🛡️ Cloud IaC Security Pipeline with Jira Integration + AWS Runtime Scanner

---

What This Project Does

- Scans Terraform for misconfigurations using Checkov
- Creates Jira tickets for failed checks (with MITRE ATT&CK + CIS Benchmark mapping)
- Scans live AWS EC2 Security Groups for risky ports open to the internet
- Generates Markdown reports for runtime misconfigurations

Tools Used

- **Checkov** – IaC misconfiguration scanner
- **Terraform** – Sample infrastructure for testing
- **Python** – Automation scripting
- **Jira API** – Ticket generation
- **Boto3** – AWS SDK for live Security Group inspection

AWS Environment Setup

If you want to run everything from within AWS instead of your local device:

1. Go to **EC2 > Launch Instance**
2. Name: `CloudSecurityScanner`
3. AMI: **Ubuntu Server 24.04 LTS**
4. Instance Type: `t2.micro` (Free Tier)

5. Create/select a key pair to SSH into the instance
6. Configure the security group:
    - Allow **SSH (port 22)** from `0.0.0.0/0` (testing only)
    - Optionally allow ports **80/443** to simulate risk
7. Launch the instance in **us-east-2**

8. SSH into the instance:

```
ssh -i your-key.pem ubuntu@<public-ip>
```

9. Install packages:

```
sudo apt update && sudo apt install -y python3-pip awscli
pip install checkov boto3 python-dotenv requests
```

10. Run `aws configure` to add your IAM user credentials (with EC2 ReadOnlyAccess)

> You can now clone your repo and run `checkov`, `jira_ticket_creator.py`, and `sg_report_generator.py` fully in AWS.

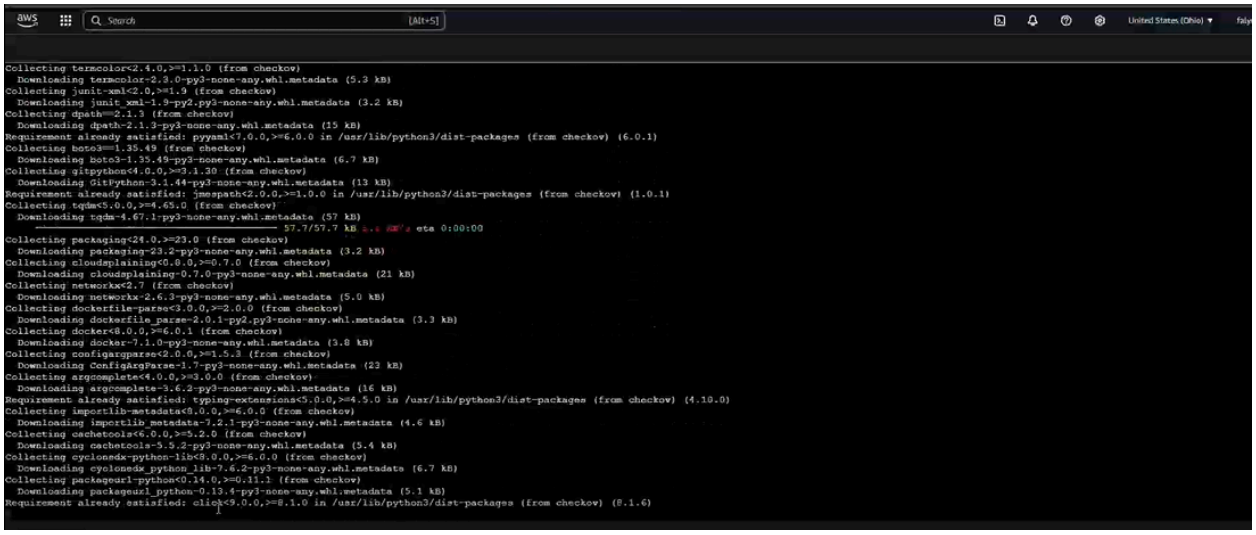
Setup Instructions

1. Clone This Repo

```
git clone https://github.com/yourusername/cloud-iac-jira-pipeline.git
cd cloud-iac-jira-pipeline
```

2. Install Python Dependencies

```
pip install checkov python-dotenv requests boto3 --break-system-packages
```



3. Configure Jira API Secrets

Create a file called `.jira_secrets.env` using the example:

```
# .jira_secrets.env
JIRA_URL=https://yourworkspace.atlassian.net
JIRA_EMAIL=you@example.com
JIRA_API_TOKEN=your-api-token
JIRA_PROJECT_KEY=AWS
```

Important: Add `.jira_secrets.env` to your `.gitignore`

🧩 Jira Setup Note

To use Jira automation, you must:

- Have an active Jira Cloud workspace (sign up at https://www.atlassian.com/software/jira)
- Create an API token under your Atlassian account settings
- Create a Jira project and copy the project key (e.g., AWS)

Terraform IaC + Checkov Scan

main.tf (Sample Insecure Config)

```
resource "aws_security_group" "insecure_sg" {
  name = "insecure_sg"

  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  tags = {
    Name = "insecure_sg"
  }
}
```

Run Checkov

```
checkov -d . -o json > checkov_results.json
```

Auto-Create Jira Tickets

jira_ticket_creator.py

```
import os, json, requests
from dotenv import load_dotenv

load_dotenv(".jira_secrets.env")
JIRA_URL = os.getenv("JIRA_URL")
JIRA_EMAIL = os.getenv("JIRA_EMAIL")
JIRA_API_TOKEN = os.getenv("JIRA_API_TOKEN")
JIRA_PROJECT_KEY = os.getenv("JIRA_PROJECT_KEY")

headers = {"Content-Type": "application/json"}
auth = (JIRA_EMAIL, JIRA_API_TOKEN)

with open("checkov_results.json", "r") as f:
```

```
    findings = json.load(f)

for result in findings:
    title = f"{result['check_id']} - {result['check_name']}"
    desc = f"""
Resource: {result['resource']}
File: {result['file_path']}
Lines: {result['file_line_range']}
Guideline: {result.get('guideline', 'N/A')}
"""

    payload = {
        "fields": {
            "project": {"key": JIRA_PROJECT_KEY},
            "summary": title,
            "description": {
                "type": "doc",
                "version": 1,
                "content": [{
                    "type": "paragraph",
                    "content": [{"type": "text", "text": desc}]
                }]
            },
            "issuetype": {"name": "Task"}
        }
    }

    response = requests.post(f"{JIRA_URL}/rest/api/3/issue", headers=headers, json=payload,
auth=auth)
    print("[+] Created Jira ticket:" if response.status_code == 201 else "[!] Failed:", title)
```

▶️ Run It

```
python3 jira_ticket_creator.py
```

Part 3: Map to MITRE ATT&CK + CIS

Checkov_mitre_cis_mapping.csv

Checkov ID	Misconfiguration	MITRE ID	Tactic	Technique	CIS ID	Description
CKV_AWS_24	SG allows 0.0.0.0/0 to port 22	T1021.004	Lateral Movement	Remote Services: SSH	4.1	Block unrestricted SSH access
CKV_AWS_23	SG rule has no description	T1609	Defense Evasion	Container Admin Command	5.1	Add rule descriptions for auditability

---

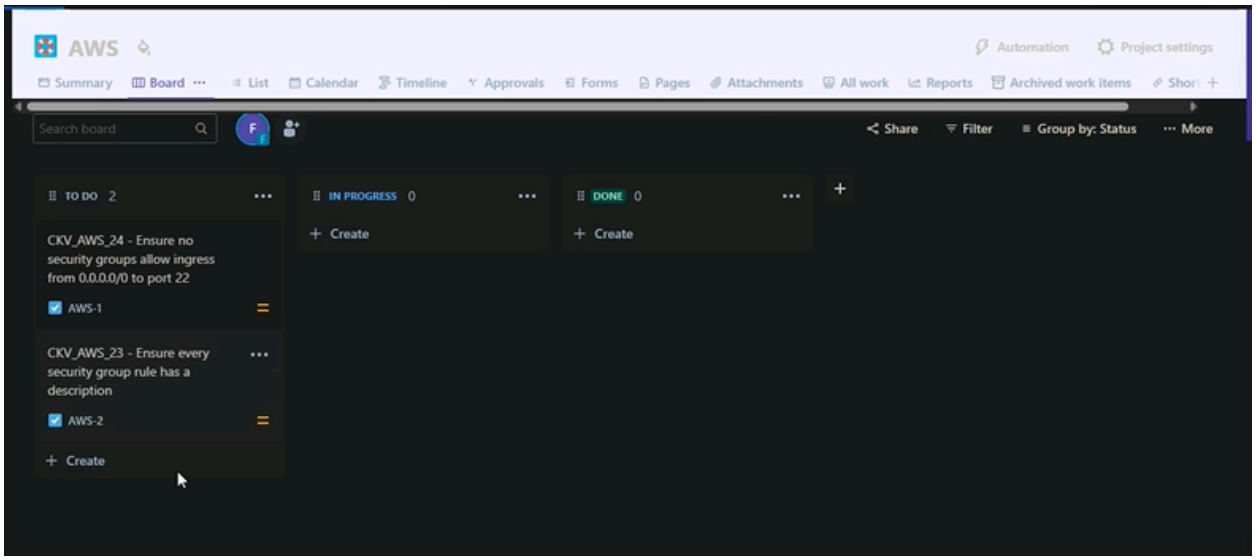
Live AWS Misconfig Scanner

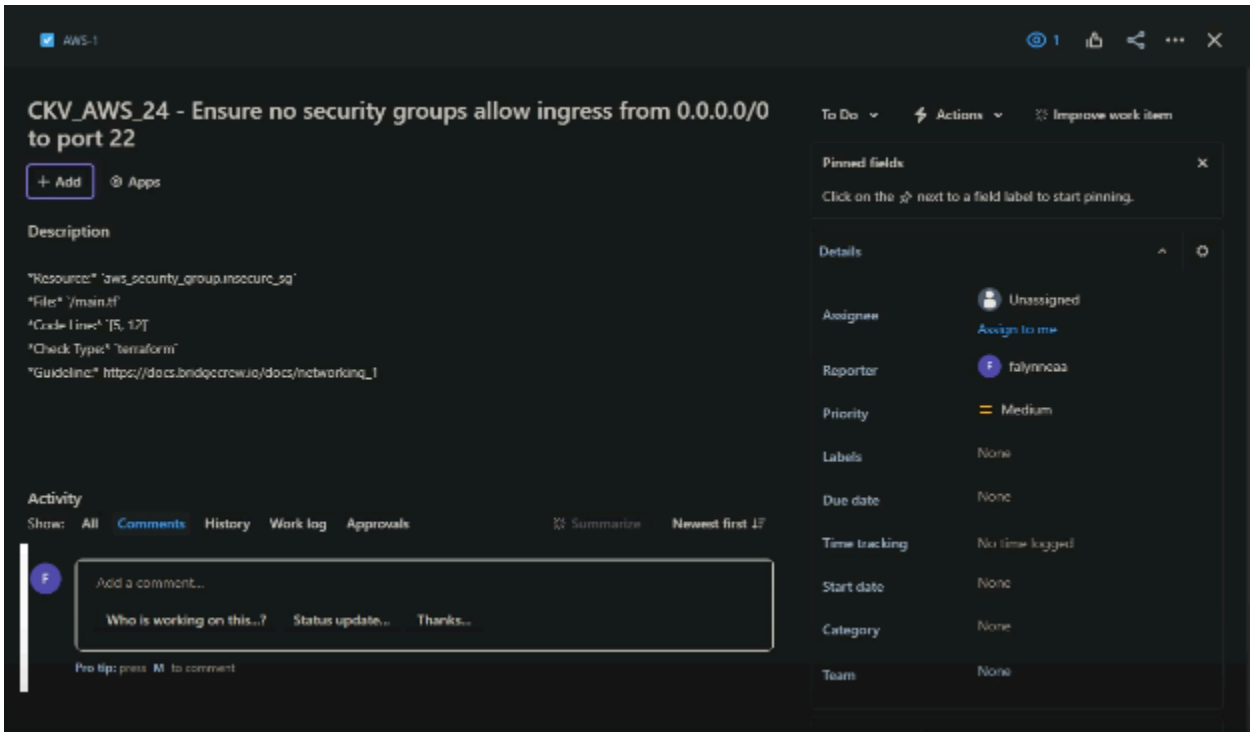
sg_report_generator.py

```
import boto3
from datetime import datetime

def scan_and_generate_report():
    ec2 = boto3.client('ec2', region_name='us-east-2')
    response = ec2.describe_security_groups()
    report_name =
f"sg_misconfig_report_{datetime.now().strftime('%Y-%m-%d_%H-%M-%S')}.md"

    with open(report_name, "w") as report:
        report.write("# Security Group Misconfiguration Report\n")
        for sg in response['SecurityGroups']:
            report.write(f"## {sg['GroupName']} ({sg['GroupId']})\n")
            for perm in sg.get('IpPermissions', []):
                for ip in perm.get('IpRanges', []):
                    if ip.get('CidrIp') == "0.0.0.0/0":
                        from_p = perm.get('FromPort', 'ALL')
                        to_p = perm.get('ToPort', 'ALL')
                        report.write(f"- Port {from_p}-{to_p} open to 0.0.0.0/0\n")
    print(f"[+] Report saved as {report_name}")

scan_and_generate_report()
```

▶️ Run It

python3 sg_report_generator.py





Remediation

CKV_AWS_24 — SSH

- Fix: Restrict CIDR range (e.g. 10.0.0.0/16)
- CIS 4.1
- MITRE T1021.004 – Remote Services: SSH

CKV_AWS_23 — Missing SG rule

- Fix: Add description = "Allow SSH from VPN"
- CIS 5.1
- MITRE T1609 – Container Administration Command

You have now built a full-stack cloud security pipeline that scans IaC and live AWS configurations, maps vulnerabilities to MITRE & CIS, and automates Jira ticket creation; all from a cloud-hosted EC2 instance. Yayyyyy!!