Security-JAWS 第32回

IaC でセキュリティを強化しよう！

～ IAMが苦手な開発者でも簡単に権限を絞れる。 そう、AWS CDKならね！ ～

みずほリサーチ&テクノロジーズ
技術開発本部
先端技術研究部

2024年2月14日

ともに挑む。ともに実る。
MIZUHO

自己紹介

氏名：松尾　優成（Matsuo Yusei）

所属：先端技術研究部 兼 プロジェクト推進部

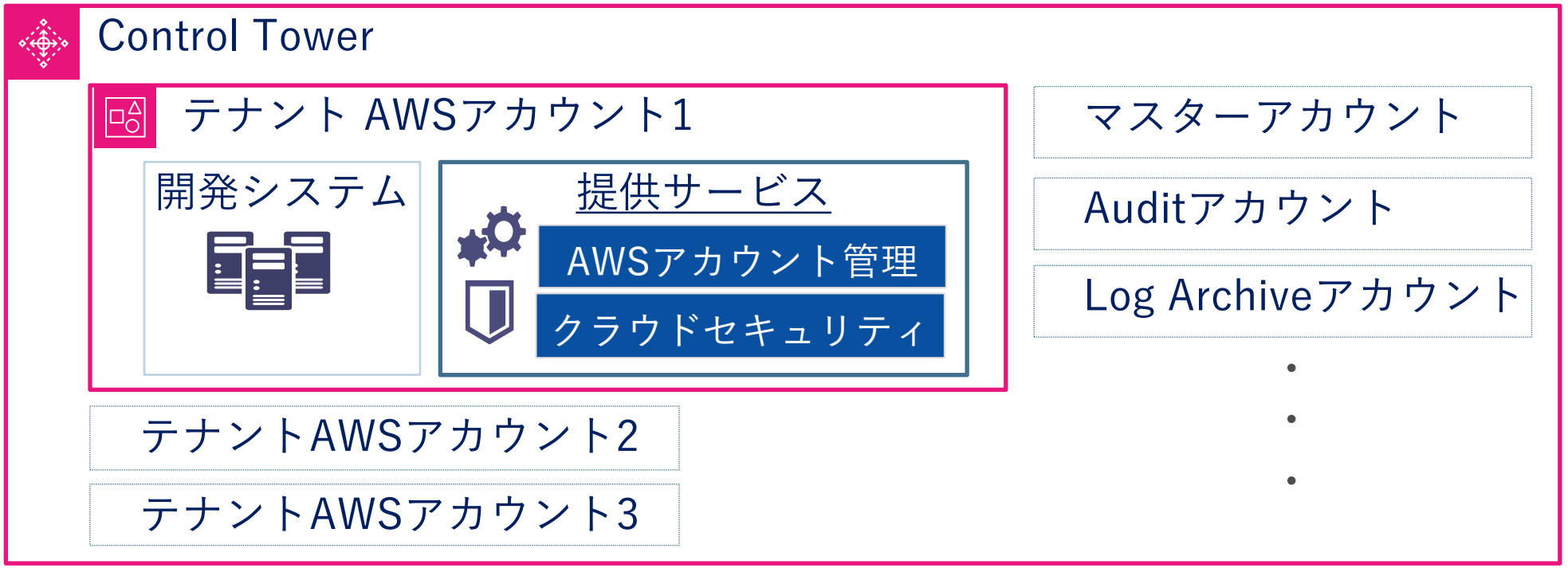
役割：社内向けAWS共通プラットフォームの
　　構築・運用

一言：#secjaws 3回目の登壇機会に感謝！

社内向けAWS共通プラットフォームの概要

詳細は#secjaws31を
Check！

- 社員が安心してAWS環境を利用できるように
  セキュアなAWSアカウントを迅速に提供するプラットフォーム
- マルチアカウント管理にAWS Control Towerを導入し、開発にAWS CDKを活用



話すこと

- セキュリティ観点でAWS CDKの良いところ
  （タイトルはIAMのみですが、IAM以外にも触れます）
- 当社プラットフォームの設定例をコード付きで紹介

話さないこと

- IaC の概念
- AWS CDKを利用するための権限設定
  （プラットフォーム側でどのように制御するかという話はしません）

吹き出しなどで補足するため
コードの詳細を理解する必要はありません

こんな悩みありませんか？

- IAMポリシーやS3バケットポリシーを書くのが大変
- CloudFormationでセキュリティ設定を定義しているが
  - 読みづらい・・・
  - 変更時に抵抗がある・・・

こんな悩みありませんか？

- IAMポリシーやS3バケットポリシーを書くのが大変
- CloudFormationでセキュリティ設定を定義しているが
  - 読みづらい・・・
  - 変更時に抵抗がある・・・

→ AWS CDKで解決できます！

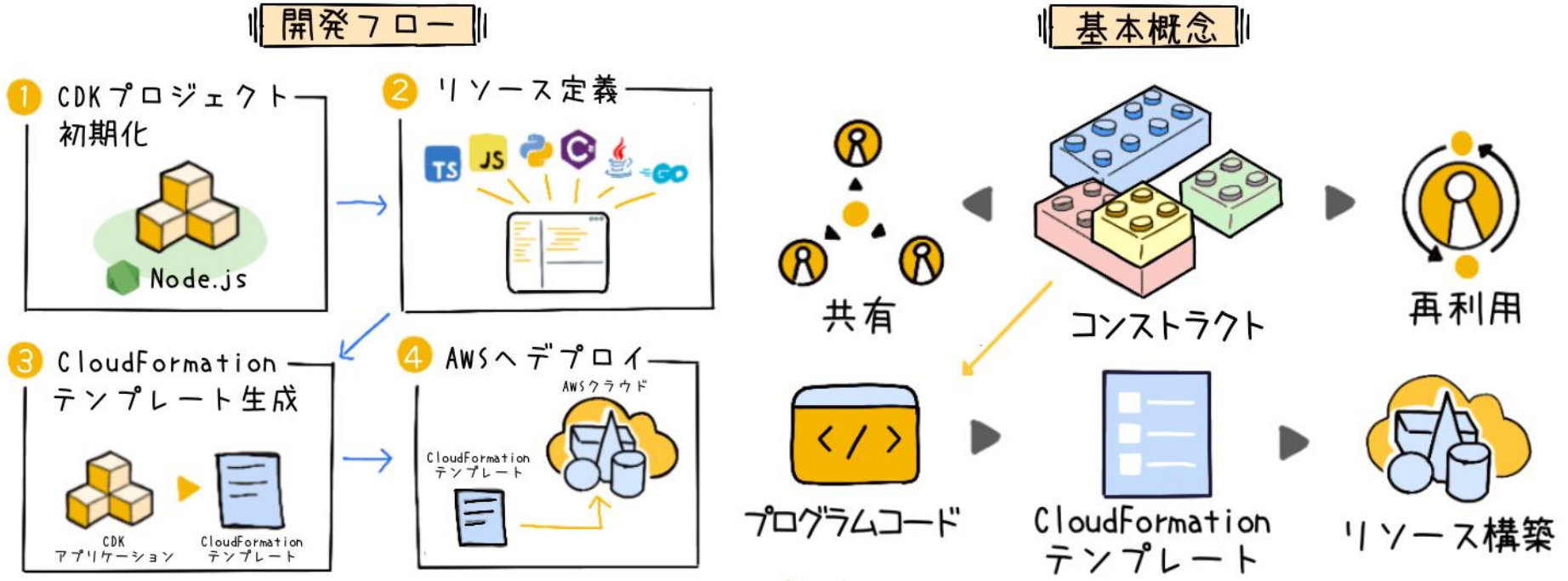
でもAWS CDKの要員は中々いない・・・なぜ？

- ハードルが高いと思われている？
- メリットが十分に伝わっていないかも？

→ セキュリティ観点でのCDKメリットをお伝えします！

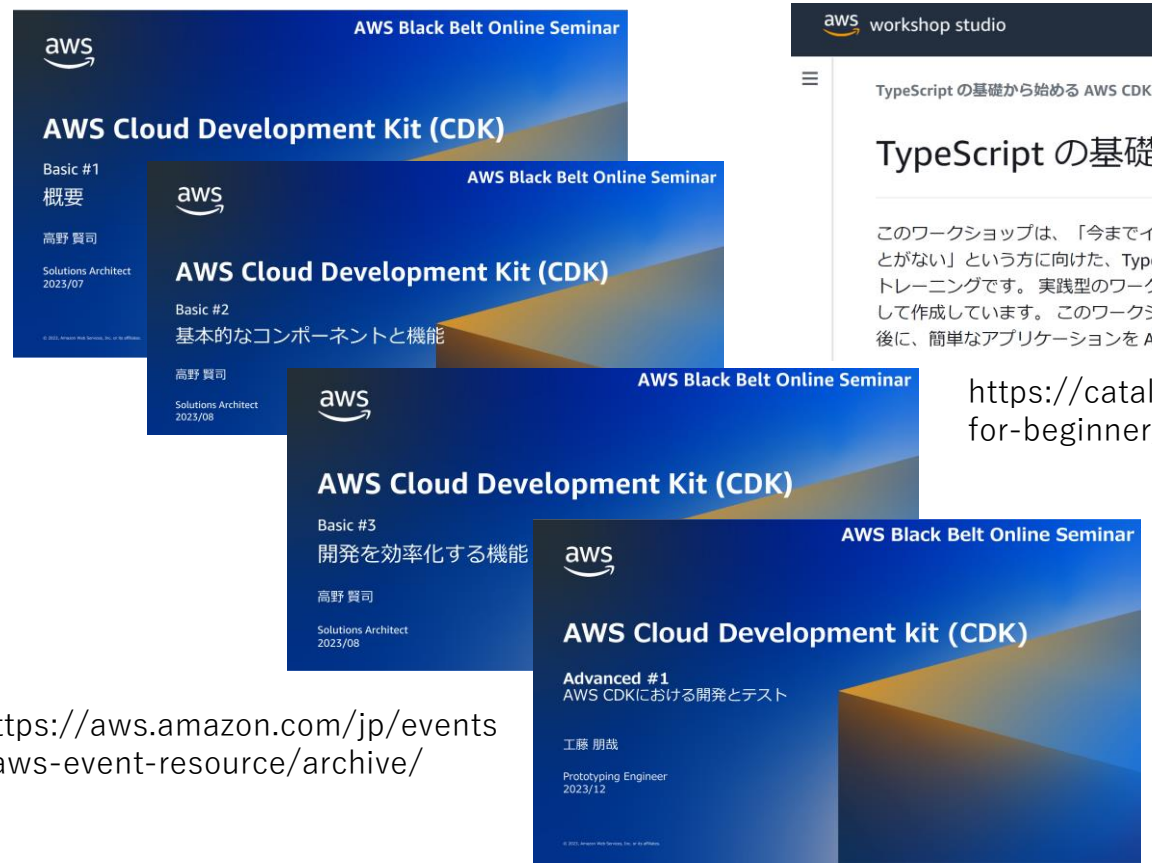
AWS CDK とは…「開発者体験に優れたOSS のIaCツール」

- アプリもインフラも使い慣れたプログラミング言語で構築可
- ベストプラクティスに沿った抽象化されたライブラリ（High-levels Constructs）により少ないコード量で記述可



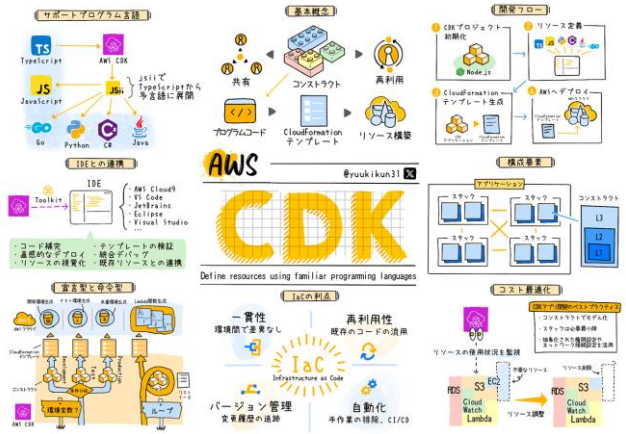
https://aws.amazon.com/jp/builders-flash/202309/awsgeek-aws-cdk/

AWS CDK は学習コンテンツが豊富！



https://aws.amazon.com/jp/events/aws-event-resource/archive/

https://catalog.workshops.aws/typescript-and-cdk-for-beginner/ja-JP



https://aws.amazon.com/jp/builders-flash/202309/awsgeek-aws-cdk/

https://catalog.us-east-1.prod.workshops.aws/workshops/10141411-0192-4021-afa8-2436f3c66bd8/ja-JP

JAWS-UG
AWS User Group - Japan
CDK

初心者にはワークショップ・ BlackBelt シリーズが特にオススメ！

IaC 関連の直近アップデートが熱い！

https://speakerdeck.com/ohmura/managing-existing-environment-with-aws-cfn-iac-generator

Amazon Web Services ブログ

AWS CloudFormation にアプリケーション全体をインポート

by Hiroki Yamazaki | on 06 2月 2024 | in AWS Cloud Development Kit, AWS CloudFormation, General | Permalink | Share

AWS 上で Infrastructure as Code (IaC) を利用することで、インフラストラクチャがスケールするように管理、モデリング、プロビジョニングできます。AWS CloudFormation を使えば YAML や JSON でコードとしてインフラストラクチャを宣言できます。一般的なプログラミング言語を使って AWS Cloud Development Kit (CDK) を利用することもできます。Application Composer で視覚的に管理することもできます。IaC 化されたリソースは、選択したバージョン管理システムで監査およびバージョン管理ができます。また、AWS CloudFormation を使ってデプロイすることで、変更セット を使ったデプロイプレビュー、自動ロールバック、Hooks を使ったリソースコンプライアンスのプロアクティブな適用などが可能になります。非常に多くのお客様が、AWS 上で IaC による安全性と信頼性の恩恵を受けています。

https://aws.amazon.com/jp/blogs/news/import-entire-applications-into-aws-cloudformation/

Amazon Web Services ブログ

CDK Migrate: AWS CDK への移行コマンドの発表

by Hiroki Yamazaki | on 13 2月 2024 | in Announcements, AWS Cloud Development Kit, AWS CloudFormation, General | Permalink | Share

本日は、AWS Cloud Development Kit (CDK) の新機能である CDK Migrate についてご紹介します。この機能を使用することで、ユーザーは以前にデプロイされた AWS CloudFormation テンプレートや CloudFormation スタック、Infrastrcture as Code (IaC) の管理外で作成されたリソースを CDK アプリケーションに移行できます。この機能は、CloudFormation の管理外で作成されたリソースをテンプレートにインポートし、新しく生成された CloudFormation スタックに取り込むのに役立つ CloudFormation IaC ジェネレーター と同時に公開されています。IaC ジェネレーターの詳細は、AWS CloudFormation にアプリケーション全体をインポート をご覧ください。

https://aws.amazon.com/jp/blogs/news/announcing-cdk-migrate-a-single-command-to-migrate-to-the-aws-cdk/

更に IaC の世界へ飛び込みやすくなっています！

ここから**セキュリティ観点**での
AWS CDKメリットを4つ紹介！

（サンプルコードはTypeScript）

・cdk：v2.103.0
・jest：v29.7.0

セキュリティ観点でのAWS CDKメリット①

「ポリシー関連の設定が抽象化されて簡単！」

S3のRead権限についてCFn例（BlackBeltから引用）

https://pages.awscloud.com/rs/112-TZM-766/images/AWS-Black-Belt_2023_AWS-CDK-Basic-1-Overview_0731_v1.pdf

特定のActionに制限

特定のバケットのみに制限

特定のユーザーのみにアタッチ

⇒ セキュアだが、参照を上下に追っていくのは認知負荷が高い？

CDKなら同じ内容をたった3行でかける！

Read以外に
WriteやReadWriteなど
豊富な権限セットを提供

https://pages.awscloud.com/rs/112-TZM-766/images/AWS-Black-Belt_2023_AWS-CDK-Basic-1-Overview_0731_v1.pdf

リソースベースポリシーも簡単にかける！

当社事例のCDKコード（アセットを組織内に展開するS3バケット）

```
const assetsBucket = new s3.Bucket(
  this,
  "AssetsBuckets",
  {
    blockPublicAccess: s3.BlockPublicAccess.BLOCK_ALL,
    encryption: s3.BucketEncryption.KMS,
    encryptionKey: key,
    autoDeleteObjects: true,
    removalPolicy: RemovalPolicy.DESTROY,
    enforceSSL: true,
  },
);
assetsBucket.grantRead(new iam.OrganizationPrincipal("o-xxxx"));
```

ポリシー生成

S3バケットポリシー抜粋

```
{
  "Action": [
    "s3:GetObject*",
    "s3:GetBucket*",
    "s3:List*"
  ],
  "Condition": {
    "StringEquals": {
      "aws:PrincipalOrgID": "o-xxxx"
    }
  },
  "Effect": "Allow",
  "Principal": {
    "AWS": "*"
  },
  "Resource": [
    {
      "Fn::GetAtt": [
        "AssetsBucket1234ABCD",
        "Arn"
      ]
    },
    {
      "Fn::Join": [
        "",
        [
          {
            "Fn::GetAtt": [
              "AssetsBucket1234ABCD",
              "Arn"
            ]
          },
          "/*"
        ]
      ]
    }
  ]
}
```

grantReadの1行で組織IDに属する全AWSアカウントへ
読み取り権限を付与（**約40行**のポリシーを生成）

SSLを強制するポリシーも1行で！

当社事例のCDKコード（アセットを組織内に展開するS3バケット）

```
const assetsBucket = new s3.Bucket(
  this,
  "AssetsBuckets",
  {
    blockPublicAccess: s3.BlockPublicAccess.BLOCK_ALL,
    encryption: s3.BucketEncryption.KMS,
    encryptionKey: key,
    autoDeleteObjects: true,
    removalPolicy: RemovalPolicy.DESTROY,
    enforceSSL: true,
  },
);
asset                                                    -xxxx"));
```

enforeSSLを有効化するだけでSSL以外のリクエストを拒否（**約35行**のポリシーを生成）

ポリシー生成

S3バケットポリシー抜粋

```
{
  "Action": "s3:*",
  "Condition": {
    "Bool": {
      "aws:SecureTransport": "false"
    }
  },
  "Effect": "Deny",
  "Principal": {
    "AWS": "*"
  },
  "Resource": [
    {
      "Fn::GetAtt": [
        "AssetsBucket1234ABCD",
        "Arn"
      ]
    },
    {
      "Fn::Join": [
        "",
        [
          {
            "Fn::GetAtt": [
              "AssetsBucket1234ABCD",
              "Arn"
            ]
          },
          "/*"
        ]
      ]
    }
  ]
}
```

S3以外にKMSなどでも簡単に権限付与！

当社事例のCDKコード（組織内で使用するCMK）

```
const key = new kms.Key(
  this, "AssetsCmk",
);
key.grantDecrypt(new iam.OrganizationPrincipal("o-xxxx"));
```

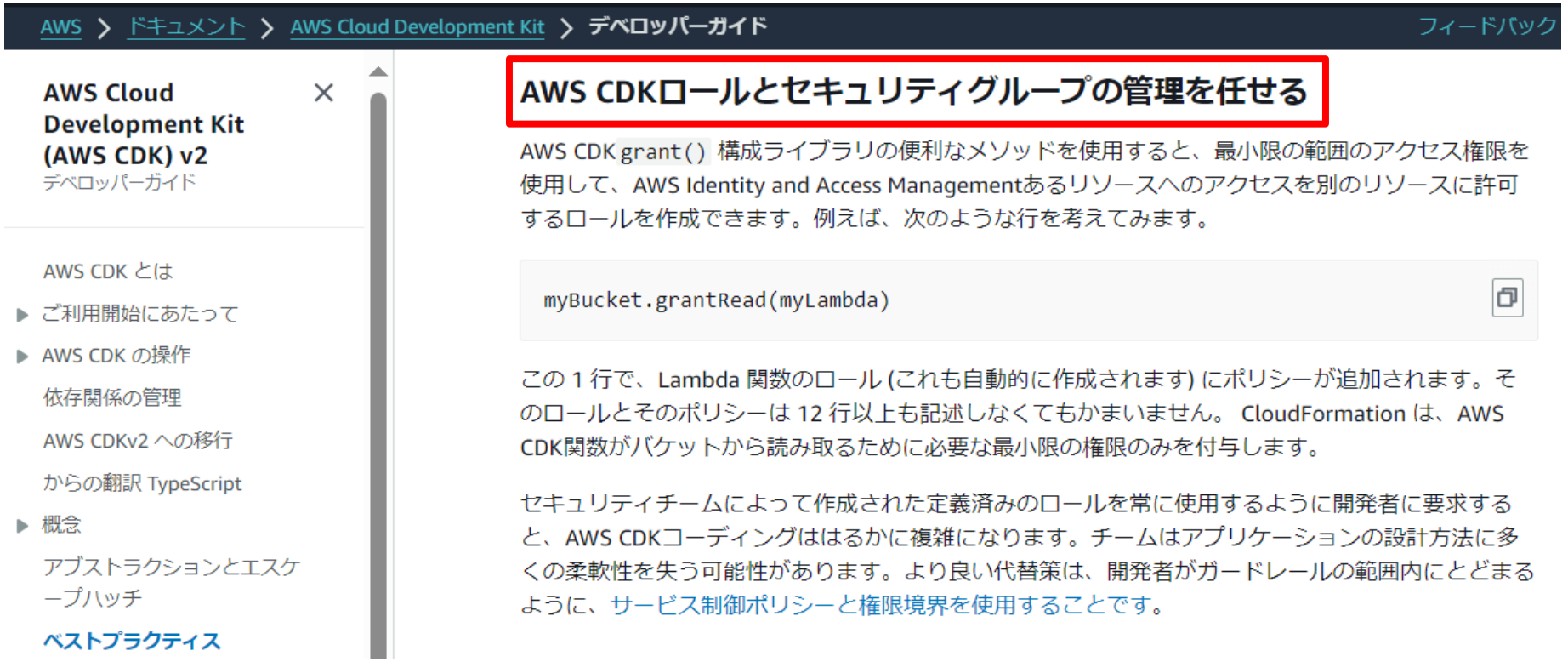
ポリシー生成

KMSキーポリシー抜粋

```
{
  "Action": [
    "kms:Decrypt",
    "kms:DescribeKey"
  ],
  "Condition": {
    "StringEquals": {
      "aws:PrincipalOrgID": "o-xxxx"
    }
  },
  "Effect": "Allow",
  "Principal": {
    "AWS": "*"
  },
  "Resource": "*"
}
```

- サンプルでは組織IDに権限付与しているが IAM RoleやLambda関数にも付与可能
  （IGrantableに対応するリソースが対象）
- 誰がどのリソースに触れるのか分かり易い！
  「アクセス先リソース.grantXX(アクセス者)」の形式

ベストプラクティスに沿って grant を積極的に使おう！



個別に用意した
IAMロール・
ポリシーも適用可！

- セキュリティグループやNACLの設定も簡単
- その他に通知やメトリクスなども抽象化されていて、とにかく便利

セキュリティ観点でのAWS CDKメリット②

「モジュール化と単体テストで設定ミス抑制」

CFnテンプレートで同様なセキュリティ設定が羅列しやすい

CloudWatchでルートユーザーの使用を検知してSNSへ通知するCFn例

```
"CIS11Alarm": {"Properties": {
    "AlarmActions": [{"Ref": "SnsTopic1234"}],
    "AlarmName": "CIS-1.1 Avoid the use of the root user",
    "ComparisonOperator": "GreaterThanOrEqualToThreshold",
    "EvaluationPeriods": 1,
    "MetricName": "CIS-1.1 Avoid the use of the root user",
    "Namespace": "LogMetrics",
    "Period": 60,
    "Statistic": "Average",
    "Threshold": 1
  }, "Type": "AWS::CloudWatch::Alarm"
},
"CIS11MetricFilter": { "Properties": {
    "FilterPattern": "{$.userIdentity.type=\"Root\" && $.userIdentity.invokedBy NOT EXISTS && $.eventType !=
    "LogGroupName": {"Ref": "TrailLogGroup1234"},
    "MetricTransformations": [{
      "MetricName": "CIS-1.1 Avoid the use of the root user",
      "MetricNamespace": "LogMetrics",
      "MetricValue": "1"
    }]
  }, "Type": "AWS::Logs::MetricFilter"
```

この塊が沢山あるイメージ

※Fn::ForEach 未使用の前提

アラーム毎に対応するメトリクスフィルタとの連携設定が必要

CloudWatchでルートユーザーの使用を検知してSNSへ通知するCFn例

```
"CIS11Alarm": {"Properties": {
    "AlarmActions": [{"Ref": "SnsTopic1234"}],
    "AlarmName": "CIS-1.1 Avoid the use of the root user",
    "ComparisonOperator": "GreaterThanOrEqualToThreshold",
    "EvaluationPeriods": 1,
    "MetricName": "CIS-1.1 Avoid the use of the root user",
    "Namespace": "LogMetrics",
    "Period": 60,
    "Statistic": "Average",
    "Threshold": 1
  }, "Type": "AWS::CloudWatch::Alarm"
},
"CIS11MetricFilter": { "Properties": {
    "FilterPattern": "{$.userIdentity.type=\"Root\" && $.userIdentity.invokedBy NOT EXISTS && $.eventType !=
    "LogGroupName": {"Ref": "TrailLogGroup1234"},
    "MetricTransformations": [{
      "MetricName": "CIS-1.1 Avoid the use of the root user",
      "MetricNamespace": "LogMetrics",
      "MetricValue": "1"
    }]
  }, "Type": "AWS::Logs::MetricFilter"
```

正常に通知を機能させるためには
MetricName/NameSpaceを揃える

アラーム側定義と一致する必要あり

アラーム毎に対応するメトリクスフィルタとの連携設定が必要

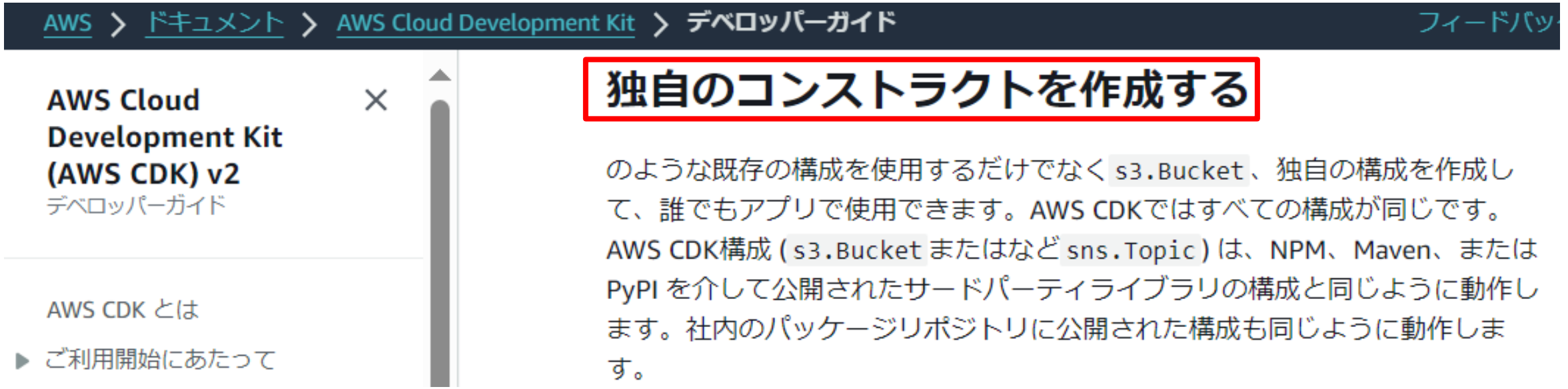
CloudWatchでルートユーザーの使用を検知してSNSへ通知するCFn例

```
"CIS11Alarm": {"Properties": {
    "AlarmActions": [{"Ref": "SnsTopic1234"}],
    "AlarmName": "CIS-1.1 Avoid the use of the root user",
    "ComparisonOperator": "GreaterThanOrEqualToThreshold",
    "EvaluationPeriods": 1,
    "MetricName": "CIS-1.1 Avoid the use of the root user",
    "Namespace": "LogMetrics",
    "Period": 60,
    "Statistic": "Average",
    "Threshold": 1
  }, "Type": "AWS::CloudWatch::Alarm"
},
"CIS11MetricFilter": { "Properties": {
    "FilterPattern": "{$.userIdentity.type=\"Root\" && $.userIdentity.invokedBy NOT EXISTS && $.eventType !=
    "LogGroupName": {"Ref": "TrailLogGroup1234"},
    "MetricTransformations": [{
      "MetricName": "CIS-1.1 Avoid the use of the root user",
      "MetricNamespace": "LogMetrics",
      "MetricValue": "1"
    }]
  }, "Type": "AWS::Logs::MetricFilter"
```

丸々コピペして定義すると、
うっかり修正漏れで通知が
うまくいかないことも・・・

同様の処理をモジュール化することで保守性向上！

AWS CDKではモジュールとして独自のコンストラクトを作成できる



https://docs.aws.amazon.com/ja_jp/cdk/v2/guide/constructs.html#constructs_author

※本発表では、以後「独自のコンストラクト」を「カスタムConstruct」と呼びます

アラーム/メトリクスフィルタをカスタムConstruct化する

当社事例のCDKコード（カスタムConstruct概要）

```
export interface CloudTrailLogsAlarmProps {
  readonly logGroupName: string;
  readonly cisName: string;
  readonly metricsPattern: string;
  readonly snsTopic: sns.Topic;
}

export class CloudTrailLogsAlarm extends Construct {

  constructor(scope: Construct, id: string, props: CloudTrailLogsAlarmProps) {…
  }
}
```

呼び出し元から渡された値を
リソース定義に使用

MetricName/NameSpaceが
一致するようにモジュール化

カスタムConstructのリソース定義抜粋

```
const metric = new cloudwatch.Metric({
  namespace: "LogMetrics",
  metricName: props.cisName,
  period: Duration.minutes(1),
  statistic: "avg",
});

new logs.MetricFilter(this, "MetricFilter" + props.cisName, {
  logGroup,
  metricNamespace: metric.namespace,
  metricName: metric.metricName,
  filterPattern: logs.FilterPattern.literal(props.metricsPattern),
  metricValue: "1",
});

const alarm = new cloudwatch.Alarm(this, "Alarm" + props.cisName, {…
});
alarm.addAlarmAction(new cw_actions.SnsAction(props.snsTopic));
```

単体テストでカスタムConstructのロジックを確認！

当社事例のCDKコード（カスタムConstructの単体テスト抜粋）

```
test("MericFilterとAlarmでMetricName/NameSpaceが一致している", () => {
  const [filterMetricName, alarmMetricName, filterNamespace, alarmNamespace] =
      [new Capture(), new Capture(), new Capture(), new Capture()];
  template.hasResourceProperties("AWS::Logs::MetricFilter", {
    MetricTransformations: [{
      MetricName: filterMetricName,
      MetricNamespace: filterNamespace,
    }],
  });
  template.hasResourceProperties("AWS::CloudWatch::Alarm", {
    MetricName: alarmMetricName,
    Namespace: alarmNamespace,
  });
  expect(filterMetricName).toEqual(alarmMetricName);
  expect(filterNamespace).toEqual(alarmNamespace);
});
```

メトリクスフィルタとアラームで
MetricName/NameSpaceが
一致していることを確認

単体テストでカスタムConstructのロジックを確認！

当社事例のCDKコード（カスタムConstructの単体テスト抜粋）

```
test("MericFilterとAlarmでMetricName/NameSpaceが一致している", () => {
  const [filterMetricName, alarmMetricName, filterNamespace, alarmNamespace] =
      [new Capture(), new Capture(), new Capture(), new Capture()];
  template.hasResourceProperties("AWS::Logs::MetricFilter", {
    MetricTransformations: [{
      MetricName: filterMetricName,
      MetricNamespace: filterNamespace,
    }],
  });
  template.hasResourceProperties("AWS::CloudWatch::Alarm", {
    MetricName: alarmMetricName,
    Namespace: alarmNamespace,
  });
  expect(filterMetricName).toEqual(alarmMetricName);
  expect(filterNamespace).toEqual(alarmNamespace);
});
```

テスト実行結果（npm test）

```
PASS test/constructs/modules/cloudTrailLogsAlarm.test.ts
  CloudTrailLogsAlarmのテスト
    リソース数の確認
      ✓ MetricFilterの数が1である (300 ms)
      ✓ Alarmの数が1である (69 ms)
    プロパティチェック
      ✓ MericFilterとAlarmでMetricName/NameSpaceが一致している
```

単体テストにより変更容易性も高まる！

テスト済のカスタムConstructなら、繰り返し処理が安全！

当社事例のCDKコード（スタック定義）

```
    for (const constArg of cisMetricsPatterns) {
      const arg = {
        logGroupName: logGroup.logGroupName,
        cisName: constArg.cisName,
        metricsPattern: constArg.metricsPattern,
        snsTopic: cisTopic,
      };
      new CloudTrailLogsAlarm(this, constArg.cisName, arg);
    }
  }
}

const cisMetricsPatterns = [
  {
    cisName: "CIS-1.1 Avoid the use of the root user",
    metricsPattern:
      '{$.userIdentity.type="Root" && $.userIdentity.invoke
  },
  {
    cisName: "CIS-3.2 Ensure a log metric filter and alarm e
    metricsPattern:
      '{ ($.eventName = "ConsoleLogin") && ($.additionalEver
```

メトリクスパターンの定義数に応じてループ処理

カスタムConstructを呼び出し（アラーム・メトリクスフィルタを作成）

メトリクスパターンの定義

セキュリティ観点でのAWS CDKメリット③

「CloudFormationサポート外リソースを
カスタムリソースで簡単かつセキュアに設定」

実務ではCFnサポート外のリソース操作に度々遭遇

【CFnサポート外の操作例】

1. GuardDutyのS3エクスポート設定
2. Configルールのタグづけ
3. Securiry Hub 標準コントロールの無効化（※2023年6月に対応済）
4. Service Catalog ポートフォリオの IAM Principal 関連付け

これらは手動設定になりがちだが何度も設定するものはIaCにしたい

一方、CFn カスタムリソースはLambdaが必要でハードルが高そう…

AWS CDKでは、カスタムリソースを簡単に記述可能！

https://pages.awscloud.com/rs/112-TZM-766/images/AWS-Black-Belt_2023_AWS-CDK-Basic-3-AppDev_0831_v1.pdf

カスタムリソース上のAPIに応じたポリシーを簡単に作成！

当社事例のCDKコード
（Service Catalog ポートフォリオの IAM Principal 関連付け）

```
new cr.AwsCustomResource(this, id, {
  onCreate: this.paramAssociatePrincipal(
    "associatePrincipalWithPortfolio",
    id,
    props
  ),
  onDelete: this.paramAssociatePrincipal(
    "disassociatePrincipalFromPortfolio",
    id,
    props
  ),
  policy: cr.AwsCustomResourcePolicy.fromSdkCalls({
    resources: cr.AwsCustomResourcePolicy.ANY_RESOURCE,
  }),
});
```

ポリシー生成

カスタムリソース（Lambda関数）のポリシー抜粋

```
{
  "Statement": [
    {
      "Action": "servicecatalog:AssociatePrincipalWithPortfolio",
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Action": "servicecatalog:DisassociatePrincipalFromPortfolio",
      "Effect": "Allow",
      "Resource": "*"
    }
  ],
  "Version": "2012-10-17"
}
```

リソースの細かい指定もでき、最小権限にしやすい！

当社事例のCDKコード
（Service Catalog ポートフォリオの IAM Principal 関連付け）

```
new cr.AwsCustomResource(this, id, {
  onCreate: this.paramAssociatePrincipal(
    "associatePrincipalWithPortfolio",
    id,
    props
  ),
  onDelete: this.paramAssociatePrincipal(
    "disassociatePrincipalFromPortfolio",
    id,
    props
  ),
  policy: cr.AwsCustomResourcePolicy.fromSdkCalls({
    resources: cr.AwsCustomResourcePolicy.ANY_RESOURCE,
  }),
});
```

API Reference | Python | Java | .NET | Go | Developer Guide | Examples | C

› Structs

resources

Type: `string[]`

The resources that the calls will have access to.

It is best to use specific resource ARN's when possible. However, you can also use `AwsCustomResourcePolicy.ANY_RESOURCE` to allow access to all resources. For example, when `onCreate` is used to create a resource which you don't know the physical name of in advance.

Note that will apply to ALL SDK calls.

https://docs.aws.amazon.com/cdk/api/v2/docs/aws-cdk-lib.custom_resources.SdkCallsPolicyOptions.html

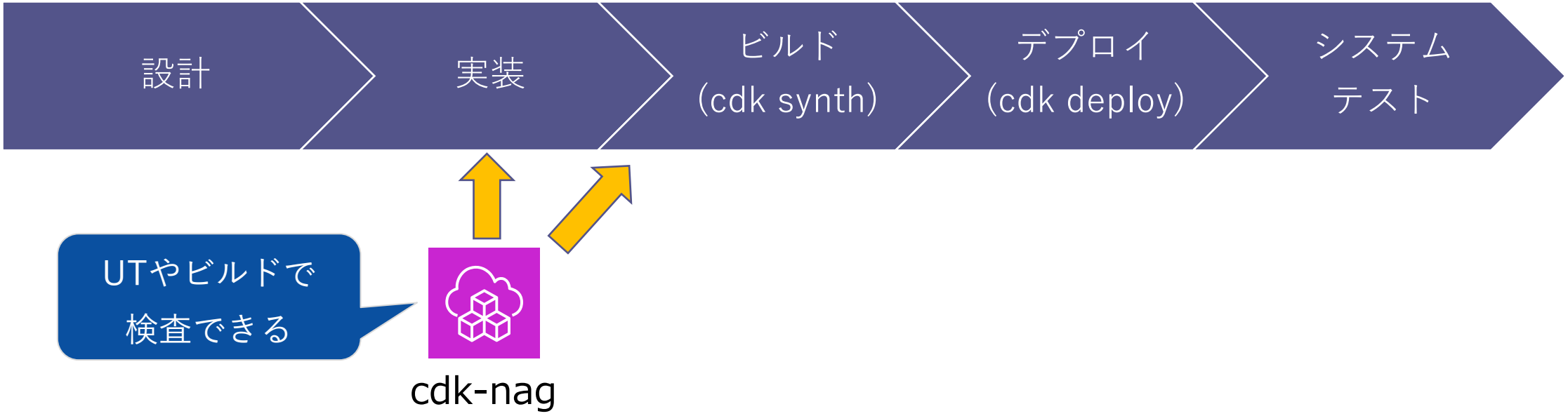
※個別に用意したIAMロール・ポリシーも使用可能

セキュリティ観点でのAWS CDKメリット④

「cdk-nag でセキュリティのシフトレフト」

Policy Validation：「cdk-nag」とは

- セキュリティ・コンプライアンスチェックツール
- デプロイ前に違反リソースがあれば、デプロイを強制停止
- 簡単に導入でき、独自ルールの作成などカスタマイズも可能



様々なルールやそれらをまとめたマネージドのパックあり

Available Rules and Packs

See RULES for more information on all the available packs.

1. AWS Solutions
2. HIPAA Security
3. NIST 800-53 rev 4
4. NIST 800-53 rev 5
5. PCI DSS 3.2.1

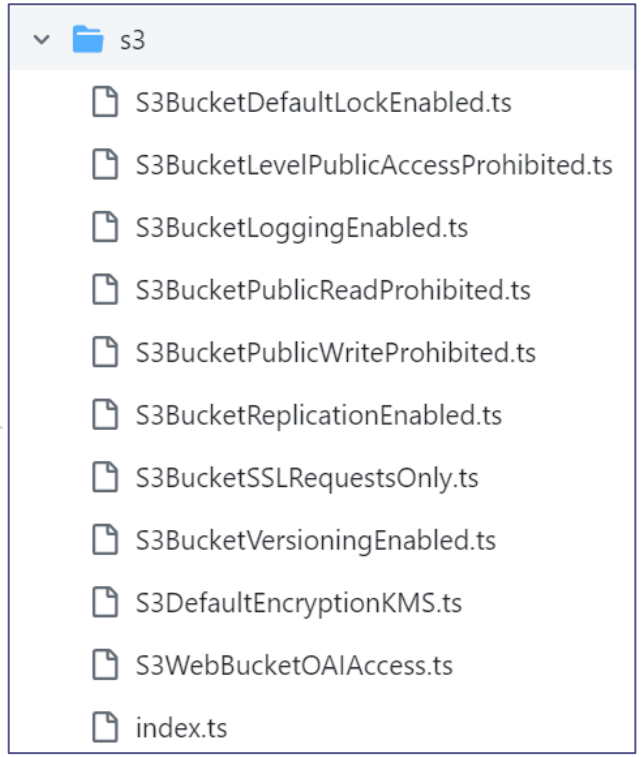
マネージドのルールパック

RULES also includes a collection of additional rules that are not currently included in any of the pre-built NagPacks, but are still available for inclusion in custom NagPacks.

Read the NagPack developer docs if you are interested in creating your own pack.

https://github.com/cdklabs/cdk-nag

S3のルール例



当社では自社ポリシーに合わせて
独自のルールパックを使用

単体テストで cdk-nag を使うと早期に違反を検知できる！

デフォルト設定のS3バケット

```
new s3.Bucket(this, 'Bucket');
```

cdk-nag 単体テスト実行例

```
● S3バケットのテスト › cdk-nagの確認 › No unsuppressed Errors

  [error at /TestStack/ServiceCatalogProductTemplate/Bucket83908E77]
    AwsSolutions-S1: The S3 Bucket has server access logs disabled.
  [error at /TestStack/ServiceCatalogProductTemplate/Bucket83908E77]
    AwsSolutions-S10: The S3 Bucket or bucket policy does not require requests to use SSL.
```

サーバーアクセスログが無効化、SSLが強制されていない旨を指摘

（近年、S3のサービスアップデートによりデフォルトでの指摘が減少している）

cdk-nag の単体テスト Tips は以下記事をご参照！

@y_matsuo_ (Yusei Matsuo) in みずほリサーチ&テクノロジーズ株式会社　先端技術研究部

【AWS】cdk-nagで単体テストのメッセージを整形し、コーディングしながら爆速でセキュリティ遵守する！

AWS Security TypeScript Jest CDK

投稿日 2023年04月07日

はじめに

近年、セキュリティのシフトレフトが注目を浴びています。セキュリティのシフトレフトとは、開発ライフサイクルの早い段階でセキュリティチェックしていこうねという考え方です。

タイトルにもある cdk-nag とは、AWS CDK と統合したセキュリティ・コンプライアンスをチェックできるツールです。cdk-nag の事例は増えており、私が所属するチームでも、年明けから cdk-nag を本格的に利用し始めました。

https://qiita.com/y_matsuo_/items/6ef17964ff3c557f6d1e

テストコード共通部分の修正

エラー件数の相違については、beforeAll を beforeEach に修正することで、回避可能です。beforeEach に変更すると、各テストケース実行前に app・スタックを再作成します。修正結果は以下の通りです。

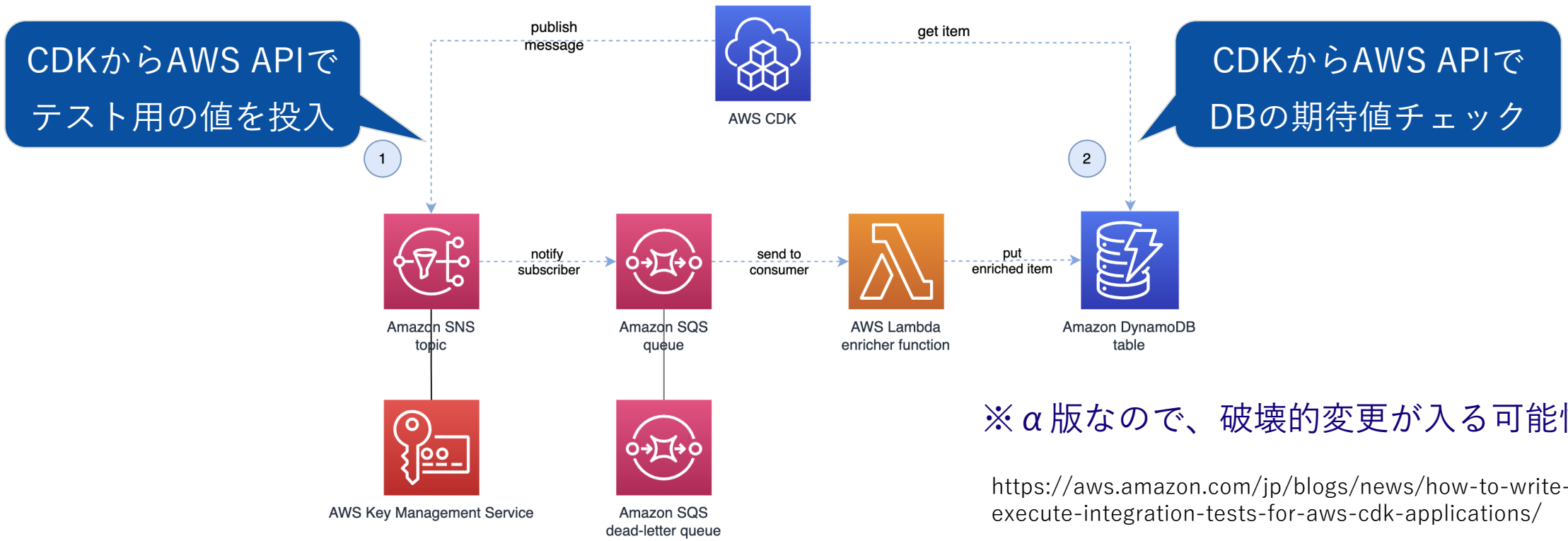
```
  // In this case we can use beforeAll() over beforeEach() since our tests
  // do not modify the state of the application
- beforeAll(() => {
+ beforeEach(() => {
    // GIVEN
    app = new App();
    stack = new CdkTestStack(app, "test");

    // WHEN
    Aspects.of(stack).add(new AwsSolutionsChecks());
  });
```

セキュリティ観点でのAWS CDKメリットになり得るかも？

「integ-tests でアクセステストの自動化」

AWS CDK integ-tests とは

- 実環境に一時的なリソースをデプロイして統合テストを行う
- HTTPやAWS API、Lambda実行などで期待通りの挙動か確認

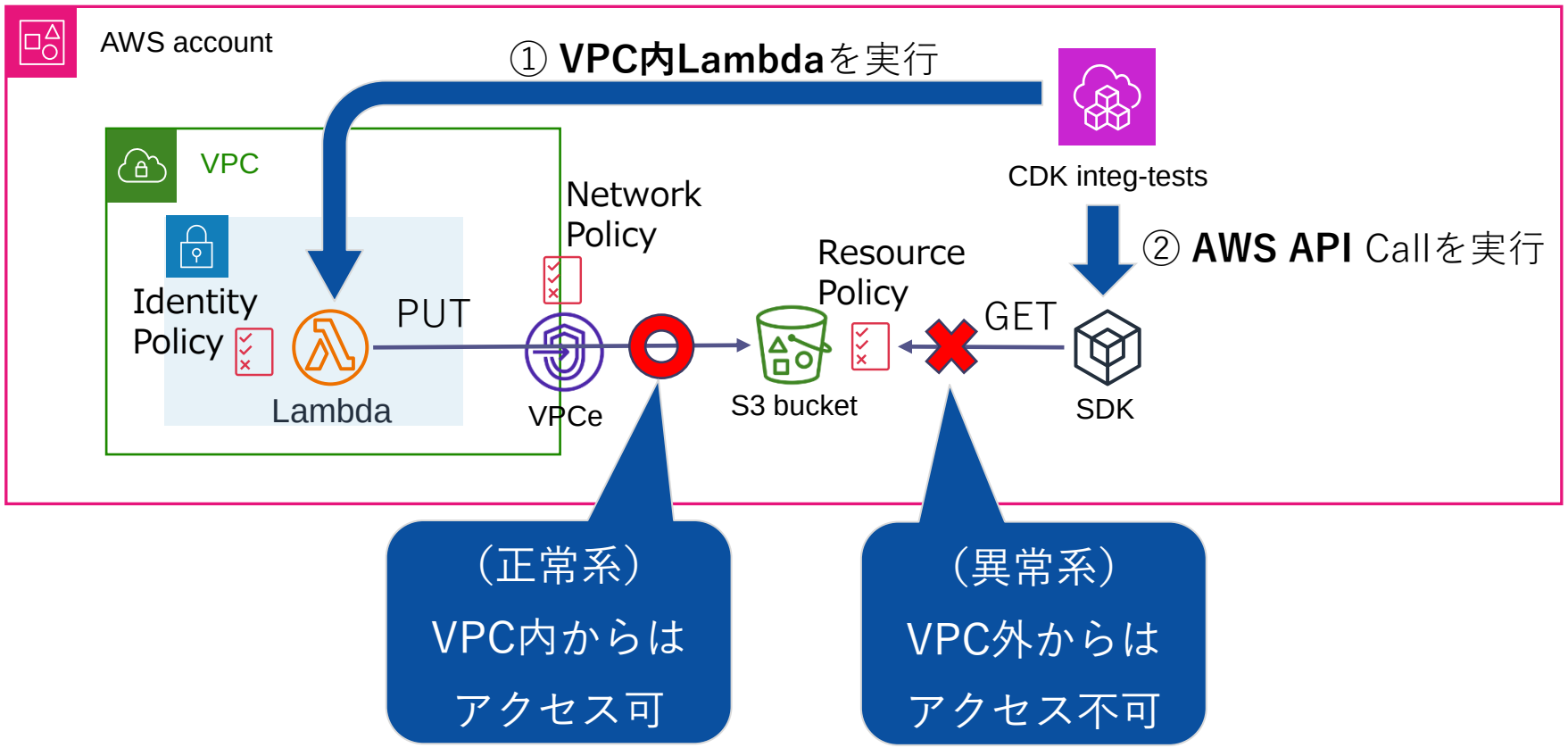


※α版なので、破壊的変更が入る可能性あり

https://aws.amazon.com/jp/blogs/news/how-to-write-and-execute-integration-tests-for-aws-cdk-applications/

integ-testsを使えば、データ境界のような複雑な環境で正常系/異常系のアクセステストを自動化できる？

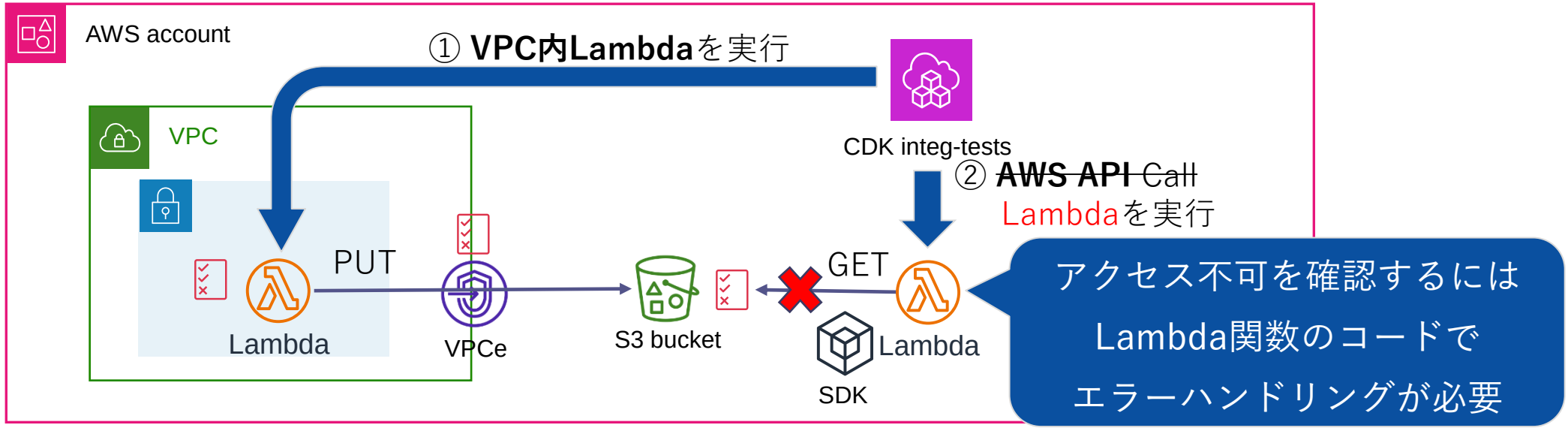
データ境界の当社例（#secjaws23 を参照）



データ境界における integ-tests の検証結果

- テスト実行時間が約25分（早い時は7～9分程度だが、バラつきあり）
- コードレスなAWS APIのメソッドでは正常アクセスのみ確認可

（Access Deniedを確認したい場合は、Lambda関数が必要）



データ境界のアクセステストにおける integ-tests の所感

- integ-tests ではアクセステストに限定せず、データ入出力に着目した正常処理の一環で権限確認するのがよさそう
  →パイプラインがあるなら無理して integ-tests を導入しなくてもよいかも？
  （パイプライン上でもテストできる）

- 各種ポリシーが固まっていない状況なら
  通常通りスタックをデプロイして手動テストする方が早そう

まとめ

- セキュリティ観点でのAWS CDKメリットは以下
  1. ポリシー関連の設定が抽象化されて簡単
  2. モジュール化と単体テストで設定ミス抑制
  3. CFnサポート外操作をカスタムリソースで簡単かつセキュアに設定
  4. cdk-nag でセキュリティのシフトレフト
- 制限の厳しい環境で integ-tests を使用する場合、
  正常処理の一環で権限確認する方がよさそう
- AWS CDKは便利！ぜひ使ってみよう！

ともに挑む。ともに実る。

MIZUHO