# クラスタリング設定

- 1. クラスタリング方式
- 2. Cookieを用いたクラスタリング
  - 2.1. 前提条件
    - 。 2.2. 設定手順
- 3. リレーショナルデータベースを用いたクラスタリング
   3.1. 前提条件
   3.2. 設定手順
- 3.2. 設定子順
   4. memcachedを用いたクラスタリング
- 5. 参考
- A1. keepalivedを使用したクラスタリング設定
   A1.1. Active-Active構成
  - A1.1. Active-Active構成
     A1.2. Active-Standby構成

# 1. クラスタリング方式

- Cookieを用いたクラスタリング
- 認証済みセッション情報をCookieに埋め込むことにより、Shibboleth IdP 3を冗長化する方式です。 • リレーショナルデータベースを用いたクラスタリング
- 認証済みセッション情報をリレーショナルデータベースに保存することにより、Shibboleth IdP 3を冗長化する方式です。 • memcachedを用いたクラスタリング
- 認証済みセッション情報をmemcachedに保存することにより、Shibboleth IdP 3を冗長化する方式です。

# 2. Cookieを用いたクラスタリング

Cookieを用いたクラスタリング方式によるShibboleth IdP 3の冗長化設定について下記に示します。

## 2.1. 前提条件

前提条件は下記の通りです。

- 技術ガイドに従って構築する2台以上のIdPを対象とします。
- マシンのホスト名は下記とします。

サービス提供用ホスト名	idp.example.ac.jp	
1台目の実ホスト名	idp1.example.ac.jp	
2台目の実ホスト名	idp2.example.ac.jp	

## 2.2. 設定手順

## 2.2.1. IdPの構築

下記の2点について変更の上、技術ガイドに従ってIdPを構築してください。

 3. jdk 7、tomcat 7をインストールする > 6. httpd の設定 にあるServerNameの設定 idp1.example.ac.jpのような実ホスト名ではなく、サービス提供用ホスト名 idp.example.ac.jpを設定します。

```
/etc/httpd/conf/httpd.conf の修正
(省略)
ServerName idp.example.ac.jp:443 ←サービス提供用ホスト名
(省略)
```

/etc/httpd/conf.d/ssl.conf の修正

```
(省略)
<VirtualHost _default_:443>
(省略)
ServerName idp.example.ac.jp:443 ←サービス提供用ホスト名
ProxyPass /idp/ ajp://localhost:8009/idp/ ←追加
(省略)
```

 4. Shibbolethのインストール > 2. インストール にあるHostNameの入力 idp1.example.ac.jpのような実ホスト名ではなく、サービス提供用ホスト名 idp.example.ac.jpを設定します。

### install.shの実行

```
Source (Distribution) Directory: [/root/PKG/shibboleth-identity-provider-3.2.0]
[Enter] ←入力なし
Installation Directory: [/opt/shibboleth-idp]
[Enter] ←入力なし
Hostname: [idp1.example.ac.jp]
idp.example.ac.jp[Enter] ←サービス提供用ホスト名
SAML EntityID: [https://idp.example.ac.jp/idp/shibboleth]
[Enter] ←入力なし
Attribute Scope: [example.ac.jp]
[Enter] ←入力なし ※表示されたスコープが違う場合、設定してください。
TLS Private Key Password: tlspass[Enter] ←任意のパスワード
Re-enter password: tlspass[Enter]
Cookie Encryption Key Password: cookiepass[Enter] ←任意のパスワード
Re-enter password: cookiepass[Enter]
 (省略)
BUILD SUCCESSFUL
Total time: 2 minutes 9 seconds
```

### 2.2.2. /opt/shibboleth/credentials/sealer.jksのコピー

クラスタを構成するIdPでは、同じ/opt/shibboleth/credentials/sealer.jksを使用する必要があります。

scpコマンドやrsyncコマンドなどでidp1.example.ac.jpの/opt/shibboleth/credentials/sealer.jksをidp2.example.ac.jpにコピーします。3台以上でクラ スタリングする場合は3台目以降も同様にコピーします。

### 2.2.3. Tomcatの再起動

Tomcatを再起動します。

Tomatの再起動

# service tomcat7 restart

## 3. リレーショナルデータベースを用いたクラスタリング

リレーショナルデータベース(以下、「RDB」とします)を用いたクラスタリング方式によるShibboleth IdP 3の冗長化設定を下記に示します。

### 3.1. 前提条件

前提条件は下記の通りです。

- 技術ガイドに従って構築する2台以上のIdPを対象とします。
- RDBサーバはIdPサーバとは独立したサーバとします。
- RDBサーバでは、CentOS 6付属のMySQL 5.1が起動しているものとします。MySQLがインストールされていない場合は、yumでインストールしてください。
- マシンのホスト名は下記とします。

サービス提供用ホスト名 idp.example.ac.jp

1台目の実ホスト名	idp1.example.ac.jp
2台目の実ホスト名	idp2.example.ac.jp
RDBサーバのホスト名	db.example.ac.jp

### 3.2. 設定手順

✓ IdPv3はストレージを統一的に扱いますので、3.2.1、3.2.4および3.2.6で行った設定をuApproveJP等で同意情報のストレージとして用いることが可能です。

### 3.2.1. MySQLの設定

db.example.ac.jpのMySQLの設定を行います。

 データベース shibbolethの作成 Shibboleth IdPで使用するデータベース shibbolethを作成します。

データベース shibbolethの作成 db\$ mysql -u root mysql> CREATE DATABASE shibboleth;

2. ユーザ作成

IdPからデータベース shibbolethにアクセスするためのユーザ shibbolethを作成し、データベース shibbolethへの権限を付与します。

ユーザ shibbolethの作成 db\$ mysql -u root mysql> CREATE USER 'shibboleth'@'localhost' IDENTIFIED BY 'shibpassword'; # ←任意のパスワード CREATE USER 'shibboleth'@'idpl.example.ac.jp' IDENTIFIED BY 'shibpassword'; # ←上記と同一のパスワード CREATE USER 'shibboleth'@'idp2.example.ac.jp' IDENTIFIED BY 'shibpassword'; # ←上記と同一のパスワード GRANT INSERT, SELECT, UPDATE, DELETE ON shibboleth.\* TO 'shibboleth'@'localhost'; GRANT INSERT, SELECT, UPDATE, DELETE ON shibboleth.\* TO 'shibboleth'@'idp1.example.ac.jp'; GRANT INSERT, SELECT, UPDATE, DELETE ON shibboleth.\* TO 'shibboleth'@'idp1.example.ac.jp'; GRANT INSERT, SELECT, UPDATE, DELETE ON shibboleth.\* TO 'shibboleth'@'idp2.example.ac.jp';

3. StorageRecordsテーブル作成

JPAStorageServiceが使用するテーブル StorageRecordsを作成します。

```
テーブル StorageRecordsの作成

db$ mysql -u root

mysql>

use shibboleth;

CREATE TABLE `StorageRecords` (

`context` varchar(255) NOT NULL,

`id` varchar(255) NOT NULL,

`value` longtext NOT NULL,

`value` longtext NOT NULL,

`version` bigint(20) NOT NULL,

PRIMARY KEY (`context`,`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

### 3.2.2. IdPの構築

下記の2点について変更の上、技術ガイドに従ってIdPを構築してください。

 3. jdk 7、tomcat 7をインストールする > 6. httpd の設定 にあるServerNameの設定 idp1.example.ac.jpのような実ホスト名ではなく、サービス提供用ホスト名 idp.example.ac.jpを設定します。 /etc/httpd/conf/httpd.confの修正

```
(省略)
ServerName idp.example.ac.jp:443 ←サービス提供用ホスト名
(省略)
```

/etc/httpd/conf.d/ssl.confの修正

```
(省略)
<VirtualHost _default_:443>
(省略)
ServerName idp.example.ac.jp:443 ←サービス提供用ホスト名
ProxyPass /idp/ ajp://localhost:8009/idp/ ←追加
(省略)
```

 4. Shibbolethのインストール > 2. インストール にあるHostNameの入力 idp1.example.ac.jpのような実ホスト名ではなく、サービス提供用ホスト名 idp.example.ac.jpを設定します。

#### install.shの実行

```
Source (Distribution) Directory: [/root/PKG/shibboleth-identity-provider-3.2.0]
[Enter] ←入力なし
Installation Directory: [/opt/shibboleth-idp]
[Enter] ←入力なし
Hostname: [idp1.example.ac.jp]
idp.example.ac.jp[Enter] ←サービス提供用ホスト名
SAML EntityID: [https://idp.example.ac.jp/idp/shibboleth]
[Enter] ←入力なし
Attribute Scope: [example.ac.jp]
[Enter] ←入力なし ※表示されたスコープが違う場合、設定してください。
TLS Private Key Password: tlspass[Enter] ←任意のパスワード
Re-enter password: tlspass[Enter]
Cookie Encryption Key Password: cookiepass[Enter] ←任意のパスワード
Re-enter password: cookiepass[Enter]
 (省略)
BUILD SUCCESSFUL
Total time: 2 minutes 9 seconds
```

### 3.2.3. /opt/shibboleth/credentials/sealer.jksのコピー

クラスタを構成するIdPでは、同じ/opt/shibboleth/credentials/sealer.jksを使用する必要があります。

scpコマンドやrsyncコマンドなどでidp1.example.ac.jpの/opt/shibboleth/credentials/sealer.jksをidp2.example.ac.jpにコピーします。3台以上でクラ スタリングする場合は3台目以降も同様にコピーします。

### 3.2.4. MySQL Connector/Jのインストール

1. IdPにMySQLへのアクセスに必要なMySQL Connector/J (mysql-connector-java.jar)をインストールします。

```
MySQL Connector/Jのインストール
```

```
# yum install mysql-connector-java
```

2. /usr/share/java 配下にインストールされているので、edit-webapp/ 配下のlib ディレクトリにシンボリックリンクを作成し、build.shコマンド を実行してidp.warに含めます。

```
MySQL Connector/Jの配置

# rpm -ql mysql-connector-java
(省略)

/usr/share/java/mysql-connector-java.jar
(省略)

# ln -s /usr/share/java/mysql-connector-java.jar /opt/shibboleth-idp/edit-webapp/WEB-INF/lib/

# /opt/shibboleth-idp/bin/build.sh

Installation Directory: [/opt/shibboleth-idp]

[Enter] ←入力なし

Rebuilding /opt/shibboleth-idp/war/idp.war ...

...done

BUILD SUCCESSFUL

Total time: 3 seconds
```

### 3.2.5. idp.session.StorageServiceの設定変更

idp.sesssion.StorageServiceの設定をshibboleth.JPAStorageServiceに変更します。

```
/opt/shibboleth-idp/conf/idp.properties の修正
(省略)
idp. session. StorageService = shibboleth. JPAStorageService
(省略)
```

### 3.2.6. shibboleth.JPAStorageServiceの設定

3.2.5. idp.session.StorageServiceの設定変更でidp.sesssion.StorageServiceに設定したshibboleth.JPAStorageServiceを定義します。

<bean id="Shibboleth.MySQLDataSource">のp:url, p:username, p:passwordは、3.2.1. MySQLの設定に合わせて設定します。

```
conf/global.xml の修正
<!-- Use this file to define any custom beans needed globally. -->
<bean id="shibboleth.JPAStorageService"</pre>
     class="org.opensaml.storage.impl.JPAStorageService"
     p:cleanupInterval="%{idp.storage.cleanupInterval:PT10M}"
     c:factory-ref="shibboleth.JPAStorageService.entityManagerFactory" />
<bean id="shibboleth.JPAStorageService.entityManagerFactory"</pre>
     class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
    <property name="packagesToScan" value="org.opensaml.storage.impl" />
    <property name="dataSource" ref="shibboleth.MySQLDataSource" />
    <property name="jpaVendorAdapter" ref="shibboleth.JPAStorageService.JPAVendorAdapter" />
    <property name="jpaDialect">
          <bean class="org.springframework.orm.jpa.vendor.HibernateJpaDialect" />
    </property>
</bean>
<bean id="shibboleth.JPAStorageService.JPAVendorAdapter"</pre>
     class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter"
     p:database="MYSQL" />
<bean id="shibboleth.MySQLDataSource"</pre>
     class="org.apache.tomcat.dbcp.dbcp.BasicDataSource"
     p:driverClassName="com.mysql.jdbc.Driver"
     p:url="jdbc:mysql://db.example.ac.jp:3306/shibboleth"
     p:username="shibboleth"
     p:password="shibpassword"
     p:maxActive="10"
     p:maxIdle="5"
     p:maxWait="15000"
     p:testOnBorrow="true"
     p:validationQuery="select 1"
     p:validationQueryTimeout="5" />
```

### 3.2.7. Tomcatの再起動

Tomcatを再起動します。

### Tomatの再起動

# service tomcat7 restart

## 4. memcachedを用いたクラスタリング

memcachedを用いたクラスタリング方式によるShibboleth IdP 3の冗長化設定については、Shibboleth wikiのMemcachedStorageServiceをご参照ください。

## 5. 参考

クラスタリングの設定する上で、参考になるドキュメントを下記に示します。

- [Shibboleth wiki] Identity Provider 3 > DeployerResources > Productionalization > Clustering
- [Shibboleth wiki] Identity Provider 3 > DeployerResources > Configuration > StorageConfiguration

## A1. keepalivedを使用したクラスタリング設定

### A1.1. Active-Active構成

keepalivedを使用しActive-Active構成によるクラスタリング設定について下記に示します。

### A1.1.1. 前提条件

前提条件は下記の通りです。

- IdPサーバから独立したLVSサーバを用意します。
- LVSサーバにてクライアントからのリクエストを受け付け、IdPサーバにリクエストを振り分けます。
- IdPサーバからの応答は直接クライアントに返すDirect Return(DR)とします。
- マシンのホスト名とIPアドレスは下記とします。

	ホスト名	IPアドレス
LVSサーバのホスト	lvs.example.ac.jp	203.0.113.1
サービス提供用ホスト	idp.example.ac.jp	203.0.113.10
1台目の実ホスト	idp1.example.ac.jp	203.0.113.11
2台目の実ホスト	idp2.example.ac.jp	203.0.113.12

### A.1.1.2.IdPサーバの設定手順

LVSサーバから割り振られたリクエストパケットを受信するために、iptablesに下記の設定を行います。

#### iptablesの設定

```
# iptables -t nat -A PREROUTING -d 203.0.113.10 -j REDIRECT
# service iptables save
```

### A1.1.3. LVSサーバの設定手順

1. LVSサーバにkeepalivedとipvsadmをインストールします。

keepalivedのインストール

# yum install keepalived ipvsadm

```
2. keepalivedの設定を行います。
```

```
keepalived.confの設定
```

```
global_defs {
  notification_email {
      root@localhost
  }
  notification_email_from lvs@lvs.example.ac.jp
  smtp_server 127.0.0.1
  smtp_connect_timeout 30
  router_id lvs.example.ac.jp
}
vrrp_instance VI_1 {
  state BACKUP
   interface eth0
  virtual_router_id 51
                          # 他のVRRPの値と重複しないようにすること
  priority 200
  advert_int 1
  authentication {
      auth type PASS
      auth_pass 1111
                          # 任意のパスワード(最大8文字)
  }
  virtual ipaddress {
      203.0.113.10/24 \text{ dev eth}0
  }
  nopreempt
}
virtual_server 203.0.113.10 443 {
  delay_loop 5
   lvs_sched sh
   lvs method DR
  protocol TCP
   real_server 203.0.113.11 443 {
      weight 1
      SSL_GET {
          url {
              path /idp/css/main.css
              status code 200
           }
          connect_port 443
           connect_timeout 3
          nb_get_retry 3
          delay_before_retry 3
      }
  }
   real_server 203.0.113.12 443 {
      weight 1
      SSL_GET {
          url {
              path /idp/css/main.css
              status_code 200
           }
           connect_port 443
          connect_timeout 3
          nb_get_retry 3
          delay_before_retry 3
      }
  }
}
virtual_server 203.0.113.10 8443 {
  delay_loop 5
   lvs_sched sh
   lvs_method DR
  protocol TCP
```

```
real_server 203.0.113.11 8443 {
       weight 1
       SSL_GET {
           url {
              path /idp/css/main.css
               status_code 200
           }
           connect_port 443
           connect_timeout 3
          nb_get_retry 3
           delay_before_retry 3
       }
   }
   real_server 203.0.113.12 8443 {
       weight 1
       SSL GET {
          url {
              path /idp/css/main.css
              status_code 200
          }
           connect_port 443
           connect_timeout 3
           nb_get_retry 3
           delay_before_retry 3
      }
   }
}
```

3. keepalivedを起動します。

### keepalivedの起動

```
# chkconfig keepalived on
# chkconfig --list keepalived
keepalived 0:off 1:off 2:on 3:on 4:on 5:on 6:off
# service keepalived start
```

4. keepalivedが正しく動作しているか確認します。

#### keepalivedの動作確認

# ipvsadm −L −n				
IP Virtual Server version 1.2.1 (s	size=4096	5)		
Prot LocalAddress:Port Scheduler F	lags			
-> RemoteAddress:Port	Forward	Weight	ActiveConn	InActConn
TCP 203.0.113.10:443 sh				
-> 203.0.113.11:443	Local	1	0	0
-> 203.0.113.12:443	Local	1	0	0
TCP 203.0.113.10:8443 sh				
-> 203.0.113.11:8443	Local	1	0	0
-> 203.0.113.12:8443	Local	1	0	0
<pre># ip addr show dev eth0 scope glob</pre>	bal			
2: eth0: <broadcast,multicast,allm< td=""><td>NULTI, UP,</td><td>LOWER_L</td><td>JP&gt; mtu 1500</td><td>0 qdisc pfifo_fast state UP qlen 1000</td></broadcast,multicast,allm<>	NULTI, UP,	LOWER_L	JP> mtu 1500	0 qdisc pfifo_fast state UP qlen 1000
link/ether XX:XX:XX:XX:XX:XX	ord ff:ft	f:ff:ff	:ff:ff	
inet 203.0.113.1/24 brd 203.0.	113.255	scope g	global eth0	
inet 203.0.113.10/24 scope gld	bal seco	ondary e	eth0	

## A1.2. Active-Standby構成

keepalivedを使用してVRRPによるActive-Standby構成によるクラスタリング設定について下記に示します。

## A1.2.1. 前提条件

前提条件は下記の通りです。

- WebサーバやIdPサーバが停止した場合は、keepalivedを停止する動作とします。復旧は手動で行います。
- マシンのホスト名、IPアドレス、VRRPのpriorityは下記とします。

	ホスト名	IPアドレス	<b>VRRP</b> Øpriority
サービス提供用ホスト	idp.example.ac.jp	203.0.113.10	-
1台目の実ホスト	idp1.example.ac.jp	203.0.113.11	200
2台目の実ホスト	idp2.example.ac.jp	203.0.113.12	100

## A1.2.2. IdPサーバの設定手順(idp1, idp2共通)

IdPサーバにkeepalivedとipvsadmをインストールします。

keepalivedのイ	ンス	トール
--------------	----	-----

# yum install keepalived ipvsadm

### A.1.2.3. IdPサーバの設定手順(idp1)

1. keepalivedの設定を行います。

keepalived.confの設定

```
global defs {
  notification_email {
       root@localhost
  }
  notification_email_from vrrp@idp1.example.ac.jp
  smtp_server 127.0.0.1
  smtp_connect_timeout 30
  router_id idp1.example.ac.jp
}
vrrp_instance VI_1 {
  state BACKUP
   interface eth0
  virtual_router_id 51
                          # 他のVRRPの値と重複しないようにすること
  priority 200
  advert_int 1
  authentication {
      \texttt{auth\_type PASS}
      auth_pass 1111
                          # 任意のパスワード(最大8文字)
  }
  virtual_ipaddress {
      203.0.113.10/24 dev eth0
  }
  nopreempt
   smtp_alert
}
virtual_server 203.0.113.10 443 {
  delay_loop 5
   lvs_sched sh
  lvs method NAT
  protocol TCP
   real_server 127.0.0.1 443 {
      weight 1
       notify_down "/sbin/service keepalived stop"
      SSL_GET {
          url {
              path /idp/css/main.css
              status_code 200
          }
          connect_port 443
          connect_timeout 3
          nb_get_retry 3
          delay_before_retry 3
      }
  }
}
```

2. keepalivedを起動します。

keepalivedの起動

```
# chkconfig keepalived on
# chkconfig --list keepalived
keepalived 0:off 1:off 2:on 3:on 4:on 5:on 6:off
# service keepalived start
```

3. keepalivedが正しく動作しているか確認します。

 keepalivedの動作確認

 # ipvsadm -L -n

 IP Virtual Server version 1.2.1 (size=4096)

 Prot LocalAddress:Port Scheduler Flags

 -> RemoteAddress:Port Forward Weight ActiveConn InActConn

 TCP 203.0.113.10:443 sh

 -> 127.0.0.1:443 Local 1 0 0

 # ip addr show dev eth0 scope global

 2: eth0: <BROADCAST, MULTICAST, ALLMULTI, UP, LOWER\_UP> mtu 1500 qdisc pfifo\_fast state UP qlen 1000

 link/ether XX:XX:XX:XX brd ff:ff:ff:ff:ff:ff:ff:ff:ff:inet 203.0.113.11/24 brd 203.0.113.255 scope global eth0

 inet 203.0.113.10/24 scope global secondary eth0

## A.1.2.4. IdPサーバの設定手順(idp2)

1. keepalivedの設定を行います。

keepalived.confの設定

```
global defs {
  notification_email {
       root@localhost
  }
  notification_email_from vrrp@idp2.example.ac.jp
  smtp_server 127.0.0.1
  smtp_connect_timeout 30
  router_id idp2.example.ac.jp
}
vrrp_instance VI_1 {
  state BACKUP
   interface eth0
  virtual_router_id 51
                          # idp1と同じ値とすること
  priority 100
  advert_int 1
  authentication {
      \texttt{auth\_type PASS}
      auth_pass 1111
                           # idp1と同じ値とすること
  }
  virtual_ipaddress {
      203.0.113.10/24 dev eth0
  }
  nopreempt
   smtp_alert
}
virtual_server 203.0.113.10 443 {
  delay_loop 5
   lvs_sched sh
  lvs method NAT
  protocol TCP
   real_server 127.0.0.1 443 {
      weight 1
       notify_down "/sbin/service keepalived stop"
      SSL_GET {
          url {
              path /idp/css/main.css
              status_code 200
          }
           connect_port 443
           connect_timeout 3
          nb_get_retry 3
          delay_before_retry 3
      }
  }
}
```

### 2. keepalivedを起動します。

#### keepalivedの起動

```
# chkconfig keepalived on
# chkconfig --list keepalived
keepalived 0:off 1:off 2:on 3:on 4:on 5:on 6:off
# service keepalived start
```

3. keepalivedが正しく動作しているか確認します。

### keepalivedの動作確認

# ipvsadm -L -n IP Virtual Server version 1.2.1 (size=4096) Prot LocalAddress:Port Scheduler Flags -> RemoteAddress:Port Forward Weight ActiveConn InActConn TCP 203.0.113.10:443 sh -> 127.0.0.1:443 Local 1 0 0 # ip addr show dev eth0 scope global 2: eth0: <BROADCAST, MULTICAST, ALLMULTI, UP, LOWER\_UP> mtu 1500 qdisc pfifo\_fast state UP qlen 1000 link/ether XX:XX:XX:XX brd ff:ff:ff:ff:ff: inet 203.0.113.12/24 brd 203.0.113.255 scope global eth0