S3 Setup for Security Testing or Storage

S3 Setup

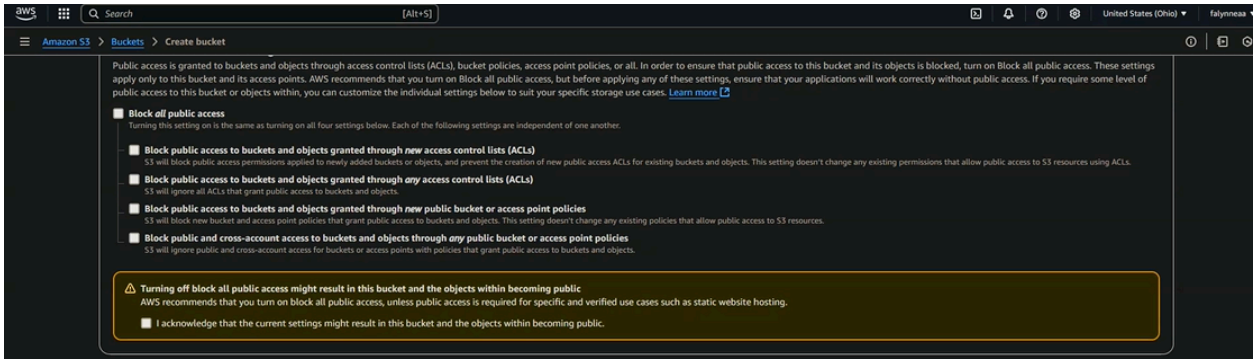
1. **Go to S3 Console:**
   https://s3.console.aws.amazon.com

2. **Click “Create Bucket”**

   - **Bucket name: `checkov-iac-test-bucket` (must be globally unique)**
   - **Region: `us-east-2`**
   - **Uncheck “Block all public access” (for testing only)**
   - **Enable versioning (optional but recommended)**
   - **Create the bucket**

3. **Upload a test file (like `main.tf` or `checkov_results.json`) to simulate access:**

   aws s3 cp main.tf s3://checkov-iac-test-bucket/main.t



Bucket Policy (Simulated Misconfiguration)

This policy allows public read access to the bucket ; a known misconfig in the real world.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadTest",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::checkov-iac-test-bucket/*"
    }
  ]
}
```

> ⚠️ Only use this in lab environments. Public S3 access is a top security finding in real cloud audits.

Use Cases for S3 in This Project

- Store `checkov_results.json` and load it via script
- Run Checkov scans directly from downloaded IaC via:
  - aws s3 cp s3://bucket-name/main.tf .
  - checkov -f main.tf
- Test security misconfig detection

AWS EC2 Instance Setup for CLI-Based Cloud Security Scanning

Set up a cloud-based **Ubuntu VM in AWS** to:

- Run Python scripts (`jira_ticket_creator.py`, `sg_report_generator.py`)
- Use AWS CLI & `boto3` with EC2-level permissions
- Keep everything self-contained inside AWS

Steps:

1. **Launch an EC2 VM**
    - **Service**: EC2 → Launch Instance
    - **Name**: `CloudSecurityScanner`
    - **AMI**: Ubuntu Server 24.04 LTS (HVM), SSD Volume Type
    - **Instance Type**: `t2.micro` (Free Tier eligible)
    - **Key Pair**: Create or select one (used to SSH into the VM)
    - **Region**: `us-east-2`
2. **Security Group Configuration**

- Created a new security group:
  - ✅ Allow SSH (port 22) from your IP or `0.0.0.0/0` for testing
  - Optionally allowed ports like 80/443 if testing other open ports
- This was also used to simulate misconfigurations for `sg_report_generator.py`

3. **Connect to the Instance**
   - Via SSH in terminal or EC2 Connect:
     - ssh -i your-key.pem ubuntu@<public-ip>

4. **Update the System + Install Dependencies**
- sudo apt update && sudo apt upgrade -y
- sudo apt install python3-pip -y
- pip install checkov python-dotenv requests boto3

5. **Install & Configure AWS CLI**
   - sudo apt install awscli -y
   - aws configure
6.
   - Entered IAM user's access key, secret, and set region to `us-east-2`

7. **Cloned GitHub Project & Ran Scripts from VM**
   - Used `git clone` or copy-pasted scripts into the EC2 VM

Ran:

- checkov -d . -o json > checkov_results.json
- python3 jira_ticket_creator.py
- python3 sg_report_generator.py

IAM Permissions Required:

- `AmazonEC2ReadOnlyAccess` (for `describe-security-groups`)
- Optional: `CloudWatchLogsReadOnlyAccess` if logging is added later