

Lecture 18

Website hosting and WebSockets

Acknowledgement: pythonanywhere.com,
ably.com/blog/websockets-vs-long-polling,

A few options for hosting your web app

- [PythonAnywhere.com](https://pythonanywhere.com)
 - Very easy, but limited features
 - Includes free forever option
- [Heroku.com](https://heroku.com)
 - Easy to get started
 - More advanced features
 - Includes free forever option
- cloud.google.com
 - More effort to get started
 - Can do anything
 - Free option is limited to trial period

PythonAnywhere.com

- Start hosting quickly
 - Quick start installers for Django, web2py, **Flask**, and Bottle
 - Simple and integrated database hosting for MySQL and Postgres
 - No need to configure or maintain a web server
- Develop anywhere
 - Write your programs in a web-based editor
 - Run a console session from a web browser
 - Access files from anywhere

pythonanywhere.com/pricing

Plans and pricing

Beginner: Free!

A **limited account** with one web app at `your-username.pythonanywhere.com`, restricted outbound Internet access from your apps, low CPU/bandwidth, no IPython/Jupyter notebook support.
It works and it's a great way to get started!

Create a Beginner account

Education accounts

Are you a teacher looking for a place your students can code Python? You're not alone. Click through to find out more about our [Education beta](#).

Hacker \$5/month

Run your Python code in the cloud from one web app and the console

A Python IDE in your browser with unlimited Python/bash consoles

One web app on a custom domain or `your-username.pythonanywhere.com`

Enough power to run a typical **100,000 hit/day website**.
([more info](#))

2,000 CPU-seconds per day for consoles, scheduled tasks and always-on tasks
([more info](#))

IPython/Jupyter notebook support

1GB disk space

Create a Hacker account

Web dev \$12/month

If you want to host small Python-based websites for you or for your clients

A Python IDE in your browser with unlimited Python/bash consoles

Up to **2 web apps** on custom domains or `your-username.pythonanywhere.com`

Enough power to run a typical **150,000 hit/day website on each web app**.
([more info](#))

4,000 CPU-seconds per day for consoles, scheduled tasks and always-on tasks
([more info](#))

IPython/Jupyter notebook support

5GB disk space

Create a Web Developer account

Startup \$99/month

Start a business and don't worry about having to scale to handle traffic spikes

A Python IDE in your browser with unlimited Python/bash consoles

Up to **3 web apps** on custom domains or `your-username.pythonanywhere.com`

Enough power to run a typical **1,000,000 hit/day website on each web app**.
([more info](#))

10,000 CPU-seconds per day for consoles, scheduled tasks and always-on tasks
([more info](#))

IPython/Jupyter notebook support

50GB disk space

Create a Startup account

Custom \$5 to \$500/month

Want a combination that's not on the list? Create your own! All custom plans have:

A Python IDE in your browser with unlimited Python/bash consoles

Up to **20 web apps**, on custom domains or `your-username.pythonanywhere.com`

As many web workers as you need to scale your site's capacity.
([more info](#))

Up to 100,000 CPU-seconds per day for consoles, scheduled tasks and always-on tasks
([more info](#))

IPython/Jupyter notebook support

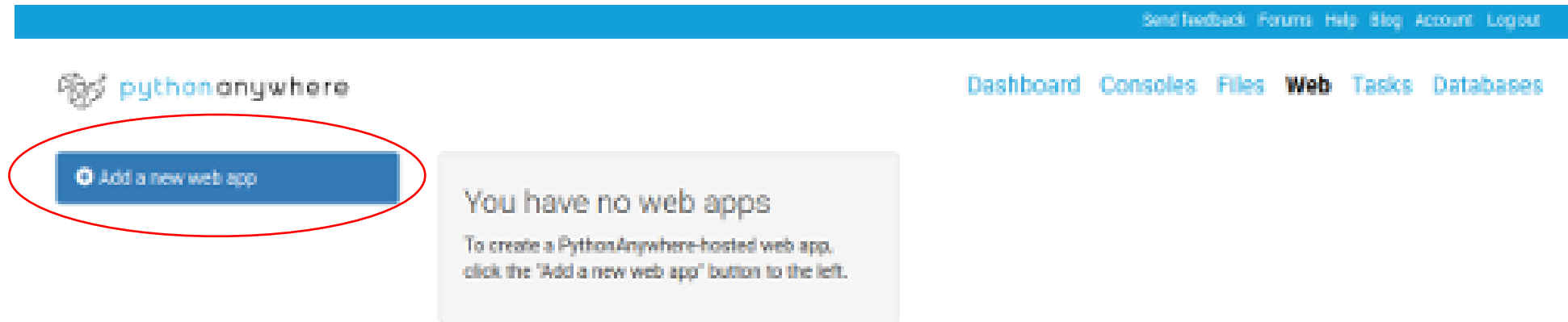
As much disk space as you choose

Create a Custom account

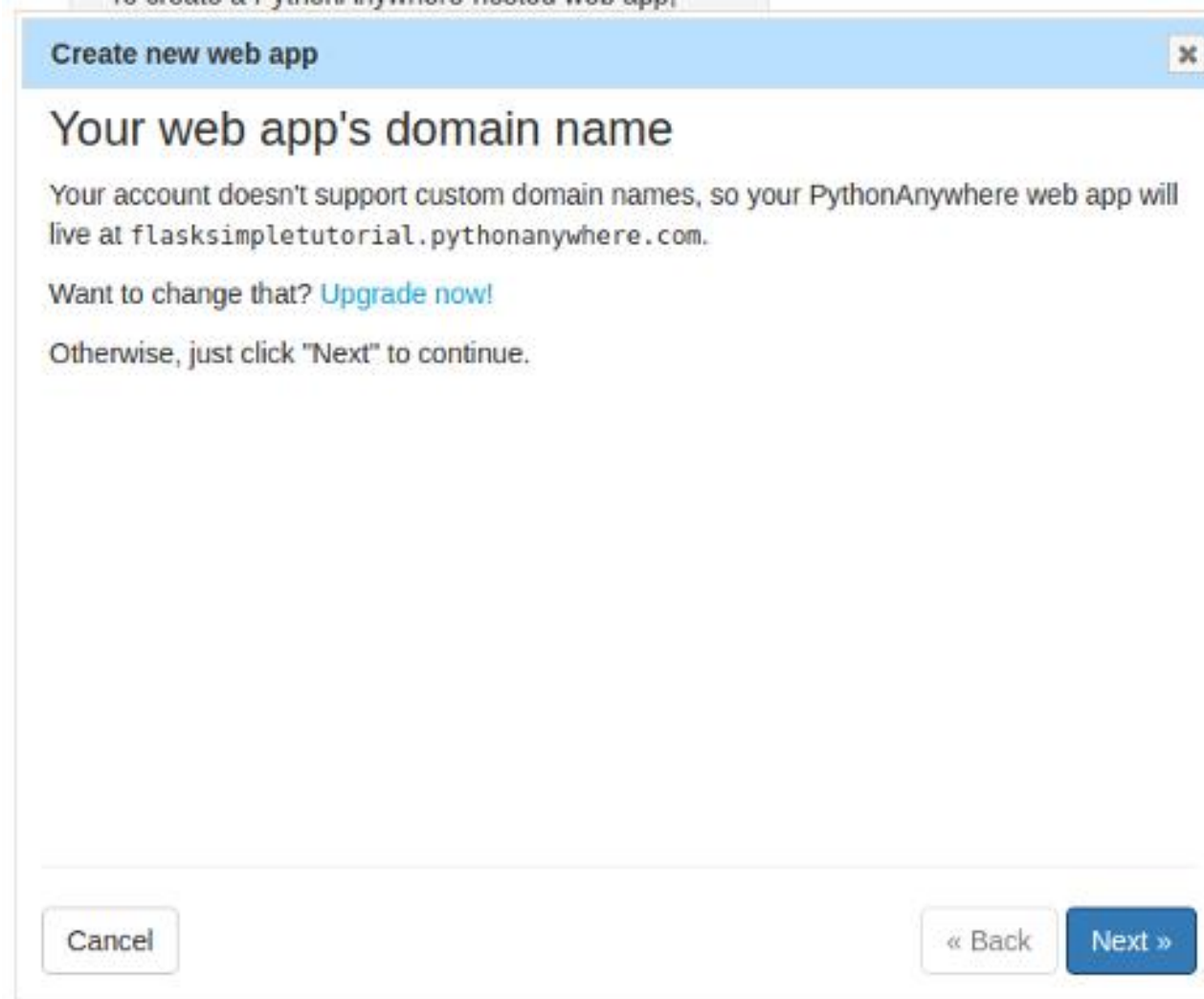
Getting started

The screenshot shows the PythonAnywhere dashboard. At the top, a blue navigation bar contains links for 'Send feedback', 'Forum', 'Help', 'Blog', 'Account', and 'Logout'. Below this, the 'pythonanywhere' logo is on the left, and a navigation menu on the right includes 'Dashboard', 'Consoles', 'Files', 'Web' (which is circled in red), 'Tasks', and 'Databases'. A yellow warning box states: 'Warning: You have not confirmed your email address yet. This means that you will not be able to reset your password if you lose it. If you cannot find your confirmation email anymore, send yourself a new one [here](#).' The main heading is 'Dashboard', and a welcome message says 'Welcome back, [flasksimpletutorial](#)'. Below this, system status is shown: 'CPU Usage: 0% used - 0.00s of 100s. Resets in 20 hours, 59 minutes' (with a 'More info' link) and 'File storage: 0% full - 0.0 KB of your 512.0 MB quota'. An 'Upgrade Account' button is in the top right. The dashboard is divided into four columns: 'Recent Consoles' (empty), 'Recent Files' (empty), 'Recent Notebooks' (empty), and 'All Web apps' (empty). Each column has a '+ Open another file' or '+ Open Web app' button. A message in the 'Recent Notebooks' column states: 'Your account does not support Jupyter Notebooks. Upgrade your account to get access!'. At the bottom left, a 'New console!' section offers '5 Bash' and '100 Python' options.

Getting started



Getting started



The screenshot shows a web browser window with a dialog box titled "Create new web app". The dialog box has a light blue header bar with the title and a close button (X). The main content area is white and contains the following text:

Your web app's domain name

Your account doesn't support custom domain names, so your PythonAnywhere web app will live at `flasksimpletutorial.pythonanywhere.com`.

Want to change that? [Upgrade now!](#)

Otherwise, just click "Next" to continue.

At the bottom of the dialog box, there are three buttons: "Cancel" on the left, and "« Back" and "Next »" on the right. The "Next »" button is highlighted in blue.

Getting started

to create a PythonAnywhere-hosted web app,

Create new web app

Select a Python Web framework

...or select "Manual configuration" if you want detailed control.

- » Django
- » web2py
- » Flask
- » Bottle
- » Manual configuration (including virtualenvs)

What other frameworks should we have here? Send us some feedback using the link at the top of the page!

Cancel

« Back

Next »

Getting started

to create a PythonAnywhere-hosted web app,

Create new web app

Select a Python version

- » Python 2.7 (Flask 0.11.1)
- » Python 3.4 (Flask 0.11.1)
- » Python 3.5 (Flask 0.11.1)
- » Python 3.6 (Flask 0.12)

Note: If you'd like to use a different version of Flask to the default version, you can use a virtualenv for your web app. There are [instructions here](#).

Cancel

« Back

Next »

Getting started


to create a PythonAnywhere-hosted web app.

Create new web app

Quickstart new Flask project

Enter a path for a Python file you wish to use to hold your Flask app. If this file already exists, its contents will be overwritten with the new app.

Path



Cancel

« Back

Next »

Getting started

The screenshot shows the PythonAnywhere dashboard. At the top, there's a blue header with links: [Get feedback](#), [Forum](#), [Help](#), [Sign Account](#), and [Logout](#). Below the header, the dashboard navigation bar includes [Dashboard](#), [Consoles](#), [Files](#), [Web](#) (selected), [Tasks](#), and [Databases](#). A green notification bar states: "All done! Your web app is now set up. Details below." The main content area is titled "Configuration for flaskimpletutorial.pythonanywhere.com". A red circle highlights this title. Below it, there's a "Reload" section with a green button labeled "Reload flaskimpletutorial.pythonanywhere.com". The "Best before date" section explains the free tier's limitations and includes a yellow button "Run until 3 months from today". A warning states: "This site will be disabled on Thursday 08 March 2018". Below that is a yellow button "Run until 3 months from today". The "Traffic" section shows a table of site activity:

How busy is your site?	
This month (previous month)	0 (0)
Today (yesterday)	0 (0)
Hour (previous hour)	0 (0)

Below the table, it says: "Want some more data? Paying accounts get pretty charts :)" The "Code" section is partially visible at the bottom.

The screenshot shows a web browser address bar with the URL `flaskimpletutorial.pythonanywhere.com`. Below the address bar, the text "Hello from Flask!" is displayed, indicating the web application is running successfully.

Update code

Code:

What your site is running.

Source code:	/home/flasksimpletutorial/mysite	➔ Go to directory
Working directory:	/home/flasksimpletutorial/	➔ Go to directory
WSGI configuration file:	/var/www/flasksimpletutorial_pythonanywhere_com_wsgi.py	
Python version:	3.6 	

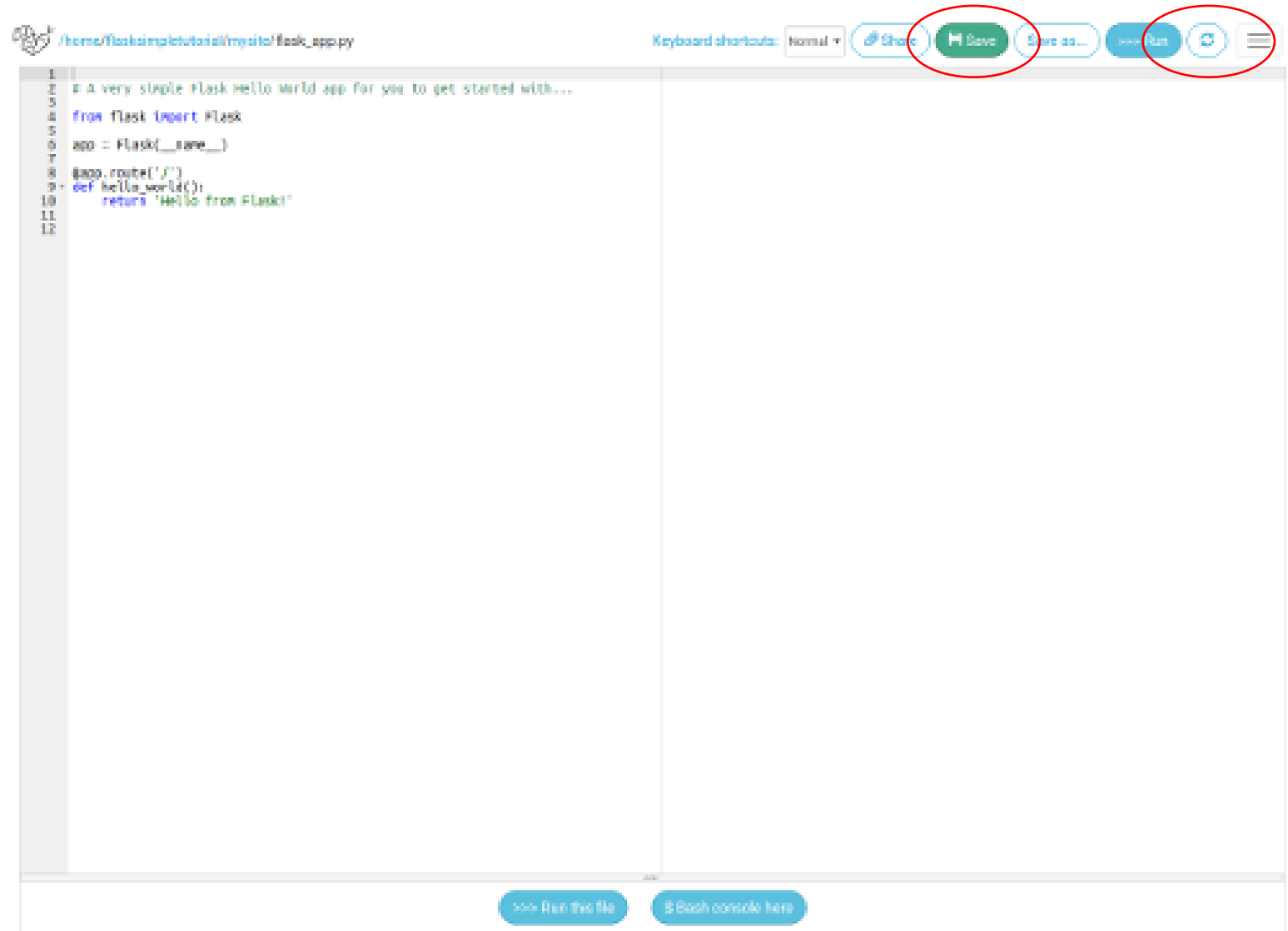
Update code

The screenshot shows the PythonAnywhere web interface. At the top, there is a blue navigation bar with links: [Send feedback](#), [Forums](#), [Help](#), [Blog](#), [Account](#), and [Logout](#). Below this, the PythonAnywhere logo is on the left, and navigation links for [Dashboard](#), [Consoles](#), [Files](#) (which is highlighted), [Web](#), [Tasks](#), and [Databases](#) are on the right. The current path is `/home/flaskimpletutorial/` with a folder icon and the text `mysite`. A status bar indicates `0% full - 72.0 KB of your 512.0 MB quota`. The interface is divided into two main sections: 'Directories' and 'Files'. The 'Directories' section has a text input 'Enter new directory name' and a 'New directory' button. The 'Files' section has a text input 'Enter new file name, eg hello.py' and a 'New file' button. Below these, there is a list of files. The first file is `__pycache__` with a folder icon. The second file is `flask_app.py`, which is highlighted with a red circle. It has a file icon, a download icon, an upload icon, and a timestamp `2017-12-08 14:52` with a size of `186 bytes`. Below the file list is a yellow button labeled 'Upload a file'.

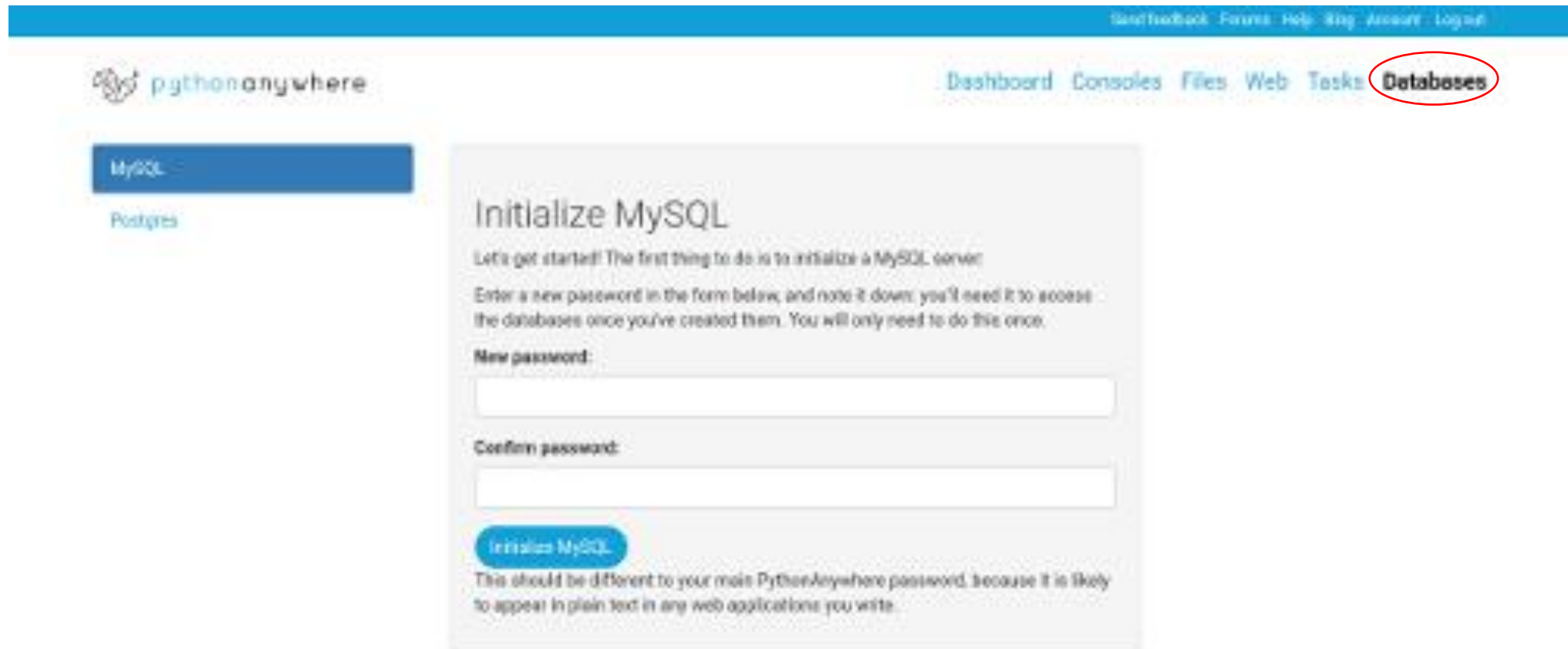
- Put your code here similar to how you have it locally

Update code

- Just save the code and reload to update your page



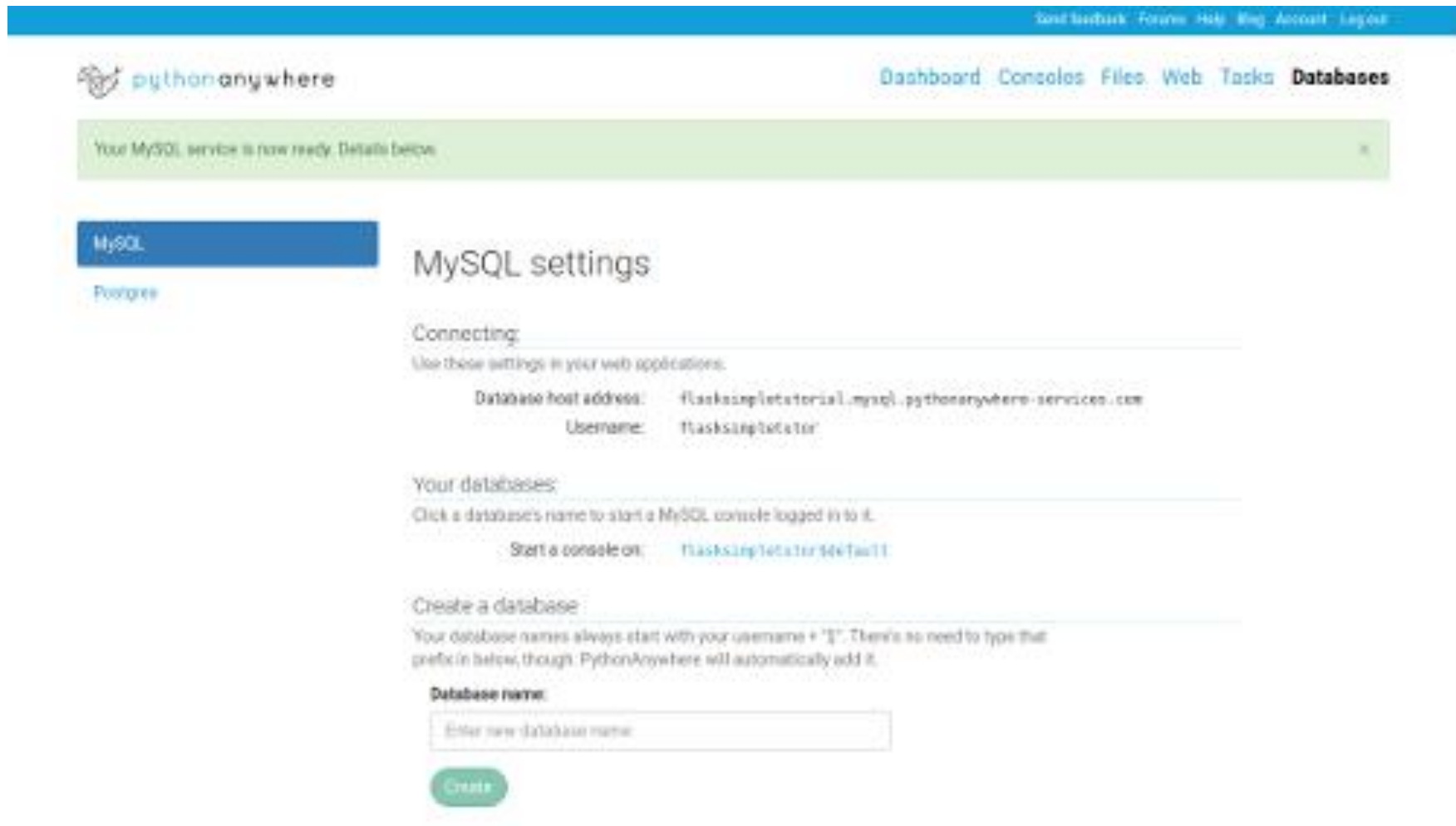
Add a database



The screenshot shows the PythonAnywhere web interface. At the top, a blue navigation bar contains links for "Feedback", "Forum", "Help", "Blog", "Account", and "Logout". Below this, the "pythonanywhere" logo is on the left, and a series of navigation links ("Dashboard", "Consoles", "Files", "Web", "Tasks", and "Databases") are on the right. The "Databases" link is circled in red. On the left side of the main content area, there are two buttons: "MySQL" (highlighted in blue) and "Postgres". The main content area is titled "Initialize MySQL" and contains the following text: "Let's get started! The first thing to do is to initialize a MySQL server. Enter a new password in the form below, and note it down: you'll need it to access the databases once you've created them. You will only need to do this once." Below this text are two input fields labeled "New password:" and "Confirm password:". At the bottom of the form is a blue button labeled "Initialize MySQL". A note at the very bottom states: "This should be different to your main PythonAnywhere password, because it is likely to appear in plain text in any web applications you write."

Create a password for your database

Add a database



The screenshot shows the PythonAnywhere web interface. At the top, a blue navigation bar contains links for 'Send feedback', 'Forum', 'Help', 'Blog', 'Account', and 'Logout'. Below this, the 'pythonanywhere' logo is on the left, and a navigation menu on the right includes 'Dashboard', 'Consoles', 'Files', 'Web', 'Tasks', and 'Databases' (which is highlighted). A green notification banner at the top of the main content area states: 'Your MySQL service is now ready. Details below'. On the left side of the main content, there is a sidebar with a blue 'MySQL' button and a 'Progress' indicator. The main content area is titled 'MySQL settings'. It includes a 'Connecting' section with instructions to use the following settings in web applications: 'Database host address: flaskinsplottutorial.mysql.pythonanywhere-services.com' and 'Username: flaskinsplottutorial'. Below this is a 'Your databases:' section with a note to click a database name to start a MySQL console, and a list showing 'Start a console on: flaskinsplottutorial447a11'. The 'Create a database' section explains that database names start with the username + '1' and provides a text input field labeled 'Enter new database name:' and a green 'Create' button.

Send feedback · Forum · Help · Blog · Account · Logout

pythonanywhere

Dashboard · Consoles · Files · Web · Tasks · **Databases**

Your MySQL service is now ready. Details below

MySQL

Progress

MySQL settings

Connecting

Use these settings in your web applications:

Database host address: flaskinsplottutorial.mysql.pythonanywhere-services.com

Username: flaskinsplottutorial

Your databases:

Click a database's name to start a MySQL console logged in to it.

Start a console on: flaskinsplottutorial447a11

Create a database

Your database names always start with your username + "1". There's no need to type that prefix in below, though. PythonAnywhere will automatically add it.

Database name:

Enter new database name:

Create

Add a database

Create a database

Your database names always start with your username + "\$". There's no need to type that prefix in below, though: PythonAnywhere will automatically add it.

Database name:

comments

Create

Add a database

Your databases:

Click a database's name to start a MySQL console logged in to it.

Start a console on: `flasksimpletutor$comments`
Start a console on: `flasksimpletutor$default`

Name includes username

Add a database

```
SQLALCHEMY_DATABASE_URI =  
"mysql+mysqlconnector://{username}:{password}@{hostname}/{databasename}".  
format(  
    username="the username from the 'Databases' tab",  
    password="the password you set on the 'Databases' tab",  
    hostname="the database host address from the 'Databases' tab",  
    databasename="yourusername$comments",  
)  
app.config["SQLALCHEMY_DATABASE_URI"] = SQLALCHEMY_DATABASE_URI  
app.config["SQLALCHEMY_POOL_RECYCLE"] = 299  
app.config["SQLALCHEMY_TRACK_MODIFICATIONS"] = False
```

Tutorial

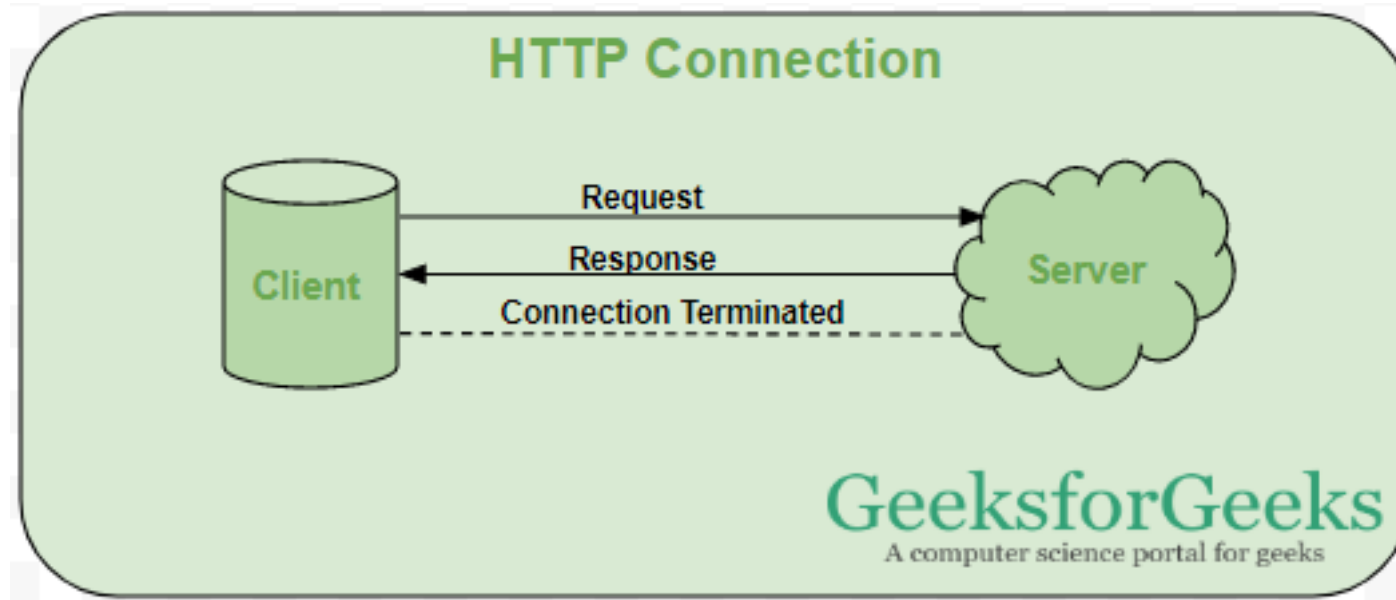
<https://blog.pythonanywhere.com/121/>

Wake-up!

- <https://youtu.be/fCi0KrxulgY>

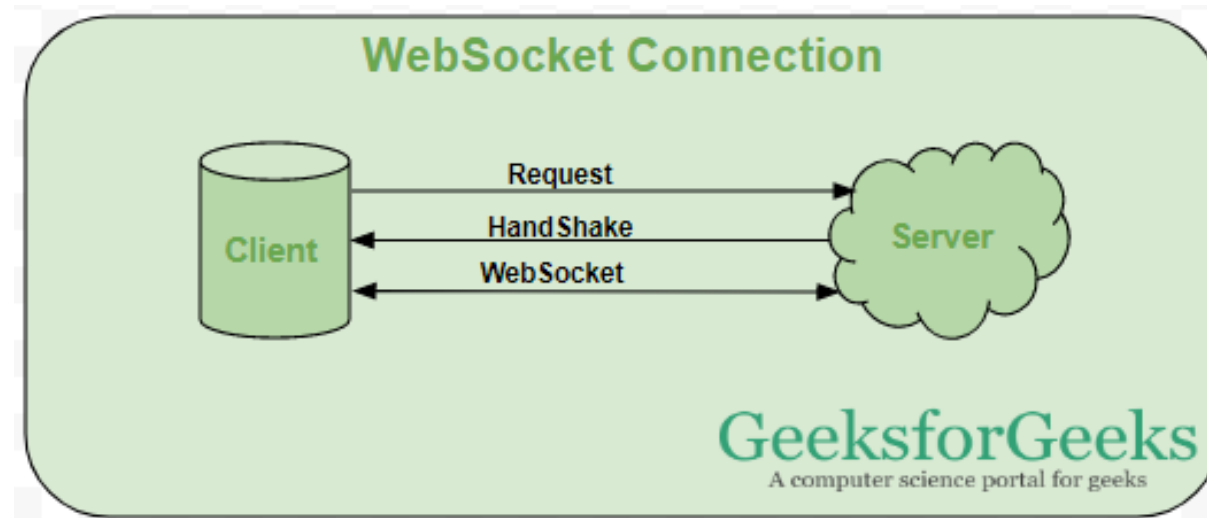
WebSocket

- HTTP is a unidirectional protocol where the client sends a request, and a server sends a response and the connection is closed



WebSocket

- WebSocket is a bidirectional protocol where the client or server can send messages initiated at anytime (uses ws instead of http)
- WebSocket has a persistent connection after the handshake which stays alive until it is terminated by the client or server



WebSocket

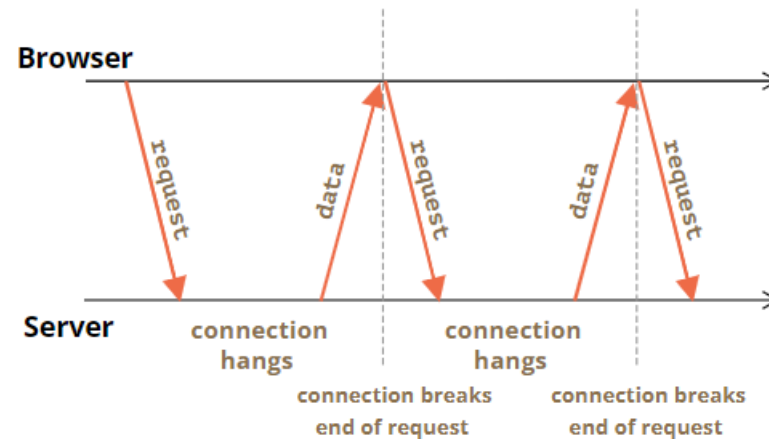
- When can a web socket be used:
 - **Real-time web application:** uses a WebSocket to show the data at the client end, which is continuously being sent by the backend server
 - **Gaming application:** data is continuously received by the server and without refreshing the UI, it will take effect on the screen, UI gets automatically refreshed without even establishing the new connection
 - **Chat application:** uses a WebSocket to establish the connection once for exchange, publishing, and broadcasting the message among the subscriber
- When not to use WebSocket:
 - Any standard situation where it can be handled with a REST API

Long Polling

- A precursor (and alternate) to WebSocket
- A simple way to have a persistent connection with the server
- Uses standard HTTP
- Better than regular polling which pings the server on an interval
- Downsides to regular polling
 - Requires a lot of useless server requests
 - Messages are delayed by time interval

Long Polling

- The flow:
 1. A request is sent to the server
 2. The server doesn't close the connection until it has a message to send
 3. When a message appears – the server responds to the request with it
 4. The browser makes a new request immediately



Why WebSocket is usually better

- Long polling is much more resource intensive on servers because they require the overhead of headers for each message sent and received
- Message ordering cannot be guaranteed with long polling if the same client opens multiple connections to the server
- Websockets work in 97% of all browsers as of 2020

Flask-SocketIO on backend

- Provides access to low-latency two-way client-server communication for Python Flask apps
- `pip install flask_socketio`
- `pip install flask_cors`

Flask-SocketIO on backend

```
from flask import Flask, render_template
from flask_socketio import SocketIO, emit
from flask_cors import CORS

app = Flask(__name__)
CORS(app)
socketio = SocketIO(app)
socketio.init_app(app, cors_allowed_origins="*", logger=True, engineio_logger=True)

@app.route('/')
def index():
    return render_template('index.html')

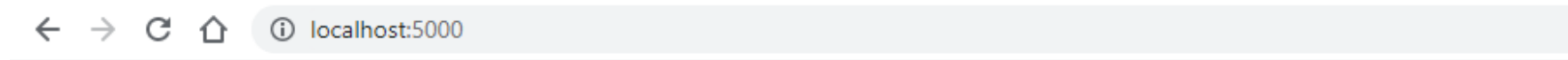
@socketio.on('connect')
def connect():
    emit('connect', {'data': 'Connected'})

if __name__ == '__main__':
    socketio.run(app)
```

Socket.IO JS library on front end

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <script src="https://code.jquery.com/jquery-3.3.1.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/4.3.0/socket.io.js"></script>
    <script type="text/javascript">
      const socket = io("ws://localhost:5000");
      socket.on("connect", function(msg) {
        if(msg != undefined) {
          $('#log').append('<br>' + $('<div/>').text('Received: ' + msg.data).html());
        }
      });
    </script>
  </head>
  <body>
    <h2>SocketIO Example</h2>
    <div id="log"></div>
  </body>
</html>
```

Flask-SocketIO



SocketIO Example

Received: Connected

Flask-SocketIO Echo (Send/Receive)

```
# This receives a message sent with 'echo' and sends a message with 'echo'
@socketio.on('echo')
def echo(message):
    print("received", message)
    emit('echo', message)
```


Flask-SocketIO Echo (Send/Receive)

```
//send
function echo(){
    socket.emit("echo", $('#msg').val());
}

//reecieve
socket.on("echo", msg => {
    $('#log').append('<br>' + $('#div/>').text('Received: ' + msg).html());
});
```

```
<body>
  <h2>SocketIO Example</h2>
  <button onClick="echo()">Echo</button>
  <input type="text" id="msg">
  <div id="log"></div>
</body>
```

Resources

- <https://socket.io/docs/v4>
- <https://flask-socketio.readthedocs.io>