COVLint

Coverity Connectサーバーの特長の一つである問題のある行に指摘が付いたソースコード画面をVisual Studio Code上に実現しました。Language Server Protocol (言語サーバー)を利用していますので、言語サーバーの機能があるAtom、VimやEmacsなどのエディターで動作します(と言われていますが未確認です)。

• Coverity Connect ソースコード画面

```
◆■ 50 Lest_lambda_function.py
         13. 条件 expected_put_api_mock !== None は true となりました。
261
                   if expected_put_api_mock is not None:
                        # put_api_mock_configが設定されていない場合はテストデータ不正として失敗にする
262
         14. 分岐条件 put_api_mock_config === None は、false に分岐しました。
263
                       if put_api_mock_config is None:
264
                           assert False, f"Test data invalid. test_number: {test_number}"
                       else:
265
         15. 条件 "call_count" in expected_put_api_mock は true となりました。
                                                                      Coverity の指摘
266
                           if "call_count" in expected_put_api_mock:
      ◆ CID 455536: (#1 of 2): null-like値の誤使用 (FORWARD_NULL)
         16. property_access: null のような値 put_api_mocker のプロパティにアクセスしています。
267
                               assert put_api_mocker.call_count == expected_put_api_mock["call_count"], f"test_number: {test_number}"
                           if "call_with" in expected_put_api_mock:
268
◆ CID 455536:(#2 of 2):null-like値の誤使用 (FORWARD_NULL) [ "問題の選択"]
269
                                put_api_mocker.assert_called_with(
                                    expected_put_api_mock["call_with"]["url"],
json=expected_put_api_mock["call_with"]["json"],
270
271
                                    headers=expected_put_api_mock["call_with"]["headers"],
                                   timeout=tuple(expected_put_api_mock["call_with"]["timeout"])
                           if "assert_has_calls" in expected_put_api_mock:
```

• COVLint ソースコード画面

```
timeout=tuple(expected_get_api_mock["call_with"]["timeout"])
              expected_put_api_mock = expected.get("put_api_mock")
                                                                      COVLint が出力した指摘
              if expected nut ani mo
                         CID: 455536: null-like値の誤使用 (FORWARD_NULL)
                         property_access: null のような値 put_api_mocker のブロバティにアクセスしています。 covlint
                 if put_
                         (function) call_count: Any
                        assert put api_mocker.call_count == expected_put_api_mock["call_count"], f"test_number: {test_number}"
                        "call with" in expected put api mock:
                        put_api_mocker.assert_called_with(
                            expected_put_api_mock["call_with"]["url"],
                            json=expected_put_api_mock["call_with"]["json"],
                           headers=expected_put_api_mock["call_with"]["headers"],
                            timeout=tuple(expected_put_api_mock["call_with"]["timeout"])
                     if "assert_has_calls" in expected_put_api_mock:
```

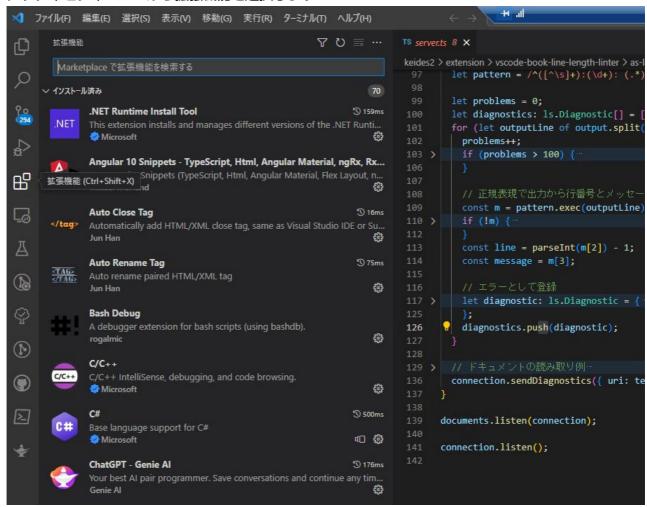
• COVLint ソースコード画面(指摘を複数表示)

```
rtension > cov_lint > feature-33417_linkage_lambda > 🍴 template.yaml > () Resources > () Table > () Properties > 📼 TableName
          CID: 439204: ストレージ バージョンの無効化 (SIGMA.storage_versioning_disabled)
          Sigma main event: Amazon DynamoDB テーブルで、ポイントインタイム リカバリー (PITR) が有効になっていません。
          その結果、誤って書き込みや削除を実行した場合の保護がありません。 covlint
          CID: 439197: 顧客入力暗号化キーの欠如 (SIGMA.vendor_provided_encryption_key)
          Sigma main event: Amazon DynamoDBサービスは、お客様が管理するKMSキーを使用せず、AWSが所有するキーまたはAWSが
          管理するKMSキーを使用してデータを暗号化するように構成されています。 covlint
          CID: 439190: データベースの暗号化が無効です (SIGMA.database_encryption_disabled)
          Sigma main event: Amazon DynamoDB テーブルでサーバー側の暗号化が有効になっていないため、保存中のデータが露出
          しています。 covlint
          Unresolved tag: !Sub YAM
                                                                    複数の指摘を表示可能
         TableName: !Sub ${Environment}-region-equipment-linkage-table
124
         AttributeDefinitions:
           - AttributeName: uid
```

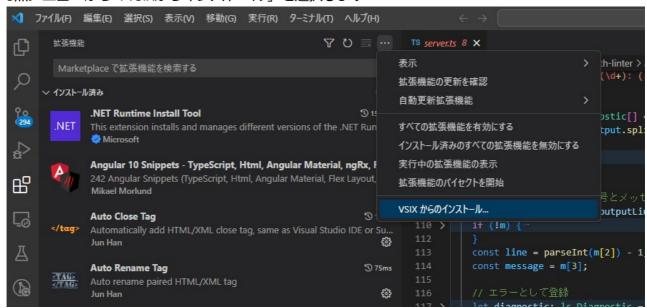
COVLintは、Coverityが解析したソースコードの指摘結果を Visual Studio Code に表示します。 指摘結果は、スクリプトcov_snapが取得したCSVファイルから読み込みます。 cov_snapは、Coverity Connect サーバーに登録されているソースコードの指摘内容を取得するスクリプトです。

実施手順

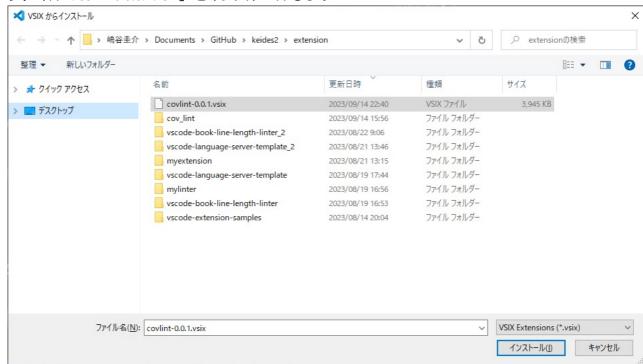
- 1. COVLint拡張機能のインストール
 - アクティビティーバーから拡張機能を選択します



• 3点メニューから「VSIXからインストール」を選択します



• ファイル「covlint-0.0.1.vsix」をインストールします



• インストール完了です

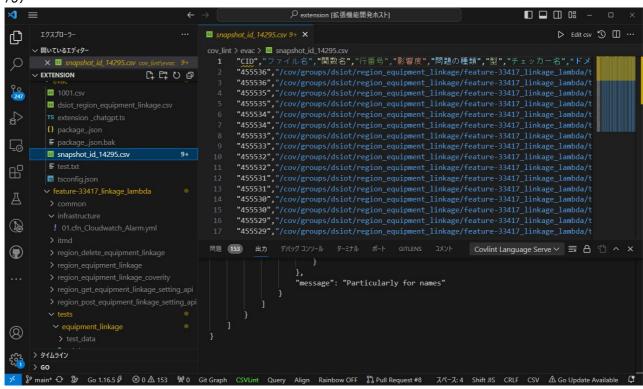




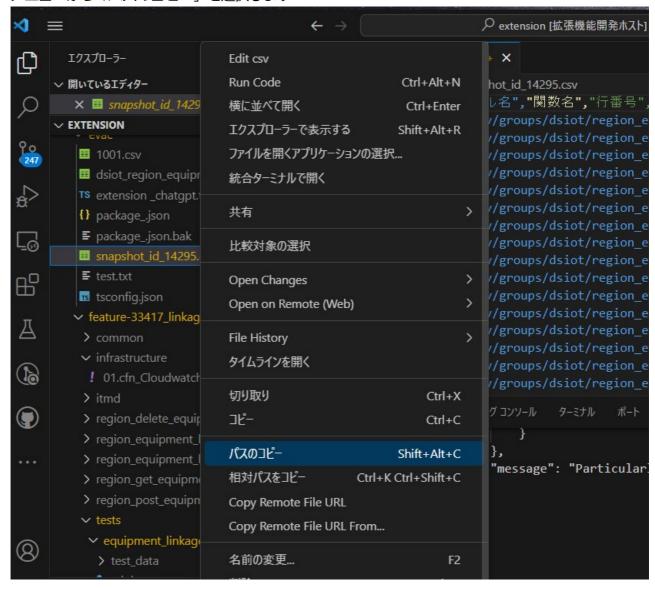
2. CSVファイルの読み込み

cov_snapで取得したスナップショットCSVファイルを読み込みます

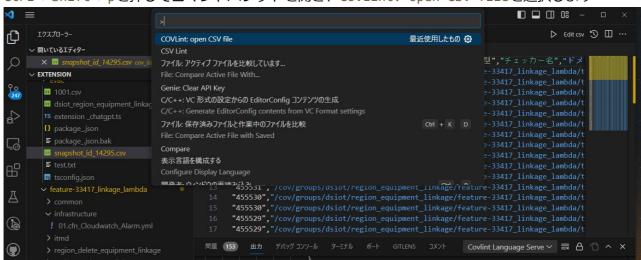
VSCodeのエクスプローラーからスナップショットを保存しているフォルダに移動し、読み込みむスナップショットを**右クリック**で選択します(左クリックでファイルの中身を表示する必要はありません)



メニューから「パスのコピー」を選択します



• Ctrl + Shift + pを押してコマンドパレットを開き、COVLint: open CSV fileを選択します



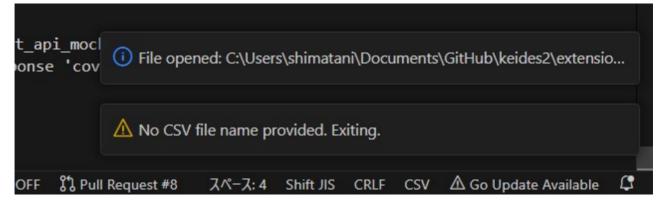
• 入力ボックスにコピーしたスナップショットCSVファイルのパスを貼り付けます



• 何も入力しなかったときのメッセージです



• 読み込みに成功したときのメッセージです

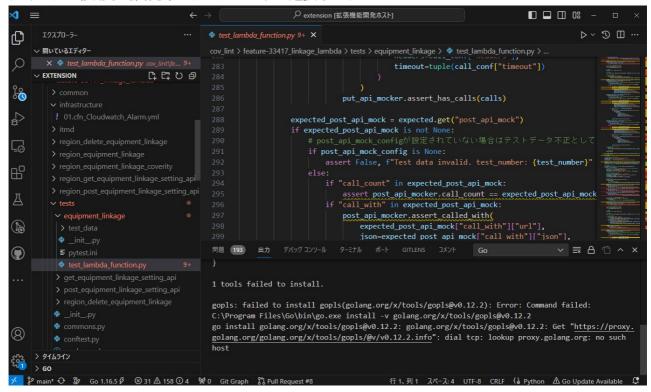


• 読み込みに失敗したときはリトライしてください

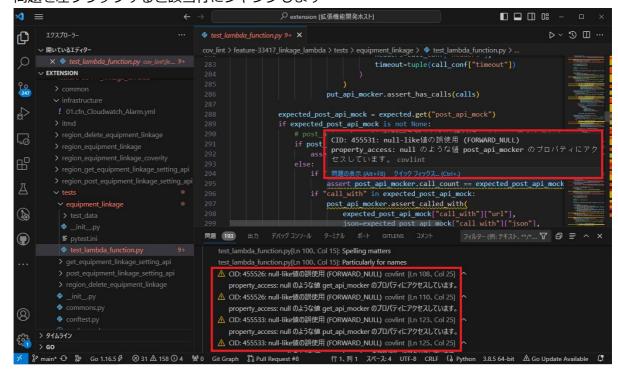


3. 指摘の表示

フォルダを移動して開発中のソースコードを選択します



- ソースコードの問題のある行に波線が入っているのでマウスオーバーします
 - 指摘がポップアップ表示されます
 - 問題パネルに指摘の一覧が表示されます
 - 問題を左クリックすると該当行にジャンプします



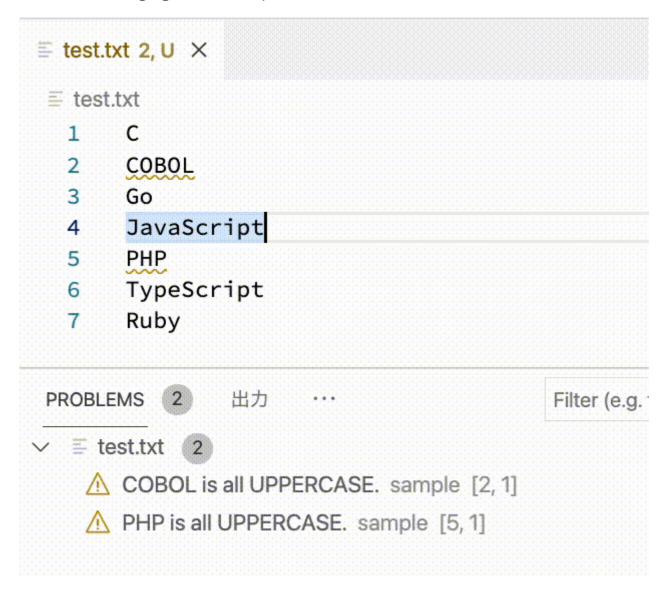
備考

Microsoftのhttps://github.com/Microsoft/vscode-extension-samplesにあるlsp-sampleや、 @Ikuyadeu 氏の記事 Language Server Protocol開発チュートリアル をベースにしましたので次の機能が残っています。

● コード検証機能

- 。 3文字以上の大文字を検出します
- コード修正機能
 - 。 警告箇所を小文字に修正します

下図はvscode-language-server-template Linter README から。



2023/09/15 嶋谷 初版