Auto Scaling

AWS Black Belt Online Seminar 2017

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 千葉 悠貴
2017.03.15

スピーカー紹介

- 千葉 悠貴 (ちば ゆうき)

- AWS ソリューションアーキテクト
  - インターネットメディア担当

- 好きな AWS のサービス
  - CloudFormation
  - Code*
  - ECS

AWS Black Belt Online Seminar とは

- AWSJのTechメンバがAWSに関する様々な事を紹介するオンラインセミナーです

【**火曜 12:00~13:00**】
主にAWSのソリューションや
業界カットでの使いどころなどを紹介
(例：IoT、金融業界向け etc.)

【**水曜 18:00~19:00**】
主にAWSサービスの紹介や
アップデートの解説
(例：EC2、RDS、Lambda etc.)

※最新の情報は下記をご確認下さい。
オンラインセミナーのスケジュール&申し込みサイト
https://aws.amazon.com/jp/about-aws/events/webinars/

内容についての注意点

- 本資料では2017年3月15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サービスを使用する場合、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.

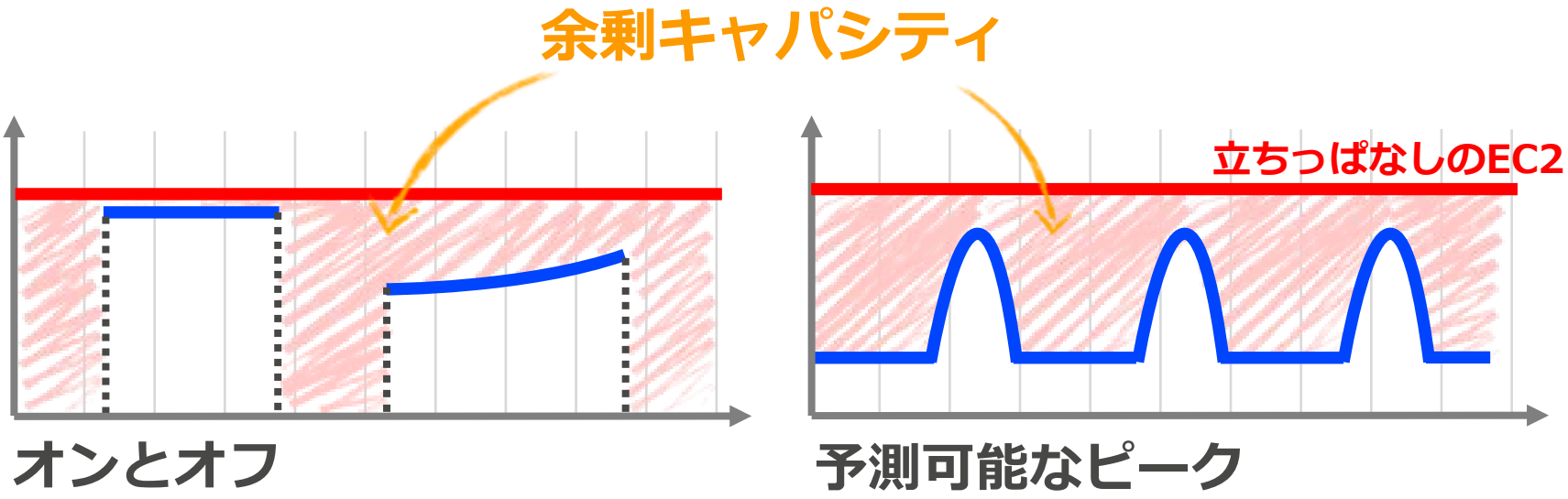
アジェンダ

- ❖Auto Scalingのコンセプト
- ❖Auto Scalingの構成要素
- ❖Auto Scaling実行の流れ
- ❖ライフサイクルと高度な制御
- ❖スケーリングの考慮事項
- ❖アーキテクチャーパターン
- ❖Application Auto Scaling
- ❖まとめ

アジェンダ

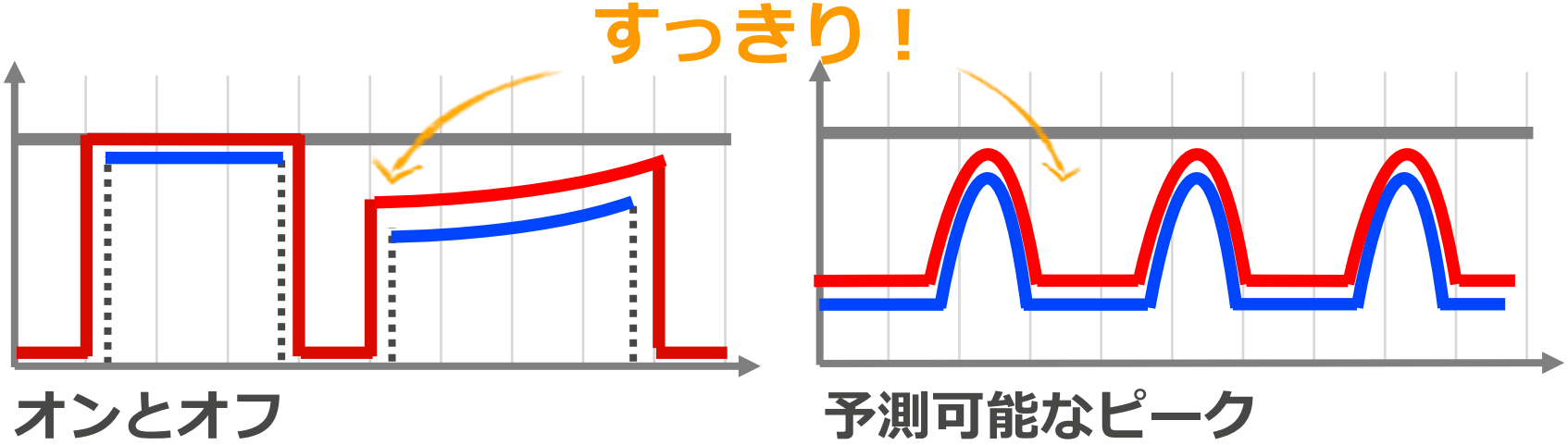
- ❖Auto Scalingのコンセプト
- ❖Auto Scalingの構成要素
- ❖Auto Scaling実行の流れ
- ❖ライフサイクルと高度な制御
- ❖スケーリングの考慮事項
- ❖アーキテクチャーパターン
- ❖Application Auto Scaling
- ❖まとめ

キャパシティプランの理想



- ❖ 無駄なサーバーは落としておきたい
- ❖ 負荷に応じて必要な台数のみを立てておきたい

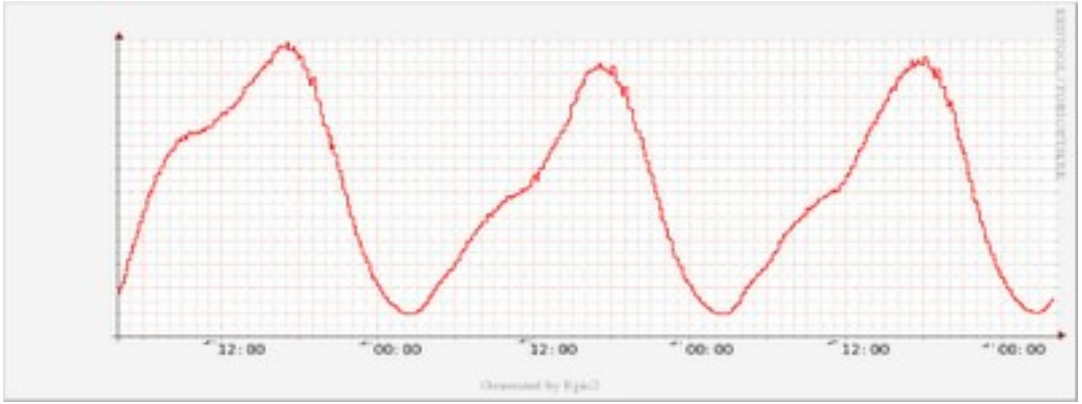
Auto Scaling



- 需要に応じて自動的にサーバが増減しコストカット
  - スケーリングのポリシーやインスタンスは柔軟に設定可能
  - 異常なインスタンスやAZを自動で切り離し高可用性を維持
- サービスが成長してきたら最大台数を増やして対応

Auto Scaling @ NETFLIX

Request

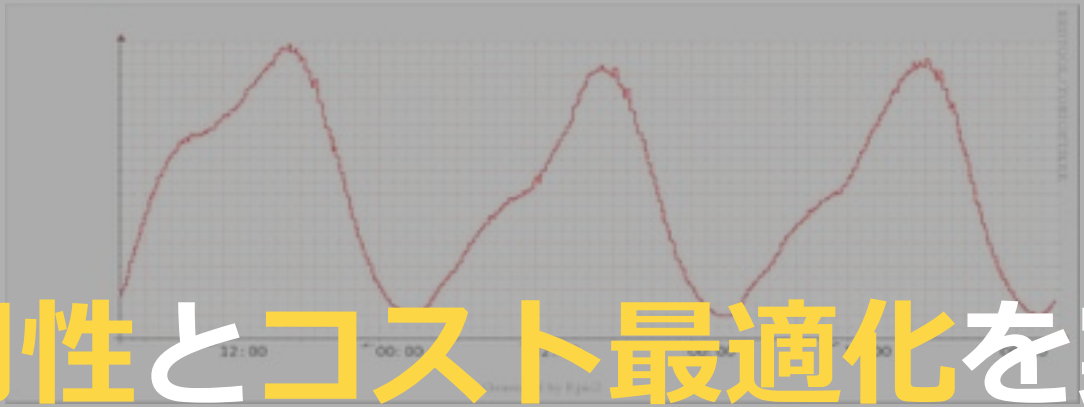


Instances

http://techblog.netflix.com/2012/01/auto-scaling-in-amazon-cloud.html

Auto Scaling @ NETFLIX

Request



高い可用性とコスト最適化を実現

Instances

http://techblog.netflix.com/2012/01/auto-scaling-in-amazon-cloud.html

アジェンダ

- ❖Auto Scalingのコンセプト
- ❖Auto Scalingの構成要素
- ❖Auto Scaling実行の流れ
- ❖ライフサイクルと高度な制御
- ❖スケーリングの考慮事項
- ❖アーキテクチャーパターン
- ❖Application Auto Scaling
- ❖まとめ

Auto Scalingの構成要素

Auto Scaling Group

Launch Configuration

Scaling Plan

Auto Scaling Group (以降ASGと記載)

Auto Scalingの全体的な情報管理単位

- 役割／機能
  - 起動インスタンスの台数を、設定した**最小値**以上に維持し続ける
  - 設定した**最大値**以上のインスタンスは起動しない
  - Desired Capacityの数量になるようにインスタンスを起動／終了
    Desired Capacity＝今時点で必要な台数
  - 起動台数をAZ間で**バランシング**する
  - AZ障害時は他のAZでインスタンスを起動する
- 設定項目
  - Launch Configuration
  - 最小/最大台数
  - VPC/サブネット
  - ELB/ALB
  - ヘルスチェック方法
  - ターミネーションポリシー
  - スケーリングポリシー
  - スケジュールアクション　など

http://docs.aws.amazon.com/ja_jp/autoscaling/latest/userguide/AutoScalingGroup.html

Launch Configuration

Auto Scalingで起動するEC2インスタンスの情報

- 役割／機能
  - **どのようなインスタンスを起動するか**を定義する
  - 通常のEC2インスタンスを起動する際の入力情報とほぼ同じ

- 設定項目
  - AMI
  - インスタンスタイプ
  - Security Group(s)
  - Key Pair
  - IAM role
  - user-data　など

http://docs.aws.amazon.com/ja_jp/autoscaling/latest/userguide/LaunchConfiguration.html

Scaling Plan

Auto Scalingで起動するEC2インスタンスの情報

- 役割／機能
  - **どのようにインスタンスをスケールするか**を定義する
  - １つのASGに複数のスケーリングプランを設定することも可能

- 種類
  - ASGサイズの維持
  - 手動スケーリング
  - スケジュールに基づくスケーリング
  - 動的なスケーリング（スケーリングポリシー）

http://docs.aws.amazon.com/ja_jp/autoscaling/latest/userguide/scaling_plan.html

Scaling Plan① : ASGサイズの維持

- 管理方法
  - **最小台数を維持**するよう設定する

- ユースケース : Auto Healing
  - インスタンスに障害が発生したとき自動的にサービスから切り離し、健全なインスタンスをサービスインさせる
  - ASGの最小値を任意の数に設定する
    - 最小値:最大値=1:1 の場合
      「常に1台起動。障害が起きたら自動的に1台起動しなおす」
    - 最小値:最大値=2:N の場合
      「常に2台起動。片方に障害があっても1台は耐える」

http://docs.aws.amazon.com/ja_jp/autoscaling/latest/userguide/as-maintain-instance-levels.html

Scaling Plan② : 手動スケーリング

- 管理方法
  - **Desired Capacityを手動で変更**する
  - ASGのインスタンスを手動でアタッチ／デタッチする
  - 操作にはコンソールまたはCLI/SDKを利用
- ユースケース : アドホックなスケーリング
  - 重めのバッチ処置を実施する前にインスタンス数を手動で増やす
  - ヘルスチェックには異常がないが動作がおかしいインスタンスを手動で切り離す

http://docs.aws.amazon.com/ja_jp/autoscaling/latest/userguide/as-manual-scaling.html

Scaling Plan③：スケジュールベース

- 管理方法
  - **定義したスケジュール**に基いて自動的にスケーリングを実施
  - スケジュールはコンソールまたはCLI/SDKで定義

- ユースケース：予測可能なスケーリング
  - 特定日時指定実行

    「2017/03/31 22:45 JST にインスタンスを30台にする」
  - 繰り返し実行(cronライクな定義方法)

    「毎週水曜日17:50に100台にして19:10に10台に戻す」
  - スケーリング開始は最大2分遅れる場合があるので注意
    - 数秒以内に開始、インスタンス初期化時間は別途必要

http://docs.aws.amazon.com/ja_jp/autoscaling/latest/userguide/schedule_time.html

Scaling Plan④：動的スケーリング

- 管理方法
  - **スケーリングポリシー**に基いて自動的にスケーリングが実施される
- スケーリングポリシー
  - 監視アラームとアクション(インスタンス追加／削除)を定義
  - 監視はCloudWatchがおこなう

    平均CPU使用率、ELB Latency、SQSのメッセージ数 etc…
  - スケーリングポリシータイプ

    1. シンプルスケーリングポリシー

       「平均CPU使用率が80%以上で2台追加」
    2. ステップスケーリングポリシー

       「平均CPU使用率60%以上で1台、80%以上でもう1台追加」

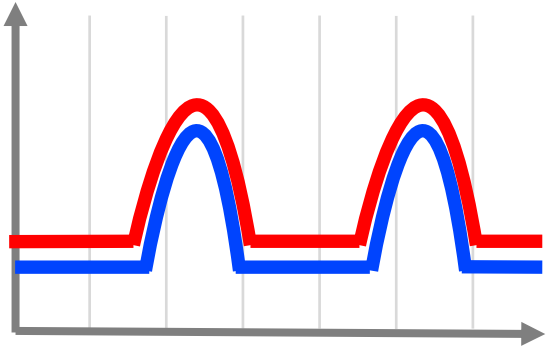
http://docs.aws.amazon.com/ja_jp/autoscaling/latest/userguide/as-scale-based-on-demand.html

Scaling Plan④：動的スケーリング

- スケーリング調整タイプ
  1. **ChangeInCapacity**
     - 指定したインスタンス数だけグループの現在の容量を増減
       「アラームが起きたら2台増やす」
  1. **ExactCapacity**
     - 指定したインスタンス数にグループの現在の容量を変更
       「アラームが起きたらASGのサイズを6台にする」
  1. **PercentChangeInCapacity**
     - 指定したインスタンス数にグループの現在の容量を変更
       「アラームが起きたら現在の台数から20%増やす」

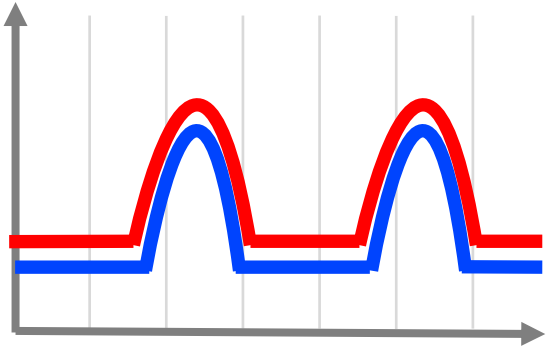
http://docs.aws.amazon.com/ja_jp/autoscaling/latest/userguide/as-scale-based-on-demand.html

複数のプランを組み合わせる

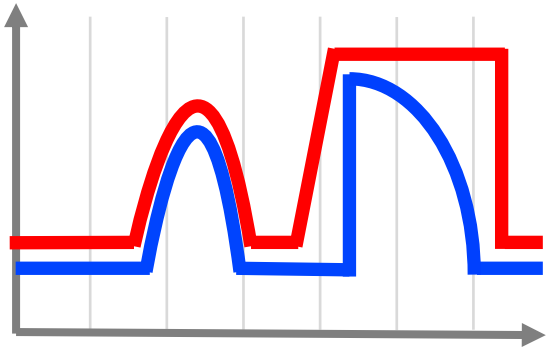


平常時は負荷傾向が
予測できる
⇩
スケジュールベース

複数のプランを組み合わせる

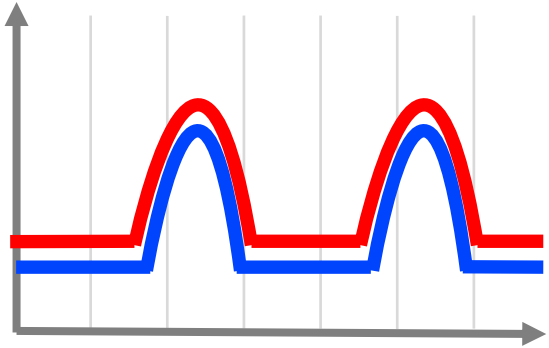


平常時は負荷傾向が
予測できる
⇩
スケジュールベース

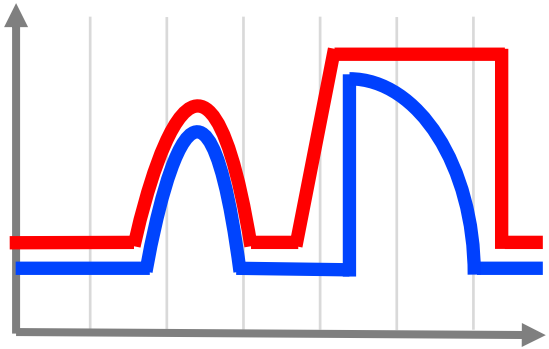


予定された大量
負荷への対策
⇩
＋手動スケーリング

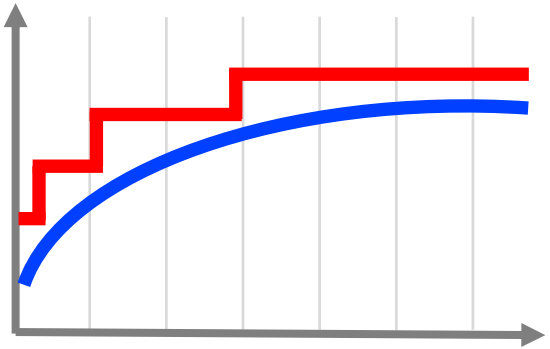
複数のプランを組み合わせる



平常時は負荷傾向が
予測できる
⇩
スケジュールベース



予定された大量
負荷への対策
⇩
＋手動スケーリング



予測ができない緩や
かな負荷変動の対策
⇩
＋動的スケーリング

Auto Scalingの構成要素

Auto Scaling Group

Launch Configuration

Scaling Plan

アジェンダ

- ❖Auto Scalingのコンセプト
- ❖Auto Scalingの構成要素
- ❖Auto Scaling実行の流れ
- ❖ライフサイクルと高度な制御
- ❖スケーリングの考慮事項
- ❖アーキテクチャーパターン
- ❖Application Auto Scaling
- ❖まとめ

Auto Scaling設定例

- Min:2, Max:6
- AZ, VPC/Subnet, ELB, Tags

- **Alarm:** HighCPUUtilization
- **Actions(Steps):**
  60 < value → add 2 Instances
  80 < value → add 10% of group
  30 > value → remove 2 Instances

Scaling Policy 1 (with steps)

- **Alarm:** HighLaytency
- **Action(only one):**
  300 < value → add 2 Instances

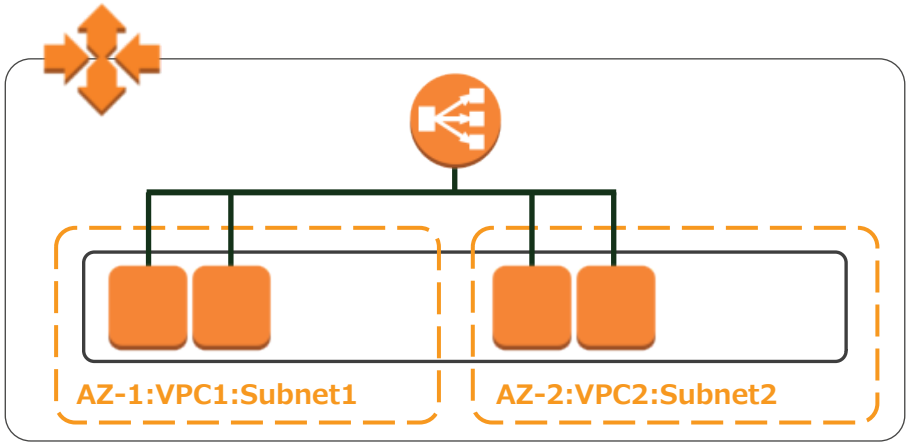
Scaling Policy 2 (simple policy)

Auto Scaling Group

Associate

- AMI, Instance Type,
  IAM Role, Security Group,
  Key Pair, user-data,
  Storage, Spot Price

Launch Configuration



①CloudWatchから、CPU使用率60%超のAlarm!

- Min:2, Max:6
- AZ, VPC/Subnet, ELB, Tags

- **Alarm:** HighCPUUtilization
- **Actions(Steps):**
  60 < value → add 2 Instances
  80 < value → add 10% of group
  30 > value → remove 2 Instances

Scaling Policy 1 (with steps)

- **Alarm:** HighLaytency
- **Action(only one):**
  300 < value → add 2 Instances

Scaling Policy 2 (simple policy)

Auto Scaling Group

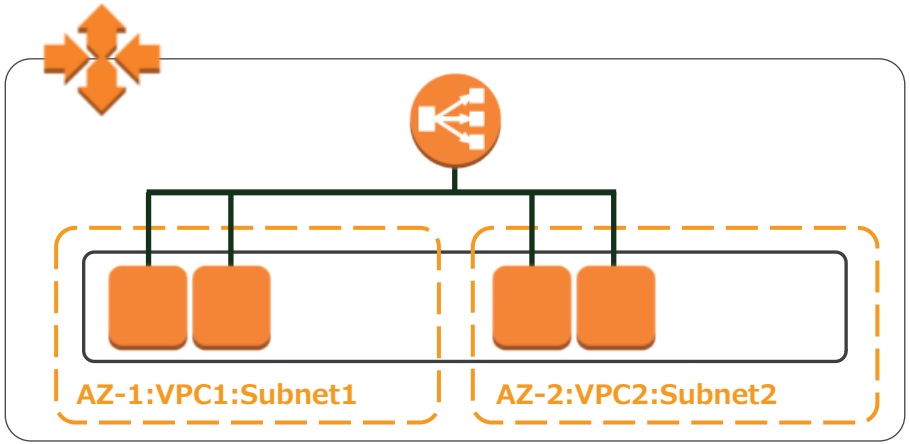
Associate

Alarm!!

CloudWatch

- AMI, Instance Type,
  IAM Role, Security Group,
  Key Pair, user-data,
  Storage, Spot Price

Launch Configuration



②Min-Max:2-6で現在4台なのであと2台増やせる

- ❑ Min:2, Max:6
- ❑ AZ, VPC/Subnet, ELB, Tags

- ❑ **Alarm:** HighCPUUtilization
- ❑ **Actions(Steps):**
  60 < value → add 2 Instances
  80 < value → add 10% of group
  30 > value → remove 2 Instances

Scaling Policy 1 (with steps)

- ❑ **Alarm:** HighLaytency
- ❑ **Action(only one):**
  300 < value → add 2 Instances

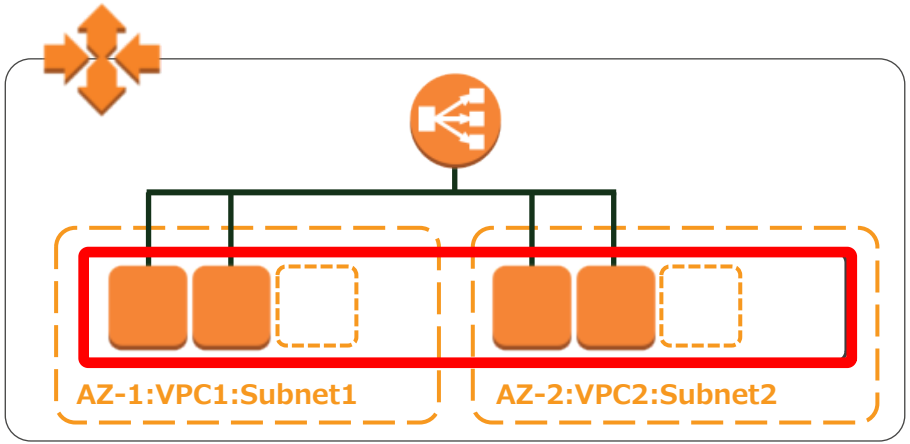
Scaling Policy 2 (simple policy)

Auto Scaling Group

Associate

- ❑ AMI, Instance Type,
  IAM Role, Security Group,
  Key Pair, user-data,
  Storage, Spot Price

Launch Configuration



③Launch Configurationに従いインスタンス起動

- Min:2, Max:6
- AZ, VPC/Subnet, ELB, Tags

- **Alarm:** HighCPUUtilization
- **Actions(Steps):**
  60 < value → add 2 Instances
  80 < value → add 10% of group
  30 > value → remove 2 Instances

Scaling Policy 1 (with steps)

- **Alarm:** HighLaytency
- **Action(only one):**
  300 < value → add 2 Instances

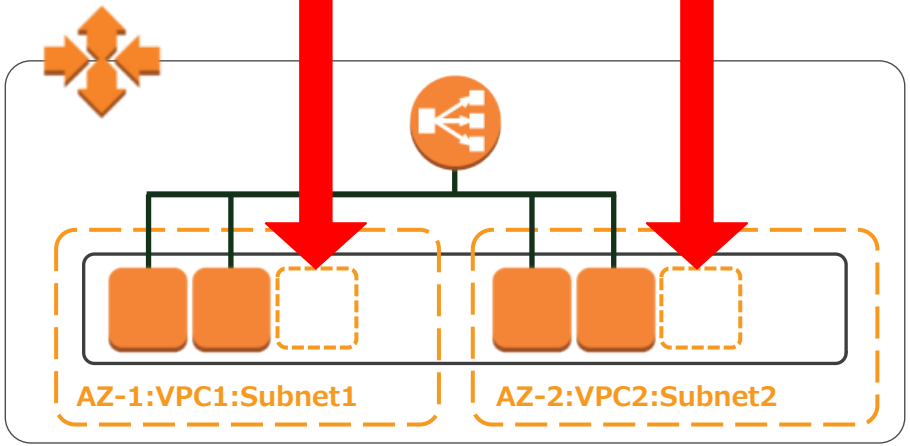
Scaling Policy 2 (simple policy)

Auto Scaling Group

Associate

- AMI, Instance Type,
  IAM Role, Security Group,
  Key Pair, user-data,
  Storage, Spot Price

Launch Configuration



④インスタンスの初期化、ヘルスチェックを実行

- Min:2, Max:6
- AZ, VPC/Subnet, ELB, Tags

- **Alarm:** HighCPUUtilization
- **Actions(Steps):**
  60 < value → add 2 Instances
  80 < value → add 10% of group
  30 > value → remove 2 Instances

Scaling Policy 1 (with steps)

- **Alarm:** HighLaytency
- **Action(only one):**
  300 < value → add 2 Instances

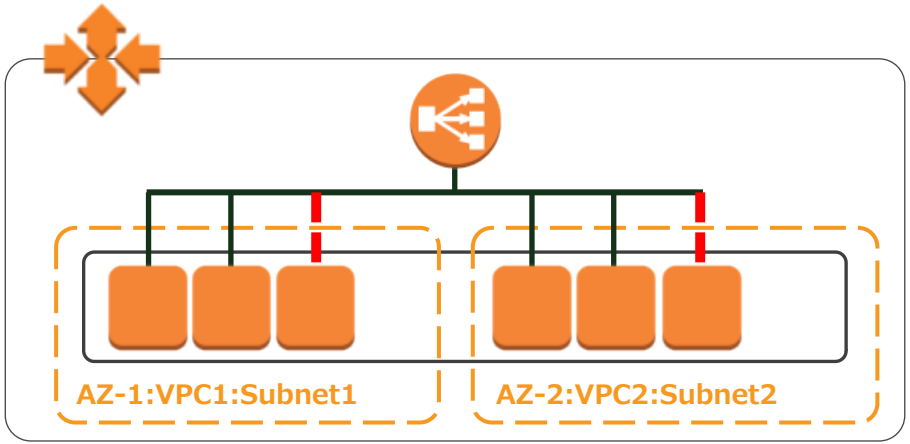
Scaling Policy 2 (simple policy)

Auto Scaling Group

Associate

- AMI, Instance Type,
  IAM Role, Security Group,
  Key Pair, user-data,
  Storage, Spot Price

Launch Configuration



⑤ヘルスチェックに問題がなければELB配下に追加

- Min:2, Max:6
- AZ, VPC/Subnet, ELB, Tags

- **Alarm:** HighCPUUtilization
- **Actions(Steps):**
  60 < value → add 2 Instances
  80 < value → add 10% of group
  30 > value → remove 2 Instances

Scaling Policy 1 (with steps)

- **Alarm:** HighLaytency
- **Action(only one):**
  300 < value → add 2 Instances

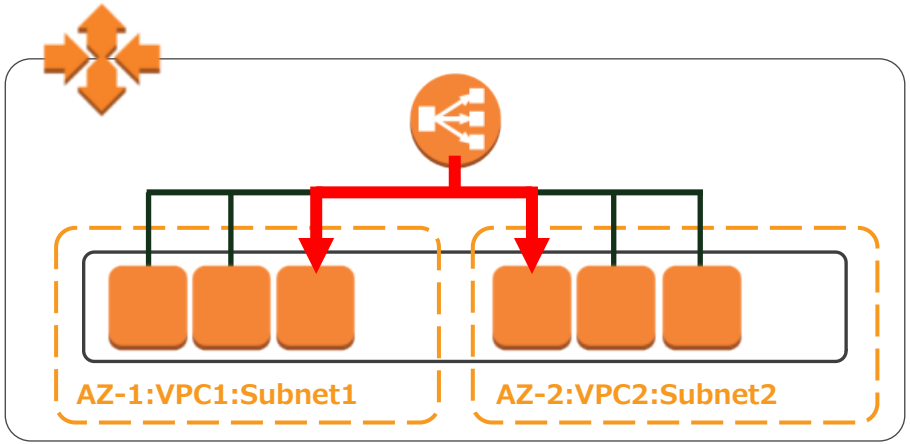
Scaling Policy 2 (simple policy)

Auto Scaling Group

Associate

- AMI, Instance Type,
  IAM Role, Security Group,
  Key Pair, user-data,
  Storage, Spot Price

Launch Configuration



⑥Auto Scalingアクション完了！

- Min:2, Max:6
- AZ, VPC/Subnet, ELB, Tags

- **Alarm:** HighCPUUtilization
- **Actions(Steps):**
  60 < value → add 2 Instances
  80 < value → add 10% of group
  30 > value → remove 2 Instances

Scaling Policy 1 (with steps)

- **Alarm:** HighLaytency
- **Action(only one):**
  300 < value → add 2 Instances

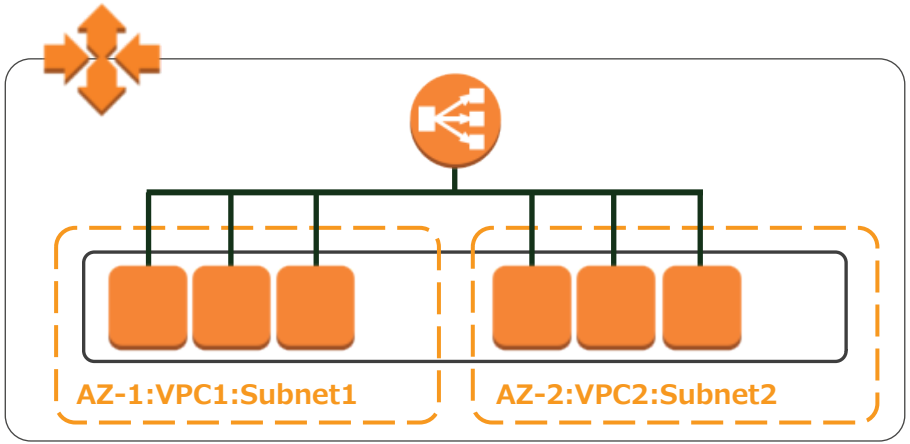
Scaling Policy 2 (simple policy)

Auto Scaling Group

Associate

- AMI, Instance Type,
  IAM Role, Security Group,
  Key Pair, user-data,
  Storage, Spot Price

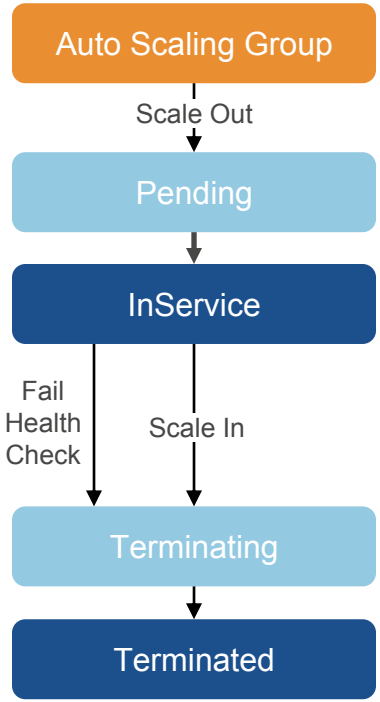
Launch Configuration



アジェンダ

- ❑Auto Scalingのコンセプト
- ❑Auto Scalingの構成要素
- ❑Auto Scaling実行の流れ
- ❑ライフサイクルと高度な制御
- ❑スケーリングの考慮事項
- ❑アーキテクチャーパターン
- ❑Application Auto Scaling
- ❑まとめ

Auto Scalingのライフサイクル



http://docs.aws.amazon.com/ja_jp/autoscaling/latest/userguide/AutoScalingGroupLifecycle.html

ヘルスチェック

ASG内のインスタンスの正常／異常を判断する仕組み

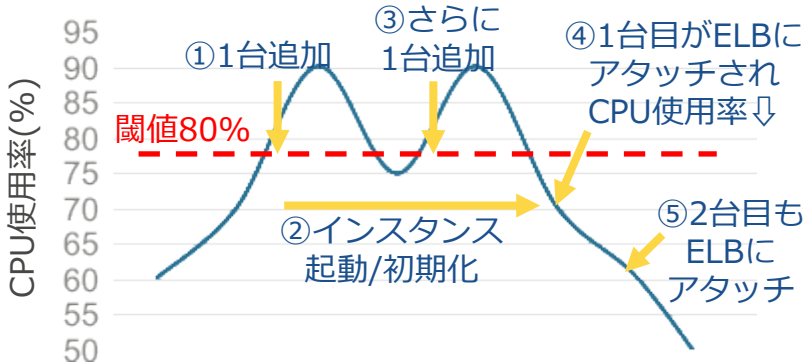
- 2種類のヘルスチェック方法から選択
  1. **EC2ヘルスチェック**
     - インスタンスのステータスがrunning以外の状態を異常と判断
  2. **ELBヘルスチェック**
     - ELBのヘルスチェックを利用
- ヘルスチェックの猶予期間
  - インスタンス起動からヘルスチェックが開始されるまでの時間を指定
  - アプリケーションデプロイの時間を考慮した時間を設定する
- 異常と判断されたインスタンスは**自動的に終了される**

http://docs.aws.amazon.com/ja_jp/autoscaling/latest/userguide/as-maintain-instance-levels.html

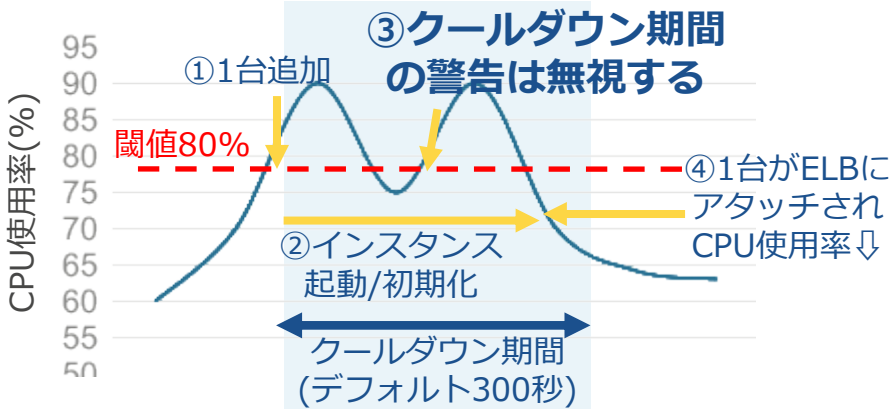
クールダウン

スケーリングアクション実行後、指定した時間内は次のスケーリングアクションを実行しない仕組み

- インスタンス初期化中の無駄なスケーリングを回避するために利用
  - クールダウンがない場合



  - クールダウンがある場合



- シンプルスケーリングポリシーにのみ対応
  - スケジュールベースやステップスケーリングポリシーでは適用されない

http://docs.aws.amazon.com/ja_jp/autoscaling/latest/userguide/Cooldown.html

ターミネーションポリシー

スケールイン時、どのインスタンスから終了するか定義する

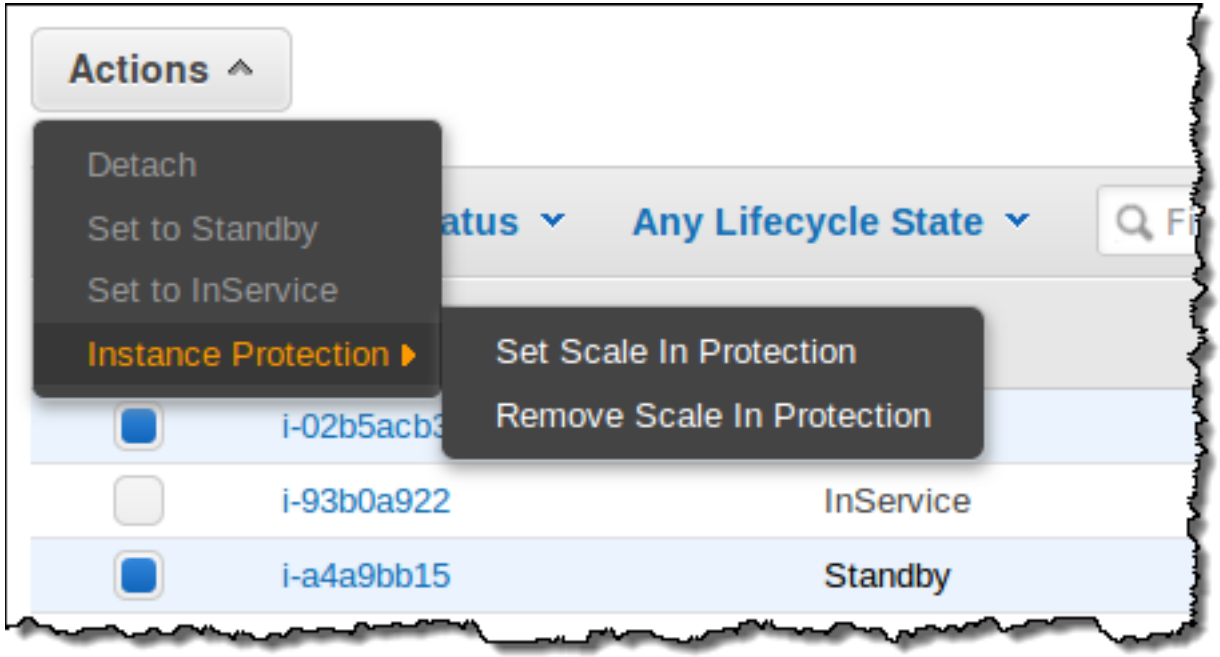
1. **OldestInstance/NewestInstance**
   - ❖ 最も古い/新しい起動時刻のインスタンス
2. **OldestLaunchConfiguration**
   - ❖ 最も古いLaunch Configurationにより起動しているインスタンス
3. **ClosestToNextInstanceHour**
   - ❖ 次の課金が始まるタイミングが最も近いインスタンス
4. **Default**
   - ❖ ②③の順に適用し、複数インスタンスが残ればランダム

❖ インスタンス保護の対象インスタンスは終了対象から除外される

http://docs.aws.amazon.com/ja_jp/autoscaling/latest/userguide/as-instance-termination.html

インスタンス保護

- スケールイン時、任意のインスタンスを削除されないよう保護できる
- 長時間かかるタスクを実行中のインスタンス、Hadoopクラスタのマスターノードなど



http://docs.aws.amazon.com/ja_jp/autoscaling/latest/userguide/as-instance-termination.html#instance-protection

Auto Scaling プロセスの中断と再開

1つ以上のAuto Scalingプロセスを中断/再開する機能

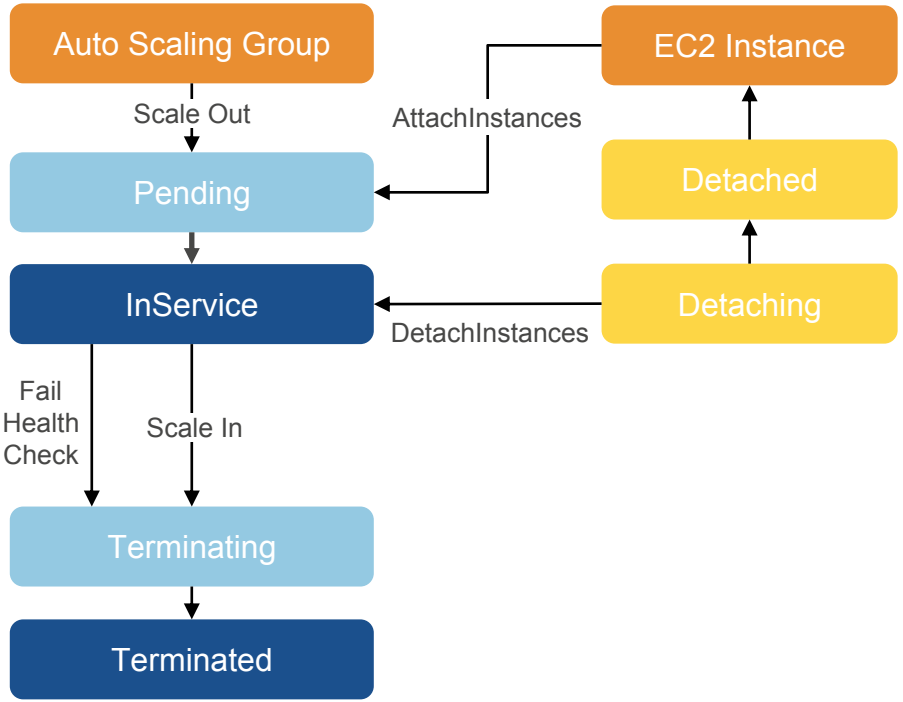
- ASG内の全インスタンスのメンテ/調査時に利用
- 中断/再開できるAuto Scalingプロセス
  - **Launch**：ASGへのインスタンス追加
  - **Terminate**：ASG内のインスタンス終了
  - **HealthCheck**：ASGのヘルスチェック
  - **AZRebalance**：各AZ内のインスタンス数のバランシング
  - **AlarmNotification**：ASGに関連付いたCloudWatchアラーム通知の受信
  - **ScheduledActions**：スケジュールされたアクションの実行
  - **AddToLoadBalancer**：ELBまたはTargetGroupへのインスタンス追加

http://docs.aws.amazon.com/ja_jp/autoscaling/latest/userguide/as-suspend-resume-processes.html

インスタンスのアタッチ/デタッチ

Auto Scaling GroupにEC2インスタンスを付け外しする仕組み

- 一時的なインスタンス単位のメンテ/調査時に利用
  - アタッチ時はDesired Capacityが自動で追加される
  - デタッチ時はDesired Capacityは変更されず、新規インスタンスが追加される
  - ASGに登録しているELBには自動的にアタッチ/デタッチされる

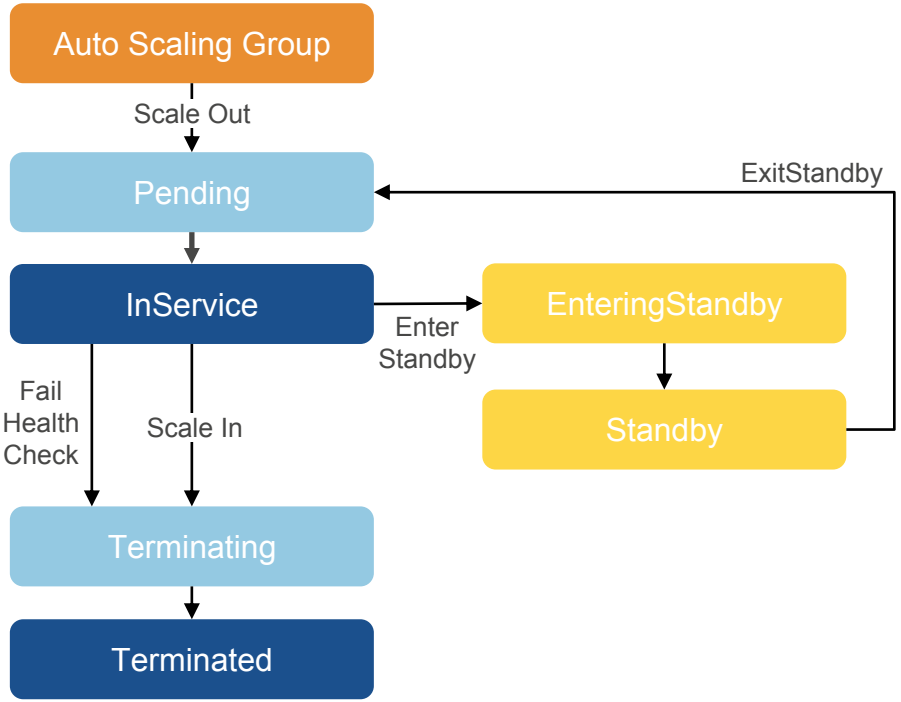


http://docs.aws.amazon.com/ja_jp/autoscaling/latest/userguide/attach-instance-asg.html
http://docs.aws.amazon.com/ja_jp/autoscaling/latest/userguide/detach-instance-asg.html

スタンバイ

ASG内のインスタンスを一時的にスタンバイ状態にする機能

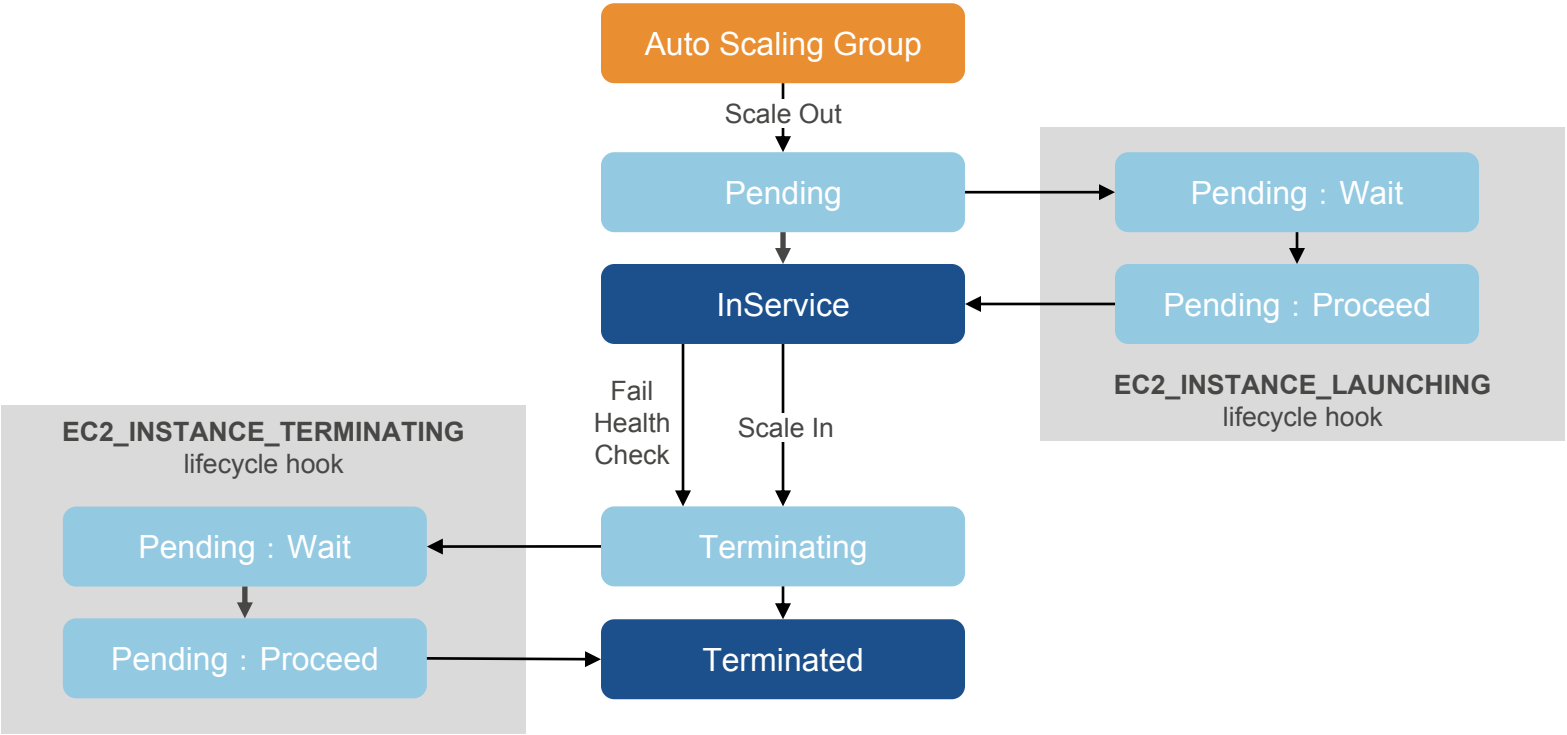
- 一時的なインスタンス単位のメンテ/調査時に利用
  - スタンバイ状態のインスタンスにはアプリトラフィックが流れない
  - ELBへのアタッチ/デタッチやDesired Capacity変更は自動で実施される



http://docs.aws.amazon.com/ja_jp/autoscaling/latest/userguide/as-enter-exit-standby.html

ライフサイクルフック

Auto Scalingによるインスタンス起動/終了処理を待機させ、起動時または終了時にカスタムアクションを実行できる仕組み



http://docs.aws.amazon.com/ja_jp/autoscaling/latest/userguide/lifecycle-hooks.html

ライフサイクルフック

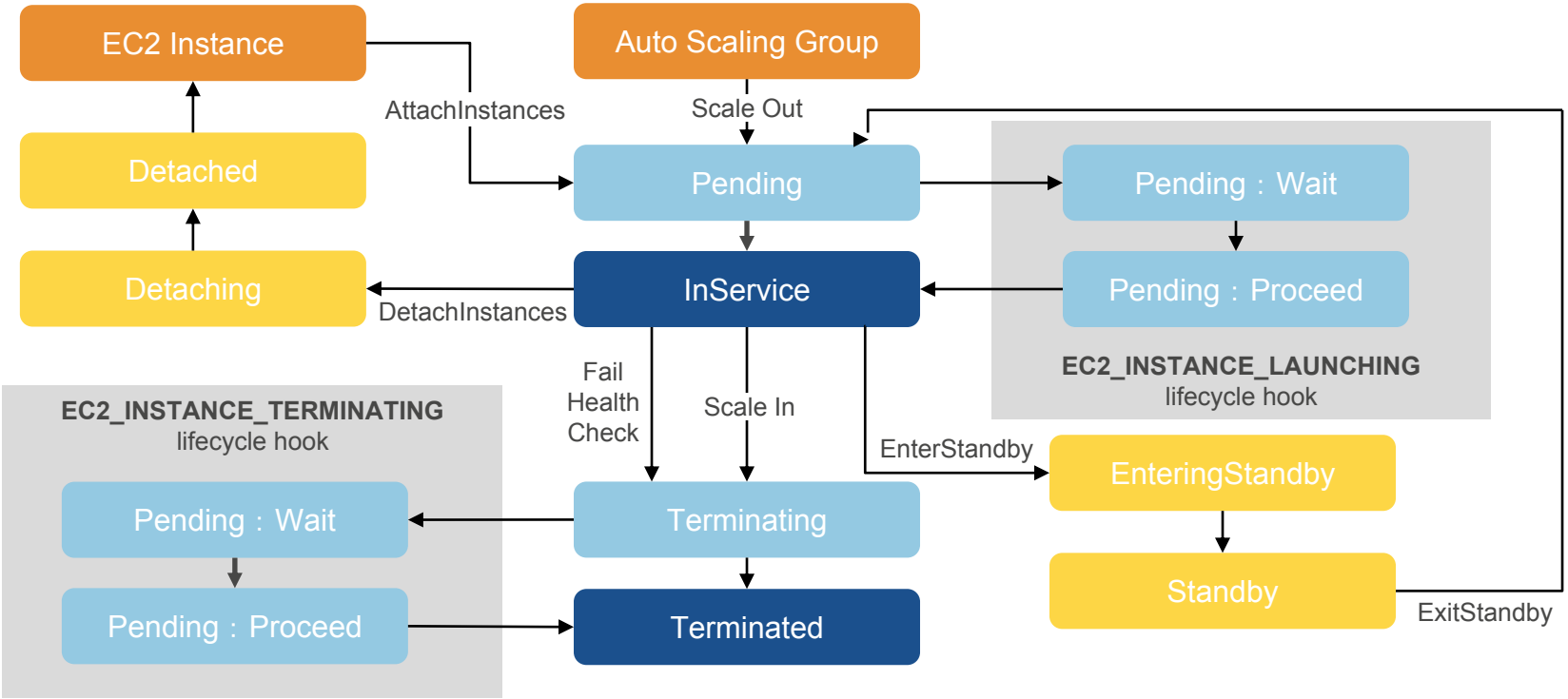
1. インスタンス起動/終了処理を待機させ、メッセージを通知
   - **PutLifecycleHook**オペレーションでライフサイクルフックを定義
   - 待機時間はデフォルト60分、最大48 時間まで延長可能
   - 通知はSNS、SQS、 CloudWatchイベント(CWE)に送られる

2. 待機時間にカスタムアクション（初期化/終了処理）を実行
   - SNS/SQS/CWEのメッセージをトリガーにアクションを実施する

3. ライフサイクルの終了または待機時間延長コマンドを実行
   - カスタムアクションの完了：**CompleteLifecycleAction**オペレーション
   - 待機時間の延長：**RecordLifecycleActionHeartbeat**オペレーション

http://docs.aws.amazon.com/ja_jp/autoscaling/latest/userguide/lifecycle-hooks.html

Auto Scalingのライフサイクル



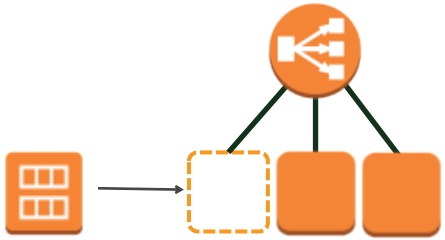
http://docs.aws.amazon.com/ja_jp/autoscaling/latest/userguide/AutoScalingGroupLifecycle.html

アジェンダ

- ❖Auto Scalingのコンセプト
- ❖Auto Scalingの構成要素
- ❖Auto Scaling実行の流れ
- ❖ライフサイクルと高度な制御
- ❖スケーリングの考慮事項
- ❖アーキテクチャーパターン
- ❖Application Auto Scaling
- ❖まとめ

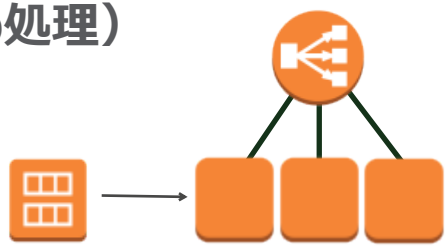
スケールアウト時の初期化処理

1. 設定済みのAMI（ゴールデンイメージ）を用いる

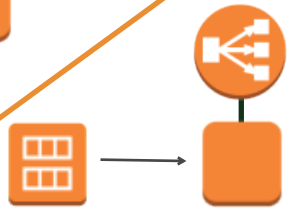


すべてインストール
＆構成済みのAMI

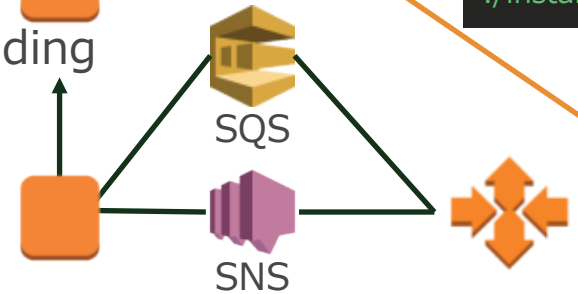
2. user-dataで初期化スクリプトを実行（Bootstrap処理）



```
#!/bin/bash
cd /home/ec2-user;
aws s3 cp s3://aws-install-sh/install
chmod +x ./install;
./install auto;
```



pending



3. Life Cycle Hooksを利用して初期化する

スケールイン時の後処理

- ターミネートに備える
  - スケールインのActionが走ったらいずれかのインスタンスはターミネートする
- サーバーをステートレスにしておく
  - セッション情報はDynamoDBやElastiCache等に外出しする
  - ログファイルはS3にアップロードするなどの実装をする

予測できないスパイクアクセスへの対応

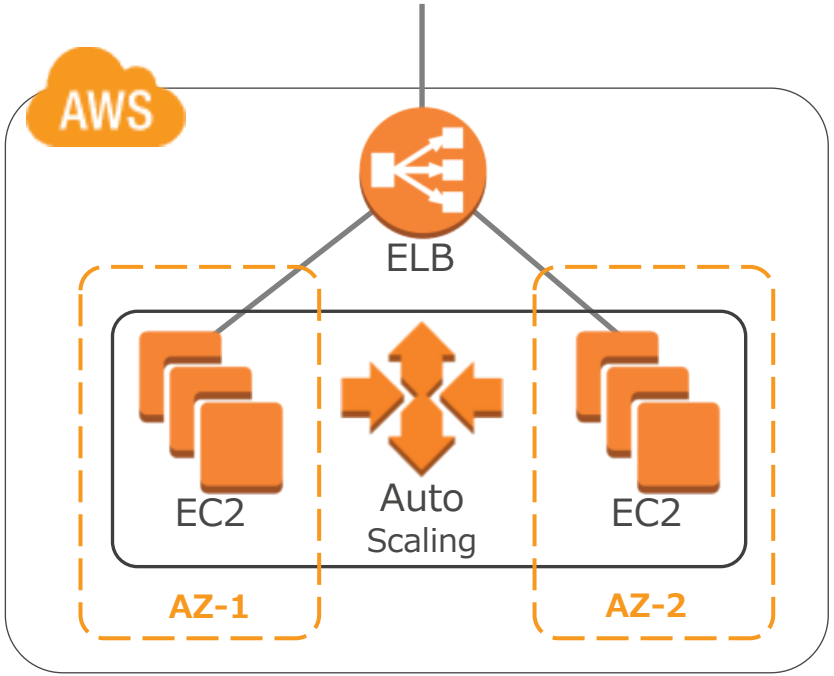
- Auto Scalingは**突発的なスパイクには向いていない**
  - どうしてもインスタンス作成～アプリ起動のリードタイムがかかってしまうため

- スパイクに対応するには
  - CloudFrontなど、あらかじめ大きなキャパシティを持ったAWSサービスに処理をオフロードする
  - スパイクをさばき切ることを諦め、API Gatewayなどを利用してスロットリング機能(処理性能の上限)を設ける
  - 一定以上の負荷を超えたらサイトを静的ページに切替える

アジェンダ

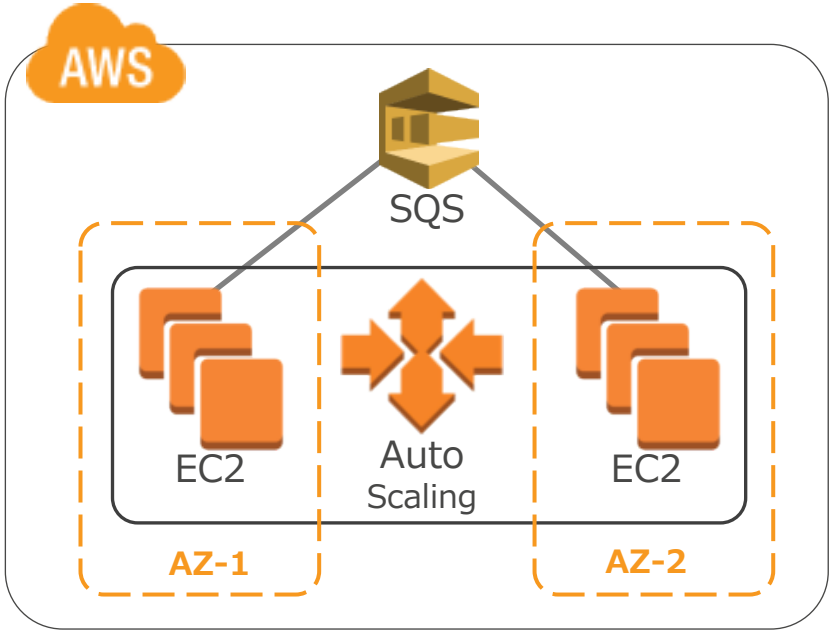
- ❖Auto Scalingのコンセプト
- ❖Auto Scalingの構成要素
- ❖Auto Scaling実行の流れ
- ❖ライフサイクルと高度な制御
- ❖スケーリングの考慮事項
- ❖アーキテクチャーパターン
- ❖Application Auto Scaling
- ❖まとめ

ELB配下のWebサーバーをAuto Scaling



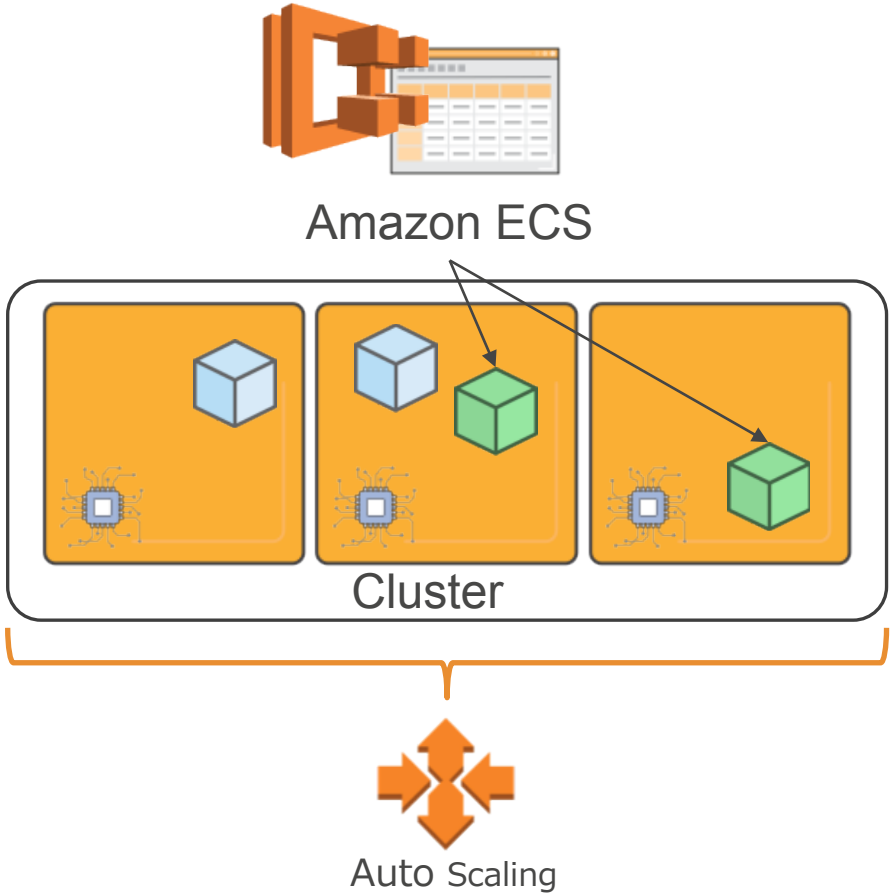
- 特徴
  - Webサーバーを負荷に応じてスケール
  - EC2は複数AZに分散し高可用性を担保
- スケーリングトリガーの例
  - ELBのRequest数
  - ASG内EC2の平均CPU使用率

SQSのジョブを処理するWorkerをAuto Scaling



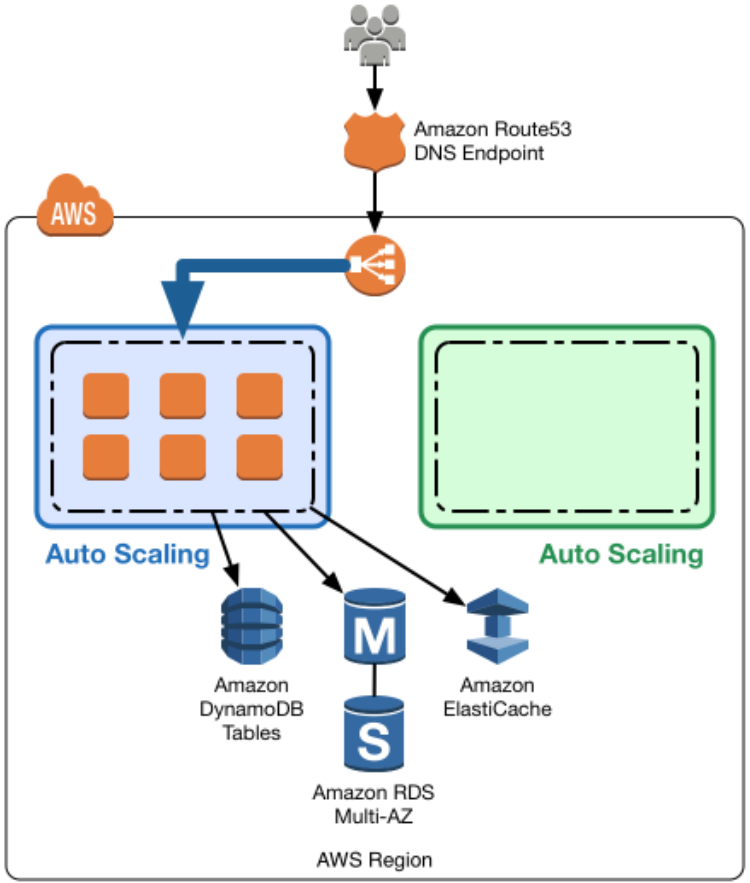
- 特徴
  - Workerを負荷に応じてスケール
  - EC2は複数AZに分散し高可用性を担保
  - スポットインスタンスがマッチするケースが多い
- スケーリングトリガーの例
  - キュー内のメッセージ（ジョブ）数

ECSホストのAuto Scaling



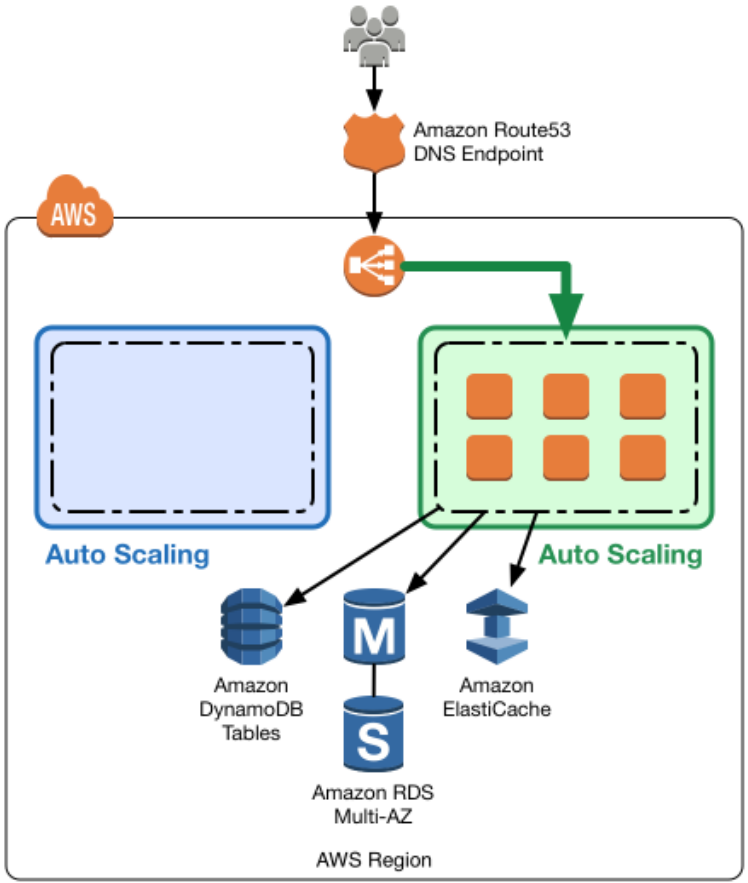
- 特徴
  - ECSクラスタ内のホストを負荷に応じてスケール
  - ホストへのタスク(コンテナ)の配置・管理はECSが実施
- スケーリングトリガーの例
  - ECSホストのCPU使用率
  - ECSホストのメモリ使用率

Blue/Greenデプロイメント



- 特徴
  - アプリケーションのデプロイ時にASGを再作成
  - ELBまたはDNSで環境を切り替え
  - ASG作成にはCloudFormationを利用することで環境構築を自動化可能

Blue/Greenデプロイメント



- 特徴
  - アプリケーションのデプロイ時にASGを再作成
  - ELBまたはDNSで環境を切り替え
  - ASG作成にはCloudFormationを利用することで環境構築を自動化可能

アジェンダ

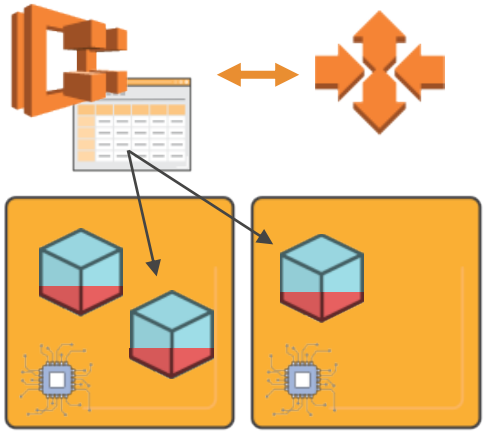
- ❖Auto Scalingのコンセプト
- ❖Auto Scalingの構成要素
- ❖Auto Scaling実行の流れ
- ❖ライフサイクルと高度な制御
- ❖スケーリングの考慮事項
- ❖アーキテクチャーパターン
- ❖Application Auto Scaling
- ❖まとめ

Application Auto Scaling

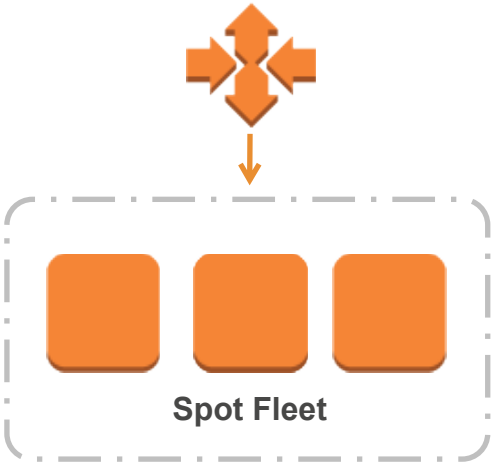
Auto Scalingと似たUXでAWSリソースの自動スケールを実現

❖ サポートされているサービス(2017年3月時点)

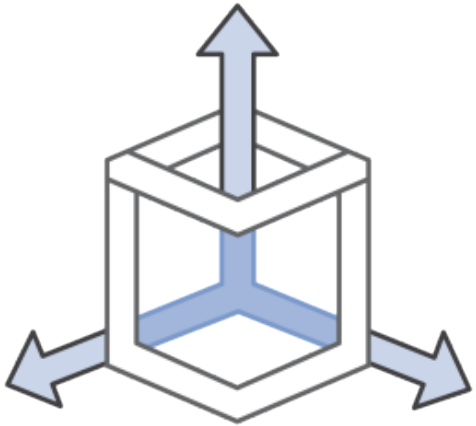
ECSサービス



Spot Fleet



EMR



http://docs.aws.amazon.com/ja_jp/ApplicationAutoScaling/latest/APIReference/Welcome.html
http://docs.aws.amazon.com/ja_jp/AmazonECS/latest/developerguide/service-auto-scaling.html
http://docs.aws.amazon.com/ja_jp/AWSEC2/latest/UserGuide/spot-fleet-automatic-scaling.html
http://docs.aws.amazon.com/ja_jp/emr/latest/ManagementGuide/emr-automatic-scaling.html

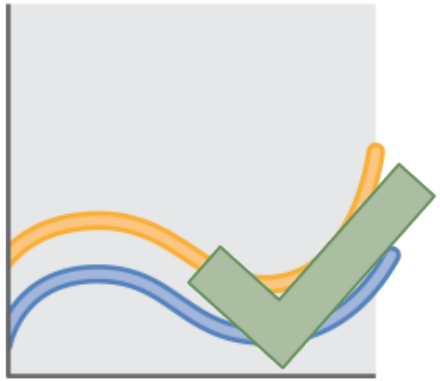
アジェンダ

- ❖Auto Scalingのコンセプト
- ❖Auto Scalingの構成要素
- ❖Auto Scaling実行の流れ
- ❖ライフサイクルと高度な制御
- ❖スケーリングの考慮事項
- ❖アーキテクチャーパターン
- ❖Application Auto Scaling
- ❖まとめ

まとめ：Auto Scalingのベネフィット

柔軟性

必要に応じて自動的に
キャパシティを調整

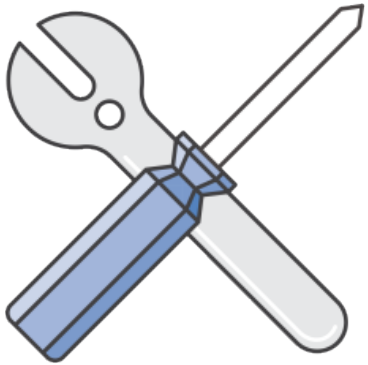


信頼性

インスタンス障害や
AZ障害に対抗

カスタマイズ性

ライフサイクルフック,
Bootstrap



運用コストをかけずにキャパシティの最適化を！

参考資料

- Auto Scaling
  - https://aws.amazon.com/jp/autoscaling/
- よくある質問
  - https://aws.amazon.com/jp/ec2/faqs/#Auto_Scaling
- Auto Scalingドキュメント
  - http://docs.aws.amazon.com/ja_jp/autoscaling/latest/userguide/WhatIsAutoScaling.html
- APIリファレンス
  - http://docs.aws.amazon.com/ja_jp/AutoScaling/latest/APIReference/Welcome.html

Q&A

[導入に関しての問い合わせ]
http://aws.amazon.com/jp/contact-us/aws-sales/

[課金・請求内容、またはアカウントに関するお問い合わせ]
https://aws.amazon.com/jp/contact-us/

オンラインセミナー資料の配置場所

- AWS クラウドサービス活用資料集
  - http://aws.amazon.com/jp/aws-jp-introduction/

- AWS Solutions Architect ブログ
  - 最新の情報、セミナー中のQ&A等が掲載されています
  - http://aws.typepad.com/sajp/

AWSの導入、お問い合わせのご相談

- AWSクラウド導入に関するご質問、お見積り、資料請求をご希望のお客様は、以下のリンクよりお気軽にご相談ください
  https://aws.amazon.com/jp/contact-us/aws-sales/

※「AWS　問い合わせ」で検索してください

amazon
web services