Web 三層モデル(2)
参考資料

目次

図の索引

1章 負荷試験

1.1. Grinder

1.1.1. Grinder について

java を使ったテストフレームワークです。
テストコード（シナリオ）を jython を使って書いて実行します。
ブラウザの操作を記憶して、シナリオを自動生成することができます。
詳しくは以下の URL を参考にしてください。

URL: http://grinder.sourceforge.net/

Grinder には、以下の3つの java アプリケーションが含まれています。

□ Grinder(Agent) シナリオを実行することができます。
□ Console Agent に対して開始、終了の指示を出したり、テストの結果を監視することができます。
□ TCPProxy HTTP リクエストを記録して、シナリオを自動生成することができます。

今回は以下の構成で使用します。(図 1:負荷試験の構成図)

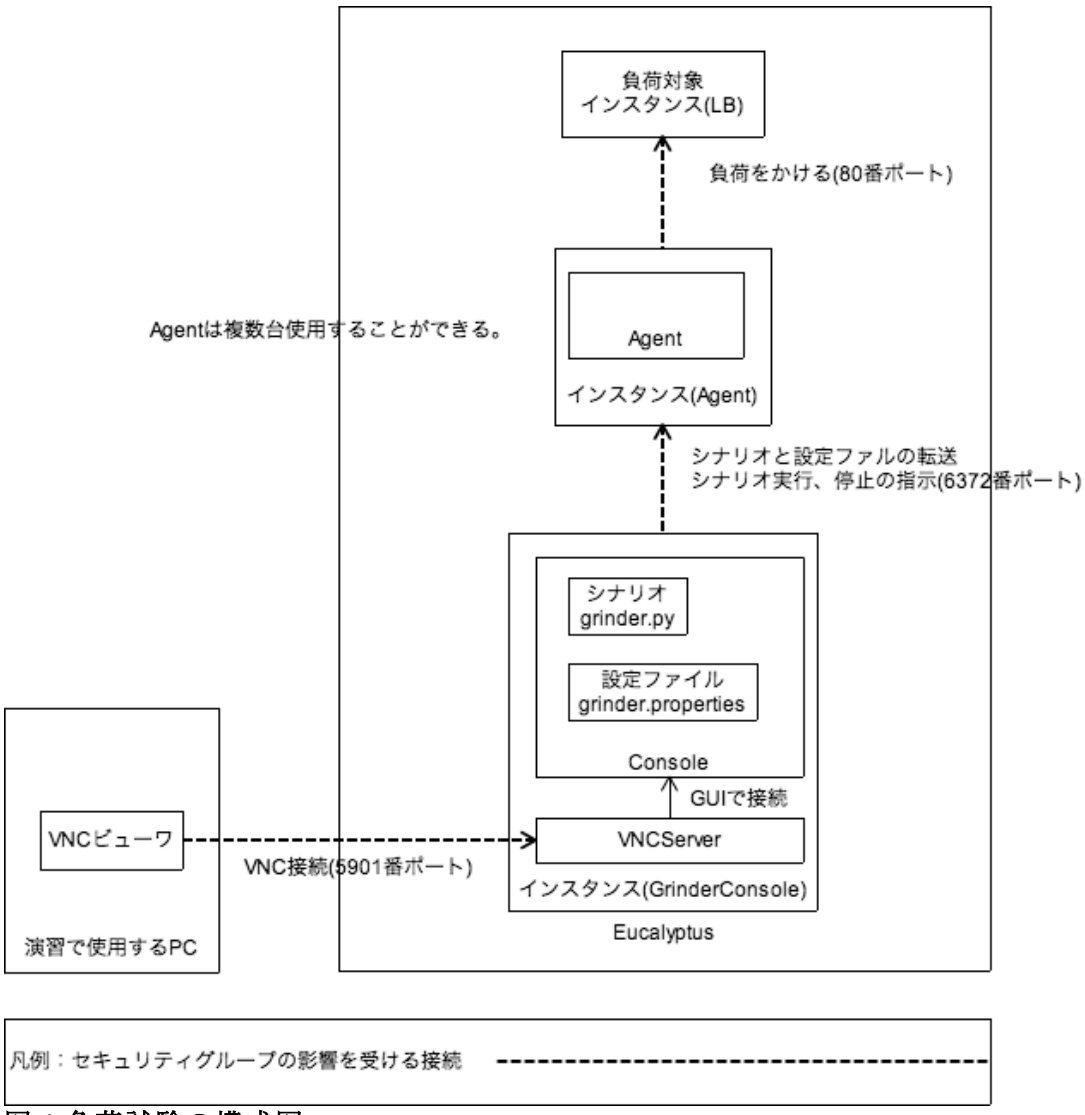

図 1:負荷試験の構成図

1.1.2. Grinder を使ってみる

用意するもの

- ☐ grinder.py(Read 用シナリオ)
- ☐ grinder.py(Write 用シナリオ)
- ☐ ecloud-answer.zip(ファイル)
- ☐ config.properties(ファイル) 前回使用したもの
- ☐ RealVNC(VNC クライアント)

シナリオは「U:¥usr¥デスクトップ」以下においてください。

ecloud-answer は前回演習で行った Apache+Tomcat のスケールアウトスクリプトの解答が入っているプロジェクトです。前回の演習が終わらなかった方は、zip ファイルを解凍して「U:¥usr¥workspace」以下に入れておいてください。
ecloud-answer を使用する場合は前回使用した config.properties を解凍して自分の「プロジェクト名」フォルダの中にあるものを「U:¥usr¥workspace¥ecloud-answer¥src」の中に上書きしておいてく

ださい。

RealVNC をダウンロードしたらダブルクリックしてインストールウィザードに従ってインストールしてください。
なおインストール場所は「U:¥usr」フォルダ以下にインストールしてください。また今回使用するのはクライアントのみなので Server はインストールしなくてよいです。

ecloud-answer を使用する場合は前回演習資料「1.2.2 eclipse 上で起動スクリプトを動かすための設定をする」を参考にインポートしておいてください。

1.1.2.1.負荷対象インスタンスを起動する

まず、負荷をかけるインスタンスを起動します。

デスクトップの eclispe アイコン[eclispe3.4.2(JRE1.6)]から「eclispe」を起動し、パッケージ・エクスプローラーにある「ecloud→src→jp.ac.nii.sample→LaunchCluster」を右クリックし、「実行→java アプリケーション」を実行してインスタンスを立ち上げてください。(図 2:負荷対象インスタンスの起動)

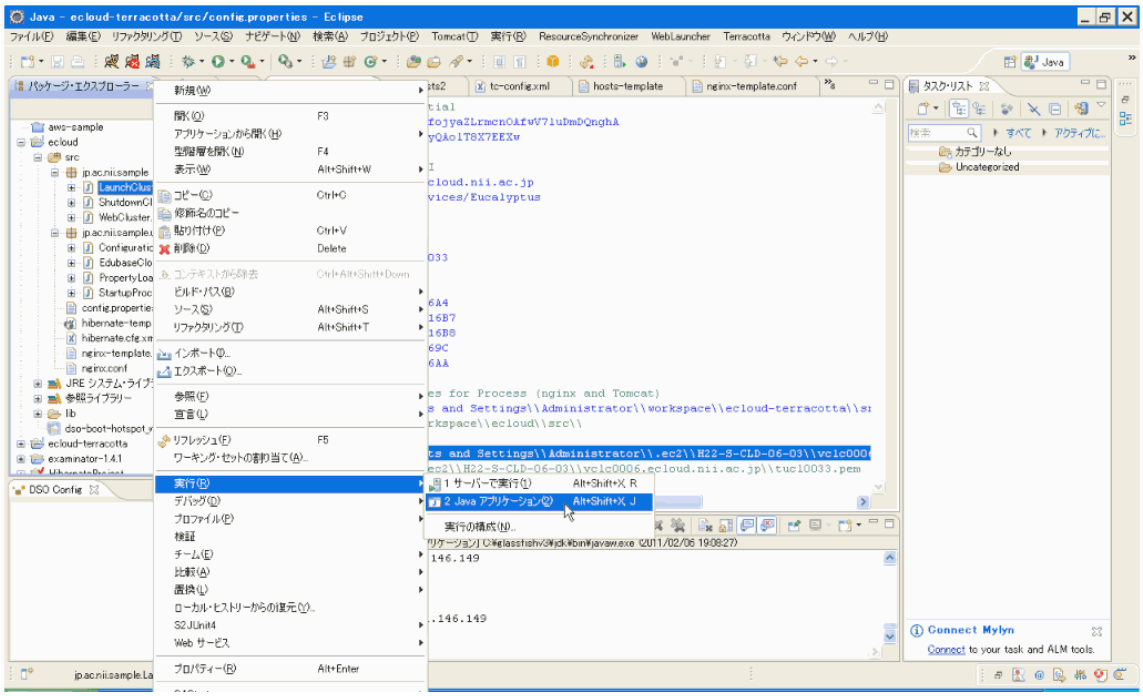

図 2:負荷対象インスタンスの起動

Windows の「スタート→すべてのプログラム→クラウドクライアント→CloudClient」を起動し仮想マシン一覧の misc フォルダの中にインスタンスが立ち上がるのを確認してください。(図 3:インスタンスの起動確認)

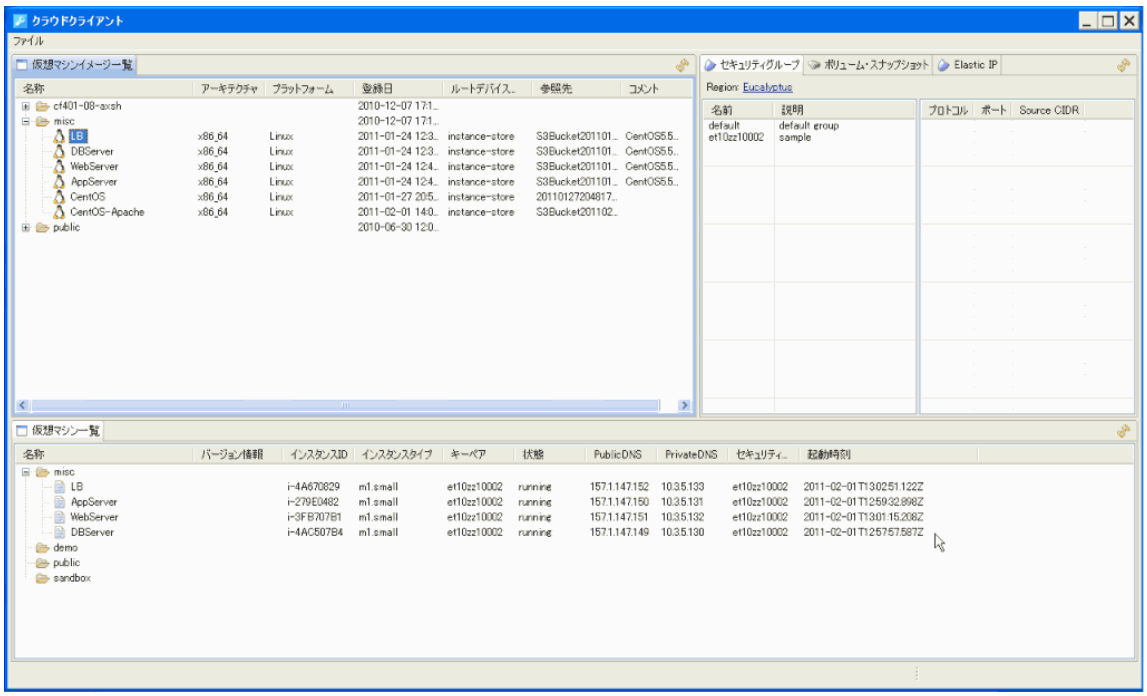

図 3:インスタンスの起動確認

なお、インスタンスの起動には数分かかります。

1.1.2.2.Console を起動する

Windows の「スタート→すべてのプログラム→クラウドクライアント→CloudClient」を起動し仮想マシンイメージ一覧の misc フォルダの中にある「GrinderConsole」を起動します。
以下の内容で、仮想マシンを起動してください。

☐	バージョン	-
☐	インスタンス	m1.small
☐	キーペア	ログインユーザーID
☐	インスタンス数	1
☐	セキュリティグループ	ログインユーザーID

仮想マシン一覧の misc フォルダの中にインスタンスが起動してきたのを確認します。
なおインスタンスの起動には数分かかります。

次に先ほどダウンロードして用意したシナリオを編集します。
Read 用 Write 用ともに同じように修正してください。
以下の当該箇所を変更します。

```
connectionDefaults.defaultHeaders = ¥
  [ NVPair('User-Agent', 'Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 2.0.50727;
InfoPath.1; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729; .NET CLR 1.1.4322; .NET4.0C; .NET4.0E)'),
    NVPair('Accept-Encoding', 'gzip, deflate'),
    NVPair('Accept-Language', 'ja'),
    NVPair('Referer', 'http://[LB のパブリック IP]/sample-kejiban/'),
    NVPair('Accept', 'image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash,
application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, application/x-ms-application,
```

```
application/x-ms-xbap, application/vnd.ms-xpsdocument, application/xaml+xml, */*'), ]

headers0= ¥
  [ ]

url0 = 'http://[LB のパブリック IP]:80'
```

次に編集したシナリオを「GrinderConsole」に転送します。まずは grinder.py(Write 用シナリオ)を転送しましょう。

WinSCP でシナリオを転送するのに必要な鍵の登録を行います。「C:¥Program Files¥PuTTY」フォルダの中にある「pagent.exe」を起動します。(図 4:Pagent の場所)

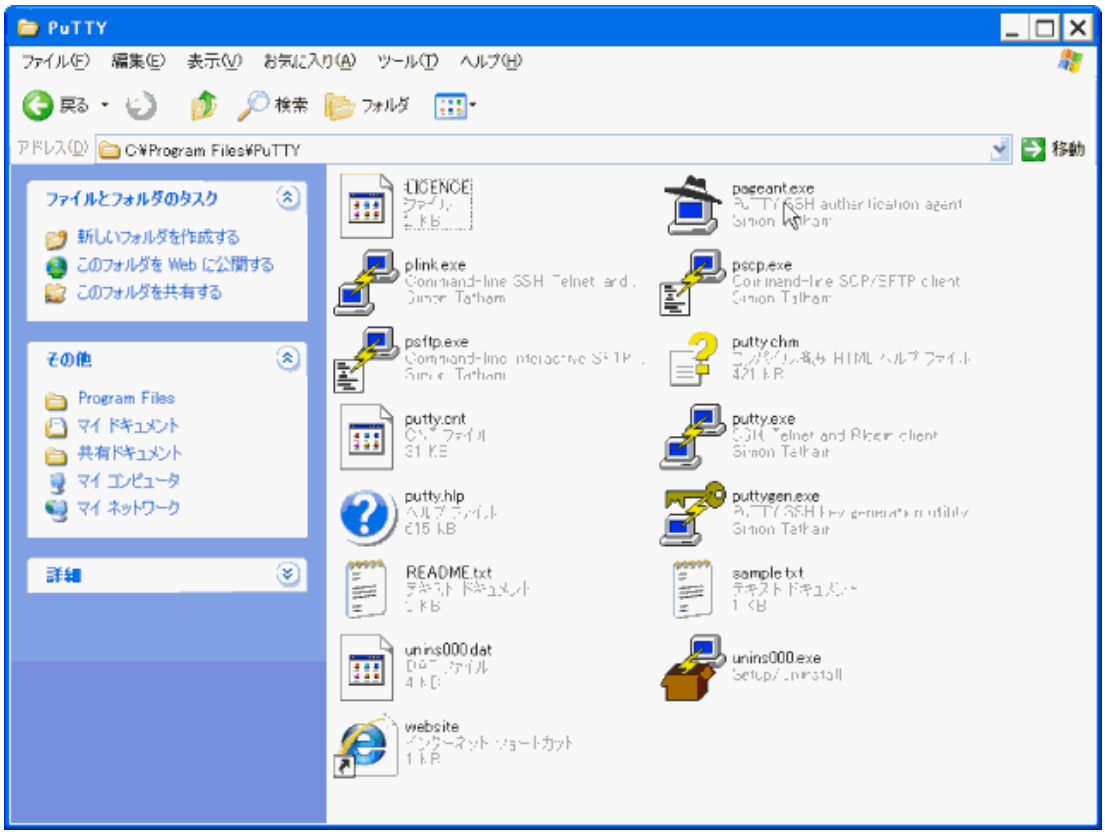

図 4:Pagent の場所

起動すると Windows のタスクバーにアイコンが出ます。

次に Windows のタスクバーに表示されている pagent のアイコンをダブルクリックして pagent の Key List を表示します。
新規に鍵を登録するために「Add Key」ボタンをクリックします。(図 5:秘密鍵のリスト)

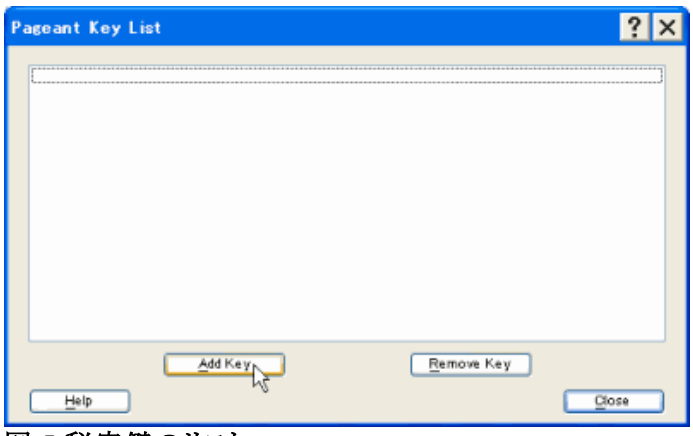

図 5:秘密鍵のリスト

前回作成した「ログイン ID.ppk」という名前の ppk ファイルを選択して「開く」ボタンをクリックします。(図 6:秘密鍵の登録)

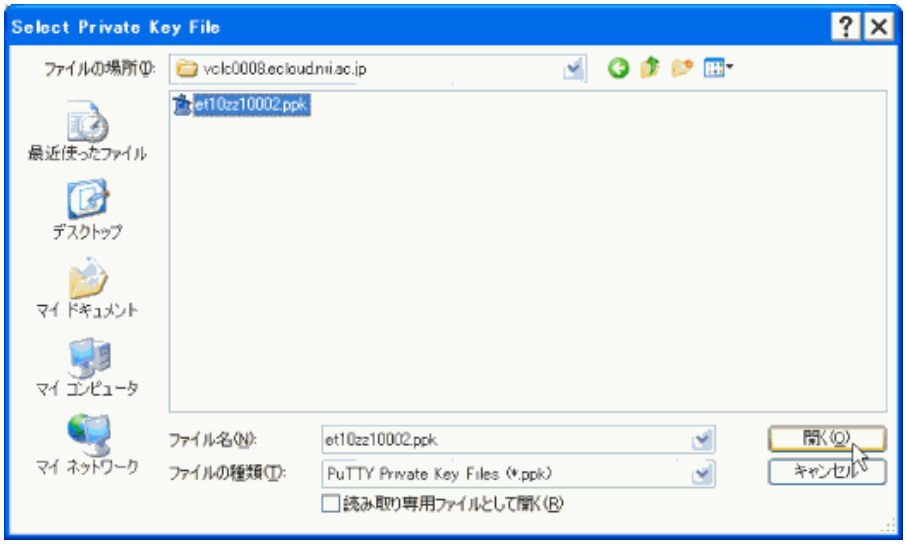

図 6:秘密鍵の登録

鍵が登録されていることを確認します。(図 7:秘密鍵の登録確認)

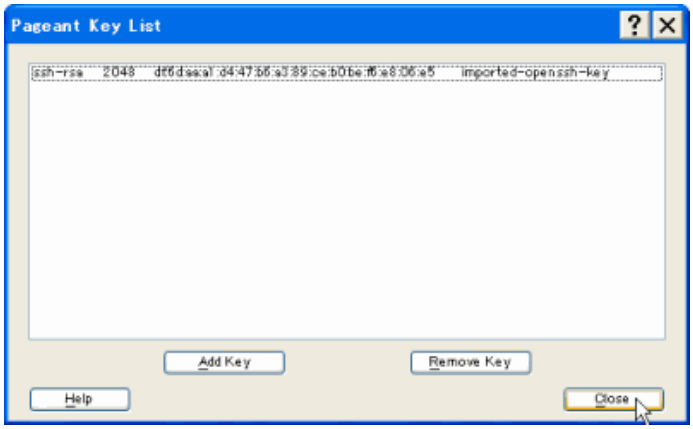

図 7:秘密鍵の登録確認

Windows の「スタート→すべてのプログラム→WinSCP→WinSCP」を起動し、「ホスト名」に「GrinderConsole のパブリック IP」、「ユーザ名」に「root」を入力して、ログインします。(図

8:GrinderConsole へログイン)

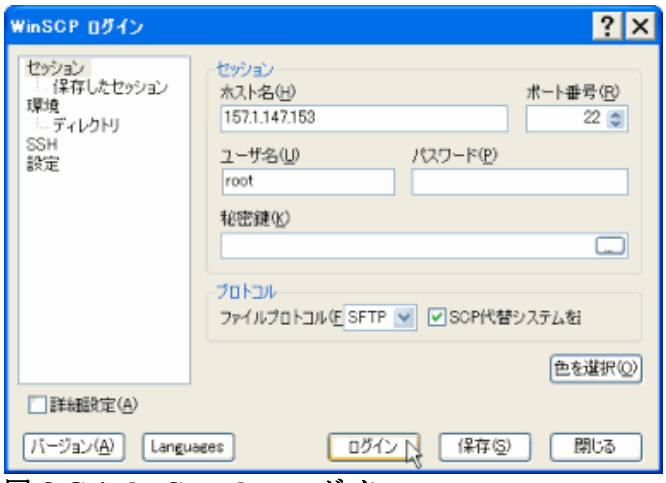

図 8:GrinderConsole へログイン

「/home/vnc/grinder-3.4」フォルダの下に先ほど用意した grinder.py(Write 用シナリオ)を転送します。

次に VNC 接続するためにセキュリティグループの設定を行います。
Windows の「スタート→すべてのプログラム→クラウドクライアント→CloudClient」を起動し、「セキュリティグループ」タブをクリックします。セキュリティグループの名前一覧から「ログインユーザーID」をクリックし、右側の一覧を右クリックして「パーミッションの追加」を選択します。(図 9:パーミッションの追加)

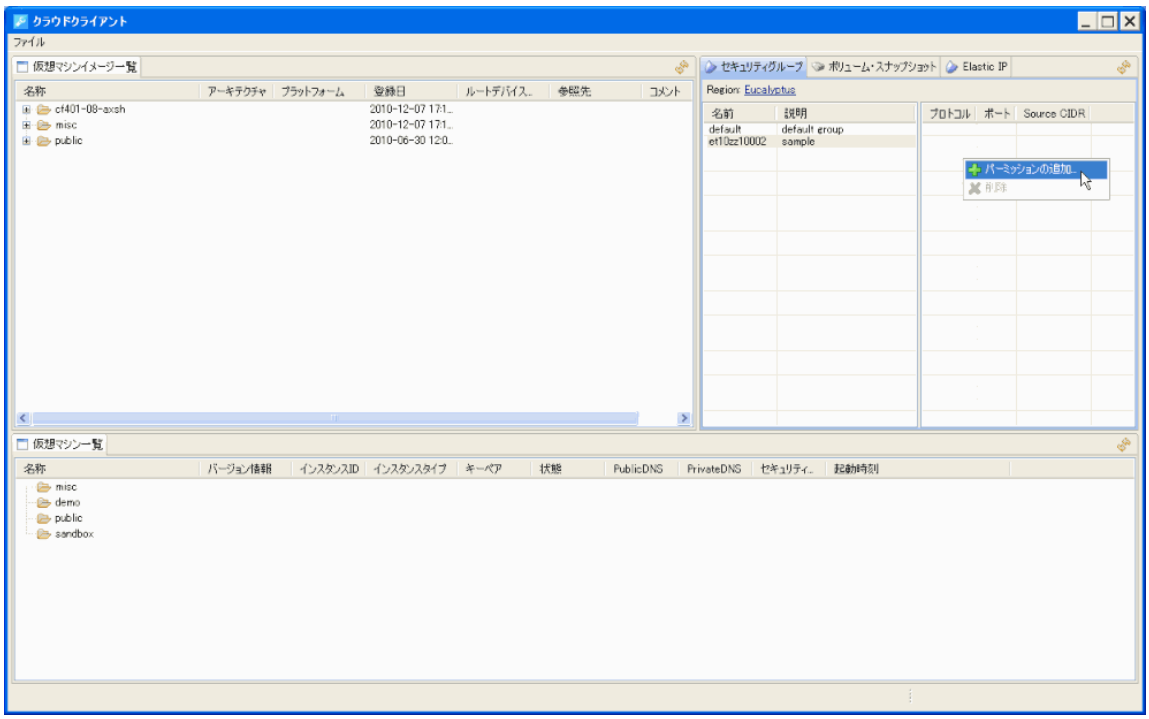

図 9:パーミッションの追加

Protocol に「TCP」、Port or Port Range に「5901」、Network Mask に「0.0.0.0/0」を入力し、「OK」ボタンをクリックします。

Console は GUI で立ち上がってくるので、VNC 接続を行います。
Windows の「スタート→すべてのプログラム→RealVNC→VNC ビューワ4→VNC ビューワの起動」を実行する。

「サーバー名」に「GrinderConsole のパブリック IP:1」を入力し「OK」ボタンをクリックする。(図 10:VNC の接続先サーバーの設定)

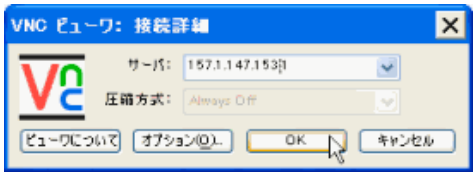

図 10:VNC の接続先サーバーの設定

「パスワード」に「vncvnc」を入力し、ログインする。(図 11:VNC のパスワード入力)

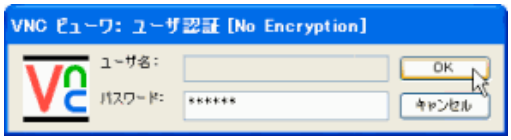

図 11:VNC のパスワード入力

スクリーンセーバが起動していて、パスワード入力を求められる場合は「3edc4rfv」を入力します。(図 12:スクリーンセーバ)

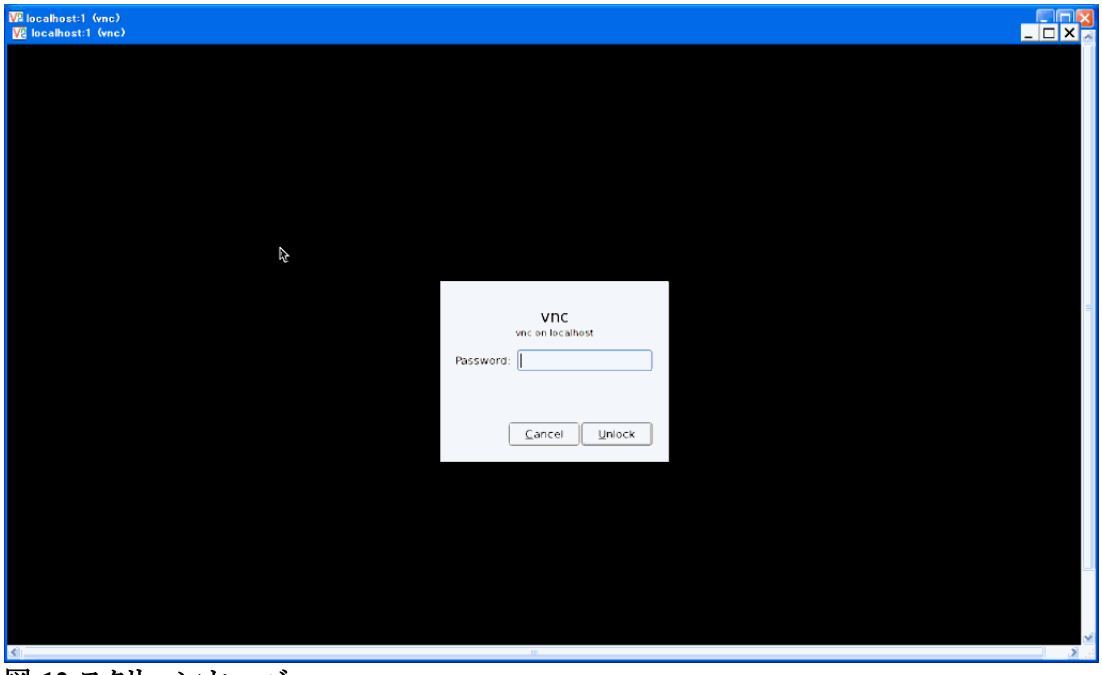

図 12:スクリーンセーバ

「Applications」メニューの「Accessories→Terminal」を起動します。(図 13:Terminal の起動)

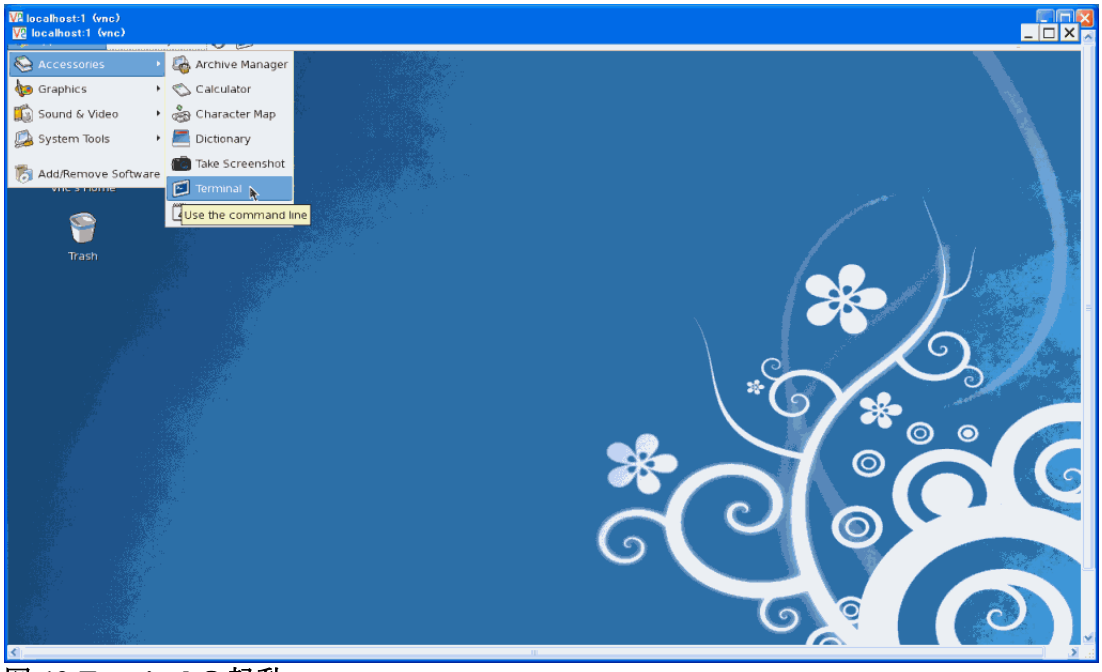

図 13:Terminal の起動

以下のコマンドで Console を起動します。

```
[vnc@localhost ~]$ java -cp grinder-3.4/lib/grinder.jar net.grinder.Console
```

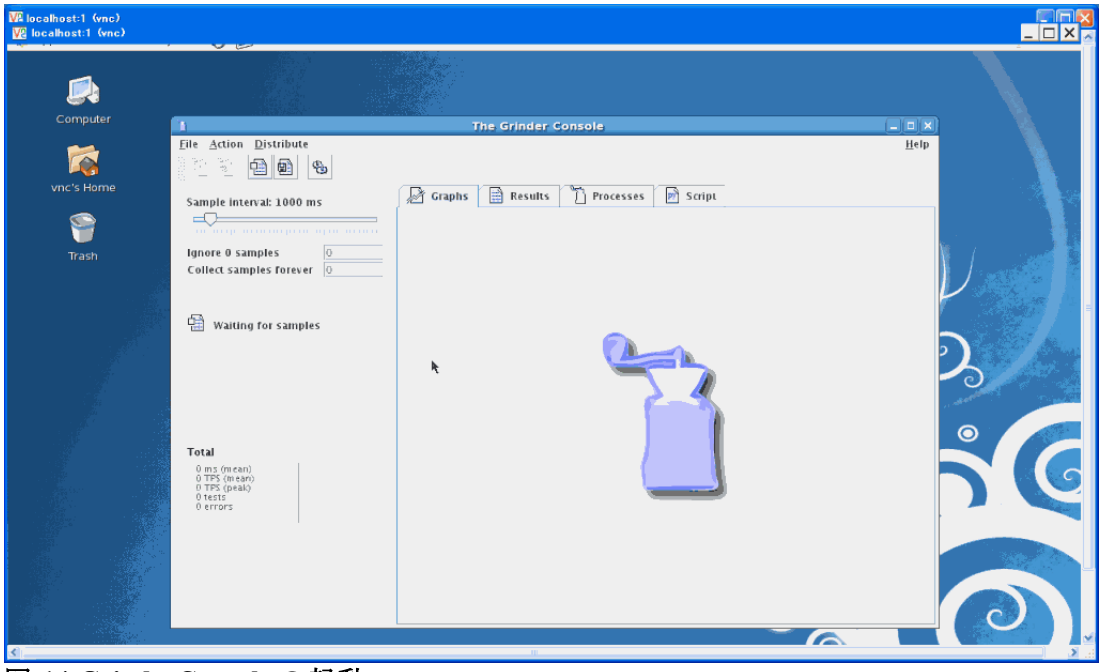

図 14:GrinderConsole の起動

1.1.2.3.Agent を起動する

Windows の「スタート→すべてのプログラム→クラウドクライアント→CloudClient」を起動し、仮想マシンイメージ一覧の misc フォルダにある「GrinderAgent」を右クリックし、「仮想マシンの起動」を実行します。
以下の内容で、仮想マシンを起動してください。

- □ バージョン -
- □ インスタンス m1.small
- □ キーペア ログインユーザーID
- □ インスタンス数 7
- □ セキュリティグループ ログインユーザーID

仮想マシン一覧の misc フォルダの中にインスタンスが起動してきたのを確認します。
なおインスタンスの起動には数分かかります。

仮想マシン一覧の misc フォルダの中にある「GrinderAgent」を右クリックし、「シェルの起動」を選択します。
以下のコマンドを実行し、Agent を起動します。

```
-bash-3.2# java -Dgrinder.consoleHost=[GrinderConsole のパブリック IP] -cp grinder-
3.4/lib/jython.jar:grinder-3.4/lib/grinder.jar net.grinder.Grinder

1/24/11 12:22:01 AM (agent): The Grinder 3.4

1/24/11 12:22:01 AM (agent): connected to console at /157.1.147.149:6372

1/24/11 12:22:01 AM (agent): waiting for console signal
```

「grinder.consoleHost」プロパティには、「GrinderConsole のパブリック IP」を入力してください。

1.1.2.4.Agent の設定をする

「GrinderConsole」が起動したら「Script」タブの「New」をクリックし設定ファイルを作成します。(図 15:GrinderAgent の新規設定ファイルの作成)

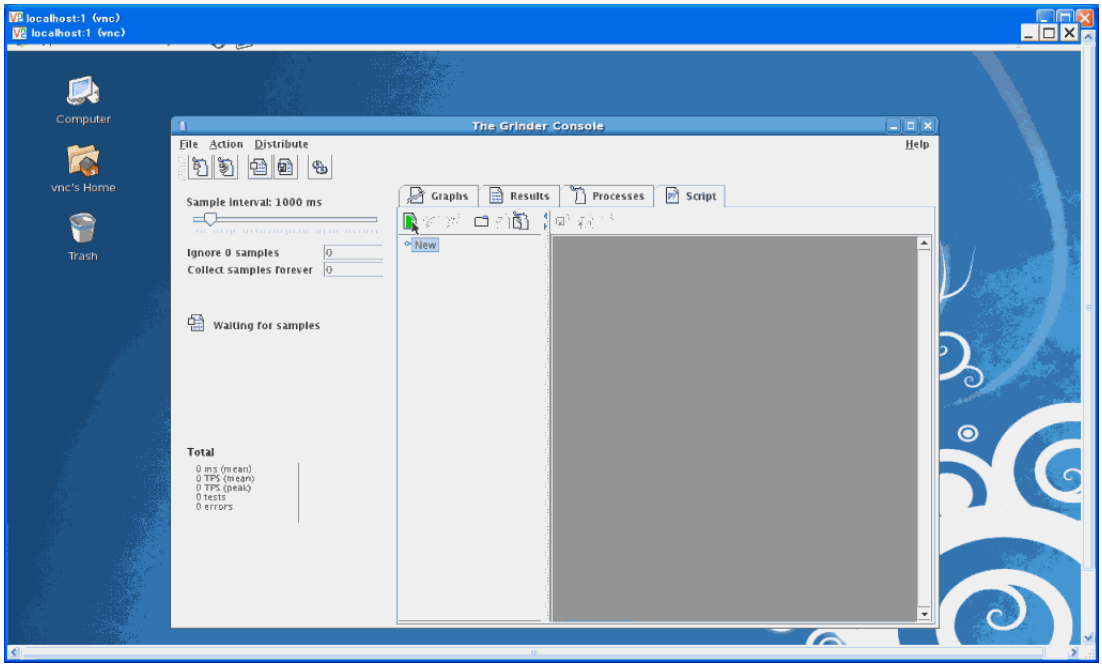

図 15:GrinderAgent の新規設定ファイルの作成

```
grinder.processes=1

grinder.threads=200

grinder.runs=0

grinder.logDirectory=log
```

「grinder-3.4」フォルダの下に「grinder.properties」という名前でファイルの種類を「All Files」にして保存します。(図 16:GrinderAgent の設定ファイルの保存)

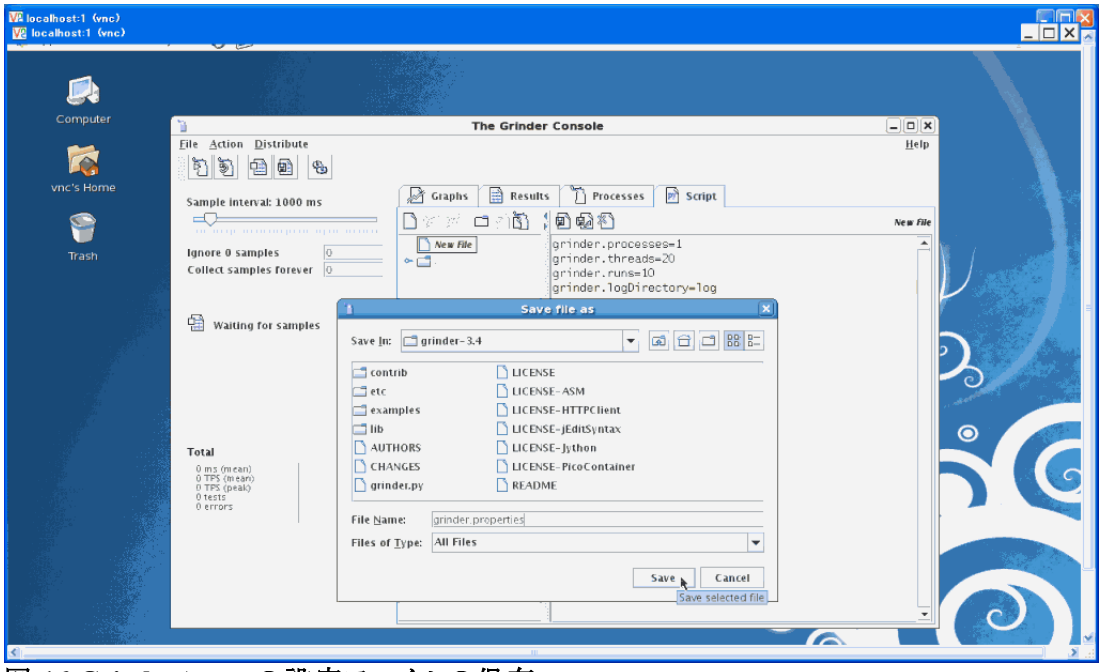

図 16:GrinderAgent の設定ファイルの保存

フォルダアイコンを開き、grinder-3.4フォルダの中に入っている先ほど作成した「grinder.properties」をダブルクリックするとファイルのアイコンに黄色の星マークがつきます。これによって実際に使用される properties ファイルが選択されます。(図 17:GrinderAgent に配布する設定ファイルの設定)

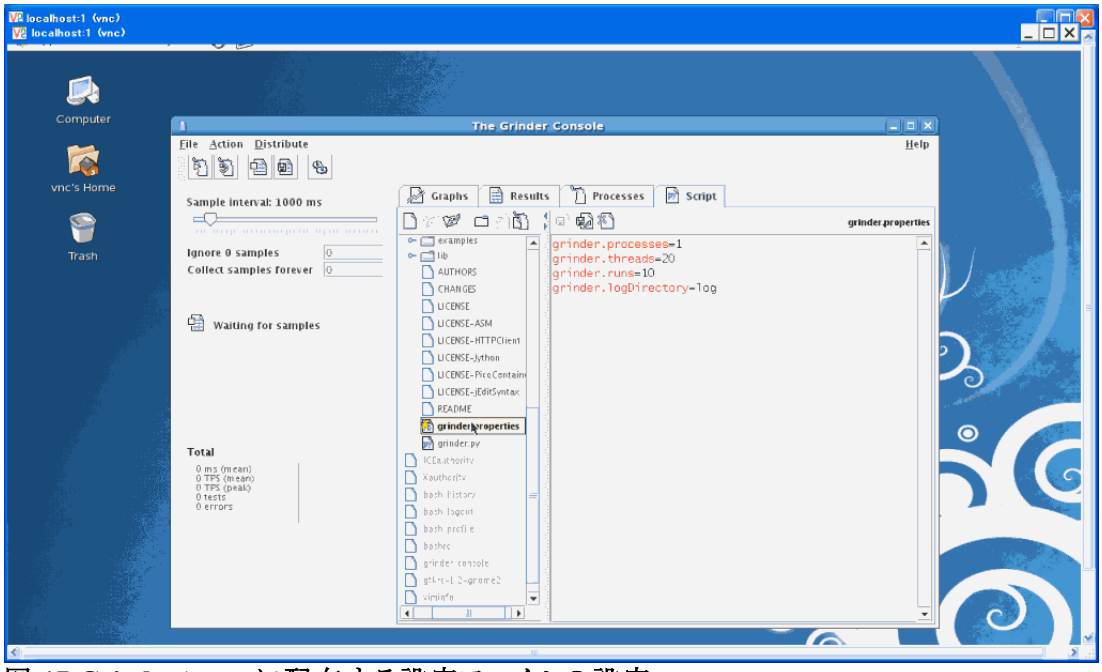

図 17:GrinderAgent に配布する設定ファイルの設定

1.1.2.5.Read/Write 負荷をかける

まずは先ほど GrinderConsole に転送してきた grinder.py(Write 用シナリオ)を使って負荷をかけてみましょう。
Agent が Console に接続されると Console 画面の左上の再生ボタンが押せるようになるので、押します。
「Start processes」ダイアログが表示されるので、「OK」ボタンをクリックします。(図 18:シナリオの実行)

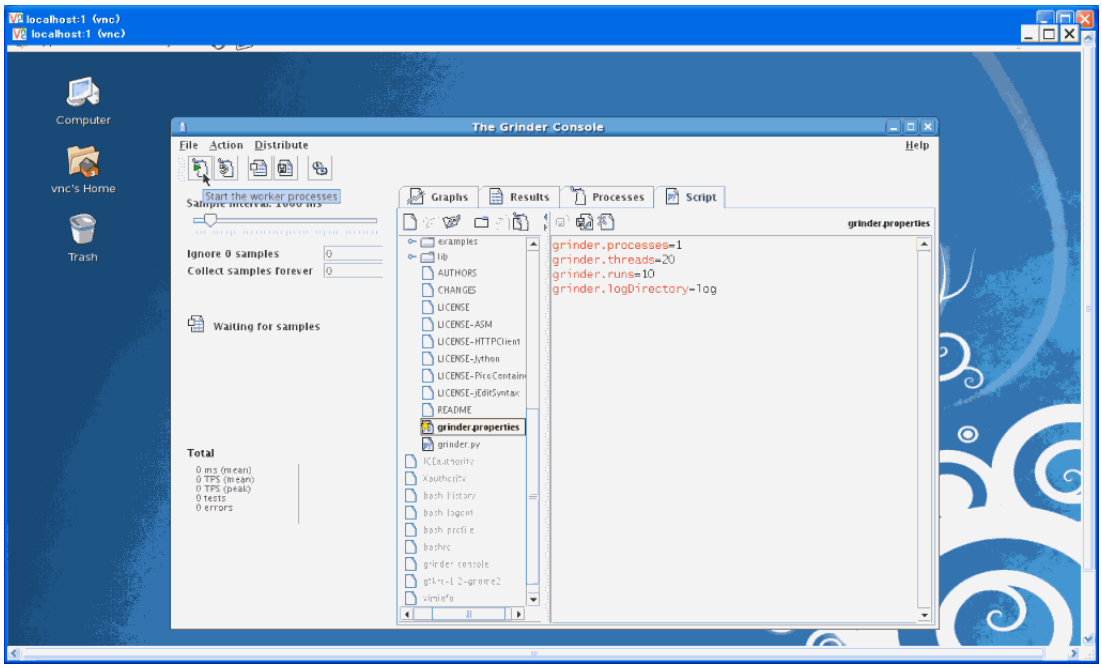

図 18:シナリオの実行

Agent がシナリオの実行を始めます。

「Graphs」タブをクリックすると監視内容が見れるようになります。(図 19:シナリオの実行情報)

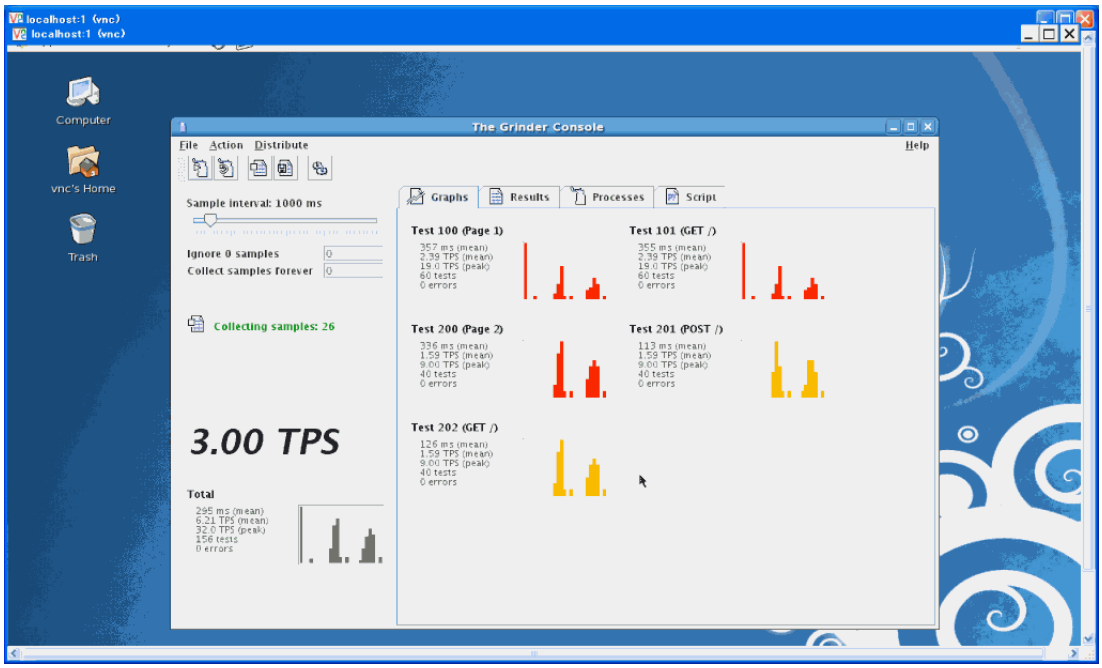

図 19:シナリオの実行情報

collecting sample の数値が100になったシナリオを終了します。
「Action→Stop processes」を実行すると各 Agent に停止の指示を送りシナリオが終了します。

実行結果は data_xxxxxxxxx.log というファイルで Agent が起動している Server に書き出されます。

仮想マシン一覧の misc フォルダの中にある「GrinderAgent」を右クリックし、「シェルの起動」を選択します。

「log」ディレクトリができているので、以下のコマンドで中を確認します。

```
-bash-3.2# ls -la log/
total 184
drwxr-xr-x 2 root root   4096 Jan 25 02:43 .
drwxr-x--- 6 root root   4096 Jan 25 02:43 ..
-rw-r--r-- 1 root root  54120 Jan 25 02:45 data_localhost-0.log
-rw-r--r-- 1 root root 112231 Jan 25 02:45 out_localhost-0.log
-bash-3.2#
```

同じように WinSCP で Read 用シナリオを GrinderConsole の grinder-3.4フォルダのなかに転送して Read 負荷もかけてみましょう。

1.1.2.6.グラフを作成する

先ほど実行結果の data_xxxxxxxxx.log をグラフにしてみましょう。

Windows の「スタート→すべてのプログラム→WinSCP→WinSCP」を起動し、「ホスト名」に「GrinderAgent のパブリック IP」、「ユーザ名」に「root」を入力して、ログインします。(図 20:GrinderAgent へログイン)

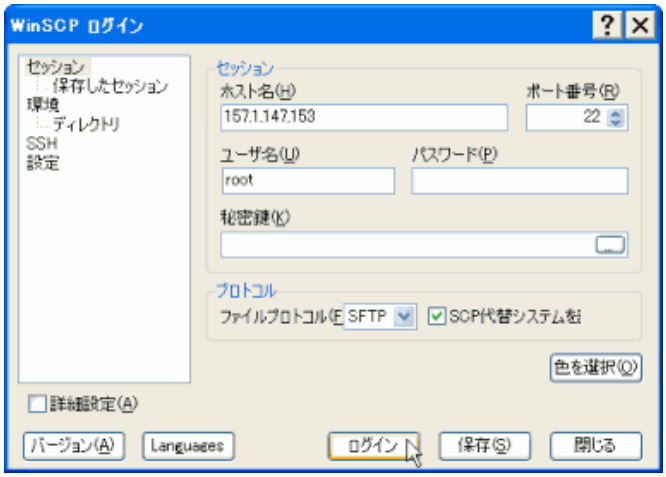

図 20:GrinderAgent へログイン

「U:\usr\.ec2\[プロジェクト名]\[ミニクラウドのホスト名]」の中に入っている「ログインユーザーID.pem」を GrinderAgent の「/root/」フォルダ以下に転送します。

Windows の「スタート→すべてのプログラム→クラウドクライアント→CloudClient」を起動し、仮想マシン一覧の misc フォルダの中にある先ほど鍵を転送した「GrinderAgent」を右クリックし、「シェルの起動」を実行します。

以下のコマンドを実行し、log フォルダを GrinderConsole の grinder analyzer フォルダへ転送します。

```
-bash-3.2# chmod 600 [ログインユーザーID].pem
-bash-3.2# scp -i ./[ログインユーザーID].pem -r log root@[GrinderConsole の host 名]:/home/vnc/
```

次に VNC ビューワを使って GrinderConsole へログインします。
Windows の「スタート→すべてのプログラム→RealVNC→VNC ビューワ4→VNC ビューワの起動」を実行する。

「サーバー名」に「GrinderConsole のパブリック IP:1」を入力し「OK」ボタンをクリックする。(図 21:VNC の接続先サーバーの設定)

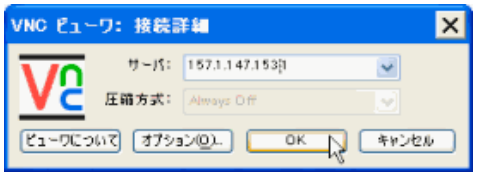

図 21:VNC の接続先サーバーの設定

「パスワード」に「vncvnc」を入力し、ログインする。(図 22:VNC のパスワード入力)

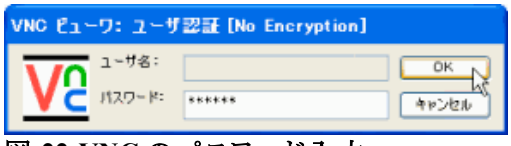

図 22:VNC のパスワード入力

「Applications」メニューの「Accessories」、「Terminal」を選択します。(図 23:Terminal の起動)

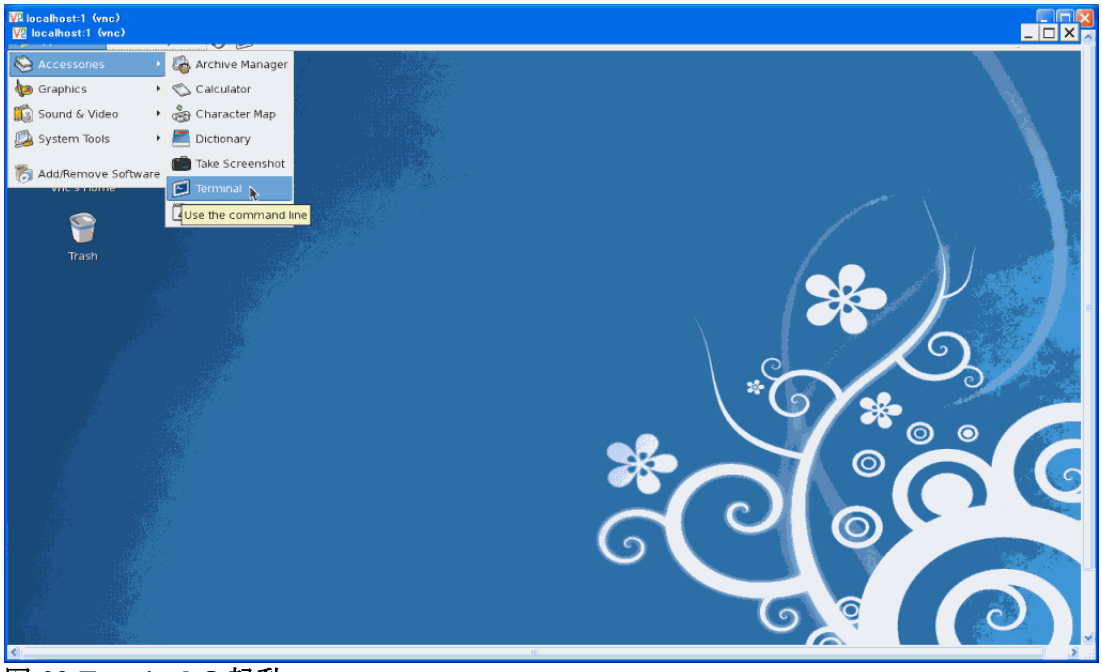

図 23:Terminal の起動

次に grinder analyzer のフォルダに移動し以下のコマンドを入力して、グラフを作成します。

```
$ jython ./analyzer.py /home/vnc/log/data_xxxxxxxxx.log  /home/vnc/log/out_xxxxxxxxx.log
```

「grinderReport」というフォルダが作成され中にグラフができていますので、確認してみてください。

後ほど、後の作業で Terracotta を起動してからもう一度負荷をかけてみましょう。

2章 Terracotta

2.1. Terracotta について

複数の javaVM 上で同じ java オブジェクトをキャッシュすることができるクラスタリングソリューションです。

また、TerracottaServer のクラスタリングや、hiavailability（HA）の設定などもできます。

詳しくは以下の URL を参照してください。

URL: http://www.terracotta.org/

用意するもの

- ☐ sample-kejiban.zip(ファイル)
- ☐ terracotta-license.key(ファイル)
- ☐ terracotta-toolkit-1.1-runtime-ee-2.0.0.jar(ファイル)
- ☐ ehcache-core-ee-2.3.1.jar(ファイル)
- ☐ ehcache-terracotta-ee-2.3.1.jar(ファイル)
- ☐ ecloud-terracotta.zip(ファイル)
- ☐ config.properties(ファイル)

- sample-kejiban は解凍して「U:¥usr¥workspace」以下に置いてください。
- terracotta-license.key は「U:¥usr¥デスクトップ」に置いてください。
- Terracotta-toolkit-1.1-runtime-ee-2.0.0.jar は「U:usr¥デスクトップ」に置いてください。
- ehcache-core-ee-2.3.1.jar「は U:usr¥デスクトップ」に置いてください。
- ehcache-terracotta-ee-2.3.1.jar は「U:usr¥デスクトップ」に置いてください。
- ecloud-terracotta.zip は解凍して「U:¥usr¥workspace」以下に置いてください。
- config.properties は解凍して自分の「プロジェクト名」フォルダのなかにあるものを「U:¥usr¥workspace¥ecloud-terracotta¥src」以下に上書きして保存してください。

アーカイブからダウンロードしておいてください。

2.2. サンプルプログラムを Terracotta に対応する

今回は、前回使用した、sample-kejiban を terracotta 対応し、以下の構成を作って動作の確認を行います。(図 24:Terracotta 対応全体構成図)

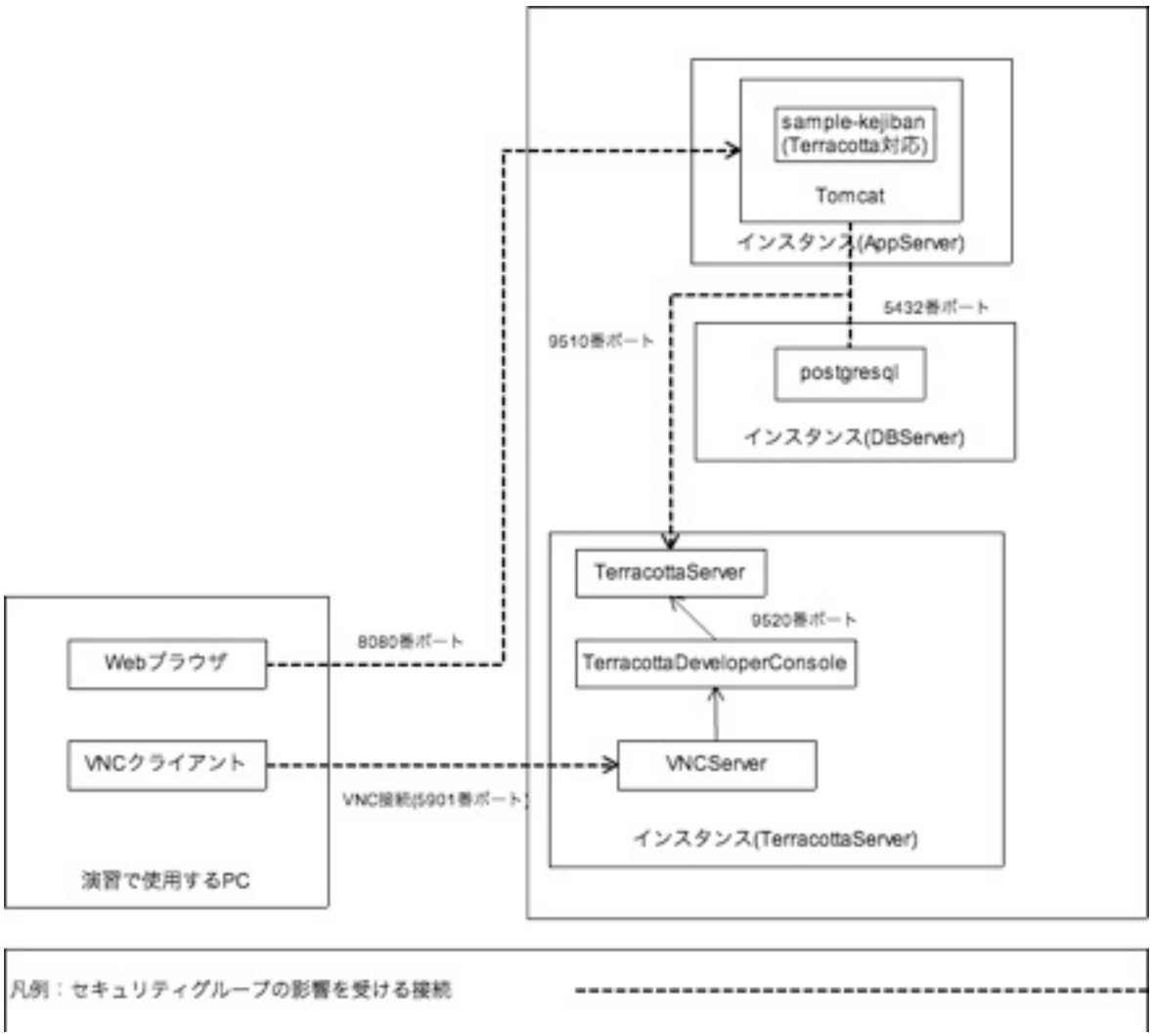

図 24:Terracotta 対応全体構成図

2.2.1. セキュリティグループを設定する

AppServer に接続するためにセキュリティグループの設定を行います。
Windows の「スタート→すべてのプログラム→クラウドクライアント→CloudClient」を起動し、「セキュリティグループ」タブをクリックします。セキュリティグループの名前一覧から「ログインユーザーID」をクリックし、右側の一覧を右クリックして「パーミッションの追加」を選択します。(図 25:パーミッションの追加)

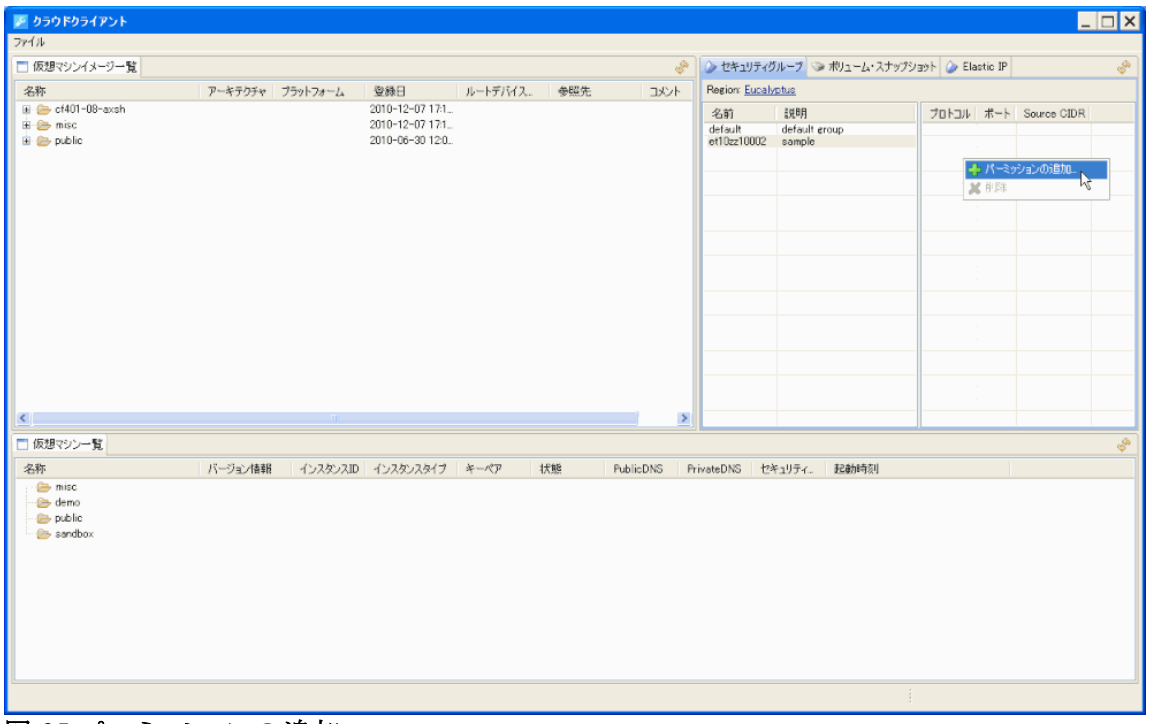

図 25:パーミッションの追加

Protocol に「TCP」、Port or Port Range に「8080」、Network Mask に「0.0.0.0/0」を入力し、「OK」ボタンをクリックします。

2.2.2. インスタンスを起動する

Windows の「スタート→すべてのプログラム→クラウドクライアント→CloudClient」を起動し、仮想マシンイメージ一覧の misc フォルダの中から TerracottaServer、DBServer、AppServer を右クリックし「仮想マシン起動」を実行します。以下の条件で起動してください。

- ☐ バージョン -
- ☐ インスタンス m1.small
- ☐ キーペア ログインユーザーID
- ☐ インスタンス数 1
- ☐ セキュリティグループ ログインユーザーID

なお起動には数分かかります。

2.2.3. sample-kejiban をインポートする

terracotta 対応する sample-kejiban を eclispe にインポートします。
デスクトップの eclipse アイコン[eclipse3.4.2(JRE1.6)]から「eclipse」を起動し、「ファイル→インポート」をクリックします。(図 26:プロジェクトのインポート)

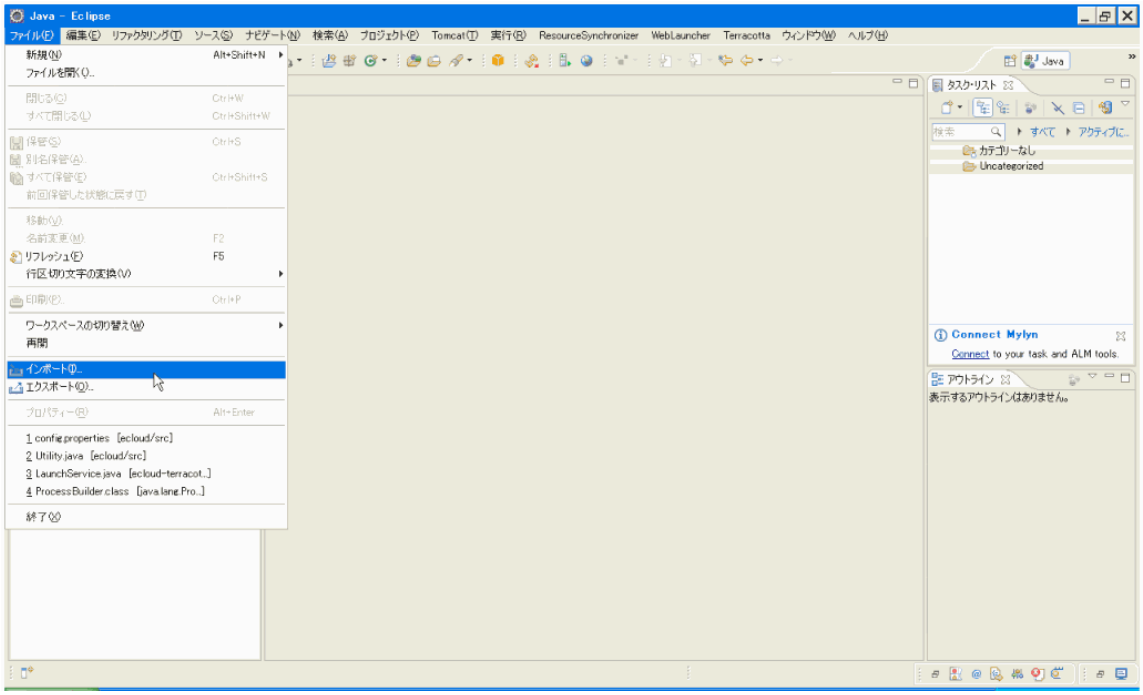

図 26:プロジェクトのインポート

「一般→既存プロジェクトをワークスペースへ」を選択し、「次へ」ボタンをクリックします。(図 27:既存プロジェクトをワークスペースへ)

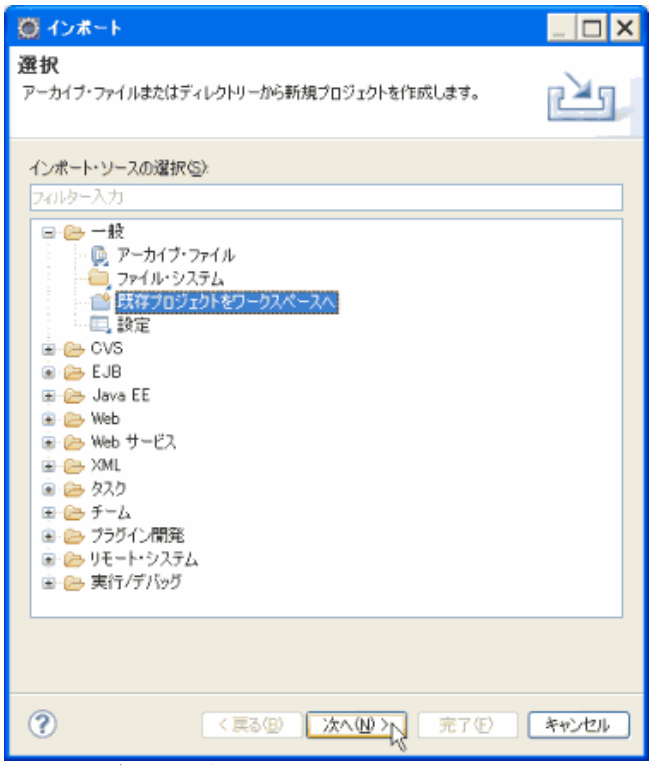

図 27:既存プロジェクトをワークスペースへ

「ルート・ディレクトリの選択」を選択し、「参照」ボタンをクリックします。(図 28:参照先の選択)

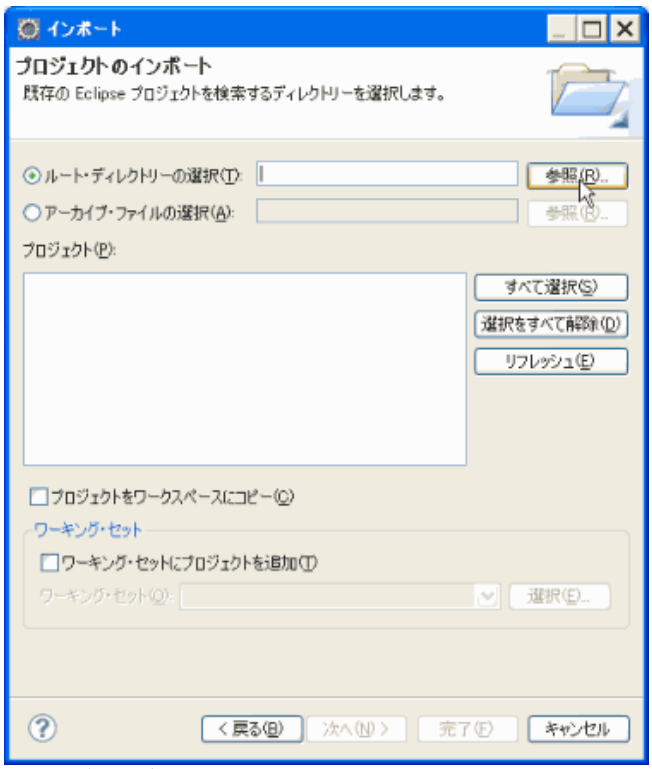

図 28:参照先の選択

「フォルダの参照」で「U:¥usr¥workspace」フォルダの中にある「sample-kejiban」を選択して「OK」ボタンをクリックします。(図 29:フォルダの選択)

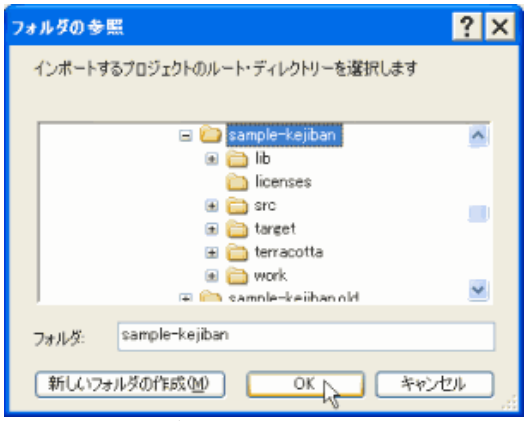

図 29:フォルダの選択

「終了」ボタンをクリックする(図 30:参照の完了)とパッケージ・エクスプローラーに「sample-kejiban」が表示されます。(図 31:プロジェクトのインポートの完了)

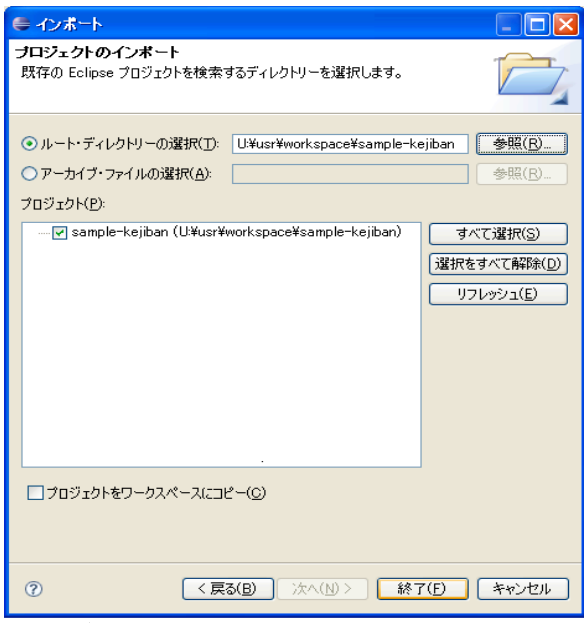

図 30:参照の完了

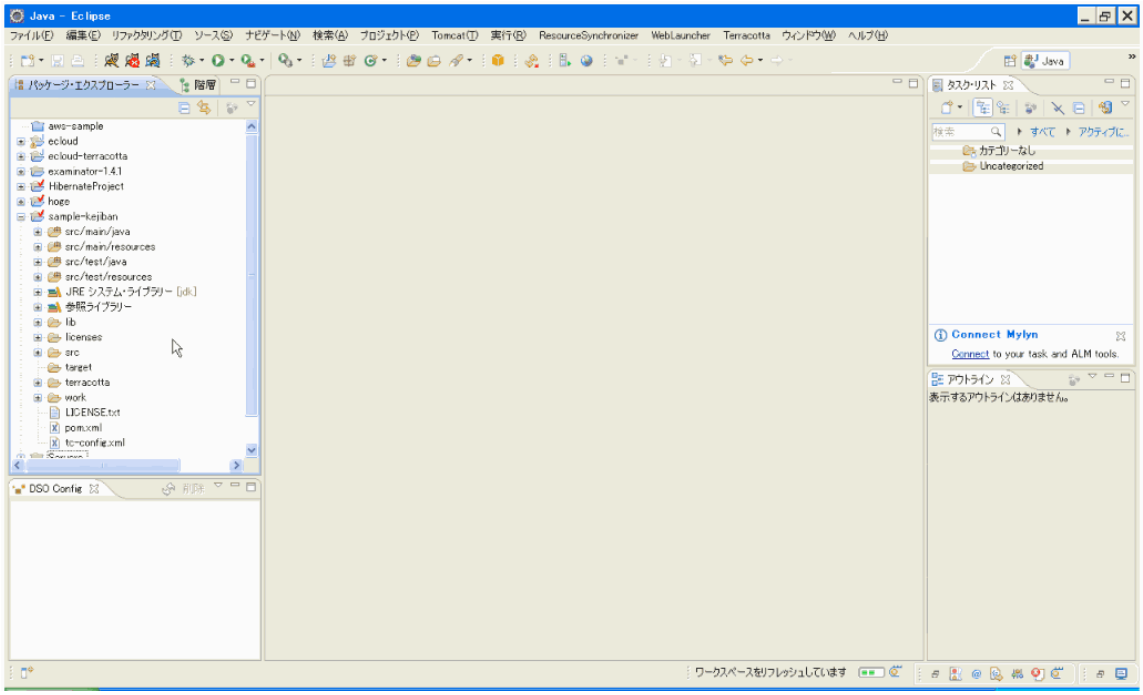

図 31:プロジェクトのインポートの完了

2.2.4. Jar ファイルを追加する

Terracotta 対応に必要な jar を追加します。
先ほどダウンロードした jar を「U:¥usr¥workspace¥sample-kejiban¥src¥main¥webapp¥WEB-INI¥lib」フォルダの中に以下の jar を追加してください。

- ☐ terracotta-toolkit-1.1-runtime-ee-2.0.0.jar
- ☐ ehcache-core-ee-2.3.1.jar
- ☐ ehcache-terracotta-ee-2.3.1.jar

次に追加した jar の「ビルド・パス」を通します。

パッケージ・エクスプローラーに表示されている「sample-kejiban」を右クリックし「ビルド・パス→ビルド・パスの構成」をクリックします。(図 32:ビルドパスの編集)

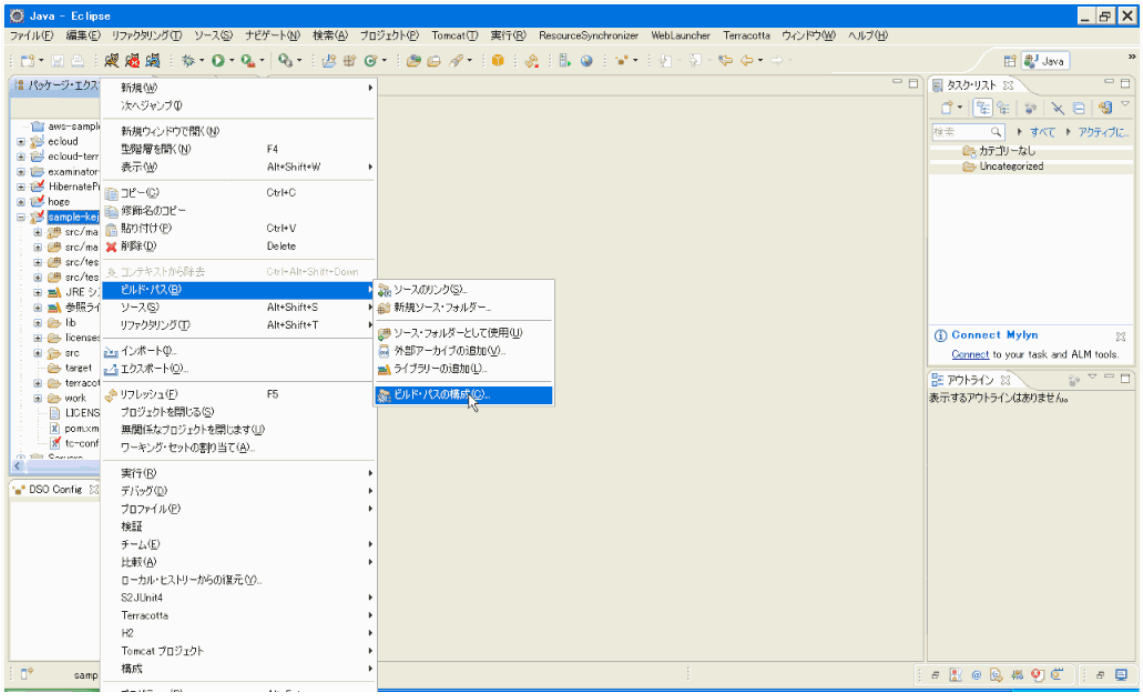

図 32:ビルドパスの編集

「ライブラリー」タブの「 Jar 追加」ボタンをクリックします。(図 33:Jar ファイルの追加)

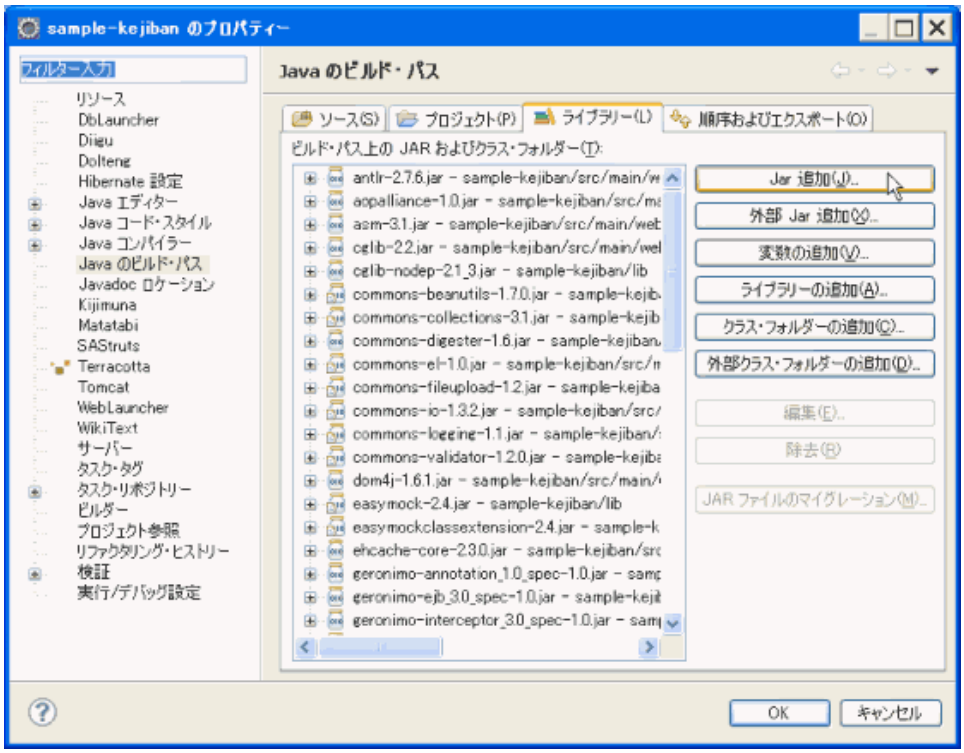

図 33:Jar ファイルの追加

「sample-kejiban¥src¥main¥webapp¥WEB-INI¥lib¥」にはいっている先ほど追加した jar を追加します。(図 34:Jar の選択)

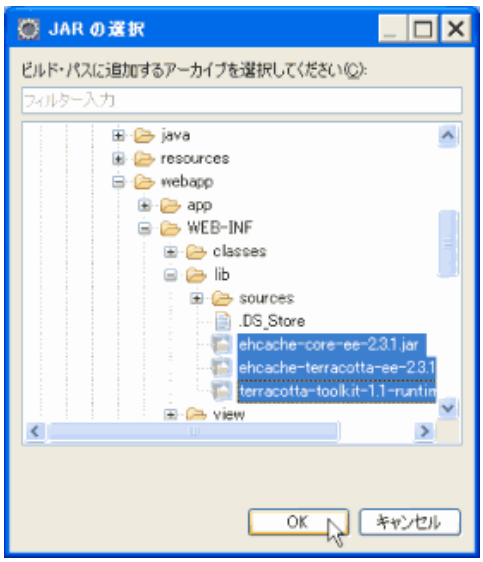

図 34:Jar の選択

先ほど追加した jar が表示されない場合は、「sampke-kejiban」を右クリックして、「リフレッシュ」ボタンををクリックしてからためしてください。

また元々使っていた「ehcache-core-2.3.0.jar」は「Remove」ボタンをクリックして削除しておきます。(図 35:Jar の除去)

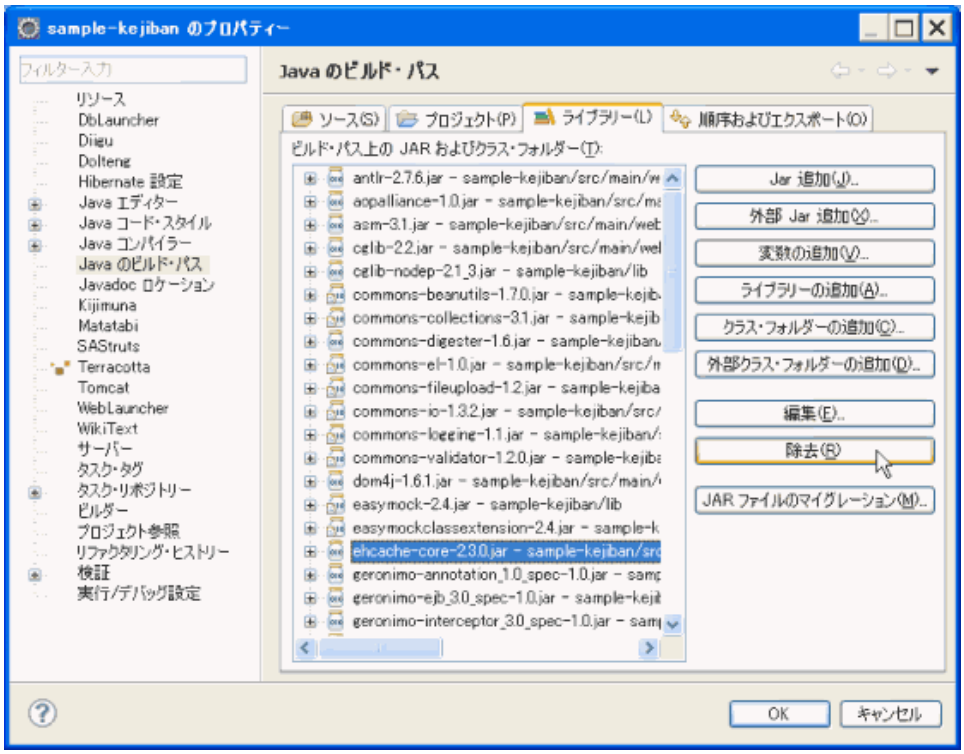

図 35:Jar の除去

2.2.5. サンプルプログラムを修正する

アプリケーションを TerracottaServer に接続するためにパッケージ・エクスプローラーに入っている「sample-kejiban→src/main/java→ehcache.xml」を編集します。

以下のように使用する cache に<terracotta/>タグを記述し、<terracottaConfig/>タグに接続先の TerracottaServer の url を記述します。

```
<?xml version="1.0" encoding="UTF-8"?>
<ehcache xsi:noNamespaceSchemaLocation="ehcache.xsd" name="sample-kejiban">
        <defaultCache maxElementsInMemory="10000"timeToIdleSeconds="3600" timeToLiveSeconds="86400">
                <terracotta/>
        </defaultCache>

        <cache name="kejiban.entity.Res" maxElementsInMemory="10000" eternal="false"  timeToIdleSeconds="0"
timeToLiveSeconds="120">
                <terracotta/>
        </cache>

        <terracottaConfig url="TerracottaServer のプライベート IP:9510"/>
</ehcache>
```

次にパッケージ・エクスプローラーに入っている「sample-kejiban→src/main/java→hibernate.xml」を編集します。
property の「hibernate.connection.url」を下記のように変更します。

```
<property name=" hibernate.connection.url" >jdbc:postgresql://[DBServerのプライベート
IP]:5432/demo</property>
```

2.2.6. TerracottaServer を設定する

2.2.6.1.tc-config.xml を作成する

TerracottaServer の config ファイルを作成します。

Windows の「スタート→すべてのプログラム→クラウドクライアント→CloudClient」を起動し、「仮想マシン一覧」から TerracottaServer を右クリックし、「シェルの起動」を選択します。(図 36:シェルの起動)

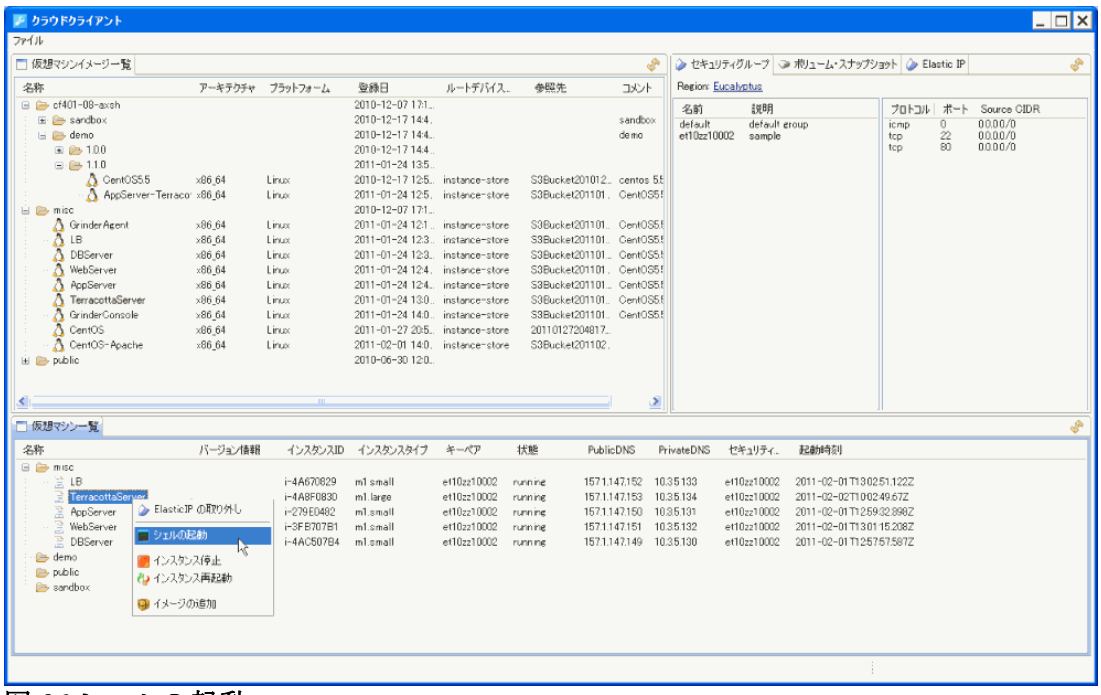

図 36:シェルの起動

テキストエディタで「tc-config.xml」ファイルを作成します。

```
-bash-3.2# vi /opt/terracotta/tc-config.xml
```

以下のように「host」、「name」、「group-name」、「member」に TerracottaServer の InstanceId を記述して作成してください。

```
<?xml version="1.0" encoding="UTF-8"?>

<con:tc-config xmlns:con="http://www.terracotta.org/config">

<tc-properties>
 <property name="productkey.path" value="/opt/terracotta/terracotta-license.key" />
</tc-properties>

<servers>
 <server host="TerracottaServer の instanceId" name="TerracottaServer の InstanceId">
 <dso-port bind="0.0.0.0">9510</dso-port>
 <jmx-port bind="0.0.0.0">9520</jmx-port>
 <data>terracotta/server-data</data>
 <logs>terracotta/server-logs</logs>
 <statistics>terracotta/cluster-statistics</statistics>
 <l2-group-port>9530</l2-group-port>
 </server>

 <mirror-groups>
 <mirror-group group-name="TerracottaServer の InstanceId">
 <members>
 <member>TerracottaServer の InstanceId</member>
 </members>
 </mirror-group>
 </mirror-groups>
</servers>
```

```
<clients>
 <logs>terracotta/client-logs</logs>
</clients>

</con:tc-config>
```

2.2.6.2.terracotta-license.key を配置する

Windows の「スタート→すべてのプログラム→WinSCP→WinSCP」を起動し、「ホスト名」に「TerracottaServer のパブリック IP」、「ユーザー名」に「root」を記述し、「ログイン」ボタンをクリックして TerracottaServer にログインします。(図 37:TeracottaServer へログイン)

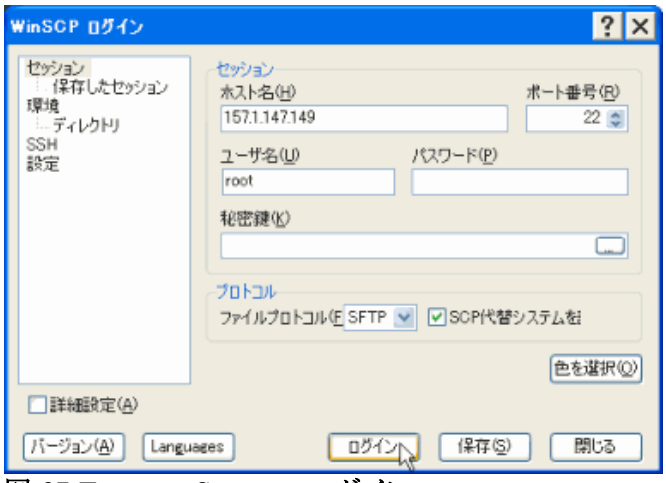

図 37:TeracottaServer へログイン

「/opt/terracotta」フォルダの中に先ほどダウンロードしてきた「terracotta-license.key」を転送します。(図 38:teracotta-license.key の転送)

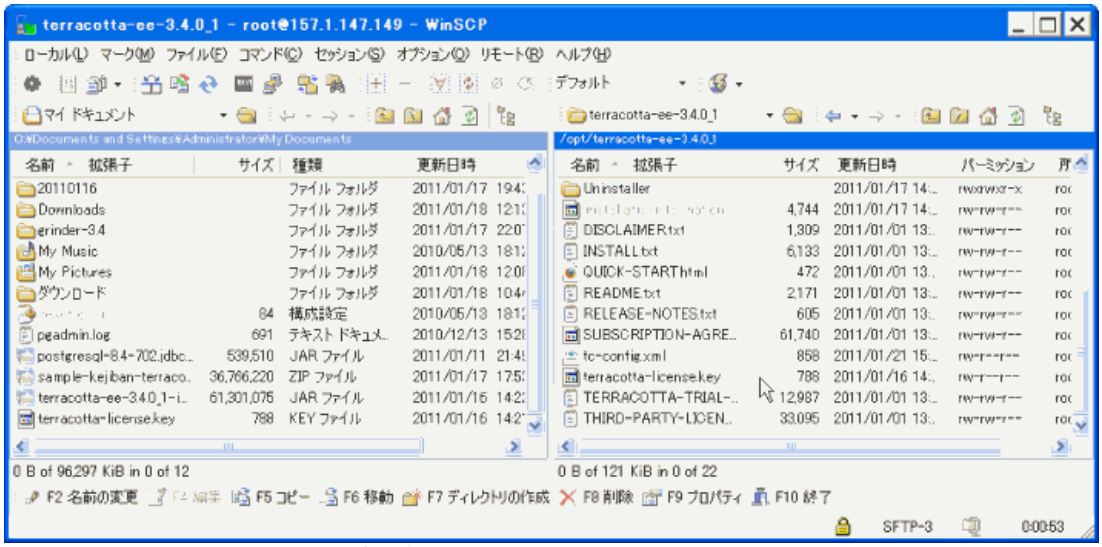

図 38:teracotta-license.key の転送

2.2.6.3.Hosts ファイルを修正する

TerracottaServer の「/etc/hosts」ファイルを編集します。

行末に一行追加してください。

```
-bash-3.2# vi /etc/hosts
127.0.0.1 localhost.localdomain localhost
127.0.0.1 TerracottaServer の InstanceId
```

2.2.6.4.TerracottaServer を起動する

以下のコマンドで TerracottaServer を起動します。

```
-bash-3.2# /etc/init.d/terracotta start TerracottaServer の InstanceId
Starting terracotta:


-bash-3.2#
```

第2引数には TerracottaServer のインスタンス Id
以下のコマンドで起動できているか確認してください。

```
-bash-3.2# tail -F /var/log/terracotta.log


Edition: FX
Expiration Date: 2011-04-15
License Number: TRIAL LICENSE KEY - AGREEMENT IS ACCEPTED
License Type: Trial
Licensee: TRIAL LICENSE USER
Max Client Count: 20
Product: Enterprise Suite
ehcache.maxOffHeap: 70G
terracotta.serverArray.maxOffHeap: 70G


2011-01-17 04:27:51,346 INFO - Available Max Runtime Memory: 490MB
2011-01-17 04:27:53,786 INFO - JMX Server started. Available at URL[service:jmx:jmxmp://0.0.0.0:9520]
2011-01-17 04:27:59,677 INFO - Becoming State[ ACTIVE-COORDINATOR ]
2011-01-17 04:27:59,716 INFO - Terracotta Server instance has started up as ACTIVE node on
0:0:0:0:0:0:0:0:9510 successfully, and is now ready for work.
```

2.2.6.5.TerracottaDeveloperConsole で確認する

Windows の「スタート→すべてのプログラム→RealVNC→VNC ビューワ4→VNC ビューワの起動」を実行する。

「サーバー名」に「TerracottaServer のパブリック IP:1」を入力し「OK」ボタンをクリックする。(図 39:VNC の接続先サーバーの設定)

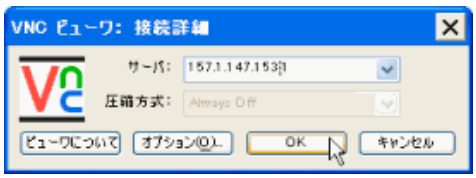

図 39:VNC の接続先サーバーの設定

「パスワード」に「vncvnc」を入力し、ログインする。(図 40:VNC のパスワード入力)

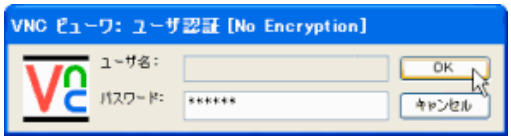

図 40:VNC のパスワード入力

スクリーンセーバが起動していて、パスワード入力を求められる場合は「vncvnc」を入力します。(図 41:スクリーンセーバ)

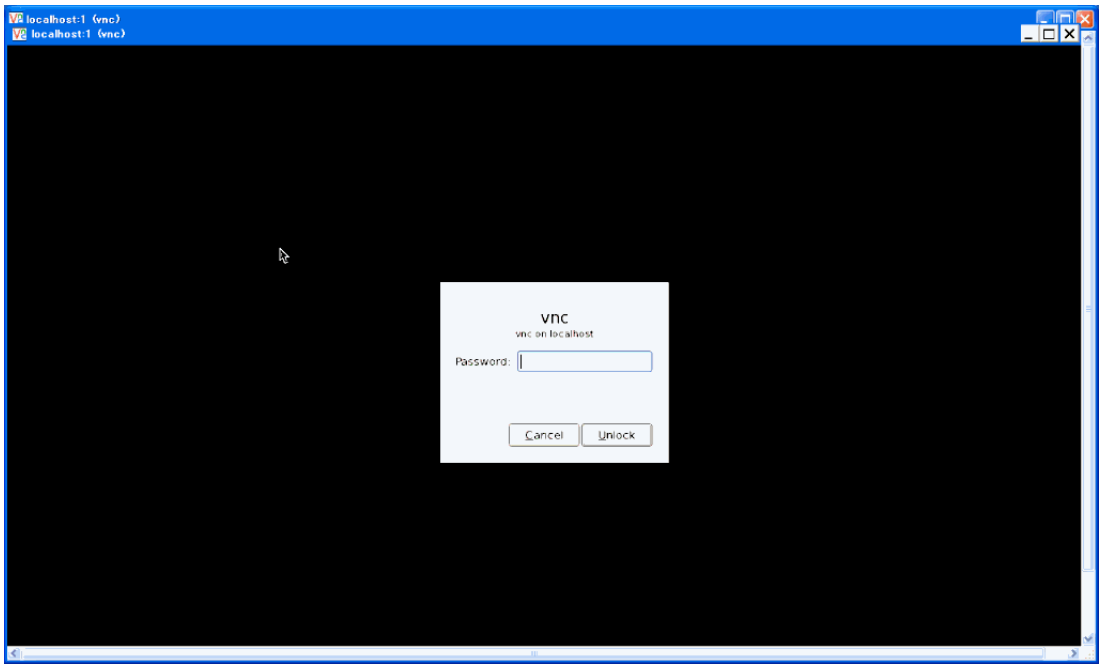

図 41:スクリーンセーバ

「Applications」メニューの「Accessories」、「Terminal」を選択します。(図 42:Terminal の起動)

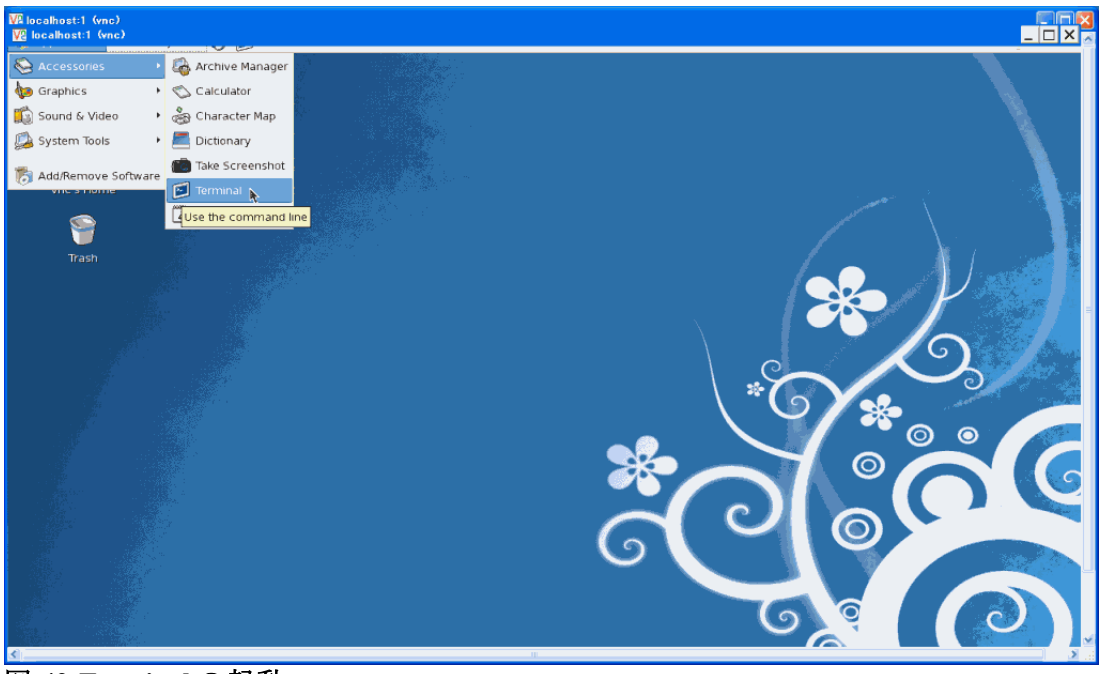

図 42:Terminal の起動

以下のコマンドで「Terracotta Developer Console」を起動します。

```
-bash-3.2# JAVA_HOME=/usr/java/default ./terracotta/bin/dev-console.sh
```

「Server host」に「TerracottaServer のパブリック IP」、「JMX port」に「9520」を入力して、「Connect」ボタンをクリックしてログインします。(図 43:DeveloperConsole のログイン)

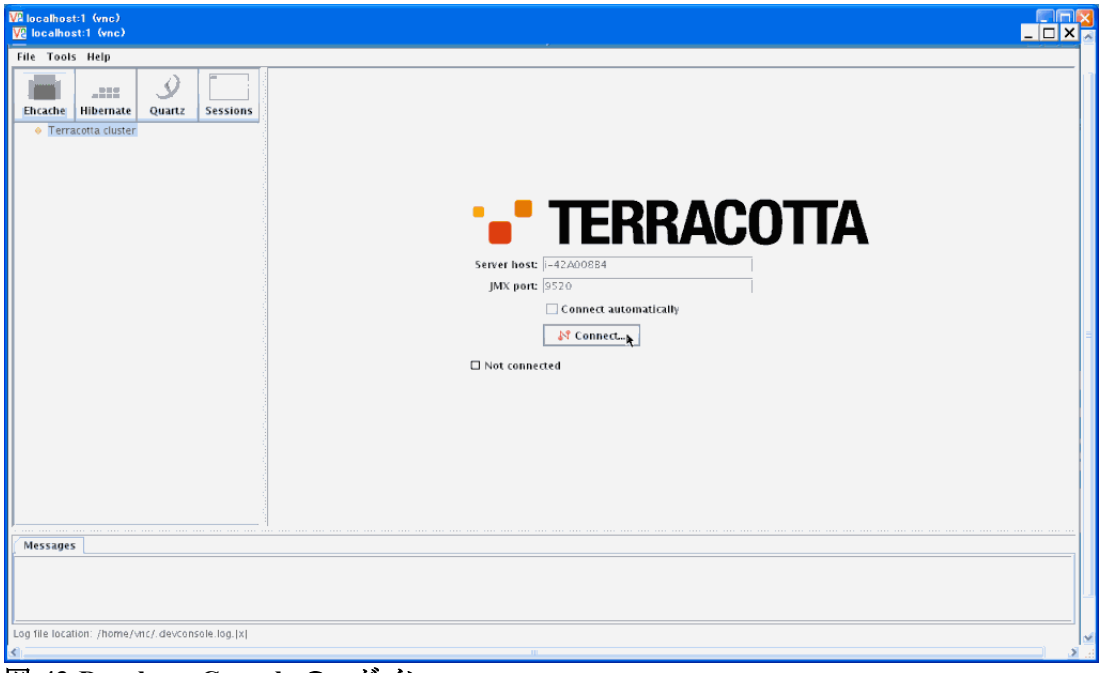

図 43:DeveloperConsole のログイン

「Topology→Server Array→Mirror group」の中に先ほど起動した TerracottaServer のインスタンス ID が表示されていることを確認してください。(図 44:TeracottaServer の接続確認)

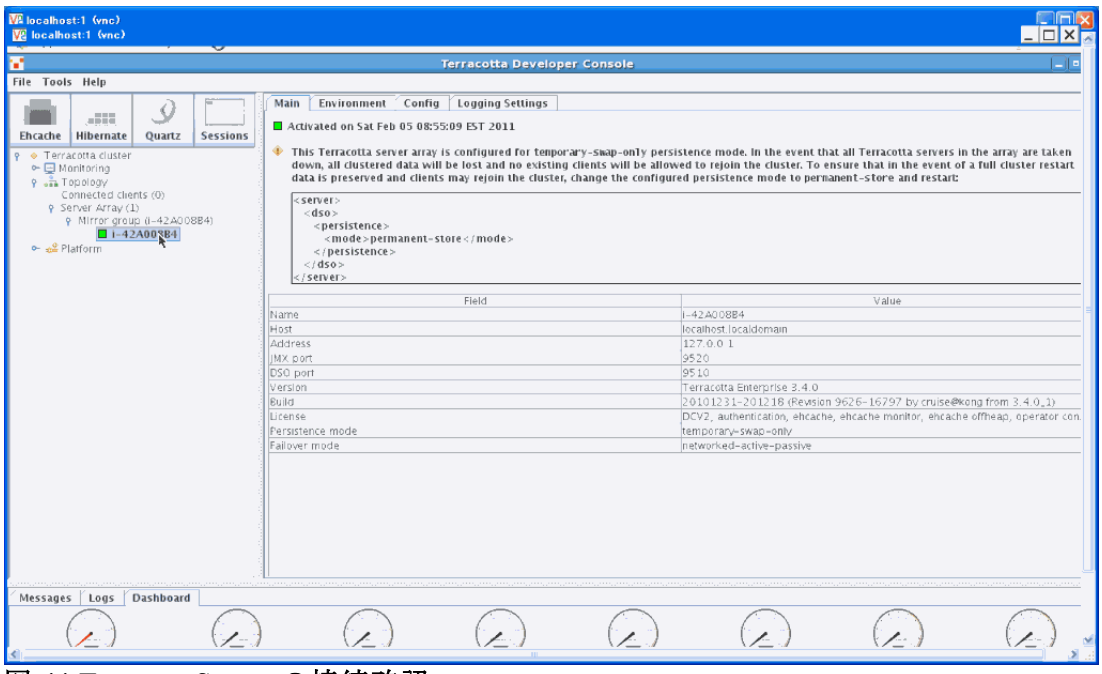

図 44:TeracottaServer の接続確認

2.2.7. AppServer を設定する

2.2.7.1.AppServer に eclispe プロジェクトを配置する

Windows の「スタート→すべてのプログラム→WinSCP→WinSCP」を起動し、「ホスト名」に「APPServer のパブリック IP」、「ユーザー名」に「root」を入力してログインします。(図 45:AppServer へログイン)

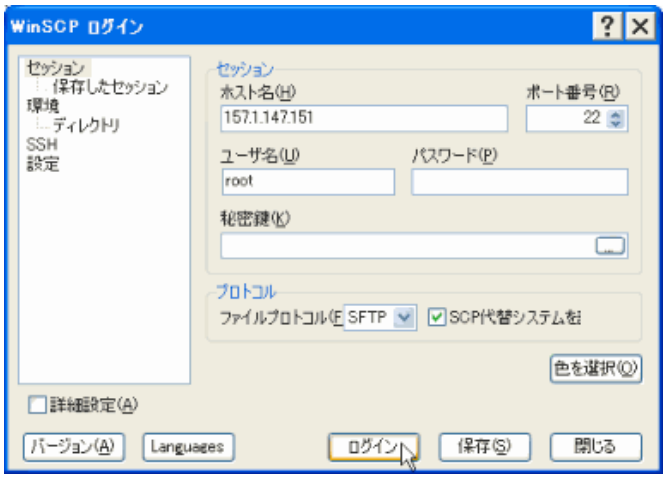

図 45:AppServer へログイン

「/opt」ディレクトリに先ほど編集した「sample-kejiban」と「terracotta-license.key」を転送します。(図 46:サンプルプログラムと teracotta-license.key の転送)
すでに server 側に sample-kejiban がある場合は、server 側の sample-kejiban を右クリックして、「名前の変更」を実行して rename してから転送してください。

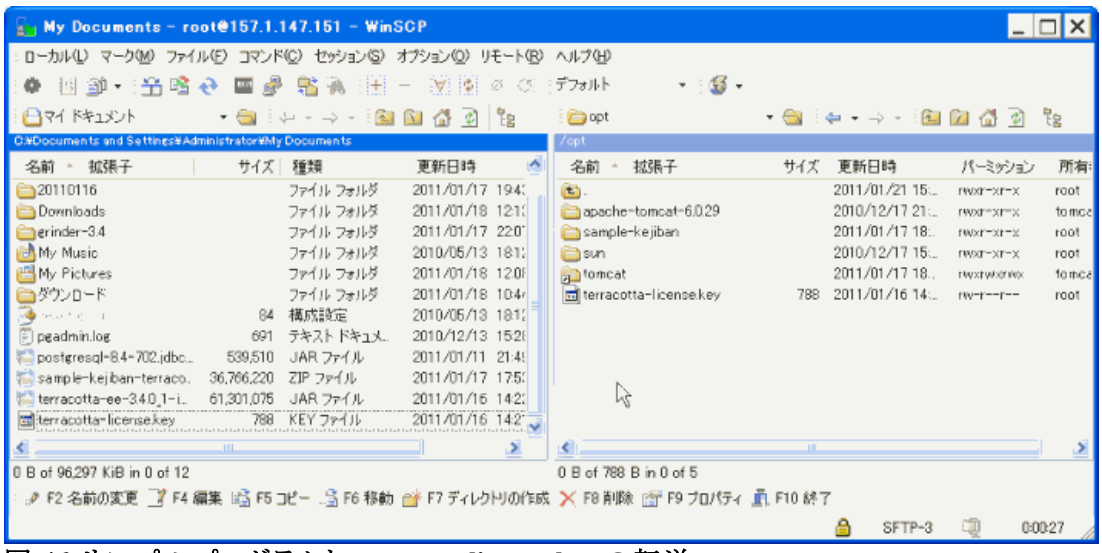

図 46:サンプルプログラムと teracotta-license.key の転送

Windows の「スタート→すべてのプログラム→クラウドクライアント→CloudClient」を起動し、仮想マシン一覧の misc フォルダの中にある APPServer を右クリックして、「シェルの起動」を選択します。(図 47:シェルの起動)

「/opt」ディレクトリに「sample-kejiban」と「terracotta-license.key」があることを確認します。

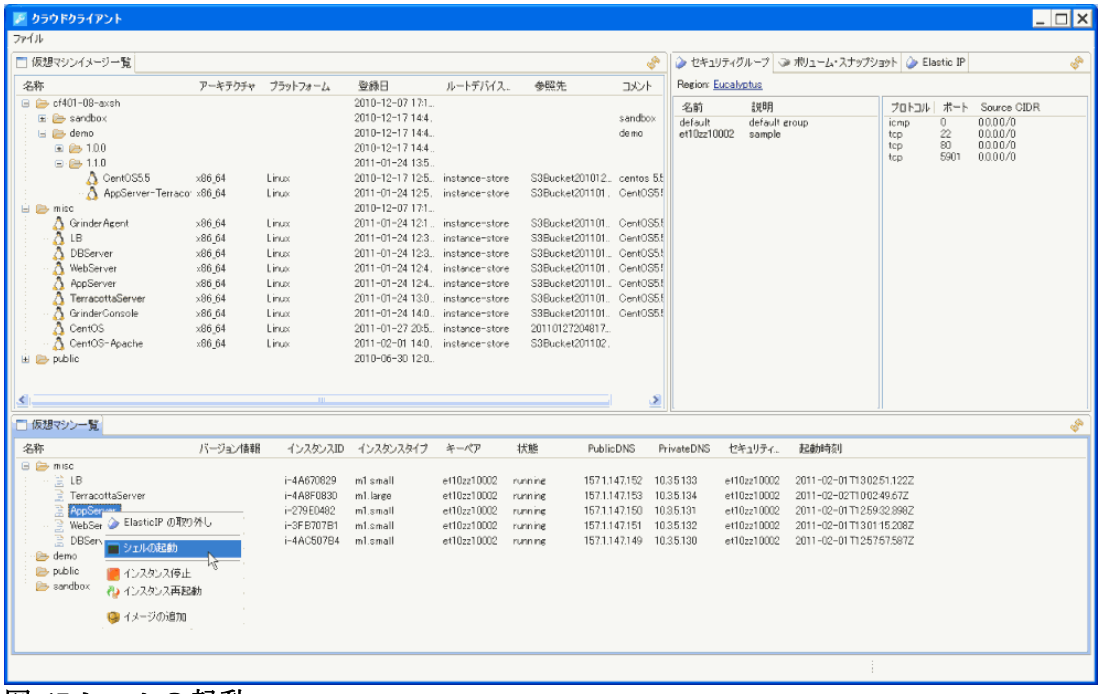

図 47:シェルの起動

「terracotta-license.key」を「sample-kejiban」の「src/main/webapp/WEB-INI/classes」フォルダの中に以下のコマンドでコピーします。

```
-bash-3.2# cp /opt/terracotta-license.key /opt/sample-kejiban/src/main/webapp/WEB-INI/classes/
```

次に「/opt/sample-kejiban/src/main/webapp」フォルダを「sample-kejiban」という名前で

「/opt/tomcat/webapps/」フォルダ以下に以下のコマンドでコピーします。

```
-bash-3.2# cp -r /opt/sample-kejiban/src/main/webapp /opt/tomcat/webapps/
-bash-3.2# mv /opt/tomcat/webapps/webapp /opt/tomcat/webapps/sample-kejiban
```

2.2.7.2.hosts ファイルを修正する

「/etc/hosts」ファイルをテキストエディタで編集します。
「TerracottaServer のプライベート IP」と「TerracottaServer の instanceId」を行末に1行追加します。

```
-bash-3.2# vi /etc/hosts
127.0.0.1 localhost.localdomain localhost
[TerracottaServer のプライベート IP] [TerracottaServer の instanceId]


例
10.3.5.130 i-58C209B4
```

2.2.7.3.Tomcat を起動する

以下のコマンドで TomcatServer を起動します。

```
-bash-3.2# /etc/init.d/tomcat start
Starting tomcat:
Using CATALINA_BASE:   /opt/tomcat
Using CATALINA_HOME:   /opt/tomcat
Using CATALINA_TMPDIR: /opt/tomcat/temp
Using JRE_HOME:        /usr/java/default
Using CLASSPATH:       /opt/tomcat/bin/bootstrap.jar
-bash-3.2#
```

2.2.8.　　動作確認する

2.2.8.1.Web ブラウザから確認する

「前回演習資料 1.2.4.1 Web ブラウザから確認する」を参考に確認してください。

2.2.8.2.データベースから確認する

「前回演習資料 1.2.4.2 データベースから確認する」を参考に確認してください。

2.2.8.3.TerracottaDeveloperConsole から確認する

Windows の「スタート→すべてのプログラム→RealVNC→VNC ビューワ4→VNC ビューワの起動」を実行する。

「サーバー名」に「TerracottaServer のパブリック IP:1」を入力し「OK」ボタンをクリックする。(図

48:VNC の接続先サーバーの設定)

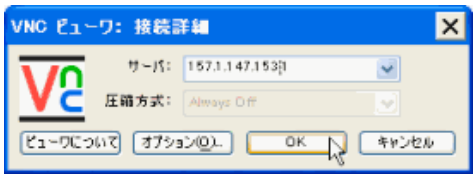

図 48:VNC の接続先サーバーの設定

「パスワード」に「vncvnc」を入力し、ログインする。(図 47:VNC のパスワード入力)

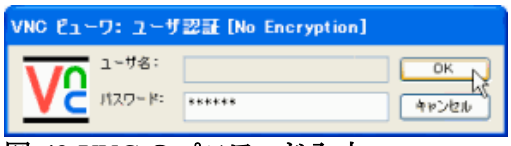

図 49:VNC のパスワード入力

スクリーンセーバが起動していて、パスワード入力を求められる場合は「vncvnc」を入力します。(図 50:スクリーンセーバ)

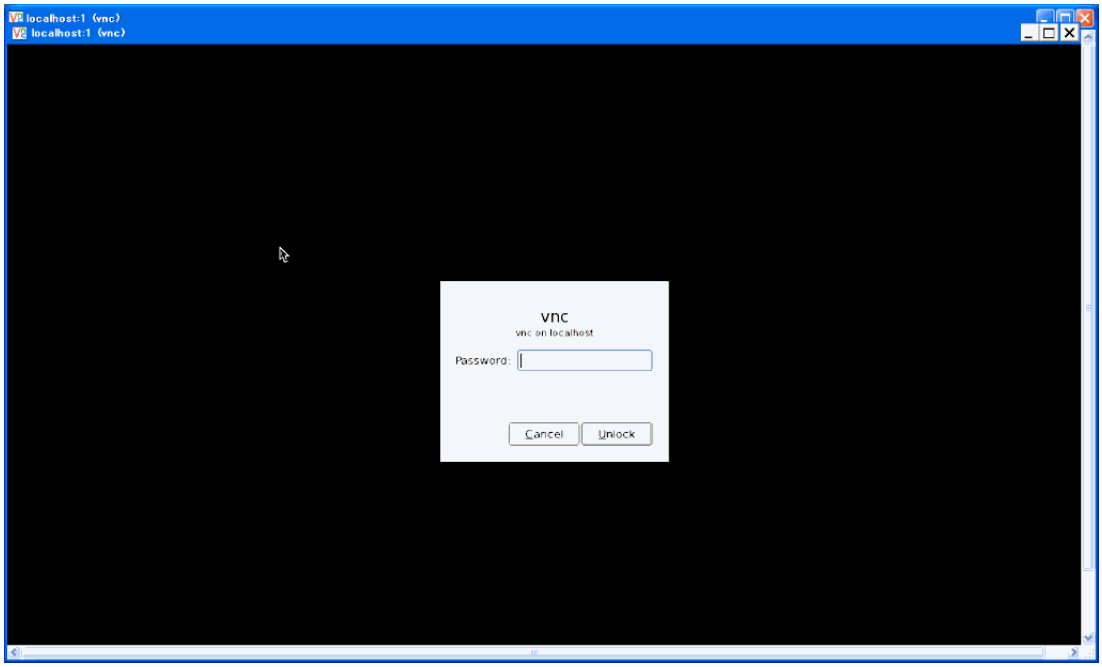

図 50:スクリーンセーバ

「Applications」メニューの「Accessories」、「Terminal」を選択します。(図 51:Terminal の起動)

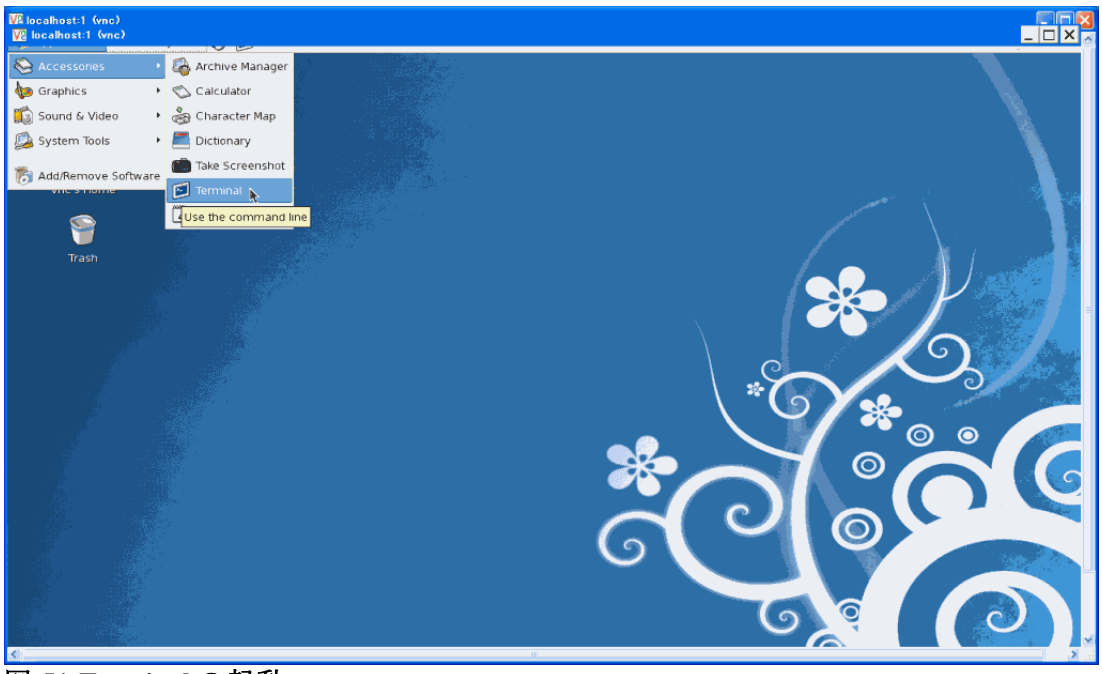

図 51:Terminal の起動

以下のコマンドで「Terracotta Developer Console」を起動します。

```
-bash-3.2# JAVA_HOME=/usr/java/default ./terracotta/bin/dev-console.sh
```

「Server host」に「TerracottaServer の instanceId」、「JMX port」に「9520」を入力して、「Connect」ボタンをクリックしてログインします。(図 52:DeveloperConsole のログイン)

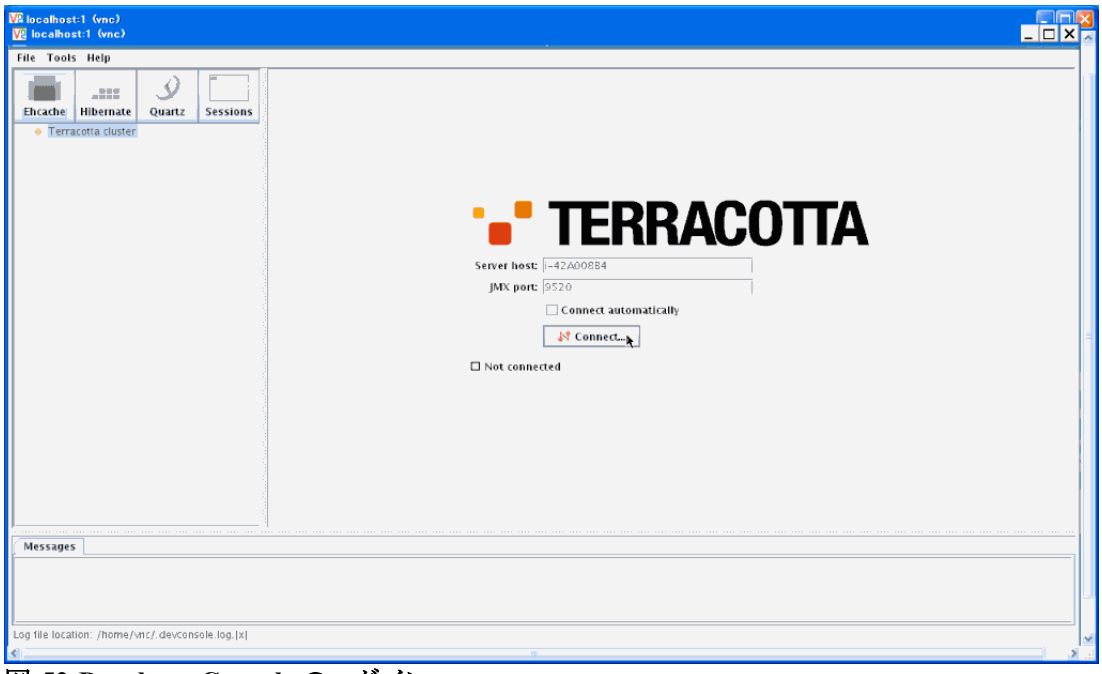

図 52:DeveloperConsole のログイン

「Topology→Connected clients」の中に「APPServer」の情報が表示されていることを確認します。(図 53:Client の接続確認)

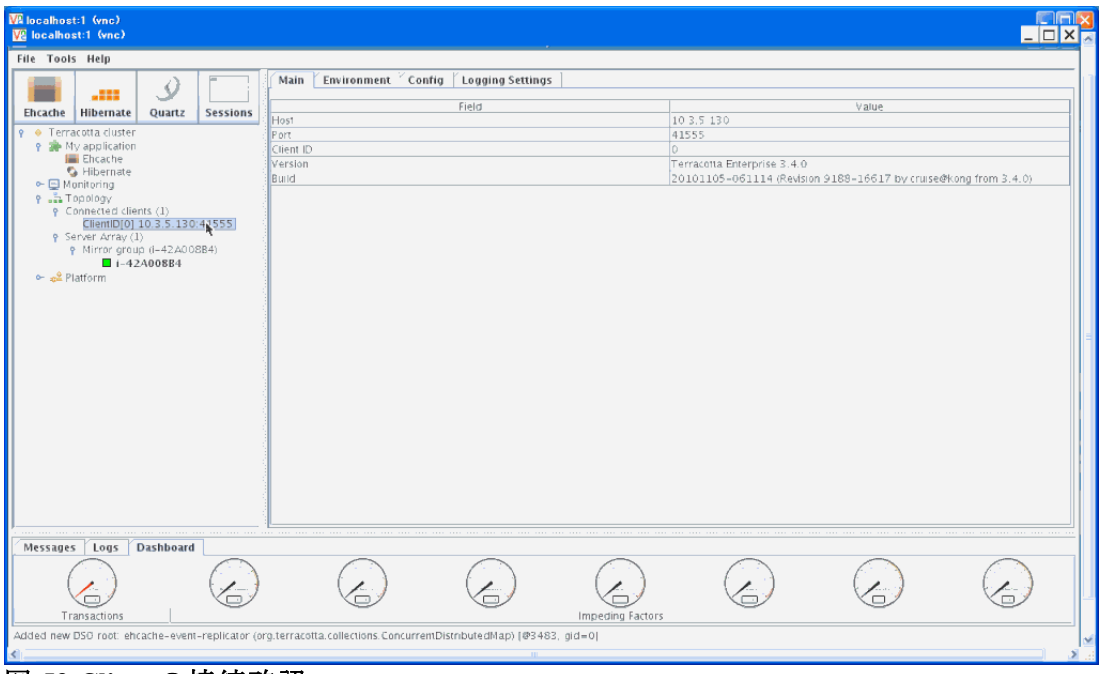

図 53:Client の接続確認

「My application」の「Ehcache」をクリックし、「performans」タブをクリックします。(図 54:Echache のパフォーマンス確認)

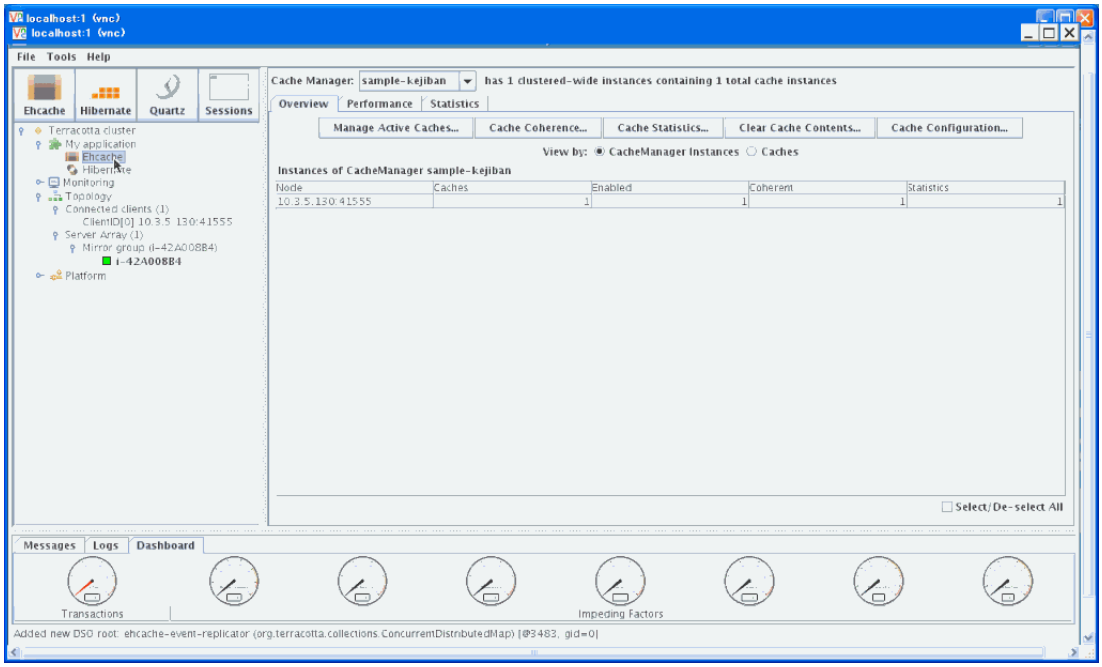

図 54:Echache のパフォーマンス確認

ダイアログボックスが表示される場合は「はい」ボタンをクリックしてください。(図 55:統計情報の有効化)

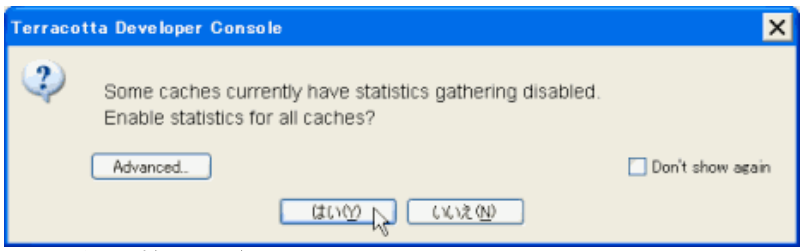

図 55:統計情報の有効化

Web ブラウザでリロードしたときに Cache が Hits されていることを確認してください。(図 56:Cache の確認)

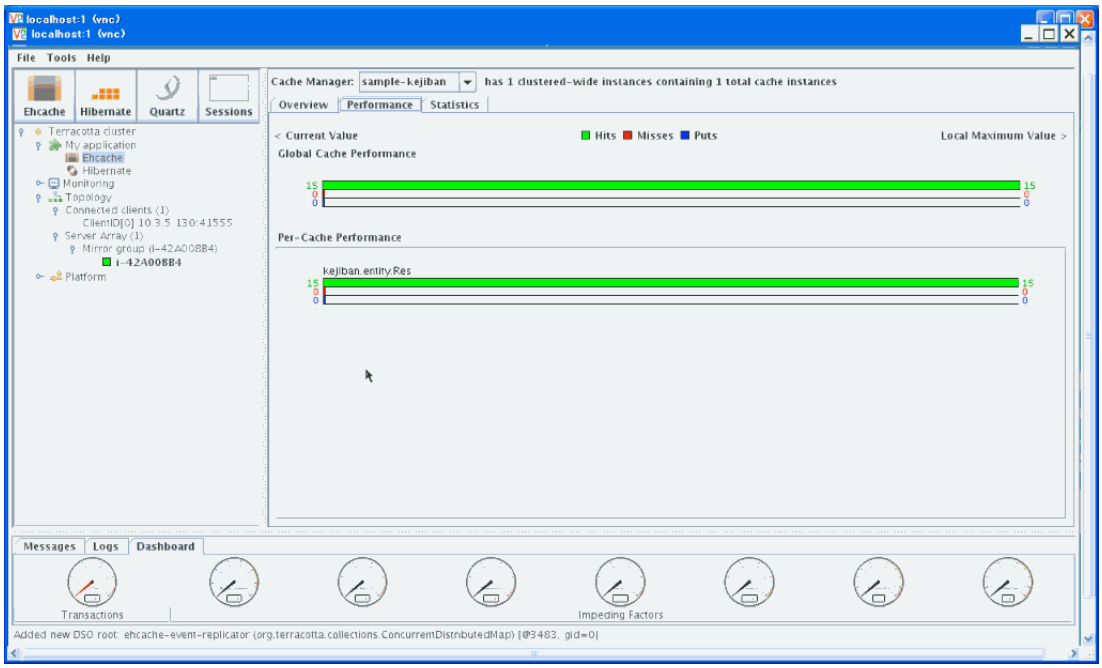

図 56:Cache の確認

2.2.9. 仮想マシンイメージを作成する

Terracotta 対応した APPServer のイメージを作成します。

Windows の「スタート→すべてのプログラム→クラウドクライアント→CloudClient」を起動し、仮想マシン一覧の misc フォルダの中にある APPServer を右クリックし、「イメージの追加」を選択します。

あらかじめ、「/etc/hosts」ファイルに追加した行と「terracotta-license.key」は削除しておいてください。

2.3. 起動スクリプトを実行する

以下の図(図 57:起動スクリプト実行時の全体の概要)が今回の起動スクリプトを実行した時の全体の概要になります。

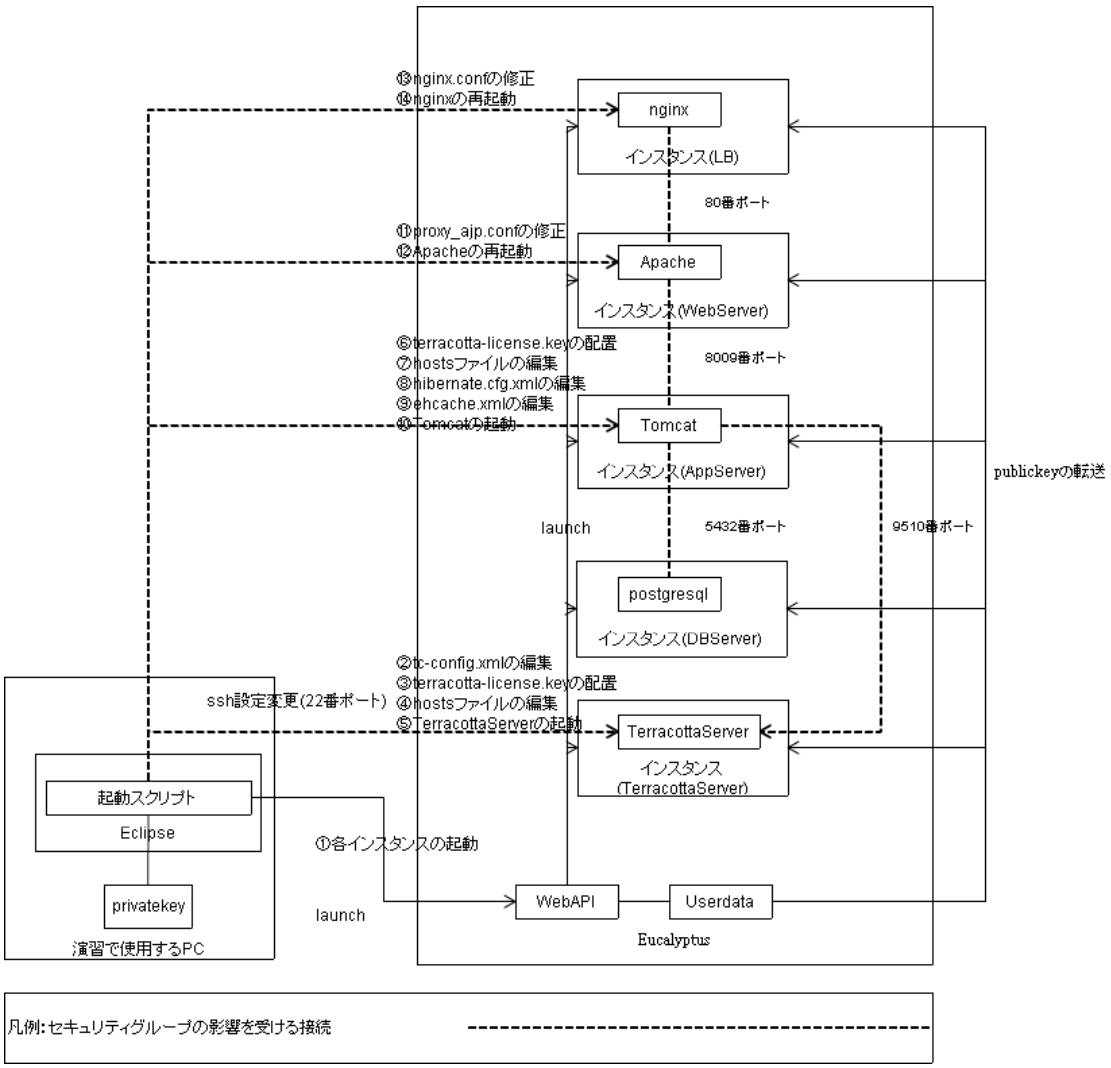

図 57:起動スクリプト実行時の全体の概要

このスクリプトは WebAPI 経由で各インスタンスを起動し、ssh、scp を使い各インスタンスの設定を行います。

詳しくは、本書「4.1起動スクリプトについて」を確認してください。

起動スクリプトを実行するために ecloud-terracotta を eclispe にインポートします。
デスクトップの eclipse アイコン[eclipse3.4.2(JRE1.6)]から「eclipse」を起動し、「ファイル→インポート」をクリックします。(図 58:プロジェクトのインポート)

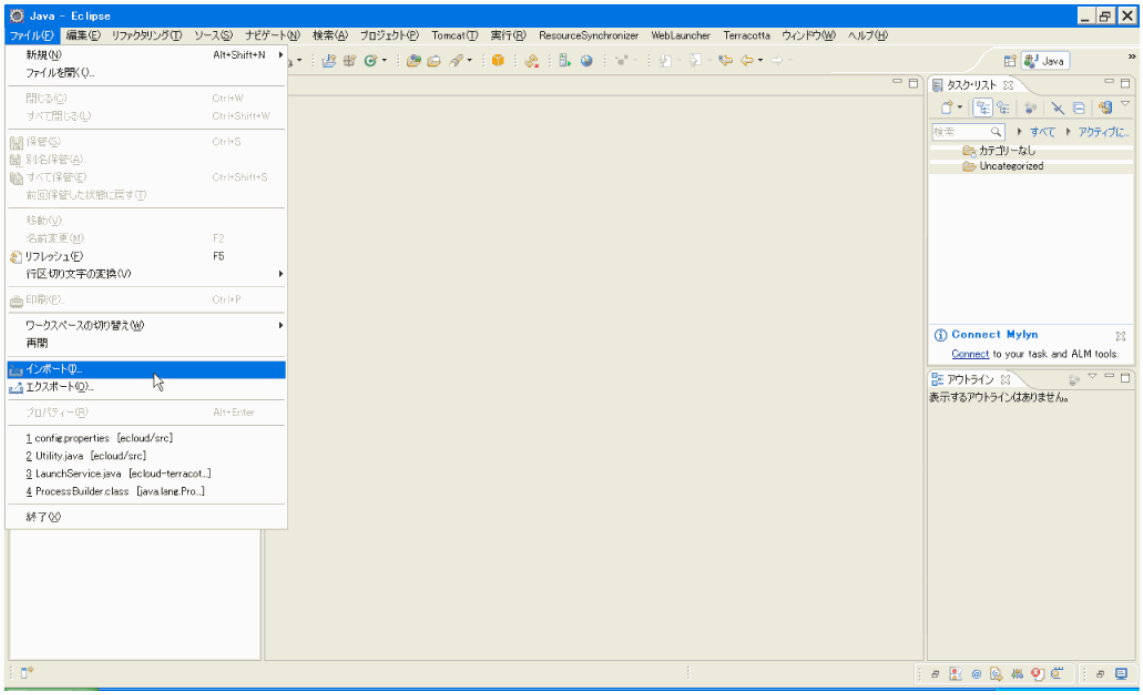

図 58:プロジェクトのインポート

「一般→既存プロジェクトをワークスペースへ」を選択し、「次へ」ボタンをクリックします。(図 59:既存プロジェクトをワークスペースへ)

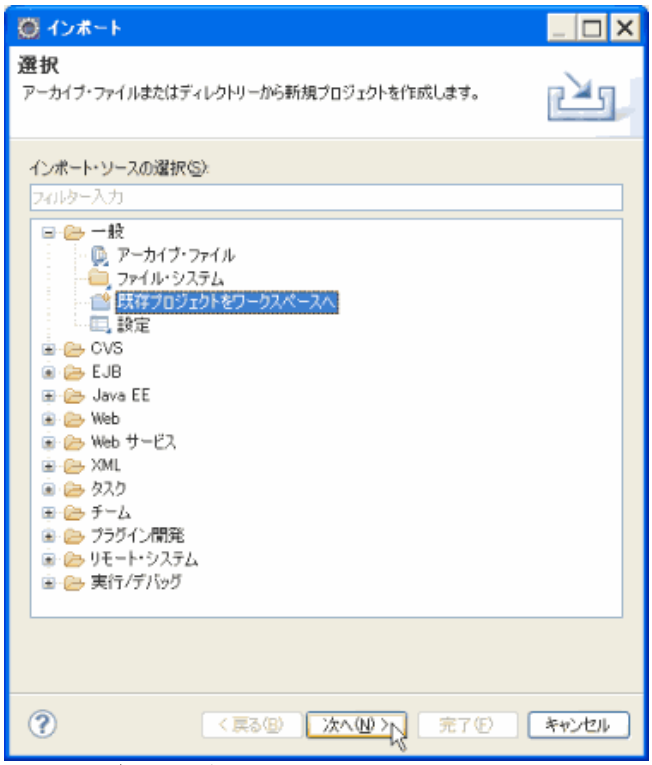

図 59:既存プロジェクトをワークスペースへ

「ルート・ディレクトリの選択」を選択し、「参照」ボタンをクリックします。(図 60:参照先の選択)

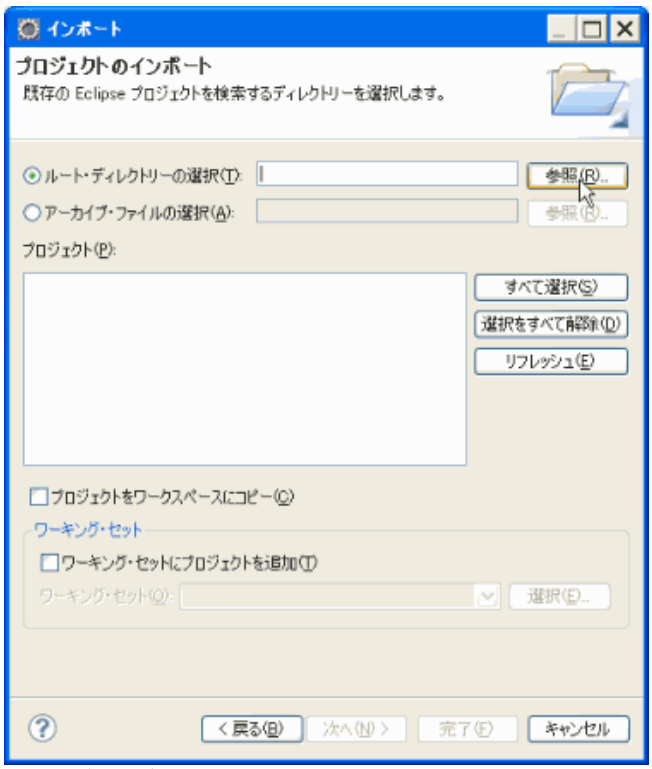

図 60:参照先の選択

「フォルダの参照」で「U:¥usr¥workspace」フォルダの中にある「ecloud-terracotta」を選択して「OK」ボタンをクリックします。(図 61:フォルダの参照)

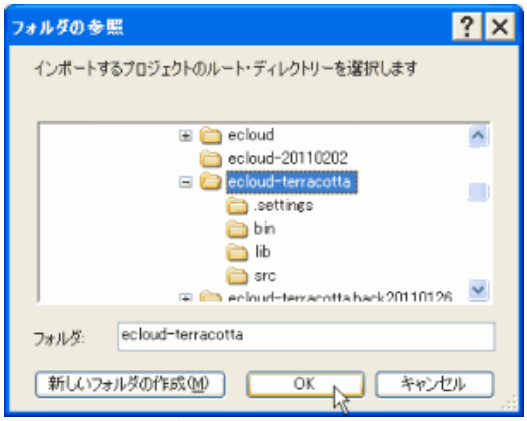

図 61:フォルダの参照

「終了」ボタンをクリックする(図 62:参照の完了)とパッケージ・エクスプローラーに「ecloud-terracotta」が表示されます。(図 63:プロジェクトのインポート確認)

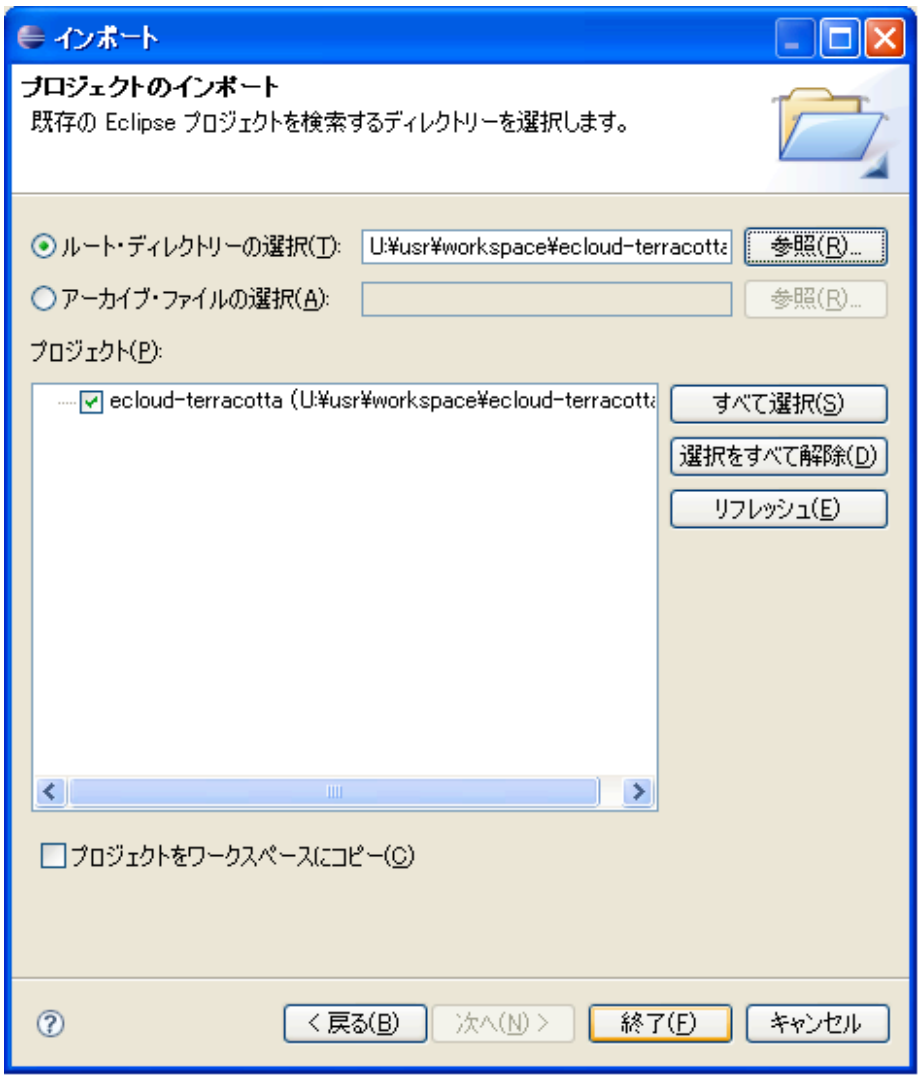

図 62:参照の完了

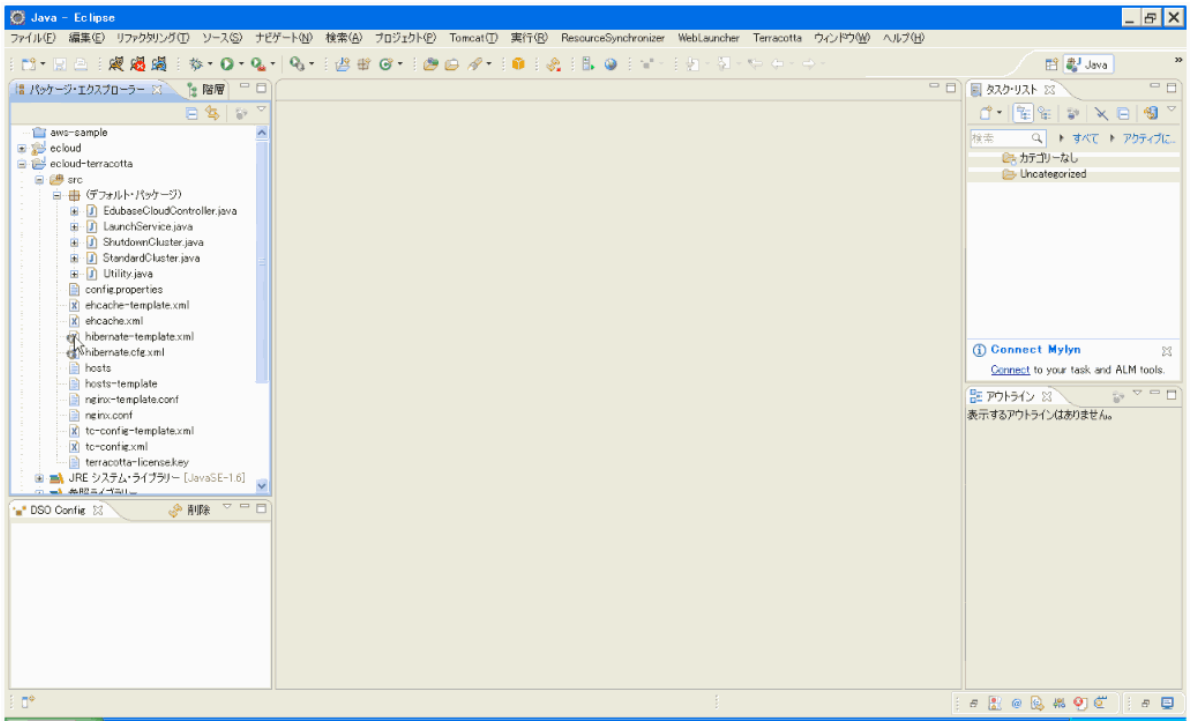

図 63:プロジェクトのインポート確認

次に起動スクリプトの設定ファイルを編集します。
パッケージ・エクスプローラーの中にある「ecloud-terracotta→src→config.properties」を編集します。

ソースコード：config.properties

```
# Eucalyptus Credential
secretKey=secretKey
accessKey=accessKey

# Eucalyptus Web API
hostname=hostname
resourcePrefix=/services/Eucalyptus
port=8773

# Instance
keyName=ログインユーザーID
securityGroup=ログインユーザーID

# Machine Images
dbImageId=イメージ ID
appImageId=イメージ ID
webImageId=イメージ ID
lbImageId=イメージ ID
tcImageId=イメージ ID

# Configuration files for Process (nginx and Tomcat)
srcDir= U:¥¥usr¥¥workspace¥¥ecloud¥¥src¥¥

# Private Key
keyFile=U:¥¥usr¥¥.ec2¥¥[プロジェクト名]¥¥[hostname]¥¥[ログインユーザーID].pem
```

以下の項目の部分を修正します。

- ☐ secretKey
- ☐ accessKey
- ☐ hostname
- ☐ keyName
- ☐ securityGroup
- ☐ keyFile
- ☐ appImageId

Windows の「スタート→すべてのプログラム→クラウドクライアント→CloudClient」を起動し、「ファイル→設定」を選択します。(図 64:クラウドクライアントの設定)

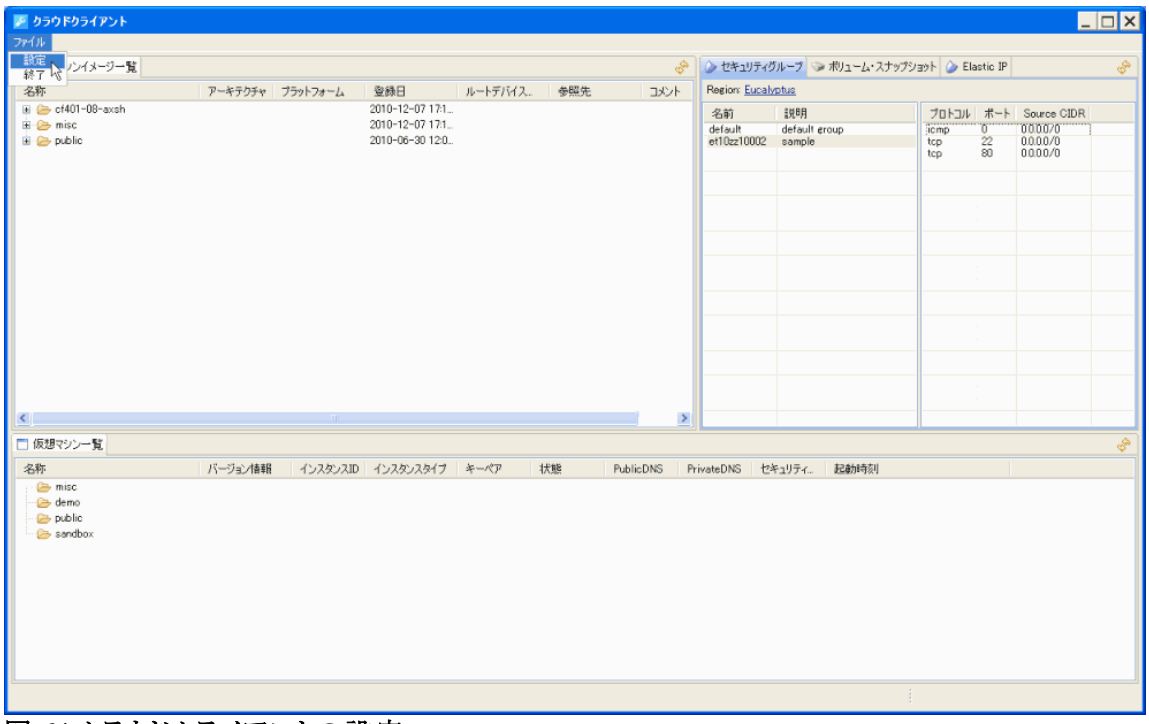

図 64:クラウドクライアントの設定

クラウドクライアントの設定を確認します。(図 65:クラウドクライアントの設定確認)

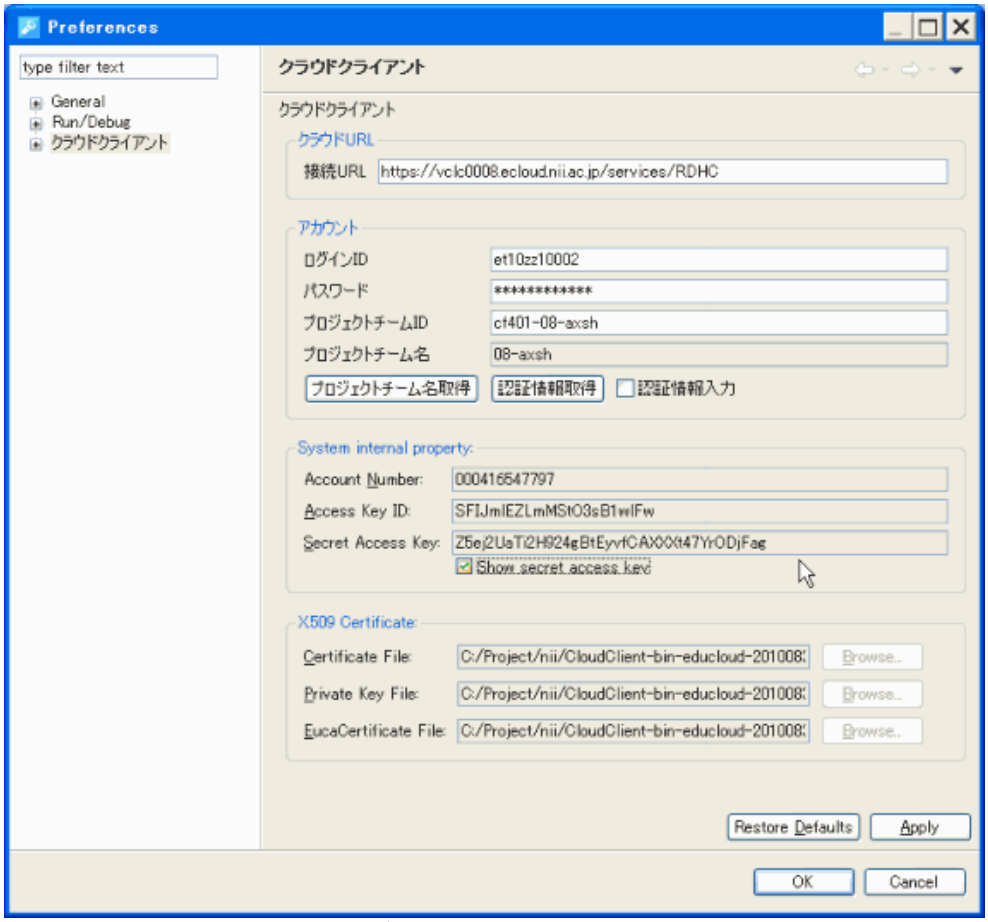

図 65:クラウドクライアントの設定確認

config.properties	クラウドクライアントの Preserceces ダイアログ
secretKey	Secret Access Key
accessKey	Access Key ID
hostname	接続 URL のホスト名(例: vclc0008.ecloud.nii.ac.jp)

「appImageId」には先ほど作成したイメージの ID を入れます。

仮想マシンイメージ一覧の misc フォルダの中にある先ほど作成したイメージの名前にマウスカーソルをあわせるとポップアップウィンドウが開きます。
イメージ ID を範囲選択してコピーします。(図 66:イメージ ID のコピー)

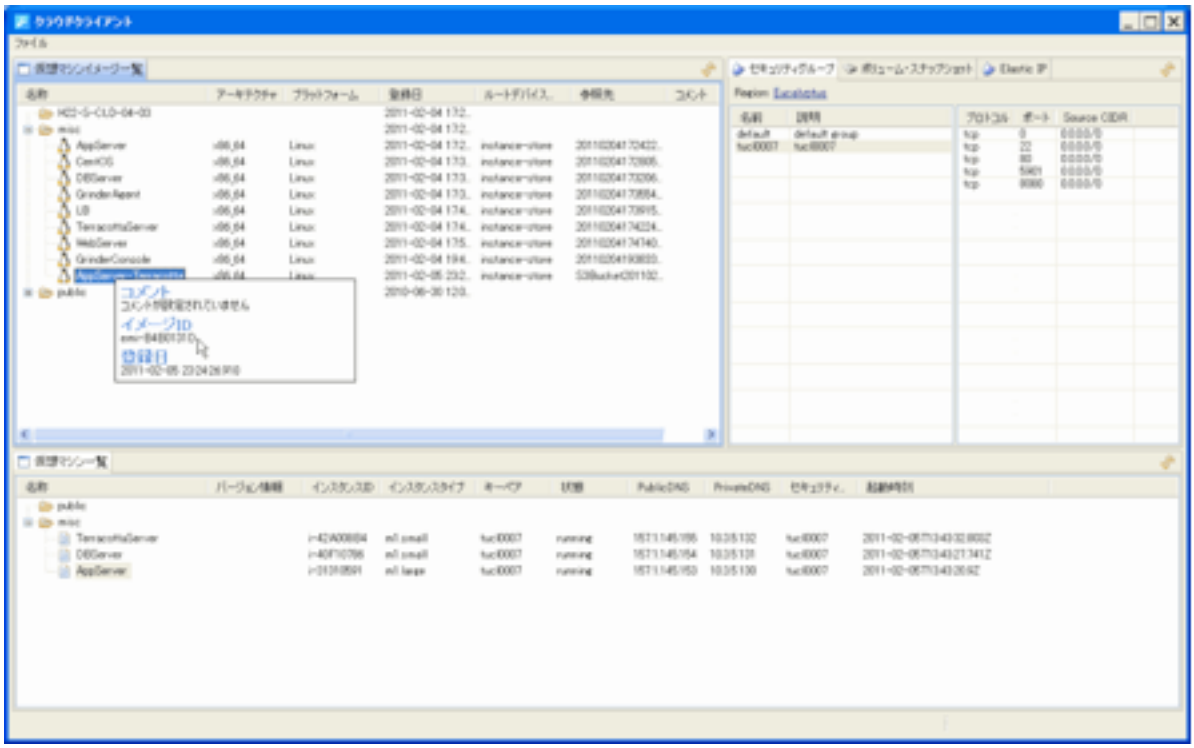

図 66:イメージ ID のコピー

「keyname」、「securityGroup」には自分が使用しているログインユーザーID をいれます。

「keyFile」には「U:\\usr\\.ec2\\[プロジェクト名]\\[hostname]\\[ログインユーザーID].pem」を入れます。

それでは起動スクリプトを実行してみましょう。
パッケージ・エクスプローラーにある「ecloud-terracotta→src→jp.ac.nii.sample→LaunchCluster」を右クリックし「実行→Java アプリケーション」を実行します。(図 67:起動スクリプトの実行)

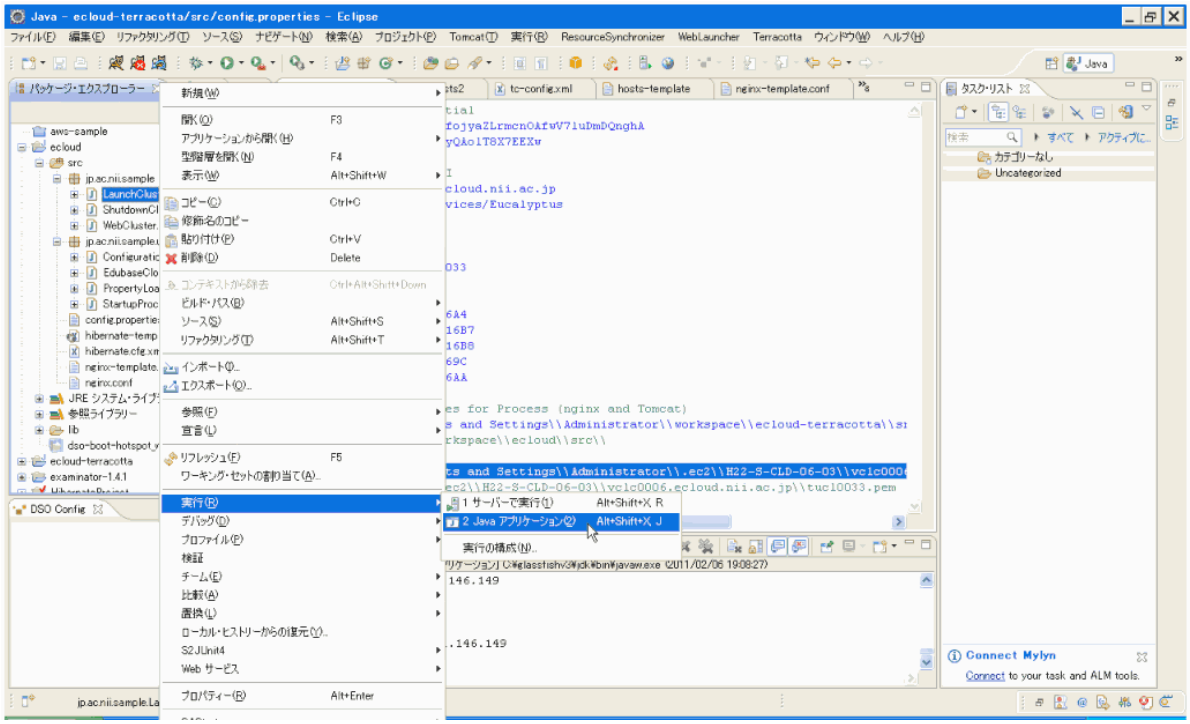

図 67:起動スクリプトの実行

以下のように Script の実行ログが表示されます。(図 68:起動スクリプト実行ログ)

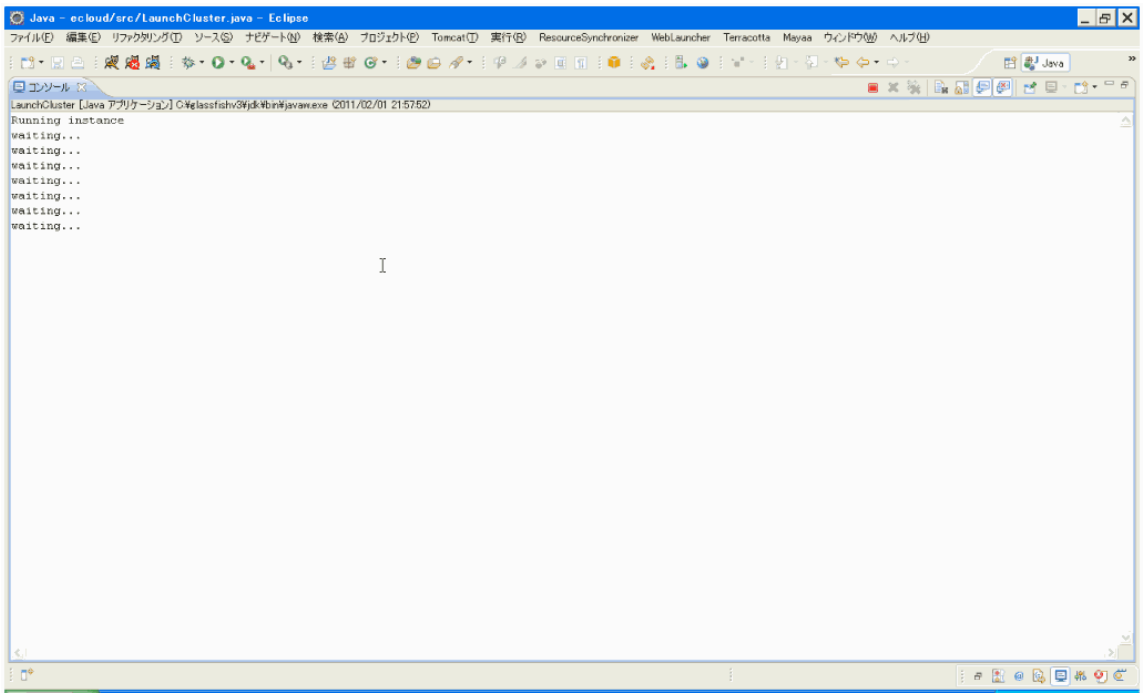

図 68:起動スクリプト実行ログ

「CloudClient」の仮想マシン一覧の右にある「リフレッシュ」ボタンを押すとサーバーが一台づつ起動してくるのが確認できます。(図 69:インスタンスの起動確認)

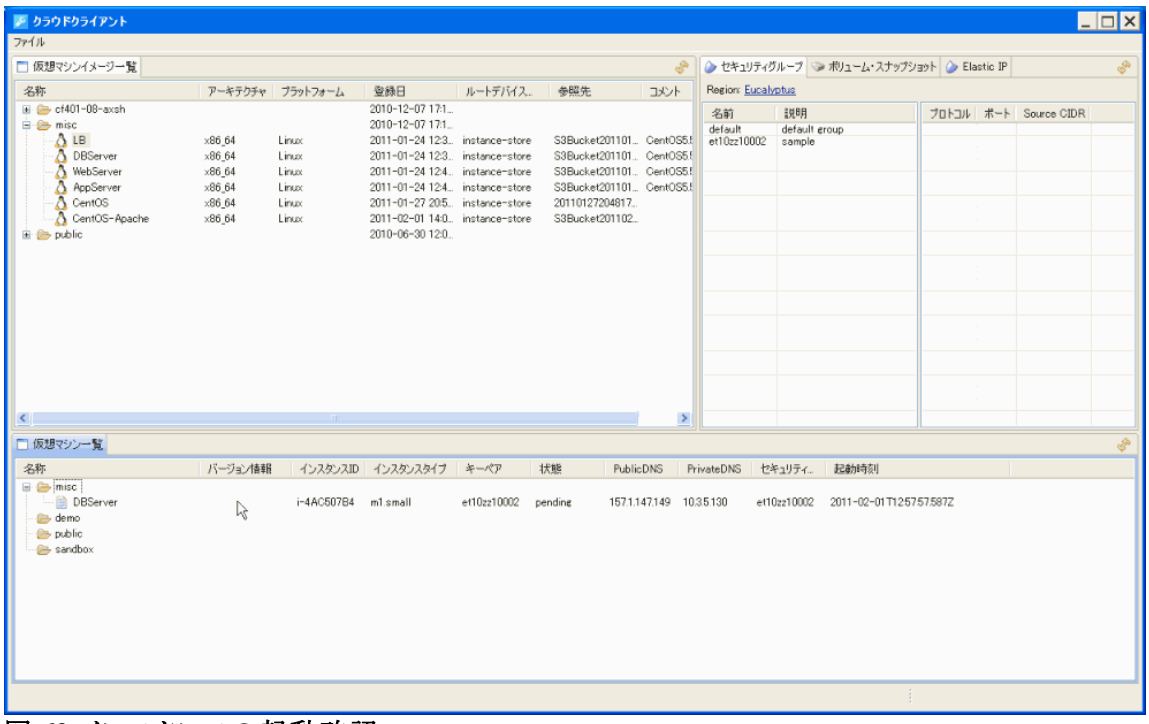

図 69:インスタンスの起動確認

全ての Server の状態が「running」になるのを確認したら動作確認してみましょう。
なお起動には数分かかります。
起動が確認できたら本書「2.2.8動作確認する」を参考に動作確認をしてみてください。

2.4. TerracottaServer をスケールアウトする

ここでは Terracotta をスケールアウトするスクリプトを実行して TerracottaServer を増やします。以下の図がスクリプトを実行した時の全体の構成図になります。(図 70:起動スクリプト実行時の全体構成図)

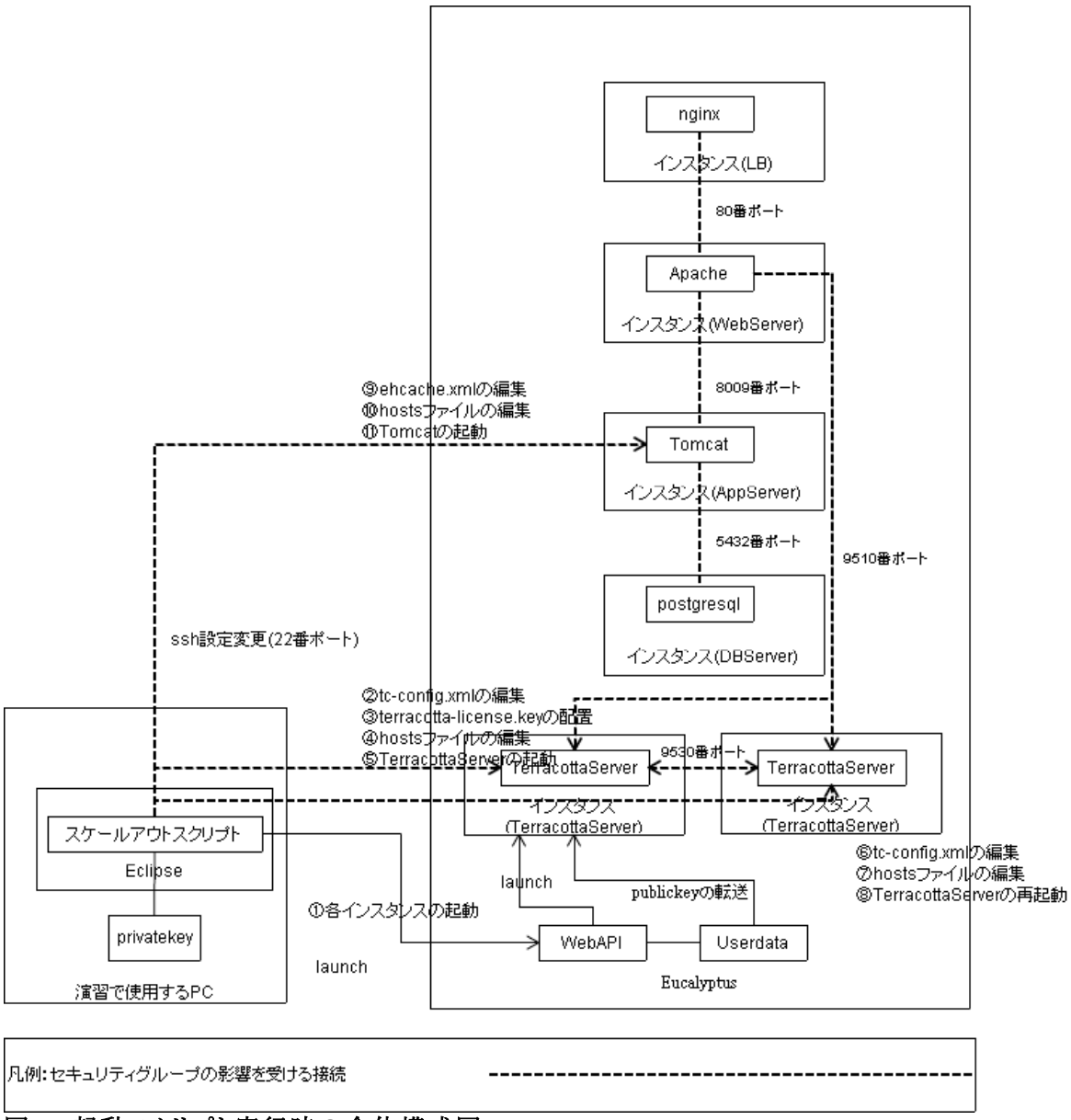

図 70:起動スクリプト実行時の全体構成図

ここでは、WebAPI 経由でインスタンス(TerracottaServer)を起動し、ssh、scp を使用して TerracottaServer と Tomcat の設定を行っています。

詳しくは、本書「4.2Terracotta スケールアウトスクリプトについて」を確認してください。

それでは起動スクリプトを実行してみましょう。
パッケージ・エクスプローラーにある「ecloud-terracotta→src→jp.ac.nii.sample→TerracottaCluster」を右クリックし「実行→Java アプリケーション」を実行します。

起動が確認できたら本書「2.2.8動作確認する」を参考に動作確認をしてみてください。

2.5. Apache2+Tomcat をスケールアウトする

ここでは Terracotta に対応した Apache2+Tomcat をスケールアウトするスクリプトを実行して Apache2と Tomcat を増やします。以下の図がスクリプトを実行した時の全体の構成図になります。

(図 71:Apache2+Tomcat スケールアウトスクリプト実行時の全体構成図)

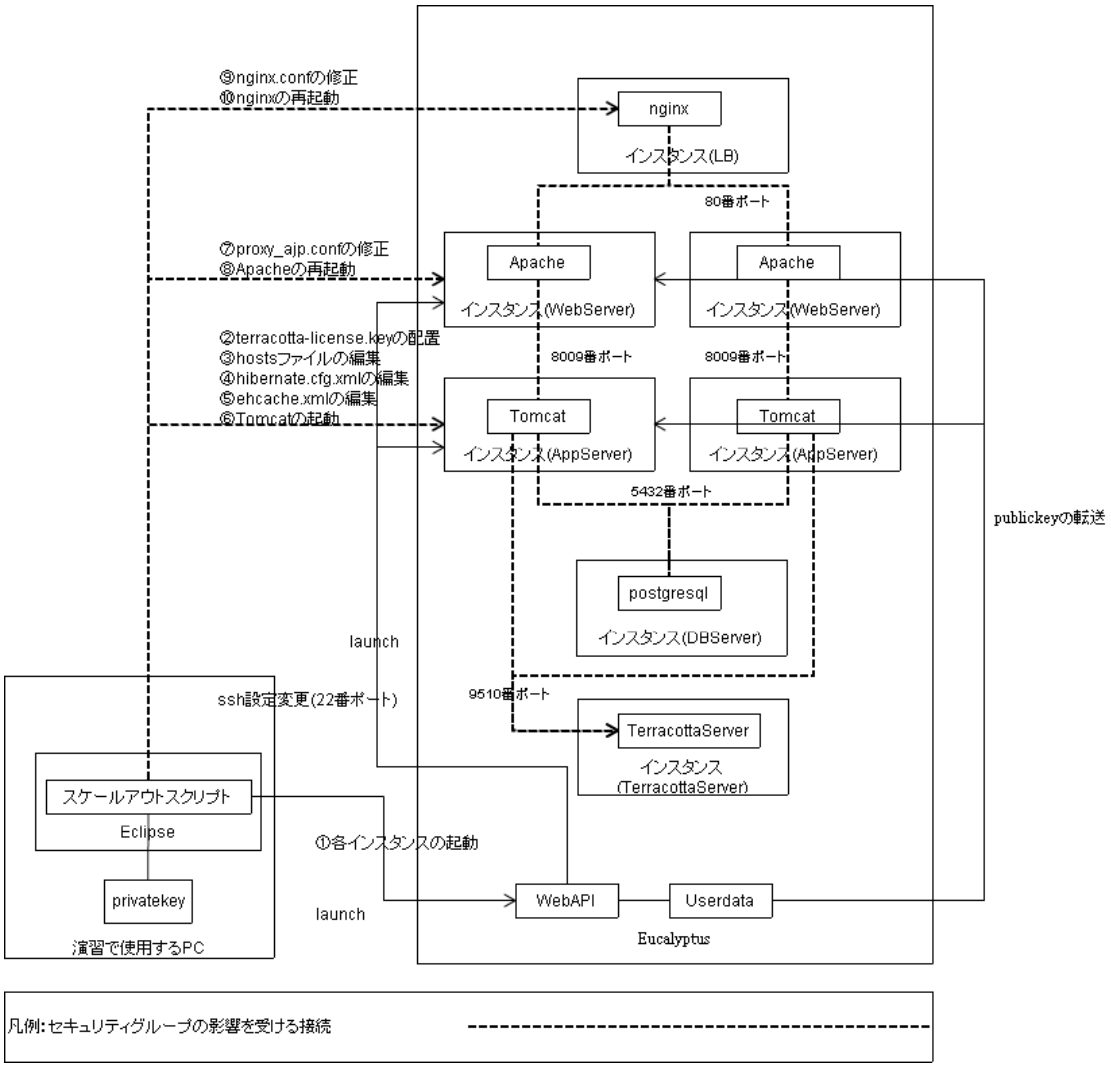

図 71:Apache2+Tomcat スケールアウトスクリプト実行時の全体構成図

ここでは、WebAPI 経由でインスタンス(AppServer)とインスタンス(WebServer)を起動し、ssh、scp を使用して TerracottaServer と Tomcat の設定を行っています。

詳しくは本書「4.3Apache2+Tomcat のスケールアウトスクリプト(Terracotta 対応)について」を確認してください。

それでは起動スクリプトを実行してみましょう。
パッケージ・エクスプローラーにある「ecloud-terracotta→src→ac.jp.nii.sample→WebCluster」を右クリックし「実行→Java アプリケーション」を実行します。

起動が確認できたら本書「2.2.8動作確認する」を参考に動作確認をしてみてください。

3章 【参考】Grinder のシナリオの作り方

Grinder の TCPProxy を使うために Windows 上に以下のサイトから Grinder をダウンロードし、ア

ーカイブを所定のフォルダに解凍します。

URL: http://sourceforge.net/projects/grinder/

Windows の「スタート→すべてのプログラム→アクセサリ→コマンドプロンプト」を実行し、先ほど Grinder を解凍したフォルダへ移動します。

TCPProxy を以下のコマンドで起動します。

```
C:¥Documents and Settings¥Administrator¥My Documents¥grinder-3.4>
java -cp lib¥grinder.jar net.grinder.TCPProxy -http > grinder.py

1/17/11 7:14:21 AM (tcpproxy): Initialising as an HTTP/HTTPS proxy with the

parameters:

  Request filters:     HTTPRequestFilter
  Response filters:    HTTPResponseFilter
  Local address:       localhost:8001

1/17/11 7:14:22 AM (tcpproxy): Engine initialised, listening on port 8001
```

次にブラウザのプロキシを設定します。

Firefox の場合

「ツール」メニューの「オプション」をクリックします。(図 72:Firefox のオプション)

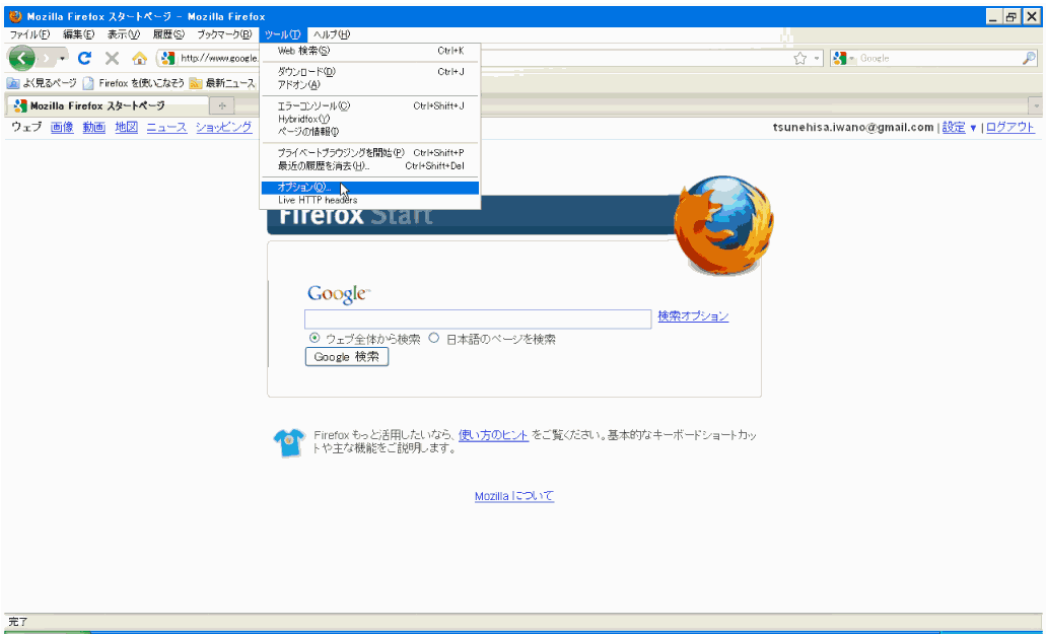

図 72:Firefox のオプション

「詳細」をクリックし「ネットワーク」タブの「接続設定」ボタンを押します。(図 73:Firefox の接続設定)

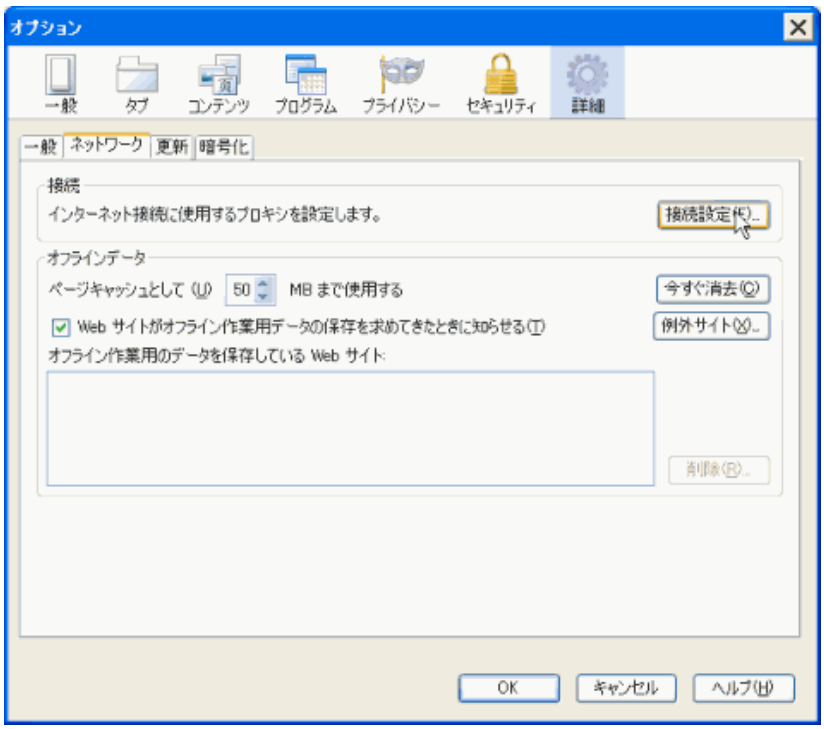

図 73:Firefox の接続設定

「手動でプロキシを設定する」を選び「HTTP プロキシ」に「localhost」、ポートに「8001」を入力して OK を押します。(図 74:Firefox のプロキシ設定)

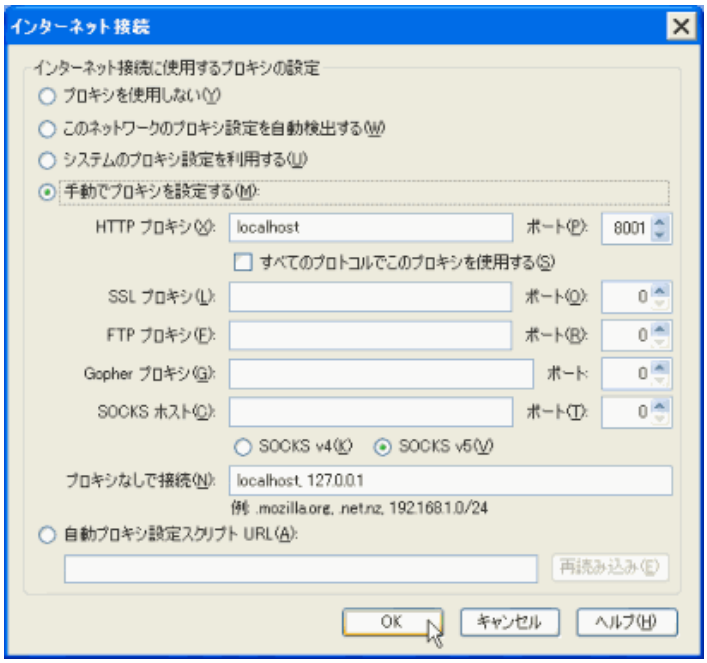

図 74:Firefox のプロキシ設定

InternetExploer の場合
「ツール」メニューの「インターネットオプション」をクリックします。(図 75:InternetExplorer のインターネットオプション)

図 75:InternetExplorer のインターネットオプション

「接続」タブの「LAN の設定」ボタンを押します。(図 76:InternetExplorer の LAN の設定)

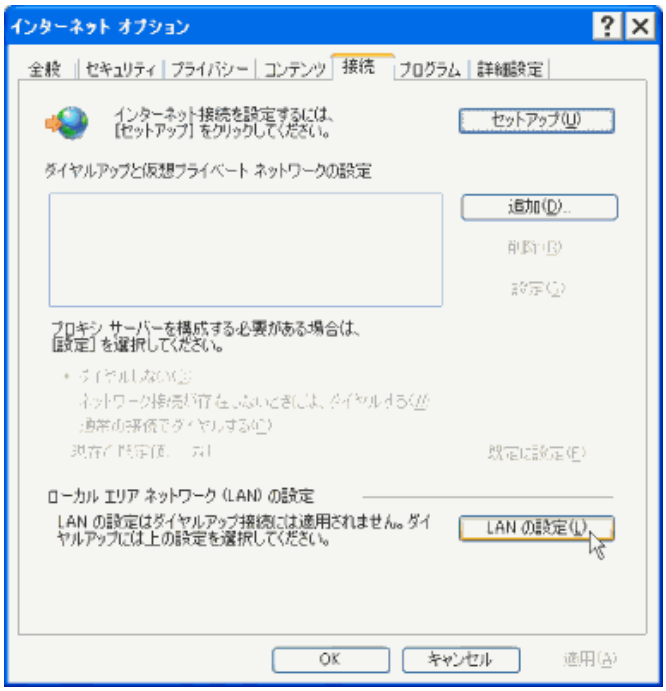

図 76:InternetExplorer の LAN の設定

「LAN にプロキシサーバーを使用する」にチェックを入れ「アドレスに」に「localhost」、「ポート」に「8001」を入力して OK を押します。(図 77:InternetExplorer のプロキシ設定)

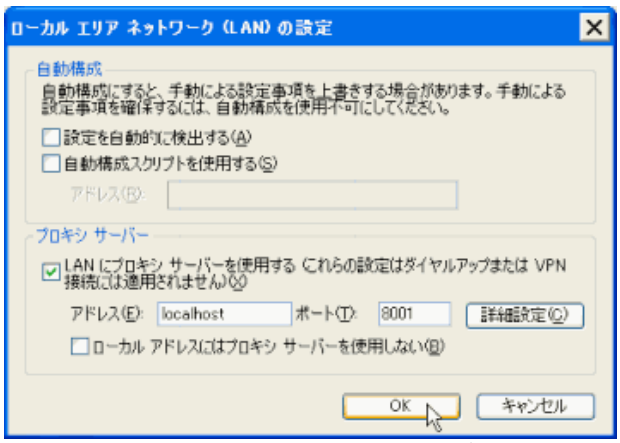

図 77:InternetExplorer のプロキシ設定

Chrome の場合
URL ツールバーの右にあるスパナのマークをクリックし、「オプションを」クリックします。(図 78:Chrome のオプション)

図 78:Chrome のオプション

「高度な設定」タブをクリックし、ネットワークの「プロキシ設定を変更」ボタンを押します。(図 79:Chrome の高度な設定)

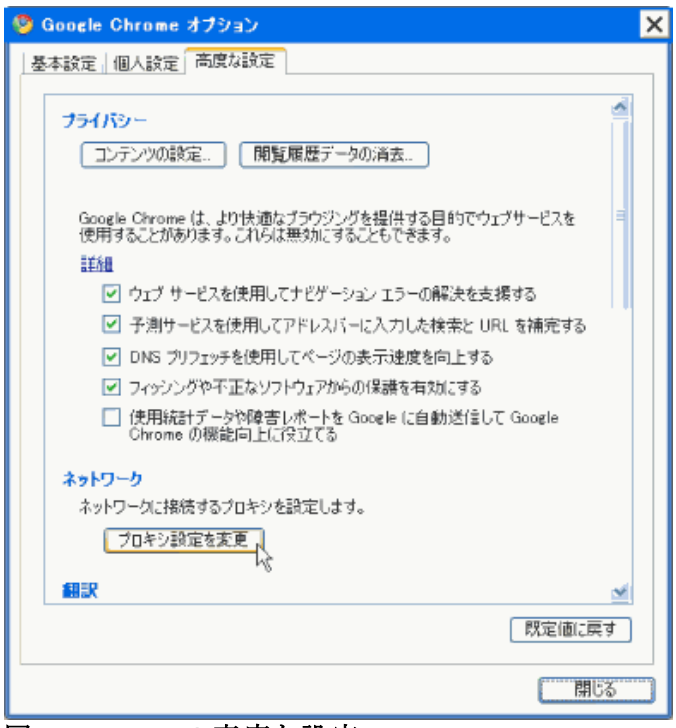

図 79:Chrome の高度な設定

「接続」タブの「LAN の設定」ボタンをクリックします。（図 80:Chrome の LAN の設定）

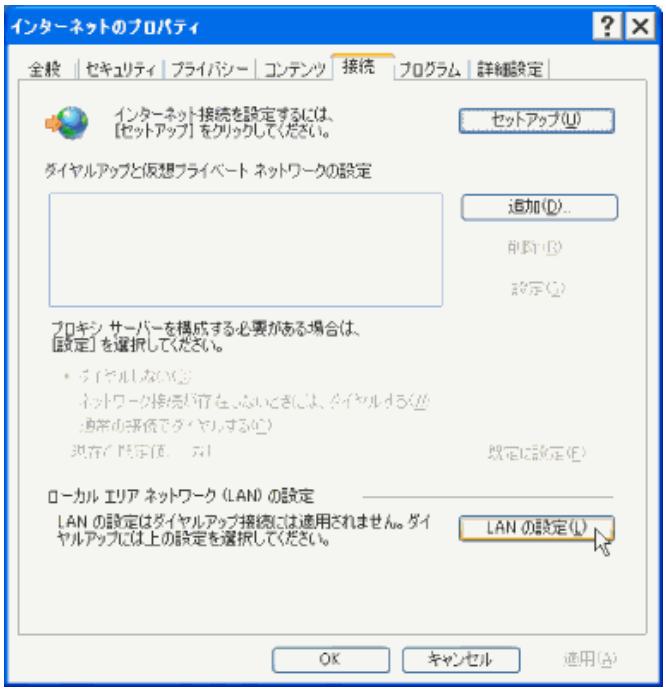

図 80:Chrome の LAN の設定

「LAN にプロキシサーバーを使用する」にチェックをいれ「アドレス」に「localhost」、「ポート」に「8001」を入力して、OK を押します。（図 81:Chrome のプロキシ設定）

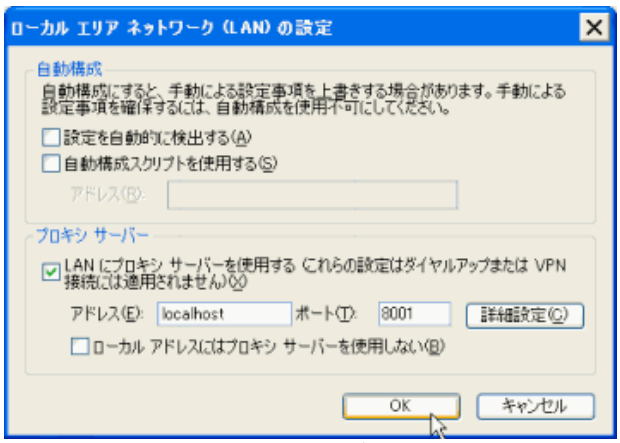

図 81:Chrome のプロキシ設定

プロキシの設定が終わったら、WEB ブラウザで以下の URL にアクセスします。(図 82:シナリオの作成)

URL: http://[Lb のパブリック IP]/sample-kejiban/

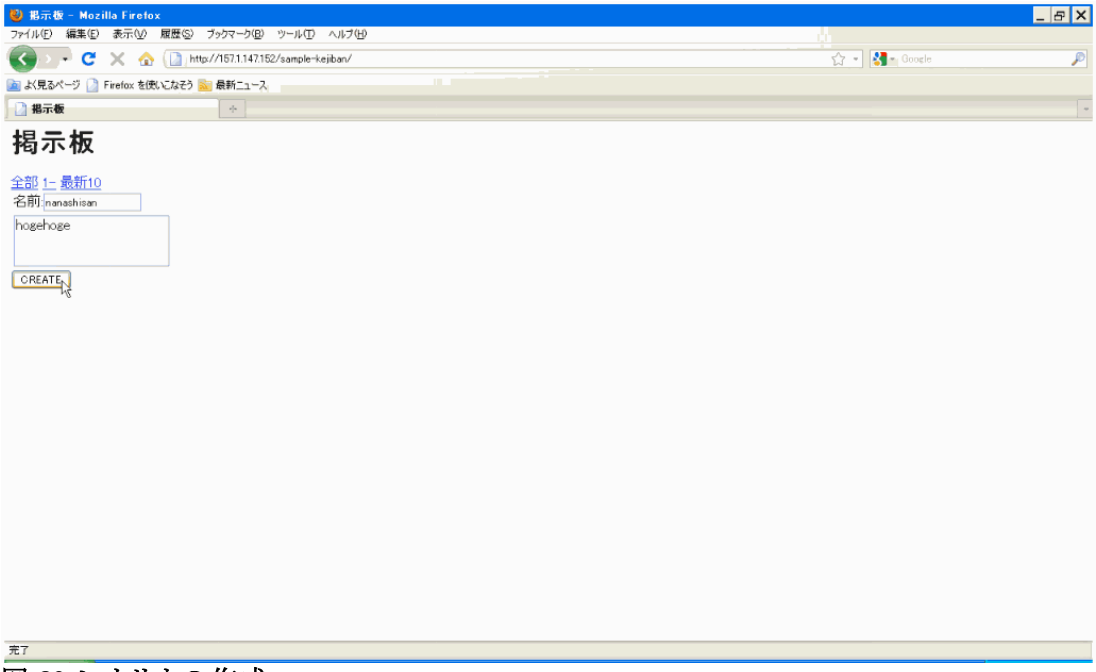

図 82:シナリオの作成

「名前」と「内容」を書き込んで「CREATE」ボタンを押します。
先ほどコマンドプロンプトで立ち上げたプロキシサーバーを落とします。(Control + C などで)
ブラウザのプロキシ設定も消しておいてください。

grinder.py ファイルができているのを確認します。
中身を見てみると今ブラウザでやったリクエストが記録されているのがわかります。

4章 【参考】スクリプトについて

4.1. 起動スクリプトについて

今回使用したスクリプトは以下のような手順で動きます。(図 83:起動スクリプト実行時の全体構成図)

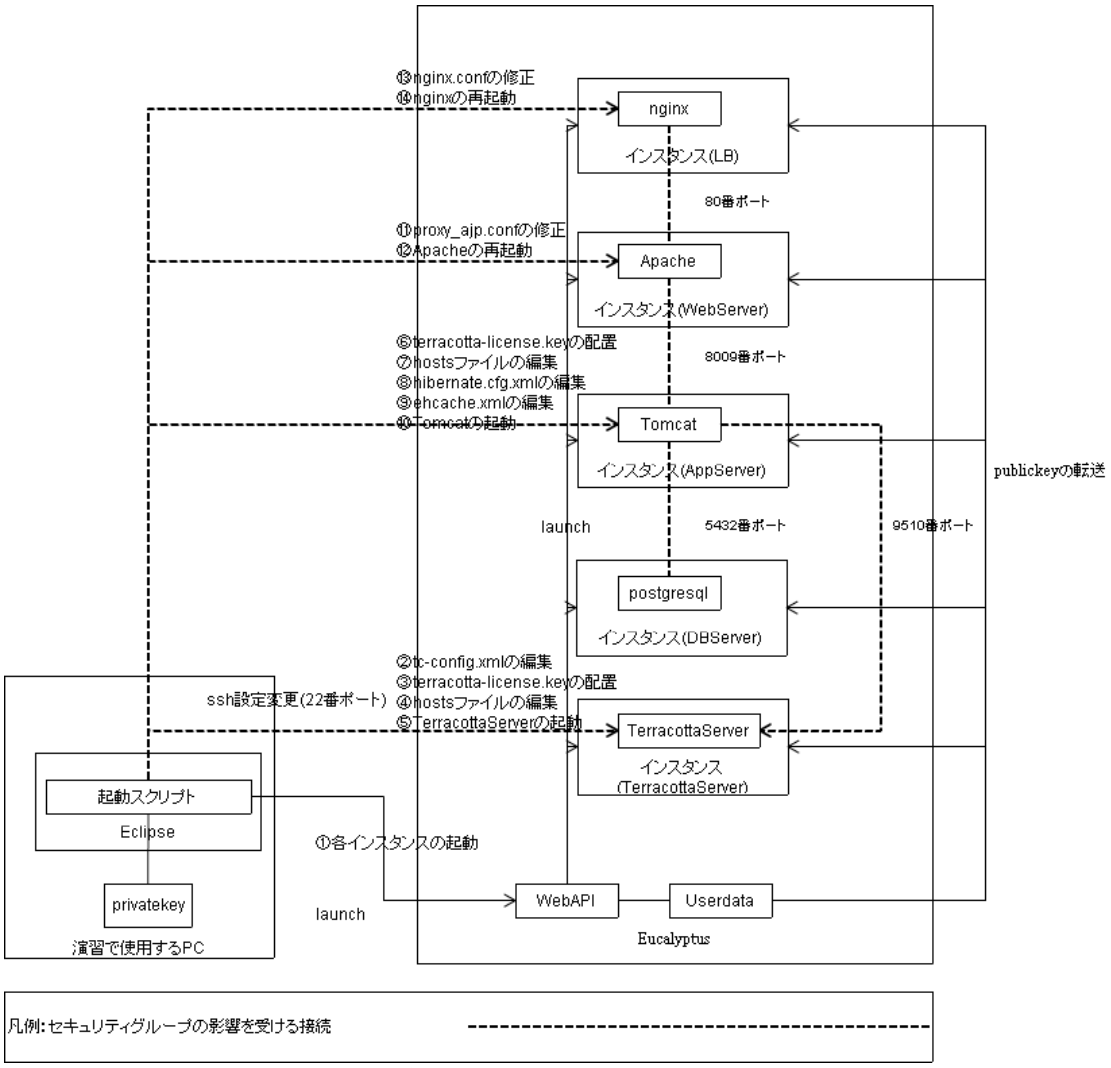

図 83:起動スクリプト実行時の全体構成図

□各インスタンスを WebAPI 経由で起動する
□tc-config.xml の編集
□terracotta-license.key の配置
□hosts ファイルの編集
□TerracottaServer の起動
□terracotta-license.key の配置
□hosts ファイルの編集
□hibernate.cfg.xml の編集
□ehcache.xml の編集
□Tomcat の起動

□proxy_ajp.conf の修正
□Apache の再起動
□nginx の修正
□nginx の再起動

前回使用した起動スクリプトとの違いは以下8つの手順になります。
□各インスタンスを WebAPI 経由で起動する(TerracottaServer の起動)
□tc-config.xml の編集
□terracotta-license.key の配置
□hosts ファイルの編集
□TerracottaServer の起動
□terracotta-license.key の配置
□hosts ファイルの編集
□ehcache.xml の編集

各ファイルの場所と変更箇所は以下になります。

tc-config.xml
ファイルはインスタンス(TerracottaServer)の「/opt/terracotta」以下にあります。

```
<?xml version="1.0" encoding="UTF-8"?>
<con:tc-config xmlns:con="http://www.terracotta.org/config">

<tc-properties>
        <property name="productkey.path" value="/opt/terracotta/terracotta-license.key" />
</tc-properties>

<servers>
        <server host="TerracottaServer の instanceId" name="TerracottaServer の InstanceId">
                <dso-port bind="0.0.0.0">9510</dso-port>
                <jmx-port bind="0.0.0.0">9520</jmx-port>
                <data>terracotta/server-data</data>
                <logs>terracotta/server-logs</logs>
                <statistics>terracotta/cluster-statistics</statistics>
                <l2-group-port>9530</l2-group-port>
         </server>

        <mirror-groups>
                <mirror-group group-name="TerracottaServer の InstanceId">
                        <members>
                                <member>TerracottaServer の InstanceId</member>
                        </members>
                </mirror-group>
        </mirror-groups>
</servers>

<clients>
        <logs>terracotta/client-logs</logs>
</clients>

</con:tc-config>
```

trerracotta-license.key

事前にアーカイブからダウンロードしてきている terracotta-license.key をインスタンス

(TerracottaServer)とインスタンス(AppServer)に WinSCP 転送します。
配置場所は以下になります。

インスタンス(TerracottaServer)側
「/opt/terracotta」

インスタンス(AppServer)側

「/opt/sample-kejiban/src/main/webapp/WEB-INI/classes」

hosts

ファイルはインスタンス(TerracottaServer)とインスタンス(AppServer)の「/etc/」以下にあります

変更箇所は以下になります。
インスタンス(TerracottaServer)側
TerracottaServer のインスタンス ID を記述します。

```
-bash-3.2# vi /etc/hosts
127.0.0.1 localhost.localdomain localhost
127.0.0.1 [TerracottaServer の InstanceID]

例
127.0.0.1 i-58C209B4
```

インスタンス(AppServer)側
TerracottaServer のプライベート IP とインスタンス ID を記述します。

```
-bash-3.2# vi /etc/hosts
127.0.0.1 localhost.localdomain localhost
[TerracottaServer のプライベート IP] [TerracottaServer の InstanceID]


例
10.3.5.130 i-58C209B4
```

ehcache.xml

変更箇所は以下になります。

```
<?xml version="1.0" encoding="UTF-8"?>
<ehcache xsi:noNamespaceSchemaLocation="ehcache.xsd" name="sample-kejiban">
        <defaultCache maxElementsInMemory="10000"timeToIdleSeconds="3600" timeToLiveSeconds="86400">
                <terracotta/>
        </defaultCache>

        <cache name="kejiban.entity.Res" maxElementsInMemory="10000" eternal="false"  timeToIdleSeconds="0"
timeToLiveSeconds="120">
                <terracotta/>
        </cache>

        <terracottaConfig url="TerracottaServer のプライベート IP:9510"/>
</ehcache>
```

各サーバーの起動、再起動方法は以下になります。

TerracottaServer の起動方法

```
-bash-3.2# /etc/init.d/terracotta start インスタンス ID
```

Tomcat の起動方法

```
-bash-3.2# /etc/init.d/tomcat start
```

Apache の再起動方法

```
-bash-3.2# /etc/init.d/httpd restart
```

Nginx の再起動方法

```
-bash-3.2# /etc/init.d/httpd restart
```

4.2. Terracotta スケールアウトスクリプトについて

Terracotta のスケールアウトスクリプトは以下の手順で動きます。(図 84:TerracottaServer スケールアウトスクリプト実行時の全体構成図)

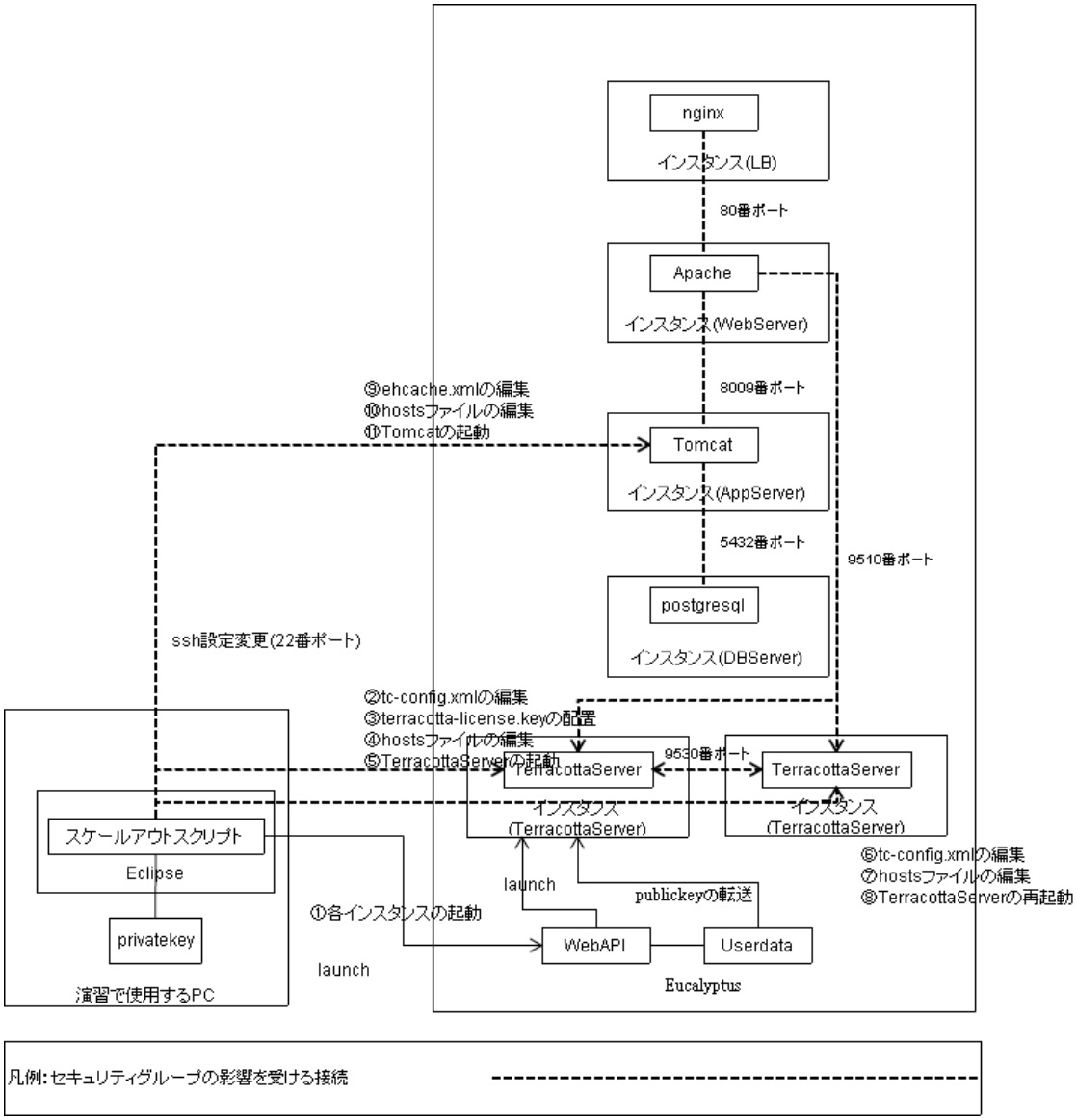

図 84:TerracottaServer スケールアウトスクリプト実行時の全体構成図

□各インスタンスの起動
□tc-config.xml の編集
□terracotta-license.key の配置
□hosts ファイルの編集
□TerracottaServer の起動
□tc-config.xml の編集
□hosts ファイルの編集
□TerracottaServer の再起動
□ehcache.xml の編集
□hosts ファイルの編集
□Tomcat の起動

起動スクリプトの変更点は以下2つになります。
□□tc-config.xml の編集
□□□hosts ファイルの編集

また今回のスクリプトではインスタンス(WebServer)とインスタンス(LB)の設定は行いません。
tc-config.xml
ファイルは、TerracottaServer の「/opt/terracotta/」にあります。

変更箇所は、以下のように Server を増やすたびに<server>タグと<mirror-group>タグを追加します。

「host」、「name」、「group-name」、「member」には TerracottaServer の「インスタンス Id」を記述します。

```
<?xml version="1.0" encoding="UTF-8"?>
<con:tc-config xmlns:con="http://www.terracotta.org/config">

<tc-properties>
 <property name="productkey.path" value="/opt/terracotta/terracotta-license.key" />
</tc-properties>

<servers>
 <server host=" TerracottaServer の InstanceId " name=" TerracottaServer の InstanceId "
bind="0.0.0.0">
 <dso-port bind="0.0.0.0">9510</dso-port>
 <jmx-port bind="0.0.0.0">9520</jmx-port>
 <data>terracotta/server-data</data>
 <logs>terracotta/server-logs</logs>
 <statistics>terracotta/cluster-statistics</statistics>
 <l2-group-port>9350</l2-group-port>
 </server>

 <server host=" TerracottaServer の InstanceId " name=" TerracottaServer の InstanceId ">
 <dso-port bind="0.0.0.0">9510</dso-port>
 <jmx-port bind="0.0.0.0">9520</jmx-port>
 <data>terracotta/server-data</data>
 <logs>terracotta/server-logs</logs>
 <statistics>terracotta/cluster-statistics</statistics>
 <l2-group-port>9530</l2-group-port>
 </server>
```

```
 <mirror-groups>
 <mirror-group group-name=" TerracottaServer の InstanceId ">
 <members>
 <member> TerracottaServer の InstanceId </member>
 </members>
 </mirror-group>
 <mirror-group group-name=" TerracottaServer の InstanceId ">
 <members>
 <member> TerracottaServer の InstanceId </member>
 </members>
 </mirror-group>
 </mirror-groups>
</servers>

<clients>
 <logs>terracotta/client-logs</logs>
</clients>
</con:tc-config>
```

今回は作成した tc-config.xml をすべての TerracottaServer に配置します。

hosts

ファイルは、インスタンス(TerracottaServer)とインスタンス(AppServer)の「/etc/」にあります。

変更箇所は、以下のようになります。

TerracottaServer 側

以下のように全ての TerracottaServer の「プライベート IP」と「インスタンス Id」を記述します。
また、自分の Server には「127.0.0.1」を記述します。

```
127.0.0.1 i-311C0582
10.3.5.130 i-58C29B4
```

AppServer 側

以下のように全ての TerracottaServer の「プライベート IP」と「インスタンス Id」を記述します。

```
10.3.5.130 i-311C0582
10.3.5.131 i-58C29B4
```

各サーバーの起動、再起動方法はいかになります。

TerracottaServer の停止方法

```
-bash-3.2# /etc/init.d/terracotta stop
```

TerracottaServer の起動方法

```
-bash-3.2# /etc/init.d/terracotta start インスタンス ID
```

Tomcat の停止方法

```
-bash-3.2# /etc/init.d/tomcat stop
```

Tomcat の起動方法

```
-bash-3.2# /etc/init.d/tomcat start
```

4.3. Apache2+Tomcat のスケールアウトスクリプト (Terracotta 対応)について

Apache2+Tomcat のスケールアウトスクリプトは以下の手順で動きます。(図 85:Apache2+Tomcat スケールアウトスクリプト実行時の全体構成図)

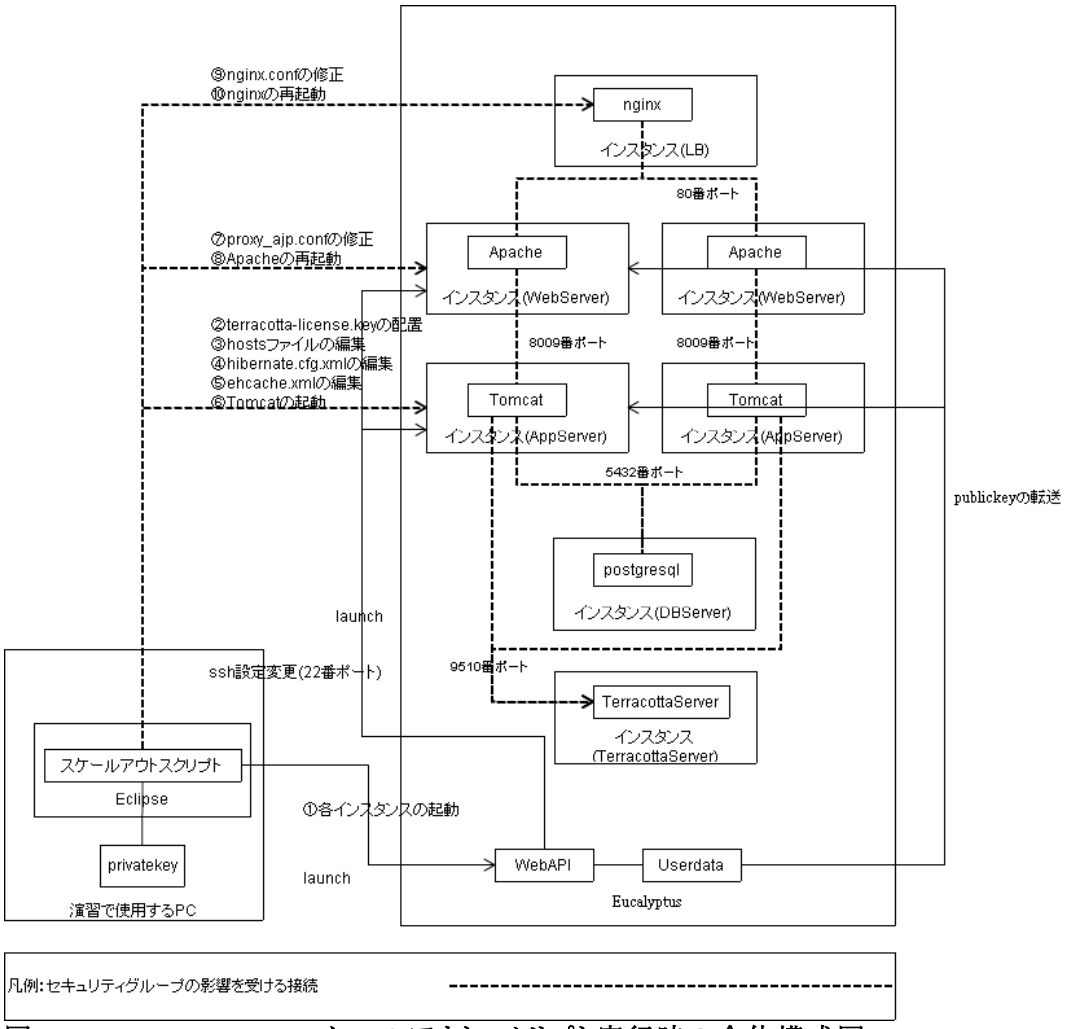

図 85:Apache2+Tomcat スケールアウトスクリプト実行時の全体構成図

□各インスタンスの起動
□terracotta-license.key の配置
□hosts ファイルの編集
□hibernate.cfg.xml の編集
□ehcache.xml の編集

□Tomcat の起動
□proxy_ajp.conf の編集
□Apache の再起動
□nginx.conf の編集
□nginx の再起動

起動スクリプトの変更点は以下1つになります。
□nginx.conf の編集

また今回は TerracottaServer 側の設定は行いません。

nginx.conf
ファイルは Lb の「/usr/local/nginx/conf/」にあります。

変更箇所は以下になります。
今回は自分が起動しているすべての WebServer のプライベート IP を記述します。

```
http {
----------------------略--------------------------
 upstream backend {
 server WebServer のプライベート IP;
 }

------------------------略-------------------------
}
```

各サーバーの起動、再起動方法は以下になります。

Tomcat の起動方法

```
-bash-3.2# /etc/init.d/tomcat start
```

Apache の再起動方法

```
-bash-3.2# /etc/init.d/httpd restart
```

Nginx の再起動方法

```
-bash-3.2# /etc/init.d/nginx restart
```