[AWS Black Belt Online Seminar]

AWS Command Line Interface

サービスカットシリーズ

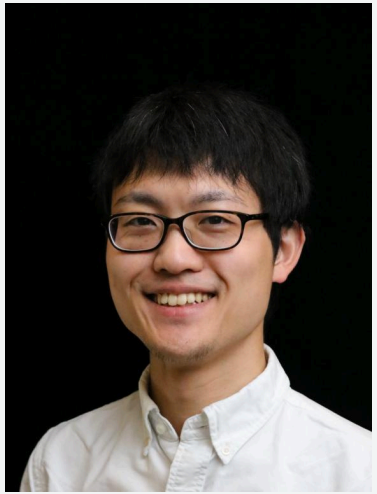
アマゾンウェブサービスジャパン
ソリューションアーキテクト

内田 大樹
2019/7/24

AWS 公式 Webinar
https://amzn.to/JPWebinar

過去資料
https://amzn.to/JPArchive

自己紹介



内田 大樹 (うちだ ひろき)

インダストリソリューション部
ソリューションアーキテクト

普段の業務

エンタープライズのお客様を担当
主に製造業のお客様を担当し、お客様のAWS活用を技術的にサポート

好きなAWSのサービス

- AWS Command Line Interface

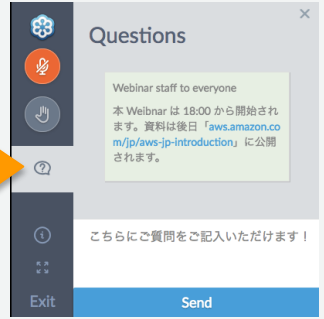
AWS Black Belt Online Seminar とは

「サービス別」「ソリューション別」「業種別」のそれぞれのテーマに分かれて、アマゾン ウェブ サービス ジャパン株式会社が主催するオンラインセミナーシリーズです。

質問を投げることができます！

- 書き込んだ質問は、主催者にしか見えません
- 今後のロードマップに関するご質問は
  お答えできませんのでご了承下さい

① 吹き出しをクリック
② 質問を入力
③ Sendをクリック



Twitter ハッシュタグは以下をご利用ください
#awsblackbelt

内容についての注意点

- 本資料では2019年7月24日時点のサービス内容および価格についてご説明しています。最新の情報はAWS公式ウェブサイト(http://aws.amazon.com)にてご確認ください。

- 資料作成には十分注意しておりますが、資料内の価格とAWS公式ウェブサイト記載の価格に相違があった場合、AWS公式ウェブサイトの価格を優先とさせていただきます。

- 価格は税抜表記となっています。日本居住者のお客様が東京リージョンを使用する場合、別途消費税をご請求させていただきます。

- AWS does not offer binding price quotes. AWS pricing is publicly available and is subject to change in accordance with the AWS Customer Agreement available at http://aws.amazon.com/agreement/. Any pricing information included in this document is provided only as an estimate of usage charges for AWS services based on certain information that you have provided. Monthly charges will be based on your actual use of AWS services, and may vary from the estimates provided.

アジェンダと想定読者

アジェンダ

- 概要
- セットアップ方法
- 操作方法
- 設定
- 操作パターン集

想定読者

- AWS Command Line Interfaceをこれから利用される方
- AWS Command Line Interfaceを利用中の方

概要

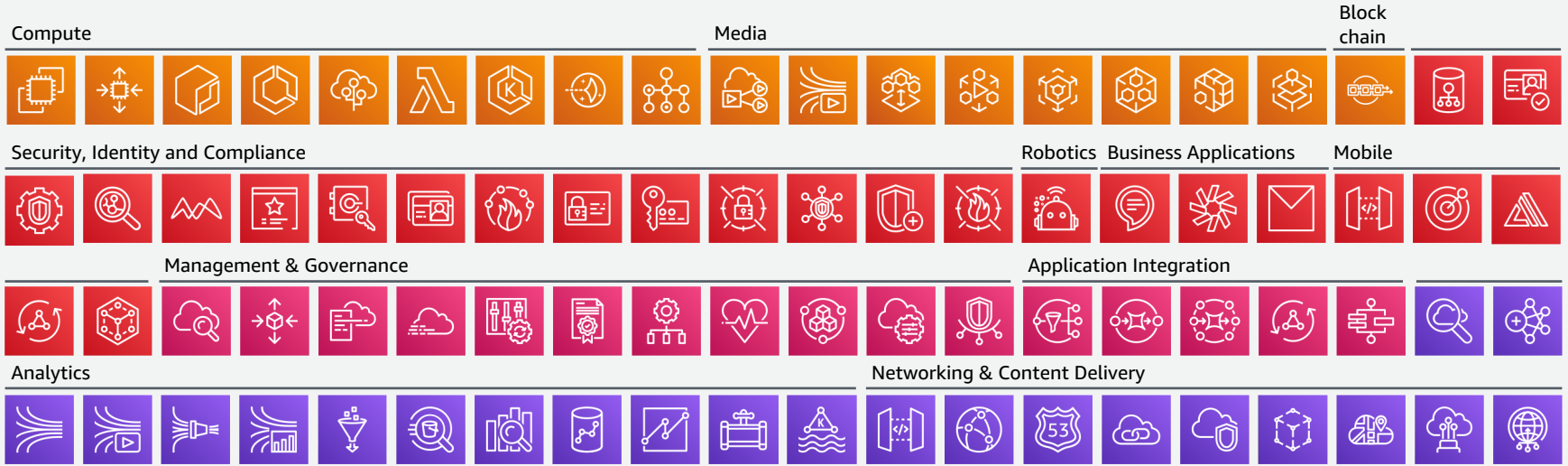
aws

AWS環境に対するオペレーション方法

	手段	実行対象
アプリケーション	手作業、自作スクリプト、CodeDeploy …	アプリケーション、アプリケーション設定…
ミドルウェア	手作業、OpsWorks、Chef、Ansible …	ミドルウェア、ミドルウェア設定…
AWS	  AWS Management Console  AWS CloudFormation  SDK	 Amazon EC2  Amazon RDS  Amazon Simple Storage Service (S3)  Amazon VPC

AWS Command Line Interface (AWS CLI)とは

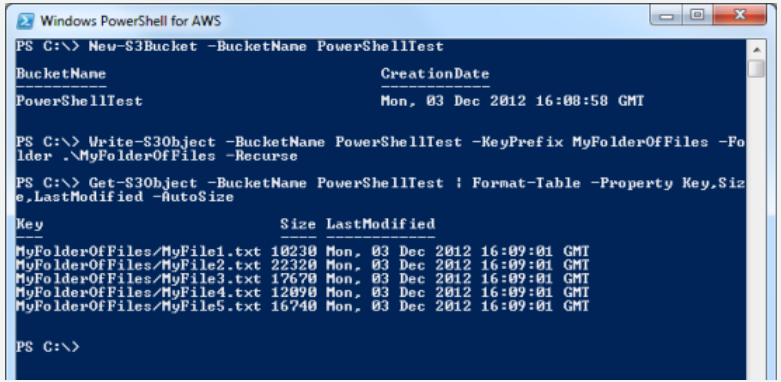
AWSの各種サービスをコマンドで操作できるOSSの統合ツールです。**AWS CLIのサブコマンドはAWSのAPI実装に追従しており、 ほぼ全てのAWSサービス、APIの操作を網羅しています。※**
Windows、Linux、Mac、Unixプラットフォームに対応しており、S3用にはsyncなどの便利な機能(High-Level API)もご利用できます。



※新機能の追加に従って常に更新されるため、定期的にAWS CLIをアップデートすることを推奨します。

(参考) AWS Tools for Windows PowerShell

`AWSPowerShell` モジュール内のコマンドレットから、AWSサービスを操作可能。前提条件としては、Windows XP以降、PowerShell 2.0以降のPowerShellです。設定手順、使用方法につきましては下記URLをご参照ください。

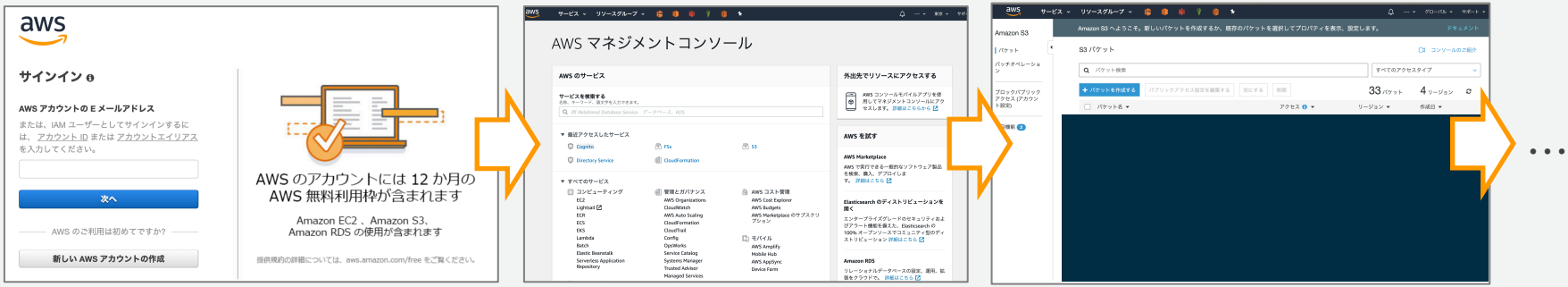


https://aws.amazon.com/jp/powershell/

何故AWS CLIを使うのか？(1/2)

コマンドラインインターフェースで迅速な作業が行えるため

ex. S3にファイルアップロード



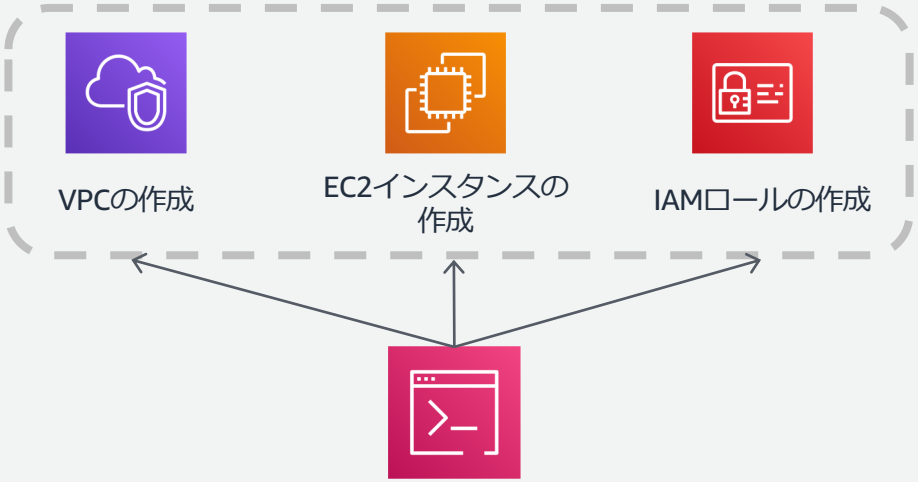

```
$ aws s3 cp test.txt s3://mybucket/test.txt
```

何故AWS CLIを使うのか？(2/2)

スクリプトによる作業自動化ができるため

ex. 開発環境の構築

VPC、EC2、 AWS Identity and Access Management (IAM) それぞれをAWS CLIから操作

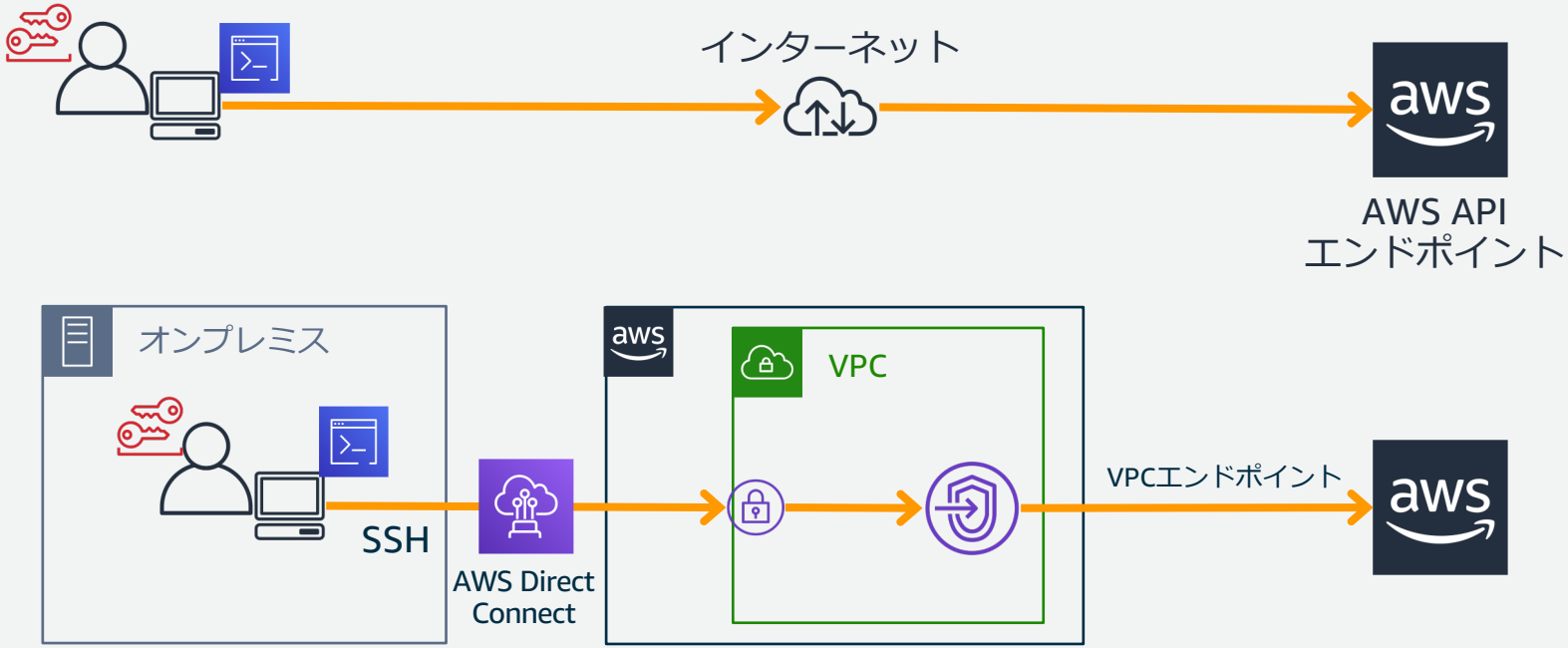


人間によるGUI操作はオペレーションミスが発生する可能性があるため、自動化することでインシデントを事前に防げます。

通信経路 (端末から操作する場合)

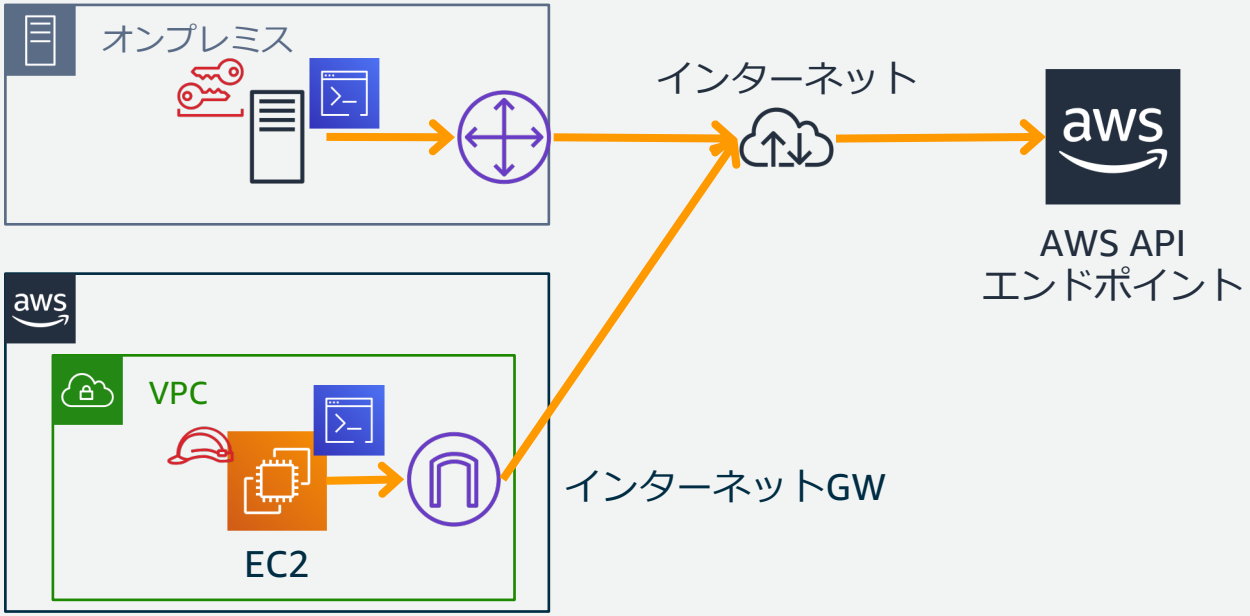
パターン1: インターネット経由でアクセスする

パターン2: 閉域でアクセスする



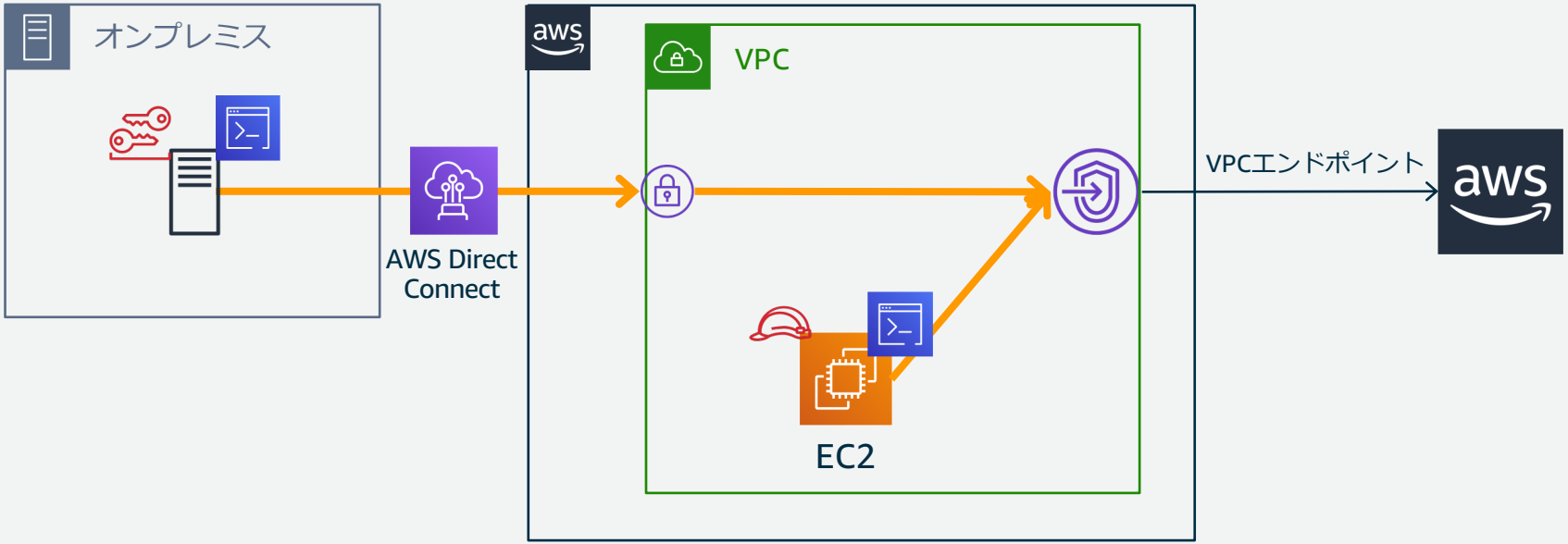
通信経路 (バッチ等でサーバから操作する場合)

パターン1: インターネット経由でアクセスする



通信経路 (バッチ等でサーバから操作する場合)

パターン2: 閉域でアクセスする



概要のまとめ

- AWS CLIはAWS環境を操作する手段のうちの1つ
- AWS CLIはAWS APIに追従しており、ほぼ全てのAWSのサービスの操作を網羅
- AWS CLIを利用することで、迅速な作業、作業自動化可能
- 通信経路として、インターネット経由、閉域網の両方を選択可能

アジェンダ

- 概要
- **セットアップ方法**
- 操作方法
- 設定
- 操作パターン集

セットアップ方法

aws

セットアップの流れ

1. IAMの設定
2. AWS CLIのインストール
3. 初期設定

各手順の詳細につきましては公式ドキュメントを参照してください。
http://docs.aws.amazon.com/ja_jp/cli/latest/userguide/cli-chap-getting-set-up.html

Step 1 - IAMの設定

1. IAMユーザ・IAMロールを用意する

PC、オンプレミスサーバーから利用

IAMユーザー
の作成

アクセスキー、
シークレットアクセス
キーの作成

EC2インスタンス上から利用

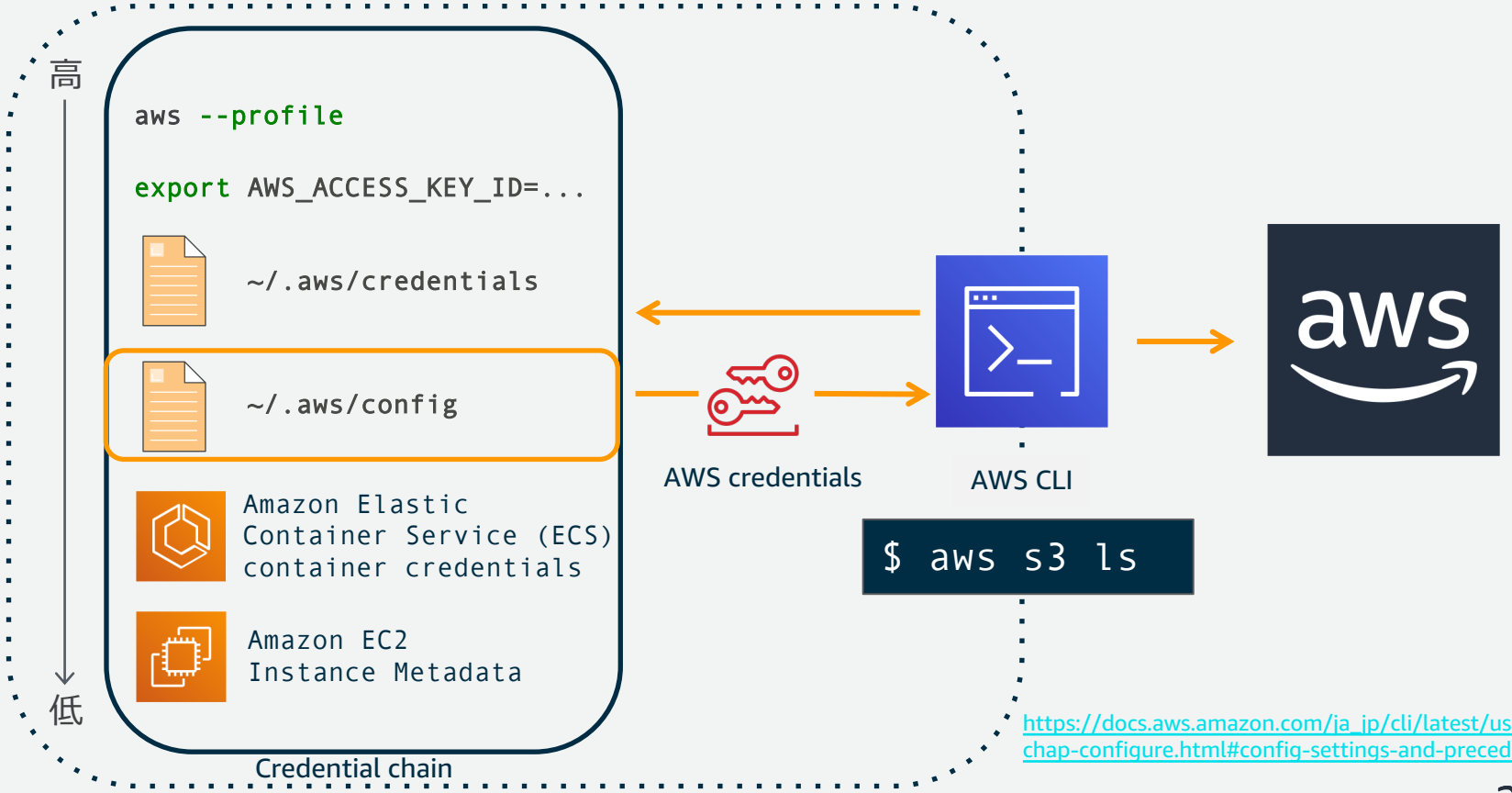
IAMロールの
作成

EC2インスタンスに
ロールの割当

いずれの方式にせよ
認証は必ず行われて
いるためセキュア

2. IAMユーザー、IAMロールにAWSのリソースを操作する権限をIAMポリシーで付与する

AWS CLIの認証の仕組み



https://docs.aws.amazon.com/ja_jp/cli/latest/userguide/cli-chap-configure.html#config-settings-and-precedence

Step 2 - AWS CLIのインストール

前提条件

- Python 2 バージョン 2.6.5+ または **Python 3 バージョン 3.3+**
- Windows、Linux, macOS, or Unix

Python2系のEOLが2020/1/1のためPython3系を推奨

セットアップ方法

各プラットフォーム毎 (Windows、Linux, macOS, Unix) にセットアップ方法が異なります。プラットフォーム毎に以下の方法から選択してください。

- pip(仮想環境なし)を利用したインストール
- pip(仮想環境あり)を利用したインストール
- インストーラを使用した インストール

http://docs.aws.amazon.com/ja_jp/cli/latest/userguide/installing.html

Step 3 - 初期設定

`aws configure` コマンドがAWS CLI のインストールをセットアップするための最も簡単な方法です。

```
$ aws configure
AWS Access Key ID [None]: xxx
AWS Secret Access Key [None]: xxx
Default region name [None]: ap-northeast-1
Default output format [None]: json
```

Step 1で作成したアクセスキー、シークレットアクセスキーをここで入力し、デフォルトリージョン、出力形式を設定します。(EC2のロールを割り当てた場合は2つのアクセスキーは不要)

ここで入力された機密性の高い認証情報 (アクセスキー、シークレットアクセスキー) 、機密性の低い設定オプション (リージョン、出力形式) をそれぞれ `~/.aws/credentials` 、`~/.aws/config` に区別して保存します。(Windowsであれば、 `$HOME/.aws/credentials` 、`$HOME/.aws/config`)

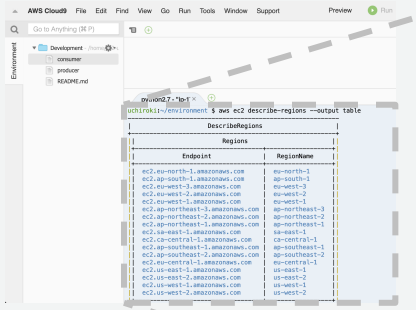
上記設定は最低限のものとなっております。その他設定につきましては後述します。

(参考) AWS CLIを簡単に実行する環境のご紹介

EC2 – Amazon Linuxインスタンスにインストール済み。AWS CLIを利用するにはロールの付与が必要です。

Cloud9 – クラウドの統合開発環境。こちらもAWS CLIがインストール済みです。ロールはデフォルトで付与されているものの、操作できないものがあるため必要に応じて権限を付与してください。

https://docs.aws.amazon.com/ja_jp/cloud9/latest/user-guide/auth-and-access-control.html#auth-and-access-control-temporary-managed-credentials



```
uchiroki:~/environment $ aws ec2 describe-regions --output table
```

DescribeRegions	
Regions	
Endpoint	RegionName
ec2.eu-north-1.amazonaws.com	eu-north-1
ec2.ap-south-1.amazonaws.com	ap-south-1
ec2.eu-west-3.amazonaws.com	eu-west-3
ec2.eu-west-2.amazonaws.com	eu-west-2
ec2.eu-west-1.amazonaws.com	eu-west-1
ec2.ap-northeast-3.amazonaws.com	ap-northeast-3
ec2.ap-northeast-2.amazonaws.com	ap-northeast-2
ec2.ap-northeast-1.amazonaws.com	ap-northeast-1
ec2.sa-east-1.amazonaws.com	sa-east-1
ec2.ca-central-1.amazonaws.com	ca-central-1
ec2.ap-southeast-1.amazonaws.com	ap-southeast-1
ec2.ap-southeast-2.amazonaws.com	ap-southeast-2
ec2.eu-central-1.amazonaws.com	eu-central-1
ec2.us-east-1.amazonaws.com	us-east-1
ec2.us-east-2.amazonaws.com	us-east-2
ec2.us-west-1.amazonaws.com	us-west-1
ec2.us-west-2.amazonaws.com	us-west-2

セットアップのまとめ

- AWS CLIは各種プラットフォームにインストール可能
- 初期設定については、 `aws configure` コマンドを利用すると簡単に設定可能

アジェンダ

- 概要
- セットアップ方法
- **操作方法**
- 設定
- 操作パターン集

操作方法

aws

AWS CLIの操作方法について

- 実行コマンドの形式
- 各種オプションについて
    - `--profile` オプション
    - `--region` オプション
    - `--output` オプション
    - `--query` オプション
- 各種サブコマンドについて
    - `help` サブコマンド
    - `wait` サブコマンド
- リターンコード
- CLIスケルトンの生成、実行
- コマンド補完

実行コマンドの形式

```
$ aws [options] <command> <subcommand> [parameters]
```

AWSコマンド全体で共通のパラメータ。 `--` から始まる。

ec2, s3 など主にサービス名（例外は configure, help）

サービスごとのコマンド

コマンド毎のパラメータ。
`--` から始まる。

例: 東京リージョンのEC2インスタンスを最大2つ表示する

```
$ aws --region ap-northeast-1 ec2 describe-instances --max-items 2
```

http://docs.aws.amazon.com/cli/latest/reference/

aws

--profile オプション

設定ファイル (プロファイル) を指定することで環境ごとの設定の切り替えができます。設定ファイルの詳細につきましては後述いたします。

```
$ aws --profile (profile name) <command> <subcommand> [parameters]
```

devを指定

~/.aws/credentials

```
[default]
aws_access_key_id=xxx
aws_secret_access_key=xxx
[profile dev]
aws_access_key_id=yyy
aws_secret_access_key=yyy
```

~/.aws/config

```
[default]
region=us-west-2
output=json
[profile dev]
region=ap-northeast-1
output=table
```

https://docs.aws.amazon.com/ja_jp/cli/latest/userguide/cli-configure-files.html

aws

--region オプション

コマンドを実行するリージョンを指定します。設定ファイルで指定されたリージョン設定より優先されます。

```
$ aws --region (region) <command> <subcommand> [parameters]
```

リージョン	リージョン名
ap-northeast-1	アジアパシフィック（東京）リージョン
ap-southeast-1	アジアパシフィック（シンガポール）リージョン
ap-southeast-2	アジアパシフィック（シドニー）リージョン
eu-central-1	欧州（フランクフルト）リージョン
eu-west-1	欧州（アイルランド）リージョン
…	…

https://docs.aws.amazon.com/ja_jp/general/latest/gr/rande.html

--output オプション

コマンドの出力形式を変更できます。JSON、Text、Tableのいずれかを指定できます。デフォルトはJSON形式です。

```
$ aws --output (output type) <command> <subcommand> [parameters]
```

json

```
{
  "Places": [
    {
      “City”: “Seattle”,
      "State": "WA"
    },
    {
      "City": ”Las Vegas",
      "State": "NV"
    }
  ]
}
```

text

```
PLACES  Seattle    WA
PLACES  Las Vegas  NV
```

table

```
---------------------------
|     SomeOperationName   |
+-------------------------+
||          Places       ||
|+------------+---------+|
|| City       | State   ||
|+------------+---------+|
|| Seattle    | WA      ||
|| Las Vegas  | NV      ||
|+------------+---------+|
```

--query オプション (1/7)

AWS CLIで出力されるJSON形式の標準出力をJMESPathの仕様に準拠してフィルタリングできるオプションです。 API のエンドポイント側ではなく、クライアント側で適用されるフィルターです。

```
$ aws --query (String) <command> <subcommand> [parameters]
```

例: EC2のEBSボリュームの1つ目を表示

```
$ aws ec2 describe-volume
```

```
{
  "Volumes": [
    {
      "Attachments": [
        {
           "AttachTime": "2019-04-
18T10:42:03.000Z",
           …
        },
        {
           …
        }
…
```

```
$ aws ec2 describe-volumes --query 'Volumes[0]'
```

```
{
  "Volumes": [
    {
      "Attachments": [
        {
           "AttachTime": "2019-04-18T10:42:03.000Z",
           …
        }
      ]
    }
  ]
}
```

--query オプション (2/7)

(--queryなし)

```
{
 "Users": [
    {
     "Arn": "arn:aws:iam::XXXX:user/james",
     "UserId": "userid",
     "CreateDate": "2013-03-09T23:36:32Z",
     "Path": "/",
     "UserName": "james"
    }
  ]
}
```

マルチセレクト

```
--query Users[0].[UserName,Path,UserId]
```

```
[
  "james",
  "/",
  "userid"
]
```

--query オプション (3/7)

```
$ aws ec2 describe-instances
```

```
{
    "Reservations": [
        {
          "Groups": [],
          "Instances": [
            {
                "InstanceId": "i-XXX",
                "InstanceType": "c3.2xlarge",
...
```

階層の絞り込み

```
$ aws ec2 describe-instances --query
‘Reservations[].Instances[].[InstanceId, InstanceType]’
```

表示項目の絞り込み

```
[
    [
        "i-XXX",
        "c3.2xlarge"
    ],
    [
        "i-YYY",
        "t2.medium"
    ]
]
```


aws

--query オプション (4/7)

表示項目の追加

```
$ aws ec2 describe-instances --query
‘Reservations[].Instances[].[InstanceI
d, InstanceType]’
```

```
[
    [
        "i-XXX",
        "c5.xlarge"
    ],
    [
        "i-YYY",
        "t2.medium"
    ]
]
```

```
$ aws ec2 describe-instances --query
'Reservations[].Instances[].{id:
InstanceId, type: InstanceType}'
```

```
[
    [
        "id": "i-XXX",
        "type": "c5.xlarge"
    ],
    [
        "id”: "i-YYY",
        "type": "t2.medium"
    ]
]
```

--query オプション (5/7)

```
$ aws ec2 describe-instances --query
‘Reservations[].Instances[].[InstanceId, InstanceType]’
```

```
[
    [
        "i-XXX",
        "t2.micro"
    ],
    [
        "i-YYY",
        "c5.xlarge"
    ]
]
```

完全一致でアイテムの絞り込み

```
$ aws ec2 describe-instances --query
'Reservations[].Instances[?InstanceType==`t2.micro`].[InstanceId,
InstanceType][]'
```

```
[
    [
        "id": "i-XXX",
        "type": "t2.micro"
    ]
]
```

部分一致でアイテムの絞り込み

```
$ aws ec2 describe-instances --query
'Reservations[].Instances[?contains(InstanceType,`t2`)!=`true`].[InstanceId,
InstanceType][]'
```

```
[
    [
        "id": "i-YYY",
        "type": "c5.xlarge"
    ]
]
```

--query オプション (6/7)

```
$ aws ec2 describe-instances --query
'Reservations[].Instances[].[InstanceI
d, InstanceType, State.Name]'
```

```
[
    [
        "i-XXX",
        ”t2.micro”
        "stopped"
    ],
    [
        "i-YYY",
        "t2.micro"
        "running"
    ] ,
    [
        "i-ZZZ",
        "t2.micro"
        "running"
    ]
]
```

AND条件で絞り込み

```
$ aws ec2 describe-instances --query
'Reservations[].Instances[?State.Name!
=`stopped`][] |
[?InstanceType==`t2.micro`].[InstanceI
d, InstanceType, State.Name]'
```

```
[
    [
        "i-XXX",
        "t2.micro"
        "stopped”
    ]
]
```

--query オプション (7/7)

```
$ aws ec2 describe-instances --query
'Reservations[].Instances[].[InstanceI
d, InstanceType, State.Name]'
```

```
[
    [
        "i-XXX",
        ”t2.micro”
        "stopped"
    ],
    [
        "i-YYY",
        "t2.micro"
        "running"
    ] ,
    [
        "i-ZZZ",
        "t2.micro"
        "running"
    ]
]
```

絞り込みした配列の数を表示

```
$ aws ec2 describe-instances --query
'length(Reservations[].Instances[?Inst
anceType==`t2.micro`][InstanceId][])'
```

```
3
```

JMESPathについて (1/3)

JMESPathとは、JSON用のクエリー言語です。パース、フィルター、整形ができます。

クエリー種別	インプット	クエリー	結果
キー	`{"a": "foo", "b": "bar"}`	`a`	`foo`

キー(階層)	`{"a": {`
	`"b": {`
	`"c":"value"`
	`}`
	`}`
	`}`
		`a.b.c`	`value`

インデックス	`["a", "b", "c", "d", "e", "f"]`	`[1]`	`b`
スライス	`[0, 1, 2, 3, 4, 5]`	`[0:4]`	`[0, 1, 2, 3]`
スライス(スキップ)	`[0, 1, 2, 3, 4, 5]`	`[::2]`	`[0, 2, 4]`
パイプ	`[0, 1, 2, 3, 4, 5]`	`[::2] | [1]`	`2`

http://jmespath.org

aws

JMESPathについて (2/3)

クエリー種別	インプット	クエリー	結果
==	{"machines": [{"name": "a", "state": "running" }, {"name": "b", "state": "stopped" }]}	machines[?state== 'running'].name	["a"]
!=	{"machines": [{"name": "a", "state": "running" }, {"name": "b", "state": "stopped" }]}	machines[?state!= 'running'].name	["b"]

マルチセレクト	{"machines": [{
	"name": "a",
	"state": {"name": "running"}
	},{
	"name": "b",
	"state": {"name": "stopped"}
	}]}
		machines[].[name,
		state.name]
			[
			["a","running"],
			["b","stopped"]
			]

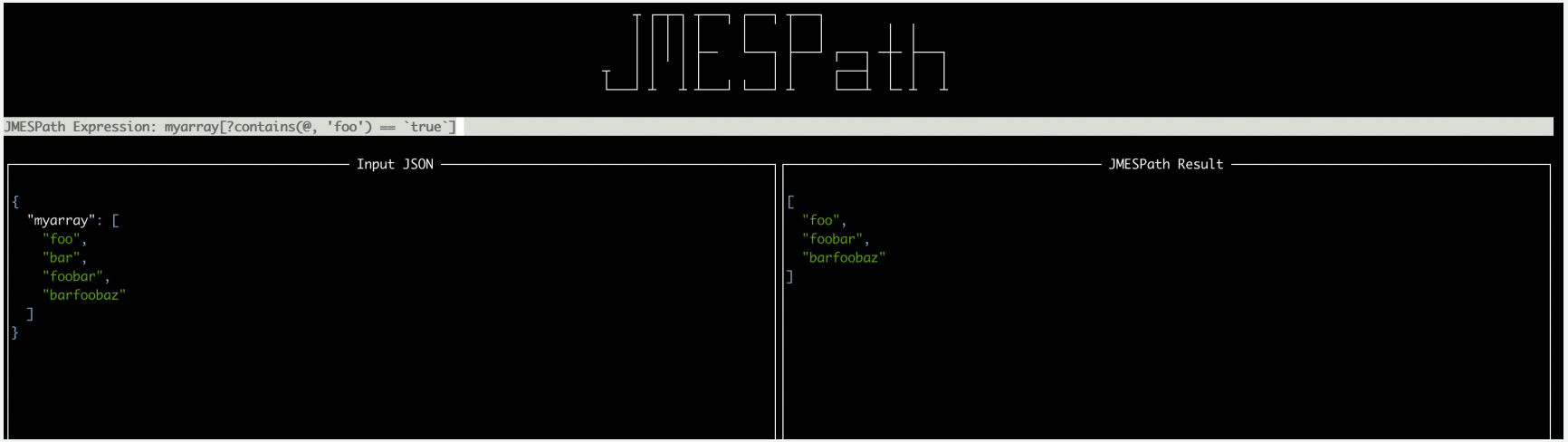
aws

JMESPathについて (3/3)

JMESPathファンクションについて

クエリー種別	インプット	クエリー	結果
length	`{ "people": [ {"name": "a" }, {"name": "b" }, {"name": "c" }]}`	`length(people)`	`3`
max_by	`{"people": [ {"name": "a", "age": 30}, {"name": "b", "age": 50}, {"name": "c", "age": 40}]}`	`max_by(people, &age).name`	`b`
contains	`{"myarray": [ "foo", "bar", "foobar", "barfoobaz"]}`	`myarray[?contains(@, 'foo') == `true`]`	`[ "foo", "foobar", "barfoobaz" ]`

(参考) JMESPathの動作確認ツール



JMESPath Terminal

https://github.com/jmespath/jmespath.terminal

```
$ echo '{"myarray": ["foo", "bar", "foobar", "barfoobaz"]}' | jpterm
$ aws ec2 describe-instances | jpterm
```

AWS CLIの出力でも JMESPathの確認ができます

処理速度向上について

`--filter`

- 実行されるのは**サーバーサイド**
- 実行できるサブコマンドは限られるので注意。 (list-* や describe-* で使えるケースが多い)
- 1000以上の大きいデータセットを取得するようなユースケースで使うと効果的。
- 上記のような大きいデータセットで `--page-size` オプションもあわせて使うとさらに効果的。
- 出力されるJSON構造を意識せずに記述できる。

`--query`

- 実行されるのは**クライアントサイド**
- すべてのサブコマンドにて実行可能です。
- JMESPathを利用したクエリで絞り込みできます。

```
$ aws ec2 describe-instances --filters
Name=instance-type,Values=c5.xlarge
```

```
$ aws ec2 describe-instances --query
"Reservations[].Instances[?InstanceType ==
'c5.xlarge']"
```

`help` サブコマンド

各コマンドの詳細を確認できるサブコマンド。**AWS CLIを使う際に必須と言えるほどよく使うサブコマンド**ですので是非使い方を覚えてください。

```
$ aws [options] <command> help [parameters]
```

例1: EC2のコマンド確認

```
$ aws ec2 help
```

例2: EC2の `describe-instances` のコマンド確認

```
$ aws ec2 describe-instances help
```

https://docs.aws.amazon.com/ja_jp/cli/latest/userguide/cli-usage-help.html

aws

wait サブコマンド

特定の条件が満たされるまでAWS CLIを待機させられます。このサブコマンドを利用することで、イベントが完了されるまで待機できます。試行回数が一定回数を超えてタイムアウトが発生した場合、255のリターンコードを返します。

```
$ aws [options] <command> wait [parameters]
```

例1: 特定のインスタンスが起動されるまで待機させる

```
$ aws ec2 wait instance-running --instance-ids $instance_id
```

例2: クラスター ID が x-123456789asdfg のクラスターが実行されるまで待機させる例

```
$ aws emr wait cluster-running --cluster-id xxxxx
```

http://docs.aws.amazon.com/cli/latest/reference/ec2/wait/index.html

aws

リターンコード

リターンコード	説明
0	エラー無しでコマンド実行が成功。
1	S3コマンドに限定され、実行されたコマンドに対して少なくとも1つ以上のS3転送が失敗。
2	パラメータ不足などのエラー。またはs3コマンドで一部ファイルのみ転送失敗。
130	SIGINT (Ctrl + C)をAWS CLIコマンドが受け取り中断した。
255	コマンド実行失敗。CLIまたはサービス側のエラー。

上記リターンコードを確認する方法は以下の通りです。

```
$ aws ec2 describe-instances
$ echo $?
0
```

なお、`--debug` オプションをつけるとコマンド実行の詳細を確認でき、エラー時に活躍します。
リターンコードについて公式ドキュメントでも確認できますが、`aws help return-codes` でも参照できます。
https://docs.aws.amazon.com/ja_jp/cli/latest/userguide/cli-usage-returncodes.html

CLIスケルトンの生成、実行

特定のAWS CLIコマンドのすべてのパラメータを含んだファイルを生成できます。

1. `--generate-cli-skeleton` を指定してコマンドを実行し、ファイルを生成

```
$ aws ec2 run-instances --generate-cli-skeleton > ec2runinst.json
```

2. ファイルを開き、不要なパラメータの削除、パラメータ修正

```
{
    "DryRun": false,
    "ImageId": "ami-dfc39aef",
    "KeyName": "mykey",
    "SecurityGroups": [
        "my-sg"
    ],
    "InstanceType": "t2.micro",
    "Monitoring": {
        "Enabled": true
    }
}
```

3. `--cli-input-json` でファイルを指定して実行

```
$ aws ec2 run-instances --cli-input-json file://ec2runinst.json
```

https://docs.aws.amazon.com/ja_jp/cli/latest/userguide/cli-usage-skeleton.html

コマンド補完

`aws` コマンドを入力後、 [Tab] キーを押す度に補完が行われます。Amazon Linuxではコマンド補完が自動的に設定されていますが、ほとんどのシステムではこの機能が設定されていないため、手動で設定する必要があります。

1. シェルを識別する
2. AWSコンプリータを見つける
3. コンプリータのフォルダをパスに追加する
4. コマンド補完を有効にする
5. コマンド補完のテスト

```
$ aws s[Tab]
s3      ses     sqs                 sts       swf
s3ap    sns     storagegateway      support
```

https://docs.aws.amazon.com/ja_jp/cli/latest/userguide/cli-configure-completion.html

操作方法のまとめ

- AWS CLIの実行コマンドの形式のご紹介
- 各種オプション、サブコマンドについて
    - `--profile` オプション
    - `--region` オプション
    - `--output` オプション
    - `--query` オプション
    - `help` サブコマンド
    - `wait` サブコマンド
- リターンコード、CLIスケルトン、コマンド補完について

アジェンダ

- 概要
- セットアップ方法
- 操作方法
- **設定**
- 操作パターン集

設定

aws

よく使うAWS CLIの設定

- 複数プロファイルの設定、実行方法
    - 複数プロファイルの作成方法、切り替え方法
    - 他アカウントのリソースへアクセス
    - プロファイルの設定項目について
    - MFA (多要素認証) を使ったアクセス
- S3コマンドの設定
- HTTPプロキシの設定

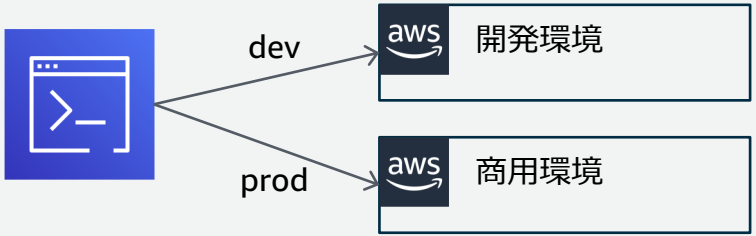
複数プロファイルの設定、実行方法 (1/5)

開発環境(dev)、商用環境(prod)のように環境ごとに設定内容を変更して保存可能。

```
$ aws --profile prod configure
AWS Access Key ID [None]: xxx
AWS Secret Access Key [None]: xxx
Default region name [None]: ap-northeast-1
Default output format [None]: table
```

また、実行時にプロファイルを指定して実行可能。

```
$ aws --profile prod ec2 describe-instances
```



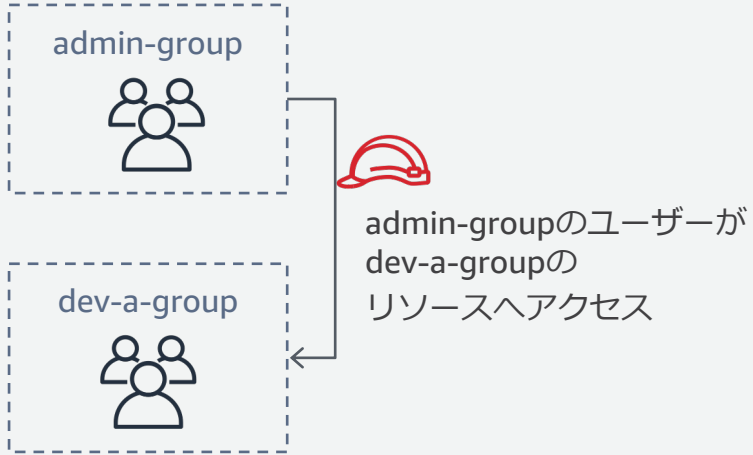
複数プロファイルの設定、実行方法 (2/5)

他アカウントのリソースへアクセスする

1. IAMでロールの作成
2. ロールの新しいプロファイルを `~/.aws/config` ファイルに追加

~/.aws/config

```
[profile admin-group]
aws_access_key_id = XXX…
aws_secret_access_key = ABC...

[profile dev-a-group]
source_profile = admin-group
role_arn = arn:...:role/admin-team-account
```



3. プロファイルを切り替えてAWS CLIを実行

```
$ aws --profile dev-a-group s3 ls
```

https://docs.aws.amazon.com/ja_jp/IAM/latest/UserGuide/id_roles_use_switch-role-cli.html
https://docs.aws.amazon.com/ja_jp/cli/latest/userguide/cli-configure-role.html

複数プロファイルの設定、実行方法 (3/5)

プロファイルの設定項目について

~/.aws/config

```
[default]
aws_access_key_id = XXX…
aws_secret_access_key = ABC...

[profile admin-group]
credential_process = …

[profile dev-a-group]
source_profile = default
role_arn = arn:...:role/admin-team-account

[profile mfa-access]
source_profile = default
role_arn = arn:...:role/mfa-access-role
mfa_serial = arn:...:mfa/mfa-of-user
```

[ ] で囲われているものがプロファイル名
[default]はデフォルトとして使われる特別なプロファイル名

外部認証との連携設定 (次スライドにて説明)

指定したプロファイルに切り替えた上で、認証、設定を引き継ぐ。

Assume RoleをするためのARN指定

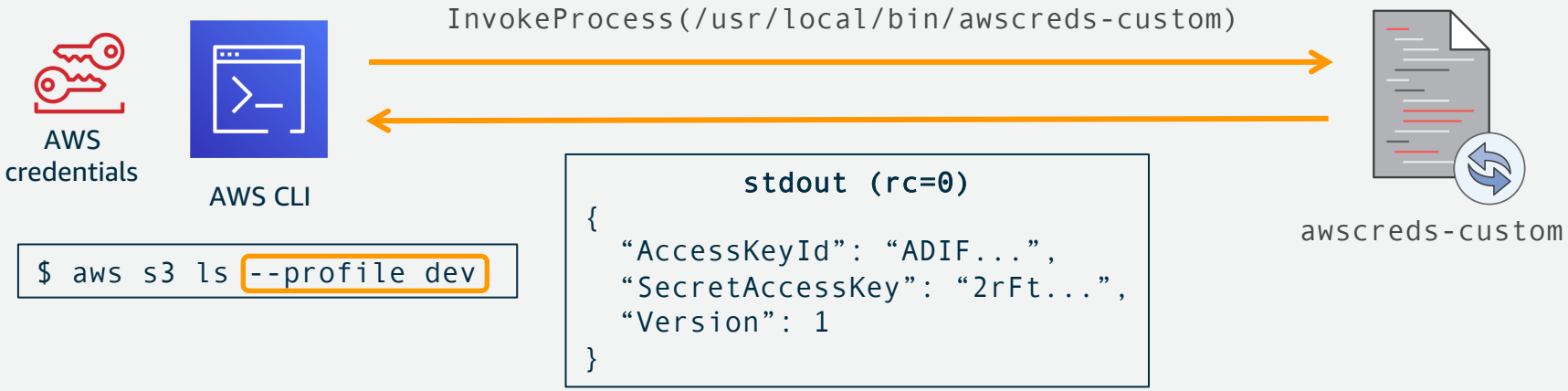
Assume Roleをする時に使われる多要素認証(MFA)デバイスの指定。セキュリティをより強固にしたい場合に用いられます。

※上記設定はあくまでサンプルのため、それぞれの項目で意味のある値ではないためご注意ください。
https://docs.aws.amazon.com/ja_jp/cli/latest/userguide/cli-configure-role.html

複数プロファイルの設定、実行方法 (4/5)

```
[default]
aws_access_key_id = AKIDF...
aws_secret_access_key = 1rHl...

[profile dev]
credential_process = /usr/local/bin/awscreds-custom
```

~/.aws/config



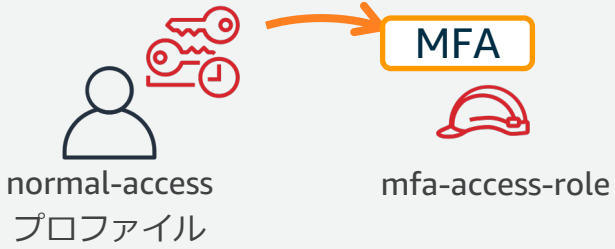
https://docs.aws.amazon.com/ja_jp/cli/latest/userguide/cli-configure-sourcing-external.html
※本説明のデモについては 「re:Invent 2017: AWS CLI: 2017 and Beyond」 24分あたりをご覧ください。

複数プロファイルの設定、実行方法 (5/5)

MFA (多要素認証) を使ったアクセス

mfa-access-role の 信頼関係

```
{"Version": "2012-10-17",
 "Statement": [ {
    "Effect": "Allow",
     "Principal": {"AWS":  {"arn:aws:iam::...:user/normal-user"}},
     "Action":       "sts:AssumeRole",
     "Condition": {"Bool": {"aws:MultiFactorAuthPresent": "true"}}
}]}
```



ロール取得時にMFAコードが必要

~/.aws/config

```
[profile normal-access]
aws_access_key_id = AKIDF...
aws_secret_access_key = 1rHl...

[profile mfa-access]
source_profile = normal-access
role_arn = arn:...:role/mfa-access-role
mfa_serial = arn:...:mfa/mfa-of-user
```

```
$ aws --profile normal-access s3 ls s3://public-bkt/
...
$ aws --profile mfa-access s3 ls s3://restrict-bkt/
Enter MFA code for arn:...:mfa/mfa-of-user
...
```

https://aws.amazon.com/jp/premiumsupport/knowledge-center/authenticate-mfa-cli/

S3コマンドの設定

S3のアップロードの設定が以下の通りにできます。

~/.aws/config

```
[default]
region=ap-northeast-1
output=json
s3 =
  max_concurrent_requests = 20
  max_queue_size = 10000
  max_bandwidth = 50MB/s
  multipart_threshold = 64MB
  multipart_chunksize = 16MB
  use_accelerate_endpoint = true
```

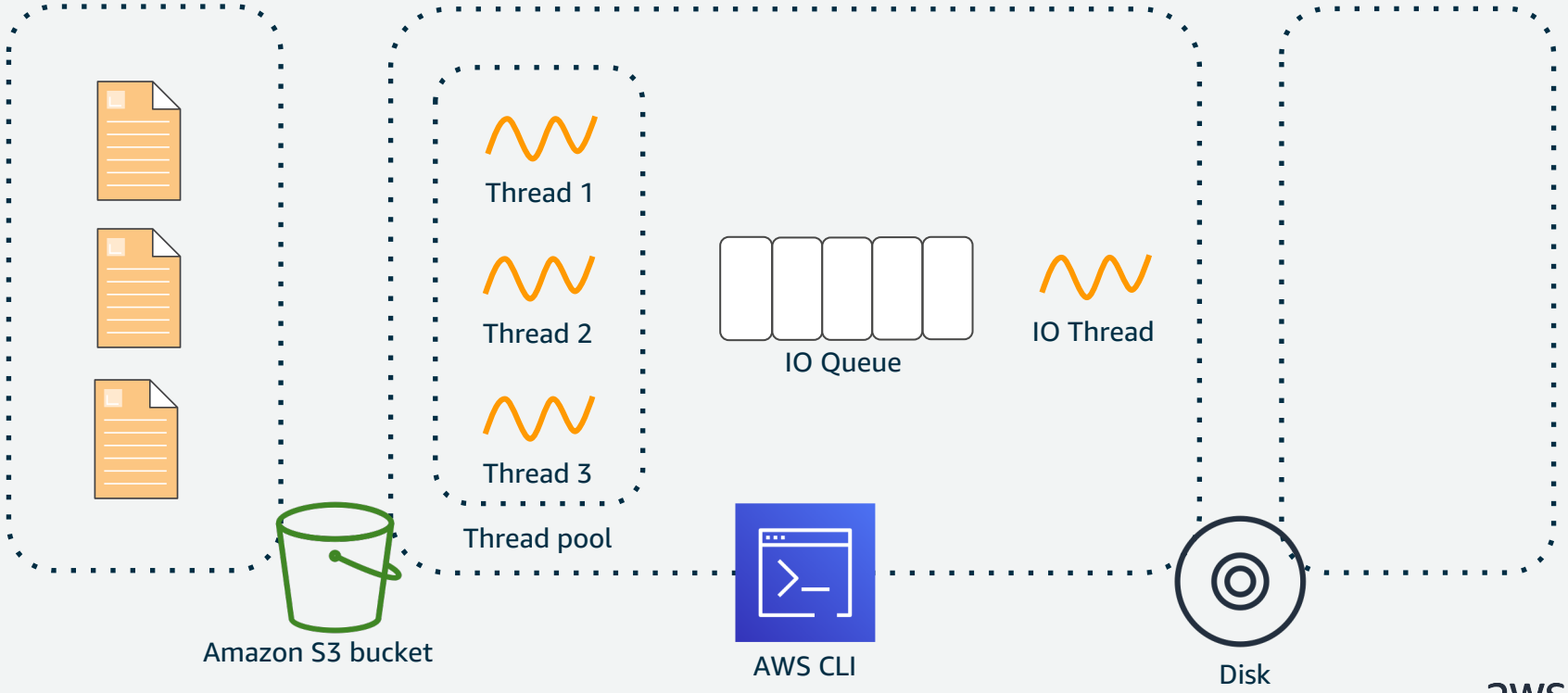
AWS CLIのスループットを検討する場合に変更するもの

- 最大同時リクエスト数
- タスクキューの最大数
- S3アップロード時に使う最大の帯域幅
- マルチパートに切り替えるファイルサイズのしきい値
- マルチパートのファイル1つあたりのサイズ
- S3 Accelerate endpointの使用可否

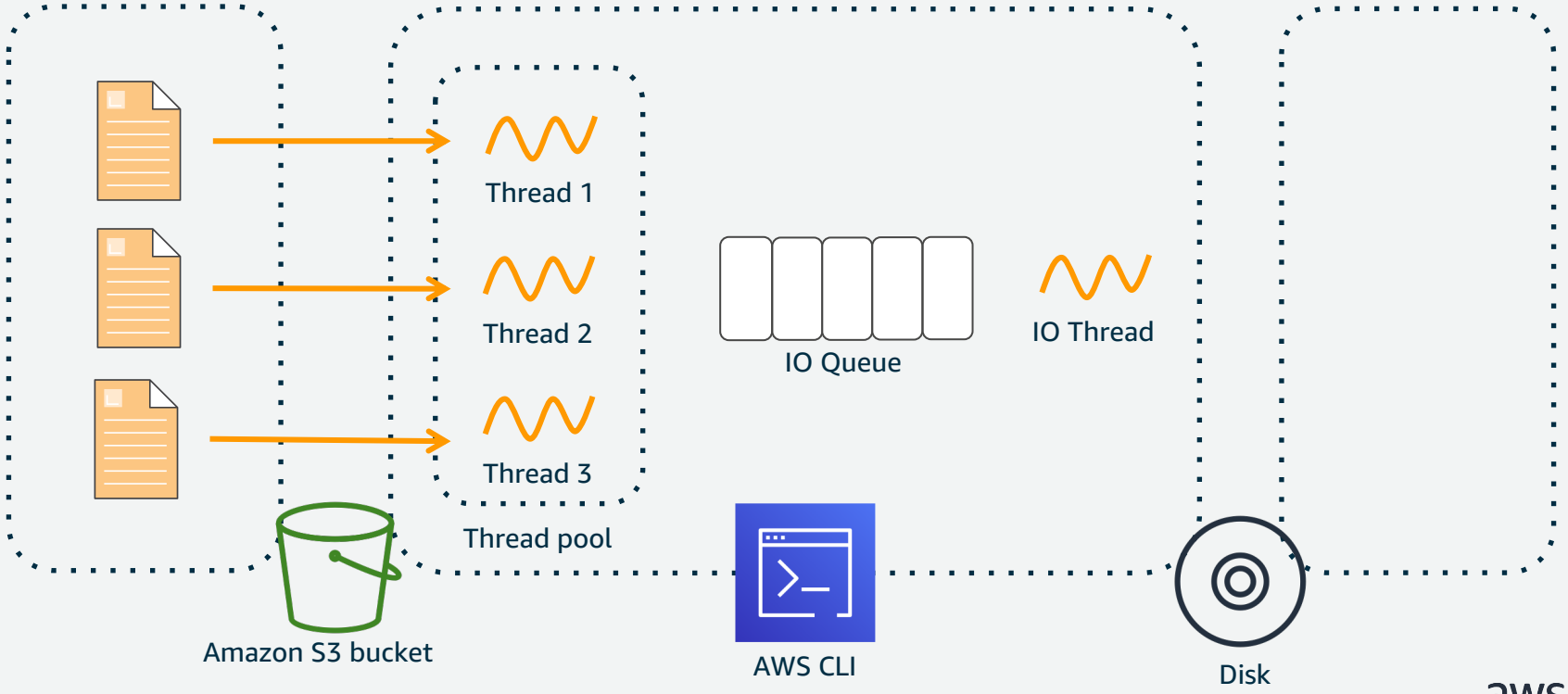
大きなファイルを扱う場合に変更するもの

※各項目のデフォルト値、詳細については公式ドキュメントを参照してください。
`aws help s3-config` , `aws help s3-faq` でも確認できます。
https://docs.aws.amazon.com/cli/latest/topic/s3-config.html

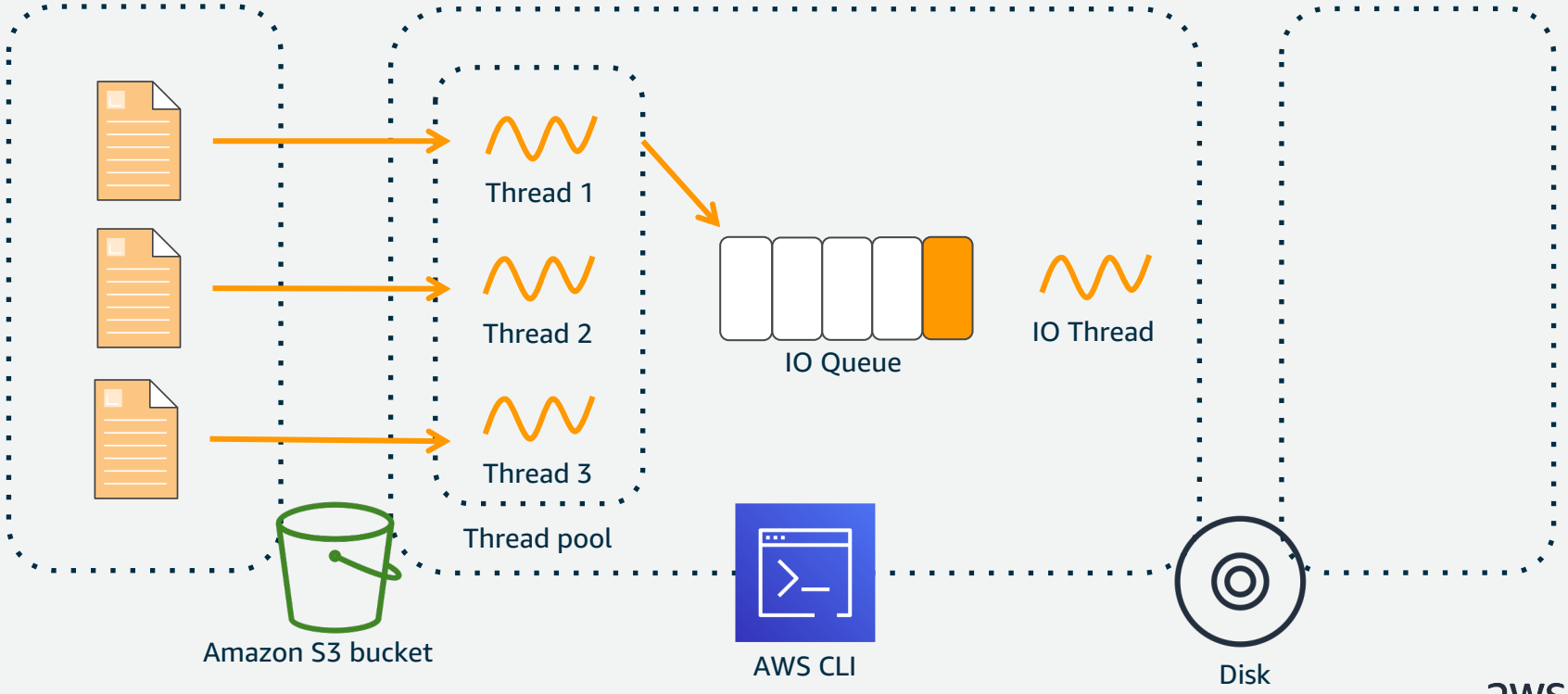
aws s3 cp の仕組み



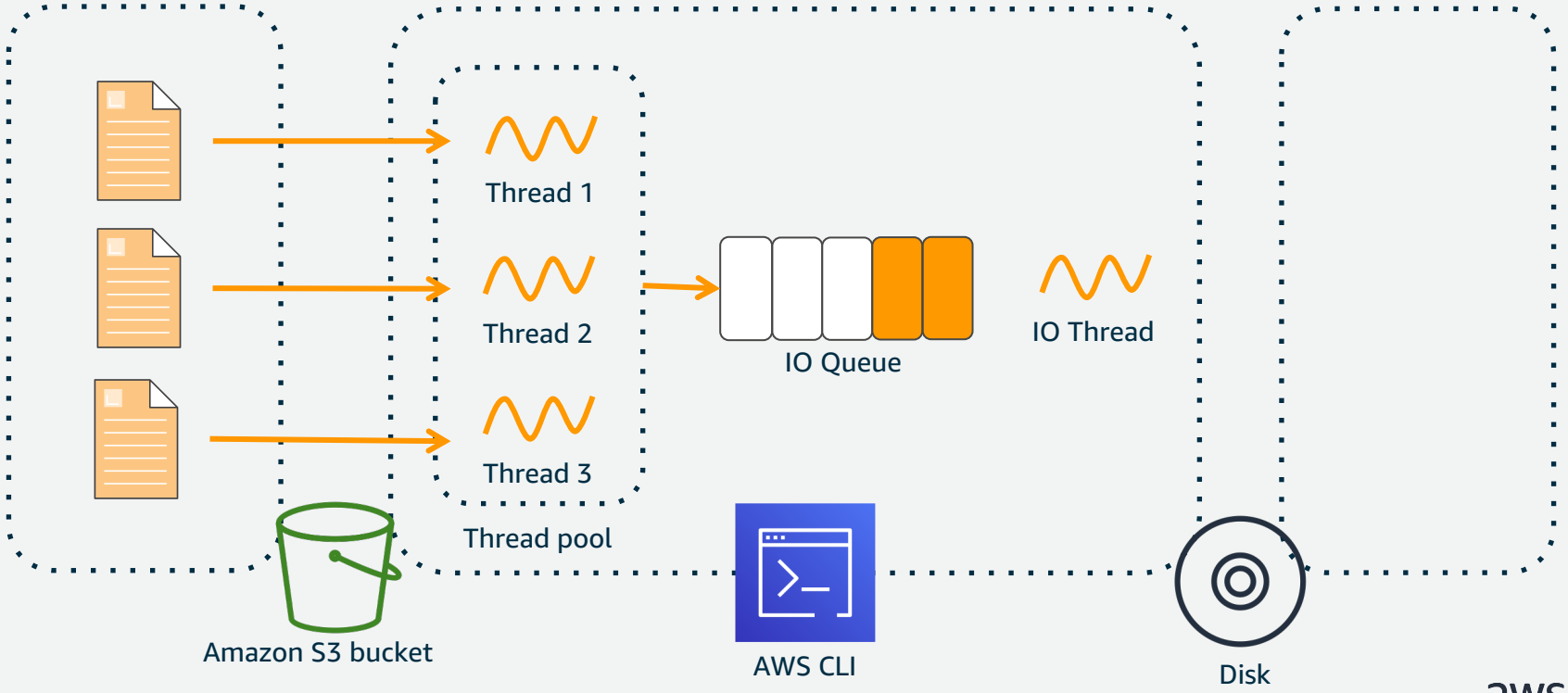
aws s3 cp の仕組み



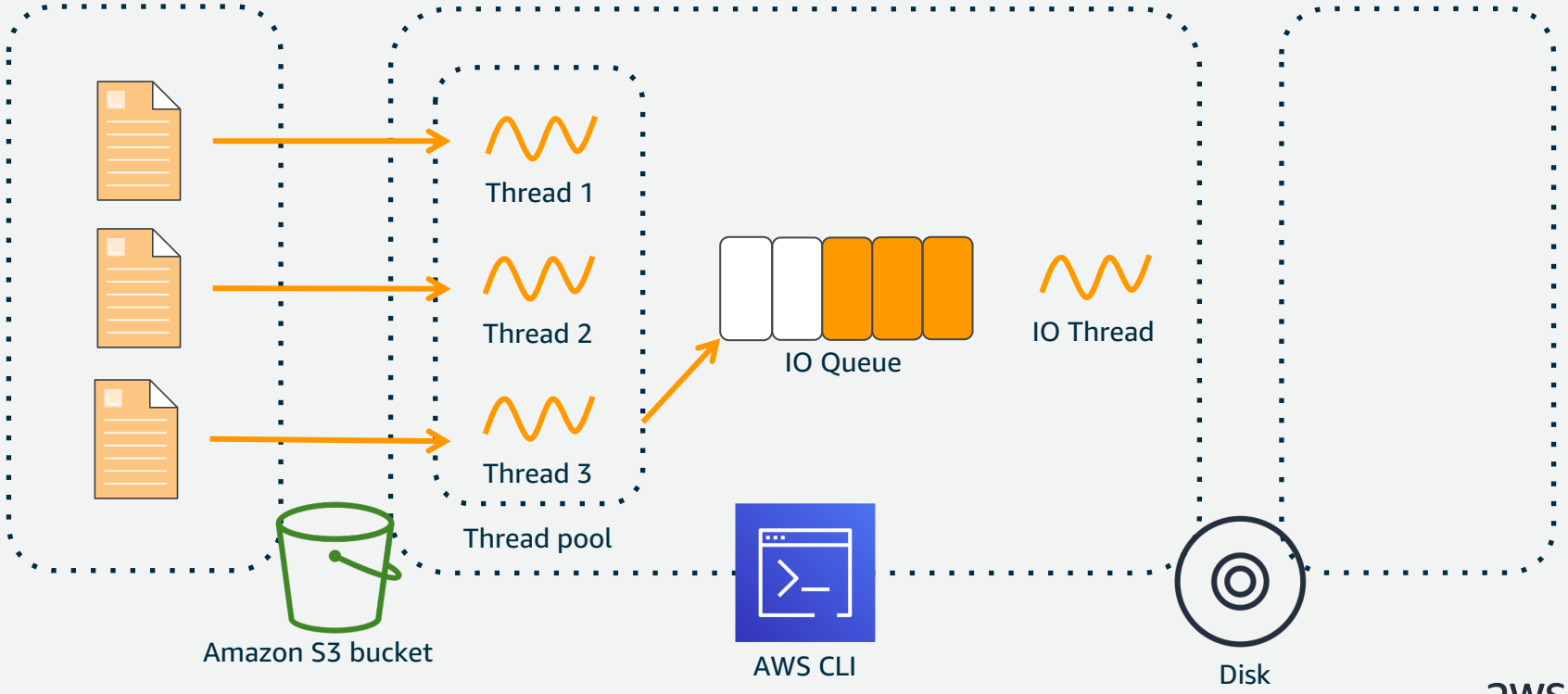
aws s3 cp の仕組み



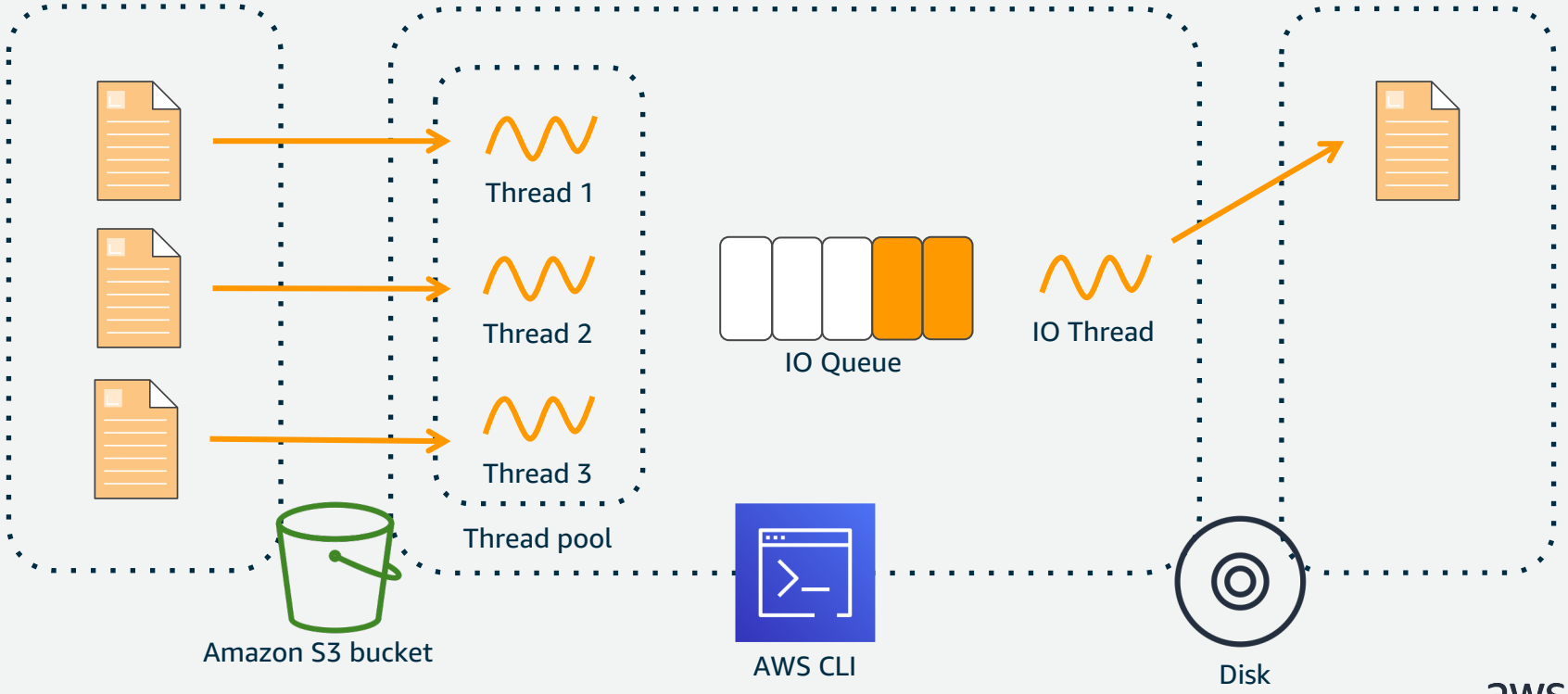
aws s3 cp の仕組み



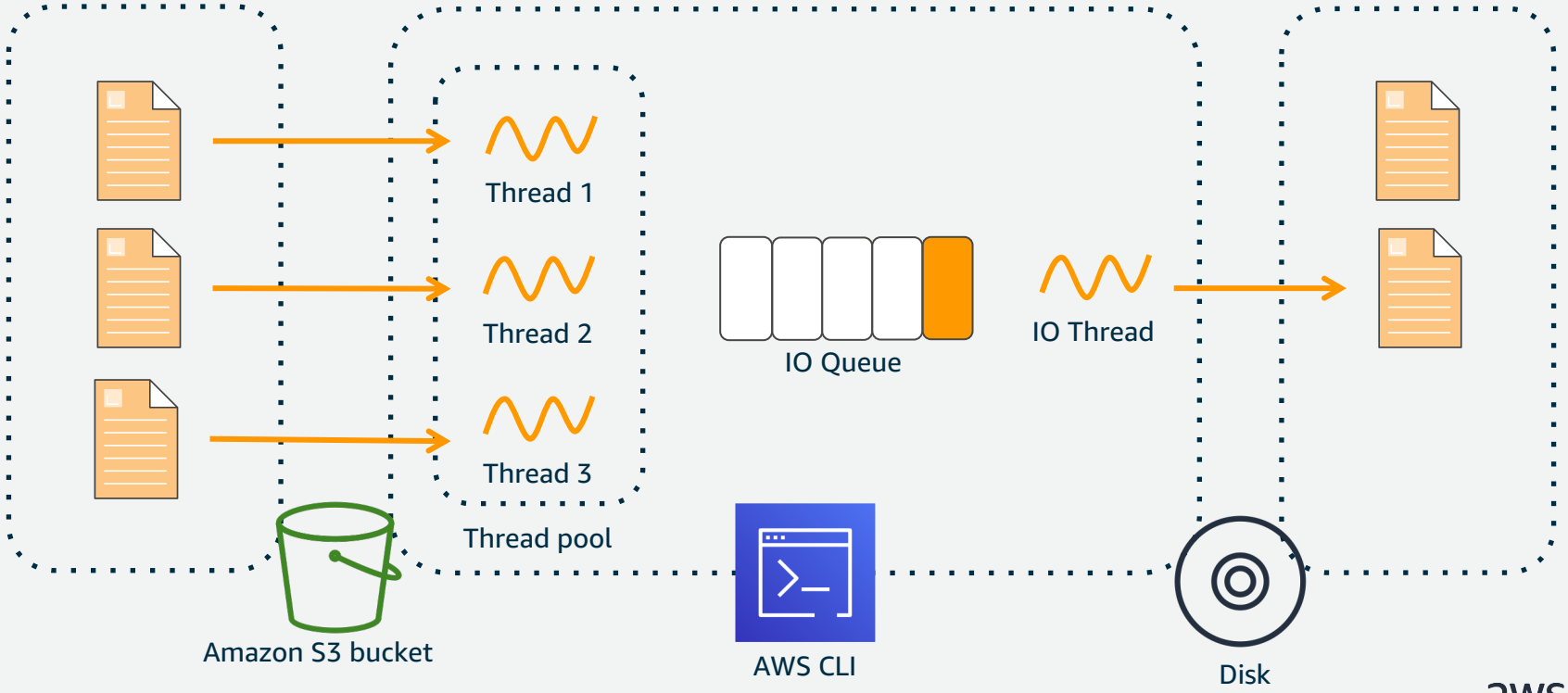
aws s3 cp の仕組み



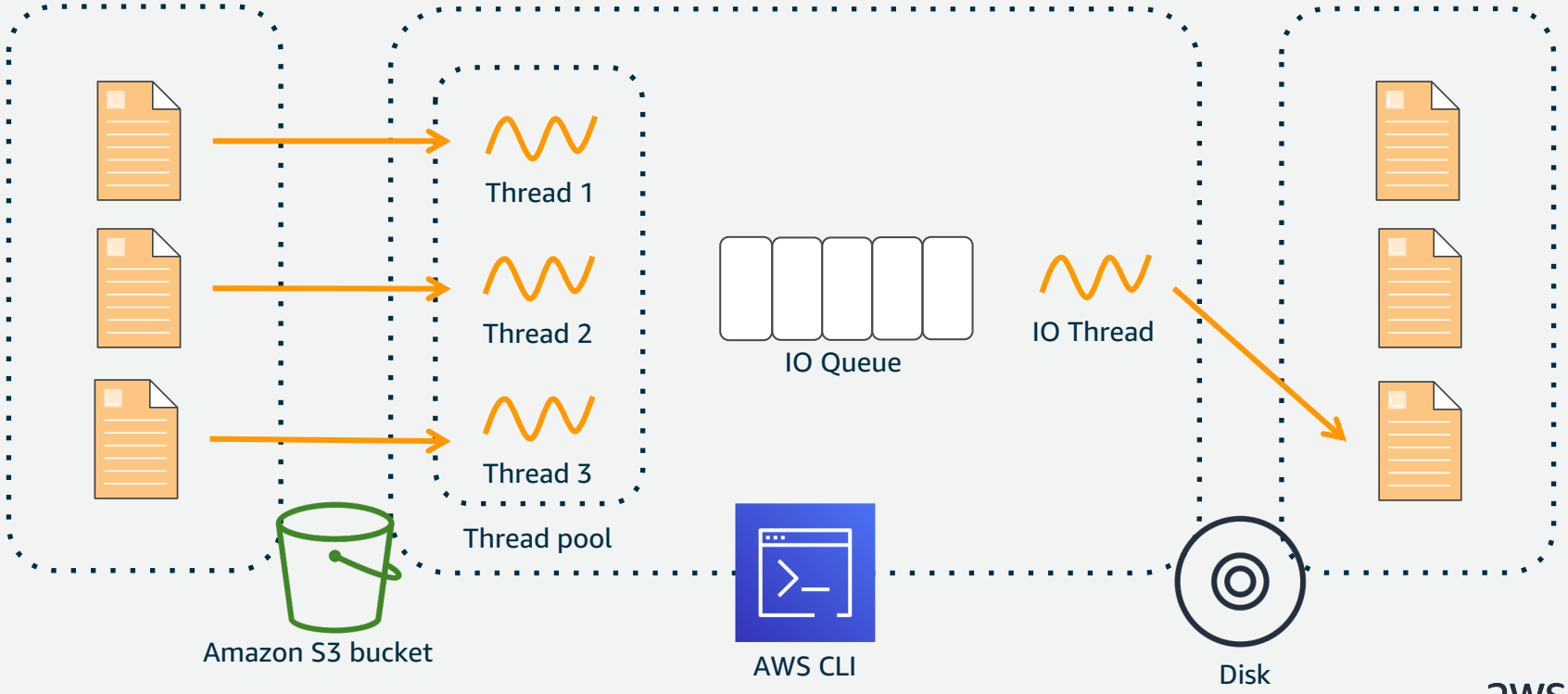
aws s3 cp の仕組み



aws s3 cp の仕組み

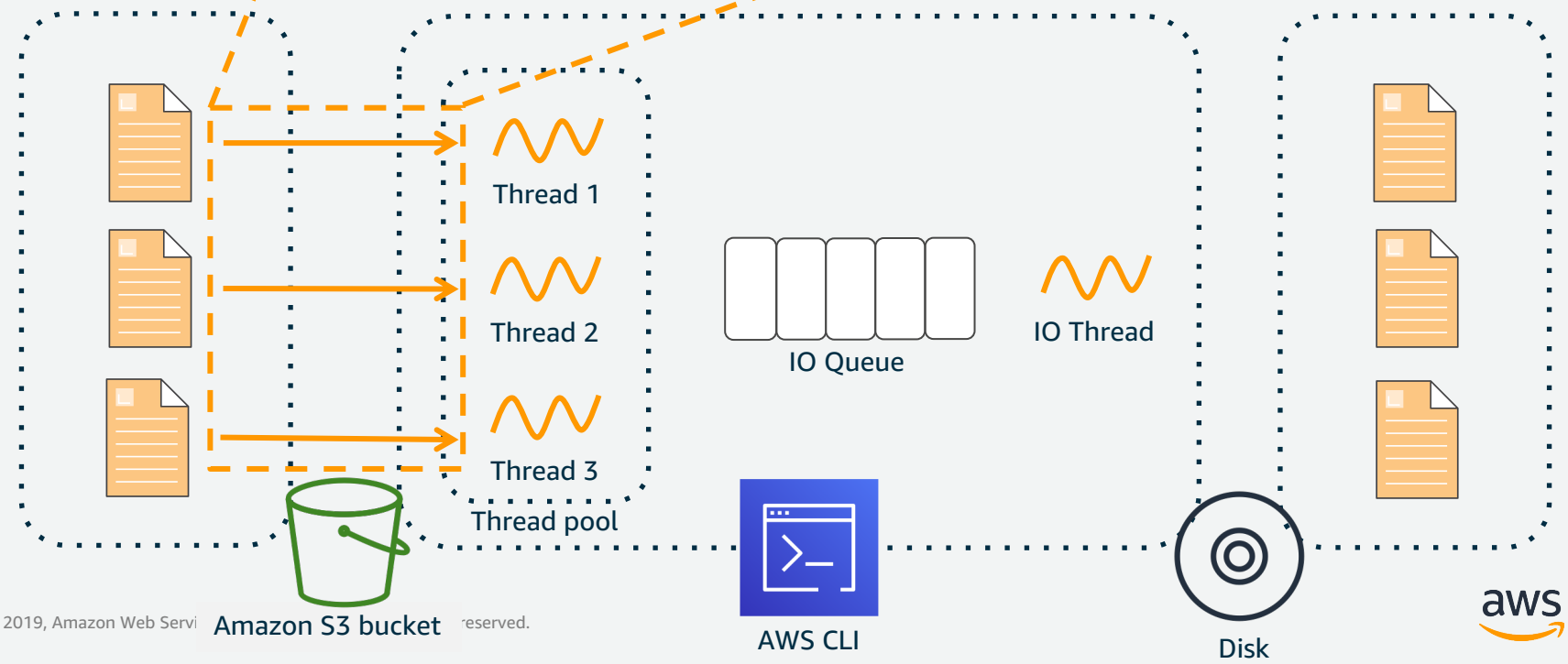


aws s3 cp の仕組み



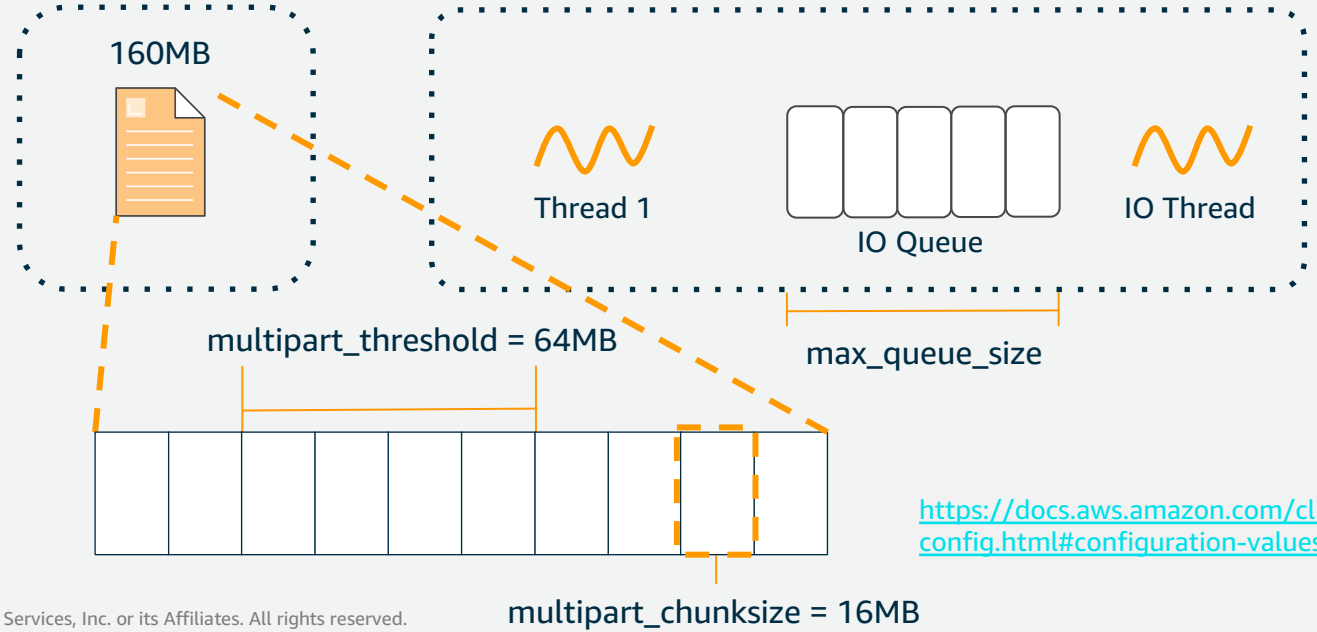
aws s3 cp の仕組み

```
[default]
region = ap-northeast-1
s3 =
    max_concurrent_requests = 3
    max_bandwidth = 50MB/s
```



aws s3 cp の仕組み

```
[default]
region = ap-northeast-1
s3 =
  max_queue_size = 10000
  multipart_threshold = 64MB
  multipart_chunksize = 16MB
```



https://docs.aws.amazon.com/cli/latest/topic/s3-config.html#configuration-values

AWS CLIのエイリアス作成方法

`aws whoami` のようなエイリアスコマンドを設定できます。設定ファイルは、`~/.aws/cli/alias` に配置します。

```
$ git clone https://github.com/awslabs/awscli-aliases.git
$ mkdir -p ~/.aws/cli
$ cp awscli-aliases/alias ~/.aws/cli/alias
```

`~/.aws/cli/alias`

```
[toplevel]
whoami = sts get-caller-identity
…
```

```
$ aws whoami
```

```
{
    "Account": ”123456789012”,
    "UserId": ”sample”,
    "Arn": "arn:aws:iam::123456789012:user/sample"
}
```

https://github.com/awslabs/awscli-aliases

HTTPプロキシの設定

プロキシサーバーを使用してAWSにアクセスするには、以下のとおりに
HTTP_PROXY および HTTPS_PROXY 環境変数にプロキシのURLを設定します。

Linux, macOS, or Unix

```
$ export HTTP_PROXY=http://10.15.20.25:1234
$ export HTTP_PROXY=http://proxy.example.com:1234
$ export HTTPS_PROXY=http://10.15.20.25:5678
$ export HTTPS_PROXY=http://proxy.example.com:5678
```

Windows

```
C:¥> set HTTP_PROXY http://10.15.20.25:1234
C:¥> set HTTP_PROXY=http://proxy.example.com:1234
C:¥> set HTTPS_PROXY=http://10.15.20.25:5678
C:¥> set HTTPS_PROXY=http://proxy.example.com:5678
```

https://docs.aws.amazon.com/ja_jp/cli/latest/userguide/cli-configure-proxy.html

設定のまとめ

- 複数プロファイルの設定、実行方法
    - 複数プロファイルの作成方法、切り替え方法
    - 他アカウントのリソースへアクセス
    - プロファイルの設定項目について
    - MFA (多要素認証) を使ったアクセス
- S3コマンドの設定
- HTTPプロキシの設定

aws

アジェンダ

概要

- セットアップ方法
- 操作方法
- 設定
- **操作パターン集**

操作パターン集

aws

AWS CLIの操作パターン集

サービス

- EC2
- S3
- IAM

ユースケース

- S3に置かれたファイルをディスクに保存せず圧縮したい
- AWSのアカウントIDを確認したい
- CloudFormationのスタックが作成完了するまで待機させたい
- AWS CLIの実行履歴を確認したい
- オンプレミス、S3間のファイルアップロード、ダウンロード

サービス (1/3)

EC2 ※オレンジ色の下線で示されているパラメータ値はご自身の値に置き換えてください。

EC2のインスタンスIDの一覧取得

```
$ aws ec2 describe-instances --query 'Reservations[].Instances[].[InstanceId, InstanceType, State.Name]' --output table
```

EC2のインスタンス停止

```
$ aws ec2 stop-instances --instance-ids ${インスタンスID}
```

EC2のインスタンス起動

```
$ aws ec2 start-instances --instance-ids ${インスタンスID}
```

EC2のインスタンス作成

```
$ aws ec2 run-instances --image-id ami-xxxxxxxx --count 1 --instance-type t2.micro --key-name MyKeyPair --security-group-ids sg-xxxxx --subnet-id subnet-xxxxx
```

https://docs.aws.amazon.com/ja_jp/cli/latest/userguide/cli-services-ec2.html

aws

サービス (2/3)

S3 High-Level Command

S3バケットの作成

```
$ aws s3 mb s3://${バケット名}
```

S3バケットの一覧表示

```
$ aws s3 ls
```

S3バケットの削除

```
$ aws s3 rb s3://${バケット名}
```

S3バケットへのファイルのアップロード

```
$ aws s3 cp file.txt s3://${バケット名}/
```

S3バケットのファイル、フォルダの同期

```
$ aws s3 sync ./folder s3://${バケット名}/
```

https://docs.aws.amazon.com/ja_jp/cli/latest/userguide/cli-services-s3.html

aws

サービス (3/3)

IAM

IAMグループの作成

```
$ aws iam create-group --group-name MyIamGroup
```

IAMユーザーの作成

```
$ aws iam create-user --user-name MyUser
```

IAMグループの詳細確認

```
$ aws iam get-group --group-name MyIamGroup
```

IAMユーザーへIAMポリシーのアタッチ

```
$ aws iam attach-user-policy --user-name MyUser --policy-arn ${ポリシーのARN}
```

IAMユーザーにアタッチされたIAMポリシーの確認

```
$ aws iam list-attached-user-policies --user-name MyUser
```

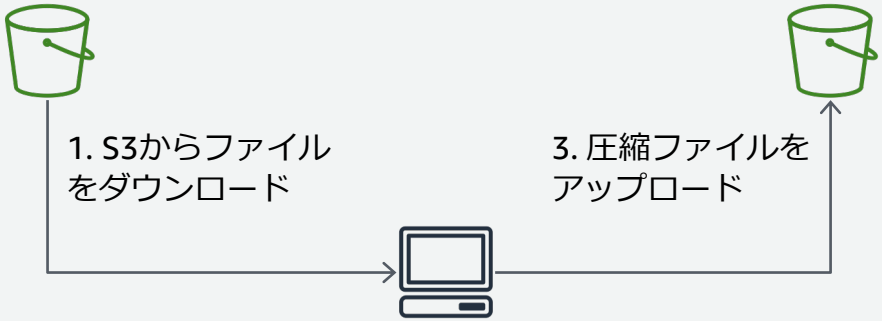
https://docs.aws.amazon.com/ja_jp/cli/latest/userguide/cli-services-iam.html

ユースケース (1/5)

S3に置かれたファイルをディスクに保存せず圧縮したい

```
$ aws s3 cp s3://bucket/key - | bzip2 --best | aws s3 cp - s3://bucket/key.bz2
```

`aws s3 cp` コマンドでは、入出力パラメーターにハイフン `-` を指定することで、標準入出力とS3との間でダウンロード、アップロードが可能です。



http://docs.aws.amazon.com/cli/latest/reference/s3/cp.html

ユースケース (2/5)

AWSのアカウントIDを確認したい

```
$ aws sts get-caller-identity
```

```
{
    "UserId": "XXX…",
    "Account": "123456789012",
    "Arn": "arn:aws:iam:: … :user/hoge"
}
```

別アカウントのプロファイルであれば、`--profile` をつけて別アカウントのアカウントIDも確認可能

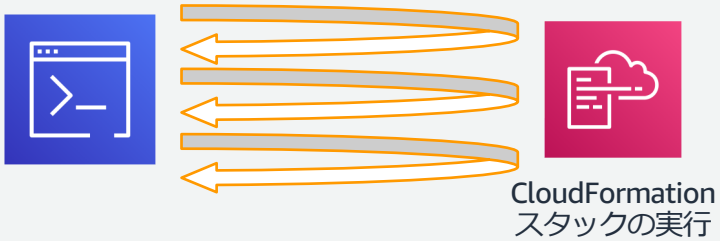
ユースケース (3/5)

CloudFormationのスタックが作成完了するまで待機させたい

```
$ aws cloudformation wait stack-create-complete --stack-name ${スタック名}
```

スタックステータスが `CREATE_COMPLETE` になるまで待機。

指定したステータスになるまで 30 秒ごとにポーリングし、確認に 120 回失敗すると、リターンコード 255 で終了します。



http://docs.aws.amazon.com/cli/latest/reference/cloudformation/wait/stack-create-complete.html

ユースケース (4/5)

AWS CLIの実行履歴を確認したい

リスト形式での表示。コマンドID、日付、引数、リターンコードが含まれる。

```
$ aws history list
```

```
XXXX-XXXX-XXXX-XXXX 2019-01-01 00:00:01 AM  iam get-group           255
XXXX-XXXX-XXXX-XXXX 2019-01-01 00:00:00 AM  ec2 describe-instances  0
```

実行結果の詳細表示。`aws history list` で表示されたコマンドIDも指定可能。

```
$ aws history show
```

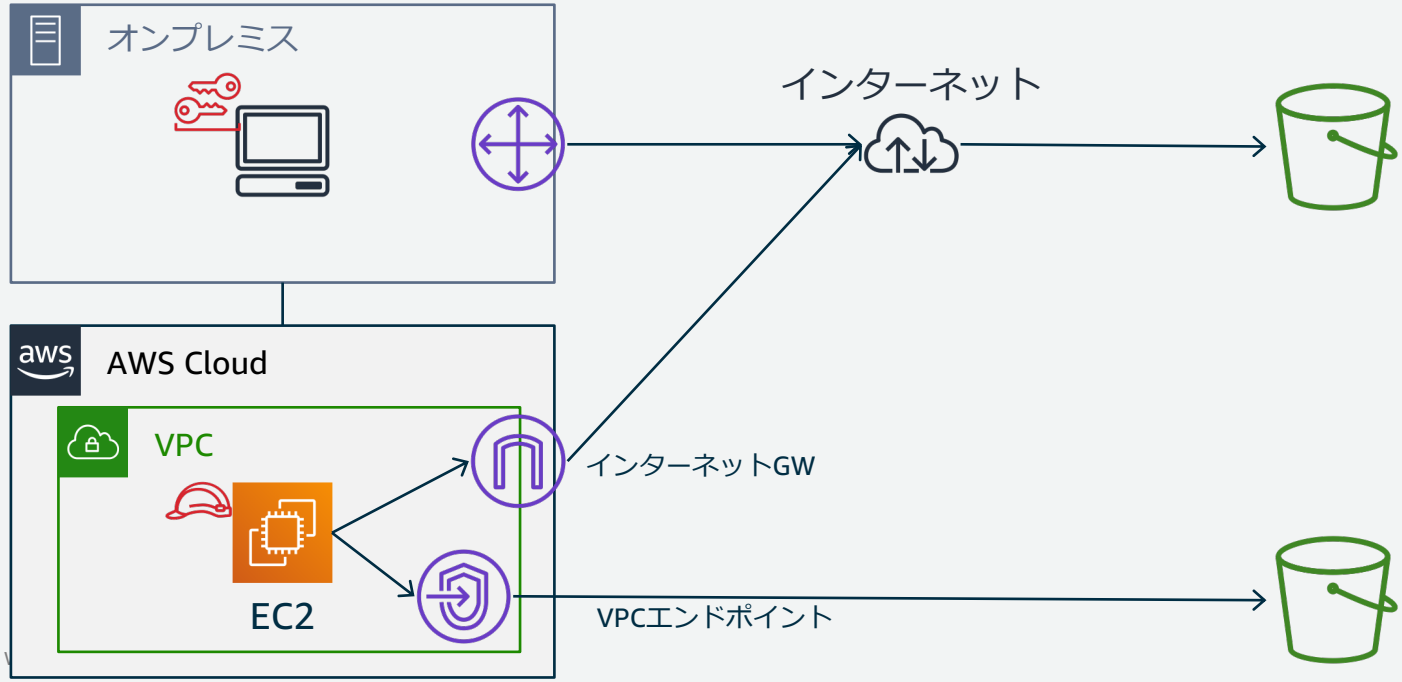
```
AWS CLI command entered
at time: 2019-00-00 00:00:00.000
with AWS CLI version: aws-cli/1.16.170 Python/3.7.3 Darwin/18.6.0 botocore/1.12.160
with arguments: ['ec2', 'describe-instances', '--output', 'table']

[0] API call made
at time: 2019-00-00 00:00:00.000
…
```

https://docs.aws.amazon.com/cli/latest/reference/history/index.html

ユースケース (5/5)

オンプレミス、S3間のファイルアップロード、ダウンロード

単一ファイルであれば、`aws s3 cp`、複数ファイルがあれば、`aws s3 sync`のコマンドで実現できます。閉域で行いたい場合は、VPCエンドポイントをご利用ください。



ユースケースのまとめ

サービス

- EC2
- S3
- IAM

ユースケース

- S3に置かれたファイルをディスクに保存せず圧縮したい
- AWSのアカウントIDを確認したい
- CloudFormationのスタックが作成完了するまで待機させたい
- AWS CLIの実行履歴を確認したい
- オンプレミス、S3間のファイルアップロード、ダウンロード

まとめ

aws

まとめ

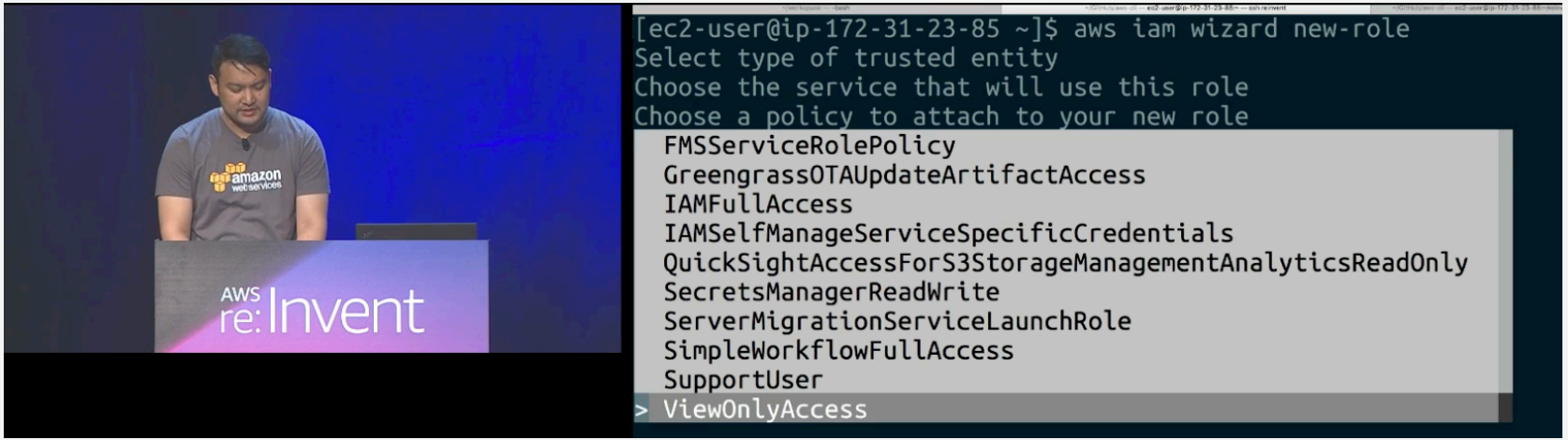
本日以下のことをお伝えしました。

- AWS CLIの概要
- AWS CLIのセットアップ方法
- AWS CLIの操作方法
- AWS CLIの設定方法
- 操作パターン集

(参考) AWS CLI Version 2について

re:Invent 2018にてVersion 2が発表されました。現在はプレビュー版ですのでご興味のある方は使ってみて、是非フィードバックをください。主な特徴は以下の通りです。

- 自動補完の性能向上
- Wizard形式での実行が可能に
- Mac、Linuxでもインストーラーを選択できるように (今まではWindowsのみ)



https://www.youtube.com/watch?v=i4Prnei87ao

https://aws.amazon.com/jp/blogs/developer/aws-cli-v2-development/

(参考) AWS CLIの過去の発表

Deep Dive: AWS Command Line Interface
https://www.youtube.com/watch?v=ZbgvG7yFoQI

AWS re:Invent 2017: AWS CLI: 2017 and Beyond
https://www.youtube.com/watch?v=W8IyScUGuGI

AWS re:Invent 2017: Introduction to the AWS CLI
https://www.youtube.com/watch?v=iC8zVT5r7Jw

AWS re:Invent 2018: [REPEAT 1] What's New with the AWS CLI
https://www.youtube.com/watch?v=i4Prnei87ao

JAWS-UG CLI専門支部のご紹介

AWS CLIに興味がある、使いこなしたい、広めたいユーザーが集まるユーザーグループです。2019年7月時点で130回を超えるイベントを開催しており、ハンズオンを通して理解を深める場となっております。気軽に相談、議論ができる場が提供されているので、ご興味のある方は是非ご参加ください。

JAWS-UG CLI専門支部: https://jawsug-cli.doorkeeper.jp/

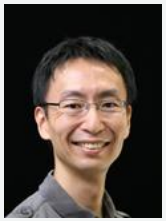
Q&A

お答えできなかったご質問については
AWS Japan Blog 「https://aws.amazon.com/jp/blogs/news/」にて
後日掲載します。

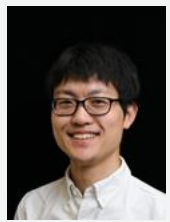
7 月の Black Belt Online Seminar 配信予定

https://amzn.to/JPWebinar

~~7/3 (水) 18:00-19:00 Amazon MQ~~
~~7/5 (金) 18:00-19:00 AWS Summit TOKYO/OSAKA 2019 振り返り 2019 年主要アップデートまとめ~~
~~7/16 (火) 12:00-13:00 Amazon Personalize~~
~~7/17 (水) 18:00-19:00 Amazon Simple Queue Service(SQS)~~
~~7/23 (火) 12:00-13:00 AWS CloudHSM~~
~~7/24 (水) 18:00-19:00 AWS Command Line Interface~~
7/30 (火) 12:00-13:00 Amazon CloudFront
7/31 (水) 18:00-19:00 Amazon ECS Deep Dive





AWS の日本語資料の場所「AWS 資料」で検索

https://amzn.to/JPArchive

AWS Well-Architected 個別技術相談会

毎週”W-A個別技術相談会”を実施中

- AWSのソリューションアーキテクト(SA)に対策などを相談することも可能
- 申込みはイベント告知サイトから

(https://aws.amazon.com/jp/about-aws/events/)

AWS イベント で[検索]

ご視聴ありがとうございました

AWS 公式 Webinar
https://amzn.to/JPWebinar

過去資料
https://amzn.to/JPArchive