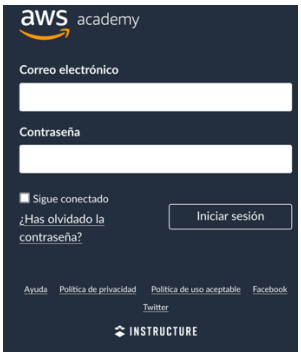
Grau Bioinformàtica
Curs 2024-2025
Distributed systems and web development
Django lab – AWS Elastic Beanstalk
Based in AWS Deploying a Django application with Elastic Beanstalk
https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create-deploy-python-django.html

1. Work environment preparation

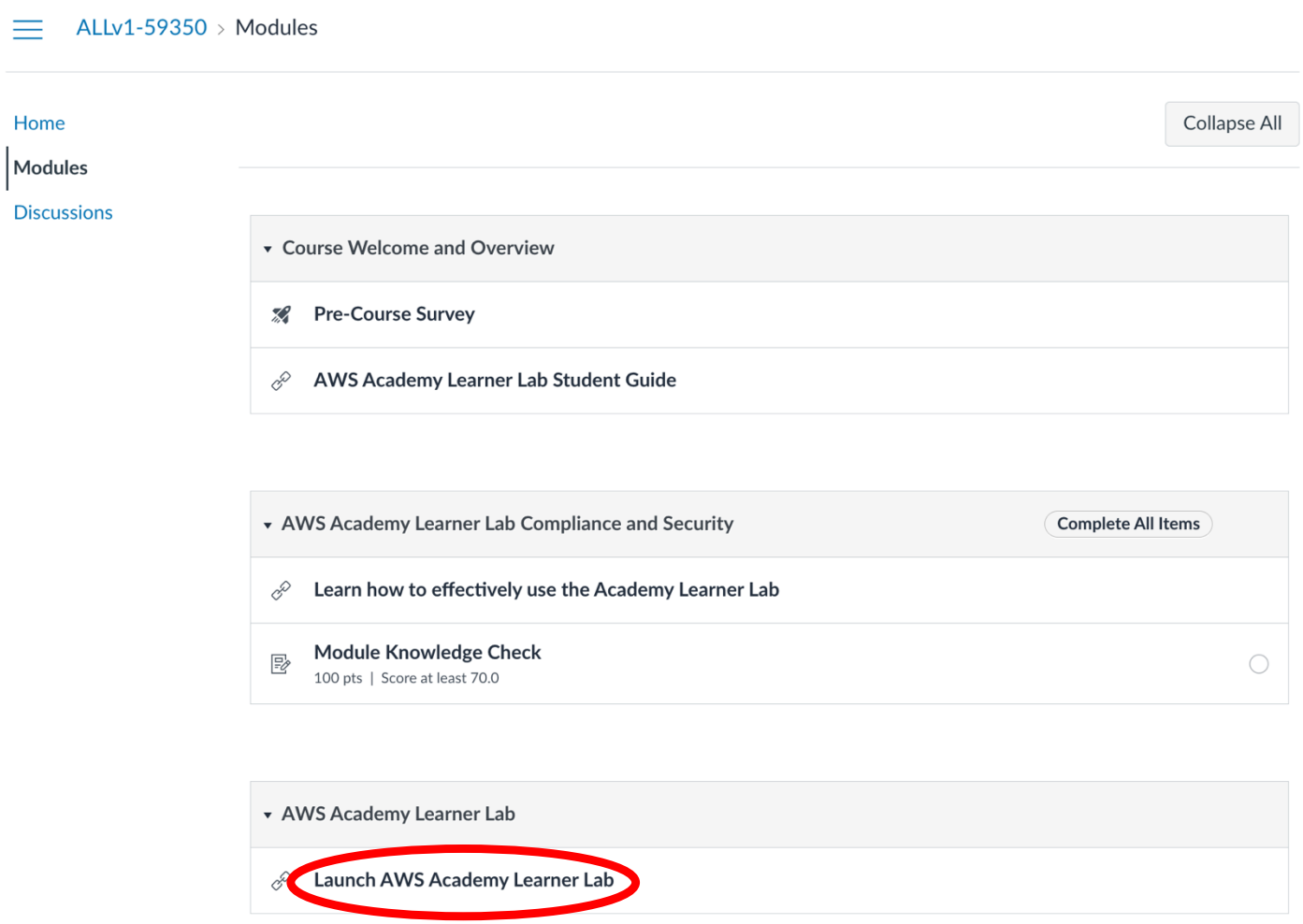
This tutorial will help you to create a simple Django website and run it in a cloud AWS Elastic Beanstalk environment running Python. We are going to create a local application and then deploy it in an AWS cloud service.

Check your inbox to find an invitation for AWS Academy courses. You will need to set up a canvas account, then you can login to AWS Academy services by using your e-mail and password:

https://awsacademy.instructure.com/login/canvas

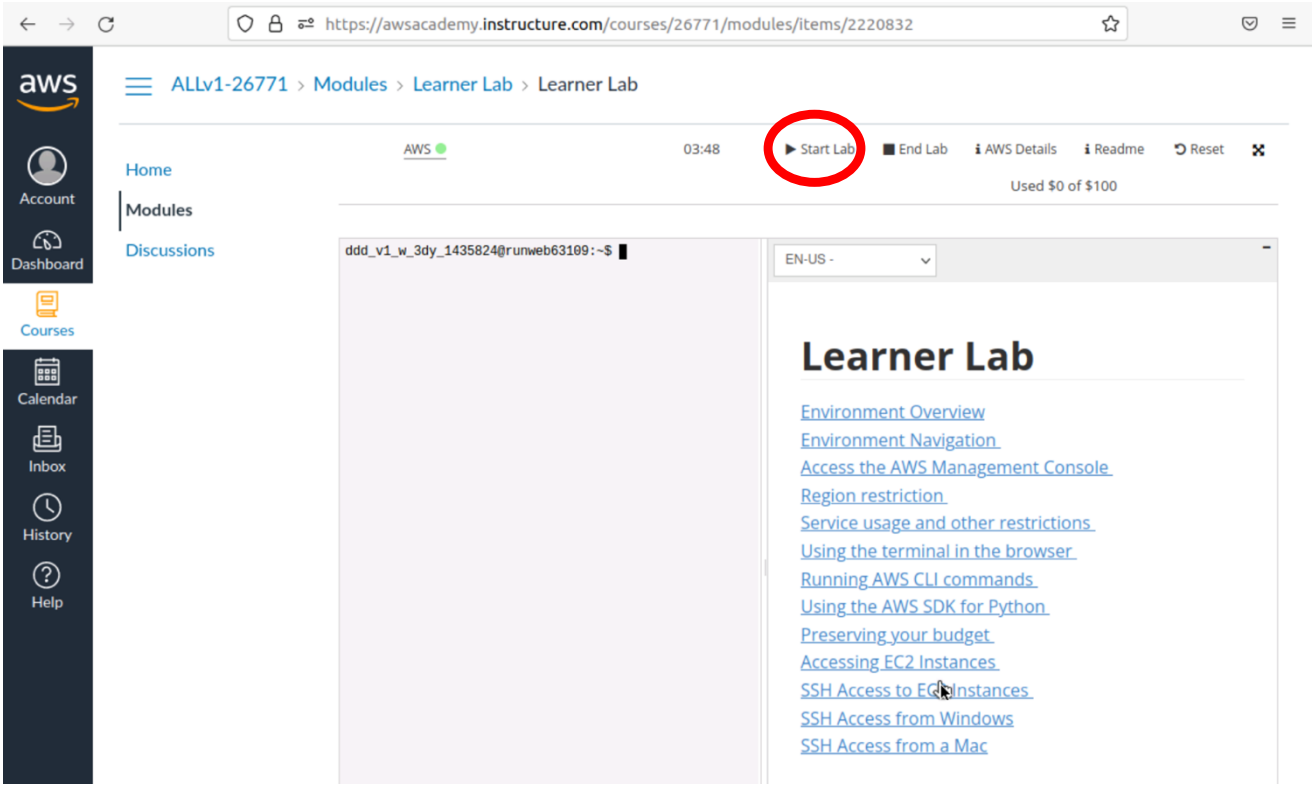


Enter the AWS Academy Learner Lab course and go to modules option in the left menu:

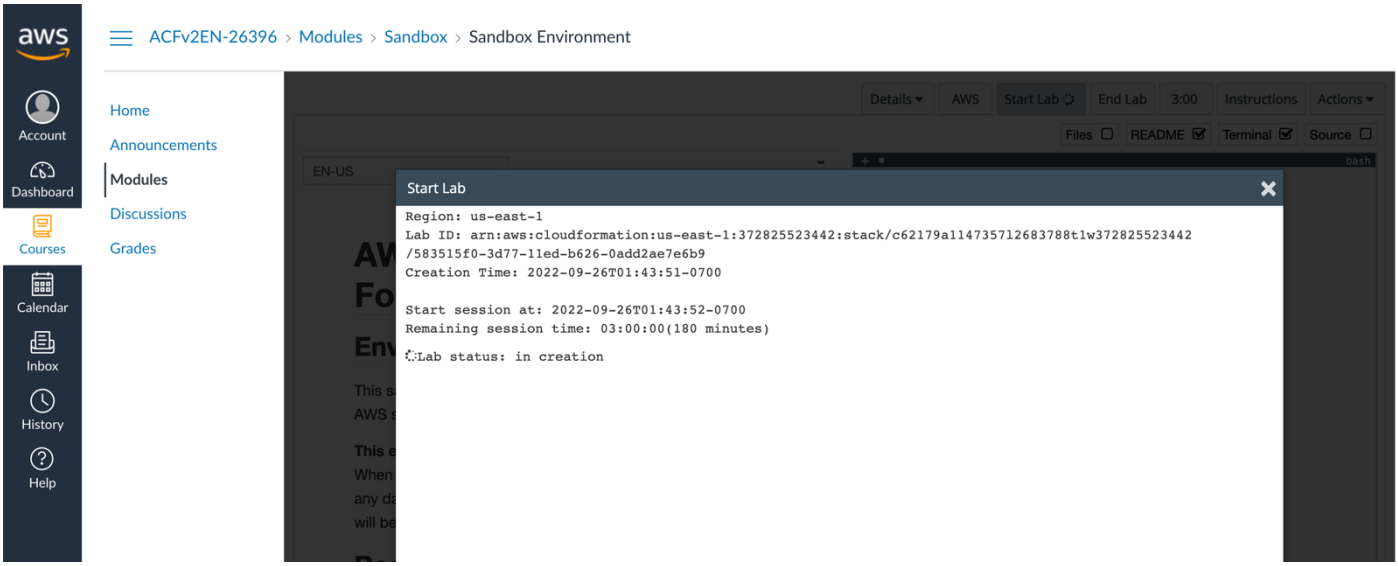


Scroll down to the bottom of the list of modules until you find the AWS Academy Learner Lab and open it. You will see a welcome page to Learner Lab Sandbox environment overview.

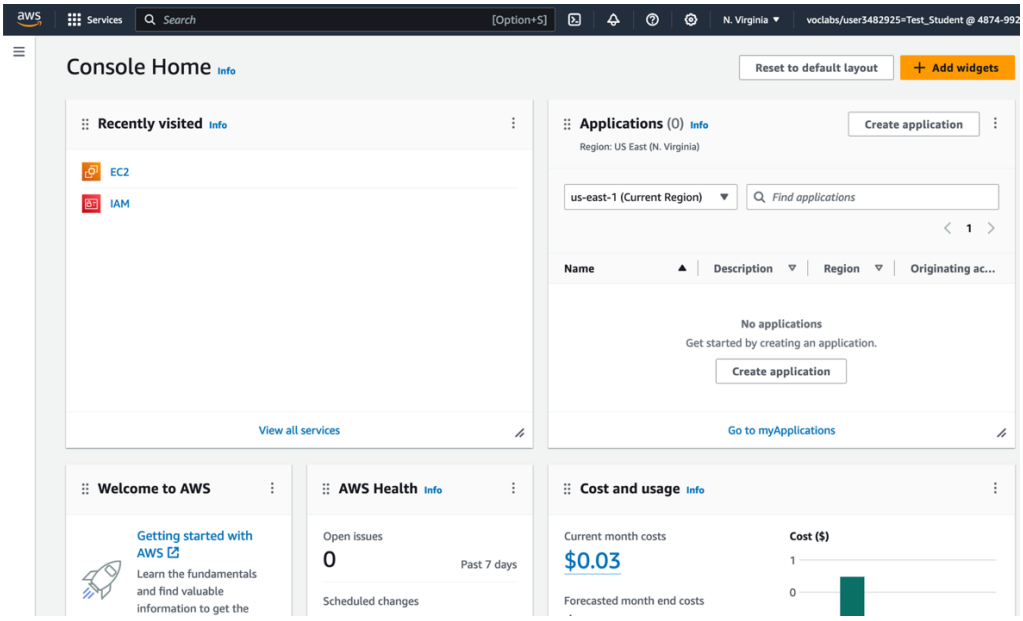
Agree with the terms and your lab environment Lab should be ready to be used. You just need to start it.



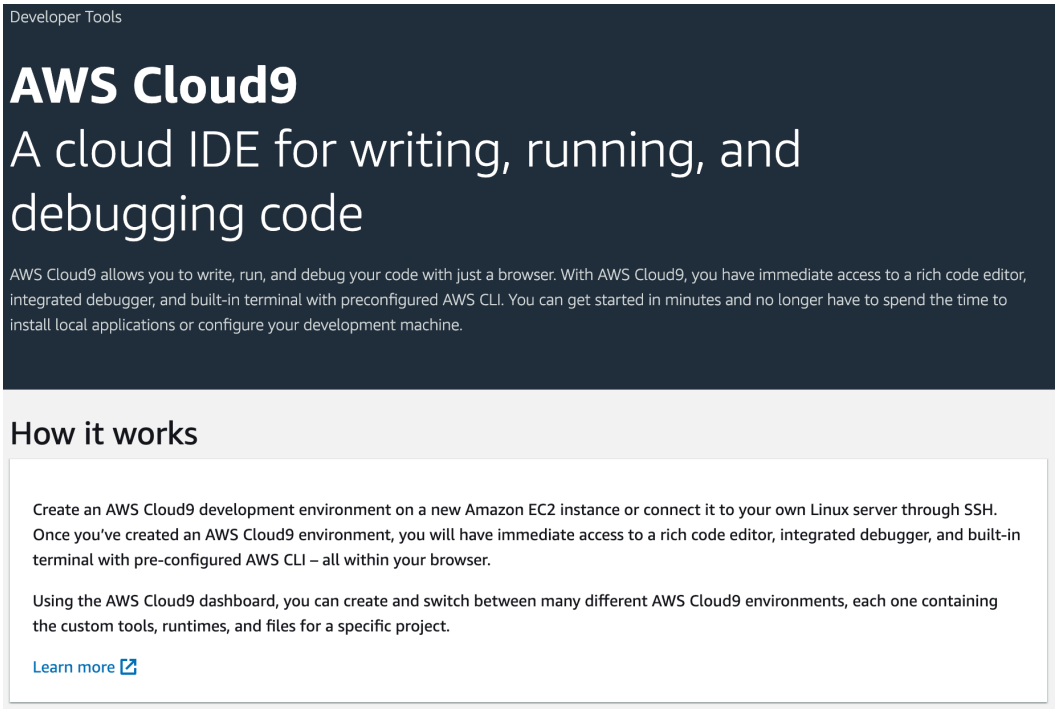
Use the Start lab button and wait for AWS infrastructure to be built (some minutes). AWS lab color will change from red to green. You will see that there is a timer set at 4:00 hours.



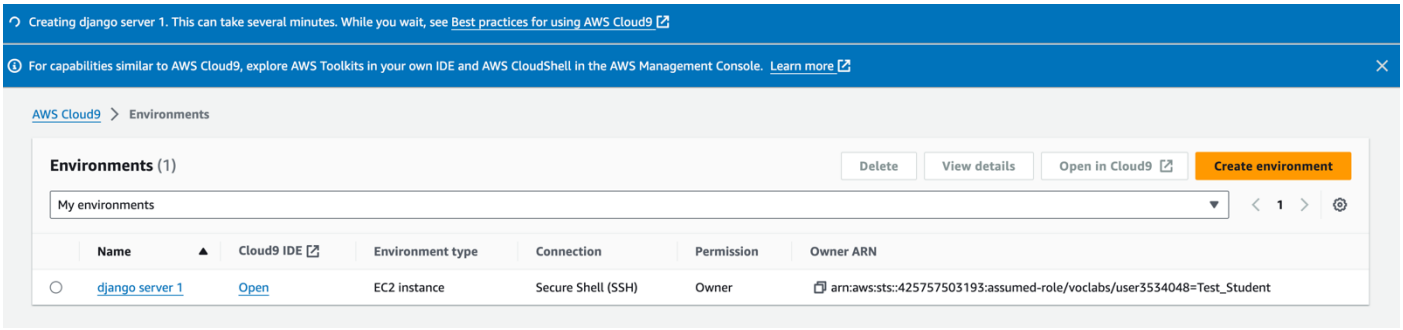
Use the AWS green bullet to enter the AWS services console home:



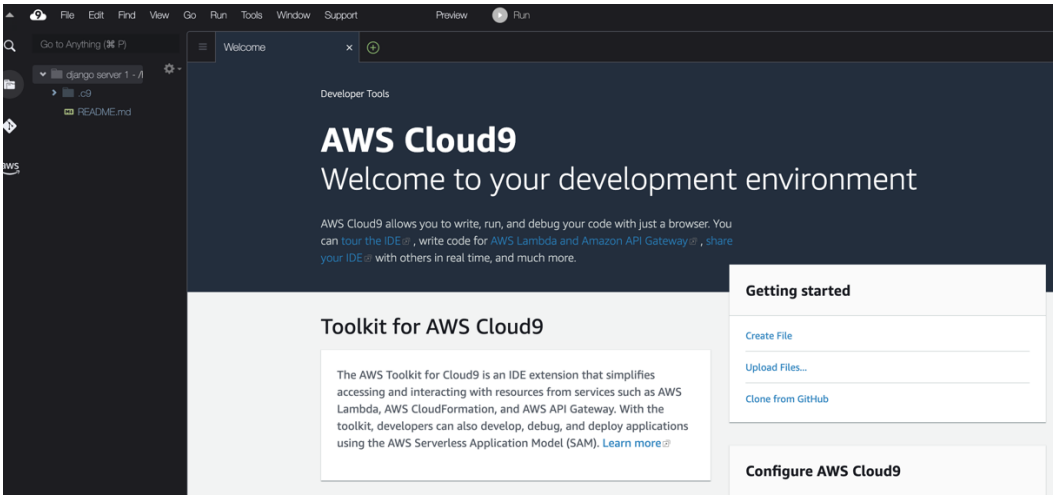
Now you must search for cloud9 IDE environment to access AWS Cloud9:



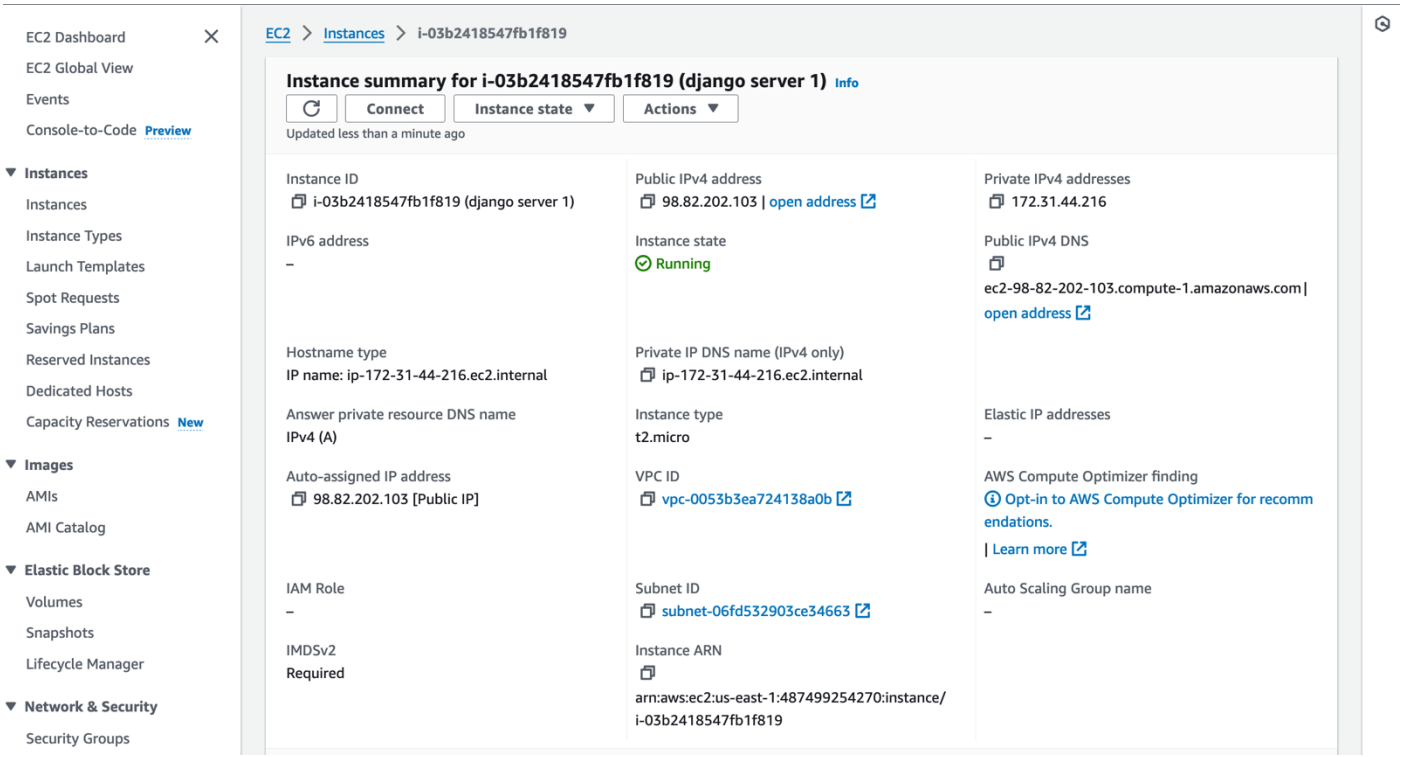
Find "Create environment" yellow button and use it to create a new working environment providing a name: "django server 1" and selecting "secure shell (SSH)" option in Network settings and then clicking on create button.



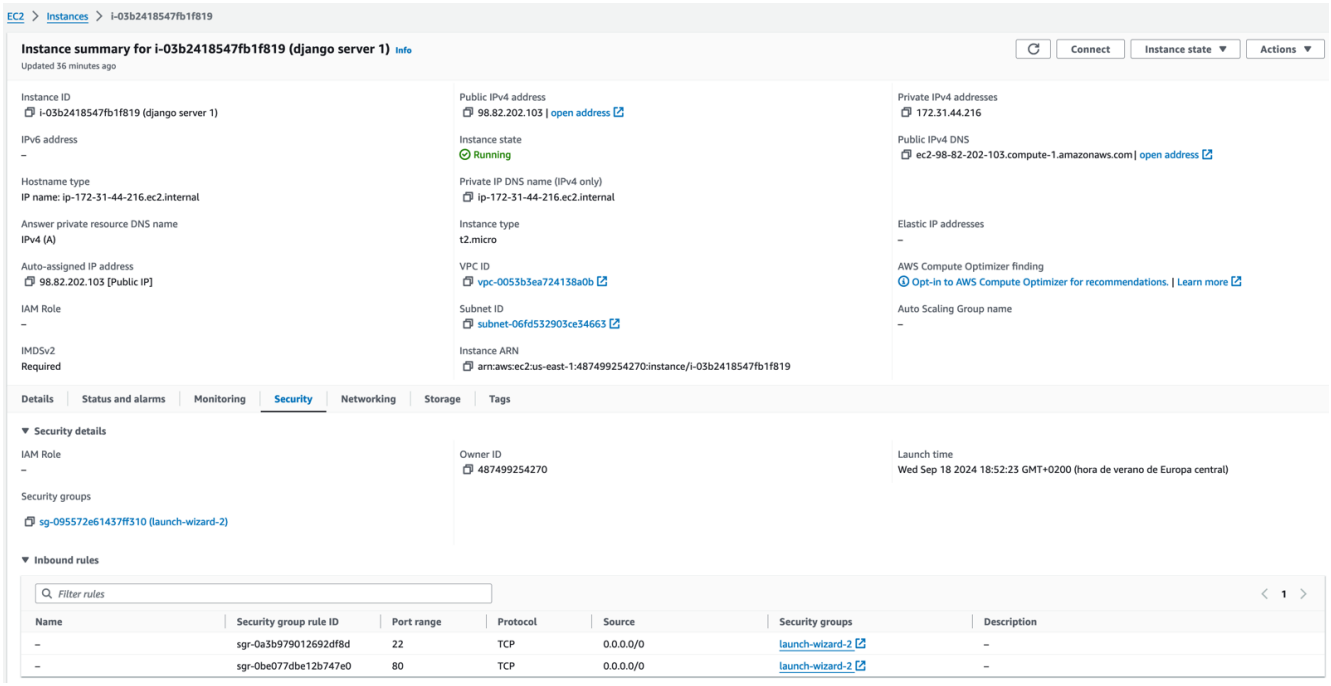
After some minutes of creation and configuration, we will have our new Django server environment. To open it, use "Open in Cloud9" button. A new IDE welcome page will be opened



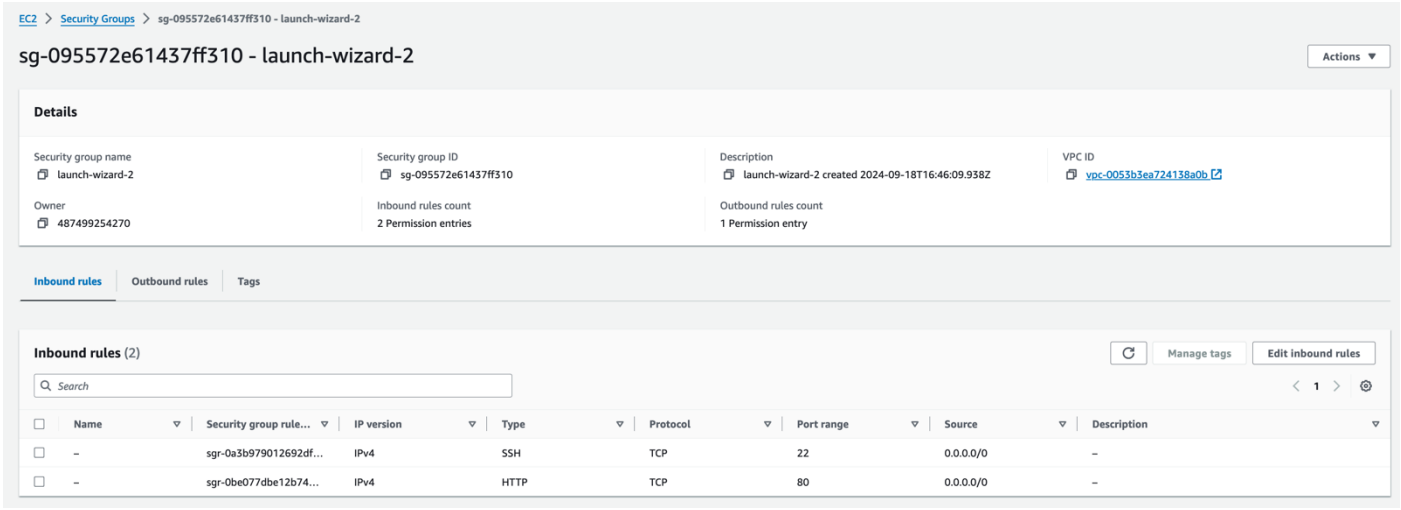
Go back to AWS services console home and search for **EC2** instances. You will find a new instance that is running. Please click on instance ID blue link to review its details:



Please write down your public IPv4 address for later use. In our example this address is 98.82.202.103. Now, we need to create an inbound rule to allow web traffic coming in. Click on the **Security** tab, then in **Security groups**

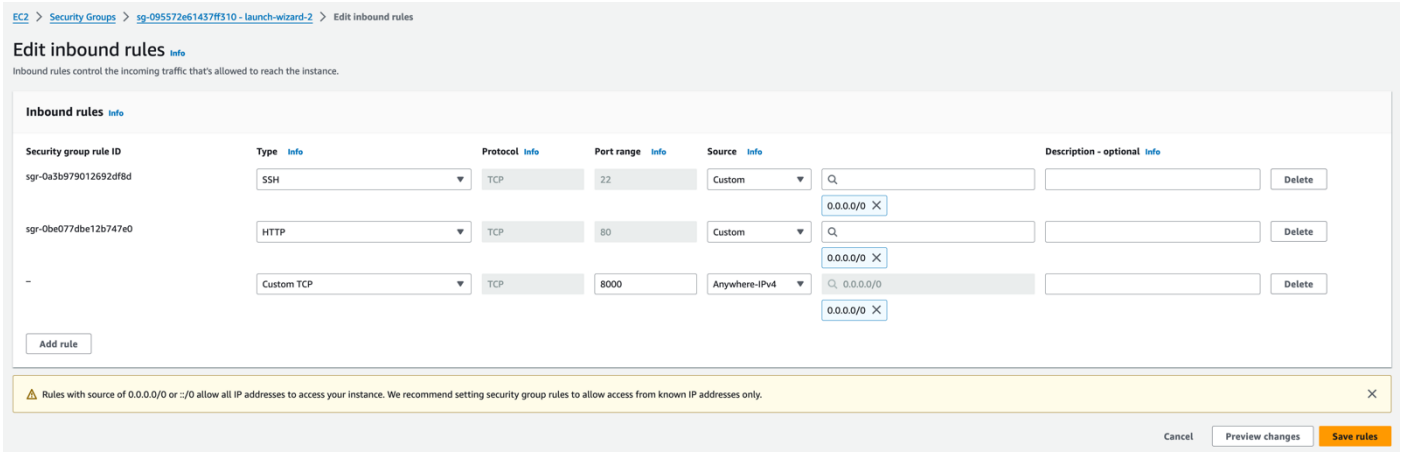


Click on **Edit inbound rules** box



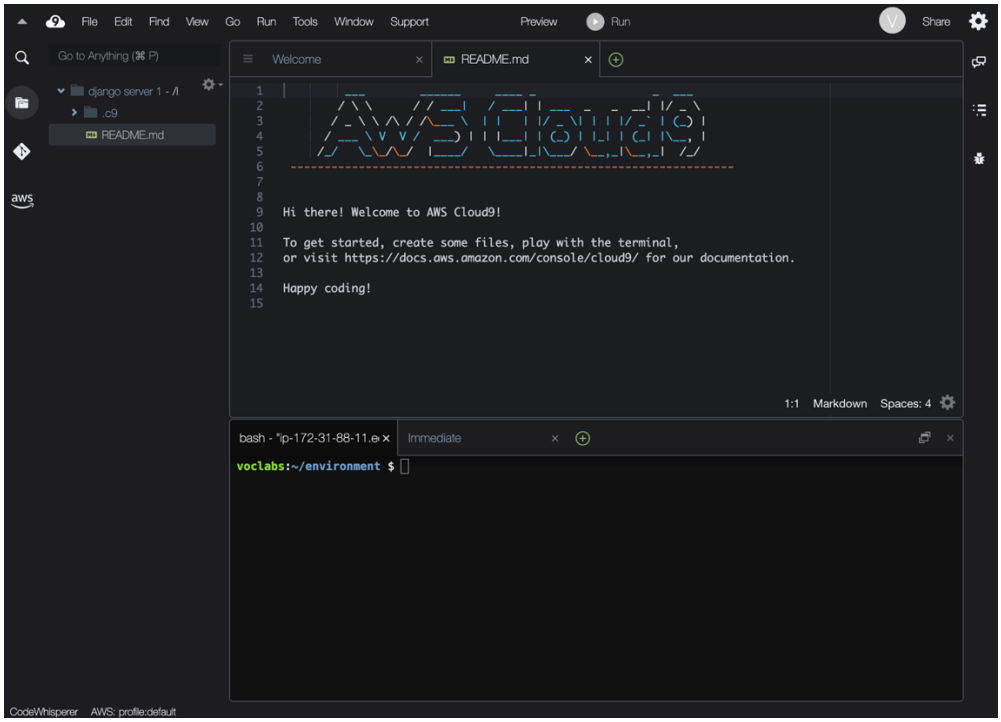
Now **Add a new Custom TCP rule** with the following parameters,
Type: Custom TCP
Port range: 8080
Source: Anywhere-IPv4, 0.0.0.0/0

and then press **Save rules**


EC2 > Security Groups > sg-095572e61437ff310 - launch-wizard-2 > Edit inbound rules
Edit inbound rules Info
Inbound rules control the incoming traffic that's allowed to reach the instance.
Inbound rules Info
Security group rule ID
Type Info
Protocol Info
Port range Info
Source Info
Description - optional Info
sgr-0a3b979012692df8d
SSH
TCP
22
Custom
0.0.0.0/0
Delete
sgr-0be077dbe12b747e0
HTTP
TCP
80
Custom
0.0.0.0/0
Delete
–
Custom TCP
TCP
8000
Anywhere-IPv4
0.0.0.0/0
0.0.0.0/0
Delete
Add rule
Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.
Cancel
Preview changes
Save rules

2. Django environment creation

Go back to your AWS Cloud 9 development environment tab in your browser to create a new Django application. Check that the browser window has a file navigation, a file viewer and a console window in the bottom



To use Django, we first need to install Python and then install Django 2.2 distribution using pip installer. In our case we are going to use a Cloud9 environment Linux system. Use the bottom terminal and type the commands to create a Django application environment:

```
cd /home/ec2-user/environment
sudo yum install python-is-python3 -y
sudo yum install pip -y
pip install virtualenv
```

In any case, you should install AWS terminal client:

```
pip install awsebcli
```

Second step: prepare a new Python application virtual environment. In this case, it will be named eb-virt:

```
virtualenv eb-virt

source eb-virt/bin/activate

pip install django==2.2
```

Finally, we create a new local Django application using the templates

```
django-admin startproject ebdjango

results are standard django site name ebdjango with this structure:
/home/alumno/ebdjango
  |-- ebdjango
  |   |-- __init__.py
  |   |-- settings.py
  |   |-- urls.py
  |   `-- wsgi.py
  `-- manage.py
```

Open ebdjango/ebdjango/settings.py file in Cloud9 IDE and edit ALLOWED_HOSTS line so that it contains this configuration:

```
ALLOWED_HOSTS = ['*',]
```

So, let's run our local server and check that everything is working fine

```
cd /home/ec2-user/environment/ebdjango

python manage.py runserver 0:8080
Django version 2.2, using settings 'ebdjango.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
[07/Sep/2018 20:14:09] "GET / HTTP/1.1" 200 16348
```

Start a web browser and open **http://<your ip here>:8080** to view the created site.

Use Ctrl+C in terminal to stop the web server and return to your virtual environment.

3. Configuration of Django application for AWS Elastic Beanstalk

Our application is ready to be executed in the cloud. AWS has a special service for running Django applications that will start a list of services that our application needs, then we will send our application code so that it is available for the users.

```
cd /home/ec2-user/environment/ebdjango
pip freeze > requirements.txt
```

Create a new directory named `.ebextensions` and create a new `django.config` file inside it.

```
mkdir .ebextensions
cd .ebextensions
```

This file sets WSGIPath, which specifies the location of the WSGI script that Elastic Beanstalk will use to start the application.

Create a new file:

/home/ec2-user/environment/ebdjango/.ebextensions/django.config.

Please follow the text indenting of the example and do not use TAB, just blank spaces

```
nano /home/ec2-user/environment/ebdjango/.ebextensions/django.config

option_settings:
  aws:elasticbeanstalk:container:python:
    WSGIPath: ebdjango.wsgi:application
```

Finally, we deactivate the virtual environment as we have finished the configuration steps

```
deactivate
```

We will need to activate this environment again if we need to add changes to our application or run it locally.

4. AWS Elastic Beanstalk deployment with EB CLI

Your **ebdjango** application directory should look like this:

```
cd /home/ec2-user/environment/ebdjango
ls -la

/home/alumno/ebdjango/
|-- .ebextensions
|   `-- django.config
|-- ebdjango
|   |-- __init__.py
|   |-- settings.py
|   |-- urls.py
|   `-- wsgi.py
|-- db.sqlite3
|-- manage.py
`-- requirements.txt
```

Now we are going to use a command line interface *(eb)* to create a new cloud application environment and then deploy our application to run in the cloud with Elastic Beanstalk. If you make a mistake in the configuration you will have to eliminate the environment (step 7) and start again.

1. Initialize your EB CLI repository with **eb init** command in your local terminal:

```
eb init -i

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 : Europe (Ireland)
5) eu-central-1 : Europe (Frankfurt)
6) ap-south-1 : Asia Pacific (Mumbai)
7) ap-southeast-1 : Asia Pacific (Singapore)
...
(default is 3): 1

Enter Application Name:
(default is "ebdjango"):
```

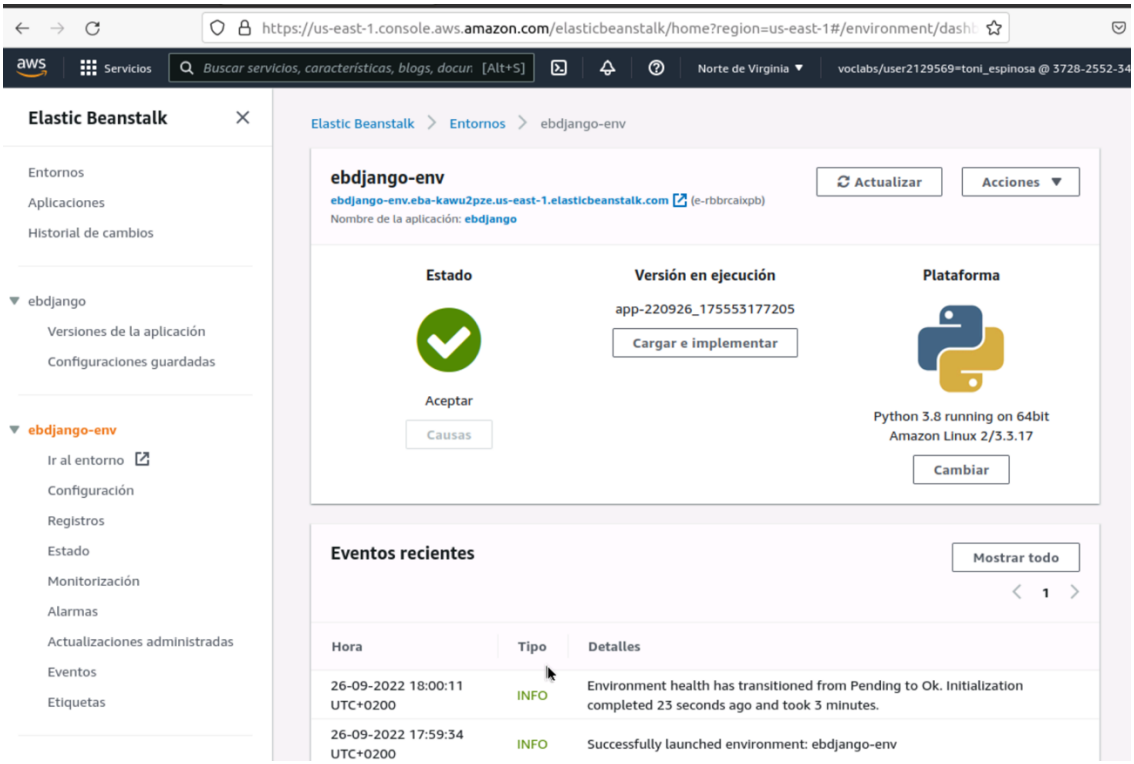
```
It appears you are using Python. Is this correct?
(y/n): y

Select a platform branch.
1) Python 3.11 running on 64 bit Amazon Linux 2023
2) Python 3.9 running on 64 bit Amazon Linux 2023
3) Python 3.8 running on 64 bit Amazon Linux 2
4) Python 3.7 running on 64 bit Amazon Linux 2 (Deprecated)
Linux (Deprecated)
(default is 1): 3

Do you want to set up SSH for your instances?
(y/n): y

Select a keypair.
1) vockey
2) [ Create new KeyPair ]
(default is 1): 1
```

2. Create an environment and deploy your application with eb create

```
eb create ebdjango-env --service-role LabRole -ip LabInstanceProfile  --max-instances 1
```

This command creates a load-balanced Elastic Beanstalk environment named **django-env**. Creating an environment takes about 5 minutes. As Elastic Beanstalk creates the resources needed to run your application, it outputs informational messages that the EB CLI relays to your terminal.

Please wait for your environment to be created. You can also use your AWS Dashboard to check **Elastic Beanstalk services state**. Use AWS button from your sandbox application.



3. When the environment creation process completes, find the domain name of your new environment by running eb status.

```
eb status
Environment details for: django-env
  Application name: django-tutorial
  ...
Status: Ready
Health: Green
```

Your environment's domain name is the value of the CNAME property. You can also search your AWS services for Elastic Beanstalk environments.

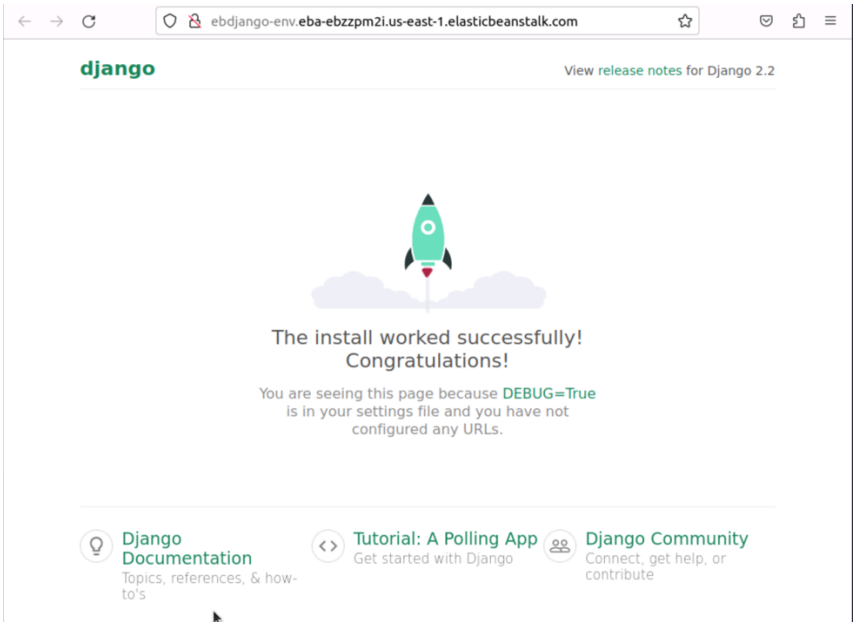
4. Save the file, and then deploy your application by running eb deploy. When you run **eb deploy**, the EB CLI bundles up the contents of your project directory and deploys it to your environment.

```
eb deploy
Creating application environment...
...
INFO Environment update completed succesfully.
```

5. When the environment update process is finished, you can open your website with **eb open**

```
eb open
```

6. When everything works, you should see the welcome to django page in your **ElasticBeanstalk environments, do not open EC2 instances at this point.**



7. When you finish, to clean up the system, terminate your Elastic Beanstalk with eb terminate

```
eb terminate ebdjango-env
```

and wait for the environment to close with the message

terminateEnvironment completed successfully

5. Task: deploy your polls app

Can you deploy your django polls application from the previous tutorial into AWS Elastic Beanstalk following the same steps described in section 4?

You will need to prepare polls application first:

- Create and activate your *mysite* environment from the *polls* tutorial
    - django-admin startproject mysite
    - <copy the last version of your code>
    - python manage.py migrate
    - create a question and some options with */admin*

- Generate *requirements.txt* file for your *mysite* project
- Create *.ebextensions/django.config* file:

```
option_settings:
  aws:elasticbeanstalk:container:python:
    WSGIPath: mysite.wsgi:application
```

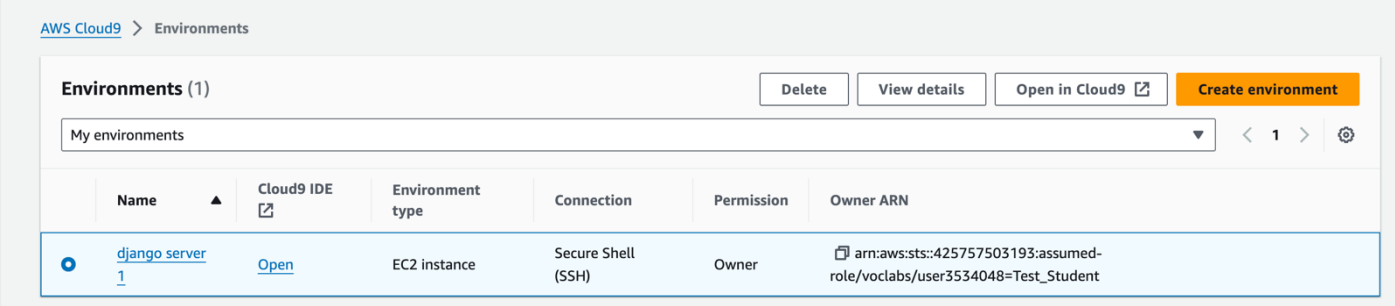
Deployment errors can be checked in file: **web.stdout.log**. This file can be downloaded from AWS Console, Elastic Beanstalk service, polls-env environment logs.

6. Cleaning the environment and closing the Sandbox session

Please remember to eliminate *polls-env* environment with the command:

```
eb terminate polls-env
```

And close your AWS Cloud9 environment before you finish. Select your environment in the AWS services console and press the **Delete option**



Then, type **Delete** and click on the delete option to remove the environment.

Now, you need press **End Lab** in the AWS Academy learner lab and logout of AWS Academy portal.