Red Hat Data Grid 8.1

Data Grid セキュリティーガイド

Data Grid セキュリティーの有効化および設定

Last Updated: 2023-12-15

Red Hat Data Grid 8.1 Data Grid セキュリティーガイド

Data Grid セキュリティーの有効化および設定

法律上の通知

概要

ネットワークから Data Grid デプロイメントを保護します。許可されたユーザーのデータアクセスを制限します。

目次

RED HAT DATA GRID

Data Grid は、高性能の分散型インメモリーデータストアです。

スキーマレスデータ構造

さまざまなオブジェクトをキーと値のペアとして格納する柔軟性があります。

グリッドベースのデータストレージ

クラスター間でデータを分散および複製するように設計されています。

エラスティックスケーリング

サービスを中断することなく、ノードの数を動的に調整して要件を満たします。

データの相互運用性

さまざまなエンドポイントからグリッド内のデータを保存、取得、およびクエリーします。

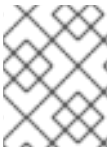
DATA GRID のドキュメント

Data Grid のドキュメントは、Red Hat カスタマーポータルで入手できます。

- Data Grid 8.1 ドキュメント
- Data Grid 8.1 コンポーネントの詳細
- Data Grid 8.1 でサポートされる設定
- Data Grid 8 機能のサポート
- Data Grid で非推奨の機能

DATA GRID のダウンロード

Red Hat カスタマーポータルで Data Grid Software Downloads にアクセスします。



注記

Data Grid ソフトウェアにアクセスしてダウンロードするには、Red Hat アカウントが必要です。

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 DATA GRID 承認の設定

承認によって、Data Grid で操作を実行してデータにアクセスする機能を制限します。ユーザーに、さまざまなパーミッションレベルを持つロールを割り当てます。

1.1. DATA GRID 承認

Data Grid を使用すると、キャッシュマネージャーとキャッシュインスタンスをセキュリティー保護するための承認を設定できます。ユーザーアプリケーションまたはクライアントが、セキュリティー保護されたキャッシュマネージャーおよびキャッシュに対して操作を実行しようとする場合、その操作を実行するための十分なパーミッションを持つロールをアイデンティティーに提供する必要があります。

たとえば、特定のキャッシュインスタンスで承認を設定して、**Cache.get()** を呼び出すには、読み取り権限を持つロールをアイデンティティーに割り当てる必要があり、**Cache.put()** を呼び出すには書き込み権限を持つロールが必要になるようにします。

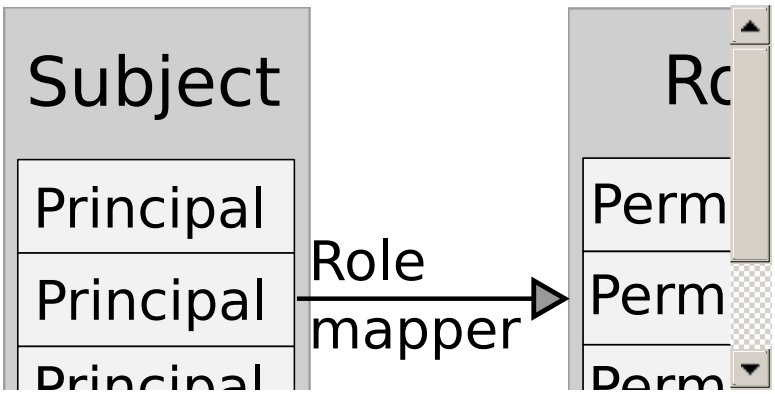
このシナリオでは、**reader** ロールが割り当てられたユーザーアプリケーションまたはクライアントがエントリーの書き込みを試みると、Data Grid はリクエストを拒否し、セキュリティー例外を出力します。**writer** ロールのあるユーザーアプリケーションまたはクライアントが書き込みリクエストを送信する場合、Data Grid は承認を検証し、後続の操作のためにトークンを発行します。

アイデンティティーからロールへのマッピング

アイデンティティーは **java.security.Principal** タイプのセキュリティープリンシパルです。**javax.security.auth.Subject** クラスで実装されたサブジェクトは、セキュリティープリンシパルのグループを表します。つまり、サブジェクトはユーザーとそれが属するすべてのグループを表します。

Data Grid はロールマッパーを使用して、セキュリティープリンシパルが 1 つ以上のパーミッションを表すロールに対応するようにします。

次の図は、セキュリティープリンシパルがどのようにロールにマッピングされるかを示しています。



1.1.1. パーミッション

パーミッションは、実行できるアクションを制限することによって、キャッシュマネージャーとキャッシュへのアクセスを制御します。パーミッションは、名前付きキャッシュなどの特定のエンティティーに適用することもできます。

表1.1 キャッシュマネージャーのアクセス許可

パーミッション	機能	説明

パーミッション	機能	説明
設定	**defineConfiguration**	新しいキャッシュ設定を定義します。
LISTEN	**addListener**	キャッシュマネージャーに対してリスナーを登録します。
ライフサイクル	**stop**	キャッシュマネージャーを停止します。
ALL	-	すべてのキャッシュマネージャーのアクセス許可が含まれます。

表1.2 キャッシュ権限

パーミッション	機能	説明
READ	**get**, **contains**	キャッシュからエントリーを取得します。
WRITE	**put**, **putIfAbsent**, **replace**, **remove**, **evict**	キャッシュ内のデータの書き込み、置換、削除、エビクト。
EXEC	**distexec**, **streams**	キャッシュに対するコードの実行を許可します。
LISTEN	**addListener**	キャッシュに対してリスナーを登録します。
BULK_READ	**keySet**, **values**, **entrySet**, **query**	一括取得操作を実行します。
BULK_WRITE	**clear**, **putAll**	一括書き込み操作を実行します。
ライフサイクル	**start**, **stop**	キャッシュを開始および停止します。

パーミッション	機能	説明
ADMIN	**getVersion**, **addInterceptor***, **removeInterceptor**, **getInterceptorChain**, **getEvictionManager**, **getComponentRegistry**, **getDistributionManager**, **getAuthorizationManager**, **evict**, **getRpcManager**, **getCacheConfiguration**, **getCacheManager**, **getInvocationContextContainer**, **setAvailability**, **getDataContainer**, **getStats**, **getXAResource**	基盤となるコンポーネントと内部構造へのアクセスを許可します。
ALL	-	すべてのキャッシュパーミッションが含まれます。
ALL_READ	-	READ パーミッションと BULK_READ パーミッションを組み合わせます。
ALL_WRITE	-	WRITE パーミッションと BULK_WRITE パーミッションを組み合わせます。

パーミッションの組み合わせ

使いやすいようにパーミッションを組み合わせることが必要になる場合があります。たとえば、"supervisors" にはストリーム操作の実行を許可し、"standard" ユーザーについては puts と gets しか実行できないように制限するには、次のマッピングを定義できます。

```
<role name="standard" permission="READ WRITE" />
<role name="supervisors" permission="READ WRITE EXEC BULK"/>
```

参照

- Data Grid Security API

1.1.2. ロールマッパー

Data Grid には、サブジェクトのセキュリティープリンシパルを承認ロールにマッピングする **PrincipalRoleMapper** API が含まれています。デフォルトで使用できるロールマッパーが 2 つあります。

IdentityRoleMapper

ロール名としてプリンシパル名を使用します。

- Java クラス: **org.infinispan.security.mappers.IdentityRoleMapper**

- 宣言型設定: **<identity-role-mapper />**

CommonNameRoleMapper

プリンシパル名が識別名 (DN) の場合は、共通名 (CN) をロール名として使用します。たとえば、**cn=managers,ou=people,dc=example,dc=com** DN は **managers** ロールにマッピングされます。

- Java クラス: **org.infinispan.security.mappers.CommonRoleMapper**
- 宣言型設定: **<common-name-role-mapper />**

org.infinispan.security.PrincipalRoleMapper インターフェイスを実装するカスタムロールマッパーを使用することもできます。カスタムロールマッパーを宣言的に設定するには、**<custom-role-mapper class="my.custom.RoleMapper"/>** を使用します。

参照

- Data Grid Security API
- org.infinispan.security.PrincipalRoleMapper

1.2. プログラムでの承認の設定

Data Grid を組み込みライブラリーとして使用する場合は、**GlobalSecurityConfigurationBuilder** クラスおよび **ConfigurationBuilder** クラスで承認を設定できます。

手順

1. 承認を有効にし、ロールマッパーを指定し、ロールおよびパーミッションのセットを定義する **GlobalConfigurationBuilder** を作成します。

```
GlobalConfigurationBuilder global = new GlobalConfigurationBuilder();
global
  .security()
    .authorization().enable() 1
      .principalRoleMapper(new IdentityRoleMapper()) 2
      .role("admin") 3
        .permission(AuthorizationPermission.ALL)
      .role("reader")
        .permission(AuthorizationPermission.READ)
      .role("writer")
        .permission(AuthorizationPermission.WRITE)
      .role("supervisor")
        .permission(AuthorizationPermission.READ)
        .permission(AuthorizationPermission.WRITE)
        .permission(AuthorizationPermission.EXEC);
```

1 Cache Manager の Data Grid 承認を有効にします。

2 ロールにプリンシパルをマップ **PrincipalRoleMapper** の実装を指定します。

3 ロールとその関連付けられたパーミッションを定義します。

2. **ConfigurationBuilder** で承認を有効にして、ユーザーロールに基づいてアクセスを制限します。

```
ConfigurationBuilder config = new ConfigurationBuilder();
config
  .security()
    .authorization()
      .enable(); (1)
```

(1) 暗黙的に、グローバル設定からすべてのロールを追加します。

すべてのロールをキャッシュに適用しない場合は、以下のようにキャッシュに承認されたロールを明示的に定義します。

```
ConfigurationBuilder config = new ConfigurationBuilder();
config
  .security()
    .authorization()
      .enable()
      .role("admin") (1)
      .role("supervisor")
      .role("reader");
```

(1) キャッシュに承認されたロールを定義します。この例では、**writer** ロールのみを持つユーザーは "secured" キャッシュには許可されていません。Data Grid は、これらのユーザーからのアクセス要求を拒否します。

参照

- org.infinispan.configuration.global.GlobalSecurityConfigurationBuilder
- org.infinispan.configuration.cache.ConfigurationBuilder

1.3. 宣言的な承認の設定

infinispan.xml ファイルで承認を設定します。

手順

1. **cache-container** でロールマッパーを指定するグローバル承認を設定し、ロールとパーミッションのセットを定義します。
2. ユーザーロールに基づいてアクセスを制限するようにキャッシュの承認を設定します。

```
<infinispan>
  <cache-container default-cache="secured" name="secured">
    <security>
      <authorization> (1)
        <identity-role-mapper /> (2)
        <role name="admin" permissions="ALL" /> (3)
        <role name="reader" permissions="READ" />
        <role name="writer" permissions="WRITE" />
```

```
        <role name="supervisor" permissions="READ WRITE EXEC"/>
      </authorization>
    </security>
    <local-cache name="secured">
      <security>
        <authorization/> 4
      </security>
    </local-cache>
  </cache-container>
</infinispan>
```

1. Cache Manager の Data Grid 承認を有効にします。
2. ロールにプリンシパルをマップ **PrincipalRoleMapper** の実装を指定します。
3. ロールとその関連付けられたパーミッションを定義します。
4. 暗黙的に、グローバル設定からすべてのロールを追加します。

すべてのロールをキャッシュに適用しない場合は、以下のようにキャッシュに承認されたロールを明示的に定義します。

```
<infinispan>
  <cache-container default-cache="secured" name="secured">
    <security>
      <authorization>
        <identity-role-mapper />
        <role name="admin" permissions="ALL" />
        <role name="reader" permissions="READ" />
        <role name="writer" permissions="WRITE" />
        <role name="supervisor" permissions="READ WRITE EXEC"/>
      </authorization>
    </security>
    <local-cache name="secured">
      <security>
        <authorization roles="admin supervisor reader"/> 1
      </security>
    </local-cache>
  </cache-container>

</infinispan>
```

1. キャッシュに承認されたロールを定義します。この例では、**writer** ロールのみを持つユーザーは "secured" キャッシュには許可されていません。Data Grid は、これらのユーザーからのアクセス要求を拒否します。

参照

- Data Grid Configuration Schema Reference

1.4. セキュアなキャッシュを使用したコード実行

Data Grid 承認を設定して **DefaultCacheManager** を構築すると、基礎となるキャッシュで操作を呼び

出す前にセキュリティーコンテキストを確認する **SecureCache** を返します。また、**SecureCache** は、アプリケーションが **DataContainer** などの低レベルの非セキュアなオブジェクトを取得できないようにします。このため、必要な承認を持つアイデンティティーでコードを実行する必要があります。

Java で特定のアイデンティティーでコードを実行すると、通常、以下のように **PrivilegedAction** 内で実行されるコードをラップします。

```
import org.infinispan.security.Security;

Security.doAs(subject, new PrivilegedExceptionAction<Void>() {
public Void run() throws Exception {
   cache.put("key", "value");
}
});
```

Java 8 を使用すると、以下のように前述の呼び出しを簡素化できます。

```
Security.doAs(mySubject, PrivilegedAction<String>() -> cache.put("key", "value"));
```

上記の呼び出しは、**Subject.doAs()** の代わりに **Security.doAs()** メソッドを使用します。Data Grid でどちらのメソッドも使用できますが、**Security.doAs()** によりパフォーマンスが向上します。

現在の Subject が必要な場合は、以下の呼び出しを使用して Data Grid コンテキストまたは AccessControlContext から取得します。

```
Security.getSubject();
```

第2章 クラスタートランスポートの暗号化

ノードが暗号化されたメッセージと通信できるように、クラスタートランスポートを保護します。また、有効なアイデンティティーを持つノードのみが参加できるように、証明書認証を実行するように Data Grid クラスターを設定することもできます。

2.1. DATA GRID クラスターのセキュリティー

クラスタートラフィックのセキュリティーを保護するには、Data Grid ノードを設定し、シークレットキーで JGroups メッセージペイロードを暗号化します。

Data Grid ノードは、以下のいずれかから秘密鍵を取得できます。

- コーディネーターノード (非対称暗号化)
- 共有キーストア (対称暗号化)

コーディネーターノードからの秘密鍵の取得

非対称暗号化は、Data Grid 設定の JGroups スタックに **ASYM_ENCRYPT** プロトコルを追加して対称暗号化を設定します。これにより、Data Grid クラスターはシークレットキーを生成して配布できます。

重要

非対称暗号化を使用する場合は、ノードが証明書認証を実行し、シークレットキーを安全に交換できるようにキーストアを提供する必要もあります。これにより、中間者 (MitM) 攻撃からクラスターが保護されます。

非対称暗号化は、以下のようにクラスタートラフィックのセキュリティーを保護します。

1. Data Grid クラスターの最初のノードであるコーディネーターノードは、秘密鍵を生成します。
2. 参加ノードは、コーディネーターとの証明書認証を実行して、相互に ID を検証します。
3. 参加ノードは、コーディネーターノードに秘密鍵を要求します。その要求には、参加ノードの公開鍵が含まれています。
4. コーディネーターノードは、秘密鍵を公開鍵で暗号化し、参加ノードに返します。
5. 参加ノードは秘密鍵を復号してインストールします。
6. ノードはクラスターに参加し、秘密鍵でメッセージを暗号化および復号化します。

共有キーストアからの秘密鍵の取得

対称暗号化は、Data Grid 設定の JGroups スタックに **SYM_ENCRYPT** プロトコルを追加して対称暗号化を設定します。これにより、Data Grid クラスターは、指定したキーストアから秘密鍵を取得できます。

1. ノードは、起動時に Data Grid クラスパスのキーストアから秘密鍵をインストールします。
2. ノードはクラスターに参加し、秘密鍵でメッセージを暗号化および復号化します。

非対称暗号化と対称暗号化の比較

証明書認証を持つ **ASYM_ENCRYPT** は、**SYM_ENCRYPT** と比較して、暗号化の追加の層を提供します。秘密鍵のコーディネーターノードへのリクエストを暗号化するキーストアを提供します。Data Grid は、そのシークレットキーを自動的に生成し、クラスタートラフィックを処理し、秘密鍵の生成時に指定します。たとえば、ノードが離れる場合に新規のシークレットキーを生成するようにクラスターを設定できます。これにより、ノードが証明書認証を回避して古いキーで参加できなくなります。

一方、**SYM_ENCRYPT** は **ASYM_ENCRYPT** よりも高速です。ノードがクラスターコーディネーターとキーを交換する必要がないためです。**SYM_ENCRYPT** への潜在的な欠点は、クラスターのメンバーシップの変更時に新規シークレットキーを自動的に生成するための設定がないことです。ユーザーは、ノードがクラスタートラフィックを暗号化するのに使用するシークレットキーを生成して配布する必要があります。

2.2. 非対称暗号化を使用したクラスタートランスポートの設定

Data Grid クラスターを設定し、JGroups メッセージを暗号化するシークレットキーを生成して配布します。

手順

1. Data Grid がノードの ID を検証できるようにする証明書チェーンでキーストアを作成します。
2. クラスター内の各ノードのクラスパスにキーストアを配置します。
   Data Grid Server の場合は、$RHDG_HOME ディレクトリーにキーストアを配置します。
3. 以下の例のように、**SSL_KEY_EXCHANGE** プロトコルおよび **ASYM_ENCRYPT** プロトコルを Data Grid 設定の JGroups スタックに追加します。

```
<infinispan>
  <jgroups>
    <stack name="encrypt-tcp" extends="tcp"> 1
      <SSL_KEY_EXCHANGE keystore_name="mykeystore.jks" 2
                        keystore_password="changeit" 3
                        stack.combine="INSERT_AFTER"
                        stack.position="VERIFY_SUSPECT"/> 4
      <ASYM_ENCRYPT asym_keylength="2048" 5
            asym_algorithm="RSA" 6
            change_key_on_coord_leave = "false" 7
            change_key_on_leave = "false" 8
            use_external_key_exchange = "true" 9
            stack.combine="INSERT_AFTER"
            stack.position="SSL_KEY_EXCHANGE"/> 10
    </stack>
  </jgroups>
  <cache-container name="default" statistics="true">
    <transport cluster="${infinispan.cluster.name}"
          stack="encrypt-tcp" 11
          node-name="${infinispan.node.name:}"/>
  </cache-container>
</infinispan>
```

1. Data Grid のデフォルト TCP スタックを拡張する "encrypt-tcp" という名前のセキュアな JGroups スタックを作成します。
2. ノードが証明書認証を実行するために使用するキーストアに名前を付けます。

3. キーストアのパスワードを指定します。
4. **stack.combine** 属性と **stack.position** 属性を使用して、デフォルトの TCP スタックの **VERIFY_SUSPECT** プロトコルの後に **SSL_KEY_EXCHANGE** を挿入します。
5. コーディネーターノードが生成する秘密鍵の長さを指定します。デフォルト値は **2048** です。
6. コーディネーターノードが秘密鍵の生成に使用する暗号化エンジンを指定します。デフォルト値は **RSA** です。
7. コーディネーターノードが変更されたときに新しい秘密鍵を生成して配布するように Data Grid を設定します。
8. ノードが離脱するときに新しい秘密鍵を生成して配布するように Data Grid を設定します。
9. 証明書認証に **SSL_KEY_EXCHANGE** プロトコルを使用するように Data Grid ノードを設定します。
10. **stack.combine** 属性と **stack.position** 属性を使用して、デフォルトの TCP スタックの **SSL_KEY_EXCHANGE** プロトコルの後に **ASYM_ENCRYPT** を挿入します。
11. セキュアな JGroups スタックを使用するように Data Grid クラスターを設定します。

検証

Data Grid クラスターを起動した際、以下のログメッセージは、クラスターがセキュアな JGroups スタックを使用していることを示しています。

```
[org.infinispan.CLUSTER] ISPN000078: Starting JGroups channel cluster with stack
<encrypted_stack_name>
```

Data Grid ノードは **ASYM_ENCRYPT** を使用している場合のみクラスターに参加でき、コーディネーターノードからシークレットキーを取得できます。それ以外の場合は、次のメッセージが Data Grid ログに書き込まれます。

```
[org.jgroups.protocols.ASYM_ENCRYPT] <hostname>: received message without encrypt header
from <hostname>; dropping it
```

参照資料

この手順の **ASYM_ENCRYPT** の設定例は、一般的に使用されるパラメーターを示しています。利用可能なパラメーターの完全なセットについては、JGroups のドキュメントを参照してください。

- JGroups 4 Manual
- JGroups 4.2 Schema

2.3. 対称暗号化を使用したクラストートランスポートの設定

指定したキーストアからの秘密鍵を使用して JGroups メッセージを暗号化するように Data Grid クラスターを設定します。

手順

1. シークレットキーが含まれるキーストアを作成します。
2. クラスター内の各ノードのクラスパスにキーストアを配置します。
   Data Grid Server の場合は、$RHDG_HOME ディレクトリーにキーストアを配置します。
3. 次の例のように、Data Grid 設定の JGroups スタックに **SYM_ENCRYPT** プロトコルを追加します。

```
<infinispan>
  <jgroups>
    <stack name="encrypt-tcp" extends="tcp"> (1)
      <SYM_ENCRYPT keystore_name="myKeystore.p12" (2)
                   keystore_type="PKCS12" (3)
                   store_password="changeit" (4)
                   key_password="changeit" (5)
                   alias="myKey" (6)
                   stack.combine="INSERT_AFTER"
                   stack.position="VERIFY_SUSPECT"/> (7)
    </stack>
  </jgroups>
  <cache-container name="default" statistics="true">
    <transport cluster="${infinispan.cluster.name}"
               stack="encrypt-tcp" (8)
               node-name="${infinispan.node.name:}"/>
  </cache-container>
</infinispan>
```

(1) Data Grid のデフォルト TCP スタックを拡張する "encrypt-tcp" という名前のセキュアな JGroups スタックを作成します。

(2) ノードが秘密鍵を取得するキーストアに名前を付けます。

(3) キーストアのタイプを指定します。JGroups はデフォルトで JCEKS を使用します。

(4) キーストアのパスワードを指定します。

(5) 秘密鍵のパスワードを指定します。

(6) 秘密鍵のエイリアスを指定します。

(7) **stack.combine** 属性と **stack.position** 属性を使用して、デフォルトの TCP スタックの **VERIFY_SUSPECT** プロトコルの後に **SYM_ENCRYPT** を挿入します。

(8) セキュアな JGroups スタックを使用するように Data Grid クラスターを設定します。

検証

Data Grid クラスターを起動した際、以下のログメッセージは、クラスターがセキュアな JGroups スタックを使用していることを示しています。

```
[org.infinispan.CLUSTER] ISPN000078: Starting JGroups channel cluster with stack
<encrypted_stack_name>
```

Data Grid ノードは、**SYM_ENCRYPT** を使用し、共有キーストアからシークレットキーを取得できる場合に限りクラスターに参加できます。それ以外の場合は、次のメッセージが Data Grid ログに書き込まれます。

> [org.jgroups.protocols.SYM_ENCRYPT] <hostname>: received message without encrypt header from <hostname>; dropping it

参照資料

この手順の **SYM_ENCRYPT** の設定例は、一般的に使用されるパラメーターを示しています。利用可能なパラメーターの完全なセットについては、JGroups のドキュメントを参照してください。

- JGroups 4 Manual
- JGroups 4.2 Schema

第3章 DATA GRID ポートおよびプロトコル

Data Grid は、ネットワークにデータを分散し、外部クライアント要求の接続を確立できるため、Data Grid がネットワークトラフィックを処理するために使用するポートおよびプロトコルを認識する必要があります。

Data Grid をリモートサーバーとして実行する場合は、ファイアウォールを介してリモートクライアントを許可する必要がある場合があります。同様に、競合やネットワークの問題を防ぐために、Data Grid ノードがクラスター通信に使用するポートを調整する必要があります。

3.1. DATA GRID SERVER ポートおよびプロトコル

Data Grid Server は、リモートクライアントアクセス用にネットワーク上のエンドポイントを公開します。

ポート	プロトコル	説明
11222	TCP	Hot Rod および REST エンドポイント
11221	TCP	デフォルトで無効にされる Memcached エンドポイント。

3.1.1. リモート接続用のネットワークファイアウォールの設定

サーバーと外部クライアント間のトラフィックを許可するためにファイアウォールルールを調整します。

手順

Red Hat Enterprise Linux (RHEL) ワークステーションでは、たとえば、以下のように firewalld を使用してポート **11222** へのトラフィックを許可できます。

```
# firewall-cmd --add-port=11222/tcp --permanent
success
# firewall-cmd --list-ports | grep 11222
11222/tcp
```

ネットワーク全体に適用されるファイアウォールルールを設定するには、nftables ユーティリティーを使用できます。

参照資料

- firewalld の使用および設定
- nftables の使用

3.2. クラスタートラフィックの TCP および UDP ポート

Data Grid は、クラスタートランスポートメッセージに以下のポートを使用します。

デフォルトのポート	プロトコル	説明
7800	TCP/UDP	JGroups クラスターバインドポート
46655	UDP	JGroups マルチキャスト

クロスサイトレプリケーション
Data Grid は、JGroups RELAY2 プロトコルに以下のポートを使用します。

7900

OpenShift で実行している Data Grid クラスターの向け。

7800

ノード間のトラフィックに UDP を使用し、クラスター間のトラフィックに TCP を使用する場合。

7801

ノード間のトラフィックに TCP を使用し、クラスター間のトラフィックに TCP を使用する場合。