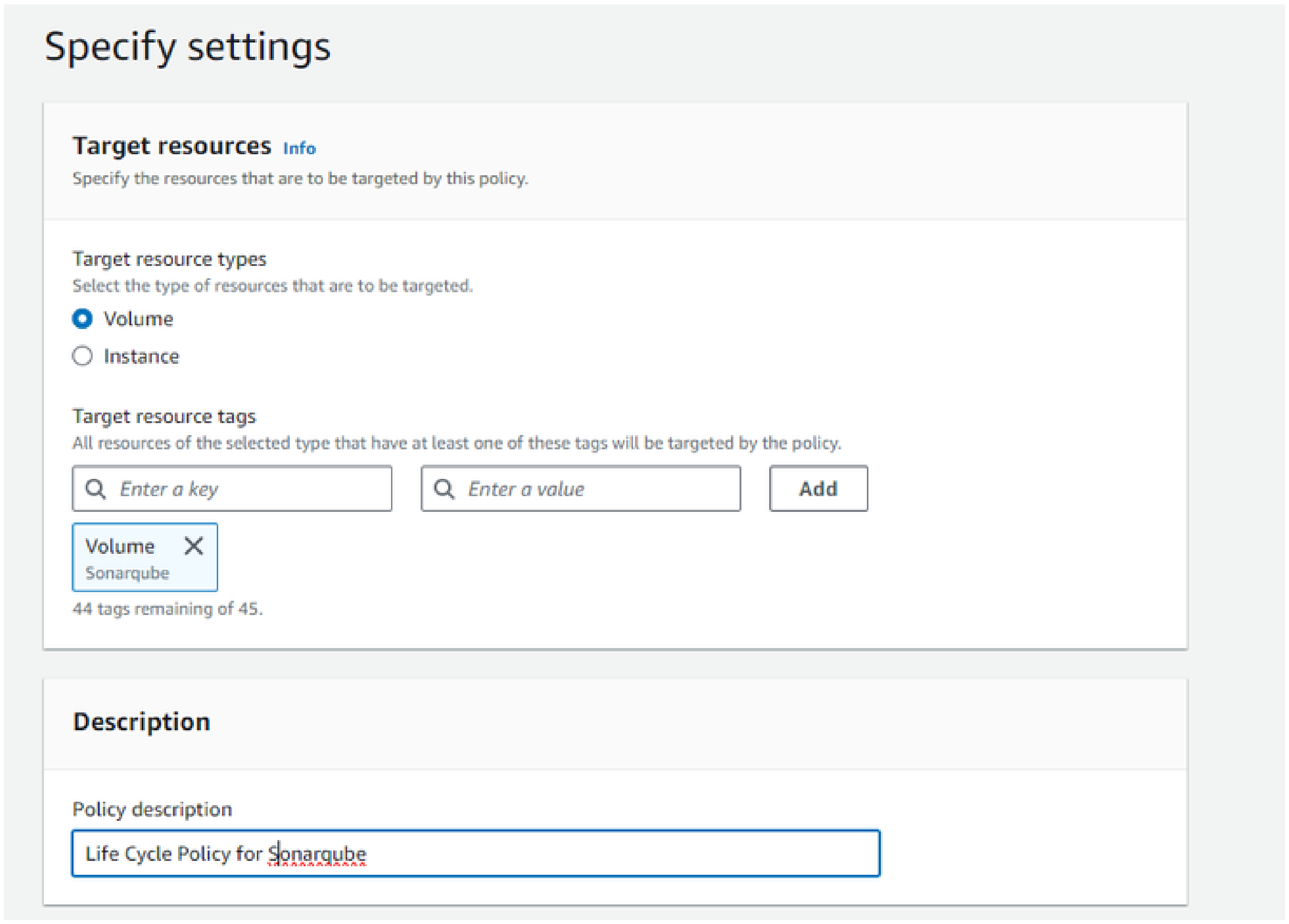
Backup Policy for Devops Tools

This is the document for the volume backups

- Create volume for each services like sonarqube, jenkins, gitlab and nexus via docker-compose files
- Create tags for each volumes
- Create lifecycle policy for each volumes to get the backups

  Below is an example for how to create the lifecycle policy

  EC2 > Elastic Block Store > Lifecycle Manager



Specify settings

Target resources Info

Specify the resources that are to be targeted by this policy.

Target resource types

Select the type of resources that are to be targeted.

[✓ Volume] Instance

Target resource tags

All resources of the selected type that have at least one of these tags will be targeted by the policy.

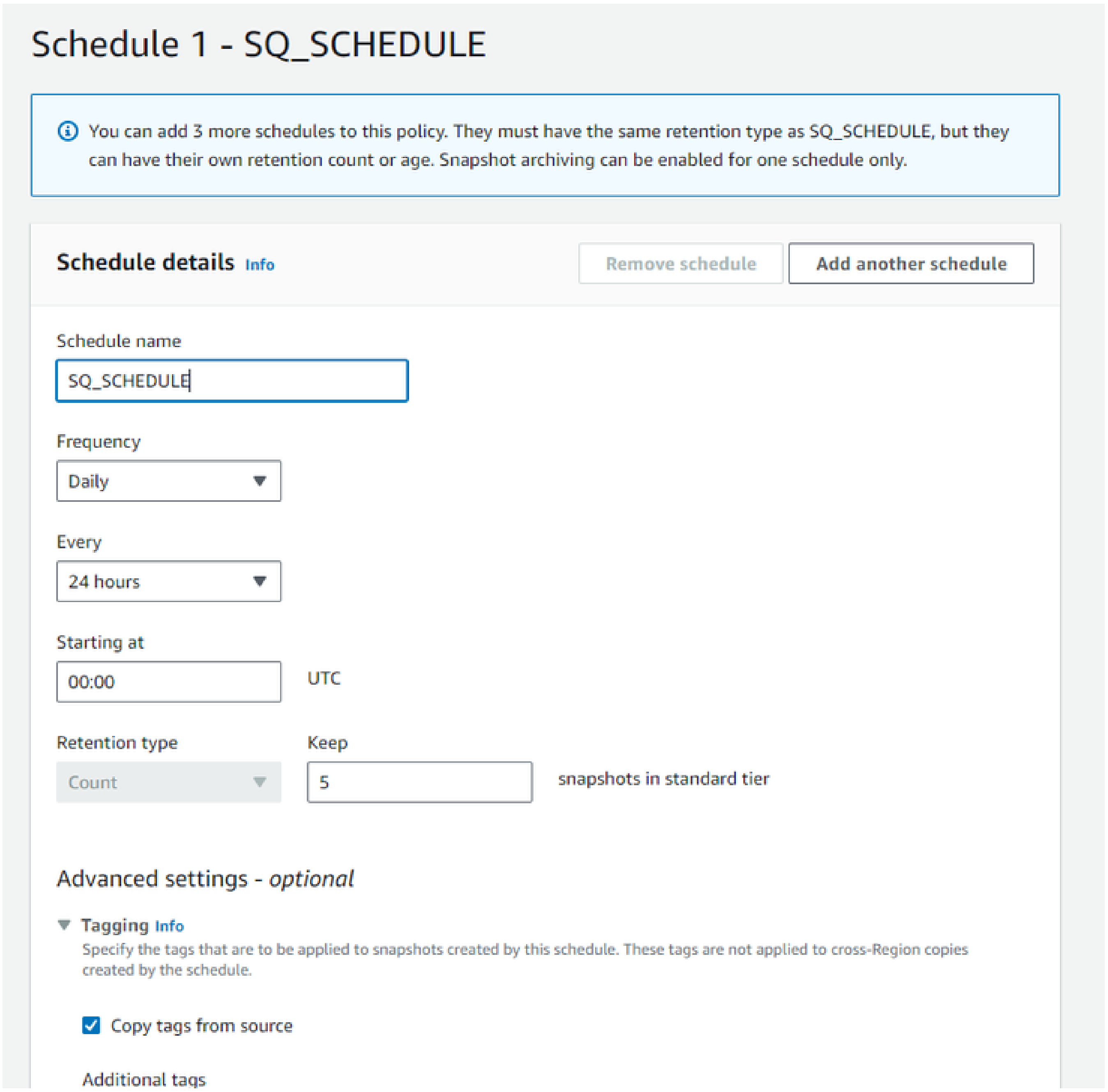
Volume: Sonarqube

44 tags remaining of 45.

Description

Policy description: Life Cycle Policy for Sonarqube

In the above screenshot the under the Target Resource Tags mention the tag that you have created for the volume that you want to create and mention the Policy description keep everything as default option



Schedule 1 - SQ_SCHEDULE

You can add 3 more schedules to this policy. They must have the same retention type as SQ_SCHEDULE, but they can have their own retention count or age. Snapshot archiving can be enabled for one schedule only.

Schedule details Info

Remove schedule | Add another schedule

Schedule name: SQ_SCHEDULE

Frequency: Daily

Every: 24 hours

Starting at: 00:00 UTC

Retention type: Count

Keep: 5 snapshots in standard tier

Advanced settings - *optional*

Tagging Info

Specify the tags that are to be applied to snapshots created by this schedule. These tags are not applied to cross-Region copies created by the schedule.

☑ Copy tags from source

Additional tags

Frequency: The snapshot is scheduled to be taken "Daily."|
Every: It is set to occur every "24 hours," which aligns with the daily frequency.
Starting at: The snapshots will be initiated at "00:00" UTC, meaning right at midnight UTC.
Retention Type: The selected retention type is "Keep."
Count: The system is set to retain "5" snapshots. This means that at any given time, the five most recent daily snapshots will be kept, and older snapshots beyond this count will be deleted as new ones are created.

Limitations:

Ref : How To Automate AWS EBS Snapshot Using AWS Snapshot Lifecycle Policy | by Bhuvanesh | Searce

You have to wait for an hour after created the policy.
If anyone of your tags are already a part of any lifecycle policy, then you can’t use the same tag for another policy. Else it’ll show the below error.

```
The following target tag(s) are already in use: {[Key=Env,Value=Dev]}
```

If you removed your policy or changed the tags from the policy, the snapshots which all are taken previously from the snapshot will not be deleted. Still you can use them.

If you copy the snapshot which is taken by the policy, it won't be coming under automated deletion. You have to remove them manually.

If you deleted the EBS volumes, the snapshots which are then from policy will not be deleted until the retention policy.

You can create snapshot at every 12hr or every 24hr only.

We can find the snapshots of Lifecycle Manager from Snapshots

How To Restore :

You can create a volume from snapshot. Please go through the below AWS documentation for creating volume from snapshot

Attach an Amazon EBS volume to an instance - Amazon Elastic Compute Cloud
Make an Amazon EBS volume available for use on Linux - Amazon Elastic Compute Cloud

Navigate to the EC2 Dashboard: Open your AWS Management Console and go to the EC2 dashboard.

1. **Access 'Volumes'**: In the "Elastic Block Store" section, click on "Volumes".
2. **Attach the Volume**: Choose the EBS volume you want to attach, click on "Actions", and select "Attach Volume".
3. **Select the Instance**: Choose the EC2 instance to attach the volume to and provide a device name (e.g., `/dev/sdf`). Then, click "Attach".

Step 2: Log Into Your EC2 Instance

- Use SSH to connect to your EC2 instance. The command generally looks like this:

  ```
  ssh -i /path/to/your-key.pem ec2-user@instance-public-dns
  ```

Step 3: Identify the Attached Volume

- Once connected, use the `lsblk` command to list all disk devices. The new volume will be listed without any mount points.

  ```
  lsblk
  ```

Step 4: Format and Mount the Volume

1. **Format the Volume**: If it's a new volume, format it using:

   ```
   sudo mkfs -t ext4 /dev/nvme1n1
   ```

   Replace `/dev/nvme1n1` with your volume's device name.
2. **Create a Mount Point**:

   ```
   sudo mkdir /custom-data
   ```
3. **Mount the Volume**:

   ```
   sudo mount /dev/nvme1n1 /custom-data
   ```

Step 5: Automate Mounting on Reboot

1. **Find the UUID of the Volume**:

   ```
   sudo blkid
   ```

   Locate the UUID for `/dev/nvme1n1`.
2. **Edit** `/etc/fstab`: Open the file in a text editor and add the following line (replace `your-uuid` with the actual UUID):

   ```
   UUID=your-uuid /custom-data ext4 defaults,nofail 0 2
   ```
3. **Save and Close**: Save the changes and exit the editor.

Step 6: Verify the Configuration

- Check if the volume is correctly mounted:

  ```
  df -h
  ```
- Optionally, reboot the instance to test the auto-mount feature:

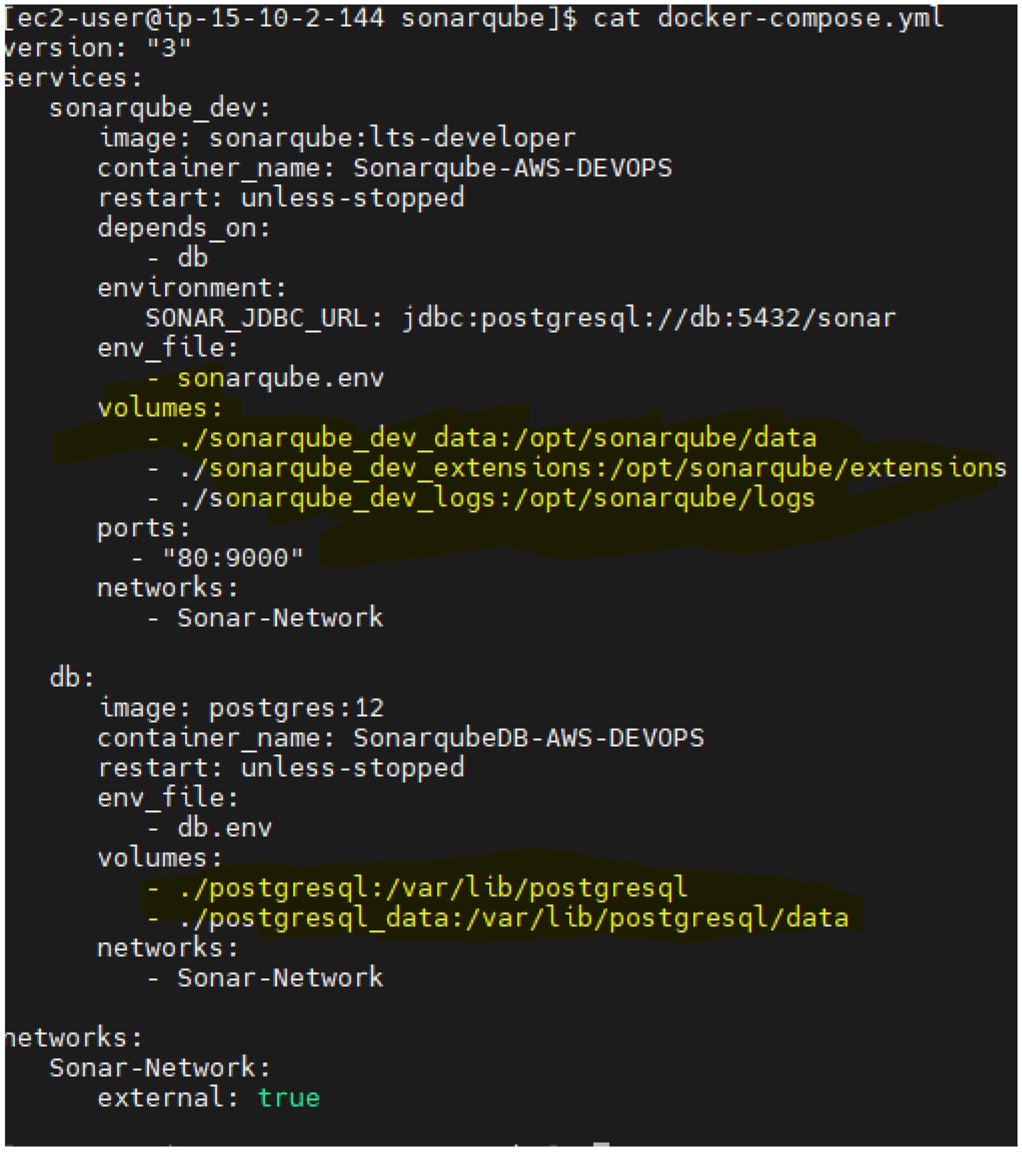
  ```
  sudo reboot
  ```

Important Notes

- Ensure the correctness of the device name and UUID when editing the `/etc/fstab` file. Incorrect entries can make your system unbootable.
- The `nofail` mount option is crucial for ensuring that boot process continues even if the mount fails.

Once you attached the volume, go to the directory where we mentioned the volumes for the docker -compose and make sure every volumes attached in the compose file is there in that directory

Below is a screenshot mentioning the volumes in the docker-compose files



```
[ec2-user@ip-15-10-2-144 sonarqube]$ cat docker-compose.yml
version: "3"
services:
   sonarqube_dev:
      image: sonarqube:lts-developer
      container_name: Sonarqube-AWS-DEVOPS
      restart: unless-stopped
      depends_on:
         - db
      environment:
         SONAR_JDBC_URL: jdbc:postgresql://db:5432/sonar
      env_file:
         - sonarqube.env
      volumes:
         - ./sonarqube_dev_data:/opt/sonarqube/data
         - ./sonarqube_dev_extensions:/opt/sonarqube/extensions
         - ./sonarqube_dev_logs:/opt/sonarqube/logs
      ports:
        - "80:9000"
      networks:
         - Sonar-Network

   db:
      image: postgres:12
      container_name: SonarqubeDB-AWS-DEVOPS
      restart: unless-stopped
      env_file:
         - db.env
      volumes:
         - ./postgresql:/var/lib/postgresql
         - ./postgresql_data:/var/lib/postgresql/data
      networks:
         - Sonar-Network

networks:
   Sonar-Network:
      external: true
```