ECE 458

Deployment Guide

Lucas Donaldson, Johnny Kumpf, Arthur Schweitzer, Niklas Sjoquist

Prerequisites: Ubuntu 14.04

Deployment Steps:

Install required python, postgresql, apache packages

```
sudo apt-get update
sudo apt-get install python3-pip apache2 libapache2-mod-wsgi-py3
sudo apt-get install libpq-dev
sudo apt-get install postgresql postgresql-contrib
sudo apt-get install git
sudo pip3 install virtualenv

### Now clone the project
git clone https://github.com/lad-47/inventory-project.git

cd inventory-project
cd mysite
virtualenv mysiteenv  ###virtualenv -p python3 mysiteenv if default is python 2 (check python
                      ###--version first)
source mysiteenv/bin/activate

pip install -r requirements.txt
```

Set up database

```
sudo -i -u postgres
createdb inventory
### Switch back to main user (bitnami for Duke VMs)
su - bitnami

sudo nano /etc/postgresql/9.3/main/pg_hba.conf

### Change the following lines
        # TYPE DATABASE USER ADDRESS METHOD
        local  all      all            peer
        local  all      postgres   peer
```

```
### From peer to trust

sudo service postgresql restart

cd inventory-project
cd mysite
source mysiteenv/bin/activate
python manage.py makemigrations
python manage.py migrate

python manage.py createsuperuser
python manage.py collectstatic
```

Set up Apache server

```
sudo nano /etc/apache2/sites-available/000-default.conf

### Add the following
```

```
WSGIDaemonProcess myproject python-path=/home/user/myproject
python-home=/home/user/myproject/myprojectenv
<VirtualHost *:80>
. . .

Alias /static /home/user/myproject/static
<Directory /home/user/myproject/static>
        Require all granted
</Directory>

<Directory /home/user/myproject/myproject>
        <Files wsgi.py>
                Require all granted
        </Files>
</Directory>


WSGIProcessGroup myproject
WSGIScriptAlias / /home/user/myproject/myproject/wsgi.py
WSGIPassAuthorization On

</VirtualHost>
```

```
sudo service apache2 restart
```

Set up SSL certificate using Let’s Encrypt

```
cd ~/
wget https://dl.eff.org/certbot-auto
chmod a+x certbot-auto

### Comment out line WSGIDaemonProcess in .conf

./certbot-auto --apache -d example.com

### Uncomment WSGIDaemonProcess line
sudo service apache2 restart
```

Initialize Database with backed up data

Step 1: Configure SSH Connectivity Between Main Development Server and Backup Server

First, we need to set the SSH connectivity from the main server to the backup server. Thus, in the **main development server shell**, switch to the user postgres, and then generate a SSH key-pair, by running:

```
sudo -su postgres
ssh-keygen -t rsa
```

Accept the default location by clicking ENTER and then set the key-pair’s passphrase to be empty by clicking ENTER two more times. This will now save the generated keys in a .ssh file within the postgres user’s home directory.

Extract the generated keys from the .ssh files, using the following commands:

```
cat ~/.ssh/bin/id_rsa.pub
```

In the **Backup server shell**, switch the user to be ‘barman’, by running:

```
sudo -su barman
```

Now, copy the key contents generated from the Main development server to a new .ssh directory stored on the Backup server, by running:

```
mkdir -p ~/.ssh
chmod 700 ~/.ssh
echo "public_key_string" >> ~/.ssh/authorized_keys
chmod 600 ~/.ssh/authorized_keys
```

***NOTE:** “public_key_string” is the extracted key from the Main development server’s .ssh files (result of the previous ‘cat’ command)

To test the connection between the two servers, **switch** back to the **Main development server shell** and run the following command:

```
ssh barman@barman-backup-server-ip
```

If successful, type ‘exit’ and disconnect from the backup server.

Step 2: Configure SSH Connectivity Between Backup Server and Main Development Server

REPEAT the procedure from Step 4, this time establishing an SSH connection from the backup server to the main development server.

Be sure that when checking the connection between the two servers, **switch** back to the **Backup server shell** and run the following command:

```
ssh postgres@main-db-server-ip
```

***NOTE:** main-db-server-ip is the INET IP address for the main development server:

If successful, type ‘exit’ and disconnect from the main development server.

Step 3: Configuring Main Development Server for Backups

In the **Backup server shell**, switch to the user ‘barman’ and find the incoming backup directory, by running:

```
sudo -su barman
barman show-server main-db-server | grep incoming_wals_directory
```

Take note of the **incoming_wals_directory**. Be sure to save this filepath somewhere, so that it can be referenced later.

In the **Main development server shell**, switch to the user ‘postgres’ and open the PostgreSQL configuration file by running:

```
sudo -su postgres
nano $PGDATA/postgresql.conf
```

Add the following settings to the PostgreSQL configuration file:

```
wal_level = archive                    # minimal, archive, hot_standby, or logical

. . .

archive_mode = on               # allows archiving to be done

. . .

archive_command = 'rsync -a %p
barman@barman-backup-server-ip:incoming_wals_directory/%f'              # command to use to
```

```
archive a logfile segment
```

***NOTE:** Here, barman-backup-server-ip is the INET IP address of the backup server and the ‘incoming_wals_directory’ is the saved filepath of the incoming backup directory.

On the **Main development server shell**, switch back to the sudo user and restart the DB, by running:

```
sudo service postgresql restart
```

Step 4: Restoring From a Backup

In the **Main development server shell,** switch to the sudo user and stop the PG service by running the following:

```
sudo service postgresql stop
```

Switch to the **Backup server shell** and locate the details of the latest backup, by running:

```
barman show-backup main-db-server latest
```

The output should look something like this:

```
Backup 20160114T173552:
 Server Name          : main-db-server
 Status               : DONE
 PostgreSQL Version     : 90405
 PGDATA directory       : /var/lib/pgsql/9.4/data

 Base backup information:

. . .

   Begin time           : 2016-01-14 17:35:53.164222-05:00
   End time             : 2016-01-14 17:35:55.054673-05:00
```

Next, run the following command, to restore the backup from the Backup server to the Main development server:

```
barman recover --target-time "Begin time"  --remote-ssh-command "ssh
postgres@main-db-server-ip"   main-db-server   backup-id   /var/lib/postgresql/9.3/main
```

***NOTE:** ‘Main-db-server-ip’ is the INET IP address of the main development server. Replace ‘Begin Time’ and ‘backup_id’ with the values from the previous output. In the case that you are interested in restoring from the last backup, you can replace the ‘backup_id’ with the string ‘latest’.

On the **Main development server shell,** switch the to the sudo user and start the PG service, by running:

```
sudo service postgresql start
```

Switch the user to be ‘postgres’ and run the following commands to check if restoration was successful:

```
sudo su - postgres
psql ‘db_name’
```

***NOTE:** In this case, the ‘db_name’ is the name given to the DB when set up on the main development server.

Run the following commands to find the DB table contents:

```
db_name=# \dt
```

Finally, compare the newly restored DB tables to the latest tables within the Backup server’s DB. If these are the same, then you have successfully restored the latest backup data to the main development server’s database.

References:

https://www.digitalocean.com/community/tutorials/how-to-serve-django-applications-with-apache-and-mod_wsgi-on-ubuntu-14-04

https://www.digitalocean.com/community/tutorials/how-to-install-and-use-postgresql-on-ubuntu-14-04

https://www.digitalocean.com/community/tutorials/how-to-secure-apache-with-let-s-encrypt-on-ubuntu-14-04