

Hinemos ミッションクリティカル機能紹介

NTTデータ先端技術株式会社

NTT DATA
Trusted Global Innovator



INDEX

1. Hinemosミッションクリティカル機能の特徴
2. アーキテクチャ詳細解説
3. ユースケース
4. appendix

1

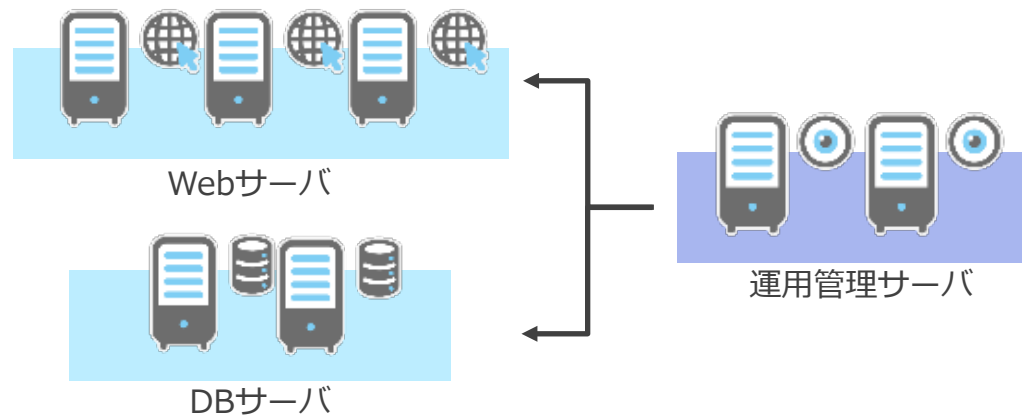
Hinemosミッションクリティカル機能の特徴

運用管理機能の多重化の必要性

監視やジョブ管理といった運用管理機能についても、様々な場面で高可用性が必要に

ミッションクリティカルシステム

システムの機能全体を多重化



業務機能を提供するサーバだけでなく
監視やジョブ管理を司る運用管理サーバも
多重化が必須

インフラのオープン化・クラウド化

オンプレミス



・ハードウェア障害



仮想化



・ハードウェア障害
・ハイパーバイザ障害



クラウド



・ハードウェア障害
・データセンタ障害

運用管理サーバの障害は、運用管理機能の停止に繋がる

オープン化に伴い単機能単一サーバの
構成が主流となり、サーバ数の増大に伴い
サーバ単位の障害は頻繁に発生

Hinemosミッションクリティカル機能とは

統合運用管理機能を提供するHinemosマネージャの多重化をワンパッケージで提供！
収集・監視・自動化の高可用性を実現

Hinemosミッションクリティカル機能の特徴

簡易構築・低コスト

- ① クラスタリングソフト・共有ディスク不要
- ② マルチプラットフォーム対応

統合運用管理機能の可用性

- ③ 障害検知と自動系切替による運用継続
- ④ 系切替中のメッセージロスト防止機構

簡易運用・オンライン復旧

- ⑤ オンライン系切戻しによる障害復旧
- ⑥ 1コマンドによる系切戻し

①クラスタリングソフト・共有ディスク不要

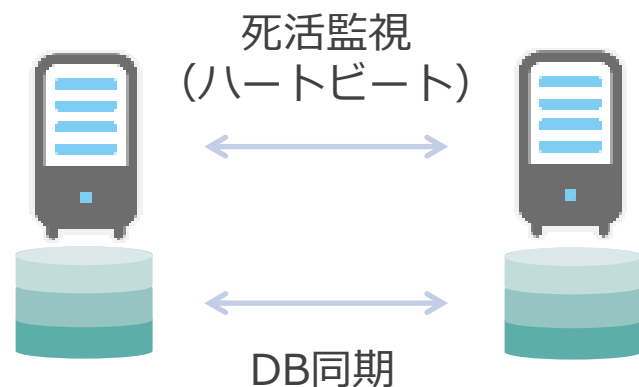
簡易構築・低コスト

通常のOS2台だけで、Hinemosマネージャを多重化！

Hinemosマネージャ

マスタサーバ

スタンバイサーバ



Hinemos自身がソフトウェアで
死活監視とDB同期を実施

クラスタリングソフトの
追加費用が不要

共有ディスクの
追加費用が不要

設計・構築時の
SE・CEコストを削減

障害発生時にも
ワンストップサポート

基盤要件がシンプル
マシンを2台用意するだけ

マルチプラットフォーム
対応

ソフト・ハードウェア費用とエンジニア工数を一気に削減

オンプレミス・仮想化・クラウド環境を全て同一アーキテクチャで対応

オンプレミス



IAサーバを2台用意するだけ

仮想化



ハイパーバイザが異なる
仮想マシンを2台用意するだけ

クラウド



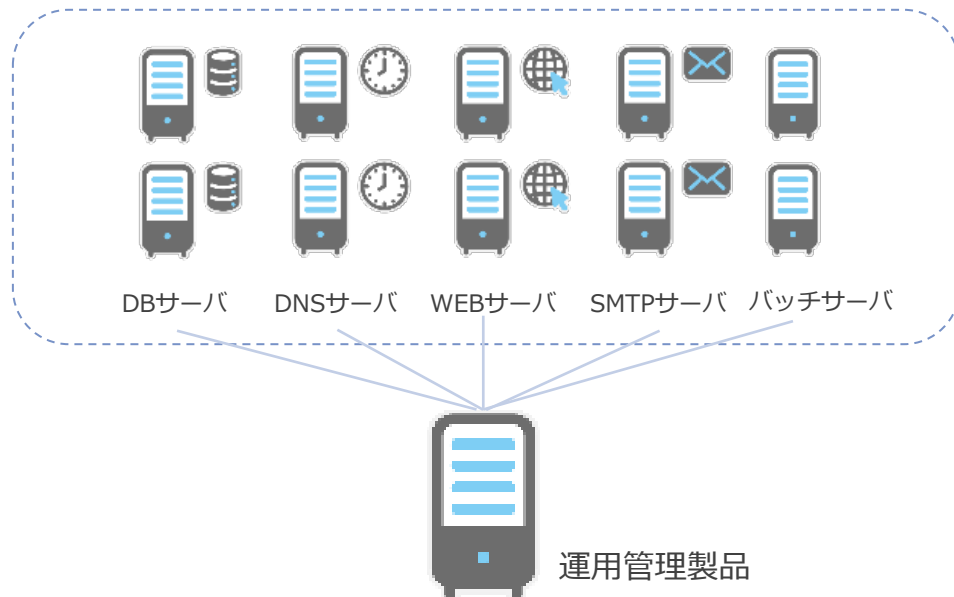
インスタンスを2台用意するだけ
VPC Peer/DirectConnectにも対応

環境が変わっても、Hinemos自身がソフトウェアで死活監視とDB同期を実施するだけ

プラットフォームが変わっても、同じスキルセットで導入可能

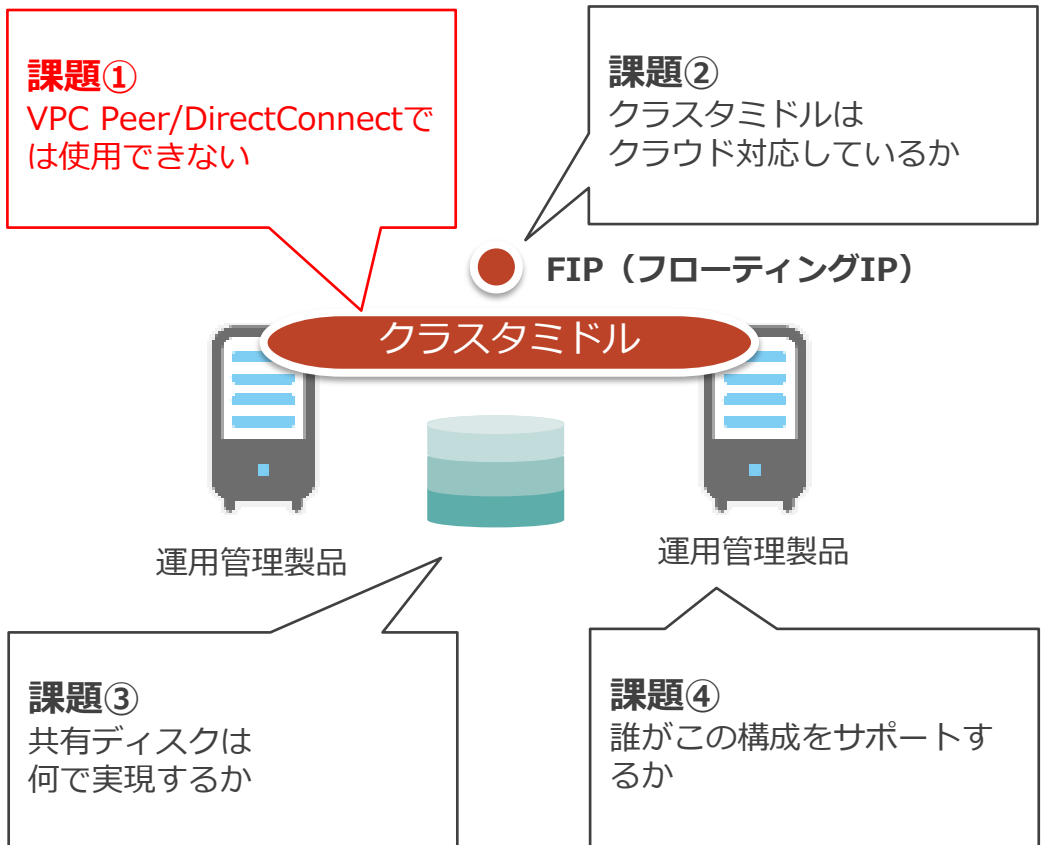
運用管理製品を通常のクラスタ構成で実現すると、クラウドでは様々な課題が出てくる

運用管理製品のNWの特徴



Web/AP/DBサーバの様に特定のサーバ間ではなく
全サーバ間との通信が必要となる
(異なるNWセグメント、異なるVPC間等)

クラウド上のクラスタ構成の課題



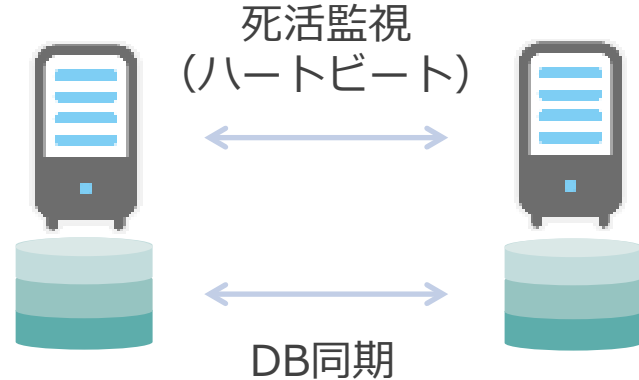
他にもAzureだと、この方式自体を推奨しないといったケースもある

クラウドに影響されないアーキテクチャでクラスタを実現

Hinemosマネージャ

マスタサーバ

スタンバイサーバ



Hinemos自身がソフトウェアで
死活監視とDB同期を実施

VPC Peer/DirectConnect環境でも利用可能
(SIPだけで接続可能)

クラスタリングソフトは不要
(Hinemosが機能として保持)

共有ディスクは不要
(Hinemos自体でDB同期)

構成全体をHinemos保守サポートだけで解決
(ソフトウェアで実現するクラスタ構成)

AWSやAzureなど様々なクラウド上で、可用性構成を簡単に実現

③障害検知と自動系切替による運用継続

環境に合った様々な障害を検知し、自動切替を実現

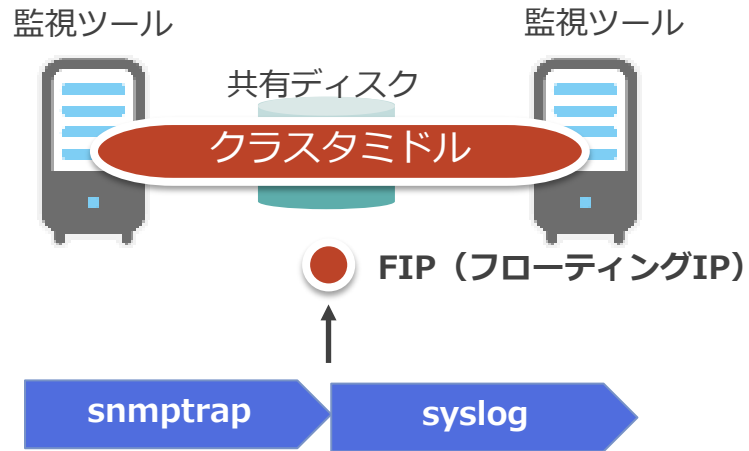


素早い自動切替により、監視やジョブ制御を止めることなく運用継続

④系切替中のメッセージロスト防止機構

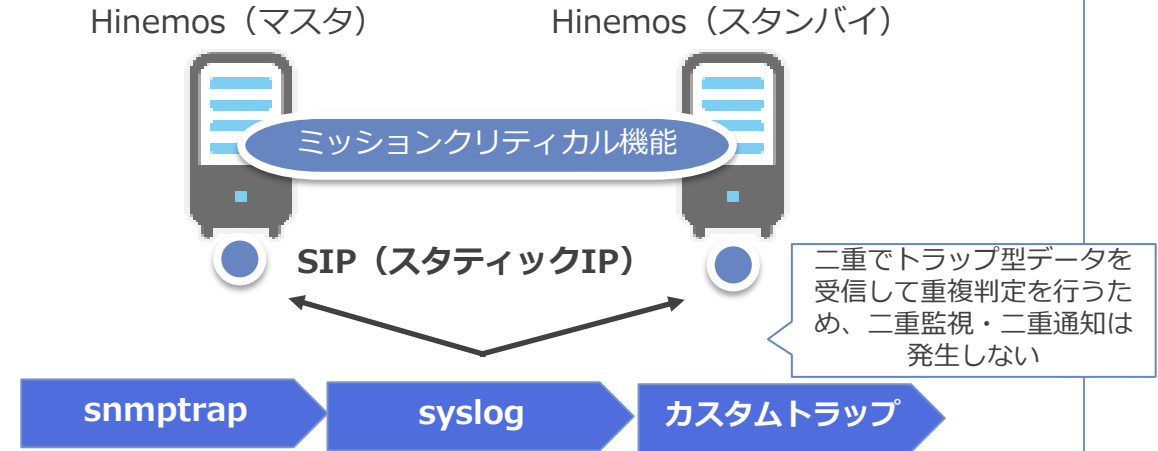
両系でトラップ型データを受信するトラップ型監視のロスト防止機構

一般的な監視ツールのクラスタ構成



FIPでトラップ型データを受信するため
系切替中のトラップ型データは受信できない

Hinemosミッションクリティカル構成



マスタ・スタンバイの両サーバの
SIPでトラップ型データを受信をして
系切替中の受信担保をする

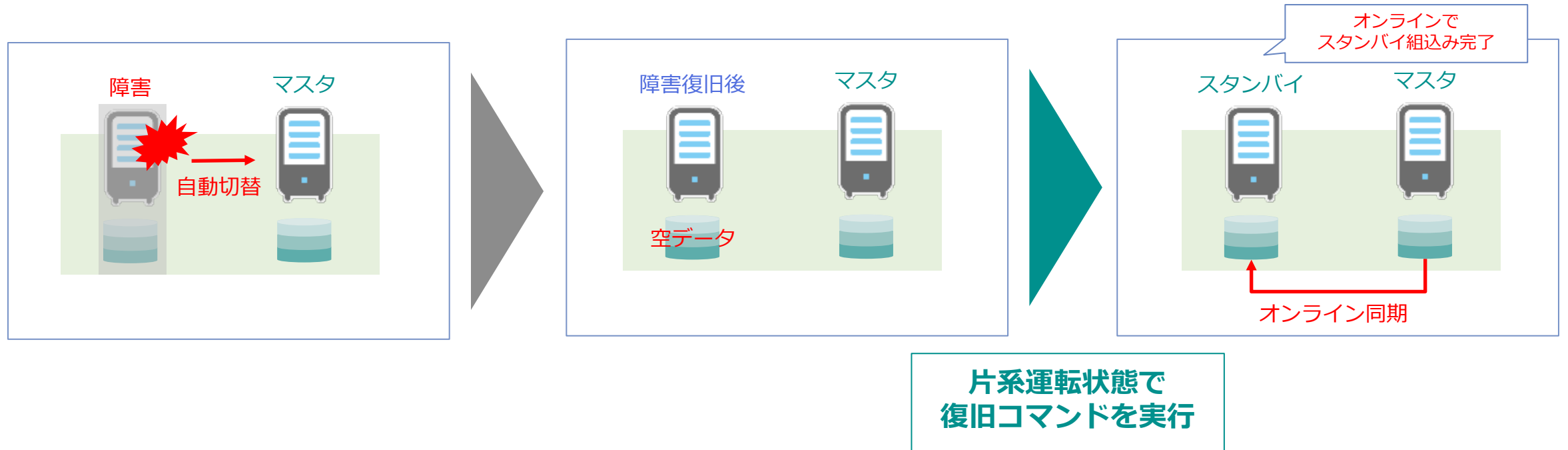
Hinemosマネージャの系切替中でも、重要なトラップをロストせずに監視

⑤ オンライン系切戻しによる障害復旧

簡易運用・オンライン復旧

片系運転からの両系運転への切戻しも、オンラインで実現

障害時の自動切替が出来ても、両系運転に切戻す際に停止が発生しては意味がない



障害発生から障害復旧まで、全てをシステムに影響なく無停止で対処！

⑥ 1コマンドによる系切戻し

複数の障害パターンに対して、提供する復旧コマンドは1つのみ

一般的な運用管理製品のクラスタ構成



- ・ 障害ポイントが多い
- ・ 対処する操作は障害ポイントによる
- ・ 複数のエンジニアが必要



パターンごとの復旧手順書が必要

Hinemosミッションクリティカル構成



- ・ Hinemos視点で障害切り分け可
- ・ 復旧コマンドは1つだけ
- ・ Hinemosエンジニアのみで十分



復旧手順書のパターンは不要

シンプルな手順書かつオペレータでも実施できる簡易操作で復旧が可能

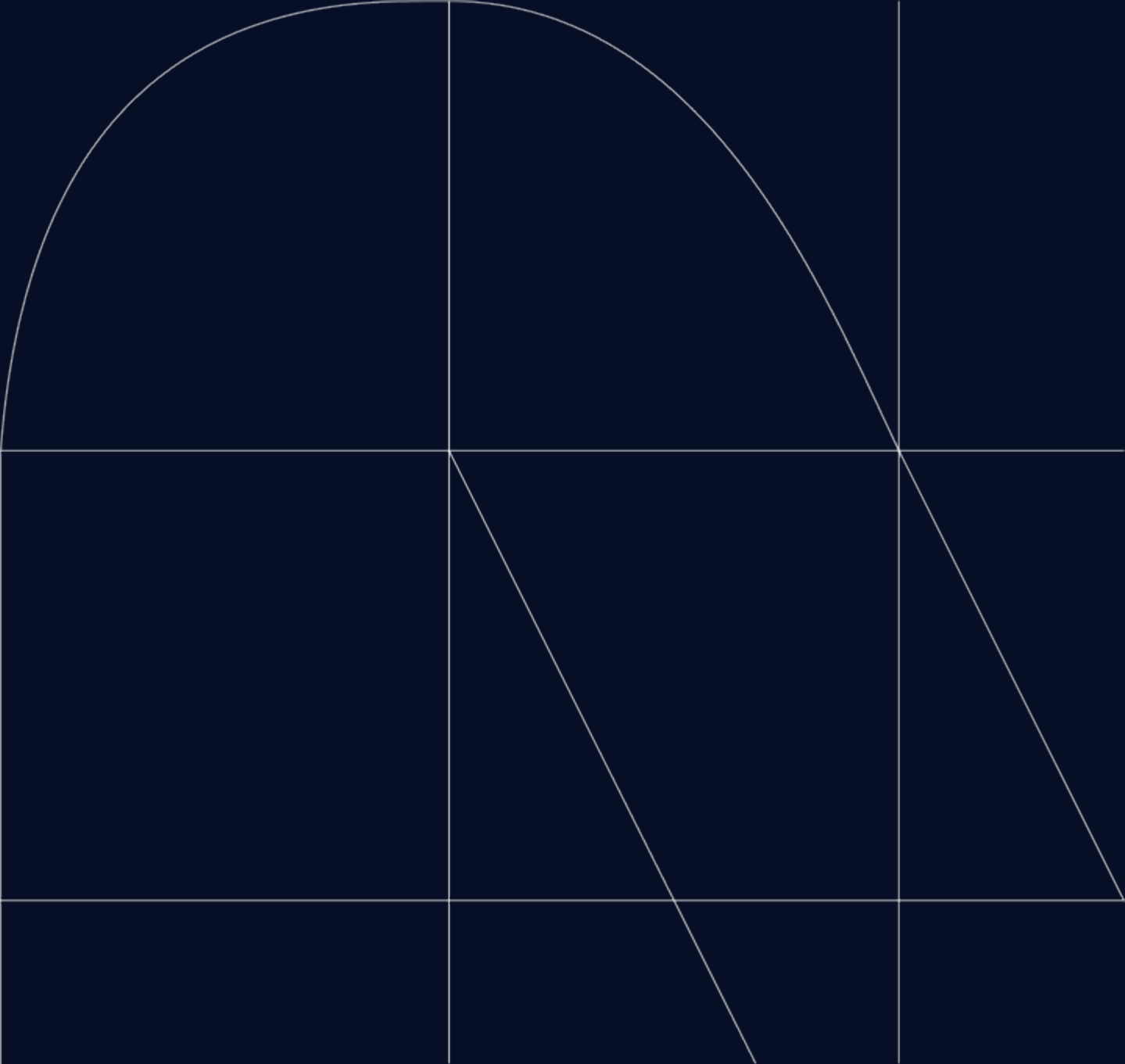
クラスタ方式からみたHinemosの優位性

	Hinemos		一般的なクラスタ方式の製品	
費用	◎	Hinemosのソフトウェア費用のみ	△	クラスタミドル、共有ディスク、構築費用など
切替時間	○	素早い切替（1～2分）	△	数分～数十分かかるものも
監視	◎	トラップ型監視のロスト防止機構	×	トラップ型監視はロスト
ジョブ	○	ジョブ制御の自動継続	○	ジョブ制御の自動継続
復旧	◎	1つの復旧コマンド	×	障害パターンごとの復旧方法
クラウド	◎	プラットフォーム非依存で全機能を高可用化	×	クラウドにより実現可否・制約がある
保守性	◎	ワンストップ保守サポート	×	障害ポイントごとに異なる保守サポート

設計・導入・運用そして費用の全てにおいて優位性がある

2

アーキテクチャ詳細解説

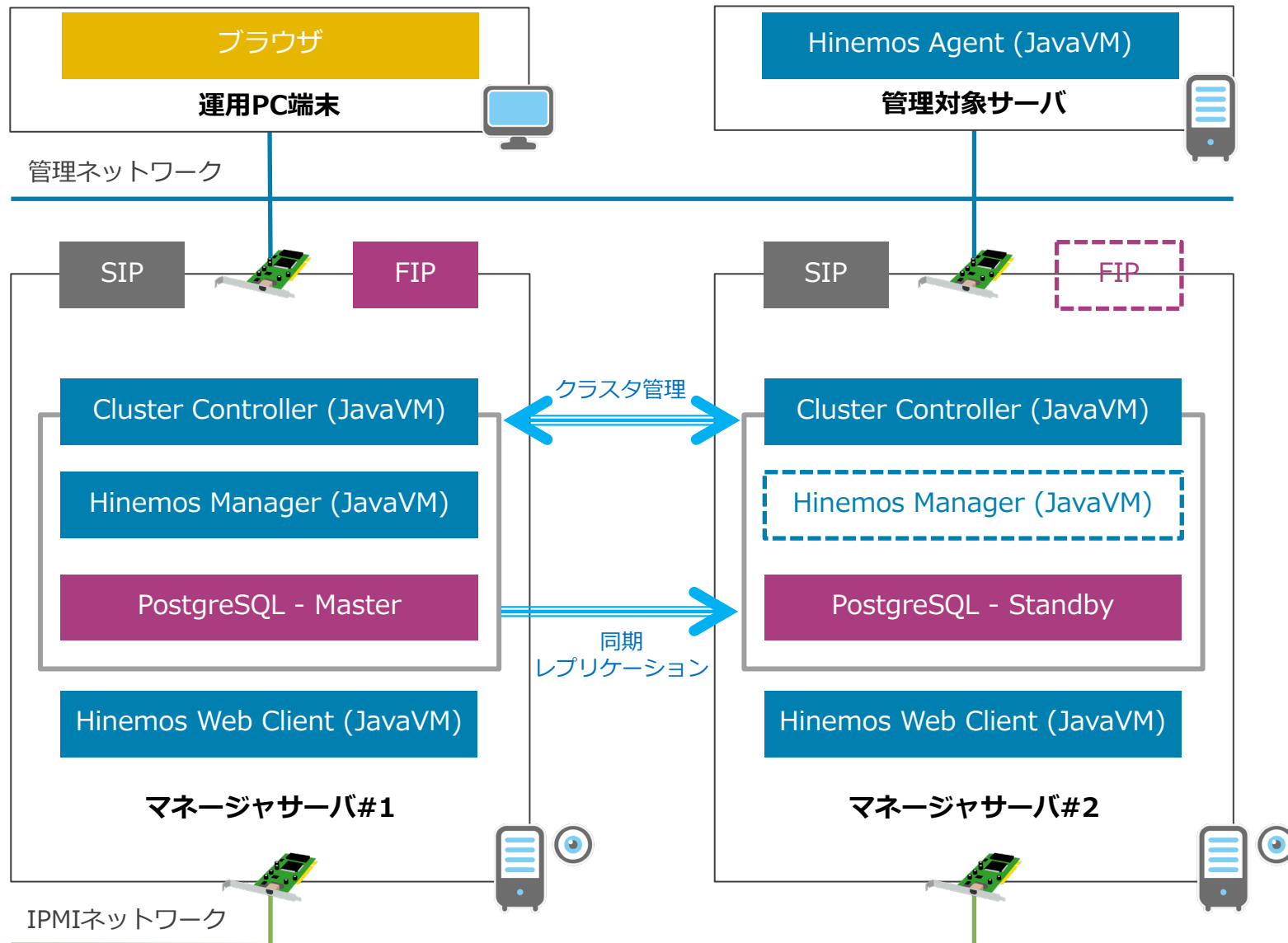


アーキテクチャ詳細解説

Linux版,Windows版マネージャ共通 アーキテクチャの主要なポイント

- ・アーキテクチャ概要
- ・クラスタの仕組み
- ・データ同期の仕組み
- ・メッセージロスト防止機構
- ・エージェント接続方式

アーキテクチャ概要 ソフトウェアで実現するクラスタ構成



アーキテクチャのポイント

Hinemosマネージャの構成

Hinemos ManagerのJavaプロセス
PostgreSQLのプロセス

Cluster Controllerの機能

ハートビートなどのクラスタ管理
FIPの管理

Hinemos内部DBの同期

PostgreSQLの同期レプリケーション

Hinemos Client/Hinemos Agent

SIP/FIPに対応した接続機構

簡易セットアップ

インストーラに従った操作で簡単構築

クラスタの仕組み

Cluster Controllerがクラスタの基本機能を実現し、系切替が必要な障害を検知

Cluster Controllerの機能

HA構成管理 赤字のヘルスチェックで障害検知し系切替

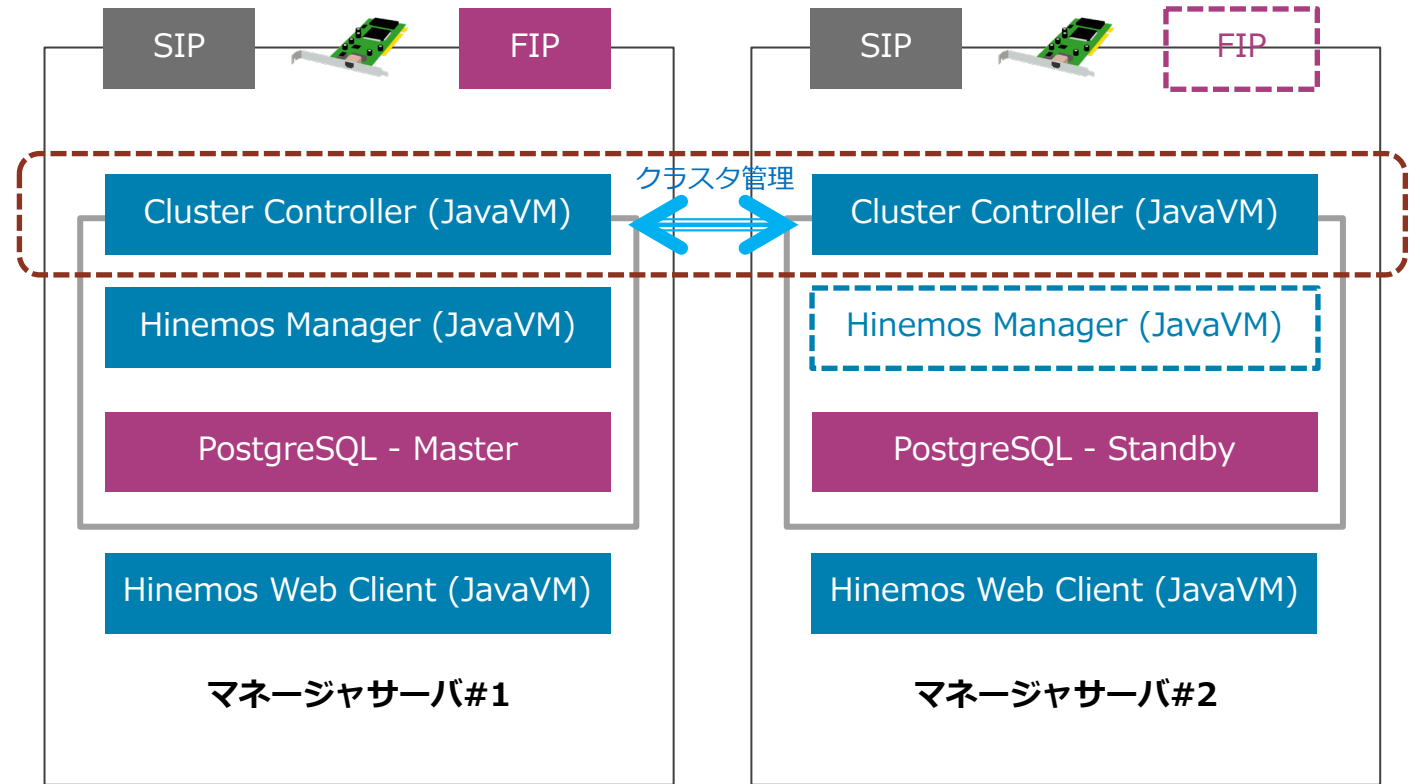
- MasterサーバおよびStandbyサーバの決定
- ハートビート通信によるヘルスチェック
- ステートフルセッションによるヘルスチェック

マネージャ上のリソース管理

- プロセス制御とネットワーク設定
- 外部ネットワークからのサーバ孤立検知
- ネットワークインタフェースのヘルスチェック
- フローティングIPアドレスのヘルスチェック
- ファイルシステムのヘルスチェック
- PostgreSQLのヘルスチェック
- Hinemos Manager (JavaVM)のヘルスチェック

トラップ(受信)系監視の可用性向上

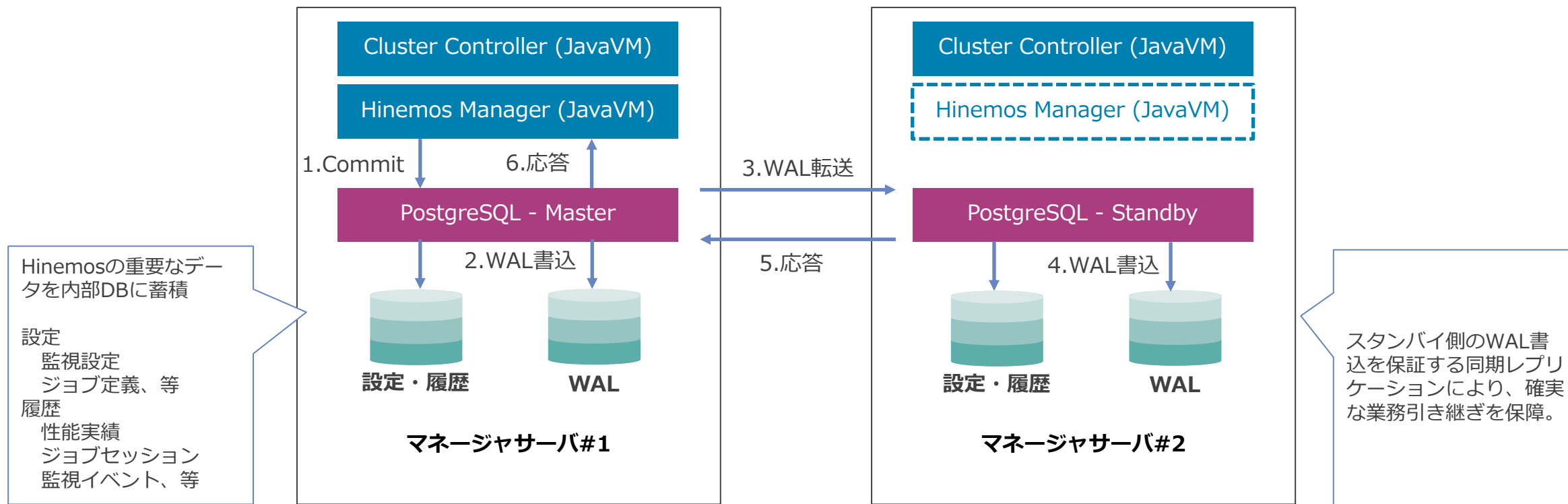
- 系切替中のトラップのロスト防止
(syslog, snmptrap, カスタムトラップ)



クラスタミドルは不要！ Hinemosだけで高可用性を実現

データ同期の仕組み

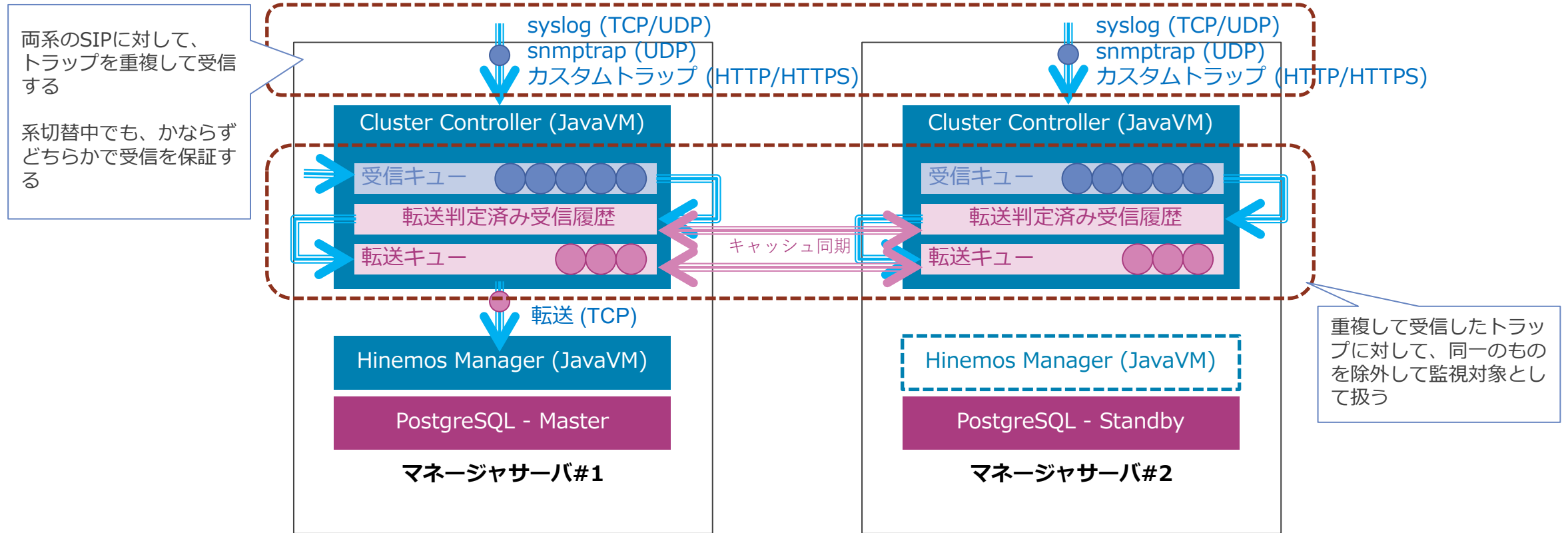
Hinemosの設定・履歴情報を蓄積する内部DBを、同期レプリケーションによりデータ同期



共有ディスクは不要！ ソフトウェアでデータ同期を実現

メッセージロスト防止機構

両系でトラップ型データを受信して、系切替中のデータロストを防止しながら重複通知も抑制

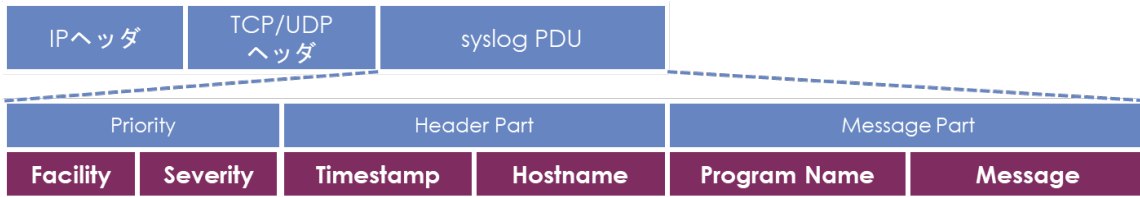


通常のクラスタ構成では実現の難しい系切替中のデータロスト防止を実現

メッセージロスト防止機構 メッセージ同一性判定

各トラップのメッセージの内容をベースに同一性判定を行う

◆syslogの場合



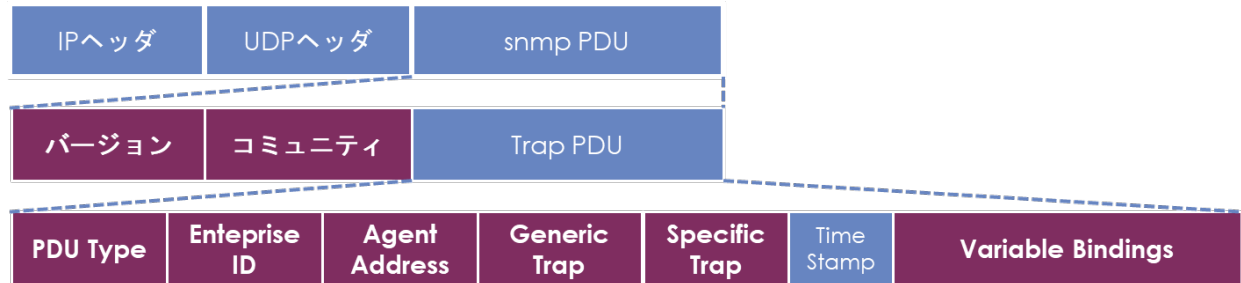
◆カスタムトラップの場合

```
{  
  "FacilityID": "送信元のファシリティID",  
  "DATA": [  
    {  
      "DATE": "yyyy-MM-dd HH:mm:ss",  
      "TYPE": "NUM" or "STRING",  
      "KEY": "キーパターン",  
      "MSG": "任意の数値"  
    }  
  ]  
}
```

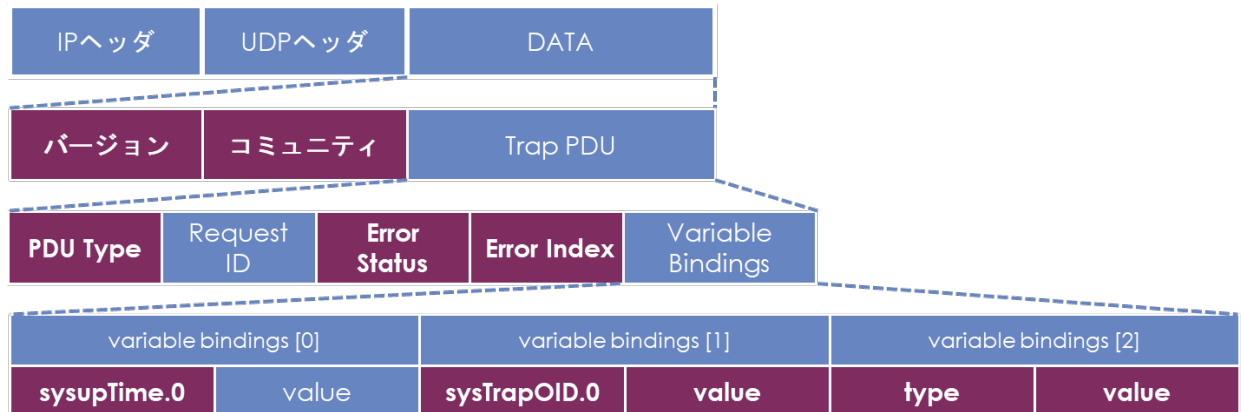
～トラップの同一性判定の条件～

- ・トラップの送信元が同一
- ・メッセージ（紫枠）が同一
- ・直近の期間(デフォルト30秒)で転送されていない

◆snmptrap (v2) の場合



◆snmptrap (v3) の場合



エージェント接続方式

シンプルな接続方式にてHinemosマネージャのFIP/SIPの両方に対応

[FIPへアクセスする場合の設定]

managerAddress=http://192.168.1.241:8081/HinemosWS/

[SIPへアクセスする場合の設定]

managerAddress=http://192.168.1.101:8081/HinemosWS/,http://192.168.1.102:8081/HinemosWS/

Agent.properties

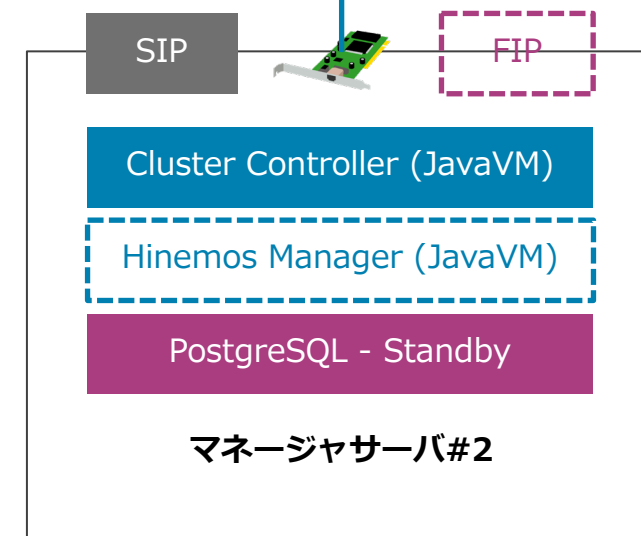
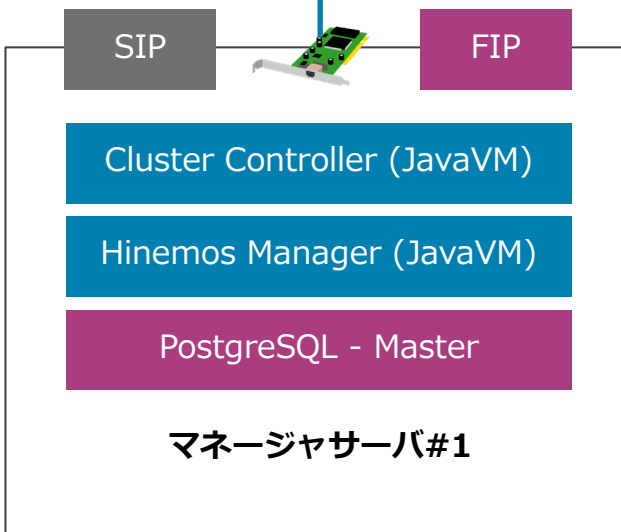
Hinemos Agent (JavaVM)

管理対象サーバ

管理ネットワーク

エージェント接続先の自動切替

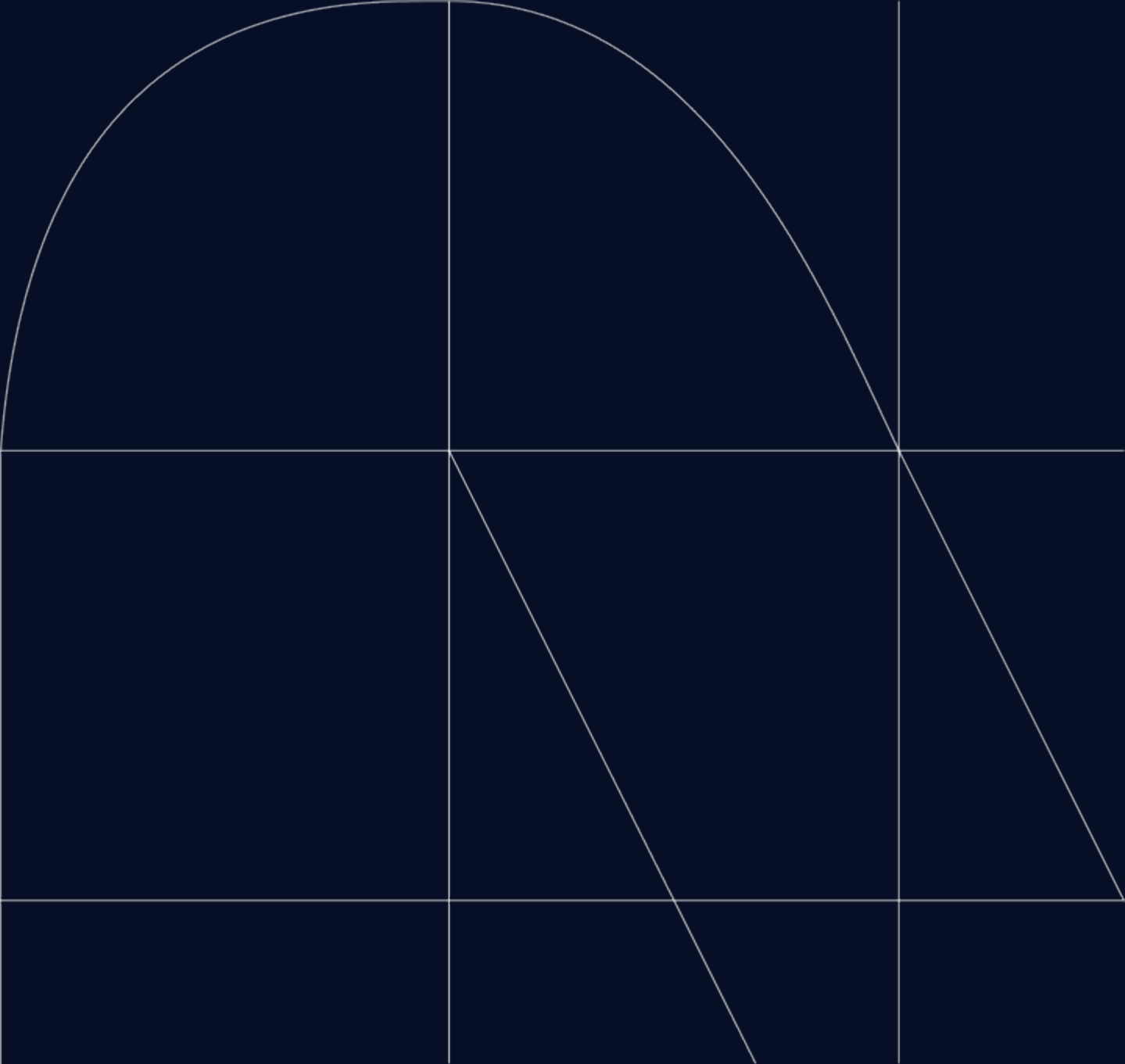
- managerAddressにHinemosマネージャの接続リスト（カンマ区切り）で指定
- 先頭から順に試行して、接続できたマネージャと通信再開（ラウンドロビン）
- 重要なデータはHinemosマネージャ側保存のため接続先を切替のみで処理継続



FIPが使用できないクラウド環境下でも、可用性構成を簡単に実現

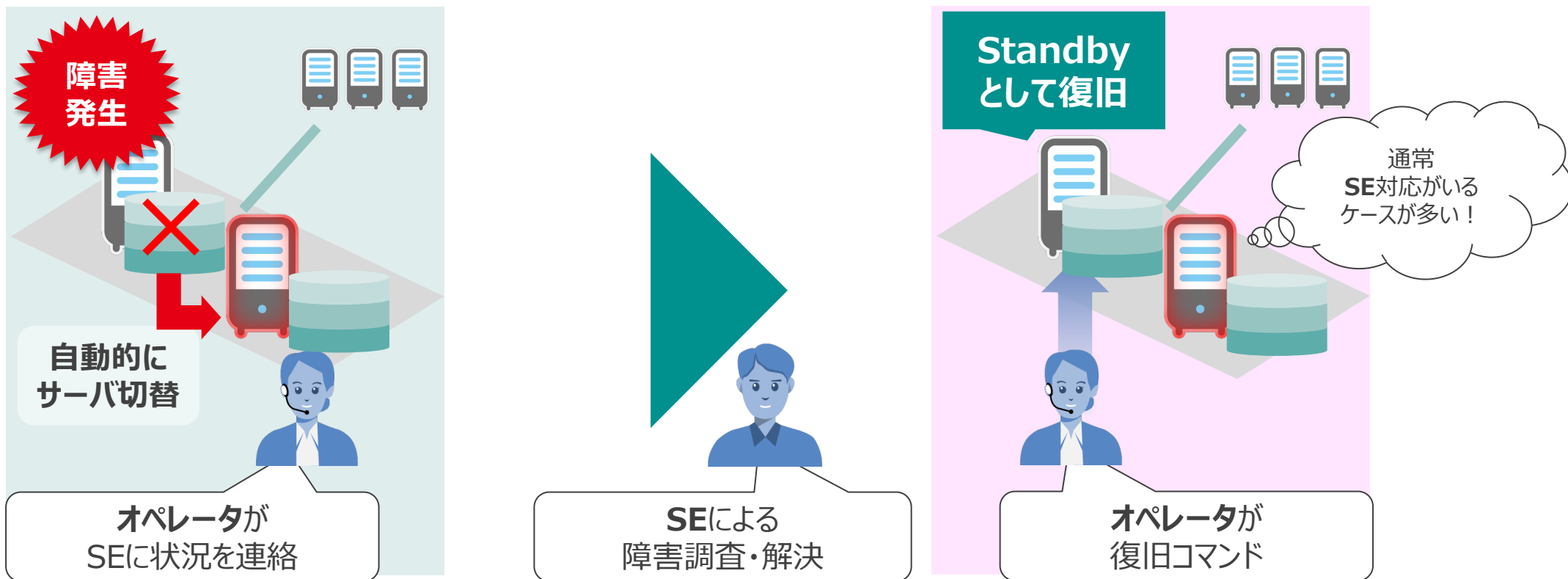
3

ユースケース



ユースケース

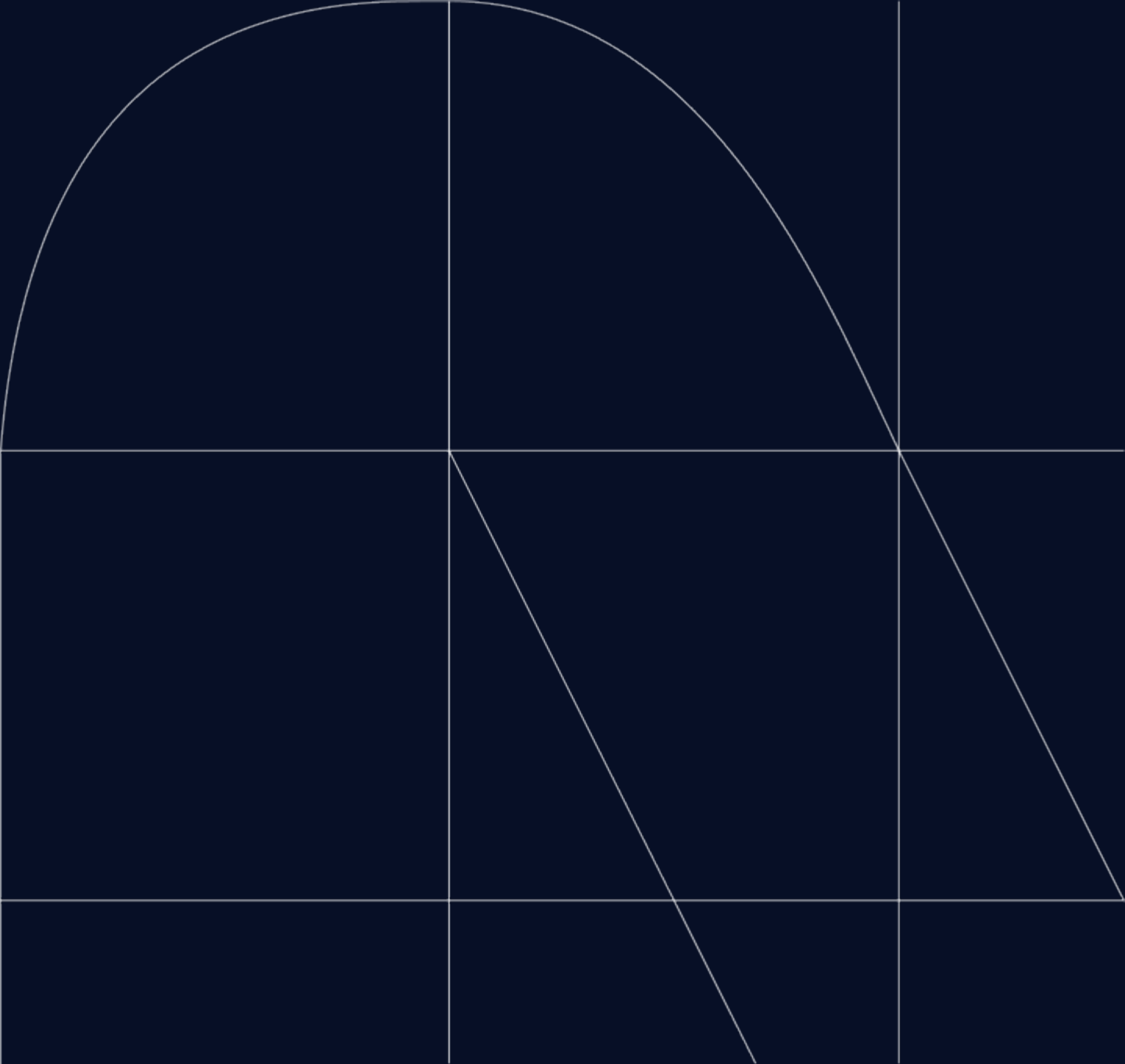
運用管理サーバのディスク障害をイメージとした例
SEは、ディスク障害の調査・解決までを対応すればOK



オペレータでHinemosに関するほとんどの作業のカバーが可能

4

appendix



Hinemosではじめる

実践ジョブ管理・自動化入門 技術評論社

2023.3.9 発売開始

最新バージョン ver.7.0に対応

Amazonから購入可能



お問い合わせはこちら

お待ちしているもに！

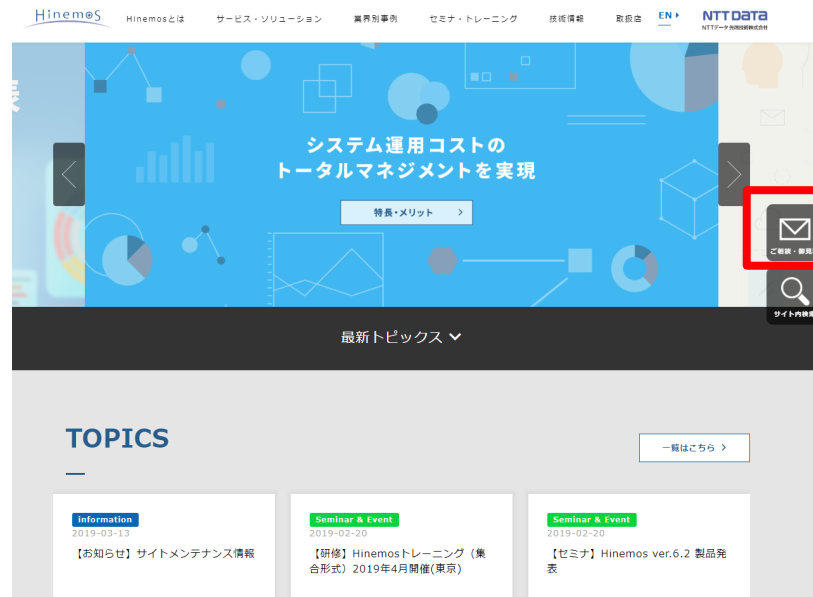


Hinemosに関するお問合せ

お気軽にお問合せください。

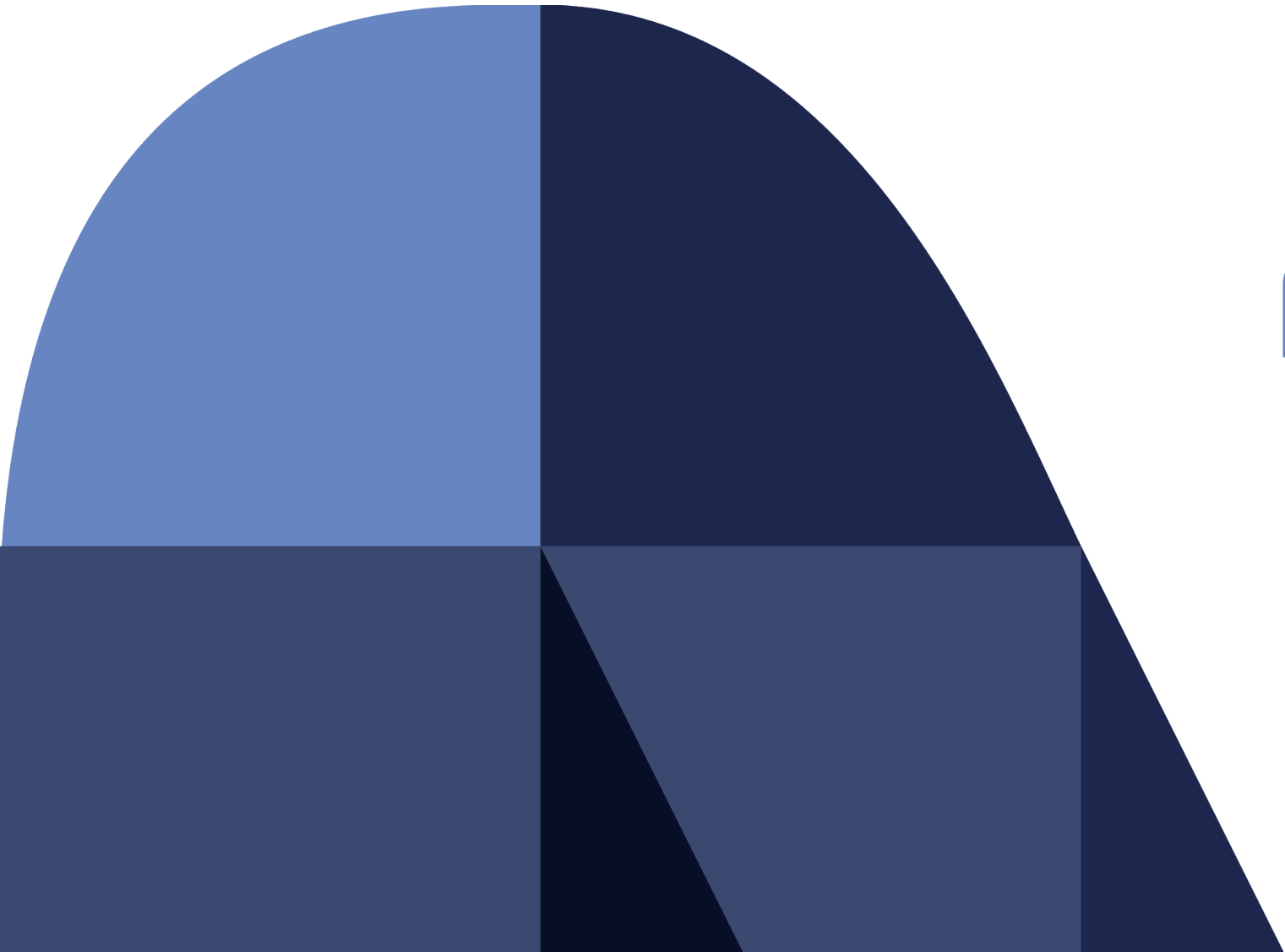
Hinemosポータルサイト

URL : <https://www.hinemos.info/contact>

ご相談フォーム





NTT DATA
Trusted Global Innovator