Lecture 17: App Design, Setup & Code Organization

AC215

Shivas Jayaram

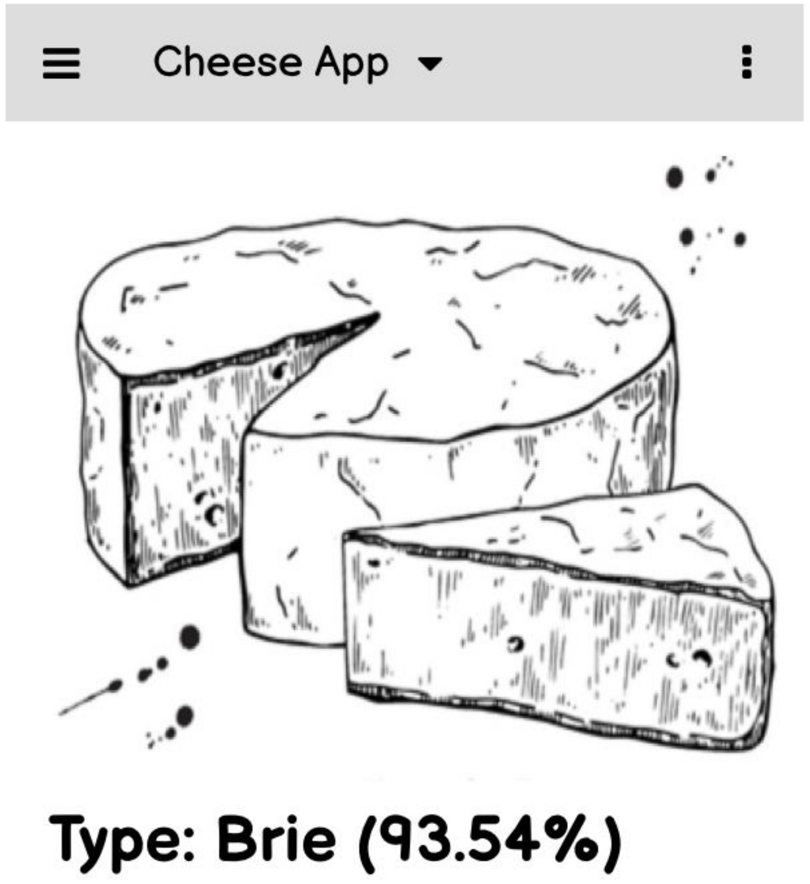
Outline

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

Outline

Recap: 🍄 Cheese App

- We want to build an app to identify a cheese by simply taking a photo of it
- Dive deeper into the world of cheese with our interactive chatbot
- How do we build the app?

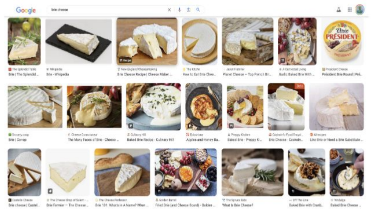


Recap: How do we build an App?

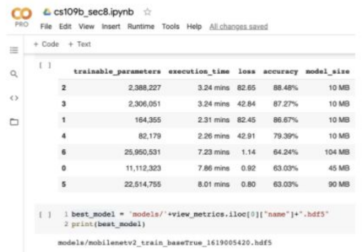
- Collaborate with team to design and develop.
- Build a robust ml pipeline for data and models.
- Expose python functions as backend APIs.
- Build a frontend using HTML & javascript.
- Deploy app to a cloud provider.
- https://formaggio.me/ [Go live]

Recap: How do we build an App?

Development



Python Script



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

Rest API

Best Model

IDE / Code Editor

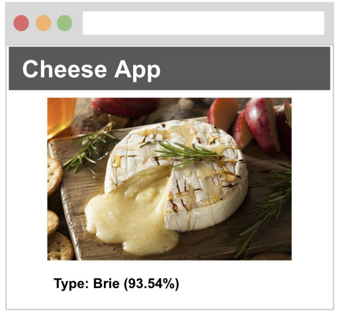
Python: pipenv
Chromium: Mac install, Windows install
OS: Mac, Windows

Python: Colab provided env
OS: Linux

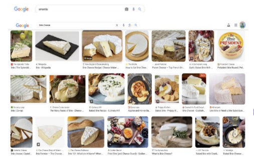
Python: pipenv
OS: Mac, Windows

Multiple developers, Using Mac and Windows OS

Deployment



Model Training



Python Script

Python: pipenv
OS: Linux

Server

Recap: Tools

Data:

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

Model:

- Gemini
- Vertex AI Fine Tuning / Training
- Vertex AI Deploy
- W&B

Operations:

- GitHub
- Docker
- Vertex AI Pipelines
- GCP
- Modal

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

Imagine being able to identify a cheese by simply taking a photo of it. Our app uses AI-powered visual recognition technology to help you identify the cheese you're looking at, and then provides you with a wealth of information about it.

Take a photo of the cheese, and our app will identify it for you. Then, dive deeper into the world of cheese with our interactive chatbot. Ask questions about the cheese's origin, production process, nutritional information, and history.

Review: Proposed Solution

Key Features:

- Visual cheese identification using AI-powered technology
- Interactive chatbot for asking questions about cheese
- In-depth information on cheese origin, production process, nutritional information, and history
- Expert advice on pairing cheese with wines, crackers, and other accompaniments
- Perfect for cheese enthusiasts, party planners, and anyone looking to explore the world of cheese

Review: Project Scope

Proof Of Concept (POC)

- Scrap cheese images and documents (books etc)
- Verify images and pdfs
- Experiment on some baseline models
- Verify new unseen cheeses are predicted by the model(s)
- Verify ideas using any instruct-LLMs

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 Cheeses and respond appropriately to a series of prompts**
- **API Server for uploading images and predicting using best model**
- **API Server for serving the language models**

Review: Project Scope

Proof Of Concept (POC)

- Scrap cheese images and documents (books etc)
- Verify images and pdfs
- Experiment on some baseline models
- Verify new unseen cheeses are predicted by the model(s)
- Verify ideas using any instruct-LLMs

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**

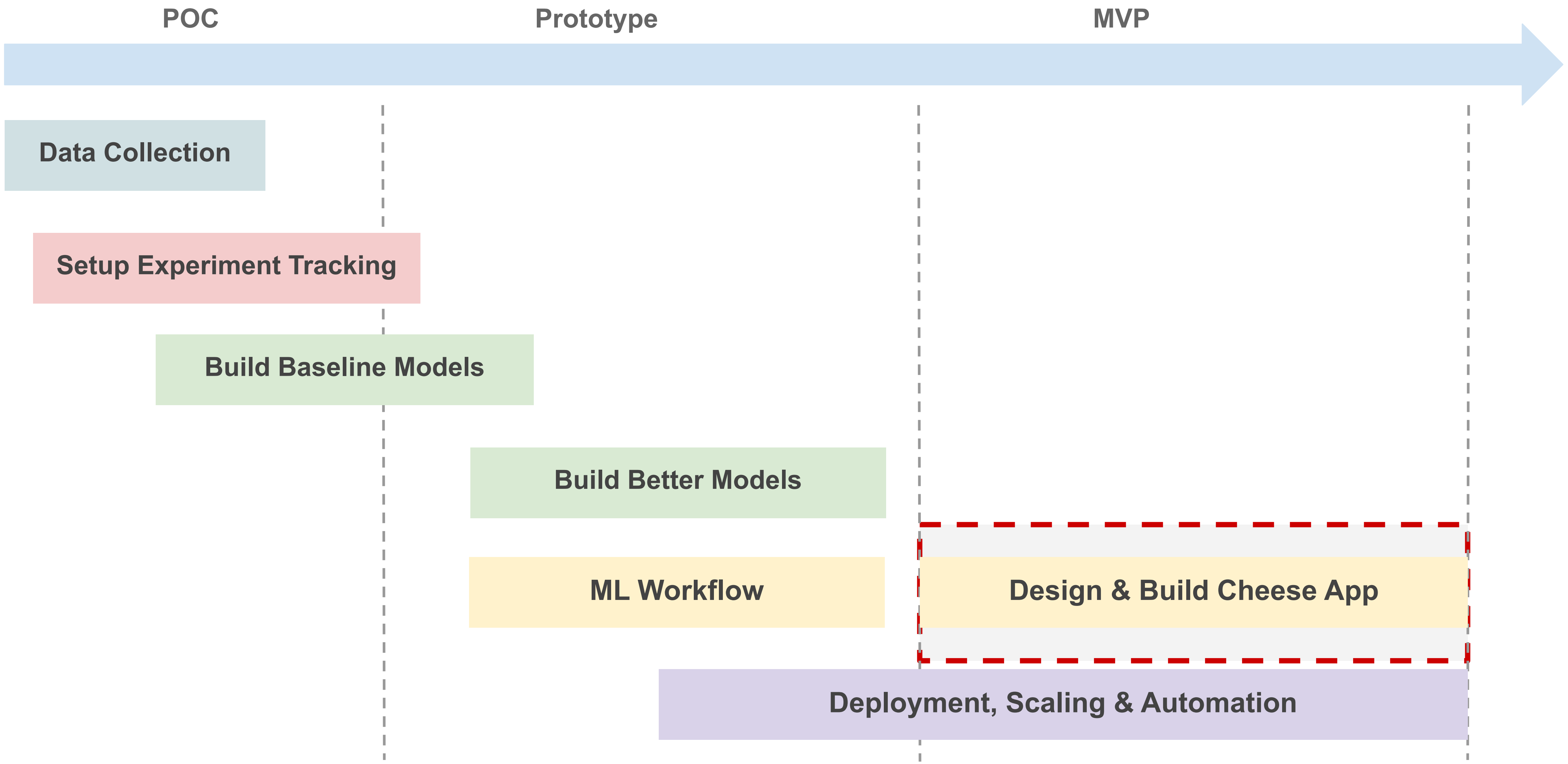
Using Streamlit

Minimum Viable Product (MVP)

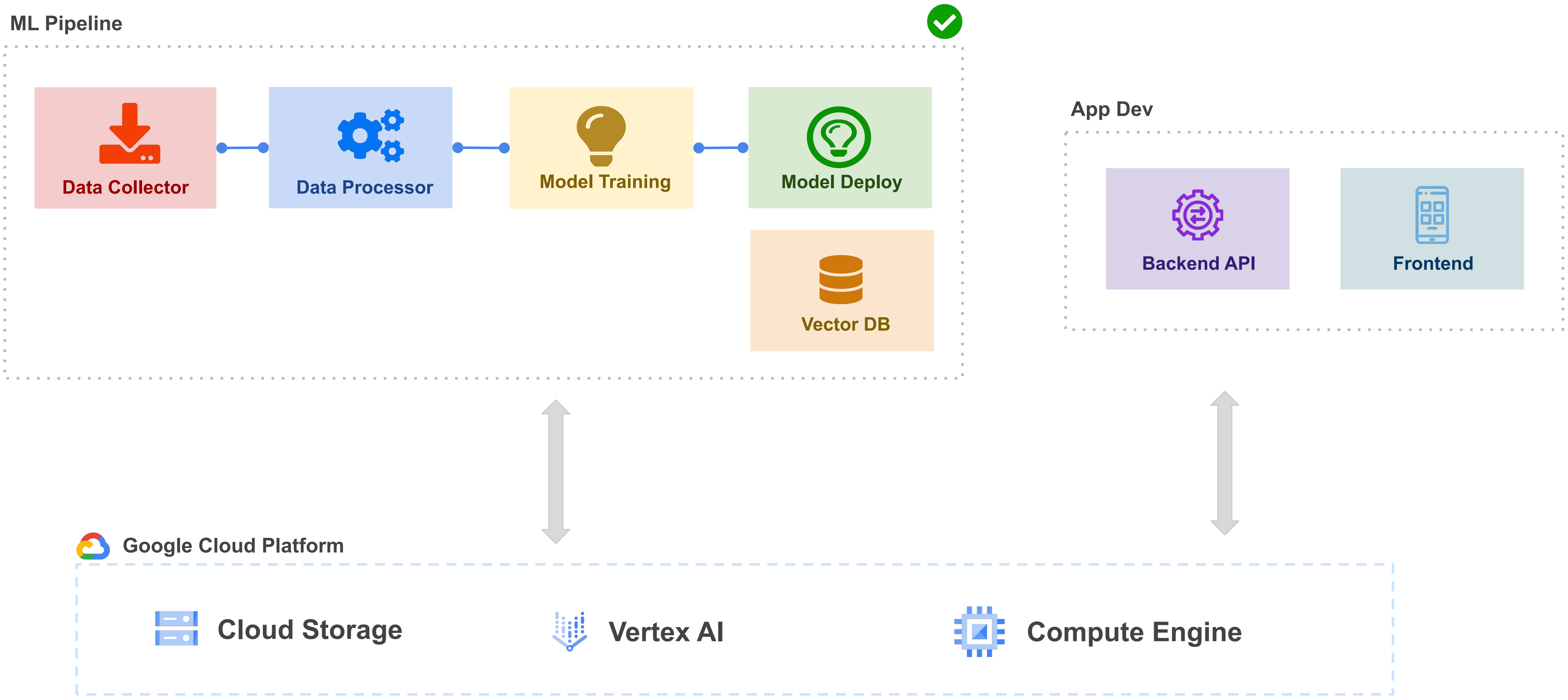
- **Create App to identify Cheeses and respond appropriately to a series of prompts**
- **API Server for uploading images and predicting using best model**
- **API Server for serving the language models**

Using Web Frameworks

Review: Cheese App Status


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

Cheese App Development



Outline

App Design

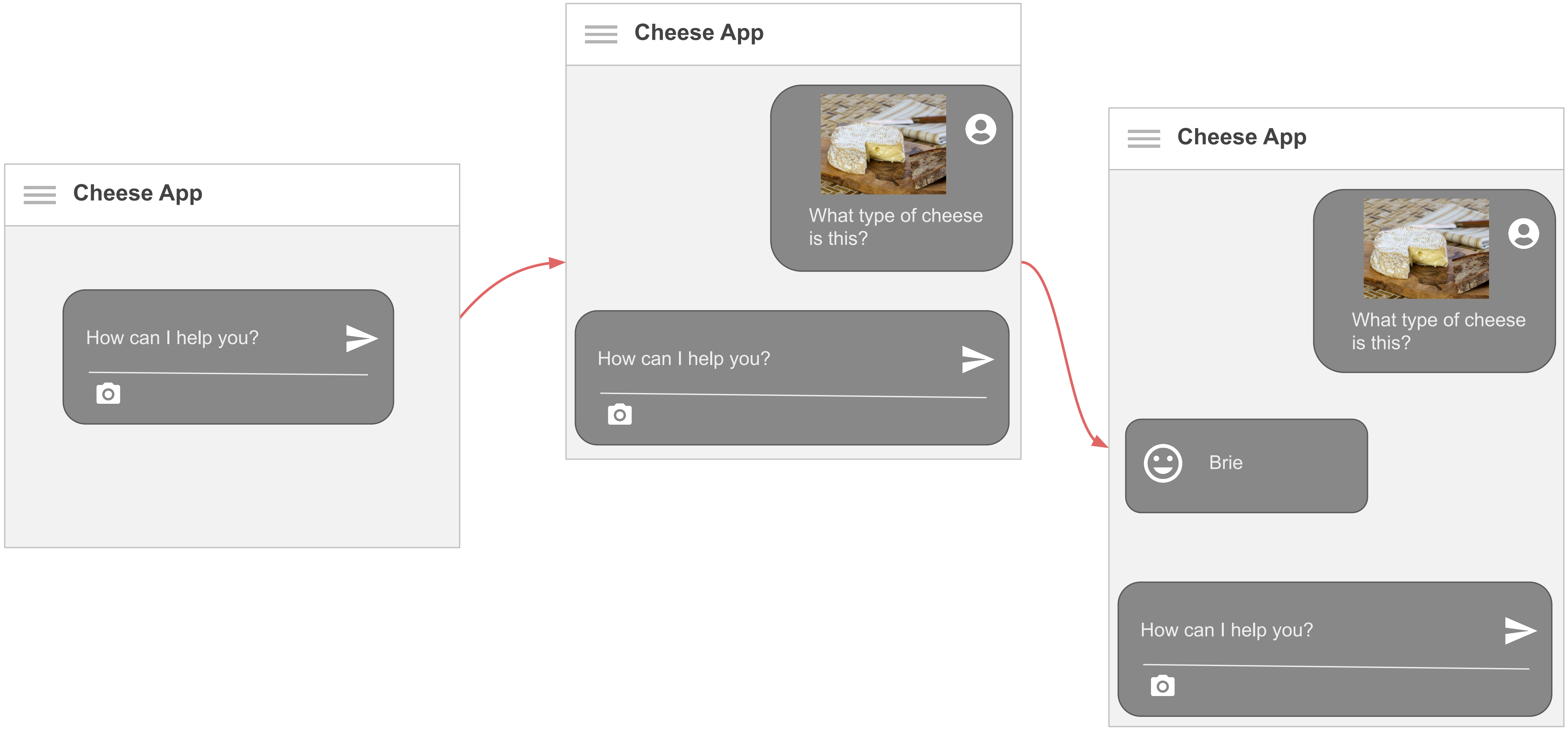
- In a traditional software app you have code and data.
- In an **AI App**, in addition you have models to perform tasks
- We will follow a structured approach to design and develop an AI App
- The design will consist of the following components:
    - Screenflow & Wireframes
    - Solution Architecture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

Outline

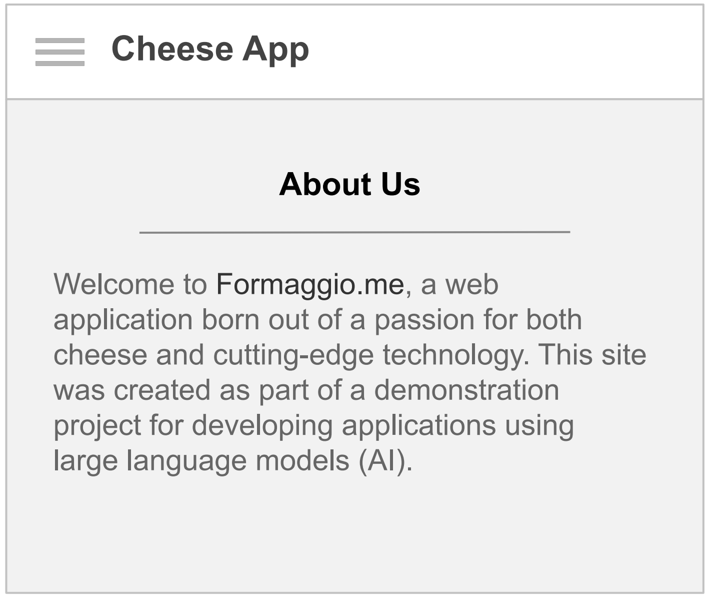
Screenflow & Wireframes

Start with brainstorming ideas on whiteboard/paper

Screenflow & Wireframes



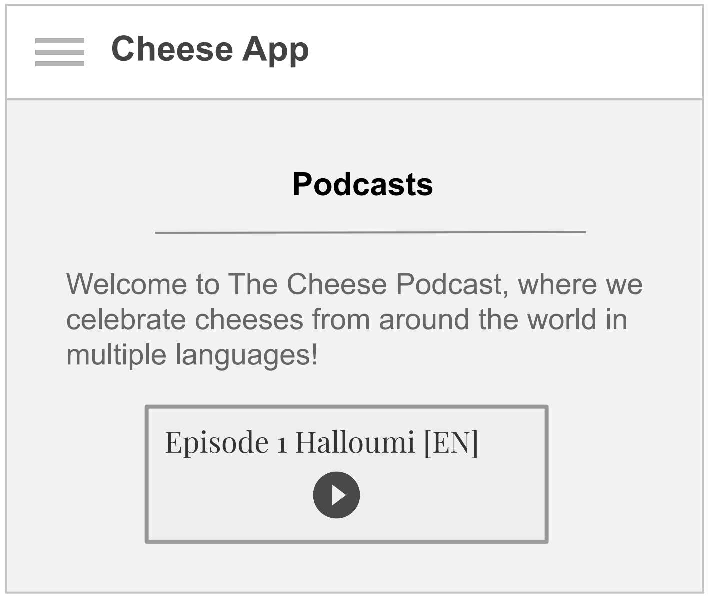
Screenflow & Wireframes



Cheese App

About Us

Welcome to Formaggio.me, a web application born out of a passion for both cheese and cutting-edge technology. This site was created as part of a demonstration project for developing applications using large language models (AI).

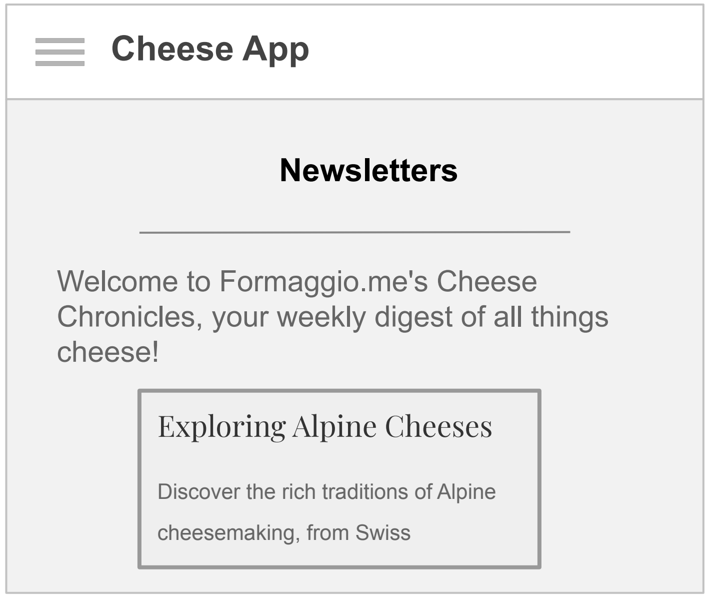


Cheese App

Podcasts

Welcome to The Cheese Podcast, where we celebrate cheeses from around the world in multiple languages!

Episode 1 Halloumi [EN]



Cheese App

Newsletters

Welcome to Formaggio.me's Cheese Chronicles, your weekly digest of all things cheese!

Exploring Alpine Cheeses

Discover the rich traditions of Alpine cheesemaking, from Swiss

Outline

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)

- User actions
- Admin tasks
- Data Scientist tasks
- Developer tasks

Execution (Code)

- Frontend apps
- Backend services
- Data science notebooks
- ML tasks
- LLM APIs
- Agents

State (Source, Data, Models)

- Source control
- Artifact registry
- Data store
- Model registry
- Knowledge base

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
- Chat with cheese expert

Data Scientists

Execution (Code)

Frontend

Backend

- Take an image and apply the same preprocessing
- Use the best model to make prediction
- Call LLM for chat
- RAG for QA
- Newsletters
- Podcasts

State (Sou

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

Source Control

Database

Data / Models

Solution Architecture

Process

Execution

State

Solution Architecture

Process

Develop App

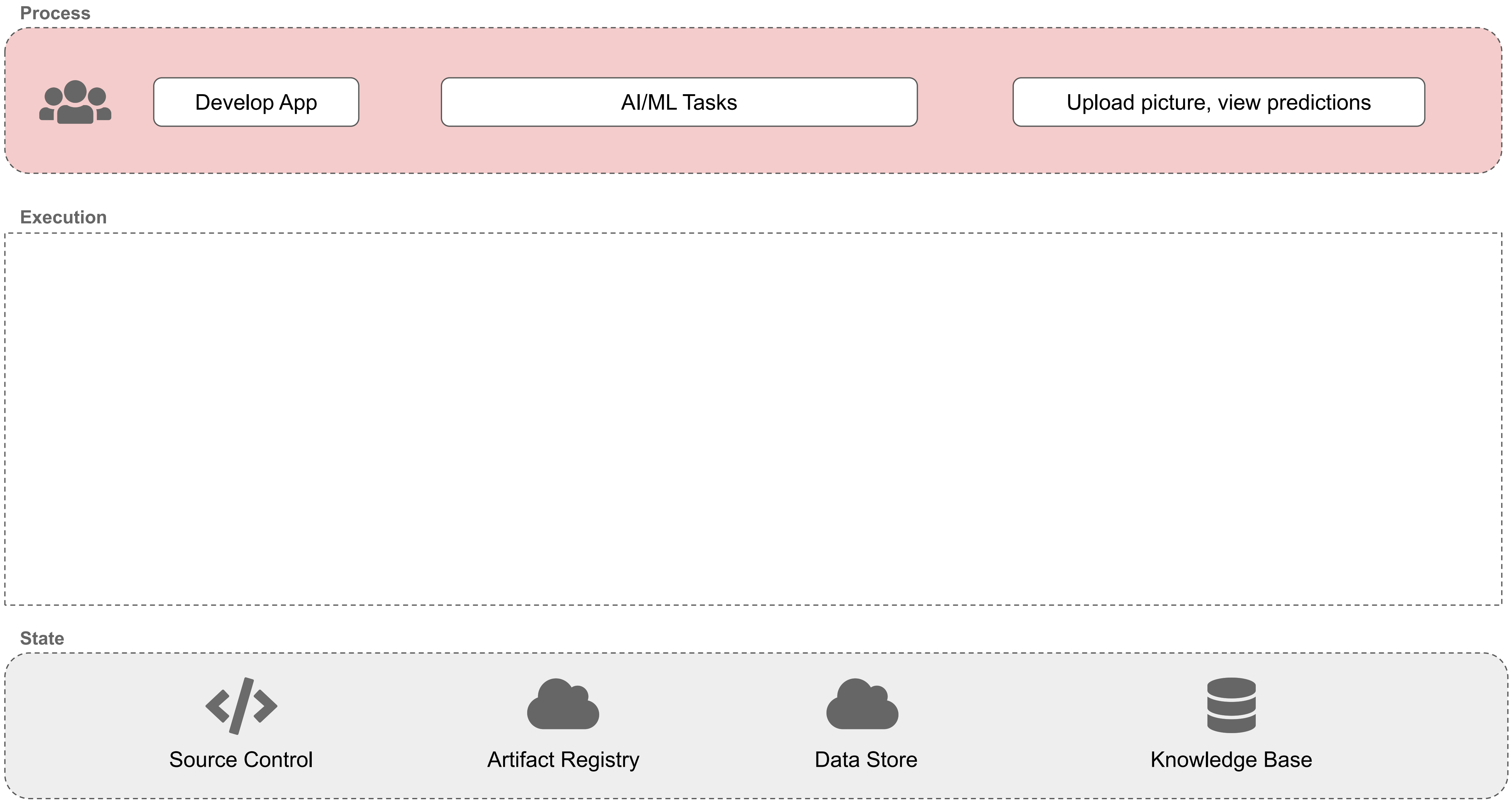
AI/ML Tasks

Upload picture, chat, etc.

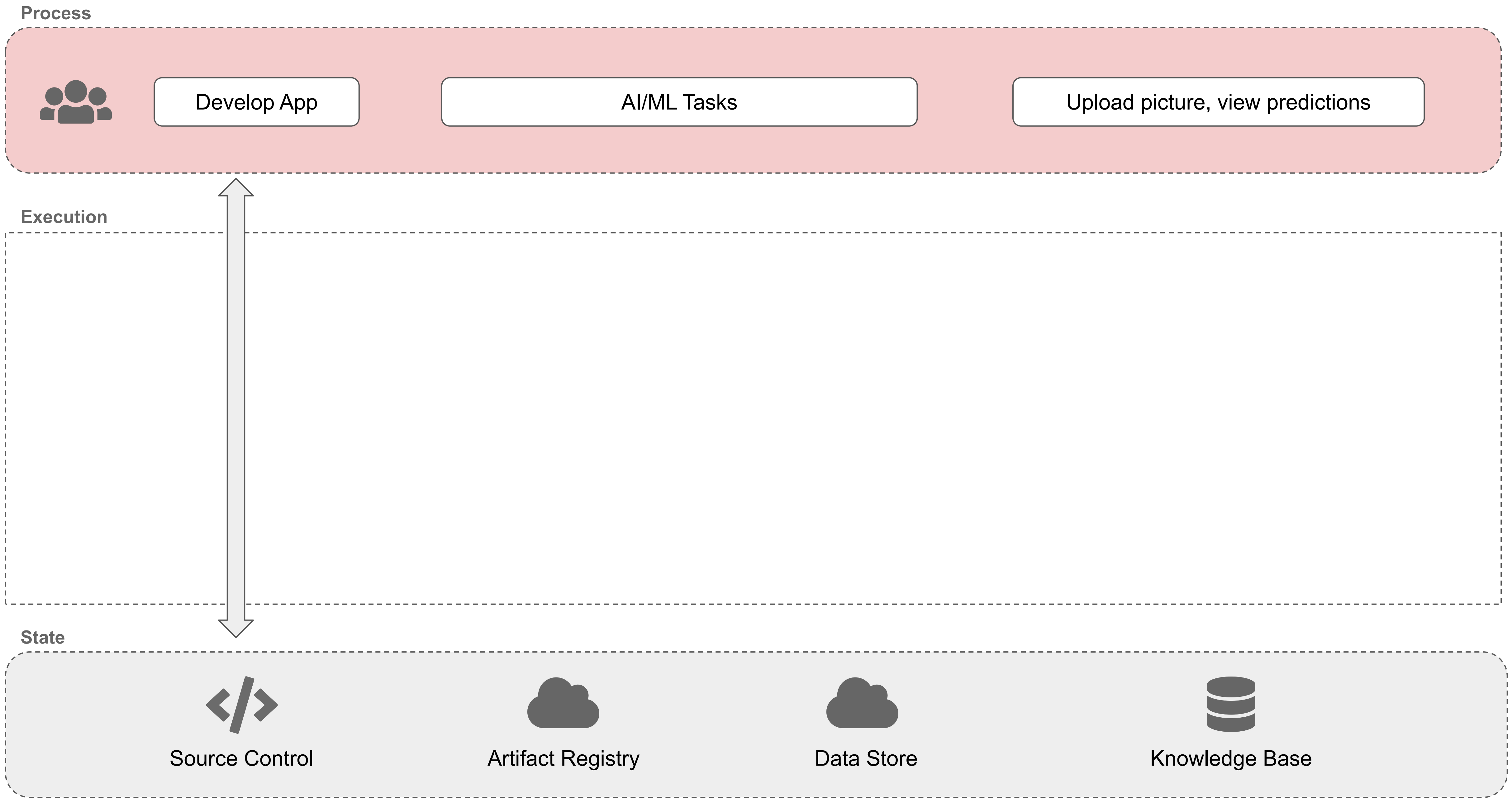
Execution

State

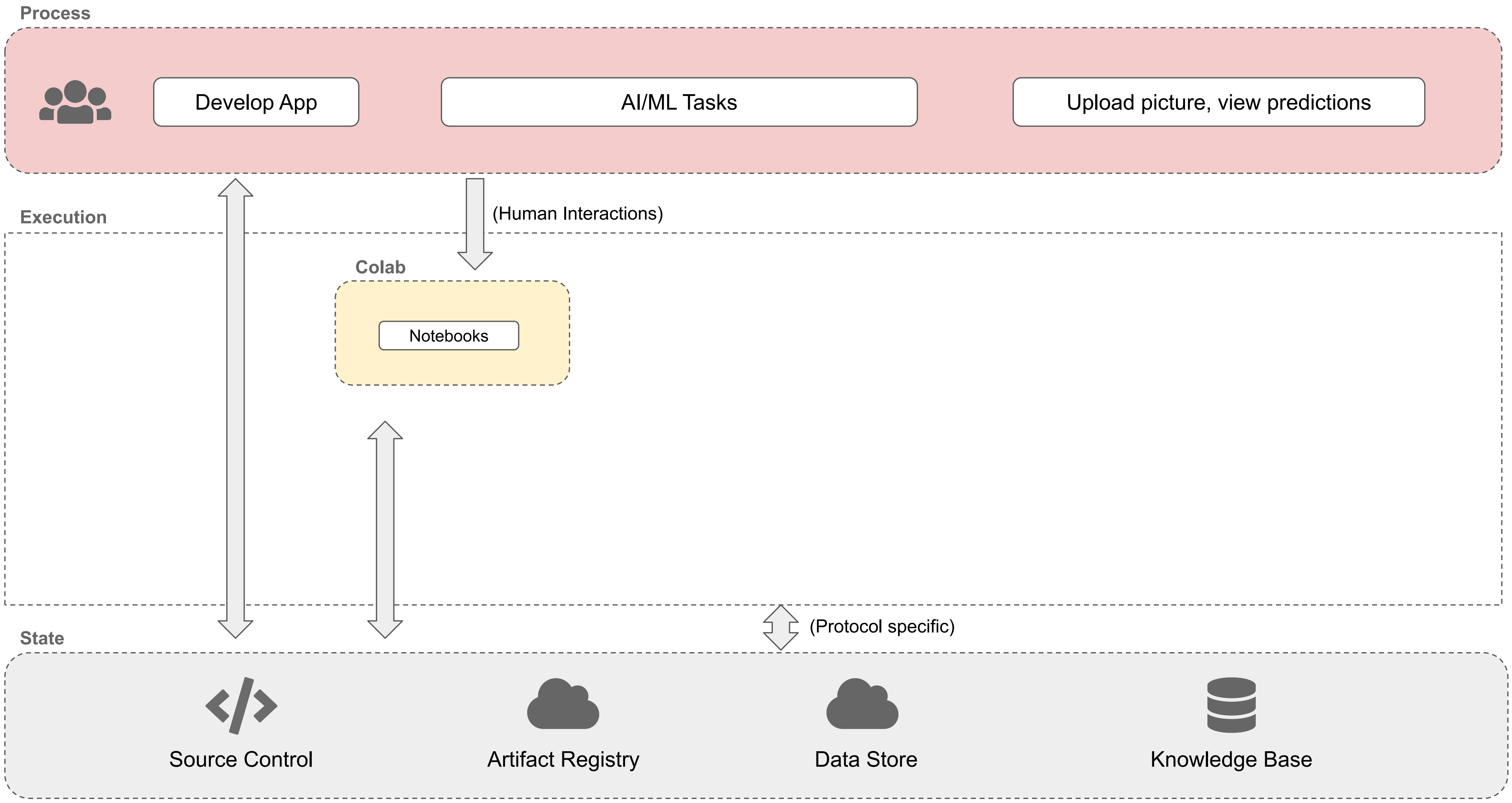
Solution Architecture



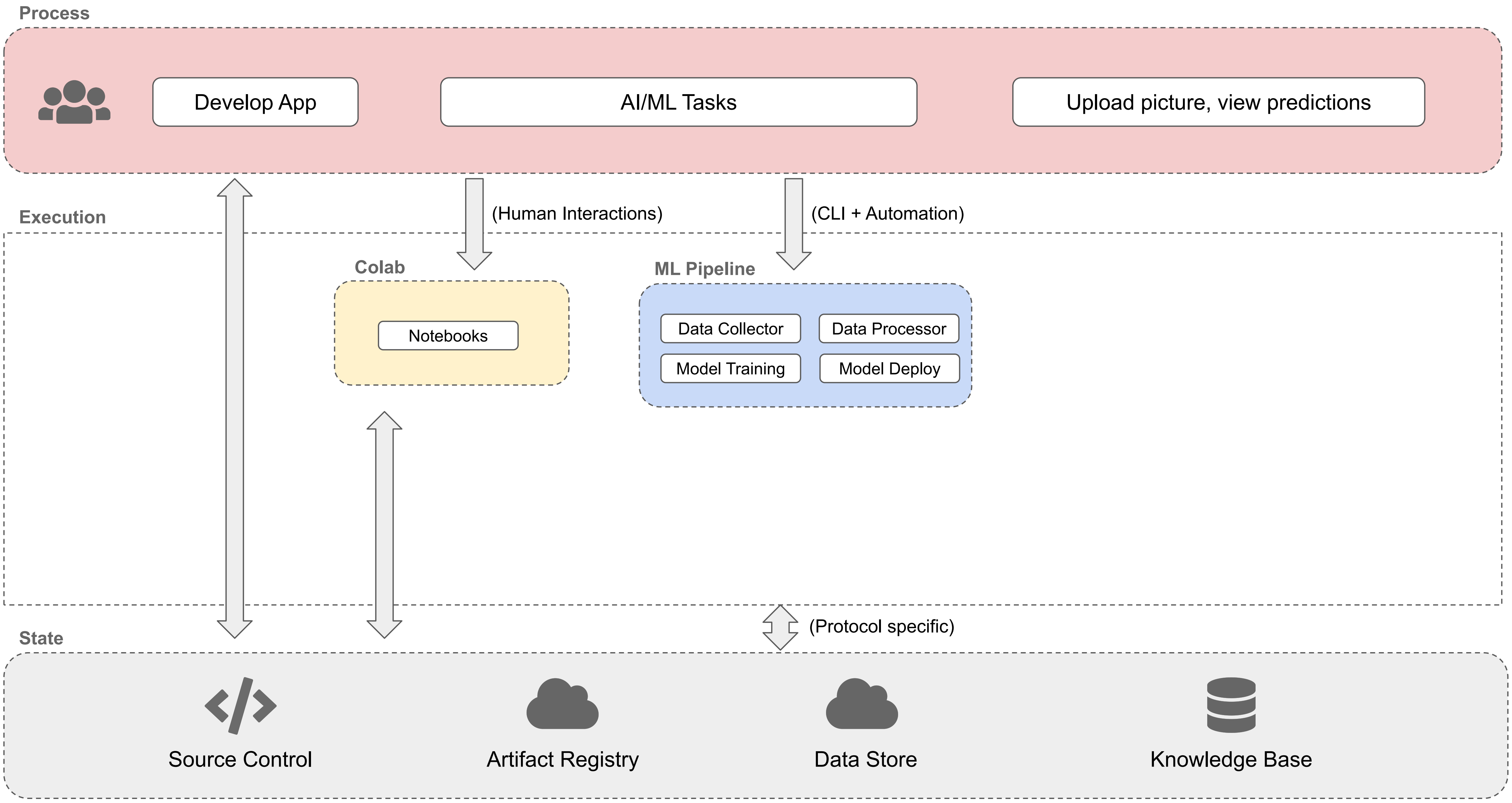
Solution Architecture



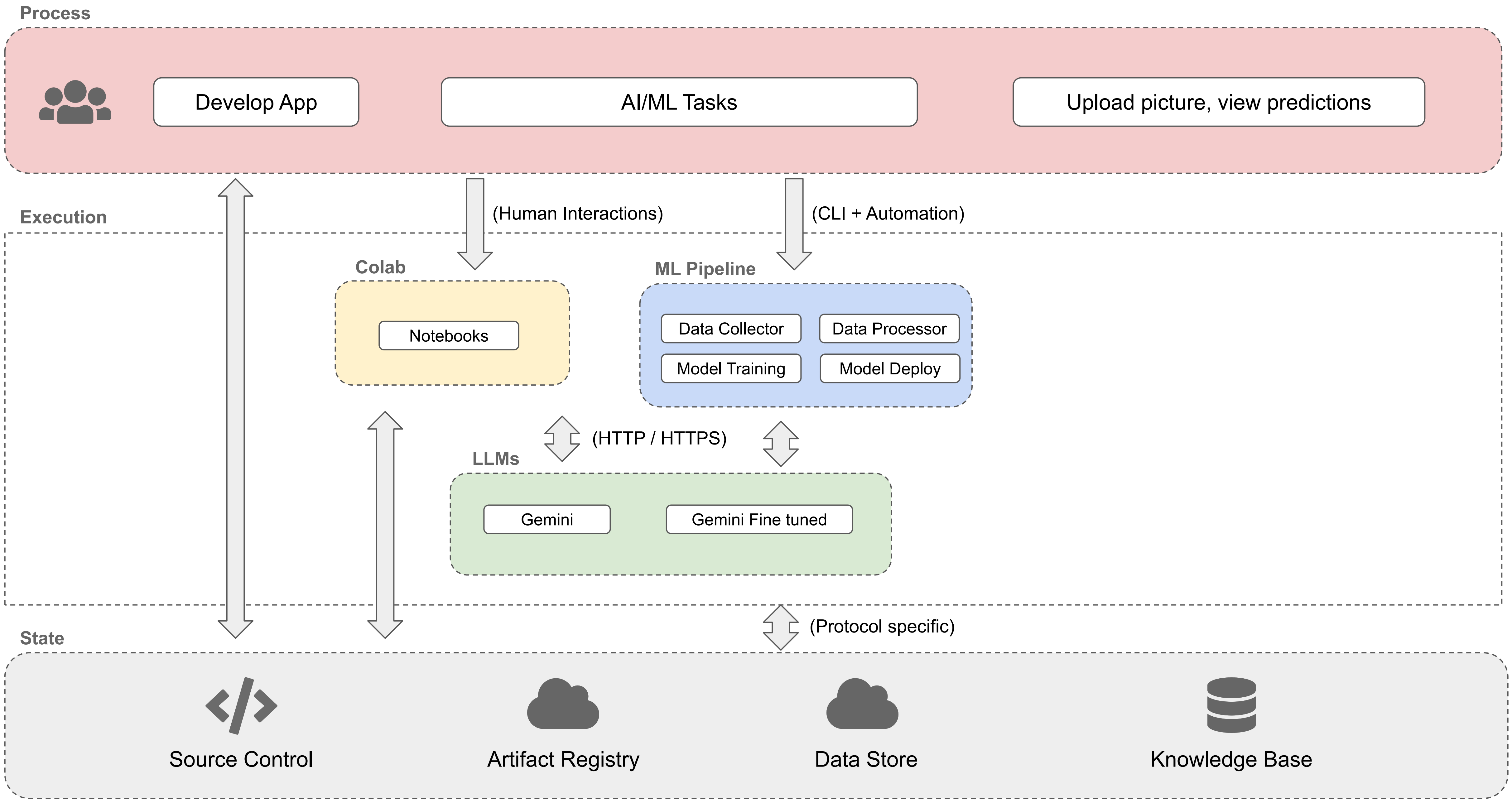
Solution Architecture



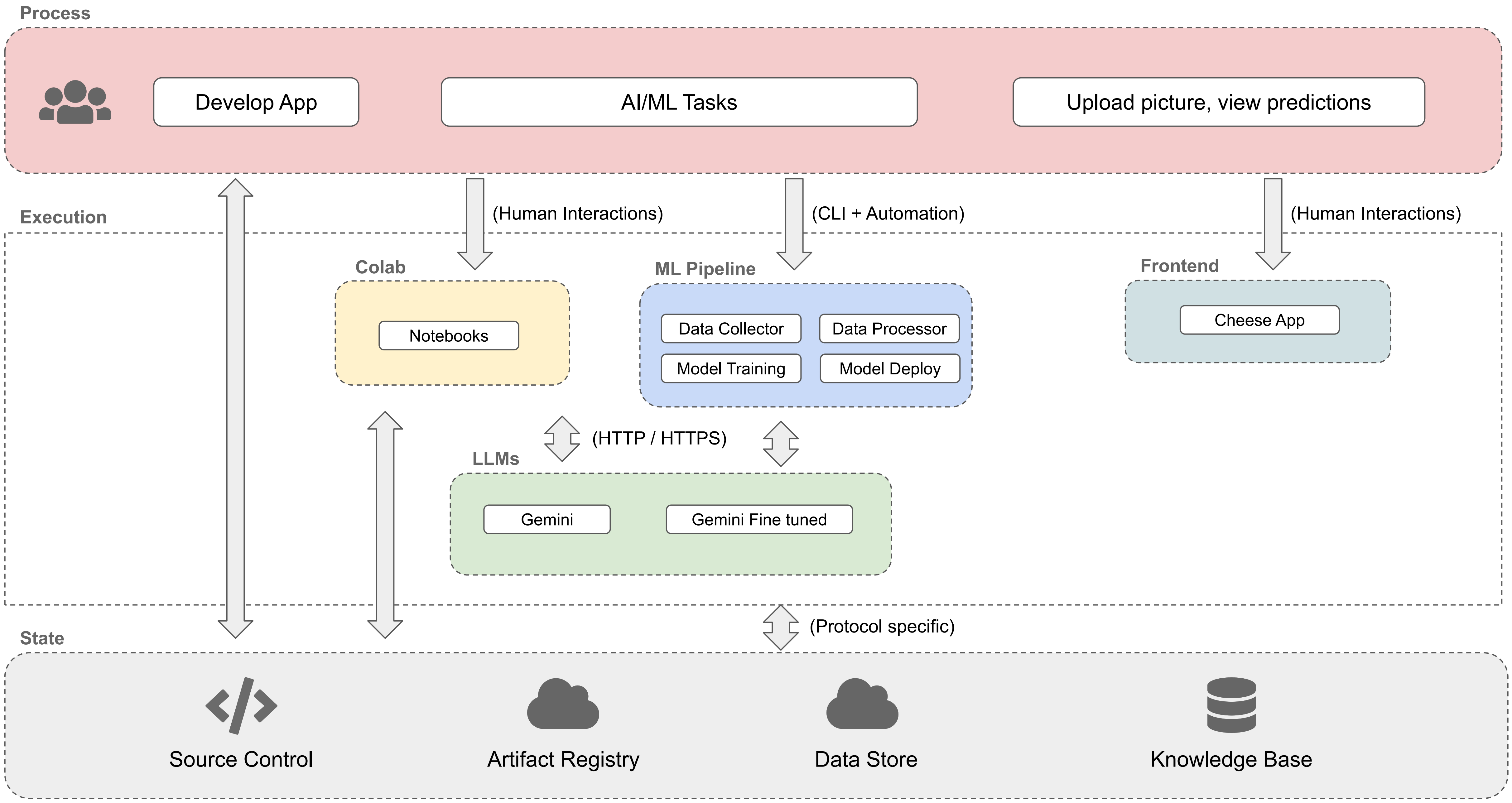
Solution Architecture



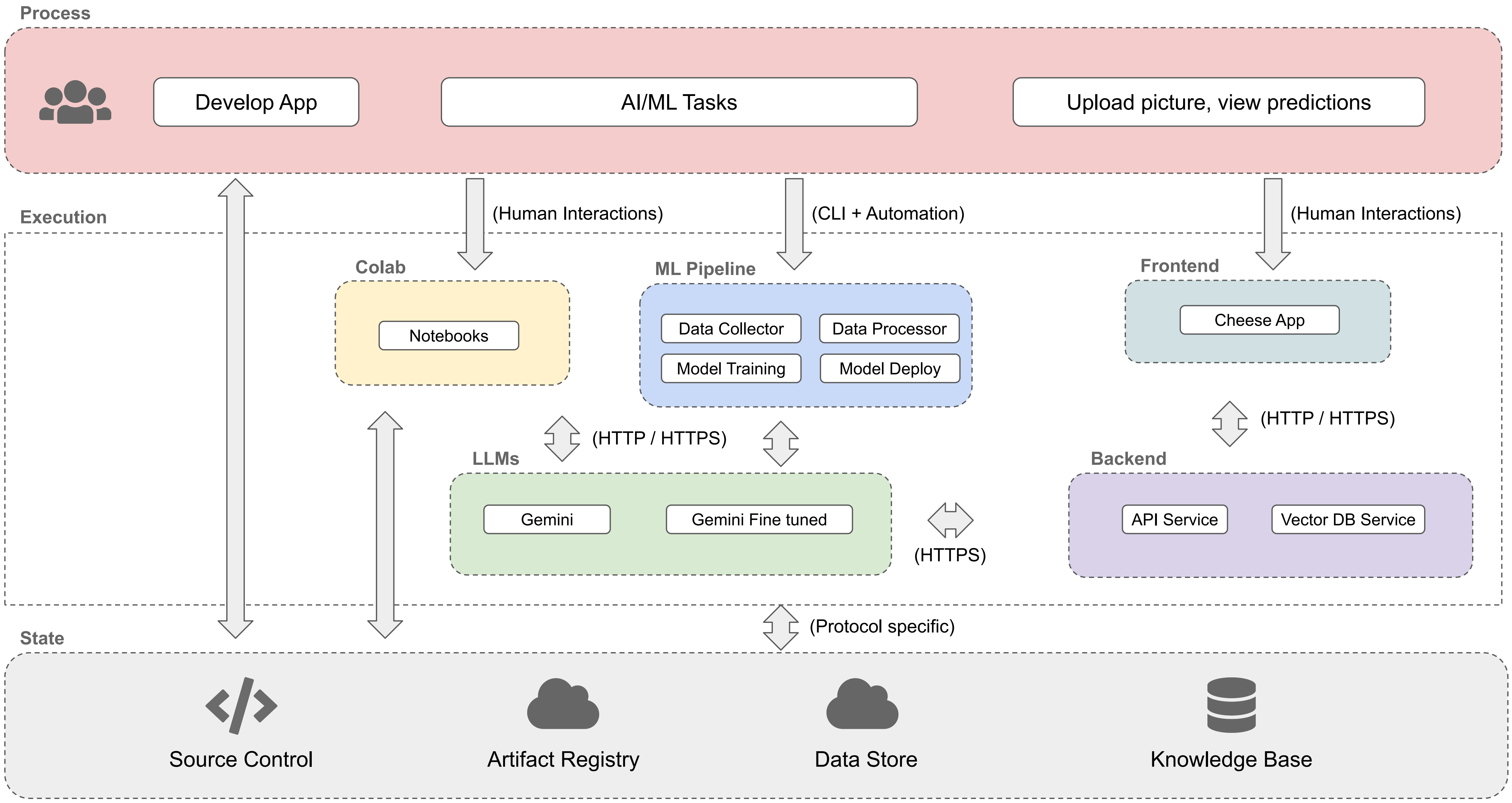
Solution Architecture



Solution Architecture



Solution Architecture



Solution Architecture Summary

- **Process**
  - Data Scientists perform ML Tasks
  - Developers build App
  - Users can upload pictures and have a chat conversation
- **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 and chat with backend
- **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

Outline

Technical Architecture

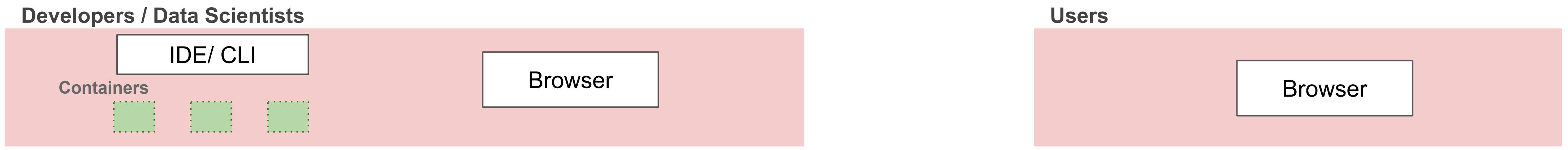
- Helps design and develop an **AI App**
- High level view from **development** to **deployment**
- Illustrates **interactions** between components/**containers**
- **Blueprint** of the system
    - Helps team members understand the big picture
    - Helps onboarding new team members

Building a Technical Architecture

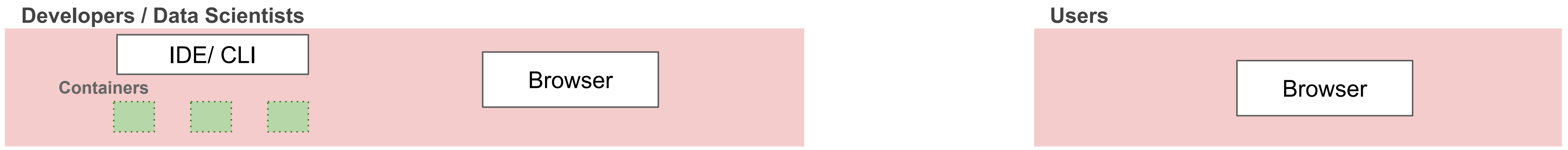
Developers / Data Scientists

Users

Building a Technical Architecture



Building a Technical Architecture



Developers:

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

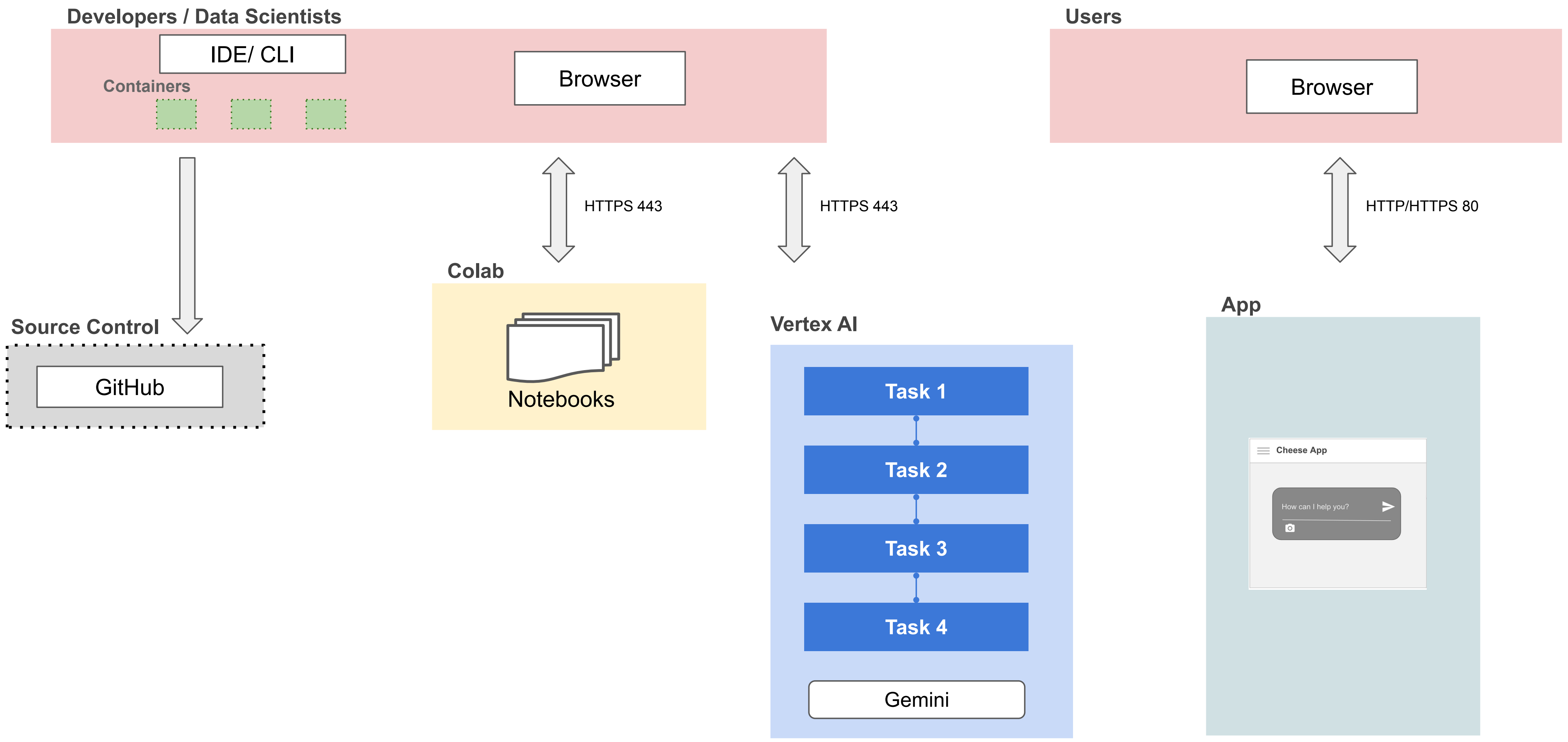
Data Scientists:

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

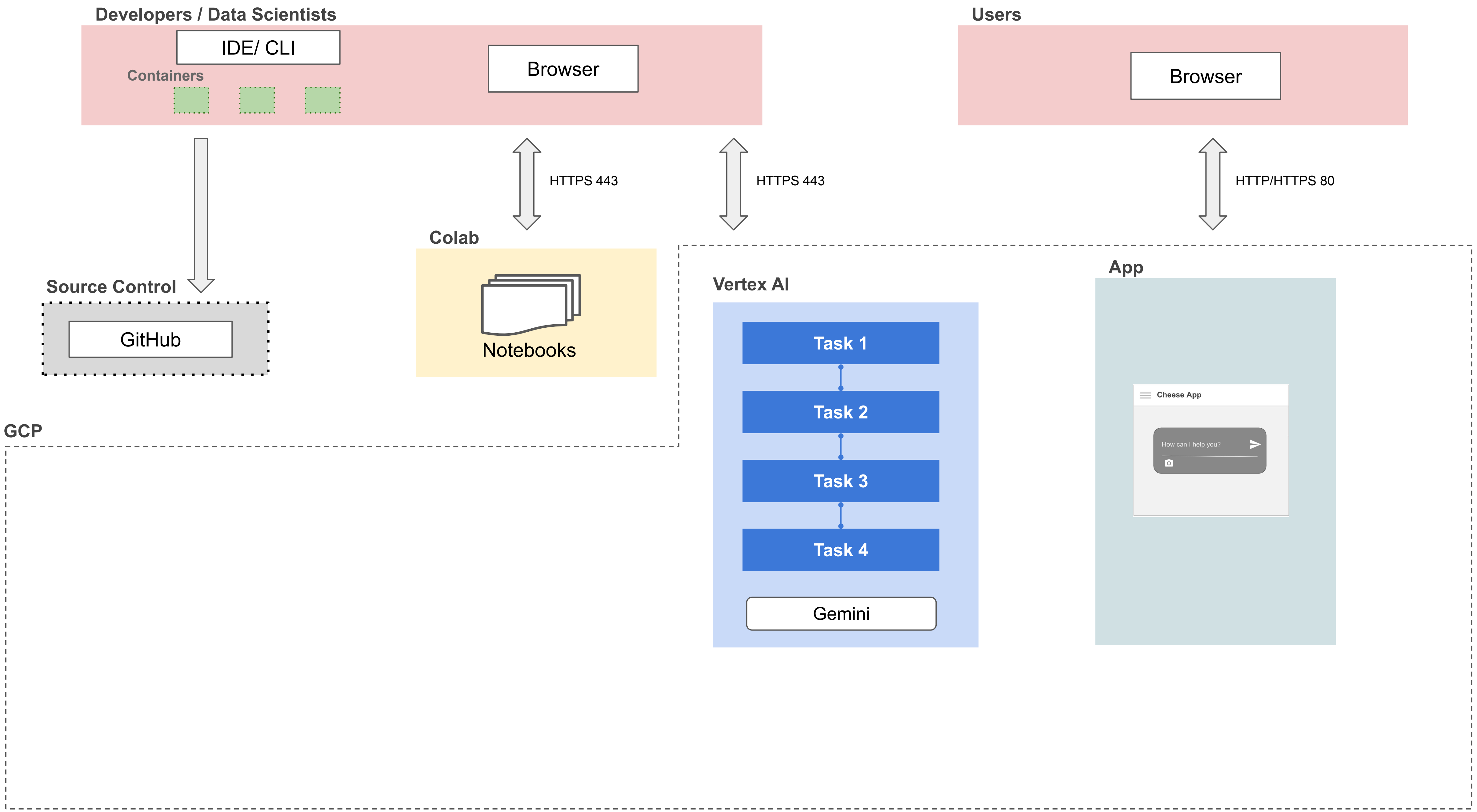
Users:

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

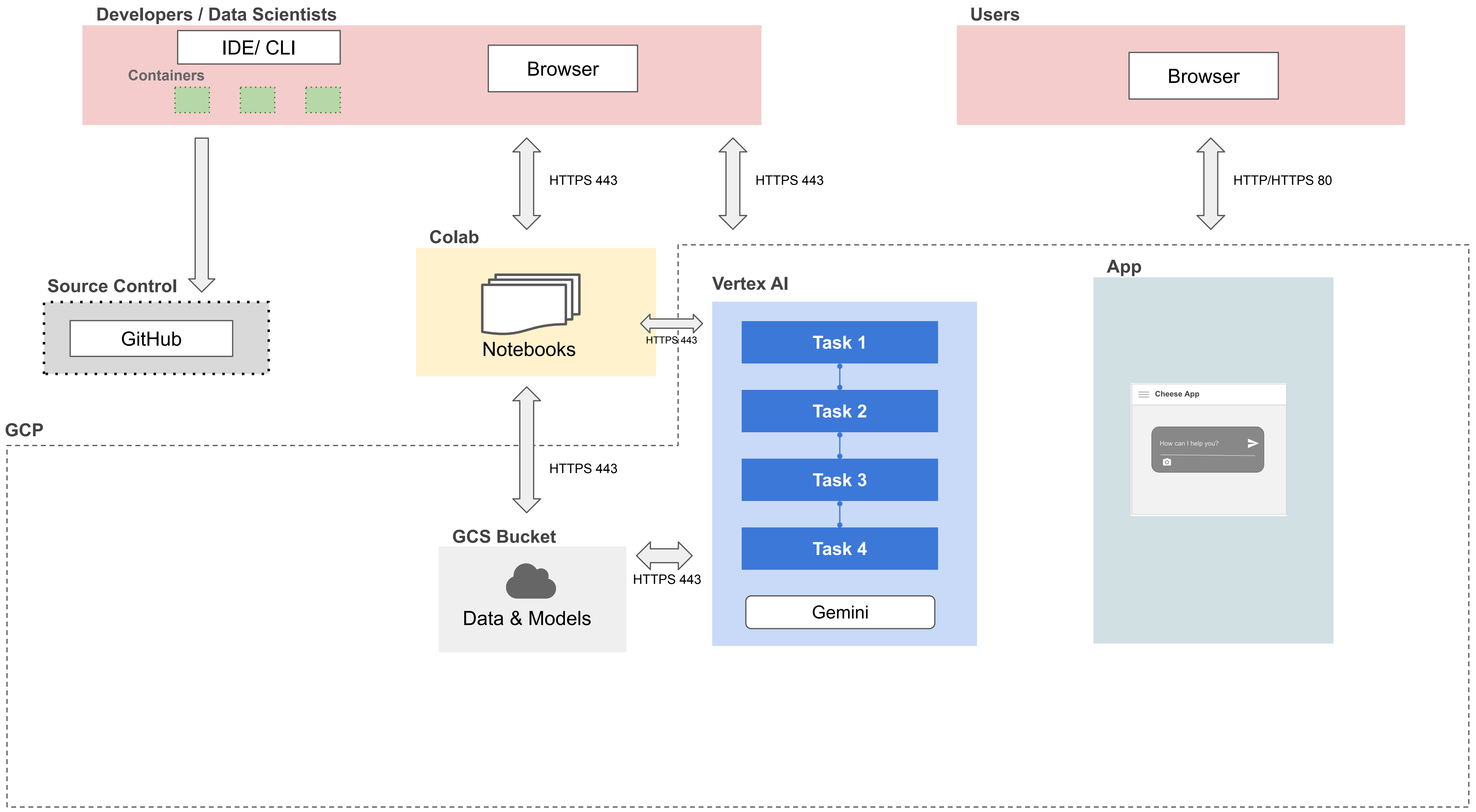
Building a Technical Architecture



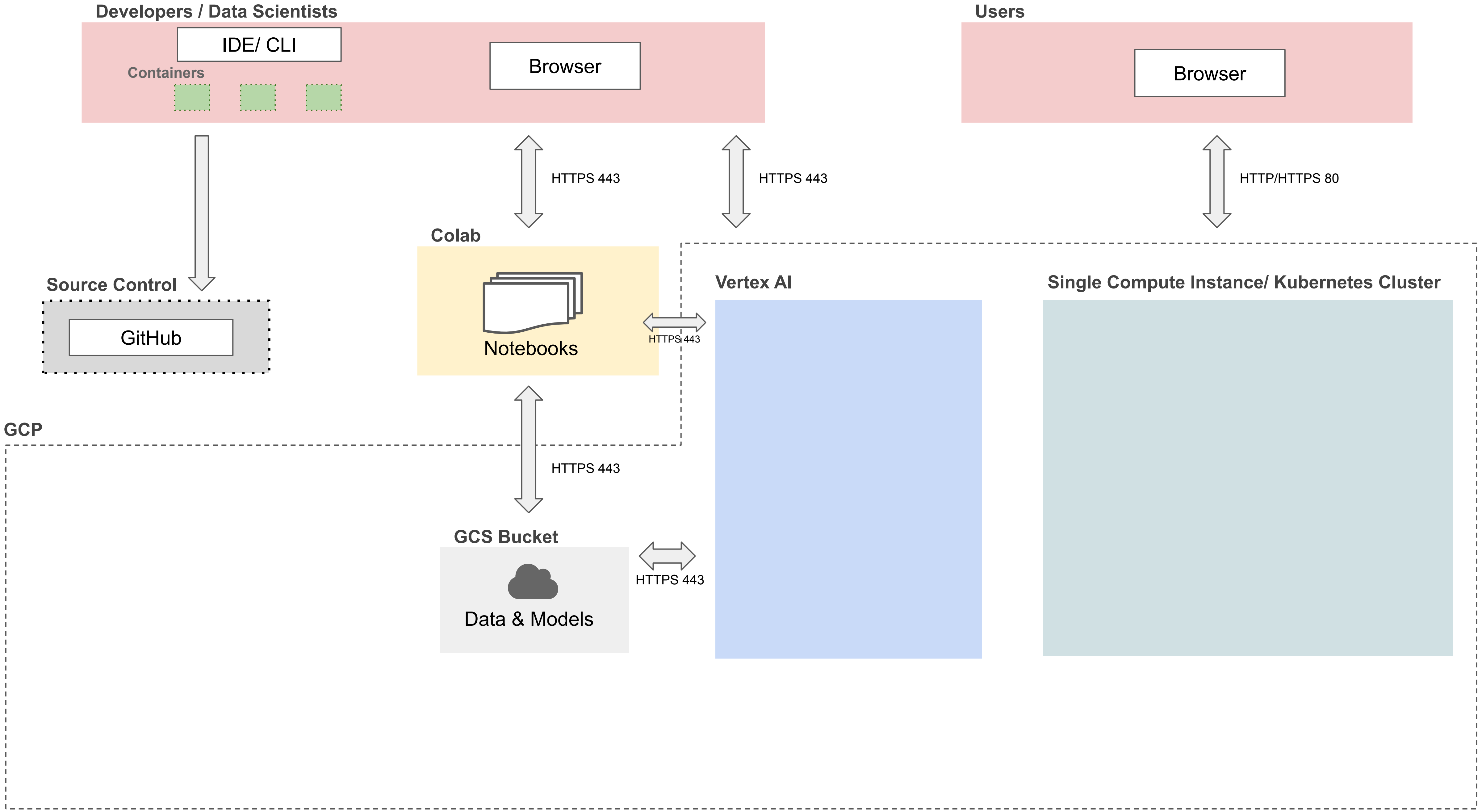
Building a Technical Architecture



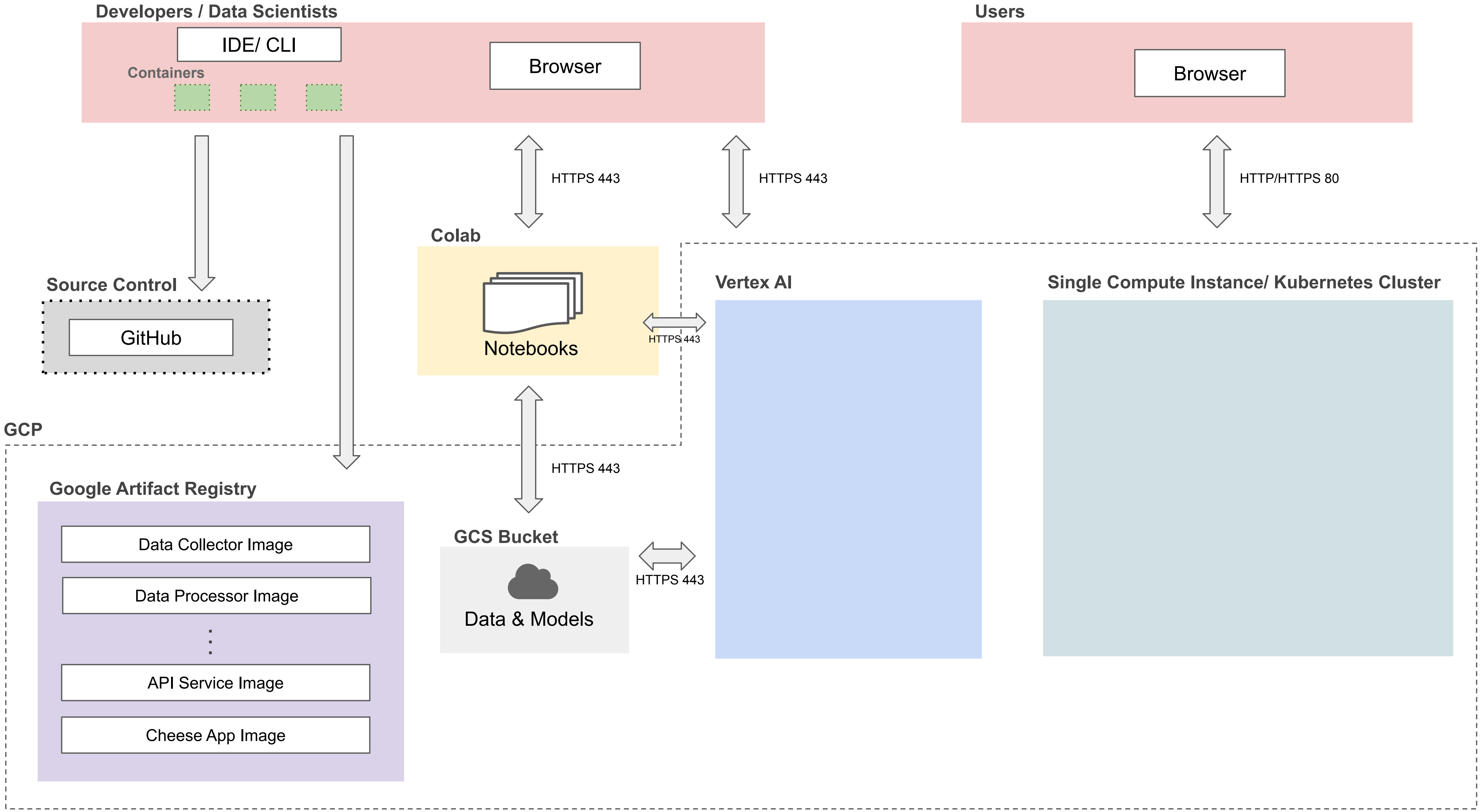
Building a Technical Architecture



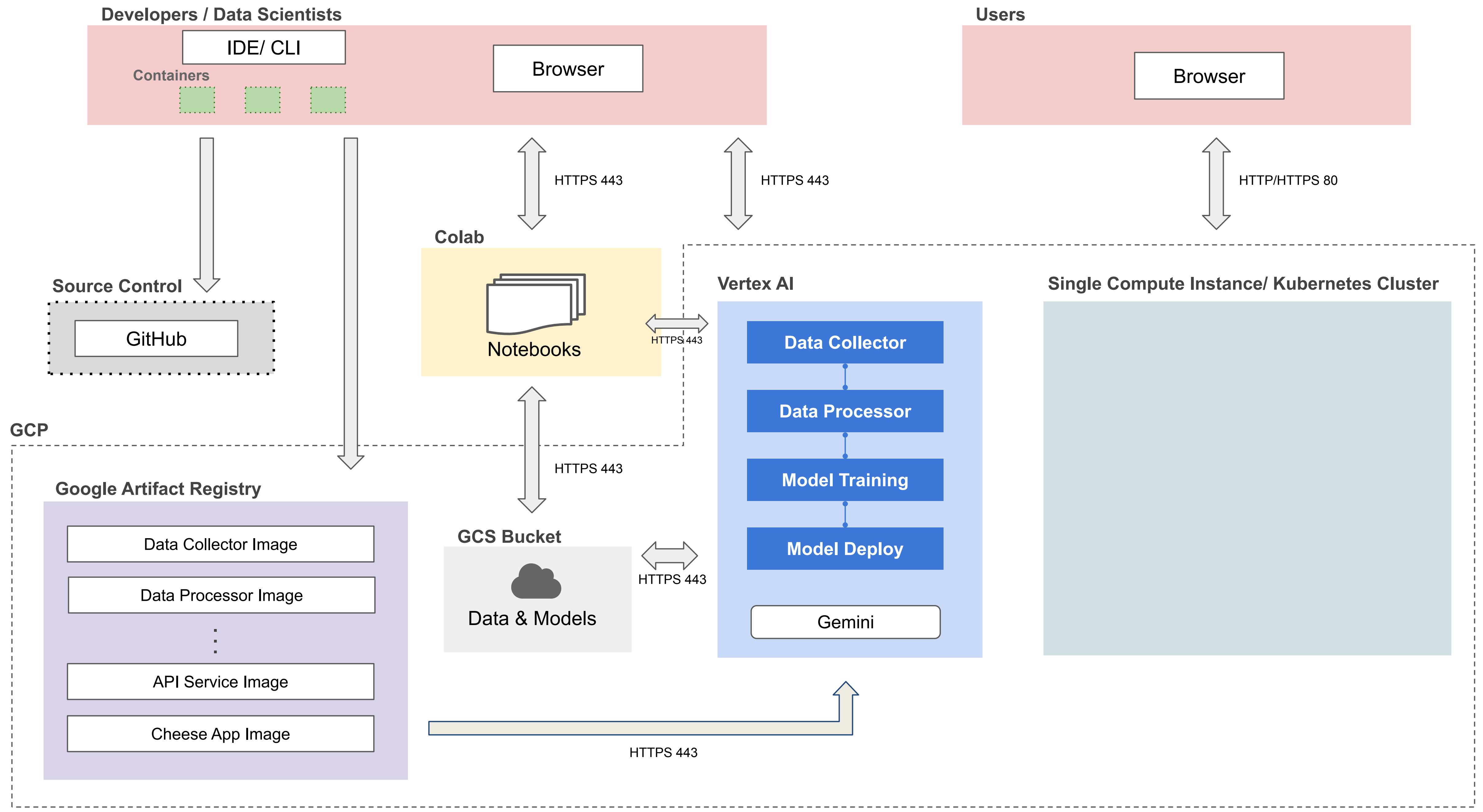
Building a Technical Architecture



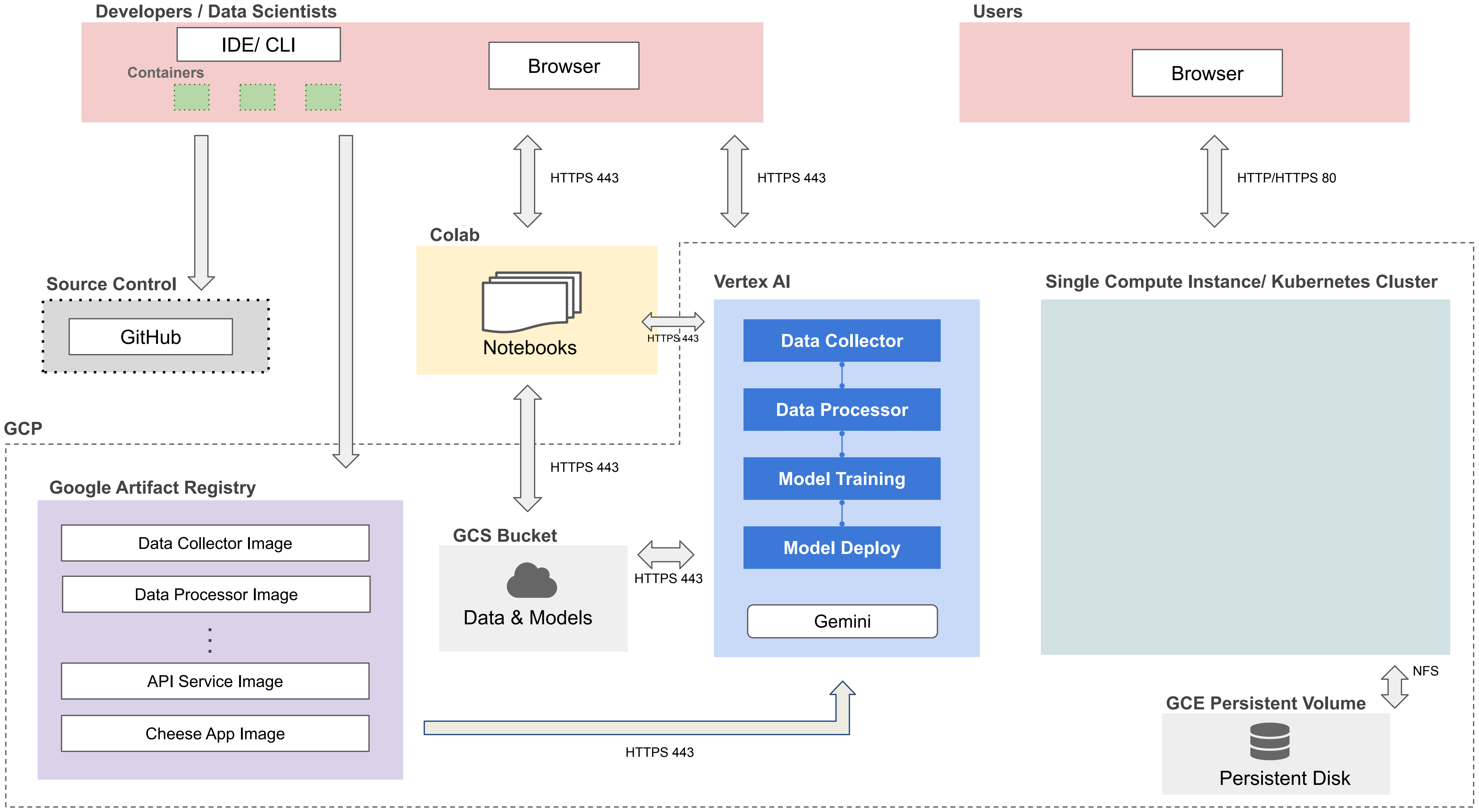
Building a Technical Architecture



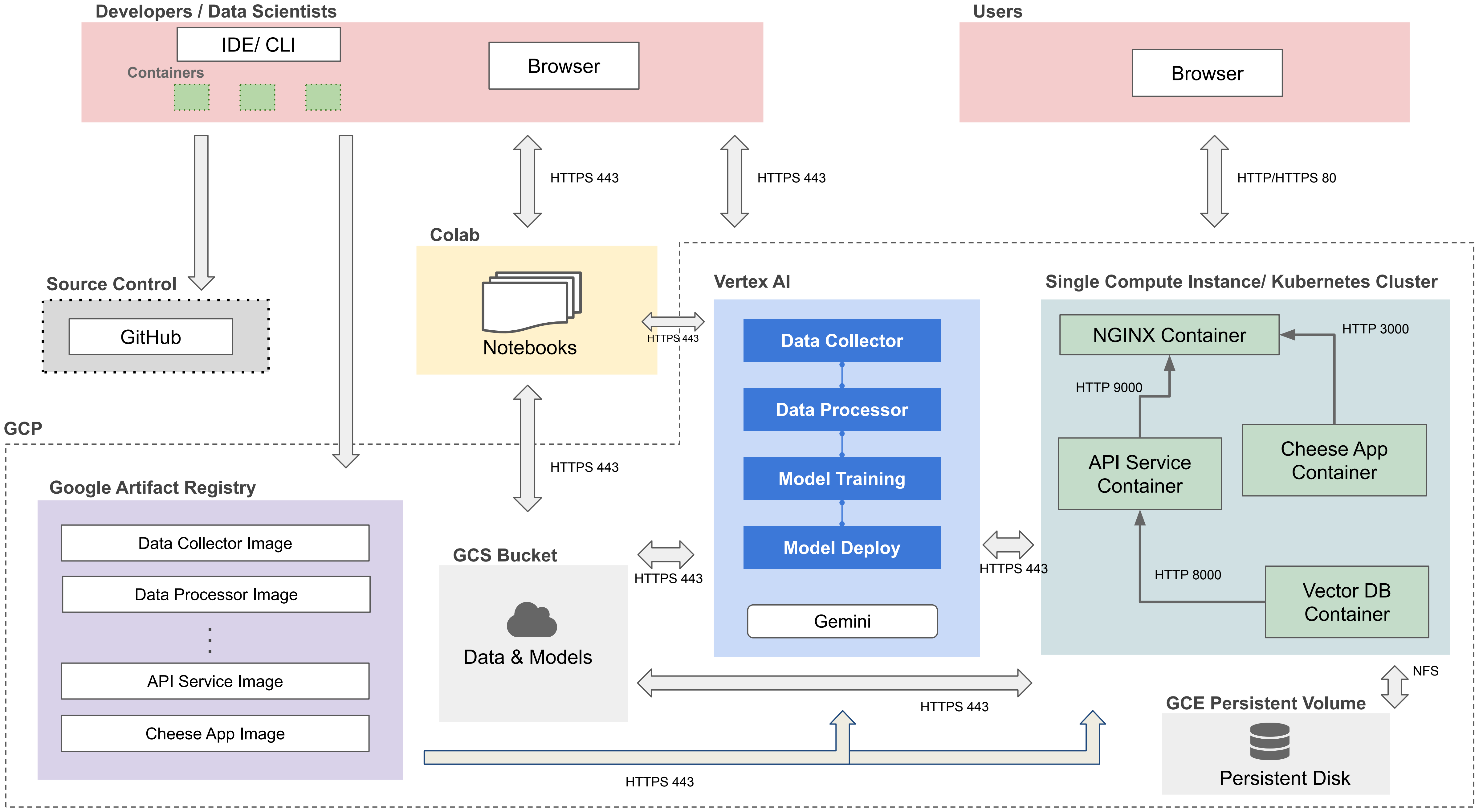
Building a Technical Architecture



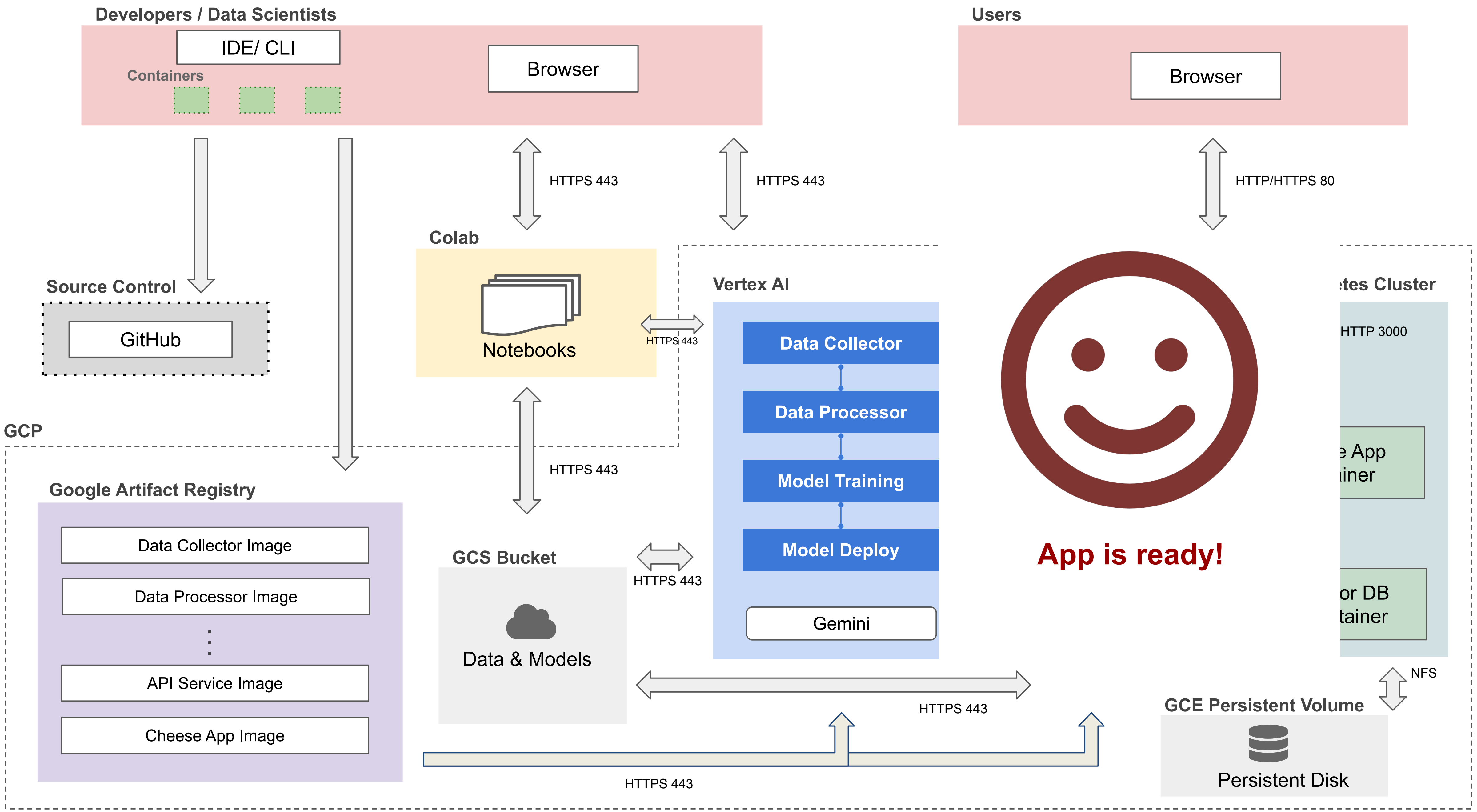
Building a Technical Architecture



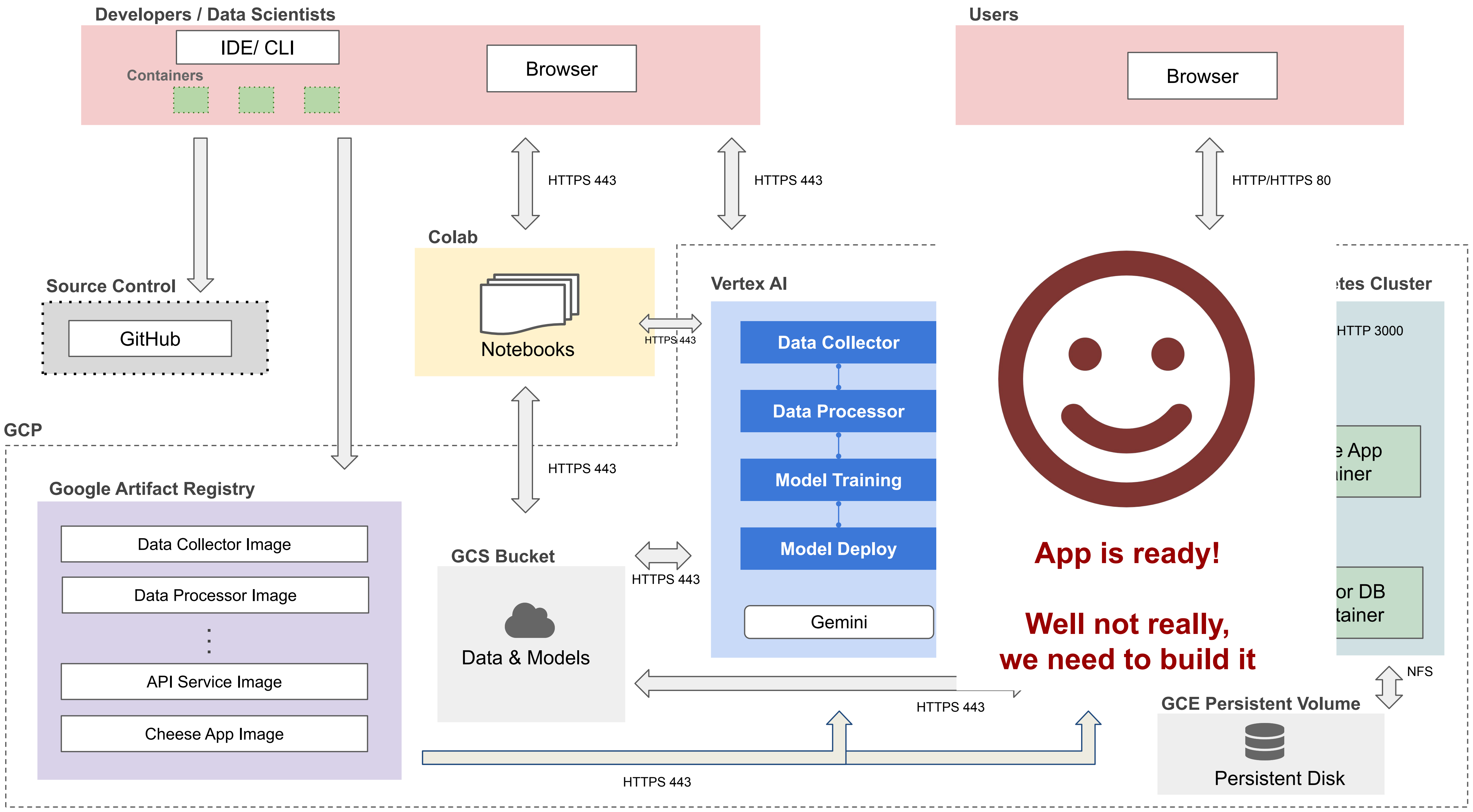
Technical Architecture



Technical Architecture



Technical Architecture



App is ready!

Well not really,
we need to build it

Technical Architecture Summary

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

- **Vertex AI**
  - Serverless ML Tasks
  - Gemini LLM
- **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 scalable the app on GCP

Tutorial: Building Solution Architecture

Steps to build a **Solution Architecture**

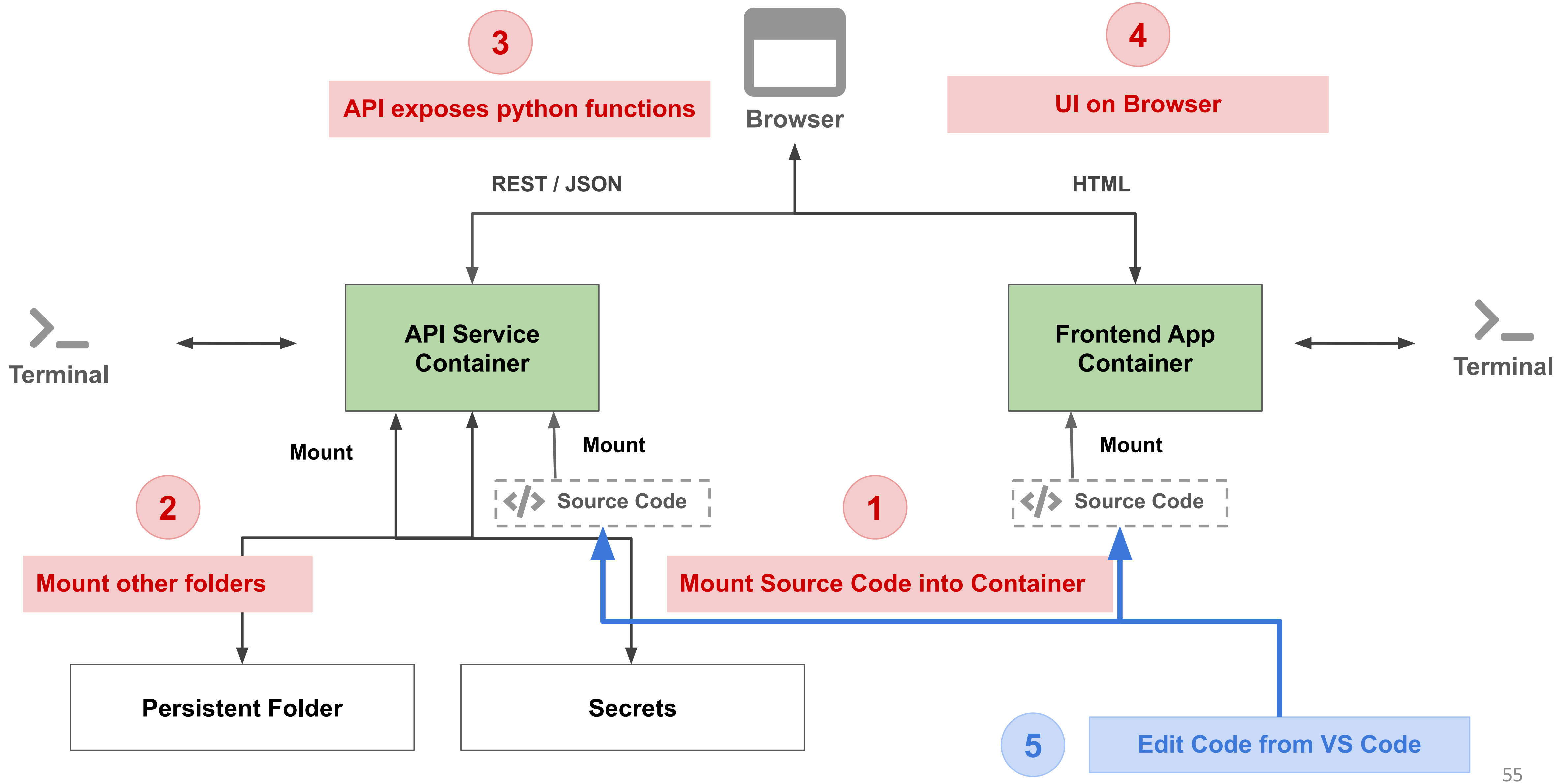
- You will work with your project group
- Go to

  https://docs.google.com/presentation/d/1lBnVcjT4tlShJThe-yhfggGuvOMmZNyPuSQJ-b7WUnE/edit?usp=sharing .
- Duplicate Slides 2,3 to the end of the slides.
- 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**

Setup & Code Organization



Tutorial: Setup & Code Organization

Cheese App - Setup & Code Organization

THANK YOU