

HO CHI MINH UNIVERSITY OF TECHNOLOGY AND EDUCATION

FACULTY OF INFORMATION TECHNOLOGY



FINAL PROJECT

Course name: CLOUD COMPUTING

CLOUD FOUNDRY PLATFORM WITH PYTHON

Course ID: 221CLCO332779E

Class ID: CLCO332779E_22_1_01CLC

Lecturer: MS. Trương Thị Ngọc Phượng

Group 10:

Nguyễn Minh Trí	-	20110422
Phan Thành Luân	-	20110380
Huỳnh Đăng Khoa	-	20110375

Thu Duc city, December 29th, 2021

TASK ASSIGNMENT

Student's name	Evaluate contribution	Taskwork
Nguyễn Minh Trí Leader	100%	Learn Cloud Foundry, do project, write report.
Phan Thành Luân	100%	Learn Cloud Foundry, do project, deploy our project to Cloud Foundry
Huỳnh Đăng Khoa	100%	Learn Cloud Foundry, do project, do PPT.

Note: %: The percentage of each student participating.

Note: %: The percentage of each student participating.

No	Goal	Schedule						
1	Understand & describes the requirements of the project.	o	o					
2	Learn & study technology, language programming to do project	o	o	o				
3	Identify the functions and algorithm to be used in the project and report.	o	o	o			o	
4	Building(Coding) & design of project	o	o	o				

5	Building & design content of report architecture.			o	o	o		
6	Testing the project.					o	o	o
7	Update and Deploy project.					o	o	o
Week		7	8	9	10	12	14	16
Note		o-Begin		O – Complete 50%			O – Complete 100%	

Score:

Remark of teacher:

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Thu Duc city, December 29th, 2022

Signature and full name of teacher

TABLE OF CONTENTS

INTRODUCTION

I. INTRODUCTION TO CLOUD FOUNDRY 1

II. COMPONENTS OF CLOUD FOUNDRY 3

2.1. Routing 3

2.2. Authentication 4

2.3. App Lifecycle 4

2.4. App Storage and Execution 5

2.5. Services 5

2.6. Messaging 6

2.7. Metrics and Logging 6

III. SECURITY AND NETWORKING 7

3.1. System Boundaries and Access 7

3.2. Protocols 9

3.3. Authentication and Authorization 9

3.4. Recommendations for Running a Secure Deployment 10

IV. ORGS, SPACES, ROLES, AND PERMISSIONS 11

4.1. Overview 11

4.2. Orgs 11

4.3. Spaces 11

4.4. User Roles 12

4.5. User Role Permissions 13

V. ADVANTAGES AND DISADVANTAGES OF CLOUD FOUNDRY PLATFORM 14

5.1. Key benefits 14

5.2. Drawback 15

VI. DEPLOYMENT AND OUR PROJECT ON CLOUD FOUNDRY WITH PYTHON 16

6.1. Deploy on cloud with python project 16

6.2. Overview about our project 25

REFERENCE DOCUMENTS 26

INTRODUCTION

We would like to express our gratitude to the professor Mrs. Truong Thi Ngoc Phuong, who helped us personally throughout the topic-making process in order to successfully complete this topic and this report. We appreciate the teacher's advice from his real-world experience in helping us meet the requirements of the chosen topic, as well as his willingness to always respond to our queries and offer suggestions and corrections. Time to assist us in overcoming our flaws and completing it successfully and on time.

We also want to extend our sincere gratitude to the instructors in the High Quality Education Division generally and the Information Technology sector specifically for their committed expertise that has helped us build a foundation. The conditions for learning and performing effectively on the topic have been created by this topic. We also want to express our gratitude to our classmates for sharing expertise and insights that helped us refine our topic.

We created the subject and report in a little amount of time, with little expertise and numerous other technical and software project implementation difficulties. Therefore, as it is inevitable that a topic may have flaws, we eagerly await the lecturers' insightful comments in order to further our knowledge and improve for the next time.

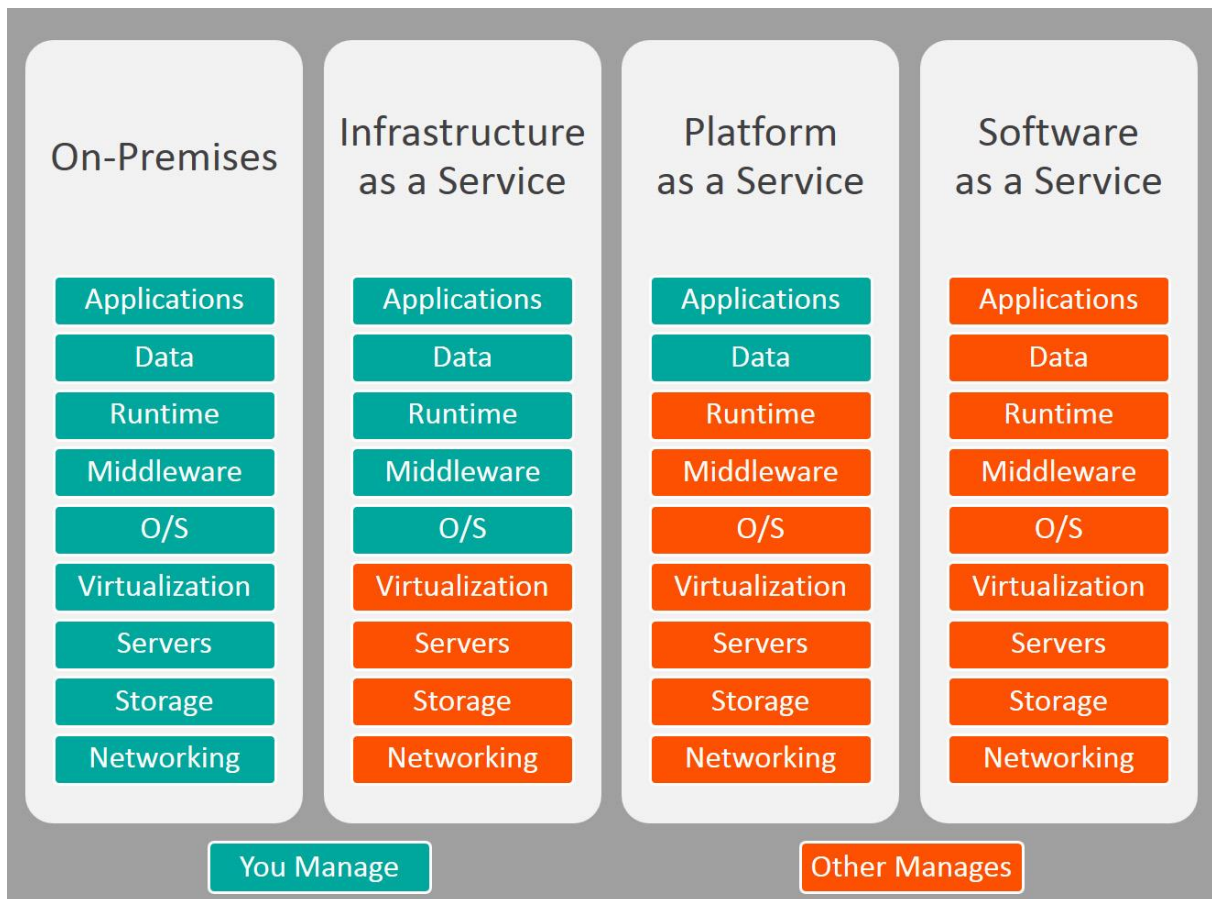
We appreciate you very much.

Finally, we would want to wish all of you teachers, ladies and gentlemen, continued health and success in your line of work with developing individuals. Again, we appreciate your kind words.

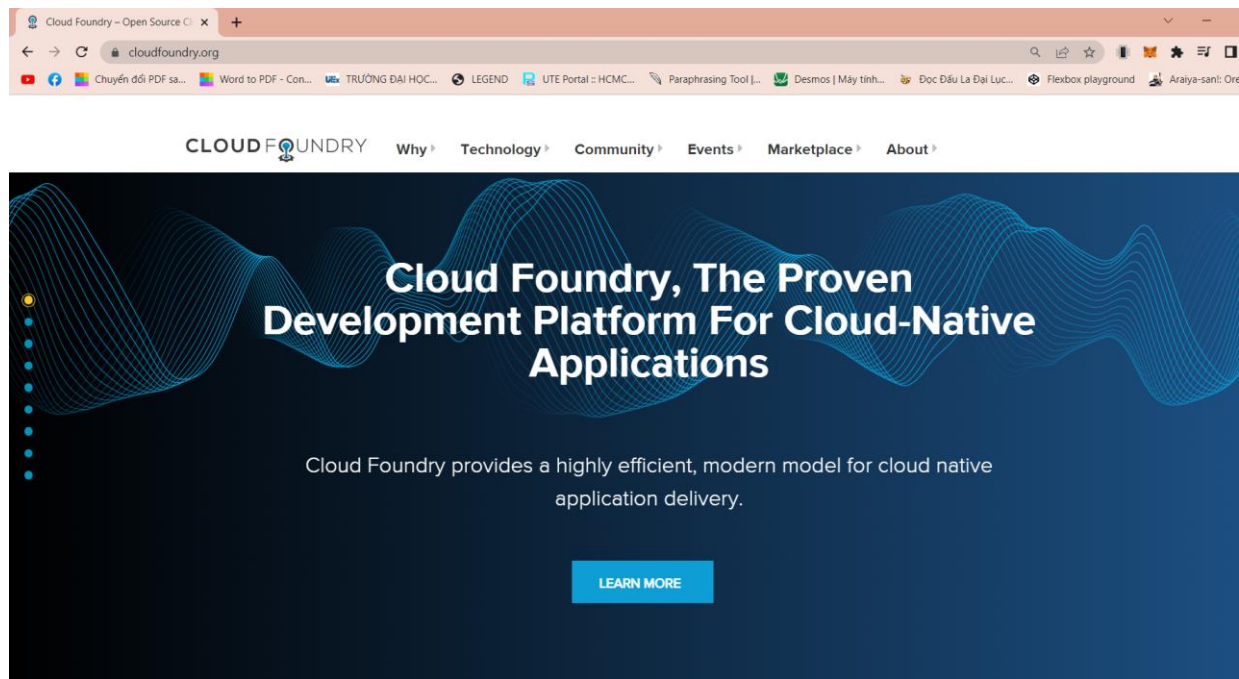
We sincerely thank you.

I. INTRODUCTION TO CLOUD FOUNDRY

Cloud Foundry is an open-source cloud Platform-as-a-Service (PaaS) on which developers can build, deploy, run and scale applications. It's written in Ruby, but intended to host any language or any other component.



- IaaS: The developer who creates the app must build the OS, Run time environment, etc. and finally deploy the app to test the types.
- PaaS: The developer who creates the app must click the deploy app button to test the types.
- SaaS: The developer who creates the app clicks a button to have a report of everything that has been declared.



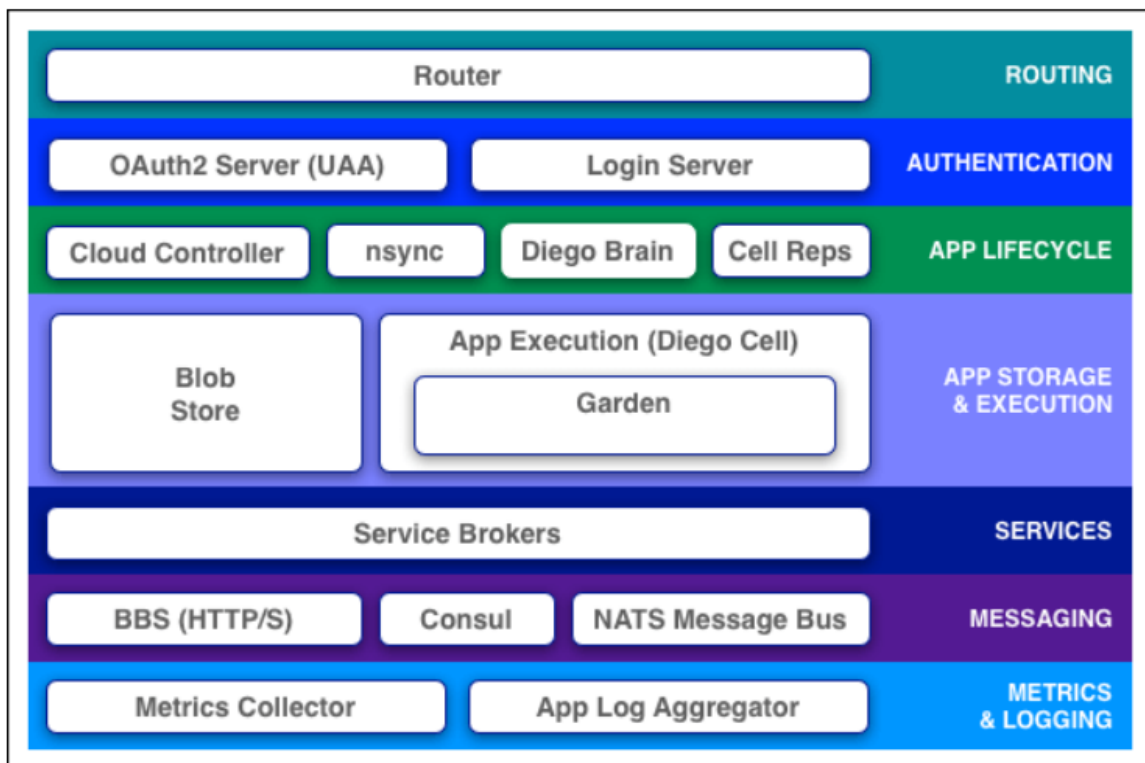
Originally conceived in 2009, Cloud Foundry was designed and developed by a small team at VMware led by Derek Collison and was originally called Project B29.

In December 2015, the Cloud Foundry Foundation announced the “Cloud Foundry PaaS Certification program” which delineated criteria to be considered a Cloud Foundry Certified Provider.

While other cloud platform services are tied to particular cloud providers (such as Amazon, Google or Microsoft), Cloud Foundry is available as a stand-alone software package. Enterprises may deploy Cloud Foundry on AWS, Azure or Google Cloud, or they may host it on their own OpenStack server or through HP’s Helion or VMware’s vSphere.

II. COMPONENTS OF CLOUD FOUNDRY

Cloud Foundry components include a self-service application execution engine, an automation engine for application deployment and lifecycle management, and a scriptable command line interface (CLI), as well as integration with development tools to ease deployment processes. Cloud Foundry has an open architecture that includes a buildpack mechanism for adding frameworks, an application services interface, and a cloud provider interface.



Components of cloud foundry

2.1. Routing

The router routes incoming traffic to the appropriate component, either a Cloud Controller component or a hosted application running on a Diego Cell.

The router periodically queries the Diego Bulletin Board System (BBS) to determine which cells and containers each application currently runs on. Using this information, the router recomputes new routing tables based on the IP addresses of each cell virtual machine (VM) and the host-side port numbers for the cell's containers.

2.2. Authentication

OAuth2 Server (UAA) and Login Server

The OAuth2 server (the UAA) and Login Server work together to provide identity management.

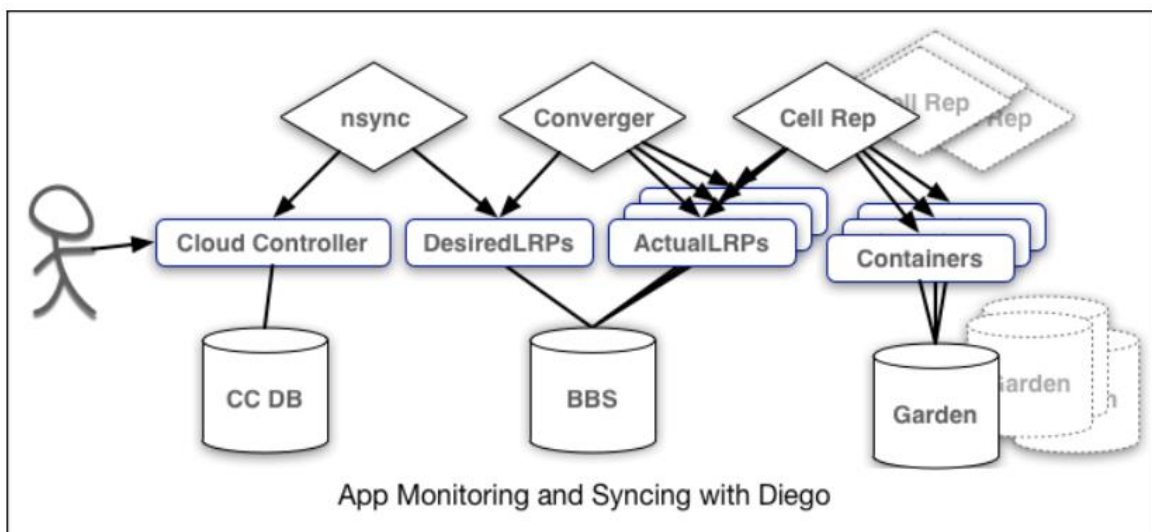
2.3. App Lifecycle

Cloud Controller and Diego Brain

The Cloud Controller (CC) directs the deployment of applications. To push an app to Cloud Foundry, you target the Cloud Controller. The Cloud Controller then directs the Diego Brain through the CC-Bridge components to coordinate individual Diego cells to stage and run applications.

The Cloud Controller also maintain records of orgs, spaces, user roles, services, and more.

To keep applications available, cloud deployments must constantly monitor their states and reconcile them with their expected states, starting and stopping processes as required.



The nsync, BBS, and Cell Rep components work together along a chain to keep apps running. At one end is the user. At the other end are the instances of applications running on widely-distributed VMs, which may crash or become unavailable.

Here is how the components work together:

- **nsync** receives a message from the Cloud Controller when the user scales an app. It writes the number of instances into a `DesiredLRP` structure in the Diego BBS database.
- **BBS** uses its convergence process to monitor the `DesiredLRP` and `ActualLRP` values. It launches or stops application instances as appropriate to ensure the `ActualLRP` count matches the `DesiredLRP` count.
- **Cell Rep** monitors the containers and provides the `ActualLRP` value.

2.4. App Storage and Execution

Blobstore

The blobstore is a repository for large binary files, which Github cannot easily manage because GitHub is designed for code. The blobstore contains the following:

- Application code packages
- Buildpacks
- Droplets

You can configure the blobstore as either an internal server or an external S3 or S3-compatible endpoint.

Diego Cell

Application instances, application tasks, and staging tasks all run as Garden containers on the Diego Cell VMs. The Diego cell rep component manages the lifecycle of those containers and the processes running in them, reports their status to the Diego BBS, and emits their logs and metrics to Loggregator.

2.5. Services

Applications typically depend on services such as databases or third-party SaaS providers. When a developer provisions and binds a service to an application, the service broker for that service is responsible for providing the service instance.

2.6. Messaging

Internal HTTPS and BBS

Cloud Foundry component VMs communicate with each other internally through HTTP and HTTPS protocols, sharing temporary messages and data stored in Diego's [Bulletin Board System \(BBS\)](#).

- BOSH Director colocates a [BOSH DNS](#) server on every deployed VM. All VMs keep up-to-date DNS records for all the other VMs in the same foundation, enabling service discovery between VMs. BOSH DNS also provides client-side load-balancing by randomly selecting a healthy VM when multiple VMs are available.
- Diego's [Bulletin Board System \(BBS\)](#) stores more frequently updated and disposable data such as cell and app status, unallocated work, and heartbeat messages, as well as longer-lived distributed locks. The BBS stores data in MySQL, using the [Go MySQL Driver](#).

The route-emitter component uses the NATS protocol to broadcast the latest routing tables to the routers.

2.7. Metrics and Logging

Loggregator

The Loggregator (log aggregator) system streams application logs to developers.

Metrics Collector

The metrics collector gathers metrics and statistics from the components. Operators can use this information to monitor a Cloud Foundry deployment.

III. SECURITY AND NETWORKING

Cloud Foundry implements the following measures to mitigate against security threats:

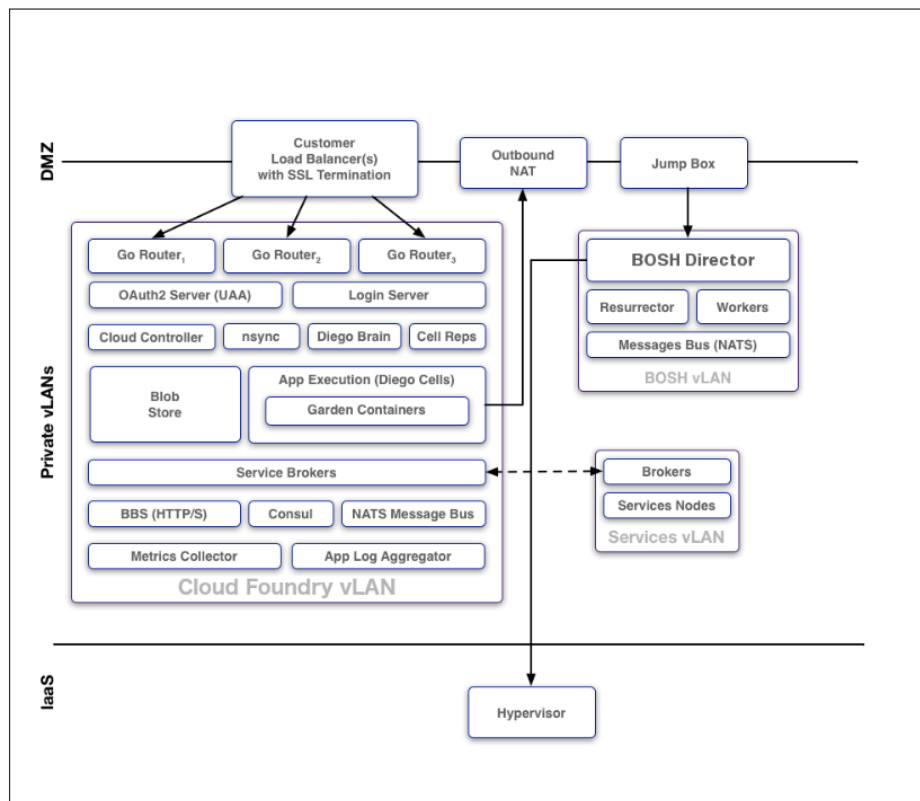
- Minimizes network surface area
- Isolates customer apps and data in containers
- Encrypts connections
- Uses role-based access controls, applying and enforcing roles and permissions to ensure that users can only view and affect the spaces for which they have been granted access
- Ensures security of app bits in a multi-tenant environment
- Prevents possible denial of service attacks through resource starvation

3.1. System Boundaries and Access

In a typical deployment of Cloud Foundry, the components run on virtual machines (VMs) that exist within a VLAN.

In this configuration, the only access points visible on a public network are a load balancer that maps to one or more Cloud Foundry routers and, optionally, a NAT VM and a jumpbox. Because of the limited number of contact points with the public internet, the surface area for possible security vulnerabilities is minimized.

As the diagram below shows the architecture of a typical deployment of Cloud Foundry:



The above diagram shows the following components:

- DMZ
 - Customer Load Balancers with SSL Termination
 - Outbound NAT
 - Jumpbox
- Private vLANs
 - Cloud Foundry vLAN
 - Three Go Routers
 - OAuth2 Server (UAA)
 - Login Server
 - Cloud Controller
 - nsync
 - Diego Brain
 - Cell Reps
 - Blob Store
 - App Execution (Diego Cells) with Garden Containers
 - Service Brokers
 - BBS (HTTP/S)

- Consul
- NATS Message Bus
- Metrics Collector
- App Log Aggregator
- BOSH vLAN
 - BOSH Director
 - Resurrector
 - Workers -Message Bus (NATS)
- Services vLAN
 - Brokers
 - Services Nodes
- IaaS
 - Hypervisor

The diagram above shows the following communications:

- Customer Load Balancers send communication to the three Go Routers
- Outbound NAT communicates with Hypervisor
- BOSH Director communicates with the App Execution (Diego Cells)
- Service Brokers in Cloud Foundry vLAN communicates with Brokers in Services vLAN

3.2. Protocols

All traffic from the public internet to the Cloud Controller and UAA happens over HTTPS. Inside the boundary of the system, components communicate over a publish-subscribe (pub-sub) message bus NATS, HTTP, and SSL/TLS.

3.3. Authentication and Authorization

[User Account and Authentication](#) (UAA) is the central identity management service for Cloud Foundry and its various components.

UAA acts as an [OAuth2](#) Authorization Server and issues access tokens for apps that request platform resources. The tokens are based on the [JSON Web Token](#) and are digitally signed by UAA.

Operators can configure the identity store in UAA. If users register an account with the Cloud Foundry platform, UAA acts as the user store and stores user passwords in the UAA database using [bcrypt](#). UAA also supports connecting to external user stores through LDAP and SAML. Once an operator has configured the external user store, such as a corporate Microsoft Active Directory, users can use their LDAP credentials to gain access to the Cloud Foundry platform instead of registering a separate account. Alternatively, operators can use SAML to connect to an external user store and enable single sign-on for users into the Cloud Foundry platform.

3.4. Recommendations for Running a Secure Deployment

To help run a secure deployment, Cloud Foundry recommends:

- Configure UAA clients and users using a BOSH manifest. Limit and manage these clients and users as you would any other kind of privileged account.
- Deploy within a VLAN that limits network traffic to individual VMs. This reduces the possibility of unauthorized access to the VMs within your BOSH-managed cloud.
- Enable HTTPS for apps and SSL database connections to protect sensitive data transmitted to and from apps.
- Ensure that the jumpbox is secure, along with the load balancer and NAT VM.
- Encrypt stored files and data within databases to meet your data security requirements. Deploy using industry standard encryption and the best practices for your language or framework.
- Prohibit promiscuous network interfaces on the trusted network.
- Review and monitor data sharing and security practices with third-party services that you use to provide additional functionality to your app.
- Store SSH keys securely to prevent disclosure, and promptly replace lost or compromised keys.
- Use Cloud Foundry's RBAC model to restrict your users' access to only what is necessary to complete their tasks.
- Use a strong passphrase for both your Cloud Foundry user account and SSH keys.

IV. ORGS, SPACES, ROLES, AND PERMISSIONS

4.1. Overview

Cloud Foundry uses a role-based access control (RBAC) system to grant appropriate permissions to Cloud Foundry users.

Admins, Org Managers, and Space Managers can assign user roles using the Cloud Foundry.

4.2. Orgs

An org is a development account that an individual or multiple collaborators can own and use. All collaborators access an org with user accounts, which have roles such as Org Manager, Org Auditor, and Org Billing Manager. Collaborators in an org share a resource quota plan, apps, services availability, and custom domains.

By default, an org has the status of *active*. An admin can set the status of an org to *suspended* for various reasons such as failure to provide payment or misuse. When an org is suspended, users cannot perform certain activities within the org, such as push apps, modify spaces, or bind services.

4.3. Spaces

A space provides users with access to a shared location for app development, deployment, and maintenance. An org can contain multiple spaces. Every app, service, and route is scoped to a space. Roles provide access control for these resources and each space role applies only to a particular space.

Org managers can set quotas on the following for a space:

- Usage of paid services
- Number of app instances
- Number of service keys
- Number of routes
- Number of reserved route ports

- Memory used across the space
- Memory used by a single app instance
- Log volume per second used across the space

4.4. User Roles

A user account represents an individual person within the context of a Cloud Foundry foundation. A user can have one or more roles. These roles define the user's permissions in orgs and spaces.

Roles can be assigned different scopes of User Account and Authentication (UAA) privileges. For more information about UAA scopes.

The following describes each type of user role in Cloud Foundry:

- **Admin:** Perform operational actions on all orgs and spaces using the Cloud Controller API. Assigned the `cloud_controller.admin` scope in UAA.
- **Admin Read-Only:** Read-only access to all Cloud Controller API resources. Assigned the `cloud_controller.admin_read_only` scope in UAA.
- **Global Auditor:** Read-only access to all Cloud Controller API resources except for secrets, such as environment variables. The Global Auditor role cannot access those values. Assigned the `cloud_controller.global_auditor` scope in UAA.
- **Org Managers:** Administer the org.
- **Org Auditors:** Read-only access to user information and org quota usage information.
- **Org Billing Managers:** Create and manage billing account and payment information.

Note: The Billing Manager role is only relevant for Cloud Foundry environments deployed with a billing engine.

- **Org Users:** Read-only access to the list of other org users and their roles. In the v2 Cloud Controller API, when an Org Manager gives a person an Org or Space role, that person automatically receives Org User status in that org. This is no longer the case in the V3 Cloud Controller API.

- **Space Managers:** Administer a space within an org.
- **Space Developers:** Manage apps, services, and space-scoped service brokers in a space.
- **Space Auditors:** Read-only access to a space.
- **Space Supporters:** Troubleshoot and debug apps and service bindings in a space.

Note: The Space Supporter role is only available for the Cloud Controller V3 API. If a user with this role tries to access a V2 endpoint, the API returns a 403.

For non-admin users, the `cloud_controller.read` scope is required to view resources, and the `cloud_controller.write` scope is required to create, update, and delete resources.

Before you assign a space role to a user, you must assign an org role to the user. The error message `Server error, error code: 1002, message: cannot set space role because user is not part of the org` occurs when you try to set a space role before setting an org role for the user.

4.5. User Role Permissions

Each user role includes different permissions in a Cloud Foundry foundation.

V. ADVANTAGES AND DISADVANTAGES OF CLOUD FOUNDRY PLATFORM

5.1. Key benefits

- Open-source format – Even though there are lots of commercial versions of Cloud Foundry, it's still an open-source project with a large international community and quite an impressive knowledge base. You can easily find Cloud Foundry tutorials and technical guides on the project's website.
- Support for different programming languages – This is perhaps the biggest advantage of this platform. Currently, it supports the most popular programming languages including Java, PHP, Ruby, Node.js, .Net, Go (Google's Golang), and more.
- Multi-Vendor support – The platform creates a healthy, multi-vendor environment where you can choose from several vendors.
- Application portability – Applications deployed to the platform can be easily migrated from a private data center to the public cloud as well as from one IaaS provider to another.
- Role-based access for deployed applications – The platform improves the security of applications by ensuring that only authorized groups of users have access to application data.
- Ease of application deployment – Code can be deployed by executing just a single line from the CLI. At the same time, the platform's CLI includes a large variety of commands allowing for various operations on application instances.
- Access to applications with SSH – You can enable or disable SSH access to your applications. When enabled, SSH access allows you to run diagnostics and view application code in real time. You can access your application's code by using either the platform's standard client, CF SSH, or other SSH tools such as SCP or SFTP clients.
- Horizontal and vertical scaling – You can scale the capacity of the platform both horizontally and vertically by adding more Virtual Machines running instances and by adding disk space and memory, respectively.
- Software vulnerability management – System updates and BOSH stemcells are released on a regular basis to ensure a high level of software protection and to patch the latest security issues.

5.2. Drawback

Aside from the common issues listed behind by the Cloud Foundry Foundation, there are several concerns about the platform, including:

- No support for stateful containers – Unfortunately, the platform still doesn't support stateful containers.
- Logging issues – While the platform supports showing logs, it doesn't seem to persist these logs anywhere.
- Not enough focus on operational processes – Even though Cloud Foundry takes a lot of load off operations teams, the main focus of the project is still on developers.
- Complexity of configuration and monitoring processes – Working with BOSH may be quite challenging, especially for less experienced workers.
- Need to follow Twelve-Factor App standards – Despite claiming to be a universal solution and supporting numerous frameworks and languages, Cloud Foundry works best for applications that are built in accordance with the Twelve-Factor App methodology.

VI. DEPLOYMENT AND OUR PROJECT ON CLOUD FOUNDRY WITH PYTHON

Deploy the application developed by Python that using the web framework Flask

- **Prerequisites:** You need to create an account and assign your own free trial Cloud Foundry Environment of SAP Cloud Platform that you can use freely by [this link](#).

6.1. Deploy on cloud with python project

✓ Step 1: Log on to SAP BTP & Set up essential files for server

First, you need to connect to the SAP BTP, Cloud Foundry environment with your enterprise (productive) subaccount. Your Cloud Foundry URL depends on the region where the API endpoint belongs to. To find out which one is yours, see: Regions and API Endpoints Available for the CF Environment

Open a command-line console.

You can download the Cloud Foundry CLI by [this link in github](#).

Set the Cloud Foundry API endpoint for your subaccount. Execute (using your actual region URL):

Bash/Shell

```
1 | cf api https://api.cf.eu20.hana.ondemand.com
```

Log in to SAP BTP, Cloud Foundry environment:

Bash/Shell

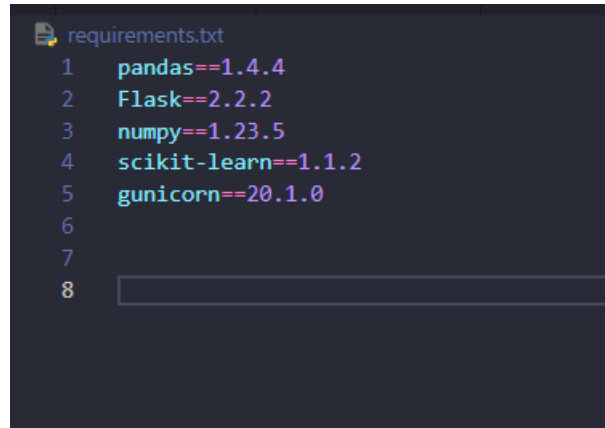
```
1 | cf login
```

Manifest.yml file

```
manifest.yml
1  ---
2  applications:
3  - name: myapp
4  - routes:
5    - route: python-1234-flowerpredict-5678.cfapps.ap21.hana.ondemand.com
6    path: ./
7    memory: 128M
8    buildpack: python_buildpack
9    command: python server.py
```

This manifest.yml file represents the configuration, describing your application and how it will be deployed to Cloud Foundry, includes the name, memory capacity, path, buildpack and command for running application

Requirement file

A screenshot of a code editor showing a file named 'requirements.txt'. The file contains five lines of code, each specifying a library and its version with an equals sign and two dots (==). The lines are: 1 pandas==1.4.4, 2 Flask==2.2.2, 3 numpy==1.23.5, 4 scikit-learn==1.1.2, and 5 gunicorn==20.1.0. The line numbers 1 through 8 are visible on the left side of the editor, with line 8 being empty.

```
requirements.txt
1 pandas==1.4.4
2 Flask==2.2.2
3 numpy==1.23.5
4 scikit-learn==1.1.2
5 gunicorn==20.1.0
6
7
8
```

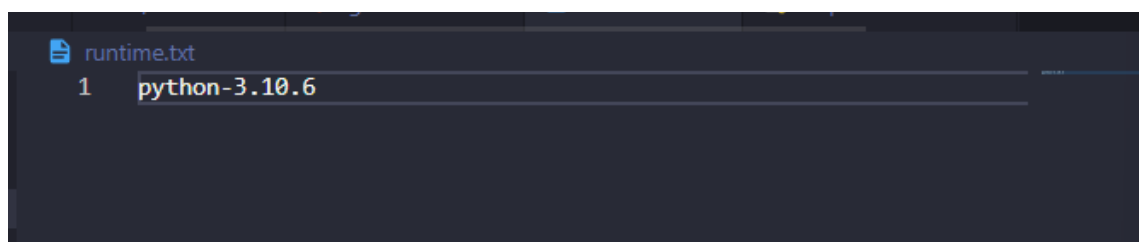
Contain the list of libraries (modules) that application is depending on . The list libraries are automatically installed by the command “pip install” during the deployment of application (Example: the execution of “cf push” command). This application will be a web server utilizing the Flask web framework ,so it has to include the Flask library.

Note:

You can use the older or newer version by using “>=” or “<=” and omit the version number.

You can check your version of each library in command prompt using command “pip freeze” to list the available libraries and its version.

Runtime file

A screenshot of a code editor showing a file named 'runtime.txt'. The file contains a single line of code: 1 python-3.10.6. The line number 1 is visible on the left side of the editor.

```
runtime.txt
1 python-3.10.6
```

In this file, we specify the Python runtime version that your application will run on.

Note:

To request the latest Python version in a patch line , replace the patch version with “x”

To request the latest Python version in a patch line , replace the minor version with “3.x”

You can check your version of Python in command prompt using command “python -V “

✓ Step 2: Create a Python application

**Html file:*

Home page:

```
<html>
<head>
  <Title>Home Page</Title>
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" integrity="sha384-rbsA2VB"
  <link rel="stylesheet" href="static/css/style.css">
</head>
<body>
  <center>

  <h1 class = "text-center"> IRIS FLOWER DETECTION </h1><br>

  <form method="POST", action="{{url_for('home')}}">
    <b> Sepal Length : <input type="text", name='a', placeholder="enter 1"> <br><br>
      Sepal Width : <input type="text", name='b', placeholder="enter 2"> <br><br>
      Petal Length : <input type="text", name='c', placeholder="enter 3"> <br><br>
      Petal Width : <input type="text", name='d', placeholder="enter 4"> <br><br><br></b>
    <input id = "btn_predict", type="submit" , value='PREDICT!!!' >
  </form>

  <img src='static\Homebackground1.png' alt="flower">

</center>

</body>
</html>
```

Result page:

```
<html>
<head>
  <Title>Prediction Result</Title>
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" integrity="sha384-rbsA2VBKQhggwzxd7pPCaAq046Mgn0"
  <link rel="stylesheet" href="static/css/style.css">
</head>
<body bgcolor=#a3cfb4>

  <center>

    <h1> PREDICTION </h1>

    {%if data == 0%}
    <h1>Iris-setosa</h1>
    <img src='static\setosa.jpg'>
    {%elif data == 1%}
    <h1>Iris-versicolor</h1>
    <img src='static\verci.jpg'>
    {%elif data == 2%}
    <h1>Iris-virginica</h1>
    <img src='static\iris_virginica_virginica_lg.jpg'>
    {%else%}
    <h1>There are no flowers with this ratio</h1>
    <img src='static\no-answer.jpg'>
    {%endif%}

    <br><br>
    <button>
      <a href='/', id="btn_Back">GO BACK TO THE HOME PAGE</a>
    </button>

  </center>

</body>
</html>
```

Decoration (using css) to display the application(python):

```
1 body{
2     background-image: linear-gradient(to left top, #00044f, #7f005a, #ce2646, #f3801e, #e8d80e);
3 }
4 button{
5     background-image: linear-gradient(to right bottom, #cc0922, #d90048, #d50078, #b700b0, #5f0ee8);
6     border: none;
7     width: 295px !important;
8     height: 50px !important;
9     padding: 15px;
10    text-align: center;
11    text-decoration: none;
12    display: inline-block;
13    font-size: 16px;
14    color: #rgb(250, 243, 243);
15 }
16 button:hover{
17     text-decoration: none;
18     border-style: solid;
19     border-width: 3px;
20     border-color: #rgb(243, 239, 239) ;
21     background-image: linear-gradient(to right bottom, #cc0922, #d90048, #d50078, #b700b0, #5f0ee8);
22     color: #rgb(250, 243, 243) !important;
23     text-align: center;
24 }
25 #btn_predict{
26     background-image: linear-gradient(to right bottom, #cc0922, #d90048, #d50078, #b700b0, #5f0ee8);
27     border: none;
28     padding: 15px ;
29     width: 295px !important;
30     height: 50px !important;
31     text-align: center;
32     text-decoration: none;
33     display: inline-block;
34     font-size: 16px;
35     color: #rgb(250, 243, 243) !important;
36 }
37 #btn_predict:hover{
38     border-style: solid;
39     border-width: 3px;
40     border-color: #rgb(247, 243, 243) ;
41     background-image: linear-gradient(to right, #cc0922, #d90048, #d50078, #b700b0, #5f0ee8);
42     color: #rgb(242, 239, 239) !important;
43 }
```


ML/DL app and server

*File for getting and training data on the iris-dataset - Using SVC Algorithm to train model ,
and save model to “iri.pkl” file

```
iris.py
1  import pandas as pd
2  import numpy as np
3  import pickle
4
5  df = pd.read_csv('iris.data')
6
7  X = np.array(df.iloc[:, 0:4])
8  y = np.array(df.iloc[:, 4:])
9
10 from sklearn.preprocessing import LabelEncoder
11 le = LabelEncoder()
12 y = le.fit_transform(y.reshape(-1))
13
14 from sklearn.model_selection import train_test_split
15 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
16
17 from sklearn.svm import SVC
18 sv = SVC(kernel='linear').fit(X_train,y_train)
19
20
21 pickle.dump(sv, open('iri.pkl', 'wb'))
22
```

Server.py – the sole Python source code file of this application

```
server.py > home
1  from flask import Flask, render_template, request
2  import pickle
3  import numpy as np
4  import os
5
6  model = pickle.load(open('iri.pkl', 'rb'))
7  app = Flask(__name__)
8  port = int(os.environ.get('PORT', 3000))
9
10
11 @app.route('/')
12 def man():
13     return render_template('home.html')
14
15
16 @app.route('/predict', methods=['POST'])
17 def home():
18     data1 = request.form['a']
19     data2 = request.form['b']
20     data3 = request.form['c']
21     data4 = request.form['d']
22     arr = np.array([[data1, data2, data3, data4]])
23     pred = model.predict(arr)
24     return render_template('after.html', data=pred)
25
26
27 if __name__ == "__main__":
28     app.run(host='0.0.0.0', port=port, debug=True)
29
30
31
```

[1] – Import Essential libraries.

[2] – Loading model from 'iri.pkl', instantiating oneself as app, and setting port number for fetching from env variable.

[3] – Route handling for home page. If the request path is "/", it will show the home page (home.html).

[4] – This section is for the path "/predict" and the request method "POST". When it is called, it will predict the result with the passing value that user enter and submit from the home page and show "after.html" with the result.

[5] – This section is for starting the application by the code "app.run" with specifying the IP address of host, port number and debug option.

✓ Step 3: Deploy the Python application

Create an account on CF (DONE)

Login and deploy app on the available container

Set the Cloud Foundry API endpoint for your subaccount.

```
CPU: 71% | RAM: 9/12GB | 0ms
powershell 9:57:33 PM | Wednesday | Desktop → STUDIES → Semester_III_HK1 → Cloud computing → Final
Project → deploy-ml-model-flask-master cf api https://api.cf.ap21.hana.ondemand.com
Setting api endpoint to https://api.cf.ap21.hana.ondemand.com...
OK
api endpoint: https://api.cf.ap21.hana.ondemand.com
api version: 2.191.0
```

Log in to SAP BTP, Cloud Foundry environment and choose a space

```
CPU: 71% | RAM: 9/12GB | 6s 563ms
powershell 9:57:51 PM | Wednesday | Desktop → STUDIES → Semester_III_HK1 → Cloud co
Project → deploy-ml-model-flask-master cf login
API endpoint: https://api.cf.ap21.hana.ondemand.com

Email: 20110380@student.hcmute.edu.vn

Password:
Authenticating...
OK

Targeted org fb515e9etrial

Select a space:
1. dev
2. pythonapp

Space (enter to skip): 1
Targeted space dev

API endpoint: https://api.cf.ap21.hana.ondemand.com (API version: 2.191.0)
User: 20110380@student.hcmute.edu.vn
Org: fb515e9etrial
Space: dev
```

Then, push app to the BTP

```
CPU: 72% | RAM: 9/12GB | 1m 58s 989ms
powershell 11:28:17 PM | Wednesday | Desktop → STUDIES → Semester_III_HK1 → Cloud computing → Final
Project → deploy-ml-model-flask-master cf push
Pushing from manifest to org fb515e9etrial / space dev as 20110380@student.hcmute.edu.vn...
Using manifest file C:\Users\Admin\Desktop\STUDIES\Semester_III_HK1\Cloud computing\Final_Project\deploy-ml-model-fla
sk-master\manifest.yml
Deprecation warning: Use of 'buildpack' attribute in manifest is deprecated in favor of 'buildpacks'. Please see https:/
/docs.cloudfoundry.org/devguide/deploy-apps/manifest-attributes.html#deprecated for alternatives and other app manifest
deprecations. This feature will be removed in the future.

Getting app info...
Updating app with these attributes...
name: myapp
path: C:\Users\Admin\Desktop\STUDIES\Semester_III_HK1\Cloud computing\Final_Project\deploy-ml-model-fla
sk-master
buildpacks:
  python_buildpack
command: python server.py
disk quota: 4G
health check type: port
instances: 1
memory: 128M
stack: cflinuxfs3
routes:
  python-1234-Animalpredict-5678.cfapps.ap21.hana.ondemand.com
  python-1234-flowerpredict-5678.cfapps.ap21.hana.ondemand.com
```

Deploy successfully

```
Waiting for app to start...

name:          myapp
requested state: started
isolation segment: trial
routes:        python-1234-flowerpredict-5678.cfapps.ap21.hana.ondemand.com,
               python-1234-Animalpredict-5678.cfapps.ap21.hana.ondemand.com
last uploaded: Wed 14 Dec 23:31:19 +07 2022
stack:         cflinuxfs3
buildpacks:    python

type:          web
instances:     1/1
memory usage:  128M
start command: python server.py

state  since      cpu    memory      disk      details
#0    running  2022-12-14T16:32:26Z  98.8%  103.5M of 128M  591.3M of 4G
```

Application after deploying

**On the server:*

Trial Home / fb515e9etrial / trial

Subaccount: trial - Overview

General

Cloud Foundry Environment

Kyma Environment

Entitlements

64

Entitlements

2

Instances and Subscriptions

Subdomain: fb515e9etrial

Tenant ID: b25b8474-5c96-4e12-af97-ffe46c51954f

Subaccount ID: b25b8474-5c96-4e12-af97-ffe46c51954f

Provider: Microsoft Azure

Region: Singapore

Environment: Multi-Environment

Used for Production: No

Beta Features: Disabled

Created By:

Created On: 22 Nov 2022, 15:05:35 (GMT+07:00)

Modified On: 22 Dec 2022, 19:53:43 (GMT+07:00)

Cloud Foundry Environment

API Endpoint: https://api.cf.ap21.hana.ondemand.com

Org Name: fb515e9etrial

Org ID: 50e1ef47-df4d-4186-908a-6a7d5fe94d9d

Manage environment instance

Disable Cloud Foundry

Spaces (2)

Create Space

Name	Applications	Service Instances	
dev	3	0	>
pythonapp	1	0	>

Kyma Environment

Application: myapp - Overview

Started

Restart Start Stop Instance Instance Delete

Application Routes

<https://python-1234-flowerpredict-5678.cfapps.ap21.hana.ondemand.com>
<https://python-1234-Animalpredict-5678.cfapps.ap21.hana.ondemand.com>

Application Information

Instances: 1
Package Uploaded: 22 Dec 2022, 19:55:52 (GMT+07:00) (STAGED)
Buildpack: python_buildpack
Stack: Cloud Foundry Linux-based filesystem (Ubuntu 18.04) (cflinuxfs3)

Instance Details

Instance Memory: 128 MB (available memory 3968 MB)
Instance Disk: 1024 MB
[Change Instance Details](#)

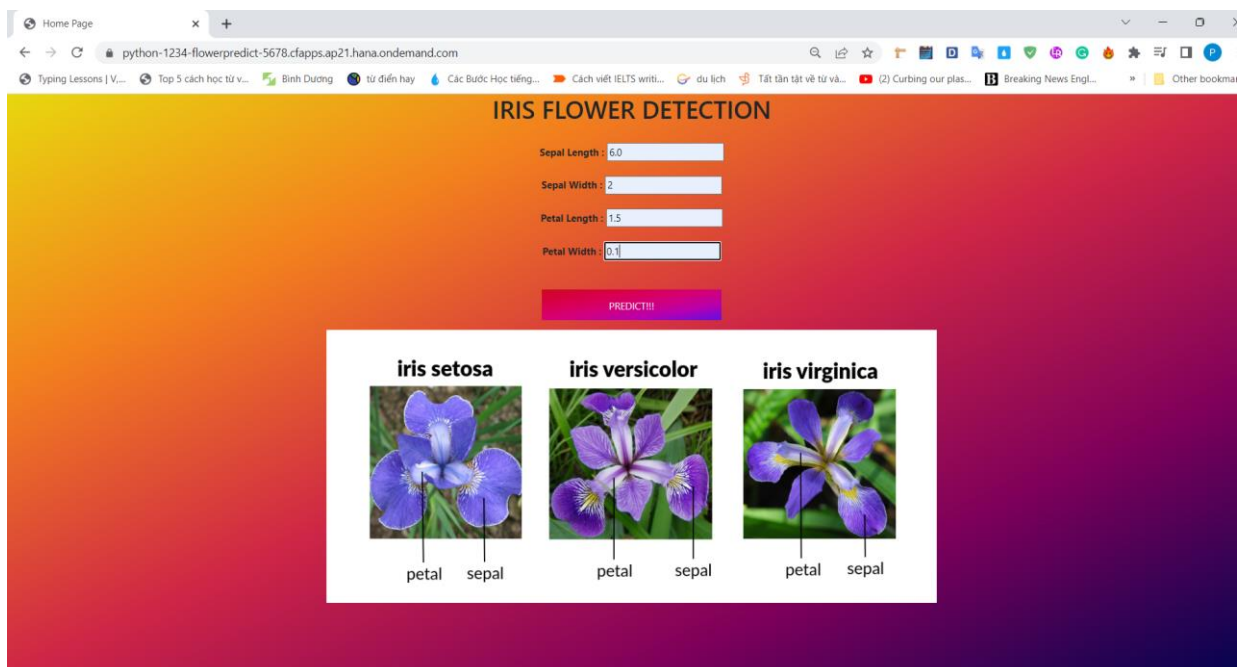
Instances

#	State	Since	CPU	Memory	Disk
0	RUNNING	22 Dec 2022, 19:55:52 (GMT+07:00)	1.6Percentage (%)	105.6 MB	591.3 MB

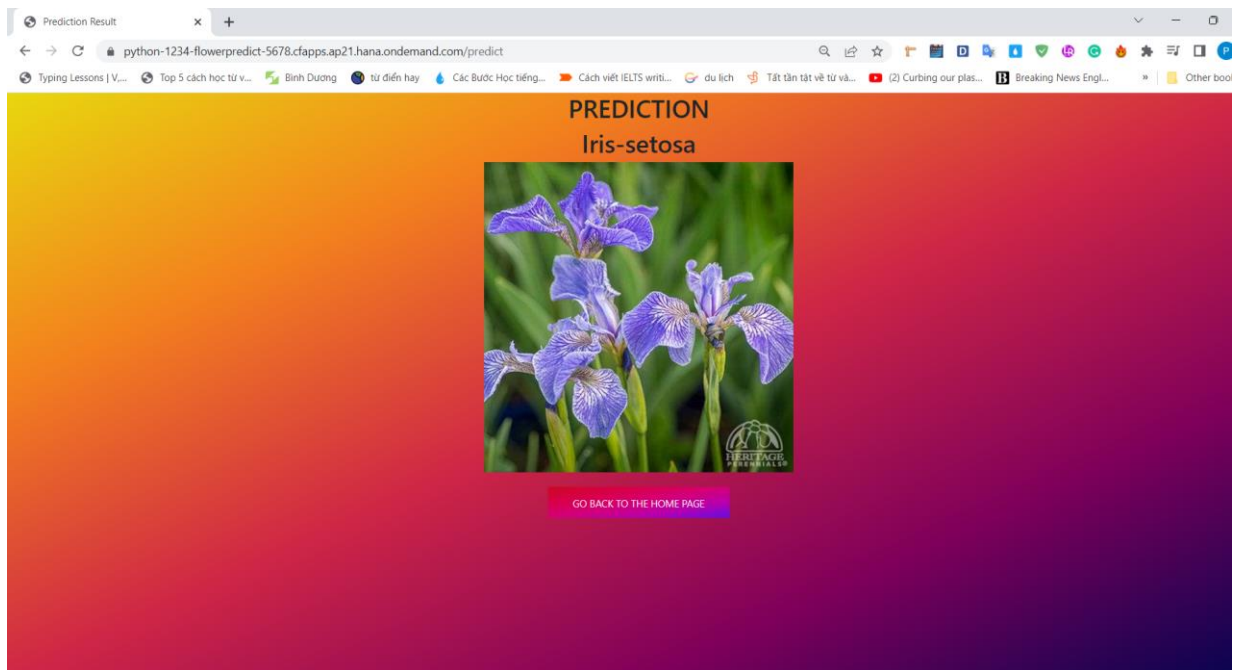
*Result:

You would again get the same URL as shown in the above image, click on that and you are good to go!

You can [click here](#) to go to our project.



This is my team's project, a machine learning program used to recognize the type of iris flower by passing in the parameters of (petal and sepal).



The result we got alter prediction!

6.2. Overview about our project

Our team's topic is a small machine learning program used to recognize iris flowers, a fairly well-known project in machine learning. Here we train in python then transmit and display the program through html file (and a bit of CSS).

we have used some technologies into this program:

- ✓ SVM (Support Vector Machine) & SGD Classification, machine learning models imported from python Scikitlearn library.
- ✓ The iris flower data set is synthesized and statistically in the simplest and most complete way through the Kaggle.com website.

REFERENCE DOCUMENTS

- [1]. Truong Ngoc Phuong (2022). *Lectures on Cloud Computing at HCMUTE*.
- [2]. K. Chandrasekaran. (2015). *Essentials of Cloud Computing*.
- [3]. Cloud Foundry (2022) Cloud Foundry Documentations. Retrieve 11/12/2022 from <https://www.cloudfoundry.org/>
- [4]. Cloud Foundry (2022) Cloud Foundry Components. Retrieve 13/12/2022 from <https://docs.cloudfoundry.org/concepts/architecture/index.html>
- [5]. Maria. (August, 2018). Cloud Foundry: A Platform for Deploying, Running, and Scaling Cloud-Native Applications. Retrieve 13/12/2022 from <https://www.apriorit.com/dev-blog/550-cloud-foundry>
- [6]. *Cloud Foundry* (n.d.). Wikipedia, the free encyclopedia. Retrieve 13/12/2022 from https://en.wikipedia.org/wiki/Cloud_Foundry