CLUSTERPRO による
Zabbix クラスタ化検証報告書

Do the Next, Open your Window
MIRACLE

目次

更新履歴

更新日付	バージョン	更新内容
2011/01/27	1.0	初版作成

1. はじめに

1.1 本検証報告書について

本書は、日本電気株式会社の協力の元、ミラクル・リナックス社で、CLUSTERPROを利用して、Zabbixサーバをクラスタ化した報告書です。

本ドキュメントは、検証作業や検証結果についてまとめられているものであり、本ドキュメントに関する内容について、ミラクル・リナックス株式会社、および日本電気株式会社が動作を保証するものではありません。 各ソフトウェアのバージョンおよび環境等の違いにより本書で解説される機能が正常に稼働しない場合があります。導入前の十分な検証を推奨いたします。
記載された会社名および製品名などは該当する各社の商標または登録商標です。

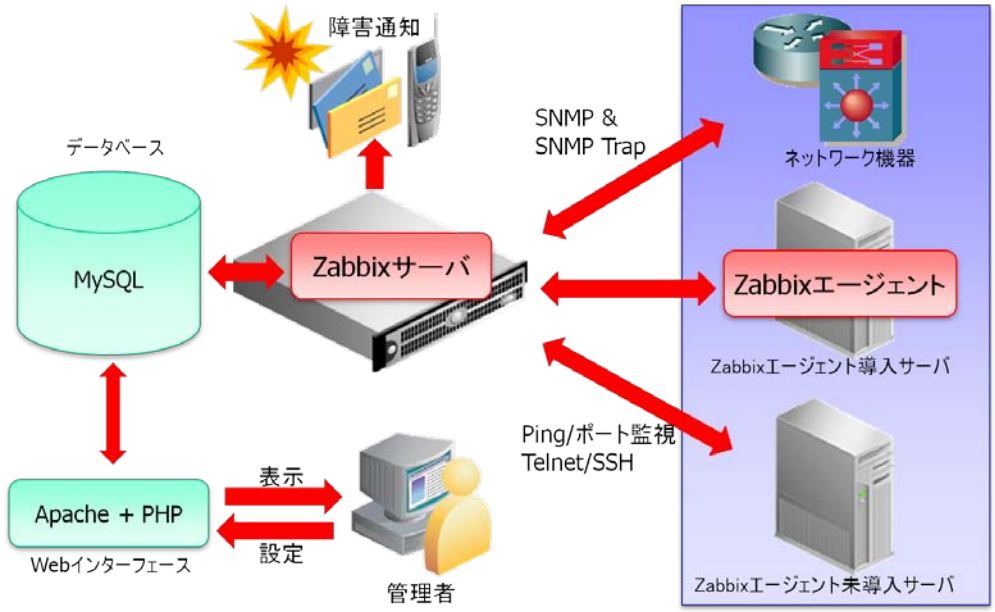
1.2 CLUSTERPRO 製品に関して

CLUSTERPROは日本電気株式会社の製品です。詳細は、以下のURLを参照してください。

http://www.nec.co.jp/clusterpro/index.html

2. Zabbix動作概要

ZabbixはZabbixサーバ、MySQL、Apacheなど複数のサービスを利用して動作します。



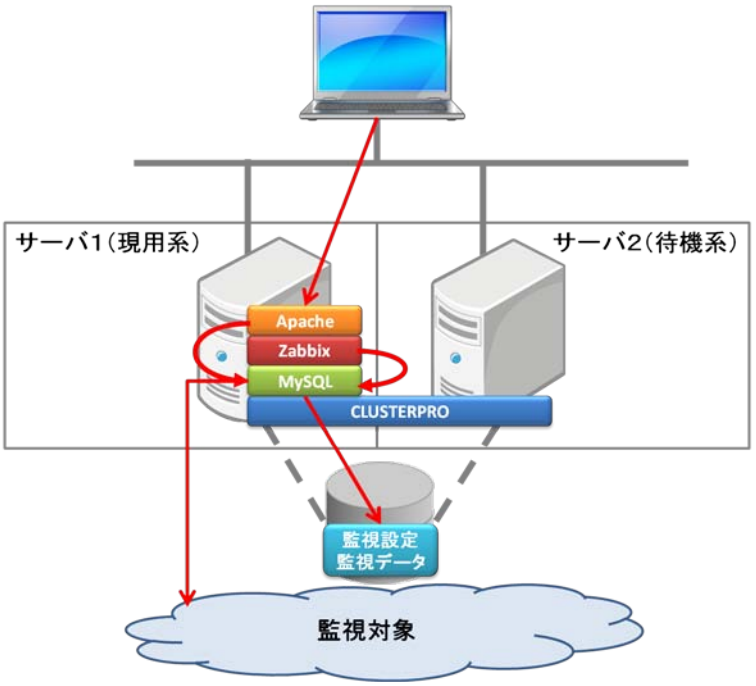
大まかな役割として、Zabbixサーバがサーバ、ネットワーク機器の監視を行い、MySQLに監視データを保存します。ユーザはWebブラウザを使ってMySQLに保存されたデータを表示させます。
なお、本書では単純に『監視』と記載する場合はZabbixによる監視、『監視リソース』と記載する場合は、CLUSTERPROによる監視とします。

2.1 機能概要

Zabbix 1.8を、CLUSTERPRO X 3.1以降の環境下で利用する場合、Zabbixサーバ、MySQL、Apacheを全て同一のサーバで動作させる片方向スタンバイ型と、それぞれ異なるホストで動作させる双方向スタンバイ型どちらの構成でも選択できます。

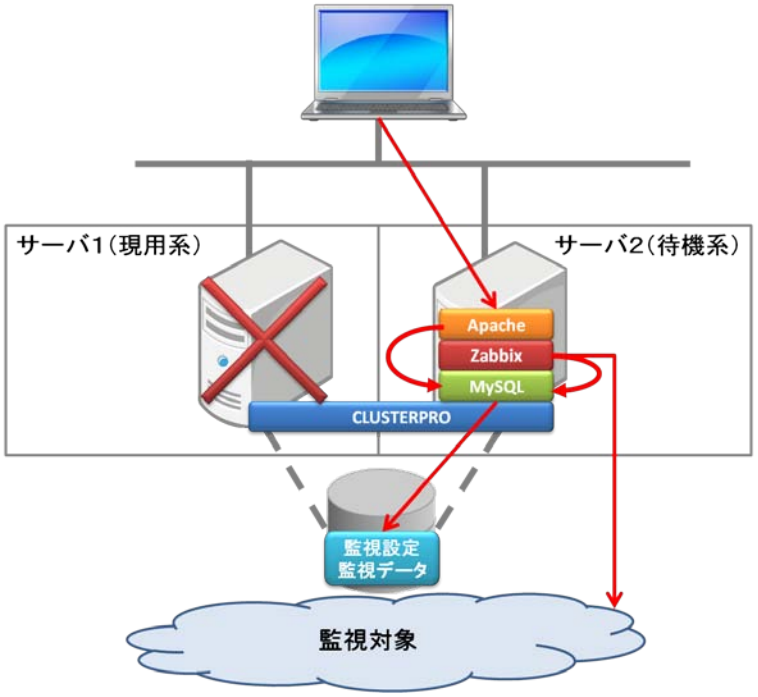
・片方向スタンバイ

下図は、片方向スタンバイ型をCLUSTERPRO環境下でサーバ1を現用系、サーバ2を待機系として動作させるときのイメージ図です。(イメージ図は共有ディスク型クラスタを想定したものですが、ミラーディスク型クラスタも構成可能です。)



片方向スタンバイではZabbixサーバ、MySQL、Apacheが全て同一のサーバで動作します。

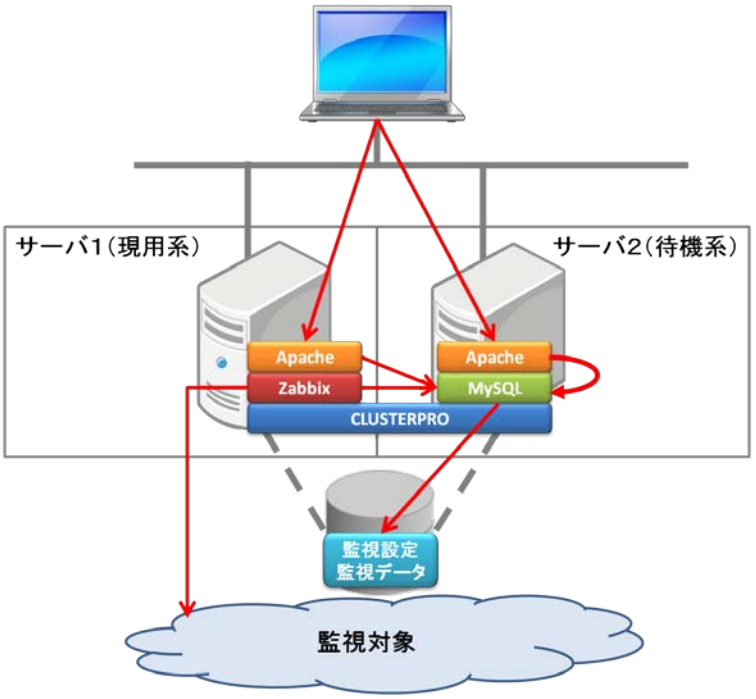
サーバ1に障害が発生すると以下の図のようになります。



障害によりフェイルオーバが完了すると、サーバ2でZabbixサーバ、MySQL、Apacheのサービスが立ち上がり監視が継続されます。フェイルオーバが発生すると、一度全てのサービスが停止／起動されるため、一時的に監視が停止します。

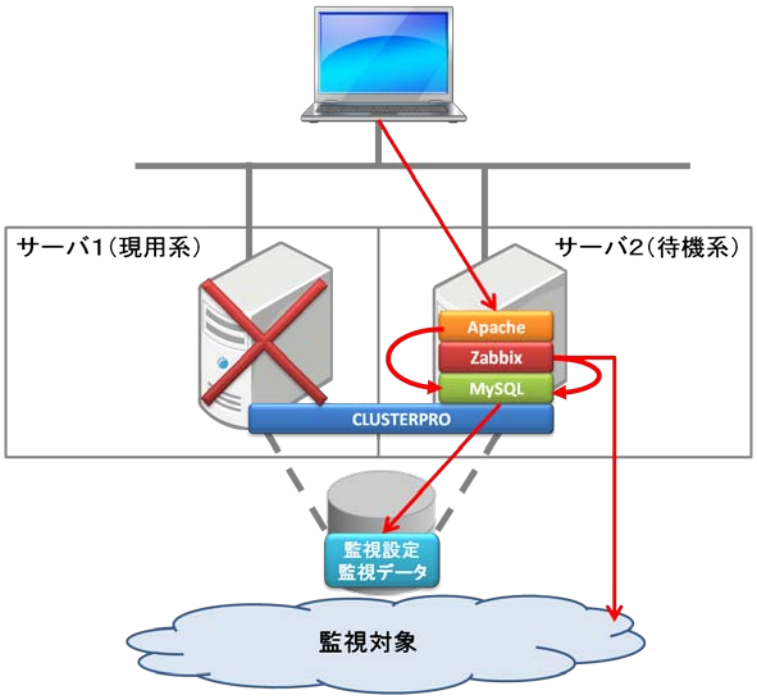
・双方向スタンバイ

下図は、双方向スタンバイ型をCLUSTERPRO環境下で動作させるときのイメージです。（イメージ図は共有ディスク型クラスタを想定したものですが、ミラーディスク型クラスタも構成可能です。）

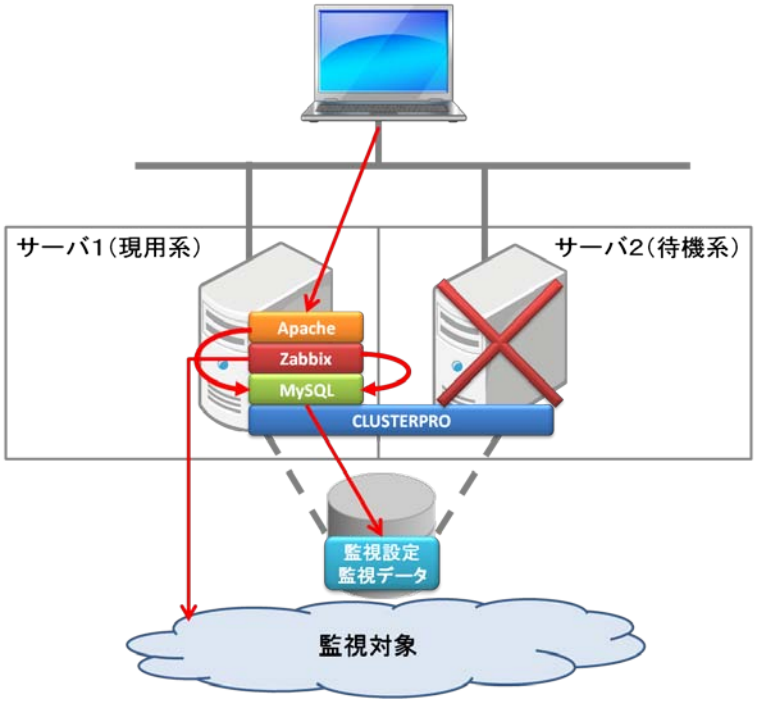


双方向スタンバイでは負荷のかかりやすいMySQLと実際に監視を行うZabbixサーバとを異なるサーバで動作させることができます。また、Apacheを両サーバで動作させることによって、グラフやマップの表示に必要な処理を両サーバで分散でき、効率よく負荷分散できます。

サーバ1で障害が発生すると以下の図のようにサーバ1で動作していたZabbixサーバがフェイルオーバし、監視が継続されます。この際、Zabbixサーバが停止／起動されるため、一時的に監視が停止します。



サーバ2で障害が発生し、フェイルオーバが発生すると、サーバ2で動作していたMySQLがサーバ1に移行します。サーバ2のMySQLに接続していたZabbixサーバは、自動的に再接続を試みMySQLがサーバ1で起動完了した時点で接続します。Zabbixサーバは監視設定や、監視で取得したデータをある程度はメモリ上に保持しているため、MySQLのフェイルオーバが発生しても監視が途切れることはありません。



本書では双方向スタンバイ型クラスタの構築方法について示します。

3. CLUSTERPRO環境下でのZabbixの設定

CLUSTERPRO環境下でZabbixを設定する場合、非クラスタ環境の場合と以下の点が異なります。

・データベースは、必ずCLUSTERPROで管理する共有ディスクまたはミラー用ディスクに格納する必要があります。

・データベースユーザは、ローカルホスト、リモートホストどちらからも接続できるよう設定する必要があります。

・データベースの作成やデータのインポートは、現用系のサーバからのみ実施します。また、待機系サーバでもZabbixからの監視やデータベースを使用することができるようにするために、現用系サーバの設定ファイルを待機系サーバにコピーして、設定を反映します。

本章では、2ノード構成のクラスタでの双方向スタンバイ環境を想定し、説明を行います。

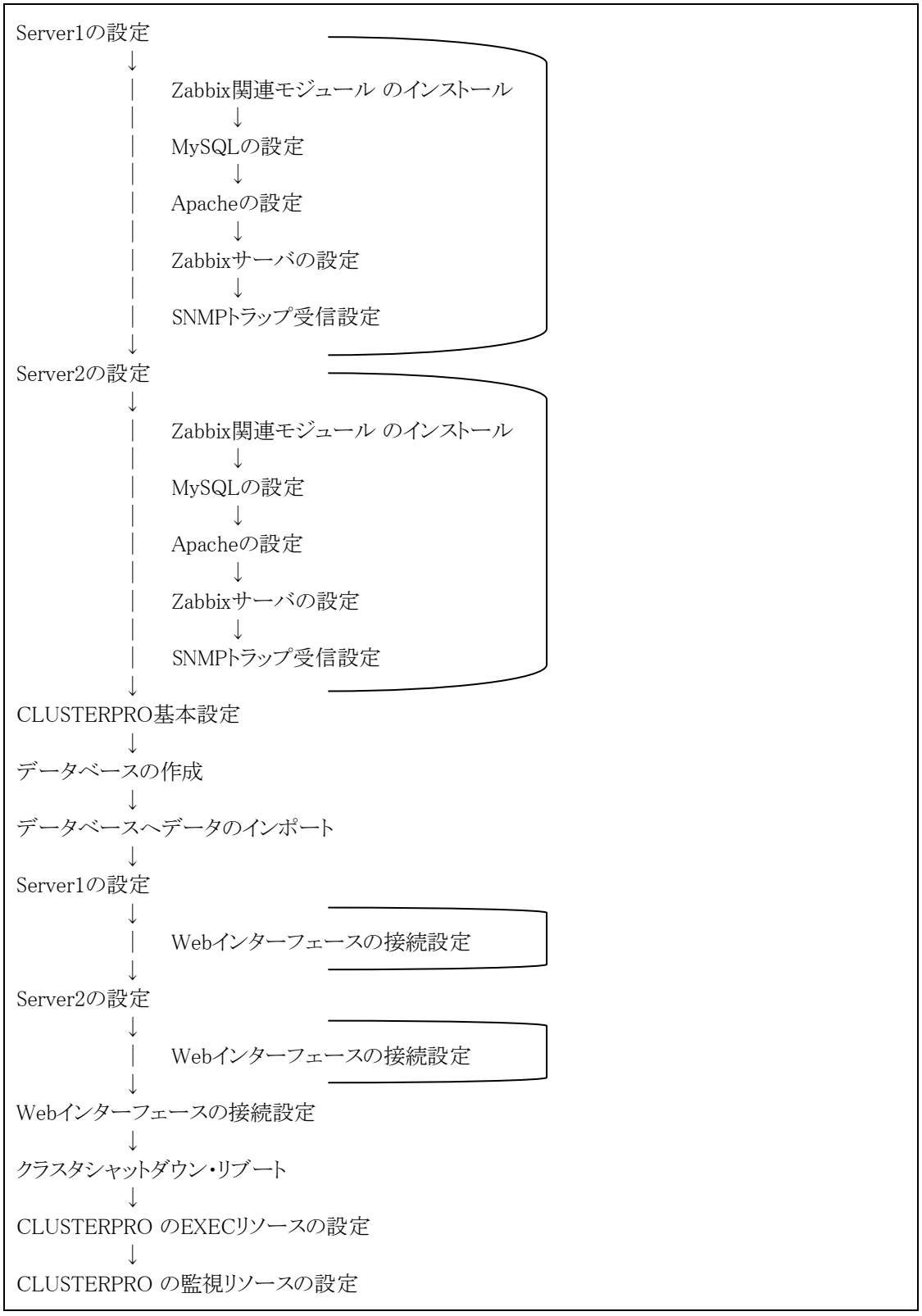
3.1 検証環境

以下の環境で検証しました。

OS	Asianux Server 4 (Hiranya SP1)
アーキテクチャ	x86_64
カーネルバージョン	2.6.32-71.7.1.el6.x86_64
CLUSTERPRO バージョン	clusterpro-3.1.1-1
Zabbix サーババージョン	zabbix-server-1.8.7-2ML6
MySQL バージョン	mysql-5.1.52-1.AXS4.1
Apache バージョン	httpd-2.2.15-9.2.0.1.AXS4

3.2 構築の流れ

双方向の環境を作成する場合は、Server1、Server2の設定を同時に行わないようにします。設定の流れは以下のようになります。



3.3 Zabbix関連モジュールのインストール

RPMコマンドを使ってZabbix、MySQL、Apacheをインストールします。

ここでは以下のパッケージをインストールしました。

zabbix-1.8.7-2ML6.x86_64.rpm

zabbix-server-1.8.7-2ML6.x86_64.rpm

zabbix-server-mysql-1.8.7-2ML6.x86_64.rpm

zabbix-web-1.8.7-2ML6.x86_64.rpm

zabbix-web-mysql-1.8.7-2ML6.x86_64.rpm

zabbix-agent-1.8.7-2ML6.x86_64.rpm

fping-2.4b2-16.AXS4.x86_64.rpm

iksemel-1.4-2.1.AXS4.x86_64.rpm

OpenIPMI-2.0.16-12.AXS4.x86_64.rpm

OpenIPMI-libs-2.0.16-12.AXS4.x86_64.rpm

net-snmp-5.5-31.AXS4.x86_64.rpm

net-snmp-libs-5.5-31.AXS4.x86_64.rpm

mysql-5.1.52-1.AXS4.1.x86_64.rpm

mysql-server-5.1.52-1.AXS4.1.x86_64.rpm

perl-DBD-MySQL-4.013-3.AXS4.x86_64.rpm

unixODBC-2.2.14-11.AXS4.x86_64.rpm

httpd-2.2.15-9.2.0.1.AXS4.x86_64.rpm

httpd-tools-2.2.15-9.2.0.1.AXS4.x86_64.rpm

php-5.3.3-3.AXS4.x86_64.rpm

php-gd-5.3.3-3.AXS4.x86_64.rpm

php-xml-5.3.3-3.AXS4.x86_64.rpm

php-mysql-5.3.3-3.AXS4.x86_64.rpm

php-mbstring-5.3.3-3.AXS4.x86_64.rpm

php-common-5.3.3-3.AXS4.x86_64.rpm

php-pdo-5.3.3-3.AXS4.x86_64.rpm

php-cli-5.3.3-3.AXS4.x86_64.rpm

php-bcmath-5.3.3-3.AXS4.x86_64.rpm

apr-util-ldap-1.3.9-3.AXS4.1.x86_64.rpm

apr-util-1.3.9-3.AXS4.1.x86_64.rpm

apr-1.3.9-3.2.0.2.AXS4.x86_64.rpm

libXpm-3.5.8-2.AXS4.x86_64.rpm

lm_sensors-libs-3.1.1-10.AXS4.x86_64.rpm

libtool-ltdl-2.2.6-15.5.AXS4.x86_64.rpm

3.4 MySQLの設定

Zabbixで利用するDBの設定を行います。テキストエディタで以下のようにファイルを修正します。

・/etc/my.cnf

```
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
# Default to using old password format for compatibility with mysql 3.x
# clients (those using the mysqlclient10 compatibility package).
old_passwords=1

# Disabling symbolic-links is recommended to prevent assorted security risks;
# to do so, uncomment this line:
# symbolic-links=0

##add by Zabbix
default-storage-engine=InnoDB
default-character-set=utf8
skip-character-set-client-handshake
innodb_file_per_table
innodb_buffer_pool_size=XXXM(物理メモリの50%を指定します)
innodb_log_file_size=64M
innodb_log_files_in_group=2
max_connections=512
thread_cache_size=512
max_allowed_packet=16MB

[mysqld_safe]
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
```

本設定は全クラスタメンバで必要となります。1つのサーバで設定を行い、設定ファイルを他のサーバにコピーしてください。

3.5 Apacheの設定

Zabbixで利用するApacheの設定を行います。テキストエディタで以下のようにファイルを修正します。

・/etc/httpd/conf.d/zabbix.conf

```
#
# Zabbix monitoring system php web frontend
#

Alias /zabbix /usr/share/zabbix

<Directory "/usr/share/zabbix">
    Options FollowSymLinks
    AllowOverride None
    Order allow,deny
    Allow from all

    php_value max_execution_time 600
    php_value date.timezone Asia/Tokyo
    php_value memory_limit 256M
    php_value post_max_size 32M
    php_value upload_max_filesize 16M
    php_value max_input_time 600
    php_value mbstring.func_overload 6
</Directory>

<Directory "/usr/share/zabbix/include">
    Order deny,allow
    Deny from all
    <files *.php>
        Order deny,allow
        Deny from all
    </files>
</Directory>

<Directory "/usr/share/zabbix/include/classes">
    Order deny,allow
    Deny from all
    <files *.php>
        Order deny,allow
        Deny from all
    </files>
</Directory>
```

本設定は全クラスタメンバで必要となります。1つのサーバで設定を行い、設定ファイルを他のサーバにコピーしてください。

3.6 Zabbixサーバの設定

Zabbixサーバの設定を行います。テキストエディタで以下のようにファイルを修正します。

・/etc/zabbix/zabbix_server.conf

```
LogFile=/var/log/zabbix/zabbix_server.log
LogFileSize=0
PidFile=/var/run/zabbix/zabbix_server.pid
DBHost=<MySQLグループのfip>
DBName=zabbix
DBUser=zabbix
DBPassword=zabbix
DBSocket=/var/lib/mysql/mysql.sock
AlertScriptsPath=/etc/zabbix/ alertscripts
ExternalScripts=/etc/zabbix/ externalscripts
```

本設定は全クラスタメンバで必要となります。1つのサーバで設定を行い、設定ファイルを他のサーバにコピーしてください。

3.7 SNMPトラップ受信設定

SNMPトラップ受信設定ZabbixではSNMPトラップ受信を監視する場合、OS付属のsnmptrapdを利用します。テキストエディタで以下のようにファイルを修正します。

・/etc/snmp/snmptrapd.conf

```
# Example configuration file for snmptrapd
#
# No traps are handled by default, you must edit this file!
#
# authCommunity   log,execute,net public
# traphandle SNMPv2-MIB::coldStart     /usr/bin/bin/my_great_script cold
authCommunity log,execute,net public
traphandle default /bin/bash /etc/zabbix/snmptrap.sh
```

snmp受信時に実行するスクリプトを作成します。SNMPトラップを受信した際には、本スクリプトが実行され、Zabbixサーバにメッセージが送信されます。テキストエディタで以下のようにファイルを作成します。

・/etc/zabbix/snmptrap.sh

```
#!/bin/bash
# CONFIGURATION
ZABBIX_SERVER="<ZabbixグループのFIP>";
ZABBIX_PORT="10051";
ZABBIX_SENDER="/usr/bin/zabbix_sender";
KEY="<メッセージを格納するアイテムのキー>";
HOST="<メッセージを送信するホスト名>";

# Execute
```

```
read hostname
read ip
read uptime
read oid
read address
read community
read enterprise
read string

oid=`echo $oid|cut -f2 -d' '`
address=`echo $address|cut -f2 -d' '`
community=`echo $community|cut -f2 -d' '`
enterprise=`echo $enterprise|cut -f2 -d' '`

oid=`echo $oid|cut -f11 -d'.'`
community=`echo $community|cut -f2 -d'"'`

str="$hostname $address $community $enterprise $oid $string"

# Output
$ZABBIX_SENDER -z "$ZABBIX_SERVER" -p $ZABBIX_PORT -s "$HOST" -k $KEY -o "$str"
#echo $ZABBIX_SENDER -z $ZABBIX_SERVER -p $ZABBIX_PORT -s $HOST -k $KEY -o "$str"
>> /tmp/zabbix_sender
```

本設定は全クラスタメンバで必要となります。1つのサーバで設定を行い、設定ファイルを他のサーバにコピーしてください。

3.8 CLUSTERPROの基本設定

CLUSTERPROのBuilderを使用して、Zabbix、MySQLの運用に使用するフェイルオーバグループを作成します。フェイルオーバグループには以下のリソースが必要となります。

・フローティングIPリソース
・ディスクリソース または ミラーディスクリソース

なお、この時点ではexecリソースは登録しないでください。

上記のリソースを追加する手順については、CLUSTERPRO X 3.1 for Linux インストール&設定ガイド「第5章 クラスタ構成情報を作成する」をご参照ください。

・クラスタシステム設定

クラスタシステム設定		
クラスタ構成	クラスタ名	ZabbixCluster
	サーバ数	2
1台目のサーバ	サーバ名	server1
	パブリックのIPアドレス	10.0.0.11
	インタコネクトのIPアドレス	192.168.0.1
		10.0.0.11

	ミラーディスクコネクト	192.168.0.1
		10.0.0.11
2台目のサーバ	サーバ名	server2
	パブリックのIPアドレス	10.0.0.12
	インタコネクトのIPアドレス	192.168.0.2
		10.0.0.12
	ミラーディスクコネクト	192.168.0.2
		10.0.0.12
ネットワークパーティション解決IP		10.0.0.1

・Zabbixグループ

Failoverグループ	
グループ名	Zabbix
デフォルト起動サーバ	Server1→Server2
起動待ち合わせ	MySQL
停止待ち合わせ	なし
起動属性	自動起動
フェイルオーバ属性	自動フェイルオーバ／起動可能なサーバ設定に従う
フェイルバック属性	手動フェイルバック
フェイルオーバ排他属性	排他なし
floating ip resource	
グループリソース名	fip-Zabbix
IPアドレス	10.0.0.13
活性リトライ	5回
フェイルオーバー	1回
活性最終動作	何もしない（次のリソースを活性しない）
非活性リトライ	0回
非活性最終動作	クラスタデーモン停止とOSシャットダウン
依存するリソース	既存の依存関係に従う

・MySQLグループ

Failoverグループ	
グループ名	MySQL
デフォルト起動サーバ	Server2→Server1
起動待ち合わせ	なし
停止待ち合わせ	なし
起動属性	自動起動
フェイルオーバ属性	自動フェイルオーバ／起動可能なサーバ設定に従う
フェイルバック属性	（手動フェイルバック）
フェイルオーバ排他属性	排他なし

floating ip resource	
グループリソース名	fip-MySQL
IP アドレス	10.0.0.14
活性リトライ	5 回
フェイルオーバー	1 回
活性最終動作	何もしない（ 次のリソースを活性しない）
非活性リトライ	0 回
非活性最終動作	クラスタデーモン停止と OS シャットダウン
依存するリソース	既存の依存関係に従う
mirror disk resource	
グループリソース名	md-MySQL
マウントポイント	/var/lib/mysql
データパーティション	/dev/sdb4
クラスタパーティション	/dev/sdb3
ディスクデバイス名	/dev/NMP2
ファイルシステム	ext4
活性リトライ	0 回
フェイルオーバー	1 回
活性最終動作	何もしない（ 次のリソースを活性しない）
非活性リトライ	0 回
非活性最終動作	クラスタデーモン停止と OS シャットダウン
依存するリソース	既存の依存関係に従う

・Apache-Server1グループ

Failover グループ	
グループ名	Apache-Server1
デフォルト 起動サーバ	Server1
起動待ち合わせ	Zabbix
停止待ち合わせ	なし
起動属性	自動起動
フェイルオーバ属性	手動フェイルオーバ
フェイルバック属性	自動フェイルバック
フェイルオーバ排他属性	排他なし

・Apache-Server2グループ

Failover グループ	
グループ名	Apache-Server2
デフォルト 起動サーバ	Server2
起動待ち合わせ	Zabbix
停止待ち合わせ	なし

起動属性	自動起動
フェイルオーバ属性	手動フェイルオーバ
フェイルバック属性	自動フェイルバック
フェイルオーバ排他属性	排他なし

フェイルオーバグループを作成し、クラスタ構成情報をアップロードしてサーバに反映させた後、フェイルオーバグループを現用系サーバで起動します。

3.9 データベースの作成

本作業はMySQLグループが稼働している(ミラーディスクリソースが活性している)サーバで作業を行います。

Zabbixで使用するデータベースの作成をします。
まずmysqldを起動します。

```
# service mysqld start
```

mysqldの起動が完了したら、DBを作成します。

```
# mysql -uroot
> create database zabbix;
> grant all privileges on zabbix.* to zabbix@"%" identified by 'zabbix';
> grant all privileges on zabbix.* to zabbix@localhost identified by 'zabbix';
> flush privileges;
```

DB作成が完了したら、必要なデータをインポートします。

```
# mysql -uzabbix -pzabbix zabbix < /usr/share/doc/zabbix-server-1.8.7/schema/mysql.sql
# mysql -uzabbix -pzabbix zabbix < /usr/share/doc/zabbix-server-1.8.7/data/data.sql
# mysql -uzabbix -pzabbix zabbix < /usr/share/doc/zabbix-server-1.8.7/data/images_mysql.sql
```

以上でデータのインポートは完了です。

3.10 Webインターフェースの接続設定

Zabbixでは設定、管理をすべてWebのインターフェースで行います。そのため、Webインターフェースの接続設定を行う必要があります。

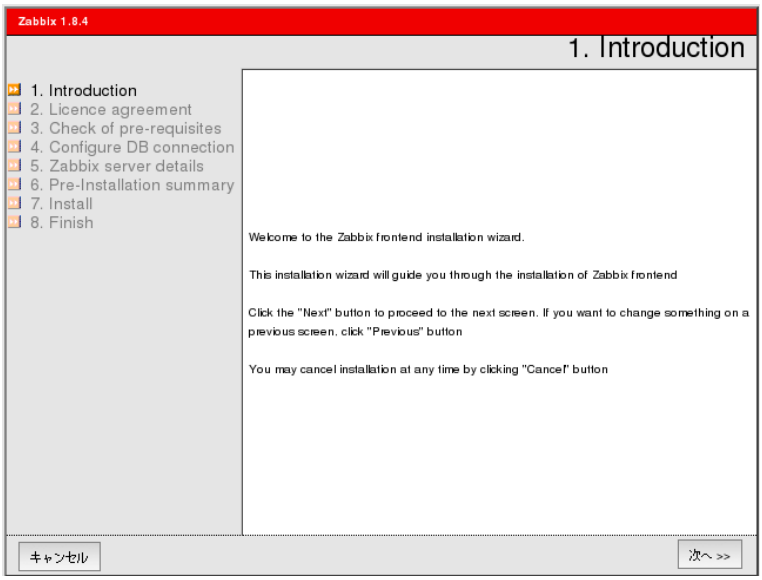
まずhttpdの起動をします。

```
# service httpd start
```

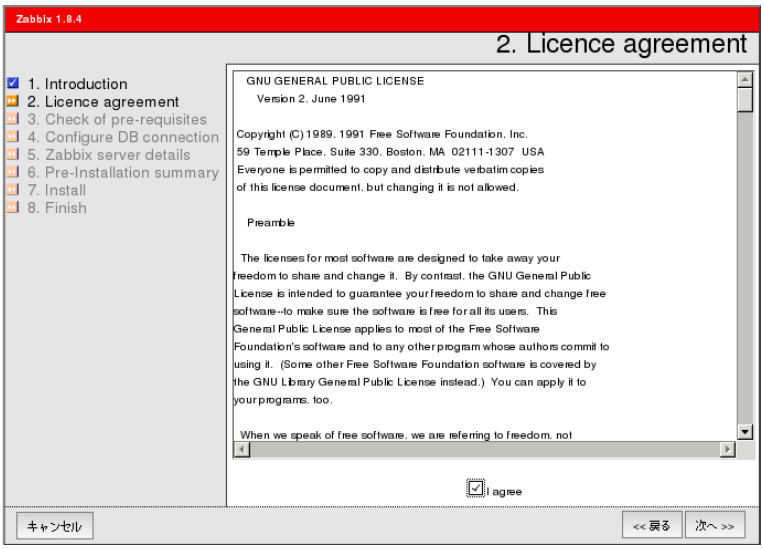
httpdの起動が完了したら、Webブラウザで以下のURLにアクセスします。

```
http://<サーバのIPアドレス>/zabbix
```

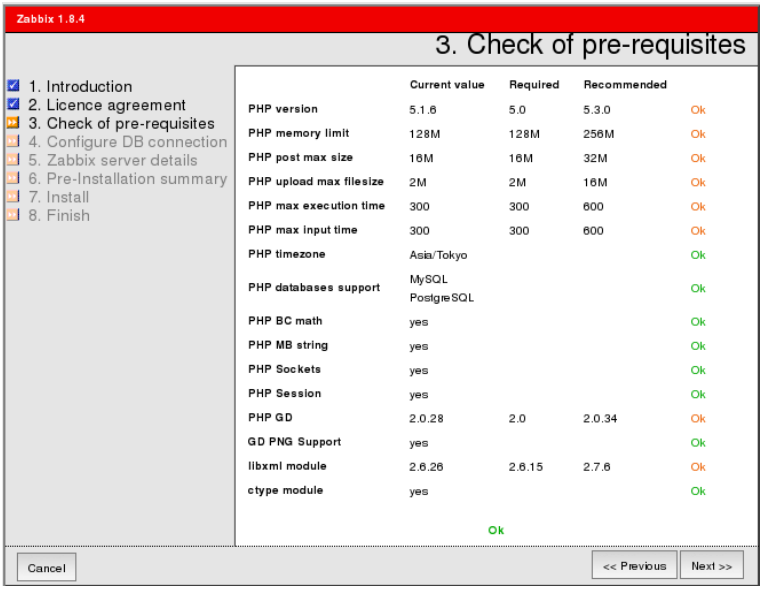
上記に正しくアクセスできると以下のような画面が表示されます。



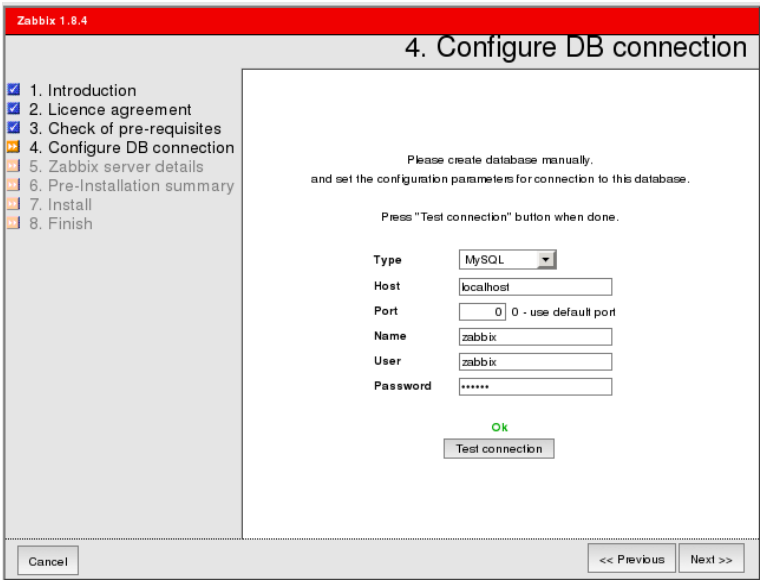
[次へ]を選択します。



ライセンス許諾画面へ移ります。
[I agree]を選択し、[次へ]を選択します。

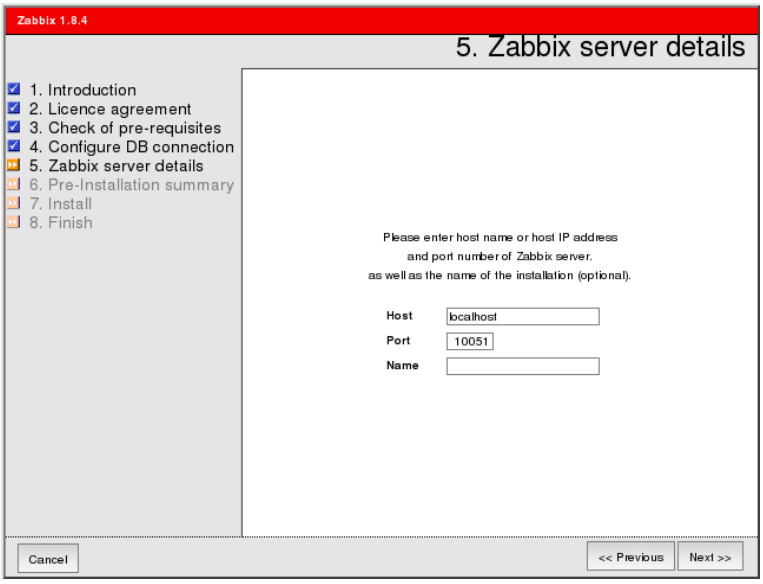


Zabbixの仕様要求画面へ移ります。すべてOKとなっていることを確認し[次へ]を選択します。



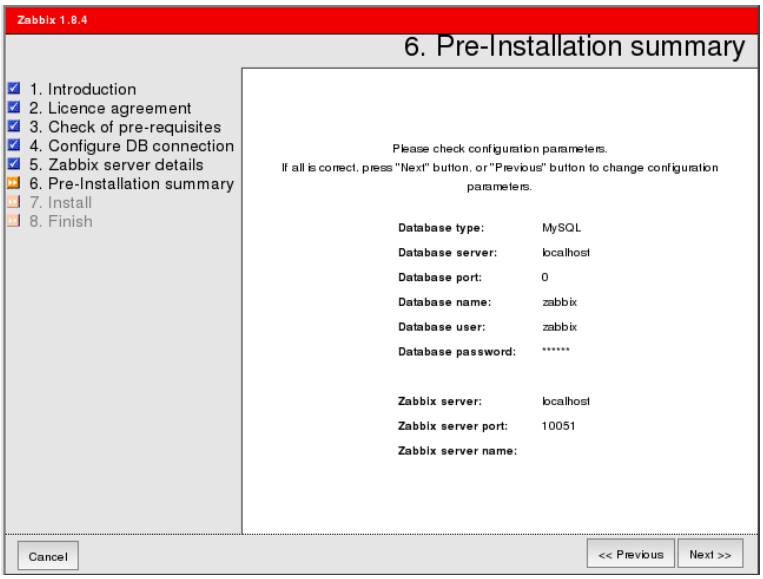
DB接続設定をします。使用DB、DBの稼働ホスト、接続ポート、DB名、接続ユーザ名、パスワードを設定します。ここでは、HostにMySQLグループのFIPを指定してください。

値の入力後、[Test connection]をクリックし、OKが表示されたら[次へ]を選択します。



Zabbixサーバの接続設定画面に移ります。ここではZabbixサーバが動作するHostを指定します。

HostにはZabbixグループのFIPを指定します。値を入植したら[次へ]を選択します。

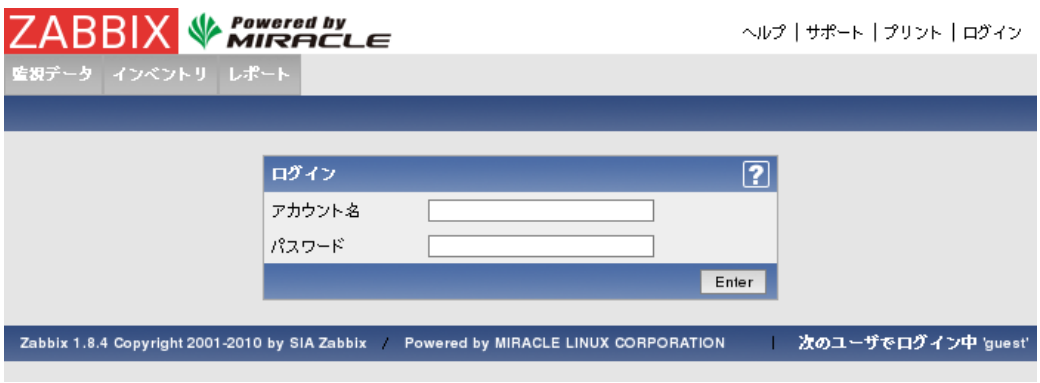


設定確認画面に移ります。内容を確認の上、[次へ]を選択します。

最終接続確認画面へ移ります。OKと表示されていることを確認し、[次へ]を選択します。

以上ですべての設定は終了です。[Finish]を選択します。

以下のようなZabbixのログイン画面へ移ります。



以上でWebインターフェースの接続設定は完了です。本設定は全クラスタメンバで必要となります。1つのサーバで設定を行い、設定ファイル(/etc/zabbix/zabbix.conf.php)を他のサーバにコピーしてください。

設定が完了したら、起動したサービスを停止します。

```
# service httpd stop
# service mysqld stop
```

3.11 CLUSTERPROのEXECリソースの設定

CLUSTERPROのBuilderを使用して、Zabbix、MySQLの起動・停止用スクリプトを実行するEXECリソース、ApacheのEXECリソースを含むフェイルオーバグループを追加します。

(1) CLUSTERPROのBuilderのツリービューのグループ名のアイコンを右クリックし、[リソースの追加]を選択します。
(2) [タイプ]には、[execute resource]を指定し、リソース名を設定して[次へ]を選択します。
(3) [この製品で作成したスクリプト]を選択し、start.shおよびstop.sh を選択してスクリプトの内容を編集します。スクリプト内容については「付録 サンプルスクリプト」のように記述します。スクリプトの作成ができたら、画面に従ってEXECリソースの追加を実施してください。

・Zabbixグループ

execute resource	
グループリソース名	exec-zabbix-server
起動スクリプト	/etc/zabbix/zabbix-server.clp start
停止スクリプト	/etc/zabbix/zabbix-server.clp stop
活性リトライ	5 回
フェイルオーバー	1 回
活性最終動作	何もしない(次のリソースを活性しない)
非活性リトライ	0 回
非活性最終動作	クラスタデーモン停止と OS シャットダウン
依存するリソース	既存の依存関係に従う
execute resource	
グループリソース名	exec-snmptrapd
起動スクリプト	/sbin/service snmptrapd start
停止スクリプト	/sbin/service snmptrapd stop
活性リトライ	1 回
フェイルオーバー	1 回
活性最終動作	何もしない(次のリソースを活性しない)
非活性リトライ	0 回
非活性最終動作	クラスタデーモン停止と OS シャットダウン
依存するリソース	exec-zabbix-server

・MySQLグループ

execute resource	
グループリソース名	exec-MySQL
起動スクリプト	/etc/zabbix/start_mysqld.sh
停止スクリプト	/sbin/service mysqld stop
活性リトライ	1 回
フェイルオーバー	1 回
活性最終動作	何もしない（ 次のリソースを活性しない）
非活性リトライ	0 回
非活性最終動作	クラスタデーモン停止と OS シャットダウン
依存するリソース	既存の依存関係に従う

・Apache-Server1グループ

execute resource	
グループリソース名	exec-httpd-Server1
起動スクリプト	start.sh
停止スクリプト	stop.sh
活性リトライ	1 回
フェイルオーバー	0 回
活性最終動作	何もしない（ 次のリソースを活性しない）
非活性リトライ	0 回
非活性最終動作	何もしない（ 次のリソースを活性しない）
依存するリソース	既存の依存関係に従う

・Apache-Server2グループ

execute resource	
グループリソース名	exec-httpd-Server2
起動スクリプト	start.sh
停止スクリプト	stop.sh
活性リトライ	1 回
フェイルオーバー	0 回
活性最終動作	何もしない（ 次のリソースを活性しない）
非活性リトライ	0 回
非活性最終動作	何もしない（ 次のリソースを活性しない）
依存するリソース	既存の依存関係に従う

3.12 CLUSTERPROの監視リソースの設定

**(注)本書ではCLUSTERPRO X Database Agent 3.1 for Linux、
CLUSTERPRO X Internet Server Agent 3.1 for Linuxを利用しています。**

CLUSTERPROのBuilderを使用して、監視リソースを追加します。

CLUSTERPROのBuilderのツリービューの[Monitors]を右クリックし、[モニタリソースの追加]を選択して[モニタリソースの定義]画面を表示します。
[タイプ]ボックスで下記一覧の監視リソースを選択します。
各種監視リソースの詳細は、CLUSTERPRO X 3.1 for Linuxリファレンスガイド「第5章 モニタリソースの詳細」をご参照ください。

・モニタリソース

process name monitor					
モニタリソース名	psw-snmptrapd				
インターバル	60 秒				
タイムアウト	60 秒				
リトライ回数	1 回				
監視開始待ち	0 秒				
監視タイミング	□	常時	■	活性時	exec-snmptrapd
監視プロセス名	/usr/sbin/snmptrapd -Lsd -p /var/run/snmptrapd.pid				
回復対象	exec-snmptrapd				
活性リトライ	1 回				
フェイルオーバー	1 回				
活性最終動作	何もしない				
process name monitor					
モニタリソース名	psw-Zabbix				
インターバル	60 秒				
タイムアウト	60 秒				
リトライ回数	1 回				
監視開始待ち	0 秒				
監視タイミング	□	常時	■	活性時	exec-zabbix-server
監視プロセス名	zabbix_server -c /etc/zabbix/zabbix_server.conf				
回復対象	exec-zabbix-server				
活性リトライ	1 回				
フェイルオーバー	1 回				
活性最終動作	何もしない				
MySQL monitor					
モニタリソース名	mysqlw				
インターバル	60 秒				
タイムアウト	120 秒				
リトライ回数	2 回				

監視開始待ち	60 秒					
監視タイミング	□	常時	■	活性時	exec-MySQL	
データベース名	zabbix					
IP アドレス	127.0.0.1					
ポート 番号	3306					
ユーザー	zabbix					
パスワード	zabbix					
監視テーブル名	mysqlwatch					
ストレージエンジン	○	ARCHIVE	○	BDB	○	CSV
	○	EXAMPLE	○	FEDERATED	○	HEAP
	○	ISAM	●	InnoDB	○	MEMORY
	○	MERGE	○	MyISAM	○	NDBCLUSTER
ライブラリパス	/usr/lib64/mysql/libmysqlclient.so.16.0.0					
回復対象	exec-MySQL					
活性リトライ	1 回					
フェイルオーバー	1 回					
活性最終動作	何もしない					
http monitor						
モニタリソース名	httpw-Server1					
インターバル	60 秒					
タイムアウト	10 秒					
リトライ回数	3 回					
監視開始待ち	0 秒					
監視タイミング	□	常時	■	活性時	exec-httpd-Server1	
接続先	localhost					
ポート 番号	80					
リクエスト URI	/zabbix					
プロトコル	■	http	□	https		
回復対象	exec-httpd-Server1					
活性リトライ	1 回					
フェイルオーバー	0 回					
活性最終動作	何もしない					
http monitor						
モニタリソース名	httpw-Server2					
インターバル	60 秒					
タイムアウト	10 秒					
リトライ回数	3 回					
監視開始待ち	0 秒					
監視タイミング	□	常時	■	活性時	exec-httpd-Server2	
接続先	localhost					
ポート 番号	80					
リクエスト URI	/zabbix					

プロトコル	■	http	□	https
回復対象	exec-httpd-Server2			
活性リトライ	1 回			
フェイルオーバー	0 回			
活性最終動作	何もしない			

3.13 クラスタシャットダウン・リブート

CLUSTERPROのCluster Managerからクラスタシャットダウン・リブートを実行します。左ツリーのクラスタを選択し、右クリック→[リブート]を選択します。

以上でZabbixクラスタの構築は完了です。OS再起動後、WebManaferに接続し、クラスタ状態が正常となっていることを確認してください。

4. 付録　サンプルスクリプト

双方向スタンバイ型Zabbixクラスタを構築するために必要なスクリプト のサンプルです。

4.1 exec-zabbix-server リソース

Zabbixサーバを起動／停止するためのスクリプトです。zabbix-server付属の標準の起動スクリプトでは起動時にDB接続できなかった場合正常終了してしまうため、Zabbixサーバ起動時、停止時ともに以下のようなスクリプトを使用します。

・/etc/zabbix/zabbix-server.clp

```
#! /bin/sh
#
# chkconfig: - 85 15
# description: zabbix server daemon
#

### BEGIN INIT INFO
# Provides: zabbix
# Required-Start: $local_fs $network
# Required-Stop: $local_fs $network
# Default-Start:
# Default-Stop: 0 1 2 3 4 5 6
# Short-Description: start and stop zabbix server
# Description: Zabbix Server
### END INIT INFO

# zabbix details
if [ -x /usr/sbin/zabbix_server ]; then
    ZABBIX=zabbix_server
elif [ -x /usr/sbin/zabbix_server_mysql ]; then
    ZABBIX=zabbix_server_mysql
elif [ -x /usr/sbin/zabbix_server_pgsql ]; then
    ZABBIX=zabbix_server_pgsql
elif [ -x /usr/sbin/zabbix_server_sqlite3 ]; then
    ZABBIX=zabbix_server_sqlite3
else
    exit 5
fi

CONF=/etc/zabbix/zabbix_server.conf
PIDFILE=/var/run/zabbix/zabbix_server.pid

# Source function library.
. /etc/rc.d/init.d/functions

# Source networking configuration.
```

```
. /etc/sysconfig/network

# Check that networking is up.
[ ${NETWORKING} = "no" ] && exit 0

[ -e $CONF ] || exit 6

RETVAL=0

case "$1" in
   start)
      echo -n "Starting zabbix server: "

DBNAME=$(grep ^DBName $CONF |cut -d'=' -f2|sed -e 's/^ *//' |sed -e "s/ *¥$//")
DBUSER=$(grep ^DBUser $CONF |cut -d'=' -f2|sed -e 's/^ *//' |sed -e "s/ *¥$//")
DBPASS=$(grep ^DBPassword $CONF |cut -d'=' -f2|sed -e 's/^ *//' |sed -e "s/ *¥$//")
DBHOST=$(grep ^DBHost $CONF |cut -d'=' -f2|sed -e 's/^ *//' |sed -e "s/ *¥$//")
if [ "$DBHOST" = "" ]; then
DBHOST="localhost"

fi

echo "select 1" |mysql -N -u $DBUSER --password=$DBPASS -h $DBHOST -D $DBNAME
1>/dev/null

if [ $? -ne 1 ]; then
            daemon $ZABBIX -c $CONF
            RETVAL=$?
            echo
            [ $RETVAL -eq 0 ] && touch /var/lock/subsys/zabbix
else
            sleep 30
            echo_failure
                        echo
                        exit 1
            fi
   ;;
   stop)
      echo -n "Shutting down zabbix server: "
      killproc $ZABBIX
      RETVAL=$?
      echo
      [ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/zabbix
      ;;
   restart)
      $0 stop
      $0 start      RETVAL=$?
      ;;
   reload)
      echo -n $"Reloading zabbix server: "
      killproc -p $PIDFILE $ZABBIX -HUP
      RETVAL=$?
```

```
        echo
        ;;
    condrestart)
        if [ -f /var/lock/subsys/zabbix ]; then
            $0 stop
            $0 start
        fi
        RETVAL=$?
        ;;
    status)
        status $ZABBIX
        RETVAL=$?
        ;;
    *)
        echo "Usage: $0 {start|stop|restart|condrestart|reload|status}"
        exit 1
        ;;
esac
exit $RETVAL
```

4.2 MySQLグループ

MySQLは標準の起動／停止スクリプトではMySQL異常終了後に起動を実行すると起動スクリプトが異常終了するという問題があります。そのため、MySQL起動時は以下のようなスクリプトを使用します。

・/etc/zabbix/start_mysqld.sh

```
#!/bin/bash
# Source function library.
. /etc/rc.d/init.d/functions

# Source networking configuration.
. /etc/sysconfig/network

exec="/usr/bin/mysqld_safe"
prog="mysqld"

# Set timeouts here so they can be overridden from /etc/sysconfig/mysqld
STARTTIMEOUT=120
STOPTIMEOUT=60
MYOPTIONS=

[ -e /etc/sysconfig/$prog ] && . /etc/sysconfig/$prog

lockfile=/var/lock/subsys/$prog

# extract value of a MySQL option from config files
# Usage: get_mysql_option SECTION VARNAME DEFAULT
# result is returned in $result
```

```
# We use my_print_defaults which prints all options from multiple files,
# with the more specific ones later; hence take the last match.
get_mysql_option(){
        result=`/usr/bin/my_print_defaults "$1" | sed -n "s/^--$2=//p" | tail -n 1`
        if [ -z "$result" ]; then
            # not found, use default
            result="$3"
        fi
}

get_mysql_option mysqld datadir "/var/lib/mysql"
datadir="$result"
get_mysql_option mysqld socket "$datadir/mysql.sock"
socketfile="$result"
get_mysql_option mysqld_safe log-error "/var/log/mysqld.log"
errlogfile="$result"
get_mysql_option mysqld_safe pid-file "/var/run/mysqld/mysqld.pid"
mypidfile="$result"
    [ -x $exec ] || exit 5
    # check to see if it's already running
    RESPONSE=`/usr/bin/mysqladmin --socket="$socketfile" --user=UNKNOWN_MYSQL_USER
ping 2>&1`
    if [ $? = 0 ]; then
        # already running, do nothing
        action $"Starting $prog: " /bin/true
        ret=0
    elif echo "$RESPONSE" | grep -q "Access denied for user"
    then
        # already running, do nothing
        action $"Starting $prog: " /bin/true
        ret=0
    else
        # prepare for start
        touch "$errlogfile"
        chown mysql:mysql "$errlogfile"
        chmod 0640 "$errlogfile"
        [ -x /sbin/restorecon ] && /sbin/restorecon "$errlogfile"
        if [ ! -d "$datadir/mysql" ] ; then
            # First, make sure $datadir is there with correct permissions
            if [ ! -e "$datadir" -a ! -h "$datadir" ]
            then
                mkdir -p "$datadir" || exit 1
            fi
            chown mysql:mysql "$datadir"
            chmod 0755 "$datadir"
            [ -x /sbin/restorecon ] && /sbin/restorecon "$datadir"
            # Now create the database
            action $"Initializing MySQL database: " /usr/bin/mysql_install_db --datadir="$datadir"
--user=mysql
            ret=$?
            chown -R mysql:mysql "$datadir"
            if [ $ret -ne 0 ] ; then
```

```
            return $ret
          fi
        fi
        chown mysql:mysql "$datadir"
        chmod 0755 "$datadir"
        # If startsos
        if [ "$1" = "sos" ] ; then
              MYOPTIONS="$MYOPTIONS --skip-grant-tables --skip-networking"
        fi
        # Pass all the options determined above, to ensure consistent behavior.
        # In many cases mysqld_safe would arrive at the same conclusions anyway
        # but we need to be sure.  (An exception is that we don't force the
        # log-error setting, since this script doesn't really depend on that,
        # and some users might prefer to configure logging to syslog.)
        # Note: set --basedir to prevent probes that might trigger SELinux
        # alarms, per bug #547485
        $exec   --datadir="$datadir" --socket="$socketfile" ¥
              --pid-file="$mypidfile" ¥
              $MYOPTIONS ¥
              --basedir=/usr --user=mysql >/dev/null 2>&1 &
        safe_pid=$!
        # Spin for a maximum of N seconds waiting for the server to come up;
        # exit the loop immediately if mysqld_safe process disappears.
        # Rather than assuming we know a valid username, accept an "access
        # denied" response as meaning the server is functioning.
        ret=0
        TIMEOUT="$STARTTIMEOUT"
        while [ $TIMEOUT -gt 0 ]; do
          RESPONSE=`/usr/bin/mysqladmin --socket="$socketfile"
--user=UNKNOWN_MYSQL_USER ping 2>&1`
          mret=$?
          if [ $mret -eq 0 ]; then
            break
          fi
          # exit codes 1, 11 (EXIT_CANNOT_CONNECT_TO_SERVICE) are expected,
          # anything else suggests a configuration error
          if [ $mret -ne 1 -a $mret -ne 11 ]; then
            echo "$RESPONSE"
            echo "Cannot check for MySQL Daemon startup because of mysqladmin failure."
            ret=1
            break
          fi
          echo "$RESPONSE" | grep -q "Access denied for user" && break
          if ! /bin/kill -0 $safe_pid 2>/dev/null; then
            echo "MySQL Daemon failed to start."
            ret=1
            break
          fi
          sleep 1
          let TIMEOUT=${TIMEOUT}-1
        done
        if [ $TIMEOUT -eq 0 ]; then
```

```
            echo "Timeout error occurred trying to start MySQL Daemon."
            ret=1
        fi
        if [ $ret -eq 0 ]; then
            action $"Starting $prog: " /bin/true
            action $"Starting $prog: " /bin/true
            touch $lockfile
        else
            action $"Starting $prog: " /bin/false
        fi
    fi
    return $ret
```

4.3 Apache-Server%グループ(%=1,2)

Apacheを起動するためのスクリプトです。標準の起動／停止スクリプトではApacheの起動／停止に失敗しても正常終了してしまうため、以下のスクリプトを使用します。

・start.sh

```
#!/bin/bash
# Source function library.
. /etc/rc.d/init.d/functions

if [ -f /etc/sysconfig/httpd ]; then
        . /etc/sysconfig/httpd
fi

HTTPD_LANG=${HTTPD_LANG-"C"}
INITLOG_ARGS=""
apachectl=/usr/sbin/apachectl
httpd=${HTTPD-/usr/sbin/httpd}
prog=httpd
pidfile=${PIDFILE-/var/run/httpd/httpd.pid}
lockfile=${LOCKFILE-/var/lock/subsys/httpd}
RETVAL=0
STOP_TIMEOUT=${STOP_TIMEOUT-10}
USER=apache

echo -n $"Starting $prog: "
LANG=$HTTPD_LANG daemon --pidfile=${pidfile} $httpd $OPTIONS
RETVAL=$?
echo
[ $RETVAL = 0 ] && touch ${lockfile}
```

・stop.sh

```
#!/bin/bash
# Source function library.
. /etc/rc.d/init.d/functions

httpd=${HTTPD-/usr/sbin/httpd}
prog=httpd
pidfile=${PIDFILE-/var/run/httpd/httpd.pid}
lockfile=${LOCKFILE-/var/lock/subsys/httpd}
RETVAL=0
STOP_TIMEOUT=${STOP_TIMEOUT-10}

USER=apache
echo -n $"Stopping $prog: "

if [ `ps auxh | awk -v httpd=$httpd '{if ($11 == httpd) print $0}'|wc -l` -gt 0 ]
then
    if [ -f $pidfile ]
    then
        PARENTPID=`cat /var/run/httpd/httpd.pid`
        EXECUSER=`ps uh -p $PARENTPID|cut -d " " -f1`

        if [ "$EXECUSER" != "root" ]
        then
            kill -9 `ps auxh | awk -v user=$USER -v httpd=$httpd '{if ($1 == user&& $11 == httpd) printf
$2" "}'`
        else
            killproc -p ${pidfile} -d ${STOP_TIMEOUT} $httpd
        fi
    else
        kill -9 `ps auxh | awk -v httpd=$httpd '{if ($11 == httpd) printf $2" "}'`
    fi
fi

RETVAL=$?
echo
[ $RETVAL = 0 ] && rm -f ${lockfile} ${pidfile}
```

以上