Red Hat JBoss Core Services 2.4.51

Apache HTTP Server コネクターおよび負荷分散ガイド

Red Hat JBoss Core Services 2.4.51 での使用

Last Updated: 2024-02-08

Red Hat JBoss Core Services 2.4.51 Apache HTTP Server コネクターおよび負荷分散ガイド

Red Hat JBoss Core Services 2.4.51 での使用

法律上の通知

概要

Red Hat JBoss Core Services Apache HTTP Server が提供する他のモジュールとともに mod_jk および mod_proxy_cluster HTTP コネクターを使用する負荷分散ソリューションをインストールして設定する

目次

RED HAT ドキュメントへのフィードバック (英語のみ)

エラーを報告したり、ドキュメントを改善したりするには、Red Hat Jira アカウントにログインし、課題を送信してください。Red Hat Jira アカウントをお持ちでない場合は、アカウントを作成するように求められます。

手順

1. **このリンクをクリック** してチケットを作成します。
2. **Summary** に課題の簡単な説明を入力します。
3. **Description** に課題や機能拡張の詳細な説明を入力します。問題があるドキュメントのセクションへの URL を含めてください。
4. **Submit** をクリックすると、課題が作成され、適切なドキュメントチームに転送されます。

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、Red Hat CTO である Chris Wright のメッセージ をご覧ください。

第1章 HTTP コネクター

Red Hat JBoss Core Services (JBCS) には、Apache HTTP Serve が一連のバックエンドサーブレットコンテナーへの HTTP リクエストの負荷分散に使用できる 2 つの異なる HTTP コネクターが含まれています。

- Apache Tomcat コネクター (**mod_jk**) は、スティッキーセッションを維持し、Apache JServ Protocol (AJP) を介して通信しながら、一連のサーブレットコンテナーに対する HTTP 要求の負荷分散をサポートします。
- JBoss HTTP コネクター (**mod_proxy_cluster**) は、**mod_jk** よりも高度なロードバランサーです。**mod_proxy_cluster** コネクターは、**mod_jk** のすべての機能と、リアルタイムの負荷分散計算、アプリケーションライフサイクル制御、自動プロキシー検出、複数プロトコルサポートなどの追加機能を提供します。

JBCS と Red Hat Enterprise Linux (RHEL) は、Apache HTTP Server を別々に配布しています。Apache HTTP サーバーの JBCS ディストリビューションを使用すると、**mod_jk**、**mod_proxy_cluster**、または **mod_proxy** コネクターをプロキシーとして使用してバックエンドアプリケーションサーバーに接続できます。

次のガイドラインを考慮してください。

- RHEL 7 以降では、Apache HTTP サーバーの JBCS および RHEL ディストリビューションで同じ **mod_proxy** モジュールが提供されています。
- RHEL バージョン 7 および 8 では、Apache HTTP サーバーの JBCS ディストリビューションのみで **mod_jk** および **mod_proxy_cluster** コネクターが提供されています。
- RHEL 9 以降では、Apache HTTP Server の JBCS および RHEL ディストリビューションで **mod_jk** コネクターと **mod_proxy_cluster** コネクターの同一コピーが提供されています。
- **groupinstall** オプションを使用してアーカイブファイルまたは RPM パッケージから Apache HTTP サーバーの JBCS ディストリビューションをインストールすると、**mod_jk** コネクターと **mod_proxy_cluster** コネクターも自動的にインストールされます。
- Apache HTTP サーバーの RHEL 9 ディストリビューションをインストールしても、**mod_jk** コネクターと **mod_proxy_cluster** コネクターは自動的にインストールされません。この状況では、RHEL Application Streams を使用して、適切な **mod_jk** または **mod_proxy_cluster** パッケージを手動でインストールできます。詳細は、**Mod_jk のインストール** および **Mod_proxy_cluster のインストールとアップグレード** を参照してください。

Apache HTTP サーバーコネクターおよび負荷分散ガイドでは、JBCS が提供する **mod_jk** および **mod_proxy_cluster** コネクターをインストールして設定する方法について説明しています。このガイドには mod_proxy_cluster を使用した負荷分散 の基本的な実践例も含まれています。

重要

このガイドで示されるファイルおよびディレクトリーパスのほとんどは、Red Hat Enterprise Linux 上にある JBoss Core Services のアーカイブインストールに使用します。他のプラットフォームについては、Red Hat JBoss Core Services Apache HTTP Server インストールガイド の説明に従い、それぞれのインストールに適切なパスを使用してください。

関連情報

- Apache Tomcat コネクター (**mod_jk**) を使用した負荷分散

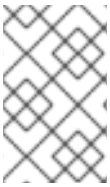
- JBoss HTTP コネクター (**mod_proxy_cluster**) を使用した負荷分散

第2章 APACHE TOMCAT コネクター (MOD_JK) を使用した負荷分散

Apache Tomcat コネクター **mod_jk** は、Apache HTTP Server が Web リクエストをバックエンドサーブレットコンテナーに転送できるようにするプラグインです。**mod_jk** モジュールを使用すると、Apache HTTP Server は、スティッキーセッションを維持しながら、サーブレットコンテナーのセットへの要求を負荷分散することもできます。

2.1. MOD_JK のインストール

Red Hat JBoss Core Services (JBCS) と Red Hat Enterprise Linux (RHEL) は、Apache HTTP Server の個別のディストリビューションを提供します。インストールする Apache HTTP Server ディストリビューションに応じて、**mod_jk** コネクターが自動または手動のどちらでインストールされるか決まります。インストールされている Apache HTTP サーバーのディストリビューションにより、**mod_jk** モジュールと設定ファイルのインストールパスも異なります。



注記

JBCS Apache HTTP サーバーでは、すべてのサポート対象オペレーティングシステムで **mod_jk** の使用がサポートされています。RHEL Apache HTTP サーバーは、RHEL 9 上でのみ **mod_jk** の使用をサポートします。

2.1.1. JBCS Apache HTTP サーバーを使用する場合の mod_jk のインストール

JBCS インストールの Apache HTTP サーバー部分では、**mod_jk** モジュールが自動的にインストールされます。

Red Hat JBoss Core Services Apache HTTP Server インストールガイドの手順に従って、オペレーティングシステム用の Apache HTTP Server をインストールできます。詳細は、**関連情報** セクションを参照してください。

JBCS Apache HTTP サーバーを使用する場合は、**mod_jk** のインストールに関する次のガイドラインを考慮してください。

- **mod_jk.so** モジュールは、**JBCS_HOME/httpd/modules** ディレクトリーにインストールされます。
- **mod_jk.conf.sample**、**workers.properties.sample**、**urworkermap.properties.sample** の設定ファイルは、**JBCS_HOME/httpd/conf.d** ディレクトリーにあります。
- **mod_jk.conf.sample** ファイルには、**mod_jk** モジュールの **LoadModule** ディレクティブが含まれています。

注記

JBCS_HOME は、JBCS インストールの最上位ディレクトリー (**/opt/jbcs-httpd24-2.4**) を表します。

関連情報

- アーカイブファイルから RHEL に JBCS Apache HTTP サーバーをインストールする
- RPM パッケージから RHEL 7 または RHEL 8 に JBCS Apache HTTP サーバーをインストールする

- Windows Server に JBCS Apache HTTP サーバーをインストールする

2.1.2. RHEL アプリケーションストリームを使用する場合の mod_jk のインストール

Application Streams を使用して RPM パッケージから Apache HTTP Server の RHEL 9 ディストリビューションをインストールする場合、RHEL は **mod_jk** パッケージを自動的にインストールしません。この状況で **mod_jk** コネクターを使用する場合は、**mod_jk** パッケージを手動でインストールする必要があります。

前提条件

- Application Streams を使用して、Apache HTTP サーバーを RHEL 9 にインストールている。

手順

- root ユーザーとして以下のコマンドを実行します。

```
# dnf install mod_jk
```

検証

- **mod_jk** パッケージが正常にインストールされたことを確認するには、次のコマンドを入力します。

```
# rpm -q mod_jk
```

前述のコマンドは、バージョンとプラットフォームの情報を含む、インストールされたパッケージの完全名を出力します。

RHEL Application Streams を使用する場合は、**mod_jk** のインストールに関する次のガイドラインを考慮してください。

- **mod_jk.so** モジュールは **/usr/lib64/httpd/modules** ディレクトリーにインストールされます。
- **mod_jk.conf.sample**、**workers.properties.sample**、**urworkermap.properties.sample** の設定ファイルは、**/etc/httpd/conf.d** ディレクトリーにあります。
- **mod_jk.conf.sample** ファイルには、**mod_jk** モジュールの **LoadModule** ディレクティブが含まれています。

関連情報

- Application Streams (AppStream)
- DNF ツールを使用したソフトウェアの管理

2.2. MOD_JK を使用する場合の APACHE HTTP SERVER 負荷分散設定

Apache HTTP Server は、**mod_jk** コネクターを使用してリクエストを一連のサーブレットコンテナーに負荷分散するように設定できます。このセットアップには、バックエンドワーカーノードの設定が含まれます。

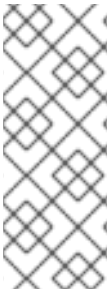
mod_jk を Red Hat JBoss Core Services (JBCS) 経由でインストールしたか、Red Hat Enterprise Linux (RHEL) Application Streams を使用してインストールしたかに応じて、次のガイドラインを考慮してください。

- JBCS の場合、**JBCS_HOME/httpd/conf.d/** ディレクトリーに **mod_jk** のサンプル設定ファイルがあります。
- RHEL の場合、**/etc/httpd/conf.d/** ディレクトリーに **mod_jk** のサンプル設定ファイルがあります。

mod_jk のサンプル設定ファイルの名前は、**mod_jk.conf.sample**、**workers.properties.sample**、**uriworkermap.properties.sample** です。独自の設定ファイルを作成する代わりにこれらの例を使用するには、**.sample** 拡張子を削除し、必要に応じてファイルの内容を変更します。

注記

また、Red Hat Customer Portal の Load Balancer Configuration ツールを使用して、**mod_jk** および Tomcat ワーカーノードに最適な設定テンプレートをすばやく生成することもできます。Apache HTTP Server 2.4.51 のロードバランサー設定ツールを使用する場合は、Apache バージョンに **2.4.x** を、バックエンド設定に **Tomcat/JWS** を選択してください。



注記

Red Hat JBoss Core Services 2.4.51 では、バックエンドの Websocket サーバーに対する接続がアップグレードされていない場合に、その接続のトンネリングはサポートされません。つまり、**mod_proxy_wstunnel** モジュールの **ProxyPass** ディレクティブを設定するときに、アップグレードパラメーターが **NONE** に設定されていないことを確認する必要があります。**mod_proxy_wstunnel** の詳細は、Apache のドキュメント を参照してください。

2.2.1. mod_jk をロードする場合の Apache HTTP Server 設定

mod_jk.conf ファイルで設定を指定することにより、**mod_jk** をロードするように Apache HTTP Server を設定できます。使用している Apache HTTP Server ディストリビューションにより、設定ファイルの場所は異なります。

次のオプションの設定手順を実行することもできます。

- **JkMount** ディレクティブの他に、**JkMountFile** ディレクティブを使用してマウントポイントの設定ファイルを指定できます。設定ファイルには、Tomcat 転送の複数の URL マッピングが含まれます。
- ロードバランサーとして機能している Apache HTTP Server を設定して、要求を処理する各ワーカーノードの詳細をログに記録できます。これは、ロードバランサーのトラブルシューティングが必要な場合に役立ちます。

前提条件

- Apache HTTP サーバーがインストール されている。
- Application Streams を使用して Apache HTTP Server の RHEL ディストリビューションをインストールした場合は、**mod_jk** を手動でインストール している。

手順

1. Apache HTTP Server 設定ディレクトリーに移動します。
   - JBCS Apache HTTP Server を使用している場合は、**JBCS_HOME/httpd/conf.d** ディレクトリーに移動します。
   - RHEL Apache HTTP Server を使用している場合は、**/etc/httpd/conf.d** ディレクトリーに移動します。
2. **mod_jk.conf** という名前の新しいファイルを作成し、次の設定の詳細を入力します。

```
# Load mod_jk module
# Specify the filename of the mod_jk lib
LoadModule jk_module modules/mod_jk.so

# Where to find workers.properties
JkWorkersFile conf.d/workers.properties

# Where to put jk logs
JkLogFile logs/mod_jk.log

# Set the jk log level [debug/error/info]
JkLogLevel info

# Select the log format
JkLogStampFormat  "[%a %b %d %H:%M:%S %Y]"

# JkOptions indicates to send SSL KEY SIZE
JkOptions +ForwardKeySize +ForwardURICompat -ForwardDirectories

# JkRequestLogFormat
JkRequestLogFormat "%w %V %T"

# Mount your applications
JkMount /application/* loadbalancer

# Add shared memory.
# This directive is present with 1.2.10 and
# later versions of mod_jk, and is needed for
# for load balancing to work properly
JkShmFile logs/jk.shm

# Add jkstatus for managing runtime data
<Location /jkstatus/>
    JkMount status
    Require ip 127.0.0.1
</Location>
```

重要

LoadModule ディレクティブが、インストールした **mod_jk** ネイティブバイナリーを参照していることを確認します。

注記

JkMount ディレクティブは、Apache HTTP Server が **mod_jk** モジュールに転送できる URL を指定します。**JkMount** ディレクティブの設定に基づいて、**mod_jk** は受信した URL を正しいサーブレットコンテナーに転送します。

Apache HTTP Server が静的コンテンツ (または PHP コンテンツ) を直接提供し、Java アプリケーションにのみロードバランサーを使用できるようにするには、前述の設定例では、Apache HTTP Server が URL **/application/*** のリクエストのみを **mod_jk** ロードバランサーに送信するよう指定します。

または、**JkMount** ディレクティブで /* を指定して、すべての URL を **mod_jk** に転送するように Apache HTTP Server を設定することもできます。

3. オプション: **JkMountFile** ディレクティブを使用してマウントポイントの設定ファイルを指定するには、次の手順を実行します。

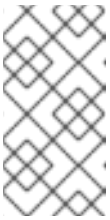
   a. Apache HTTP Server 設定ディレクトリーに移動します。

      - JBCS Apache HTTP Server を使用している場合は、**JBCS_HOME/httpd/conf.d** ディレクトリーに移動します。
      - RHEL Apache HTTP Server を使用している場合は、**/etc/httpd/conf.d** ディレクトリーに移動します。

   b. **uriworkermap.properties** という名前のファイルを作成します。

   c. 転送する URL とワーカー名を指定します。
   以下に例を示します。

```
# Simple worker configuration file

# Mount the Servlet context to the ajp13 worker
/application=loadbalancer
/application/*=loadbalancer
```



注記

必要な構文は次の形式です: **/URL = WORKER_NAME**

上記の例では、**/application** のリクエストを JBoss Web Server Tomcat バックエンドに転送するように **mod_jk** を設定しています。

   d. **mod_jk.conf** ファイルに、次のディレクティブを入力します。

```
# Use external file for mount points.
# It will be checked for updates each 60 seconds.
# The format of the file is: /url=worker
# /examples/*=loadbalancer
JkMountFile conf.d/uriworkermap.properties
```

4. オプション: Apache HTTP Server のログを有効にするには、次のいずれかの手順を実行します。

   - **mod_jk.conf** 設定に関する前の手順で示したように、**JkRequestLogFormat** ディレクティ

ブに **%w** を含めます。

- **%{JK_WORKER_NAME}n** を Apache HTTP Server **LogFormat**(s) に含めて、使用する **mod_jk** ワーカーの名前をログに記録します。

関連情報

- The Apache Tomcat Connectors - Web Server HowTo : mod_jk ディレクティブ
- Apache HTTP Server のドキュメント: ログファイル

2.2.2. mod_jk でのワーカーノードの設定

works.properties ファイルで設定を指定することで、Apache HTTP Server がサーブレットコンテナーに転送するリクエストを処理するように複数のワーカーノードを設定できます。使用している Apache HTTP Server ディストリビューションにより、設定ファイルの場所は異なります。

この手順の例では、2 つのサーブレットコンテナー間でスティッキーセッションを使用する加重ラウンドロビン設定で 2 つの **mod_jk** ワーカーノードを定義する方法を示します。

前提条件

- **worker.properties** ディレクティブ の形式に精通していること。
- **mod_jk** をロードするように Apache HTTP Server を設定しました。

手順

1. Apache HTTP Server 設定ディレクトリーに移動します。
   - JBCS Apache HTTP Server を使用している場合は、**JBCS_HOME/httpd/conf.d** ディレクトリーに移動します。
   - RHEL Apache HTTP Server を使用している場合は、**/etc/httpd/conf.d** ディレクトリーに移動します。
2. **workers.properties** という名前のファイルを作成します。
3. 次の設定の詳細を入力します。

```
# Define list of workers that will be used
# for mapping requests
worker.list=loadbalancer,status

# Define Node1
# modify the host as your host IP or DNS name.
worker.node1.port=8009
worker.node1.host=node1.mydomain.com
worker.node1.type=ajp13
worker.node1.ping_mode=A
worker.node1.lbfactor=1
worker.node1.secret=<YourSecret>

# Define Node2
# modify the host as your host IP or DNS name.
worker.node2.port=8009
```

```
worker.node2.host=node2.mydomain.com
worker.node2.type=ajp13
worker.node2.ping_mode=A
worker.node2.lbfactor=1
worker.node1.secret=<YourSecret>

# Load-balancing behavior
worker.loadbalancer.type=lb
worker.loadbalancer.balance_workers=node1,node2
worker.loadbalancer.sticky_session=1

# Status worker for managing load balancer
worker.status.type=status
```

注記

前の例で、**host**、**port**、および **secret** の設定を環境に適した値に置き換えてください。

重要

secret プロパティーは Tomcat AJP コネクターを使用する場合に必要になります。ワーカーノードまたはロードバランサーの **シークレット** プロパティーは、**workers.properties** ファイルで指定できます。以下に例を示します。

worker.<WORKER_NAME>.secret=<YOUR_AJP_SECRET>

前の例で、**<WORKER_NAME>** と **<YOUR_AJP_SECRET>** を環境に適した値に置き換えます。

2.2.3. mod_jk と連携する場合の JBoss Web Server 設定

デフォルトでは、JBoss Web Server は **mod_jk** コネクターから Apache JServ Protocol (AJP) トラフィックを受信するように設定されています。JBoss Web Server ホストでは、AJP コネクターはデフォルトで **JWS_HOME/tomcat <VERSION>/conf/server.xml** ファイルに設定されます。

ただし、**mod_jk** でワーカーノードを使用するには、次の設定手順を追加で実行する必要があります。

- JBoss Web Server ホスト上の **server.xml** ファイルで、各ワーカーノードのエンジンの **jvmRoute** 属性に対して一意の値を設定する必要があります。
- Apache HTTP Server ホスト上の **works.properties** ファイルで、ワーカーノードまたはロードバランサーの **secret** プロパティーを指定する必要があります。使用している Apache HTTP Server ディストリビューションにより、**workers.properties** ファイルの場所は異なります。

注記

secret プロパティーは Tomcat AJP コネクターを使用する場合に必要になります。

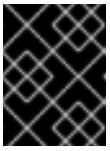
手順

1. JBoss Web Server ホスト上で、各ワーカーノードのエンジンの **jvmRoute** 属性に対して一意の値を設定するには、以下を実行します。

a. **JWS_HOME/tomcat_<VERSION>_/conf/server.xml** ファイルを開きます。

b. 以下の詳細を入力します。

```
<Engine name="Catalina" jvmRoute="node1" >
```



重要

jvmRoute 属性値が、Apache HTTP Server ホスト上の **workers.properties** ファイルで指定したワーカー名と一致していることを確認してください。

2. Apache HTTP Server ホスト上で、ワーカーノードまたはロードバランサーの **secret** プロパティーを指定するには、以下を実行します。

a. Apache HTTP Server 設定ディレクトリーに移動します。

- JBCS Apache HTTP Server を使用している場合は、**JBCS_HOME/httpd/conf.d** ディレクトリーに移動します。
- RHEL Apache HTTP Server を使用している場合は、**/etc/httpd/conf.d** ディレクトリーに移動します。

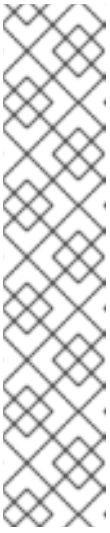
b. **works.properties** ファイルを開きます。

c. **secret** プロパティーが次の形式で指定されていることを確認してください。

```
worker.<WORKER_NAME>.secret=<YOUR_AJP_SECRET>`
```

注記

<WORKER_NAME> と **<YOUR_AJP_SECRET>** は、使用している環境に適した値に置き換えてください。



注記

ProxyPass ディレクティブを使用してロードバランサーに **secret** を設定すると、ロードバランサーの全メンバーがこの **secret** を継承します。以下に例を示します。

```
<Proxy balancer://mycluster>`
   BalancerMember ajp://node1:8009 route=node1
secret=YOUR_AJP_SECRET
   BalancerMember ajp://node2:8009 route=node2
secret=YOUR_AJP_SECRET
</Proxy>
ProxyPass /example/ balancer://mycluster/example/
stickysession=JSESSIONID|jsessionid
```

第3章 JBOSS HTTP コネクター (MOD_PROXY_CLUSTER) を使用した負荷分散

mod_proxy_cluster コネクターは、設定が簡素化されたインテリジェントな負荷分散ソリューションです。これを使用することで、Apache HTTP サーバーはバックエンドの JBoss Web Server または JBoss EAP ホストに接続できるようになります。**mod_proxy_cluster** モジュールは、JBoss **mod_cluster** コミュニティープロジェクトが最初に開発した技術に基づいています。

3.1. MOD_PROXY_CLUSTER の主な特徴とコンポーネント

mod_proxy_cluster モジュールは、JBoss EAP および JBoss Web Server ワーカーノードへの HTTP リクエストの負荷を分散します。**mod_proxy_cluster** モジュールは、プロキシーサーバーとして Apache HTTP Server を使用します。

mod_proxy_cluster の主な機能

他の **mod_proxy_cluster** コネクターと比べて **mod_cluster** コネクターには複数の利点があります。

- **mod_proxy_cluster** モジュールを有効にすると、**mod_proxy_cluster** Management Protocol (MCMP) が Tomcat サーバーと Apache HTTP Server の間の追加接続となります。Tomcat サーバーは、MCMP を使用して、HTTP メソッドのカスタムセットを使用して、サーバー側の負荷数値とライフサイクルイベントを Apache HTTP Server に送信します。
- Apache HTTP Server with **mod_proxy_cluster** の動的な設定により、**mod_proxy_cluster** リスナーを持つ Tomcat サーバーは、手動で設定しなくても負荷分散配置に加わることができます。
- Tomcat サーバーは、Apache HTTP Server に依存せず、負荷計算を実行します。これにより、負荷分散メトリックが他のコネクターよりも正確になります。
- **mod_proxy_cluster** コネクターにより、アプリケーションライフサイクルを細かく制御できるようになります。各 Tomcat サーバーは、Web アプリケーションコンテキストのライフサイクルイベントを Apache HTTP Server に転送します。これらのライフサイクルイベントには、特定のコンテキストのルーティングリクエストを開始または停止するように Apache HTTP Server に通知することが含まれます。これにより、リソースが利用できないためにエンドユーザーに HTTP エラーが表示されるのを防ぎます。
- **mod_proxy_cluster** では、Apache JServ Protocol (AJP)、Hypertext Transfer Protocol (HTTP)、または Hypertext Transfer Protocol Secure (HTTPS) トランスポートを使用できます。

Mod_proxy_cluster コンポーネント

プロキシーサーバーでは、**mod_proxy_cluster** は 4 つの Apache モジュールで設定されます。

コンポーネント	説明
mod_cluster_slotmem.so	Shared Memory Manager モジュールは、リアルタイムのワーカーノード情報を複数の Apache HTTP Server プロセスと共有します。
mod_manager.so	Cluster Manager モジュールは、ノードの登録、ノードの負荷データ、ノードアプリケーションのライフサイクルイベントなど、ワーカーノードからメッセージを受信および確認します。

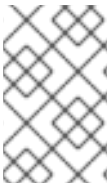
コンポーネント	説明
mod_proxy_cluster.so	Proxy Balancer モジュールは、クラスターノードへの要求ルーティングを処理します。Proxy Balancer は、クラスターのアプリケーションの場所、各クラスターノードの現在の状態、およびセッション ID (リクエストが確立されたセッションの一部である場合) に基づいて適切な宛先ノードを選択します。
mod_advertise.so	Proxy Advertisement Module は、UDP マルチキャストメッセージを介してプロキシーサーバーの存在をブロードキャストします。サーバーのアドバタイズメッセージには、負荷分散クラスターに参加するワーカーノードからの応答をプロキシーサーバーがリッスンしている IP アドレスとポート番号が含まれます。

関連情報

- **mod_proxy_cluster** コネクターモジュール

3.2. MOD_PROXY_CLUSTER のインストールとアップグレード

Red Hat JBoss Core Services (JBCS) と Red Hat Enterprise Linux (RHEL) は、Apache HTTP Server の個別のディストリビューションを提供します。インストールする Apache HTTP Server ディストリビューションに応じて、**mod_proxy_cluster** コネクターが自動または手動のどちらでインストールされるか決まります。インストールされている Apache HTTP サーバーのディストリビューションにより、**mod_proxy_cluster** モジュールと設定ファイルのインストールパスも異なります。



注記

JBCS Apache HTTP サーバーでは、すべてのサポート対象オペレーティングシステムで **mod_proxy_cluster** の使用がサポートされています。RHEL Apache HTTP Server では、RHEL 9 でのみ **mod_proxy_cluster** の使用がサポートされています。

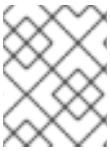
3.2.1. JBCS Apache HTTP Server を使用する場合の mod_proxy_cluster のインストール

JBCS インストールの Apache HTTP Server 部分では、**mod_proxy_cluster** モジュールが自動的にインストールされます。

Red Hat JBoss Core Services Apache HTTP Server インストールガイドの手順に従い、オペレーティングシステム用の最新の JBCS Apache HTTP Server リリースをインストールまたはアップグレードできます。詳細は、**関連情報** セクションを参照してください。

JBCS Apache HTTP サーバーを使用する場合は、**mod_proxy_cluster** のインストールに関する次のガイドラインを考慮してください。

- **mod_proxy_cluster.so**、**mod_cluster_slotmem.so**、**mod_manager.so**、**mod_advertise.so** モジュールは、**JBCS_HOME/httpd/modules** ディレクトリーにインストールされます。
- **mod_proxy_cluster.conf.sample** 設定ファイルは、**JBCS_HOME/httpd/conf.d** ディレクトリーにあります。
- **mod_proxy_cluster.conf.sample** ファイルには、**mod_proxy_cluster** モジュールの **LoadModule** ディレクティブが含まれています。



注記

JBCS_HOME は、JBCS インストールの最上位ディレクトリー (**/opt/jbcs-httpd24-2.4**) を表します。

関連情報

- アーカイブファイルから RHEL に JBCS Apache HTTP サーバーをインストールする
- RPM パッケージから RHEL 7 または RHEL 8 に JBCS Apache HTTP サーバーをインストールする
- Windows Server に JBCS Apache HTTP サーバーをインストールする

3.2.2. 以前の JBCS リリースからの mod_proxy_cluster のアップグレード

JBCS が 2.4.37 以前のリリースで提供していた **mod_cluster-native** パッケージは、2.4.51 リリースでは **mod_proxy_cluster** という名前に変更されました。この変更の一環として、以前のリリースで利用可能だった **mod_cluster.conf** ファイルの名前も、2.4.51 リリースで **mod_proxy_cluster.conf** に変更されています。JBCS は、JBCS をアーカイブファイルと RPM パッケージのどちらからインストールしたかに応じた方法で、既存の **mod_proxy_cluster** 設定をアップグレードします。

RPM パッケージからインストールした場合の mod_proxy_cluster 設定のアップグレード
アップグレードする既存の JBCS インストールが RHEL 7 または RHEL 8 上の RPM パッケージからインストールされている場合は、次のガイドラインを考慮してください。

- JBCS 2.4.37 以前からアップグレードする場合、JBCS はアップグレード中に既存の **mod_cluster.conf** ファイルを保持します。この場合、アップグレードされた JBCS 2.4.51 デプロイメントには、既存の **mod_cluster.conf** ファイルとデフォルトの **mod_proxy_cluster.conf** ファイルの両方が含まれます。その後、**mod_proxy_cluster.conf** を使用するように移行する場合は、セットアップ要件に合わせてデフォルトの **mod_proxy_cluster.conf** ファイルを手動で更新できます。
- JBCS 2.4.51 の既存リリースからアップグレードする場合、JBCS はアップグレード中に既存の **mod_proxy_cluster.conf** ファイルを保持します。この場合、アップグレードされた BCS 2.4.51 デプロイメントには、既存の **mod_proxy_cluster.conf** ファイルとデフォルトの **mod_proxy_cluster.conf.rpmnew** ファイルの両方が含まれます。

アーカイブファイルからインストールした場合の mod_proxy_cluster 設定のアップグレード
アーカイブファイルからインストールした既存の JBCS インストールをアップグレードする場合は、次のガイドラインを考慮してください。

- JBCS 2.4.37 以前からアップグレードする場合は、2.4.51 アーカイブファイルを展開する以外に必要な操作はありません。JBCS 2.4.51 にはデフォルトの **mod_cluster.conf** ファイルが含まれていないため、製品のアップグレード中も既存の **mod_cluster.conf** ファイルがそのまま残ります。この場合、アップグレードされた JBCS 2.4.51 デプロイメントには、既存の **mod_cluster.conf** ファイルとデフォルトの **mod_proxy_cluster.conf** ファイルの両方が含まれます。その後、**mod_proxy_cluster.conf** を使用するように移行する場合は、セットアップ要件に合わせてデフォルトの **mod_proxy_cluster.conf** ファイルを手動で更新できます。
- JBCS 2.4.51 の既存リリースからアップグレードする場合は、最初に既存の **mod_proxy_cluster.conf** ファイルを一時的な場所にコピーする必要があります。JBCS 2.4.51 にはデフォルトの **mod_proxy_cluster.conf** ファイルが含まれており、製品のアップグレード中に既存の **mod_proxy_cluster.conf** ファイルが自動的に上書きされます。最新の 2.4.51 アーカイブファイルを抽出したら、既存の **mod_proxy_cluster.conf** ファイルのバックアップを正しい場所にコピーして、デフォルトファイルを上書きできます。

3.2.3. Application Streams を使用する場合の mod_proxy_cluster のインストール

Application Streams を使用して RPM パッケージから Apache HTTP Server の RHEL 9 ディストリビューションをインストールする場合、RHEL は **mod_proxy_cluster** パッケージを自動的にインストールしません。この状況で **mod_proxy_cluster** コネクターを使用する場合は、**mod_proxy_cluster** パッケージを手動でインストールする必要があります。

前提条件

- Application Streams を使用して、Apache HTTP サーバーを RHEL 9 にインストールている。

手順

- root ユーザーとして以下のコマンドを実行します。

```
# dnf install mod_proxy_cluster
```

検証

- **mod_proxy_cluster** パッケージが正常にインストールされたことを確認するには、次のコマンドを入力します。

```
# rpm -q mod_proxy_cluster
```

  前述のコマンドは、バージョンとプラットフォームの情報を含む、インストールされたパッケージの完全名を出力します。

RHEL Application Streams を使用する場合は、**mod_proxy_cluster** のインストールに関する次のガイドラインを考慮してください。

- **mod_proxy_cluster.so**、**mod_cluster_slotmem.so**、**mod_manager.so**、**mod_advertise.so** モジュールは、**/usr/lib64/httpd/modules** ディレクトリーにインストールされます。
- **mod_proxy_cluster.conf.sample** 設定ファイルは、**/etc/httpd/conf.d** ディレクトリーにあります。
- **mod_proxy_cluster.conf.sample** ファイルには、**mod_proxy_cluster** モジュールの **LoadModule** ディレクティブが含まれています。

関連情報

- Application Streams (AppStream)
- DNF ツールを使用したソフトウェアの管理

3.3. MOD_PROXY_CLUSTER を使用する場合の APACHE HTTP SERVER 負荷分散設定

Apache HTTP Server 2.1 以降のバージョンでは、**mod_proxy_cluster** はデフォルトで Apache HTTP Server 用に正しく設定されています。カスタム設定の設定について詳しくは、基本プロキシーサーバーの設定を参照してください。

mod_proxy_cluster のサンプル設定ファイル

mod_proxy_cluster を Red Hat JBoss Core Services (JBCS) 経由でインストールしたか、Red Hat Enterprise Linux (RHEL) Application Streams を使用してインストールしたかに応じて、次のガイドラインを考慮してください。

- JBCS の場合、**JBCS_HOME/httpd/conf.d/** ディレクトリーに **mod_proxy_cluster** のサンプル設定ファイルがあります。
- RHEL の場合、**/etc/httpd/conf.d/** ディレクトリーに **mod_proxy_cluster** のサンプル設定ファイルがあります。

mod_proxy_cluster のサンプル設定ファイルの名前は **mod_proxy_cluster.conf.sample** です。独自の設定ファイルを作成する代わりにこれらのサンプルを使用する場合は、**.sample** 拡張子を削除し、必要に応じてファイルの内容を変更します。

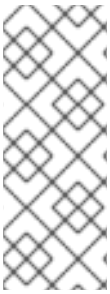
注記

また、Red Hat Customer Portal の Load Balancer Configuration ツールを使用して、**mod_proxy_cluster** および Tomcat ワーカーノードに最適な設定テンプレートをすばやく生成することもできます。Apache HTTP Server 2.4.51 のロードバランサー設定ツールを使用する場合は、Apache バージョンに **2.4.x** を、バックエンド設定に **Tomcat/JWS** を選択してください。

mod_proxy_cluster の使用に関するガイドライン

mod_proxy_cluster コネクターを使用する場合は、次のガイドラインを考慮してください。

- **mod_proxy_cluster** コネクターを使用する場合は、**mod_proxy** モジュールを有効にして、**mod_proxy_balancer** モジュールを無効にする必要があります。
- **mod_proxy_cluster** で Apache JServ Protocol (AJP) を使用する場合は、 **proxy_ajp_module** を有効にする必要があります。
- AJPSecret **your_secret** を使用して、AJP バックエンドのシークレットを指定します。**your_secret** がバックエンドで設定された値に対応していない場合には、このバックエンドでは、プロキシー経由で送信される全要求に対して **503** エラー応答を送信します。



注記

Red Hat JBoss Core Services 2.4.51 では、バックエンドの websocket サーバーに対する接続がアップグレードされていない場合に、その接続のトンネリングはサポートされません。つまり、**mod_proxy_wstunnel** モジュールの **ProxyPass** ディレクティブを設定するときに、アップグレードパラメーターが **NONE** に設定されていないことを確認する必要があります。**mod_proxy_wstunnel** の詳細は、Apache のドキュメント を参照してください。

3.3.1. 基本のプロキシーサーバーの設定

Web クライアントとバックエンド Web サーバーの間で要求と応答を転送するプロキシーサーバーとして機能するように Apache HTTP Server を設定できます。バックエンドのワーカーノードから接続要求と応答を受信するには、プロキシーサーバーリスナーを設定する必要があります。**mod_proxy_cluster** を使用する負荷分散プロキシーサーバーを設定する場合は、管理チャネル用の仮想ホストも設定する必要があります。

前提条件

- Apache HTTP サーバーがインストール されている。

- Application Streams を使用して Apache HTTP Server の RHEL ディストリビューションをインストールした場合は、**mod_proxy_cluster を手動でインストール** している。
- プロキシーサーバーリスナーに指定するポートは、受信 TCP 接続用に開いている必要があります。

手順

1. Apache HTTP Server 設定ディレクトリーに移動します。
   - JBCS Apache HTTP Server を使用している場合は、**JBCS_HOME/httpd/conf.d** ディレクトリーに移動します。
   - RHEL Apache HTTP Server を使用している場合は、**/etc/httpd/conf.d** ディレクトリーに移動します。
2. **mod_proxy_cluster.conf** ファイルを開きます。
3. プロキシーサーバーの **Listen** ディレクティブを作成するには、**mod_proxy_cluster.conf** ファイルに次の行を入力します。

```
Listen IP_ADDRESS:PORT_NUMBER
```

注記

上記の例では、プロキシーサーバーがワーカーノードと通信するために使うサーバーネットワークインターフェイスのアドレスに **IP_ADDRESS** をプロキシーサーバーがワーカーノードと通信するのに使うサーバーネットワークインターフェイスのアドレスに、をプロキシーサーバーがリッスンするポートに置き換えてください。**PORT_NUMBER** をプロキシーサーバーがリッスンするポートに置き換えます。

着信 TCP 接続用にポートが開いていることを確認します。

4. 仮想ホストを作成するには、**mod_proxy_cluster.conf** ファイルに次の詳細を入力します。

```
<VirtualHost IP_ADDRESS:PORT_NUMBER>

  <Directory />
    Require ip IP_ADDRESS
  </Directory>

  KeepAliveTimeout 60
  MaxKeepAliveRequests 0

  ManagerBalancerName mycluster
  AdvertiseFrequency 5
  EnableMCPMReceive On

</VirtualHost>
```

注記

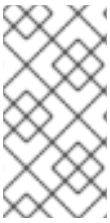
前の例で、**IP_ADDRESS** と **PORT_NUMBER** を、**Listen** ディレクティブに指定したサーバーネットワークインターフェイスのアドレスとポート番号に置き換えます。

このアドレスとポートの組み合わせは、**mod_proxy_cluster** 管理メッセージにのみ使用されます。このアドレスとポートの組み合わせは、一般的なトラフィックには使用されません。

Apache HTTP Server サービスの開始の詳細については、Red Hat JBoss Core Services Apache HTTP Server インストールガイド を参照してください。

3.3.1.1. サーバー広告の無効化

プロキシーサーバーは、UDP マルチキャストを使用して自身をアドバタイズします。**AdvertiseFrequency** ディレクティブは、デフォルトでサーバー通知メッセージを 10 秒ごとに送信するようにサーバーに指示します。サーバー通知メッセージには、**VirtualHost** 定義で指定した **IP_ADDRESS** と **PORT_NUMBER** が含まれています。サーバーアドバタイズに応答するように設定されたワーカーノードは、この情報を使用してプロキシーサーバーに登録されます。ワーカーノードがプロキシーサーバーに登録されないようにする場合は、オプションでサーバーアドバタイズメントを無効にすることができます。



注記

プロキシーサーバーとワーカーノードの間で UDP マルチキャストが使用可能な場合、サーバーアドバタイズメントは、プロキシーサーバーでさらに設定する必要なく、ワーカーノードを追加します。サーバーアドバタイズには、ワーカーノードでの最小限の設定のみが必要です。

前提条件

- 基本的なプロキシーサーバーを設定しました。

手順

1. Apache HTTP Server 設定ディレクトリーに移動します。
   - JBCS Apache HTTP Server を使用している場合は、**JBCS_HOME/httpd/conf.d** ディレクトリーに移動します。
   - RHEL Apache HTTP Server を使用している場合は、**/etc/httpd/conf.d** ディレクトリーに移動します。
2. **mod_proxy_cluster.conf** ファイルを開きます。
3. 次のディレクティブを **VirtualHost** 定義に追加します。

```
ServerAdvertise Off
```

注記

サーバーのアドバタイズが無効になっている場合や、UDP マルチキャストがプロキシーサーバーとワーカーノードの間のネットワークで利用できない場合、ワーカーノードをプロキシーサーバーの静的リストで設定します。いずれの場合も、ワーカーノードのリストを使用してプロキシーサーバーを設定する必要はありません。

関連情報

- プロキシーサーバーの静的リストを使用したワーカーノードの設定

3.3.1.2. ワーカーノードの詳細のログ記録

mod_proxy_cluster を使用する負荷分散プロキシーサーバーを設定する場合、必要に応じて、要求を処理する各ワーカーノードの詳細をログに記録するように Apache HTTP Server を設定できます。ワーカーノードの詳細をログに記録すると、ロードバランサーのトラブルシューティングが必要な場合に役立ちます。

前提条件

- 基本的なプロキシーサーバーを設定しました。

手順

1. Apache HTTP Server 設定ディレクトリーに移動します。
   - JBCS Apache HTTP Server を使用している場合は、**JBCS_HOME/httpd/conf.d** ディレクトリーに移動します。
   - RHEL Apache HTTP Server を使用している場合は、**/etc/httpd/conf.d** ディレクトリーに移動します。
2. **mod_proxy_cluster.conf** ファイルを開きます。
3. 次の詳細を Apache HTTP Server の **LogFormat** ディレクティブに追加します。

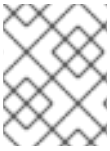
```
%{BALANCER_NAME}e ::
The name of the balancer that served the request.

%{BALANCER_WORKER_NAME}e ::
The name of the worker node that served the request.
```

関連情報

- ログファイルに関する Apache HTTP Server のドキュメント

3.3.2. mod_proxy_cluster での JBoss Web Server ワーカーノードの設定

mod_proxy_cluster を使用すると、バックエンドワーカーノードを非クラスターモードのみで動作する JBoss Web ServerTomcat サービスとして設定できます。この状況で **mod_proxy_cluster** が負荷分散係数の計算で同時に使用できる負荷メトリクスは 1 つだけです。



注記

JBoss Web Server ワーカーノードは、**mod_proxy_cluster** 機能のサブセットのみをサポートします。完全な **mod_proxy_cluster** 機能は JBoss EAP で利用できます。

前提条件

- **mod_proxy_cluster** のプロキシーおよびプロキシー検出設定属性 に精通している。

手順

1. JBoss Web Server にリスナーを追加するには、**JWS_HOME/tomcat <VERSION>/conf/server.xml** ファイルで、他の **Listener** 要素の下に次の **Listener** 要素を追加します。

```
<Listener
className="org.jboss.modcluster.container.catalina.standalone.ModClusterListener"
advertise="true" stickySession="true" stickySessionForce="false"
stickySessionRemove="true" />
```

2. ワーカーノードに一意のアイデンティティーを与えるには、**JWS_HOME/tomcat <VERSION>/conf/server.xml** ファイルで、**jvmRoute** 属性と値を **Engine** 要素に追加します。

```
<Engine name="Catalina" defaultHost="localhost" jvmRoute="worker01">
```

3. **STATUS MCMP** メッセージの頻度を設定するには、**org.jboss.modcluster.container.catalina.status-frequency** Java システムプロパティーを変更します。
以下に例を示します。

```
-Dorg.jboss.modcluster.container.catalina.status-frequency=6
```

注記

JBoss Web Server は、現在の負荷ステータスを含むステータスメッセージを Apache HTTP Server バランサーに定期的に送信します。これらのメッセージのデフォルトの頻度は 10 秒です。数百のワーカーノードがある場合、**STATUS MCMP** メッセージによって Apache HTTP Server ネットワークのトラフィック輻輳が増える可能性があります。

MCMP メッセージ頻度を設定するには、**org.jboss.modcluster.container.catalina.status-frequency** Java システムプロパティーを変更します。デフォルトでは、プロパティーは秒単位で指定された値を 10 倍して受け入れます。たとえば、プロパティーを **1** に設定すると 10 秒になります。前の例では、プロパティーは **6** に設定されており、これは 60 秒を意味します。

4. オプション: プロキシーサーバーのアドバタイズ用にファイアウォールを設定するには、次のいずれかの手順を実行して、ワーカーノードのファイアウォールで UDP 接続用のポート **23364** を開きます。

- RHEL の場合:

```
firewall-cmd --permanent --zone=public --add-port=23364/udp
```

- PowerShell を使用する Windows Server の場合:

```
Start-Process "$psHome\powershell.exe" -Verb Runas -ArgumentList '-command "NetSh
Advfirewall firewall add rule name="UDP Port 23364" dir=in  action=allow protocol=UDP
localport=23364"'
Start-Process "$psHome\powershell.exe" -Verb Runas -ArgumentList '-command "NetSh
Advfirewall firewall add rule name="UDP Port 23364" dir=out action=allow protocol=UDP
localport=23364"'
```

注記

プロキシーサーバーが **mod_proxy_cluster** を使用する場合、プロキシーサーバーは UDP マルチキャストを使用して自身をアドバタイズできます。ほとんどのオペレーティングシステムのファイアウォールは、デフォルトでサーバーアドバタイズメント機能をブロックします。サーバーアドバタイズメントを有効にしてこれらのマルチキャストメッセージを受信するには、前述の例に示すように、ワーカーノードのファイアウォールで UDP 接続用にポート **23364** を開くことができます。

3.3.3. プロキシーサーバーの静的リストを操作するためのワーカーノードの設定

サーバーアドバタイズメントにより、ワーカーノードがプロキシーサーバーを動的に検出して登録できるようになります。UDP マルチキャストが利用できない場合、またはサーバーアドバタイズメントが無効になっている場合は、プロキシーサーバーのアドレスとポートの静的リストを使用して JBoss Web Server ワーカーノードを設定する必要があります。

前提条件

- JBoss Web Server ワーカーノードを設定しました。
- Tomcat のプロキシー設定パラメーターについて 理解しています。

手順

1. **JWS_HOME/tomcat<VERSION>/conf/server.xml** ファイルを開きます。
2. **mod_proxy_cluster** リスナーを定義して動的プロキシー検出を無効にするには、**ModClusterListener** の **Listener** 要素を追加または変更します。
   以下に例を示します。

```
<Listener
className="org.jboss.modcluster.container.catalina.standalone.ModClusterListener"
advertise="false" stickySession="true" stickySessionForce="false"
stickySessionRemove="true"/>
```

注記

advertise プロパティーを **false** に設定していることを確認してください。

3. 静的プロキシーサーバーリストを作成するには、プロキシーのコンマ区切りリストを、**IP_ADDRESS:PORT,IP_ADDRESS:PORT** の形式で追加して、**proxyList** プロパティーを更新します。

以下に例を示します。

```
<Listener
className="org.jboss.modcluster.container.catalina.standalone.ModClusterListener"
advertise="false" stickySession="true" stickySessionForce="false"
stickySessionRemove="true" proxyList="10.33.144.3:6666,10.33.144.1:6666"/>
```

3.4. MOD_PROXY_CLUSTER の 文字制限

mod_proxy_cluster モジュールは共有メモリーを使用してノードの説明を保持します。共有メモリーは Apache HTTP Server の起動時に作成され、各項目の構造は固定されます。

プロキシーサーバーとワーカーノードのプロパティーを定義するときは、次の文字制限を必ず守ってください。

プロパティー	最大文字数制限	説明
エイリアスの長さ	100 文字	エイリアスは、それぞれの仮想ホストのネットワーク名に対応します。名前は **Host** 要素で定義されます。
コンテキストの長さ	40 文字	たとえば、**myapp.war** が **/myapp** にデプロイされている場合、**/myapp** がコンテキストに含まれます。
バランサー名の長さ	40 文字	**<Listener>** 要素の **balancer**。
JVMRoute 文字列の長さ	80 文字	**<Engine>** 要素の **JVMRoute**。
ドメイン名の長さ	20 文字	**<Listener>** 要素の **loadBalancingGroup**。
ノードのホスト名の長さ	64 文字	これは、**<Connector>** 要素のホスト名アドレスです。
ノードのポート長	7 文字	これは、**<Connector>** 要素のポートプロパティーです。たとえば、**8009** は 4 文字です。
ノードのスキームの長さ	6 文字	これはコネクターのプロトコルです。可能な値は **http**、**https**、および **ajp** です。
Cookie 名の長さ	30 文字	これは、セッション ID のヘッダー Cookie 名です。デフォルト値は、**org.apache.catalina.Globals.SESSION_COOKIE_NAME** プロパティーに基づく **JSESSIONID** です。

プロパティー	最大文字数制限	説明
パス名の長さ	30 文字	これは、セッション ID のパラメーター名です。デフォルト値は、**org.apache.catalina.Globals.SESSION_PARAMETER_NAME** プロパティーに基づく **JSESSIONID** です。
Session ID length	120 文字	セッション ID の形式は、**BE81FAA969BF64C8EC2B6600457EAAAA.node01** です。

第4章 MOD_PROXY_CLUSTER による負荷分散の設定例

Red Hat Enterprise Linux システムでの負荷分散に **mod_proxy_cluster** コネクターを使用するように JBCS を設定できます。

mod_proxy_cluster を使用する負荷分散ソリューションを設定する場合は、次のタスクを実行する必要があります。

1. JBCS をプロキシーサーバーとして設定 します。
2. Tomcat ワーカーノードを設定 します。
3. iptables ファイアウォールルールを定義 します。

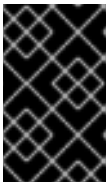
4.1. JBCS をプロキシーサーバーとして設定

mod_proxy_cluster を使用するように JBCS を設定する場合は、 **mod_proxy_cluster.conf** ファイルに設定の詳細を指定して、JBCS をプロキシーサーバーとしてセットアップする必要があります。

手順

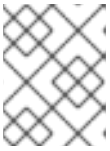
1. **JBCS_HOME/httpd/conf.d/** ディレクトリーに移動します。
2. **mod_proxy_cluster.conf** という名前のファイルを作成します。
3. 次の設定の詳細を入力します。

```
LoadModule proxy_cluster_module modules/mod_proxy_cluster.so
LoadModule cluster_slotmem_module modules/mod_cluster_slotmem.so
LoadModule manager_module modules/mod_manager.so
LoadModule advertise_module modules/mod_advertise.so

MemManagerFile cache/mod_proxy_cluster

<IfModule manager_module>
 Listen 6666
 <VirtualHost *:6666>
  <Directory />
   Require ip 127.0.0.1
  </Directory>
  ServerAdvertise on
  EnableMCPMReceive
  <Location /mod_cluster_manager>
   SetHandler mod_cluster-manager
   Require ip 127.0.0.1
  </Location>
 </VirtualHost>
</IfModule>
```



重要

前の例に示したとおり、**mod_proxy_cluster** パッケージでは、**conf.d** ファイルの **MemManagerFile** ディレクティブを **cache/mod_proxy_cluster** に設定する必要があります。



注記

前述の例は、**localhost** をリッスンするプロキシーサーバーとして JBCS を設定する方法を示しています。

4.2. TOMCAT ワーカーノードの設定

mod_proxy_cluster を使用するように JBCS を設定する場合、 **server.xml** ファイルに **Listener** 要素を追加して、Tomcat ワーカーノードを設定する必要があります。

前提条件

- JBCS をプロキシーサーバーとして設定 している。

手順

1. **JWS_HOME/tomcat<VERSION>/conf/server.xml** ファイルを開きます。
2. 次の **Listener** 要素を追加します。

```
<Listener
className="org.jboss.modcluster.container.catalina.standalone.ModClusterListener"
advertise="true"/>
```

4.3. IPTABLES ファイアウォールルールの定義の例

mod_proxy_cluster を使用するように JBCS を設定する場合は、 **iptables** を使用してファイアウォールルールを定義する必要があります。

前提条件

- Tomcat ワーカーノードを設定しました。

手順

- **iptables** を使用して、一連のファイアウォールルールを定義します。
以下に例を示します。

```
/sbin/iptables -I INPUT 5 -p udp -d 224.0.1.0/24 -j ACCEPT -m comment --comment
"mod_proxy_cluster traffic"
/sbin/iptables -I INPUT 6 -p udp -d 224.0.0.0/4 -j ACCEPT -m comment --comment "JBoss
Cluster traffic"
/sbin/iptables -I INPUT 9 -p udp -s 192.168.1.0/24 -j ACCEPT -m comment --comment
"cluster subnet for inter-node communication"
/sbin/iptables -I INPUT 10 -p tcp -s 192.168.1.0/24 -j ACCEPT -m comment --comment
"cluster subnet for inter-node communication"
/etc/init.d/iptables save
```

注記

前の例は、**192.168.1.0/24** サブネット上のクラスターノードのファイアウォール規則を定義することを示しています。

付録A MOD_PROXY コネクターモジュール

mod_proxy コネクターは、標準の Apache HTTP Server モジュールのセットで設定されます。これらのモジュールにより、Apache HTTP サーバーは、さまざまなタイプのプロトコルを介して Web クライアントとバックエンドサーバーの間で Web トラフィックを送信するためのプロキシー/ゲートウェイとして機能できるようになります。

この付録では、**mod_proxy** コネクターが使用するモジュールについて説明します。

A.1. MOD_PROXY.SO モジュール

mod_proxy.so モジュールは、サーバーが AJP (Apache JServ Protocol)、FTP、CONNECT (SSL 用)、および HTTP プロトコルを介して転送されるデータのプロキシーとして機能できるようにする、標準の Apache HTTP Server モジュールです。**mod_proxy** モジュールには、追加の設定は必要ありません。**mod_proxy** モジュールの識別子は **proxy_module** です。

関連情報

- Apache モジュール mod_proxy

A.2. MOD_PROXY_AJP.SO モジュール

mod_proxy_ajp.so モジュールは、Apache JServ Protocol (AJP) プロキシーのサポートを提供する標準の Apache HTTP Server モジュールです。**mod_proxy_ajp** モジュールを使用することにより、Apache HTTP サーバーは、Web クライアントとバックエンドサーバーの間で AJP リクエストと応答を送信するための仲介者として機能します。AJP は、データ暗号化をサポートしないクリアテキストプロトコルです。

mod_proxy_ajp を使用する場合は、**mod_proxy** モジュールも必要です。**mod_proxy_ajp** モジュールの識別子は **proxy_ajp_module** です。

また、**secret** プロパティーは Tomcat AJP コネクターを使用する場合に必要になります。次のコマンドを使用して、**secret** プロパティーを **ProxyPass** 設定に追加できます。

```
ProxyPass /example/ ajp://localhost:8009/example/ secret=YOUR_AJP_SECRET
```

注記

ロードバランサーに **secret** を設定すると、そのすべてのメンバーがこの **secret** を継承します。

mod_proxy_ajp モジュールは設定ディレクティブを提供しません。

関連情報

- Apache Module mod_proxy_ajp

A.3. MOD_PROXY_HTTP.SO モジュール

mod_proxy_http.so モジュールは、ハイパーテキスト転送プロトコル (HTTP) およびハイパーテキスト転送プロトコルセキュア (HTTPS) プロキシーのサポートを提供する標準の Apache HTTP サーバーモジュールです。**mod_proxy_http** モジュールを使用することにより、Apache HTTP サーバーは、Web

クライアントとバックエンドサーバーの間で HTTP または HTTPS リクエストを転送するための仲介者として機能します。**mod_proxy_http** モジュールは、HTTP/1.1 およびそれ以前のバージョンの HTTP プロトコルをサポートします。

mod_proxy_http を使用する場合は、**mod_proxy** モジュールも必要です。**mod_proxy_http** モジュールの識別子は **proxy_http_module** です。

mod_proxy_http モジュールは設定ディレクティブを提供しません。**mod_proxy_http** モジュールは、**mod_proxy** モジュールの動作を制御する設定とともに、HTTP プロトコルプロバイダーの動作を制御する一連の 環境変数 を使用します。

関連情報

- Apache Module mod_proxy_http

A.4. MOD_PROXY_HTTP2.SO モジュール

mod_proxy_http2.so モジュールは、ハイパーテキスト転送プロトコル 2.0 (HTTP/2) プロキシーのサポートを提供する標準の Apache HTTP サーバーモジュールです。**mod_proxy_http2** モジュールを使用することにより、Apache HTTP サーバーは Web クライアントとバックエンドサーバーの間で HTTP/2 リクエストを転送する仲介者として機能します。

mod_proxy_http2 モジュールは、通信プロトコルとして HTTP/1.1 または HTTP/2 を使用するクライアント要求をサポートします。ただし、**mod_proxy_http2** モジュールでは、Apache HTTP サーバーとバックエンドサーバー間のすべての通信で HTTP/2 のみを使用する必要があります。

同じバックエンド宛先を持つクライアント要求の場合、Apache HTTP サーバーは可能な限り同じ TCP 接続を再利用します。ただし、複数のクライアントリクエストを同じバックエンドに転送する場合でも、Apache HTTP サーバーは HTTP/1.1 クライアントリクエストごとに個別の HTTP/2 プロキシーリクエストを転送します。

mod_proxy_http2 を使用する場合は、**mod_proxy** モジュールも必要です。**mod_proxy_http2** モジュールの識別子は **proxy_http2_module** です。

mod_proxy_http2 モジュールは設定ディレクティブを提供しません。

注記

mod_proxy_http2 モジュールは、コア HTTP/2 エンジン用の **libnghttp2** ライブラリーの使用を必要とする実験的な Apache 機能です。

関連情報

- JBCS Apache HTTP Server の HTTP/2 を有効化する
- Apache Module mod_proxy_http2

付録B MOD_JK コネクターモジュール

Apache Tomcat コネクター **mod_jk** は、Apache Tomcat プロジェクトが提供する Web サーバープラグインです。Apache HTTP サーバーは、**mod_jk** モジュールを使用して、スティッキーセッションを維持し、Apache JServ プロトコル (AJP) 経由で通信しながら、バックエンドサーブレットコンテナーに対する HTTP クライアントリクエストの負荷を分散できます。**mod_jk** モジュールは、JBoss Core Services インストールの Apache HTTP Server に含まれています。

mod_jk モジュールでは、Apache HTTP Server ホスト上に **mod_jk.conf** ファイルと **works.properties** ファイルの両方を作成する必要があります。**mod_jk.conf** ファイルは、**mod_jk.so** モジュールをロードして設定するための設定を指定します。**works.properties** ファイルは、バックエンドワーカーノードの詳細を指定します。**mod_jk** サポートを有効にするには、JWSShortName ホストでいくつかの設定を設定する必要もあります。

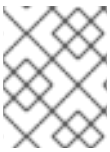
関連情報

- Apache Tomcat Connector (**mod_jk**) を使用した負荷分散
- **mod_jk** の **Workers.properties** ファイル

付録C MOD_PROXY_CLUSTER コネクターモジュール

mod_proxy_cluster コネクターは、JBoss mod_cluster コミュニティープロジェクトが元々開発したテクノロジーに基づく、設定を削減したインテリジェントな負荷分散ソリューションです。**mod_proxy_cluster** コネクターを使用すると、Apache HTTP サーバーが、JBoss Web Server または JBoss EAP ホスト上で実行されているバックエンドアプリケーションにトラフィックを転送するための高度なロードバランサーとして機能できるようになります。**mod_proxy_cluster** コネクターは、**mod_jk** のすべての機能と、リアルタイムの負荷分散計算、アプリケーションライフサイクル制御、自動プロキシー検出、複数プロトコルサポートなどの追加機能を提供します。

この付録では、**mod_proxy_cluster** コネクターが使用するモジュールについて説明します。



注記

mod_proxy_cluster コネクターは、**ProxyIOBufferSize** などの **mod_proxy** の設定可能なディレクティブを使用して設定できます。

C.1. MOD_MANAGER.SO モジュールとディレクティブ

クラスターマネージャーモジュール **mod_manager.so** は、ワーカーノードの登録、ワーカーノードの負荷データ、およびワーカーノードのアプリケーションのライフサイクルイベントなどのノードからメッセージを受信および確認します。

```
LoadModule manager_module modules/mod_manager.so
```

mod_manager.so の設定可能なディレクティブ

<VirtualHost> 要素の設定可能なディレクティブは以下のとおりです。

EnableMCPMReceive

VirtualHost が mod_cluster management protocol (MCMP) メッセージを受信できるようにします。**mod_proxy_cluster** が正常に動作できるように、Apache HTTP Server 設定に **EnableMCPMReceive** ディレクティブを 1 つ追加します。**advertise** が設定された場所の **VirtualHost** 設定に **EnableMCPMReceive** を追加する必要があります。

MaxMCMPMaxMessSize

MCMP メッセージの最大サイズを定義します。デフォルト値は、他の **Max** ディレクティブから計算されます。最小値は **1024** です。

AllowDisplay

mod_cluster-manager メインページで追加表示を切り替えます。デフォルト値は **off** で、**mod_cluster-manager** メインページにはバージョン情報のみが表示されます。

AllowCmd

mod_cluster-manager URL を使用してコマンドのパーミッションを切り替えます。デフォルト値は **on** で、コマンドを許可します。

ReduceDisplay

mod_cluster-manager ページに表示される情報の縮減を切り替えます。情報を減らすと、ページでより多くのノードを表示できます。デフォルト値は **off** で、利用可能な情報をすべて表示することができます。

MemManagerFile

mod_manager が設定の詳細を保存するファイルの場所を定義します。mod_manager は、共有メモリーおよびロックファイルに生成された鍵にもこの場所を使用します。**絶対パス名である必要があ**

ります。NFS 共有ではなく、ローカルドライブ上のこのパスを使用することが推奨されます。デフォルト値は **/logs/** です。

Maxcontext

mod_proxy_cluster が使用するコンテキストの最大数。デフォルト値は **100** です。

Maxnode

mod_proxy_cluster が使用するワーカーノードの最大数。デフォルト値は **20** です。

Maxhost

mod_proxy_cluster が使用するホスト (エイリアス) の最大数。これは、ロードバランサーの最大数にもなります。デフォルト値は **20** です。

Maxsessionid

保存されたアクティブなセッション識別子の最大数。セッションから情報が 5 分間受信されない場合、セッションは非アクティブとみなされます。これはデモおよびデバッグの目的のみで使用されます。デフォルト値は **0** で、このロジックを無効にします。

ManagerBalancerName

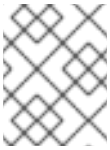
ワーカーノードがロードバランサー名を指定しない場合に使用するロードバランサーの名前。デフォルト値は **mycluster** です。

PersistSlots

on に設定された場合、ノード、エイリアス、およびコンテキストはファイルに永続化されます。デフォルト値は **off** です。

CheckNonce

on に設定された場合、セッション識別子をチェックして、一意で、これまでに発生していないことを確認します。デフォルトは **on** です。



注記

このディレクティブを **off** に設定すると、サーバーがリプレイ攻撃に対して脆弱になります。

SetHandler mod_cluster-manager

ハンドラーを定義して、クラスター内のワーカーノードについての情報を表示します。これは、**Location** 要素で定義されます。

```
<Location $LOCATION>
  SetHandler mod_cluster-manager
  Require ip 127.0.0.1
</Location>
```

ここでは、**$LOCATION** も **mod_cluster_manager** として定義されていました。

ブラウザーの **Location** 要素で定義されている **$LOCATION** にアクセスする場合は、次のガイドラインに留意してください。

- **Transferred** は、ワーカーノードに送信された POST データに一致します。
- **Connected** は、このステータスページが要求されたときに処理されたリクエストの数に対応します。
- **Sessions** はアクティブなセッションの数に対応します。**Maxsessionid** が **0** の場合、このフィールドは存在しません。

C.2. MOD_PROXY_CLUSTER.SO モジュールとディレクティブ

プロキシーロードバランサーモジュール **mod_proxy_cluster.so** は、クラスターノードへの要求のルーティングを処理します。プロキシーバランサーは、クラスターのアプリケーションの場所、各クラスターノードの現在の状態、およびセッション ID (要求が確立されたセッションの一部である場合) に基づいて要求を転送するために適切なノードを選択します。

```
LoadModule proxy_cluster_module modules/mod_proxy_cluster.so
```

mod_proxy_cluster.so の設定可能なディレクティブ

<VirtualHost> 要素に以下のディレクティブを設定して、負荷分散の動作を変更することもできます。

CreateBalancers

ロードバランサーが Apache HTTP Server の仮想ホストでどのように作成されるかを定義します。**CreateBalancers** では、以下の値が使用できます。

- **0**: Apache HTTP Server で定義されたすべての仮想ホストにロードバランサーを作成します。**ProxyPass** ディレクティブでロードバランサーを設定するのを忘れないようにしてください。
- **1**: バランサーを作成しません。この値を使用する場合は、**ProxyPass** または **ProxyPassMatch** にロードバランサー名を定義する必要もあります。
- **2**: メインサーバーのみ作成します。これは **CreateBalancers** のデフォルト値です。

UseAlias

定義された **Alias** が **ServerName** に対応するかどうかを定義します。**UseAlias** については、以下の値有効です。

- **0**: ワーカーノードのエイリアス情報を無視します。これは **UseAlias** のデフォルト値です。
- **1**: 定義されたエイリアスがワーカーノードのサーバー名に対応していることを確認します。

LBstatusRecalTime

ワーカーノードのステータスを計算するプロキシーの間隔 (秒単位) を定義します。デフォルトの間隔は **5** 秒です。

ProxyPassMatch; ProxyPass

ProxyPass は、リモートサーバーをローカルサーバーの名前空間にマッピングします。ローカルサーバーに **http://local.com/** のアドレスがある場合、**ProxyPass** ディレクティブは **http://local.com/requested/file1** のローカルリクエストを **http://worker.local.com/file1** のプロキシーリクエストに変換します。

```
ProxyPass /requested/ http://worker.local.com/
```

ProxyPassMatch は正規表現を使用して、プロキシーされた URL が適用されるローカルパスを照合します。

いずれかのディレクティブで、**!** は指定したパスがローカルであることを示します。そのパスのリクエストはリモートサーバーにルーティングしないでください。たとえば、以下のディレクティブは **gif** ファイルをローカルで提供するように指定します。

```
ProxyPassMatch ^(/.*\.gif)$ !
```

UseNocanon

元の URL パスを変更せずにバックエンドに転送するかどうかを定義します。
デフォルト値は **Off** です。**UseNocanon** ディレクティブが **Off** に設定されている場合、プロキシーは変更された URL をバックエンドに転送できます。ただし、クライアントが要求した元の URL パスをバックエンドアプリケーションが想定している場合は、変更された URL パスが原因で予期しない問題が発生する可能性があります。

UseNocanon ディレクティブを **On** に設定すると、プロキシーは元の URL パスを変更せずにバックエンドに転送できます。この場合、プロキシーの動作は、**mod_proxy_cluster.conf** ファイルで要求された URL のコンテキストと **ProxyPass** ディレクティブも定義しているかどうかによって異なります。コンテキストは、**仮想ホスト定義** とも呼ばれます。

UseNocanon ディレクティブを **On** に設定する場合は、次のガイドラインを考慮してください。

- 要求された URL のコンテキストは定義したが、その URL の **ProxyPass** ディレクティブは定義していない場合、プロキシーは **UseNocanon** ディレクティブを使用します。
- 要求された URL に対してコンテキストと **ProxyPass** ディレクティブの両方を定義し、**ProxyPass** ディレクティブに **nocanon** フラグが含まれている場合、プロキシーは **nocanon** フラグを使用し、**UseNocanon** ディレクティブを無視します。
- 要求された URL に対してコンテキストと **ProxyPass** ディレクティブの両方を定義し、**ProxyPass** ディレクティブで **nocanon** フラグが除外されている場合、プロキシーは **UseNocanon** ディレクティブを無視します。

注記

要求された URL のコンテキストを定義していない場合、**mod_proxy_cluster** は **404** エラーを返します。

C.3. MOD_ADVERTISE.SO モジュールとディレクティブ

mod_advertise.so は、UDP マルチキャストメッセージを介してプロキシーサーバーの存在をブロードキャストします。サーバーのアドバタイズメッセージには、プロキシーが負荷分散クラスターに参加するノードからの応答をリッスンする IP アドレスとポート番号が含まれます。

mod_advertise モジュールは、**VirtualHost** 要素で **mod_manager** モジュールと共に定義する必要があります。次の例では、**mod_advertise** モジュールの識別子は **advertise_module** です。

```
LoadModule advertise_module modules/mod_advertise.so
```

mod_advertise.so の設定可能なディレクティブ

mod_advertise モジュールは、次のディレクティブを使用して設定できます。

ServerAdvertise

アドバタイズメカニズムの使用方法を定義します。
デフォルト値は **Off** です。**Off** に設定すると、プロキシーはその場所を公開しません。

On に設定された場合、アドバタイズメカニズムを使用して、ワーカーノードがこのプロキシーにステータス情報を送信するように指示します。以下の構文でホスト名およびポートを指定することもできます: **ServerAdvertise On http://HOSTNAME:PORT/**これは、名前ベースの仮想ホストを使用する場合や、仮想ホストが定義されていない場合にのみ必要です。

AdvertiseGroup

アドバタイズするマルチキャストアドレスを定義します。構文は **AdvertiseGroup ADDRESS:PORT** です。ここでは、**ADDRESS** が **AdvertiseGroupAddress** に一致し、**PORT** がワーカーノードの **AdvertisePort** に一致している必要があります。
ワーカーノードが JBoss EAP ベースで、起動時に **-u** スイッチが使用される場合、デフォルト値 **AdvertiseGroupAddress** は **-u** スイッチ経由で渡されます。

デフォルト値は **224.0.1.105:23364** です。ポートが指定されていない場合、ポートはデフォルトで **23364** に設定されます。

AdvertiseFrequency

IP アドレスとポートをアドバタイズするマルチキャストメッセージの間隔 (秒単位)。デフォルト値は **10** です。

AdvertiseSecurityKey

Apache HTTP Server で **mod_proxy_cluster** を識別するために使用される文字列を定義します。デフォルトでは、このディレクティブは設定されず、情報は送信されません。

AdvertiseManagerUrl

ワーカーノードが情報をプロキシーサーバーに送信するために使用する URL を定義します。デフォルトでは、このディレクティブは設定されず、情報は送信されません。

AdvertiseBindAddress

マルチキャストメッセージを送信するアドレスとポートを定義します。構文は **AdvertiseBindAddress ADDRESS:PORT** です。これにより、複数の IP アドレスを持つマシンにアドレスを指定できます。デフォルト値は **0.0.0.0:23364** です。

C.4. MOD_CLUSTER_SLOTMEM.SO モジュール

mod_cluster_slotmem.so モジュールは、データセットがスロットとして編成されている共有メモリーセグメントを作成してアクセスするための共有メモリープロバイダーです。

mod_cluster_slotmem モジュールには、設定ディレクティブは必要ありません。

C.5. 関連情報 (または次の手順)

- **mod_proxy_cluster** のワーカーノード設定リファレンス

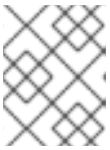
付録D MOD_JK の WORKERS.PROPERTIES ファイル

mod_jk コネクターを使用する場合は、Apache HTTP Server ホスト上に **JBCS_HOME/httpd/conf/workers.properties** ファイルを作成して、バックエンドワーカーノードを定義する必要があります。ワーカーノードは、**mod_jk** ロードバランサーにマップできるサーブレットコンテナーです。**works.properties** ファイルは、サーブレットコンテナーの場所と、これらのサーブレットコンテナー間で呼び出しの負荷分散を行う方法を指定します。

この付録では、**workers.properties** ファイルのレイアウトと内容について説明します。

使用している Apache HTTP Server ディストリビューションにより、**workers.properties** ファイルの場所は異なります。

- JBCS Apache HTTP Server を使用している場合、**workers.properties** ファイルは **JBCS_HOME/httpd/conf.d** ディレクトリーにあります。
- RHEL Apache HTTP Server を使用している場合、**workers.properties** ファイルは **/etc/httpd/conf.d** ディレクトリーにあります。



注記

RHEL Apache HTTP サーバーは、RHEL 9 上でのみ **mod_jk** の使用をサポートします。

D.1. WORKERS.PROPERTIES の概要

workers.properties ファイルには、グローバルプロパティーセクションとワーカープロパティーセクションが含まれています。

グローバルプロパティー
　このセクションには、すべてのワーカーに適用されるディレクティブが含まれます。

ワーカープロパティー
　このセクションには、各ワーカーに適用されるディレクティブが含まれます。

各ノードはワーカープロパティーの命名規則を使用して定義されます。ワーカー名には、小文字、大文字、数字、および特定の特殊文字 (_、/) のみを含めることができます。

worker プロパティーの構造は **worker.WORKER_NAME.DIRECTIVE** です。

worker
　すべてのワーカープロパティーの定数接頭辞。

WORKER_NAME
　ワーカーに指定された任意の名前。たとえば、**node1**、**node_01**、**Node_1** など。

DIRECTIVE
　必要な特定のディレクティブ。

D.2. WORKERS.PROPERTIES ディレクティブ

workers.properties ファイルのディレクティブは、グローバル、必須、接続、および負荷分散の分類に分けられます。

worker.properties のグローバルディレクティブ

worker.list

mod_jk が使用するワーカー名のリストを指定します。このリストのワーカーは、要求をマップするために利用できます。

注記

ロードバランサーによって管理されない単一ノード設定は **worker.list=WORKER_NAME** に設定する必要があります。

workers.properties の必須ディレクティブ

type

ワーカーに適用されるディレクティブを決定するワーカーのタイプを指定します。デフォルト値は **ajp13** です。これは、Web サーバーと Apache HTTP Server 間の通信に選択するワーカータイプです。
その他の値には、**lb** および **status** が含まれます。

AJPv13 の詳細は、Apache Tomcat Connector - AJP Protocol Reference を参照してください。

workers.properties の接続ディレクティブ

host

ワーカーのホスト名または IP アドレス。ワーカーノードは ajp13 プロトコルスタックをサポートする必要があります。デフォルト値は **localhost** です。
ホスト名または IP アドレスの後にポート番号を追加すると、**port** ディレクティブを host ディレクティブの一部として指定できます。たとえば、**worker.node1.host=192.168.2.1:8009** または **worker.node1.host=node1.example.com:8009** のようになります。

port

定義されたプロトコルリクエストをリッスンしているバックエンドサーバーインスタンスのポート番号。デフォルト値は **8009** で、AJPv13 ワーカーのデフォルトのリッスンポートです。

ping_mode

現在のネットワークの正常性に対して接続がプローブされる条件を指定します。
プローブは、**CPing** に空の AJPv13 パケットを使用し、指定のタイムアウト内で **CPong** を返すことを想定します。

ディレクティブフラグの組み合わせを使用して条件を指定します。フラグはコンマ区切りではありません。たとえば、正しいディレクティブフラグセットは **worker.node1.ping_mode=CI** です。

C - Connect (接続)。

サーバーへの接続後に接続がプローブされるよう指定します。**connect_timeout** ディレクティブを使用してタイムアウトを指定します。指定しない場合は、**ping_timeout** の値が使用されます。

P (プレポスト)

各リクエストをサーバーに送信する前に接続がプローブされることを指定します。**prepost_timeout** ディレクティブを使用してタイムアウトを指定します。指定しない場合は、**ping_timeout** の値が使用されます。

I - Interval (間隔)。

定期的な内部メンテナンスサイクル中に接続がプローブされることを指定します。**connection_ping_interval** ディレクティブを使用してインターバルでアイドルのタイムアウトを指定します。指定しない場合は、**ping_timeout** の値が使用されます。

A - All (すべて)。

すべてのディレクティブフラグが適用されることを指定する最も一般的な設定。高度なディレクティブ *_timeout の詳細は、Apache Tomcat Connector - Reference Guide を参照してください。

ping_timeout

CPing 接続プローブへの **CPong** 応答を待つ時間を指定します (**ping_mode** を参照)。デフォルトの値は **10000** ミリ秒です。

worker.properties の負荷分散ディレクティブ

lbfactor

個別のワーカーの負荷分散係数を指定します。ロードバランサーのメンバーワーカーにのみ指定されます。
このディレクティブは、クラスター内の他のワーカーと比較して、ワーカーに分散される HTTP 要求負荷の相対量を定義します。

このディレクティブが適用される一般的な例は、クラスターで処理能力の高いサーバーを他のサーバーと区別する場合です。たとえば、あるワーカーに他のワーカーの 3 倍の負荷を割り当てる場合は **worker.WORKER_NAME.lbfactor=3** を指定します。

balance_workers

ロードバランサーが管理する必要があるワーカーノードを指定します。ディレクティブは同じロードバランサーに複数回使用でき、**workers.properties** ファイルに指定されるワーカー名のコンマ区切りリストで設定されます。

sticky_session

SESSION ID を持つワーカーのリクエストが同じワーカーにルーティングされるかどうかを指定します。デフォルトは **0** (false) です。**1** (true) に設定すると、ロードバランサーの永続性が有効になります。
たとえば、**worker.loadbalancer.sticky_session=0** を指定する場合、各要求はクラスター内の各ノード間で負荷分散されます。つまり、同じセッションの異なるリクエストは、サーバーの負荷に基づいて異なるサーバーに送信できます。

worker.loadbalancer.sticky_session=1 を指定した場合は、セッションが終了するまで各セッションが 1 台のサーバーに永続化されます (そのサーバーが使用可能な場合)。

関連情報

- Apache Tomcat コネクター - リファレンスガイド

付録E MOD_PROXY_CLUSTER のワーカーノード設定リファレンス

E.1. ワーカーノード設定

設定値は、以下の条件下でプロキシーに送信されます。

- サーバーの起動時。
- プロキシーがアドバタイズメカニズムで検出される場合。
- エラーが発生すると、プロキシーの設定がリセットされます。

表E.1 Tomcat のプロキシー設定値

値	デフォルト	説明
stickySession	true	あるセッションの後続リクエストを可能な限り同じノードへルーティングするべきかどうかを指定します。
stickySessionRemove	false	バランサーがリクエストをスタックしたノードへルーティングできない場合、Apache HTTP Server プロキシーがセッションのスティッキネスを削除するかどうかを指定します。**stickySession** が **false** の場合は、このプロパティーは無視されます。
stickySessionForce	true	バランサーがリクエストをスタックしたノードへルーティングできない場合、Apache HTTP Server プロキシーがエラーを返すかどうかを指定します。**stickySession** が **false** の場合は、このプロパティーは無視されます。
workerTimeout	-1	リクエストを処理するためにワーカーが利用可能になるまで待機する秒数を指定します。バランサーのすべてのワーカーが使用できなくなると、しばらく (**workerTimeout**/**100**) した後に **mod_proxy_cluster** を再試行して使用可能なワーカーを見つけます。**-1** の値は、Apache HTTP Server がワーカーが利用可能になるまで待機せず、利用可能なワーカーがない場合にエラーを返すことを示します。
maxAttempts	1	Apache HTTP Server プロキシーがワーカーに指定のリクエストの送信を試みる回数を指定します。この回数試行した後に送信を断念します。最小値は **1** で、1回送信を試みた後、断念します。
flushPackets	false	パケットのフラッシュが有効または無効化されるかどうかを指定します。

値	デフォルト	説明
flushWait	-1	パケットをフラッシュするまでの待機時間を指定します。**-1** を値として指定すると、永久に待機します。
ping	10	ping に対する pong 応答を待つ時間 (秒単位)。
smax		soft maximum アイドル接続数 を指定します。最大値は Apache HTTP Server スレッド設定 (**ThreadsPerChild** または **1**) によって決まります。
ttl	60	**smax** しきい値を超えるアイドル接続の期間 (秒単位) を指定します。
nodeTimeout	-1	**mod_proxy_cluster** がエラーを返す前にバックエンドサーバーの応答を待機する時間 (秒単位) を指定します。**mod_proxy_cluster** は、リクエストを転送する前に常に **cping**/**cpong** を使用します。**mod_proxy_cluster** によって使用される **connectiontimeout** 値は、ping 値です。
balancer	mycluster	ロードバランサーの名前を指定します。
loadBalancingGroup		同じ負荷分散グループ内の **jvmRoutes** 間の負荷分散を指定します。**loadBalancingGroup** はドメインディレクティブと論理的に同等です。

E.2. MOD_PROXY_CLUSTER のプロキシーおよびプロキシー検出設定属性

次の表には、**mod_proxy_cluster** のプロキシーおよびプロキシー検出設定属性に関する属性と情報が含まれています。

表E.2 mod_proxy_cluster のプロキシー検出設定属性

属性	プロパティー	デフォルト値
proxy-list	**proxyList**	
proxy-url	**proxyURL**	
advertise	**advertise**	true
advertise-security-key	**advertiseSecurityKey**	
excluded-contexts	**excludedContexts**	
auto-enable-contexts	**autoEnableContexts**	true

属性	プロパティー	デフォルト値
stop-context-timeout	**stopContextTimeout**	10 秒 (秒単位)
socket-timeout	**nodeTimeout**	20 秒 (ミリ秒単位)

注記

nodeTimeout が定義されていない場合は、**ProxyTimeout** ディレクティブ **Proxy** が使用されます。**ProxyTimeout** が定義されていない場合は、サーバーのタイムアウト (**Timeout**) が使用されます (JBCS httpd.conf ではデフォルトで 120 秒)。**nodeTimeout**、**ProxyTimeout**、および **Timeout** はソケットレベルで設定されます。

表E.3 mod_proxy_cluster のプロキシー設定属性

属性	プロパティー	デフォルト値
sticky-session	**stickySession**	true
sticky-session-remove	**stickySessionRemove**	false
sticky-session-force	**stickySessionForce**	true
node-timeout	**workerTimeout**	-1
max-attempts	**maxAttempts**	1
flush-packets	**flushPackets**	false
flush-wait	**flushWait**	-1
ping	**ping**	10 (秒)
smax	**smax**	-1 (デフォルト値を使用)
ttl	**ttl**	-1 (デフォルト値を使用)
domain	**loadBalancingGroup**	
load-balancing-group	**loadBalancingGroup**	

E.3. TOMCAT の負荷設定

Apache Tomcat で **mod_proxy_cluster** を使用する場合は、負荷メトリックの次の追加プロパティーを設定できます。

表E.4 Tomcat の負荷設定

属性	デフォルト値	説明
loadMetricClass	**org.jboss.modcluster.load.metric.impl.BusyConnectorsLoadMetric**	**org.jboss.load.metric.LoadMetric** を実装するオブジェクトのクラス名
loadMetricCapacity	1	**loadMetricClass** プロパティーにより定義された負荷メトリックの容量
loadHistory	9	負荷分散係数の計算で考慮する必要のある、過去の負荷値の数
loadDecayFactor	2	これまでの負荷値を若干減少させる係数

付録F マルチプロセッシングモジュール (MPM)

Red Hat JBoss Core Services には、さまざまなマルチプロセッシングモジュール (MPM) が含まれています。これらの MPM を使用して、Apache HTTP Server が着信要求に応答する方法をカスタマイズできます。

注記

MPM は相互に排他的です。特定の時点で 1 つの MPM のみを有効にして使用すできます。

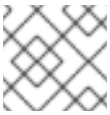
F.1. MPM の概要

Multi-Processing Modules (MPM) は、Red Hat Enterprise Linux (RHEL) と Windows Server の両方で使用できます。RHEL では、デフォルトの MPM はオペレーティングシステムのバージョンにより異なります。

RHEL 用の MPM

prefork

prefork MPM は、スレッド化前の Web サーバーを実装します。 **prefork** MPM は単一の制御プロセスを使用し、着信接続をリッスンしてサービスを提供する子プロセスを起動します。単一のプロセスが特定のリクエストを処理するため、各リクエストが分離され、他のリクエストに影響を与えません。



注記

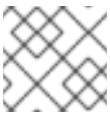
prefork MPM は、RHEL 7 のデフォルト MPM です。

worker

worker MPM はハイブリッドマルチプロセス、マルチスレッドサーバーを実装します。各子プロセスにより、決まった数のサーバースレッドが作成され、サーバーは少ないシステムリソースで多数のリクエストを処理できます。

event

event MPM は **worker** MPM に基づいています。 **event** MPM を使用すると、一部の処理作業をリスナースレッドに委任することで、追加のリクエストを同時に処理できます。これにより、ワーカースレッドが解放され、新しいリクエストを処理できるようになります。



注記

event MPM は、RHEL バージョン 8 および 9 のデフォルト MPM です。

MPM for Microsoft Windows

winnt

winnt MPM は、Windows システムで使用できる唯一の MPM です。 **winnt** MPM は単一の制御プロセスを使用します。この制御プロセスは、着信リクエストのスレッドを作成する別のプロセスを起動します。

F.2. MPM の切り替え

サーバーは、Apache HTTP Server ホスト上の **00-mpm.conf** ファイル内の **LoadModule** ディレクティブに基づき MPM を選択します。**00-mpm.conf** ファイルにある対象の MPM の **LoadModule** ディレクティブからコメント文字 (**#**) を削除することにより、特定の MPM を選択できます。

使用している Apache HTTP Server ディストリビューションにより、**00-mpm.conf** ファイルの場所は異なります。

- JBCS Apache HTTP Server を使用している場合、**00-mpm.conf** ファイルは **JBCS_HOME/httpd/conf.modules.d** ディレクトリーにあります。
- RHEL Apache HTTP Server を使用している場合、**00-mpm.conf** ファイルは **/etc/httpd/conf.modules.d** ディレクトリーにあります。

使用しているオペレーティングシステムのバージョンに応じて、次のガイドラインに留意してください。

- RHEL バージョン 8 および 9 では、**event** MPM がデフォルトで選択されています。以下に例を示します。

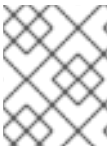
```
# event MPM: A variant of the worker MPM with the goal of consuming
# threads only for connections with active processing
# See: http://httpd.apache.org/docs/2.4/mod/event.html
#
LoadModule mpm_event_module modules/mod_mpm_event.so
```

  event MPM はマルチスレッドであり、最適化されたパフォーマンスを提供するように設計されています。RHEL バージョン 8 または 9 を使用している場合、**prefork** などの別の MPM に切り替えると、パフォーマンスの問題が発生する可能性があります。

- RHEL 7 では、**prefork** MPM がデフォルトで選択されています。以下に例を示します。

```
# prefork MPM: Implements a non-threaded, pre-forking web server
# See: http://httpd.apache.org/docs/2.4/mod/prefork.html
LoadModule mpm_prefork_module modules/mod_mpm_prefork.so
```

  RHEL 7 を使用している場合は、パフォーマンスの問題の可能性を回避するために、**worker** や **event** などの別の MPM に切り替えることを検討してください。



注記

分かりやすくするため、次の手順では **prefork** MPM から **worker** MPM に切り替える方法について説明します。

手順

1. **00-mpm.conf** ファイルが格納されているディレクトリーに移動します。
   - JBCS Apache HTTP Server を使用している場合は、**JBCS_HOME/httpd/conf.modules.d** ディレクトリーに移動します。
   - RHEL Apache HTTP Server を使用している場合は、**/etc/httpd/conf.modules.d** ディレクトリーに移動します。
2. **00-mpm.conf** を編集して、**prefork** MPM の **LoadModule** ディレクティブにコメント (**#**) 文字を追加します。
   以下に例を示します。

```
# prefork MPM: Implements a non-threaded, pre-forking web server
# See: http://httpd.apache.org/docs/2.4/mod/prefork.html
#LoadModule mpm_prefork_module modules/mod_mpm_prefork.so
```

3. 同じ **00-mpm.conf** ファイルで、切り替え先の MPM の **LoadModule** ディレクティブからコメント (**#**) 文字を削除します。これらの行は、**prefork** MPM のすぐ下にあります。
たとえば、**worker** MPM をロードするには、**worker** MPM の **LoadModule** ディレクティブからコメント (**#**) 文字を削除します。

```
# worker MPM: Multi-Processing Module implementing a hybrid
# multi-threaded multi-process web server
# See: http://httpd.apache.org/docs/2.4/mod/worker.html
LoadModule mpm_worker_module modules/mod_mpm_worker.so
```

検証

- MPM が正しく設定されていることを確認するには、次のコマンドを入力します。

```
$ sbin/apachectl -V
```

上記のコマンドは、現在の MPM を表示します。

以下に例を示します。

```
Server MPM:     worker
```

F.3. MPM パフォーマンス設定

MPM の種類ごとに、MPM のパフォーマンスを最適化するためにさまざまな設定を設定できます。

MPM パフォーマンス設定の種類
MPM パフォーマンス設定では、次のタイプの基準を指定します。

- 起動時に作成するサーバープロセスの初期数
- アイドル状態のスレッドまたはサーバープロセスの最小数と最大数
- リクエストの処理に使用できるスレッドまたはサーバープロセスの最大数
- 個々のサーバープロセスが処理できるリクエストの最大数
- 各サーバープロセスが作成するスレッドの数(**worker** および **event** MPM のみ)
- サーバーの存続期間中に開始できるサーバープロセスの最大数の上限 (**prefork** MPM のみ)

MPM パフォーマンス設定の設定ファイル
JBCS 2.4.51 以降では、**mpm.conf** ファイルで MPM パフォーマンス設定を設定できます。使用している Apache HTTP Server ディストリビューションにより、**mpm.conf** ファイルの場所は異なります。

- JBCS Apache HTTP Server を使用している場合、**mpm.conf** ファイルは **JBCS_HOME/httpd/conf.d** ディレクトリーにあります。
- RHEL Apache HTTP Server を使用している場合、**mpm.conf** ファイルは **/etc/httpd/conf.d** ディレクトリーにあります。

重要

JBCS 2.4.37 以前のリリースでは、**conf.modules.d/00-mpm.conf** ファイルに MPM パフォーマンス設定が含まれていました。JBCS 2.4.51 以降、**conf.d/mpm.conf** ファイルにはこれらの設定が含まれています。

JBCS 2.4.37 以前からアップグレードする場合は、**conf.modules.d/00-mpm.conf** で以前に設定したカスタマイズ設定と一致するように、アップグレードした 2.4.51 インストールの **conf.d/mpm.conf** ファイルを設定してください。そうしないと、アップグレードされた JBCS 2.4.51 インストールで **conf.d/mpm.conf** ファイルのデフォルト設定が自動的に使用され、予期しないパフォーマンスの問題が発生する可能性があります。

使用可能なパフォーマンス設定および関連するデフォルト値の詳細については、Apache HTTP Server インストール内の **conf.d/mpm.conf** ファイルを参照してください。

関連情報

- Apache MPM 共通ディレクティブ