IDCFクラウド

活用マニュアル

～オートスケールを設定する～

オートスケールを設定する

目次

IDCF Cloud

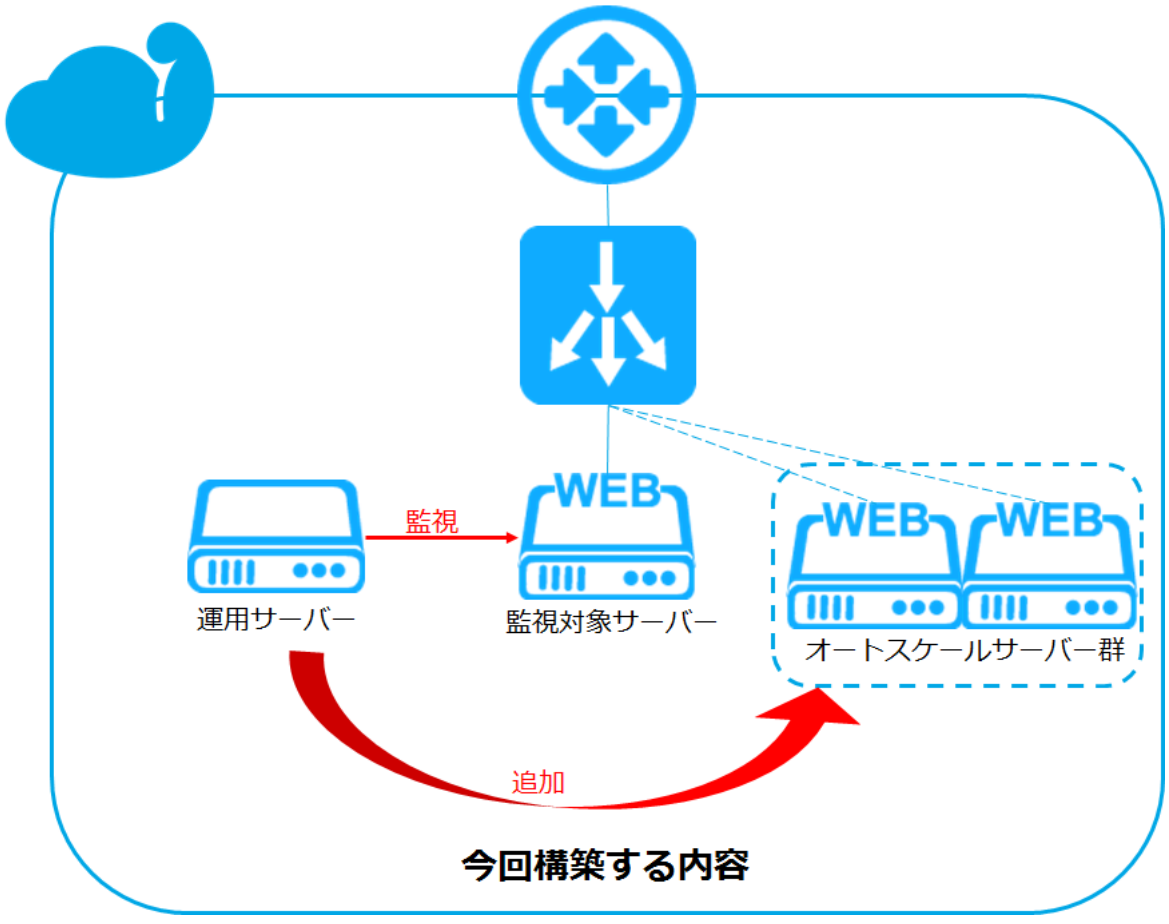
最終更新日：2016/1/31

オートスケールを設定する

IDCFクラウドでは仮想ルーターを使用してロードバランシングをする事が可能です。スクリプトにより、Webサーバー等への負荷が継続した際に検知をし、自動でテンプレートから仮想マシンを作成してロードバランサーへ登録（スケールアウト）する事ができます。スケールアウトを使用する事でWebサーバーへの負荷が急激に上昇しても、負荷の閾値を設定していれば自動で仮想サーバーを作成し、各仮想サーバーへの負荷分散を行います。負荷を軽減する事で、急激なアクセスでもサーバーからのレスポンスを一定にコントロールする事ができます。サーバーの増減を自動で行う（オートスケール）ことで、運用負荷も軽減させる事ができます。

オートスケールを使用する形で構築するには、元となるサーバーのテンプレートを作成してAPIが使用出来る環境を作成する必要があります。また、閾値を監視する為、監視元と監視対象のサーバーを用意する必要があります。（本書ではテスト実行の為、最終的に3台構成となります。）

本書の中で監視元サーバーを「運用サーバー」、監視対象サーバーを「監視対象サーバー」と記載しています。



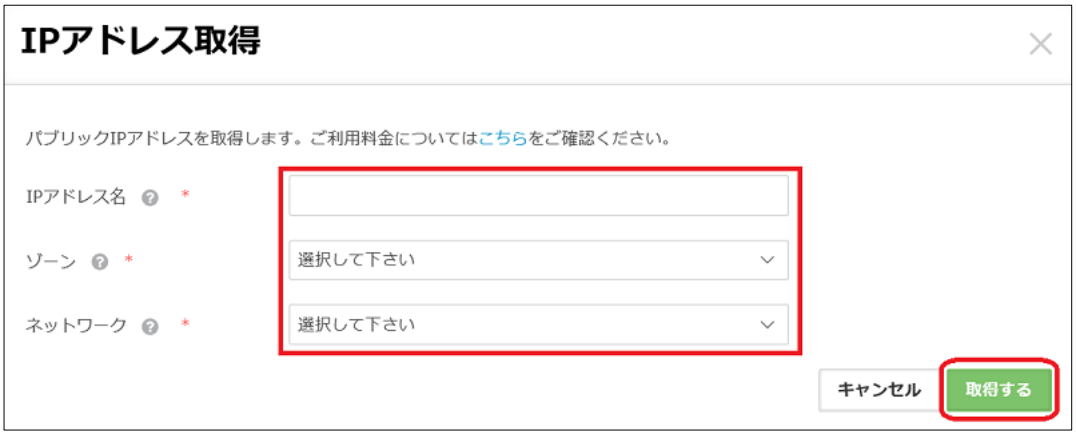
(1)監視対象サーバーの構築とテンプレート準備

監視対象サーバーを構築し、テンプレートを作成します。監視対象サーバーの作成は、「Webサイトの本番環境を構築したい（Web1台構成）」を参照して行います。以降の手順は「Webサイトの本番環境を構築したい（Web1台構成）」の(1)仮想マシンの作成まで完了した状態からの手順となります。尚、本書の仮想マシン名は「scale01」としている為、仮想マシン名指定の際には「sacle01」として下さい。（https://www.idcf.jp/cloud/pdf/manual_002.pdf）

① コンピューティングのメニューより「IPアドレス」をクリックし、IPアドレス設定画面を表示します。右側の「IPアドレス追加」をクリックします。

② コンピューティングのメニューより「IPアドレス」をクリックし、IPアドレス設定画面を表示します。右側の「IPアドレス追加」をクリックします。

項目	設定内容
IPアドレス名	Scale（任意）
ゾーン	仮想マシンを作成したゾーン
ネットワーク	仮想マシンを作成したネットワーク



③ IPアドレス取得しました。というメッセージが表示されるので、取得した「IPアドレス」をクリックします。

④ ファイアウォールをクリックし、ルールを図のように3つ作成します。作成は1つづつ行い、「＋」ボタンで追加します。

・ssh-web

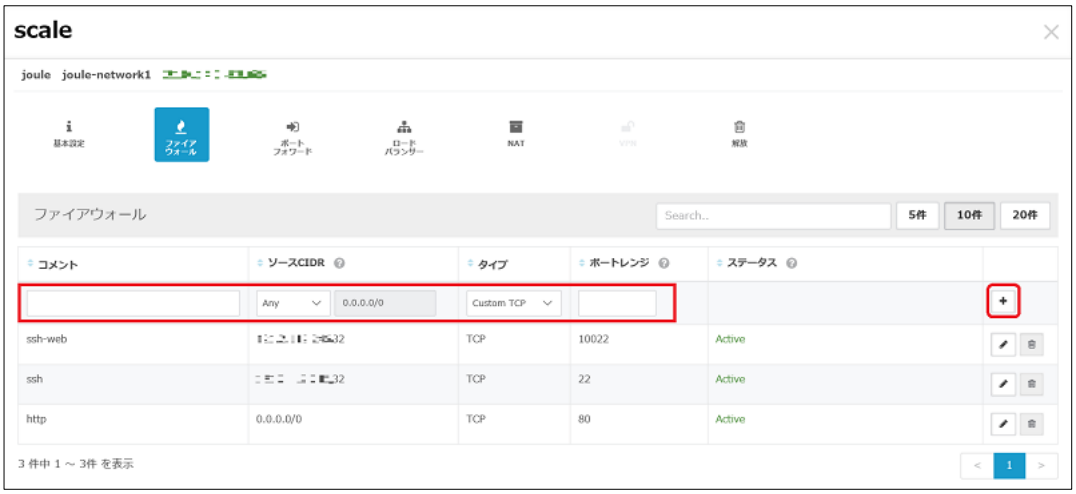
項目	設定内容
コメント	ssh-web
ソースCIDR	My IP
タイプ	Custom TCP
ポートレンジ	10022

・ssh

項目	設定内容
コメント	ssh
ソースCIDR	My IP
タイプ	SSH
ポートレンジ	自動

・http

項目	設定内容
コメント	http
ソースCIDR	Any（httpの設定）
タイプ	HTTP（httpの設定）
ポートレンジ	自動（httpの設定）



⑤ ポートフォワードをクリックし、ssh-webのsshポートフォワード設定を行います。監視対象サーバーは基本的に外部からログインする機会が殆どない為、22番ポートは運用サーバーで使用し、10022番ポートを監視対象サーバー宛てに設定します。本設定も同様に「＋」ボタンで追加します。

項目	設定内容
コメント	ssh-web
プライベートポート	22
パブリックポート	10022
プロトコル	TCP
仮想マシン	scale01

⑥ ロードバランサーに監視対象サーバーを追加します。「ロードバランサー」をクリックします。Webでサーバーの負荷分散として、今回はHTTPでアクセスを来た際の振り分け設定を実施し、scale01を予め追加しておきます。設定が完了したら「×」で設定ウィンドウを閉じます。

項目	設定内容
管理名	Scale（任意）
パブリックポート	HTTP
プライベートポート	80
アルゴリズム	Roundrobin
接続維持	なし
仮想マシン	scale01

⑦ 監視対象サーバーにログインします。活用マニュアルの「Webサイトの本番環境を構築したい（Web1台構成）」を参照して行います。 (3)仮想マシンへのアクセスを実施し、Webサーバーを作成します。アクセスの際にはsshのポートを10022と指定してログインしてください。（https://www.idcf.jp/cloud/pdf/manual_002.pdf）

⑧ 監視対象サーバーのテンプレートを作成します。活用マニュアルの「スケーラブルなWebサイトを構築したい（Web2台構成編）」を参照して行います。 (1)スナップショットの取得と(2)テンプレートの作成を実施してテンプレートの作成を行います。（http://www.idcf.jp/help/cloud/guide/pdf/manual_005.pdf）

以上で監視対象サーバーの作業は完了です。次の章でも一部で監視対象サーバーの作業がありますので、作業するサーバーを注意して進めて下さい。

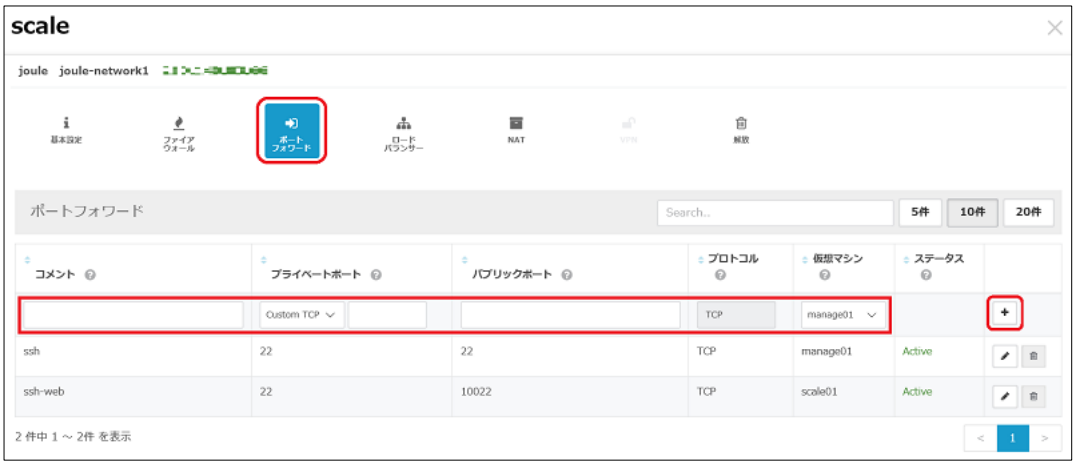
(2)運用サーバーの作成とスクリプトの設定

オートスケールのスクリプトを実行する運用サーバーを作成します。運用サーバーにはcloudstack-apiとIDCフロンティアで用意しているオートスケール用のサンプルプログラムを設置します。運用サーバーの作成は、「Webサイトの本番環境を構築したい（Web1台構成）」を参照して行います。以降の手順は「Webサイトの本番環境を構築したい（Web1台構成）」の(1)仮想マシンの作成まで完了した状態からの手順となります。尚、作成する際の仮想マシンのイメージは、おすすめtemplatesの「CentOS 6.6 64-bit」を選択してください。また、仮想マシン名指定の際には「manage01」として下さい。※必ず監視対象サーバーと同じゾーンに作成して下さい。
（https://www.idcf.jp/cloud/pdf/manual_002.pdf）

① コンピューティングのメニューから「IPアドレス」をクリックし、(1)で作成したIPアドレス名（ここではscale）をクリックします。

② 「ポートフォワード」をクリックしてmanage01にsshログインする為の情報を記載し、「＋」をクリックして設定を追加します。

項目	設定内容
コメント	ssh
プライベートポート	SSH
パブリックポート	22
プロトコル	自動
仮想マシン	manage01



③ 運用サーバーにログインします。活用マニュアルの「Webサイトの本番環境を構築したい（Web1台構成）」を参照して行います。
（https://www.idcf.jp/cloud/pdf/manual_002.pdf）

④ pythonを予めアップデートし、pipをインストールする為のスクリプトをダウンロードし、実行してインストールします。pipコマンドを実行して正常にインストールされているか確認します。

```
[root@manage01 ~]# yum -y update python
読み込んだプラグイン:fastestmirror
更新処理の設定をしています
～～～～～～～～～～～～～～～
依存性を更新しました:
  python-libs.x86_64 0:2.6.6-64.el6
完了しました!
[root@manage01 ~]# curl -kL https://bootstrap.pypa.io/get-pip.py | python
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 1476k  100 1476k    0     0   9.9M      0 --:--:-- --:--:-- --:--:-- 22.8M
～～～～～～～～
https://urllib3.readthedocs.org/en/latest/security.html#insecureplatformwarning.
[root@manage01 ~]# pip --version
pip 8.0.3 from /usr/lib/python2.6/site-packages (python 2.6)
[root@manage01 ~]#
```

⑤ cloudstack-apiをインストールする為、依存関係があるソフトウェアをインストールします。

```
[root@manage01 ~]# yum -y install libxml2 libxml2-devel libxslt libxslt-devel ¥
gcc python-devel
読み込んだプラグイン:fastestmirror
インストール処理の設定をしています
Loading mirror speeds from cached hostfile
～～～～～～～～～～～～～～～
依存性を更新しました:
  libgcc.x86_64                                                 0:4.4.7-16.el6
libgomp.x86_64 0:4.4.7-16.el6

完了しました!
[root@manage01 ~]#
```

⑥ cloudstack-apiをインストールして、インストール出来ているか確認します。

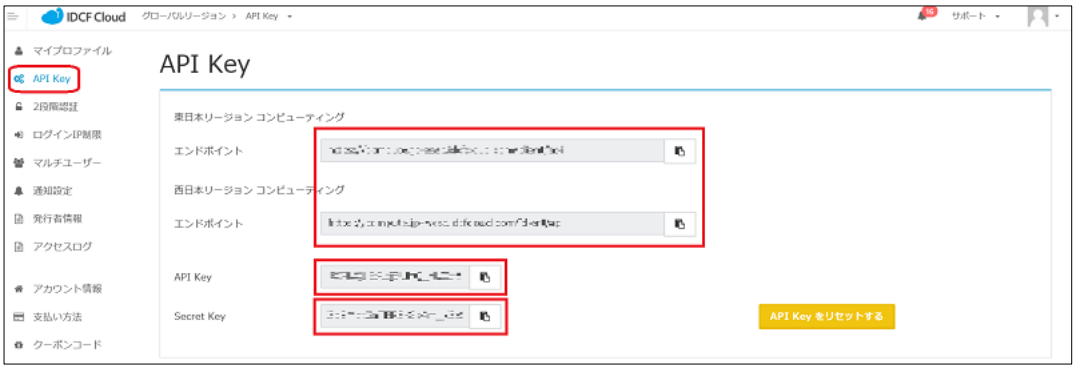
```
[root@manage01 ~]# pip install git+https://github.com/idcf/cloudstack-api
DEPRECATION: Python 2.6 is no longer supported by the Python core team, please upgrade
your Python. A future version of pip will drop support for Python 2.6
Collecting git+https://github.com/idcf/cloudstack-api
～～～～～～～～～～～～～～～
  Running setup.py install for cloudstack.compute ... done
Successfully installed cloudstack.compute-0.10.2 httplib2-0.9.2 lxml-3.5.0
parsedatetime-0.8.7 prettytable-0.5 simplejson-3.8.1
[root@manage01 ~]# cloudstack-api --version
cloudstack-api v0.10.2
[root@manage01 ~]#
```

⑦ IDCフロンティアで用意しているオートスケール用スクリプトをインストールします。

```
[root@manage01 ~]# yum -y localinstall¥
 http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm

読み込んだプラグイン:fastestmirror
ローカルパッケージ処理の設定をしています
～～～～～～～～～～
インストール:
  epel-release.noarch 0:6-8
完了しました!

[root@manage01 ~]# yum -y localinstall¥
 http://repo.cloud.idc.jp/Linux/CentOS/6/idc/x86_64/idcf-release-8-0.0.idcf.el6.noa
 rch.rpm

読み込んだプラグイン:fastestmirror
ローカルパッケージ処理の設定をしています
idcf-release-8-0.0.idcf.el6.noarch.rpm
| 4.2 kB     00:00
～～～～～～～～～～～～～～～
依存性関連をインストールしました:
  yum-plugin-priorities.noarch 0:1.1.30-30.el6
完了しました!

[root@manage01 ~]# yum -y install idcf.compute.auto_scale
読み込んだプラグイン:fastestmirror, priorities
インストール処理の設定をしています
～～～～～～～～～～～～～～～
  python-simplejson.x86_64 0:2.0.9-3.1.el6
完了しました!

 [root@manage01 ~]#
```

⑧ APIキーを取得するため、コントロールパネル右上の「アイコン」をクリックし、プルダウンメニューから「アカウント設定」をクリックします。

⑨ メニューから「API Key」を選択しAPI Key画面を表示させます。エンドポイント（今回サーバを構築しているリージョンによって異なります。）、API Key、SecretKeyをメモ帳等にコピーして次の項目で使用出来るようにしておきます。



⑩ 設定のサンプルファイルをコピーします。vi等のテキストエディタでコピーしたサンプルファイルを開き、⑨で確認した項目を記入します。

```
[root@manage01 ~]# cp /usr/share/doc/idcf.compute.auto_scale-*/sample.idcfrc .idcfrc
[root@manage01 ~]# vi .idcfrc
以下を編集して保存
----------------
[account]
host=[コピーしたエンドポイント]
api_key=[コピーしたAPI Key]
secret_key=[コピーしたSecret Key]
----------------

[root@manage01 ~]#
```

⑪ cloudstack-apiが正しく動作するかzone-idを取得して確認します。このzone-idは後に使用するので、今回サーバーを作成したzoneのidをメモ帳等にコピーしておいて下さい。
※尚、エラーになる場合は設定が間違っている、もしくはコントロールパネルにてAPI IPアドレス制限が掛かってる可能性がある為、設定を見直してください。

```
[root@manage01 ~]# cloudstack-api listZones -t id,name

+--------------------------------------+--------+
|                  id                  |  name  |
+--------------------------------------+--------+
| 9703cdbb-aee7-41ba-ba80-4807eaa68b80 | henry  |
| a117e75f-d02e-4074-806d-889c61261394 | tesla  |
| baf86a6e-4e3b-428e-8fd0-7fda43e468d4 | joule  |
| f0954b9b-2626-4549-82ad-ca421073b3bc | pascal |
+--------------------------------------+--------+
[root@manage01 ~]#
```

⑫ テンプレートidを取得します。このテンプレートidは後に使用するので、メモ帳等にコピーしておいて下さい。

```
[root@manage01 ~]# cloudstack-api listTemplates --templatefilter self -t id,name

+--------------------------------------+--------+
|                  id                  |  name  |
+--------------------------------------+--------+
| xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx | scale  |
+--------------------------------------+--------+
[root@manage01 ~]#
```

⑬ サーバを紐づけるLoad Balancer Rule ID(ここではscaleのid)を取得します。
このLoad Balancer Rule IDは後に使用するので、メモ帳等にコピーしておいて下さい。

```
[root@manage01 ~]# cloudstack-api listLoadBalancerRules -t id,name,publicip

+--------------------------------------+-------+---------------+
|                  id                  |  name |    publicip   |
+--------------------------------------+-------+---------------+
| xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx | scale | xxx.xxx.xx.xx |
+--------------------------------------+-------+---------------+
[root@manage01 ~]#
```

⑭ マシンタイプのid(ServiceOffering id)を取得します。今回はlight.S1のidを使用します。
このServiceOffering idは後に使用するので、メモ帳等にコピーしておいて下さい。

```
[root@manage01 ~]# cloudstack-api listServiceOfferings -t id,displaytext

+--------------------------------------+------------------------------------+
|                  id                  |            displaytext             |
+--------------------------------------+------------------------------------+
| 12e39b73-3ce6-4e57-9036-3dac0c2b2b06 | highmem.M16( 2CPU / 16GB RAM )     |
| 354c62e6-b99b-42f2-b5c7-e741f1085422 | standard.XL32( 8CPU / 32 GB RAM )  |
| 435c1aab-e796-42c7-9320-22ebdc8f50aa | highcpu.L8( 4CPU / 8GB RAM )       |
| 55621f17-4d38-457c-ba34-e6199701b67b | standard.S4( 1CPU / 4GB RAM )      |
| 6a99ff4c-1a24-4aa6-b4cc-600220987ed0 | standard.L16( 4CPU / 16GB RAM )    |
| 6fda5e0c-e64d-46ea-893d-7e2ac9e128e7 | highcpu.XL16 ( 8CPU / 16GB RAM )   |
| 7ae143a6-5662-4f1d-bc4c-10defa775bcb | standard.M8( 2CPU / 8GB RAM )      |
| 7c548831-427b-437c-9c8b-80dde8031303 | highcpu.2XL32( 16 CPU / 32GB RAM ) |
| 8cf15770-c3c8-4efc-8ae5-b8327790db76 | highcpu.M4( 2CPU / 4GB RAM )       |
| 95edba75-7cb5-4654-a2f5-f40b7acf7a57 | standard.S8( 1CPU / 8GB RAM )      |
| 9a2f3ee4-af46-4790-9331-753674c16e68 | highio.5XL128( 40CPU /128GB RAM)   |
| d1aac6d2-bb47-4106-90d0-6a73ac3ae78e | light.S2( 1CPU / 2GB RAM )         |
| d59817bc-ed79-4083-8b71-51b26c76d311 | highmem.L32( 4CPU / 32GB RAM )     |
| e01a9f32-55c4-4c0d-9b7c-d49a3ccfd3f6 | light.S1( 1CPU / 1GB RAM )         |
| ec13a7d8-26ce-4c4e-a223-7ef832bb1243 | light.S2( 1CPU / 2GB RAM )         |
| ee5ee568-76b2-46ad-9221-c695e6f2149d | highmem.XL64( 8CPU / 64GB RAM )    |
+--------------------------------------+------------------------------------+
[root@manage01 ~]#
```

⑮ 監視対象サーバーに監視スクリプトでログイン出来るように運用サーバーでsshの鍵交換をします。
※監視スクリプトでパスフレーズを入力する為、パスフレーズを設定して下さい。

```
[root@manage01 ~]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):  [Enter]
Enter passphrase (empty for no passphrase):  [パスフレーズを入力]
Enter same passphrase again:  [もう1度パスフレーズを入力]
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
～～～～～～～～～～～～～～～～～～～～
+-----------------+
[root@manage01 ~]#
```

⑯ ※※この工程は監視対象サーバーでの作業です※※

運用サーバーから監視対象サーバーに、一時的にパスワードでログイン出来るようにsshのパスワードログインを有効化してユーザーパスワードを設定します。

```
[root@scale01 ~]# sed -i".org" -e "s/PasswordAuthentication no//g" /etc/ssh/sshd_config
[root@scale01 ~]# /etc/init.d/sshd restart
sshd を停止中:                                             [  OK  ]
sshd を起動中:                                             [  OK  ]
[root@scale01 ~]# passwd
ユーザー root のパスワードを変更。
新しいパスワード: [パスワードを入力]
新しいパスワードを再入力してください: [再度パスワードを入力]
passwd: 全ての認証トークンが正しく更新できました。
[root@scale01 ~]#
```

⑰ ※※この工程は運用サーバーでの作業です※※

運用サーバーから監視対象サーバーに公開鍵を送ります。鍵認証でログイン出来る事を確認します。（scale01はホスト名です。変更している場合は読み替えて下さい。）

```
[root@manage01 ~]# cat /root/.ssh/id_rsa.pub | ssh root@scale01 ¥
"cat - >> ~/.ssh/authorized_keys"

The authenticity of host 'scale01 (10.15.0.101)' can't be established.
～～～～～～～～～～～
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'scale01,10.15.0.101' (RSA) to the list of known hosts.
root@scale01's password: [パスワードを入力]

[root@manage01 ~]# ssh scale01
Enter passphrase for key '/root/.ssh/id_rsa': [鍵のパスフレーズを入力]
～～～～～～～～～～～
[root@scale01 ~]# exit
logout
Connection to scale01 closed.
[root@manage01 ~]#
```

⑱ ※※この工程は監視対象サーバーでの作業です※※

監視対象サーバーのパスワードログインを無効化して元に戻します。

```
[root@scale01 ~]# mv -f /etc/ssh/sshd_config.org /etc/ssh/sshd_config
[root@scale01 ~]# /etc/init.d/sshd restart
sshd を停止中:                                             [  OK  ]
sshd を起動中:                                             [  OK  ]
[root@scale01 ~]#
```

⑲ ※※本章のこの工程以降は全て運用サーバーでの作業です※※
監視スクリプト（ssh_scale.py）で呼び出す設定ファイルをviで編集します。

```
[root@manage01 ~]# vi .idcfrc

----------------------------------------------------------------
[account]
host=[⑩で設定済み]
api_key=[⑩で設定済み]
secret_key=[⑩で設定済み]

[monitoring]
host=[監視対象VMの仮想マシン名(今回はscale01)]
user=root
private_key_path=/root/.ssh/id_rsa
pass_phrase=[パスフレーズ]

[launch_config]
serviceofferingid=[⑭で確認したServiceOffering ID]
templateid=[⑫で確認したテンプレートID]
zoneid=[⑪で確認したzone ID]

[scalling]
min_size=[オートスケール時の最小VM台数(ここでは2)]
max_size=[オートスケール時の最大VM台数(ここでは4)]
load_balancer=[⑬で確認したLoad Balancer Rule ID]

[scalling_policy_out]
threshold=[スケールアウトを開始するLoad Average(ここでは2)]

[scalling_policy_in]
threshold=[スケールインを開始するLoad Average(ここでは1)]
----------------------------------------------------------------

[root@manage01 ~]#
```

⑳ 実際にssh_scale.pyを実行してテストします。現在はscale01のみがロードバランサー配下で稼働していますが、⑲でmin_sizeを2に指定している為、1台追加されます。コントロールパネルでも「VM-」から始まるサーバーが1台追加されている事を確認します。

```
[root@manage01 ~]# ssh_scale.py
2016-01-27 13:42:41,471 transport[1379] [INFO]: Connected (version 2.0, client OpenSSH_5.3)
2016-01-27 13:42:41,565 transport[1379] [INFO]: Authentication (publickey) successful!
～～～～～～～～～～～～～～～～～～～～
2016-01-27 13:44:01 *** scale in check ***
2016-01-27 13:44:01 la_threshold_in  : 1
2016-01-27 13:44:01 now_la           : 0
2016-01-27 13:44:01 now_vm_count     : 2
[root@manage01 ~]#
```

21 自動監視スクリプトを起動する為に、運用サーバーのcronにて自動実行するように設定します。

```
[root@manage01 ~]# (crontab -l; echo ¥
"* * * * * /usr/bin/ssh_scale.py >> /var/log/ssh_scale.log 2>&1") | crontab -
crontab: installing new crontab（crontab登録の初回のみ出るメッセージ）
[root@manage01 ~]# crontab -l
* * * * * /usr/bin/ssh_scale.py >> /var/log/ssh_scale.log 2>&1
[root@manage01 ~]#
```

以上で運用サーバーの設定は完了です。次に実際にスケールアウトを自動で行うかテストをします。

(3)スケールアウトのテスト

実際にスケールアウトするかテストで負荷をかけて確認をします。本章では運用サーバー、監視対象サーバー、双方で作業をするので作業するサーバーに注意してください。また、運用サーバーと監視対象サーバーは、それぞれ別々のssh接続で作業を行ってください。

① ※※この工程は監視対象サーバーでの作業です※※

監視対象サーバーで負荷をかける為にストレステストツールを導入します。
※(1)にてAppTemplateを使用してサーバーを作成していない場合、(2)⑦を参考にしてepel-releaseをインストールして下さい。

```
[root@scale01 ~]# yum -y install stress
Loaded plugins: fastestmirror, priorities
Loading mirror speeds from cached hostfile
 * base: www.ftp.ne.jp
 * epel: mirrors.hustunique.com
 * extras: www.ftp.ne.jp
～～～～～～～～～～
Installed:
  stress.x86_64 0:1.0.4-4.el6
Complete!
[root@scale01 ~]#
```

② ※※この工程は監視対象サーバーでの作業です※※

ストレステストツールを実行します。

```
[root@scale01 ~]# stress -c 4 > /dev/null &
[1] 3154　（pidの為、必ずしも同じにはなりません。）
[root@scale01 ~]#
```

③ ※※この工程は監視対象サーバーでの作業です※※

何回かwコマンドを実行してLoadAverageが閾値(ここでは2)を超える事を確認します。

```
[root@scale01 ~]# w
 14:22:02 up 10:37,  3 users,  load average: 0.43, 0.63, 0.44
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
～～～～～～～～～～～～～～～
[root@scale01 ~]# w
 14:22:17 up 10:37,  3 users,  load average: 1.22, 0.80, 0.49
～～～～～～～～～～～～～～～
[root@scale01 ~]# w
 14:22:39 up 10:38,  3 users,  load average: 2.01, 1.00, 0.57
～～～～～～～～～～～～～～～
[root@scale01 ~]#
```

④ ※※この工程は監視対象サーバーでの作業です※※
tailコマンドでssh_scale.pyの実行ログを確認し、サーバを増やす挙動がある事を確認します。（出ていない場合は、1分程度待って再度実行します。）

```
[root@manage01 ~]# tail /var/log/ssh_scale.log
2016-01-27 14:23:10 query job         : jobid: 2dd58c04-5666-481e-b756-d3a89ed61614,
jobstatus: 0
2016-01-27 14:23:20 query job         : jobid: 2dd58c04-5666-481e-b756-d3a89ed61614,
jobstatus: 0
2016-01-27 14:23:30 query job         : jobid: 2dd58c04-5666-481e-b756-d3a89ed61614,
jobstatus: 0
2016-01-27 14:23:40 query job         : jobid: 2dd58c04-5666-481e-b756-d3a89ed61614,
jobstatus: 0
2016-01-27 14:23:50 query job         : jobid: 2dd58c04-5666-481e-b756-d3a89ed61614,
jobstatus: 1
2016-01-27 14:23:50 now vm count    : 3
2016-01-27 14:23:50 *** scale in check ***
2016-01-27 14:23:50 la_threshold_in  : 1
2016-01-27 14:23:50 now_la           : 3
2016-01-27 14:23:51 now_vm_count     : 3
[root@manage01 ~]#
```

⑤ (2) 運用サーバーの作成とスクリプトの設定　の⑳を参考にしてコントロールパネルにVM-から始まるサーバーが増えている事を確認します。

⑥ ※※この工程は監視対象サーバーでの作業です※※
ストレステストツールを終了して負荷を下げます。

```
[root@scale01 ~]# killall stress
[1]+  終了しました      stress -c 4 > /dev/null
[root@scale01 ~]#
```

⑦ ※※この工程は監視対象サーバーでの作業です※※
数分置いてからwコマンドを実行してLoadAverageが閾値（ここでは1）以下に下がった事を確認します。

```
[root@scale01 ~]# w
14:38:28 up 10:54,  3 users,  load average: 0.20, 2.06, 2.04
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
～～～～～～～～～～～～～～～
[root@scale01 ~]#
```

⑧ 対象サーバーが削除されて、ステータスが「Running」のVM-から始まるサーバーが1台しか無い事を確認します。（ステータスがDestroyedであれば削除されていますがstopping等の状態の場合は、数分待ってからブラウザを更新して下さい。）

⑪ ※※この工程は監視対象サーバーでの作業です※※
tailコマンドでssh_scale.pyの実行ログを確認し、正常に稼働している事を確認します。実行時間が1分間の間に発生していて、now_vm_countが設定したmin_sizeと同じ事を確認します。

```
[root@manage01 ~]# tail /var/log/ssh_scale.log
2016-01-27 15:18:01,888 transport[1379] [INFO]: Secsh channel 1 opened.
2016-01-27 15:18:02,018 transport[1379] [INFO]: Secsh channel 2 opened.
2016-01-27 15:18:01 *** scale out check ***
2016-01-27 15:18:01 la_threshold_out : 2
2016-01-27 15:18:01 now_la           : 0
2016-01-27 15:18:02 now_vm_count     : 2
2016-01-27 15:18:02 *** scale in check ***
2016-01-27 15:18:02 la_threshold_in  : 1
2016-01-27 15:18:02 now_la           : 0
2016-01-27 15:18:02 now_vm_count     : 2
[root@manage01 ~]#
```

これでWebサーバーを自動で追加して負荷分散するオートスケールの設定ができるようになりました。

本書では新しくIPアドレスを追加してロードバランサーの設定を行いましたが、ゾーンにデフォルトで払い出されるIPアドレスでも実施可能です。新しくIPアドレスを追加したくない、もしくは簡単に試してみたいといった場合は、ゾーンにデフォルトで払い出されるIPアドレスを使用する事をおすすめ致します。

オートスケールの機能を使う事により夜間休日問わず、不測の急激なアクセスに対応をし、Webサーバーのレスポンス速度を負荷分散によって平準化したり、耐障害性を高める事が自動で出来るようになります。今回はIDCフロンティアにて用意をしたサンプルスクリプトにて動作確認を行いましたが、APIは公開されておりますので同じ仕組みで自作スクリプトを作成し、自由に柔軟なスケーリングを構築する事も出来ます。APIのコントロールパネルには実行IPアドレスを制限するセキュリティ強化機能も付属していますので、合わせて使用する事をおすすめ致します。

本書の設定については以下URLを元に作成されております。
こちらもご覧下さい。

IDCFドキュメント「Getting Started」
http://docs.idcf.jp/cloud/introduction/

以上