Lecture 13: App Design, Setup & Code Organization

AC215

Pavlos Protopapas

SEAS/ Harvard

Outline

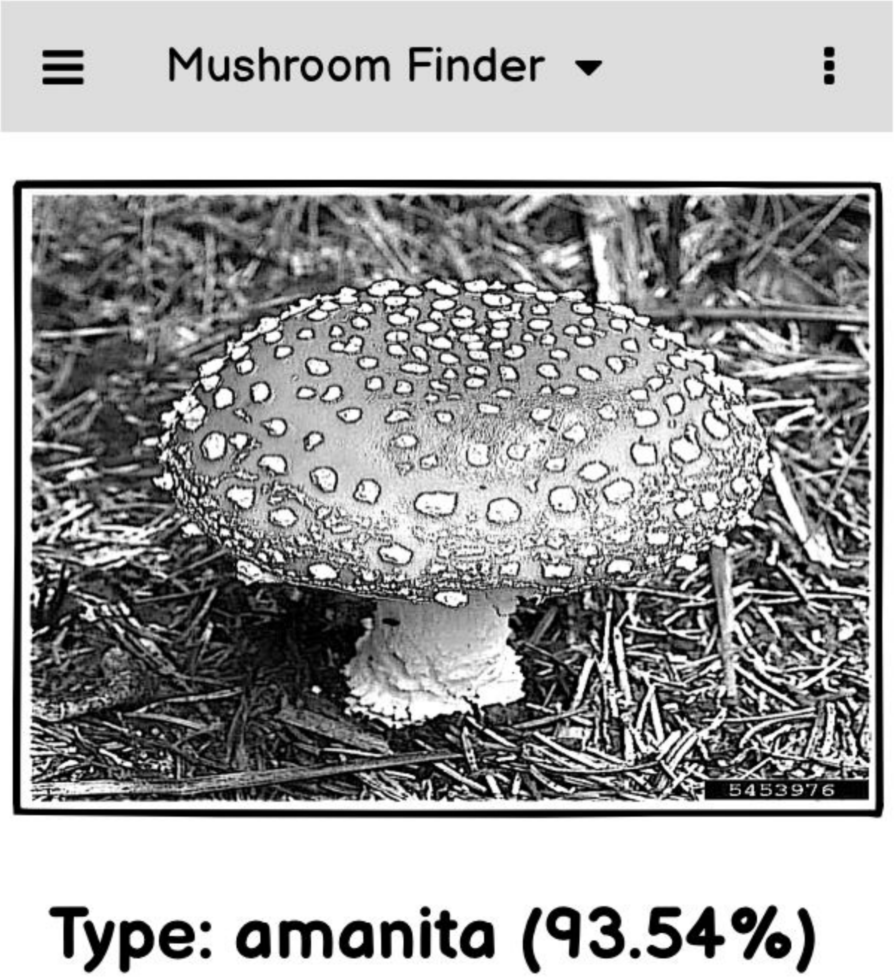
1. Recap
2. Motivation
3. App Design
4. Screenflow & Wireframes
5. Solution Architecture
6. Technical Architecture
7. Setup & Code Organization

Outline

1. **Recap**
2. Motivation
3. App Design
4. Screenflow & Wireframes
5. Solution Architecture
6. Technical Architecture
7. Setup & Code Organization

Recap: 🍄 Mushroom App

- We want to build an app to take a photo of a mushroom and it helps us identify the type of mushroom
- How do we build the app?



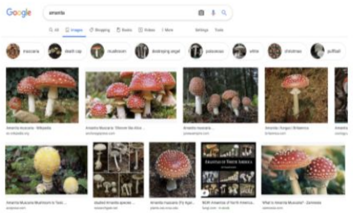
Type: amanita (93.54%)

Recap: How do we build an App?

- Collaborate with team to design and develop.
- Build a robust ml pipeline for data and models.
- Expose python functions as backend APIs.
- Build a frontend using HTML & javascript.
- Deploy app to a cloud provider.
- http://awesome-mushroom-app.com [Go live]

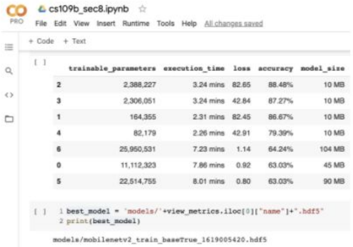
Recap: How do we build an App?

Development



Python Script

Python: pipenv
Chromium: Mac install, Windows install
OS: Mac, Windows



cs109b_sec8.ipynb

	trainable_parameters	execution_time	loss	accuracy	model_size
2	2,388,227	3.24 mins	82.65	88.48%	10 MB
3	2,306,051	3.24 mins	42.84	87.27%	10 MB
1	164,355	2.31 mins	82.45	86.67%	10 MB
4	82,179	2.26 mins	42.91	79.39%	10 MB
6	25,950,531	7.23 mins	1.14	64.24%	104 MB
0	11,112,323	7.86 mins	0.92	63.03%	45 MB
5	22,514,755	8.01 mins	0.80	63.03%	90 MB

```
best_model = 'models/'+view_metrics.iloc[0]["name"]+".hdf5"
print(best_model)
```

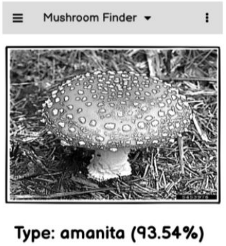
```
models/mobilenetv2_train_baseTrue_1619005420.hdf5
```

Colab

Python: Colab provided env
OS: Linux

Rest API

Best Model



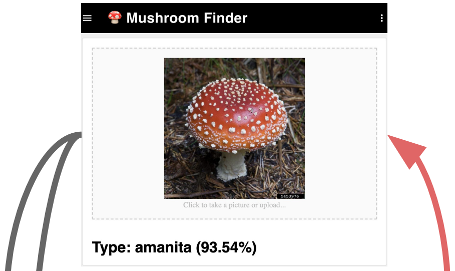
Type: amanita (93.54%)

IDE / Code Editor

Python: pipenv
OS: Mac, Windows

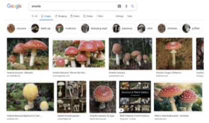
Multiple developers, Using Mac and Windows OS

Deployment



Type: amanita (93.54%)

Model Training



Python Script

Python: pipenv
OS: Linux

Server

Recap: Tools

Data:

- Google Cloud Storage
- Dask
- TensorFlow Data / Records
- Label Studio
- DVC

Model:

- W&B
- Vertex AI Training / Deploy
- WhyLabs

Operations:

- GitHub
- Docker
- Vertex AI Pipelines

Outline

1. Recap
2. **Motivation**
3. App Design
4. Screenflow & Wireframes
5. Solution Architecture
6. Technical Architecture
7. Setup & Code Organization

Before you build your App

- Our **ML Pipeline** is ready
- We want to build an app that uses the **ML Components**
- Expose model and python functions as **APIs**
- Identify **user needs** that can fulfilled by APIs
- Design **user interface** needs

How do we do this?

Review: Problem Definition

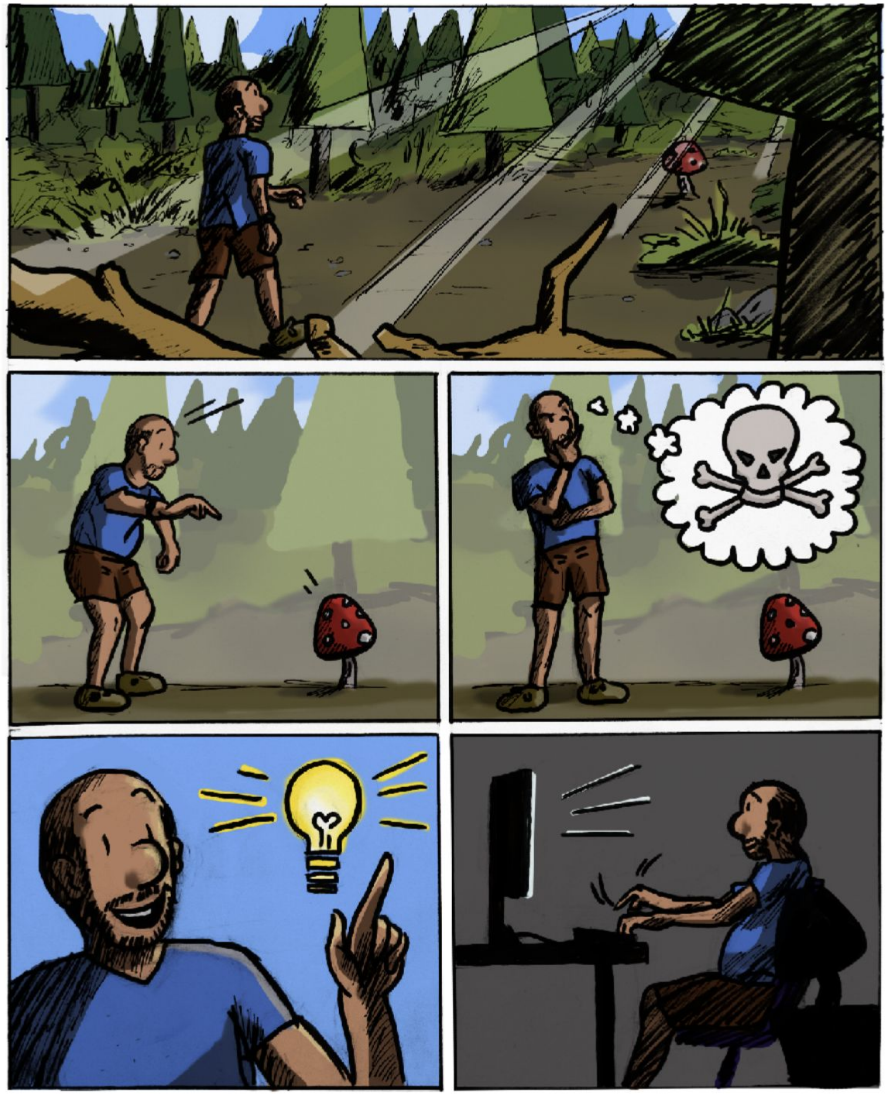
Pavlos like to go to the forest to do mushroom picking. It is a fun activity and also rewarding as some mushrooms are edible. The problem is in the forest where Pavlos goes to pick mushrooms there are many varieties of poisonous mushrooms. Some of the mushrooms are obvious but there are some which he requires help in identification.

Review: Proposed Solution

Pavlos will have his phone with him when he is in the forest. What if he could just take a picture of the mushrooms and and app could tell him what type of mushroom it is and whether it is poisonous or not

Review: Proposed Solution

- Pavlos likes to go to the forest for mushroom picking
- Some mushrooms can be poisonous
- Help build an app to identify mushroom type and if poisonous or not



Credit: Nikolas Protopapas

Review: Project Scope

Proof Of Concept (POC)

- Scrap mushroom data
- Verify images
- Experiment on some baseline models
- Verify new unseen mushrooms are predicted by the model(s)
- Visualize model activations to analyse what the model is seeing

Prototype

- **Create a mockup of screens to see how the app could look like**
- **Deploy one model to Fast API to service model predictions as an API**

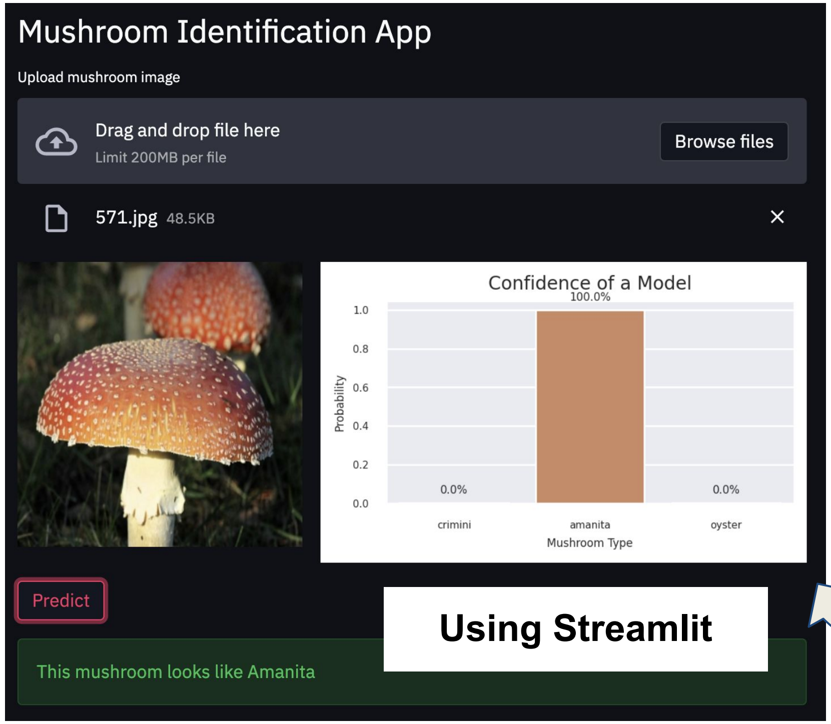
Minimum Viable Product (MVP)

- **Create App to identify Mushrooms**
- **API Server for uploading images and predicting using best model**

Review: Project Scope

Proof Of Concept (POC)

- Scrap mushroom data
- Verify images



Mushroom Identification App

Upload mushroom image

Drag and drop file here

Limit 200MB per file

Browse files

571.jpg 48.5KB

This mushroom looks like Amanita

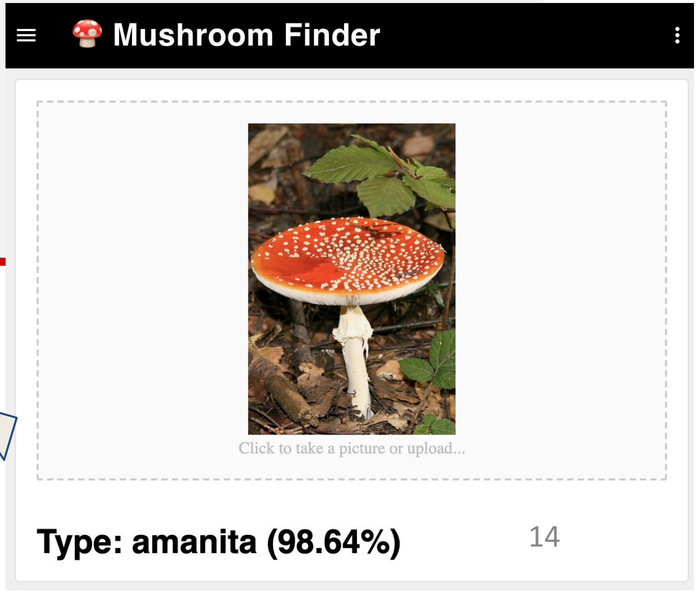
Using Streamlit

Prototype

- **Create a mockup of screens to see how the app could look like**
- **Deploy one model to Fast API to service model predictions as an API**

Minimum Viable Product (MVP)

- **Create App to identify Mushrooms**
- **API Server for uploading images and predicting using best model**

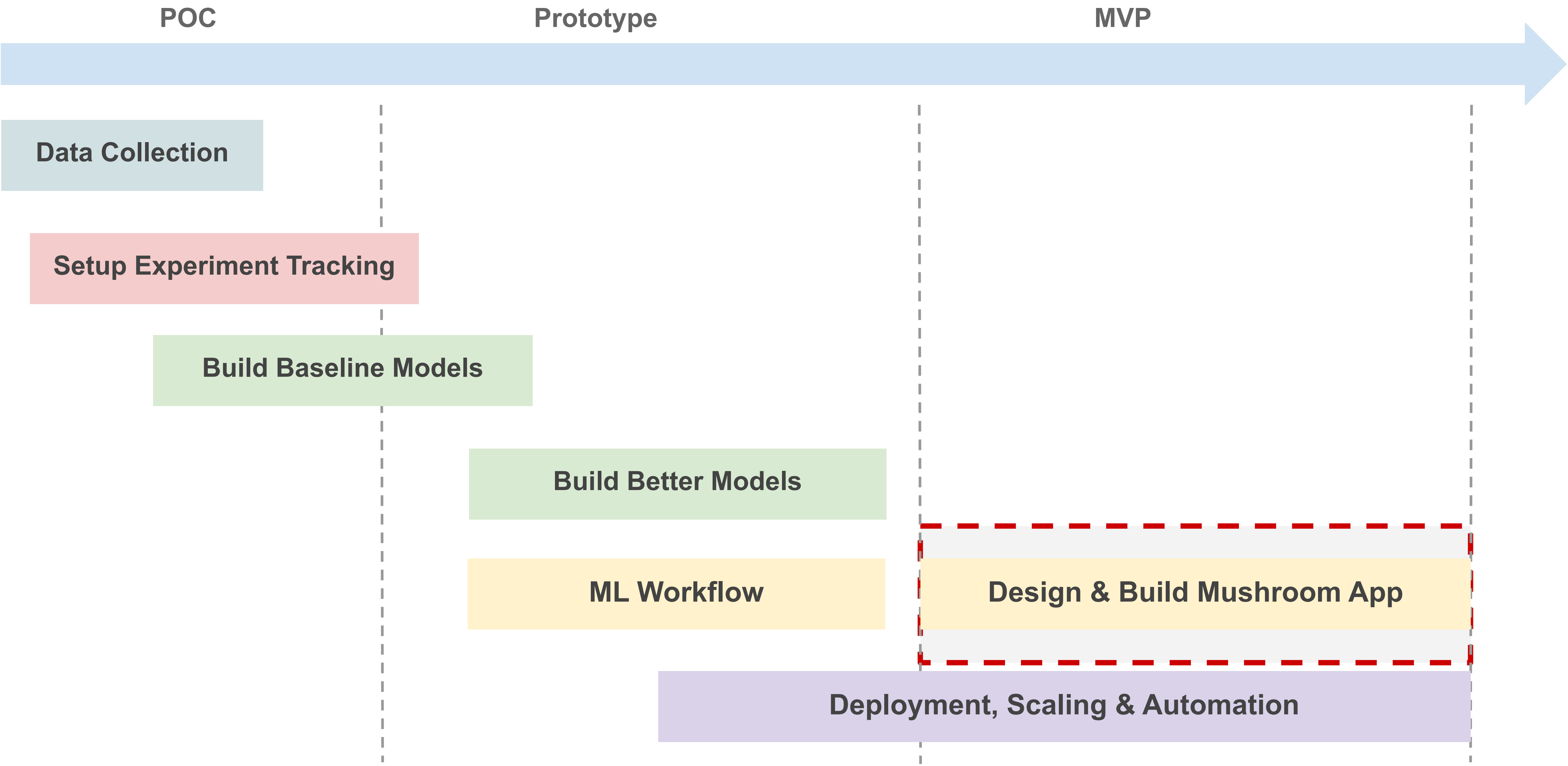


Mushroom Finder

Click to take a picture or upload...

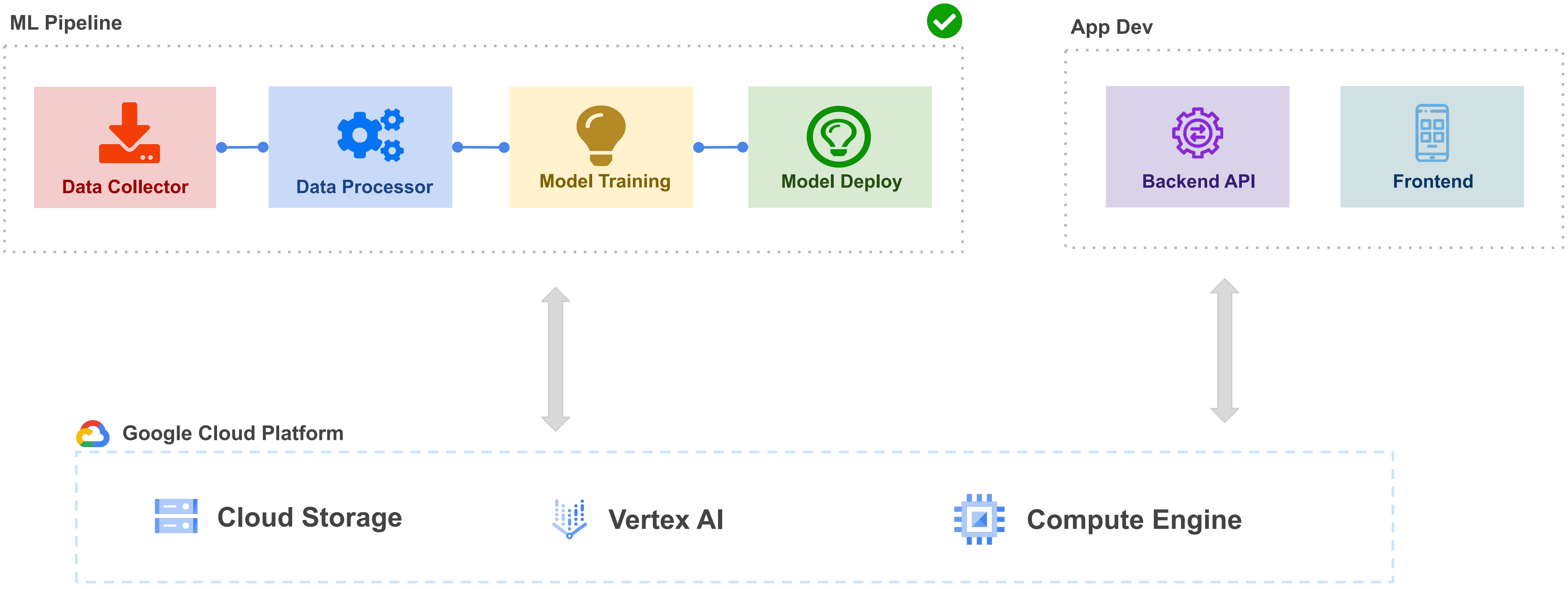
Type: amanita (98.64%)

Review: Mushroom App Status



POC
Prototype
MVP
Data Collection
Setup Experiment Tracking
Build Baseline Models
Build Better Models
ML Workflow
Design & Build Mushroom App
Deployment, Scaling & Automation

Mushroom App Development



Outline

1. Recap
2. Motivation
3. **App Design**
4. Screenflow & Wireframes
5. Solution Architecture
6. Technical Architecture
7. Setup & Code Organization

App Design

- In a regular software app you have code and data.
- In an **AI App**, in addition you have models to perform tasks
- We will follow a structured approach to design and develop an AI App
- The design will consist of the following components:
  - Screenflow & Wireframes
  - Solution Architecture
  - Technical Architecture

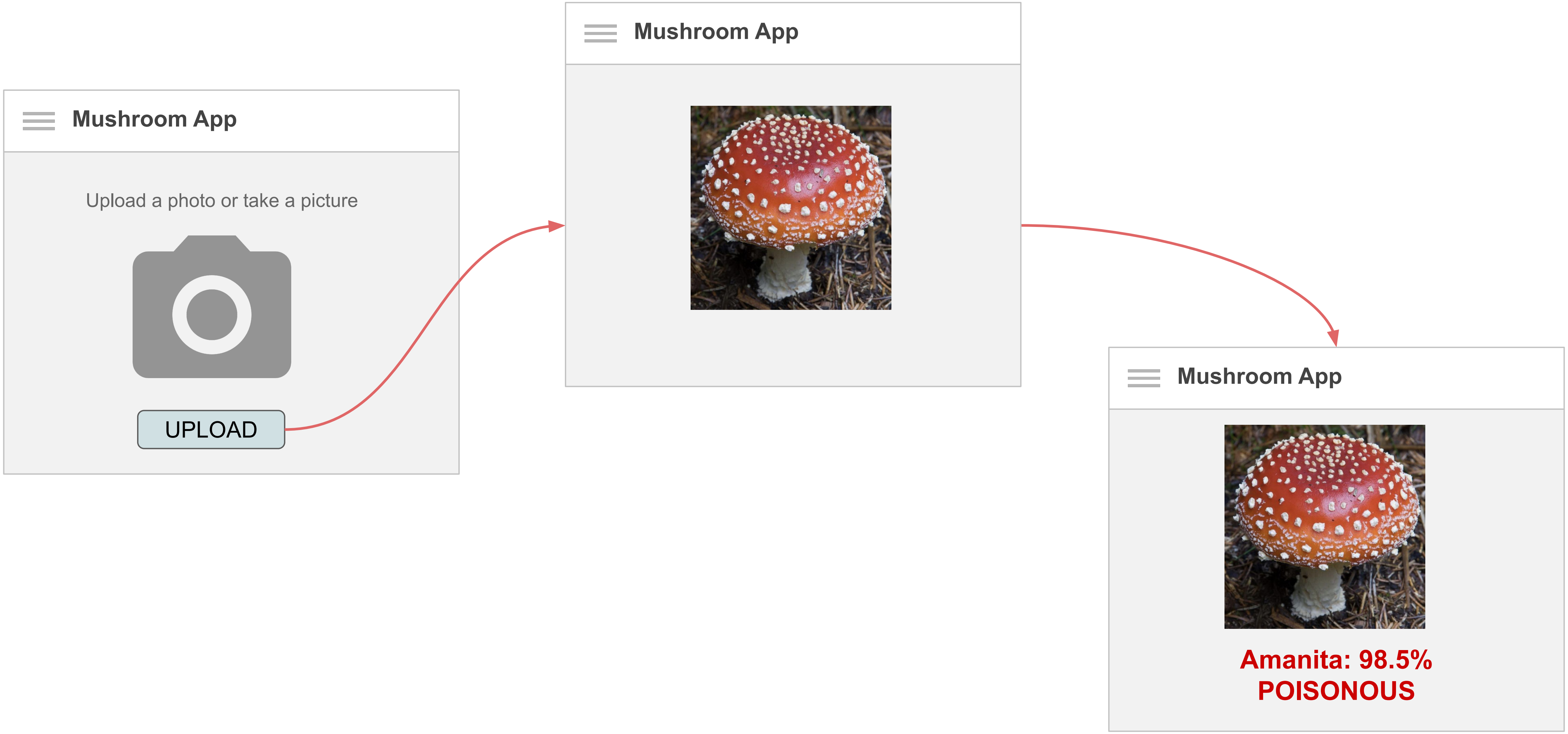
Outline

1. Recap
2. Motivation
3. App Design
4. **Screenflow & Wireframes**
5. Solution Architecture
6. Technical Architecture
7. Setup & Code Organization

Screenflow & Wireframes

Start with brainstorming ideas on whiteboard/paper

Screenflow & Wireframes



Mushroom App

Upload a photo or take a picture

UPLOAD

Mushroom App

Mushroom App

Amanita: 98.5%
POISONOUS

Outline

1. Recap
2. Motivation
3. App Design
4. Screenflow & Wireframes
5. **Solution Architecture**
6. Technical Architecture
7. Setup & Code Organization

Solution Architecture

- Helps to identify the building **blocks** in an App
- Start by asking how will your **App** address the **Problem Statement**
- Identifying the following:
    - The **Process** being performed by the user
    - The code **Execution** blocks required to fulfil the **Process**
    - The **State** required during the life cycle of the App

Solution Architecture

Process (People)

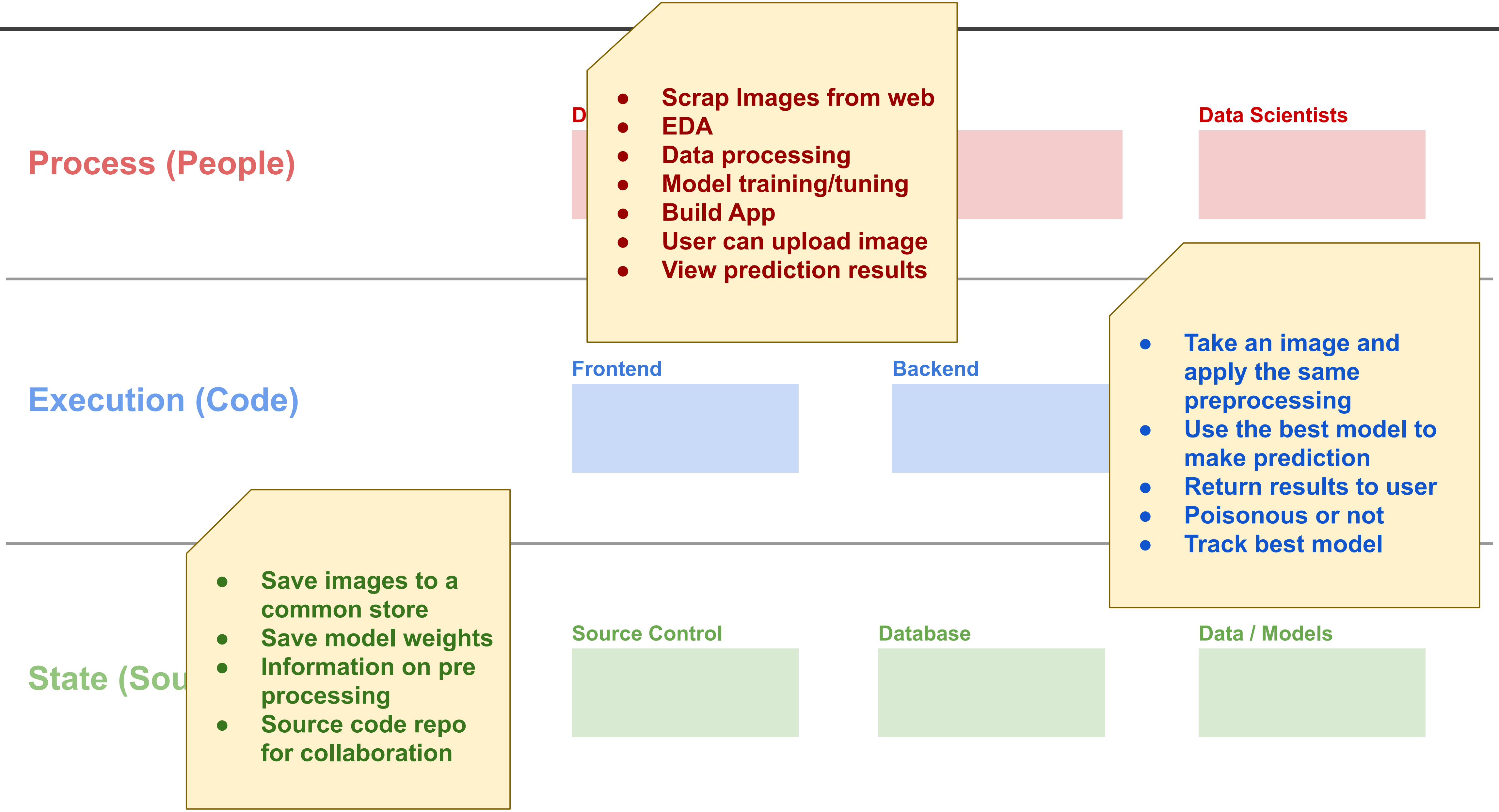
Execution (Code)

State (Source, Data, Models)

Solution Architecture **AI App**

Process (People)	**Developers**	**Users**	**Data Scientists**
Execution (Code)	**Frontend**	**Backend**	**ML Tasks**
State (Source, Data, Models)	**Source Control**	**Database**	**Data / Models**

Solution Architecture



Process (People)

D

- Scrap Images from web
- EDA
- Data processing
- Model training/tuning
- Build App
- User can upload image
- View prediction results

Data Scientists

Execution (Code)

Frontend

Backend

- Take an image and apply the same preprocessing
- Use the best model to make prediction
- Return results to user
- Poisonous or not
- Track best model

State (Sou

- Save images to a common store
- Save model weights
- Information on pre processing
- Source code repo for collaboration

Source Control

Database

Data / Models

Solution Architecture

Process

Execution

State

Solution Architecture

Process

Develop App

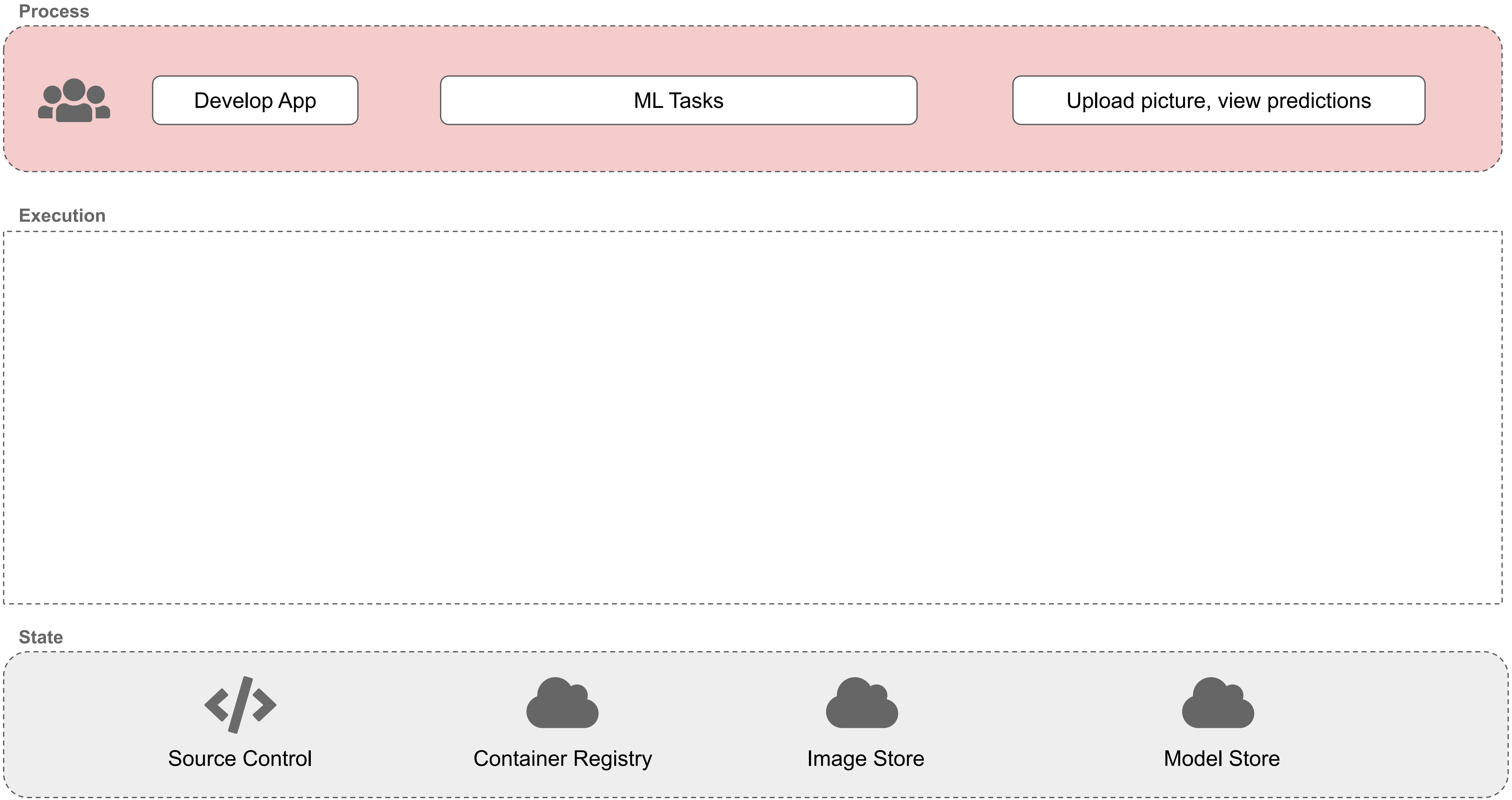
ML Tasks

Upload picture, view predictions

Execution

State

Solution Architecture



Process

Develop App

ML Tasks

Upload picture, view predictions

Execution

State

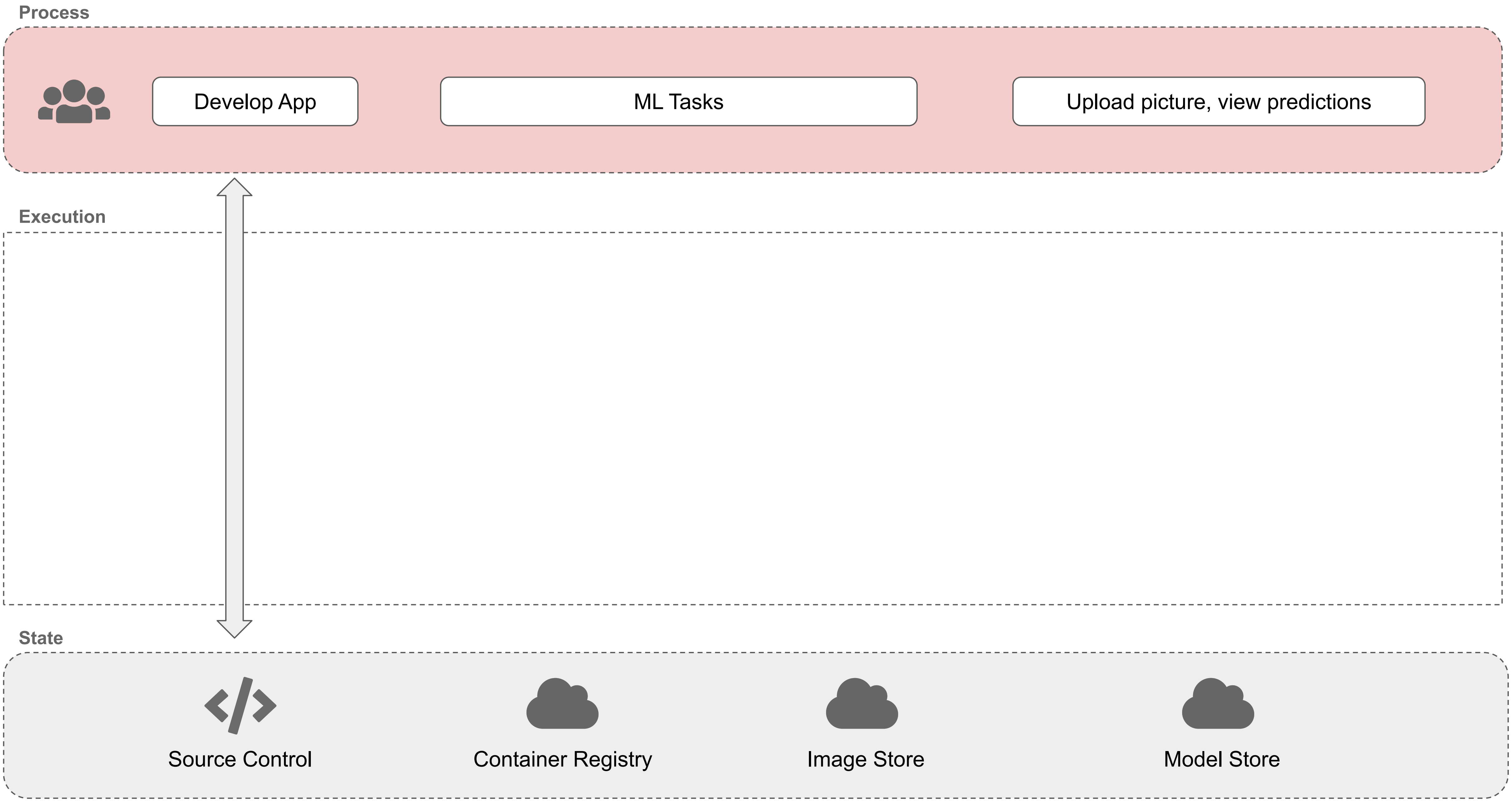
Source Control

Container Registry

Image Store

Model Store

Solution Architecture



Process

Develop App

ML Tasks

Upload picture, view predictions

Execution

State

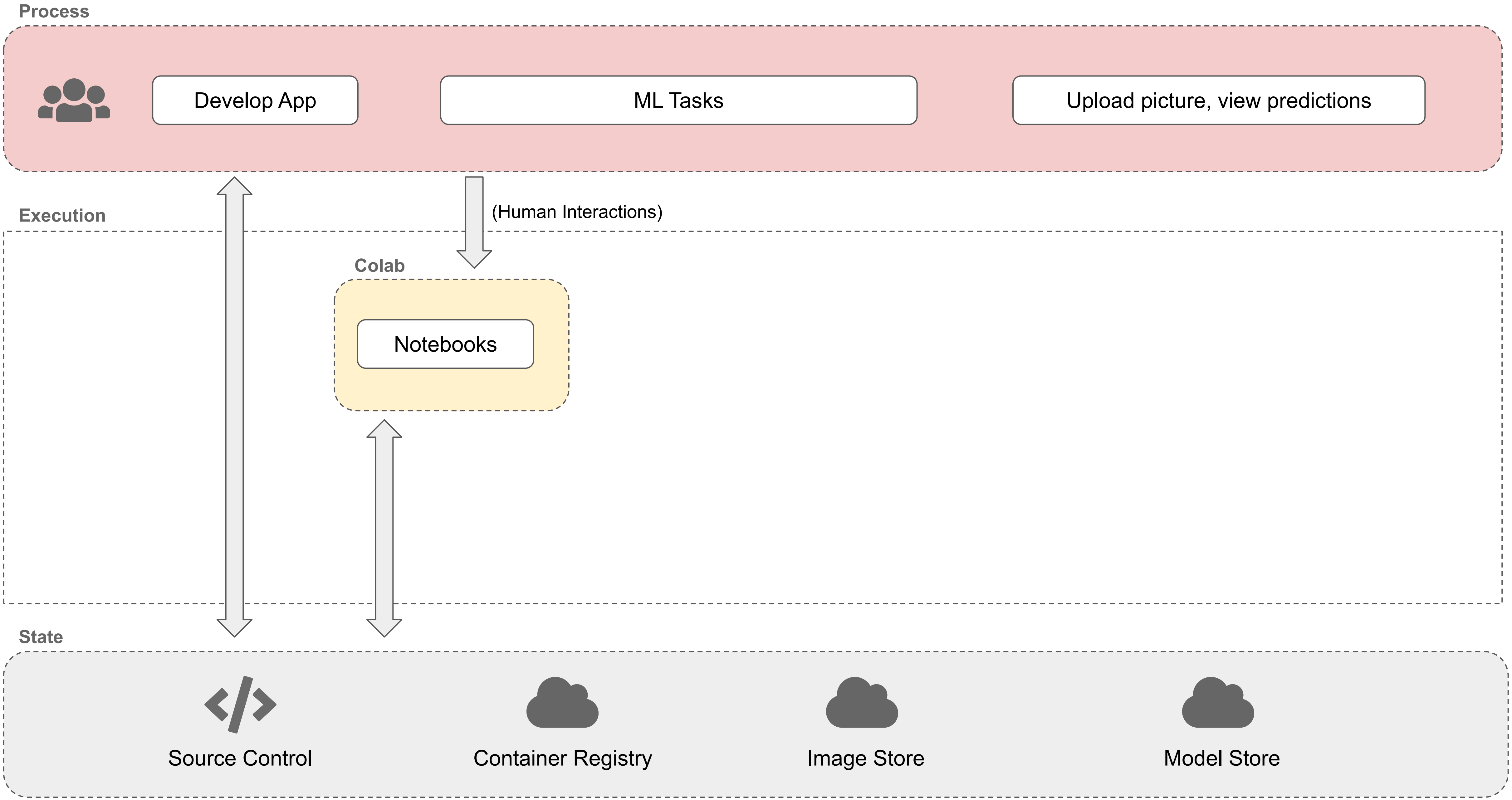
Source Control

Container Registry

Image Store

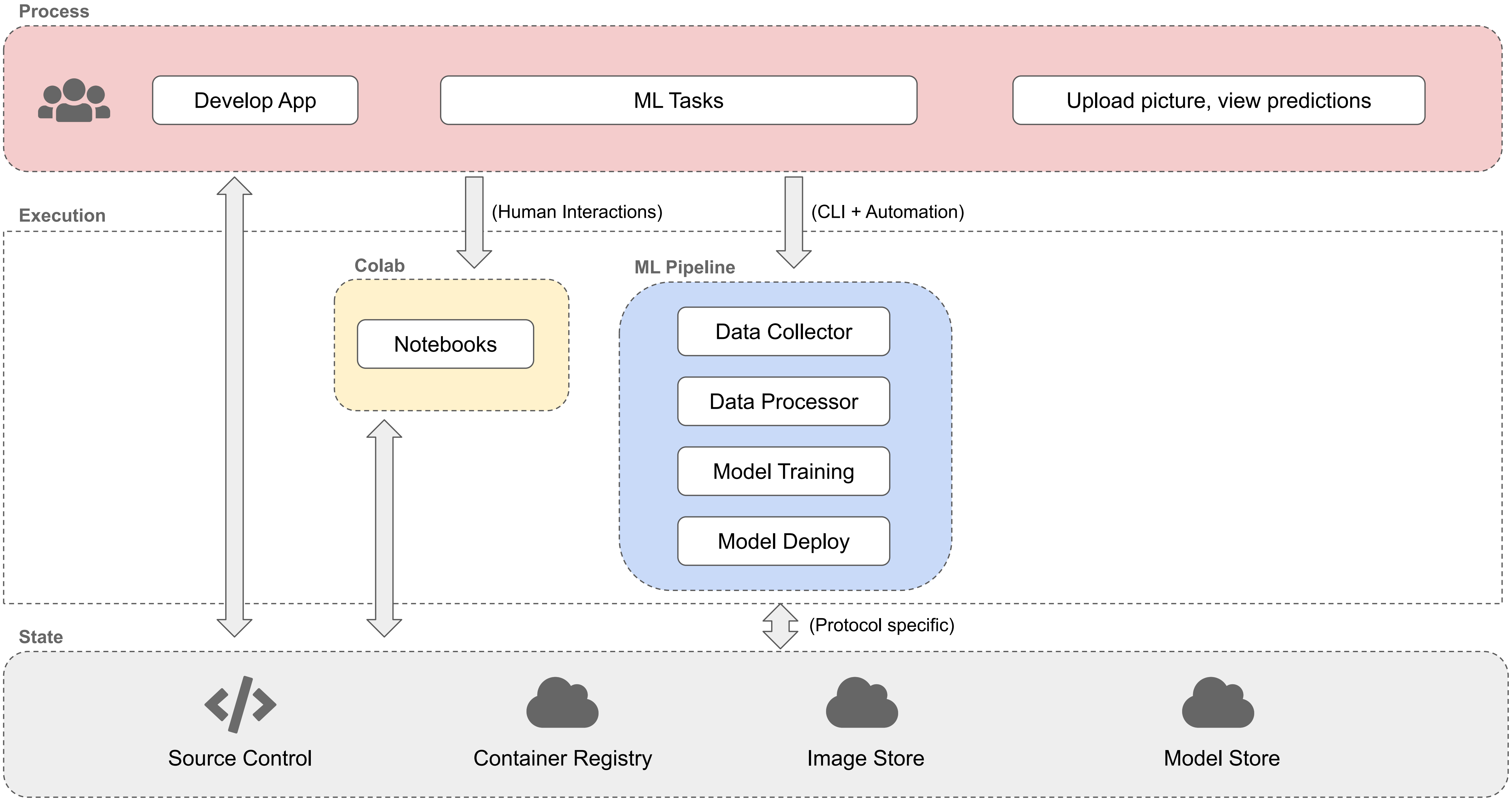
Model Store

Solution Architecture

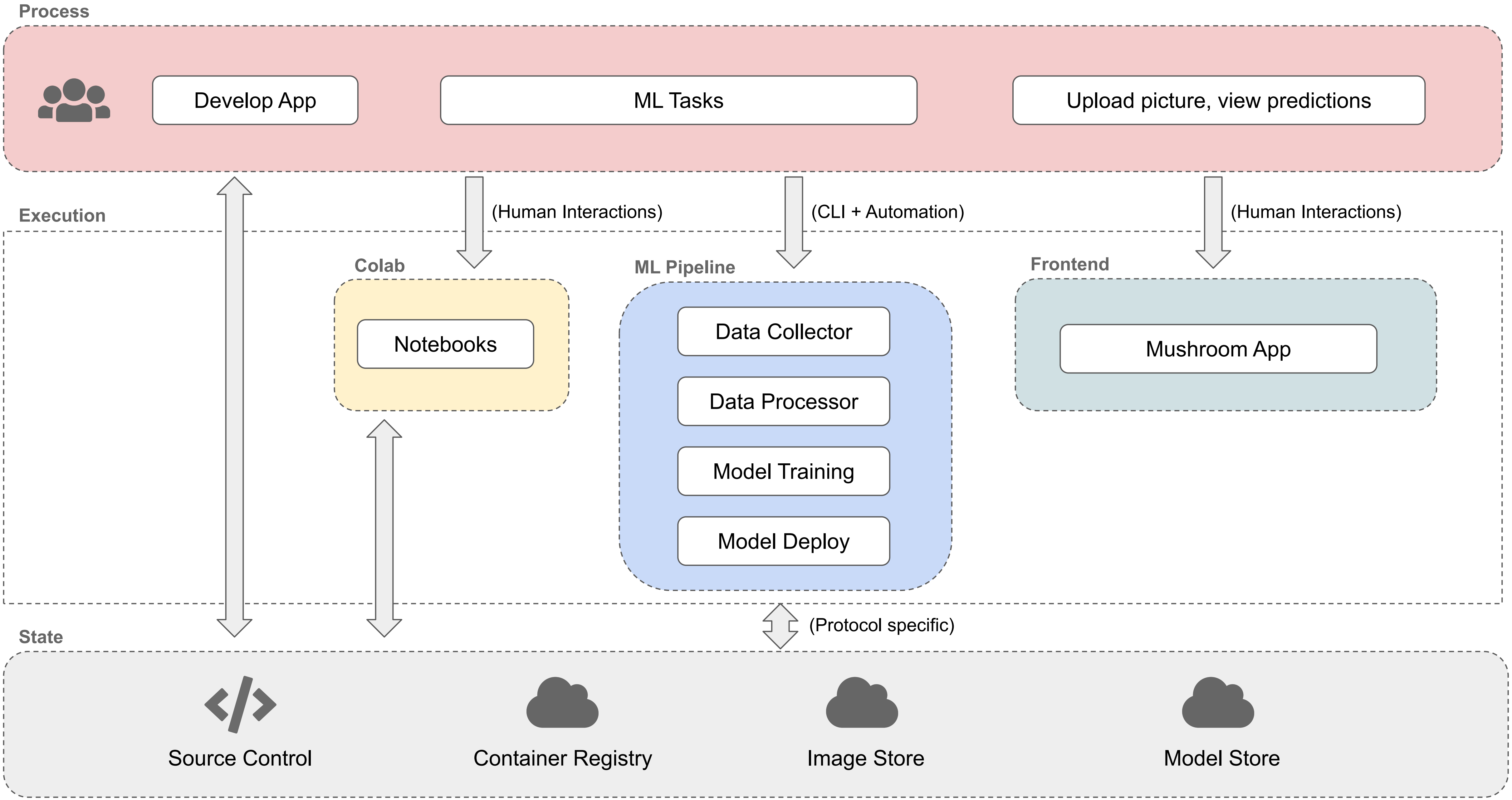


]

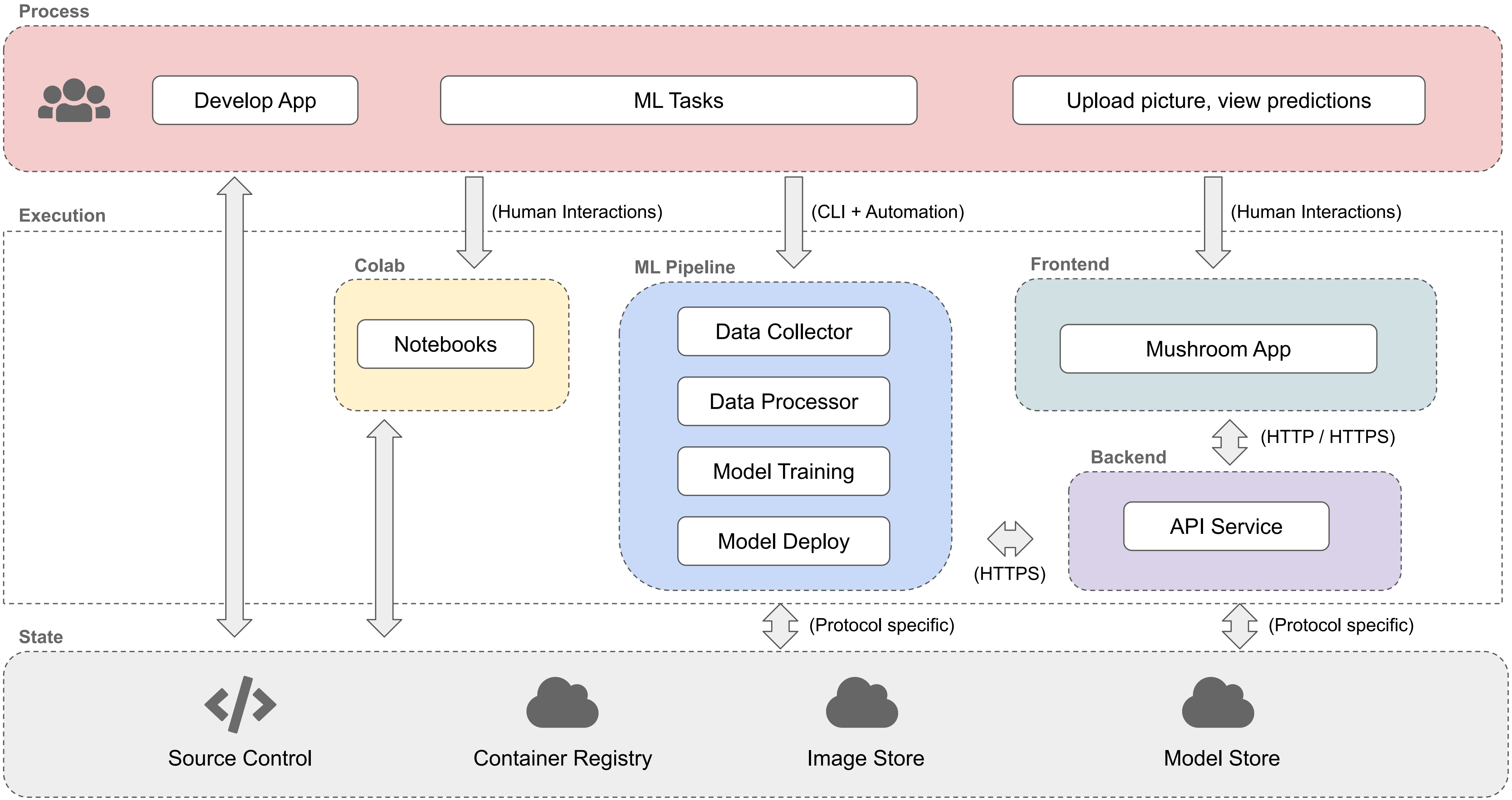
Solution Architecture



Solution Architecture



Solution Architecture



Solution Architecture Summary

- **Process**
  - Data Scientists perform ML Tasks
  - Developers build App
  - Users can upload pictures and view predictions
- **Colab**
  - Web based hosted notebook solution from Google to experiment ML task
- **ML Pipeline**
  - Containerized ML components
  - Helps to automate and run ML tasks

- **Frontend**
  - User friendly single page app with capabilities to upload an image and view prediction results
- **Backend**
  - API server to expose python functions to frontend
- **State**
  - Source control to store/version code
  - Container registry for docker images
  - Image store for data
  - Models and model artifacts store

Tutorial: Building Solution Architecture

Steps to build a **Solution Architecture**

- You will work with your project group
- Go to
  https://docs.google.com/presentation/d/15pNPFBn5U5RcSXOAxrmbtD_HFJahObLSWeyYI51-qtc/edit?usp=sharing .
- Duplicate Slides 2,3 to the end.
- Put your group name in the slides.
- Identify Process, Execution, State for your project.
- For later: Complete Solution Architecture slide for your project.

Outline

1. Recap
2. Motivation
3. App Design
4. Screenflow & Wireframes
5. Solution Architecture
6. **Technical Architecture**
7. Setup & Code Organization

Technical Architecture

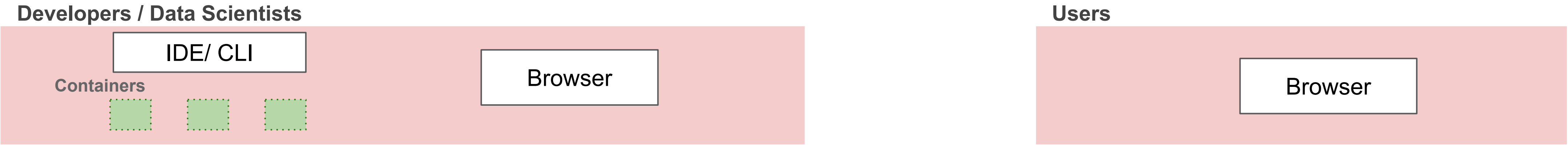
- Helps design and develop an **AI App**
- High level view from **development** to **deployment**
- Illustrates **interactions** between components/**containers**
- **Blueprint** of the system
  - Helps team members understand the big picture
  - Helps onboarding new team members

Building a Technical Architecture

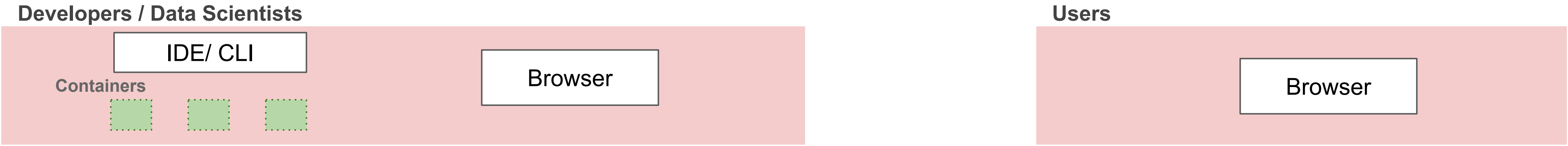
Developers / Data Scientists

Users

Building a Technical Architecture



Building a Technical Architecture



Developers:

- Use IDE (VSCode), CLI to build app
- All development is containerized

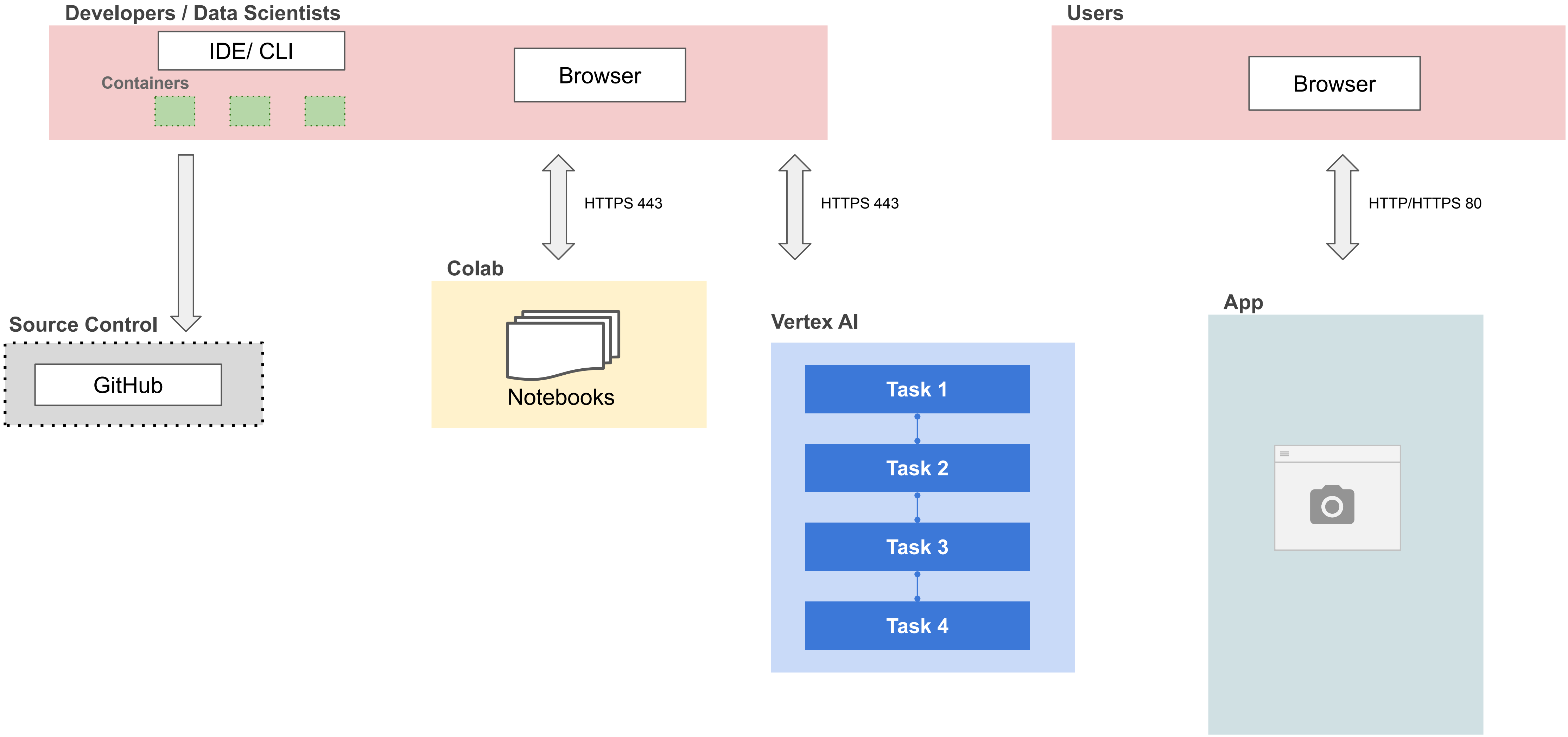
Data Scientists:

- Use Colab/JupyterHub
- EDA on notebooks
- Data & Model experimentation on notebooks
- Use IDE (VSCode), CLI to build ML Tasks
- All development is containerized

Users:

- Access the App using a browser
- Upload images and view prediction results

Building a Technical Architecture



Developers / Data Scientists

IDE/ CLI

Containers

Browser

Users

Browser

HTTPS 443

HTTPS 443

HTTP/HTTPS 80

Source Control

GitHub

Colab

Notebooks

Vertex AI

Task 1

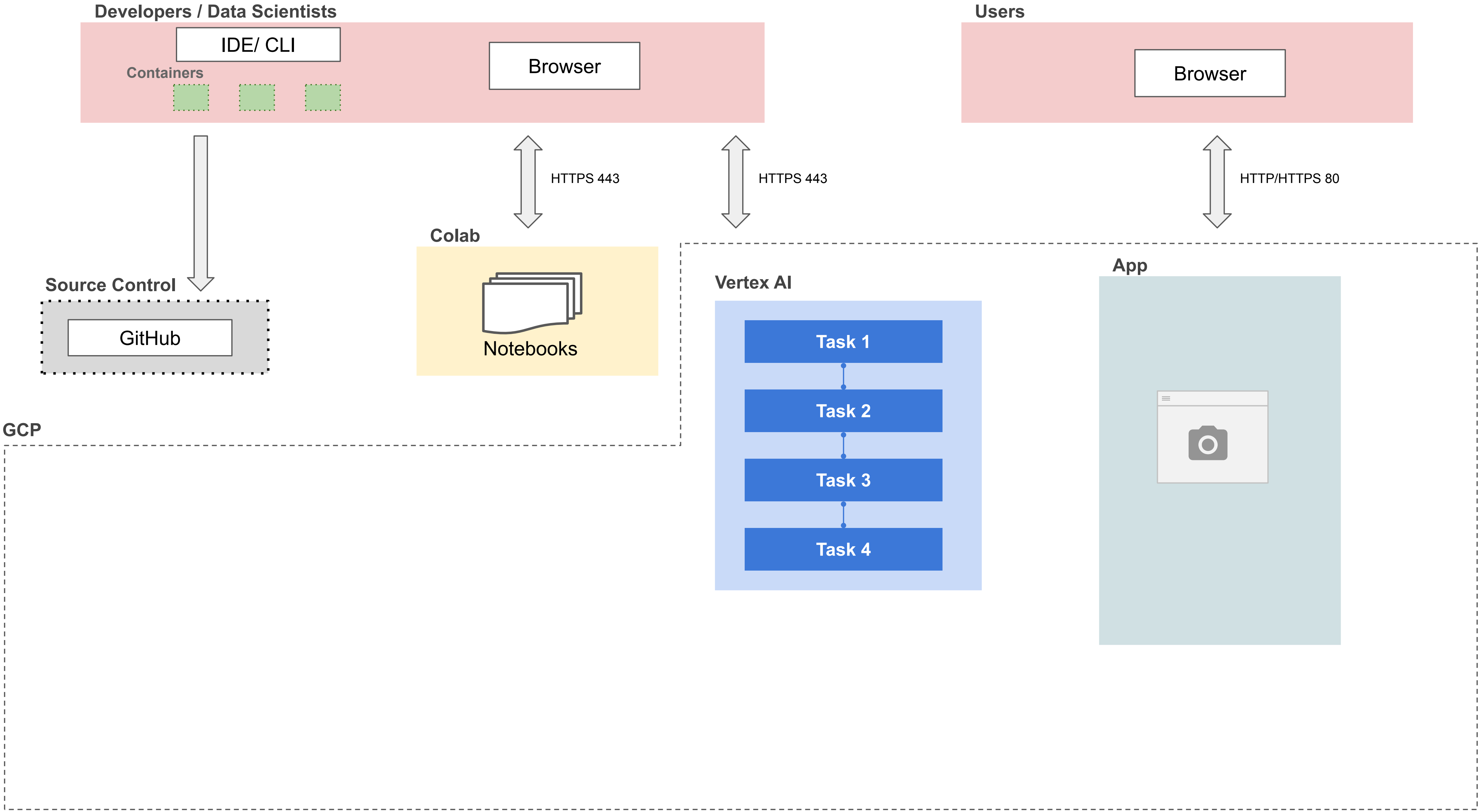
Task 2

Task 3

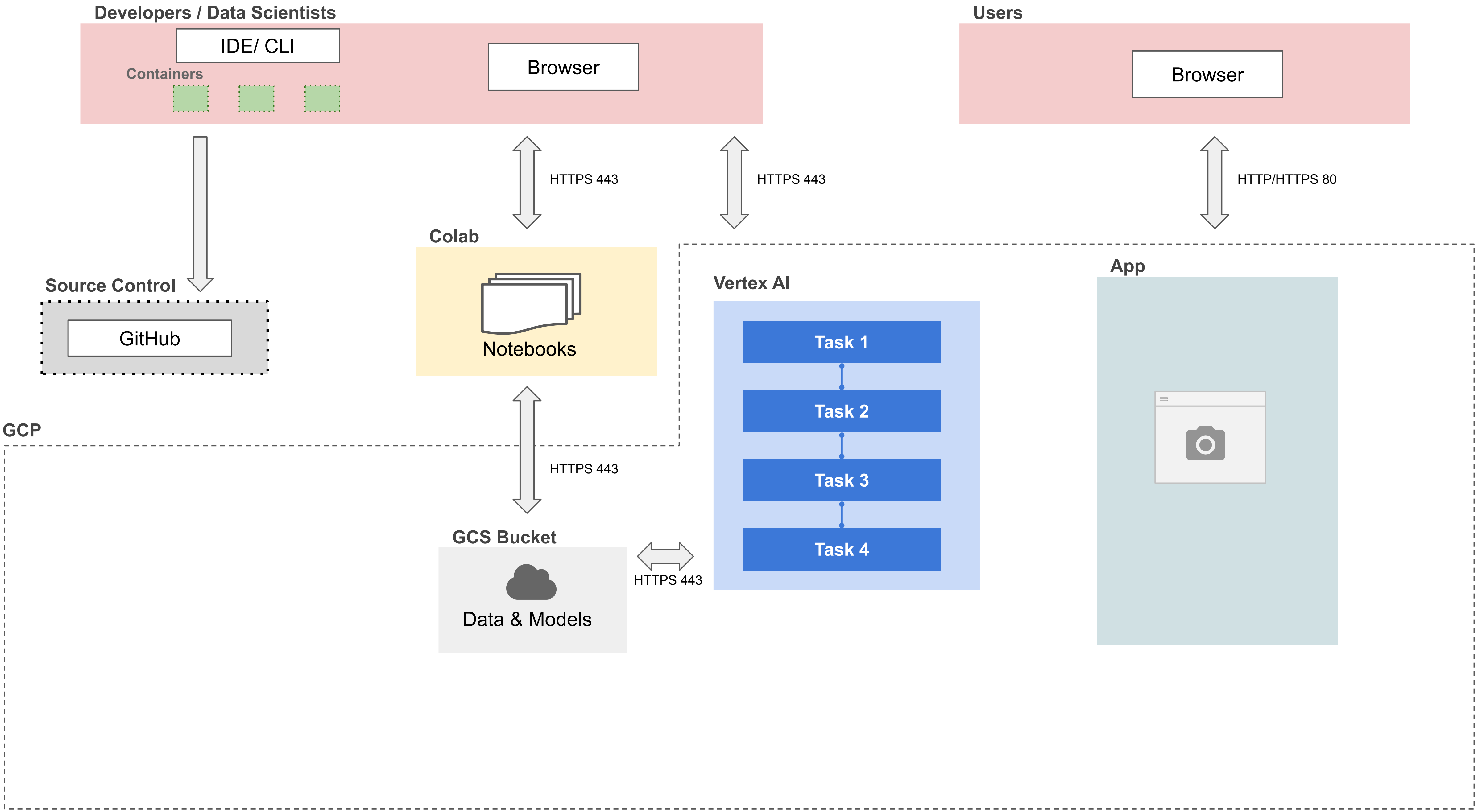
Task 4

App

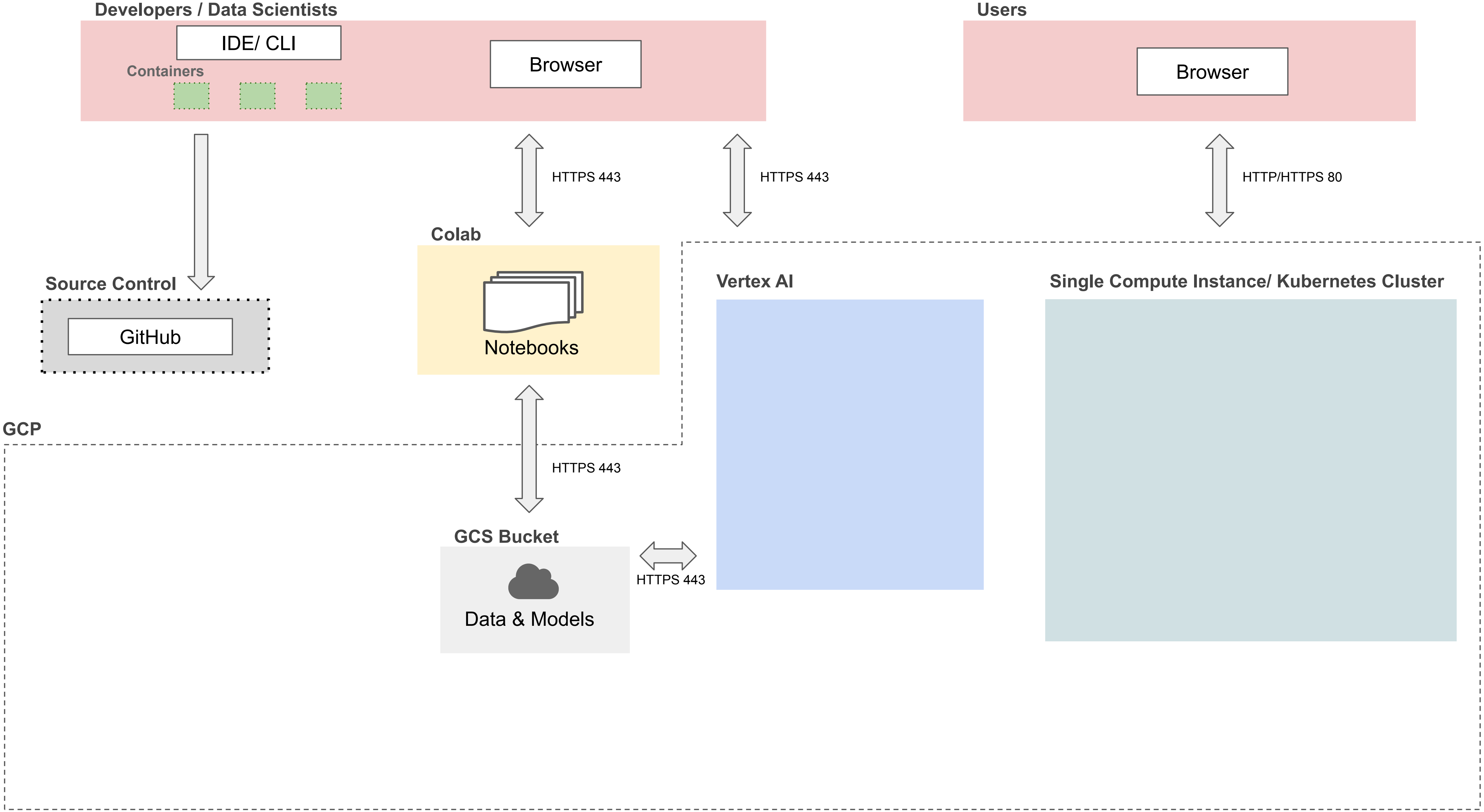
Building a Technical Architecture



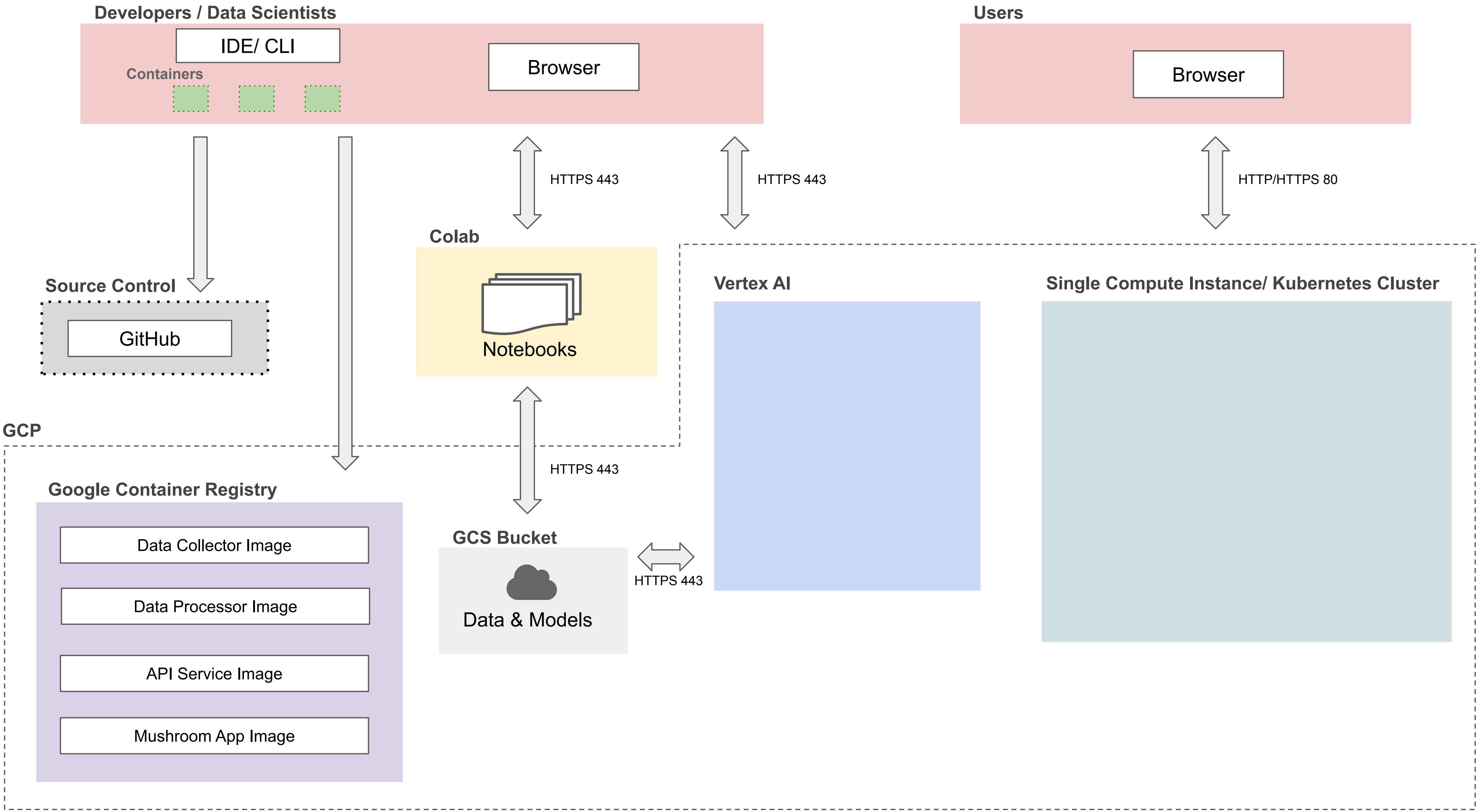
Building a Technical Architecture



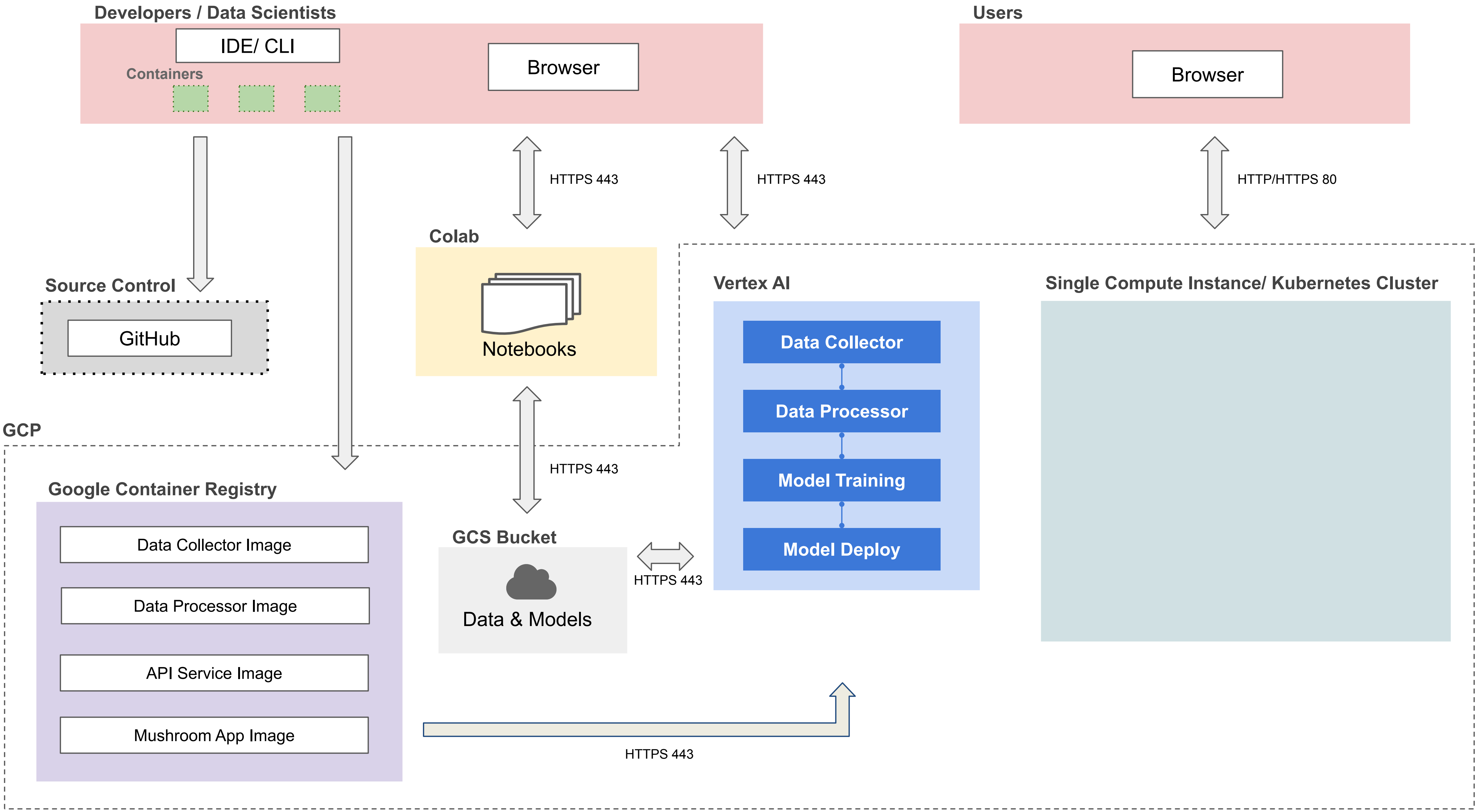
Building a Technical Architecture



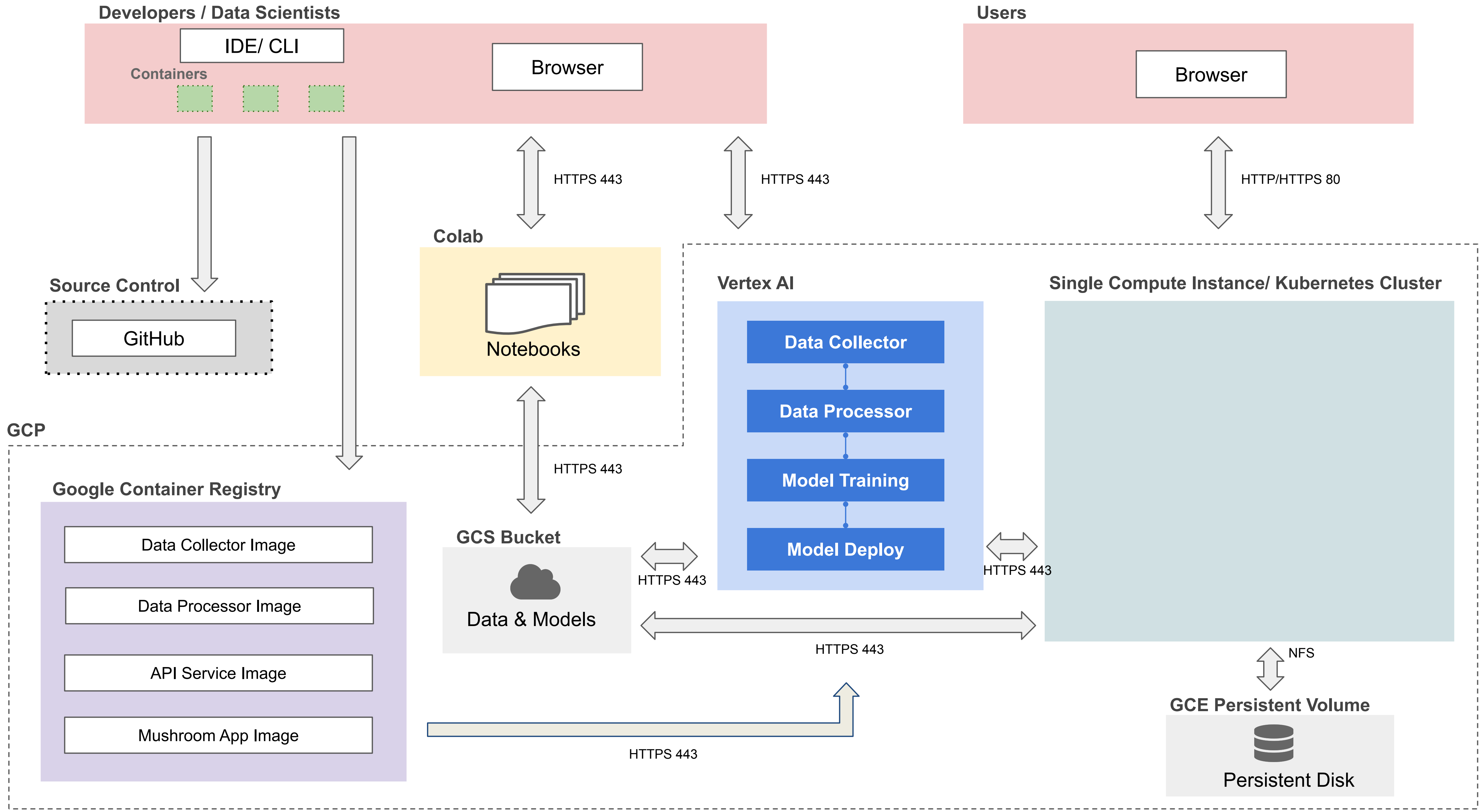
Building a Technical Architecture



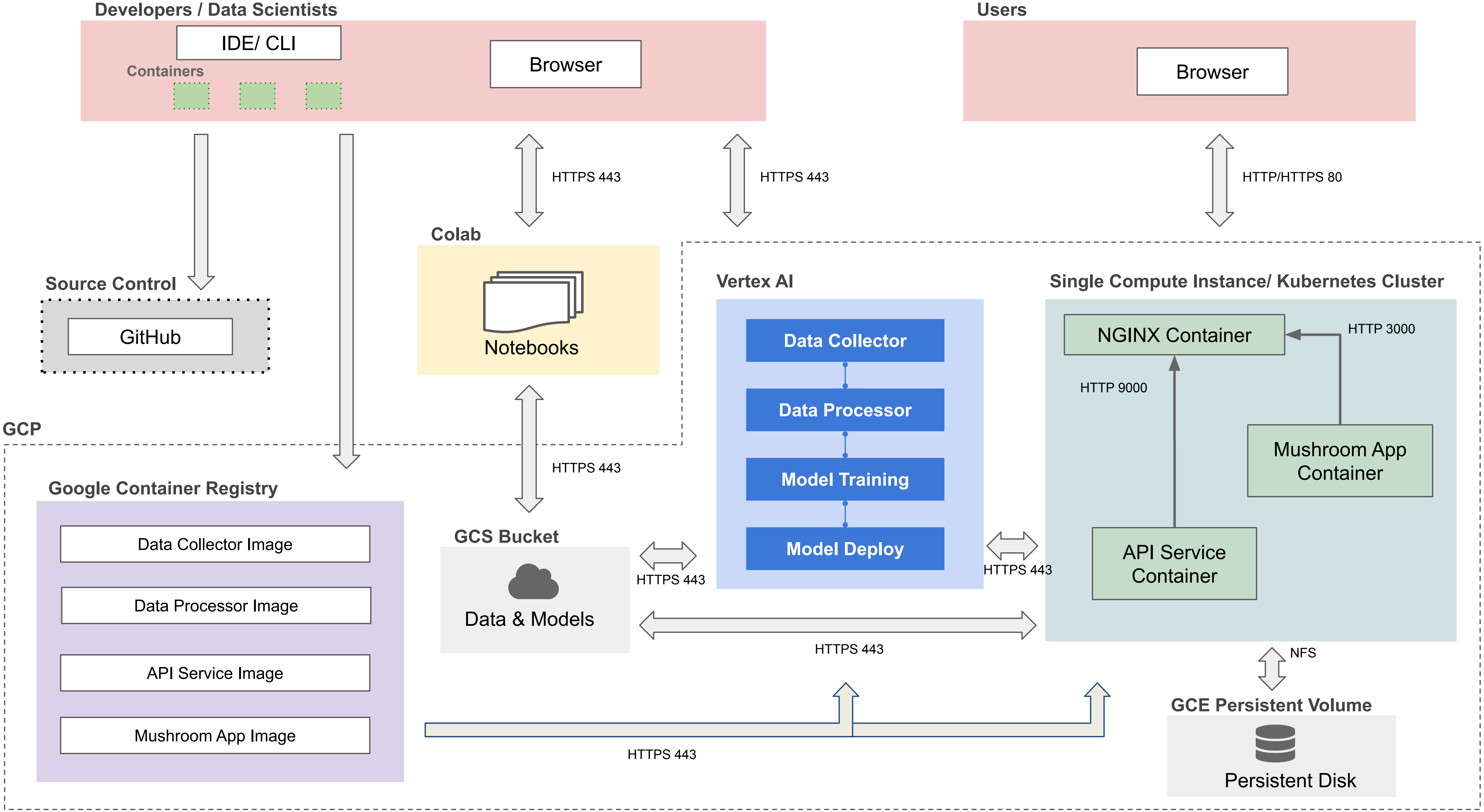
Building a Technical Architecture



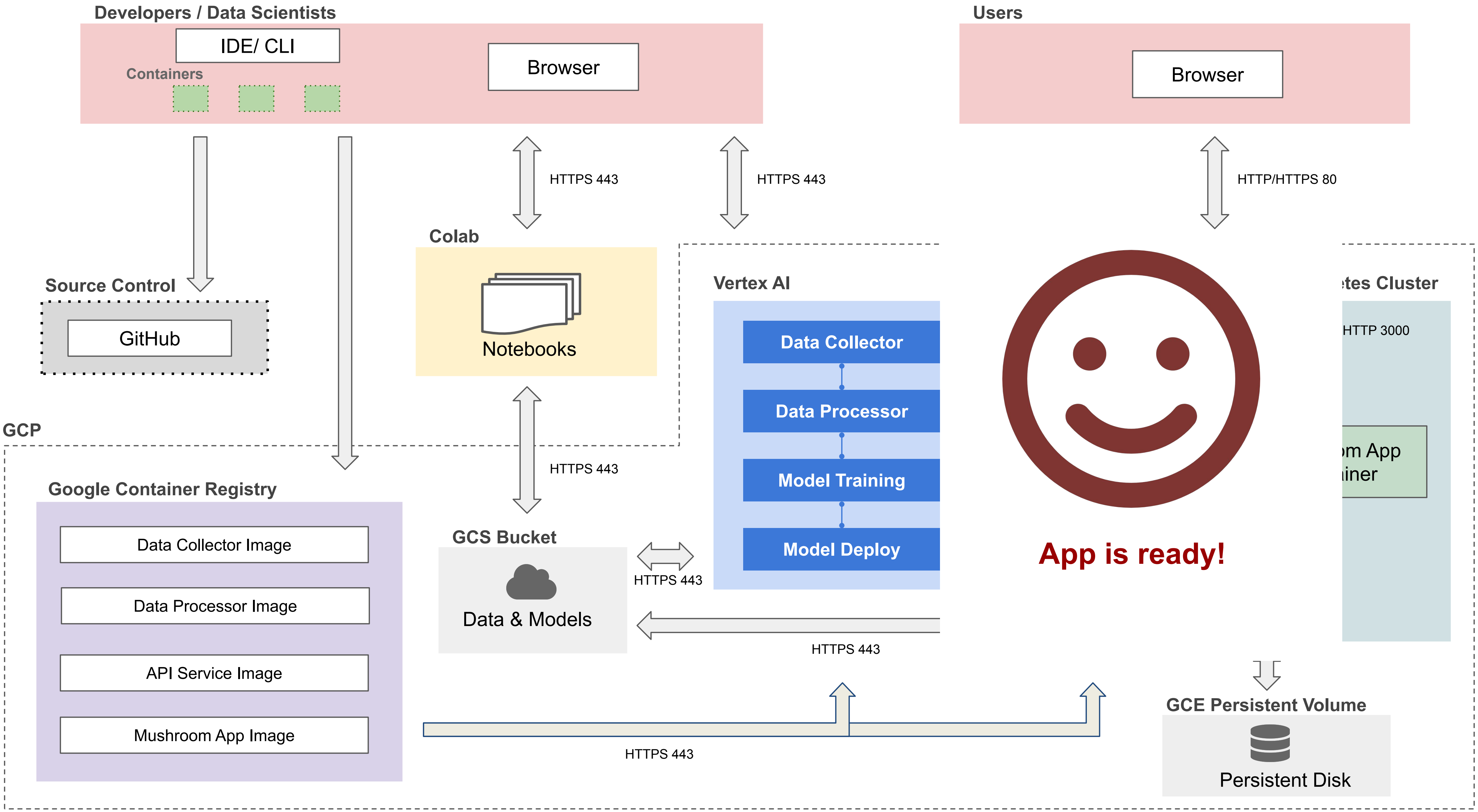
Building a Technical Architecture



Technical Architecture



Technical Architecture



Developers / Data Scientists

IDE/ CLI

Containers

Browser

Users

Browser

HTTPS 443

HTTPS 443

HTTP/HTTPS 80

Colab

Notebooks

Source Control

GitHub

Vertex AI

Data Collector

Data Processor

Model Training

Model Deploy

tes Cluster

HTTP 3000

m App
iner

App is ready!

GCP

HTTPS 443

Google Container Registry

Data Collector Image

Data Processor Image

API Service Image

Mushroom App Image

GCS Bucket

Data & Models

HTTPS 443

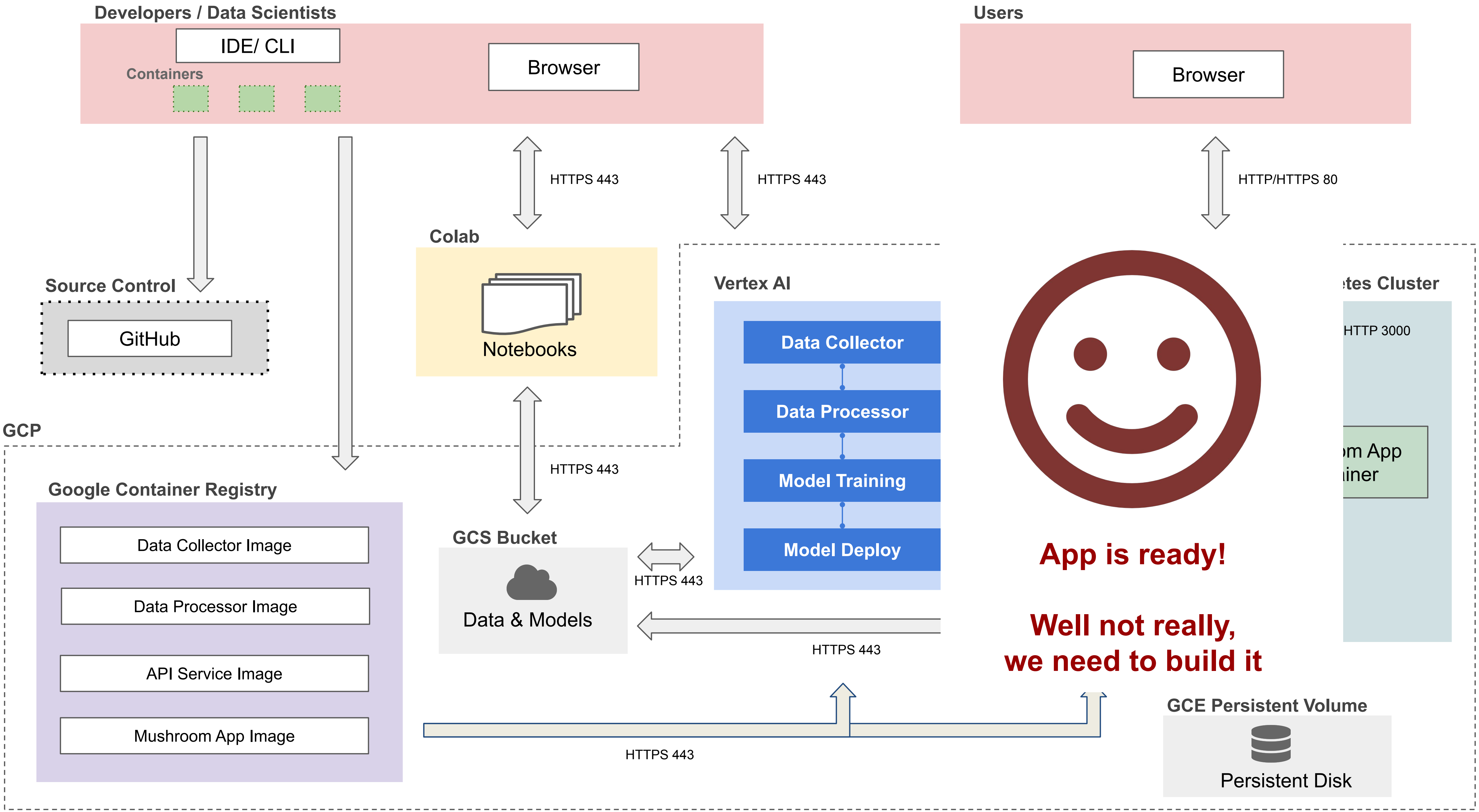
HTTPS 443

HTTPS 443

GCE Persistent Volume

Persistent Disk

Technical Architecture



App is ready!

Well not really,
we need to build it

Technical Architecture Summary

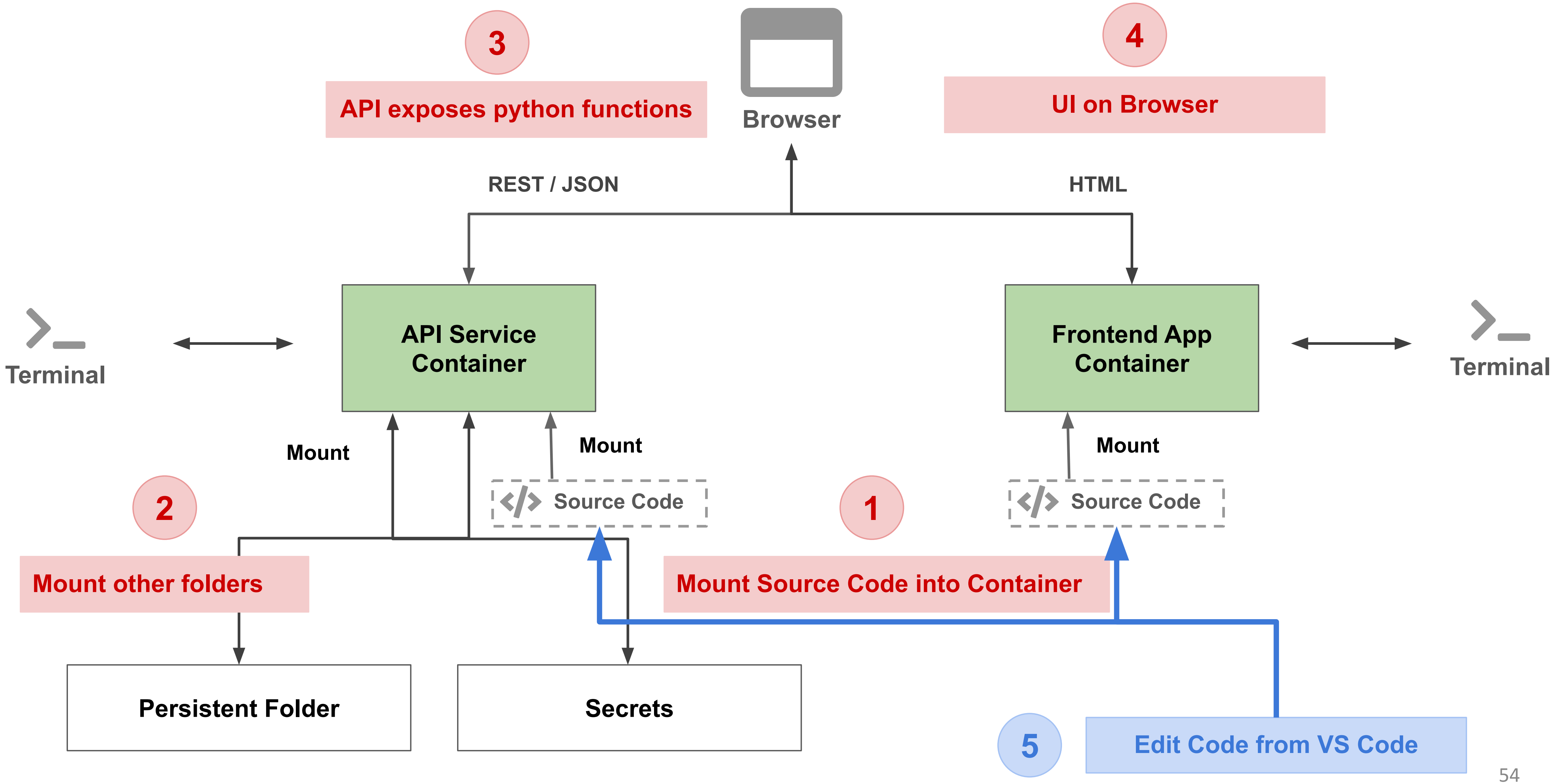
- **Source Control**
  - GitHub
- **Google Cloud Platform (GCP)**
  - GCP for deployment
- **Google Container Registry**
  - GCR to host all the container images
- **GCS Buckets**
  - Storage buckets for models and model artifacts
  - Data(Image) store

- **Vertex AI**
  - Serverless ML Tasks
- **GCE Persistent Volume**
  - Any files that need to be persisted when container images are updated
- **Compute Instance**
  - Hosting single instance of all containers
- **Kubernetes Cluster**
  - Kubernetes cluster will be used to deploy a scalable version of the app on GCP

Outline

1. Recap
2. Motivation
3. App Design
4. Screenflow & Wireframes
5. Solution Architecture
6. Technical Architecture
7. **Setup & Code Organization**

Setup & Code Organization



Tutorial: Setup & Code Organization

Mushroom App - Setup & Code Organization

THANK YOU