Automated Jenkins Backup & Restore with S3 (Least IAM Privilege)

Mohammed Amir

Project Overview

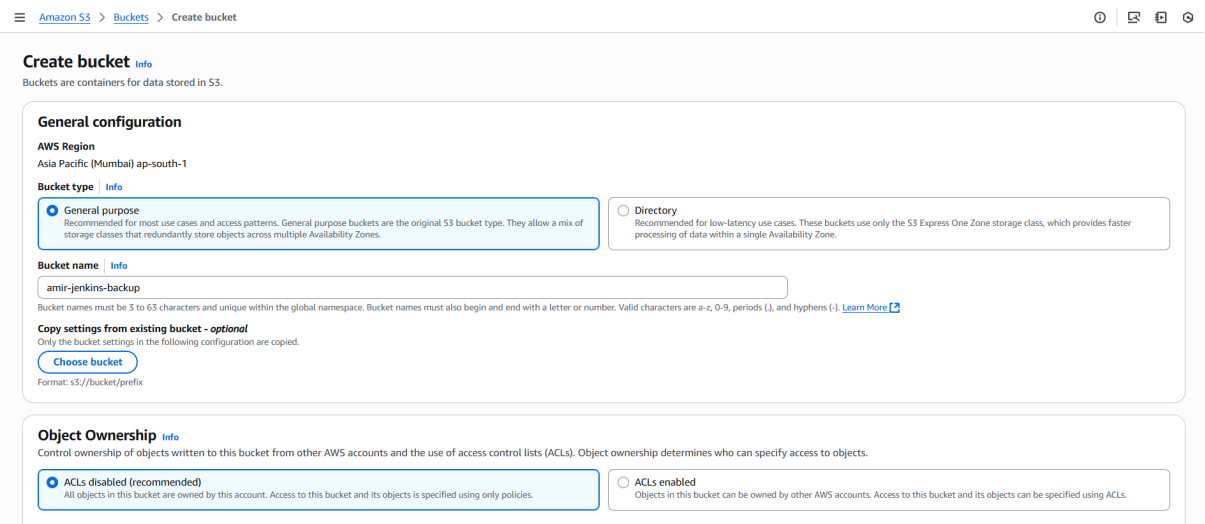
This document outlines the process of **backing up the Jenkins from localhost (WSL) to an S3 bucket** and **restoring it on a EC2 instance** using **IAM policies with least privilege access** both manually and automatically (cronjobs).

Key Skills: AWS (S3, IAM), Jenkins, Linux, Cron, Shell Scripting

Backup and Upload Procedure

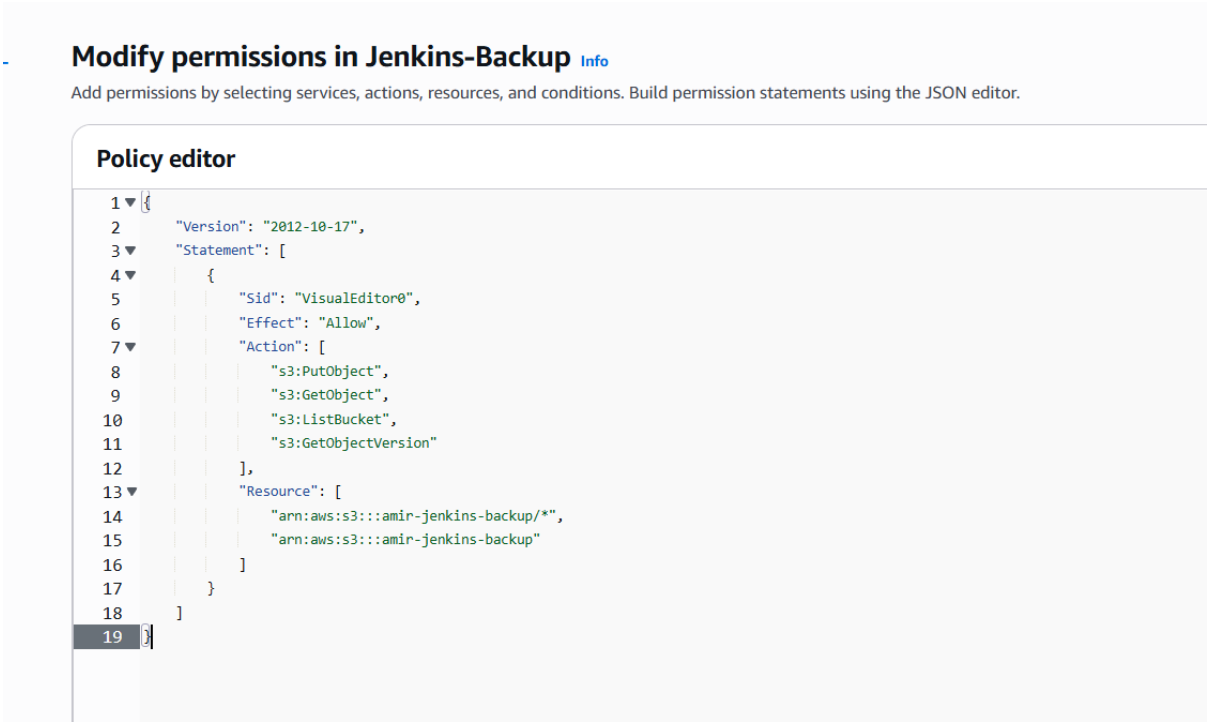
Step 1: Create an S3 Bucket

An S3 bucket was created in the AWS account to store Jenkins backup files.



Step 2: Create IAM Policy

An IAM policy was defined with minimal permissions `s3:PutObject`, `s3:ListBucket` for the specific bucket `amir-jenkins-backup`. This policy was attached to an IAM Role (Jenkins-Backup-S3) and user (amir).



Step 3: Install AWS CLI

AWS CLI was installed on the local machine running Linux via WSL to enable interaction with AWS.

```
~sudo apt update
~sudo apt install awscli -y
```

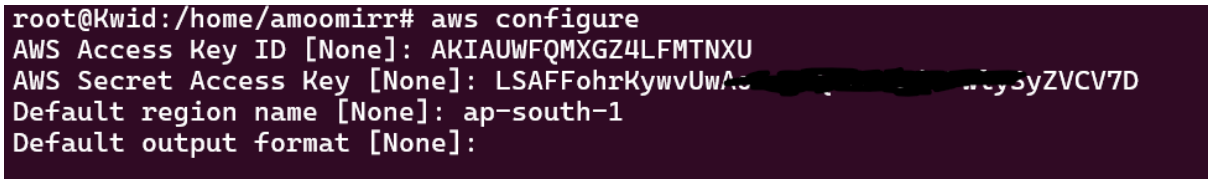
Step 4: Configure AWS CLI

AWS CLI was configured using credentials from the IAM user (amir).

```
~aws configure
```

Inputs required:

- AWS Access Key ID
- AWS Secret Access Key
- Default Region
- Output format (e.g., json)



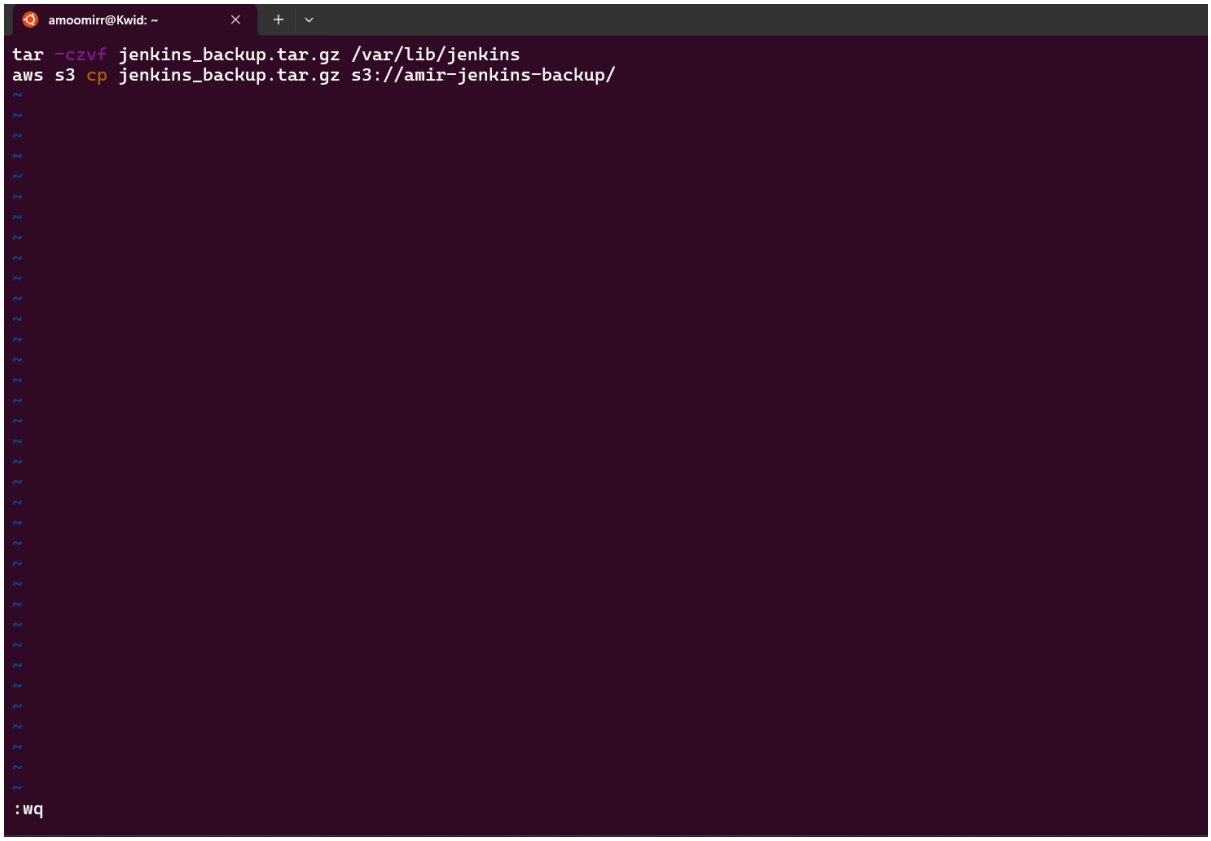
Step 5: Create a Backup and Upload Script

A shell script was written to compress the Jenkins home directory and upload the archive to the S3 bucket.

```
#!/bin/bash
~tar -czvf jenkins_backup.tar.gz /var/lib/jenkins
~aws s3 cp jenkins_backup.tar.gz s3://your-bucket-name/
```

Explanation of `tar -czvf`:

- `-c`: Create a new archive
- `-z`: Compress using gzip
- `-v`: Verbose output
- `-f`: Specify archive file name



A separate upload script was also created, and a cron job was configured to automate the process.

```
amoomirr@Kwid: ~        root@ip-172-31-11-232: /var/lit
  GNU nano 7.2                                   upload.sh *
#!/bin/bash

#Stop Jenkins
systemctl stop jenkins

# Set timestamp format
TIMESTAMP=$(date +%Y%m%d)

# Define backup file name with timestamp
BACKUP_FILE="/home/amoomirr/jenkins_backup_$TIMESTAMP.tar.gz"

# Backup Jenkins home directory with timestamped filename
tar -czvf "$BACKUP_FILE" /var/lib/jenkins

# Define S3 destination with same timestamped name
DESTINATION="s3://amir-jenkins-backup/jenkins_backup_$TIMESTAMP.tar.gz"

# Upload the backup to AWS S3
aws s3 cp "$BACKUP_FILE" "$DESTINATION" --region ap-south-1

#Start Jenkins
systemctl start jenkins

^G Help      ^O Write Out  ^W Where Is   ^K Cut       ^T Execute    ^C Location   M-U Undo
^X Exit      ^R Read File  ^\ Replace    ^U Paste     ^J Justify    ^/ Go To Line M-E Redo
```

The scripts were made executable:

```
~chmod +x backup_script.sh
~chmod +x upload.sh
```

Step 6: Run the Script

The script was tested manually using:

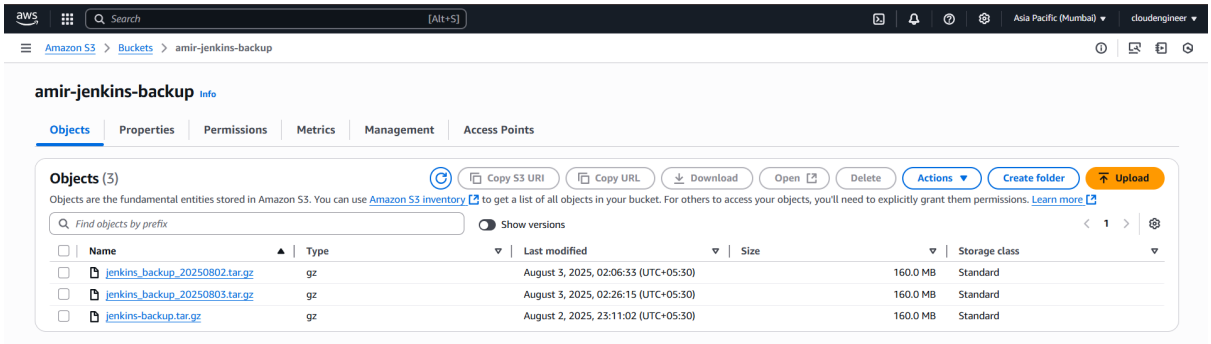
```
~sudo sh backup.sh
```

To automate, a cron job was scheduled (2:25am every Sunday):

```
~sudo crontab -e    # Edit cron table
~sudo crontab -l    # List current cron jobs
```

```
amoomirr@Kwid: ~        root@ip-172-31-11-232: /var/lit
  GNU nano 7.2                                   /tmp/crontab.hDN55D/crontab
25 2 * * 0 /home/amoomirr/upload.sh 2>> /home/amoomirr/upload.log
```

Backups were verified in the S3 bucket. Logs were reviewed in case of errors.

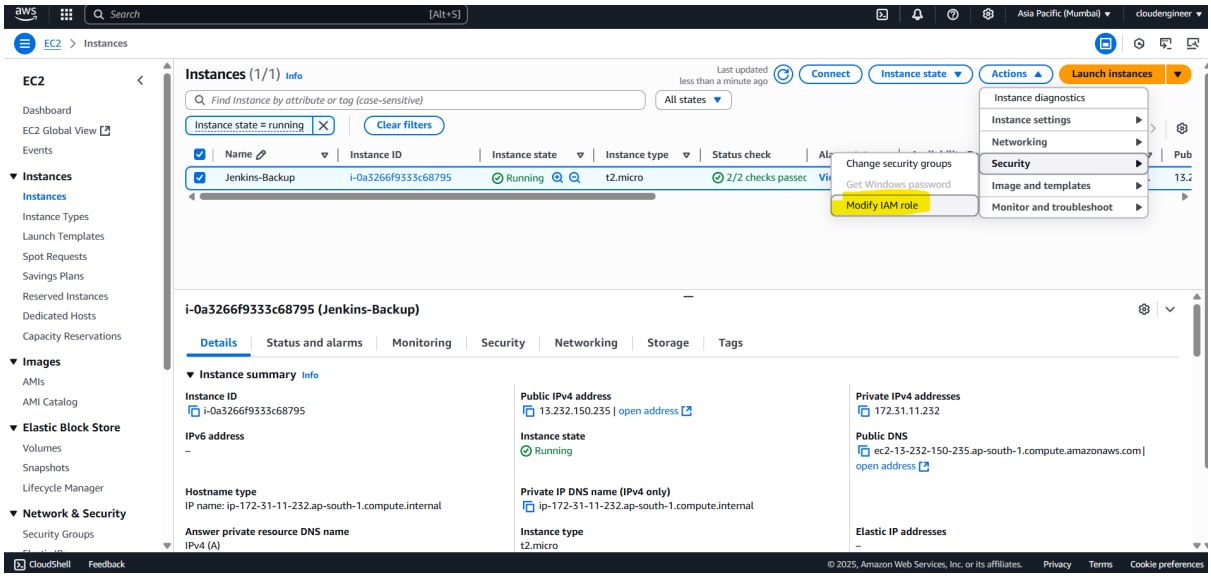


Restore Procedure

Step 1: Launch EC2 and SSH

A new EC2 instance was launched and the IAM role created earlier was attached to it.

```
#Attach the IAM role to EC2 instance.
```



SSH access to the EC2 instance was established using the appropriate private key.

```
amoomirr@Kwid:~$ ls -l | grep "Jenkins"
-r-------- 1 amoomirr amoomirr      1674 Jul 31 00:40 Jenkins-Server-Keypair.pem
amoomirr@Kwid:~$ ssh -i "Jenkins-Server-Keypair.pem" ubuntu@ec2-13-232-150-235.ap-south-1.compute.amazonaws.com
The authenticity of host 'ec2-13-232-150-235.ap-south-1.compute.amazonaws.com (13.232.150.235)' can't be established.
ED25519 key fingerprint is SHA256:8o5VXAXYurmnFe/hPxmgAV0aLeNzG9YIqkM/+Ax7AUI.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? Yes
Warning: Permanently added 'ec2-13-232-150-235.ap-south-1.compute.amazonaws.com' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-1029-aws x86_64)
```

Step 2: Download the Backup File

The backup file was downloaded from the S3 bucket using AWS CLI.

```
#aws s3 cp s3://your-bucket-name/jenkins_backup_20250803.tar.gz .
```

```
root@ip-172-31-11-232:/home/ubuntu# aws s3 cp s3://amir-jenkins-backup/jenkins_backup_20250803.tar.gz jenkins_
backup_20250803.tar.gz
download: s3://amir-jenkins-backup/jenkins_backup_20250803.tar.gz to ./jenkins_backup_20250803.tar.gz
root@ip-172-31-11-232:/home/ubuntu#
```

Step 3: Extract the Backup

The Jenkins service was stopped, and the backup archive was extracted into the Jenkins home directory and the service was restarted.

```
~sudo systemctl stop jenkins
~sudo tar -xzvf jenkins_backup20250803.tar.gz -C /
~sudo systemctl start jenkins
```

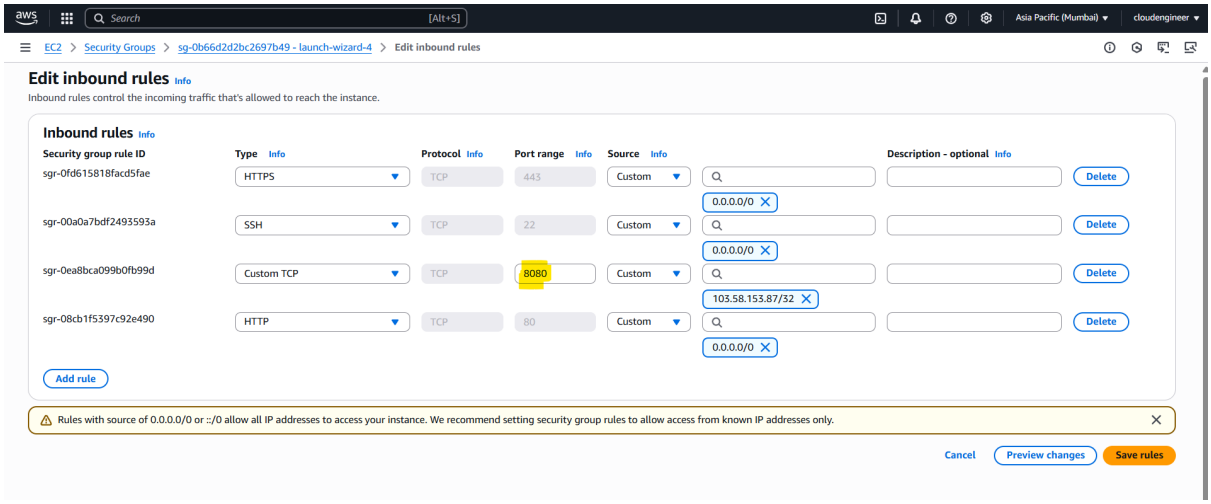
Explanation of `tar -xzvf -C /`:

- `-x`: Extract files from archive
- `-z`: Decompress using gzip
- `-v`: Verbose output
- `-f`: Specify archive file
- `-C /`: Extract to root directory

```
root@ip-172-31-11-232:/home/ubuntu# cd /var/lib/jenkins/workspace
root@ip-172-31-11-232:/var/lib/jenkins/workspace# ls
Nginx-App  Nginx-App@tmp  flask-app  flask-app@tmp  results.json
```

Step 4: Allow Inbound Port 8080 on EC2

Open port 8080 in the EC2 security group to access Jenkins via web browser (HostIP:8080)



Conclusion

- **Achieved Fully Automated Backups:** Implemented an end-to-end automated backup and restore solution for Jenkins using scheduled cron jobs and shell scripts.
- **Utilized AWS S3 for Reliable Storage:** Backups are securely uploaded to Amazon S3, ensuring availability and durability.
- **Enforced Least Privilege Security:** Configured an IAM user with only necessary permissions (put, get, delete on specific bucket path), following best security practices.
- **Shell Scripting & Cron Integration:** Leveraged Bash scripting and cron scheduling to automate backup and upload tasks without manual intervention.
- **Disaster Recovery Ready:** Restore script pulls backup from S3 and safely extracts it to the Jenkins directory, enabling rapid recovery in case of failure.
- **Error Logging Implemented:** Each step in the backup and upload process is logged, aiding in debugging and monitoring the system.