NEWS PICKS

を支える技術と怖い話

2014.12.18 - EBISTA #1 @ UZABASE, Inc.

アジェンダ

自己紹介

サービスの概要

サービスの特徴

NewsPicks を支える技術

本当にあった怖い話

まとめ

文字 拓郎

TAKURO MONJI

@monzou

経歴

・金融機関のデリバティブトレーディングシステム開発
・リッチクライアント + 大規模分散計算
・Web 経験すくなめ（去年から）
・2014 年 9 月 UZABASE 入社（3 ヶ月ほど経ちました）

UZABASE 入社後

・NewsPicks の開発担当（サーバーサイドと Web がメイン）
・今日のイベントの手配あれこれ
・畑違いの分野から来たので色々と新鮮です

アジェンダ

自己紹介

サービスの概要

サービスの特徴

NewsPicks を支える技術

本当にあった怖い話

まとめ

ご存知ですよね？

Naoya Ito
@naoya_ito

Following

NewsPicks のコメント欄がウゼーとか放言したところ昨日中の人に会ってしまい、ばれてないか非常にドキドキしたのですが案の定ばれてました。NewsPicks 最高ッス

RETWEETS 6　FAVORITES 35

3:52 PM - 16 Dec 2014

あの naoya 氏も絶賛

NewsPicks

経済に特化したニュースアプリ
世界中のビジネスマンが新しいビジネス情報を発見するための
情報インフラ（世界一の経済メディア）を目指している

- iPhone
- iPad
- Android
- Web

などマルチチャネルで展開

- 著名人
- 有識者
- 意思決定者層

などによる記事に対するコメント

サービスが提供するコンテンツ

・様々な経済情報をワンストップで提供
・著名人／有識者／意思決定者層によるキュレーション
・ユーザーのコメントにより多角的な視座を提供

Contents

無料コンテンツ

REUTERS
現代ビジネス
日経ビジネス ONLINE
ITmedia
東洋経済 TOYOKEIZAI ONLINE

自社編集部による
独自記事

有料コンテンツ

The New York Times
週刊ダイヤモンド
THE WALL STREET JOURNAL.
Metal & Technology 鉄鋼新聞
财新网 Caixin.com

サービスが提供するコンテンツ

・様々な経済情報をワンストップで提供
・著名人／有識者／意思決定者層によるキュレーション
・ユーザーのコメントにより多角的な視座を提供

有料課金モデル

Contents

無料コンテンツ

REUTERS
現代ビジネス
日経ビジネス ONLINE
ITmedia
東洋経済 TOYOKEIZAI ONLINE

自社編集部による
独自記事

有料コンテンツ

The New York Times
週刊ダイヤモンド
THE WALL STREET JOURNAL.
Metal & Technology 鉄鋼新聞
财新网 Caixin.com

サービスが提供するコンテンツ

・様々な経済情報をワンストップで提供
・著名人／有識者／意思決定者層によるキュレーション
・ユーザーのコメントにより多角的な視座を提供

有料課金モデル

無料コンテンツ

有料コンテンツ

Contents

REUTERS
現代ビジネス
日経ビジネス ONLINE
ITmedia
東洋経済 TOYOKEIZAI ONLINE

自社編集部による
独自記事

The New York Times
週刊ダイヤモンド
THE WALL STREET JOURNAL.
Metal & Technology 鉄鋼新聞
财新网 Caixin.com

Picker

サービスが提供するコンテンツ

・様々な経済情報をワンストップで提供
・著名人／有識者／意思決定者層によるキュレーション
・ユーザーのコメントにより多角的な視座を提供

有料課金モデル

無料コンテンツ

有料コンテンツ

Contents

REUTERS
現代ビジネス
日経ビジネス ONLINE
ITmedia
東洋経済 ONLINE

自社編集部による
独自記事

The New York Times
週刊ダイヤモンド
THE WALL STREET JOURNAL.
Metal & Technology 鉄鋼新聞
財新网 Caixin.com

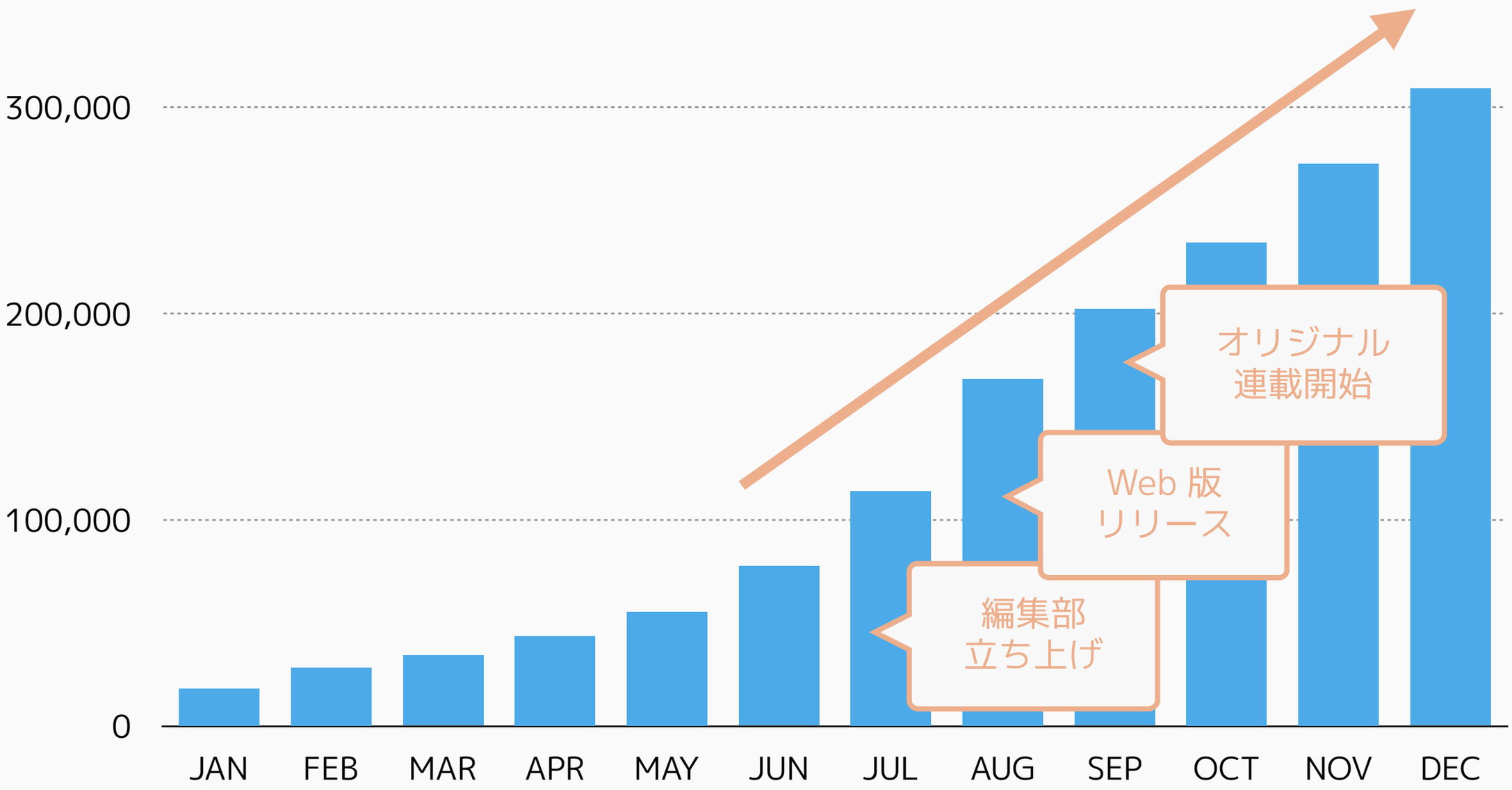
Picker

有識者による
選別・コメント

Platform

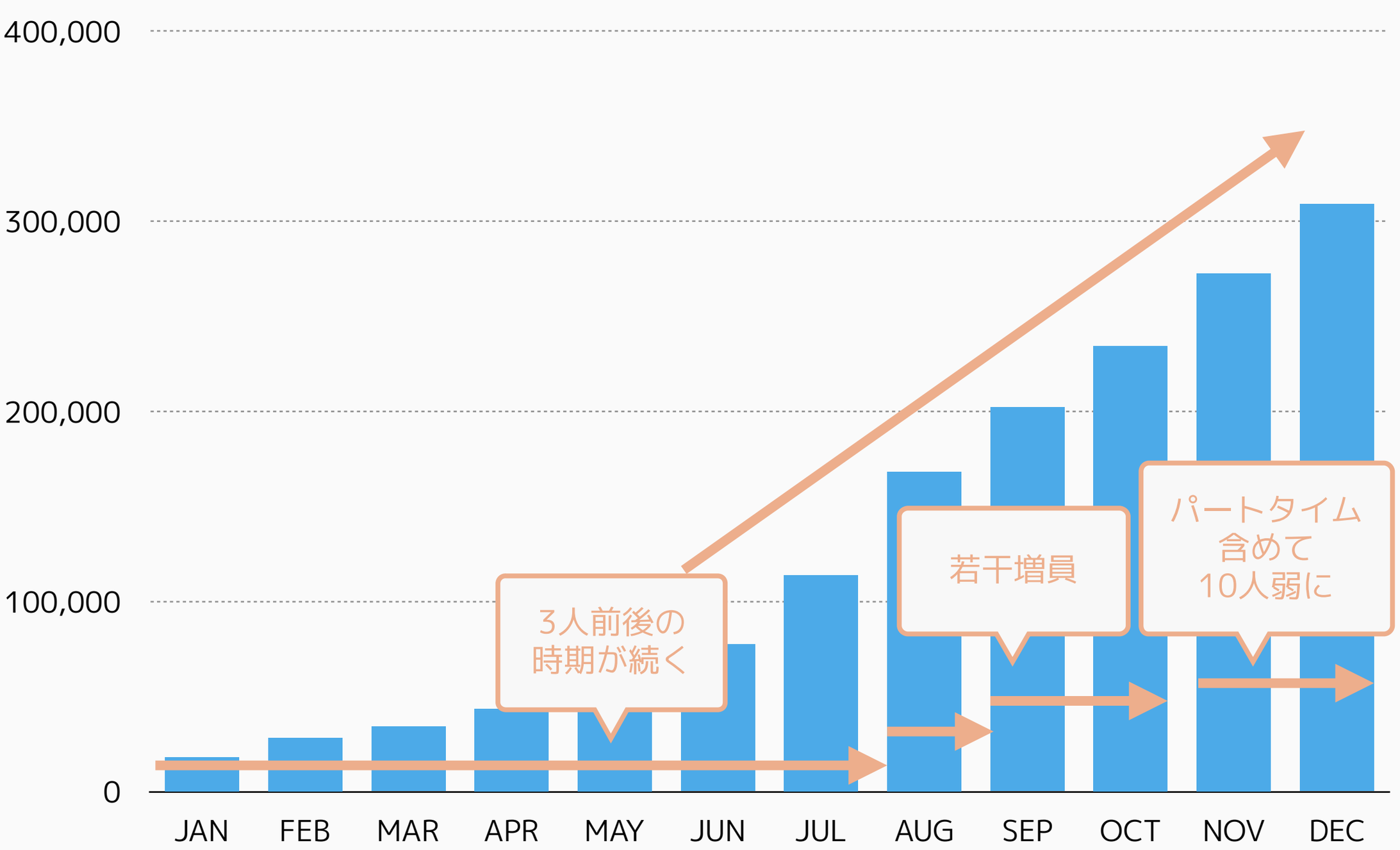
サービスの成長

ユーザー数は 30 万人を突破



開発チームの成長

最近ようやくチームっぽく …



アジェンダ

自己紹介

サービスの概要

サービスの特徴

NewsPicks を支える技術

本当にあった怖い話

まとめ

特徴 ① 自社コンテンツ

社内に編集部を設置し, オリジナルコンテンツを提供
3 年後には編集部だけで 100 名体制へ
世界一の経済メディアを目指す

・単なるニュースのキュレーションアプリでなく経済メディアへ
・独自記事だけでなく記事の編成なども行う

編集部用の社内システムも構築・運用

・独自記事入稿・効果測定
・おすすめニュースの編成
・おすすめユーザーの編成

特徴 ② ヒトの手による価値向上

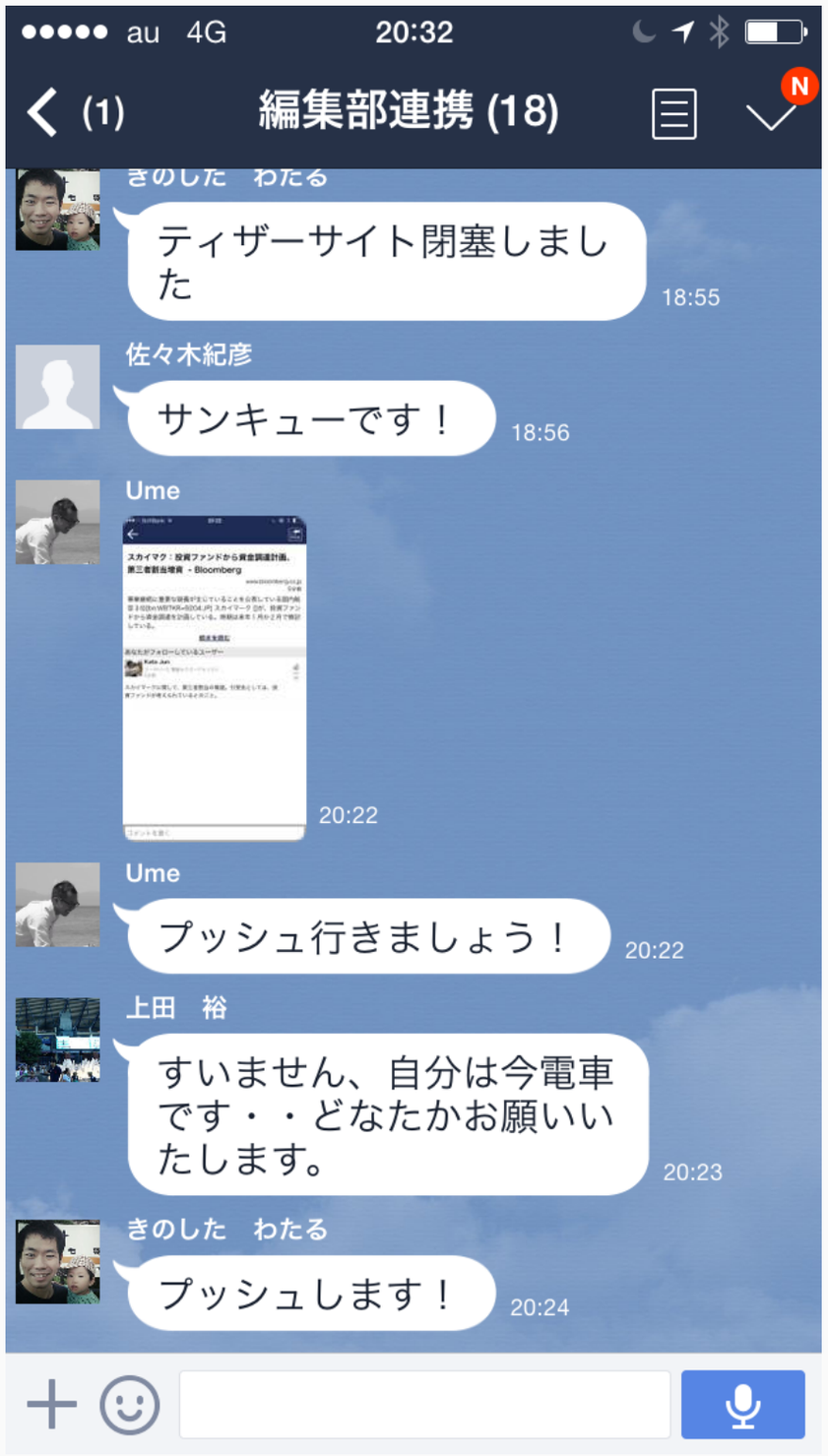
記事編成・レコメンド

・編集部 + アナリスト + エンジニアのコラボレーション
・社内システムによってオススメ記事やオススメユーザーを管理
・ヒトとアルゴリズムの融合によるレコメンド

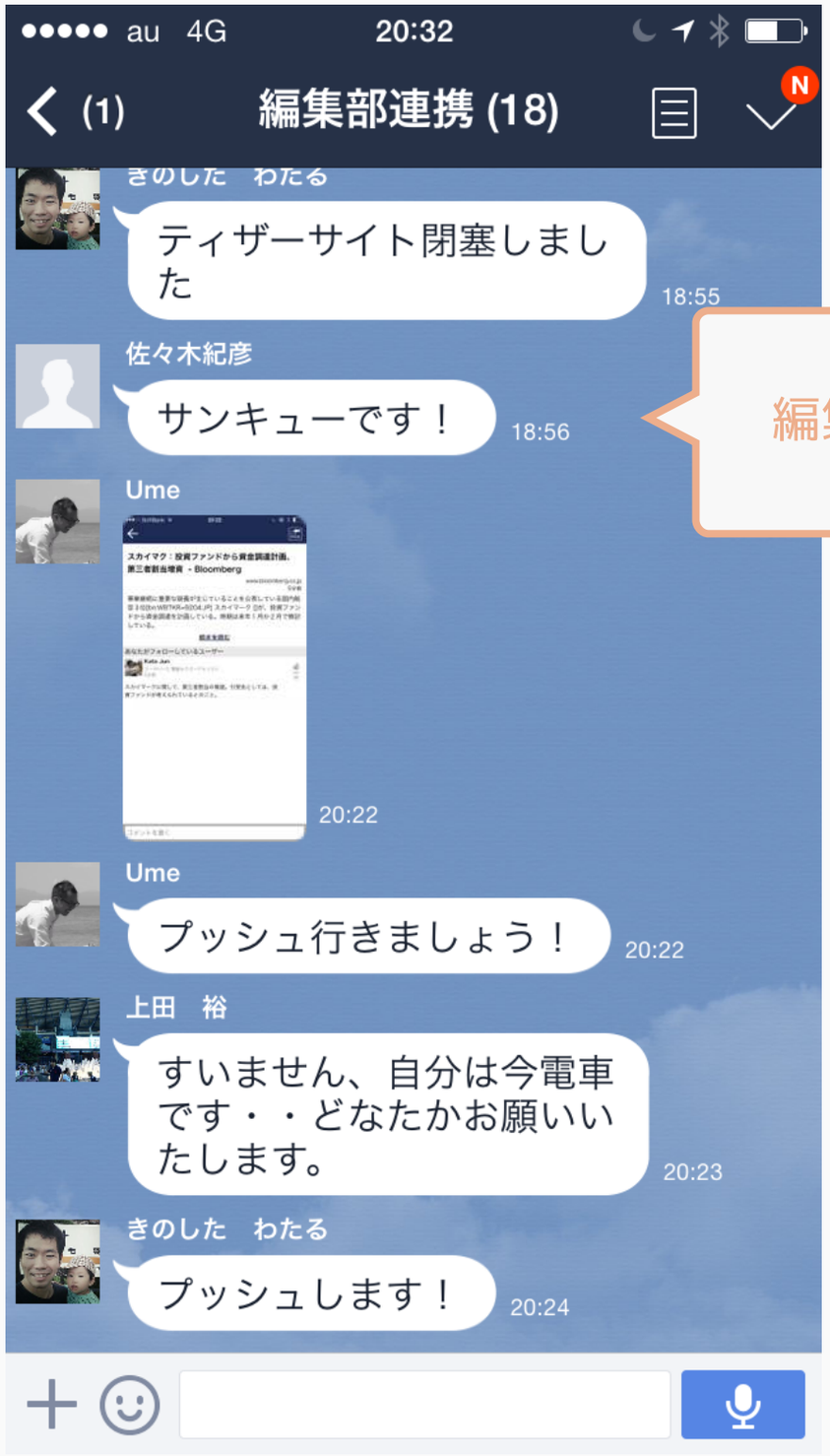
有料会員向けのイベントや NewsPicks Paper の配布

・有料会員限定のリアルイベントの開催
・NewsPicks Paper などの特別媒体を配布

やる気に満ちた経営陣と荒ぶる LINE

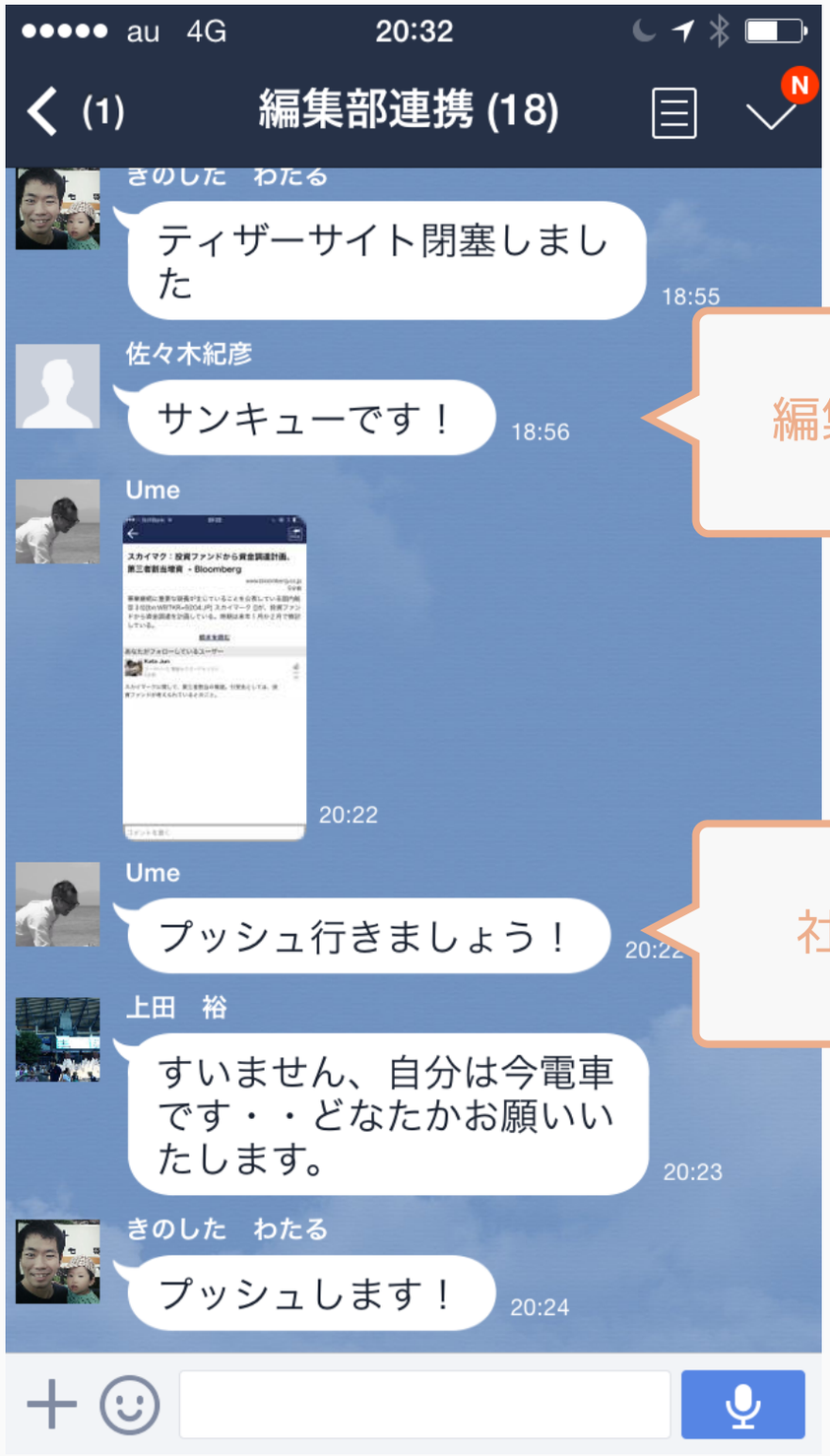


やる気に満ちた経営陣と荒ぶる LINE



編集長です

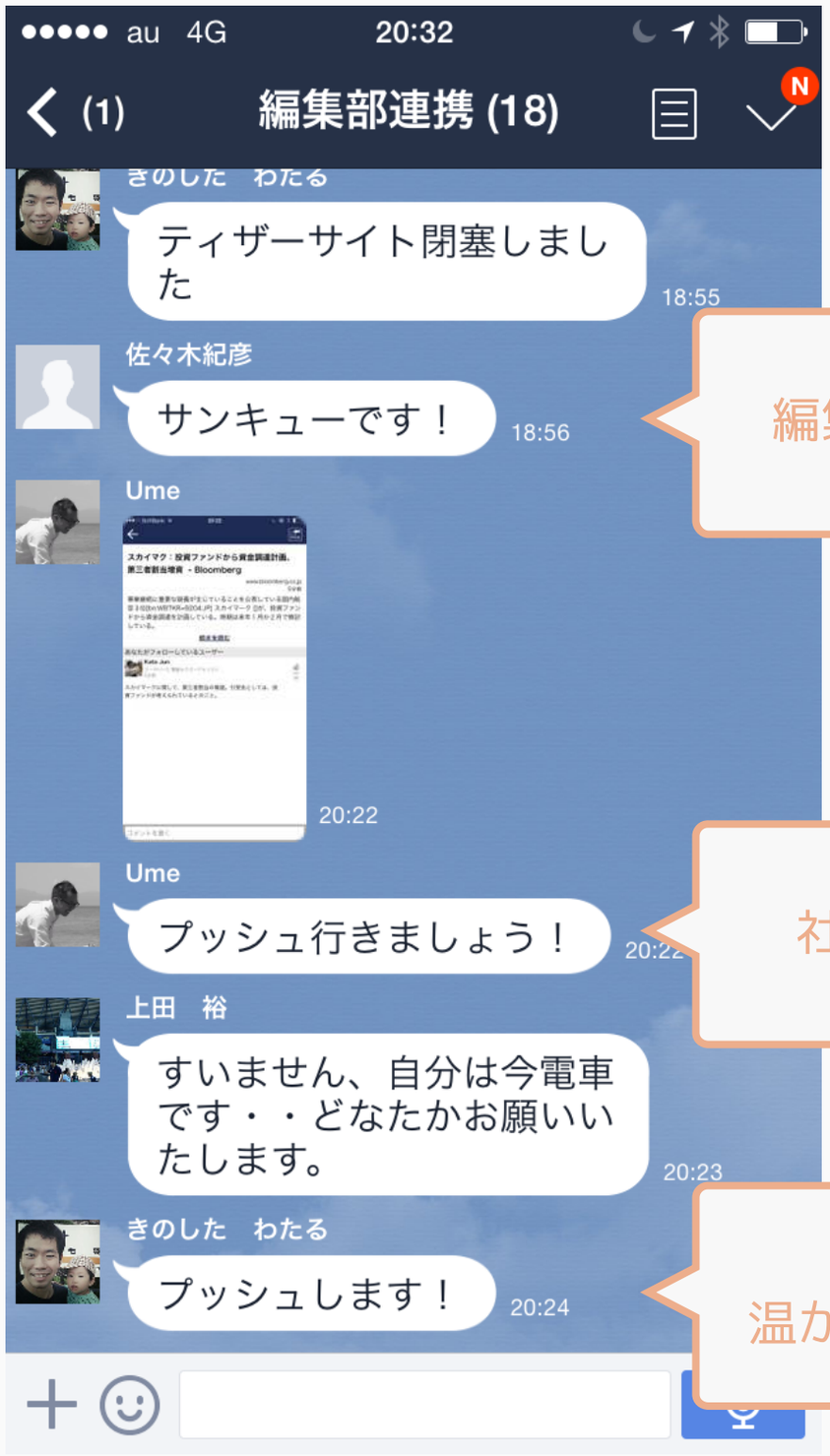
やる気に満ちた経営陣と荒ぶる LINE



編集長です

社長です

やる気に満ちた経営陣と荒ぶる LINE



編集長です

社長です

職人の手による
温かみのある手動プッシュ通知の様子です

特徴 ③ SPEEDA の資産を活用

SPEEDA

・UZABASE が提供する企業・産業分析用の情報プラットフォーム
・SPEEDA の膨大な資産を NewsPicks に活用

具体的な活用内容

・SPEEDA で利用している記事分類アルゴズム（機械学習）の流用
・社内アナリストによる質の高いコメントや分析記事
・今後 SPEEDA のデータベースを活用したより高度な連携も予定

アジェンダ

自己紹介

サービスの概要

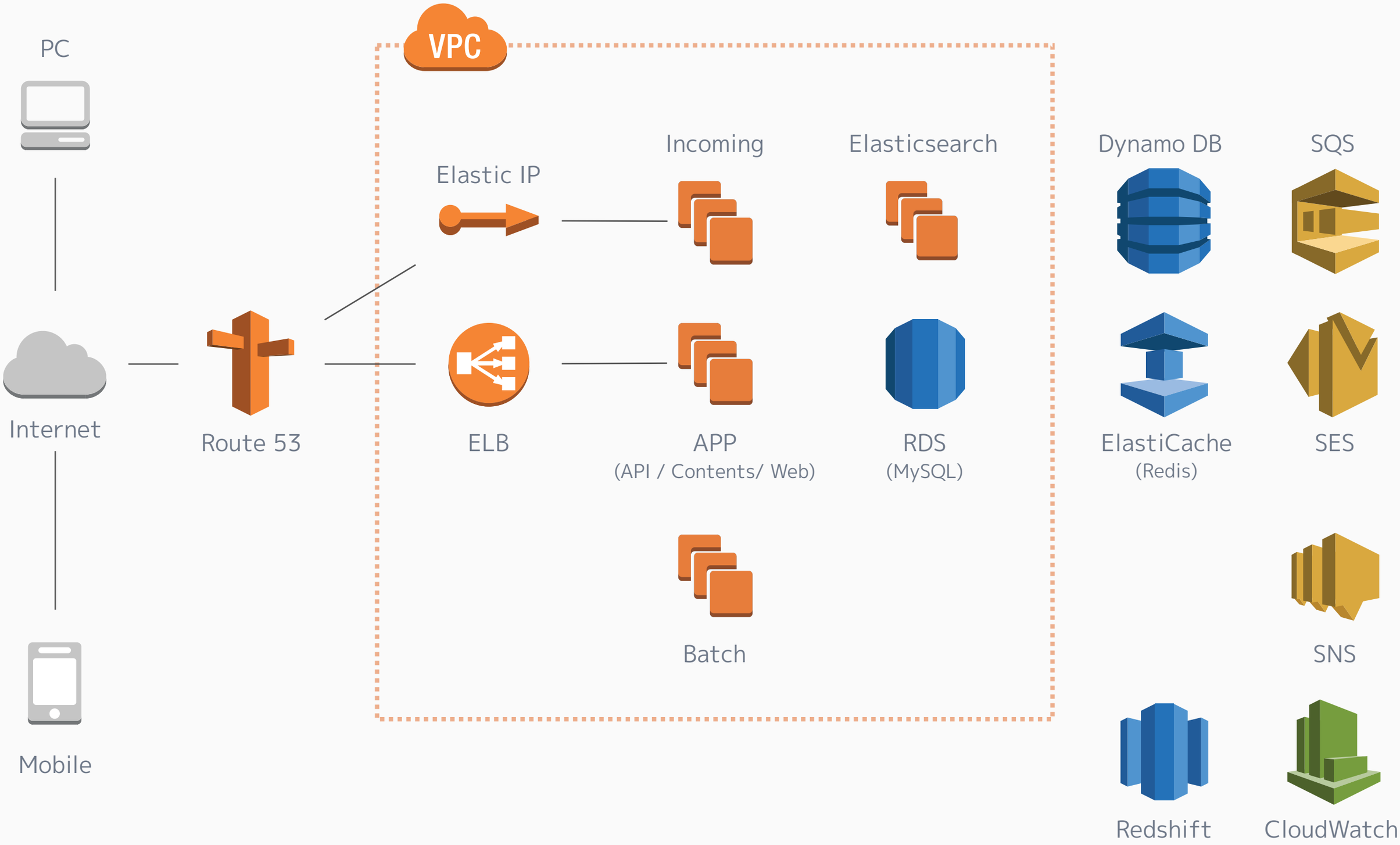
サービスの特徴

NewsPicks を支える技術

本当にあった怖い話

まとめ

インフラ構成の概要


PC
VPC
Incoming
Elasticsearch
Dynamo DB
SQS
Elastic IP
Internet
Route 53
ELB
APP
(API / Contents/ Web)
RDS
(MySQL)
ElastiCache
(Redis)
SES
Batch
SNS
Mobile
Redshift
CloudWatch

インフラ構成の特徴

全面的に AWS に依存

・人数も少ないのでフルマネージドな環境が魅力的

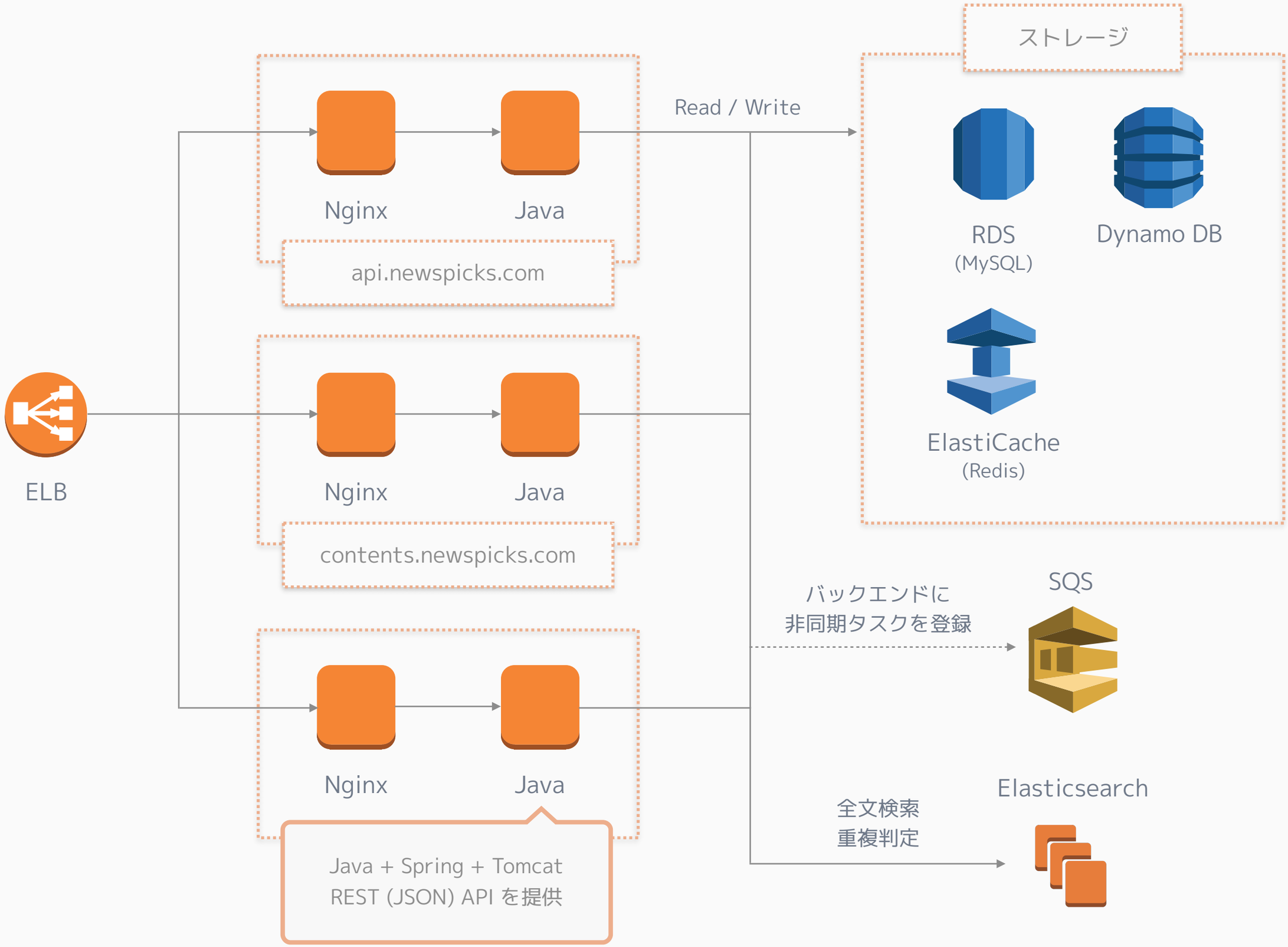
複数のストレージ

・RDS (MySQL) → ユーザーなどのマスタデータなど
・ElastiCache (Redis) → タイムラインやランキングデータなど
・Dynamo DB → 記事やコメントなどのトランザクションデータなど
・Memcached → キャッシュ（一部）
・Elasticsearch → 全文検索・重複記事チェックなど

SQS を介してバックエンドを分散

・突発的な負荷に備えて非同期に処理
・各バックエンドサービスは SQS からメッセージを消費

フロントエンドの構成


ストレージ
Read / Write
Nginx
Java
api.newspicks.com
RDS
(MySQL)
Dynamo DB
ElastiCache
(Redis)
ELB
Nginx
Java
contents.newspicks.com
バックエンドに
非同期タスクを登録
SQS
Nginx
Java
Elasticsearch
全文検索
重複判定
Java + Spring + Tomcat
REST (JSON) API を提供

いまどき Java ですか？

最近の Java はかなりモダンになってきているので問題ない

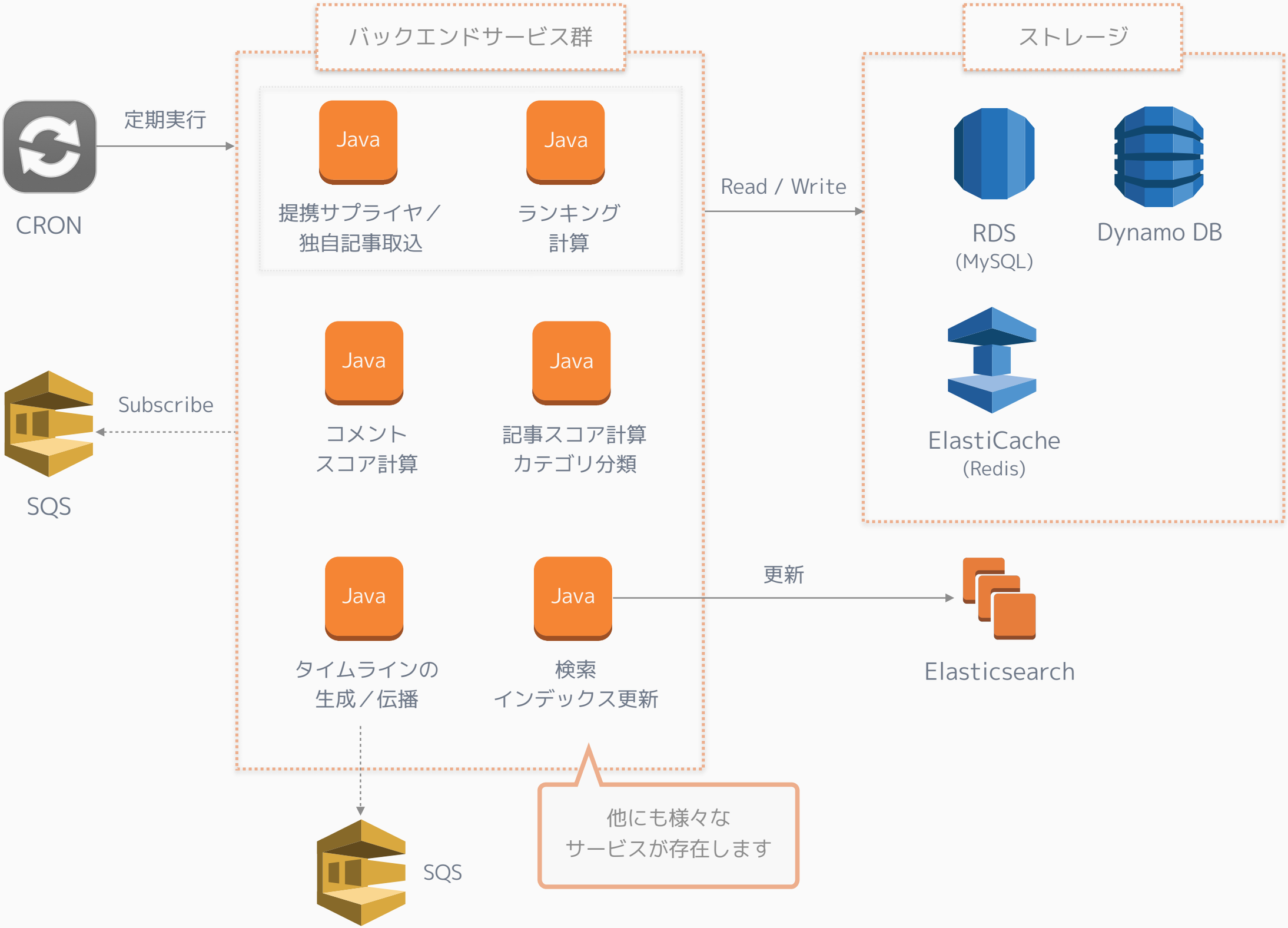
- 静的型付けの安心感
- Java 8 + Lombok の軽快感

```
@Builder
@Value
public class SearchCondition {
    final String query;
    final Integer offset;
    final Integer limit;
    final Order order;
}
```

```
SearchCondition condition = SearchCondition.builder()
    .query(“uzabase”)
    .order(Order.DESC)
    .limit(10)
    .build();

List<Pick> picks = service.findByCondition(condition);
List<String> pickComments = picks.stream()
    .map(Pick::getComment)
    .filter(comment -> !isNullOrEmpty(comment))
    .collect(Collectors.toList());
```

バックエンドの構成


バックエンドサービス群
ストレージ
定期実行
CRON
Java
提携サプライヤ／
独自記事取込
Java
ランキング
計算
Read / Write
RDS
(MySQL)
Dynamo DB
Java
コメント
スコア計算
Java
記事スコア計算
カテゴリ分類
ElastiCache
(Redis)
Subscribe
SQS
Java
タイムラインの
生成／伝播
Java
検索
インデックス更新
更新
Elasticsearch
SQS
他にも様々な
サービスが存在します

例：記事取込時の処理フローのイメージ

Worker が非同期に連携して記事取込 → 分類 → タイムライン伝播
タイムラインは当初 Dynamo DB で実装していたが, パフォーマンスに難があり
Redis を利用して push 型のタイムラインを生成することに …

Article

Categorize Queue

Propagate Queue

Java

Feed Service

Java

Categorize Service

Java

Propagate Service

Dynamo DB

CPP

機械学習
エンジン

Vowpal Wabbit
Scikit Learn etc …

ユーザー毎の
タイムライン

ElastiCache
(Redis)

例：記事取込時の処理フローのイメージ

Worker が非同期に連携して記事取込 → 分類 → タイムライン伝播
タイムラインは当初 Dynamo DB で実装していたが, パフォーマンスに難があり
Redis を利用して push 型のタイムラインを生成することに …

① RSS etc の更新

Article

Categorize Queue

Propagate Queue

Java

Feed Service

Java

Categorize Service

Java

Propagate Service

Dynamo DB

CPP

機械学習
エンジン

Vowpal Wabbit
Scikit Learn etc …

ユーザー毎の
タイムライン

ElastiCache
(Redis)

例：記事取込時の処理フローのイメージ

Worker が非同期に連携して記事取込 → 分類 → タイムライン伝播
タイムラインは当初 Dynamo DB で実装していたが, パフォーマンスに難があり
Redis を利用して push 型のタイムラインを生成することに …

① RSS etc の更新

Article

② Poll

Categorize Queue

Propagate Queue

Java

Feed Service

Java

Categorize Service

Java

Propagate Service

Dynamo DB

CPP

機械学習
エンジン

Vowpal Wabbit
Scikit Learn etc …

ユーザー毎の
タイムライン

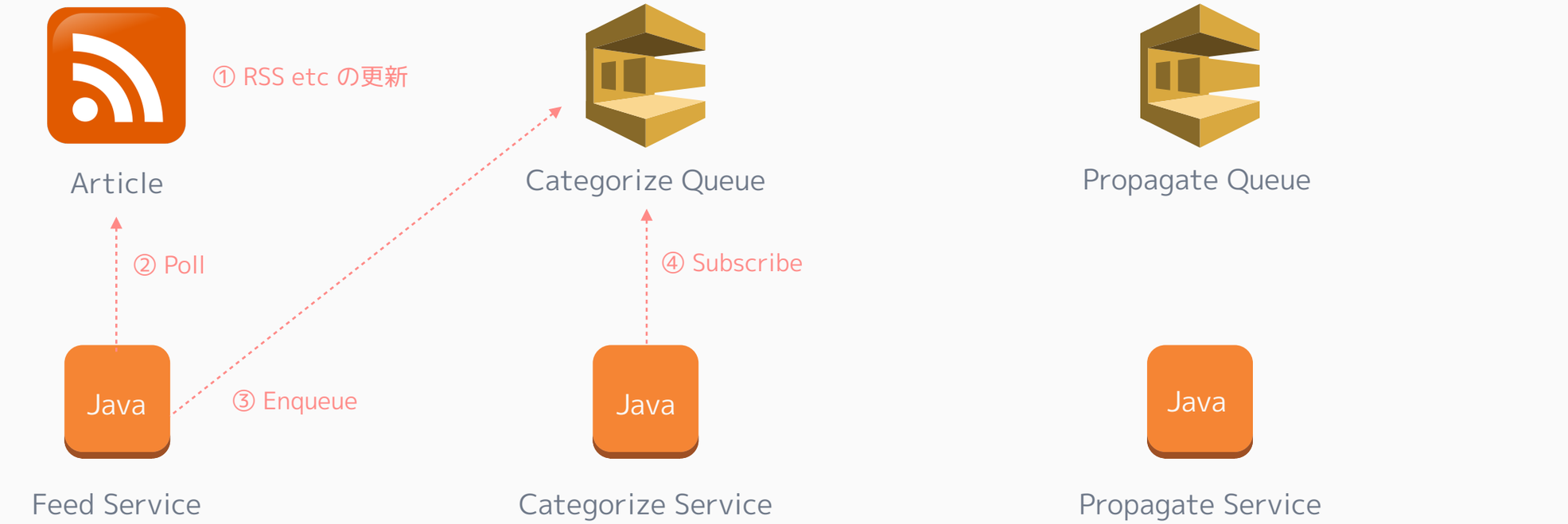
ElastiCache
(Redis)

例：記事取込時の処理フローのイメージ

Worker が非同期に連携して記事取込 → 分類 → タイムライン伝播
タイムラインは当初 Dynamo DB で実装していたが, パフォーマンスに難があり
Redis を利用して push 型のタイムラインを生成することに …

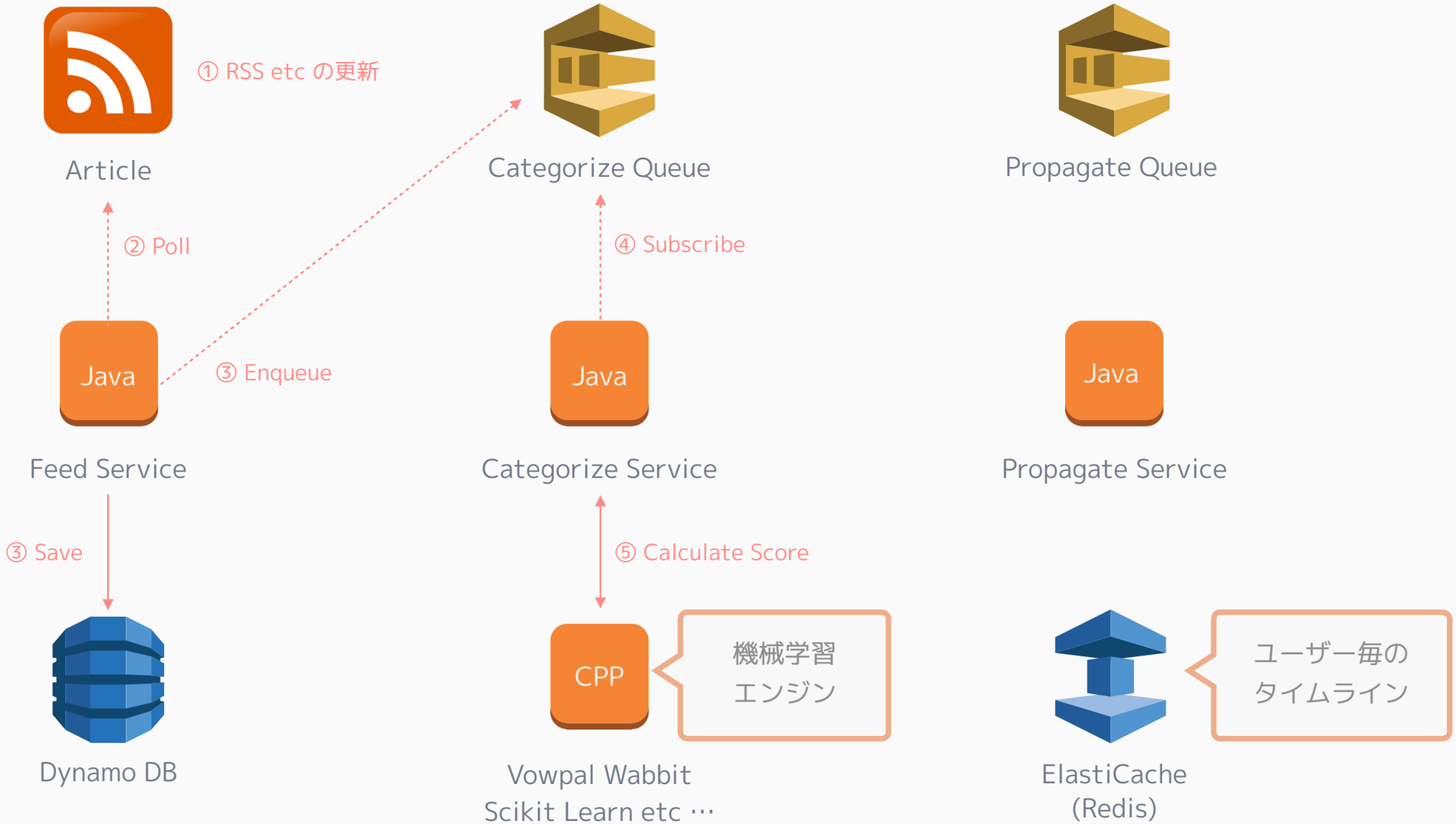
例：記事取込時の処理フローのイメージ

Worker が非同期に連携して記事取込 → 分類 → タイムライン伝播
タイムラインは当初 Dynamo DB で実装していたが, パフォーマンスに難があり
Redis を利用して push 型のタイムラインを生成することに …



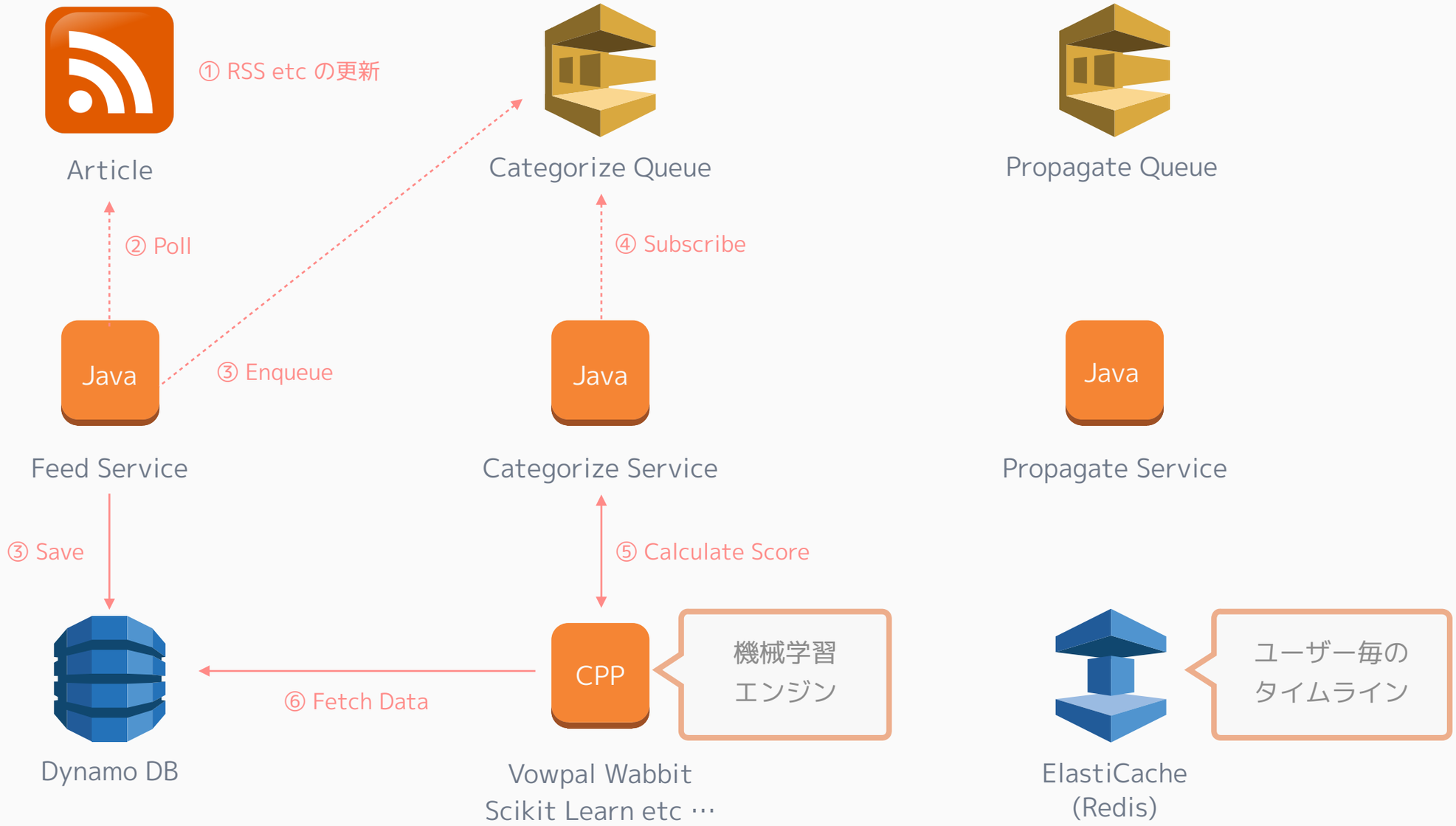
例：記事取込時の処理フローのイメージ

Worker が非同期に連携して記事取込 → 分類 → タイムライン伝播
タイムラインは当初 Dynamo DB で実装していたが, パフォーマンスに難があり
Redis を利用して push 型のタイムラインを生成することに …



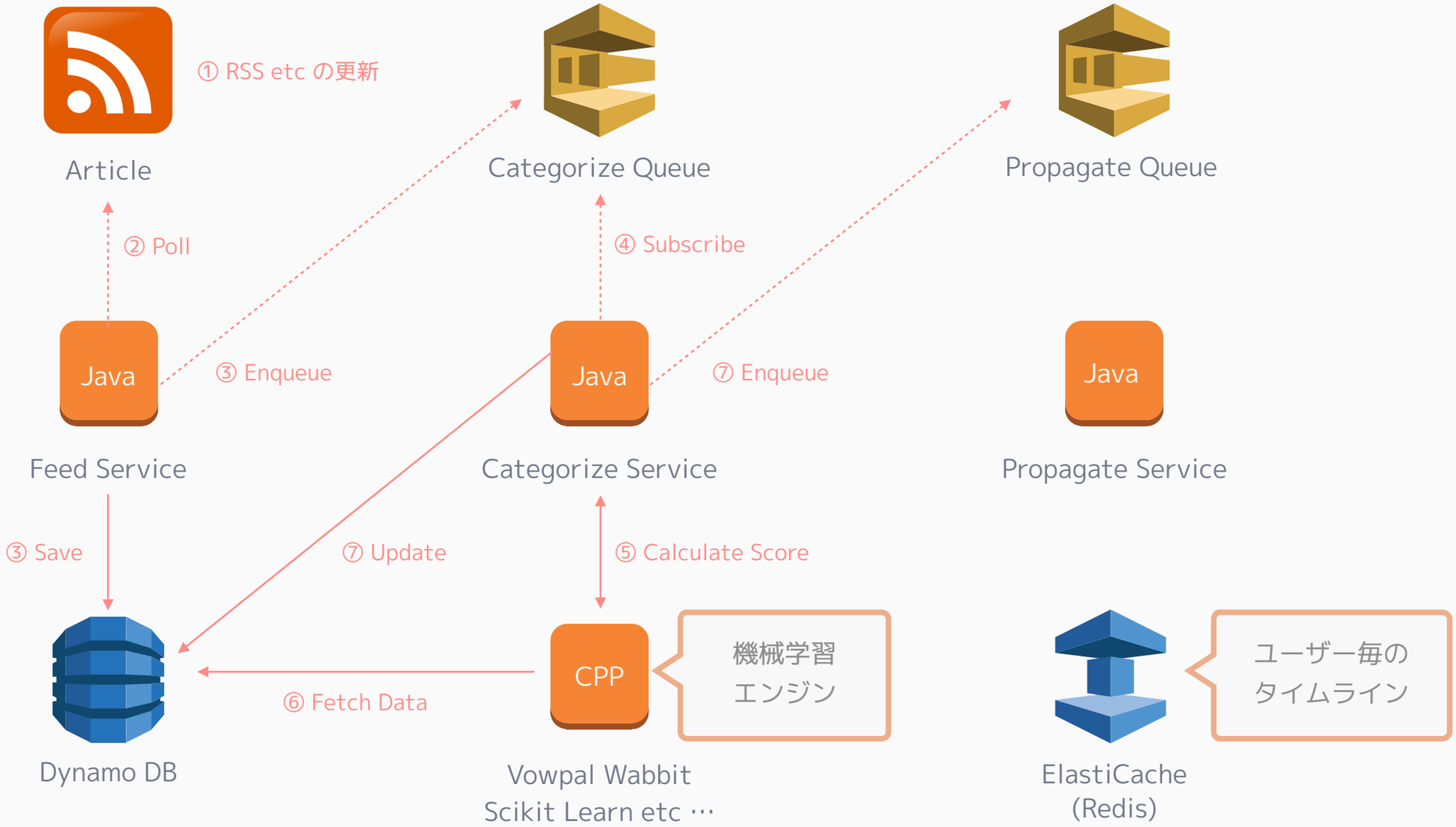
例：記事取込時の処理フローのイメージ

Worker が非同期に連携して記事取込 → 分類 → タイムライン伝播
タイムラインは当初 Dynamo DB で実装していたが, パフォーマンスに難があり
Redis を利用して push 型のタイムラインを生成することに …



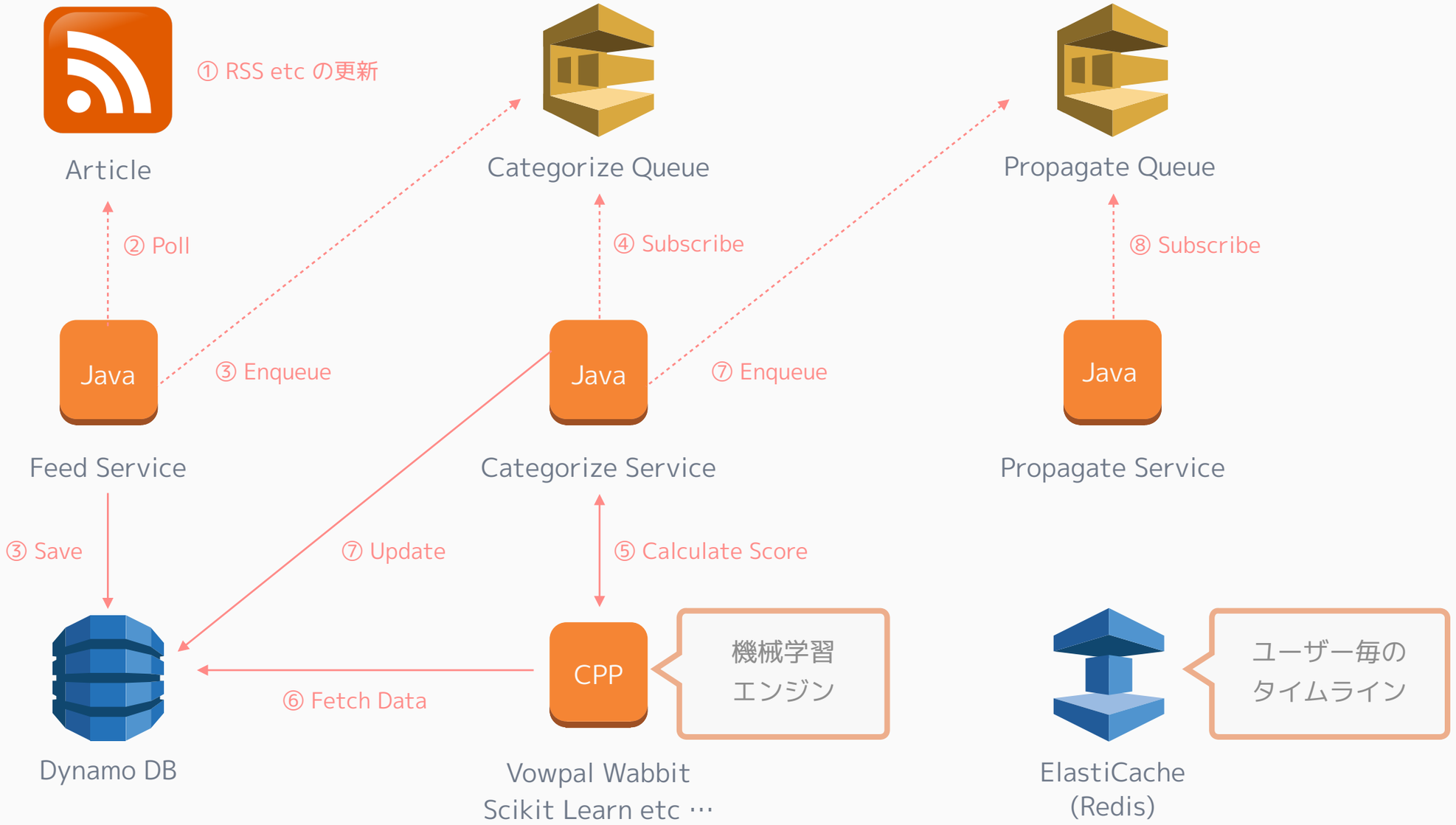
例：記事取込時の処理フローのイメージ

Worker が非同期に連携して記事取込 → 分類 → タイムライン伝播
タイムラインは当初 Dynamo DB で実装していたが, パフォーマンスに難があり
Redis を利用して push 型のタイムラインを生成することに …



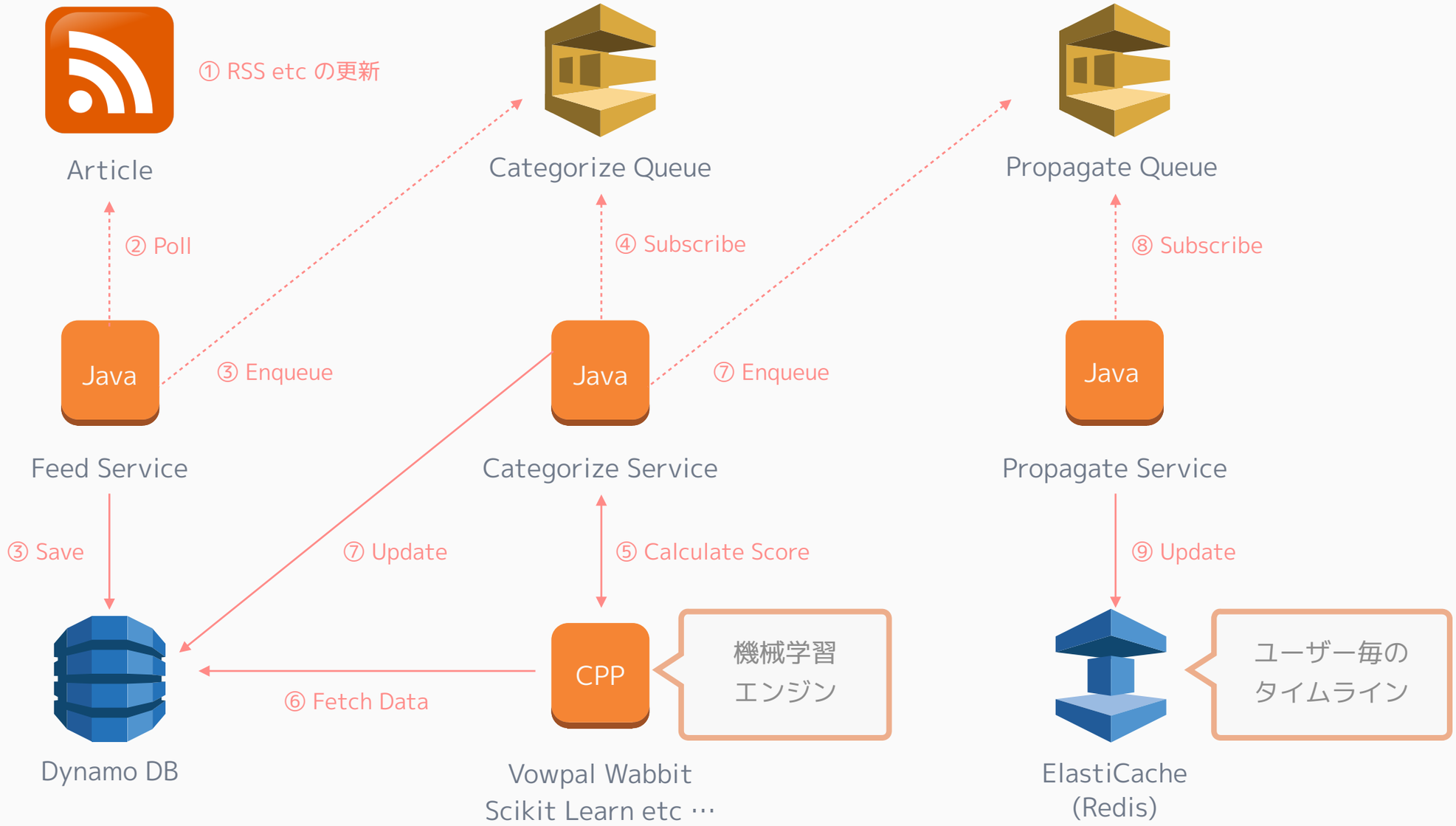
例：記事取込時の処理フローのイメージ

Worker が非同期に連携して記事取込 → 分類 → タイムライン伝播
タイムラインは当初 Dynamo DB で実装していたが, パフォーマンスに難があり
Redis を利用して push 型のタイムラインを生成することに …



例：記事取込時の処理フローのイメージ

Worker が非同期に連携して記事取込 → 分類 → タイムライン伝播
タイムラインは当初 Dynamo DB で実装していたが, パフォーマンスに難があり
Redis を利用して push 型のタイムラインを生成することに …



SQS (Amazon Simple Queue Service)

利用目的

・コストのかかる計算処理を分散 → ピーク時のスループット向上
・機械学習エンジンなどのバックエンドを分離 → 独立したサービスを育てる

特徴とメリット

・可用性・拡張性が担保された分散キュー
・ふつう MQ を自前で運用しようと思うと結構大変だけど何も考えなくて良い

注意点

・キューの処理順は担保されていない
・複数回同一のメッセージを Receive することがある

→ SQS を利用するバックエンドサービスは冪等に実装すること！

アジェンダ

自己紹介

サービスの概要

NewsPicks を支える組織

NewsPicks を支える技術

本当にあった怖い話

まとめ

((((;°д°))))

3 ヶ月の振り返りも兼ねて
本当にあった怖い話をします

荒ぶる Redis - 迫る X デー

問題

・オンラインでのタイムライン書き込みが遅延 → タイムラインが更新されない
・夜間バッチでの古いタイムラインの切り詰め処理が遅延 → 深夜アラート

→ このままユーザーが増えたら死んでしまう … 迫る X デーに震える日々

荒ぶる Redis - 迫る X デー

問題

・オンラインでのタイムライン書き込みが遅延 → タイムラインが更新されない
・夜間バッチでの古いタイムラインの切り詰め処理が遅延 → 深夜アラート

→ このままユーザーが増えたら死んでしまう … 迫る X デーに震える日々

原因

・push 型のタイムラインを形成しているため, 大量更新が発生
・もともとタイムライン用の Redis は 1 台
・Redis はイベントループモデルなので 1 コアで処理する → CPU 使用率高騰

荒ぶる Redis - 迫る X デー

問題

・オンラインでのタイムライン書き込みが遅延 → タイムラインが更新されない
・夜間バッチでの古いタイムラインの切り詰め処理が遅延 → 深夜アラート

→ このままユーザーが増えたら死んでしまう … 迫る X デーに震える日々

原因

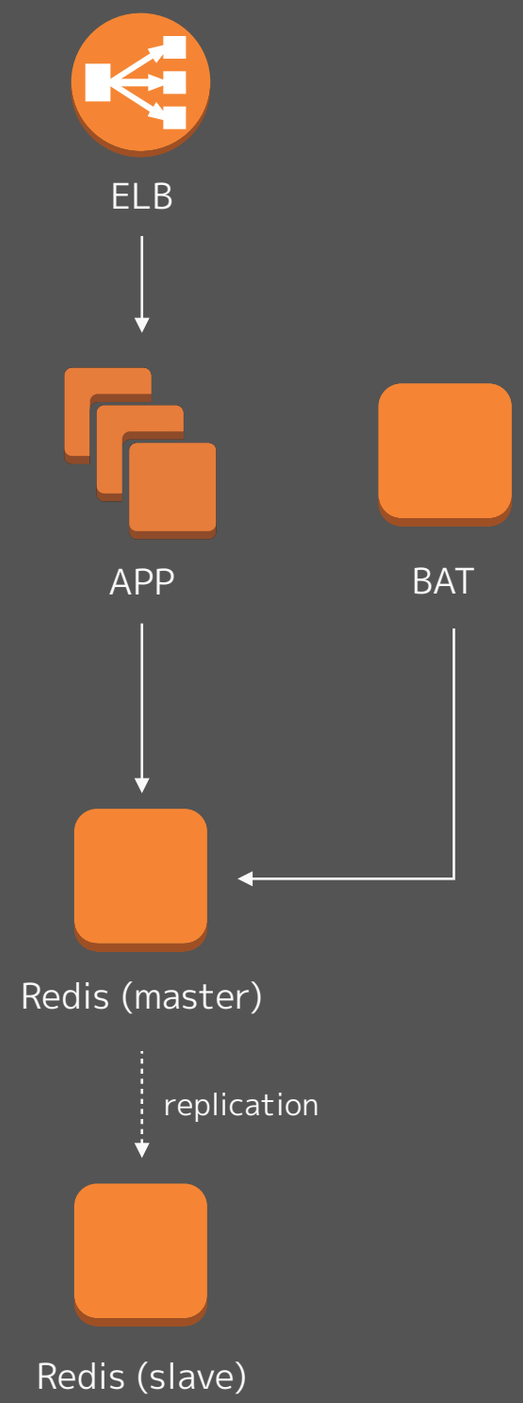
・push 型のタイムラインを形成しているため, 大量更新が発生
・もともとタイムライン用の Redis は 1 台
・Redis はイベントループモデルなので 1 コアで処理する → CPU 使用率高騰

解決

・ElastiCache に移行し, SPOF となっていた Redis の台数を増やす
・ユーザーパーティショニングによる垂直分散
・タイムラインを更新するバックエンドサービスをスケールアウト

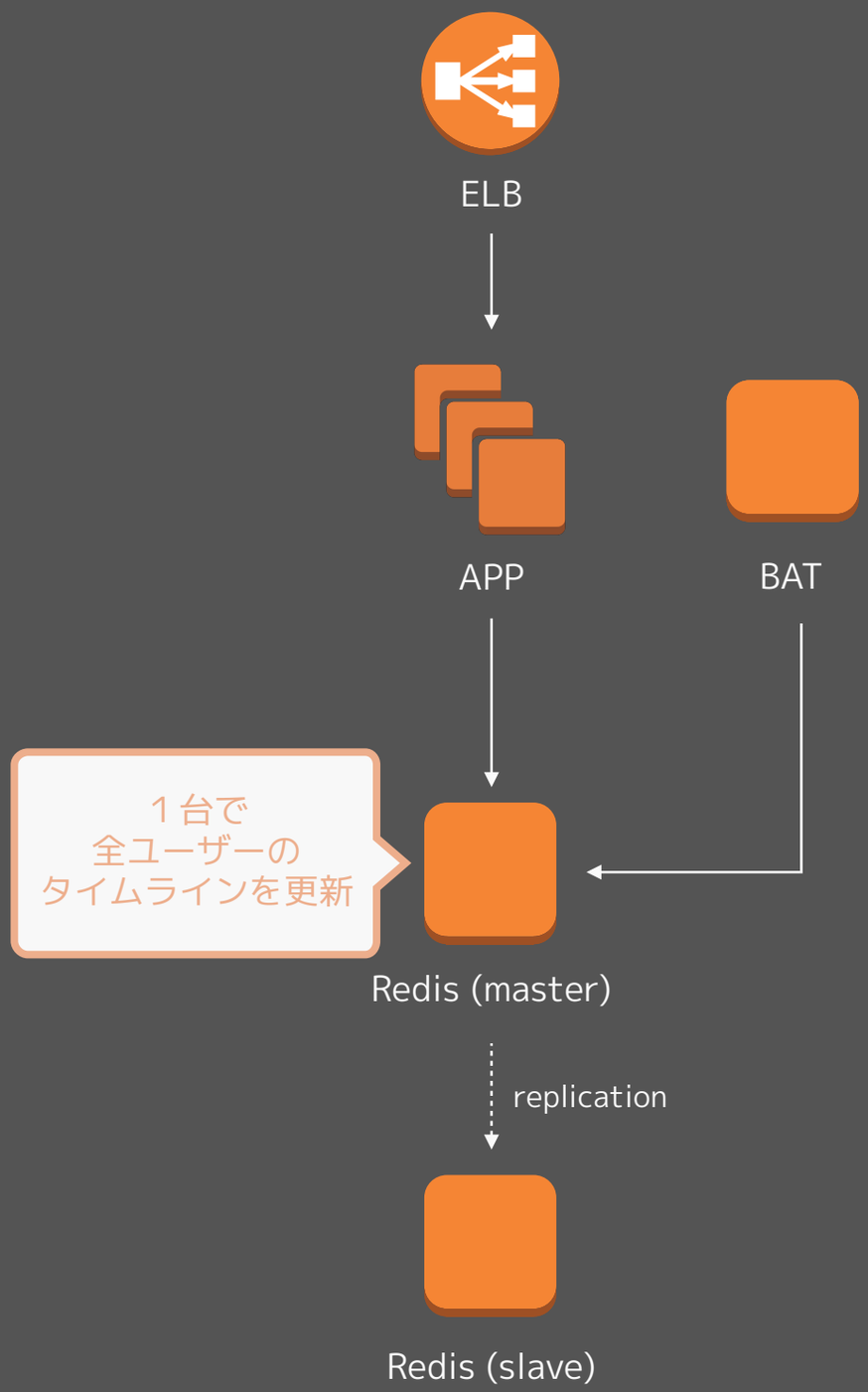
Redis - タイムラインの垂直分散

BEFORE



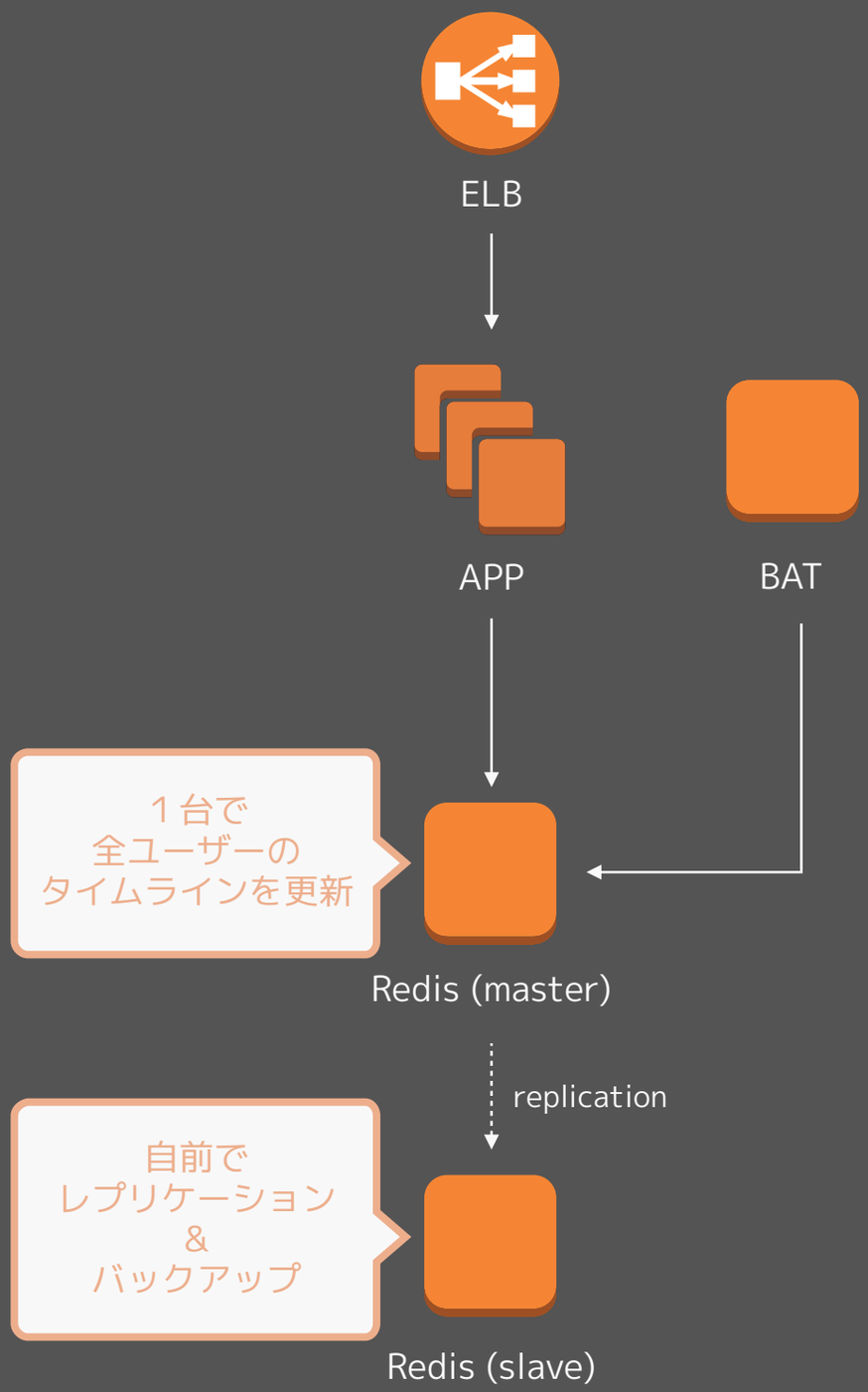
Redis - タイムラインの垂直分散

BEFORE



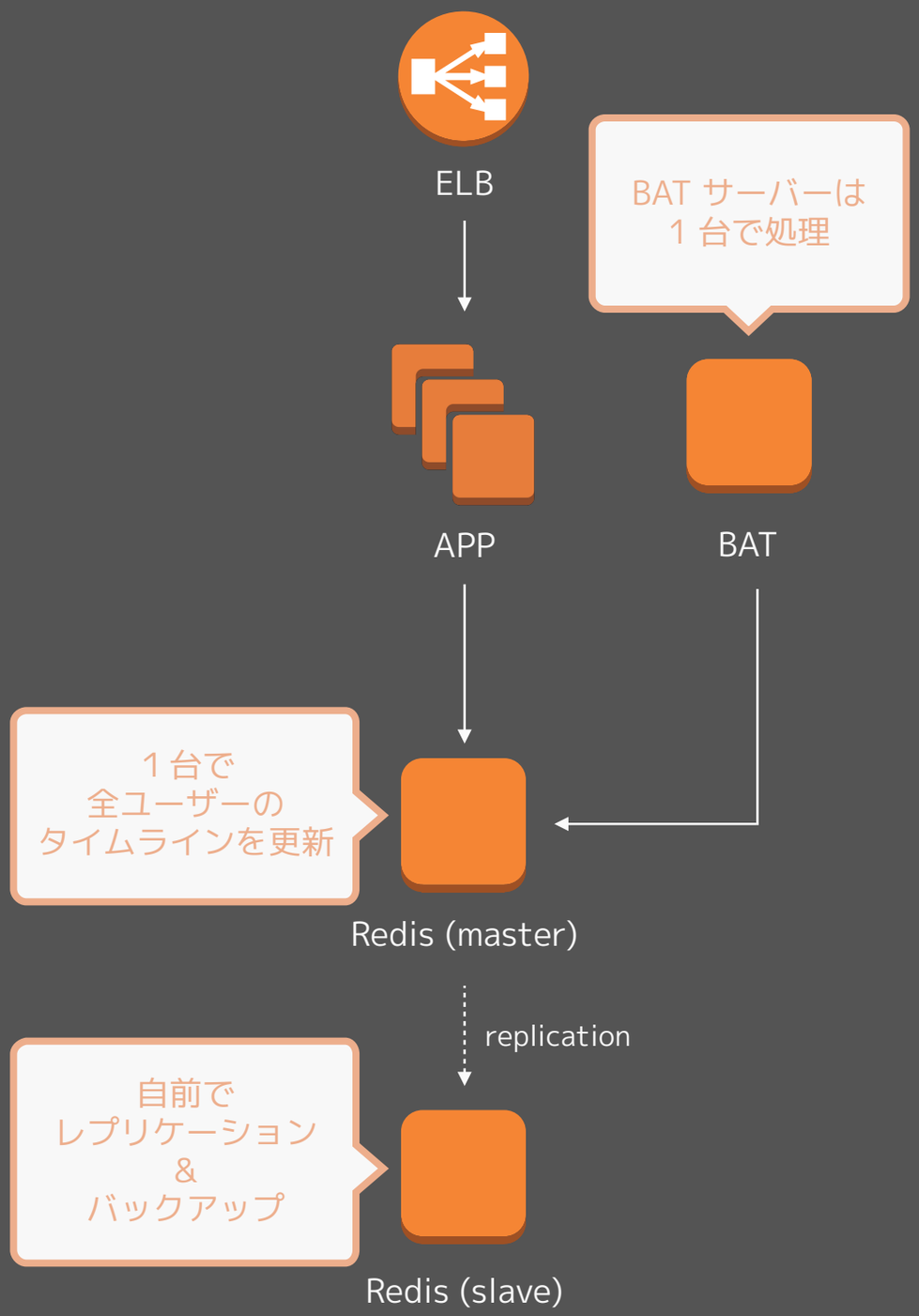
Redis - タイムラインの垂直分散

BEFORE



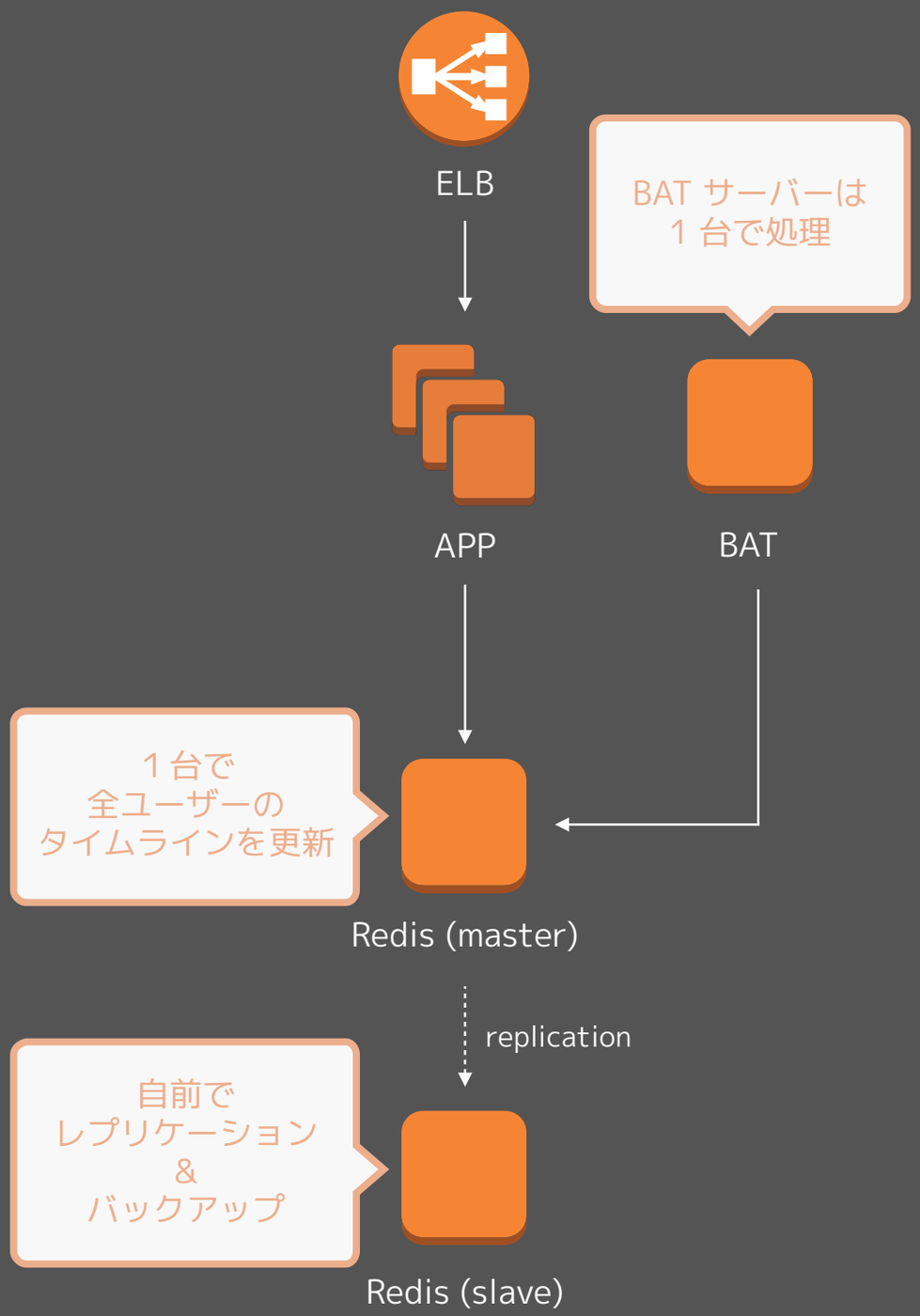
Redis - タイムラインの垂直分散

BEFORE

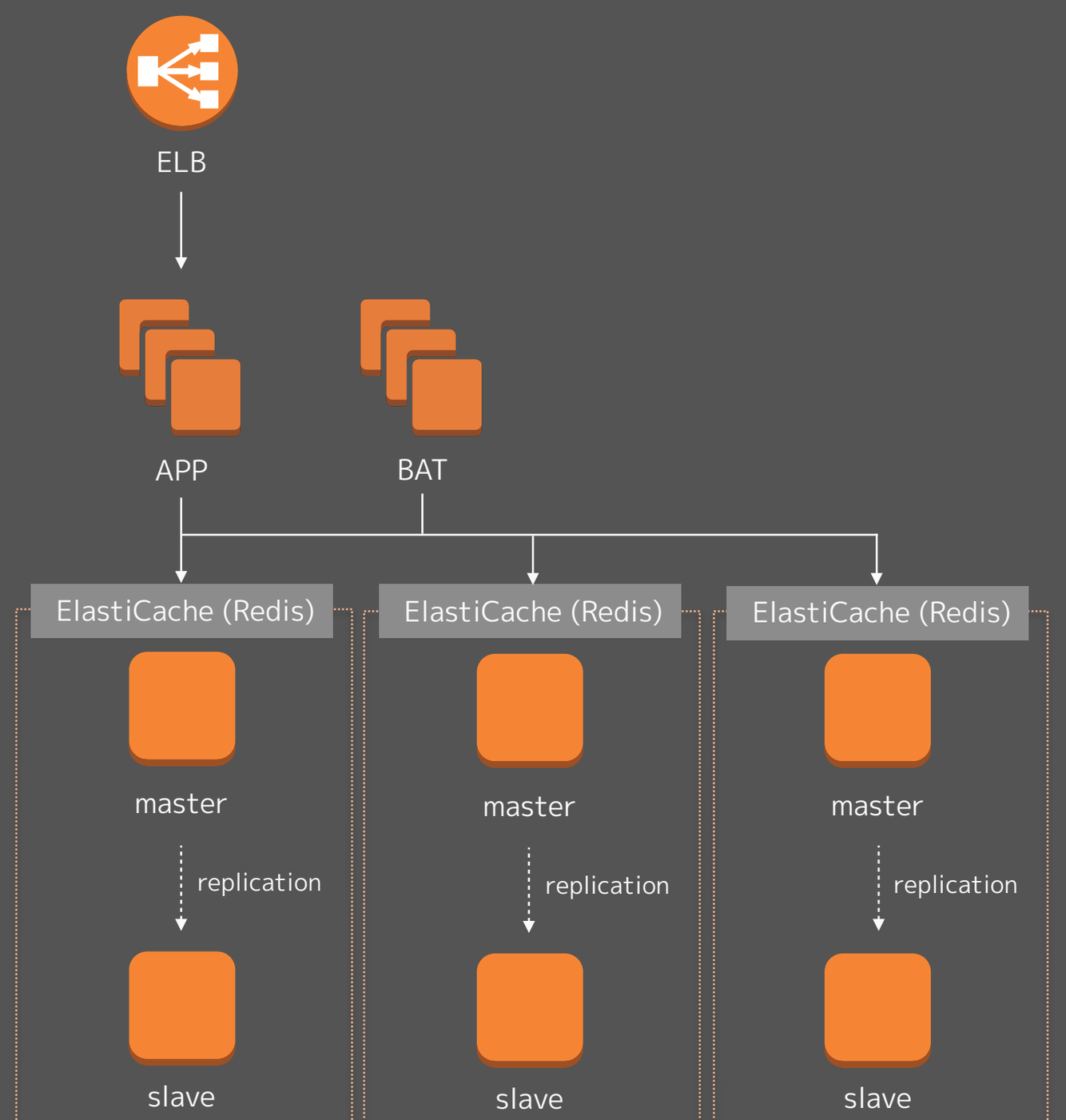


Redis - タイムラインの垂直分散

BEFORE

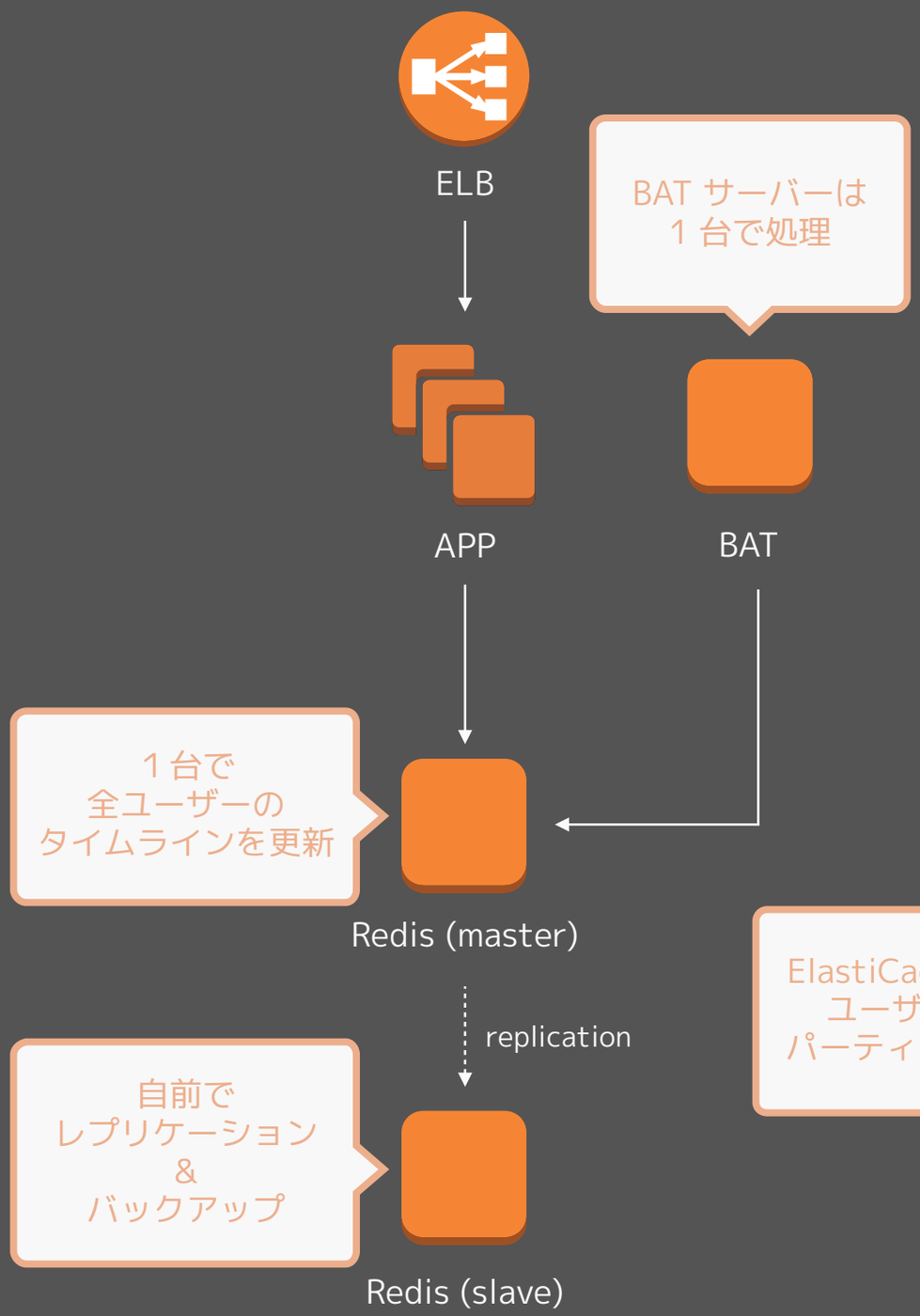


AFTER

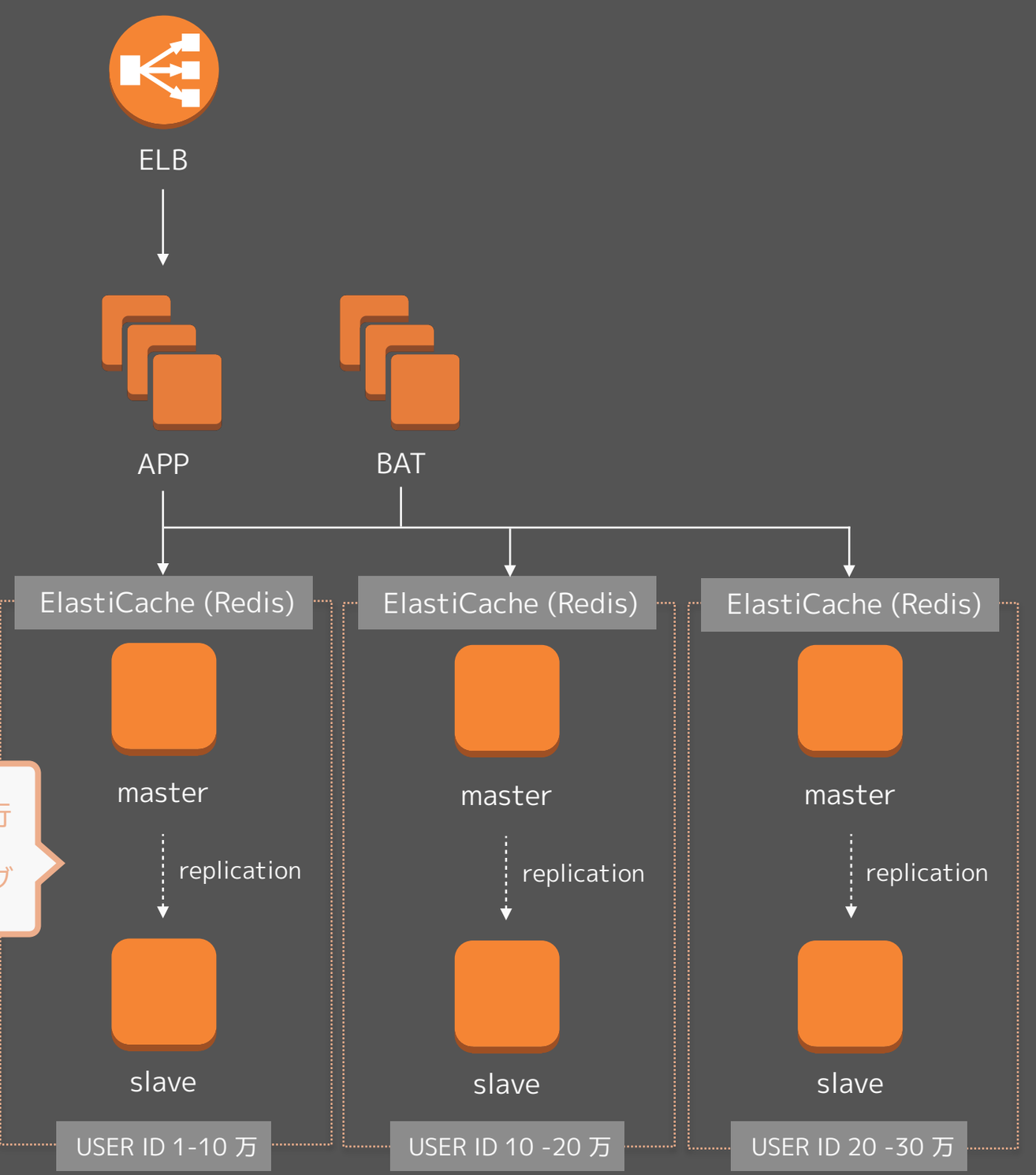


Redis - タイムラインの垂直分散

BEFORE

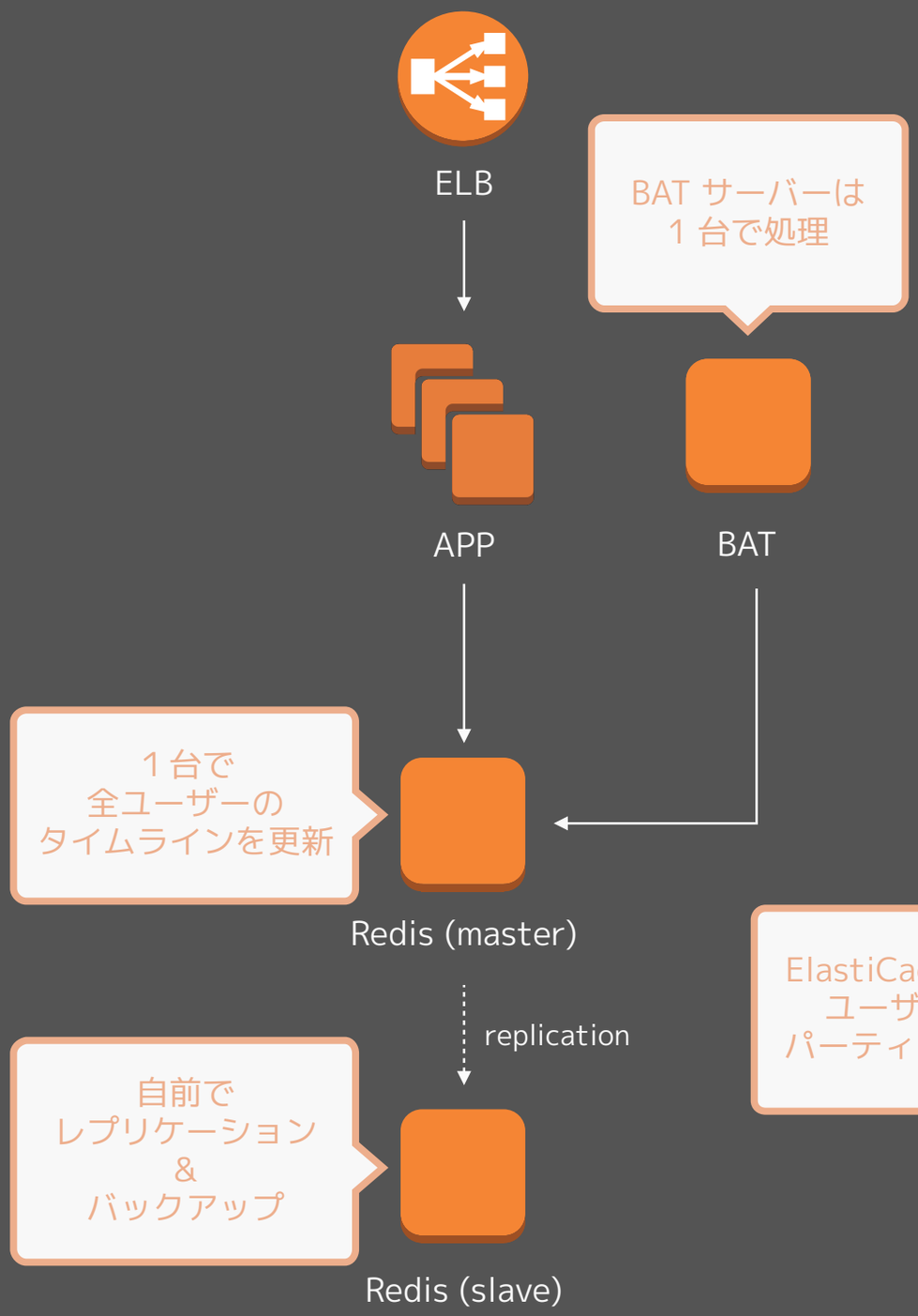


AFTER

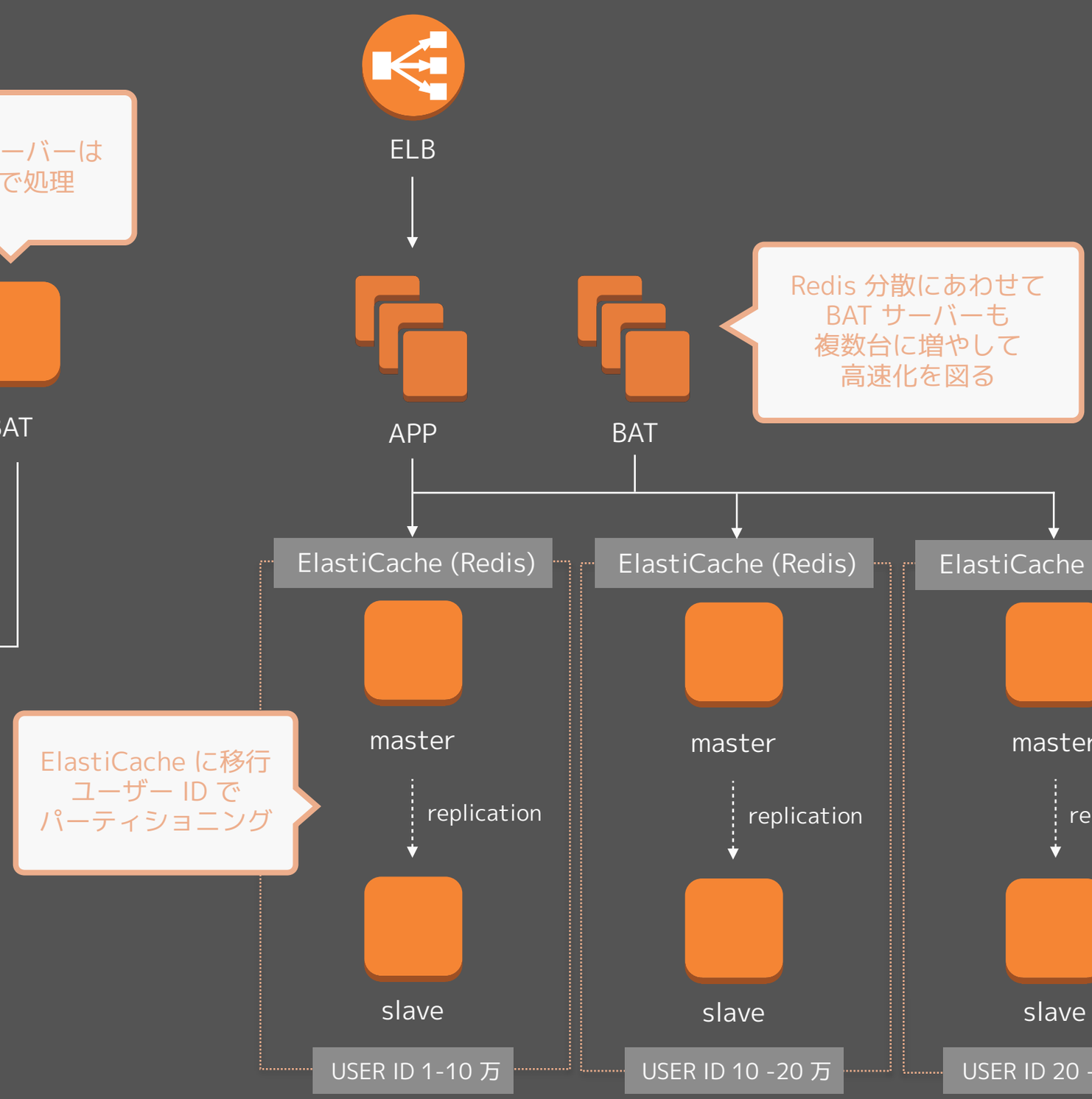


Redis - タイムラインの垂直分散

BEFORE



AFTER



荒ぶる Redis - 次なる X デーに備えて

問題

・タイムライン以外の Redis は 1 台（SPOF）
・ピーク時に Read / Write が 1 台に集中して遅延

→ 募る不安 … 次の X デーパーティーの会場はココですか？

荒ぶる Redis - 次なる X デーに備えて

問題

・タイムライン以外の Redis は 1 台（SPOF）
・ピーク時に Read / Write が 1 台に集中して遅延

→ 募る不安 … 次の X デーパーティーの会場はココですか？

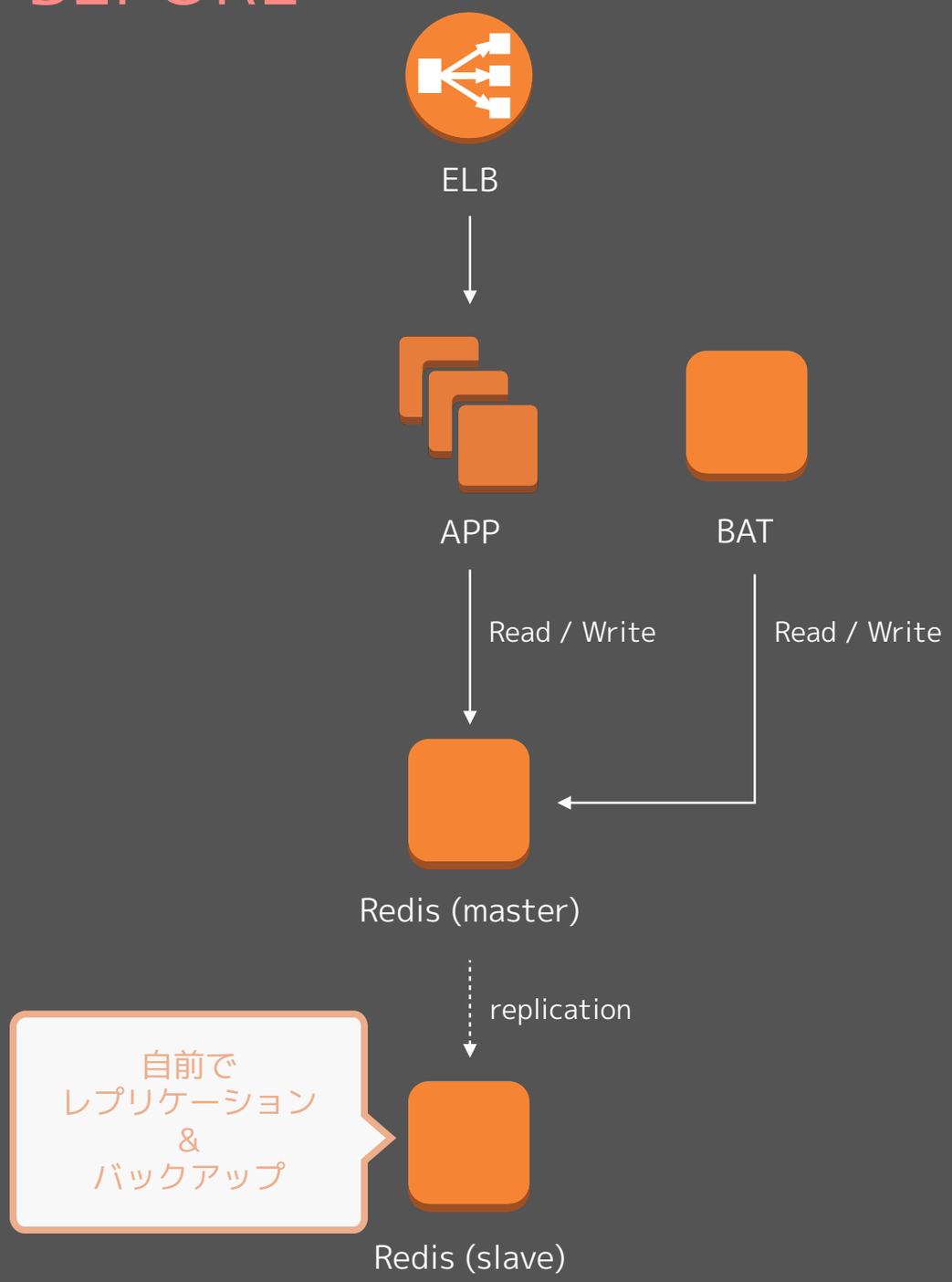
解決

・ElastiCache に移行し, SPOF となっていた Redis の台数を増やす
・読み込み処理はリードレプリカに接続して負荷分散
・アプリ側で問題が起きないような仕組みを導入

→ レプリタイミングによって先祖返りしそうなのでスティッキーに
→ リードレプリカが落ちた場合に備えてフェイルオーバーするように

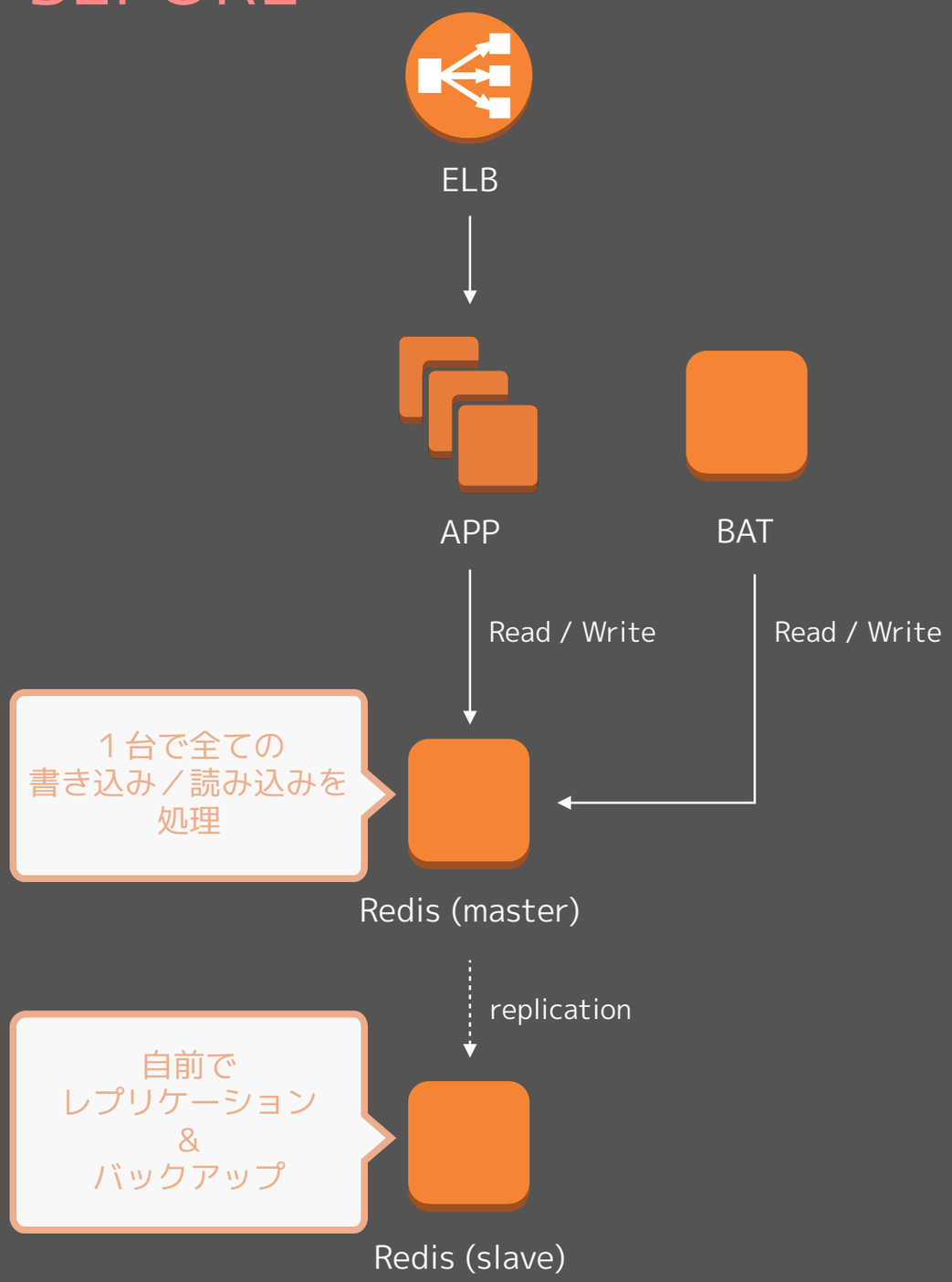
Redis - Read / Write 水平分散

BEFORE



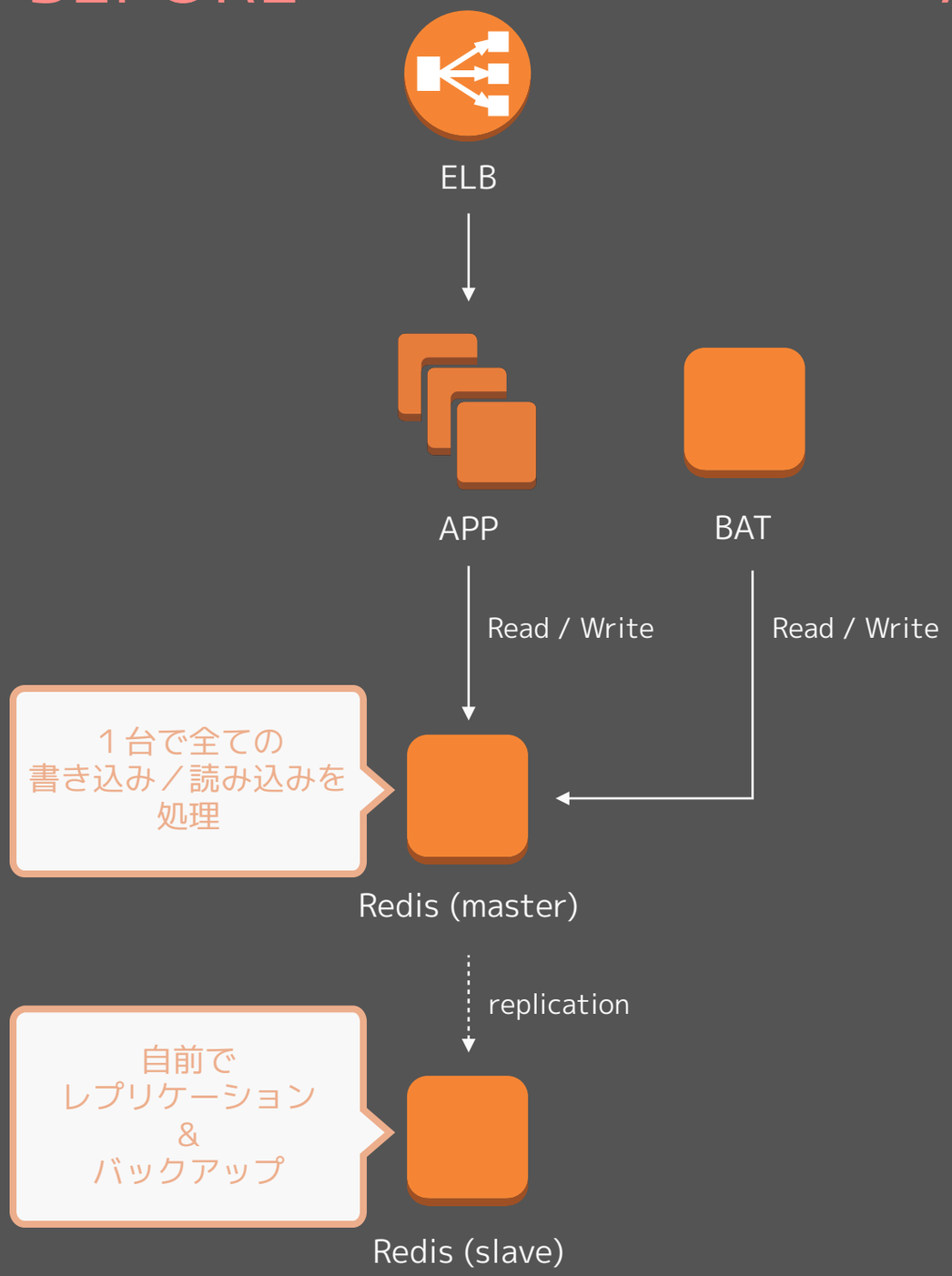
Redis - Read / Write 水平分散

BEFORE

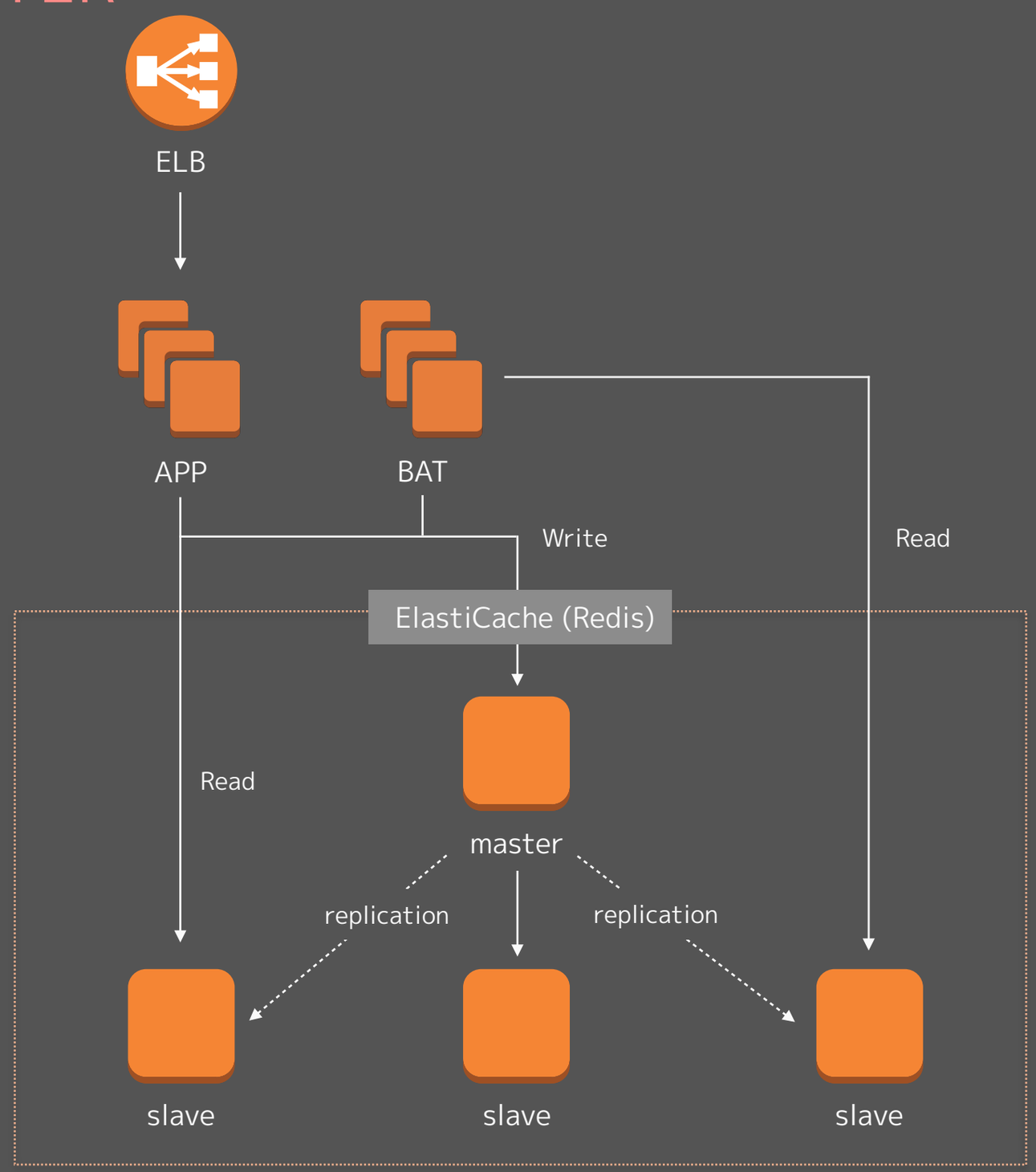


Redis - Read / Write 水平分散

BEFORE

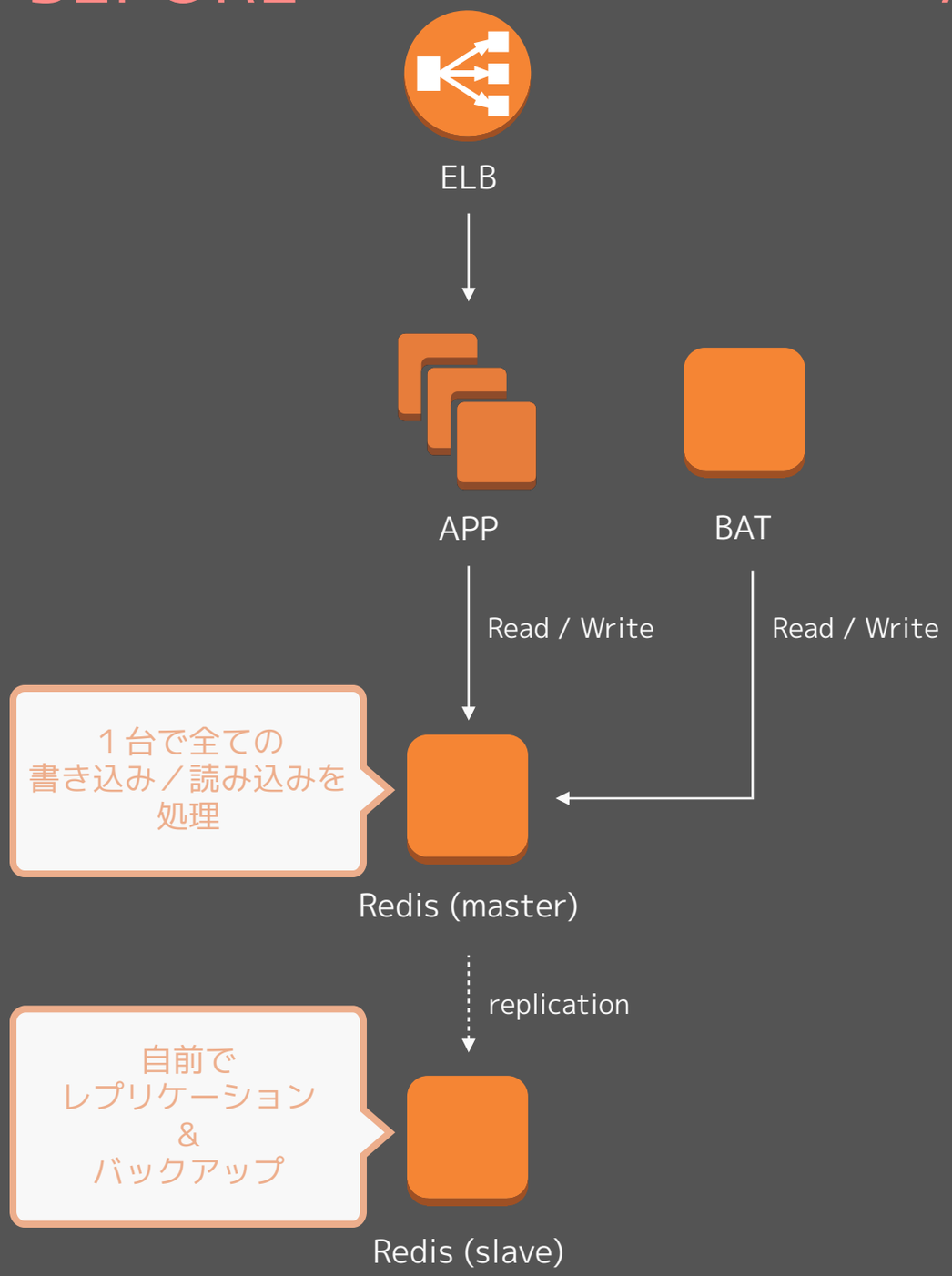


AFTER

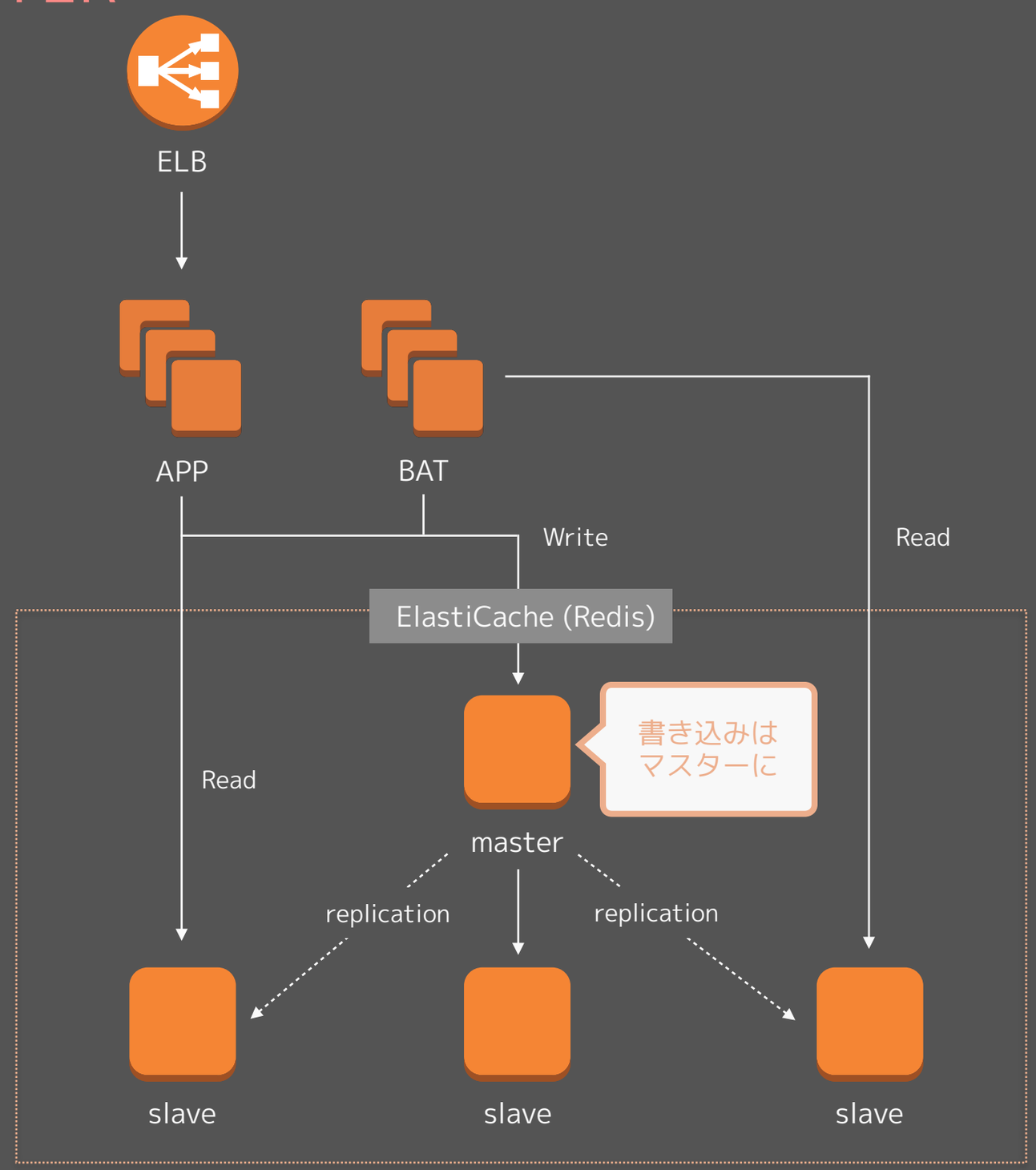


Redis - Read / Write 水平分散

BEFORE

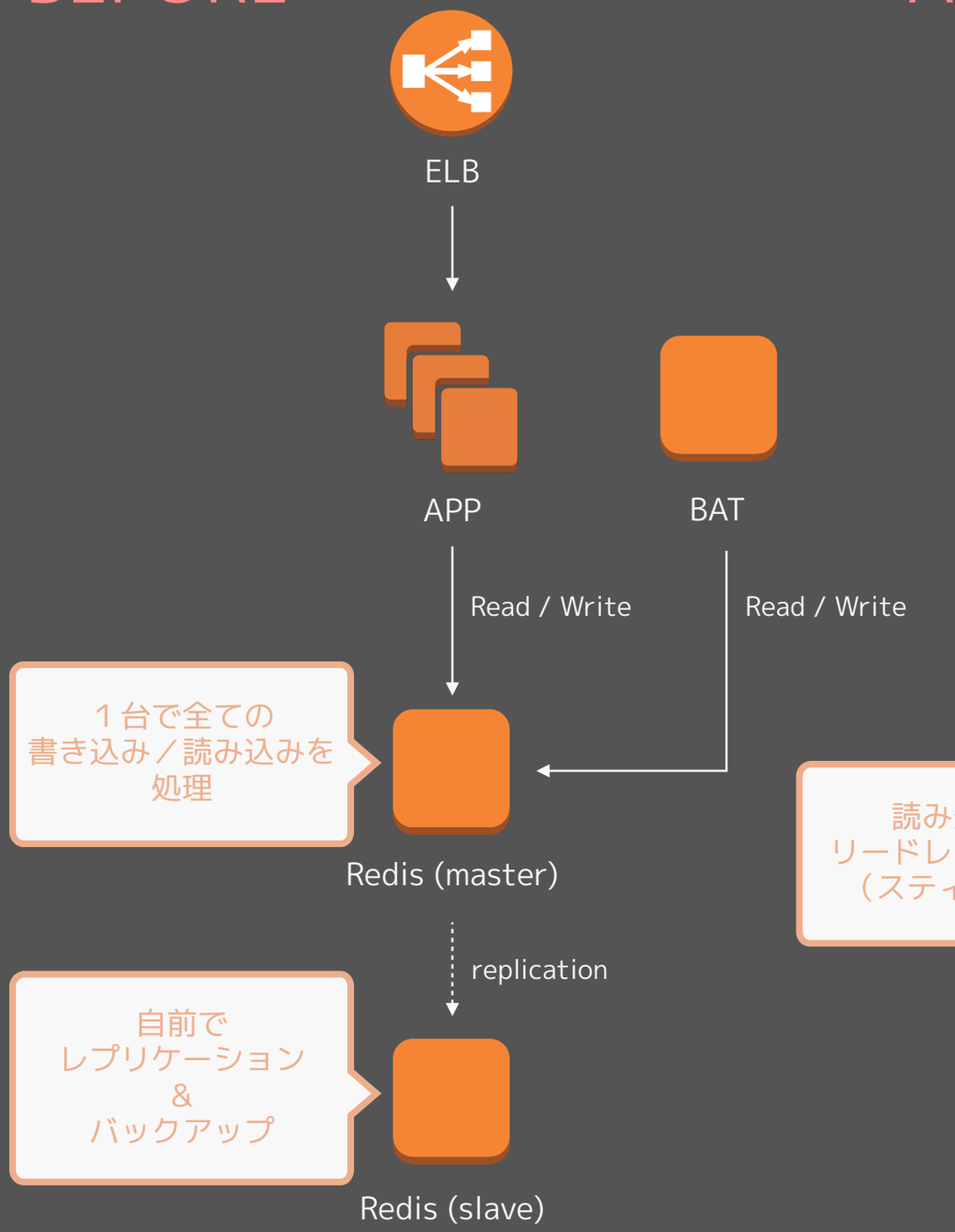


AFTER

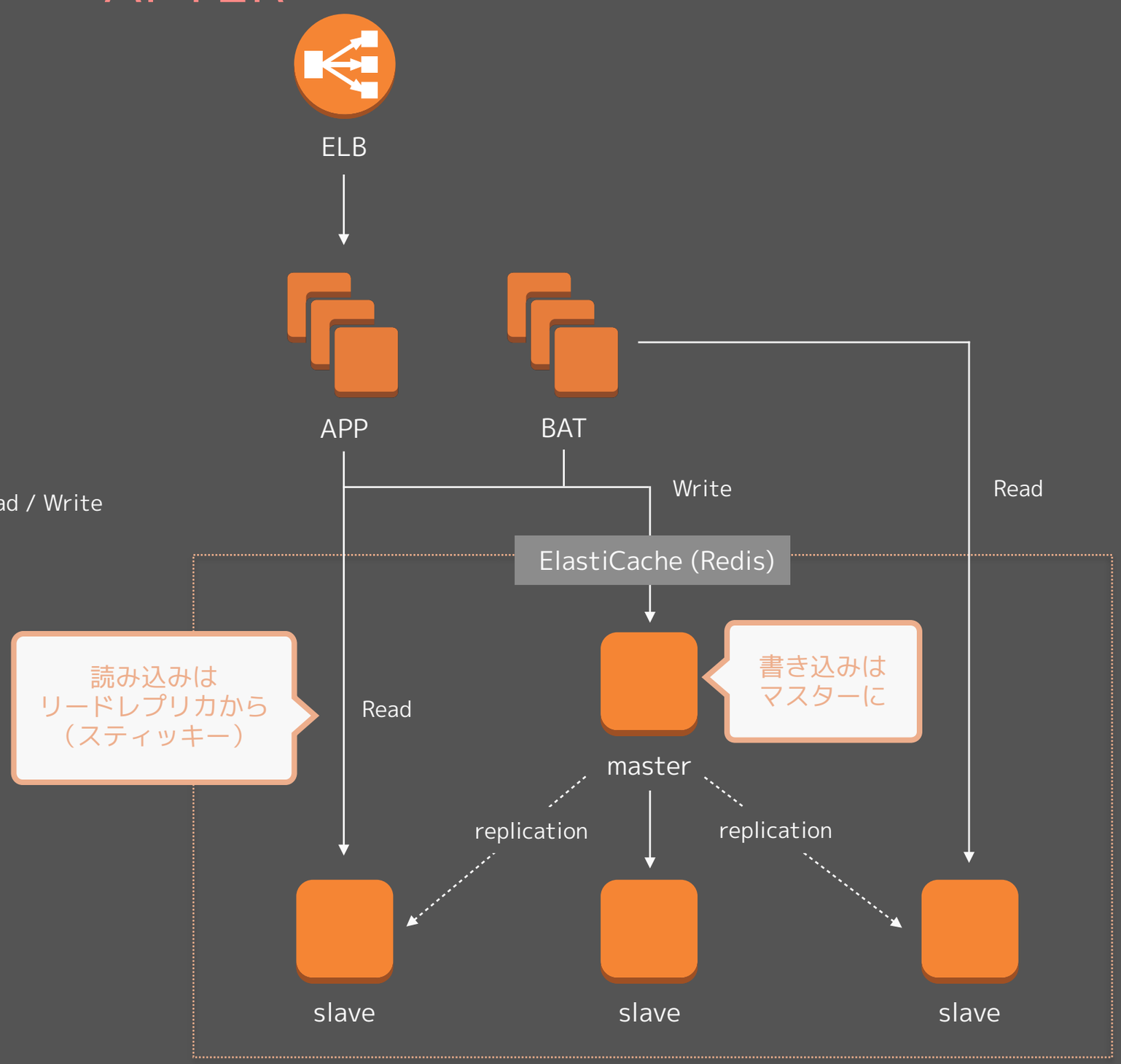


Redis - Read / Write 水平分散

BEFORE

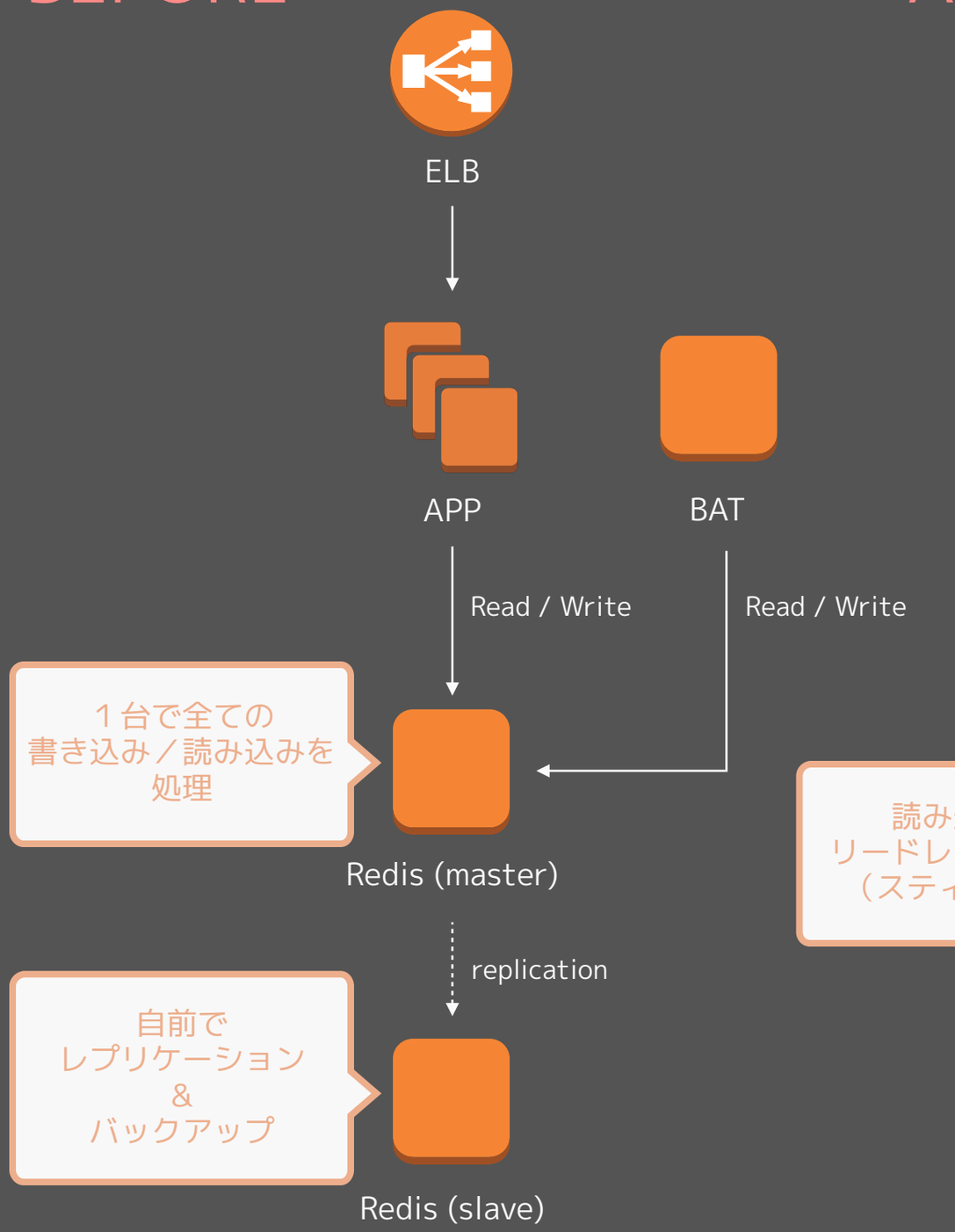


AFTER

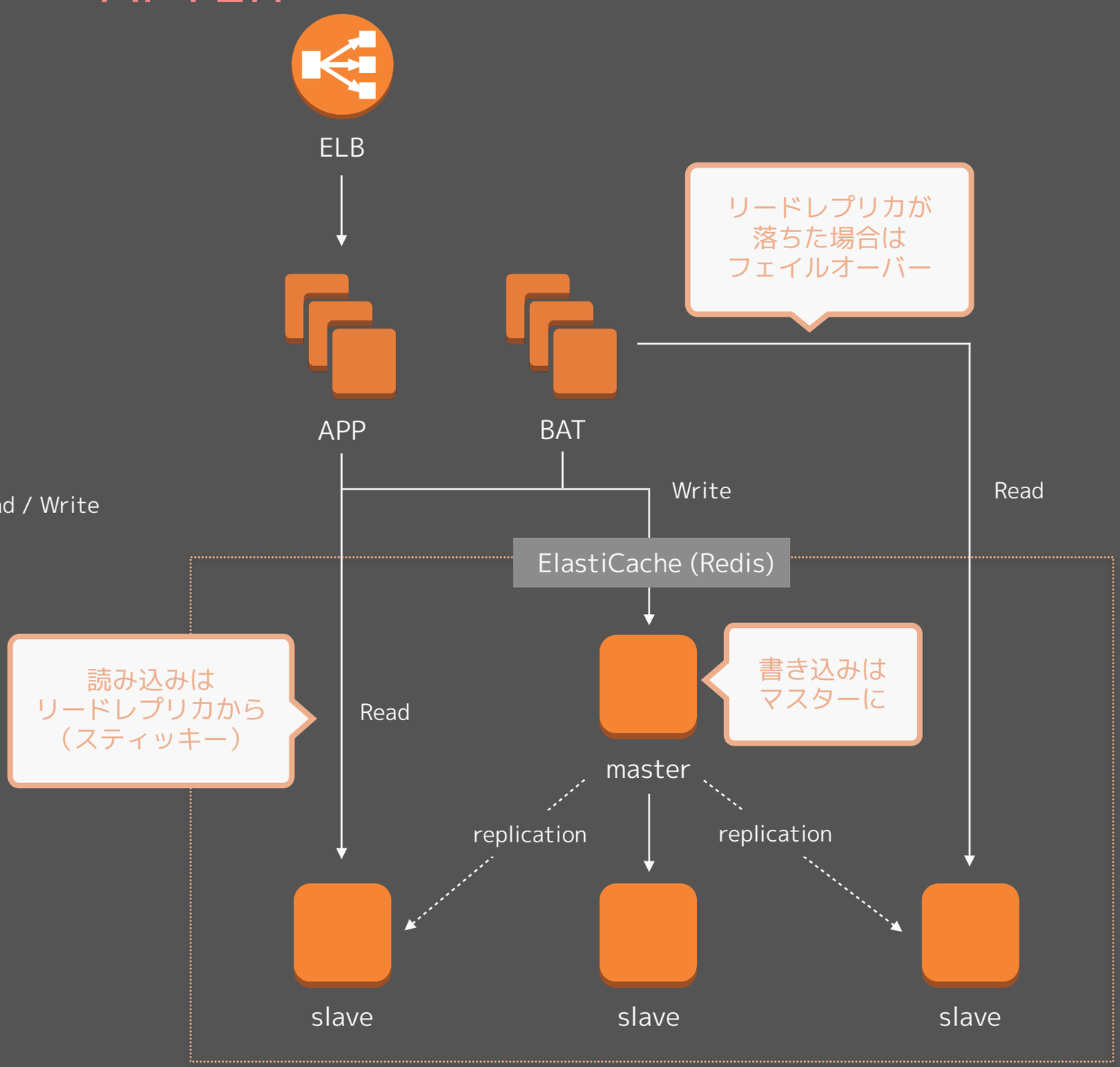


Redis - Read / Write 水平分散

BEFORE



AFTER



荒ぶる Phantom - 無数のクロールエラー

問題

・Phantom JS が暴走して大量のクロールエラーが発生
・正しくインデックスが作成されず, 検索からユーザーが流入しない

→ 編集部が良い記事を書いても検索から流入しない
→ 新規ユーザーを獲得出来ない

荒ぶる Phantom - 無数のクロールエラー

問題

・Phantom JS が暴走して大量のクロールエラーが発生
・正しくインデックスが作成されず, 検索からユーザーが流入しない

→ 編集部が良い記事を書いても検索から流入しない
→ 新規ユーザーを獲得出来ない

原因

・夏に Web 版をリリースしたが, Angular なので SEO 対策されていなかった
・node + node-phantom でスナップショットを返すようにしたら暴走した

荒ぶる Phantom - 無数のクロールエラー

問題

・Phantom JS が暴走して大量のクロールエラーが発生
・正しくインデックスが作成されず, 検索からユーザーが流入しない

→ 編集部が良い記事を書いても検索から流入しない
→ 新規ユーザーを獲得出来ない

原因

・夏に Web 版をリリースしたが, Angular なので SEO 対策されていなかった
・node + node-phantom でスナップショットを返すようにしたら暴走した

解決

・node-phantom の使用を止めた
・child_process で直接 Phantom を呼び出すようにした

荒ぶる Phantom - 無数のクロールエラー

BEFORE

荒ぶる Phantom - 無数のクロールエラー

BEFORE

激増するクロールエラー

AFTER

クロールエラーが減り始める

互換性肥満 - 何でもアリのモデル

問題

・何のために使用されているのか分からない謎のプロパティが大量に存在
・同じプロパティでも経路によって全く異なる値が設定される

→ 修正するのが怖い／すぐデグレする → 開発スピードの低下

互換性肥満 - 何でもアリのモデル

問題

・何のために使用されているのか分からない謎のプロパティが大量に存在
・同じプロパティでも経路によって全く異なる値が設定される

→ 修正するのが怖い／すぐデグレする → 開発スピードの低下

原因

・API の後方互換性を保つために同じモデルがひたすら拡張されていた
・全く異なる API でも同じモデルが使い回されていた
・業務レイヤのモデルが API の I/O と共有されていた

互換性肥満 - 何でもアリのモデル

問題

・何のために使用されているのか分からない謎のプロパティが大量に存在
・同じプロパティでも経路によって全く異なる値が設定される

→ 修正するのが怖い／すぐデグレする → 開発スピードの低下

原因

・API の後方互換性を保つために同じモデルがひたすら拡張されていた
・全く異なる API でも同じモデルが使い回されていた
・業務レイヤのモデルが API の I/O と共有されていた

解決

・業務レイヤと Web レイヤをきちんと分離
・カジュアルに API のバージョンアップが出来る仕組みをつくる

カジュアルな API のバージョンアップ

```
@Logging
@Controller
@RequiredArgsConstructor(onConstructor = @_(@Inject))
@RequestMapping("/news")
public class NewsController extends ControllerBase {

    private final NewsFacade facade;

    @RequestMapping(value = "/{id}/picks", method = GET, produces = ContentTypes.JSON, headers = Headers.API_VERSION_2)
    @ResponseBody
    public PageableCollectionDto<PickViewDtoV2> getPicks(
        @PathVariable Long id, @ModelAttribute PickSearchParams params) {
        return getPicks(id, params).map(PickViewDtoV2.mapper());
    }

    @RequestMapping(value = "/{id}/picks", method = GET, produces = ContentTypes.JSON, headers = Headers.API_VERSION_3)
    @ResponseBody
    public PageableCollectionDto<PickViewDtoV3> getPicks(
        @PathVariable Long id, @ModelAttribute PickSearchParams params) {
        return getPicks(id, params).map(PickViewDtoV3.mapper());
    }

    private PageableCollectionDto<PickDto> getPicks(Long id, PickSearchParams params) {
        return facade.getPicks(id, params.getSorting(), params.getPage());
    }

}
```

/api/v2 はきっと来ない
エンドポイント毎に
非互換なバージョンアップを可能にしたい

他にも色々ありました

・虫喰いレイヤー：浸食されたドメインモデル
・依存性肥満：ひたすら共有されるコンポーネント

etc, etc ...

とはいえサービスが急成長する中，　限られたリソースで
全てを理想的にこなすことは難しい …

→ 攻めと守りのバランス
→ サービスの成長速度を維持しつつ, 直すべきところは直す

アジェンダ

自己紹介

サービスの概要

サービスの特徴

NewsPicks を支える技術

本当にあった怖い話

まとめ

まとめ

ヒトと技術を融合して良いサービスをつくる

・編集部 + アナリスト + エンジニアの融合で価値をつくる
・エンジニアもビジネスにコミットする

当たり前のことを当たり前にやってカオスに立ち向かう

・攻めと守りのバランスを考えつつ …
・サービスを成長させるために必要なエンジニアリングを

今後の構想

CMP（コンテンツマーケティングプラットフォーム）

・記事入稿・バズトラッキング
・アドネットワーク

よりユーザーに価値を届けるために

・自社編集部を拡大してコンテンツを強化・クオリティメディアへ
・ヒト + アルゴリズムの強化によって「より良い情報」を届けられるように

一緒に仕事してくれるエンジニアを募集しています！

まだまだカオスの職場です

「世界一の経済メディア」を一緒に作りたい人は是非ご連絡を！

NEWS PICKS

ありがとうございました