

Using the AWS command line interface to launch an EC2 server

Ronald (Ryy) Glenn Thomas

2023-06-20

Table of contents

1	Introduction	1
2	Scripts	2
2.1	Create security group script	3
2.2	Create new key pair with a project name flag . .	4
2.3	Generate instance	4
2.4	Appendix 1 Set up AWS IAM	7

1 Introduction

In a separate post ([here](#)) we discuss setting up a virtual server on Amazon Web Services (AWS) using the interactive Elastic cloud Compute (EC2) dashboard. While its instructive to use the EC2 console interface to set up a work environment and launch a custom server, it can become a tedious process after the first two or three times. In this post we'll present bash shell scripts to perform the same task making use of the AWS command line interface (CLI).

To get started: on your workstation, configure the aws cli app via the command.



Photo by Nathan Waters on Unsplash

```
zsh> brew install awscli
```

Instructions for installing the AWS CLI can be found ([here](#)).

```
zsh> aws configure
```

The app will open a dialog asking for your IAM credentials. If you don't have an IAM ID Appendix 1 [here](#) has details on obtaining IAM credentials from your AWS account.

2 Scripts

nine parameters are required. Determine them and store as environment variables

```
# vpc_id and subnet_id are determined by user location
export vpc_id="vpc-14814b73"
export subnet_id="subnet-f02c90ab"
# ami_id, storage_size and instance_type define the OS and capability of the server
export ami_id="ami-014d05e6b24240371"
export instance_type="t2.micro"
export storage_size="30"
# key_name and security group identify the ssh and encrypted web channels
export key_name="aws_key"
export security_grp="sg-0fef542d93849669c"
# static_ip and domain_name identify the server on the web
export static_ip="13.57.139.31"
```

Below we offer four bash scripts.

- 1) The first generates a security group for the virtual server, i.e. a firewall.
- 2) The second creates a key pair to allow encrypted ssh communication between the server and your workstation.
- 3) The third script generates the virtual server taking instance characteristics, firewall, static IP and domain name as parameters.

- 4) The fourth script installs required software following server launch.

2.1 Create security group script

```
#!/usr/bin/env bash
# The script generates a new security group
# the group name is "max_restrict"
# only ports 22 and 443 are open.
# to open other ports replicate the last paragraph and change the port number.
# Will fail if group name "max_restrict" is already in use.
# reads vpc_id from the environment variables set in .zshrc
#
aws ec2 create-security-group \
    --group-name max_restrict \
    --description "most restrictive: ports 22 and 443 only" \
    --tag-specifications \
    'ResourceType=security-group,Tags=[{Key=Name,Value=max_restrict}]' \
    --vpc-id $vpc_id
wait
export security_grp=`aws ec2 describe-security-groups | \
jq -r '.SecurityGroups[] | select(.GroupName=="max_restrict").GroupId`

aws ec2 authorize-security-group-ingress \
    --group-id $security_grp \
    --protocol tcp \
    --port 22 \
    --cidr "0.0.0.0/0"

aws ec2 authorize-security-group-ingress \
    --group-id $security_grp \
    --protocol tcp \
    --port 443 \
    --cidr "0.0.0.0/0"
```

2.2 Create new key pair with a project name flag

```
#!/usr/bin/env bash
base=`basename $PWD`
if [ -z "$1" ]
then
    key_pair_name=$base
else
    key_pair_name="$1"
fi

echo "key_pair_name is $key_pair_name"

read -p "Continue (y/n)?" CONT
if [ "$CONT" = "y" ]; then
    echo "Here we go!";
else
    echo "too bad. bye."; exit;
fi

cd ~/.ssh
rm -f ~/.ssh/$key_pair_name.pem
aws ec2 create-key-pair --key-name $key_pair_name \
    --query 'KeyMaterial' --output text > ~/.ssh/$key_pair_name.pem

wait
chmod 600 ~/.ssh/$key_pair_name.pem
```

2.3 Generate instance

start up script. aws_create_instance.sh

```
#!/usr/bin/env bash
while getopts s:t:k:p: flag
do
```

```

        case "${flag}" in
            s) size=${OPTARG};;
            t) type=${OPTARG};;
            k) key_name=${OPTARG};;
            p) proj_name=${OPTARG};;
        esac
done
base=`basename $PWD`
if [ -z "$proj_name" ]
then
    proj_name=$base
fi

if [ -z "$type" ]
then
    type="t2.micro"
fi

if [ -z "$size" ]
then
    size=30
fi

echo "Review parameters: "
echo "---"
echo "proj_name is $proj_name"
echo "key_name is $key_name"
echo "vpc_id: $vpc_id";
echo "subnet_id: $subnet_id";
echo "ami_id: $ami_id";
echo "security_grp: $security_grp";
echo "static_ip: $static_ip";
echo "type: $type";
echo "size: $size";

read -p "Review Notes (y/n)?" NOTES
if [ "$NOTES" = "y" ]; then

```

```

echo "Notes on current parameters:"
echo "security group should be in place already. check on EC2.
If not, run ./awscli_create_security.sh.
Key pair should be in place. check in ~/.ssh.
If not run ./create_keypair.sh.
ami id is for ubuntu linux 22.04 LTS.
If not what is desired check EC2 list of instances. ";
else
    echo "I guess you know what you're doing";
fi

read -p "Continue (y/n)?" CONT
if [ "$CONT" = "y" ]; then
    echo "Here we go!";
else
    echo "too bad. bye."; exit;
fi

aws ec2 run-instances \
    --image-id $ami_id \
    --count 1 \
    --instance-type $type \
    --key-name $key_name \
    --security-group-ids $security_grp \
    --subnet-id $subnet_id \
    --block-device-mappings "[{\"DeviceName\":\"/dev/sda1\",\"Ebs\":{\"VolumeSize\":$size}}]" \
    --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=$proj_name}]" \
    --user-data file://~/Dropbox/prj/c060/aws_startup_code.sh
wait
iid=`aws ec2 describe-instances --filters "Name=tag:Name,Values=$proj_name" | \
    jq -r '.Reservations[].Instances[].InstanceId'`
aws ec2 wait instance-running --instance-ids $iid
echo "the generated instance has ID: $iid"
running=`aws ec2 describe-instance-status --instance-id $iid
echo "the instance status is $running"
aws ec2 associate-address --public-ip $static_ip --instance-id $iid`

```

aws_startup.sh

```
#!/bin/bash
sudo apt update
sudo apt install docker.io -y
sudo apt install -y curl debian-keyring debian-archive-keyring apt-transport-https
curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/gpg.key' | \
sudo gpg --dearmor -o /usr/share/keyrings/caddy-stable-archive-keyring.gpg
curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/debian.deb.txt' | \
sudo tee /etc/apt/sources.list.d/caddy-stable.list
sudo apt update
sudo apt install caddy -y
```

Tip 1.

For convenience, construct a config file in ~/.ssh as:

```
Host rgtlab.org
HostName 13.57.139.31 # static IP
User ubuntu # default user on ubuntu server
Port 22 # the default port ssh uses
IdentityFile ~/.ssh/power1_app.pem
```

then we can ssh into the new server with

```
sh> ssh rgtlab.org
```

Change the access permissions: `sudo chmod 600 power1ssh.pem`
to be more restrictive.

2.4 Appendix 1 Set up AWS IAM

To initiate batch processing via the AWS cli app. Set up `aws` access via the `aws configure` program.

To get the needed credentials to configure command line `aws` use the AWS IAM service.

Details follow:

Log into AWS console.

Search for **IAM service**. Navigate to IAM dashboard.

Select **Users** in left hand panel.

Then select **Add Users** button (in upper right).

Then enter a **User name** in the form. Click **Next** (lower right)

Then **Create User**.

Click on the user name

In the page that comes up. Select **Security Credentials** tab (center of page).

Under **Access Keys** panel click **Create access key** (right side or bottom of panel).

Click **Command Line Interface CLI**

and at the bottom of the page click the checkbox “I understand...”.

Finally select **Create access key** and

choose **Download .csv file** (lower right).

Navigate Download screen to local `~/.aws` directory.

Click **Done**

Now in the terminal on your workstation, configure the aws cli app via the command.

```
aws configure
```

Enter info from the credentials file just downloaded. After entering the **AWS Access Key ID** and **AWS Secret Access Key** information you are asked for a Region, (my region is **us-west-1**), and an output format (suggested output format is **JSON**).