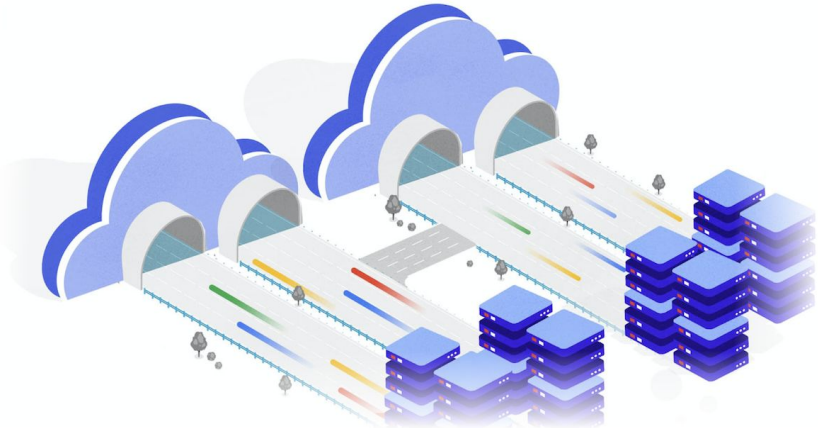
クラウド時代における
インフラ ストラクチャの
高可用性確保のための要点

Google Cloud
Infrastructure Modernization Specialist
片岡 義雅



本日の目標

本日は短い時間となりますので、システムの可用性に焦点を置き、以下の目標が達成できるようポイントを絞ってお話をさせていただきます

- ポイント 1 : システムの特性に応じた構成を検討できるようになること
  - 構成を検討する上で考慮する指標について
  - 可用性に応じた複数の構成例について
- ポイント 2 : システム復旧作業の正確性や作業効率を上げる方法を検討できるようになること
  - 復旧作業の自動化について
  - 復旧作業のテストについて

Agenda

1. オンプレミスとクラウドにおける可用性担保
2. 可用性構成を検討する際の指標 (ポイント 1)
3. 復旧作業の効率化 (ポイント 2)
4. まとめ

オンプレミスとクラウドにおける可用性担保

01

高可用性とは

サーバーの冗長化やバックアップ /リストアなどの手段によってシステムが使えない時間を可能な限り短くすることを目的とする。オンプレミスやクラウドなどの環境の違いによる考え方の違いはない。

システムが使えない時間が長いと企業にも様々な影響を及ぼす

システム停止に伴う
ビジネス停止

機会損失

直接的な影響

信頼の喪失

ユーザー離れ

顧客満足度の低下

間接的な影響

オンプレミスと Google Cloud の比較

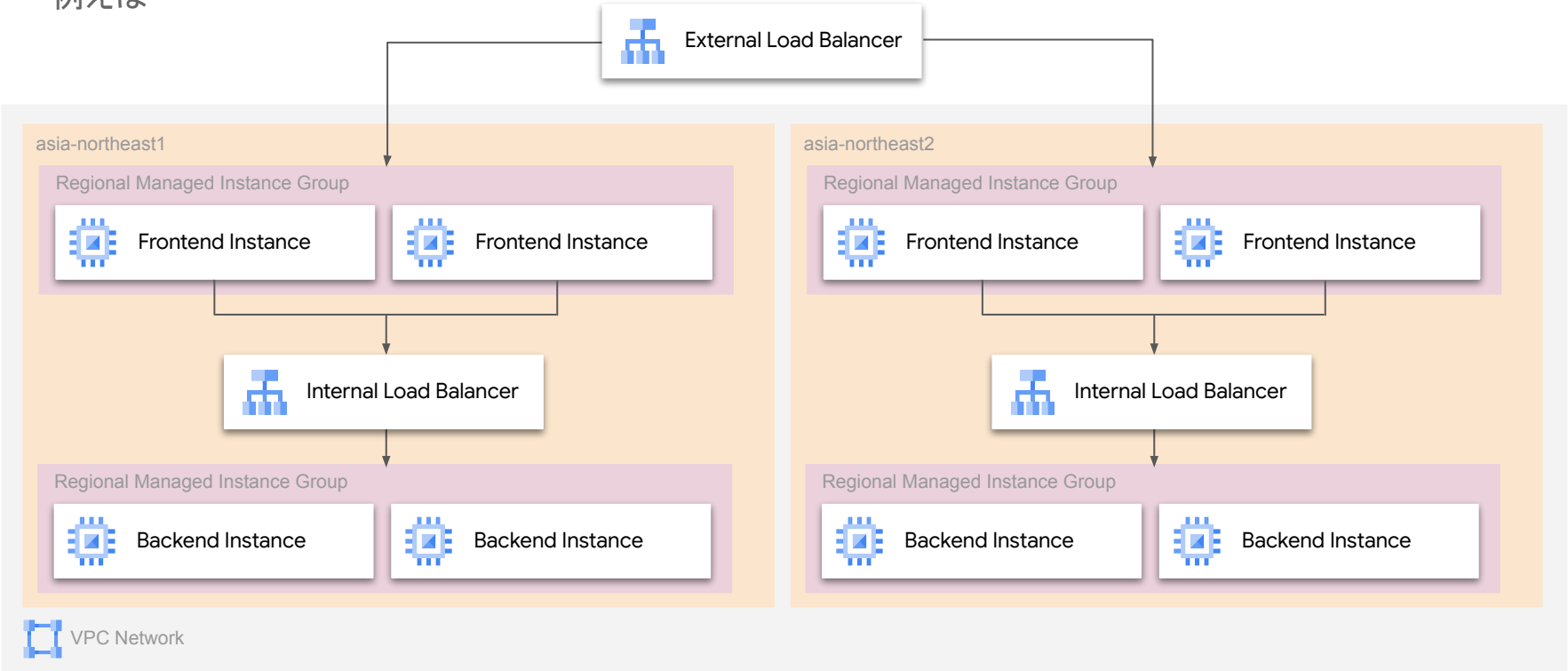
冗長化やバックアップ/リストアはクラウドに限った話ではないが、オンプレミス環境の場合には考慮すべきポイントは多岐に渡る。一方、Google Cloud の場合にはユーザーがワークロードに集中できるための様々な特長がある(Capex/Opex 両方の削減が可能)

	オンプレミス	Google Cloud
コスト	データセンターの賃料、サーバー購入費用、工事費用等は基本的にユーザー負担	初期費用は不要、従量課金制の課金形態によってコストの効率化が可能
スケーラビリティ	リソースを調達するまでのリードタイムが長い	必要なときに必要な分だけのリソースがすぐに、かつ簡単にデプロイ可能
ネットワーク	特に地理的冗長構成を取る場合はネットワーク レイテンシが高い	世界有数の規模を誇る Google のコンピュータ ネットワークを利用した高速通信が可能
セキュリティ	データ保護のみならず物理的なセキュリティ対策やコンプライアンスへの準拠の検討が必要	Gmail や Google Workspace などの Google の他のアプリケーションの運用経験に基づいて構築されたセキュリティ モデルを採用し、第三者による定期的な監査によって各種コンプライアンス規格に準拠

ここで注意点

「クラウドを利用すれば必ず安くなる！」というわけではない

例えば・・・



可用性構成を検討する上でのポイント

システムの可用性を高める上で検討すべき (もしくはした方が良い) 項目は様々存在するが、
本日は以下 2 点に着目する

- **RTO と RPO**
- **復旧作業の自動化とテスト**

その他、以下のような点を検討することをおすすめ

- Cloud Monitoring / Cloud Logging などを用いた監視 /アラート発報の設定
  - システム異常を事前検知することで被害を最小限に留め、かつ事後調査に利用する

- 復旧計画に関する各作業のオーナーの明確化
  - いわゆる “二遊間” 的なグレーゾーンをなくす

- 復旧計画のドキュメント整備 (特に手作業が発生する場合)
  - 作業内容を明確化することでミスを減らす

可用性構成を
検討する際の指標

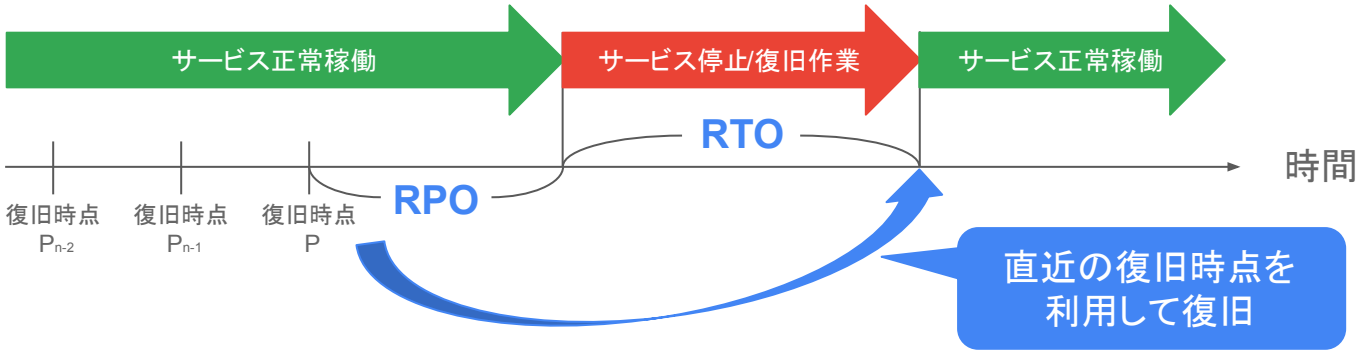
02

RTO と RPO

可用性を高める方法として様々な構成が考えられるクラウド環境において、まず検討すべきはビジネスへの影響の指標となる RTO と RPO

- **RTO (復旧時間目標)**
  - アプリケーションがオフラインである状態が許容される最大時間

- **RPO (復旧時点目標)**
  - 重大なインシデントが原因でアプリケーションからデータが失われている状態が許容される最大時間



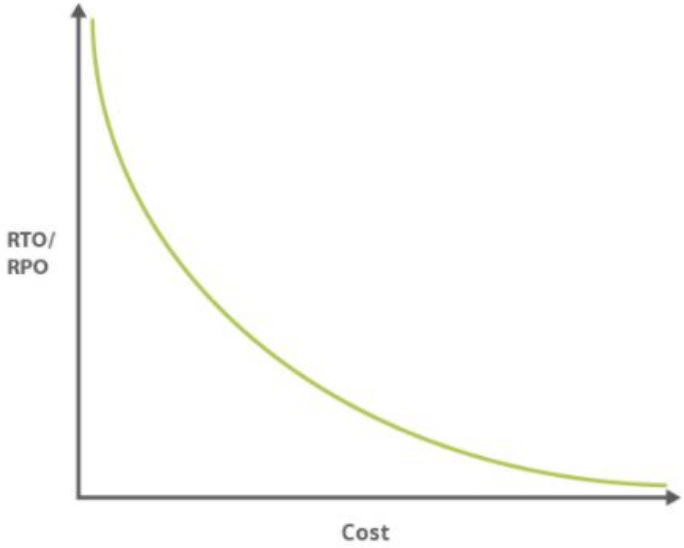
RTO/RPO とコスト

RTO/RPO とコストやシステムの複雑性は反比例するため、ビジネスへの影響を踏まえて必要十分な構成を取ることでコストの最適化を目指す

例 1) 静的コンテンツが多い Web サイト
→ RPO を長く設定することでコスト削減

例 2) ダウンタイムが許容できるイントラネット サーバー
→ RTO を長く設定することでコスト削減

例 3) 業務に欠かせないメール サーバー
→ コストをかけてでも RTO/RPO ともに短く設定



高可用性構成の選択肢

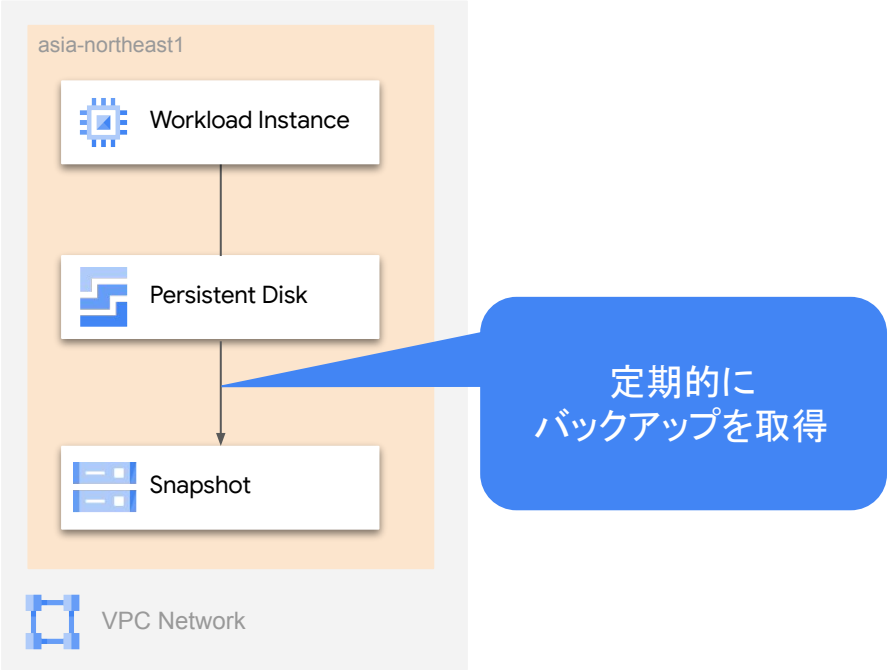
可用性の程度に応じてシステム構成の選択肢が複数存在

構成	用途 (例)	可用性	コスト
バックアップ & リストア	1 日以上の停止が許容可能な 社内ブログ システム	低	低
パイロット ライト	1 日数回程度実行される バッチ ジョブ システム		
ウォーム スタンバイ	業務内容の管理を行う ERP/CRM システム		
マルチサイト	ミッション クリティカルな 株取引システム	高	高

バックアップ & リストア

定期的にバックアップを取得して障害時にリストアする

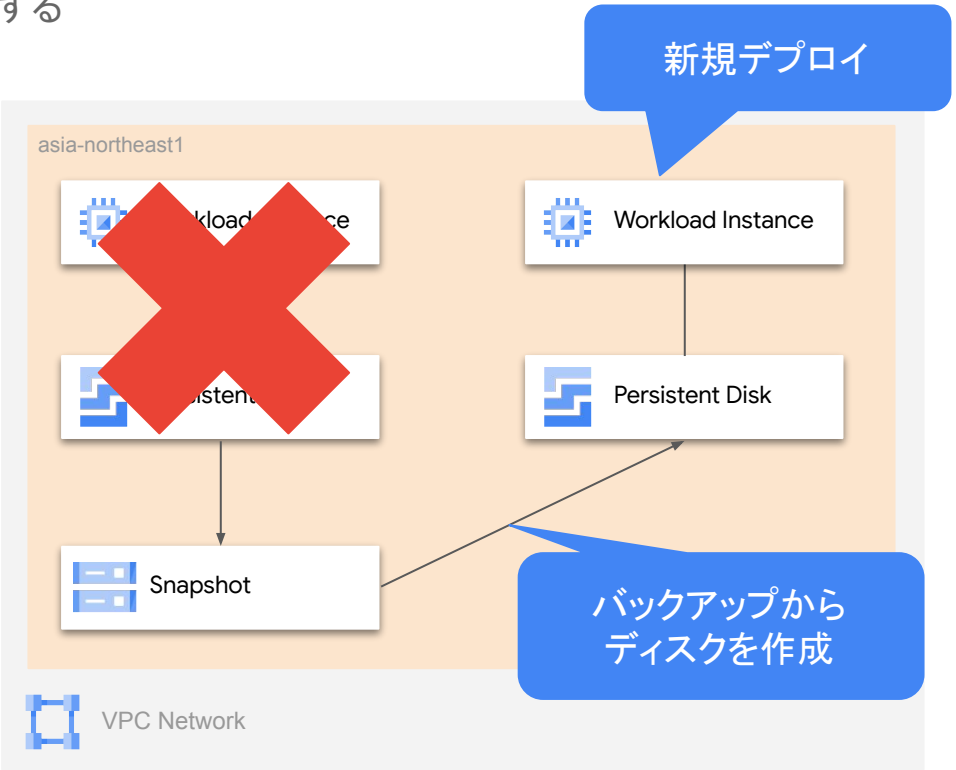
- 最もシンプルで低コストな構成
- RPO はバックアップ頻度に依存
- データ容量やシステムの構成要素に応じて RTO が長くなる



バックアップ & リストア

定期的にバックアップを取得して障害時にリストアする

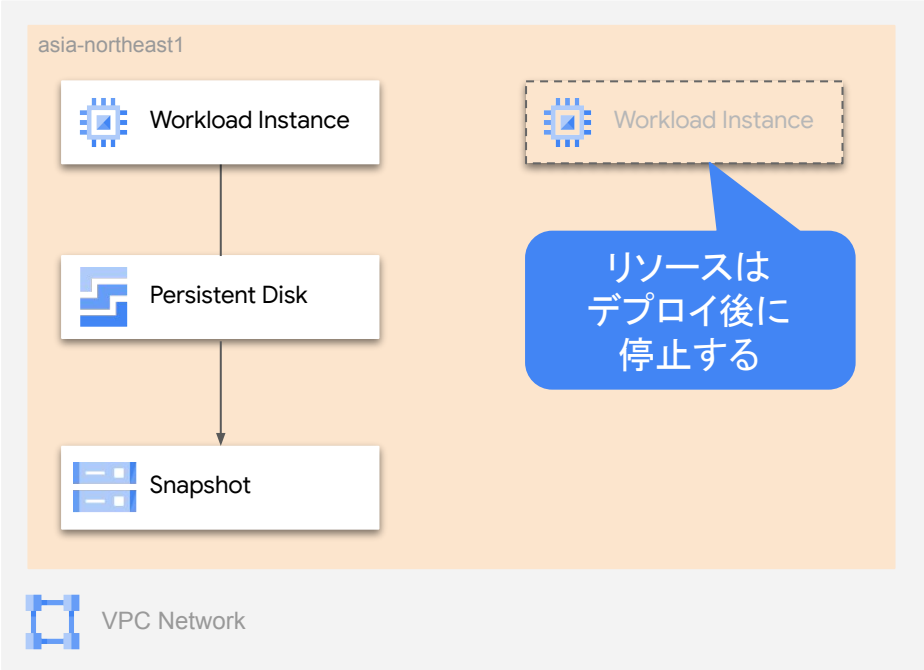
- 最もシンプルで低コストな構成
- RPO はバックアップ頻度に依存
- データ容量やシステムの構成要素に応じて RTO が長くなる



パイロット ライト

予め別環境を用意 (平常時は停止) し、必要なタイミングで起動する

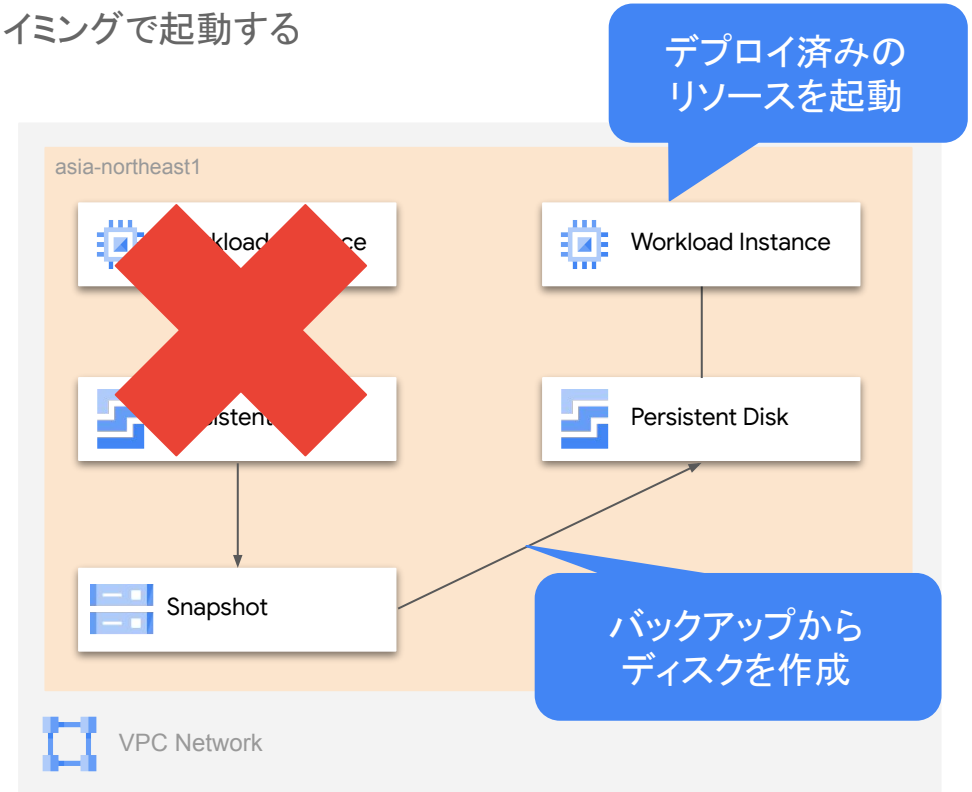
- リソースを停止することで課金を抑える
- 停止できないリソースは継続課金
- リソースの新規デプロイが不要なため、リストアと比較して RTO が短い (複雑な構成であればあるほどリストアと比較して RTO が短い)



パイロット ライト

予め別環境を用意 (平常時は停止) し、必要なタイミングで起動する

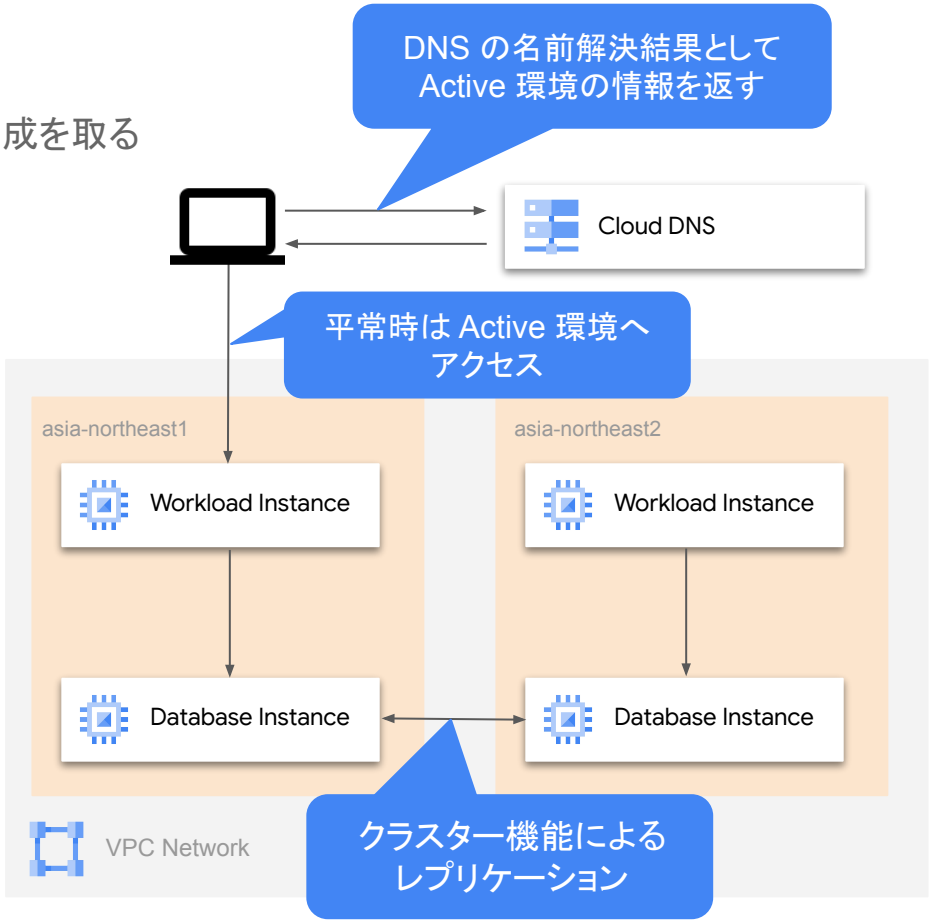
- リソースを停止することで課金を抑える
- 停止できないリソースは継続課金
- リソースの新規デプロイが不要なため、リストアと比較して RTO が短い (複雑な構成であればあるほどリストアと比較して RTO が短い)



ウォーム スタンバイ

複数のリージョン (ゾーン) で Active - Standby 構成を取る

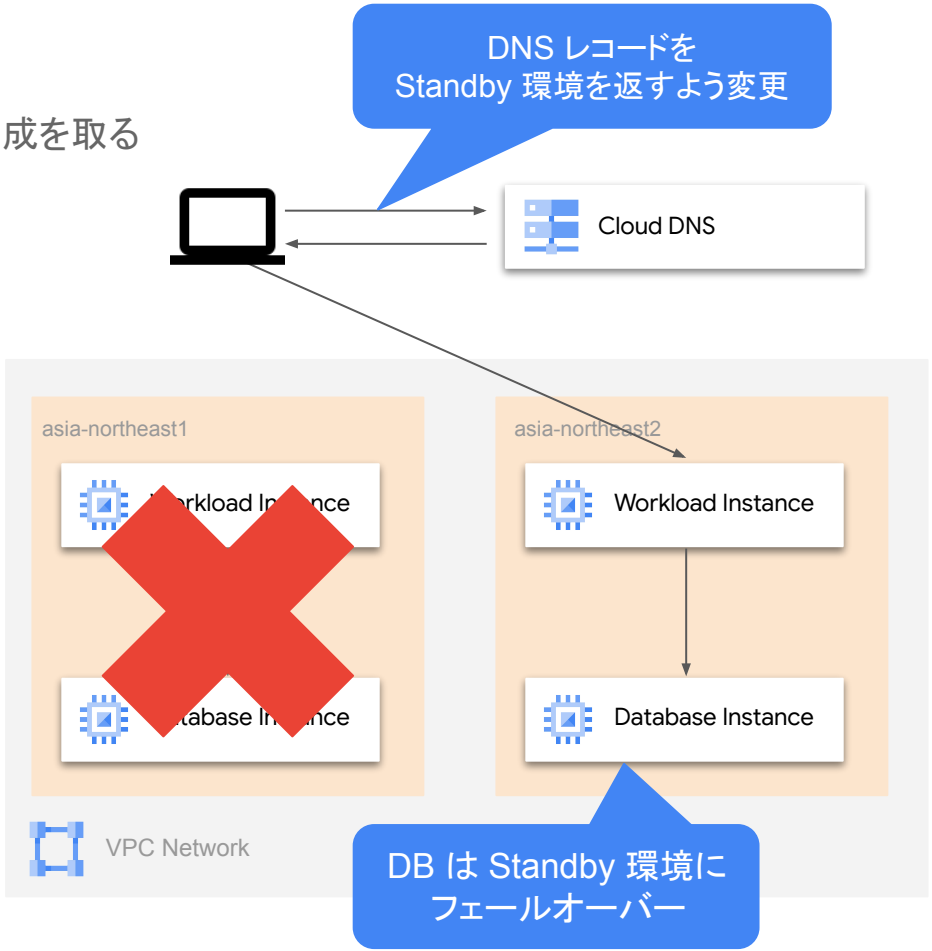
- 継続的なレプリケーションによって RPO が低くなる
- DNS サーバーのレコードは Active 環境を向き、基本的には Active 環境にアクセス (Standby 環境も稼働はしている状態)
- Standby 環境のリソース スペックを Active 環境より低くすることでコストの削減が可能
- クラスター機能等による自動フェールオーバー



ウォーム スタンバイ

複数のリージョン (ゾーン) で Active - Standby 構成を取る

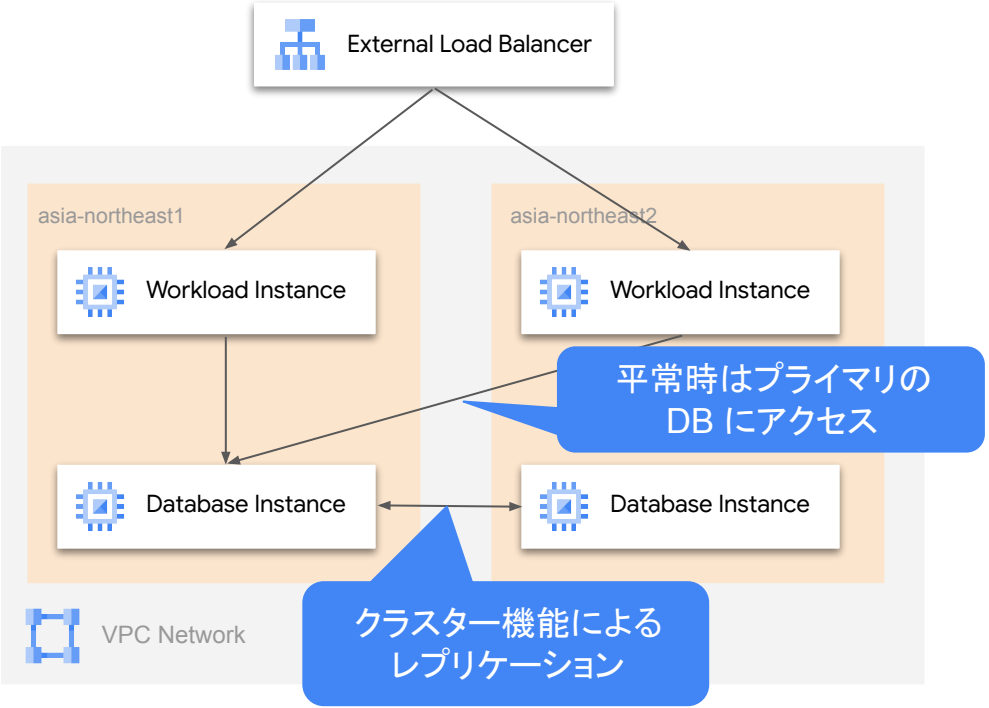
- 継続的なレプリケーションによって RPO が低くなる
- DNS サーバーのレコードは Active 環境を向き、基本的には Active 環境にアクセス (Standby 環境も稼働はしている状態)
- Standby 環境のリソース スペックを Active 環境より低くすることでコストの削減が可能
- クラスター機能等による自動フェールオーバー



マルチサイト

複数のリージョン (ゾーン) で Active - Active 構成を取る

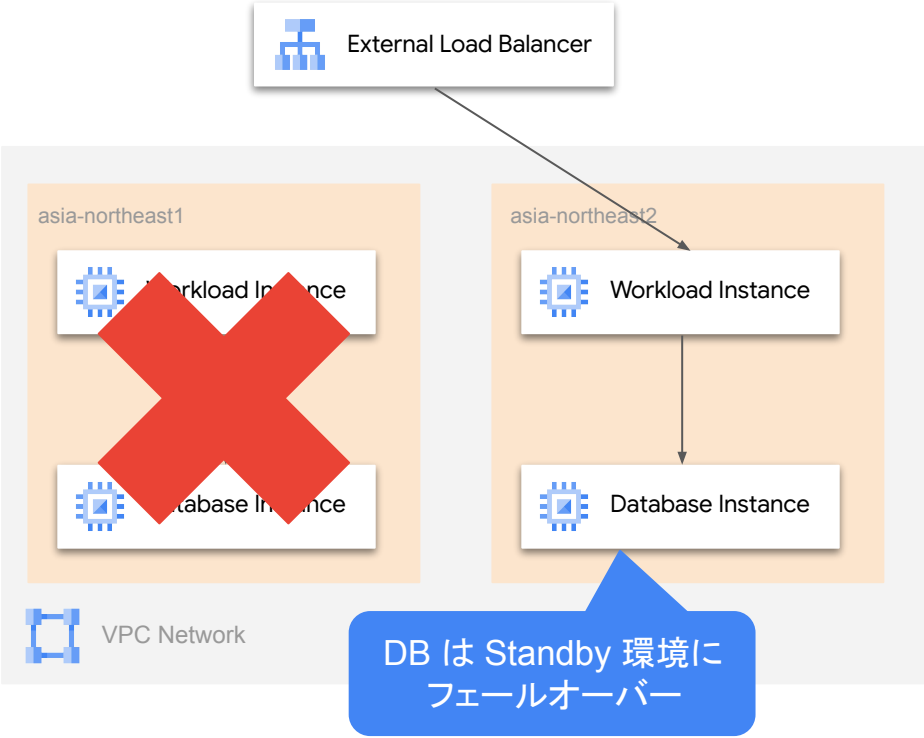
- 複数の環境が必要となるため、最も高コストな構成
- クラスター等の設定に依存するが、基本的には RTO/RPO ともになし
- クラスター機能等による自動フェールオーバー



マルチサイト

複数のリージョン (ゾーン) で Active - Active 構成を取る

- 複数の環境が必要となるため、最も高コストな構成
- クラスター等の設定に依存するが、基本的には RTO/RPO ともになし
- クラスター機能等による自動フェールオーバー



RTO/RPO とシステム構成の指標

RTO や RPO はビジネス影響を鑑みて策定する必要があるが、ざっくりと以下のようなイメージで考えると良い

構成	バックアップ & リストア	パイロット ライト	ウォーム スタンバイ	マルチサイト
RTO	1 日 ~ (※1)	~ 1 日 (※1)	~ 15 分 (※1)	~ 数秒
RPO	バックアップ頻度に依存	バックアップ頻度に依存	基本なし (※2)	基本なし

※1: 構成や復旧作業の内容、運用体制に大きく依存
※2: レプリケーションの仕組みに依存

ここまでのまとめ

本日の目標のポイント 1 を振り返ります

- ポイント 1 : システムの特性に応じた構成を検討できるようになること
  - ビジネス要件に則った RTO や RPO を定める
  - バックアップ & リストア、パイロット ライト、ウォーム スタンバイ、マルチサイトという選択肢
- ポイント 2 : システム復旧作業の正確性や作業効率を上げる方法を検討できるようになること
  - 復旧作業の自動化について
  - 復旧作業のテストについて

復旧作業の効率化

03

障害発生からの流れ

障害発生後には以下の作業が必要となる（マルチサイト構成の場合は環境切替などは不要だが、障害が発生した環境の調査や復旧作業は必要）

障害発生

- リソース障害
- リージョン/ゾーン障害
- アプリケーション障害

復旧作業

- 新規リソース デプロイ
  - GCE
  - Persistent Disk
- データ リストア
  - バックアップからのリストア

動作確認

- 新規環境の動作確認
  - データベースへの接続
  - データの整合性

環境切替

- DNS レコードの切替

実際の復旧作業は複雑

例えばバックアップ & リストアの構成を取る場合、新たなリソースのデプロイだけでも少なくとも以下のような作業が発生する (ワークロードが GCE の MIG で稼働している場合)

1. スナップショットから GCE のカスタム イメージを作成
2. カスタム イメージからイメージ テンプレートを作成
3. イメージ テンプレートからマネージド インスタンス グループ (MIG) を作成
4. MIG を Cloud Load Balancer のバックエンド サービスに設定
5. 再びの障害に備えて再度バックアップを取得するよう設定

Google Cloud

復旧作業の自動化

Cloud Deployment Manager を利用してリソース デプロイなどの作業を可能な限り自動化することで復旧作業のミスを減らす

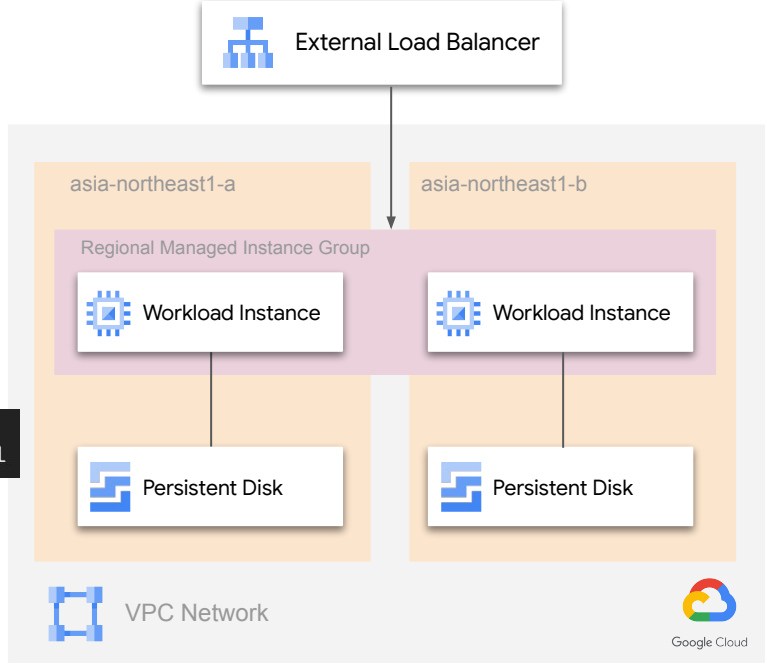
- クラウド管理を簡素化
  システムの構成要素を yaml で定義可能
- 再現可能なデプロイ プロセス
  コードでインフラストラクチャ構成を定義することで必ず同じ構成でリソースがデプロイ可能
- 宣言型言語
  システムの “あるべき姿” を宣言するだけでよく、構成の手順を気にする必要がない (リソース間の依存関係を自動で認識)

Cloud Deployment Manager の利用イメージ

サンプル: https://cloud.google.com/deployment-manager/docs/reference/cloud-foundation-toolkit

```
resources:
- name: myInstanceTemplate
  type: compute.v1.instanceTemplate
  properties:
    diskImage: customDiskImage

- name: myMig
  type: compute.v1.regionInstanceGroupManager
  properties:

- name: myService
  type: compute.v1.backendService

- name: myForwardingRule
  type: compute.v1.forwardingRule

- name: myurlMap
  type: compute.v1.urlMap

- name: myVPC
  type: compute.v1.network

:
```

```
$ gcloud deployment-manager deployments \
    create deployment --config config.yaml
```



復旧作業のテスト

策定した復旧計画や自動化用テンプレートが期待した動作となるかを確認するため、定期的なテストを実施することをおすすめ

- Cloud Deployment Manager のテスト
  - テンプレートが期待した通りにリソースをデプロイするかのテスト

- 障害挿入による復旧計画のテスト
  - VM のシャットダウン、プロセス強制停止などのシナリオをシミュレートする
  - データベースなど、クラスター構成となっているミドルウェア等が正常にフェールオーバーするかを確認する

- ゾーン障害やメンテナンスのテスト
  - クラウド特有の概念であるゾーンの障害やメンテナンス イベント時の動作をシミュレートできていると良い
    - ゾーン障害をシミュレートするためのサンプル スクリプトはこちら
    - メンテナンス イベントのシミュレート方法は こちら

ここまでのまとめ

本日の目標のポイント 2 を振り返ります

- ポイント 1 : システムの特性に応じた構成を検討できるようになること
  - ビジネス要件に則った RTO や RPO を定める
  - バックアップ & リストア、パイロット ライト、ウォーム スタンバイ、マルチサイトという選択肢
- ポイント 2 : システム復旧作業の正確性や作業効率を上げる方法を検討できるようになること
  - Cloud Deployment Manager を利用した作業の自動化
  - 様々なシナリオを想定した復旧計画の動作確認
  -

まとめ

04

まとめ

1. システムの可用性はビジネスにダイレクトに影響を及ぼす
2. インフラとしての Google Cloud のメリットを享受する
3. ビジネス影響をシステム構成決定時の判断基準とする
4. 自動化、テストによっていざというときに備える

参考情報

本日お話した内容に関する参考情報は以下の通りです

- 障害復旧計画ガイド
  https://cloud.google.com/solutions/dr-scenarios-planning-guide

- 高可用性を確保する為のデザイン パターン
  https://events.withgoogle.com/solution-design-pattern-infra-rdb-network/high-availability/#content

また、こちらのブログ記事もおすすめです

- SRE チームの評価に役立つレベル別チェック リスト
  https://cloud.google.com/blog/ja/products/gcp/how-to-start-and-assess-your-sre-journey

Thank you