FINATEXT HOLDINGS

CloudWatch Alarm の Terraform Module を作って監視を展開する

2025-02-20

株式会社Finatext

松崎稔矢 @tm8619_pro

自己紹介

松崎稔矢
Toshiya Matsuzaki

株式会社Finatext

職種: バックエンドエンジニア　チームリーダー

趣味: ボウリング🎳/ゲーム🎮/ダーツ🎯/ポーカー🃏/乗馬🐴/お酒🍶🍺🍷　など

技術: Go/AWS/Terraform 競プロ経験あり

好きなAWSサービス: AWS RDS(Aurora)

簡単に監視入れられるように

CloudWatch Alarmをterraformでmodule化した過程と手順の話

以下くらいの背景を要求します

●CloudWatch Metricsの画面は見たことある
●Alarmの存在は知っている

後半出てきますが、あんまり知らなくても大丈夫

●Terraform Moduleについて

インフラ監視って難しいですよね

何が難しいか

- 何を設定するべきか精査する
- しきい値を考える
- 設定を管理・メンテナンスする
- 設定したものがちゃんと監視されるようにする

これをプロダクト開発しつつやる必要がある

今までの経験

プロダクト新規開発します！アプリケーション開発ギリギリ終わってテスト乗り越えてリリース！！

監視の仕組み作る時間ない！！とりあえずDatadog入れとくか！！

- ALBやTargetGroup 5XXエラーを検知
- RDS CPU使用率が高いのも拾うか
  - とりあえず50%とか70%くらい…？

よしなに作っていくと…

- ノイズなアラートがたくさん出る
  - RDSの50%って別に全然大丈夫
- 監視すべき点が抜ける

目的

監視基盤を作って

網羅的にインフラの健康状態が見れる状態にしたい

簡単にAlarmの導入を出来るようにしたい

能動的に監視したい　→　CloudWatch Dashboard

受動的に監視したい　→　CloudWatch Alarm

今回はインフラ障害発生を未然に防いだり、発生時に調査がしやすくなるよう、Alarmの導入にフォーカスします。

CloudWatch Alarmの設定

CloudWatch Alarmの作り方

何のメトリクスに対して
どんなしきい値で設定したらいいの？🤔

AWSが推奨アラームをまとめています！

まずはここを見よう

AWS公式の推奨アラーム

https://docs.aws.amazon.com/ja_jp/AmazonCloudWatch/latest/monitoring/Best_Practice_Recommended_Alarms_AWS_Services.html

Amazon RDS

CPUUtilization

ディメンション: DBInstanceIdentifier

アラームの説明: このアラームは、常に高い CPU 使用率をモニタリングするのに役立ちます。CPU 使用率は、アイドル状態でない時間を測定します。拡張モニタリングまたは Performance Insights を使用して、MariaDB、MySQL、Oracle、および PostgreSQL の CPU 時間 (`guest`、`irq`、`wait`、`nice` など) を最も多く消費している待機時間を確認することを検討してください。次に、CPU を最も多く消費しているクエリを評価します。ワークロードを調整できない場合は、より大きな DB インスタンスクラスへの移行を検討してください。

目的: このアラームは、常に高い CPU 使用率を検出して、非常に長い応答時間やタイムアウトを防ぐために使用されます。CPU 使用率がマイクロバーストしていることを確認したい場合は、アラームの評価時間を短く設定できます。

統計: Average

推奨しきい値: 90.0

しきい値の根拠: CPU 使用率がランダムに増加してもデータベースのパフォーマンスは低下しないかもしれませんが、CPU 使用率が高い状態が続くと、今後のデータベースリクエストが妨げられる可能性があります。データベース全体のワークロードによっては、RDS/Aurora インスタンスの CPU 使用率が高いと、全体的なパフォーマンスが低下する可能性があります。

期間: 60

アラームを発生させるデータポイント数: 5

評価期間数: 5

比較演算子: GREATER_THAN_THRESHOLD

Amazon RDS

CPUUtilization

ディメンション: DBInstanceIdentifier

アラームの説明: このアラームは、常に高い CPU 使用率をモニタリングするのに役立ちます。CPU 使用率は、アイドル状態でない時間を測定します。拡張モニタリングまたは Performance Insights を使用して、MariaDB、MySQL、Oracle、および PostgreSQL の CPU 時間 (`guest`、`irq`、`wait`、`nice` など) を最も多く消費している待機時間を確認することを検討してください。次に、CPU を最も多く消費しているクエリを評価します。ワークロードを調整できない場合は、より大きな DB インスタンスクラスへの移行を検討してください。

目的: このアラームは、常に高い CPU 使用率を検出して、非常に長い応答時間やタイムアウトを防ぐために使用されます。CPU 使用率がマイクロバーストしていることを確認したい場合は、アラームの評価時間を短く設定できます。

統計: Average

推奨しきい値: 90.0

しきい値の根拠: CPU 使用率がランダムに増加してもデータベースのパフォーマンスは低下しないかもしれませんが、CPU 使用率が高い状態が続くと、今後のデータベースリクエストが妨げられる可能性があります。データベース全体のワークロードによっては、RDS/Aurora インスタンスの CPU 使用率が高いと、全体的なパフォーマンスが低下する可能性があります。

期間: 60

アラームを発生させるデータポイント数: 5

評価期間数: 5

比較演算子: GREATER_THAN_THRESHOLD

Amazon RDS

CPUUtilization

ディメンション: DBInstanceIdentifier

アラームの説明: このアラームは、常に高い CPU 使用率をモニタリングするのに役立ちます。CPU 使用率は、アイドル状態でない時間を測定します。拡張モニタリングまたは Performance Insights を使用して、MariaDB、MySQL、Oracle、および PostgreSQL の CPU 時間 (`guest`、`irq`、`wait`、`nice` など) を最も多く消費している待機時間を確認することを検討してください。次に、CPU を最も多く消費しているクエリを評価します。ワークロードを調整できない場合は、より大きな DB インスタンスクラスへの移行を検討してください。

目的: このアラームは、常に高い CPU 使用率を検出して、非常に長い応答時間やタイムアウトを防ぐために使用されます。CPU 使用率がマイクロバーストしていることを確認したい場合は、アラームの評価時間を短く設定できます。

統計: Average

推奨しきい値: 90.0

しきい値の根拠: CPU 使用率がランダムに増加してもデータベースのパフォーマンスは低下しないかもしれませんが、CPU 使用率が高い状態が続くと、今後のデータベースリクエストが妨げられる可能性があります。データベース全体のワークロードによっては、RDS/Aurora インスタンスの CPU 使用率が高いと、全体的なパフォーマンスが低下する可能性があります。

期間: 60

アラームを発生させるデータポイント数: 5

評価期間数: 5

比較演算子: GREATER_THAN_THRESHOLD

Amazon RDS

CPUUtilization

ディメンション: DBInstanceIdentifier

アラームの説明: このアラームは、常に高い CPU 使用率をモニタリングするのに役立ちます。CPU 使用率は、アイドル状態でない時間を測定します。拡張モニタリングまたは Performance Insights を使用して、MariaDB、MySQL、Oracle、および PostgreSQL の CPU 時間 (`guest`、`irq`、`wait`、`nice` など) を最も多く消費している待機時間を確認することを検討してください。次に、CPU を最も多く消費しているクエリを評価します。ワークロードを調整できない場合は、より大きな DB インスタンスクラスへの移行を検討してください。

目的: このアラームは、常に高い CPU 使用率を検出して、非常に長い応答時間やタイムアウトを防ぐために使用されます。CPU 使用率がマイクロバーストしていることを確認したい場合は、アラームの評価時間を短く設定できます。

統計: Average

推奨しきい値: 90.0

しきい値の根拠: CPU 使用率がランダムに増加してもデータベースのパフォーマンスは低下しないかもしれませんが、CPU 使用率が高い状態が続くと、今後のデータベースリクエストが妨げられる可能性があります。データベース全体のワークロードによっては、RDS/Aurora インスタンスの CPU 使用率が高いと、全体的なパフォーマンスが低下する可能性があります。

期間: 60

アラームを発生させるデータポイント数: 5

評価期間数: 5

比較演算子: GREATER_THAN_THRESHOLD

Amazon RDS

CPUUtilization

ディメンション: DBInstanceIdentifier

アラームの説明: このアラームは、常に高い CPU 使用率をモニタリングするのに役立ちます。CPU 使用率は、アイドル状態でない時間を測定します。拡張モニタリングまたは Performance Insights を使用して、MariaDB、MySQL、Oracle、および PostgreSQL の CPU 時間 (`guest`、`irq`、`wait`、`nice` など) を最も多く消費している待機時間を確認することを検討してください。次に、CPU を最も多く消費しているクエリを評価します。ワークロードを調整できない場合は、より大きな DB インスタンスクラスへの移行を検討してください。

目的: このアラームは、常に高い CPU 使用率を検出して、非常に長い応答時間やタイムアウトを防ぐために使用されます。CPU 使用率がマイクロバーストしていることを確認したい場合は、アラームの評価時間を短く設定できます。

統計: Average

推奨しきい値: 90.0

しきい値の根拠: CPU 使用率がランダムに増加してもデータベースのパフォーマンスは低下しないかもしれませんが、CPU 使用率が高い状態が続くと、今後のデータベースリクエストが妨げられる可能性があります。データベース全体のワークロードによっては、RDS/Aurora インスタンスの CPU 使用率が高いと、全体的なパフォーマンスが低下する可能性があります。

期間: 60

アラームを発生させるデータポイント数: 5

評価期間数: 5

比較演算子: GREATER_THAN_THRESHOLD

期間: 60

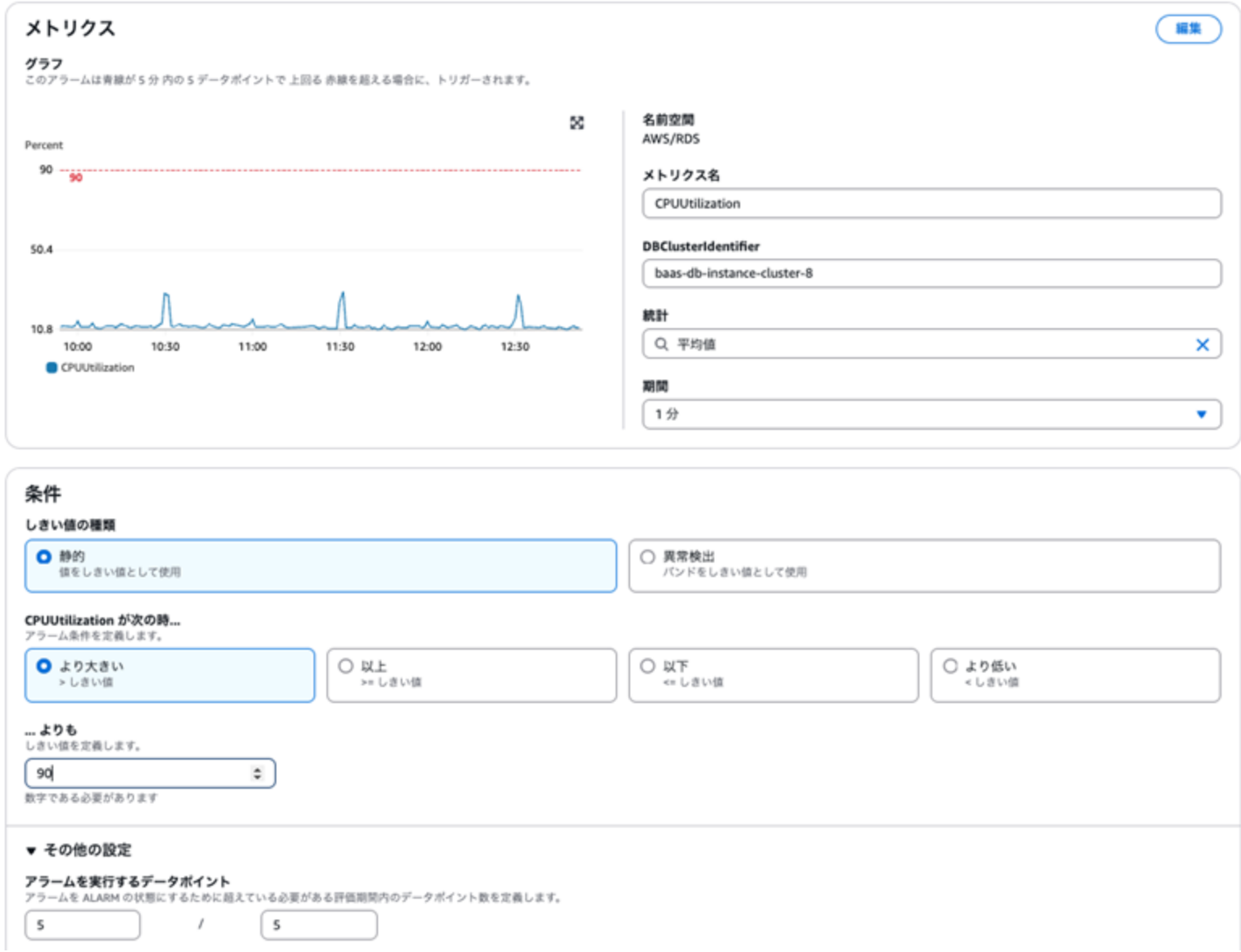
アラームを発生させるデータポイント数: 5

評価期間数: 5

比較演算子: GREATER_THAN_THRESHOLD

統計: Average

推奨しきい値: 90.0



CloudWatch Alarmの作り方

え、これ全部やるの…？ 

CloudWatch Alarmの作り方

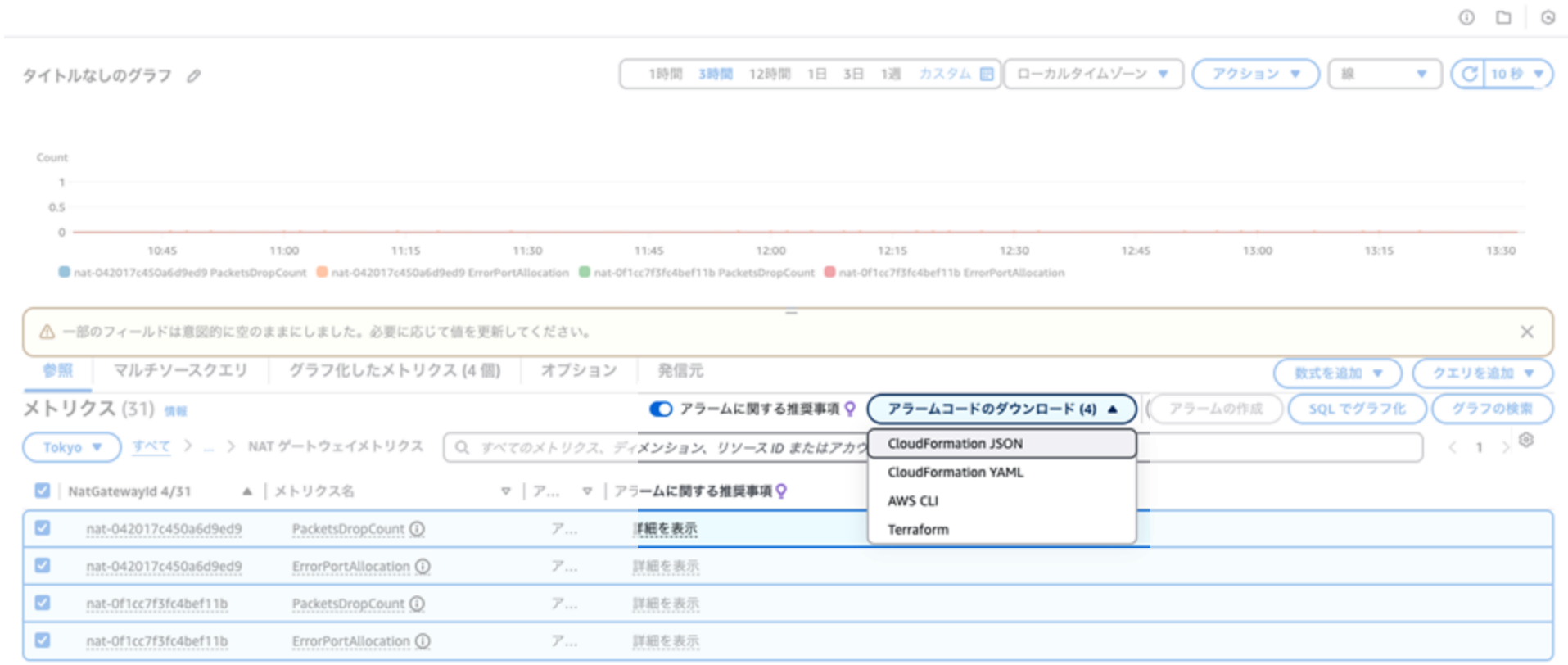
IaCコード、ダウンロードできます！

CloudWatch Alarmの作り方

IaCコード、ダウンロードできます！
しかも複数の形式で！

CloudWatch Alarmの作り方

アラームに関する推奨事項にチェックすると、必要なメトリクスが現れ、コードのダウンロードが可能



CloudWatch Alarmの作り方

こんな感じのコードをダウンロード出来るので、簡単にTerraform化が出来てしまう

```
resource "aws_cloudwatch_metric_alarm" "cloudwatch_alarm_1" {
  # Intent            : "このアラームは、非常に長い応答時間やタイムアウトを防ぐことを目的として、一貫して高い CPU 使用率を検出するために使用されます。CPU 使用率のマイクロバーストを確認したい場合は、アラーム評価時間をより短く設定できます。"
  # Threshold Justification : "CPU の消費がランダムに急増してもデータベースのパフォーマンスが妨げられることはありませんが、CPU 使用率が高い状態が続くと、今後のデータベースリクエストが妨げられる可能性があります。全体的なデータベースのワークロ

  alarm_name          = "AWS/RDS CPUUtilization DBInstanceIdentifier=hogehoge"
  alarm_description   = "このアラームは、CPU 使用率が一貫して高くなっていないかどうかをモニタリングするのに役立ちます。CPU 使用率は非アイドル時間を測定します。MariaDB、MySQL、Oracle、PostgreSQL について、[拡張モニタリング](https://doc
  actions_enabled     = false
  ok_actions          = []
  alarm_actions       = []
  insufficient_data_actions = []
  metric_name         = "CPUUtilization"
  namespace           = "AWS/RDS"
  statistic           = "Average"
  period              = 60
  dimensions = {
    DBInstanceIdentifier = "hogehoge"
  }
  evaluation_periods  = 5
  datapoints_to_alarm = 5
  threshold           = 90
  comparison_operator = "GreaterThanThreshold"
  treat_missing_data  = "breaching"
}
```

Terraform Module化していく

共通化したい部分はどこか？

```
resource "aws_cloudwatch_metric_alarm" "cloudwatch_alarm_1" {
  # Intent            : "このアラームは、非常に長い応答時間やタイムアウトを防ぐことを目的として、一貫して高い CPU 使用率を検出するために使用されます。CPU 使用率のマイクロバーストを確認したい場合は、アラーム評価時間をより短く設定できます。"
  # Threshold Justification : "CPU の消費がランダムに急増してもデータベースのパフォーマンスが妨げられることはありませんが、CPU 使用率が高い状態が続くと、今後のデータベースリクエストが妨げられる可能性があります。全体的なデータベースのワークロ

  alarm_name          = "AWS/RDS CPUUtilization DBInstanceIdentifier=hogehoge"
  alarm_description   = "このアラームは、CPU 使用率が一貫して高くなっていないかどうかをモニタリングするのに役立ちます。CPU 使用率は非アイドル時間を測定します。MariaDB、MySQL、Oracle、PostgreSQL について、[拡張モニタリング](https://doc
  actions_enabled     = false
  ok_actions          = []
  alarm_actions       = []
  insufficient_data_actions = []
  metric_name         = "CPUUtilization"
  namespace           = "AWS/RDS"
  statistic           = "Average"
  period              = 60
  dimensions = {
    DBInstanceIdentifier = "hogehoge"
  }
  evaluation_periods  = 5
  datapoints_to_alarm = 5
  threshold           = 90
  comparison_operator = "GreaterThanThreshold"
  treat_missing_data  = "breaching"
}
```

Terraform Module化していく

今回は2つを変数にして、module化

rds_instance_ids　　:RDSインスタンスID

rds_cpu_utilization_threshold　:CPU利用率しきい値

```
resource "aws_cloudwatch_metric_alarm" "rds_cpu_utilization_alarm" {
  for_each = toset(var.rds_instance_ids)

  alarm_name          = "awsrds-${each.value}-CPUUtilization-TooHigh"
  alarm_description = "過去5分間のCPUUtilizationが${var.rds_cpu_utilization_threshold}%を超え続けています"
  alarm_actions       = [local.sns_topic_alarm_arn]
  metric_name         = "CPUUtilization"
  namespace           = "AWS/RDS"
  statistic           = "Average"
  period              = 60
  dimensions = {
    DBInstanceIdentifier = "${each.value}"
  }
  evaluation_periods  = 5
  datapoints_to_alarm = 5
  threshold           = var.rds_cpu_utilization_threshold
  comparison_operator = "GreaterThanThreshold"
  treat_missing_data  = "breaching"

  actions_enabled = var.is_enabled
}
```

Terraform Module化していく

これを推奨アラーム数の分だけやります！
ここは筋肉💪💪💪

ベースとなるTerraformコードはある
設定値も基本はAWSの推奨値で問題ない

やるべきこと

リソースのIDと設定値を受け取れるようにする
出来ればドキュメントをしっかり読み、このタイミングで設定値を検討する

module化が完了すれば、rds instance1つに対しmoduleを呼べば必要なアラームが全部設定される

```
module "cw_alarm_rds" {
  source = "../../modules/common/template/cw_alarm/rds"

  rds_instance_ids  = module.hoge.rds_instance_ids
  rds_cluster_id    = module.hoge.rds_cluster_id
  rds_instance_type = data.aws_db_instance.rds_instance[module.hoge.rds_instance_ids[0]].db_instance_class
}
```

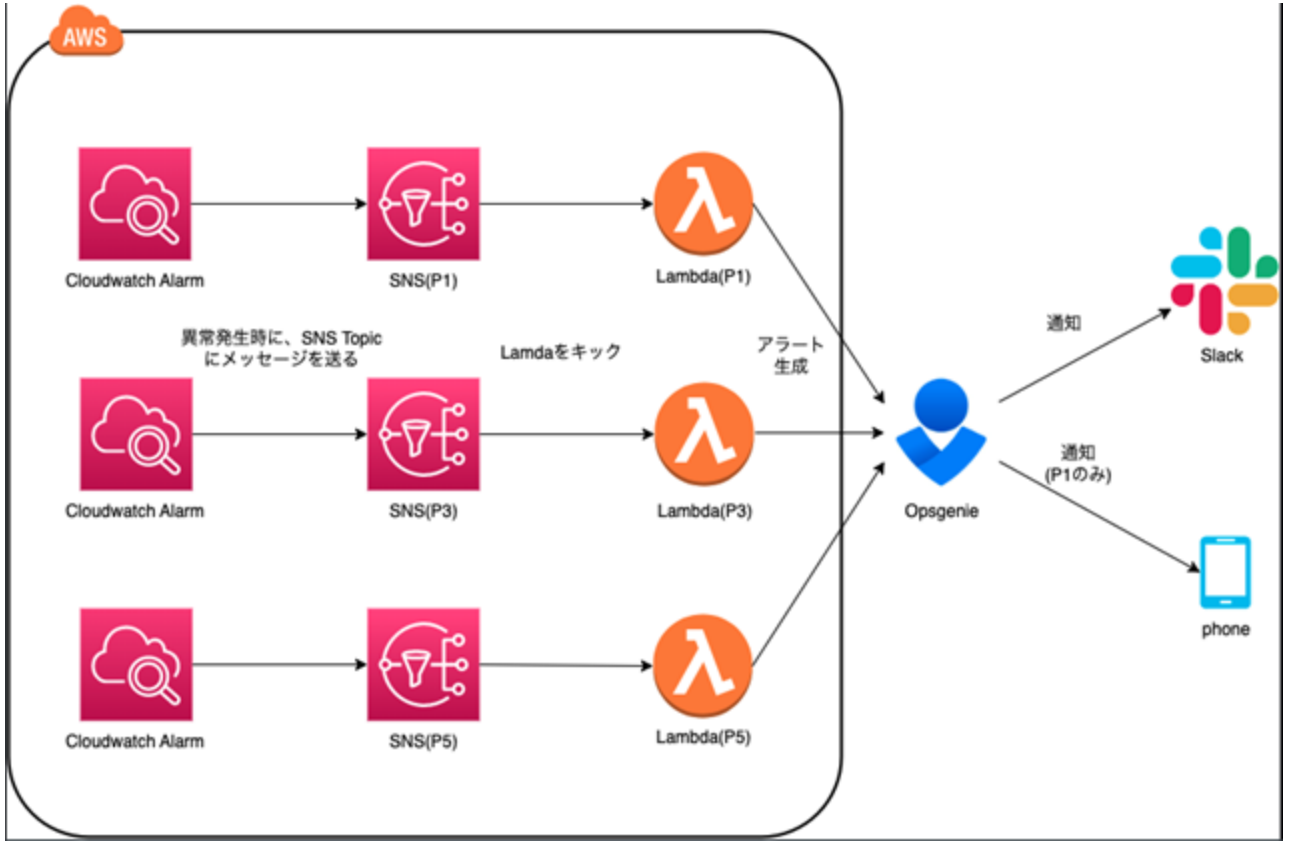
これ書くだけでRDSで必要な監視を全て網羅！

追加機能・実際の運用

opsgenieというアラート管理ツールを利用

アラートを作成すると、優先度に応じてチケット管理や電話を鳴らせるサービス

電話鳴らしたりslack通知を行ったり



障害対応の初動が早まる

アプリケーションエラーが多数発生したとき…

- RDSのCPU使用率アラート
  - ○DB負荷が高い　静観かスケールアウトを検討
- NAT Gatewayのアラート
  - ○AWSの障害　静観なりリージョン切り替えを検討
- インフラ異常は何もない
  - ○アプリケーションロジックのミス・外接エラー

障害を未然に防げる

DBのメモリを食いつぶしたり、保存容量を食いつぶしたりする前に検知可能

本日のまとめ

AWSが推奨する監視設定があるので一度読むと良い

Terraform Module使って設定を展開する事例紹介

FINATEXT

HOLDINGS