

[Open in app](#)

Published in Towards Data Science



Rahul Agarwal

[Follow](#)

Dec 11, 2019 · 7 min read · ✨ · ⏰ Listen

[Save](#)

...

Image by [Gerd Altmann](#) from [Pixabay](#).

How to Deploy a Streamlit App using an Amazon Free ec2 instance?

Data Apps on the web in 10 minutes

A Machine Learning project is never really complete if we don't have a good way to showcase it.

While in the past, a well-made visualization or a small PPT used to be enough for showcasing a data science project, with the advent of dashboarding tools like RShiny and Dash, a good data scientist needs to have a fair bit of knowledge of web frameworks to get along.

And Web frameworks are hard to learn. I still get confused in all that HTML, CSS, and Javascript with all the hit and trials, for something seemingly simple to do.

Not to mention the many ways to do the same thing, making it confusing for us data science folks for whom web development is a secondary skill.

This is where *StreamLit* comes in and delivers on its promise to create web apps just using Python.

In my [last post on Streamlit](#), I talked about how to write Web apps using simple Python for Data Scientists.

But still, a major complaint, if you would check out the comment section of that post, was regarding the inability to deploy Streamlit apps over the web.

And it was a valid complaint.

A developer can't show up with his laptop every time the client wanted to use the app. What is the use of such an app?

So in this post, we will go one step further deploy our Streamlit app over the Web using an Amazon Free ec2 instance.

Setting up the Amazon Instance

Before we start with using the amazon ec2 instance, we need to set one up. You might need to sign up with your email ID and set up the payment information on the [AWS website](#). Works just like a simple sign-on. From here, I will assume that you have an AWS account and so I am going to explain the next essential parts so you can follow through.

- Go to AWS Management Console using <https://us-west-2.console.aws.amazon.com/console>.
- On the AWS Management Console, you can select “Launch a Virtual Machine”. Here we are trying to set up the machine where we will deploy our Streamlit app.
- In the first step, you need to choose the AMI template for the machine. I select the 18.04 Ubuntu Server since it is applicable for the Free Tier. And Ubuntu.

[1. Choose AMI](#) [2. Choose Instance Type](#) [3. Configure Instance](#) [4. Add Storage](#) [5. Add Tags](#) [6. Configure Security Group](#) [7. Review](#)

Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Cancel and Exit

ubuntu

Quick Start (8)

My AMIs (0)	AWS Marketplace (271)	Community AMIs (33638)	Free tier only (1)
			<input checked="" type="checkbox"/> Free tier eligible

Ubuntu Server 18.04 LTS (HVM), SSD Volume Type - ami-06d51e91cea0dac8d (64-bit x86) / ami-02cbcd67225579b2c (64-bit Arm)

Ubuntu Server 18.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Ubuntu Server 16.04 LTS (HVM), SSD Volume Type - ami-0994c095691a46fb5 (64-bit x86) / ami-033a024887b09d8a8 (64-bit Arm)

Ubuntu Server 16.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Deep Learning AMI (Ubuntu 16.04) Version 26.0 - ami-07728e9e2742b0662

MXNet-1.6.0rc0, Tensorflow-2.0.8.1.5, PyTorch-1.3.1, Keras-2.2, & other frameworks, configured with Neuron, NVIDIA CUDA, cuDNN, NCCL, Intel MKL-DNN, Docker & NVIDIA-Docker. For fully managed experience, check: <https://aws.amazon.com/sagemaker>

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select 64-bit (x86) 64-bit (Arm)

Select 64-bit (x86) 64-bit (Arm)

Select 64-bit (x86)

- In the second step, I select the `t2.micro` instance as again it is the one which is eligible for the free tier. As you can see `t2.micro` is just a single CPU instance with 512 MB RAM. You can opt for a bigger machine if you are dealing with a powerful model or are willing to pay.

Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more about instance types and how they can meet your computing needs.

Filter by: [All instance types](#) [Current generation](#) Show/Hide Columns

Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)

Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
General purpose	t2.micro <small>Free tier eligible</small>	1	1	EBS only	-	Low to Moderate	Yes
General purpose	t2.small	1	2	EBS only	-	Low to Moderate	Yes
General purpose	t2.medium	2	4	EBS only	-	Low to Moderate	Yes

- Keep pressing Next until you reach the “6. Configure Security Group” tab. You will need to add a rule with Type: “Custom TCP Rule”, Port Range:8501, and Source: Anywhere. We use the port 8501 here since it is the custom port used by Streamlit.

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more about Amazon EC2 security groups.](#)

Assign a security group: Create a new security group Select an existing security group

Security group name: launch-wizard-6
Description: launch-wizard-6 created 2019-12-10T00:15:46.817+05:30

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	Custom 0.0.0.0/0	e.g. SSH for Admin Desktop
Custom TCP	TCP	8501	Anywhere 0.0.0.0/0, ::/0	e.g. SSH for Admin Desktop

Add Rule

- You can click on “Review and Launch” and finally on the “Launch” button to launch the instance. Once you click on Launch you might need to create a new key pair. Here I am creating a new key pair named streamlit and downloading that using the “Download Key Pair” button. Keep this key safe as it would be required every time you need to login to this particular machine. Click on “Launch Instance” after downloading the key pair

Select an existing key pair or create a new key pair X

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

1. Create a new key pair

2. Key pair name: streamlit

3. Download Key Pair

4.

 You have to download the **private key file** (*.pem file) before you can continue. **Store it in a secure and accessible location.** You will not be able to download the file again after it's created.

Cancel **Launch Instances**

- You can now go to your instances to see if your instance has started. Hint: See the Instance state, it should be showing “Running”

The screenshot shows the AWS EC2 Instances page. There are two instances listed:

- i-0697ecf1**: t2.micro, stopped, Instance State: stopped, Status Checks: None, Public DNS (IPv4): -
- i-096b8c86de9636e66**: t2.micro, running, Instance State: running, Status Checks: 2/2 checks ..., Alarm Status: None, Public DNS (IPv4): ec2-35-167-158-251.us-west-2.compute.amazonaws.com, IPv4 Public IP: 35.167.158.251, IPv6 IPs: -, Key Name: streamlit

A green arrow points to the Public DNS (IPv4) address of the running instance, which is highlighted in green.

- Select your instance, and copy the **Public DNS(IPv4) Address** from the description. It should be something starting with ec2.
- Once you have that run the following commands in the folder you saved the `streamlit.pem` file. I have masked some of the information here.

```
chmod 400 streamlit.pem
```

```
ssh -i "streamlit.pem" ubuntu@<Your Public DNS(IPv4) Address>
```

```
3. ubuntu@... ~ (ssh)
Last login: Mon Dec  9 19:59:50 on ttys002
~ cd Downloads
~/Downloads chmod 400 streamlit.pem
~/Downloads ssh -i "streamlit.pem" ubuntu@ec2-...
The authenticity of host 'ec2-...' can't be established.
ECDSA key fingerprint is ...
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added ... to the list of known hosts.
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-1051-aws x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

 System information as of Mon Dec  9 19:03:41 UTC 2019

 System load: 0.0          Processes:           85
 Usage of /: 13.6% of 7.69GB   Users logged in:      0
 Memory usage: 15%          IP address for eth0: 172.31.37.50
 Swap usage:  0%
```

Installing Required Libraries

Whoa, that was a handful. After all the above steps you should be able to see the ubuntu prompt for the virtual machine. We will need to set up this machine to run our app. I am going to be using the same `streamlit_football_demo` app that I used in my [previous post](#).

We start by installing miniconda and adding its path to the environment variable.

```
sudo apt-get update  
wget https://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86\_64.sh -O ~/miniconda.sh  
bash ~/miniconda.sh -b -p ~/miniconda  
echo "PATH=$PATH:$HOME/miniconda/bin" >> ~/.bashrc  
source ~/.bashrc
```

We then install additional dependencies for our app to run. That means I install `streamlit` and `plotly_express`.

```
pip install streamlit  
pip install plotly_express
```

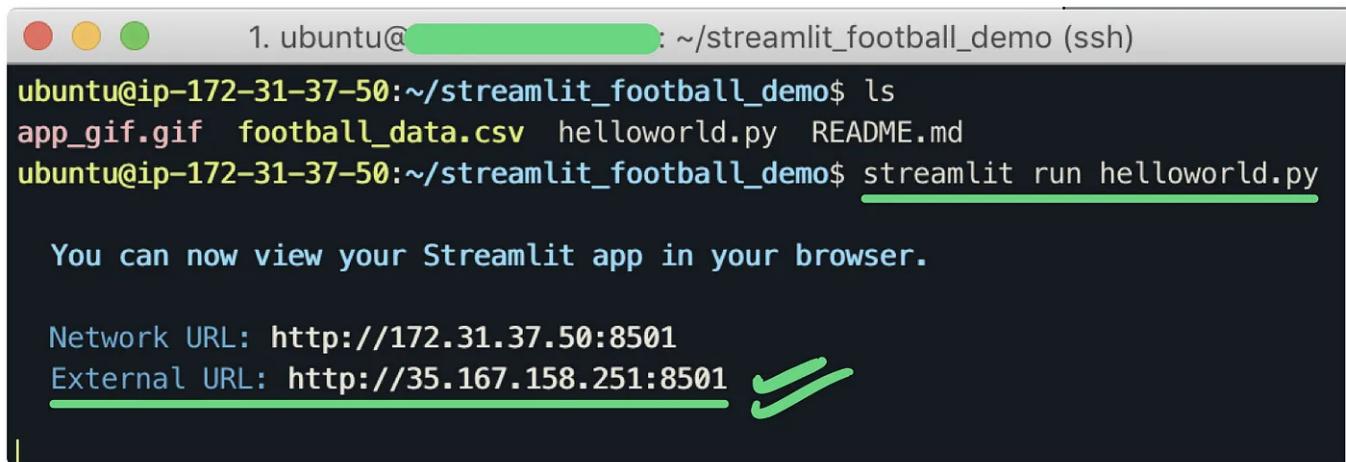
And our machine is now prepped and ready to run.

Running Streamlit on Amazon ec2

As I am set up with the instance, I can get the code for my demo app from [Github](#). Or you can choose to create or copy another app as you wish.

```
git clone https://github.com/MLWhiz/streamlit\_football\_demo.git
```

```
cd streamlit_football_demo
streamlit run helloworld.py
```

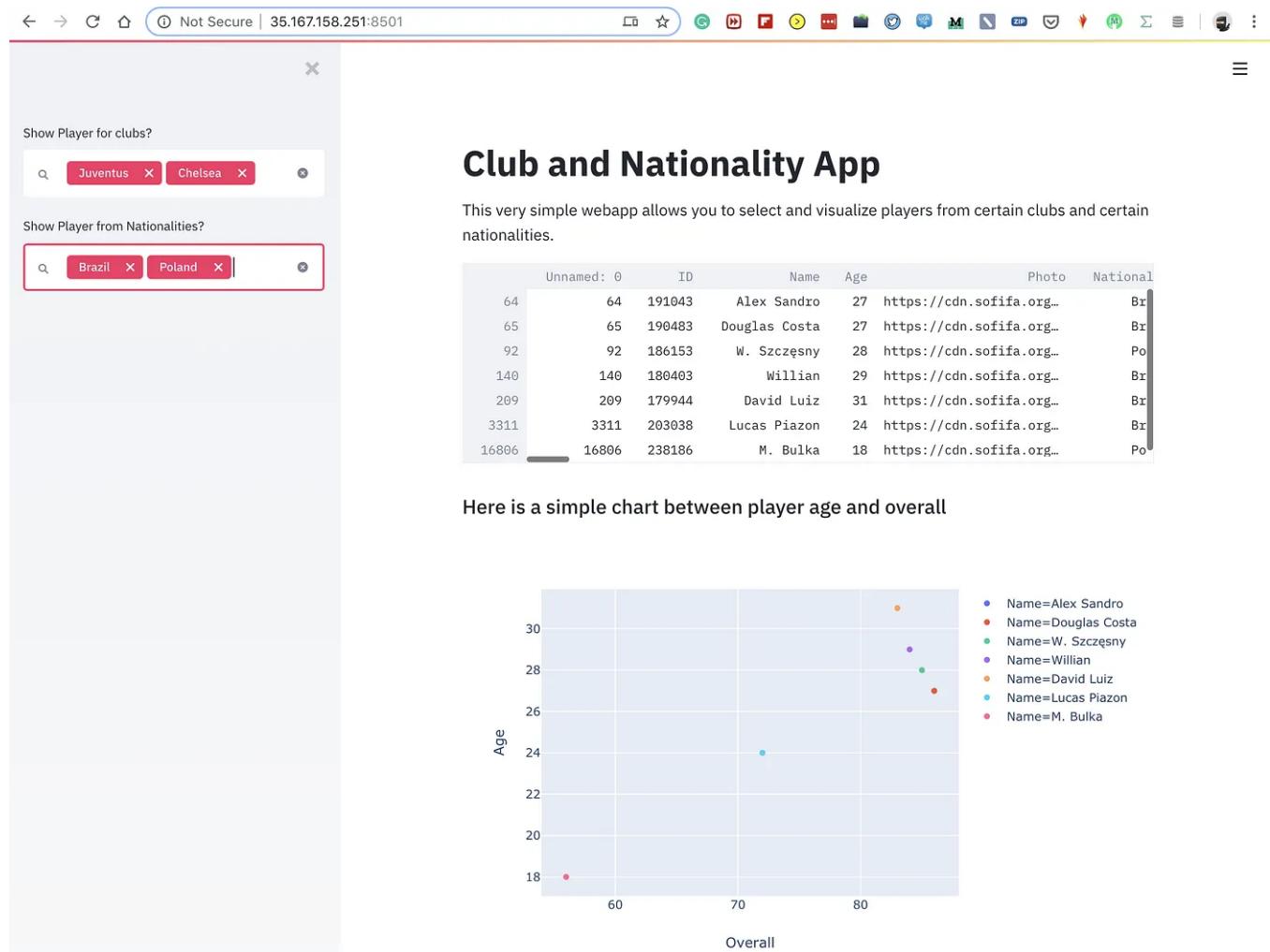


```
1. ubuntu@ip-172-31-37-50:~/streamlit_football_demo (ssh)
ubuntu@ip-172-31-37-50:~/streamlit_football_demo$ ls
app_gif.gif  football_data.csv  helloworld.py  README.md
ubuntu@ip-172-31-37-50:~/streamlit_football_demo$ streamlit run helloworld.py

You can now view your Streamlit app in your browser.

Network URL: http://172.31.37.50:8501
External URL: http://35.167.158.251:8501
```

Now you can go to a browser and type the external URL to access your app. In my case the address is <http://35.167.158.251:8501>. Here is the output. This app will be up right now if you want to play with it.

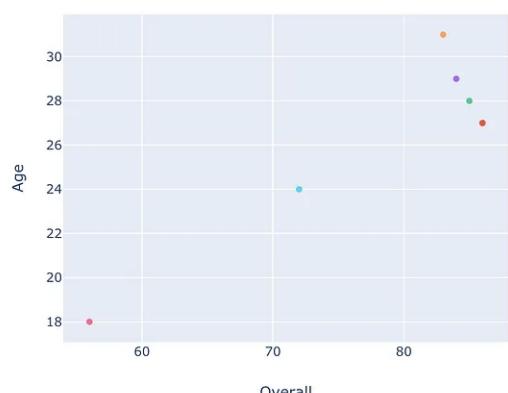


Club and Nationality App

This very simple webapp allows you to select and visualize players from certain clubs and certain nationalities.

	Unnamed:	ID	Name	Age	Photo	National
64	64	191043	Alex Sandro	27	https://cdn.sofifa.org...	Br
65	65	190483	Douglas Costa	27	https://cdn.sofifa.org...	Br
92	92	186153	W. Szczęsny	28	https://cdn.sofifa.org...	Po
140	140	180403	Willian	29	https://cdn.sofifa.org...	Br
209	209	179944	David Luiz	31	https://cdn.sofifa.org...	Br
3311	3311	203038	Lucas Piazon	24	https://cdn.sofifa.org...	Br
16806	16806	238186	M. Bulká	18	https://cdn.sofifa.org...	Po

Here is a simple chart between player age and overall



The scatter plot shows the relationship between Age (Y-axis, ranging from 18 to 30) and Overall (X-axis, ranging from 60 to 80). The data points are color-coded according to the player's name:

- Name=Alex Sandro (Blue)
- Name=Douglas Costa (Red)
- Name=W. Szczęsny (Green)
- Name=Willian (Purple)
- Name=David Luiz (Orange)
- Name=Lucas Piazon (Teal)
- Name=M. Bulká (Pink)

A Very Small Problem Though

We are up and running with our app for the world to see. *But whenever you are going to close the SSH terminal window the process will stop and so will your app.*

So what do we do?

TMUX to the rescue. TMUX allows us to keep running our sessions even after we leave the terminal window. It also helps with a lot of other things but I will just go through the steps we need.

First, we stop our app using `Ctrl+C` and install `tmux`

```
sudo apt-get install tmux
```

We start a new `tmux` session using the below command. We keep the name of our session as `StreamSession`. You could use any name here.

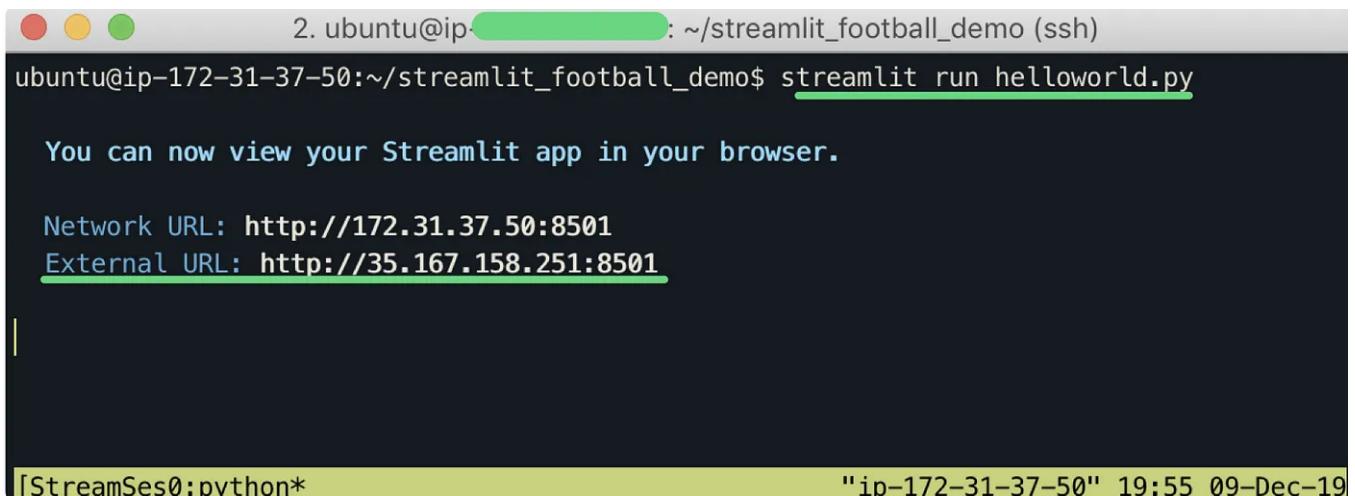
```
tmux new -s StreamSession
```

The screenshot shows a terminal window with the following details:

- Title bar: "2. ubuntu@ip-172-31-37-50: ~/streamlit_football_demo (ssh)"
- Prompt: "ubuntu@ip-172-31-37-50:~/streamlit_football_demo\$"
- Bottom status bar: "[StreamSes0: bash*]" and "ip-172-31-37-50" 19:52 09-Dec-19
- An annotation with a green arrow points from the left towards the session name "StreamSession" in the title bar, which is underlined and written in green.

You can see that the session name is “`StreamSession`” at the bottom of the screen. You can now start running `streamlit` in the `tmux` session.

```
streamlit run helloworld.py
```



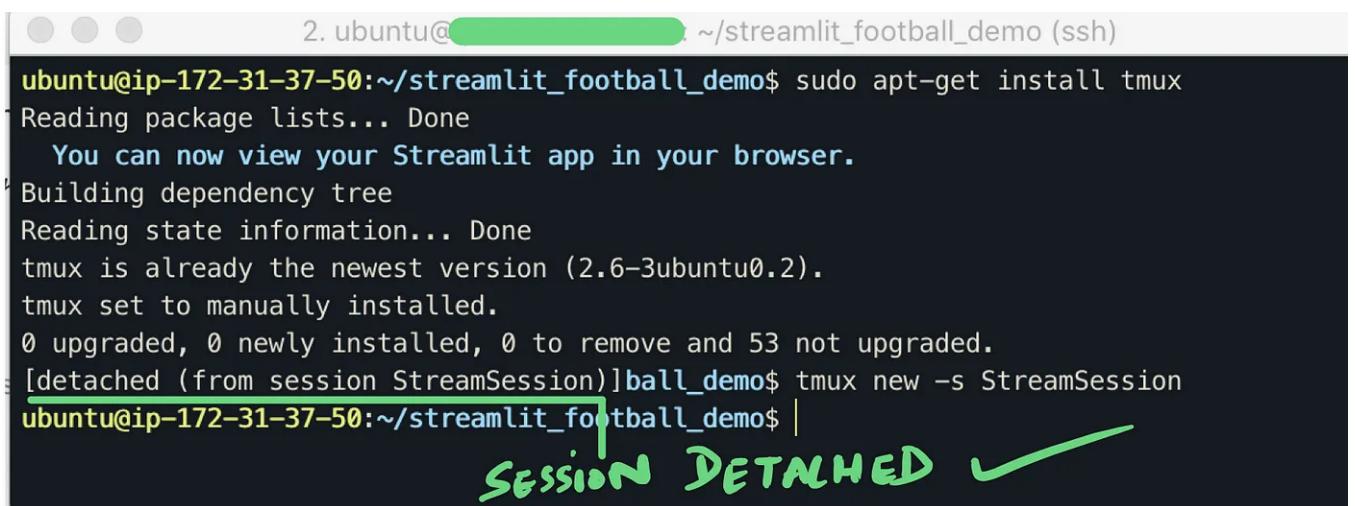
```
2. ubuntu@ip-172-31-37-50:~/streamlit_football_demo (ssh)
ubuntu@ip-172-31-37-50:~/streamlit_football_demo$ streamlit run helloworld.py

You can now view your Streamlit app in your browser.

Network URL: http://172.31.37.50:8501
External URL: http://35.167.158.251:8501

[StreamSes0:python*] "ip-172-31-37-50" 19:55 09-Dec-19
```

You will be able to see your app at the External URL. The *next step is to detach our TMUX session* so that it continues running in the background when you leave the SSH shell. To do this just press `Ctrl+B` and then `D` (Don't press Ctrl when pressing D)



```
2. ubuntu@ip-172-31-37-50:~/streamlit_football_demo (ssh)
ubuntu@ip-172-31-37-50:~/streamlit_football_demo$ sudo apt-get install tmux
Reading package lists... Done
You can now view your Streamlit app in your browser.
Building dependency tree
Reading state information... Done
tmux is already the newest version (2.6-3ubuntu0.2).
tmux set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 53 not upgraded.
[detached (from session StreamSession)]ball_demo$ tmux new -s StreamSession
ubuntu@ip-172-31-37-50:~/streamlit_football_demo$ |
```

SESSION DETACHED ✓

You can now close your SSH session and the app will continue running at the External URL.

And Voila! We are up and running.

Pro TMUX Tip: You can reattach to the same session by using the `attach` command below. The best part is that you can close your SSH shell and then maybe come back after some hours and reattach to a session and keep working from wherever you were when you closed the SSH shell.

```
tmux attach -t StreamSession
```

Simple Troubleshooting:

If your app is not hosting at 8501, it means that an instance of streamlit app is already running on your system and you will need to stop that. You can do so by first finding the process ID

```
ps aux | grep streamlit
```

You will see something like:

```
ubuntu  20927  2.4 18.8 713780 189580 pts/3    Sl+  19:55   0:26
/home/ubuntu/miniconda/bin/python
/home/ubuntu/miniconda/bin/streamlit run helloworld.py
```

You will need to *kill this process*. You can do this simply by

```
kill -9 20947
```

Conclusion



Show Player for clubs?

Choose an option

Show Player from Nationalities?

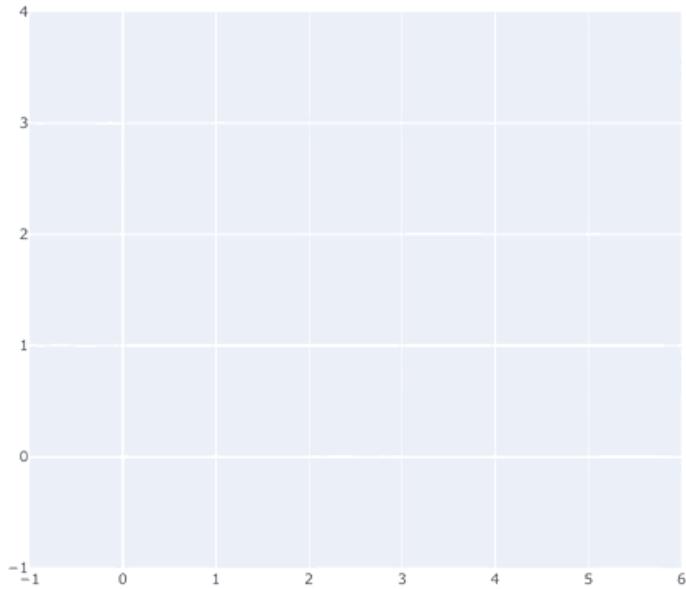
Choose an option

Club and Nationality App

This very simple webapp allows you to select and visualize players from certain clubs and certain nationalities.

empty

Here is a simple chart between player age and overall



Our Final App

Streamlit has democratized the whole process to create apps, and I couldn't recommend it more. If you want to learn more about how to create awesome web apps with Streamlit then read up my [last](#) post.

In this post, we deployed a simple web app on AWS using amazon ec2.

In the process of doing this, we created our own Amazon ec2 instance, logged into the SSH shell, installed miniconda and dependencies, ran our Streamlit application and learned about TMUX. Enough learning for a day?

So go and show on these Mad skills. To end on a lighter note, as Sten Sootla says in his satire piece which I thoroughly enjoyed:

The secret: it's not what you know, it's what you show.

If you want to learn more about how to structure a Machine Learning project and the best practices, I would like to call out his excellent third course named Structuring Machine learning projects in the Coursera Deep Learning Specialization. Do check it out.

Thanks for the read. I am going to be writing more beginner-friendly posts in the future too. Follow me up at Medium or Subscribe to my blog to be informed about them. As always, I welcome feedback and constructive criticism and can be reached on Twitter @mlwhiz.

Also, a small disclaimer — There might be some affiliate links in this post to relevant resources, as sharing knowledge is never a bad idea.

Data Science

Programming

Artificial Intelligence

Machine Learning

Web Development

Enjoy the read? Reward the writer. Beta

Your tip will go to Rahul Agarwal through a third-party platform of their choice, letting them know you appreciate their story.

[Give a tip](#)

Sign up for The Variable

By Towards Data Science

Every Thursday, the Variable delivers the very best of Towards Data Science: from hands-on tutorials and cutting-edge research to original features you don't want to miss. [Take a look.](#)

Emails will be sent to shivakmuddam25@gmail.com. [Not you?](#)

[Get this newsletter](#)