Checkov

Introduction

Checkov is an open-source static code analysis tool designed to identify security and compliance issues in cloud infrastructure as code (IaC) templates and configuration files. Developed by Bridgecrew, it offers automated scanning capabilities to help ensure that cloud environments adhere to best practices, security standards, and compliance regulations.

Checkov offers several key benefits:

Automated Compliance Checking

It scans infrastructure as code templates and configuration files to uncover violations of security best practices, industry standards, and regulatory requirements like GDPR, HIPAA, and PCI-DSS.

Early Detection of Misconfigurations

By analyzing IaC files during development and deployment, Checkov aids in identifying misconfigurations and security weaknesses before they reach production, reducing the risk of security incidents and data breaches.

Integration with CI/CD Pipelines

Seamless integration with CI/CD pipelines enables automated security and compliance checks throughout the development process, ensuring security is ingrained in every step.

Scalability and Extensibility

Checkov supports various cloud platforms, including AWS, Azure, Google Cloud, Kubernetes, and Terraform, and offers extensibility through plugins and custom policies, enabling tailored solutions for diverse environments.

Cost-Efficiency

By addressing security and compliance issues early on, Checkov helps organizations avoid expensive remediation efforts, regulatory fines, and reputational damage associated with security incidents.

To install Checkov on Ubuntu, you can follow these steps:

Ensure that AWS CLI is installed and configured, along with Terraform.

Update the package index:

```
sudo apt update
```

Install the software-properties-common package:

```
sudo apt install software-properties-common
```

Install Python 3

```
sudo apt install python3
```

Install pip for Python 3

```
sudo apt install python3-pip
```

Upgrade pip to the latest version if already installed

```
sudo python3  -m pip install -U pip
```

Check if Python and pip are installed on your system by running the following commands

```
python --version
pip --version
```

This sequence of commands ensures that your system is up to date, installs the required dependencies, and then installs Checkov using Python 3 and pip.

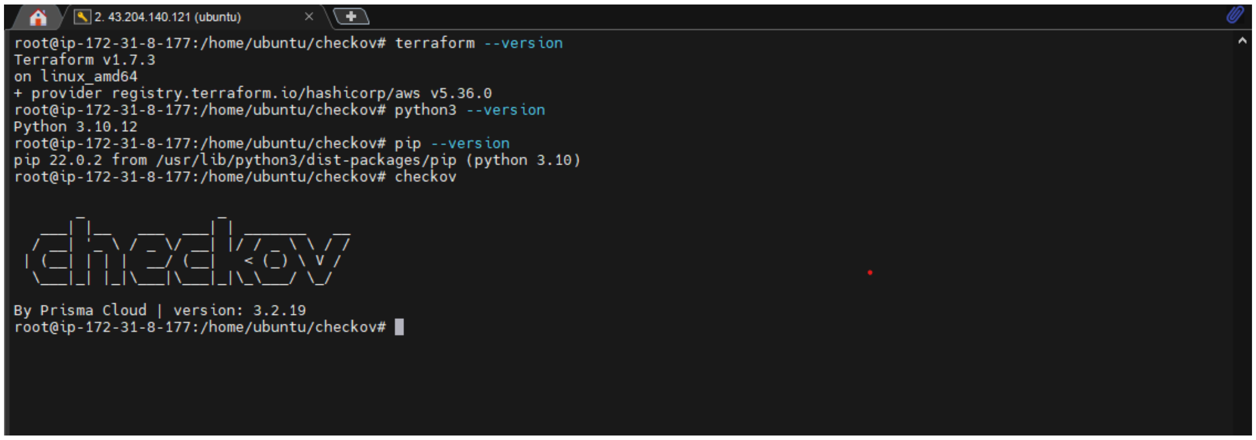
Finally, install Checkov using pip:

```
sudo python3.7 -m pip install -U checkov
```

Validate the installation of Checkov by executing the command:

Checkov

This command will confirm whether Checkov is properly installed and ready for use.

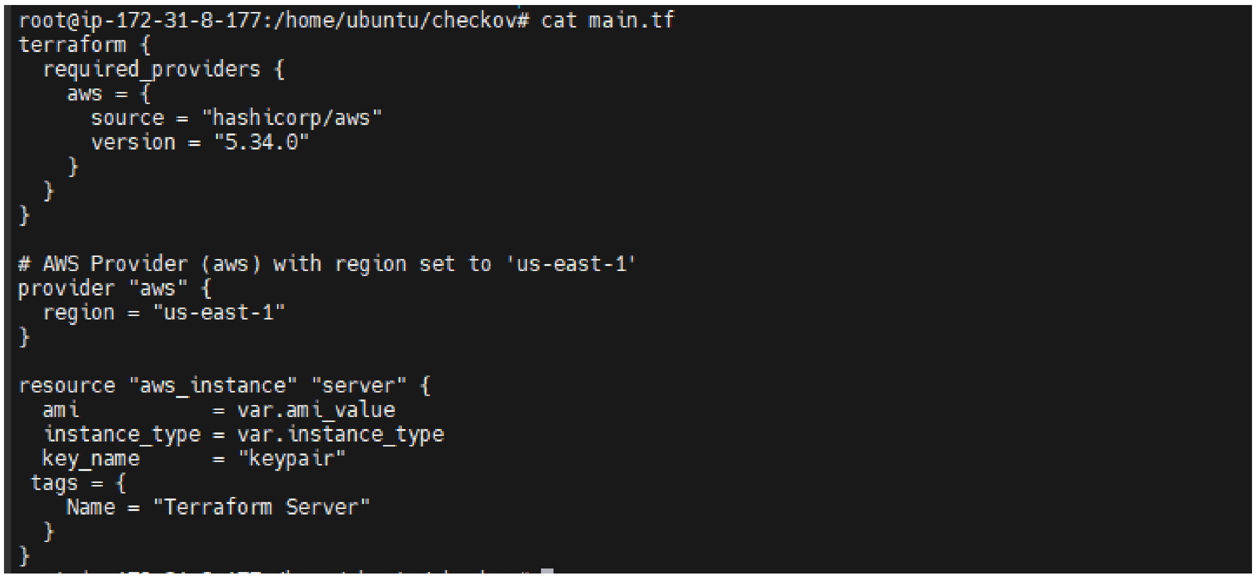


```
root@ip-172-31-8-177:/home/ubuntu/checkov# terraform --version
Terraform v1.7.3
on linux_amd64
+ provider registry.terraform.io/hashicorp/aws v5.36.0
root@ip-172-31-8-177:/home/ubuntu/checkov# python3 --version
Python 3.10.12
root@ip-172-31-8-177:/home/ubuntu/checkov# pip --version
pip 22.0.2 from /usr/lib/python3/dist-packages/pip (python 3.10)
root@ip-172-31-8-177:/home/ubuntu/checkov# checkov

       _               _
   ___| |__   ___  ___| | _______   __
  / __| '_ \ / _ \/ __| |/ / _ \ \ / /
 | (__| | | |  __/ (__|   < (_) \ V /
  \___|_| |_|\___|\___|_|\_\___/ \_/

By Prisma Cloud | version: 3.2.19
root@ip-172-31-8-177:/home/ubuntu/checkov#
```

Simple usage example to demonstrate how Checkov works

EC2 Configuration:

Consider the configuration of an ec2 instance as represented in the Terraform sample below.



```
root@ip-172-31-8-177:/home/ubuntu/checkov# cat main.tf
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.34.0"
    }
  }
}

# AWS Provider (aws) with region set to 'us-east-1'
provider "aws" {
  region = "us-east-1"
}

resource "aws_instance" "server" {
  ami           = var.ami_value
  instance_type = var.instance_type
  key_name      = "keypair"
 tags = {
    Name = "Terraform Server"
  }
}
```

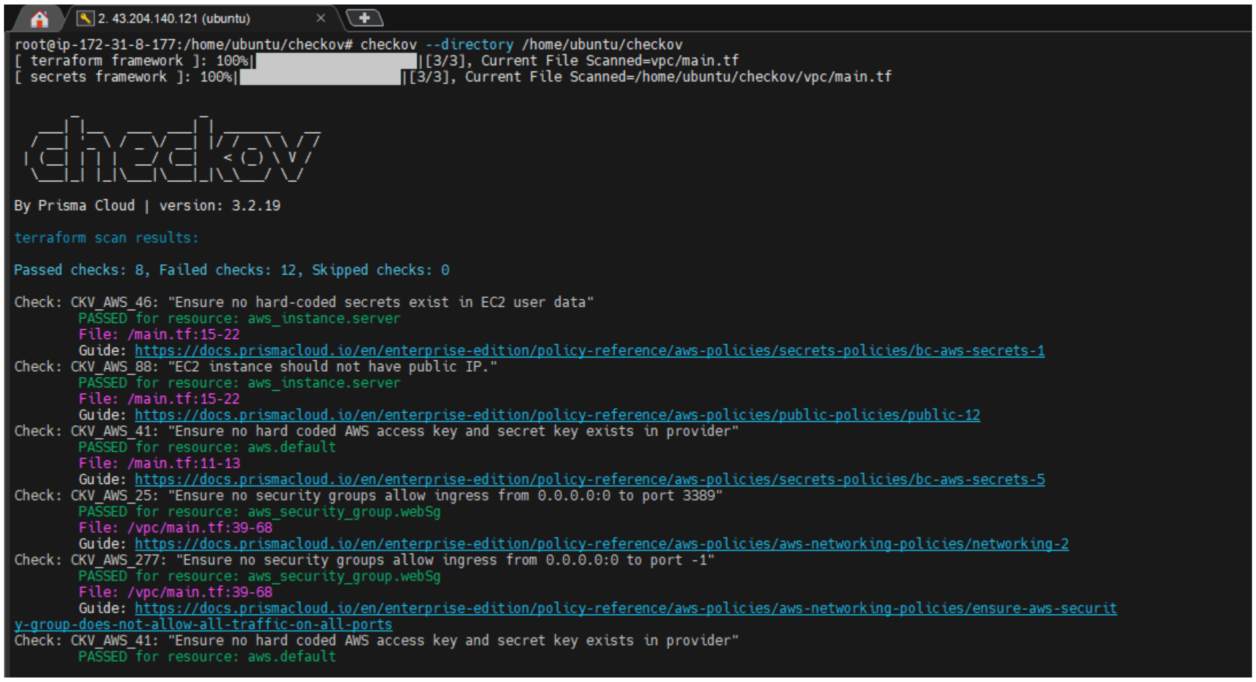
Running Checkov:

Before provisioning lets run Checkov against template, navigate to the directory containing the file and execute the checkov command:

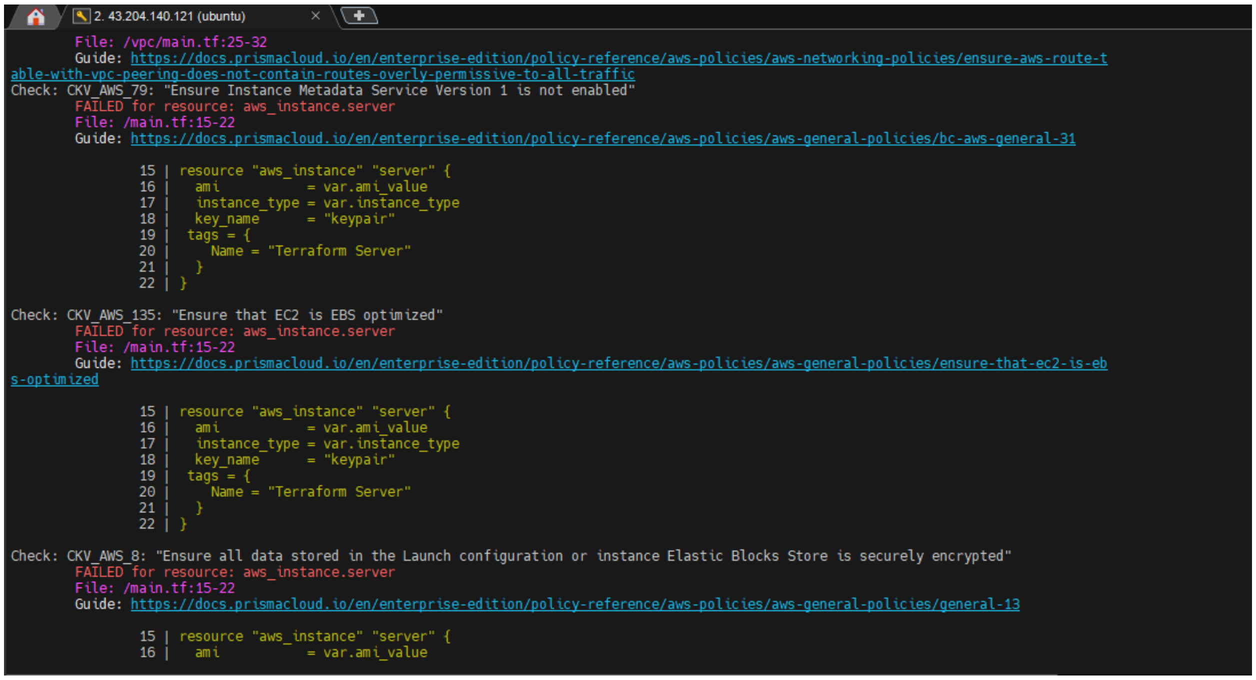
To Configure a folder:

```
checkov --directory /home/ubuntu/checkov
```

After running **checkov --directory /home/ubuntu/checkov**, the output will display the results of the Checkov analysis for the files found in the /home/ubuntu/checkov directory. The output typically includes information about any security and compliance issues detected in the analyzed files. As shown below



```
root@ip-172-31-8-177:/home/ubuntu/checkov# checkov --directory /home/ubuntu/checkov
[ terraform framework ]: 100%|████████████████████|[3/3], Current File Scanned=vpc/main.tf
[ secrets framework ]: 100%|████████████████████|[3/3], Current File Scanned=/home/ubuntu/checkov/vpc/main.tf


       _               _
   ___| |__   ___  ___| | _______   __
  / __| '_ \ / _ \/ __| |/ / _ \ \ / /
 | (__| | | |  __/ (__|   < (_) \ V /
  \___|_| |_|\___|\___|_|\_\___/ \_/

By Prisma Cloud | version: 3.2.19

terraform scan results:

Passed checks: 8, Failed checks: 12, Skipped checks: 0

Check: CKV_AWS_46: "Ensure no hard-coded secrets exist in EC2 user data"
        PASSED for resource: aws_instance.server
        File: /main.tf:15-22
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-policies/secrets-policies/bc-aws-secrets-1
Check: CKV_AWS_88: "EC2 instance should not have public IP."
        PASSED for resource: aws_instance.server
        File: /main.tf:15-22
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-policies/public-policies/public-12
Check: CKV_AWS_41: "Ensure no hard coded AWS access key and secret key exists in provider"
        PASSED for resource: aws.default
        File: /main.tf:11-13
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-policies/secrets-policies/bc-aws-secrets-5
Check: CKV_AWS_25: "Ensure no security groups allow ingress from 0.0.0.0:0 to port 3389"
        PASSED for resource: aws_security_group.webSg
        File: /vpc/main.tf:39-68
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-policies/aws-networking-policies/networking-2
Check: CKV_AWS_277: "Ensure no security groups allow ingress from 0.0.0.0:0 to port -1"
        PASSED for resource: aws_security_group.webSg
        File: /vpc/main.tf:39-68
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-policies/aws-networking-policies/ensure-aws-securit
y-group-does-not-allow-all-traffic-on-all-ports
Check: CKV_AWS_41: "Ensure no hard coded AWS access key and secret key exists in provider"
        PASSED for resource: aws.default
```



```
        File: /vpc/main.tf:25-32
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-policies/aws-networking-policies/ensure-aws-route-t
able-with-vpc-peering-does-not-contain-routes-overly-permissive-to-all-traffic
Check: CKV_AWS_79: "Ensure Instance Metadata Service Version 1 is not enabled"
        FAILED for resource: aws_instance.server
        File: /main.tf:15-22
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-policies/aws-general-policies/bc-aws-general-31

                15 | resource "aws_instance" "server" {
                16 |   ami           = var.ami_value
                17 |   instance_type = var.instance_type
                18 |   key_name      = "keypair"
                19 |  tags = {
                20 |     Name = "Terraform Server"
                21 |   }
                22 | }

Check: CKV_AWS_135: "Ensure that EC2 is EBS optimized"
        FAILED for resource: aws_instance.server
        File: /main.tf:15-22
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-policies/aws-general-policies/ensure-that-ec2-is-eb
s-optimized

                15 | resource "aws_instance" "server" {
                16 |   ami           = var.ami_value
                17 |   instance_type = var.instance_type
                18 |   key_name      = "keypair"
                19 |  tags = {
                20 |     Name = "Terraform Server"
                21 |   }
                22 | }

Check: CKV_AWS_8: "Ensure all data stored in the Launch configuration or instance Elastic Blocks Store is securely encrypted"
        FAILED for resource: aws_instance.server
        File: /main.tf:15-22
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-policies/aws-general-policies/general-13

                15 | resource "aws_instance" "server" {
                16 |   ami           = var.ami_value
```

Overall, the output of checkov --directory /home/ubuntu/checkov provides valuable insights into the security and compliance posture of the infrastructure as code files located in the specified directory, helping users identify and address potential risks and vulnerabilities effectively.

VPC Configuration:

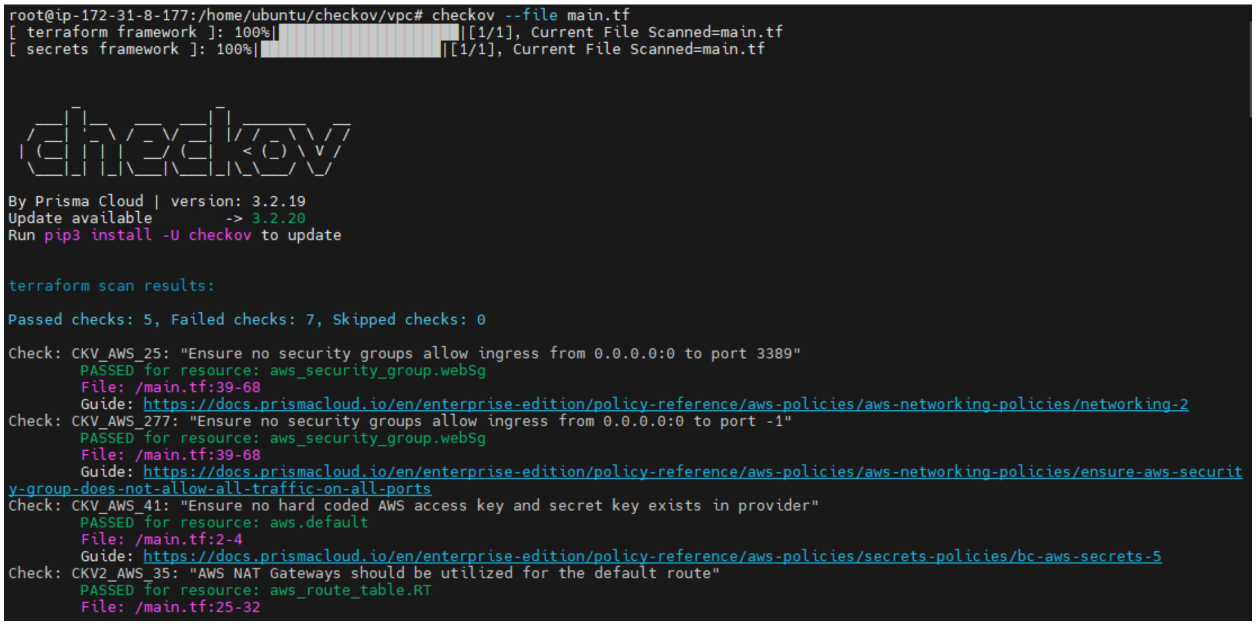
Similarly lets run Checkov against template aws vpc, navigate to the directory containing the file and execute the checkov command:

Running Checkov:

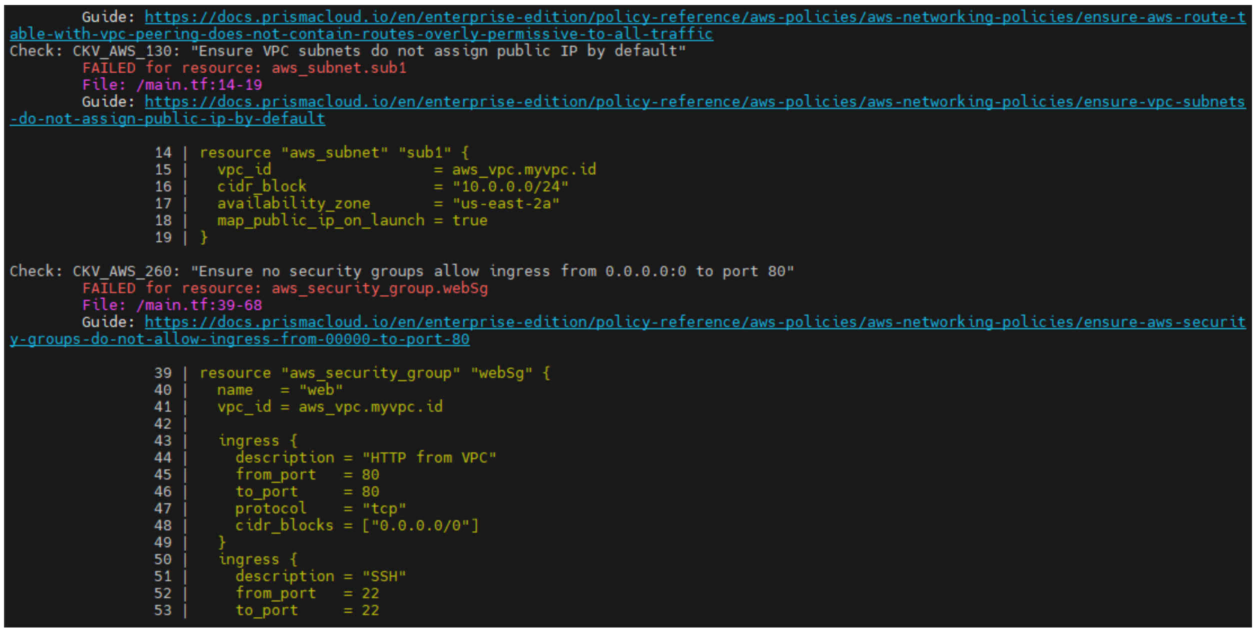
To Configure a folder:

checkov --file /home/ubuntu/checkov/vpc/main.tf

After running checkov --file /home/ubuntu/checkov/vpc/main.tf, the output will display the results of the Checkov analysis for the files found in the /home/ubuntu/checkov directory. The output typically includes information about any security and compliance issues detected in the analyzed files. As shown below



```
root@ip-172-31-8-177:/home/ubuntu/checkov/vpc# checkov --file main.tf
[ terraform framework ]: 100%|████████████████████|[1/1], Current File Scanned=main.tf
[ secrets framework ]: 100%|████████████████████|[1/1], Current File Scanned=main.tf


       _               _
   ___| |__   ___  ___| | _______   __
  / __| '_ \ / _ \/ __| |/ / _ \ \ / /
 | (__| | | |  __/ (__|   < (_) \ V /
  \___|_| |_|\___|\___|_|\_\___/ \_/

By Prisma Cloud | version: 3.2.19
Update available 3.2.19 -> 3.2.20
Run pip3 install -U checkov to update


terraform scan results:

Passed checks: 5, Failed checks: 7, Skipped checks: 0

Check: CKV_AWS_25: "Ensure no security groups allow ingress from 0.0.0.0:0 to port 3389"
        PASSED for resource: aws_security_group.webSg
        File: /main.tf:39-68
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-policies/aws-networking-policies/networking-2
Check: CKV_AWS_277: "Ensure no security groups allow ingress from 0.0.0.0:0 to port -1"
        PASSED for resource: aws_security_group.webSg
        File: /main.tf:39-68
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-policies/aws-networking-policies/ensure-aws-securit
y-group-does-not-allow-all-traffic-on-all-ports
Check: CKV_AWS_41: "Ensure no hard coded AWS access key and secret key exists in provider"
        PASSED for resource: aws.default
        File: /main.tf:2-4
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-policies/secrets-policies/bc-aws-secrets-5
Check: CKV2_AWS_35: "AWS NAT Gateways should be utilized for the default route"
        PASSED for resource: aws_route_table.RT
        File: /main.tf:25-32
```



```
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-policies/aws-networking-policies/ensure-aws-route-table-with-vpc-peering-does-not-contain-routes-overly-permissive-to-all-traffic
Check: CKV_AWS_130: "Ensure VPC subnets do not assign public IP by default"
        FAILED for resource: aws_subnet.sub1
        File: /main.tf:14-19
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-policies/aws-networking-policies/ensure-vpc-subnets-do-not-assign-public-ip-by-default

                14 | resource "aws_subnet" "sub1" {
                15 |   vpc_id                  = aws_vpc.myvpc.id
                16 |   cidr_block              = "10.0.0.0/24"
                17 |   availability_zone       = "us-east-2a"
                18 |   map_public_ip_on_launch = true
                19 | }

Check: CKV_AWS_260: "Ensure no security groups allow ingress from 0.0.0.0:0 to port 80"
        FAILED for resource: aws_security_group.webSg
        File: /main.tf:39-68
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-policies/aws-networking-policies/ensure-aws-security-groups-do-not-allow-ingress-from-00000-to-port-80

                39 | resource "aws_security_group" "webSg" {
                40 |   name   = "web"
                41 |   vpc_id = aws_vpc.myvpc.id
                42 |
                43 |   ingress {
                44 |     description = "HTTP from VPC"
                45 |     from_port   = 80
                46 |     to_port     = 80
                47 |     protocol    = "tcp"
                48 |     cidr_blocks = ["0.0.0.0/0"]
                49 |   }
                50 |   ingress {
                51 |     description = "SSH"
                52 |     from_port   = 22
                53 |     to_port     = 22
```

Summery:

terraform scan results show:
Passed checks: 5, Failed checks: 7, Skipped checks: 0
This summary indicates the overall outcome of the scan, showing the number of checks that passed, failed, and were skipped.

So below are the few of the individual checks:

Passed checks:
Passed for resource: aws_security_group.webSg
Passed for resource: aws.default
Passed for resource: aws_route_table.RT

Failed checks:
Failed for resource: aws_subnet.sub1
Failed for resource: aws_security_group.webSg
Failed for resource: aws_security_group.webSg

So the output provides a detailed overview of the Checkov scan results, indicating which checks passed and which failed, along with additional information and links to guides for further reference.

Supported IaC types

Checkov scans these IaC file types:

- Terraform (for AWS, GCP, Azure and OCI)
- CloudFormation (including AWS SAM)
- Azure Resource Manager (ARM)
- Serverless framework
- Helm charts
- Kubernetes
- Docker

Custom policies

Custom policies can be created to check cloud resources based on configuration attributes (in Python or YAML or connection states (in YAML). For composite policies, Checkov creates a cloud resource connection graph for deep misconfiguration analysis across resource relationships.

References

Checkov https://www.checkov.io/1.Welcome/What%20is%20Checkov.html