モバイルネットワークをエミュレーションする検証環境をAWSに構築した話

日本電信電話株式会社 (NTT)
ネットワークサービスシステム研究所 (NS研)
ネットワーク基盤技術研究プロジェクト（ネ技P）
サービス処理基盤研究グループ（技サG）

鳥井　隆史

自己紹介

名前	鳥井 隆史	
年次	4年目（R3入社）	
経歴	NTT ME（出向）	埼玉NWSC（北浦和）
		BSC（さいたま新都心）
	日本電信電話（転籍）	NS研（三鷹）
資格	AWSx13 LinuC3 ネスペ等	
心持ち	見た目年齢 ＜ 実年齢 ＜ 精神年齢	
趣味	筋トレ、カメラ	

電話局の保守
DX推進

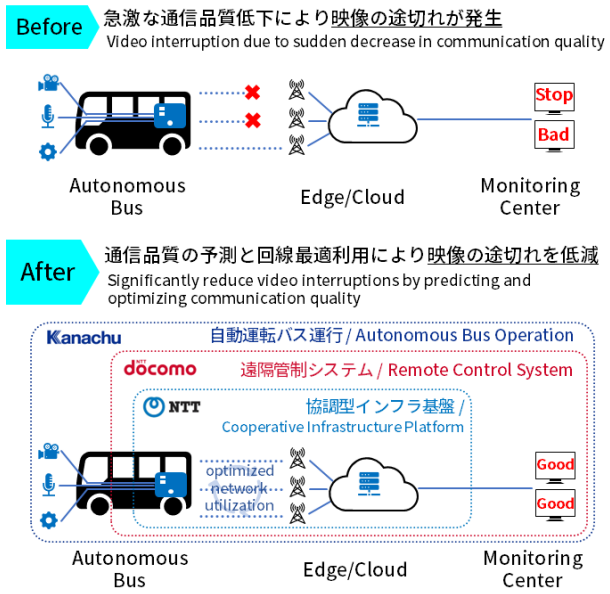
ハイブリッド
監視システムの開発

自動運転車両向け遠隔監視
システムの開発

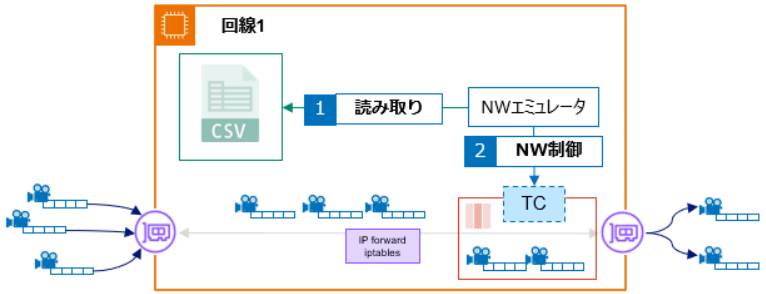
今回お話する内容

以下３つのことをお話します。

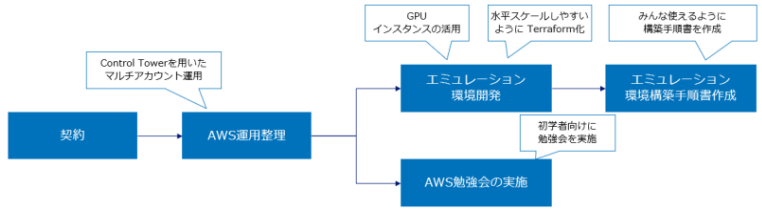
1 グループの紹介

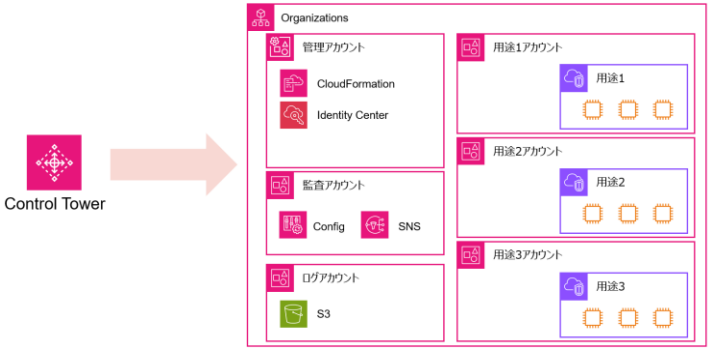


2 エミュレーションの紹介



3 AWS導入の工夫



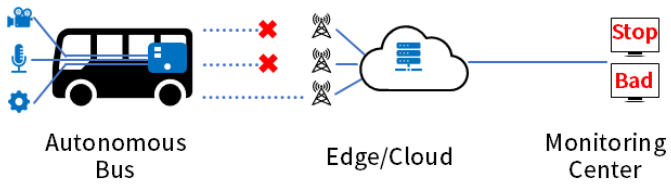


グループの紹介

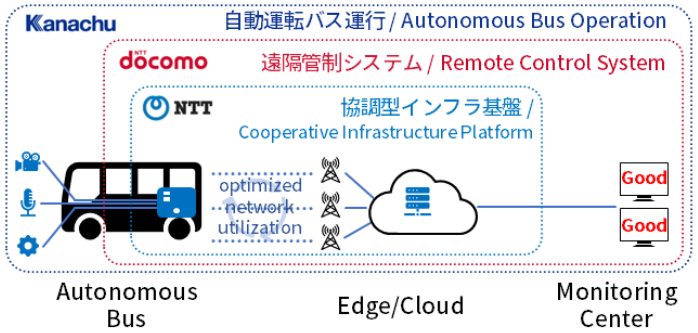
サービス処理基盤研究グループ（技サG）についてご紹介します。

1 グループの紹介

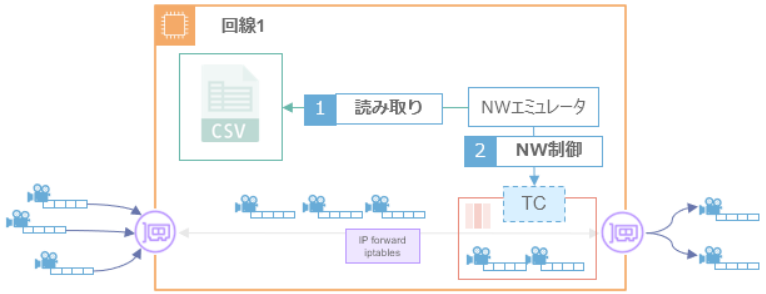
Before 急激な通信品質低下により映像の途切れが発生
Video interruption due to sudden decrease in communication quality



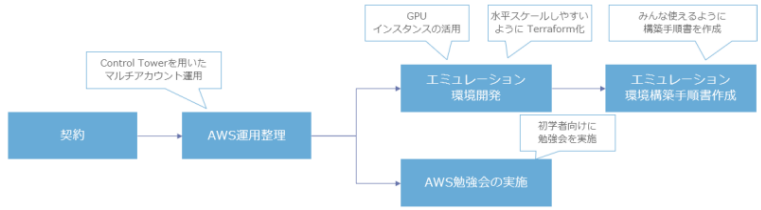
After 通信品質の予測と回線最適利用により映像の途切れを低減
Significantly reduce video interruptions by predicting and optimizing communication quality

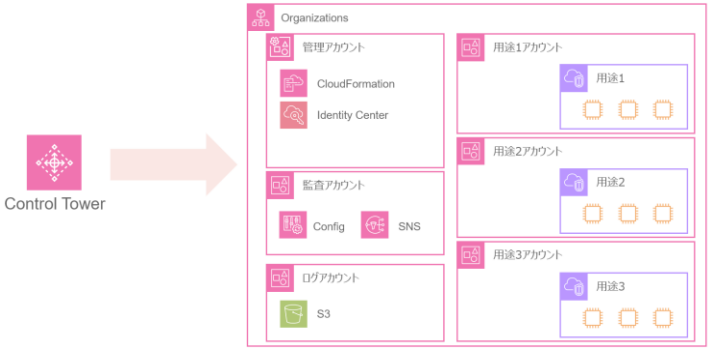


2 エミュレーションの紹介



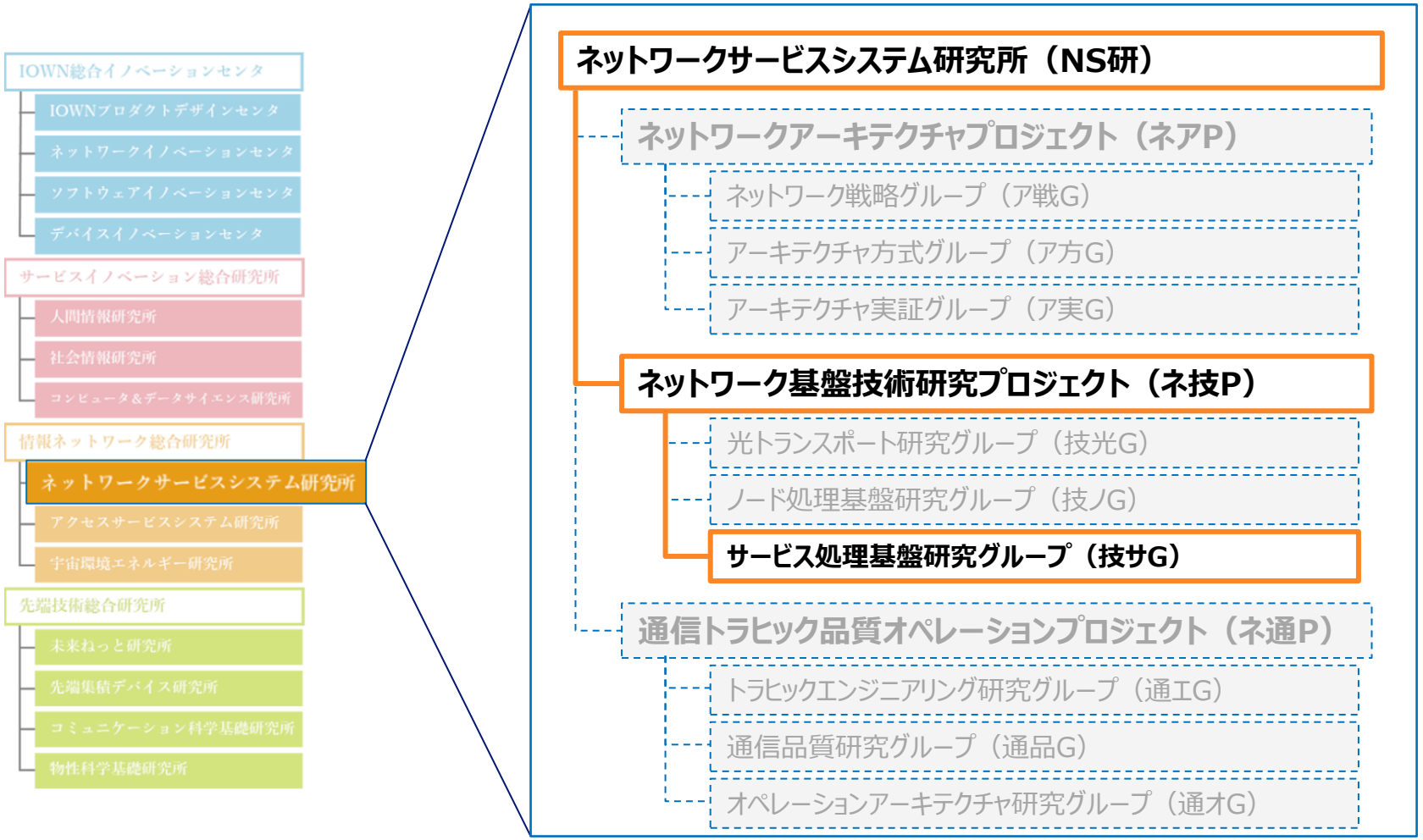
3 AWS導入の工夫





サービス処理基盤研究グループ（技サG）

情報ネットワーク総合研究所 ネットワークサービスシステム研究所 ネットワーク基盤技術研究プロジェクト に属しています。

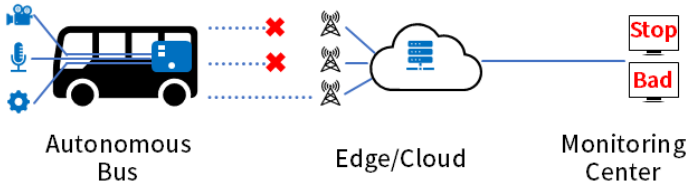


サービス処理基盤研究グループ（技サG）の取り組みについて

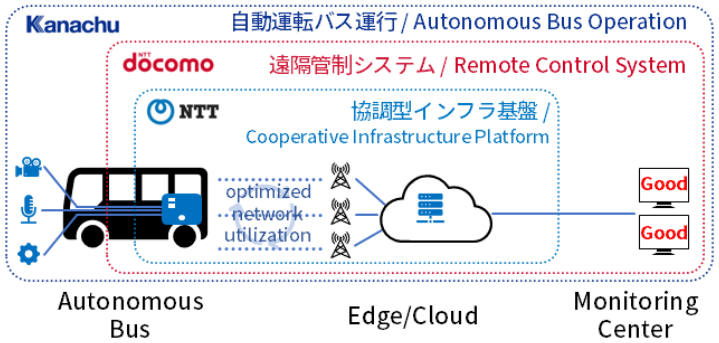
自動運転車両向けの映像伝送システムの開発等を行っています。

R&Dフォーラム展示

Before 急激な通信品質低下により映像の途切れが発生
Video interruption due to sudden decrease in communication quality



After 通信品質の予測と回線最適利用により映像の途切れを低減
Significantly reduce video interruptions by predicting and optimizing communication quality



神奈川中央交通、docomoと連携し平塚市などでフィールド実証を行っています。

公式YouTube

（https://www.youtube.com/watch?v=DGsOJLCkjQs&t=64s）

協調型インフラ基盤の自動運転遠隔管制システムへの適用
789 回視聴・2 か月前
NTT official channel
協調型インフラ基盤を活用した途切れないネットワークにより、複数の自動運転車の車内外の映像をリアルタイムに監視します。

技術ジャーナル

（https://journal.ntt.co.jp/article/30199）

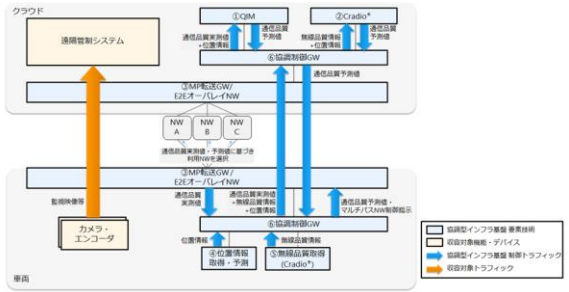


図1　協調型インフラ基盤のイメージ図

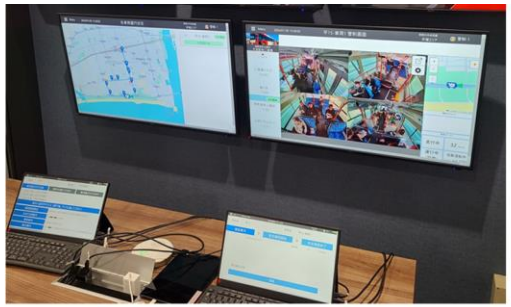
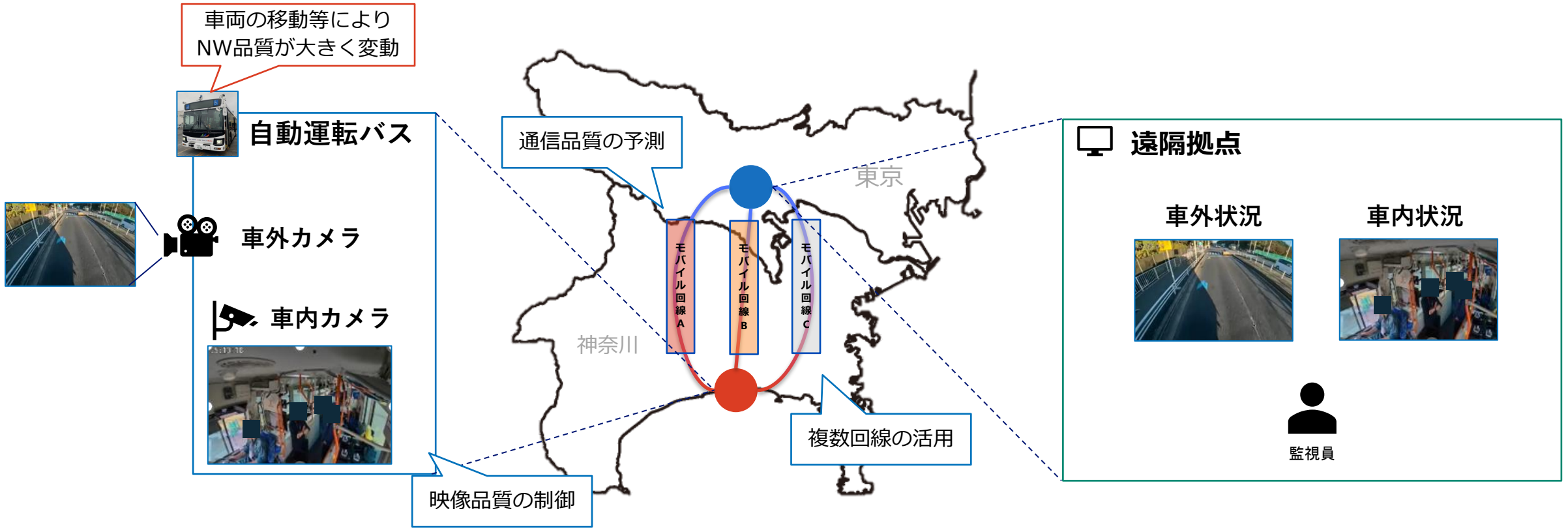


図2　ドコモ管制システム

サービス処理基盤研究グループ（技サG）の取り組みについて

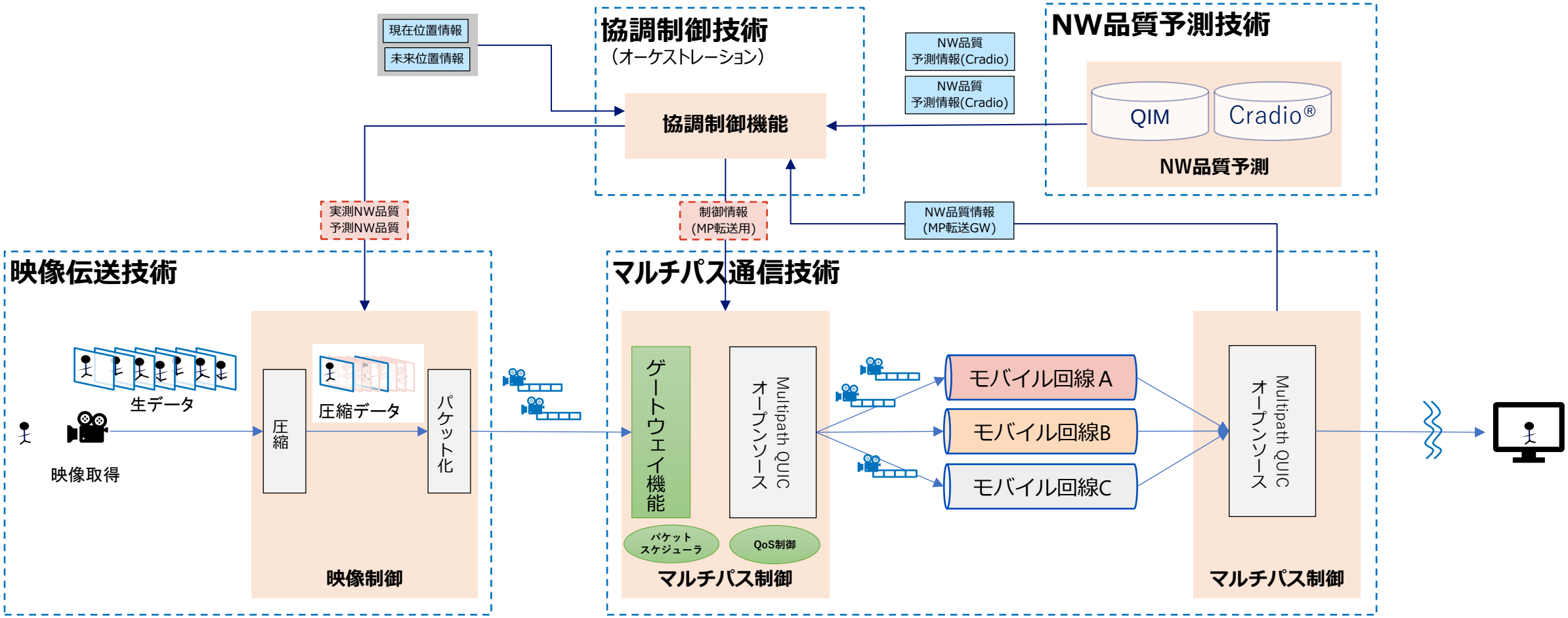
日本では自動運転レベル4の車両を動かす際 **「常時車両の状況を把握できること」**を道交法で定めています。
車両の移動で変動するNW品質の中でも映像伝送ができるよう技術開発をしています。



参考：道路交通法 | e-Gov 法令検索

サービス処理基盤研究グループ（技サG）の技術

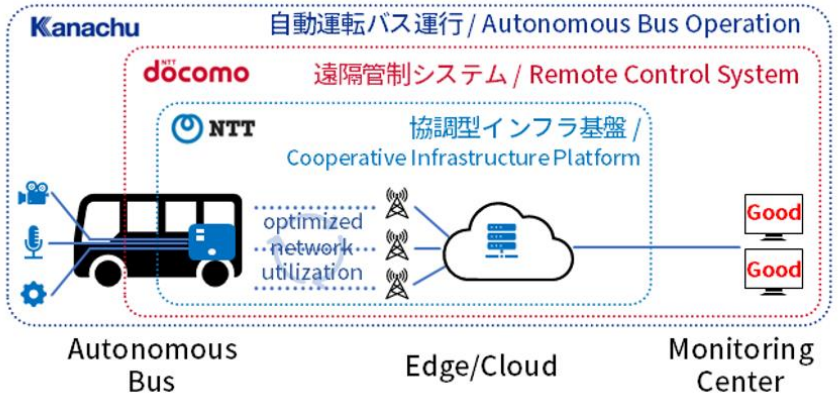
マルチパス通信、ネットワーク品質予測、映像伝送技術 とそのオーケストレーション機能からなるシステムの開発をしています。



サービス処理基盤研究グループ（技サG）の強み

グループ各社との連携、外部企業との実証など積極的に行っていますので、
事業・サービスで利用してみたい技術などありましたらお声掛けください。

グループ各社連携



外部企業との実証

自動運転バス

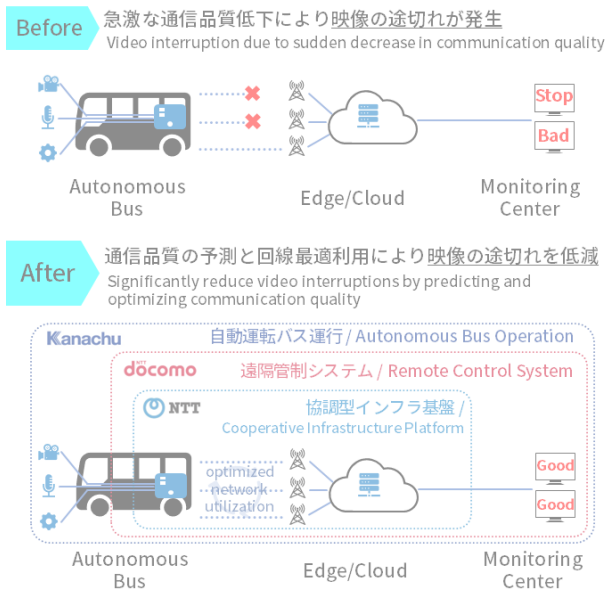
遠隔管制室

▲車外映像　▼車内映像

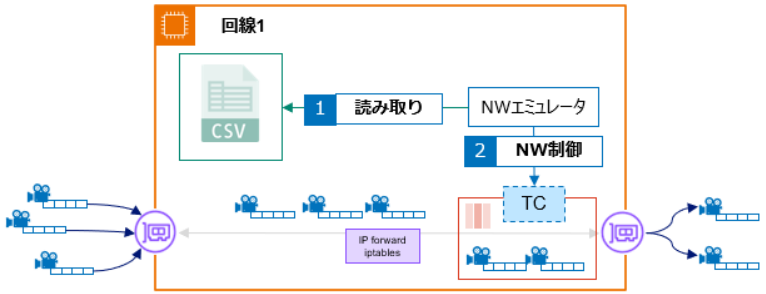
ネットワークエミュレーションの紹介

ネットワークエミュレーションの紹介をします。

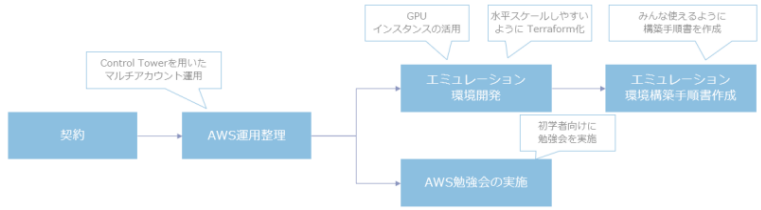
1 グループの紹介

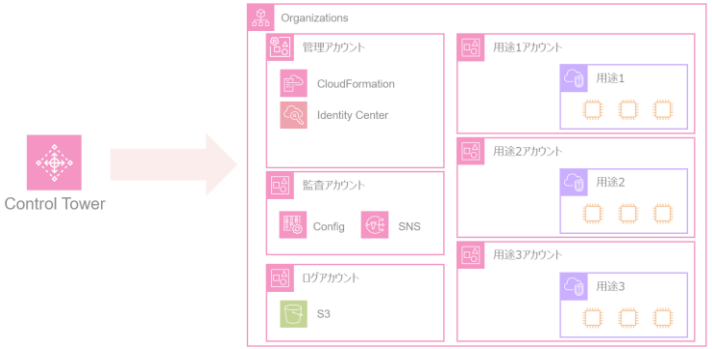


2 エミュレーションの紹介



3 AWS導入の工夫

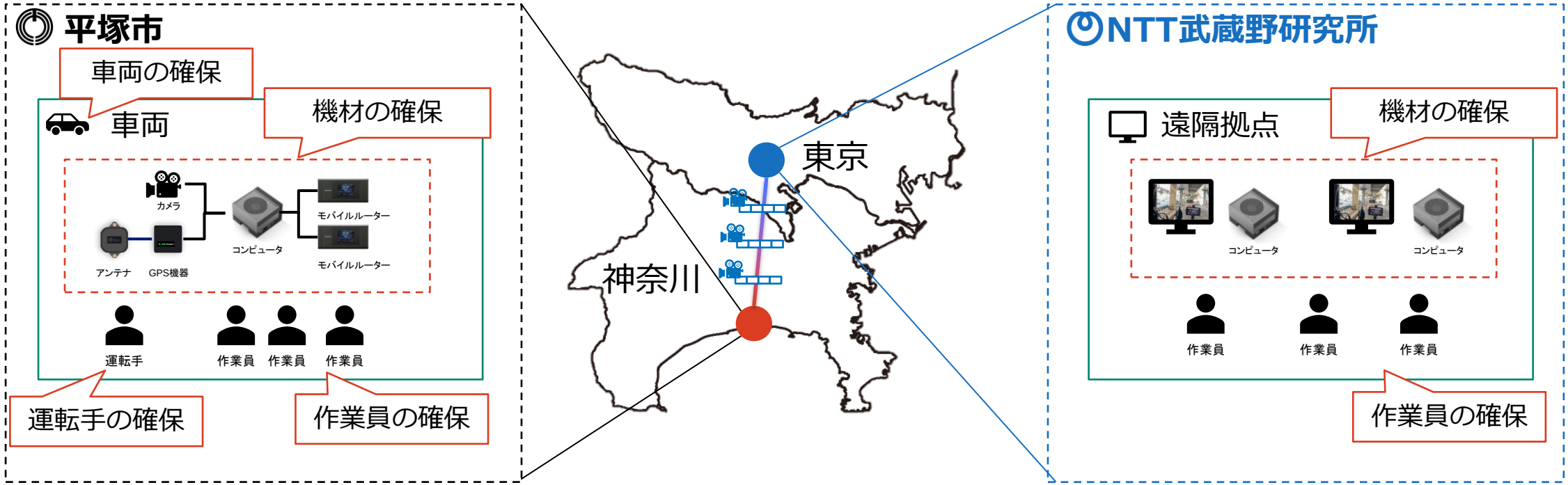




実車走行検証の課題

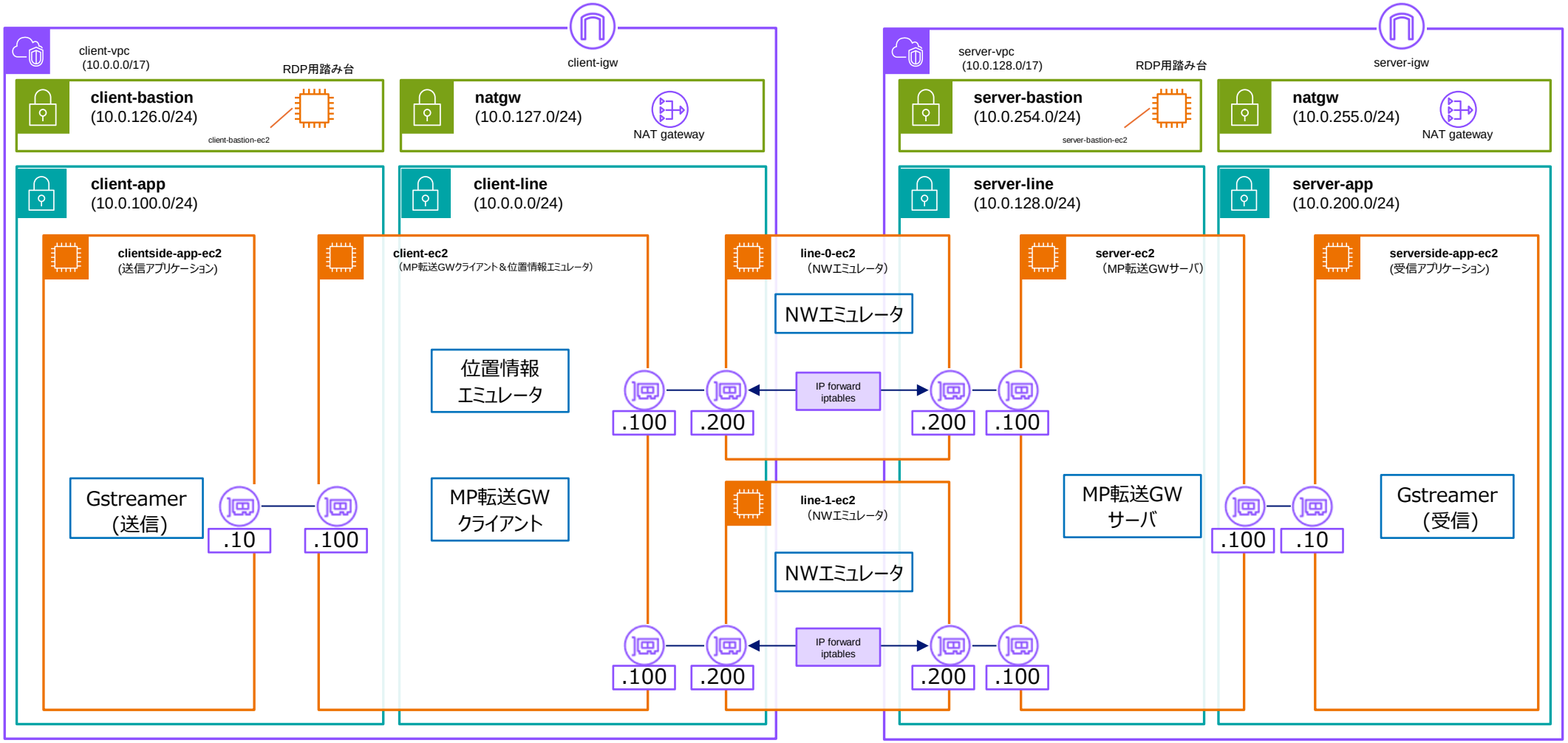
実車走行で検証を行う場合、**「検証に使うリソースが多い」**という課題があります。
（車両・運転手・機材・作業員の確保、走行ルートの選定など多くのリソースが必要になります。）

※例：平塚で実証を行う場合



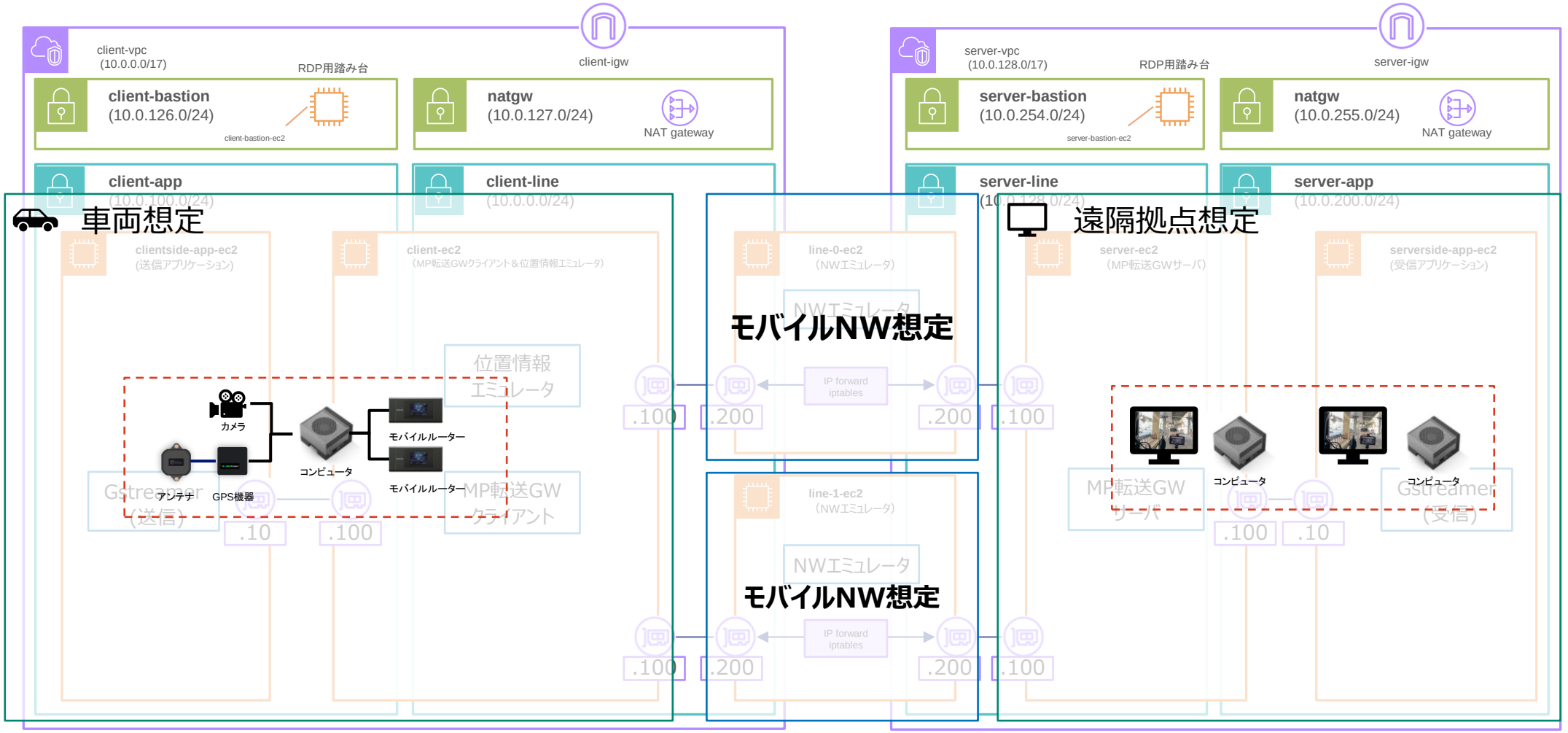
解決方法としてのエミュレーション

少ない稼働で検証を行えるように **AWS** に **エミュレーション環境** を構築



解決方法としてのエミュレーション

2つのVPCと車両想定のEC2、遠隔拠点想定のEC2、モバイルNW想定のEC2で作成

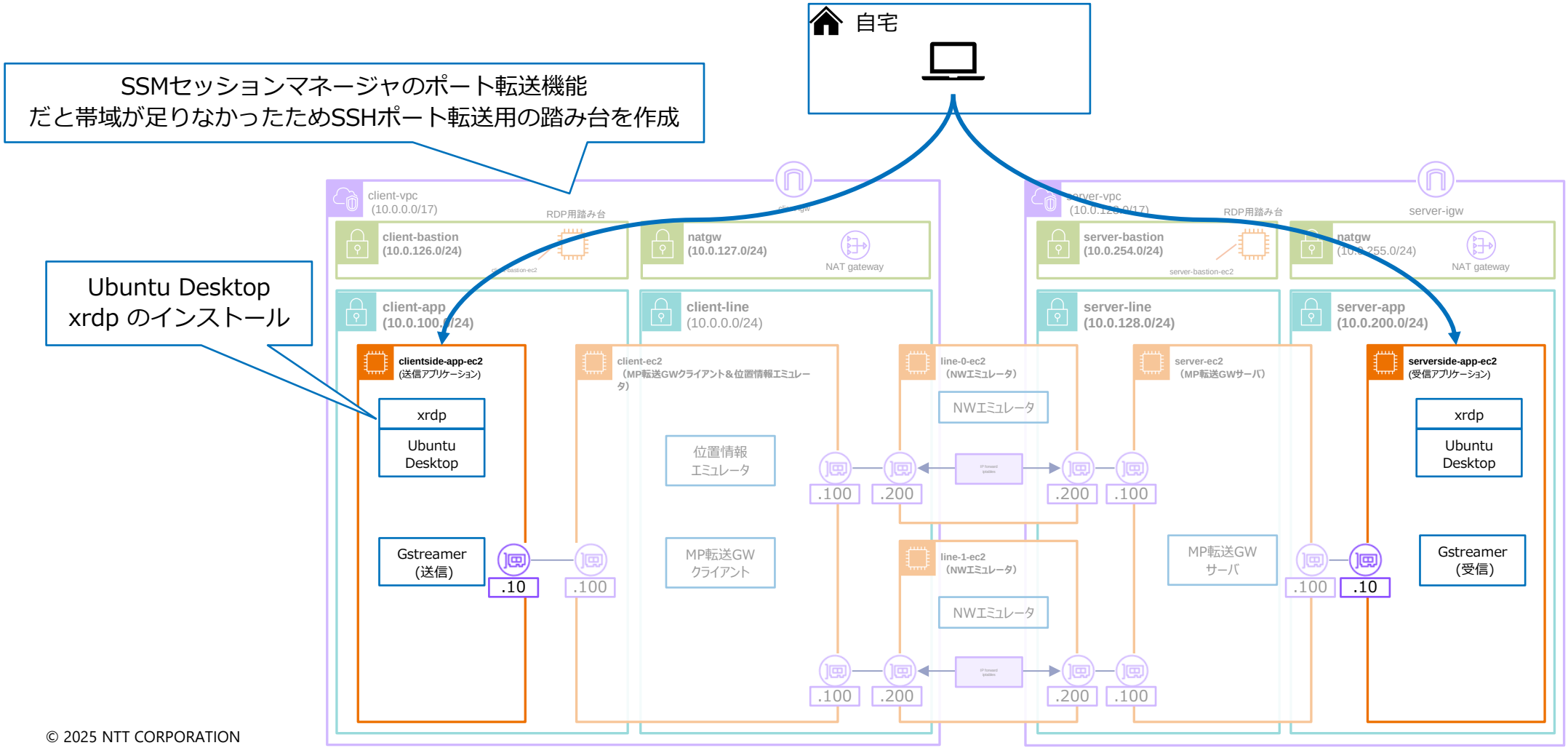


検証環境への接続

EC2(Ubuntu)にxrdpとUbuntu Desktopをインストールし自宅PCからRDP接続後
映像伝送の検証を行います。

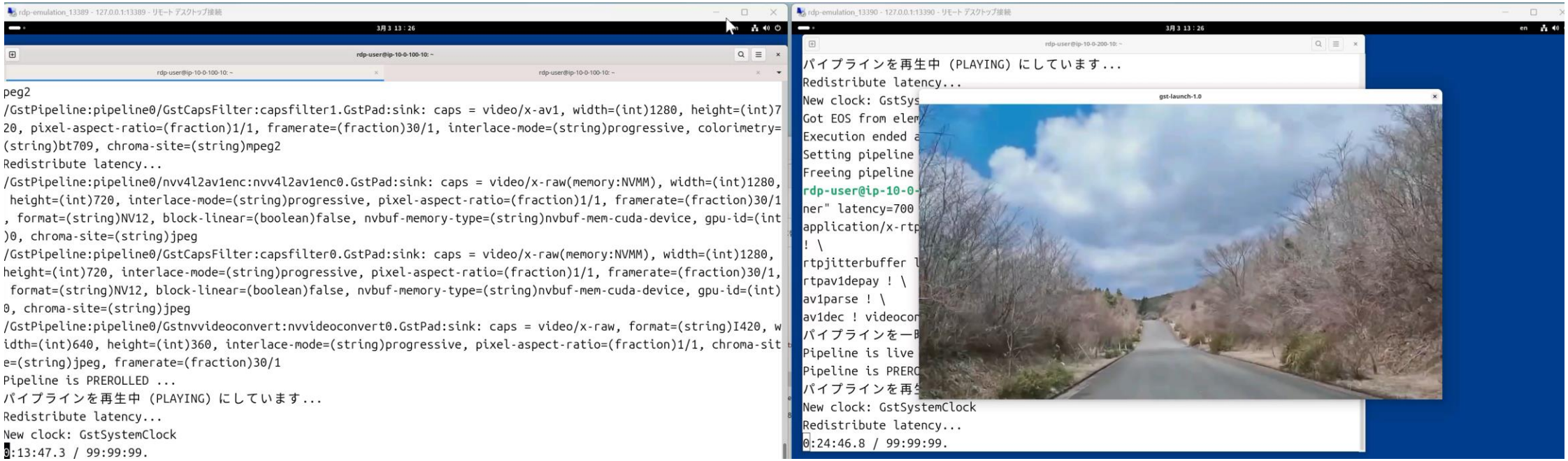
SSMセッションマネージャのポート転送機能
だと帯域が足りなかったためSSHポート転送用の踏み台を作成

Ubuntu Desktop
xrdp のインストール



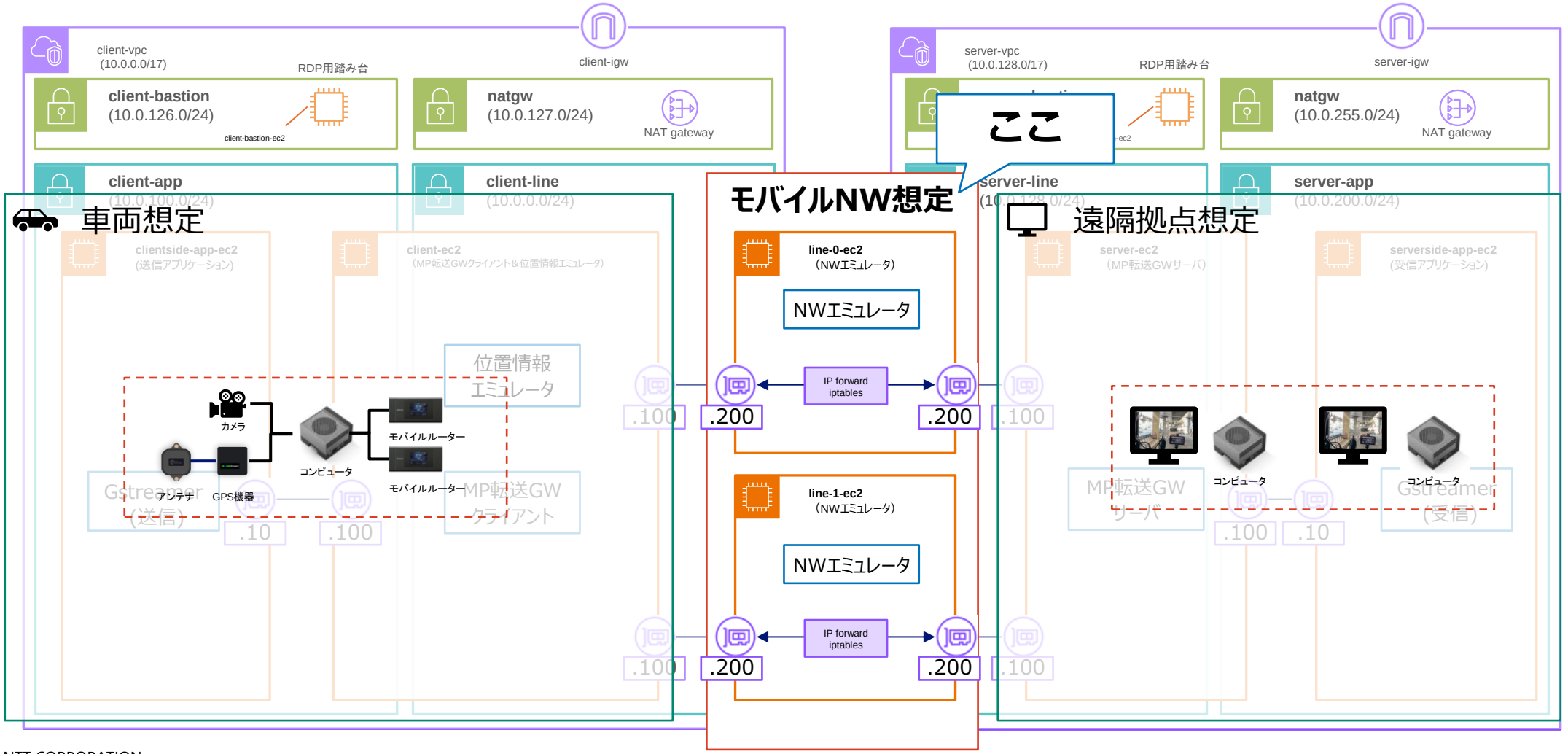
実際の映像

このように動画を流しながら、裏ででネットワークエミュレーションを行い検証を可能にしています



通信のエミュレーションの手法

通信のエミュレーションの手法を紹介します。



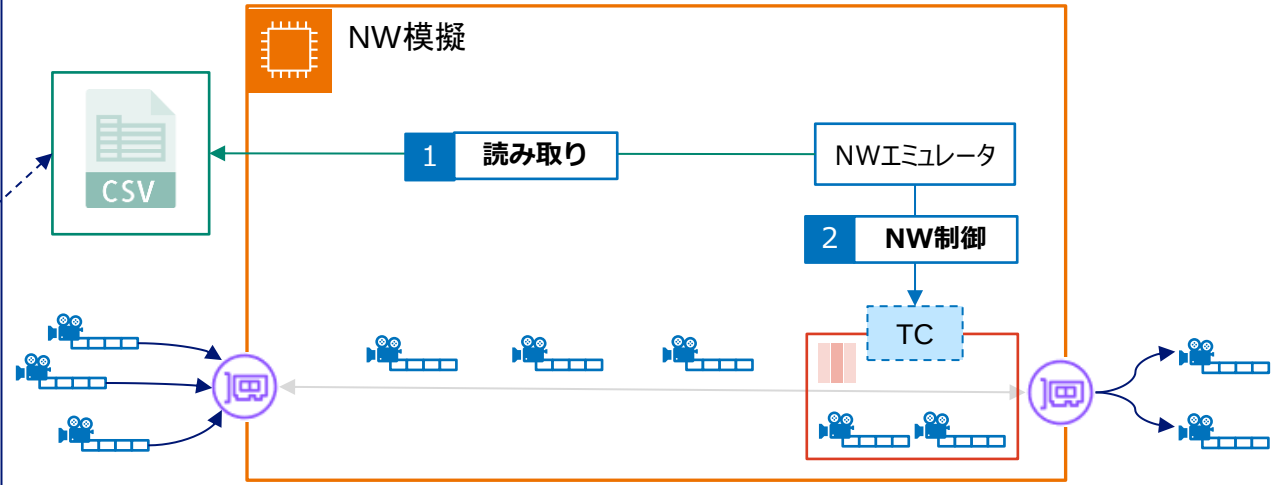
通信のエミュレーションの手法

まずネットワーク品質をiperfで測定します。そして測定データの補正を行い、帯域、遅延などのカラムを持つCSVを作成後、作成したCSVをもとにTCでネットワーク制御をするプログラムを組みエミュレートをしています。

1 iperfでモバイルネットワークの品質を測定

2 測定データの補正を行いCSV作成

3 NWエミュレータで再現



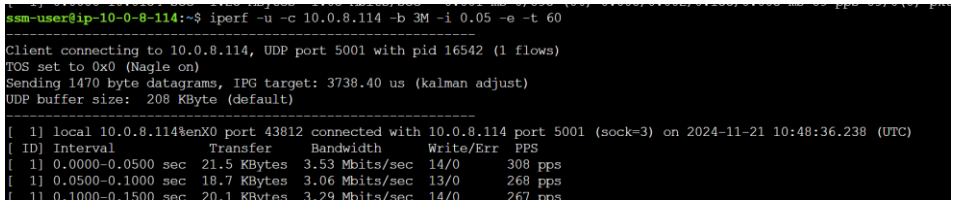
iperfとは

ネットワーク品質を測定できるソフトウェア。クライアントサーバで動作し実際にパケットを流しネットワークの品質を測定する。
遅延（片方向*/双方向）、ジッタ、スループット、パケットロスの測定が可能。
UDP、TCPの指定、MTUの設定など細かい設定も可能。

コマンド例 (クライアント)

```
iperf -u -c IPアドレス -b 3M -i 0.05 -e -t 60
```

出力



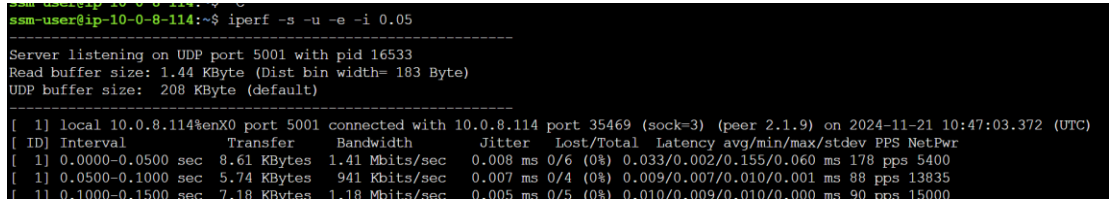
カラム名	値例	意味
ID	[1]	識別番号
Interval（インターバル）	0.0000-0.0500 sec	測定が行われた時間
Transfer（転送量）	21.5 KBytes	インターバル内で転送されたデータ量
Bandwidth（帯域幅）	3.53 Mbits/sec	インターバル内の平均データ転送速度
Write/Err（書き込み/エラー）	14/0	書き込み操作の回数とエラーの数
PPS（Packets Per Second）	308 pps	毎秒の送信パケット数

※片方向遅延の測定はiperf2のみ
　iperfの種類によって利用できるOS、測定項目は異なるので注意

コマンド例 (サーバ)

```
iperf -s -u -e -i 0.05
```

出力



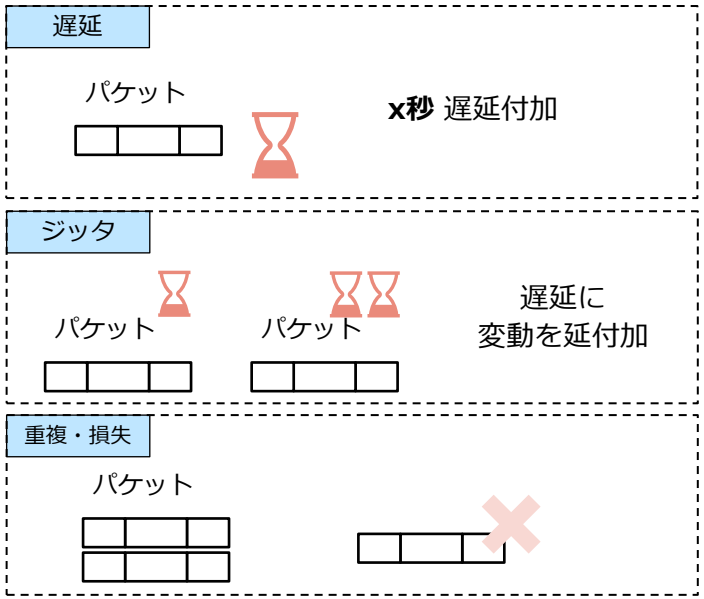
カラム名	値例	意味
ID	[1]	識別番号
Interval（インターバル）	0.0000-0.0500 sec	測定が行われた時間間隔
Transfer（転送量）	8.61 KBytes	インターバル内で受信したデータ量
Bandwidth（帯域幅）	1.41 Mbits/sec	インターバル内の平均データ受信速度
Jitter（ジッター）	0.008 ms	パケット到着時間の変動（遅延のばらつき）
Lost/Total（パケットロス）	0/6 (0%)	失われたパケット数と総パケット数、およびパケットロスの割合
Latency avg/min/max/stdev（レイテンシ 平均/最小/最大/標準偏差）	0.033/0.002/0.155/0.060 ms	レイテンシの平均値、最小値、最大値、標準偏差
PPS（Packets Per Second）	178 pps	毎秒の受信パケット数
NetPwr（ネットワークパワー）	5400	スループット/レイテンシ

(参考) iperf2ログ(サーバー)

```
------------------------------------------------------------
Server listening on UDP port 5001 with pid 12670
Read buffer size: 1.44 KByte (Dist bin width= 183 Byte)
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  1] local 10.0.8.114%enX0 port 5001 connected with 106.146.31.82 port 41159 (sock=3) (peer 2.1.5) on 2024-11-27 11:58:17.216 (UTC)
[ ID] Interval           Transfer     Bandwidth       Jitter   Lost/Total  Latency avg/min/max/stdev PPS NetPwr
[  1] 0.0000-0.0500 sec  2.87 KBytes   470 Kbits/sec   2.125 ms 0/2 (0%) 105.480/84.678/120.350/18.561 ms 75 pps 0.557442
[  1] 0.0000-0.0500 sec  1 datagrams received out-of-order
[  1] 0.0500-0.1000 sec  1.44 KBytes   235 Kbits/sec   4.177 ms 0/1 (0%) 155.298/155.298/155.298/0.000 ms 29 pps 0.189310
[  1] 0.1000-0.1500 sec  4.31 KBytes   706 Kbits/sec   7.129 ms 0/3 (0%) 202.814/191.546/219.077/14.429 ms 40 pps 0.434873
[  1] 0.1500-0.2000 sec  2.87 KBytes   470 Kbits/sec   8.185 ms 0/2 (0%) 248.368/245.303/251.434/4.335 ms 50 pps 0.236740
[  1] 0.2000-0.2500 sec  2.87 KBytes   470 Kbits/sec   9.759 ms 0/2 (0%) 285.967/277.817/294.118/11.527 ms 40 pps 0.205614
[  1] 0.2500-0.3000 sec  2.87 KBytes   470 Kbits/sec  11.395 ms 0/2 (0%) 335.858/330.257/341.458/7.920 ms 36 pps 0.175071
[  1] 0.3000-0.3500 sec  5.74 KBytes   941 Kbits/sec  10.822 ms 0/4 (0%) 362.263/347.811/376.661/12.054 ms 80 pps 0.324620
[  1] 0.3500-0.4000 sec  4.31 KBytes   706 Kbits/sec  10.834 ms 0/3 (0%) 404.065/392.778/410.311/9.793 ms 67 pps 0.218277
[  1] 0.4000-0.4500 sec  4.31 KBytes   706 Kbits/sec  10.302 ms 0/3 (0%) 427.855/421.631/434.066/6.218 ms 86 pps 0.206141
[  1] 0.4500-0.5000 sec  7.18 KBytes  1.18 Mbits/sec  10.308 ms 0/5 (0%) 472.496/461.748/485.464/10.198 ms 71 pps 0.311108
[  1] 0.5000-0.5500 sec  4.31 KBytes   706 Kbits/sec  10.192 ms 0/3 (0%) 504.761/496.840/514.398/8.904 ms 75 pps 0.174733
[  1] 0.5500-0.6000 sec  7.18 KBytes  1.18 Mbits/sec   9.880 ms 0/5 (0%) 545.046/530.511/560.578/11.026 ms 77 pps 0.269697
[  1] 0.6000-0.6500 sec  5.74 KBytes   941 Kbits/sec   8.786 ms 0/4 (0%) 572.536/566.887/580.692/6.362 ms 114 pps 0.205398
[  1] 0.6500-0.7000 sec  5.74 KBytes   941 Kbits/sec   9.601 ms 0/4 (0%) 613.766/596.855/630.642/14.531 ms 61 pps 0.191600
[  1] 0.7000-0.7500 sec  5.74 KBytes   941 Kbits/sec   9.078 ms 0/4 (0%) 652.519/646.858/660.628/6.359 ms 89 pps 0.180221
[  1] 0.7500-0.8000 sec  8.61 KBytes  1.41 Mbits/sec   7.908 ms 0/6 (0%) 680.127/666.917/693.356/9.425 ms 109 pps 0.259358
[  1] 0.8000-0.8500 sec  5.74 KBytes   941 Kbits/sec   7.847 ms 0/4 (0%) 710.264/699.623/723.420/10.143 ms 89 pps 0.165569
[  1] 0.8500-0.9000 sec  5.74 KBytes   941 Kbits/sec   8.057 ms 0/4 (0%) 745.282/734.579/758.475/10.171 ms 80 pps 0.157789
[  1] 0.9000-0.9500 sec  5.74 KBytes   941 Kbits/sec   8.517 ms 0/4 (0%) 782.819/769.598/798.590/12.782 ms 73 pps 0.150223
[  1] 0.9500-1.0000 sec  5.74 KBytes   941 Kbits/sec   8.305 ms 0/4 (0%) 815.317/804.752/828.417/10.111 ms 89 pps 0.144235
```

TC (Linux Traffic Control)とは

TC では帯域制限、遅延、ジッタ、パケットの重複、パケットロスの再現が可能。



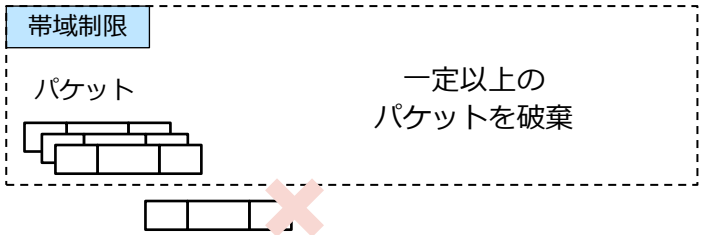
TCでnetem、HTBを使い
帯域、遅延などの再現が可能

netem (Network Emulation)

tc qdisc add dev <デバイス名> root netem delay <遅延>ms <ジッタ>ms

tc qdisc add dev eth0 root netem loss 1%

tc qdisc add dev eth0 root netem duplicate 1%

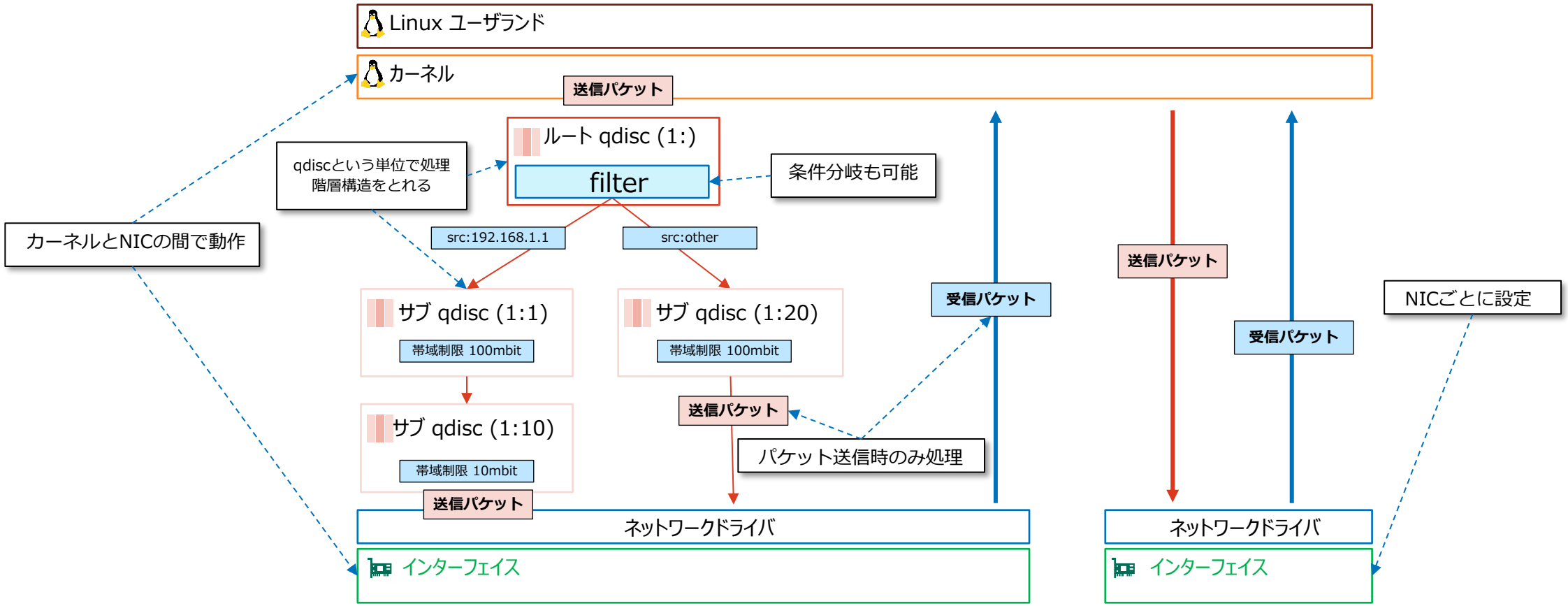


HTB (Hierarchical Token Bucket)

tc qdisc add dev eth0 root handle 1: htb rate 1mbit

TC による制御の動作ポイント

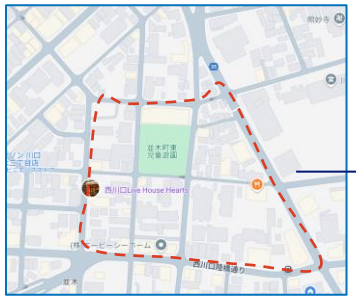
通信の制御はカーネル以下のレイヤ(qdisc)で行われる。
qdiscは階層構造をとることができ、5タプルを利用して条件分岐をすることなども可能。



通信のエミュレーションの手法

このようにiperfで測定した時系列でNW品質のデータを取得し、測定データを基にCSVを作成し、それを読み込み、
TCを実行するプログラムを用いることで、モバイルネットワークの再現をしています。
ブラックボックス化されたネットワークの再現がこれによって可能になります。

1 iperfでモバイルネットワークの品質を測定



測定

時刻_server.log

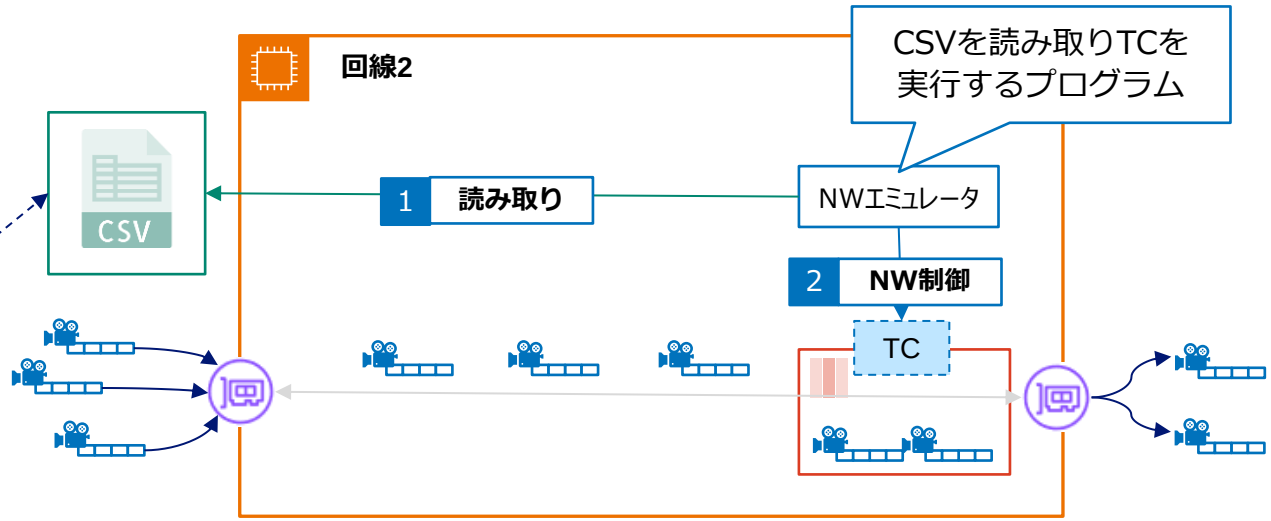
2 測定データの補正を行いCSV作成

時刻_server.log

補正

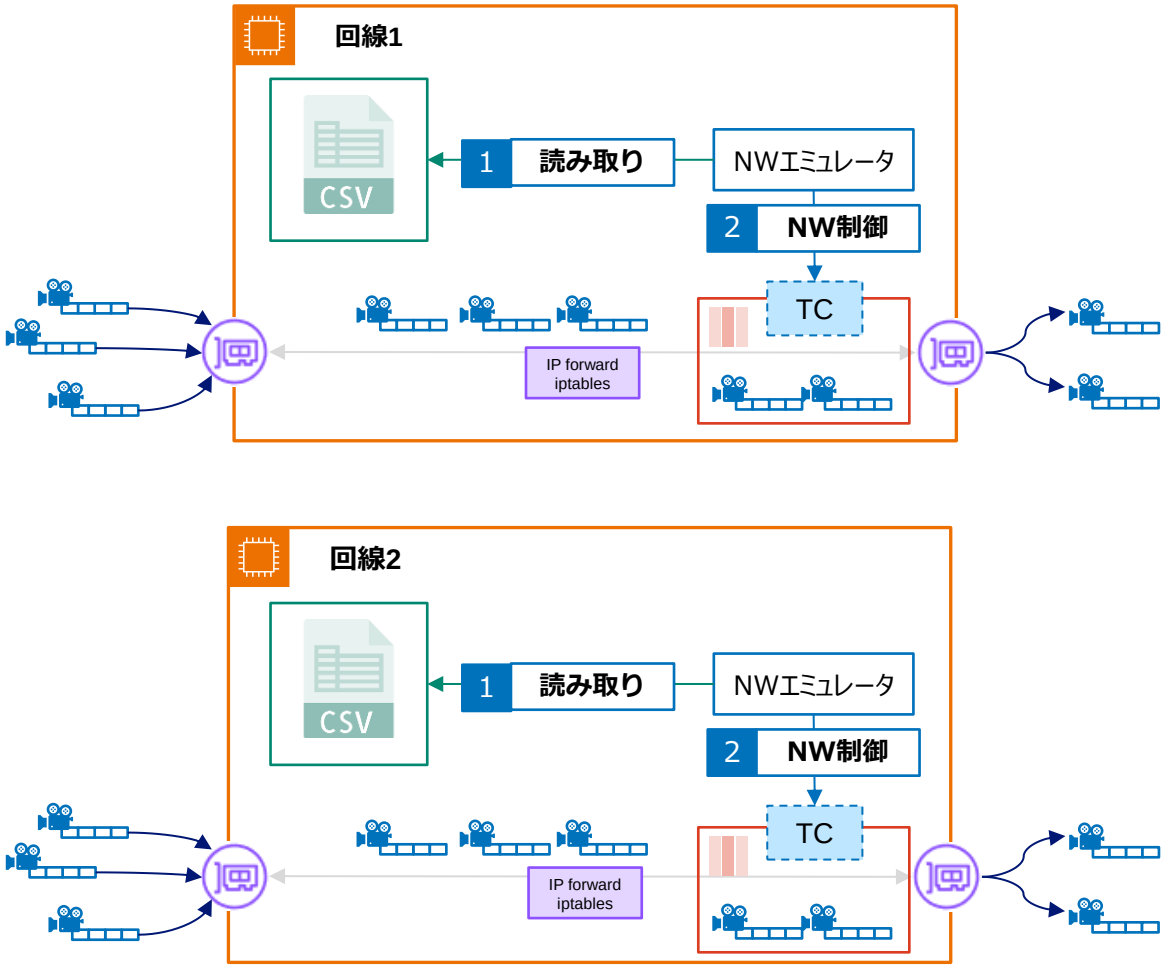
CSV

3 NWエミュレータで再現



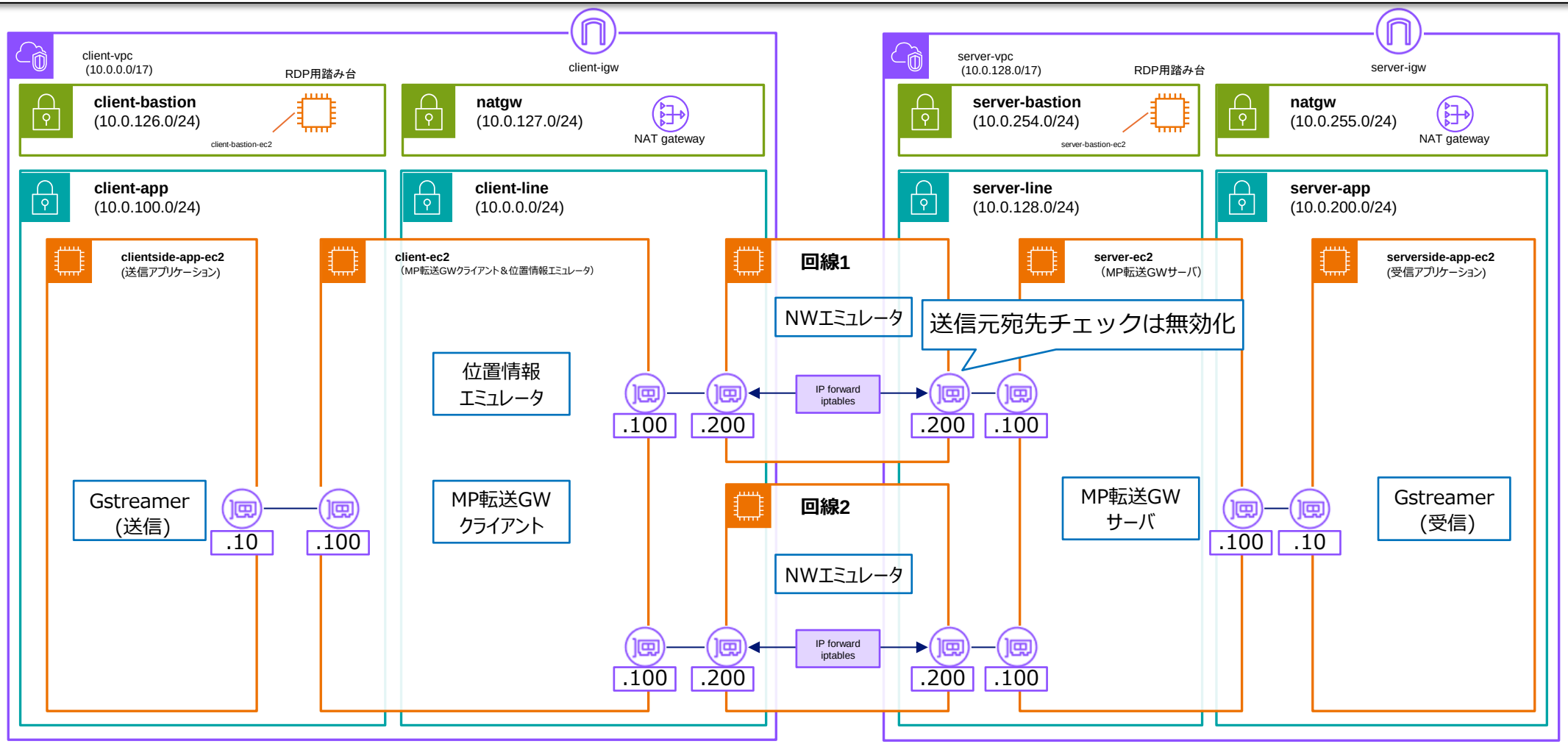
複数ネットワークの再現

複数モバイルネットワークのエミュレーションは各キャリアごとに測定したネットワーク品質を再現することで実現できます。



エミュレーション全体像と補足

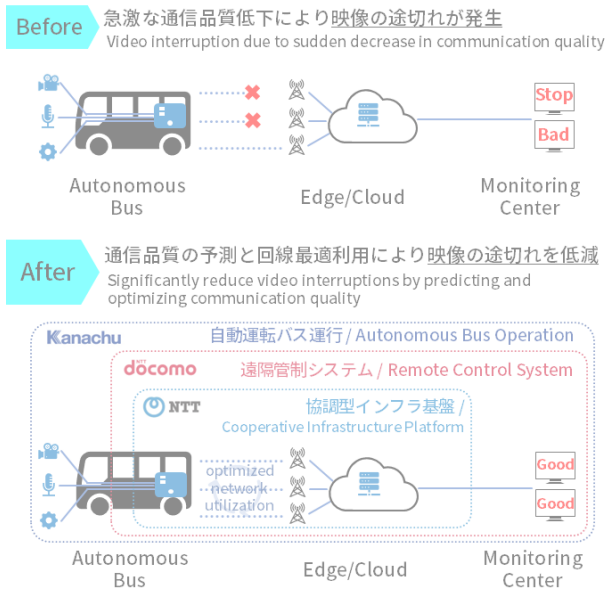
回線を再現するEC2はマルチVPCのENIアタッチ(2023/10 GA) を使いVPCをまたぐようにしています（通信経路を明確にするため）
また、**ipforward** と **iptables** でルーティングを許可する設定を入れています。



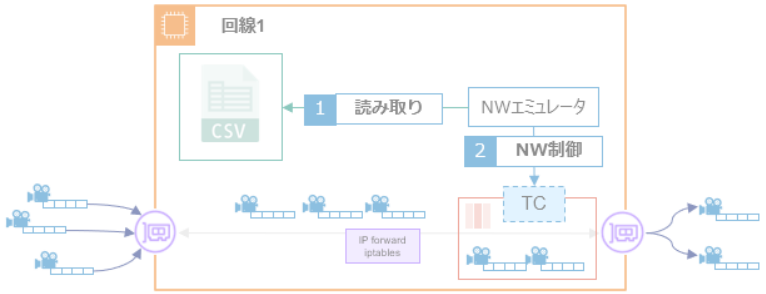
AWS導入の工夫

AWS導入の工夫点について紹介します。

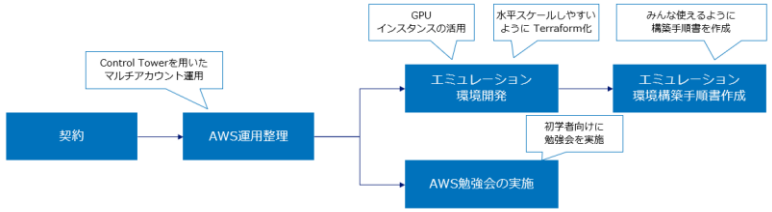
1 グループの紹介

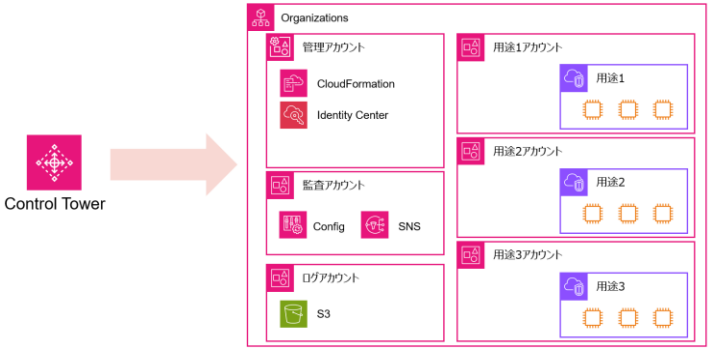


2 エミュレーションの紹介



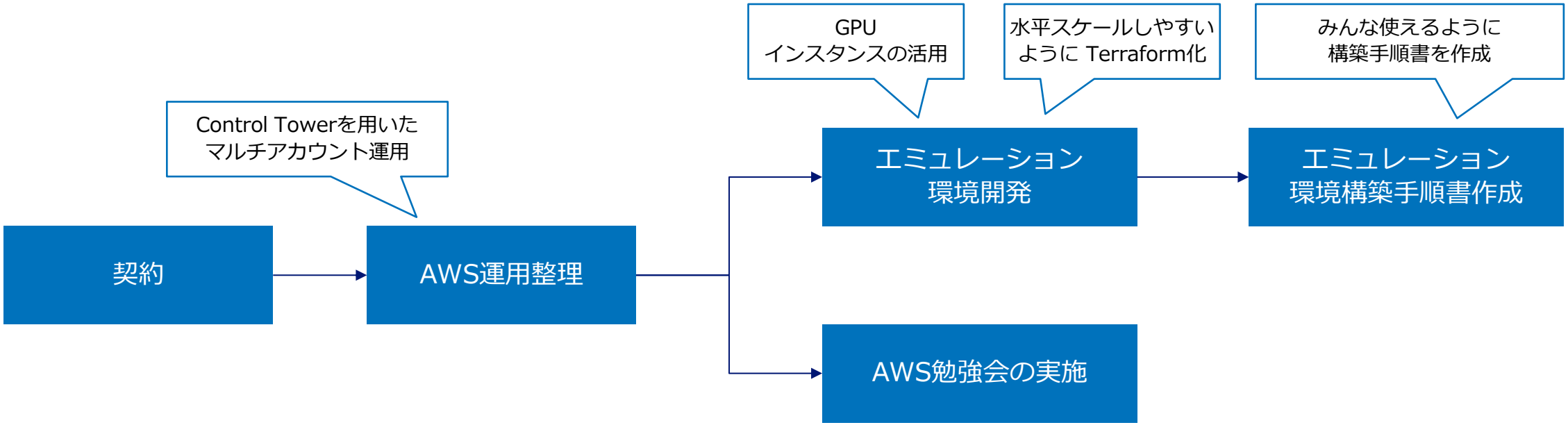
3 AWS導入の工夫





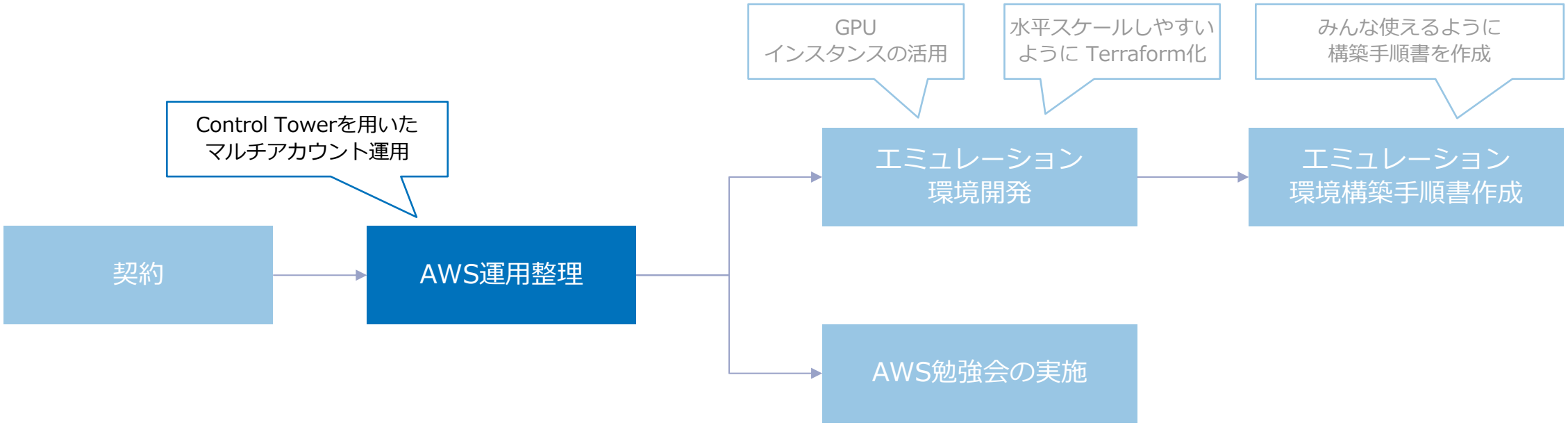
AWSをグループに導入する際の工夫

契約からエミュレーション環境の構築まで、様々な観点で工夫を実施



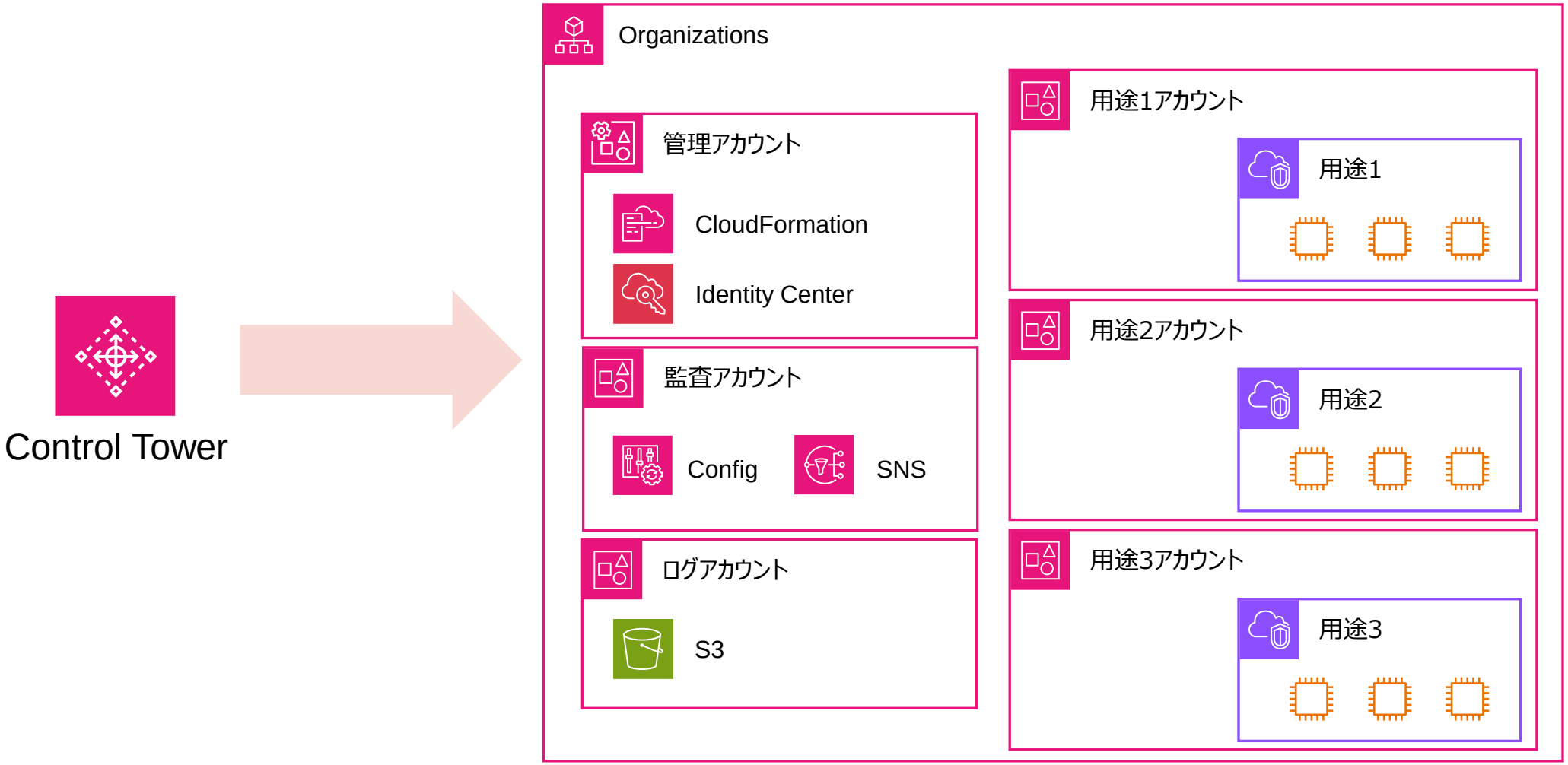
AWSをグループに導入する際の工夫

マルチアカウント運用についてからご紹介します



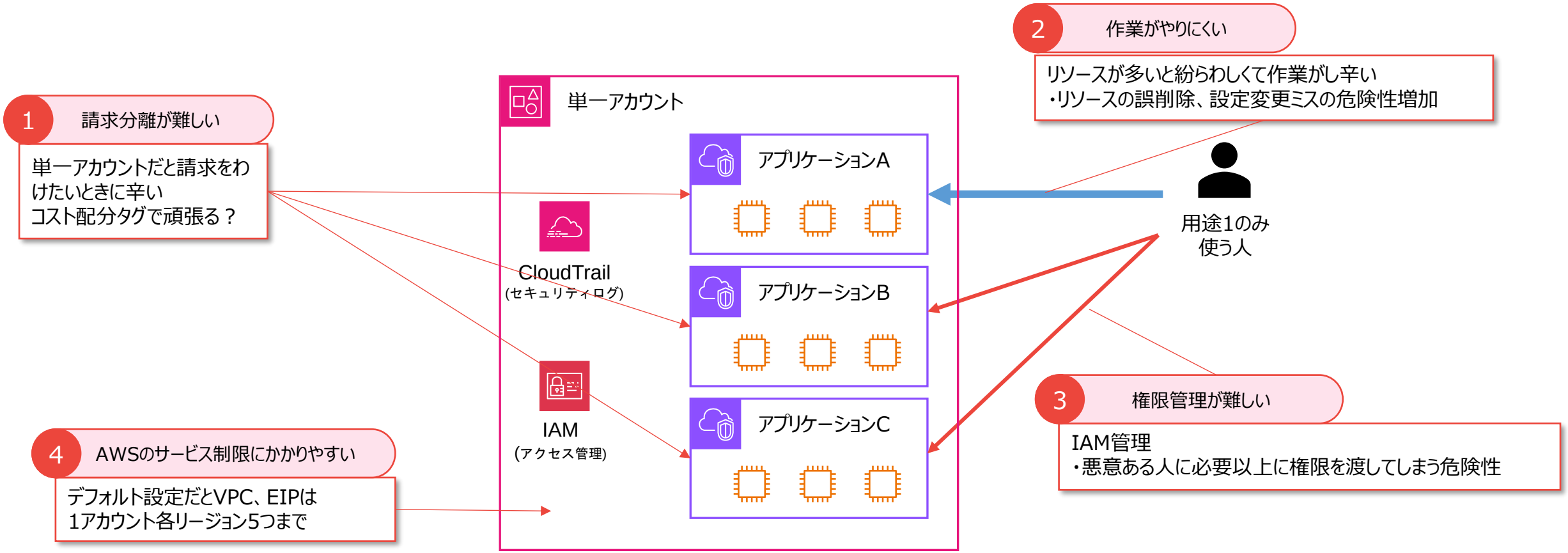
Control Towerとは

マルチアカウント管理を効率的に設定できるサービス。
マルチアカウントの辛いところである Organizations、Identity Center、CloudTrail、Config設定などを助けてくれる。



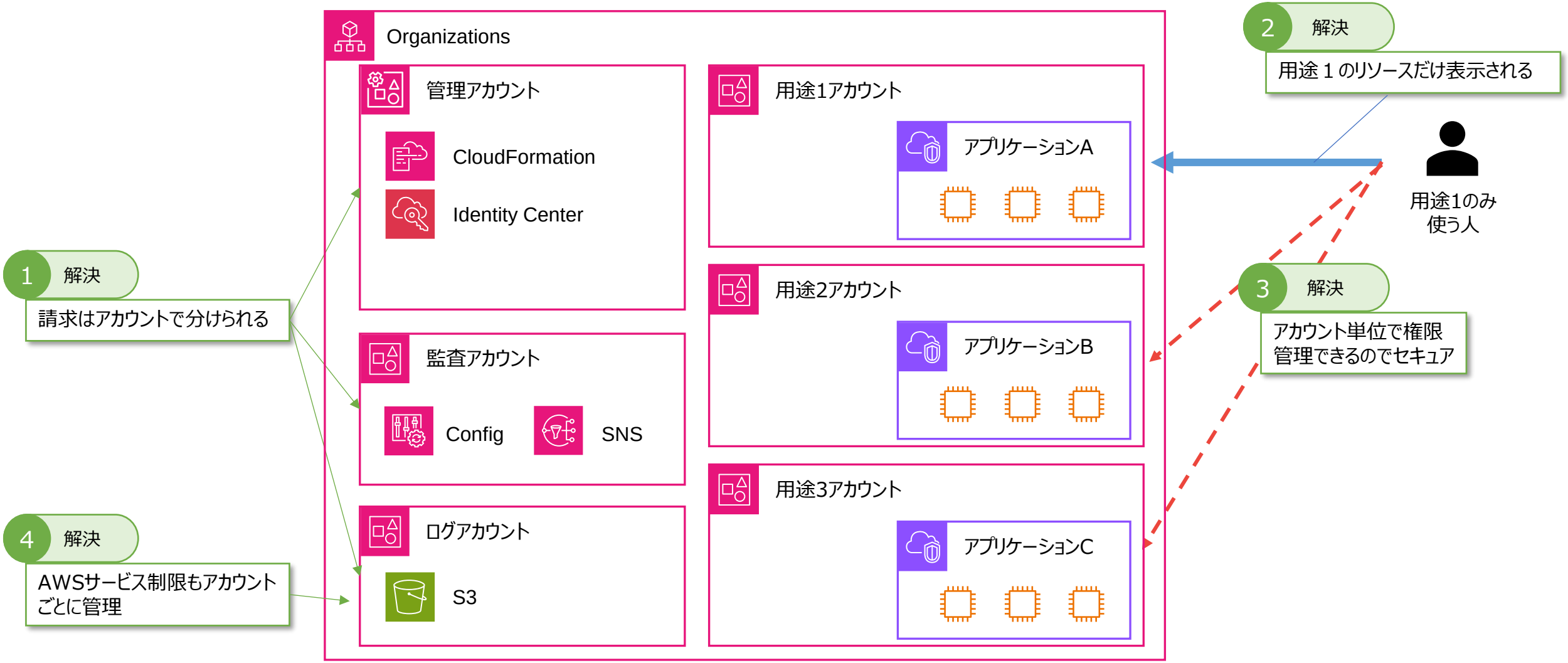
シングルアカウント運用のデメリット

シングルアカウント運用のデメリット、大きく以下４つあります。
「請求の分離が難しい」「リソースが多くなり作業がやりにくい」「権限管理が難しい」「AWSのサービス制限にかかりやすい」



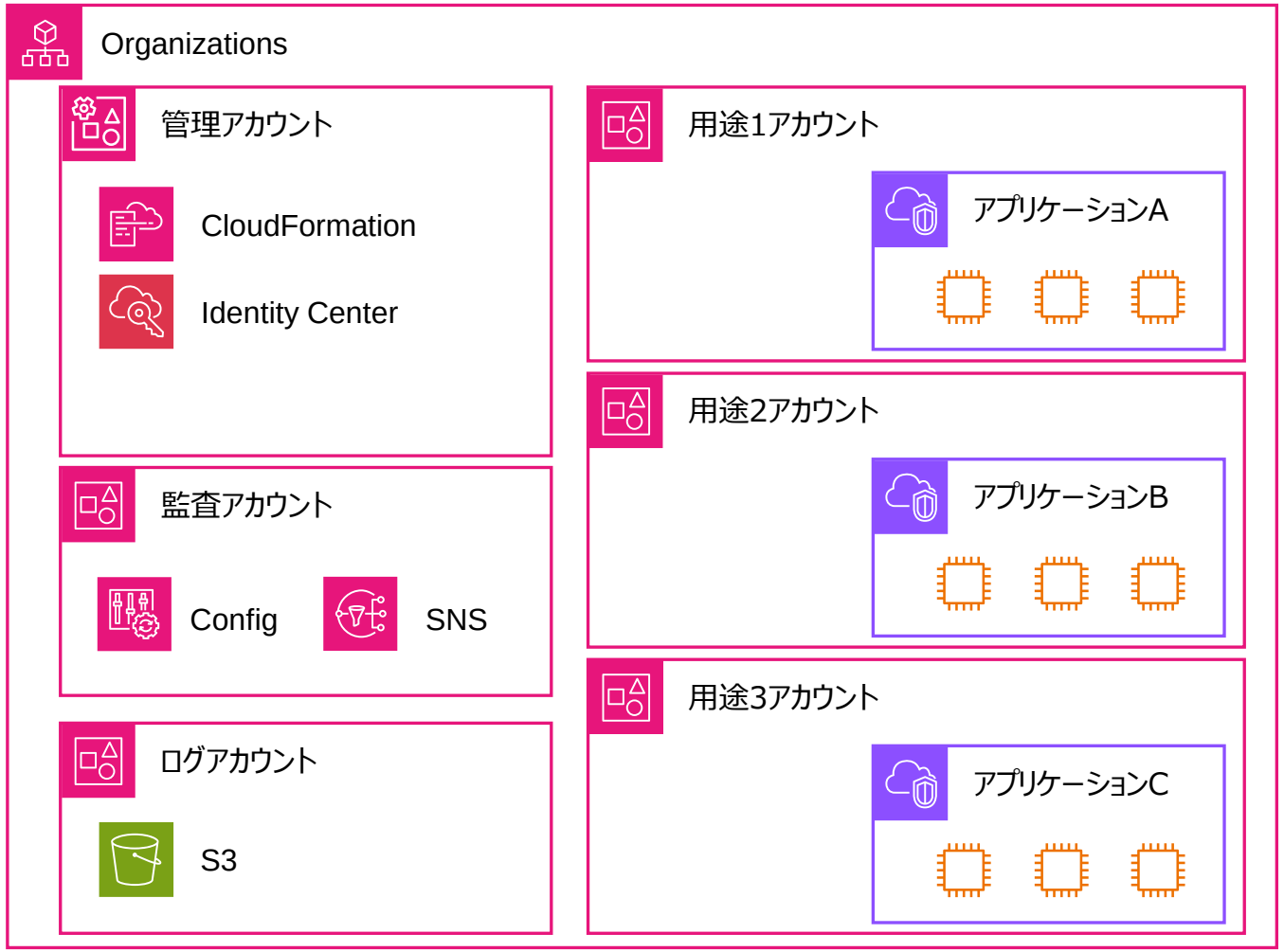
マルチアカウント運用のメリット

権限・リソース・請求・サービス制限の分離が簡単に可能！



マルチアカウント運用のデメリット

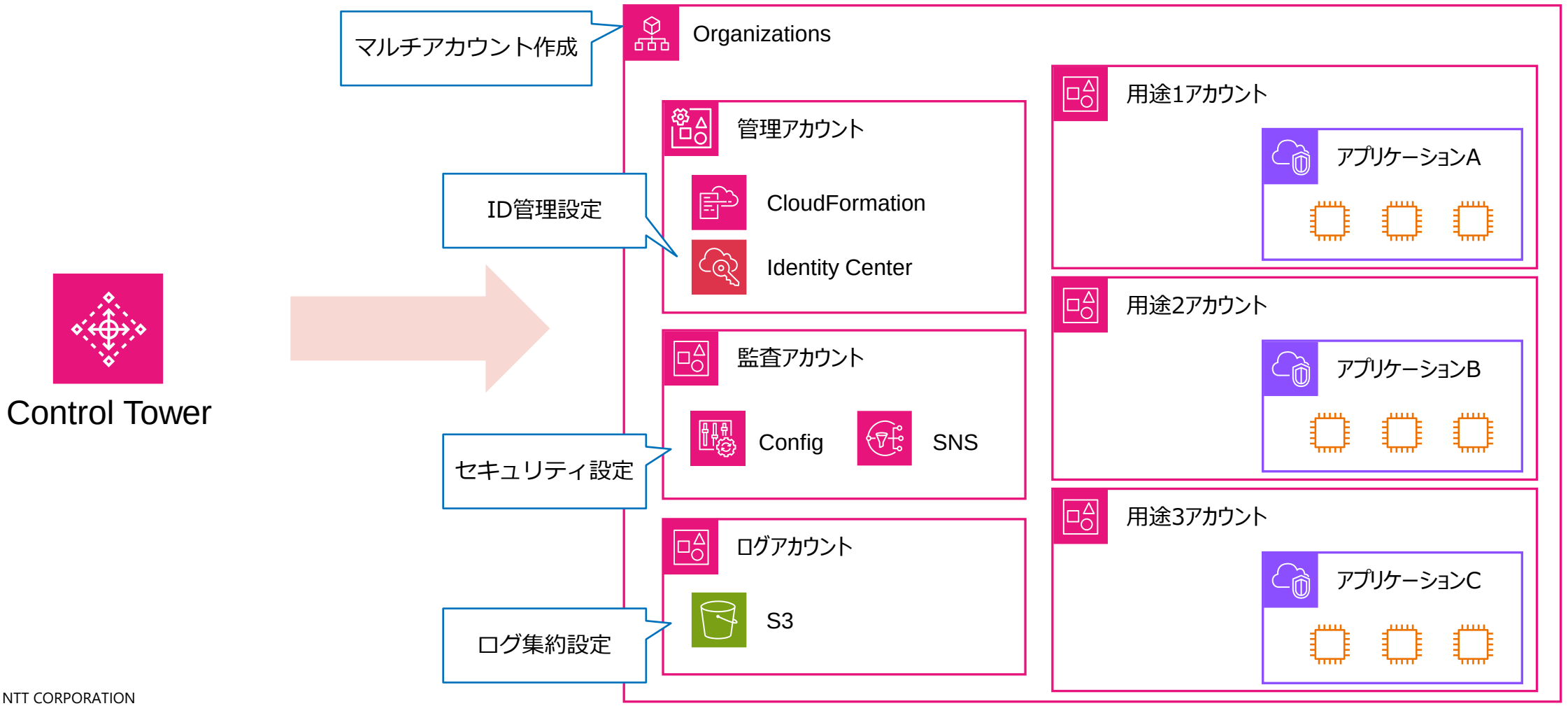
マルチアカウント自体の管理が大変・・・？
アカウント作成、ID管理、ログ管理、セキュリティ設定…　などなど



構築自体が大変

これを簡単にするサービスがControl Tower

Control Towerを使うとマルチアカウント管理を簡単に実現可能
AWSのベストプラクティスを盛り込んだ設定の適用が可能



自分の部署でAWSを利用する場合

新規で契約する場合 → Control Tower でランディングゾーンを作成
自社で Organizations を持っている場合 → 自社ルールで払い出し
（※持っているがIAMユーザ等レガシーな認証管理をしている場合 **Control Tower, Organizations, Identity Center** など導入を検討する）

新規で契約する場合

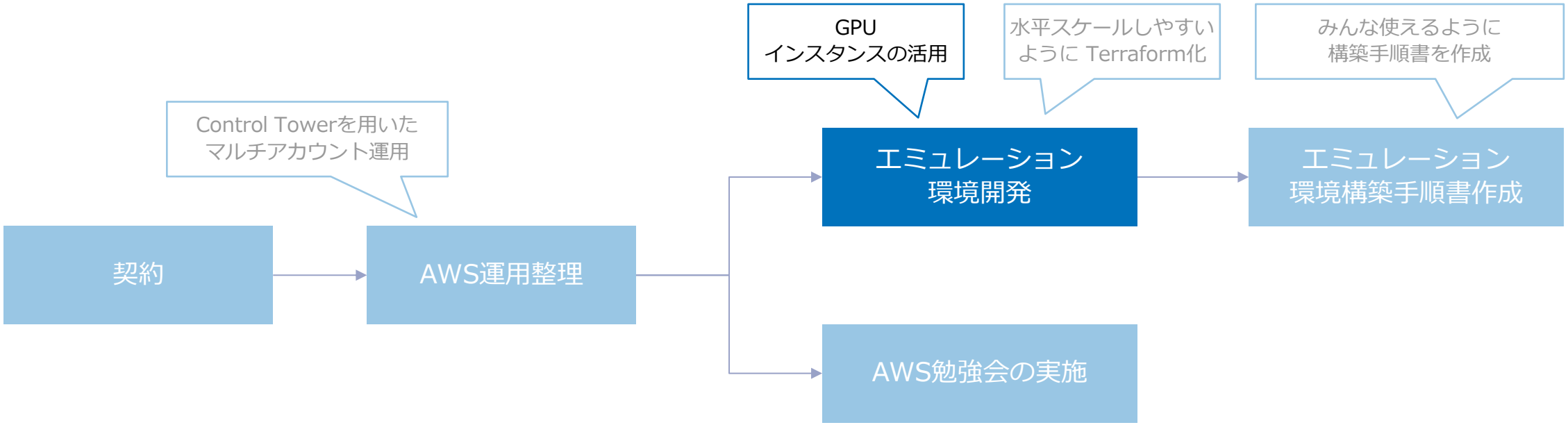
Control Towerでランディングゾーンを構築

自社で Organizations を持っている場合

自社ルールで払い出し

AWSをグループに導入する際の工夫

GPUインスタンスについてご紹介します



AWSでGPUインスタンスの活用

Service Quotaで上限緩和申請をすることで利用可能。（初期は0なので必須）
スポットインスタンスとオンデマンドインスタンスの上限緩和申請の項目が分かれているので注意。
申請はvcpu数で申請。

AWSでGPUインスタンス（スポット）

スポットインスタンスを活用することで半額程度でGPUインスタンスが活用可能
(総需要量が読めないため、Savings Plansは利用していません)

今回活用しているGPUインスタンス

NVIDIA L4 GPUを搭載している G6インスタンス(2024/4 GA)を活用しています。
L4は24GBのメモリをもっています。オンデマンドなら(1.16＄/時間)、スポットなら(0.6＄/時間)程度で活用できますので
AI利用など試験的にGPUを活用したい際は候補にあがるインスタンスかと思います。

Amazon EC2 G6 インスタンスの一般提供を発表

投稿日: Apr 4, 2024

本日、NVIDIA L4 Tensor Core GPU を搭載した Amazon EC2 G6 インスタンスの一般提供開始を発表します。G6 インスタンスは、グラフィックス集約型のユースケースや機械学習のユースケースに幅広く使用できます。Amazon EC2 G4dn インスタンスと比較して、G6 インスタンスでは深層学習推論とグラフィックワークロードのパフォーマンスが最大 2 倍向上します。

お客様は G6 インスタンスを使用して、自然言語処理、言語翻訳、動画と画像の分析、音声認識、パーソナライゼーションのほか、グラフィックワークロード (リアルタイムの映画品質のグラフィックやゲームストリーミングの作成・レンダリングなど) 用の ML モデルをデプロイできます。G6 インスタンスには、GPU あたり 24 GB のメモリを備えた最大 8 つの NVIDIA L4 Tensor Core GPU と、第 3 世代 AMD EPYC プロセッサが搭載されています。また、最大 192 の vCPU、最大 100 Gbps のネットワーク帯域幅、最大 7.52 TB のローカル NVMe SSD ストレージをサポートしています。

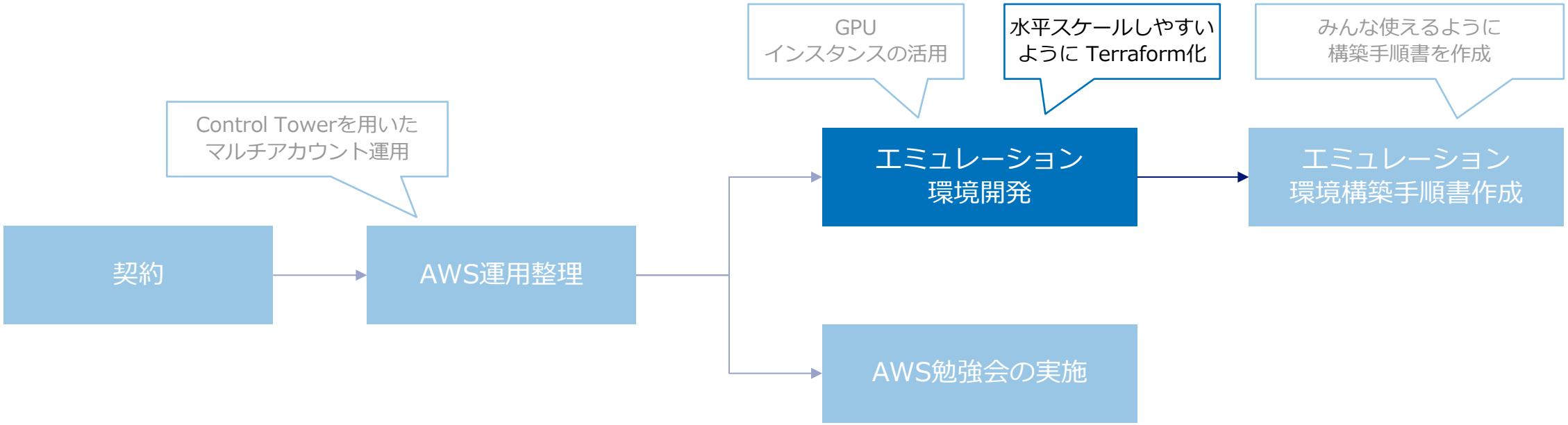
Amazon EC2 G6 インスタンスは、米国東部 (バージニア北部、オハイオ) および米国西部 (オレゴン) の AWS リージョンで本日よりご利用いただけます。G6 インスタンスは、オンデマンドインスタンス、リザーブドインスタンス、スポットインスタンス、または Savings Plans の一部としてご購入いただけます。

使用を開始するには、AWS マネジメントコンソール、AWS コマンドラインインターフェイス (CLI)、および AWS SDK にアクセスしてください。詳細については、G6 インスタンスのページをご覧ください。

https://www.nvidia.com/ja-jp/data-center/l4/

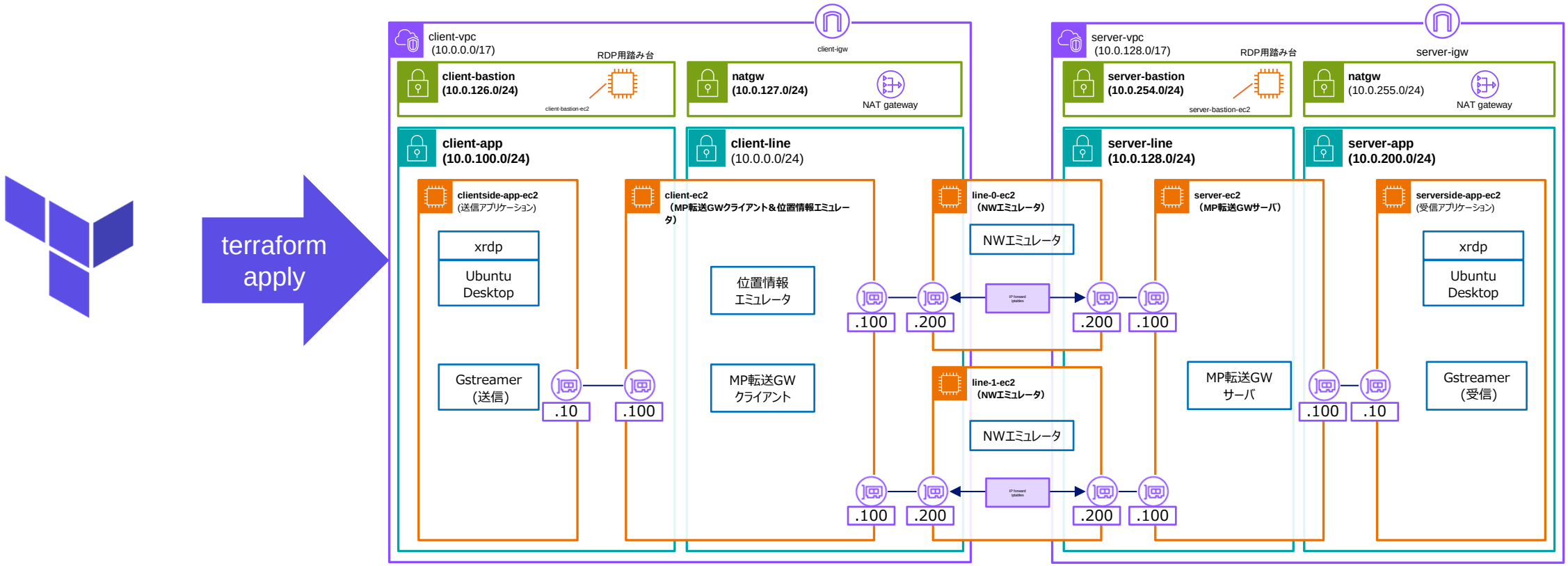
AWSをグループに導入する際の工夫

Terraform化についてご紹介します。



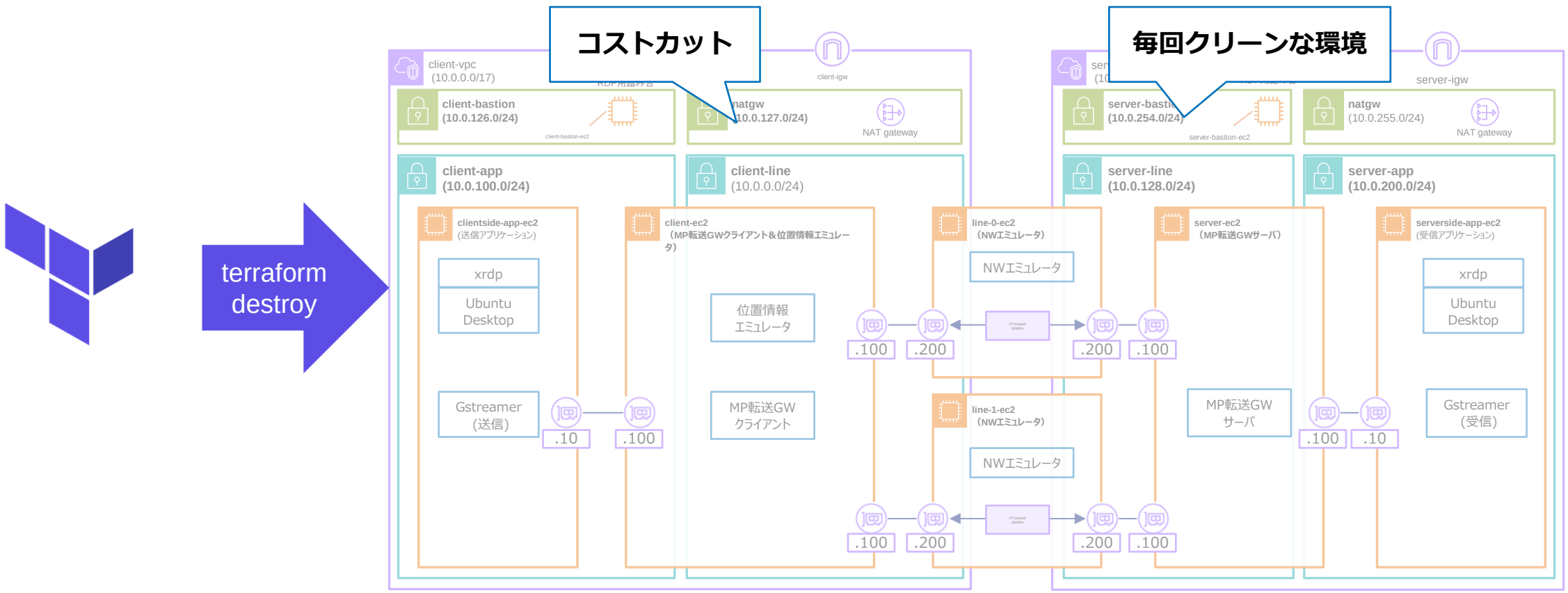
水平スケールしやすいように Terraform化

エミュレーション環境の部分を水平スケールしやすいようTerraformモジュールにしています。
apply一つで各インスタンスの設定を自動で行うように記載



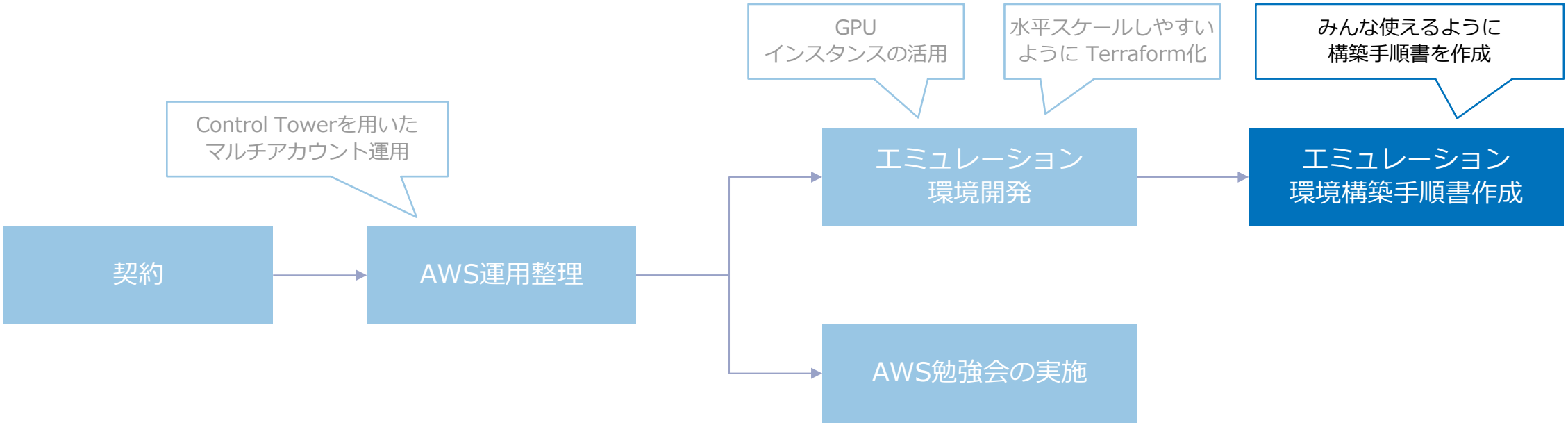
検証時だけ作成し、検証完了後削除することでコストカット、再現性の向上が可能

開発用途のため、検証時だけ作成し、検証完了時環境を削除することでコストカットを実現
毎回クリーンな環境を作成することで高い再現性を確保



構築手順書を作成

契約からエミュレーション環境の構築まで、様々な観点で工夫を実施



手順書の作成

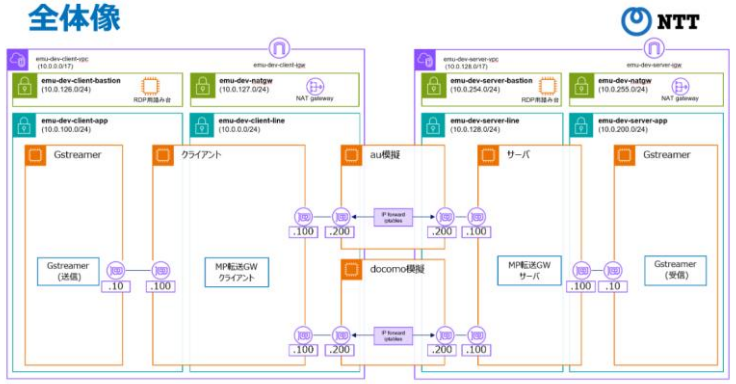
誰でも使えるようにマークダウンで構築手順書を作成。
Terraformのインストールから構築方法までまとめています。

構成図

エミュレーション環境の構成図

ここではざっくりとした構成図を記載しています。 各リソースについて詳しく見たい場合はTerraformの設定ファイルを確認してください。 また、実際のTerraform設定ファイルと以下内容に差分がある場合、Terraform設定ファイルの記述内容が正となります。

全体像



1.1. 各種EC2について

emu-dev-clientside-app-ec2（送信アプリケーション）

- OS: Ubuntu 24.10
- インスタンス: g6.xlarge
  - CPU
  - メモリ
  - GPU
- 追加設定
  - ハードウェアエンコード関連

Terraformインストール手順書

1. Terraformのインストール

1.1 TerraformのDL

https://developer.hashicorp.com/terraform/install?product_intent=terraform

1. 自身のPCに合わせたものをDL

1.2 zipを解凍し保存

1. 任意のフォルダに解凍

例:以下フォルダに解凍
C:/terraform/terraform.exe

1.3 解凍先のフォルダを環境変数PATHに設定

※Windowsの場合

- [Windows]+[R]で検索窓を開く
- sysdm.cplを検索しシステムのプロパティを開く

- 詳細設定タブの[環境変数]ボタンをクリック

構築手順書

0. 事前準備

1. 構成図の確認

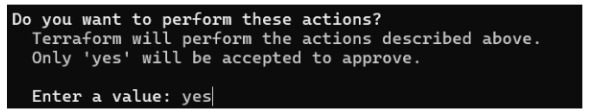
エミュレーション環境の構成図.md

2. Terraformのインストール

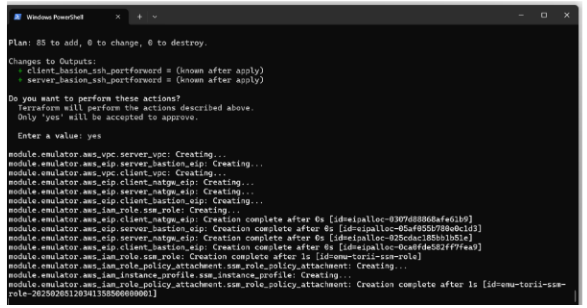
Terraformのインストール方法.md

3. Gitlabの権限の確認

4. 作成していいか聞かれるのでyesを入力



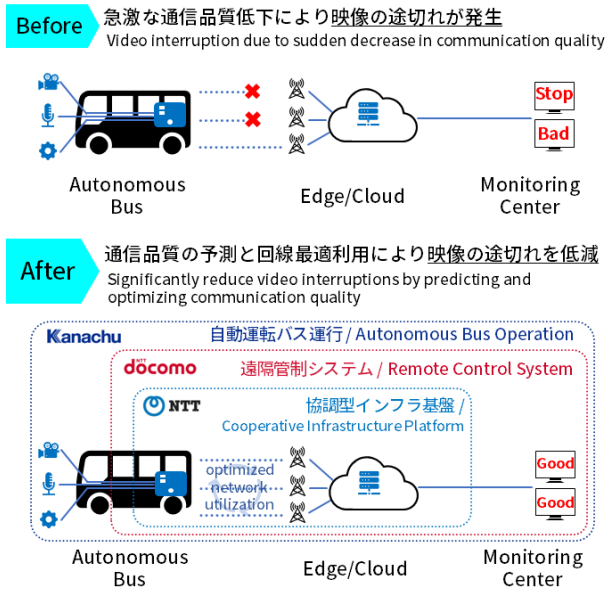
> 構築が始まる。



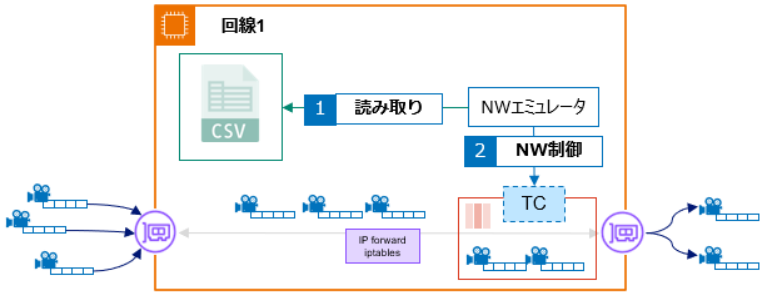
まとめ

グループの紹介、エミュレーションの紹介、AWS導入の工夫についてお話しました。
共有した内容が皆さんの業務で役に立てば幸いです。ご清聴ありがとうございました。

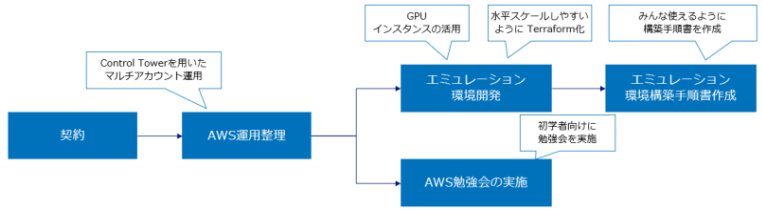
グループの紹介

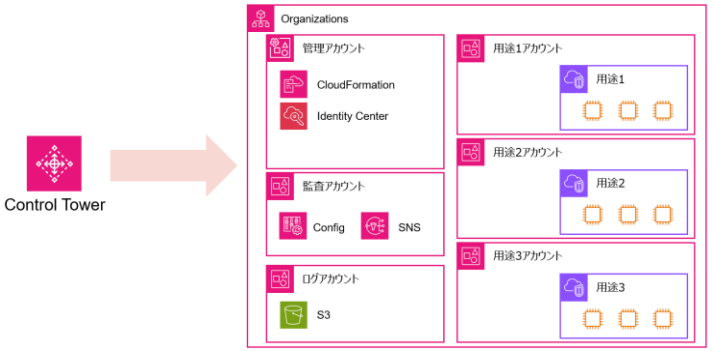


エミュレーションの紹介



AWS導入の工夫

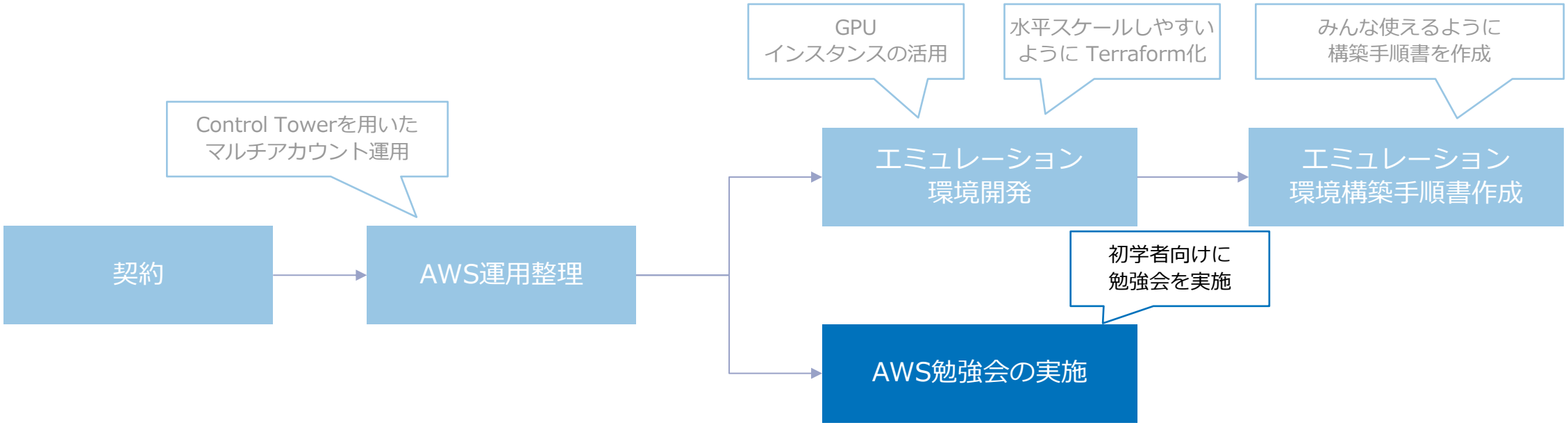




以下参考資料

AWS勉強会の実施

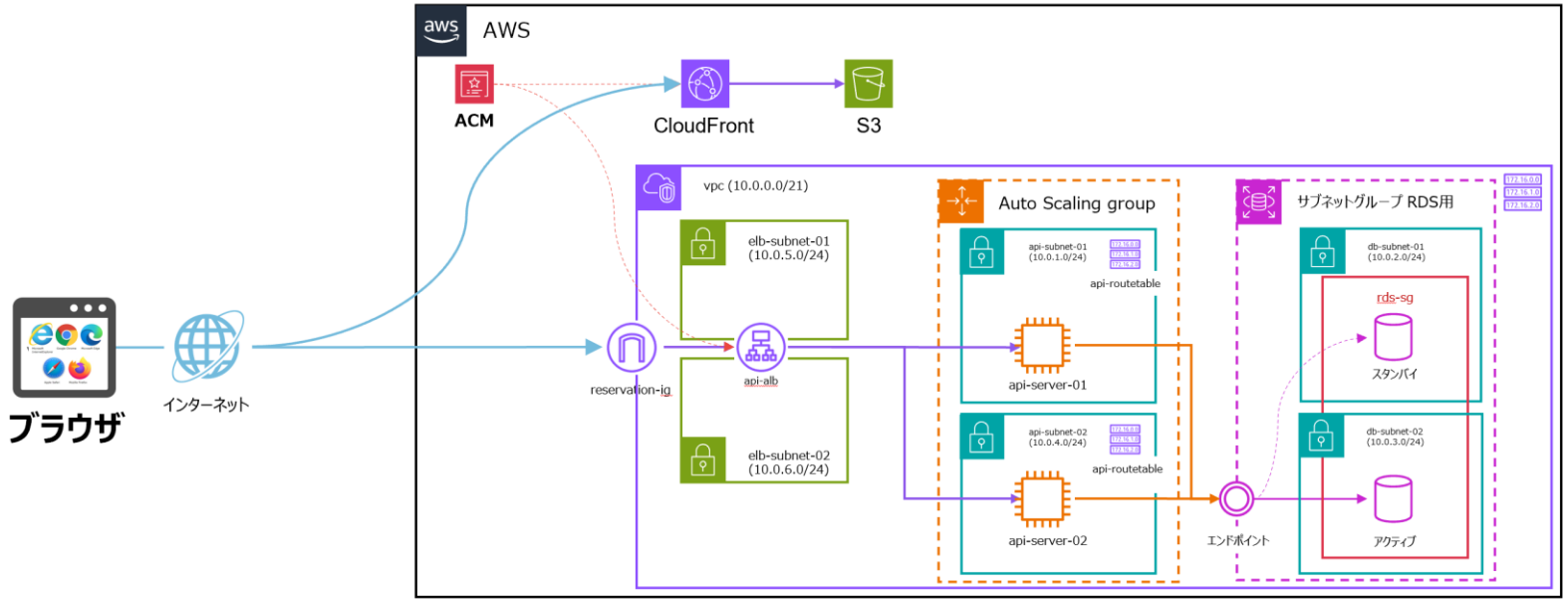
AWS勉強会についてからご紹介します



初学者向けにAWS勉強会の実施

グループでAWSを導入するのは初めてでしたので、勉強会を実施。
ハンズオン形式で各自Webサイトの構築をしてもらいました。

ハンズオン形式で一人一人Webサイトを構築しながら基本サービスの理解をできる

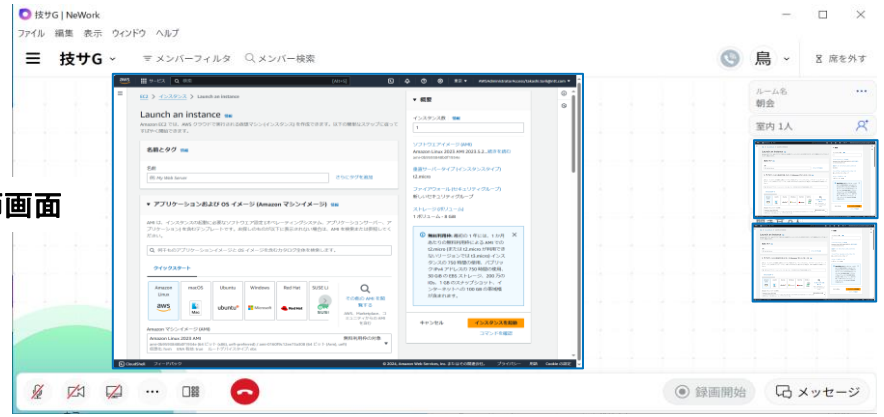


オンライン勉強会のナレッジ

ハンズオン形式の場合、全員画面共有が有効（NTTコミュニケーションズのNeWorkを活用）
手順書を作成したのも好評でした。

NeWork で 全員同時画面共有

イメージ



講師画面

参加者画面

参加者画面

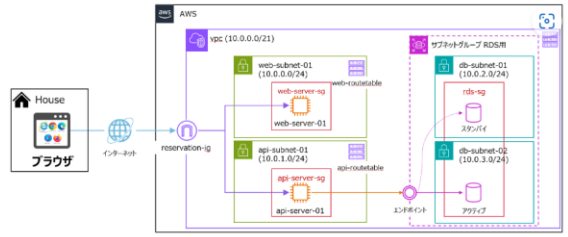
・会議ツールはNeWork
・全員同時画面共有が可能なため
・→Slackは2人まで、Teamsは1人まで

マークダウンで手順書作成

概要

ステップ1で構築した内容に＋αとしてデータベースを追加・デプロイします。

構成図



事前準備

ステップ1の内容が構築できていない方は先に構築しましょう。

AWS環境構築手順

1. サブネットグループの作成

1.1 db-subnet-01、db-subnet-02を作成

1. VPCのダッシュボードを開く
2. 左側のメニューから `サブネット` を選択
3. 右上の `サブネットを作成` をクリック
4. VPC IDに、 `handson-{自分の名前}-vpc` を選択