CC Mini Project Report
Mail Server on Amazon EC2

Introduction:

In this project, we aim to demonstrate the elasticity in cloud computing by using infrastructure as a service. For the same we have demonstrated setting up a mail server using Amazon EC2 server. Mail Server here represents a simple SMTP server used to send mail to any existing or custom email account. The same is done via cloud computing to ensure that the clients do not need to run any resource intensive servers for sending these mails.

Problem definition:

While there are a plethora of reasons to use the existing email systems, it is preferred that an organization uses its own email server. The reasons for the same are:
1. You have the necessary privacy.
2. You have more control over your email clients and blacklisting.
3. You are running your own internet accessible service.

There 35 AMI cloud services at your disposal at Amazon for a mail server. We can choose the simplest possible AMI since there is little processing at the server side to be done. We will be choosing **Amazon Linux AMI 2017.09.1 (HVM), SSD Volume Type** - ami-7c87d913.

For a small scale organization, we would be choosing a small level instance for this image. For eg, we can choose t2.micro instance. This instance comes with Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only storage.

EBS: Amazon Elastic Block Store (Amazon EBS) provides block level storage volumes for use with EC2 instances. EBS volumes are highly available and reliable storage volumes that can be attached to any running instance that is in the same Availability Zone. EBS volumes that are attached to an EC2 instance are exposed as storage volumes that persist independently from the life of the instance.

Amazon EBS is recommended when data must be quickly accessible and requires long-term persistence. EBS volumes are particularly well-suited for use as the primary storage for file systems, databases, or for any applications that require fine granular updates and access to raw, unformatted, block-level storage. Amazon EBS is well suited to both database-style applications that rely on random reads and writes, and to throughput-intensive applications that perform long, continuous reads and writes.

For simplified data encryption, you can launch your EBS volumes as encrypted volumes. Amazon EBS encryption offers you a simple encryption solution for your EBS volumes without the need for you to build, manage, and secure your own key management infrastructure. When you create an encrypted EBS volume and attach it to a supported instance type, data stored at rest on the volume, disk I/O, and snapshots created from the volume are all encrypted. The encryption occurs on the servers that hosts EC2 instances, providing encryption of data-in-transit from EC2 instances to EBS storage.

You can create EBS General Purpose SSD (gp2), Provisioned IOPS SSD (io1), Throughput Optimized

HDD (st1), and Cold HDD (sc1) volumes up to 16 TiB in size. You can mount these volumes as devices on your Amazon EC2 instances. You can mount multiple volumes on the same instance, but each volume can be attached to only one instance at a time. You can dynamically change the configuration of a volume attached to an instance.

For our project we will be using a small 8GB storage for presentation purpose.

Steps to set up the mail server:

Create DNS records
To start, you will need to point the following records to your instance IP address (A records):

mail.yourdomainname.com - this will be used by the mail server and mail clients to send/receive emails
webmail.yourdomainname.com - this domain will be used for webmail interface (Roundcube)
vim.yourdomainname.com - here we will host ViMbAdmin, which is responsible for accounts, aliases and groups management
Also, you will need to add an MX record:

```
yourdomain.com MX 0 mail.yourdomainname.com
```

Configure instance
Before setting up the mail server, there are a few steps you need to take to configure general server settings.

Configure partition
By default the second drive you created (50GB) will not be mounted and available - you need to create a partition and mount it first.

Launch fdisk for the second drive

```
$ sudo fdisk /dev/xvdb
```

Hit n to create new partition, then p for primary partition. Leave the rest default (Enter-Enter-Enter)
Hit w to write changes to disk and exit
Now that you have created a partition, it is time to create a filesystem on it:

```
$ sudo mkfs.ext4 /dev/xvdb1
```

Create a mountpoint

```
$ sudo mkdir /var/vmail/
```

Mount the partition:

```
$ sudo bash -c ‘echo “/dev/xvdb1 /var/vmail/ ext4 defaults 0 0” >> /etc/fstab’

$ sudo mount /var/vmail

$ sudo rm -rf /var/vmail/lost+found
```

Create swap file

EC2 instances do not have a swap file by default - let’s create one to improve system stability. This will create a 4GB swapfile in /var/swapfile and mount it:

```
$ sudo dd if=/dev/zero of=/var/swapfile bs=1M count=4096

$ sudo chmod 600 /var/swapfile

$ sudo mkswap /var/swapfile

$ sudo bash -c ‘echo “/var/swapfile /dev/null swap defaults 0 2” >> /etc/fstab’

$ sudo swapon -a
```

Install packages
Update package information:

```
$ sudo apt-get update
```

Install nginx and letsencrypt

```
$ sudo apt-get install nginx letsencrypt
```

Configure nginx
Create dhparam file:

```
$ sudo openssl dhparam -out /etc/nginx/dhparam.pem 2048
```

Replace contents of /etc/nginx/nginx.conf with the following: https://pastebin.com/iCddfxJC

And delete default vhost configuration file: /etc/nginx/sites-enabled/default

Lastly, restart nginx:

```
$ sudo service nginx restart
```

Configure letsencrypt
Letsencrypt allows us to obtain and renew a free fully-functional SSL certificate. Letsencrypt certificates have 3 months TTL so they must be renewed quite often.

Obtain SSL certificates

```
$ sudo letsencrypt certonly –webroot -w /tmp -d mail.yourdomainname.com

$ sudo letsencrypt certonly –webroot -w /tmp -d webmail.yourdomainname.com

$ sudo letsencrypt certonly –webroot -w /tmp -d vim.yourdomainname.com
```

Automatically renew certificates
To avoid any downtime and extra maintenance, add cron job to automatically renew certificates and reload all affected services:

```
$ sudo crontab -e
```

Add the following record to root’s crontab:

```
## This will renew all SSL certificates once a week and reload nginx, postfix and dovecot

0 3 * * 2 /usr/bin/letsencrypt renew; systemctl restart postfix; systemctl restart dovecot; systemctl reload nginx
```

Install and configure Postfix
Install packages

```
$ sudo apt-get install postfix postfix-mysql dovecot-core dovecot-imapd dovecot-pop3d dovecot-lmtpd dovecot-mysql mysql-server dovecot-sieve dovecot-managesieved
```

Depending on previously installed package list, you may or may not see prompts to configure mysql, postfix, etc.

You will be prompted to enter and confirm mysql root password. Use password generator to create a safe 16+ character password and note it down.

Next, you will need to choose Postfix configuration type. Select “Internet site”.

The next prompt will ask for the correct mail name. Enter the domain name you are configuring the mail server for.

Create a mysql database and user for vimbadmin
Log in to mysql using:

```
$ mysql -uroot -p
```

And enter mysql root password when prompted.

Then, run the following SQL:

```
CREATE DATABASE ‘vimbadmin’;

GRANT ALL ON ‘vimbadmin’.* TO ‘vimbadmin’@’127.0.0.1’ IDENTIFIED BY ‘password’;

FLUSH PRIVILEGES;
```

Configure Postfix
Navigate to /etc/postfix and edit master.cf configuration file:

```
$ cd /etc/postfix

$ sudo nano master.cf
```

There, find and uncomment the following two lines:

```
submission inet n       -       -       -       -       smtpd
```

```
smtps     inet  n       -       -       -       -       smtpd
```

Next, open mail.cf and add the following lines towards the end of the file. Don't forget to update the paths for tls cert and key files!

```
# Change postfix TLS parameter to use dovecot

#smtpd_tls_session_cache_database = btree:${data_directory}/smtpd_scache

#smtp_tls_session_cache_database = btree:${data_directory}/smtp_scache

smtpd_tls_cert_file=/etc/letsencrypt/live/mail.yourdomain.com/fullchain.pem

smtpd_tls_key_file=/etc/letsencrypt/live/mail.yourdomain.com/privkey.pem

smtpd_use_tls=yes

#smtpd_tls_auth_only = yes

#Handle SMTP authentication using Dovecot

smtpd_sasl_type = dovecot

smtpd_sasl_path = private/auth

smtpd_sasl_auth_enable = yes

smtpd_recipient_restrictions =

        permit_sasl_authenticated,

        permit_mynetworks,

        reject_unauth_destination

# other destination domains should be handled using virtual domains

mydestination = localhost

# using Dovecot's LMTP for mail delivery and giving it path to store mail

virtual_transport = lmtp:unix:private/dovecot-lmtp

# virtual mailbox setups

virtual_uid_maps = static:5000
```

virtual_gid_maps = static:5000

virtual_alias_maps = mysql:/etc/postfix/mysql/virtual_alias_maps.cf

virtual_mailbox_domains = mysql:/etc/postfix/mysql/virtual_domains_maps.cf

virtual_mailbox_maps = mysql:/etc/postfix/mysql/virtual_mailbox_maps.cf

Now, let’s create configuration to store all mailbox-related information in mysql:

$ sudo mkdir /etc/postfix/mysql

Create a file /etc/postfix/mysql/virtual_alias_maps.cf

$ sudo nano /etc/postfix/mysql/virtual_alias_maps.cf

And paste the following in it (don’t forget to put the correct mysql credentials in!):

user = user

password = password

hosts = 127.0.0.1

dbname = vimbadmin

query = SELECT goto FROM alias WHERE address = ‘%s’ AND active = ‘1’

Next, create /etc/postfix/mysql/virtual_domains_maps.cf

$ sudo nano /etc/postfix/mysql/virtual_domains_maps.cf

And paste the following:

user = user

password = password

hosts = 127.0.0.1

dbname = vimbadmin

query = SELECT domain FROM domain WHERE domain = ‘%s’ AND backupmx = ‘0’ AND active = ‘1’

Lastly, create /etc/postfix/mysql/virtual_mailbox_maps.cf

$ sudo nano /etc/postfix/mysql/virtual_mailbox_maps.cf

And paste the following:

```
user = user

password = password

hosts = 127.0.0.1

dbname = vimbadmin

query = SELECT maildir FROM mailbox WHERE username = ‘%s’ AND active = ‘1’
```

Dovecot configuration
Dovecot is an IMAP and POP server. It also implements security/authentication for IMAP/POP as well as SMTP (via Postfix).

Create Linux system user that will own all email in the system

```
$ sudo groupadd -g 5000 vmail

$ sudo useradd -g vmail -u 5000 vmail -d /var/vmail -m
```

Restart postfix

```
$ sudo service postfix restart
```

Enable required protocols
Edit /etc/dovecot/dovecot.conf

```
$ sudo nano /etc/dovecot/dovecot.conf
```

And enable required protocols (add the protocols = … line):

```
# Enable installed protocols

!include_try /usr/share/dovecot/protocols.d/*.protocol
protocols = imap pop3 lmtp sieve
```

Configure mail storage location
Edit /etc/dovecot/conf.d/10-mail.conf

```
$ sudo nano /etc/dovecot/conf.d/10-mail.conf
```

And set mail_location to:

```
mail_location = maildir:/var/vmail/%d/%n
```

Configure authentication
Edit /etc/dovecot/conf.d/10-auth.conf and make sure that the following options are set to:

```
disable_plaintext_auth = no
```

```
auth_mechanisms = plain login
```

Also comment out line: !include auth-system.conf.ext to disable system user authentication.

Add the following lines at the end of the file:

```
passdb {

   driver = sql
   args = /etc/dovecot/dovecot-sql.conf.ext

}

userdb {
   driver = static
   args = uid=5000 gid=5000 home=/var/vmail/%d/%n allow_all_users=yes

}
```

Configure mysql parameters in dovecot
Edit /etc/dovecot/dovecot-sql.conf.ext and paste the following at the bottom:

```
driver = mysql
connect = host=127.0.0.1 dbname=vimbadmin user=user password=password
password_query = \
SELECT username AS user, password, \
   homedir AS userdb_home, uid AS userdb_uid, gid AS userdb_gid \
 FROM mailbox WHERE username = '%u'
iterate_query = SELECT username AS user FROM mailbox
```

Change master config file
Edit /etc/dovecot/conf.d/10-master.conf and make sure it looks like this:

```
service lmtp {
unix_listener /var/spool/postfix/private/dovecot-lmtp {
  mode = 0600
  user = postfix
  group = postfix
 }
}
service auth {
unix_listener /var/spool/postfix/private/auth {
   mode = 0666
   user = postfix
   group = postfix

 }

 unix_listener auth-userdb {
   mode = 0600
   user = vmail

 }
 user = dovecot
}
service auth-worker {
 user = vmail
}
```

Configure Logging
Edit /etc/dovecot/conf.d/10-logging.conf and set

```
log_path = /var/log/dovecot.log
```

Restart Dovecot

```
$ sudo service dovecot restart
```

Install and configure ViMbAdmin
Install PHP and composer
For Ubuntu 16.04 default PHP version is 7.0.x:

Install packages

```
$ sudo apt-get install php7.0-fpm php7.0-mcrypt php-memcache php7.0-json php7.0-mysql php-gettext
php7.0-zip

$ sudo curl -sS https://getcomposer.org/installer | php

$ sudo mv composer.phar /usr/local/bin/composer
```

Set time zone in PHP
Edit /etc/php/7.0/fpm/php.ini

```
$ sudo nano /etc/php/7.0/fpm/php.ini
```

And set timezone: date.timezone = UTC

Then, restart php:

```
$ sudo service php7.0-fpm restart
```

Install ViMbAdmin

```
$ cd /usr/local

$ sudo git clone git://github.com/opensolutions/ViMbAdmin.git vimbadmin

$ sudo git checkout 3.0.15
$ cd /usr/local/vimbadmin

$ sudo composer install

$ sudo chown -R www-data: /usr/local/vimbadmin
```

Edit vimbadmin config file

```
$ cd /usr/local/vimbadmin

$ sudo cp application/configs/application.ini.dist application/configs/application.ini

$ sudo nano application/configs/application.ini
```

And update the contents:

```
securitysalt = “superadmin-password”  ← Generate super-secure password here

defaults.mailbox.uid = 5000

defaults.mailbox.gid = 5000

defaults.mailbox.homedir = “/var/vmail/”

resources.doctrine2.connection.options.driver   = ‘pdo_mysql’

resources.doctrine2.connection.options.dbname   = ‘vimbadmin’  ← Don’t forget to update this!

resources.doctrine2.connection.options.user     = ‘vimbadmin’  ← Don’t forget to update this!

resources.doctrine2.connection.options.password = ‘password’ ← Don’t forget to update this!

resources.doctrine2.connection.options.host     = ‘localhost’
```

Create mysql tables

```
$ cd /usr/local/vimbadmin

$ sudo mv public/.htaccess.dist .htaccess

$ sudo ./bin/doctrine2-cli.php orm:schema-tool:create

$ sudo chown -R www-data:www-data /usr/local/vimbadmin
```

Configure nginx

Create file /etc/nginx/sites-enabled/vma.yourdomain.com and paste the following:

```
server {

listen 443 ssl;
server_name vma.yourdomain.com;

ssl_certificate /etc/letsencrypt/live/vma.yourdomain.com/fullchain.pem;

ssl_certificate_key /etc/letsencrypt/live/vma.yourdomain.com/privkey.pem;

access_log   /var/log/nginx/vma.yourdomain.com.access.log;

error_log    /var/log/nginx/vma.yourdomain.com.error.log;

root /usr/local/vimbadmin/public;

index index.php;

location / {

try_files $uri $uri/ /index.php?$args;

}

location ~ \.php$ {

try_files $uri =404;

        fastcgi_split_path_info ^(.+\.php)(/.+)$;

        fastcgi_param   SCRIPT_FILENAME $document_root$fastcgi_script_name;

include fastcgi_params;

fastcgi_pass unix:/var/run/php/php7.0-fpm.sock;

}

}
```

Install and configure RoundCube
RoundCube is webmail interface. Since email can be checked using any email client, this part is optional but recommended for convenience.

Install packages

```
$ sudo apt-get install roundcube roundcube-plugins roundcube-plugins-extra
```

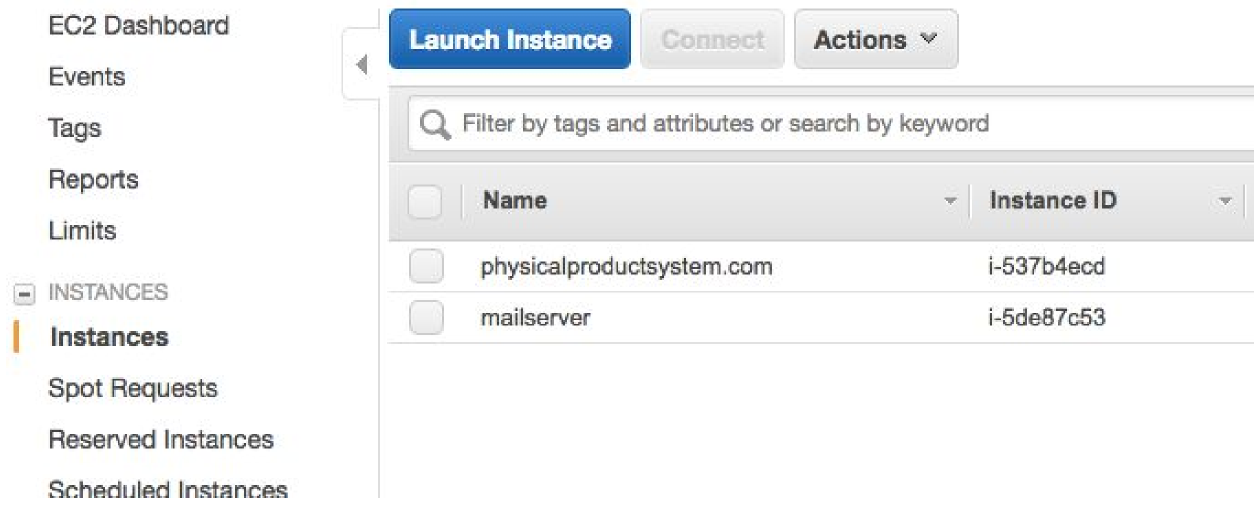
Answer “Yes” to “Configure database for roundcube with dbconfig-common? ” prompt.

Configure nginx
Create /etc/nginx/sites-enabled/webmail.yourdomain.com and paste there the following:

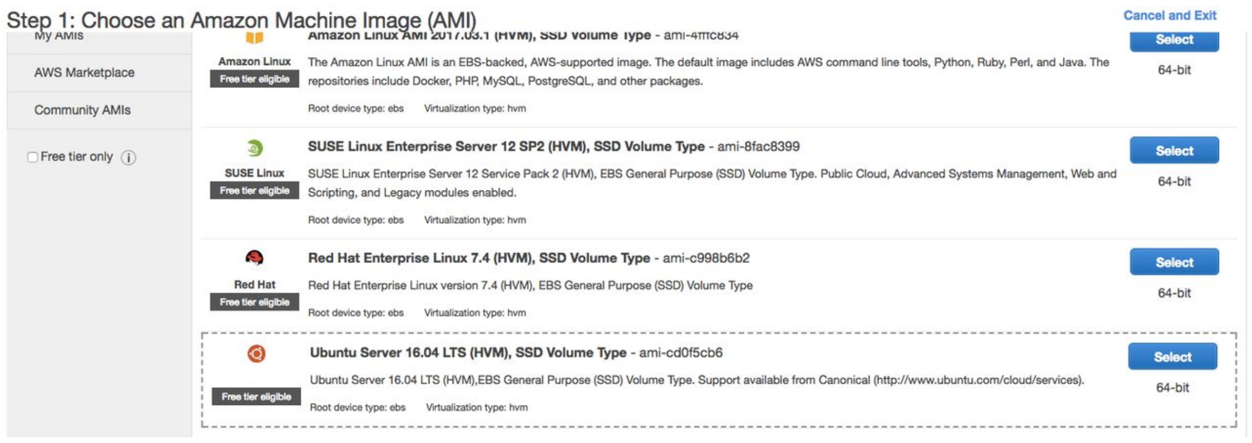
```
server {

listen 443 ssl;

ssl_certificate /etc/letsencrypt/live/webmail.yourdomain.com/fullchain.pem;

ssl_certificate_key /etc/letsencrypt/live/webmail.yourdomain.com/privkey.pem;

server_name webmail.yourdomain.com;

access_log   /var/log/nginx/webmail.yourdomain.com.access.log;

error_log    /var/log/nginx/webmail.yourdomain.com.error.log;

root /usr/share/roundcube;

index index.php;

location / {

try_files $uri $uri/ /index.php?$args;

}

location ~ \.php$ {

try_files $uri =404;

              fastcgi_split_path_info ^(.+\.php)(/.+)$;

               fastcgi_param   SCRIPT_FILENAME $document_root$fastcgi_script_name;

include fastcgi_params;

fastcgi_pass unix:/var/run/php/php7.0-fpm.sock;

}

}
```

Restart nginx:

```
$ sudo service nginx restart
```



	Name	Instance ID
☐	physicalproductsystem.com	i-537b4ecd
☐	mailserver	i-5de87c53



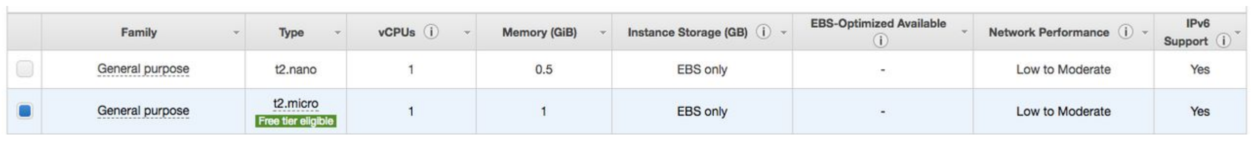
Step 1: Choose an Amazon Machine Image (AMI)

Amazon Linux AMI 2017.03.1 (HVM), SSD Volume Type - ami-4fffc834
Amazon Linux — Free tier eligible
The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.
Root device type: ebs Virtualization type: hvm
Select — 64-bit

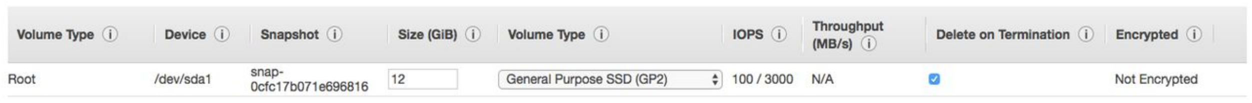
SUSE Linux Enterprise Server 12 SP2 (HVM), SSD Volume Type - ami-8fac8399
SUSE Linux — Free tier eligible
SUSE Linux Enterprise Server 12 Service Pack 2 (HVM), EBS General Purpose (SSD) Volume Type. Public Cloud, Advanced Systems Management, Web and Scripting, and Legacy modules enabled.
Root device type: ebs Virtualization type: hvm
Select — 64-bit

Red Hat Enterprise Linux 7.4 (HVM), SSD Volume Type - ami-c998b6b2
Red Hat — Free tier eligible
Red Hat Enterprise Linux version 7.4 (HVM), EBS General Purpose (SSD) Volume Type
Root device type: ebs Virtualization type: hvm
Select — 64-bit

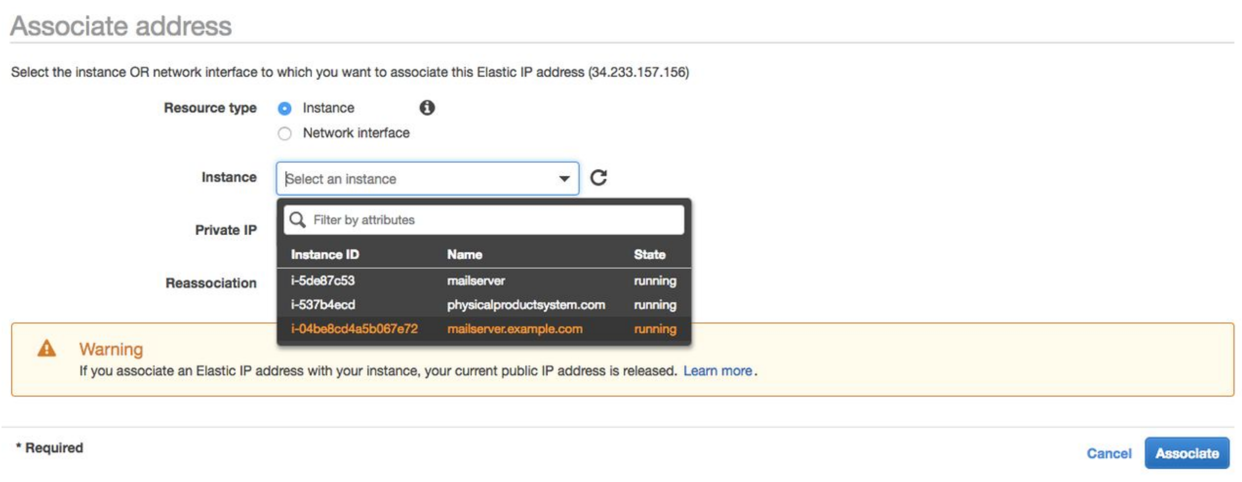
Ubuntu Server 16.04 LTS (HVM), SSD Volume Type - ami-cd0f5cb6
Free tier eligible
Ubuntu Server 16.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).
Root device type: ebs Virtualization type: hvm
Select — 64-bit



	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
☐	General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
☑	General purpose	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes



Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encrypted
Root	/dev/sda1	snap-0cfc17b071e696816	12	General Purpose SSD (GP2)	100 / 3000	N/A	✓	Not Encrypted



Associate address

Select the instance OR network interface to which you want to associate this Elastic IP address (34.233.157.156)

Resource type: [✓ Instance] Network interface

Instance: Select an instance

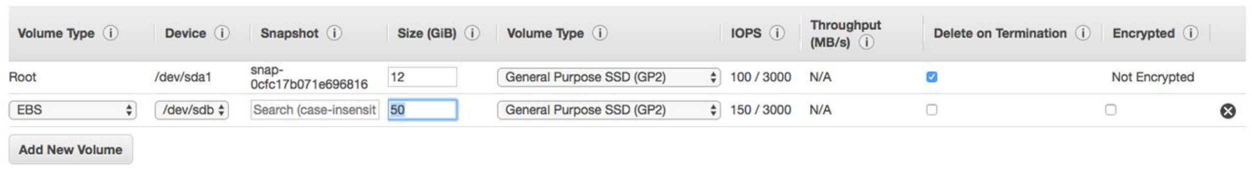
Private IP

Reassociation

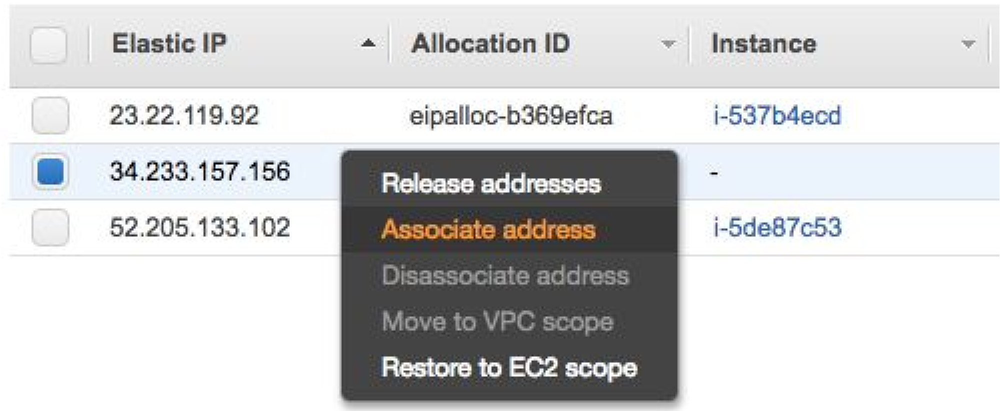
Instance ID	Name	State
i-5de87c53	mailserver	running
i-537b4ecd	physicalproductsystem.com	running
i-04be8cd4a5b067e72	mailserver.example.com	running

Warning
If you associate an Elastic IP address with your instance, your current public IP address is released. Learn more.

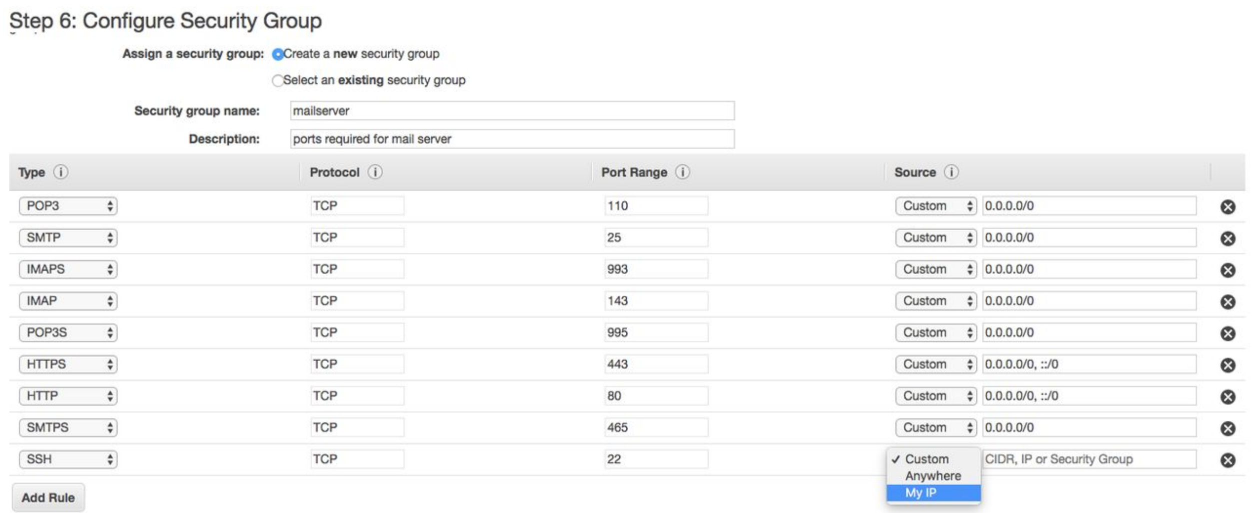
* Required



Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encrypted
Root	/dev/sda1	snap-0cfc17b071e696816	12	General Purpose SSD (GP2)	100 / 3000	N/A	✓	Not Encrypted
EBS	/dev/sdb	Search (case-insensit	50	General Purpose SSD (GP2)	150 / 3000	N/A	☐	☐



Elastic IP	Allocation ID	Instance
23.22.119.92	eipalloc-b369efca	i-537b4ecd
34.233.157.156		-
52.205.133.102		i-5de87c53



Step 6: Configure Security Group

Assign a security group: [✓ Create a **new** security group] Select an **existing** security group

Security group name: mailserver

Description: ports required for mail server

Type	Protocol	Port Range	Source	
POP3	TCP	110	Custom	0.0.0.0/0
SMTP	TCP	25	Custom	0.0.0.0/0
IMAPS	TCP	993	Custom	0.0.0.0/0
IMAP	TCP	143	Custom	0.0.0.0/0
POP3S	TCP	995	Custom	0.0.0.0/0
HTTPS	TCP	443	Custom	0.0.0.0/0, ::/0
HTTP	TCP	80	Custom	0.0.0.0/0, ::/0
SMTPS	TCP	465	Custom	0.0.0.0/0
SSH	TCP	22	✓ Custom	CIDR, IP or Security Group

Conclusion:

Thus we have set up a mail server using Amazon EC2 instance.