



SIOS Protection Suite for Linux v9.4

Chef Support Document

2019年10月

本書およびその内容は SIOS Technology Corp. (旧称 SteelEye® Technology, Inc.) の所有物であり、許可なき使用および複製は禁止されています。SIOS Technology Corp. は本書の内容に関していかなる保証も行いません。また、事前の通知なく本書を改訂し、本書に記載された製品に変更を加える権利を保有しています。SIOS Technology Corp. は、新しい技術、コンポーネント、およびソフトウェアが利用可能になるのに合わせて製品を改善することを方針としています。そのため、SIOS Technology Corp. は事前の通知なく仕様を変更する権利を保有します。

LifeKeeper、SteelEye、および SteelEye DataKeeper は SIOS Technology Corp. の登録商標です。

本書で使用されるその他のブランド名および製品名は、識別のみを目的として使用されており、各社の商標が含まれています。

目次

1. はじめに.....	4
2. 概要	5
2-2 Chef をサポートすることで得られるメリット	6
2-3 サポート構成	7
3. Chef サポートの利用手順.....	10
3-1 準備	10
3-2 既存クラスタ情報の抽出.....	11
3-3 既存クラスタ情報の Chef ファイルへの変換	13
3-4 新規クラスタ生成の準備.....	16
3-5 attribute ファイルの編集.....	20
3-6 新規クラスタの生成.....	24
4. 終わりに.....	26

1. はじめに

LifeKeeper v9.0.0 より、Chef がサポートされるようになりました。

これにより、LifeKeeper for Linux(以下 LifeKeeper)の検証環境に構築したリソース階層を本番環境へ容易に移行できるようになります。

本資料は、Chef で LifeKeeper の既存リソース再構築をするための要件や基本操作を解説するものです。

なお、本資料は LifeKeeper、Chef に関して知識を有している方を対象とし、LifeKeeper や Chef の基本的な設定や操作、技術的な詳細情報を解説するものではありません。

本構成の前提となる LifeKeeper や Chef に関する用語、操作、技術情報等につきましては、関連のマニュアル等をご確認ください。

2. 概要

2-1 Chef とは

従来、手順書を使い手動で行っていたサーバなどの管理を、Ruby コードで記述し、そのコードを元に自動実行する手段の一つです。

<https://www.chef.io/chef/>

従来インフラを構築する際には手順書、チェックリストを用意し、それに従い、1ステップ毎に手入力で作業していくことが一般的でした。

しかし、このやり方は、

- ・ サーバが多くなると構築作業に時間と手間がかかる
- ・ 人手を介在するためミスが発生しやすくなる
- ・ 異なる環境毎に手順書を作成しなければならず、手順書の管理が複雑になる等の問題があります。

ここで、手順をコードで管理することにより、従来の問題を解決することが出来ます。

- ・ サーバが多くても、コードを適用するだけで構築できるので手間と時間が大幅に短縮できる
- ・ 人手を介さないため、人為的ミスが発生しない
- ・ コードを流用することで異なる環境でも、少ない手間でインフラ構築ができる
- ・ 冪等性（べきとうせい、何度実行しても結果は同じになること）を確保できる

このような考えは Infrastructure as Code と呼ばれ、Chef の他にも Ansible、CFEngine、Puppet などの製品があります。

LifeKeeper v9.0.0 では、このうち数多くの実績があり、コードを Ruby で記述できるなど、メンテナンスも容易な Chef をサポートすることになりました。

2-2 Chef をサポートすることで得られるメリット

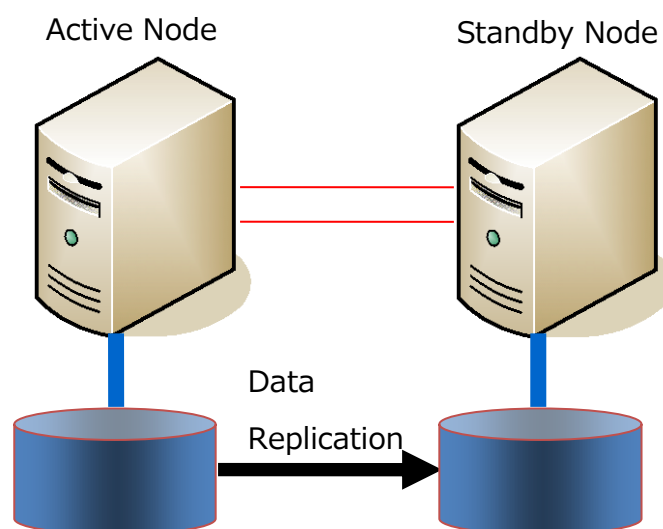
Chef をサポートすることによる、お客様の得られるメリットは、以下の様になります。

- 既存クラスタから Chef attribute を抽出することが可能になり、クラスタの複製が容易になる
- 手作業で行っていた、サーバ構築やリソース作成が自動でできるため、エンジニアの負担を軽減可能
- 人手を介することで発生する人為的ミスを軽減することができる
- ハードウェア入れ替えなどによるリソース再構築をコード実行だけで簡単に実行できる
- 検証環境から本番環境への移行が容易にできる

2-3 サポート構成

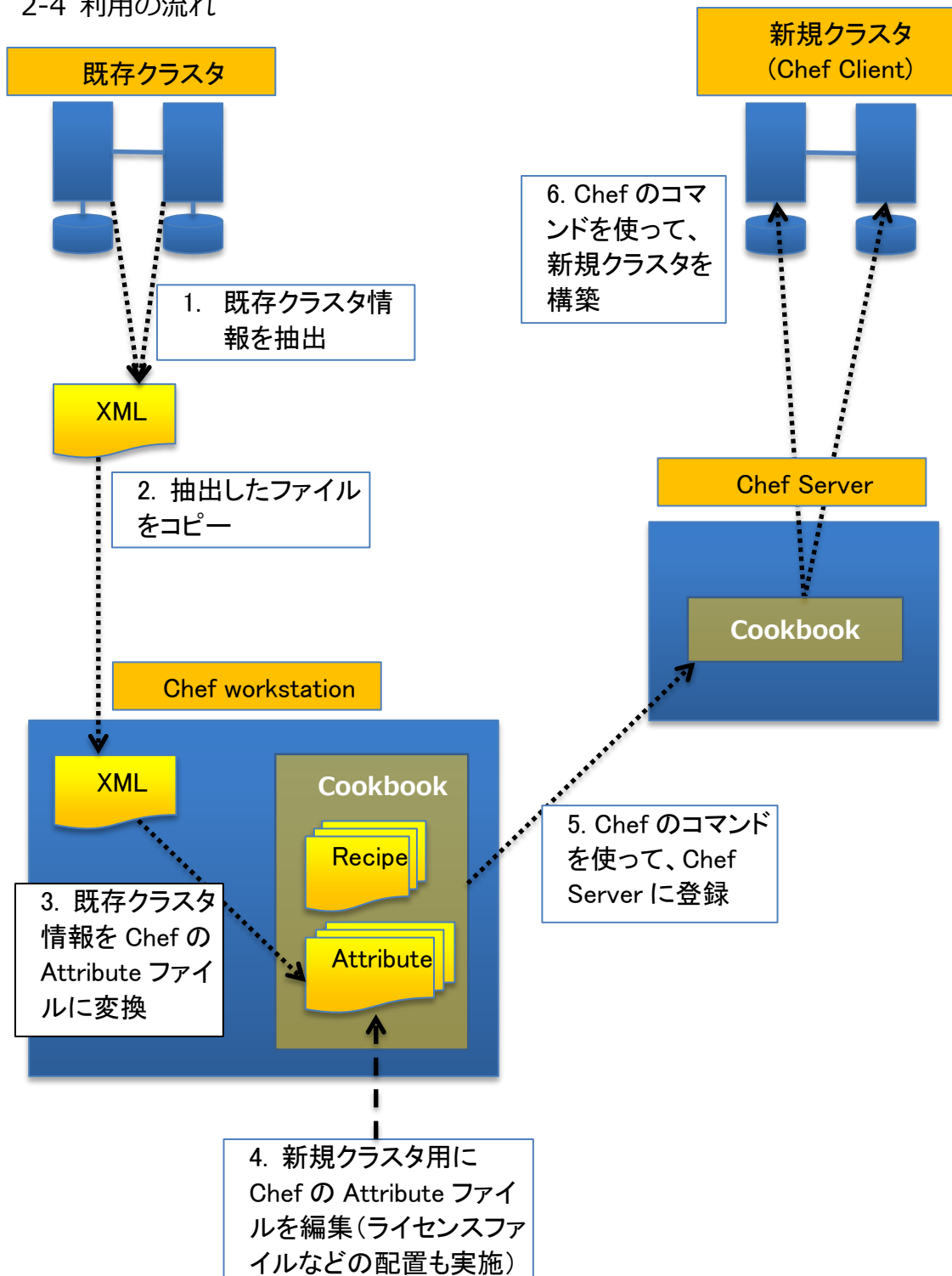
Chef サポートの対象となる LifeKeeper の構成は、以下のとおりです。

- OS :
 - Red Hat Enterprise Linux version 5, 6, 7
 - Community ENTerprise Operating System (CentOS) version 5, 6, 7
 - Oracle Linux version 5, 6, 7
- 構成
 - 2 ノードの DataReplication 構成
 - 対象 ARK: IP, File System, Apache, MySQL, PostgreSQL



サポートする Chef の cookbook は、LifeKeeper インストール、コミュニケーションパス登録、リソース作成の 3 種類になります。

2-4 利用の流れ



-
- (1)元になる既存クラスタから xml 形式でリソース構成を抽出します。
 - (2)抽出した XML ファイルを Chef workstation にコピーします。
 - (3)抽出した XML ファイルを専用スクリプトで Chef 用のパラメータファイル「attribute」に変換します。attribute はコミュニケーションパス用とリソース用に 2 つ生成されます。
 - (4)変換した 2 種類の attribute ファイルはそれぞれ、Chef workstation の cookbook 配下のコミュニケーションパス、及びリソース用の attribute フォルダにコピーします。コピーした attribute ファイルについて、アトリビュートに埋め込まれたホスト名や IP アドレスなどのパラメータを各環境に合わせて修正します。また、インストールで使用する rpm ファイル、LifeKeeper のライセンスファイルも同様に cookbook 配下にコピーします。
 - (5)Chef workstation で Knife コマンドにより cookbook 配下のファイルを Chef Server に登録します。
 - (6)新規クラスタの各ノードで chef-client を実行することで、リソースを再構築することが出来ます。

3. Chef サポートの利用手順

3-1 準備

3-1-1 既存クラスタ

既存クラスタでは LifeKeeper v9.0.0 以降がインストールされていれば、特に準備は不要です。

3-1-2 Chef Server

Chef Server を用意してください

3-1-3 Chef workstation

◆Chef サポート用ファイルのコピー

LifeKeeper インストールイメージファイルを Chef workstation でマウントし、attribute 変換スクリプトをコピーします。

詳しい手順は以下の様になります。

- ① LifeKeeper インストールイメージファイル sps.img を Linux 環境の/mnt などにマウントする。

例

```
# mount sps.img -t iso9660 -o loop /mnt
```

- ② Chef サポートファイルを確認する。

```
$ ls /mnt/Chef/  
TRANS.TBL  exp2chef.pl  nodes/  recipe/
```

- ③ 変換スクリプトを適切なディレクトリにコピーする。

例 ~/ 配下に Chef ディレクトリを作成してコピーする。

```
$ mkdir ~/Chef  
$ cp /mnt/Chef/exp2chef.pl ~/Chef
```

◆exp2chef.pl 実行環境のセットアップ

exp2chef.pl を実行するために Chef workstation に Perl5 及び, XML::Simple モジュールが必要です。ディストリビューションで配布されているものか、CPAN から入手してインストールしてください。

CentOS 6 の場合の例 :

```
# yum install perl-XML-Simple
```

3-2 既存クラスタ情報の抽出

既存クラスタ上で下記コマンドを実行することでクラスタのリソース情報が出力されますのでコピーペーストやリダイレクションを使用してファイルに保存します。この時、すべてのリソースをサービス中にしてください。停止しているリソースは復元時にエラーになります。

```
/opt/LifeKeeper/lkadm/bin/lkexportxml
```

サポート外のリソースがある場合はエラーになりますのでご注意ください。

◆使用例

ここでは、出力ファイルを resource.xml とします。ディレクトリは root 配下とします。

```
# /opt/LifeKeeper/lkadm/bin/lkexportxml>/root/resource.xml
```

出力結果例

```
<?xml version='1.0'?>
<lifekeeper>
  <node name="node1">
    <commpath remote="node2">
      <baudrate>0</baudrate>
      <device>192.168.100.1/192.168.100.2</device>
      <ipaddress>192.168.100.1</ipaddress>
      <priority>1</priority>
      <remoteaddress>192.168.100.2</remoteaddress>
      <type>TCP</type>
    </commpath>
      :
      <中略>
      :
    <instance order="3" tag="/DATA2">
      <ID>/DATA2</ID>
      <app>gen</app>
      <info>
        <altblock>0</altblock>
        <perm>rw,barrier=0</perm>
        <type>ext4</type>
      </info>
      <init>SEC_ISP</init>
      <state>OSU</state>
      <switchback>INTELLIGENT</switchback>
      <typ>filesystem</typ>
    </instance>
  </node>
</lifekeeper>
```

ここで作成したファイルは「3-1-3 Chef workstation」で attribute 変換スクリプト exp2chef.pl をコピーした Linux 環境にコピーしておきます。

3-3 既存クラスタ情報の Chef ファイルへの変換

「3-1-3 Chef workstation」で exp2chef.pl をコピーした Linux 環境にログインします。

引数としてクラスタ構成 XML ファイルを指定し、スクリプトを実行します。

```
~/Chef/exp2chef.pl <クラスタ情報 XML ファイル>
```

例 クラスタ情報 XML ファイルは~/Chef/resource.xml とした場合

```
$ ~/Chef/exp2chef.pl ~/Chef/resource.xml
```

スクリプトを実行すると入力の XML ファイルと同じディレクトリにコミュニケーションパス、リソース用の 2 種類の attribute ファイルが生成されます。

ファイル名はクラスタ情報 XML ファイルにそれぞれ comm、res のサフィックス+拡張子.rb が付加された形式で生成されます。

例

クラスタ情報 XML ファイル	…	resource.xml
コミュニケーションパス用 attribute ファイル	…	resource.comm.rb
リソース用 attribute ファイル	…	resource.res.rb

attribute 変換出力ファイルの例を図示します

コミュニケーションパス用 attribute ファイル

```
default['LKROOT']=" /opt/LifeKeeper"

    default['node1']['commpath']['0'] = {
      "priority" => "1",
      "baudrate" => "0",
      "remoteaddress" => "192.168.100.2",
      "device" =>
"192.168.100.1/192.168.100.2",
      "remote" => "node2",
      "type" => "TCP",
      "ipaddress" => "192.168.100.1",
    }

    default['node1']['commpath']['1'] = {
      "priority" => "2",
      "baudrate" => "0",
      "remoteaddress" => "192.168.0.2",
      "device" => "192.168.0.1/192.168.0.2",
      "remote" => "node2",
      "type" => "TCP",
      "ipaddress" => "192.168.0.1",
    }

    default['node2']['commpath']['0'] = {
      .....<中略>.....

    default['node2']['commpath']['1'] = {
      "priority" => "2",
      "baudrate" => "0",
      "remoteaddress" => "192.168.0.1",
      "device" => "192.168.0.2/192.168.0.1",
      "remote" => "node1",
      "type" => "TCP",
      "ipaddress" => "192.168.0.2",
    }
```

リソース用 attribute ファイル

```
default['LKROOT']=" /opt/LifeKeeper"

default['node1']['dependency']['0'] = {
  "parent" => "/DATA1",
  "child" => "datarep-DATA1",
}

default['node1']['dependency']['1'] = {
  "parent" => "/DATA2",
  "child" => "datarep-DATA2",
}

default['node1']['equivalency']['datarep-
DATA1'] = {
  "priority" => "1",
  "rtag" => "datarep-DATA1",
  "tag" => "datarep-DATA1",
  "type" => "SHARED",
  "remote" => "node2",
  "rpriority" => "10",
}

default['node1']['equivalency']['/DATA1']
= {
  "priority" => "1",
  ..... <中略> .....
  "perm" => "rw,barrier=0",
  "app" => "gen",
  "init" => "SEC_ISP",
  "state" => "OSU",
  "order" => "3",
  "tag" => "/DATA2",
  "typ" => "filesys",
  "switchback" => "INTELLIGENT",
}
```

3-4 新規クラスタ生成の準備

下記の Chef の cookbook を用意します。

- LifeKeeper インストール用
- コミュニケーションパス登録用
- リソース作成用

cookbook 名はそれぞれ lkinstall,commpath,resources とします。

cookbook の作成は knife コマンドを使います。

作成した cookbook に LifeKeeper パッケージ,ライセンスキーファイル、Chef recipe/attribute ファイルなど必要ファイルをコピーします。

コピーが必要なファイル一覧を次ページにリストアップします。

全てのファイルを指定されたディレクトリにコピーして下さい。

■ LifeKeeper 本体 rpm ファイル

コピー先<cookbook path>/lkinstall/files/default

対象ファイル	備考
共通	
/mnt/common/steeleye-perl-*.rpm	
/mnt/common/steeleye-openssl-*.rpm	
/mnt/common/steeleye-openssl-perl-*.rpm	
/mnt/common/steeleye-libgpg-error-*.rpm	
/mnt/common/steeleye-libgrypt-*.rpm	
/mnt/common/steeleye-libcurl-*.rpm	
/mnt/common/steeleye-curl-*.rpm	
/mnt/common/steeleye-readline-*.rpm	
/mnt/common/steeleye-gnutls-*.rpm	
/mnt/common/steeleye-gnutls-utils-*.rpm	
/mnt/common/steeleye-libxml2-*.rpm	
/mnt/common/steeleye-libxml2-static-*.rpm	
/mnt/common/steeleye-pcre-*.rpm	
/mnt/common/steeleye-perl-addons-*.rpm	
/mnt/common/steeleye-lighttpd-*.rpm	
/mnt/common/steeleye-lighttpd-fastcgi-*.rpm	
/mnt/common/steeleye-lkapi-*.rpm	
/mnt/common/steeleye-lkapi-client-*.rpm	
/mnt/common/steeleye-pdksh-*.rpm	
/mnt/common/steeleye-runit-*.rpm	
/mnt/core/steeleye-lk*.rpm	
ライセンスキーファイル	製品には含まれていません

対象ファイル	備考
各 OS 固有	
RedHat Enterprise Linux	
/mnt/RHAS/HADR-RHAS-2.6.32-all.x86_64*.rpm	6.x のみ
/mnt/RHAS/HADR-RHAS-3.10.0-all.x86_64*.rpm	7.x のみ
/mnt/RHAS/steeleye-lkRHAS-*.rpm	
CentOS	
/mnt/CentOS/HADR-CentOS-2.6.32-all.x86_64*.rpm	6.x のみ
/mnt/CentOS/HADR-CentOS-3.10.0-all.x86_64*.rpm	7.x のみ
/mnt/CentOS/steeleye-lkCentOS-*.rpm	
Oracle Linux	
/mnt/OEL/HADR-OEL-2.6.32-all.x86_64*.rpm	6.x のみ(除 UEK)
/mnt/OEL/HADR-OEL-3.10.0-all.x86_64*.rpm	7.x のみ(除 UEK)
/mnt/OEL/steeleye-lkOEL-*.rpm	

■ 必須パッケージ

コピー先<cookbook path>/lkinstall/files/default

対象ファイル	備考
/mnt/java/jre-*-linux-x64.rpm	

■ ARK 用

コピー先<cookbook path>/lkinstall/files/default

対象ファイル	備考
/mnt/kits/steeleye-lkAPA-*.noarch.rpm	Apache ARK
/mnt/kits/steeleye-lkDR-*.noarch.rpm	DataKeeper
/mnt/HADR-generic-*.rpm	
/mnt/kits/steeleye-lkPGSQL-*.noarch.rpm	PostgreSQL ARK
/mnt/kits/steeleye-lkSQL-*.noarch.rpm	MySQL ARK

※インストールする ARK のみコピーしてください

■ Chef サポートファイル

対象ファイルコピー元	コピー先
/mnt/Chef/recipe/lkinstall.rb	<cookbook path>/lkinstall/recipe/default.rb
/mnt/Chef/recipe/commppath.rb	<cookbook path>/commppath/recipe/default.rb
/mnt/Chef/recipe/resources.rb	<cookbook path>/resources/recipe/default.rb
/mnt/Chef/attribute/lkinstall.rb	<cookbook path>/lkinstall/attribute/default.rb

■ exp2chef.pl で生成した Chef attribute ファイル

生成したファイル	コピー先
comm path attribute	<cookbook path>/commppath/attribute/default.rb
resource attribute	<cookbook path>/resources/attribute/default.rb

3-5 attribute ファイルの編集

attribute ファイルにはクラスタ構成に依存した情報が格納されているので、環境に合わせて編集してください。Chef でクラスタを構築する際には IP アドレス、ノード名は変更される事が殆どですから、IP アドレス、ノード名周りの修正は必須となります。

以下は各 attribute パラメータの説明一覧です。

コミュニケーションパスの attribute

パラメータ	説明	編集
node name	default[""]で始まるカッコ内の文字列	必須
priority	優先度	
baudrate	ボーレート~TTY 接続でのみ	
remoteaddress	対向ノードの IP アドレス	必須
Device	自 IP アドレス/リモート IP アドレス	必須
Remote	対向ノード名	必須
Type	コミュニケーションパス接続形式 TTY/TCP	
IpAddress	自 IP アドレス	必須

resource の attribute

- dependency セクション

パラメータ	説明	編集
Parent	親リソース リソースタグ	必須※
Child	子リソース リソースタグ	必須

※タグ名を変更しなければ不要

・ equivalency セクション

パラメータ	説明	編集
priority	優先度	
rtag	リモートタグ	必須※
tag	自タグ	必須※
type		
remote	リモートのノード名	必須
rpriority	リモートの優先度	

※タグ名を変更しなければ不要です

・ instance セクション

インスタンス部分はパラメータが多いので、各インスタンス共通部とインスタンス独自の主要パラメータを抜粋してリストアップします。

各インスタンス共通

パラメータ	説明	編集
ID	リソースの ID	
typ	リソースタイプ	
tag	リソースタグ	
switchback	スイッチバックタイプ	
state	リソースの状態	

・ パラメータ typ=ip

パラメータ	説明	編集
primach	ノード名	必須
priif	NIC	
mask	net mask	
ipaddr	IP リソースの IP アドレス	必須

・ パラメータ typ=netraid

パラメータ	説明	編集
ID	デバイス ID (/dev/sdb1 など)	
num	md の番号	
async	同期モード	
mountpoint	マウントポイント	
bitmap	bitmap ファイル場所	
ipaddr	自ノードとリモートの IP アドレス	必須※

※何れかのコミュニケーションパスと一致させること

・ パラメータ typ=apache

パラメータ	説明	編集
root	httpd.conf のあるディレクトリ	
path	httpd デーモンプログラムの場所	

・ パラメータ typ=pgsql

パラメータ	説明	編集
osexec	Postgres 実行ファイルのパス	
datadir	保護対象の Postgres データディレクトリ	
port	クライアントからの通信用 Port	
socket	クライアントからの通信用 Socket へのフルパス	
clientexec	Postgres 実行ファイル pg_ctl のパス	
dbuser	Postgres データ管理者ユーザー名	
execpath	Postgres 実行ファイル psql のパス	
logfile	Postgres ログファイル用のパス	
osuser	os user id	

・ パラメータ typ=mysql

パラメータ	説明	編集
insno	Protection Instance Number	
bindir	MySQL バイナリの場所	
confdir	MySQL 設定ファイル (my.cnf) の場所のフルパス名 (ファイル名は除く)	
datadir	データベースのデータディレクトリ	

■ linstall

各ノードにインストールするライセンスファイルを記述します

例

```
default[node]['license'] = ['example1.lic','example2.lic']
```

- node ライセンスをインストールするノード名を指定します
- license ライセンスファイルの指定
- example1.lic, example2...ライセンスファイルをコンマ区切りで列挙します

3-6 新規クラスタの生成

以下は一般的な手順です。詳細は Chef のドキュメントを参照してください。

①server へ cookbook をアップロード

knife cookbook upload で cookbook をサーバにアップロードします。

②run list の登録

knife node run_list add で run list へ recipe を登録します。

③chef-client のインストール

knife bootstrap ノード名

④recipe の実行

chef-client を実行します。②で登録した runlist の recipe が実行されます

※ご注意

- **特定環境での DR 構成について**

特定の環境 (vSphere 等、IDE ディスクエミュレーションを使用する仮想環境など) では LifeKeeper インストール後に device_pattern ファイルに/dev/sdx*(sdx は保護対象となる device 名です。お客様の環境によって異なります)を追加する必要があります。

詳しくは[テクニカルドキュメンテーションの LifeKeeper > トラブルシューティング > 既知の問題と制限](#) 及び [DataKeeper > トラブルシューティング](#)をご確認ください。

- **IP リソース構成時の注意事項**

IP リソースはブロードキャスト ping を発行することで状態を監視しています。ネットワーク上にブロードキャストブロードキャスト ping に応答する機器が同一ネットワーク内に存在しない場合、リソース作成前に設定の変更が必要になります。詳しくは [IP Recovery Kit 管理ガイドの設定 > IP Configuration Properties の確認および編集](#)をご確認ください。

③と④を各ノードに対して実行します

この時アクティブなリソースを持っているノードから実行してください

逆にするとリソースの登録に失敗します

⑤いずれかのノードで LifeKeeper の GUI を起動し、各リソースが意図した通り出来ているかを確認します。

⑥リソース作成後は DataKeeper リソースを除いて、全て OSU になっています。

ソースノードで in service を実行し、サービスを開始します。

4. 終わりに

本構成における留意点

特にありません

既知の問題点とトラブルシューティング

- xml エクスポート

- ・ エラーが出て Postfix,Samba リソースがエクスポートできません。
「2-3 サポート対象」で定義している ARK 以外はサポートしていません。
- ・ 3 ノード構成だとエラーになる
サポートは 2 ノードまでです。

- attribute エクスポート

attribute エクスポートに関して既知の問題点とトラブルシューティングはありません。

- Chef 実行

- ・ アクティブなノードから実行しないと正常に登録できません
アクティブーアクティブ構成をとる場合は、一旦どちらかのノードだけをアクティブに切り替えてください