

ECS向けのドリフト検知機構を 実装してみた

2024-09-25
JAWS-UG東京

@t-kikuc

菊池 哲哉 Kikuchi Tetsuya



@t-kikuc

@t_kikuc

- PipeCDメンテナ @CyberAgent, Inc.
 - ECS、Lambda関連よく見えます



- 好きなAWSサービス: ECS、Support
- JAWS-UG初登壇
- 年内にAWS認定10点ぐらい失効します🎆

PipeCDとは？



- CD(継続的デリバリー)のOSS
- 主な特徴
 - ECS、Lambda、K8s、Cloud Run、Terraformにデプロイできる (2024.09現在)
 - Canary、Blue/Greenに強み
 - GitOpsスタイル
- 沿革
 - サイバーエージェント発のOSS
 - 2023.05～ CNCF Sandbox Project
 - 沿革詳細: <https://speakerdeck.com/ffjlabo/our-journey-from-in-house-cd-system-to-open-source>
- 入門ハンズオン(日本語版)
 - <https://github.com/pipe-cd/tutorial/blob/main/content/ja/README.md>

PipeCDとは？

要は

ECSとかLambdaに
デプロイできるOSS

[source](#)



[昨日リリース] Drift Detection for ECS

Release v0.49.0

Latest



t-kikuc released this 5 hours ago

· 1 commit to master since this release



v0.49.0



909fd76



Release v0.49.0 with changes since v0.48.9

New Features

- [Lambda] Support Drift Detection for Lambda ([#5186](#))
- [Lambda] Support LiveState Store, Reporter, and UI for Lambda ([#5143](#))
- [Lambda] Support Plan Preview for Lambda ([#5046](#))
- [ECS] Support Drift Detection for ECS ([#5122](#))
- [ECS] Implement LiveState UI for ECS ([#5021](#))
- [ECS] Implement LiveStateStore&Reporter for ECS ([#4979](#))
- Support OIDC for the SSO ([#5008](#))

<https://github.com/pipe-cd/pipecd/releases/tag/v0.49.0>

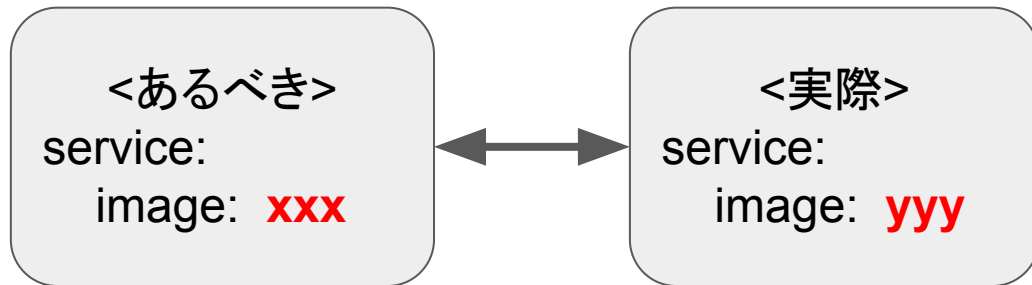
アジェンダ

1. ドリフト検知とは
2. どんなものを実装したか
3. 実装概説
4. 備考

1. ドリフト検知とは

ドリフト検知とは

- 「あるべき状態」と「実際の状態」のズレを検知する



- 「意図せぬインフラ変更」を検知できる
- 何が嬉しい？
 - 「IaCのコードを見れば、インフラがわかる」状態を保てる
 - 「謎のズレによりインフラ変更がコケる→調査が必要」を防げる
 - 緊急対応で手動変更した後など

参考: CloudFormationのドリフト検出

Differences (4) < 1 >

☒	Property	Change	Expected value	Current value
☒	AttributeDefinitions.1	ADD	-	{ "AttributeName": "stuff", "AttributeType": "S" }
☒	ProvisionedThroughput.ReadCapacityUnits	NOT_EQUAL	5	6
☒	ProvisionedThroughput.WriteCapacityUnits	NOT_EQUAL	5	4
☒	Tags	REMOVE	[{"Key": "test", "Value": "test"}]	-

Details

Expected

```
{
  "AttributeDefinitions": [
    {
      "AttributeName": "Name",
      "AttributeType": "S"
    }
  ],
  "KeySchema": [
    {
      "AttributeName": "Name",
      "KeyType": "HASH"
    }
  ],
  "ProvisionedThroughput": {
    "ReadCapacityUnits": 5,
    "WriteCapacityUnits": 5
  },
  "Tags": [
    {
      "Key": "test",
      "Value": "test"
    }
  ]
}
```

Actual

```
{
  "AttributeDefinitions": [
    {
      "AttributeName": "Name",
      "AttributeType": "S"
    },
    {
      "AttributeName": "stuff",
      "AttributeType": "S"
    }
  ],
  "KeySchema": [
    {
      "AttributeName": "Name",
      "KeyType": "HASH"
    }
  ],
  "ProvisionedThroughput": {
    "ReadCapacityUnits": 6,
    "WriteCapacityUnits": 4
  }
}
```

<https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/detect-drift-stack.html>

参考: ecspresso diff

```
$ ecspresso diff
--- arn:aws:ecs:ap-northeast-1:123456789012:service/ecspresso-test/nginx-local
+++ ecs-service-def.json
@@ -38,5 +38,5 @@
   },
   "placementConstraints": [],
   "placementStrategy": [],
-  "platformVersion": "1.3.0"
+  "platformVersion": "LATEST"
 }

--- arn:aws:ecs:ap-northeast-1:123456789012:task-definition/ecspresso-test:202
+++ ecs-task-def.json
@@ -1,6 +1,10 @@
{
  "containerDefinitions": [
    {
      "cpu": 0,
      "environment": [],
      "essential": true,
-     "image": "nginx:latest",
+     "image": "nginx:alpine",
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
```

デプロイ前の確認
にも使える

<https://github.com/kayac/ecspresso?tab=readme-ov-file#diff>

2. どんなものを実装したか

実行結果: ドリフト有り

ecs-driftdetection

↻ SYNC

❗ Out of Sync ⚠️ Unknown

The ecs definition files are not synced [HIDE DETAILS](#) 📄

Diff between the defined state in Git at commit 4593939 and actual live state:

--- Actual (LiveState)

+++ Expected (Git)

ServiceDefinition:

#ServiceDefinition.DesiredCount

- DesiredCount: 1

+ DesiredCount: 0

TaskDefinition:

ContainerDefinitions:

#TaskDefinition.ContainerDefinitions.1

+ - Command: <invalid Value>

+ Cpu: 100

+ DependsOn: <invalid Value>

+ DisableNetworking: <invalid Value>

+ DnsSearchDomains: <invalid Value>

+ DnsServers: <invalid Value>

実行結果: ドリフト有り

ecs-drift-detection

❗ Out of Sync

⚠ Unknown

↻ SYNC

▼

The ecs definition files are not synced [HIDE DETAILS](#) 

Diff between the defined state in Git at commit 4593939 and actual live state:

```
--- Actual (LiveState)
+++ Expected (Git)

ServiceDefinition:
  #ServiceDefinition.DesiredCount
- DesiredCount: 1
+ DesiredCount: 0

TaskDefinition:
  ContainerDefinitions:
    #TaskDefinition.ContainerDefinitions.1
+   - Command: <invalid Value>
+   Cpu: 100
+   DependsOn: <invalid Value>
+   DisableNetworking: <invalid Value>
+   DnsSearchDomains: <invalid Value>
+   DnsServers: <invalid Value>
```

サービスのドリフト

タスク定義のドリフト

実行結果: ドリフト無し



The screenshot displays the PipeCD web interface. At the top, the header bar includes the PipeCD logo, a dropdown menu labeled 'pipecd', and navigation links for 'Applications', 'Deployments', and 'Chains'. A user profile picture is visible on the right. Below the header, the main content area shows a single application entry named 'ecs-drift-detection'. To the right of this entry are two buttons: a 'SYNC' button with a circular arrow icon and a dropdown arrow. Below the application name, there are two status indicators: a green checkmark icon followed by the text 'Synced', and a yellow warning icon followed by the text 'Unknown'. The 'Synced' status is highlighted with a red rectangular box.

Application	Status	Actions
ecs-drift-detection	✓ Synced ⚠ Unknown	↻ SYNC ▼

3. 実装概説

0. 初期状態



GitHub

<expected>

service: ...

taskdef: ...



PipeCD



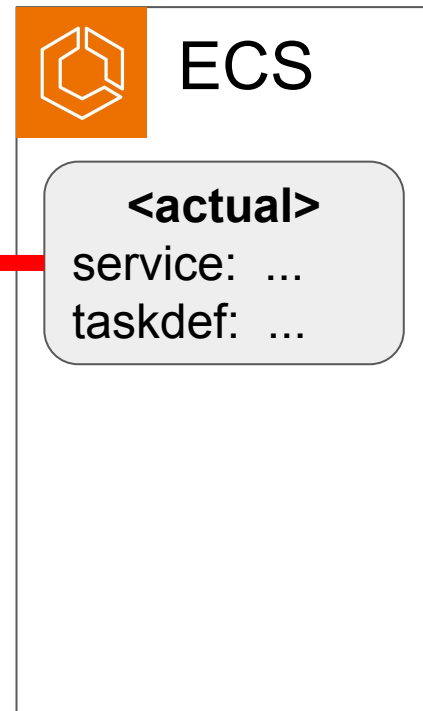
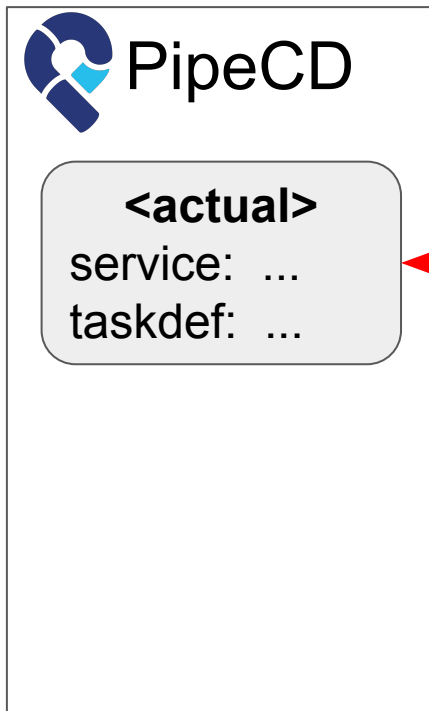
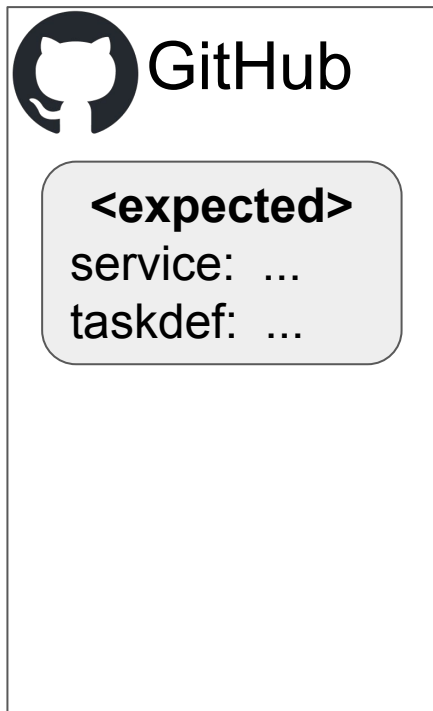
ECS

<actual>

service: ...

taskdef: ...

1/3. 実際の状態を取得



2/3. あるべき状態を取得



GitHub

<expected>
service: ...
taskdef: ...



PipeCD

<actual>
service: ...
taskdef: ...

<expected>
service: ...
taskdef: ...



ECS

3/3. 比較



GitHub



PipeCD

<actual>

service: ...

taskdef: ...



<expected>

service: ...

taskdef: ...



ECS

実装上の工夫1/2

- 「取得した実リソース」のパラメータを““いい感じ””にする
 - 背景: Create/UpdateXxxInput と DescribeXxxOutputが一致しない
 - 検出不要な差分を無視する
- 対応例
 - 実リソースにしか存在しない項目 (例)RunningCount、Status、CreatedAt
 - → 無視
 - ECS側で自動ソートされる項目 (例)Subnets
 - → あるべき状態側をソート
 - デフォルト値が使われる項目
 - 固定値 → その値を利用 (例)AssignPublicIp、PortMappings.Protocol
 - 状態依存 → 無視 (例)RoleArn、PlatformVersion

実装: <https://github.com/pipe-cd/pipecd/blob/v0.49.0/pkg/app/piped/driftdetector/ecs/detector.go#L240-L311>

実装上の工夫2/2

- AutoScaling利用時には、DesiredCountはドリフトとして検出しない
 - 無視しないと、常にドリフト検出され続けてしまう
 - 「AutoScaling利用時」: DesiredCountが未指定(or 0)かどうかで判断

備考

備考

- ドリフト検知はデフォルトで有効のため、設定不要です
- Standalone Taskは未対応です
- いくつかの項目は、ドリフトがあっても無視してます...
 - 例) service.Tag, service.LoadBalancers
<https://github.com/pipe-cd/pipecd/blob/v0.49.0/pkg/app/piped/driftdetector/ecs/detector.go#L230-L239>
- Lambdaでも実装・リリースしました
 - <https://github.com/pipe-cd/pipecd/pull/5186>

関連資料

- PipeCD入門ハンズオン([日本語版](#))
 - <https://github.com/pipe-cd/tutorial/blob/main/content/ja/README.md>
- ドリフト検知の実装(DriftDetection)
 - ECS: <https://github.com/pipe-cd/pipecd/pull/5122>
 - Lambda: <https://github.com/pipe-cd/pipecd/pull/5186>
 - 実リソースの取得処理:
 - ECS: <https://github.com/pipe-cd/pipecd/pull/4979>
 - Lambda: <https://github.com/pipe-cd/pipecd/pull/5143>
 - 簡単な試し方:
 - 1. [PipeCD Tutorial](#)でECSにデプロイ
 - 2. ECSを手動で更新してドリフトを発生させる → PipeCDのコンソールで確認
- DriftDetection詳細Docs
 - <https://pipecd.dev/docs-v0.49.x/user-guide/managing-application/configuration-drift-detection/>
- PipeCDでECS AutoScalingと併用する方法
 - <https://t-kikuc.hatenablog.com/entry/2024/08/11/064753>



PipeCD

日本語版Tutorial 🎉

