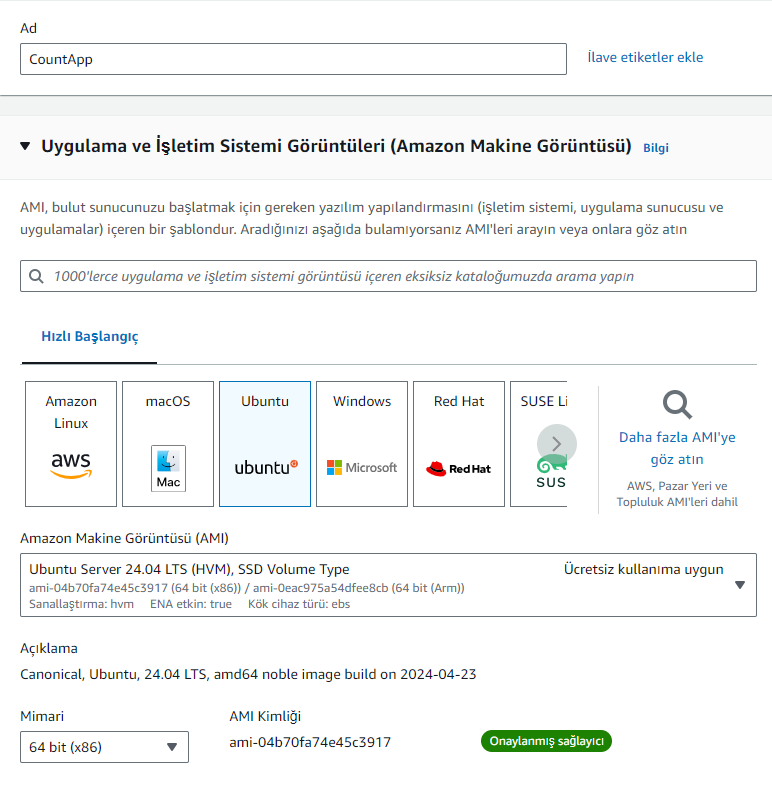
**AWS Cloud Computing Project**  
**Word Counting App with PuTTY & Django Configuration**  
**Student 1: Ali Cemal GÜLMEZ - 211805078**  
**Student 2: Engin Halil YEDİRMEZ - 211805118**

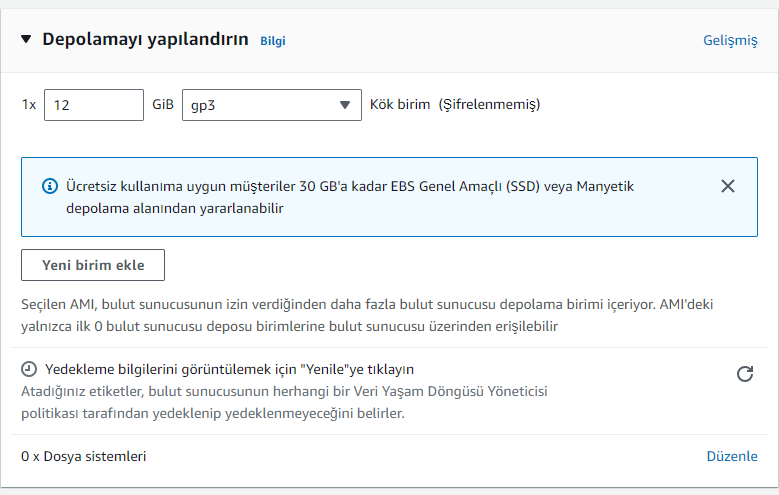
Hello. This is our guide to creating a Django project with PuTTY and running it on Amazon AWS. Lets start.

We log in to AWS Academy and press the Start Lab button. To understand that our lab has started, we wait for the red dot next to the AWS text in the upper left to first turn yellow and then green. When it turns green, it means our Lab has started.

After our lab starts, we click on the AWS text on the top left and go to the AWS Control Management section. In this section, we enter the Cloud Servers section. Then, in the Cloud Servers section, we will click on the Launch an Instance section.

There are settings we need to make in the Launch an Instance section. These are, respectively:   
  


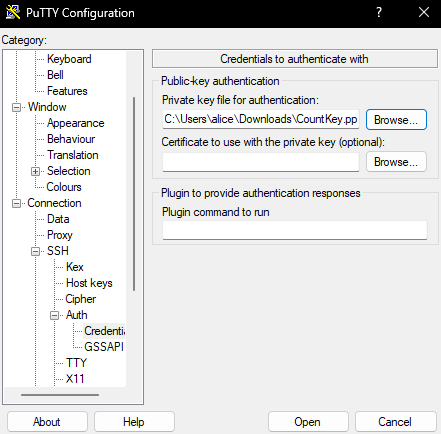
1) We choose a name for our project. We will choose the name Launcher.   
2) We will select Ubuntu in Amazon Machine Image. Our platform will be Ubuntu. After checking that the architecture is the default (64 bit x86), we move directly to the Key Pair section.   
3) In the Key Pair section, click on the "Create a new key pair" button on the right.   
4) We enter a name for our key pair. We will enter the Launcher name again. We choose our key pair type as RSA and our file format as .ppk. Since we will be processing via PuTTY, it is important to choose .ppk. When we register, AWS automatically downloads our key.   
5) We go to network settings, click the Edit button at the top. We add the necessary security rules and complete this stage.   
6) In the storage configuration section, we change the GP3 from 1x8 to 1x12 and install it without going into advanced settings. We can start our server by pressing the start cloud server button on the right.



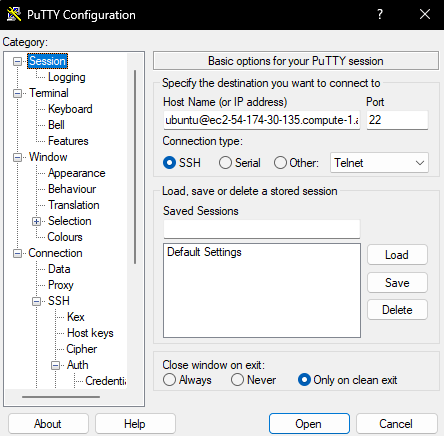
Our server is simply completed. Now we download PuTTY from a reliable source, install it and run it.

PuTTY's main screen appears. What we need to do in this context:

1) We copy the Public IPv4 DNS address of our cloud server, add "ubuntu@" at the beginning and paste it into the Host Name section of PuTTY's Session section. In this context, ours will be "ubuntu@ec2-18-207-155-122.compute-1.amazonaws.com". Our port remains 22 and our connection type remains SSH and Telnet.



2) We press the plus next to SSH under the Connection section on the left. Then we press the plus next to the Auth section and enter the Credentials section. Here, we click Browse right next to the phrase "Private key for authentication" and find the key we downloaded and bring it here. Then we click Open and we are done. You can continue by clicking Accept on the screen that appears.



If PuTTY gives a Fatal Error and does not connect to the server, we go to the Instances section, find our server and open a port 22 for SSH. On top of that, if you need to turn on Telnet services on your computer, turn them on. Disable the firewall in Windows.

Now PuTTY is now connected to our server. We should bring Python and Django to Ubuntu. First of all, for Python we say "$sudo apt install python3". Then we say "$sudo apt install python3-pip".

Here we often get the following error when installing pip:

"Package python3-pip is not available but is referred to by another package.

This may mean that the package is missing, has been obsoleted, or

is only available from another source

E: Package 'python3-pip' has no installation candidate"

We are developing a solution to this situation by writing the following codes:

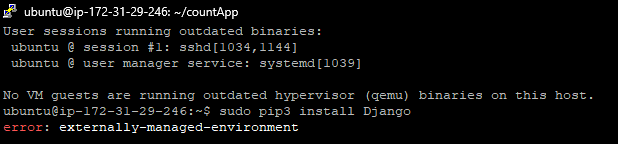
$sudo apt-get update

$sudo apt-get upgrade (Optional)

$sudo apt install python3-pip

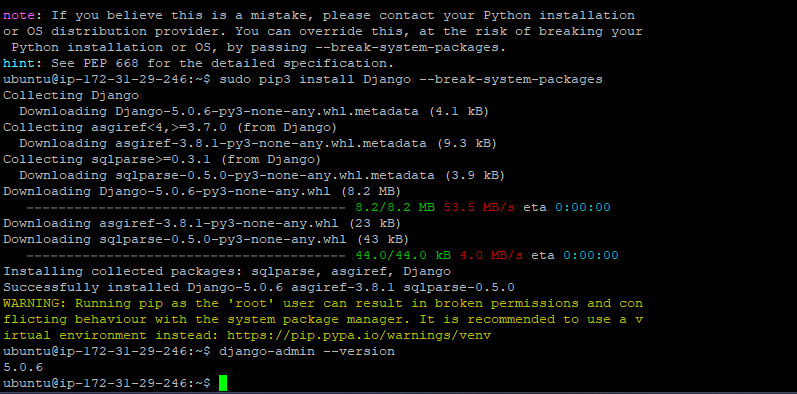
We have Python installed. Django is next. We say "$sudo pip3 install Django" and Django takes its place in our project.

If we get an error like "error: externally-managed-environment" when installing Django, we can add the pip argument to our download command.



We update it as "$sudo pip3 install Django --break-system-packages".

You can run the code "$django-admin --version" to see if Django is installed. As you can see, we have version 5.0.6 available.



We installed Django. Now we will need to create a directory for our application. In this context, we create a directory by typing mkdir "~/wordCountApp" and enter the directory of our project by doing "cd ~/wordCountApp".

We need to start a project within the directory. To start a project in Django, we need to write our code as "$django-admin startproject myproject". We can write our project name instead of "myproject", we will write wordCountApp here.

If we want, after our project starts, we can see the name of our project by typing ls to check that our project is included in the directories.

We're inside our wordCountApp directory, but not yet inside our Django application. In this context, we say "cd wordCountApp" again and enter our Django application. Then, we create another directory within our application by typing "python3 manage.py startapp wordcount".

We need to make modifications to the settings.py file in our wordCountApp directory. For this, we write "nano wordCountApp/settings.py" and enter our settings file. We add our application wordcount to the INSTALLED\_APPS list with the 'wordcount' syntax and put a comma (,) at the end. Then we go directly to the ALLOWED\_HOSTS section and add the addresses we can connect to, including localhost. We chose ['localhost', '127.0.0.1', '52.90.223.173', '0.0.0.0', 'ec2-52-90-223-173.compute-1.amazonaws.com'].



Next up we have URL edits. To do this, we visit the file by typing "nano wordCountApp/urls.py". We design the code inside as in the image.



After editing the URL, we type "nano wordcount/urls.py" to create our own URL design for our wordcount application. Then, we do our coding as follows. Thus, we introduced our application to the URL section.

Our application must have a views file. In this context, to create it, we typed "nano wordcount/views.py" into the terminal, and added our view function and adjusted the code as shown in the image.



Next up are the HTML pages our site needs to display. To do this, we will first create a subdirectory called templates within our application directory and create HTML paths in it. We write "mkdir -p wordcount/templates/wordcount". Then, we create the wordcount.html file by typing "nano wordcount/templates/wordcount/wordcount.html" in "templates".

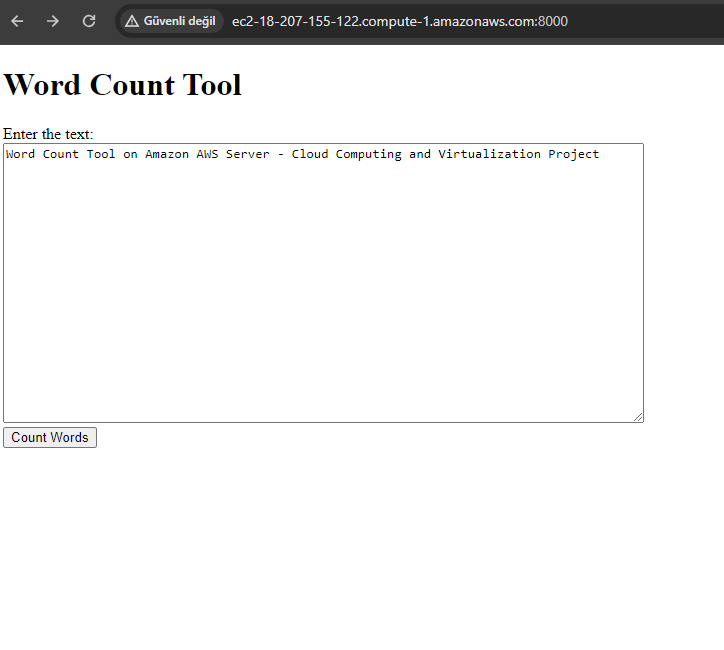


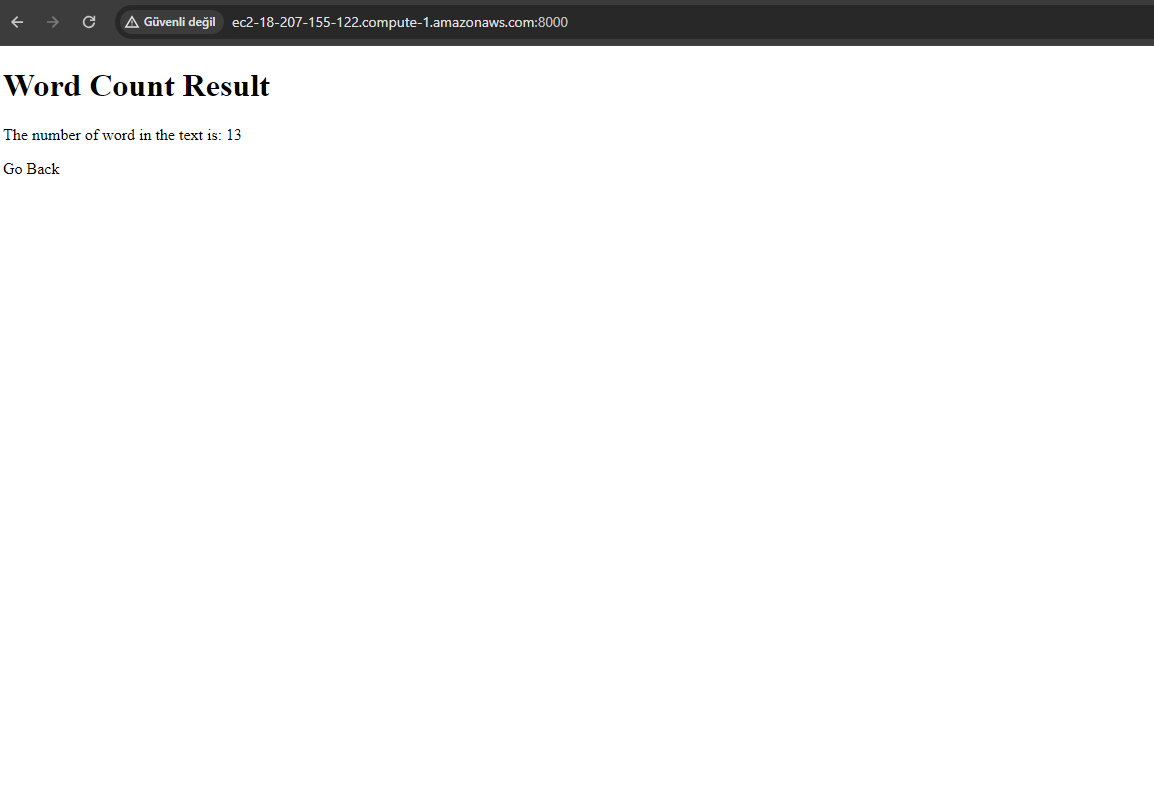
Here we write the necessary functions for the general view. We left an attachment in the image.

Next, we create a "result.html" file for our screen design where we need to display the result after word counting. In this context, it is enough to type "nano wordcount/templates/wordcount/result.html" into the terminal. We write our commands inside as follows.



In general, our transaction is completed as of now. When we type "python3 manage.py runserver 0.0.0.0:8000" in the console and go to "http://ec2-18-207-155-122.compute-1.amazonaws.com:8000/" from our browser, we see the screen of our application.





Specific problems we encountered:

1) No Module Named '...' problem

We received potential errors like NoModuleNamed 'wordcountdjango' several times because we did not include comma (,) where required. We achieved the solution by adding commas where necessary.

2) ImportError: cannot import name 'views' from 'wordCountApp' (/home/ubuntu/wordCountApp/wordCountApp/wordCountApp/\_\_init\_\_.py) error

This error occurs because there is no \_\_init\_\_.py file. The solution is to create the init.py file in the correct directory. We saw that the problem can be solved by typing "touch wordcount/\_\_init\_\_.py" into the console. When the problem occurs, it can be solved by changing the "from . import views" line in "urls.py" to "from wordcount import views".

3) Problem exporting the project

WinSCP, a third-party software, was sufficient to export the project. We installed WinSCP from reliable sources. We chose the file protocol SCP. We entered the server name as the same as in PuTTY (ubuntu@ec2-18-207-155-122.compute-1.amazonaws.com) and kept our port at 22. We clicked on the Advanced section, selected the "Authentication" section under SSH and entered our key. In this way, we had the opportunity to access our files in the cloud and download a copy.

That was it for our guide. Thanks.