Red Hat OpenStack 3.0 運用検証報告書

(VM レンタルサービス編)

Version 1.0

2014 年 1 月

目　次

略号の説明

略　号	意　味
RHEL6	Red Hat Enterprise Linux 6 (64-bit x86_64) RHEL6 全体を示す。
RHEL6.4	Red Hat Enterprise Linux 6.4 (64-bit x86_64) RHEL6 の中でも RHEL6.4 に限定して示す。
RHOS	Red Hat OpenStack 3.0
S/W	Software

1. はじめに

本書は、Red Hat が提供する Red Hat OpenStack3.0(以下、RHOS)を用いて、社内向け VM レンタルサービスを想定した運用の実現性を検証した内容を示すものである。RHOS の運用上の実用性を明らかにすると共に、運用上の課題や対応策、今後の展望について述べる。

2. 想定運用イメージ

本検証では、最大 100VM 程度を稼働させる社内向け VM レンタルサービスを想定した。
サービスの想定運用イメージを「図 2-1」に示す。

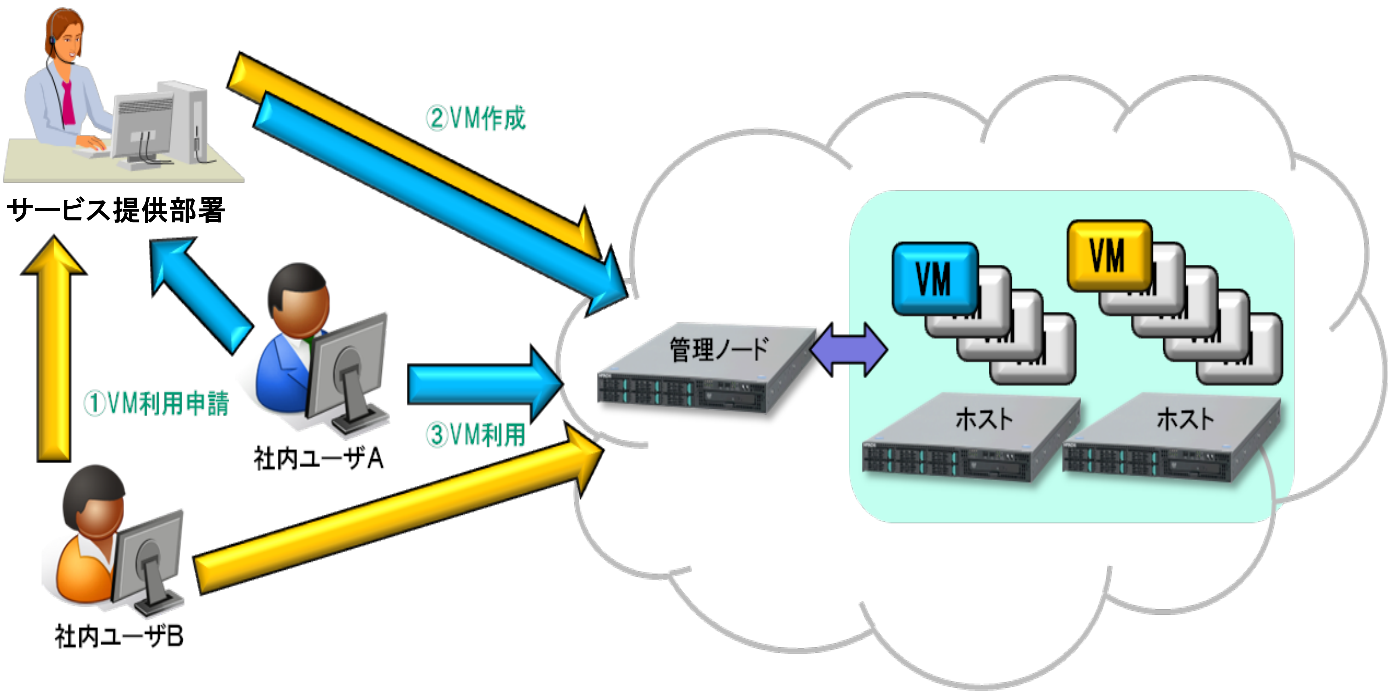

図 2-1 社内向け VM レンタルサービスのイメージ

想定運用は以下の通りである。
① 社内ユーザから VM の利用申請を受付(VM のリソースや OS は一定範囲内からの選択)
② RHOS 環境上に VM を作成
③ 社内ユーザに作成した VM を提供

3. 検証環境

本検証を実施した環境を「図 3-1」に示す。

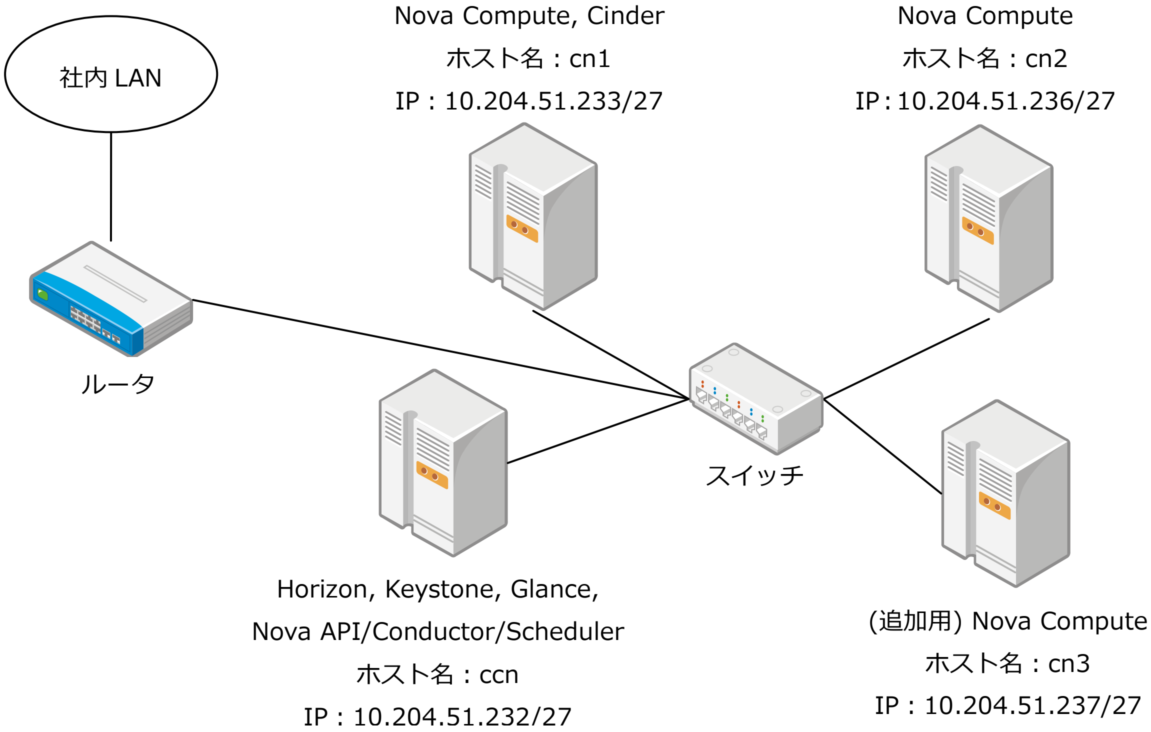

図 3-1 検証環境

Horizon/Keyston/Glance/Nova/Cinder は、RHOS の各サービスのコードネームを示す。Nova API、Nova Conductor、Nova Scheduler、Nova Compute は、Nova の各機能を示す。これらの詳細については、「7.参考文献」の[1]を参照。

なお、cn3 は「5.3 ホストの動的追加」の検証だけで使用する。

各ホストに搭載されたリソースを以下の「表 3-1」に示す。

表 3-1 使用機材

機材	仕様	OS
Dell PowerEdge C6220 	CPU：Intel Xeon E5-2620 2.00GHz, 2CPU*6Core (HyperThreading ON、OS 上は 24Core)	Red Hat Enterprise Linux 6.4
	MEM：64GB	
	DISK：2TB SATA	

今回の検証では、デルの高密度クラウドサーバ PowerEdge C6220 を利用した。本サーバは電源、ファンを搭載したシャーシ(2U)に、4 台の 2 ソケットサーバを搭載できるモデルである。

また、利用した RHOS のバージョンは RHOS3.0(Grizzly)である。

4. 検証項目

本検証では、社内向け VM レンタルサービスの運用に必要となる以下の項目を確認する。

(1) VM の一斉作成/起動、一斉削除の実行

社内ユーザから同時に多数の VM の利用を申請された場合は、一度に多数の VM を作成/起動する可能性がある。また、月単位で VM の利用を管理している場合など、利用が終了した VM を一度に多数削除する運用が考えられる。このように、一度に多数の VM に対するアクションを実行した場合に、正常に処理できることを確認する。

(2) 運用に必要なリソース情報の参照

社内向け VM レンタルサービスでは、各 VM のリソースを固定メニューから選ばせるため、主に VM に割り当てた仮想 CPU 数やメモリサイズなどが参照できればよく、実際に使用している CPU 時間やメモリサイズの管理までは不要である。このように、最低限必要な情報が Horizon(Dashboard)から参照できるかを確認する。また、そもそも Dashboard で参照可能なリソース情報は何かも確認する。

(3) ホスト(計算機ノード)の動的追加

利用ユーザが増加するに伴い、ホストの CPU やメモリが不足することが考えられる。その場合は、ホストを動的に追加することが必要となる。このような運用が可能であることを確認する。

(4) ユーザごとの VM へのアクセス制御

各ユーザに Dashboard の利用を許可する場合、ユーザには自分が利用する以外の VM を参照・操作させてはいけない。このような運用が可能であることを確認する。

(5) VM の CPU 使用量、およびディスク I/O 帯域の制御

VM には仮想 CPU の個数を割り当てるが、オーバーコミット状態の場合は、1 つの VM が他の VM の CPU リソースを圧迫する可能性がある。このため、VM ごとに CPU 使用量を制御できることが望ましい。また、同様にディスク I/O 帯域も制御できることが望ましい。このため、これらの制御ができることを確認する。

なお、本検証ではネットワーク機器の関係上、ネットワーク関連の検証は実施していない。

5. 検証結果

5.1 VM の一斉作成/起動/削除

VM の一斉作成/起動と一斉削除の検証としては、以下の 6 つを実施した。元々は検証 1 および検証 4 のみを実施する予定であったが、それらで問題が生じたため、検証 2,3,5,6 を追加で実施した。

検証 1：Dashboard から 100VM を 1 操作で作成/起動
検証 2：Dashboard から 10VM を 1 操作で作成/起動
検証 3：nova コマンドで 100VM をシリアルに連続作成/起動
検証 4：Dashboard から 100VM を 1 操作で削除
検証 5：Dashboard から 10VM を 1 操作で削除
検証 6：nova コマンドで 100VM をシリアルに連続削除

検証準備の手順と各検証の手順および結果を以下に示す。なお、Dashboard 上では、VM は"インスタンス"と表現されている。

■検証準備：検証用のプロジェクトとインスタンスタイプを作成
●手順
(1) Dashboard に admin でログイン
(2) Dashboard の[管理]-[プロジェクト]メニューをクリック
(3) [プロジェクト]画面で[プロジェクトの作成]ボタンをクリック
(4) [プロジェクトの作成]画面で以下の項目を入力し、[プロジェクトの作成]ボタンをクリック
  [プロジェクトの情報]タブ
    名前：vm100
  [プロジェクトのメンバー]タブ
    プロジェクトノメンバー：admin
  [クォータ]タブ
    仮想 CPU：120 (1 個/VM で 100 以上を指定)
    インスタンス：120 (100 以上を指定)
    メモリ(MB)：120000 (1024MB/VM で 102400 以上を指定)
    Floating IP：120 (1IP/VM で 100 以上を指定)
  (上記指定値以外はデフォルト値)
(5) [管理]-[インスタンスタイプ]メニューをクリック
(6) [インスタンスタイプ]画面で[インスタンスタイプの作成]ボタンをクリック
(7) [インスタンスタイプの作成]画面で以下の項目を入力し、[インスタンスタイプの作成]ボタンをクリック
    名前：1cpu.1gbmem.0disk
    仮想 CPU：1
    メモリ MB：1024
    ルートディスク GB：0
    一時ディスク GB：0
    スワップディスク MB：0

■検証 1：Dashboard から 100VM を 1 操作で作成/起動

●手順

(1) Dashboard に admin でログイン
(2) Dashboard の[プロジェクト]タブをクリック
(3) [プロジェクト]メニューの[現在のプロジェクト]をクリックし、表示されたリストの vm100 をクリック
(4) [プロジェクト]-[インスタンス]をクリック
(5) [インスタンス]画面の[インスタンスの起動]ボタンをクリック
(6) [インスタンスの起動]画面で以下の項目を入力し、[起動]ボタンをクリック

[詳細]タブ

インスタンス・ソース：イメージ
イメージ：RHEL64x86_64[1]
インスタンス名：vmtest
インスタンスタイプ：1cpu.1gbmem.0disk
インスタンス数：100 (100VM 同時作成を指定)

インスタンス名に"-<UUID>"が付加された名前(例：vmtest-2ddbb421-7f60-4a29-b0a3-ffff334d0e13)のインスタンスが一斉に作成される。[インスタンス]画面の[状態]列がすべて"Active"または"Error"となるまで待ち、前者を成功数、後者を失敗数としてカウント。それから次の手順を実行。

(7) [インスタンス]画面の[インスタンス名]の左横のチェックボックスをチェックし、すべてのインスタンスが選択された状態で[インスタンスの削除]ボタンをクリック
(8) [Confirm インスタンスの削除]画面で[インスタンスの削除]ボタンをクリック
(9) [状態]列が"Error"となり、インスタンスが削除されない場合は、すべてのインスタンスが削除されるまで繰り返し手順(7)(8)を実行
(10) 手順(5)～(9)を合計 3 回実行

●結果

Dashboard から 100VM を 1 操作で作成/起動した結果を「表 5-1」に示す。

表 5-1 Dashboard 100VM 作成

	成功数	失敗数
1 回目	31	69
2 回目	20	80
3 回目	29	71
平均	26.7	73.3

上記の通り、Dashboard から 100VM を 1 操作で作成/起動すると、その多くが失敗してしまう結果となった(「表 6-1 問題点一覧」「項番 1」)。

[1] https://rhn.redhat.com/rhn/software/channel/downloads/Download.do?cid=16952 からダウンロードした RHEL6.4(x86_64)のイメージ。イメージの作成手順については、「7.参考文献」の[1]を参照。

成功数が最低でも 20 であったことから、10VM 程度ならすべて成功すると考え、追加で 10VM を 1 操作で作成/起動する検証(検証 2)を実施した。

また、novaコマンド[2]でVMを作成/起動することで代替策となるかを確認するため、novaコマンドで 100VMをシリアルに連続作成/起動する検証(検証 3)を実施した。

また、VM の作成/起動や削除に失敗する問題の発生に伴い、未割り当ての Floating IP が割り当て不可となる問題(「表 6-1 問題点一覧」「項番 2」)が発生した。詳細については、後述の「参考」の「(2)」参照。

(1) VM 数カウント方法

作成/起動に成功した VM 数、および失敗した VM 数は、以下のコマンドでカウントした。なお、行頭の"ccn#"は、ホスト ccn の root ユーザのプロンプトを表す。

成功数：

```
ccn# nova --os-username admin --os-password admina --os-tenant-name vm100 --os-auth-url
http://127.0.0.1:5000/v2.0 list | grep ACTIVE | wc -l
```

失敗数：

```
ccn# nova --os-username admin --os-password admina --os-tenant-name vm100 --os-auth-url
http://127.0.0.1:5000/v2.0 list | grep ERROR | wc -l
```

(2) 未割り当ての Floating IP が割り当て不可となる現象について

[現象]

未割り当ての Floating IP をインスタンスに割り当てることができない。

[原因]

インスタンスの作成/起動または削除の処理がエラーとなることにより、Floating IP が不正な割り当て状態となることが原因である。

[備考]

Floating IP の割り当て状況は、Dashboard の[インスタンス]画面で確認できるが、インスタンスの作成や削除に失敗した場合は、[インスタンス]画面には表示されない割り当て済み状態の Floating IP ができてしまうことがある。

このような不正な割り当て状態となった Floating IP は、正常な未割り当て状態に戻すまで、インスタンスに割り当てることができない。

不正な割り当て状態となったFloating IPは、ホストccn上でnova-manage floating listコマンド[3]を実行することで確認できる。正常にインスタンスに割り当てられたFloating IP、不正な割り当て状態となったFloating IP、正常な未割り当て状態のFloating IPの出力例を以下に示す。

[2] nova コマンドの詳細については、man ページ nova(1)参照。

[3] nova-manage コマンドの詳細は、man ページ nova-manage(1)参照。

正常にインスタンスに割り当てられた Floating IP：

```
1cf5610a1f3646f090d7a4c80c304ec9          192.168.2.230
b9200b45-eea6-4038-b239-a41b27cb7218     nova     eth1
```

不正な割り当て状態となった Floating IP：

```
1cf5610a1f3646f090d7a4c80c304ec9          192.168.2.169    None     nova     eth1
```

正常な未割り当て状態の Floating IP：

```
None     192.168.2.14     None     nova     eth1
```

出力は、左からプロジェクト ID、Floating IP、割り当て先インスタンス ID である。
不正な割り当て状態では、Floating IP がインスタンスに割り当てられていないことがわかる。

[対処方法]

nova floating-ip-delete コマンドを実行し、不正な割り当て状態の Floating IP を正常な未割り当て状態に戻せばよい。コマンドの実行例を以下に示す。

```
ccn# nova --os-username admin --os-password admina --os-tenant-name
1cf5610a1f3646f090d7a4c80c304ec9 --os-auth-url http://127.0.0.1:5000/v2.0 floating-ip-delete
192.168.2.169
```

なお、該当プロジェクト自体を削除してしまうと、上記コマンドでも Floating IP を未割り当て状態に戻せなくなるため注意すること。

■検証 2：Dashboard から 10VM を 1 操作で作成/起動

●手順

(1) 検証 1 の手順(1)～(5)を実行

(2) [インスタンスの起動]画面で以下の項目を入力し、[起動]ボタンをクリック

[詳細]タブ

インスタンス・ソース：イメージ
イメージ：RHEL64x86_64
インスタンス名：vmtest
インスタンスタイプ：1cpu.1gbmem.0disk
インスタンス数：10 (10VM 同時作成を指定)

インスタンス名に"-<UUID>"が付加された名前(例：vmtest-2ddbb421-7f60-4a29-b0a3-ffff334d0e13)のインスタンスが一斉に作成される。[インスタンス]画面の[状態]列がすべて"Active"または"Error"となるまで待ち、前者を成功数、後者を失敗数としてカウント。それから次の手順を実行。

(3) 手順(1)(2)を合計 3 回実行

●結果

Dashboard から 10VM を 1 操作で作成/起動した結果を「表 5-2」に示す。

表 5-2 Dashboard 10VM 作成

	成功数	失敗数
1 回目	10	0
2 回目	10	0
3 回目	10	0
平均	10	0

上記の通り、Dashboard から 10VM を 1 操作で作成/起動した場合は、すべて成功した。このため、多数の VM を一度に作成/起動する必要がある場合は、10VM 程度の単位に分割して実行すればよいと思われる。ただし、100VM での失敗の原因が判明していないため、この結果はあくまで参考情報である。

■検証 3：nova コマンドで 100VM をシリアルに連続作成/起動

●手順

(1) 以下の内容でスクリプトを作成し、ホスト ccn 上で実行

```
for i in $(seq -w 1 100); do
    nova --os-username admin --os-password admina --os-tenant-name vm100 ¥
        --os-auth-url http://127.0.0.1:5000/v2.0 ¥
        boot --flavor 1cpu.1gbmem.0disk --image RHEL64x86_64 test${i}
done
```

Dashboard の[インスタンス]画面の[状態]列がすべて"Active"または"Error"となるまで待ち、前者を成功数、後者を失敗数としてカウント。それから次の手順を実行。

(2) 検証 1 の手順(7)～(9)を実行

(3) 手順(1)(2)を合計 3 回実行

●結果

nova コマンドで 100VM をシリアルに連続作成/起動した結果を「表 5-3」に示す。

表 5-3 nova コマンド 100VM 作成

	成功数	失敗数
1 回目	100	0
2 回目	100	0
3 回目	100	0
平均	100	0

上記の通り、nova コマンドで 100VM をシリアルに連続作成/起動した場合は、すべて成功した。このため、多数の VM を一度に作成/起動する必要がある場合は、nova コマンドを使用するスクリプトを作成し、実行すればよいことが分かった。

■検証 4：Dashboard から 100VM を 1 操作で削除

●手順

(1) 検証 3 の手順(1)を実行

(2) 検証 1 の手順(1)～(3)を実行

(3) [インスタンス]画面の[インスタンス名]の左横のチェックボックスをチェックし、すべてのインスタンスが選択された状態で[インスタンスの削除]ボタンをクリック

[インスタンス]画面に表示されているインスタンスがすべて無くなるか、表示されているインスタンスの[状態]列がすべて"Error"となるまで待ち、"Error"となったインスタンスの数を失敗数としてカウント。それから次の手順を実行。

(4) 検証 1 の手順(7)～(9)を実行

(5) 手順(1)～(4)を合計 3 回実行

●結果

Dashboard から 100VM を 1 操作で削除した結果を「表 5-4」に示す。

表 5-4 Dashboard 100VM 削除

	成功数	失敗数
1 回目	22	78
2 回目	31	69
3 回目	16	84
平均	23	77

上記の通り、Dashboard から 100VM を 1 操作で削除すると、その多くが失敗してしまう結果となった(「表 6-1 問題点一覧」「項番 3」)。

成功数が最低でも 16 であったことから、10VM 程度ならすべて成功すると考え、追加で 10VM を 1 操作で削除する検証(検証 5)を実施した。

また、nova コマンドで VM を削除することで代替策となるかを確認するため、nova コマンドで 100VM をシリアルに連続削除する検証(検証 6)を実施した。

■検証 5：Dashboard から 10VM を 1 操作で削除

●手順

(1) 検証 2 の手順(1)(2)を実行

(2) 検証 4 の手順(2)～(4)を実行

(3) 手順(1)(2)を合計 3 回実行

●結果

Dashboard から 10VM を 1 操作で削除した結果を「表 5-5」に示す。

表 5-5 Dashboard 10VM 削除

	成功数	失敗数
1 回目	10	0
2 回目	10	0
3 回目	10	0
平均	10	0

上記の通り、Dashboard から 10VM を 1 操作で削除した場合は、すべて成功した。このため、多数の VM を一度に削除する必要がある場合は、10VM 程度の単位に分割して実行すればよいこと思われる。ただし、100VM での失敗の原因が判明していないため、この結果はあくまで参考情報である。

■検証 6：nova コマンドで 100VM をシリアルに連続削除

●手順

(1) 検証 3 の手順(1)を実行

(2) 以下の内容でスクリプトを作成し、ホスト ccn 上で実行

```
for i in $(seq -w 1 100); do
    nova --os-username admin --os-password admina --os-tenant-name vm100 ¥
        --os-auth-url http://127.0.0.1:5000/v2.0 delete test${i}
done
```

[インスタンス]画面に表示されているインスタンスがすべて無くなるか、表示されているインスタンスの[状態]列がすべて"Error"となるまで待ち、"Error"となったインスタンスの数を失敗数としてカウント。それから次の手順を実行。

(3) 手順(1)(2)を合計 3 回実行

●結果

nova コマンドで 100VM をシリアルに連続削除した結果を「表 5-6」に示す。

表 5-6 nova コマンド 100VM 削除

	成功数	失敗数
1 回目	100	0
2 回目	100	0
3 回目	100	0
平均	100	0

上記の通り、nova コマンドで 100VM をシリアルに連続削除した場合は、すべて成功した。このため、多数の VM を一度に削除する必要がある場合は、nova コマンドを使用するスクリプトを作成し、実行すればよいことが分かった。

5.2 リソース情報

リソース情報の検証としては、Dashboard でどのようなリソースが参照できるか、および社内向け VM レンタルサービスの運用で必要となる各種リソースの割り当て状況を参照できるかを確認した。

Dashboard の[管理]タブで参照できるリソースの一覧を「表 5-7」に、[プロジェクト]タブで参照できるリソースの一覧を「表 5-8」に示す。

表 5-7 Dashboard の[管理]タブで参照できるリソース一覧

項番	画面	表示名	内容
1	[概要][4]	稼働中のインスタンス	インスタンス数。一時停止(Paused)および休止(Suspended)状態のインスタンスも含む。
2		使用中のメモリ	インスタンスに割り当てられたメモリサイズ。
3		今月の仮想 CPU 時間	選択月に全インスタンスが使用したホストのCPU時間の合計[5]
4		今月の GB 時間	選択月の全インスタンスのディスクI/Oの合計
5		[使用状況] 仮想 CPU	各プロジェクトのインスタンスに割り当てられた仮想 CPU の合計数。
6		[使用状況] ディスク	各プロジェクトのインスタンスに割り当てられたルートディスクと一時ディスクの合計サイズ。
7		[使用状況] メモリ	各プロジェクトのインスタンスに割り当てられたメモリの合計サイズ。
8		[使用状況] 仮想 CPU 時間	各プロジェクトのインスタンスが使用したホストのCPU時間
9		[使用状況] ディスク GB 時間	各プロジェクトのインスタンスのディスクI/O
10	[インスタンス]	IP アドレス	各インスタンスに割り当てられた IP アドレス。Fixed IP/Floating IP の区別は無く、両方が同じ列に表示される。
11		サイズ	各インスタンスに割り当てられたメモリサイズ、仮想 CPU 数、ルートディスクサイズ。
12	[ボリューム]	サイズ	ボリューム(ブロックストレージデバイス)のサイズ

[4] 月単位で使用状況の表示を切り替えられる。また、csv 形式でデータを出力することができる。
[5] 本項目は表示名から内容を想定しているが、詳細については未調査である。

表 5-8 Dashboard の[プロジェクト]タブで参照できるリソース一覧

項番	画面	表示名	内容
1	[概要][6]	[クォータ概要] 使用済み <数字> / <数字> 利用可能なインスタンス	プロジェクトが利用可能なインスタンス数と作成済みのインスタンス数。
2		[クォータ概要] 使用済み <数字> / <数字> 利用可能な仮想 CPU	プロジェクトが利用可能な仮想 CPU 数と割り当て済みの仮想 CPU 数。
3		[クォータ概要] 使用済み <サイズ> / <サイズ> 利用可能なメモリ	プロジェクトが利用可能なメモリサイズと割り当て済みのメモリサイズ。
4		[クォータ概要] 使用済み <数字> / <数字> 利用可能なボリューム	プロジェクトが利用可能なボリューム数と作成済みのボリューム数。
5		[クォータ概要] 使用済み <サイズ> / <サイズ> 利用可能なボリュームストレージ	プロジェクトが利用可能なボリュームサイズと作成済みボリュームサイズ。
6		稼働中のインスタンス	インスタンス数。一時停止(Paused)および休止(Suspended)状態のインスタンスも含む。
7		使用中のメモリ	インスタンスに割り当てられたメモリサイズ。
8		今月の仮想 CPU 時間	選択月に全インスタンスが使用したホストのCPU時間の合計
9		今月の GB 時間	選択月の全インスタンスのディスクI/Oの合計
10		[使用状況]　仮想 CPU	各インスタンスに割り当てられた仮想 CPU 数。
11		[使用状況]　ディスク	各インスタンスに割り当てられたルートディスクと一時ディスクの合計サイズ。
12		[使用状況]　メモリ	各インスタンスに割り当てられたメモリサイズ。
13		[使用状況]　稼働時間	インスタンスが作成されてからの経過時間。つまり、一時停止(Paused)および休止(Suspended)状態の時間も含まれる。
14	[インスタンス]	IP アドレス	各インスタンスに割り当てられた IP アドレス。Fixed IP/Floating IP の区別は無く、両方が同じ列に表示される。
15		サイズ	各インスタンスに割り当てられたメモリサイズ、仮想 CPU 数、ルートディスクサイズ。
16	[ボリューム]	サイズ	ボリューム(ブロックストレージデバイス)のサイズ。
17	[コンテナー]	－	オブジェクトストレージにアップロードされた各ファイルのサイズ。

[6] 月単位で使用状況の表示を切り替えられる。また、csv 形式でデータを出力することができる。

社内向けVMレンタルサービスの運用で必要と考えられるリソース情報と、それがDashboardで参照できるかの確認結果を「表 5-8」に示す。

表 5-9 社内向け VM レンタルサービスの運用に必要なリソース情報

項番	リソース種別	リソース情報	Dashboard 参照可否
1	仮想 CPU	Nova Compute ホストが提供する仮想 CPU 数	×
2	仮想 CPU	Nova Compute ホストの割り当て済み仮想 CPU 数または未割り当て仮想 CPU 数	×
3	仮想 CPU	各プロジェクトの最大利用仮想 CPU 数 (各プロジェクトが利用可能な仮想 CPU 数)	○
4	仮想 CPU	各プロジェクトの割り当て済み仮想 CPU 数または未割り当て仮想 CPU 数	○
5	メモリ	Nova Compute ホストが提供するメモリサイズ	×
6	メモリ	Nova Compute ホストの割り当て済みメモリサイズまたは未割り当てメモリサイズ	×
7	メモリ	各プロジェクトの最大利用メモリサイズ (各プロジェクトが利用可能なメモリサイズ)	○
8	メモリ	各プロジェクトに割り当て済みのメモリサイズまたは未割り当てメモリサイズ	○
9	VM	各プロジェクトが利用可能なインスタンス数	○
10	VM	各プロジェクトが稼働させているインスタンス数または追加で稼働可能なインスタンス数	○
11	一時ディスク	Nova Compute ホストが提供する一時ディスク(ルートディスク/一時ディスク/スワップディスク)サイズ	×
12	一時ディスク	Nova Compute ホストの一時ディスクの使用済みサイズまたは空きサイズ	×
13	一時ディスク	各プロジェクトの最大利用一時ディスクサイズ (各プロジェクトが利用可能な一時ディスクサイズ)	×
14	一時ディスク	各プロジェクトの一時ディスクの使用済み(割り当て済み)サイズまたは空き(未割り当て)サイズ	×
15	ボリューム	各プロジェクトが利用可能なボリュームサイズ	○
16	ボリューム	各プロジェクトが使用済み(割り当て済み)のボリュームサイズ	○

●結果

社内向けVMレンタルサービスの運用に必要なリソース情報は、Dashboardで参照可能な範囲ではNova Computeホストに関する情報、および一時ディスクに関する情報が不足していることが分かった(「表 6-1 問題点一覧」「項番4」)。これらの情報は、各種コマンドを実行することで参照および算出する必要がある。

●参考

(1) ルートディスクについて

ルートディスクは、インスタンスタイプを作成する際にサイズを設定できるが、通常は0を設定すべきである。0 を設定すると、イメージが持つルートディスクのサイズがそのまま使用される。その場合は、Dashboard 上でもディスクサイズが 0 と表示される。

0 以外の値を設定した場合は、以下の通りとなる。

イメージが持つルートディスクよりも大きい値を設定した場合：

ルートディスク(/dev/vda)のサイズが設定したサイズとなる。ルートパーティション(/dev/vda1)のサイズはイメージが持つサイズのままであるため、fdisk などでパーティションを作成することで残りのディスクサイズを利用できる。

イメージが持つルートディスクよりも小さい値を設定した場合：

作成したインスタンスを起動できなくなる。

(2) インスタンスが使用する各ディスクの保存先とマウントポイントについて

インスタンスが使用する各ディスクのファイルイメージは、Nova Compute ホスト上の以下のディレクトリに保存される。

/var/lib/nova/<インスタンス UUID>/instances

つまり、Nova Compute ホストの/var ディレクトリには、相応のディスク空き容量が必要である。

また、インスタンスが使用する各ディスクのゲスト OS 上のマウントポイント、および Nova Compute ホスト上のファイル名を以下に示す。

用途	マウントポイント	ファイル名
ルートディスク	/	disk
一時ディスク	/mnt	disk.local
スワップ	swap	disk.swap

5.3 ホストの動的追加

ホストの動的追加の検証としては、手動で Nova Compute ホストを追加できるかを確認した。

検証の手順および結果を以下に示す。なお、本検証を実施するにあたり、Red Hat OpenStack 3.0 (Grizzly) Installation and Configuration Guide(以降、ガイド。「7.参考文献」の[2])の「16.2. Adding Compute Resources」を参考にした。手順中の[章番号 タイトル]は、ガイドの章を示す。

●設定手順

[16.2. Adding Compute Resources]

以下の章の手順を実施するよう記載されている。

10.3.3. Installing the Compute Service
10.3.4.3. Setting the Message Broker
10.3.4.4. Configuring Resource Overcommitment
10.3.4.5. Reserving Host Resources
10.3.4.6.2. Updating the Compute Configuration
10.3.4.6.3. Configuring the L2 Agent
10.3.4.6.4. Configuring Virtual Interface Plugging
10.3.4.7. Configuring the Firewall
10.3.6. Starting the Compute Services

[10.3.3. Installing the Compute Service]

(1) Nova Compute 関連パッケージを追加する Nova Compute ホスト(ホスト cn3)にインストール(行頭の "cn3#"は、ホスト cn3 の root ユーザのプロンプトを表す)

```
cn3# yum install -y openstack-nova-compute python-cinderclient
```

※ ガイドには openstack-nova-api、openstack-nova-conductor、openstack-nova-scheduler も記載されているが、これらはホスト ccn 上に構築されているため除外した。

[10.3.4.3. Setting the Message Broker]

手順 1

(2) nova.conf に rpc_backend の設定を追加

```
cn3# openstack-config --set /etc/nova/nova.conf DEFAULT rpc_backend
nova.openstack.common.rpc.impl_qpid
```

手順 2

(3) nova.conf に qpid_hostname の設定を追加

```
cn3# openstack-config --set /etc/nova/nova.conf DEFAULT qpid_hostname 10.204.51.232
```

※ QPID サーバはホスト ccn であるため、ホスト ccn の IP アドレスを設定した。

手順 3

(4) nova.conf に qpid_username および qpid_password の設定を追加

```
cn3# openstack-config --set /etc/nova/nova.conf DEFAULT qpid_username guest
cn3# openstack-config --set /etc/nova/nova.conf DEFAULT qpid_password guest
```

※ 既存の Nova Compute ホスト(ホスト cn1)の設定と同一の値を設定した。

手順 4

(5) nova.conf に qpid_protocol および qpid_port の設定を追加

```
cn3# openstack-config --set /etc/nova/nova.conf DEFAULT qpid_protocol tcp
cn3# openstack-config --set /etc/nova/nova.conf DEFAULT qpid_port 5672
```

※ 既存の Nova Compute ホスト(ホスト cn1)の設定と同一の値を設定した。

<ガイドで不足している手順があったため追加>

(6) nova.conf に glance_api_servers の設定を追加

```
cn3# openstack-config --set /etc/nova/nova.conf DEFAULT glance_api_servers 10.204.51.232:9292
```

※ 詳細については「8.1 設定不備によるエラーと対応策」の(2)参照。

(7) nova.conf に auth_strategy の設定を追加

```
cn3# openstack-config --set /etc/nova/nova.conf DEFAULT auth_strategy keystone
```

※ 詳細については「8.1 設定不備によるエラーと対応策」の(3)参照。

(8) nova.conf に VNC の設定を追加

```
cn3# openstack-config --set /etc/nova/nova.conf DEFAULT novncproxy_base_url
http://10.204.51.232:6080/vnc_auto.html
cn3# openstack-config --set /etc/nova/nova.conf DEFAULT vncserver_listen 10.204.51.237
cn3# openstack-config --set /etc/nova/nova.conf DEFAULT vncserver_proxyclient_address
10.204.51.237
```

※ 詳細については「8.1 設定不備によるエラーと対応策」の(4)参照。

[10.3.4.4. Configuring Resource Overcommitment]

overcommit ratio は、既存の Nova Compute ホストもデフォルトのままであるためスキップ

[10.3.4.5. Reserving Host Resources]

ホストリソースのリザーブ設定は、既存の Nova Compute ホストもデフォルトのままであるためスキップ

[10.3.4.6.2. Updating the Compute Configuration]

手順 1～7 は、検証環境では quantum を使用しないためスキップ

手順 8(firewall_driver の設定)は、既存の Nova Compute ホストにも設定されていないためスキップ

[10.3.4.6.3. Configuring the L2 Agent]

検証環境では quantum を使用しないためスキップ

[10.3.4.6.4. Configuring Virtual Interface Plugging]
libvirt_vif_driver は、既存の Nova Compute ホストにも設定されていないためスキップ

[10.3.4.7. Configuring the Firewall]
手順 1～4 は、ホスト cn3 上で iptables を動作させていないためスキップ

[10.3.6. Starting the Compute Services]
手順 1
(9) messagebus サービスの起動

```
cn3# service messagebus start
cn3# chkconfig messagebus on
```

手順 2
(10) libvirtd サービスの起動

```
cn3# service libvirtd start
cn3# chkconfig libvirtd on
```

手順 3～5 は、openstack-nova-api、openstack-nova-conductor、openstack-nova-scheduler をホスト ccn 上で起動しているためスキップ

手順 6
(11) openstack-nova-compute サービスの起動

```
cn3# service openstack-nova-compute start
cn3# chkconfig openstack-nova-compute on
```

手順 7 は、検証環境では openstack-nova-cert、openstack-nova-network、openstack-nova-objectstore を使用していないためスキップ

設定は以上で完了。

●動作確認
動作確認としては、以下の 4 つを実施した。

確認 1：ホストリストに cn3 が出力されること
確認 2：Dashboard からインスタンスを作成し、ホスト cn3 上でインスタンスが起動すること
確認 3：Dashboard からホスト cn3 上で稼働しているインスタンスのコンソールを操作できること
確認 4：既存の Nova Compute ホストで稼働しているインスタンスには影響が無いこと

○確認 1
ホスト ccn で以下のコマンドを実行。

```
ccn# nova-manage host list
```

出力：

host	zone
ccn	internal
cn1	nova
cn2	nova
cn3	nova

上記の通り cn3 が出力されたため、正常に Nova Compute ホストとして認識されていることが確認できた。

○確認 2
Dashboard からインスタンスを作成し、状態が Active となること、および[管理]-[インスタンス]画面で、作成したインスタンスのホスト列が cn3 となっていることを確認した。
また、ホスト cn3 上で ps コマンドを実行し、qemu-kvm プロセスが存在することを確認した。

以上より、ホスト cn3 上でインスタンスが起動することを確認できた。

○確認 3
Dashboard からホスト cn3 上で稼働しているインスタンスのコンソールに接続し、操作できることが確認できた。

○確認 4
Nova Compute ホストを追加する手順を実施する前から、既存の Nova Compute ホストで稼働しているインスタンスで top コマンドを実行していたが、上記確認 3 まで実施した後も top コマンドは動作し続けていた。

以上より、既存の Nova Compute ホストで稼働しているインスタンスには、Nova Compute ホストを追加することによる影響が無いことが確認できた。

●結果
運用中の RHOS 環境に Nova Compute ホストを追加することは、問題ないことが分かった。

5.4 VM へのアクセス制御

VM へのアクセス制御の検証としては、以下の環境を想定し、アクセス制御条件を満たせるかを確認した。

□想定環境
- グループ A
  - VM-A1
  - VM-A2
- グループ B
  - VM-B1

□アクセス制御条件
全体の管理者：すべての VM にアクセス可能
グループ A 管理者：グループ A 内のすべての VM(VM-A1,VM-A2)だけにアクセス可能
グループ A ユーザ X：VM-A1 だけにアクセス可能
グループ B ユーザ Y：VM-B1 だけにアクセス可能

しかし、OpenStack にはグループという概念がなく、かつアクセス制御はプロジェクト単位でしかできない(「表 6-1 問題点一覧」「項番 5」)ことが判明した(「7.参考文献」の[3]を参照)。
このため、上記想定環境とアクセス制御条件を OpenStack で実現可能な同等のものに再定義し、検証を実施した。再定義した想定環境とアクセス制御条件を以下に示す。

□想定環境(再定義版)
プロジェクト A1: VM-A1
プロジェクト A2: VM-A2
プロジェクト B1: VM-B1

□アクセス制御条件(再定義版)
全体の管理者：すべての VM にアクセス可能
プロジェクト A 管理者：VM-A1,VM-A2 だけにアクセス可能
プロジェクト A1 ユーザ X：VM-A1 だけにアクセス可能
プロジェクト B1 ユーザ Y：VM-B1 だけにアクセス可能

Dashboard でプロジェクトをユーザに割り当てる手順については、「7.参考文献」の[3]の「第 10 章 プロジェクトとユーザの管理」を参照。

●結果
アクセス制御条件通りに VM へのアクセスを制御することができた。このため、社内向け VM レンタルサービスで必要となるようなアクセス制御は、RHOS で実現可能であると言える。
なお、アクセス制御はプロジェクト単位のみという点には注意する必要がある。アクセス権限を分けたい VM は、すべて別プロジェクトにする必要がある。

●参考

(1) ユーザのロールについて

ユーザを作成する際にロールを"admin"にすると、そのユーザはすべてのプロジェクトへのアクセスが可能となる。つまり、全プロジェクトの管理者となる。

このため、通常はロールを"Member"または"_member_"で作成するべきである。なお、Dashboard のロールにデフォルトで表示される"Member"と"_member_"は、検証した限りでは両者に違いはなかった。

5.5 VM の CPU 使用量・ディスク I/O 帯域制御

VM の CPU 使用量・ディスク I/O 帯域制御の検証としては、以下の 2 つを実施した。

検証 1：VM の CPU 使用量の制御
検証 2：VM のディスク I/O 帯域の制御

検証の手順および結果を以下に示す。なお、本検証を実施するにあたり、Openstack.org － Wiki －Instance Resource Quota(「7.参考文献」の[4][5])を参考にした。

■検証 1：VM の CPU 使用量の制御

●参考文献概要

VM の CPU 使用量の制御は、nova-manage コマンドを使用し、インスタンスタイプに対して flavor を設定することで実現できる。

CPU 使用量の制御で使用する flavor の項目は以下の 2 つである。

cpu_period
cpu_quota

共に単位はミリ秒であり、cpu_period に指定した期間に、cpu_quota で指定した時間まで CPU を使用できる。

つまり、cpu_period=5000、cpu_quota=2500 の場合は、5 秒間に 2.5 秒まで CPU を使用できるため、CPU 使用率は最大 50%ということになる。

●検証準備

CPU に負荷をかけるため、以下の内容で cpu_load.sh を作成。

```
while true; do
        i=$(($i+1))
done
```

●確認手順

(1) Dashboard でインスタンスタイプ cputest100、cputest50 および cputest30 を作成(それぞれのインスタンスタイプの設定内容は検証に無関係のため省略)

(2) インスタンスタイプ cputest50 に CPU 使用率が最大 50%となる値を設定(行頭の"ccn#"は、ホスト ccn の root ユーザのプロンプトを表す)

```
ccn# nova-manage flavor set_key --name cputest --key quota:cpu_period --value 5000
ccn# nova-manage flavor set_key --name cputest --key quota:cpu_quota --value 2500
```

(3) インスタンスタイプ cputest30 に CPU 使用率が最大 30%となる値を設定

```
ccn# nova-manage flavor set_key --name cputest --key quota:cpu_period --value 10000
ccn# nova-manage flavor set_key --name cputest --key quota:cpu_quota --value 3000
```

(4) インスタンスタイプ cputest100 からインスタンス cpu100 を、cputest50 から cpu50 を、cputest30 から cpu30 を作成

(5) インスタンス cpu100、cpu50 および cpu30 で cpu_load.sh を実行

(6) インスタンス cpu100、cpu50 およｂcpu30 で top コマンドを実行し、CPU 使用率を確認

●結果

それぞれのインスタンスの CPU 使用率は、設定どおり以下の結果となった。

cpu100(設定なし): 約 100%
cpu50(50%設定): 約 50%
cpu30(30%設定): 約 30%

以上より、CPU 使用率を VM ごとに制御する運用ができることが分かった。つまり、オーバーコミットで CPU を割り当てても、各 VM の CPU 使用率を制御することにより、CPU 使用率を保障したり、一つの VM の CPU 使用状況が他の VM に影響を与えない環境にすることができる。

●参考

(1) 各インスタンスタイプの flavor の確認方法

ホスト ccn 上で以下のコマンドを実行する。

```
# nova-manage flavor list
```

出力例：

```
cputest100: Memory: 1024MB, VCPUS: 1, Root: 0GB, Ephemeral: 0Gb, FlavorID:
798032ab-5911-47ee-8300-12cd59f99986, Swap: 0MB, RXTX Factor: 1.0, public, ExtraSpecs {}
cputest50: Memory: 1024MB, VCPUS: 1, Root: 0GB, Ephemeral: 0Gb, FlavorID:
abdad447-ab52-43ce-a34c-0ef2badcd148, Swap: 0MB, RXTX Factor: 1.0, public, ExtraSpecs
{u'quota:cpu_period': u'5000', u'quota:cpu_quota': u'2500'}
cputest30: Memory: 1024MB, VCPUS: 1, Root: 0GB, Ephemeral: 0Gb, FlavorID:
63b80bdb-6450-446e-a641-f7dbb50f9d03, Swap: 0MB, RXTX Factor: 1.0, public, ExtraSpecs
{u'quota:cpu_period': u'10000', u'quota:cpu_quota': u'3000'}
```

上記出力例の通り、quota:cpu_period や quota:cpu_quota に値が設定されていることが分かる。

(2) インスタンスの CPU 使用量制御設定の確認方法

インスタンスが起動しているホスト上で以下のコマンドを実行する。

```
# virsh schedinfo インスタンス名
```

出力例(デフォルト：flavor の設定なし)：

```
スケジューラー: posix
cpu_shares       : 1024
vcpu_period      : 100000
vcpu_quota       : -1
emulator_period: 100000
emulator_quota : -1
```

(3) 作成済みインスタンスのインスタンスタイプに flavor 設定を追加し場合

作成済みインスタンスのインスタンスタイプに flavor 設定を追加してみたが、インスタンスの設定は変わらなかった。また、インスタンスを再起動しても設定は変わらなかった。

つまり、インスタンスタイプの flavor 設定を変更しても、作成済みのインスタンスには影響しない。

■検証 2：VM のディスク I/O 帯域の制御

●参考文献概要

VM のディスク I/O 帯域の制御は、nova-manage コマンドを使用し、インスタンスタイプに対して flavor を設定することで実現できる。

ディスク I/O 帯域の制御で使用する flavor の項目は以下の 6 つである。

disk_read_bytes_sec
disk_read_iops_sec
disk_write_bytes_sec
disk_write_iops_sec
disk_total_bytes_sec
disk_total_iops_sec

*_bytes_sec 項目の単位はバイト/秒、*_iops_sec 項目の単位は I/O 数/秒である。

●確認手順

検証 1 の手順と同様に、「表 5-10」のインスタンスタイプおよびインスタンスを作成した。

表 5-10 I/O 帯域制御検証用インスタンス一覧

インスタンスタイプ名	インスタンス名	flavor 項目	設定値
read10m.test	read10m	disk_read_bytes_sec	10240000
read1000iops.test	read1000iops	disk_read_iops_sec	1000
write10m.test	write10m	disk_write_bytes_sec	10240000
write1000iops.test	write1000iops	disk_write_iops_sec	1000
rw10m.test	rw10m	disk_total_bytes_sec	10240000
rw1000iops.test	rw1000iops	disk_total_iops_sec	1000

インスタンスの作成がエラー(Dashboard の[インスタンス]画面の[状態]列が"Error")となり、Nova Compute ホストの/var/log/nova/compute.log に以下のログが出力された。

```
libvirtError: unsupported configuration: block I/O throttling not supported with this QEMU binary
```

virsh コマンドにも VM のディスク I/O 帯域制御の flavor 項目と類似のオプションがあるため、Nova Compute ホスト上で以下のコマンドを実行した。

```
# virsh blkdeviotune インスタンス名 vda --read-bytes-sec 10240000
# virsh blkdeviotune インスタンス名 vda --read_iops_sec 1000
# virsh blkdeviotune インスタンス名 vda --write-bytes-sec 10240000
# virsh blkdeviotune インスタンス名 vda --write_iops_sec 1000
# virsh blkdeviotune インスタンス名 vda --total-bytes-sec 10240000
# virsh blkdeviotune インスタンス名 vda --total-iops-sec 1000
```

上記コマンドを実行した結果、インスタンス作成と同様の以下のエラーメッセージがコンソールに出力された。

```
エラー: ブロック I/O スロットルを変更できません
エラー: unsupported configuration: block I/O throttling not supported with this QEMU binary
```

●結果

VMのディスクI/O帯域の制御は、Novaとしては機能があるものの、RHOS環境ではqemu-kvm[7]がサポートしていないために使用できないという結果となった(「表 6-1 問題点一覧」「項番 6」)。RHOSの今後のバージョンでサポートされるものと思われる。

[7] 検証環境の qemu-kvm は、qemu-kvm-rhev-0.12.1.2-2.355.el6_4.5.x86_64.rpm パッケージに含まれるものである。アップストリームの qemu-kvm を使用することで VM のディスク I/O 帯域を制御できる可能性があるが、本検証では実施していない。

6. まとめ

本書では、RHOS での社内向け VM レンタルサービスの運用検証を実施した。その結果、6 件の問題があり、同時に対応策についても明らかにした(表 6-1)。本検証の結果から、こうした対策を取ることによって、RHOS をそのまま利用しても社内向け VM レンタルサービス程度であれば最低限運用していける水準に達していることがわかった。しかしながら、その他のサービスの場合についてはそれぞれに検証が必要である。また、VM レンタルサービスについても、より効率的な運用をするのであれば、シェルスクリプトの補完や他のソフトウェアと組み合わせるなどの対処が必要となる。このほか、RHOS3.0 以降のアップデートで正式にサポートされる予定の Ceilimeter(Metering 機能)や Heat(Orchestration 機能)を利用することにより、参照可能なリソース情報が充実するなど、効率的な管理が可能となることが予想される。

表 6-1 問題点一覧

項番	問題	対応策	章	レベル※
1	Dashboard から 1 操作で多数の VM を作成/起動することができない。	nova boot コマンドを連続実行するシェルスクリプトを作成して VM を作成/起動する。	5.1	C
2	未割り当て Floating IP が不正な割り当て状態となり、割り当て不可となる。	nova floating-ip-delete コマンドで不正な割り当て状態の Floating IP を未割り当て状態に戻す。	5.1	C
3	Dashboard から 1 操作で多数の VM を削除することができない。	nova delete コマンドを連続実行するシェルスクリプトを作成して VM を削除する。	5.1	C
4	Dashboard では Nova Compute ホストのリソース情報や一時ディスクのリソース情報を参照できない。	各種コマンドを使用してリソース情報を参照または算出する。	5.2	C
5	OpenStack にはグループという概念がない。	アクセス権限を分けたい VM は、すべて別プロジェクトにする。	5.4	C
6	VM のディスク I/O 帯域の制御ができない(RHOS の qemu-kvm がサポートしていない)。	VM レンタルサービス運用の上で、必須機能ではない。また、近くサポートされる見込み。	5.5	D

※レベルの凡例を以下に示す。

A: RHOS 環境に影響を与えるような致命的な問題。

B: 対応策がない問題。

C: 対応策がある問題。

D: 運用に影響がない問題。

7. 参考文献

[1] Red Hat OpenStack 3.0 (Grizzly) Getting Started Guide
URL=https://access.redhat.com/site/documentation/en-US/Red_Hat_OpenStack/3/pdf/Getting_Started_Guide/Red_Hat_OpenStack-3-Getting_Started_Guide-en-US.pdf

[2] Red Hat OpenStack 3.0 (Grizzly) Installation and Configuration Guide
URL=https://access.redhat.com/site/documentation/en-US/Red_Hat_OpenStack/3/pdf/Installation_and_Configuration_Guide/Red_Hat_OpenStack-3-Installation_and_Configuration_Guide-en-US.pdf

[3] OpenStack 運用ガイド – 第 10 章 プロジェクトとユーザーの管理
URL=http://dream.daynight.jp/openstack/openstack-ops/content/projects_users.html

[4] Openstack.org – Wiki – Instance Resource Quota
URL=https://wiki.openstack.org/wiki/InstanceResourceQuota

[5] OpenStack Cloud Administrator Guide – Flavors
URL=http://docs.openstack.org/admin-guide-cloud/content/customize-flavors.html

8. 付録

8.1 設定不備によるエラーと対応策

本検証を実施した際に、設定不備により発生したエラーと、その対策法を以下に示す。

(1) インスタンス(VM)を起動できない 1

[現象]

インスタンス(VM)を起動できない。

インスタンスを起動しようとした Nova Compute ホストの/var/log/nova/compute.log に以下のようなログが出力される。

```
libvirtError: internal error no supported architecture for os type 'hvm'
```

また、/var/log/dmesg に以下のログが出力される。

```
kernel: kvm: no hardware support
kernel: kvm: disabled by bios
```

lsmod コマンドで確認すると、kvm モジュールはロードされているが、kvm-intel モジュールがロードされていない。

modprobe kvm-intel コマンドを実行すると、以下のようなメッセージが出力される。

```
FATAL: Error inserting kvm_intel
(/lib/modules/2.6.32-358.118.1.openstack.el6.x86_64/kernel/arch/x86/kvm/kvm-intel.ko):
Operation not supported
```

[原因]

BIOS で Intel-VT が無効になっていることが原因である。

[対応策]

BIOS で Intel-VT を有効にすることで対策できる。

(2) インスタンス(VM)を起動できない 2

[現象]

インスタンス(VM)を起動できない。

インスタンスを起動しようとした Nova Compute ホストの/var/log/nova/compute.log に以下のようなログが出力される。

```
2013-10-28 20:02:54.750 85913 TRACE nova.openstack.common.rpc.amqp
GlanceConnectionFailed: Connection to glance host 10.204.51.237:9292 failed: Error
communicating with http://10.204.51.237:9292 [Errno 111] ECONNREFUSED
```

[原因]

当該 Nova Compute ホストから glance サービスに接続できないことが原因である。
/etc/nova/nova.conf の glance_api_servers の設定が不正、glance サービスが起動していない、ネットワーク経路の問題などが考えられる。
なお、「7.参考文献」の[2] Red Hat OpenStack 3.0 (Grizzly) Installation and Configuration Guide の「16.2. Adding Compute Resources」を参照し、既存の RHOS 環境に Nova Compute ホストを追加した場合は、/etc/nova/nova.conf の glance_api_servers の設定が不足しているため本現象が発生する。

[対応策]

Nova Compute ホストから glance サービスに接続できるようにすることで対策できる。

/etc/nova/nova.conf の glance_api_servers の設定が不足している場合は、当該 Nova Compute ホストで以下のコマンドを実行し、openstack-nova-compute サービスを再起動することで対策できる。コマンドの行頭の#は、root ユーザのプロンプトを表す。

実行コマンド(例)：

```
# openstack-config --set /etc/nova/nova.conf DEFAULT glance_api_servers
10.204.51.232:9292
```

(3) インスタンス(VM)を起動できない 3

[現象]

インスタンス(VM)を起動できない。

インスタンスを起動しようとした Nova Compute ホストの/var/log/nova/compute.log に以下のようなログが出力される。

```
2013-10-28 20:57:14.706 86358 TRACE nova.openstack.common.rpc.amqp
ImageNotAuthorized: Not authorized for image c8dd0d85-4e1b-44dd-b4fe-7d5e3015159b.
```

[原因]

当該 Nova Compute ホストの/etc/nova/nova.conf の auth_strategy の設定が不正であることが原因である。
なお、「7.参考文献」の[2] Red Hat OpenStack 3.0 (Grizzly) Installation and Configuration Guide の「16.2. Adding Compute Resources」を参照し、既存の RHOS 環境に Nova Compute ホストを追加した場合は、/etc/nova/nova.conf の auth_strategy の設定が不足しているため、本現象が発生する。

[対応策]

当該 Nova Compute ホストで以下のコマンドを実行し、openstack-nova-compute サービスを再起動することで対策できる。コマンドの行頭の#は、root ユーザのプロンプトを表す。

実行コマンド：

```
# openstack-config --set /etc/nova/nova.conf DEFAULT auth_strategy keystone
```

(4) Dashboard からインスタンスのコンソールに接続できない

[現象]

Dashboard からインスタンスのコンソールに接続できない。

Dashboard に以下のようなエラーメッセージ(127.0.0.1 への接続が不可)が表示される。

[原因]

インスタンスが起動している Nova Compute ホストの/etc/nova/nova.conf に、VNC の設定がされていないことが原因である。

当該 Nova Compute ホストで netstat -tlnp コマンドを実行すると、以下のように qemu-kvm プロセスが 127.0.0.1 で VNC のポート(6080)を LISTEN していることが分かる。127.0.0.1 では、リモートから接続できないため、このような現象が発生する。

netstat -tlnp 出力例：

```
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address               Foreign Address             State
PID/Program name
tcp        0      0 127.0.0.1:6080              *:*                         LISTEN
86856/qemu-kvm
```

なお、「7.参考文献」の[2] Red Hat OpenStack 3.0 (Grizzly) Installation and Configuration Guide の「16.2. Adding Compute Resources」を参照し、既存の RHOS 環境に Nova Compute ホストを追加した場合は、VNC の設定が不足しているため、本現象が発生する。

[対応策]

当該 Nova Compute ホストで以下のコマンドを実行し、その後 openstack-nova-compute サービスを再起動することで対策できる。コマンドの行頭の#は、root ユーザのプロンプトを表す。

実行コマンド(例)：

```
# openstack-config --set /etc/nova/nova.conf DEFAULT novncproxy_base_url
http://10.204.51.232:6080/vnc_auto.html
# openstack-config --set /etc/nova/nova.conf DEFAULT vncserver_listen 10.204.51.237
# openstack-config --set /etc/nova/nova.conf DEFAULT vncserver_proxyclient_address
10.204.51.237
```

なお、当該 Nova Compute ホストで稼働中のインスタンスは、127.0.0.1:6080 で LISTEN したままであるため、インスタンスをリセットする必要がある。インスタンスをリセットするには、Dashboard の[プロジェクト]-[インスタンス]で当該インスタンスの[さらに]ボタンをクリックし、表示されたメニューから[インスタンスのハードリブート]をクリックする。[インスタンスのソフトリブート]では、127.0.0.1 が LISTEN されたままであることに注意。

対策後は、qemu-kvm プロセスは、以下のようにリモートからアクセス可能な IP アドレスで VNC のポート(6080)を LISTEN する。

netstat -tlnp 出力例：

```
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address               Foreign Address             State
PID/Program name
tcp        0      0 10.204.51.237:vnc-server    *:*                         LISTEN
87927/qemu-kvm
```


以上