Checkmarx
The world runs on code. We secure it.

KICS Workshop

Checkmarx Professional Services

Agenda

KICS *explained*

> KICS stands for **K**ipping **I**nfrastructure as **C**ode **S**ecure

> Checkmarx product to find in IaC

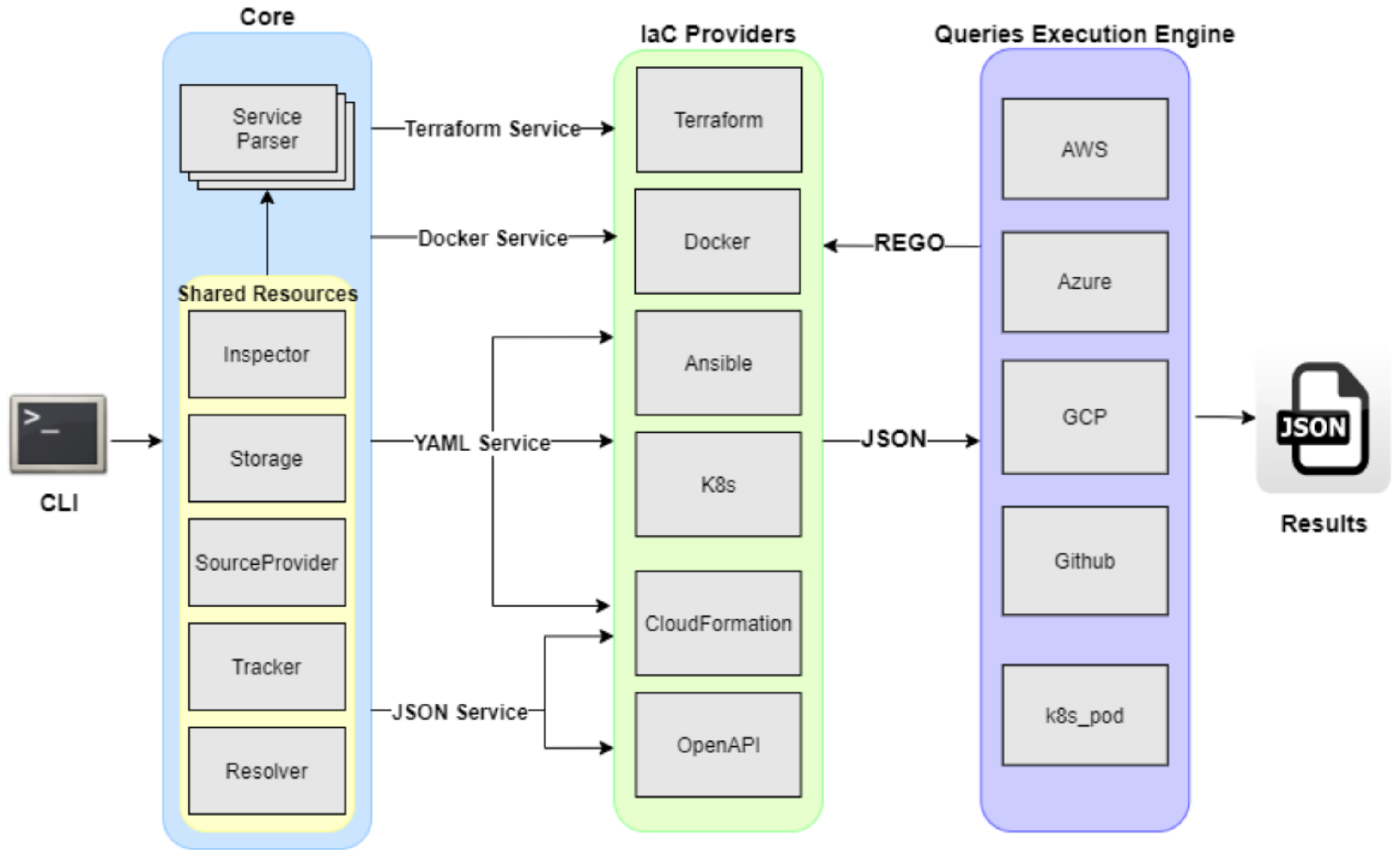
+ Security issues
+ Compliance issues
+ Infrastructure misconfigurations

> Open-source

+ Written in Golang using Open Policy Agent (OPA)
+ Security Queries written in Rego

> Covers vulnerability check in AWS, GCP, Azure

KICS *architecture*



KICS Supported Platforms

Ansible	**Azure Resource Manager**	**AWS CloudFormation**	**Crossplane**
.yaml	.json	.json .yaml	.yaml

KICS Supported Platforms

Azure Blueprints

Note: For the queries porposes, this is recognized as Azure Resource Manager.

.json

Docker

.dockerfile

Docker Compose

.yaml

gRCP

Google Remote Procedure Call

.proto

KICS Supported Platforms

Helm	Knative	Kubernetes	OpenAPI
Note: For queries porposes, this is recognized as Kubernetes. .yaml	.yaml	.yaml	.json .yaml **Swagger** 2.0 and **OpenAPI** 3.0

KICS Supported Platforms

Pulumi

.yaml

ServlessFW

.yml

Google Deployment Manager

.yaml

AWS SAM

Servless Application Model

Note: For queries porposes, this is recognized as Cloud Formation

.yaml

Supported Platforms

Terraform

Terraform Plan

.tf

.terraform.tfvars

.auto.tfvars

.json

...

Install KICS *Windows locally*

> Install go from https://golang.org/dl/

> Clone KICS repo

  + git clone https://github.com/Checkmarx/kics.git

> Build kics binaries

  + make build

> Scan

  + ./bin/kics scan -p '<path-of-your-project-to-scan>' --report-formats json -o ./results

Install KICS *Docker*

> Pre-requisit

  + Have docker running

> Get the image

  + docker pull checkmarx/kics:latest

> Run scan

  + docker run -t -v "{path_to_host_folder_to_scan}":/path checkmarx/kics scan -p "/path"-o "/path/"

Install KICS *some flags*

- -p
  - Path of the volume
  - docker run -v "C:\Users\soniad\Documents\kics":/path checkmarx/kics:latest scan -p "/path
  - docker run checkmarx/kics scan -p git::https://github.com/SoniaDias/docker_helloWorld
- -v
  - to create a local volume sync with container volume
  - docker run -v "C:\Users\soniad\Documents\kics":/path checkmarx/kics:latest
- -o
  - for the output folder
  - docker run -v "C:\Users\soniad\Documents\kics":/path checkmarx/kics:latest scan -p "/path" -o "/path"

Install KICS *some flags*

- --log-level DEBUG
- -v *verbose*
- docker run checkmarx/kics scan -p git::https://github.com/SoniaDias/docker_helloWorld --log-level DEBUG -v
- -d or –payload-path to generate the payload
  - *JSON model(to be explained in queries module, some slide ahead)*

Checkmarx

Install KICS *reporting*

- https://docs.kics.io/latest/commands/
- Flag
  - --report-formats
- Supported
  - --report-formats "glsast"
  - --report-formats "all"
  - --report-formats "sarif"
  - --report-formats "junit"
  - --report-formats "sonarqube"
  - --report-formats "html"
  - --report-formats "pdf"
  - --report-formats "cyclonedx"
  - --report-formats "asff"
  - --report-formats "csv"
  - --report-formats "codeclimate"

Install KICS - DEMO

> Get KICS docker image

> Execute a KICS scan in a local folder

> The folder can be cloned from

  + https://github.com/SoniaDias/docker_helloWorld

Install KICS – *useful documentation*

> https://docs.kics.io/latest/documentation/

> https://github.com/Checkmarx/kics/blob/master/docs/getting-started.md

CI/CD Pipelines

> Supported pipelines
  + https://github.com/Checkmarx/kics/blob/master/docs/integrations.md

> For this workshop, will be used
  + GiHubActions
  + Azure DevOps

> Code being scanned
  + Azure Resource Manager (ARM) .json
  + https://github.com/SoniaDias/Azure-Resource-Manager-ARM-Example

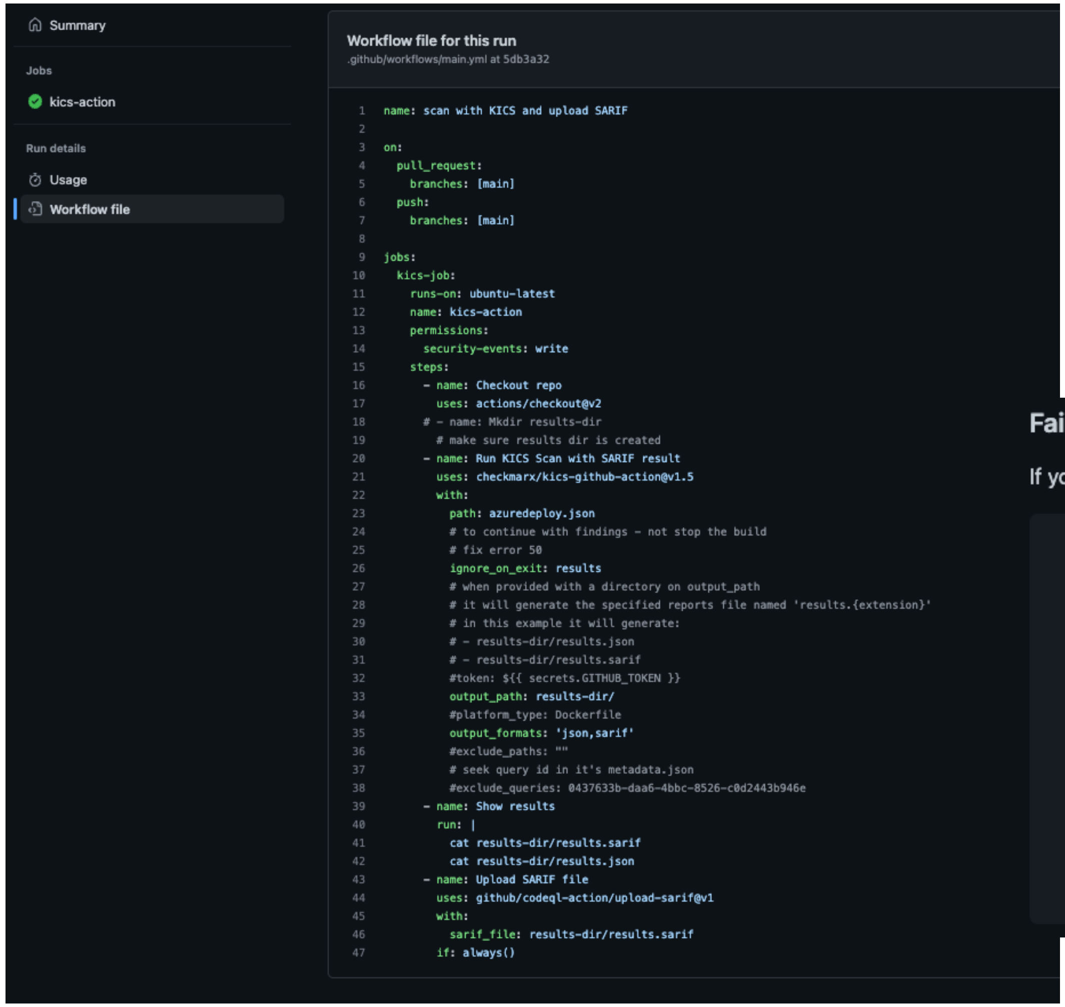
CI/CD Pipelines – *GitHub Actions DEMO*

> Referrer links

> https://github.com/Checkmarx/kics/blob/master/docs/integrations_ghactions.md

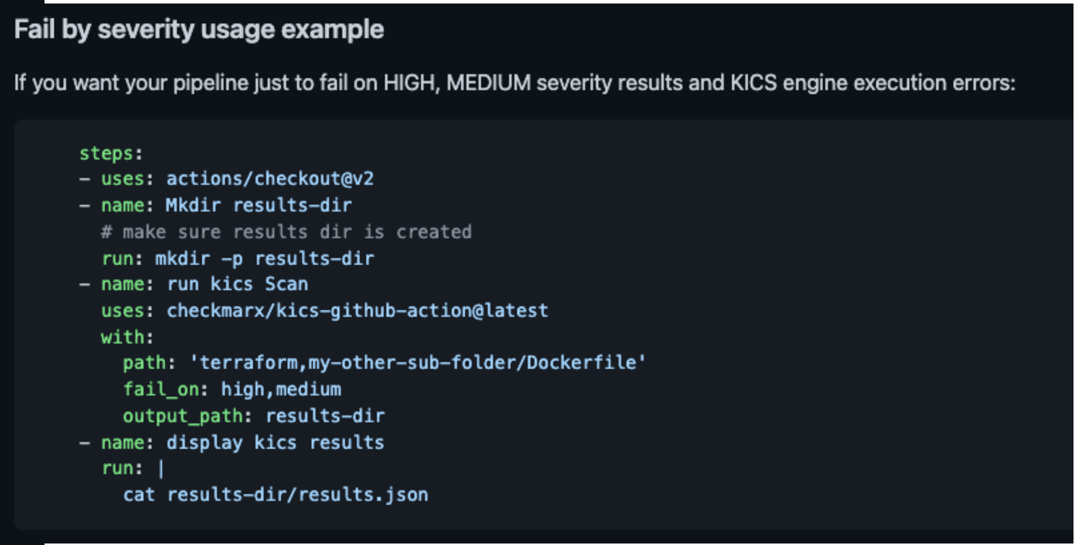
> https://docs.kics.io/latest/integrations_ghactions/

CI/CD Pipelines – *GitHub Actions DEMO*



Workflow file for this run
.github/workflows/main.yml at 5db3a32

```yaml
name: scan with KICS and upload SARIF

on:
  pull_request:
    branches: [main]
  push:
    branches: [main]

jobs:
  kics-job:
    runs-on: ubuntu-latest
    name: kics-action
    permissions:
      security-events: write
    steps:
      - name: Checkout repo
        uses: actions/checkout@v2
      # - name: Mkdir results-dir
        # make sure results dir is created
      - name: Run KICS Scan with SARIF result
        uses: checkmarx/kics-github-action@v1.5
        with:
          path: azuredeploy.json
          # to continue with findings - not stop the build
          # fix error 50
          ignore_on_exit: results
          # when provided with a directory on output_path
          # it will generate the specified reports file named 'results.{extension}'
          # in this example it will generate:
          # - results-dir/results.json
          # - results-dir/results.sarif
          #token: ${{ secrets.GITHUB_TOKEN }}
          output_path: results-dir/
          #platform_type: Dockerfile
          output_formats: 'json,sarif'
          #exclude_paths: ""
          # seek query id in it's metadata.json
          #exclude_queries: 0437633b-daa6-4bbc-8526-c0d2443b946e
      - name: Show results
        run: |
          cat results-dir/results.sarif
          cat results-dir/results.json
      - name: Upload SARIF file
        uses: github/codeql-action/upload-sarif@v1
        with:
          sarif_file: results-dir/results.sarif
        if: always()
```



Fail by severity usage example

If you want your pipeline just to fail on HIGH, MEDIUM severity results and KICS engine execution errors:

```yaml
    steps:
    - uses: actions/checkout@v2
    - name: Mkdir results-dir
      # make sure results dir is created
      run: mkdir -p results-dir
    - name: run kics Scan
      uses: checkmarx/kics-github-action@latest
      with:
        path: 'terraform,my-other-sub-folder/Dockerfile'
        fail_on: high,medium
        output_path: results-dir
    - name: display kics results
      run: |
        cat results-dir/results.json
```

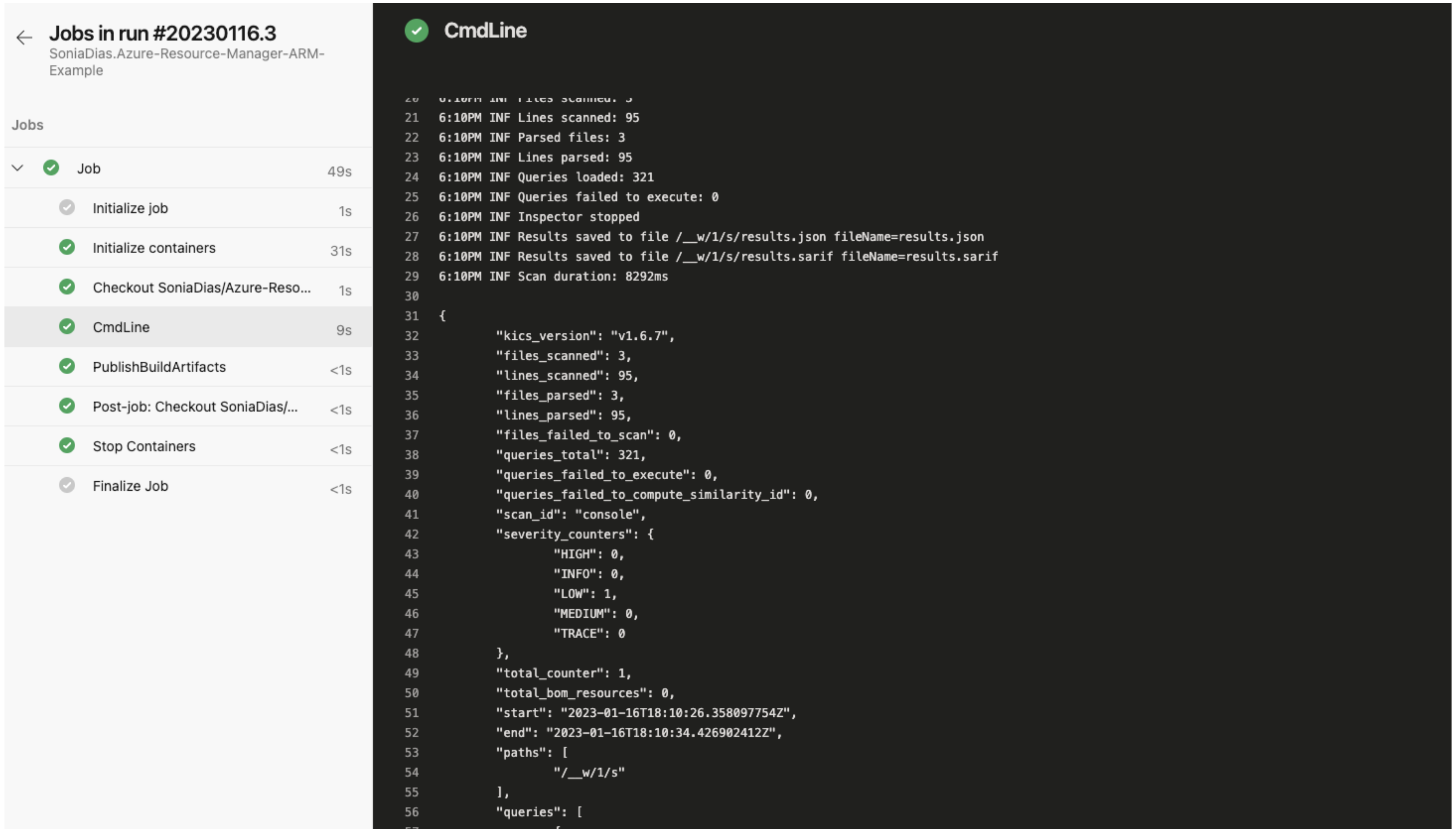
CI/CD Pipelines – *Azure DevOps DEMO*

> Referrer links

> https://docs.kics.io/latest/integrations_azurepipelines/

> https://github.com/Checkmarx/kics/blob/master/docs/integrations_azurepipelines.md

CI/CD Pipelines – *Azure DevOps DEMO*



Jobs in run #20230116.3
SoniaDias.Azure-Resource-Manager-ARM-Example

Jobs

Job	49s
Initialize job	1s
Initialize containers	31s
Checkout SoniaDias/Azure-Reso...	1s
CmdLine	9s
PublishBuildArtifacts	<1s
Post-job: Checkout SoniaDias/...	<1s
Stop Containers	<1s
Finalize Job	<1s

CmdLine

```
6:10PM INF Files scanned: 3
6:10PM INF Lines scanned: 95
6:10PM INF Parsed files: 3
6:10PM INF Lines parsed: 95
6:10PM INF Queries loaded: 321
6:10PM INF Queries failed to execute: 0
6:10PM INF Inspector stopped
6:10PM INF Results saved to file /__w/1/s/results.json fileName=results.json
6:10PM INF Results saved to file /__w/1/s/results.sarif fileName=results.sarif
6:10PM INF Scan duration: 8292ms

{
        "kics_version": "v1.6.7",
        "files_scanned": 3,
        "lines_scanned": 95,
        "files_parsed": 3,
        "lines_parsed": 95,
        "files_failed_to_scan": 0,
        "queries_total": 321,
        "queries_failed_to_execute": 0,
        "queries_failed_to_compute_similarity_id": 0,
        "scan_id": "console",
        "severity_counters": {
                "HIGH": 0,
                "INFO": 0,
                "LOW": 1,
                "MEDIUM": 0,
                "TRACE": 0
        },
        "total_counter": 1,
        "total_bom_resources": 0,
        "start": "2023-01-16T18:10:26.358097754Z",
        "end": "2023-01-16T18:10:34.426902412Z",
        "paths": [
                "/__w/1/s"
        ],
        "queries": [
```

CI/CD Pipelines – *useful documentation*

> https://github.com/Checkmarx/kics/blob/master/docs/integrations.md

> https://docs.kics.io/latest/integrations/

KICS Security Queries

> KICS gets the IaC file to scan and internally creates a *JSON* file in a universal format

+ Same structure for almost all platforms being scanned
+ So,
  - You pass a *.tf* or a *.yaml* file
  - KICS generates a *.json* from it

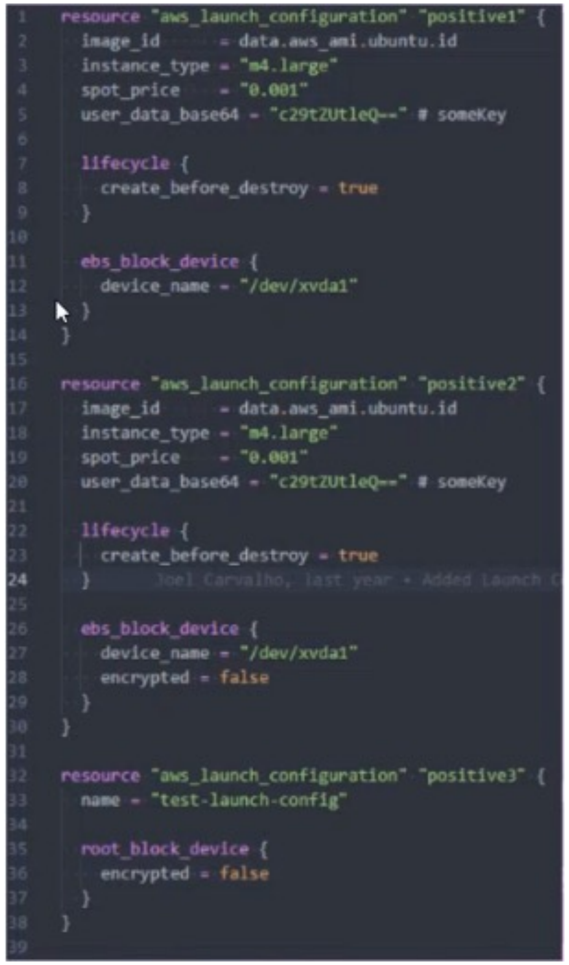
> This *JSON* is an internal representation of the code being scanned

> Array of Documents

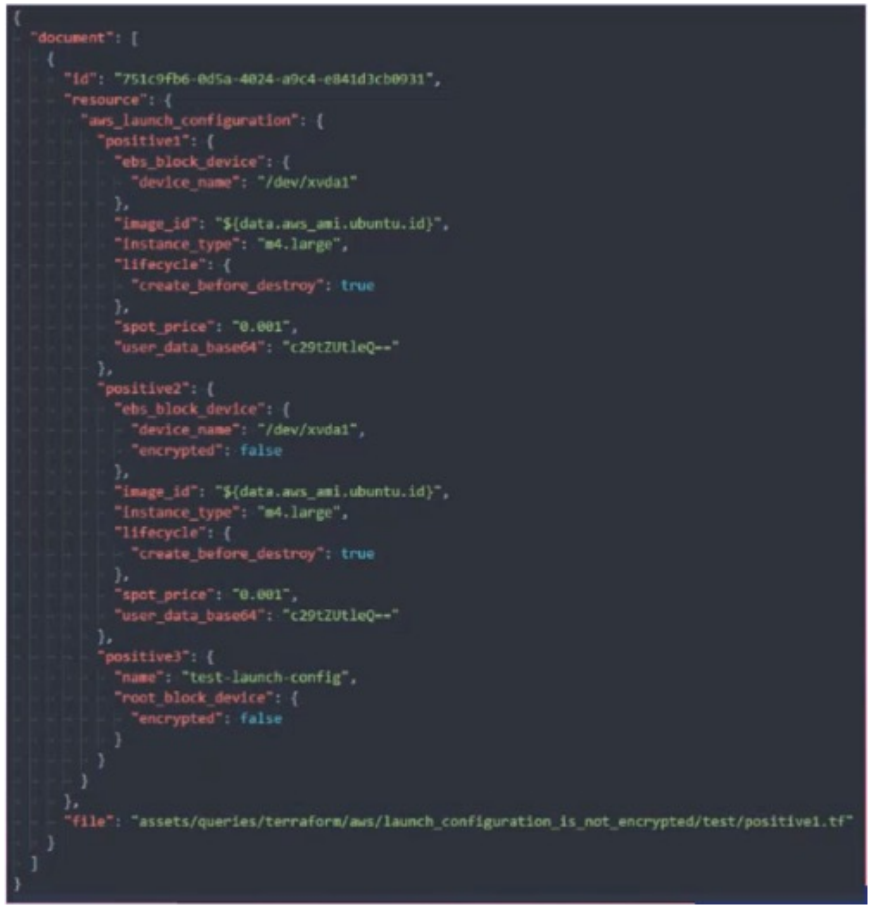
KICS Security Queries

- The KICS queries will be executed in this *JSON* file
- If we want to edit our queries to remove or add results, generate this file is our first step
- The *JSON* can be outputted so the user of KICS can see what is going to be scanned
  - To obtain the *JSON* file use **–d <path of the file>** or **- - payload-path <path of the file>**

KICS Security Queries – *json creation*

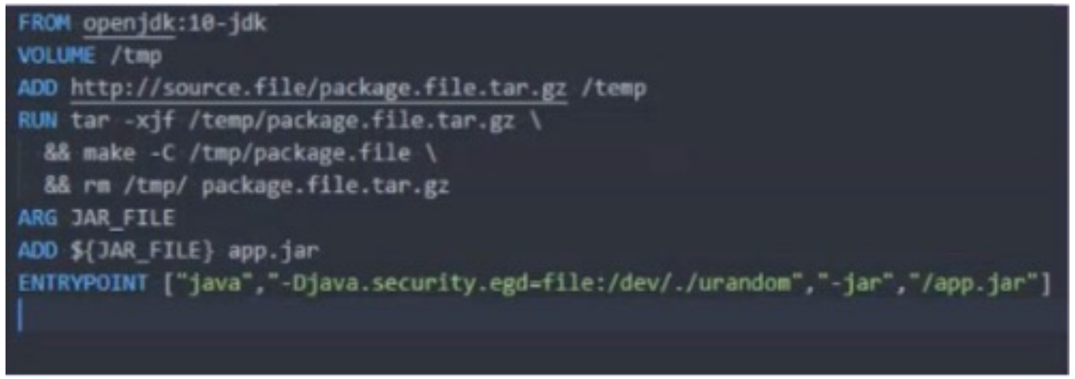


```
resource "aws_launch_configuration" "positive1" {
  image_id      = data.aws_ami.ubuntu.id
  instance_type = "m4.large"
  spot_price    = "0.001"
  user_data_base64 = "c29tZUtleQ==" # someKey

  lifecycle {
    create_before_destroy = true
  }

  ebs_block_device {
    device_name = "/dev/xvda1"
  }
}

resource "aws_launch_configuration" "positive2" {
  image_id      = data.aws_ami.ubuntu.id
  instance_type = "m4.large"
  spot_price    = "0.001"
  user_data_base64 = "c29tZUtleQ==" # someKey

  lifecycle {
    create_before_destroy = true
  }     Joel Carvalho, last year • Added launch c

  ebs_block_device {
    device_name = "/dev/xvda1"
    encrypted = false
  }
}

resource "aws_launch_configuration" "positive3" {
  name = "test-launch-config"

  root_block_device {
    encrypted = false
  }
}
```

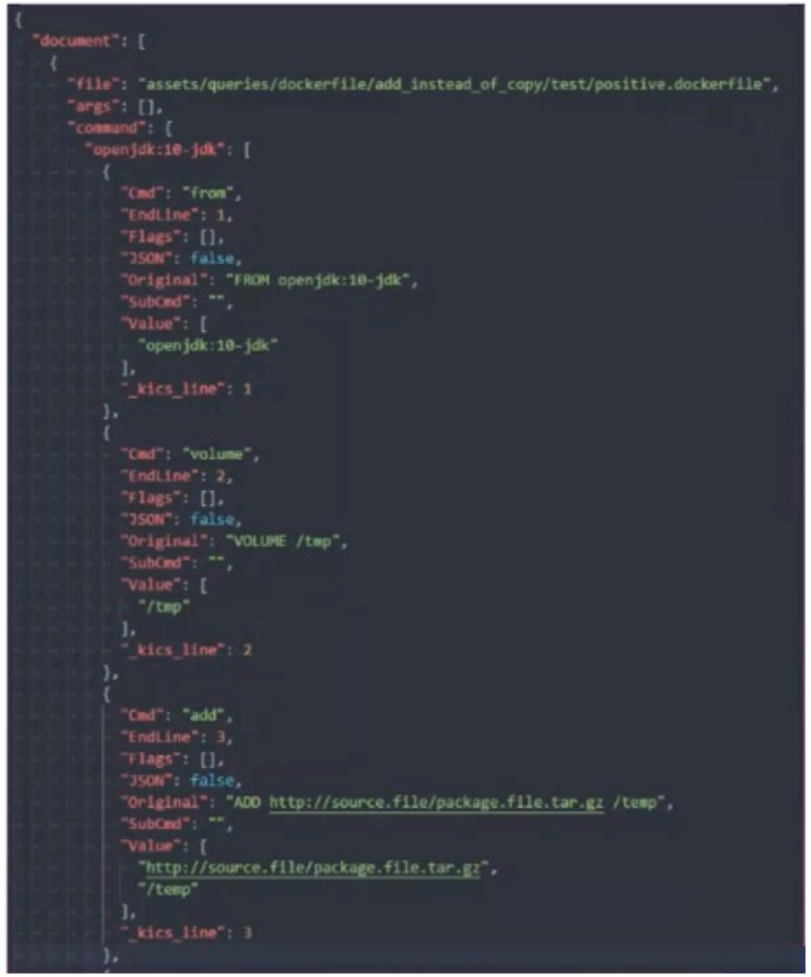


```
{
  "document": [
    {
      "id": "751c9fb6-0d5a-4024-a9c4-e841d3cb0931",
      "resource": {
        "aws_launch_configuration": {
          "positive1": {
            "ebs_block_device": {
              "device_name": "/dev/xvda1"
            },
            "image_id": "${data.aws_ami.ubuntu.id}",
            "instance_type": "m4.large",
            "lifecycle": {
              "create_before_destroy": true
            },
            "spot_price": "0.001",
            "user_data_base64": "c29tZUtleQ=="
          },
          "positive2": {
            "ebs_block_device": {
              "device_name": "/dev/xvda1",
              "encrypted": false
            },
            "image_id": "${data.aws_ami.ubuntu.id}",
            "instance_type": "m4.large",
            "lifecycle": {
              "create_before_destroy": true
            },
            "spot_price": "0.001",
            "user_data_base64": "c29tZUtleQ=="
          },
          "positive3": {
            "name": "test-launch-config",
            "root_block_device": {
              "encrypted": false
            }
          }
        }
      },
      "file": "assets/queries/terraform/aws/launch_configuration_is_not_encrypted/test/positive1.tf"
    }
  ]
}
```

KICS Security Queries – *JSON creation*



```
FROM openjdk:10-jdk
VOLUME /tmp
ADD http://source.file/package.file.tar.gz /temp
RUN tar -xjf /temp/package.file.tar.gz \
  && make -C /tmp/package.file \
  && rm /tmp/ package.file.tar.gz
ARG JAR_FILE
ADD ${JAR_FILE} app.jar
ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar"]
```



```
{
  "document": [
    {
      "file": "assets/queries/dockerfile/add_instead_of_copy/test/positive.dockerfile",
      "args": [],
      "command": {
        "openjdk:10-jdk": [
          {
            "Cmd": "from",
            "EndLine": 1,
            "Flags": [],
            "JSON": false,
            "Original": "FROM openjdk:10-jdk",
            "SubCmd": "",
            "Value": [
              "openjdk:10-jdk"
            ],
            "_kics_line": 1
          },
          {
            "Cmd": "volume",
            "EndLine": 2,
            "Flags": [],
            "JSON": false,
            "Original": "VOLUME /tmp",
            "SubCmd": "",
            "Value": [
              "/tmp"
            ],
            "_kics_line": 2
          },
          {
            "Cmd": "add",
            "EndLine": 3,
            "Flags": [],
            "JSON": false,
            "Original": "ADD http://source.file/package.file.tar.gz /temp",
            "SubCmd": "",
            "Value": [
              "http://source.file/package.file.tar.gz",
              "/temp"
            ],
            "_kics_line": 3
          },
```

KICS Security Queries

> *We have the JSON* file

> We need to create **positive** and **negative cases**

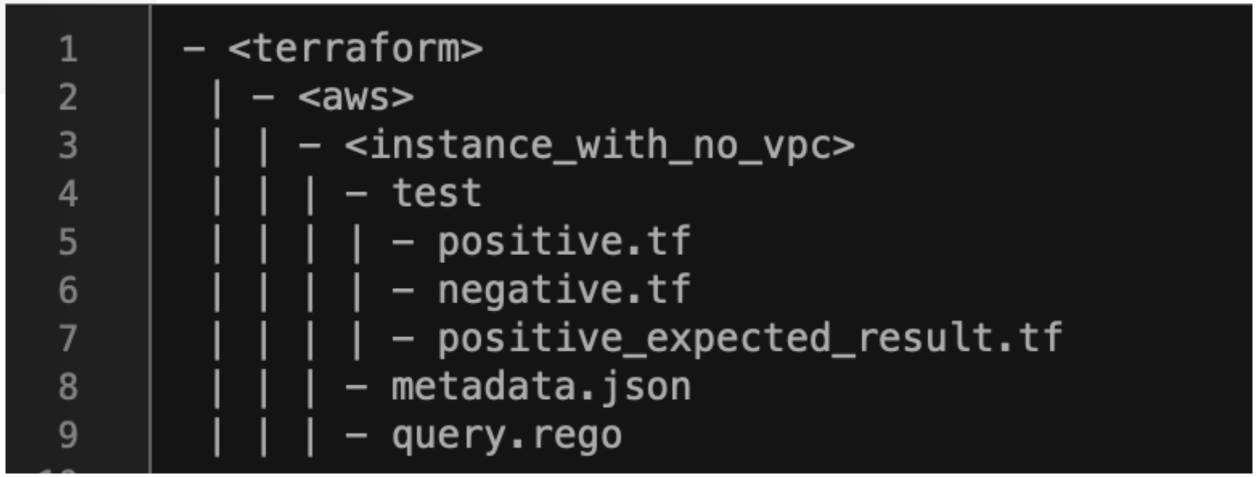
> We need to create out **metadata.json**

> We need to write our **security queries**

> **The security queries in KICS are written in rego**

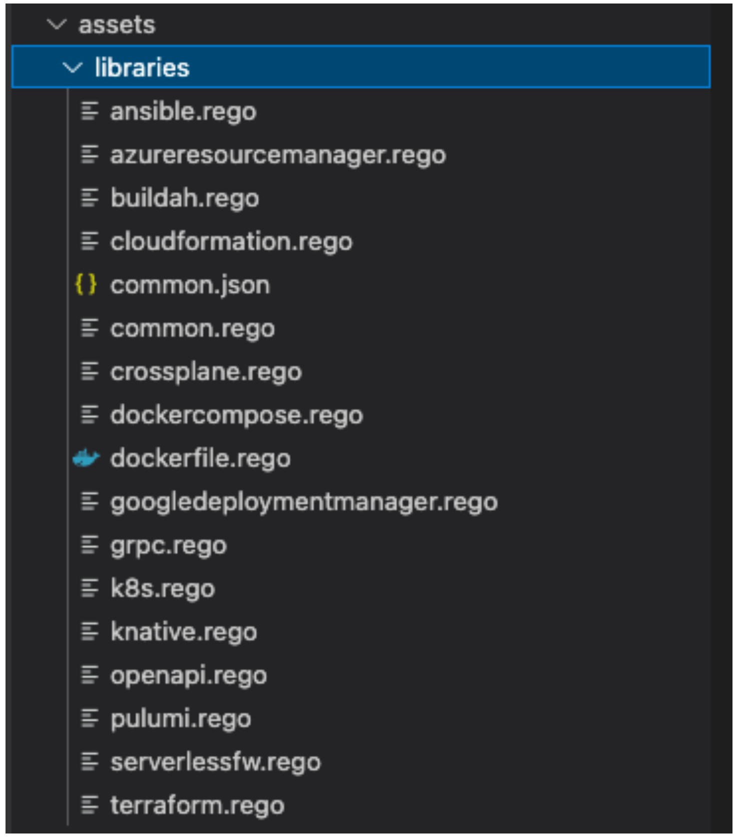
KICS Security Queries – *query file tree*

```
- <technology>
    |- <provider>
    |    |- <queryfolder>
    |    |    |- test
    |    |    |    |- positive1<.ext>
    |    |    |    |- positive2<.ext>
    |    |    |    |- negative1<.ext>
    |    |    |    |- negative2<.ext>
    |    |    |    |- positive_expected_result.json
    |    |    |- metadata.json
    |    |    |- query.rego
```

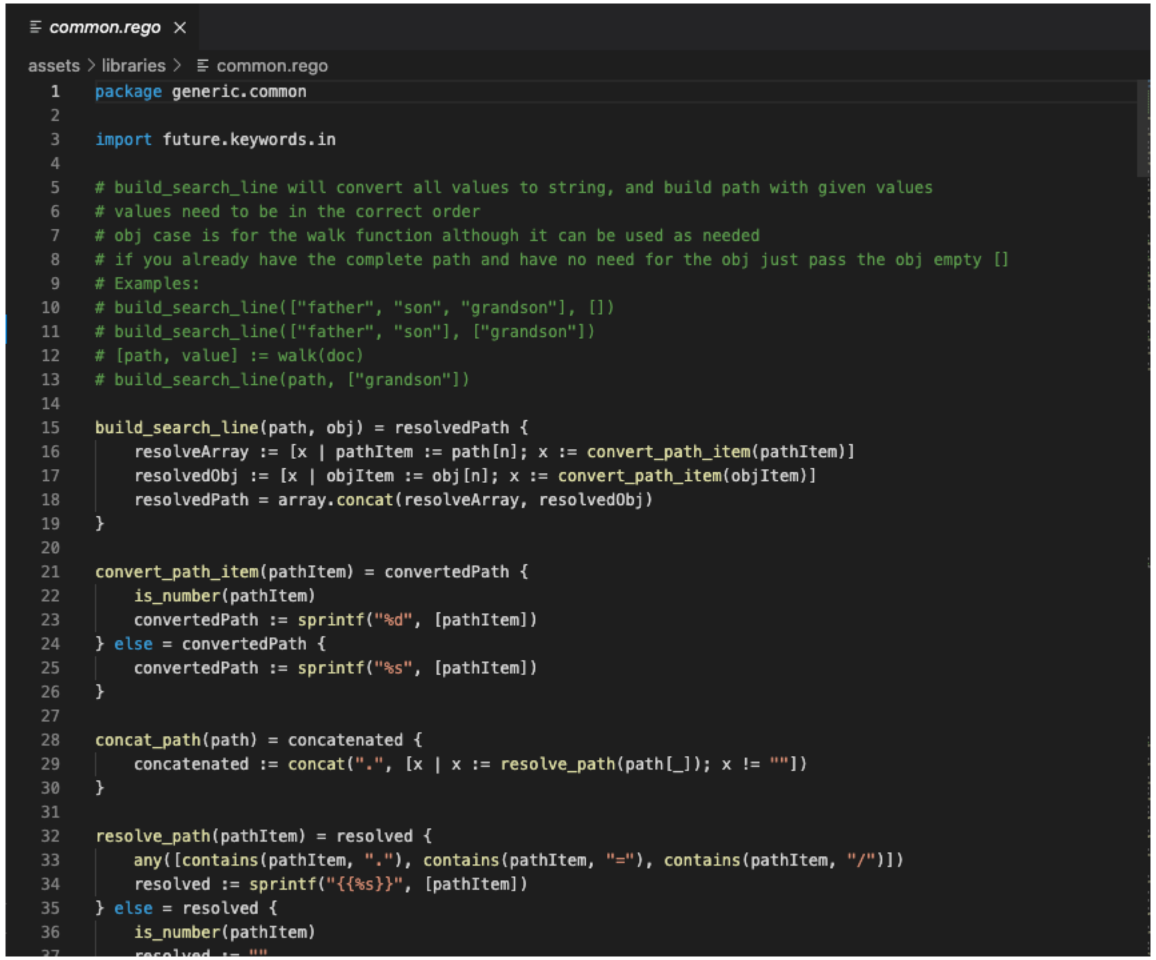


```
- <terraform>
 | - <aws>
 | | - <instance_with_no_vpc>
 | | | - test
 | | | | - positive.tf
 | | | | - negative.tf
 | | | | - positive_expected_result.tf
 | | | - metadata.json
 | | | - query.rego
```

KICS Security Queries – *libraries*



- assets
  - libraries
    - ansible.rego
    - azureresourcemanager.rego
    - buildah.rego
    - cloudformation.rego
    - common.json
    - common.rego
    - crossplane.rego
    - dockercompose.rego
    - dockerfile.rego
    - googledeploymentmanager.rego
    - grpc.rego
    - k8s.rego
    - knative.rego
    - openapi.rego
    - pulumi.rego
    - serverlessfw.rego
    - terraform.rego

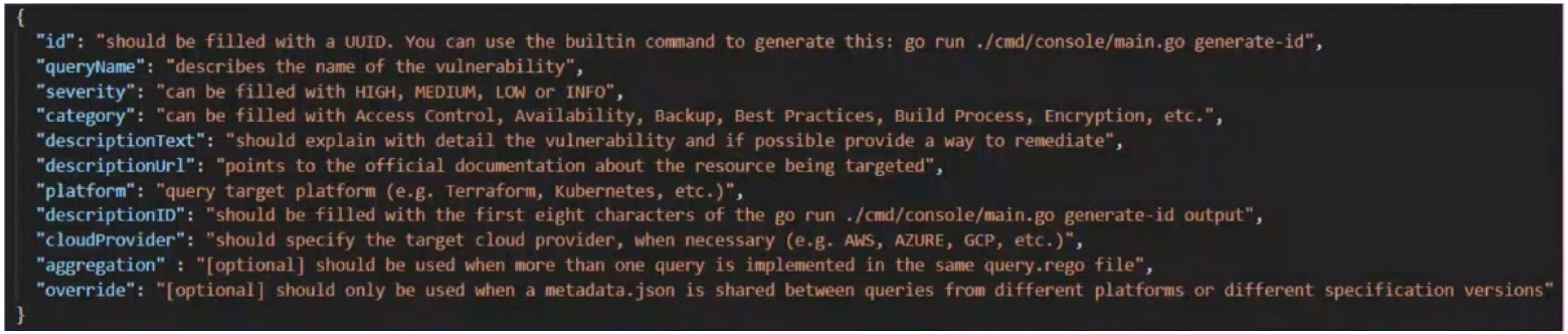


common.rego

assets > libraries > common.rego

```
package generic.common

import future.keywords.in

# build_search_line will convert all values to string, and build path with given values
# values need to be in the correct order
# obj case is for the walk function although it can be used as needed
# if you already have the complete path and have no need for the obj just pass the obj empty []
# Examples:
# build_search_line(["father", "son", "grandson"], [])
# build_search_line(["father", "son"], ["grandson"])
# [path, value] := walk(doc)
# build_search_line(path, ["grandson"])

build_search_line(path, obj) = resolvedPath {
    resolveArray := [x | pathItem := path[n]; x := convert_path_item(pathItem)]
    resolvedObj := [x | objItem := obj[n]; x := convert_path_item(objItem)]
    resolvedPath = array.concat(resolveArray, resolvedObj)
}

convert_path_item(pathItem) = convertedPath {
    is_number(pathItem)
    convertedPath := sprintf("%d", [pathItem])
} else = convertedPath {
    convertedPath := sprintf("%s", [pathItem])
}

concat_path(path) = concatenated {
    concatenated := concat(".", [x | x := resolve_path(path[_]); x != ""])
}

resolve_path(pathItem) = resolved {
    any([contains(pathItem, "."), contains(pathItem, "="), contains(pathItem, "/")])
    resolved := sprintf("{{%s}}", [pathItem])
} else = resolved {
    is_number(pathItem)
```

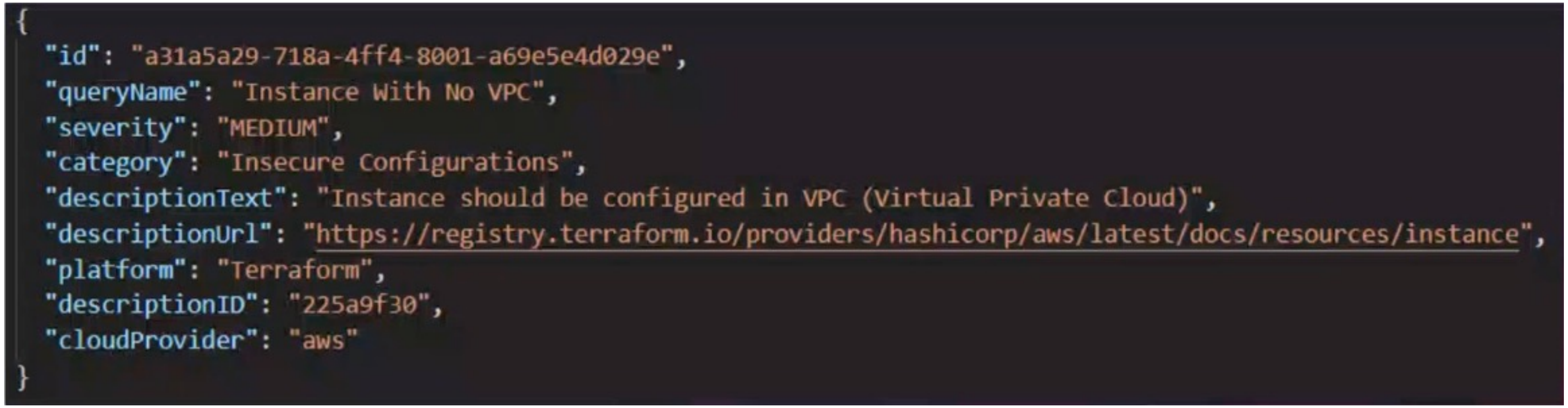
KICS Security Queries - *metadata*



```
{
  "id": "should be filled with a UUID. You can use the builtin command to generate this: go run ./cmd/console/main.go generate-id",
  "queryName": "describes the name of the vulnerability",
  "severity": "can be filled with HIGH, MEDIUM, LOW or INFO",
  "category": "can be filled with Access Control, Availability, Backup, Best Practices, Build Process, Encryption, etc.",
  "descriptionText": "should explain with detail the vulnerability and if possible provide a way to remediate",
  "descriptionUrl": "points to the official documentation about the resource being targeted",
  "platform": "query target platform (e.g. Terraform, Kubernetes, etc.)",
  "descriptionID": "should be filled with the first eight characters of the go run ./cmd/console/main.go generate-id output",
  "cloudProvider": "should specify the target cloud provider, when necessary (e.g. AWS, AZURE, GCP, etc.)",
  "aggregation" : "[optional] should be used when more than one query is implemented in the same query.rego file",
  "override": "[optional] should only be used when a metadata.json is shared between queries from different platforms or different specification versions"
}
```

KICS Security Queries - *metadata*

> Terraform AWS EC2 with no VPC example

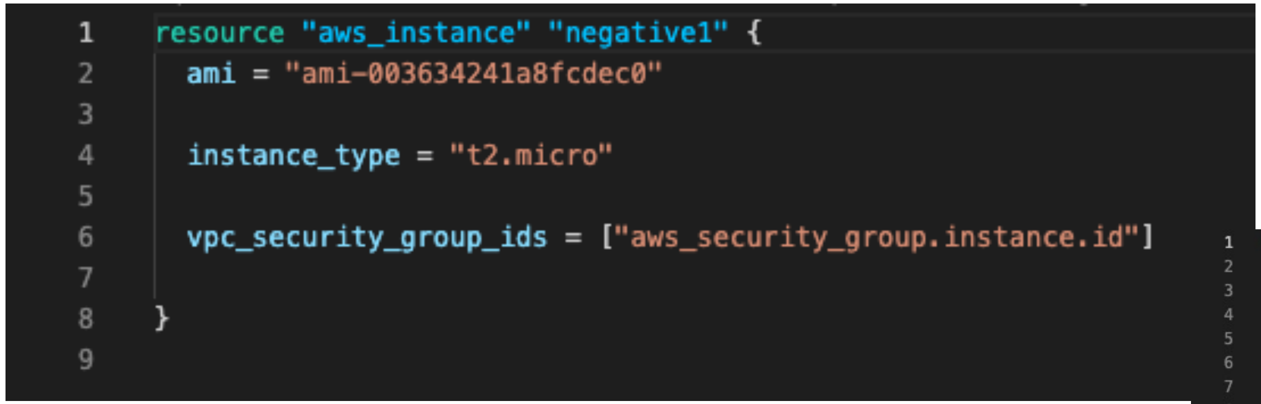


```
{
  "id": "a31a5a29-718a-4ff4-8001-a69e5e4d029e",
  "queryName": "Instance With No VPC",
  "severity": "MEDIUM",
  "category": "Insecure Configurations",
  "descriptionText": "Instance should be configured in VPC (Virtual Private Cloud)",
  "descriptionUrl": "https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/instance",
  "platform": "Terraform",
  "descriptionID": "225a9f30",
  "cloudProvider": "aws"
}
```

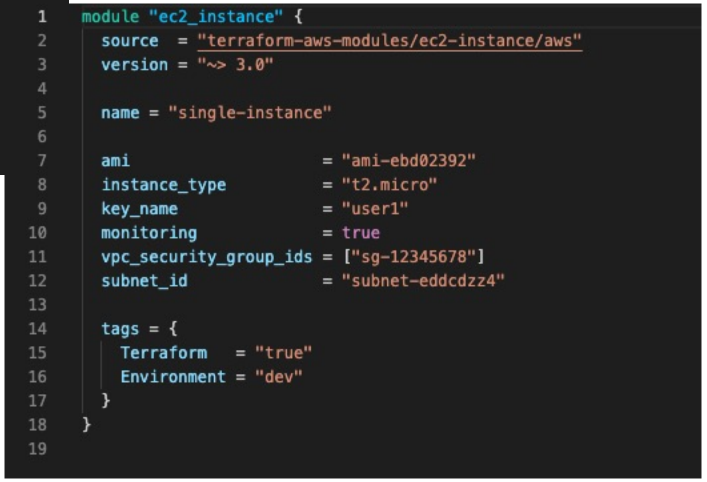
KICS Security Queries – *negative.tf*

> No result to be outputted

> The condition that needs to be verified so **there is no** vulnerability



```
resource "aws_instance" "negative1" {
  ami = "ami-003634241a8fcdec0"

  instance_type = "t2.micro"

  vpc_security_group_ids = ["aws_security_group.instance.id"]

}
```

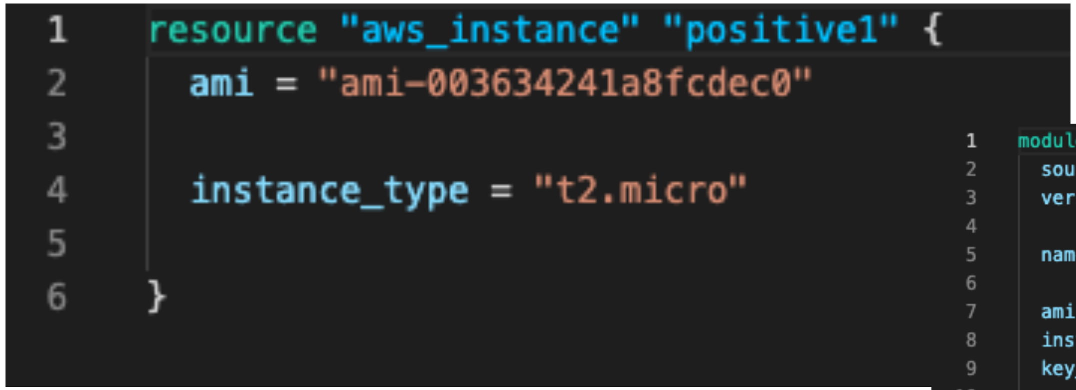


```
module "ec2_instance" {
  source  = "terraform-aws-modules/ec2-instance/aws"
  version = "~> 3.0"

  name = "single-instance"

  ami                    = "ami-ebd02392"
  instance_type          = "t2.micro"
  key_name               = "user1"
  monitoring             = true
  vpc_security_group_ids = ["sg-12345678"]
  subnet_id              = "subnet-eddcdzz4"

  tags = {
    Terraform   = "true"
    Environment = "dev"
  }
}
```

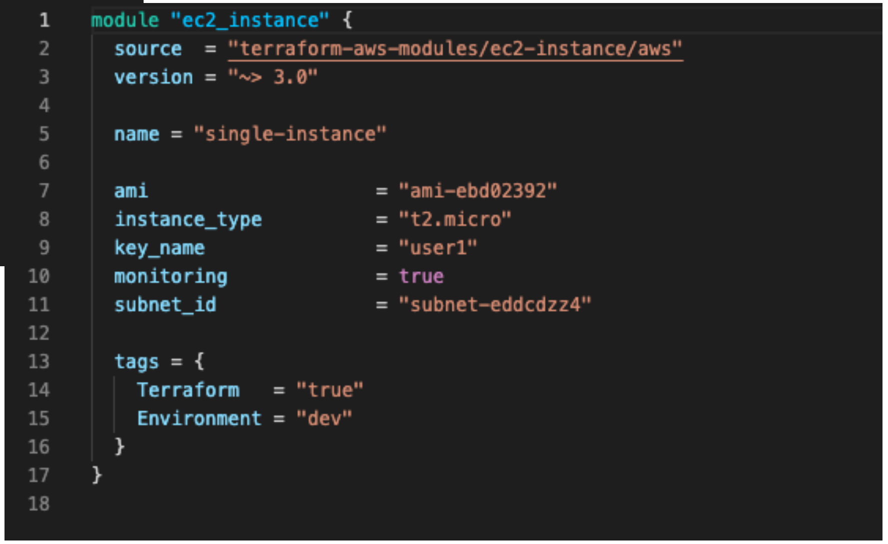
KICS Security Queries – *positive.tf*

> What outputs a result

> The condition that needs to be verified so **there is** a security vulnerability



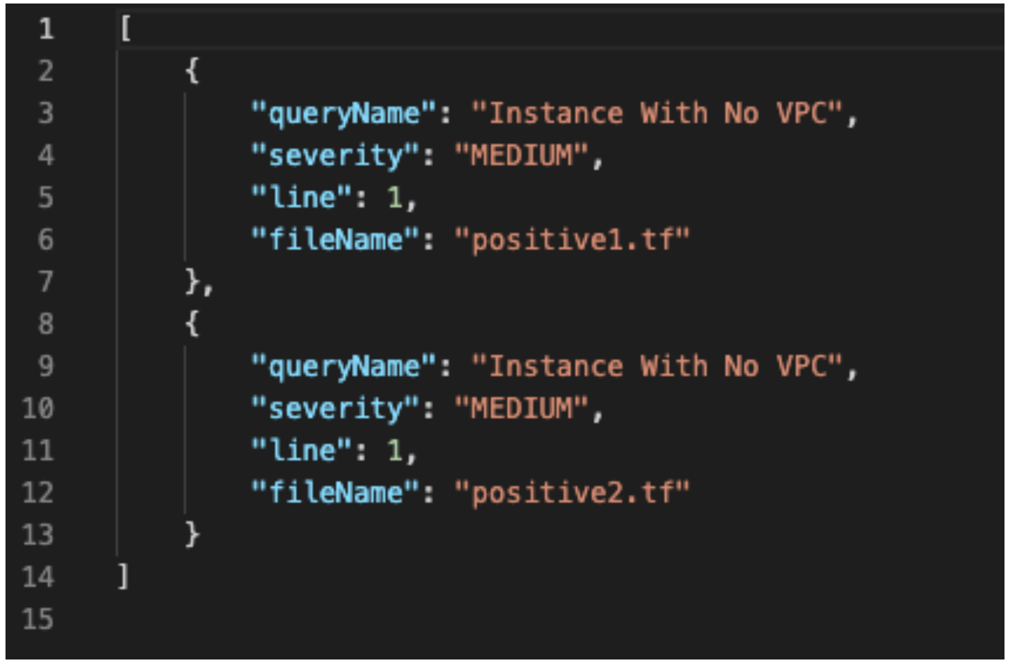
```
resource "aws_instance" "positive1" {
  ami = "ami-003634241a8fcdec0"

  instance_type = "t2.micro"

}
```



```
module "ec2_instance" {
  source  = "terraform-aws-modules/ec2-instance/aws"
  version = "~> 3.0"

  name = "single-instance"

  ami                    = "ami-ebd02392"
  instance_type          = "t2.micro"
  key_name               = "user1"
  monitoring             = true
  subnet_id              = "subnet-eddcdzz4"

  tags = {
    Terraform   = "true"
    Environment = "dev"
  }
}
```

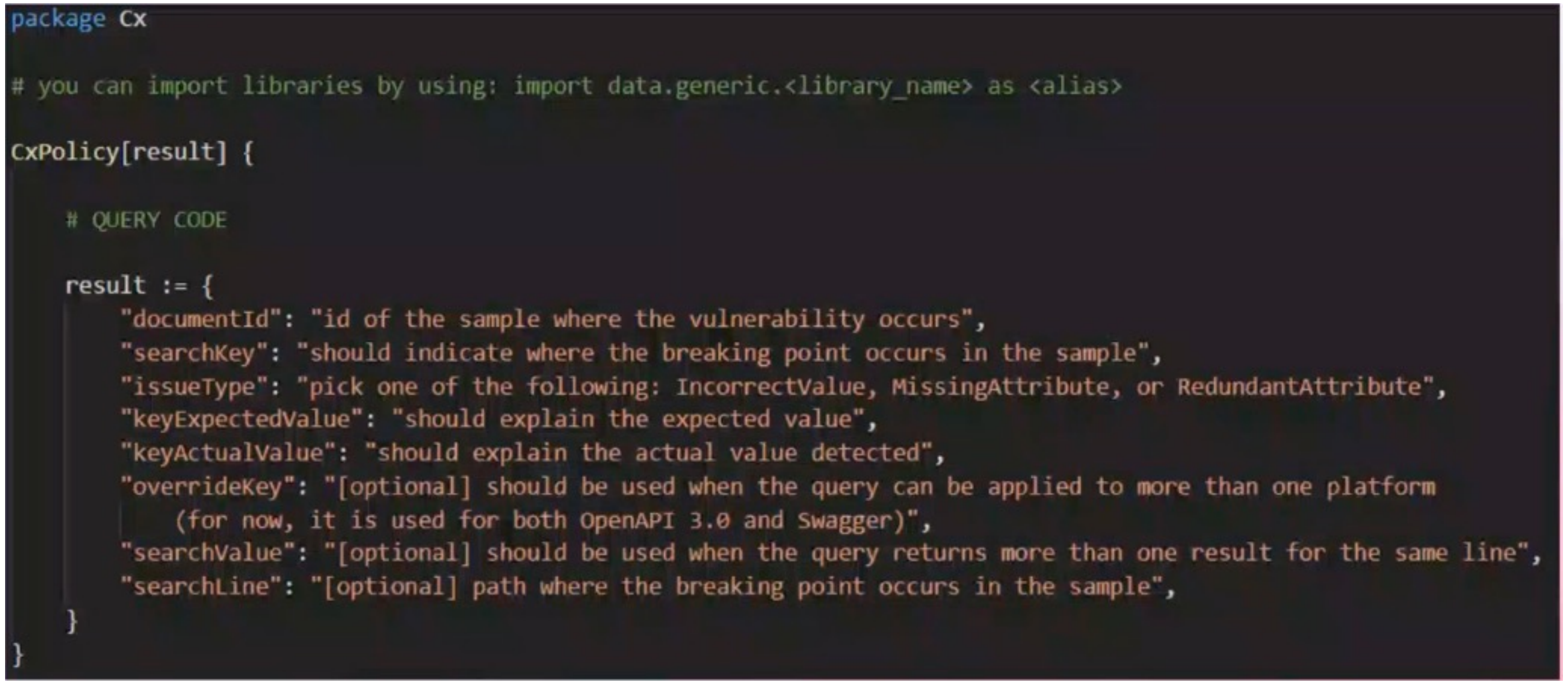
KICS Security Queries – *expected.tf*

> Is expected a. output being created is conditions on *positive_expected_result.json* are met – meaning is vulnerable



```json
[
    {
        "queryName": "Instance With No VPC",
        "severity": "MEDIUM",
        "line": 1,
        "fileName": "positive1.tf"
    },
    {
        "queryName": "Instance With No VPC",
        "severity": "MEDIUM",
        "line": 1,
        "fileName": "positive2.tf"
    }
]
```

KICS Security Queries – *query.rego*

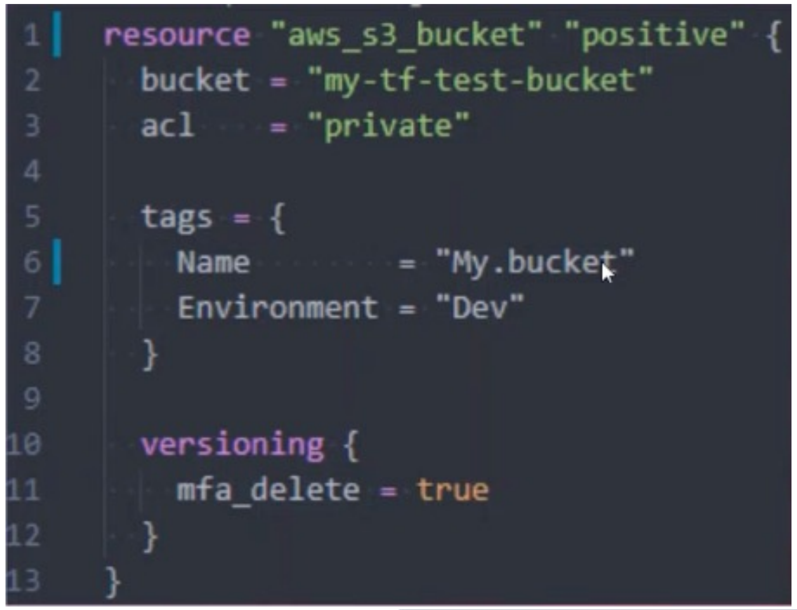


```
package Cx

# you can import libraries by using: import data.generic.<library_name> as <alias>

CxPolicy[result] {

    # QUERY CODE

    result := {
        "documentId": "id of the sample where the vulnerability occurs",
        "searchKey": "should indicate where the breaking point occurs in the sample",
        "issueType": "pick one of the following: IncorrectValue, MissingAttribute, or RedundantAttribute",
        "keyExpectedValue": "should explain the expected value",
        "keyActualValue": "should explain the actual value detected",
        "overrideKey": "[optional] should be used when the query can be applied to more than one platform
            (for now, it is used for both OpenAPI 3.0 and Swagger)",
        "searchValue": "[optional] should be used when the query returns more than one result for the same line",
        "searchLine": "[optional] path where the breaking point occurs in the sample",
    }
}
```

KICS Security Queries – *query.rego*

> Search key
  + String representation of what you are looking for in the file
  + KICS will look for it in the file

> Special charcaters
  + =
    - Is the value in key=value
  + .
    - Break line
  + {{}}
    - Ignore special characters
  + []
    - Same line



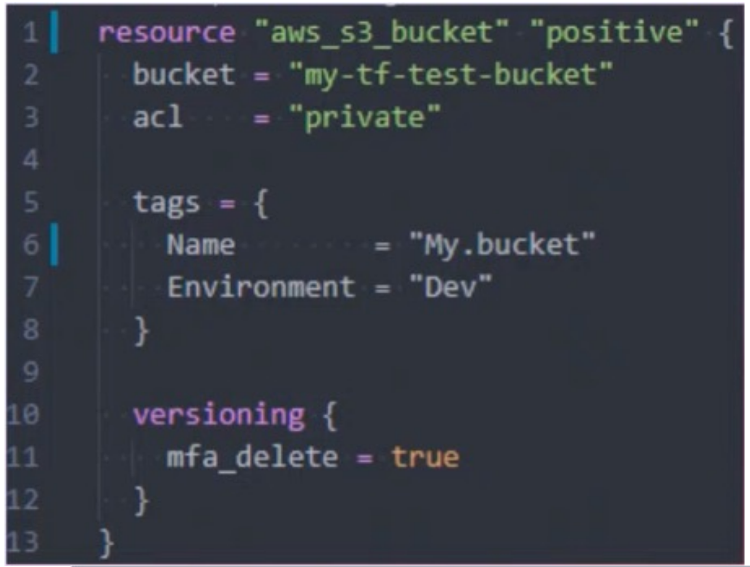
```
resource "aws_s3_bucket" "positive" {
  bucket = "my-tf-test-bucket"
  acl    = "private"

  tags = {
    Name        = "My.bucket"
    Environment = "Dev"
  }

  versioning {
    mfa_delete = true
  }
}
```

```
resource[positive].tags.Name={{My.bucket}}
```

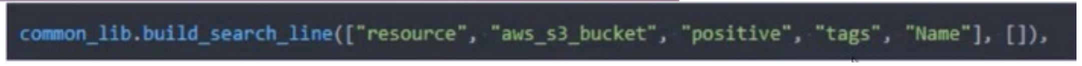
KICS Security Queries – *query.rego*

> Search line

+ Uses *JSON* path to find line
+ Uses common lib function build_search_line
  - It receives 2 arrays
  - First array is the path
  - Last array can be empty



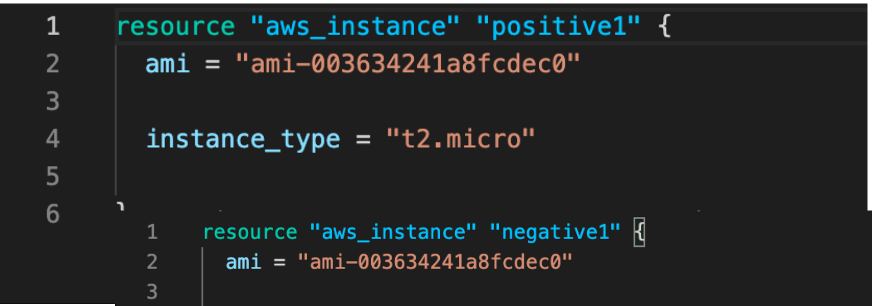
```
resource "aws_s3_bucket" "positive" {
  bucket = "my-tf-test-bucket"
  acl    = "private"

  tags = {
    Name        = "My.bucket"
    Environment = "Dev"
  }

  versioning {
    mfa_delete = true
  }
}
```



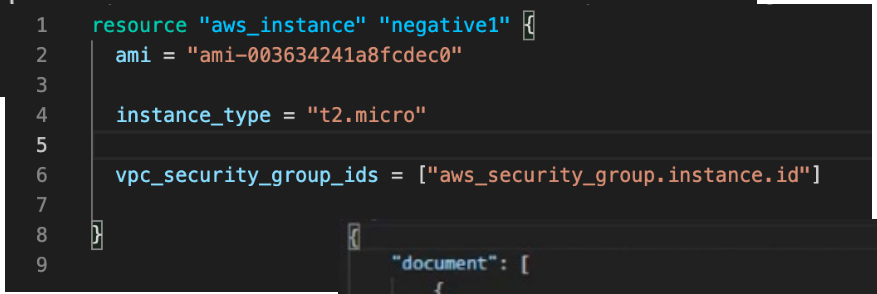
```
common_lib.build_search_line(["resource", "aws_s3_bucket", "positive", "tags", "Name"], []),
```

KICS Security Queries – *query.rego*

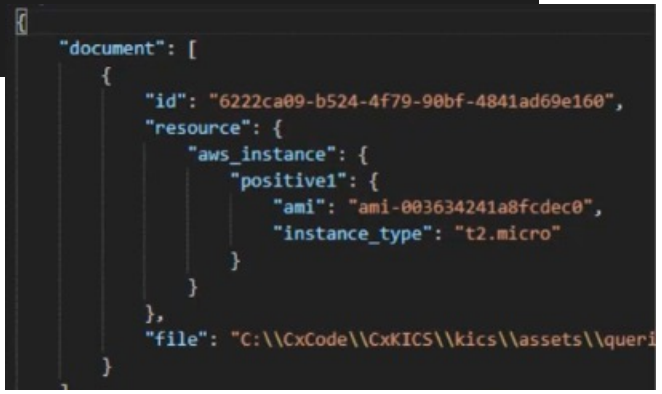
> In our Instance with no VPC example



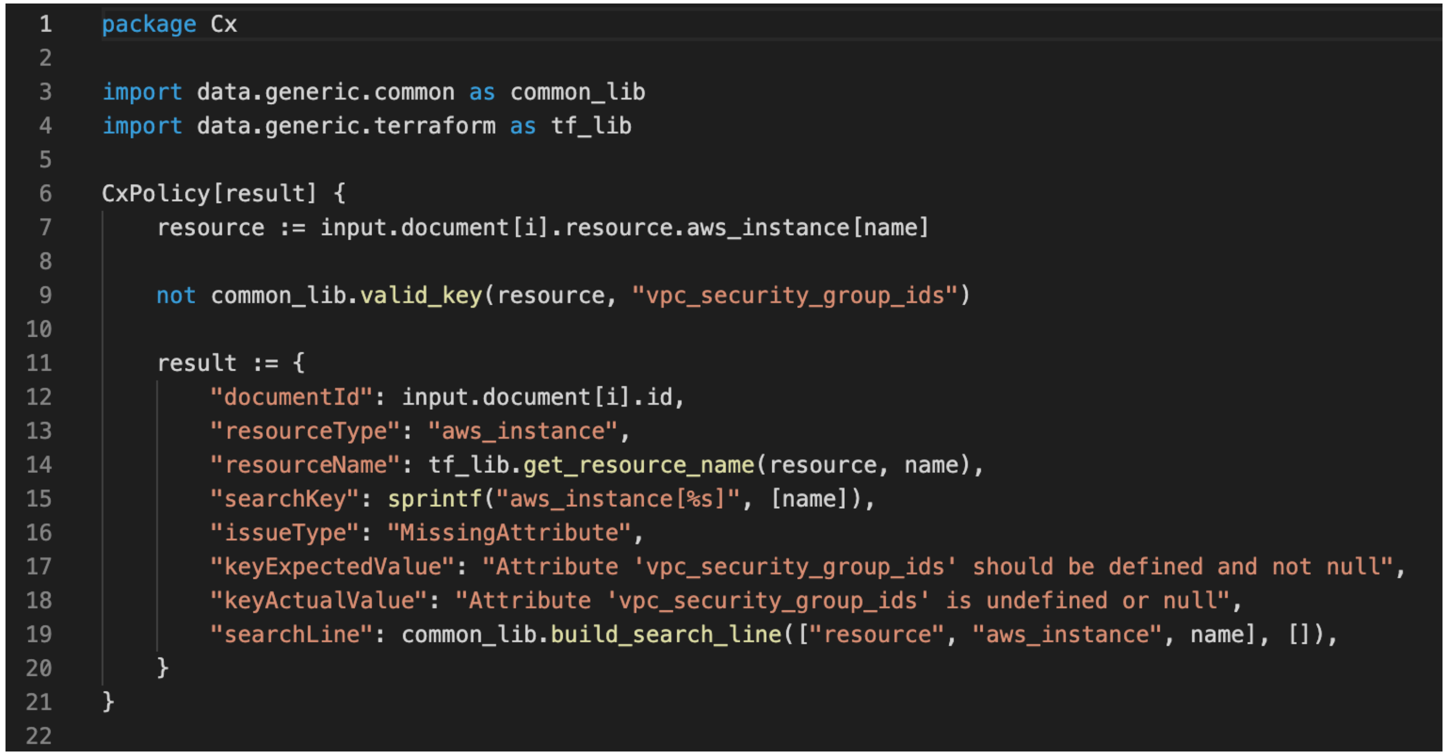
```
resource "aws_instance" "positive1" {
  ami = "ami-003634241a8fcdec0"

  instance_type = "t2.micro"

}
```



```
resource "aws_instance" "negative1" {
  ami = "ami-003634241a8fcdec0"

  instance_type = "t2.micro"

  vpc_security_group_ids = ["aws_security_group.instance.id"]

}
```



```
{
  "document": [
    {
      "id": "6222ca09-b524-4f79-90bf-4841ad69e160",
      "resource": {
        "aws_instance": {
          "positive1": {
            "ami": "ami-003634241a8fcdec0",
            "instance_type": "t2.micro"
          }
        }
      },
      "file": "C:\\CxCode\\CxKICS\\kics\\assets\\queri
    }
```



```
package Cx

import data.generic.common as common_lib
import data.generic.terraform as tf_lib

CxPolicy[result] {
    resource := input.document[i].resource.aws_instance[name]

    not common_lib.valid_key(resource, "vpc_security_group_ids")

    result := {
        "documentId": input.document[i].id,
        "resourceType": "aws_instance",
        "resourceName": tf_lib.get_resource_name(resource, name),
        "searchKey": sprintf("aws_instance[%s]", [name]),
        "issueType": "MissingAttribute",
        "keyExpectedValue": "Attribute 'vpc_security_group_ids' should be defined and not null",
        "keyActualValue": "Attribute 'vpc_security_group_ids' is undefined or null",
        "searchLine": common_lib.build_search_line(["resource", "aws_instance", name], []),
    }
}
```

KICS Security Queries – *useful functions*

> valid_key (obj, key)
  + Receives object and key that needs to be checked
  + Checks if the key exists or not

> json_unmarshall(json)
  + Deserializes a *JSON* encoded string to a term
  + Receives a *JSON* string as argument
  + Returns an empty *JSON* is there is no string

> walk(x, [path, value])
  + Creates a path and value pair for documents under x – x is the resource (an array)

KICS Security Queries *DEMO*

- > Create a query that finds Terraform resources with encryption disabled
  - + Encryption should be set to false
  - + Use walk function
- > Resources
  - + query_development_training_encryption_disabled zip
  - + Docker
  - + Rego Playground to test the code, https://play.openpolicyagent.org/

KICS Security Queries *DEMO*

> Create the query with the help of rego playground

- Edit the /query_development_training/query_development_training_encryption_disabled/query.rego based on /query_development_training/payload.jsonpayload.json

> Get docker image of KICS and execute the query

```
docker pull checkmarx/kics:latest
docker run -v /YOUR_PATH_TO_THE_QUERY:/query checkmarx/kics:latest scan -p /query -q /query --no-progress
```

Note: to verbose execution add –v in the end

Note2: to create a report add –o /query --report-formats "pdf”

Note3: to create a JSON respresentation

KICS Security Queries – *important links*

> https://github.com/Checkmarx/kics/blob/master/docs/queries.md

> https://docs.kics.io/latest/queries/

KICS Support

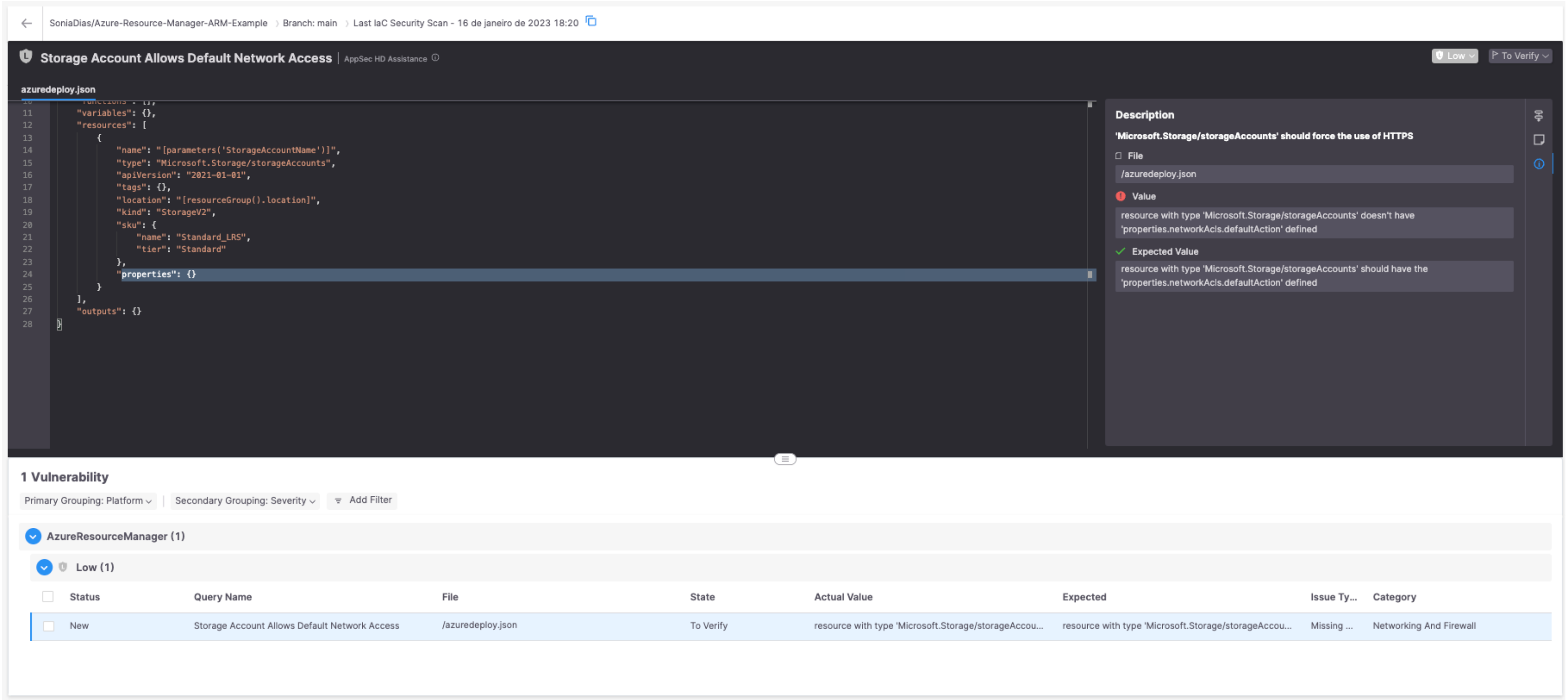
- https://github.com/Checkmarx/kics/issues
  - https://github.com/Checkmarx/kics/issues/5559#issuecomment-1246576485

- https://github.com/Checkmarx/kics/discussions

KICS *how to contribute?*

> https://docs.kics.io/latest/CONTRIBUTING/

KICS in CxOne



SoniaDias/Azure-Resource-Manager-ARM-Example 〉 Branch: main 〉 Last IaC Security Scan - 16 de janeiro de 2023 18:20

Storage Account Allows Default Network Access | AppSec HD Assistance

azuredeploy.json

```
    "variables": {},
    "resources": [
        {
            "name": "[parameters('StorageAccountName')]",
            "type": "Microsoft.Storage/storageAccounts",
            "apiVersion": "2021-01-01",
            "tags": {},
            "location": "[resourceGroup().location]",
            "kind": "StorageV2",
            "sku": {
                "name": "Standard_LRS",
                "tier": "Standard"
            },
            "properties": {}
        }
    ],
    "outputs": {}
}
```

Description

'Microsoft.Storage/storageAccounts' should force the use of HTTPS

File

/azuredeploy.json

Value

resource with type 'Microsoft.Storage/storageAccounts' doesn't have 'properties.networkAcls.defaultAction' defined

Expected Value

resource with type 'Microsoft.Storage/storageAccounts' should have the 'properties.networkAcls.defaultAction' defined

1 Vulnerability

AzureResourceManager (1)

Low (1)

Status	Query Name	File	State	Actual Value	Expected	Issue Ty...	Category
New	Storage Account Allows Default Network Access	/azuredeploy.json	To Verify	resource with type 'Microsoft.Storage/storageAccou...	resource with type 'Microsoft.Storage/storageAccou...	Missing ...	Networking And Firewall

Checkmarx

The world runs on code. We secure it.

Checkmarx

Checkmarx

Thank you

Questions. Feedback. Contact details.

Take your application security to the next level
checkmarx.com

Follow us on
LinkedIn/checkmarx and Twitter/checkmarx