Automatic AWS EC2 Backups

If you have a lot of developers working on the same server, there is nothing worse than having to fix something that went horribly wrong with it. That is why I wrote a script (see at bottom of this page) to help other developers to back up their AWS EC2 instances daily and set the number of versions to keep. If a developer screws up the server, that is ok. You can just restore a copy from last night.

First thing you will need to do is create an AWS IAM user to allow you to specify a backup policy. This user will be restricted to very limited abilities. Once the user has been created apply a policy that just allows backups. I suggest AWSBackupFullAccess . Please avoid using full access policies. They can allow someone to do crazy dangerous things (like spinning up multizone servers $$$ outch).

Once you have created a user with the required backup policy, create an Access Key. You will use the generated Access Key and Secret in the script below.

Now you can SSH into your EC2 instance (Ubuntu in my case) and install the AWS cli tool.

```
sudo apt-get -y install awscli; aws configure;
```

Fill in the appropriate values for the configuration prompts. Remember to use the Key and Secret you just created. You can see an example of what values the config tool expects in the script code below. Make sure you know the region as the backup will only work if the region matched the EC2 instances you are backing up.

Next you need to get the id of the EC2 insance or instances you want to backup. In the examples script below it is only backing up one server but you can do many. Example below.

```
instances+=("autobackup_developmemt|i-0ed78a1f3583e1543")
instances+=("autobackup_staging|i-0ed72a1f3583e343")
```

Once that is done you are ready to add the script to your server. It will run off a cron. Make sure you put the file someplace this is not accessible to the public obviously (e.g. not in a public website folder).

Make the script executable using the chmod +x command. Then give it a test.

Once you know it runs as you can see the AMI (EC2 backup image) created or being created, you can add a cron to automate the backups. Use this command to create or edit the crontab. Note that for ubuntu the typical user is “ubuntu” but for AWS Linux it will be “ec2-user”.

```
sudo crontab -e -u ubuntu
```

Add the following line (adjust the path to your script)

```
# backup EC2 instances nightly 4:30 am GMT
30 4 * * * . $HOME/.profile;  /var/devops/ec2_backup.sh
```

Now are are done. You can sleep at night knowing that no matter how much someone screws up the server, they won’t screw up your day.

Here the full script:

```
#!/bin/bash

# prior to using this script you will need to install the aws cli on the local machine

# https://docs.aws.amazon.com/AmazonS3/latest/dev/setup-aws-cli.html

# AWS CLI - will need to configure this
# sudo apt-get -y install awscli
# example of current config - july 10, 2020
#aws configure
#aws configure set key ARIAW5YUMJT7PO2N7L *fake - user your own*
#aws configure secret X2If+xa/rFITQVMrgdQVpFLx1c7fwP604QkH/x *fake - user your own*
#aws configure set region us-east-2
#aws configure set format json



# backup EC2 instances nightly 4:30 am GMT
# 30 4 * * * . $HOME/.profile;  /var/www/devopstools/shell-
scripts/file_backup_scripts/ec2_backup.sh
```

```
script_dir="$(dirname "$0")"

# If you want live notifications about backups, use this example with a correct slack key
#SLACK_API_URL="https://hooks.slack.com/services/T6VQ93KM/BT8REK5/hFYEDUCoO1Bw72wxxFSj7oY"


prevday1=$(date --date="2 days ago" +%Y-%m-%d)
prevday2=$(date --date="3 days ago" +%Y-%m-%d)
today=$(date +"%Y-%m-%d")

instances=()
# add as many instances to backup as needed
instances+=("autobackup_impressto|i-0ed78a1f3583e1543")


for ((i = 0; i < ${#instances[@]}; i++)); do

    instance=${instances[$i]}

    instanceName="$(cut -d'|' -f1 <<<"$instance")"
    instanceId="$(cut -d'|' -f2 <<<"$instance")"

    prevImageName1="${instanceName}_${prevday1}_$instanceId"
    prevImageName2="${instanceName}_${prevday2}_$instanceId"
    newImageName="${instanceName}_${today}_$instanceId"

    consoleout --green "Begin backing $instanceName [$instanceId]"

    aws ec2 create-image \
        --instance-id $instanceId \
        --name "$newImageName" \
        --description "$instanceName" \
        --no-reboot

    if [ $? -eq 0 ]; then
        echo "$newImageName created."
        echo ""
        if [ ! -z "${SLACK_API_URL}" ]; then
            curl -X POST -H 'Content-type: application/json' --data '{"text":":rotating_light:
Backing up *'$newImageName'* to AMI. :rotating_light:"}' ${SLACK_API_URL}        fi
```

```
        echo -e "\e[92mBacking up ${newImageName} to AMI."
    else
        echo "$newImageName not created."
        echo ""
    fi

    imageId=$(aws ec2 describe-images --filters "Name=name,Values=${prevImageName1}" --query
'Images[*].[ImageId]' --output text)

    if [ ! -z "${imageId}" ]; then

        echo "Deregistering ${prevImageName1} [${imageId}]"
        echo ""
        echo "aws ec2 deregister-image --image-id ${imageId}"
        aws ec2 deregister-image --image-id ${imageId}
    fi

    imageId=$(aws ec2 describe-images --filters "Name=name,Values=${prevImageName2}" --query
'Images[*].[ImageId]' --output text)

    if [ ! -z "${imageId}" ]; then

        echo "Deregistering ${prevImageName2} [${imageId}]"
        echo ""
        echo "aws ec2 deregister-image --image-id ${imageId}"
        aws ec2 deregister-image --image-id ${imagesId}
    fi

    consoleout --green "Completed backing $instanceName"

done
```

Revision #1
Created 2024-06-14 01:00:08 UTC by peterd
Updated 2024-06-14 01:00:33 UTC by peterd