4 sessions

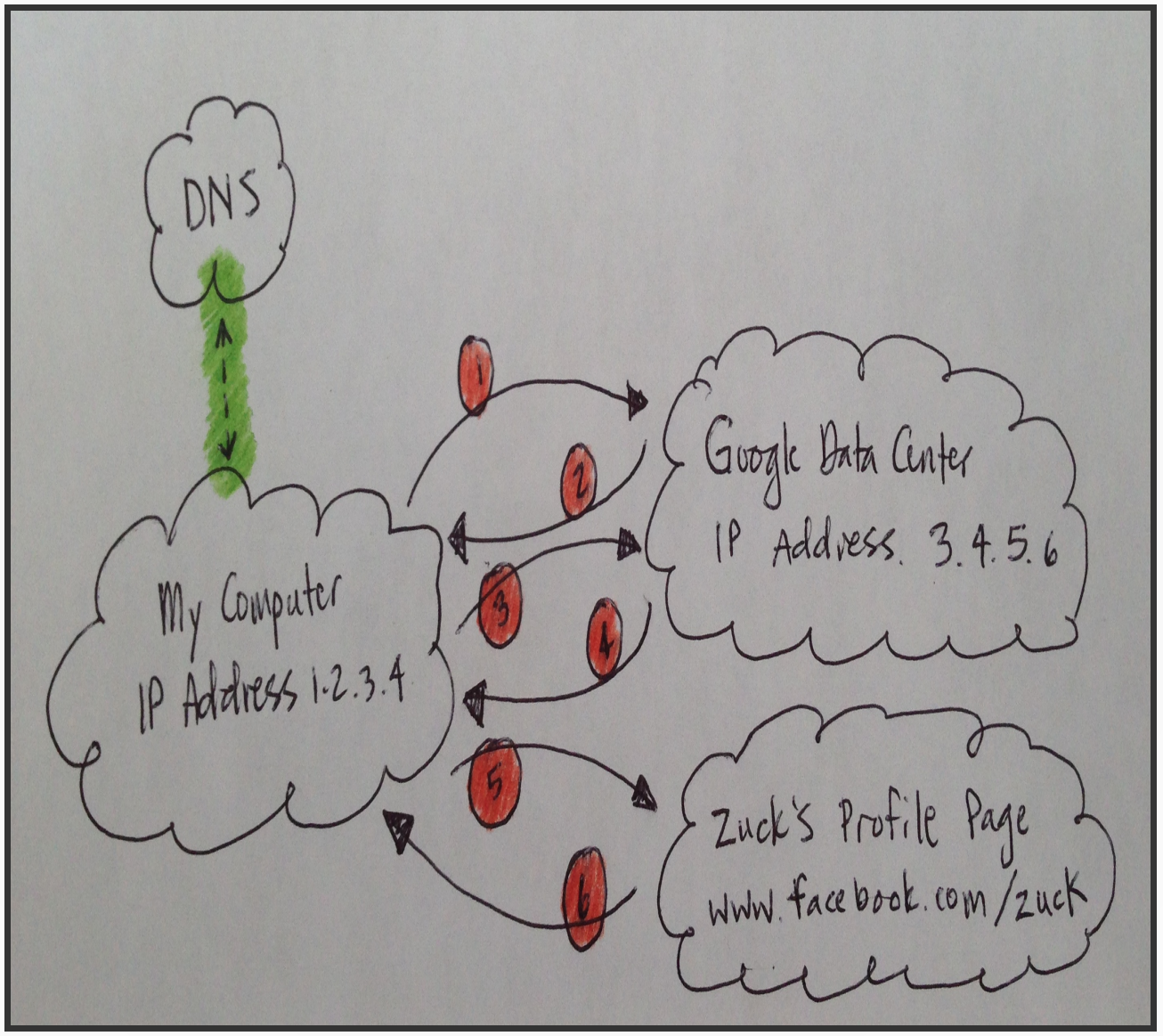
Session 1 - What are servers? What is the internet?

Session 2 - How to interact with servers

Session 3 - How to setup and use NGINX

Session 4 - How to use Vagrant

Session 1 Review



But ...

To control servers we'll cover

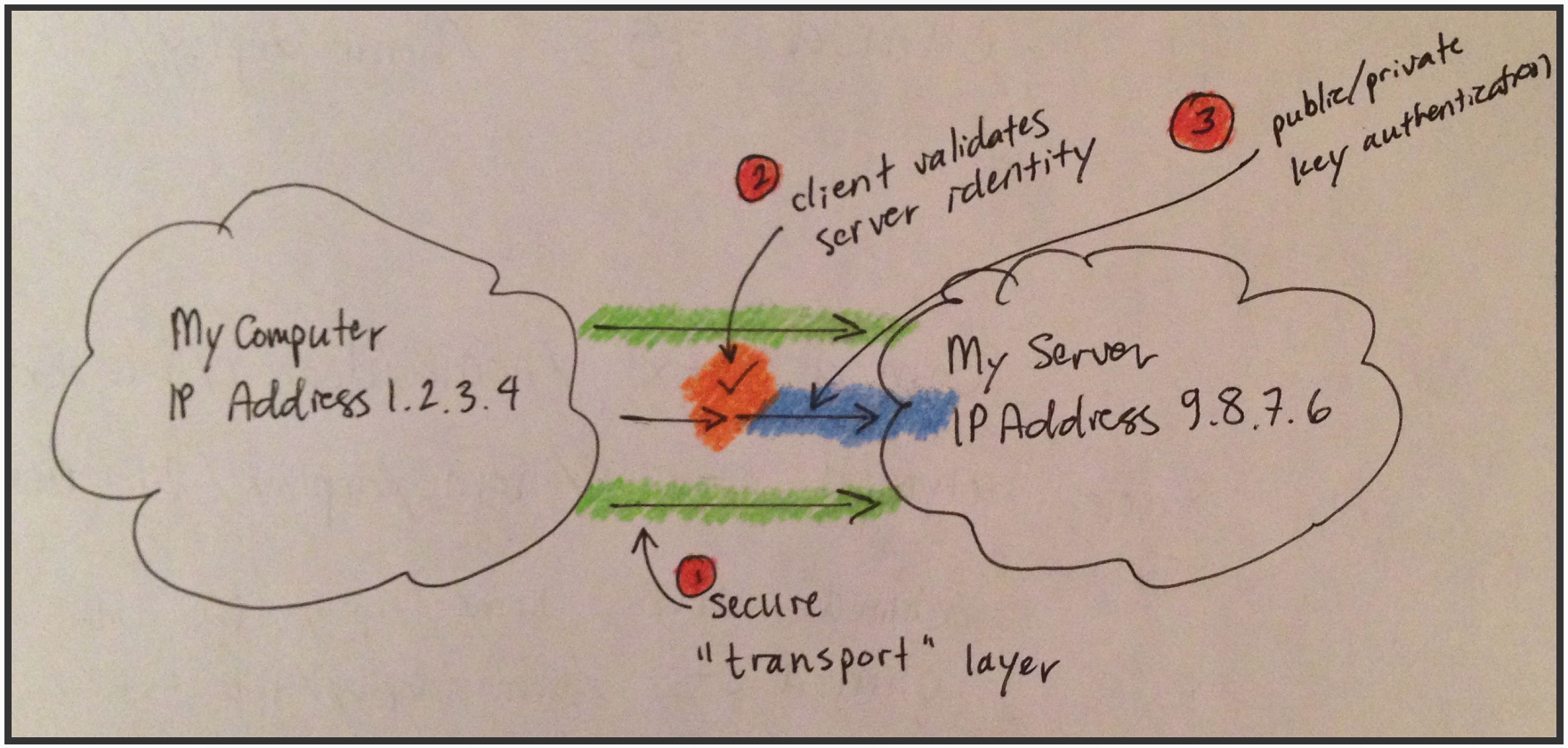
- SSH - (secure shell)

- Unix users
- Unix groups
- Unix permissions

- Packages

SSH - Secure Shell

encrypted tunnel connection between to 2 "clouds".



Exercise 1 - SSH into your server

```
ssh root@9.8.7.6  # (it will ask you for your password)
```

Enter the follow commands and write down the output

```
whoami
```

```
pwd
```

```
echo "I am in the cloud"
```

Windows users: use git bash

Mac users: use the terminal

Questions about Exercise 1

```
whoami    # OUTPUT => root
```

Why were you root?

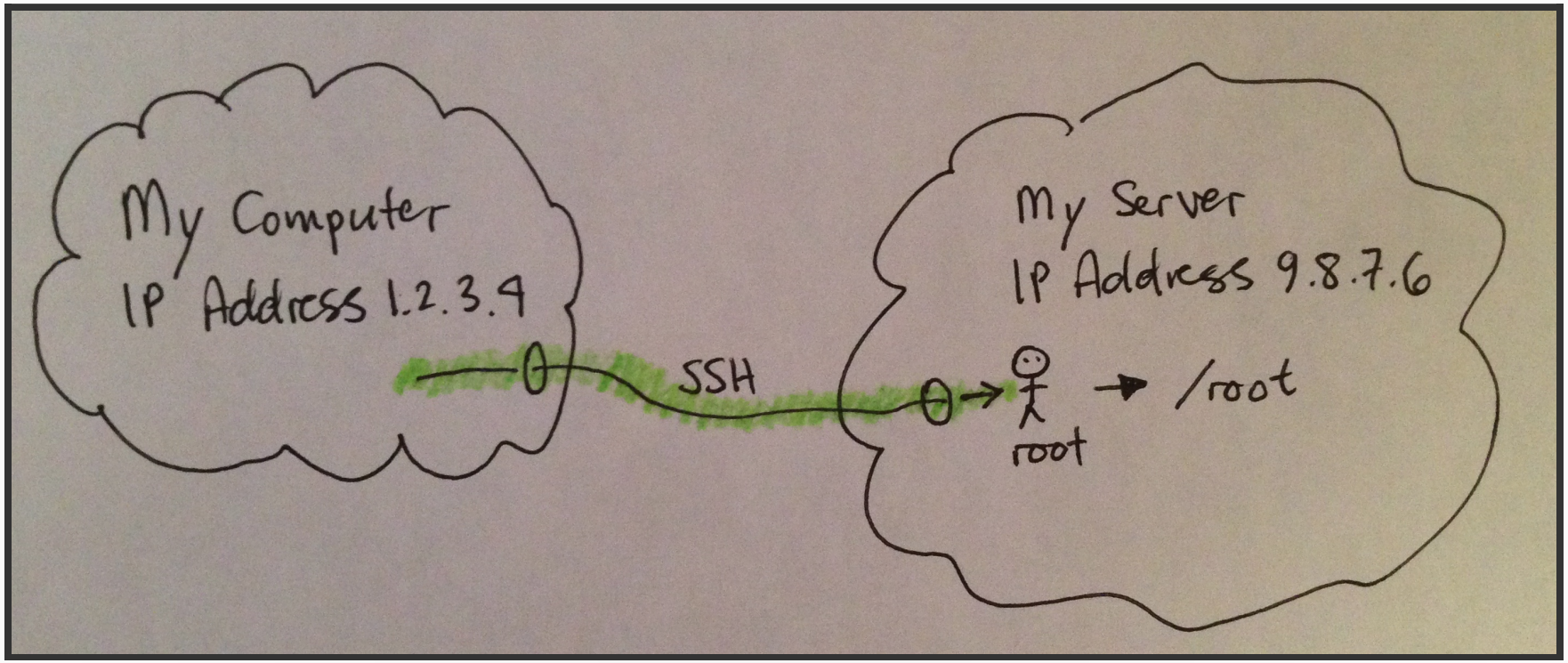
```
pwd       # OUTPUT => /root
```

What does this tell you about your "physical" location?

```
echo "I am in the cloud"     # OUTPUT => echo "I am in the cloud"
```

Did this command surprise you?

Update our cloud illustration



Unix users

Being the "root" user is great, but ...

- authenticating by typing in password = weak
- allowing root to login into your server = weak
- we want to learn about Unix users

weak = security vulnerability, you get hacked

therefore, we're going to create another user!

Unix users

What did the root user have?

- Username
- Home Directory

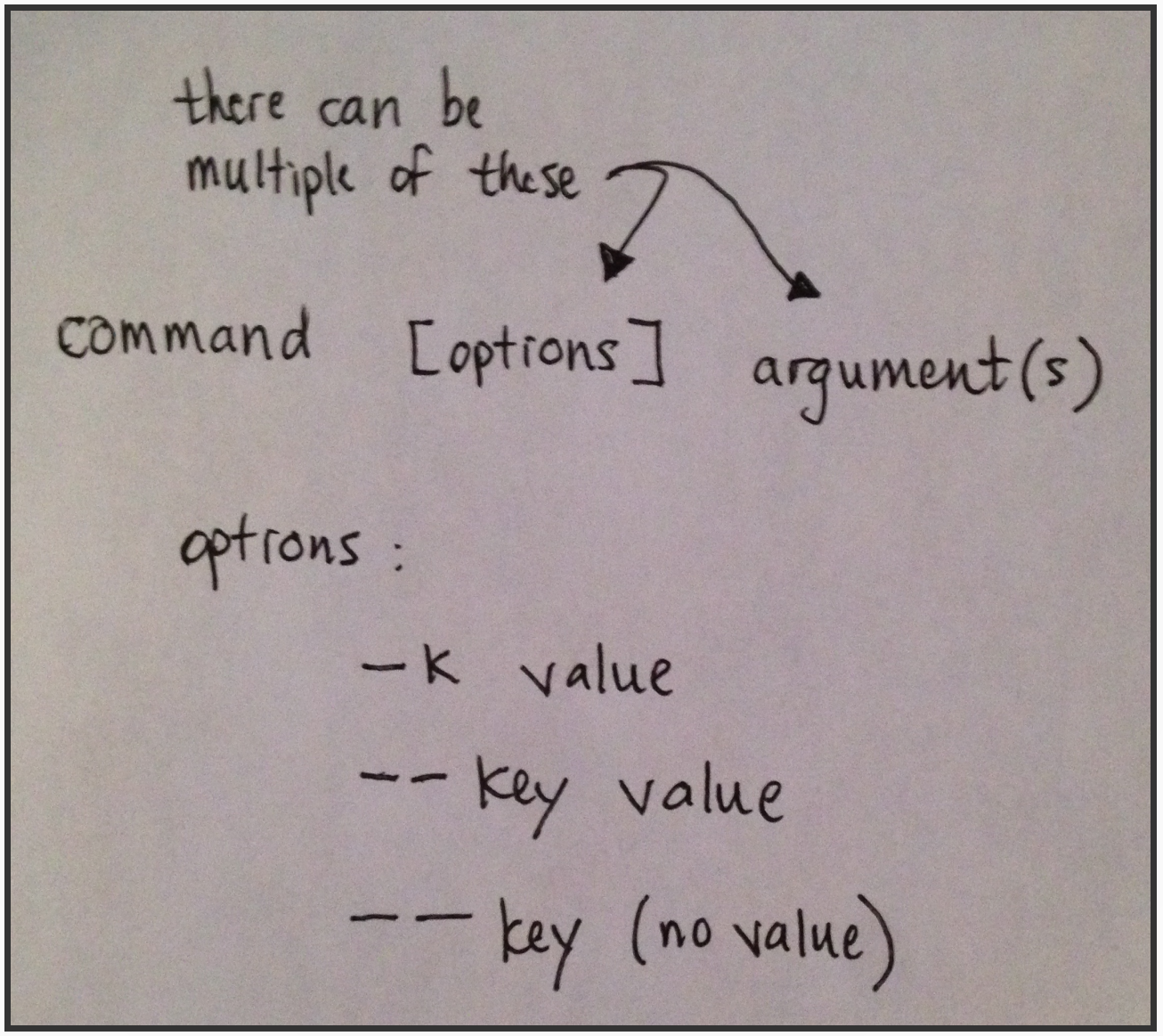
shell command = useradd + options

- Password

shell command = passwd

Yes, "passwd". Omit the "o" and "r".

Uxix shell commands



Exercise 2 - Step 1 - useradd

Command

```
useradd
```

Argument

```
deploy
```

Options

```
key   = -d
value = /home/deploy

key   = -m
value =

key   = -s
value = /bin/bash
```

What's the resulting command?

```
_______    __  ______   ___   __  _______    _________
```

Exercise 2 - Step 2 - passwd

Command

```
passwd
```

Argument

```
deploy
```

Options

```
(none)
```

Resulting Command

```
_____   _______
```

Enter password = "gdi" twice

Exercise 2 - Step 3 - test it

SSH into server as new "deploy" user

```
ssh deploy@9.8.7.6
```

Test who you are

```
whoami
```

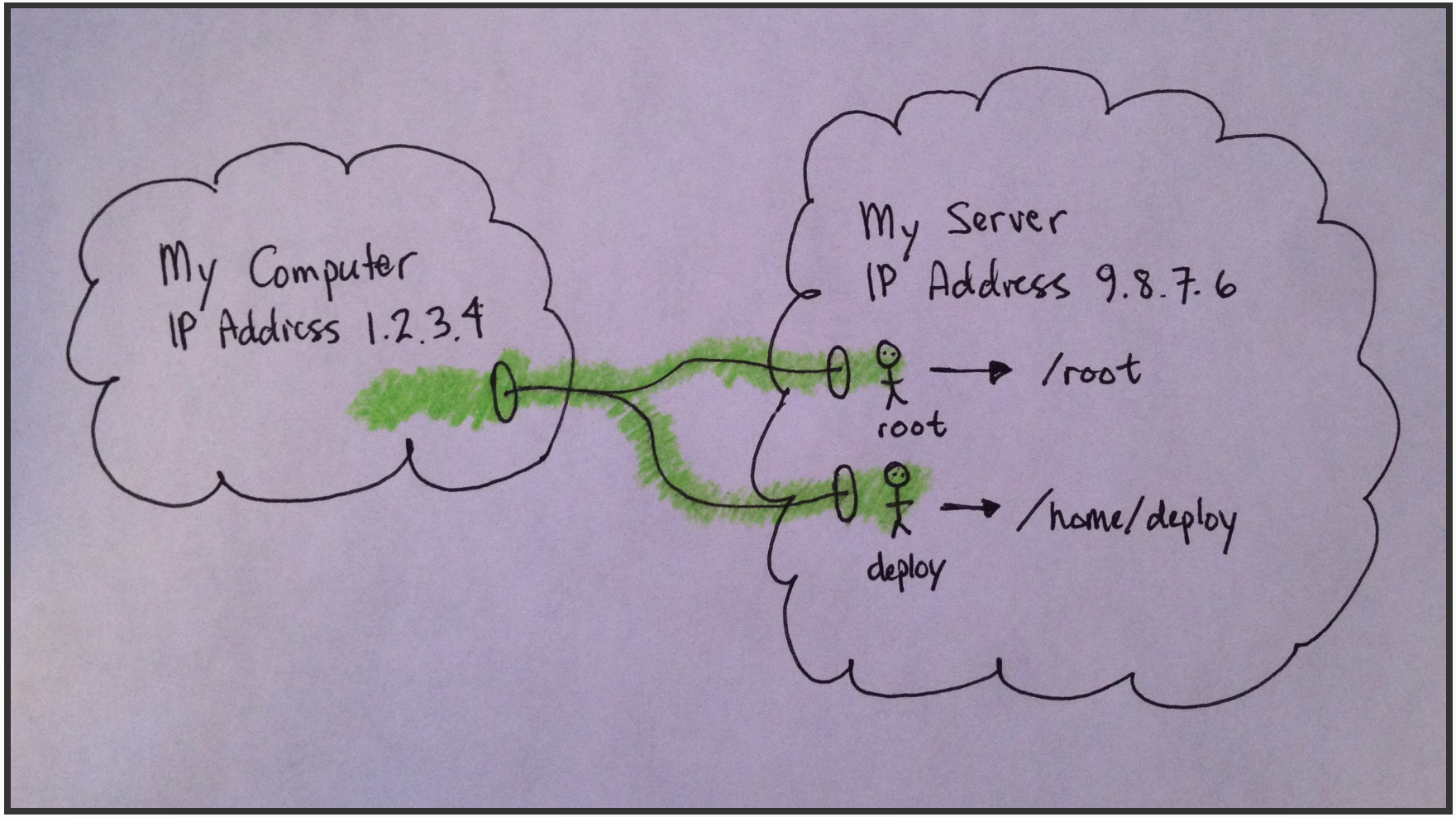
Test where you are

```
pwd
```

Test out the "echo" command

```
echo "I am in the cloud"
```

Update our Cloud picture



Unix groups and permissions

What can a user do once they login into the server?
How do we control what happens on the server?

Macs have a "System Adminstrator"
Window machines have an "Admin"

Unix systems have a sudo group

Unix groups

For our purposes, the sudo group membership grants access to do almost anything.

Selectively grant permissions to users by adding them to groups

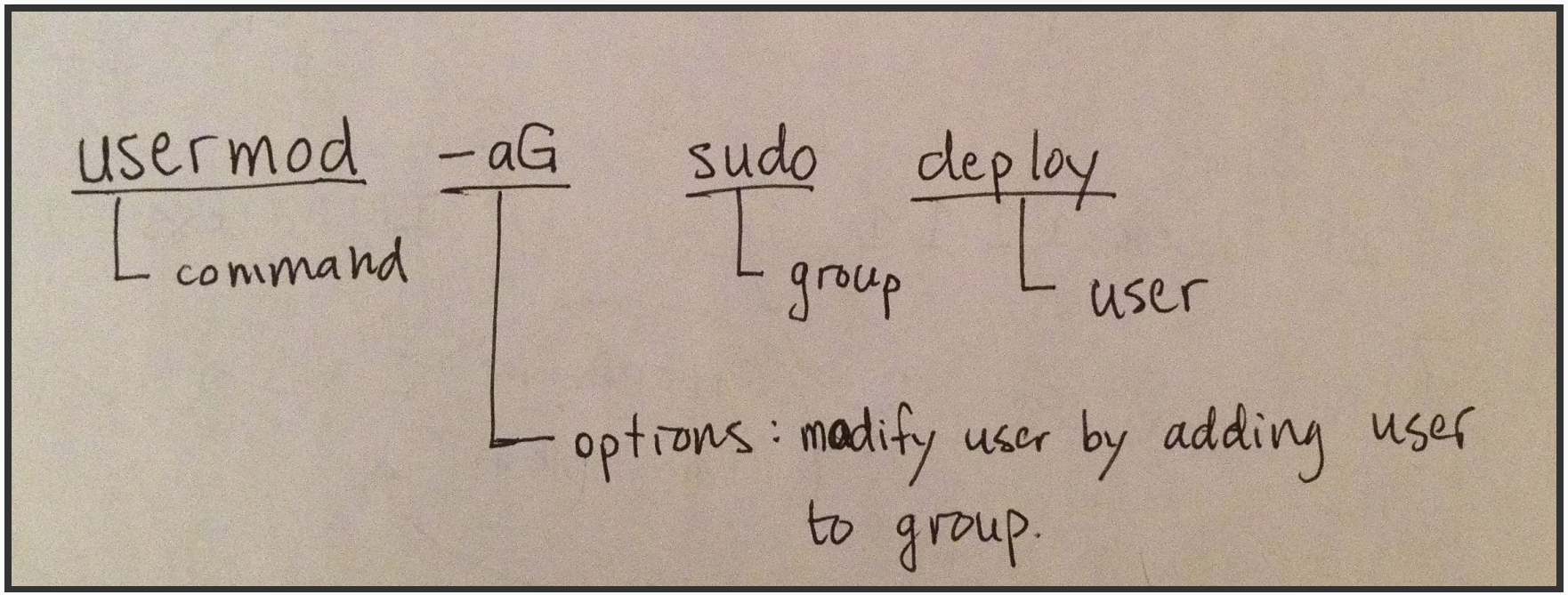
By invoking sudo while running a shell command, you're invoking root acccess

```
sudo echo "I am root here"
```

Unix groups

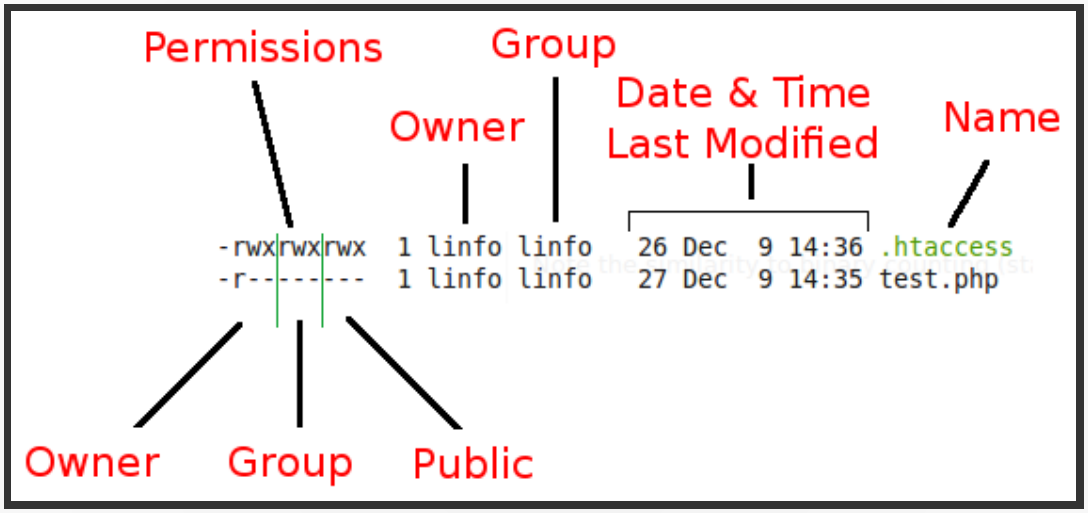
Add user to group:

```
usermod -aG sudo deploy
```



Unix permissions

Permissions are access rules set on directories and files.

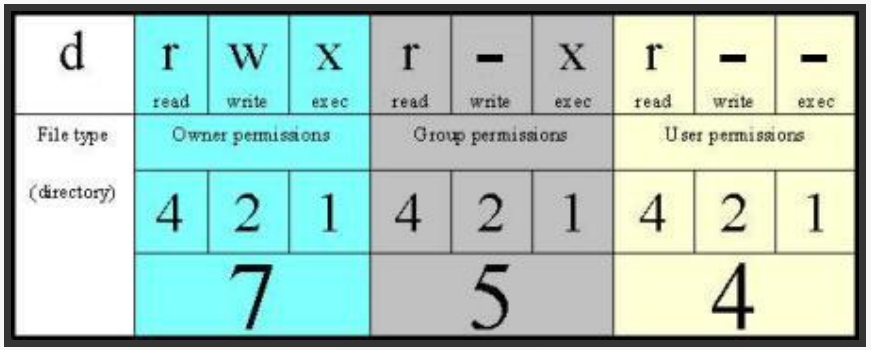


3 types of access rules:

- r --- read
- w -- write
- x --- execute

Unix Permissions

There's math in Permissions



d	r read	w write	x exec	r read	- write	x exec	r read	- write	- exec
File type (directory)	Owner permissions			Group permissions			User permissions		
	4	2	1	4	2	1	4	2	1
	7			5			4		

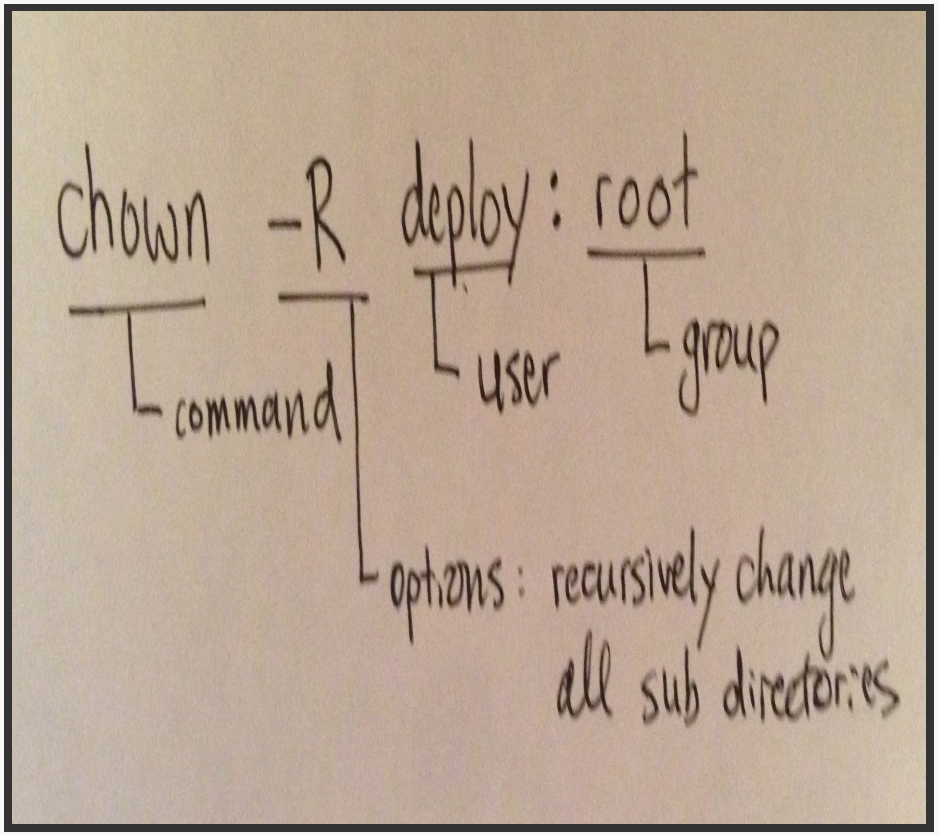
http://krnlpanic.com/wp/wp-content/uploads/2013/03/chmod.jpg

- r --- read = 4
- w -- write = 2
- x --- execute = 1

Unix Permissions

Change owner and/or group

chown



Unix Permissions

Change permissions (read, write, execute)

chmod

Numbers - set user, group, and others all at once

```
sudo chmod 755 /home/deploy/files.txt
```

Words - Edit selectively

```
sudo chmod u+w /home/deploy/files.txt
```

```
sudo chmod g-r /home/deploy/files.txt
```

Words - Set multiple

```
sudo chmod ux=rx /home/deploy/files.txt
```

Exercise 3 - Step 1
Add deploy to "sudo" group

ssh into server as "root"

```
ssh root@9.8.7.6
```

Command

```
usermod
```

Arguments

```
arguemnt 1 = sudo
argument 2 = deploy
```

Options (required)

```
key   = -aG
value =
```

What's the resulting command?

```
_____   ___   _____   ______
```

Exercise 3 - Step 2
Ensure deploy added to group

SSH into server as "deploy"

```
ssh deploy@9.8.7.6
```

Command

```
groups
```

Argument (optional)

```
# (blank - defaults to current user)

deploy
```

What's the resulting command?

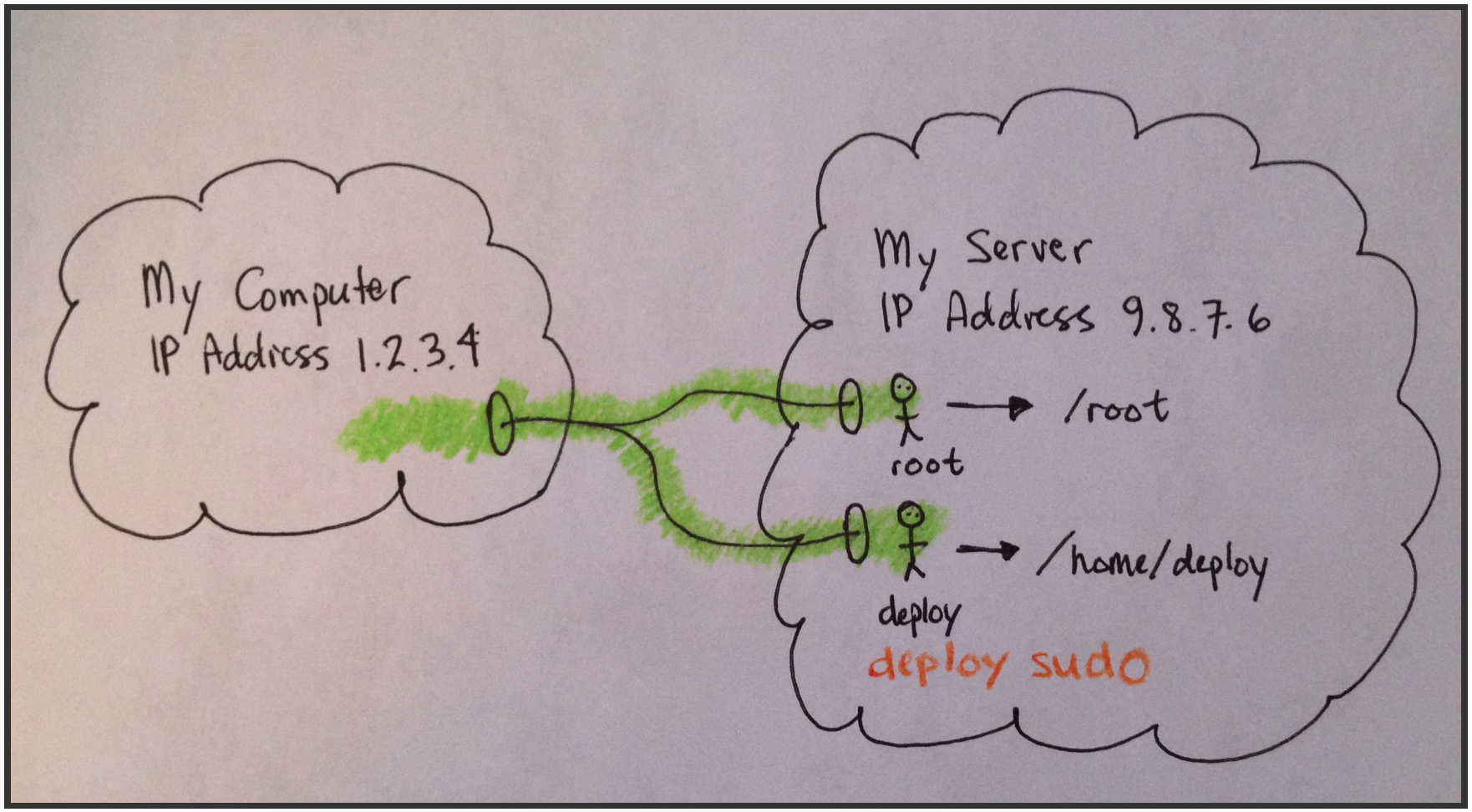
```
______
______    _______
```

Exercise 3 - Test out sudo

Use sudo with echo

```
sudo echo "I have power"
```

Update our Cloud picture



To control servers we'll cover

- SSH - (secure shell)

- Unix users
- Unix groups
- Unix permissisions

- Packages

Unix Packages

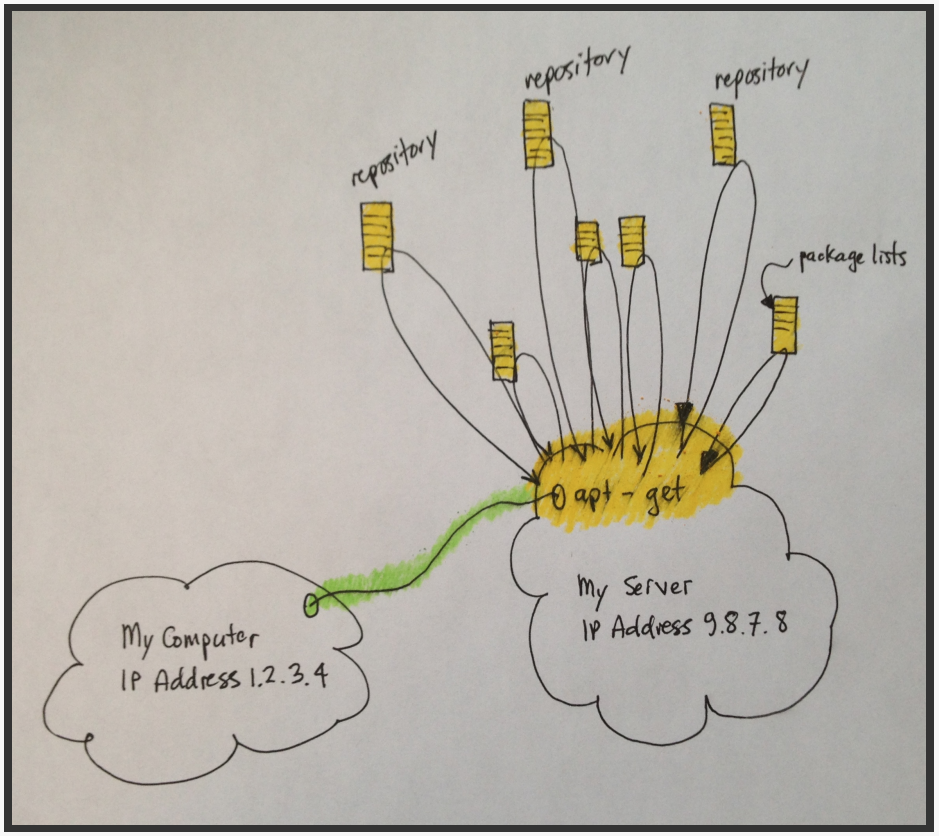
A "Package" is a software program you install

Default package manager for Linux/Ubuntu

apt-get

What's a package manager?

Gathers package lists from multiple sources



Install nginx walk through

You'll need to use root access via the sudo commnd

Update the apt-get package lists

```
sudo apt-get update
```

Install nginx

```
sudo apt-get install nginx
```

What happened?

```
deploy@intro-00:~$ sudo apt-get install nginx
[sudo] password for deploy:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  libgd2-noxpm libjpeg-turbo8 libjpeg8 libxslt1.1 nginx-common nginx-full
Suggested packages:
  libgd-tools
The following NEW packages will be installed:
  libgd2-noxpm libjpeg-turbo8 libjpeg8 libxslt1.1 nginx nginx-common
  nginx-full
0 upgraded, 7 newly installed, 0 to remove and 56 not upgraded.
Need to get 882 kB of archives.
After this operation, 2,692 kB of additional disk space will be used.
Do you want to continue [Y/n]?
```

```
Y
```

```
Get:1 http://mirrors.digitalocean.com/ubuntu/ precise-updates/main libjpeg-tu
....
Setting up nginx (1.1.19-1ubuntu0.5) ...
Processing triggers for libc-bin ...
ldconfig deferred processing now taking place
deploy@intro-100:~$
```

How do we figure out where nginx was installed?

we didn't tell if where to install

apt-get didn't log that info

what if there was a tool ...

there is a tool

```
where
```

```
where  # => /etc/nginx
```

How do we move around within the server?

cd - "change directory"

```
cd /etc/nginx
```

Word of caution - you cannot

```
sudo cd /root
```

Why not?

sudo works on executables, and "cd" is a shell builtin

How do we see what's in the of a folder?

ls - List files and directories

```
deploy@intro-00:/etc/nginx$ ls

conf.d          koi-win           naxsi.rules   scgi_params       uwsgi_params
fastcgi_params  mime.types        nginx.conf    sites-available   win-utf
koi-utf         naxsi_core.rules  proxy_params  sites-enabled
```

also list user, groups, and permissions

```
deploy@intro-00:/etc/nginx$ ls -als
total 68
4 drwxr-xr-x  5 root root 4096 Feb  8 09:38 .
4 drwxr-xr-x 90 root root 4096 Feb  8 09:38 ..
4 drwxr-xr-x  2 root root 4096 Nov 21 22:15 conf.d
4 -rw-r--r--  1 root root  898 Dec 13  2011 fastcgi_params
...
4 drwxr-xr-x  2 root root 4096 Feb  8 09:38 sites-enabled
4 -rw-r--r--  1 root root  497 Sep 25  2011 uwsgi_params
4 -rw-r--r--  1 root root 3071 Sep 25  2011 win-utf
```

Files are present, but how do we know if it's running?

There's a lot of ways. Easiest way,

```
ps aux | grep nginx
```

Control NGINX via the service command

The service command connects to nginx the "init" script (/etc/init.d/nginx)

What can it do?

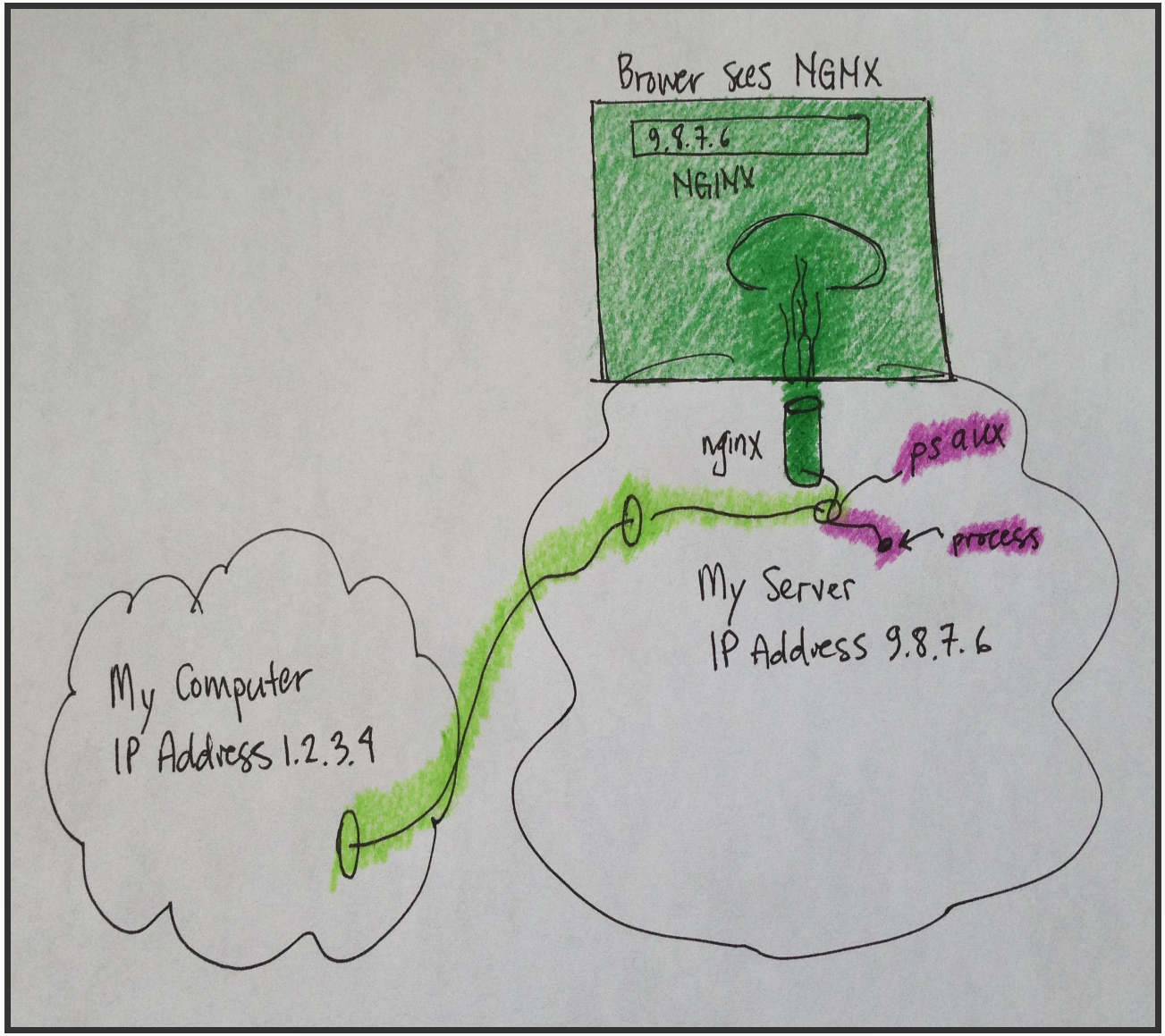
```
deploy@intro-00:/etc/init.d$ sudo service nginx

Usage: nginx {start|stop|restart|reload|force-reload|status|configtest}
```

Start nginx

```
sudo service nginx start
```

Update our cloud illustration



See the result in your browser

Copy and paste the IP address into your browser

Exercise 4 - Install nginx

- Update apt-get
- Use "apt-get" to install nginx
- Use "where" to find install location
- Use "ls" to confirm files are present
- Use "ps" to see if nginx is running
- Start nginx
- See NGINX in your browser

If time allows

Exercise 5
SSH public/private key auth

1. Follow the Github tutorial

help.github.com/articles/generating-ssh-keys

create your SSH public/private keys

2. Make ".ssh" directory

```
ssh deploy@9.8.7.6
cd /home/deploy
mkdir .ssh
```

3. Copy your "id_rsa.pub"

SSH public/private key auth

4. "Create" a file and paste in id_rsa.pub

Start by,

```
cd /home/deploy/.ssh
nano authorized_keys
```

Paste in your "id_rsa.pub"

Save, close file, and exit out of SSH connection

5. SSH into server (you should not need password)

```
ssh deploy@9.8.7.6
```

SSH - Secure Shell

encrypted tunnel connection between to 2 "clouds".

We covered

- SSH - (secure shell)

- Unix users
- Unix groups
- Unix permissions

- Packages
- SSH without passwords!

4 sessions

Session 1 - What are servers? What is the internet?

Session 2 - How to interact with servers

Session 3 - How to setup and use NGINX

Session 4 - How to use Vagrant