you'll only be a cloudacharya when

you've killed and raised ten thousand fiery hosts

Tadpoles Three

In this cool cloud miniquest you get to do the following:

- Fire up an aws ec2 instance and configure it properly
- Get an Elastic IP and associate it with your instance
- Set me up with a way to enter it
- Plant a secret message on your machine that only I can retrieve securely
- Set up a public facing web site
- Configure your web server such that I can serve my own content from my own home page in **my** home directory on **your** instance
- Serve the greeting "hello world" through another publicly accessible, but secret text file

You'll find the videos on our youtube channel (nonlinearmedia) pretty much walk you through most of the steps needed to complete this quest. But I've added in a few new surprises here to keep up the excitement. Where necessary, I'll explain these new things in this spec.

Try and complete all the miniquests using the aws management console. Spare the command line for now. You get to play with the shell inside your instance.

The instructions in this spec are deliberately brief. *When in doubt*, *try it out*. Read the aws documentation or ask us in the discussion forums.

And before you start, here is an important note about these quests

The spec may say "do this" or "do that". Try to do all of it *even though* only a subset of these actions (known only to me) will be tested.

Sometimes I might test something and tell you if something went wrong. Other times, for non-critical but good tests, I will NOT tell you if it failed, but WILL award extra questing points if it succeeded. So your best bet is to do everything asked.

Let's get started.

Your first miniquest - Fire up an EC2 instance

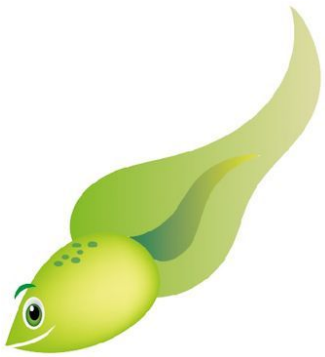
Use the management console to launch an EC2 instance. Configure it as you like (I recommend using a free-tier AMI and instance type). However, you must use a standard Linux image. If you're only comfortable with other environments, consider this an opportunity to get to know the Linux world.

Create a firewall that opens up the ports for SSH and HTTP (but **not** HTTPS).

If you didn't already get a public IP during the launch process, obtain an Elastic IP (EIP) and associate it with this instance as it's booting up. If you don't know how, see the week 1 video on our YouTube channel or ask.

Now get the public DNS name of this instance which you can discover in its details page.

This is the hostname you must submit into the questing site. But hang on. Complete the remaining miniquests before you do.



Your second miniquest - Give me a way to enter it

In this miniquest, you're going to set me up as a user (not in your AWS account, but in the Linux box and give me a way to enter your instance. Here's what you'll do.

1. Create a user called `anand`
2. Obtain my public key
3. Install it within **my** home directory on **your** machine

There's an old photo you'll find somewhere. It kinda looks like it has most of the commands you need. I can't guarantee its correctness. You gotta find out yourself what works and what doesn't. Of course, if you're stuck, I'm only a holler away. And you can always ask your classmates too. Anyone who's comfortable with the above process should feel free to offer step by step directions for those that need help in the discussion forums. Here is more detail:

1. First fetch my public key, which can be found at

   https://quests.nonlinearmedia.org/keys/cs55a_quest_1.id_rsa.pub

2. Use the `wget` or `curl` commands. If anything is confusing, all you have to do is ask. That way we can find out how much actually needs to go into the next version of this spec.
3. Put this file within the ~/`.ssh` subdir of my home dir and rename it to *authorized_keys*. In most command line shells on various Unix flavors, you can use the tilde character, ~, as a prefix to a user's name (default is your own) as a synonym for their home directory. So ~anand would be /home/anand, or /users/anand, or something else, depending on the underlying system. Mostly you don't have to care.

4. Set the permissions and ownerships of the installed files like so:
    a. `~anand/.ssh` should be 0700
    b. `~anand/.ssh/authorized_keys` should be 0644
    c. Both should be owned by `anand`

    You probably have to use the Unix commands `chown` and `chmod,` and have to `sudo` a fair bit. I recommend you use `sudo` rather than turn yourself into root. If you're lost, PLEASE ask and I can update this spec within an hour (mostly).

5. The commands you're after are probably like (not exhaustive and not in any particular order)
    a. `sudo useradd anand`
    b. `sudo cp /tmp/cs55a_quest_1.id_rsa.pub .`
    c. `sudo chown anand .ssh`
    d. `cat authorized_keys`
    e. `cat /tmp/cs55a_quest_1.id_rsa.pub`
    f. `cd .ssh/`
    g. `sudo chown anand ./authorized_keys`
    h. etc.

How can you make sure your setup works in letting me in? Here's a suggestion:

Make your own public and private key pair. Use the `ssh-keygen` command on your Linux instance to make it. Read up its documentation, or ask one of us how to use it.

1. When it asks you for the name of the file to save the key in, give it a name, like `'expt1-joe.id_rsa'` or something.
2. Make sure NOT to protect your key pair with a password when prompted.
3. This will leave two files in your own `~/.ssh` directory. These will be `~/.ssh/expt1-joe.id_rsa` and `~/.ssh/expt1-joe.id_rsa.pub`.
4. The first file is your private key. Change its mode to 400 and leave it alone.
5. The second is its matching public key.

When you create your experimental user, say expt1-joe or something, you want to install this public key in that user's authorized_key's file in the same way as above. When that's done, you should be able to login to this machine as expt1-joe by using the command:

```
ssh -o "StrictHostKeyChecking no" -i {key} expt1-joe@{your domain name}
```

The -o option tells the ssh command to not interactively confirm with you whether it's ok to connect to a previously unknown host.

If this command succeeds from any connected machine on which the named private key can be found, then you should have no problems in acing this miniquest.

Hey! Look what I found. Some old piece of parchment in the pond. This is what I could salvage. Maybe it'll be of some help…

Your third miniquest - Install a web server and give me my own home page

This is a fun little miniquest. You get to install `httpd` (the web server). But before you start it, you need to make a small change to its configuration so that I can host my home page on your machine within my own `~anand/public_html` directory.

Again, to make sure that you pass this quest, try to do it for your experimental user and make sure it works.

Here are the details:

1. Install `httpd` (Consider using `sudo yum install httpd`)
2. Configure it to serve per-user home pages. You can find the full details at

   https://httpd.apache.org/docs/2.4/howto/public_html.html

   But, spare yourself the bother for now. All you have to do, for this miniquest, is (yeah, that reminds me, you may have to install your favorite editor if you can't find it. E.g. run something like *sudo yum install emacs*.):

   1. Edit the file `/etc/httpd/conf.d/userdir.conf` - of course, this file wouldn't exist until after you've installed `httpd`.
   2. Find a line that says "`#UserDir public_html`". Remove the `#` to uncomment the line.

3. Save the file.

Now you're ready to start `httpd` (e.g `sudo service httpd start`)

When `httpd` comes up, you'll notice that you can access your own `~/public_html` pages at

```
http://{your domain name}/~ec2-user.
```

If this ends in *Permission Denied* errors it's almost always the case that the permissions on some component of the absolute path leading up to your ~/public_html is a no-fly zone for the world. You don't have control over your own home's parent. However, you can make sure your home offers fly-over to the world seeking to land in your public_html directory. Do:

```
cd; chmod 711 .
```

Make sure you get all the characters above, including the semicolon and the period. They're little details that make all the difference.

Hooray!

Try to confirm that you follow the process end-to-end by doing the same thing for your experimental user.

Your fourth miniquest - Share a secret

You're gonna do a few interesting new things here.

1. You will need to tell me your student id *secretly*. You do this by creating a file with the name `.cs55a-secret` within my home directory ~anand (not my public home on the web). It should contain exactly one line with your student id. The file name should match exactly. For example, if your ID is 12345678, you would create the file `~anand/.cs55a-secret` which contains the line 12345678
2. Now you're going to determine a secret location in your public document root (yes - anyone can access it if they knew the URL. This is like unlisted videos on YouTube. The location will be hard to guess for those who don't know your student id.

   How? You will use the unix command `md5sum`. It's located in the directory `/usr/bin`, which is part of your default command search path for ec2-user. You need to create a single text file called `secret.txt`, and place it at the following url:

   ```
   http://{your domain name}/{secret id}/secret.txt
   ```

YES - the file you're creating is a text file with a `.txt` extension (not html). `Httpd` knows to serve text files too.

This file should contain one line with the words *hello world*. It should go into your web server's document root (/var/www/html) at the location `/var/www/html/{secret id}/secret.txt`

Replace `{secret id}` in the above url with the md5 checksum of your student id. However, you must be careful not to include the newline character or other invisible characters in calculating this checksum. I suggest using the following command:

```
echo -n {Your student id} | md5sum
```

The `-n` option to the `echo` command tells it to print the rest of the command (up to the vertical bar) without a trailing newline. The vertical bar takes the output of the `echo` command and *feeds* it to the `md5sum` command, which should tell you the checksum you need.

The hyphen at the end of its output simply says that the calculation was done for characters that `md5sum` received on its standard input (for which the hyphen is an alias). You can safely ignore it.

BTW, it just occurred to me that although you may have gotten rewards for all previous miniquests, unless you pass this one, none of those points actually counts! So make sure you pass this fourth miniquest.

Until then you can't say "*all my rewards are belongs to me*"

One final detail. Protect your secret directory by creating an empty `index.html` in it. This will discourage snoopers who try to find what the directory contains.

Testing your setup

Like I said, the best way to make sure you ace all the miniquests is to set it all up in such a way that your experimental user is able to do all the things I can.

And nothing that I can't.

Submission

When you think you're happy with your code and it passes all your own tests, it is time to see if it will also pass mine.

1. Head over to https://quests.nonlinearmedia.org
1. Enter the secret password for this quest in the box.

2. Drag and drop (or copy and paste) your public domain name for your host. This should be the aws assigned name. Not another domain name you have attached to your IP.
3. Wait for me to complete my tests and report back (usually a minute or less).

Points and Extra Credit Opportunities

I monitor the discussion forums closely and award extra credit points for well-thought out and helpful discussions.

May the best cloudsters win. That may just be all of you.

Happy Questing,

&