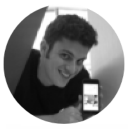
PubNub / blog

BUILD - 5 MIN READ
Jan 11, 2019

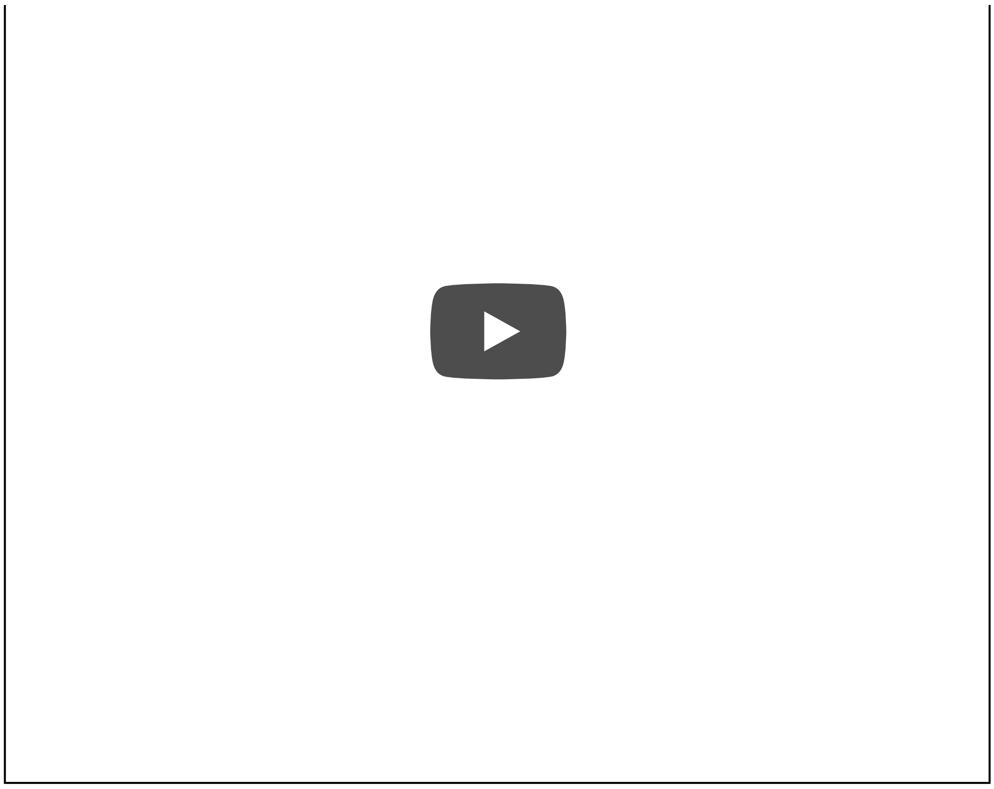
# Socket Programming in Python: Client, Server, and Peer Examples

Sockets (aka socket programming) enable programs to send and receive data, bi-directionally, at any given moment. This tutorial walks through how you can send data from device-to-device, client-to-ser...



ADAM BAVOSA
@realadambavosa
Developer Relations Manager, PubNub

Sockets (aka socket programming) enable programs to send and receive data, **bi-directionally**, at any given moment. This tutorial walks through how you can **send data** from **device-to-device, client-to-server**, and **vice versa** using socket programming in Python



## Why use sockets to send data?

Internet-connected applications that need to operate in realtime greatly benefit from the implementation of **sockets** in their **networking code**. Some examples of apps that use socket programming are:

- Web pages that show live notifications (Facebook, Twitch, eBay)
- Multiplayer online games (League of Legends, WoW, Counter Strike)
- Chat apps (WhatsApp, WeChat, Slack)
- Realtime data dashboards (Robinhood, Coinbase)
- IoT devices (Nest, August Locks)

Python, unlike JavaScript, is a language that executes synchronously. This is why **asyncio** was developed – to make Python more robust, particularly for the nature of socket programming.

GitHub Repo

GITHUB REPO

## Python Socket Programming Tutorial

Natively, Python provides a socket class so developers can easily implement **socket objects** in their source code. To use a socket object in your program, start off by importing the socket library. No need to install it with a package manager, it comes out of the box with Python.

```
import socket
```

Now we can create socket objects in our code.

```
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

This code creates a socket object that we are storing in the “sock” variable. The constructor is provided a family and type parameter respectively.  The family parameter is set to the default value, which is the **Address Format Internet**.

The type parameter is set to **Socket Stream**, also the default which enables “sequenced, reliable, two-way, connection-based byte streams” over **TCP**[1].

Once we have an initialized socket object, we can use some methods to open a **connection**, **send** data, **receive** data, and finally **close** the connection.

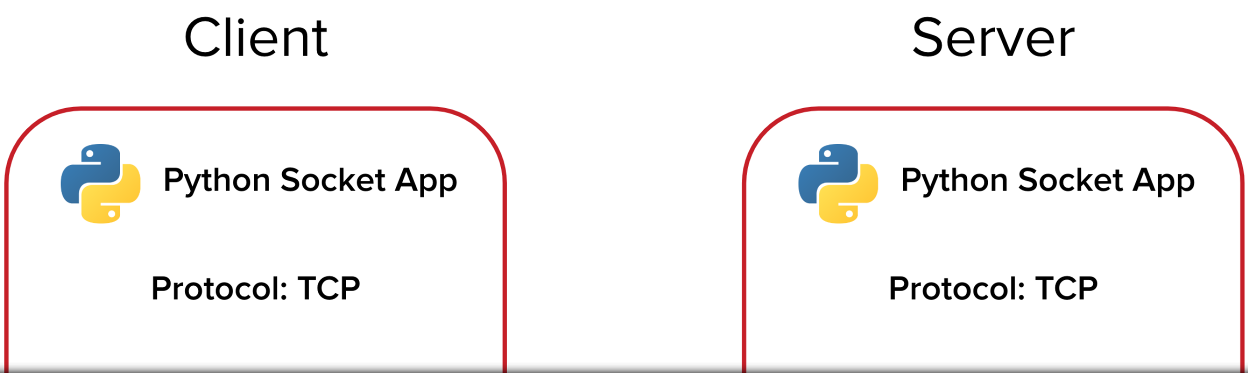
```
## Connect to an IP with Port, could be a URL
sock.connect(('0.0.0.0', 8080))

## Send some data, this method can be called multiple times
sock.send("Twenty-five bytes to send")

## Receive up to 4096 bytes from a peer
sock.recv(4096)
```

PubNub / blog

```python
import socket

serv = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

serv.bind(('0.0.0.0', 8080))
serv.listen(5)

while True:
    conn, addr = serv.accept()
    from_client = ''

    while True:
        data = conn.recv(4096)
        if not data: break
        from_client += data
        print from_client

        conn.send("I am SERVER\n")

    conn.close()
    print 'client disconnected'
```

This code makes a socket object, and binds it to **localhost's port 8080** as a **socket server**. When clients connect to this address with a socket connection, the server listens for data, and stores it in the "data" variable.

Next, the program logs the client data using "print," and then sends a string to the client: **I am SERVER**.



Here is the **client** socket demo code.

```
import socket

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(('0.0.0.0', 8080))

client.send("I am CLIENT\n")

from_server = client.recv(4096)

client.close()

print from_server
```

This client opens up a socket connection with the server, but **only if the server program is currently running**. To test this out yourself, you will need to use **2 terminal windows** at the same time.

Next, the client sends some data to the server: **I am CLIENT**

Then the client receives some data it anticipates from the server.

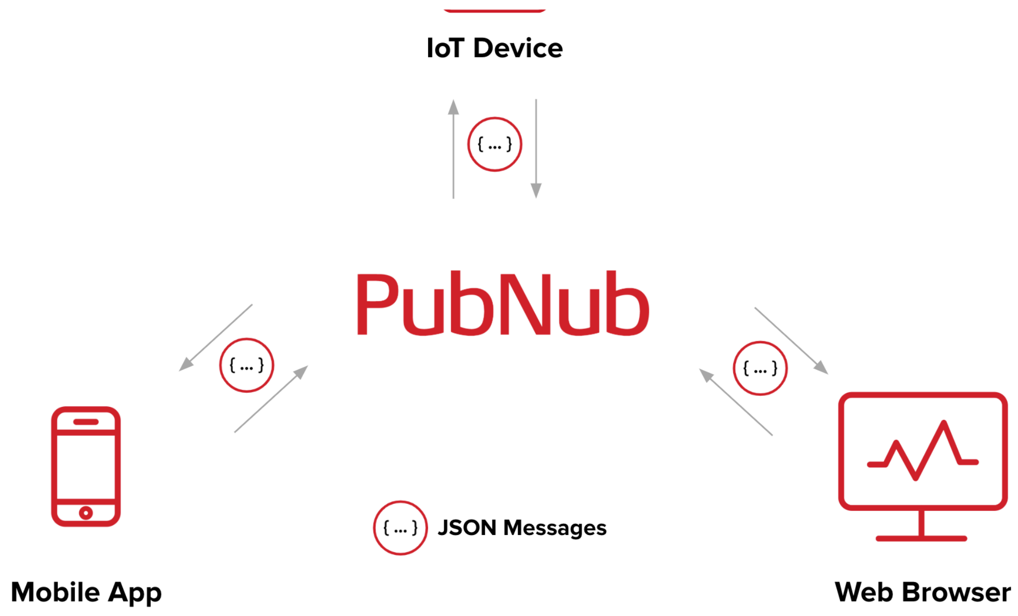
Done! You can now get started **streaming data between clients and servers** using some basic Python network programming.

## How Do You Send Data Between Clients?

Sending data **between 2 or more client devices** over the internet is tricky. Due to protections implemented by network security, not all devices connected to the world wide web have a publicly accessible internet protocol (IP) address.

This means that the Python code that we implemented will not be 100% reliable for sending peer-to-peer data in our realtime app. How do we achieve **reliability** and **speed** when transmitting **peer-to-peer data**?

PubNub / blog



PubNub does this best with the Pub/Sub API. It is fast, reliable, secure, and easy to implement on **any client device**. Whether you have a Python server, a JavaScript website, or anything in between, you can use PubNub to send data to anyone in **under 250ms**.

With **One-to-Many**, **One-to-One**, or **Many-to-Many**, PubNub **scales automatically** to support any application load. Using the API opens up an instant, always-on connection between all clients that have the Pub/Sub API keys. This accomplishes the same objectives as a socket connection.

## PubNub and Python with an SSL Connection

Here is an example of **peer-to-peer data** that is sent with PubNub, on a single channel, with **SSL**. You can think of this like sending data over a TCP socket. When you sign up for a free PubNub account, you can use a practically **infinite number of channels** to send messages in realtime. Before you try the code, be sure to **make a free PubNub account**.

PubNub / blog

account? Login

Register with Gmail

Or Sign Up with Email

Email Address

Password

At least 8 characters, contain one uppercase, one lowercase, one number and one special character.

Sign Up For Free

By clicking on Sign Up, or Register with Gmail, you agree to our Terms & Conditions and Privacy Policy.

Privacy - Terms

## Client 1

```python
from pubnub.callbacks import SubscribeCallback
from pubnub.enums import PNStatusCategory
from pubnub.pnconfiguration import PNConfiguration
from pubnub.pubnub import PubNub
import time
import os

pnconfig = PNConfiguration()

pnconfig.publish_key = 'your pubnub publish key here'
pnconfig.subscribe_key = 'your pubnub subscribe key here'
pnconfig.ssl = True

pubnub = PubNub(pnconfig)

def my_publish_callback(envelope, status):
```

```python

pubnub.add_listener(MySubscribeCallback())
pubnub.subscribe().channels("chan-1").execute()

## publish a message
while True:
    msg = raw_input("Input a message to publish: ")
    if msg == 'exit': os._exit(1)
    pubnub.publish().channel("chan-1").message(str(msg)).pn_async(my_publish_callback)
```

## Client 2

Strings can be entered on the command line for these 2 client programs. Maximum message size for PubNub publishing is 32kb. Use 2 terminal windows to try out the code!

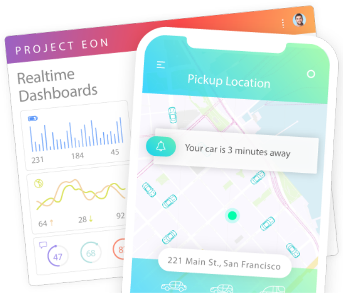
```python
from pubnub.callbacks import SubscribeCallback
from pubnub.enums import PNStatusCategory
from pubnub.pnconfiguration import PNConfiguration
from pubnub.pubnub import PubNub
import time
import os

pnconfig = PNConfiguration()

pnconfig.publish_key = 'your pubnub publish key here'
pnconfig.subscribe_key = 'your pubnub subscribe key here'
pnconfig.ssl = True

pubnub = PubNub(pnconfig)

def my_publish_callback(envelope, status):
    # Check whether request successfully completed or not
    if not status.is_error():
        pass

class MySubscribeCallback(SubscribeCallback):
    def presence(self, pubnub, presence):
        pass
```

PubNub / blog

```
        if msg == 'exit': os._exit(1)
        pubnub.publish().channel("chan-
1").message(str(msg)).pn_async(my_publish_callback)
```

All of the code in this post is hosted on GitHub in the Python Socket Demo repository.

PubNub is entirely free up to **1 million messages per month**. For more capability of the API, check out the PubNub Python v4 SDK documentation, or any of the other 75+ PubNub client SDKs.



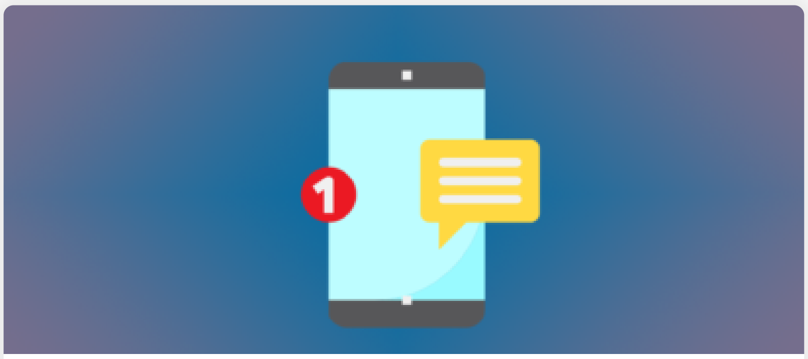
TAGS: PUBNUB DATA STREAMS PYTHON WEBSOCKETS

MORE FROM PUBNUB

PubNub / blog

in Only 10 Line...

CHANDLER MAYO
Developer Relations Engineer, PubNub



BUILD - 9 MIN READ Sep 6, 2018

How to Setup Push Notifications in React Native (iOS & Andro...

CHANDLER MAYO
Developer Relations Engineer, PubNub

BUILD - 8 MIN READ Jun 1, 2018

PubNub / blog

PRICING

USE CASES

Chat

Live Notifications

IoT Device Control

FEATURES

Pub/Sub Messaging

Presence

Functions

Gateways

DEVELOPERS

70+ SDKs and Full Documentation

Quickstart Guide

SDK Download

General Concepts

Tutorials

Project EON

PubNub Ambassador

PubNub Partners

SUPPORT

Help & Support

Resource Center

PubNub / blog