

Deployment 4 - Documentation

First I started my EC2 Instance that has Jenkins

I then logged into my AWS account w/ Admin Privileges and **created an IAM user**.
I went to IAM aws service and selected users section

▼ Access management

User groups

Users

I then added a new user

I called it "**Jenkins-user**" and gave it **Programmatic access**


Select AWS access type


Select how these users will access AWS. Access keys and autogenerated passwords are provided in the last step. [Learn more](#)


- Access type* ☒ **Programmatic access**
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.
- ☐ **AWS Management Console access**
Enables a **password** that allows users to sign-in to the AWS Management Console.

I then selected Attach existing policies directly

▼ Set permissions


 Add user to group

 Copy permissions from existing user

 Attach existing policies directly


Create policy

And Selected


<input type="checkbox"/>	▶	 AdministratorAccess-AWSElasticBeanstalk	AWS managed	Pe
--------------------------	---	---	-------------	----



Tags are not necessary

Once you create the user, **download the .csv** that has the user name, Access Key ID and secret access key

 **Success**
You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.

Users with AWS Management Console access can sign-in at: <https://rixardo-aws.signin.aws.amazon.com/console>

 **Download .csv**

	User	Access key ID	Secret access key
▶ 	test	AKIATVTG2RK6CP4CCRW3 	***** Show


Create an Elastic Beanstalk next

Application information

Application name

Up to 100 Unicode characters, not including forward slash (/).

Application tags

Apply up to 50 tags. You can use tags to group and filter your resources. A tag is a key-value pair. The key must be unique within the resource and is case-sensitive. [Learn more](#) 

Key	Value	
<input type="text" value="environment"/>	<input type="text" value="dev-app"/>	<input type="button" value="Remove tag"/>

49 remaining

Platform

Platform

Python

▼

Platform branch

Python 3.8 running on 64bit Amazon Linux 2

▼

Platform version

3.3.4 (Recommended)

▼

Application code

☒ Sample application

Get started right away with sample code.

☐ Upload your code

Upload a source bundle from your computer or copy one from Amazon S3.

Cancel

Configure more options

Create application

OPTIONAL

Configure more options

Select configure more options and select Security. Add your Key pair to SSH into beanstalk EC2

Security

Service role: `arn:aws:iam::[REDACTED]:role/aws-elasticbeanstalk-service-role`

Virtual machine key pair: --

Virtual machine instance profile: `aws-elasticbeanstalk-ec2-role`

Edit

Modify security

Service role

Service role

aws-elasticbeanstalk-service-role ▼ ↻

Virtual machine permissions

EC2 key pair

rixardo ▼ ↻

IAM instance profile

aws-elasticbeanstalk-ec2-role ▼ ↻

Cancel

Save

Save and then press create APP

Go into Jenkins and manage Jenkins -> manage Plugins

Install the following plugins

AWSEB Deployment Plugin

CloudBees Credentials Plugin

I then created a Deployment 4 folder that has 3 separate folders inside with each name based on the application.

Useful Commands

```
python3 -m venv venv
```

```
pip freeze > requirements.txt
```

```
source ./Scripts/Activate
```

```
FLASK_APP=app.py flask run
```

Now create a Freestyle Project inside the corresponding application folder.

Source Code Management

☐ None
☒ Git

Repositories

Repository URL

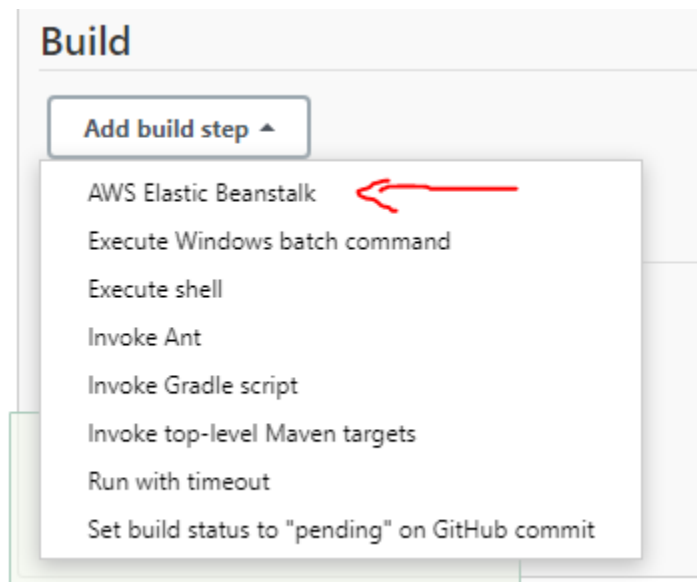
Credentials

Make sure the branch is set correctly. (If its master, change it to master)

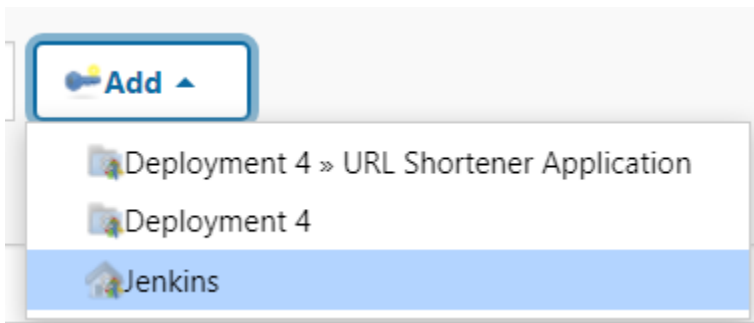
Branches to build

Branch Specifier (blank for 'any')

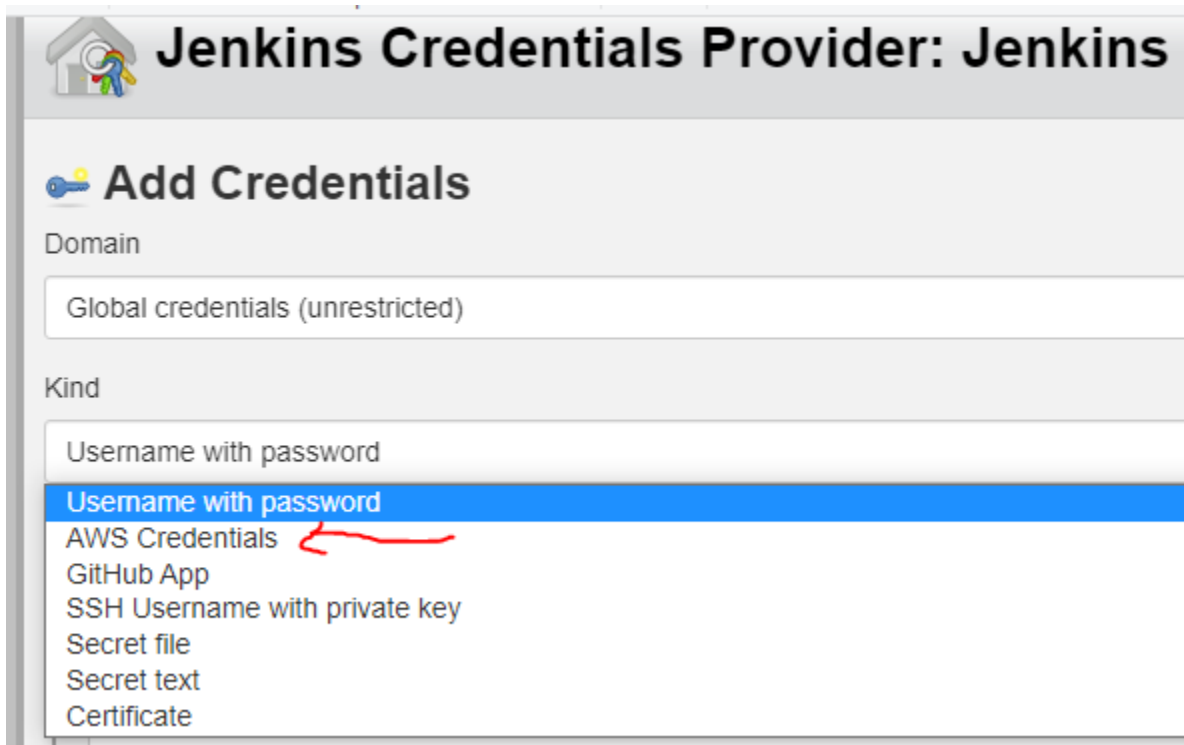
Scroll down to Build and select AWS Elastic Beanstalk



To create a new credentials select Jenkins



Under kind, select AWS Credentials

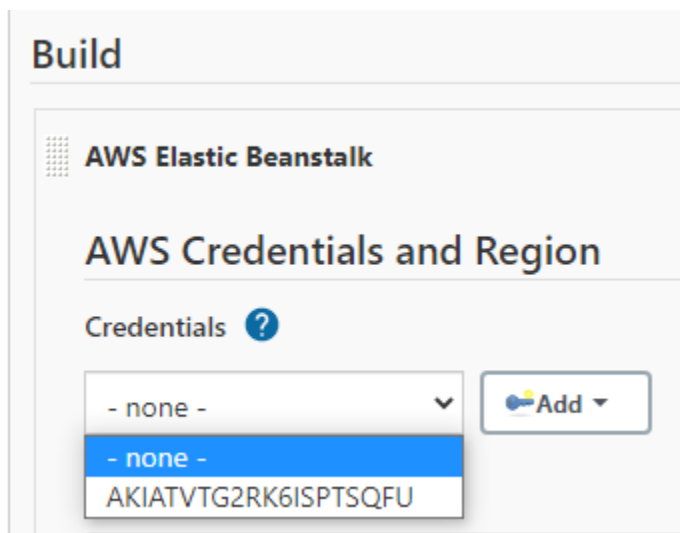


The screenshot shows the Jenkins 'Add Credentials' page. The 'Domain' is set to 'Global credentials (unrestricted)'. The 'Kind' dropdown menu is open, showing options: 'Username with password', 'AWS Credentials', 'GitHub App', 'SSH Username with private key', 'Secret file', 'Secret text', and 'Certificate'. A red arrow points to 'AWS Credentials'.

For ID just put a “**Jenkins-user**”

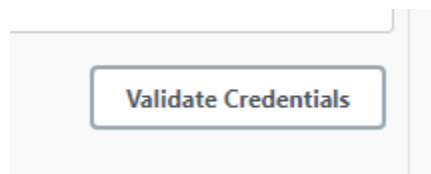
For **Access KEY ID** and **Secret Access Key** - Use the credentials from the IAM user you downloaded.

Press Add and now select the credentials from the dropdown



The screenshot shows the 'Build' configuration page for 'AWS Elastic Beanstalk'. Under the 'AWS Credentials and Region' section, the 'Credentials' dropdown menu is open, showing options: '- none -' and 'AKIATVTG2RK6ISPTSQFU'. An 'Add' button is visible next to the dropdown.

Select the AWS Region where the Elastic BeanStalk was created
Then press **Validate Credentials** in the bottom right



You should see a response like this...

- Building Client (credentialId: 'Jenkins-user', region: 'us-east-1')
- Testing Amazon S3 Service (endpoint: https://s3.amazonaws.com)
- Buckets Found: 4
- Testing AWS Elastic Beanstalk Service (endpoint: https://elasticbeanstalk.us-east-1.amazonaws.com)
- Applications Found: 1 (youtube)

Go back to AWS Beanstalk and find the environment you created. Make a note of the **Environment name** and **application name**

Environment name ▲	Health ▼	Application name
Youtube-env	Ok	youtube

Scroll down to Application and Environment and enter the correct info from AWSEB,

Application and Environment

Application Name ?
youtube

Environment Name(s) ?
Youtube-env

Then press validate coordinates

Validate Coordinates

You should see something like this..

Environment found (environmentId: ~~xxxxxxxxxx~~)

In Packaging, add a “.” to the Root object

Packaging

Root Object (File / Directory) ?

.

In versioning put **python-01\${BUILD_ID}**

Versioning

Version Label Format ?

python-01\${BUILD_ID}

Then press SAVE and **BUILD NOW**



Build Now

Should take about ~5 minutes to build successfully

Once you have a successful build, Go back to AWS Elastic Bean and click on the **URL** for your environment.

URL Shortener

Source: https://github.com/Deodutt/DEPLOY4_FLASK_APP

Successful Build:



Default Web Page:

Website
Short Name
<input type="text" value="Ricardo"/>
Website URL
<input type="text" value="https://www.linkedin.com/in/rixardo/"/>
<input type="button" value="Shorten"/>

Web page once I enter information and press shorten

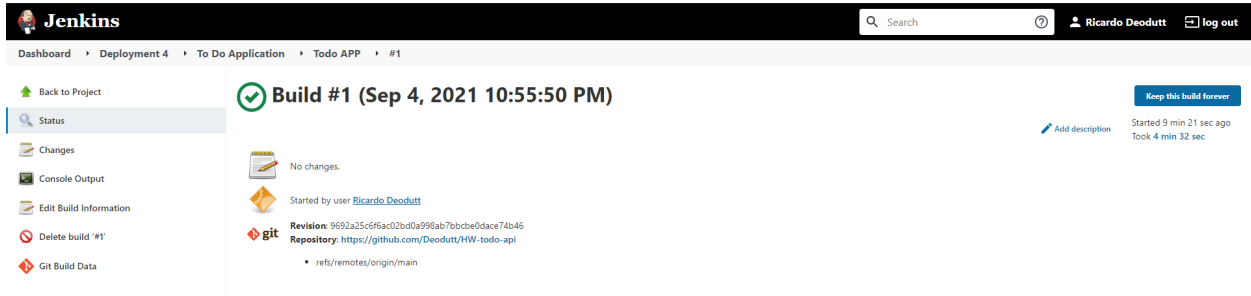
URL Created

This takes me to my LinkedIn Page

Todo

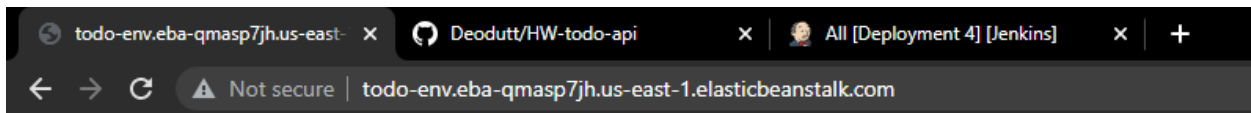
Source: <https://github.com/Deodutt/HW-todo-api>

Successful Build:



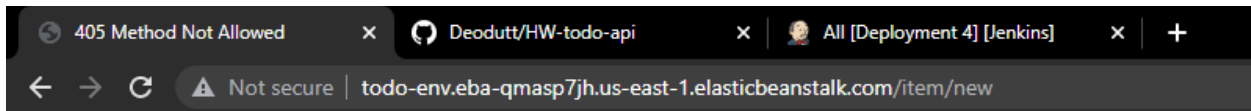
The screenshot shows the Jenkins web interface. The top navigation bar includes the Jenkins logo, a search bar, and the user name 'Ricardo Deodutt' with a 'log out' button. The breadcrumb trail is 'Dashboard > Deployment 4 > To Do Application > Todo APP > #1'. The left sidebar contains links: 'Back to Project', 'Status' (selected), 'Changes', 'Console Output', 'Edit Build Information', 'Delete build #1', and 'Git Build Data'. The main content area displays 'Build #1 (Sep 4, 2021 10:55:50 PM)' with a green checkmark icon. It includes a 'Keep this build forever' button, an 'Add description' link, and build statistics: 'Started 9 min 21 sec ago' and 'Took 4 min 32 sec'. Below this, it shows 'No changes' and 'Started by user Ricardo Deodutt'. A Git icon indicates the build source: 'Revision: 9692a25c6f6ac02bd0a998ab7bbcb0dace74b46' and 'Repository: https://github.com/Deodutt/HW-todo-api', with a link to 'refs/remotes/origin/main'.

Route: /



The browser window shows three tabs: 'todo-env.eba-qmasp7jh.us-east-1', 'Deodutt/HW-todo-api', and 'All [Deployment 4] [Jenkins]'. The address bar displays 'Not secure | todo-env.eba-qmasp7jh.us-east-1.elasticbeanstalk.com'. The page content shows 'Hello World!' in a large, bold, black serif font.

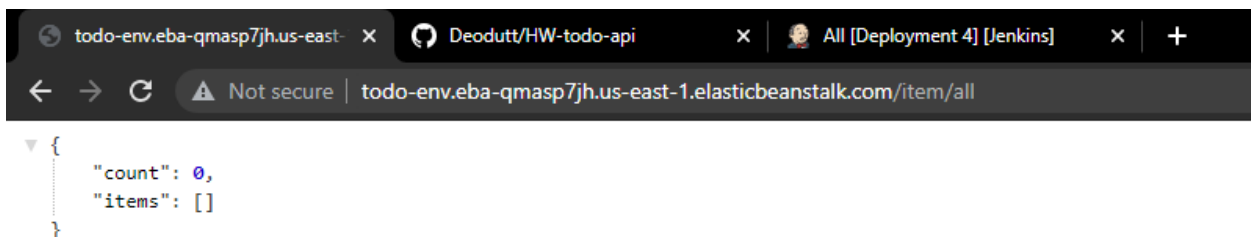
Route: /item/new



The browser window shows three tabs: '405 Method Not Allowed', 'Deodutt/HW-todo-api', and 'All [Deployment 4] [Jenkins]'. The address bar displays 'Not secure | todo-env.eba-qmasp7jh.us-east-1.elasticbeanstalk.com/item/new'. The page content shows 'Method Not Allowed' in a large, bold, black serif font.

The method is not allowed for the requested URL.

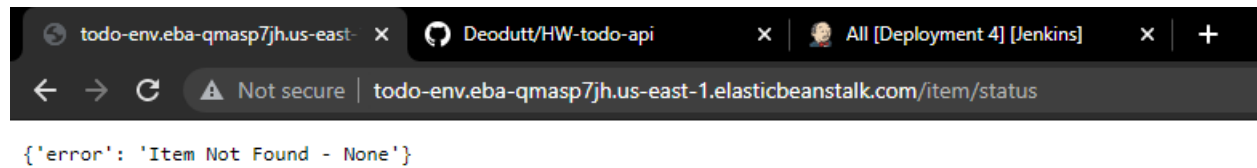
Route: /item/all



The browser window shows three tabs: 'todo-env.eba-qmasp7jh.us-east-1', 'Deodutt/HW-todo-api', and 'All [Deployment 4] [Jenkins]'. The address bar displays 'Not secure | todo-env.eba-qmasp7jh.us-east-1.elasticbeanstalk.com/item/all'. The page content shows a JSON response in a code block:

```
{
  "count": 0,
  "items": []
}
```

Route: /item/status

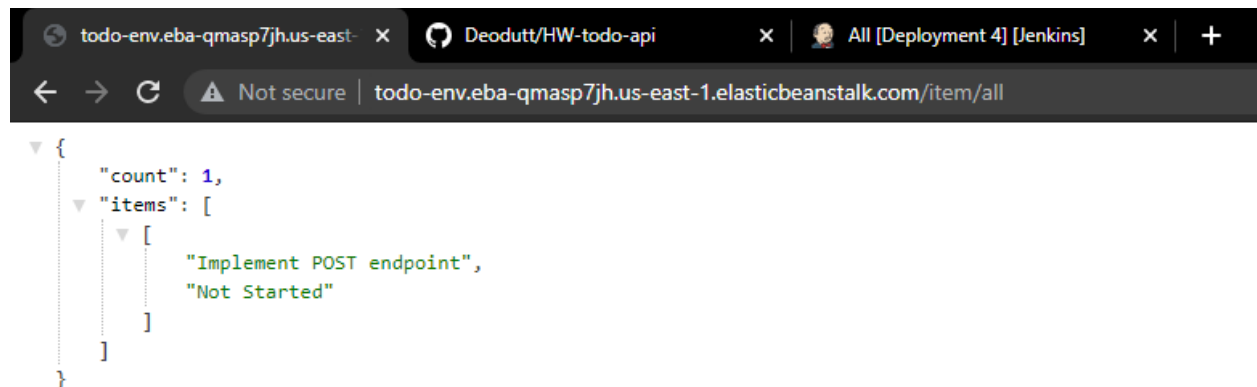


To update the database run the following commands...

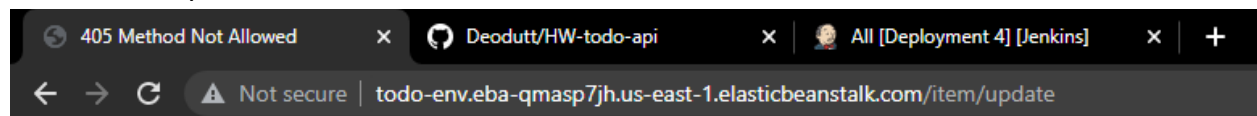
```
curl -X POST
http://todo-env.eba-qmasp7jh.us-east-1.elasticbeanstalk.com/item/new -d
'{"item": "Implement POST endpoint"}' -H 'Content-Type: application/json'
```

```
robin@robin MINGW64 ~
$ curl -X POST http://todo-env.eba-qmasp7jh.us-east-1.elasticbeanstalk.com/item/new -d '{"item": "Implement POST endpoint"}' -H 'Content-Type: applica
tion/json'
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload  Total   Spent    Left   Speed
100    95    100    60    100    35     587    342  --:--:--  --:--:--  --:--:--   940{"item": "Implement POST endpoint", "status": "Not Started"}
```

Route: /item/all



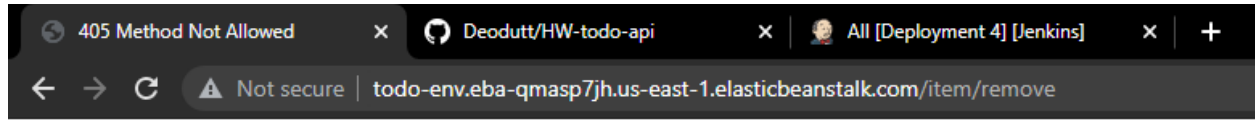
Route: /item/update



Method Not Allowed

The method is not allowed for the requested URL.

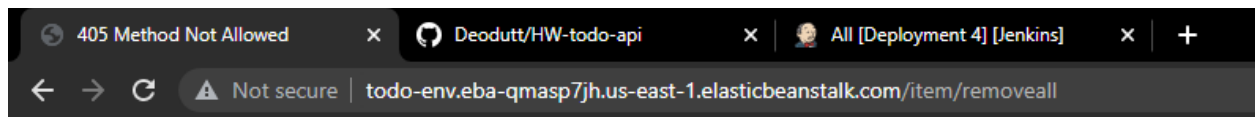
Route: /item/remove



Method Not Allowed

The method is not allowed for the requested URL.

Route: /item/removeall



Method Not Allowed

The method is not allowed for the requested URL.

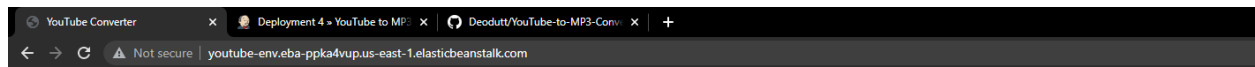
YouTube

Source: <https://github.com/Deodutt/YouTube-to-MP3-Converter-API>

Successful Build



Default web page when I click on the URL from Elastic Beanstalk



Welcome to Youtube to MP3 Converter!

Please enter a link you want to convert!



YouTube ID is None

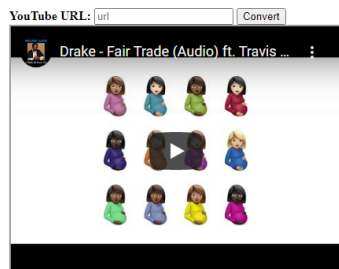
Error: No Streams found.

Web page once I enter an URL and press convert.



Welcome to Youtube to MP3 Converter!

Please enter a link you want to convert!

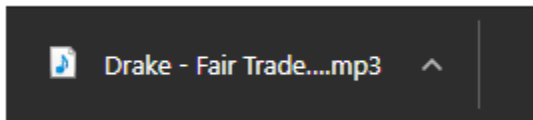


Youtube link is <https://www.youtube.com/watch?v=THVbtGqEO1o>

YouTube ID is THVbtGqEO1o

MP3	MP3	MP3	MP3
320 kbps	256 kbps	192 kbps	128 kbps
11.14 MB	8.91 MB	6.68 MB	4.46 MB

Successful download after selecting a download option



YouTube Converter

(This is for educational purposes)

As a person who listens to music a lot, I wanted to make this application to make my life a little easier. I prefer to work on projects that benefit me. For example, I made a VPN application so I can use it. I prefer to download music online and put it on my phone rather than using streaming services. I do not like paywalls so I thought this was a great project for me. It's a weird way of listening to music but I have been doing it for a long time since Limewire days.

While looking up ideas, I faced some issues. I tried looking up official YouTube APIs to download videos but I found out it's not supported because it's against their terms of services so I had to look for different options. I came across this [repository](#) and thought it would be great to add on. The repo creator supplies an API that I could use to embed into a website. It gives the user different options to download videos in different qualities. While creating the application, I used Ibrahima's [repository](#) as a template. His repository was a great way to understand how to write to HTML files and use variables in them. Everything was simple and easy to understand. While creating the application, I found this really great way of [getting the YouTube ID](#) and thought it would be great to incorporate it.

For my application, once a user enters the URL inside the web page and presses convert, it would do a GET method. This would then call the converter() function inside of application.py and that would get the string the user entered. The application will then call a id_grabber function from the helper.py file, and that will use urlparse to parse the URL and extract the youtube ID from the URL. It goes to many different cases to incase the user enters a different youtube link format. Once it gets the ID it will return the ID in a variable called youtube_id. The youtube_id will then concat with a string to the API and be assigned to a variable called api_converter_link. This is then returned into a function called render_templates() which basically renders a template from the template folder which has an index.html file. This will basically let the HTML page use variables.

example

```
{% if youtube_id %}
<p>YouTube ID is {{youtube_id}}</p>
{% endif %}
```

URL Shortener Application: https://github.com/Deodutt/DEPLOY4_FLASK_APP

To Do Application: <https://github.com/Deodutt/HW-todo-api>

YouTube Converter Application: <https://github.com/Deodutt/YouTube-to-MP3-Converter-API>