Element Manager
Installation Manual

第 0.9 版

2017 年 10 月

更新履歴

日付	版数	記入者	変更箇所	変更内容
2017/10	0.9	−	−	初版登録

目次

表目次

1. はじめに

1.1. 目的

本ドキュメントは、Element Manager （以下 EM）に含まれる EM モジュールのインストールマニュアルです。本ソフトウェアを動作するにあたり、事前に内容をご確認のうえ、ご使用下さい。

1.2. 適用範囲

本ドキュメントの適用範囲は、EM モジュールの構成品になります。

上述以外に、本書を適用することは出来ません。

1.3. 文言表現の意味

本文では、以下のような文字表現をしています。内容をよく理解してから、本文をお読み下さい。

- **【XX XX】**： 太字中括弧囲み。
  この場合は、Linux OS へのコマンド投入を意味します。
- **X [Enter]**：太字＋[Enter]。
  この場合は、コンソール画面での、キー入力＋エンターキー押下を意味します。

1.4. 商標

本ドキュメントに記載されている会社名、商品名は、各社の登録商標または商標です。

Linux®

Linus Torvalds 氏の米国及びその他の国における登録商標あるいは商標です。

PostgreSQL®

PostgreSQL の米国およびその他の国における商標です。

1.5. 付属品の構成

本ドキュメントのインストール手順に必要な付属品について以下の表 1-1 付属品に示す。
また、備考に事前ダウンロードの記述がある項目については本資料のインストールを実施する前にダウンロードを行い入手すること。

表 1-1 付属品

No	フォルダ構成			ファイル名	説明	備考
1.	em			-	-	-
2.		bin		em	リソースエージェント	-
3.				em_ctl.sh	EM 起動スクリプト	-
4.				EmMonitor.pyc	死活監視クライアント	-
5.		lib		MsfEmMain.pyc	メインモジュール	-
6.				GlobalModule.pyc	グローバルモジュール	-
7.				EmCommonLog.pyc	EM 共通ログモジュール	-
8.				__init__.pyc	初期設定モジュール	-
9.			CommonDriver	EmCommonDriver.pyc	ドライバ共通部モジュール	-
10.				__init__.pyc	初期設定モジュール	-
11.			Config	EmConfigManagement.pyc	コンフィグ管理モジュール	-
12.				__init__.pyc	初期設定モジュール	-
13.			DB	EmDBControl.pyc	DB 制御モジュール	-
14.				__init__.pyc	初期設定モジュール	-
15.			DriverUtility	EmDriverCommonUtilityDB.pyc	ドライバ共通ユーティリティ(DB)モジュール	-
16.				EmDriverCommonUtilityLog.pyc	ドライバ共通ユーティリティ(Log)モジュール	-
17.				__init__.pyc	初期設定モジュール	-
18.			NetconfServer	EmNetconfServer.pyc	EM Netconf Server モジュール	-
19.				__init__.pyc	初期設定モジュール	-
20.			OrderflowControl	EmOrderflowControl.pyc	オーダフローコントロールモジュール	-
21.				__init__.pyc	初期設定モジュール	-
22.			Protocol	EmNetconfProtocol.pyc	対装置プロトコル処理モジュール	-
23.				__init__.pyc	初期設定モジュール	-
24.			Scenario	EmCeLagDelete.pyc	CE 向け LAG 削除シナリオ	-
25.				EmCeLagMerge.pyc	CE 向け LAG 追加シナリオ	-
26.				EmInternalLagDelete.pyc	内部 Link 向け LAG 削除シナリオ	-
27.				EmInternalLagMerge.pyc	内部 Link 向け LAG 追加シナリオ	-
28.				EmL2SliceDelete.pyc	L2 スライス削除シナリオ	-
29.				EmL2SliceGet.pyc	L2 スライス情報整合シナリオ	-
30.				EmL2SliceMerge.pyc	L2 スライス追加シナリオ	-
31.				EmL3SliceDelete.pyc	L3 スライス削除シナリオ	-
32.				EmL3SliceGet.pyc	L3 スライス情報整合シナリオ	-
33.				EmL3SliceMerge.pyc	L3 スライス追加シナリオ	-
34.				EmLeafDelete.pyc	Leaf 減設シナリオ	-
35.				EmLeafMerge.pyc	Leaf 増設シナリオ	-
36.				EmSpineDelete.pyc	Spine 減設シナリオ	-
37.				EmSpineMerge.pyc	Spine 増設シナリオ	-
38.				EmSeparateScenario.pyc	個別シナリオモジュール	-

No	フォルダ構成	ファイル名	説明	備考
39.		__init__.pyc	初期設定モジュール	-
40.	SeparateDriver	CiscoDriver.pyc	Cisco ドライバモジュール	-
41.		JuniperDriver5100.pyc	Juniper5100 ドライバモジュール	-
42.		JuniperDriver5200.pyc	Juniper5200 ドライバモジュール	-
43.		EmSeparateDriver.pyc	ドライバ個別モジュール	-
44.		__init__.pyc	初期設定モジュール	-
45.	SystemUtility	EmSysCommonUtilityDB.pyc	システム共通(DB)ユーティリティモジュール	-
46.		__init__.pyc	初期設定モジュール	-
47.	conf	conf_driver.conf	ドライバ個別部動作設定ファイル	-
48.		conf_if_process.conf	IF 処理部動作設定ファイル	-
49.		conf_scenario.conf	シナリオ個別部動作設定ファイル	-
50.		conf_sys_common.conf	EM 共通設定ファイル	-
51.	installer	-	-	-
52.		whl_package.tar	使用 Python ライブラリパッケージ	事前ダウンロード
53.		pip-8.1.2.tar.gz	Python ライブラリインストール用 PIP コマンド	事前ダウンロード
54.		setuptools-28.6.0.tar	pip 依存パッケージ	事前ダウンロード
55.	dhcp.v4.2.5	dhcp-4.2.5-42.el7.centos.x86_64.rpm	DHCP インストールパッケージ	事前ダウンロード
56.	ntp.v4.2	autogen-libopts-5.18-5.el7.x86_64.rpm	NTP インストールパッケージ	事前ダウンロード
57.		ntpdate-4.2.6p5-22.el7.centos.x86_64.rpm	NTP インストールパッケージ	事前ダウンロード
58.		ntp-4.2.6p5-22.el7.centos.x86_64.rpm	NTP インストールパッケージ	事前ダウンロード
59.	postgresql.v9.3.13	postgresql93-9.3.13-1PGDG.rhel7.x86_64.rpm	PostgreSQL インストールパッケージ	事前ダウンロード
60.		postgresql93-contrib-9.3.13-1PGDG.rhel7.x86_64.rpm	PostgreSQL インストールパッケージ	事前ダウンロード
61.		postgresql93-devel-9.3.13-1PGDG.rhel7.x86_64.rpm	PostgreSQL インストールパッケージ	事前ダウンロード
62.		postgresql93-libs-9.3.13-1PGDG.rhel7.x86_64.rpm	PostgreSQL インストールパッケージ	事前ダウンロード
63.		postgresql93-server-9.3.13-1PGDG.rhel7.x86_64.rpm	PostgreSQL インストールパッケージ	事前ダウンロード
64.		uuid-1.6.2-26.el7.x86_64.rpm	PostgreSQL 依存パッケージ	事前ダウンロード
65.		libxslt-1.1.28-5.el7.x86_64.rpm	PostgreSQL 依存パッケージ	事前ダウンロード
66.	pacemaker.v1.1.14-1.1	pacemaker-1.1.14-1.el7.x86_64.rpm	Pacemaker インストールパッケージ	事前ダウンロード
67.		corosync-2.3.5-1.el7.x86_64.rpm	Corosync インストールパッケージ	事前ダウンロード
68.		crmsh-2.1.5-1.el7.x86_64.rpm	crm コマンドインストールパッケージ	事前ダウンロード
69.		pcs-0.9.143-15.el7.x86_64.rpm	pcs コマンドインストールパッケージ	最新パッケージは 0.9.149-1 だが、試用版のため一つ前の ver を使用
70.		cluster-glue-1.0.12-2.el7.x86_64.rpm	Pacemaker 依存パッケージ	事前ダウンロード
71.		cluster-glue-libs-1.0.12-2.el7.x86_64.rpm	Pacemaker 依存パッケージ	事前ダウンロード

No	フォルダ構成	ファイル名	説明	備考
72.		corosynclib-2.3.5-1.el7.x86_64.rpm	Corosync 依存パッケージ	事前ダウンロード
73.		ipmitool-1.8.13-9.el7_2.x86_64.rpm	Pacemaker 依存パッケージ	事前ダウンロード
74.		libqb-1.0-1.el7.x86_64.rpm	Pacemaker 依存パッケージ	事前ダウンロード
75.		libtool-ltdl-2.4.2-21.el7_2.x86_64.rpm	Pacemaker 依存パッケージ	事前ダウンロード
76.		libxslt-1.1.28-5.el7.x86_64.rpm	Pacemaker 依存パッケージ	事前ダウンロード
77.		libyaml-0.1.4-11.el7_0.x86_64.rpm	Pacemaker 依存パッケージ	事前ダウンロード
78.		lm_sensors-libs-3.3.4-11.el7.x86_64.rpm	Pacemaker 依存パッケージ	事前ダウンロード
79.		nano-2.3.1-10.el7.x86_64.rpm	crm 依存パッケージ	事前ダウンロード
80.		net-snmp-agent-libs-5.7.2-24.el7_2.1.x86_64.rpm	Corosync 依存パッケージ	事前ダウンロード
81.		net-snmp-libs-5.7.2-24.el7_2.1.x86_64.rpm	Corosync 依存パッケージ	事前ダウンロード
82.		openhpi-libs-3.4.0-2.el7.x86_64.rpm	Pacemaker 依存パッケージ	事前ダウンロード
83.		OpenIPMI-libs-2.0.19-11.el7.x86_64.rpm	Pacemaker 依存パッケージ	事前ダウンロード
84.		OpenIPMI-modalias-2.0.19-11.el7.x86_64.rpm	Pacemaker 依存パッケージ	事前ダウンロード
85.		pacemaker-cli-1.1.14-1.el7.x86_64.rpm	Pacemaker 依存パッケージ	事前ダウンロード
86.		pacemaker-cluster-libs-1.1.14-1.el7.x86_64.rpm	Pacemaker 依存パッケージ	事前ダウンロード
87.		pacemaker-libs-1.1.14-1.el7.x86_64.rpm	Pacemaker 依存パッケージ	事前ダウンロード
88.		pacemaker-all-1.1.14-1.1.el7.noarch.rpm	Pacemaker 依存パッケージ	事前ダウンロード
89.		perl-5.16.3-286.el7.x86_64.rpm	Pacemaker 依存パッケージ	事前ダウンロード
90.		perl-Carp-1.26-244.el7.noarch.rpm	Pacemaker 依存パッケージ	事前ダウンロード
91.		perl-constant-1.27-2.el7.noarch.rpm	Pacemaker 依存パッケージ	事前ダウンロード
92.		perl-Encode-2.51-7.el7.x86_64.rpm	Pacemaker 依存パッケージ	事前ダウンロード
93.		perl-Exporter-5.68-3.el7.noarch.rpm	Pacemaker 依存パッケージ	事前ダウンロード
94.		perl-File-Path-2.09-2.el7.noarch.rpm	Pacemaker 依存パッケージ	事前ダウンロード
95.		perl-File-Temp-0.23.01-3.el7.noarch.rpm	Pacemaker 依存パッケージ	事前ダウンロード
96.		perl-Filter-1.49-3.el7.x86_64.rpm	Pacemaker 依存パッケージ	事前ダウンロード
97.		perl-Getopt-Long-2.40-2.el7.noarch.rpm	Pacemaker 依存パッケージ	事前ダウンロード
98.		perl-HTTP-Tiny-0.033-3.el7.noarch.rpm	Pacemaker 依存パッケージ	事前ダウンロード
99.		perl-libs-5.16.3-286.el7.x86_64.rpm	Pacemaker 依存パッケージ	事前ダウンロード
100.		perl-macros-5.16.3-286.el7.x86_64.rpm	Pacemaker 依存パッケージ	事前ダウンロード
101.		perl-parent-0.225-244.el7.noarch.rpm	Pacemaker 依存パッケージ	事前ダウンロード
102.		perl-PathTools-3.40-5.el7.x86_64.rpm	Pacemaker 依存パッケージ	事前ダウンロード
103.		perl-Pod-Escapes-1.04-286.el7.noarch.rpm	Pacemaker 依存パッケージ	事前ダウンロード
104.		perl-podlators-2.5.1-3.el7.noarch.rpm	Pacemaker 依存パッケージ	事前ダウンロード

No	フォルダ構成	ファイル名	説明	備考
105.		perl-Pod-Perldoc-3.20-4.el7.noarch.rpm	Pacemaker 依存パッケージ	事前ダウンロード
106.		perl-Pod-Simple-3.28-4.el7.noarch.rpm	Pacemaker 依存パッケージ	事前ダウンロード
107.		perl-Pod-Usage-1.63-3.el7.noarch.rpm	Pacemaker 依存パッケージ	事前ダウンロード
108.		perl-Scalar-List-Utils-1.27-248.el7.x86_64.rpm	Pacemaker 依存パッケージ	事前ダウンロード
109.		perl-Socket-2.010-3.el7.x86_64.rpm	Pacemaker 依存パッケージ	事前ダウンロード
110.		perl-Storable-2.45-3.el7.x86_64.rpm	Pacemaker 依存パッケージ	事前ダウンロード
111.		perl-Text-ParseWords-3.29-4.el7.noarch.rpm	Pacemaker 依存パッケージ	事前ダウンロード
112.		perl-threads-1.87-4.el7.x86_64.rpm	Pacemaker 依存パッケージ	事前ダウンロード
113.		perl-threads-shared-1.43-6.el7.x86_64.rpm	Pacemaker 依存パッケージ	事前ダウンロード
114.		perl-TimeDate-2.30-2.el7.noarch.rpm	Pacemaker 依存パッケージ	事前ダウンロード
115.		perl-Time-HiRes-1.9725-3.el7.x86_64.rpm	Pacemaker 依存パッケージ	事前ダウンロード
116.		perl-Time-Local-1.2300-2.el7.noarch.rpm	Pacemaker 依存パッケージ	事前ダウンロード
117.		pm_crmgen-2.1-1.el7.noarch.rpm	Pacemaker 依存パッケージ	事前ダウンロード
118.		pm_diskd-2.2-1.el7.x86_64.rpm	Diskd RA パッケージ	事前ダウンロード
119.		pm_extras-2.2-1.el7.x86_64.rpm	VIPCheck RA パッケージ	事前ダウンロード
120.		pm_logconv-cs-2.2-1.el7.noarch.rpm	Pacemaker 依存パッケージ	事前ダウンロード
121.		psmisc-22.20-9.el7.x86_64.rpm	Pacemaker 依存パッケージ	事前ダウンロード
122.		pssh-2.3.1-5.el7.noarch.rpm	crm 依存パッケージ	事前ダウンロード
123.		python-clufter-0.50.4-1.el7.x86_64.rpm	pcs 依存パッケージ	事前ダウンロード
124.		python-dateutil-1.5-7.el7.noarch.rpm	pcs 依存パッケージ	事前ダウンロード
125.		python-lxml-3.2.1-4.el7.x86_64.rpm	pcs 依存パッケージ	事前ダウンロード
126.		resource-agents-3.9.7-1.2.6f56.el7.x86_64.rpm	仮想 IPRA を含む標準 RA パッケージ	事前ダウンロード
127.		ruby-2.0.0.598-25.el7_1.x86_64.rpm	pcs 依存パッケージ	事前ダウンロード
128.		rubygem-bigdecimal-1.2.0-25.el7_1.x86_64.rpm	pcs 依存パッケージ	事前ダウンロード
129.		rubygem-io-console-0.4.2-25.el7_1.x86_64.rpm	pcs 依存パッケージ	事前ダウンロード
130.		rubygem-json-1.7.7-25.el7_1.x86_64.rpm	pcs 依存パッケージ	事前ダウンロード
131.		rubygem-psych-2.0.0-25.el7_1.x86_64.rpm	pcs 依存パッケージ	事前ダウンロード
132.		rubygem-rdoc-4.0.0-25.el7_1.noarch.rpm	pcs 依存パッケージ	事前ダウンロード
133.		rubygems-2.0.14-25.el7_1.noarch.rpm	pcs 依存パッケージ	事前ダウンロード
134.		ruby-irb-2.0.0.598-25.el7_1.noarch.rpm	pcs 依存パッケージ	事前ダウンロード
135.		ruby-libs-2.0.0.598-25.el7_1.x86_64.rpm	pcs 依存パッケージ	事前ダウンロード
136.		pacemaker_install.sh	パッケージインストーラ	事前ダウンロード
137.	script	-	-	-
138.		ra_config.xlsx	リソースエージェント設定ファイル	-
139.		create_table.sql	テーブル作成スクリプト	-

2. 動作環境

2.1. コントローラの EM モジュール起動サーバ

以下の Linux マシン環境上での動作を推奨する。

表 2-1 推奨ハードウェア構成

No.	機器	動作保証
1.	OS	CentOS7.2 x86_64
2.	CPU	Intel(R) Xeon(R) CPU E5-2420 v2 @ 2.20GHz 12Core
3.	メモリ	32GB 以上
4.	ハードディスク空き容量	空き容量 500G 以上
5.	NIC	2 ポート以上

3. コントローラサーバインストール準備

3.1. OS インストール

OS インストール方法は「Fabric_Controller_Installation_Manual」を参照のこと。

4. コントローラサーバインストール

4.1. 環境インストール

＜実行ホスト：ACT/SBY＞

インストール時にファイルを配置するワーク用のフォルダ（以下、ワークフォルダ）を作成する。
（環境インストールが完了した時点で削除する）

【mkdir ~/setup】[Enter]

作成後、付属品の em フォルダをワークフォルダに配置する。

4.1.1. ファイアウォール設定

4.1.1.1. ファイアウォールの確認

＜実行ホスト：DB/ACT/SBY＞

ファイアウォールが設定されているかの確認を行う。

【firewall-cmd --state】[Enter]

ファイアウォールが設定されている場合、画面には以下のメッセージが表示される。

running

この場合続けて、4.1.1.2～4.1.1.6 までの操作を行う。

Firewall がインストールされていない、もしくは起動していない場合、以下の表示になる。

（起動していない場合）not running

（インストールされていない場合）bash: firewall-cmd: command not found

この場合、以下の 4.1.1.2～4.1.1.6 までの操作は不要である。

4.1.1.2. Netconf 要求（EC、起動通知）の接続許可

＜実行ホスト：ACT/SBY＞

以下のコマンドを実行し、Netconf 要求で使用するポートの接続許可を行う。
(本ドキュメントに記載するポートはデフォルト値の為、コンフィグと共に変更も可能)

【firewall-cmd --permanent --add-port=830/tcp】[Enter]

4.1.1.3. PostgreSQL の接続許可

＜実行ホスト：DB＞

以下のコマンドを実行して、PostgreSQL に使用するポートの接続許可を行う。

【firewall-cmd --permanent --add-port=5432/tcp】[Enter]

4.1.1.4. NTP の接続許可

＜実行ホスト：ACT/SBY＞

以下のコマンドを実行して、NTP に使用するポートの接続許可を行う。

【firewall-cmd --permanent --add-service=ntp】[Enter]

4.1.1.5. Pacemaker/Corosync の接続許可

＜実行ホスト：ACT/SBY＞

以下のコマンドを実行して、Pacemaker/Corosync に使用するポートの接続許可を行う。

【firewall-cmd --permanent --add-service=high-availability】[Enter]

4.1.1.6. ファイアウォールの最終設定

＜実行ホスト：ACT/SBY＞

上記、4.1.1 ファイアウォール設定が完了次第、以下のコマンドを実行し、設定を反映する。

【firewall-cmd --reload】[Enter]

【systemctl restart firewalld】[Enter]

設定反映後、設定情報を以下のコマンドを実行して、確認する（網掛け部分を特に確認する）。

【firewall-cmd --list-all】[Enter]

```
public (default, active)
  interfaces: eth0 eth1
  sources:
  services: dhcpv6-client ntp ssh high-availability
  ports: 830/tcp
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
```

＜実行ホスト：DB＞

DB 側でも、以下のコマンドを実行し、設定を反映する。

【firewall-cmd --reload】[Enter]

【systemctl restart firewalld】[Enter]

設定反映後、設定情報を以下のコマンドを実行して、確認する（網掛け部分を特に確認する）。

【firewall-cmd --list-all】[Enter]

```
public (default, active)
  interfaces:
  sources:
  services: ssh
  ports: 5432/tcp
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
```

4.1.2. Pacemaker のインストール及び設定

4.1.2.1. pacemaker のインストール

＜実行ホスト：ACT/SBY＞

以下のコマンドを実行し、インストーラを起動し、インストールを行う。

【cd ~/setup/em/installer/pacemaker.v1.1.14-1.1】[Enter]

【sh pacemaker_install.sh】[Enter]

補足：インストール中「警告: pssh-2.3.1-5.el7.noarch.rpm: ヘッダー V3 RSA/SHA256 Signature、鍵 ID 352c64e5: NOKEY」といった形で鍵がないことを理由に画面へ警告が表示されるが、インストールは進行するので問題ない。

4.1.2.2. pacemaker のインストール確認

＜実行ホスト：ACT/SBY＞

以下のコマンドを実行して、Corosync のバージョンを確認する。

【corosync -version】[Enter]

インストールが正常に完了した場合は、以下が表示される。

Corosync Cluster Engine, version '2.3.5'

Copyright (c) 2006-2009 Red Hat, Inc.

以下のコマンドを実行して、pcs のバージョンを確認する。

【pcs --version】[Enter]

インストールが正常に完了した場合は、以下が表示される。

0.9.143

以下のコマンドを実行して、Pacemaker のバージョンを確認する。

【crmadmin --version】[Enter]

インストールが正常に完了した場合は、以下が表示される。

Pacemaker 1.1.14-1.el7

Written by Andrew Beekhof

以下のコマンドを実行して、crm のバージョンを確認する。

【crm --version】[Enter]

インストールが正常に完了した場合は、以下が表示される。

2.1.5-1.el7 (Build unknown)

以下のコマンドを実行して、用いるリソースエージェントがインストールされているかを確認する。

【ls /lib/ocf/resource.d/pacemaker/】[Enter]

インストールが正常に完了した場合は、以下が表示される。

diskd

以下のコマンドを実行して、用いるリソースエージェントがインストールされているかを確認する。

【ls /lib/ocf/resource.d/heartbeat/】[Enter]

インストールが正常に完了した場合は、以下が表示される。

VIPcheck、IPAddr2

4.1.2.3. host の設定

＜実行ホスト：ACT/SBY＞

稼働系と待機系で用いるノードのホストを登録する。

この作業は稼働系、待機系両ノードで実施する。

以下のコマンドを実行して、hosts ファイルを開く。

【vi /etc/hosts】[Enter]

編集モードにて、文末に以下を追加する。

(待機系のインターコネクト用 IP アドレス) (待機系のホスト名)

(稼働系のインターコネクト用 IP アドレス) (稼働系のホスト名)

Hosts の設定を有効にするため再起動を行う。

【reboot】[Enter]

4.1.2.4. パスワードの設定

＜実行ホスト：ACT/SBY＞

再起動後、corosync によるサーバ間通信の認証で使用される"hacluster"ユーザのパスワードを設定する。パッケージインストール時に自動的にユーザが作成されているため、パスワード変更のみ行う。パスワードはクラスタを構成するすべてのサーバにて同一とする。

この作業は稼働系、待機系両ノードで実施する。

以下のコマンドを実行して、パスワード変更を行う。

【passwd hacluster】[Enter]

コマンド実行後、以下の文章が表示されるので、新しいパスワードの入力を行う。

Changing password for user hacluster.

New password:

【(新しいパスワード)】[Enter]

[Enter]押下後、確認のため再度パスワードを入力するよう、指示される。

Retype new password:

同じパスワードを入力する。

【(新しいパスワード)】[Enter]

パスワードが同一であれば、以下の文章が表示され、パスワードの変更が完了する。

passwd: all authentication tokens updated successfully.

4.1.2.5. pcsd サービスの設定

＜実行ホスト：ACT/SBY＞

pcsd サービスを起動および有効化する。pcsd は Pacemaker や corosync とは独立したサービスであり、これが起動していないとクラスタ構成時に使用する pcs コマンドが使えない。

この作業は稼働系、待機系の両ノードで実施する。

以下のコマンドを実行して、pcsd サービスの有効化を行う。

【systemctl start pcsd】[Enter]

【systemctl enable pcsd】[Enter]

enable 操作後、以下のメッセージが画面に表示される。

Created symlink from /etc/systemd/system/multi-user.target.wants/pcsd.service to /usr/lib/systemd/system/pcsd.service.

4.1.2.6. pcsd サービス状況の確認

＜実行ホスト：ACT/SBY＞

以下のコマンドを実行して、pcsd のサービス状況を確認する。

この作業は稼働系、待機系の両ノードで実施する。

【systemctl status pcsd】[Enter]

正しく起動できていれば、以下のメッセージが画面に表示される。

```
   Loaded: loaded (/usr/lib/systemd/system/pcsd.service; disabled; vendor preset: disabled)
   Active: active (running) since *** 2016-**-** **:**:** UTC; **s ago
 Main PID: **** (pcsd)
   CGroup: /system.slice/pcsd.service
           ├**** /bin/sh /usr/lib/pcsd/pcsd start
           ├**** /bin/bash -c ulimit -S -c 0 >/dev/null 2>&1 ; /usr/bin/ruby -I/usr/lib/...
           └**** /usr/bin/ruby -I/usr/lib/pcsd /usr/lib/pcsd/ssl.rb
```

4.1.2.7. ノード認証

＜実行ホスト：ACT/SBY＞

以下のコマンドを実行して、"pcs cluster auth"コマンドにより互いのノードの認証を行う。

これにより、corosync によるサーバ間通信が可能となる。

この作業は稼働系、待機系の両ノードで実施する。

【pcs cluster auth (稼働系ノード名) (待機系のノード名) -u hacluster -p (hacluster のパスワード) --force】[Enter]

正しく認証できれば、以下のメッセージが画面に表示される。

(待機系ホスト名 or 稼働系ホスト名): Authorized

(待機系ホスト名 or 稼働系ホスト名): Authorized

4.1.2.8. 初期クラスタ作成

＜実行ホスト：ACT＞

以下のコマンドを実行して、"pcs cluster setup"コマンドにより、初期クラスタの作成を行う。

この作業は稼働系のノードのみで行う。

【pcs cluster setup --name (設定するクラスタ名) (稼働系のノード名) (待機系のノード名)】

正しくクラスタが生成できれば、以下のメッセージが画面に表示される。

Shutting down pacemaker/corosync services...
Redirecting to /bin/systemctl stop pacemaker.service
Redirecting to /bin/systemctl stop corosync.service
Killing any remaining services...
Removing all cluster configuration files...
(稼働系のノード名): Succeeded
(待機系のノード名): Succeeded
Synchronizing pcsd certificates on nodes em-it1-ha-00.novalocal, em-it1-ha-01.novalocal...
(稼働系のノード名): Succeeded
(待機系のノード名): Succeeded
Restaring pcsd on the nodes in order to reload the certificates...
(稼働系のノード名): Succeeded
(待機系のノード名): Succeeded

4.1.2.9. ノード同期

＜実行ホスト：ACT＞

以下のコマンドを実行して、両ノードの同期を開始し、クラスタとして成り立たせる。

この作業は稼働系のノードのみで行う。

【pcs cluster start --all】[Enter]

正しくクラスタが開始できれば、以下のメッセージが画面に表示される。

(稼働系のノード名): Starting Cluster...
(待機系のノード名): Starting Cluster...

4.1.2.10. ノード間通信状況確認

＜実行ホスト：ACT/SBY＞

以下のコマンドを実行して、corosync によるノード間通信の状況を確認する。

この作業は稼働系、待機系、両ノードで行う。

【corosync-cfgtool -s】[Enter]

正しくクラスタが開始している場合、以下のメッセージが画面に表示される。"status"が"active"かつ"no faults"であれば、問題なく通信が行えている。

Printing ring status.
Local node ID (1 or 0)
RING ID 0
id = (稼働系 or 待機系の IP アドレス)
status = ring 0 active with no faults

4.1.2.11. クラスタ状態確認

＜実行ホスト：ACT or SBY＞

以下のコマンドを実行して、クラスタ状態を確認する。
この作業は待機系、稼働系のどちらかで行えば良い。

【pcs status】[Enter]

コマンド実行後、画面にクラスタの状態が表示されるが、表示最上部が以下のように、STONITHについての警告が表示される。

Cluster name: (設定したクラスタ名)
WARNING: no stonith devices and stonith-enabled is not false
Last updated: Thu Oct 13 02:06:57 2016 Last change: Thu Oct 13 02:06:49 2016

今回の環境では STONITH は利用できないため、STONITH の利用を FALSE に設定する。

【pcs property set stonith-enabled=false】[Enter]

最後にもう一度クラスタの状態を確認する。

【pcs status】[Enter]

コマンド実行後、画面に、クラスタの状態が表示される。正しく設定できていれば以下のように表示される。

Cluster name: emcluster
Last updated: WDW MMM DD HH:MM:DD YYYY Last change: WDW MMM DD HH:MM:DD YYYY by root via cibadmin on (稼働系のノード名)
Stack: corosync
Current DC: (待機系のノード名) (version 1.1.14-1.el7-70404b0) - partition with quorum
2 nodes and 0 resources configured

Online: [(稼働系のノード名) (待機系のノード名)]

Full list of resources:

PCSD Status:

(稼働系のノード名): Online
(待機系のノード名): Online

Daemon Status:
corosync: active/disabled
pacemaker: active/disabled
pcsd: active/enabled

4.1.2.12. pacemaker/corosync サービス有効化

＜実行ホスト：ACT/SBY＞

以下のコマンドを実行して、再起動後も corosync と pacemaker の自動起動を有効にするため、両サービスを enable にする。

この作業は待機系、稼動系の両ノードで行う。

【systemctl enable pacemaker】[Enter]

【systemctl enable corosync】[Enter]

4.1.3. Python ライブラリのインストール、及び設定

4.1.3.1. setup-tools のインストール

＜実行ホスト：ACT/SBY＞

以下のコマンドを実行し、setup-tools をインストールする。

【cd ~/setup/em/installer/】[Enter]

【tar zxvf setuptools-28.6.0.tar.gz】[Enter]

【cd setuptools-28.6.0】[Enter]

【python setup.py install】[Enter]

4.1.3.2. PIP のインストール

＜実行ホスト：ACT/SBY＞

以下のコマンドを実行し、PIP をインストールする。

【cd ~/setup/em/installer/】[Enter]

【tar zxvf pip-8.1.2.tar.gz】[Enter]

【cd pip-8.1.2】[Enter]

【python setup.py install】 [Enter]

4.1.3.3. PIP のインストール確認

＜実行ホスト：ACT/SBY＞

以下のコマンドを実行して、バージョンを確認する。

【pip --version】[Enter]

インストールが正常に完了した場合は、以下が表示される。

pip 8.1.2 from /usr/lib/python2.7/site-packages/pip-8.1.2-py2.7.egg (python 2.7)

4.1.3.4. Python ライブラリのインストール

＜実行ホスト：ACT/SBY＞

以下のコマンドを実行し、パッケージを展開する。

【cd ~/setup/em/installer/】[Enter]

【tar xvf whl_package.tar】[Enter]

以下のコマンドを実行し、展開したパッケージのフォルダへ移動する。

【cd whl_package】[Enter]

以下のコマンドを実行し、ライブラリのインストールを行う。

【pip install --use-wheel --no-index --find-links=. netconf-0.4.1-py2-none-any.whl】[Enter]

【pip install --use-wheel --no-index --find-links=. ncclient-0.5.2-py2-none-any.whl】[Enter]

【pip install --use-wheel --no-index --find-links=. oslo.db-4.13.3-py2.py3-none-any.whl】[Enter]

【pip install --use-wheel --no-index --find-links=. xmltodict-0.10.2-py2-none-any.whl】[Enter]

【pip install --use-wheel --no-index --find-links=. psycopg2-2.6.2-cp27-cp27mu-linux_x86_64.whl】[Enter]

4.1.3.5. Python ライブラリのインストール確認

＜実行ホスト：ACT/SBY＞

以下のコマンドを実行して、インストールされているライブラリのリストと各ライブラリのバージョンを表示する。

【pip list】[Enter]

各ライブラリのインストールが正常に完了した場合、リスト中に以下の内容が表示される。

alembic (0.8.8)
Babel (2.3.4)
cffi (1.8.3)
cryptography (1.5.2)
debtcollector (1.8.0)
decorator (3.4.0)
enum34 (1.1.6)
funcsigs (1.0.2)
idna (2.1)
ipaddress (1.0.17)
iso8601 (0.1.11)
lxml (3.6.4)
Mako (1.0.4)
MarkupSafe (0.23)
monotonic (1.2)
ncclient (0.5.2)
netaddr (0.7.18)
netconf (0.4.1)
netifaces (0.10.5)
oslo.config (3.17.0)
oslo.context (2.9.0)
oslo.db (4.13.3)
oslo.i18n (3.9.0)
oslo.utils (3.16.0)
paramiko (2.0.2)
pbr (1.10.0)
positional (1.1.1)
psycopg2 (2.6.2)
pyasn1 (0.1.9)
pycparser (2.14)
pyparsing (2.1.9)
python-editor (1.0.1)
pytz (2016.7)

rfc3986 (0.4.1)
setuptools (28.6.0)
six (1.9.0)
SQLAlchemy (1.0.15)
sqlalchemy-migrate (0.10.0)
sqlparse (0.2.1)
sshutil (0.9.7)
stevedore (1.17.1)
Tempita (0.5.2)
wrapt (1.10.8)
xmltodict (0.10.2)

4.1.4. NTP のインストール、及び設定

4.1.4.1. インストール

＜実行ホスト：ACT/SBY＞

以下のコマンドを実行し、ntp をインストールする。

【cd ~/setup/em/installer/ntp.v4.2/】[Enter]

【rpm -ivh autogen-libopts-5.18-5.el7.x86_64.rpm】[Enter]

【rpm -ivh ntpdate-4.2.6p5-22.el7.centos.x86_64.rpm】[Enter]

【rpm -ivh ntp-4.2.6p5-22.el7.centos.x86_64.rpm】[Enter]

4.1.4.2. drift ファイルの作成

＜実行ホスト：ACT/SBY＞

以下のコマンドを実行し、空の drift ファイルを作成する。

【touch /var/lib/ntp/drift】[Enter]

4.1.4.3. NTP 設定ファイルの修正

＜実行ホスト：ACT/SBY＞

ルートユーザにて、NTP 設定ファイル(/etc/ntp.conf)に以下の内容(網掛け部分)を追加する。

【vi /etc/ntp.conf】[Enter]

～（略）～

restrict default nomodify notrap nopeer noquery

restrict default ignore

～（略）～

#restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap

restrict xxx.xxx.xxx.xxx noquery nomodify
server xxx.xxx.xxx.xxx iburst

～（略）～

4.1.4.4. NTP サーバとの同期

＜実行ホスト：ACT/SBY＞

以下のコマンドを実行し、NTP が起動していないことを確認する。

【systemctl status ntpd.service】[Enter]

<NTP 未起動時の出力>

～（略）～

Active: inactive (dead)

～（略）～

<NTP 起動時の出力>

～（略）～

Active: active (running)

～（略）～

NTP が起動している場合、以下のコマンドを実行して NTP を停止する。

【systemctl stop ntpd.service】[Enter]

NTP サーバ(IP アドレス：xxx.xxx.xxx.xxx)と時刻を合わせる。

【ntpdate xxx.xxx.xxx.xxx】[Enter]

4.1.4.5. NTP の再起動

＜実行ホスト：ACT/SBY＞

以下のコマンドを実行し、NTP を再起動する。

【systemctl restart ntpd.service】[Enter]

【systemctl enable ntpd.service】[Enter]

以下のコマンドを実行し、NTP サーバとの同期を確認する。

【ntpq -p】[Enter]

<同期成功時出力例>

```
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
*xxx.xxx.xxx.xxx LOCAL(0)        11 u   55   64  377    0.130   -0.017   0.017
```

4.1.5. PostgreSQL のインストール、及び設定

4.1.5.1. インストール

＜実行ホスト：DB/ACT/SBY＞

以下のコマンドを実行し、postgresql をインストールする。

【cd ~/setup/em/installer/postgresql.v9.3.13】[Enter]

【rpm -ivh libxslt-1.1.28-5.el7.x86_64.rpm】[Enter]

【rpm -ivh uuid-1.6.2-26.el7.x86_64.rpm】[Enter]

【rpm -ivh postgresql93-libs-9.3.13-1PGDG.rhel7.x86_64.rpm】[Enter]

【rpm -ivh postgresql93-9.3.13-1PGDG.rhel7.x86_64.rpm】[Enter]

【rpm -ivh postgresql93-server-9.3.13-1PGDG.rhel7.x86_64.rpm】[Enter]

【rpm -ivh postgresql93-devel-9.3.13-1PGDG.rhel7.x86_64.rpm】[Enter]

【rpm -ivh postgresql93-contrib-9.3.13-1PGDG.rhel7.x86_64.rpm】[Enter]

4.1.5.2. PostgreSQL のコンフィグ修正

＜実行ホスト：DB＞

以下のように内容を修正する。

【vi /var/lib/pgsql/.bash_profile】[Enter] （網掛け部分を修正および追記する）

PGDATA=/usr/local/pgsql/9.3/data

export PGDATA

export PATH=$PATH:/usr/pgsql-9.3/bin

【source /var/lib/pgsql/.bash_profile】[Enter]

4.1.5.3. データベース作成、及び権限付与

＜実行ホスト：DB＞

以下のコマンドを実行して、データベースのインストール先フォルダを作成する。

【cd /usr/local/】[Enter]

【mkdir -pm 777 /usr/local/pgsql/9.3】[Enter]

【chown -R postgres:postgres pgsql】[Enter]

postgres ユーザで以下のコマンドを実行してデータベースを作成する。

・データベースフォルダの作成

【su - postgres】[Enter]

【cd /usr/local/pgsql/9.3/】[Enter]

【mkdir -m 700 data】[Enter]

・データベースの初期化

【initdb --encoding=UTF8 --no-locale --pgdata=/usr/local/pgsql/9.3/data --auth=ident】[Enter]

コマンド実行後、以下のような出力がされるので確認する。

The files belonging to this database system will be owned by user "postgres".

This user must also own the server process.

The database cluster will be initialized with locale "C".
The default text search configuration will be set to "english".

Data page checksums are disabled.

fixing permissions on existing directory /usr/local/pgsql/9.3/data ... ok
creating subdirectories ... ok
selecting default max_connections ... 100
selecting default shared_buffers ... 128MB
creating configuration files ... ok
creating template1 database in /usr/local/pgsql/9.3/data/base/1 ... ok
initializing pg_authid ... ok
initializing dependencies ... ok
creating system views ... ok
loading system objects' descriptions ... ok
creating collations ... ok
creating conversions ... ok
creating dictionaries ... ok
setting privileges on built-in objects ... ok
creating information schema ... ok
loading PL/pgSQL server-side language ... ok
vacuuming database template1 ... ok
copying template1 to template0 ... ok
copying template1 to postgres ... ok
syncing data to disk ... ok

Success. You can now start the database server using:

    postgres -D /usr/local/pgsql/9.3/data
or
    pg_ctl -D /usr/local/pgsql/9.3/data -l logfile start

・データベースの作成

【pg_ctl -D /usr/local/pgsql/9.3/data -l logfile start】[Enter]

実行後、”server starting”と画面に表示。

【psql -c "alter user postgres with password ""】[Enter]

実行後、”ALTER ROLE”と画面に表示。

【psql】[Enter]

【create role root login createdb password ';】[Enter]

実行後、”CREATE ROLE”と画面に表示。

【¥q】[Enter]

【pg_ctl -D /usr/local/pgsql/9.3/data -l logfile stop】[Enter]

実行後、画面には以下のメッセージが表示される。

waiting for server to shut down.... done

server stopped

【exit】[Enter]

ルートユーザで以下のコマンドを実行する。

【chkconfig postgresql-9.3 on】[Enter]

【systemctl daemon-reload】[Enter]

4.1.5.4. データベースのコンフィグ修正

＜実行ホスト：DB＞

以下のように内容を修正する。

【vi /usr/local/pgsql/9.3/data/postgresql.conf】[Enter]

修正前

～（略）～

#listen_addresses = 'localhost'

#port = 5432

～（略）～

修正後

～（略）～

listen_addresses = '*'

port = 5432

～（略）～

以下のように内容を修正する。（網掛け部分は許可するサーバのセグメントを記載）

【vi /usr/local/pgsql/9.3/data/pg_hba.conf】[Enter]

修正前

～（略）～

TYPE　DATABASE　　USER　　ADDRESS　　METHOD

"local" is for Unix domain socket connections only

local　all　　all　　peer

IPv4 local connections:

```
host    all             all             127.0.0.1/32            ident
# IPv6 local connections:
host    all             all             ::1/128                 ident
# Allow replication connections from localhost, by a user with the
# replication privilege.
#local   replication     postgres                                peer
#host    replication     postgres        127.0.0.1/32            ident
#host    replication     postgres        ::1/128                 ident
```

修正後

```
～（略）～
# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
#local   all             all                                     peer
# IPv4 local connections:
#host    all             all             127.0.0.1/32            ident
# IPv6 local connections:
#host    all             all             ::1/128                 ident
# Allow replication connections from localhost, by a user with the
# replication privilege.
#local   replication     postgres                                peer
#host    replication     postgres        127.0.0.1/32            ident
#host    replication     postgres        ::1/128                 ident

local   all             postgres                                peer
local   all             all                                     trust
host    all             all             192.168.53.0/24         trust
host    all             all             127.0.0.1/32            trust
```

以下のように内容を修正する。（網掛け部分を修正する）

【vi /usr/lib/systemd/system/postgresql-9.3.service】[Enter]

```
～（略）～
# Location of database directory
Environment=PGDATA=/usr/local/pgsql/9.3/data/
～（略）～
```

4.1.5.5. データベースの再起動

＜実行ホスト：DB＞

以下のコマンドを postgres ユーザで実行する。

【systemctl daemon-reload】[Enter]

【systemctl start postgresql-9.3】[Enter]

以下のコマンドにより、postgres の起動を確認する。

【systemctl status postgresql-9.3】[Enter]

コマンド実行後、以下の文章が出力されるが、以下の部分を確認する。

Active: active (running)　← **running** となっていること

CGroup: /system.slice/postgresql-9.3.service

tq***** /usr/pgsql-9.3/bin/postgres -D /usr/local/pgsql/9.3/data　← 今回作成したフォルダが指定されていること

4.2.　EM モジュールインストール

以下、記載の$EM_HOME はユーザ指定の任意の場所とする。

4.2.1.　ライブラリ配置

＜実行ホスト：ACT/SBY＞

以下のコマンドを実行して付属品内のライブラリファイルを配置する。

【mkdir -p $EM_HOME/lib】[Enter]

【cp -r ~/setup/em/EmModule/lib/* $EM_HOME/lib/】[Enter]

以下の手順で配置したファイルに実行権を付与する。

【cd $EM_HOME/lib】[Enter]

【chmod 777 MsfEmMain.pyc】[Enter]

4.2.2.　コンフィグ配置

＜実行ホスト：ACT/SBY＞

以下のコマンドを実行して付属品内のコンフィグファイルを配置する。

【mkdir -p $EM_HOME/conf】[Enter]

【cp -r ~/setup/em/EmModule/conf/* $EM_HOME/conf/】[Enter]

以下のコマンドにて、EM モジュール設定ファイルを修正する。

【vi $EM_HOME/conf/[ファイル名]】[Enter]

ファイル名には、

・conf_sys_common.conf

・conf_scenario.conf

・conf_if_process.conf

・conf_driver.conf

がはいる。

※インストールしたサーバに合わせて、コンフィグ値を修正して下さい。

修正内容は「別紙_コンフィグ仕様書. xlsx」を参照とする。

4.2.3.　起動シェルの配置

＜実行ホスト：ACT/SBY＞

以下のコマンドを実行して付属品内の起動シェルファイルを配置する。

【mkdir -p $EM_HOME/bin】[Enter]

【cp -r ~/setup/em/EmModule/bin/* $EM_HOME/bin/】[Enter]

以下の手順で配置したファイルに実行権を付与する。

【cd $EM_HOME/bin】[Enter]

【chmod 777 em_ctl.sh】[Enter]

4.2.4.　ログフォルダの作成

＜実行ホスト：ACT/SBY＞

以下のコマンドを実行してログのためのフォルダを作成する。

【mkdir -p $EM_HOME/logs/em/log】[Enter]

4.2.5.　スキーマ作成

＜実行ホスト：DB＞

以下のコマンドにて、スキーマを作成する。

【createdb　「スキーマ名」】[Enter]

以下のコマンドにて、上で作成したスキーマにテーブルを作成する。

【cd ~/setup/em/script/】[Enter]

【psql　「スキーマ名」　＜create_table.sql】[Enter]

4.2.6.　リソースエージェント用監視アカウントの設定

＜実行ホスト：ACT/SBY＞

起動スクリプトに EM 監視用の IP アドレス、接続用ユーザ名、パスワードを設定する。

以下のコマンドにて、起動スクリプトの環境定義を修正する。

【cd $EM_HOME/bin】[Enter]

【vi em_ctl.sh】[Enter]

以下の部分を編集する。（網掛け部分を修正する）

＜修正前＞

```
# 環境定義
# 監視用モジュールで EM にアクセスするための情報を設定する
#EM のログインユーザ名
USERNAME="root"
#EM のログインパスワード
PASSWORD=""
#EM の待受 IP アドレス
IPV4="127.0.0.1"
#EM の待受ポート
PORT=830
```

＜修正後＞

```
# 環境定義
# 監視用モジュールで EM にアクセスするための情報を設定する
#EM のログインユーザ名
USERNAME = “(EM の conf_if_process.conf の Account に設定したユーザ名)”
#EM のログインパスワード
PASSWORD = “(EM の conf_if_process.conf の Password に設定したパスワード)”
#EM の待受 IP アドレス
```

IPV4 =”(EM の conf_if_process.conf の Netconf_server_address に設定した IP アドレス)”

PORT = (EM の conf_if_process.conf の Port_number に設定したポート番号)

編集完了後保存する。

4.2.7. 環境変数の設定

＜実行ホスト：ACT/SBY＞

以下のコマンドで環境変数を登録するためのファイルを開く。

【vi /root/.bash_profile】[Enter]

ファイルの末尾に以下を追記する。

```
EMTOPPATH="(環境に応じて変更。 EM のトップパスまでを記述。)"
EMLIBTOPPATH="$EMTOPPATH/lib"
PYTHONPATH="$EMLIBTOPPATH:$EMLIBTOPPATH/CommonDriver"
PYTHONPATH="$PYTHONPATH:$EMLIBTOPPATH/Config"
PYTHONPATH="$PYTHONPATH:$EMLIBTOPPATH/DB"
PYTHONPATH="$PYTHONPATH:$EMLIBTOPPATH/DriverUtility"
PYTHONPATH="$PYTHONPATH:$EMLIBTOPPATH/NetconfServer"
PYTHONPATH="$PYTHONPATH:$EMLIBTOPPATH/OrderflowControl"
PYTHONPATH="$PYTHONPATH:$EMLIBTOPPATH/Protocol"
PYTHONPATH="$PYTHONPATH:$EMLIBTOPPATH/Scenario"
PYTHONPATH="$PYTHONPATH:$EMLIBTOPPATH/SeparateDriver"
PYTHONPATH="$PYTHONPATH:$EMLIBTOPPATH/SystemUtility"
PYTHONPATH="$PYTHONPATH:/lib/ocf/resource.d/heartbeat"
export PYTHONPATH
EM_LIB_PATH="$EMTOPPATH/lib/"
export EM_LIB_PATH
EM_CONF_PATH="$EMTOPPATH/conf/"
export EM_CONF_PATH
```

以下の環境変数 PATH を以下のように修正する。

【修正前】

PATH=$PATH:$HOME/bin

【修正後】（網掛け部分を追加する）

PATH=$PATH:$HOME/bin:$EMTOPPATH/bin:/usr/bin/python

追記完了後ファイルを保存して、閉じる。

4.3. リソースエージェントの登録と設定

4.3.1. リソースエージェントの配置

＜実行ホスト：ACT/SBY＞

EM インストールフォルダの起動シェルフォルダより、EM のリソースエージェントを既定のリソースエージェントフォルダへとコピーする。

【cp $EM_HOME/bin/em /lib/ocf/resource.d/heartbeat/】[Enter]

その後コピーした EM リソースエージェントに実行権を付与する。

【cd /lib/ocf/resource.d/heartbeat/】[Enter]

【chmod 775 em】[Enter]

4.3.2. crm ファイルの作成

＜実行ホスト：ACT＞

ファイルを配置するワーク用のフォルダ（以下、ワークフォルダ）を作成する。（設定が完了した時点で削除する）

【mkdir ~/setup】[Enter]

付属品のリソースエージェントの設定を記した ra_config.xlsx ファイルを編集し、必要な情報を更新した上で、csv ファイルに変換し、ワークフォルダへと配置する。

csv ファイルを配置したフォルダ上で次のコマンドを実行し、csv ファイルをリソースエージェントに登録するための crm ファイルへと変換する。

【pm_crmgen -o crm_conf.crm (配置した csv ファイル名).csv】[Enter]

正しく変換できた場合なにも表示されないが、csv ファイルに誤りがあった場合、修正箇所が表示される。

4.3.3. crm ファイルの投入

＜実行ホスト：ACT＞

次のコマンドでリソースエージェントの登録を行う。

【crm configure load update crm_conf.crm】[Enter]

正しく投入できた場合、何も表示されないため次項において確認を行う。（※VIPcheck については規定よりも短い秒数が設定されていると表示がでるが問題ないので、そのまま続けて良い）

設定に重大な誤りがあった場合、誤りのある部分と警告が表示され、続けて投入するかどうかを Y/N で聞かれる。この表記がでる場合、設定ファイルに誤りがあるため、【N】[Enter]として回答する。

4.3.4. 投入結果の確認

＜実行ホスト：ACT or SBY＞

次のコマンドで投入したリソースエージェントの動作状況を確認する。

【pcs status】[Enter]

正しく投入できた場合、以下のように表示される。

Cluster name: emcluster
Last updated: WDW MMM DD HH:MM:SS YYYY Last change: WDW MMM DD HH:MM:SS YYYY by root via cibadmin on (稼動系のノード名 or 待機系のノード名)
Stack: corosync
Current DC: (待機系のノード名 or 待機系のノード名) (version 1.1.14-1.el7-70404b0) - partition with quorum
2 nodes and 5 resources configured

Online: [(稼動系のノード名) (待機系のノード名)]

Full list of resources:

Resource Group: grpEM
vipCheck (ocf::heartbeat:VIPcheck): Started (稼動系のノード名)
prmIP (ocf::heartbeat:IPaddr2): Started (稼動系のノード名)
prmEM (ocf::heartbeat:em): Started (稼動系のノード名)

Clone Set: clnDiskd [prmDiskd]
Started: [(稼動系のノード名)(待機系のノード名)]

PCSD Status:
(稼動系のノード名): Online
(待機系のノード名): Online

Daemon Status:
corosync: active/enabled
pacemaker: active/enabled
pcsd: active/enabled