Step 0: Create an AWS account

What are you waiting for? Sign up!

Note: AWS requires a credit card for registration. But our example will exist entirely on the AWS Free Tier, so you won’t be charged.

Step 1: Download the Flask example code

(NOTE we are doing this to copy the config files)

Fork my Github repo: https://github.com/inkjet/flask-aws-tutorial

We’re going to deploy a simple app that reads and writes from a database using Flask-SQLAchemy. Dig into the code if you’d like — it should run locally if you run **python application.py** (after you set you the environment in Step 4).

Step 2: Set up your Flask environment

In the directory where the example code exists, create a Python virtual environment.

```
$ virtualenv flask-aws
```

```
$ source flask-aws/bin/activate
```

Then install the packages needed for this demo with:

```
$ pip install -r requirements.txt
```

Step 3: (Optional) Play with the Flask App

You can check out the site before you deploy it. Now that your environment is set up, run the app by

```
$ python application.py
```

And you can see it in your browser at http://0.0.0.0:5000

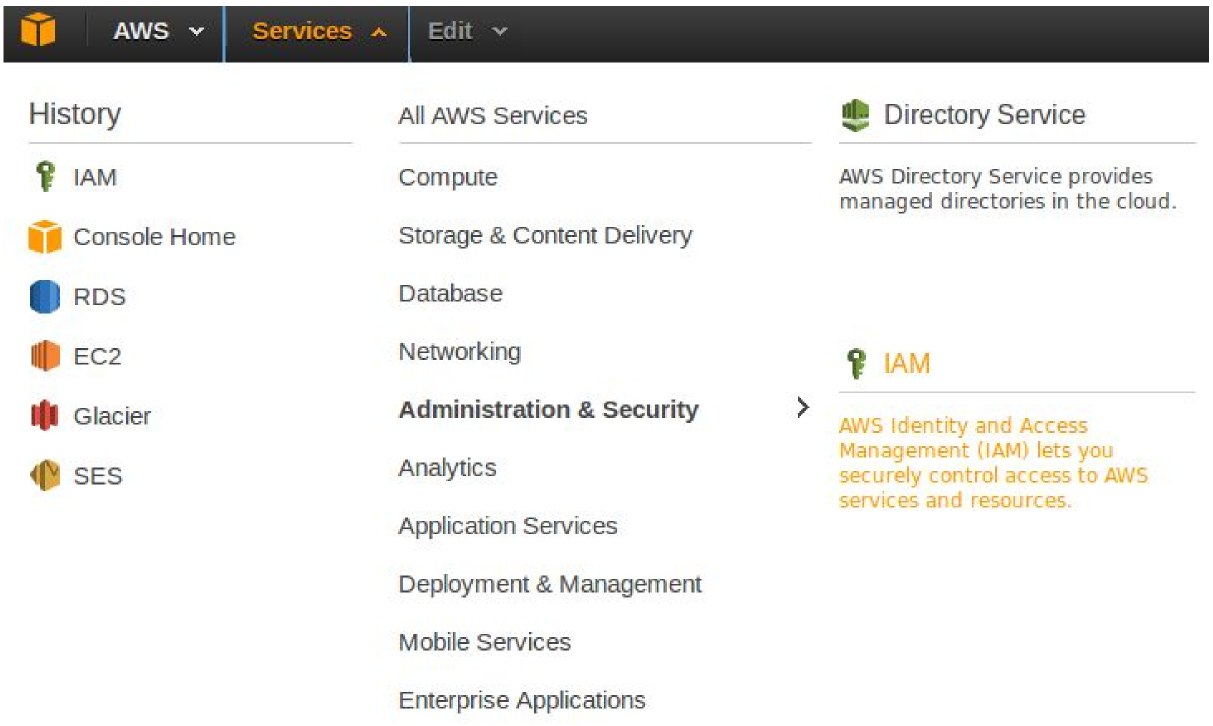
Step 4: Set up Elastic Beanstalk Environment

Amazon has made deployment easy with their Elastic Beanstalk Command Line Interface (EBCLI). It’s available on PyPi, so in your virtual environment type:

```
$ pip install awsebcli
```

Once that finishes installing, we can initialize and deploy our Flask app. We could do this using our AWS root access. However, let’s follow AWS Best Practices and create a new user for this demo. This will keep our master ID and secret key (which you should have stored somewhere) safe.

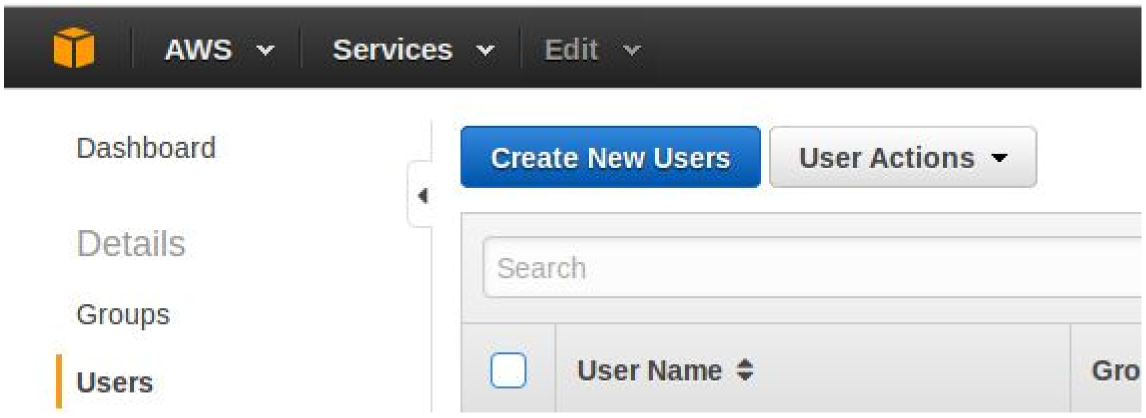
To create a new user, go to the AWS Console and select Identity and Access Management (IAM):



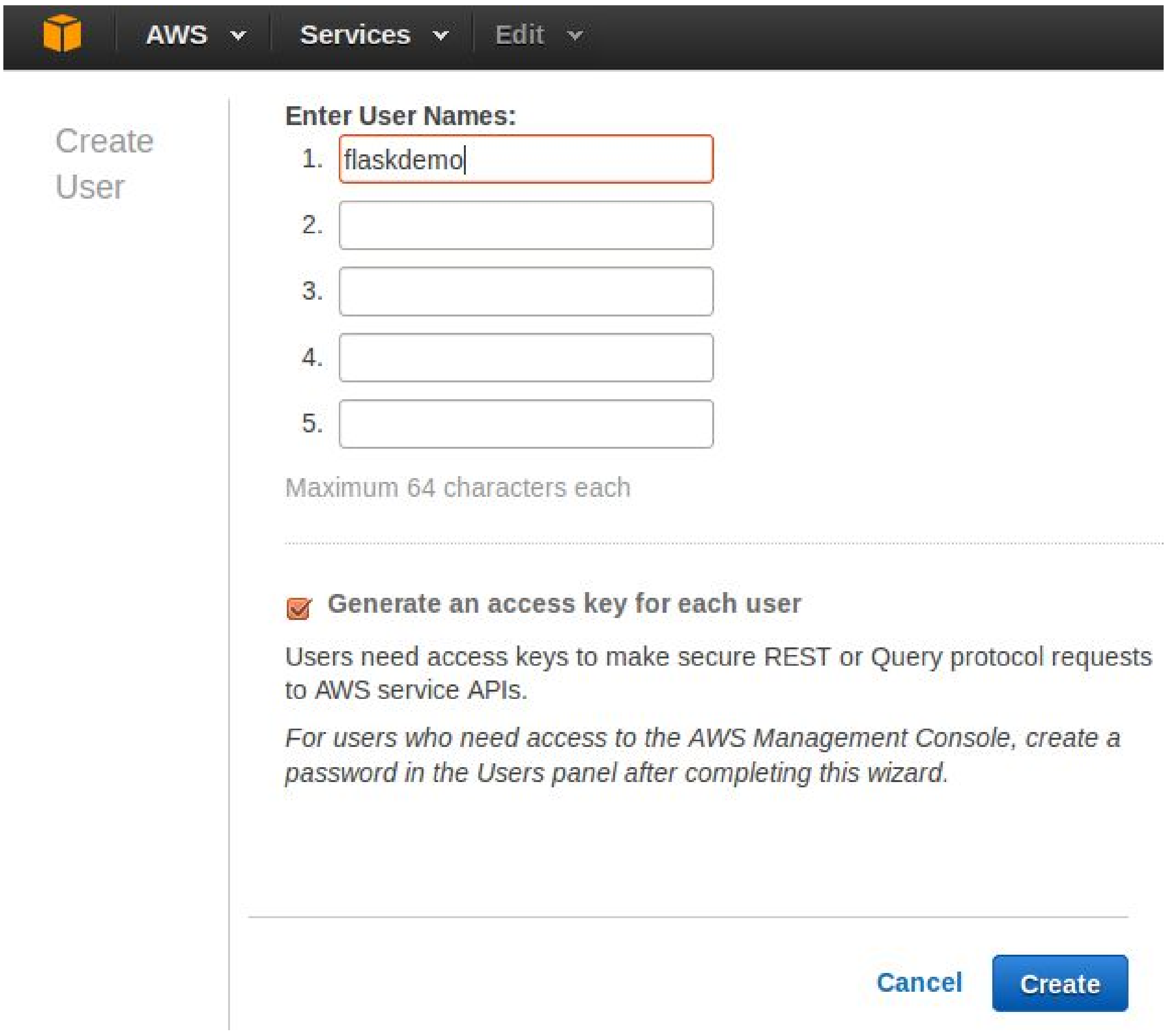
AWS Directory Service provides managed directories in the cloud.

AWS Identity and Access Management (IAM) lets you securely control access to AWS services and resources.

Next, select “Create New Users” or “ADD USER “:



We’ll only need one user for this demo, and we’ll call him/her “flaskdemo” or any other of your choice:



Enter User Names:

1. flaskdemo
2.
3.
4.
5.

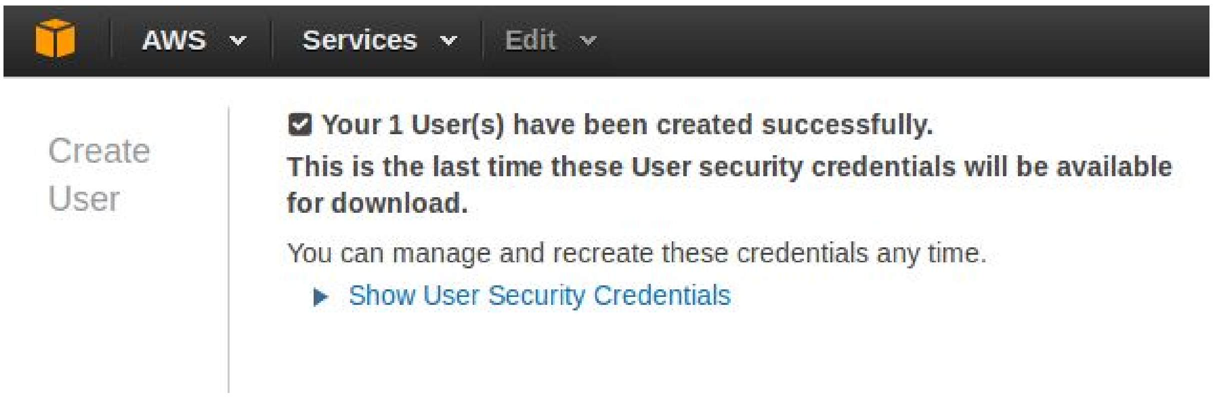
Maximum 64 characters each

✓ **Generate an access key for each user**

Users need access keys to make secure REST or Query protocol requests to AWS service APIs.

For users who need access to the AWS Management Console, create a password in the Users panel after completing this wizard.

Click “Create” and the user is established.



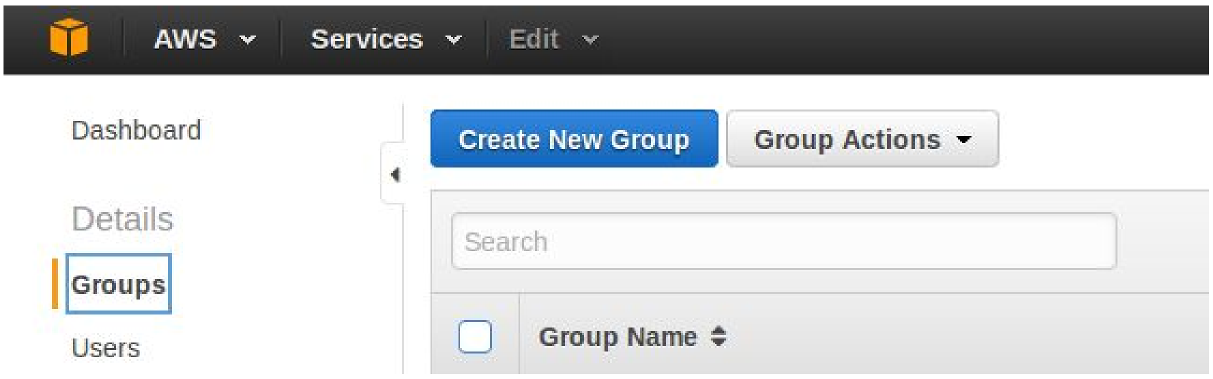
✓ **Your 1 User(s) have been created successfully.**
This is the last time these User security credentials will be available for download.

You can manage and recreate these credentials any time.

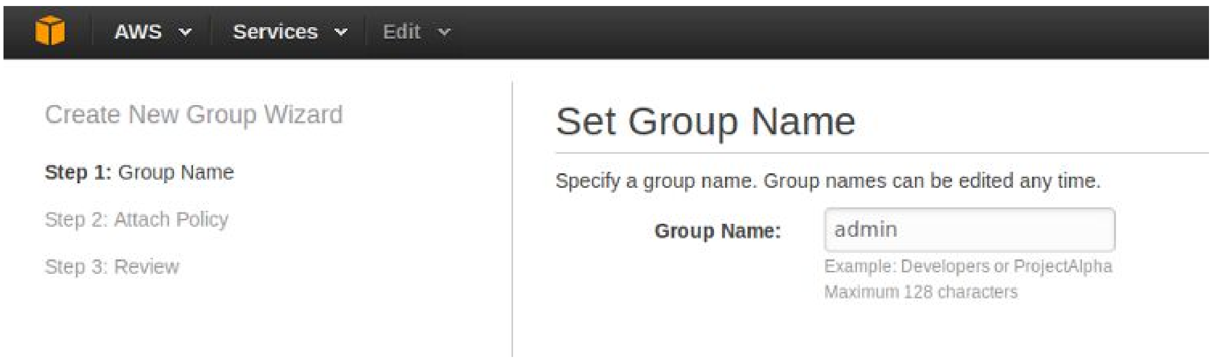
▸ Show User Security Credentials

IMPORTANT: You need to click “Show User Security Credentials” and copy down the ID and Secret Key for this user. We’ll need this to connect via the EBCLI interface.

We’ve created a user, but they have no permissions (just like our database). Let’s grant this user admin access. In the IAM window, select “Groups” and then “Create New Group”:



In the next screen, type “admin” as the group name:



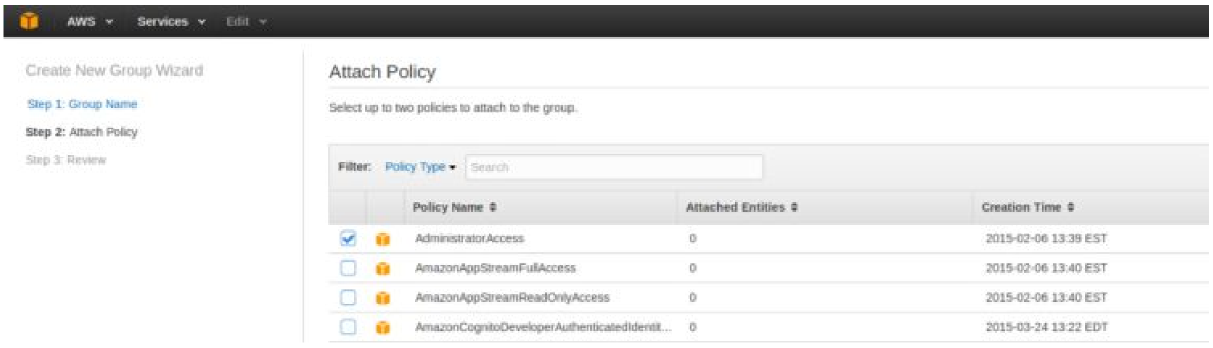
Set Group Name

Specify a group name. Group names can be edited any time.

Group Name: admin

Example: Developers or ProjectAlpha
Maximum 128 characters

Click “Next” and you’ll see a list of preconfigured Admin accounts with varying degrees of access:



Create New Group Wizard

Step 1: Group Name

Step 2: Attach Policy

Step 3: Review

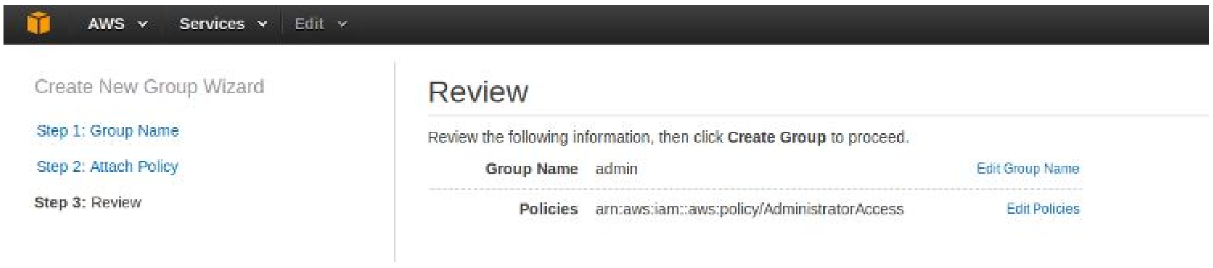
Attach Policy

Select up to two policies to attach to the group.

Filter: Policy Type ▾ Search

		Policy Name ⇕	Attached Entities ⇕	Creation Time ⇕
✓		AdministratorAccess	0	2015-02-06 13:39 EST
		AmazonAppStreamFullAccess	0	2015-02-06 13:40 EST
		AmazonAppStreamReadOnlyAccess	0	2015-02-06 13:40 EST
		AmazonCognitoDeveloperAuthenticatedIdentit...	0	2015-03-24 13:22 EDT

Many of these policies are specialized for specific AWS services. For simplicity choose the first one, “AdministratorAccess” and click “Next Step”.



Create New Group Wizard

Step 1: Group Name

Step 2: Attach Policy

Step 3: Review

Review

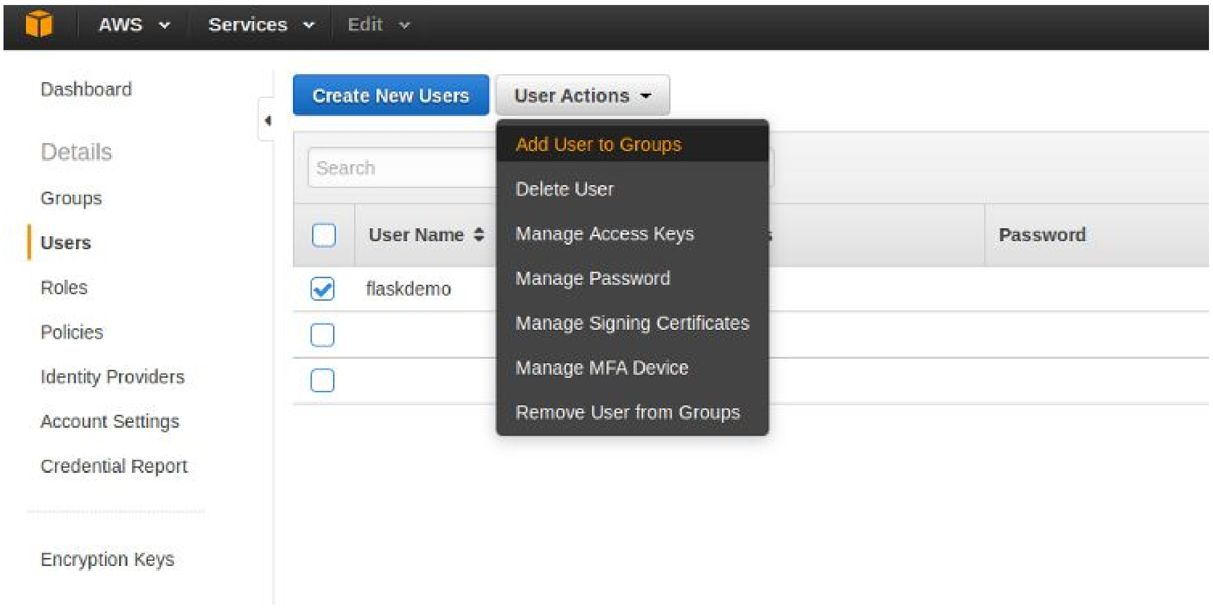
Review the following information, then click **Create Group** to proceed.

Group Name	admin	Edit Group Name
Policies	arn:aws:iam::aws:policy/AdministratorAccess	Edit Policies

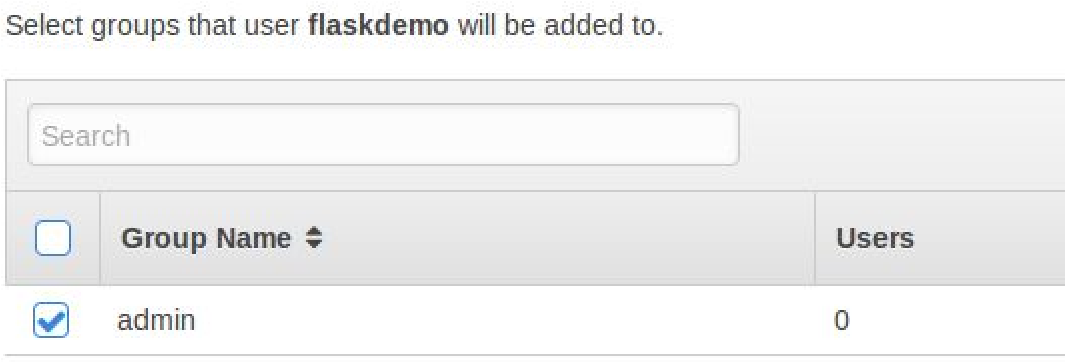
Click “Create Group” and the admin group is now available.

But we’re not done yet. The final step (and one that I sometimes forget) is to assign this group to our user. So back in the IAM main page, select “Users”, and then click the “User

Actions” drop down:



Click “Add User to Groups” and we’ll see our “admin” group:



Select groups that user **flaskdemo** will be added to.

	Group Name	Users
✓	admin	0

Click “Next” and we’ve created an admin.

STEP 5:
Now let's finally initialize our Elastic Beanstalk environment. In your command window, type:

```
$ eb init
```

You'll see:

```
Select a default region
1) us-east-1 : US East (N. Virginia)
2) us-west-1 : US West (N. California)
3) us-west-2 : US West (Oregon)
4) eu-west-1 : EU (Ireland)
5) eu-central-1 : EU (Frankfurt)
6) ap-southeast-1 : Asia Pacific (Singapore)
7) ap-southeast-2 : Asia Pacific (Sydney)
8) ap-northeast-1 : Asia Pacific (Tokyo)
9) sa-east-1 : South America (Sao Paulo)
(default is 3): 1
```

Chose the location closest to you (mine is Northern Virginia). Next you'll be prompted for the AWS ID and Secret Key for the user "flaskdemo" you saved somewhere:

```
You have not yet set up your credentials or your
credentials are incorrect
You must provide your credentials.
(aws-access-id): <enter the 20 digit AWS ID>
(aws-secret-key): <enter the 40 digit AWS secret key>
```

Next you'll see:

```
Select an application to use
1) [ Create new Application ]
(default is 1): 1
```

Next we create the environment name. Hit “Enter” to use the default values:
Enter Application Name
(default is “flask-aws-tutorial”):
Application flask-aws-tutorial has been created.

Now the EBCLI just wants to make sure we’re using Python:
It appears you are using Python. Is this correct?
(y/n): y

Select your Python version. I’m a fan of 2.7 and wrote this example using it, so users of 3+ may find some incompatibilities with this code.
Select a platform version.
1) Python 3.4
2) Python 2.7
3) Python
4) Python 3.4 (Preconfigured — Docker)
(default is 1): 2

You have the option of creating an SSH connection to this instance. We won’t need to use it, so I recommend “no.” (If you need to ssh into this instance later, you can change the preferences of your EC2 instance from the AWS console later.)
Do you want to set up SSH for your instances?
(y/n): n

Okay, now we’re all set up. Time to deploy this bad boy.

Step 6: Deploy our Flask Application

From the command line, type:

```
$ eb create
```

Now we have to create an environment name and DNS CNAME for our app. The domain name is the most important — it will show up in the URL as http://<dns cname>.elasticbeanstalk.com. It has to be unique, or the command line will make your choose another:

```
Enter Environment Name
(default is flask-aws-tutorial-dev):
Enter DNS CNAME prefix
(default is flask-aws-tutorial-dev): thisisacoolflaskapp
```

Once you’ve selected a unique DNS CNAME, you’ll see status updates as the app is deployed. They’ll look like this:

```
WARNING: You have uncommitted changes.
Creating application version archive “c1bf”.
Uploading flask-aws-tutorial/c1bf.zip to S3. This may take a
while.
Upload Complete.
Environment details for: flask-aws-tutorial-dev Application
name: flask-aws-tutorial Region: us-east-1 Deployed
Version: c1bf Environment ID: e-a2ppnms3xj Platform:
64bit Amazon Linux 2015.03 v1.4.1 running Python 2.7 Tier:
WebServer-Standard CNAME:
```

```
thisisacoolflaskapp.elasticbeanstalk.com Updated:
2015–05–28 16:20:28.363000+00:00
Printing Status:
INFO: createEnvironment is starting.
INFO: Using elasticbeanstalk-us-east-1–611827146380 as
Amazon S3 storage bucket for environment data.
INFO: Created load balancer named:
awseb-e-a-AWSEBLoa-1DZU2FYBGSE55
```

Etc…
When the uploading finishes, you'll see:

```
INFO: Application available at
thisisacoolflaskapp.elasticbeanstalk.com.
INFO: Successfully launched environment:
flask-aws-tutorial-dev
```

Point your web browser to that URL and you'll see your Flask app live!

Or enter

```
$ eb open
```