Deploy Django

Ubuntu 21.04 - Apache - Django - MySQL

Learn Full Stack Development

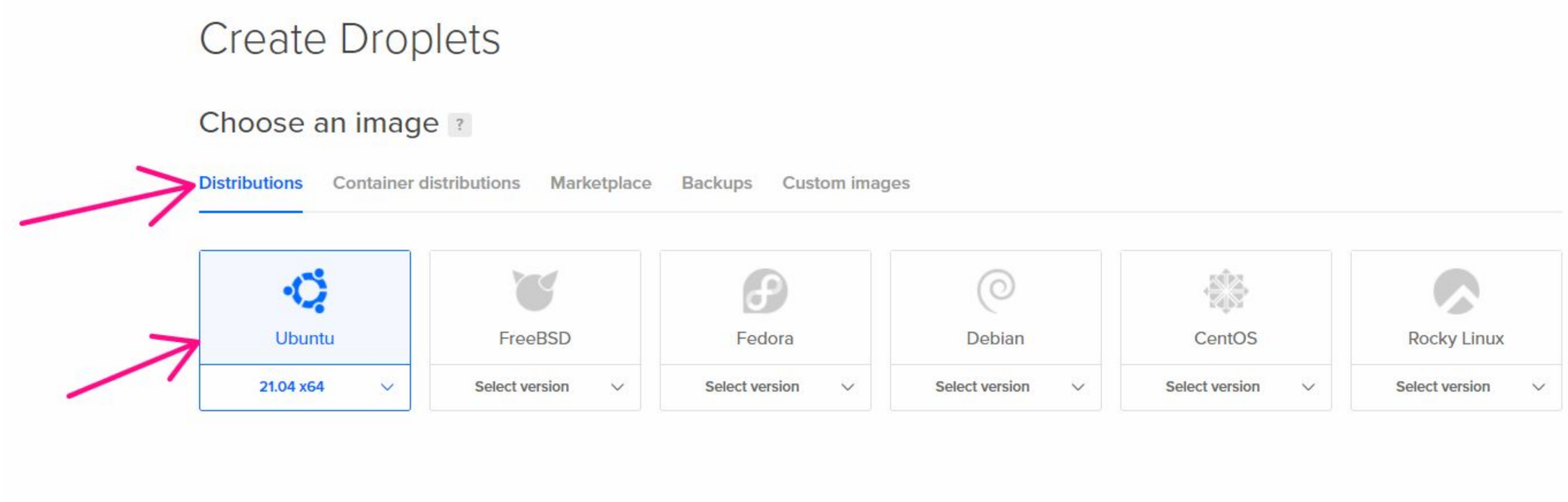
www.projectfullstack.com

Step 1: Setup Server

- Using Digital Ocean as a VPS provider:
  - Use this link (also in description) for $100 in Digital Ocean credits
    - https://m.do.co/c/e30460a6fcfc
  - Register / Login
  - Create a new droplet
    - Distribution: 21.04
    - Size: $5/month
      - 1 GB RAM
      - 1 CPU
      - 25GB SSD Disk
    - Datacenter: The closest one to you
    - Hostname: An easily identifiable name

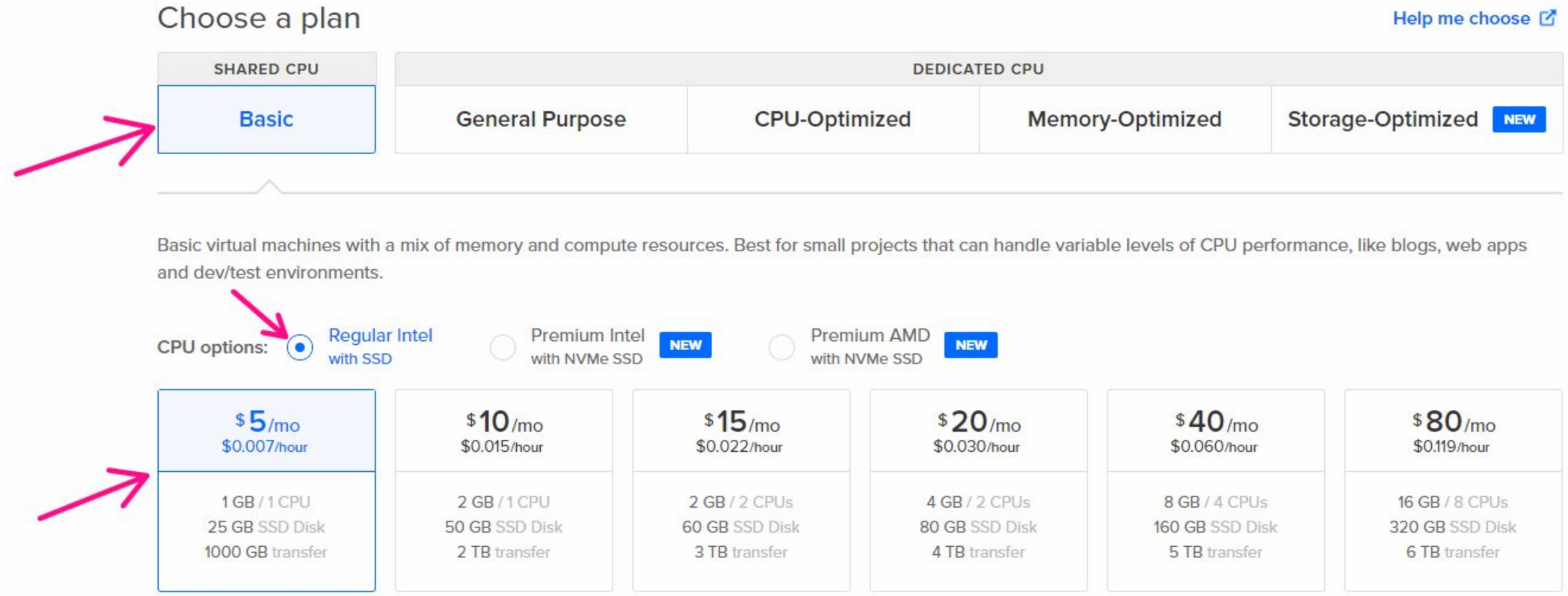
Step 1: Setup Server

- Distribution



Step 1: Setup Server

- Choose A Plan

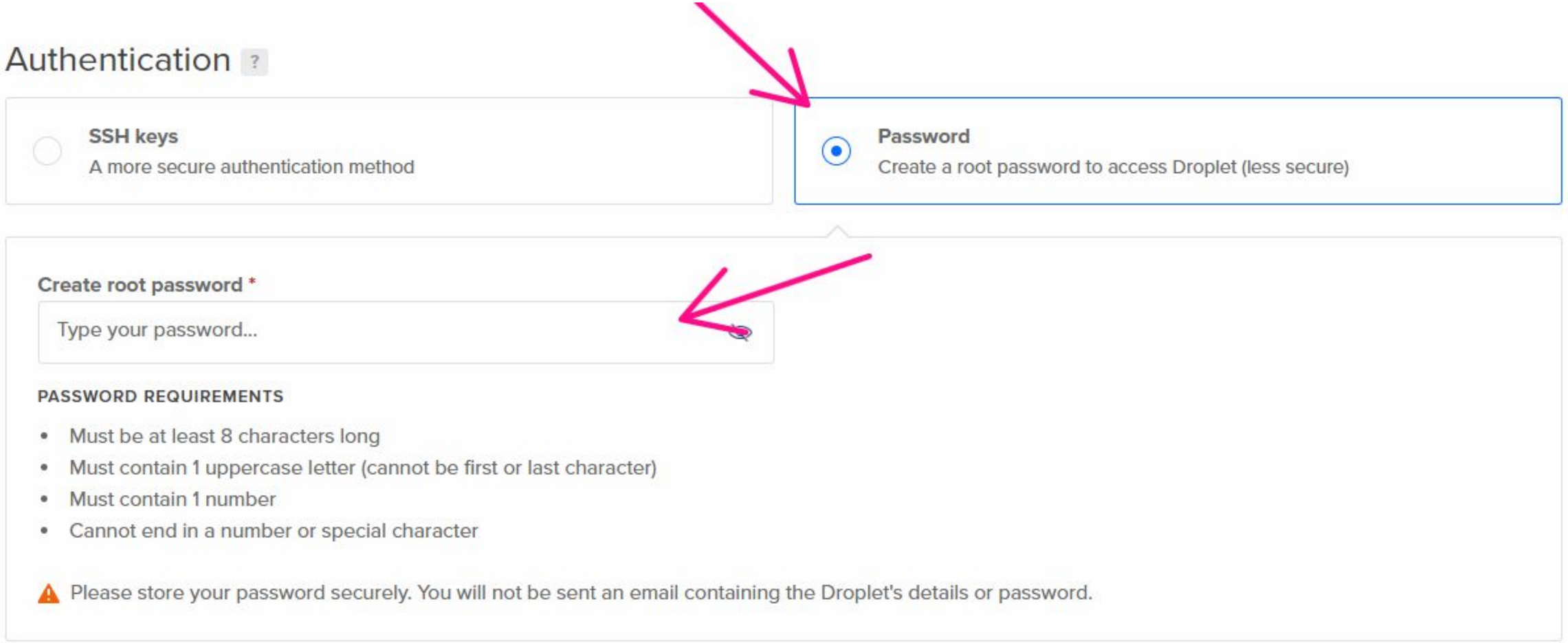


Step 1: Setup Server

- Datacenter

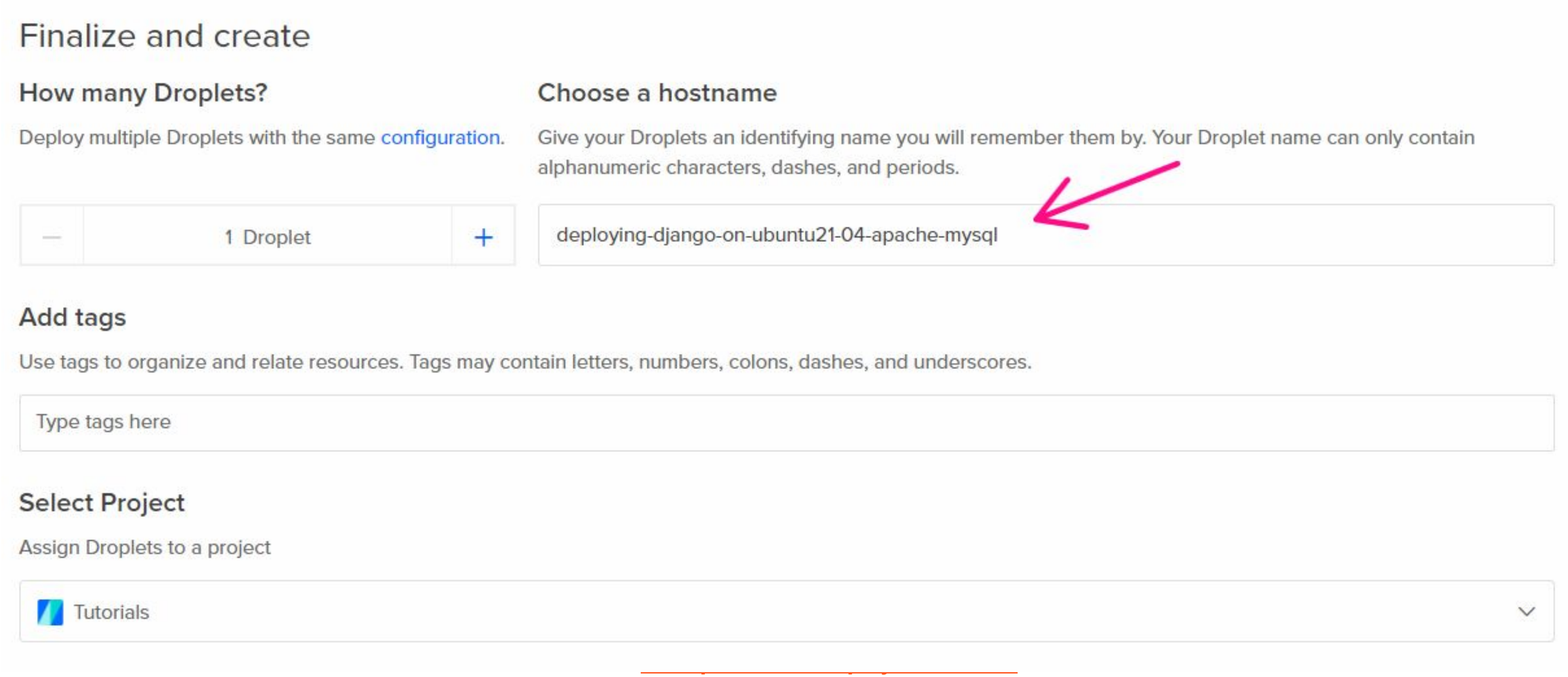
Step 1: Setup Server

- Authentication - NOTE: I will use a root password for this tutorial. I HIGHLY recommend you use SSH keys for your production level deployments



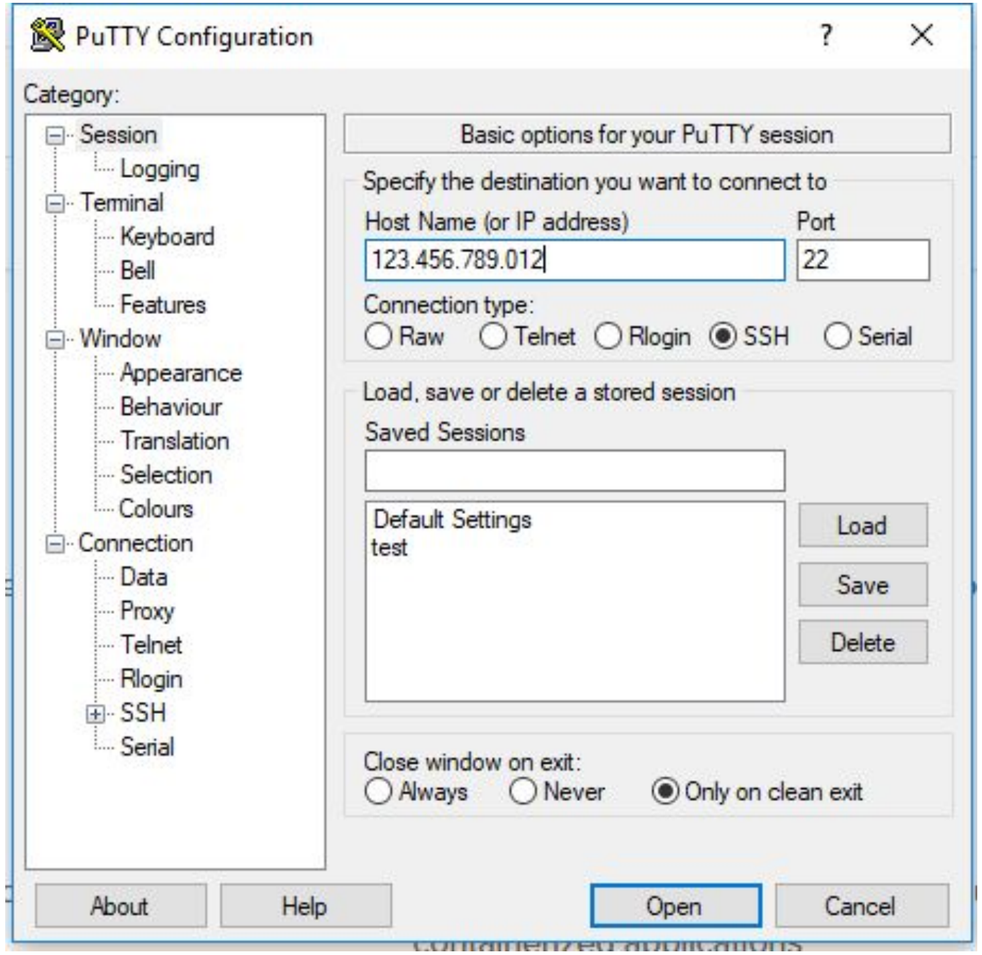
Step 1: Setup Server

- Finalize and Create- Make sure to use a descriptive host name. This will make it easier in the future if you have several droplets



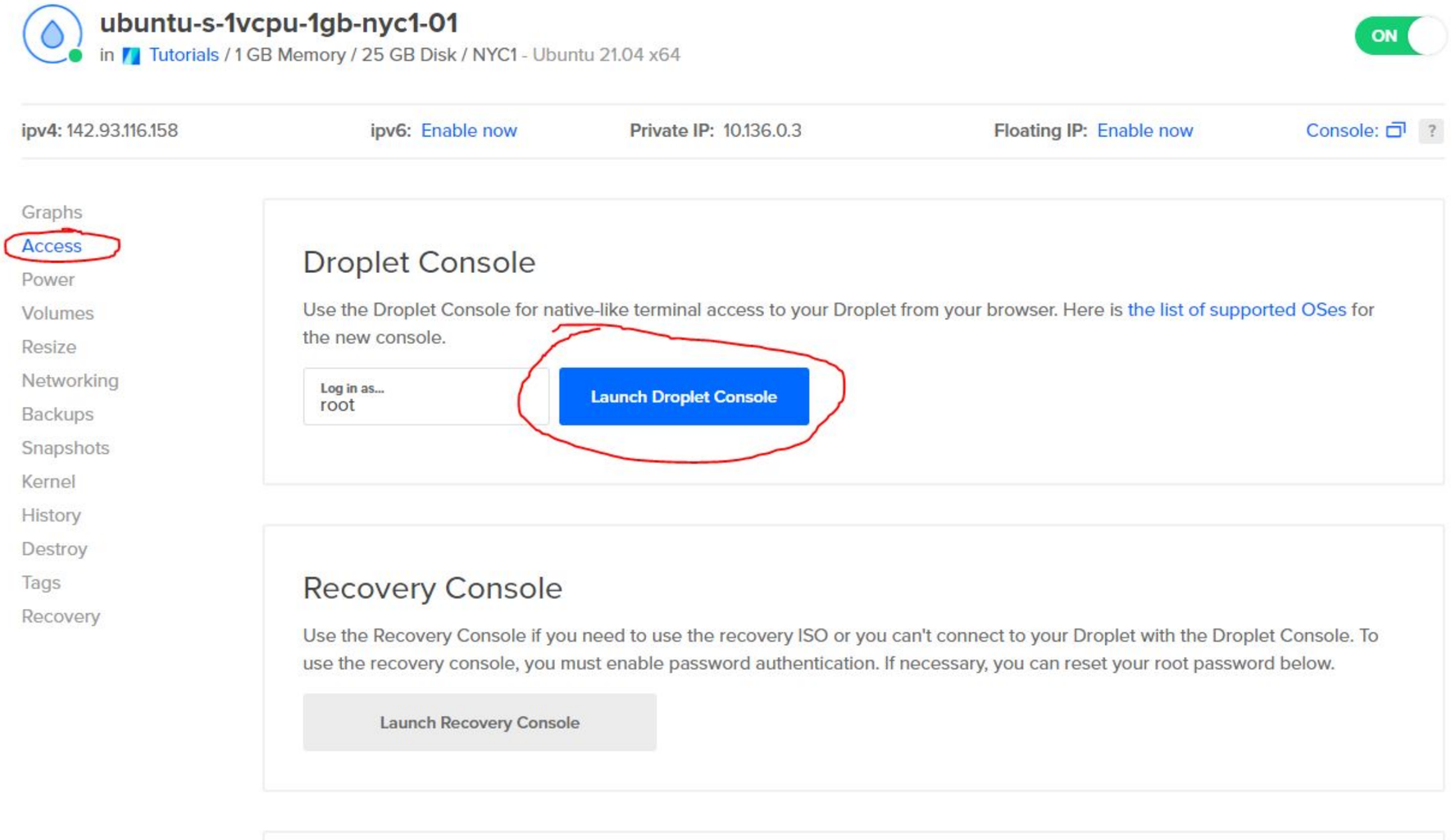
Step 2: Connect To Server

- Using Linux / Mac open a terminal:
  - SSH
    - $ ssh root@<your_ip>
    - Example: ssh root@123.456.789.012
    - Select yes to add the server to your known hosts
    - Enter password
- Using Windows:
  - PuTTY
    - Select Session from the left sidebar
    - Enter your ip address in the Host Name input
    - Click the “Open” button
    - Select “Accept” at the popup
    - Enter “root” in the login input
    - Enter password



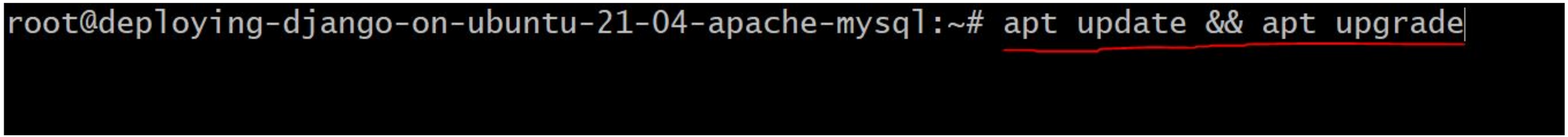
Step 2: Connect To Server

- Using Digital Ocean dashboard
  - In digital ocean, select your droplet and go to the “access” tab, then select Launch Droplet Console



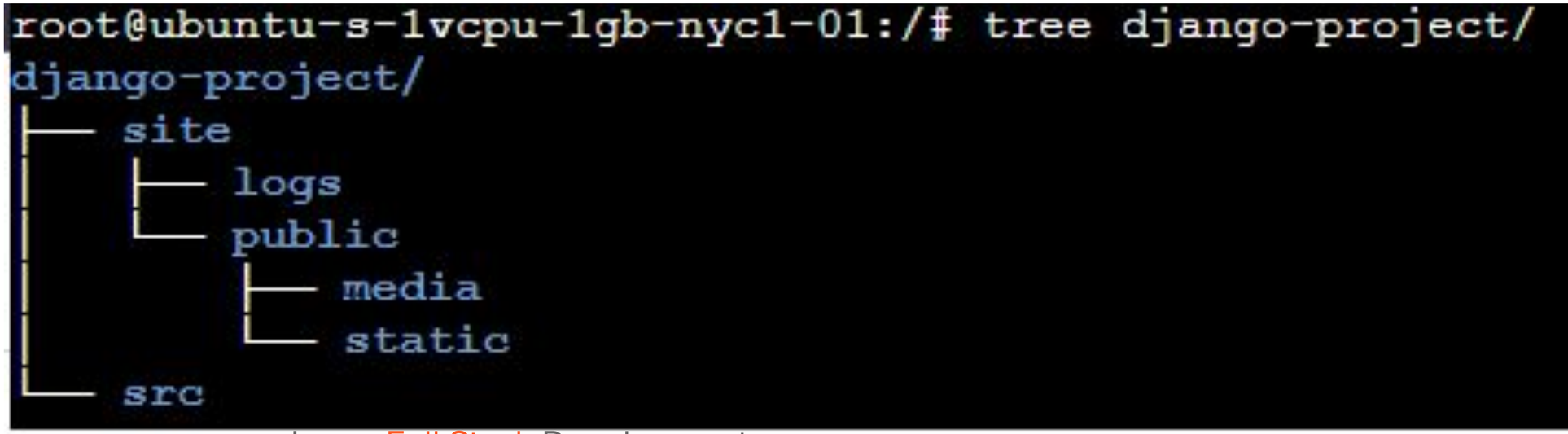
Step 3: Apply Software Updates

- Update the system to make sure we have the most up to date software
- $ apt update && apt upgrade
- $ sudo reboot ← reboots the system, optional but recommended



Step 4: Create directory structure

- CD to the root of your system
  - $ cd /
- Create the following directory structure
  - Project-name (I will use “django-project” for this)
    - src/
    - site/
      - logs/
      - public/
        - media/
        - static/



Step 5: Install PIP & Setup VirtualEnv

- PIP is a package manager we use to install python packages
  - Django is a python package!
- Virtualenv is used to manage Python packages for different projects.
  - Using virtualenv allows you to avoid installing Python packages globally which could break system tools or other projects. You can install virtualenv using pip.
- $ sudo apt install python3-pip
- $ sudo pip3 install virtualenv
- Create a virtualenv
  - Make sure you are in /project-name/ directory
  - $ virtualenv venv -p python3
  - $ source venv/bin/activate
  - $ pip install django==3.2.8

Step 6: Create Django Project

- Make sure your venv is activated and django is installed
  - $ pip freeze
- $ cd /project-name/src
- $ django-admin startproject some-project-name . ← don't forget the period at the end!
- Add your server's IP address to settings.py Allowed Host constant
- Run the django development server
  - python manage.py runserver 0.0.0.0:8000
- Open a web browser, goto <your-ip-address>:8000
  - Example: 123.456.789.010:8000
- You should see the default django welcome page

Step 7: Install MySQL

- $ sudo apt install mysql-server
- $ sudo mysql_secure_installation
  - Yes to everything!
- Creating a user and a database
  - $ mysql
  - mysql> CREATE USER ‘djangouser’@’localhost’ IDENTIFIED BY ‘XLq4zhNFPjfwyHgX’;
  - mysql> CREATE DATABASE ‘djangoproject’;
  - mysql> GRANT ALL PRIVILEGES ON ‘djangoproject’.* to ‘djangouser’@’localhost’;
  - mysql> FLUSH PRIVILEGES;

Step 8: Connect MySQL and Django

- Install “mysqlclient” python package
  - $ sudo apt install python3-dev
  - $ sudo apt install libmysqlclient-dev
  - $ pip install mysqlclient
- Add the following to settings.py

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        ‘NAME’: ‘tutorial’,
        ‘USER’: ‘djangouser’,
        ‘PASSWORD’:  ‘your-password’,
        ‘HOST’: ‘localhost, # default is localhost, so this can be omitted
        ‘PORT’: ‘3306’ # default is 3306, so this can be omitted
    }
```

Step 8: Connect MySQL and Django

- Check django, create superuser, make migrations, runserver
  - $ python manage.py check
  - $ python manage.py migrate
  - $ python manage.py createsuperuser
  - $ python manage.py runserver 0.0.0.0:8000

In a browser go to your-ip-address:8000/admin and log in with the superuser credentials

Step 9: Install & Configure Apache2

- $ sudo apt install apache2 libapache2-mod-wsgi-py3
- Check apache installation
  - In a browser go to your-ip-address
  - You should see the Apache2 default page
- CD to /etc/apache2/sites-available/
  - Change 000-default.conf to the following:
    - See next slide

Step 9: Install & Configure Apache2

- CD to /etc/apache2/sites-available/
  - Change 000-default.conf to the following:

```
<VirtualHost *:80>

    ErrorLog /django-project/site/logs/error.log
    CustomLog /django-project/site/access.log combine

  alias /static /django-project/site/public/static
  <Directory /django-project/site/public/static>
        Require all granted
    </Directory>

  <Directory /django-project/src/tutorial>
        <Files wsgi.py>
            Require all granted
        </Files>
    </Directory>

    WSGIDaemonProcess tutorial python-home=/django-project/venv python-path=django-project/src/
    WSGIProcessGroup tutorial
    WSGIScriptAlias / /django-project/src/tutorial/wsgi.py

</VirtualHost>
```

Step 9: Install & Configure Apache2

- Make sure your syntax is correct for the 000-default.conf
  - $ sudo apachectl configtest
- Restart Apache2 for changes to take effect
  - $ sudo service apache2 restart
- In a browser visit your ip address

Step 10: Configuring Static Files

- In a browser, visit your-ip-address/admin
  - Notice the static files (css/javascript/etc) are not loading
- First we need a place for static files to live on our server
  - $ mkdir /project_name/site/public/static
- Add the following to settings.py:
  - STATIC_ROOT = ‘project_name/site/public/static
- Now that django knows where to put static files, we have to tell it to do so:
  - python manage.py collectstatic
- Finally we have to tell Apache to look in this directory for static files
  - Add the following to 000-default.conf
  - alias /static /project_name/site/public/static
    <Directory /project_name/site/public/static>
    Require all granted
    </Directory>
- In a browser, visit your-ip-address/admin. You should now have static files being served