01 - Creating a Droplet on Digital Ocean

A Digital Ocean (DO) droplet is a virtual private server (VPS). Because you have access to the virtual machine's (VM) OS, a droplet is an example of Infrastructure-as-a-Service (IaaS).

Configure your computer to securely connect to your future cloud server: SSH

SSH means Secure SHell. It's an encrypted network protocol that lets you securely connect to a remote computer. You use SSH by using a pair of public-private keys: a public one that you'll somehow give to the remote machine you'd like to securely connect to and a private one that's stored on your machine & is used by the server to authenticate you. The *TLS Encryption* section of the new *LS170 Networking Foundations* course explains how this works really clearly.

DO recommends using SSH to sign in & interact with your remote machines. Here's a link to their explanation of how to set up SSH keys on a Linux OS. (Instructions for Macs appear similar).

If you provide your public key to DO, DO can automatically register it with your droplets, letting you securely connect to your servers via SSH without enabling password authentication.

Create a Droplet

Basic walkthrough

Secure your Droplet by turning on a firewall and disabling root access

Basic walkthrough

- Use SSH to connect to your droplet using the public IP address DO assigns to your droplet
    - `ssh root@ipAddress`
- By default, we connect to our server with an unprivileged, superuser account called **root**. Best practices state that we should not use this account and instead create a user

account that can request elevated permissions when it needs to do things that require superuser account. We'll create a new account for ourselves on the machine and add it to the *sudo* group so it can request these permissions when needed.
    - `adduser userName`
    - `usermod -aG sudo userName`
- Next, to be able to SSH into the server and authenticate as our new account, we need to copy the public key DO registered with the server under the *root* account to our new account.
    - `rsync --archive --chown=userName:userName ~/.ssh /home/userName`
- Confirm that you can connect to your server using your username and that *sudo* works
    - `ssh userName@ipAddress`
    - Try running `sudo apt-get update` for a low-risk way to test sudo
- By setting up SSH as our way of connecting to the server, DO turns off password authentication by default. To further secure our server from malicious attack, we can also enable its firewall after we permit SSH connections to go through it:
    - `ufw status`
    - `ufw allow OpenSSH`
    - `ufw enable`
- Final steps for the justifiably paranoid--disable *root*
    - Open *sshd_config* by running `sudo nano /etc/ssh/sshd_config` (*nano* can be replaced with *vim* if you prefer that editor!)
    - Set *PermitRootLogin* to *no*
    - Run `sudo service ssh restart` to make the change take effect

Potential Next Steps

- Install Node
    - `sudo apt update`
    - `sudo apt install nodejs`
    - `sudo apt install npm`
- Set up a "starter" Express project. Follow the *Backend Setup* series.
- Use git & gitHub to get your code to your server