

AWS Observability (可観測性) ワークショップ

2021/03/30

シニアエバンジェリスト

亀田 治伸

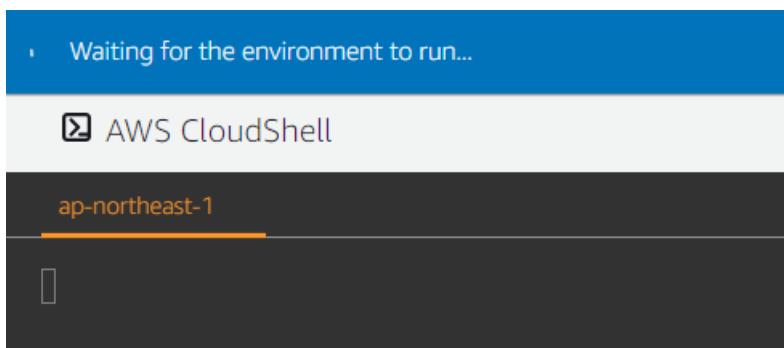
(東京で古い AZ をお持ちのアカウントで作業される方は、バージニア北部リージョンで作業をお願いします)

[環境構築]

1. AWS のマネジメントコンソール右上から CloudShell を開きます



2. 初期化が行われますので、数分間まちます



3. 以下のコマンドをペーストして実行し、Cloud9 を構築します

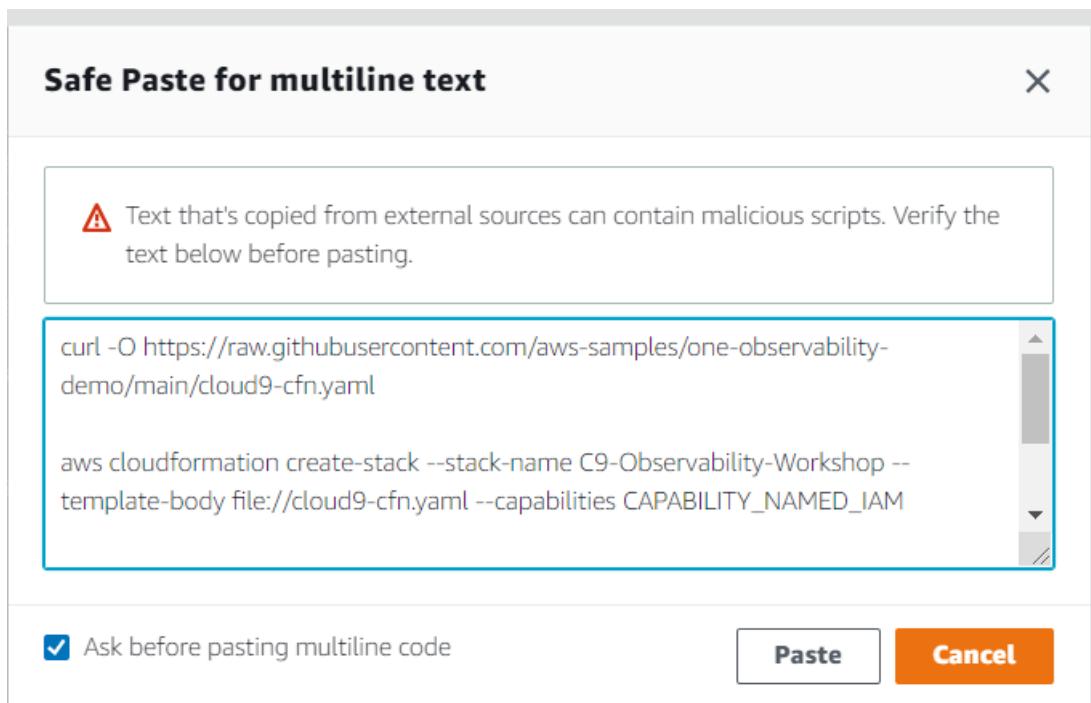
```
curl -O https://raw.githubusercontent.com/aws-samples/one-observability-demo/main/cloud9-cfn.yaml
```

```
aws cloudformation create-stack --stack-name C9-Observability-Workshop --template-body file://cloud9-cfn.yaml --capabilities CAPABILITY_NAMED_IAM
```

```
aws cloudformation wait stack-create-complete --stack-name C9-Observability-Workshop
```

```
echo "Cloud9 Instance is Ready!!"
```

その際以下のダイアログがでますので、「Paste」のボタンをおします



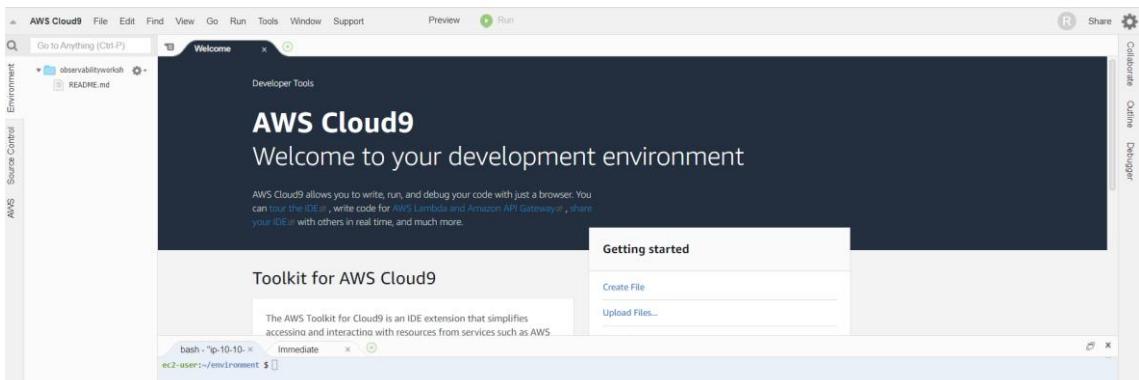
- CloudFormation の画面をブラウザ別タブでひらき、2 つの Stack 実行が完了するまでまちます

CloudFormation > スタック				
スタック (2)		<input type="button" value="C"/>	<input type="button" value="削除"/>	<input type="button" value="更新する"/>
Q スタック名によるフィルター		アクティブ	<input checked="" type="radio"/> ネスト表示	< 1 > ⌂
スタックの名前	ステータス	作成時刻	説明	
aws-cloud9-observabilityworkshop-952cd696a5ba47f5a30cae4fdb4acf6e	CREATE_COMPLETE	2021-03-30 11:09:01 UTC+0900	-	
C9-Observability-Workshop	CREATE_IN_PROGRESS	2021-03-30 11:08:11 UTC+0900	Cloud9 For Observability Workshop	

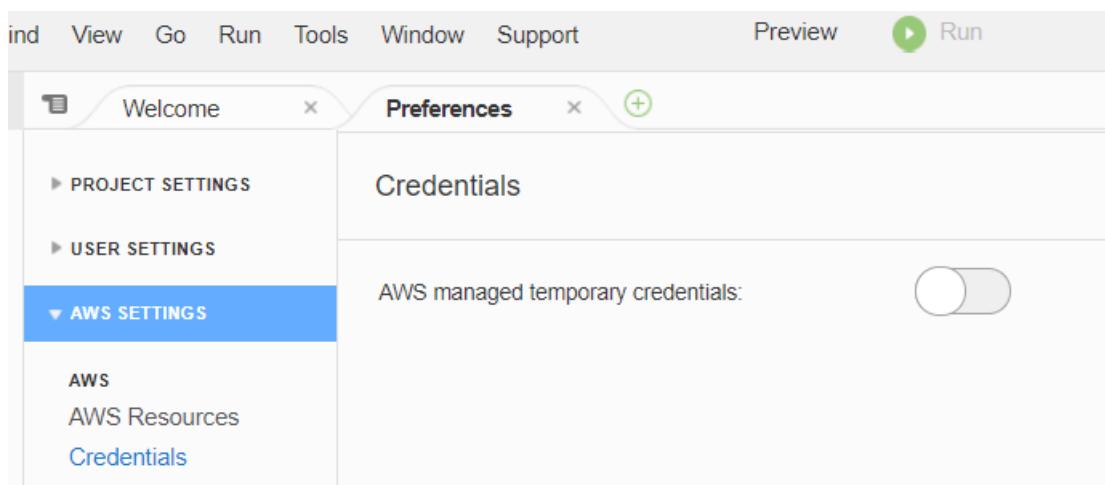
- 完了したらマネージメントコンソールで Cloud9 の画面にいき、「observability workshop」のインスタンスを「Open IDE」のボタンをおして開きます



- Cloud9 が起動したら、画面右上の歯車ボタンをおします



- 左のペインから「AWS SETTINGS」→「Credentials」を選び、つまみをオフにします
し、タブを閉じます



8. 以下のコマンドを、1行づつ順番に実行します

```
rm -vf ${HOME}/.aws/credentials
```

```
curl -sSL https://raw.githubusercontent.com/aws-samples/one-observability-demo/main/PetAdoptions/envsetup.sh | bash -s stable
```

9. 以下のコマンドを纏めてコピペして実行します。(別に1行づつでも問題ありません)

```
export ACCOUNT_ID=$(aws sts get-caller-identity --output text --query Account)
export AWS_REGION=$(curl -s 169.254.169.254/latest/dynamic/instance-identity/document | jq -r '.region')
echo "export ACCOUNT_ID=${ACCOUNT_ID}" | tee -a ~/.bash_profile
echo "export AWS_REGION=${AWS_REGION}" | tee -a ~/.bash_profile
aws configure set default.region ${AWS_REGION}
aws configure get default.region
```

10. 以下のコマンドでディレクトリを移動します

```
cd workshopfiles/one-observability-demo/PetAdoptions/cdk/pet_stack
```

11. 以下のコマンドで npm パッケージをインストールします

```
npm install
```

12. CDK のブートストラップを行います

```
cdk bootstrap
```

13. 以下のコマンドを纏めてコピペしスタックをデプロイします。(10分程度の待ちが発生します)

```
EKS_ADMIN_ARN=$(../../getrole.sh)
```

```
echo "Role ${EKS_ADMIN_ARN} will be part of system:masters group"
```

```
cdk deploy --context admin_role=$EKS_ADMIN_ARN Services --require-approval never  
cdk deploy Applications --require-approval never
```

実行状況は CloudFormation の画面で確認できます。(Cloud9 の画面は改行表示が変になるので不安な方はそちらで確認しましょう)

14. 以下のコマンドを実行し kubeconfig を設定します

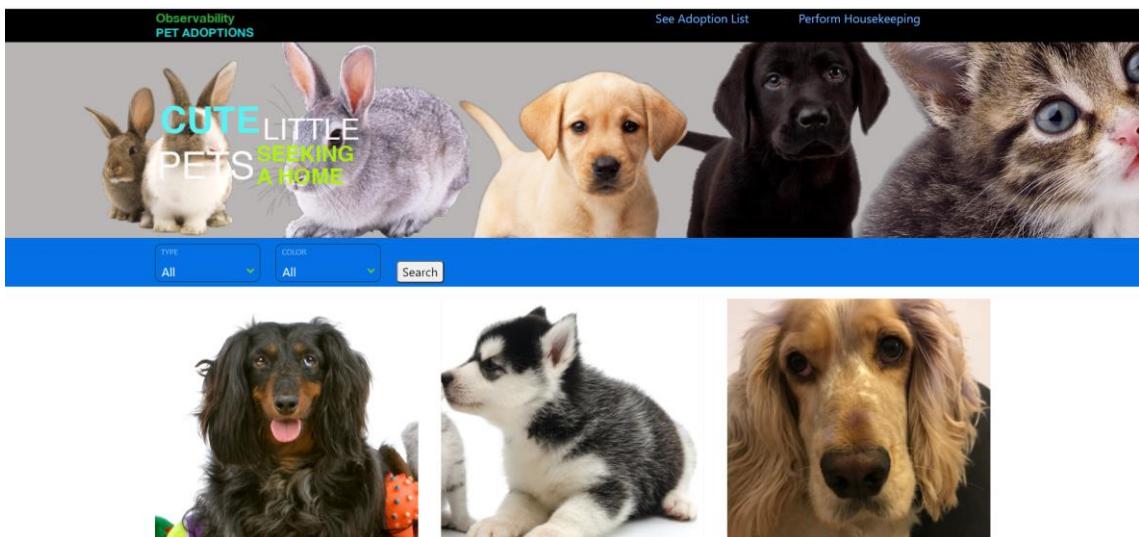
```
aws eks update-kubeconfig --name PetSite --region $AWS_REGION
```

15. 以下のコマンドを実行して、URL を取得します

```
aws ssm get-parameter --name '/petstore/petsiteurl' | jq -r .Parameter.Value
```

16. ブラウザの別タブで URL にアクセスし、以下が表示されれば環境構築完了です。

URL は後で使うのでメモをお願いします



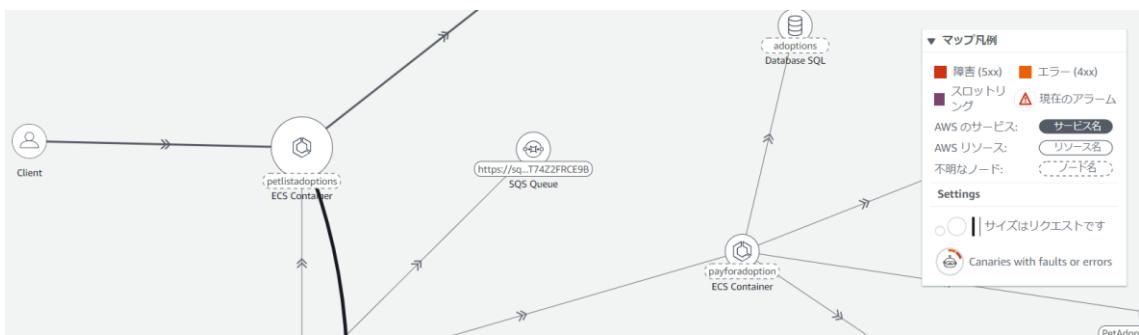
[CloudWatch ServiceLens]

17. 以下の URL を開きます

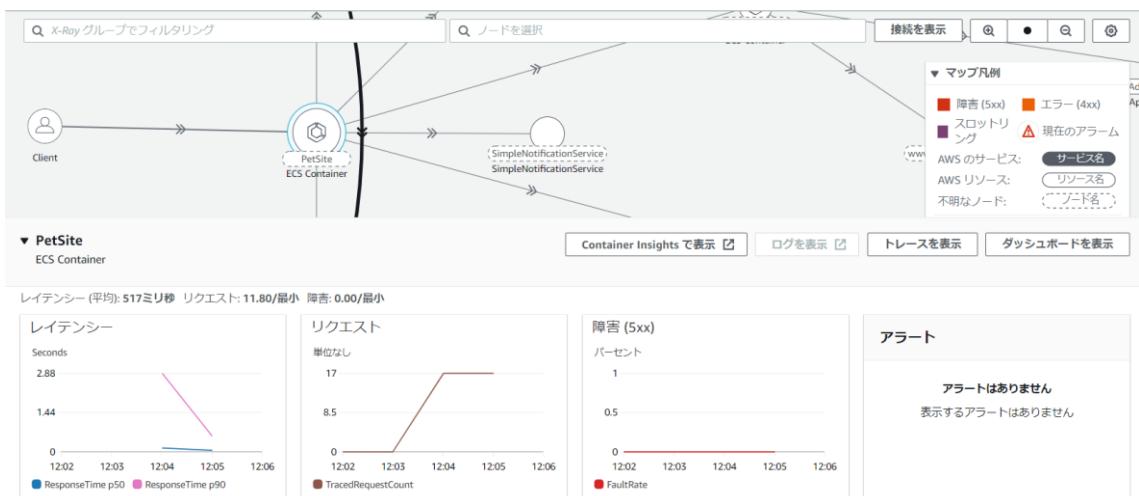
<https://console.aws.amazon.com/cloudwatch/home#servicelens:map?~%28query~%288%29~context~%28timeRange~%28delta~300000%29%29%29>

18. 作成された環境のサービスマップが表示されます。されない場合以下の URL にアクセスしてみてください

<https://ap-northeast-1.console.aws.amazon.com/cloudwatch/home?region=ap-northeast-1#servicelens:map>

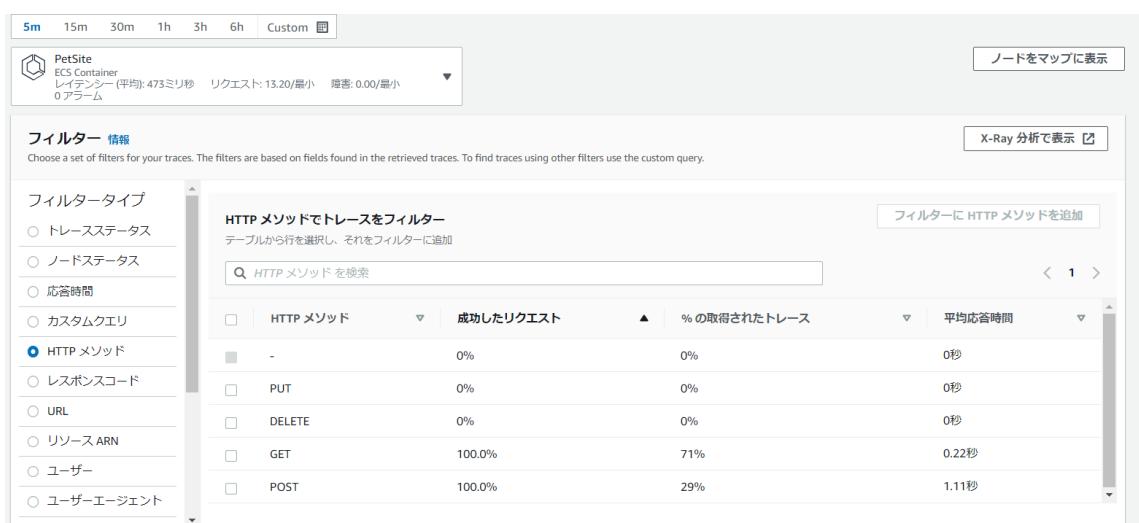


19. Petsite を選択すると画面下に、稼働状況を示すダッシュボードが出てきます。前段の環境構築で、この環境にリクエストをし続けるジェネレーターが生成されているため、自動でリクエストが発生しています



(その他 API Gateway や Lambda 関数など他のノードのパフォーマンスもいろいろ見てみてください)

20. 再度 PetSite を選び、「トレースを表示」のボタンをおします。以下のように様々なトレース情報が出力されます



トレース (66)							詳細を表示
The table only shows up to 1000 most recent traces. Look for any trace ID here.							
<input type="text" value="Q トレースを検索"/>							< 1 2 3 4 5 >
ID	トレースステータス	タイムスタンプ	レスポンスコード	応答時間	HTTP メソッド	URL アドレス	
1-60629513-4a121c8e0 2a1059587daad0f	OK	11.5最小 (2021-0 3-30 12:03:47)	200	0.12秒	GET	http://11.0.195.8:80/metrics	
1-6062951a-291b22dc1 a5dbd7aa296e0b5	OK	11.7最小 (2021-0 3-30 12:03:54)	200	0.466秒	GET	http://11.0.195.8/	
1-60629596-9e8df729 3e8be4fcfb7a7e1	OK	9.7最小 (2021-03 -30 12:05:58)	200	0.006秒	GET	http://11.0.128.176:80/metrics	
1-606295ae-217b53b57 8b4e1d27f969153	OK	9.3最小 (2021-03 -30 12:06:22)	200	0.031秒	GET	http://servi-petsi-acxy4km5il0i-343440114.ap-northeast-1.elb.amazonaws.com/?selectedPetType=puppy&selectedPetColor=brown	
1-60629529-336ce5ec1 2a7824618063f71	OK	11.5最小 (2021-0 3-30 12:04:09)	200	3.056秒	POST	http://servi-petsi-acxy4km5il0i-343440114.ap-northeast-1.elb.amazonaws.com/Adoption/TakeMeHome	
1-606295bd-69c681e54 48ea0ed145996f2	OK	9.0最小 (2021-03 -30 12:06:37)	200	0.053秒	GET	http://servi-petsi-acxy4km5il0i-343440114.ap-northeast-1.elb.amazonaws.com/PetListAdoptions	
1-6062955a-1a10e5041 6010c8a41a8867c	OK	10.7最小 (2021-0 3-30 12:04:58)	200	0.057秒	GET	http://servi-petsi-acxy4km5il0i-343440114.ap-northeast-1.elb.amazonaws.com/PetListAdoptions	
1-606295ae-7e0b0c691 5c4bb576a9bb6a8	OK	9.3最小 (2021-03 -30 12:06:22)	200	0.366秒	POST	http://servi-petsi-acxy4km5il0i-343440114.ap-northeast-1.elb.amazonaws.com/Payment/MakePayment	

21. トレースの欄に例えば[payment]と入力してみてください。関連するトレース情報だけが絞り込まれます。

トレース (66)							詳細を表示
The table only shows up to 1000 most recent traces. Look for any trace ID here.							
<input type="text" value="Q payment"/>							X
ID	トレースステータス	タイムスタンプ	レスポンスコード	応答時間	HTTP メソッド	URL アドレス	
1-606295ae-7e0b0c6915c 4bb576a9bb6a8	OK	10.5最小 (2021-0 3-30 12:06:22)	200	0.366秒	POST	http://servi-petsi-acxy4km5il0i-343440114.ap-northeast-1.elb.amazonaws.com/Payment/MakePayment	
1-606295ea-6dbfe1707bb 19cd971d901b1	OK	9.5最小 (2021-03 -30 12:07:22)	200	0.362秒	POST	http://servi-petsi-acxy4km5il0i-343440114.ap-northeast-1.elb.amazonaws.com/Payment/MakePayment	
1-60629549-6461a3cd79a 4f92b7cfdec3c	OK	12.2最小 (2021-0 3-30 12:04:41)	200	0.396秒	POST	http://servi-petsi-acxy4km5il0i-343440114.ap-northeast-1.elb.amazonaws.com/Payment/MakePayment	
1-60629553-4b13f99301d 7219a1764ef4	OK	12.0最小 (2021-0 3-30 12:04:51)	200	0.346秒	POST	http://servi-petsi-acxy4km5il0i-343440114.ap-northeast-1.elb.amazonaws.com/Payment/MakePayment	
1-60629584-073eb2930e9 255a66a3d0139	OK	11.2最小 (2021-0 3-30 12:05:40)	200	0.575秒	POST	http://servi-petsi-acxy4km5il0i-343440114.ap-northeast-1.elb.amazonaws.com/Payment/MakePayment	
1-60629563-48c41c2941d dfc302d746132	OK	11.7最小 (2021-0 3-30 12:05:07)	200	0.364秒	POST	http://servi-petsi-acxy4km5il0i-343440114.ap-northeast-1.elb.amazonaws.com/Payment/MakePayment	
1-6062952e-1731b91327e 91d6f072d4d6b	OK	12.6最小 (2021-0 3-30 12:04:14)	200	0.374秒	POST	http://servi-petsi-acxy4km5il0i-343440114.ap-northeast-1.elb.amazonaws.com/Payment/MakePayment	

22. 画面上の PetSite のドロップダウンを切り替えることで他のトレースも確認ができるようになります

CloudWatch > トレース

トレース 情報

フィルターを使用して、トレースのテーブルを絞り込むことができます。パフォーマンスの問題を示すトレースを表示します。

5m 15m 30m 1h 3h 6h Custom

サービス	コンテナ	レイテンシー (平均)	リクエスト	障害
PetSite	ECS Container	473ミリ秒	13.20/最小	0.00/最小
SSM		221ミリ秒	0.20/最小	0.00/最小
PetSite	ECS Container	473ミリ秒	13.20/最小	0.00/最小
Services-StepFnlambdastepreadDDBF7497E96-153TGPLT9QU69	Lambda Context	5.03秒	0.40/最小	0.00/最小
payforadoption	ECS Container	295ミリ秒	1.60/最小	0.00/最小
PetSearch	ECS Container	42ミリ秒	162.80/最小	0.00/最小
Services-ddbpetadoption7B7CFEC9-FAYXKJ8QFNS9	DynamoDB Table	14ミリ秒	159.60/最小	0.00/最小

検索:

To find traces using other filters

成功したリクエスト

0%

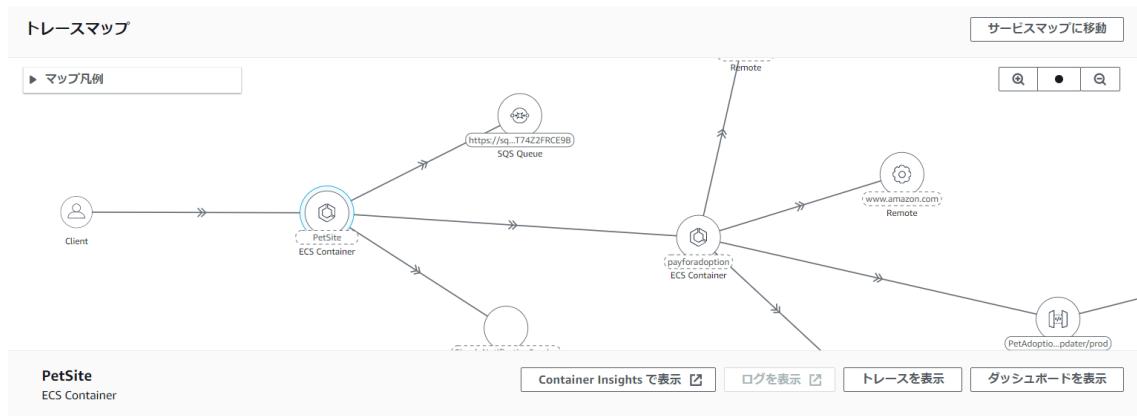
0%

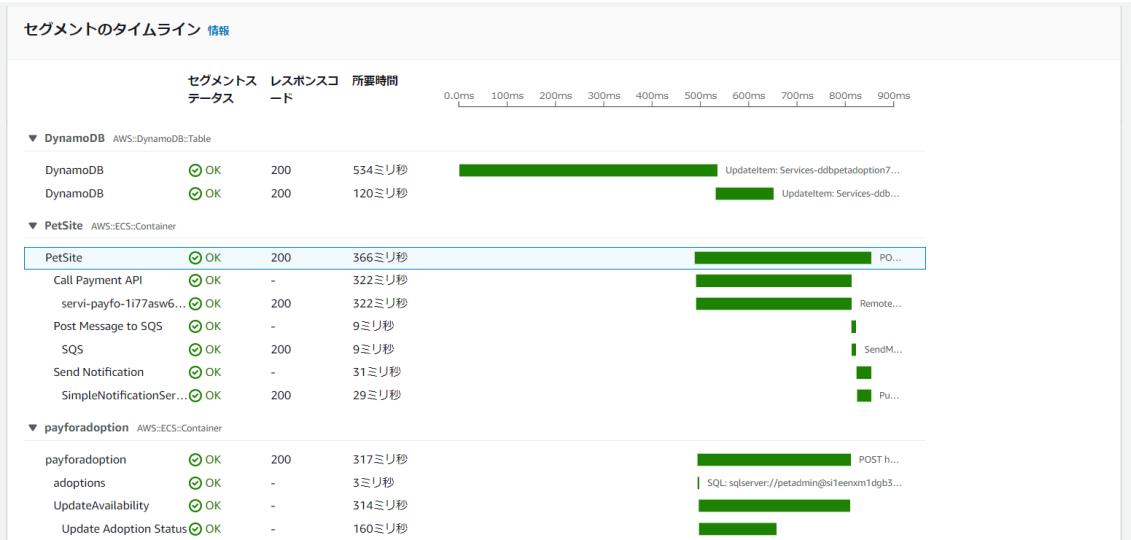
0%

0.00%

POST 100.0%

23. トレース情報を何かクリックすると以下のようなセグメントのタイムラインが表示されます





[X-Ray]

24. 以下の URL を開き、X-Ray の画面にいきます。先ほどと同じ環境の別のマップが表示されます

<https://console.aws.amazon.com/xray/home#/service-map>



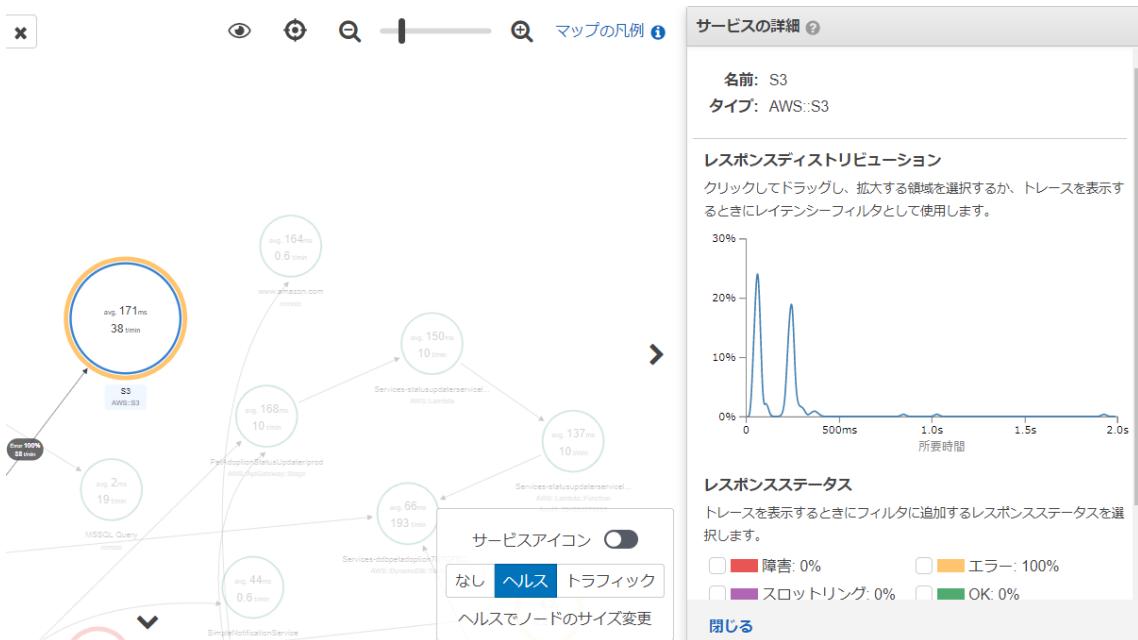
25. 右下のトグルを「トラフィック」から「ヘルス」に変えてみてください



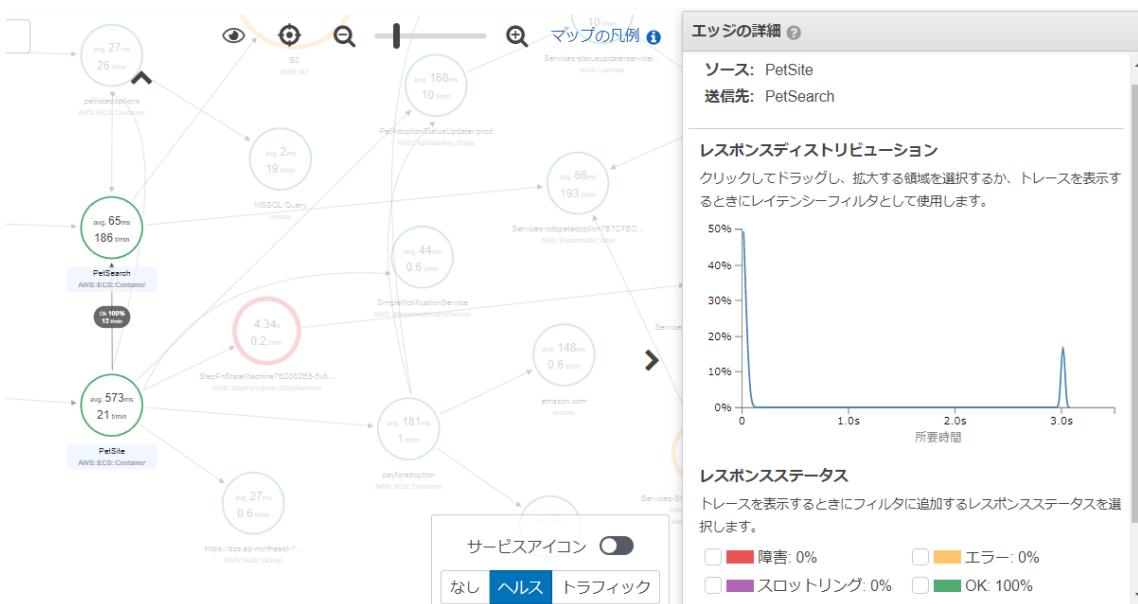
トラフィックの量が円の大きさだったものが、エラーの量が円の大きさに代わり、遅

延をより視覚的に判断できるようになります

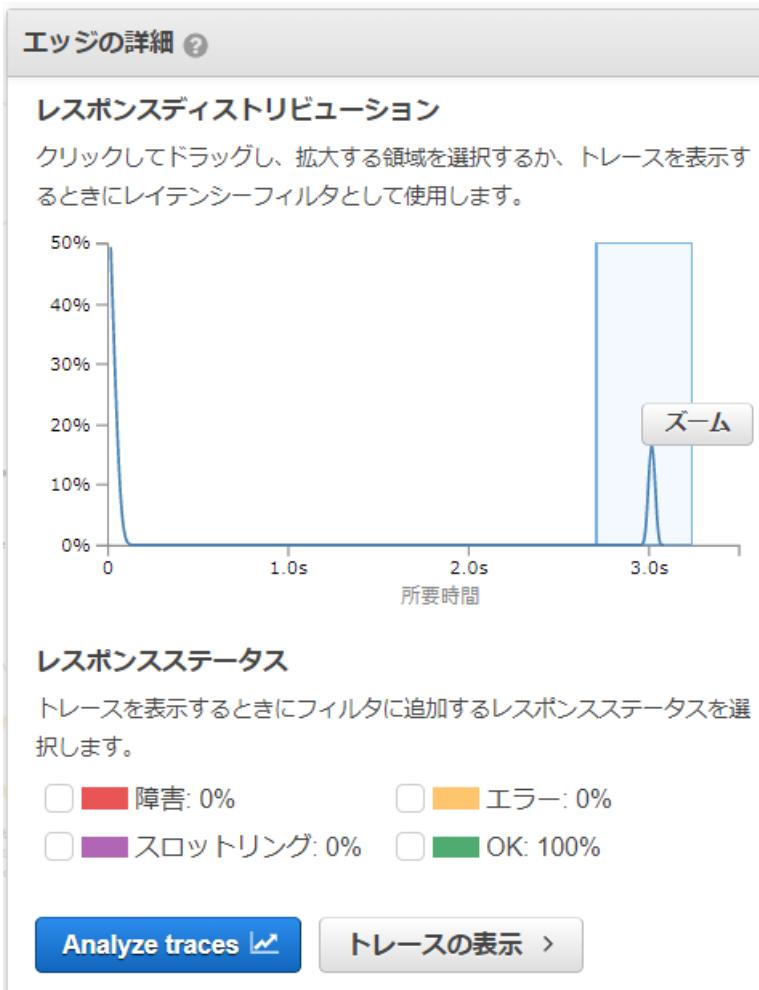
26. 一番円が大きい S3 をクリックしてみて下さい。リクエストが全てエラーとなってることがわかります



27. PeetSite から PetSearch に流れている矢印をクリックしてください。トラフィックの状態が確認できます



28. 表示されたグラフでは、3秒程度処理に時間がかかったトラフィックがあることがわかります。マウスのドラッグで選択し、「トレースの表示」ボタンをおします



29. 調査を行うための個別 トランザクション一覧が表示されます

Trace の概要

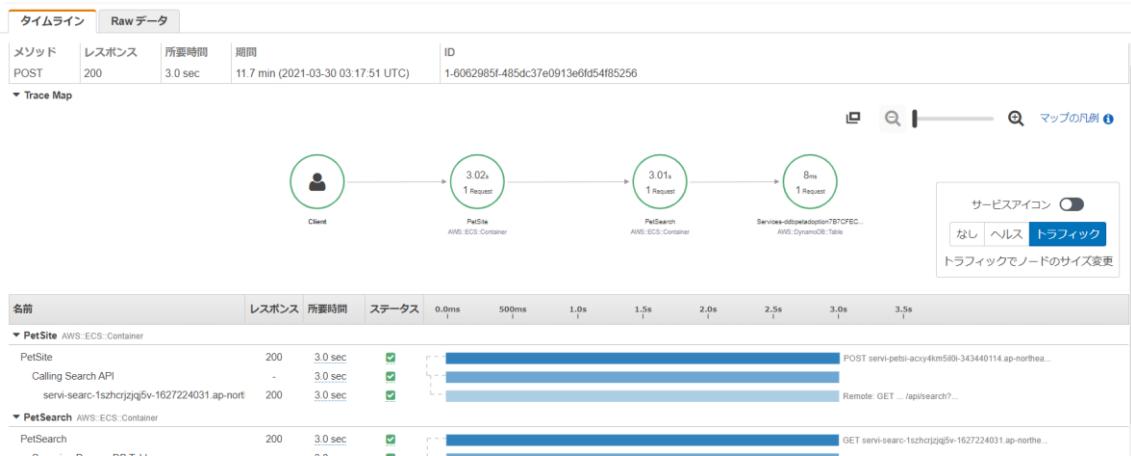
グループ化の条件: URL 完了 100% をスキャンしました (15 件のトレースが見つかりました)

URL	平均レイテンシー	TRACE の %	レスポンス
http://servi-petsi-acxy4km5li0i-343440114.ap-northeast...	3.0 sec	86.67%	13 OK, 0 調整済み, 0 エラー, 0 障害
http://servi-petsi-acxy4km5li0i-343440114.ap-northeast...	3.0 sec	13.33%	2 OK, 0 調整済み, 0 エラー, 0 障害

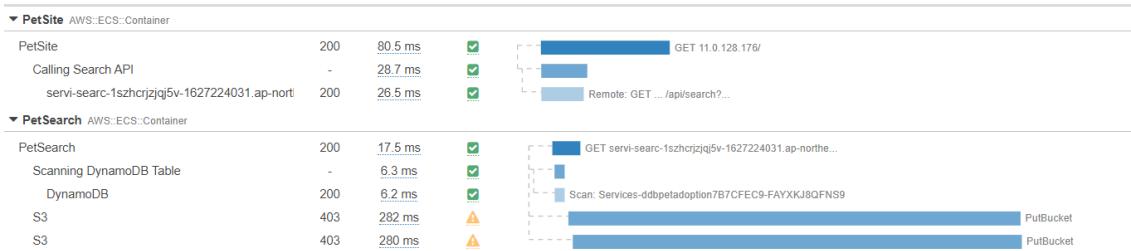
トレースのリスト

ID	経過時間	メソッド	レスポンス	レイテンシー	URL	クライアント IP	注釈
3e6fd54fb5256	11.4 min	POST	200	3.0 sec	http://servi-petsi-acx...	18.181.106.23	1
b12d0a7908a5	8.2 min	POST	200	3.0 sec	http://servi-petsi-acx...	18.181.106.23	1
ca5e51a0deac	8.7 min	POST	200	3.0 sec	http://servi-petsi-acx...	18.181.106.23	1
86f780031570f4	11.9 min	POST	200	3.0 sec	http://servi-petsi.acx...	18.181.106.23	1
cd5ef7ce54750	11.5 min	POST	200	3.0 sec	http://servi-petsi.acx...	18.181.106.23	1
029f4e19f12fc	8.7 min	GET	200	3.0 sec	http://servi-petsi.acx...	18.181.106.23	1

30. ID をクリックするとより詳細な情報が表示されます



31. 先ほど確認した S3 のエラーの詳細を見るため、PetSearch から S3 への矢印をサービスマップの画面でクリックし、同様に「トレースを表示」をおしてください



32. S3 の部分をクリックすると、エラーの詳細が表示されます



概要 **リソース** **注釈** **メタデータ** **例外**

作業ディレクトリ /app
パス --

原因

AmazonS3Exception: Access Denied

```

at Amazon.Runtime.Internal.HttpErrorResponseExceptionHandler.HandleException
at Amazon.Runtime.Internal.ErrorHandler.ProcessException
at Amazon.Runtime.Internal.ErrorHandler+<InvokeAsync>d__5`1.MoveNext
at System.Runtime.ExceptionServices.ExceptionDispatchInfo.Throw
at System.Runtime.CompilerServices.TaskAwaiter.HandleNonSuccessAndDebuggerNotification
at Amazon.Runtime.Internal.CallbackHandler+<InvokeAsync>d__9`1.MoveNext
at System.Runtime.ExceptionServices.ExceptionDispatchInfo.Throw
at System.Runtime.CompilerServices.TaskAwaiter.HandleNonSuccessAndDebuggerNotification
at Amazon.Runtime.Internal.EndpointDiscoveryHandler+<InvokeAsync>d__2`1.MoveNext
at System.Runtime.ExceptionServices.ExceptionDispatchInfo.Throw
at System.Runtime.CompilerServices.TaskAwaiter.HandleNonSuccessAndDebuggerNotification
at Amazon.Runtime.Internal.CredentialsRetriever+<InvokeAsync>d__7`1.MoveNext

```

権限の設定ミスであることがわかります

- 「Service Map」に戻り、 payforadoption をクリックし、 トレースの表示ボタンをおします



- 「/Payment/MakePayment」を含む URL をクリックし、 トレース ID のどれかをクリックします

トレース

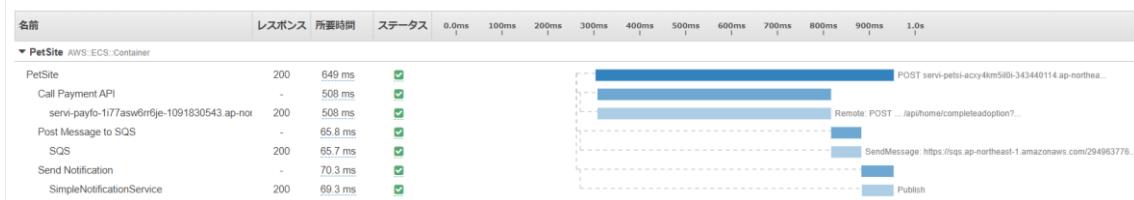
Trace の概要

URL	平均レイテンシー	TRACE の %	レスポンス
http://servi-petsi-acxy4km5l0i-343440114.ap-northeast-1.elb.amazonaws.com/Payment/MakePayment	520 ms	100.00%	3 OK, 0 調整済み, 0 エラー, 0 障害

トレースのリスト

ID	経過時間	メソッド	レスポンス	レイテンシー	URL	クライアント IP	注釈
23321d35803d	18.7 min	POST	200	649 ms	http://servi-petsi-acxy...	18.181.106.23	4
bb99283b4520	18.1 min	POST	200	342 ms	http://servi-petsi-acxy...	18.181.106.23	4
5d9372bb5d18	16.9 min	POST	200	570 ms	http://servi-petsi-acxy...	18.181.106.23	4

35. 以下に表示されている「Call Payment API」をクリックします

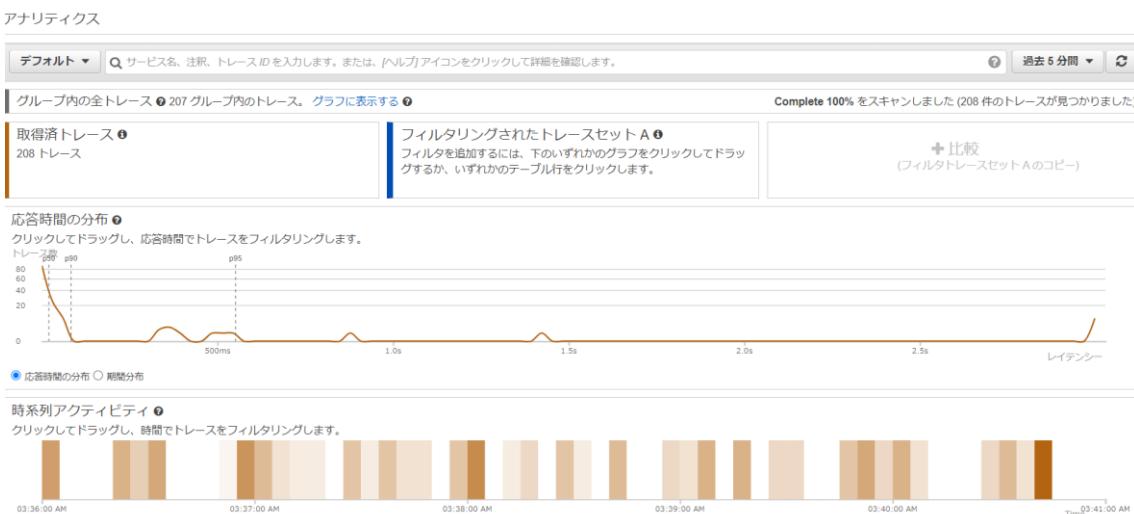


36. API コールのパラメータが表示されます

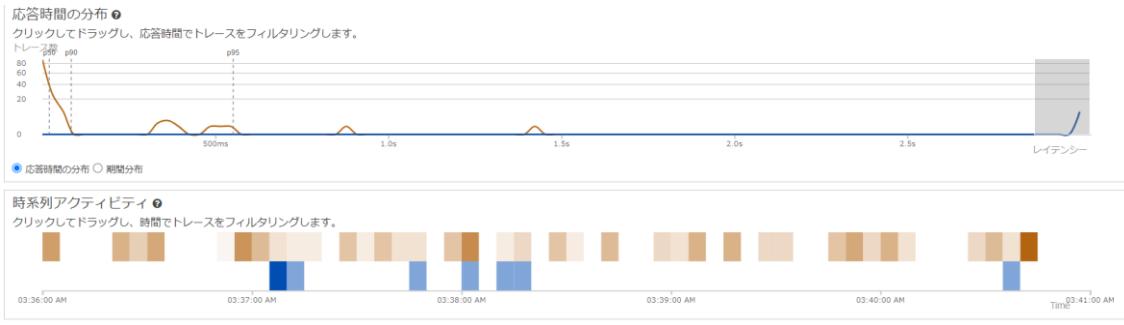
Subsegment - Call Payment API

概要	レスポンス	所要時間	ステータス	0.0ms	100ms	200ms	300ms	400ms	500ms	600ms	700ms	800ms	900ms	1.0s	
PetSite	200	649 ms	<input checked="" type="checkbox"/>												
Call Payment API	-	508 ms	<input checked="" type="checkbox"/>												
servi-paylo-1f77aws6rr6je-1091830543.ap-nor...	200	508 ms	<input checked="" type="checkbox"/>												
Post Message to SQS	-	65.8 ms	<input checked="" type="checkbox"/>												
SQS	200	65.7 ms	<input checked="" type="checkbox"/>												
Send Notification	-	70.3 ms	<input checked="" type="checkbox"/>												
SimpleNotificationService	200	69.3 ms	<input checked="" type="checkbox"/>												

37. 今度は画面左のペインから「アナリティクス」を選びます



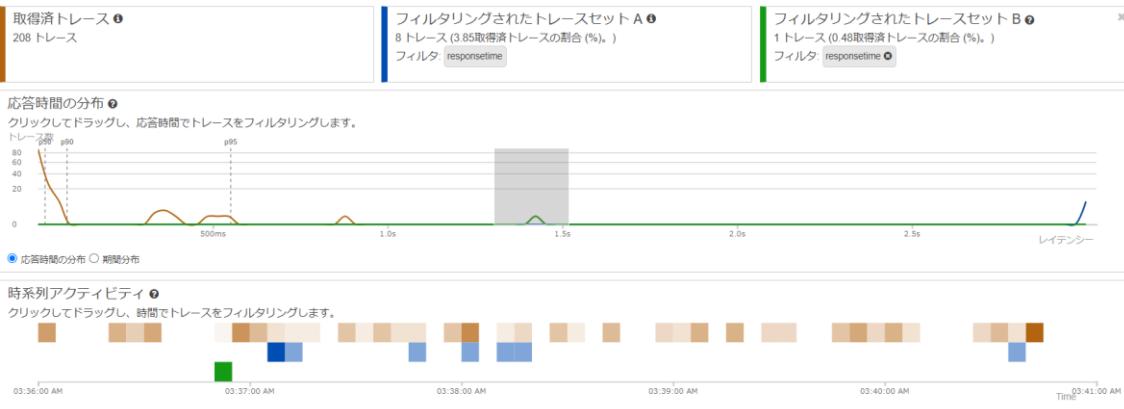
38. 線グラフの右端（3秒以上経過しているレスポンス）当たりをドロップします



39. 画面下部に出力される、以下の情報から、検索において「bunny」を指定したときに遅延が発生することがわかります。(他の検索実行はデータに含まれていないため)

トレースのリスト		USERS	RESPONSE	RESPONSE TIME
URL	http://servi-petsi-acxy4km5l0i-343440114.ap-northeast-1.elb.amazonaws.com/?selectedPetType=bunny&selectedPetColor=bro...	200	3.028	
	http://servi-petsi-acxy4km5l0i-343440114.ap-northeast-1.elb.amazonaws.com/?selectedPetType=bunny&selectedPetColor=bro...	200	3.028	
	http://servi-petsi-acxy4km5l0i-343440114.ap-northeast-1.elb.amazonaws.com/Adoption/TakeMeHome	200	3.02	
	http://servi-petsi-acxy4km5l0i-343440114.ap-northeast-1.elb.amazonaws.com/Adoption/TakeMeHome	200	3.02	
	http://servi-petsi-acxy4km5l0i-343440114.ap-northeast-1.elb.amazonaws.com/Adoption/TakeMeHome	200	3.018	
	http://servi-petsi-acxy4km5l0i-343440114.ap-northeast-1.elb.amazonaws.com/Adoption/TakeMeHome	200	3.017	
	http://servi-petsi-acxy4km5l0i-343440114.ap-northeast-1.elb.amazonaws.com/Adoption/TakeMeHome	200	3.017	
	http://servi-petsi-acxy4km5l0i-343440114.ap-northeast-1.elb.amazonaws.com/Adoption/TakeMeHome	200	3.017	

40. 画面右上の比較を押すと、追加で比較対象の範囲を指定できます。

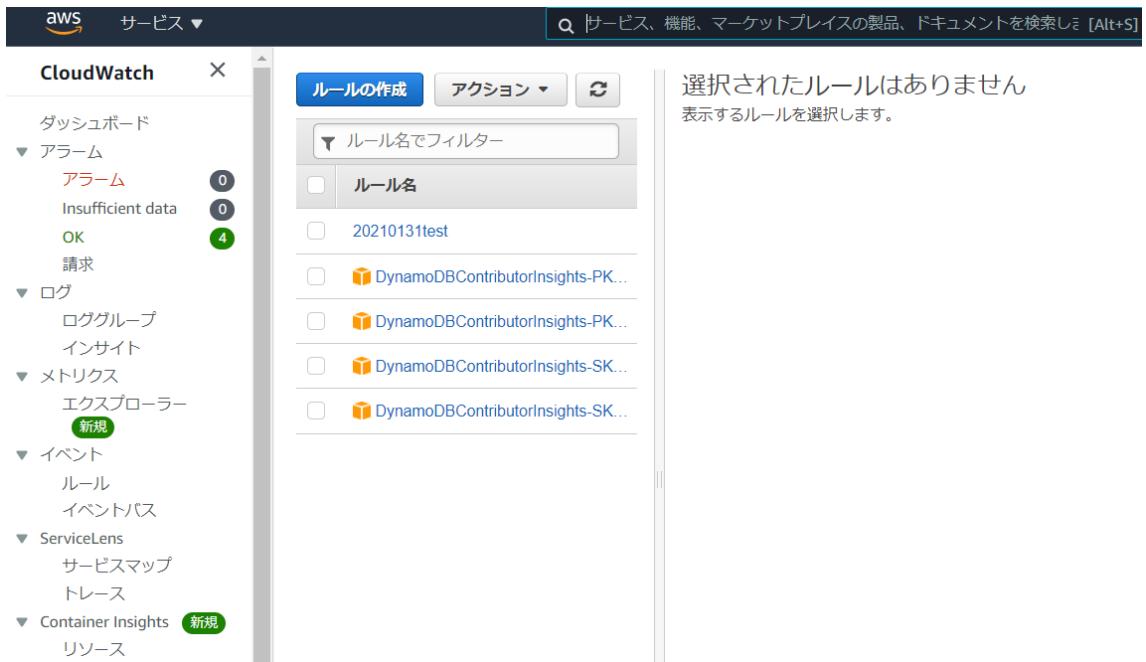


[CloudWatch Contributor Insights]

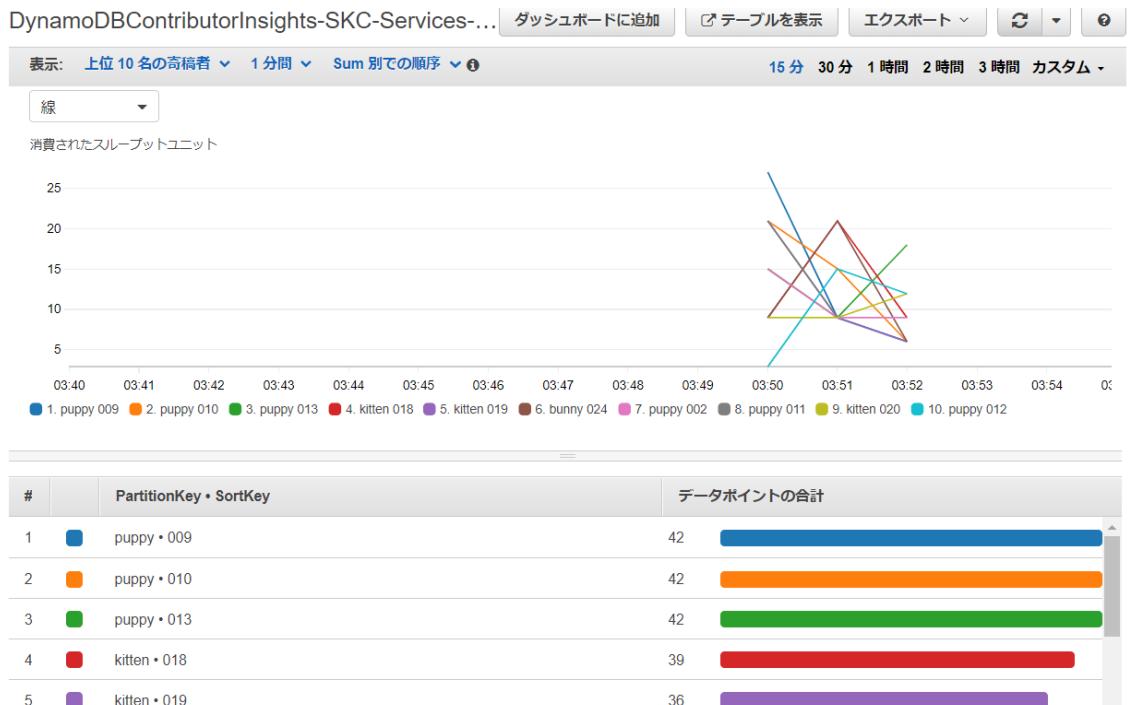
41. Cloud9 のコンソールに戻り以下を実行します

```
DDB_CONTRIB=$(aws ssm get-parameter --name
'/petstore/dynamodbtablename' | jq .Parameter.Value -r)
aws dynamodb update-contributor-insights --table-name $DDB_CONTRIB --
contributor-insights-action ENABLE
```

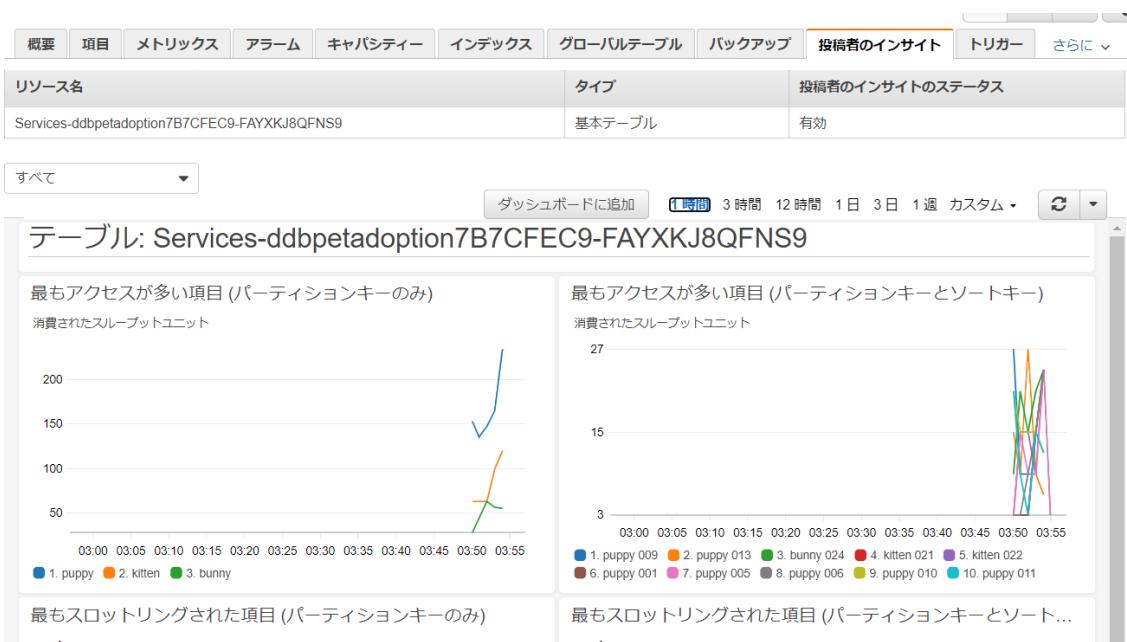
42. これにより、DynamoDB に対して何が負荷を与えてるか、の統計がとれるようになります。数分間待ち、マネージメントコンソールの以下 URL にアクセスします
<https://console.aws.amazon.com/cloudwatch/home?#contributorinsights>:



43. DynamoDB の項目名が表示されていますので、何でもいいので選んでください。しばらく待つと以下のようなグラフ等が表示され、DynamoDB のパフォーマンスチューニングに必要な情報が手に入ります。項目によりデータが早く表示されるものとあまりデータがたまらないものがありますので、複数選んでみてください。



44. DynamoDB のマネージメントコンソールの「投稿者のインサイト」でも同様の情報を取得できます



45. 以下の URL で API Gateway の画面に移動します。同じリージョンであることに注意してください

<https://console.aws.amazon.com/apigateway/main/apis?>

46. 「PetAdoptionStatusUpdater」をクリックします
47. 「ステージ」 → 「Prod」 → 「ログ/トレース」タブを選びます

48. リクエスト/レスポンスをすべてログ、にチェックを付け「変更を保存」ボタンをおします

設定 ログ/トレース ステージ変数 SDK の生成 エクスポート デプロイ履歴 ドキュメント履歴 Canary

ステージのロギングおよびトレース設定を指定します。

CloudWatch 設定

CloudWatch ログを有効化 [?](#)

ログレベル

リクエスト/レスポンスをすべてログ

詳細 CloudWatch メトリクスを有効化 [?](#)

カスタムアクセスのログ記録

アクセスログの有効化

X-Ray トレース [詳細はこちら](#)

X-Ray トレースの有効化 [?](#) X-Ray サンプリングルールの設定

[変更を保存](#)

49. CloudWatch Contributor Insights の画面に戻り 「ルールの作成」 ボタンをおします

aws サービス ▾

サービス、機能

請求

- ▼ ログ
 - ロググループ
 - インサイト
- ▼ メトリクス
 - エクスプローラー
 - 新規**
- ▼ イベント
 - ルール
 - イベントバス
- ▼ ServiceLens
 - サービスマップ
 - トレース
- ▼ Container Insights **新規**
 - リソース
 - パフォーマンスのモニタリング
- ▼ Lambda Insights **新規**
 - パフォーマンスのモニタリング
- ▼ Synthetics
 - Canary
- Contributor Insights**
- 設定

ルールの作成 アクション ▾

ルール名でフィルター

ルール名

20210113test

DynamoDBContributorInsights-PK...

DynamoDBContributorInsights-PK...

DynamoDBContributorInsights-SK...

DynamoDBContributorInsights-SK...

50. 「カスタムルール」を選び、適当な名前をつけます

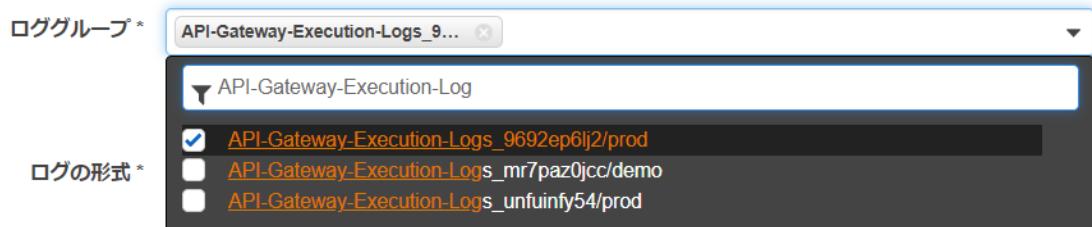
ウィザード 構文

ルールタイプ カスタムルール
 サンプルルール サンプルルールの選択 ▾

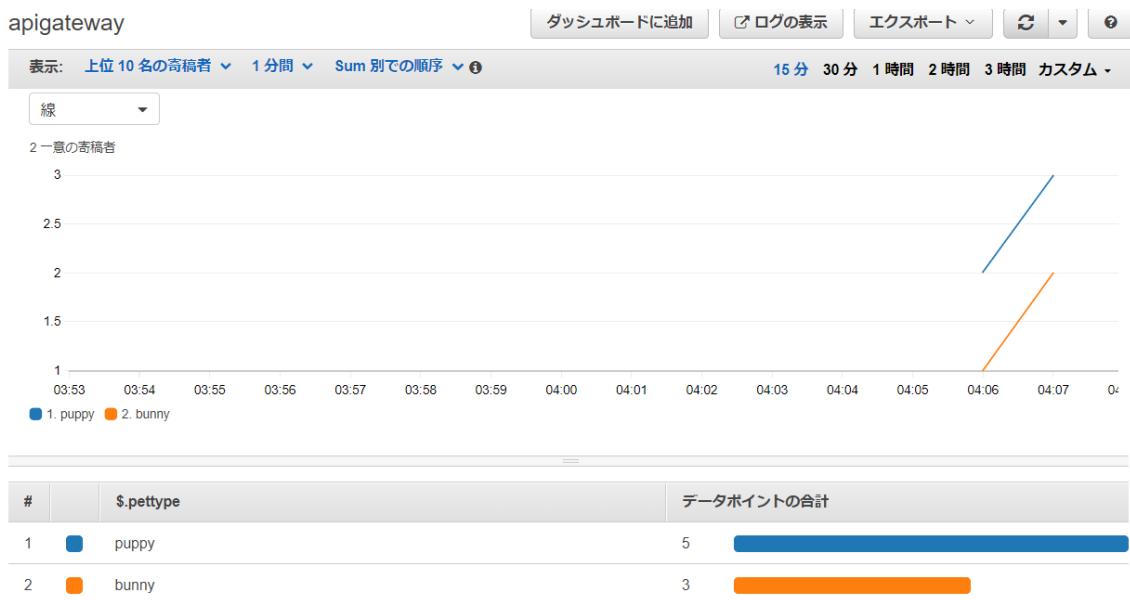
ルール名 *



51. 「ロググループ」から API-Gateway-Execution-Logs で始まるロググループを探します。複数ある場合、API Gateway が持っている URL 名で特定します



52. コントリビューションに pettype と入力し、「作成」ボタンをおします。しばらく待つと以下のようないグラフが表示されます。



[CloudWatch Synthetics]

53. Synthetics が外形監視です。外部 URL を監視するのに用います。以下の URL を開きます。

<https://console.aws.amazon.com/cloudwatch/home?#synthetics:canary/list>



54. 「Canary を作成」をクリックします

The screenshot shows the 'Create Canary' wizard in the AWS CloudWatch Metrics console. It is on the 'Design Diagram' step, which offers several monitoring options:

- 設計図を使用する** (Use a design diagram): Selected. Description: "テンプレートスクリプトから作業する" (Work with template scripts).
- スクリプトをアップロードする** (Upload a script): Description: "独自のスクリプトから作業する" (Work with your own script).
- S3 からインポート** (Import from S3): Description: "S3 から既存のスクリプトを使用する" (Use existing scripts from S3).

Below the diagram, there are six monitoring steps listed:

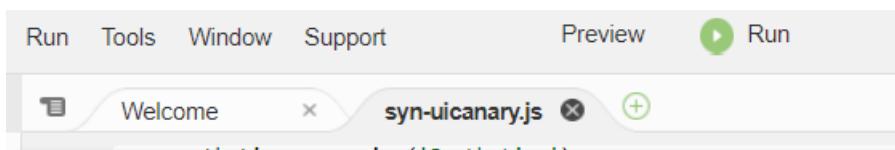
- ハートビートのモニタリング** (Heartbeat monitoring): Description: "1 つの URL で基本的なページのロードを実行します" (Runs a basic page load on one URL).
- API Canary**: Description: "Monitor your APIs as HTTP steps."
- リンク切れチェック** (Link checker): Description: "指定された URL で基本的なウェブクローラーを実行し、アクセスした最初の破損したページを報告します" (Runs a basic web crawler on the specified URL and reports the first broken page accessed).
- Canary Recorder**: Description: "Use the AWS Canary Recorder plugin."
- GUI ワークフロービルダー** (GUI Workflow Builder): Description: "ウェブページで実行するアクションと検証を含む GUI ワークフローを作成します" (Creates a GUI workflow including actions and validations run on a web page).

55. Cloud9 のコンソールに戻り 「/cdk/pet_stack/resources」 に移動します。手順通りであれば「cd resources」の実行で移動できます

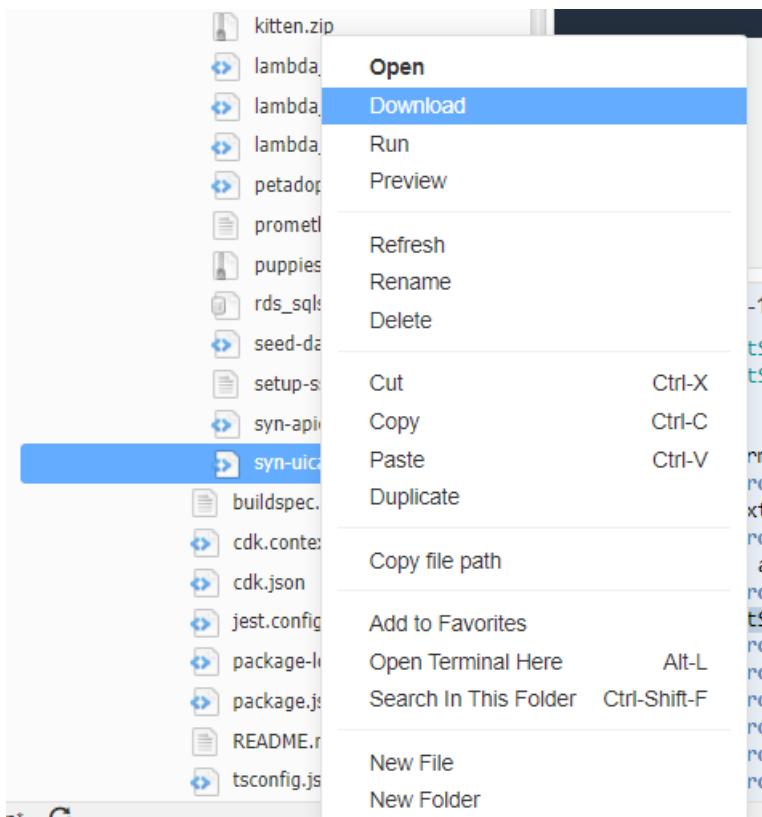
56. 「syn-uicanary.js」を開き 「let url = "<WEBSITE_URL>";」 の URL 部分を前にめもった URL に修正します。以下のようになるはずです。

```
const flowBuilderBlueprint = async function () {
    // INSERT URL here
    //let url = "http://petsite-1081345346.us-east-1.elb.amazonaws.com/";
    let url = "http://Servi-PetSi-ACXY4KM5IL0I-343440114.ap-northeast-1.elb.amazonaws.com";
```

57. タブの右上を押すと、保存ダイアログがでてきますので、保存します。



58. ファイルを右クリックしてダウンロードします



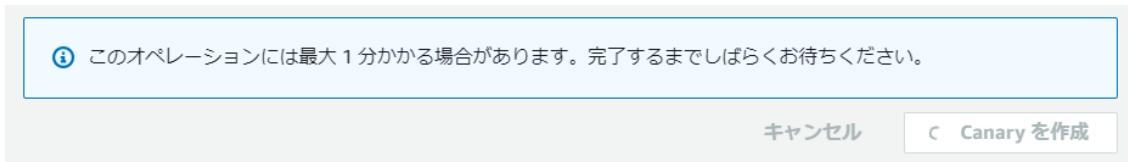
59. CloudWatch の Canary を作成画面に戻り 「スクリプトをアップロードする」 を選び、
適当な名前をつけます

The screenshot shows the 'CloudWatch > Canary を作成' (Create Canary) page. At the top, it says 'Canary を作成' (Create) and '情報' (Information). It asks to select the way to create a Canary. Three options are shown: '設計図を使用する' (Use diagram), 'スクリプトをアップロードする' (Upload script), and 'S3 からインポート' (Import from S3). The second option, 'スクリプトをアップロードする', is highlighted with a blue border. Below this, there's a section titled 'Canary ビルダー' (Canary Builder) with a '名前' (Name) field containing 'test'. A note below the field states: '名前は、21文字までの小文字、数字、ハイフン、またはアンダースコアで構成され、スペースを含めることはできません。' (The name must consist of lowercase letters, numbers, hyphens, or underscores, and cannot contain spaces.)

60. 「Upload Script」ボタンを押し、先ほどダウンロードしたものをアップロードします。そして Lambda ハンドラーに「exports.handler」と設定します



61. 「Canary を作成」ボタンをおします



62. このように外形監視が動作し始めます



63. Canary をクリックすると以下のように、テスト用トランザクションの詳細結果などが表示されます

Steps	スクリーンショット	ログ	HAR ファイル			
Steps Executed (2)						
Step	Step name	Status	Description	Destination URL	Duration	Screen
1	navigateToUrl	Passed	OK	http://servi-petsi-acxy4km5il0i-343440114.ap-northeast-...	1623 ms	2
2	customerActions	Passed	OK	http://servi-petsi-acxy4km5il0i-343440114.ap-northeast-...	7756 ms	2

Steps | **スクリーンショット** | ログ | HAR ファイル

Screenshots (4)

02-navigateToUrl-succeeded
Step passed
http://servi-petsi-acxy4k... ↗

Steps	スクリーンショット	ログ	HAR ファイル
Requests	Status code	Response size	Duration
http://servi-petsi-acxy4km5il0i-343440114.ap-northeast-1.elb.amazonaws.com/			
GET servi-petsi-acxy4km5il0i-343440114.ap-northeast-1.elb.amazonaws.com/	200 OK	97.9 KB	120ms
GET bootstrap.min.css	200 OK	152.1 KB	9.5ms
GET site.css	200 OK	1.3 KB	21.3ms
GET petstyles.css	200 OK	2.9 KB	16.3ms
GET brand.png	200 OK	3.9 KB	17.1ms
GET main_banner.text.png	200 OK	7.9 KB	47.4ms
GET p9.jpg?X-Amz-Expires=300&x-amz-sei	200 OK	88.2 KB	612.7ms
GET p10.jpg?X-Amz-Expires=300&x-amz-sei	200 OK	99 KB	680.4ms
GET p1.jpg?X-Amz-Expires=300&x-amz-sei	200 OK	78.3 KB	641.4ms
GET p11.jpg?X-Amz-Expires=300&x-amz-sei	200 OK	80 KB	641.3ms
GET p3.jpg?X-Amz-Expires=300&x-amz-sei	200 OK	78.3 KB	680.7ms
GET p6.jpg?X-Amz-Expires=300&x-amz-sei	200 OK	64.5 KB	640.5ms
GET jquery.min.js	200 OK	84.9 KB	62.4ms

[CloudWatch Container Insights]

64. Cloud9 のコンソールに戻り以下のコマンドを実行します

```
aws ecs list-clusters | jq '.clusterArns[]' -r
```

```

arn:aws:ecs:ap-northeast-1:294963776963:cluster/Services-PetListAdoptions1706E0DF-bWc59QVtihm
arn:aws:ecs:ap-northeast-1:294963776963:cluster/Services-PetSearchF16438F8-wwuFbLJlcfZP
arn:aws:ecs:ap-northeast-1:294963776963:cluster/Services-PayForAdoptionAAD8027-rq0BiAVaYz6j

```

65. 「Services-PetSeearch…」をコピーし、以下の<cluster-name>と置き換えます

```
aws ecs describe-clusters --clusters <cluster-name> | jq '.clusters[0].settings'
```

66. 正しく実行されれば以下のような値が戻ります

```

ec2-user:~/environment/workshopfiles/one-observability-demo/PetAdoptions/cdk/pet_stack/resources (main) $ aws ecs describe-clusters --clusters Services-PetSearchF16438F8-wwuFbLJlcfZP | jq '.clusters[0].settings'
[
  {
    "name": "containerInsights",
    "value": "enabled"
  }
]
ec2-user:~/environment/workshopfiles/one-observability-demo/PetAdoptions/cdk/pet_stack/resources (main) $ 

```

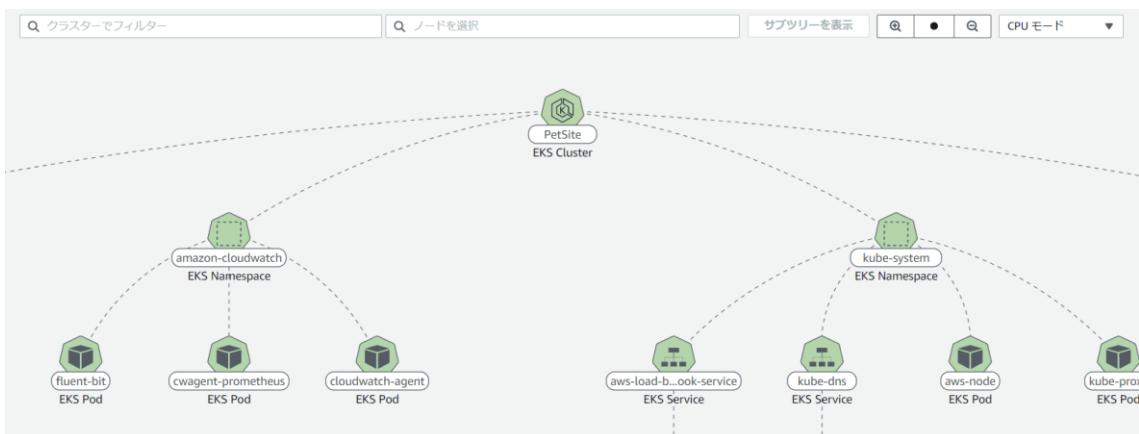
67. 以下の URL で Container Insights の画面に移動します

<https://console.aws.amazon.com/cloudwatch/home?#container-insights:infrastructure>

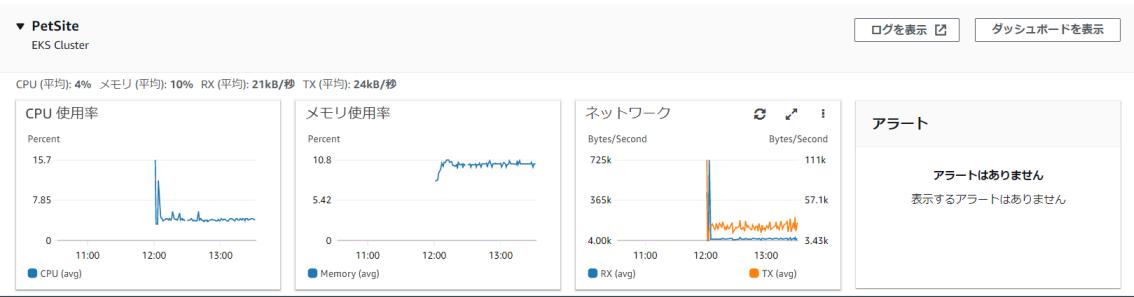
68. 以下のようにクラスターごとの稼働状況一覧が表示されます

リソース (30)		名前	タイプ	Cluster name	アラーム	Prometheus	平均 CPU (%)	平均メモリ (%)
<input type="radio"/>	PetSite	EKS Cluster	PetSite	-	-	-	4.2%	10.3%
<input type="radio"/>	Services-PayForAdoptionAAD8027-rq0BiAVaYz6j	ECS Cluster	Services-PayForAdoptionAAD8027-rq0BiAVaYz6j	-	-	-	0.1%	2.8%
<input type="radio"/>	Services-PetListAdoptions1706E0DF-bWc59QVtihm	ECS Cluster	Services-PetListAdoptions1706E0DF-bWc59QVtihm	-	-	-	0.4%	5.5%
<input type="radio"/>	Services-PetSearchF16438F8-wwuFbLJlcfZP	ECS Cluster	Services-PetSearchF16438F8-wwuFbLJlcfZP	-	-	-	1.8%	8.8%
<input type="radio"/>	Services-listadoptionserviceecsserviceServiceDFDCBE22-wVv6ePwmaN1d	ECS ServiceName	Services-PetListAdoptions1706E0DF-bWc59QVtihm	-	-	-	0.3%	3.3%
<input type="radio"/>	Services-payforadoptionserviceecsserviceService90C1D43F-OxUWcgknRP	ECS ServiceName	Services-PayForAdoptionAAD8027-rq0BiAVaYz6j	-	-	-	0.1%	2.8%
<input type="radio"/>	Services-searchserviceecsserviceService7FE869C-jDYqAUTIJ9FP	ECS ServiceName	Services-PetSearchF16438F8-wwuFbLJlcfZP	-	-	-	1.8%	8.8%

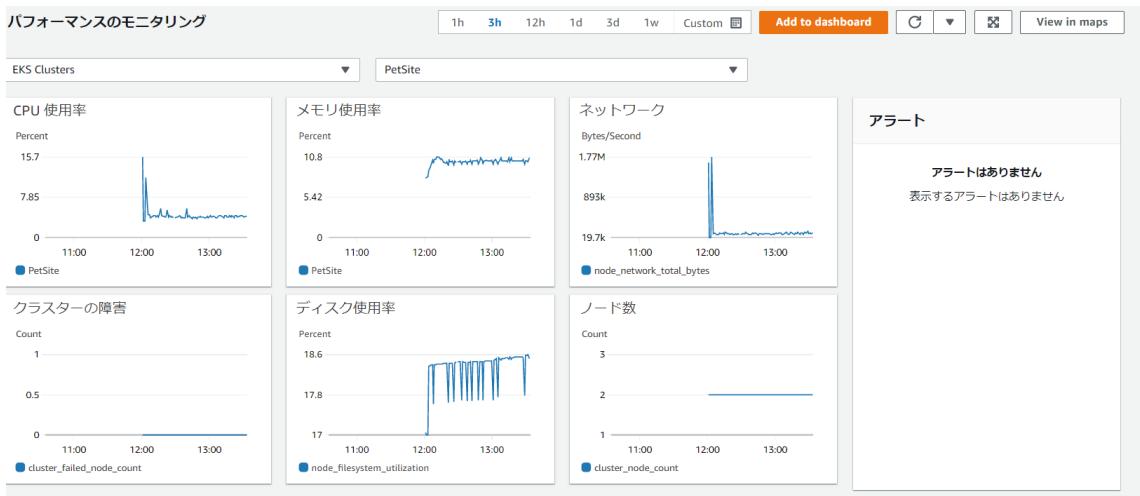
69. 「マップ表示」ボタンをおします。今回構築した環境のコンテナ全体像を見ることができます



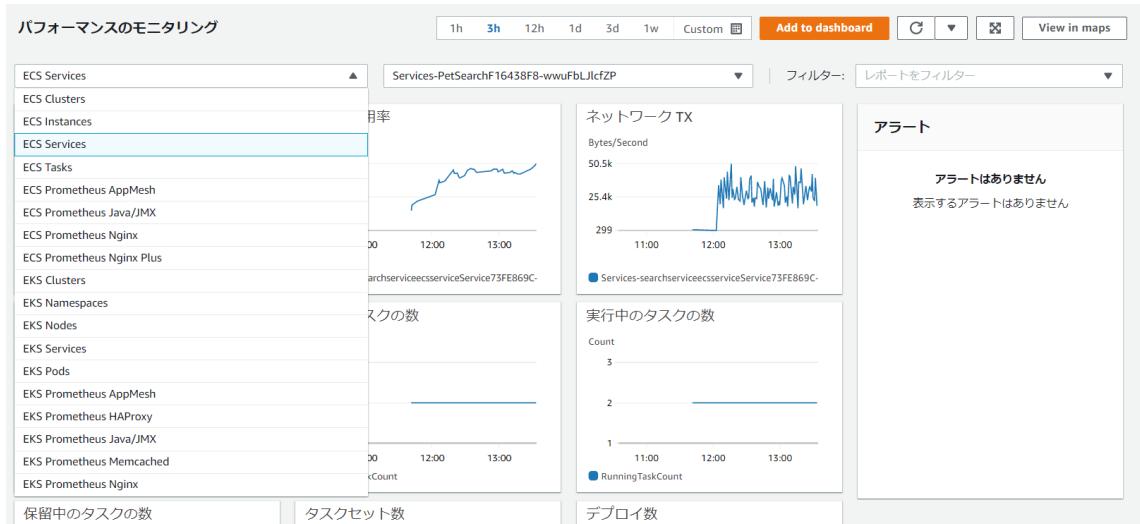
70. 個別ノードをクリックすると以下のようにコンピュートリソース状況が表示されます



71. 「ダッシュボードを表示」を押すと、さらにダッシュボードが表示されます



72. ドロップダウンを使うことで個別環境のダッシュボードに切り替えることができます



73. ECS Services を選択し、画面下部「タスクのパフォーマンス」から何かを選び「アクション」から View application logs を選んでください

タスクのパフォーマンス (1/2)			
		アクション ▲	
		View application logs View AWS X-Ray traces View performance logs	
Task	Task definition	Service	Avg CPU (%) ▾ memory (%)
<input checked="" type="checkbox"/> 96fc2f04c1f64a09b0fcabae3b799505	ServicespayforadoptionserviceTaskDefinition36C5E9A9	Services-payforadoptionservicecsserviceService90C1D43F-CxUWcgknRPC	0.1% 2.7%
<input type="checkbox"/> c29ed29dc8842b3b4a708146e2186f8	ServicespayforadoptionserviceTaskDefinition36C5E9A9	Services-payforadoptionservicecsserviceService90C1D43F-CxUWcgknRPC	0.1% 2.7%

74. 表示される「クエリの実行」ボタンをおすと、トランザクションごとの詳細データが
出力されます

The screenshot shows the CloudWatch Metrics Insights interface. At the top, there's a search bar labeled "ロググループを選択" and a dropdown for time intervals (5m, 30m, 1h, 3h, 12h, Custom). Below that is a code editor window containing a CloudWatch Metrics Log Query:

```
1 fields @timestamp, @log, @logStream, @message
2 | sort @timestamp desc
3 | limit 50
```

Below the code editor are three buttons: "クエリの実行" (Execute Query) in orange, "保存" (Save), and "履歴" (History).

Under the "ログ" tab, there's a histogram titled "9,799 の 50 の一致したレコードの表示" (Showing 50 of 9,799 matching records) with a note "9,799 レコード (1.6 MB) が 2.95 @ 3,362 records/s (635.6 KB/s) でスキャンされました". To the right are buttons for "結果をエクスポート" (Export Results) and "ダッシュボードに追加" (Add to Dashboard).

The main area displays a log stream with 5 entries, each with timestamp, log, log stream, and message fields. The log stream column shows log IDs like "c29ed29dc8842b3b4a708146e2186f8...".

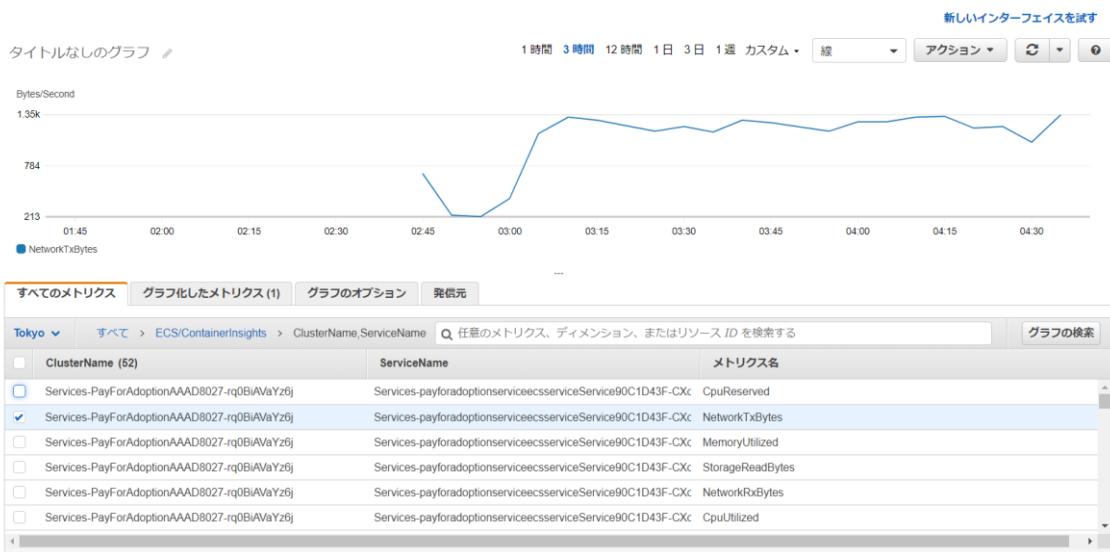
75. 以下の URL でメトリクス画面に移動します

<https://console.aws.amazon.com/cloudwatch/home?#metricsV2:graph=~%28%29>



76. [ECS/ContainerInsights]をクリックしてください

77. さらに何かを選択し、適当なメトリクスをクリックするとグラフが表示されます。以下の例では、「Services-PayForAdoptionAAAD8027-rq0BiAVaYz6j」クラスターのネットワーク帯域使用量を表示しています



78. 今まで ECS を見てきましたが、今度は Cloud9 の画面に戻り、EKS の設定を行います。以下のコマンドを実行してください

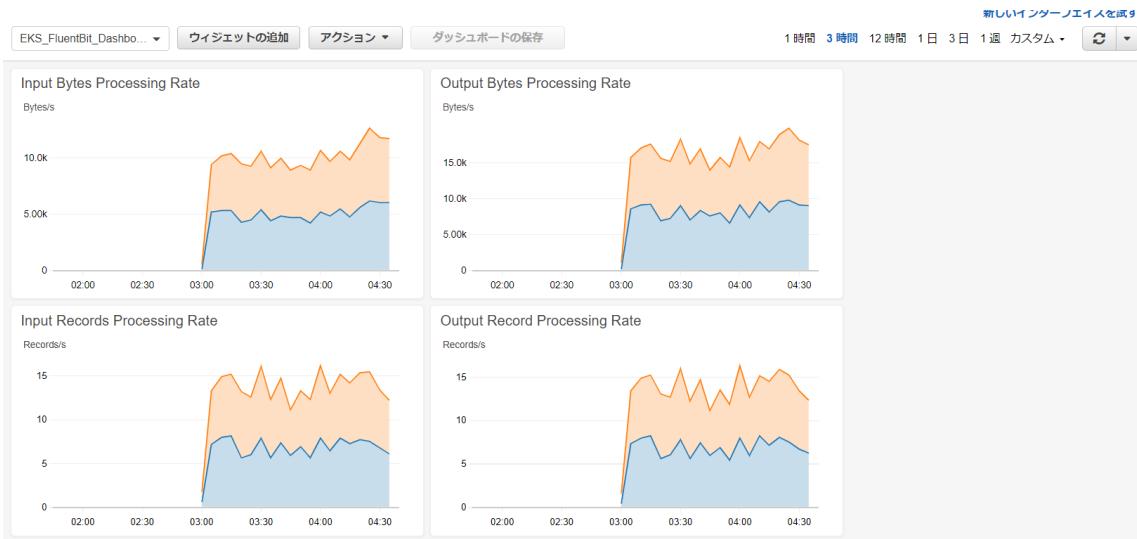
```
kubectl get pods -n amazon-cloudwatch
```

```
ec2-user:~/environment/workshopfiles/one-observability-demo/PetAdoptions/cdk/pet_stack/resources (main) $ kubectl get pods -n amazon-cloudwatch
NAME                      READY   STATUS    RESTARTS   AGE
cloudwatch-agent-bxnts     1/1     Running   0          105m
cloudwatch-agent-trmfd     1/1     Running   0          105m
cwagent-prometheus-84694b6b49-97xm  1/1     Running   0          104m
fluent-bit-hvraq6          1/1     Running   0          105m
fluent-bit-md2g7           1/1     Running   0          105m
```

EKS から fluent bit 経由でログが出力されていることがわかります

79. 以下の URL でダッシュボードに移動します

https://console.aws.amazon.com/cloudwatch/home?#dashboards:name=EKS_FluentBit_Dashboard



80. 各メトリクスの取得などは ECS と同様の画面から行えます。以下の URL で EKS の稼働状態を確認して下さい

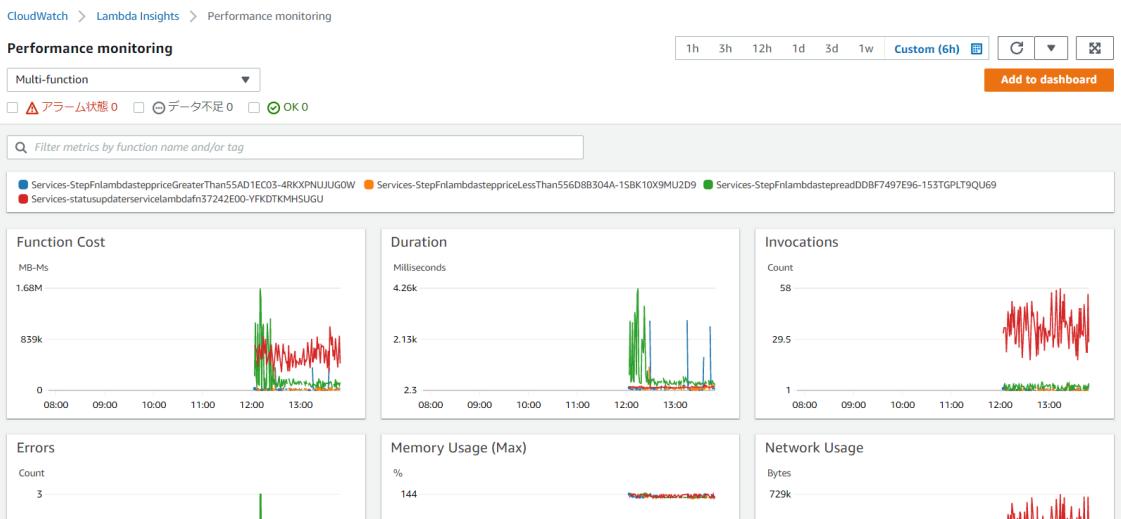
<https://console.aws.amazon.com/cloudwatch/home?#container-insights:infrastructure>
<https://console.aws.amazon.com/cloudwatch/home?#metricsV2:graph=~%28%29>
 (pod で絞り込んでください)

[CloudWatch Lambda Insights]

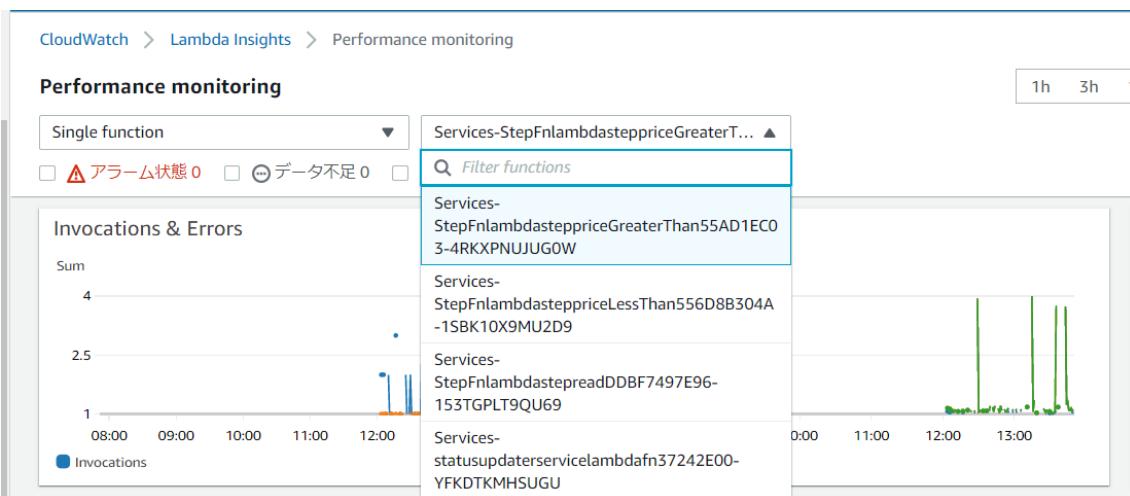
81. CloudWatch マネージメントコンソール左ペインから「Lambda Insights」を選び
 「Multi-function」を押します



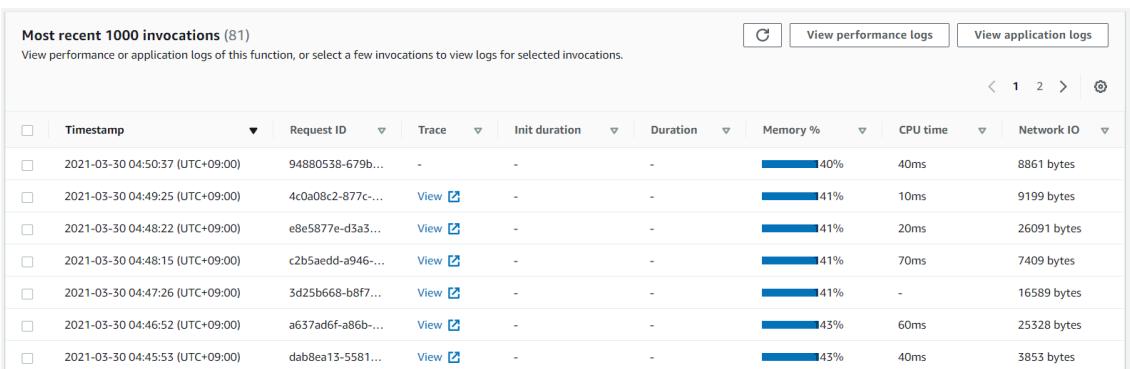
82. 以下のように Lambda 関数全体の実行状況がグラフで出力されます

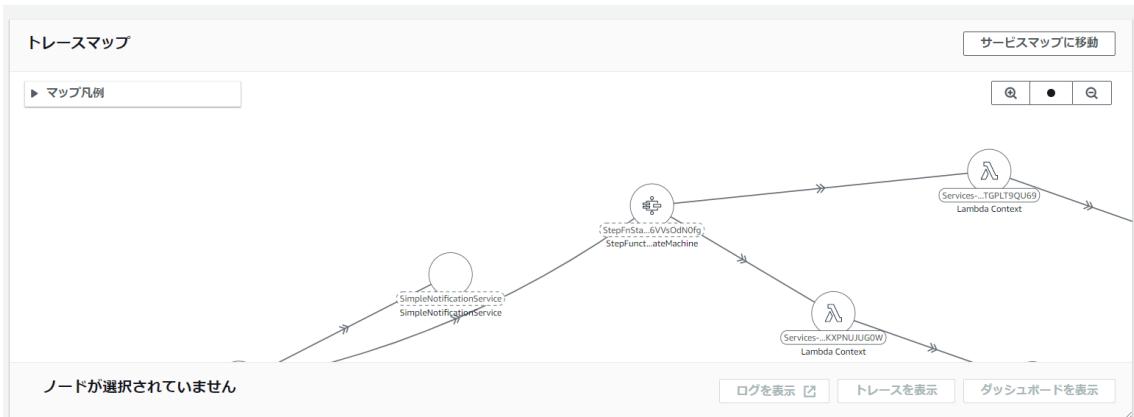


83. ドロップダウンでシングルに切りかえることで個別 Lambda 関数の状態が確認できます



84. 画面下部に表示される詳細の Trace に表示されている「View」を押すと、先ほどの Service Lens の画面がでてきます。メモリ利用率がそれなりに高いことがわかります。





85. (オプション) Lambda に詳しい方は、関数のメモリ設定を変更してみましょう。以下のようにパフォーマンスが異なることがわかります

The screenshot shows the AWS Lambda Metrics table titled 'Most recent 1000 invocations (13)'. It lists 13 recent invocations with the following details:

	Timestamp	Request ID	Trace	Init duration	Duration	Memory %	CPU time	Network IO
□	2021-03-30 04:58:31 (UTC+09:00)	597ecbf9-1903...	-	-	-	76%	-	746 bytes
□	2021-03-30 04:58:13 (UTC+09:00)	58d63619-80e1...	-	-	-	76%	1090ms	84 bytes
□	2021-03-30 04:51:08 (UTC+09:00)	d9c720fd-6b17...	-	-	-	41%	10ms	13268 bytes
□	2021-03-30 04:50:37 (UTC+09:00)	94880538-679b...	-	-	-	40%	40ms	8861 bytes
□	2021-03-30 04:50:29 (UTC+09:00)	037e6044-a7c5...	-	-	-	40%	10ms	16025 bytes

[CloudWatch Anomaly Detection]

86. 以下の URL に移動します

https://console.aws.amazon.com/cloudwatch/home?#metricsV2:graph=~%28%29;query=~%27*7bContainerInsights*2cClusterName*2cNamespace*2cPodName*7d*20MetricName*3d*22pod_cpu_utilization*22



87. Pod Name 「petsite-deployment」 にチェックを付けます

88. 「グラフ化したメトリックス」タブをクリックし、異常検知ボタンをおします

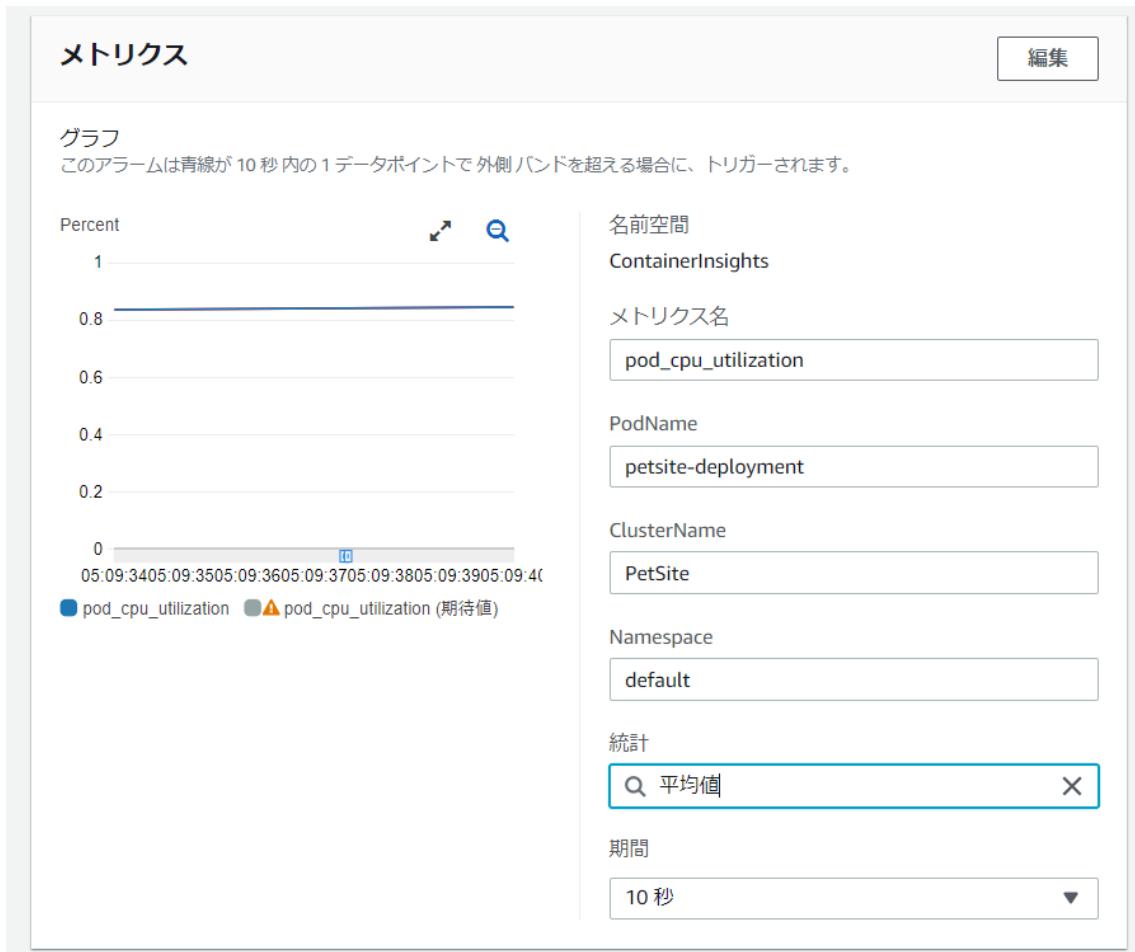
89. 異常検知が有効化され、これから 2 週間学習期間にはいりました



90. (m1,2)と表示されている部分は修正可能です。m1 は異常検知対象メトリクス、2 は許容する偏差（その範囲であれば異常ではないという閾値）。

91. 「モデルの編集」を押します。例えばあらかじめわかっているシステムメンテナンスなどはここで設定することで学習期間から除外可能です

92. ベルのマークを押すことで、異常が検知された際の挙動を設定可能です



条件

しきい値の種類

静的
値をしきい値として使用

異常検出
バンドをしきい値として使用

pod_cpu_utilization が次の時...
アラーム条件を定義

バンドの外
> しきい値または < しきい値

バンドより大きい
> しきい値

バンドより低い
< しきい値

異常検出のしきい値
標準偏差に基づいています。数字が大きいほどバンドが広く、小さいほどバンドが狭いことを意味します。

0.2

正の数にする必要があります

▶ その他の設定

おつかれさまでした！

削除手順：

1. 以下のコマンドを Cloudshell から実行する

```
curl https://raw.githubusercontent.com/aws-samples/one-observability-demo/main/PetAdoptions/cdk/pet_stack/resources/destroy_stack.sh | bash
```

2. 残った CF n を削除

3. CloudWatch Synthetics Canary を削除 (中止、してから削除)

4. CloudWatch Logs ロググループ (pet という文字列を含むものすべて)

5. Lambda 関数 (cwsyn-test ……)

6. S3 バケット (services-s3bucketpetadoption…)