Nintendo Switch™ 向け プッシュ通知システム 「NPNS」

任天堂　ネットワークシステム部
渡邉（わたなべ） 大洋（たいよう）
小澤（こざわ） 謹裕（のりひろ）

今日の内容

サービス紹介

構成要素

インフラ設計

技術詳細

振り返り

まとめと展望

渡邉 大洋

- 所属
  - ネットワークシステム部
  - Web エンジニア
- キャリア入社5年目（組み込み→Web）
- これまで
  - ニンテンドーネットワークID, NPNS

小澤 謹裕

- 所属
  - ネットワークシステム部
  - インフラエンジニア
- 入社9年目
- これまで
  - ゲームサーバ, Miiverse, NPNS

サービス紹介

Nintendo Switch

世界 1779万台 （2018年3月末時点）

NPNS

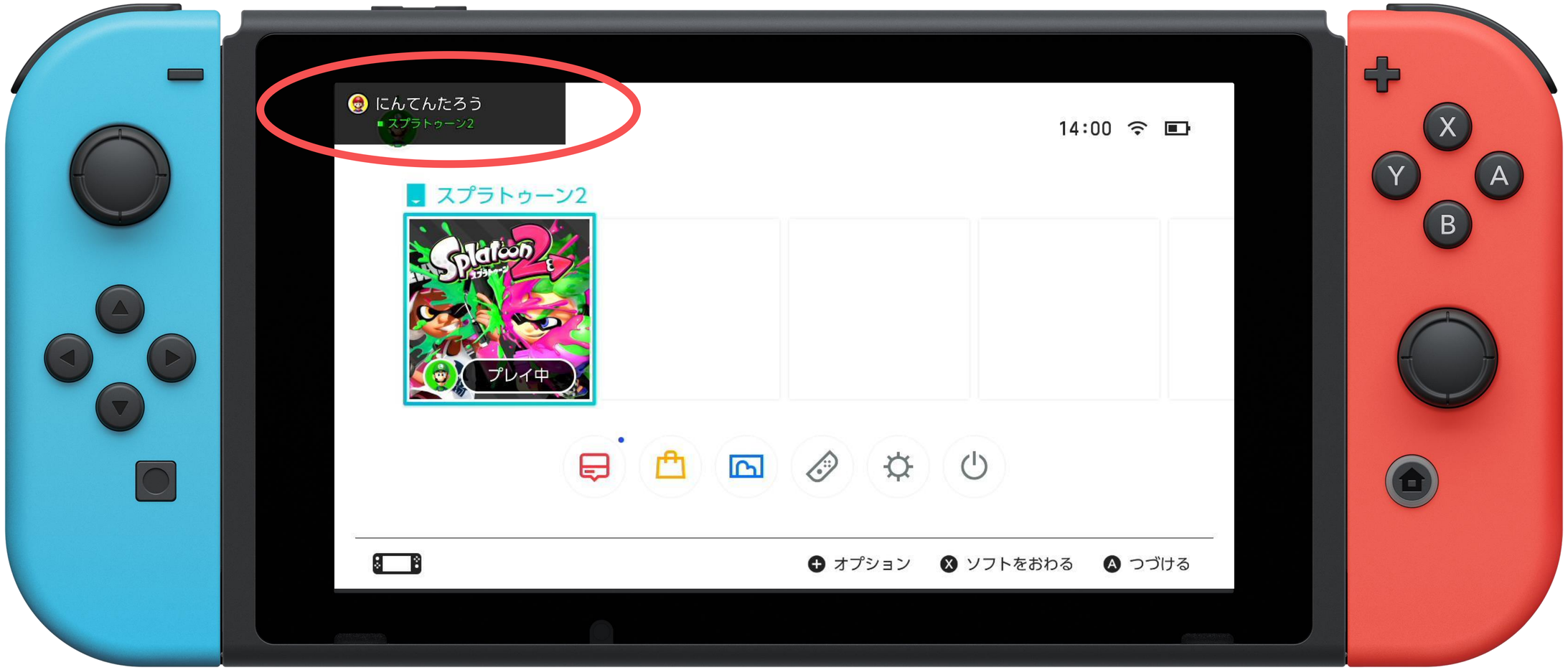
ひとことで

- プッシュ通知システム
- Nintendo Push Notification Service
- 常時接続を使用

他社プラットフォーム

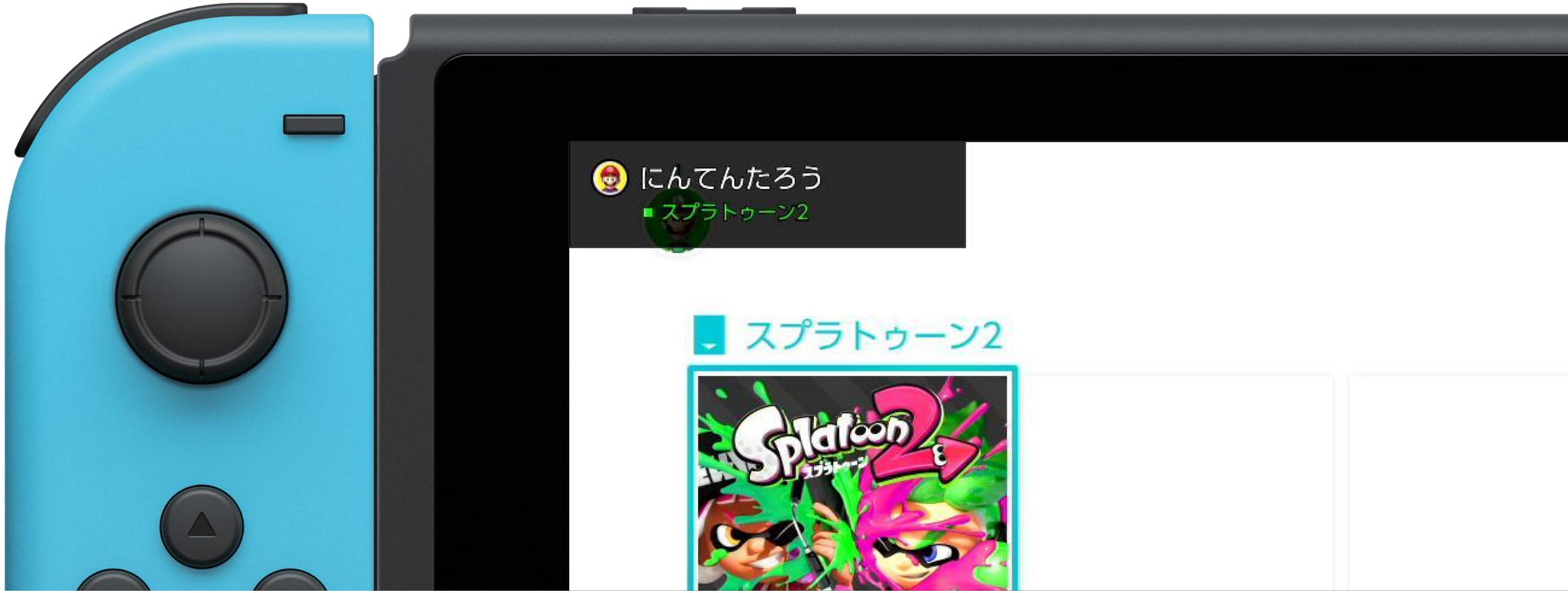
Apple	APNS
Google	FCM

Nintendo Switch 上の通知



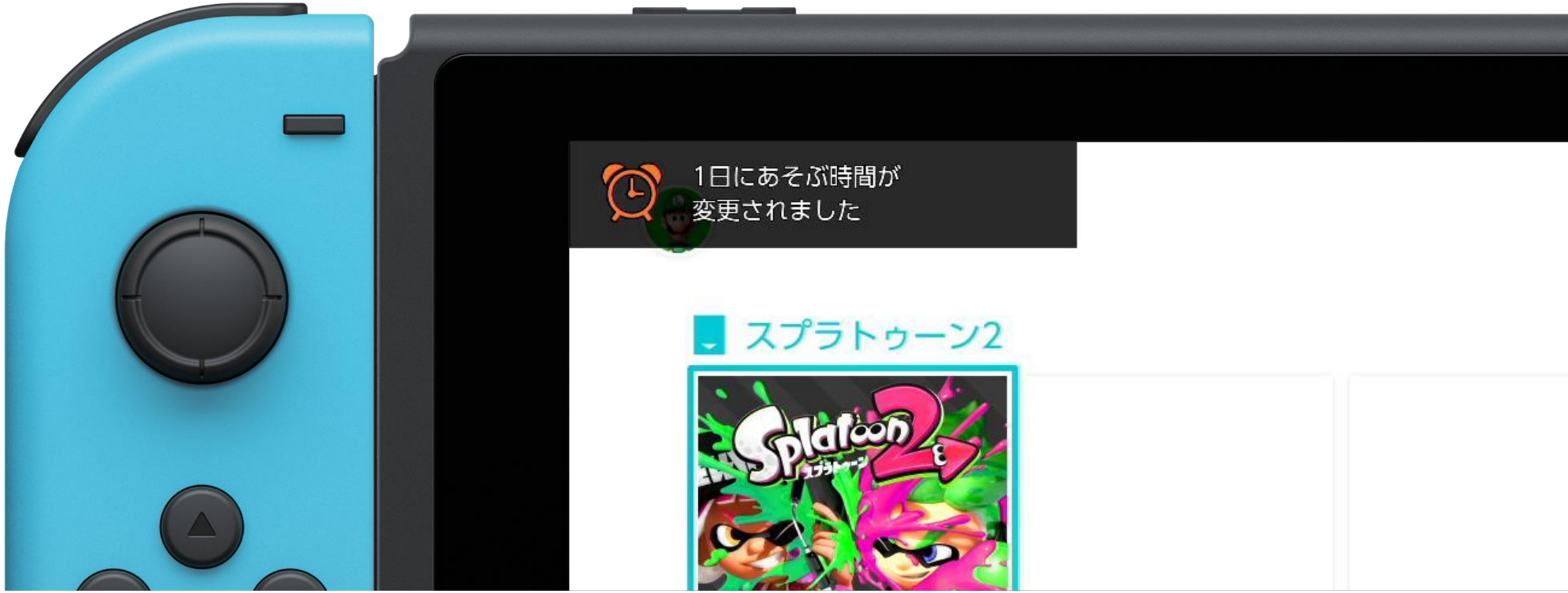
利用例

フレンドのオンライン通知



利用例

Nintendo みまもり Switch™ での設定変更



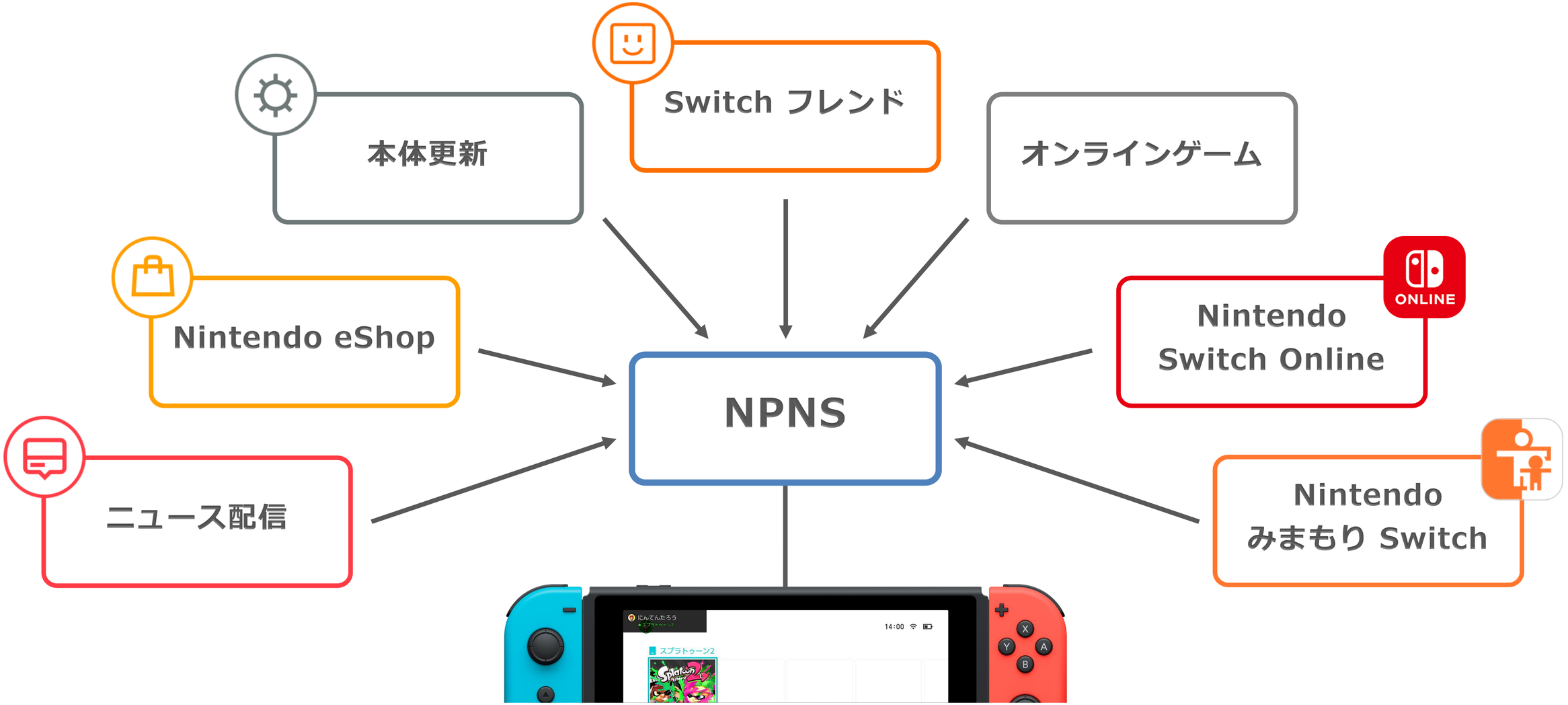
利用例

スマートフォンや PC での購入後の DL 開始指示

利用例

※DL後に完了メッセージ

NPNS利用サービス


本体更新
Switch フレンド
オンラインゲーム
Nintendo eShop
Nintendo
Switch Online
ONLINE
NPNS
ニュース配信
Nintendo
みまもり Switch
にんてんたろう
14:00
スプラトゥーン2
X
Y
A
B

性能要件など

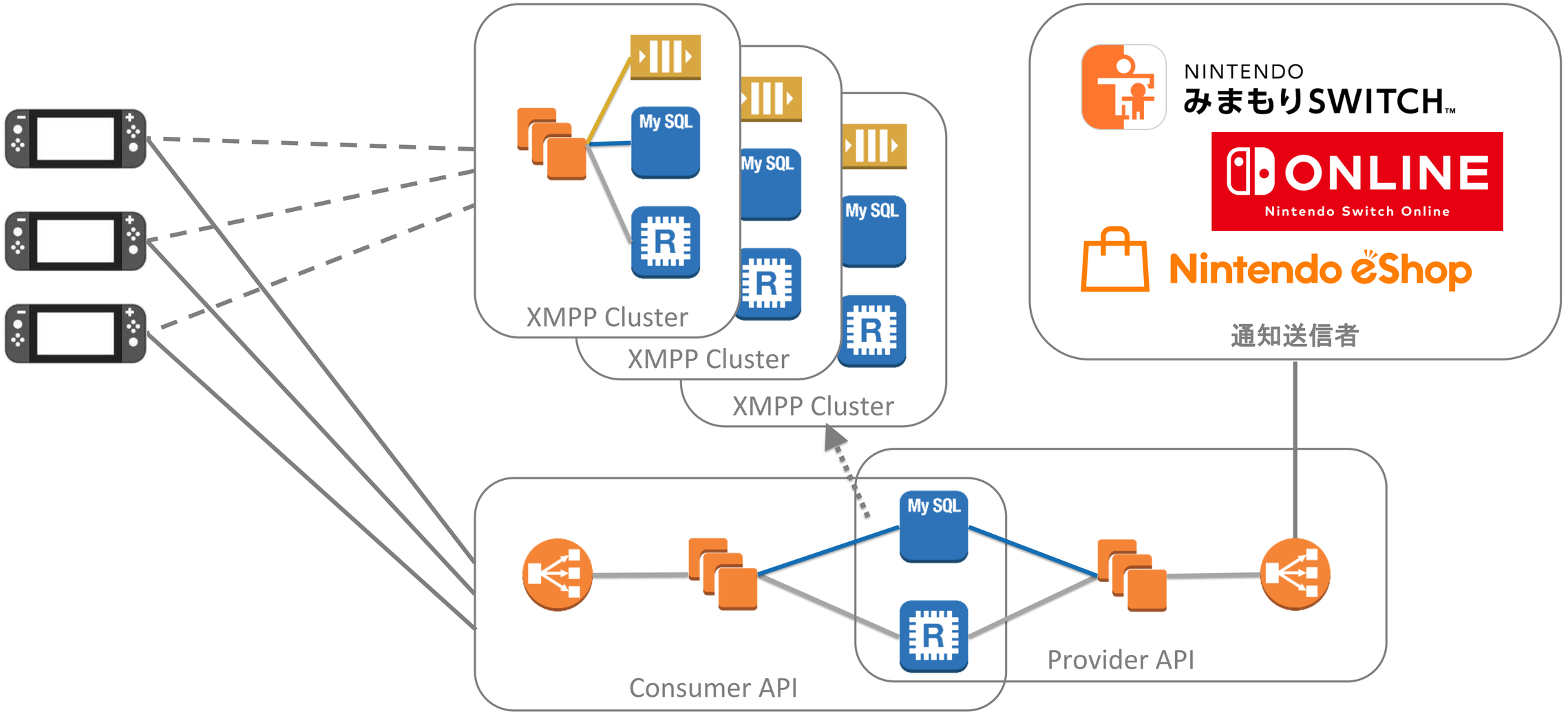
スケーラビリティ	１億台に備える
可用率	AWSリージョン規模の障害未満は稼動
遅延	～数秒（正常時）、～数分（異常時）
インフラ費用	適切な範囲に抑える

構成要素

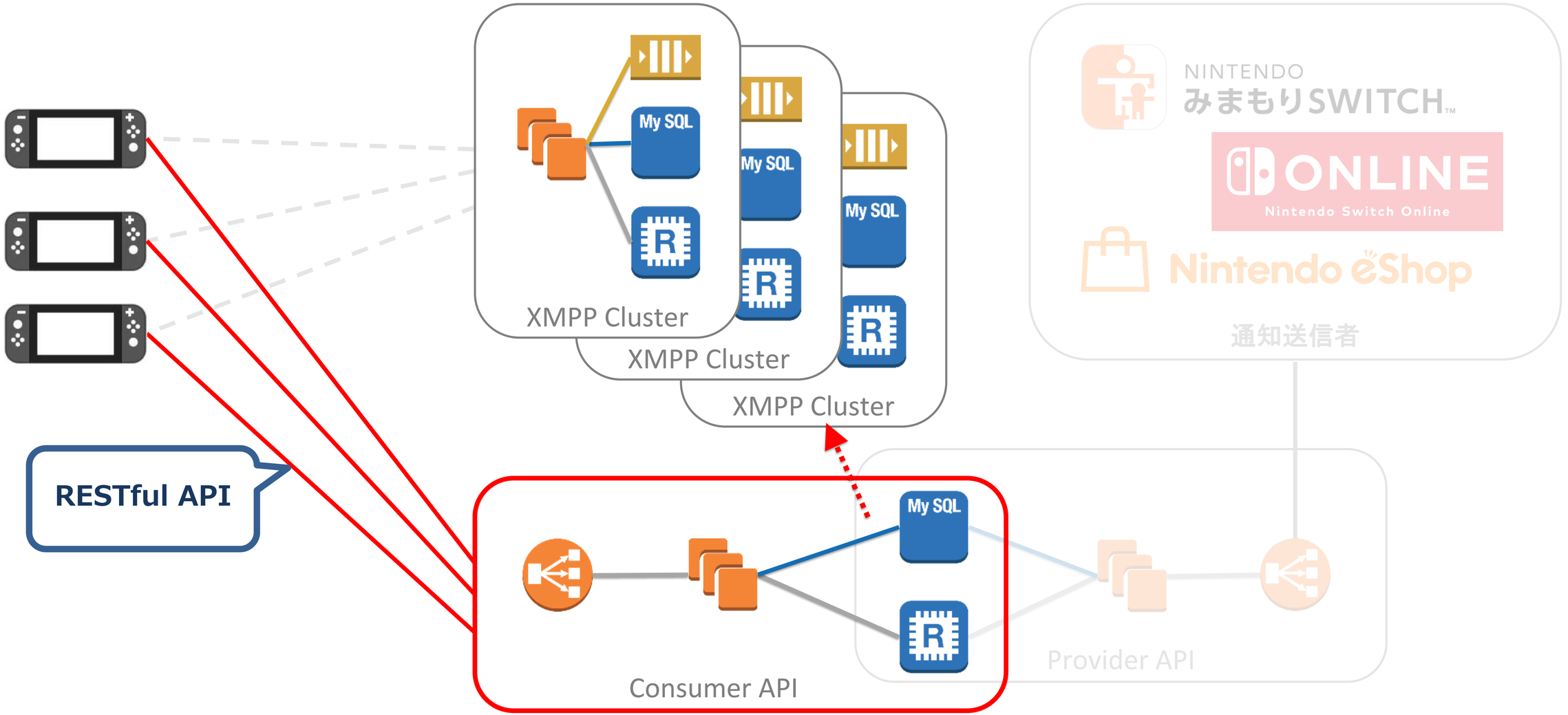
機能分割

- XMPP Cluster
  - Nintendo Switch 向け、常時接続 & 通知
- Consumer API
  - Nintendo Switch 向け、ID 管理
- Provider API
  - サーバ間連携向け、通知送信要求

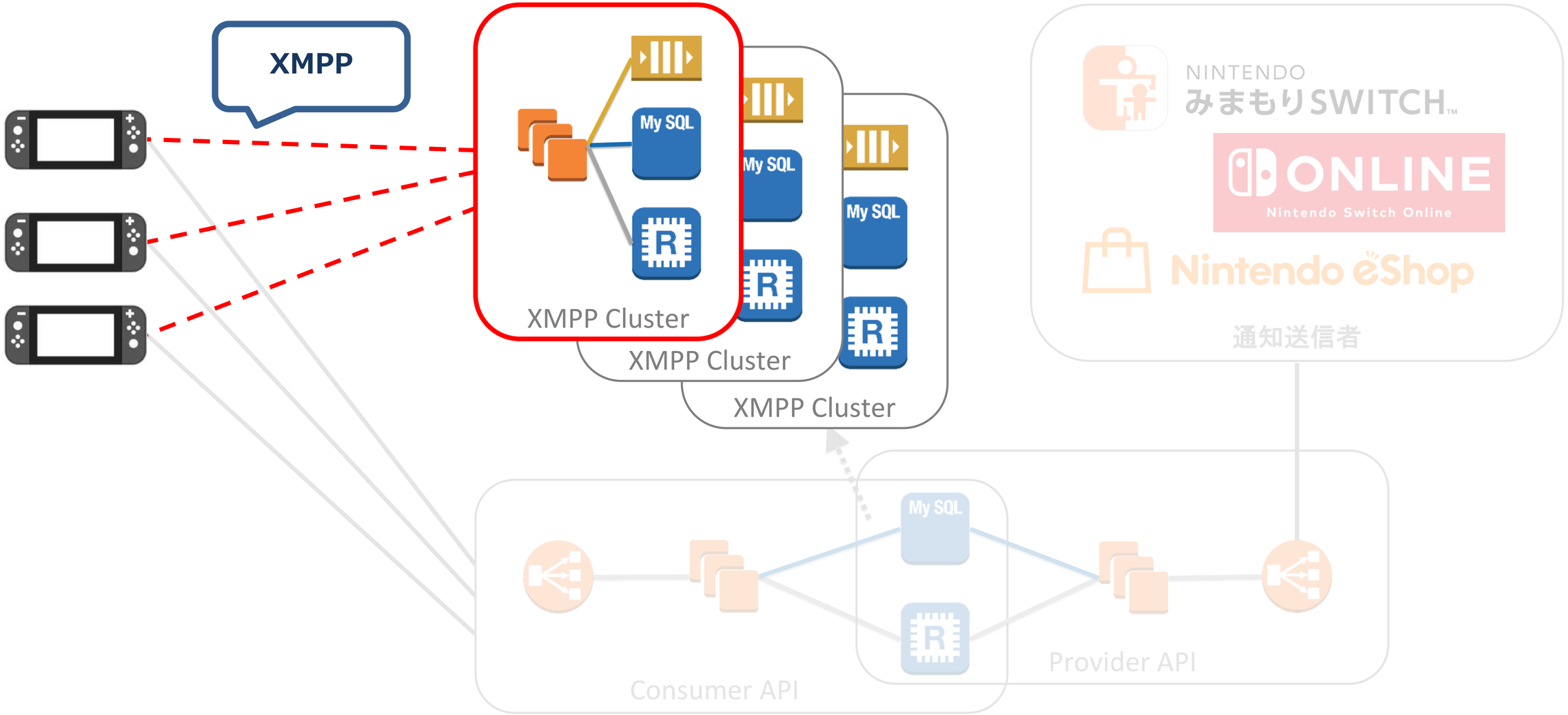
関係図


My SQL
R
XMPP Cluster
XMPP Cluster
XMPP Cluster
NINTENDO
みまもりSWITCH™
ONLINE
Nintendo Switch Online
Nintendo eShop
通知送信者
My SQL
R
Consumer API
Provider API

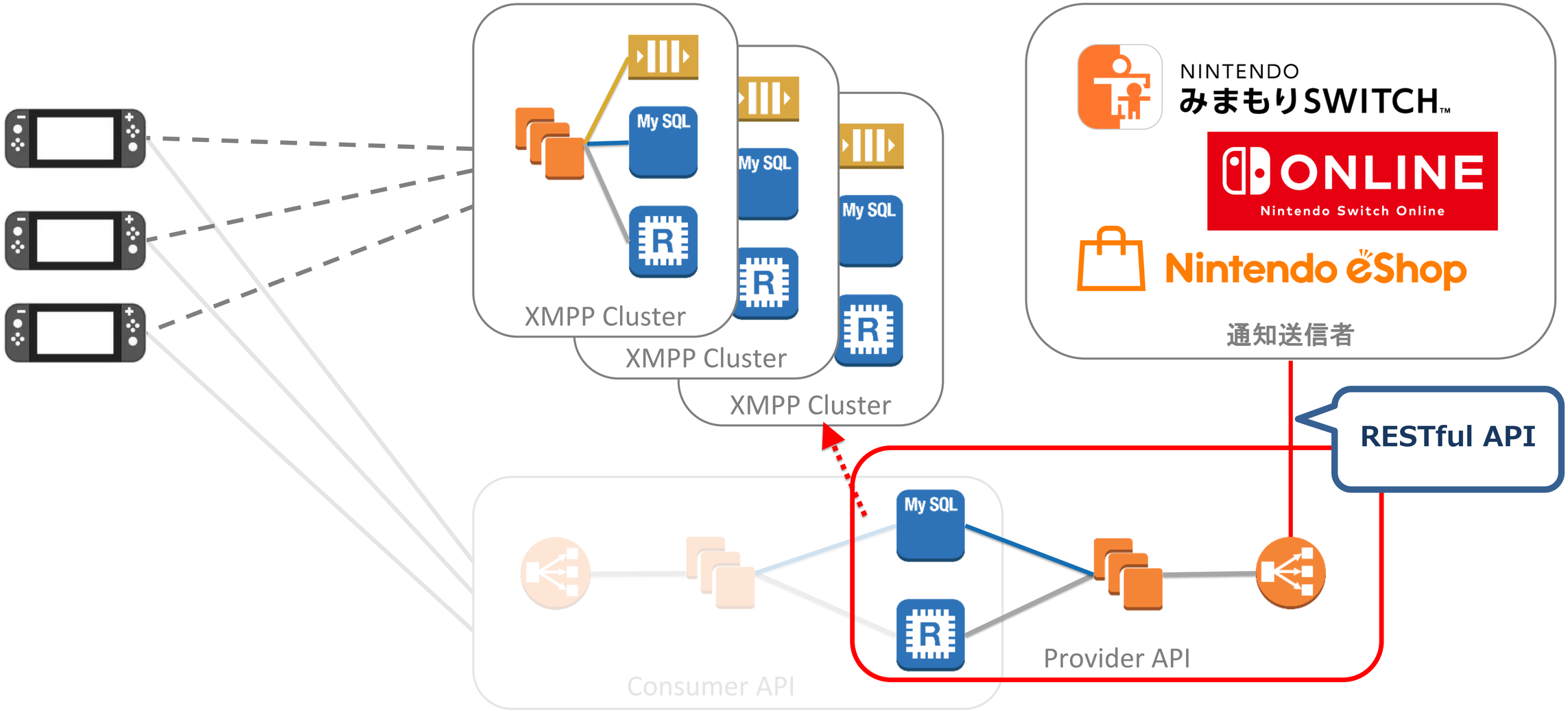
Consumer API



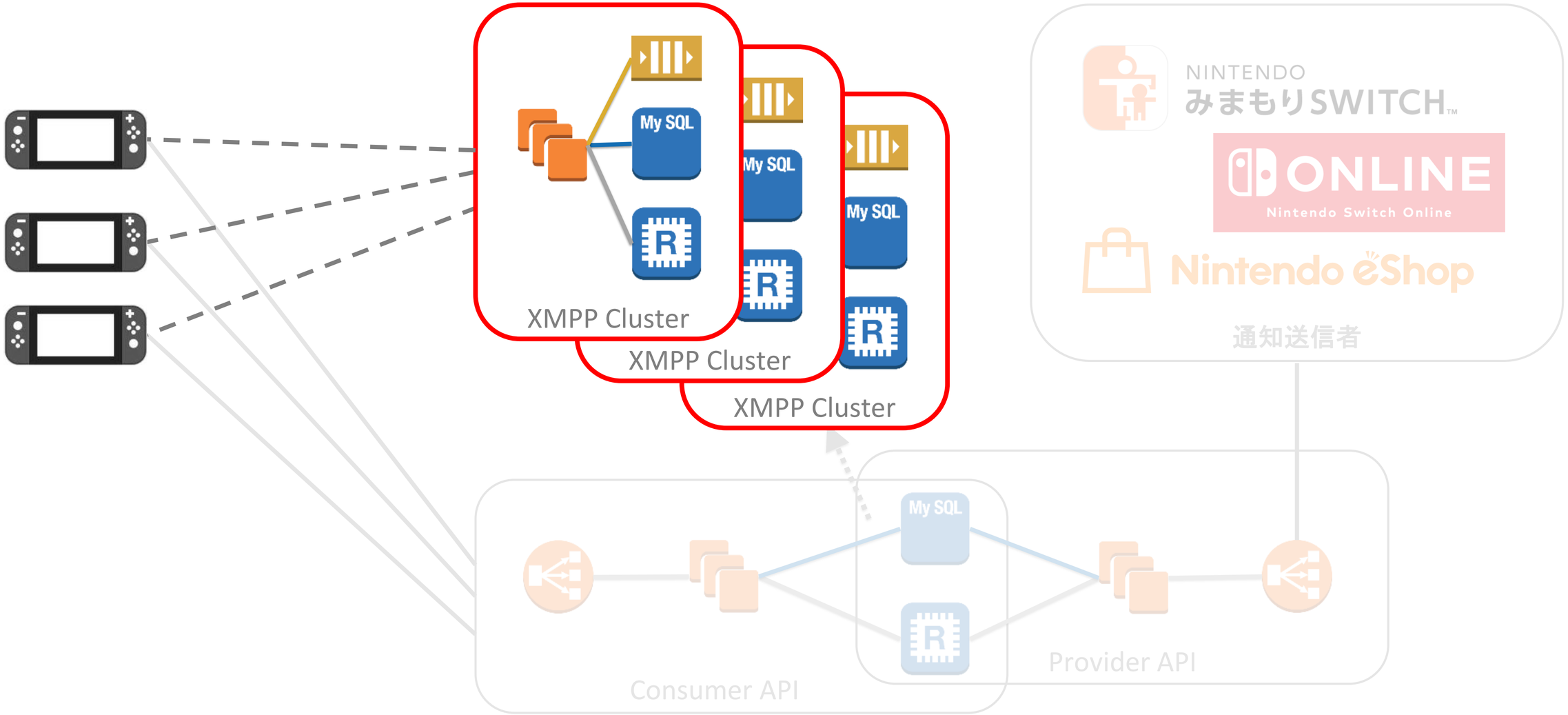
XMPP Cluster



Provider API



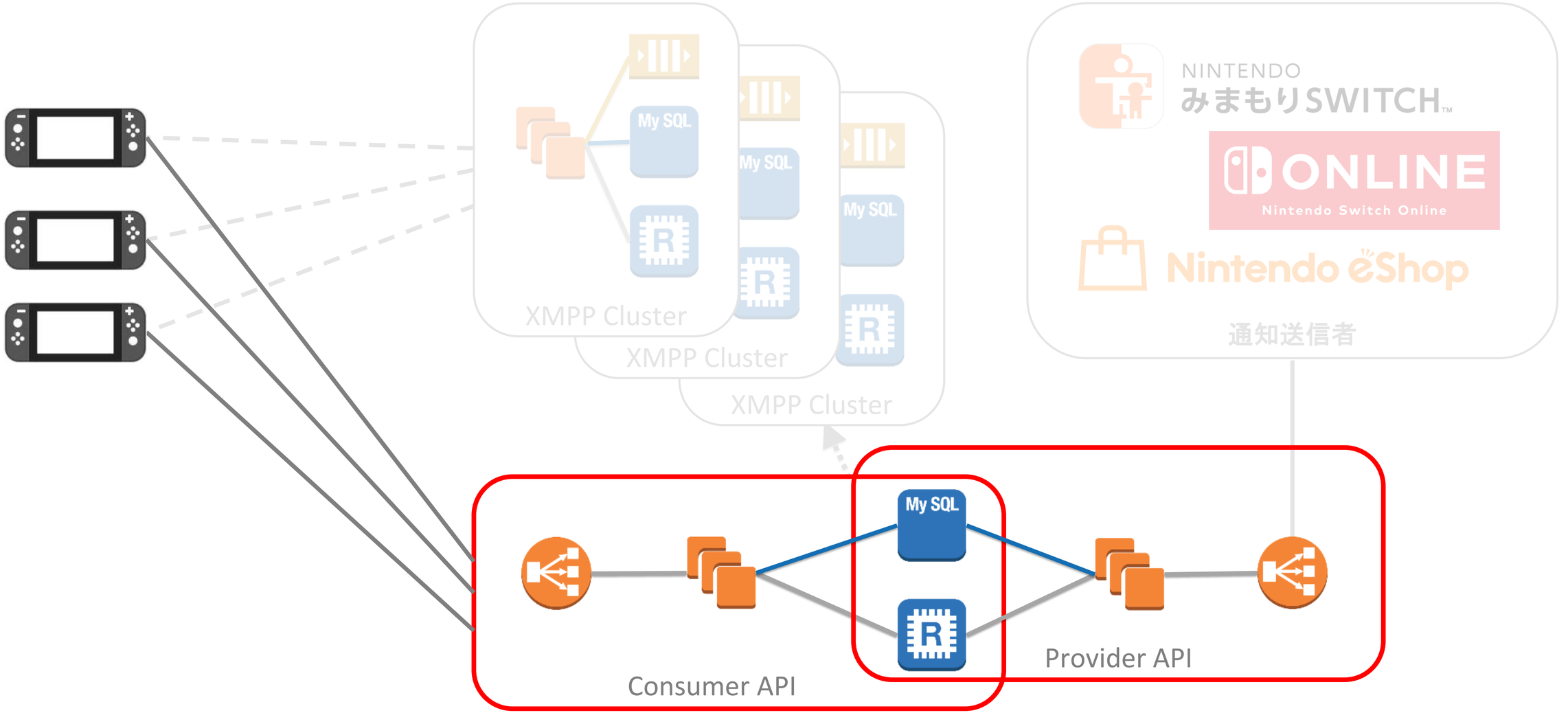
XMPP Cluster



XMPP Cluster 採用技術

- ejabberd
  - Erlang で書かれた OSS
  - クラスタ対応の XMPP サーバ
  - 大規模な利用実績が多い
-  RDS for MySQL 5.7
  - 高負荷時のレプリ遅延が少ない
-  ElastiCache for Redis

Consumer/Provider



Consumer/Provider 採用技術

- Ruby on Rails
  - 社内での開発 / 運用の実績から
-  Amazon Aurora 5.6
  - RDS for MySQL から乗り換え
  - 柔軟なスケール(レプリカ数やサイズ変更)
-  ElastiCache for Redis

その他

-  Amazon SQS
-  ELB
-  Route53
-  Consul
  - クラスタ動作、ネットワーク分断に強い
  - ejabberd の Failure Detection

インフラ設計

クラウド選定

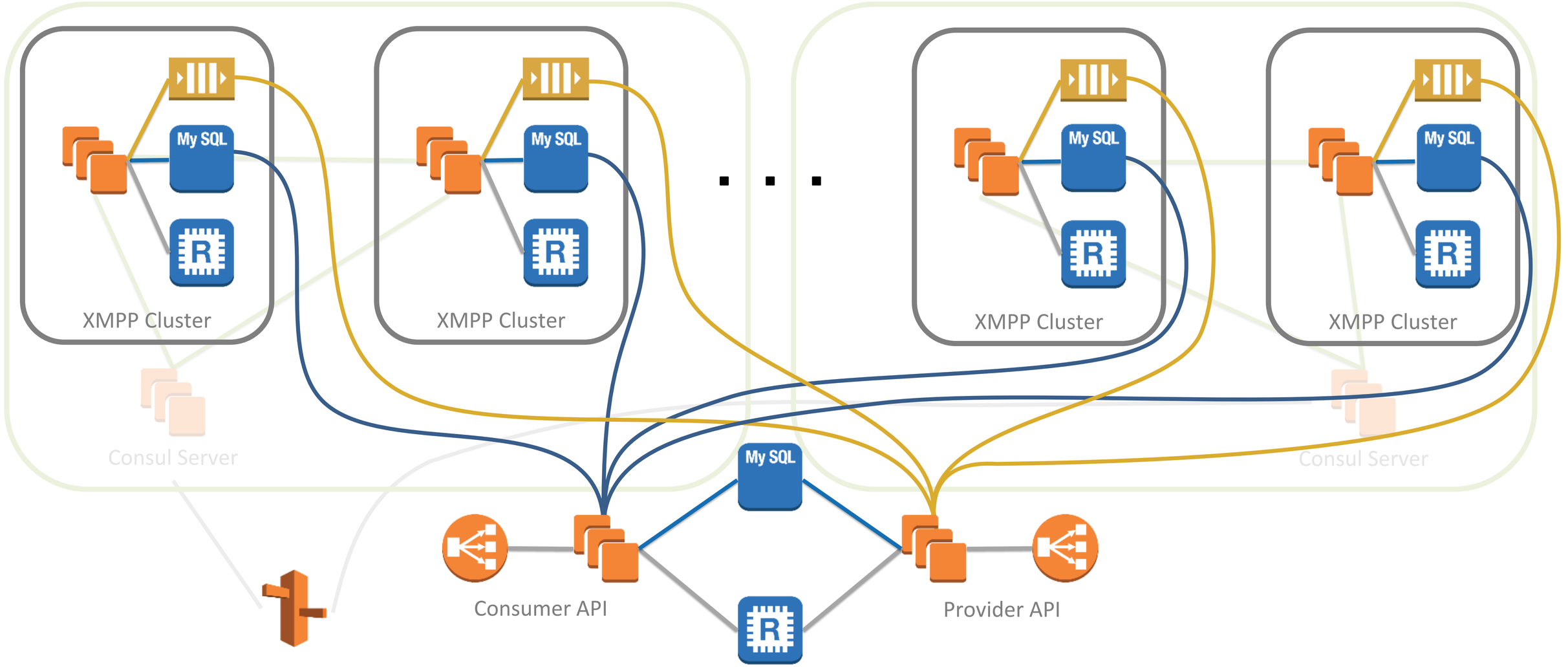
AWS を IaaS として利用

- 社内実績
- メンバーの経験／スキル
- マネージドサービス (RDS, Aurora, SQS)

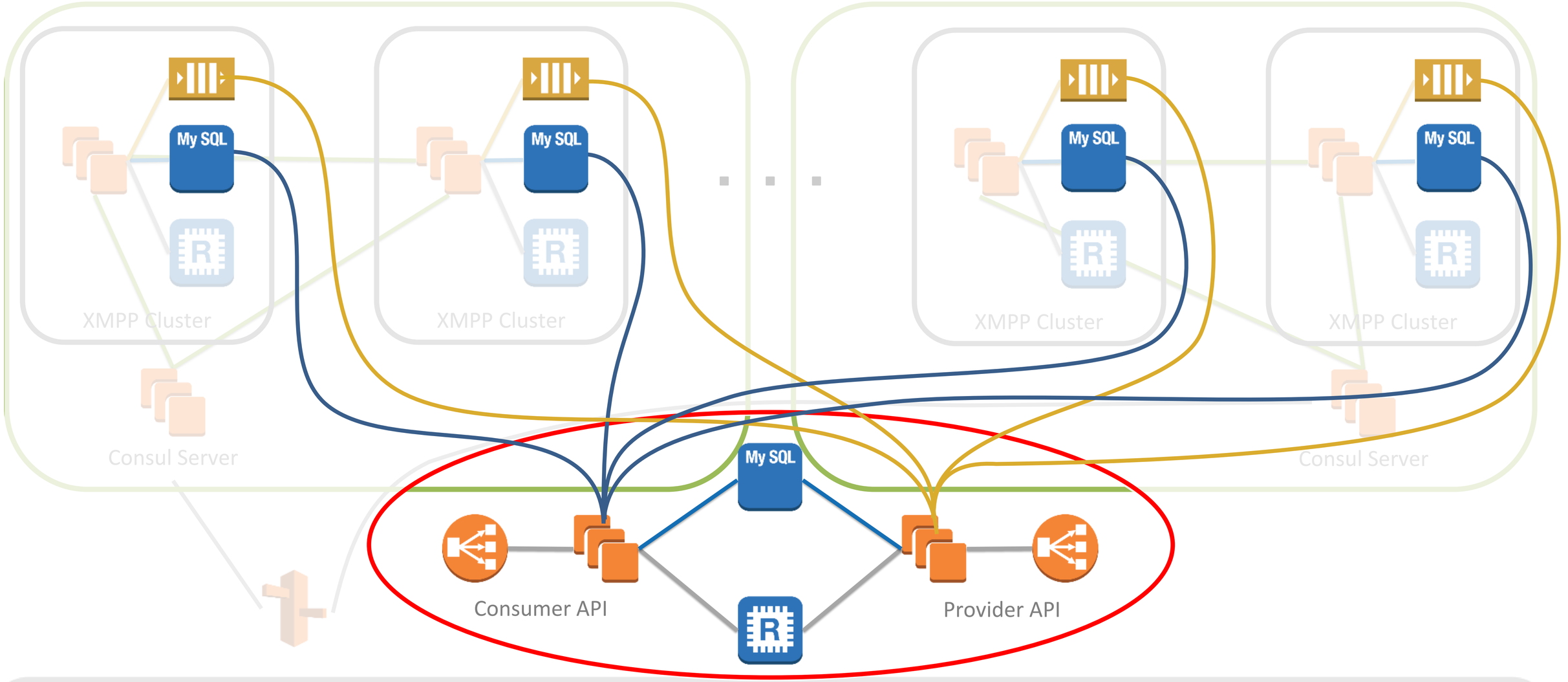
配置

- シングルリージョン
  - ネットワーク安定性、リソースの利用効率向上
- Multi-AZ

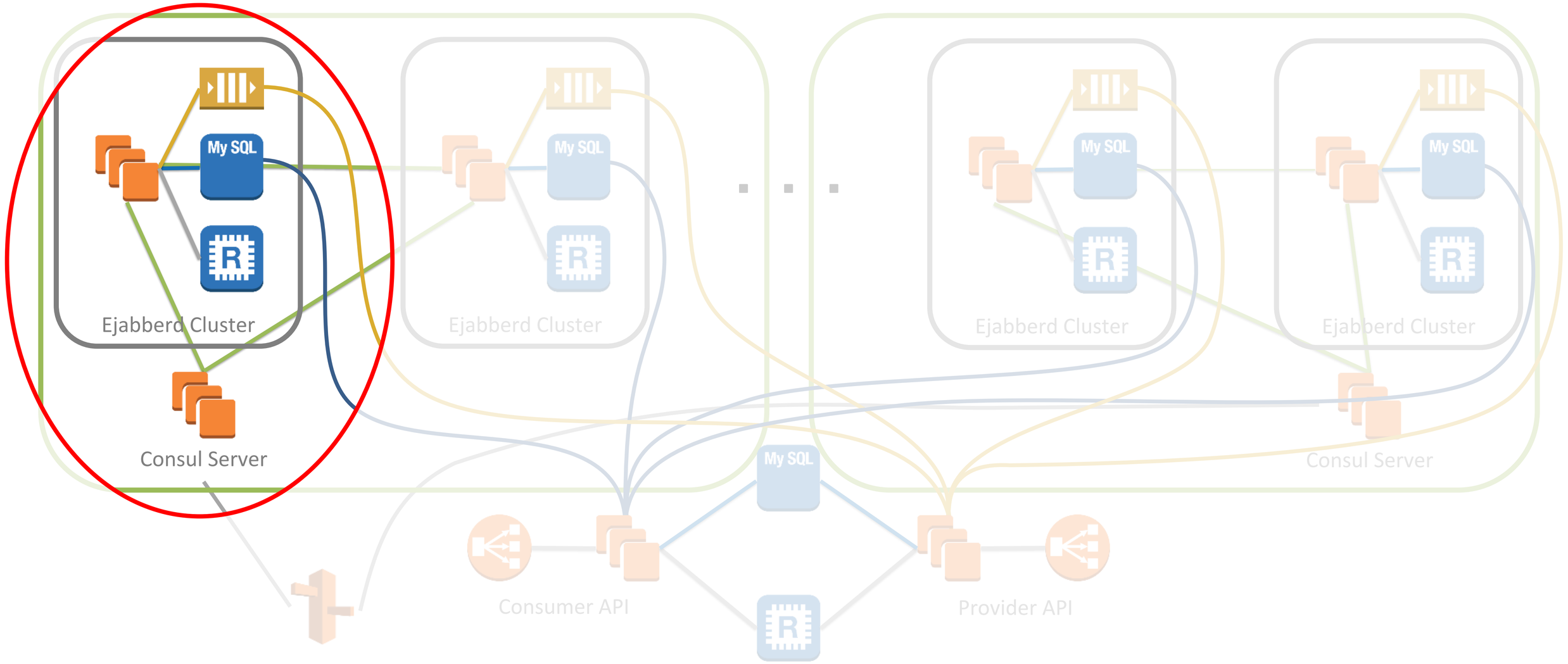
全体像


My SQL
R
XMPP Cluster
XMPP Cluster
XMPP Cluster
XMPP Cluster
Consul Server
Consul Server
My SQL
R
Consumer API
Provider API

Consumer/Provider



XMPPクラスタ


My SQL
R
Ejabberd Cluster
Consul Server
Consumer API
Provider API

XMPPクラスタ分割

スケーラビリティ

- クラスタ数変更でスケール
  - アプリ層の分割 よりも インフラ層で分割

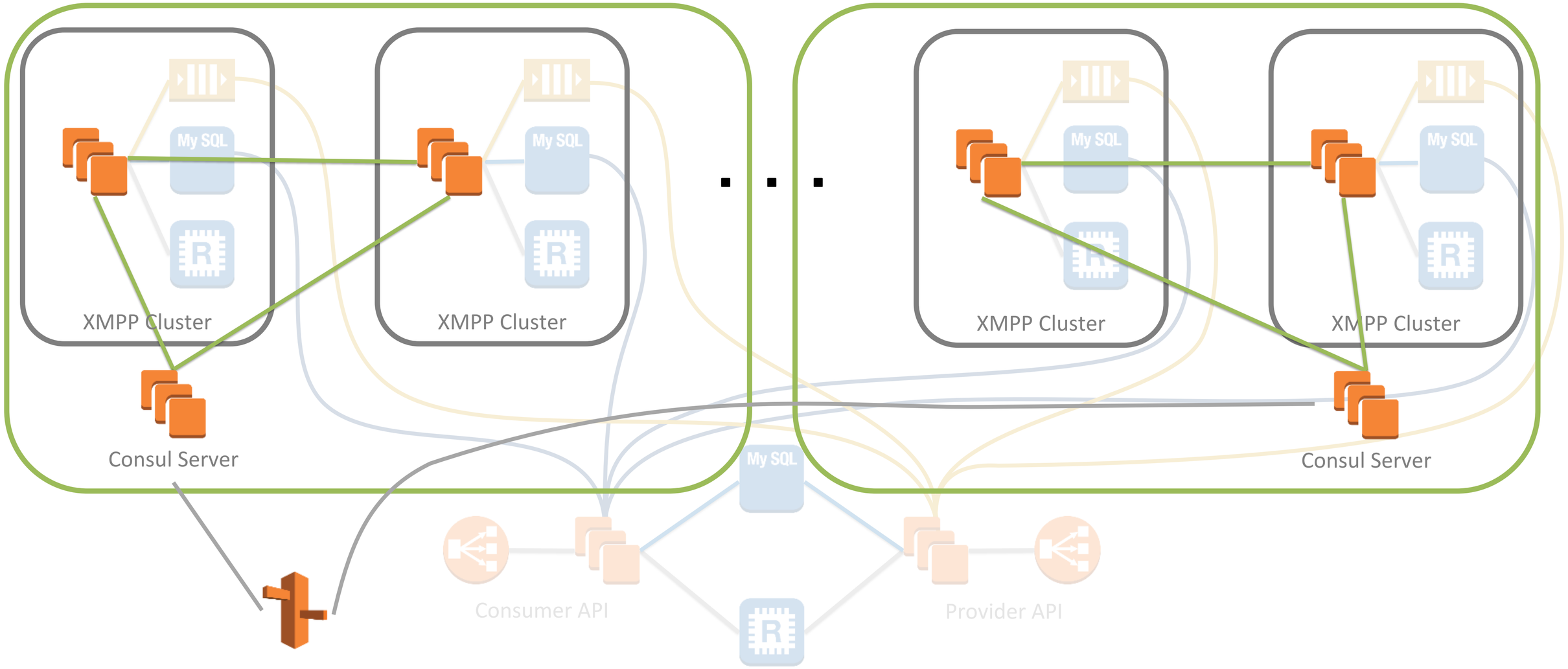
安定性

- クラスタ肥大化の防止
- 障害の局所化

運用効率

- 小さくデプロイ（カナリアリリース）

XMPP+Consul


My SQL
R
XMPP Cluster
XMPP Cluster
XMPP Cluster
XMPP Cluster
Consul Server
Consul Server
My SQL
Consumer API
Provider API

XMPPのロードバランス

ELB は不採用

- Classic Load Balancer
  - 常時接続に使えない
- Application Load Balancer
  - プロトコルがあわない
- Network Load Balancer
  - 検証時期とリリース時期があわない

XMPPのロードバランス

直結＋DNSラウンドロビン

- Route53 に全ノードの A レコードを登録
- Consul が Route53 のレコードを更新

ディスカバリは用意しない

- 外部向け I/F は増やさない

技術詳細

1. クラスタ維持方法
2. デプロイ方法
3. 省メモリ化
4. TCP 切断対策

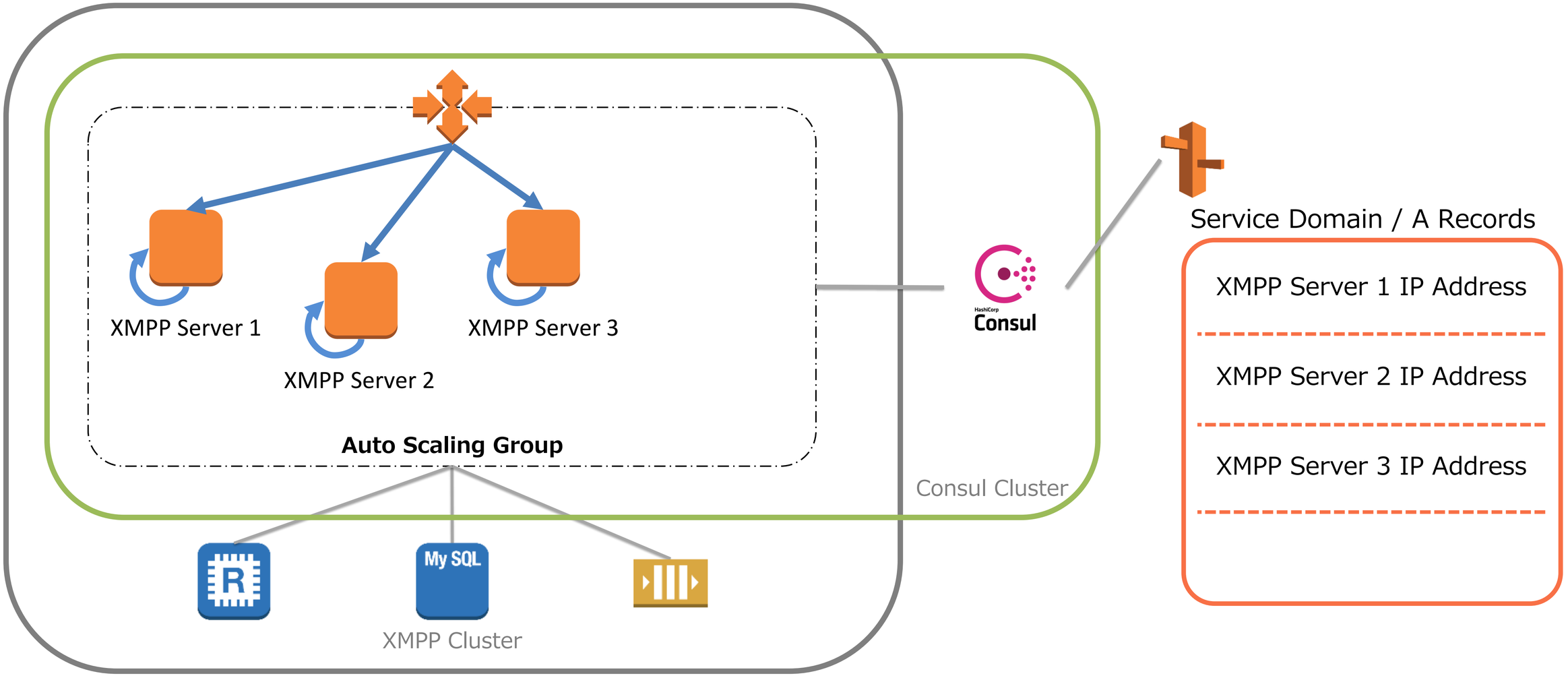
クラスタ維持

ejabberdを二方向から死活監視

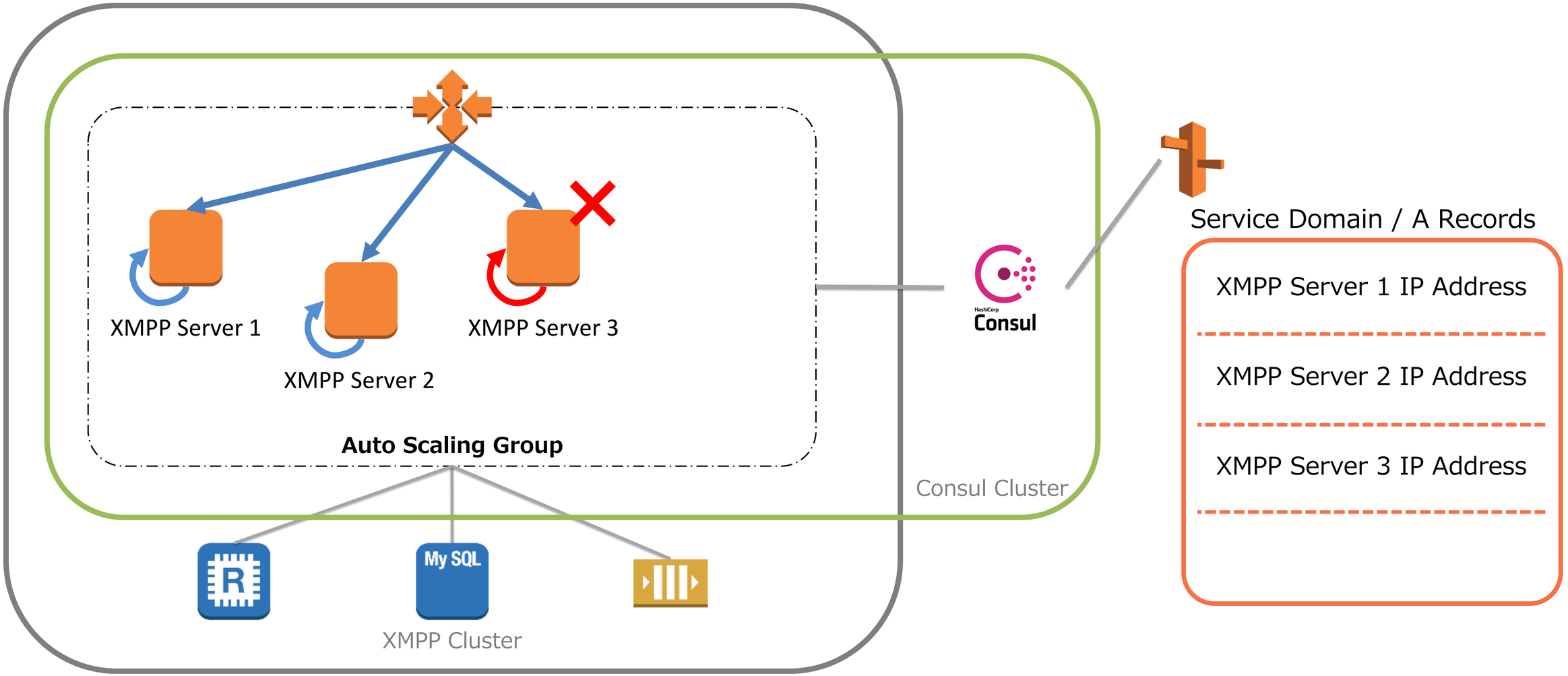
- 

Consul
  - すぐ反応
  - Route53 エントリの管理 (TTL=30)
- 
Auto Scaling
  - 慎重に様子見
  - 異常のあるインスタンスの置き換え

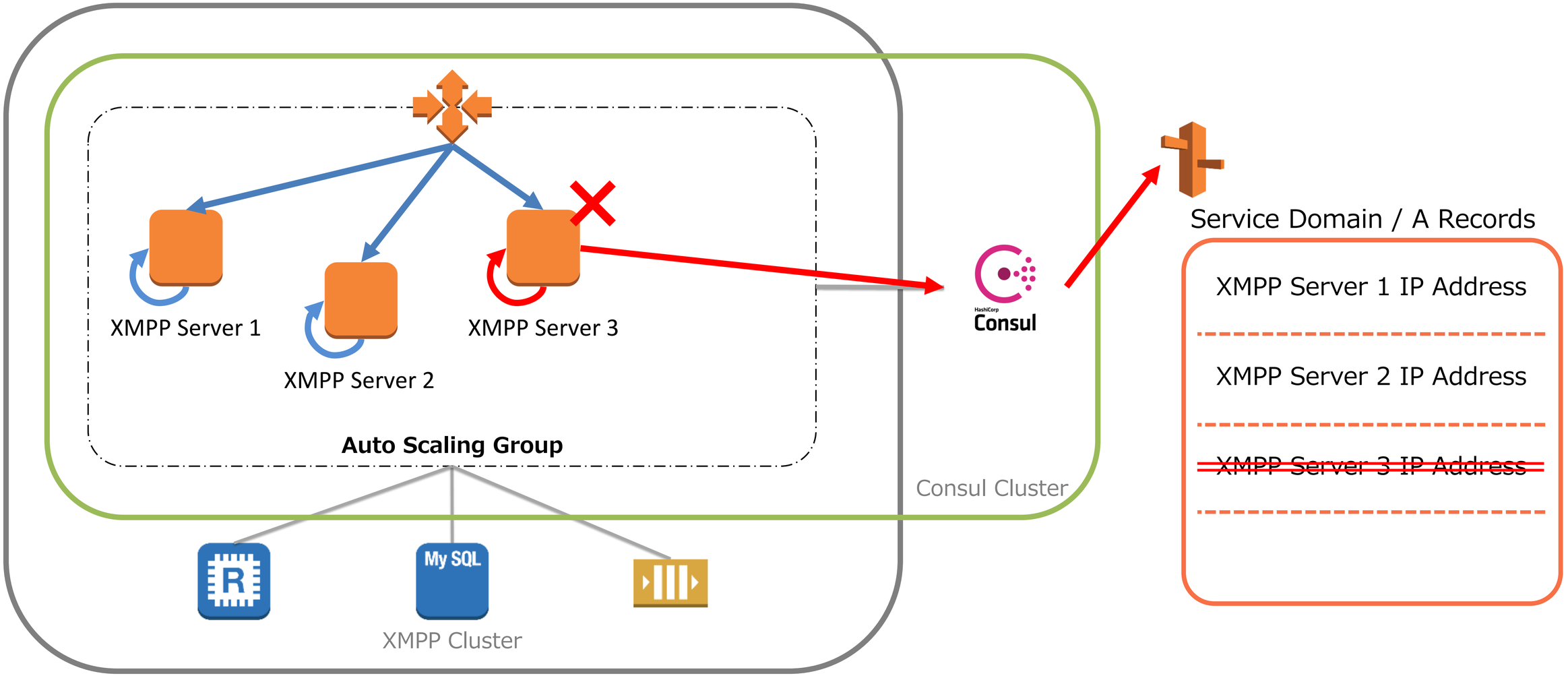
クラスタ維持 [1/6]


XMPP Server 1
XMPP Server 2
XMPP Server 3
Auto Scaling Group
HashiCorp
Consul
Consul Cluster
My SQL
XMPP Cluster
Service Domain / A Records
XMPP Server 1 IP Address
XMPP Server 2 IP Address
XMPP Server 3 IP Address

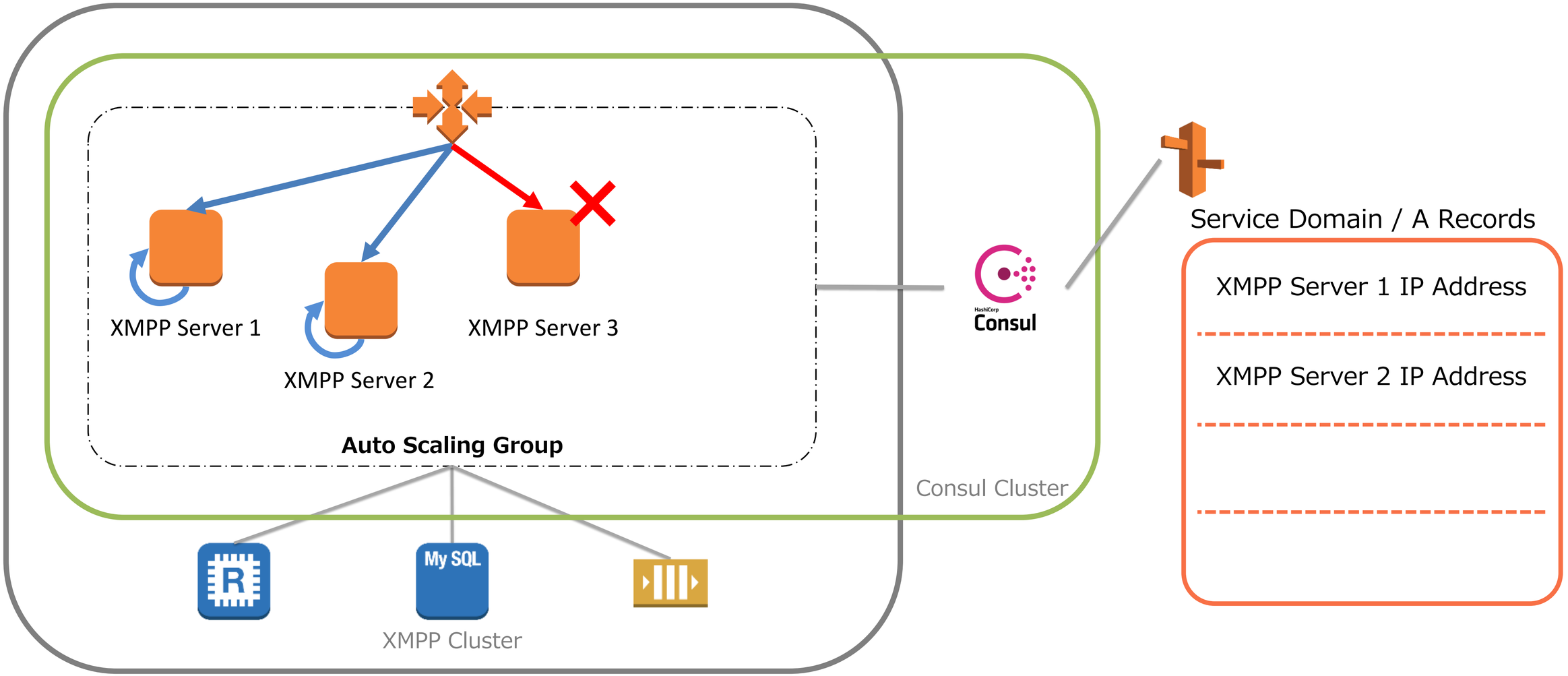
クラスタ維持 [2/6]


XMPP Server 1
XMPP Server 2
XMPP Server 3
Auto Scaling Group
HashiCorp
Consul
Consul Cluster
My SQL
XMPP Cluster
Service Domain / A Records
XMPP Server 1 IP Address
XMPP Server 2 IP Address
XMPP Server 3 IP Address

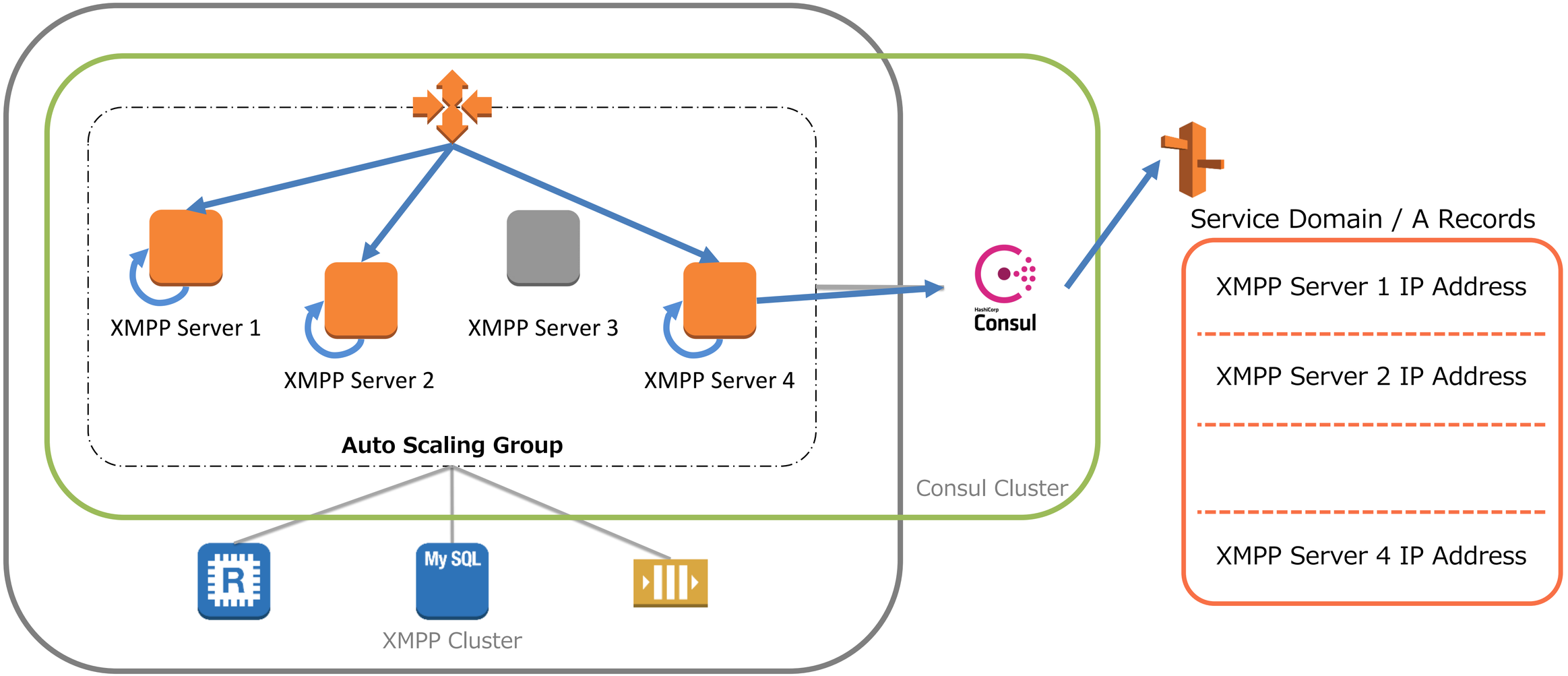
クラスタ維持 [3/6]


XMPP Server 1
XMPP Server 2
XMPP Server 3
Auto Scaling Group
Consul Cluster
HashiCorp
Consul
My SQL
XMPP Cluster
Service Domain / A Records
XMPP Server 1 IP Address
XMPP Server 2 IP Address
XMPP Server 3 IP Address

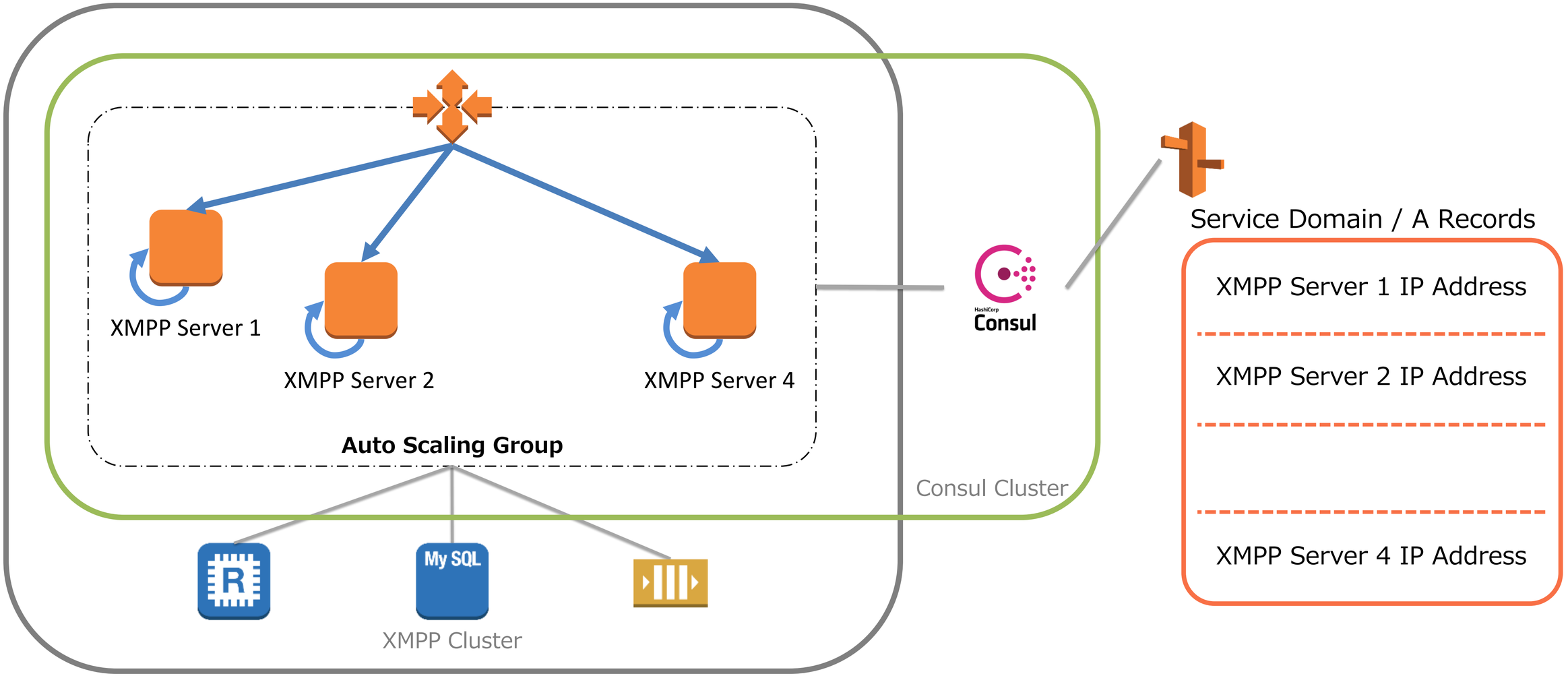
クラスタ維持 [4/6]


XMPP Server 1
XMPP Server 2
XMPP Server 3
Auto Scaling Group
HashiCorp
Consul
Consul Cluster
My SQL
XMPP Cluster
Service Domain / A Records
XMPP Server 1 IP Address
XMPP Server 2 IP Address

クラスタ維持 [5/6]


XMPP Server 1
XMPP Server 2
XMPP Server 3
XMPP Server 4
Auto Scaling Group
HashiCorp
Consul
Consul Cluster
My SQL
XMPP Cluster
Service Domain / A Records
XMPP Server 1 IP Address
XMPP Server 2 IP Address
XMPP Server 4 IP Address

クラスタ維持 [6/6]


XMPP Server 1
XMPP Server 2
XMPP Server 4
Auto Scaling Group
Consul Cluster
HashiCorp
Consul
My SQL
XMPP Cluster
Service Domain / A Records
XMPP Server 1 IP Address
XMPP Server 2 IP Address
XMPP Server 4 IP Address

デプロイ

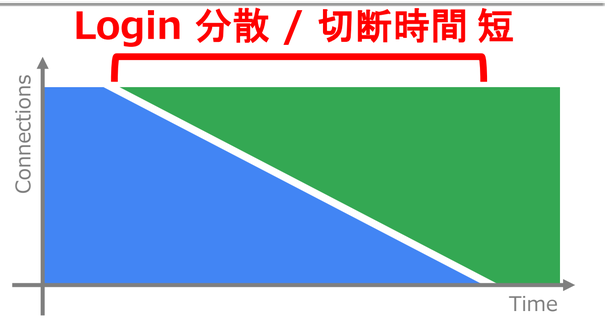
Blue/Green Deploy

- ASG 単位でドメイン切り替え
- 新旧両方を1つのクラスタに
- 旧ノードからユーザを一定レートで追い出す
  ＝ドリップ処理

ドリップ処理

スムーズな切り替え

- 接続を少しずつ Blue → Green に移動



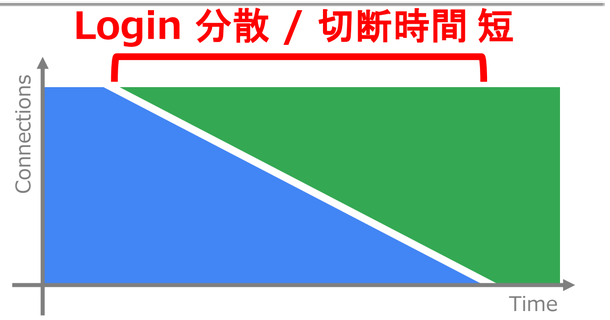
ドリップ処理

スムーズな切り替え

- 接続を少しずつ Blue → Green に移動

常時接続では接続処理が最も高負荷

- 再接続タイミングを分散させたい



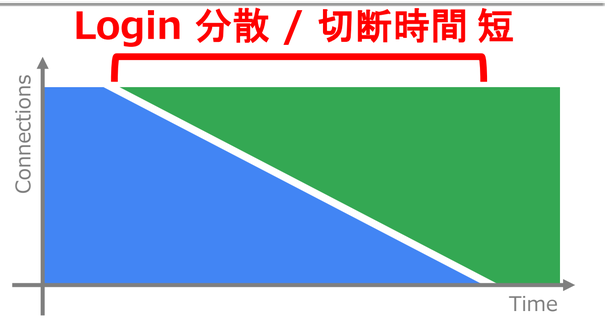
ドリップ処理

スムーズな切り替え

- 接続を少しずつ Blue → Green に移動

常時接続では接続処理が最も高負荷

- 再接続タイミングを分散させたい
- 切断期間は短くしたい



ドリップ処理

スムーズな切り替え

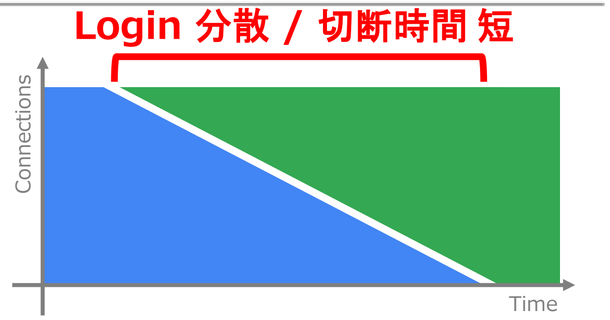
- 接続を少しずつ Blue → Green に移動

常時接続では接続処理が最も高負荷

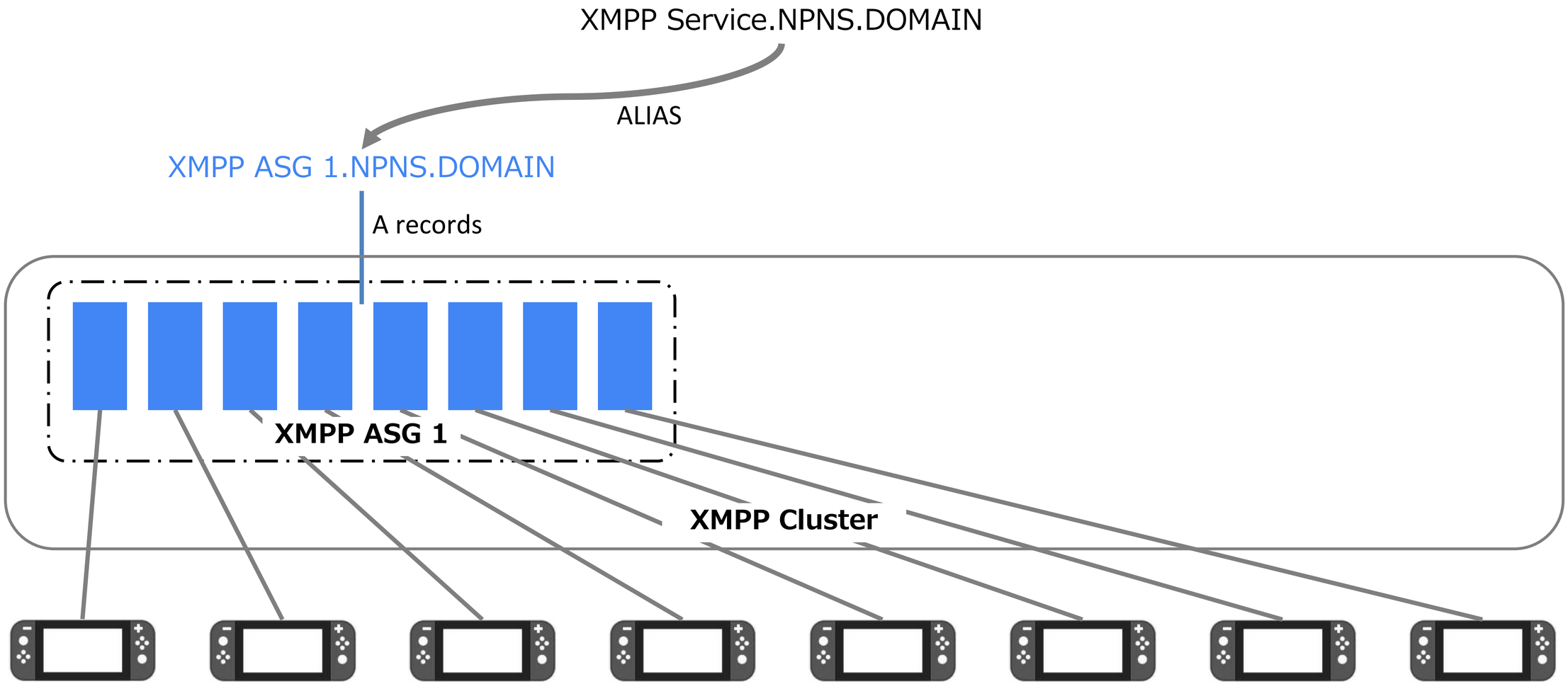
- 再接続タイミングを分散させたい
- 切断期間は短くしたい

やっかいな点

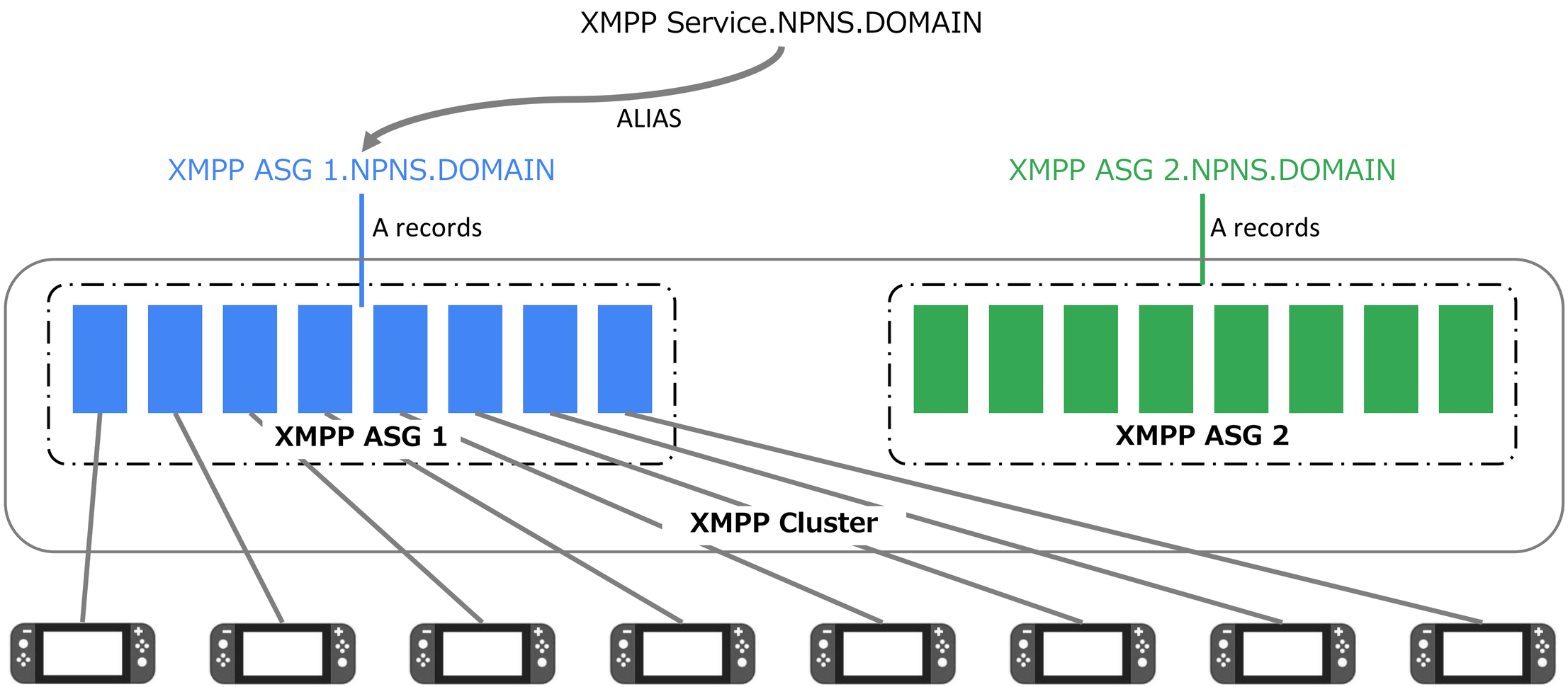
- デプロイが長時間化 → 今は約2時間
- インスタンス費用がその間2倍に



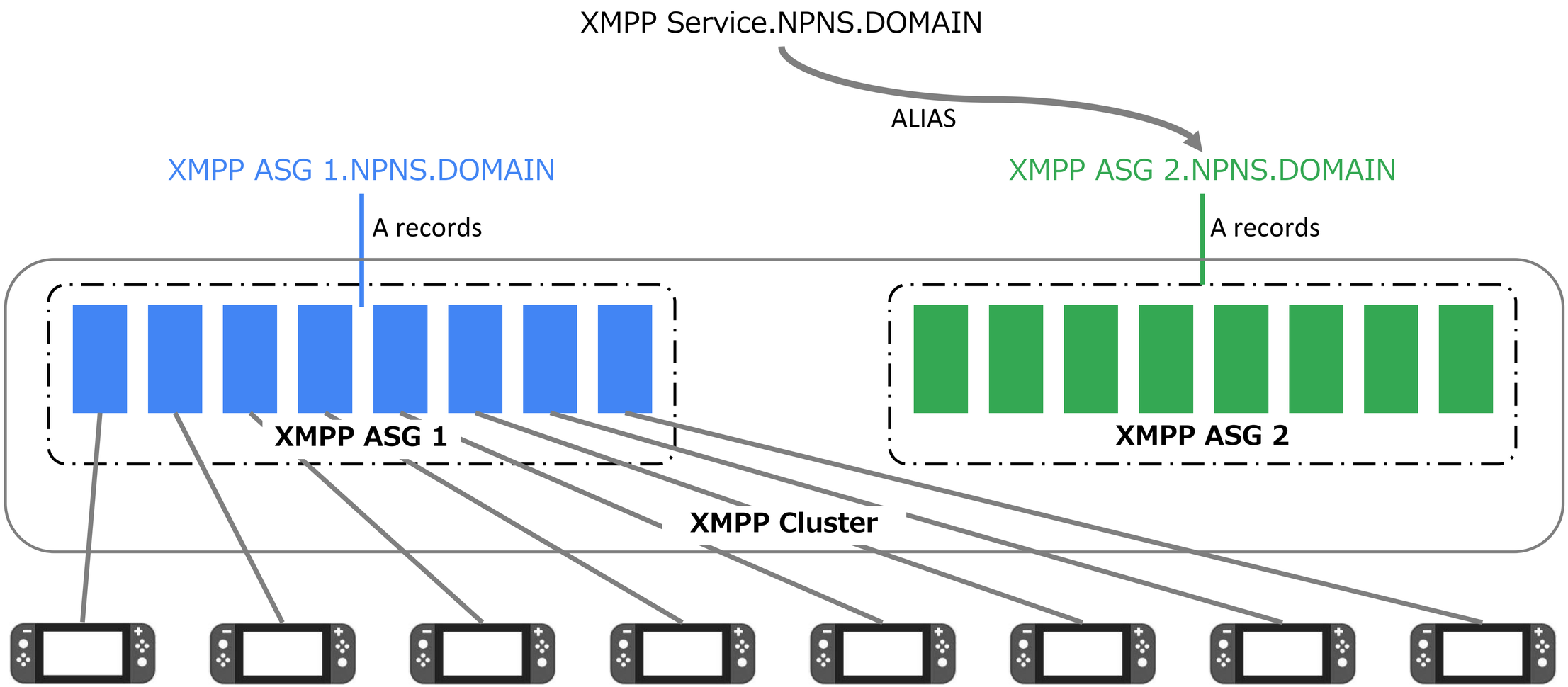
Blue/Green デプロイ [1/6]



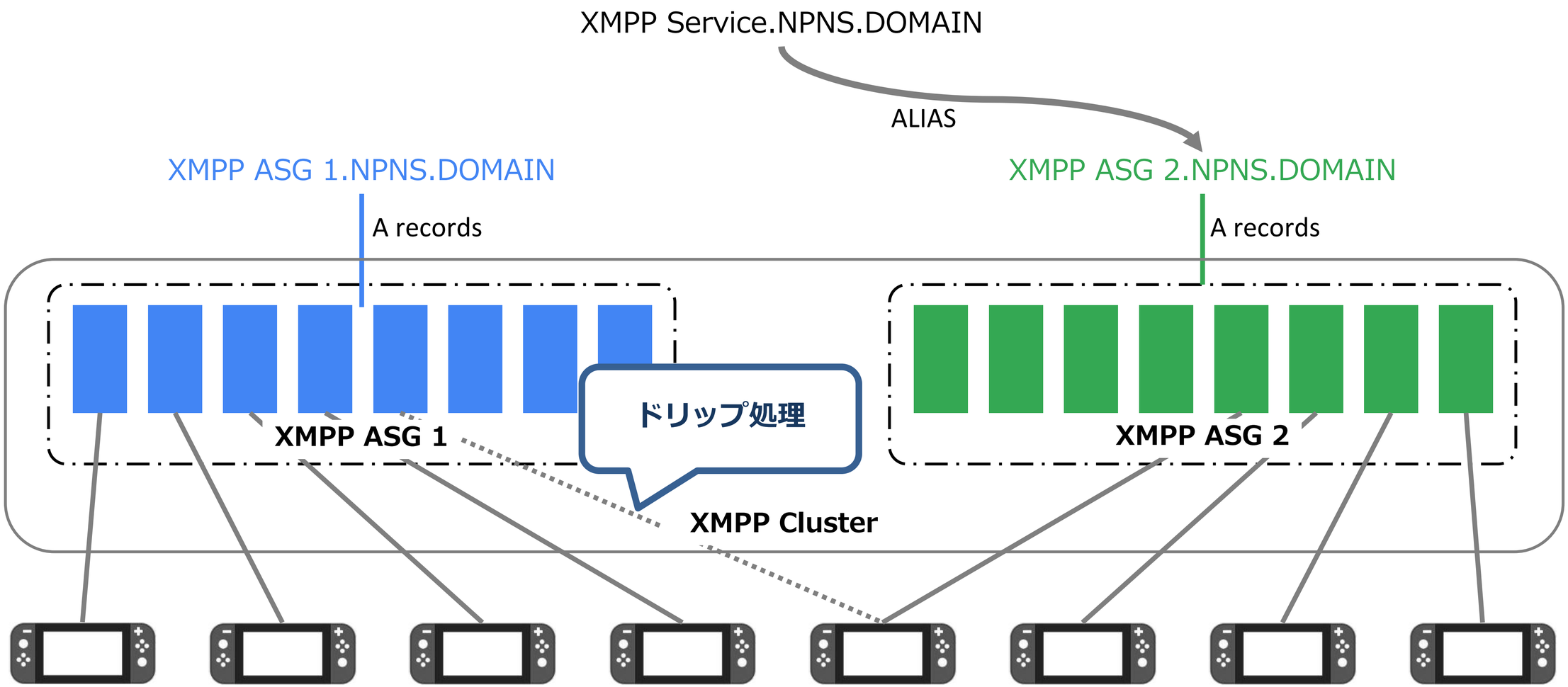
Blue/Green デプロイ [2/6]



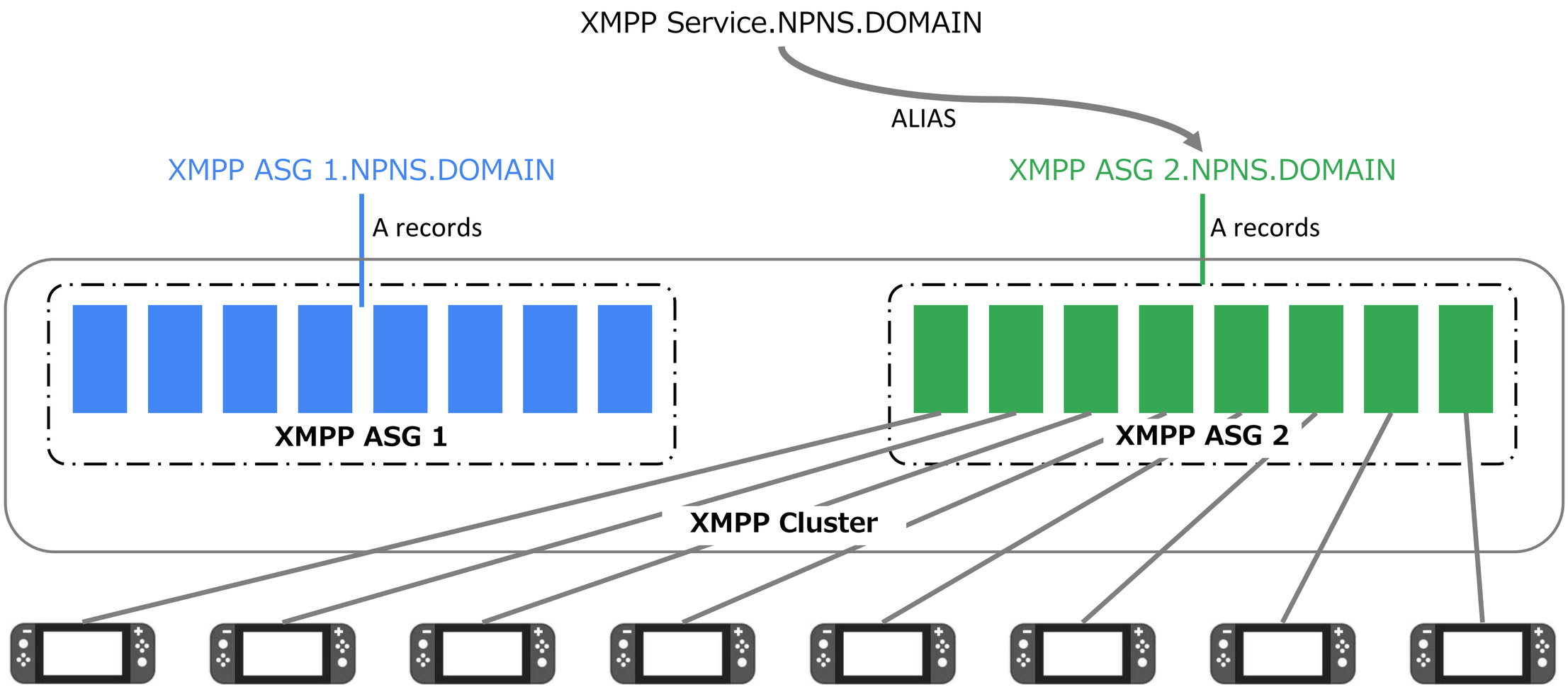
Blue/Green デプロイ [3/6]



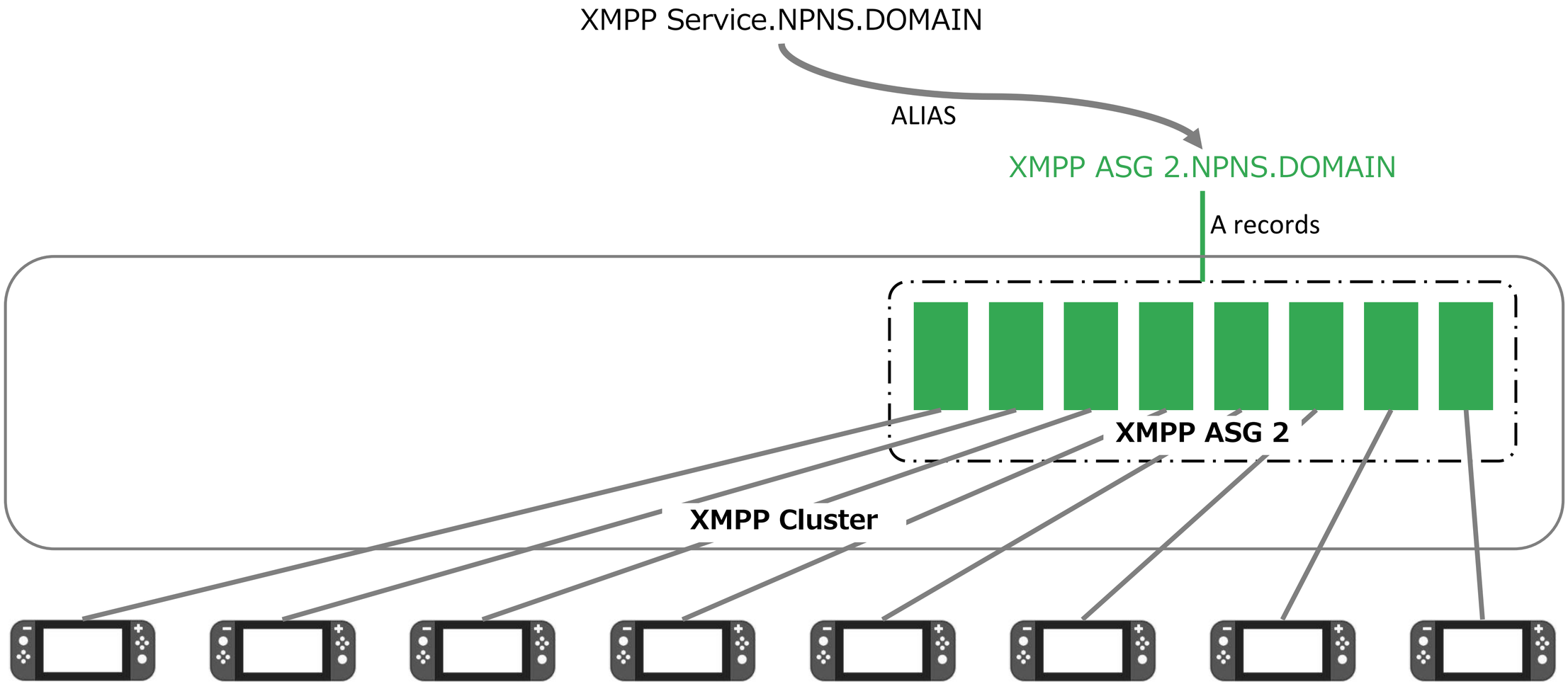
Blue/Green デプロイ [4/6]



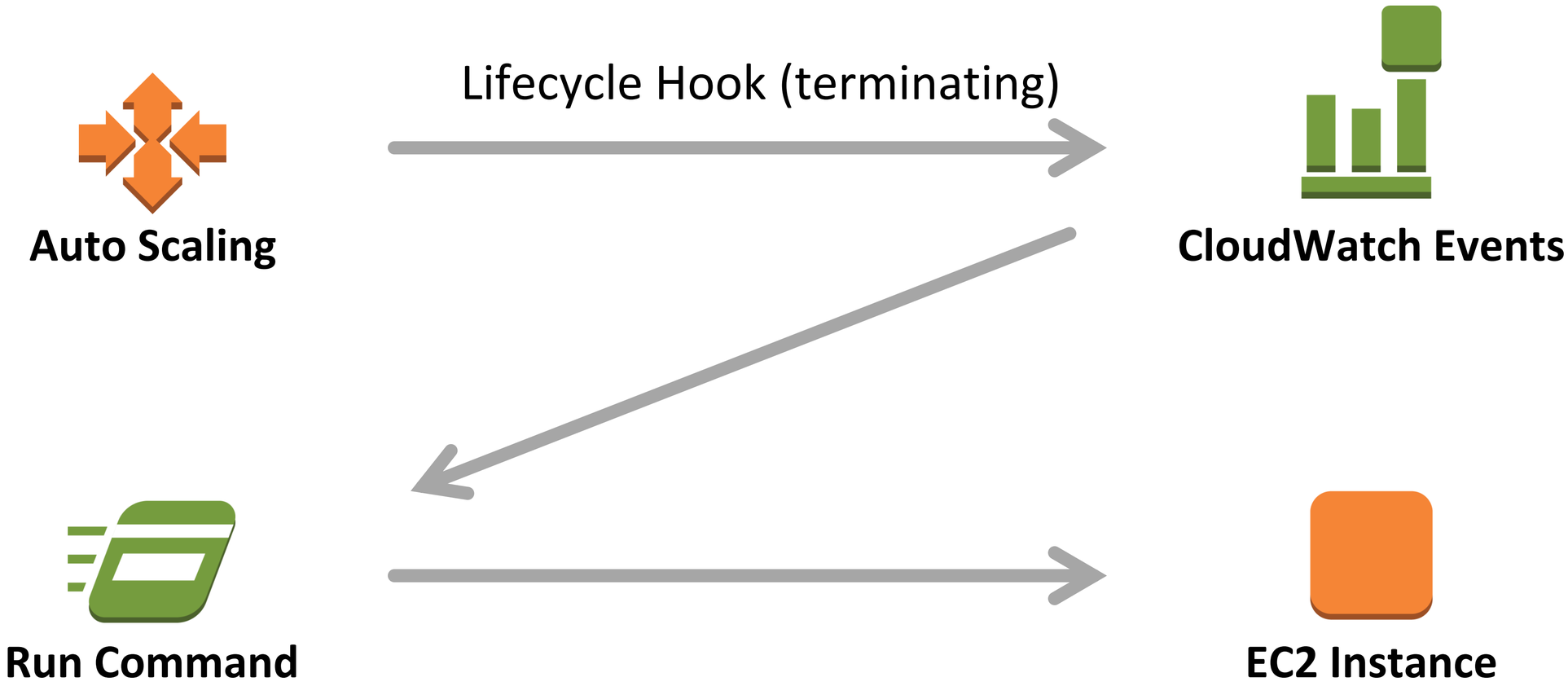
Blue/Green デプロイ [5/6]



Blue/Green デプロイ [6/6]



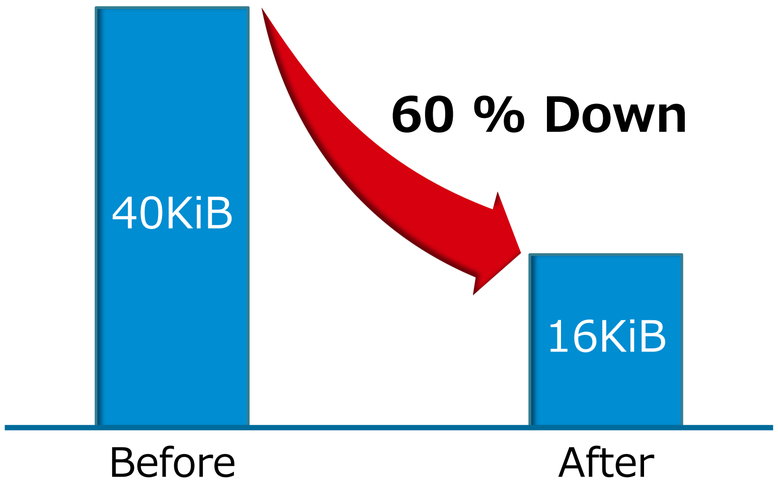
ドリップ処理の自動化


Lifecycle Hook (terminating)
Auto Scaling
CloudWatch Events
Run Command
EC2 Instance

省メモリ化

インフラ費用を抑えたい

- ejabberd を省メモリ化
  - OpenSSL のメモリ削減＆解放
  - hibernate 前にリソースを解放
- r3.large で1台あたり72万接続



ところが

- CPU もメモリも空きがあるのに接続数が増えない

省メモリ化

Security Group の制限を回避

- セッションの上限にあたっていた
- Security Group を無効に

Type	Protocol	Port Range	Source	Description
ALL Traffic	ALL	ALL	0.0.0.0/0	

- 限界まで接続可能に

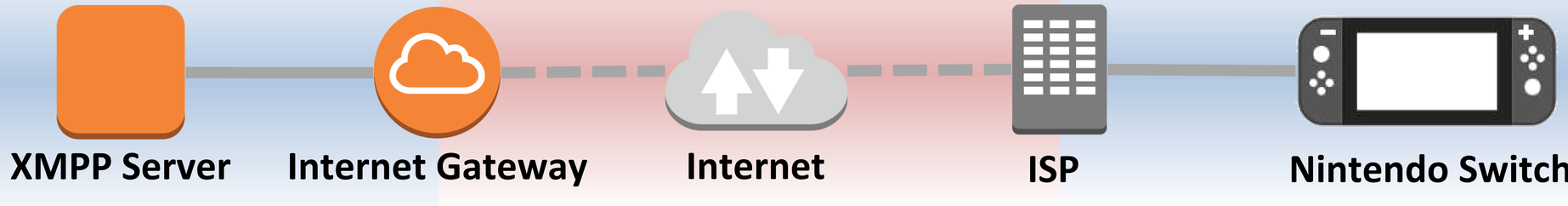
セキュリティ

- 外部アクセスは Network ACLs で制限

TCP 切断対策

途中経路での異常切断を防ぐ

- 無通信期間に KeepAlive (L4, L7)
- 最適間隔を本番環境で実験



AWS内

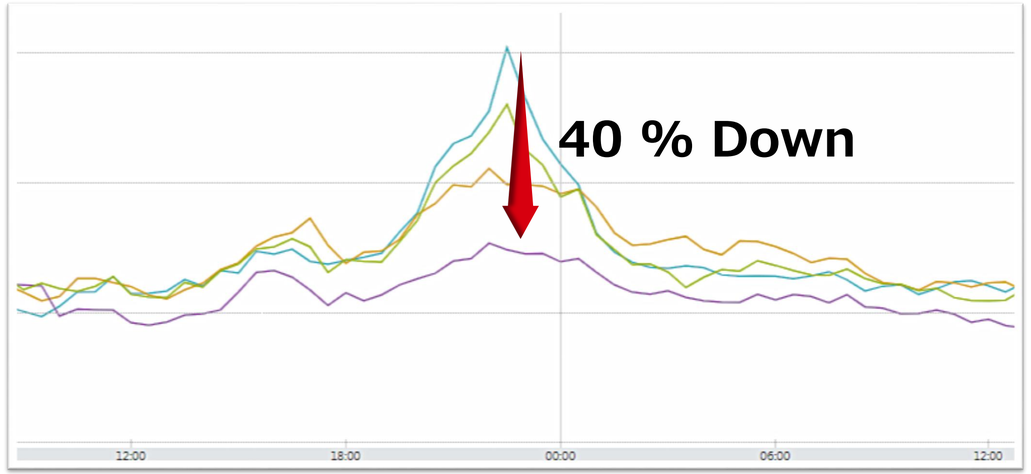
途中経路

ユーザ回線

TCP 切断対策

TCP KeepAlive (L4)

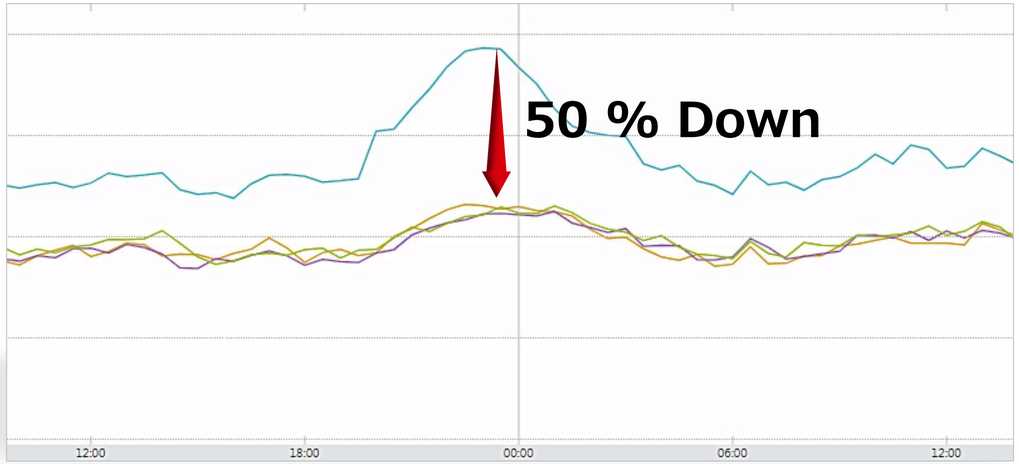
- TCP ヘッダのみの特殊パケットを定期的に送信
- time: ユーザごとに可変
- interval x probes: クラスタ間で対照実験
  - 10 x 2 → 15 x 10
  - 切断を 40% 削減



TCP 切断対策

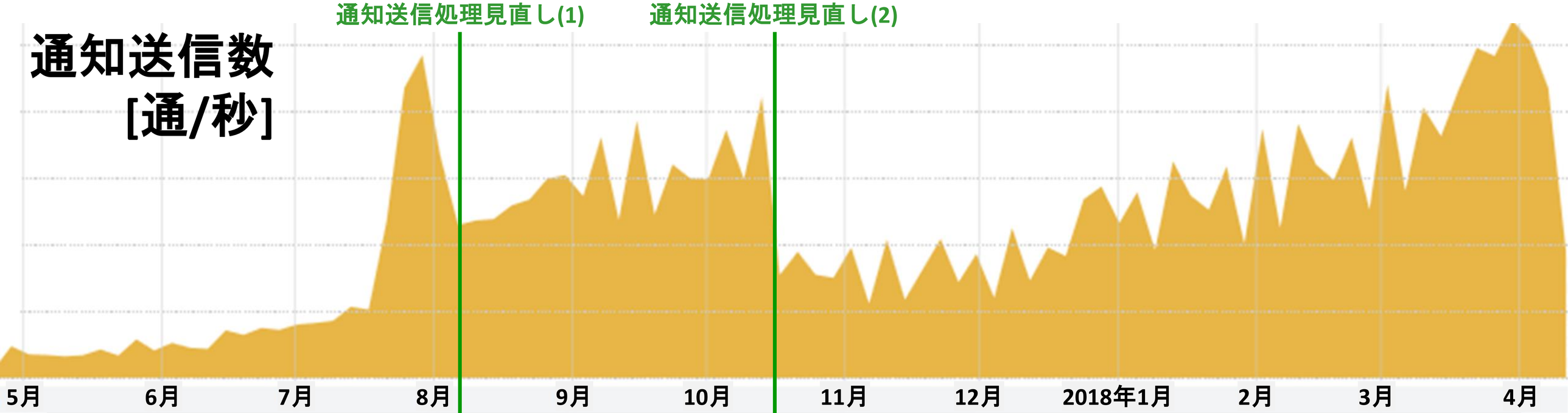
アプリケーション KeepAlive (L7)

- データを入れたパケットを定期的に送信
- 送信間隔: クラスタ間で対照実験
  - 送信無し → 65秒間隔
  - 切断を 50% 削減



振り返り


同時接続数
[台]
5月
6月
7月
8月
9月
10月
11月
12月
2018年1月
2月
3月
4月


通知送信処理見直し(1)
通知送信処理見直し(2)
通知送信数
[通/秒]
5月
6月
7月
8月
9月
10月
11月
12月
2018年1月
2月
3月
4月

規模

約700万　同時接続

約2万　通/秒

約200億　通/月

AWS の異常

発生した異常

- 局所的なネットワーク異常
- RDS I/O 遅延
- EBS 遅延
- EC2 インスタンス異常停止

NPNS サービスへの影響

- AZ 冗長等により、サービスは正常稼動を維持

Erlang での運用

良かった点

- 安定性
  - VM、クラスタ、プロセス監視ツリー
- メモリ効率
  - 軽量プロセス、hibernate
- 並列性
  - プロセス構成にだけは注意
- hot code loading
  - 緊急時に使用

ejabberd 改造内容

並列性の向上

- プロセス構成
  - gen_server 見直し
  - Supervisor 見直し
  - 自分でやる or ワーカー生成
- DB アクセス最少化

安定性・耐障害性の向上

- Mnesia のノード間同期を停止
- DB 切断時に XMPP も早期切断
- 大量切断に耐える

省メモリ化

- Hibernate まわり
- OpenSSL の使い方

ログ出力の詳細化

- ユーザー行動追跡
- CRASH ログで汚さない

独自機能

- 全員に同報通知
- デプロイ用の機能
- TCP 接続維持

まとめと展望

まとめ

Nintendo Switch 向けプッシュ通知システムを構築

必要要件を達成

- スケーラビリティ: XMPP クラスタ分割, Aurora
- 可用性: Auto Scaling, B/G デプロイ, TCP 切断対策
- インフラ費用: 省メモリ化

AWS 起因のサービス停止は無かった

サービス規模は順調に成長

今後の展望

NLB

- Cross-AZ 指定ができるようになった今こそ
- XMPP クラスタロードバランサに利用
- consul を取り外してシンプル化

Fargate

- Consumer/Provider に導入予定

We Are Hiring!

- Webエンジニア (Rails, SpringBoot, ...)
- ネットワークインフラエンジニア (Public Cloud, k8s, ...)
- ネットワークサービスシステムエンジニア
- サーバセキュリティエンジニア

...

「任天堂 キャリア採用」で検索

https://www.nintendo.co.jp/jobs/career/

ご静聴ありがとうございました