試して覚える Pacemaker-2.0入門

『シェアードナッシングな高可用クラスタの実現』

2021/3/5 OSC2021 Online/Spring
Linux-HA Japan プロジェクト
松浦 健太

目次

- Pacemakerとシェアードナッシング
- シェアードナッシング構成の構築
- おわりに

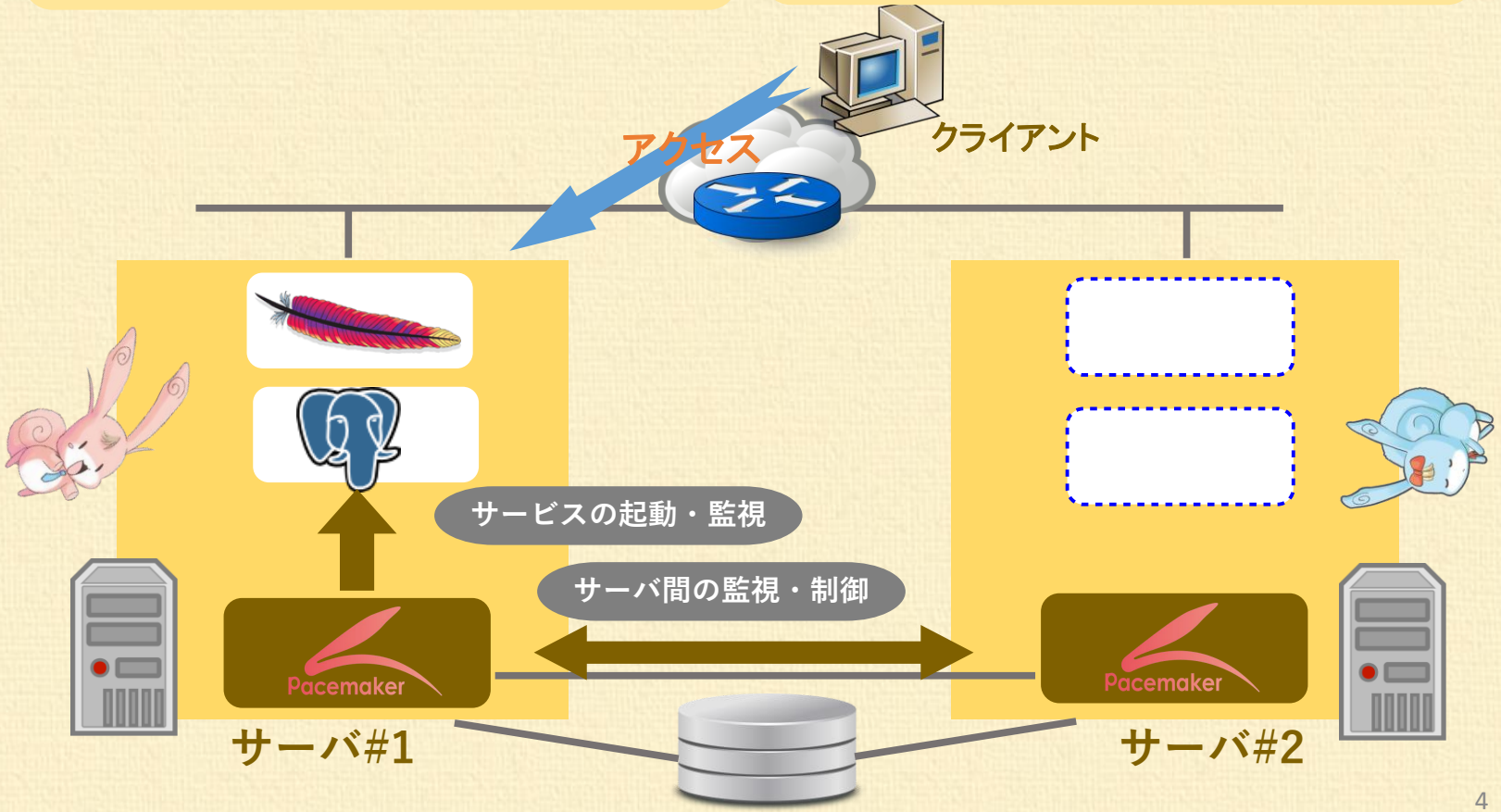
目次

- **Pacemakerとシェアードナッシング**
- シェアードナッシング構成の構築
- おわりに

高可用クラスタソフト(1/2)

アプリ起動・停止の自動管理

サーバ・アプリの故障監視

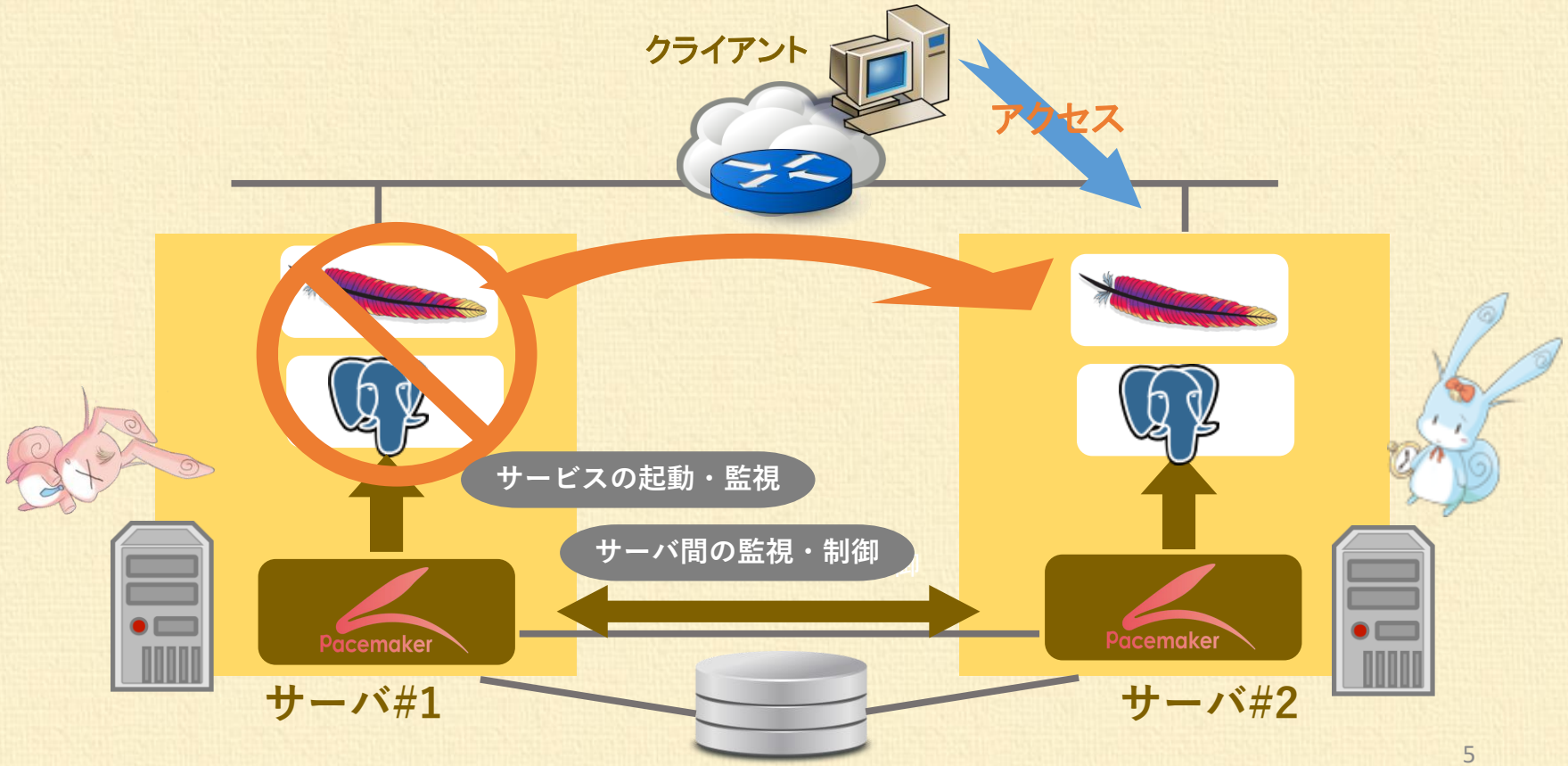


高可用クラスタソフト(2/2)

故障検知時に自動的にフェイルオーバ

ダウンタイムの最小化

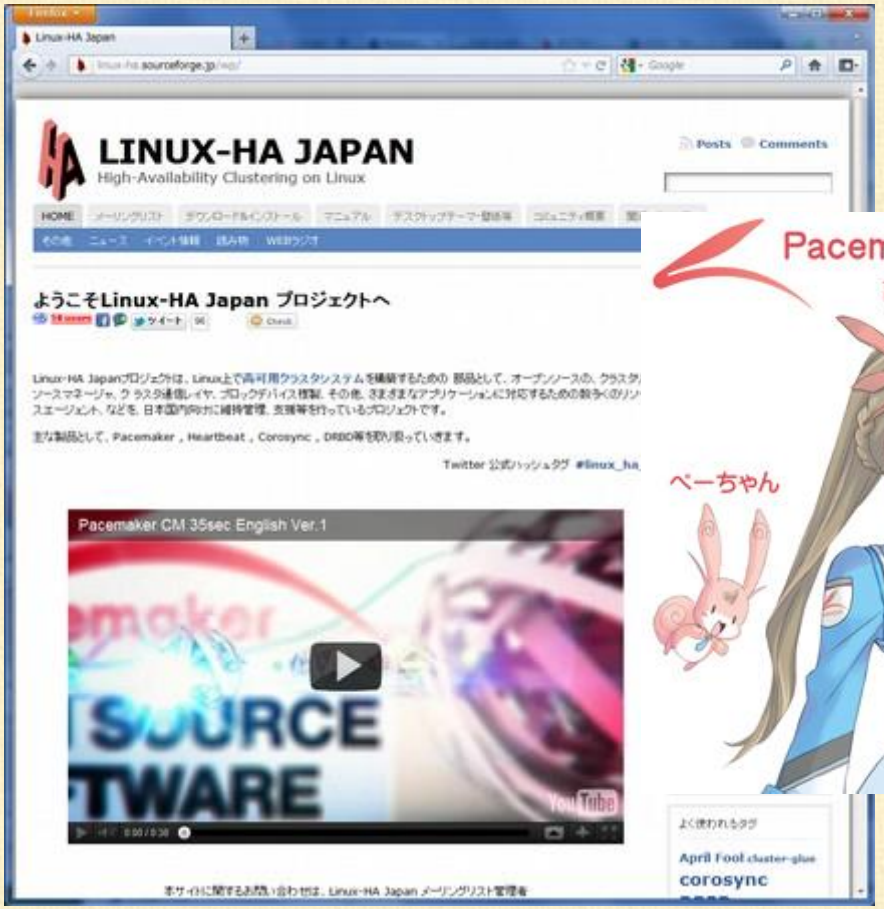
STONITH※によるデータの安全性確保



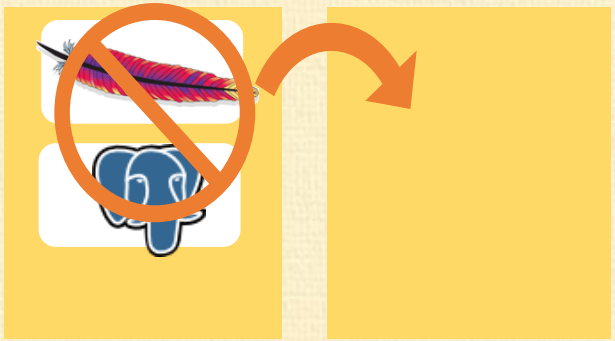
※ スプリットブレイン、F/Oにおけるリソース停止失敗時に対向サーバを強制電源断する機能

もっと知りたい方はこちらまで

http://linux-ha.osdn.jp/

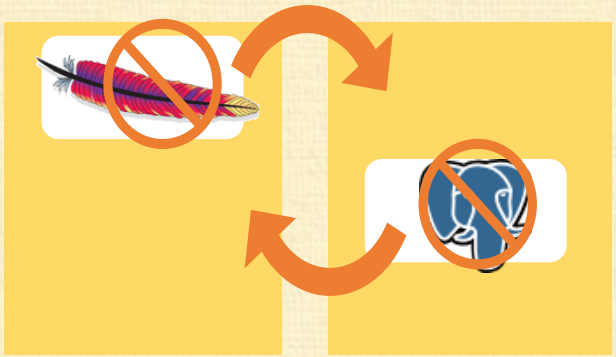


さまざまな高可用クラスタ構成(1/2)

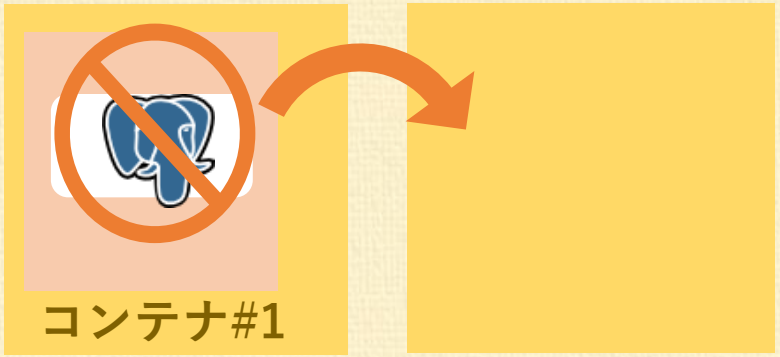


1+1構成

N+1構成(N+M構成)

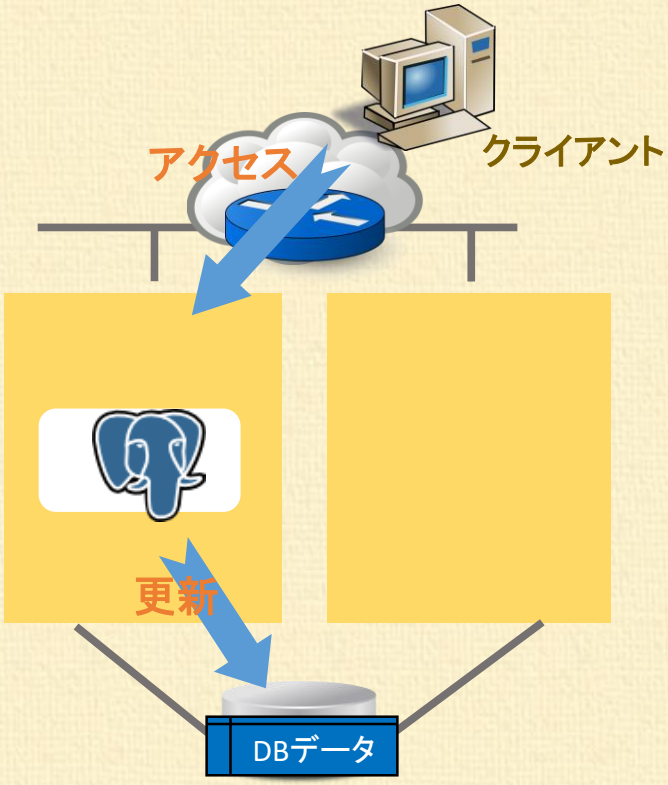


1+1Cross構成

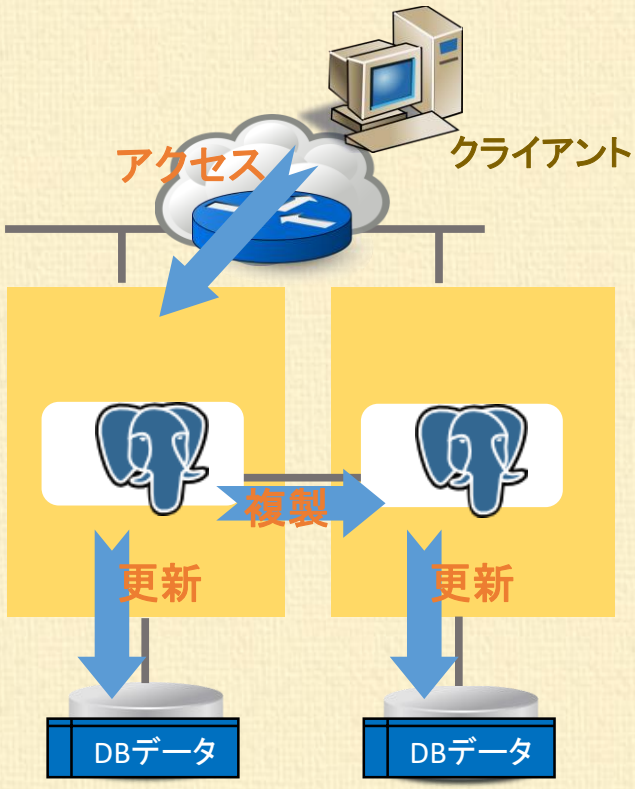


コンテナの管理(bundle)

様々な高可用クラスタ構成(2/2)



シェアードディスク構成



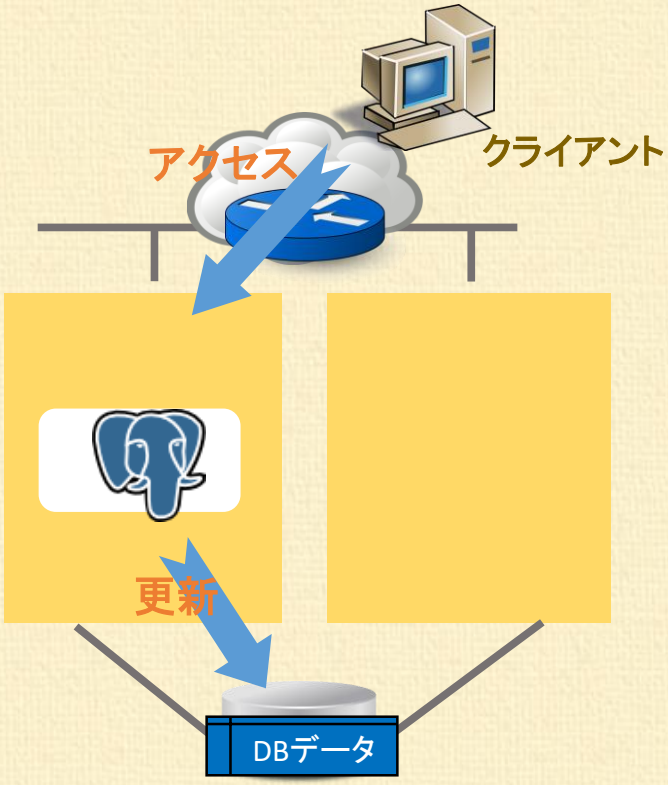
シェアードナッシング構成

(備考) シェアードディスク構成 vs シェアードナッシング構成

	シェアードディスク構成	シェアードナッシング構成
費用	共有ディスク(相当のもの)必須	 普通のHDDでよい
運用のしやすさ	 データは1箇所のみ	2箇所のデータの整合性を考慮
データの安全性	最新データは共有ディスク上のみ	 最新データは2箇所に分散
DBの性能	 レプリケーションのオーバーヘッド(※)なし (※) ネットワーク遅延など	レプリケーションのオーバーヘッドあり
負荷分散	構成上不可能	 データの複製に使用するソフトウェア依存で読書き可能なサーバと読込みのみ可能なサーバに分散できる
実績	 多数あり	 多数あり

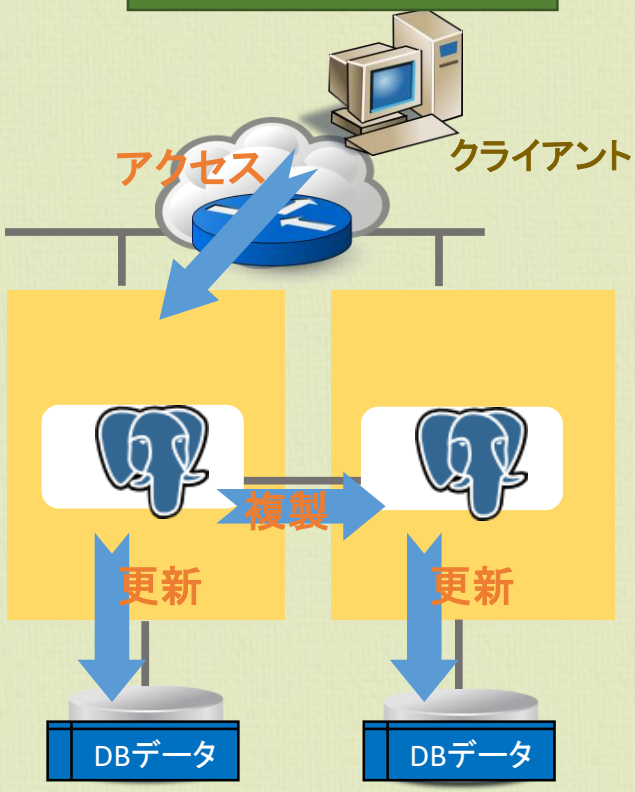
一長一短。 サービスの要件に応じて選択すること！

本日のテーマ



シェアードディスク構成

こちら



シェアードナッシング構成

目次

- Pacemakerとシェアードナッシング
- **シェアードナッシング構成の構築**
- おわりに

シェアードナッシング構成の実現(1/3)

シェアードナッシング構成を実現するにはデータを複製し、適切に管理する仕組みが必要。

本セミナーでは

DRBD によるデータの複製

および

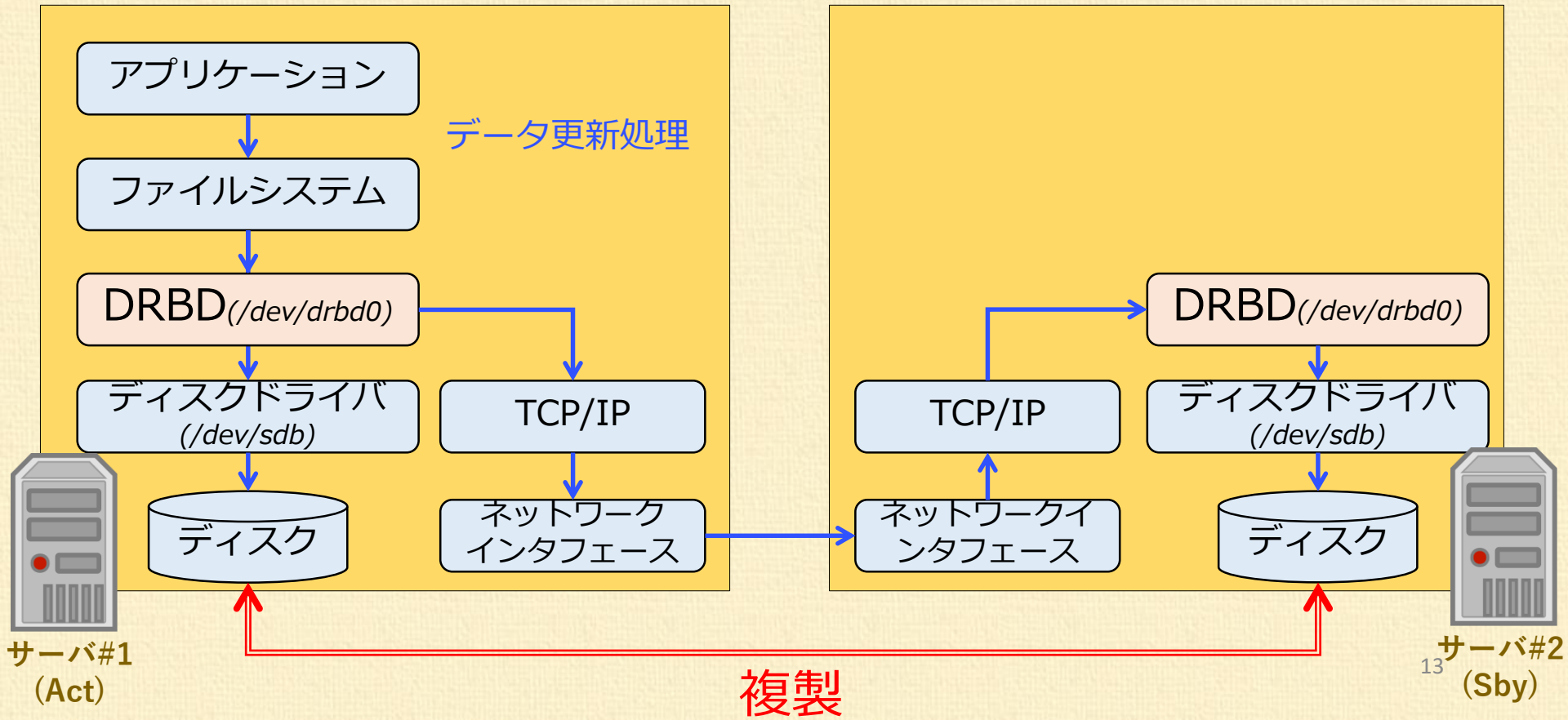
PacemakerのPromotableリソースを使用したクラスタ管理

による実現方法を例に紹介します。

シェアードナッシング構成の実現(2/3)

DRBDとは、

ディスクドライバへの更新依頼を別サーバのディスクドライバにも行い、データを複製するソフトウェア。

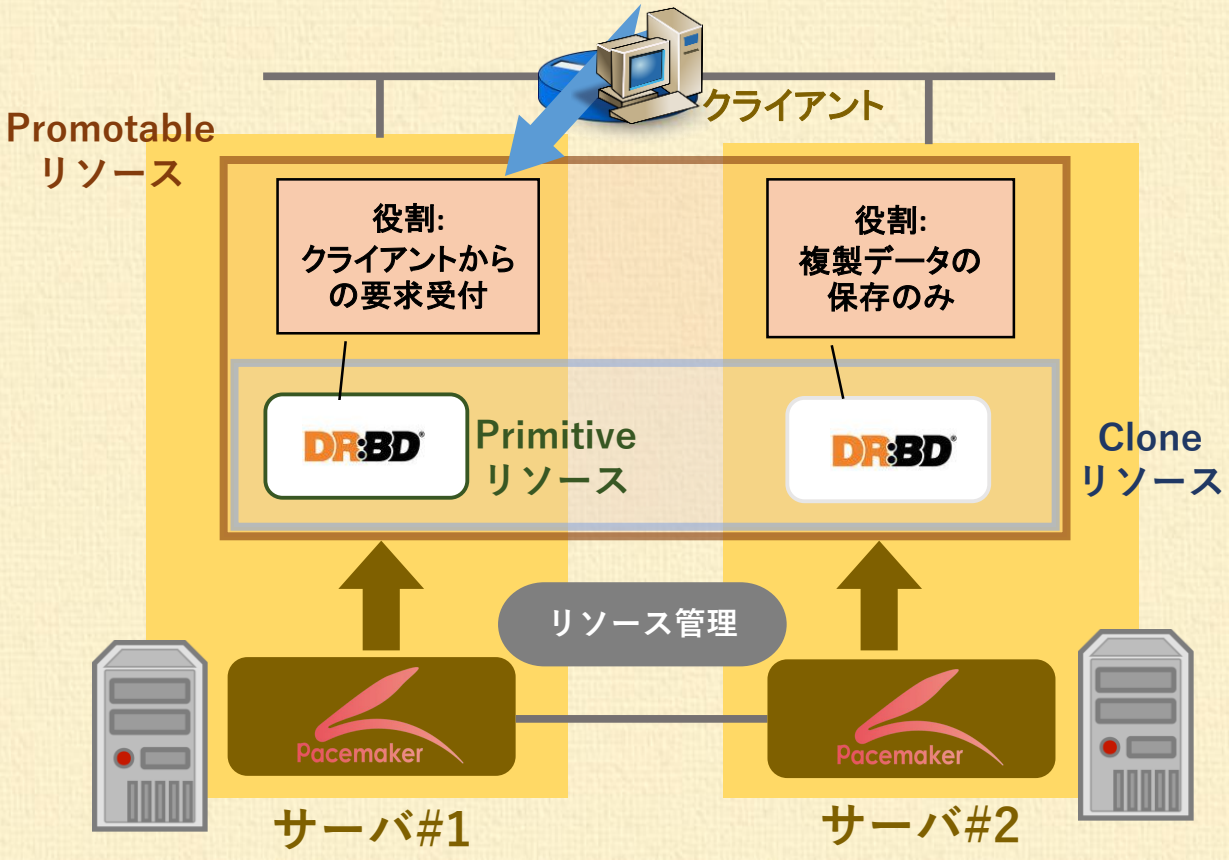


シェアードナッシング構成の実現(3/3)

Promotableリソースとは、

複数サーバでリソースを起動し、かつ役割を個別に管理することができるPacemakerの機能。

(Pacemaker-1.1系ではMaster/Slaveリソースと呼ばれていた機能)



Promotableリソースの用語

機能名は以下の様に見直しが行われてます。

バージョン	用語
Pacemaker-1.0系、Pacemaker-1.1系	Master/Slaveリソース
Pacemaker-2.0系、Pacemaker-2.1系(※1)	Promotableリソース

また、役割の名称(MasterおよびSlave)の用語も以下の様になります。

アプリケーション	用語	
Pacemaker-2.0 までの用語(本資料で採用)	Master	Slave
Pacemaker-2.1 以降の用語(※1)	promoted	unpromoted
PG-REX(PostgreSQL) での用語	Primary	Standby
DRBD での用語	Primary	Secondary

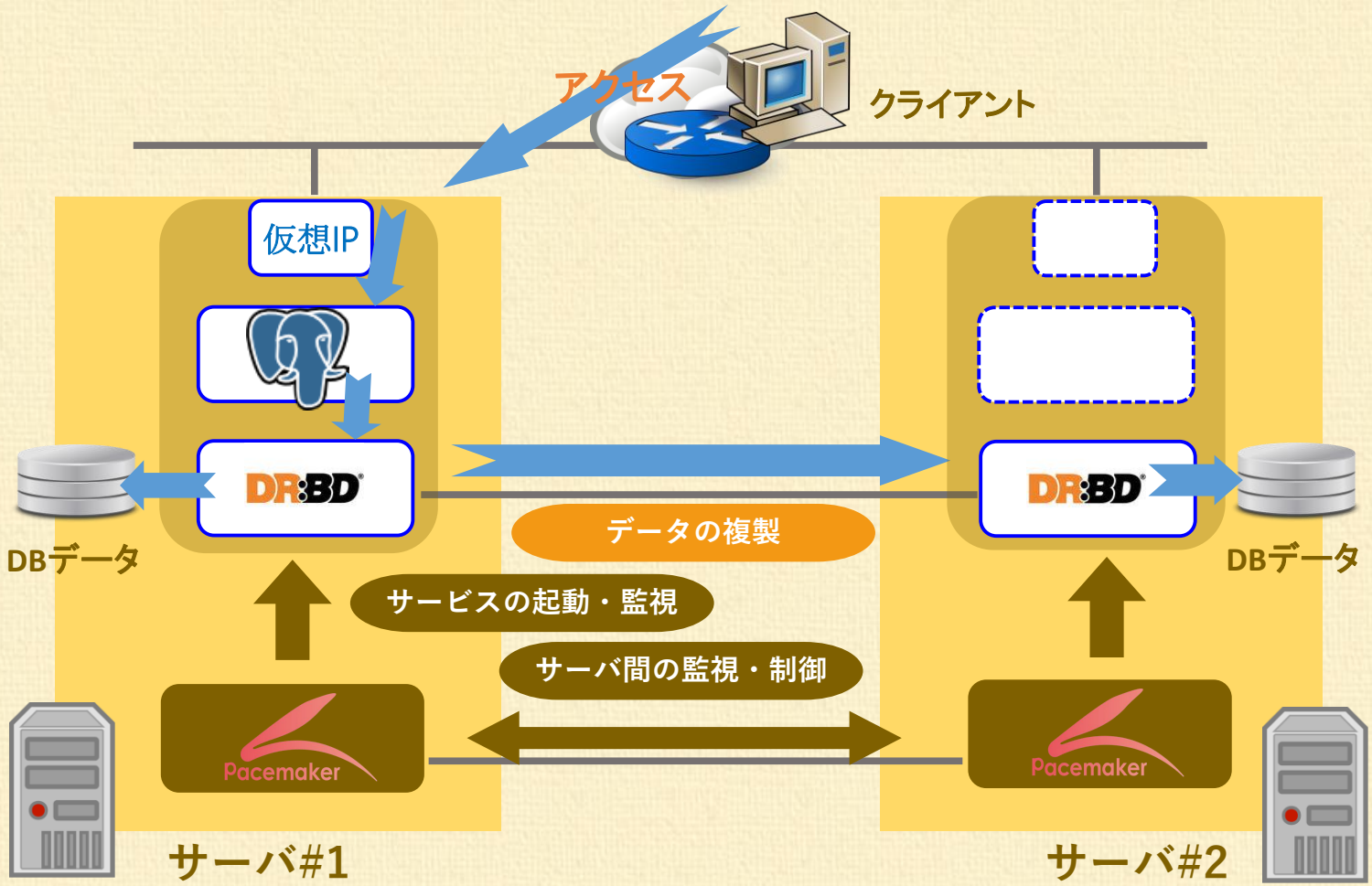
この様に場面によって呼び方が変わっています。
本資料では**枠線**で囲った用語を使用します。

※1 Pacemaker-2.1系の情報は開発中のものであり変更される場合があります。

では、さっそく
DRBDとPromotableリソースを
使用したシェアードナッシング構成
を構築...

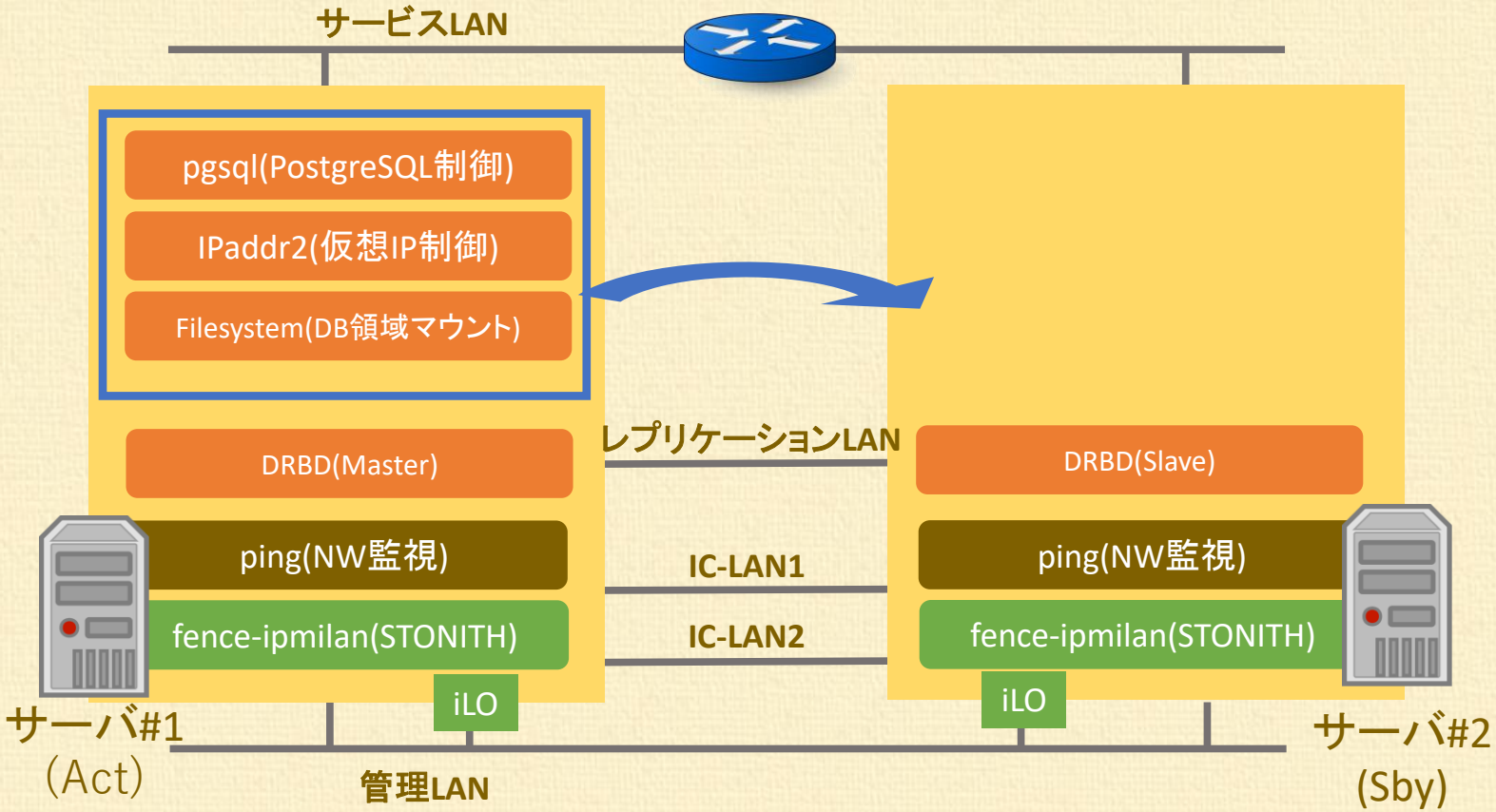
構築するDRBD構成(1/3)

以下の様なシェアードナッシングのシステムを
構築します。



構築するDRBD構成(1/3)

Pacemakerが管理するリソースは以下の様な構成。



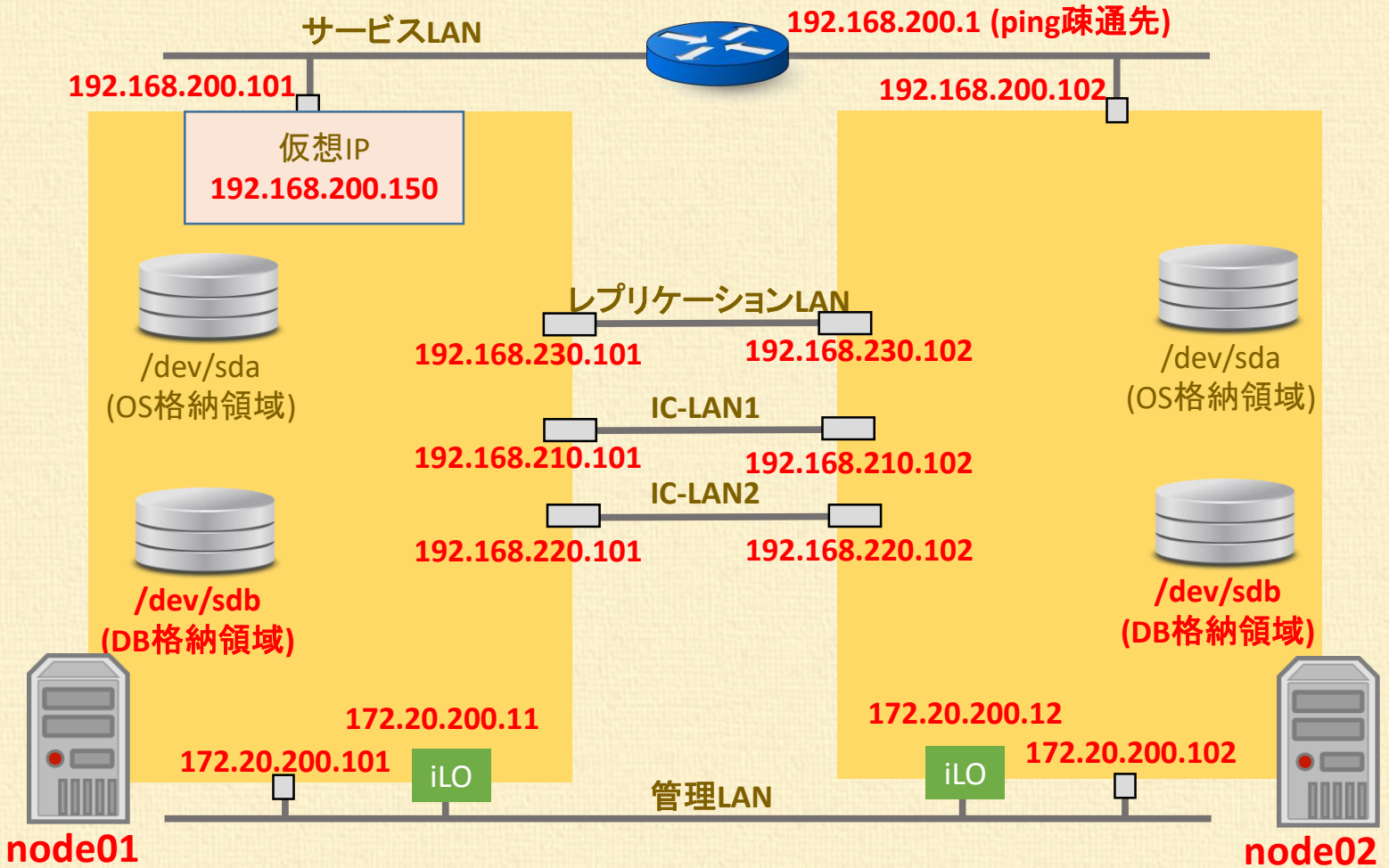
構築するDRBD構成(2/3)

アプリケーションは以下のパッケージを使用します。

製品	パッケージ
Cent OS	CentOS-8.3.2011-x86_64
Pacemaker	CentOS HighAvailability 同梱版 (pacemaker-2.0.4-6.el8.x86_64)
pm_extra_tools	pm_extra_tools-1.2-1.el8.noarch
DRBD	kmod-drbd-9.0.27_4.18.0_240.1.1.el8_3.x86_64-1.x86_64 drbd-utils-9.16.0-1.el8.x86_64 drbd-pacemaker-9.16.0-1.el8.x86_64 (LINBIT 提供パッケージ)
PostgreSQL	postgresql12-12.6-1PGDG.rhel8.x86_64 postgresql12-server-12.6-1PGDG.rhel8.x86_64 postgresql12-libs-12.6-1PGDG.rhel8.x86_64 postgresql12-contrib-12.6-1PGDG.rhel8.x86_64 postgresql12-docs-12.6-1PGDG.rhel8.x86_64 (PostgreSQLコミュニティパッケージ: https://yum.postgresql.org/rpmchart/)

構築するDRBD構成(3/3)

ディスクやネットワーク、サーバ名は以下を使用します。
環境にあわせて読み替えてください。



前提

OSインストールやネットワーク設定などのOS関連の設定は両系とも完了済みであるとし、以下を紹介します。

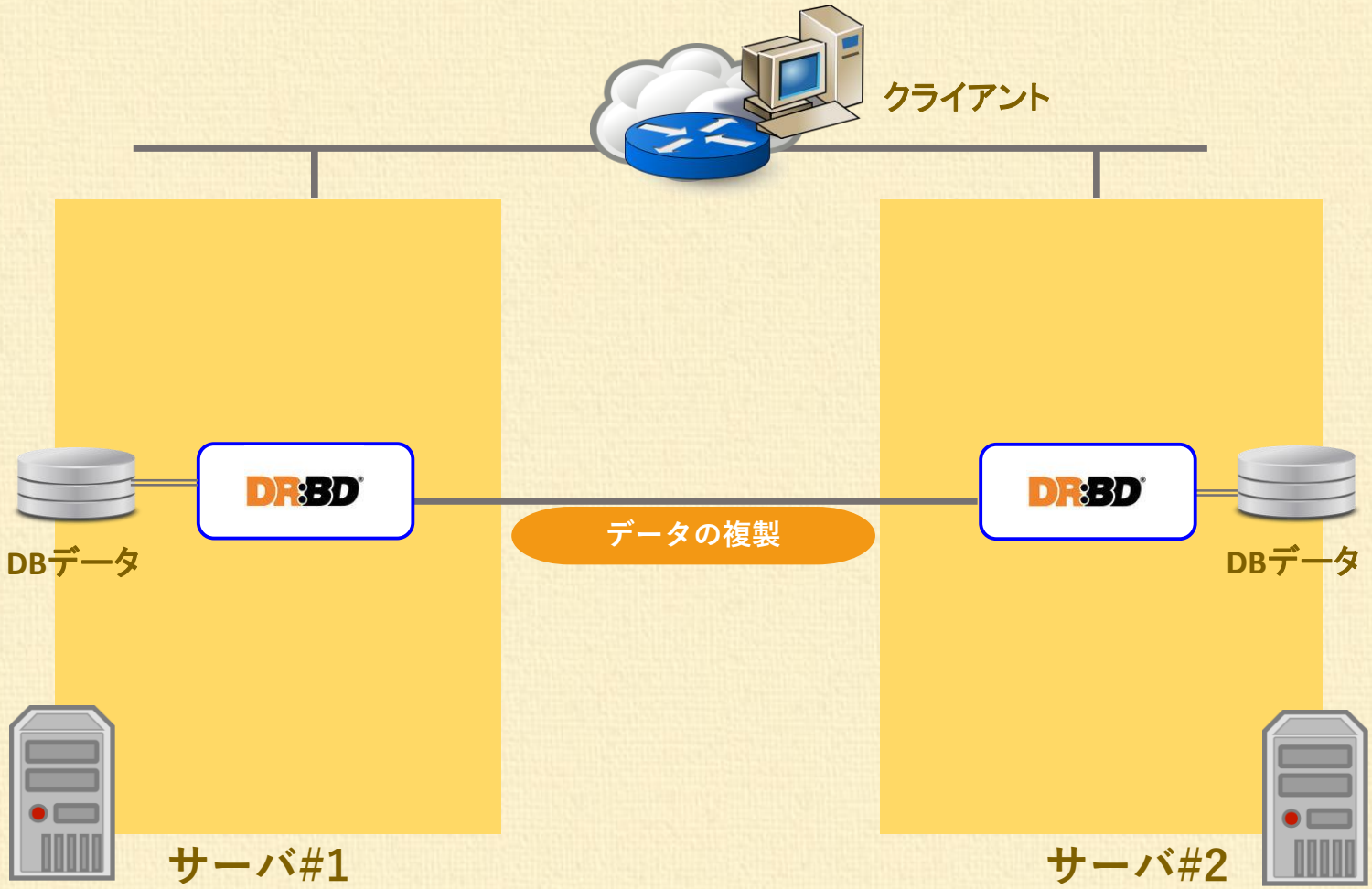
1. DRBDのインストール、設定、同期
2. PostgreSQLのインストール、設定
3. Pacemakerのインストール、設定、リソース定義
4. 動作確認

なお、Pacemakerのインストールや設定については下記のセミナーの設定例(シェアードディスク構成)と異なる点について重点的に紹介します。

試して覚えるPacemaker-2.0入門『構築・リソース設定』
http://linux-ha.osdn.jp/wp/archives/4970

1. DRBDのインストール、設定、同期

DRBDを構築し、データの複製を行います。



1. DRBDのインストール、設定、同期

DRBDをインストール・設定を行い、サーバ間でディスク領域(/dev/sdb)を同期します。

- DRBDのインストール

```
# ls　[両系で実施]
drbd-pacemaker-9.16.0-1.el8.x86_64.rpm
drbd-utils-9.16.0-1.el8.x86_64.rpm
kmod-drbd-9.0.27_4.18.0_240.1.1.el8_3.x86_64-1.x86_64.rpm

# yum install *　[両系で実施]
(snip)
インストール済み:
  drbd-pacemaker-9.16.0-1.el8.x86_64
  drbd-utils-9.16.0-1.el8.x86_64
  kmod-drbd-9.0.27_4.18.0_240.1.1.el8_3.x86_64-1.x86_64

完了しました!
```

※ソースからビルドする手順は下記を参照。
https://blog.drbd.jp/drbd-users-guide-9.0/ch-install-packages.html

1. DRBDのインストール、設定、同期

- DRBDの設定(グローバル設定)

```
# vi /etc/drbd.d/global_common.conf　[両系で実施]
global {
        usage-count no;
        udev-always-use-vnr;
}
common {
        handlers {
                pri-on-incon-degr "echo o > /proc/sysrq-trigger; halt -f";
        }
        startup {
        }
        options {
        }
        disk {
                on-io-error detach;
                disk-barrier no;
                c-min-rate 40M;
        }
        net {
                protocol C;
                fencing resource-only;
                cram-hmac-alg sha512;
                shared-secret password;
                verify-alg sha512;
                csums-alg sha512;
                max-buffers 8192;
                max-epoch-size 8192;
                sndbuf-size 512k;
        }
}
```

1. DRBDのインストール、設定、同期

- DRBDの設定(DRBDリソース設定)

```
# vi /etc/drbd.d/r0.res　[両系で実施]
resource r0 {
        meta-disk internal;
        device /dev/drbd0;
        disk /dev/sdb;
        handlers {
                fence-peer "/usr/lib/drbd/crm-fence-peer.9.sh --logfacility=syslog";
                unfence-peer "/usr/lib/drbd/crm-unfence-peer.9.sh --logfacility=syslog";
        }
        on node01 {
                address 192.168.230.101:7790;
        }
        on node02 {
                address 192.168.230.102:7790;
        }
}
```

※ パラメータの詳細は下記を参照。
https://blog.drbd.jp/drbd-man-pages-9.0/drbdmeta.8.html

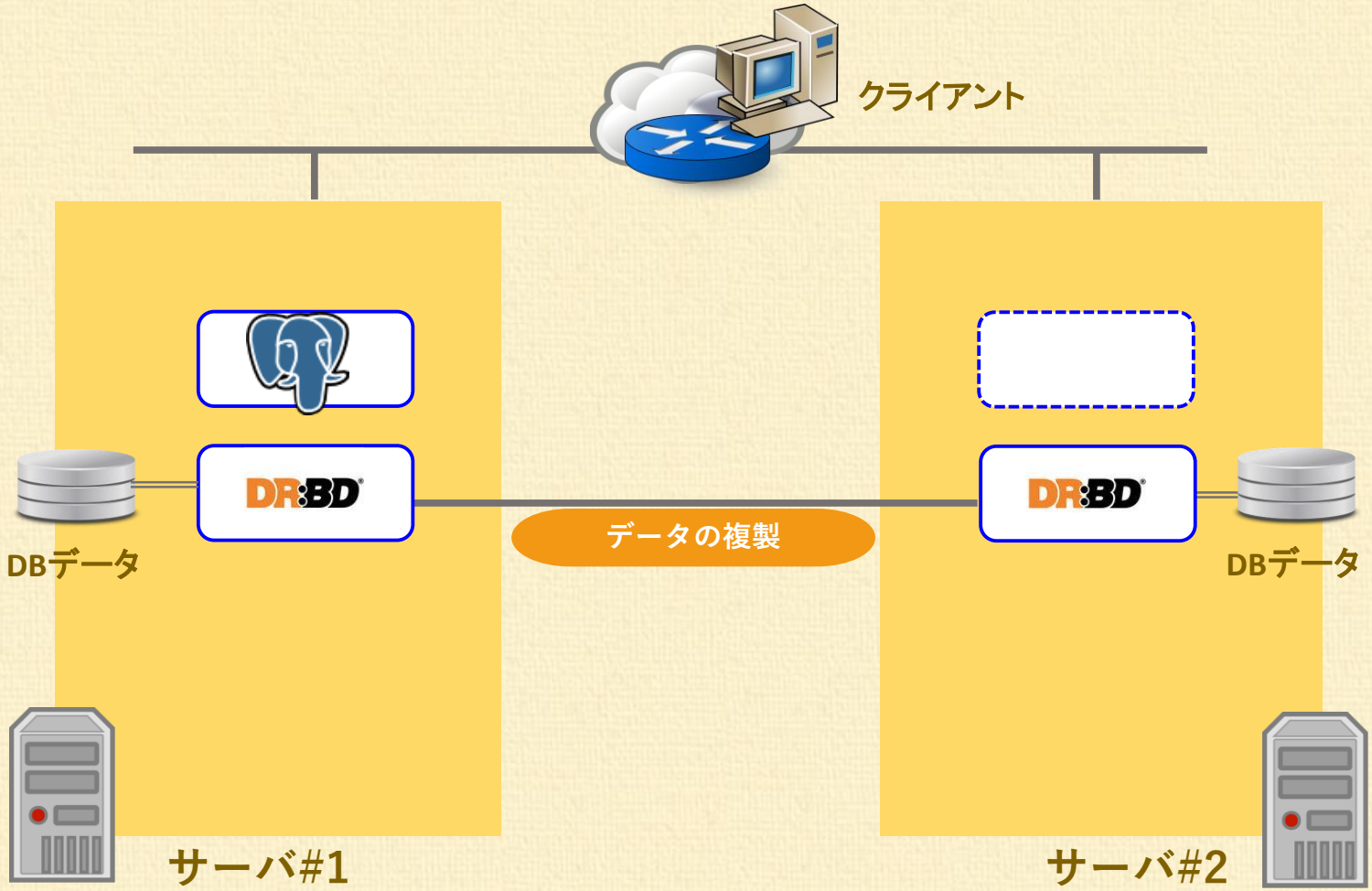
1. DRBDのインストール、設定、同期

➢ DRBDの同期

```
# # DRBDのメタ領域を作成　[両系で実施]
# drbdadm create-md all
initializing activity log
initializing bitmap (128 KB) to all zero
Writing meta data...
New drbd meta data block successfully created.

# # DRBDを起動　[両系で実施]
# drbdadm up all

# # DRBDをMasterとして同期を実施　[片系(node01)で実施]
# drbdadm primary --force all

# # 同期完了まで待機
# drbdadm status all
r0 role:Primary
  disk:UpToDate
  node02 role:Secondary
    replication:SyncSource peer-disk:Inconsistent done:7.97      ← "done"が100になると同期完了

# ls /dev/drbd0
/dev/drbd0      ← DRBDが提供する同期されたデバイス
                  このDRBDデバイス上にPostgreSQLを構築する
```

2. PostgreSQLのインストール、設定

DRBDで複製したデータ領域にPostgreSQLを構築します。



2. PostgreSQLのインストール、設定

- PostgreSQLのインストール

```
# ls　[両系で実施]
postgresql12-12.6-1PGDG.rhel8.x86_64.rpm
postgresql12-contrib-12.6-1PGDG.rhel8.x86_64.rpm
postgresql12-docs-12.6-1PGDG.rhel8.x86_64.rpm
postgresql12-libs-12.6-1PGDG.rhel8.x86_64.rpm
postgresql12-server-12.6-1PGDG.rhel8.x86_64.rpm

# yum install -y *　[両系で実施]
(snip)
インストール済み:
  postgresql12-12.6-1PGDG.rhel8.x86_64
  postgresql12-contrib-12.6-1PGDG.rhel8.x86_64
  postgresql12-docs-12.6-1PGDG.rhel8.x86_64
  postgresql12-libs-12.6-1PGDG.rhel8.x86_64
  postgresql12-server-12.6-1PGDG.rhel8.x86_64

完了しました!
```

2. PostgreSQLのインストール、設定

➢ PostgreSQLの設定

```
# # DRBDデバイスをマウントするディレクトリとログ出力先ディレクトリを作成　[両系で実施]
# mkdir /dbfp /var/log/pg_log

# # DRBDデバイス上にファイルシステムを作成してマウント　[片系(node01)で実施]
# mkfs -t xfs /dev/drbd0
(snip)
# mount /dev/drbd0 /dbfp
# df /dev/drbd0
ファイルシス　1K-ブロック　使用　使用可 使用% マウント位置
/dev/drbd0            4183900   62248 4121652    2% /dbfp

# # PostgreSQLユーザに権限を変更　[両系で実施]
# chown -R postgres:postgres /dbfp /var/log/pg_log

# # PostgreSQLの環境変数を設定　[両系で実施]
# su - postgres
$ vi .pgsql_profile
PGDATA=/dbfp/pgdata/data
export PGDATA
export PATH=/usr/pgsql-12/bin:$PATH

$ # DBインスタンスを作成　[片系(node01)で実施]
$ initdb -D /dbfp/pgdata/data --encoding=UTF8 --no-locale --data-checksums
(snip)
成功しました。以下のようにしてデータベースサーバを起動することができます:
    pg_ctl -D /dbfp/pgdata/data -l ログファイル start
```

/dev/sdbではなく/dev/drbd0を指定している点に注意！

2. PostgreSQLのインストール、設定

➢ PostgreSQLの設定

```
$ # DBインスタンスの設定を変更　[片系(node01)で実施]
$ vi /dbfp/pgdata/pg_hba.conf
(snip)
# IPv4 local connections:
host    all             all             127.0.0.1/32            trust
host    all             all             192.168.200.150/32      md5
host    all             all             192.168.200.101/32      md5
host    all             all             192.168.200.102/32      md5
(snip)
$ vi /dbfp/pgdata/postgresql.conf
(snip)
listen_addresses = '*'
(snip)
log_directory = '/var/log/pg_log'
log_filename = 'postgresql-%Y-%m-%d.log'
(snip)
```

※ PostgreSQLの詳細は下記を参照。
https://pgsql-jp.github.io/

2. PostgreSQLのインストール、設定

- PostgreSQLの設定

```
$ # PostgreSQLの起動確認　[片系(node01)で実施]
$ pg_ctl start
(snip)
完了
サーバ起動完了
$ psql -l
 名前　 | 所有者 | エンコーディング | 照合順序 | Ctype(変換演算子) |　 アクセス権限
-----------+----------+-----------------+----------+------------------+-----------------------
 postgres  | postgres | UTF8          | C      | C              |
 template0 | postgres | UTF8          | C      | C              | =c/postgres        +
           |          |               |        |                | postgres=CTc/postgres
 template1 | postgres | UTF8          | C      | C              | =c/postgres        +
           |          |               |        |                | postgres=CTc/postgres
(3 行)
$ pg_ctl stop
サーバ停止処理の完了を待っています....完了
サーバは停止しました

$ # DRBDデバイスをアンマウント　[片系(node01)で実施]
$ exit
# umount /dev/drbd0

# DRBDを停止
# drbdadm down all　[Slave(node02) --> Master(node01) の順で実施]
```

3. Pacemakerのインストール、設定、リソース定義

Pacemakerをインストールし、サーバ間でクラスタを構築できるようにします。

3. Pacemakerのインストール、設定、リソース定義

- PacemakerとPm_extra_toolsのインストール

```
# # Pacemakerのインストール　[両系で実施]
# dnf install pcs pacemaker fence-agents-all --enablerepo=HighAvailability
(snip)
完了しました。

# # pm_extra_toolsのインストール　[両系で実施]
# ls
pm_extra_tools-1.2-1.el8.noarch.rpm
# yum install -y *
(snip)
完了しました!
```

※ pm_extra_toolsパッケージは以下のURLからダウンロード。
https://osdn.net/projects/linux-ha/releases/p16919

3. Pacemakerのインストール、設定、リソース定義

- Pacemakerの設定

```
# # pcsdサービスの起動　[両系で実施]
# systemctl start pcsd.service
# systemctl enable pcsd.service
Created symlink /etc/systemd/system/multi-user.target.wants/pcsd.service →
 /usr/lib/systemd/system/pcsd.service.

# # haclusterユーザのパスワード設定　[両系で実施]
# passwd hacluster
ユーザー hacluster のパスワードを変更。
新しいパスワード:
新しいパスワードを再入力してください:
passwd: すべての認証トークンが正しく更新できました。

# # クラスタサーバの認証設定　[片系(node01)で実施]
# pcs host auth node01 addr=172.20.200.101 node02 addr=172.20.200.102
Username: hacluster
Password:
node02: Authorized
node01: Authorized
```

3. Pacemakerのインストール、設定、リソース定義

- Pacemakerの設定(続き)

```
# # クラスタの作成　[片系(node01)で実施]
# pcs cluster setup mycluster node01 addr=192.168.210.101 addr=192.168.220.101 node02 ¥
 addr=192.168.210.102 addr=192.168.220.102

(snip)
Cluster has been successfully set up.

# # ACPI Soft-Off機能の無効化(IPMI STONITH機能を使用する場合のみ)　[両系で実施]
# vi /etc/systemd/logind.conf
(snip)
HandlePowerKey=ignore
(snip)
# systemctl restart systemd-logind.service
```

※ 上記ACPI Soft-Off機能の無効化手順はmulti-user.target設定でのみ有効であり、graphical.targetなどの他のデフォルトターゲットの場合は下記の手順が必要。

https://access.redhat.com/documentation/ja-jp/red_hat_enterprise_linux/8/html/using_the_desktop_environment_in_rhel_8/customizing-gnome-desktop-features_using-the-desktop-environment-in-rhel-8#changing-behavior-when-pressing-the-powerbutton_customizing-gnome-desktop-features

3. Pacemakerのインストール、設定、リソース定義

PacemakerからDRBD、PostgreSQLなどのリソースを管理できるように設定します。

3. Pacemakerのインストール、設定、リソース定義

➢ Pacemakerのリソース定義

pm_extra_tools同梱の下記のエクセルファイルを取得し、リソース定義を行う。
ls /usr/share/pm_extra_tools/pm_pcsgen_sample.xlsx

 # pm_pcsgen 環境定義書 ファイル形式バージョン: 1.1

#表 1-1 クラスタ設定 ... クラスタ・ノード属性

NODE

#	uname ノード名	type パラメータ種別	name 項目	value 設定内容	備考

#表 2-1 クラスタ設定 ... クラスタ・プロパティ

PROPERTY

#	name 項目	value 設定内容	概要	備考
	priority-fencing-delay	10s		

#表 3-1 クラスタ設定 ... リソース・デフォルト

RSC_DEFAULTS

#	name 項目	value 設定内容	概要	備考
	resource-stickiness	200	リソース割当て	
	migration-threshold	1	リソース故障可能回数	

#表 3-2 クラスタ設定 ... オペレーション・デフォルト

OP_DEFAULTS

#	name 項目	value 設定内容	概要	備考

#表 4-1 クラスタ設定 ... リソース構成

RESOURCES

#	resourceItem リソース構成要素	resourceItem	resourceItem	id リソースID	概要	備考
	Group			pgsql-group		
		Primitive		filesystem		

3. Pacemakerのインストール、設定、リソース定義

各リソースを順番に定義(Filesystem、IPaddr2)

#表 7-1 クラスタ設定 … Primitiveリソース

PRIMITIVE

P	id	class	provider	type	
#	リソースID	class	provider	type	概要
	filesystem	ocf	heartbeat	Filesystem	

A	type	name	value	
#	パラメータ種別	項目	設定内容	概要
	options	device	/dev/drbd0	
		directory	/dbfp	
		fstype	xfs	
		force_unmount	safe	

O	type	timeout	interval	on-fail	role	start-delay	
#	オペレーション	タイムアウト値	監視間隔	障害時の動作	役割	起動前待機時間	備考
	start	60s		restart			
	monitor	60s	10s	restart			
	stop	60s		fence			

PRIMITIVE

P	id	class	provider	type	
#	リソースID	class	provider	type	概要
	ipaddr	ocf	heartbeat	IPaddr2	

A	type	name	value	
#	パラメータ種別	項目	設定内容	概要
	options	ip	192.168.200.150	
		nic	enp0s3	
		cidr_netmask	24	

O	type	timeout	interval	on-fail	role	start-delay	
#	オペレーション	タイムアウト値	監視間隔	障害時の動作	役割	起動前待機時間	備考
	start	60s		restart			
	monitor	60s	10s	restart			
	stop	60s		fence			

3. Pacemakerのインストール、設定、リソース定義

各リソースを順番に定義(pgsql、ping)

PRIMITIVE

P	id	class	provider	type	
#	リソースID	class	provider	type	概要
	pgsql	ocf	linuxhajp	pgsql	

A	type	name	value	
#	パラメータ種別	項目	設定内容	概要
	options	pgctl	/usr/pgsql-12/bin/pg_ctl	
		psql	/usr/pgsql-12/bin/psql	
		pgdata	/dbfp/pgdata/data	
		pgdba	postgres	
		pgport	5432	
		pgdb	template1	

O	type	timeout	interval	on-fail	role	start-delay	
#	オペレーション	タイムアウト値	監視間隔	障害時の動作	役割	起動前待機時間	備考
	start	300s		restart			
	monitor	60s	10s	restart			
	stop	300s		fence			

PRIMITIVE

P	id	class	provider	type	
#	リソースID	class	provider	type	概要
	ping	ocf	pacemaker	ping	

A	type	name	value	
#	パラメータ種別	項目	設定内容	概要
	options	name	ping-status	
		host_list	192.168.200.1	
		attempts	2	
		timeout	2	
		debug	true	

O	type	timeout	interval	on-fail	role	start-delay	
#	オペレーション	タイムアウト値	監視間隔	障害時の動作	役割	起動前待機時間	備考
	start	60s		restart			
	monitor	60s	10s	restart			
	stop	60s		fence			

3. Pacemakerのインストール、設定、リソース定義

各リソースを順番に定義(drbd)

PRIMITIVE

P	id	class	provider	type	
#	リソースID	class	provider	type	概要
	drbd	ocf	linbit	drbd	

A	type	name	value	
#	パラメータ種別	項目	設定内容	概要
	options	drbd_resource	r0	
		unfence_if_all_uptodate	true	

O	type	timeout	interval	on-fail	role	start-
#	オペレーション	タイムアウト値	監視間隔	障害時の動作	役割	起動
	start	240s		restart		
	monitor	20s	20s	restart		
	monitor	20s	10s	restart	Master	
	promote	90s		restart		
	demote	90s		fence		
	stop	100s		fence		

Promotableリソース特有の設定。
monitorはMaster用とSlave用を定義する。
※ role列でMaster/Slaveを指定(省略時はSlaveとなる)。
　intervalは必ず異なる値を設定すること。

promoteはSlaveからMasterへ遷移するオペレーション、
demoteはMasterからSlaveへ遷移するオペレーション。

3. Pacemakerのインストール、設定、リソース定義

各リソースを順番に定義(fence_ipmilan)

#表 8-1 クラスタ設定 … STONITHリソース

STONITH

P	id	type		
#	STONITHリソースID	type		概要
	fence1-ipmilan	fence_ipmilan		
A	type	name	value	
#	パラメータ種別	項目	設定内容	
	options	pcmk_host_list	node01	
		ip	172.20.200.11	
		username	pacemaker	
		password	ipmipass1	
		lanplus	1	
O	type	timeout	interval	on-fail
#	オペレーション	タイムアウト値	監視間隔	障害時の動作
	start	60s		restart
	monitor	60s	3600s	restart
	stop	60s		ignore

前回のセミナー(CentOS 8.2)では"delay"を設定して相打ちを防止していたが、Pacemaker-2.0.4(CentOS 8.3以降)ではpriority-fencing-delay機能で相打ちを防止できるようになった。この機能については後述。

STONITH

P	id	type		
#	STONITHリソースID	type		概要
	fence2-ipmilan	fence_ipmilan		
A	type	name	value	
#	パラメータ種別	項目	設定内容	概要
	options	pcmk_host_list	node02	
		ip	172.20.200.12	IPMIポートの
		username	pacemaker	
		password	ipmipass1	
		lanplus	1	
O	type	timeout	interval	on-fail
#	オペレーション	タイムアウト値	監視間隔	障害時の動作
	start	60s		restart
	monitor	60s	3600s	restart
	stop	60s		ignore

3. Pacemakerのインストール、設定、リソース定義

定義したリソースの構成と動作を設定

#表 4-1 クラスタ設定 … リソース構成

RESOURCES

#	resourceItem	resourceItem	resourceItem	id	概要	備考
	リソース構成要素			リソースID		
	Group			pgsql-group		
		Primitive		filesystem		
		Primitive		ipaddr		
		Primitive		pgsql		
	Promotable			drbd-clone		
		Primitive		drbd		
	Clone			ping-clone		
		Primitive		ping		
	Stonith			fence1-ipmilan		
	Stonith			fence2-ipmilan		

Promotableを指定することでMasterとSlaveの管理が行えるようになる。
リソースIDは必ず定義したリソースのIDに"-clone"を付け加えたものを指定すること。

#表 5-1 クラスタ設定 … リソース・パラメータ (meta)

RSC_ATTRIBUTES

#	id	name	value	備考
	リソースID	項目	設定内容	
	drbd-clone	promoted-max	1	
		promoted-node-max	1	
		clone-max	2	
		clone-node-max	1	
		notify	true	
		priority	1	

Promotableリソースの詳細動作を設定。
clone-max: クラスタ内にリソースをいくつ起動するか。
clone-node-max: 1つのサーバにリソースをいくつ起動するか。
promoted-max: クラスタ内にMasterのリソースをいくつ起動するか。
promoted-node-max: 1つのサーバにMasterのリソースをいくつ起動するか。

3. Pacemakerのインストール、設定、リソース定義

リソース制約を設定

#表 9-1 クラスタ設定 ... リソース配置制約 (ノード)

LOCATION_NODE

#	rsc リソースID	prefers/avoids prefers/avoids	node ノード	score スコア	備考 (※roleを設定したい場合は、表9-2 を使用し
	fence1-ipmilan	avoids	node01		
	fence2-ipmilan	avoids	node02		

#表 9-2 クラスタ設定 ... リソース配置制約 (ルール)

LOCATION_RULE

#	rsc リソースID	score スコア	bool_op and/or	attribute 条件属性名	op 条件	value 条件値	role 役割	備考
	drbd-clone	-INFINITY	or	ping-status	lt	1		
				ping-status	not_defined			

#表 10-1 クラスタ設定 ... リソース同居制約

COLOCATION

#	rsc 制約関連リソースID	with-rsc 制約対象リソースID	score スコア(重み付け)	rsc-role 制約関連リソースの役割	with-rsc-role 制約対象リソースの役割	備考
	drbd-clone	ping-clone	INFINITY			
	pgsql-group	drbd-clone	INFINITY		Master	

#表 11-1 クラスタ設定 ... リソース起動順序制約

ORDER

#	first-rsc 先に起動するリソースID	then-rsc 後に起動するリソースID	kind Optional/Mandatory/Seriali	first-action 先起動リソースのアクション	then-action 後起動リソースのアクション	symmetrical 起動と逆順に停止(
	ping-clone	drbd-clone	Optional			false
	drbd-clone	pgsql-group		promote	start	true

with-rsc-role に"Master"を指定することでMasterのリソースと同居する様に指定できる。

Promotableリソースは最初Slaveとして起動され、promoteすることでMasterとなる。
promote、demoteのタイミングはfirst-action、then-actionにて指定できる。

3. Pacemakerのインストール、設定、リソース定義

priority-fencing-delayを設定(Pacemaker-2.0.4以降のみ)

※ STONITH実行時に遅延時間を設けて相打ちを防止するための機能。
priority(優先度)を確認し、高いサーバを停止する際に一定時間待つ。

#表 2-1 クラスタ設定 … クラスタ・プロパティ

PROPERTY

#	name 項目	value 設定内容	概要	備考
	priority-fencing-delay	10s		

遅延時間を設定。

#表 5-1 クラスタ設定 … リソース・パラメータ (meta)

RSC_ATTRIBUTES

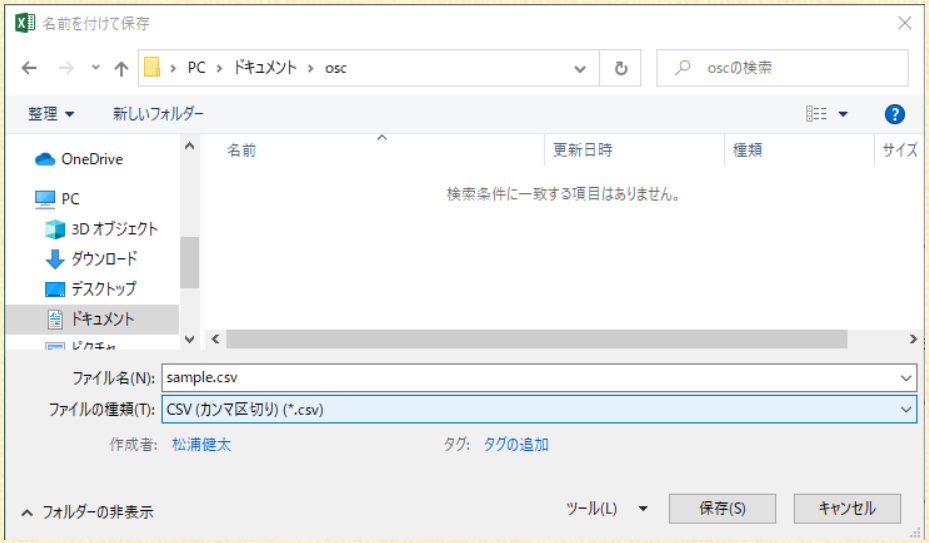
#	id リソースID	name 項目	value 設定内容	備考
	drbd-clone	promoted-max	1	
		promoted-node-max	1	
		clone-max	2	
		clone-node-max	1	
		notify	true	
		priority	1	

リソースが起動しているサーバのpriorityを設定。
Promotableリソースの場合はMaster側が優先される。

3. Pacemakerのインストール、設定、リソース定義

あとは、
Pacemakerへ投入して完成です！

エクセルを『名前をつけて保存』からcsv形式で保存。



3. Pacemakerのインストール、設定、リソース定義

先ほど構築したPacemakerにリソース定義を反映。

```
# # csvファイルをxmlファイル変換　[片系(node01)で実施]
# pm_pcsgen sample.csv
sample.xml (CIB), sample.sh (PCS) を出力しました。

# # Pacemakerを起動　[片系(node01)で実施]
# pcs cluster start --all
node01: Starting Cluster...
node02: Starting Cluster...

# # xmlファイルを反映
# pcs cluster cib-push sample.xml
CIB updated
```

3. Pacemakerのインストール、設定、リソース定義

完成

```
# pcs status --full
(snip)
Node List:
  * Online: [ node01 (1) node02 (2) ]

Full List of Resources:
  * Clone Set: drbd-clone [drbd] (promotable):
    * drbd      (ocf::linbit:drbd):      Slave node02
    * drbd      (ocf::linbit:drbd):      Master node01
  * Resource Group: pgsql-group:
    * filesystem       (ocf::heartbeat:Filesystem):     Started node01
    * ipaddr    (ocf::heartbeat:IPaddr2):       Started node01
    * pgsql     (ocf::linuxhajp:pgsql):  Started node01
  * Clone Set: ping-clone [ping]:
    * ping      (ocf::pacemaker:ping):   Started node02
    * ping      (ocf::pacemaker:ping):   Started node01
  * fence1-ipmilan      (stonith:fence_ipmilan):        Started node02
  * fence2-ipmilan      (stonith:fence_ipmilan):        Started node01
(snip)
```

3. Pacemakerのインストール、設定、リソース定義

実際に故障を発生させてみると...

```
# # PostgreSQLを異常停止　[片系(node01)で実施]
# pkill postgres

# pcs status --full
(snip)
Full List of Resources:
  * Clone Set: drbd-clone [drbd] (promotable):
    * drbd      (ocf::linbit:drbd):      Master node02
    * drbd      (ocf::linbit:drbd):      Slave node01
  * Resource Group: pgsql-group:
    * filesystem       (ocf::heartbeat:Filesystem):     Started node02
    * ipaddr    (ocf::heartbeat:IPaddr2):       Started node02
    * pgsql     (ocf::linuxhajp:pgsql):  Started node02
  * Clone Set: ping-clone [ping]:
    * ping      (ocf::pacemaker:ping):   Started node02
    * ping      (ocf::pacemaker:ping):   Started node01
  * fence1-ipmilan      (stonith:fence_ipmilan):        Started node02
  * fence2-ipmilan      (stonith:fence_ipmilan):        Started node01
(snip)
Failed Resource Actions:
  * pgsql_monitor_10000 on node01 'not running' (7): call=52, status='complete', exitreason='',
      last-rc-change='2021-02-24 13:53:35 +09:00', queued=0ms, exec=0ms
(snip)
```

3. Pacemakerのインストール、設定、リソース定義

復旧手順は以下の通り。

```
# # 故障情報をクリア　[片系(node01)で実施]
# pcs resource cleanup pgsql node=node01

# # node01へ系切替
# pcs resource move pgsql-group node01
# pcs resource clear pgsql-group
Removing constraint: cli-prefer-pgsql-group

# pcs status --full
(snip)
Full List of Resources:
  * Clone Set: drbd-clone [drbd] (promotable):
    * drbd      (ocf::linbit:drbd):      Slave node02
    * drbd      (ocf::linbit:drbd):      Master node01
  * Resource Group: pgsql-group:
    * filesystem       (ocf::heartbeat:Filesystem):     Started node01
    * ipaddr    (ocf::heartbeat:IPaddr2):       Started node01
    * pgsql     (ocf::linuxhajp:pgsql):  Started node01
  * Clone Set: ping-clone [ping]:
    * ping      (ocf::pacemaker:ping):   Started node02
    * ping      (ocf::pacemaker:ping):   Started node01
(snip)
```

目次

- Pacemakerとシェアードナッシング
- シェアードナッシング構成の構築
- **おわりに**

PostgreSQLのデータを シェアードナッシングで構築する場合は DRBDではなくPG-REXも便利！

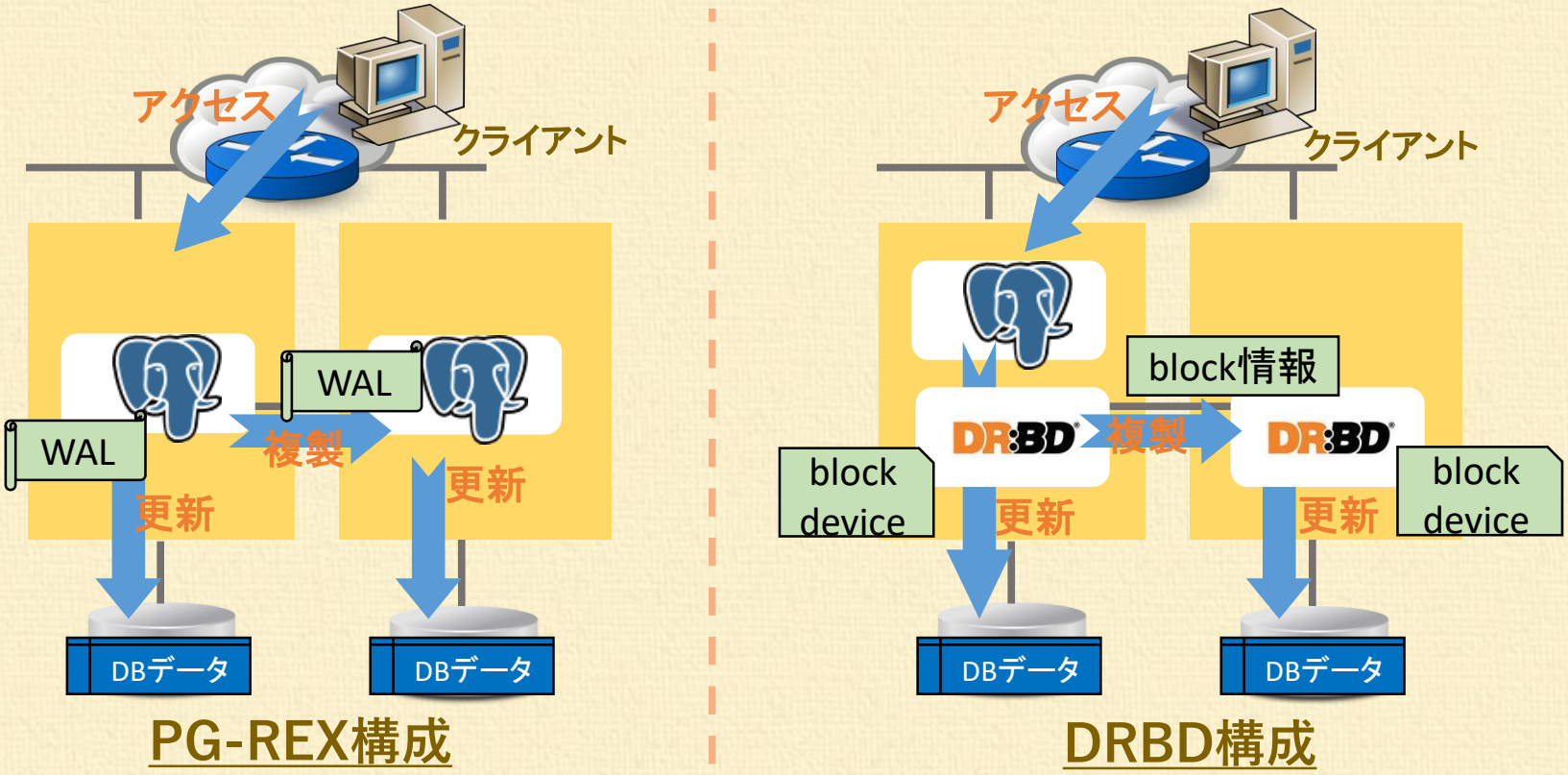
➢PG-REX構成

- PostgreSQLのストリーミングレプリケーション機能(WAL転送) によりDBMSのデータを複製

➢DRBD構成

- DRBDのブロックデバイスレベルでの同期レプリケーション機能によりDBMS(およびファイルシステム)より下層のレイヤでデータを複製

PG-REX構成とDRBD構成のイメージ



PG-REX構成の構築方法は下記にて紹介。
ぜひ、こちらも構築してみてください！
https://ja.osdn.net/projects/pg-rex/

構築した所感としてはPacemaker-1.1とほとんど変わりませんでした！

DRBDもPG-REXもPacemaker-2.0対応は既に落ち着いていて、Pacemaker-1.1系と同じように使用できそうな感じでした。

Pacemaker-2.0の情報は下記のセミナー資料も公開しているので、そちらも参考にしながら構築してみてください！

Linux-HA Japan プロジェクトのこれまでとこれから

http://linux-ha.osdn.jp/wp/archives/4942

試して覚えるPacemaker-2.0入門『構築・リソース設定』

http://linux-ha.osdn.jp/wp/archives/4970

それでも分からないことがあればMLへ
http://linux-ha.osdn.jp/wp/ml

今後もPacemakerを
よろしくお願いします