CUS-39

クラウド型 IT 資産管理サービスを、サーバレスアーキテクチャを活用してスケーラブルに構築する取り組み

森田 詳基
エムオーテックス株式会社
開発本部 サービス開発1部 サービス開発1課・課長

自己紹介

森田 詳基 (もりた よしき)

現在　　：エムオーテックス株式会社 2009年入社

　　LANSCOPE クラウド版 アプリ開発チーム

経歴　　：グループウェア開発・LANSCOPE オンプレミス版開発

アマゾン ウェブ サービス（AWS）歴：約4年

好きな AWS サービス： AWS Lambda

アジェンダ

- クラウド型サービス事業への挑戦
- クラウド型サービス・マルチテナント運用の課題 / 工夫
- 事例紹介「ログ受信基盤改善」
  - サーバーレスアーキテクチャを活用した取り組み
- まとめ

会社紹介

私たちは企業の**資産**である「**情報**」を守り

IT環境における**安全・生産性を追求**することで

社会の進歩発展に貢献します。

MOTEX

Secure Productivity

大阪に**本社**を置く、**ソフトウェアメーカー**です！

プロダクト紹介

LANSCOPE

ウイルス対策も情報漏洩対策もこの1本で

IT資産管理なのに、iOS・Androidにも対応
MDMなのに、Windowsの操作ログも管理できる

iOS　Android　Windows　MacOS

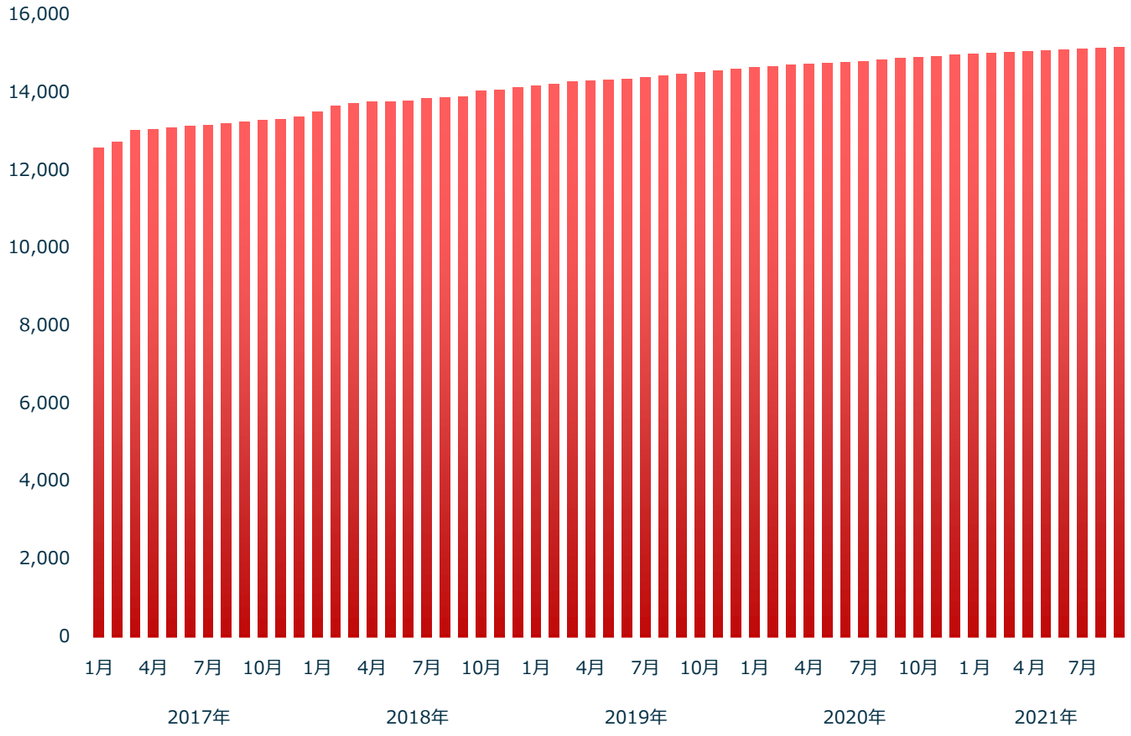
IT資産管理　内部不正対策　外部脅威対策

LANSCOPEオンプレミス版、クラウド版の実績

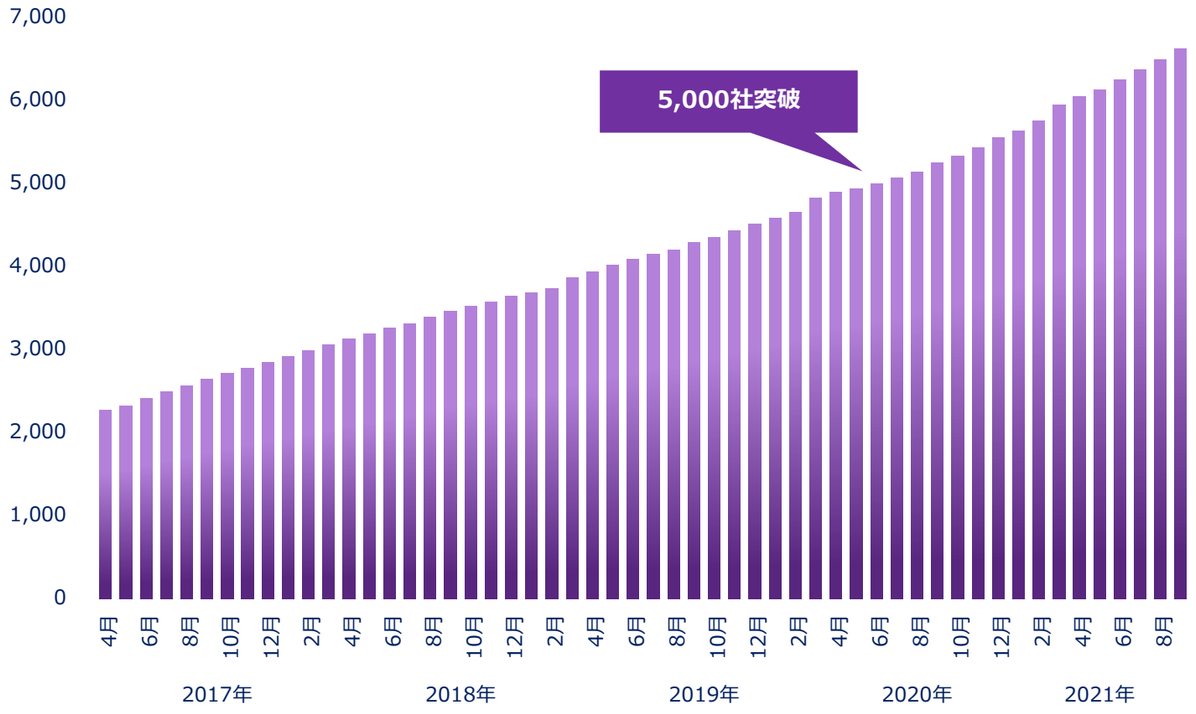
導入社数20,000社突破！導入後も93%以上の方が使い続けているLANSCOPEシリーズ

導入後も93%以上の方に使い続けていただける「製品力」と「サポート力」が強み

＜　LANSCOPEオンプレミス版　導入社数実績 推移　＞



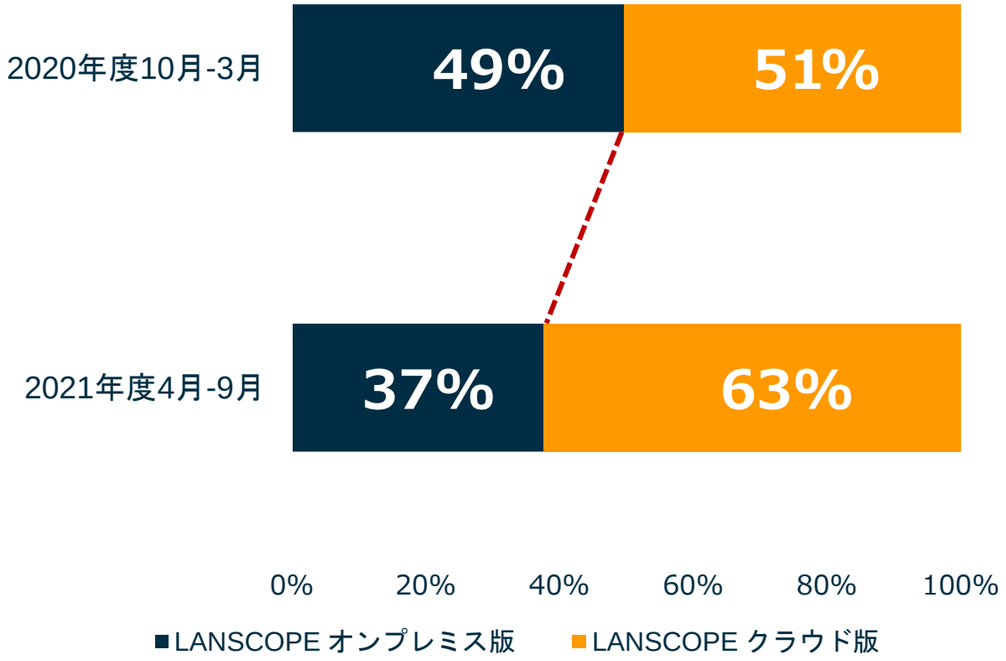
＜　LANSCOPEクラウド版　導入社数実績 推移　＞



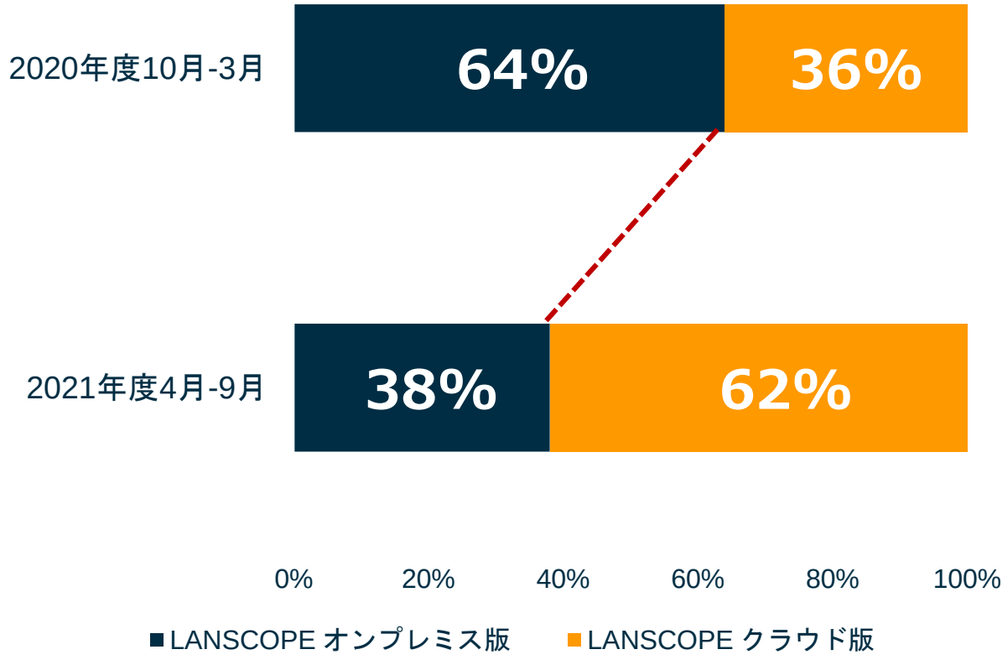
※2021年9月末現在

弊社におけるクラウドシフト傾向

■製品購入・検討に関する問い合わせの内訳



■LANSCOPE シリーズ 新規導入の内訳



製品導入に関するお問い合わせ・新規導入ともに、
クラウドサービス製品が増加している

クラウド型サービス事業への挑戦

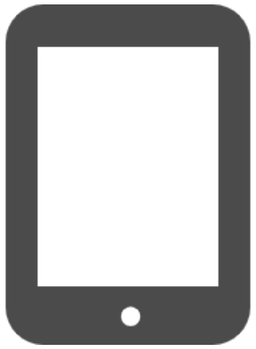
- LANSCOPEの歴史
- 2012 年、当時のアーキテクチャ
- 2018 年、スケーラブルな設計を考慮したアーキテクチャへ

LANSCOPEの歴史

PC管理ツール LanScope Cat をリリースして以降、24年間弊社の主力製品

1997 LanScope Cat

Cat



LanScope An

2012

An

2018

An 3.0

モバイルデバイス管理ツール
LanScope An をリリース
初めてのSaaS製品

マルチテナントへ移行
クラウド基盤もAWSへ移行

LANSCOPEの歴史

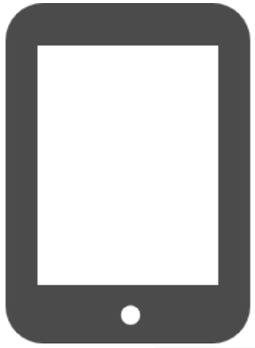
LanScope Cat

2021

LANSCOPE on-premises

リブランディング

Windows管理のノウハウ移植



2020

LanScope An

2021

LANSCOPE cloud

An 4.0

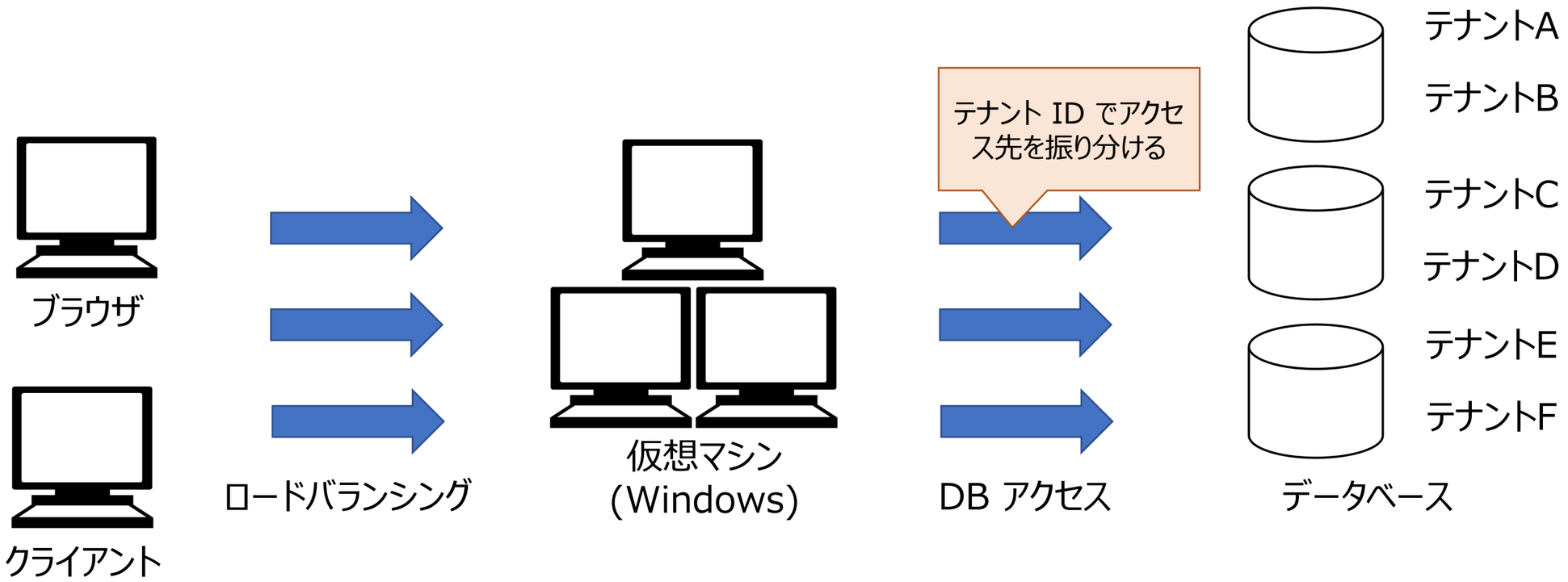
リブランディング

Windows管理機能を大幅強化

リブランディングを行い、オンプレミス版とクラウド版へ名称変更

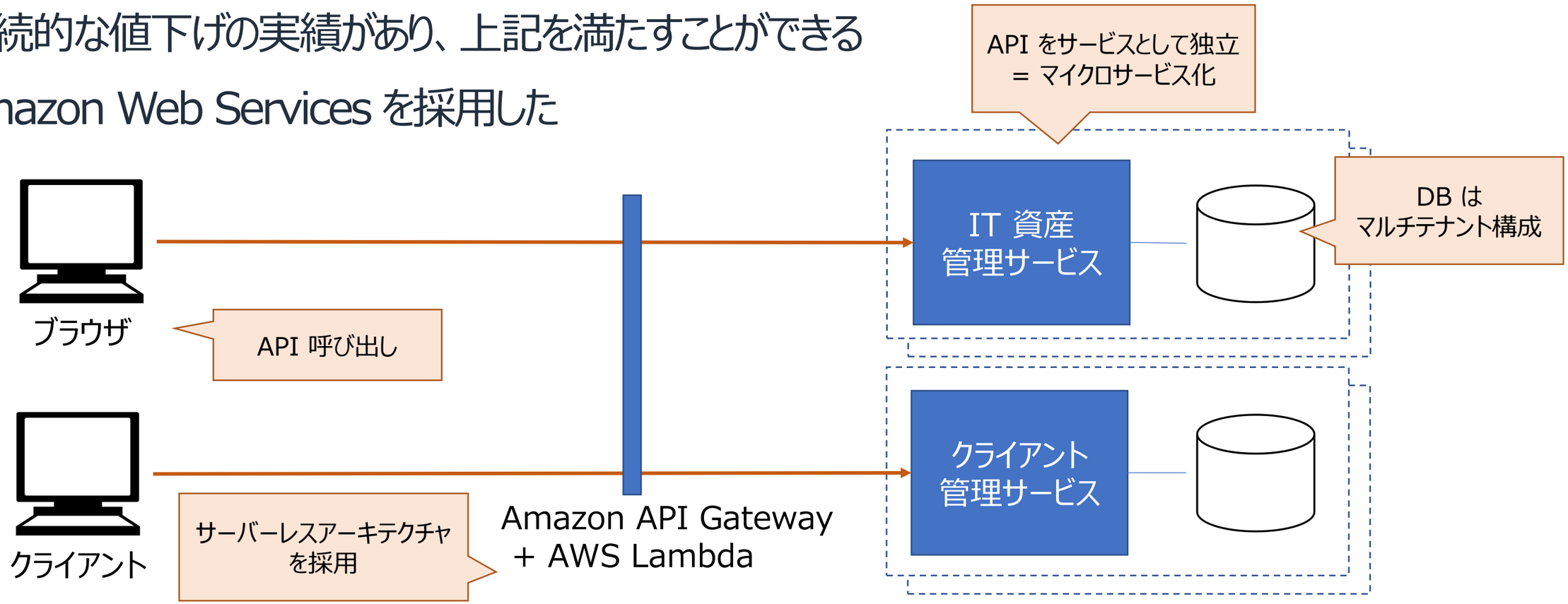
2012 年、当時のアーキテクチャ

- 2012 年、LanScope An をリリース
- オンプレミス版と同じシングルテナントに近い構成をとっており、一定のテナント数までは運用できていたが、データベースやテーブルが増え、Web サーバーも追加し運用コストが高くなり性能限界が見えてきた
- スケールアップして性能確保するにも CPU やメモリも限界がある



2018 年、スケーラブルな設計を考慮したアーキテクチャへ

- 性能問題を解決するため、テナントが増えてもコストをかけることなくスケーラブルに対応できるように、サーバーレスアーキテクチャを取り入れてオートスケーリング対応のマネージドサービスの活用を検討した
- また、コストメリットもあるマルチテナント型を採用した
- 継続的な値下げの実績があり、上記を満たすことができる Amazon Web Services を採用した



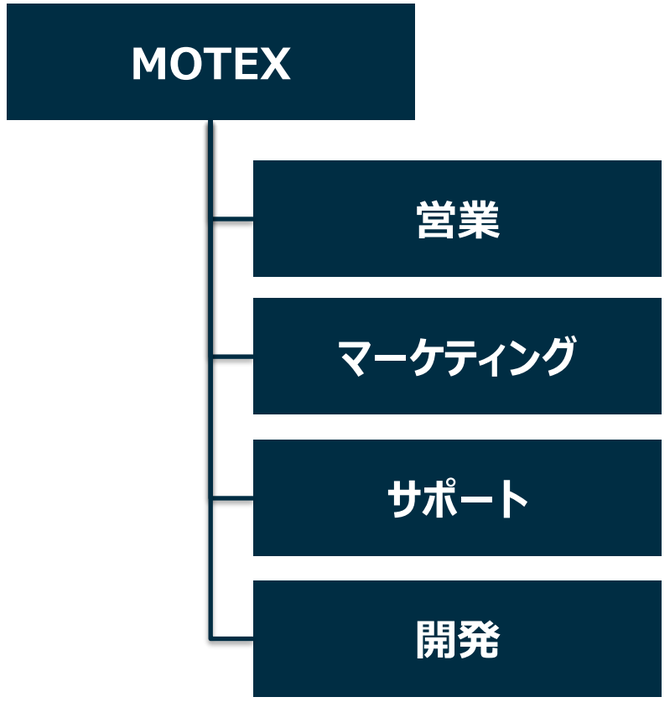
クラウド型サービス・マルチテナント運用の課題 / 工夫

- サービス立ち上げ時の組織体制
- AWS 移行時の組織体制
- テナント間でパフォーマンス影響を受けないようにするための対応
- サービスの運用監視コストを削減するための対応
- セキュリティや信頼性を可視化するための対応

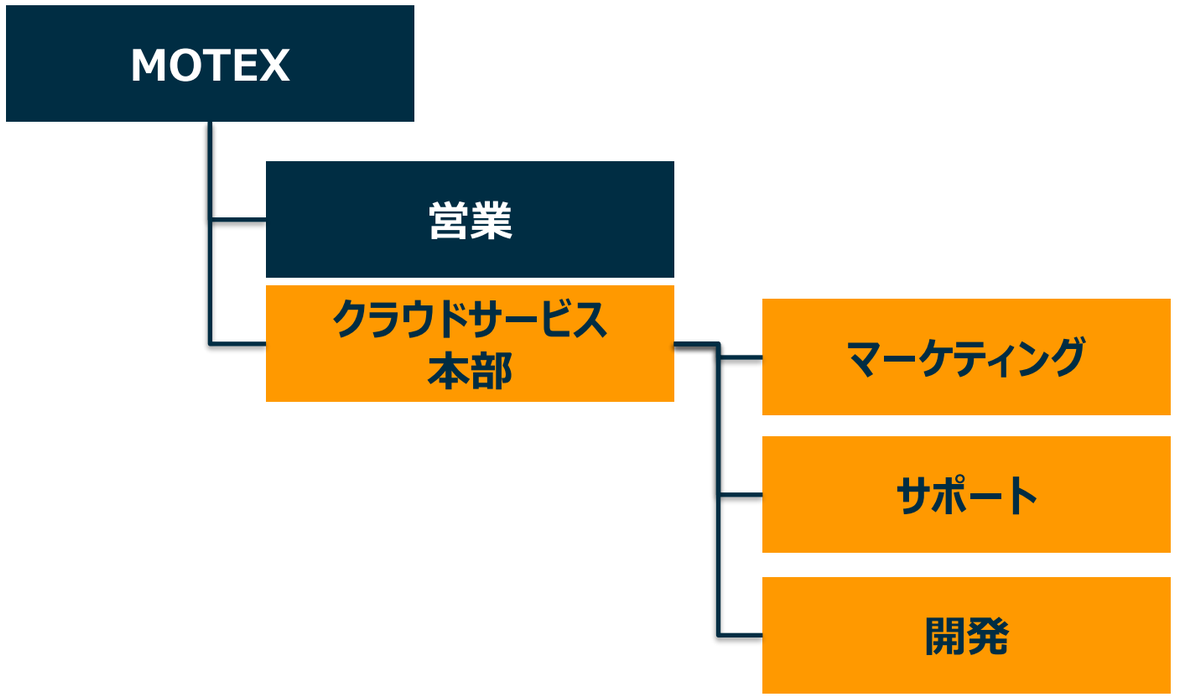
サービス立ち上げ時、既存事業主体の組織体制への対応

事業立ち上げと顧客要望反映を迅速に行うために、
製品開発・マーケティング・サポートをひとつの部門へ

■ MOTEX従来の組織構造



■ クラウドビジネス立ち上げを専任組織化

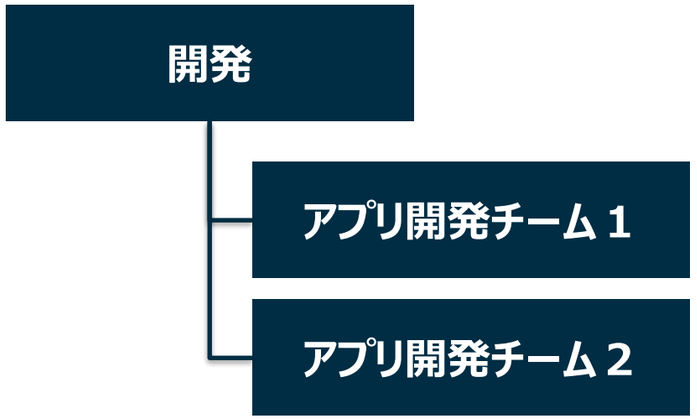


結果、毎月リリースを継続できている

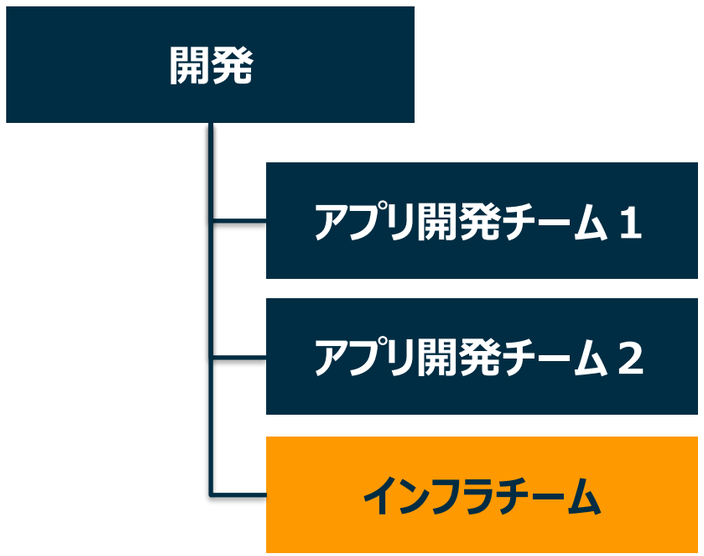
AWS 移行時、インフラ運用体制への対応

インフラ運用保守の効率化、専門化によるセキュリティ対策、コスト削減のために、インフラ運用体制を専任組織化

■ AWS 移行前の組織構造



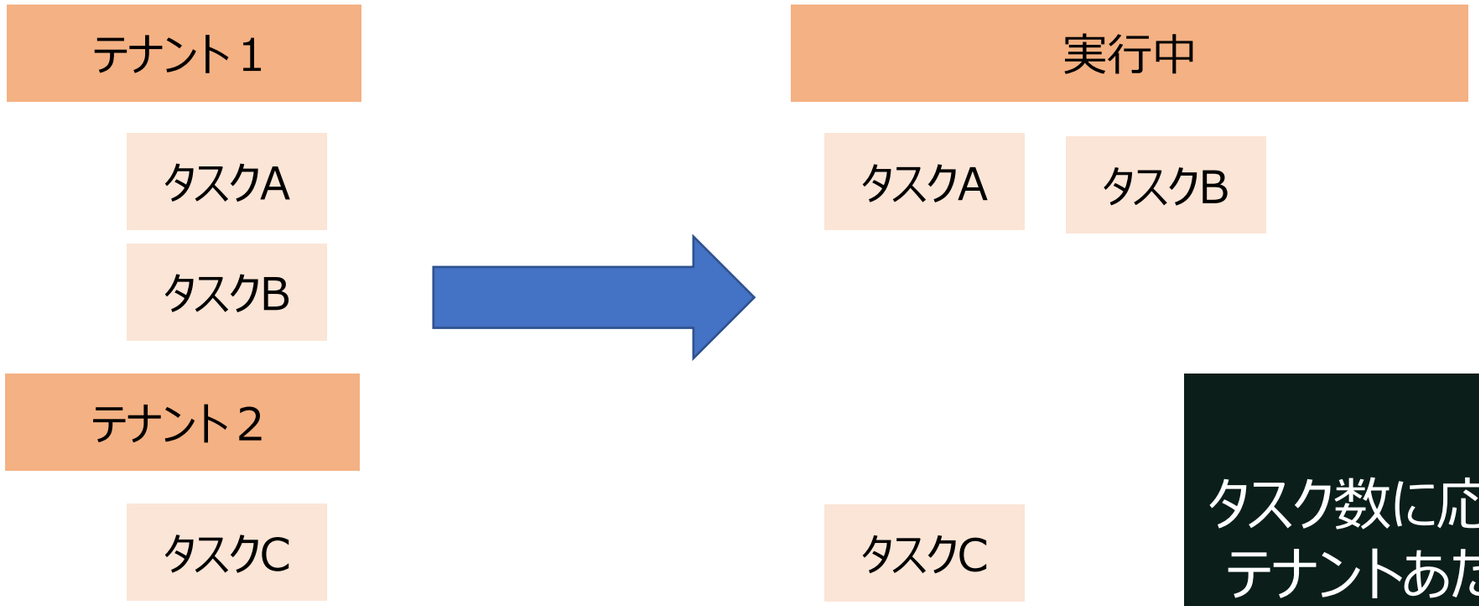
■ AWS 以降時にインフラ運用を専任組織化



インフラチーム主導によるインフラ運用の標準化、自動化を実施できている

テナント間でパフォーマンス影響を受けないようにするための対応

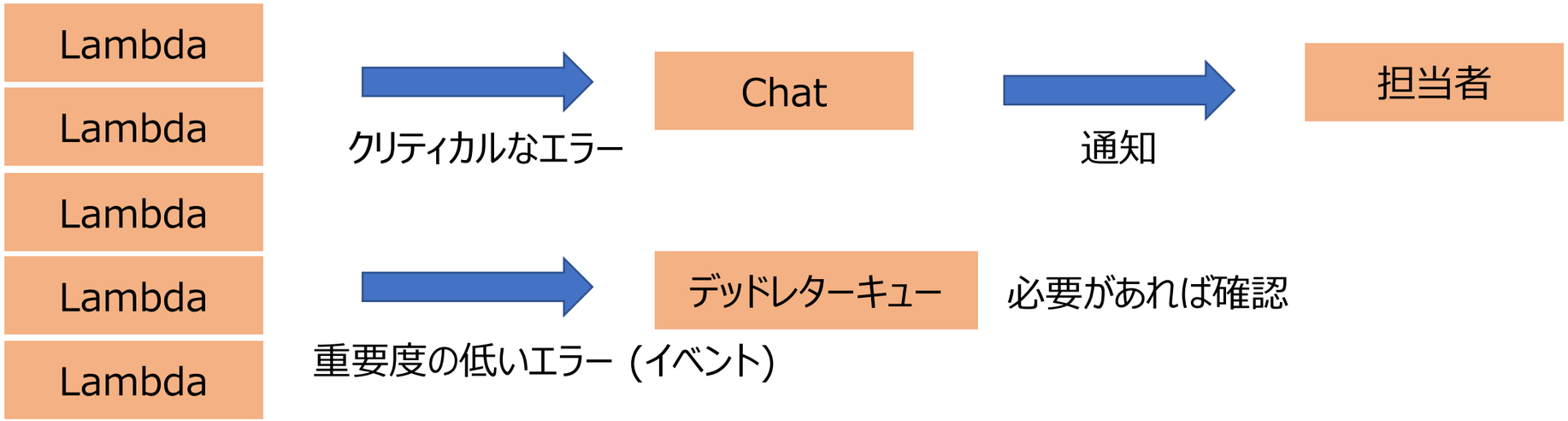
- マルチテナントはリソースを共有するモデル
- 高負荷な処理を実行するテナントが存在すると全体のパフォーマンスが低下してしまう可能性がある
- 負荷状況に合わせて自動でスケールアウトさせたり、テナント単位で同時に実行できるタスク数を制限するなどの仕組みが必要だった



例：
タスク数に応じて柔軟にスケールアウトするが、
テナントあたりの処理能力には上限を設ける

サービスの運用監視コストを削減するための対応

- サーバーレスアーキテクチャでマイクロサービスを作成した
- モジュール数が必然と多くなり、監視対象のアプリケーションもそれだけ増える
- 可能な限りシステムでリトライを行い、クリティカルなエラーのみ通知されるようにする
- 重要度の低いアプリケーションエラーは後で調査できるようにデッドレターキューにイベントをためておいて定期的にクリアして運用負荷を軽減する



セキュリティや信頼性を可視化するための対応

- モジュール数が多いと、チェック対象の DB やアプリケーションもそれだけ増える
- DR（ディザスタリカバリ）の計画策定や、新機能追加時に適切にセキュリティを考慮しているかなどの管理が大変だった
- AWS Foundational Technical Review (FTR) に通過することで、課題を把握・整理し、対策の実施、定期的な監査計画の策定などが実現できた
- ISO/IEC 27017（クラウドサービスセキュリティ）などの認証制度を取得する際にも役立つ

AWS ファンデーショナルテクニカルレビュー

https://aws.amazon.com/jp/partners/foundational-technical-review/

事例紹介「ログ受信基盤改善」

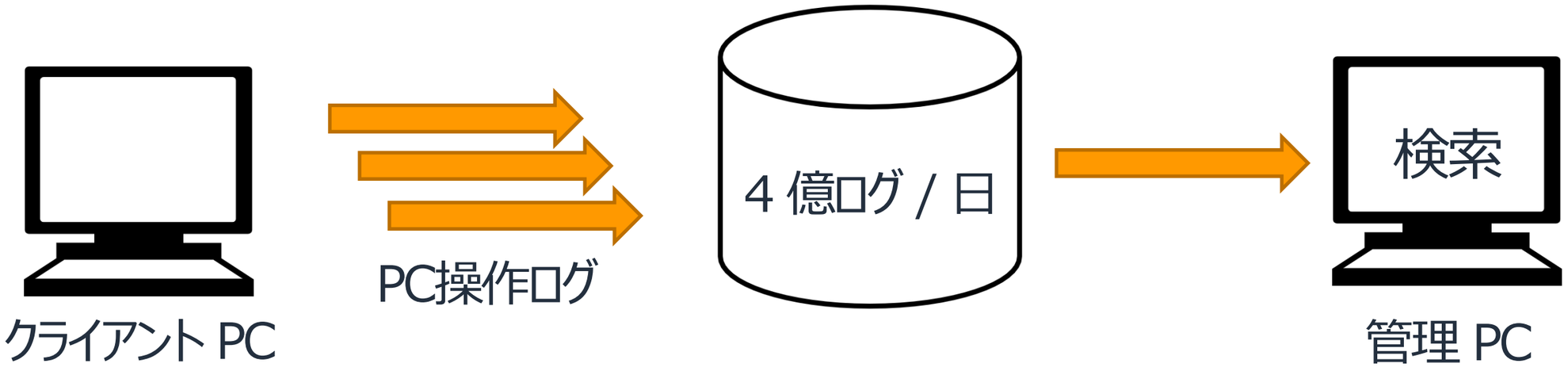
- プロジェクト概要
- 2020年、改善前のアーキテクチャ
- 2022年、改善後のアーキテクチャ
- 工夫ポイント
- 改善後の効果

プロジェクト概要

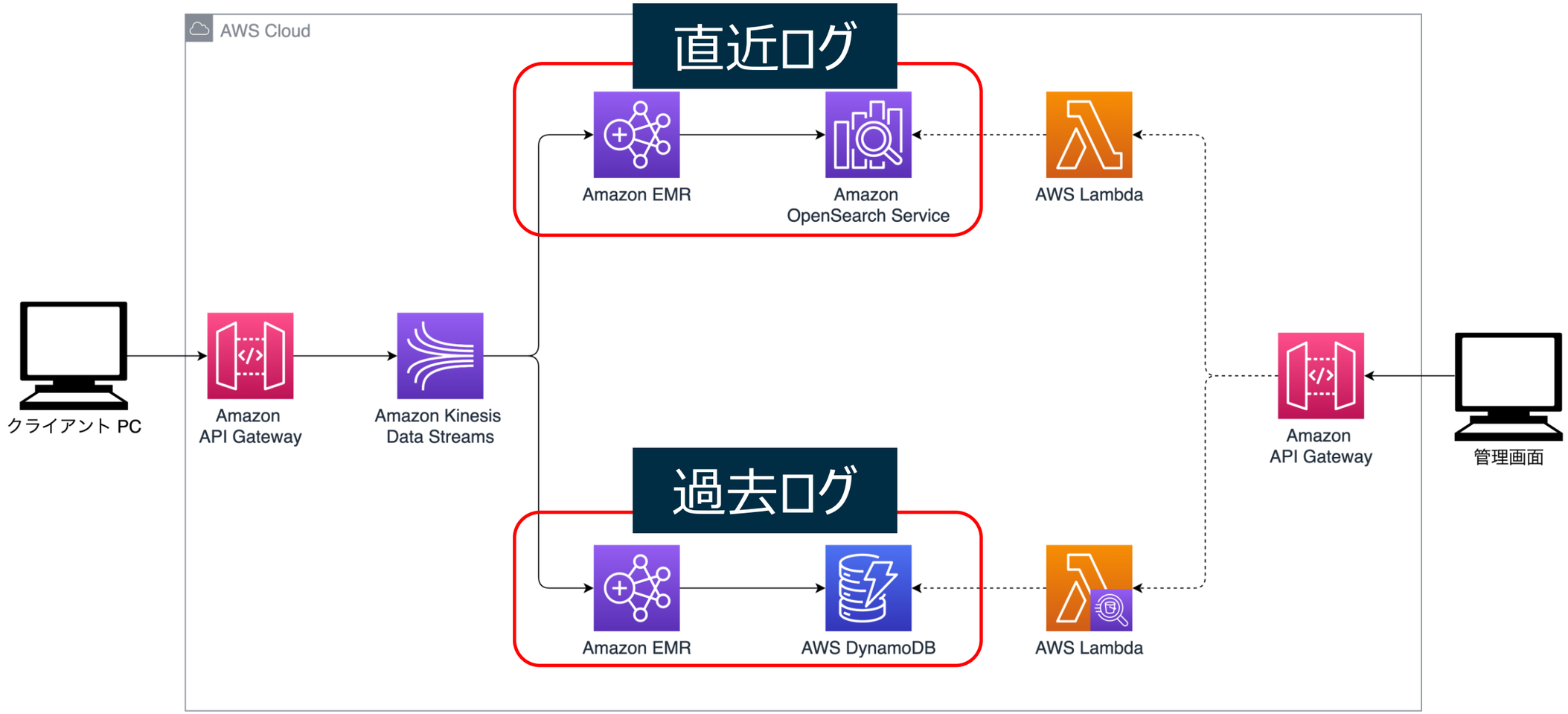
- IT 資産管理ツールの機能として、お客様の PC 操作をログとして取得している

  アプリケーション利用ログ や Web 閲覧ログなど

- クライアント PC からの操作ログ（データ）を受信し、検索用に前処理するアーキテクチャを改善する

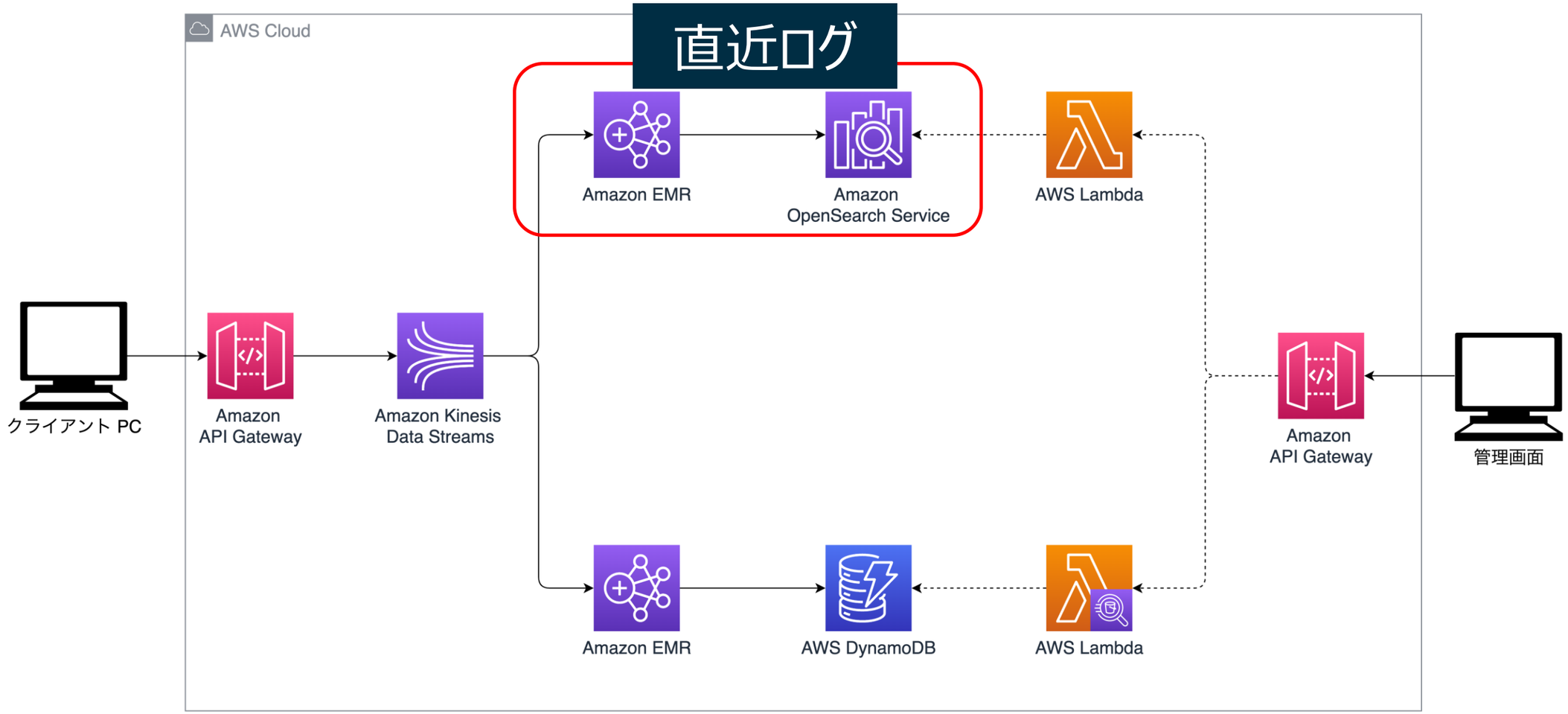


2020年、改善前のアーキテクチャ



今日を含む4日分のログ(直近ログ) と、100日保存用のログ(過去ログ) を
ストレージを分けて保存している

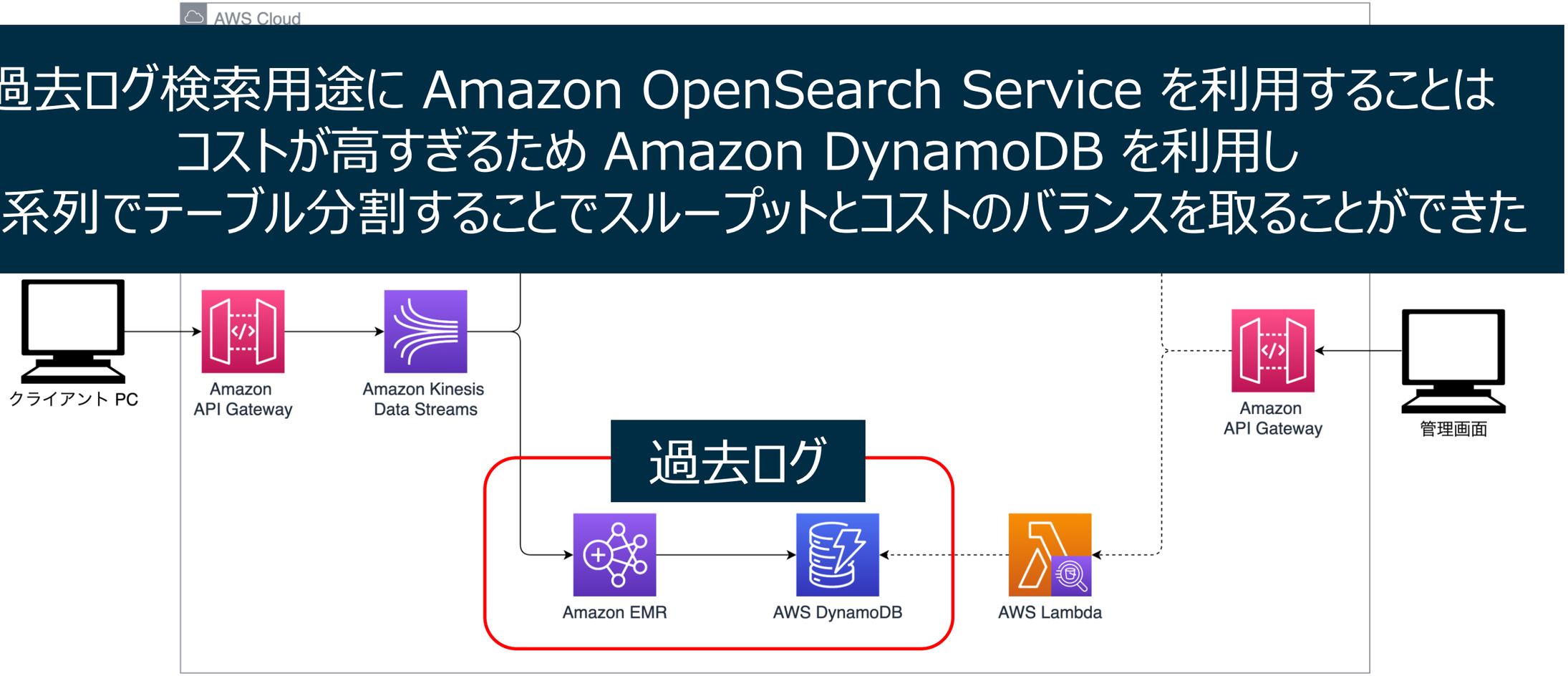
2020年、改善前のアーキテクチャ



直近ログは検索用全文検索エンジンに Amazon OpenSearch Service を利用して高速に検索できるように設計した

2020年、改善前のアーキテクチャ

過去ログ検索用途に Amazon OpenSearch Service を利用することは
コストが高すぎるため Amazon DynamoDB を利用し
時系列でテーブル分割することでスループットとコストのバランスを取ることができた



2020年、改善前の初期設計時の事情

- クライアント PC からは ログが複数まとまって送信されるため、保存の際には分割する変換処理が必要
- Amazon Kinesis Data Streams + AWS Lambda で変換処理するには相対的にコストが高く考慮対象から外れた
- コスト削減のために、自前で Kubernetes + Apache Kafka + Apache Flink を Amazon EC2 上で構築することも検討したが、運用コストが高く考慮対象から外れた

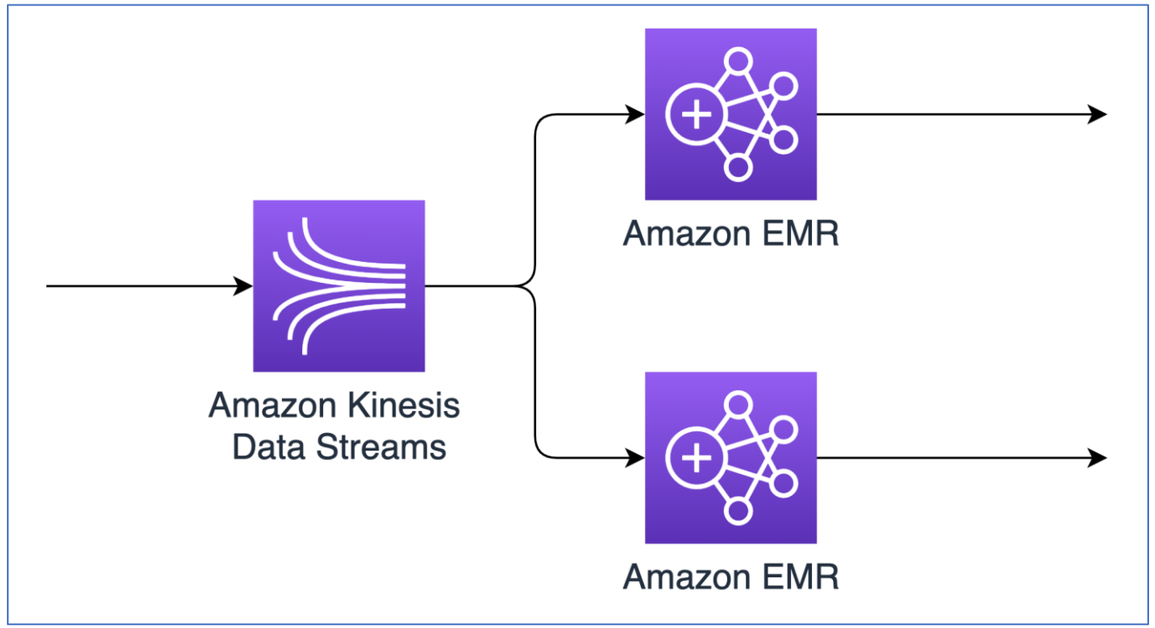
AWS Lambda はサービスアップデートによって現在はコスト効率が良くなっている

- 2020年2月：Savings Plan のアップデート: AWS Lambda ワークロードで最大 17% の節約
  https://aws.amazon.com/jp/blogs/news/savings-plan-update-save-up-to-17-on-your-lambda-workloads/
- 2020年12月：New for AWS Lambda – 1ms Billing Granularity Adds Cost Savings
  https://aws.amazon.com/jp/blogs/aws/new-for-aws-lambda-1ms-billing-granularity-adds-cost-savings/

2020年、改善前の初期設計時の事情

Amazon Kinesis Data Streams + Spark Streaming on Amazon EMR の組み合わせを採用した

- Amazon EC2 を直接管理することなく利用可能だった
- コストパフォーマンスも良かった
- Spark Streaming は Scala による実装が行えた



2020年、改善前の初期設計時の事情

	AWS Lambda	Kubernetes + Apache Kafka + Apache Flink on Amazon EC2	Spark Streaming on Amazon EMR
構築容易性	○	×	○
運用容易性	○	×	○
AWS コスト	×	○	○
選択肢の補足	当時はコストが高かった	構築・運用コストが高い	Scala で実装可能

2020年、運用開始後に浮き彫りになった課題

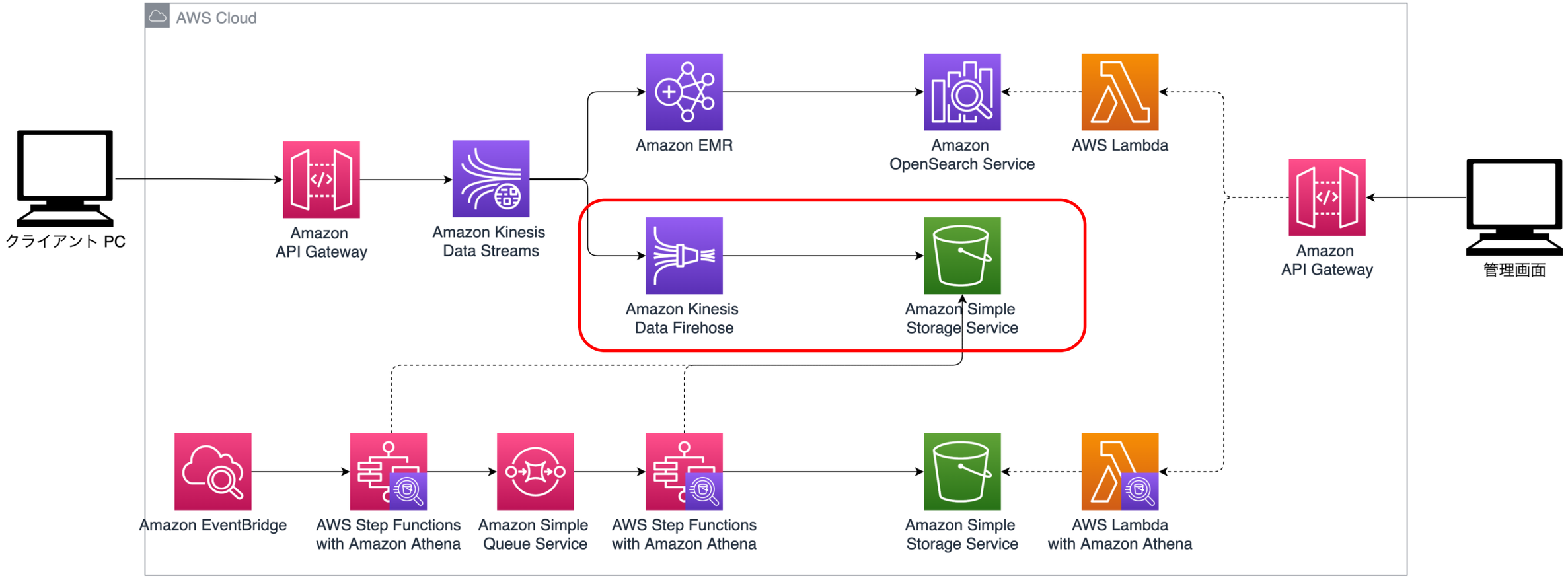
- クライアント PC あたりの実際のログ量は、2020 年アーキテクチャ選定の検証時の想定よりも多かった
- BatchWriteItem の制限に引っかかり、データの再送が発生した
- Amazon DynamoDB のパーティションキーの選択が適切ではなかった
- 結果、特定のパーティションに偏りが出るようになり、Amazon DynamoDB への書き込みスロットルが多発してパフォーマンスが低下してしまった
- ログ保存期間を 5 年に延長する機能改修を行う必要もあり、アーキテクチャを見直す必要が出てきた

- BatchWriteItem ※1 リクエスト 25 アイテム (合計サイズ 1 MB まで)。1 アイテム 64 KB まで

  https://docs.aws.amazon.com/amazondynamodb/latest/APIReference/API_BatchWriteItem.html
- 書き込みシャーディングを使用してワークロードを均等に分散させる

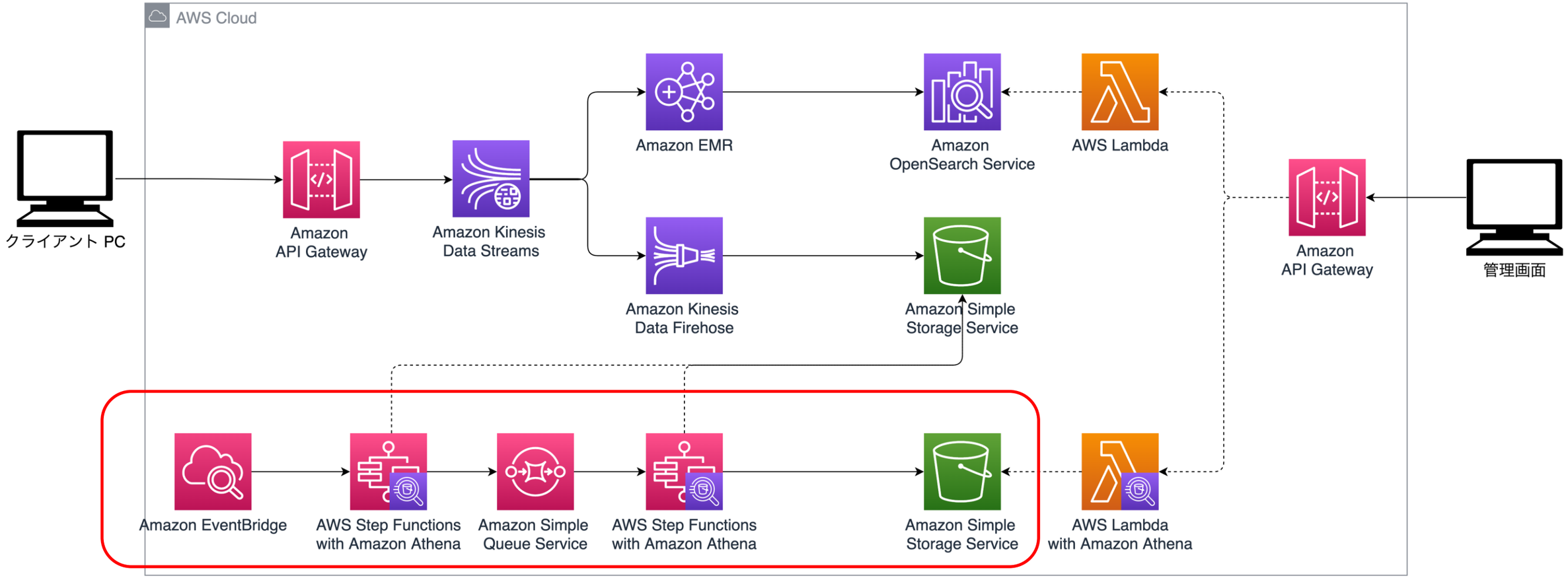
  https://docs.aws.amazon.com/ja_jp/amazondynamodb/latest/developerguide/bp-partition-key-sharding.html

2022年、改善後のアーキテクチャ



過去ログの保存先を Amazon DynamoDB から Amazon S3 に変更した
しかし、そのままの形式では検索しにくい

2022年、改善後のアーキテクチャ

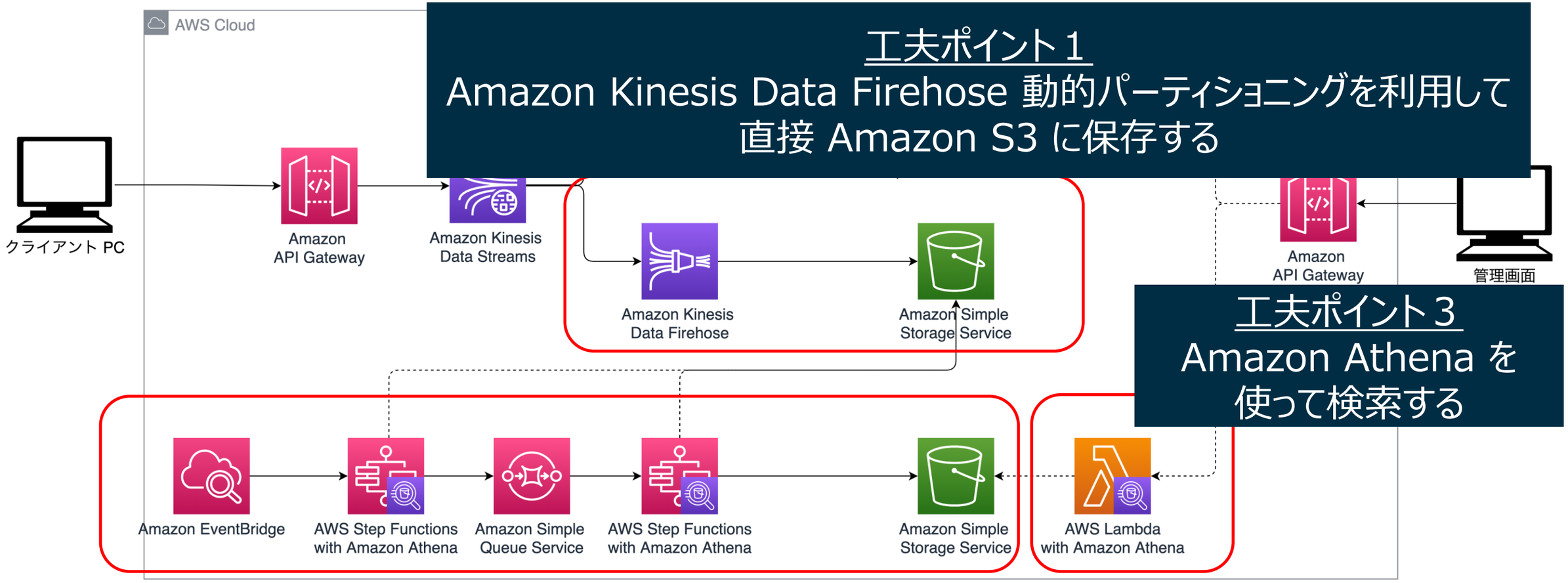


AWS Step Functions + Amazon Athena を利用して
ログを検索しやすい形式に変換し検索用の Amazon S3 へ保存する

2022年、改善後のアーキテクチャ

	Spark Streaming on Amazon EMR	Amazon Kinesis data Firehose + Amazon S3 + Amazon Athena
構築容易性	○	○
運用容易性	○	◎
AWS コスト	○	◎
選択肢の補足	Scala で実装可能	課題解決 ＋ 運用・AWS コストでメリット大

2022年、改善後のアーキテクチャ



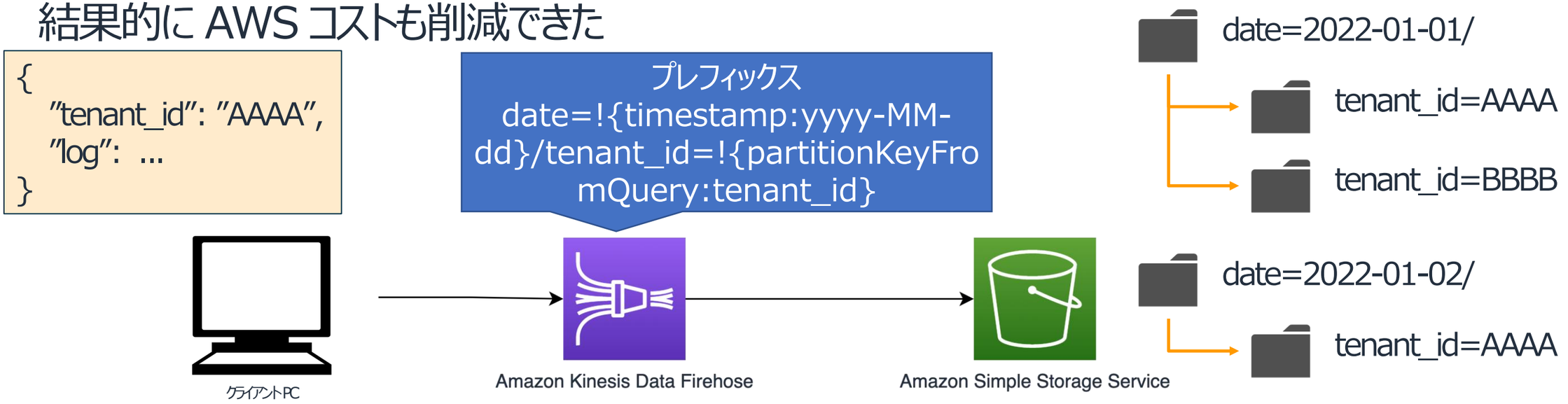
工夫ポイント１
Amazon Kinesis Data Firehose 動的パーティショニングを利用して
直接 Amazon S3 に保存する

工夫ポイント３
Amazon Athena を
使って検索する

工夫ポイント２
Amazon Athena で変換処理対象日の一覧を取得、
SQS でテナントごとに並列化し、Amazon Athena CTAS でクエリした結果を
検索用 Amazon S3 に保存する

工夫ポイント１ Amazon Kinesis Data Firehose 動的パーティショニング

- Amazon Kinesis Data Firehose 経由で直接 Amazon S3 に保存することでパフォーマンスを向上しコストを低減した
- Amazon Kinesis Data Firehose の動的パーティショニングで、テナント・ログの日付ごとに Amazon S3 に保存することで、その後の変換処理と検索を効率化した
- AWS Lambda や AWS Glue での変換が不要となり、構築・チューニングなど運用の手間がかからず、結果的に AWS コストも削減できた



※timestamp 名前空間を使用

工夫ポイント2 Athena CREATE TABLE AS SELECT (CTAS) クエリ

- 変換処理は Amazon Athena の CTAS クエリで 1 日 1 回処理できるように最適化した
- ステート管理に AWS Step Functions を利用、実装や運用の容易性を向上した
- パーティション数の制限に抵触しないよう、テナント単位でテーブルを作成し日付でパーティショニングした
  （日付ごとに 1,825 日: 365日 * 5 年 保存する必要があった）
- テナント + 日付で検索できるため、大量データの検索を効率化できた
- AWS Glue を使わず Amazon Athena のみで変換処理を実装することで、ジョブの並列実行やジョブフローの定義など複雑な制御が不要となり効率的に実装できた
- 並列処理することで、失敗した場合にテナント単位でリトライでき、他テナントに影響がない

工夫ポイント3 Amazon Athena のワークグループ

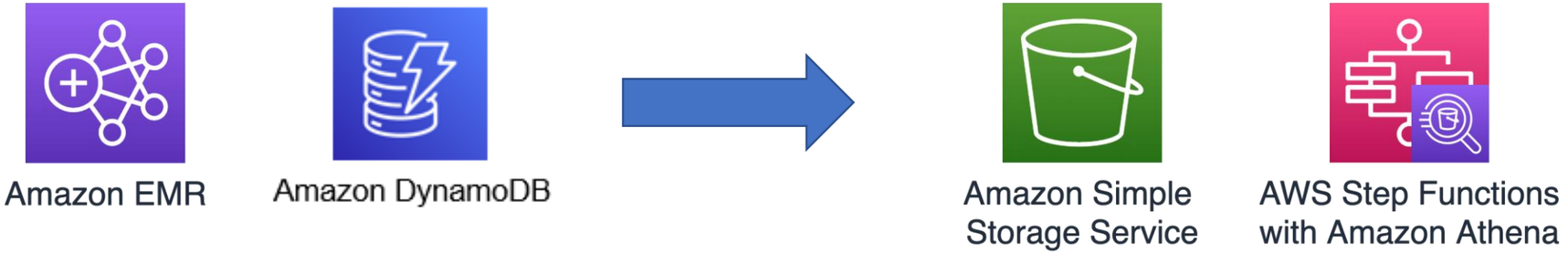
- 検索にも Amazon Athena を利用した
- Amazon Athena のワークグループでスキャンされるデータの合計量を制限することで、一定のスループットを超えないように制御した
- 複数のワークグループを作成して検索 (クエリ) のアクセスごとに割り振ることで、一部のテナントで高負荷が発生することを防ぎ、全体のパフォーマンスが低下しないように設計した
- 意図しない制限によるクエリキャンセルが発生することを防ぐために、CloudWatch Alarm の設定も併せて行うことが推奨されている（今後検討予定）

データ使用量の制御制限の設定 | Amazon Athena
https://docs.aws.amazon.com/ja_jp/athena/latest/ug/workgroups-setting-control-limits-cloudwatch.html

改善後の効果

「Amazon DynamoDB」から「Amazon S3 + Amazon Athena」に変更しての改善効果

- PC操作ログ保存のパフォーマンスをあげることができた
- リアルタイムに変換処理を行っていたが、１日１回 Amazon Athena で変換するだけで十分になった
- 検索にも Amazon Athena を採用したことで画面からのデータ検索も高速にできた
- 構成にかかるコストが４分の１にできた
- Spark Streaming on Amazon EMR のパフォーマンス最適化のケアを自前で行う必要があったが、２つから１つになったため運用コストを半減できた



所感

- 変換・検索の Amazon Athena は当初の想定を上回る性能を発揮した
- Amazon Kinesis Data Firehose の動的パーティションを利用することで実装がとても楽になった
- Amazon Athena のテーブル設計やパーティション設計を慎重に実施すれば、実装効率が良かった
  - Amazon Athena のクエリを記述するだけで変換処理が実装できた
- サービスアップデートをキャッチアップすることで設計の選択肢が増えることを実感した
  - Amazon S3 + Amazon Athena の組み合わせのアイデアが生まれた
- 今回の改善は開発チームの現場からの提案で実現できた。こういったサイクルを継続したい

まとめ

まとめ

- AWS のマネージドサービスを活用しているため、提供するサービスの要件の変更にあわせて部分的な再実装やマイグレーションも柔軟に実施できた
- AWS の新機能のキャッチアップも重要です！
- お客様の選択肢を増やすためにも、クラウドの選択肢を製品に検討しましょう
- AWS SA さんのサポートを最大限に活用しましょう（ありがとうございます）

全てはお客様のために、クラウドという選択肢を

採用してます！

- カジュアル面談やってます！

https://motex.snar.jp/index.aspx

MOTEX

Secure Productivity

Thank you!

森田 詳基

エムオーテックス株式会社
開発本部 サービス開発1部 サービス開発1課・課長