クラウドのサービスレベルを上げる運用管理

フォースクーナ株式会社
代表取締役
三角　正樹

会社紹介

フォースクーナ株式会社

　2003年設立　9年目

　システム運用サービスを中心にインテグレーションサービスを提供している。

概要

1. パブリッククラウドのメリットとサービスレベル
2. パブリッククラウドの障害事例
3. パブリッククラウドを利用していたサービス事業者はどうしたか
4. 障害に対して、何が必要か

パブリッククラウドのメリット

- すぐに使えて、すぐに止められる
- 資産ではなく、損金
- スケーラビリティを持てる
- 再構築の容易さ
- 運用を任せられる

パブリッククラウドのサービスレベル

- nifty cloud

  月間のサーバー稼働率　　99.95%

- AWS EC2

  年間使用可能時間割合　　99.95%

障害可能時間

nifty cloud　　月間21分以内

EC2　　年間4.32時間以内

良くある障害

- インスタンスが固まってしまう
- ネットワーク遅延

現在、パブリッククラウドを使用されている方？

この中で、パブリッククラウド側の障害にあった事があるという方はどのくらいいらっしゃいますか？

残念な事に、、

パブリッククラウド側のオペミスでインスタンスが飛んでしまう事や、長時間のネットワーク障害が発生したり、SLA以上の障害が起きる可能性もあります。

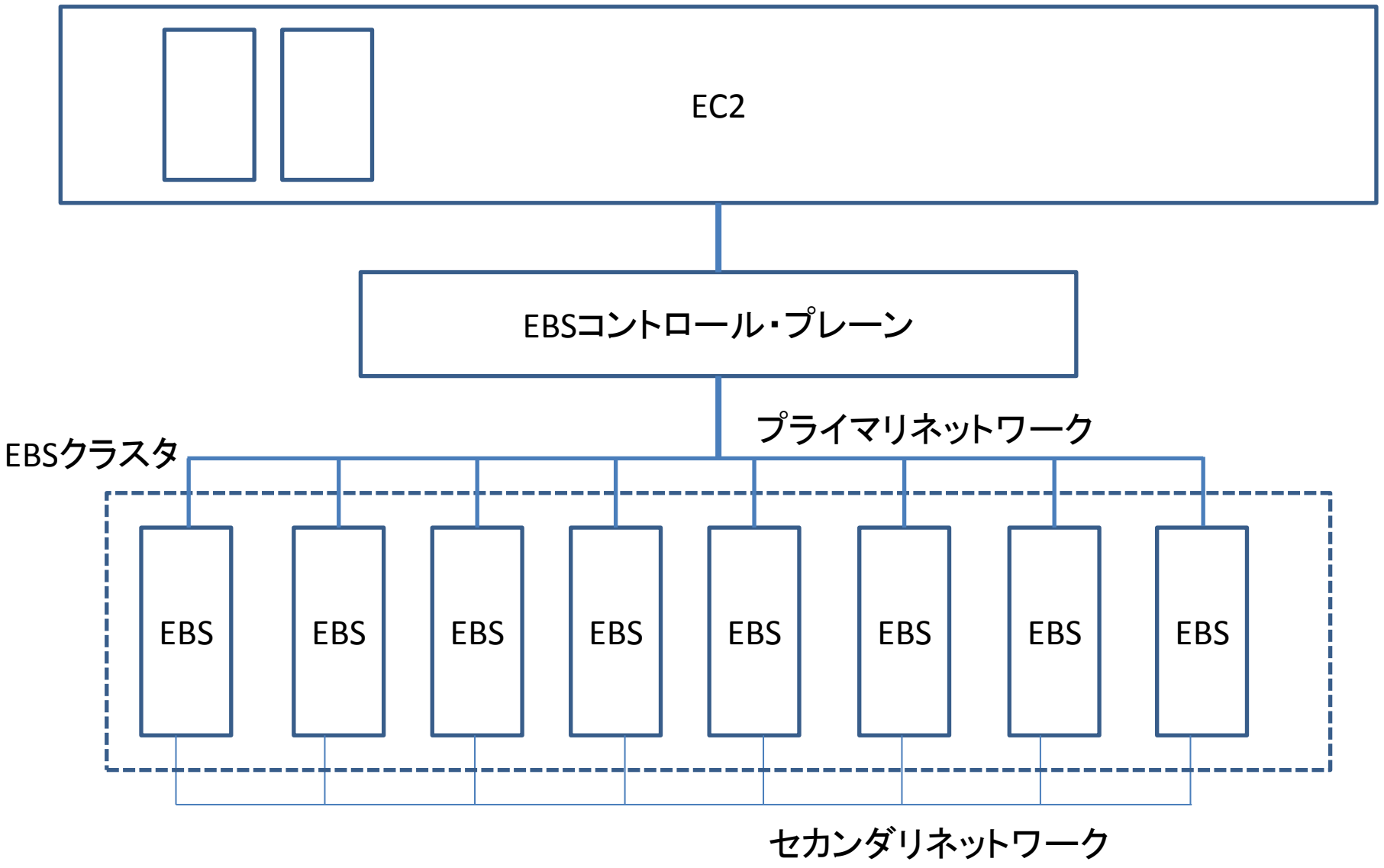
障害事例の研究が必要

2011年4月21日に起きたAWSでの障害

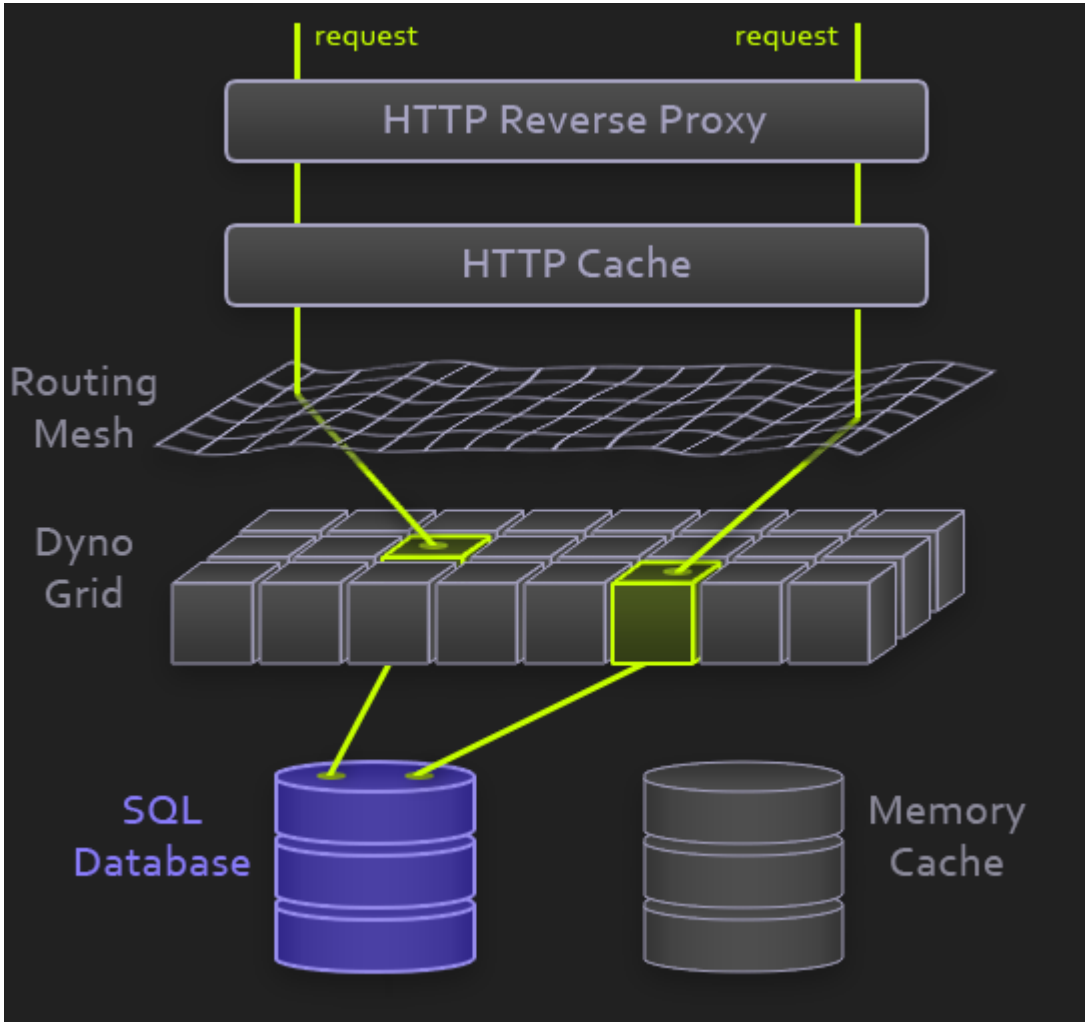
- 米国時間4月21日にAmazon Web Servicesで、ネットワークの構成変更作業におけるミスを発端として、ブロックストレージサービス「Amazon Elastic Block Store（EBS）」および、リレーショナルデータベースサービスの「Amazon Relational Database Service（RDS）」が約4日間にわたる障害が発生。

米国東リージョンにおけるAmazon EC2とAmazon RDSのサービス障害の概要
http://aws.amazon.com/jp/messages/65648/

AWS構成（イメージ）


EC2
EBSコントロール・プレーン
プライマリネットワーク
EBSクラスタ
EBS
EBS
EBS
EBS
EBS
EBS
EBS
EBS
セカンダリネットワーク

Herokuとは



AWSの上にRuby, Rails, PostgreSQL、memcachedなどオープンな技術を用いて、作られたPaaSを提供している会社。

2011年1月にセールスフォースに買収され、現在はnode.jsなど他の言語対応も進めている。

2011年4月21日に起きたAWSでの障害 ~Herokuの例～

- **Herokuを4年間運用してきて最大の障害**
- **専用データベースを利用している大規模アプリケーションでは最大16時間のダウンタイム**
- **共有データベースを利用している小規模アプリケーションでは最大60時間のダウンタイム**
- **アプリケーションのデプロイについてはプラットフォームの広範囲にわたり約76時間（3日間以上）の間不可能となった**

Herokuのステータスページ
http://status.heroku.com/incident/151

障害の発生

4月21日　午前0時47分（PDT）　AWSでの障害発生

午前1時15分（PDT）　監視システムからの警告が届く
ステータスページに問題発生の
一報を掲載

Issue: We are investigating high error rates. We'll post an update when we know more.

ポイント

- **AWSでの障害発生から検知まで28分。ちょっと遅く感じるが、ステータスページへの反映が同時行われているので、検知はもっと早かったのではないか。**
- **第一報の掲載が本当に早い（障害発生から30分以内）**

原因の調査と再構築

4月21日　午前1時52分（PDT）

**広範囲に渡るネットワークエラーによって、
Webやキャッシュ、ルーティングサービスがタイムアウトしていることを確認**

Update: The elevated error rates are due to connectivity issues. We're continuing to work with our network service provider to fully restore connectivity. Applications and tools are working intermittently at this time.

最初の数時間は、挙動のおかしいインスタンスをシャットダウンして新しいインスタンスに置き換える作業を試みた。Herokuのプラットフォームは通常であればこの方法で復旧し、ユーザーにとってはごく小さい障害で済む。

原因の調査と再構築

しかし、今回は事態が悪くなっていっていることがわかった。
最大の問題は使用してるEBSドライブだった。

ポイント

- パブリッククラウドは問題が発生した場合、通常は問題を修復するより、インスタンスを入れ替えるなど再構築した方が良い。
- EBSが問題と特定できた。

障害対応

EBSドライブがどんどん予測できない挙動を示しはじめた。一部のドライブは、新しいインスタンスに接続し直しても完全に反応しなくなってしまった

対処方法を聞くためAWSのテクニカルアカウントマネージャーと1日中連絡をとっていたが、残念ながらそれらも役には立たず、障害はどんどん拡大していった。これまでの前例から見て、このような場合に取るべき最善の方法は、サービスを動かしつづけるようベストを尽くしつつAWSが問題を解決するのを待つことだ 。

ポイント

- 問題は見えてきたものの、出来ることは待つだけだった。

障害対応

16時間以内に大部分のアプリケーションが復旧したが、いくつかのサーバーには問題が発生していた

それからの48時間は、エンジニアがAWSと共に可能な限り早くサービスを復旧させることに費やされた 。EBSディスクが応答を返し始めるとともにサーバーが徐々に稼働し始めるのを、ゆっくりではあるが確認できた。

ポイント

- 重要顧客のシステムはバックアップから、別リージョンにリストアし、AWSの障害復旧よりも早く復旧した模様
- プロバイダー側の問題が解決してからが勝負！

パブリッククラウドの障害に対して

落ちないシステムはない

障害は確実に起こる事を前提とし、
障害が発生したときにどれだけ早く復旧させるか

起こってしまうと出来る事が少ないので、事前に手を打っておくことが肝心

- システム的な対処を施しておく
  - SPOF（単一障害点）を無くす
  - HA,DR等の準備
  - Design for Failure（アプリケーションで可用性を担保する）
- 体制や人的な対処を施しておく
  - 障害対応時間の短縮
  - 情報共有・改善

障害検知

検知が重要。如何に正しく早く検知するか。
ここをミスすると復旧が遅れるし、先にユーザーから指摘されるという事も。

ユーザー視点で障害が発生していないか検知できるようにしておく。httpのステータスコードだけ見ていて、ページが真っ白になっていても気が付かないなんてこともある。

検知しようとしすぎてFalse alert(障害ではないトリガによるアラート)が沢山あがると、本当の障害が埋もれるので、閾値の調整や関連性を踏まえてアラートを上げるシステムが必要。

障害対応

パブリッククラウドの障害対応の基本は、修復ではなく再構築。
（インスタンスのstop、start。スナップショットからの立ち上げなど）

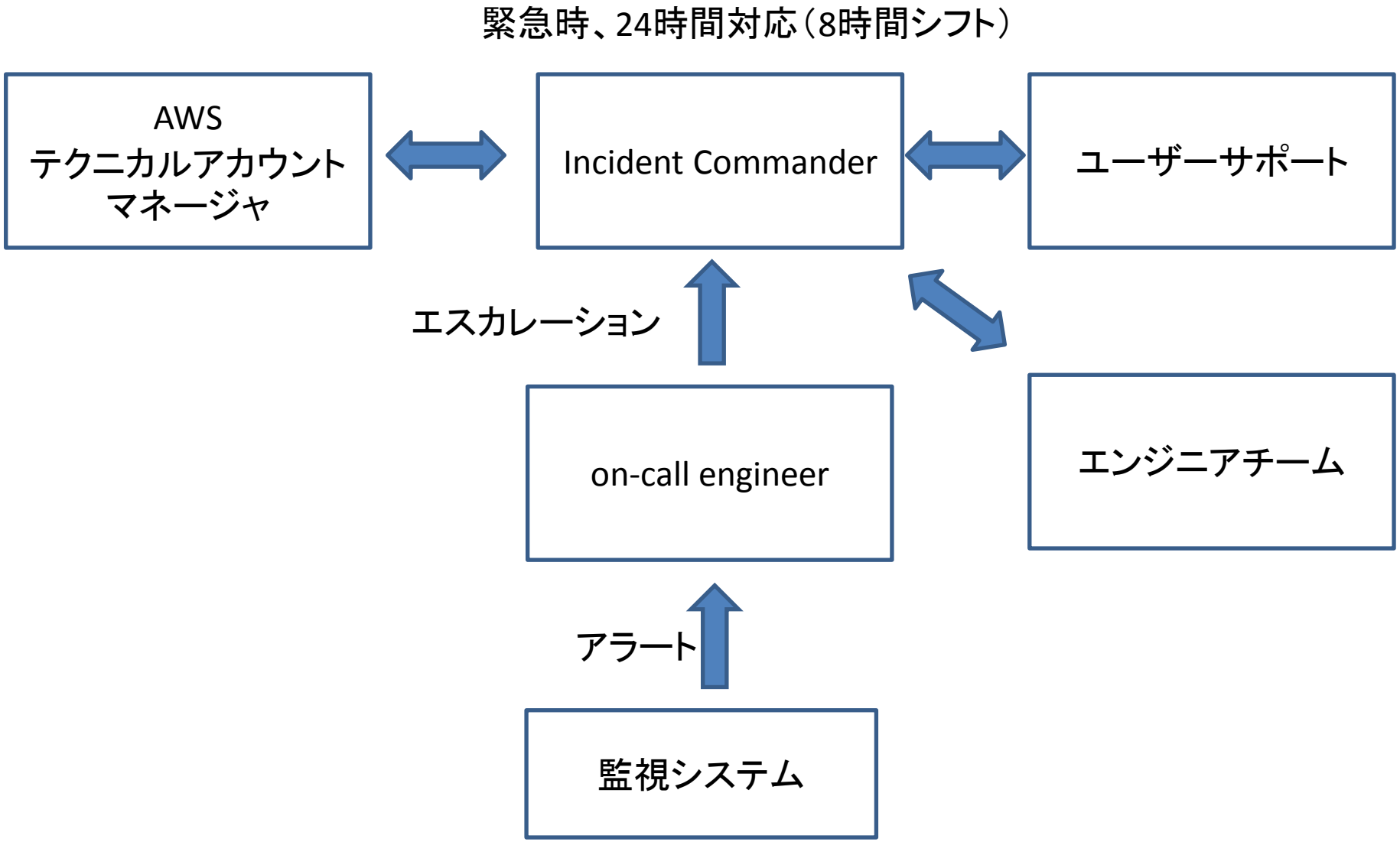
勘所がわかるエンジニアが必要。
　クラウドインフラをわかっているエンジニア
　（プライベートクラウドよりは楽なはず。。）
　自社内にいない場合は、そのようなエンジニアがいる会社と組む

情報共有

原因追究をしていると、作業者にとってはあっという間に時間が立つが待っている側は非常に長い時間に感じられる為、作業者の状況を小まめに吸い上げる仕組みが必要。

ユーザーとも作業進捗を共有する必要がある。稼働率100%のシステムは世の中に無いので、あいまいにせずに誠実な報告をする必要がある。ここの対応次第でユーザーとの信頼関係に差がつく。

Herokuの体制


緊急時、24時間対応（8時間シフト）
AWS
テクニカルアカウント
マネージャ
Incident Commander
ユーザーサポート
エスカレーション
on-call engineer
エンジニアチーム
アラート
監視システム

Herokuの考察

- 障害検知
  監視システムは、障害を素早く検知した

- 緊急障害の体制
  on-call engineerは素早く問題の大きさを判断し、Incident Commanderを起こすことが出来た。Incident CommanderはAWSにすぐに連絡をし、内部の他のエンジニアを起こし、緊急体制を確立できた

Herokuの考察

- 障害対応
  金額規模が大きい顧客から対応をはじめ、それらの顧客のシステムは障害を短く収める事が出来た。

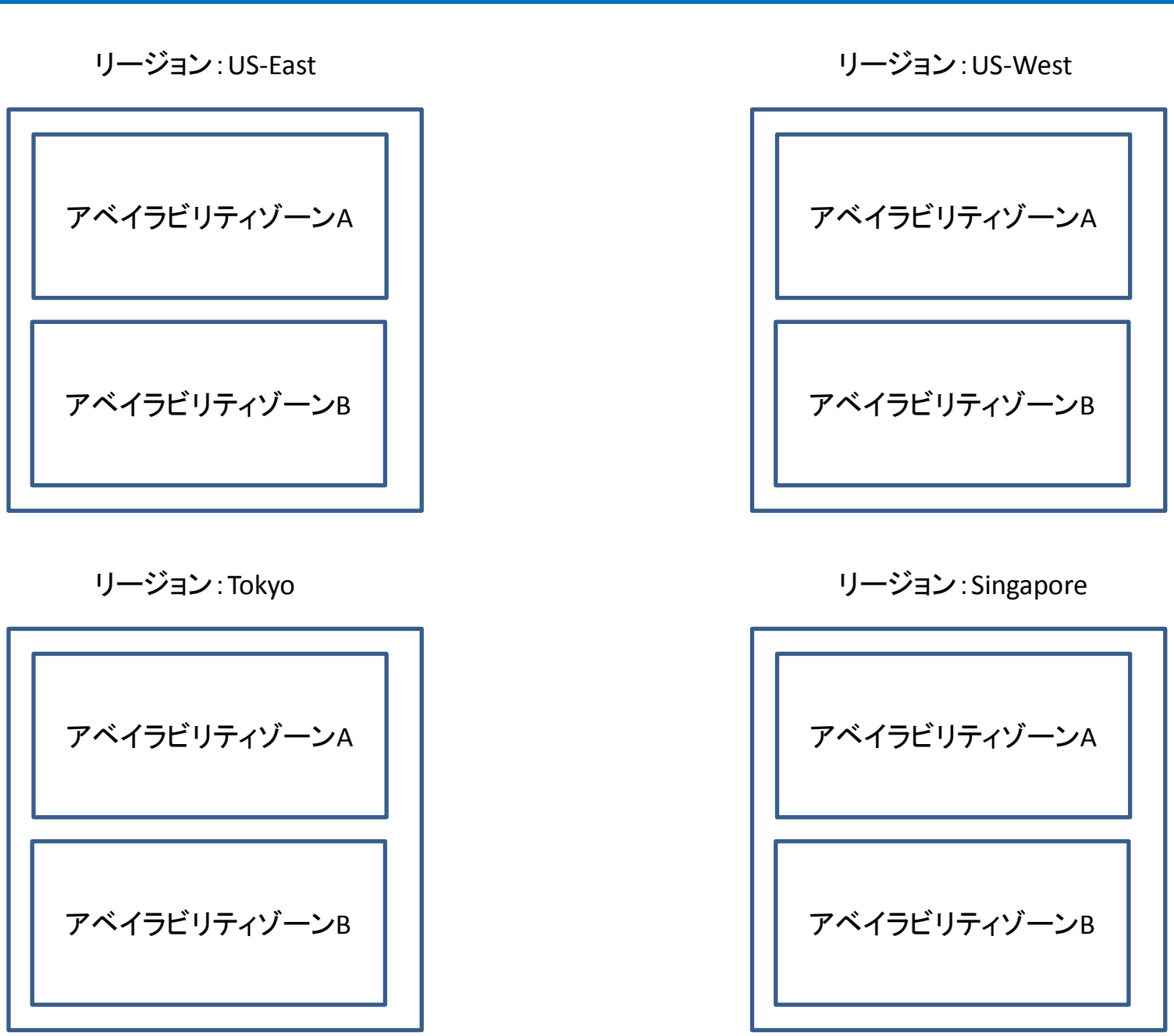
- 情報共有
  何人かから障害対応状況の情報が足りないという指摘を受けたが、バックアップからデータをリストアしたり、システムのリプレースを行っていたので、1時間くらいの間隔では大きな進捗が無い場合もある。障害対応状況の更新により、我々が全力を尽くしているという事は伝わった。

Herokuの考察

- 改善点

  IaaSのレイヤーの障害だったとはいえ、顧客に対してこれらの心配を取り除くすべての責任はHerokuにある為、下記の3点を検討している

- マルチリージョン対応
- EBSの影響を受けにくくする
- リアルタイムバックアップの導入


リージョン：US-East
アベイラビリティゾーンA
アベイラビリティゾーンB
リージョン：US-West
アベイラビリティゾーンA
アベイラビリティゾーンB
リージョン：Tokyo
アベイラビリティゾーンA
アベイラビリティゾーンB
リージョン：Singapore
アベイラビリティゾーンA
アベイラビリティゾーンB

Herokuの考察

- システム的な対処を施しておく　**改善点あり**
  - SPOF（単一障害点）を無くす
  - HA,DR等の準備
  - Design for Failure（アプリケーションで可用性を担保する）
- 体制や人的な対処を施しておく　**問題なし**
  - 障害対応時間の短縮
  - 情報共有・改善

まとめ

- 落ちないシステムはない
- 障害が発生したときにどれだけ早く復旧させるかが肝心
- パブリッククラウド側で障害は、打つ手が少ない
- 事前にシステム的な備えと、人的・体制的な備えが必要

まとめ

- **障害対応時間短縮の為に、早く検知できるようにする**
- **勘所のわかるエンジニアを普段から意識して育てておく**
- **障害時のユーザー報告は素早く、細かく行う**

クラウドのサービスレベル以上のサービスを提供できるように
頑張っていきましょう！