Lecture 15: App Design, Setup & Code Organization

Advanced Practical Data Science, MLOps

AC215

Pavlos Protopapas
Institute for Applied Computational Science, Harvard

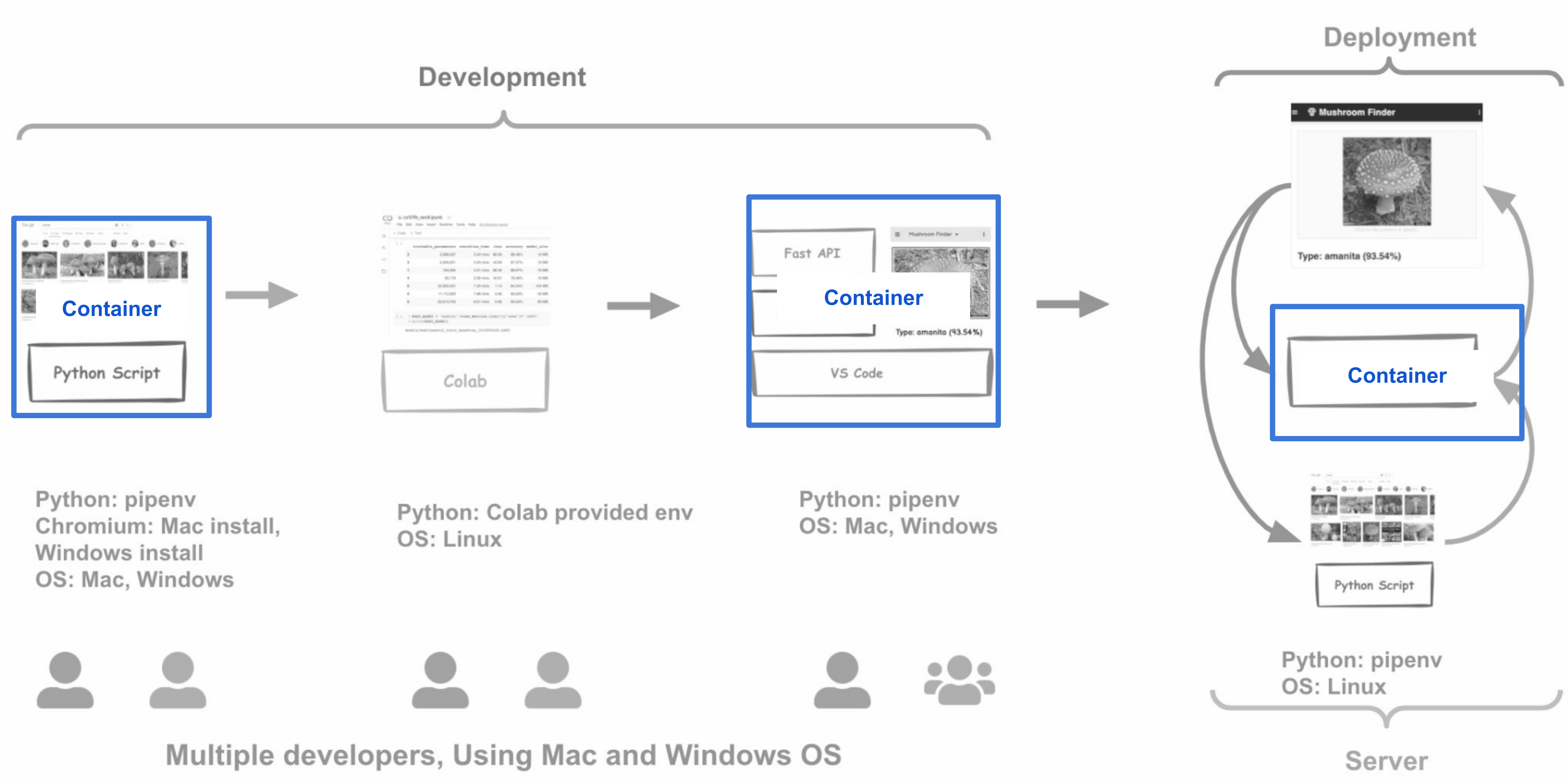
Outline

1. Recap
2. Motivation
3. App Design
4. Setup & Code Organization

Outline

1. **Recap**
2. Motivation
3. App Design
4. Setup & Code Organization

Recap: Isolate work into reusable containers?



Development

Deployment

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

Python: Colab provided env
OS: Linux

Python: pipenv
OS: Mac, Windows

Python: pipenv
OS: Linux

Multiple developers, Using Mac and Windows OS

Server

Outline

1. Recap
2. **Motivation**
3. App Design
4. Setup & Code Organization

Before you build your App

- You do **NOT** want to build your entire app in one container
- Start thinking of functionality that can be **isolated**
- Identify components that can be **containerized**

How do we do this?

Review: Problem Definition

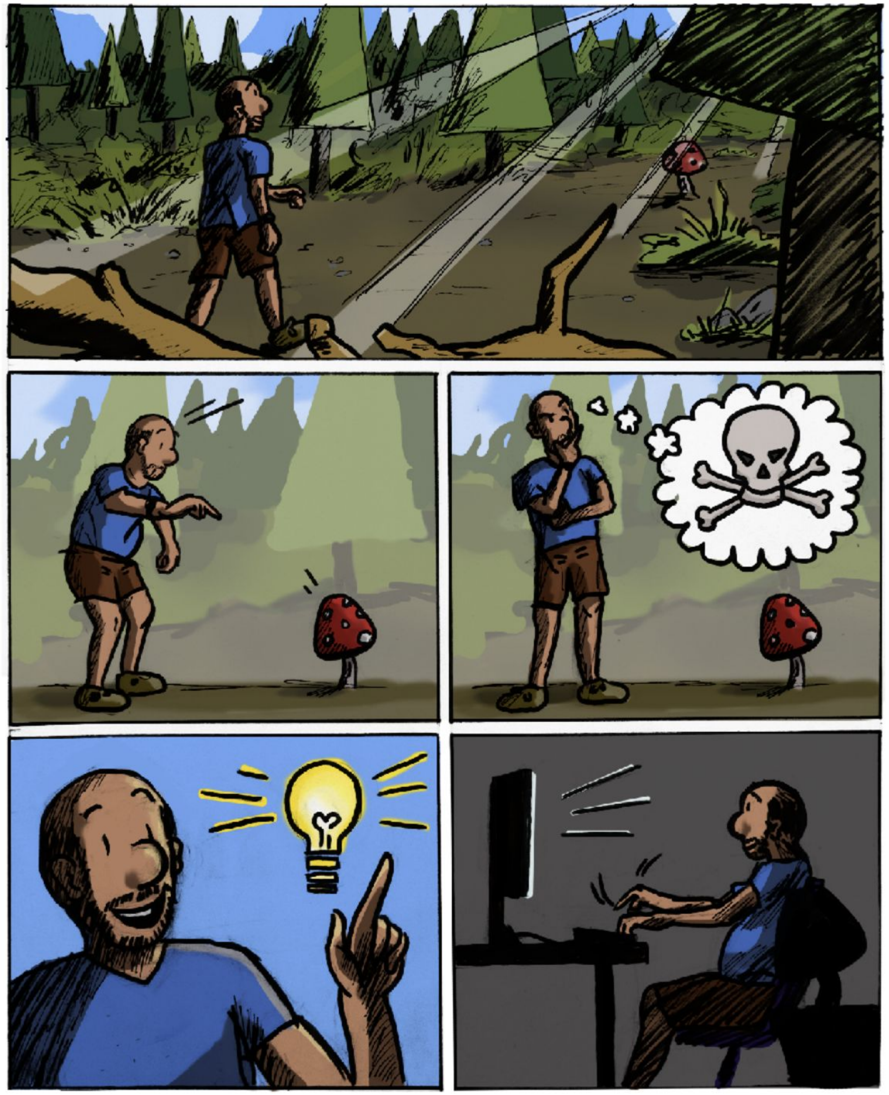
Pavlos like to go to the forest to do mushroom picking. It is a fun activity and also rewarding as some mushrooms are edible. The problem is in the forest where Pavlos goes to pick mushrooms there are many varieties of poisonous mushrooms. Some of the mushrooms are obvious but there are some which he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

Review: Proposed Solution

Pavlos will have is 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 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)

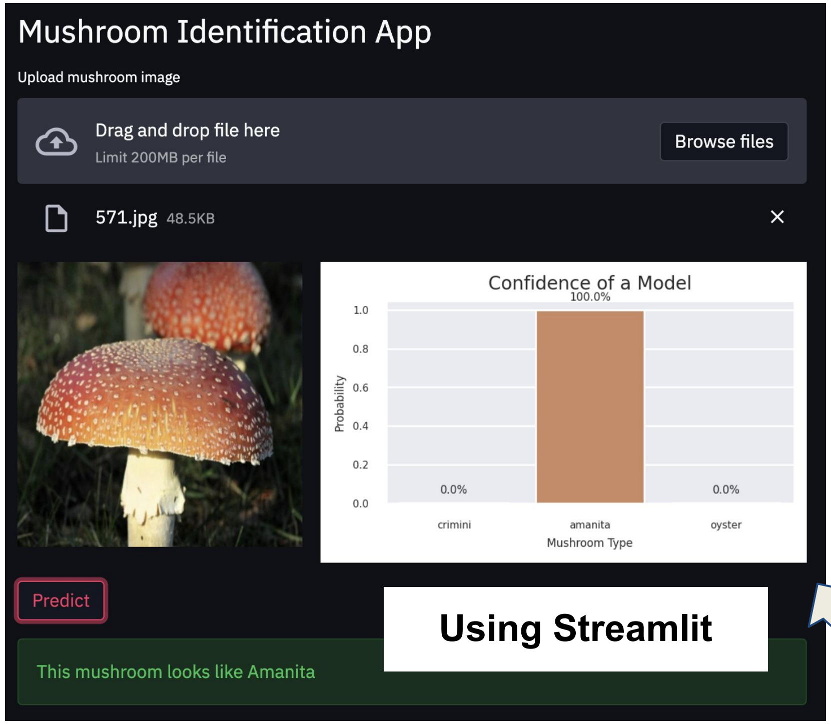
- Scrap mushroom data
- Verify images

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 Identification App

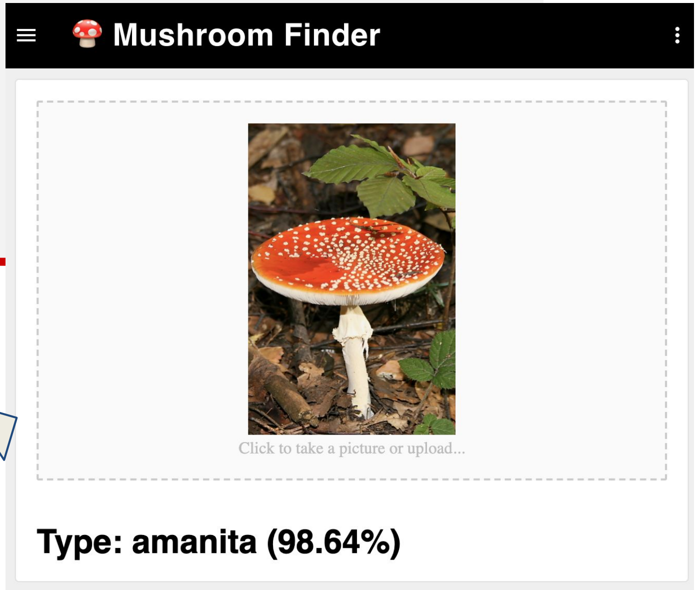
Upload mushroom image

Drag and drop file here
Limit 200MB per file

571.jpg 48.5KB

This mushroom looks like Amanita

Using Streamlit

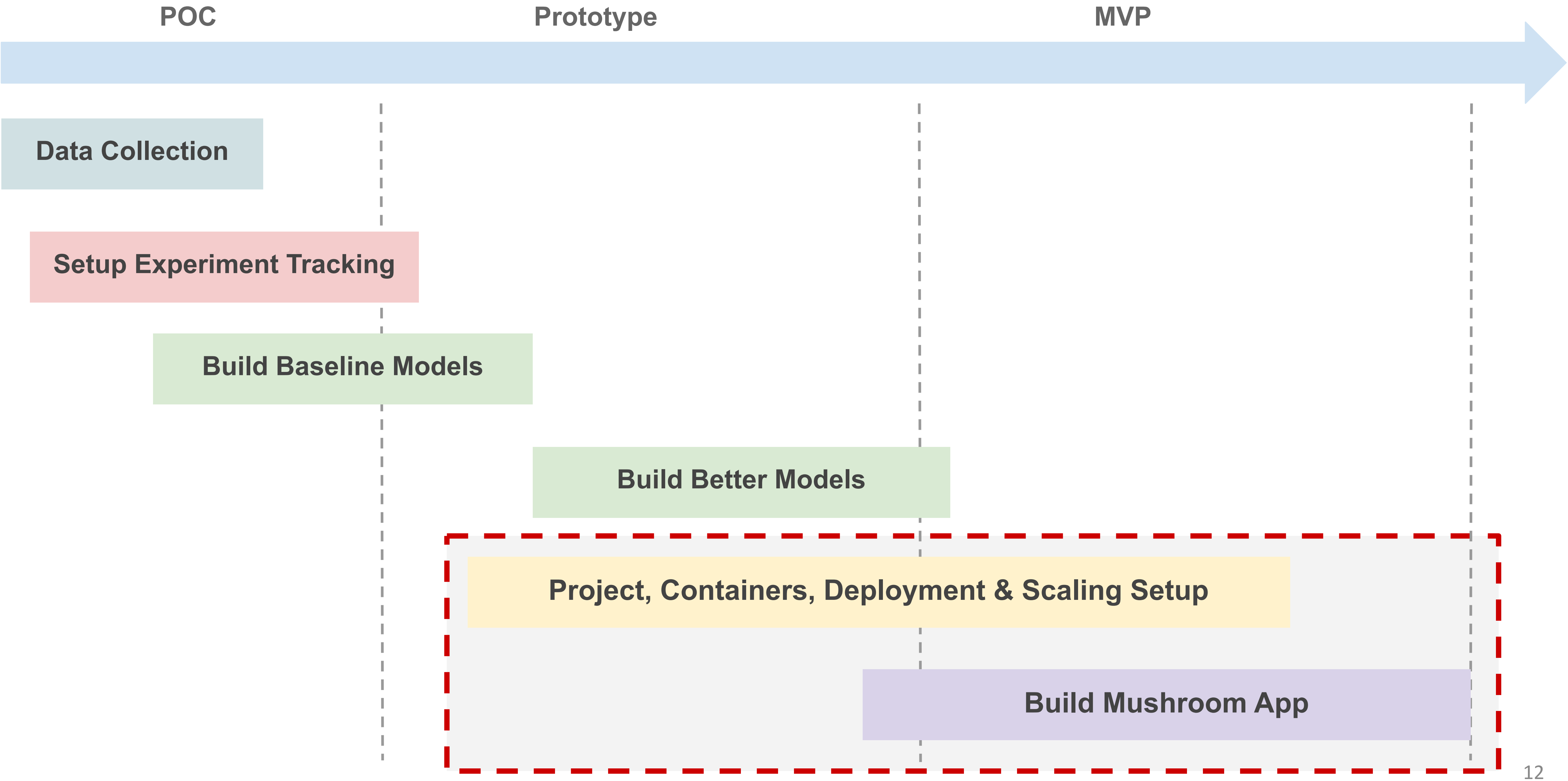


Mushroom Finder

Click to take a picture or upload...

Type: amanita (98.64%)

Review: Project Workflow



POC

Prototype

MVP

Data Collection

Setup Experiment Tracking

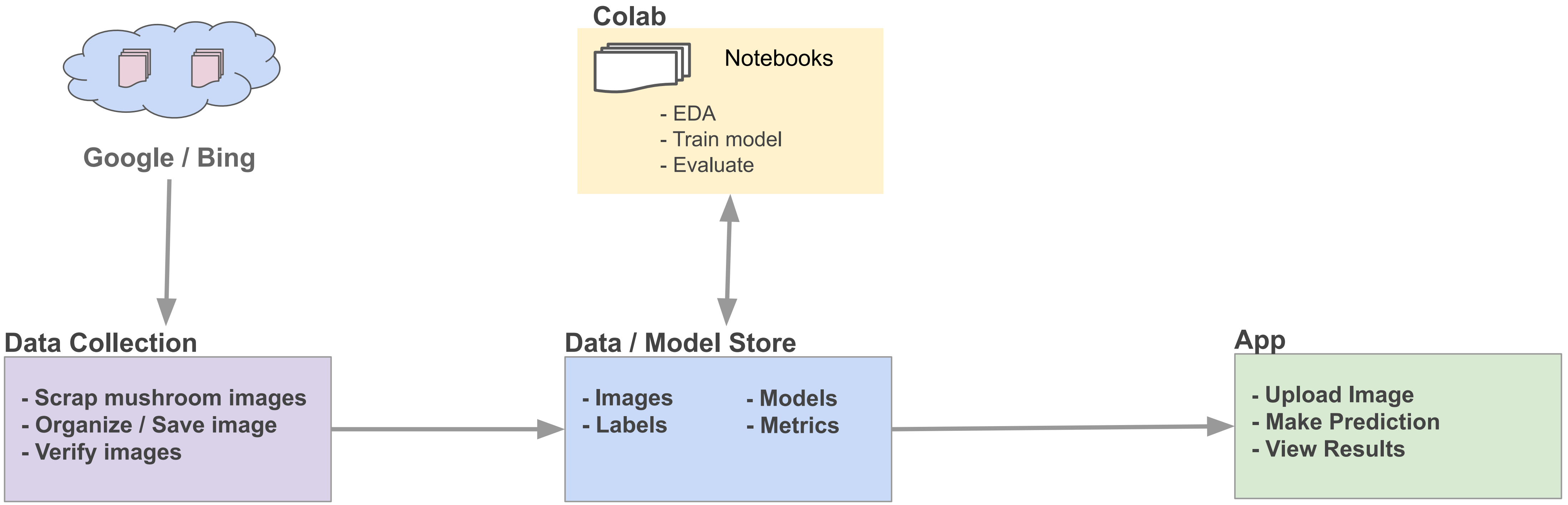
Build Baseline Models

Build Better Models

Project, Containers, Deployment & Scaling Setup

Build Mushroom App

Review: Process Flow



Google / Bing

Colab

Notebooks

- EDA
- Train model
- Evaluate

Data Collection

- Scrap mushroom images
- Organize / Save image
- Verify images

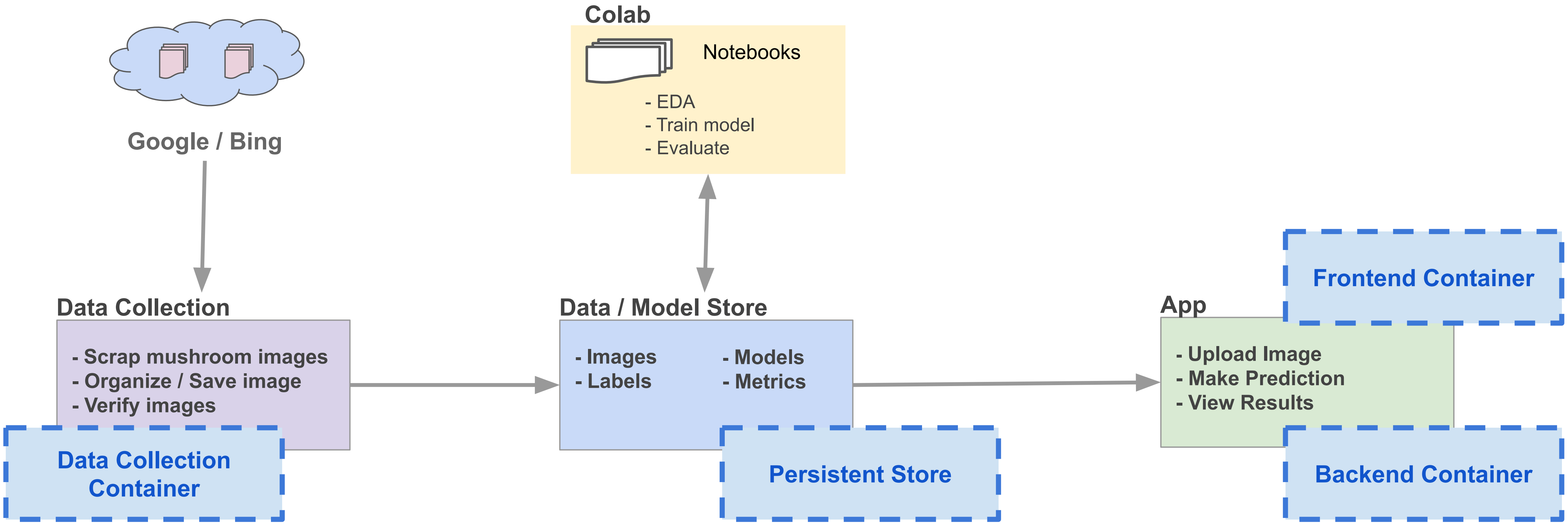
Data / Model Store

- Images
- Labels
- Models
- Metrics

App

- Upload Image
- Make Prediction
- View Results

Review: Process Flow



Google / Bing

Colab

Notebooks

- EDA
- Train model
- Evaluate

Data Collection

- Scrap mushroom images
- Organize / Save image
- Verify images

Data Collection Container

Data / Model Store

- Images
- Labels

- Models
- Metrics

Persistent Store

App

- Upload Image
- Make Prediction
- View Results

Frontend Container

Backend Container

Mushroom App: Identifying Components

- Script to download images from Google
- A persistent storage for data and models
- Backend APIs
- Frontend App

Outline

1. Recap
2. Motivation
3. **App Design**
4. Setup & Code Organization

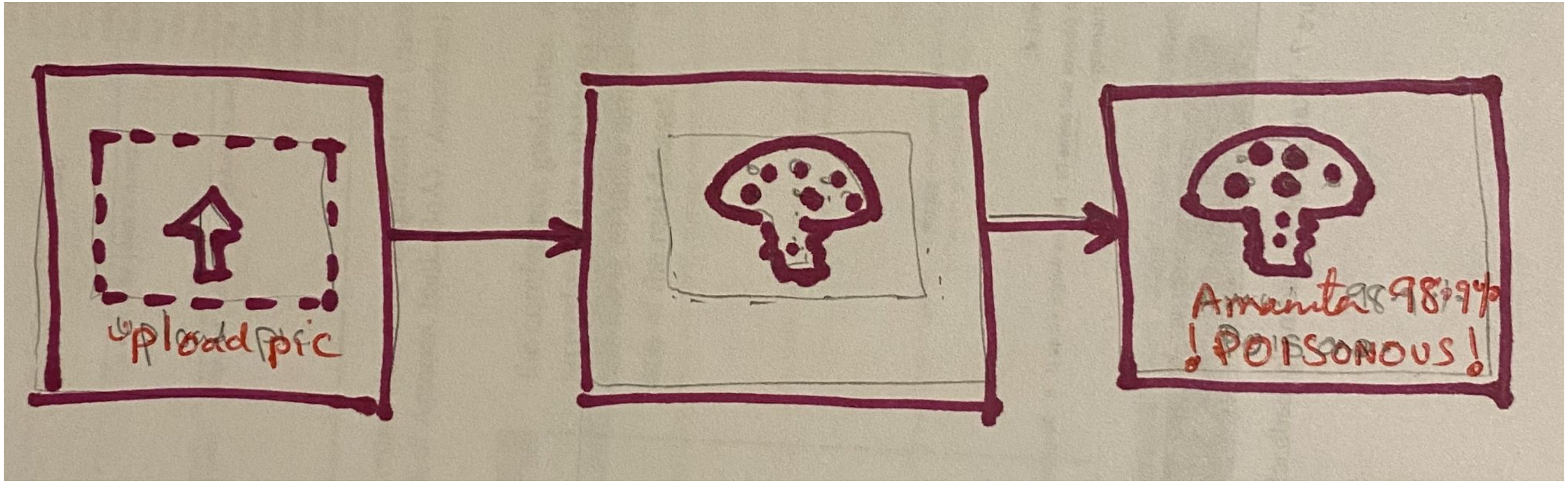
App Design

- 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

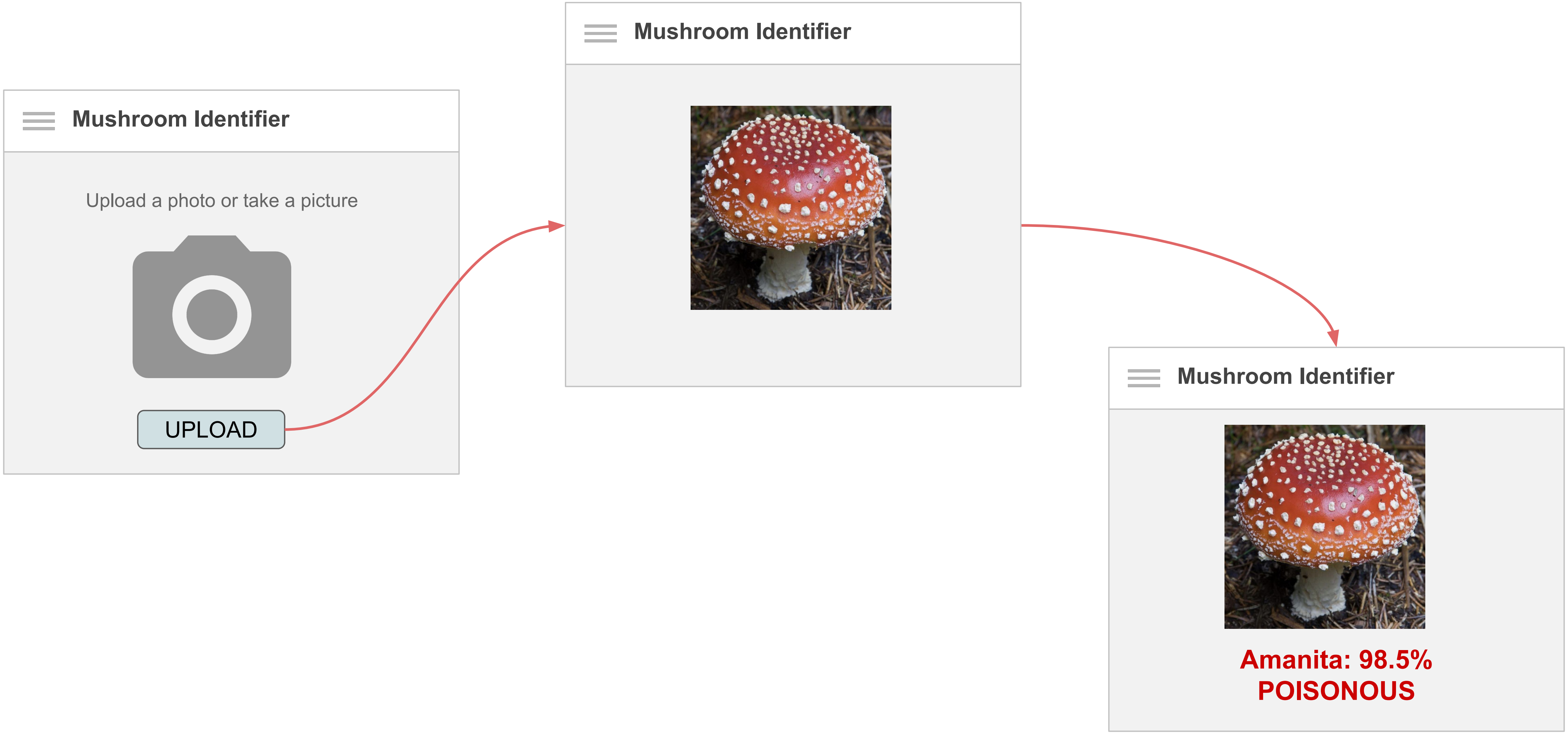
Screenflow & Wireframes

Start with brainstorming ideas on whiteboard/paper

Screenflow & Wireframes



Screenflow & Wireframes



Mushroom Identifier

Upload a photo or take a picture

Mushroom Identifier

Mushroom Identifier

Amanita: 98.5%
POISONOUS

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

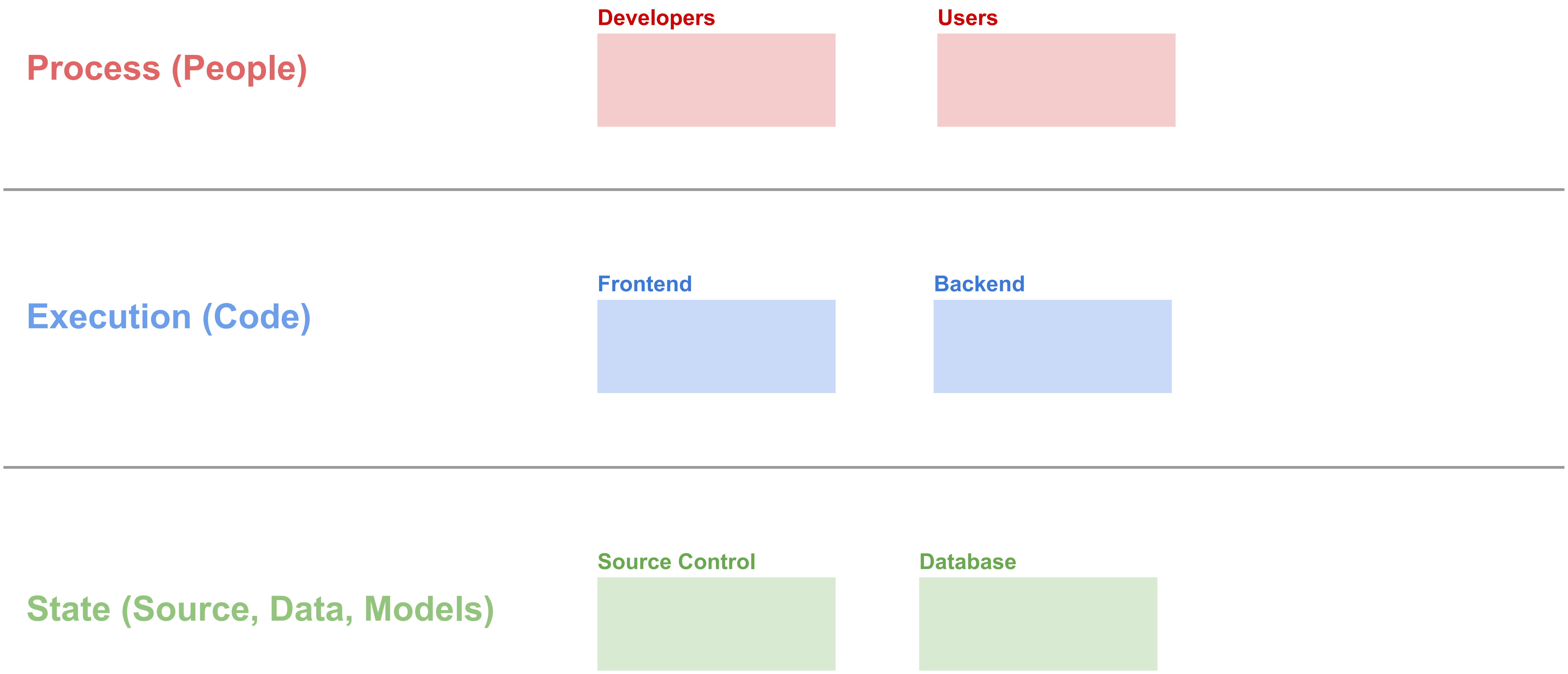
Solution Architecture

Process (People)

Execution (Code)

State (Source, Data, Models)

Solution Architecture



Process (People)

Developers

Users

Execution (Code)

Frontend

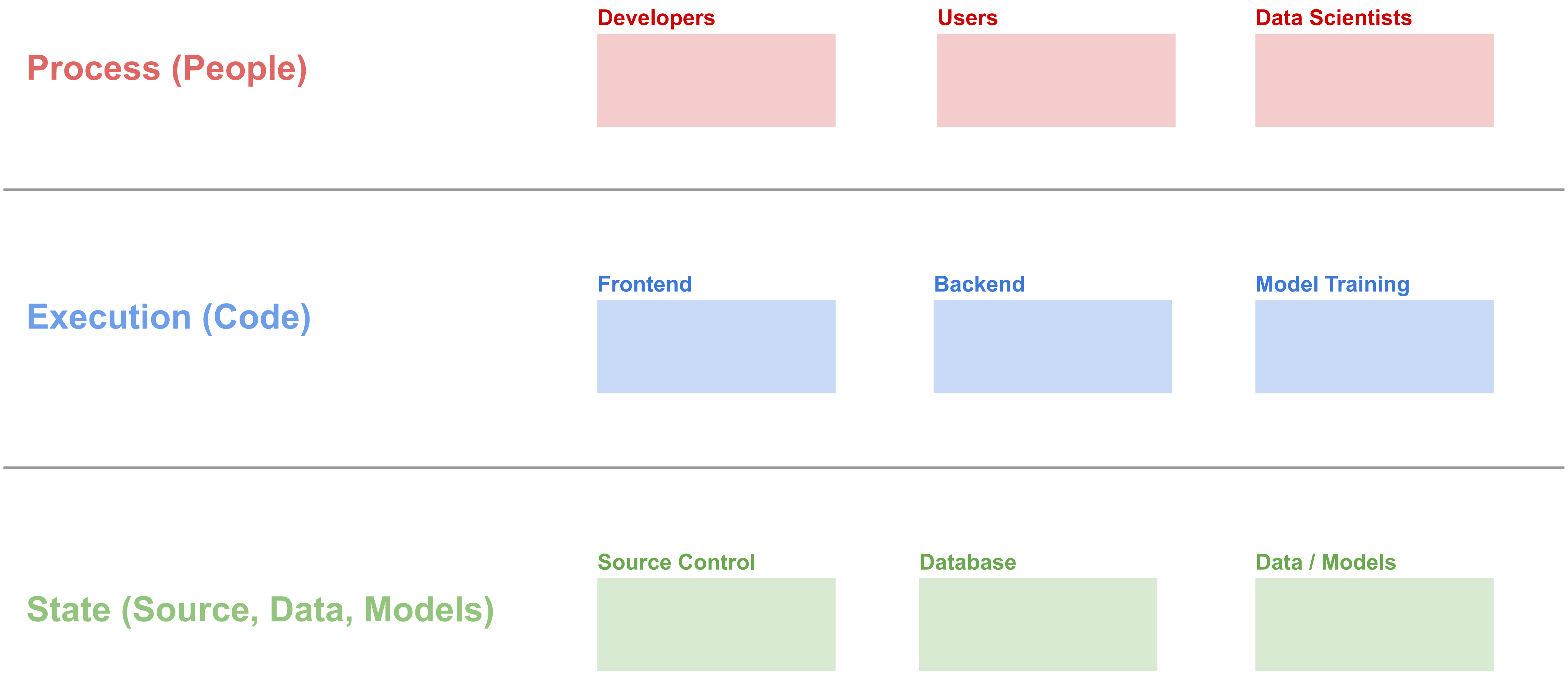
Backend

State (Source, Data, Models)

Source Control

Database

Solution Architecture



Process (People)

Developers

Users

Data Scientists

Execution (Code)

Frontend

Backend

Model Training

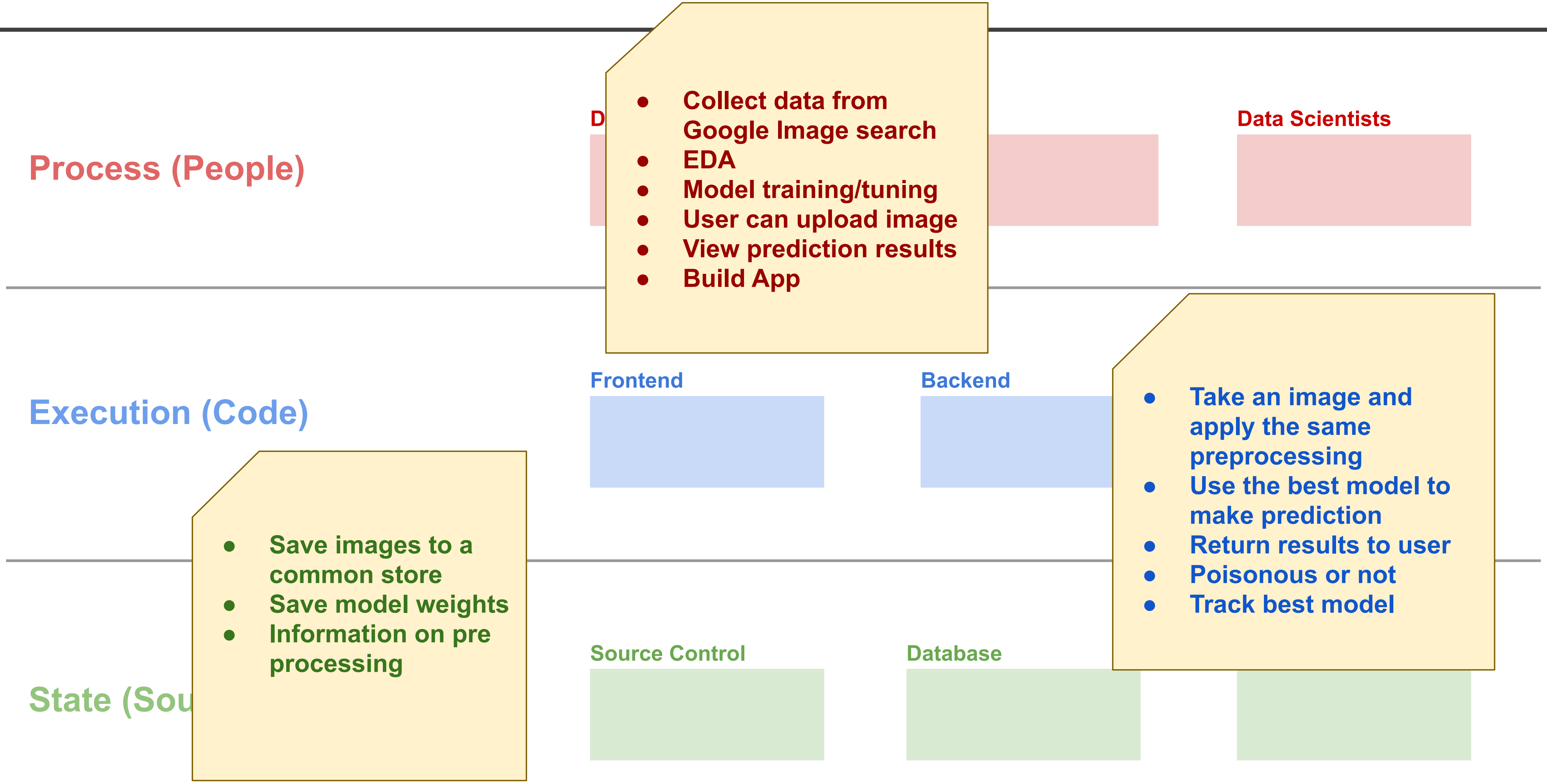
State (Source, Data, Models)

Source Control

Database

Data / Models

Solution Architecture



Process (People)

D

Data Scientists

- Collect data from Google Image search
- EDA
- Model training/tuning
- User can upload image
- View prediction results
- Build App

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

Source Control

Database

- Save images to a common store
- Save model weights
- Information on pre processing

Solution Architecture

Process

Execution

State

Solution Architecture

Process

Develop App

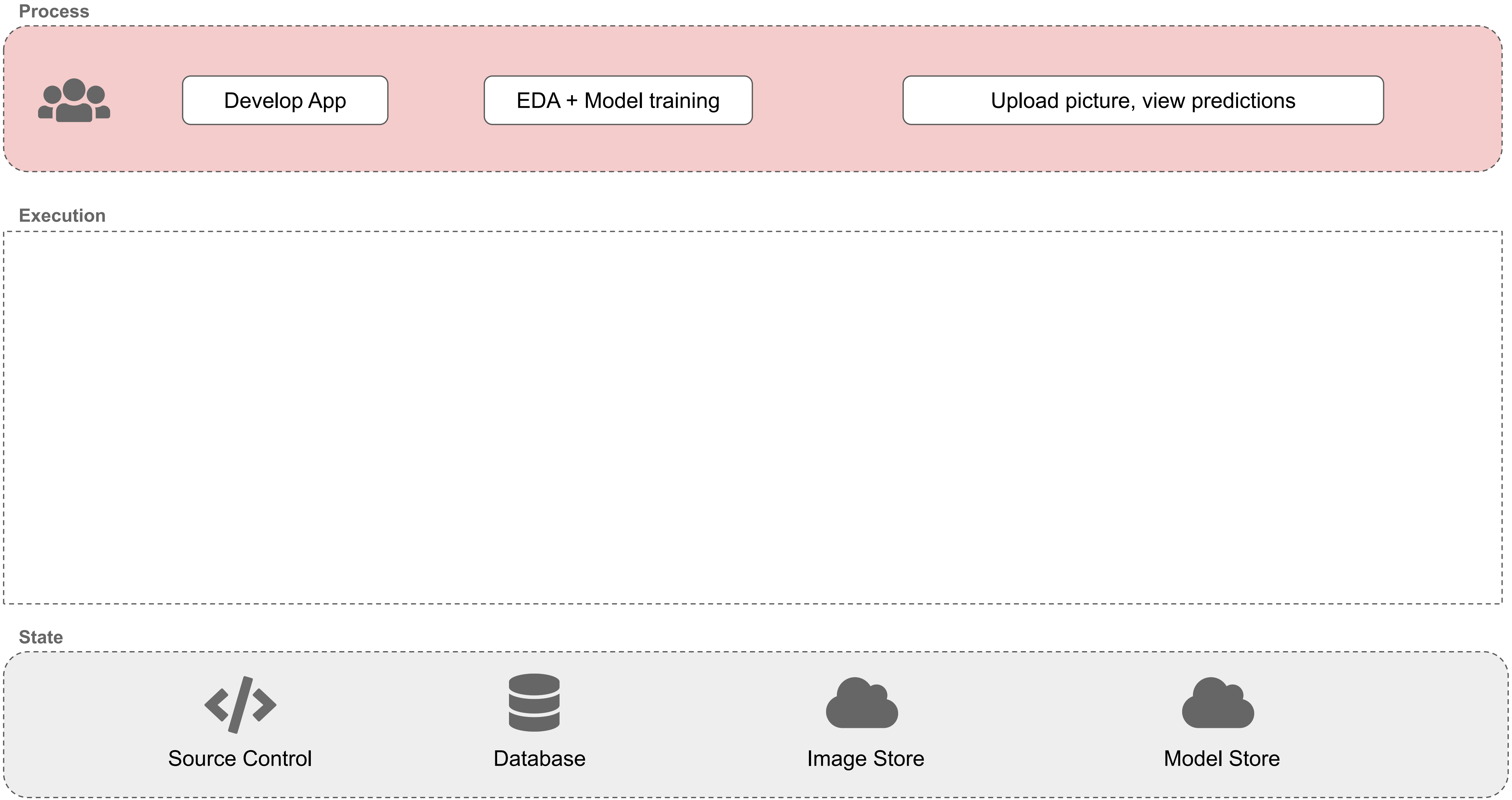
EDA + Model training

Upload picture, view predictions

Execution

State

Solution Architecture



Process

Develop App

EDA + Model training

Upload picture, view predictions

Execution

State

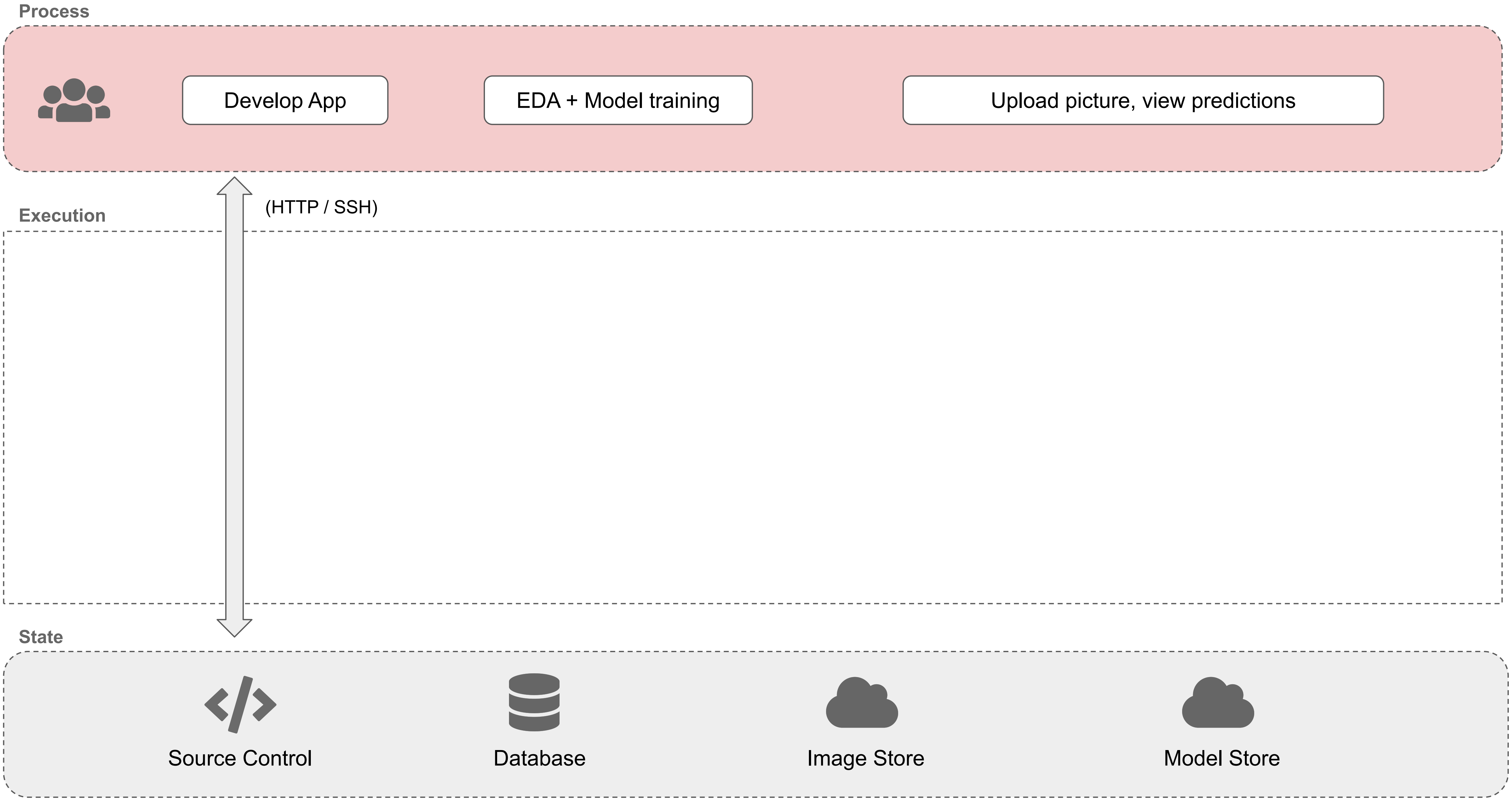
Source Control

Database

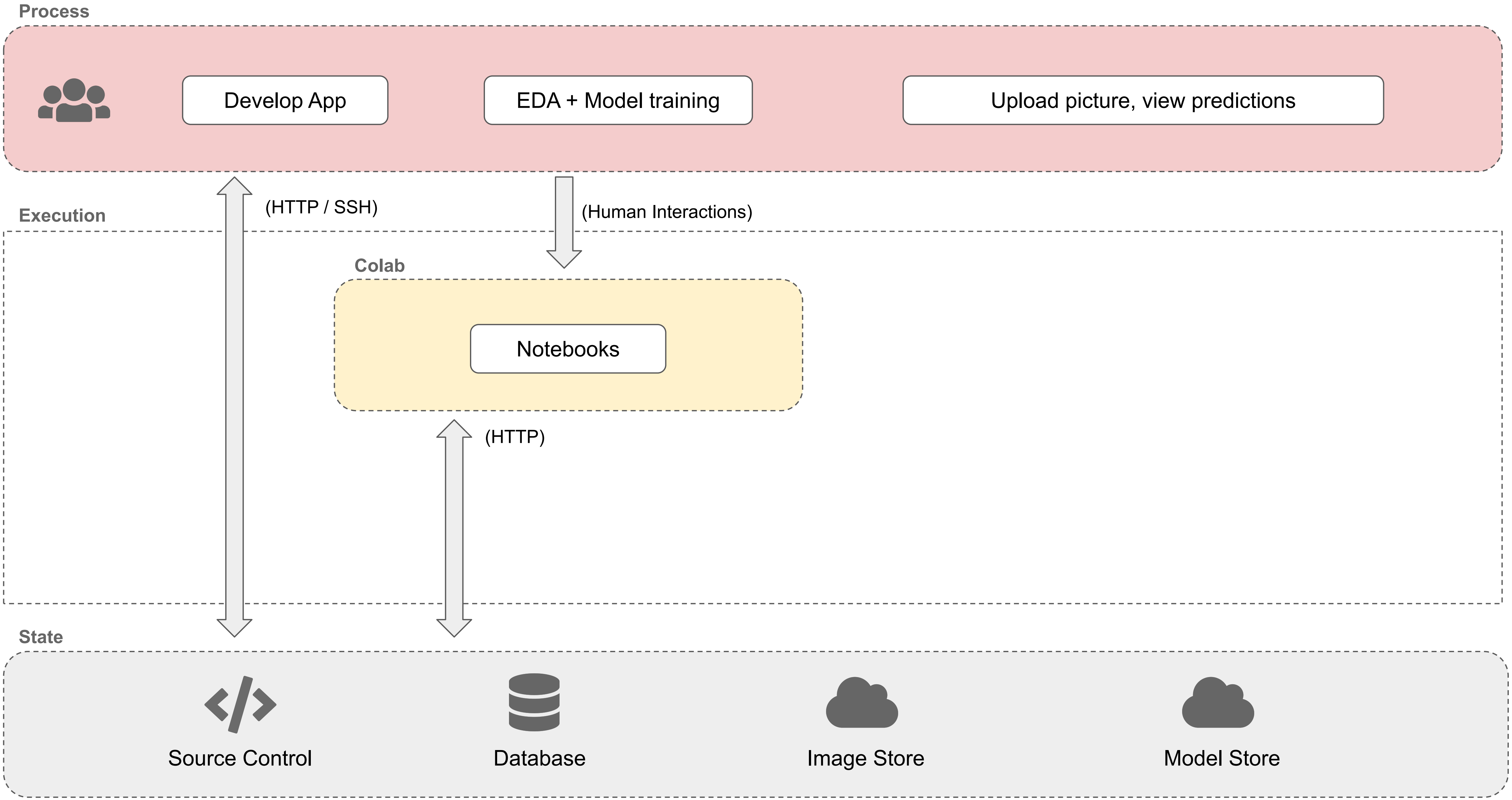
Image Store

Model Store

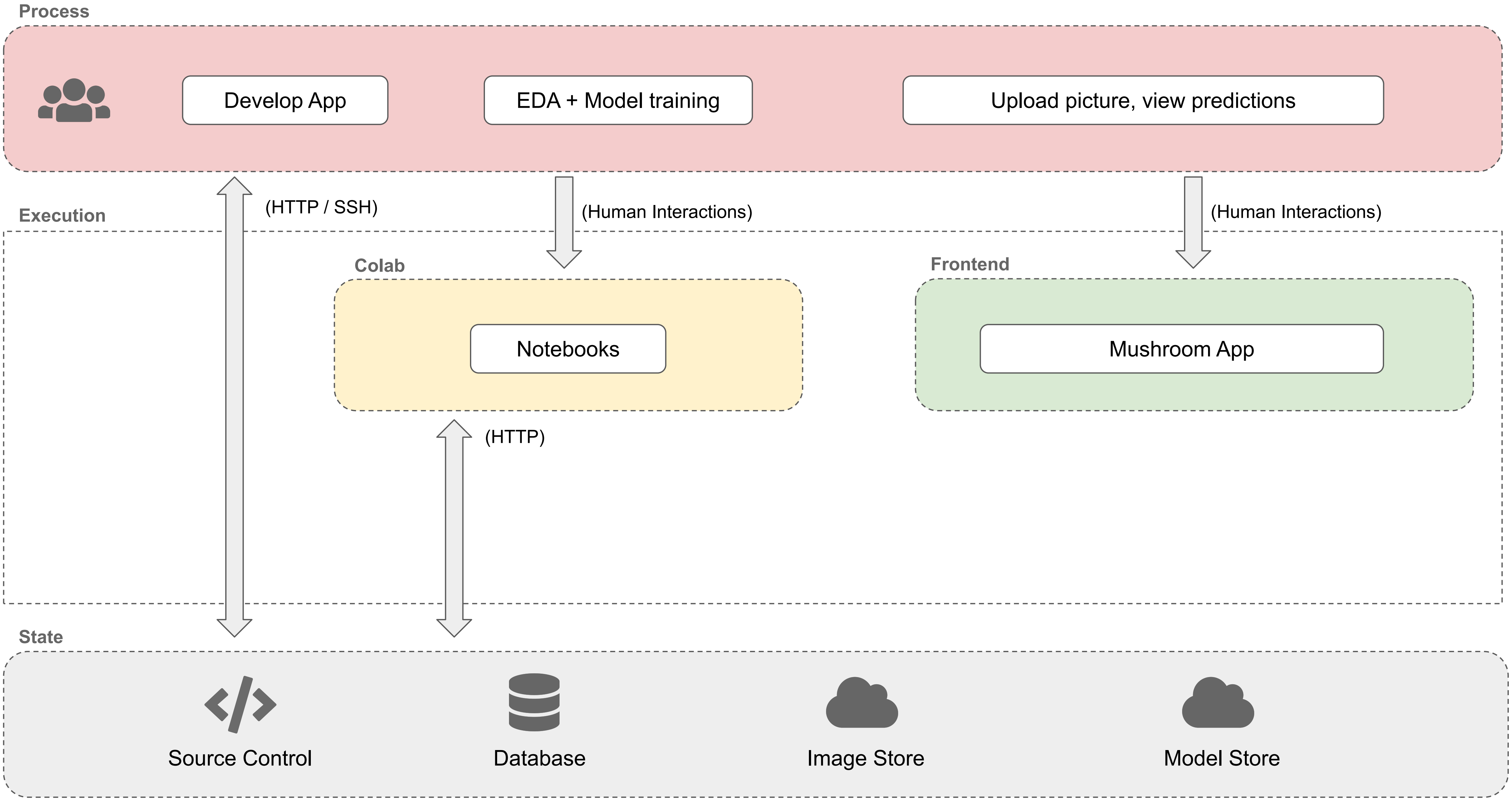
Solution Architecture



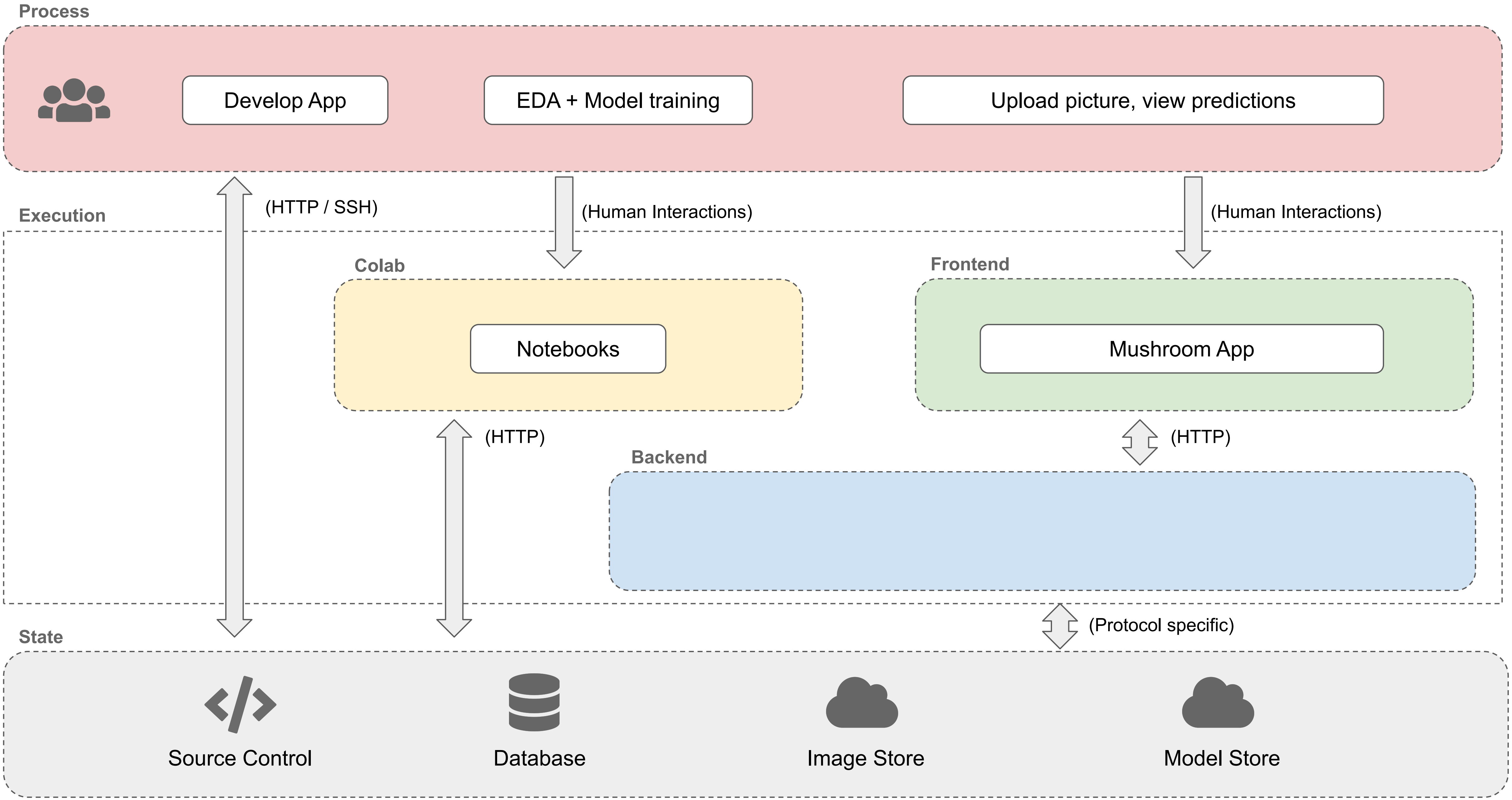
Solution Architecture



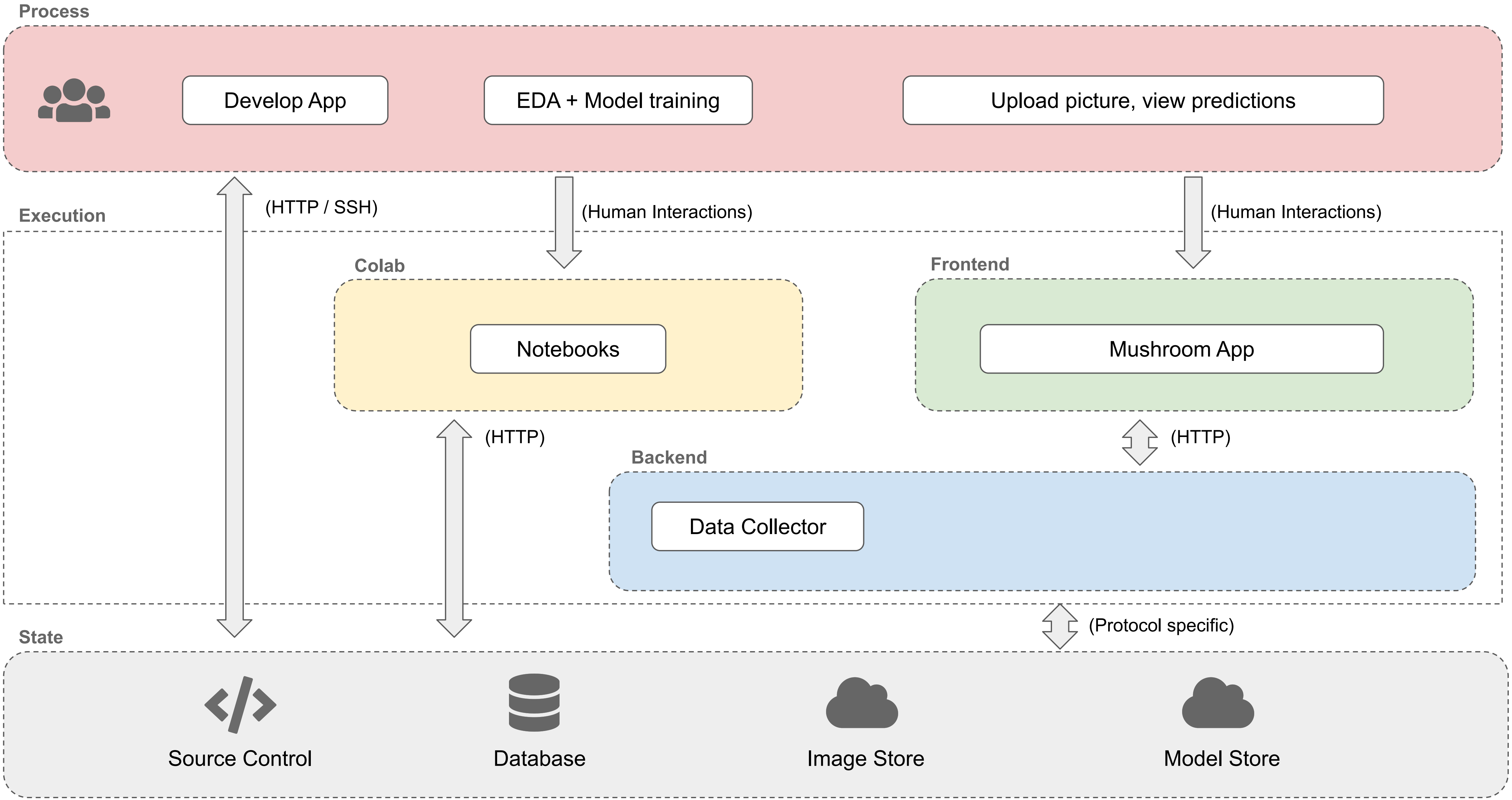
Solution Architecture



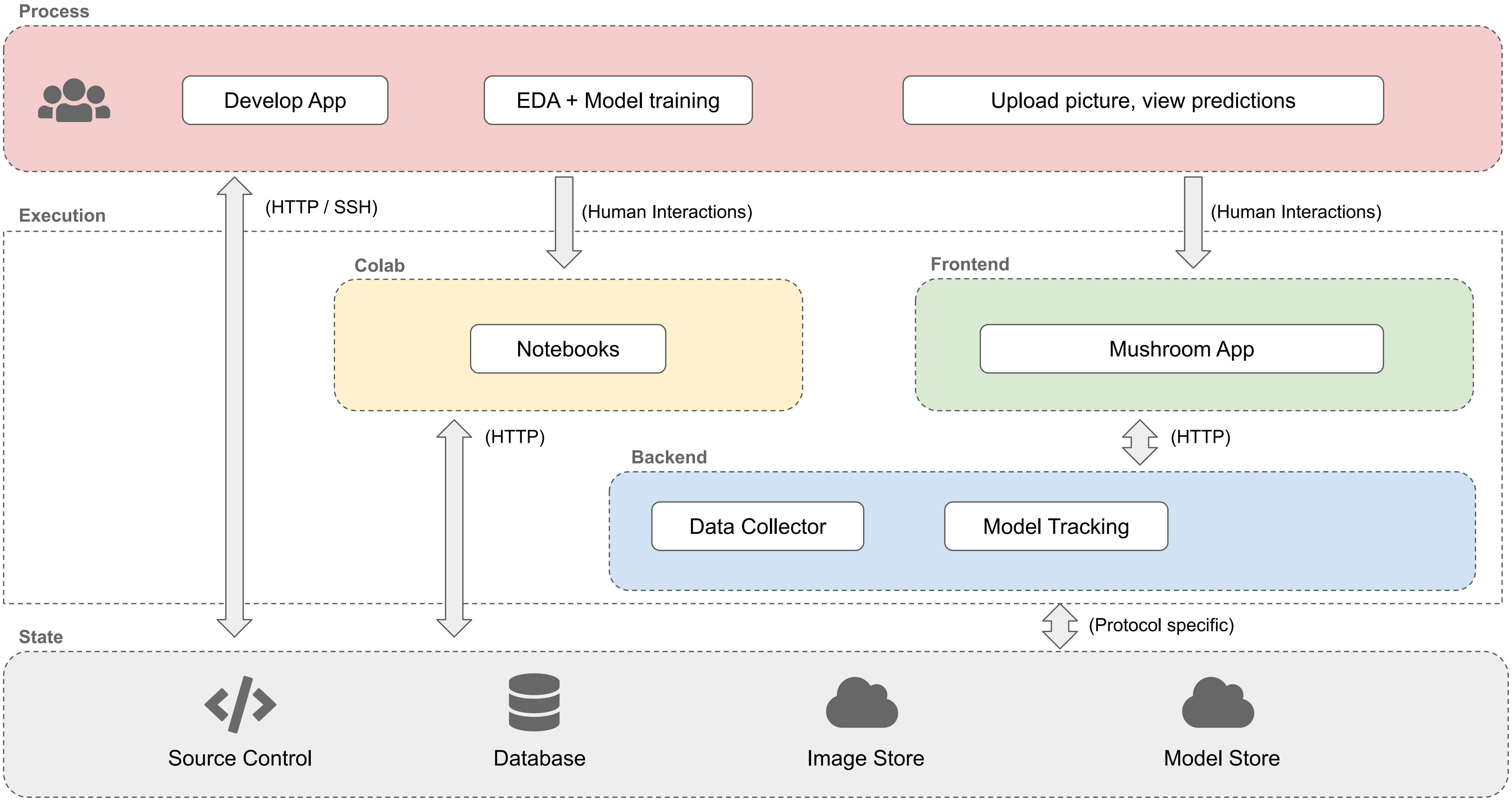
Solution Architecture



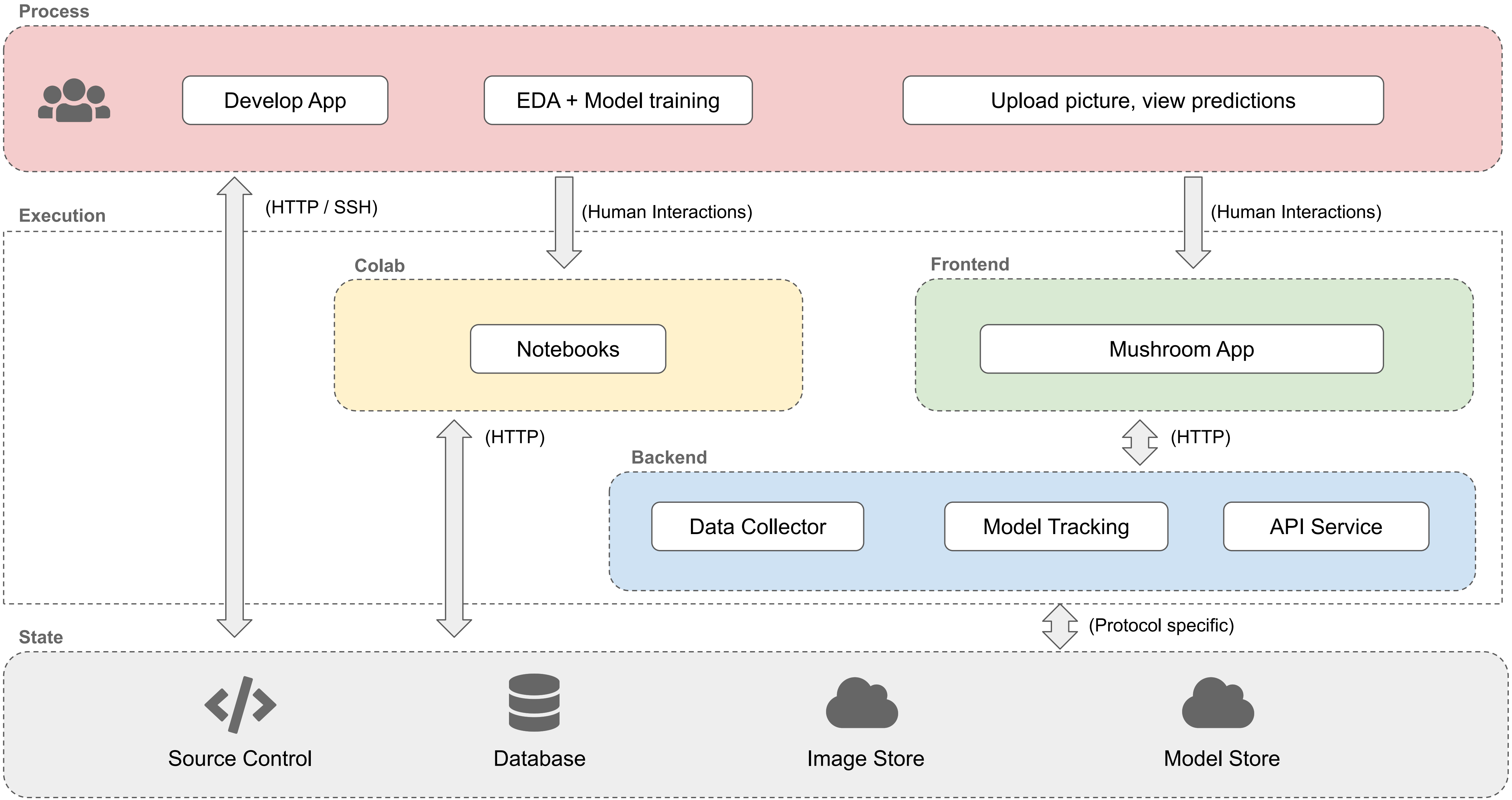
Solution Architecture



Solution Architecture



Solution Architecture



Solution Architecture Summary

- **Process**
  - Developers build App
  - Users can upload pictures and view predictions
  - Data Scientists perform model training
- **Colab**
  - Web based hosted notebook solution from Google with access to GPUs for model training
- **Frontend**
  - User friendly single page app with capabilities to upload an image and view prediction results
- **Backend**
  - API server
  - Data collector
  - Model Tracking
- **State**
  - Source control to store/version code
  - Database to store the prediction metrics or other metadata
  - Image store for the raw image file
  - Models and model artifacts store

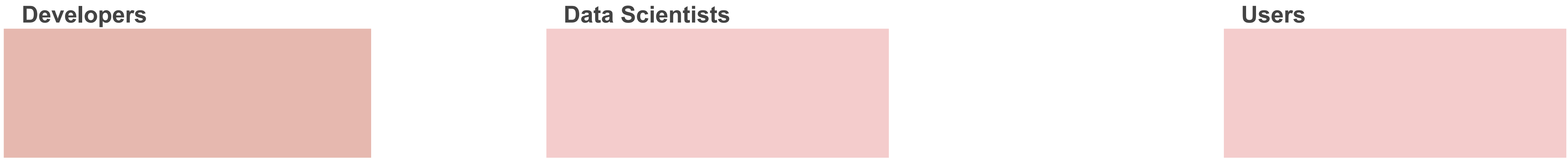
Tutorials

[Building Solution Architecture for your Project]

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

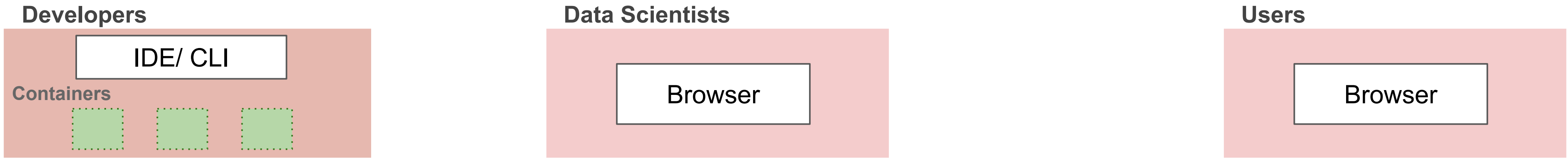


Developers

Data Scientists

Users

Building a Technical Architecture



Developers

IDE/ CLI

Containers

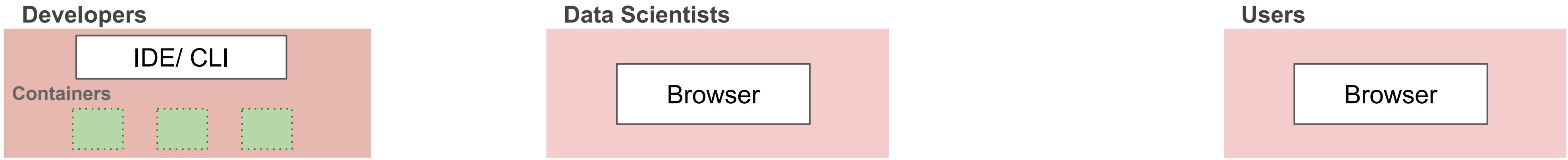
Data Scientists

Browser

Users

Browser

Building a Technical Architecture



Developers

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

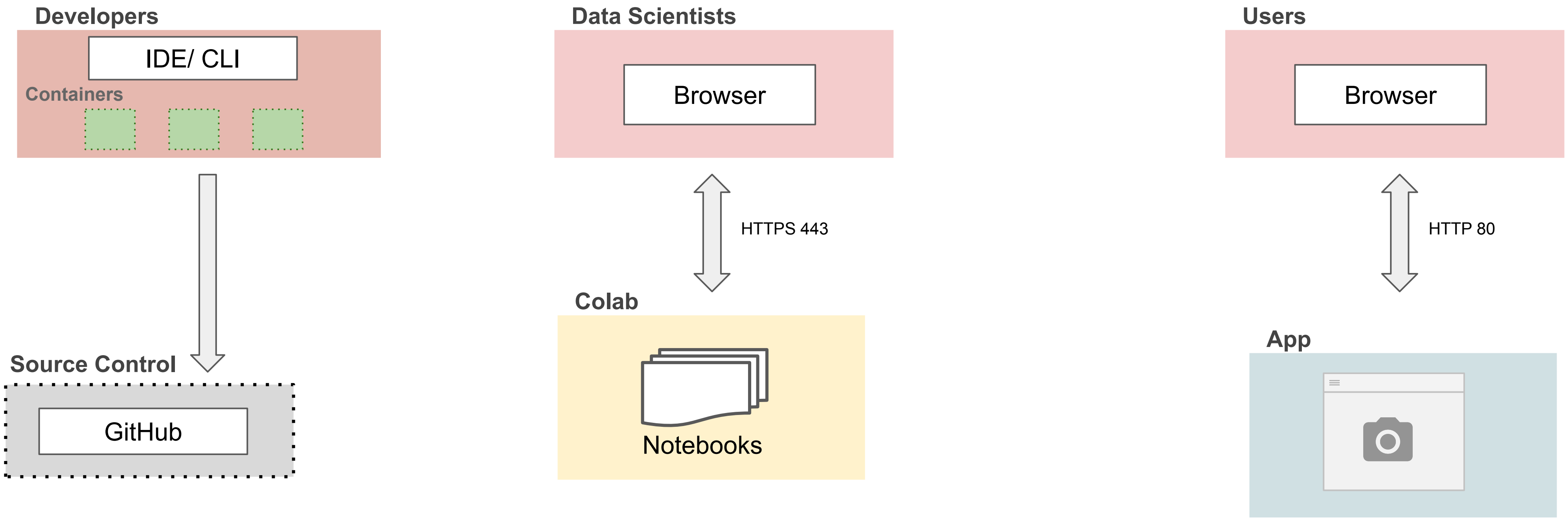
Data Scientists

- Use Colab/JupyterHub
- EDA & Modeling done using browser

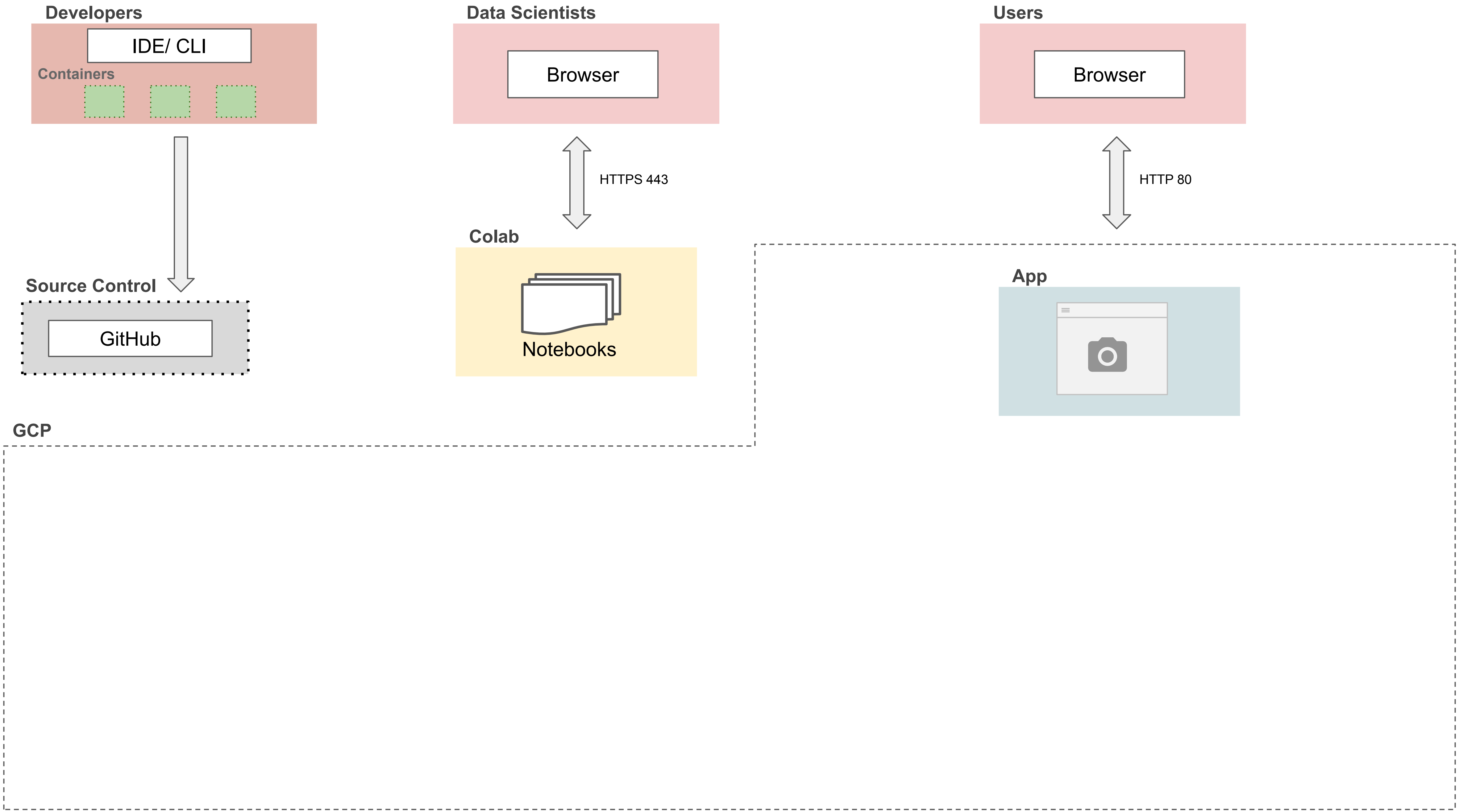
Users

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

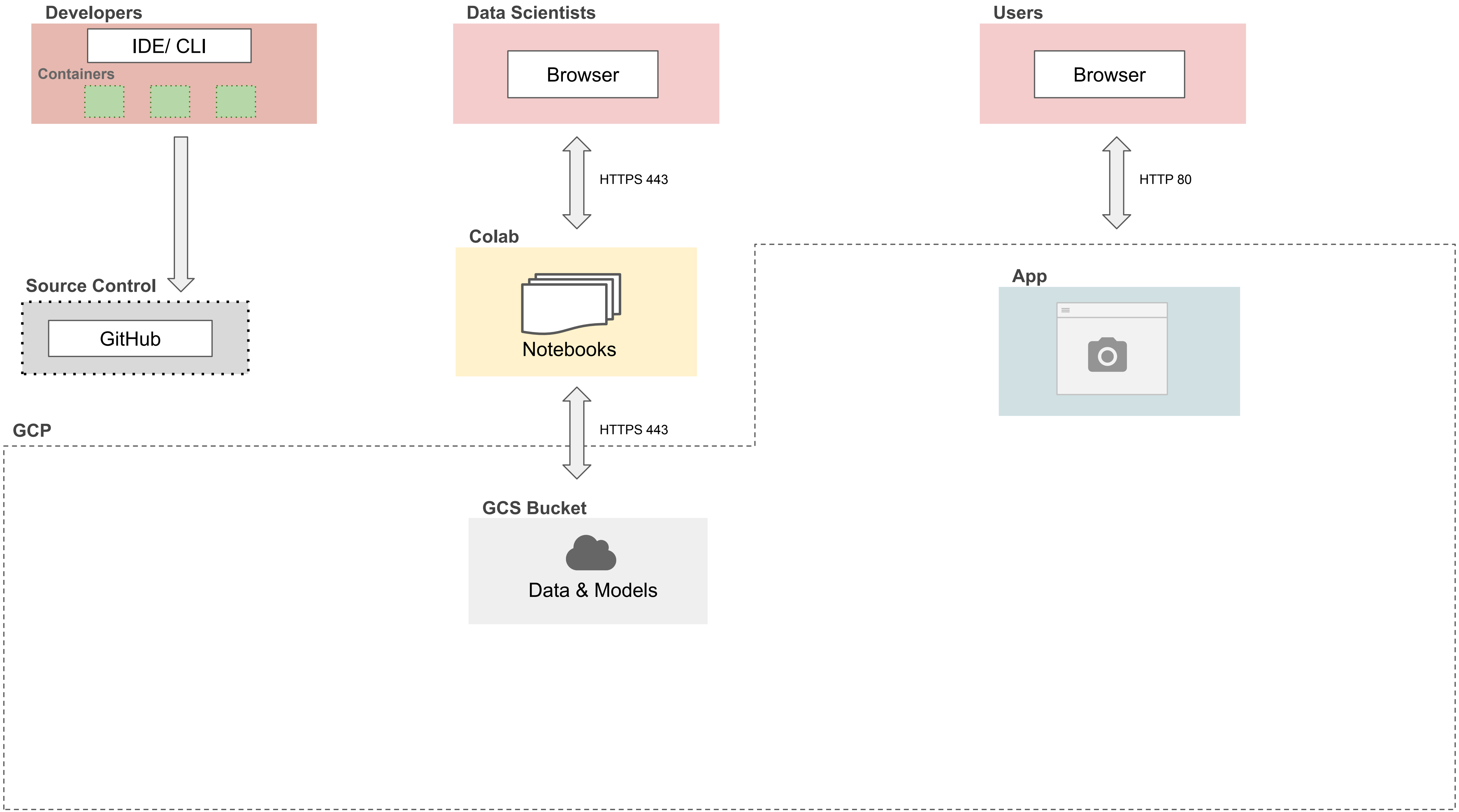
Building a Technical Architecture



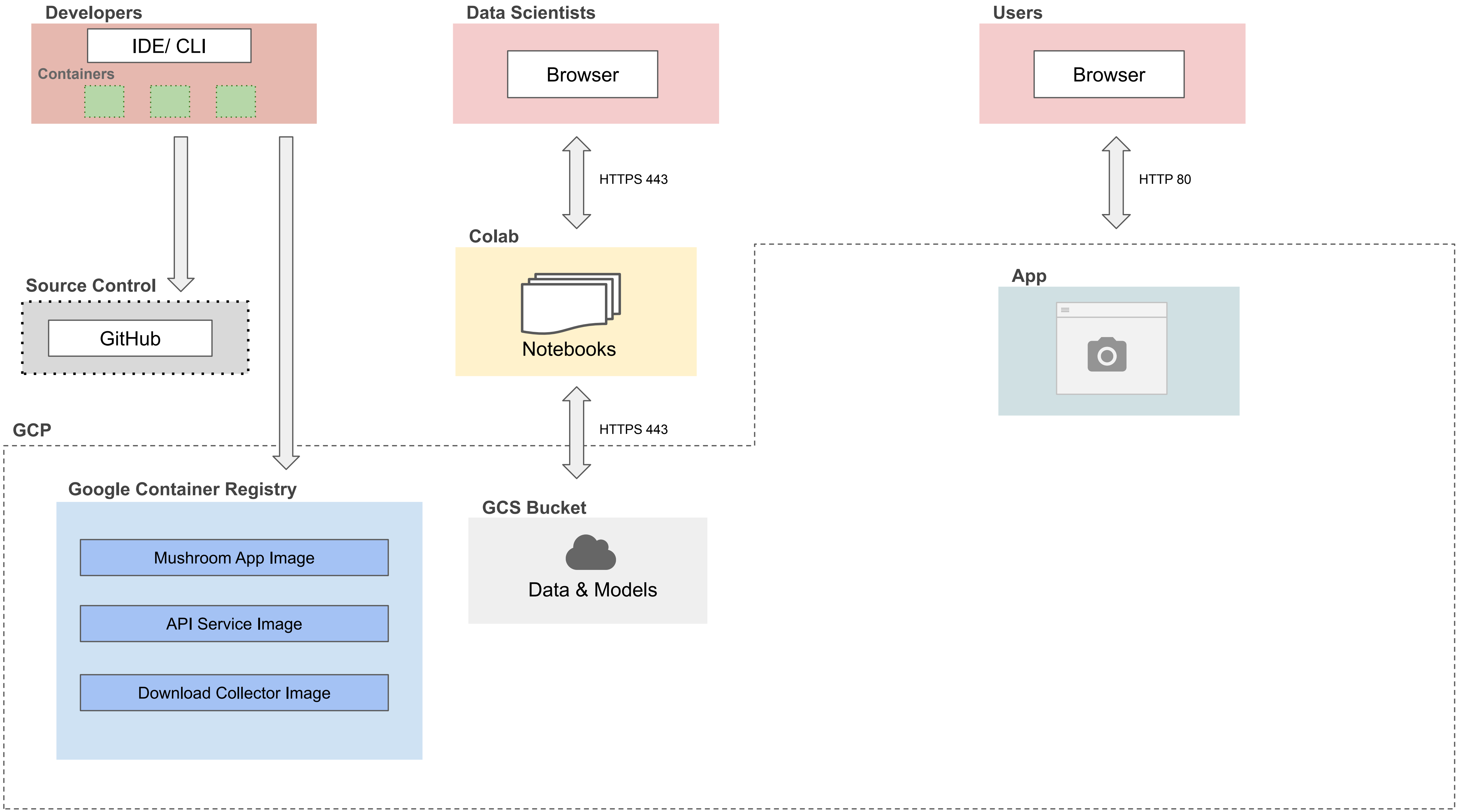
Building a Technical Architecture



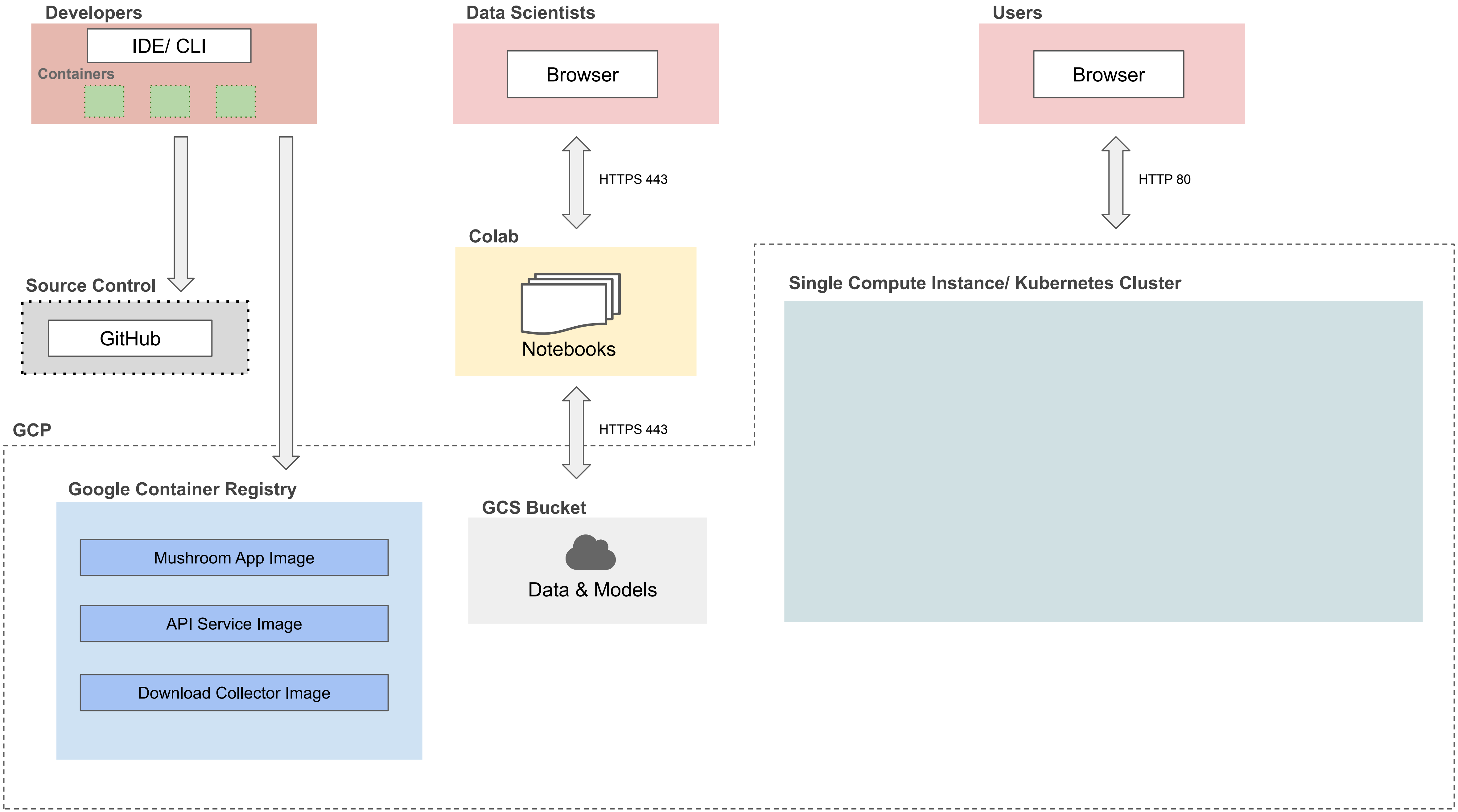
Building a Technical Architecture



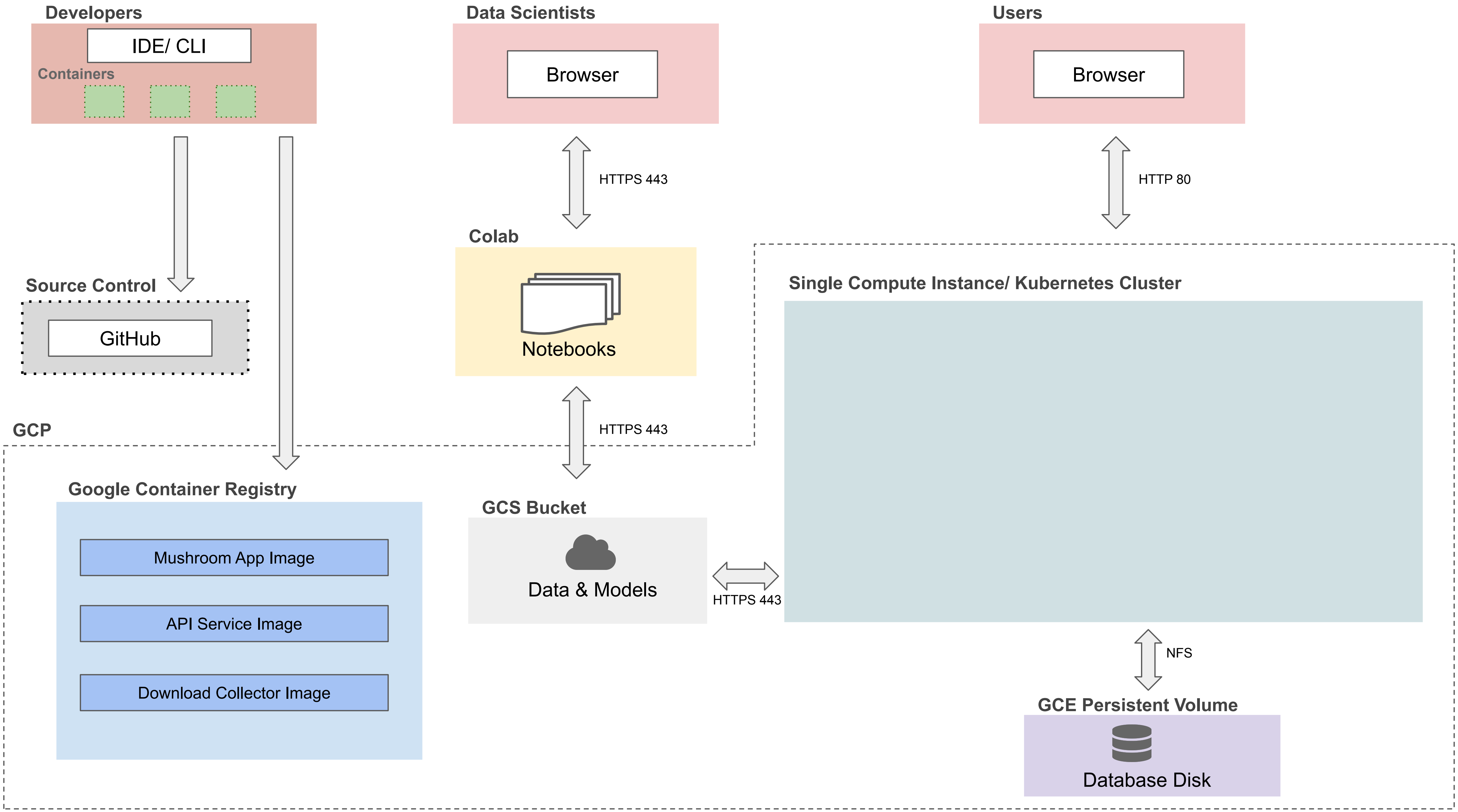
Building a Technical Architecture



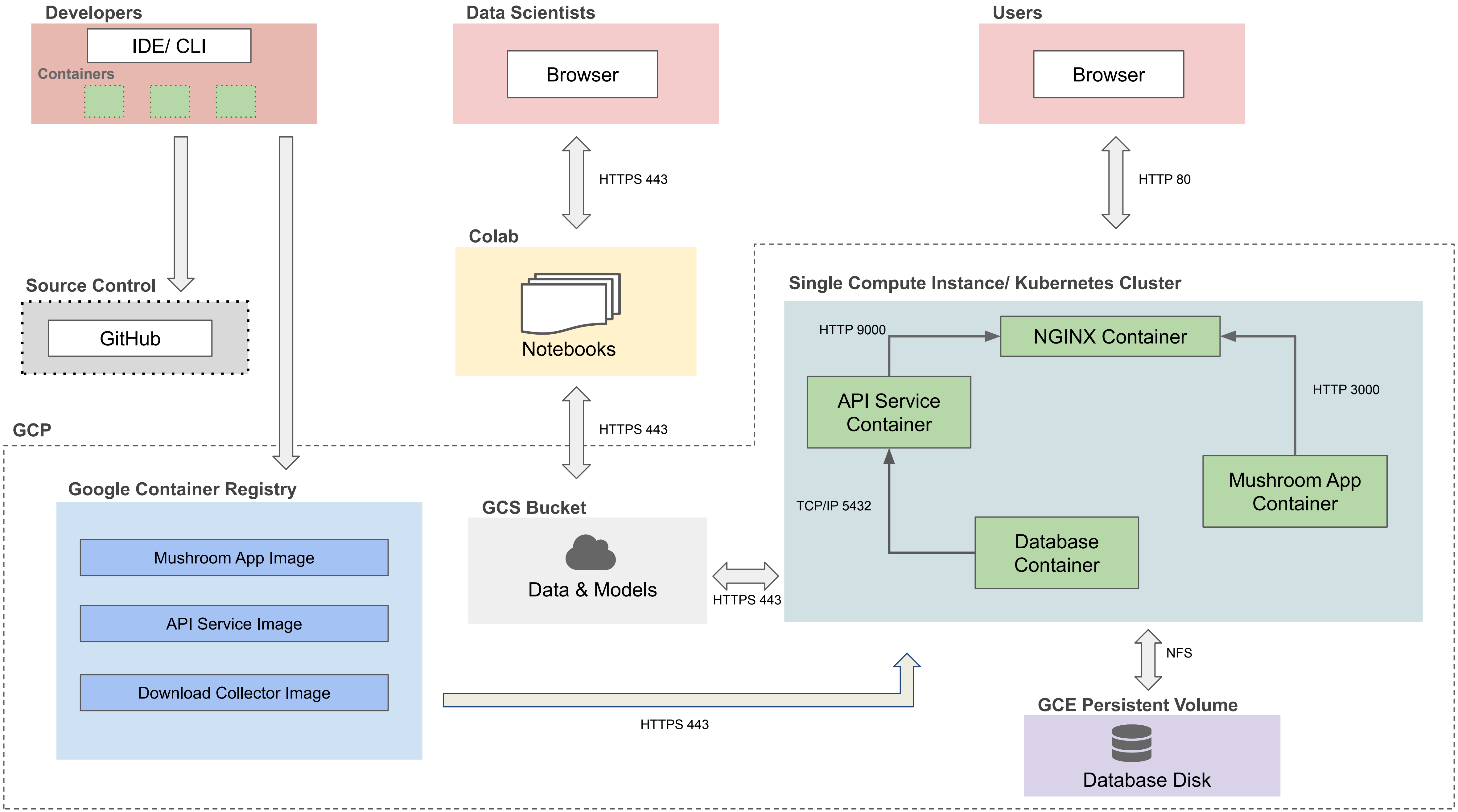
Building a Technical Architecture



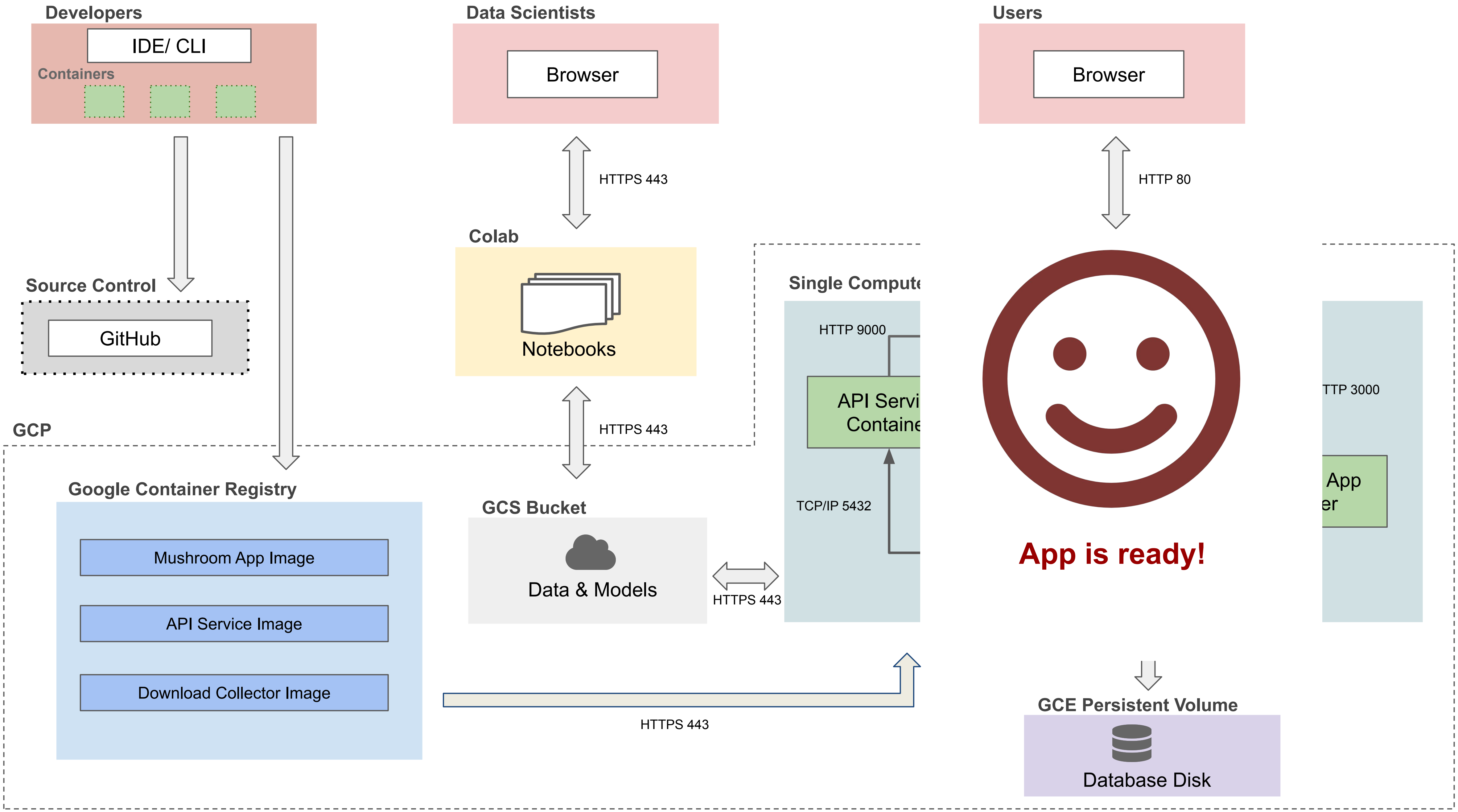
Building a Technical Architecture



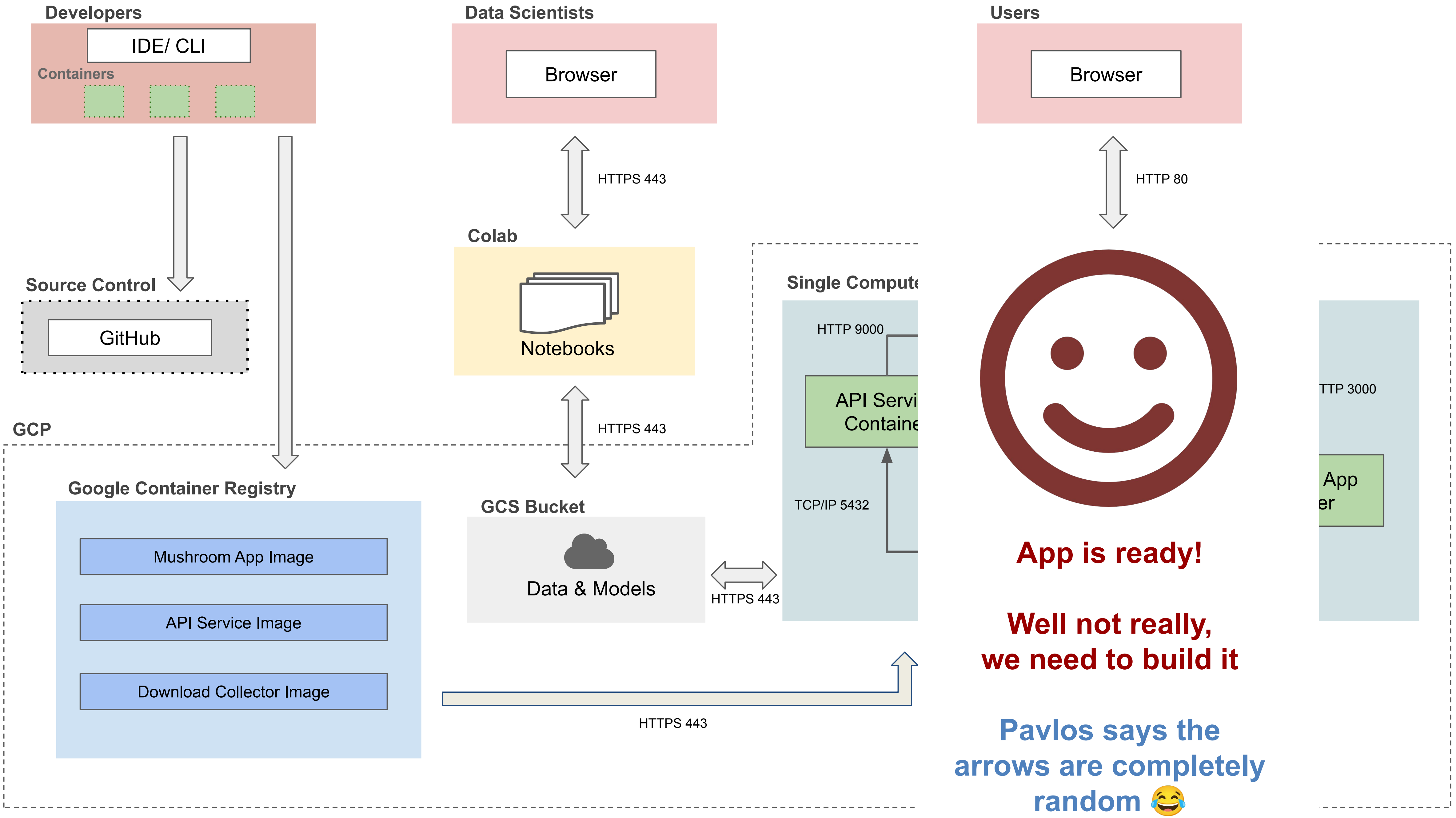
Technical Architecture



Technical Architecture



Technical Architecture



App is ready!

**Well not really,
we need to build it**

**Pavlos says the
arrows are completely
random 😂**

Technical Architecture Summary

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

- **GCE Persistent Volume**
  - Database store
- **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. **Setup & Code Organization**

Setup & Code Organization

1. Create a root project folder **mushroom-app**
2. Organize containers into sub folders
    a. api-service
    b. data-collector
    c. frontend-simple
3. Setup containers, mount folders for
    a. Persistent storage
    b. Secrets (to store GCP account keys)

Tutorials

Mushroom App - Setup & Code Organization

THANK YOU