画面に映る資料の撮影などによる本セッション資料の転用を禁止しております

CUS-28

DevOpsチーム構築のための AWSサーバーレス起点の技術ガイドライン策定

山根　径

トランスコスモス株式会社　執行役員

画面に映る資料の撮影などによる本セッション資料の転用を禁止しております

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

自己紹介

山根　径

トランスコスモス株式会社　執行役員

サービス推進総括デジタルテクノロジー推進本部副本部長

兼 サービス推進総括デジタルテクノロジー推進本部TCIプラットフォーム 推進統括部長

兼 グローバル事業統括付

兼 SOCIAL GEAR PTE LTD CEO & CTO

トランスコスモスグループの国内・海外の事業で使われるIT技術基準を策定して
その基準に基づいた開発体制を構築するプロジェクトの執行責任者で
今はシンガポールの拠点にいます

© 2021, transcosmos, inc. or its affiliates. All rights reserved.

トランスコスモス

会社概要

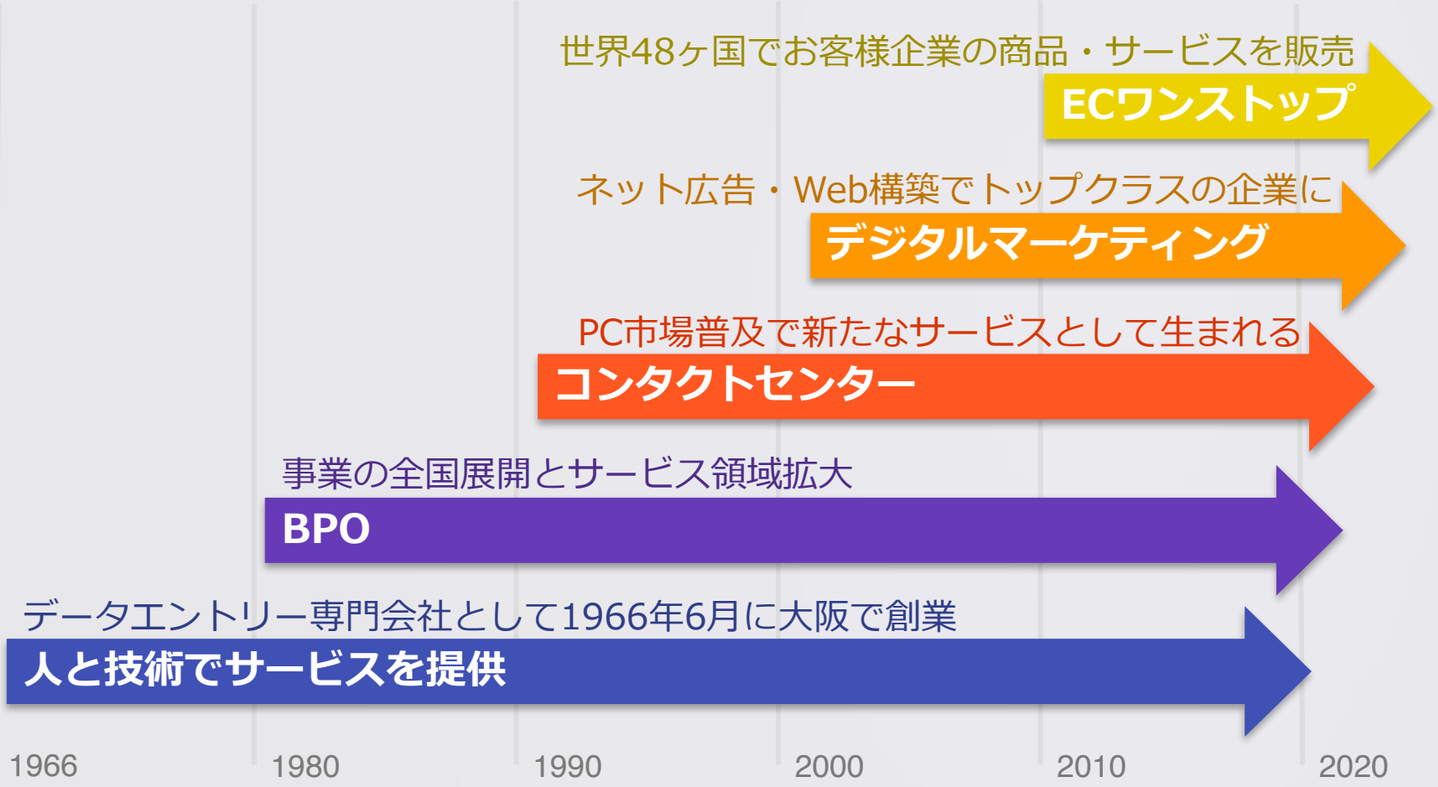
© 2021, transcosmos, inc. or its affiliates. All rights reserved.

トランスコスモス

1966年創業

東証1部上場・売上高3,119億円・世界30ヵ国/169拠点

- 従業員数　6.2万人 (国内4.3万人 海外1.9万人)
- 対応言語/国　23言語 (コンタクトセンター) / 48ヶ国 (ECワンストップ)
- 取引先企業　3,000社



お客様企業の変革を支援する
デジタル技術を活用したサービスを提供

Global Digital Transformation Partner

お客様企業のデジタルトランスフォーメーション
を支援する『戦略的パートナー』を目指す

© 2021, transcosmos, inc. or its affiliates. All rights reserved.

トランスコスモス

世界30カ国/169拠点

- 従業員数　6.2万人 (国内4.3万人 海外1.9万人)
- 対応言語/国　23言語 (コンタクトセンター) / 48ヶ国 (ECワンストップ)

1989年 北米	transcosmos America
1995年 中国	transcosmos China
2000年 韓国	transcosmos Korea
2004年 タイ	transcosmos Thailand
2013年 フィリピン	transcosmos Asia Philippines
2013年 インドネシア	transcosmos Indonesia
2013年 マレーシア	transcosmos Malaysia
2014年 ベトナム	transcosmos Vietnam
2014年 イギリス	transcosmos UK
2016年 台湾	transcosmos Taiwan

© 2021, transcosmos, inc. or its affiliates. All rights reserved.

外的環境

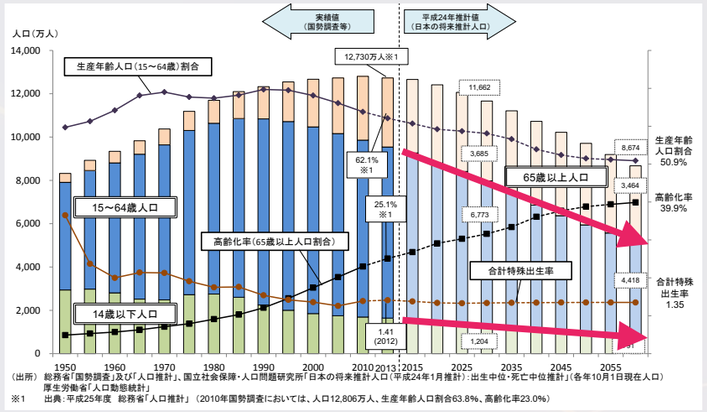
© 2021, transcosmos, inc. or its affiliates. All rights reserved.

外的環境

ベース **日本の人口減少**

生産年齢人口（15~64歳）が減り続けている

日本の人口は毎年100万人減少
高齢化＝若年・生産年齢層は100万人以上減少



生産年齢人口が減少していく

国内のデジタル化推進・DX市場の拡大

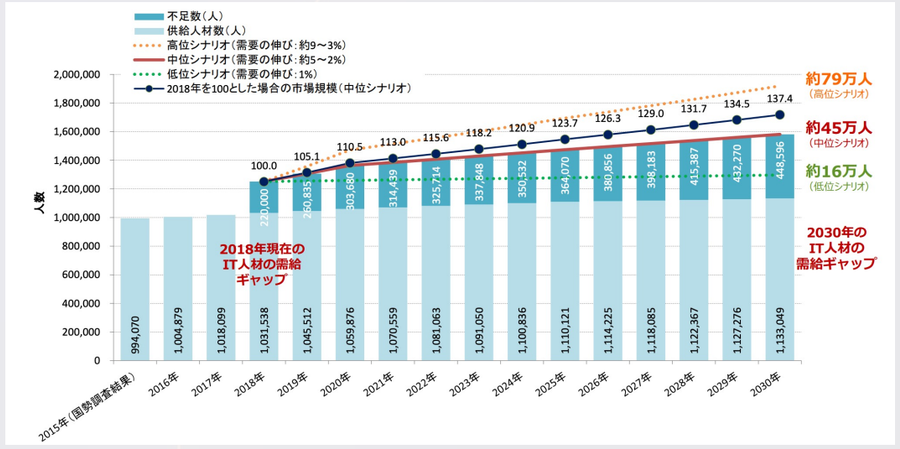
働き方改革（リモートワーク）

情報セキュリティの担保

コロナ禍でさらに加速

日本国内のIT人材不足

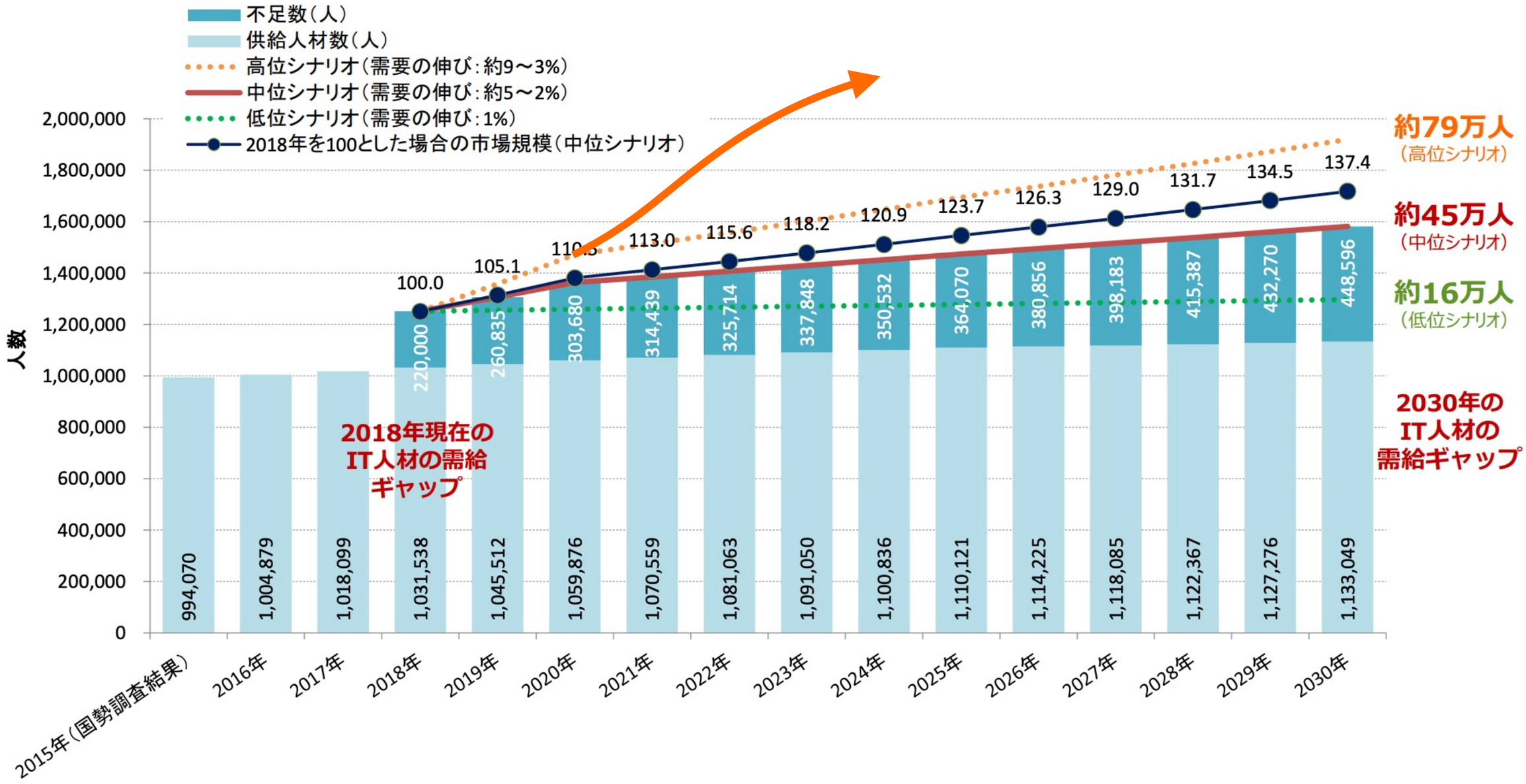
2030年には約60万人のIT人材が不足する by 経済産業省



AWS

© 2021, transcosmos, inc. or its affiliates. All rights reserved.

外的環境



© 2021, transcosmos, inc. or its affiliates. All rights reserved.

内的環境と機会

© 2021, transcosmos, inc. or its affiliates. All rights reserved.

内的環境と機会

ベース

クライアント毎にテーラーメイドで
寄り添って成長してきた歴史のある会社

→ プロジェクト毎に個別最適化されている

クラウドサーバー利用を原則禁止していた
需要に押されて利用開始

→ 更に個別最適化が進んだ

- システムがサイロ化していて汎用性に欠ける
- プロジェクトや人に依存しがち
- セキュリティ対策などの共通項目の全体最適化が難しい
- 日本利用・日本語対応中心のシステム
- 社内IT人材不足

DX（Digital Transformation）**は**
単純なデジタル化・個別最適化では
上手く行かない

サーバーレスという新しい仕組みが出てきた

アジア圏に100%資本の開発会社が複数ある

→ **変わる機会が到来**

© 2021, transcosmos, inc. or its affiliates. All rights reserved.

トランスコスモスグループ 技術中期計画 一部（2020~2022年）

統一基準のシステムで解決

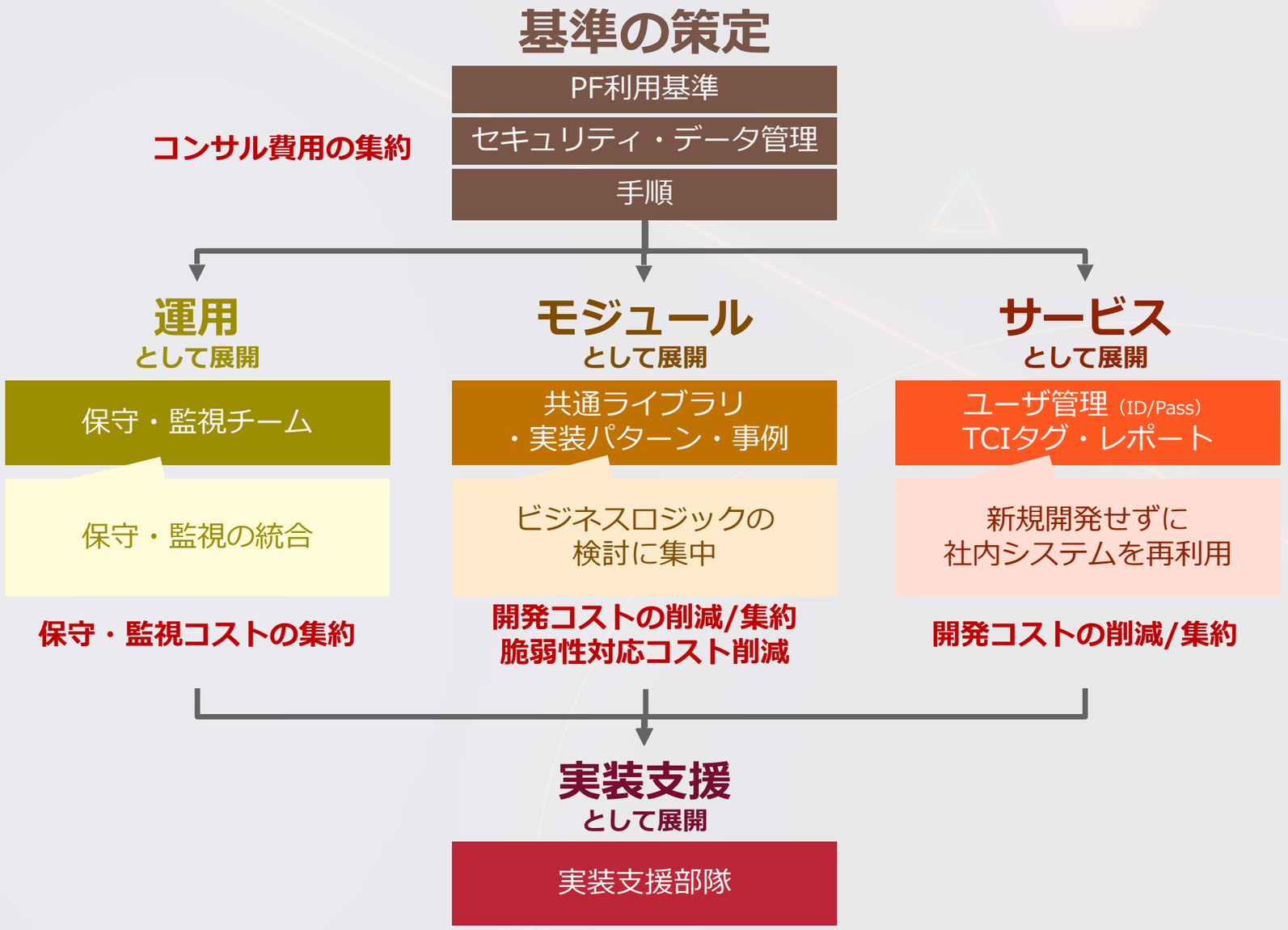
システムのセキュリティや
監視・運用など基準統一をはかる

共通利用できるものを本社で準備し
各部の対応費用のコスト削減へ貢献

グループ力で解決

日本だけではなくグループで
構築・管理・運用する

内部にクラウドネイティブの
DevOps部隊を作って
自身の技術力を高める



© 2021, transcosmos, inc. or its affiliates. All rights reserved.

ガイドライン全体感

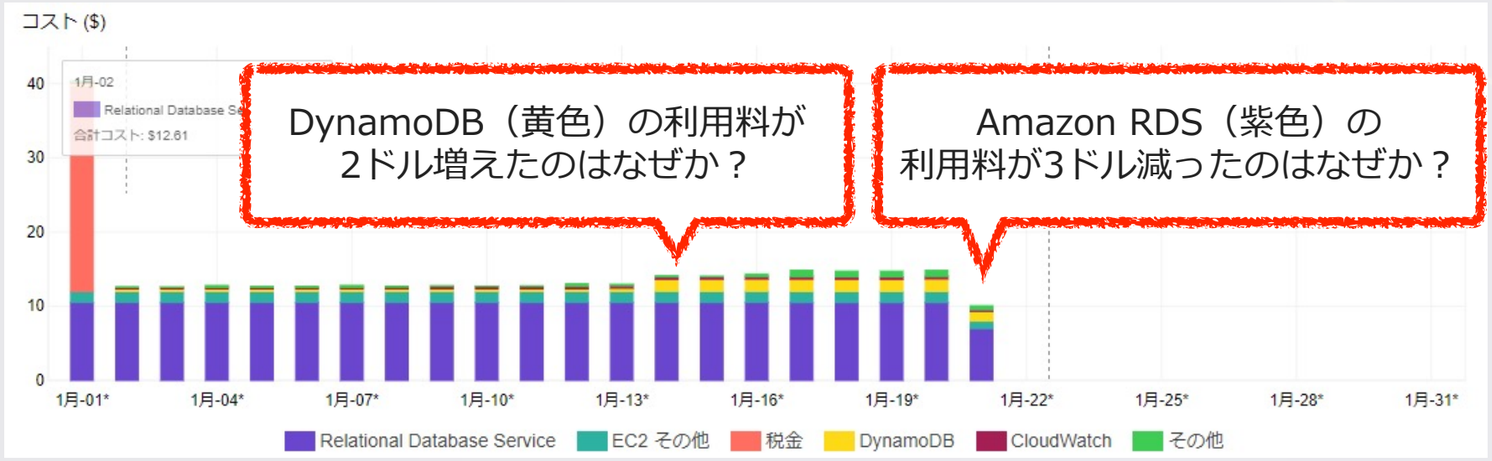
総ページ数		個別ページ	言語	現在
ガイドライン AWSを如何に 安全に効率良く 利用するか	セキュリティ	105	日・英	AWSのサーバーレス 実例Tips50項目
	アカウント管理	9	日・英	
229	計画	20	日・英	
	運用	12	日・英	
	開発	71	日・英	
開発時のルール 開発時の共通認識	コーディング規約	25	日・英	ベースとして 集約・記載
	UI規定	1	日・英	
	ファイルの管理方法	4	日・英	
60	データベース	16	日・英	
	エラーハンドリング	6	日・英	
	ログ管理	8	日・英	
開発支援ライブラリ 誰がやっても同じ結果に ならないといけないもの	インプットデータ検証	38	日・英	JavaScript React.js/Node.js で準備
	ワーディング管理	16	日・英	
71	簡易問合せ管理	17	日・英	
	プロジェクトテンプレート		日・英	
サービス	TCI IAM（ユーザアクセス管理）	24	日・英	稼働予定
	TCI ATM（システムアクセス管理）	19	日・英	
67	Wiki	24	日・英	
既存サービス 利用時の利用方法	ログモニタリング	71	日・英	ログ監視・BIツール
71	BIダッシュボード			

© 2021, transcosmos, inc. or its affiliates. All rights reserved.

理由

© 2021, transcosmos, inc. or its affiliates. All rights reserved.

開発現場でおきていること



DynamoDB（黄色）の利用料が
2ドル増えたのはなぜか？

Amazon RDS（紫色）の
利用料が3ドル減ったのはなぜか？

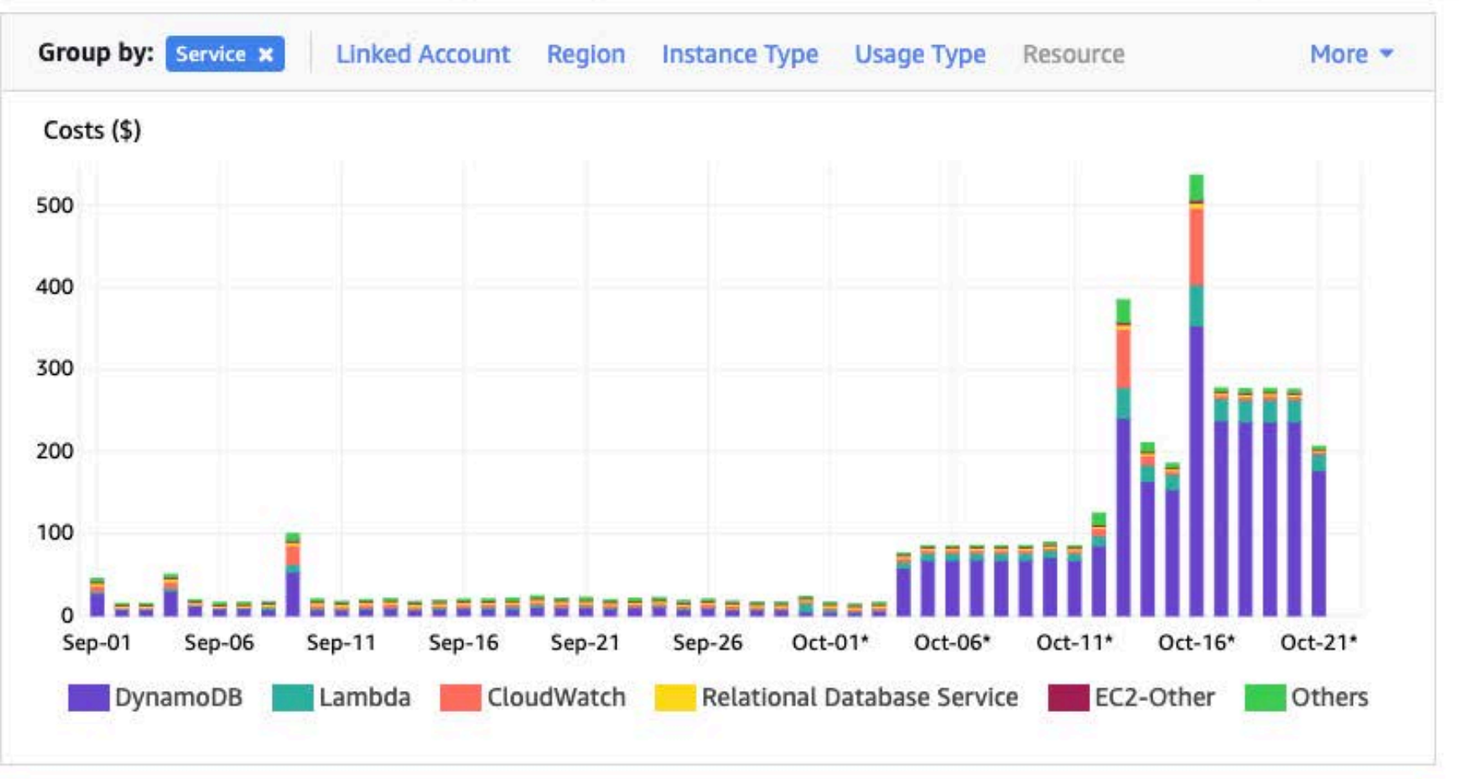
利用状況を日々確認するコスト管理は重要

表のページネーションの表示について
以下の点に関して、追記して下さい。
①ページ遷移した際の現在値はオレンジ色で表示されるのか
②現在値から前後いくつまでのページ数を表示するのか
③現在値によって前後のページ数の表示を切り替えるのであれば、その閾値を記載して下さい

仕様とそれにそった動作確認は重要

© 2021, transcosmos, inc. or its affiliates. All rights reserved.

利用状況を確認を怠った結果



サーバーレスは使った分だけ課金される
ロジックミスが発生して気がつかなかったら大量の課金が発生する

利用状況を日々確認する
コスト管理は重要

実際にこういう事が発生した

© 2021, transcosmos, inc. or its affiliates. All rights reserved.

他にも

セキュリティ
システムの監視・運用方法
ログイン周りの処理
大量処理
異常系発生時の対応
：
等々

全部重要なこと

ただ確認はするけど
どうあるべきかの基準や手順書は無い

ちゃんと出来ているべき・こうあるべきだ
でも、どうやるかは知らない・それは任せる

© 2021, transcosmos, inc. or its affiliates. All rights reserved.

よく言われること

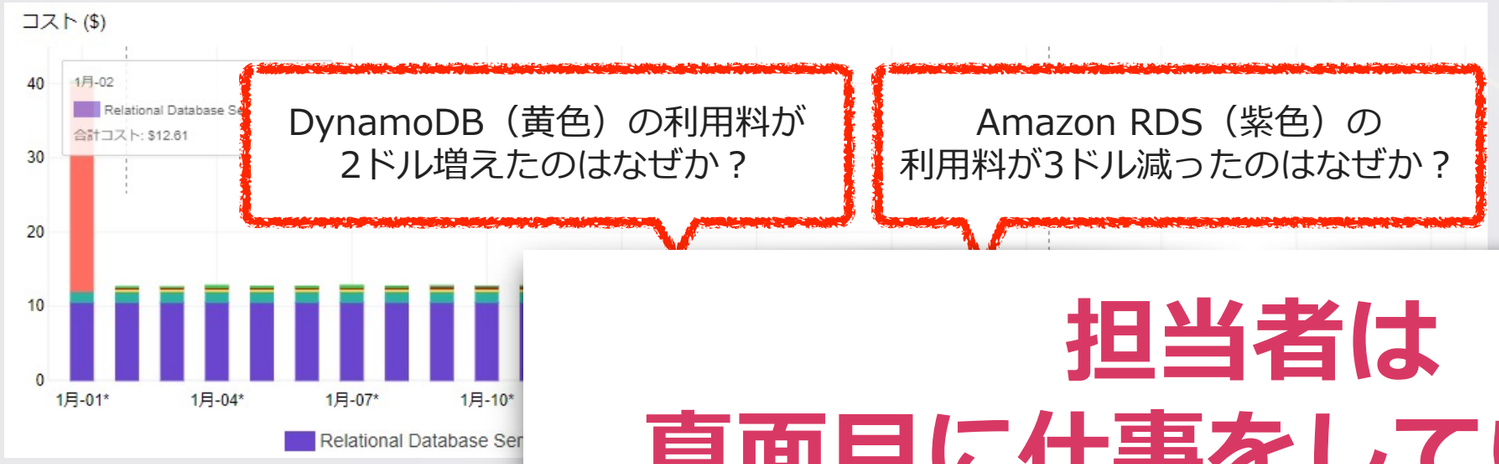
毎日 各自が5分使って確認・作業するだけで
人数が集まれば大きな効果を生み出す

確かにそうです!!

毎日 各自が効率化をして残業を少なくすれば
コストが下がる

© 2021, transcosmos, inc. or its affiliates. All rights reserved.

開発現場では



DynamoDB（黄色）の利用料が
2ドル増えたのはなぜか？

Amazon RDS（紫色）の
利用料が3ドル減ったのはなぜか？

担当者は
真面目に仕事をしているだけ
自分に与えられた役割の中で
評価されるために「改善活動」をしている

①ページ遷移した際の**現在値はオレンジ色**で表示されるのか
②現在値から**前後いくつまでのページ数を表示**するのか
③現在値によって前後のページ数の**表示を切り替えるのであれば、その閾値を記載**して下さい

利用状況を日々確認するコスト管理は重要
でも、1日数ドルの差分のための確認作業が走る
稼働・スキルに依存している

仕様とそれにそった動作確認は重要
でも、生産性のない汎用的な機能に対して
各プロジェクト担当者の時間を費やしている

© 2021, transcosmos, inc. or its affiliates. All rights reserved.

グローバルに勝つためには

でも、1日数ドルの差分のための確認作業が走る
確認作業も各PJ担当者の稼働・スキルに依存している

でも、生産性のない汎用的な機能に対して
各プロジェクト担当者の時間を費やしている

管理・組織の問題

生産性の低い作業を個人のスキルや稼働に頼ってはグローバルでは勝てない

**個人に依存しない
ルール化・仕組み化をしないといけない**

© 2021, transcosmos, inc. or its affiliates. All rights reserved.

良識・常識のアセット化

頭の中

日本開発者の
良識・常識

**海外の開発者は
全く判らない**

日本国内の各開発部門・チーム・プロジェクトでも
基準が統一されていない状況

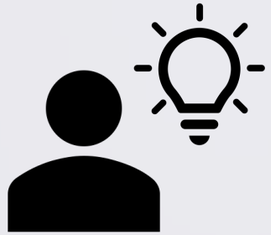
「開発は普通こういうところまでやるよね」
という日本人開発者の常識・良識をベースに実施

日本人開発者の良識・常識を海外でも活用できるようにアセット化

ガイドライン

明文化・基準化

ライブラリ化



**海外の開発者も
判るようになる**

明文化・基準化することで判る事が出来る

誰がやっても同じものはライブラリ化すれば
作業効率が上がる・ミスがなくなる

© 2021, transcosmos, inc. or its affiliates. All rights reserved.

良識・常識のアセット化の重要性

採用しづらい状況の中
「どうすれば使い安いシステムになるのか？」
「どうすれば売上に繋がる機能になるのか？」
「早く処理するロジックはないか？」

生産的な事に
人材の頭・時間を使うべき

- **プロジェクト単位でROIを見ていると**
  「喉元過ぎれば‥」で終わる
- **人手も足りないので自分たちのプロジェクトで精一杯**
- **開発チーム内で基準を作っても**
  組織・チーム変更で散り散りになる
- **開発基準の標準化プロジェクト単体ではコストになる**
- **そもそも全体で決めるべき内容で**
  チームや部門だけではROIが出しづらい

会社として決める必要がある

© 2021, transcosmos, inc. or its affiliates. All rights reserved.

やるなら徹底的に!!

© 2021, transcosmos, inc. or its affiliates. All rights reserved.

サーバーレスを中核に

© 2021, transcosmos, inc. or its affiliates. All rights reserved.

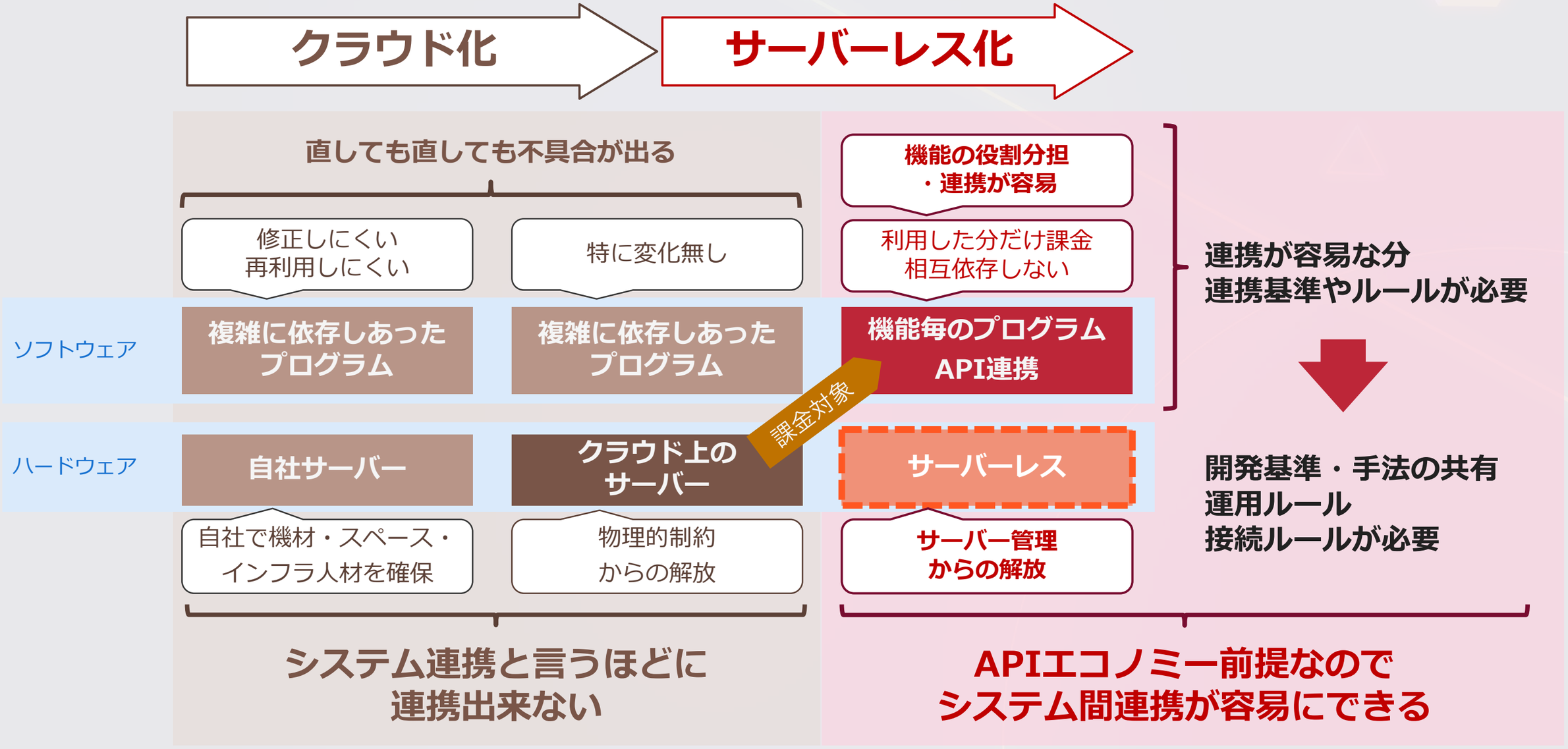
サーバーレスを中核に

新しい開発・実行環境であるサーバーレス型の開発を中心にして
既存システムのセキュリティ設定まで第一ステップとして実施

	新規		既存	
	クラウド			スマホアプリ
	サーバーレス	サーバーレス・サーバー混合	サーバー	
セキュリティ設定	STEP1	STEP1	STEP1	STEP2 ?
データの持ち方	STEP1	STEP1		
構成指針	STEP1	STEP2		
開発手法	STEP1	STEP2		
フレームワーク	STEP1	STEP2		
運用監視・保守	STEP1	STEP2		

© 2021, transcosmos, inc. or its affiliates. All rights reserved.

なぜサーバレスか？

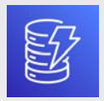


© 2021, transcosmos, inc. or its affiliates. All rights reserved.

サーバーレスの良いところ

Amazon
API Gateway

AWS Lambda



Amazon
DynamoDB

AWS Lambda

API

関数

データ テーブル

関数

独立

API

関数

データ テーブル

© 2021, transcosmos, inc. or its affiliates. All rights reserved.

サーバーレス開発のチーム（敢えて別けるなら）

UI/UX	Vue.js / React.js	**UX/CX・ビジネスロジックの核** UI要素は共通化出来る	**UIチーム・QCチーム** 一番力を入れるポイント
Interface	Cloud Front / API Gateway / AppSync	**決まった設定で解決できる** 設定は共有出来る	**インフラチーム** AWSをベースとして マルチプラットフォーム化
Computing	Lambda / Fargate	**機能毎に準備** プロジェクト間で 共有出来る	**開発チーム** Lambda/Fargate対応出来れば 開発言語やフレームワークは何でも良い
DB	DynamoDB / RDS		
Storage	Amazon S3	**決まった設定で解決できる** 設定は共有出来る	**インフラチーム** AWSをベースとして マルチプラットフォーム化
Basement	NW / IAM Account		

全体の最適化

© 2021, transcosmos, inc. or its affiliates. All rights reserved.

サーバーレス開発のチーム（敢えて別けるなら）

UI/UX	Vue.js / React.js	UX/CX・ ビジネスロジックの核 UI要素は共通化出来る	**UIチーム・QCチーム** 一番力を入れるポイント
Computing	Lambda / Fargate	機能毎に準備 プロジェクト間で 共有出来る	**開発チーム** Lambda/Fargate対応出来れば 開発言語やフレームワークは何でも良い
DB	DynamoDB / RDS		
Interface	Cloud Front / API Gateway / AppSync	決まった設定で 解決できる 設定は共有出来る	**インフラチーム** AWSをベースとして マルチプラットフォーム化
Storage	Amazon S3		
Basement	NW / IAM Account		

全体の最適化

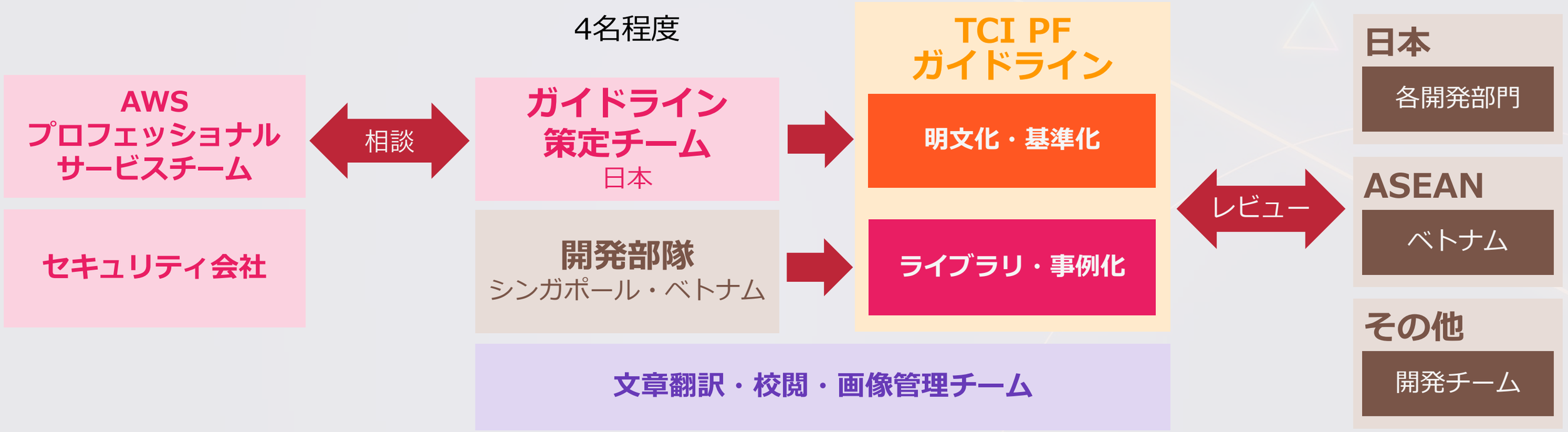
© 2021, transcosmos, inc. or its affiliates. All rights reserved.

プロジェクト体制

ガイドラインのまとめ方

© 2021, transcosmos, inc. or its affiliates. All rights reserved.

プロジェクト体制



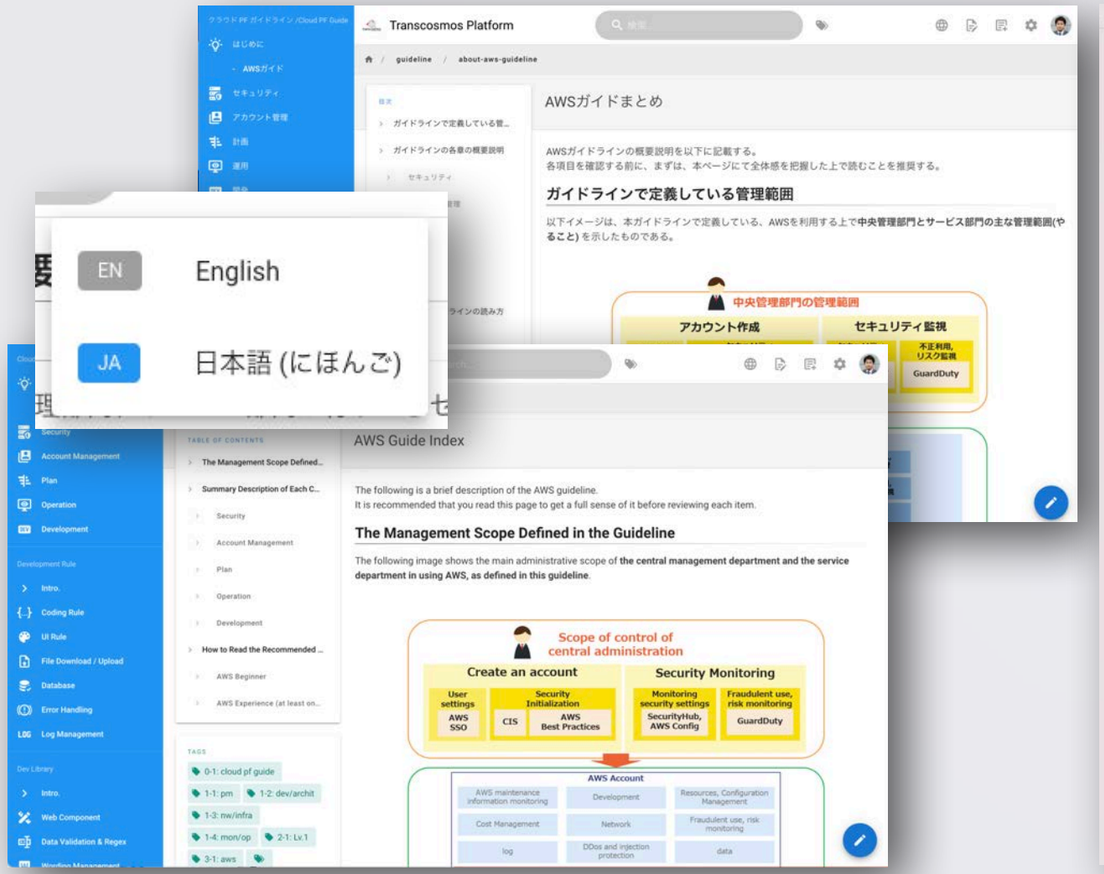
© 2021, transcosmos, inc. or its affiliates. All rights reserved.

Wiki形式で記述

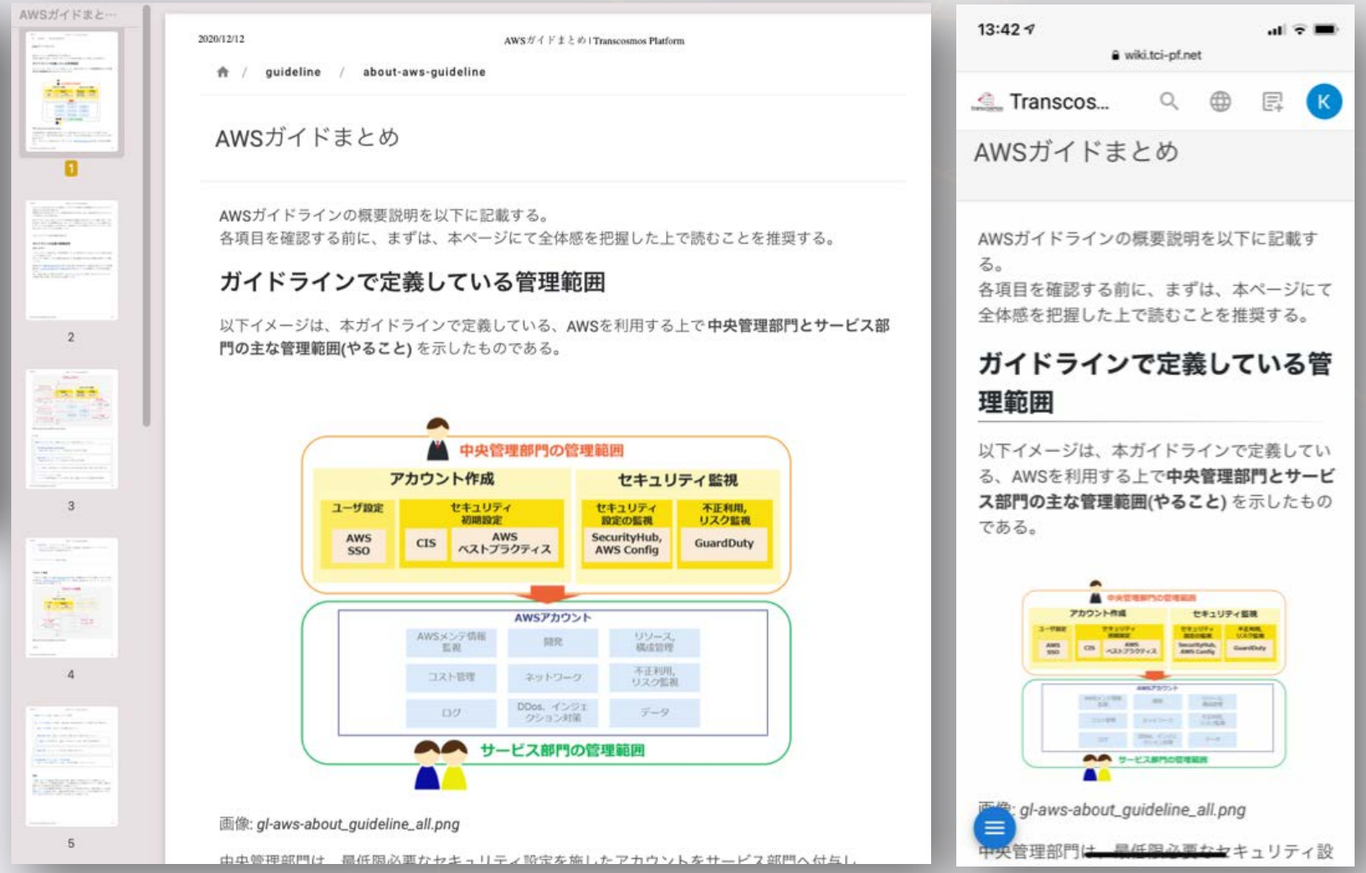
マルチ言語対応
印刷対応・モバイル端末表示対応

グループ内SSOを介して
国内・海外各社のメンバーも管理されたアクセスが出来るようになる

Webサイト



印刷・PDF



モバイル

© 2021, transcosmos, inc. or its affiliates. All rights reserved.

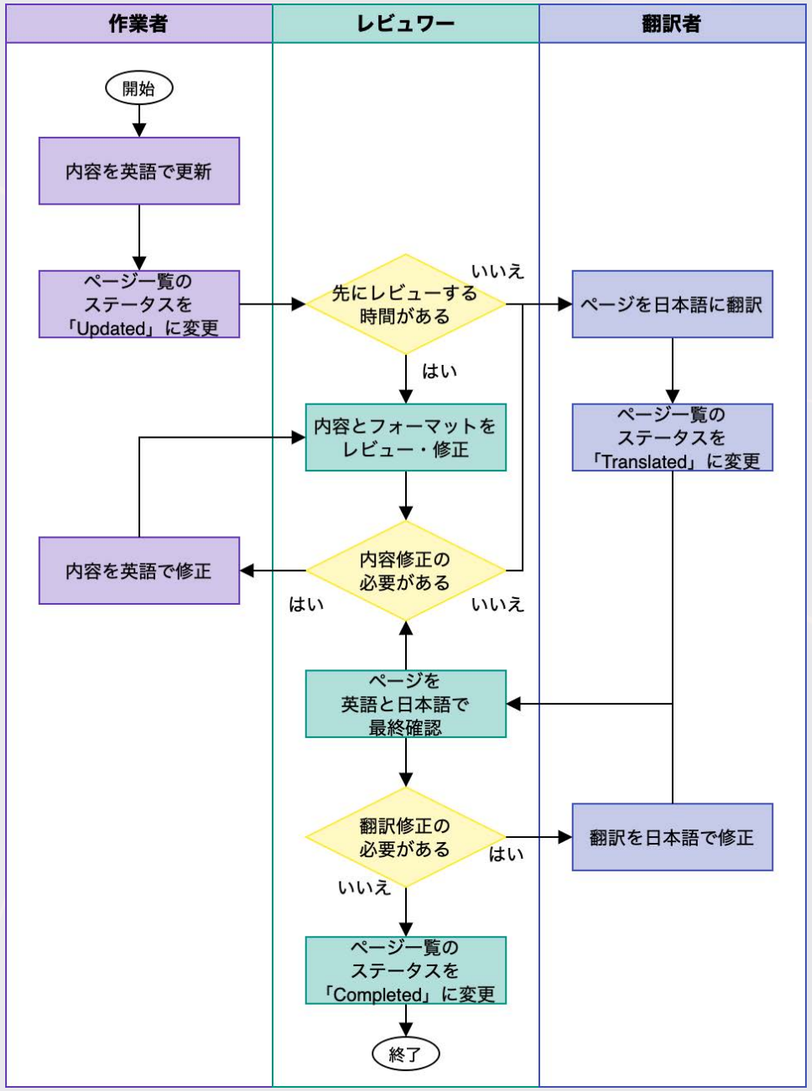
Wikiの編集ルール策定

綺麗に見せるためには
パワーポイントやワードとは違ったお作法が必要

項目

Wiki大枠の決定
Wikiチームの結成
Wikiツールの準備
Wikiアクセスの設定
ページ一覧の作成
担当者のアサイン
資料共有場所の決定
記載対象へのアクセス設定
記載言語の決定
ワークフローの作成
命名規則の決定
テンプレートの作成
細かい表現形式の決定
注意点の共有

編集ルールもWikiに
日本語・英語で記載して対応



日本語と英語でシステムマニュアルを作る場合の作業実施例

© 2021, transcosmos, inc. or its affiliates. All rights reserved.

ガイドライン全体感

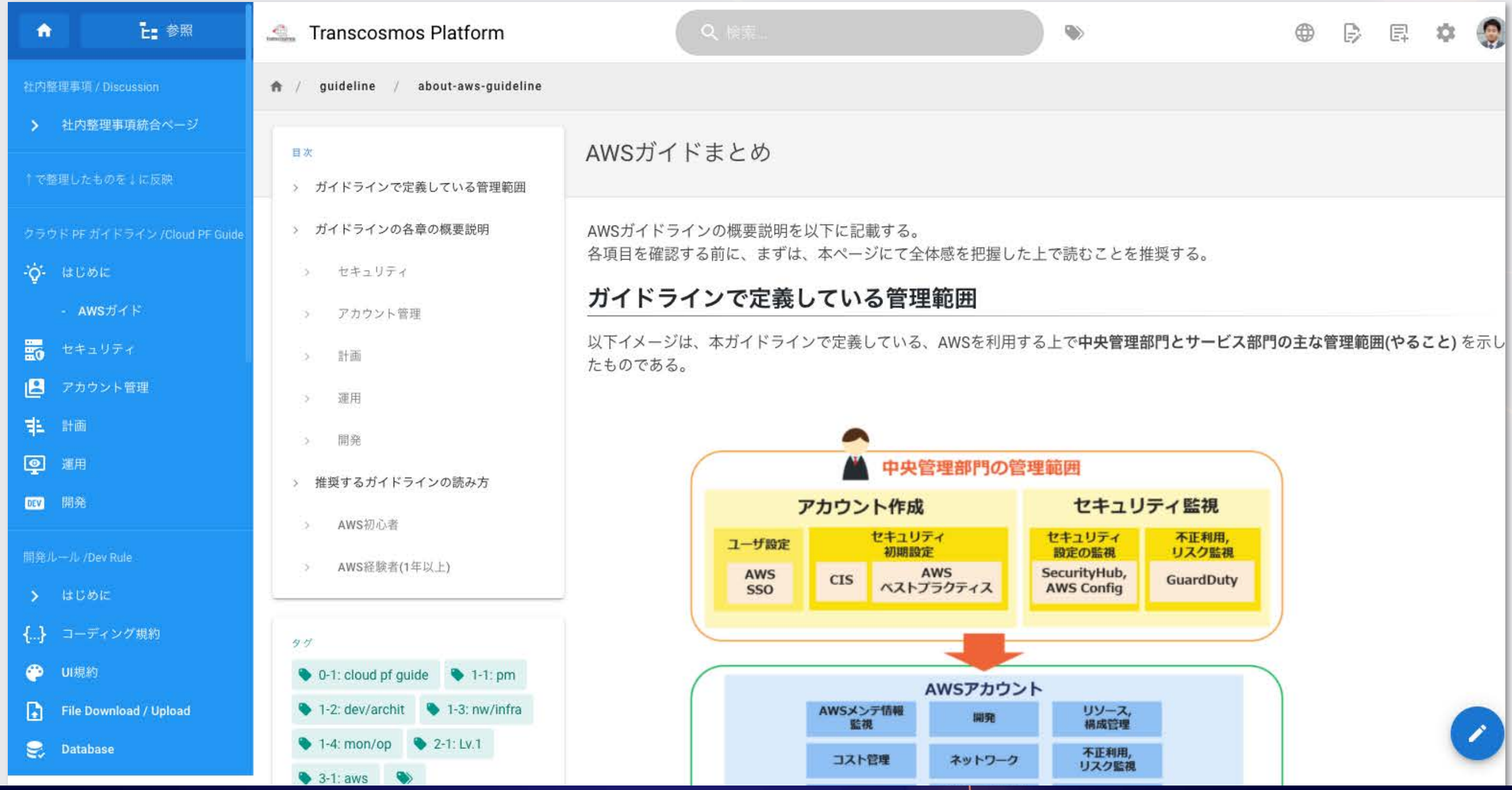
© 2021, transcosmos, inc. or its affiliates. All rights reserved.

ガイドライン全体感

	総ページ数		個別ページ	言語	現在	今後
ガイドライン AWSを如何に 安全に効率良く 利用するか	**229**	セキュリティ	105	日・英	AWSのサーバーレス 実例Tips50項目	AWSその他サービスも
		アカウント管理	9	日・英		
		計画	20	日・英		
		運用	12	日・英		
		開発	71	日・英		
開発時のルール 開発時の共通認識	**60**	コーディング規約	25	日・英	ベースとして 集約・記載	内容をもっと拡充 バグ発生時の判断基準 開発者の評価基準
		UI規定	1	日・英		
		ファイルの管理方法	4	日・英		
		データベース	16	日・英		
		エラーハンドリング	6	日・英		
		ログ管理	8	日・英		
開発支援ライブラリ 誰がやっても同じ結果に ならないといけないもの	**71**	インプットデータ検証	38	日・英	JavaScript React.js/Node.js で準備	ライブラリ種別追加 Java/PHPなども検討
		ワーディング管理	16	日・英		
		簡易問合せ管理	17	日・英		
		プロジェクトテンプレート		日・英		
サービス	**67**	TCI IAM（ユーザアクセス管理）	24	日・英	稼働予定	利用ケースの拡大
		TCI ATM（システムアクセス管理）	19	日・英		
		Wiki	24	日・英		
既存サービス 利用時の利用方法	**71**	ログモニタリング	71	日・英	ログ監視・BIツール	利用ケースの拡大
		BIダッシュボード				

© 2021, transcosmos, inc. or its affiliates. All rights reserved.

PFガイドライン - AWSガイドまとめ



© 2021, transcosmos, inc. or its affiliates. All rights reserved.

PFガイドライン - セキュリティについて

© 2021, transcosmos, inc. or its affiliates. All rights reserved.

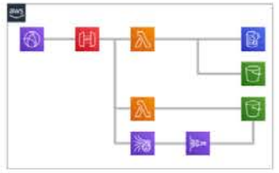
PFガイドライン - セキュアコーディング

© 2021, transcosmos, inc. or its affiliates. All rights reserved.

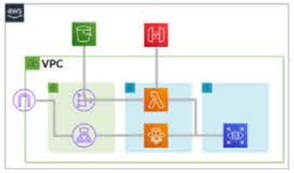
PFガイドライン - ネットワーク構成方針

構成例

非VPC構成　　VPC構成



画像: gl-aws-nw_nonevpc_example.png



画像: gl-aws-nw_vpc_example.png

VPC構成の判断基準

条件によっては**VPC構成**を必要とする場合があるため、判断基準や判断例を記載する。

判断基準　判断ロジック　判断例

- **VPC必須サービスによる判断**
  AWSサービスの中にはVPC必須のサービスがあるため、それらを利用する場合は**VPC構成**が必須となる
- **データ重要度による判断**
  データの重要度に応じて、VPC構成を必須とするかを定義する(※)
- **要件(NWの分割)による判断**
  PCI DSSへの対応など、NW分割に関する要件がある場合、サービス部門の判断で**VPC構成**を採用し、要件に対応する

※データ重要度に応じたVPC要否対応表

機密レベル	VPC要否
4：厳秘	**VPC必須**
3：機密	非VPC可
2：社外秘	非VPC可
1：一般	非VPC可

厳秘情報は、データ保護に対する対策を可能な限り実施する必要があるため、ネットワーク分離(VPC構成)による保護を行う

VPC必須のサービス　VPC上に展開する必要のあるサービス一覧

VPC構成における推奨構成

VPC構成を選択する場合、どのようにサブネットを分割するかを検討する必要がある。
本ガイドラインでは、データの重要度に応じた推奨構成を記載する。

サービス部門がVPC構成における推奨構成を低コストで実現できるよう、アカウント作成時に中央管理部門でビルトインVPCを設定し、提供する。
アカウント作成フローに記載したとおり、4：「厳秘」を扱う場合は汎用と厳秘の2種類のアカウントを払い出し、ビルトインVPCもそれぞれのアカウントに作成する。

ビルトイン
アカウント払い出しの際に、サービス部門が自由に使えるよう中央部門が準備するリソース

方針

関連項目: `TPG-AWS-Best-EC2.6`

- **ビルトインVPCの利用と削除**
  ビルトインVPCの利用は任意であり、非VPC構成をとる場合や、サービス部門にて独自のVPCを作成する場合は削除してもよい。
- **ビルトインVPCの対象リージョン**
  ビルトインVPCは、東京リージョン(ap-northeast-1)に作成する。
- **ビルトインVPC利用時の検討項目**
  ビルトインVPCを利用する場合、汎用アカウントのビルトインVPC/厳秘アカウントのビルトインVPCの利用方法から、**サービス部門が行う検討、設定項目**を確認すること。

- **独自VPCの利用**
  ビルトインVPCが要件に合わない場合や、東京リージョン以外でAmazon VPC を利用する場合は、サービス部門で独自のVPCを作成してもよい。
- **独自VPCの利用時のセキュリティ**
  独自VPCを作成する場合、ビルトインVPCの考え方を参考に、セキュリティや冗長化等を考慮すること。
  特に、**VPCフローログの有効化は必ず実施**すること。

汎用アカウントのビルトインVPC

「3：機密」「2：社外秘」「1：一般」構成に使う、汎用アカウントのビルトインVPCの初期設定と利用方法を記載する。

初期設定　利用方法「3：機密」　利用方法 2：「社外秘」、1：「一般公開」

汎用アカウント ビルトインVPCの初期設定は以下の通り。

構成

画像: gl-aws-nw_builtinvpc_generic.png

設定内容

#	アカウント	項目	内容
(1)	汎用	サブネット	(汎用)DMZ、(汎用)Trust、(汎用)DBの3種類のサブネットを作成 冗長化を想定して3つのAZに同じ種類のサブネットを作成する[詳細] サブネットのIPアドレスはこちらを参照

© 2021, transcosmos, inc. or its affiliates. All rights reserved.

PFガイドライン - データバックアップ方針

推奨されるバックアップ手法と設定

データベース (Amazon RDS , Amazon DynamoDB)

サーバレス開発においては、まず「Amazon RDS」「Amazon DynamoDB」などのデータベースのバックアップを計画する必要がある。
データベースのバックアップには主に以下２つの手法が存在する。

- **自動バックアップ(推奨)**
- **AWS Backup**

バックアップ手法の比較

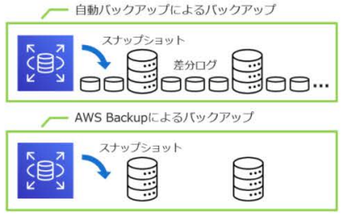
関連項目: IPA-AWS-A.1.3.1 IPA-AWS-C.1.2.2 ～ IPA-AWS-1.2.5

それぞれのバックアップ手法における特徴は以下となる。

-	自動バックアップ	AWS Backup
ポイントインタイムリカバリ(※1)の利用	可	不可
バックアップスケジュール設定	可	可
複数サービスのバックアップ統合管理	不可	Amazon EC2 , Amazon RDS など、複数サービスのバックアップの統合管理が可能
最大保持期間	35日	無期限
マルチリージョンでのバックアップ	不可	可
低価格ストレージへの利用	不可	可

※1：ポイントインタイムリカバリ：バックアップ保存期間中の任意の時点にテーブルを復元できる機能。詳細は下記。
Amazon RDSのポイントインタイムリカバリ
Amazon DynamoDBのポイントインタイムリカバリ

- 自動バックアップとAWS Backupのバックアップイメージの比較



画像: *gl-aws-backup_rds_backuptype_ja.png*

推奨バックアップ手法

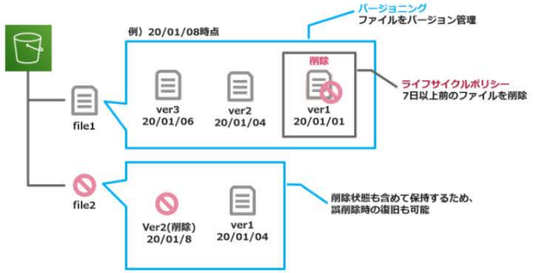
関連項目: IPA-AWS-A.1.3.1 IPA-AWS-C.1.2.2 ～ IPA-AWS-1.2.5

「自動バックアップ」の利用を推奨とする。

これはIPAの「非機能要件グレード2018」の重要項目「A.1.3.1.目標復旧水準(RPO(目標復旧地点))」にて
障害からの復旧時、データは障害発生時点からなるべく近い時間に復旧させる事が要求されており、

バージョニング	バージョニングの有効化	ON
ライフサイクルポリシー	ルールスコープを選択	バケット内のすべてのオブジェクトに適用
ライフサイクルポリシー	ストレージクラスの移行	OFF
ライフサイクルポリシー	失効の設定	以前のバージョン ※現行バージョンはOFF
ライフサイクルポリシー	前のバージョンを完全に削除する	7日間
ライフサイクルポリシー	期限切れのオブジェクト削除マーカーをクリーンアップする	ON
ライフサイクルポリシー	不完全なマルチパートアップロードをクリーンアップする	7日間

- 動作イメージ



画像: *gl-aws-backup_s3_versioning_and_lifecyclepolicy_ja.png*

「前のバージョンを完全に削除する」≒保持期間である。
本設定項目はデータが誤って削除された事に気づきづらい様なシステムの場合(※1)はより長期間を指定する事とする。
※1：利用頻度の低い機能で該当データを利用している場合等

その他

- マルチリージョンでの可用性が要求される場合はAmazon S3 の「レプリケーション」機能を用いる事とする。この機能ではS3バケット単位で別リージョンに対して複製を行う事が出来る。
- バックアップの観点とは異なるが、ほぼ更新が入らない様なファイルであればAmazon S3 の「オブジェクトロック」機能を用いて
ファイルの更新/削除などを完全に禁ずる事も可能であるため、誤操作を抑制したい場合はこちらの機能も検討する。

© 2021, transcosmos, inc. or its affiliates. All rights reserved.

PFガイドライン - リファレンスアーキテクチャ

- リクエストの内容により細かい処理分岐を行う必要がある場合
- リクエスト1つ1つの整合性よりも、多数のリクエストを受けても安定したスループットを維持することを優先する場合
- 1つのリクエストから前後関係のない複数の独立したワークフローを開始させる場合（注文処理・キャンペーン適用処理・リアルタイム集計等）

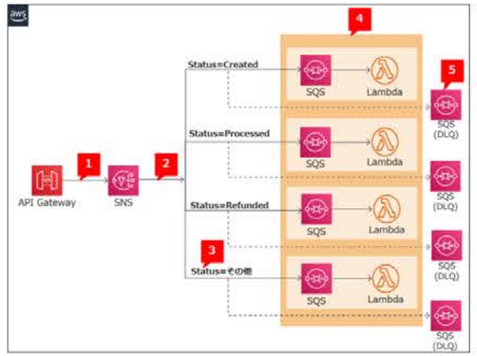
アーキテクチャ　設計思想　利用サービス

概要

- REST APIのリクエストメソッドではなく、リクエストの内容によって処理が分岐する場合は Amazon SNS によるファンアウト構成をとることができる。SNSメッセージのフィルタリング機能を利用し、該当する内容のリクエストを適切なSQSキューに入れて非同期で順次処理する。分岐と並列制御をAWSサービスの機能にオフロードできるため、コード量を削減できる。
- 分岐は必ず1:1でなければならないわけではなく、1つのリクエストから一つまたは複数の処理を柔軟に実施させることができ、完全な疎結合の構成でインタフェースとバックエンドの開発の依存性を無くすことができる。

ファンアウトとは

Amazon SNS で受信したメッセージを複製し、同一のメッセージを複数のSQSキューに並列的に送信する仕組み



画像: *gl-aws-architect_restapi_fanout.png*

ポイント	説明
1	Amazon API Gateway で受けたリクエストは、直接 Amazon SNS にそのまま引き渡す
2	Amazon SNS のメッセージフィルタリングで各サブスクライバーの Amazon SQS がどのリクエストを受け取るか分岐させる
3	他のサブスクリプションフィルタの条件に合わなかった「その他」のメッセージを入れるためのキューを設定する 設定したすべてのサブスクリプションフィルタ条件と合わなかったメッセージのみを受け取るためには、フィルタ条件の **anything-but** を活用する。除外リスト方式で、指定した条件以外のものを指定することが可能。 ・フィルタ条件の設定方法
4	それぞれの処理に対するバックエンド関数を用意し、非同期で処理する。Amazon SNS は低レイテンシでメッセージの送信が可能なため、アプリケーションで条件分岐させるよりパフォーマンスメリットがある。
5	Amazon SNS のメッセージ配信失敗に備えたDead Letter Queue（DLQ）をサブスクライバー毎に設定する ・ Dead Letter Queueの設定方法

1	Amazon EventBridge（旧Amazon CloudWatch Events）でスケジュールされた定期起動イベントを設定する
2	AWS Step Functions で、複数の関数で構成された処理を順番に実行する
3	ダウンロードした外部データは既定の形式に加工し、データベースやS3バケット等に保存した後、後日、集計・分析を行う

計算量の多い処理の振り分け

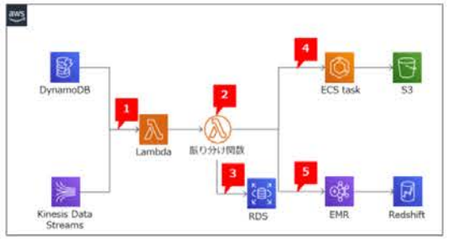
ユースケース

- Webサイトにアップロードされる画像に対して、そのまま保存/サムネイル作成/機械学習でタグ付け等、施す処理が多様な場合
- 計算量の少ない処理と多い処理の比率が流動的で予想が難しく、柔軟に対応する必要がある場合

アーキテクチャ　設計思想　利用サービス

概要

同じストリームに入ってくるデータの中に計算量（処理時間、必要なCPUリソースのレベル等）の大きく異なる処理が混在する場合、Lambda関数の処理時間の上限を回避しつつ、高負荷処理向けの他のコンピューティングサービスのインスタンス数も適切に抑える。
最初にLambda関数で振り分けを行い、すぐに処理できるものはLambda関数内で完結させ、計算量の多いものはコンテナや Amazon EMR 等で時間をかけて処理する。
※Amazon DynamoDB / Amazon Kinesis Data Streams 以外に Amazon SQS をデータ元とすることも可能



画像: *gl-aws-architect_datadispatch.png*

ポイント	説明
1	Amazon DynamoDB Streams で新規レコードの挿入を自動的にキャッチする、または Amazon Kinesis Data Streams を経由する等複数のデータソースからストリーミングデータを受ける
2	ストリームデータを取得し、必要な処理の種類によって異なるコンピューティングリソースに振り替える
3	短時間で処理できるものはLambda関数がそのまま Amazon RDS に書き込む
4	数分以上のバッチ処理が必要な場合はコンテナで処理して Amazon S3 に結果を保存
5	さらに時間を要する分析処理は Amazon EMR等、分析に特化したサービスを通じて Amazon Redshift に保存

リファレンスアーキテクチャ詳細

代表的なリファレンスアーキテクチャを以下に示す。
掲載しているアーキテクチャ図は、資料の構成上、利用サービスを一部、割愛している。

ログの集約と同時にリアルタイム処理

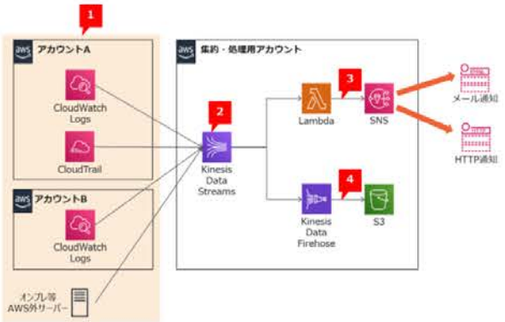
ユースケース

- ECサイトの大型キャンペーンで、詳細な分析とは別にリアルタイムで大口の購入を検知し速報値を表示したい
- サービスの操作証跡は全量残しつつ、不正な利用に対してリアルタイムで検知し自動で対処したい
- ログの出力元となるアプリケーションの設置場所が多岐に渡っている

アーキテクチャ　設計思想　利用サービス

概要

Amazon Kinesis Data Streams が持つデータ集約機能を利用し、アプリケーションログやAWSサービスの利用ログ等を複数のソースから一箇所に集約する。
後日、分析のためにS3バケットに全量を保存するのと同時に、リアルタイムで気づきたい問題や興味のある内容はピックアップして通知する。



画像: *gl-aws-architect_streamingdata.png*

ポイント	説明
1	ログは他AWSアカウントの Amazon CloudWatch Logs や AWS CloudTrail に加えて、外部のアプリケーションから Amazon Kinesis のライブラリを利用し送信することも可能。
2	Amazon Kinesis Data Streams に一旦すべてのソースからのデータを集約する。データの量によってスケールアウトも可能。
3	Lambda関数でリアルタイムで処理し、条件に合う内容を Amazon SNS 経由で通知する。
4	Amazon Kinesis Data Streams からは複数の宛先にデータを提供できるため、Amazon Kinesis Data Firehose 経由で並行して全てのログを Amazon S3 に保存する。保存したデータは後日、分析に活用することが可能。

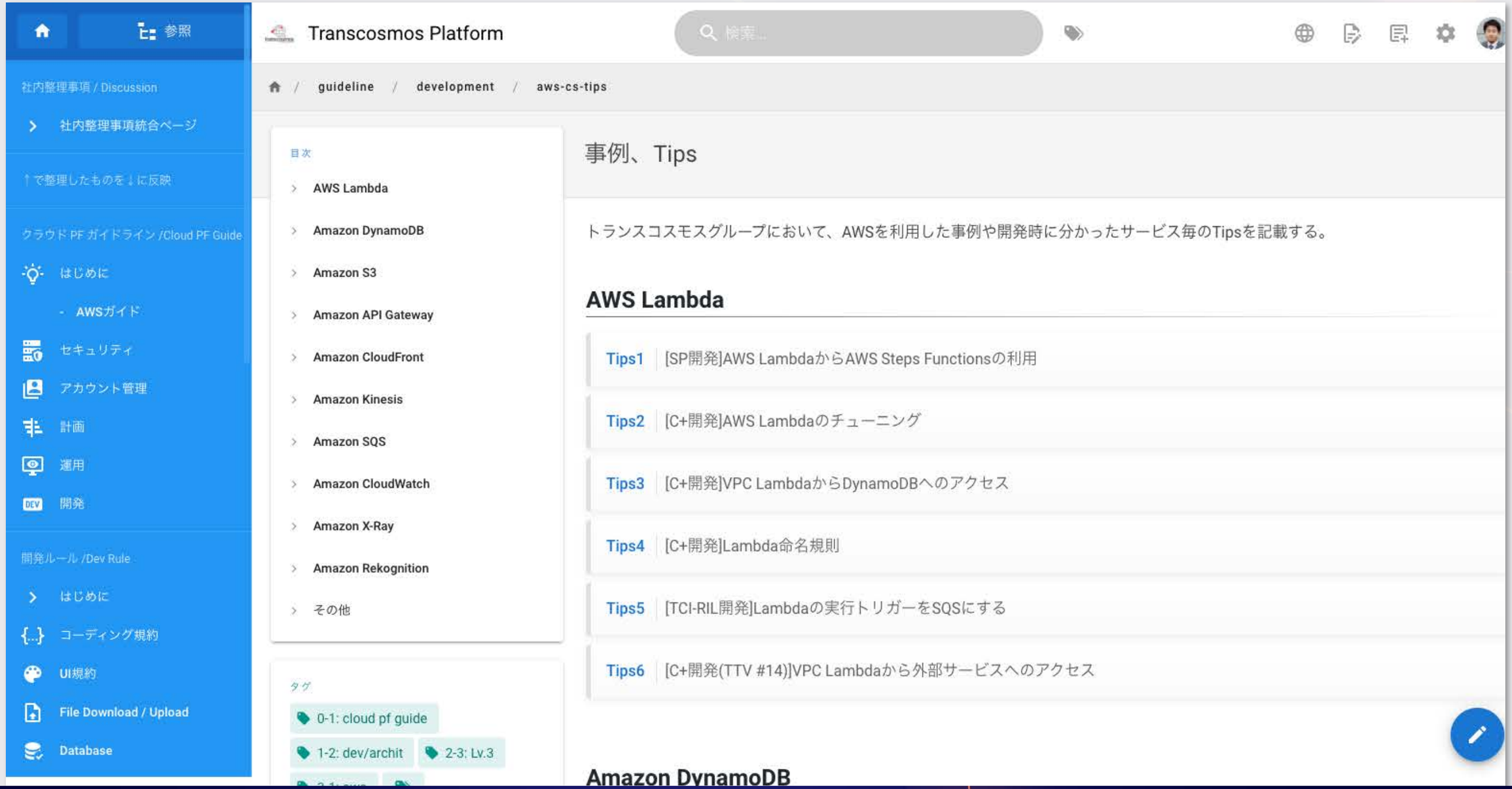
外部データの自動取り込みと集計

ユースケース

- サードパーティのデータプロバイダから提供されるデータ（経済データや天候データ等）を取得し、時系列で蓄積した上分析に用

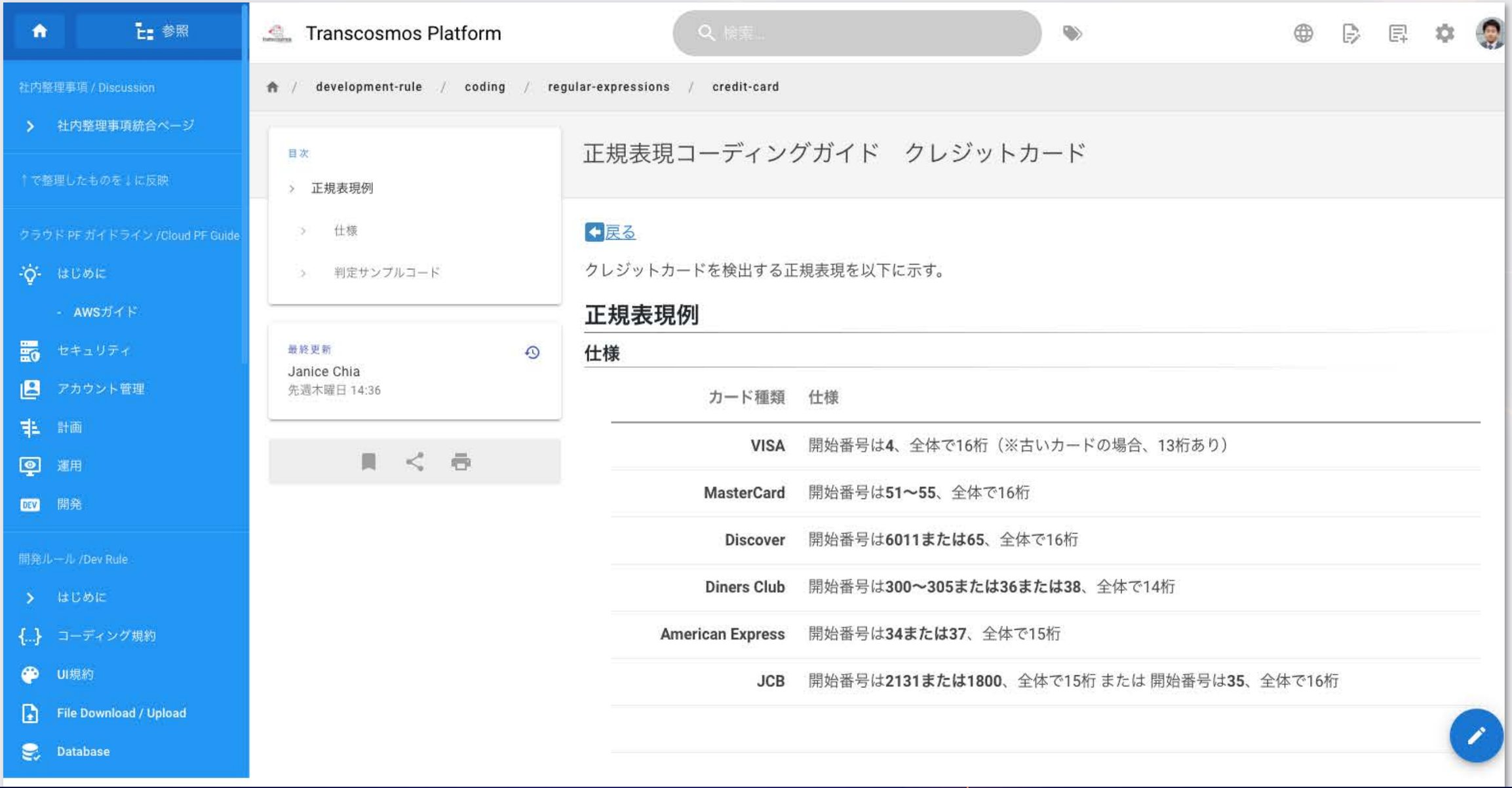
© 2021, transcosmos, inc. or its affiliates. All rights reserved.

PFガイドライン - 事例、Tips



© 2021, transcosmos, inc. or its affiliates. All rights reserved.

コーディング規約 - 正規表現



© 2021, transcosmos, inc. or its affiliates. All rights reserved.

コーディング規約 - 正規表現

画像: *regular-expressions_cre07_jp.svg*

各種正規表現サンプル

各種、正規表現の事例を以下に示す。

利用頻度の高い正規表現	半角英数字など、利用頻度の高いもの
文書検索	文章の中に指定したキーワードに関するもの
郵便番号、住所	郵便番号、住所に関するもの
電話番号	電話番号に関するもの
クレジットカード	クレジットカードに関するもの
ネットワーク関連	ネットワーク関連(IPアドレス/MACアドレス)に関するもの
メールアドレス	メールアドレスに関するもの
URL	URLに関するもの
日付、時間	日付、時間に関するもの

ード

```
= '4234567890123456';  // VISA
A = /^4[0-9]{12}(?:[0-9]{3})?$/;
ter = /^5[1-5][0-9]{14}$/;
cover = /^6011[0-9]{12}|65[0-9]{14}$/;
ers = /^3(?:0[0-5]|[68][0-9])[0-9]{11}$/;
ican = /^3[47][0-9]{13}$/;
= /^(?:2131|1800|35[0-9]{3})[0-
.exec(cardNo)) {
e.log('VISA');
egMaster.exec(cardNo)) {
.log('Master Card');
egDiscover.exec(cardNo)) {
.log('Discover Card');
egDiners.exec(cardNo)) {
.log('Diners Club');
egAmerican.exec(cardNo)) {
.log('American Express');
gJCB.exec(cardNo)) {
log('JCB Card');

log('no much.');
```

日本語入力

住所

電話番号

2021 transcosmos inc.. All rights reserved. | Powered by Wiki.js

© 2021, transcosmos, inc. or its affiliates. All rights reserved.

良識・常識のアセット化

頭の中

日本開発者の
良識・常識

**海外の開発者は
全く判らない**

日本国内の各開発部門・チーム・プロジェクトでも
基準が統一されていない状況

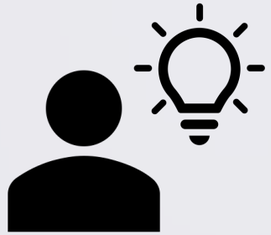
「開発は普通こういうところまでやるよね」
という日本人開発者の常識・良識をベースに実施

日本人開発者の良識・常識を海外でも活用できるようにアセット化

ガイドライン

明文化・基準化

ライブラリ化



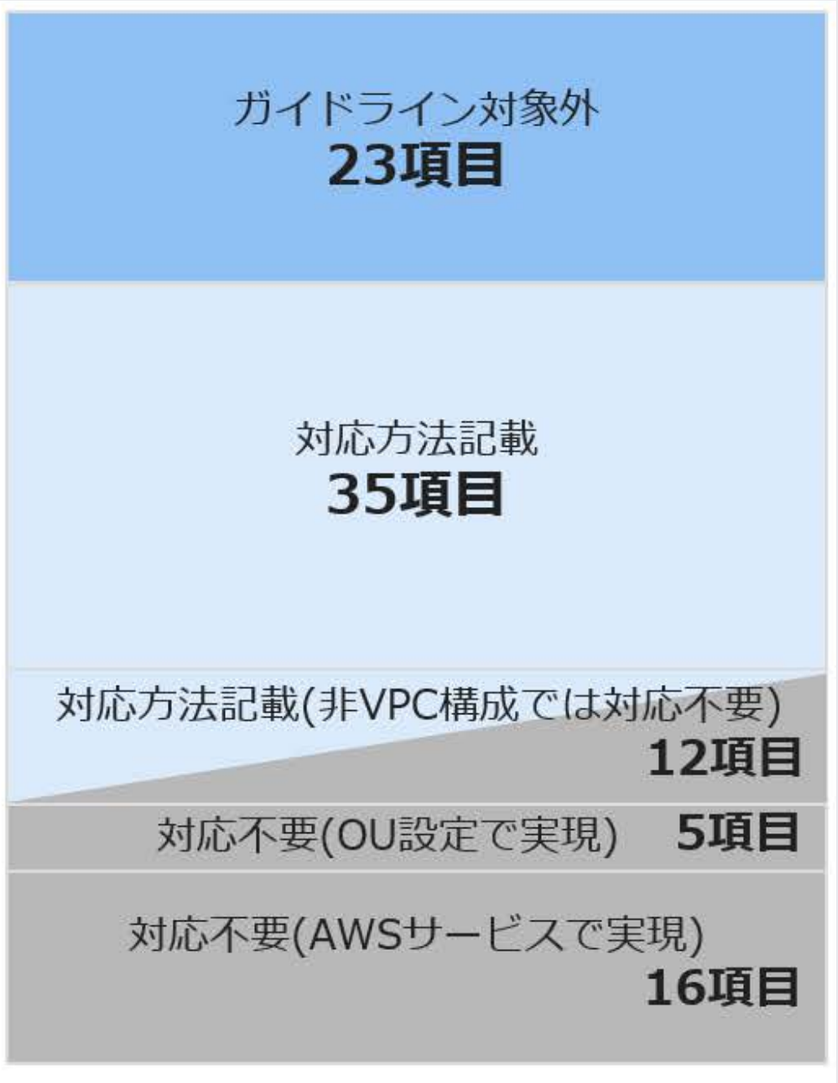
**海外の開発者も
判るようになる**

明文化・基準化することで判る事が出来る

誰がやっても同じものはライブラリ化すれば
作業効率が上がる・ミスがなくなる

© 2021, transcosmos, inc. or its affiliates. All rights reserved.

IPA非機能要件との付け合わせ



#	分類	項目数	ガイドライン記載内容
1	サービス部門側で独自検討・対応が必要	23項目	システムの特性により対応方法が異なるため、要件レベルと検討すべき内容を記載
2	ガイドラインに対応方法を記載	35項目	AWSリソースを利用した対応方法を記載
3	ガイドラインに対応方法を記載(非VPC構成で対応不要)	12項目	AWSリソースを利用した対応方法を記載 ただし、非VPC構成(サーバレス)であれば対応不要(※1)
4	サービス部門側で対応不要(標準OUで対応)	5項目	中央管理部門が提供する標準OUアカウントを利用することで、最高レベルの要件を充足可能であり、どのような設定で対応しているかを記載
5	サービス部門側で対応不要(AWSで対応)	16項目	AWSを利用することで、最高レベルの要件を充足可能であり、対応不要となる理由を記載

© 2021, transcosmos, inc. or its affiliates. All rights reserved.

事例やTips等も集約

- AWS Lambda最適化による費用削減
- Amazon DynamoDBのテーブル、アクセス設計による費用削減
- 不要なリソースの削除
- API Gatewayで入出力検証を実行することでコスト削減
- 費用管理(アラート設定、費用内訳解析)
- Amazon S3のライフサイクル設定による費用削減

- 大量アクセス時のSQSの処理漏れ回避
- 大量アクセス時のDB接続不足回避
- 大量アクセス時のDBデッドロック回避
- 大量アクセスを想定したDynamoDBの設定

- API Gatewayタイムアウトを想定したエラー処理
- エラー発生時のリトライ対応
- DeadLetterQueueを用いたエラー処理

- ログレベルの方針
- エラー解析に有用なCloudWatch Insightの利用
- エラー通知内容

- Amazon S3のデータ保護
- セキュリティグループの設定方針
- 内部サービスを安全に利用するための踏み台サーバ設定

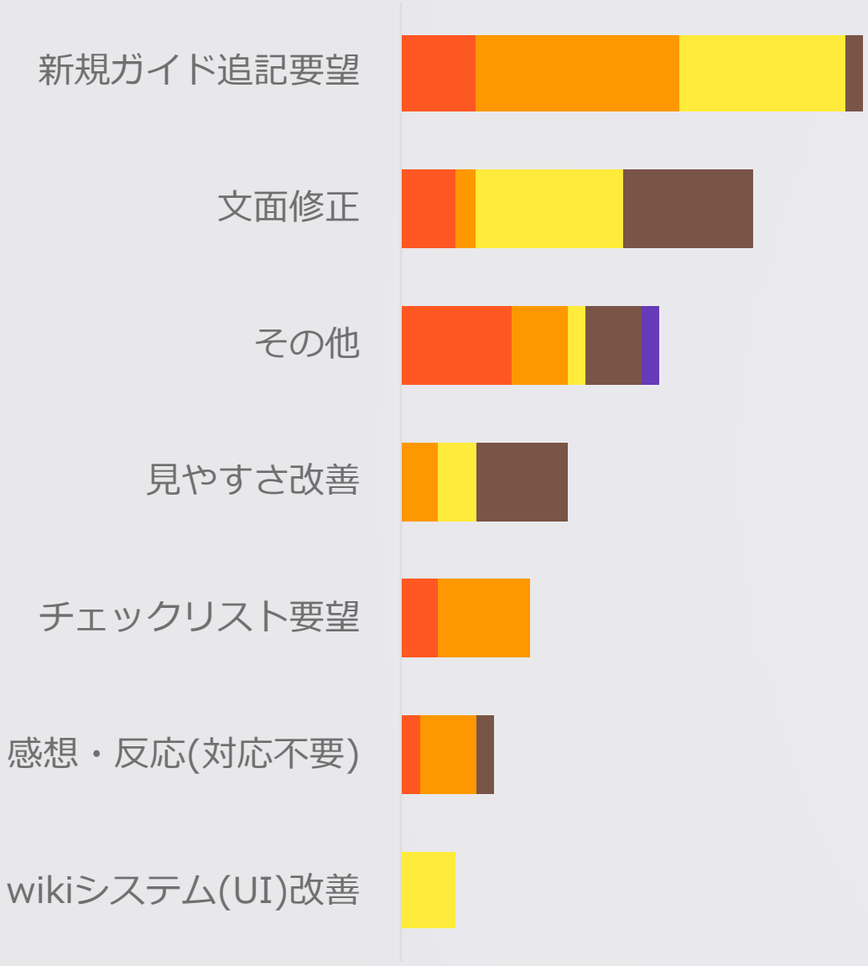
- FIFO SQSで、並列処理を実施
- X-Rayによる実行ボトルネック解析
- AppSync(GraphQL)を用いた効率の良いデータ送受信

- VPC Lambdaに必要なNW設定
- イベント処理に対する二重処理回避

© 2021, transcosmos, inc. or its affiliates. All rights reserved.

各部からのフィードバック

これまでこういう類のサイト・情報が社内には無かったということで
意義、目的、目標などについては好評　追加要望も来ている



- 外部セキュリティチェックリスト、監査への対応事例
- 中央管理が難しいアカウントを前提とした統制の指針
- 記載されていないAWSサービスについてのガイドの追加

- 出展の記述
- 前提条件の追記
- 誤記修正

- ガイドライン準拠によるオンラインサービス利用申請簡略化
- wiki更新情報のキャッチアップ方法
- 社内規約、ISO、Pマークなどへの準拠、考慮状況

- 検索性の強化
- 全体像をより把握しやすく チェックリスト要望
- ガイドラインへ準拠出来ているかのチェックリストの提供

- 開発者が適正なサービス・設定値の選択および必要な運用措置を検討する上で、大きな手助けになる
- 社内共通のルールが記載されたものは、業務を遂行する上でとても有意義
- 継続して現場視点でアップデートしてくれたら重要なサイトになる

- wikiメニュー表示、遷移時の挙動改善

© 2021, transcosmos, inc. or its affiliates. All rights reserved.

今後も拡充

© 2021, transcosmos, inc. or its affiliates. All rights reserved.

今後も拡充

日本

サービスプロダクト管理
日本語・要件詳細詰めなど
日本でしか難しいところを担当

日本固有のライブラリ化
・ドキュメント化の依頼

要件詳細
ライブラリ

海外PM

プロジェクト管理
開発管理・基準・ライブラリ
ASEANの文化や現場の理解など
日本では分からないところを担当

実装支援・詳細指示

開発拠点

開発管理
基準の履行
要望に沿って開発・実装
開発時のルールの拡充

保守
更新

拡充

ガイドライン
- 明文化・基準化
- ライブラリ
・パターン化

利用

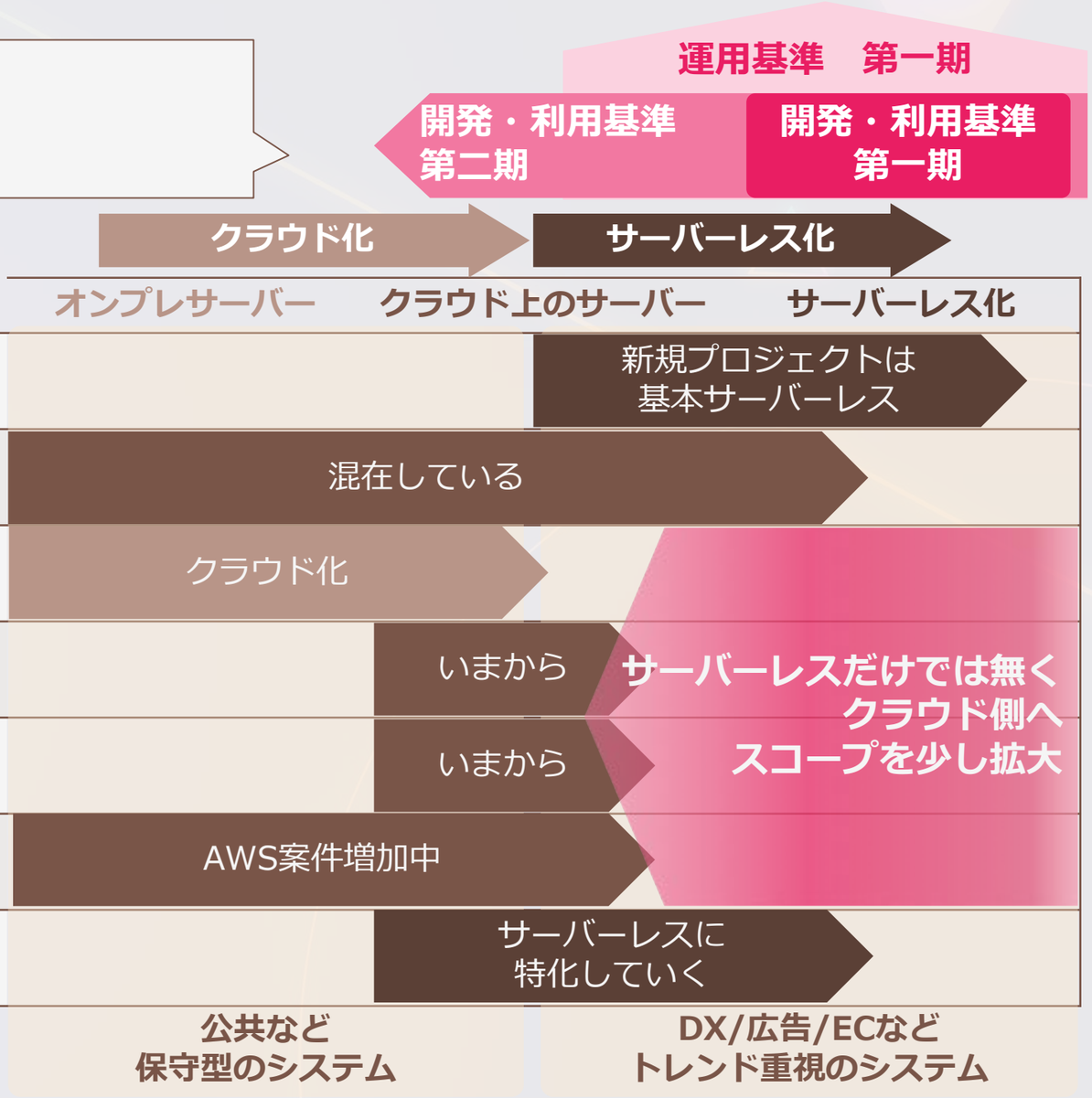
各開発現場が
効率良く開発できる

- 日本
- 韓国
- ベトナム
- その他

© 2021, transcosmos, inc. or its affiliates. All rights reserved.

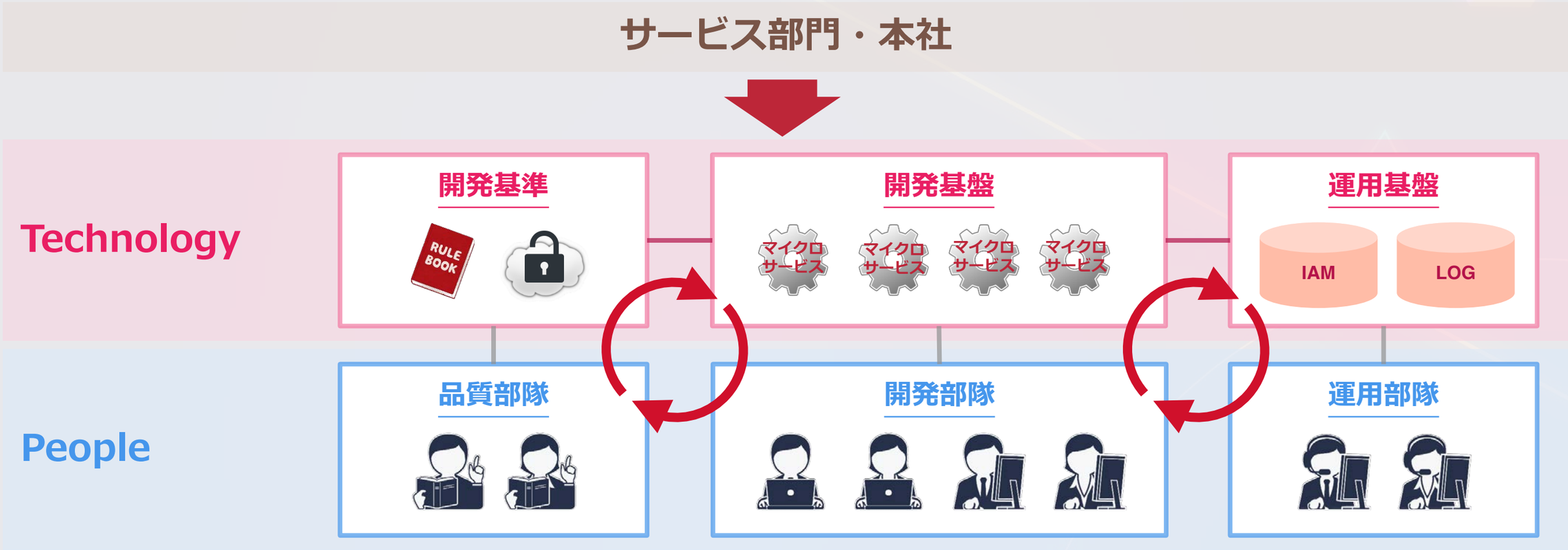
サーバー型・運用・他のPFも対応

サーバーレス中心に立ち上がったので、
今後はその周辺とサーバー型や他のPFも対応する
また、開発物の運用基準を策定する



© 2021, transcosmos, inc. or its affiliates. All rights reserved.

開発運用体制を構築していく



最終的には対象システムや人材を認定していきたい

まずは有効性を証明する

© 2021, transcosmos, inc. or its affiliates. All rights reserved.

Thank you!

山根　径

DevOpsチーム構築のための
AWSサーバーレス起点の技術ガイドライン策定

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.