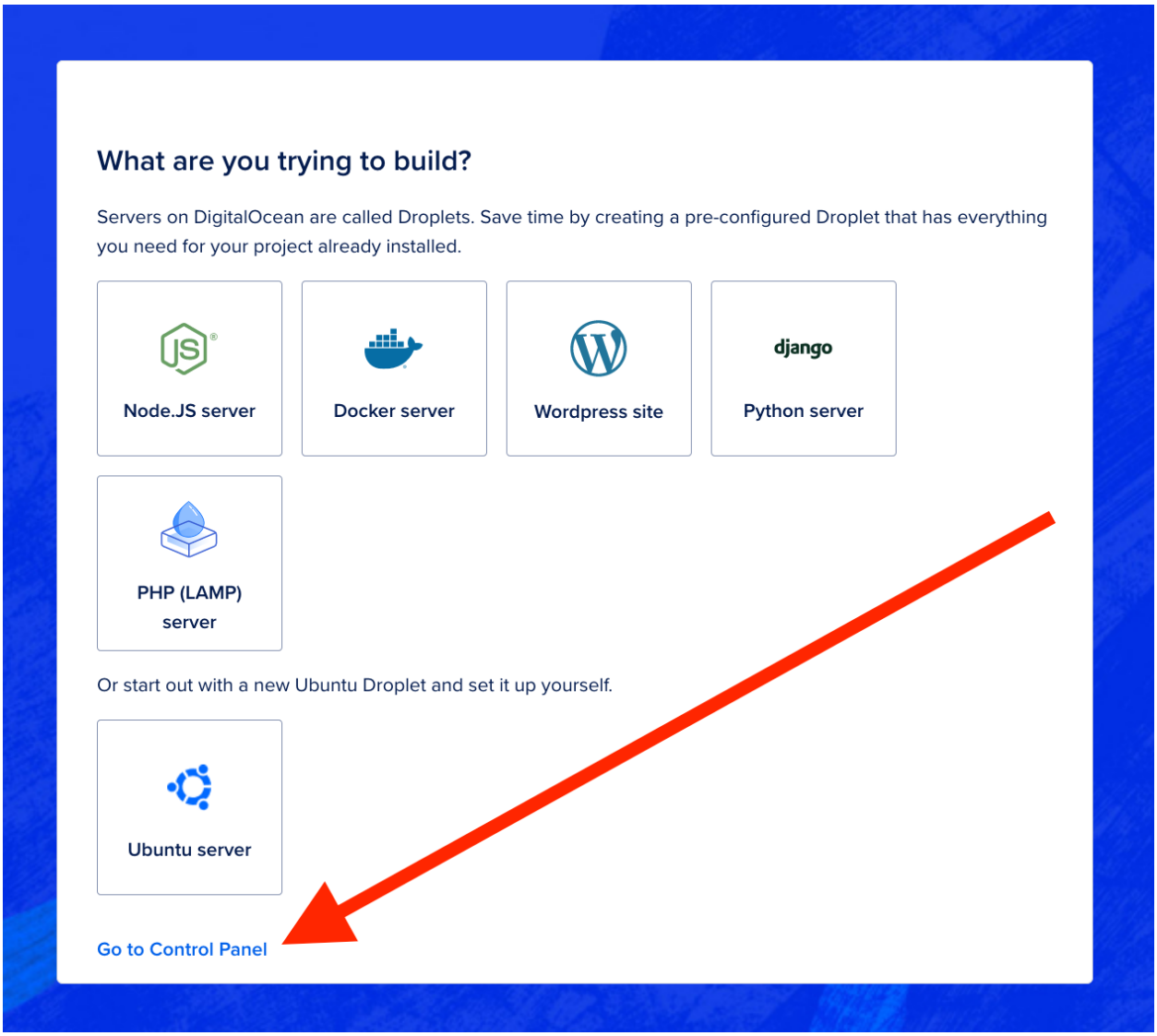
https://www.digitalocean.com/community/tutorials/how-to-set-up-an-ubuntu-20-04-server-on-a-digitalocean-droplet
https://learnwagtail.com/launch-your-wagtail-website-digital-ocean-ubuntu-18/



YOUTUBE Videos by Chris Bocchino

Street Card MobaXterm Setup: https://youtu.be/hpsf-SXKf_k

Street Card SSH Connections: https://youtu.be/bzkw-8dA8Gw

Choose an image

Distributions | Container distributions | Marketplace | Custom images

Ubuntu	FreeBSD	Fedora	Debian	CentOS
20.04 (LTS) x64	Select version	Select version	Select version	Select version

Choose a plan

Help me choose

SHARED CPU	DEDICATED CPU		
Basic	General Purpose	CPU-Optimized	Memory-Optimized

Basic virtual machines with a mix of memory and compute resources. Best for small projects that can handle variable levels of CPU performance, like blogs, web apps and dev/test environments.

$5/mo $0.007/hour	$10/mo $0.015/hour	$15/mo $0.022/hour	$20/mo $0.030/hour	$40/mo $0.060/hour	$80/mo $0.119/hour
1 GB / 1 CPU 25 GB SSD Disk 1000 GB transfer	2 GB / 1 CPU 50 GB SSD Disk 2 TB transfer	2 GB / 2 CPUs 60 GB SSD Disk 3 TB transfer	4 GB / 2 CPUs 80 GB SSD Disk 4 TB transfer	8 GB / 4 CPUs 160 GB SSD Disk 5 TB transfer	16 GB / 8 CPUs 320 GB SSD Disk 6 TB transfer

Our Basic Droplet plans, formerly called Standard Droplet plans, range from 1 GB of RAM to 16 GB of RAM. General Purpose Droplets have more overall resources and are best for production environment, and Memory-Optimized Droplets have more RAM and disk options for RAM intensive applications.

Each Droplet adds more data transfer to your account.

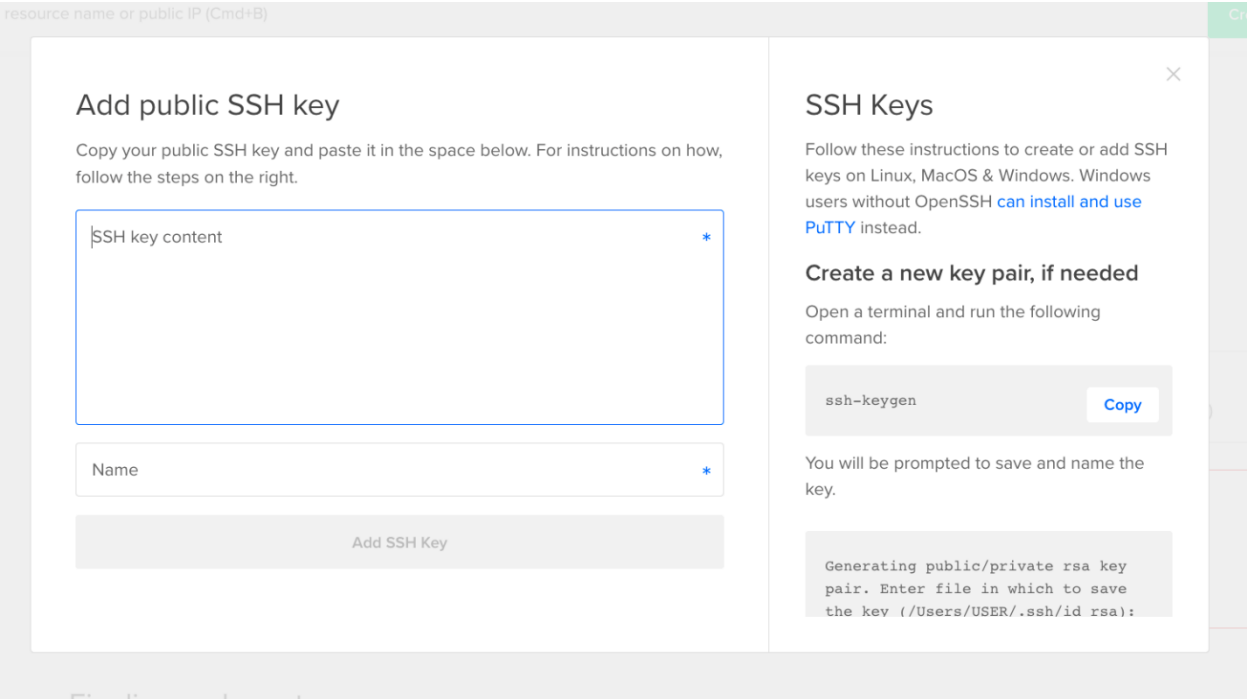
Authentication

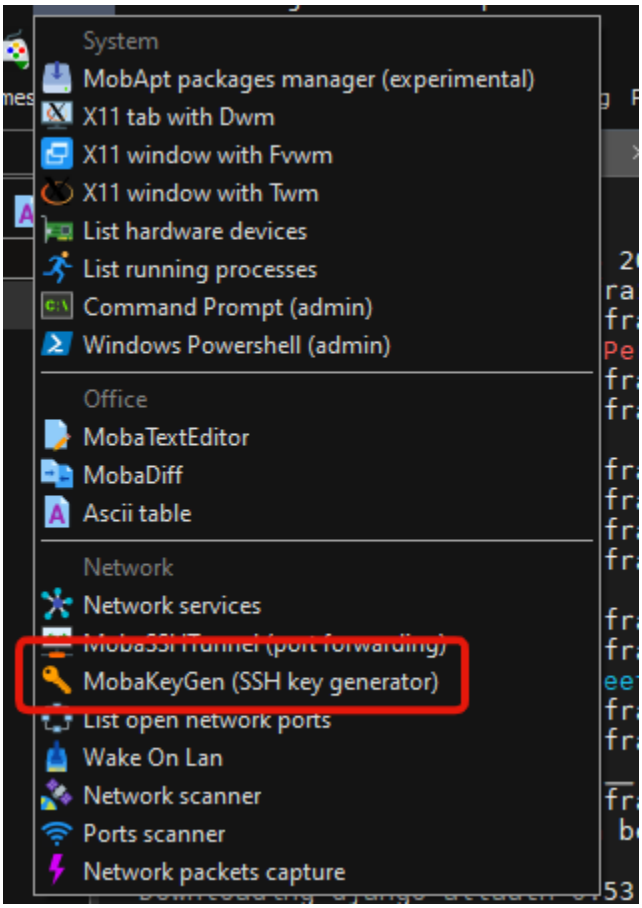
SSH keys
A more secure authentication method

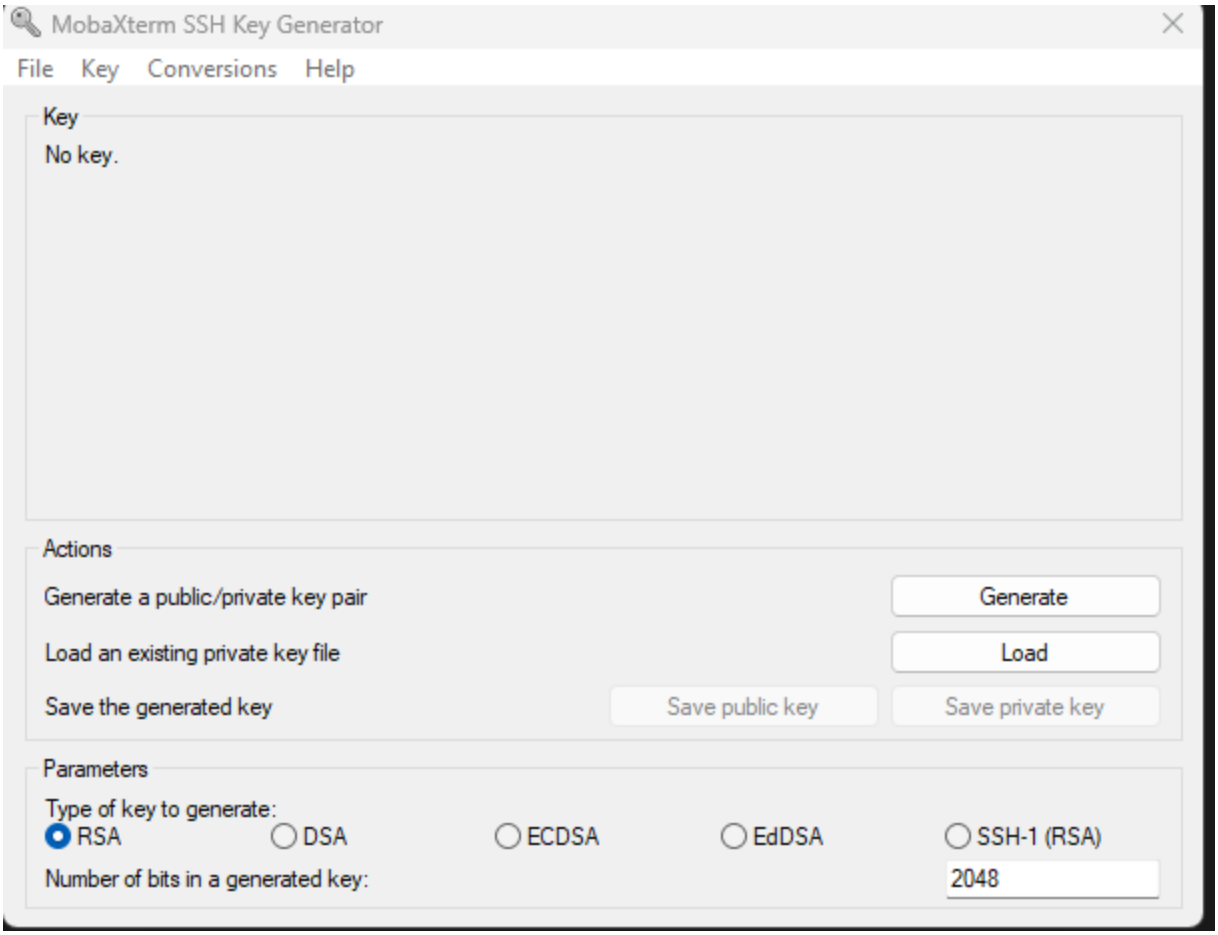
Password
Create a root password to access Droplet (less secure)

Choose your SSH keys Select at least one key.

New SSH Key


resource name or public IP (Cmd+B)
Add public SSH key
Copy your public SSH key and paste it in the space below. For instructions on how, follow the steps on the right.
SSH key content *
Name *
Add SSH Key
SSH Keys
Follow these instructions to create or add SSH keys on Linux, MacOS & Windows. Windows users without OpenSSH can install and use PuTTY instead.
Create a new key pair, if needed
Open a terminal and run the following command:
ssh-keygen
Copy
You will be prompted to save and name the key.
Generating public/private rsa key pair. Enter file in which to save the key (/Users/USER/.ssh/id_rsa):


System
MobApt packages manager (experimental)
X11 tab with Dwm
X11 window with Fvwm
X11 window with Twm
List hardware devices
List running processes
Command Prompt (admin)
Windows Powershell (admin)
Office
MobaTextEditor
MobaDiff
Ascii table
Network
Network services
MobaKeyGen (SSH key generator)
List open network ports
Wake On Lan
Network scanner
Ports scanner
Network packets capture



Generate a Public and Private key and passphrase “RSA” 2048 bits. Save both. Paste the public key to above digital ocean account box.

Finalize and create

How many Droplets?

Deploy multiple Droplets with the same configuration.

1 Droplet

Choose a hostname

Give your Droplets an identifying name you will remember them by. Your Droplet name can only contain alphanumeric characters, dashes, and periods.

ubuntu-s-1vcpu-1gb-nyc1-01

Add tags

Use tags to organize and relate resources. Tags may contain letters, numbers, colons, dashes, and underscores.

Type tags here

Select Project

Assign Droplets to a project

erin

Add backups

Enable backups RECOMMENDED

A system-level backup is taken once a week, and each backup is retained for 4 weeks.

$1.00/mo (per Droplet)
20% of the Droplet price

Create Droplet

Resources Activity Settings

DROPLETS (1)

ubuntu-s-1vcpu-1gb-nyc1-01 167.172.146.33

Create something new

Create a Managed Database
Worry-free database management

Start using Spaces
Deliver data with scalable object storage

Spin up a Load Balancer
Distribute traffic between multiple Droplets

Build on what you have

Build a Node.js application
Set up a Node.js application for production on Ubuntu

Go containerized with Docker
Install and use Docker containers to build containerized applications

Build a PHP web application
Install Linux, Apache, MySQL, PHP (LAMP) on DigitalOcean 1-Click App

Learn more

Product Docs
Technical overviews, how-tos, release notes, and support material

Tutorials
DevOps and development guidelines

API Docs
Run your resources programmatically

Ask a question
Connect, share and learn

- **Create a new Ubuntu 18 Droplet**
- **You'll need your Droplet IP address**
  `ssh root@167.172.xxx.xx`
- **Create a new user on your new ubuntu server**
  `adduser newuser`
- **Give your new user admin privileges**
  `usermod -aG sudo newuser`
- **Make sure OpenSSH is enabled**

  ```
  ufw app list
  ufw allow OpenSSH
  ufw enable
  ufw status
  ```

- **Copy your root SSH Key to the new Ubuntu user account**
  `rsync --archive --chown=newuser:newuser ~/.ssh /home/newuser`
- **SSH into your server as your new Ubuntu user (don't use root)**
  exit your ssh session with `ctrl + d`
  `ssh newuser@167.172.xxx.xx`
- **Update Ubuntu**
  `sudo apt update`
  `sudo apt install python3-pip python3-dev libpq-dev postgresql postgresql-contrib nginx curl`
- **Log into a postgres session**
  `sudo -u postgres psql`
- **Create a new database**
  `CREATE DATABASE your_db_name;`
- **Create a new postres user with a password**
  `CREATE USER your_db_user WITH PASSWORD 'your_db_password';`
- **Alter the postgres role**

  ```
  ALTER ROLE your_db_user SET client_encoding TO 'utf8';
  ALTER ROLE your_db_user SET default_transaction_isolation TO 'read committed';
  ALTER ROLE your_db_user SET timezone TO 'UTC';
  ```

- **Make the postgres user an admin**
  `GRANT ALL PRIVILEGES ON DATABASE your_db_name TO your_db_user;`
- **Quit postgres**
  `\q`
- **Upgrade pip and install virtualenv**
  `sudo -H pip3 install --upgrade pip && sudo -H pip3 install virtualenv`
- **Create a new project directory**
  `mkdir ~/yourprojectname && cd ~/yourprojectname`
- **Clone your project from github into this directory**
  `git clone https://github.com/your-repo-url/ .`
- **Create a new virtualenv**
  `virtualenv .venv`
- **Activate your virtualenv**
  `source .venv/bin/activate`
- **Install gunicorn and psycopg2-binary**
  `pip install gunicorn psycopg2-binary`
- **Install your project requirements**
  `pip install -r requirements.txt`

- **Collect static with:**
  `python manage.py collectstatic --settings=yourprojectname.settings.production`
- **Re-run your server with**
  `python manage.py runserver 0.0.0.0:8000 --settings=yourprojectname.settings.production`
  At this point you should see migrations are required.
- **Cancel your server and run it normally with**
  Set the DJANGO SETTINGS MODULE with:
  `export DJANGO_SETTINGS_MODULE='yourprojectname.settings.production'`
  Re run the server and we no longer need to specify a settings file
- **Apply migrations**
  `python manage.py migrate`
- **Create a new superuser**
  `python manage.py createsuperuser`
- **Allow port 8000 through ufw**
  `sudo ufw allow 8000`
- **Run the server on port 8000 and preview it**
  `python manage.py runserver 0.0.0.0:8000`
  Go to http://167.172.xxx.xx:8000/ and you'll see it at least loads. It'll look terrible, but it works!
  The site now only work on port 8000. That's no good. We need it to run all the time.
- **Make sure your in your main directory**
  `cd ~/yourprojectname`
- **Run gunicorn on port 8000**
  `gunicorn --bind 0.0.0.0:8000 yourprojectname.wsgi`
- **Preview your site on port 8000 again, but notice this time we are running it with gunicorn**
  Go to http://167.172.xxx.xx:8000/ and you'll it loads.
- **Cancel gunicorn and deactivate your virtualenv**
  `ctrl + c` and `deactivate`
- **Create a gunicorn socket file**
  `sudo nano /etc/systemd/system/gunicorn.socket`

Add this to it:

```
[Unit]
Description=gunicorn socket

[Socket]
ListenStream=/run/gunicorn.sock

[Install]
WantedBy=sockets.target
```

- **Create a systemd file for gunicorn with sudo privileges**

  ```
  sudo nano /etc/systemd/system/gunicorn.service
  ```

  And add this into it:

  ```
  [Unit]
  Description=gunicorn daemon
  Requires=gunicorn.socket
  After=network.target

  [Service]
  User=newuser
  Group=www-data
  WorkingDirectory=/home/newuser/yourprojectname
  ExecStart=/home/newuser/yourprojectname/.venv/bin/gunicorn \
          --access-logfile - \
          --workers 3 \
          --bind unix:/run/gunicorn.sock \
          yourprojectname.wsgi:application

  [Install]
  WantedBy=multi-user.target
  ```

- **Start and enable the gunicorn**

  ```
  sudo systemctl start gunicorn.socket && sudo systemctl enable gunicorn.socket
  ```

- **Check the status of the process with:**

  ```
  sudo systemctl status gunicorn.socket
  ```

  Should say active listening

- **Check the existence of the new socket file**

  ```
  file /run/gunicorn.sock
  ```

- **Check the gunicorn status with**

  ```
  sudo systemctl status gunicorn
  ```

  You should see INACTIVE DEAD

- **Test the socket activation with a curl command**

  ```
  curl --unix-socket /run/gunicorn.sock localhost
  ```

  You should see the html output of your site.
  If you didnt, something is wrong with gunicorn. Double check your wsgi.py file, double check the gunicorn paths.

- **At this point it doesnt hurt to restart gunicorn with**

  ```
  sudo systemctl daemon-reload && sudo systemctl restart gunicorn
  ```

- **Create a new server block in nginx**
  `sudo nano /etc/nginx/sites-available/yourprojectname` And add this:

```
server {
    listen      80;
    listen      [::]:80;
    server_name 167.172.xxx.xx;
    charset     UTF-8;

    error_log   /home/newuser/yourprojectname/nginx-error.log;
    location = /favicon.ico { access_log off; log_not_found off; }
    location /static/ {
        alias /home/newuser/yourprojectname/static/;
    }

    location /media/ {
        alias /home/newuser/yourprojectname/media/;
    }

    location / {
        include proxy_params;
        proxy_pass http://unix:/run/gunicorn.sock;
    }
}
```

- **Create a file by linking it to the sites-enabled directory**
  `sudo ln -s /etc/nginx/sites-available/yourprojectname /etc/nginx/sites-enabled`
- **Test nginx with:**
  `sudo nginx -t`
- **If there were no errors, restart nginx**
  `sudo systemctl restart nginx`
- **Open the firewall to normal traffic with Nginx, and delete port 8000**
  `sudo ufw delete allow 8000 && sudo ufw allow 'Nginx Full'`
  If NGINX shows the welcome to nginx page, double check your server_name ip in your nginx config file (the one we created earlier).
  Add your IP to your domain DNS.
- **When launching your website update your nginx settings**
  `sudo nano /etc/nginx/sites-available/yourprojectname` Replace 167.172.xxx.xx with yourwebsite.com
- **Test nginx settings with**
  `sudo nginx -t`
- **Restart nginx with**
  `sudo systemctl restart nginx`
- **Add your new domain to your allowed hosts**
  `sudo nano yourprojectname/settings/production.py` Add yourwebsite.com to the `ALLOWED_HOSTS` and Remove the ip address of 167.172.xxx.xx
- Add 167.172.xxx.xx to your domain DNS settings and wait for it to propogate
- View your new website at yourwebsite.com
- **Update the wagtail site settings**
  Go to http://yourwebsite.com/admin/sites/2/ and: change localhost to yourwebsite.com