

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>



Creative Commons Attribution License

Last updated: June 2024

Outline

1. Why do this?
2. Setting up *awscli*
3. Choosing an AMI
4. Creating an EC2 instance
5. Logging in and configuring
6. Installing Postgres
7. Connecting to Postgres

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*

The screenshot shows the AWS Management Console interface. At the top, the navigation bar includes the AWS logo, 'Services', 'Resource Groups', and user information for 'Christine Morgan' in 'N. Virginia'. The main heading is 'AWS Management Console'. Below this, there's a 'Find Services' search bar with an example query 'Example: Relational Database Service, database, RDS'. The 'All services' section is organized into a grid of categories:

- Compute:** EC2, Lightsail, Lambda, Batch, Elastic Beanstalk, Serverless Application Repository, AWS Outposts, EC2 Image Builder.
- Developer Tools:** CodeStar, CodeCommit, CodeArtifact, CodeBuild, CodeDeploy, CodePipeline, Cloud9, X-Ray.
- Machine Learning:** Amazon SageMaker, Amazon Augmented AI, Amazon CodeGuru, Amazon Forecast, Amazon Fraud Detector, Amazon Kendra, Amazon Lex, Amazon Personalize, Amazon Polly, Amazon Rekognition, Amazon Transact, Amazon Transcribe, Amazon Translate, AWS DeepComposer, AWS DeepLens, AWS DeepRacer.
- AR & VR:** Amazon Sumerian.
- Application Integration:** Step Functions, Amazon AppFlow, Amazon EventBridge, Amazon MQ, Simple Notification Service, Simple Queue Service, SWF.
- Customer Engagement:** Amazon Connect, Pinpoint, Simple Email Service.
- Business Applications:** (No specific services listed).
- Containers:** Elastic Container Registry, Elastic Container Service, Elastic Kubernetes Service.
- Storage:** S3, SFS.
- Customer Enablement:** AWS IQ, Support, Managed Services.
- Robotics:** AWS RoboMaker.

On the right side, there's a 'Stay connected to the-go' section with a mobile app download prompt and an 'Explore AWS' section featuring 'Amazon SageMaker Autopilot', 'Amazon Personalize', 'RDS Read Replicas', and 'Amazon S3 Glacier' with brief descriptions and 'Learn more' links.

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

Create an Access Key

Identity and Access Management (IAM)

- Dashboard
- Access management
 - Groups
 - Users
 - Roles
 - Policies
- Identity providers
- Account settings
- Access reports
 - Access analyzer
 - Archive rules
 - Analysts
 - Settings
- Credential report
- Organization activity
- Service control policies (SCPs)

Search IAM

AWS account ID: 477199424013

Your Security Credentials

Use this page to manage the credentials for your AWS account. To manage credentials for AWS Identity and Access Management (IAM) users, use the [IAM Console](#).

To learn more about the types of AWS credentials and how they're used, see [AWS Security Credentials](#) in AWS General Reference.

- ▲ Password
- ▲ Multi-factor authentication (MFA)
- ▼ Access keys (access key ID and secret access key)

Use access keys to make programmatic calls to AWS from the AWS CLI, Tools for PowerShell, the AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. [Learn more](#)

Created	Access Key ID	Last Used	Last Used Region	Last Used Service	Status	Actions
Jan 19th 2013	AKIAJBNVSLZ355PV56CQ	2020-09-08 17:29 EDT	us-east-1	ec2	Active	Make Inactive Delete

[Create New Access Key](#)

Root user access keys provide unrestricted access to your entire AWS account. If you need long-term access keys, we recommend creating a new IAM user with limited permissions and generating access keys for that user instead. [Learn more](#)

- ▲ CloudFront key pairs
- ▲ X.509 certificate
- ▲ Account identifiers

Feedback English (US)

© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy Policy](#) [Terms of Use](#)

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

Ec2 Console

The screenshot shows the AWS Management Console for the EC2 service. The top navigation bar includes the AWS logo, 'Services', 'Resource Groups', and user information for 'Christine Monjan' in the 'N. Virginia' region. A sidebar on the left contains navigation options like 'New EC2 Experience', 'EC2 Dashboard', 'Events', 'Tags', 'Limits', 'Instances', 'Images', 'Elastic Block Store', and 'Network & Security'. The main content area is divided into several panels:

- Resources:** A summary table showing the following counts for Amazon EC2 resources in the US East (N. Virginia) Region:

Running instances	0	Elastic IPs	0	Dedicated Hosts	0
Snapshots	0	Volumes	0	Load balancers	0
Key pairs	1	Security groups	1	Placement groups	0
- Launch Instance:** A section with instructions on how to launch an Amazon EC2 instance and a prominent orange 'Launch Instance' button. A note states: 'Your instances will launch in the US East (N. Virginia) Region.'
- Service health:** Shows the region as 'US East (N. Virginia)' and the status as 'This service is operating normally'.
- Zone status:** A table listing the availability zones and their status:

Zone	Status
us-east-1a (use1-az1)	Zone is operating normally
us-east-1b (use1-az2)	Zone is operating normally
us-east-1c (use1-az4)	Zone is operating normally
us-east-1d (use1-az6)	Zone is operating normally
us-east-1e (use1-az3)	Zone is operating normally
- Scheduled events:** Shows 'US East (N. Virginia)' with 'No scheduled events'.
- Migrate a machine:** A section for migrating machines.
- Account attributes:** Lists supported platforms (EC2, VPC), settings (EBS encryption, Zones, Default credit specification, Console experiments), and additional information (Getting started guide, Documentation, All EC2 resources, Forums, Pricing, Contact).

<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]: YYY  
Default region name [None]: us-east-1  
Default output format [None]: text
```

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

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}"
```

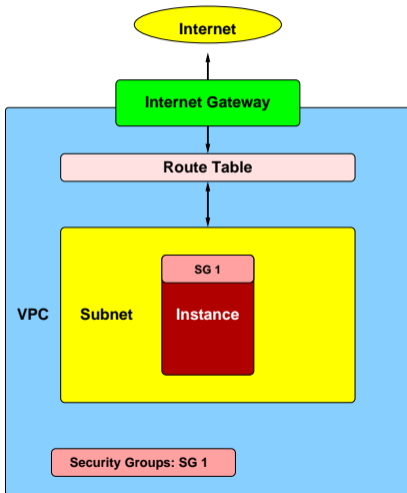
```
# us-east-1e doesn't have t3a.nano
```

```
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

EC2 Internals



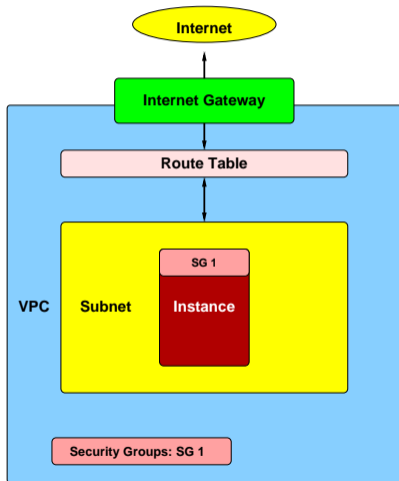
EC2 Internals

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.

```
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,stopped,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

Resources

You are using the following Amazon EC2 resources in the US East (N. Virginia) Region:

Running instances	1	Elastic IPs	0	Dedicated Hosts	0
Snapshots	0	Volumes	1	Load balancers	0
Key pairs	2	Security groups	2	Placement groups	0

Launch Instance

To get started, launch an Amazon EC2 Instance, which is a virtual server in the cloud.

[Launch Instance](#)

Note: Your instances will launch in the US East (N. Virginia) Region

Scheduled events

US East (N. Virginia)
No scheduled events

Migrate a machine

Service health

Region: US East (N. Virginia) | Status: ✔ This service is operating normally

Zone status

Zone	Status
us-east-1a (use1-az1)	✔ Zone is operating normally
us-east-1b (use1-az2)	✔ Zone is operating normally
us-east-1c (use1-az4)	✔ Zone is operating normally
us-east-1d (use1-az6)	✔ Zone is operating normally
us-east-1e (use1-az3)	✔ Zone is operating normally

Account attributes

Supported platforms: [EC2](#), [VPC](#)

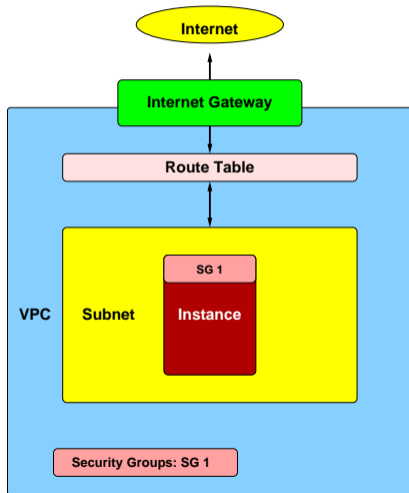
Settings: [EBS encryption](#), [Zones](#), [Default credit specification](#), [Console experiments](#)

Additional information

Getting started guide: [Documentation](#), [All EC2 resources](#), [Forums](#), [Pricing](#), [Contact us](#)

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

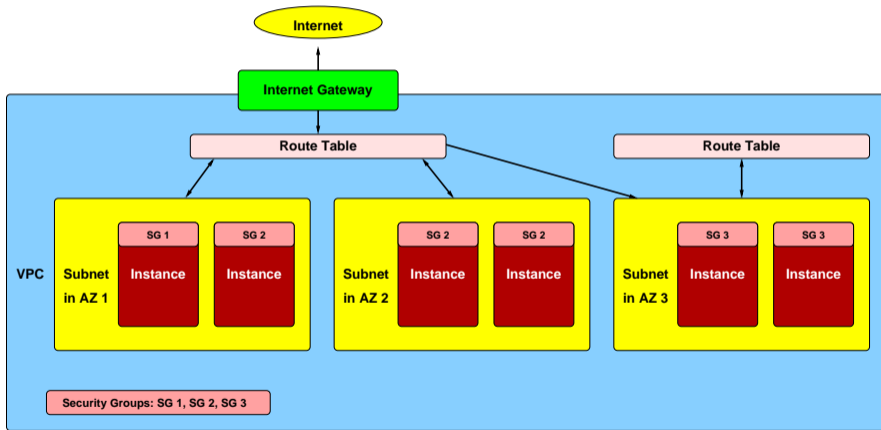
EC2 Internals



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[f1]
```

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 &&  
> bzipcat 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/>