DRBD9 LINSTOR 構築手順書

サイオステクノロジー株式会社

バージョン 0.1.10 2019/01/10

目次

1. 変更履歴

1.1. v0.1.10 2019/01/10

- drbdtop でビルドする golang をダウンロードに変更

1.2. v0.1.9 2019/01/08

- サイオステクノロジーに変更

1.3. v0.1.8 2018/10/02

- LINSTORの設定ファイルの保存箇所の仕様変更のため、初期化の項目を修正

1.4. v0.1.7 2018/09/18

- linstor-server (0.6.4) build,rpm: generate noarch packages

1.5. v0.1.6 2018/09/12

- linstor-server (0.6.0) の driver 名の位置変更に対応 # linstor storage-pool create lvm drbdnode sp_name

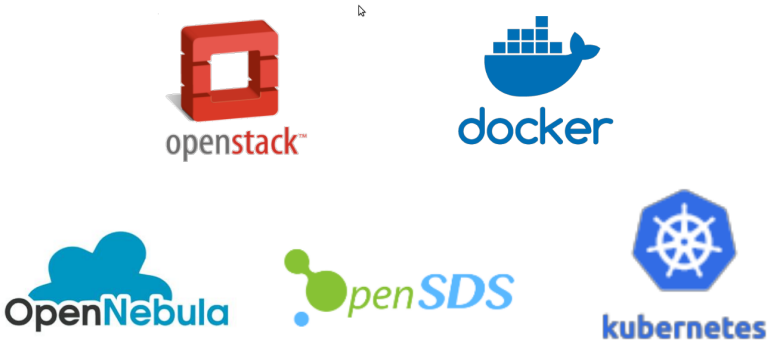
1.6. v0.1.5 2018/08/21

- linstor-server (0.5.0) の分割に対応(linstor-controller, linstor-satellite)

2. はじめに

本書は、DRBD9のSDS環境をLINSTORを使って円滑に構築するための手順書になります。LINSTORとはDRBDマネージ(drbdmanage)に代わるLinuxシステム上のストレージ構成管理のシステムです。複数のノード上に存在するLVM論理ボリューム、またはZFSのZVOLを管理します。DRBDを使用してノード間のボリュームのレプリケーションを行い、ブロックデバイスをユーザーやアプリケーションに提供します。

LINSTORで管理された冗長性のあるブロックデバイスは、OpenStackやKubernetes等のいろいろなOSSプロダクトから利用することができます。



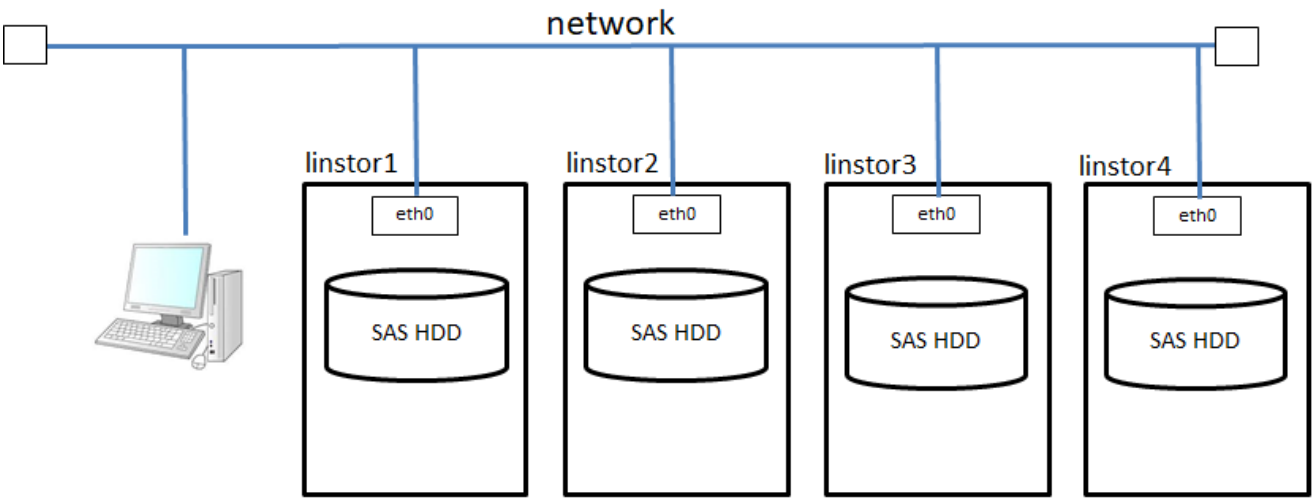
3. 本書で使用するソフトウェアの仕様

3.1. 仕様

本書にて使用したソフトウェア情報を以下に記載します。

ソフトウェア名	バージョン
OS	Centos7.6
カーネル	3.10.0-957.1.3
kmod-drbd	9.0.16_3.10.0_957.1.3-1
drbd-utils	9.7.0-1
drbdtop	0.2.1-1
linstor-controller	0.7.5-1
linstor-satellite	0.7.5-1
linstor-common	0.7.5-1
linstor-client	0.7.3-1
python-linstor	0.7.3-1

3.2. 構成図



4. ネットワーク関連パラメーター

以下のホスト名やパラメータを使用します。

パラメータ	一号機	二号機	三号機	四号機
ホスト名	linstor1	linstor2	linstor3	linstor4
NIC(eth0)	10.30.10.1/16	10.30.10.2/16	10.30.10.3/16	10.30.10.4/16

5. OSインストール後の設定

本書は、対象マシーンにそれぞれCentos7.5がインストールされていると想定していますので、OSのインストール方法は割愛いたします。それぞれ各ノードにて設定を行います。

5.1. システムアップデート

Centos7の最新パッケージに更新します。

```
# yum -y update
```

ほとんどの場合、カーネルも更新されるので、アップデートが終わったら一旦サーバを再起動します。

5.2. SELinuxの無効化

この構築手順では、SELinuxは無効にします。
configファイルを修正してOSを再起動します。

```
# getenforce      ; 現在の状態を確認します。
# sed -i s/SELINUX=enforceing/SELINUX=disabled/g /etc/selinux/config
# reboot
# getenforce      ; 変更後の状態を確認します。
```

5.3. ファイアウォールの停止

本書ではfirewalldを止めて設定を行います。

```
# systemctl stop firewalld  #各ノードにて実行
```

5.4. /etc/hostsの作成

DNSサーバにアクセスできなくても相互に通信できるよう、linstor1,linstor2,linstor3,linstor4のIPアドレスを/etc/hostsに登録します。各ノードの/etc/hostsファイルに設定します。

```
# vi /etc/hosts
10.30.10.1  linstor1
10.30.10.2  linstor2
10.30.10.3  linstor3
10.30.10.4  linstor4
```

5.5. SSHホスト鍵の作成

サーバ間では、root権限で相互にパスワードなしでログインできると、作業効率と精度が高まります。

```
# ssh-keygen -t rsa    #各ノードにて実行
```

ノードlinstor1でssh鍵の交換を実行します。

```
# ssh-copy-id linstor2
# ssh-copy-id linstor3
# ssh-copy-id linstor4
```

※ノードlinstor1での実行が終わったら、linstor2、linstor3とノードを変えて、すべての組み合わせの鍵交換を行なって下さい。

5.6. SSHのログイン確認

パスワード無しでsshできるか確認します。

```
# ssh root@10.30.10.1
# ssh root@10.30.10.2
# ssh root@10.30.10.3
# ssh root@10.30.10.4
※各ノードにて実行します。
```

5.7. DRBD用データ領域の作成

fdiskコマンドを実行して、作成を行います。

```
# fdisk /dev/sda
コマンド (m  でヘルプ):n
Select (default p):p
パーティション番号 :ENTER
最初 sector:ENTER
Last sector, +sectors or +size{K,M,G}:ENTER
コマンド (m でヘルプ):t
パーティション番号 :ENTER
Hex code (type L to list all codes):15
コマンド (m  でヘルプ):w
コマンド (m でヘルプ):q

# shutdown -r now
※各ノードにて実行します。
```

6. LINBITリポジトリからパッケージをインストール

Linux-HAクラスタ・スタックサポートを契約されている方は、LINBIT社のリポジトリに登録されたバイナリーパッケージを使ってインストールができます。
※このセクションは、すべてのノードでまったく同様に実行する必要があります。

6.1. yumリポジトリ修正

LINBITクラスタ・スタックサポートを構成するソフトウェアと同名のソフトウェアをCentosのリポジトリから誤ってダウンロードすると、様々な想定外の動作不良を引き起こす可能性があります。このような事態を避けるために、/etc/yum.repos.d/CentOS-Base.repoのすべてのセクションに次のexclude文を追記してください。

```
exclude=drbd* kmod-drbd* libq* linstor*
```

6.2. LINBITクラスタ・スタック用リポジトリ定義ファイル

LINBITクラスタ・スタックをインストールするために、次の内容で/etc/yum.repos.d/linbit.repoを作成してください。<hash>はLINBITクラスタ・スタックサポートの契約ユーザに提供されるアクセスキーです。

```
[linbit.repo]
name=LINBIT repo
baseurl=https://packages.linbit.com/<hash>/yum/rhel7/drbd-9.0/x86_64
gpgcheck=0
```

6.3. 追加パッケージのインストール

LINBITクラスタ・スタックのパッケージを追加インストールします。

```
# yum -y install drbd kmod-drbd drbdtop linstor-controller linstor-satellite linstor-
client  ;各ノードにて実行
```

lvmのパッケージを追加インストールします。

```
# yum -y install lvm2    ;各ノードにて実行
```

7. パッケージを自分でビルド

LINBIT社のリポジトリからパッケージをインストールする場合はこのセクションを飛ばしてください。

最新の LINSTOR は DRBD9 の最新環境で動作確認してますので、DRBD カーネル、drbd-utils もソースからビルドして使用します。

7.1. ビルドマシンの準備

CentOS7 のビルドマシンを用意しビルドに必要なパッケージを追加します。

```
# yum -y groupinstall "Development Tools"
# yum -y install kernel-devel libxslt docbook-style-xsl help2man git
```

LINSTOR のビルドに必要なパッケージもインストールしておきます。

```
# yum -y install java-1.8.0-openjdk-devel protobuf-compiler python-setuptools wget
```

7.2. DRBD kernel

まずは DRBD9 カーネルをビルドします。

```
# mkdir -p ~/rpmbuild/{BUILD,BUILDROOT,RPMS,SOURCES,SPECS,SRPMS}
# git clone git://github.com/LINBIT/drbd-9.0.git
# cd drbd-9.0
# make tarball
# make kmp-rpm
```

ここで

```
SORRY, kernel makefile not found. You need to tell me a correct KDIR!
```

というエラーがでたら、Kernel source がある正しいパスを与えます。例えば

```
export KDIR=/usr/src/kernels/3.10.0-862.3.2.el7.x86_64
```

で与え、make を再度実行します。最新の kernel に update していないとこのエラーが出る可能性があります。

```
# make kmp-rpm
```

成功すると ~/rpmbuild/RPMS/x86_64/ に

```
drbd-kernel-debuginfo-*.el7.x86_64.rpm
kmod-drbd-9.0.*.el7.x86_64.rpm
```

が作成されます。

7.3. drbd-utils

引き続き drbd-utils をビルドします。

```
# git clone git://github.com/LINBIT/drbd-utils.git
# cd drbd-utils
# ./autogen.sh
# ./configure --prefix=/usr --localstatedir=/var --sysconfdir=/etc
# make tarball
# cp drbd-utils-*.tar.gz ~/rpmbuild/SOURCES
# ./configure --enable-spec
# rpmbuild -bb drbd.spec --without sbinsymlinks --without heartbeat
```

成功すると ~/rpmbuild/RPMS/x86_64/ に

```
drbd-*.el7.x86_64.rpm
drbd-bash-completion-*.el7.x86_64.rpm
drbd-debuginfo-*.el7.x86_64.rpm
drbd-kernel-debuginfo-*.el7.x86_64.rpm
drbd-man-ja-*.el7.x86_64.rpm
drbd-pacemaker-*.el7.x86_64.rpm
drbd-udev-*.el7.x86_64.rpm
drbd-utils-*.el7.x86_64.rpm
drbd-xen-*.el7.x86_64.rpm
kmod-drbd-9.0.*.el7.x86_64.rpm
```

が作成されます。

7.4. LINSTOR

次に linstor-server, linstor-client, python-linstor をそれぞれビルドします。

7.4.1. linstor-server

```
# wget https://services.gradle.org/distributions/gradle-4.7-bin.zip
# unzip -o gradle-4.7-bin.zip
# export PATH=$PWD/gradle-4.7/bin:$PATH

# git clone --recursive git://github.com/LINBIT/linstor-server.git
# linstor-server
# make tarball
# cp linstor-server-*.tar.gz ~/rpmbuild/SOURCES
# rpmbuild -bb linstor.spec
```

成功すると ~/rpmbuild/RPMS/noarch に次のファイルが作成されます。

```
linstor-controller-*.rpm
linstor-satellite-*.rpm
linstor-common-*.rpm
```

7.4.2. linstor-client

```
# git clone --recursive git://github.com/LINBIT/linstor-client.git
# cd linstor-client
# make rpm
# cp dist/linstor-client-*noarch.rpm ~/rpmbuild/RPMS
```

7.4.3. python-linstor

```
# git clone --recursive git://github.com/LINBIT/linstor-api-py.git
# cd linstor-api-py
# make rpm
# cp dist/python-linstor-*noarch.rpm ~/rpmbuild/RPMS
```

成功すると ~/rpmbuild/RPMS に次のファイルが作成されます。

```
linstor-client-*.noarch.rpm
python-linstor-*.noarch.rpm
```

7.5. drbdtop

最後に golang をダウンロードし、drbdtop をビルドします。

```
# cd $HOME
# wget https://dl.google.com/go/go1.10.4.linux-amd64.tar.gz
# tar xzf go1.10.4.linux-amd64.tar.gz
# PATH=$HOME/go/bin:$PATH
# export GOPATH=$HOME

# git clone https://github.com/LINBIT/drbdtop
# cd drbdtop
# go get github.com/linbit/drbdtop
# go get gopkg.in/alecthomas/kingpin.v2
# make
```

トップディレクトリに drbdtop のバイナリが作成されます。

7.6. ビルドしたパッケージのインストール

ビルドマシンで作成した ~/rpmbuild/RPMS を各ノードにコピーして次のコマンドでインストールします。

```
# cd RPMS/x86_64
# yum install *
# cd ../noarch
# yum install *
# cd ..
# yum install *
```

drbdtop はバイナリのみですので、drbdtop のトップディレクトリに作成された drbdtop バイナリをコピーして使います。

8. LINSTORを使ったDRBDの設定

リアルタイムでレプリケートするために、DRBDを設定します。

8.1. LVM設定

drbdpoolという名前のVGを作成します。DRBD9のボリュームはこのVGの中に作られます。

```
# pvcreate /dev/sda1
# vgcreate drbdpool /dev/sda1
```

LVMの設定はLINSTORを使うすべてのノードで実行します。

8.2. LINSTORの起動

各ノードにて、LINSTORを起動します。
ノードlinstor1でだけlinstor-controllerとlinstor-satelliteを起動します。

```
# systemctl start linstor-controller
# systemctl start linstor-satellite
```

ノードlinstor2、linstor3、linstor4ではlinstor-satelliteを起動します。

```
[root@linstor2]# systemctl start linstor-satellite
```

8.3. DRBDノード追加

DRBDクラスタにノードを追加します。
このコマンドはすべてlinstor1で実行します。

```
# linstor node create linstor1 10.30.10.1 --node-type Combined
# linstor node create linstor2 10.30.10.2
# linstor node create linstor3 10.30.10.3
# linstor node create linstor4 10.30.10.4
# linstor node list                                    #確認コマンド
```

8.4. ストレージプールの定義

ストレージプールの定義を行います。ここで定義します。
このコマンドもlinstor1で実行します。

```
# linstor storage-pool-definition create pool0
```

pool0という名前のストレージプールが作成されました。この名前はあとの設定でも使用しますので、判りやすい名前で定義します。

8.5. ストレージプールの割当

定義したストレージプールをどのノードに割り当てるか設定を行います。 このコマンドもlinstor1で実行します。

```
# linstor storage-pool create lvm linstor1 pool0 drbdpool
# linstor storage-pool create lvm linstor2 pool0 drbdpool
# linstor storage-pool create lvm linstor3 pool0 drbdpool
```

8.6. リソース名の定義

リソース名の定義を行います。ここではres0という名前のリソースを定義します。

```
# linstor resource-definition create res0
```

8.7. volumeの定義

ここでは、volumeの設定を行います。下記の例は、res0のリソースに対して1Gのvolumeを定義する設定になります。

```
# linstor volume-definition create res0 1G
```

8.8. リソースに対するノードのアサイン設定

リソースにアサインするノードの設定を行います。
下記の例では、ノードlinstor3に対してはストレージを割り当てませんがDRBDの仮想デバイスは使用できます。DRBDのデバイスに書き込むと、ノードlinstor1とlinstor2でデータが保存されます。

```
# linstor resource create linstor1 res0 --storage-pool pool0
# linstor resource create linstor2 res0 --storage-pool pool0
# linstor resource create linstor3 res0 --diskless
# linstor resource list
```

8.9. ファイルシステムの作成

ファイルシステムを作成します。下記の例ではxfsファイルシステムを作成します。

```
# mkfs.xfs /dev/drbd1000
```

mkfsで指定するブロックデバイス名"/dev/drbd1000"はボリューム毎に変わり、1000から数字が順番に割り当てられます。この番号は、"linstor volume-definition list"コマンドを実行して調べることができます。

```
[root@linstor1 ~]# linstor volume-definition list
+-------------------------------------------------------+
| ResourceName | VolumeNr | VolumeMinor | Size  | State |
|-------------------------------------------------------|
| res0         | 0        | 1000        | 1 GiB | ok    |
+-------------------------------------------------------+
```

VolumeMinorの番号がデバイスの番号になります。

8.10. ファイルシステムのマウント

作成したファイルシステムをマウントします。
ファイルシステムのマウントは、ノードlinstor1、linstor2、linstor3のどのノードでも行えます。

```
# mount /dev/drbd1000 /mnt
```

mountできることを確認します。

8.11. ボリュームの削除

作成したボリュームを削除します。削除する前にファイルシステムのマウントをまず解除します。

```
# umount /dev/drbd1000
```

マウントが解除出来たら次のコマンドを実行してボリュームを削除します。

```
# linstor volume-definition delete res0 0
```

8.12. リソースの削除

次にリソースも削除します。

```
# linstor resource-definition delete res0
```

9. LINSTOR環境の初期化

LINSTORのデータベースはコントローラが動いているノードの/var/lib/linstor/にlinstordb.mv.dbとlinstordb.trace.dbという名前の２つのファイルとして存在します。このファイルを削除するとLINSTORを初期の状態に戻せます。

```
# systemctl stop linstor-controller.service
# rm -f /var/lib/linstor/linstordb.*.db
# systemctl start linstor-controller.service
```

10. スナップショットの作成

スナップショット(LVM-Thin)を使った設定を行います。DRBDはスナップショットをLVMのThinプールの機能を利用して実装しています。前に設定したストレージプールの設定ではThinプールは使えないので、LINSTOR環境を一旦初期化して下さい。

10.1. DRBDノード追加

新たな環境でノードを追加します。
このコマンドはすべてlinstor1で実行します。

```
# linstor node create linstor1 10.30.10.1 --node-type Combined
# linstor node create linstor2 10.30.10.2
# linstor node create linstor3 10.30.10.3
# linstor node create linstor4 10.30.10.4
# linstor node list
```

10.2. LVM-Thin設定

VGのdrbdpoolにThin設定を追加します。すべてのノードにて実行します。

```
# lvcreate --thin -l 100%FREE drbdpool/thin
```

10.3. ストレージプールの定義

ストレージプールの定義を行います。ここで定義します。
このコマンドもlinstor1で実行します。

```
# linstor storage-pool-definition create pool0
```

pool0という名前のストレージプールが作成されました。

10.4. ストレージプールの割当

定義したストレージプールをどのノードに割り当てるか設定を行います。
このコマンドもlinstor1で実行します。

```
# linstor storage-pool create lvmthin linstor1 pool0 drbdpool/thin
# linstor storage-pool create lvmthin linstor2 pool0 drbdpool/thin
# linstor storage-pool create lvmthin linstor3 pool0 drbdpool/thin
```

10.5. リソースの定義とボリュームのマウント

前節の通常のLVM環境で行なったリソースの定義とボリュームのマウントを一気に実行します。

```
# linstor resource-definition create res0
# linstor volume-definition create res0 1G
# linstor resource create linstor1 res0 --storage-pool pool0
# linstor resource create linstor2 res0 --storage-pool pool0
# linstor resource create linstor3 res0 --diskless
# mkfs.xfs /dev/drbd1000
# mount /dev/drbd1000 /mnt
```

10.6. スナップショットの作成

リソースres0に対してのスナップショットを作成します。

```
# linstor snapshot create res0 snap0
```

10.7. スナップショットの復元

復元用のリソースを定義してスナップショットをマウントします。

```
# linstor resource-definition create restore_snap
# linstor snapshot volume-definition restore --from-resource res0 \
                --from-snapshot snap0 --to-resource restore_snap
# linstor snapshot resource restore --from-resource res0 \
                --from-snapshot snap0 --to-resource restore_snap
# mount -o nouuid /dev/drbd/by-res/restore_snap/0 /snap
```

10.8. スナップショットの削除

スナップショットの削除は次のコマンドで行います。

```
# linstor snapshot delete res0 snap0
```

復元用のリソースは通常のリソースと同じように削除できます。

```
#  linstor resource-definition delete restore_snap
```

以上