Topic	Video Chat App - DevOps	
Class Description	**Student will learn about DevOps, i.e. Development and Operations, which means, they will be deploying an application on a remote Heroku Server!**	
Class	**C-218**	
Class time	**45 mins**	
Goal	• Learning about PeerJS • Implementation of PeerJS and PeerJS server	
Resources Required	• Teacher Resources: ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ○ Visual Studio Code • Student Resources: ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ○ Visual Studio Code	
Class structure	**Warm-Up** **Teacher - led Activity 1** **Student - led Activity 1** **Wrap-Up**	**10 mins** **15 mins** **15 mins** **5 mins**
WARM UP SESSION - 10mins		
Teacher Action		**Student Action**

Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today?	**ESR**: Hi, thanks, yes, I am excited about it!
Q&A Session	
Question	**Answer**
Why should web apps be deployed? A. To make it available for all B. To make it for two or three people C. To use it local D. None of the above	**A**
Why do we use github? A. To make code repositories and to maintain code versions B. To make pdf repositories C. To make folder repositories D. None of the above	**A**
TEACHER-LED ACTIVITY - 15mins	
Teacher Initiates Screen Share	
ACTIVITY • **Understanding about DevOps Engineering** • **Creating a master github repository**	
Teacher Action	**Student Action**
In the last class, we successfully implemented PeerJS, where we linked the functionality of our Rooms with Sockets, and also added users to a chatroom when the peers open the same page on the browser.	

Do you have any doubts in the last class? *Teacher clears the doubts, if any* Great! We encountered a problem in the last class, which was that because PeerJS makes a lot of requests back and forth from client to server, it is impossible to test it on NGROK! Also, we were running the app locally, but the PeerJS server that we created wanted to use runs on port 443, which we don't have access to locally, since port 443 is HTTPS! With all these issues, how do you think we can solve our problem? One universal solution is to deploy our application on a remote server, so anyone can access it! Do you notice how you enter a website's name and the website opens up? Similarly, when we try to run our application through NGROK, it provides us with a URL that we can use to access the application from any device, right? Can you tell me what actually happens when we run the application from NGROK and how it opens up from any device using the URL?	**ESR:** Varied **ESR:** Varied! **ESR:** Yes **ESR:** The application is running on our device and people are using a tunnel through the URL to access the application on our device!

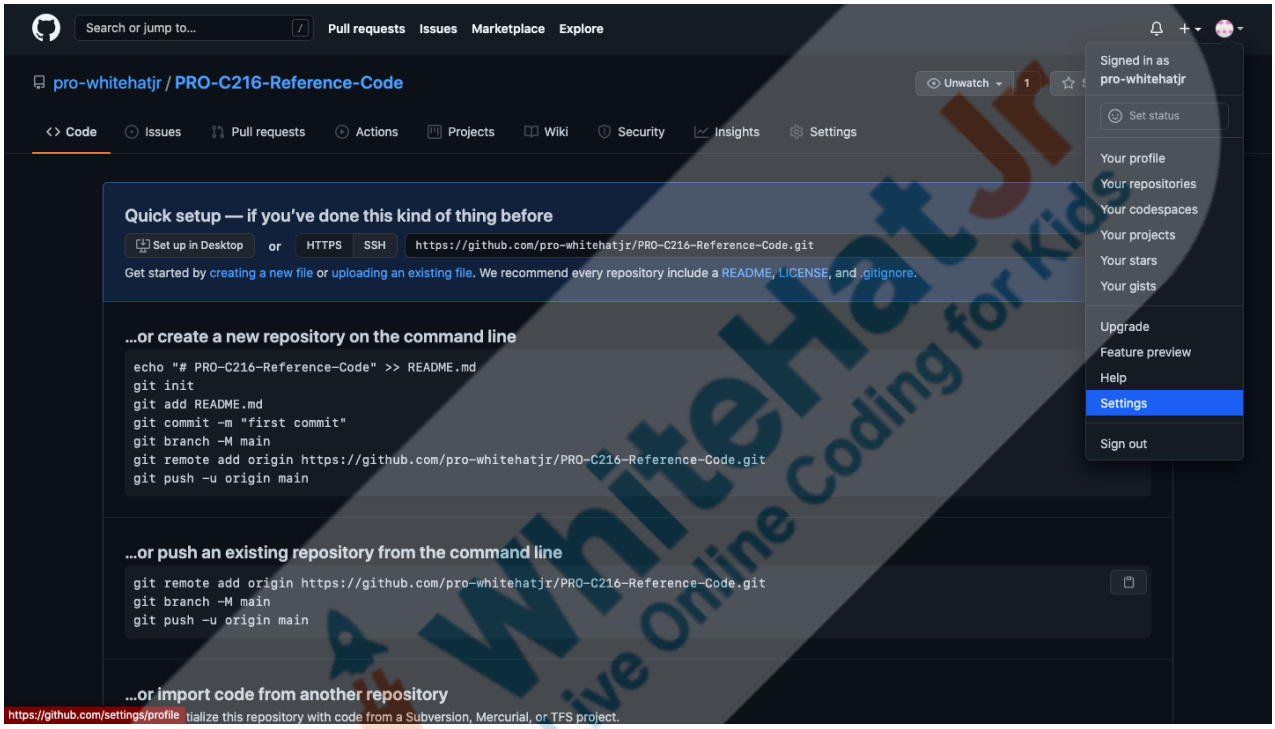
That's awesome! Now, we often close our devices, or are away from our devices. We can't keep the application running on our device forever right? Also, NGROK only gives us a limit of 2 hours in which someone can access the website. What do you think a more permanent solution for this would be? A better solution for this would be to run our application on a remote device (or we can call it a remote server) which we don't have to monitor. This is a specialised field in Software Engineering, known as **DevOps,** which means **Development and Operations**. Today, we are going to host our video chat application on a remote device, or a remote server. We will learn about how DevOps actually work, and you will be deploying your very first application! Are you excited? Let's get started then!	**ESR:** Varied! **ESR:** Yes!
Let's understand a little bit about how DevOps work in big companies and for big softwares! There are situations where, at times, multiple developers are working on the same project/application at the same time! Now we know that to collaborate and maintain a single codebase, github can be used. Github can also be used to clone a remote repository into our devices. Similarly, let's say we have a machine allocated to us that is	**ESR:** Varied

at a different place! Can we clone our code into that machine too, if we get access to it?	**ESR:** Yes!
And what after we clone it? Can we run the application on that machine once we have it there?	**ESR:** Yes!
That's what DevOps is all about!	
How it works is that big softwares contains a single repository, to which multiple developers contribute! Now, as soon as the code updates on these repositories, a devops engineer pulls the latest update into the remote device on which they are running the application, and then they run the application again.	
This way, the application keeps getting updated.	
Now, it looks like tedious work to pull all the updates and latest code into a remote machine and run the application again and again! What devops engineers do in these cases is that they automate the process!	
This means that as soon as the code gets updated into the Github Repository, it also gets updated on the remote server, and the application restarts itself!	
Cool, isn't it?	**ESR:** Yes!
Okay! Now we understand the process, and a part of it we will understand as we implement it. The first thing that we should do is that we should create a master github repository, on which we will make our changes from now onwards! *Teacher guides the student in creating a new Github repository. This is a student only activity in which the teacher is guiding. The teacher may or may not do this, as*	

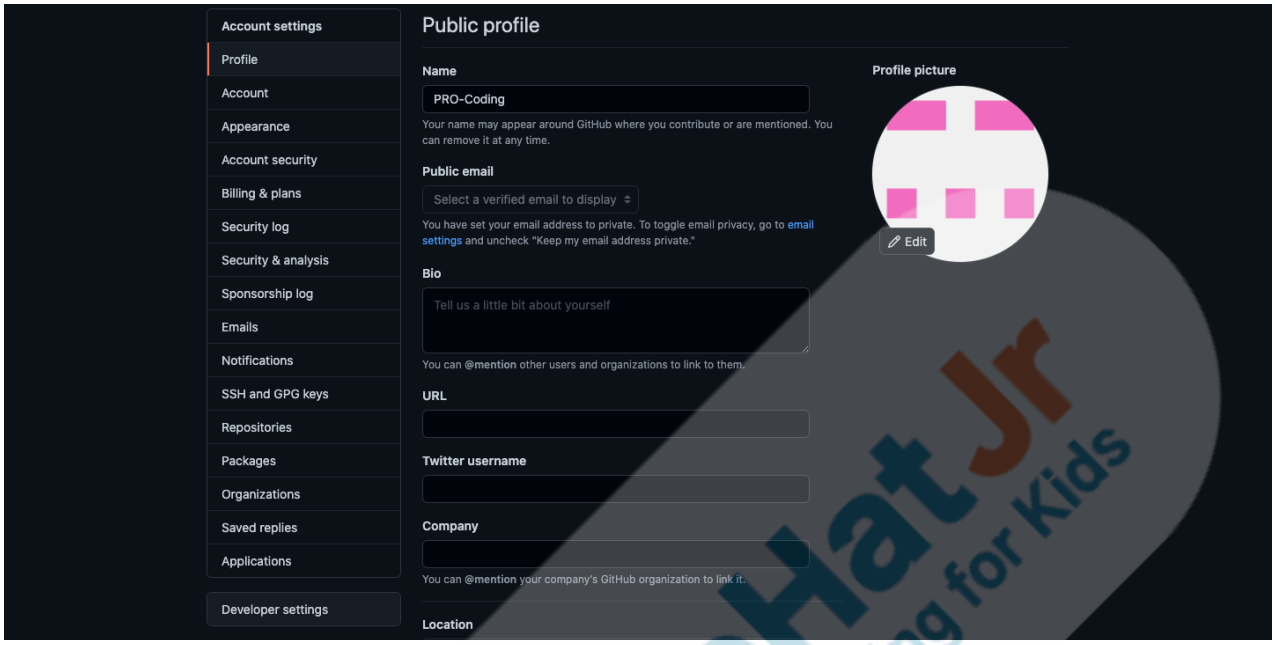
per their wish. *Note - The previous class code is available in [Teacher Activity 1]*	*Student opens [Student Activity 1]*
Here, we have the code from the last class! Let's clone it and then unzip the files from it! *Student clones the code from [Student Activity 1] and unzips the code.*	
Now let's create a new github repository! *Teacher guides the student to create a new Github Repository for the video chat app*	*Student creates a new Github Repository for Video Chat App*
	
Alright, now we need to push our code into this repository!	

*Github has changed the rules of pushing anything on github recently on **13th August 2021**. We need a personal token now to do so, and the steps mentioned below.*

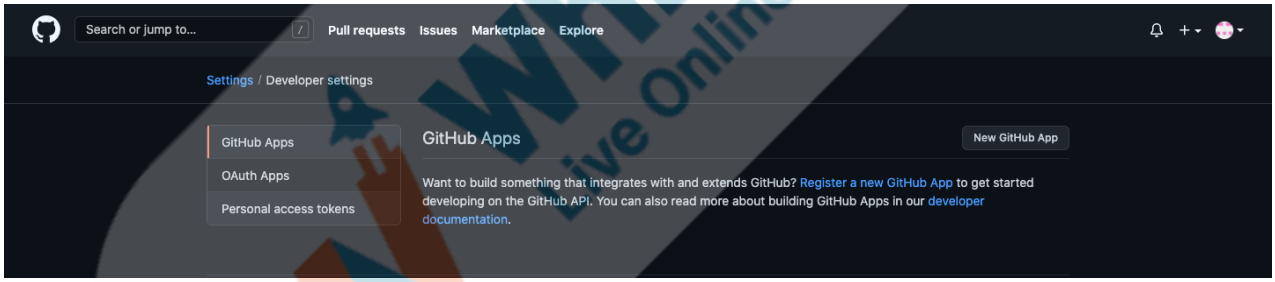
To create a personal token, we will first need to go to the settings -



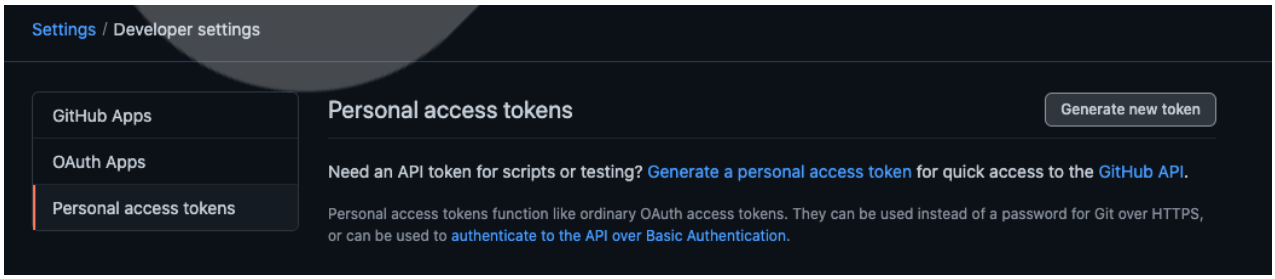
In Settings, go to Developer Settings

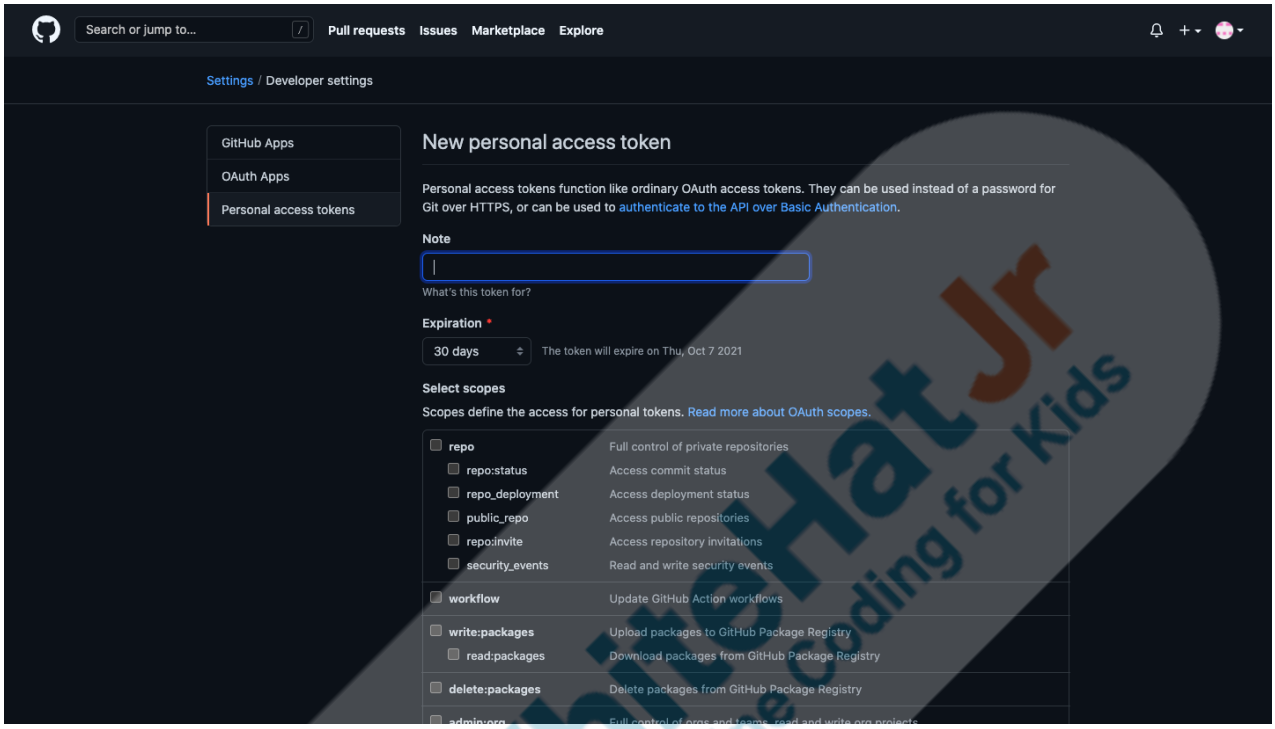


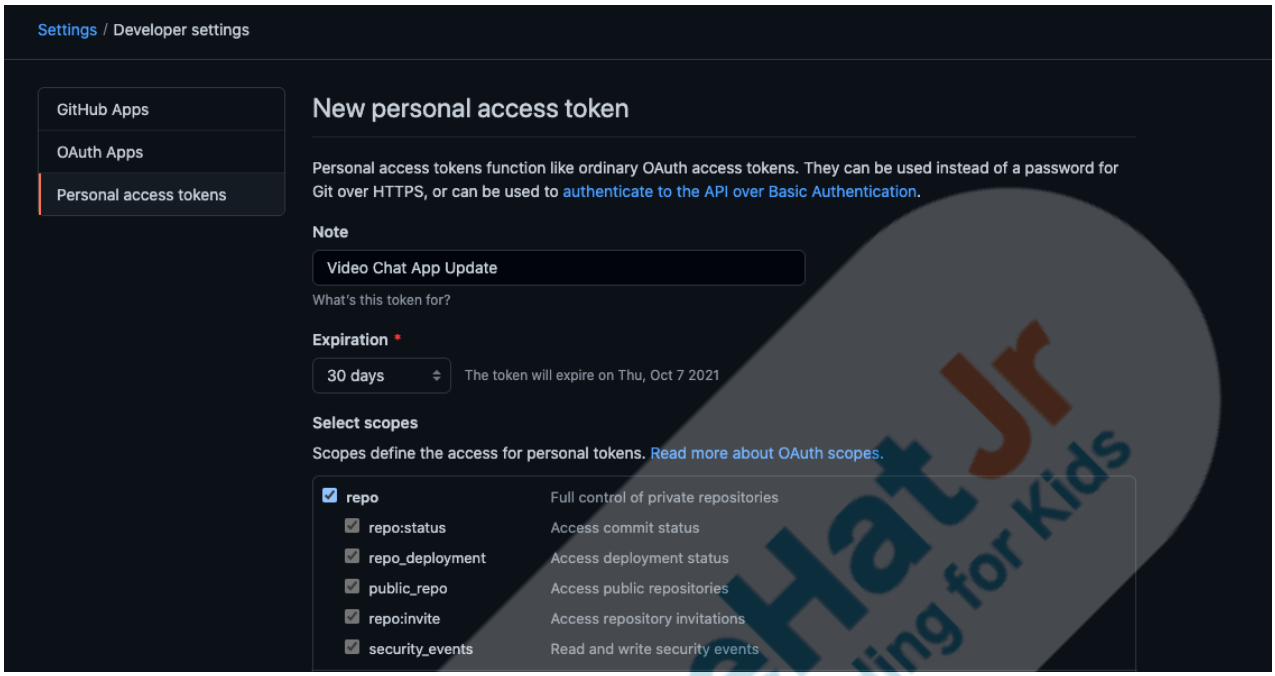
Here, select the 3rd option, which is for ***Personal Access Token*** -



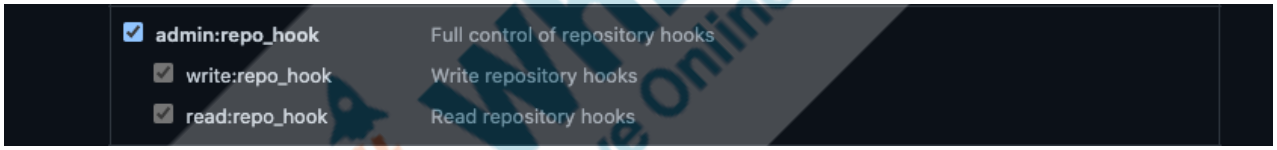
Next, click on Generate Access Token -



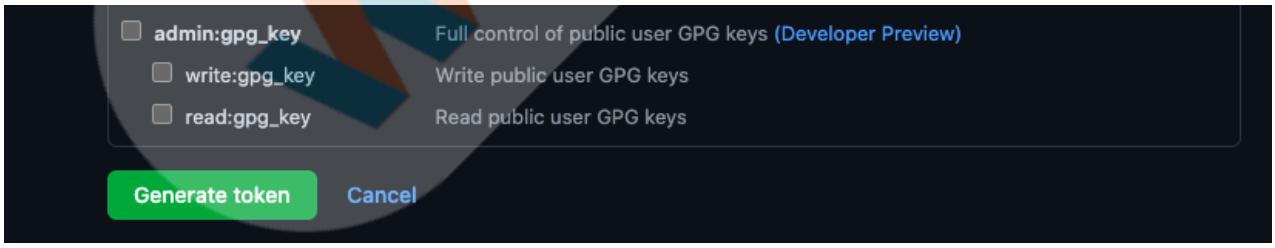
You will see the following screen -	
	
Type anything in the **Note** section as per you like. It could be to update the video chat app! Also, select the ***repo*** and ***admin:repo_hook*** from the checkbox options!	



And



Finally, click on the ***generate token*** button and Voila, the token is generated!



Copy the token that you see here! It will only be displayed once so be careful with it or the entire list of steps above would need to be repeated again.

Better yet, open a new tab and re-open the github repository that you just created!

Next, open a ***cmd/terminal*** window and navigate to the project that you just unzipped.

```
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/pro-whitehatjr/PRO-C216-Reference-Code.git
git push -u origin main
```

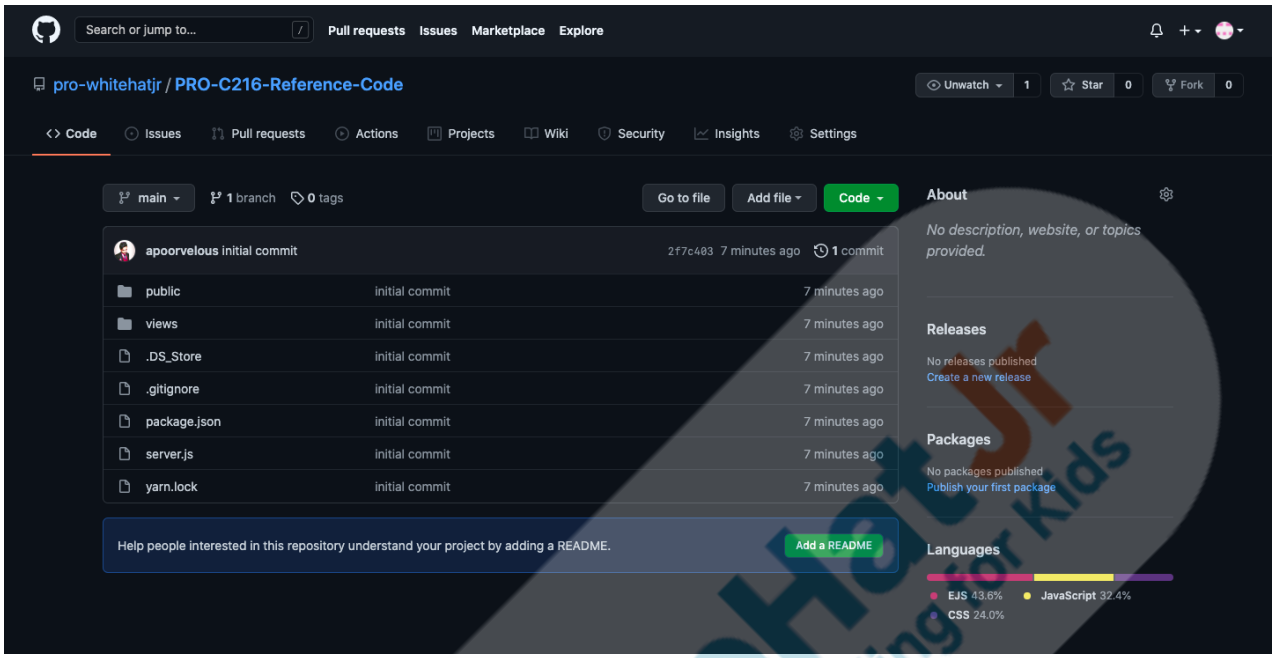
On running the last command, which is ***git push -u origin main***, you ***may*** encounter an error!

Run the following commands -

git config --global user.name "Your Username"
git config --global user.password "Personal Access Token"

Here, do note that in the first command, ***"Your Username"*** would be the student's (or teacher's if the teacher is doing it) username and in the second command, ***"Personal Access Token"*** would be the personal access token that you just generated!

Once this is done, run the ***git push -u origin main*** command again, and you will see the code pushed into your repository!



Awesome! Now, our repository is all ready! We can now use it to deploy our code into our remote device (or our remote server!)

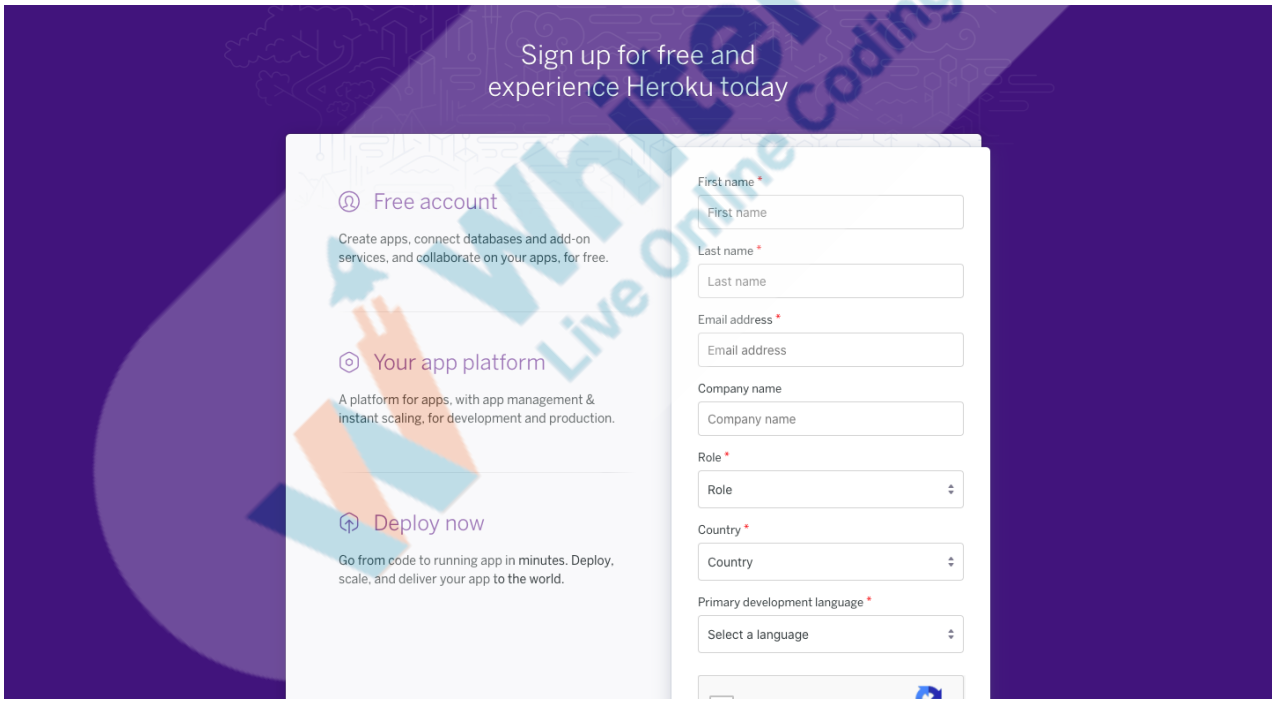
Teacher Stops Screen Share

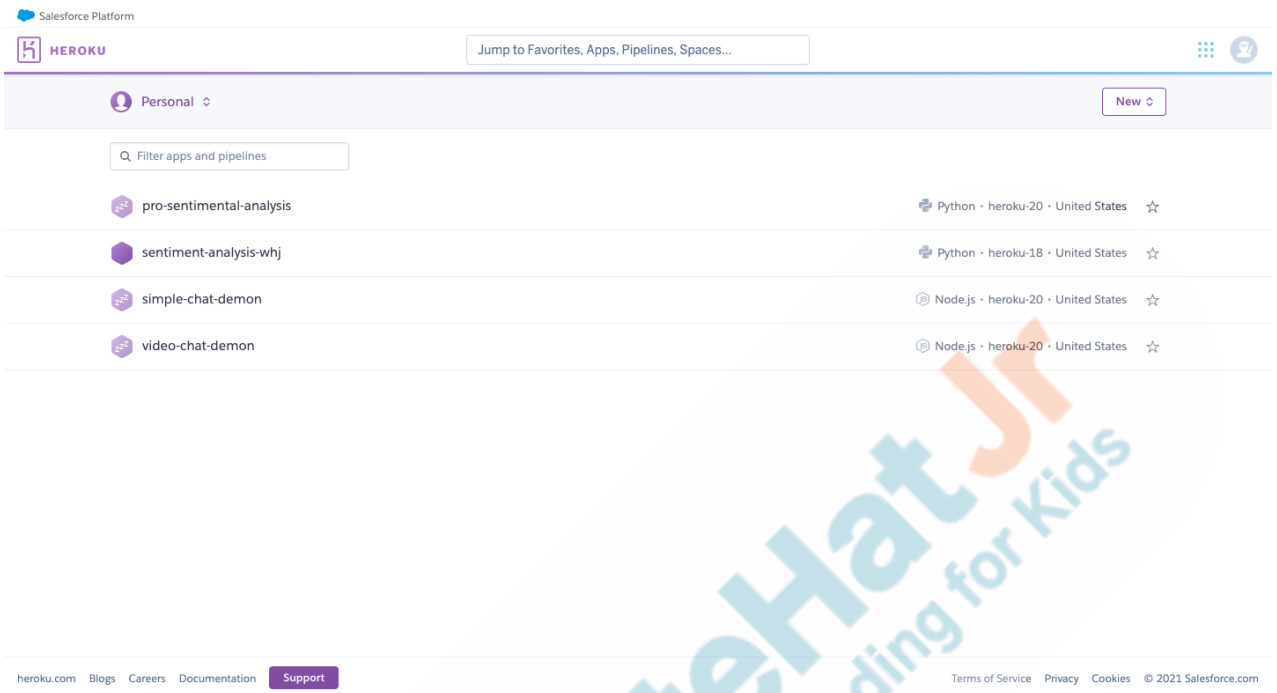
STUDENT-LED ACTIVITY - 15 mins

- **Ask the student to press the ESC key to come back to the panel.**
- **Guide the student to start Screen Share.**
- **The teacher gets into Full Screen.**

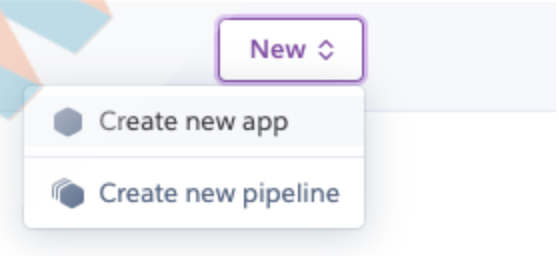
ACTIVITY

- **Introduction to Heroku!**
- **Deploying your first Web Application!**

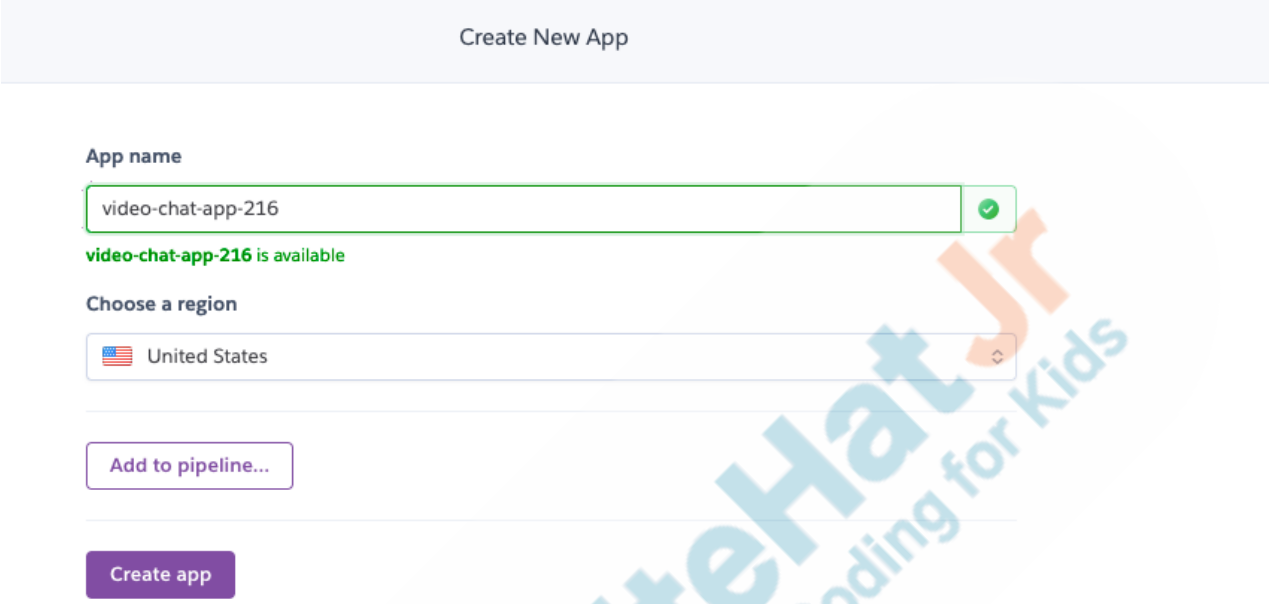
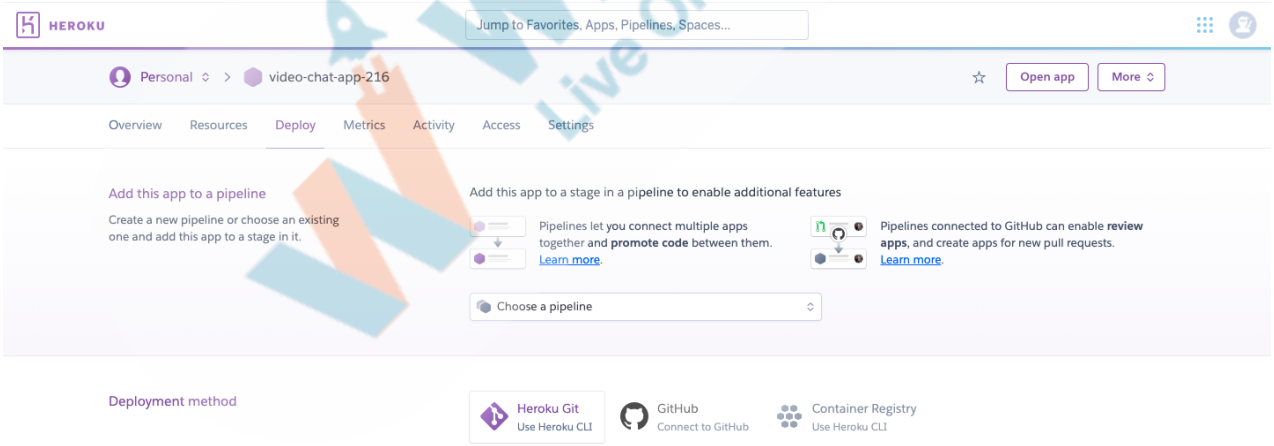
Teacher Action	Student Action
You must be wondering, where will we find this remote device? Well, there is an excellent platform just for this, known as ***Heroku***. It is a platform that offers remote devices to host/deploy applications, and these remote devices are known as ***dyno***. Let's start by signing up on ***Heroku!*** *Teacher opens [Teacher Activity 2]() and guides the student to sign up*	*Student opens [Student Activity 2]() and signs up*
	
Now, let's login into Heroku and understand the dashboard. On successful login, you will see the following screen -	

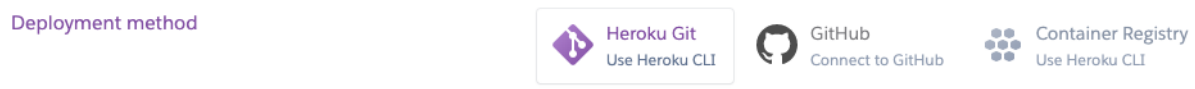


Here, for you, there will not be any projects as demonstrated in the screenshot, since your account is new. Now, we can see a button on the top right, which says ***New***. Click on it, and then click on ***Create a new app*** *Teacher guides the student*	*Student follows the instructions*



Next, enter a name for your app and click on the ***Create App*** button!	

Teacher guides the student	*Student follows the instructions*
	
Once done, we will be prompted to the following screen -	
	
Here, you will notice a section that says **Deployment methods** -	



Here, the two of the most famous methods used to deploy on Heroku are -

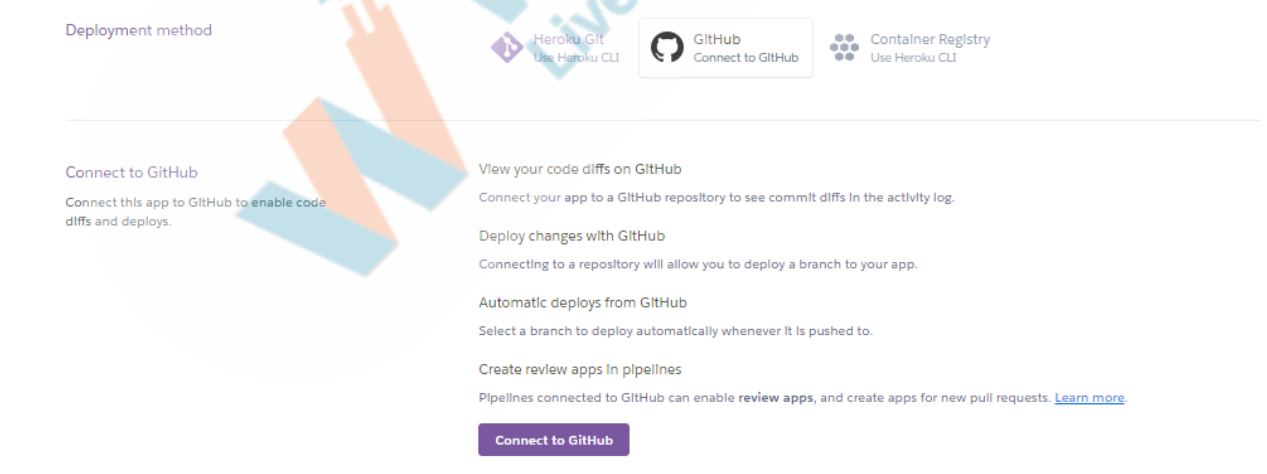
1. **Heroku Git** - Default Method
2. **Github -** Widely used Method

Heroku Git method is a rather complicated one, where you have to download and install a tool called ***heroku*** which works on ***terminal/cmd***.

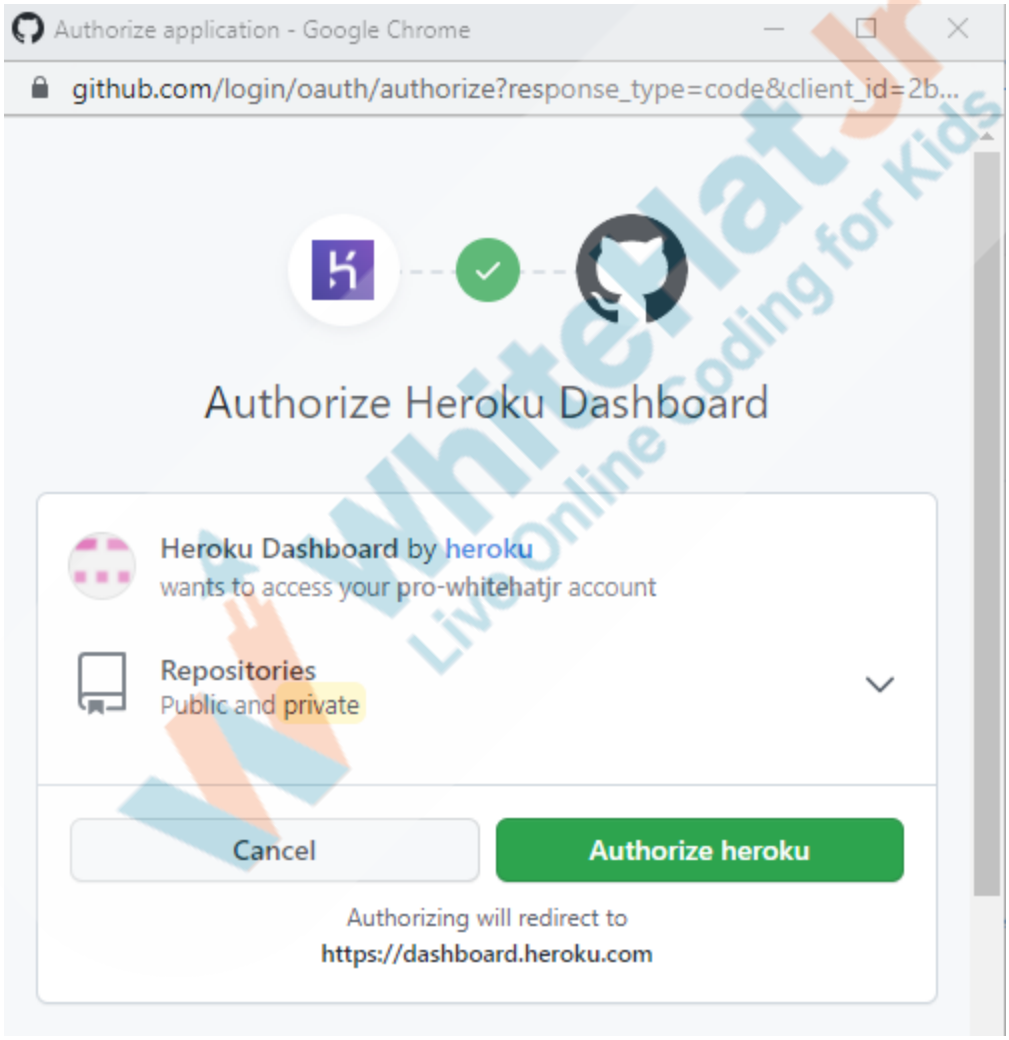
In this method, Heroku provides you with a git repository in which you have to push your code for it to work.

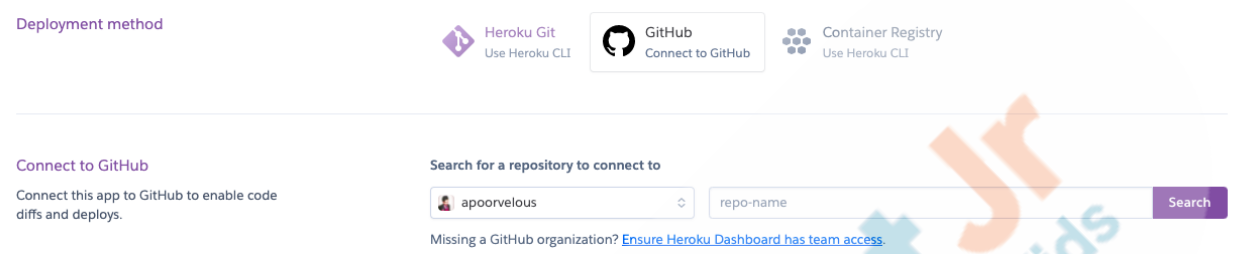
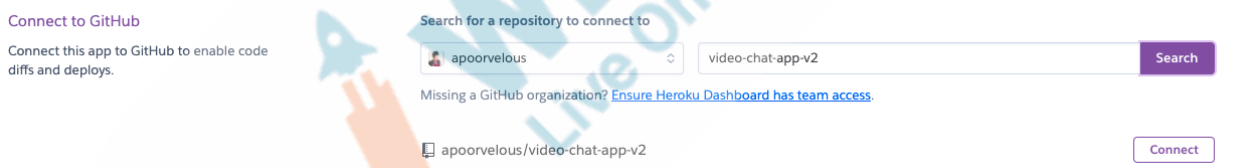
For us, we already have created a Github repository, so why not use it directly? Right.

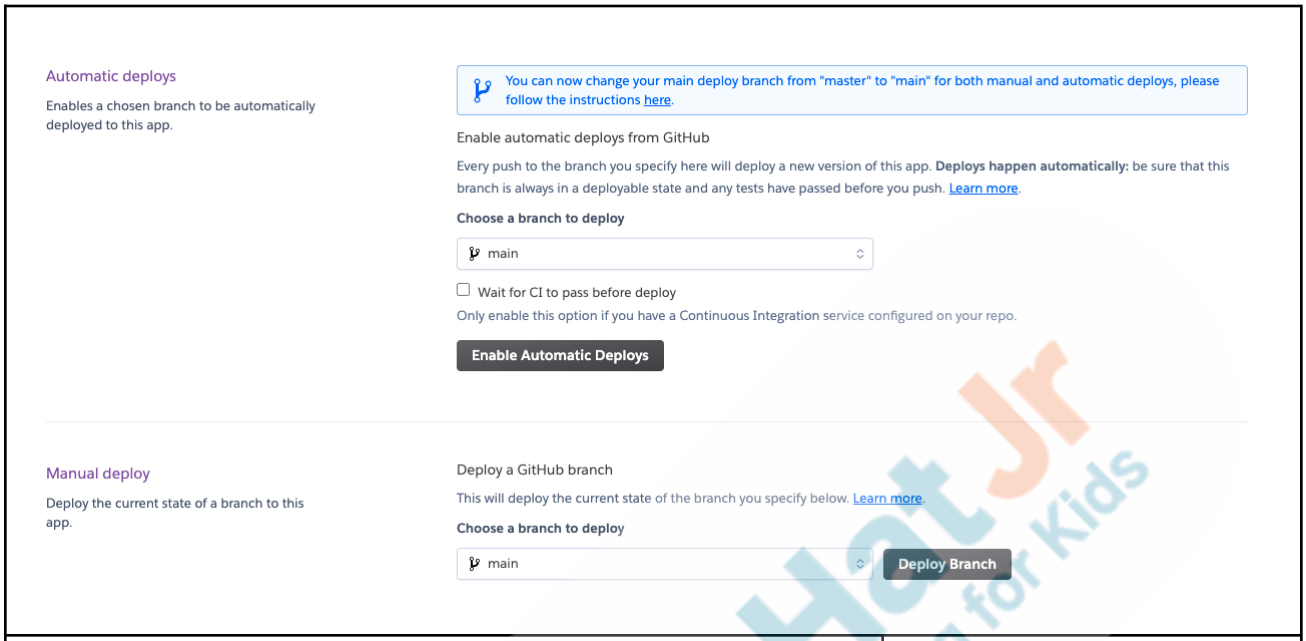
Let's select the **Github** method from this prompt and see the options we get -



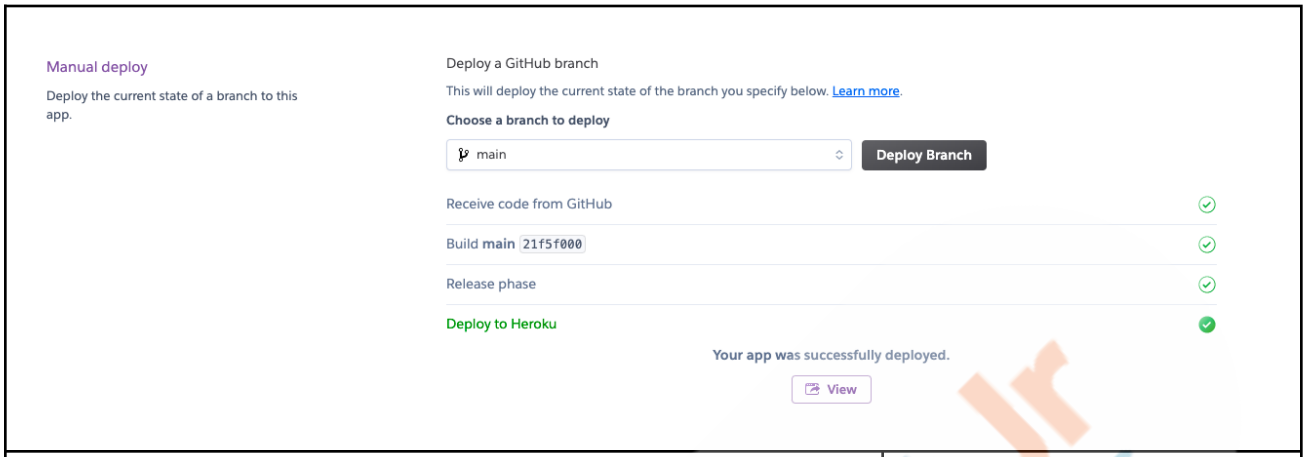
Here, we see an option below which says -

Connect to Github Let's click on it and connect our Heroku with Github *Teacher guides the student*	*Student follows the instructions*
	
A dialog box opens up, where you can select the Github account which you want to authorise with Heroku, and click	

on **Authorize heroku** *Teacher guides the student*	*Student follows the instructions*
	
Once done, we can see that Heroku asks us for a Repo name that we want to connect with our app. Let's enter the name of the repository that we just created and click on ***search*** - *Teacher guides the student*	*Student follows the instructions*
	
Once you search, you will see that Heroku has located the repository, and we can click on ***connect*** to connect that particular repo to the Heroku app we created. Let's do that - *Teacher guides the student*	*Student follows the instructions*

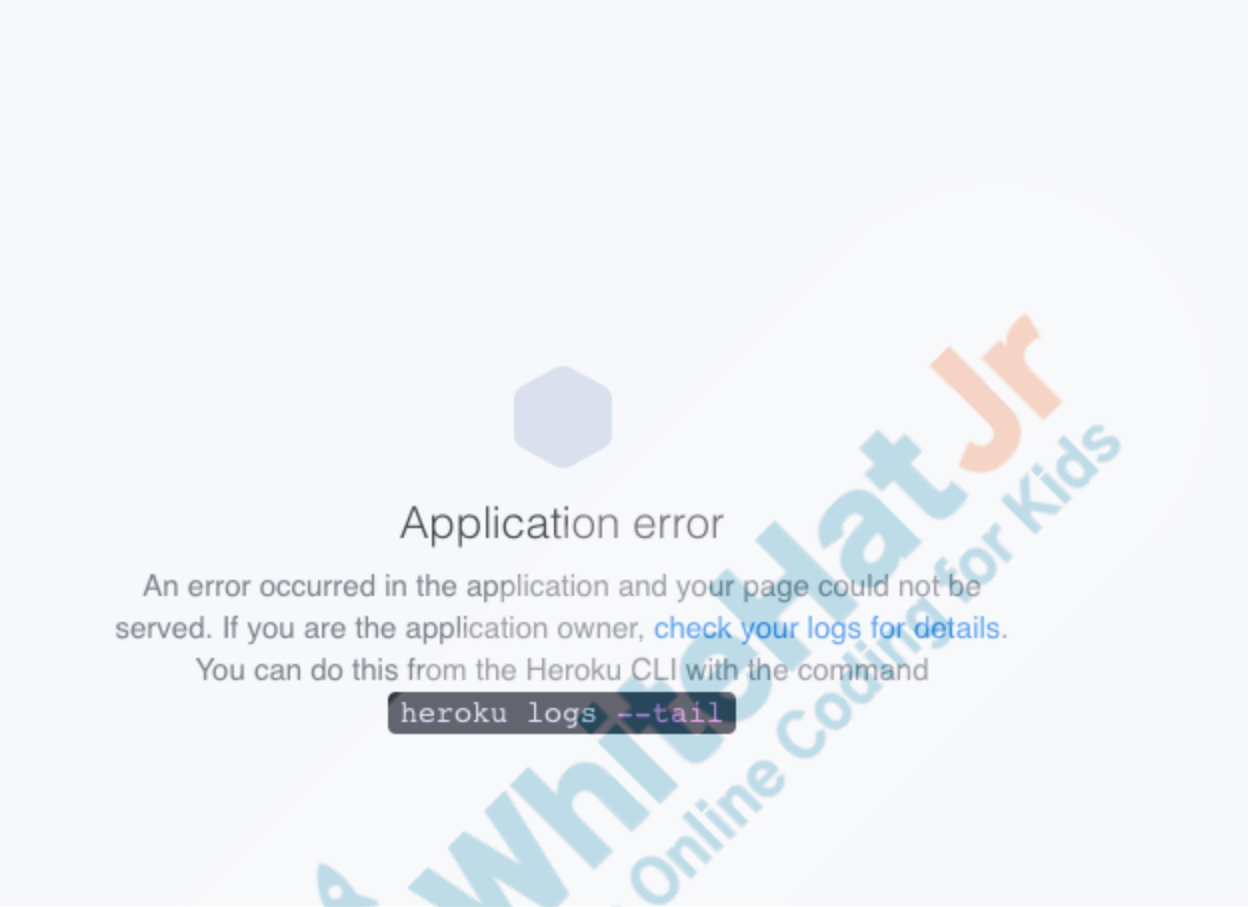


Now that the app is connected, we can see that the options are different now. On scrolling down to the bottom, we can see a ***Deploy Branch*** button in the last section which says ***Manual deploy***. We can select the branch that we want to deploy. For us, it will be the default ***main*** branch. Let's click on it so our ***app*** gets deployed - *Teacher guides the student*	*Student follows the instructions*



Once deployed, we can click on the ***View*** button to see if it is deployed or not.

Let's do that -



Oh no! It shows an Application Error! We can check the logs by clicking on the ***check you logs for details*** link.	

```
2021-09-07T12:08:36.514312+00:00 app[web.1]: > video-chat-app@1.0.0 start /app
2021-09-07T12:08:36.514312+00:00 app[web.1]: > node server.js
2021-09-07T12:08:36.514312+00:00 app[web.1]:
2021-09-07T12:09:33.558035+00:00 heroku[web.1]: Error R10 (Boot timeout) -> Web process failed to bind
 to $PORT within 60 seconds of launch
2021-09-07T12:09:33.606748+00:00 heroku[web.1]: Stopping process with SIGKILL
2021-09-07T12:09:33.688757+00:00 heroku[web.1]: Process exited with status 137
2021-09-07T12:09:34.042752+00:00 heroku[web.1]: State changed from starting to crashed
2021-09-07T12:09:34.071754+00:00 heroku[web.1]: State changed from crashed to starting
2021-09-07T12:09:36.097478+00:00 heroku[web.1]: Starting process with command `npm start`
2021-09-07T12:09:38.106544+00:00 app[web.1]:
2021-09-07T12:09:38.106556+00:00 app[web.1]: > video-chat-app@1.0.0 start /app
2021-09-07T12:09:38.106557+00:00 app[web.1]: > node server.js
2021-09-07T12:09:38.106557+00:00 app[web.1]:
2021-09-07T12:10:36.325565+00:00 heroku[web.1]: Error R10 (Boot timeout) -> Web process failed to bind
 to $PORT within 60 seconds of launch
2021-09-07T12:10:36.490875+00:00 heroku[web.1]: Stopping process with SIGKILL
2021-09-07T12:10:36.600111+00:00 heroku[web.1]: Process exited with status 137
2021-09-07T12:10:36.654706+00:00 heroku[web.1]: State changed from starting to crashed
2021-09-07T12:13:25.813781+00:00 heroku[router]: at=error code=H10 desc="App crashed" method=GET path=
"/" host=video-chat-app-216.herokuapp.com request_id=8f3e6bd2-b2c9-4f13-b852-770b2fa3f710 fwd="205.254
.164.98" dyno= connect= service= status=503 bytes= protocol=https
2021-09-07T12:13:26.400591+00:00 heroku[router]: at=error code=H10 desc="App crashed" method=GET path=
"/favicon.ico" host=video-chat-app-216.herokuapp.com request_id=1c6b54b9-ddc9-4c2b-82a1-1e968449d631 f
wd="205.254.164.98" dyno= connect= service= status=503 bytes= protocol=https
```

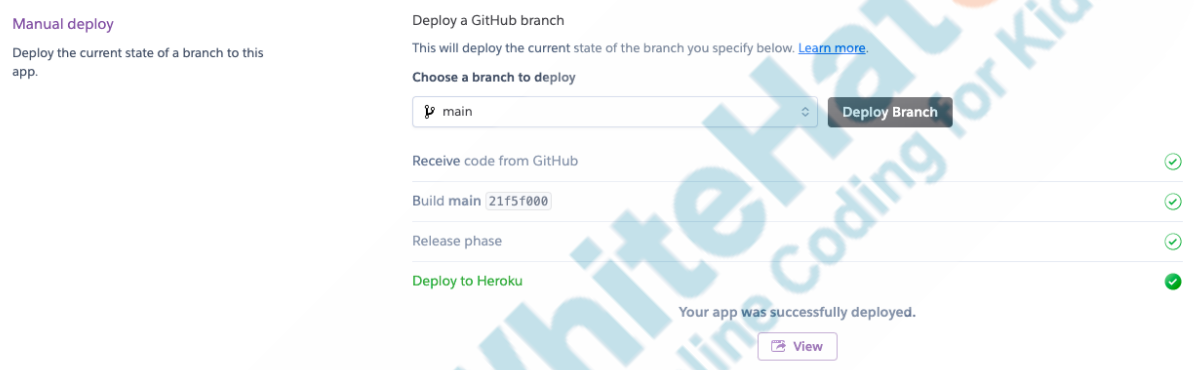
Here, one of the logs tells us that it was not able to bind with the port within 60 seconds! The reason behind this is that we run the app locally on port 3030, but Heroku uses a different port! To tackle this, let's make a small change into our ***server.js*** file -

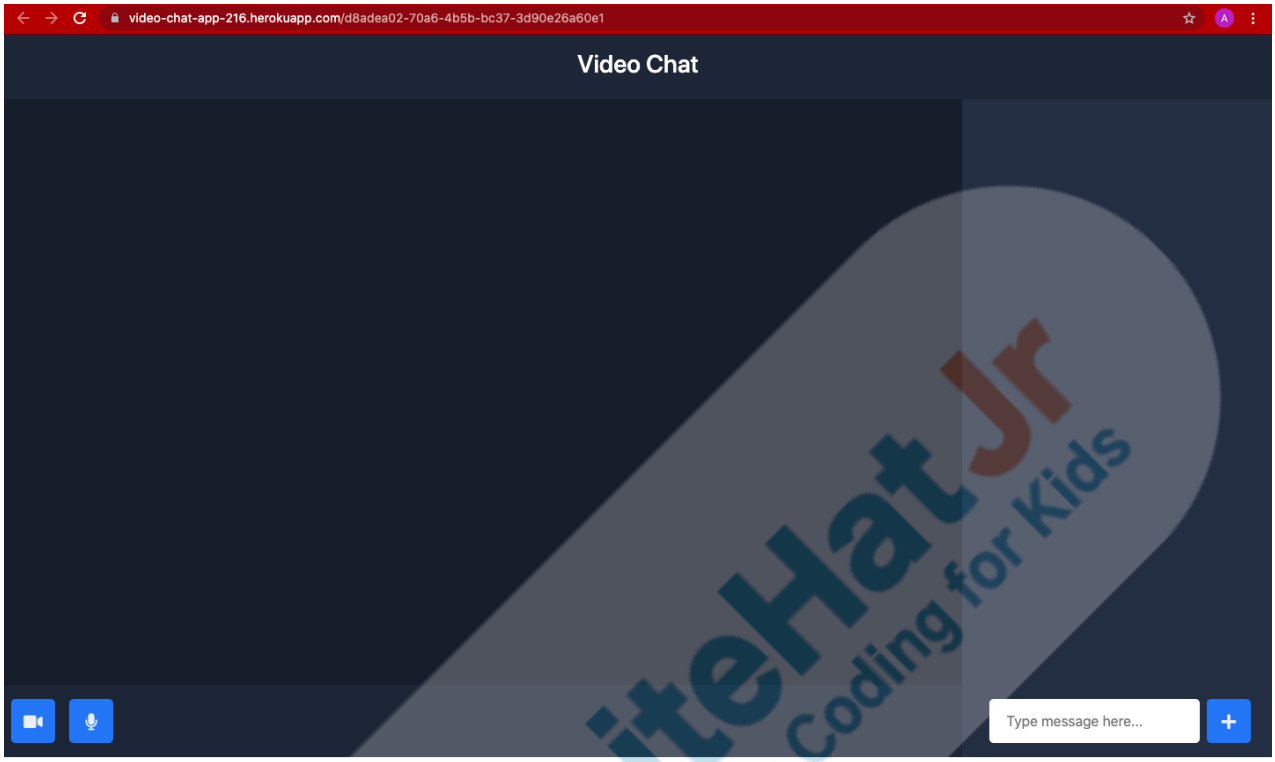
Teacher guides the student to make the change

Student makes the change

```
server.listen(process.env.PORT || 3030);
```

Instead of ***server.listen(3030)***, we would need to do ***server.listen(process.env.PORT || 3030)***. This will let the app see if there is a port server is offering. If not, it will use 3030 by default. This way, it will run both locally and on Heroku.

Now, let's update this small change on Github with the following commands - ***git add -A*** ***git commit -m "small change"*** ***git push*** *Teacher helps the student in running the command* Now that it is updated on Github, let's click on ***Deploy Branch*** again on the Heroku Dashboard page.	*Student runs the command*
	
Awesome! Let's open the app again to see if it works this time! *Teacher guides the student in opening the app*	*Student opens the app*



Awesome! Just awesome! Our app works on the browser now! Let's test it! *Teacher takes the link from the student and enters it in her browser to enter the same room. Teacher and student tries to chat*	*Student and teacher tries to chat!*

Awesome, simply Awesome isn't it? You just deployed your very first application and learnt about DevOps Engineering! Kudos to you! Now, we will complete our video chat functionality in the coming classes, and you can go on to have video chat with your friends and family on an application that you built!	
Teacher Guides Student to Stop Screen Share	
WRAP UP SESSION - 5 Mins	
Quiz time - Click on in-class quiz	
Question	**Answer**
What is cmd? A. Command Prompt B. Command Terminal	**D**

C. Command D. All of the above	
Why do we use git add -A? A. Add Files to the repository B. Delete the file from the repository C. Change files to the repository D. None of the above	
What does git commit -m "small change" mean? A. Small change and commit B. Commit C. Small Change D. None of the above	
End the quiz panel	
FEEDBACK • **Appreciate the students for their efforts in the class.** • **Ask the student to make notes for the reflection journal along with the code they wrote in today's class.**	
Teacher Action	**Student Action**
You get Hats off for your excellent work! In the next class, we will be getting into WebRTC!	*Make sure you have given at least 2 Hats Off during the class for:* Creatively Solved Activities +10

	Great Question +10 Strong Concentration +10
Project Discussion In this class, we learnt how we can deploy an app on Heroku. In this project, you'll be deploying the forum chat app to Heroku. John's app is up and running! He has been trying to deploy it on Heroku so that everyone can start using it, but he needs help from an expert. Help him deploy the app on Heroku.	
Teacher Clicks End Class	
ADDITIONAL ACTIVITIES	
Additional Activities *Encourage the student to write reflection notes in their reflection journal using markdown.* Use these as guiding questions: ● What happened today? ○ Describe what happened. ○ The code I wrote. ● How did I feel after the class?	*The student uses the markdown editor to write her/his reflections in the reflection journal.*

• What have I learned about programming and developing games? • What aspects of the class helped me? What did I find difficult?	

ACTIVITY LINKS		
Activity Name	**Description**	**Link**
Teacher Activity 1	Previous Class Code	https://github.com/pro-whitehatjr/PRO-C217-ReferenceCode
Teacher Activity 2	Heroku Signup	https://signup.heroku.com/
Teacher Activity 3	Reference Code	https://github.com/pro-whitehatjr/PRO-C218-ReferenceCode
Student Activity 1	Previous Class Code	https://github.com/pro-whitehatjr/PRO-C217-ReferenceCode
Student Activity 2	Heroku Signup	https://signup.heroku.com/