COMP 7940 Cloud Computing

2023/24 S2 Lab5: Deploying the Chatbot to Public Cloud

ROLE	NAME	EMAIL
Instructor	Dr. Qichen WANG	qcwang@hkbu.edu.hk
Teaching Assistant	Mr. Yu XU	csyuxu@comp.hkbu.edu.hk
Teaching Assistant	Mr. Hongduan TIAN	cshdtian@comp.hkbu.edu.hk

Intended Learning Outcomes

Throughout this lab session, you are expected to:

- Master the skill of using a public cloud.

---

1. Registration

In this lab session, we are going to use fly.io as a cloud service platform. First of all, an important thing that you are required to do is apply a free account from fly.io. If you already have owned a Github account, I recommend you to directly sign up with your Github account.

2. Installing CLI of fly.io

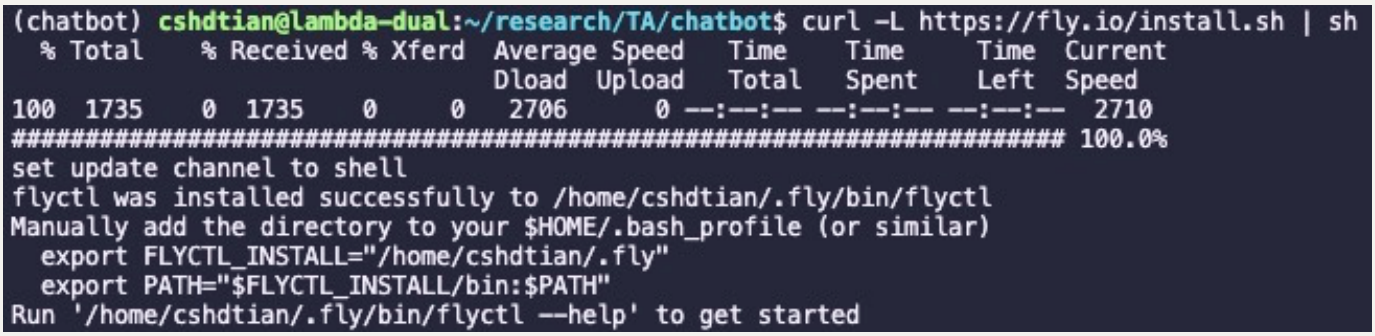
Throughout the entire deployment in this lab session, all practical exercises will be conducted on fly.io platform. Thus, the first step is installing the CLI based on the operation system of your PC with the following command.

For MacOS & Linux:

```
curl -L https://fly.io/install.sh | sh
```

For Windows:

```
pwsh -Command "iwr https://fly.io/install.ps1 -useb | iex"
```



NOTE: For those whose use Linux / MacOS system, you have to manually add the path of DETA to `~/.bashrc` / `~/.bash_profile` file via:

```
vim ~/.bash_profile
# Then add the following two command in this file:
#   export FLYCTL_INSTALL="/home/cshdtian/.fly"
#   export PATH="$FLYCTL_INSTALL/bin:$PATH"
# "Esc + :wq"  -->  quit editting and save the modification
source ~/.bash_profile
```

After installing, in order to verify that you have successfully installed the flyctl, type `fly --help`, you can see:

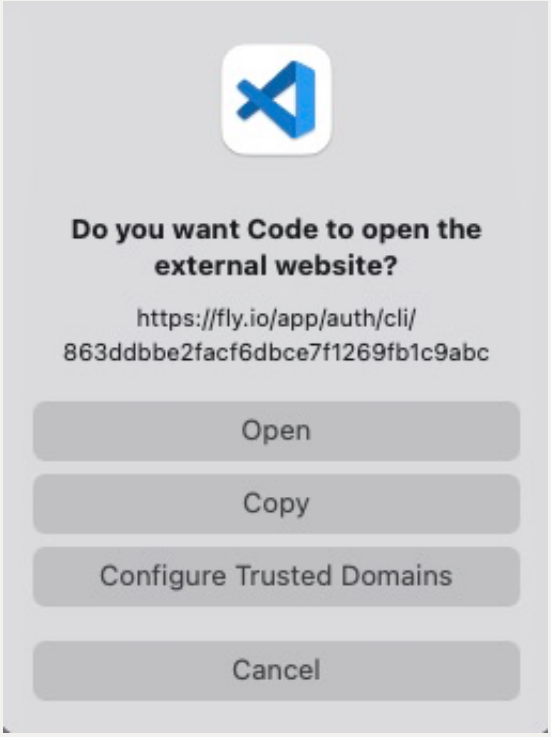
```
(chatbot) cshdtian@lambda-dual:~/research/TA/chatbot$ /home/cshdtian/.fly/bin/flyctl --help
This is flyctl, the Fly.io command line interface.

Usage:
  flyctl [flags]
  flyctl [command]

Deploying apps & machines
  apps         Manage apps
  deploy       Deploy Fly applications
  launch       Create and configure a new app from source code or a Docker image
  machine      Manage Fly Machines.
  status       Show app status

Configuration & scaling
  certs        Manage certificates
  config       Manage an app's configuration
  image        Manage app image
  ips          Manage IP addresses for apps
  scale        Scale app resources
  secrets      Manage application secrets with the set and unset commands.
  volumes      Manage Fly Volumes.

Monitoring & managing things
  checks       Manage health checks
  console      Run a console in a new or existing machine
  dashboard    Open web browser on Fly Web UI for this app
  dig          Make DNS requests against Fly.io's internal DNS server
  logs         View app logs
  ping         Test connectivity with ICMP ping messages
  proxy        Proxies connections to a Fly Machine.
  releases     List app releases
  services     Show the application's services
  sftp         Get or put files from a remote VM.
  ssh          Use SSH to log into or run commands on Machines
  wireguard    Commands that manage WireGuard peer connections

Databases & extensions
  consul       Enable and manage Consul clusters
  extensions   Extensions are additional functionality that can be added to your Fly apps
  litefs-cloud LiteFS Cloud management commands
  planetscale  Provision and manage PlanetScale MySQL databases
  postgres     Manage Postgres clusters.
  redis        Launch and manage Redis databases managed by Upstash.com
  storage      Provision and manage Tigris object storage buckets

Access control
  auth         Manage authentication
  orgs         Commands for managing Fly organizations
  tokens       Manage Fly.io API tokens

Help & troubleshooting
  docs         View Fly documentation
  doctor       The DOCTOR command allows you to debug your Fly environment
  platform     Fly platform information

Additional Commands:
  agent        Commands that manage the Fly agent, a background process that manages flyctl wireguard connection
s
  completion   Generate the autocompletion script for the specified shell
  help         Help about any command
  jobs         Show jobs at Fly.io
  settings     Manage flyctl settings
  version      Show version information for the flyctl command

Flags:
  -t, --access-token string   Fly API Access Token
      --debug                 Print additional logs and traces
  -h, --help                  help for flyctl
      --verbose               Verbose output

Use "flyctl [command] --help" for more information about a command.
```

3. Login fly.io via terminals

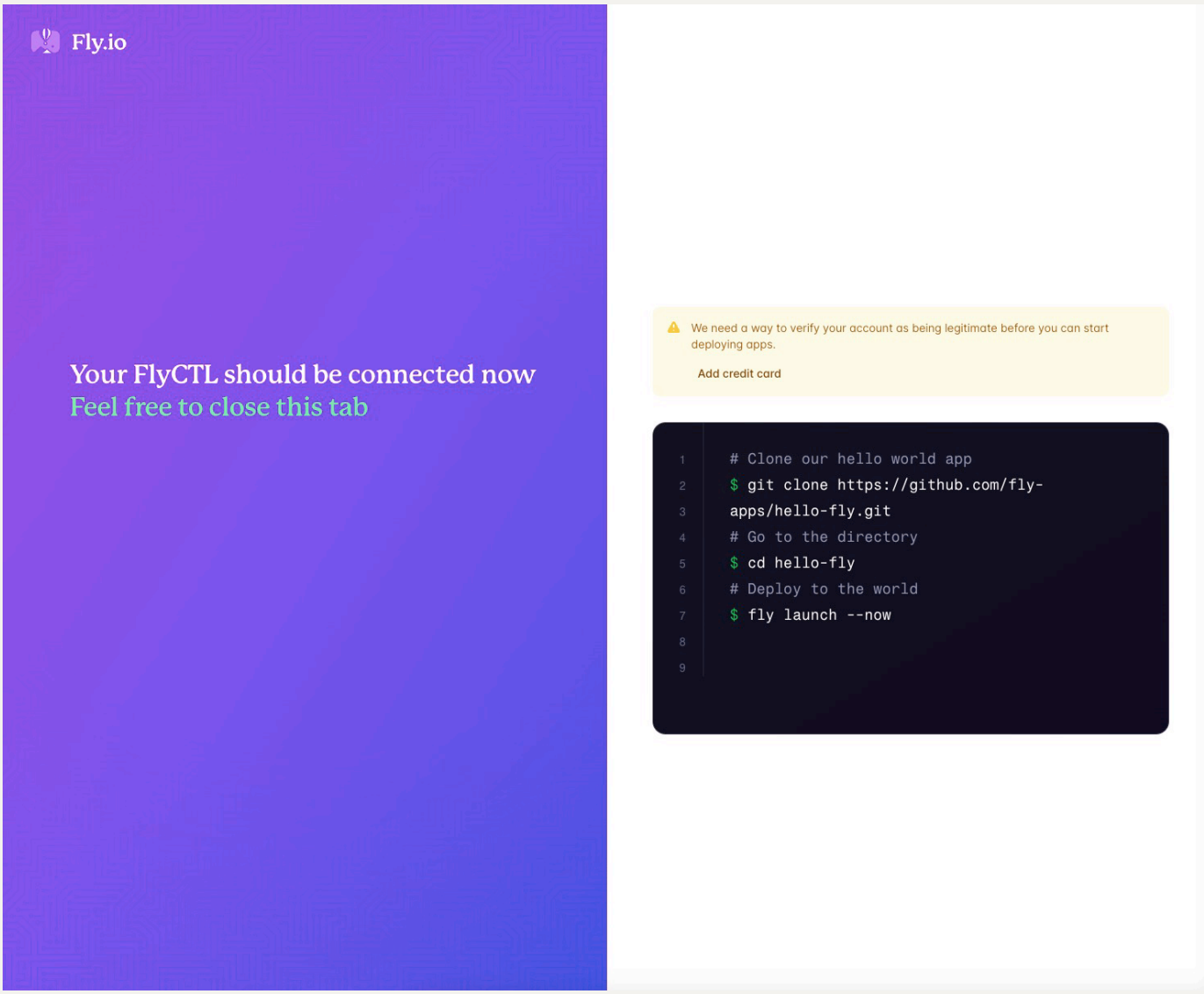
The way of loginning fly.io is using the command `fly auth login`.

```
(chatbot) cshdtian@lambda-dual:~/research/TA/chatbot$ /home/cshdtian/.fly/bin/flyctl auth signup
Opening https://fly.io/app/auth/cli/a8c043d1a77d770cbda4cb03e3fe3bb7 ...

Waiting for session... Done
successfully logged in as hongduan.tian@gmail.com
2024/02/17 11:38:59 traces export: context deadline exceeded: rpc error: code = Unavailable desc = connection er
ror: desc = "transport: Error while dialing: failed to do connect handshake, status code: 403 Forbidden"
```

Then, a window will pop out:



Click `Open` and sign up in your browser.



Then, close the browser.

NOTE: You have to add your credit card here so that you can further launch your app in the next step. **Please carefully read the instruction email sent by Dr. Wang before you add the credit card.**

4. Launch your own chatbot

(1) Enter the folder of Lab 4, where you have finished a chatbot equipped with ChatGPT. The folder ought to contain at least 2 files: `config.ini` and your python script for chatbot.

(2) In the last lab session, a `requirements.txt` file is editted. Here, you can also generate the corresponding `requirements.txt` file regarding your experimental environment with the command:

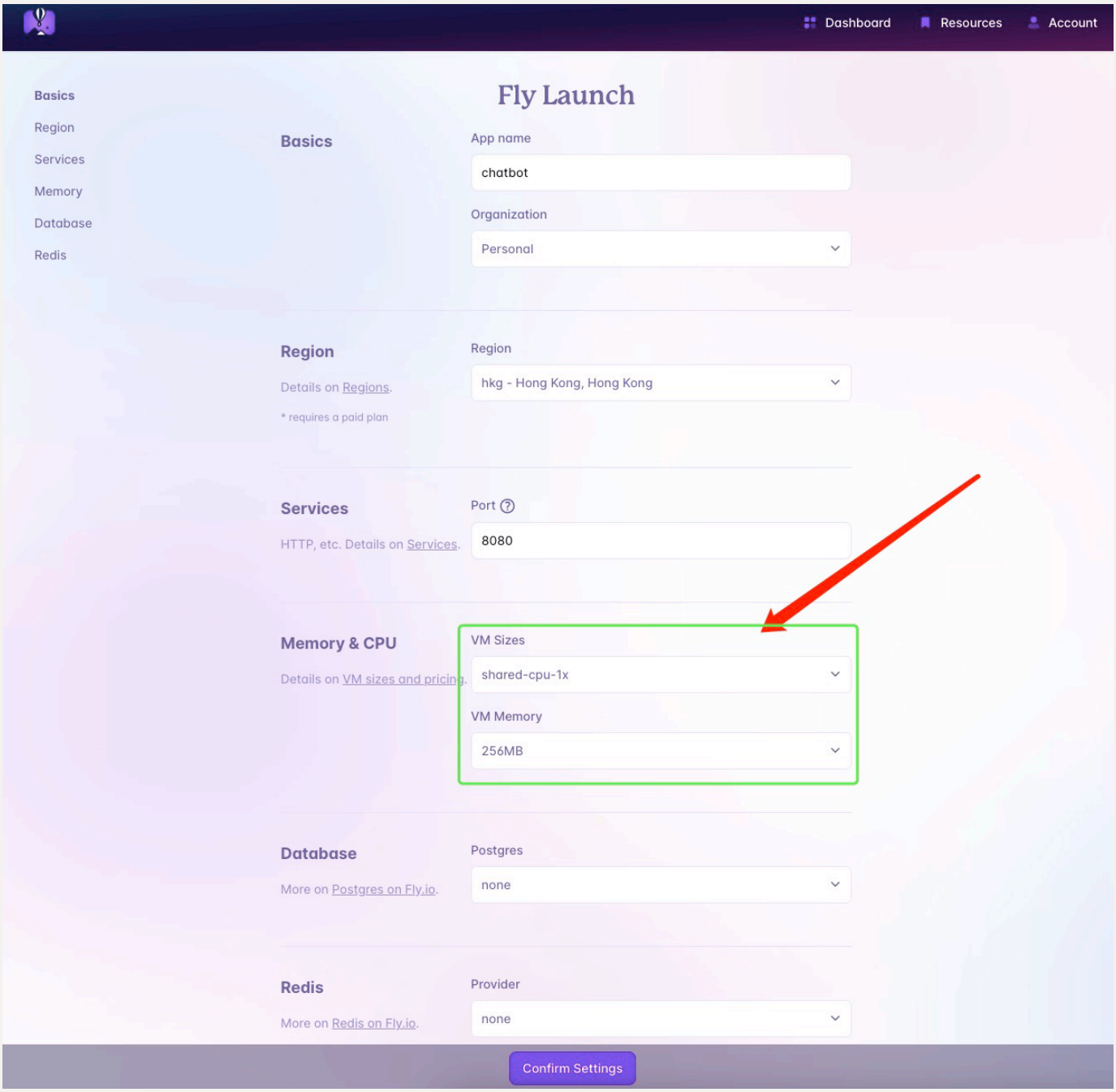
```
pip freeze > requirements.txt
```

```
(chatbot) tianhongduan@192 chatbot_comp7940 % pip freeze > requirements.txt
(chatbot) tianhongduan@192 chatbot_comp7940 % ls
app.py                  config.ini              requirements.txt
```

Then, run `fly launch --no-deploy` to launch a session of your app.

```
(chatbot) tianhongduan@192 chatbot_comp7940 % fly launch --no-deploy
Scanning source code
Detected a Python app
Using the following build configuration:
        Builder: paketobuildpacks/builder:base
Creating app in /Users/tianhongduan/上课:教学任务/TA-COMP7940/Lab5/chatbot_comp7940
We're about to launch your Python app on Fly.io. Here's what you're getting:

Organization: Hongduan Tian          (fly launch defaults to the personal org)
Name:         chatbot-comp7940       (derived from your directory name)
Region:       Hong Kong, Hong Kong   (this is the fastest region for you)
App Machines: shared-cpu-1x, 1GB RAM (most apps need about 1GB of RAM)
Postgres:     <none>                 (not requested)
Redis:        <none>                 (not requested)

? Do you want to tweak these settings before proceeding? Yes
Opening https://fly.io/cli/launch/d60b9ec1da44db4d8cb0597dea1d74dd ...
```

Then, you are able to configure your app from the browser. To make sure that there isn't any extra charges in the future, please set the cpu and memory as following.

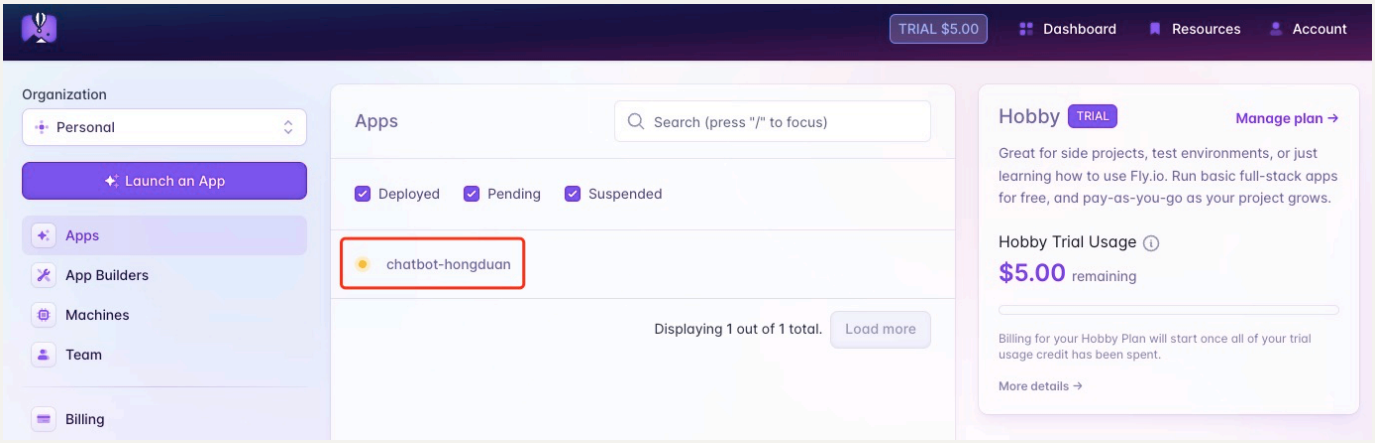


Then, you can successfully configure your app and there will be a `Procfile` and a `fly.toml` file in your source project directory.

```
(chatbot) tianhongduan@192 chatbot_comp7940 % fly launch --no-deploy
Scanning source code
Detected a Python app
Using the following build configuration:
        Builder: paketobuildpacks/builder:base
Creating app in /Users/tianhongduan/上课:教学任务/TA-COMP7940/Lab5/chatbot_comp7940
We're about to launch your Python app on Fly.io. Here's what you're getting:

Organization: Hongduan Tian          (fly launch defaults to the personal org)
Name:         chatbot-comp7940       (derived from your directory name)
Region:       Hong Kong, Hong Kong   (this is the fastest region for you)
App Machines: shared-cpu-1x, 1GB RAM (most apps need about 1GB of RAM)
Postgres:     <none>                 (not requested)
Redis:        <none>                 (not requested)

? Do you want to tweak these settings before proceeding? Yes
Opening https://fly.io/cli/launch/d60b9ec1da44db4d8cb0597dea1d74dd ...

Waiting for launch data... Done
Created app 'chatbot-hongduan' in organization 'personal'
Admin URL: https://fly.io/apps/chatbot-hongduan
Hostname: chatbot-hongduan.fly.dev
Wrote config file fly.toml
Validating /Users/tianhongduan/上课:教学任务/TA-COMP7940/Lab5/chatbot_comp7940/fly.toml
✓ Configuration is valid
We have generated a simple Procfile for you. Modify it to fit your needs and run "fly deploy" to d
eploy your application.
```

```
(chatbot) tianhongduan@192 chatbot_comp7940 % ls
Procfile                config.ini              requirements.txt
app.py                  fly.toml
```

Meanwhile, you can see a project in the `Dashboard` in your fly.io account.



(3) Modify the `fly.toml` file as following. The modification mainly focuses on the https service part. If interested, you can compare the original `fly.toml` file with the modified version.

```
# fly.toml app configuration file generated for chatbot-hongduan on
2024-02-18T14:33:24+08:00
#
```

```
# See https://fly.io/docs/reference/configuration/ for information
about how to use this file.
#

app = 'chatbot-hongduan'
primary_region = 'nrt'

[build]
  builder = 'paketobuildpacks/builder:base'

[env]
  PORT = '8080'

[[services]]
  internal_port = 8000
  protocol = "tcp"

  [services.concurrency]
    hard_limit = 25
    soft_limit = 20
    type = "connections"

  [[services.ports]]
    handlers = ["http"]
    port = 80

  [[services.ports]]
    handlers = ["tls", "http"]
    port = 443

[[vm]]
  size = 'shared-cpu-1x'
```

(4) Modify you `Procfile` with:

```
web: python <YOUR_SCRIPT_FILE_NAME>.py
```

```
# TODO: Modify this Procfile to fit your needs
# web: gunicorn app:app
web: python app.py
~
```

5. Push your repository and deploy your app

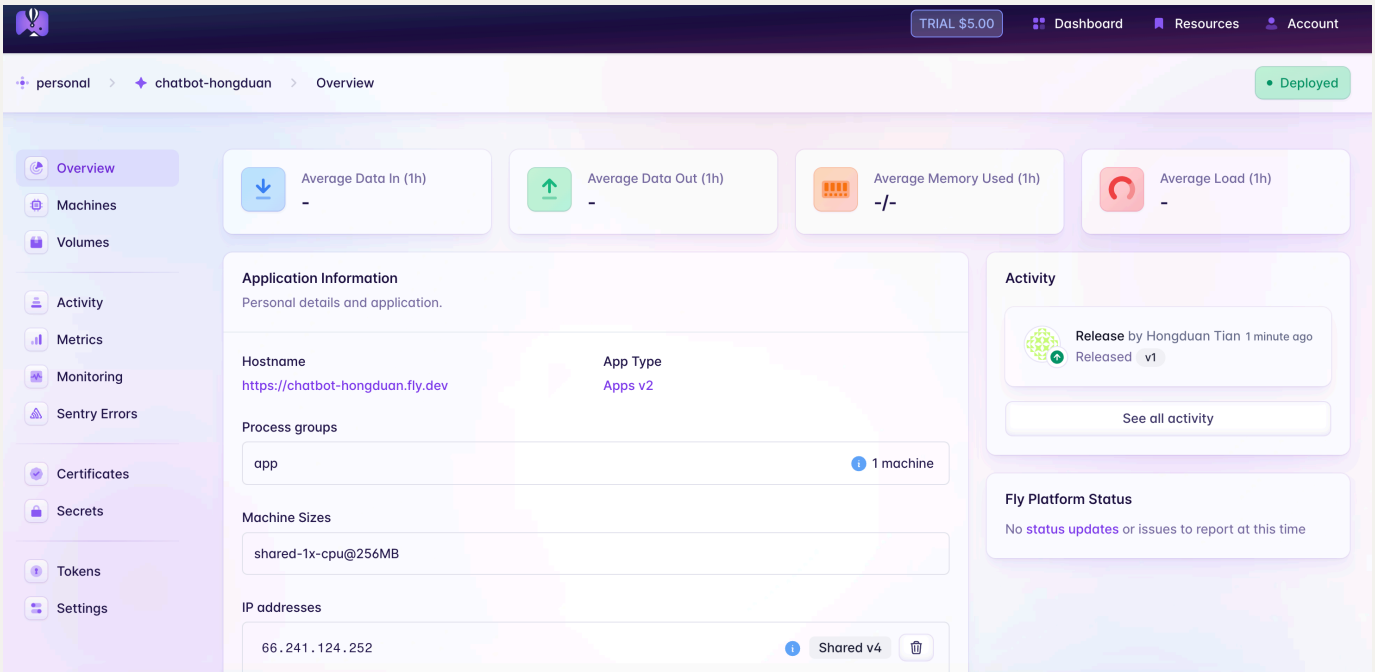
Here, I intentionally skip the practical details of pushing your modifications to the repository you forked before since we have learned this part in our previous lab sessions. If you have forgotten how to push the files to a GitHub repository, you can first review these parts and then come back for this part.

After that, run `fly deploy --remote-only` to deploy your app.

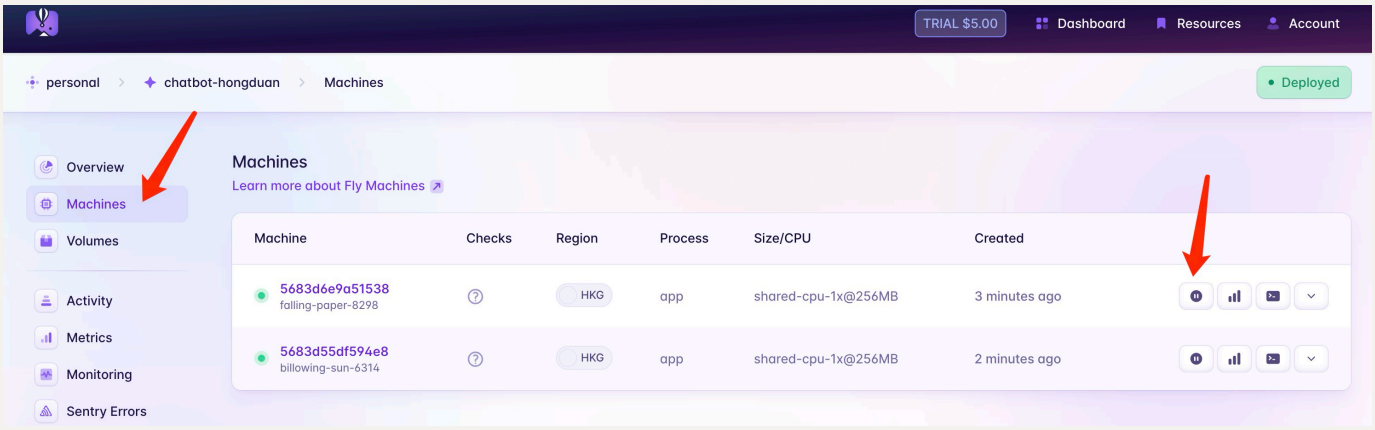
```
(chatbot) tianhongduan@192 chatbot_comp7940 % fly deploy --remote-only
==> Verifying app config
Validating /Users/tianhongduan/上课:教学任务/TA-COMP7940/Lab5/chatbot_comp7940/fly.toml
✓ Configuration is valid
--> Verified app config
==> Building image
Remote builder fly-builder-floral-morning-8104 ready
Remote builder fly-builder-floral-morning-8104 ready
==> Building image with Buildpacks
--> docker host: 20.10.12 linux x86_64
base: Pulling from paketobuildpacks/builder
Digest: sha256:17ea21162ba8c7717d3ead3ee3836a368aced7f02f2e59658e52029bd6d149e7
Status: Image is up to date for paketobuildpacks/builder:base
base-cnb: Pulling from paketobuildpacks/run
Digest: sha256:1af9935d8987fd52b2266d288200c9482d1dd5529860bbf5bc2d248de1cb1a38
Status: Image is up to date for paketobuildpacks/run:base-cnb
===> ANALYZING
Restoring data for SBOM from previous image
===> DETECTING
6 of 9 buildpacks participating
paketo-buildpacks/ca-certificates 3.6.3
paketo-buildpacks/cpython         1.8.11
paketo-buildpacks/pip             0.17.4
paketo-buildpacks/pip-install     0.5.16
paketo-buildpacks/python-start    0.14.11
paketo-buildpacks/procfile        5.6.4
===> RESTORING
Restoring metadata for "paketo-buildpacks/ca-certificates:helper" from app image
Restoring metadata for "paketo-buildpacks/cpython:cpython" from app image
Restoring metadata for "paketo-buildpacks/pip:pip" from cache
Restoring metadata for "paketo-buildpacks/pip-install:packages" from app image
Restoring metadata for "paketo-buildpacks/pip-install:cache" from cache
Restoring data for "paketo-buildpacks/cpython:cpython" from cache
Restoring data for "paketo-buildpacks/pip:pip" from cache
Restoring data for "paketo-buildpacks/pip-install:cache" from cache
Restoring data for "paketo-buildpacks/pip-install:packages" from cache
Restoring data for SBOM from cache
===> BUILDING

Paketo Buildpack for CA Certificates 3.6.3
```

```
8812c86dc680: Layer already exists
88462eecbe63: Pushed
238313ee683f: Layer already exists
8dc7cd674c3b: Layer already exists
e6d3999803da: Layer already exists
6fcdf2880dd8: Layer already exists
1eb5983d7301: Layer already exists
39d381810cef: Layer already exists
115fc79fb3d1: Layer already exists
fd93afbbe1ce: Layer already exists
f92983442b23: Layer already exists
4d274d05ee12: Layer already exists
548a79621a42: Layer already exists
deployment-01HQ0A84ZNM8A97GREW9KA8EEC: digest: sha256:f81a5fe005ce54429386a6e813402d331e2073714391
e63706c6bb4aed60876c size: 3453
--> Pushing image done
image: registry.fly.io/chatbot-hongduan:deployment-01HQ0A84ZNM8A97GREW9KA8EEC
image size: 290 MB

Watch your deployment at https://fly.io/apps/chatbot-hongduan/monitoring

-------
Updating existing machines in 'chatbot-hongduan' with rolling strategy

WARNING The app is not listening on the expected address and will not be reachable by fly-proxy.
You can fix this by configuring your app to listen on the following addresses:
  - 0.0.0.0:8000 for job
Found these processes inside the machine with open listening sockets:
  PROCESS        | ADDRESSES
-----------------*--------------------------------------
  /.fly/hallpass | [fdaa:6:a21f:a7b:1d7:b42a:f51e:2]:22


-------
 ✔ [1/2] Machine 5683d55df594e8 [app] update succeeded
 ✔ [2/2] Machine 5683d6e9a51538 [app] update succeeded
-------
Checking DNS configuration for chatbot-hongduan.fly.dev

Visit your newly deployed app at https://chatbot-hongduan.fly.dev/
```

Then, you can see that your app is deployed in the dashboard of fly.io.



Now, you can test your chatbot by running the app and interacting with it on telegram app by start your app in `Machines` menu.



6. Avoid extra charges [Optional]

In order to avoid extra charges from fly.io, you are recommended to change the IP address via firstly delete the original IP address:

```
fly ips release <ip 4> -a <YOUR_APP_NAME>
```

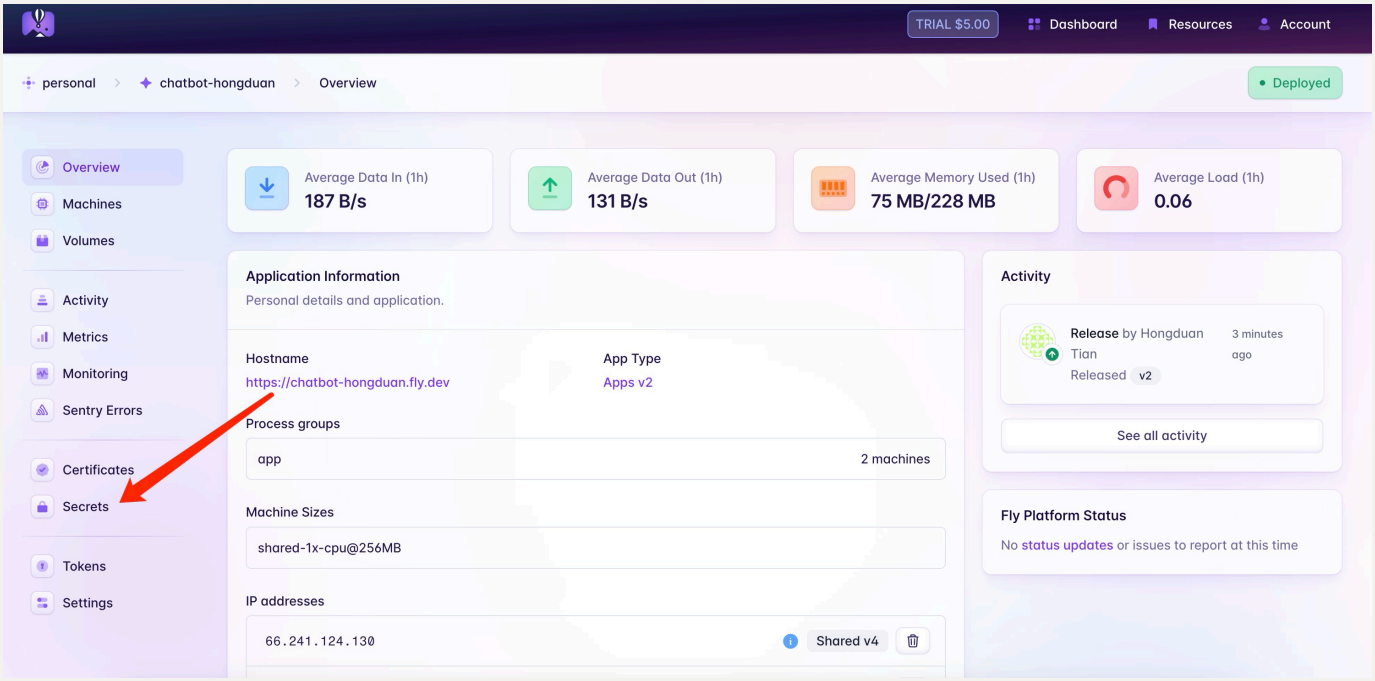
And then, change to the shared IP 4:

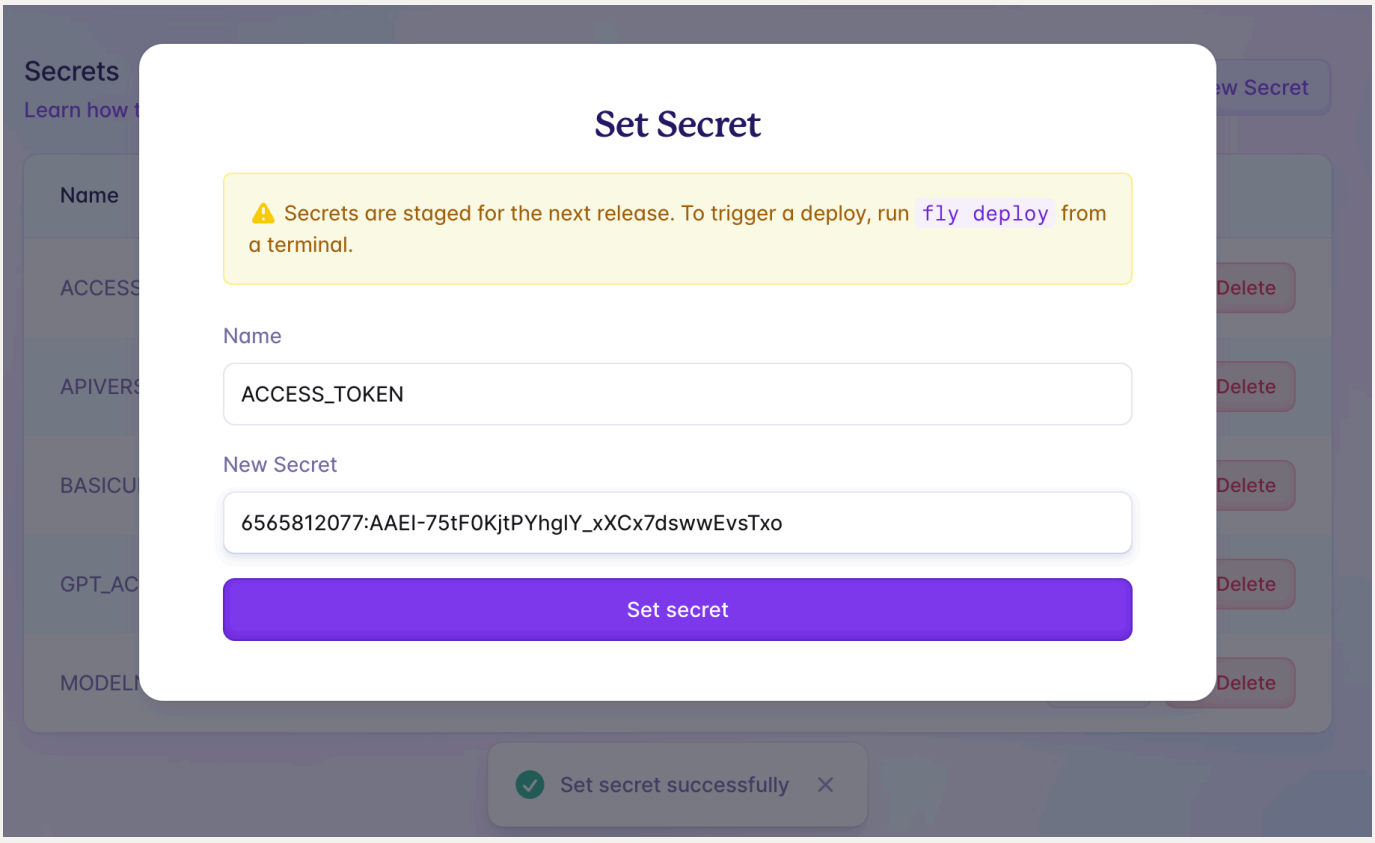
```
fly ips allocate-v4 --shared -a <YOUR_APP_NAME>
```

7. Migrate to a securer version

It is rather unwise to leave your password with the developer. Thus, we need to change the code a little bit.

Firstly, we need to add the corresponding important informations to the secret part of your app.





In this way, you can delet the `config.ini` file and directly read these sensitive information from fly.io's environment with code `os.environ[KEY_WORDS]`. Here, we also intentionally skip other parts where configuration information is required. We hope that you can find out and modify them.

```python
## this file is based on version 13.7 of python telegram chatbot
## and version 1.26.18 of urllib3
## chatbot.py
import os
import telegram
from telegram.ext import Updater, MessageHandler, Filters
# The messageHandler is used for all message updates
#import configparser
import logging

#import subprocess
#import redis_server
#subprocess.Popen([redis_server.REDIS_SERVER_PATH])

#from gptbot import HKBU_GPT
import requests


def main():
    # Load your token and create an Updater for your Bot
    #config = configparser.ConfigParser()
    #config.read('config.ini')
    #print(config['TELEGRAM']['ACCESS_TOKEN'])
    #updater = Updater(token=(config['TELEGRAM']['ACCESS_TOKEN']), use_context=True)
    updater = Updater(token=(os.environ['ACCESS_TOKEN']), use_context=True)
    dispatcher = updater.dispatcher
    # You can set this logging module,
    # so you will know when and why things do not work as expected
    logging.basicConfig(format='%(asctime)s - %(name)s - %(levelname)s - %(message)s', level=logging.INFO)
    # register a dispatcher to handle message:
    # here we register an echo dispatcher
    # echo_handler = MessageHandler(Filters.text & (~Filters.command), echo)
    # dispatcher.add_handler(echo_handler)
    global chatgpt
    #chatgpt = HKBU_GPT(config)
    chatgpt = HKBU_GPT()
    chatgpt_handler = MessageHandler(Filters.text & (~Filters.command),equiped_chatgpt)
    dispatcher.add_handler(chatgpt_handler)
    # To start the bot:
    updater.start_polling()
    updater.idle()
```