Postgres in the Cloud: The Hard Way

BRUCE MOMJIAN

There are many ways to easily install Postgres in the cloud strictly from the command line.

https://momjian.us/presentations

Last updated: June 2024

Outline

1. Why Do This?

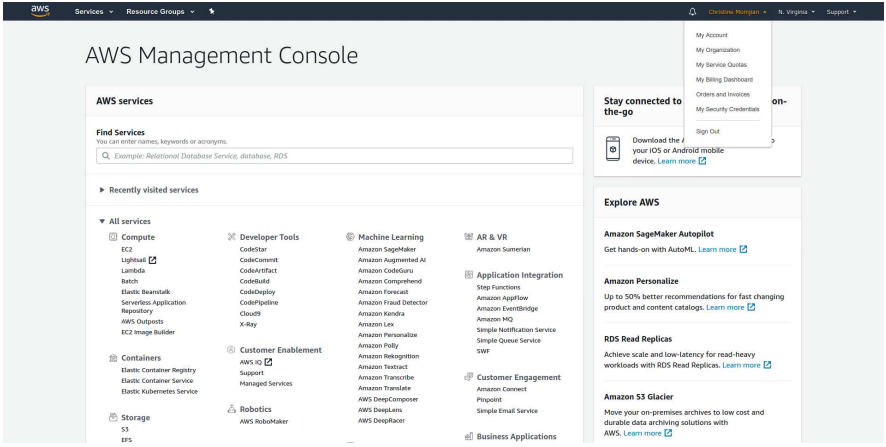
There are many ways to use the cloud

- GUI: AWS console, RDS
- Packages: RPM, DEB, installers
- Containers: Docker, Kubernetes
- Orchestration software: Puppet, Chef, Ansible, Terraform

What Are We Going to Use?

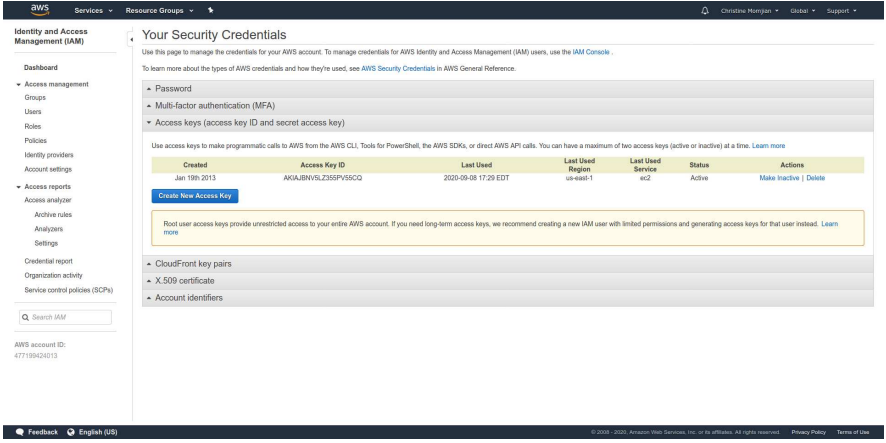
- Debian 10 (Buster)
- *awscli*
- AWS console
- PostgreSQL source code

2. Setting Up *awscli*



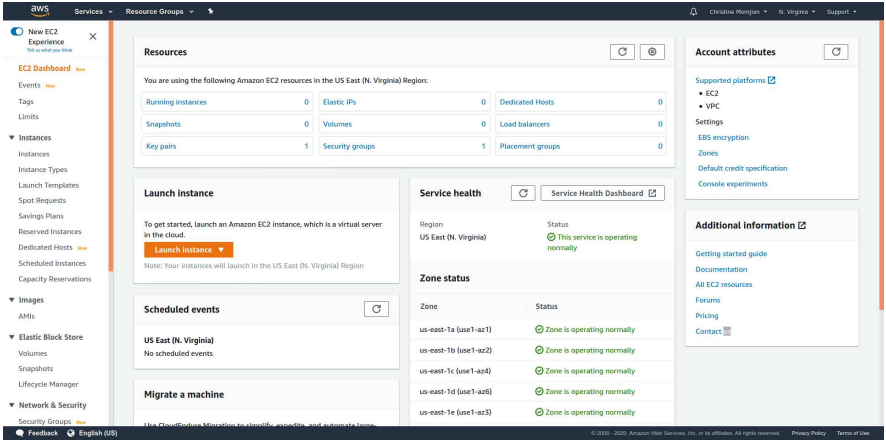
https://console.aws.amazon.com/console

Create an Access Key



https://console.aws.amazon.com/iam/#/security_credentials

Ec2 Console



https://console.aws.amazon.com/ec2/v2/

Install *awscli*

```
# apt-get install awscli
```

Configure *awscli*

```
$ aws configure
AWS Access Key ID [None]: XXXX
AWS Secret Access Key [None]: YYYY
Default region name [None]: us-east-1
Default output format [None]: text
```

https://aws.amazon.com/cli/

Create a Key Pair

```
$ aws ec2 create-key-pair --key-name AWS-ssh > "$HOME"/.aws/AWS-ssh.pem
$ chmod 0400 "$HOME"/.aws/AWS-ssh.pem
```

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html

Getting *awscli* Help

```
$ aws help
AWS()                                                                AWS()

NAME
       aws -

DESCRIPTION
       The  AWS  Command  Line  Interface is a unified tool to manage your AWS
       services.
SYNOPSIS
          aws [options] <command> <subcommand> [parameters]
       Use aws command help for information on a  specific  command.  Use  aws
       help  topics  to view a list of available help topics. The synopsis for
       each command shows its parameters and their usage. Optional  parameters
       are shown in square brackets.
```

Getting *awscli* Help

```
$ aws ec2 help
$ aws ec2 run-instances help
$ aws ec2 authorize-security-group-ingress help
```

3. Choosing an AMI

An Amazon Machine Image (AMI) is needed to initialize an Elastic Compute Cloud (EC2) instance. While any AMI can be used, it is ideal to restrict the selection to specific owners.

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/finding-an-ami.html

Debian AMIs

```
$ DEBIAN_AMI='136693071363'

$ aws ec2 describe-images \
        --owners "$DEBIAN_AMI" \
        --filters \
                'Name=state, Values=available' \
                'Name=architecture, Values=x86_64' \
                'Name=root-device-type, Values=ebs' \
                'Name=virtualization-type, Values=hvm' \
                'Name=description, Values=Debian*' \
        --query 'reverse(sort_by(Images, &CreationDate))[*].[CreationDate, \
                 ImageId, Description]' \
        --output text
```

Uses JMESPath, see https://jmespath.org/specification.html. All shell scripts in this presentation are at https://momjian.us/main/writings/pgsql/hard-shell.tgz.

Debian AMIs

Creation	Image ID	Description
2020-08-03T13:55:39.000Z	ami-05c0d7f3fffb419c8	Debian 10 (20200803-347)
2020-06-10T20:29:32.000Z	ami-0c24eddbea3a65909	Debian 10 (20200610-293)
2020-06-10T14:58:34.000Z	ami-080eb589703af6acf	Debian 10 (20200610-292)

4. Creating an EC2 Instance: What Will You Be Charged For?

- Instance running, per hour
- Storage, GB/month
- Storage I/O, provisioned IOPS
- Network output
- Elastic IP

https://www.apptio.com/blog/guide-to-aws-ec2-costs/

Find AMI Device

```
# Debian default root device
AMI='ami-05c0d7f3fffb419c8'  # from previous slide

DEVICE=$(aws ec2 describe-images \
    --filters "Name=image-id, Values=$AMI" \
    --query 'Images[*].RootDeviceName' --output text)
```

Cheap Setup

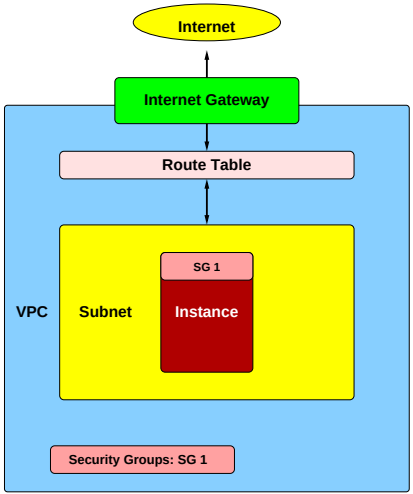
```
INSTANCE_OPTS='--instance-type t3a.nano \
               --credit-specification CpuCredits=standard'

EBS="--block-device-mappings \
     DeviceName='$DEVICE',Ebs={VolumeType='standard',VolumeSize=8}"

# us-east-1e doesn't have t3a.nano
AZONE='us-east-1f'
```

Creating an EC2 Instance

1. Create a Virtual Private Cloud (VPC), which also creates a security group and route table
2. Create an internet gateway and attach it to the VPC
3. Add a route table entry for the gateway
4. Create a subnet
5. Connect the subnet to the route table
6. Open the security group for ssh (port 22) and Postgres (port 5432)
7. Create an instance in the subnet

https://www.simplilearn.com/tutorials/aws-tutorial/aws-vpc

EC2 Internals



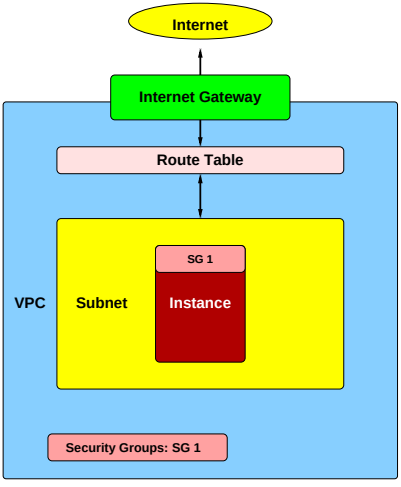
EC2 Internals

Internet gateway: allows traffic from the VPC to/from the Internet

Route table: allows traffic between subnets and to/from the gateway

Subnet: allows traffic between instances without using the route table

Security group: filters incoming traffic to an instance



Create VPC, With Security Group and Route Table

```
# In the web interface, deleting VPC deletes all dependent objects.b
VPC=$(aws ec2 create-vpc \
        --cidr-block 10.0.0.0/28 \
        --query 'Vpc.VpcId' \
        --output text)

# enable a public DNS entry for this VPC
aws ec2 modify-vpc-attribute \
        --vpc-id "$VPC" \
        --enable-dns-hostnames '{"Value": true}'
```

Create Gateway and Attach to VPC

```
GATEWAY=$(aws ec2 create-internet-gateway \
        --query 'InternetGateway.InternetGatewayId' \
        --output text)

aws ec2 attach-internet-gateway \
        --vpc-id "$VPC" \
        --internet-gateway-id "$GATEWAY"
```

Add Route Table Entry for the Gateway

```
# get route table
ROUTETBL=$(aws ec2 describe-route-tables \
        --filters "Name=vpc-id, Values=$VPC" \
        --query 'RouteTables[*].RouteTableId' \
        --output text)

aws ec2 create-route \
        --route-table-id "$ROUTETBL" \
        --destination-cidr-block 0.0.0.0/0 \
        --gateway-id "$GATEWAY"
```

Create Subnet

```
SUBNET=$(aws ec2 create-subnet \
        --availability-zone "$AZONE" \
        --vpc-id "$VPC" \
        --cidr-block 10.0.0.0/28 \
        --query 'Subnet.SubnetId' \
        --output text)
```

Connect the Subnet to the Route Table

```
aws ec2 associate-route-table \
        --subnet-id "$SUBNET" \
        --route-table-id "$ROUTETBL"
```

Adjust Security Group

```
# get security group
SECGROUP=$(aws ec2 describe-security-groups \
        --filters "Name=vpc-id, Values=$VPC" \
        --query 'SecurityGroups[*].GroupId' \
        --output text)

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

# Postgres
aws ec2 authorize-security-group-ingress \
        --group-id "$SECGROUP" \
        --protocol tcp --port 5432 --cidr 0.0.0.0/0
```

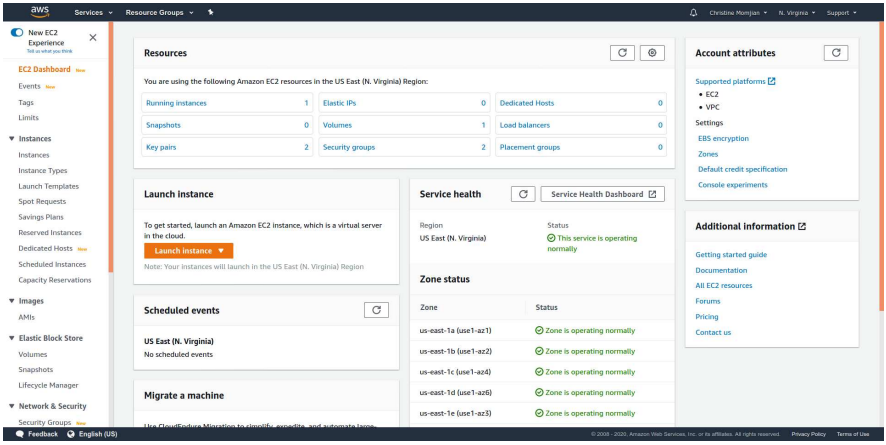
Create Instance in the Subnet

```
INSTANCE=$(aws ec2 run-instances \
        --image-id "$AMI" \
        --subnet-id "$SUBNET" \
        --associate-public-ip-address \
        $INSTANCE_OPTS \
        $EBS \
        --security-group-ids "$SECGROUP" \
        --key-name AWS-ssh \
        --tag-specifications "ResourceType=instance,\
              Tags=[{Key=Name, Value=Debian-default}]" \
        --query 'Instances[*].InstanceId' \
        --output text)
```

Start the Instance

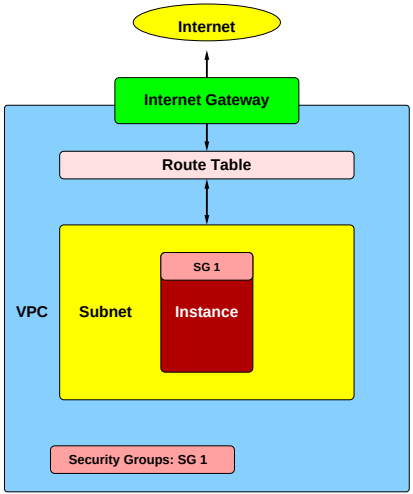
```
aws ec2 start-instances --instance-id "$INSTANCE"

# get instance status
aws ec2 describe-instances --filters Name=instance-state-name, \
        Values=pending,running,shutting-down,stopping,stopping,stopped \
        --query sort_by(Reservations, \
&Instances[0].BlockDeviceMappings[0].Ebs.AttachTime)[*].Instances[0].[InstanceId, \
        BlockDeviceMappings[0].Ebs.AttachTime, LaunchTime, State.Name] \
        --output text
```

Running EC2 Console



https://console.aws.amazon.com/ec2/v2/

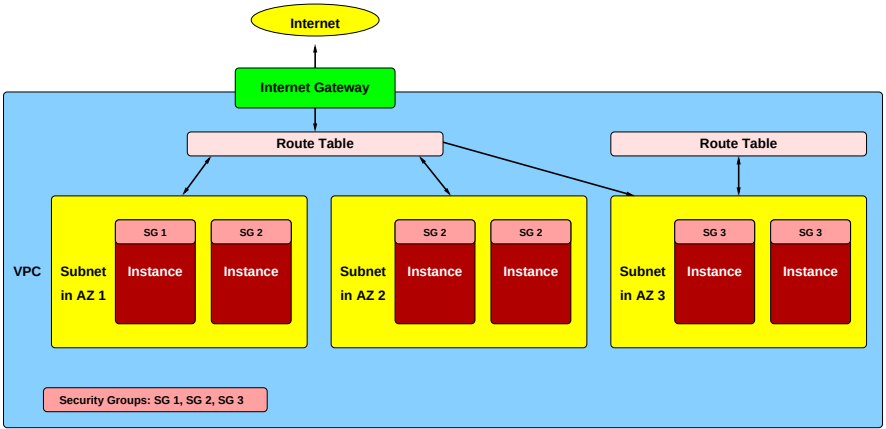
EC2 Internals



More Complexity

- A single security group can be assigned to multiple instances
- Multiple instances can be placed in a subnet
- Multiple subnets can use the same route table
- A VPC can have only one internet gateway

Complex Configuration



5. Logging in and Configuring

```
# disable host key checking
# http://linuxcommando.blogspot.com/2008/10/
how-to-disable-ssh-host-key-checking.html
SSH_OPT='-o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no'

LOGIN_USER='admin'
HOST=$(aws ec2 describe-instances --instance-ids "$INSTANCE" \
        --query 'Reservations[*].Instances[*].PublicDnsName' --output text)

ssh -i "$HOME"/.aws/AWS-ssh.pem $SSH_OPT "$LOGIN_USER"@"$HOST"
```

Setup Environment

```
$ ssh -i "$HOME"/.aws/AWS-ssh.pem $SSH_OPT \
    admin@ec2-18-205-56-189.compute-1.amazonaws.com
Linux ip-10-0-0-5 4.19.0-10-cloud-amd64 #1 SMP Debian 4.19.132-1 …
…
admin@ip-10-0-0-5:~$ exec sudo --login
root@ip-10-0-0-5:~# PS1='aws# '
aws# apt-get update &&b
> apt-get -y install build-essential libreadline-dev zlib1g-dev &&
> apt-get -y install mutt htop dnsutils
```

Setup Shell Scripts

```
aws# ln -s /usr/local/bin /usr/lbin &&
> echo 'exec ls -CF "$@"' > /usr/local/bin/lf &&
> echo 'exec ls -l "$@"' > /usr/local/bin/ll &&
> chmod +x /usr/local/bin/l[fl]
```

Setup Email

```
aws# # https://unix.stackexchange.com/questions/20570/
mutt-how-to-safely-store-password
# set up SMTP authentication
cat <<END_MUTT > .muttrc
set smtp_url = "smtp://laptop@smtp.momjian.us:25/"
# PASSWORD HERE
set smtp_pass = "XXXXXX"
set from = "bruce@momjian.us"
set realname = "Bruce Momjian"
END_MUTT
```

Set Prompts

```
aws# echo 'export PS1="aws\$ "' >> ~/.bashrc &&
> echo 'export PATH=$PATH:/usr/local/pgsql/bin:.' >> ~/.profile
```

Set Environment Variables

```
aws# cat <<'PROF_END' >> ~/.profile
> # use REST API, https://www.1strategy.com/blog/2018/12/11/creating-dynamic-scripts
> export PRVHOST=$(wget -q -O - \
>        'http://instance-data/latest/meta-data/local-hostname')
> export PUBHOST=$(wget -q -O - \
>        'http://instance-data/latest/meta-data/public-hostname')
> PROF_END
```

Cleanup

```
aws# echo 'syntax off' >> ~/.vimrc &&
> echo 'exec sudo --login' >> ~admin/.profile
```

6. Installing Postgres

```
aws# PGVER='12.4'

aws# wget \
> https://ftp.postgresql.org/pub/source/v$PGVER/postgresql-$PGVER.tar.bz2 &&
> bzcat postgresql-$PGVER.tar.bz2 | tar xf -

aws# cd postgresql-$PGVER

aws# ./configure &&
> make &&
> make install

aws# adduser --quiet --gecos 'Postgres' --disabled-login postgres

aws# echo 'export PS1="aws\$ "' >> ~postgres/.bashrc &&
> echo 'export PATH=$PATH:/usr/local/pgsql/bin:.' >> ~postgres/.profile
```

Creating the Data Directory

```
aws# . ~/.profile # set PATH
aws# mkdir /usr/local/pgdata
aws# chown postgres.postgres /usr/local/pgdata
aws# chmod 0700 /usr/local/pgdata

aws$ su postgres -c 'initdb /usr/local/pgdata'
```

Configuring Security

```
aws# su postgres
aws# cd /usr/local/pgdata
aws# echo 'host all all 0.0.0.0/0 scram-sha-256' >> pg_hba.conf

aws# sed \
    -e 's/#password_encryption = md5/password_encryption = scram-sha-256/' \
    -e "s/#listen_addresses = 'localhost'/listen_addresses = '\*'/" \
     postgresql.conf > /tmp/$$ && mv /tmp/$$ postgresql.conf

aws# pg_ctl -l /usr/local/pgdata/server.log -D /usr/local/pgdata start
```

Configuring Password

```
aws$ psql postgres
psql (12.4)
Type "help" for help.

postgres=# \password
Enter new password:
Enter it again:
postgres=#
```

7. Connecting to Postgres

```
$ # no ssl, no certificate verification, no channel binding
$ psql -h ec2-18-205-56-189.compute-1.amazonaws.com postgres
Password for user postgres:
psql (14devel, server 12.4)
Type "help" for help.

postgres=> SELECT inet_server_addr();
 inet_server_addr
------------------
 10.0.0.5
```

Using SSH Tunneling

```
$ ssh -i "$HOME"/.aws/AWS-ssh.pem $SSH_OPT \
      -L 63333:localhost:5432 "$LOGIN_USER"@"$HOST"
aws$ # keep open

$ psql -h localhost -p 63333 postgres
psql (14devel, server 12.4)
Type "help" for help.

postgres=> SELECT inet_server_addr();
 inet_server_addr
------------------
 127.0.0.1
```

Conclusion

https://momjian.us/presentations

https://www.flickr.com/photos/adai/