**source-dir** と **source-dir-root** のいずれかを指定する必要があります。
source-dir-root		各コンテナーインスタンスのホスト上で異なるサブディレクトリーを使用した、コンテナーにマッピングされるホストのファイルシステム上のパスの開始。このサブディレクトリーには、バンドル名と同じ名前が付けられ、ダッシュと 0 から始まる整数カウンターも加えられます。**storage-map** パラメーターを設定する際には、**source-dir** と **source-dir-root** の 1 つだけを指定する必要があります。

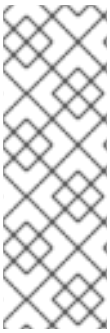
フィールド	デフォルト	説明
target-dir		ホストストレージがマッピングされるコンテナー内のパス名 (必須)
options		ストレージをマッピングする際に使用するファイルシステムマウントオプション

ホスト上のサブディレクトリーが source-dir-root パラメーターを使用して命名される方法の例として、source-dir-root=/path/to/my/directory、target-dir=/srv/appdata、バンドルの名前が replicas=2 の mybundle という名前で、クラスターは mybundle- 0 と mybundle - 1 というホスト名を持つ 2 つのコンテナーインスタンスを作成します。そして、コンテナーを実行しているホストに /path/to/my/directory/mybundle-0 と / path/to/my /directory/mybundle-1 の 2 つのディレクトリー を作成します。各コンテナーには、これらのディレクトリーのいずれかが与えられ、コンテナー内で実行されているアプリケーションには /srv/appdata というディレクトリーが表示されます。



注記

Pacemaker は、ソースディレクトリーがすでにホストに存在しない場合の動作を定義しません。ただし、コンテナーテクノロジーまたはそのリソースエージェントがソースディレクトリーを作成します。



注記

バンドルに Pacemaker リソースが含まれる場合、Pacemaker は自動的に source-dir=/etc/pacemaker/authkeytarget-dir=/ etc/pacemaker/authkey および source-dir-root=/var/log/pacemaker/bundlestarget-dir=/var/ logと同等のものをコンテナーにマップするため、storage-map パラメーターを設定するときにこれらのパスを指定する必要はありません。

重要

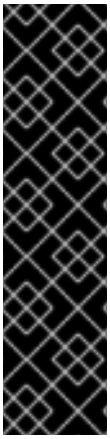
PCMK_authkey_location 環境変数は、クラスターのノード上の /etc/pacemaker/authkey のデフォルト以外に設定することはできません。

9.5.2. バンドルでの Pacemaker リソースの設定

バンドルは必要に応じて、1 つの Pacemaker クラスターリソースを含めることができます。バンドルに含まれていないリソースと同様に、クラスターリソースには、操作、インスタンス属性、メタデータ属性を定義することができます。バンドルにリソースが含まれている場合は、コンテナーイメージに Pacemaker Remote デーモンを含める必要があり、ip-range-start または control-port をバンドルで設定する必要があります。Pacemaker は、接続用に暗黙的な ocf:pacemaker:remote リソースを作成し、コンテナー内で Pacemaker Remote を起動して、Pacemaker リモートを使用してリソースを監視

および管理します。バンドルに複数のコンテナーインスタンス（レプリカ）がある場合、Pacemaker リソースは暗黙的なクローンとして機能します。バンドルが promoted-max オプションをゼロ以上に設定した場合、これは多状態クローンになります。

コマンドの bundle パラメーターと、リソースを含める バンドル ID を指定して、pcs resource create コマンドで Pacemaker バンドルにリソースを作成します。リソースを含む Pacemaker バンドルの作成例は、「Pacemaker バンドル設定の例」 を参照してください。



重要

リソースを含むバンドルのコンテナーにはアクセス可能なネットワーク環境が必要です。それにより、クラスターノードの Pacemaker はコンテナー内の Pacemaker Remote にコンタクトできます。たとえば、docker オプション --net=none はリソースと併用しないでください。デフォルトの（コンテナー内の個別のネットワーク領域を使用）は、ip-range-start パラメーターと組み合わせて機能します。docker オプション --net=host が使用されている場合（コンテナーがホストのネットワーク領域を共有する）、バンドルごとに一意の control-port パラメーターを指定する必要があります。ファイアウォールでは、control-port へのアクセスを許可する必要があります。

9.5.2.1. ノード属性とバンドルリソース

バンドルにクラスターリソースが含まれる場合、リソースエージェントはマススコアなどのノード属性を設定する可能性があります。ただし、コンテナーでは、ノードが属性を取得するべきかどうかはっきりしません。

コンテナーがホストされているノードに関係なく、コンテナーが同じ共有ストレージを使用している場合は、バンドルノード自体でマススコアを使用することが適切です。一方、コンテナーが、基礎となるホストからエクスポートされたストレージを使用する場合は、基礎となるホストでマススコアを使用することがより適切です。これは特定の状況に依存するため、container-attribute-target リソースメタデータ属性を使用すると、使用するアプローチを指定できます。host に設定されている場合、ユーザー定義のノード属性は基礎となるホストでチェックされます。その他の場合は、ローカルノード (この場合はバンドルノード) が使用されます。この動作はユーザー定義の属性にのみ適用されます。クラスターは、#uname などのクラスター定義属性に対してローカルノードを常にチェックします。

container-attribute-target が host に設定されている場合、クラスターは追加の環境変数をリソースエージェントに渡して、ノード属性を適切に設定します。

9.5.2.2. メタデータ属性とバンドルリソース

バンドルで設定されているメタデータ属性は、バンドルに含まれるリソースや、バンドルに対して Pacemaker によって作成されたリソースによって継承されます。これには、priority、target-role、is-

managed などのオプションが含まれます。

9.5.3. Pacemaker バンドルの制限

Pacemaker バンドルは以下の制限で動作します。

- バンドルはグループに含まれていないことや、pcs コマンドで明示的にクローン化されていない場合もあります。これには、バンドルが含むリソースや、バンドルに対して Pacemaker によって明示的に作成されたリソースが含まれます。ただし、バンドルが replicas の値が 1 より大きい場合、バンドルはクローンであるかのように動作することに注意してください。

- バンドルが管理されていない場合や、クラスターがメンテナンスモードの際に Pacemaker を再起動すると、バンドルが不具合を起こすことがあります。

- バンドルには、インスタンス属性、使用率属性、または操作がありません。しかし、バンドルに含まれるリソースには、これらがあります。

- リソースを含むバンドルは、バンドルが個別の control-port を使用する場合にのみ、Pacemaker リモートノードで実行できます。

9.5.4. Pacemaker バンドル設定の例

以下の例では、httpd- bundle というバンドル ID で Pacemaker bundle を作成します。これには、httpd というリソース ID を持つ ocf:heartbeat:apache リソースが含まれます。

この手順には、以下の前提設定が必要です。

- Docker が、クラスターの各ノードでインストールされ有効化されている。

- pcmktest:httpという名前の既存の Docker イメージがあります。

- コンテナーイメージに、Pacemaker Remote デーモンが含まれている。

- コンテナーイメージに、設定済みの Apache Web サーバーが含まれている。

- クラスターのすべてのノードには、/var/local/containers/httpd-bundle-0、/var/local/containers/httpd-bundle-1、および /var/local/containers/httpd-bundle-2 ディレクトリーがあり、Web サーバーの root の index.html ファイルが含まれます。実稼働環境では、単一の共有ドキュメント root の方が高くなりますが、この例では、この設定では、Web サーバーに接続し、提供される index.html ファイルを検証できるように、各ホスト上の index.html ファイルを異なるものにすることが可能です。

この手順により、Pacemaker バンドルに以下のパラメーターが設定されます。

- バンドル ID は httpd-bundle です。

- 以前に設定された Docker コンテナーイメージは pcmktest:http です。

- この例は、3 コンテナーインスタンスを起動します。

- この例では、コマンドラインオプション --log-driver=journald を docker run コマンドに渡します。このパラメーターは必須ではありませんが、追加のオプションを docker コマンドに渡す方法を示すために含まれています。--log-driver=journald の値は、コンテナー内のシステムログが基礎となるホストの systemd ジャーナルにログインすることを意味します。

- Pacemaker は、3 つの連続した暗黙的な ocf:heartbeat:IPaddr2 リソースを作成します。これは、各コンテナーイメージ用に 1 つ、IP アドレス 192.168.122.131 で始まります。

- IP アドレスは、ホストインターフェイス eth0 で作成されます。

- IP アドレスは、CIDR ネットマスクが 24 で作成されます。

- この例では、http-port のポートマップ ID を作成します。コンテナーの割り当てられた IP アドレスのポート 80 への接続がコンテナーネットワークに転送されます。

- この例では、httpd-root のストレージマップ ID を作成します。このストレージマッピングについて以下で説明します。

- source-dir-root の値は /var/local/containers です。これは、各コンテナーインスタンスに対してホスト上の異なるサブディレクトリーを使用して、コンテナーにマップされるホストのファイルシステム上のパスの開始を指定します。

- target-dir の値は /var/www/html で、ホストストレージがマップされるコンテナー内のパス名を指定します。

- ファイルシステム rw マウントオプションは、ストレージのマッピング時に使用されます。

- この例のコンテナーにはリソースが含まれているため、Pacemaker は自動的にコンテナーに source-dir=/etc/pacemaker/authkey と同等のものをマッピングします。そのため、ストレージマッピングにそのパスを指定する必要はありません。

この例では、既存のクラスター設定が temp-cib.xml という名前の一時ファイルに配置され、temp-cib.xml.deltasrc という名前のファイルにコピーされます。クラスター設定に対するすべての変更は、tmp-cib.xml ファイルに対して行われます。udpates が完了すると、この手順では pcs cluster cib-push コマンドの diff-against オプションを使用して、設定ファイルへの更新のみがアクティブな設定ファイルにプッシュされるようにします。

```
# pcs cluster cib tmp-cib.xml
# cp tmp-cib.xml tmp-cib.xml.deltasrc
# pcs -f tmp.cib.xml resource bundle create httpd-bundle \
container docker image=pcmktest:http replicas=3 \
options=--log-driver=journald \
network ip-range-start=192.168.122.131 host-interface=eth0 \
host-netmask=24 port-map id=httpd-port port=80 \
storage-map id=httpd-root source-dir-root=/var/local/containers \
target-dir=/var/www/html options=rw \
# pcs -f tmp-cib.xml resource create httpd ocf:heartbeat:apache \
statusurl=http://localhost/server-status bundle httpd-bundle
# pcs cluster cib-push tmp-cib.xml diff-against=tmp-cib.xml.deltasrc
```

9.6. 使用と配置ストラテジー

Pacemaker は、すべてのノードのリソース割り当てスコアに従って、リソースを配置する場所を決定します。このリソースは、リソースのスコアが最も高いノードに割り当てられます。この割り当てスコアは、リソースの制約、リソース スティッキネス設定、各ノードのリソース の以前の障害履歴、各ノードの使用率などの要素の組み合わせから派生します。

すべてのノードでリソース割り当てスコアが等しい場合、デフォルトの配置ストラテジーにより、Pacemaker は、負荷を分散するために、割り当てられたリソースの数が最も少ないノードを選択します。各ノードのリソースの数が等しい場合は、CIB の最初の対象ノードがリソースを実行するのに選択されます。

ただし、多くの場合、リソースが使用するノードの容量 (メモリーや I/O など) の割合は、状況によって大きく異なります。そのため、ノードに割り当てられているリソースの数のみを考慮して、いつでも思想的な負荷分散が行われるとは限りません。また、リソースの合計要件が、指定の容量を超えるように配置されている場合は、リソースが完全に起動できなくなるか、パフォーマンスが低下した状態で起動することがあります。このような要因を考慮するために、Pacemaker では次の要素を設定できます。

- 特定のノードの容量
- 特定のリソースが必要な容量
- リソースの配置の全体的なストラテジー

以下のセクションでは、これらのコンポーネントの設定方法を説明します。

9.6.1. 使用率属性

ノードの容量、またはリソースが必要な容量を設定するには、ノードとリソースに *utilization attributes* を使用します。これを行うには、リソースの使用率変数を設定し、その変数に値を割り当ててリソースに必要なものを示してから、同じ使用率変数をノードに設定し、その変数に値を割り当ててそのノードが提供するものを示します。

設定に応じて使用率属性に名前を付け、設定に必要な数だけ名前と値のペアを定義できます。使用率属性の値は整数である必要があります。

Red Hat Enterprise Linux 7.3 では、pcs コマンドを使用して使用率属性を設定できます。

以下の例では、2 つのノードに対して CPU 容量の使用率属性を設定し、属性 cpu という名前を設定します。また、RAM 容量の使用率属性を設定し、属性 memory に命名します。この例では、以下のように設定されています。

- ノード 1 は、CPU 容量 2 と、RAM 容量 2048 を指定するように定義されています。

- ノード 2 は、CPU 容量 4 と、RAM 容量 2048 を指定するように定義されています。

```
# pcs node utilization node1 cpu=2 memory=2048
# pcs node utilization node2 cpu=4 memory=2048
```

次の例では、3 つの異なるリソースに必要な、同じ使用率属性を指定します。この例では、以下のように設定されています。

- リソース ダミーには、CPU 容量 1 と RAM 容量 1024 が 必要です。

- リソース ダミーに は、2 の CPU 容量と RAM 容量 2048 が必要です。

- リソース ダミー拡大 には、CPU 容量 1 および 3072 の RAM 容量が必要です。

```
# pcs resource utilization dummy-small cpu=1 memory=1024
# pcs resource utilization dummy-medium cpu=2 memory=2048
# pcs resource utilization dummy-large cpu=3 memory=3072
```

使用率属性で定義されているリソースの要件を満たすのに十分な空き容量があれば、ノードはリソースに適格と見なされます。

9.6.2. 配置ストラテジー

ノードが提供する容量とリソースが必要とする容量を設定したら、placement-strategy クラスタープロパティーを設定する必要があります。そうしないと、容量の設定が効果がありません。クラスターのプロパティーの詳細は 12章*Pacemaker クラスターのプロパティー* を参照してください。

placement-strategy クラスタープロパティーには、4 つの値を使用できます。

- default - 使用率の値は全く考慮されません。リソースは、割り当てスコアに従って割り当てられます。スコアが同じであれば、リソースはノード間で均等に分散されます。

- utilization - 使用率の値は、ノードが適格であると見なされるかどうか（つまり、リソース

の要件を満たすのに十分な空き容量があるかどうか）を決定するときにのみ考慮されます。負荷分散は、ノードに割り当てられたリソースの数に基づいて行われます。

- balanced - 使用率の値は、ノードがリソースを提供する資格があるかどうかを決定するときと負荷分散を行う際に考慮されるため、リソースのパフォーマンスを最適化する方法でリソースを分散しようとします。

- minimal - 使用率の値は、ノードがリソースを提供する資格があるかどうかを決定する場合にのみ考慮されます。負荷分散は、できるだけ少ないノードにリソースを集中させて、残りのノードで電力を節約できるようにします。

以下の例のコマンドでは、placement-strategy の値を balanced に設定します。このコマンドを実行すると、Pacemaker は、複雑な一連のコロケーション制約を使用せずに、リソースからの負荷がクラスター全体に均等に分散されるようにします。

```
# pcs property set placement-strategy=balanced
```

9.6.3. リソースの割り当て

以下のサブセクションでは、Pacemaker がリソースを割り当てる仕組みをまとめています。

9.6.3.1. ノードの優先順位

Pacemaker は、以下のストラテジーに従ってリソースを割り当てる際に優先されるノードを決定します。

- 重みが最も高いノードが最初に消費されます。ノードの重みは、ノードの状態を表すためにクラスターによって維持されるスコアです。

- ノードの重みが、複数のノードで同じ場合は、以下のようになります。

  - placement-strategy クラスタープロパティーが default または utilization の場合：

    - 割り当てられているリソースの数が最も少ないノードが最初に使用されます。

    - 割り当てられているリソースの数が等しい場合は、CIB に登録されている最初

の対象ノードが最初に使用されます。

- placement-strategy クラスタープロパティーが分散されている 場合:
  - 空き容量が最も多いノードが最初に使用されます。
  - ノードの空き容量が等しい場合は、割り当てられているリソースの数が最も少ないノードが最初に使用されます。
  - ノードの空き容量が等しく、割り当てられているリソースの数が等しい場合は、CIB に最初に登録されている対象ノードが最初に使用されます。
- placement-strategy クラスタープロパティーが 最小限 の場合、CIB に一覧表示されている最初の対象ノードが最初に使用されます。

9.6.3.2. ノードの容量

Pacemaker は、以下のストラテジーに従って、どのノードに最も空き容量があるかを判断します。

- 使用率属性が 1 種類だけ定義されている場合、空き容量は単純な数値比較となります。
- 定義されている使用属性の種類が複数になる場合は、ほとんどの属性タイプで数値が最も高いノードの空き容量が、最も大きくなります。以下に例を示します。
  - NodeA の空き CPU が多く、NodeB の空きメモリーが多い場合は、互いの空き容量は等しくなります。
  - NodeA の空き CPU が多く、NodeB の空きメモリーとストレージが多い場合、NodeB の方が空き容量が多くなります。

9.6.3.3. リソースの割り当て設定

Pacemaker は、以下のストラテジーに従って、最初に割り当てられるリソースを決定します。

- 優先度の最も高いリソースが最初に割り当てられます。リソースの優先度の設定は、表6.3「リソースのメタオプション」 を参照してください。

- リソースの優先度が等しい場合は、リソースのシャッフルを防ぐために、実行中のノードで最も高いスコアを持つリソースが最初に割り当てられます。

- リソースが実行しているノードのリソーススコアが等しい場合や、リソースが実行していない場合は、優先ノードで最もスコアが高いリソースが最初に割り当てられます。この場合、優先ノードのリソーススコアが等しい場合は、CIB に最初に登録されている実行可能なリソースが最初に割り当てられます。

9.6.4. リソース配置ストラテジーガイドライン

リソースに対する Pacemaker の配置ストラテジーの動作効率を最適化するためにも、システムを設定する際には以下を考慮してください。

- 物理容量が十分にあることを確認します。

  ノードの物理容量が、通常の状態でほぼ最大に使用されているとすると、フェイルオーバーの際に問題が発生する可能性があります。使用率機能がなくても、タイムアウトや二次障害が発生する可能性があります。

- ノードに設定する容量にバッファーをいくつか構築します。

  物理的に存在するよりもわずかに多くのノードリソースが通知されます。ここでは、Pacemaker リソースが、設定した CPU、メモリーなどを常に 100% 使用しないことが想定されます。このような状況は、オーバーコミットと呼ばれることもあります。

- リソースの優先度を指定します。

  クラスターがサービスを犠牲にする場合、犠牲にするサービスは一番重要でないことが理想的です。最も重要なリソースが最初にスケジュールされるように、リソースの優先度が適切に設定されていることを確認してください。リソースの優先度の設定は、表6.3「リソースのメタオプション」 を参照してください。

9.6.5. NodeUtilization リソースエージェント (Red Hat Enterprise Linux 7.4 以降)

Red Hat Enterprise Linux 7.4 は NodeUtilization リソースエージェントに対応しています。NodeUtilization エージェントは、使用可能な CPU、ホストメモリーの可用性、およびハイパーバイザーのメモリーの可用性に関するシステムパラメーターを検出し、このようなパラメーターを CIB に追加します。エージェントをクローンリソースとして実行して、各ノードにこのようなパラメーターを自動的に入力することができます。

NodeUtilization リソースエージェントと、このエージェントのリソースオプションの詳細は、pcs resource describe NodeUtilization コマンドを実行してください。

9.7. PACEMAKER で管理されていないリソースの依存関係の起動順の設定 (RED HAT ENTERPRISE LINUX 7.4 以降)

クラスターは、クラスターが管理していない依存関係を持つリソースを含めることができます。この場合は、Pacemaker を起動する前にその依存関係を起動し、Pacemaker が停止した後に停止する必要があります。

Red Hat Enterprise Linux 7.4 では、systemd resource-agents-deps ターゲットを使用して、この状況に対応するように起動順序を設定できます。このターゲットに対して systemd ドロップインユニットを作成でき、Pacemaker はこのターゲットに対して相対的な順序を適切に設定します。

たとえば、クラスターに、クラスターが管理していない外部サービス foo に依存するリソースが含まれている場合は、以下を含むドロップインユニット /etc/systemd/system/resource-agents-deps.target.d/foo.conf を作成できます。

```
[Unit]
Requires=foo.service
After=foo.service
```

ドロップインユニットを作成したら、systemctl daemon-reload コマンドを実行します。

この方法で指定するクラスターの依存関係はサービス以外のものとなります。たとえば、/srv にファイルシステムをマウントする依存関係がある場合は、systemd のドキュメントに従って、systemd ファイル srv.mount を作成し、ここで説明しているように、foo . service の srv.mount を使用してドロップインユニットを作成し、ディスクがマウントされた後に Pacemaker が起動することを確認します。

9.8. SNMP での PACEMAKER クラスターを照会 (RED HAT ENTERPRISE LINUX 7.5 以降)

Red Hat Enterprise Linux 7.5 では、pcs_snmp_agent デーモンを使用して、SNMP でデータにつ

いて Pacemaker クラスターを照会できます。pcs_snmp_agent デーモンは、agentx プロトコルを使用してマスターエージェント(snmpd)に接続する SNMP エージェントです。pcs_snmp_agent エージェントは、マスターエージェントにデータのみを提供するため、スタンドアロンエージェントとしては機能しません。

以下の手順では、システムが Pacemaker クラスターで SNMP を使うための基本的な設定を行います。この手順は、SNMP を使用してクラスターのデータを取得する、クラスターの各ノードで行います。

1. クラスターの各ノードに pcs-snmp パッケージをインストールします。これにより、snmp デーモンを提供する net-snmp パッケージもインストールされます。

```
# yum install pcs-snmp
```

2. /etc/snmp/snmpd.conf 設定ファイルに以下の行を追加して、snmpd デーモンを master agentx として設定します。

```
master agentx
```

3. /etc/snmp/snmpd.conf 設定ファイルに以下の行を追加して、同じ SNMP 設定で pcs_snmp_agent を有効にします。

```
view    systemview    included   .1.3.6.1.4.1.32723.100
```

4. pcs_snmp_agent サービスを開始します。

```
# systemctl start pcs_snmp_agent.service
# systemctl enable pcs_snmp_agent.service
```

5. 設定を確認するには、pcs status でクラスターのステータスを表示し、SNMP からデータを取得して、出力に対応するかどうかを確認します。SNMP を使用してデータを取得する際には、プリミティブなリソースのみが与えられることに注意してください。

以下の例は、アクションに 1 回失敗した実行中のクラスターでの pcs status コマンドの出力を示しています。

```
# pcs status
Cluster name: rhel75-cluster
Stack: corosync
```

```
Current DC: rhel75-node2 (version 1.1.18-5.el7-1a4ef7d180) - partition with quorum
Last updated: Wed Nov 15 16:07:44 2017
Last change: Wed Nov 15 16:06:40 2017 by hacluster via cibadmin on rhel75-node1

2 nodes configured
14 resources configured (1 DISABLED)

Online: [ rhel75-node1 rhel75-node2 ]

Full list of resources:

 fencing        (stonith:fence_xvm):    Started rhel75-node1
 dummy5 (ocf::pacemaker:Dummy): Stopped (disabled)
 dummy6 (ocf::pacemaker:Dummy): Stopped
 dummy7 (ocf::pacemaker:Dummy): Started rhel75-node2
 dummy8 (ocf::pacemaker:Dummy): Started rhel75-node1
 dummy9 (ocf::pacemaker:Dummy): Started rhel75-node2
 Resource Group: group1
     dummy1     (ocf::pacemaker:Dummy): Started rhel75-node1
     dummy10    (ocf::pacemaker:Dummy): Started rhel75-node1
 Clone Set: group2-clone [group2]
     Started: [ rhel75-node1 rhel75-node2 ]
 Clone Set: dummy4-clone [dummy4]
     Started: [ rhel75-node1 rhel75-node2 ]

Failed Actions:
* dummy6_start_0 on rhel75-node1 'unknown error' (1): call=87, status=complete,
exitreason='',
    last-rc-change='Wed Nov 15 16:05:55 2017', queued=0ms, exec=20ms
```

```
# snmpwalk -v 2c -c public localhost PACEMAKER-PCS-V1-MIB::pcmkPcsV1Cluster
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterName.0 = STRING: "rhel75-cluster"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterQuorate.0 = INTEGER: 1
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterNodesNum.0 = INTEGER: 2
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterNodesNames.0 = STRING: "rhel75-node1"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterNodesNames.1 = STRING: "rhel75-node2"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterCorosyncNodesOnlineNum.0 = INTEGER: 2
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterCorosyncNodesOnlineNames.0 = STRING:
"rhel75-node1"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterCorosyncNodesOnlineNames.1 = STRING:
"rhel75-node2"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterCorosyncNodesOfflineNum.0 = INTEGER: 0
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterPcmkNodesOnlineNum.0 = INTEGER: 2
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterPcmkNodesOnlineNames.0 = STRING:
"rhel75-node1"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterPcmkNodesOnlineNames.1 = STRING:
"rhel75-node2"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterPcmkNodesStandbyNum.0 = INTEGER: 0
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterPcmkNodesOfflineNum.0 = INTEGER: 0
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterAllResourcesNum.0 = INTEGER: 11
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterAllResourcesIds.0 = STRING: "fencing"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterAllResourcesIds.1 = STRING: "dummy5"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterAllResourcesIds.2 = STRING: "dummy6"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterAllResourcesIds.3 = STRING: "dummy7"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterAllResourcesIds.4 = STRING: "dummy8"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterAllResourcesIds.5 = STRING: "dummy9"
```

```
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterAllResourcesIds.6 = STRING: "dummy1"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterAllResourcesIds.7 = STRING: "dummy10"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterAllResourcesIds.8 = STRING: "dummy2"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterAllResourcesIds.9 = STRING: "dummy3"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterAllResourcesIds.10 = STRING: "dummy4"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterRunningResourcesNum.0 = INTEGER: 9
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterRunningResourcesIds.0 = STRING:
"fencing"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterRunningResourcesIds.1 = STRING:
"dummy7"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterRunningResourcesIds.2 = STRING:
"dummy8"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterRunningResourcesIds.3 = STRING:
"dummy9"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterRunningResourcesIds.4 = STRING:
"dummy1"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterRunningResourcesIds.5 = STRING:
"dummy10"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterRunningResourcesIds.6 = STRING:
"dummy2"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterRunningResourcesIds.7 = STRING:
"dummy3"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterRunningResourcesIds.8 = STRING:
"dummy4"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterStoppedResroucesNum.0 = INTEGER: 1
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterStoppedResroucesIds.0 = STRING:
"dummy5"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterFailedResourcesNum.0 = INTEGER: 1
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterFailedResourcesIds.0 = STRING: "dummy6"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterFailedResourcesIds.0 = No more variables
left in this MIB View (It is past the end of the MIB tree)
```

9.9. クリーンノードのシャットダウンで停止するようにリソースを設定 (RED HAT ENTERPRISE LINUX 7.8 以降)

クラスターノードがシャットダウンしたときの Pacemaker のデフォルトの応答は、シャットダウンが正常なシャットダウンであっても、そのノードで実行中のすべてのリソースを停止し、別の場所でリソースを復元することです。Red Hat Enterprise Linux 7.8 以降では、ノードが正常にシャットダウンすると、ノードに接続されているリソースがノードにロックされ、シャットダウンしたノードがクラスターに再度参加するときに再び起動するまで、他の場所で起動できないように、Pacemaker を設定できます。これにより、ノードのリソースをクラスター内の他のノードにフェイルオーバーせずに、サービスの停止が許容できるメンテナンスウィンドウ中にノードの電源を切ることができます。

9.9.1. クリーンノードシャットダウンで停止するようにリソースを設定するためのクラスタープロパティー

ノードの正常なシャットダウンでリソースがフェイルオーバーしないようにする機能は、以下のクラスタープロパティーで実装されます。

shutdown-lock

このクラスタープロパティーをデフォルト値の **false** に設定すると、クラスターは、ノードで正常にシャットダウンされているノードでアクティブなリソースを復旧します。このプロパティーを **true** に設定すると、適切にシャットダウンしているノードでアクティブなリソースは、クラスターに再参加した後にノードで再起動するまで別の場所で起動できません。

shutdown-lock プロパティーはクラスターノードまたはリモートノードのいずれかで機能しますが、ゲストノードでは機能しません。

shutdown-lock を **true** に設定すると、ノードがダウンしたときに 1 つのクラスターリソースのロックを削除し、次のコマンドを使用してノードで手動更新を実行してリソースを別の場所で起動できるようになります。

```
pcs resource refresh resource --node node
```

リソースのロックが解除されると、クラスターはリソースを別の場所に移動できるようになることに注意してください。リソースのスティッキネスの値または場所の設定を使用することにより、これが発生する可能性を抑制できます。

注記

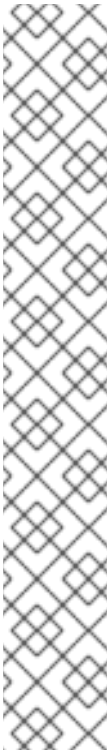
手動更新は、最初に次のコマンドを実行するとリモートノードで機能します。

1. リモートノードで **systemctl stop pacemaker_remote** コマンドを実行してノードを停止します。
2. **pcs resource disable *remote-connection-resource*** コマンドを実行します。

その後、リモートノードで手動更新を実行できます。

shutdown-lock-limit

このクラスタープロパティーをデフォルト値の **0** 以外の値に設定すると、シャットダウンを開始してから指定した時間内にノードが再参加しない場合に、他のノードの復旧にリソースが利用可能になります。ただし、この間隔は **cluster -recheck-interval** クラスター プロパティーの値よりも頻繁にチェックされないことに注意してください。



注記

shutdown-lock-limit プロパティーは、以下のコマンドを最初に実行した場合に限りリモートノードで動作します。

1. **リモートノードで systemctl stop pacemaker_remote コマンドを実行してノードを停止します。**

2. **pcs resource disable *remote-connection-resource* コマンドを実行します。**

このコマンドを実行すると、shutdown-lock-limit で指定した時間が経過すると、リモートノード上で実行しているリソースが他のノードの復旧に利用できます。

9.9.2. shutdown-lock クラスタープロパティーの設定

以下の例では、サンプルのクラスターで shutdown-lock クラスタープロパティーを true に設定し、ノードをシャットダウンして再起動したときの影響を示しています。この例のクラスターは、z1.example.com、z2.example.com、および z3.example.com の 3 つのノードで設定されます。

1. **shutdown-lock プロパティーを true に設定し、その値を確認します。この例では、shutdown-lock-limit プロパティーはデフォルト値の 0 を維持します。**

```
[root@z3.example.com ~]# pcs property set shutdown-lock=true
[root@z3.example.com ~]# pcs property list --all | grep shutdown-lock
shutdown-lock: true
shutdown-lock-limit: 0
```

2. **クラスターのステータスを確認します。この例では、3 番 目 と 5 番目 のリソースが z1.example.com で実行しています。**

```
[root@z3.example.com ~]# pcs status
...
Full List of Resources:
...
* first (ocf::pacemaker:Dummy): Started z3.example.com
* second (ocf::pacemaker:Dummy): Started z2.example.com
* third (ocf::pacemaker:Dummy): Started z1.example.com
```

```
* fourth (ocf::pacemaker:Dummy): Started z2.example.com
* fifth (ocf::pacemaker:Dummy): Started z1.example.com
...
```

3. **z1.example.com をシャットダウンします。これにより、そのノードで実行されているリソースを停止します。**

```
[root@z3.example.com ~] # pcs cluster stop z1.example.com
Stopping Cluster (pacemaker)...
Stopping Cluster (corosync)...
```

pcs status コマンドを実行すると、ノード z1.example.com がオフラインで、z1.example.com で実行していたリソースがノードの停止時に LOCKED であることを示しています。

```
[root@z3.example.com ~]# pcs status
...

Node List:
* Online: [ z2.example.com z3.example.com ]
* OFFLINE: [ z1.example.com ]

Full List of Resources:
...
* first (ocf::pacemaker:Dummy): Started z3.example.com
* second (ocf::pacemaker:Dummy): Started z2.example.com
* third (ocf::pacemaker:Dummy): Stopped z1.example.com (LOCKED)
* fourth (ocf::pacemaker:Dummy): Started z3.example.com
* fifth (ocf::pacemaker:Dummy): Stopped z1.example.com (LOCKED)
...
```

4. **クラスターサービスを z1.example.com で再度起動し、クラスターに再参加できるようにします。ロックされたリソースは、そのノードで開始する必要がありますが、いったん起動すると、必ずしも同じノードに留まるわけではありません。**

```
[root@z3.example.com ~]# pcs cluster start z1.example.com
Starting Cluster...
```

この例では、3 番目と 5 番目のリソースが z1.example.com ノードで復元されます。

```
[root@z3.example.com ~]# pcs status
...

Node List:
* Online: [ z1.example.com z2.example.com z3.example.com ]
```

```
Full List of Resources:
..
* first (ocf::pacemaker:Dummy): Started z3.example.com
* second (ocf::pacemaker:Dummy): Started z2.example.com
* third (ocf::pacemaker:Dummy): Started z1.example.com
* fourth (ocf::pacemaker:Dummy): Started z3.example.com
* fifth (ocf::pacemaker:Dummy): Started z1.example.com

...
```

第10章 クラスタークォーラム

Red Hat Enterprise Linux High Availability Add-On クラスターは、スプリットブレインの状況を回避するために、votequorum サービスをフェンシングと併用します。クラスターの各システムには多くの投票数が割り当てられ、過半数の票を取得しているものだけがクラスターの操作を継続できます。サービスは、すべてのノードに読み込むか、いずれのノードにも読み込まないようにする必要があります。サービスをクラスターノードのサブセットに読み込むと、結果が予想できなくなります。votequorumサービスの設定および操作の詳細は、votequorum (5)の man ページを参照してください。

10.1. クォーラムオプションの設定

pcs cluster setup コマンドを使用してクラスターを作成する場合は、クォーラム設定の特殊な機能を設定できます。表10.1「クォーラムオプション」 では、このようなオプションをまとめています。

表10.1 クォーラムオプション

オプション	説明
--auto_tie_breaker	これを有効にすると、クラスターは、決定論的に最大 50% のノードが同時に失敗しても存続されます。クラスターパーティション、または **auto_tie_breaker_node** で設定された **nodeid** （設定されていない場合は最小の **nodeid** ）と通信しているノードのセットはクォーラムのままになります。その他のノードはクォーラムに達しません。 **auto_tie_breaker** オプションを指定すると、均等の分割でクラスターが動作を継続できるようになるため、主に偶数個のノードがあるクラスターで使用されます。複数で不均等の分割などの、より複雑な障害は、「クォーラムデバイス」 で説明されているようにクォーラムデバイスを使用することが推奨されます。**auto_tie_breaker** オプションは、クォーラムデバイスと互換性がありません。
--wait_for_all	有効にすると、最低 1 回、同時にすべてのノードが現れた後に、初回だけ、クラスターがクォーラムに達します。 **wait_for_all** オプションは、主にクォーラムデバイス **lms** (last man standing)アルゴリズムを使用する 2 ノードクラスターおよび偶ノードクラスターに使用されます。 **wait_for_all** オプションは、クラスターに 2 つのノードがあり、クォーラムデバイスを使用せず、**auto_tie_breaker** が無効になっている場合に自動的に有効になります。**wait_for_all** を明示的に 0 に設定すると、このオプションをオーバーライドできます。
--last_man_standing	有効にすると、クラスターは特定の状況で **expected_votes** およびクォーラムを動的に再計算できます。このオプションを有効にする場合は、**wait_for_all** を有効にする必要があります。**last_man_standing** オプションには、クォーラムデバイスとの互換性がありません。
--last_man_standing_window	クラスターがノードを失うと、**expected_votes** およびクォーラムを再計算するまで待機する時間（ミリ秒単位）。

これらのオプションの設定および使用に関する詳細は、votequorum(5)の man ページを参照してください。

10.2. クォーラム管理コマンド (RED HAT ENTERPRISE LINUX 7.3 以降)

クラスターの稼働後、以下のクラスタークォーラムコマンドを実行できます。

次のコマンドは、クォーラムの設定を表示します。

```
pcs quorum [config]
```

次のコマンドは、クォーラムのランタイム状態を表示します。

```
pcs quorum status
```

長時間クラスターからノードを取り除き、それらのノードが失われてクォーラムが失われた場合は、**pcs quorum expected-votes** コマンドを使用して、ライブクラスターの expected_votes パラメーターの値を変更できます。これにより、クォーラムがない場合でも、クラスターは操作を継続できます。

警告

ライブクラスターで期待される票数 (vote) を変更する場合は、細心の注意を払って行ってください。期待される票数を手動で変更したために、実行しているクラスターが 50% 未満となる場合は、クラスターの他のノードを個別に起動してクラスターサービスを開始できるため、データの破損や予期せぬ結果が発生することがあります。この値を変更する場合は、wait_for_all パラメーターが有効になっていることを確認する必要があります。

次のコマンドは、ライブクラスターで期待される票数を、指定の値に設定します。これはライブクラスターにのみ影響し、設定ファイルは変更されません。再読み込みの際に、expected_votes の値は設定ファイルの値にリセットされます。

```
pcs quorum expected-votes votes
```

10.3. クォーラムオプションの変更 (RED HAT ENTERPRISE LINUX 7.3 以降)

Red Hat Enterprise Linux 7.3 より、pcs quorum update コマンドを使用してクラスターの一般的なクォーラムオプションを変更できます。実行中のシステムの quorum.two_node および quorum.expected_votes オプションを変更できます。その他すべてのクォーラムオプションについては、このコマンドを実行するには、クラスターを停止する必要があります。クォーラムオプションの詳細は、votequorum(5)の man ページを参照してください。

pcs quorum update コマンドの形式は次のとおりです。

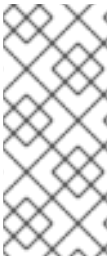
```
pcs quorum update [auto_tie_breaker=[0|1]] [last_man_standing=[0|1]] [last_man_standing_window=
[time-in-ms] [wait_for_all=[0|1]]
```

以下の一連のコマンドは、wait_for_all クォーラムオプションを変更し、オプションの更新済みステータスを表示します。クラスターの稼働中はこのコマンドを実行できないことに注意してください。

```
[root@node1:~]# pcs quorum update wait_for_all=1
Checking corosync is not running on nodes...
Error: node1: corosync is running
Error: node2: corosync is running

[root@node1:~]# pcs cluster stop --all
node2: Stopping Cluster (pacemaker)...
node1: Stopping Cluster (pacemaker)...
node1: Stopping Cluster (corosync)...
node2: Stopping Cluster (corosync)...

[root@node1:~]# pcs quorum update wait_for_all=1
Checking corosync is not running on nodes...
node2: corosync is not running
node1: corosync is not running
Sending updated corosync.conf to nodes...
node1: Succeeded
node2: Succeeded

[root@node1:~]# pcs quorum config
Options:
  wait_for_all: 1
```

10.4. クォーラムアンブロック (QUORUM UNBLOCK) コマンド

クォーラムに達していない状態でクラスターにリソース管理を続行させたい場合、以下のコマンドを使用してクォーラムの確立時にクラスターがすべてのノードを待たないようにします。



注記

このコマンドは細心の注意を払って使用する必要があります。このコマンドを実行する前に、現在クラスターにないノードの電源を切り、共有リソースにアクセスできない状態であることを確認する必要があります。

```
# pcs cluster quorum unblock
```

10.5. クォーラムデバイス

Red Hat Enterprise Linux 7.4 は、個別のクォーラムを設定する機能に完全に対応しています。主要な用途は、クォーラムルールによって許容されるノード障害の数よりも多くのノード障害をクラスターが許容するようにすることです。クォーラムデバイスは、偶数のノードを持つクラスターに推奨されます。2 ノードクラスターでクォーラムデバイスを使用すると、スプリットブレインの状況で存続するノードをより適切に判別できます。

クォーラムデバイスを設定する場合は、以下を考慮する必要があります。

- クォーラムデバイスは、クォーラムデバイスを使用するクラスターと同じ場所にある別の物理ネットワークで実行することが推奨されます。理想としては、クォーラムデバイスホストを、メインクラスターとは別のラックに置くか、少なくとも別の PSU に置くようにします。corosync リングと同じネットワークセグメントには置かないようにしてください。
- 複数のクォーラムデバイスをクラスターで同時に使用することはできません。
- 複数のクォーラムデバイスをクラスターで同時に使用することはできません。ただし、複数のクラスターが 1 つのクォーラムデバイスを同時に使用することはできます。アルゴリズムやクォーラムオプションはクラスターノード自体に保存されるため、同じクォーラムデバイスを使用する各クラスターが、複数のアルゴリズムやクォーラムオプションを使用できます。たとえば、単一のクォーラムデバイスは、ffsplit (fifty/fifty 分割)アルゴリズムを持つ 1 つのクラスター、およびレム(last man standing)アルゴリズムを持つ 2 番目 の クラスターで使用できます。
- クォーラムデバイスは、既存のクラスターノードで実行しないでください。

10.5.1. クォーラムデバイスパッケージのインストール

クラスターにクォーラムデバイスを設定するには、以下のパッケージをインストールする必要があります。

- 既存クラスターのノードに **corosync-qdevice** をインストールします。

```
[root@node1:~]# yum install corosync-qdevice
[root@node2:~]# yum install corosync-qdevice
```

- クォーラムデバイスホストに **pcs** および **corosync-qnetd** をインストールします。

```
[root@qdevice:~]# yum install pcs corosync-qnetd
```

- **pcsd** サービスを起動し、システムの起動時に **pcsd** がクォーラムデバイスホストで有効にします。

```
[root@qdevice:~]# systemctl start pcsd.service
[root@qdevice:~]# systemctl enable pcsd.service
```

10.5.2. クォーラムデバイスの設定

本セクションでは、**Red Hat High Availability** クラスターでクォーラムデバイスを設定する手順例を説明します。以下の手順では、クォーラムデバイスを設定し、そのクォーラムデバイスをクラスターに追加します。この例の場合:

- クォーラムデバイスに使用されるノードは **qdevice** です。
- クォーラムデバイスモデルは **net** で、現在サポートされている唯一のモデルです。**net** モデルは、以下のアルゴリズムに対応します。
  - **ffsplit: fifty-fifty** 分割。これにより、アクティブなノードの数が最も多いパーティションに 1 票が提供されます。
  - **ldM: last-man-standing**。ノードが **qnetd** サーバーを表示できるクラスター内に残っている唯一のノードである場合、投票を返します。

警告

LMS アルゴリズムにより、ノードが 1 つしか残っていなくてもクラスターはクォーラムを維持できますが、number_of_nodes - 1 と同じであるため、クォーラムデバイスの投票力が大きいことを意味します。クォーラムデバイスとの接続が失われると、number_of_nodes - 1 の票が失われます。つまり、(クォーラムデバイスを無効にすることで) すべてのノードがアクティブなクラスターのみがクォーラムに達したままになります。他のクラスターは、クォーラムに達しなくなります。

これらのアルゴリズムの実装の詳細は、corosync-qdevice(8)の man ページを参照してください。

- **クラスターノードは node1 および node2 です。**

以下の手順は、クォーラムデバイスを設定し、そのクォーラムデバイスをクラスターに追加します。

1. **クォーラムデバイスをホストするために使用するノードで以下のコマンドを使用し、クォーラムデバイスを設定します。このコマンドは、クォーラムデバイスモデル net を設定および開始し、システムの起動時にデバイスが開始するように設定します。**

```
[root@qdevice:~]# pcs qdevice setup model net --enable --start
Quorum device 'net' initialized
quorum device enabled
Starting quorum device...
quorum device started
```

クォーラムデバイスの設定後、そのステータスを確認できます。corosync-qnetd デーモンが実行中で、この時点でクライアントが接続されていないことが分かります。--full コマンドオプションを指定すると詳細が出力されます。

```
[root@qdevice:~]# pcs qdevice status net --full
QNetd address:                  *:5403
TLS:                            Supported (client certificate required)
Connected clients:              0
Connected clusters:             0
Maximum send/receive size:      32768/32768 bytes
```

–

2. **以下のコマンドを実行して、firewalld で high-availability サービスを有効にして、pcsd デーモンおよび net クォーラムデバイスに必要なファイアウォールのポートを有効にします。**

```
[root@qdevice:~]# firewall-cmd --permanent --add-service=high-availability
[root@qdevice:~]# firewall-cmd --add-service=high-availability
```

3. **既存クラスター内のノードのいずれかから、クォーラムデバイスをホストしているノードで hacluster ユーザーを認証します。**

```
[root@node1:~] # pcs cluster auth qdevice
Username: hacluster
Password:
qdevice: Authorized
```

4. **クォーラムデバイスをクラスターに追加します。**

クォーラムデバイスを追加する前に、後で比較するために、クォーラムデバイスの現在の設定と状況を確認できます。このコマンドの出力は、クラスターがクォーラムデバイスを使用していないことを示しています。

```
[root@node1:~]# pcs quorum config
Options:
```

```
[root@node1:~]# pcs quorum status
Quorum information
------------------
Date:             Wed Jun 29 13:15:36 2016
Quorum provider:  corosync_votequorum
Nodes:            2
Node ID:          1
Ring ID:          1/8272
Quorate:          Yes

Votequorum information
----------------------
Expected votes:   2
Highest expected: 2
Total votes:      2
Quorum:           1
Flags:            2Node Quorate

Membership information
----------------------
    Nodeid      Votes    Qdevice Name
```

```
    1          1          NR node1 (local)
    2          1          NR node2
```

以下のコマンドは、作成しておいたクォーラムデバイスをクラスターに追加します。複数のクォーラムデバイスをクラスターで同時に使用することはできません。ただし、複数のクラスターが 1 つのクォーラムデバイスを同時に使用することはできます。上記のコマンド例は、**ffsplit** アルゴリズムを使用するようにクォーラムデバイスを設定します。クォーラムデバイスの設定オプションの詳細は、**corosync-qdevice(8)**の **man** ページを参照してください。

```
[root@node1:~]# pcs quorum device add model net host=qdevice algorithm=ffsplit
Setting up qdevice certificates on nodes...
node2: Succeeded
node1: Succeeded
Enabling corosync-qdevice...
node1: corosync-qdevice enabled
node2: corosync-qdevice enabled
Sending updated corosync.conf to nodes...
node1: Succeeded
node2: Succeeded
Corosync configuration reloaded
Starting corosync-qdevice...
node1: corosync-qdevice started
node2: corosync-qdevice started
```

5. クォーラムデバイスの設定状態をチェックします。

クラスター側から以下のコマンドを実行すると、設定の変更内容を確認できます。

pcs quorum config は、設定されたクォーラムデバイスを表示します。

```
[root@node1:~]# pcs quorum config
Options:
Device:
  Model: net
    algorithm: ffsplit
    host: qdevice
```

pcs quorum status コマンドは、クォーラムデバイスのステータスを表示し、クォーラムデバイスが使用中であることを示します。

```
[root@node1:~]# pcs quorum status
Quorum information
------------------
Date:             Wed Jun 29 13:17:02 2016
```

```
Quorum provider:  corosync_votequorum
Nodes:            2
Node ID:          1
Ring ID:          1/8272
Quorate:          Yes

Votequorum information
----------------------
Expected votes:   3
Highest expected: 3
Total votes:      3
Quorum:           2
Flags:            Quorate Qdevice

Membership information
----------------------
    Nodeid      Votes    Qdevice Name
         1          1    A,V,NMW node1 (local)
         2          1    A,V,NMW node2
         0          1            Qdevice
```

pcs quorum device status には、クォーラムデバイスのランタイムステータスが表示されます。

```
[root@node1:~]# pcs quorum device status
Qdevice information
-------------------
Model:                  Net
Node ID:                1
Configured node list:
    0   Node ID = 1
    1   Node ID = 2
Membership node list:   1, 2

Qdevice-net information
----------------------
Cluster name:           mycluster
QNetd host:             qdevice:5403
Algorithm:              ffsplit
Tie-breaker:            Node with lowest node ID
State:                  Connected
```

クォーラムデバイスから以下の **status** コマンドを実行すると、**corosync-qnetd** デーモンのステータスを表示できます。

```
[root@qdevice:~]# pcs qdevice status net --full
QNetd address:                  *:5403
TLS:                            Supported (client certificate required)
Connected clients:              2
Connected clusters:             1
Maximum send/receive size:      32768/32768 bytes
```

```
Cluster "mycluster":
  Algorithm:        ffsplit
  Tie-breaker:      Node with lowest node ID
  Node ID 2:
    Client address:       ::ffff:192.168.122.122:50028
    HB interval:          8000ms
    Configured node list: 1, 2
    Ring ID:              1.2050
    Membership node list: 1, 2
    TLS active:           Yes (client certificate verified)
    Vote:                 ACK (ACK)
  Node ID 1:
    Client address:       ::ffff:192.168.122.121:48786
    HB interval:          8000ms
    Configured node list: 1, 2
    Ring ID:              1.2050
    Membership node list: 1, 2
    TLS active:           Yes (client certificate verified)
    Vote:                 ACK (ACK)
```

10.5.3. クォーラムデバイスサービスの管理

PCS は、以下のコマンド例に示すように、ローカルホスト(corosync-qnetd)でクォーラムデバイスサービスを管理する機能を提供します。このコマンドは、corosync-qnetd サービスにのみ影響することに注意してください。

```
[root@qdevice:~]# pcs qdevice start net
[root@qdevice:~]# pcs qdevice stop net
[root@qdevice:~]# pcs qdevice enable net
[root@qdevice:~]# pcs qdevice disable net
[root@qdevice:~]# pcs qdevice kill net
```

10.5.4. クラスターでのクォーラムデバイス設定の管理

以下のセクションは、クラスターでクォーラムデバイス設定を管理するために使用する PCS コマンドについて説明し、「クォーラムデバイスの設定」 のクォーラムデバイス設定を基にした例を使用します。

10.5.4.1. クォーラムデバイス設定の変更

クォーラムデバイスの設定を変更するには、pcs quorum device update コマンドを使用します。

警告

クォーラムデバイスモデル net の ホスト オプションを変更するには、旧ホストと新しいホストが同じマシンでない限り、pcs quorum device remove コマンドおよび pcs quorum device add コマンドを使用して設定を正しく設定します。

以下のコマンドは、クォーラムデバイスアルゴリズムを lms に変更します。

```
[root@node1:~]# pcs quorum device update model algorithm=lms
Sending updated corosync.conf to nodes...
node1: Succeeded
node2: Succeeded
Corosync configuration reloaded
Reloading qdevice configuration on nodes...
node1: corosync-qdevice stopped
node2: corosync-qdevice stopped
node1: corosync-qdevice started
node2: corosync-qdevice started
```

10.5.4.2. クォーラムデバイスの削除

以下のコマンドを使用して、クラスターノードに設定されたクォーラムデバイスを削除します。

```
[root@node1:~]# pcs quorum device remove
Sending updated corosync.conf to nodes...
node1: Succeeded
node2: Succeeded
Corosync configuration reloaded
Disabling corosync-qdevice...
node1: corosync-qdevice disabled
node2: corosync-qdevice disabled
Stopping corosync-qdevice...
node1: corosync-qdevice stopped
node2: corosync-qdevice stopped
Removing qdevice certificates from nodes...
node1: Succeeded
node2: Succeeded
```

クォーラムデバイスを削除すると、クォーラムデバイスの状態を表示するときに、次のエラーメッセージが表示されます。

```
[root@node1:~]# pcs quorum device status
Error: Unable to get quorum status: corosync-qdevice-tool: Can't connect to QDevice socket (is
QDevice running?): No such file or directory
```

10.5.4.3. クォーラムデバイスの破棄

クォーラムデバイスホストのクォーラムデバイスを無効にするか停止して、設定ファイルをすべて削除するには、次のコマンドを実行します。

```
[root@qdevice:~]# pcs qdevice destroy net
Stopping quorum device...
quorum device stopped
quorum device disabled
Quorum device 'net' configuration files removed
```

第11章 PACEMAKER ルール

ルールは、設定をより動的にするのに使用できます。ルールには、(ノード属性を使用して) 時間ベースで異なる処理グループにマシンを割り当て、場所の制約の作成時にその属性を使用する方法があります。

各ルールには、日付などの様々な式だけでなく、その他のルールも含めることができます。式の結果がルールの boolean-op フィールドに基づいて組み合わされ、最終的にルールが true または false に評価されるかどうかを判断します。次の動作は、ルールが使用される状況に応じて異なります。

表11.1 ルールのプロパティー

フィールド	説明
role	リソースが指定のロールにある場合にのみ適用するルールを制限します。許可される値： Started、Slave、 および Master。注記： role="Master" が指定されたルールは、クローンインスタンスの最初の場所を判断できません。どのアクティブインスタンスが昇格されるかにのみ影響します。
score	ルールが true に評価された場合に適用するスコア。場所の制約として、ルールでの使用に制限されます。
score-attribute	ルールが true に評価される場合に検索し、スコアとして使用するノード属性。場所の制約として、ルールでの使用に制限されます。
boolean-op	複数の式オブジェクトからの結果を組み合わせる方法。使用できる値 は and または. です。デフォルト値は and です。

11.1. ノード属性の式

ノードで定義される属性に応じてリソースを制御する場合に使用されるノード属性の式です。

表11.2 式のプロパティー

フィールド	説明
attribute	テストするノード属性。
type	値をテストする方法を指定します。使用できる値は、string、整数、バージョン.デフォルト値は stringです。
operation	実行する比較動作。設定できる値は以下のとおりです。 * lt - ノード属性の値が 値よりも小さい場合は True * gt - ノード属性の値が 値よりも大きい場合は True * LTE - ノード属性の値が値以下である場合に True

フィールド	説明 * gE - ノード属性の値が値より大きい場合は True
	* eq - ノード属性の値が値と等しい場合は True * ne - ノード属性の値が値と等しくない場合は True * defined - ノードが名前付き属性を持っている場合は True * not_defined - ノードに名前付き属性がない場合は True
value	比較のためにユーザーが提示する値 (必須)

管理者が追加する属性のほかに、表11.3「組み込みノード属性」 で説明されているように、クラスターは、使用可能な各ノードに特殊な組み込みノード属性を定義します。

表11.3 組み込みノード属性

名前	説明
#uname	ノード名

名前	説明
#id	ノード ID
#kind	ノードタイプ。使用できる値は、cluster、remote、および container です。kind の値は remote です。ocf:pacemaker:remote リソースで作成された Pacemaker リモートノードの場合は、Pacemaker リモートゲストノードおよびバンドルノードの コンテナー です。
#is_dc	このノードが指定コントローラー(DC)の場合は true、そうでない場合は false
#cluster_name	cluster -name クラスター プロパティーの値（設定されている場合）。
#site_name	site-name ノード属性の値（設定されている場合）。それ以外は #cluster-nameと同じ。
#role	関連する多状態リソースがこのノードで果たすロール。多状態リソースの場所の制約のルール内でのみ有効です。

11.2. 時刻と日付ベースの式

日付の式は、現在の日付と時刻に応じてリソースまたはクラスターオプションを制御する場合に使用します。オプションで日付の詳細を含めることができます。

表11.4 日付の式のプロパティー

フィールド	説明
start	ISO 8601 仕様に準じた日付と時刻。
end	ISO 8601 仕様に準じた日付と時刻。

フィールド	説明
operation	状況に応じて、現在の日付と時刻を start と end のいずれかの日付、または両方の日付と比較します。設定できる値は以下のとおりです。 * gt - 現在の日時が 開始後であれば True * lt - 現在の日時が 終了している場合は True * in-range - 現在の日付/時刻が 開始 後および 終了後である場合は True * date-spec - 現在の日付/時刻に対して cron のような比較を実行します。

11.3. 日付の詳細

日付の詳細は、時間に関係する cron のような式を作成するのに使用されます。各フィールドには 1 つの数字または範囲が含まれます。指定のないフィールドは、デフォルトを 0 に設定するのではなく、無視されます。

たとえば、monthdays="1 " は毎月の最初の日と一致し、hour ="09-17" は 9 am から 5 pm (inclusive)までの時間に一致します。ただし、weekdays="1,2 " または weekdays="1-2, 5-6" は、複数の範囲 が含まれるため、指定できません。

表11.5 日付詳細のプロパティー

フィールド	説明
id	日付の一意の名前
hours	使用できる値 - 0~23
monthdays	使用できる値 - 0~31 (月と年に応じて異なる)
weekdays	使用できる値 - 1~7 (1=月曜日、7=日曜日)
yeardays	使用できる値 - 1~366 (年に応じて異なる)
months	使用できる値 - 1~12
weeks	許可される値：1-53 (週性により異なる)

フィールド	説明
years	グレゴリオ暦 (新暦) に準じる年
weekyears	Gregorian 年と異なる場合があります。たとえば、2005-001 Ordinal も 2005-01-01 Gregorian も 2004-W53-6 Weekly です。
moon	使用できる値 - 0~7 (0 は新月、4 は満月)

11.4. 期間

duration は、in_range 操作に指定されていない場合に end の値を計算するために使用されます。これには date_spec オブジェクトと同じフィールドが含まれていますが、制限はありません（つまり、19 カ月間を指定できます）。date_specs と同様に、指定されていないフィールドは無視されます。

11.5. PCS でのルールの設定

pcs を使用してルールを設定するには、「ルールを使用したリソースの場所の確定」 の説明に従って、ルールを使用する場所の制約を設定します。

ルールを削除する場合は次のコマンドを使用します。削除しているルールがその制約内で最後のルールになる場合は、その制約も削除されます。

```
pcs constraint rule remove rule_id
```

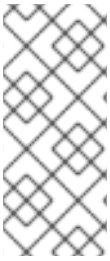
第12章 PACEMAKER クラスターのプロパティー

クラスターのプロパティーは、クラスター動作中に発生する可能性がある状況に直面した場合に、クラスターの動作を制御します。

- 表12.1「クラスターのプロパティー」 では、クラスターのプロパティーオプションを説明します。
- 「クラスターのプロパティーの設定と削除」 では、クラスタープロパティーの設定方法を説明します。
- 「クラスタープロパティー設定のクエリー」 では、現在設定されているクラスタープロパティーを一覧表示する方法を説明しています。

12.1. クラスタープロパティーとオプションの要約

表12.1「クラスターのプロパティー」 には、Pacemaker クラスターのプロパティーのデフォルト値や、設定可能な値などをまとめています。



注記

この表に記載しているプロパティー以外にも、クラスターソフトウェアで公開されるクラスタープロパティーがあります。このようなプロパティーでは、デフォルト値を別の値には変更しないことが推奨されます。

表12.1 クラスターのプロパティー

オプション	デフォルト	説明
batch-limit	0	クラスターを並列に実行できるリソースアクションの数。正しい値は、ネットワークおよびクラスターノードの速度と負荷によって異なります。
migration-limit	-1 (無制限)	クラスターが、ノードで並行に実行することが許可されている移行ジョブの数。

オプション	デフォルト	説明
no-quorum-policy	stop	クラスターにクォーラムがない場合のアクション。設定できる値は以下のとおりです。 * ignore - 全リソースの管理を続行する * freeze - リソース管理は継続するが、影響を受けるパーティションに含まれないノードのリソースは復帰させない * stop - 影響を受けるクラスターパーティション内の全リソースを停止する * suicide - 影響を受けるクラスターパーティション内の全ノードをフェンスする
symmetric-cluster	true	リソースを、デフォルトで任意のノードで実行できるかどうかを示します。
stonith-enabled	true	障害が発生したノードと、停止できないリソースが含まれるノードをフェンスする必要があることを示します。データを保護するには、**true** に設定する必要があります。 **true**、または未設定の場合は、1つ以上の STONITH リソースが設定されていない限り、リソースの開始を拒否します。
stonith-action	reboot	STONITH デバイスに送るアクション。許可される値：**reboot**、**off**。**poweroff** 値も使用できますが、レガシーデバイスにのみ使用されます。
cluster-delay	60s	(アクションの実行を除く) ネットワーク上のラウンドトリップ遅延です。正しい値は、ネットワークおよびクラスターノードの速度と負荷によって異なります。
stop-orphan-resources	true	削除されたリソースを停止すべきかどうかを示します。
stop-orphan-actions	true	削除されたアクションをキャンセルするかどうかを示します。

オプション	デフォルト	説明
start-failure-is-fatal	true	特定のノードでリソースの起動に失敗した場合に、そのノードで開始試行を行わないようにするかを示します。**false** に設定すると、クラスターは、リソースの現在の失敗数と移行しきい値に基づいて、同じノードで開始を再試行するかどうかを決定します。リソースの **migration-threshold** オプションの設定に関する詳細は、「障害発生によるリソースの移動」 を参照してください。 **start-failure-is-fatal** を **false** に設定すると、リソースを起動できない障害のあるノードが、すべての依存アクションを保持できるというリスクが発生します。これが、**start-failure-is-fatal** のデフォルトは **true** であるためです。**start-failure-is-fatal=false** を設定するリスクは、移行しきい値を低く設定すると軽減できます。これにより、多くの障害の後に他のアクションを続行できます。
pe-error-series-max	-1 (すべて)	ERROR となる PE 入力を保存する数。問題を報告する場合に使用されます。
pe-warn-series-max	-1 (すべて)	WARNING となる PE 入力を保存する数。問題を報告する場合に使用されます。
pe-input-series-max	-1 (すべて)	normal となる PE 入力を保存する数。問題を報告する場合に使用されます。
cluster-infrastructure		Pacemaker が現在実行しているメッセージングスタック。情報提供および診断目的に使用されます。ユーザーは設定できません。
dc-version		クラスターの DC (Designated Controller) で Pacemaker のバージョン。診断目的に使用され、ユーザーは設定できません。
last-lrm-refresh		epoca 以降の秒単位で指定されたローカルリソースマネージャーの最終更新です。診断目的に使用され、ユーザーは設定できません。
cluster-recheck-interval	15 分	時間ベースでオプション、リソースパラメーター、および制約を変更するポーリング間隔。使用できる値は、ポーリングを無効にする 0 (ゼロ) と、秒単位で間隔を示す正の値です (5min など、他の SI 単位が指定されていない場合に限ります)。この値は確認が行われる間隔に対する最大時間であることに注意してください。クラスターイベントが、この値で指定された時間よりも早く発生した場合は、確認が行われるタイミングが早くなります。
maintenance-mode	false	メンテナンスモードでは、クラスターが干渉されないモードになり、指示されない限り、サービスを起動したり、停止したりしません。メンテナンスモードが完了すると、クラスターは、サービスの現在の状態のサニティーチェックを実行してから、これを必要とするサービスを停止するか、開始します。

オプション	デフォルト	説明
shutdown-escalation	20min	正常にシャットダウンして終了を試みるのをやめる時間。高度な使用のみ。
stonith-timeout	60s	STONITH アクションが完了するのを待つ時間。
stop-all-resources	false	クラスターがすべてのリソースを停止します。
enable-acl	false	(Red Hat Enterprise Linux 7.1 以降) **pcs acl** コマンドで設定したように、クラスターがアクセス制御リストを使用できるかどうかを示します。
placement-strategy	**default**	クラスターノードでリソースの配置を決定する際に、クラスターが使用率属性を考慮にいれるかどうかと、どのように考慮するかを示します。使用率属性および配置ストラテジーの詳細は、「使用と配置ストラテジー」 を参照してください。
fence-reaction	**stop**	(Red Hat Enterprise Linux 7.8 以降) 独自のフェンシングの通知を受信した場合は、クラスターノードがどのように反応するかを決定します。クラスターノードは、フェンシングの設定が間違っている場合に独自のフェンシングの通知を受信するか、またはファブリックフェンシングがクラスター通信を遮断しない状態である可能性があります。許可される値は、Pacemaker をすぐに **停止** して停止したままにするのを停止するか、ローカルノードを直ちに再起動して失敗した場合に停止する **panic** です。

12.2. クラスターのプロパティーの設定と削除

クラスタープロパティーの値を設定するには、次の pcs コマンドを使用します。

```
pcs property set property=value
```

たとえば、symmetric-cluster の値を false に設定するには、次のコマンドを使用します。

```
# pcs property set symmetric-cluster=false
```

設定からクラスタープロパティーを削除する場合は、次のコマンドを使用します。

```
pcs property unset property
```

代わりに pcs property set コマンドの値フィールドを空白のままにして、クラスタープロパティーを設定から削除できます。これにより、そのプロパティーの値がデフォルト値に戻されます。たとえ

ば、symmetric-cluster プロパティーを false に設定している場合、以下のコマンドは設定から設定した値を削除し、symmetric-cluster の値をデフォルト値の true に戻します。

```
# pcs property set symmetic-cluster=
```

12.3. クラスタープロパティー設定のクエリー

ほとんどの場合、pcs コマンドを使用してさまざまなクラスターコンポーネントの値を表示する場合は、pcs list または pcs show を相互に置き換え可能なものにすることができます。次の例では、pcs list は、複数のプロパティーのすべての設定全体を表示するために使用される形式です。一方、pcs show は、特定のプロパティーの値を表示するために使用される形式です。

クラスターに設定されたプロパティー設定の値を表示する場合は、次の pcs コマンドを使用します。

```
pcs property list
```

明示的に設定されていないプロパティー設定のデフォルト値など、クラスターのプロパティー設定の値をすべて表示する場合は、次のコマンドを使用します。

```
pcs property list --all
```

特定のクラスタープロパティーの現在の値を表示する場合は、次のコマンドを使用します。

```
pcs property show property
```

たとえば、cluster-infrastructure プロパティーの現在の値を表示するには、以下のコマンドを実行します。

```
# pcs property show cluster-infrastructure
Cluster Properties:
 cluster-infrastructure: cman
```

情報提供の目的で、次のコマンドを使用して、プロパティーがデフォルト以外の値に設定されているかどうかに関わらず、プロパティーのデフォルト値のリストを表示できます。

```
pcs property [list|show] --defaults
```

第13章 クラスターイベントのスクリプトのトリガー

Pacemaker クラスターはイベント駆動型のシステムで、イベントはリソースやノードの障害、設定の変更、またはリソースの開始や停止になります。Pacemaker クラスターアラートを設定すると、クラスターイベントの発生時に外部で一部の処理を行うことができます。クラスターアラートを設定するには、以下の 2 つの方法の 1 つを使用します。

- Red Hat Enterprise Linux 7.3 より、アラートエージェントを使用して Pacemaker アラートを設定できるようになりました。アラートエージェントは、リソース設定と操作を処理するためにクラスター呼び出しのリソースエージェントと同様にクラスターが呼び出す外部プログラムです。Pacemaker アラートエージェントの説明は 「Pacemaker アラートエージェント (Red Hat Enterprise Linux 7.3 以降)」 を参照してください。

- ocf:pacemaker:ClusterMon リソースはクラスターのステータスを監視し、各クラスターイベントでアラートをトリガーできます。このリソースは、crm_mon コマンドを一定間隔でバックグラウンドで実行します。ClusterMon リソースの詳細は 「モニタリングのリソースを使ったイベント通知」 を参照してください。

13.1. PACEMAKER アラートエージェント (RED HAT ENTERPRISE LINUX 7.3 以降)

クラスターイベントの発生時に Pacemaker アラートエージェントを作成して外部で一部の処理を行うことができます。クラスターは、環境変数を用いてイベントの情報をエージェントに渡します。エージェントは、E メールメッセージの送信、ログのファイルへの記録、監視システムの更新など、この情報を自由に使用できます。

- Pacemaker は、デフォルトで /usr/share/pacemaker/alerts にインストールされるアラートエージェントのサンプルを複数提供します。これらのサンプルスクリプトは、コピーしてそのまま使用したり、目的に合わせて編集するテンプレートとして使用することもできます。対応する全属性は、サンプルエージェントのソースコードを参照してください。サンプルアラートエージェントを使用するアラートの基本的な設定手順の例は、「サンプルアラートエージェントの使用」 を参照してください。

- アラートエージェントの設定および管理に関する一般的な情報は、「アラートの作成」、「アラートの表示、編集、および削除」、「アラートの受信側」、「アラートメタオプション」、および 「アラート設定コマンドの例」 を参照してください。

- Pacemaker アラートの独自のアラートエージェントを作成することができます。アラートエージェントの作成に関する詳細は、「アラートエージェントの作成」 を参照してください。

13.1.1. サンプルアラートエージェントの使用

サンプルアラートエージェントの 1 つを使用するとき、スクリプトがニーズにあっていることを確認してください。サンプルエージェントは、特定のクラスター環境用のカスタムスクリプトを作成するためのテンプレートとして提供されます。Red Hat は、Pacemaker との通信にアラートエージェントスクリプトが使用するインターフェイスをサポートしますが、カスタムエージェント自体にはサポートを提供していないことに注意してください。

サンプルアラートエージェントの 1 つ を使用するには、クラスターの各ノードにエージェントをインストールする必要があります。たとえば、以下のコマンドは、alert_file.sh.sample スクリプトを alert_file.sh としてインストールします。

```
# install --mode=0755 /usr/share/pacemaker/alerts/alert_file.sh.sample
/var/lib/pacemaker/alert_file.sh
```

スクリプトをインストールしたら、スクリプトを使用するアラートを作成できます。

以下の例では、インストールされた alert_file.sh アラートエージェントを使用してイベントをファイルに記録するアラートを設定します。アラートエージェントは、最低限のパーミッションを持つ hacluster ユーザーとして実行します。

この例では、イベントの記録に使用するログファイル pcmk_alert_file.log を作成します。また、アラートエージェントを作成し、その受信先としてログファイルへのパスを追加します。

```
# touch /var/log/pcmk_alert_file.log
# chown hacluster:haclient /var/log/pcmk_alert_file.log
# chmod 600 /var/log/pcmk_alert_file.log
# pcs alert create id=alert_file description="Log events to a file." path=/var/lib/pacemaker/alert_file.sh
# pcs alert recipient add alert_file id=my-alert_logfile value=/var/log/pcmk_alert_file.log
```

以下の例では、alert_snmp.sh.sample スクリプトを alert_snmp.sh としてインストールし、インストールされた alert_snmp.sh アラートエージェントを使用してクラスターイベントを SNMP トラップとして送信するアラートを設定します。デフォルトでは、正常な監視呼び出し以外のすべてのイベントを SNMP サーバーに送信します。この例では、タイムスタンプの形式をメタオプションとして設定します。メタオプションの詳細は、「アラートメタオプション」 を参照してください。この例では、アラートの設定後にアラートの受信側が設定され、アラート設定が表示されます。

```
# install --mode=0755 /usr/share/pacemaker/alerts/alert_snmp.sh.sample
/var/lib/pacemaker/alert_snmp.sh
# pcs alert create id=snmp_alert path=/var/lib/pacemaker/alert_snmp.sh meta timestamp-
format="%Y-%m-%d,%H:%M:%S.%01N"
# pcs alert recipient add snmp_alert value=192.168.1.2
# pcs alert
Alerts:
 Alert: snmp_alert (path=/var/lib/pacemaker/alert_snmp.sh)
```

```
 Meta options: timestamp-format=%Y-%m-%d,%H:%M:%S.%01N.
 Recipients:
  Recipient: snmp_alert-recipient (value=192.168.1.2)
```

次の例では、**alert_smtp.sh** エージェントをインストールし、インストールされたアラートエージェントを使用するアラートを設定して、クラスターイベントを電子メールメッセージとして送信します。この例では、アラートの設定後に受信側が設定され、アラート設定が表示されます。

```
# install --mode=0755 /usr/share/pacemaker/alerts/alert_smtp.sh.sample
/var/lib/pacemaker/alert_smtp.sh
# pcs alert create id=smtp_alert path=/var/lib/pacemaker/alert_smtp.sh options
email_sender=donotreply@example.com
# pcs alert recipient add smtp_alert value=admin@example.com
# pcs alert
Alerts:
 Alert: smtp_alert (path=/var/lib/pacemaker/alert_smtp.sh)
  Options: email_sender=donotreply@example.com
  Recipients:
   Recipient: smtp_alert-recipient (value=admin@example.com)
```

pcs alert create および **pcs alert recipient add** コマンドの形式に関する詳細は、「アラートの作成」 および 「アラートの受信側」 を参照してください。

13.1.2. アラートの作成

次のコマンドは、クラスターアラートを作成します。設定するオプションは、追加の環境変数として指定するパスで、アラートエージェントスクリプトに渡されるエージェント固有の設定値です。**id** の値を指定しないと、値が生成されます。アラートメタオプションの詳細は、「アラートメタオプション」 を参照してください。

```
pcs alert create path=path [id=alert-id] [description=description] [options [option=value]...] [meta
[meta-option=value]...]
```

複数のアラートエージェントを設定できます。クラスターは、各イベントに対して、すべてのアラートエージェントを呼び出します。アラートエージェントはクラスターノードでのみ呼び出されます。アラートエージェントは、**Pacemaker** リモートノードが関係するイベントに対して呼び出されますが、このようなノードでは呼び出されません。

以下の例は、各イベントに対して **myscript.sh** を呼び出す簡単なアラートを作成します。

```
# pcs alert create id=my_alert path=/path/to/myscript.sh
```

サンプルアラートエージェントの 1 つを使用するクラスターアラートの作成方法の例は、「サンプ

ルアラートエージェントの使用」 を参照してください。

13.1.3. アラートの表示、編集、および削除

次のコマンドは、設定されたすべてのアラートと、設定されたオプションの値を表示します。

```
pcs alert [config|show]
```

以下のコマンドは、指定した ***alert-id*** 値を持つ既存のアラートを更新します。

```
pcs alert update alert-id [path=path] [description=description] [options [option=value]...] [meta [meta-option=value]...]
```

以下のコマンドは、指定の ***alert-id*** 値を持つアラートを削除します。

```
pcs alert remove alert-id
```

代わりに **pcs alert delete** コマンドを実行できます。これは **pcs alert remove** コマンドと同じです。**pcs alert delete** コマンドおよび **pcs alert remove** コマンドの両方を使用すると、複数のアラートを削除できます。

13.1.4. アラートの受信側

通常、アラートは受信側に送信されます。したがって、各アラートには、1 人以上の受信者を追加で設定できます。クラスターは、受信側ごとに別々にエージェントを呼び出します。

受信側は、IP アドレス、メールアドレス、ファイル名、特定のエージェントがサポートするものなど、アラートエージェントが認識できるものを設定します。

次のコマンドは、新しい受信側を指定のアラートに追加します。

```
pcs alert recipient add alert-id value=recipient-value [id=recipient-id] [description=description] [options [option=value]...] [meta [meta-option=value]...]
```

次のコマンドは、既存のアラート受信側を更新します。

```
pcs alert recipient update recipient-id [value=recipient-value] [description=description] [options
[option=value]...] [meta [meta-option=value]...]
```

次のコマンドは、指定のアラート受信側を削除します。

```
pcs alert recipient remove recipient-id
```

代わりに **pcs alert recipient delete** コマンドを実行できます。これは **pcs alert recipient remove** コマンドと同じです。**pcs alert recipient remove** コマンドおよび **pcs alert recipient delete** コマンドの両方を使用すると、複数のアラート受信者を削除できます。

以下のコマンド例は、受信側 ID が **my-recipient-id** のアラート受信側 **my-alert-recipient** をアラート **my-alert** に追加します。これにより、各イベントの **my-alert** 用に設定されたアラートスクリプトを呼び出すようにクラスターが設定され、受信者 **some-address** が環境変数として渡されます。

```
#  pcs alert recipient add my-alert value=my-alert-recipient id=my-recipient-id options value=some-
address
```

13.1.5. アラートメタオプション

リソースエージェントと同様に、メタオプションをアラートエージェントに対して設定すると、**Pacemaker** の呼び出し方法を調整できます。表13.1「アラートメタオプション」 は、アラートメタオプションを示しています。メタオプションは、アラートエージェントごと、または受信側ごとに設定できます。

表13.1 アラートメタオプション

メタ属性	デフォルト	説明
timestamp-format	%H:%M:%S.%06N	イベントのタイムスタンプをエージェントに送信するときにクラスターが使用する形式です。**date**(1)コマンドで使用される文字列です。
timeout	30s	アラートエージェントがこの時間内に完了しないと終了させられます。

以下の例では、**myscript.sh** スクリプトを呼び出すアラートを設定し、2 つの受信側をアラートに追加します。最初の受信側には、ID が **my-alert-recipient1** で、2 番目の受信側の ID は **my-alert-recipient2** になります。スクリプトは各イベントで 2 回呼び出され、呼び出しのタイムアウト値はそれぞれ 15 秒です。1 つの呼び出しは受信者 **someuser@example.com** に渡され、タイムスタンプの形式は **%D %H:%M** になります。もう 1 つの呼び出しは受信者 **otheruser@example.com** に渡され、タイムスタンプの形式は **%c** になります。

```
# pcs alert create id=my-alert path=/path/to/myscript.sh meta timeout=15s
# pcs alert recipient add my-alert value=someuser@example.com id=my-alert-recipient1 meta
timestamp-format="%D %H:%M"
# pcs alert recipient add my-alert value=otheruser@example.com id=my-alert-recipient2 meta
timestamp-format=%c
```

13.1.6. アラート設定コマンドの例

以下の例は、基本的なアラート設定コマンドの一部と、アラートの作成、受信側の追加、および設定されたアラートの表示に使用される形式を表しています。アラートエージェント自体はクラスター内の各ノードにインストールする必要がありますが、**pcs** コマンドの実行は 1 回だけで済みます。

以下のコマンドは簡単なアラートを作成し、アラートに 2 つの受信側を追加した後、設定された値を表示します。

- アラート ID の値が指定されていないため、**alert** のアラート ID を作成します。
- 最初の受信者の作成コマンドは、**rec_value** の受信者を指定します。このコマンドには受信側 ID が指定されていないため、**alert-recipient** の値が受信側 ID として使用されます。
- 2 番目の受信者の作成コマンドは、**rec_value2** の受信者を指定します。このコマンドは、宛先に **my-recipient** の受信者 ID を指定します。

```
# pcs alert create path=/my/path
# pcs alert recipient add alert value=rec_value
# pcs alert recipient add alert value=rec_value2 id=my-recipient
# pcs alert config
Alerts:
 Alert: alert (path=/my/path)
  Recipients:
   Recipient: alert-recipient (value=rec_value)
   Recipient: my-recipient (value=rec_value2)
```

以下のコマンドは、2 番目のアラートとそのアラートの受信側を追加します。2 番目のアラートのアラート ID は **my-alert** で、受信側の値は **my-other-recipient** です。受信側 ID が指定されていないため、受信側 ID は **my-alert-recipient** となっています。

```
# pcs alert create id=my-alert path=/path/to/script description=alert_description options
option1=value1 opt=val meta timeout=50s timestamp-format="%H%B%S"
# pcs alert recipient add my-alert value=my-other-recipient
# pcs alert
Alerts:
 Alert: alert (path=/my/path)
```

```
 Recipients:
  Recipient: alert-recipient (value=rec_value)
  Recipient: my-recipient (value=rec_value2)
Alert: my-alert (path=/path/to/script)
 Description: alert_description
 Options: opt=val option1=value1
 Meta options: timestamp-format=%H%B%S timeout=50s
 Recipients:
  Recipient: my-alert-recipient (value=my-other-recipient)
```

以下のコマンドは、my-alert、受信側 my-alert -recipient の アラート 値を変更します。

```
# pcs alert update my-alert options option1=newvalue1 meta timestamp-format="%H%M%S"
# pcs alert recipient update my-alert-recipient options option1=new meta timeout=60s
# pcs alert
Alerts:
 Alert: alert (path=/my/path)
  Recipients:
   Recipient: alert-recipient (value=rec_value)
   Recipient: my-recipient (value=rec_value2)
 Alert: my-alert (path=/path/to/script)
  Description: alert_description
  Options: opt=val option1=newvalue1
  Meta options: timestamp-format=%H%M%S timeout=50s
  Recipients:
   Recipient: my-alert-recipient (value=my-other-recipient)
    Options: option1=new
    Meta options: timeout=60s
```

以下のコマンドは、アラートから受信側 my- alert -recipient を削除します。

```
# pcs alert recipient remove my-recipient
# pcs alert
Alerts:
 Alert: alert (path=/my/path)
  Recipients:
   Recipient: alert-recipient (value=rec_value)
 Alert: my-alert (path=/path/to/script)
  Description: alert_description
  Meta options: timestamp-format="%M%B%S" timeout=50s
  Meta options: m=newval meta-option1=2
  Recipients:
   Recipient: my-alert-recipient (value=my-other-recipient)
    Options: option1=new
    Meta options: timeout=60s
```

次のコマンドは、設定から myalert を削除します。

```
# pcs alert remove my-alert
```

```
# pcs alert
Alerts:
 Alert: alert (path=/my/path)
  Recipients:
   Recipient: alert-recipient (value=rec_value)
```

13.1.7. アラートエージェントの作成

Pacemaker アラートには、ノードアラート、フェンスアラート、およびリソースアラートの 3 種類があります。アラートエージェントに渡された環境変数は、アラートのタイプにより異なる可能性があります。表13.2「アラートエージェントに渡される環境変数」 は、アラートエージェントに渡される環境変数を示し、環境変数が特定のアラートタイプに関連付けられるタイミングを指定します。

表13.2 アラートエージェントに渡される環境変数

環境変数	説明
CRM_alert_kind	アラートの種類 (ノード、フェンス、またはリソース)
CRM_alert_version	アラートを送信する Pacemaker のバージョン
CRM_alert_recipient	設定された送信側
CRM_alert_node_sequence	アラートがローカルノードで発行されるたびに増加するシーケンス番号。これは、Pacemaker によりアラートが発行された順序を参照するのに使用できます。後で発生したイベントのアラートは、先に発生したイベントのアラートよりもシーケンス番号が大きくなります。この番号は、クラスター全体を対象とする番号ではないことに注意してください。
CRM_alert_timestamp	**timestamp-format** メタオプションで指定された形式で、エージェントの実行前に作成されたタイムスタンプ。これにより、エージェントは、エージェント自体が呼び出されたタイミング (システムの負荷やその他の状況により遅延する可能性があります) に関係なく、信頼できる高精度のイベント発生時間を使用できます。
CRM_alert_node	影響を受けるノードの名前
CRM_alert_desc	イベントの詳細。ノードアラートの場合は、ノードの現在の状態 (番号または lost) になります。フェンスアラートの場合は、フェンス操作の要求元、ターゲット、フェンス操作のエラーコードなどを含む要求されたフェンス操作の概要になります。リソースアラートの場合、これは **CRM_alert_status** と同等の文字列です。
CRM_alert_nodeid	状態が変更したノードの ID (ノードアラートの場合のみ提供)。
CRM_alert_task	要求されたフェンスまたはリソース操作 (フェンスおよびリソースアラートの場合のみ提供)。
CRM_alert_rc	フェンスまたはリソース操作の数値の戻りコード (フェンスおよびリソースアラートの場合のみ提供)。

環境変数	説明
CRM_alert_rsc	影響を受けるリソースの名前 (リソースアラートのみ)。
CRM_alert_interval	リソース操作の間隔 (リソースアラートのみ)
CRM_alert_target_rc	操作の予期される数値の戻りコード (リソースアラートのみ)。
CRM_alert_status	Pacemaker が、操作の結果を示すために使用する数値コード (リソースアラートのみ)。

アラートエージェントを記述する際は、以下を考慮する必要があります。

- **アラートエージェントは受信者なしで呼び出されることがあります (受信者が設定されていない場合)。したがって、エージェントは、このような状況では終了しかしない場合でも、この状態に対応できなければなりません。設定を段階的に変更し、後で受信側を追加することもできます。**

- **1 つのアラートに複数の受信側が設定されると、アラートエージェントは受信側ごとに 1 回呼び出されます。エージェントが同時に実行できない場合は、受信側を 1 つのみ設定する必要があります。エージェントは、受信側をリストとして解釈することができます。**

- **クラスターイベントの発生時、すべてのアラートは別々のプロセスとして同時に発生します。設定されているアラートと受信者の数、およびアラートエージェント内で行われている内容に応じて、負荷が急激に増加する可能性があります。たとえば、リソースを大量に消費するアクションを直接実行するのではなく、別のインスタンスのキューに追加することで、これを考慮に入れるようにエージェントを作成できます。**

- **アラートエージェントは、最低限のパーミッションを持つ hacluster ユーザーとして実行されます。エージェントに追加の特権が必要な場合は、適切な特権を持つ別のユーザーとしてエージェントが必要なコマンドを実行できるように、sudo を設定することが推奨されます。**

- **CRM_alert_timestamp（それらのコンテンツは、ユーザーが設定した timestamp-formatで指定される）、CRM_alert_recipient、およびすべてのアラートオプションなど、ユーザーが設定したパラメーターを検証およびサニタイズすることに注意してください。これは、設定エラーから保護するために必要です。さらに、一部のユーザーがクラスターノードへの hacluster-level アクセスがなくても CIB を変更できる場合は、セキュリティー上の問題となる可能性があるため、コードの挿入の可能性を回避する必要があります。**

- **onfail パラメーターが fence に設定されている操作を持つリソースがクラスターに含まれる場合、障害発生時に複数の フェンス 通知が発生します（このパラメーターが設定されている**

リソースごとに 1 つと、追加の通知 1 つが含まれます）。STONITH デーモンと crmd デーモンの両方が通知を送信します。この場合、送信される通知の数に関係なく、Pacemaker は 1 つのフェンス操作のみを実際に実行します。

注記

アラートインターフェイスは、ocf:pacemaker:ClusterMon リソースで使用される外部スクリプトインターフェイスと後方互換性を持つように設計されています。この互換性を維持するために、アラートエージェントに渡される環境変数に CRM_notify_ および CRM_alert_ が追加されます。互換性違反の 1 つは、ClusterMon リソースが root ユーザーとして外部スクリプトを実行し、アラートエージェントは hacluster ユーザーとして実行されることです。ClusterMon によってトリガーされるスクリプトの設定については、「モニタリングのリソースを使ったイベント通知」 を参照してください。

13.2. モニタリングのリソースを使ったイベント通知

ocf:pacemaker:ClusterMon リソースはクラスターのステータスを監視し、各クラスターイベントでアラートをトリガーできます。このリソースは、crm_mon コマンドを一定間隔でバックグラウンドで実行します。

デフォルトでは、crm_mon コマンドはリソースイベントのみをリッスンします。フェンシングイベントのリストを有効にするには、ClusterMon リソースの設定時に --watch-fencing オプションをコマンドに指定します。crm_mon コマンドはメンバーシップの問題を監視しませんが、フェンシングが開始され、そのノードに対して監視が開始されたときにメッセージが出力されます。これはメンバーがクラスターに参加したことを意味します。

ClusterMon リソース は外部プログラムを実行して、extra_options パラメーターを使用してクラスター通知を行うものを判別できます。表13.3「外部監視プログラムへ渡される環境変数」 は、発生したクラスターイベントのタイプを記述する、プログラムに渡される環境変数を一覧表示します。

表13.3 外部監視プログラムへ渡される環境変数

環境変数	説明
CRM_notify_recipient	リソース定義からの静的な外部受信側。
CRM_notify_node	状態が変更したノード。
CRM_notify_rsc	状態を変更したリソースの名前。
CRM_notify_task	状態が変更する原因となった操作。
CRM_notify_desc	状態が変更する原因となった操作 (該当の操作がある場合) のテキスト出力の関連エラーコード。

環境変数	説明
CRM_notify_rc	操作の戻りコード。
CRM_target_rc	操作の予期される戻りコード。
CRM_notify_status	操作の状態の数値表現。

以下の例は、外部プログラム crm_logger.sh を実行する ClusterMon リソースを設定します。このプログラムは指定されたイベント通知をログに記録します。

以下の手順は、このリソースが使用する crm_logger.sh プログラムを作成します。

1. **クラスターのノードの 1 つで、イベント通知をログに記録するプログラムを作成します。**

```
# cat <<-END >/usr/local/bin/crm_logger.sh
#!/bin/sh
logger -t "ClusterMon-External" "${CRM_notify_node} ${CRM_notify_rsc} \
${CRM_notify_task} ${CRM_notify_desc} ${CRM_notify_rc} \
${CRM_notify_target_rc} ${CRM_notify_status} ${CRM_notify_recipient}";
exit;
END
```

2. **プログラムの所有者とパーミッションを設定します。**

```
# chmod 700 /usr/local/bin/crm_logger.sh
# chown root.root /usr/local/bin/crm_logger.sh
```

3. **scp コマンドを使用して crm_logger.sh プログラムをクラスターの他のノードにコピーし、それらのノードの同じ場所にプログラムを配置し、プログラムに同じ所有者とパーミッションを設定します。**

以下の例は、プログラム /usr/local/bin/crm_logger.sh を実行する ClusterMon -External という名前の ClusterMon リソースを設定します。ClusterMon リソースは、クラスターのステータスを html ファイル（この例では /var/www/html/cluster_mon.html ）に出力します。pidfile は ClusterMon がすでに実行されているかどうかを検出します（この例では /var/run/crm_mon-external.pid ファイル）。このリソースはクローンとして作成されるため、クラスターの各ノードで実行されます。watch-fencing は、フェンスリソースの start/stop/monitor、start/monitor など、リソースイベントなどのフェンスイベントの監視を有効にするために指定されます。

```
# pcs resource create ClusterMon-External ClusterMon user=root \
```

```
update=10 extra_options="-E /usr/local/bin/crm_logger.sh --watch-fencing" \
htmlfile=/var/www/html/cluster_mon.html \
pidfile=/var/run/crm_mon-external.pid clone
```

注記

このリソースが実行され、手動で実行できる crm_mon コマンドは次のとおりです。

```
# /usr/sbin/crm_mon -p /var/run/crm_mon-manual.pid -d -i 5 \
-h /var/www/html/crm_mon-manual.html -E "/usr/local/bin/crm_logger.sh" \
--watch-fencing
```

以下の例は、この例によって生成される監視通知の出力形式になります。

```
Aug  7 11:31:32 rh6node1pcmk ClusterMon-External: rh6node2pcmk.examplerh.com ClusterIP
st_notify_fence Operation st_notify_fence requested by rh6node1pcmk.examplerh.com for peer
rh6node2pcmk.examplerh.com: OK (ref=b206b618-e532-42a5-92eb-44d363ac848e) 0 0 0 #177
Aug  7 11:31:32 rh6node1pcmk ClusterMon-External: rh6node1pcmk.examplerh.com ClusterIP start
OK 0 0 0
Aug  7 11:31:32 rh6node1pcmk ClusterMon-External: rh6node1pcmk.examplerh.com ClusterIP
monitor OK 0 0 0
Aug  7 11:33:59 rh6node1pcmk ClusterMon-External: rh6node1pcmk.examplerh.com fence_xvms
monitor OK 0 0 0
Aug  7 11:33:59 rh6node1pcmk ClusterMon-External: rh6node1pcmk.examplerh.com ClusterIP
monitor OK 0 0 0
Aug  7 11:33:59 rh6node1pcmk ClusterMon-External: rh6node1pcmk.examplerh.com ClusterMon-
External start OK 0 0 0
Aug  7 11:33:59 rh6node1pcmk ClusterMon-External: rh6node1pcmk.examplerh.com fence_xvms
start OK 0 0 0
Aug  7 11:33:59 rh6node1pcmk ClusterMon-External: rh6node1pcmk.examplerh.com ClusterIP start
OK 0 0 0
Aug  7 11:33:59 rh6node1pcmk ClusterMon-External: rh6node1pcmk.examplerh.com ClusterMon-
External monitor OK 0 0 0
Aug  7 11:34:00 rh6node1pcmk crmd[2887]:   notice: te_rsc_command: Initiating action 8: monitor
ClusterMon-External:1_monitor_0 on rh6node2pcmk.examplerh.com
Aug  7 11:34:00 rh6node1pcmk crmd[2887]:   notice: te_rsc_command: Initiating action 16: start
ClusterMon-External:1_start_0 on rh6node2pcmk.examplerh.com
Aug  7 11:34:00 rh6node1pcmk ClusterMon-External: rh6node1pcmk.examplerh.com ClusterIP stop
OK 0 0 0
Aug  7 11:34:00 rh6node1pcmk crmd[2887]:   notice: te_rsc_command: Initiating action 15: monitor
ClusterMon-External_monitor_10000 on rh6node2pcmk.examplerh.com
Aug  7 11:34:00 rh6node1pcmk ClusterMon-External: rh6node2pcmk.examplerh.com ClusterMon-
External start OK 0 0 0
Aug  7 11:34:00 rh6node1pcmk ClusterMon-External: rh6node2pcmk.examplerh.com ClusterMon-
External monitor OK 0 0 0
Aug  7 11:34:00 rh6node1pcmk ClusterMon-External: rh6node2pcmk.examplerh.com ClusterIP start
OK 0 0 0
Aug  7 11:34:00 rh6node1pcmk ClusterMon-External: rh6node2pcmk.examplerh.com ClusterIP
monitor OK 0 0 0
```

第14章 PACEMAKER を用いたマルチサイトクラスターの設定

クラスターが複数のサイトにまたがる場合は、サイト間のネットワーク接続の問題が原因でスプリットブレインが発生する可能性があります。接続が切断されたときに、別のサイトのノードで障害が発生したのか、サイト間の接続に失敗した状態で別サイトのノードが機能しているかどうかをノードが判断する方法がありません。さらに、同時を維持するには離れすぎている 2 つのサイト間で高可用性サービスを提供することが問題になることがあります。

この問題に対応するため、Red Hat Enterprise Linux 7. 4 は Booth クラスターチケットマネージャーを使用して複数のサイトにまたがる高可用性クラスターを設定する機能に完全に対応しています。Booth *チケットマネージャー*は、特定サイトでクラスターノードに接続するネットワークとは異なる物理ネットワークで実行するための分散サービスです。これにより、*Booth フォーメーション* という別の非厳密なクラスターがサイトの通常クラスターの上に置かれます。この集約通信層は、個別の Booth チケットに対して合意ベースの決定プロセスを促進します。

Booth *チケット* は Booth フォーメーションのシングルトンで、時間依存の移動可能な承認の単位を表します。実行には特定のチケットを要求するようにリソースを設定できます。これにより、1 つまたは複数のチケットが付与されているサイトで、リソースは一度に 1 つのサイトでのみ実行されるようになります。

Booth フォーメーションは、複数のサイトで実行しているクラスターで構成され、元のクラスターがすべて独立しているオーバーレイクラスターと考えることができます。チケット付与の有無についてクラスターと通信するのは Booth サービスで、Pacemaker のチケット制約に基づいてクラスターでリソースを実行するかどうかを判断するのは Pacemaker です。これは、チケットマネージャーを使用する場合に、各クラスターが独自のリソースと共有リソースを実行できることを示しています。たとえば、リソース A、B、および C は 1 つのクラスターでのみ実行され、リソース D、E、および F は別のクラスターでのみ実行されるとします。リソース G および H は、チケットによって決定されたこの 2 つのクラスターのいずれかで実行されます。また、別のチケットにより、この 2 つのクラスターのいずれかで実行できるリソース J を追加することもできます。

以下の手順は、Booth チケットマネージャーを使用するマルチサイト設定を設定する手順の概要を示しています。

ここで使用するコマンド例は以下を前提とします。

- Cluster 1 は、ノード cluster1-node1 および cluster1-node2で設定されます。
- Cluster 1 に割り当てられたフローティング IP アドレスは 192.168.11.100 です。

- Cluster 2 は、cluster2 -node1 および cluster2-node2 で設定されます。
- Cluster 2 に割り当てられたフローティング IP アドレスは 192.168.22.100 です。
- arbitrator ノードは、IP アドレスが 192.168.99.100 の arbitrator-node です。
- この設定が使用する Booth チケットの名前は apacheticketです。

上記のコマンド例は、Apache サービスのクラスターリソースが、各クラスターの apachegroup リソースグループの一部として設定されていることを前提としています。各クラスターの Pacemaker インスタンスは独立しているため、このリソースのチケット制約を設定するために、各クラスターでリソースとリソースグループが同じである必要はありませんが、これはフェイルオーバーの一般的な事例になります。

クラスターで Apache サービスを設定するクラスター設定の完全な手順は、『High Availability Add-On の管理』 の例を参照してください。

設定手順のどの時点でも、pcs booth config コマンドを実行すると、現在のノードまたはクラスターの Booth 設定を表示できます。また、pcs booth status コマンドを実行すると、ローカルノードの現在の Booth ステータスを表示できます。

1. 両方のクラスター の各ノードに booth サイト の Booth チケットマネージャーパッケージをインストールします。

```
[root@cluster1-node1 ~]# yum install -y booth-site
[root@cluster1-node2 ~]# yum install -y booth-site
[root@cluster2-node1 ~]# yum install -y booth-site
[root@cluster2-node2 ~]# yum install -y booth-site
```

2. pcs パッケージ、booth-core パッケージ、および booth-arbitrator パッケージを arbitrator ノードにインストールします。

```
[root@arbitrator-node ~]# yum install -y pcs booth-core booth-arbitrator
```

3. 9929/tcp ポートと 9929/udp ポートがすべてのクラスターノードと仲裁ノードで解放されていることを確認します。

たとえば、両方のクラスターのすべてのノードと仲裁ノードで次のコマンドを実行すると、そのノードの **9929/tcp** ポートおよび **9929/udp** ポートにアクセスできます。

```
# firewall-cmd --add-port=9929/udp
# firewall-cmd --add-port=9929/tcp
# firewall-cmd --add-port=9929/udp --permanent
# firewall-cmd --add-port=9929/tcp --permanent
```

この手順自体により、任意のマシンがノード上の **9929** ポートにアクセスできることに注意してください。お使いのサイトで、ノードが必要なノードに対してのみ開いていることを確認する必要があります。

4. **1** つのクラスターの **1** つのノードで **Booth** 設定を作成します。各クラスターおよび仲裁ノードに指定するアドレスは **IP** アドレスでなければなりません。各クラスターにはフローティング **IP** アドレスを指定します。

```
[cluster1-node1 ~] # pcs booth setup sites 192.168.11.100 192.168.22.100 arbitrators
192.168.99.100
```

このコマンドを実行すると、**/etc/booth/booth.conf** および **/etc/booth/booth.key** 設定ファイルがノードに作成されます。

5. **Booth** 設定のチケットを作成します。このチケットは、このチケットがクラスターに付与された場合のみリソースの実行を許可するリソース抑制を定義するのに使用します。

このフェイルオーバー設定手順は基本的な手順で、チケットを **1** つだけ使用します。各チケットが別の **1** つ以上のリソースに関連付けられる、より複雑な事例では追加のチケットを作成します。

```
[cluster1-node1 ~] # pcs booth ticket add apacheticket
```

6. 現在のクラスターのすべてのノードに対して **Booth** 設定を同期します。

```
[cluster1-node1 ~] # pcs booth sync
```

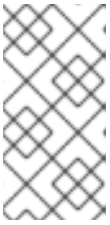
7. 仲裁ノードから、**Booth** 設定を仲裁者へプルします。これをまだ行っていない場合は、まず設定をプルするノードに **pcs** を認証する必要があります。

```
[arbitrator-node ~] # pcs cluster auth cluster1-node1
[arbitrator-node ~] # pcs booth pull cluster1-node1
```

8. **Booth 設定を別のクラスターにプルし、そのクラスターのすべてのノードを同期します。arbitrator ノードの場合と同様に、以前に行っていない場合は、最初に設定をプルするノードに pcs を認証する必要があります。**

```
[cluster2-node1 ~] # pcs cluster auth cluster1-node1
[cluster2-node1 ~] # pcs booth pull cluster1-node1
[cluster2-node1 ~] # pcs booth sync
```

9. **仲裁ノードで Booth を開始して有効にします。**



注記

Booth はクラスターで Pacemaker リソースとして実行するため、クラスターのノードで Booth を手動で開始したり有効にしたりしないでください。

```
[arbitrator-node ~] # pcs booth start
[arbitrator-node ~] # pcs booth enable
```

10. **Booth を設定し、両方のクラスターサイトでクラスターリソースとして実行されるようにします。booth-ip および booth-service をグループのメンバーとして持つリソースグループが作成されます。**

```
[cluster1-node1 ~] # pcs booth create ip 192.168.11.100
[cluster2-node1 ~] # pcs booth create ip 192.168.22.100
```

11. **各クラスターに定義したリソースグループにチケット制約を追加します。**

```
[cluster1-node1 ~] # pcs constraint ticket add apacheticket apachegroup
[cluster2-node1 ~] # pcs constraint ticket add apacheticket apachegroup
```

次のコマンドを実行すると、現在設定されているチケット制約を表示できます。

```
pcs constraint ticket [show]
```

12. **この設定用に作成したチケットを最初のクラスターに付与します。**

チケットを付与する前にチケット抑制を定義する必要はありません。最初にクラスターにチケットを許可した後、pcs booth ticket revoke コマンドを使用して手動で上書きしない限り、Booth はチケット管理を引き継ぎます。pcs booth 管理コマンドの詳細は、pcs booth コマンドの PCS ヘルプ画面 を参照してください。

```
[cluster1-node1 ~] # pcs booth ticket grant apacheticket
```

チケットは、いつでも (この手順の完了後でも) 追加および削除できます。ただし、チケットを追加または削除した後、この手順の説明どおりに、他のノード、クラスター、および仲裁ノードに対して設定ファイルを同期し、チケットを付与する必要があります。

Booth 設定ファイル、チケット、およびリソースのクリーンアップおよび削除に使用できる Booth 管理コマンドは、pcs booth コマンドの PCS ヘルプ画面を参照してください。

付録A OCF 戻りコード

この付録では、OCF 戻りコードと、Pacemaker に解釈される仕組みを説明します。

エージェントがコードを返したときに、クラスターがまず行うことは、期待されている結果通りにコードを返しているかどうかを確認します。そして、結果が期待されている値に一致しない場合、その操作が失敗したものとみなされ、復元操作が開始されます。

起動するには、起動した操作の結果の呼び出し元を通知する、定義した戻りコードでリソースエージェントを終了する必要があります。

表A.1「クラスターが行う復旧タイプ」 の説明どおり、障害回復には 3 種類あります。

表A.1 クラスターが行う復旧タイプ

タイプ	説明	クラスターが行った操作
軽度	一時的なエラーが発生しました。	リソースを再起動するか、新しい場所に移します。
重度	現在のノードに固有である可能性のある一時的ではないエラーが発生しました。	リソースを別の場所に移動し、現在のノードで再試行されないようにします。
致命的	すべてのクラスターノードに共有となる一時的でないエラーが発生しました (例: 指定された設定がよくありません)。	リソースを停止し、いかなるクラスターノードでも起動されないようにします。

表A.2「OCF 戻りコード」 では、OCF 戻りコードと、不具合のあるコードを受信したときにクラスターが開始する復旧タイプについて説明しています。0 を返すアクションでも、0 が想定さ れた戻り値でない場合に、失敗したと見なされることができます。

表A.2 OCF 戻りコード

戻りコード	OCF ラベル	説明
0	OCF_SUCCESS	操作が無事に完了しました。これは、起動、停止、昇格、降格コマンドに対して想定される戻りコードです。 予期しない場合のタイプ: ソフト
1	OCF_ERR_GENERIC	この操作は、一般的なエラーを返しました。 タイプ: ソフト リソースマネージャーは、リソースの復元と、新しい場所への移動を試行します。
2	OCF_ERR_ARGS	このマシンのリソースの設定が正しくありません。たとえば、ノードで見つからない場所を参照しています。 タイプ: 重度 リソースマネージャーがリソースを別の場所に移動し、現在のノードで再試行されないようにします。

戻りコード	OCF ラベル	説明
3	OCF_ERR_UNIMPLEMENTED	要求された操作が実装されていません。 タイプ: 重度
4	OCF_ERR_PERM	このリソースエージェントには、このタスクを完了するのに十分な特権がありません。これは、エージェントが特定のファイルを開けない場合や、特定のソケットでリッスンできない場合、ディレクトへの書き込みを行えない場合が考えられます。 タイプ: 重度 特に設定されていない限り、リソースマネージャーは、別のノード (パーミッションが存在しない) でリソースを再起動することで、このエラーで失敗したリソースの復旧を試行します。
5	OCF_ERR_INSTALLED	操作が実行されたノードに、必要なコンポーネントが欠如しています。これは、必要なバイナリーが実行不可であるか、重要な設定ファイルが読み込み不可になっていることが原因の場合があります。 タイプ: 重度 特に設定されていない限り、リソースマネージャーは、別のノード (必要なファイルまたはバイナリーが存在しない) でリソースを再起動することで、このエラーで失敗したリソースの復旧を試行します。

戻りコード	OCF ラベル	説明
6	OCF_ERR_CONFIGURED	ローカルノード上のリソースの設定が正しくありません。 タイプ: 致命的 このコードがかえされると、Pacemaker は、サービス設定がその他のノードで正しくても、クラスター内のノードでリソースが実行されないようにします。
7	OCF_NOT_RUNNING	このリソースは安全に停止します。これは、リソースが正常にシャットダウンされたか、起動されていないことを意味します。 予期しない場合のタイプ: ソフト クラスターは、いかなる操作に対しても、これを返すリソースの停止を試行しません。
8	OCF_RUNNING_MASTER	リソースはマスターモードで実行されています。 予期しない場合のタイプ: ソフト

戻りコード	OCF ラベル	説明
9	OCF_FAILED_MASTER	リソースはマスターモードで実行されていますが、不具合が発生しています。 タイプ: ソフト リソースは降格されて停止し、再起動されています (再び昇格されている可能性もあります)。
その他	該当なし	カスタムエラーコード

付録B RED HAT ENTERPRISE LINUX 6 および RED HAT ENTERPRISE LINUX 7 でのクラスターの作成

Pacemaker を使用して Red Hat Enterprise Linux 7 で Red Hat High Availability クラスターを設定するには、rgmanager を使用して Red Hat Enterprise Linux 6 でクラスターを設定する場合とは異なる設定ツールが必要になります。「クラスター作成 - rgmanager と Pacemaker」 では、さまざまなクラスターコンポーネント間の設定の違いをまとめています。

Red Hat Enterprise Linux 6.5 以降のリリースは pcs 設定ツールを使用した Pacemaker によるクラスター設定に対応しています。「Red Hat Enterprise Linux 6 および Red Hat Enterprise Linux 7 での Pacemaker のインストール」 では、Red Hat Enterprise Linux 6 と Red Hat Enterprise Linux 7 における Pacemaker インストールの相違点をまとめています。

B.1. クラスター作成 - RGMANAGER と PACEMAKER

表B.1「gmanager と Pacemaker を使用した場合のクラスター設定に関する比較」 では、Red Hat Enterprise Linux 6 で rgmanager を使用した場合と Red Hat Enterprise Linux 7 で Pacemaker を使用したクラスターのコンポーネントの設定方法を比較しています。

表B.1 gmanager と Pacemaker を使用した場合のクラスター設定に関する比較

設定コンポーネント	rgmanager	Pacemaker
クラスター設定ファイル	各ノードのクラスター設定ファイルは、直接編集できる **cluster.conf** ファイルです。それ以外の場合は、luci または **ccs** インターフェイスを使用してクラスター設定を定義します。	クラスターおよび Pacemaker の設定ファイルは **corosync.conf** および **cib.xml** です。**cib.xml** ファイルは直接編集しないでください。代わりに **pcs** または **pcsd** インターフェイスを使用してください。
ネットワーク設定	クラスター設定前に IP アドレスおよび SSH を設定	クラスター設定前に IP アドレスおよび SSH を設定
クラスター設定ツール	**cluster.conf** ファイルの手動編集で、ccci、**ccs** コマンド。	pcs または pcsd
インストール	**rgmanager** をインストールします(**ricci**、**luci**、リソースおよびフェンシングエージェントを含むすべての依存関係をプルします)。必要に応じて **lvm2-cluster** と **gfs2-utils** をインストールします。	**pcs** と必要なフェンシングエージェントをインストールします。必要に応じて **lvm2-cluster** と **gfs2-utils** をインストールします。

設定コンポーネント	rgmanager	Pacemaker
クラスターサービスの起動	次の手順でクラスターサービスを起動、有効にする 1. **rgmanager**、**cman**、および必要に応じて **clvmd** と **gfs2** を起動します。 2. luci インターフェイスを使用する場合は **luci** を開始し、**luci** を開始します。 3. **必要なサービスに対して chkconfig on を実行し、各ランタイムで起動するようにします。** **または、ccs --start を入力してクラスターサービスを開始および有効化することもできます。**	次の手順でクラスターサービスを起動、有効にする 1. すべてのノードで **systemctl start pcsd.service** を実行した後に **systemctl enable pcsd.service** を実行し、ランタイム時に **pcsd** が起動するようにします。 2. クラスター内の 1 つのノードで、**pcs cluster start --all** を入力し、**corosync** および **pacemaker** を起動します。
設定ツールへのアクセスの制御	**luci** の場合、root ユーザーまたは **luci** 権限を持つユーザーは luci にアクセスできます。すべてのアクセスには、**ノードの** ラクショットパスワードが必要です。	**pcsd** gui では、共通のシステムユーザーであるユーザー **hacluster** として認証する必要があります。root ユーザーは **hacluster** のパスワードを設定できます。
クラスター作成	クラスターに名前を付け、**luci** または **ccs** を使用してクラスターに追加するノードを定義するか、**cluster.conf** ファイルを直接編集します。	**pcs cluster setup** コマンドまたは **pcsd** Web UI を使用して、クラスターに名前を付け、ノードを含めます。**pcs cluster node add** コマンドまたは **pcsd** Web UI を使用すると、ノードを既存のクラスターに追加できます。
クラスター設定を全ノードに伝える	クラスターを **luci** で設定する場合、伝播は自動的に行われます。**ccs** で、**--sync** オプションを使用します。**cman_tool version -r** コマンドを使用することもできます。	クラスターおよび Pacemaker 設定ファイル **corosync.conf** および **cib.xml** は、クラスターの設定時またはノードまたはリソースの追加時に自動的に伝播されます。

設定コンポーネント	rgmanager	Pacemaker
グローバルのクラスタープロパティー	以下の機能は、Red Hat Enterprise Linux 6 の **rgmanager** で対応しています。 * クラスターネットワーク内での IP マルチキャスティングに使用するマルチキャストアドレスの選択をシステム側で行うよう設定することが可能 * IP マルチキャスティングが利用できない場合に UDP Unicast トランスポートメカニズムが使用可能 * RRP プロトコルを使用したクラスター設定が可能	Red Hat Enterprise Linux 7 の Pacemaker はクラスターに対して以下の機能をサポートします。 * クラスターに **no-quorum-policy** を設定し、クラスターにクォーラムがない場合のシステムの動作を指定できます。 * 設定可能な追加のクラスタープロパティーについては、表12.1「クラスターのプロパティー」 を参照してください。
ロギング	グローバルおよびデーモン固有のログ記録設定が可能	ログ記録を手動で設定する方法については、ファイル **/etc/sysconfig/pacemaker** を参照してください。
クラスターの検証	クラスター検証は、クラススキーマを使用して luci および **ccs** で自動的に行われます。クラスターは起動時に自動的に検証されます。	クラスターは起動時に自動的に検証されるか、**pcs cluster verify** を使用してクラスターを検証できます。
2 ノードクラスターのクォーラム	2 ノードのクラスターの場合、システムによるクォーラムの決定方法を設定できます。 * クォーラムディスクの設定 * **ccs** を使用するか、**cluster.conf** ファイルを編集して **two_node=1** および **expected_votes=1** を設定し、単一ノードがクォーラムを維持できるようにします。	**pcs** は、2 ノードクラスターに必要なオプションを **corosync** に自動的に追加します。
クラスターの状態	luci では、クラスターの現在のステータスがインターフェイスのさまざまなコンポーネントに表示され、更新できます。**ccs** コマンドの **--getconf** オプションを使用して、現在の設定ファイルを確認できます。**clustat** コマンドを使用して、クラスターのステータスを表示できます。	**pcs status** コマンドを使用すると、現在のクラスターのステータスを表示できます。
リソース	定義されたタイプのリソースを追加し、luci または **ccs** コマンドを使用するか、**cluster.conf** 設定ファイルを編集してリソース固有のプロパティーを設定します。	**pcs resource create** コマンドまたは **pcsd** Web UI を使用して、定義されたタイプのリソースを追加し、リソース固有のプロパティーを設定します。Pacemaker を使用してクラスターリソースを設定する方法は、*6章 クラスターリソースの設定* を参照してください。

設定コンポーネント	rgmanager	Pacemaker
リソースの動作、グループ化、起動と停止の順序	リソースの通信方法の設定にクラスターの **サービス** を定義	Pacemaker では、同時に配置され、順番に開始および停止される必要があるリソースのセットを定義する簡単な方法としてリソースグループを使用します。さらに、以下の方法でリソースの動作および対話方法を定義できます。 * リソース動作の一部はリソースオプションとして設定 * 場所の制約を使ってリソースを実行させるノードを指定 * 順序の制約を使ってリソースの実行順序を指定 コロケーション制約を使って任意のリソースの場所が別のリソースの場所に依存することを指定 これらのトピックの詳細については *6章 クラスターリソースの設定* および *7章 リソースの制約* を参照してください。
リソース管理: リソースの移動、起動、停止	luci を使用すると、クラスター、個別のクラスターノード、およびクラスターサービスを管理できます。**ccs** コマンドを使用すると、クラスターを管理できます。**clusvadm** を使用して、クラスターサービスを管理できます。	**pcs cluster standby** コマンドを使用して、ノードを一時的に無効にし、リソースをホストできないようにすることができます。これにより、リソースが移行されます。**pcs resource disable** コマンドを使用して、リソースを停止できます。
クラスターの設定を完全に削除	luci では、削除してクラスターを完全に削除するためにクラスター内のすべてのノードを選択できます。クラスターの各ノードから **cluster.conf** を削除することもできます。	**pcs cluster destroy** コマンドを使用するとクラスター設定を削除できます。
複数のノードで実行中のリソース、複数モードで複数のノード上で実行中のリソース	該当なし。	Pacemaker ではリソースのクローンを作成し複数のノードで実行させることが可能で、作成したリソースのクローンをマスターリソースとスレーブリソースとして定義し複数のモードで実行させることが可能です。クローン作成したリソースおよびマスター/スレーブリソースに関する情報は、*9章 高度な設定* を参照してください。

設定コンポーネント	rgmanager	Pacemaker
フェンス機能 -- 1 ノードにつき 1 フェンス	フェンスデバイスをグローバルに作成するか、またはローカルに作成し、ノードに追加します。**post-fail delay** と **post-join delay** の値を全体として定義できます。	**pcs stonith create** コマンドまたは **pcsd** Web UI を使用して、各ノードにフェンスデバイスを作成します。複数のノードをフェンスできるデバイスでは、各ノードで個別に定義せずに 1 度に定義する必要があります。**pcmk_host_map** を定義して、1 つのコマンドで全ノードのフェンスデバイスを設定することもできます。**pcmk_host_map** の詳細は、表 5.1「フェンスデバイスの一般的なプロパティー」 を参照してください。クラスターの **stonith-timeout** の値を全体として定義できます。
1 ノードごとに複数の (バックアップ) フェンスデバイス	luci または **ccs** コマンドを使用してバックアップデバイスを定義するか、**cluster.conf** ファイルを直接編集します。	フェンスレベルを設定

B.2. RED HAT ENTERPRISE LINUX 6 および RED HAT ENTERPRISE LINUX 7 での PACEMAKER のインストール

Red Hat Enterprise Linux 6.5 以降のリリースは pcs 設定ツールを使用した Pacemaker によるクラスター設定に対応しています。Pacemaker を使用する際、Red Hat Enterprise Linux 6 と Red Hat Enterprise Linux 7 におけるクラスターのインストールには、いくつか相違点があります。

以下のコマンドは、Red Hat Enterprise Linux 6 で Pacemaker が必要とする Red Hat High Availability Add-On ソフトウェアパッケージをインストールし、cman なしで corosync が起動しないようにします。クラスターの各ノードでこれらのコマンドを実行する必要があります。

```
[root@rhel6]# yum install pacemaker cman pcs
[root@rhel6]# chkconfig corosync off
[root@rhel6]# chkconfig cman off
```

クラスターの各ノードで、hacluster という名前の pcs 管理アカウントのパスワードを設定し、pcsd サービスを開始して有効にします。

```
[root@rhel6]# passwd hacluster
[root@rhel6]# service pcsd start
[root@rhel6]# chkconfig pcsd on
```

クラスターの 1 つのノードでは、クラスターのノードの管理者アカウントの認証を行います。

```
[root@rhel6]# pcs cluster auth [node] [...] [-u username] [-p password]
```

Red Hat Enterprise Linux 7 では、クラスターの各ノードで以下のコマンドを実行し、Pacemaker が必要とする Red Hat High Availability Add-On ソフトウェアパッケージをインストールして hacluster という名前の pcs 管理アカウントのパスワードを設定し、pcsd サービスを起動および有効にします。

```
[root@rhel7]# yum install pcs pacemaker fence-agents-all
[root@rhel7]# passwd hacluster
[root@rhel7]# systemctl start pcsd.service
[root@rhel7]# systemctl enable pcsd.service
```

Red Hat Enterprise Linux 7 では、Red Hat Enterprise Linux 6 と同様に、クラスターのノードで以下のコマンドを実行して、クラスターのノードの管理者アカウントを認証します。

```
[root@rhel7]# pcs cluster auth [node] [...] [-u username] [-p password]
```

Red Hat Enterprise Linux 7 のインストールは、1章*Red Hat High Availability Add-On 設定および管理リファレンスの概要* と 4章*クラスターの作成と管理* を参照してください。

付録C 更新履歴

改訂 8.1-1 7.8 Beta 公開用ドキュメントバージョン	Fri Feb 28 2020	Steven Levine
改訂 7.1-1 7.7 GA 公開用ドキュメントバージョン	Wed Aug 7 2019	Steven Levine
改訂 6.1-1 7.6 GA 公開用ドキュメントバージョン	Thu Oct 4 2018	Steven Levine
改訂 5.1-2 7.5 GA 公開用ドキュメントバージョン	Thu Mar 15 2018	Steven Levine
改訂 5.1-0 7.5 ベータ版公開用ドキュメントバージョン	Thu Dec 14 2017	Steven Levine
改訂 4.1-9 7.4 のバージョンを更新	Tue Oct 17 2017	Steven Levine
改訂 4.1-5 7.4 GA 公開用ドキュメントバージョン	Wed Jul 19 2017	Steven Levine
改訂 4.1-2 7.4 ベータ版公開用ドキュメントの準備	Wed May 10 2017	Steven Levine
改訂 3.1-10 7.3 GA 公開用バージョンの更新。	Tue May 2 2017	Steven Levine
改訂 3.1-4 7.3 GA リリースのバージョン	Mon Oct 17 2016	Steven Levine
改訂 3.1-3 7.3 ベータ公開用ドキュメントの準備	Wed Aug 17 2016	Steven Levine
改訂 2.1-8 7.2 GA 公開用ドキュメントの準備	Mon Nov 9 2015	Steven Levine
改訂 2.1-5 7.2 ベータ公開用ドキュメントの準備	Mon Aug 24 2015	Steven Levine
改訂 1.1-9 7.1 GA リリース向けのバージョン	Mon Feb 23 2015	Steven Levine
改訂 1.1-7 7.1 ベータリリース向けバージョン	Thu Dec 11 2014	Steven Levine
改訂 0.1-41 7.0 GA リリース向けバージョン	Mon Jun 2 2014	Steven Levine
改訂 0.1-2 初回ドラフトの初版	Thu May 16 2013	Steven Levine

索引

A

B

C

D

T