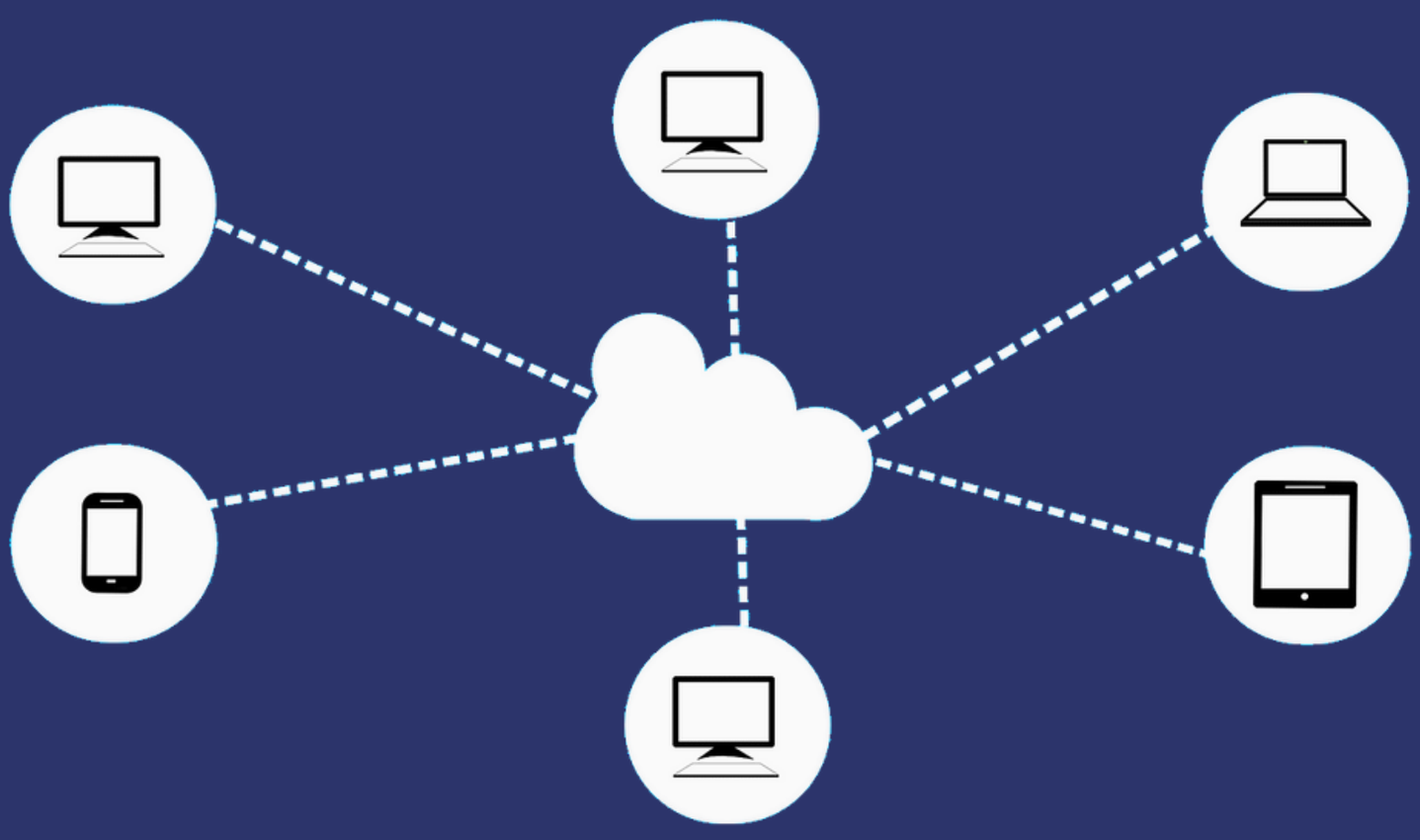
DOCUMENTATION

UDP KING

RIO 3D STUDIOS



NETWORKING

a guide to create your networking aplication

learn step by step to develop your mmo!

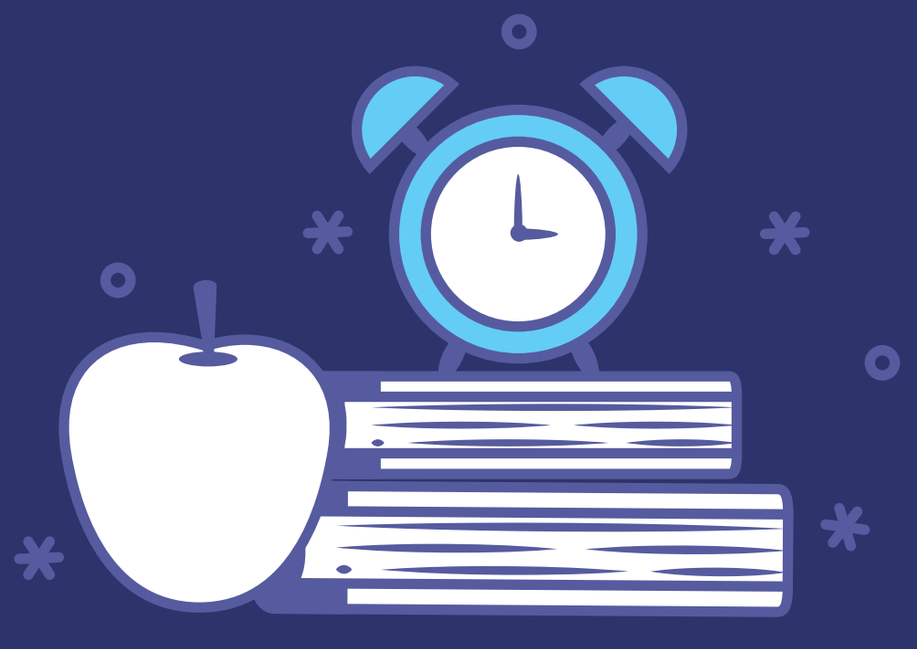
SUMMARY

QUICK START

HOW TO DEVELOP

HOSTING SERVER

CONTACT & SUPPORT

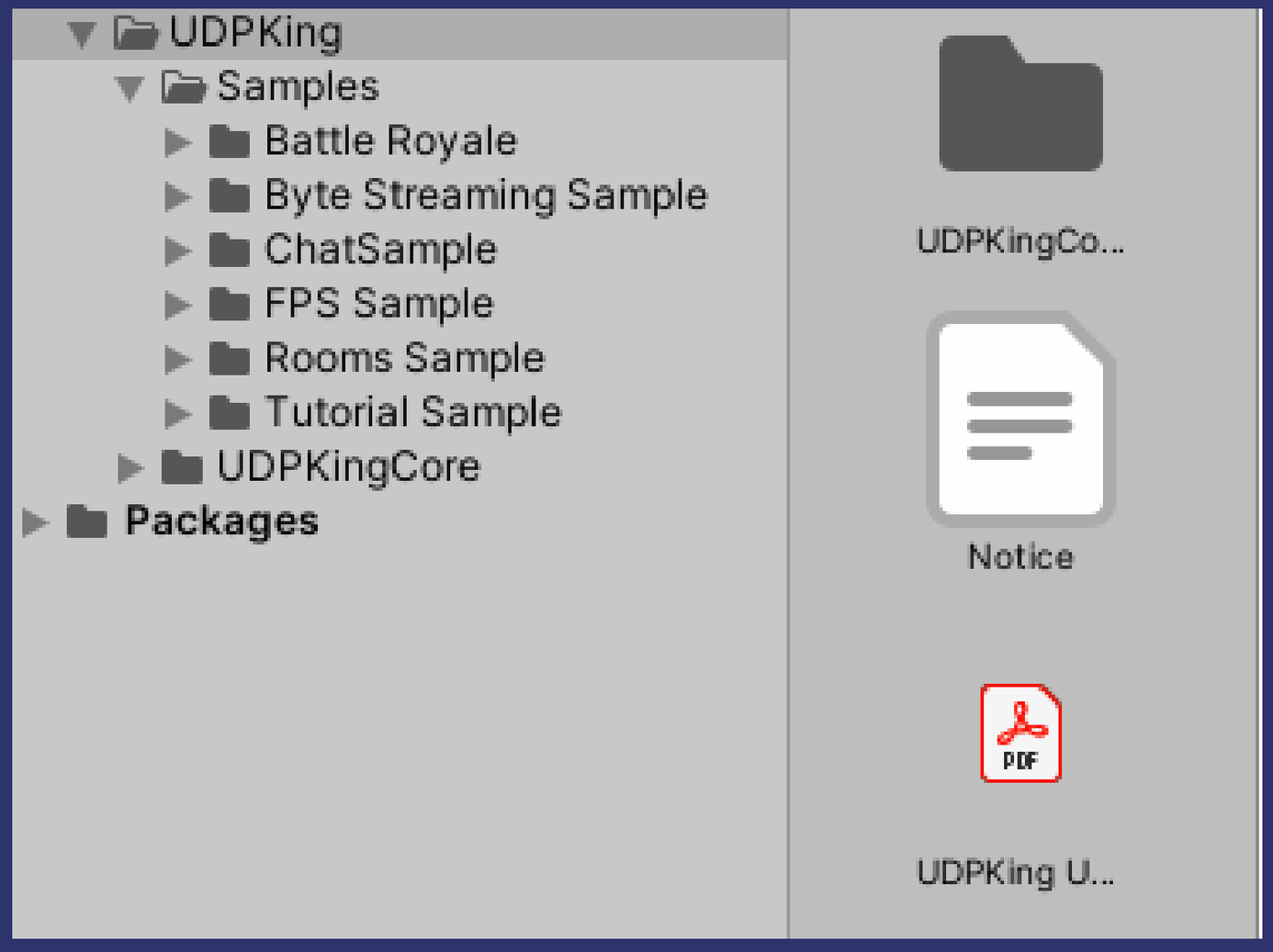


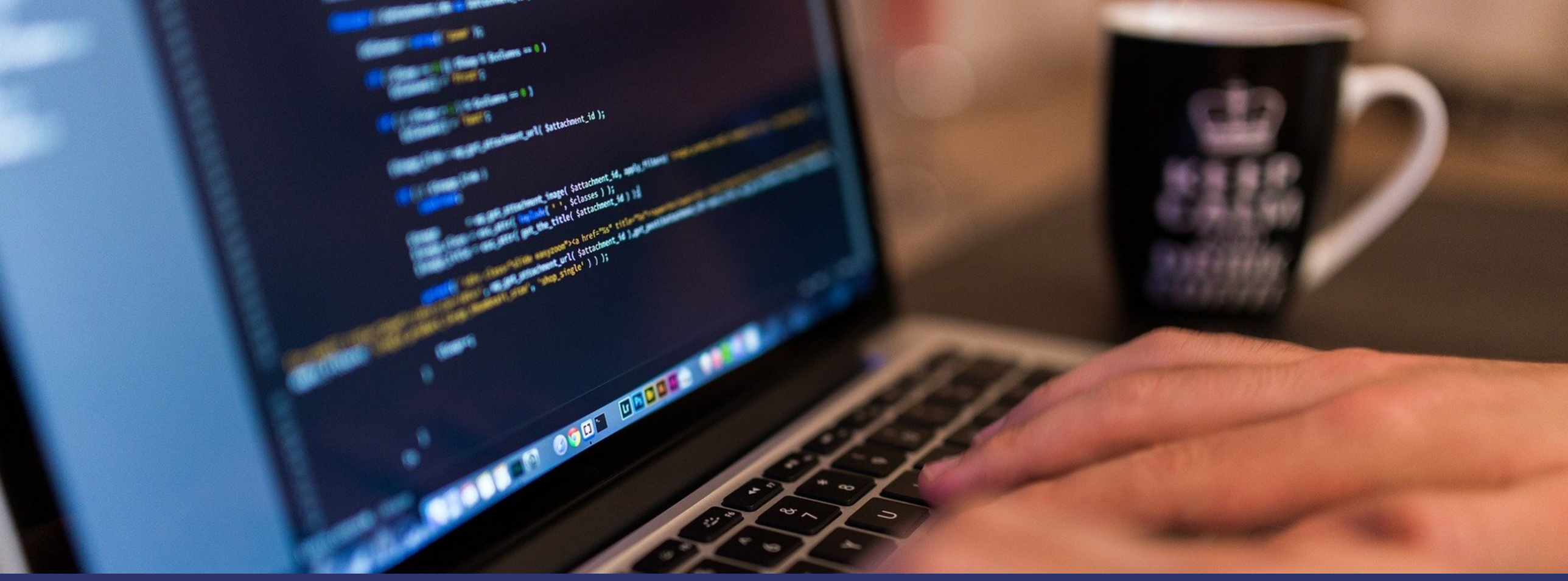
QUICK START

SETUP MMO APLICATION

This guide will show you how to create lots of amazing applications with few code lines!

In the project, choose an sample of your preference and select the main scene of the sample:





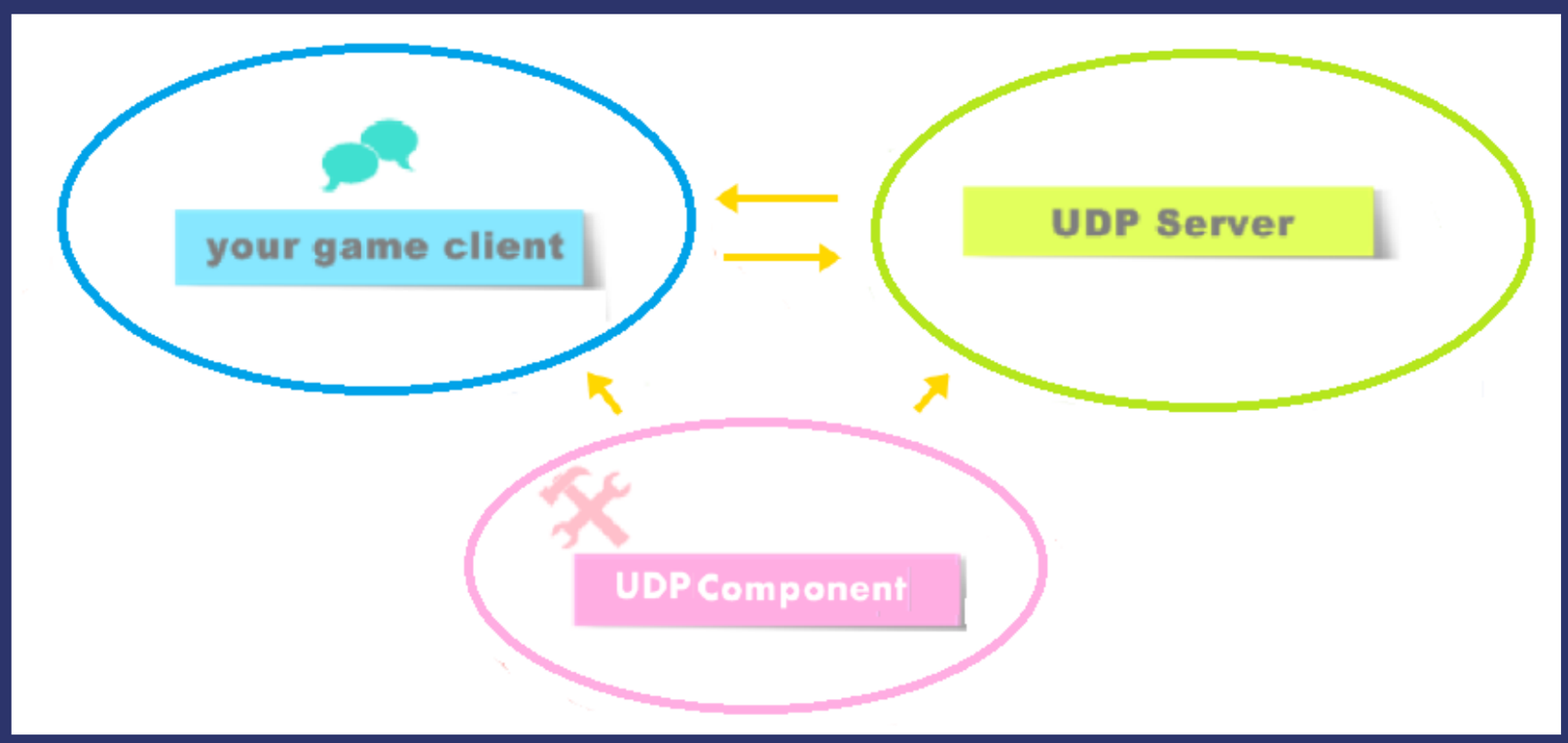
How to develop

Before developing your network aplication, keep in mind that the application will be divided into two parts:

- The Front-end: all visual, game mechanics and physics will be done in unity. Whatever game you have in mind, we will just use a main class called Network Manager to do all the interaction with the server.

- The Back-end: here comes the server, which will make the connection between all game clients.

The UDPComponent



The UDP King library is responsible for encapsulating all low-level operations required to open a UDP Socket and send messages to a UDP server.

There is no need to make any changes or modifications to this module.

1. The main class of this module is **UDPComponent.cs**, this class has the ability to act as a client or server, depending on the your choice.

2. **UDPComponent** as a client, provides three methods responsible for:
   - UDPComponent .**connect()**: Connect to the server
   - UDPComponent .**EmitToServer()**: Send packets to the server
   - UDPComponent .**On()** : Receive packets from the server

3. Acting as a server also offers three methods:
   - UDPComponent.**CreateServer()** : start the server
   - UDPComponent.**EmitToClient()** : send packages to the clients
   - UDPComponent.**On()** : receive packages from the server

Note: the operations of this library
is very similar to the popular
socket.io library (for nodeJS).

CONNECT GAME CLIENT TO THE SERVER

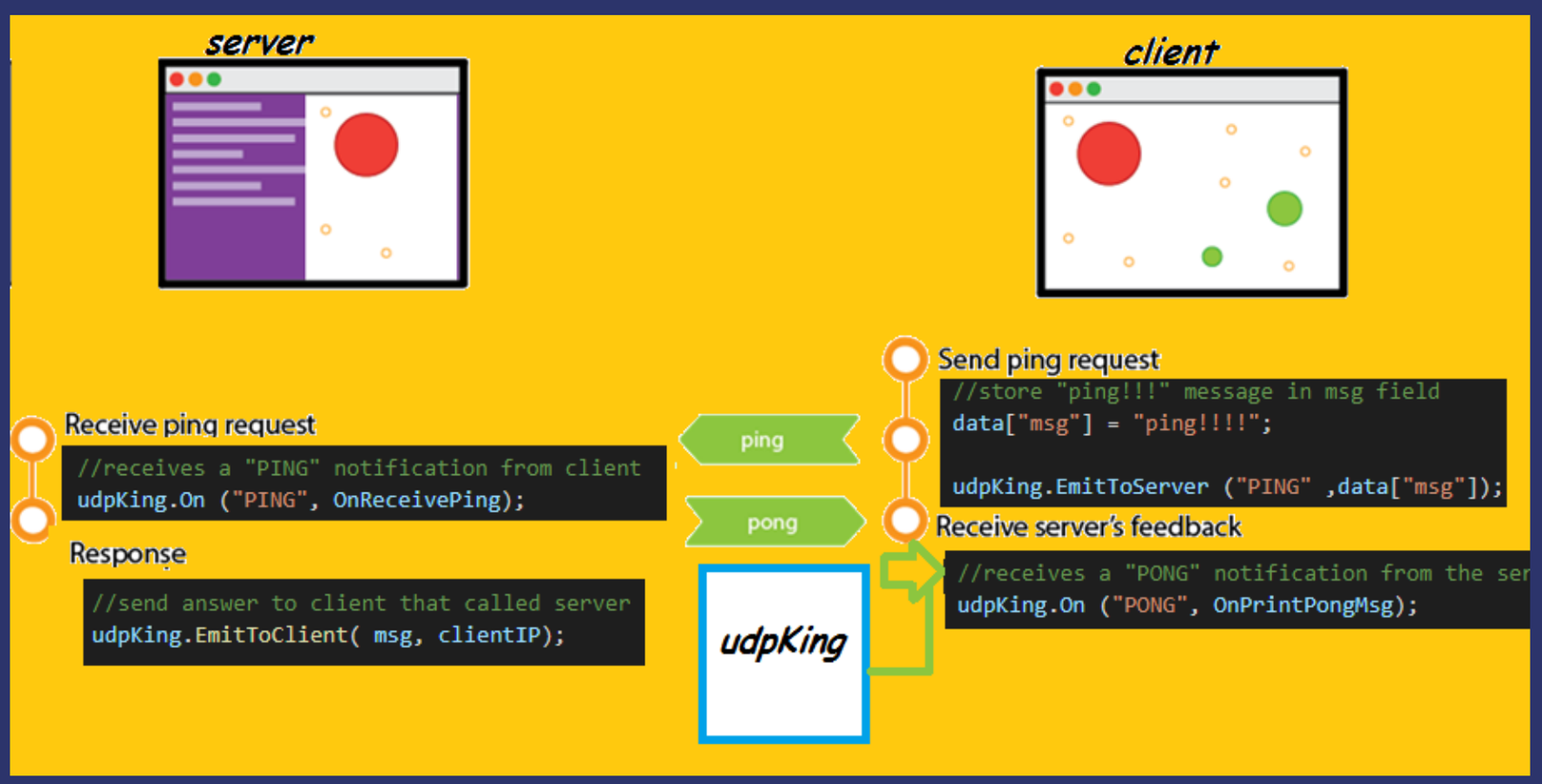
Connecting to a server with just one single line:

NetworkManager.cs

```
udpKing = gameObject.GetComponent<UDPComponent>();

//establish a connection to the server
udpKing.connect ();
```

SEND AND RECEIVE PACKAGES FROM THE SERVER



You can use send information to server using the **UDPComponent.EmitToServer** method:

NetworkManager.cs

```
// <summary>
/// sends ping message to server.
/// </summary>
public void EmitPing() {

    //hash table <key, value>
    Dictionary<string, string> data = new Dictionary<string, string>();

    //Identifies with the name "PING", the notification to be transmitted to the server
    data["callback_name"] = "PING";

    //store "ping!!!" message in msg field
    data["msg"] = "ping!!!!";

    udpKing.EmitToServer ("PING" ,data["msg"]);
}
```

The server get the client packages using:

server.js

```
//receives a "PING" notification from client
udpKing.On ("PING", OnReceivePing);
```

Receiving packages from server.

In server you can use send information to client using the **EmitToClient** method:

```
//send answer to client that called server
udpKing.EmitToClient( msg, data.anyIP);
```

To receive messages from the server you can create a listener with the following command:

```
//receives a "PONG" notification from the server
udpKing.On ("PONG", OnPrintPongMsg);
```

with the server information in hand, we can update the game status, create a function to treat the listener event:

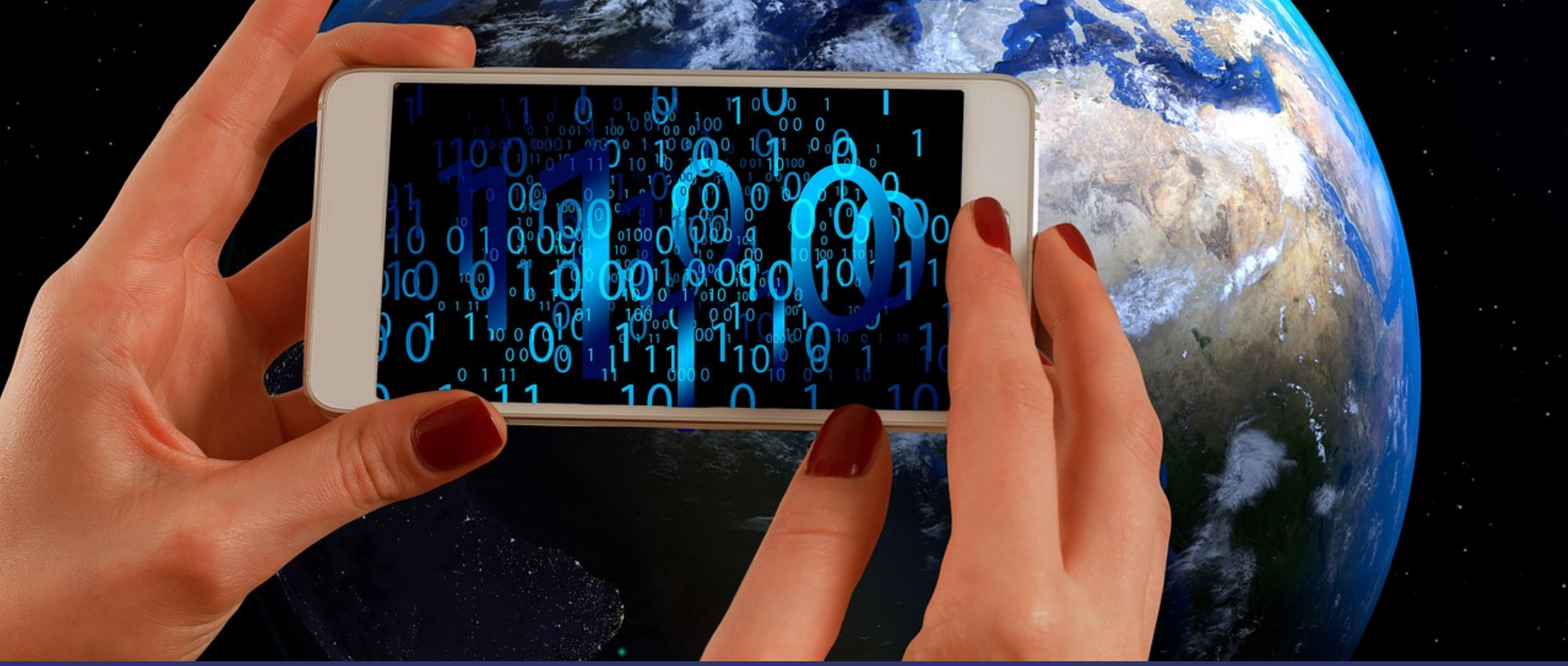
```
/// <summary>
/// Prints the pong message which arrived from server.
/// </summary>
/// <param name="data">received package from server.</param>
public void OnPrintPongMsg(UDPKingEvent data)
{

    /*
     * data.pack[0]= CALLBACK_NAME: "PONG"
     * data.pack[1]= "pong message!!!"
    */

    Debug.Log ("received message: "+data.pack[1] +" from server by callbackID: "+data.pack[0]);

}
```

- The same steps can be applied for the other network functions in your game!



Streaming byte array

In some applications we may be interested in sending and receiving byte arrays between client and server. For this, the UDPComponent class offers the **EmitBytesToServer** and **SendBytesToClient** methods.

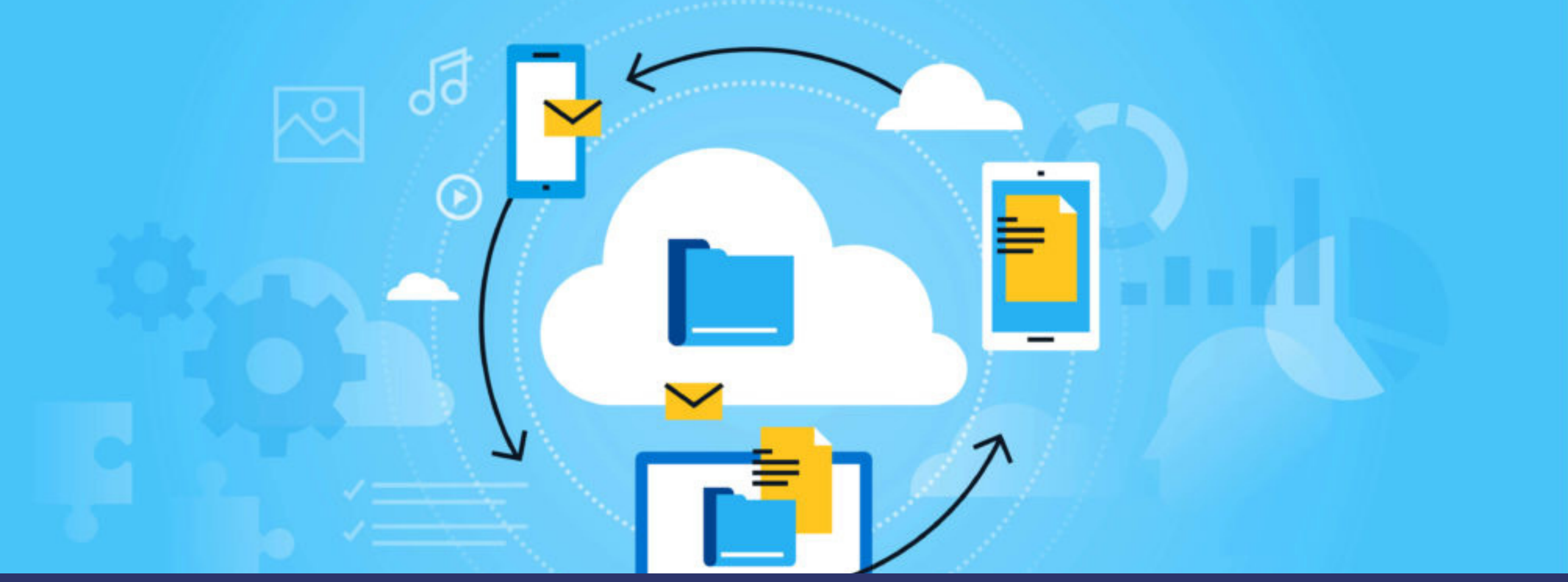
```
byte[] imageBytes = cameraTexture.EncodeToJPG();

//send camera texture to server
udpKing.EmitBytesToServer("SEND_TEXTURE_STREAM", CLZF2.Compress(imageBytes));
```

```
foreach (KeyValuePair<string, Client> entry in connectedClients) {

    //send streaming to all clients in connectClients list
    udpKing.SendBytesToClient("RECEIVE_TEXTURE_STREAM", data.byteArray, entry.Value.remoteEP);

}//END_FOREACH
```



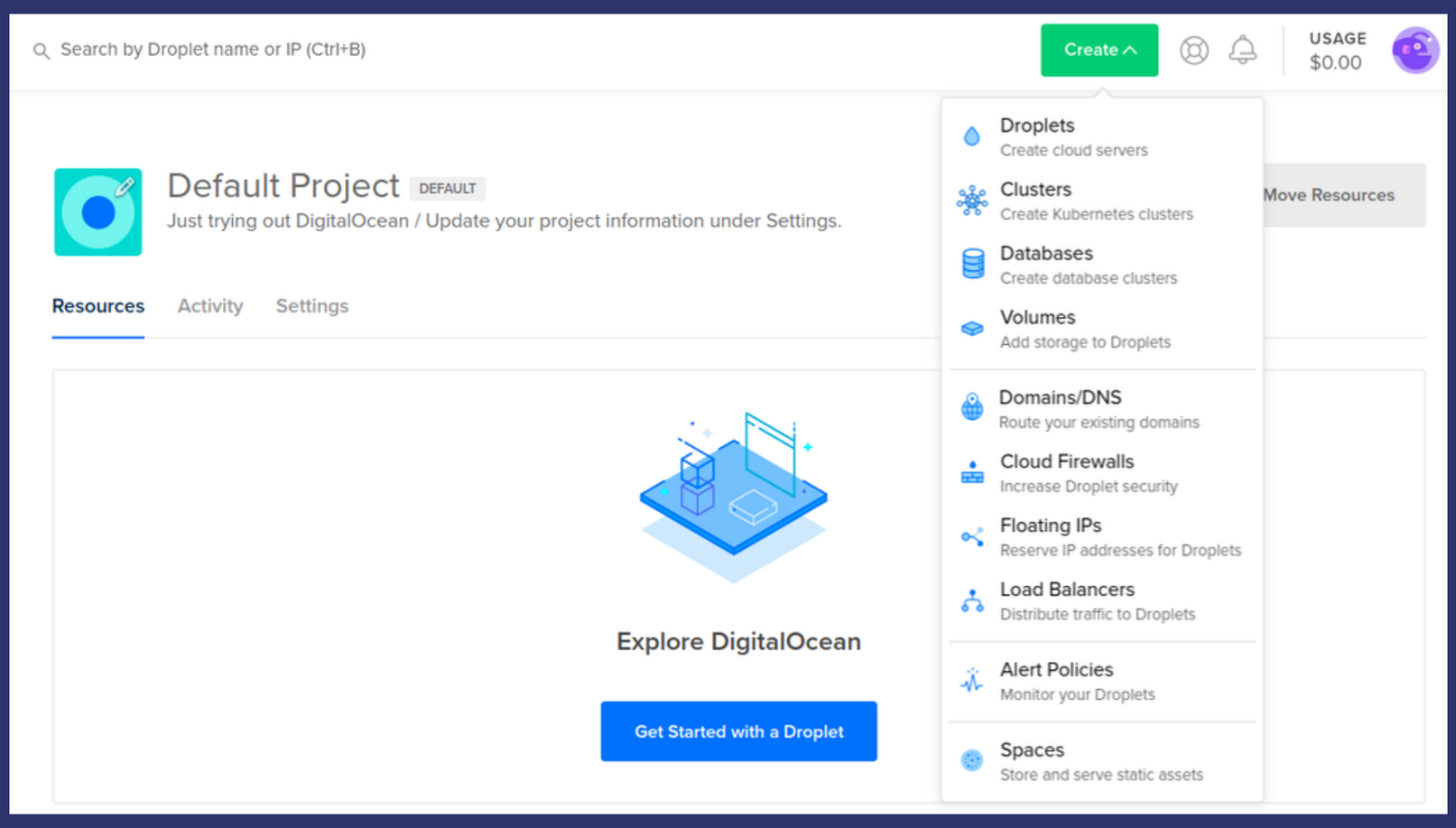
Hosting server

Server hosting, it's pretty simple on Digital Ocean, amazon aws, google cloud and azure, they all work very similary!

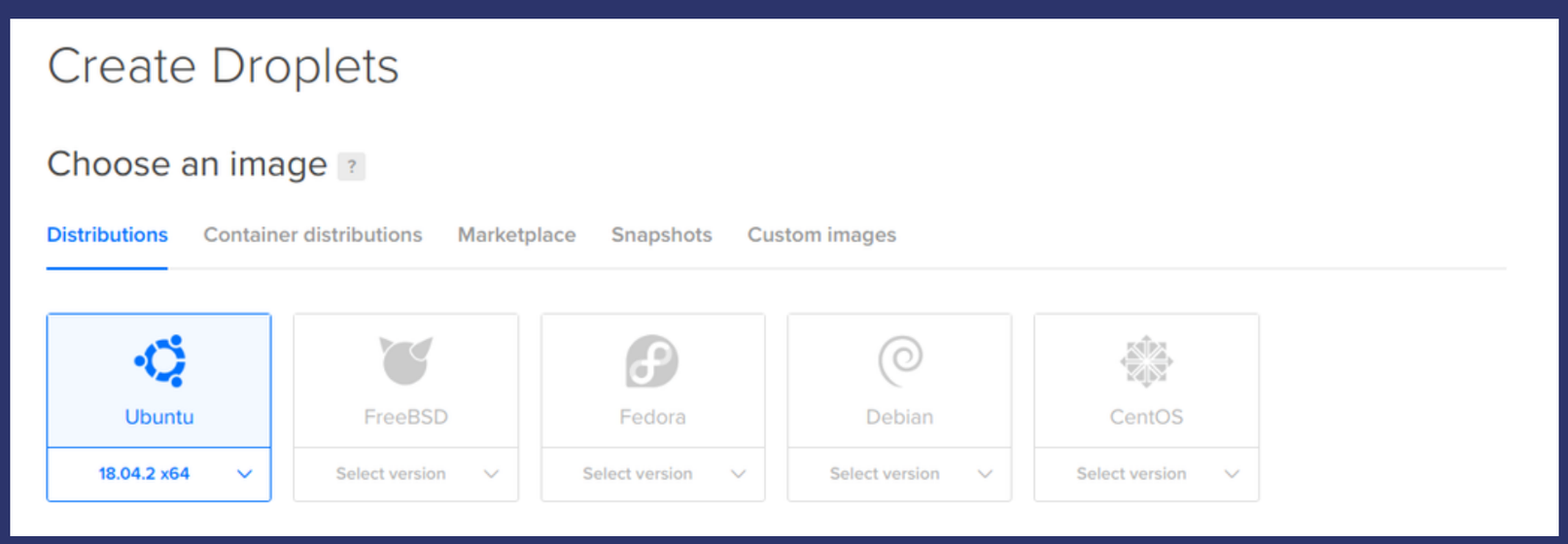
in this tutorial we will teach you how to create a dedicated server in Digital Ocean.

DigitalOcean Droplets are Linux-based virtual machines (VMs) that run on top of virtualized hardware. Each Droplet you create is a new server you can use, either standalone or as part of a larger, cloud-based infrastructure.

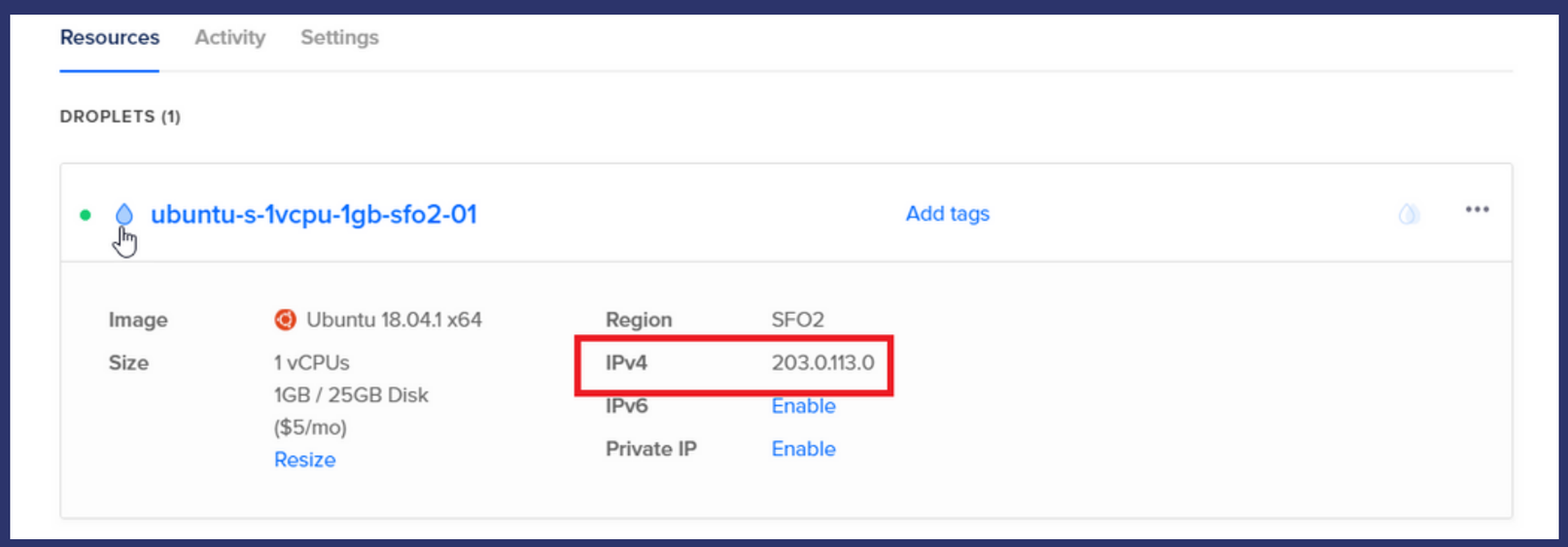
Create a Droplet



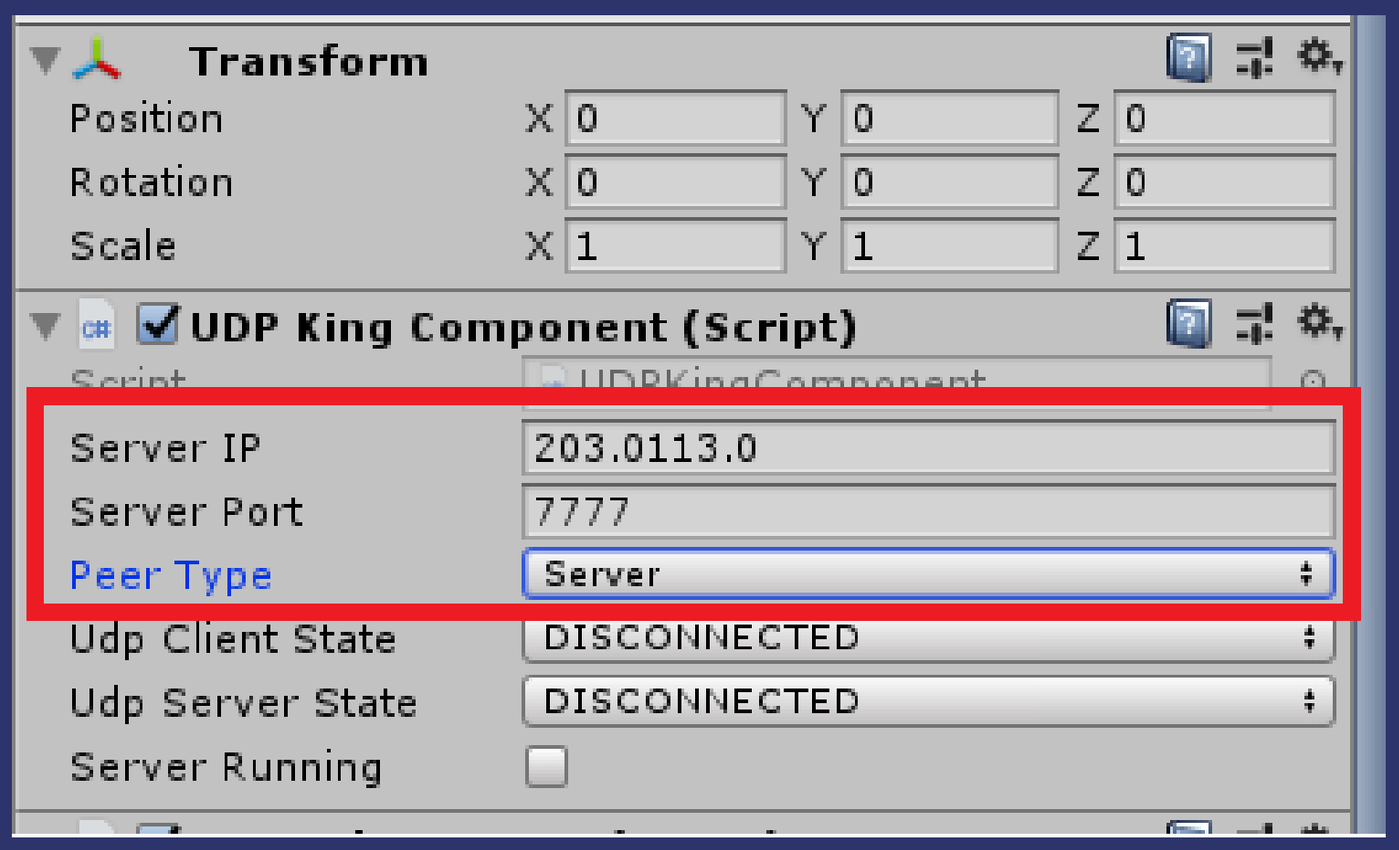
Choose an image



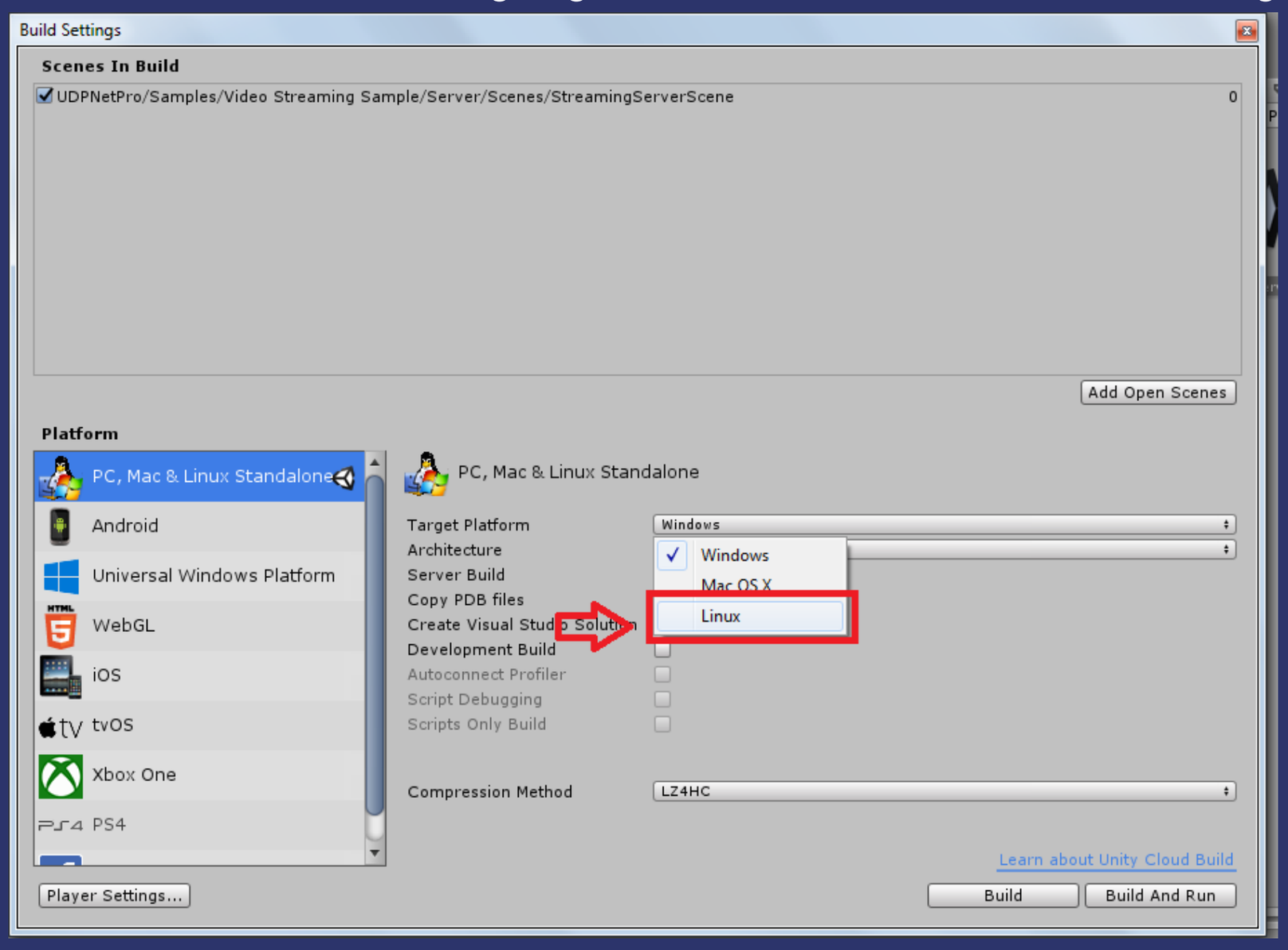
Then setup config for the droplet as you wish then press Create
After finish remember the **IP** we'll use it later



Before you build the server, in Unity you might like to set the server IP and port, you can set it in UDPKingComponent -> Server IP and Server Port fields



In this example will use Ubuntu x64, so we're going to build it as Linux in the Build Settings

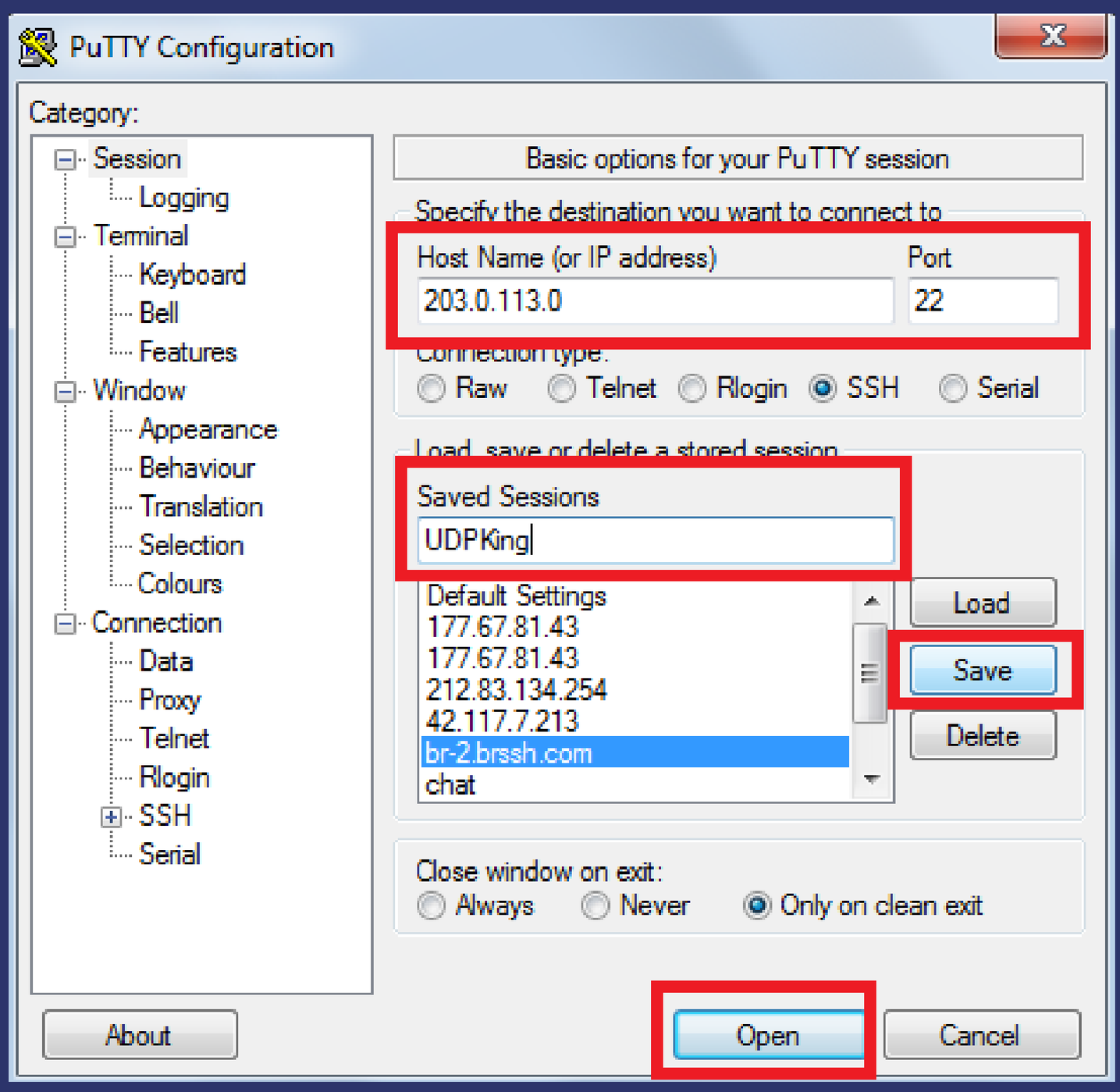


Then select the Architecture to x86_64 because thisgoing to run on Ubuntu x64 and set Server build to true

Then build it, after that we will uploads it to the digital ocean server.
Before that, we need to access our server via command prompt once to setup password
We use Putty(http://www.putty.org/) to do that

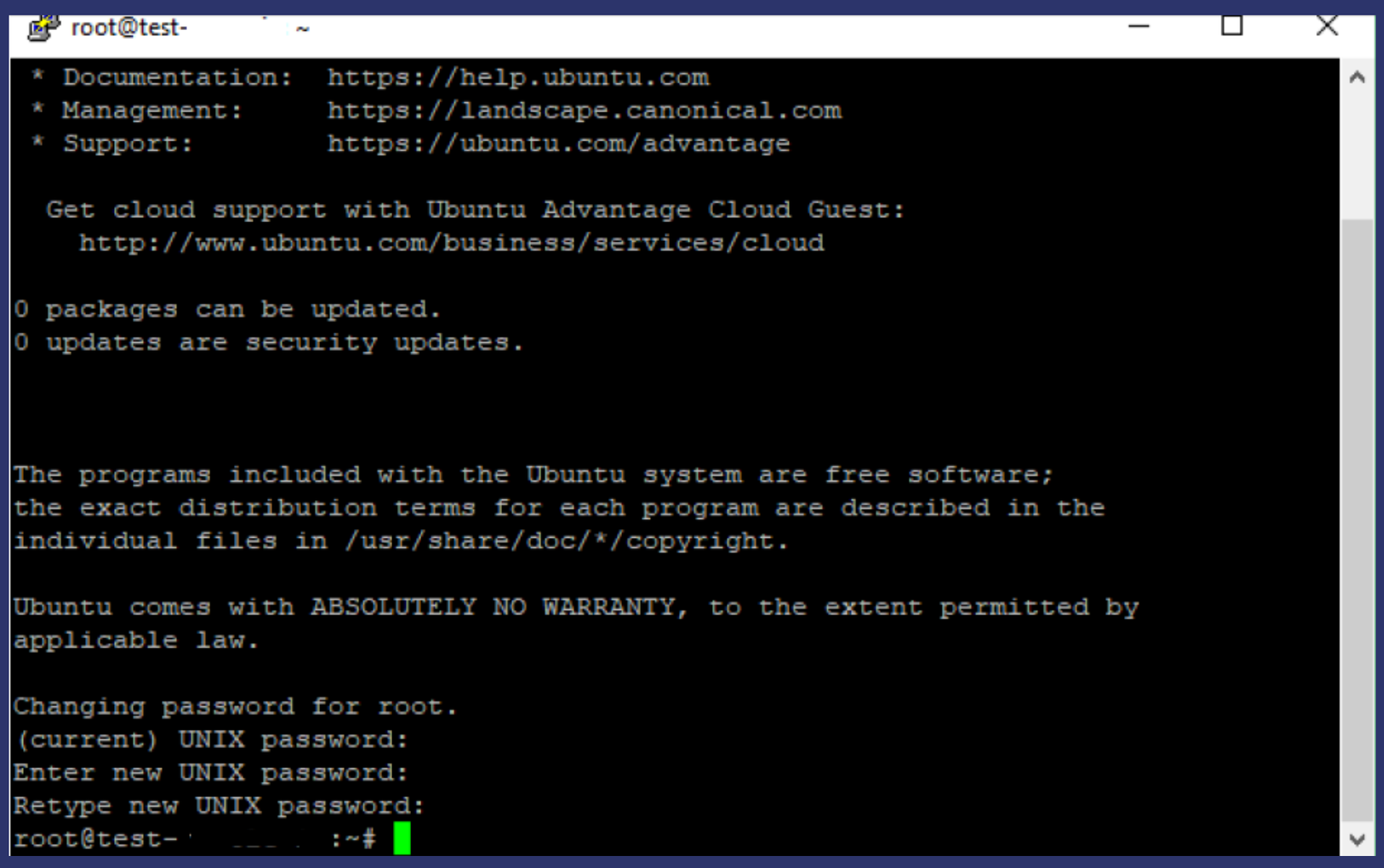
PuTTY is an open-source SSH and Telnet client for Windows. It allows you to securely connect to remote servers from a local Windows computer.

So open Putty, copy your server IP to Host Name(or IP Address) input field you may save these settings by enter Saved Sessions then press Save button. if you are ready, press Open



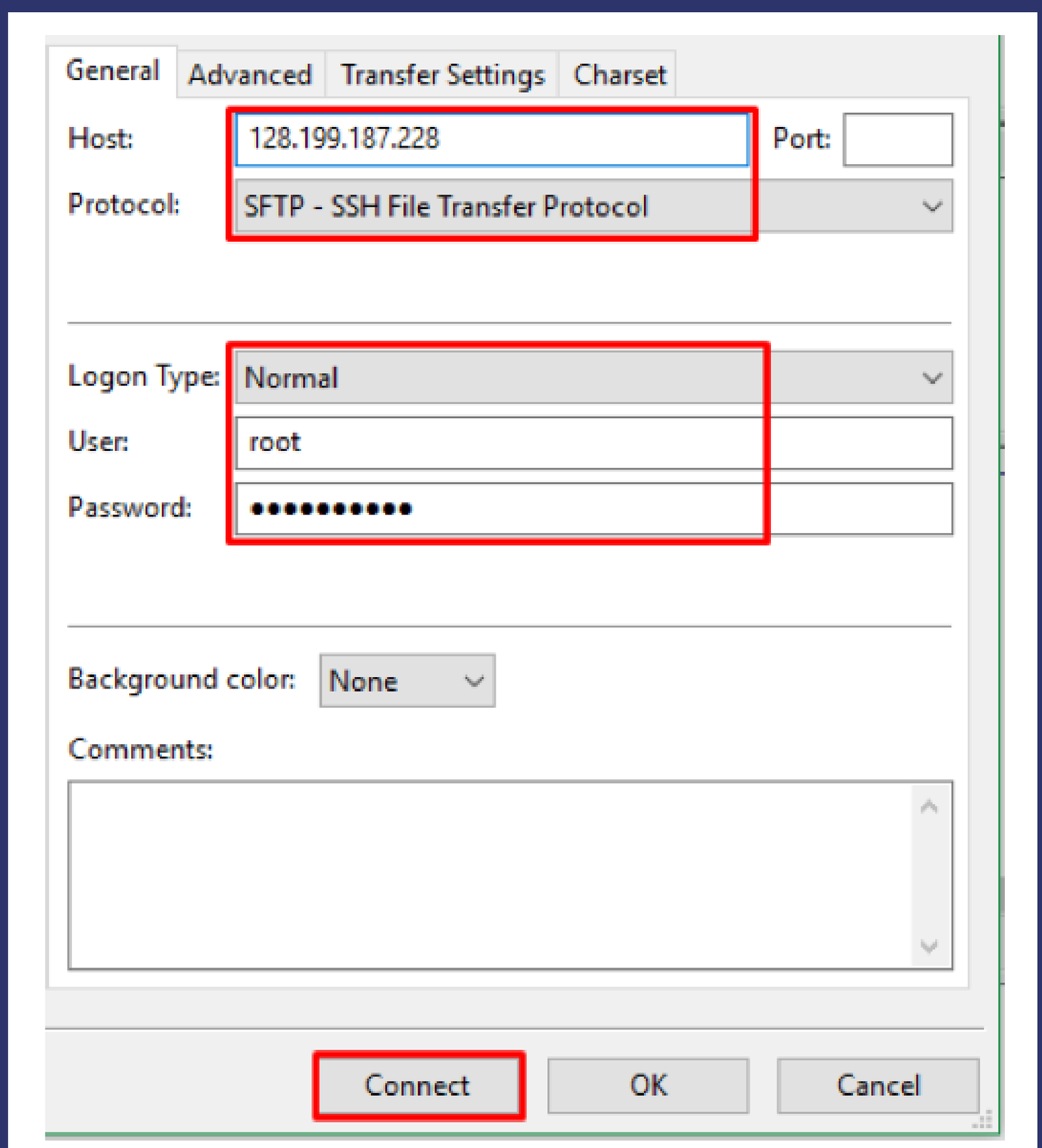
Then login as: root, it will asks for your password, you can find it in your email that you've use it for register Digital Ocean, copy it then right click in Putty to paste

Then it will request you to enter password again to create new password, so right click to do that then enter new password

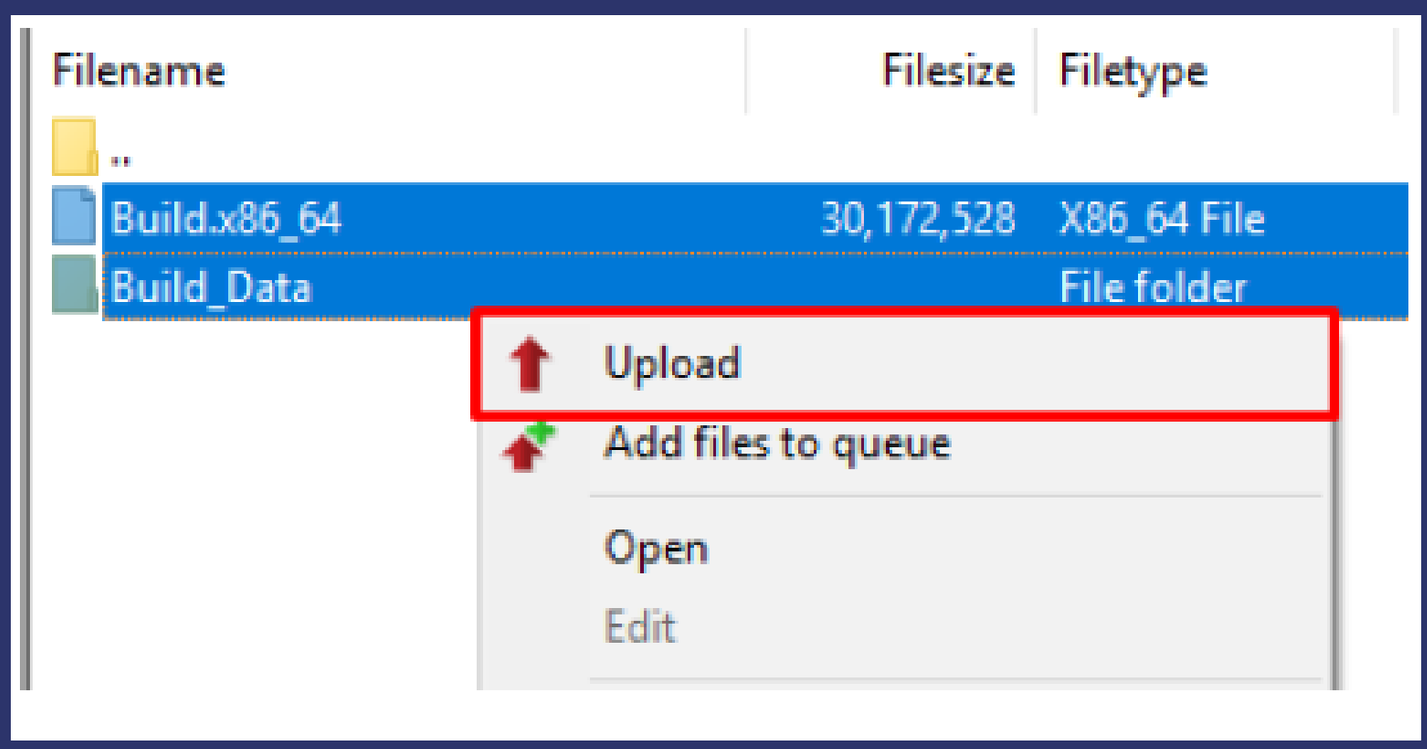


Then upload built server, I will use FileZilla Client(https://filezilla-project.org/download.php) to do that

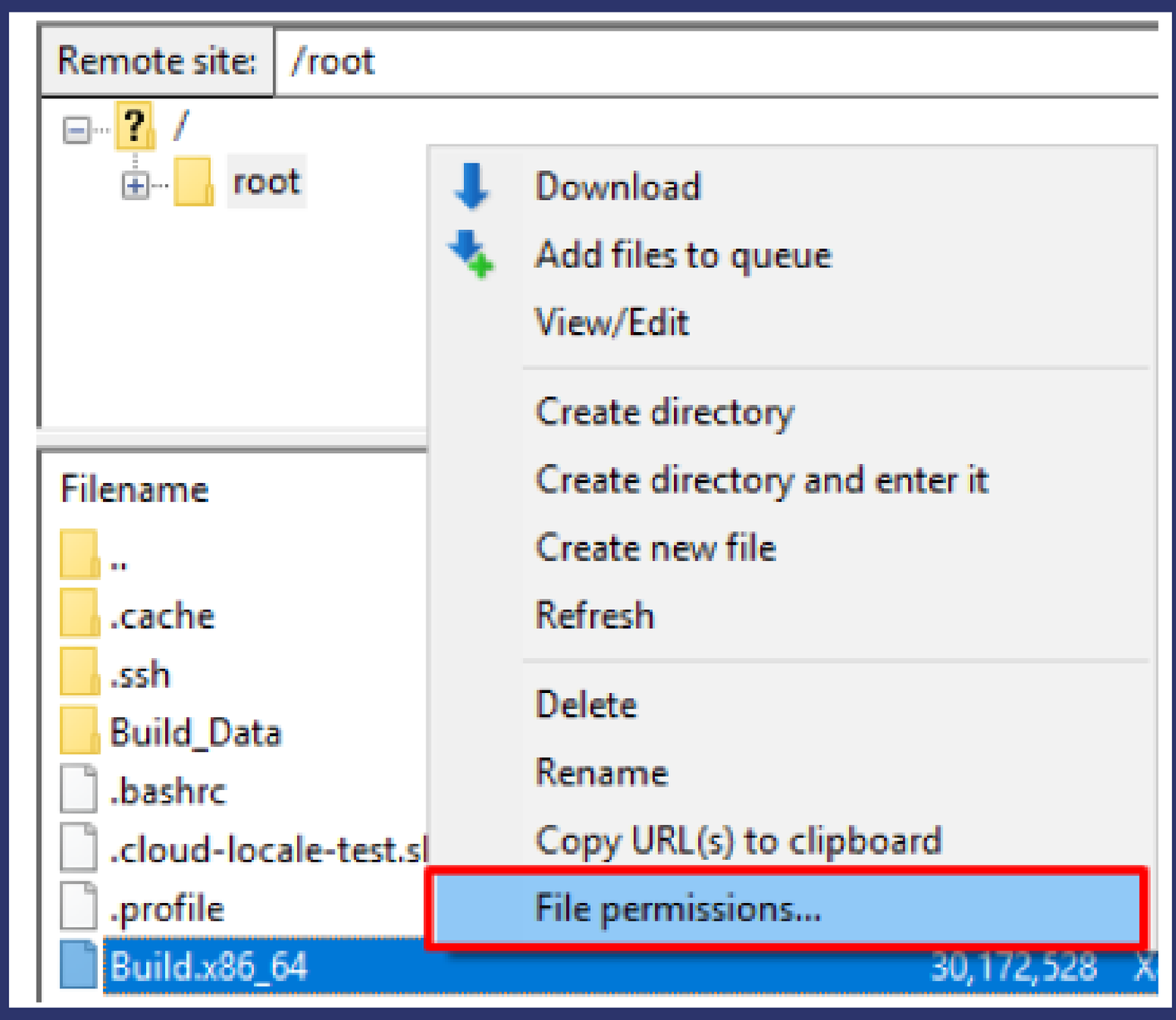
Open FileZilla Client go to menu File -> Site Manager… then press New Site button then enter its name, host, user and password, set protocol to SFTP then press Connect button



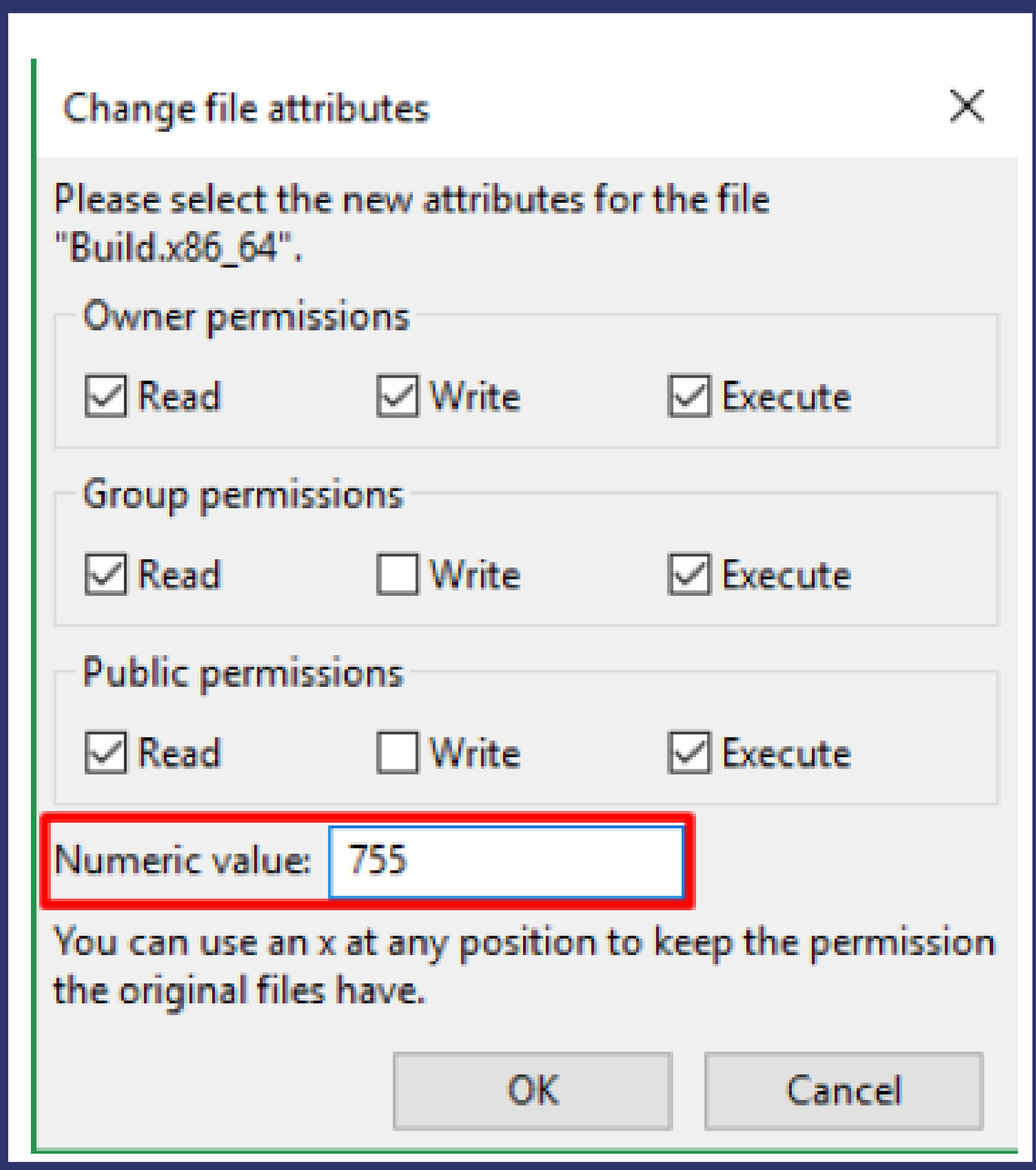
After connected access to place where you built the server from local site section then upload all files by right click selected files and press Upload



After uploaded right click on your build file at remote site section then select File permissions…



Then set Numeric value to 755



Then go back to Putty go to root folder with command

cd ~

Then run the build with command

./Build.x86_64

Then go back to your project (Unity) open game client scene at NetworkManager -> UDPKingComponent -> Server IP and Server Port IP to your IP and Online Network Port to your server port for this example it is 7777.

hands-on!

in this asset you will find the most varied examples for you to start learning how to develop your own application!

Contact us

rio3dstudios@gmail.com

rio3dstudios.com