AWS

SUMMIT

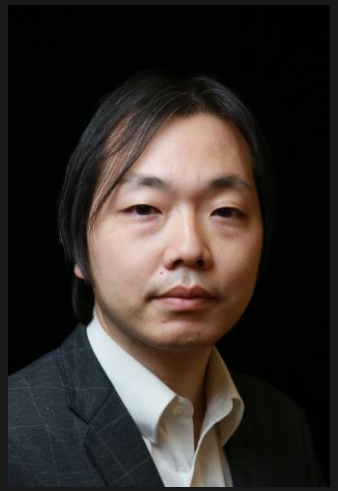
Amazon EC2 Systems Managerによるハイブリッド環境の管理

アマゾン ウェブ サービス ジャパン株式会社
ソリューションアーキテクト　渡邉源太

2017/6/1

amazon web services

自己紹介



名前

- 渡邉源太

所属

- アマゾン ウェブ サービス ジャパン株式会社
  - ソリューションアーキテクト

好きなAWSサービス

- Amazon WorkSpaces/AppStream 2.0

前提条件とセッションのゴール

想定する前提条件

- Amazon EC2/S3/VPC/IAMなど基本的なAWSサービスについての知識
- WindowsおよびLinuxに関する一般的なシステム管理の知識・経験
- AWS CLIの使い方に関する知識

セッションのゴール

- Amazon EC2 Systems Managerの概要と使い方について理解する
- 代表的な利用シナリオにおけるメリットについて理解する

アジェンダ

- Amazon EC2 Systems Managerとは
- 利用シナリオ
  - 管理タスクの実行
  - パッチ管理
  - AMI作成の自動化
- 料金
- まとめ

クラウドはNew Normal

お客様の課題

従来のITツールはクラウド向けではない

エンタープライズ規模での可視性の維持が困難

複数の製品の展開によるオーバーヘッド

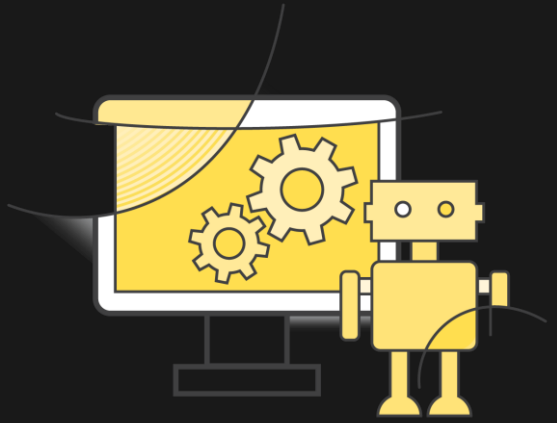
ライセンスのコストと複雑さ

従来のツールによるクラウドとハイブリッド環境の管理は複雑で高コスト


Prepare
NEW
Amazon EC2 Systems Manager
Collection of AWS tools for package installation, patching, resource configuration, and task automation
Generally Available Today

Amazon EC2 Systems Managerとは

Amazon EC2、またはオンプレミスで実行される
Windows、Linuxに対してシステムの自動構成と継続的な
管理を可能にする一連のサービス



EC2 Systems Managerの構成要素

デプロイ、構成
および管理

Run Command

State Manager

共有コンポーネント

Maintenance
Window

Parameter Store

トラッキングと
アップデート

Inventory

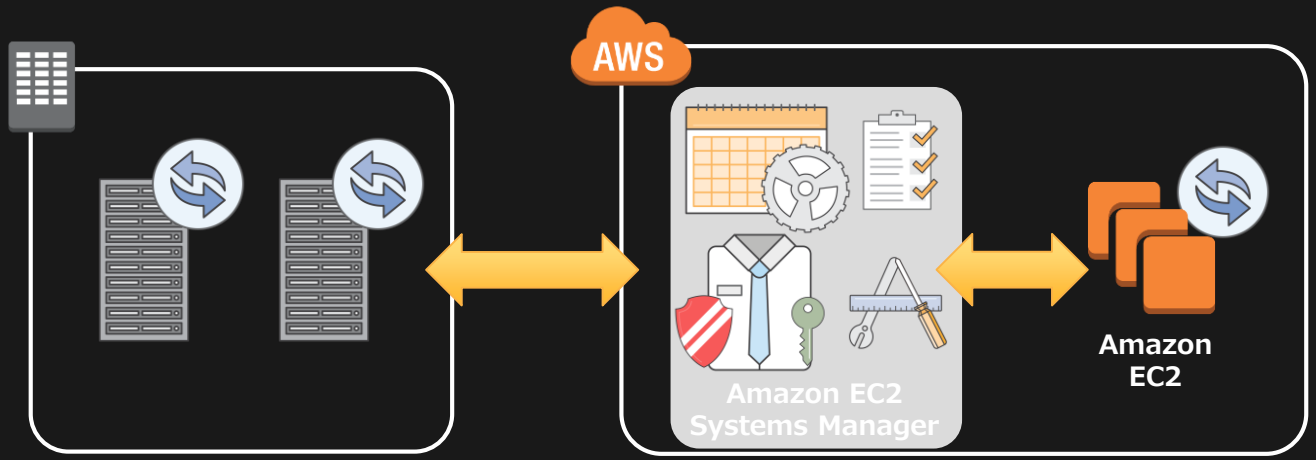
Patch Manager

Automation

利用イメージ (1)

Amazon EC2、ハイブリッド環境のシステム管理

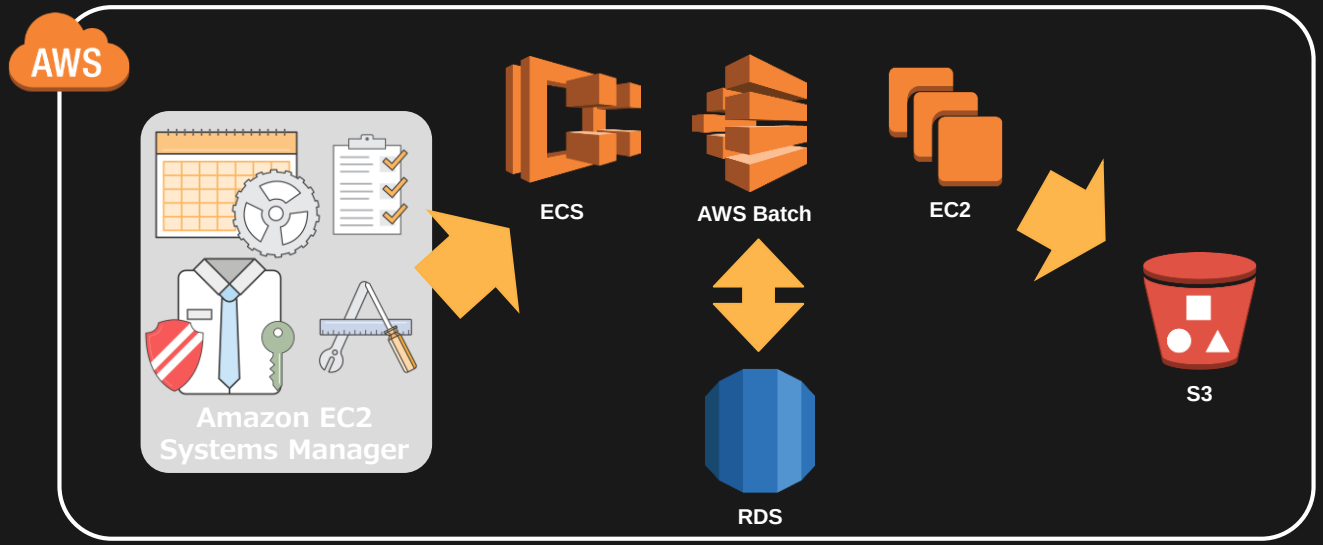
- Run Commandのみでコマンド実行、管理
- 情報はInventory、Patch Managerによるパッチ適用



利用イメージ (2)

認証情報の保存、取り出し

- Parameter Storeを使ってバッチ処理



EC2 Systems Managerの前提条件

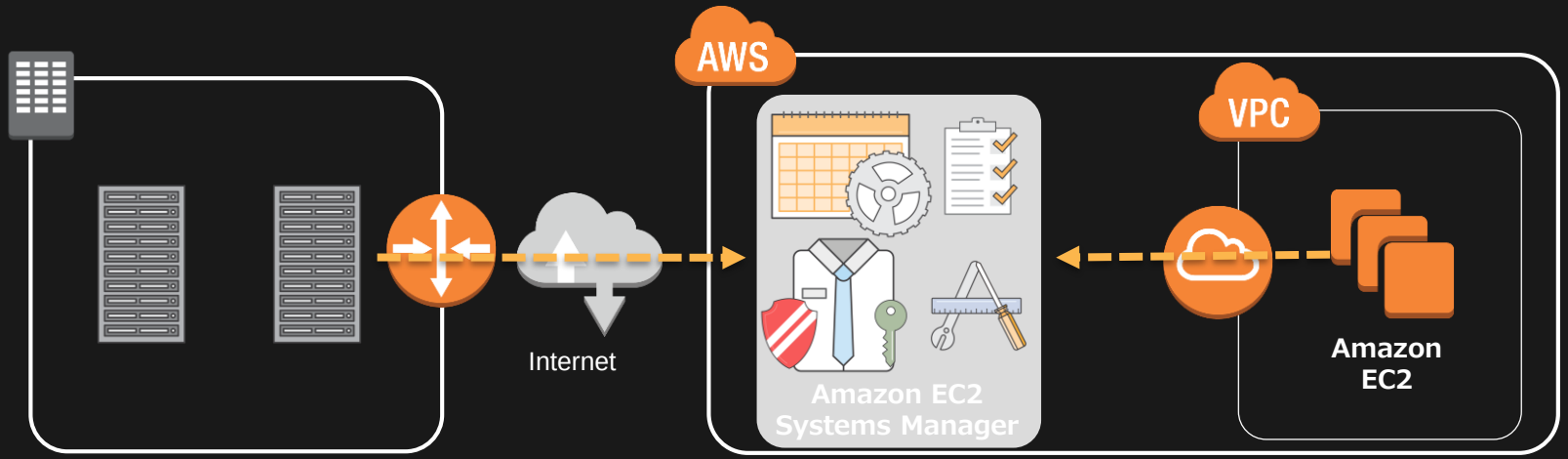
サポートされているOS

- Windows(32bit、64bit)
  - Windows Server 2003～2016(R2含む)
- Linux(32bit)
  - Amazon Linux 2014.09、2014.03 以降
  - Ubuntu Server 16.0.4 LTS、14.04 LTS、または 12.04 LTS
  - Red Hat Enterprise Linux (RHEL) 6.5 以降
  - CentOS 6.3 以降
- Linux(64bit)
  - Amazon Linux 2015.09、2015.03 以降
  - Red Hat Enterprise Linux (RHEL) 7.x 以降
  - CentOS 7.1 以降

EC2 Systems Managerの前提条件

インターネットへのアクセスが出来る事

- SSM Agentが各種APIへアクセスするため
- VPCにNAT-Gatewayを設置するのも方法の1つ



利用シナリオ：
管理タスクの実行

Run Command

管理作業をリモートから実行

- リモートから任意のコマンド実行が可能
  - ソフトウェアのインストール、パッチング、アップデート
  - ユーザーの追加・削除、サービスの起動・停止、状態取得
- JSONベースのドキュメントでコマンド、タスクを定義
- 定義済みのドキュメントも提供、コミュニティ版もあり
- 実行結果はS3に保存可能、実行状態に合わせてSNSを使って通知
- SSH、RDPの接続ポートを閉じる事でセキュアに運用

SSM Agent

管理対象に常駐し、各種サービスからの要求を実行

- インストールタイプのエージェント
- AWSの各種サービスAPIと通信が発生
- AWSから提供している以下のAMIにはインストール済み
  - 2016年11月以降に提供しているWindows AMI(2003-2012R2、2016)
- Amazon Linuxは別途インストールが必要
- Source CodeはGithubで公開
  - https://github.com/aws/amazon-ssm-agent

インスタンスの死活監視

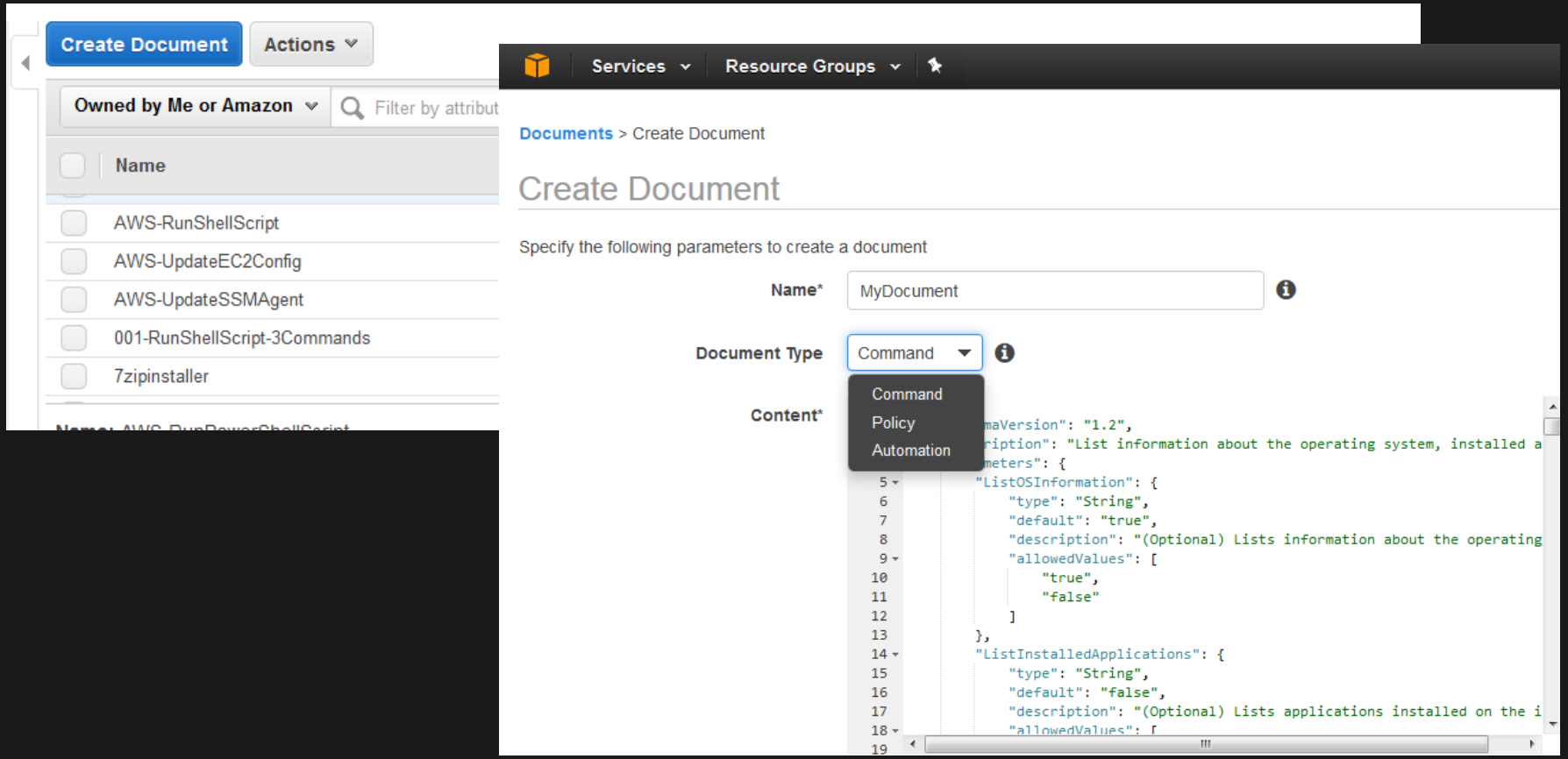
```
D:¥Users¥gentaw>aws ssm describe-instance-information
{
    "InstanceInformationList": [
        {
            "IsLatestVersion": false,
            "PingStatus": "Online",
            "InstanceId": "i-c6d69773",
            "ResourceType": "EC2Instance",
            "AgentVersion": "3.17.1032",
            "PlatformVersion": "6.2.9200",
            "PlatformName": "Windows Server 2012 Standard",
            "PlatformType": "Windows",
            "LastPingDateTime": 1477203028.78
        },
```

ドキュメント

- EC2 Systems Managerの動作はドキュメントで定義する
- ドキュメントはバージョン管理、共有が可能
- 定義済みのドキュメントの使用、カスタマイズドキュメントの作成可能
  - 例：Software Inventory を収集するDocument

タイプ	使用用途
コマンドのドキュメント	Run Command State Manager
ポリシードキュメント	State Manager
自動化ドキュメント	Automation

Systems Manager – ドキュメント



コマンドの送信

```
aws ssm send-command
--document-name AWS-RunPowerShellScript
--instance-id i-1234567
--parameters  commands=“mkdir C:¥Demo”
--service-role-arn <my-service-role>
-- notification-config NotificationArn=<my-topic-
arn>,NotificationEvents=“Success”,NotificationType=“Command”
```

リモートでインスタンス上にディレクトリを作成して完了したらSNSで通知

Run Command – スケール

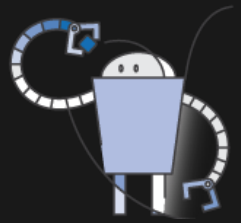
- タグのクエリをベースにコマンドを送信
- 多重度のコントロールとエラーハンドリング

```
aws ssm send-command --document-name <value> --targets
“Key=tag:ServerRole;Values=WebFrontEnd” […]

aws ssm send-command --max-concurrency 10 …

aws ssm send-command --max-errors 10 …
```

State Manager

OSとアプリケーションの設定を定義、状態を維持する

- 事前に定義しておいた状態にOSを設定
- JSONベースのドキュメントを使用してポリシーを定義
- 構成を適用するEC2、オンプレサーバーを個別、タグで管理
- 企業全体の構成ポリシーへの準拠を支援
- 例：ウィルス対策ソフト、マルウェア対策ソフトの定義ファイル更新



関連付けの作成

```
aws ssm create-association
--document-name WebServerDocument
--document-version ¥$DEFAULT
--schedule-expression cron(0 */30 * * * ? *)
--targets “Key=tag:Name;Values=WebServer”
--output-location "{ ¥"S3Location¥": { ¥"OutputS3Region¥":
¥“us-east-1¥", ¥"OutputS3BucketName¥": ¥“MyBucket¥",
¥"OutputS3KeyPrefix¥": ¥“MyPrefix¥" } }“
```

タグのクエリにマッチするすべてのインスタンスを構成して30分ごとに再適用する

利用シナリオ：
パッチ管理

エンタープライズシステムのパッチ管理の課題

- うんざりする繰り返しによる時間の浪費
- 既存のソリューションではスケーラビリティが不安
- エンタープライズのパッチ管理はマニュアルかつ複雑
- エラーによるダウンタイムとセキュリティ問題

Patch Manager

ベースラインを定義してWindowsのパッチを適用

- Patch Baselineを使ってカスタムパッチポリシーを定義
  - 例：クリティカルなパッチが提供された場合には1日後に適用
- パッチ適用は指定したMaintenance Window内で実施
- 実施されたパッチングの結果はレポートされる
  - インストールされたパッチ、スキップ、失敗したパッチ等
- 重要なアップデートやゼロデイ脆弱性への対応を自動化、時間を短縮

Patch Manager – Getting Started

1. 許可するパッチの定義のため **Patch Baseline** を作成

2. インスタンス群にパッチをスケジュールするため **Maintenance Window** を作成

3. **Maintenance Window** がパッチを適用

4. **Patch Compliance** で結果を監査

Patch Manager – オーバービュー

Patch Baseline

- ルールのクライテリア
    - プロダクト (Windows Server 2012 R2)
    - MSRC 重要度 (緊急)
    - 承認の遅延 (5日)
- 承認および拒否するパッチ (KB2032276, KB2124261)
- Patch Groupタグを使用してターゲットのインスタンスを登録

- 例: Patch Group:Prod instancesに対してWindows Server 2012 R2のすべての緊急アップデートをリリースの5日後に承認、ただしKB2032276をのぞく

Maintenance Window

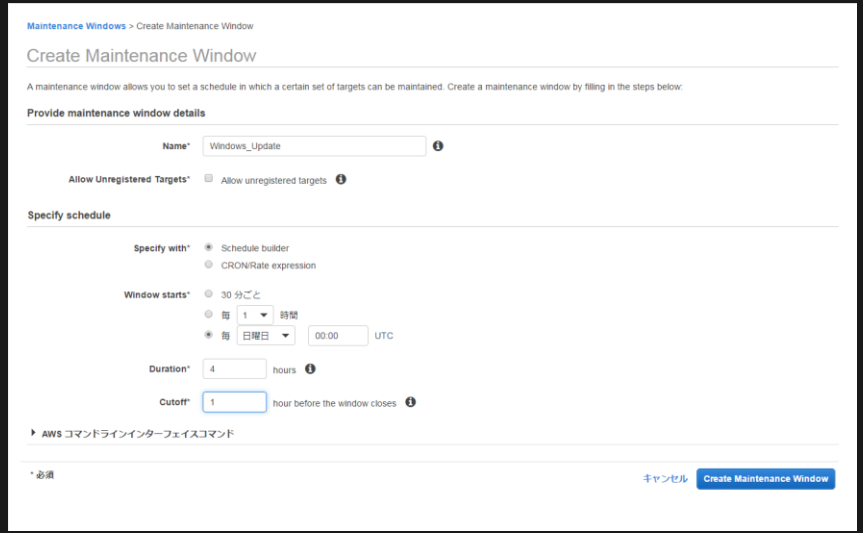
事前に設定した時間でメンテナンスを実施

- システムの中断を許容できる時間を指定
  - OSのアップデート、各種ドライバの更新、ソフトウェアのインストール　etc…
- 組み込み済みのコマンド、Run Commandの実行が可能
- メンテナンス実行の時間を明確に決めておく事で可用性と信頼性向上
- 考え方はAmazon RDSのMaintenance Windowと同様

Maintenance Windowの作成

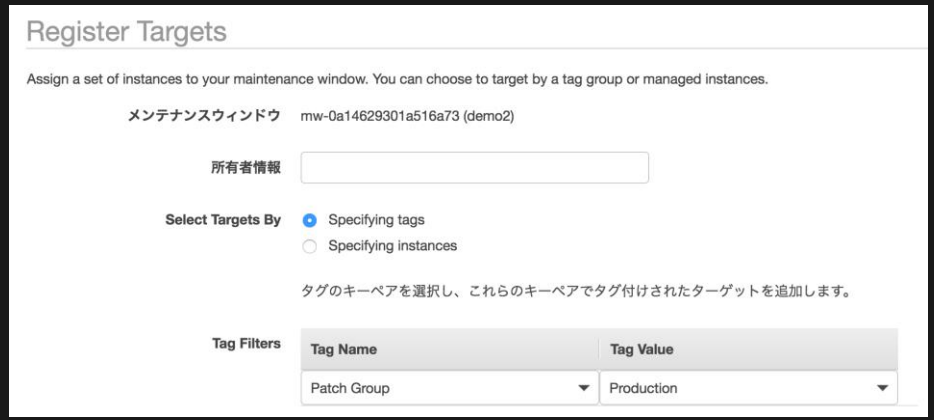
業務に影響のあるオペレーションを実行する時期の定義

- [Name]パッチ適用の名前
- [Specify schedule]スケジュールオプション(例：毎月第二火曜日)
- [Duration]実行する時間
- [Cutoff]終了してから新しいタスクの開始を停止するまでの時間



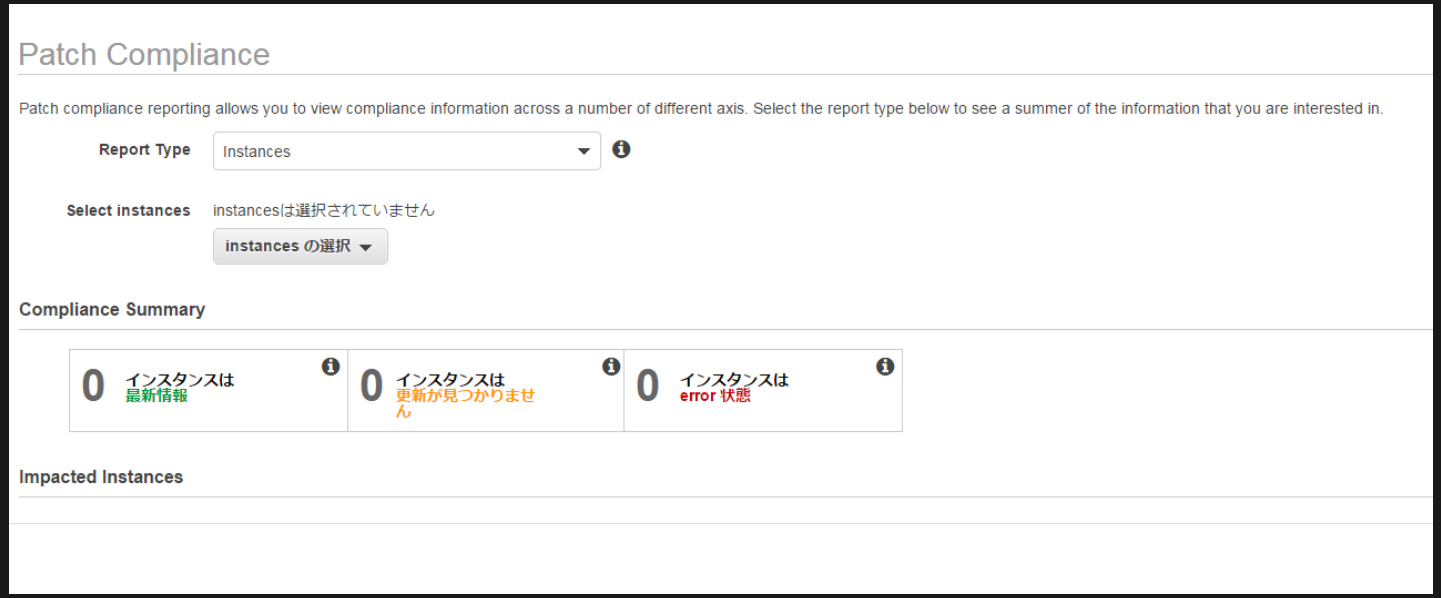
インスタンスへのパッチ適用

- Maintenance Windowを作成してパッチ適用のオペレーションをスケジュール
- タグ名のキーとしてPatch Groupおよび”Production”などを値として指定してターゲットを選択
- AWS-ApplyPatchBaselineコマンドをタスクとして登録
- 同時にパッチ適用する最大インスタンス数、エラーの閾値の設定が可能



Patch Compliance

- フリート全体のパッチステータスのサマリ
- ダッシュボードにインスタンスのコンプライアンス状況を表示



利用シナリオ：
AMI作成の自動化

自動化のペインポイント：AMIの作成

- パッチ適用、ハードニング、アプリケーションの更新などによるトリガー
- 終わりのないプロセス
- 時間の消費、とくにビルド失敗時
- ビルドサービスを管理するための手間がかかる

Automation

シンプルなワークフローを使って一般的なタスクを自動化

- Amazon Machine Images(AMI)の作成と管理に最適化
    - AMIからEC2を起動 → パッチ適用 → 更新されたAMIを作成
- JSONベースのドキュメントでワークフローを定義
- Run CommandとLambdaファンクションをサポート
- 企業で管理する「ゴールデンイメージ」管理をサポート

Automation – Getting Started

1. Automation ドキュメントの作成

2. Automation の実行

3. Automation のモニタリング

Automation – ドキュメント

- あらかじめ定義されたAutomationドキュメントを利用可能
  - AWS-UpdateWindowsAmi
  - AWS-UpdateLinuxAmi
- Automationドキュメントを作成することでカスタムのワークフローを定義して実行することが可能
  - Automationドキュメントにはワークフローの実行時に実行されるアクションがふくまれる

Automation – アクション

- **aws:changeInstanceState**: インスタンスの状態を変更
- **aws:copyImage**: 任意のリージョンから現在のリージョンに AMI をコピー、暗号化もサポート
- **aws:createImage**: 実行中のインスタンスから AMI を作成
- **aws:createStack**: テンプレートから新しいAWS CloudFormationスタックを作成
- **aws:createTags**: Amazon EC2 インスタンスまたは Systems Manager マネージドインスタンス用の新しいタグを作成
- **aws:deleteImage**: AMI を削除
- **aws:deleteStack**: AWS CloudFormationスタックを削除
- **aws:invokeLambdaFunction**: Automation ワークフローで外部ワーカーファンクションを実行
- **aws:runCommand**: リモートコマンドを実行
- **aws:runInstances**: 1 つ以上のインスタンスを起動
- **aws:sleep**: 指定した時間の間 Automationの実行を遅延

Automation – システム変数

- インプットおよびアウトプットパラメータを設定
  - デフォルト値の作成、または実行時にアサイン
  - Parameter Storeの統合
  - システム変数 (DATE, DATE_TIME, REGION, EXECUTION_ID)
- パラメータの例

ドキュメントパラメータの名前	デフォルト値
sourceAMIid	“{{ssm:sourceAMI}}”
targetAMIname	“patchedAMI-{{global:DATE_TIME}}”

Parameter Store

- Run Command、Automation、State Managerから参照可能
  - 例：Run Commandから参照する場合

```
aws ssm send-command --instance-ids i-1a2b3c4d5e6f7g8 ¥
  --document-name AWS-RunPowerShellScript ¥
  --parameter '{"commands":["echo {{ssm:param}}"]}'
```

- KMSで暗号化していた値を参照する場合
  - Parameter Storeを参照するインスタンスにIAM Roleを付与後に参照

```
{
  "Effect": "Allow",
  "Action": [
    "kms:Decrypt",
],
  "Resource": [
            "arn:aws:kms:region:account_id:key/key_id"
    ]
```

Automationの実行

- AWS-UpdateLinuxAmiドキュメントを使用して以下のタスクを実行することが可能
  - Amazon Linux/Red Hat/Ubuntu/Cent OS AMIのすべてのディストリビューションパッケージおよびAmazonソフトウェアをアップグレード
  - SSM AgentをインストールしてSystem Manager機能を有効化
  - 追加のソフトウェアパッケージをインストール

- AWSマネージメントコンソールまたはAWS CLIなどを使用してAutomationタスクを実行

```
aws ssm start-automation-execution --document-name “AWS-
UpdateLinuxAmi” --parameters
“AutomationAssumeRole=arn:aws:iam::1234561213:role/MyAutomationRole,
SourceAmiId=ami-e6d5d2f1, InstanceIamRole=MyEc2InstanceProfileRole"
```

Automation – モニタリング

- Automationイベントを通知するようにAmazon CloudWatch Eventsを設定
  - Automationのタスクが成功または失敗したときに通知を送信
  - ステップレベルまたはワークフローレベルの通知

- イベントをトリガーにSNSトピックによる通知や、Lambdaファンクションの実行など各種のターゲットを設定可能

Amazon EC2 Systems Managerを使うことで

- 運用コストを減らしてビジネスに注力
- 運用の品質安定、標準化が可能
- オンプレミス、クラウドの管理を一元化
- セキュアなオペレーションを実現

EC2 Systems Managerの料金

- EC2 Systems Manager自体の料金は**無料**
  - オンプレミス、EC2ともに台数による追加料金無し
  - 起動しているEC2インスタンスは別途料金発生
    (CloudFormationなどと同じ)

- S3、CloudTrail、AWS Configを利用すると発生
  - 料金体系は各サービスに準ずる

まとめ

- Amazon EC2 Systems Managerは、**ハイブリッド環境**でシステムの自動構成と継続的な管理を実現するサービス
- リモートでのコマンド実行、パッチ適用やAMI作成などの**タスクの自動化**が可能
- EC2 Systems Manager自体の料金は**無料**

本セッションのFeedbackをお願いします

受付でお配りしたアンケートに本セッションの満足度やご感想などをご記入ください
アンケートをご提出いただきました方には、もれなく素敵なAWSオリジナルグッズを
プレゼントさせていただきます

アンケートは受付、又はパミール3FのEXPO展示会場内にて回収させて頂きます

AWS

SUMMIT

amazon web services

Inventory

ソフトウェアインベントリの情報収集

- EC2、オンプレの各種インベントリ情報を収集、管理
- AWSが定義する収集テンプレートを利用可能
- JSON形式で取得したいデータを定義する事でカスタマイズも可能
- AWS Configを有効にする事でインベントリ情報の変更履歴を追跡
- ソフトウェアのライセンス使用状況確認、ソフトウェアバージョンの管理が簡素化される事でのセキュリティ脆弱性の早期発見

Parameter Store

ITリソースの集中管理

- ログイン、DB接続情報などを一元管理
- Run Commnad、State Manager、Automation Service等から参照可能
  - Management Consoleからも参照、更新が可能
  - AWS CLI、各種SDKからも参照、更新が可能
- 細かい権限管理で必要な人に必要な情報を提供
- Parameter Storeに格納した情報はKMSで暗号化
- 企業の機密情報の分散管理、メンテナンスを簡素化