文書番号：LK20240306-381-001

LifeKeeper for Linux Quorum/Witness Kit Storage mode on Azure Step by step Guide

第 1 版

サイオステクノロジー株式会社

目次

改訂履歴

日付	班	変更情報
2023/12/15	第 1 版	第 1 版 新規作成

1. 始めに

本ドキュメントに含まれる情報は、公表の日付におけるサイオステクノロジー株式会社のポリシーに基づいています。サイオステクノロジー株式会社は記載されている内容をお約束しているわけではありません。また、それらの内容を保証するものでもありません。本ドキュメントは情報提供のみを目的としています。また、記載内容は予告無く変更する場合があります。予めご了承ください。

2. 本ドキュメント

本ドキュメントは、Microsoft Azure（以下 Azure）環境上の Linux 仮想マシンに LifeKeeper for Linux（以下 LifeKeeper）及び Quorum/Witness Kit（以下 QWK）の storage モードを用いる構成例・構築手順例・検証手順例を記したものです。LifeKeeper の運用における詳細な情報は含まれていません

2.1 対象読者について

本ドキュメントは Azure 上で QWK の storage モードの構築を計画している方が対象となります。そのため、LifeKeeper の操作や Azure に関する基本的な知識を持っている技術者を対象としています。

2.2 必要なソフトウェアイメージ

- LifeKeeper for Linux インストールメディア
- Red Hat Enterprise Linux OS イメージ

2.3 製品に関する情報

LifeKeeper for Linux / Single Server Protection 製品に関する詳細は下記リンクよりご参照ください。製品のリリースノートや Recovery Kit の詳細がご確認いただけます。
https://support.us.sios.com/aspx/jpdocs_us_sios_com_home/

また、Recovery Kit の処理概要、脆弱性情報、ライセンスの取得方法等、製品のサポートに関する情報は下記リンクのユーザーポータルでご確認いただけます。
https://lkdkuserportal.sios.jp/hc/ja

3. QWK によるスプリットブレイン対策

本章では HA クラスターで発生するスプリットブレインと呼ばれる現象と、LifeKeeper における対策である QWK の一つ、 storage モードの概要を紹介します。

3.1 スプリットブレインとは

スプリットブレインとは、HA クラスターシステムにおいてハードウェアやノード間通信の障害により、システムが分断され、1 つのサービスが複数のノードで同時に起動する現象を指します。具体的には、複数のシステム上で同時に LifeKeeper の階層が In Service になる状況です。

スプリットブレインが発生すると、各ノードがアプリケーションを制御できると認識しているため、両ノードでデータへのアクセス、共有ストレージデバイスにデータを書き込もうとして、リソースの整合性やデータの一貫性を損なう可能性があります。

3.2 QWK の概要

LifeKeeper に QWK を組み合わせることにより、「スプリットブレイン」の発生リスクを大幅に軽減しながら、より高い信頼度でフェイルオーバーを実行することができるようになります。

QWK は以下の主要なチェックで構成されます。

1. **Quorum チェック** : 通信障害が発生した時、各ノード間で最も多数の合意を得られたノードを Quorum を保持しているとし、リソースを起動することが許可されます。
   Quorum を持っている状態を Quorum チェック成功とします。
   主にスプリットブレインを抑止します。

2. **Witness チェック** : Quorum チェック成功すると実施されます。クラスターには組み込まれていない第三者ノードを Witness ノードと呼びます。クラスターのノード A と、Witness ノードが障害のあったノードのステータスの確認を行い、同じ意見である場合のみ、ノード A のリソースの起動が許可されます。
   主にノード間で発生する単純な通信障害から発生するフェイルオーバーを回避します。

Quorum チェックのモードとして、以下の 4 つが用意されています。

- majority
- tcp_remote
- storage
- none または off

本ドキュメントでは storage モードを使用します。

※他のモードに関しましてはテクニカルドキュメントをご参照ください。

Quorum/Witness

3.3 QWK の storage モードの概要

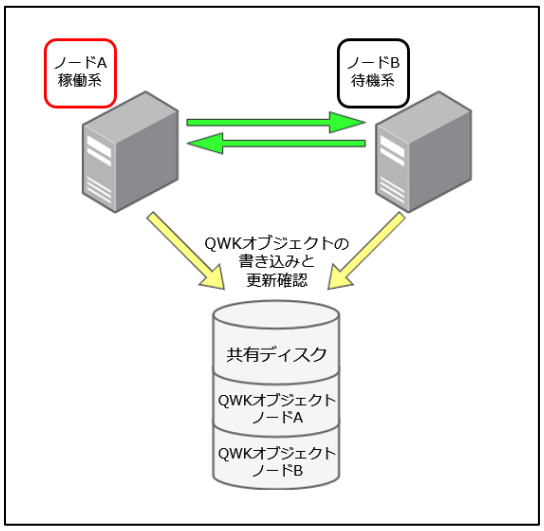

図 storage モードの標準構成

storage モードはクラスター内のすべてのノードからアクセスできる共有ストレージを Witness デバイスとして用いる Quorum のモードです。共有ストレージを介してノードの情報交換を行いますが、この共有ストレージに書き込まれたノードの情報を **QWK オブジェクト**と呼びます。

Quorum チェックでは共有ストレージにアクセスできることで Quorum チェック成功と判断します。Witness チェックでは、障害が疑われるノードの QWK オブジェクトを確認します。一定期間内に更新されていれば障害無し、更新が止まっていれば障害が発生していると考えます。

両方のチェックが成功すると、クラスターは正常に動作していると判断されます。

4. Azure の環境構築

本ドキュメントにおいて Azure を使用して Virtual Machine を構築する際に必要な設定について説明します。

4.1 ネットワーク構成図

本ドキュメントで構成する仮想ネットワークは図の通りとなります。

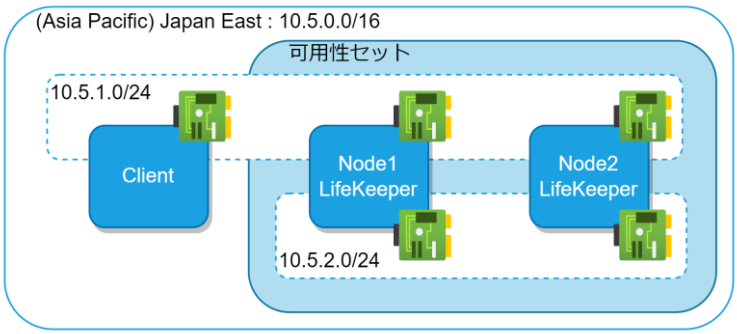

図 4.1-1 ネットワーク構成図

4.2 リソースグループの作成

クラスターを構成するために、リソースグループを作成します。リソースグループは Azure の各種リソースを一括で管理することができます。

1. リソースグループのページを開き、「作成」をクリックします。
   以降、Azure でリソースを作成する場合は上記の手順を行います。

図 4.2-1 リソースグループ　初期画面

2. サブスクリプションを選択し、リソースグループに名前を書き込みます。
   ここでは「LKL-QWK-storage」とします。

図 4.2-2 リソースグループの設定画面

3. 作成者がわかるようにタグをつけます。
   同様に、以降のリソースの全てにタグをつけます。

図 4.2-3 タグの設定画面

4. 「確認及び作成」タブに移動し、「作成」を押すことでリソースグループの作成を終了します。

図 4.2-4 リソースグループの作成

4.3 仮想ネットワークの作成

1. Azure 画面上部の検索窓から、仮想ネットワークを検索し、作成ボタンを押します。
2. 仮想ネットワークの「基本」タブでは以下の表の通り設定を行います。

項目	設定値
リソースグループ	LKL-QWK-storage
仮想ネットワーク名	LKL-QWK-storage-Vnet
地域	(Asia Pacific) Japan East

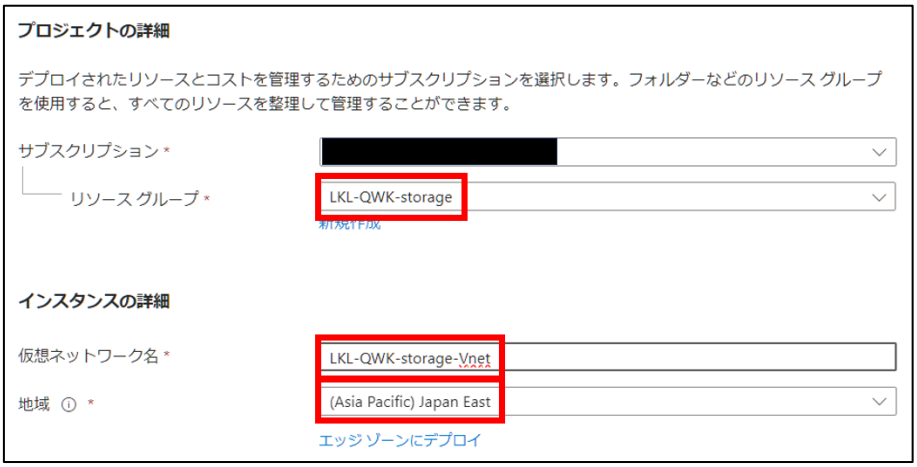

図 4.3-1 仮想ネットワーク 基本の設定画面

3. 「セキュリティ」タブではデフォルトの設定のまま進めます。

4. IPv4 アドレス空間の設定を行います。

   設定は以下の表の通りに行います。

項目	設定値
アドレス空間の種類	IPv4
開始アドレス	10.5.0.0
アドレス空間のサイズ	/16 (65536 個のアドレス)

   まず、デフォルトのアドレス空間の削除を行い、改めて IPv4 アドレス空間の追加を行い、表の設定値と同じ値を入れます。

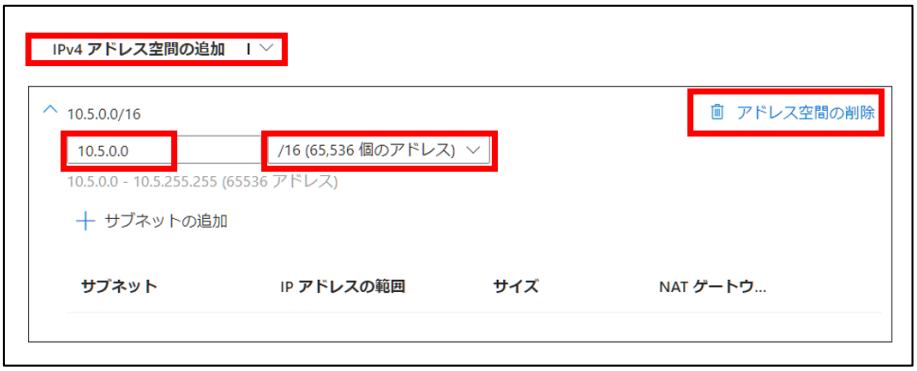

図 4.3-2 IP アドレス空間の設定画面

5. サブネットを二つ追加します。

   サブネットの設定値は表の通りです。

   プライベートサブネットとセキュリティに関しましてはデフォルトのまま進めます。

項目	サブネット 1	サブネット 2
サブネットの目的	Default	Default
名前	LKL-QWK-storage-subnet1	LKL-QWK-storage-subnet2
開始アドレス	10.5.1.0	10.5.2.0
サイズ	/24 (24 個のアドレス)	/24 (24 個のアドレス)

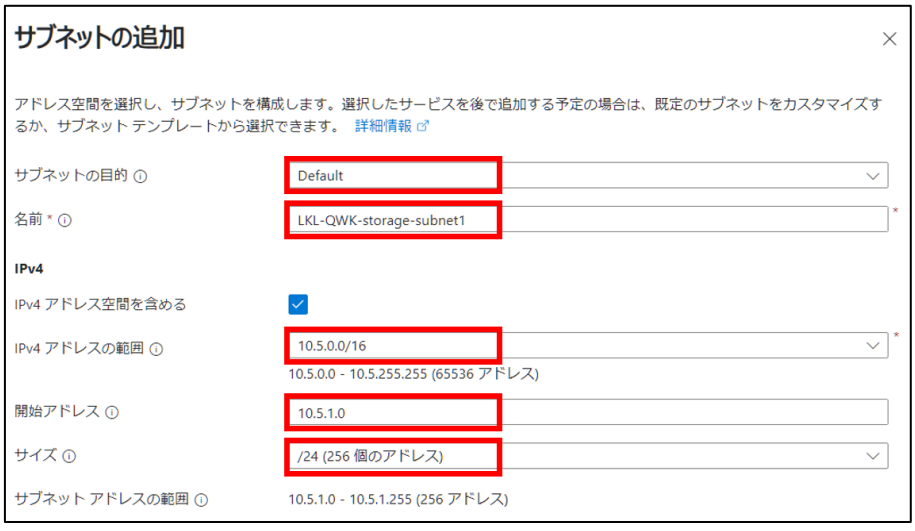

図 4.3-3 サブネットの追加画面

6. 仮想ネットワークを作成します。

4.4 可用性セットの作成

Azure の可用性セットを利用し、インフラストラクチャの冗長設定を行います。

1. Azure 画面上部の検索窓から、仮想ネットワークを検索し、作成ボタンを押します。
2. 以下の通りに可用性セットの設定を行います。記載していない項目につきましてはデフォルトの値で進めます。

項目	設定値
リソースグループ	LKL-QWK-storage
名前	LKL-QWK-storage-availability
地域	(Asia Pacific) Japan East
マネージドディスクを使用	はい (配置)

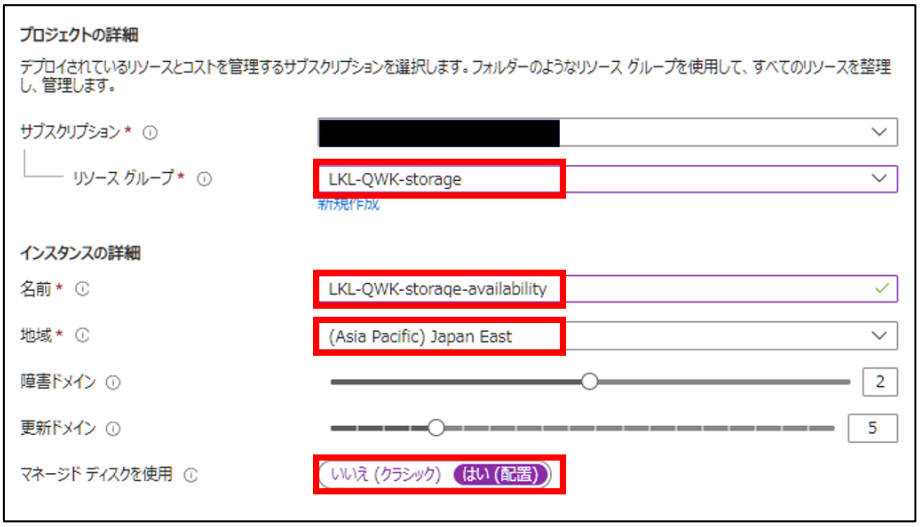

図 4.4-1 可用性セットの設定画面

4.5 仮想マシンの作成

本ドキュメントでは仮想マシンを 3 台作成します。Witness/クライアントノード（以下 Witness ノード）とクラスターノード 2 台（以下ノード A、ノード B）です。

1) Witness ノードの作成

Witness サーバとして使用する仮想マシンを作成します。

また、クラスターノードへ接続するための踏み台サーバと、PostgreSQL のクライアントとしても使用します。

1. Azure 画面上部の検索窓から、仮想ネットワークを検索し、作成ボタンを押しま

す。そして Azure 仮想マシンを選択します。

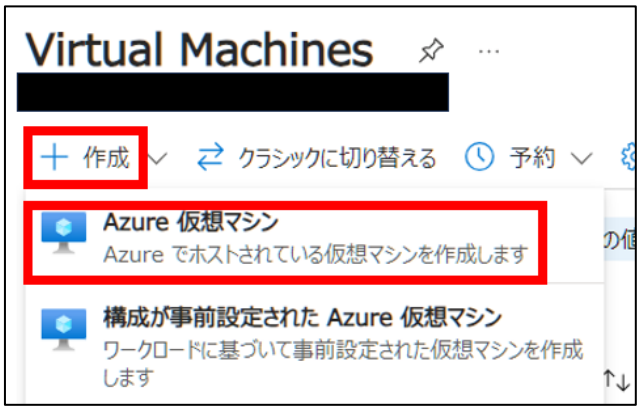

図 4.5-1 仮想マシンの作成

2. 以下の通りに「基本」タブの設定を行います。
記載していない項目はデフォルトのまま進めます。

項目	設定値
リソースグループ	LKL-QWK-storage
仮想マシン名	Witness-node
地域	(Asia Pacific) Japan East
可用性オプション	インフラストラクチャ冗長は必要ありません
セキュリティの種類	Standard
イメージ	Red Hat Enterprise Linux バージョンは 9.2(LVM) - x64 Gen 2
サイズ	Standard_B2s -2 vcpu, 4GiB のメモリ
認証の種類	パスワード
ユーザ名	lkadmin
パスワード	********(自分で設定した値)
パブリック受信ポート	選択したポートを許可する
受信ポートを選択	SSH (22)

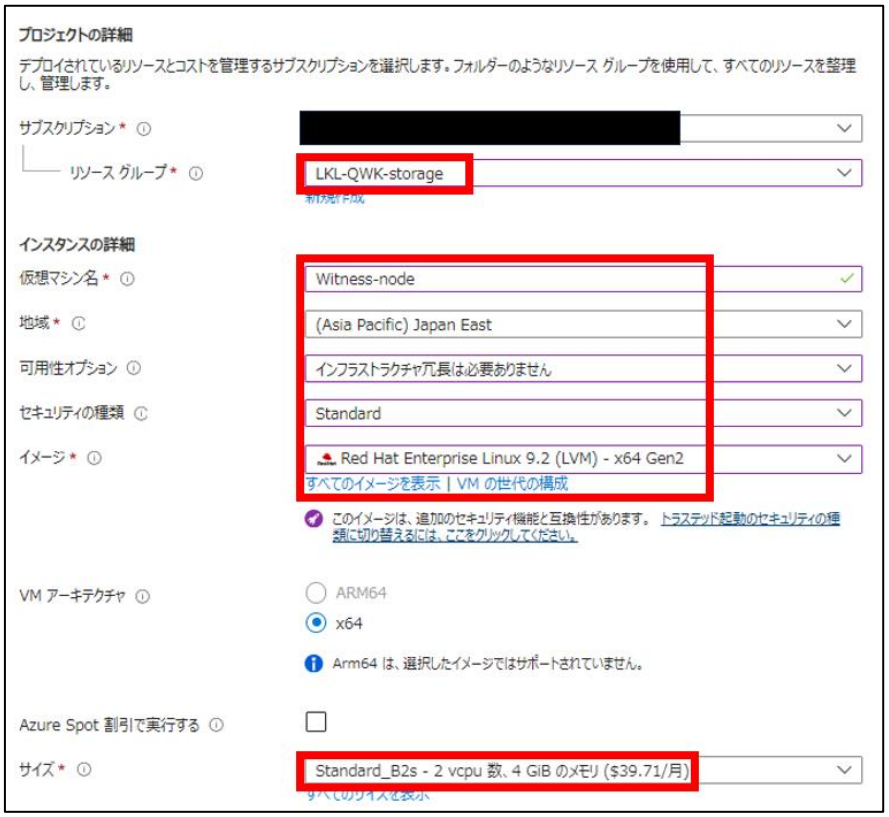

図 4.5-2 基本設定 1

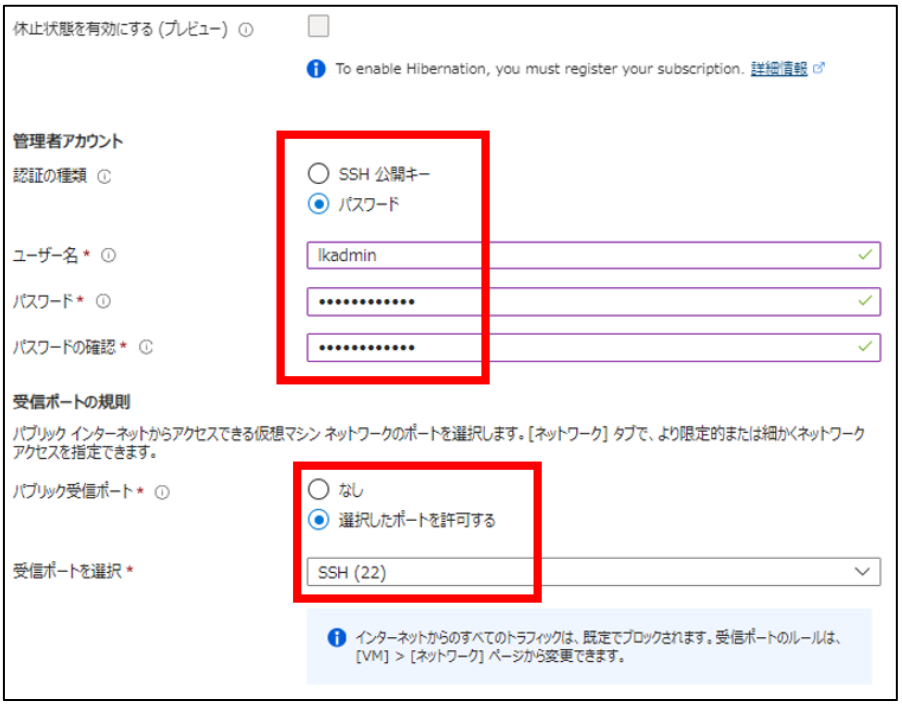

図 4.5-3 基本設定 2

3. 「ディスク」タブの設定をします。

OS ディスクの種類を Premium SSD から Standard SSD に変更します。

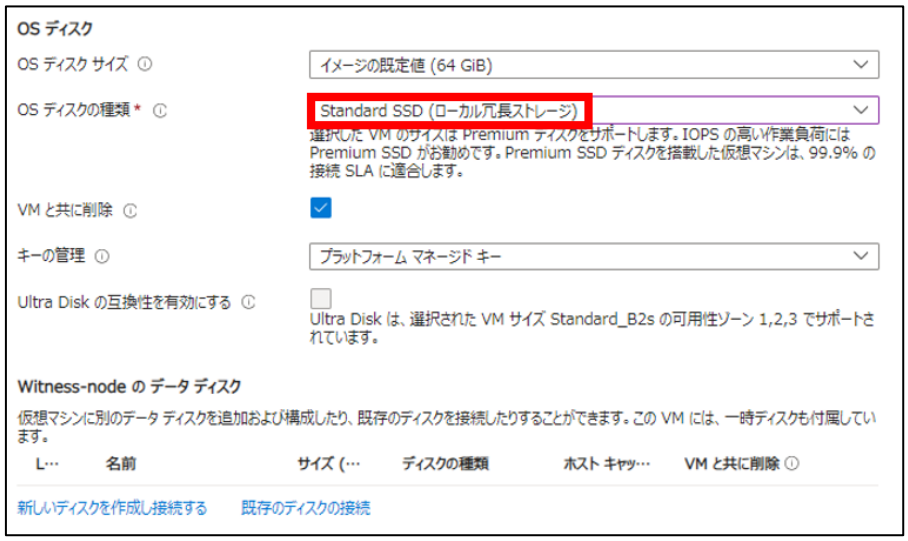

図 4.5-4 ディスクの設定

4. 「ネットワーク」の設定を行います。

以下の表の通りに設定を行います。

他の設定項目はデフォルトの値を使用します。

項目	設定値
仮想ネットワーク	LKL-QWK-storage-Vnet
サブネット	LKL-QWK-storage-subnet1 (10.5.1.0/24)
パブリック IP	新規作成 (デフォルトの値を使用)
VM が削除されたときにパブリック IP と NIC を削除する	チェックを入れる

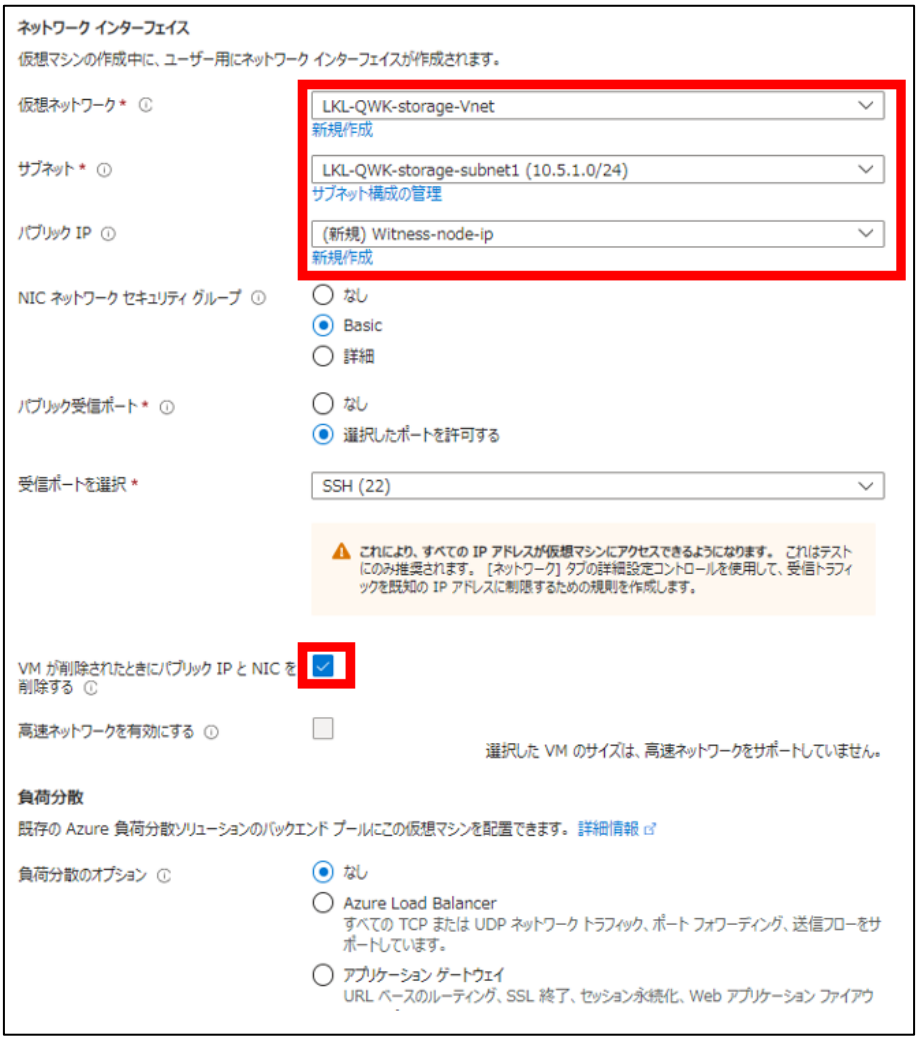

図 4.5-5 ネットワークの設定

5. 「管理・監視・詳細」タブの設定をします。
   LifeKeeper のクラスター構築に必要な設定はないため、設定はスキップします。
   必要に応じて適宜設定してください。

6. 仮想マシンを作成します。

2) クラスターノードの作成

1. 仮想マシンを作成します。Witness ノードの作成手順と同様に行います。

2. 「基本」タブの設定です。Witness ノードと変更になる設定は赤字で示しています。記載していない項目につきましてはデフォルトの値で進めます。

項目	設定値
リソースグループ	LKL-QWK-storage
仮想マシン名	Clstr-nodeA
地域	(Asia Pacific) Japan East
可用性オプション	可用性セット
可用性セット	LKL-QWK-storage-availability
セキュリティの種類	Standard
イメージ	Red Hat Enterprise Linux バージョンは 9.2(LVM) - x64 Gen 2
サイズ	Standard_B2s -2 vcpu, 4GiB のメモリ
認証の種類	パスワード
ユーザ名	lkadmin
パスワード	********(自分で設定した値)
パブリック受信ポート	なし

3. 「ディスク」タブの設定をします。

   OS ディスクの種類を Premium SSD から Standard SSD に変更します。

   新しいディスクを作成します。

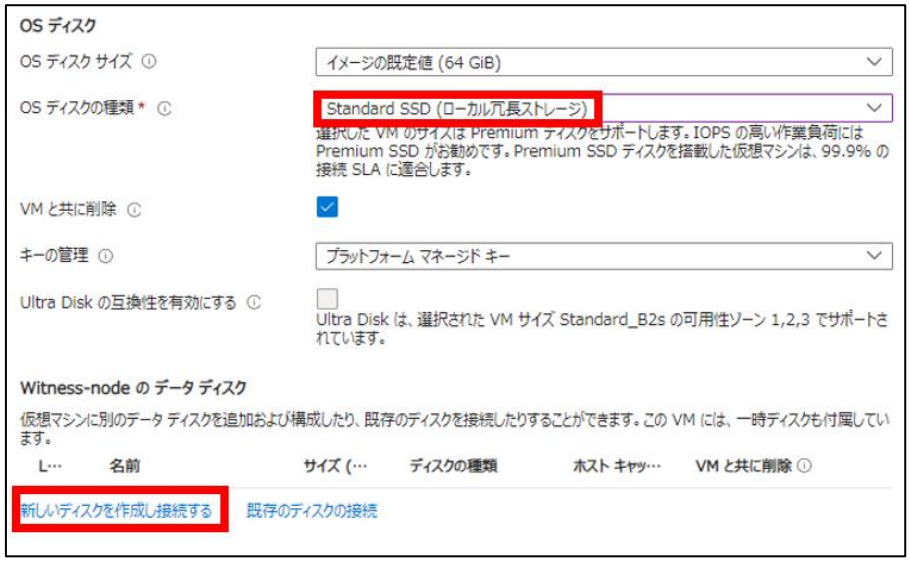

図 4.5-6 ディスクの設定

新しいディスクではサイズを変更します。

また、「VM とともにディスクを削除」にチェックを入れます。

図 4.5-7 新しいディスクの作成

ストレージの種類は Standard SSD (ローカル冗長ストレージ)、サイズは 16GiB です。

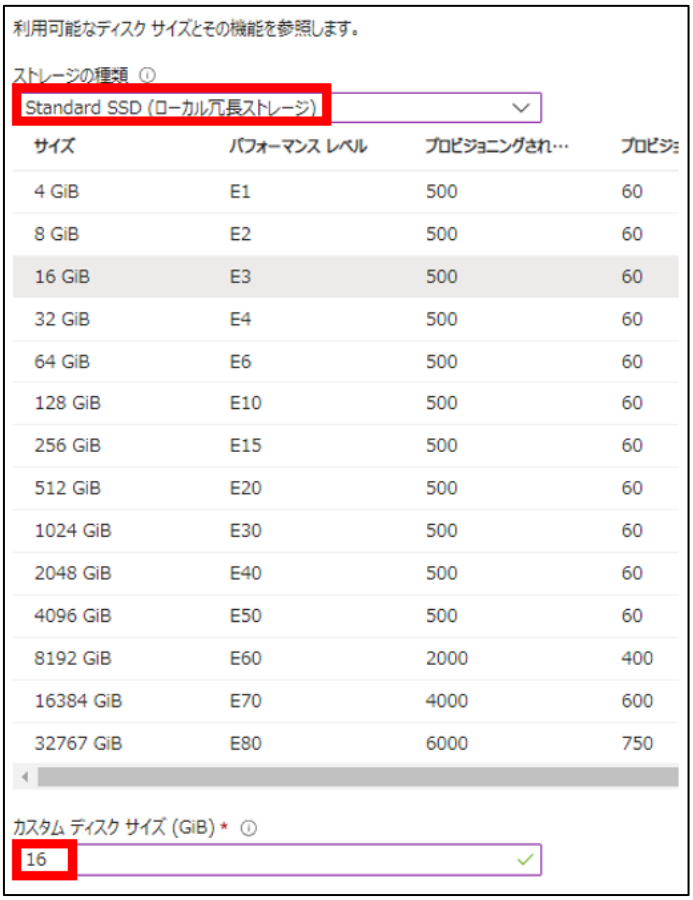

図 4.5-8 新しいディスクの設定

4. 「ネットワーク」の設定を行います。

以下の表の通りに設定を行います。

他の設定項目はデフォルトの値を使用します。

項目	設定値
仮想ネットワーク	LKL-QWK-storage-Vnet
サブネット	LKL-QWK-storage-subnet1 (10.5.1.0/24)
パブリック IP	なし
VM が削除されたときにパブリック IP と NIC を削除する	チェックを入れる

5. 残りの操作は Witness ノードと同様の操作を行い、仮想マシンを作成します。

3) クラスターノード B の作成

1. 仮想マシンを作成します。ノード A の作成手順と同様に行います。以下の通りに設定します。

「基本」タブの設定

項目	設定値
リソースグループ	LKL-QWK-storage
仮想マシン名	Clstr-nodeB
地域	(Asia Pacific) Japan East
可用性オプション	可用性セット
可用性セット	LKL-QWK-storage-availability
セキュリティの種類	Standard
イメージ	Red Hat Enterprise Linux バージョンは 9.2(LVM) - x64 Gen 2
サイズ	Standard_B2s -2 vcpu, 4GiB のメモリ
認証の種類	パスワード
ユーザ名	lkadmin
パスワード	********(自分で設定した値)
パブリック受信ポート	なし

「ディスク」タブの設定

項目	設定値
OS ディスクの種類	Standard SSD
ストレージの種類	Standard SSD

ディスクのサイズ	16GiB

「ネットワーク」タブの設定

項目	設定値
仮想ネットワーク	LKL-QWK-storage-Vnet
サブネット	LKL-QWK-storage-subnet1 (10.5.1.0/24)
パブリック IP	なし
VM が削除されたときにパブリック IP と NIC を削除する	チェックを入れる

4.6 NIC の設定

1. 各 NIC に振られる IP アドレスを静的 IP アドレスに変更します。

まず、対象の仮想マシンの概要画面から、「ネットワーク設定」 -> 「ネットワークインターフェイス」へ進みます。

図 4.6-1 仮想マシンの概要画面

ネットワークインターフェイスの概要画面に移動後、「IP 構成」 -> 「ipconfig1」へ進みます。

図 4.6-2 IP 構成編集画面

続いて、NIC に割り当てられている IP を静的にし、割り振るアドレスを入力します。

図 4.6-3 IP アドレス設定画面

上記の操作を全てのノードで行います。静的 IP は以下のように設定します。

ノード	IP アドレス
Clstr-nodeA	10.5.1.11
Clstr-nodeB	10.5.1.12
Witness-node	10.5.1.20

2. 各ノードにデータレプリケーション用の NIC を追加します。

再度対象の仮想マシンの概要画面から、「ネットワーク設定」 -> 「ネットワークインター

フェイス」へ進みます。ここではノード A の場合の設定を行います。

NIC を追加するため、「ネットワークインターフェイスのアタッチ」を押します。

図 4.6-4 NIC の追加

追加する NIC を新規作成します。

図 4.6-5 新しい NIC の作成

以下の表の通りに設定します。

他の値はデフォルトの値のまま進めます。

項目	設定値
リソースグループ	LKL-QWK-storage
ネットワークインターフェイスの名前	Clstr-nodeA-Vnet
サブネット	LKL-QWK-storage-subnet2 (10.5.2.0/24)
プライベート IP アドレスの割り当て	静的
プライベート IP アドレス	10.5.2.11

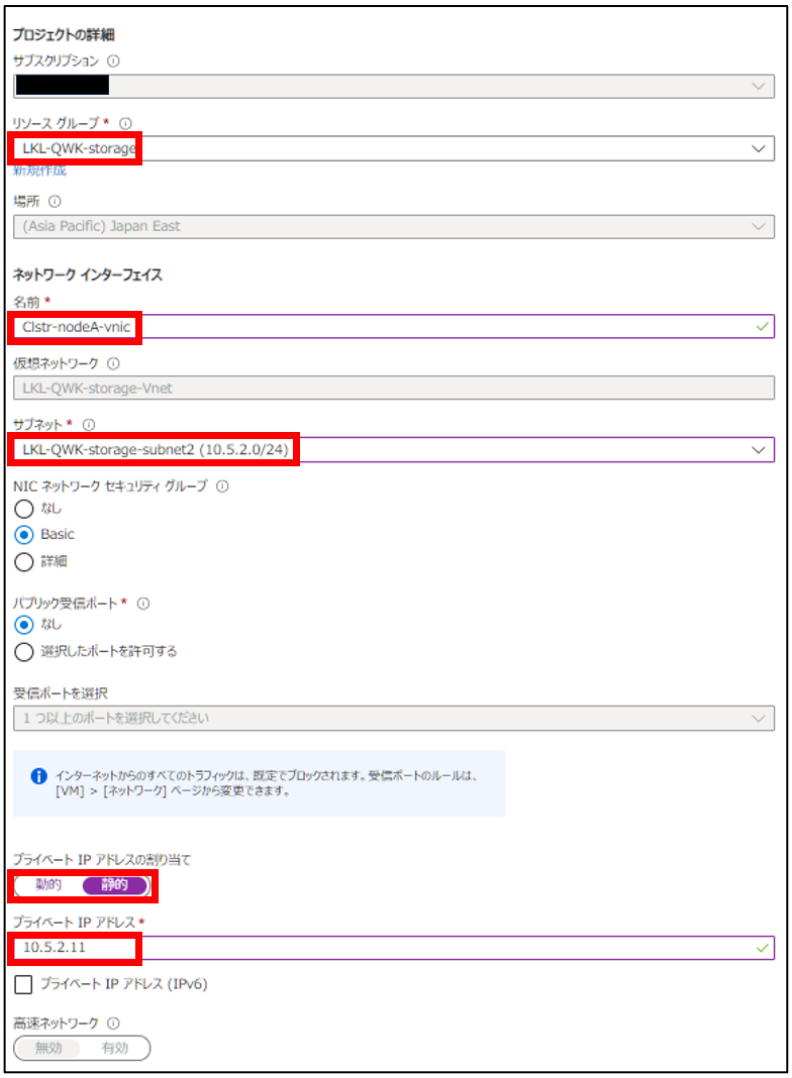

図 4.6-6 ネットワークインターフェイスの設定

ノード B も上記の手順で NIC を追加します。

追加時の設定値は以下の表の通りになります。

ノード B

項目	設定値
リソースグループ	LKL-QWK-storage
ネットワークインターフェイスの名前	Clstr-nodeB-Vnet
サブネット	LKL-QWK-storage-subnet2 (10.5.2.0/24)
プライベート IP アドレスの割り当て	静的
プライベート IP アドレス	10.5.2.12

4.7　インターナルロードバランサー(ILB)の作成

Azure では IP リソースが保護する仮想 IP アドレスを Azure の仮想ネットワークで認識することができません。この影響で、通常 LifeKeeper for Linux が想定している仮想 IP アドレスによるネットワーク通信を行うことができません。そのため LifeKeeper では ILB の導入を行い、ILB が設定する仮想 IP アドレスをネットワーク通信経路として設定します。

詳細につきましてはテクニカルドキュメントからご確認いただけます。

https://docs.us.sios.com/spslinux/9.8.0/ja/topic/lifekeeper-specific-configurations-in-azure

1. ロードバランサーの作成を作成します。Azure 画面上部の検索窓から、仮想ネットワークを検索し、作成ボタンを押します。
2. 以下の表の通りに「基本」タブの設定をします。記載していない項目につきましてはデフォルトの値で進めます。

項目	設定値
リソースグループ	LKL-QWK-storage
ロードバランサーの名前	Clstr-nodeB-Vnet

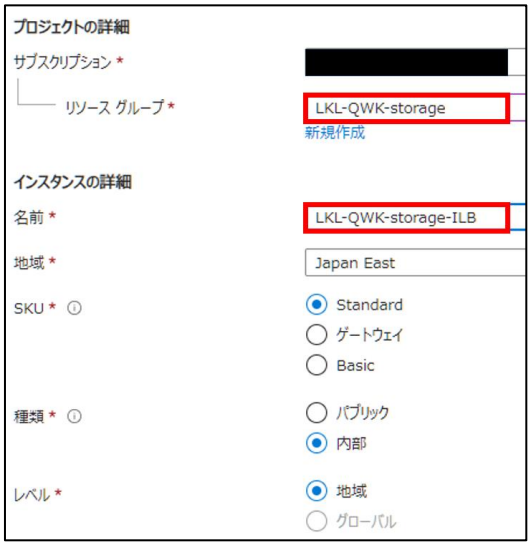

図 4.7-1 ロードバランサーの設定

3. 続いて、「フロントエンド IP 構成」タブの設定をします。
   フロントエンド IP 構成の追加ボタンを押します。
   次の表に従い、フロントエンド IP の設定を行います。

項目	設定値
名前	LKL-QWK-storage-FEIP
仮想ネットワーク	LKL-QWK-storage-Vnet
サブネット	LKL-QWK-storage-subnet1 (10.5.1.0/24)
割り当て	静的
IP アドレス	10.5.1.100
可用性ゾーン	ゾーン冗長

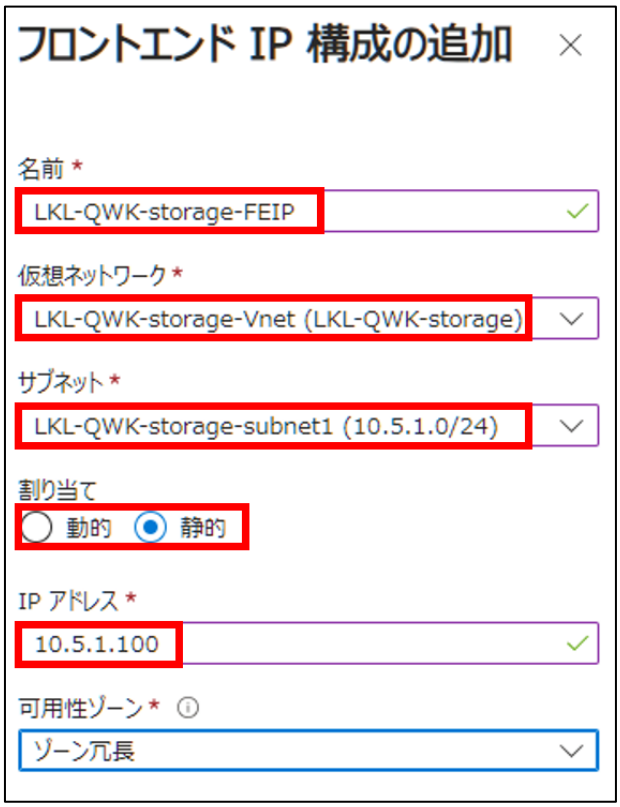

図 4.7-2 フロントエンド IP 構成の設定

設定を終えたら追加ボタンを押し、フロントエンド IP 構成を追加します。

4. 続いて、バックエンドプールを追加します。
バックエンドプールの名前を付け、IP 構成の NIC 追加を行います。

図 4.7-3 バックエンドプールの追加

追加する NIC はノード A とノード B の 10.5.1.0/24 内にある NIC を追加します。

図 4.7-4 NIC の選択

5. 負荷分散規則を追加します。「インバウンド規則」タブから「負荷分散規則の追加」へ進みます。

「基本」タブでは、以下の表の通りに設定します。記載していない項目はデフォルトの値のまま進めます。

項目	設定値
名前	LKL-QWK-storage-rule
フロントエンド IP アドレス	LKL-QWK-storage-FEIP (10.5.1.100)
バックエンドプール	LKL-QWK-storage-BEP

ポート	5432
バックエンドポート	5432
フローティング IP を有効にする	チェック

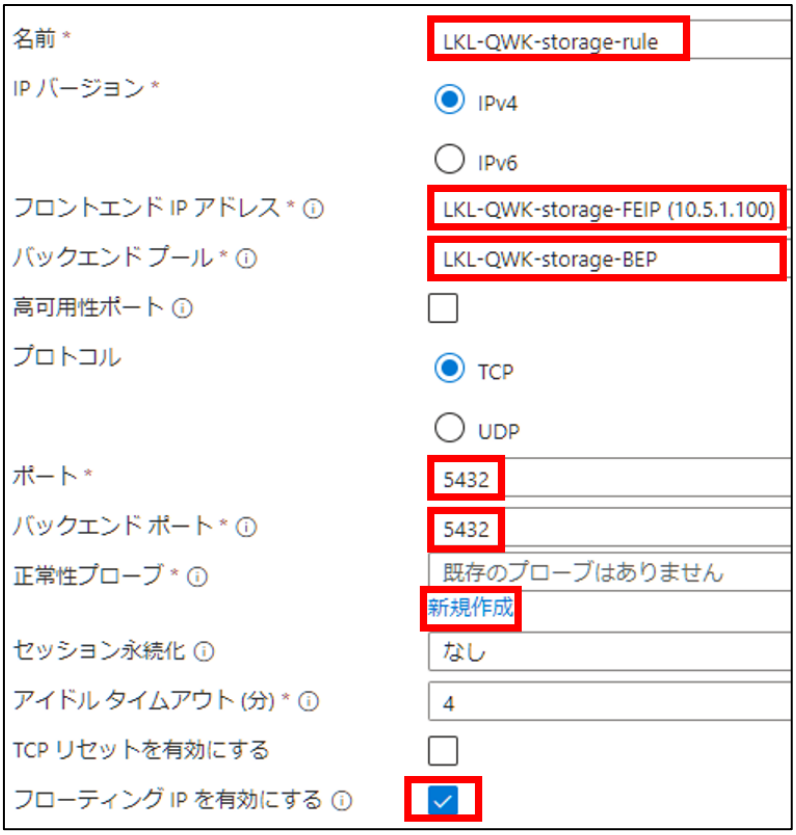

図 4.7-5 負荷分散規則の作成

正常性プローブは新規作成をします。以下の表の通りに設定します。

項目	設定値
名前	LKL-QWK-storage-probe
ポート	12345[1]

[1] 本ドキュメントではポート番号を 12345 とします。お客様の環境に合わせて設定してください。ここで設定したポート番号は後述する LB Health Check リソース作成に使用します。

図 4.7-6 正常性プローブの作成

以上で Azure 上の操作は終了です。

5. OS の設定

5.1　仮想マシンのログイン

使用する ssh クライアントについて

LifeKeeper の GUI を使用するために、X Window システムを使います。そのため、X Window システムを利用できる ssh クライアントをお選びください。

1.　パブリック IP アドレスの確認

Witness ノードのパブリック IP アドレスを確認します。

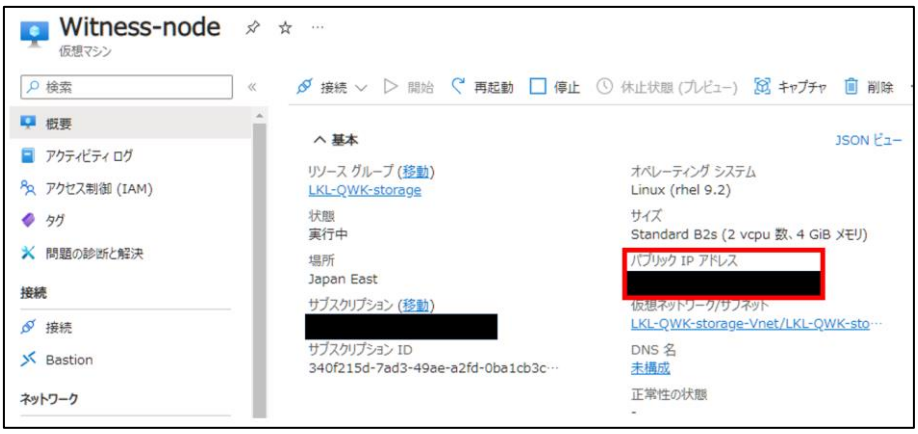

図 5.1-1 パブリック IP アドレスの確認

2.　Witness ノードへログイン

ssh クライアントを利用して Witness ノードへ ssh を用いてログインします。

ssh ログインにはパブリック IP、ユーザ名、パスワードが要求されるので入力しログインします。

3.　クラスターノードへログイン

Witness ノードへログインができたら、クラスターノードへ ssh ログインができることを確認します。

クラスターノードへ ssh 接続

```
$ ssh XXX.XXX.XXX.XXX
```

パスワードが要求されるので入力します。

接続が確認出来たらログアウトします。

```
$ exit
```

両方のノードでログインできることを確認してください。

5.2 ssh の設定

1. 公開鍵認証の設定

ノード A とノード B へ公開鍵認証が可能となるよう設定を行います。

Witness ノードでキーペアを作成します。

```
$ ssh -keygen -t rsa
```

上記コマンド実行後、/home/lkadmin/.ssh/ 配下に秘密鍵 id_rsa と id_rsa.pub が生成されます。

続いて、クラスターノードへ公開鍵を転送します。

```
$ ssh-copy-id lkadmin@XXX.XXX.XXX.XXX
```

2. sshd_config の編集

パスワード認証の無効化と X11 Forwarding の有効化を行います。

/etc/ssh/sshd_config の一部を以下のように変更します。

公開鍵認証の有効化 **(すべてのノードで実施します。)**

変更前	変更後
PasswordAuthentication yes	PasswordAuthentication no

X11 Forwarding の有効化 **(すべてのノードで実施します。)**

変更前	変更後
X11Forwarding no	X11Forwarding yes

最後にすべてのノードで sshd サービスを再起動します。

```
# systemctl restart sshd
```

X Server を有効化以降は クラスターノードへ ssh 接続する際に、-X オプションをつけることで、クラスターノード上の GUI 画面をローカルの画面で表示することができます。

```
$ ssh -X lkadmin@XXX.XXX.XXX.XXX
```

5.3 NIC アドレスの固定化

Azure インフラストラクチャ更新に伴って NIC アドレスが更新されることを防ぐため、NIC アドレスを固定します。

クラスターノードで実施します。

1. NIC の情報の確認

仮想マシンの NIC 情報を表示し、各インターフェース名と MAC アドレスを控えます。

```
[lkadmin@Clstr-nodeA ~]$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.5.1.11  netmask 255.255.255.0  broadcast 10.5.1.255
        inet6 fe80::20d:3aff:fecc:fbda  prefixlen 64  scopeid 0x20<link>
        ether 00:0d:3a:cc:fb:da  txqueuelen 1000  (Ethernet)
        RX packets 67109  bytes 14595447 (13.9 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 82566  bytes 19457398 (18.5 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.5.2.11  netmask 255.255.255.0  broadcast 10.5.2.255
        inet6 fe80::a2f9:e168:ac0a:bcf8  prefixlen 64  scopeid 0x20<link>
        ether 60:45:bd:64:e0:4c  txqueuelen 1000  (Ethernet)
        RX packets 2  bytes 762 (762.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 30  bytes 2584 (2.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

2. 設定ファイルの編集

デフォルトでは eth0 のネットワーク設定ファイルしかないため、 eth1 のネットワーク設定ファイルを作成します。

```
# cd /etc/sysconfig/network-scripts/
# cp ifcfg-eth0 ifcfg-eth1
```

eth0、 eth1 それぞれの設定ファイルを編集します。 HWADDR には、前の手順で控えた値を使用します。

eth0 の例

```
# Created by cloud-init on instance boot automatically, do not edit.
#
AUTOCONNECT_PRIORITY=999
BOOTPROTO=dhcp
DEVICE=eth0
```

```
HWADDR=00:0d:3a:cc:fb:da
ONBOOT=yes
TYPE=Ethernet
USERCTL=no
```

eth1 の例

```
# Created by cloud-init on instance boot automatically, do not edit.
#
AUTOCONNECT_PRIORITY=999
BOOTPROTO=dhcp
DEVICE=eth1
HWADDR=00:0d:3a:cc:fb:da
ONBOOT=yes
TYPE=Ethernet
USERCTL=no
```

5.4 SELinux の無効化

SELinux が enforcing モードの場合、LifeKeeper は動作しません。

また、必要な場合を除いて、permissive モードは推奨されません。

そのため、以下のコマンドを実行し、SELinux を無効化させます。

クラスターノードで実施します。

```
# sed -i 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/selinux/config
```

変更は再起動後に適用されます。

LifeKeeper 上で SELinux を Permissive モードで動作させる事を検討している場合は以下のページをご参照ください。

LifeKeeper for Linux v9.8.0 テクニカルドキュメンテーション – インストール前の要件
https://docs.us.sios.com/spslinux/9.8.0/ja/topic/installing-the-software

5.5 ファイアウォールの無効化

このセクションでは、SIOS Protection Suite for Linux をインストールするために、firewalld サービスを無効にします。ファイアウォールは有効なままにしておくことができますが、SIOS Protection Suite と保護するアプリケーションに必要なポートを設定する必要があります（ここでは示していません）。

以下のコマンドを実行し、ファイアウォールを無効化させます。

すべてのノードで実施します。

```
# systemctl disable firewalld.service --now
```

5.6　ホスト名の名前解決

ノード間の名前解決のために /etc/hosts ファイルにホスト名と IP アドレスを追記します。/etc/hosts に以下を追加します。

すべてのノードで実施します。

```
10.5.1.11 Clstr-nodeA
10.5.1.12 Clstr-nodeB
10.5.1.20 Witness-node
```

6. ローカルリポジトリの設定

LifeKeeper のインストール時に 、各 OS のリポジトリから依存関係にあるソフトウェアのインストールも同時に行います。Azure 上の Red Hat 仮想マシンは Azure の RHUI をリポジトリとして利用することができますが、オフラインノード上からは Azure の RHUI にアクセスできません。
そのため、OS のインストールイメージを使用しローカルリポジトリを設定することで、オフラインノード上でも LifeKeeper のインストールを行うことができます。

6.1 /home の拡張

Azure では RHEL 仮想マシンを作成後に明示的に /home の拡張を行う必要があります。
以下のコマンドを実行してください。**すべてのノードで実施します。**

```
$ cd /
$ sudo su -
# cd /
# umount /home
# lvextend -L +16384 /dev/mapper/rootvg-homelv
# mount /home
# xfs_growfs /dev/mapper/rootvg-homelv
```

6.2 OS イメージのマウント

次に、RHEL のインストールイメージをローカル PC からクラスターノードに転送します。
今回、仮想マシンのイメージに使用した OS は Red Hat Enterprise Linux 9.2 のため、
Red Hat 公式から rhel-9.2-x86_64-dvd.iso を事前にダウンロードしておき、
始めに踏み台となる Witness サーバへ転送、その後クラスターノードに転送します。
転送には主に scp を用います。

OS イメージはマウントする必要があります。
fstab を編集することで、OS の起動時に自動的に OS イメージをマウントされ、ローカルリポジトリを楽に使用することができます。**以下の手順はすべてのノードで実施します。**
マウントポイントを作成します。

```
# mkdir /media/cdrom
```

/etc/fstab を編集し、以下を追加します。

```
/home/lkadmin/rhel-9.2-x86_64-dvd.iso /media/cdrom iso9660 ro,loop 0 0
```

マウントを行います。

```
# mount /media/cdrom
```

正常に実行されたら、fstab の編集は終了になります。

6.3 ローカルリポジトリの作成

ローカルリポジトリを以下の設定で /etc/yum.repos.d/rhel-dvd.repo に作成します。

すべてのノードで実施します。

```
[rhel-dvd-BaseOS]
name=Red Hat Enterprise Linux 9.2-x86_64-DVD
baseurl=file:///media/cdrom/BaseOS/
enabled=1
gpgcheck=1
gpgkey=file:///media/cdrom/RPM-GPG-KEY-redhat-release

[rhel-dvd-AppStream]
name=Red Hat Enterprise Linux 9.2-x86_64-DVD
baseurl=file:///media/cdrom/AppStream/
enabled=1
gpgcheck=1
gpgkey=file:///media/cdrom/RPM-GPG-KEY-redhat-release
```

6.4 rh-cloud-base.repo の無効化

ローカルリポジトリの設定が終了したため、Azure の RHUI を以下のコマンドを使用し、無効にします。

すべてのノードで実施します。

```
# sed -i 's/enabled=1/enabled=0/g' /etc/yum.repos.d/rh-cloud-base.repo
```

6.5 ローカルリポジトリの確認

ローカルリポジトリが正しく設定されているか確認します。
以下のコマンドでパッケージマネージャーが使用しているリポジトリのリストを確認できます。

正常にローカルリポジトリの設定が完了していると、以下のように表示されます。

```
[lkadmin@Clstr-nodeA ~]$ yum repolist
repo id                       repo name
rhel-dvd-AppStream            Red Hat Enterprise Linux 9.2-x86_64-DVD
rhel-dvd-BaseOS               Red Hat Enterprise Linux 9.2-x86_64-DVD
```

6.6 GUI 設定

全ノード上で GUI の ssh 転送に必要なソフトウェアのインストールを行います。

以下のコマンドを実行し、X11 関連パッケージをインストールします。

すべてのノードで実施します。

```
# dnf install xhost xauth
```

また、LifeKeeper の管理を root ユーザで行うため、/root/.Xauthority に GUI が使用する DISPLAY 変数を格納する必要があります。

そのため、以下のコマンドを実行します。

すべてのノードで実施します。

```
$ echo "sudo xauth add \$(xauth -f /home/lkadmin/.Xauthority list|tail -1)"\
>> .bashrc
```

ここまで終了したら、LifeKeeper のインストールの前にすべてのノードを再起動します。

7. LifeKeeper のインストール

ここでは LifeKeeper のインストール方法を説明します。

7.1 LifeKeeper インストールイメージの転送

6.2 と同様に、クラスターノードに LifeKeeper のインストールイメージをローカル PC から転送します。

7.2 root パスワードの変更

LifeKeeper の GUI 表示には root パスワードが必要となるため、ルートパスワードの設定を行います。
パスワードをクラスター全体で共通にすることで、LifeKeeper の GUI ログインを 1 回で済ませることができます。

```
#su -
# passwd
Changing password for user root.
New password:<新しいパスワード>
Retype new password:<新しいパスワード(確認)>
passwd: all authentication tokens updated successfully.
```

7.3 LifeKeeper のインストール

LifeKeeper のインストールを行います。
詳しくは LifeKeeper for Linux v9.8.0 スタートアップガイドを参照してください。
スタートアップガイド

以下の操作はクラスターノード両方で行います。

1. インストールメディアに格納されている setup スクリプトを実行します。
   実行時、ファイアウォールが無効になっているにも関わらす、ファイアウォールの警告文が表示されることがありますが、無視してそのまま進めてください。

2. インストール画面では以下の項目のチェック欄 [ ] にチェックを入れます。

[ ] Use Quorum / Witness Functions
Recovery Kit Selection Menu
- Networking ---> [ ] LifeKeeper LB Health Check Kit
- Database ---> [ ] LifeKeeper PostgreSQL RDBMS Recovery Kit
- Storage ---> [ ] DataKeeper for Linux

3. License キーのインストールを行います。

```
# /opt/LifeKeeper/bin/lkkeyins <ライセンスキーのパス>
```

4. ブロードキャスト Ping を無効化します。

   Azure では IP リソースが保護する仮想 IP アドレスを Azure の仮想ネットワークで認識することができません。そのため /etc/default/LifeKeeper で以下のパラメーターを変更します。

変更前	変更後
NOBCASTPING=0	NOBCASTPING=1

5. 以下のコマンドで LifeKeeper を起動します。

```
# /opt/LifeKeeper/bin/lkstart
```

LifeKeeper の管理に使用するコマンドの実行ファイルは /opt/LifeKeeper/bin に
LifeKeeper の管理コマンドのレファレンスマニュアルは /opt/LifeKeeper/man に
それぞれ保存されています。
そのため、以下のコマンドを実行することで、次回以降の root ログイン以降、絶対パスでコマンドを実行する必要がなくなります。

```
echo "PATH=$PATH:/opt/LifeKeeper/bin" >> /root/.bash_profile
echo "MANPATH=$MANPATH:/opt/LifeKeeper/man" >> /root/.bash_profile
echo "export PATH MANPATH" >> /root/.bash_profile
```

7.4 GUI ログイン

lkGUIapp コマンドを使用し、LifeKeeper の GUI へログインします。

```
# /opt/LifeKeeper/bin/lkGUIapp &
```

ユーザ名とパスワードが要求されます。ユーザ名は root、パスワードは 7.2 で設定したものを入力します。

ログインに成功すると以下の画面が表示されます。

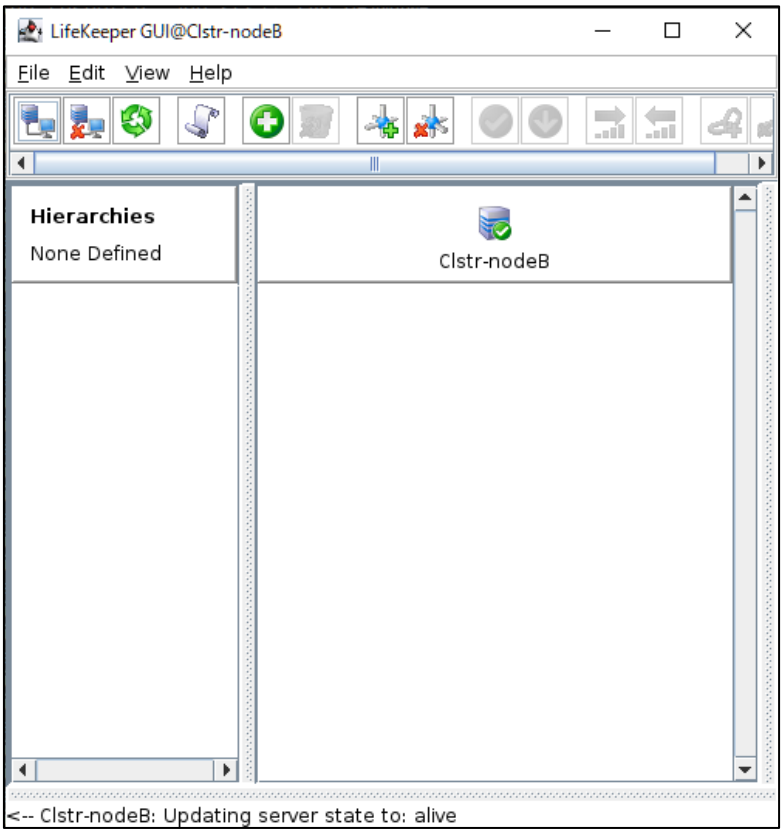

図 7.4-1 LifeKeeper GUI ログイン画面

8. コミュニケーションパス・リソースの作成

各ノードを LifeKeeper のコミュニケーションパスでつなぎ、クラスターを構築し、リソースを保護します。本ドキュメントでは PostgreSQL データベースのデータを DataKeeper で保護します。稼働系ディスクへ書き込みが行われた際は待機系のディスクと同期し、同様のデータになります。PostgreSQL へのアクセスは仮想 IP アドレス 10.5.1.100 を通して行われます。

システム構成図は以下の通りになります。

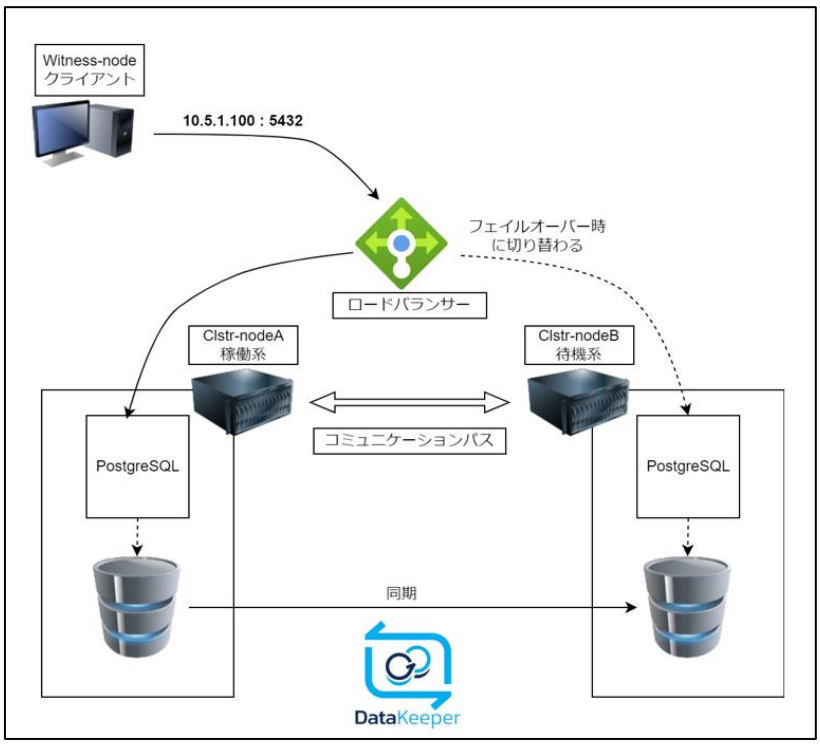

図 8-1 システム構成図

8.1 コミュニケーションパスの作成

1. GUI 画面の赤枠をクリックし、コミュニケーションパスの作成画面に移ります。

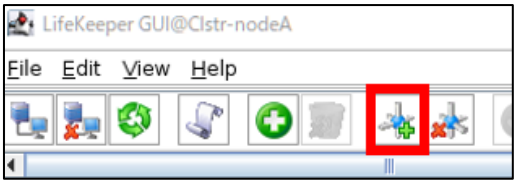

図 8.1-1 コミュニケーションパス作成

下記の表の通りに設定します。

項目	設定値
Local Server	Clstr-nodeA
Remote Server(s)	Clstr-nodeB
Device Type	TCP
Local IP Address(es)	10.5.1.11
Remote IP Address	10.5.1.12
Priority	1

同様に、もう一つコミュニケーションパスを作成します。

項目	設定値
Local Server	Clstr-nodeA
Remote Server(s)	Clstr-nodeB
Device Type	TCP
Local IP Address(es)	10.5.2.11
Remote IP Address	10.5.2.12
Priority	2

コミュニケーションパスを二本作成すると、以下のような状態になります。

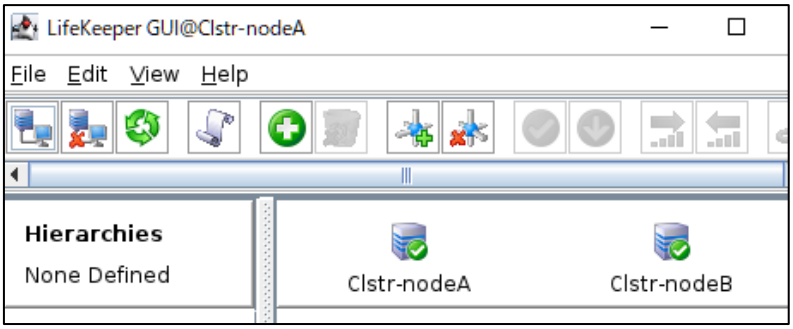

図 8.1-2 コミュニケーションパス作成後

8.2 DataKeeper リソースの作成

仮想マシンを作成した際に追加したディスクを DataKeeper で保護します。
データディスクが共有でなくとも各ノードのデータを同期することで、フェイルオーバーが発生した場合でも整合性のあるデータにアクセスすることができます。

1. リソース作成ウィザードを起動します。
   以下の赤枠のボタンもしくは Edit -> Server -> Create Resource Hierarchy をクリックします。

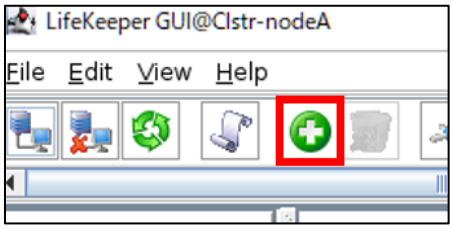

図 8.2-1 リソース作成

2. 以下の通りに設定を進めます。

Create Resource Wizard

項目	設定値
Please Select Recovery Kit	Data Replication
Switchback Type	intelligent
Server	Clstr-nodeA
Hierarchy Type	Replicate New Filesystem
Source Disk	/dev/sda (16.0 GB)[2]
New Mount Point	/data
New Filesystem Type	xfs
Data Replication Resource Tag	datarep-data
File System Resource Tag	/data
Bitmap File	/opt/LifeKeeper/bitmap__data[3]
Enable Asynchronous Replication	no

確認画面に移ります。「Create」をクリックしてください。

作成が完了すると、「Next」のボタンが表示されるので、クリックして次に進みます。

3. Pre Extend ウィザードの設定を行います。

クラスターノードへリソースの拡張を行います。

以下の通りに設定を行います。

Pre-Extend Wizard

項目	設定値
Target Server	Clstr-nodeB
Switchback Type	intelligent
Template Priority	1
Target Priority	10

Pre Extend checks were successful と表示されれば成功です。

[2] お客様環境により表示名が変わります。

[3] デフォルトの値となっています。

「Next」 から次へ進みます。

4. Extend ウィザードの設定を行います。
   以下の通りに設定を行います。

項目	設定値
Target Disk	/dev/sda (16.0 GB)
Data Replication Resource Tag	datarep-data
Bitmap File	/opt/LifeKeeper/bitmap__data
Replication Path	10.5.2.11/10.5.2.12
Mount Point	/data
Root Tag	/data

Hierarchy successfully extended と出力されたら、リソースの拡張も正常に終了です。
Finish を押して DataReplication リソースの作成は以上になります。

DataReplication リソースを作成した後すぐは、データレプリケーションリソースが以下のように Wait to Resync (再同期中)という表示になります。Wait to Resync から Target の表示に変わるまで少しお待ちください。

図 8.2-2 DataReplication リソースの再同期と GUI 表示

作成後は以下のような画面になります。

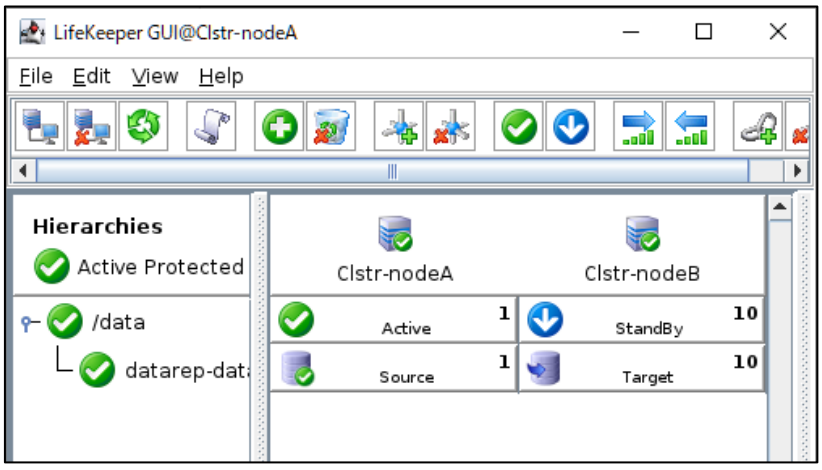

図 8.2-3 DataKeeper リソース作成後の GUI 画面

8.3 スイッチオーバーのテスト

スイッチオーバーのテストを行います。

スイッチオーバーのテストは各リソースの作成時に必ず実施できることを確認してください。

待機系の /data リソースを左クリックし、「In Service」ボタンを押し、正常にスイッチオーバーができるか確認します。

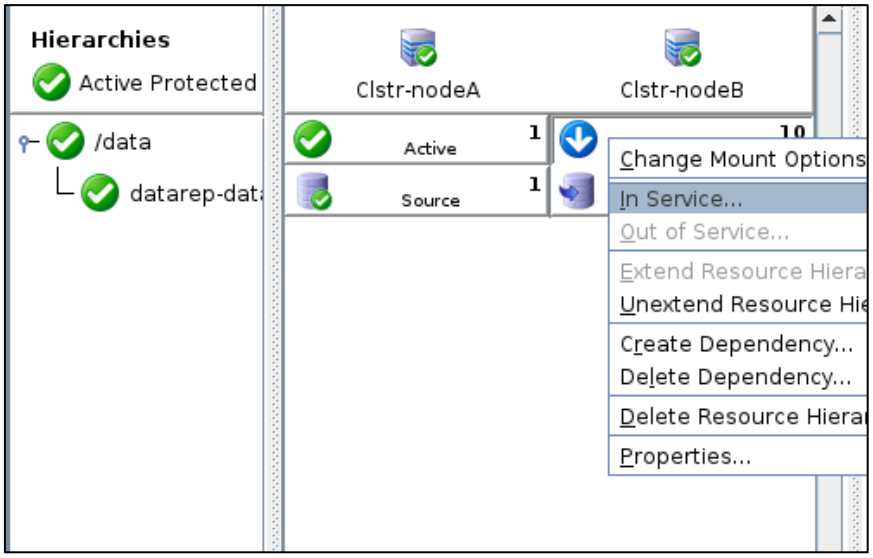

図 8.2-4 スイッチオーバーのテスト

8.4 PostgreSQL リソースの作成

1. PostgreSQL サーバおよびクライアントのインストールをします。

   以下のコマンドで PostgreSQL をインストールします。

   クラスターノードで実施します。

   ```
   # dnf install postgresql-server
   ```

   また、接続の確認を行う Witness ノードでは、以下のコマンドで PostgreSQL クライアントをインストールしてください。

   ```
   # dnf install postgresql
   ```

   PostgreSQL のデータディレクトリを DataKeeper が保護しているディスク上に作成します。

   また、ディレクトリの所有者を postgres ユーザに変更します。

   稼働系ノードがノード A であることを確認し、以下のコマンドを実行します。

稼働系のみで実施します。

```
# mkdir -m 755 -p /data/pgsql/data
# chown -R postgres:postgres /data/pgsql
```

以上でデータディレクトリの作成は終了です。

2. データベースの初期化を行います。

   postgres ユーザになり、データベースの初期化を行います。

   稼働系のみで実施します。

```
# su – postgres
$ initdb -D /data/pgsql/data
```

3. PostgreSQL の設定変更を行います。

   稼働系のみで実施します。

   以下の通りに変更します。

   /data/pgsql/data/pg_hba.conf の設定

変更前	変更後
# IPv4 local connections: host all all 127.0.0.1/32 trust	# IPv4 local connections: host all all 10.5.0.0/16 trust
# replication privilege. local replication all trust host replication all 127.0.0.1/32 trust	# replication privilege. local replication all trust host replication all 10.5.0.0/16 trust

   /data/pgsql/data/postgresql.conf の設定

変更前	変更後
# - Connection Settings – #listen_addresses = ‘localhost' #port = 5432	# - Connection Settings – listen_addresses = ‘*’ port = 5432

4. PostgreSQL デーモンの起動と確認をします。

   稼働系と Witness ノードで実施します。

   ノード A にて、以下のコマンドで PostgreSQL を起動します。

```
$ pg_ctl start -D /data/pgsql/data -p “-p 5432” -w
```

   稼働系では以下のコマンドで PostgreSQL の起動を確認できます。

   (postgres ユーザで実施してください)

```
$ psql -l
```

また、Witness ノード (PostgreSQL クライアント) からは以下のコマンドで正常に接続できることを確認します。

```
$ psql -h 10.5.1.11 -U postgres -d postgres
```

正常に接続ができた場合、以下のプロンプトが返ってきます。

```
psql (13.10)
Type "help" for help.


postgres=# q
```

5. PostgreSQL リソースの作成を行います。
   DataReplication リソースを作成時と同様に、Create Resource Hierarchy を選択します。
   以下の表に従ってリソースの作成を進めます。

Create Resource Wizard

項目	設定値
Please Select Recovery Kit	PostgreSQL Database
Swithchback Type	intelligent
Server	Clstr-nodeA
PostgreSQL Executable Location	/usr/bin
PostgreSQL Client Executable Location	/usr/bin/psql
PostgreSQL Administration Executable Location	/usr/bin/pg_ctl
PostgreSQL Data Directory	/data/pgsql/data
PostgreSQL Port	5432
PostgreSQL Socket Path	/var/run/postgresql/.s.PGSQL.5432
Enter Database Administrator User	postgres
PostgreSQL Logfile	/tmp/pgsql-5432.lk.log
PostgreSQL Database Tag	pgsql-5432

Pre-Extend Wizard

項目	設定値
Target Server	Clstr-nodeB
Switchback Type	intelligent
Template Priority	1
Target Priority	10

Extend Wizard

項目	設定値
PostgreSQL Executable Location	/usr/bin
PostgreSQL Database Tag	pgsql-5432

正常に PostgreSQL リソースが追加されると、GUI 上で PostgreSQL が追加されます。

6. スイッチオーバーのテストを行います。
   正常に LifeKeeper で保護できていることを確認します。

8.5 LB Health Check リソースの作成

LB Health Check Kit は、ロードバランサーの負荷分散対象インスタンスへのヘルスチェックプローブを待ち受けて応答する仕組みを提供します。
導入することで IP リソースによる仮想 IP アドレスの切り替えが使用可能となります。

1. リソースの作成を行います。
   これまでのリソース作成時と同様に、Create Resource Hierarchy を選択します。
   以下の表に従ってリソースの作成を進めます。

Create Resource Wizard

項目	設定値
Please Select Recovery Kit	LB Health Check
Swithchback Type	intelligent
Server	Clstr-nodeA
Reply daemon Port	12345[4]
Reply daemon message	(何も入力しません)
LB Health Check Resource Tag	lbhc-12345

Pre-Extend Wizard

項目	設定値
Target Server	Clstr-nodeB
Switchback Type	intelligent
Template Priority	1
Target Priority	10

[4] 4.7 で正常性プローブのポート番号で 12345 以外の値を設定した方は、正常性プローブで設定した値を入力してください。

Extend Wizard

項目	設定値
LB Health Check Resource Tag	lbhc-12345

2. スイッチオーバーのテストを行います。
   正常に LifeKeeper で保護できていることを確認します。

3. Azure ポータル上から LB Health Check リソースの動作を確認することができます。
   ロードバランサー「LKL-QWWK-storage-ILB」の概要ページから、「分析情報」->「Show Metrics Pane」へ進みます。

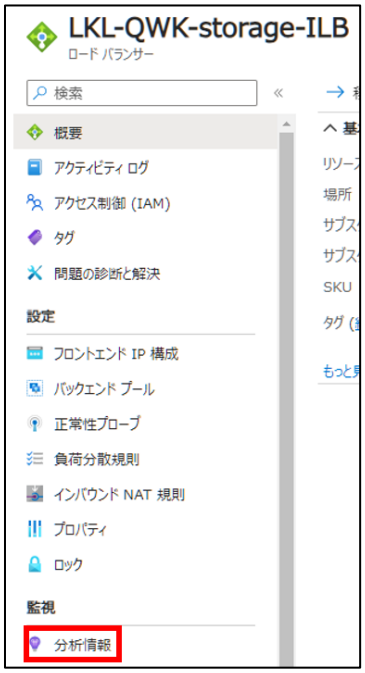

図 8.5-1 ロードバランサー概要画面

図 8.5-2 ロードバランサー分析情報画面

バックエンド IP アドレスごとの正常性プローブの状態を確認します。
スイッチオーバーが正常に動作していると、図のように切り替わりが確認できます。

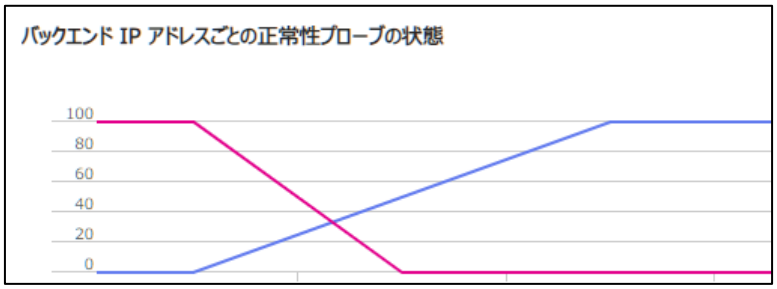

図 8.5-3 正常性プローブ

8.6 IP リソースの作成

PostgreSQL への接続に使用できる仮想 IP アドレスを作成、保護します。フェイルオーバーが発生しても、PostgreSQL への接続先 IP アドレスを変えることなくアクセスすることが可能になります。

1. リソースの作成を行います。
   これまでのリソース作成時と同様に、Create Resource Hierarchy を選択します。
   以下の表に従ってリソースの作成を進めます。

Create Resource Wizard

項目	設定値
Please Select Recovery Kit	IP
Swithchback Type	intelligent
Server	Clstr-nodeA
IP Resource	10.5.1.100[5]
Netmask	255.255.255.255
Network Interface	eth0
IP Resource Tag	ip-10.5.1.100

Pre-Extend Wizard

項目	設定値
Target Server	Clstr-nodeB
Switchback Type	intelligent
Template Priority	1
Target Priority	10

Extend Wizard

項目	設定値

[5] 4.7 フロントエンド IP で設定した IP を入力します。

IP Resouorce	10.5.1.100
Netmask	255.255.255.255
Network Interface	eth0
IP Resource Tag	ip-10.5.1.100

2. スイッチオーバーのテストを行います。
   正常に LifeKeeper で保護できていることを確認します

8.7 リソース依存関係の作成

各リソースの依存関係を定義することで、リソース階層を作成します。
IP リソース作成直後は以下の図のようなリソースの関係になっています。

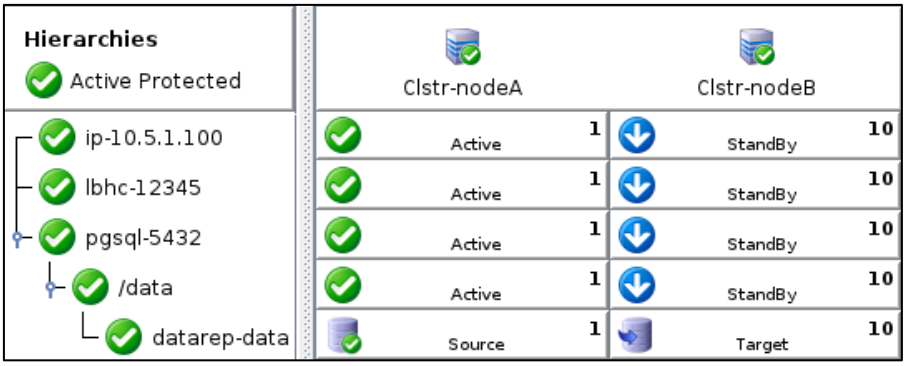

図 8.7-1 IP リソース作成直後のリソース関係

これは、IP リソース、LB Health Check リソース、PostgreSQL リソースに依存関係がなく、それぞれが別のノードで動作し、サービスの提供ができない恐れがあります。そのため、リソース間に適切な依存関係を作成する必要があります。
本ドキュメントでは PostgreSQL リソースの配下に IP リソースと LB Health Check リソースを配置します。

1. IP リソースに依存関係を定義します。
   GUI 上で 「Edit」->「Resource」->「Create Dependency」を選択します。
   以下の表に従い、設定します。

項目	設定値
Server	Clstr-nodeA
Parent Resource Tag	PostgreSQL リソースのタグ名
Child Resource Tag	IP リソースのタグ名

2. LB Health Check リソースに依存関係を定義します。
   上記と同様の手順で行います。子リソースの定義の際に、LB Health Check リソースのタグ名を指定してください。

3. 1 と 2 の手順を完了させると、以下の図のようにリソースの依存関係が作成できます。

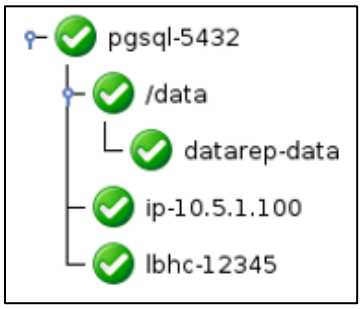

図 8.7-1 完成したリソース依存関係

4. PostgreSQL にすべてのリソースとの依存関係を作成したので、スイッチオーバーのテストを行います。PostgreSQL を待機系で「In Service」にします。最上位のリソースを起動する際は、配下のリソースも立ち上がるので、全リソースが待機系で起動することになります。

5. クライアントから PostgreSQL への接続確認をします。
   以下のコマンドをクライアントノードから使用し、正常に PostgreSQL に接続できるか検証します。

```
$ psql -h 10.5.1.100 -U postgres -d postgres
```

9. QWK の導入と設定

スプリントブレイン対策として、QWK を導入します。
本ドキュメントでは Quorum モードに storage モードを使用します。
storage モードはクラスター内の全てのノードからアクセスできる共有ストレージを用います。
Witness ノードを NFS サーバとすることで、共有ストレージとみなします。

9.1 NFS ファイル共有の設定

1. Witness ノードを NFS サーバとして構築します。
   NFS 構築するには必要なパッケージをインストールする必要があります。
   パッケージは NFS サーバと NFS クライアントの機能を含んでいるため、**すべてのノードでインストールします。**
   以下のコマンドを入力し、インストールします。

```
# dnf install nfs-utils
```

2. NFS サーバ側の設定を行います。
   Witness ノードで実施してください。
   NFS マウントするディレクトリを作成します。

```
# mkdir /root/qwkshare
```

   /etc/expots ファイルを作成します。
   このファイルはリモートホストにどのファイルシステムをエクスポートするかを制御し、オプションを指定します。
   /etc/expots を新規で作成し、以下を追加します。

```
/root/qwkshare  Clstr-nodeA(rw,no_root_squash,sync,no_wdelay)
/root/qwkshare  Clstr-nodeB(rw,no_root_squash,sync,no_wdelay)
```

   このファイルを反映します。
   以下のコマンドを入力し、反映されたことを確認します。

```
# exportfs -v
```

   以下のコマンドを入力します。

```
# exportfs -a
```

以下のコマンドを入力し、反映されたことを確認します。

```
# exportfs -v
```

確認ができたら、NFS サーバのサービスを起動します。

また、OS 起動時に自動でサービスを開始するよう設定します。

```
# systemctl start nfs-server
# systemctl enable nfs-server
```

3. NFS クライアント側の設定を行います。

クラスターノードで以下の操作を実施します。

マウントポイントの作成を行います。

```
# mkdir /mnt/qwkmnt
```

作成したマウントポイントに、NFS サーバのディレクトリをマウントします。

OS 起動時に自動マウントするために /etc/fstab を編集し、以下を追加します。

都合上改行していますが、改行は無視して入力してください。

```
Witness-node:/root/qwkshare /mnt/qwkmnt nfs
 soft,timeo=20,retrans=1,noac 0 0
```

マウントします。

```
# mount /mnt/qwkmnt
```

マウントが成功しているか確認します。

```
# mount | grep qwkmnt
```

ノード A の場合、成功していると以下の情報が表示されます。

```
Witness-node:/root/qwkshare on /mnt/qwkmnt type nfs4
(rw,relatime,sync,vers=4.2,rsize=524288,wsize=524288,namlen=255,acreg
min=0,acregmax=0,acdirmin=0,acdirmax=0,soft,noac,proto=tcp,timeo=20,r
etrans=1,sec=sys,clientaddr=10.5.1.11,local_lock=none,addr=10.5.1.20)
```

両方のノードでマウントしたことを確認します。

9.2 QWK storage モードの導入

1. クラスターノードの /etc/default/LifeKeeper の内容を編集します。

クラスター両方のノードで以下の操作を実施します。

以下の内容を変更します。

変更前	変更後
QUORUM_MODE=majority	QUORUM_MODE=storage
WITNESS_MODE=remote_verify	WITNESS_MODE=storage

さらに、以下の内容を追加します。

```
QWK_STORAGE_TYPE=file
QWK_STORAGE_HBEATTIME=6
QWK_STORAGE_NUMHBEATS=4
QWK_STORAGE_OBJECT_Clstr_nodeA=/mnt/qwkmnt/Clstr_nodeA
QWK_STORAGE_OBJECT_Clstr_nodeB=/mnt/qwkmnt/Clstr_nodeB
```

変更していないパラメーター「Quorum_LOSS_ACTION」は Quorum が失われた場合の動作を指定しています。本ドキュメントではデフォルトとなっており、その値は「fastkill」です。Quorum を喪失した場合、そのノードは直ちに停止のアクションを取ります。

※各パラメーターに関しましてはテクニカルドキュメントをご参照ください。
Quorum パラメーター一覧

2. クラスター両方のノードで初期化を行います。
   ノード A で以下のコマンドを入力してください。

```
[root@Clstr-nodeA ~]# qwk_storage_init
```

すると、以下のように表示されます。

```
ok: LifeKeeper is running.
ok: The LifeKeeper license key is successfully installed.
ok: QWK parameter is valid.
            QWK object of /mnt/qwkmnt/Clstr_nodeA is not yet avail.
ok: The path of QWK object is valid.
ok: down: /opt/LifeKeeper/etc/service/qwk-storage: 8159s
ok: Initialization of QWK object of own node is completed.
```

上記が表示された状態でもう一つのノードについても初期化を行います。

```
[root@Clstr-nodeB ~]# qwk_storage_init
```

Successful と表示されれば成功です。
これで QWk の設定が完了しました。

3. 各ノード上の QWK オブジェクトが、指定された間隔で更新されていることを確認します。

/mnt/qwkmnt/Clstr_nodeA の中身を確認します。

```
# cat /mnt/qwkmnt/Clstr_nodeA
signature=lifekeeper_qwk_object
local_node=Clstr-nodeA
time=XXXXXXXXXXXXXX
sequence=52
node=Clstr-nodeB
commstat=UP
checksum= XXXXXXXXXXXXXX
```

/mnt/qwkmnt/Clstr_nodeB も同様の内容になっているか確認します。
なお、両ノードのタイムスタンプは同じになるとは限りません。

9.3 フェイルオーバーシナリオ

実際にサーバに障害を起こし、フェイルオーバーを誘発させます。
本ドキュメントでは稼働系ノード A に通信障害を発生させ、他ノードとの通信を切断します。

この場合、ノード A は以下の動作を行います。

1. 他ノードとの通信が切断されていることを検知します。
2. Witness ノードの両方と通信ができないため、共有ストレージにアクセスすることができません。そのため Quorum チェックに失敗し、Quorum を喪失していることを検知します。
3. QUORUM_LOSS_ACTION を実行します。本ドキュメントでは LifeKeeper とリソースが停止します。

ノード B は以下の動作を行います。

1. ノード A とのコミュニケーションパスが全断していることを検知します。
2. Witness ノードとの通信が可能、かつ共有ストレージにアクセスできるため、自ノードが Quorum を得ることになります。
3. 障害が起こったとされるノード A の情報を Witness ノードにも問い合わせ、ノード A の状態に関して Witness ノードと自ノードで同意見であることを確認し、Witness チェックが成功となります。
4. Witness チェックが成功となったことでリソース起動の許可を得て、フェイルオーバーを開始し、保護対象リソースがノード B 上で起動されます。

では実際にフェイルオーバーを誘発させます。

リソースが全てノード A にあることを確認します。

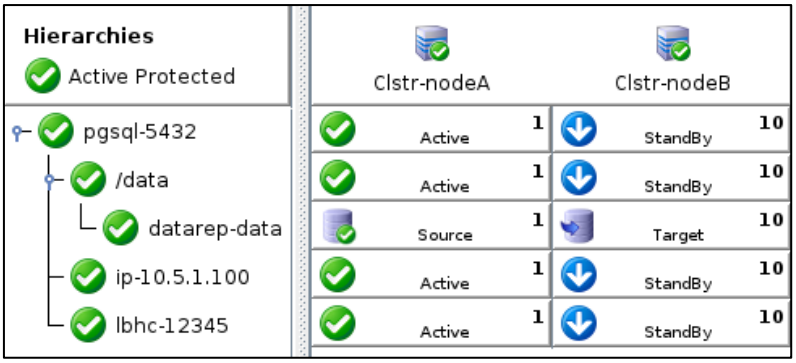

図 9.3-1 障害発生前のリソースの状態

この状態でノード A の NIC を無効に設定し、他ノードとの通信を切断します。

以下のコマンドで一時的に NIC を無効に設定します。

```
# ifconfig eth1 down
# ifconfig eth0 down
```

コマンド実行後、SSH 接続は切断されるので、ノード A の再起動は手動で Azure ポータル上から行う必要があります。

再起動後に NIC は自動的に接続されます。

NIC 無効後はノード B の GUI は以下のように、表示されます。

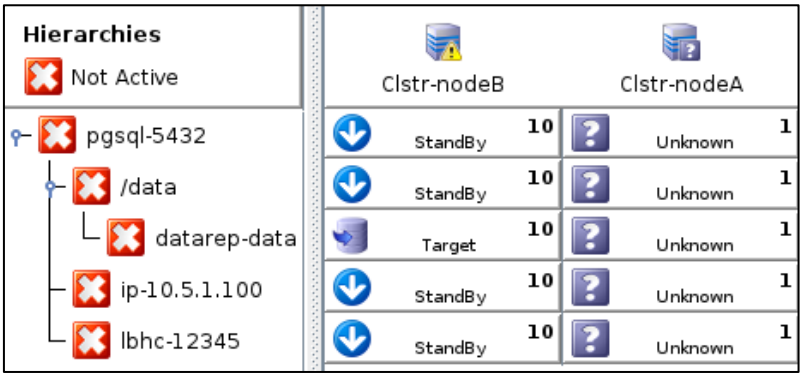

図 9.3-2 NIC 無効後のリソースの状態

一定時間後、フェイルオーバーが開始され、ノード B で自動的にリソースが起動されます。

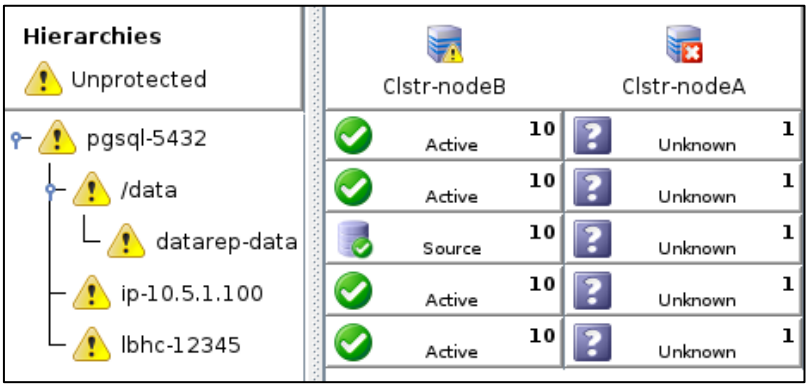

図 9.3-3 フェイルオーバーが発生したリソースの状態

上記でフェイルオーバーのテストは終了です。

Quorum / Witness Kit で想定される動作につきましては以下のドキュメントも参照ください。

Quorum Witness Server Support Package for Linux 利用ガイド

9.4 留意事項

Quorum / Witness サーバで障害が発生した場合、コミュニケーションパスの全断が起きるとすべてのサーバでリソースが停止します。

そのため、Witness サーバに障害が発生した場合は直ちに正常の状態へ戻すためのアクションを行う必要があります。

10. お問い合わせ

本書の記載内容についてのお問い合わせ先

■ LifeKeeper 製品の導入を検討中のお客様

弊社パートナー営業部までお問い合わせください。

お問い合わせメールフォーム

https://mk.sios.jp/BC_Web_Free-entry_Inquiry.html

■ LifeKeeper 製品をご購入済みのお客様

お問い合わせの一次窓口が弊社ではない場合があります。

お問い合わせの際はサポート証書よりサポート窓口をご確認ください。

サポート窓口が弊社の場合は、下記の Web サイトよりお問い合わせください。

弊社のサポート窓口

https://bccs.sios.jp/contact/

11. 免責事項

- 本書に記載された情報は予告なしに変更、削除される場合があります。最新のものをご確認ください。
- 本書に記載された情報は、全て慎重に作成され、記載されていますが、本書をもって、その妥当性や正確性についていかなる種類の保証もするものではありません。
- 本書に含まれた誤りに起因して、本書の利用者に生じた損害については、サイオステクノロジー株式会社は一切の責任を負うものではありません。
- 第三者による本書の記載事項の変更、削除、ホームページ及び本書等に対する不正なアクセス、その他第三者の行ためにより本書の利用者に生じた一切の損害について、サイオステクノロジー株式会社は一切の責任を負うものではありません。
- システム障害などの原因によりメールフォームからのお問い合せが届かず、または延着する場合がありますので、あらかじめご了承ください。お問い合せの不着及び延着に関し、サイオステクノロジー株式会社は一切の責任を負うものではありません。

サイオステクノロジー株式会社
住所 ： 〒106-0047
東京都港区南麻布 2 丁目 12-3　サイオスビル
URL：https://sios.jp