JANOG54 Meeting in NARA

機器監視のためのZabbixを、DB含めてL3冗長化した話

株式会社 Jストリーム

ネットワークインフラ部 ネットワークインフラ課

小山 拓海

◯ 今井 宏謙

2024-07-04

もっと素敵な伝え方を。

自己紹介

株式会社Jストリーム　ネットワークインフラ部 ネットワークインフラ課

今井 宏謙

- 2020年度入社、新卒5年目
- CDNの基盤ネットワーク・サーバインフラの構築・運用
- JANOG参加は10回目ぐらい
  - 初参加はJANOG39@金沢に若者支援で参加しました
- JANOG54 BAKUCHIKU参加してます
  - 会場Wi-Fi、問題なく提供出来てるといいな

採用した構成

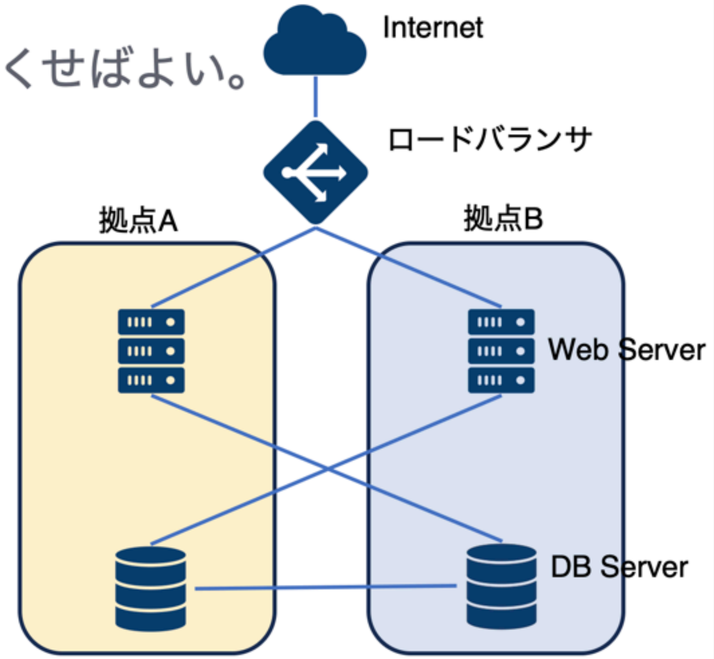
もっと素敵な伝え方を。 Jstream

- この構成について説明します

パターン5 Active-Standby (Zabbix-HA)

もっと素敵な伝え方を。 Jstream

- パターン3を拡張
- VIPの問題があるのであれば、VIPをなくせばよい。
- 実現方法を考えてみた。



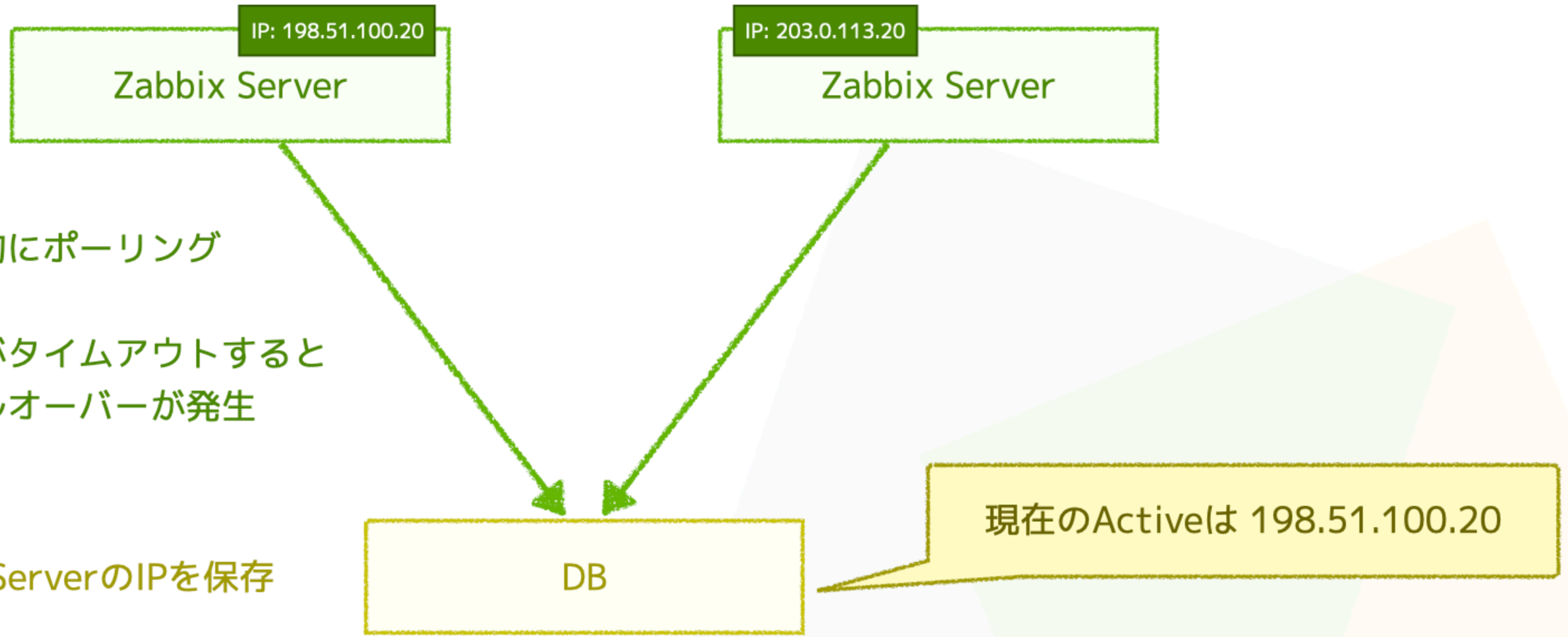
17

Zabbix HAの仕組み

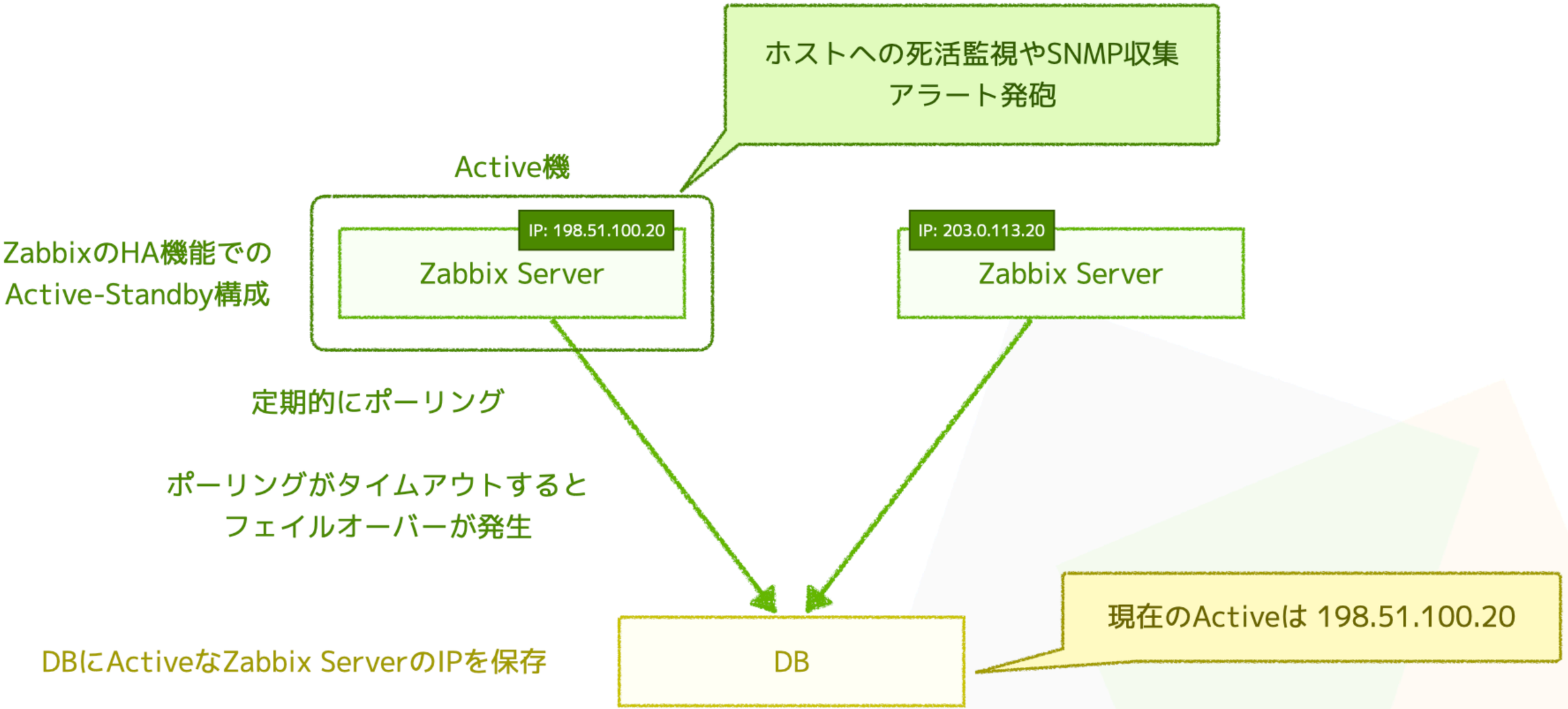
ZabbixのHA機能での
Active-Standby構成

Zabbix HAの仕組み

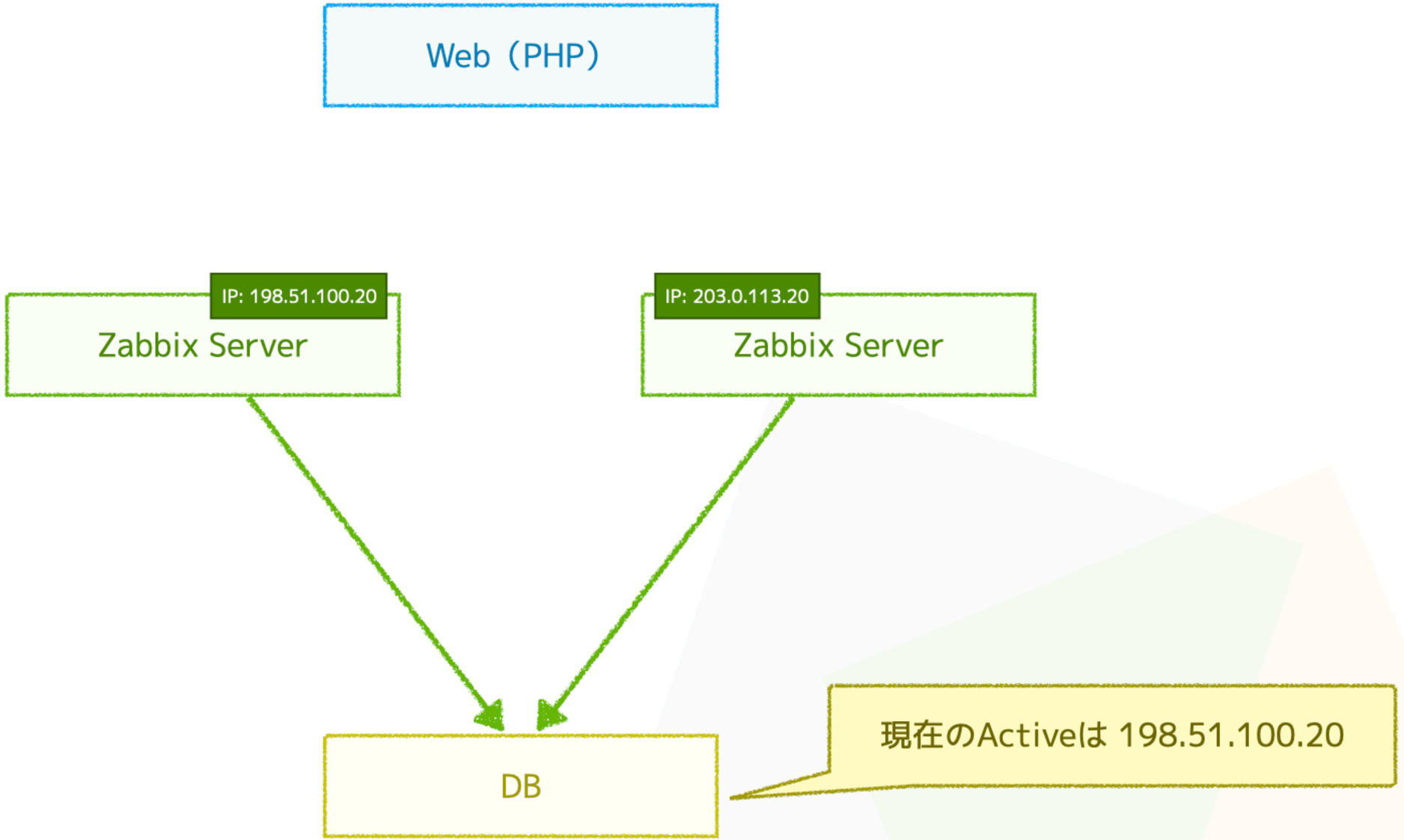
ZabbixのHA機能での
Active-Standby構成



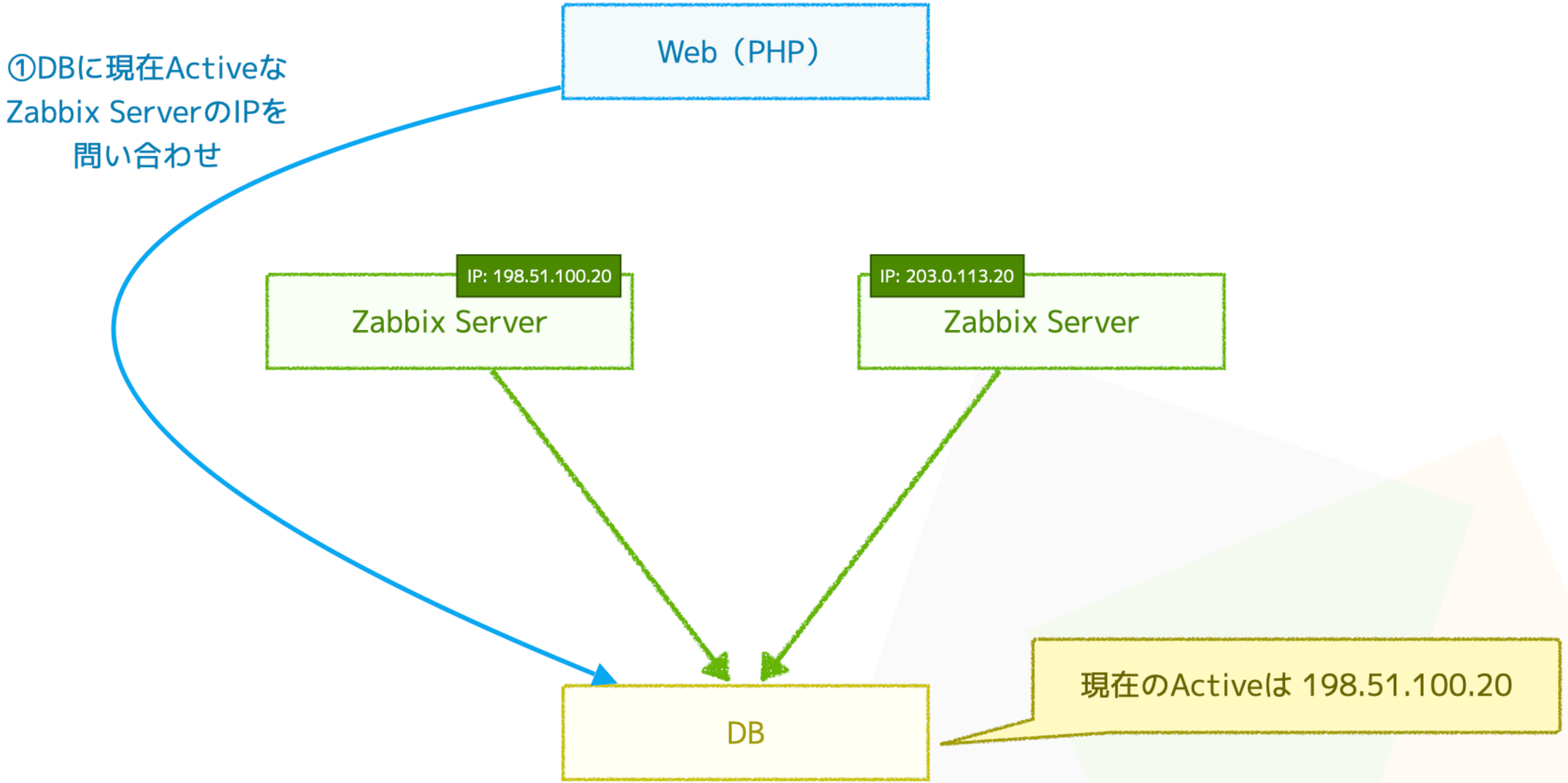
Zabbix HAの仕組み



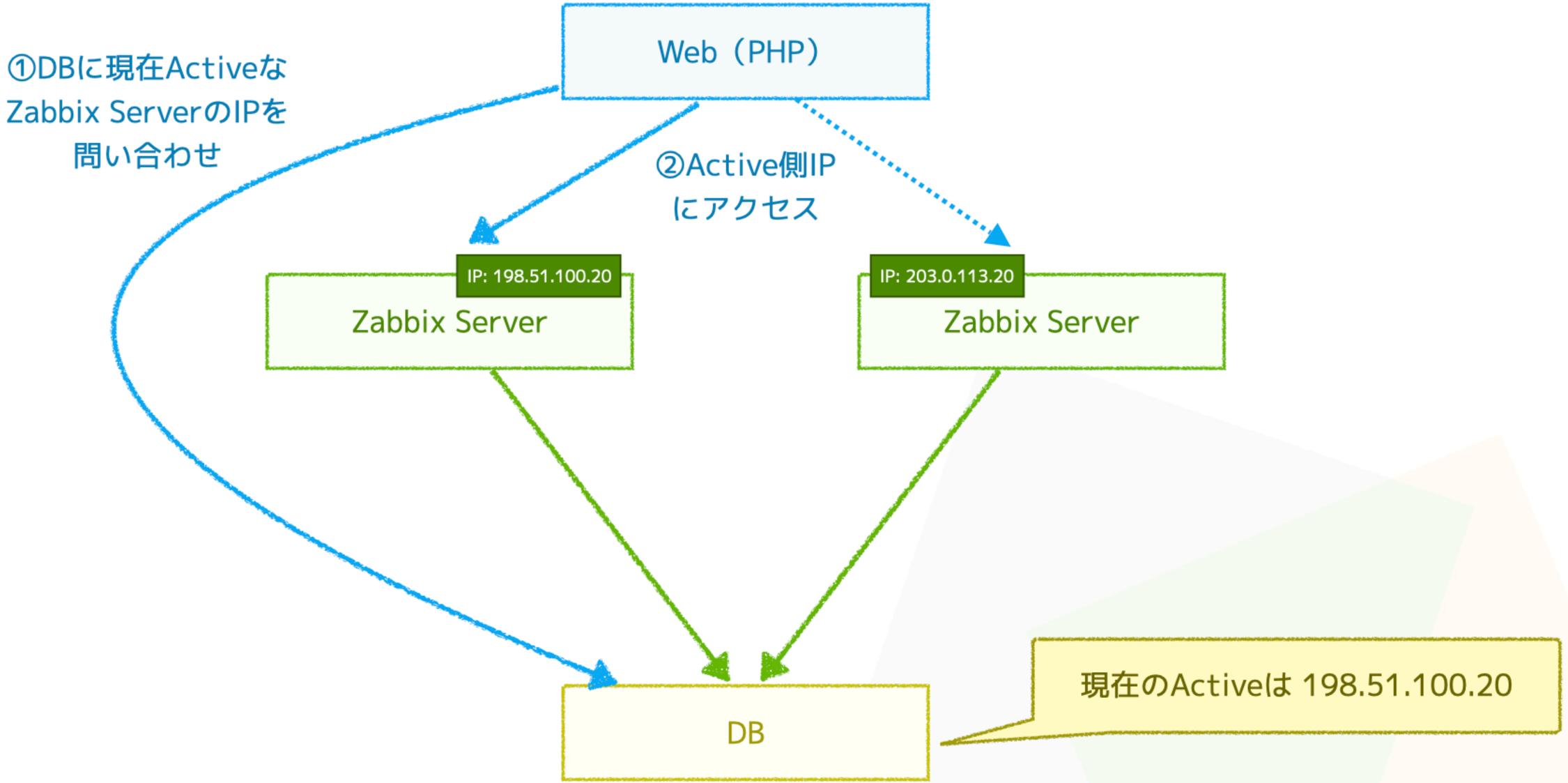
Zabbix HAの仕組み



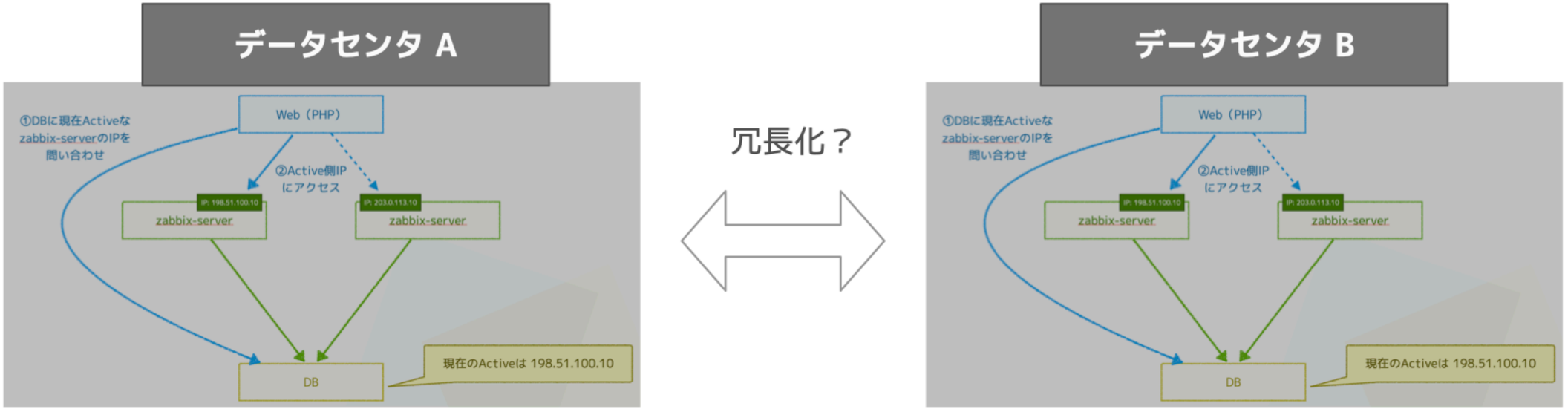
Zabbix HAの仕組み



Zabbix HAの仕組み

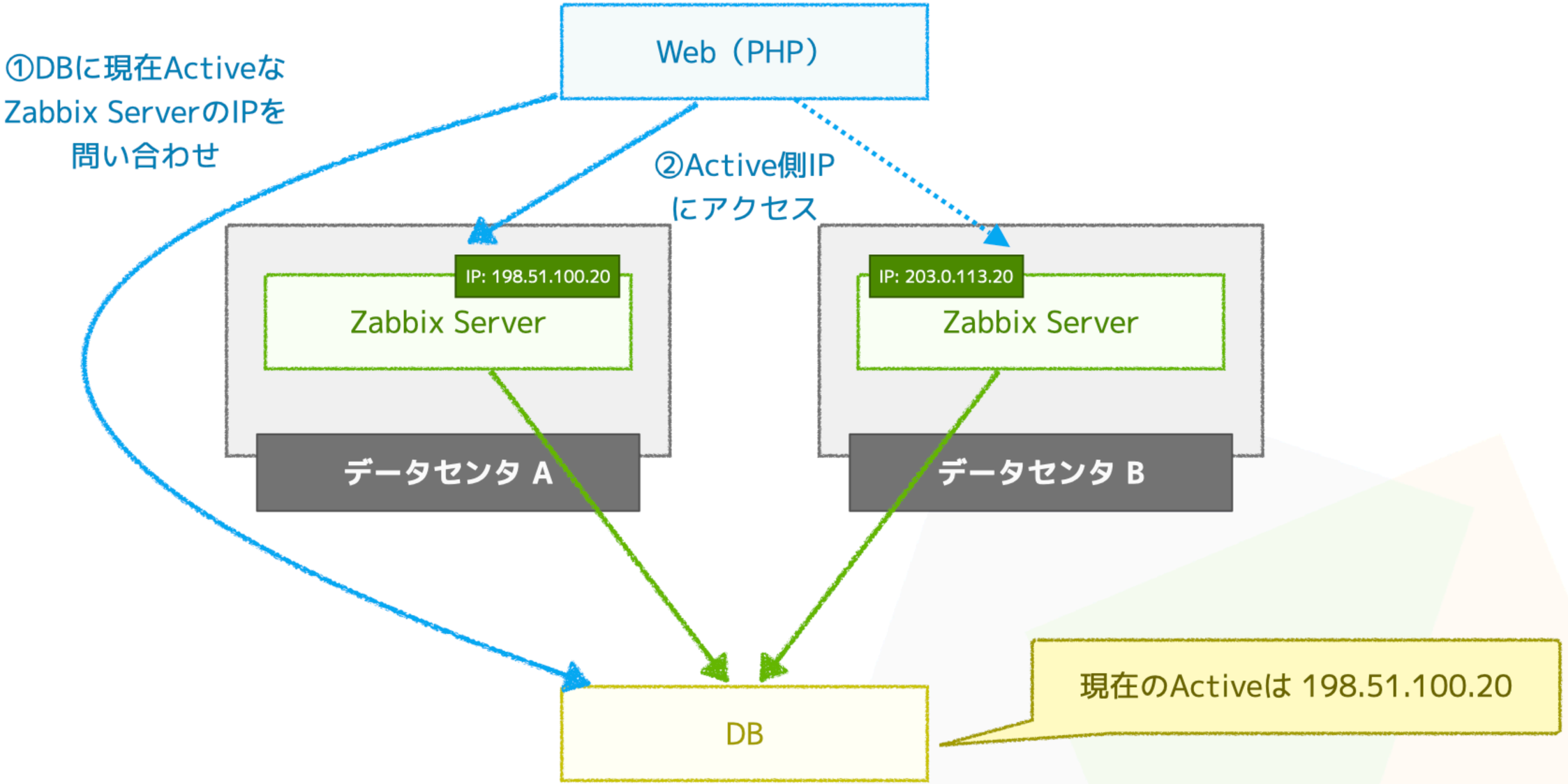


Zabbix HAのデータセンタ冗長



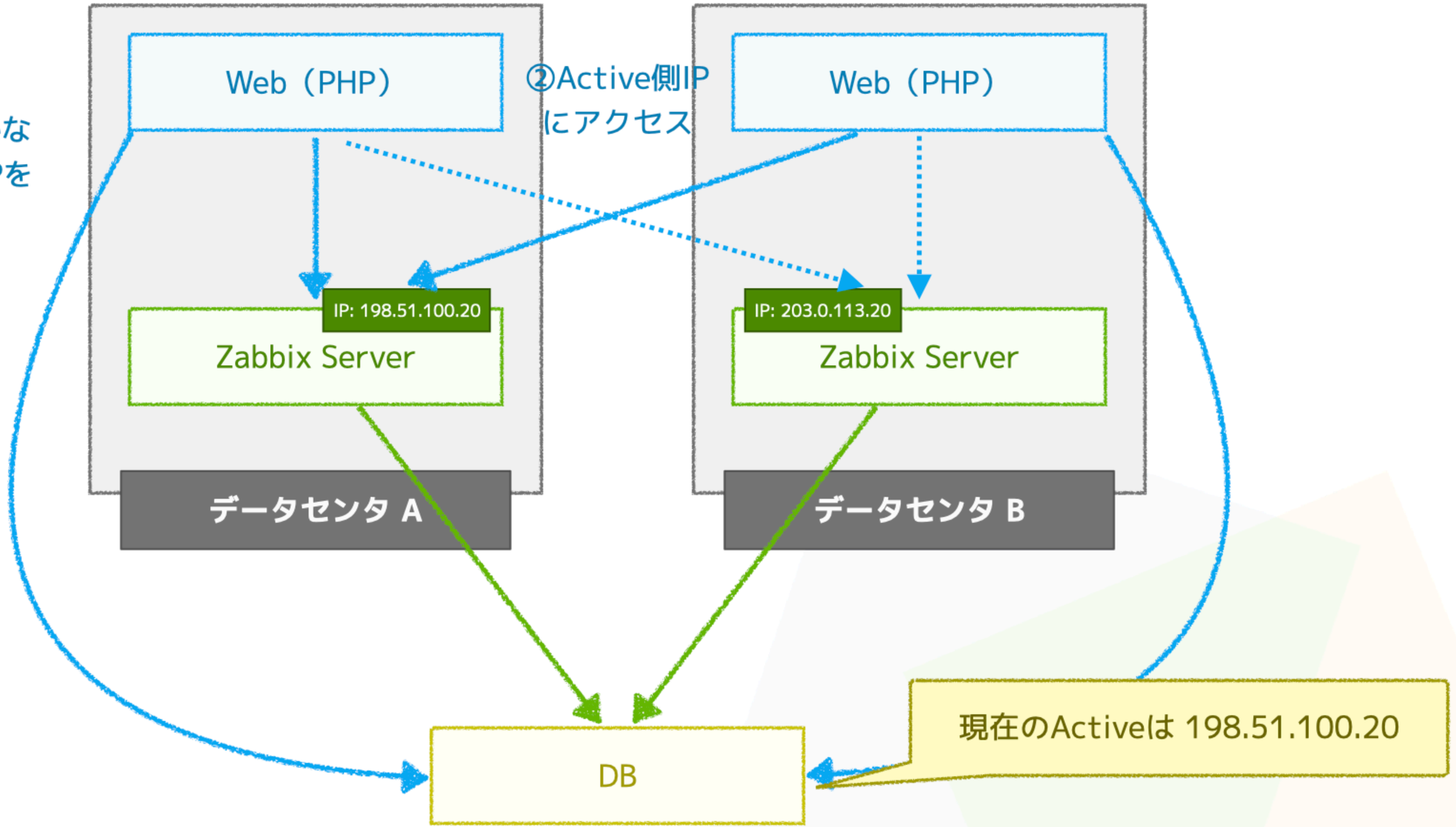
- 拠点を跨いだ冗長を行うには、一工夫必要そう
- Zabbixのパートナー企業様の資料によるとHAの方法はあるが、
  L2ネットワーク内で考えられているものが多く、拠点をまたぐのは難しそう
  - DC冗長を構築するドキュメントは乏しい、、、
  - DB周りはZabbix公式でドキュメントが見当たらなかった

Zabbix Serverの冗長化



Webの冗長化

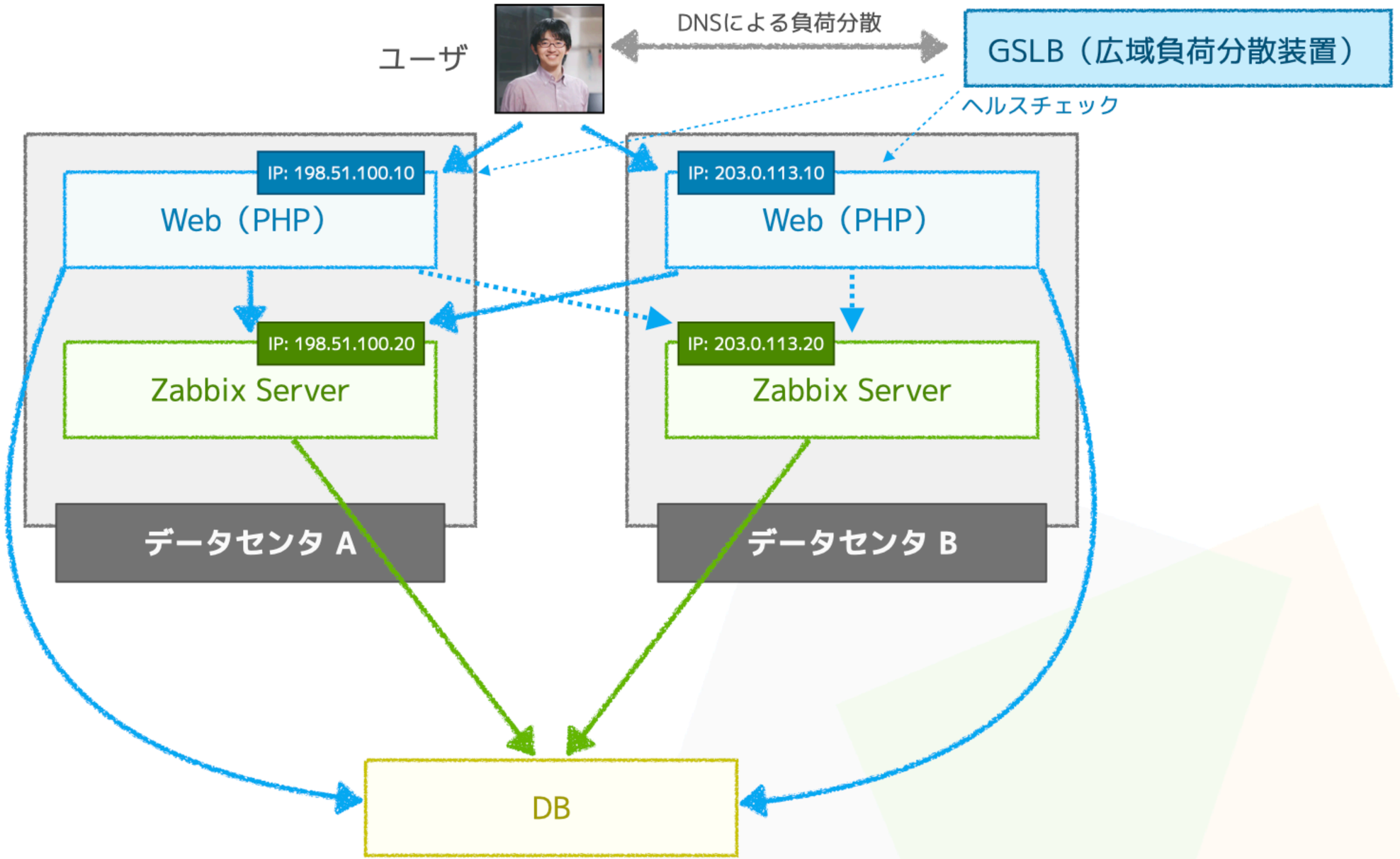
①DBに現在Activeな
Zabbix ServerのIPを
問い合わせ



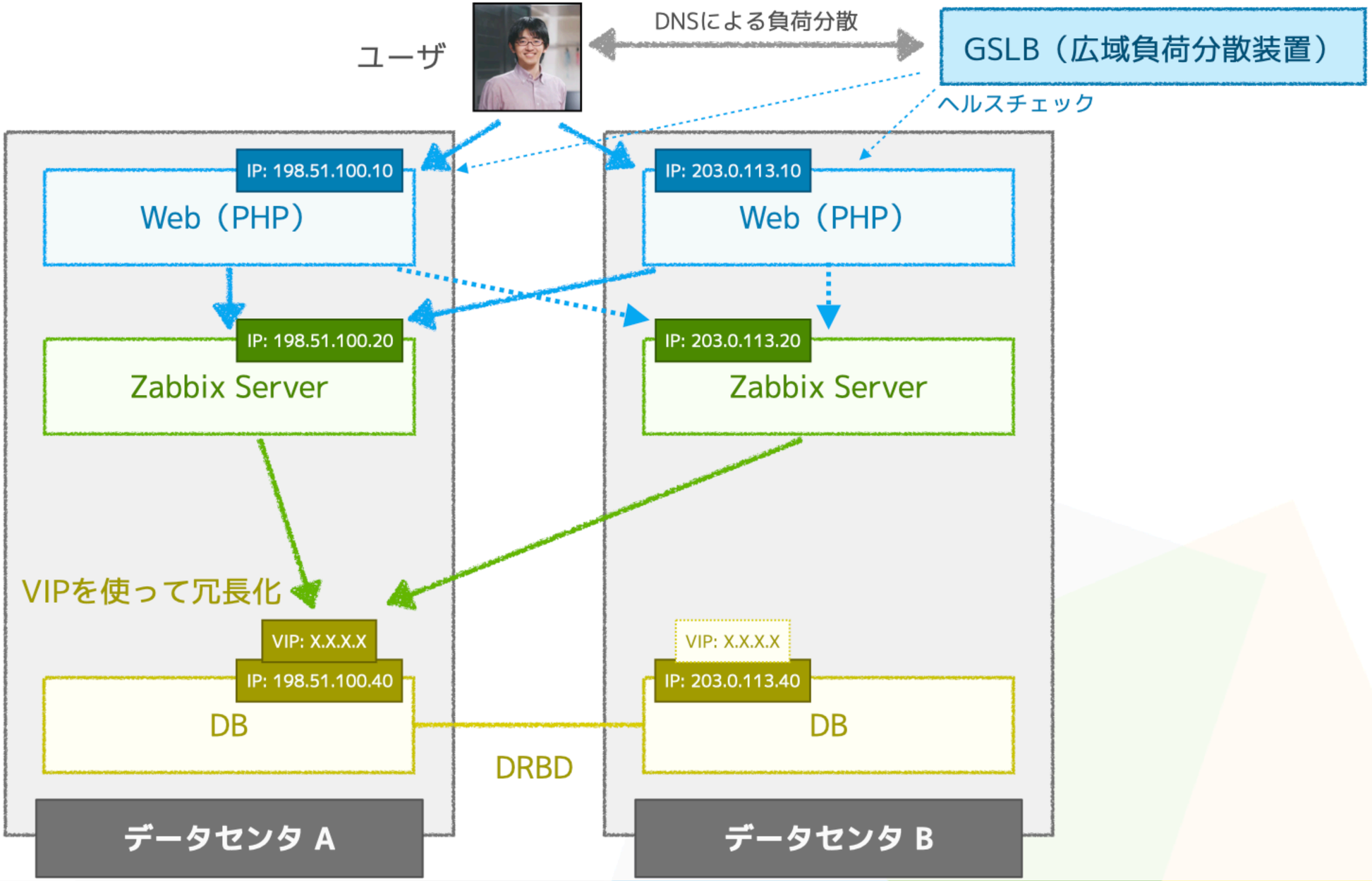
Webの冗長化

もっと素敵な伝え方を。 Jstream

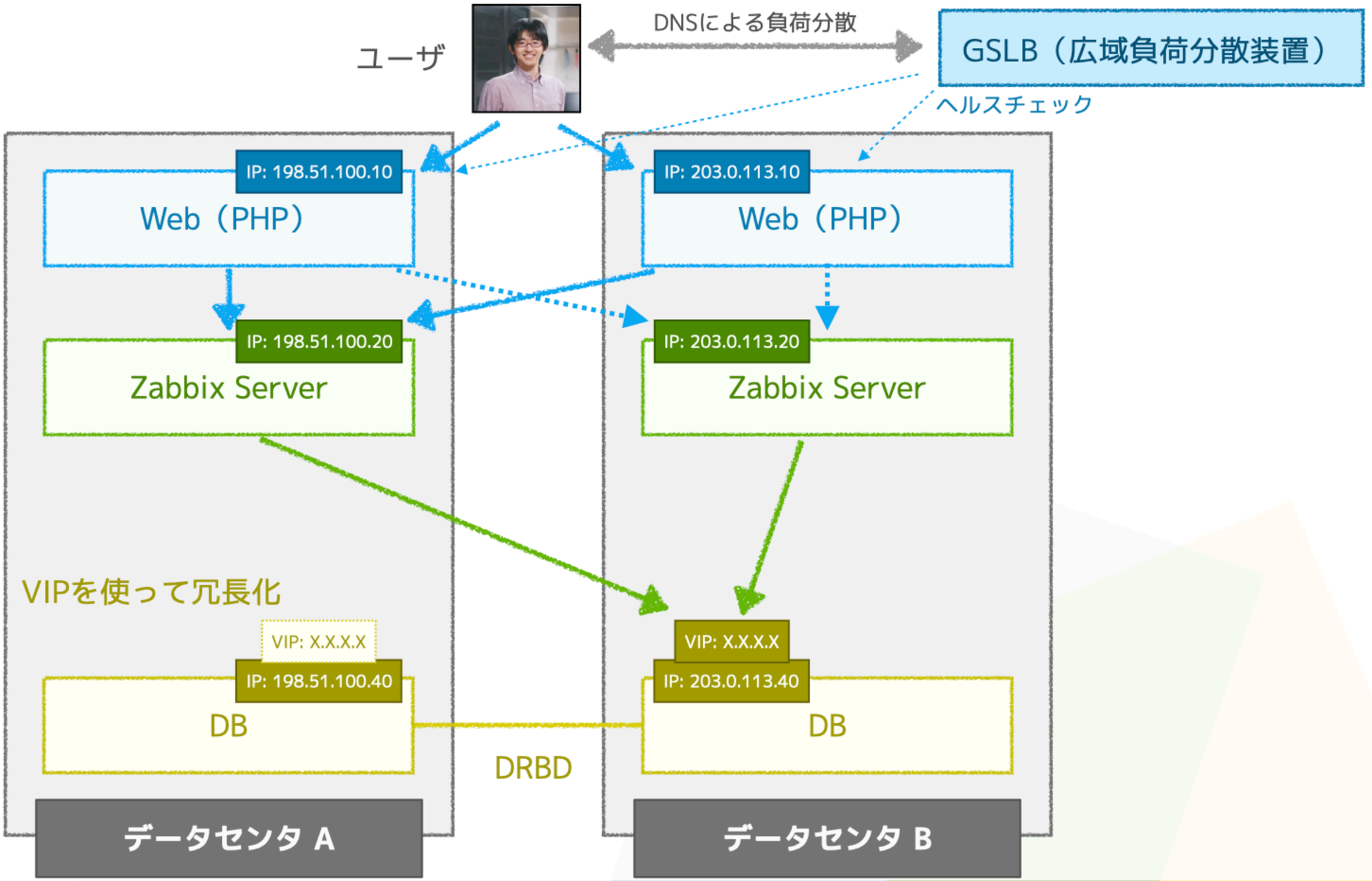
GLSBを用いて
Active-Activeの分散



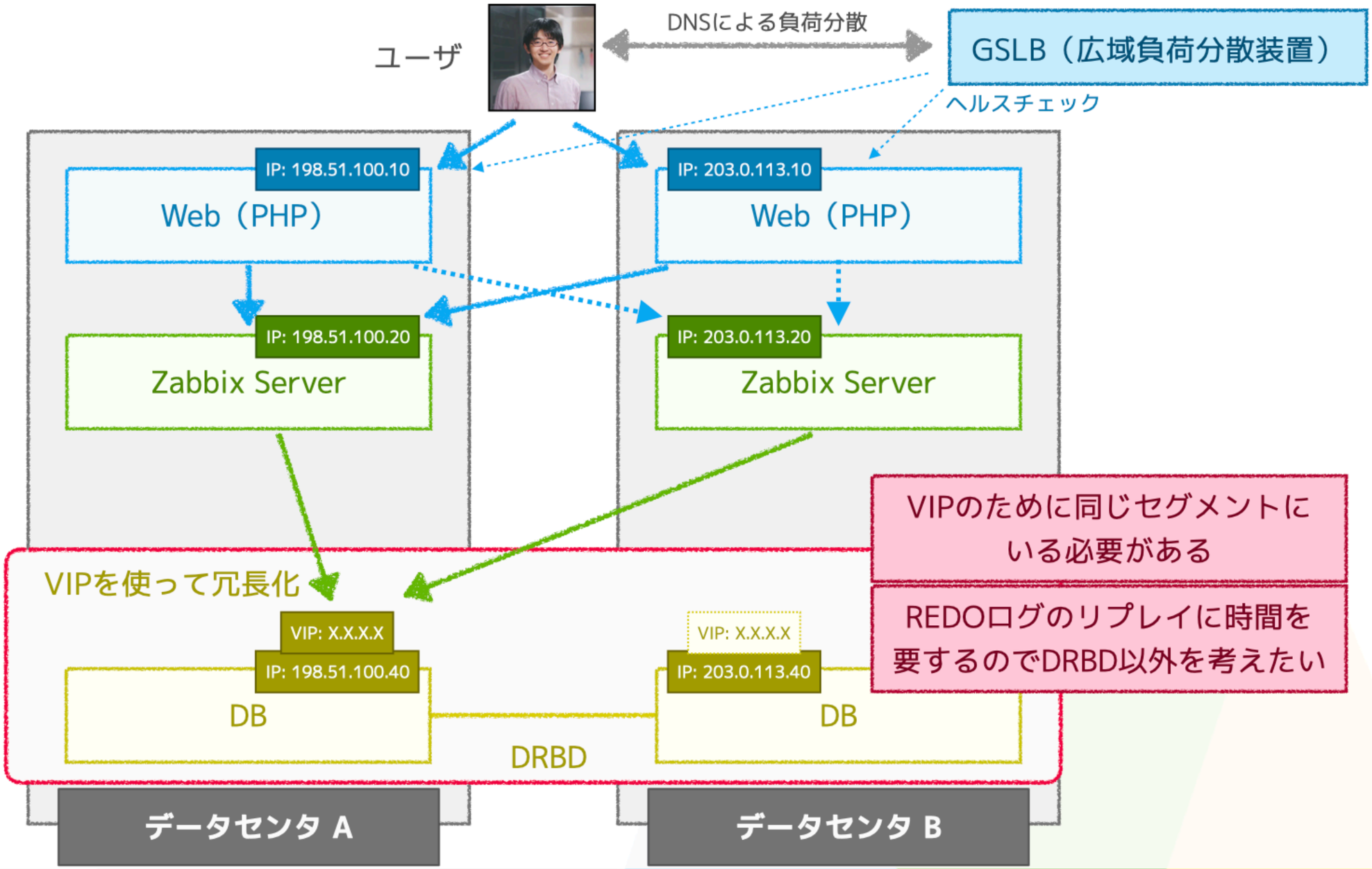
DBの冗長化



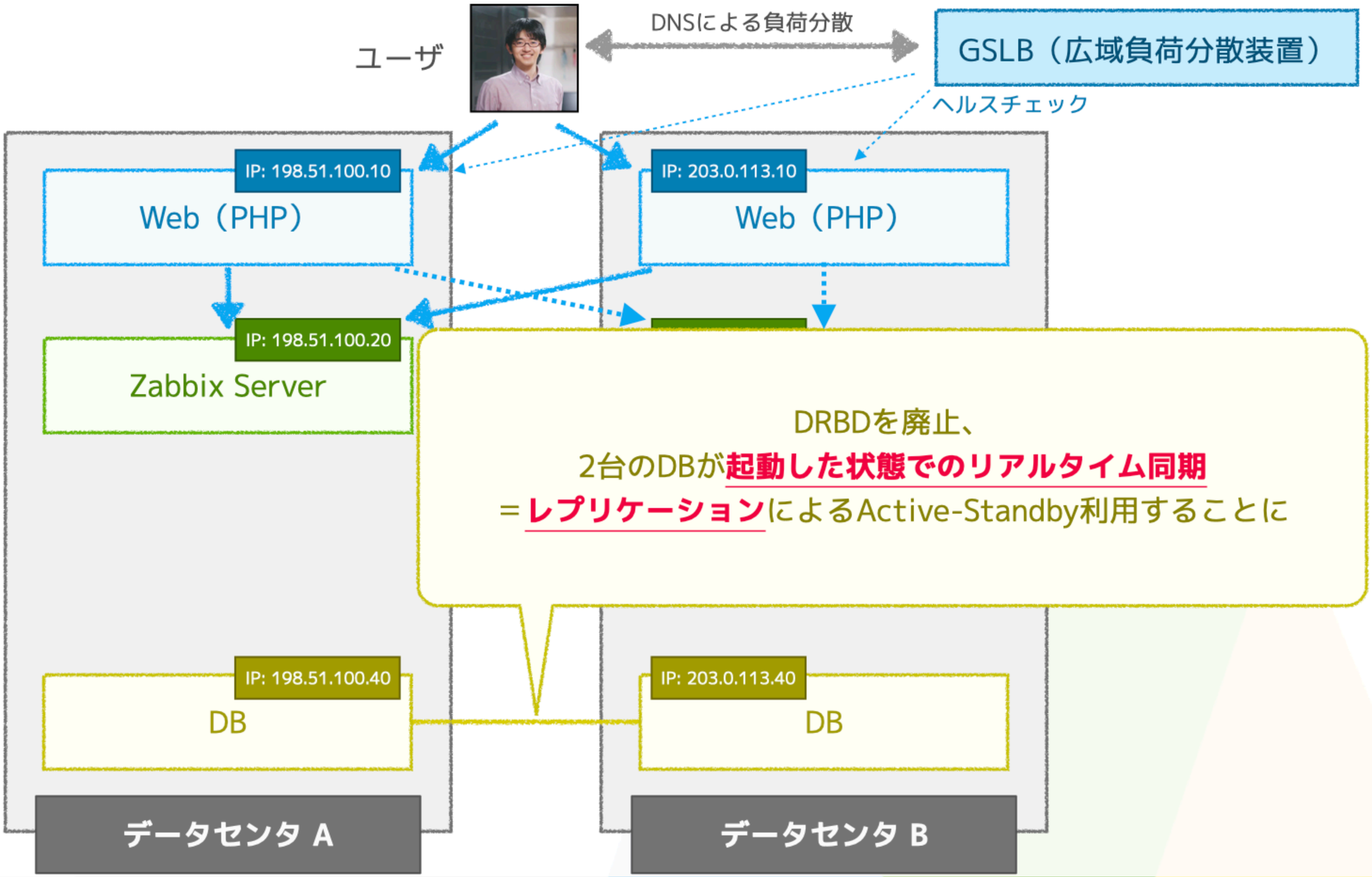
DBの冗長化



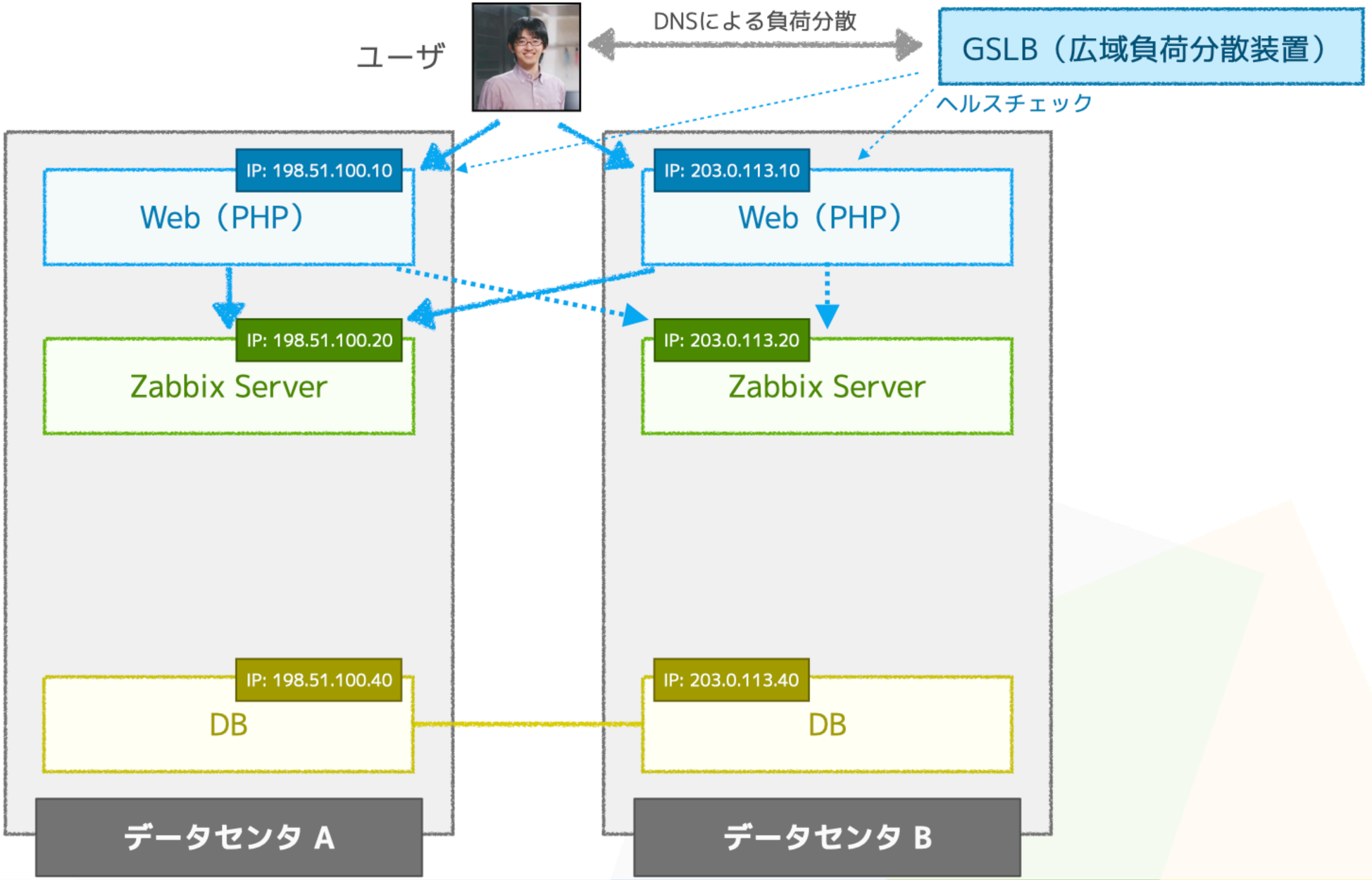
DBの冗長化



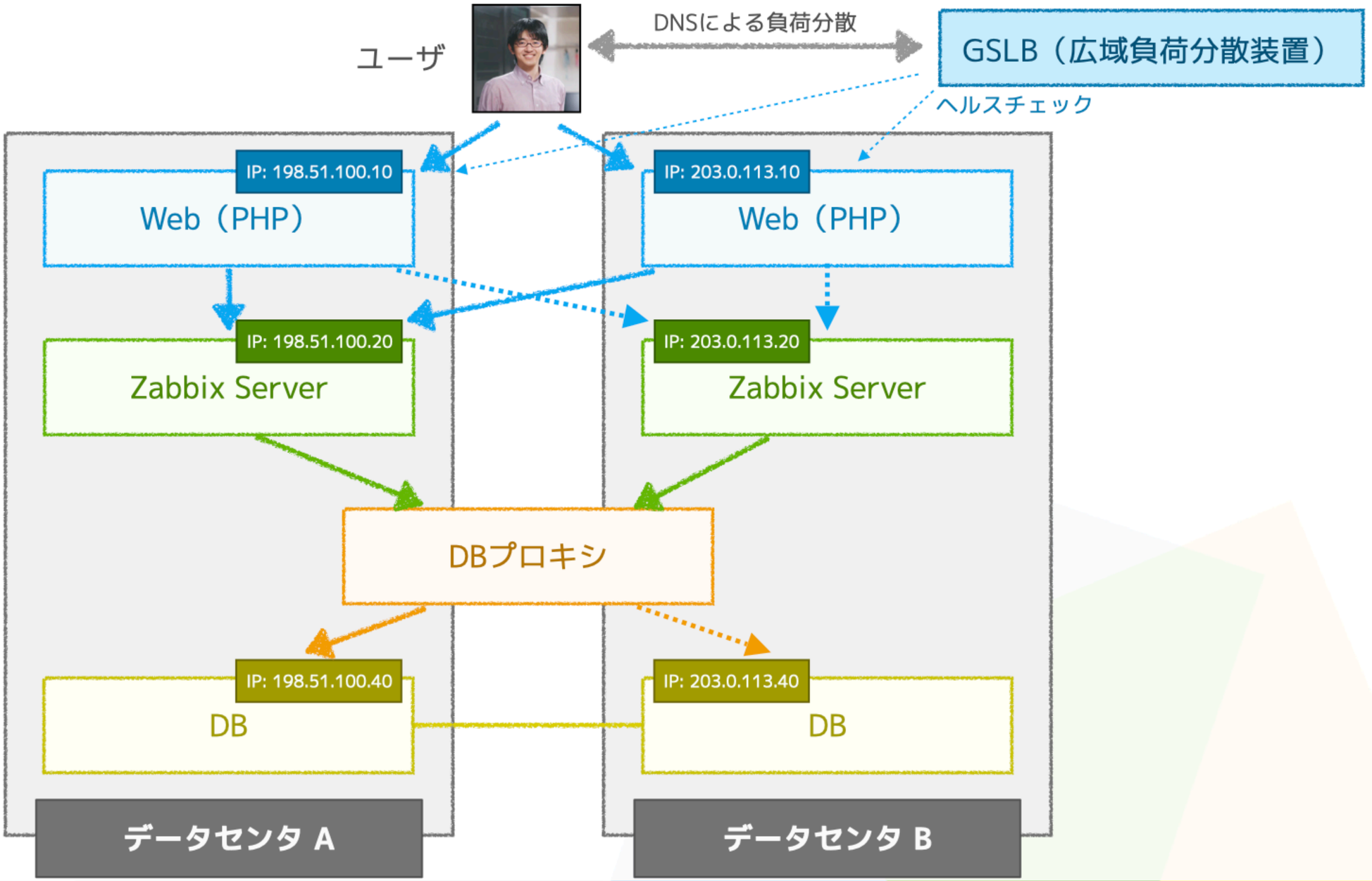
DBの冗長化



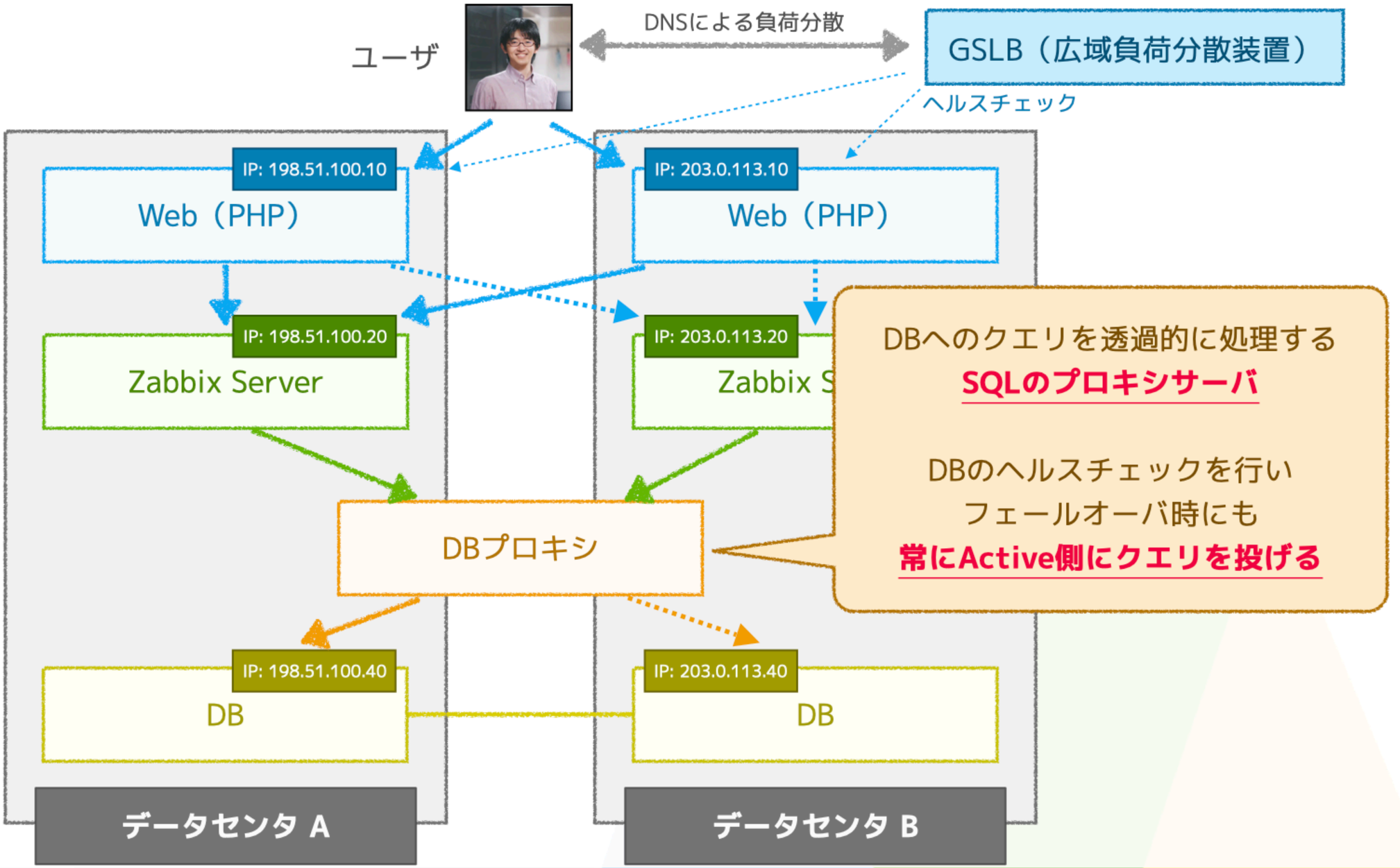
DBの冗長化



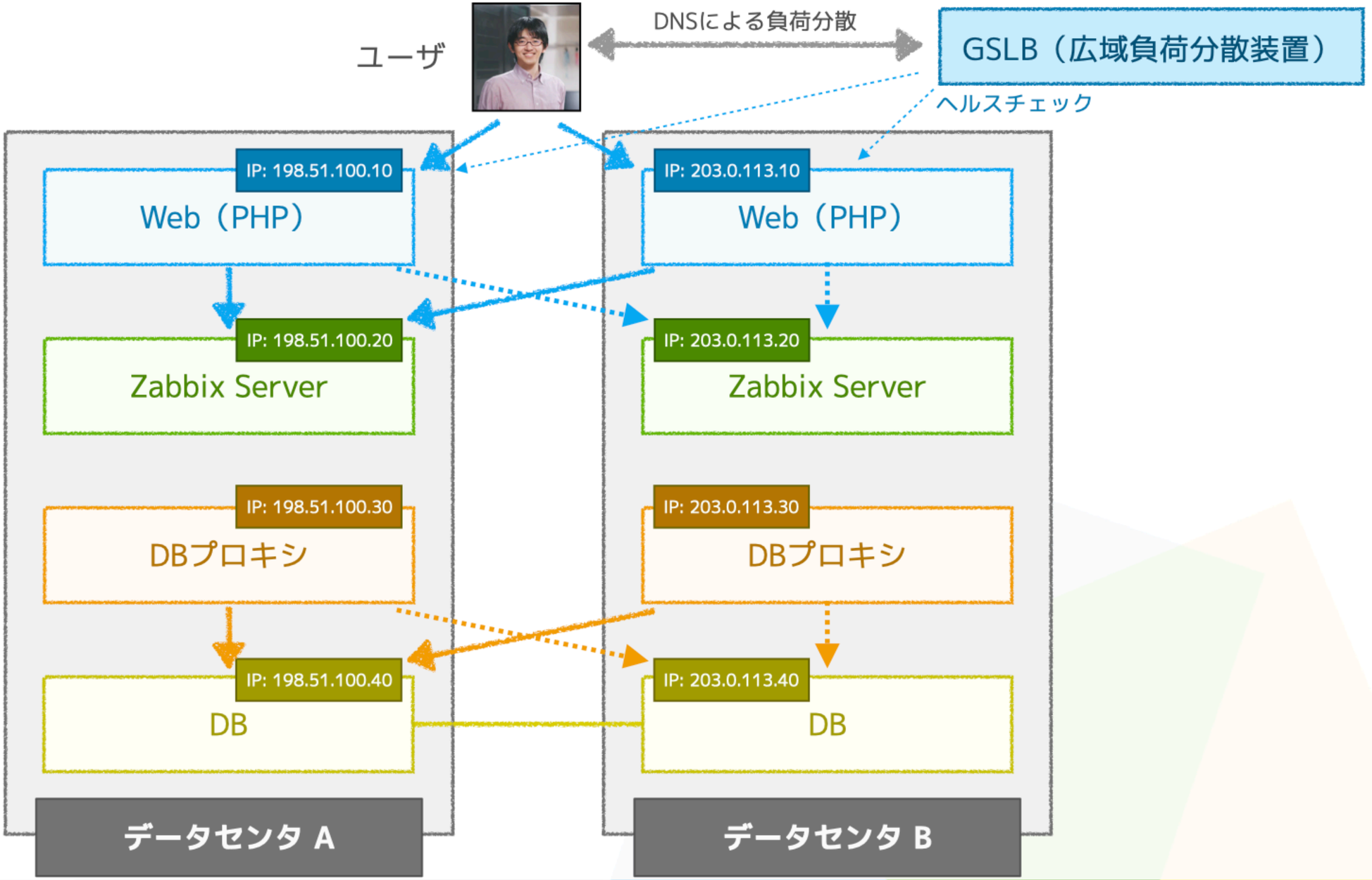
DBの冗長化



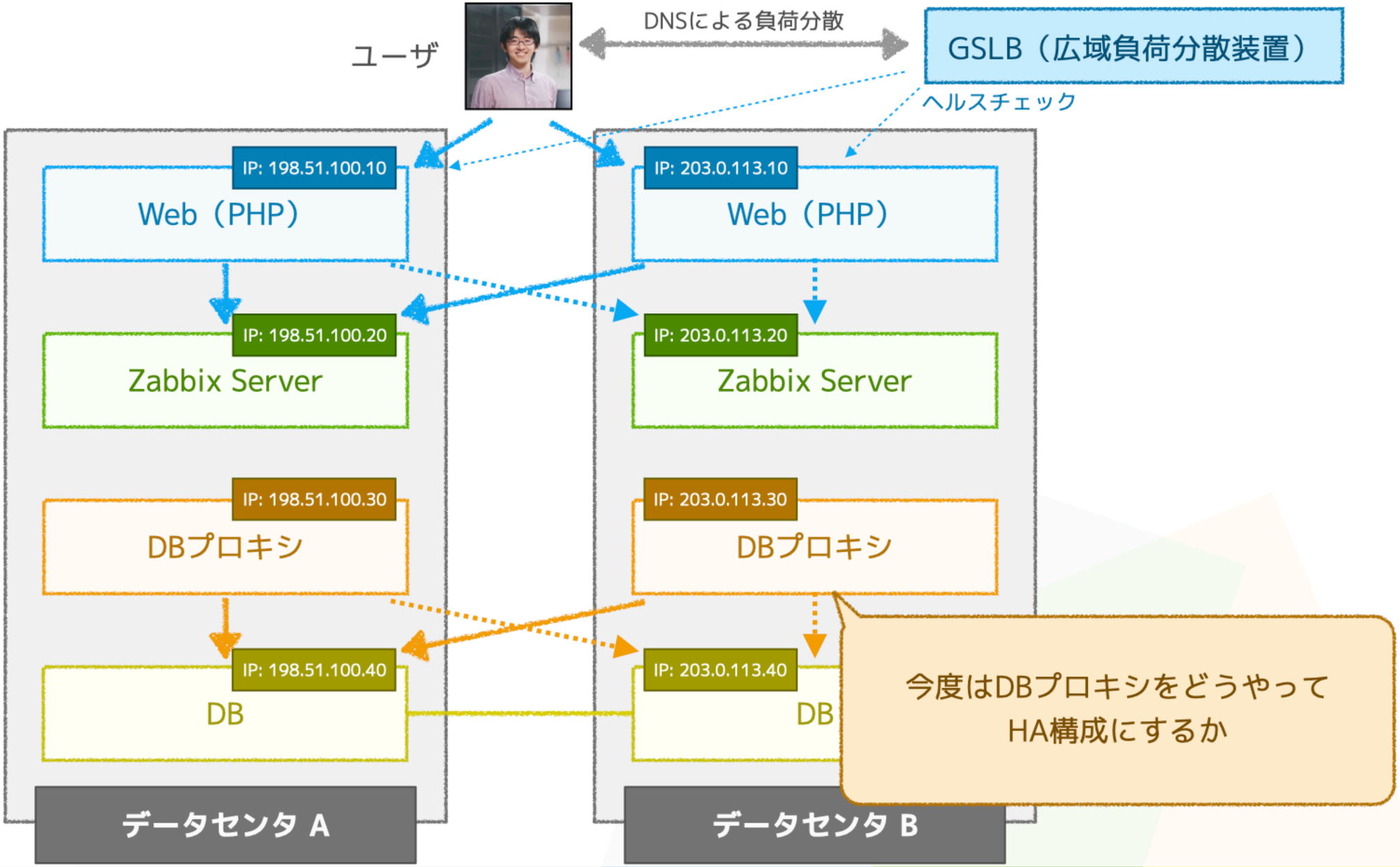
DBの冗長化



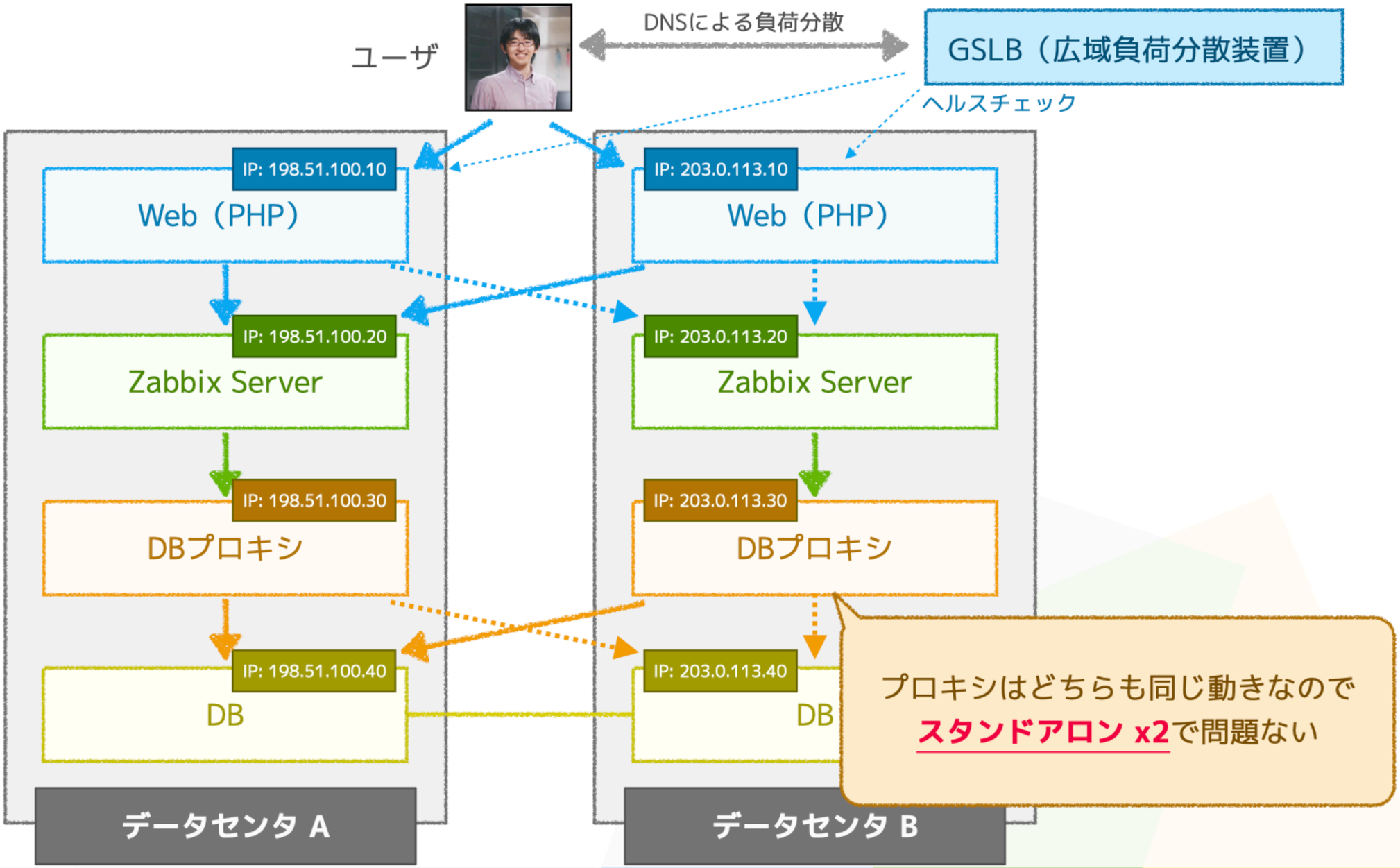
DBの冗長化



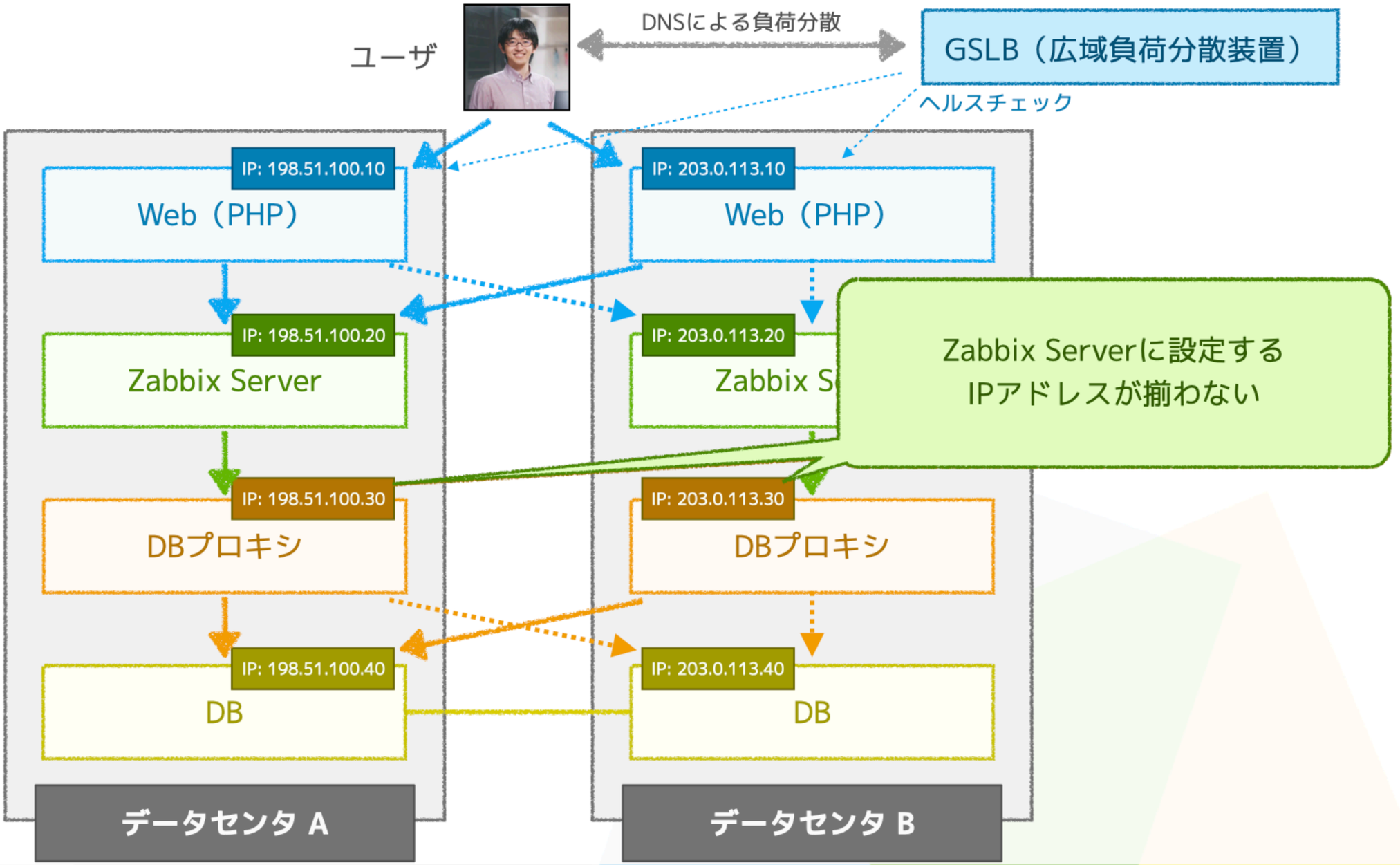
DBの冗長化



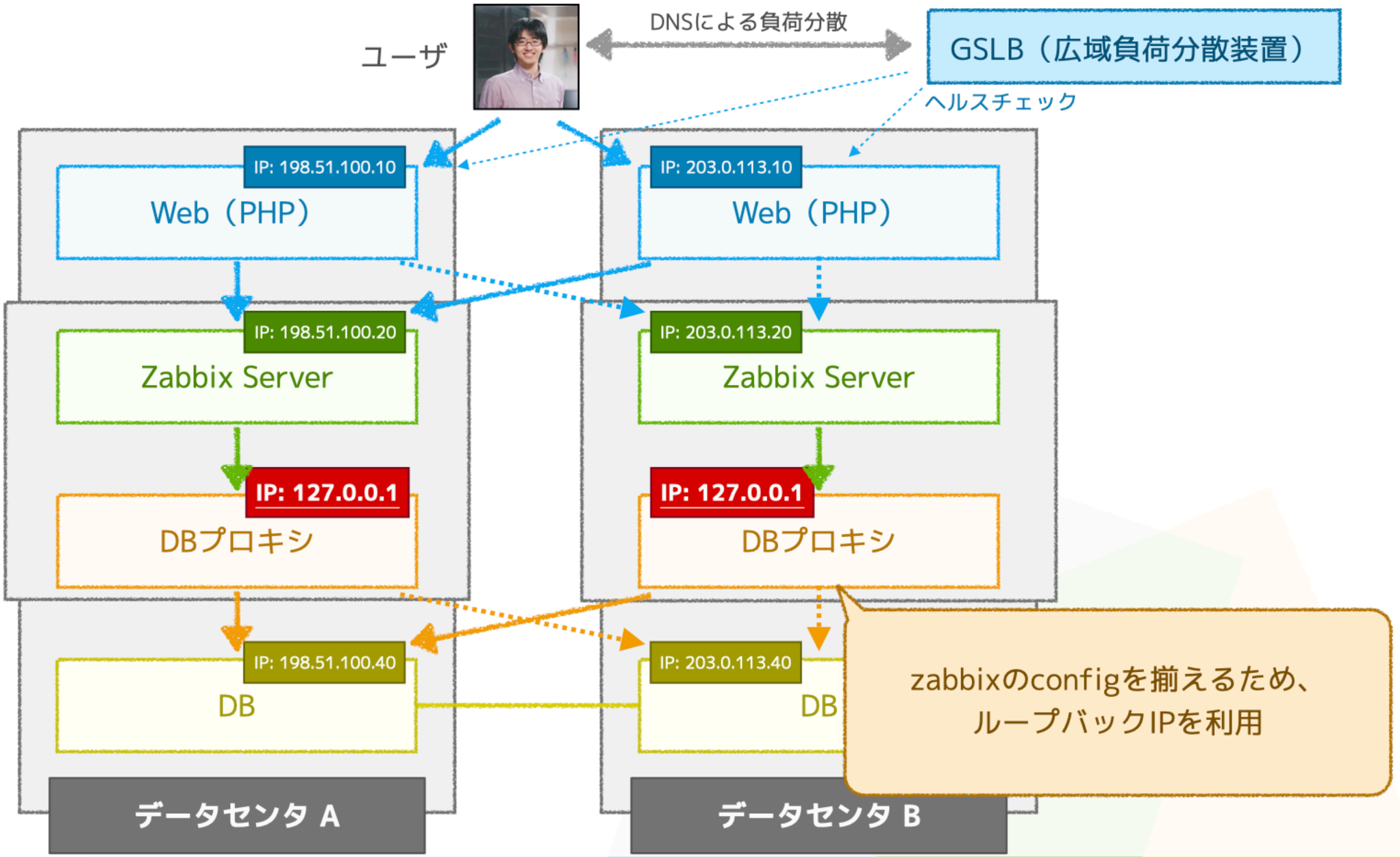
DBの冗長化



DBの冗長化



DBの冗長化

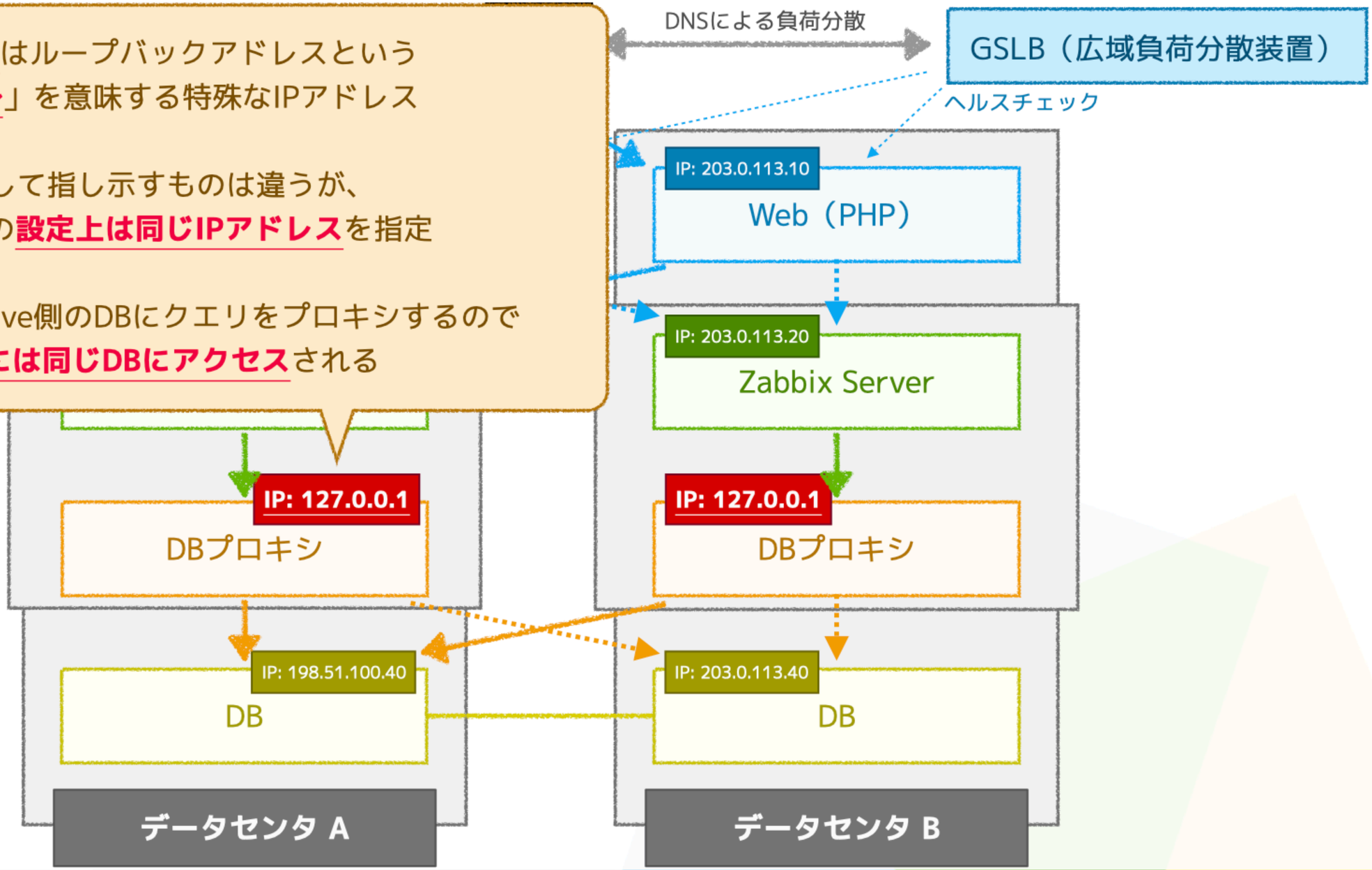


DBの冗長化

127.0.0.1はループバックアドレスという
「**自分自身**」を意味する特殊なIPアドレス

実体として指し示すものは違うが、
Zabbix HAの**設定上は同じIPアドレス**を指定

DBプロキシがActive側のDBにクエリをプロキシするので
最終的には同じDBにアクセスされる



DBの冗長化

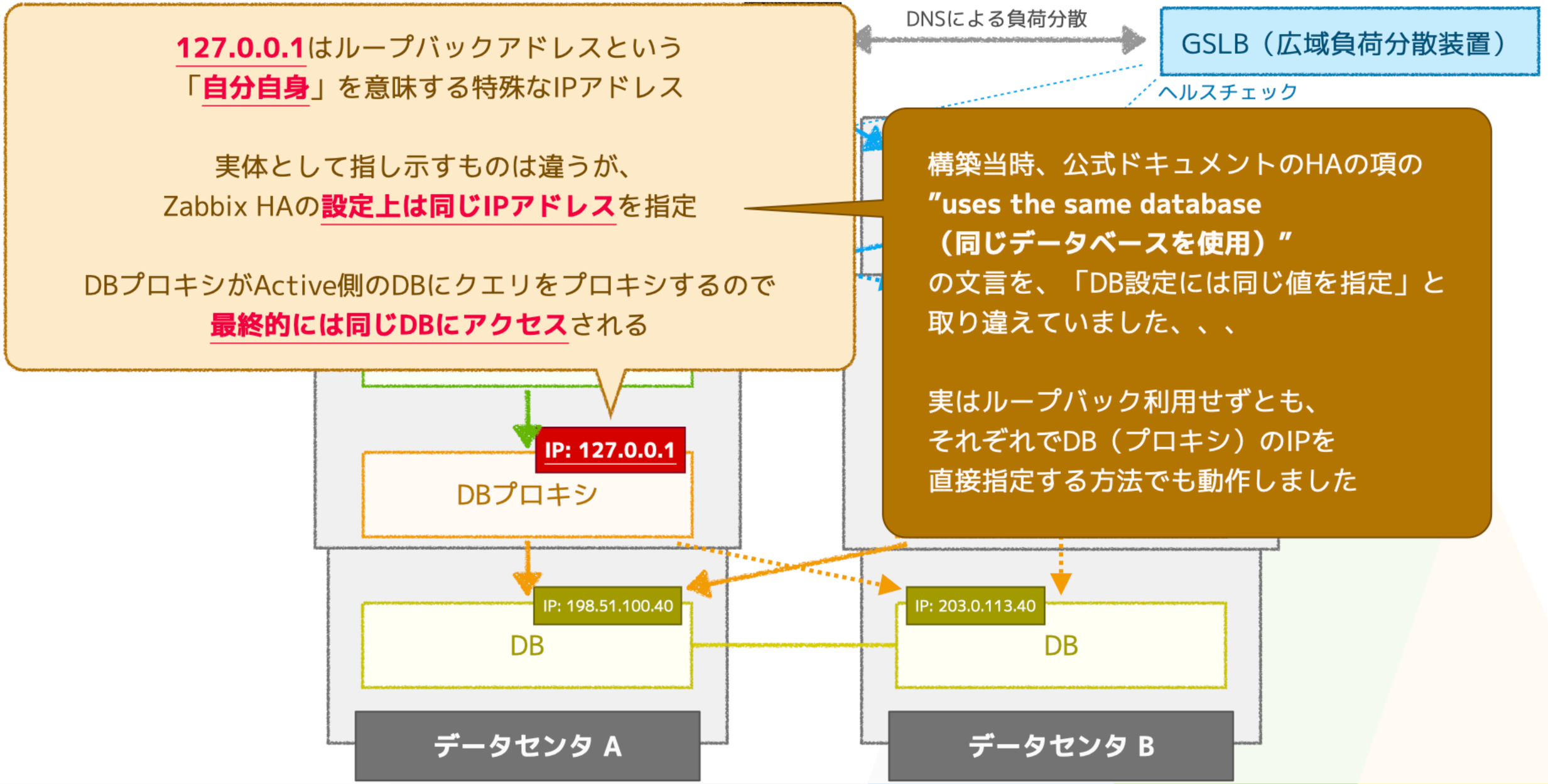
127.0.0.1はループバックアドレスという
「**自分自身**」を意味する特殊なIPアドレス

実体として指し示すものは違うが、
Zabbix HAの**設定上は同じIPアドレス**を指定

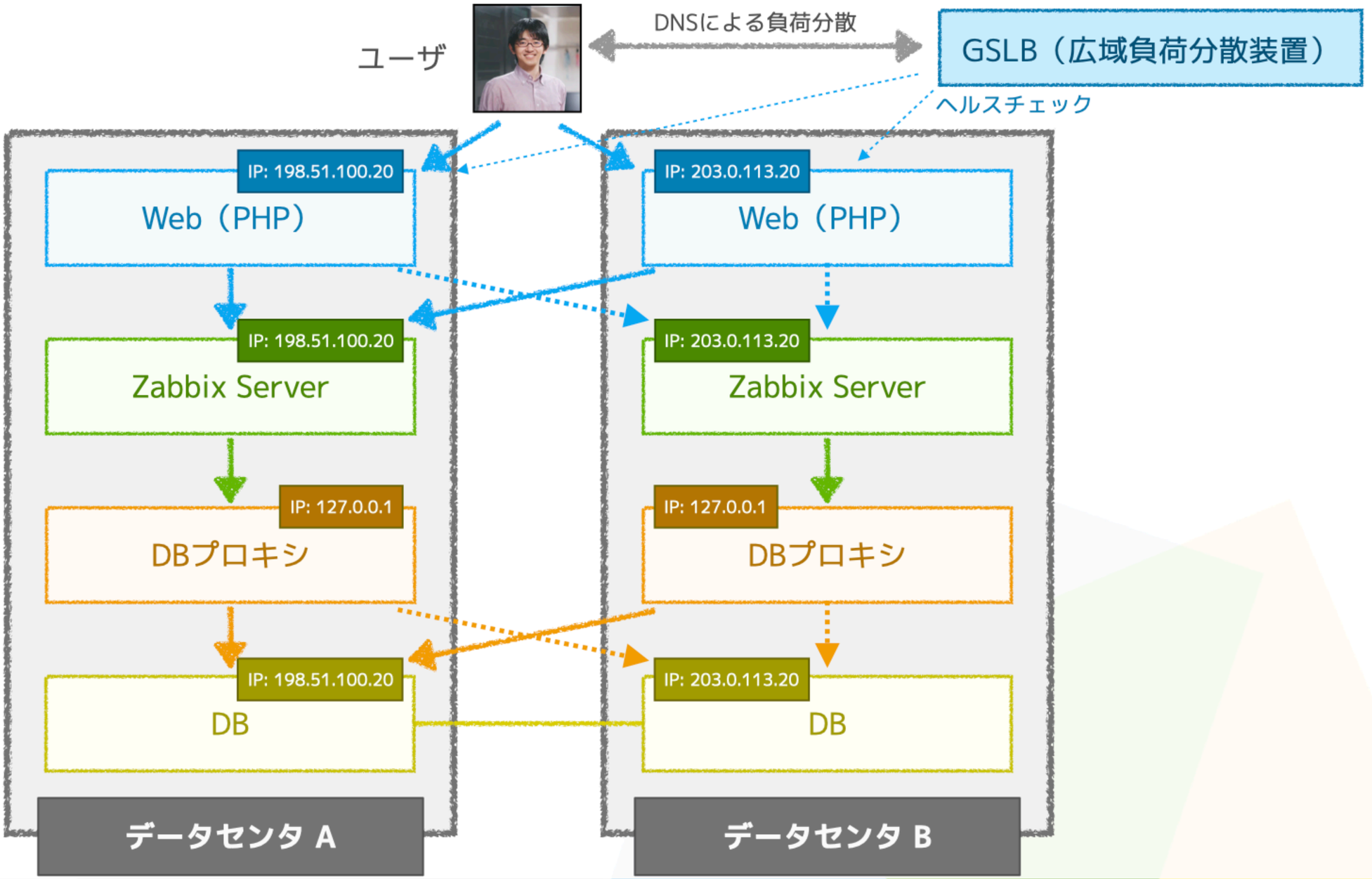
DBプロキシがActive側のDBにクエリをプロキシするので
最終的には同じDBにアクセスされる

構築当時、公式ドキュメントのHAの項の
**"uses the same database
（同じデータベースを使用）"**
の文言を、「DB設定には同じ値を指定」と
取り違えていました、、、

実はループバック利用せずとも、
それぞれでDB（プロキシ）のIPを
直接指定する方法でも動作しました



DBの冗長化



DBの選定

- MySQL
  - 既存のzabbix環境で利用していた
  - DBプロキシにはProxySQLというツールを利用
- PostgreSQL
  - 社内の他のインフラ系ツールで利用している
  - DBプロキシにはPgpool-IIというツールを利用

本業はネットワークエンジニアリングのため、
DB固有のトラブルにハマりたくない

ノウハウを再利用できるほうが嬉しい

PostgreSQL（ストリーミングレプリケーション）
+
Pgpool-II

最終構成

GLSBを用いて
Active-Activeの分散

ZabbixのHA機能での
Active-Standby構成

PostgreSQLのHAのため
Pgpool-IIを導入

PostgreSQLの機能での
レプリケーション &
Pgpool-IIによるHA

ユーザ

DNSによる負荷分散

GSLB（広域負荷分散装置）

ヘルスチェック

IP: 198.51.100.20
Web（PHP）

IP: 203.0.113.20
Web（PHP）

IP: 198.51.100.20
Zabbix Server

IP: 203.0.113.20
Zabbix Server

IP: 127.0.0.1
Pgpool-II

IP: 127.0.0.1
Pgpool-II

IP: 198.51.100.20
DB（PostgreSQL）

IP: 203.0.113.20
DB（PostgreSQL）

レプリケーション

データセンタ A

データセンタ B

今後の課題点

もっと素敵な伝え方を。 Jstream

ユーザ

DNSによる負荷分散

GSLB（広域負荷分散装置）

ヘルスチェック

GLSBを用いて
Active-Activeの

ZabbixのHA機能
Active-Standby

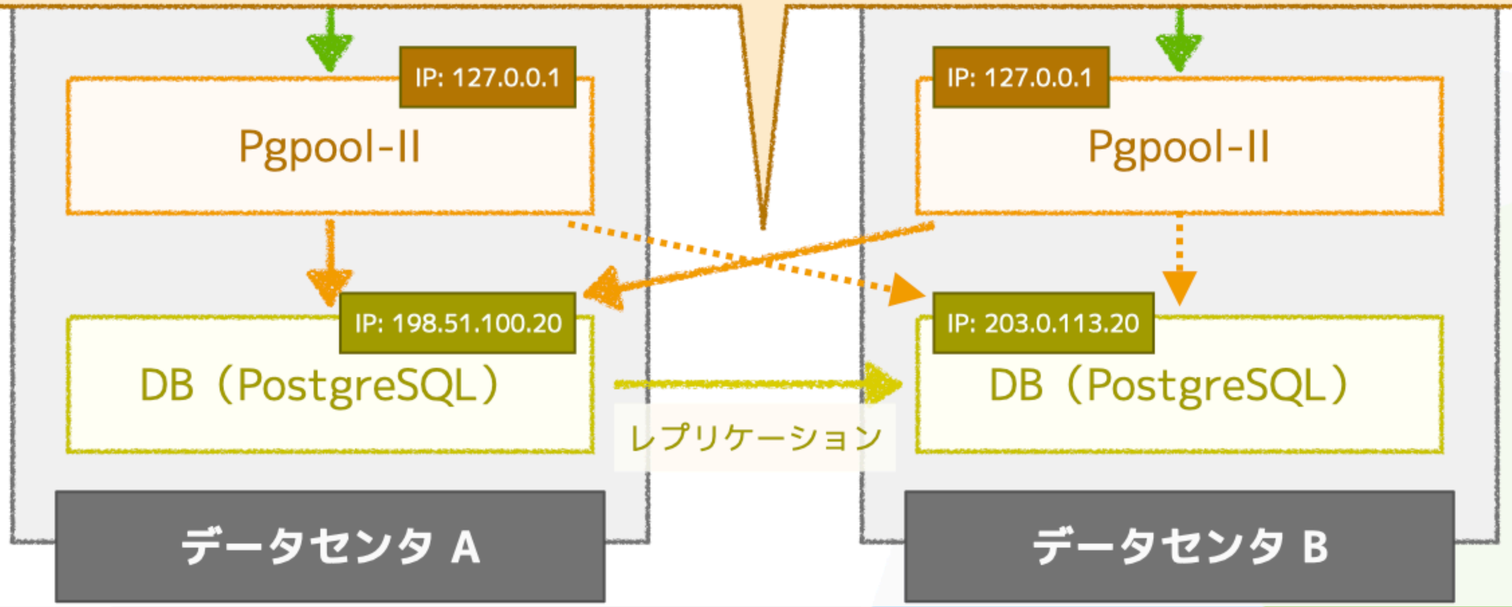
PostgreSQLのHAのため
Pgpool-IIを導入

PostgreSQLの機能での
レプリケーション &
Pgpool-IIによるHA

フェールオーバー周りは注意が必要

PostgreSQLはレプリケーションの機能はあるが、自動フェールオーバーの機能はなく
Pgpool-IIがPostgreSQLのヘルスチェック及びフェールオーバーをキックする

2台のPgpool-IIは完全に独立しているため
DBがダウンしたときに、**フェールオーバーが2回発生**してしまう



Pgpool-IIによるフェールオーバー

- Pgpool-IIが定期的にバックエンドのpostgresqlにヘルスチェックを行う
- ヘルスチェックが失敗するとフェールオーバースクリプトがキックされる
  - ヘルスチェックが失敗したノードをdownとしてマークし、プロキシ先から除外
  - Standbyだったノードを昇格させ、Activeとして振る舞うようにさせる
- 同期レプリケーションのため、Standby側もダウンした際には切り離しが必要

普通にフェールオーバーを仕込むと、、？

- フェールオーバーがダブる
- L3で冗長化すると**経路上のNW障害によるヘルスチェック失敗**がある
  - フェールオーバーを早まると、スプリットブレーンが発生してしまう

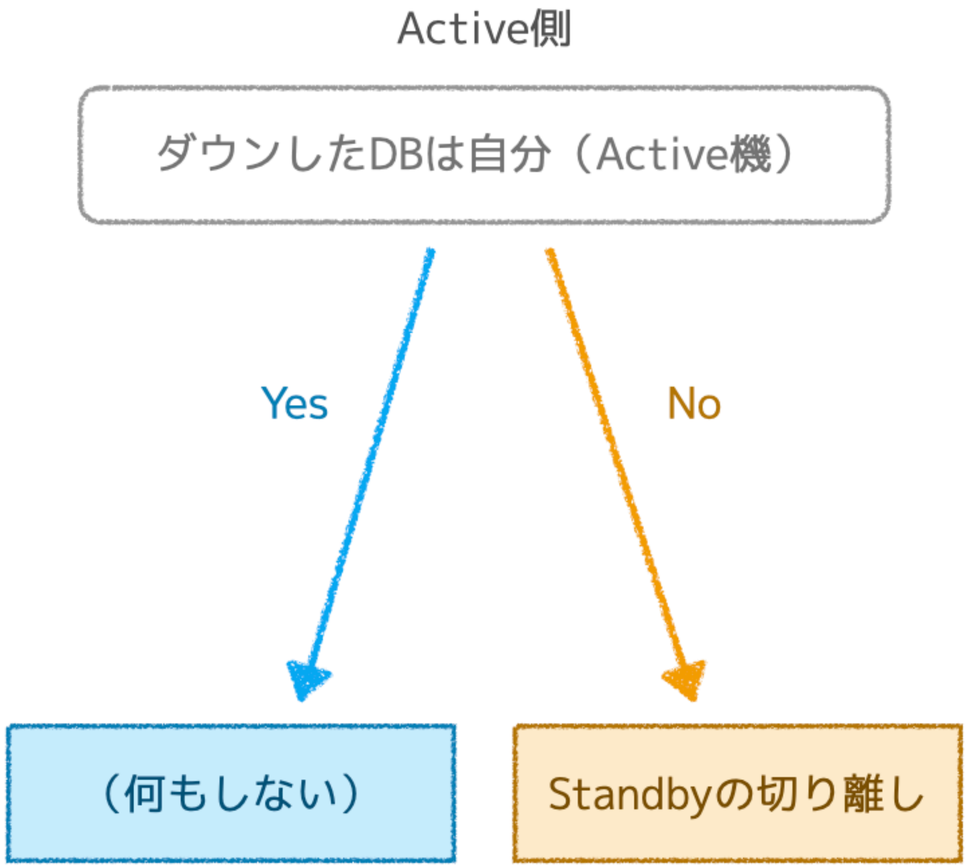
Pgpool-IIによるフェールオーバー

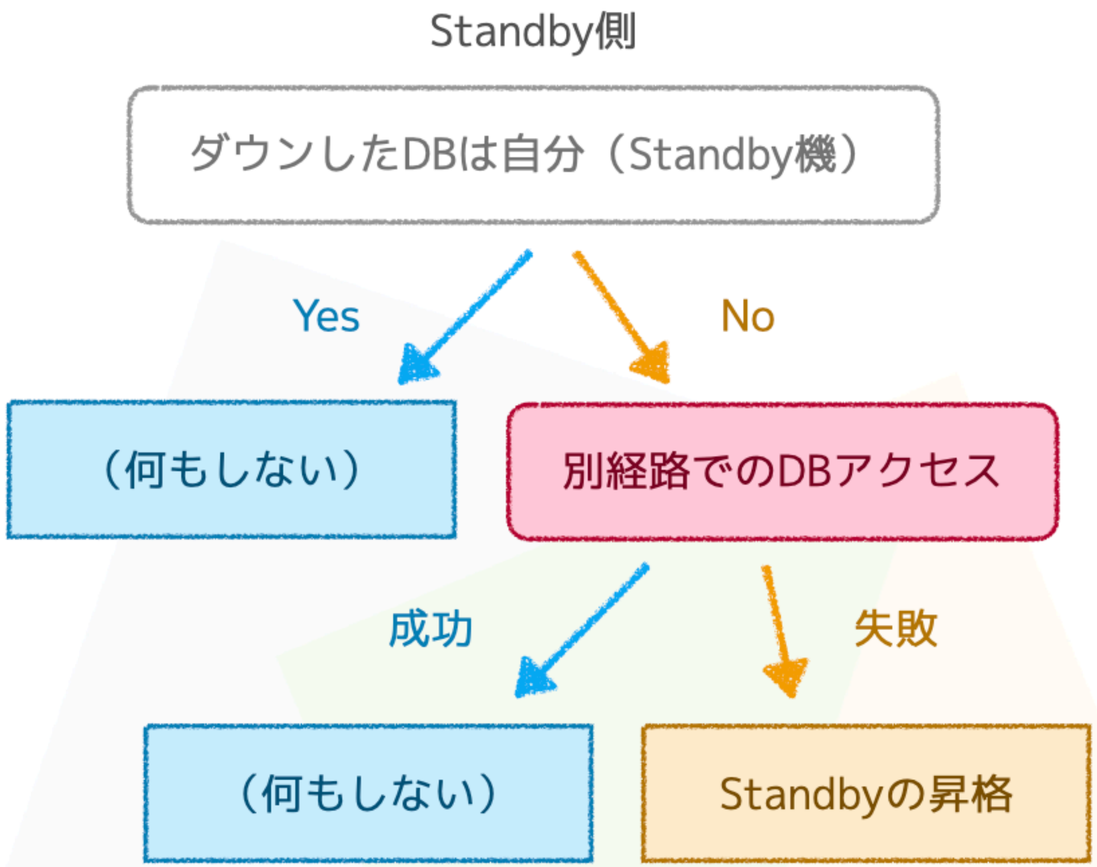
障害のパターン分けを行い、必要に合わせてフェールオーバーをキックするように

- Active側/Standby側のサーバ全体がハング
- Active側/Standby側のpgpoolがハング
- Active側/Standby側のpostgresqlがハング
- Active-Standby間のネットワーク疎通がダウン

Pgpool-IIによるフェールオーバー

- 各ホストのPgpool-IIは、同一ホスト上のPostgreSQLのフェールオーバーのみ実施
- Active・Standbyそれぞれで**独立してフェールオーバー**をキック





構築で苦労した点

- 珍しい（想定されていないだろう）構成のため、何が正解であるかがわからない
  - 本当に求めているものができたか検証を繰り返し行うひつようがある
    - 要件定義をしっかりと行い、1つずつテストを実施した
- Pgpool-IIのフェールオーバー
  - これも想定されていないだろう構成のため起こり得るパターンを列挙して1つずつテスト

- 今後のアップデート
  - アップデートによりこの構成だと上手くいかない機能が出てきてしまうのではないか
  - 都度検証が必要
    - それはそれで楽しいからヨシ？