SIOS Protection Suite for Linux v9.1.1

Chef Support Document

Jan 2017

To maintain the quality of our publications, we welcome your comments on the accuracy, clarity, organization, and value of this document.

Address correspondence to:
ip@us.sios.com

Table of Contents

1. Introduction

Starting with v9.0.0 LifeKeeper for Linux provides support for Chef.
This allows you to transfer resource hierarchies constructed in a verification environment of LifeKeeper for Linux (Below is called LifeKeeper) to a production environment easily.
This documentation includes requirements and basic operations to reconstruct existing resources of LifeKeeper with Chef.

This document also assumes you have appropriate knowledge of LifeKeeper and Chef. Basic configurations and information on detailed technical matters are not included.
Please refer to the manual for terms, operations, and technical information of LifeKeeper and Chef.

2. Overview

2-1 What is Chef?

Managing servers are traditionally performed manually by using a procedure. Chef is one of the measures to write Ruby Code, and perform the operation automatically based on the code.

https://www.chef.io/chef/

A procedure and a check list are generally required to construct infrastructure, and each step of it are performed manually.
However, there are some issues:

- Construction works take labor and time when there are many servers.
- Artificial errors often occur due to manual intervention.
- Procedures must be created per environment, and managing it tends to be complicated.

By managing above procedures by code, the above issues can be resolved.

- Constructing infrastructure only by applying code can be remarkably shortened your labor and time.
- Artificial errors do not occur since it can be constructed without manual intervention.
- Infrastructure can be constructed in different environments by the same code with using less labor and time.
- Ensuring idempotency (To give the same result every time it executed).

This kind of idea is called Infrastructure as Code. There are some products such as Ansible, CFEngine, and Puppet other than Chef.
In LifeKeeper v9.0.0, Chef is supported since it has been used successfully. Also, the code can be written in Ruby, and its maintenance is easy.

2-2 Customer’s merits by supporting Chef

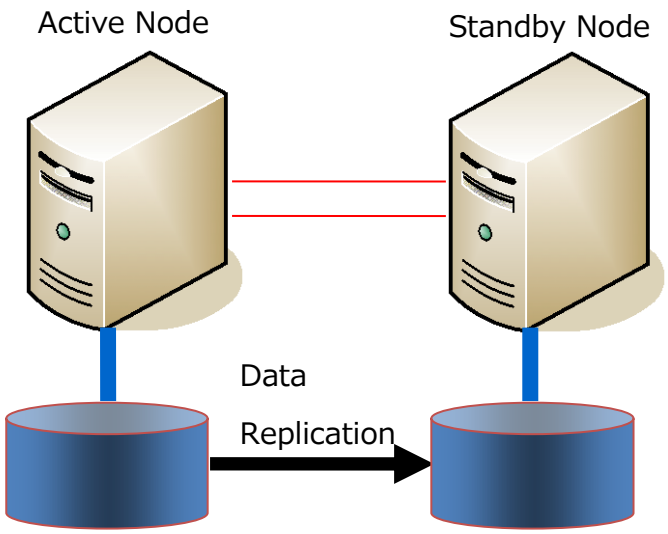
Here are customer’s merits by supporting Chef:

- Extracting Chef Attribute from the existing clusters becomes available, and cluster replications can be easy.
- Burdens of engineers can be reduced since a server construction or a resource creation that was usually performed manually is performed automatically.
- Artificial errors that occur due to manual intervention can be reduced.
- A resource construction due to replacing hardwares can be performed only by executing code.
- A verification environment can be easily transferred to a production environment.

2-3 Supported OS and Recovery Kit

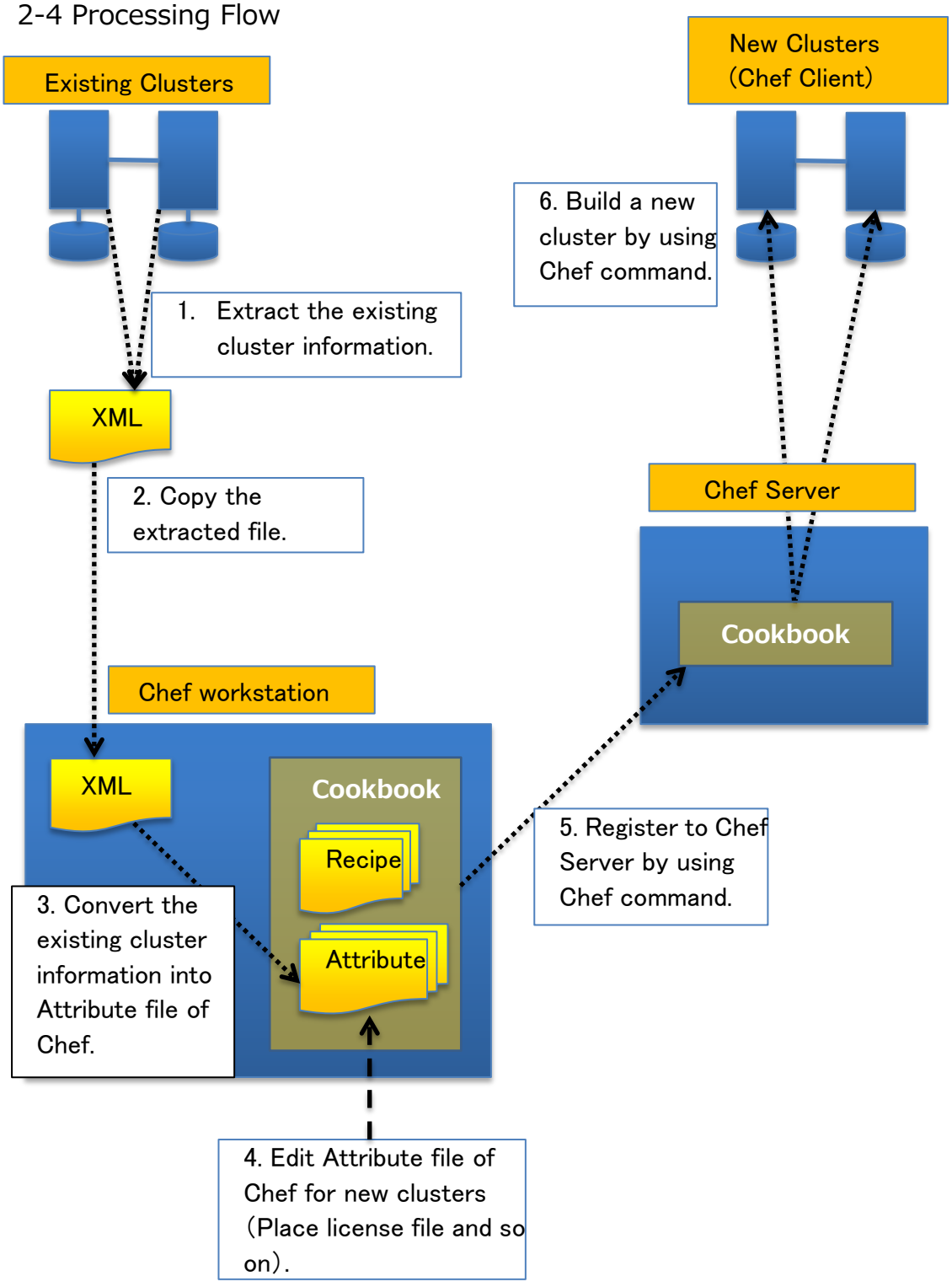
Here are supported OS and Recovery Kit of the supported configurations by LifeKeeper.

- OS :
  - Red Hat Enterprise Linux version 5, 6, 7
  - Community ENTerprise Operating System (CentOS) version 5, 6, 7
  - Oracle Linux version 5, 6, 7
- Configuration
  - 2-node Data Replication
  - Arks to be applied: IP, FileSystem, Apache, MySQL, PostgreSQL



LifeKeeper installation, a communication path registration, and a resource creation are the cookbooks of Chef to be supported.

2-4 Processing Flow



(1) Extract the resource configuration with the xml format from the existing cluster.
(2) Copy the extracted XML file to Chef workstation.
(3) Convert the extracted XML file into the Chef Parameter file “attribute” by using a dedicated script. Two “attribute” are generated. One is for communication paths, and the other is for resources.
(4) Copy the two converted attribute files to each attribute folder under cookbook of Chef workstation. Modify parameters such as the embedded host names or IP addresses matching each environment. Also, copy the rpm file that is used for installation and the license file of LifeKeeper under cookbook.
(5) Resister the files under cookbook to Chef Server in Chef workstation by using the Knife command.
(6) By performing chef-client on each node of new cluster, a resource can be reconstructed.

3. Chef Procedure of Chef support

3-1 Preparation

3-1-1 Existing clusters

For existing clusters, preparation is not necessary if LifeKeeper v.9.0.0 and later is already installed.

3-1-2 Chef Server

Prepare for Chef Server.

3-1-3 Chef workstation

◆Copy a file for Chef support

Mount the LifeKeeper installation image file in Chef workstation, and copy the attribute conversion script.

See below for detailed procedure.

① Mount the LifeKeeper installation image file “sps.img” to such as /mnt of Linux environment.

Example

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

② Check the Chef support file.

③ $ ls /mnt/Chef/

```
TRANS.TBL  exp2chef.pl  nodes/  recipe/
```

④ Copy the conversion script to the appropriate directory.

Example: Create and copy Chef Directory under ~/.

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

◆Setting up an execution environment for exp2chef.pl

To perform exp2chef.pl, Perl5 and XML::Simple are required for Chef workstation. Acquire and install it from distribution or CPAN.

Example of CentOS 6:

yum install perl-XML-Simple

3-2 Extracting the existing cluster information

By performing the command below on the existing cluster, the resource information of cluster is output. Save it in a file using copy & paste or redirection.

When performing the above procedure, bring all resources In Service. Resources in Out of Service cause an error for restoration.

/opt/LifeKeeper/lkadm/bin/lkexportxml

Note that unsupported resources cause an error.

◆Usage examples

Suppose that the output file is resource.xml, and its directory is under root.

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

Examples of output result

```
<?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>
                  :
                  < Partially omitted >
                  :
    <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>filesys</typ>
    </instance>
  </node>
</lifekeeper>
```

The file created here is copied to the Linux environment where the attribute conversion script “exp2chef.pl” is copied in ”3-1-3 Chef workstation”.

3-3 Converting the existing cluster information into Chef file

Log in to the Linux environment where the exp2chef.pl is copied in “3-1-3 Chef workstation”.

Specify the cluster configuration XML file as an argument, and perform a script.

~/Chef/exp2chef.pl <XML file of cluster information >

Example: The XML file of cluster information is ~/Chef/resource.xml

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

When performing a script, two kinds of attribute files are generated in the same directory of the input XML file. One is for communication paths, and the other is for resources.

File names are generated for the XML file of cluster information in the format that added comm for communication paths and suffix+extension.rb for resources.

Examples

XML file of cluster information	…	resource.xml
Attribute file for communication paths	…	resource.comm.rb
Attribute file for resources	…	resource.res.rb

Here are examples of the attribute conversion output file.

Attribute file for communication paths

```
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'] = {

     ……<Partially omitted>……

     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",
     }
```

Attribute file for resources

```
      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",
      ……<Partially omitted>……
       "perm" => "rw,barrier=0",
       "app" => "gen",
       "init" => "SEC_ISP",
       "state" => "OSU",
       "order" => "3",
       "tag" => "/DATA2",
       "typ" => "filesys",
       "switchback" => "INTELLIGENT",
      }
```

3-4 Preparing for a new cluster generation

Prepare for the Chef cookbooks below.

- For installation of LifeKeeper
- For a registration of communication path
- For a resource creation

Cookbook names are clkinstall, commpath, and resources for each cookbook.

Cookbook is created by using the knife command.

Copy the LifeKeeper package, License key file, and Chef recipe/attribute file as required to the created cookbook.

File list that required copy is on next page.

Copy all files to the specified directory.

■LifeKeeper rpm file

Copy to <cookbook path>/lkinstall/files/default

Target file	Note
Common	
/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-libgcrypt-*.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	
License Key File	Not included in this product.

Target file	Note
Each OS specific	
RedHat Enterprise Linux	
/mnt/RHAS/HADR-RHAS-2.6.32-all.x86_64*.rpm	6.x only
/mnt/RHAS/HADR-RHAS-3.10.0-all.x86_64*.rpm	7.x only
/mnt/RHAS/steeleye-lkRHAS-*.rpm	
CentOS	
/mnt/CentOS/HADR-CentOS-2.6.32-all.x86_64*.rpm	6.x only
/mnt/CentOS/HADR-CentOS-3.10.0-all.x86_64*.rpm	7.x only
/mnt/CentOS/steeleye-lkCentOS-*.rpm	
Oracle Linux	
/mnt/OEL/HADR-OEL-2.6.32-all.x86_64*.rpm	6.x only (Except UEK)
/mnt/OEL/HADR-OEL-3.10.0-all.x86_64*.rpm	7.x only (Except UEK)
/mnt/OEL/steeleye-lkOEL-*.rpm	

■Required package

Copy to <cookbook path>/lkinstall/files/default

Target file	Note
/mnt/java/jre-*-linux-x64.rpm	

■For ARK

Copy to <cookbook path>/lkinstall/files/default

Target file	Note
/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

Note: Copy only the Ark to be installed.

■Chef Support file

Source file	Copy destination
/mnt/Chef/recipe/lkinstall.rb	<cookbook path>/lkinstall/recipe/default.rb
/mnt/Chef/recipe/commpath.rb	<cookbook path>/commpath/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

■Chef attribute files that generated by exp2chef.pl

Generated file	Copy to
comm path attribute	<cookbook path>/commpath/attribute/default.rb
resource attribute	<cookbook path>/resources/attribute/default.rb

3-5 Editing an attribute file

Edit an attribute file matching each environment. Information depending on its cluster configuration is stored in it. When building a cluster in Chef, it is required to change IP addresses and node names since they are usually changed.

Here is a list of each attribute parameter.

Attribute for communication paths

Parameter	Description	Edit
node name	Text string that starting with default[']	Required
priority	Priority	
baudrate	Only with baud rate to TTY connection	
remoteaddress	IP address of the opposite node	Required
Device	Own IP address/Remote IP address	Required
Remote	Name of the opposite node	Required
Type	Type of communication path connection TTY/TCP	
Ipaddress	Own IP address	Required

Attribute for resources

・Dependency section

Parameter	Description	Edit
Parent	Resource tag of Parent resource	Required*
Child	Child tag of Child resource	Required

*This parameter is not required if not changing tag name.

・Equivalency section

Parameter	Description	Edit
priority	Priority	
rtag	Remote tag	Required*
tag	Own tag	Required*
type		
remote	Node name of Remote	Required
rpriority	Priority of Remote	

*This parameter is not required if not changing tag name.

・Instance section

Main parameters of common parts for each instance and original instances are extracted and listed up since there are so many.

Common for each instance

Parameter	Description	Edit
ID	Resource ID	
typ	Resource type	
tag	Resource tag	
switchback	Switchback type	
state	Resource status	

・Parameter typ=ip

Parameter	Description	Edit
primach	Node name	Required
priif	NIC	
mask	net mask	
ipaddr	IP address of IP resource	Required

・Parameter typ=netraid

Parameter	Description	Edit
ID	Device ID（/dev/sdb, etc）	
num	md numbber	
async	Sync mode	
mountpoint	Mount point	
bitmap	bitmap file location	
ipaddr	IP address of own node and remote	Required*

*This parameter is required to match one of the communication paths.

・Parameter typ=apache

Parameter	Description	Edit
root	Directory where the httpd.conf at	
path	Location of httpd daemon program	

・Parameter typ=pgsql

Parameter	Description	Edit
osexe	Path of the Postgres execution file	
datadir	The protected Postgres data directory	
port	Port for communication from client	
socket	Specify the full path to Socket for communication from client	
clientexe	The pg_ctl path for Postgres execution file	
dbuser	User name of Postgres data administrator	
exepath	The psql path of Postgres execution file	
logfile	Path for the Postgres log file	
osuser	os user id	

・Parameter typ=mysql

Parameter	Description	Edit
insno	Protection Instance Number	
bindir	Location of MySQL binary	
confdir	The full path name (Except file name) where MySQL configuration file (my.cnf) is located	
datadir	Data directory of database	

■ lkinstall

License file that install to each node is described.

Example

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

- node Specify a node name to install a license.
- license Specify a license file.
- exsample1.lic, example2… A comma separated license files are enumerated.

3-6 Generating a new cluster

Here are general procedures. For detailed information, see the Chef manual.

① Upload a cookbook to a server.

Upload a cookbook to a server by the knife cookbook upload.

② Register run list

Register a recipe to run list by using the knife node run_list add.

③ Install chef-client

knife bootstrap node name

④ Perform a recipe

Perform the chef-client. The recipe registered at step ② of the run list is performed.

Note

- **DR configuration in specific environments**

In specific environments, to edit the DEVNAME/device_pattern file is required to configure DR before a resource creation.

For detailed information, see **Technical Documentation > LifeKeeper > Troubleshooting > Known Issues and Restrictions,** or **DataKeeper > Troubleshooting.**

- **Precaution of IP resource configuration**

IP resources monitor a status by issuing broadcast pings. If the device responded to broadcast pings do not exist in the same network, to change the configuration is required before a resource creation. For detailed information, See **IP Recovery Kit Administration Guide >Viewing and Editing IP Configuration Properties.**

Perform ③ and ④ for each node.
Perform these steps from the node that has an active resource.
A resource registration will be failed when performing these with reversing the order.

⑤ Start the GUI of LifeKeeper in one of the nodes. Confirm if each resource is generated as expected.

⑥ After creating a resource, all are OSU except for the DataKeeper resource. Perform In Service in source node, and start a service.

4. Conclusion

Points of Attention in this configuration.

None

Known issues and Troubleshooting

●xml export

・The Postfix and Samba resources cannot be exported with an error.
The arks defined in “2-3 Supported OS and Recovery Kit” are only supported.

・An error occurs with the 3–node configuration.
LifeKeeper can support up to the 2–node configuration.

●Attribute export

None

●Chef execution

・It can be registered normally only when executing from an active node.
When requiring the active-active configuration, switch one of the nodes once.