

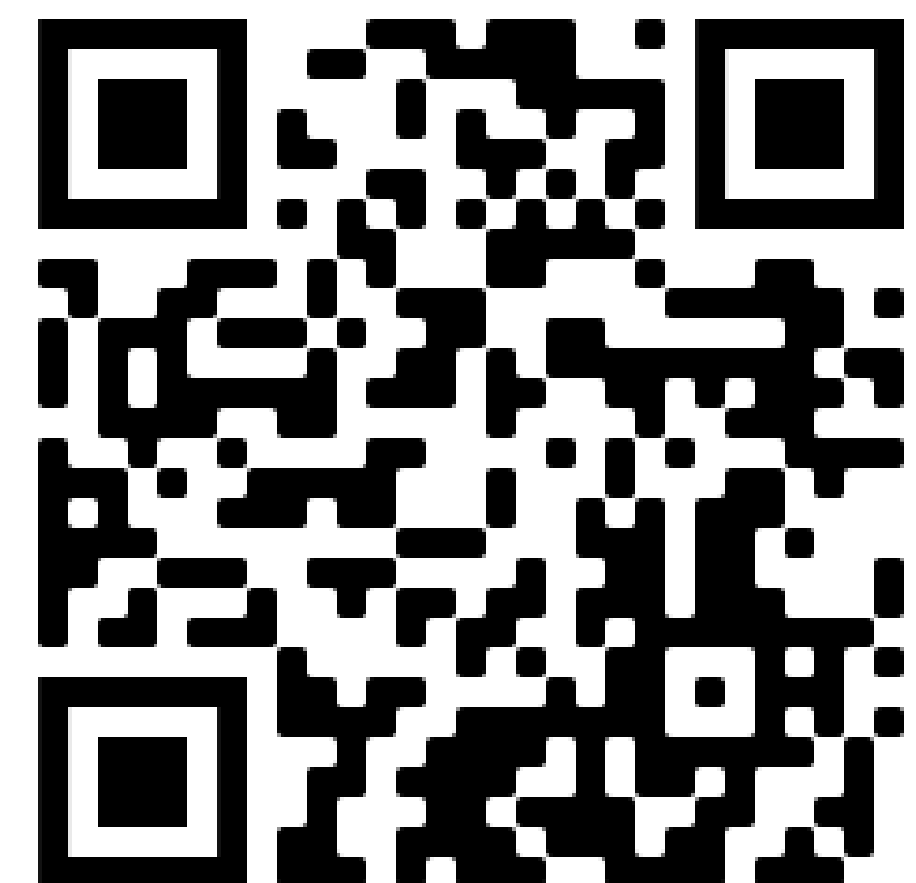
Generative AI Innovative Applications

NVIDIA Building RAG Agents with LLMs Part II

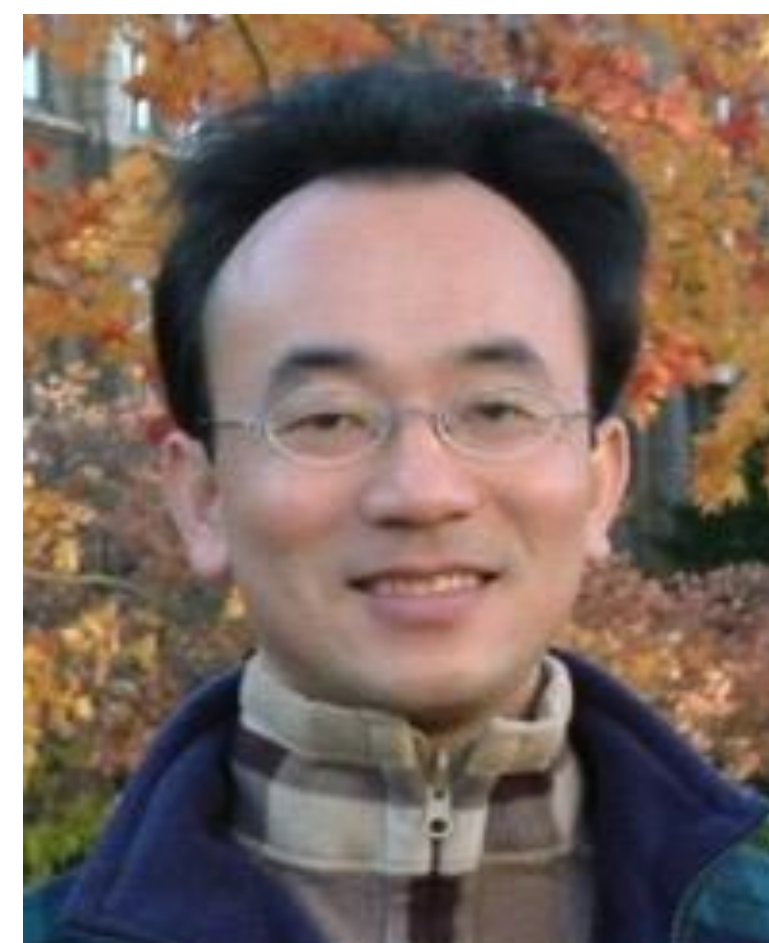
1132GAIIA04

MBA, IM, NTPU (M6031) (Spring 2025)

Tue 2, 3, 4 (9:10-12:00) (B3F17)



<https://meet.google.com/pai-zhhj-mya>

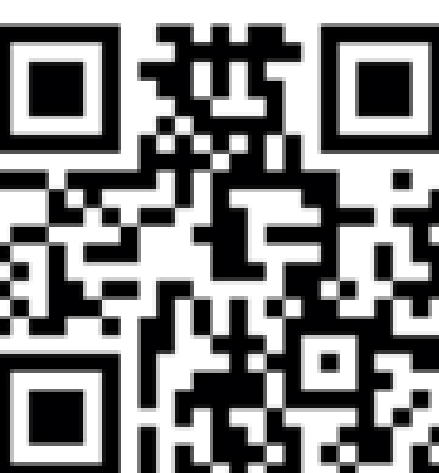


Min-Yuh Day, Ph.D.,
Professor

Institute of Information Management, National Taipei University

<https://web.ntpu.edu.tw/~myday>

2025-03-18



Syllabus

Week Date Subject/Topics

1 2025/02/18 Introduction to Generative AI Innovative Applications

**2 2025/02/25 Transformers for Natural Language Processing and
Computer Vision**

**3 2025/03/04 Large Language Models (LLMs),
NVIDIA Building RAG Agents with LLMs Part I**

4 2025/03/11 Case Study on Generative AI Innovative Applications I

5 2025/03/18 NVIDIA Building RAG Agents with LLMs Part II

6 2025/03/25 NVIDIA Building RAG Agents with LLMs Part III

Syllabus

Week Date Subject/Topics

7 2025/04/01 Self-Learning

8 2025/04/08 Midterm Project Report

9 2025/04/15 Generative AI for Multimodal Information Generation

10 2025/04/22 NVIDIA Generative AI with Diffusion Models Part I

11 2025/04/29 NVIDIA Generative AI with Diffusion Models Part II

12 2025/05/06 Case Study on Generative AI Innovative Applications II

Syllabus

Week Date Subject/Topics

13 2025/05/13 NVIDIA Generative AI with Diffusion Models Part III

14 2025/05/20 AI Agents and Large Multimodal Agents (LMAs)

15 2025/05/27 Final Project Report I

16 2025/06/03 Final Project Report II

NVIDIA Building RAG Agents with LLMs Part II

[0,1,2] [3, 4] [5, 6] [7,8]

Spring 2025

Generative AI Innovative Applications



University Ambassador



This certificate acknowledges that

Min-Yuh Day

has been certified to deliver NVIDIA instructor-led workshop for
academia

A handwritten signature in black ink, appearing to read "Greg A. Estes", written over a horizontal line.

Greg Estes

Vice President, NVIDIA

Issue Date: : March 7, 2025

Ambassador Certification ID: cCFh1ZWWTvqKTq7dcKkEWw

<https://learn.nvidia.com/certificates?id=cCFh1ZWWTvqKTq7dcKkEWw>



Certified Instructor

This certificate acknowledges that

Min-Yuh Day

has been certified to deliver the instructor-led workshop

Building RAG Agents with LLMs

A handwritten signature in black ink, appearing to read "Greg A. Estes", written over a horizontal line.

Greg Estes

Vice President, NVIDIA

Issue Date: : March 7, 2025

Certification ID: OVmqY4cSSya0BdMQBWHxzw

<https://learn.nvidia.com/certificates?id=OVmqY4cSSya0BdMQBWHxzw>

NVIDIA Certified Instructors (12 from Taiwan)

Shop Drivers Support 

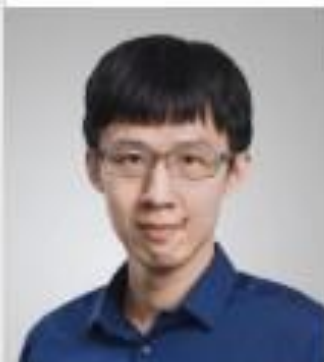
Don't miss NVIDIA CEO Jensen Huang's GTC keynote March 18. Join early for the pregame event. 01d:04h:15m:12s [Add to Calendar](#)

Deep Learning Institute Find Training Self Paced Courses Instructor-Led Workshops Educator Programs ...

Filters

Workshop Certification Location [1] Organization Specialization Reset filters Search for instructors by name. Minimum

Sort by: Name A-Z

Chi-Hung Chuang  Chung Yuan Christian University in Featured Workshop Fundamentals of Deep Learning, Taiwanese	Chia Yu Hsu  National Taiwan University of Science and Technology(NTUST) Featured Workshop Applications of AI for Predictive Maintenance, Taiwanese	Chien-Yu Chen  National Taiwan University (NTU) Featured Workshop Fundamentals of Deep Learning, Taiwanese
Chun-Yi Lee  National Taiwan University (NTU) Platinum Instructor Featured Workshop Fundamentals of Deep Learning, Taiwanese	David Tseng  Cavedu Platinum Instructor Featured Workshop Getting Started with AI on Jetson Nano, Taiwanese	Hsinmin Lu  National Taiwan University (NTU) Featured Workshop Fundamentals of Deep Learning, English
Min-Yuh Day  National Taipei University (NTPU) in Featured Workshop Building RAG Agents with LLMs, English	Ming-Che Chen  Southern Taiwan University of Science and Technology(STUST) Featured Workshop Getting Started with AI on Jetson Nano, Taiwanese	MingChe Hu  Chung Yuan Christian University Featured Workshop Fundamentals of Deep Learning, Taiwanese
Ping-Chun Hsieh  National Yang Ming Chiao Tung University (NYCU) Featured Workshop Fundamentals of Deep Learning, English	Po-Chih Kuo  National Tsing Hua University (NTHU) in Featured Workshop Fundamentals of Deep Learning, English	Shu-Kai Hsieh  National Taiwan University (NTU) in Featured Workshop Fundamentals of Deep Learning, Taiwanese

NVIDIA Developer Program

<https://developer.nvidia.com/join-nvidia-developer-program>

NVIDIA

Deep Learning Institute (DLI)

<https://learn.nvidia.com/>

Get NVIDIA DLI Certificate

- Welcome to NVIDIA DLI "**Building RAG Agents with LLMs**" **workshop** scheduled for **March 11th-25th, 2025** at **NTPU**.
- Step 1. Visit <https://learn.nvidia.com/dli-event>
- Step 2. Enter the event code: **NTPU_RAG_AMBASSADOR_MA25**
- Step 3. Complete the NVIDIA DLI course and review the course datasheet (on March 25, 2025) at <https://developer.nvidia.com/dli/getready>

Get NVIDIA DLI Certificate Before the NVIDIA Workshop

- **Step 1. Join NVIDIA Developer Program (Free)**
<https://developer.nvidia.com/join-nvidia-developer-program>
- **Step 2. Visit NVIDIA Deep Learning Institute (DLI)**
<https://learn.nvidia.com/>
- **Step 3. Enroll "Building RAG Agents with LLMs" Self-Paced Course (Free)**
https://learn.nvidia.com/courses/course-detail?course_id=course-v1:DLI+S-FX-15+V1

NVIDIA Deep Learning Institute (DLI)

Building RAG Agents with LLMs workshop

March 11th-25th, 2025 at NTPU

Self-Paced Course

**Building RAG Agents With
LLMs**

Certificate available
Free
8 hours

Instructor-Led Workshop

**Building RAG Agents With
LLMs**

Certificate available
\$500
8 hours

Step 1.

Visit

<https://learn.nvidia.com/dli-event>

Step 2.

Enter the event code:

NTPU_RAG_AMBASSADOR_MA25



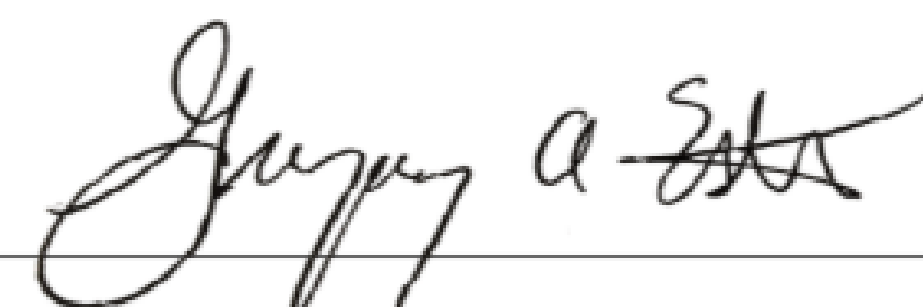
Certificate of Competency

This certificate is awarded to

Min-Yuh Day

for demonstrating competence in the completion of

Building RAG Agents with LLMs



Greg Estes

Vice President, NVIDIA

Issue Date: : December 8, 2024

Certification ID: ed-qOCIMQaatzU8SNUNxgw

[Download Certificate](#)[Add to Profile](#)

Join the NVIDIA Developer Program

take one of the
complimentary
technical self-
paced courses
(worth up to \$90)

Generative AI and LLMs Graphics and Simulation Accelerated Computing Data Science Deep Learning

8 hours
Getting Started With Deep Learning
Explore the fundamentals of deep learning by training neural networks and using results to improve performance and capabilities.

2 hours
Modeling Time-Series Data With Recurrent Neural Networks in Keras
Explore how to classify and forecast time-series data using recurrent neural networks (RNNs), such as modeling a patient's health over time.

4 hours
Deploying a Model for Inference at Production Scale
Learn how to deploy your own machine learning models on a GPU server.

8 hours
Building Real-Time Video AI Applications
Gain the knowledge and skills needed to enable the real-time transformation of raw video data from widely deployed camera sensors into deep learning-based insights.

2 hours
Introduction to Graph Neural Networks
Learn the basic concepts, models, and applications of graph neural networks.

4 hours
Introduction to Physics-Informed Machine Learning With Modulus
Learn the various building blocks of NVIDIA Modulus, which turbocharges use cases by building physics-based deep learning models that are 100,000X faster than traditional methods and offers high-fidelity simulation results.

2 hours
Get Started With Highly Accurate Custom ASR for Speech AI
Learn to build, train, fine-tune, and deploy a GPU-accelerated automatic speech recognition (ASR) service with NVIDIA® Riva that includes customized features.

2 hours
Integrating Sensors With NVIDIA DRIVE
Find out how to integrate automotive sensors into your applications using NVIDIA DRIVE®.

<https://developer.nvidia.com/join-nvidia-developer-program>

NVIDIA Deep Learning Institute (DLI)

Self-Paced Course

Generative AI Explained

Free
2 hours

Self-Paced Course

Getting Started With Deep Learning

Certificate available
\$90
8 hours

Instructor-Led Workshop

Fundamentals of Deep Learning

Certificate available
\$500
8 hours

Self-Paced Course

Introduction to Transformer-Based Natural Language Processing

Certificate available
\$30
6 hours

Self-Paced Course

Building RAG Agents With LLMs

Certificate available
Free
8 hours

Instructor-Led Workshop

Building RAG Agents With LLMs

Certificate available
\$500
8 hours

Self-Paced Course

Generative AI with Diffusion Models

Certificate available
\$90
8 hours

Instructor-Led Workshop

Generative AI with Diffusion Models

Certificate available
\$500
8 hours

What do you want to learn today?

Filters

Level +

Format +

Topics -

- ☐ Deep Learning
- ☐ Accelerated Computing
- ☐ Generative AI/LLM
- ☐ Graphics and Simulation
- ☐ OpenUSD
- ☐ Data Science
- ☐ NIMS
- ☐ NIM
- ☐ RAPIDS

Free / Paid +

Language +

Generative AI



Sort by: --

Showing 19 results

Generative AI x

Generative AI

All Courses

Self-paced

Generative AI Explained

Free
02:00

Self-paced

Generative AI with Diffusion Models

\$90
08:00

Instructor-Led

Generative AI with Diffusion Models

08:00

Self-paced

Augment your LLM Using

Self-paced

Introduction to Transformer-

Instructor-Led

Rapid Application

<https://learn.nvidia.com/en-us/training/find-training?q=Generative+AI>

Building RAG Agents with LLMs

Deep Learning Institute

[Find Training](#)

[Self Paced Courses](#)

[Instructor-Led Workshops](#)

[Educator Programs](#)

[Enterprise Solutions](#)

[Certification](#)

[Resources](#)

Self-paced Course

Building RAG Agents with LLMs

Agents powered by large language models (LLMs) have shown great retrieval capability for using tools, looking at documents, and plan their approaches. This course will show you how to deploy an agent system in practice with the flexibility to scale up your system to meet the demands of users and customers.

[About Course](#)

[Objectives](#)

[Topics Covered](#)

[Course Outline](#)

[Stay Informed](#)

[Contact Us](#)

[Continue Learning](#)

About this Course

This course is free for a limited time.

The evolution and adoption of large language models (LLMs) have been nothing short of revolutionary, with retrieval-based systems at the forefront of this technological leap. These models are not just tools for automation; they are partners in enhancing productivity, capable of holding informed conversations by interacting with a vast array of tools and documents. This course is designed for those eager to explore the potential of these systems, focusing on practical deployment and the efficient implementation required to manage the considerable demands of both users and deep learning models. As we delve into the intricacies of LLMs, participants will gain insights into advanced orchestration techniques that include internal reasoning, dialog management, and effective tooling strategies.

Course Details

Duration: 08:00

Price: Free

Level: Technical - Intermediate

Subject: Generative AI/LLM

Language: English

Course Prerequisites:

Introductory deep learning knowledge, with comfort

https://learn.nvidia.com/courses/course-detail?course_id=course-v1:DLI+S-FX-15+V1

Generative AI Explained

Deep Learning Institute

[Find Training](#)

[Self Paced Courses](#)

[Instructor-Led Workshops](#)

[Educator Programs](#)

[Enterprise Solutions](#)

[Certification](#)

[Resources](#)

Self-paced Course

Generative AI Explained

In this no-coding course, learn Generative AI concepts and applications, as well as the challenges and opportunities in this exciting field.

[About Course](#)

[Objectives](#)

[Topics Covered](#)

[Course Outline](#)

[Stay Informed](#)

[Contact Us](#)

[Continue Learning](#)

About this Course

Generative AI describes technologies that are used to generate new content based on a variety of inputs. In recent time, Generative AI involves the use of neural networks to identify patterns and structures within existing data to generate new content. In this course, you will learn Generative AI concepts, applications, as well as the challenges and opportunities in this exciting field.

Learning Objectives

Upon completion, you will have a basic understanding of Generative AI and be able to more effectively use the various tools built on this

Course Details

Duration: 02:00

Price: Free

Level: Technical - Beginner

Subject: Generative AI/LLM

Language: English

https://learn.nvidia.com/courses/course-detail?course_id=course-v1:DLI+S-FX-15+V1

Introduction to Transformer-Based Natural Language Processing

Self-paced Course

Introduction to Transformer-Based Natural Language Processing

Learn how Transformers are used as the building blocks of modern large language models (LLMs). You'll then use these models for various NLP tasks, including text classification, named-entity recognition (NER), author attribution, and question answering.

About Course

Objectives

Topics Covered

Course Outline

Stay Informed

Contact Us

Continue Learning

About this Course

Large Language Models (LLMs), or Transformers, have revolutionized the field of natural language processing (NLP). Driven by recent advancements, applications of NLP and generative AI have exploded in the past decade. With the proliferation of applications like chatbots and intelligent virtual assistants, organizations are infusing their businesses with more interactive human-machine experiences. Understanding how Transformer-based large language models (LLMs) can be used to manipulate, analyze, and generate text-based data is essential. Modern pre-trained LLMs can encapsulate the nuance, context, and sophistication of language, just as humans do. When fine-tuned and deployed correctly, developers can use these LLMs to build powerful NLP applications that provide natural and seamless human-computer interactions within chatbots, AI voice agents, and more. In this course, you'll learn how Transformers are used as the building blocks of modern large language models (LLMs). You'll then use these models for various NLP

Course Details

Duration: 06:00

Price: \$30

Level: Technical - Beginner

Subject: Generative AI/LLM

Language: English

https://learn.nvidia.com/courses/course-detail?course_id=course-v1:DLI+S-FX-08+V1

Generative AI with Diffusion Models

Self-paced Course

Generative AI with Diffusion Models

Take a deeper dive into denoising diffusion models, which are a popular choice for text-to-image pipelines, with applications in creative content generation, data augmentation, simulation and planning, anomaly detection, drug discovery, personalized recommendations, and more.

About Course

Objectives

Topics Covered

Course Outline

Stay Informed

Contact Us

Continue Learning

About this Course

Thanks to improvements in computing power and scientific theory, generative AI is more accessible than ever before. Generative AI plays a significant role across industries due to its numerous applications, such as creative content generation, data augmentation, simulation and planning, anomaly detection, drug discovery, personalized recommendations, and more. In this course, learners will take a deeper dive into denoising diffusion models, which are a popular choice for text-to-image pipelines.

Learning Objectives

Course Details

Duration: 08:00

Price: \$90

Subject: Generative AI/LLM

Language: English

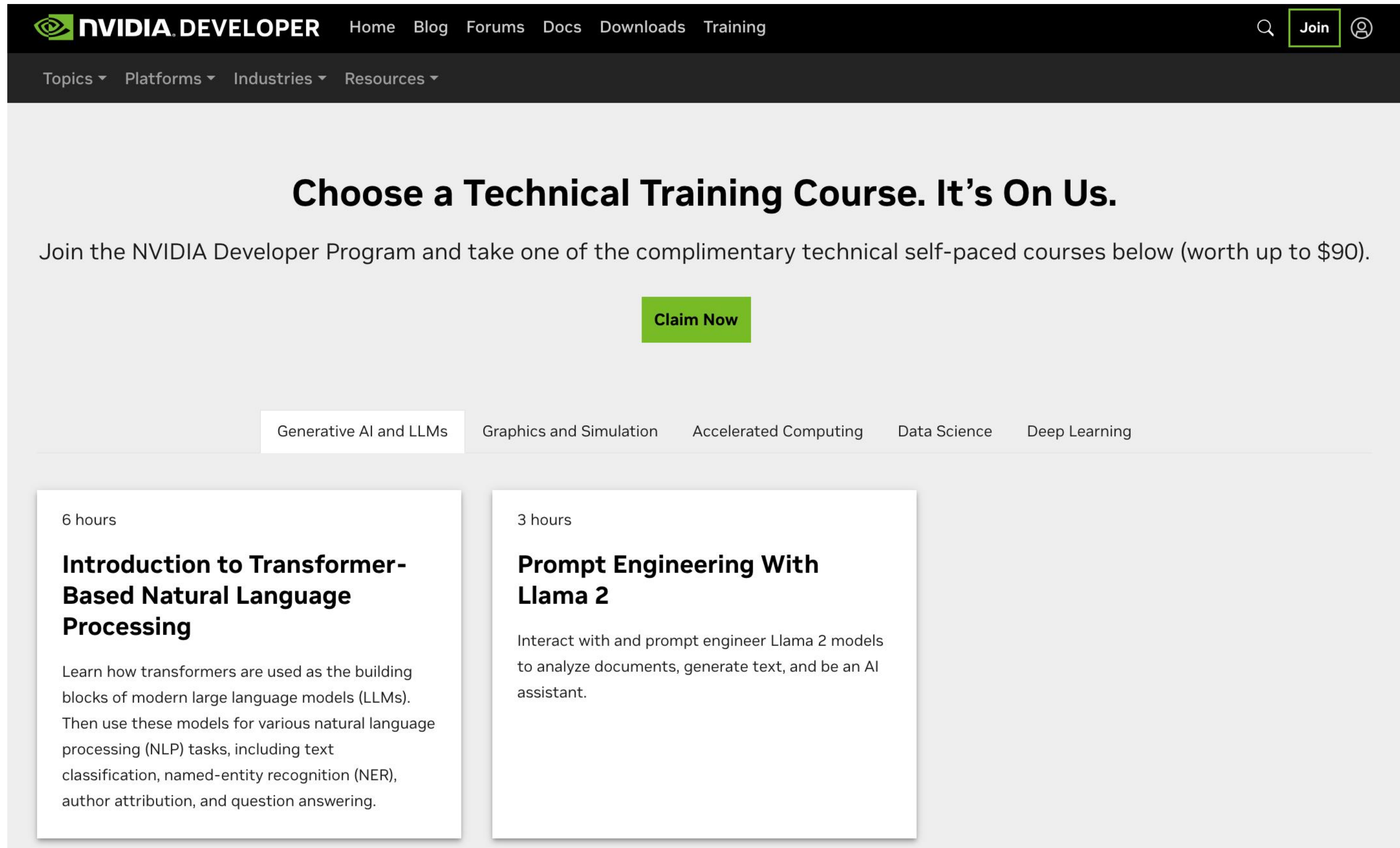
Course Prerequisites:

A basic understanding of [Deep Learning Concepts](#).

https://learn.nvidia.com/courses/course-detail?course_id=course-v1:DLI+S-FX-14+V1

Join the NVIDIA Developer Program

take one of the complimentary technical self-paced courses (worth up to \$90)



The screenshot shows the NVIDIA Developer Program website. At the top is a black navigation bar with the NVIDIA logo, the text 'NVIDIA DEVELOPER', and links for Home, Blog, Forums, Docs, Downloads, and Training. On the right of this bar are a search icon, a 'Join' button, and a user profile icon. Below the navigation bar is a dark grey bar with dropdown menus for Topics, Platforms, Industries, and Resources. The main content area has a light grey background. It features a large heading 'Choose a Technical Training Course. It's On Us.' followed by a paragraph: 'Join the NVIDIA Developer Program and take one of the complimentary technical self-paced courses below (worth up to \$90).' Below this is a green 'Claim Now' button. A horizontal menu below the button lists five categories: 'Generative AI and LLMs' (which is highlighted with a white background), 'Graphics and Simulation', 'Accelerated Computing', 'Data Science', and 'Deep Learning'. Under the 'Generative AI and LLMs' category, there are two course cards. The first card is titled 'Introduction to Transformer-Based Natural Language Processing', has a duration of '6 hours', and describes learning about transformers and their use in NLP tasks like text classification and question answering. The second card is titled 'Prompt Engineering With Llama 2', has a duration of '3 hours', and describes interacting with and engineering Llama 2 models for tasks like document analysis and AI assistance.

NVIDIA DEVELOPER Home Blog Forums Docs Downloads Training

Topics ▾ Platforms ▾ Industries ▾ Resources ▾

Choose a Technical Training Course. It's On Us.

Join the NVIDIA Developer Program and take one of the complimentary technical self-paced courses below (worth up to \$90).

Claim Now

Generative AI and LLMs Graphics and Simulation Accelerated Computing Data Science Deep Learning

6 hours

Introduction to Transformer-Based Natural Language Processing

Learn how transformers are used as the building blocks of modern large language models (LLMs). Then use these models for various natural language processing (NLP) tasks, including text classification, named-entity recognition (NER), author attribution, and question answering.

3 hours

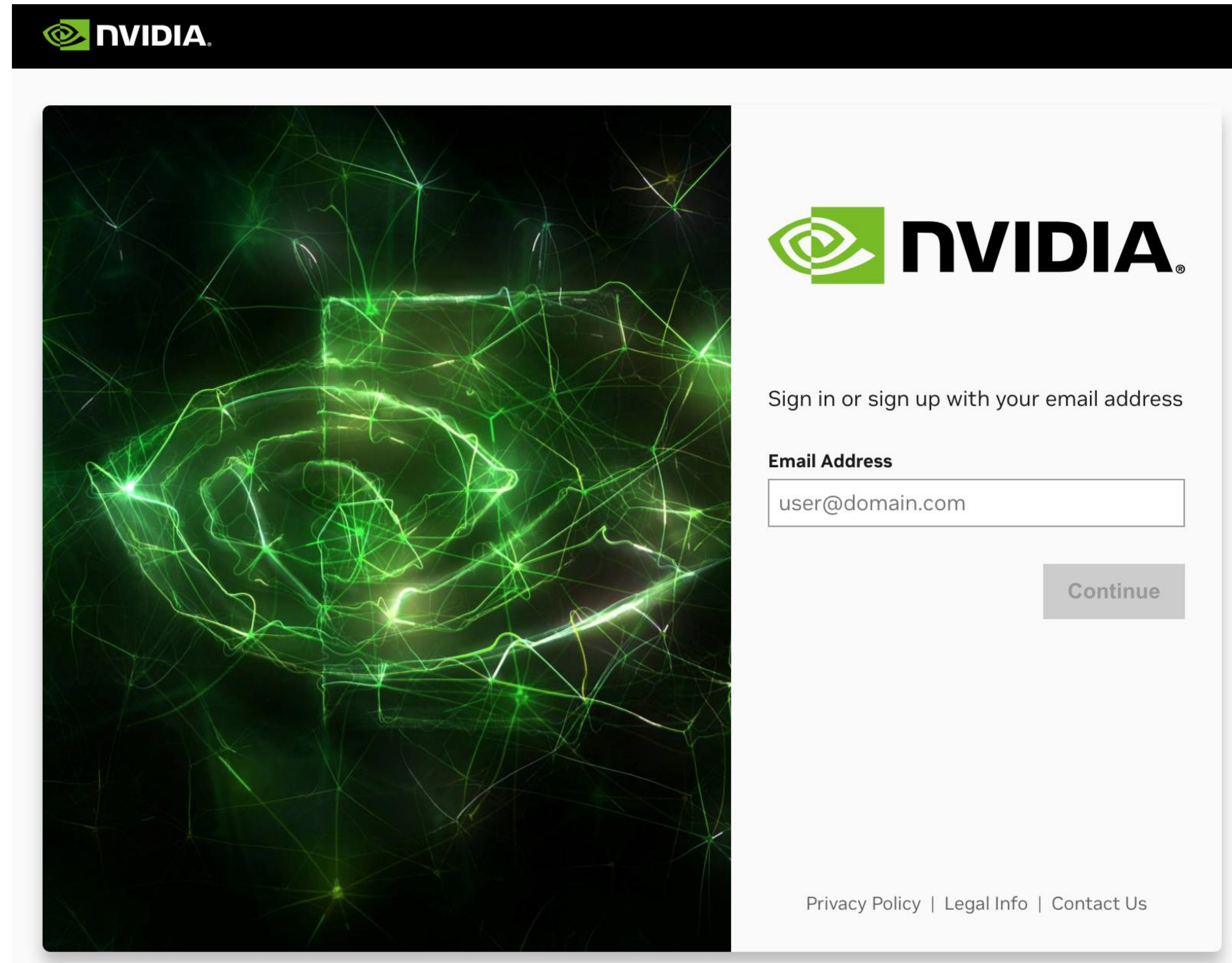
Prompt Engineering With Llama 2

Interact with and prompt engineer Llama 2 models to analyze documents, generate text, and be an AI assistant.

<https://developer.nvidia.com/join-nvidia-developer-program>

Join the NVIDIA Developer Program

take one of the complimentary technical self-paced courses (worth up to \$90)



The screenshot shows the NVIDIA Developer Program sign-up page. On the left is a large image of a glowing green neural network. On the right is a white sign-up form with the NVIDIA logo at the top. The form includes the text "Sign in or sign up with your email address", an "Email Address" label, a text input field containing "user@domain.com", and a "Continue" button. At the bottom of the form are links for "Privacy Policy", "Legal Info", and "Contact Us".

 NVIDIA

 NVIDIA

Sign in or sign up with your email address

Email Address

user@domain.com

Continue

[Privacy Policy](#) | [Legal Info](#) | [Contact Us](#)

Join the NVIDIA Developer Program

take one of the complimentary technical self-paced courses (worth up to \$90)

Date of birth
1970-00-20

Password
.....

Confirm password
.....

☒ Stay logged in

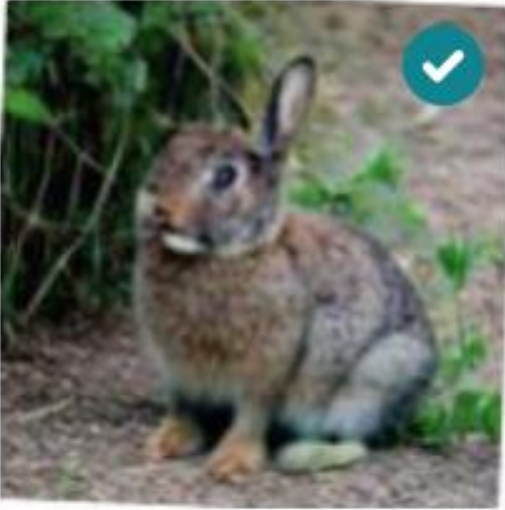
☐ By proceeding, I agree to the NVIDIA Developer Program Terms of Service

Privacy Policy | Legal Info | Contact Us

Copyright © 2024 NVIDIA Corporation

Select the images matching the toy in the example image.
If there are None, click Skip



EN

⋮ ↺  Verify

English (US) ▾

<https://developer.nvidia.com/join-nvidia-developer-program>



Join the NVIDIA Developer Program

take one of the complimentary technical self-paced courses (worth up to \$90)

 **NVIDIA DEVELOPER** DEEP LEARNING INSTITUTE PROGRAM BENEFITS

SELECT YOUR FREE COURSE.

Thank you for your participation in the NVIDIA Developer Program. Please select your free DLI course below.

English

Integrating Sensors with NVIDIA DRIVE®

Getting Started with Deep Learning

Deploying a Model for Inference at Production Scale

Get Started with Highly Accurate Custom ASR for Speech AI

Introduction to Graph Neural Networks

Introduction to Transformer-Based Natural Language Processing

Prompt Engineering with LLaMA-2 (Access Expires Dec. 5th 2025)

Generative AI with Diffusion Models

Building Real-Time Video AI Applications

Introduction to Robotic Simulations in Isaac Sim

Generative AI with Diffusion Models

Take a deeper dive into denoising diffusion models, which are a popular choice for text-to-image pipelines, with applications in creative content generation, data augmentation, simulation and planning, anomaly detection, drug discovery, personalized recommendations, and more.

 Certificate Available

 Duration: 08:00

Continue >

<https://developer.nvidia.com/join-nvidia-developer-program>



Join the NVIDIA Developer Program

take one of the complimentary technical self-paced courses (worth up to \$90)

 **NVIDIA. DEVELOPER** DEEP LEARNING INSTITUTE PROGRAM BENEFITS

SELECT YOUR FREE COURSE.

Thank you for your participation in the NVIDIA Developer Program. Please select your free DLI course below.

English ▾

Modeling Time Series Data with Recurrent Neural Networks in Keras (Access ends 10/16/2024)

Optimizing CUDA Machine Learning Codes With Nsight Profiling Tools

Getting Started with Accelerated Computing in CUDA C/C++

Fundamentals of Accelerated Computing with CUDA Python

Fundamentals of Accelerated Computing with OpenACC

Integrating Sensors with NVIDIA DRIVE®

Getting Started with Deep Learning

Deploying a Model for Inference at Production Scale

Get Started with Highly Accurate Custom ASR for Speech AI

Getting Started with Deep Learning

Learn how deep learning works through hands-on exercises in computer vision and natural language processing.

 Certificate Available

 Duration: 08:00

Continue >

<https://developer.nvidia.com/join-nvidia-developer-program>



Join the NVIDIA Developer Program

take one of the complimentary technical self-paced courses (worth up to \$90)

 **NVIDIA. DEVELOPER** DEEP LEARNING INSTITUTE PROGRAM BENEFITS

SELECT YOUR FREE COURSE.

Thank you for your participation in the NVIDIA Developer Program. Please select your free DLI course below.

English

Get Started with Highly Accurate Custom ASR for Speech AI

Introduction to Graph Neural Networks

Introduction to Transformer-Based Natural Language Processing

Prompt Engineering with LLaMA-2 (Access Expires Dec. 5th 2025)

Generative AI with Diffusion Models

Building Real-Time Video AI Applications

Introduction to Robotic Simulations in Isaac Sim

Introduction to Physics-informed Machine Learning with Modulus

Essentials of USD in Omniverse: Access Expires 09/18/2025

Synthetic Data Generation for Training Computer Vision Models

Generative AI with Diffusion Models

Take a deeper dive into denoising diffusion models, which are a popular choice for text-to-image pipelines, with applications in creative content generation, data augmentation, simulation and planning, anomaly detection, drug discovery, personalized recommendations, and more.

 Certificate Available

 Duration: 08:00

Continue >

<https://developer.nvidia.com/join-nvidia-developer-program>



NVIDIA Deep Learning Institute (DLI)

Deep Learning Institute

[Find Training](#)[Self Paced Courses](#)[Instructor-Led Workshops](#)[Educator Programs](#)[Enterprise Solutions](#)[Certification](#)[Resources](#)

Search

Monthly Activity

Skill Points

0

Time Spent

Courses in Progress

1

Courses Completed

0

Watched Videos

Assessments

Skills

Certificates

No Certificates

You don't have any certificates yet.

Courses in Progress

Self-paced

Generative AI with Diffusion Models

0% Completed

08:00



NVIDIA Deep Learning Institute (DLI)



Products Solutions Industries For You

Shop Drivers Support Min-Yuh Day

Deep Learning Institute

Find Training

Self Paced Courses

Instructor-Led Workshops

Educator Programs

Enterprise Solutions

Certification

Resources

Generative AI with Diffusion Models

Course

Progress

Bookmarks

Updates

Generative AI with Diffusion Models > Start Here > 0: Server Access

Generative AI with Diffusion Models

Start Here

Next Steps

Feedback

Previous

Next



0: Server Access

Bookmark this page

Welcome to Generative AI with Diffusion Models. Please click "Next" below to get started.

Underneath each video is a link to start your own private server for hands-on coding practice. Click the "Start" button to boot up the server. In a few minutes after the server is done loading, click "Launch" to access the code labs.

1: From U-Nets to Diffusion

Bookmark this page

Theory

<https://learn.nvidia.com/my-learning>





Monthly Activity

Skill Points	0
Time Spent	
Courses in Progress	16
Courses Completed	12
Watched Videos	
Assessments	

Skills

Certificates

Introduction to Transformer-Based Natural Language Processing

Building RAG Agents with LLMs

Building RAG Agents with LLMs

Accelerating End-to-End Data Science Workflows

Generative AI with Diffusion Models

Building Agentic AI Applications with LLMs

Certificates

Introduction to Transformer-Based Natural Language Processing

Building RAG Agents with LLMs

Building RAG Agents with LLMs

Accelerating End-to-End Data Science Workflows

Generative AI with Diffusion Models

Building Agentic AI Applications with LLMs

Building Agentic AI Applications with LLMs

Rapid Application Development with Large Language Models (LLMs)

Completed Courses

[View more](#)

Self-paced

Sizing LLM Inference Systems

100% Completed

03:00

Self-paced

Augment your LLM Using Retrieval Augmented Generation

100% Completed

01:00

Self-paced

Building RAG Agents with LLMs

100% Completed

08:00

Self-paced

Generative AI Explained

100% Completed

02:00

Self-paced

Introduction to Transform Based Natural Language Processing

100% Completed

06:00

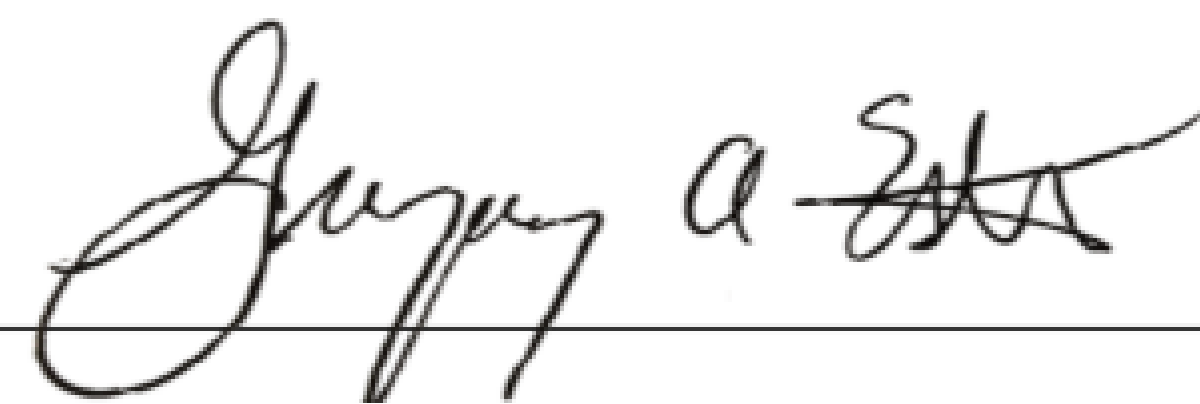
Certificate of Completion

This certificate is awarded to

Min-Yuh Day

for successfully completing

Building RAG Agents with LLMs



Greg Estes

Vice President, NVIDIA

Issue Date: : December 8, 2024

Certification ID: ed-qOCIMQaatzU8SNUNxgw |

https://learn.nvidia.com/certificates?id=ed-qOCIMQaatzU8SNUNxgw/courses/course?course_id=course-v1:DLI+S-FX-15+V1

<https://learn.nvidia.com/certificates?id=ed-qOCIMQaatzU8SNUNxgw>

All Self-Paced Courses

Accelerated Computing

Data Science

Deep Learning

Generative AI/LLM

Graphics and Simulation

Infrastructure

[Share Generative AI/LLM Courses](#) 

Self-paced

Generative AI Explained

Free
02:00

Self-paced

Introduction to NVIDIA NIM™
Microservices

Free
02:00

Self-paced

Introduction to Deploying
RAG Pipelines for Production
at Scale

\$90
03:00

Self-paced

Generative AI with Diffusion
Models

\$90
08:00

Self-paced

Techniques for Improving the
Effectiveness of RAG Systems

\$30
03:00

Self-paced

Introduction to Transformer-
Based Natural Language
Processing

\$30
06:00

Self-paced

Building LLM Applications
With Prompt Engineering

\$90
08:00

Self-paced

Synthetic Tabular Data
Generation Using
Transformers

\$30
04:00

Self-paced

Sizing LLM Inference Systems

Free
03:00

Self-paced

Building RAG Agents with
LLMs

Free
08:00

Self-paced

Augment your LLM Using
Retrieval Augmented
Generation

Free
01:00

NVIDIA Deep Learning Institute (DLI)

Building RAG Agents with LLMs

Self-paced Course

Building RAG Agents with LLMs

Agents powered by large language models (LLMs) have shown great retrieval capability for using tools, looking at documents, and plan their approaches. This course will show you how to deploy an agent system in practice with the flexibility to scale up your system to meet the demands of users and customers.



[Continue Learning](#)

About this Course

This course is free for a limited time.

The evolution and adoption of large language models (LLMs) have been nothing short of revolutionary, with retrieval-based systems at the forefront of this technological leap. These models are not just tools for automation; they are partners in enhancing productivity, capable of holding informed conversations by interacting with a vast array of tools and documents. This course is designed for those eager to explore the potential of these systems, focusing on practical deployment and the efficient implementation required to manage the considerable demands of both users and deep learning models. As we delve into the intricacies of LLMs, participants will gain insights into advanced orchestration techniques that include internal reasoning, dialog management, and effective tooling strategies.

Course Details

Duration: 08:00

Price: Free

Level: Technical - Intermediate

Subject: Generative AI/LLM

Language: English

Course Prerequisites:

Introductory deep learning knowledge, with comfort

https://learn.nvidia.com/courses/course-detail?course_id=course-v1:DLI+S-FX-15+V1

Deep Learning Institute

Find TrainingSelf Paced CoursesInstructor-Led WorkshopsEducator ProgramsEnterprise SolutionsCertificationResources

Building RAG Agents with LLMs

Course

Progress

Bookmarks

Updates

Building RAG Agents with LLMs

Introduction

Introduction

Building RAG Agents with LLMs

Previous

Next

Introduction

Environment and LLMs

LangChain

Documents and Embeddings

Retrieval-Augmented
Generation

Next Steps

Feedback



Building RAG Agents with LLMs

Introduction



Building RAG Agents with LLMs

Deep Learning Institute

Find Training

Self Paced Courses

Instructor-Led Workshops

Educator Programs

Enter

ertifi

Building RAG Agents with LLMs

Training Home

My Learning

NVIDIA Account

Logout

Course

Progress

Bookmarks

Updates

Building RAG Agents with LLMs

Introduction

Introduction

Building RAG Agents with LLMs

Building RAG Agents with LLMs

Course

Progress

Bookmarks

Updates

Building RAG Agents with LLMs

Environment and LLMs

Environment [0, 1, 2]

Building RAG Agents with LLMs
Introduction
Environment and LLMs
Environment [0, 1, 2]
Part 1: Course Environment
Part 2: LLM Services
LangChain
Environment [3, 4]
Part 3: LangChain
Part 4: Running States
Documents and Embeddings
Environment [5, 6]

Previous

Next

Bookmark this page

Welcome to **Building RAG Agents with LLMs**. In this first section, we will get introduced to the overall course environment, LLM services, and recommended workflows!

This tab contains the course environment for this section, which will contain the notebooks for the next two videos! Please click through the videos in the remaining tabs to watch the material and work through the exercises!

Please click the "Start" button to start up your own private server for hands-on coding practice. It will take a few minutes to start up, so go ahead and click it now and then proceed to the next video! After a few minutes when the server has loaded, click "Launch" to access the code labs.

START

Building RAG Agents with LLMs

Building RAG Agents with LLMs

Course Progress Bookmarks Updates

Building RAG Agents with LLMs Environment and LLMs Environment [0, 1, 2]

Building RAG Agents with LLMs

Introduction

Introduction

Course Slides

Environment and LLMs

Environment [0, 1, 2]

Part 1: Course Environment

Part 2: LLM Services

LangChain

Environment [3, 4]

Part 3: LangChain

Previous

Next

welcome to **Building RAG Agents with LLMs**. In this first section, we will get introduced to the overall course environment, LLM services, and recommended workflows!

This tab contains the course environment for this section, which will contain the notebooks for the next two videos! Please click through the videos in the remaining tabs to watch the material and work through the exercises!

Please click the "Start" button to start up your own private server for hands-on coding practice. It will take a few minutes to start up, so go ahead and click it now and then proceed to the next video! After a few minutes when the server has loaded, click "Launch" to access the code labs.



DEEP
LEARNING
INSTITUTE

This Lab 0 : 01 : 06 / 2 : 00 : 00

Course 13 : 45 : 51 / 32 : 00 : 00



LAUNCH



STOP TASK

Building RAG Agents with LLMs

The screenshot displays the JupyterLab web interface. The top navigation bar includes a menu (File, Edit, View, Run, Kernel, Tabs, Settings, Help) and a browser address bar showing '54.211.119.108/lab'. On the left, a file browser pane shows a directory structure with folders like 'chatbot', 'composer', 'docker_router', 'frontend', 'imgs', 'llm_client', and 'slides'. Below these are four Jupyter Notebook files: '00_jupyterlab.ipynb', '01_microservices.ipynb', '02_llms.ipynb', and '99_table_of_contents.ipynb', all marked as 'yesterday'. The right pane, titled 'Launcher', offers options to open a 'Notebook' or 'Console', both using a 'Python 3 (ipykernel)' environment. At the bottom of the launcher are icons for 'Terminal', 'Text File', 'Markdown File', 'Python File', and 'Show Contextual Help'. The bottom status bar shows 'Simple' mode, a kernel icon, and a 'Launcher' tab indicator.

Building RAG Agents with LLMs

The screenshot displays the JupyterLab web interface. At the top, a browser address bar shows the URL `54.211.119.108/lab/lab/tree/00_jupyterlab.ipynb`. Below this is a menu bar with options: File, Edit, View, Run, Kernel, Tabs, Settings, and Help. On the left, a file browser sidebar shows a directory structure with folders like `chatbot`, `composer`, `docker_router`, `frontend`, `imgs`, `llm_client`, and `slides`. It also lists several Jupyter notebooks, with `00_jupyterlab.ipynb` currently selected. The main workspace area contains a notebook titled `00_jupyterlab.ipynb`, which is open to a Markdown cell. This cell features the NVIDIA logo and the text "DEEP LEARNING INSTITUTE". Below the logo, the title **Notebook 0: JupyterLab** is displayed in green. The text in the notebook explains that JupyterLab is used to manage the environment and provides access to interactive iPython notebooks, a folder structure, and a terminal window. It also describes the interface components: a menu bar at the top, a file browser in the left sidebar, and a main work area. At the bottom of the notebook, a green arrow points to the menu bar, which is labeled "Menu bar". The bottom status bar shows the current mode as "Command", the kernel as "Python 3 (ipykernel)", and the file name `00_jupyterlab.ipynb`.

Building RAG Agents with LLMs

Environment and LLMs

Environment [0, 1, 2]

Part 1: Course Environment

Part 2: LLM Services

LangChain

Environment [3, 4]

Part 3: LangChain

Part 4: Running States

Documents and Embeddings

Retrieval-Augmented
Generation

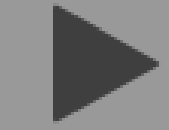
Next Steps

Feedback



DEEP
LEARNING
INSTITUTE

This Lab 0 : 10 : 32 / 2 : 00 : 00



Course 13 : 55 : 17 / 32 : 00 : 00

LAUNCH



STOP TASK



Stopping Task

Click confirm to stop this task.

Confirm

Cancel

Building RAG Agents with LLMs

Building RAG Agents with LLMs

Introduction

Environment and LLMs

LangChain

Documents and Embeddings

Environment [5, 6]

Part 5: Documents

Part 6: Embeddings

Retrieval-Augmented Generation

Environment [7, 8, Assessment]

Part 7: Vector Stores

Part 8: Evaluation

Next Steps

Previous Next

Bookmark this page

In this section, we will combine all of our prior efforts to integrate and evaluate retrieval-augmented generation pipelines! Along the way, you will also get the opportunity to work through the assessment, which will involve Gradio, LangServe, FAISS, RAG, and Evaluation! **Good Luck!**

Please click the "Start" button to start up your own private server for hands-on coding practice. It will take a few minutes to start up, so go ahead and click it now and then proceed to the next video! After a few minutes when the server has loaded, click "Launch" to access the code labs.



This Lab 0 : 15 : 39 / 4 : 00 : 00
Course 14 : 12 : 18 / 32 : 00 : 00 LAUNCH STOP TASK ASSESS TASK

Notebook 8 [Assessment]: RAG Evaluation

Welcome to the last notebook of the course! In the previous notebook, you integrated a vector store solution into a RAG pipeline! In this notebook, you will take that same pipeline and evaluate it using numerical RAG evaluation techniques incorporating LLM-as-a-Judge metrics!

Learning Objectives:

- Learn how to integrate the techniques from prior notebooks to numerically approximate the goodness of your RAG pipeline.



Building RAG Agents with LLMs

Introduction



Building RAG Agents with LLMs

Introduction

Course Environment

EDIT

**DEEP
LEARNING
INSTITUTE**

Please click the start button to the top-right. The course will take about 5 minutes to load. The course lecture slide decks will appear here when you launch the course. The numbers on the lectures correspond to the numbers of the interactive notebooks in the course.

Large Language Models

Backbones for Language Understanding



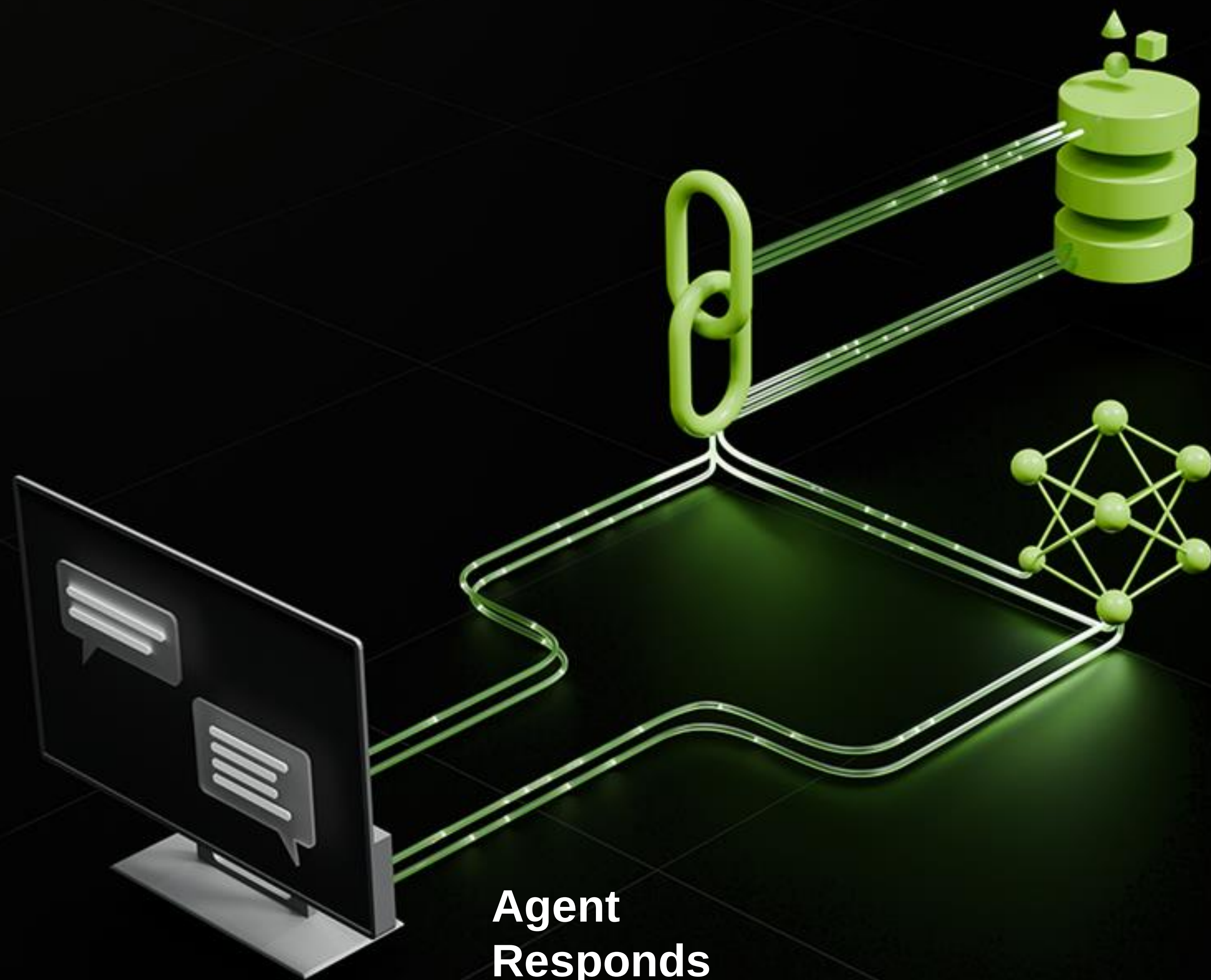
Backbones for language tasks, including classification and generation.

Dialog/Retrieval Agents

LLMs with Context and Control

User Asks
Something

Agent
Responds



Dialog/Retrieval Agents

LLMs with Context and Control

LLM Orchestration: Software + LLM helps to route to software and LLMs.

**User Asks
Something**

**Agent
Responds**



Dialog/Retrieval Agents

LLMs with Context and Control

LLM Orchestration: Software + LLM helps to route to software and LLMs.

Retrieval: Tool runs algorithms (database, code execution, semantic search, return a constant value, etc) to provide context

User Asks Something

Agent Responds

Dialog/Retrieval Agents

LLMs with Context and Control

LLM Orchestration: Software + LLM helps to route to software and LLMs.

Retrieval: Tool runs algorithms (database, code execution, semantic search, return a constant value, etc) to provide context

Augmented: Based on tool responses, the software pipeline synthesizes some “context” to feed to LLM w/ question.

User Asks Something

Agent Responds

Dialog/Retrieval Agents

LLMs with Context and Control

LLM Orchestration: Software + LLM helps to route to software and LLMs.

User Asks Something

Agent Responds

Retrieval: Tool runs algorithms (database, code execution, semantic search, return a constant value, etc) to provide context

Augmented: Based on tool responses, the software pipeline synthesizes some “context” to feed to LLM w/ question.

Generation: Based on question, instructions, and enhanced context, the LLM returns a response.

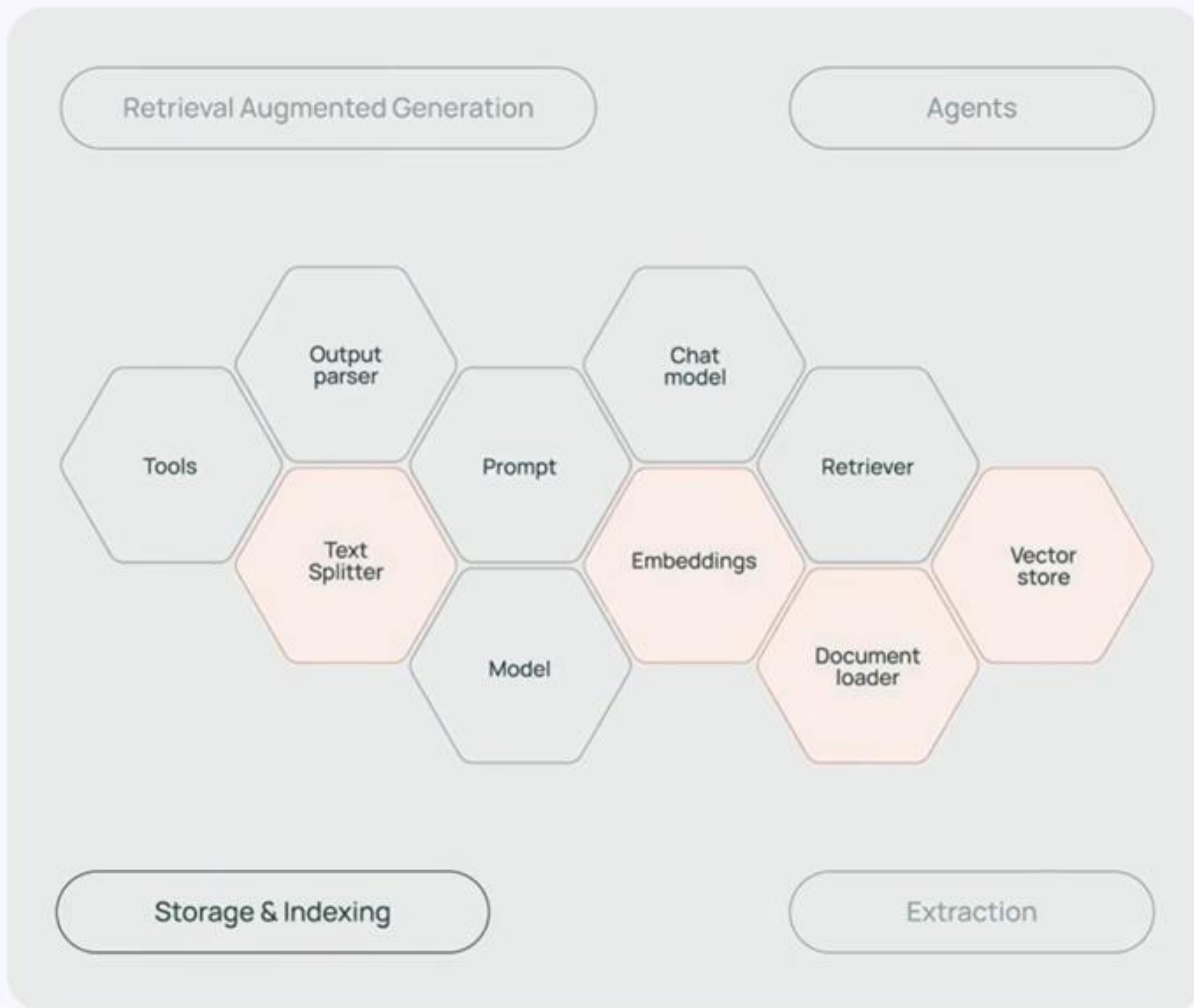
Chat Applications

Full Applications Build with LLMs



Chat Applications

Full Applications Build with LLMs



Build your app with LangChain

Build context-aware, reasoning applications with LangChain's flexible framework that leverages your company's data and APIs. Future-proof your application by making vendor optionality part of your LLM infrastructure design.

[Learn more about LangChain ↗](#)

Full Applications Build with LLMs

Instantly Run and Deploy Generative AI

Explore the latest community-built AI models with APIs optimized and accelerated by NVIDIA, then deploy anywhere with NVIDIA NIM.



Prerequisites

RAG Agents in Production

- Prior LLM/LangChain Exposure
- Intermediate Python Experience
- Exposure to Web Engineering



Course Objectives

RAG Agents in Production

- Environment
- LLM Services
- Intro to LangChain (LCEL)
- Running State Chains
- Document Loading
- Embeddings
- Document Retrieval
- RAG Evaluation

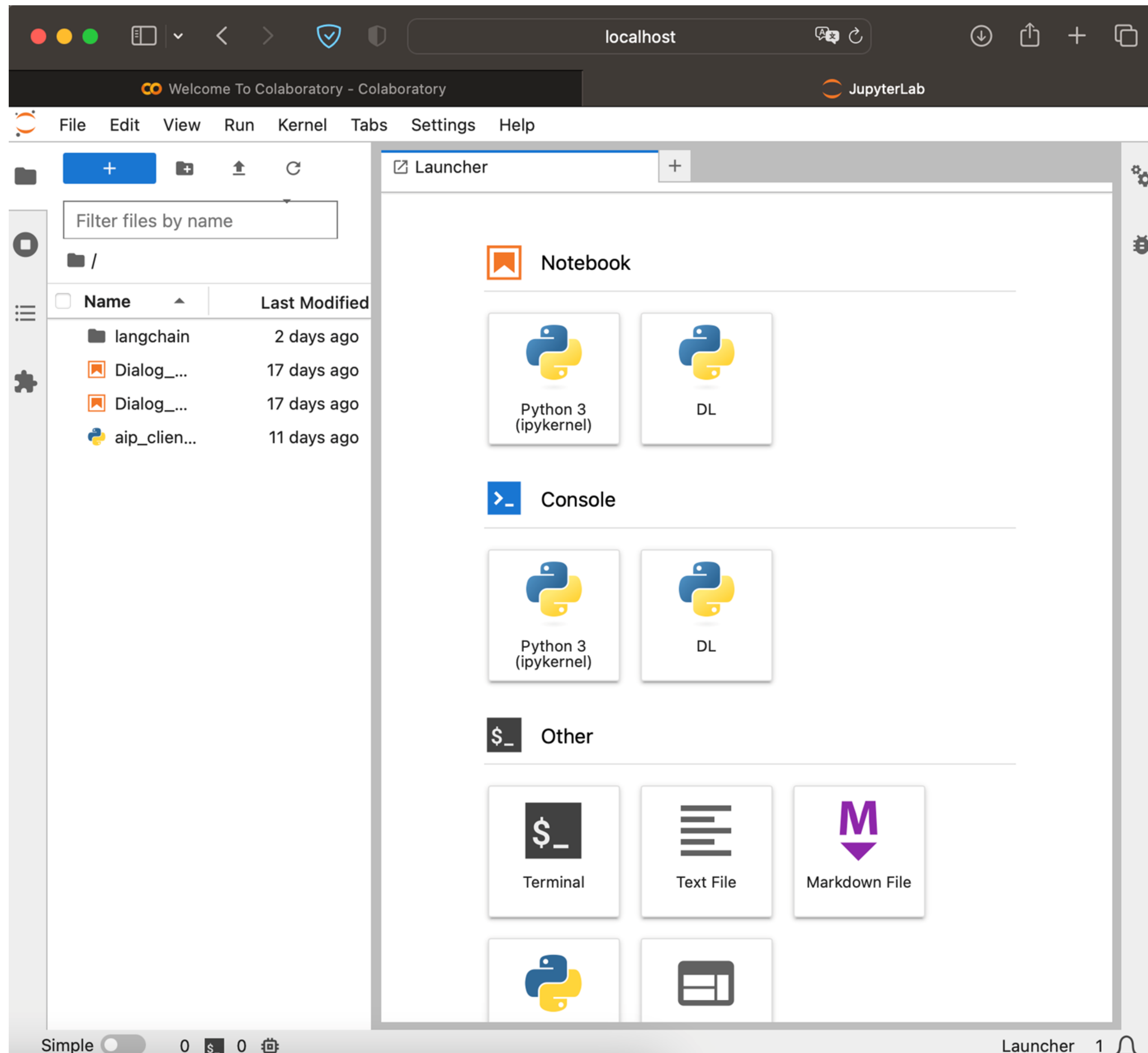




Building RAG Agents with LLMs

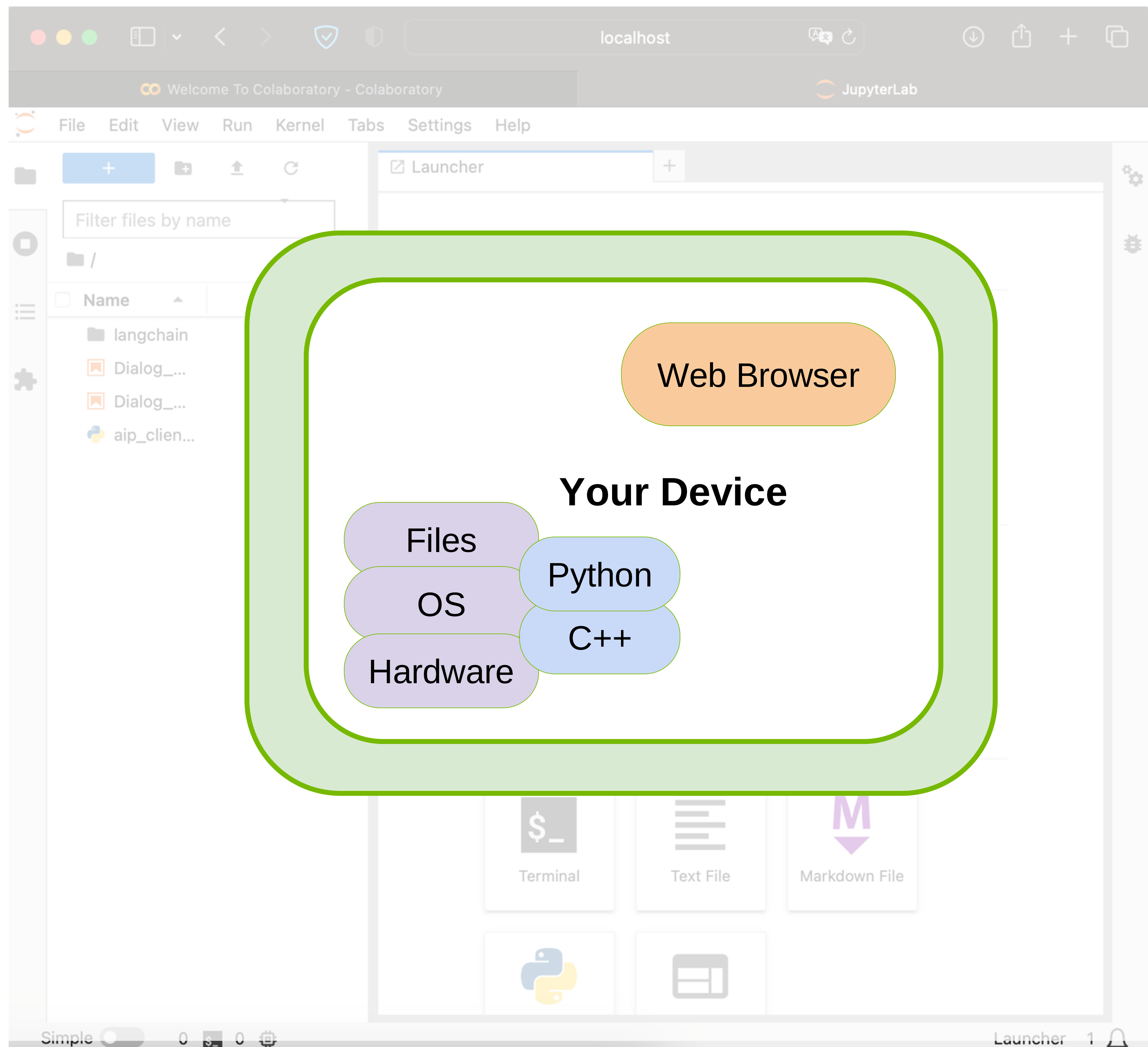
Part 1: Your Course Environment

Typical Jupyter Labs Interface



jupyter lab . & open -a Safari http://localhost:8888/lab

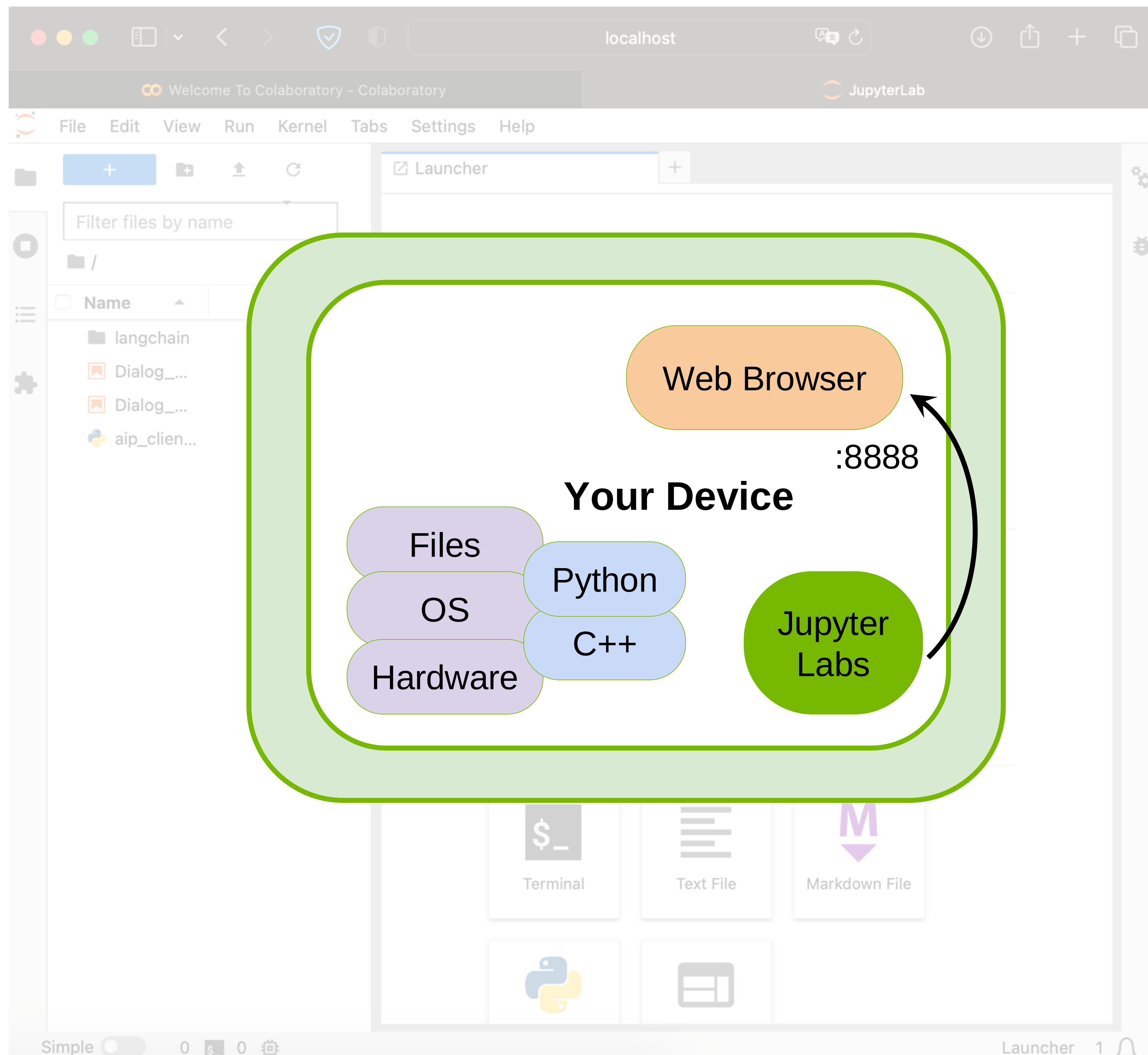
Typical Jupyter Labs Interface



`jupyter lab . &`; open -a Safari <http://localhost:8888/lab>

Source: NVIDIA DLI (2024), Building RAG Agents with LLMs, https://learn.nvidia.com/courses/course-detail?course_id=course-v1:DLI+S-FX-15+V1

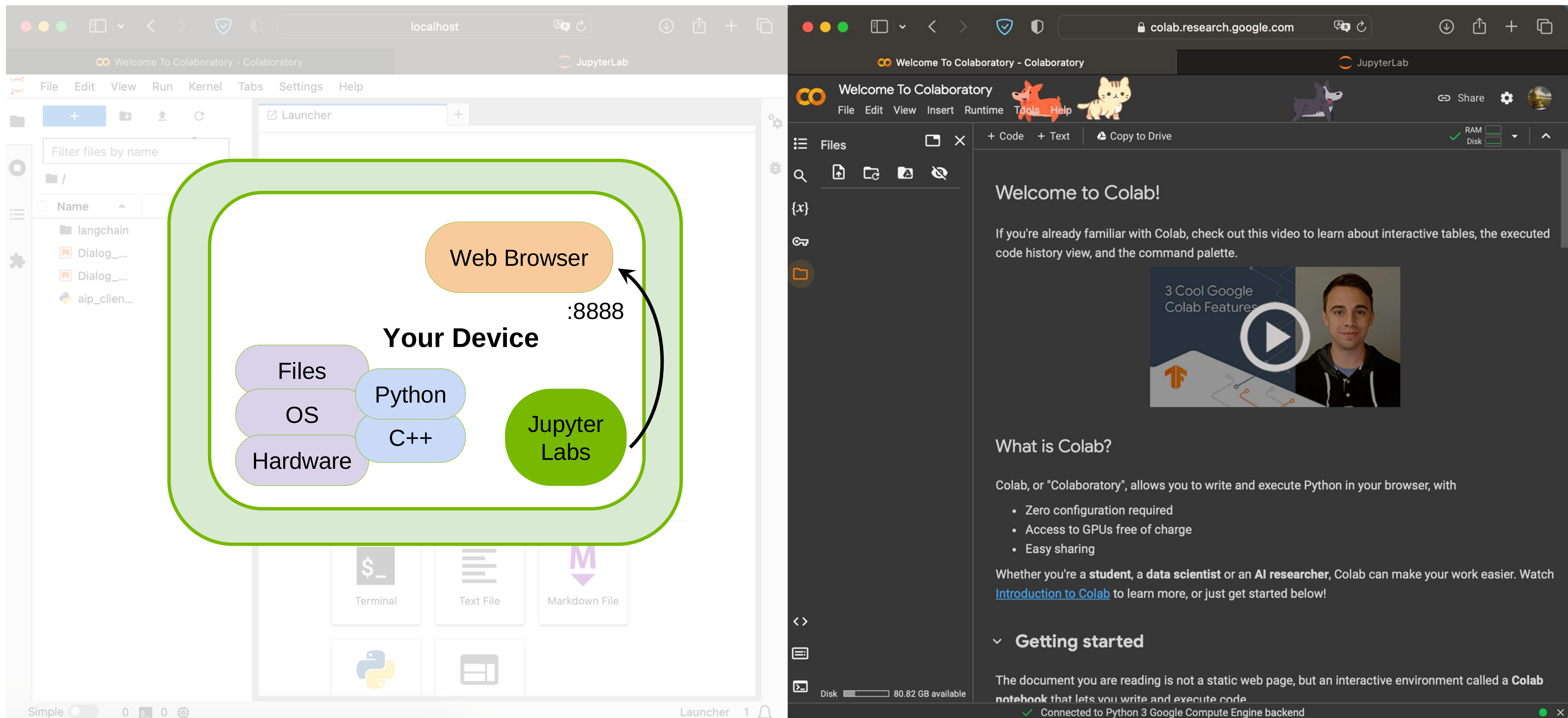
Typical Jupyter Labs Interface



jupyter lab . & open -a Safari <http://localhost:8888/lab>

Source: NVIDIA DLI (2024), Building RAG Agents with LLMs, https://learn.nvidia.com/courses/course-detail?course_id=course-v1:DLI+S-FX-15+V1

Typical Jupyter Labs Interface

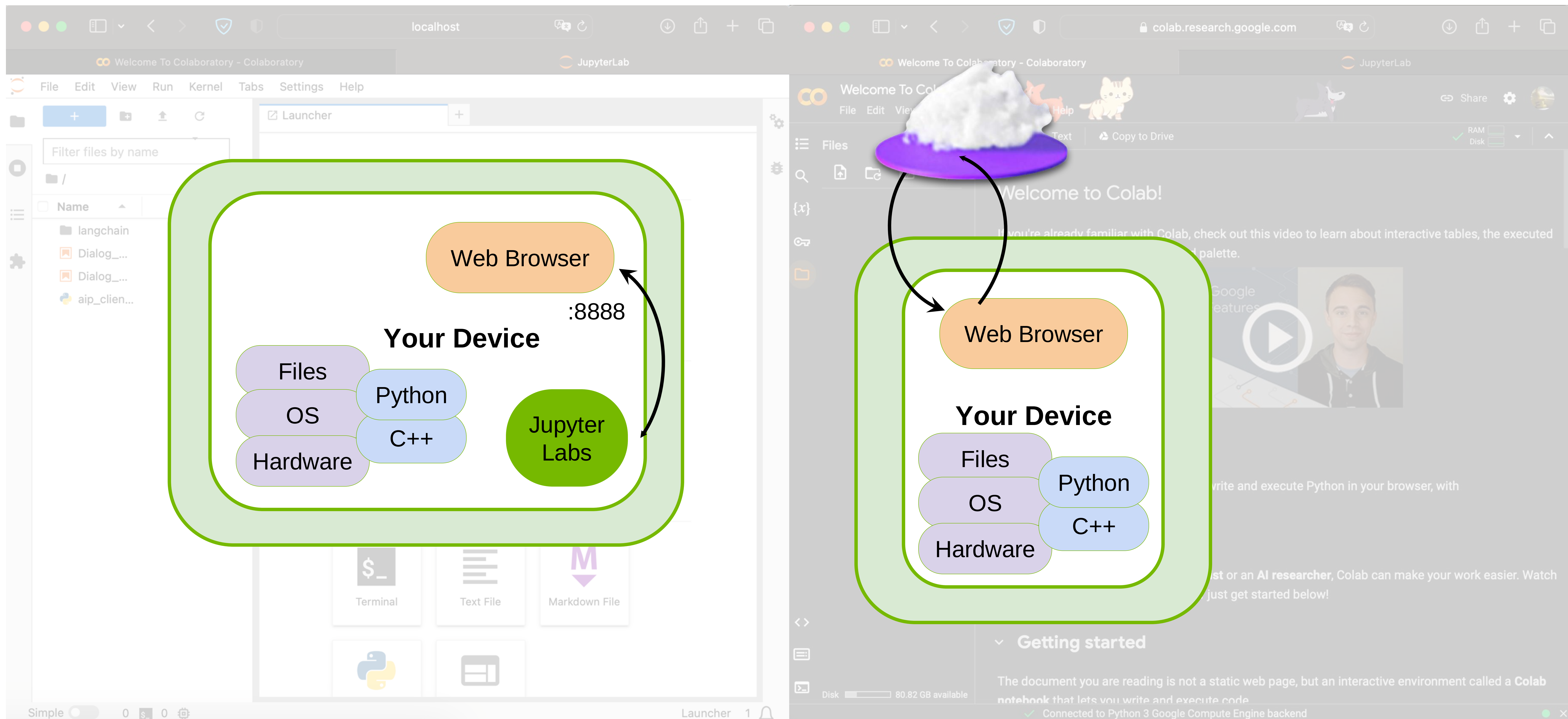


jupyter lab . & open -a Safari <http://localhost:8888/lab>

Source: NVIDIA DLI (2024), Building RAG Agents with LLMs, https://learn.nvidia.com/courses/course-detail?course_id=course-v1:DLI+S-FX-15+V1

<https://colab.research.google.com>

Typical Jupyter Labs Interface

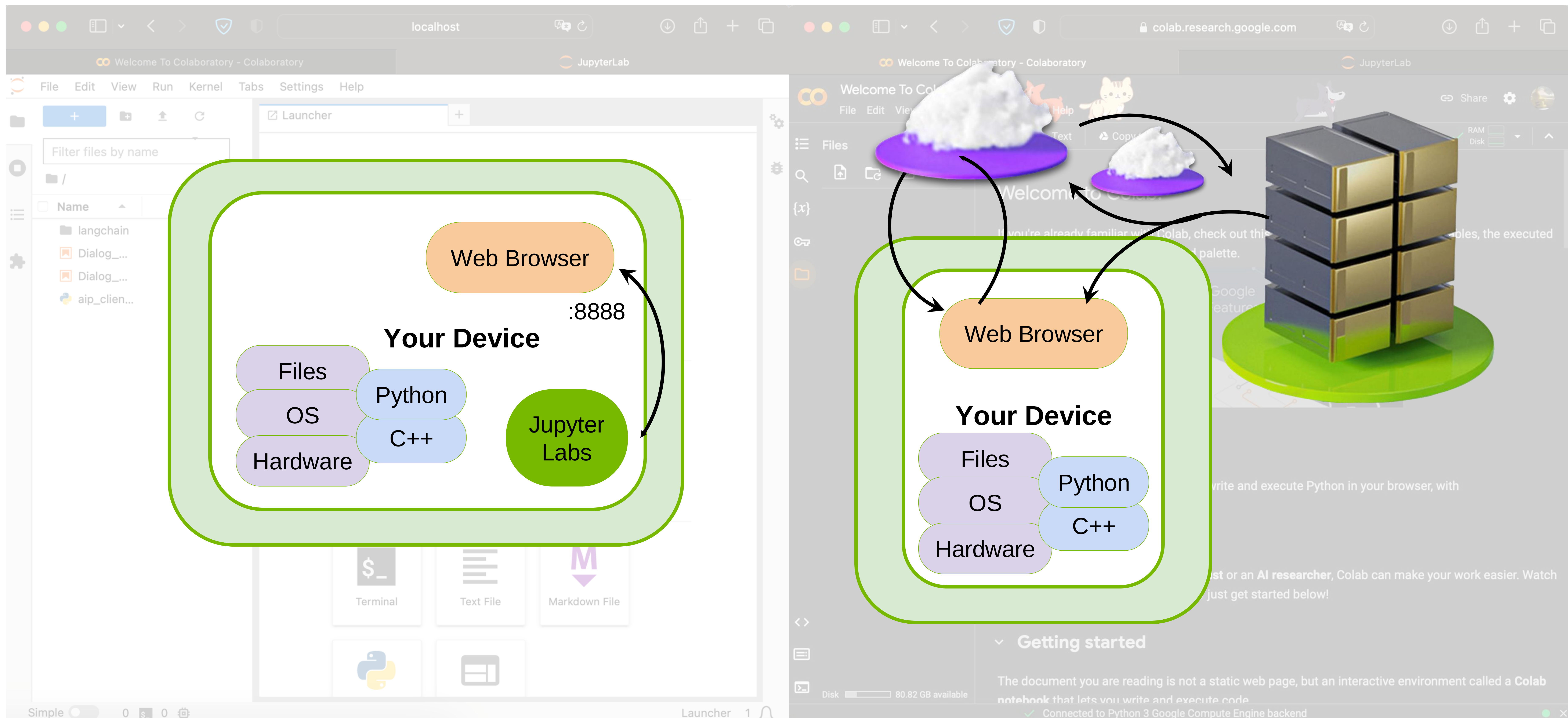


jupyter lab . & open -a Safari <http://localhost:8888/lab>

Source: NVIDIA DLI (2024), Building RAG Agents with LLMs, https://learn.nvidia.com/courses/course-detail?course_id=course-v1:DLI+S-FX-15+V1

<https://colab.research.google.com>

Typical Jupyter Labs Interface

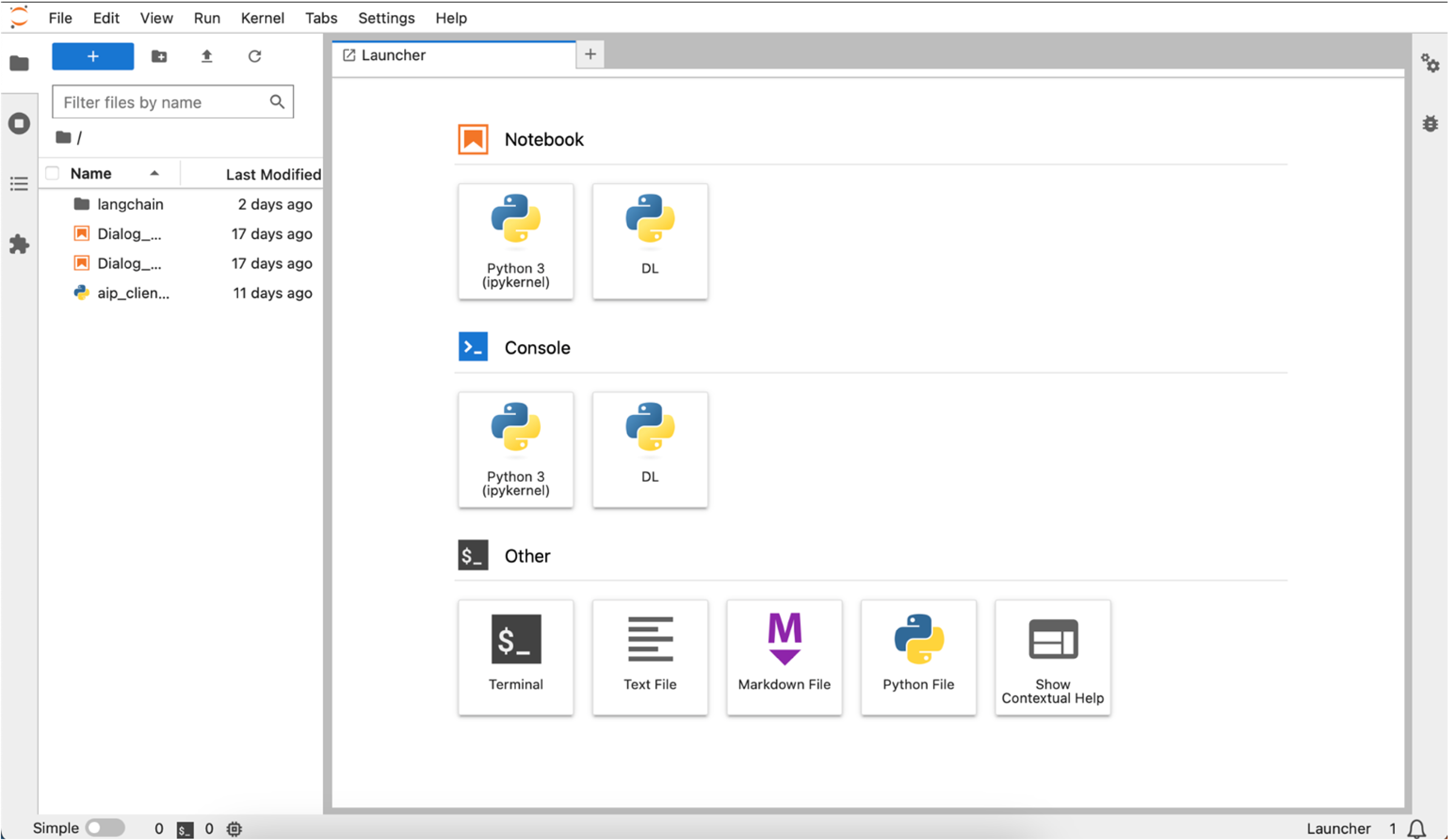


jupyter lab . & open -a Safari <http://localhost:8888/lab>

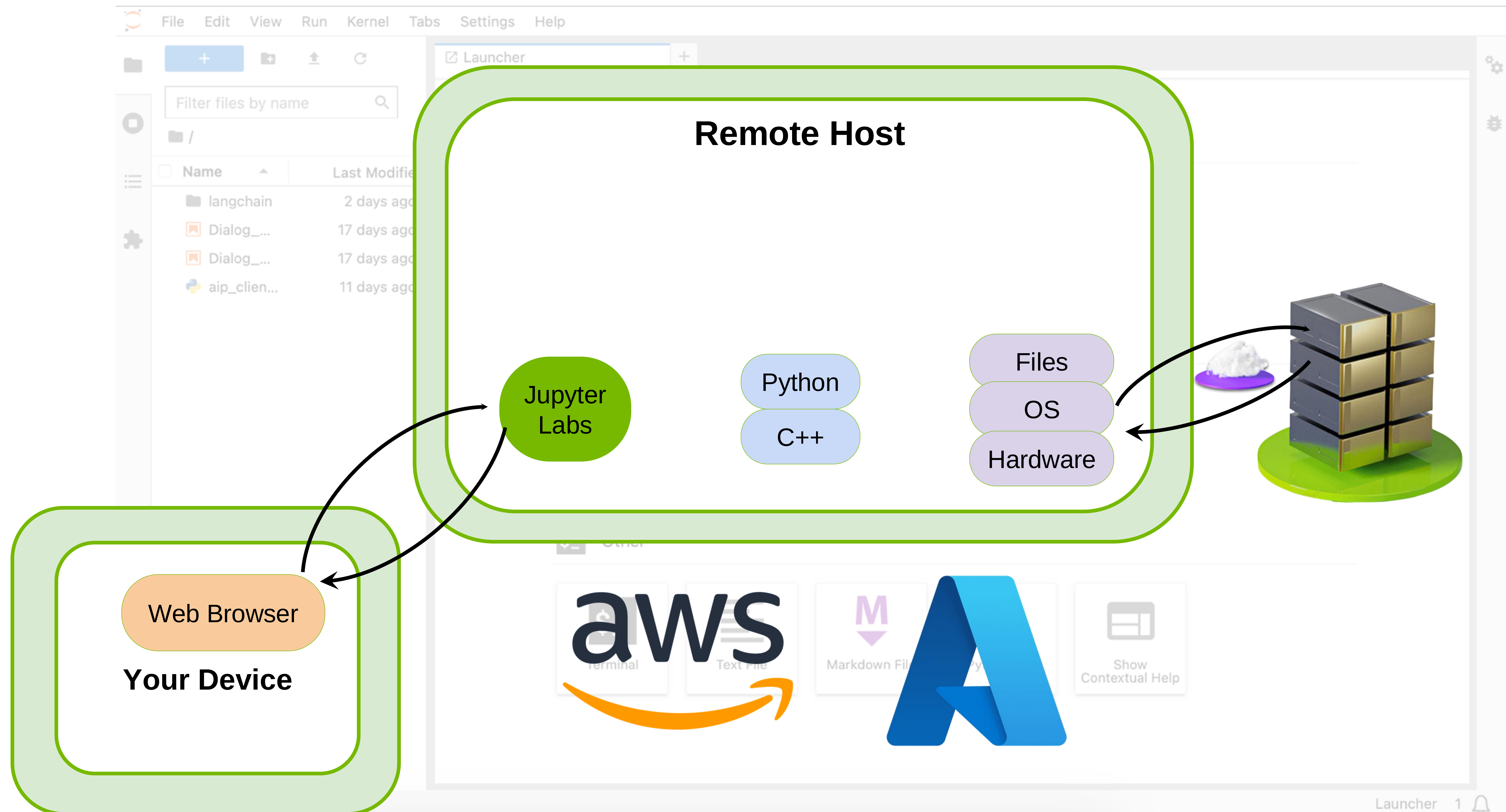
Source: NVIDIA DLI (2024), Building RAG Agents with LLMs, https://learn.nvidia.com/courses/course-detail?course_id=course-v1:DLI+S-FX-15+V1

<https://colab.research.google.com>

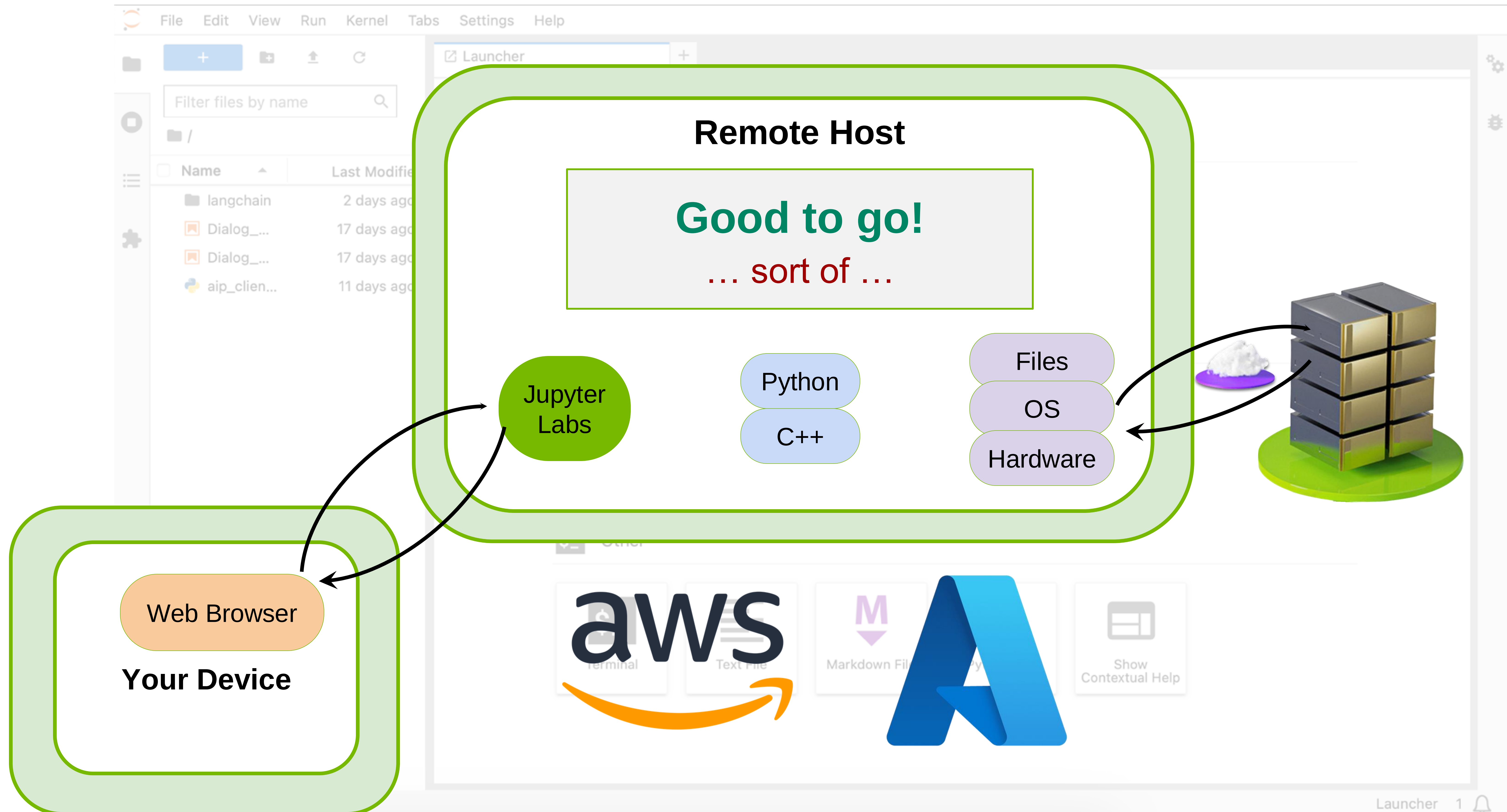
DLI Jupyter Labs Interface



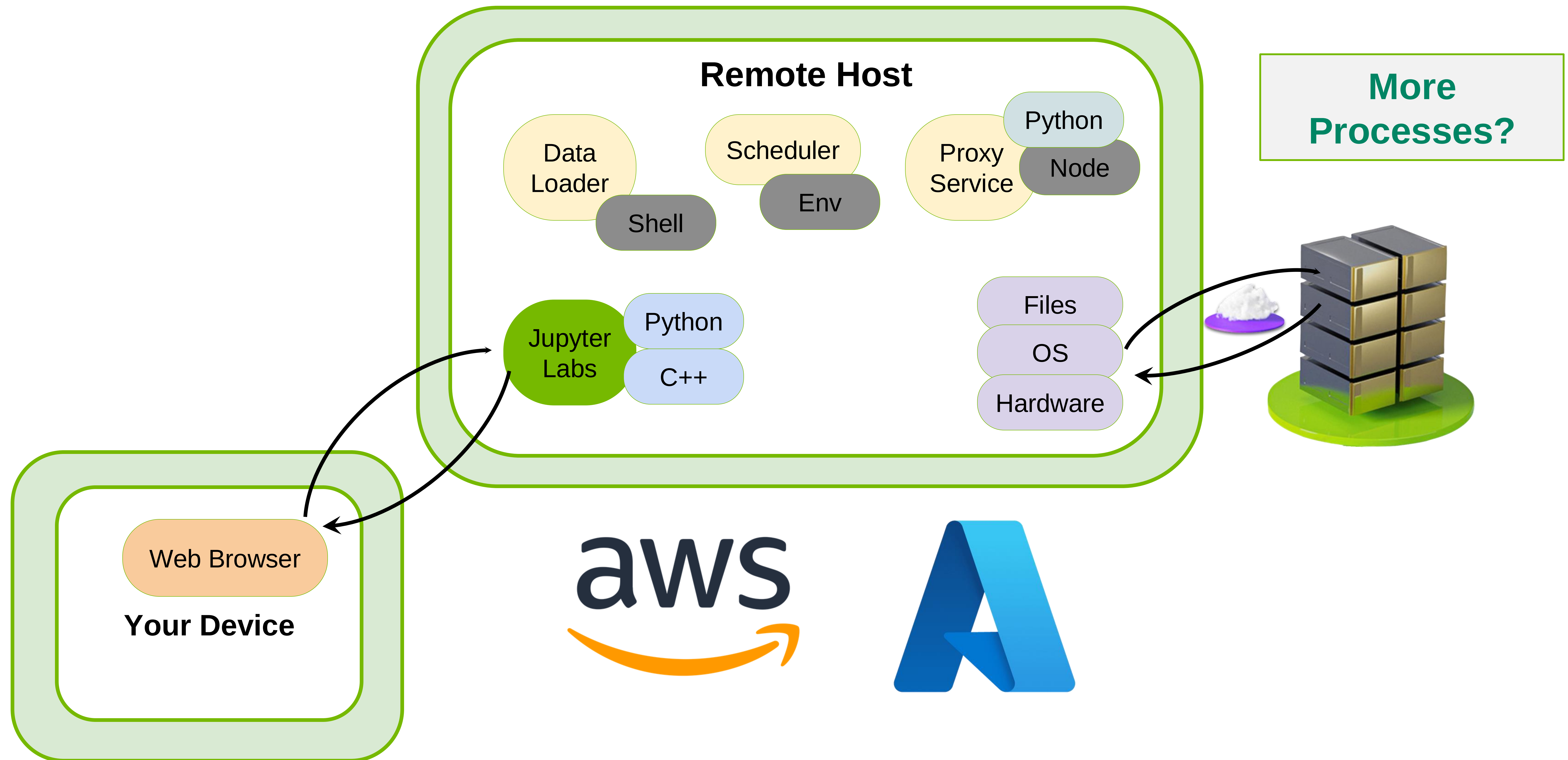
DLI Jupyter Labs Interface



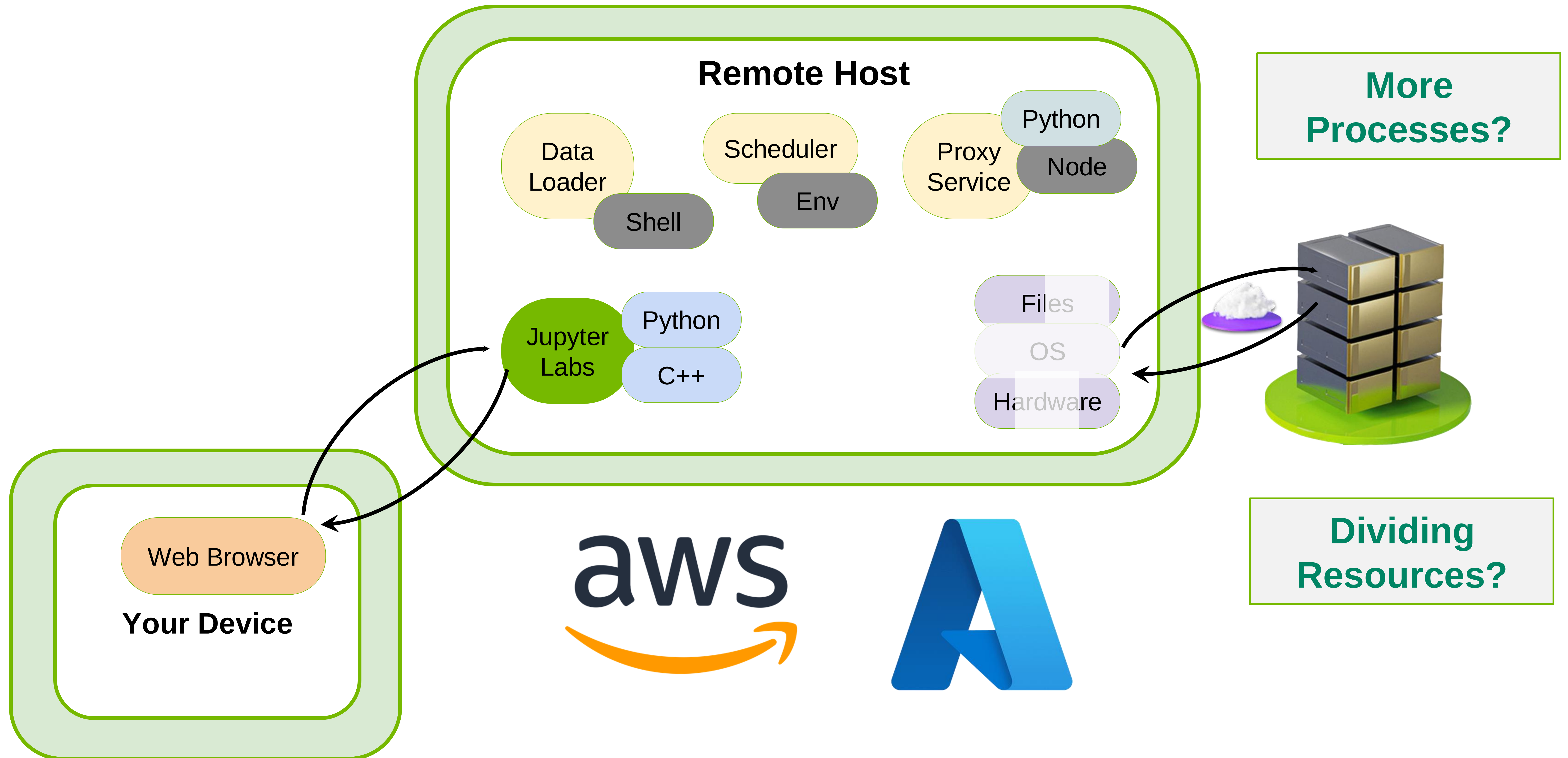
DLI Jupyter Labs Interface



DLI Jupyter Labs Interface

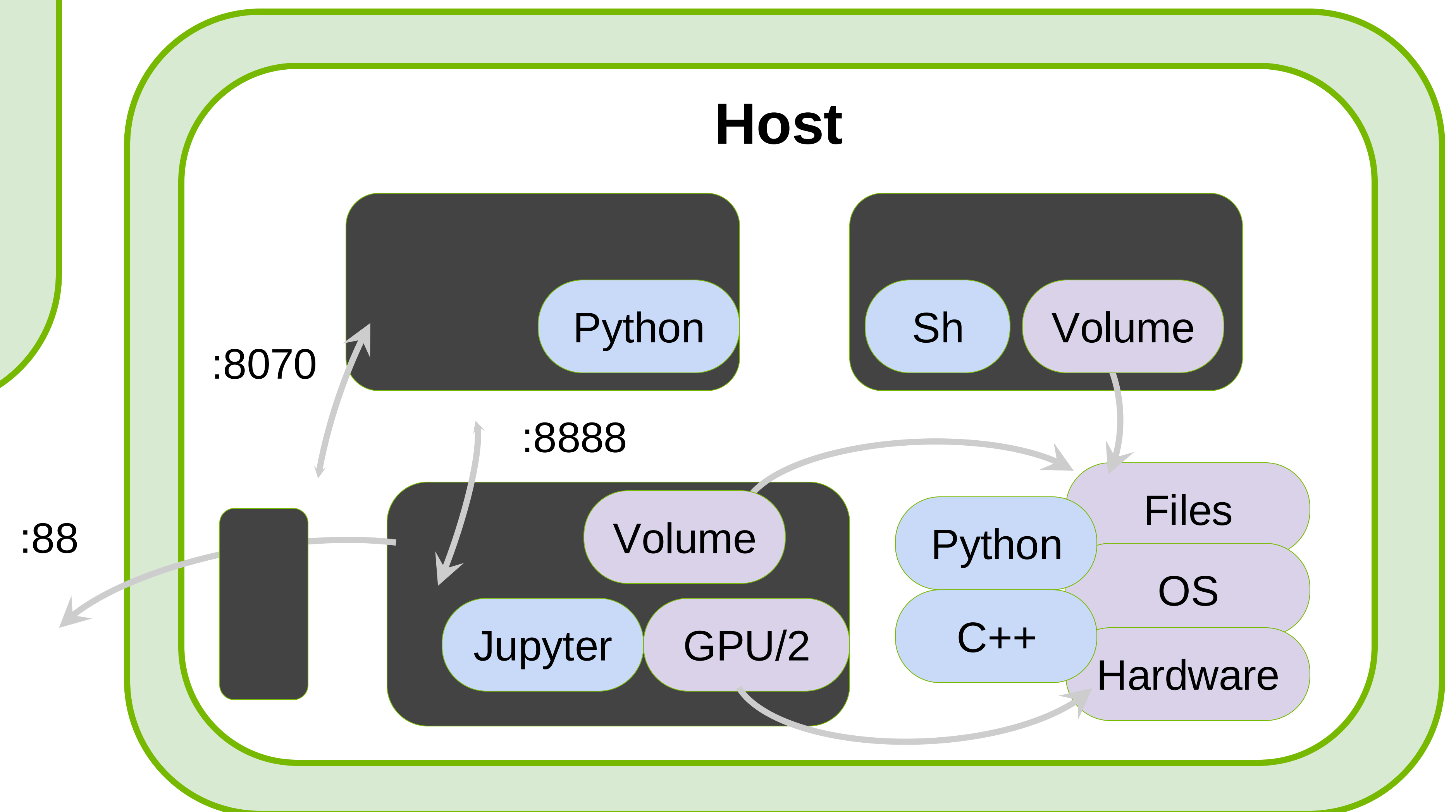
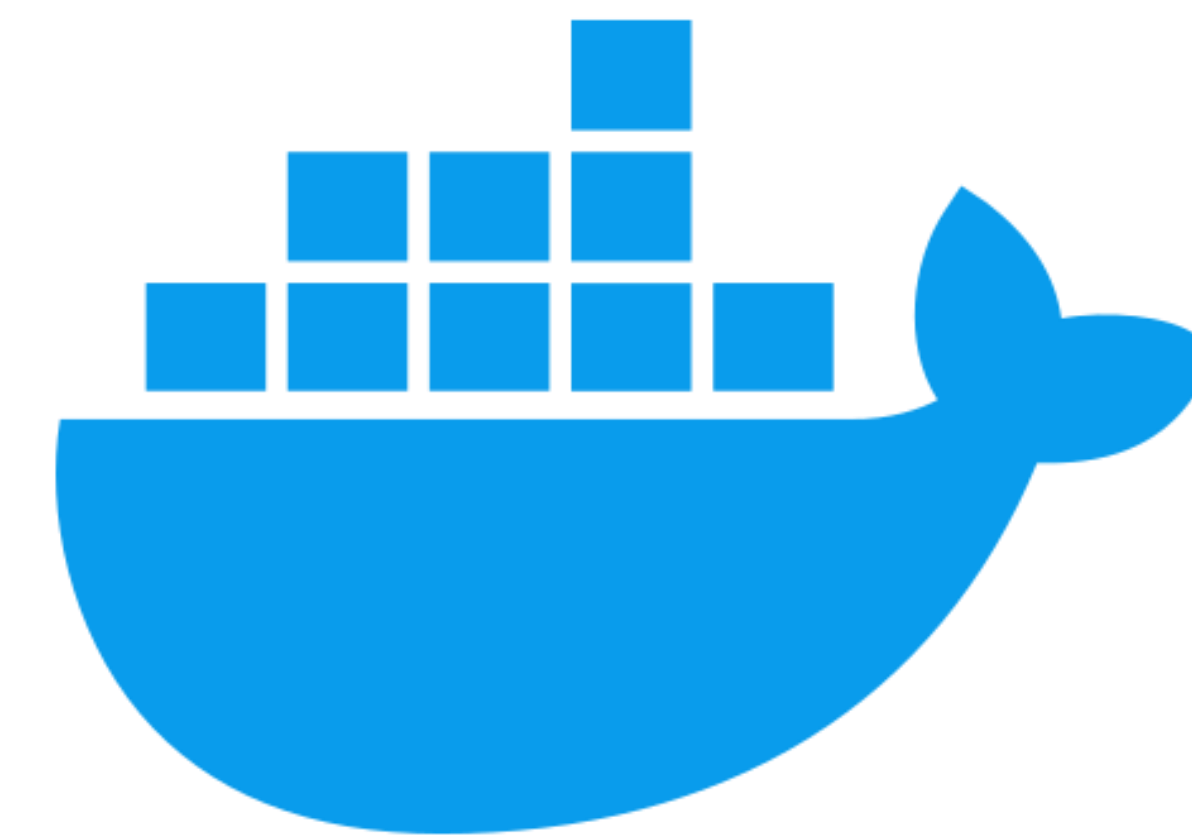
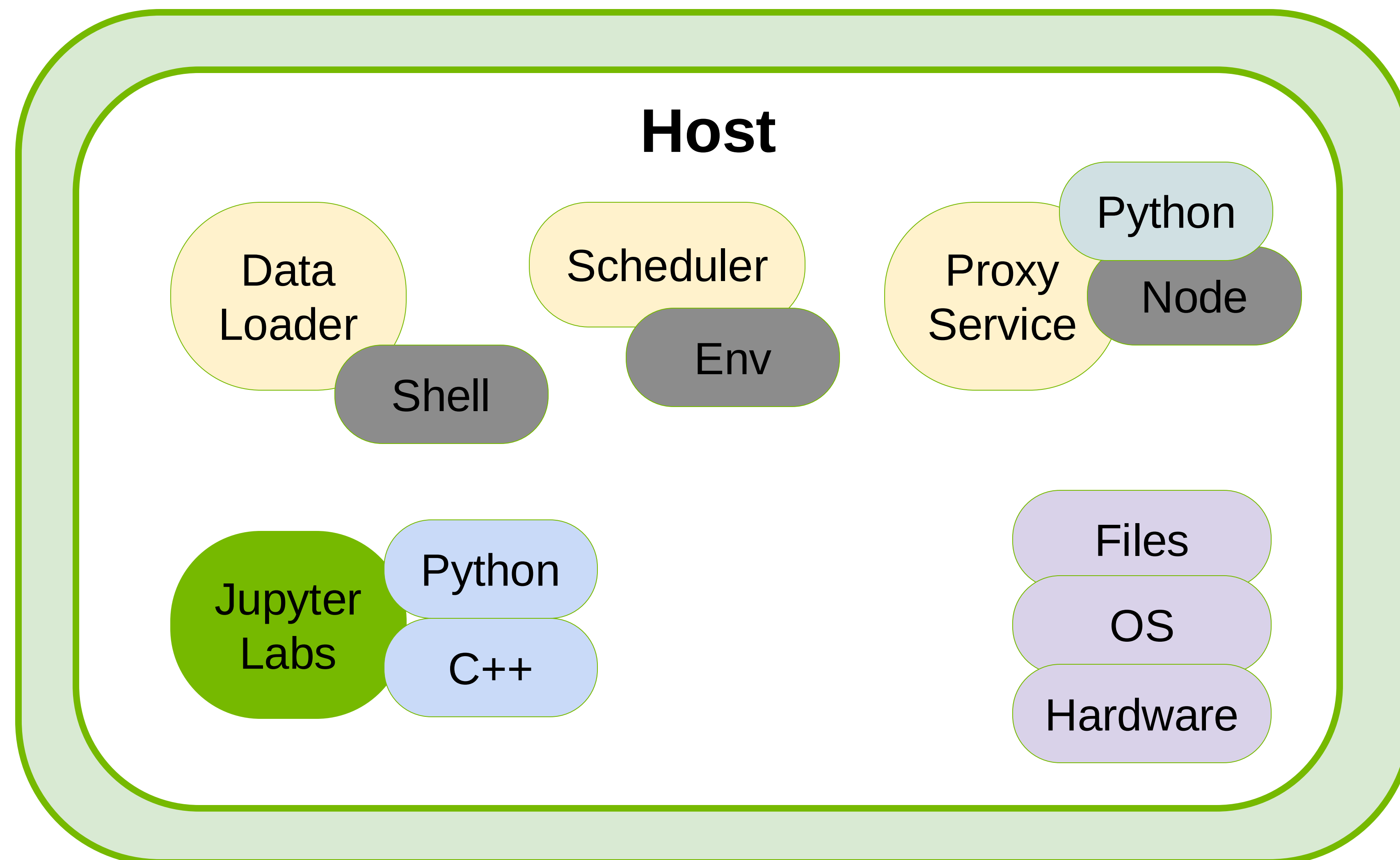


DLI Jupyter Labs Interface



Containerization with Docker

Compartmentalizing Functionality Into Microservices



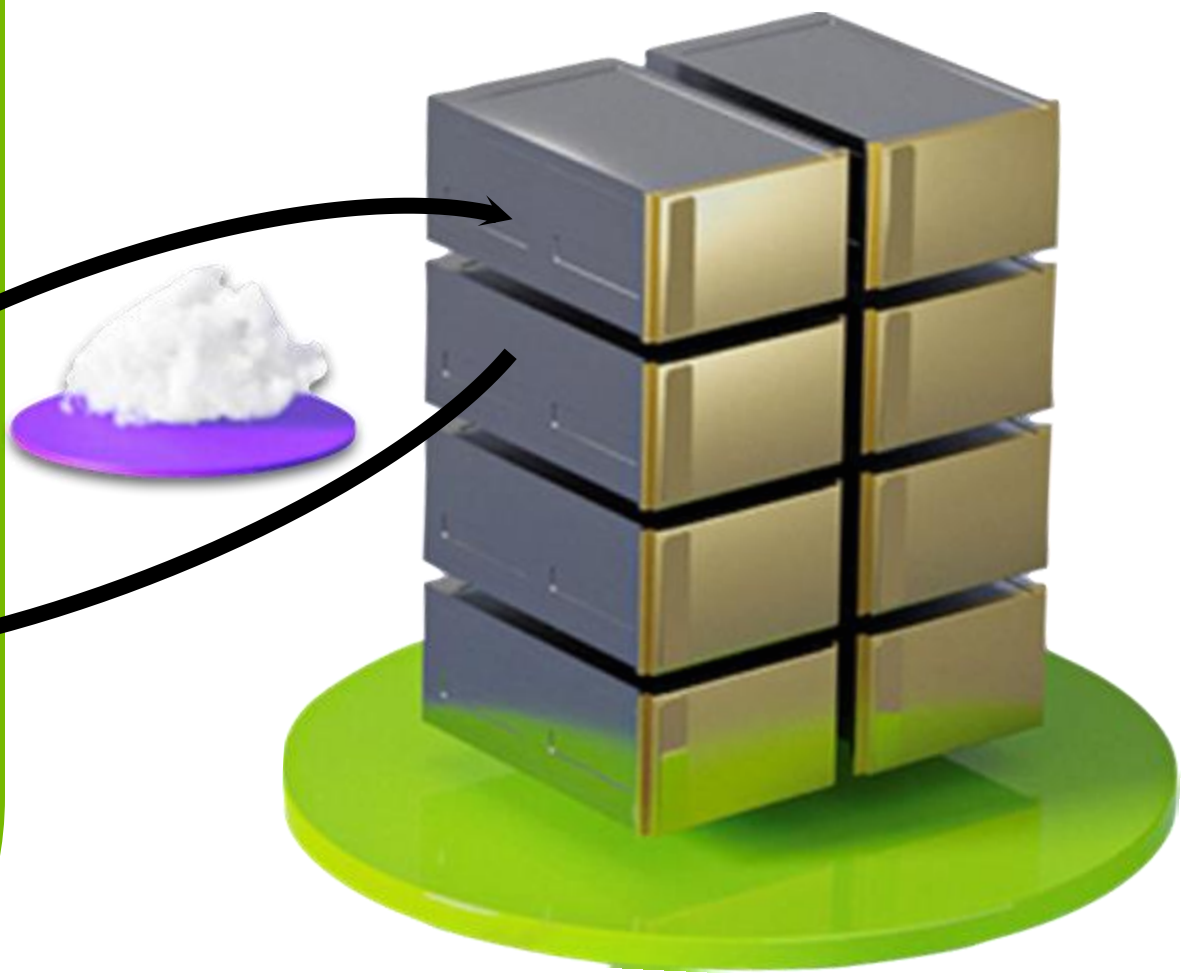
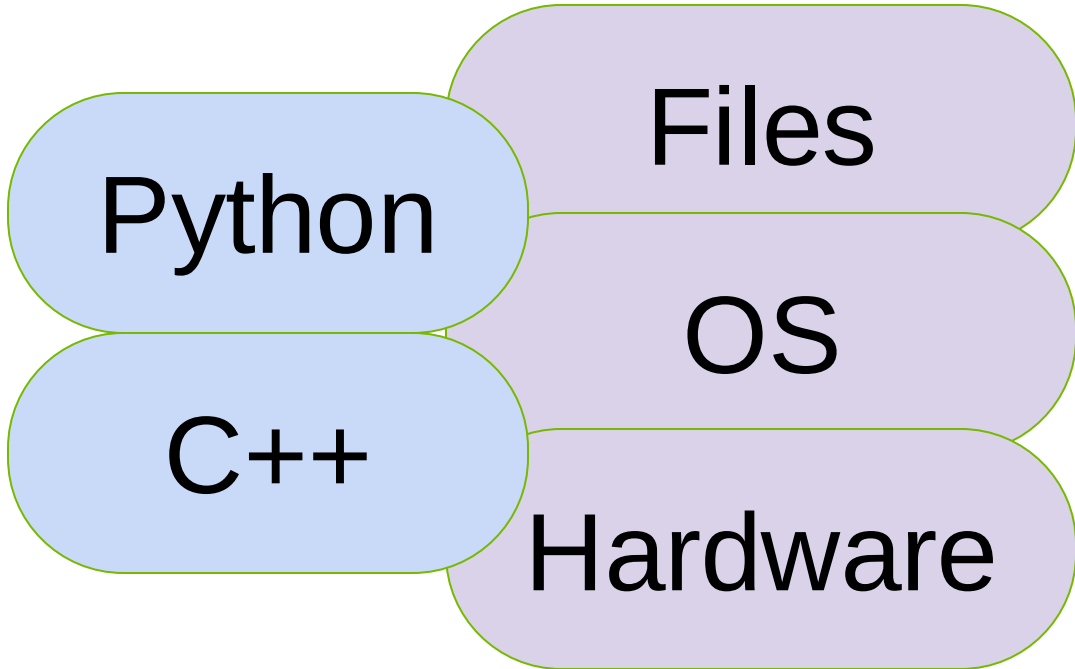
Microservices Workflow



Microservices Workflow

- 1. Allocate Resources
- 2. Define Services
- 3. Construct Containers
- 4. Start Processes

Remote Host

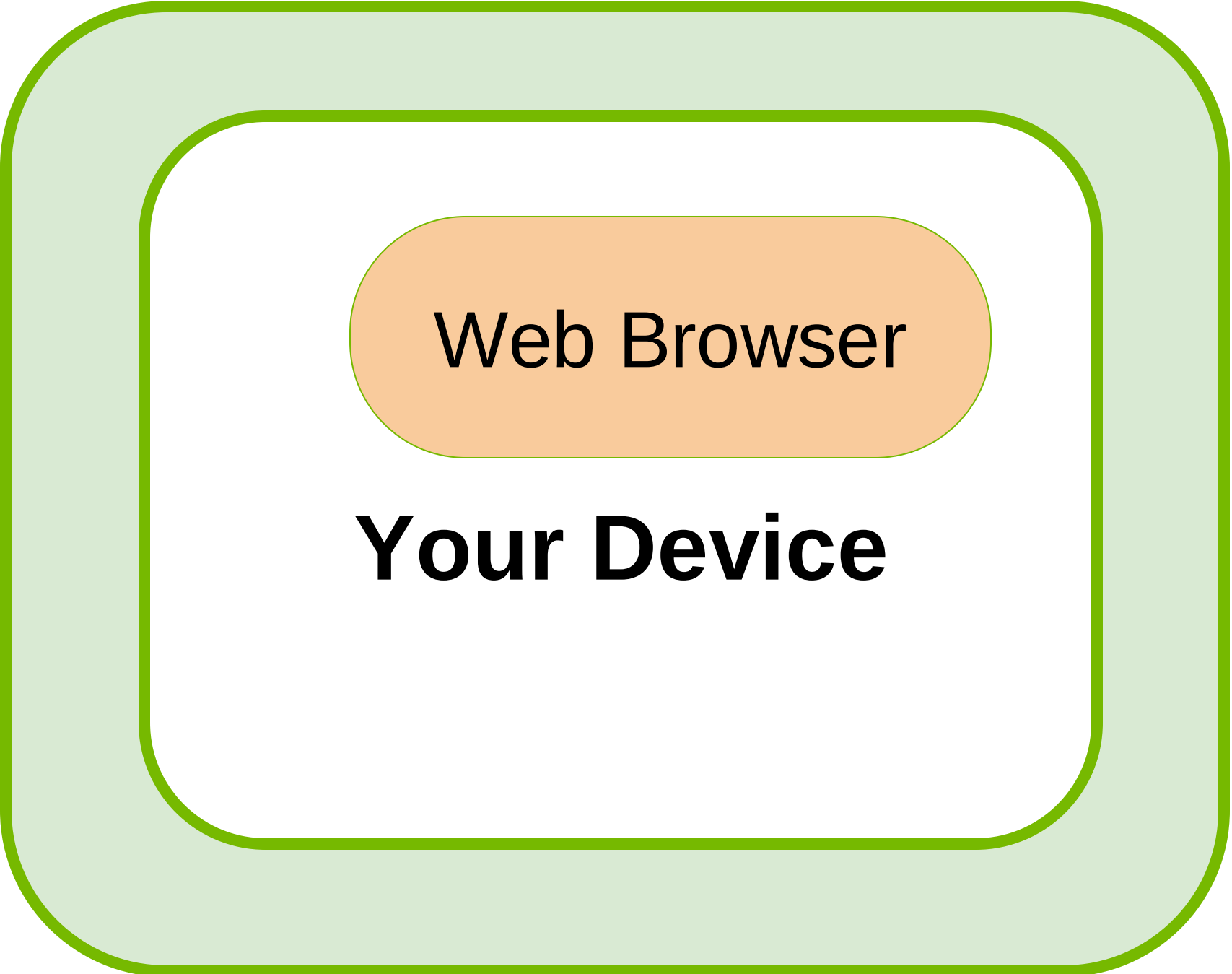
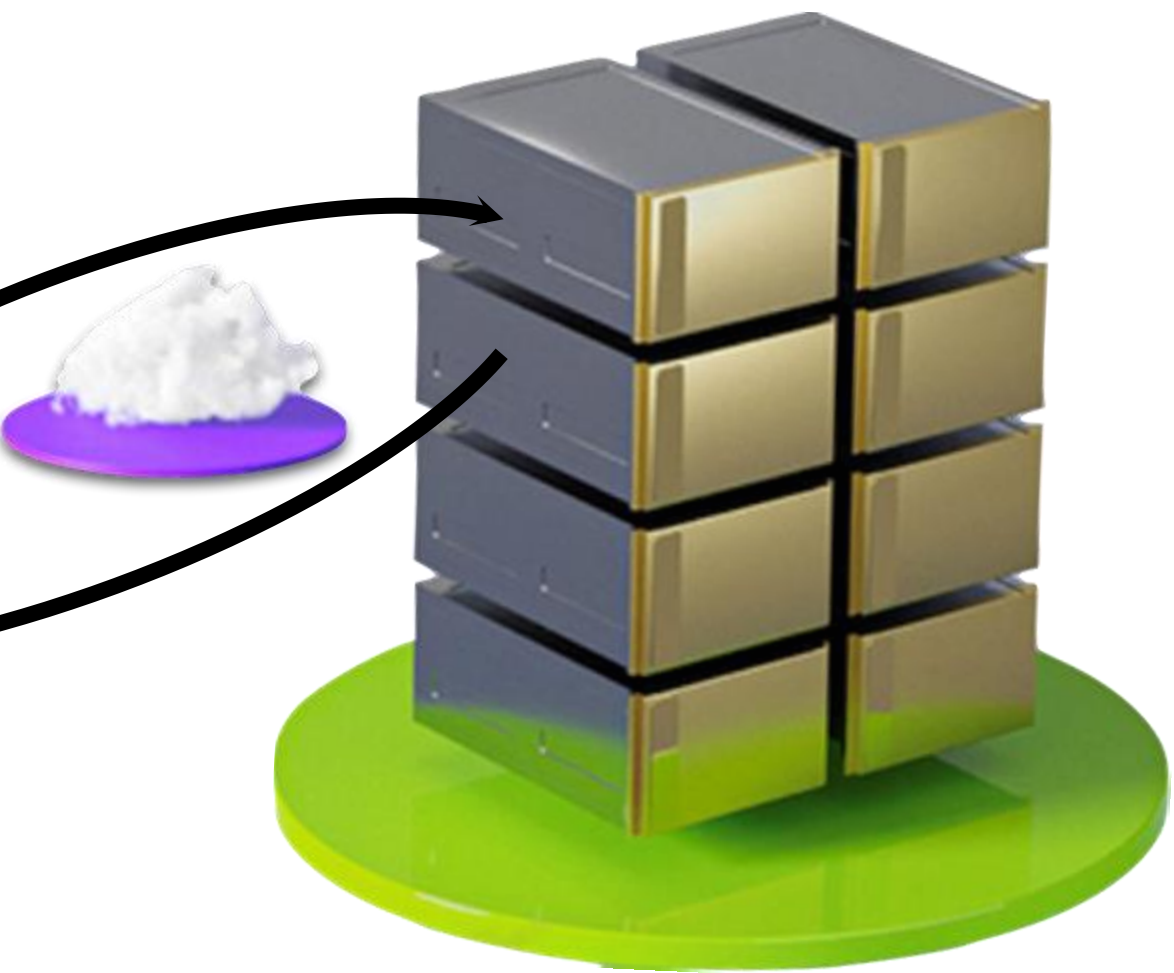
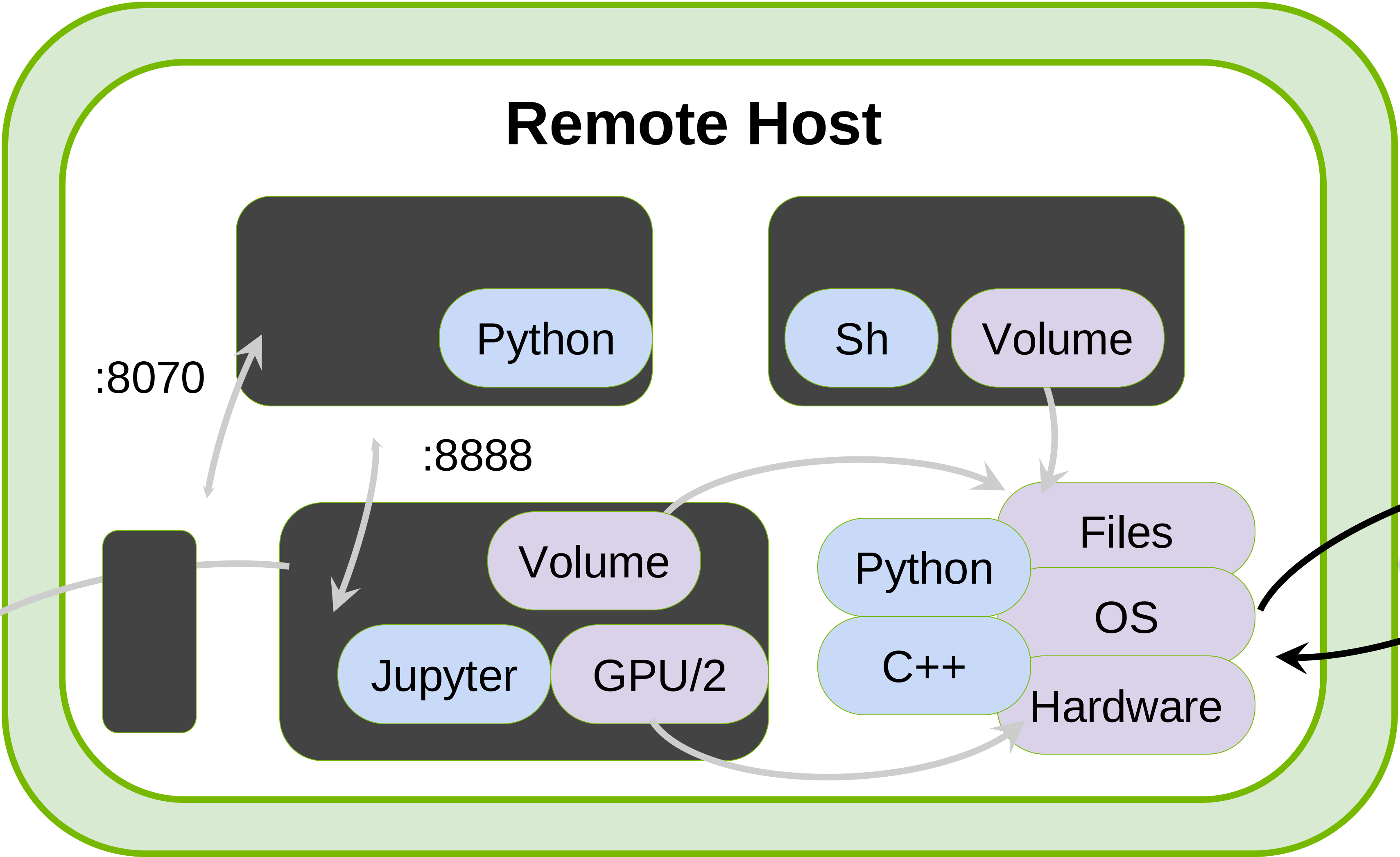


Web Browser

Your Device

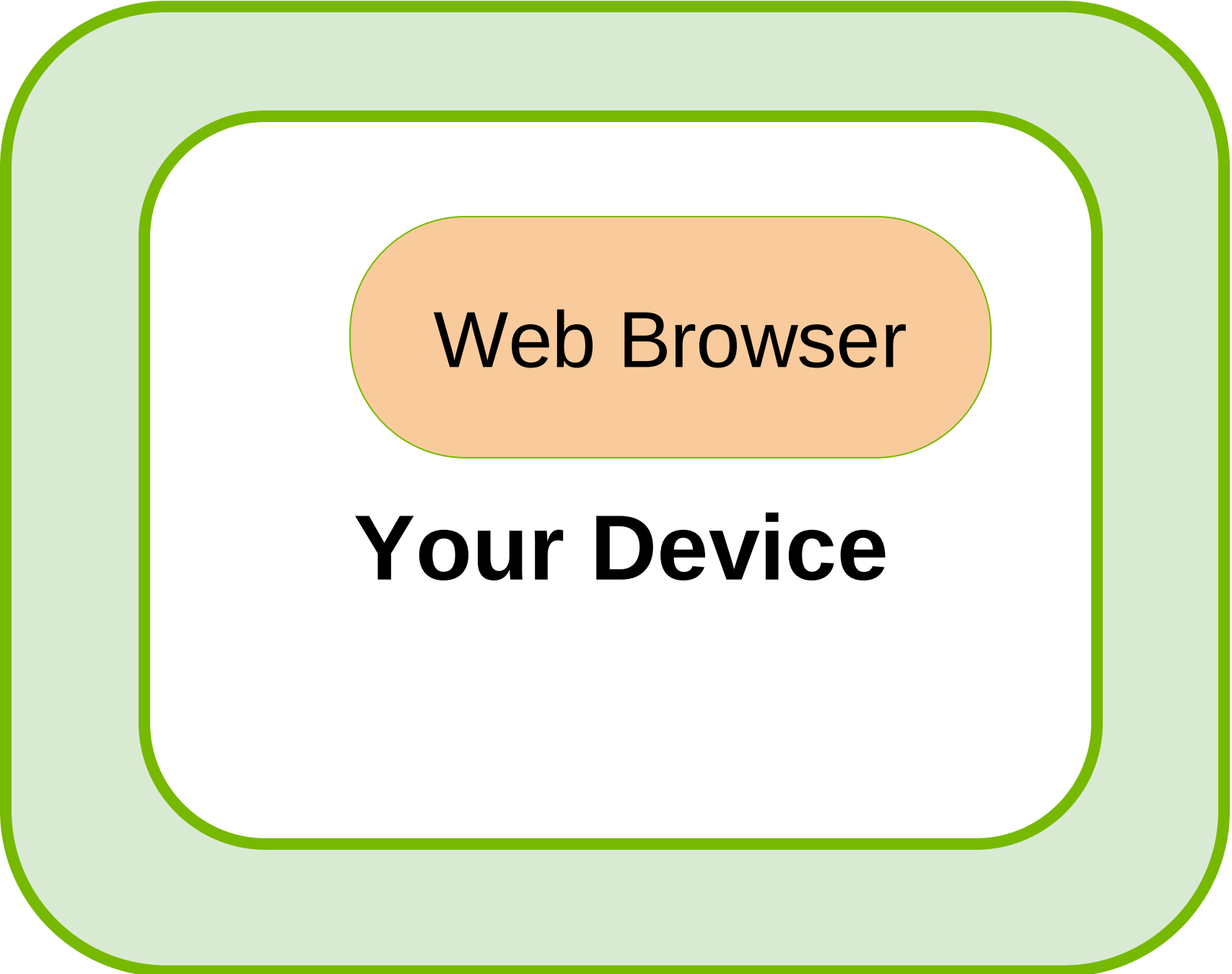
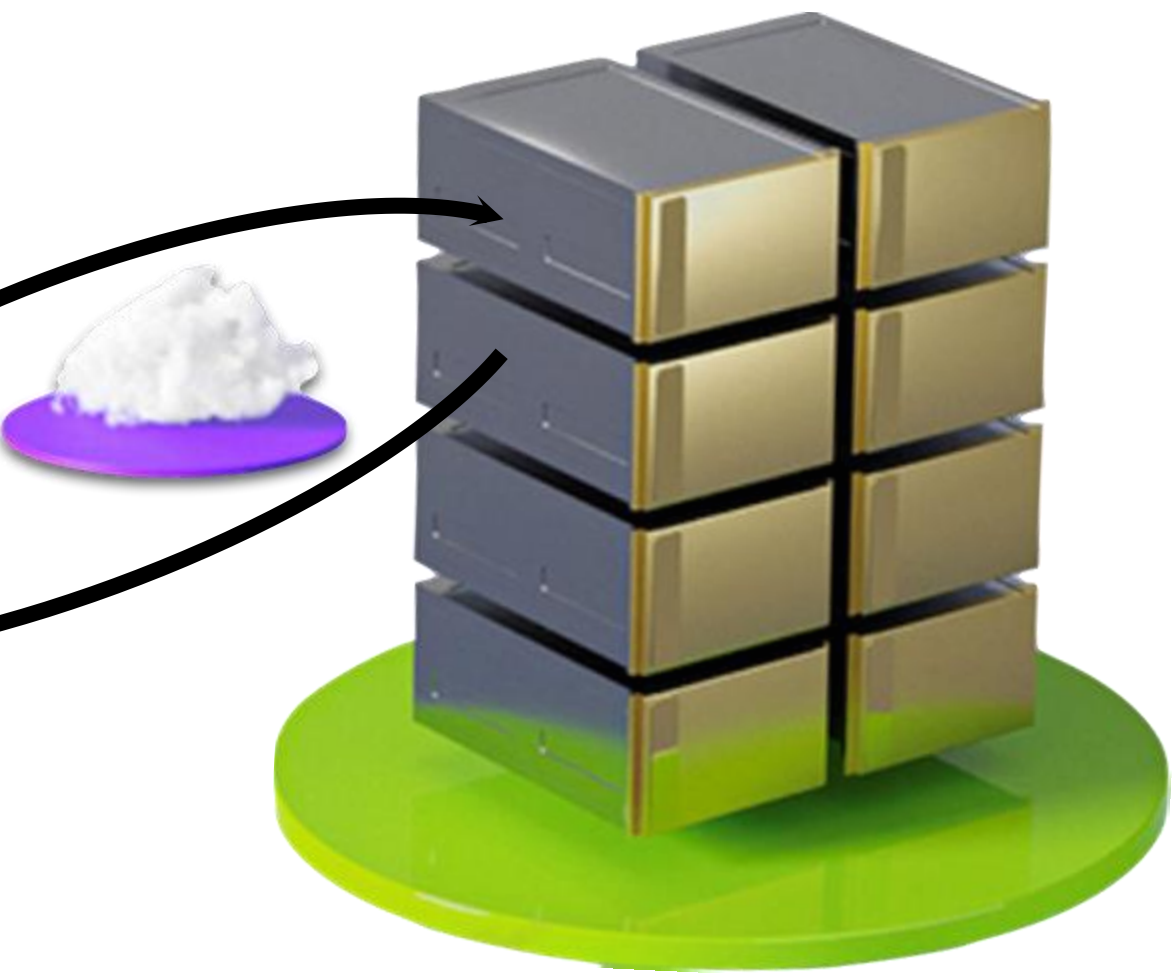
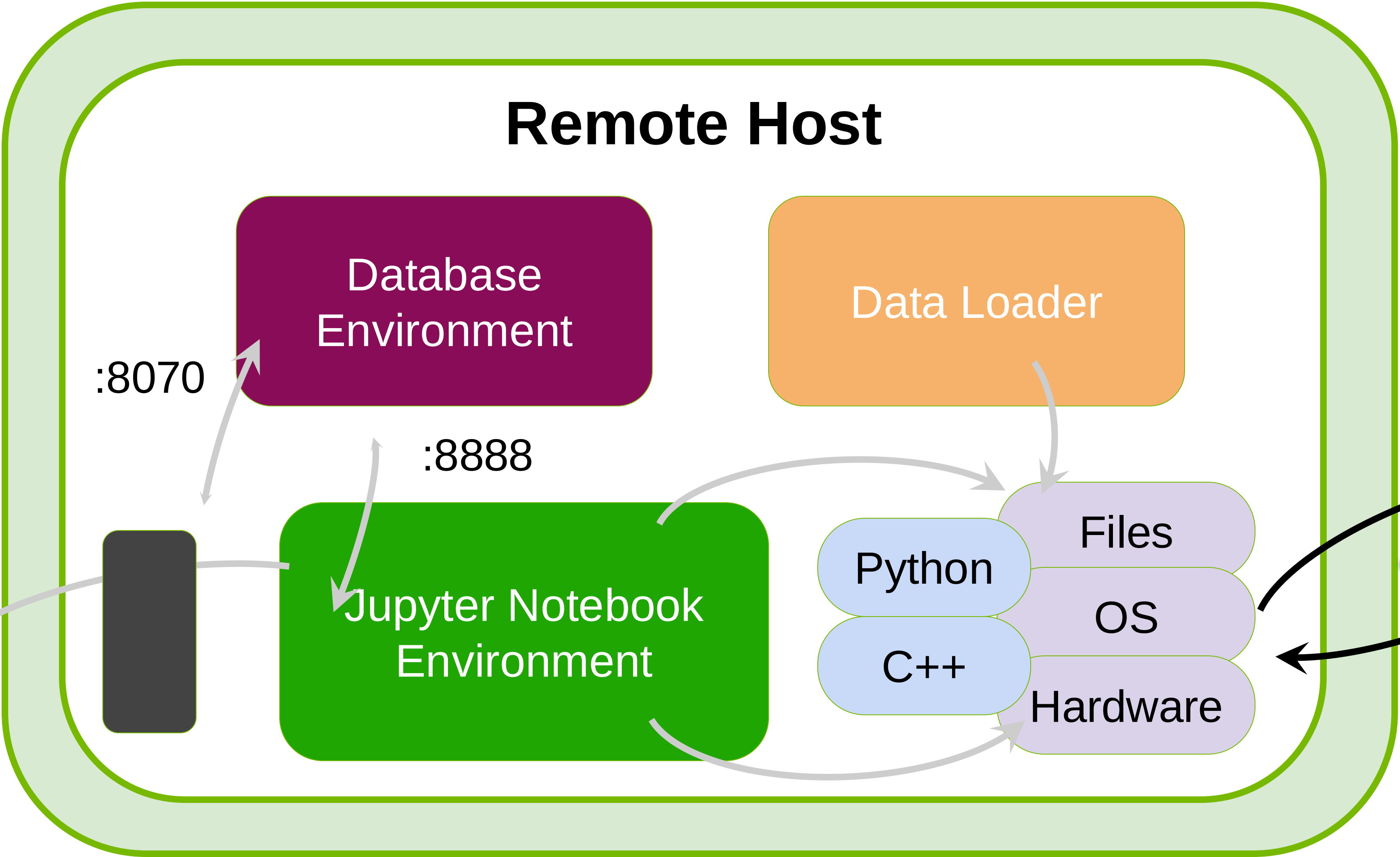
Microservices Workflow

- 1. Allocate Resources
- 2. Define Services
- 3. Construct Containers
- 4. Start Processes

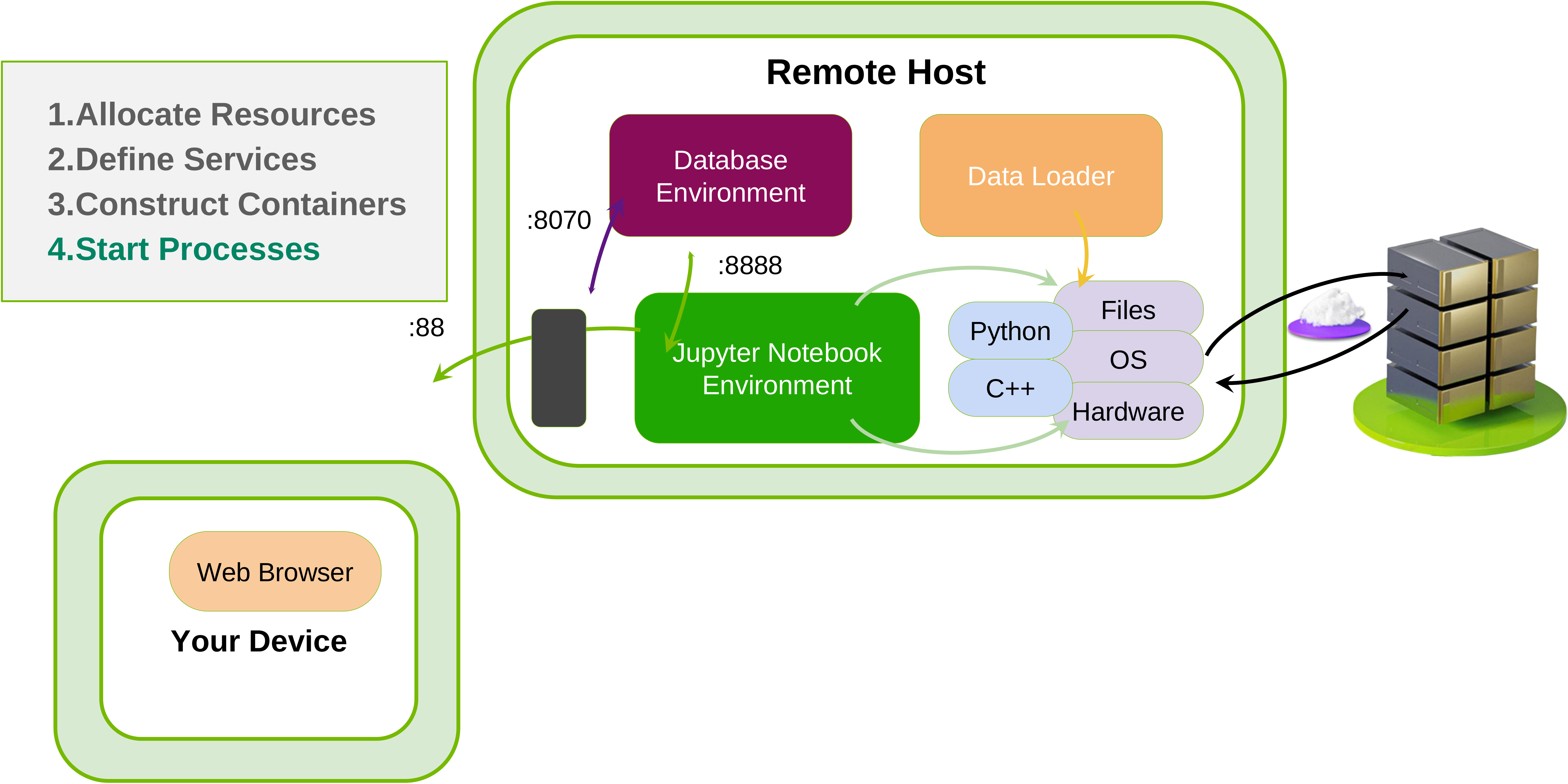


Microservices Workflow

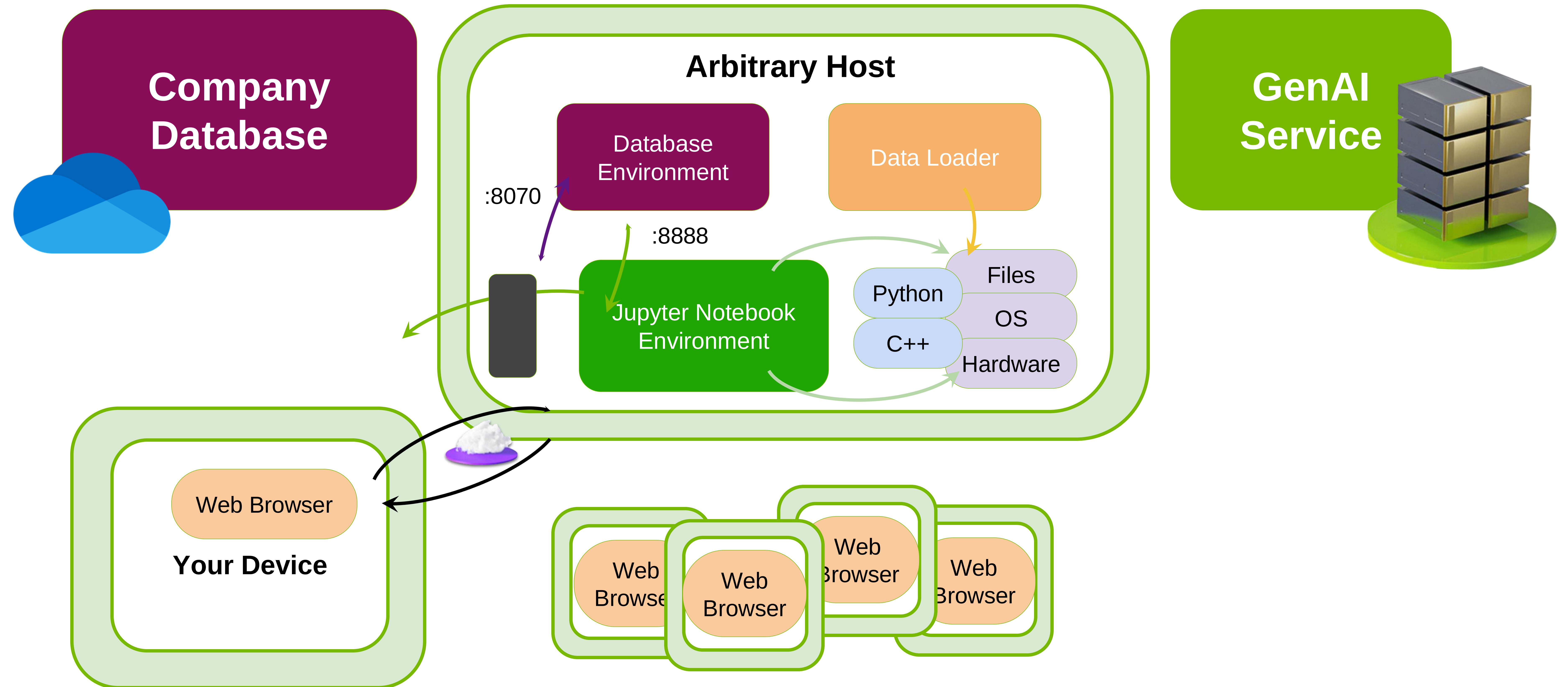
- 1. Allocate Resources
- 2. Define Services
- 3. Construct Containers
- 4. Start Processes



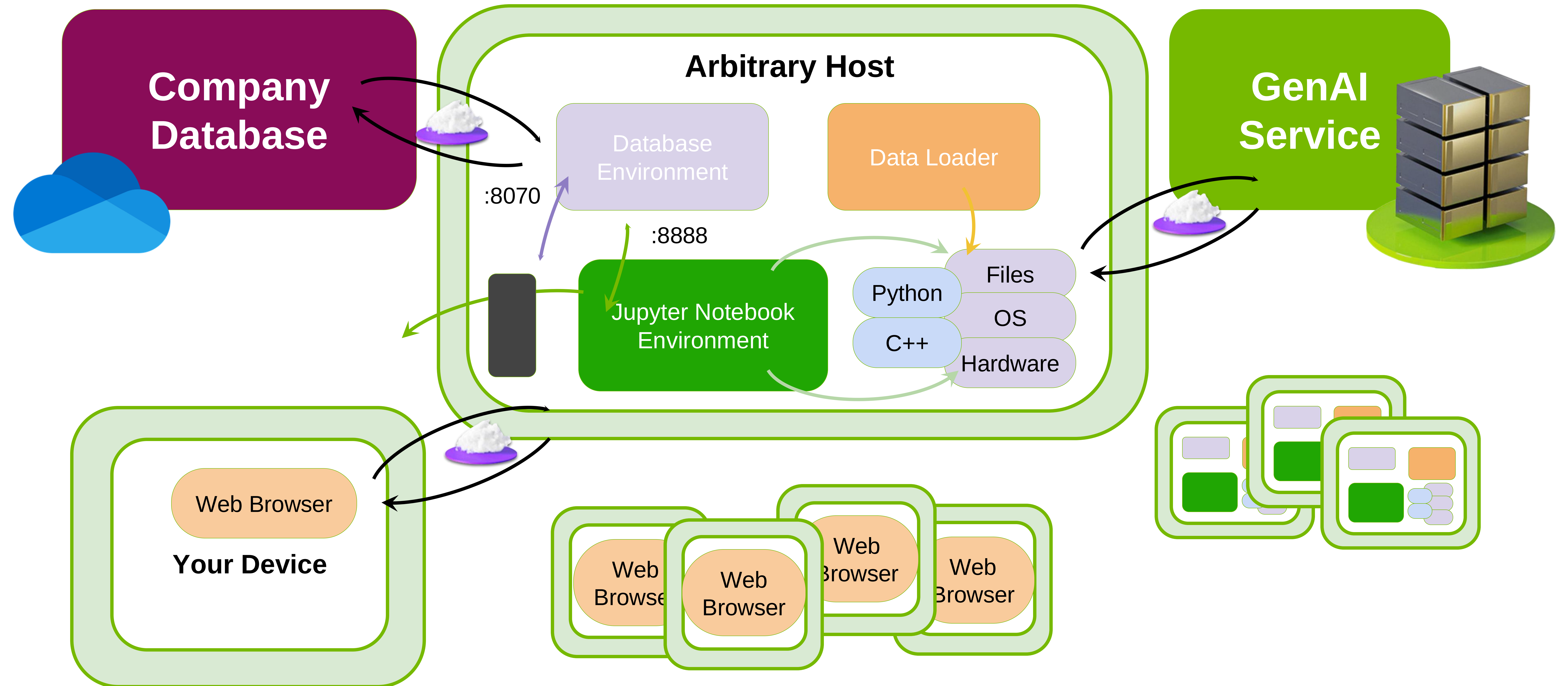
Microservices Workflow



Scaling Containerized Applications

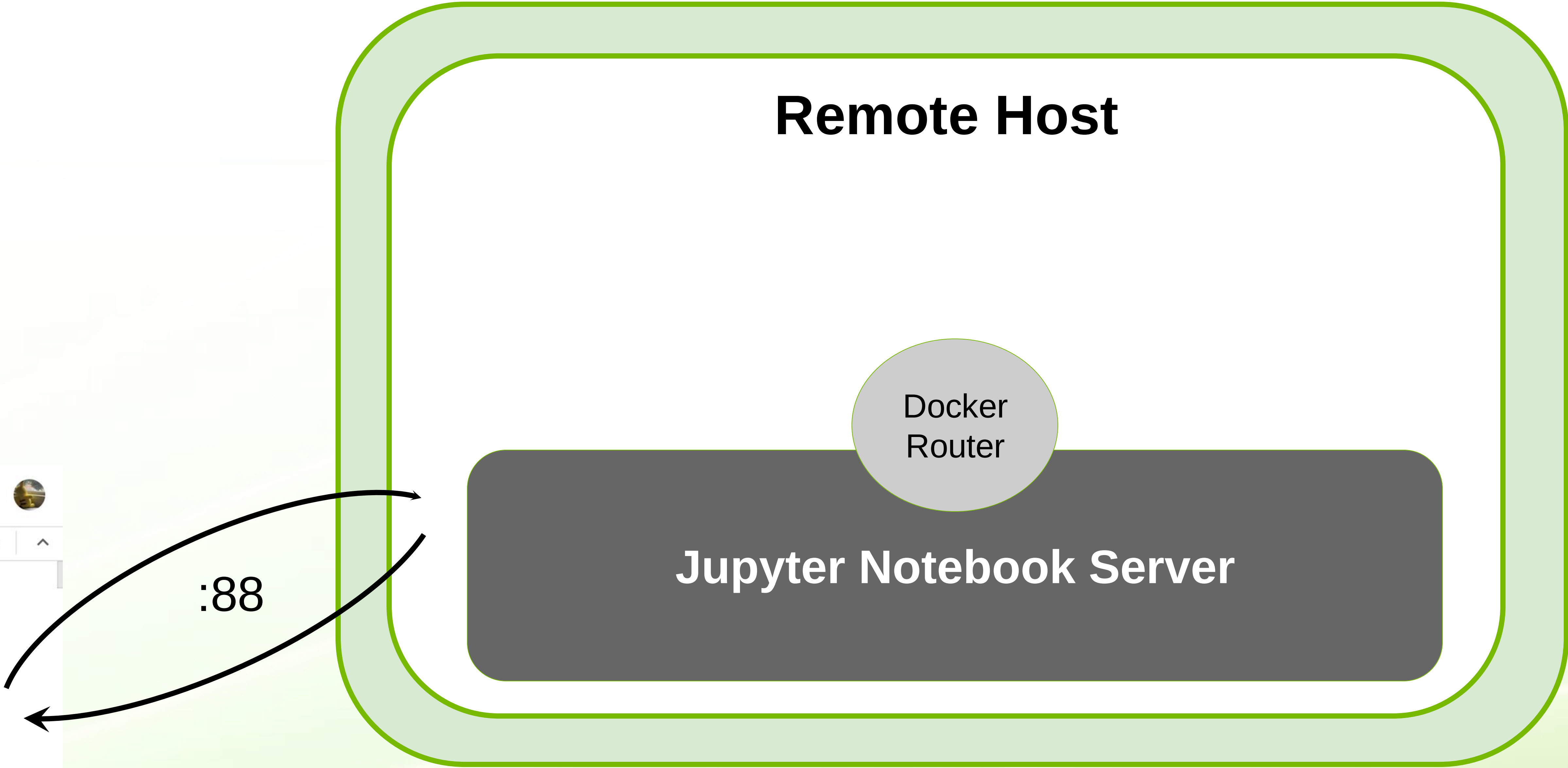
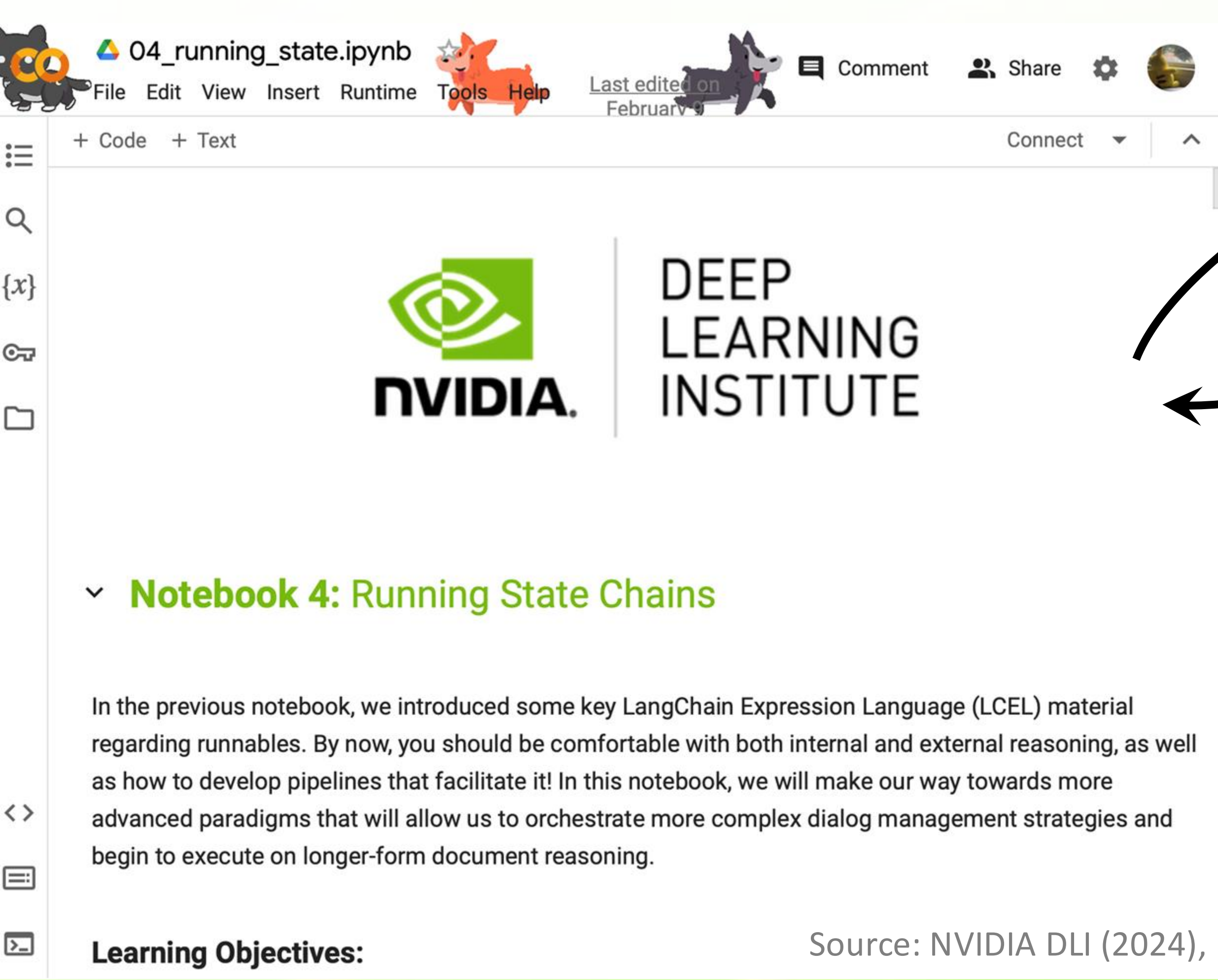


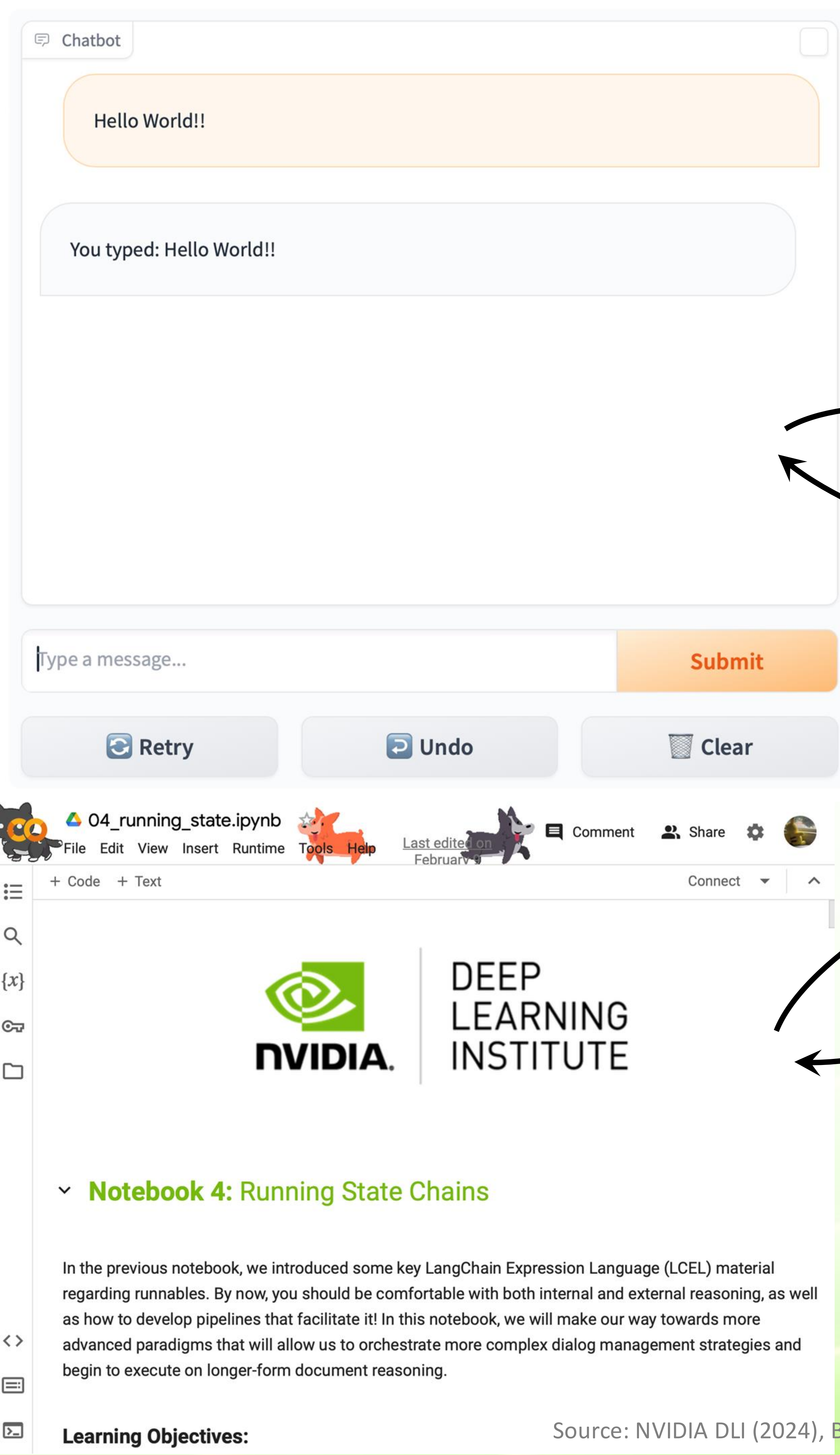
Scaling Containerized Applications



Our Environment

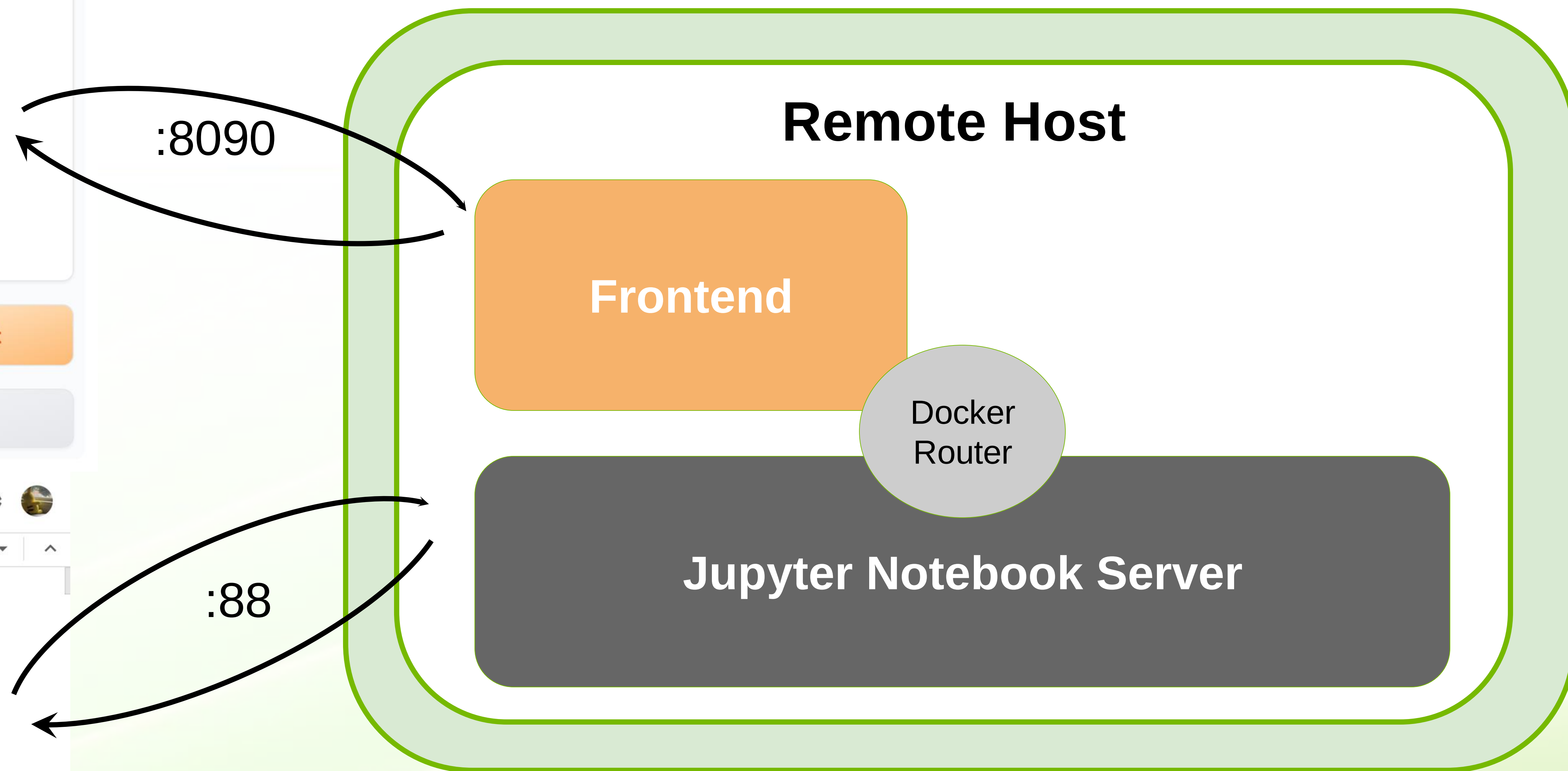
Jupyter Notebooks



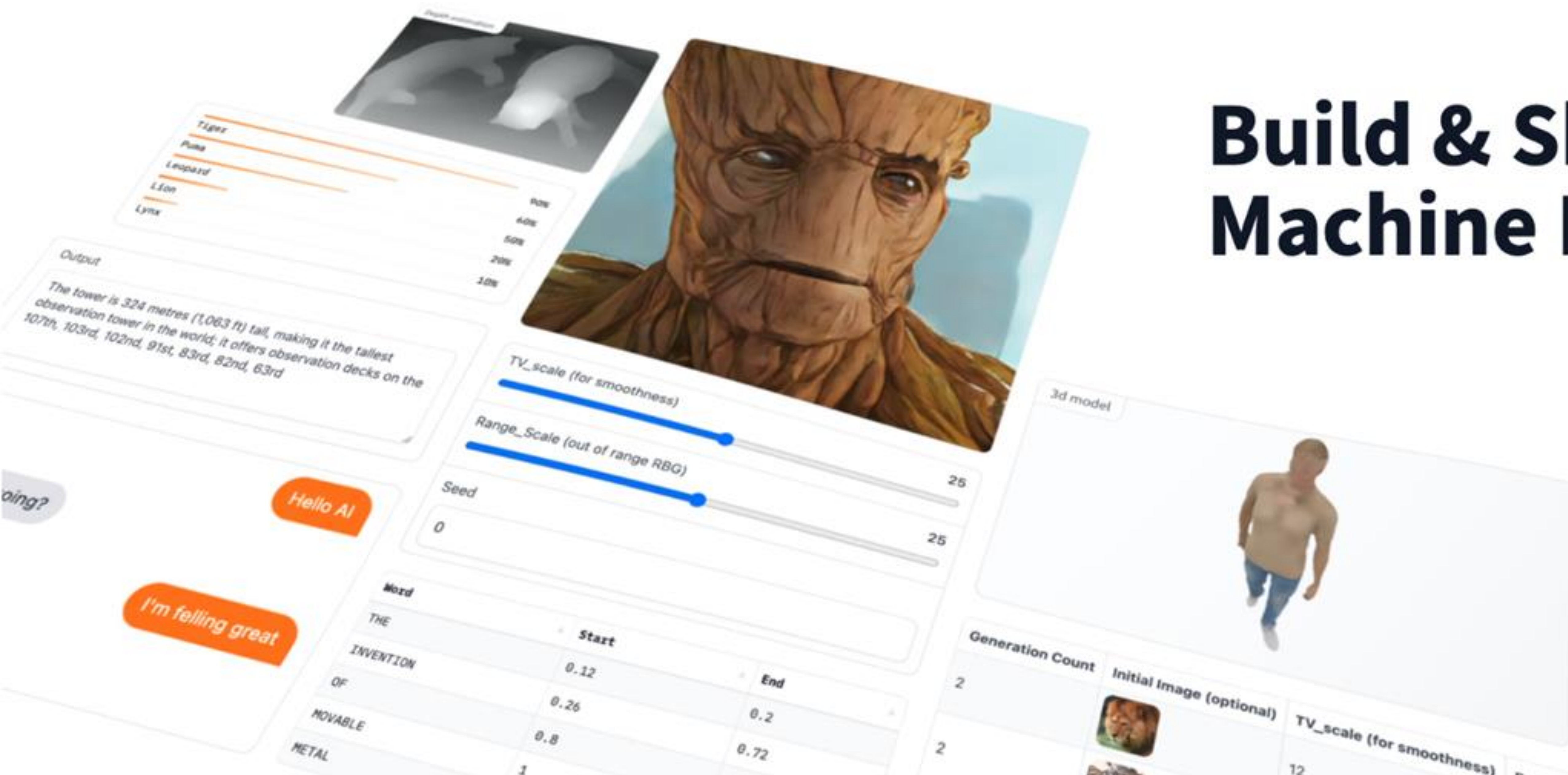


Our Environment

Jupyter Notebooks + Frontend



Gradio

[⚡ Quickstart](#)[📖 Docs](#)[💡 Guides](#)[🎨 Playground^{NEW}](#)[👋 Community ▾](#)

Build & Share Delightful Machine Learning Apps

Gradio is the fastest way to demo your machine learning model with a friendly web interface so that anyone can use it, anywhere!

[Get Started](#)[🌟 Star](#)[23997](#)

Simple Gradio ChatInterface

Chatbot

Hello World!!

You typed: Hello World!!

Type a message...

Submit

Retry

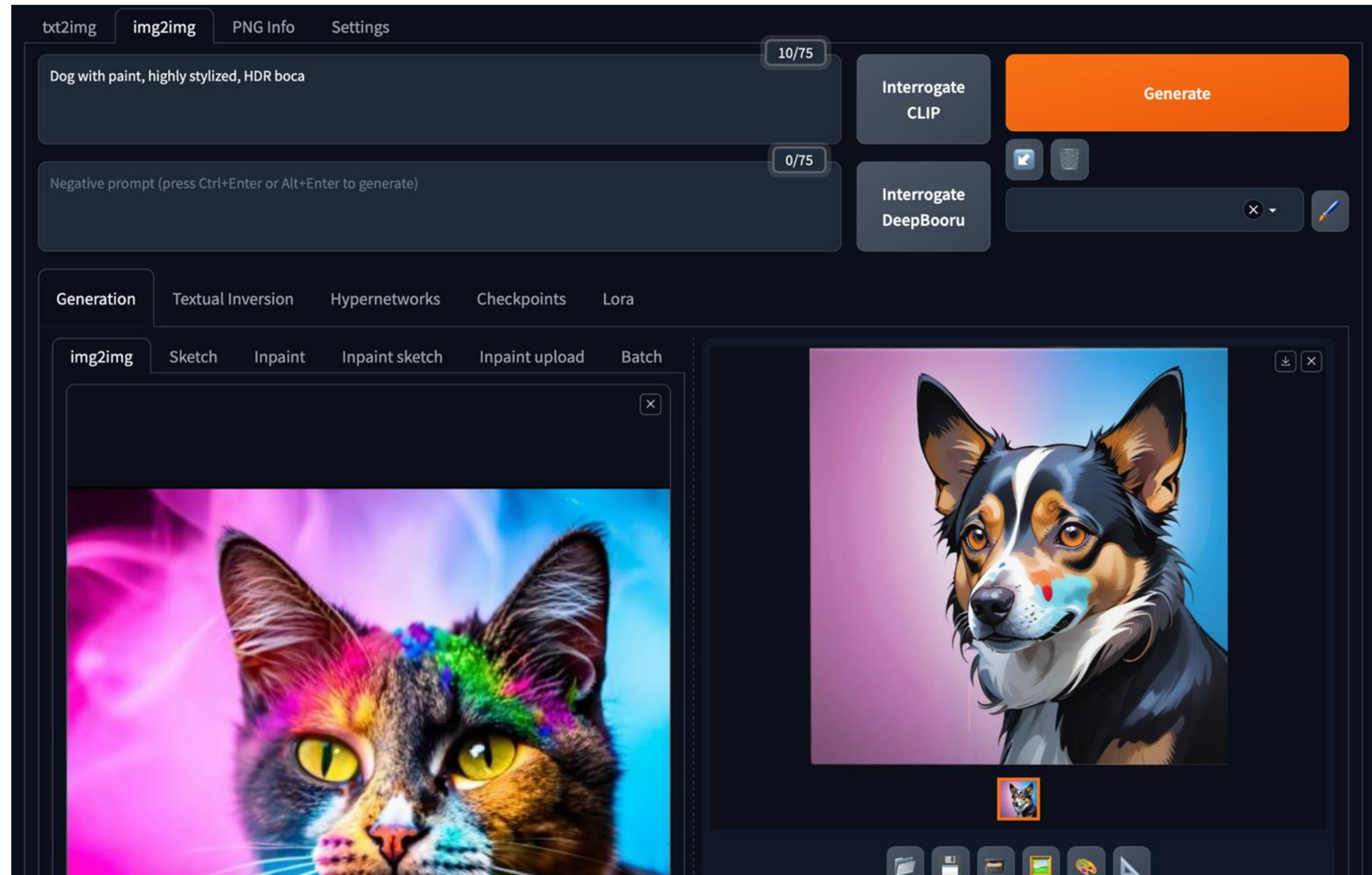
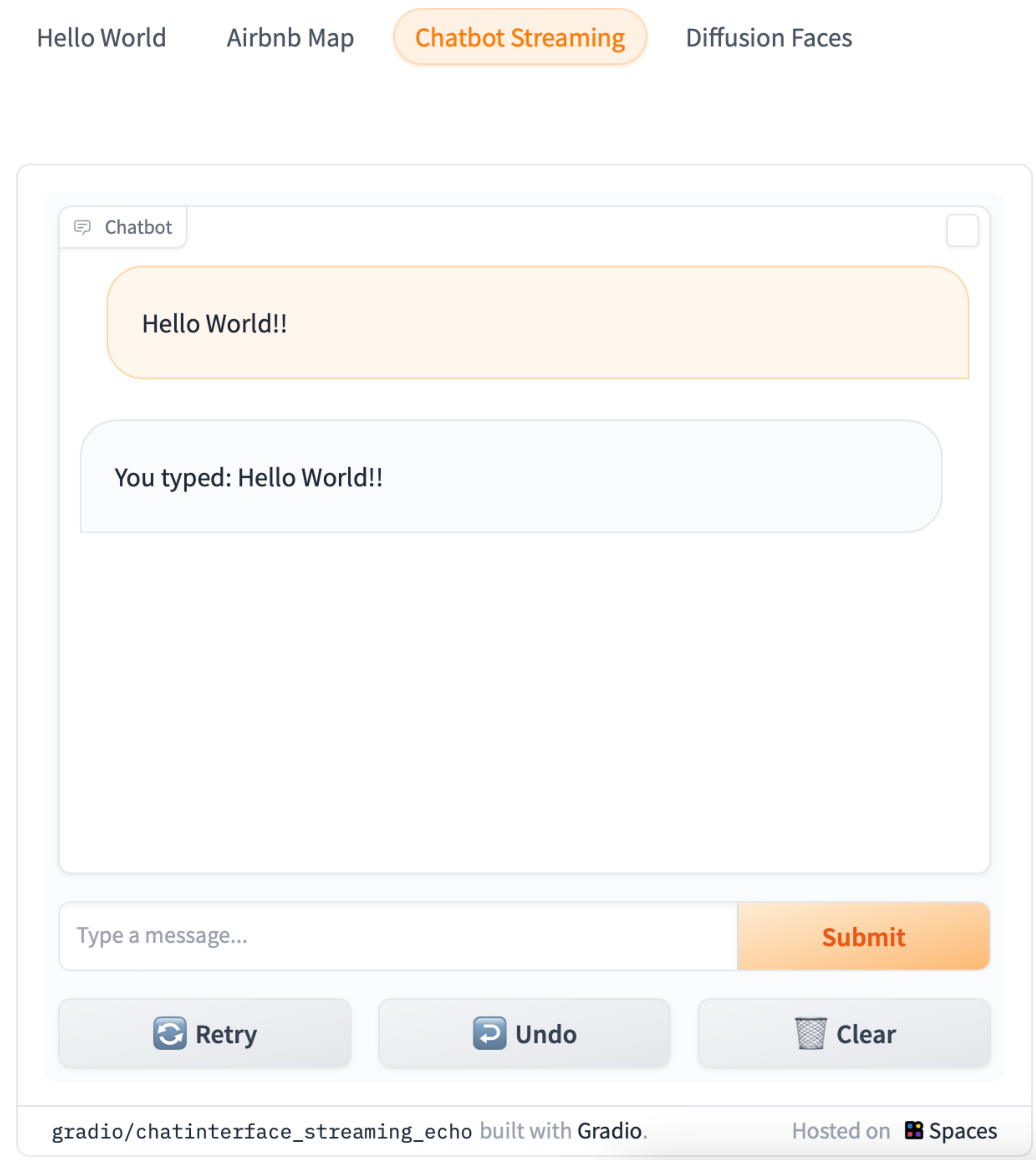
Undo

Clear

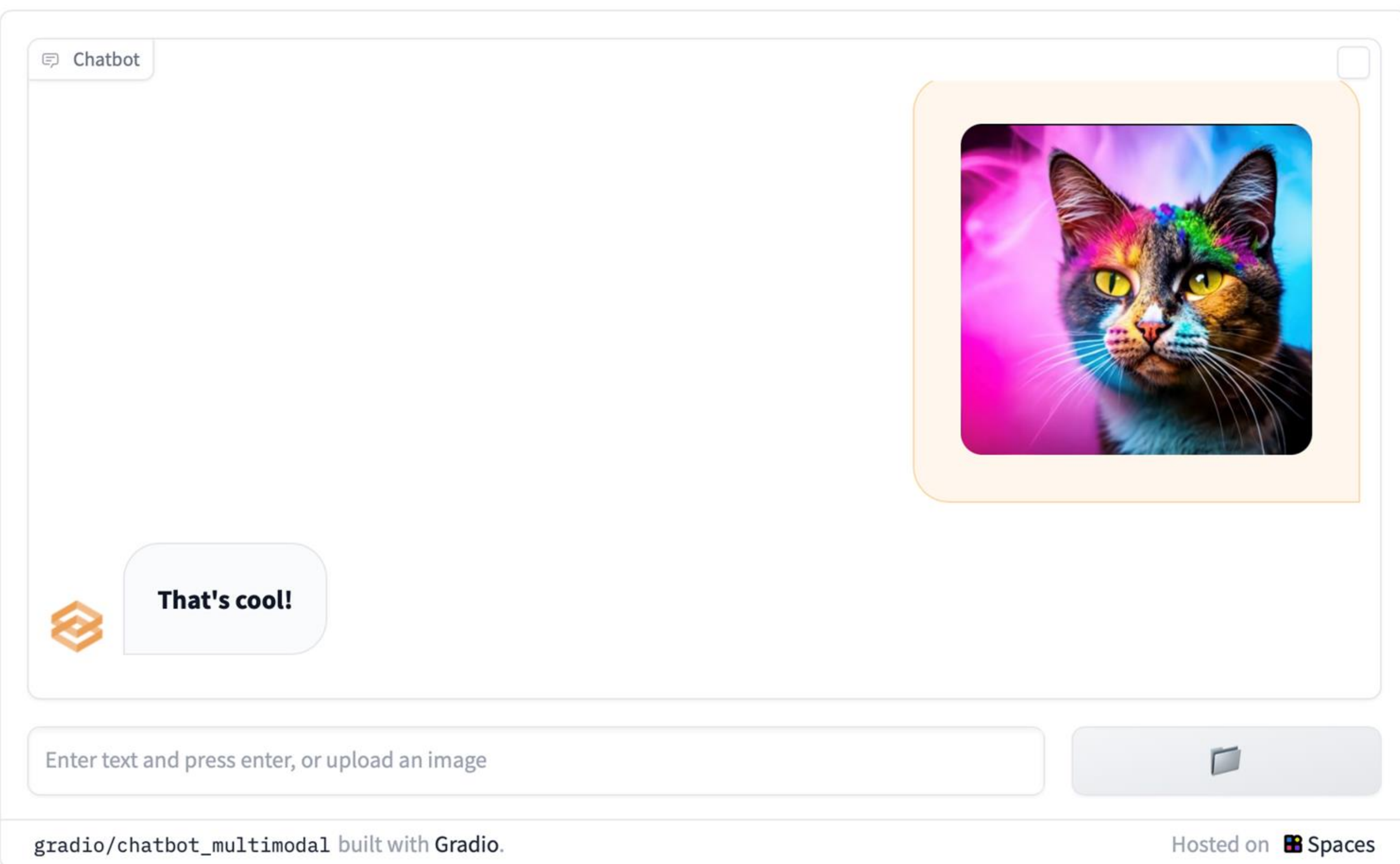
gradio/chatinterface_streaming_echo built with Gradio. Hosted on Spaces

```
1 import time
2 import gradio as gr
3
4 def slow_echo(message, history):
5     for i in range(len(message)):
6         time.sleep(0.05)
7         yield "You typed: " + message[: i+1]
8
9 demo = gr.ChatInterface(slow_echo).queue()
10
11 if __name__ == "__main__":
12     demo.launch()
13
```

Gradio in HuggingFace Spaces



Custom Gradio Block Interface



```
with gr.Blocks() as demo:
    chatbot = gr.Chatbot(
        [],
        elem_id="chatbot",
        bubble_full_width=False,
        avatar_images=(None, (os.path.join(os.path.dirname(__file__), "avatar.png"))),
    )

    with gr.Row():
        txt = gr.Textbox(
            scale=4,
            show_label=False,
            placeholder="Enter text and press enter, or upload an image",
            container=False,
        )
        btn = gr.UploadButton("📁", file_types=["image", "video", "audio"])

    txt_msg = txt.submit(add_text, [chatbot, txt], [chatbot, txt], queue=False).then(
        bot, chatbot, chatbot, api_name="bot_response"
    )
    txt_msg.then(lambda: gr.Textbox(interactive=True), None, [txt], queue=False)
    file_msg = btn.upload(add_file, [chatbot, btn], [chatbot], queue=False).then(
        bot, chatbot, chatbot
    )

demo.queue()
demo.launch()
```

Chatbot

Hello World!!

You typed: Hello World!!

Type a message...

Submit

Retry

Undo

Clear

04_running_state.ipynb

File Edit View Insert Runtime Tools Help

Last edited on February 9

Comment Share

+ Code + Text

Connect

DEEP LEARNING INSTITUTE

▼ Notebook 4: Running State Chains

In the previous notebook, we introduced some key LangChain Expression Language (LCEL) material regarding runnables. By now, you should be comfortable with both internal and external reasoning, as well as how to develop pipelines that facilitate it! In this notebook, we will make our way towards more advanced paradigms that will allow us to orchestrate more complex dialog management strategies and begin to execute on longer-form document reasoning.

Learning Objectives:

Source: NVIDIA DLI (2024), Building RAG Agents with LLMs, https://learn.nvidia.com/courses/course-detail?course_id=course-v1:DLI+S-FX-15+V1

Our Environment

Jupyter Notebooks + Frontend

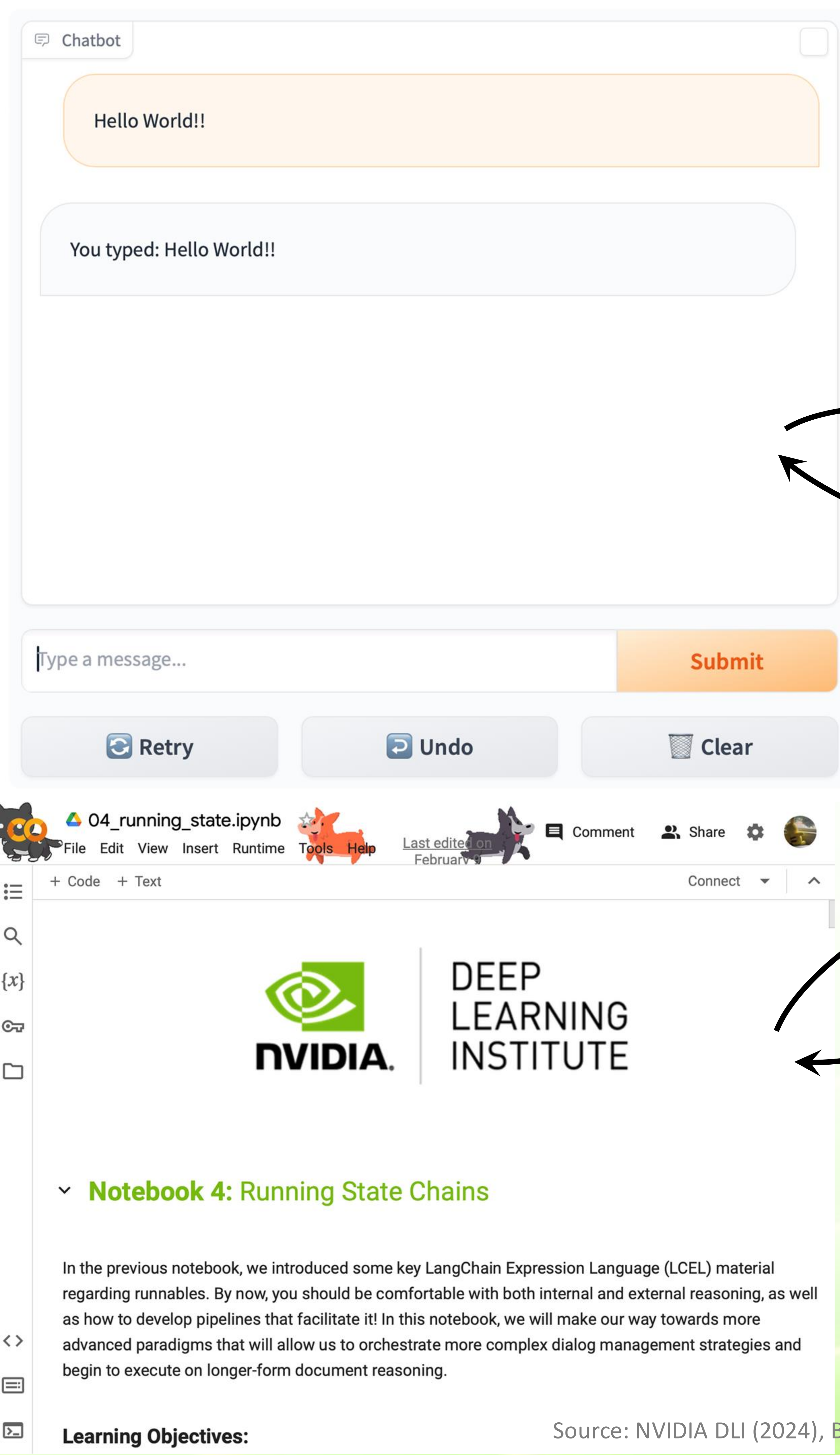
The diagram illustrates the environment architecture. A large light green rounded rectangle represents the **Remote Host**. Inside this host, there are two main components: an orange rounded rectangle labeled **Frontend** and a dark gray rounded rectangle labeled **Jupyter Notebook Server**. A gray oval labeled **Docker Router** is positioned between the Frontend and the Jupyter Notebook Server. Two curved arrows originate from the left side of the diagram and point towards the Remote Host. The top arrow is labeled **:8090** and points to the Frontend. The bottom arrow is labeled **:88** and points to the Jupyter Notebook Server.

83



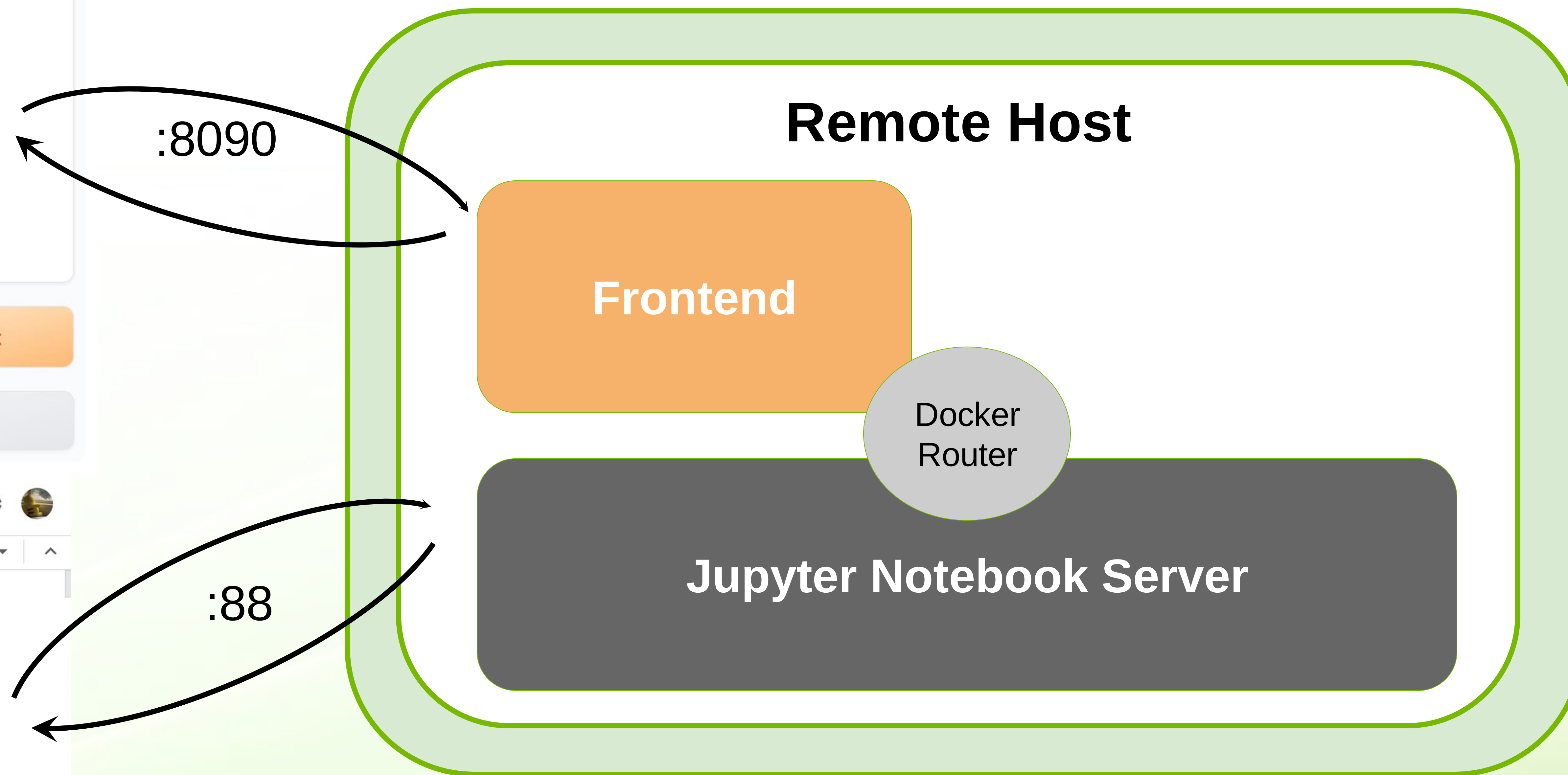
Building RAG Agents with LLMs

Part 2: LLM Services



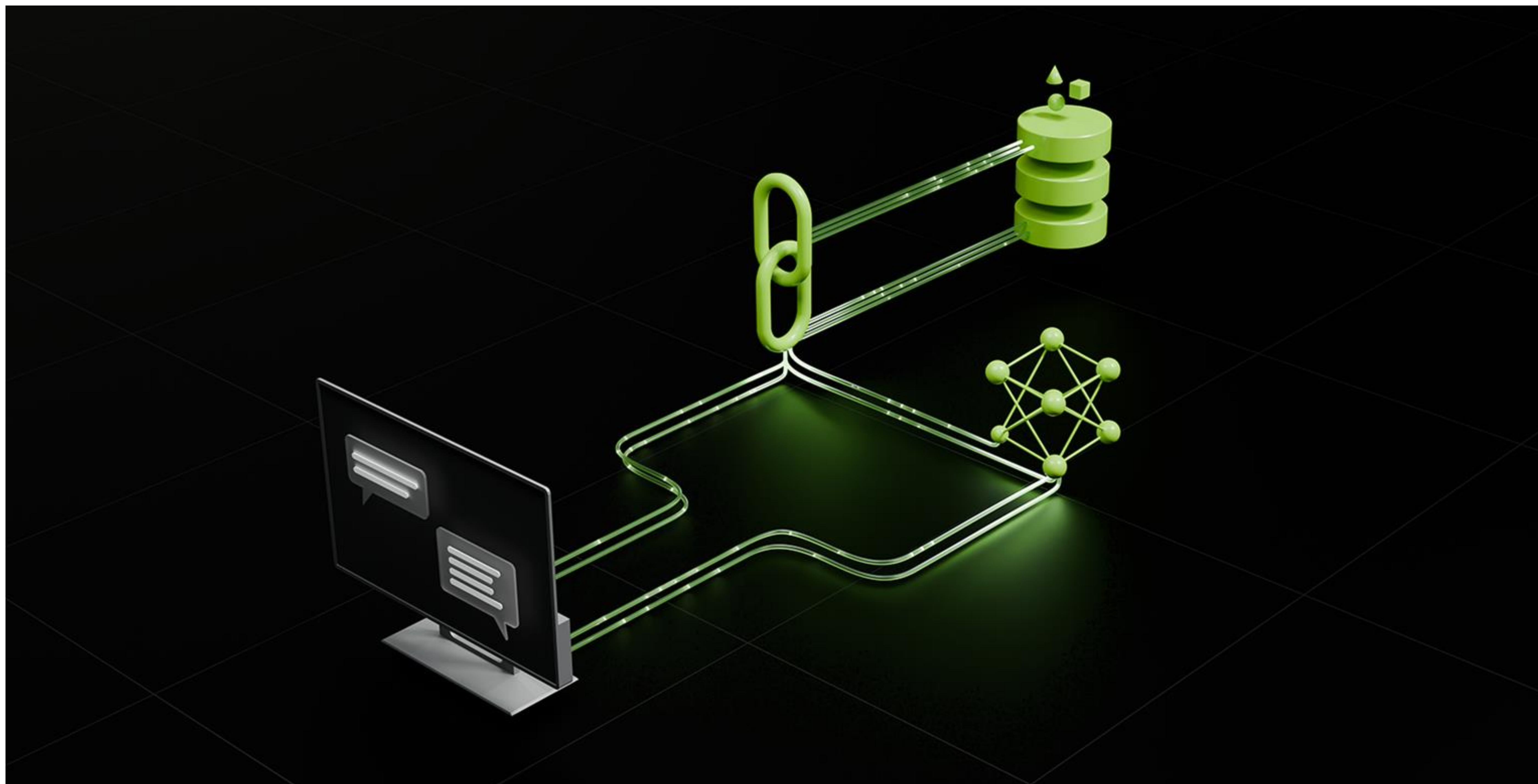
Our Environment

Jupyter Notebooks + Frontend



Dialog/Retrieval Agents

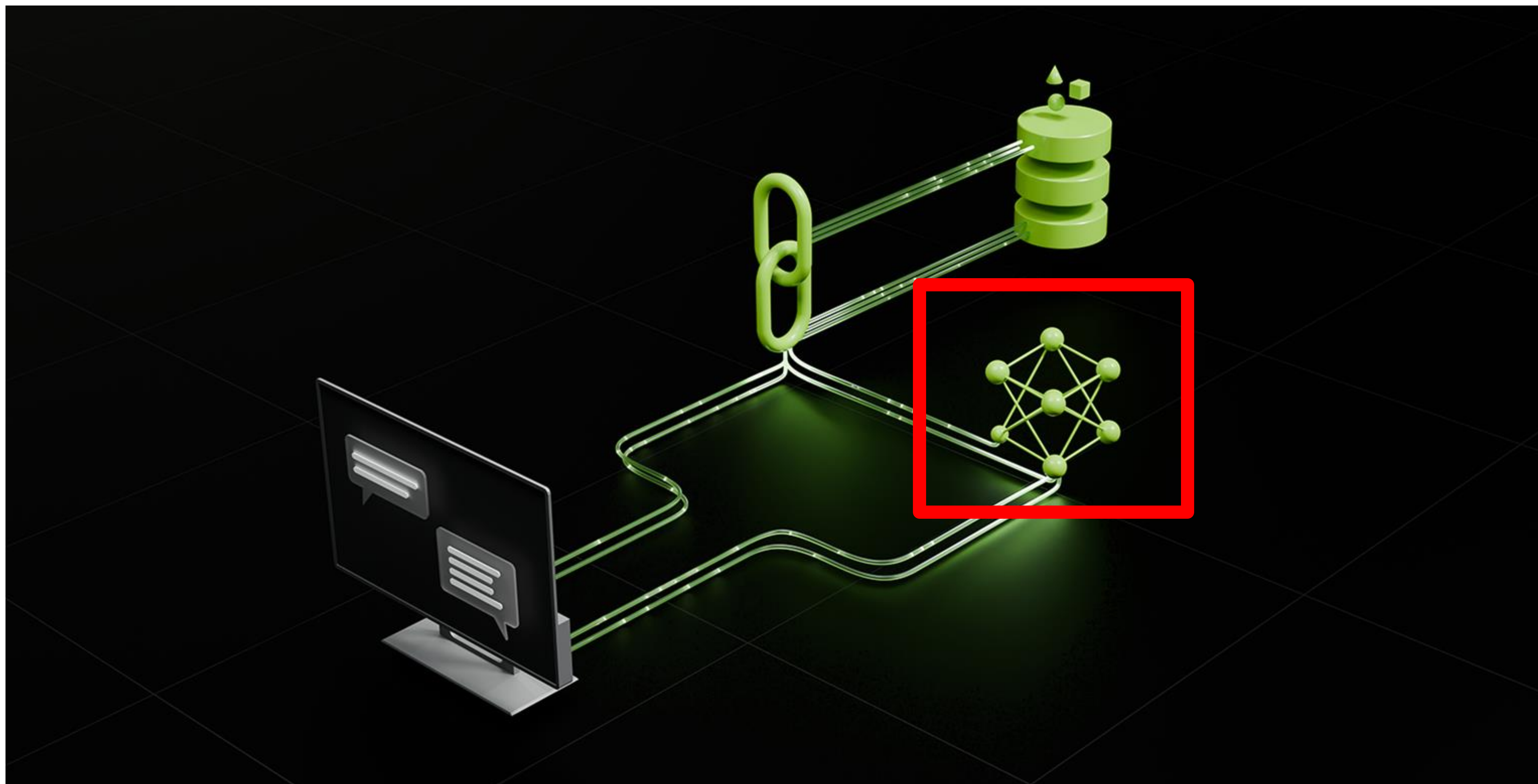
LLMs with Context and Control



Execute with dialog management and information retrieval in production

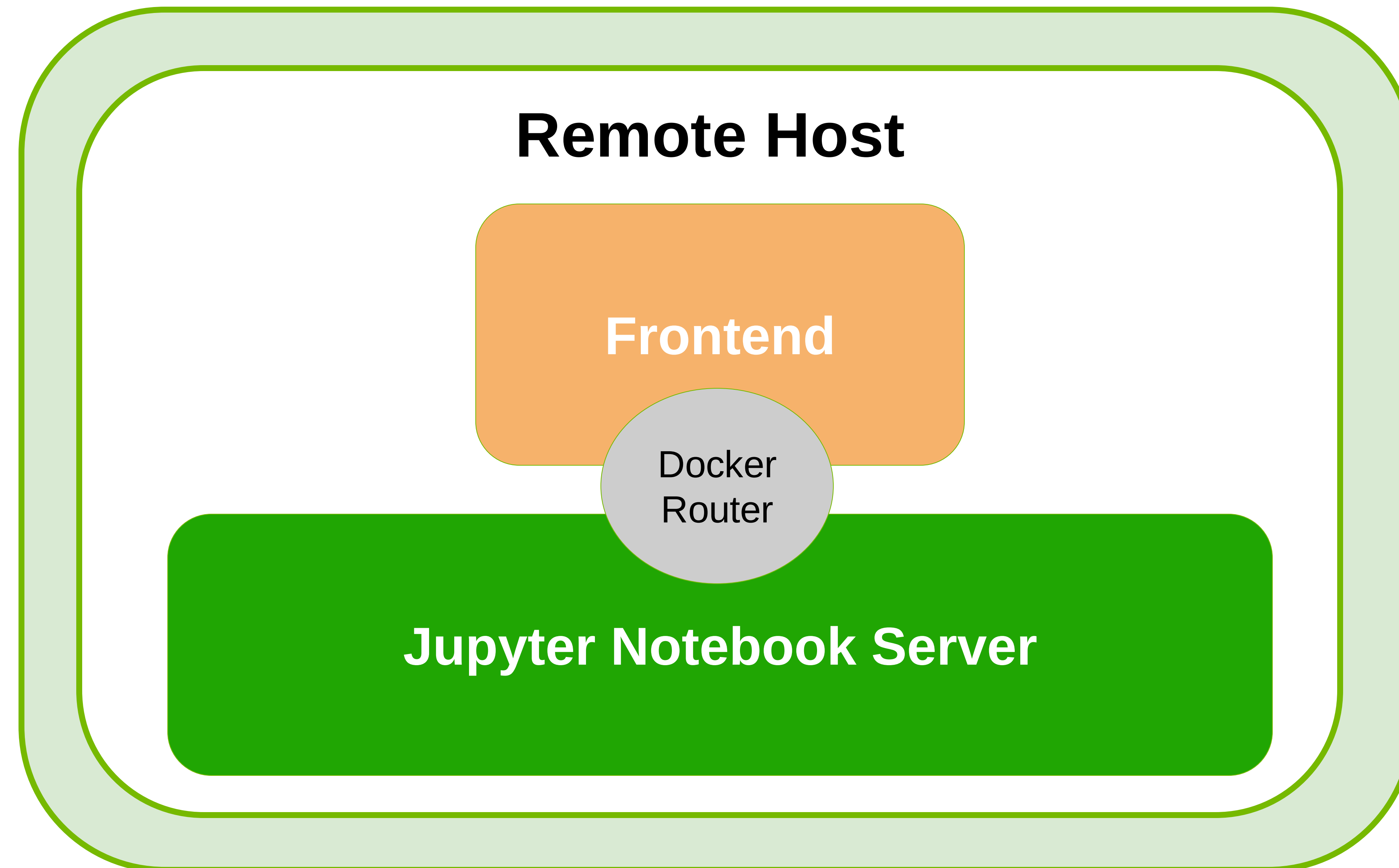
Dialog/Retrieval Agents

LLMs with Context and Control

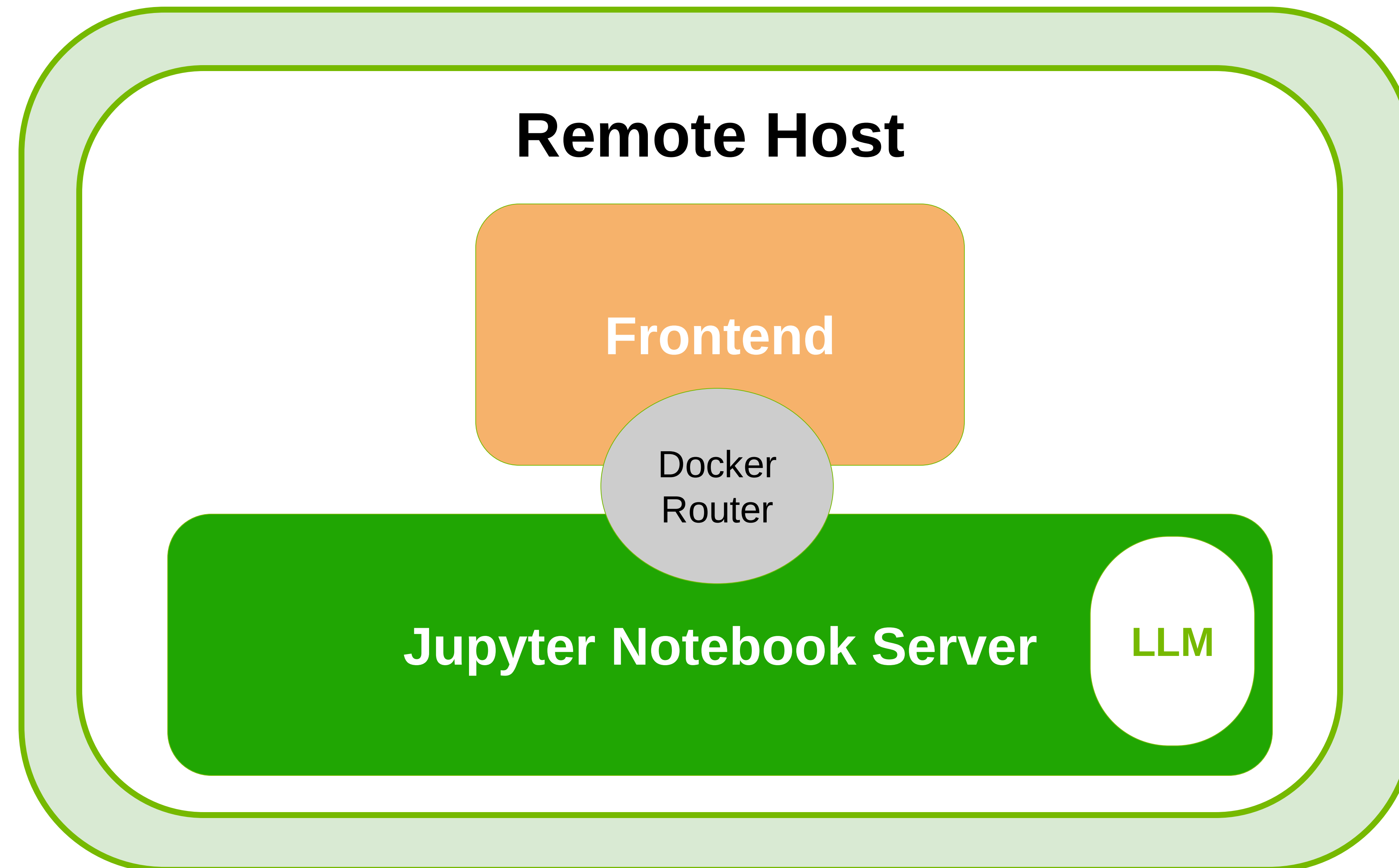


Execute with dialog management and information retrieval in production

Standalone Environment LLM



Standalone Environment LLM



Standalone Environment LLM

Jupyter Notebook

H200

A100

Jupyter Notebook

A10

4070

Jupyter Notebook

VRAM-bound

CPU-only



```
<s>[INST]<<SYS>>
{{system_message}}
<</SYS>>

{{instruction}} [/INST]
{{primer}}
```

```
<s>[INST]<<SYS>>
You are a code generator.
Please provide Python code
per the instruction.
<</SYS>>

Write a Fibonacci method [/INST]
```python
Implementation of Fibonacci w/
```

# Standalone Environment LLM

Jupyter Notebook

H200

A100

Jupyter Notebook

A10

4070

Jupyter Notebook

VRAM-bound

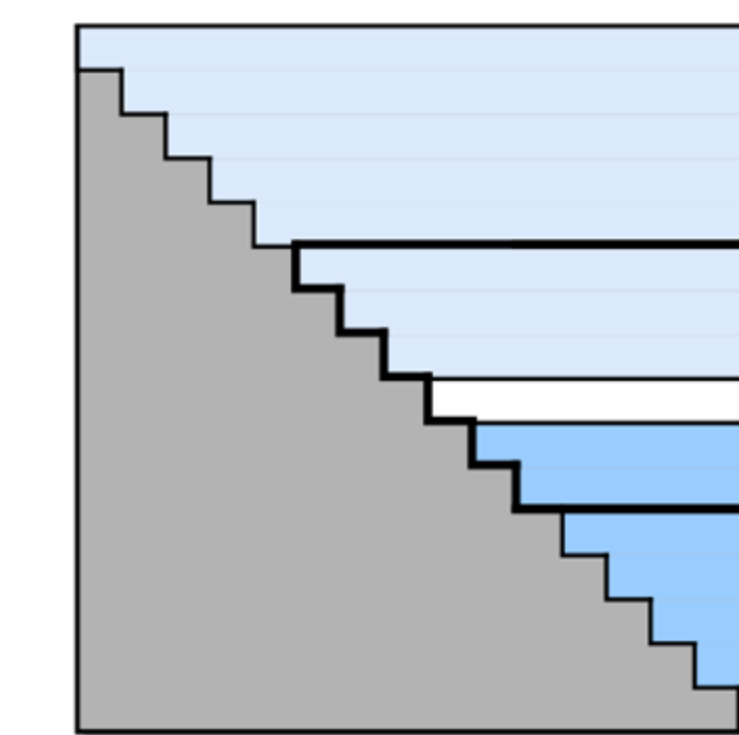
CPU-only



64	134	217
76	119	21
3	81	99

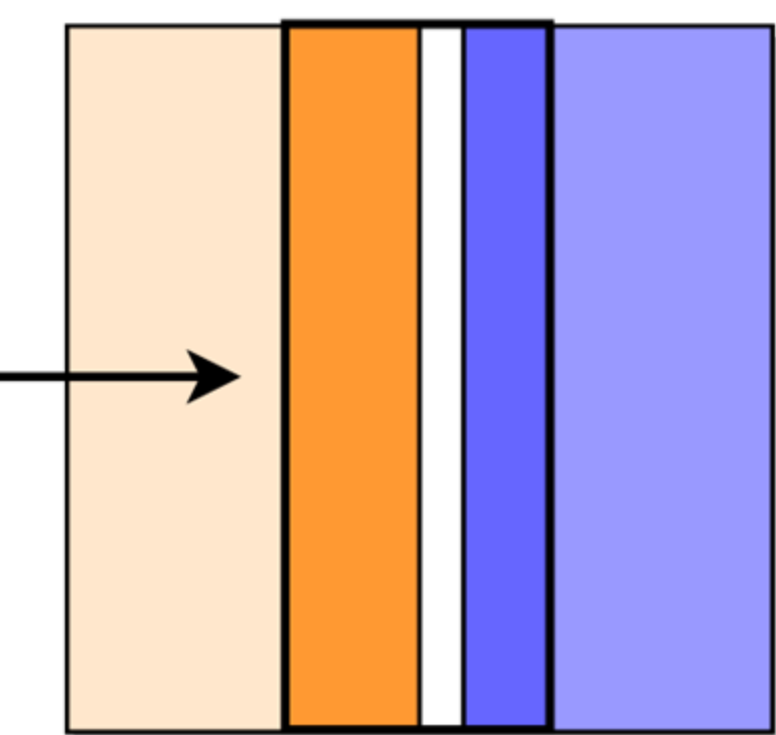
INT8

Inverse Layer Hessian  
(Cholesky Form)



computed initially

Weight Matrix / Block



block  $i$  quantized recursively  
column-by-column



# Standalone Environment LLM

Jupyter Notebook

H200

A100

Jupyter Notebook

A10

4070

Jupyter Notebook

VRAM-bound

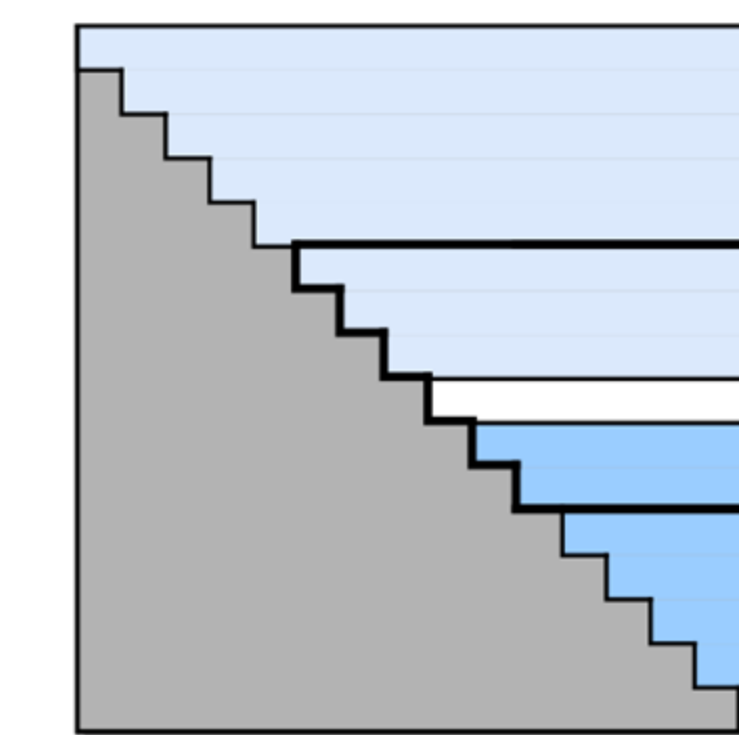
CPU-only



64	134	217
76	119	21
3	81	99

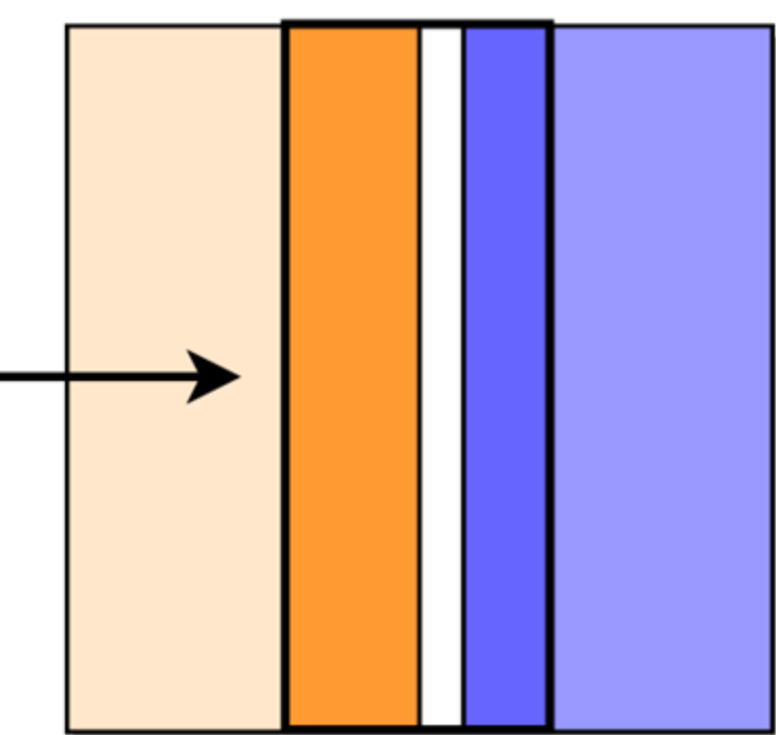
INT8

Inverse Layer Hessian  
(Cholesky Form)



computed initially

Weight Matrix / Block



block  $i$  quantized recursively  
column-by-column



# Remote LLM Access

## Remote Host

Frontend

Jupyter Notebook

VRAM-bound

CPU-only

```
<s>[INST]<<SYS>>
{{system_message}}
<</SYS>>
```

```
{{instruction}} [/INST]
{{primer}}
```

```
<s>[INST]<<SYS>>
You are a code generator.
Please provide Python code
per the instruction.
<</SYS>>
```

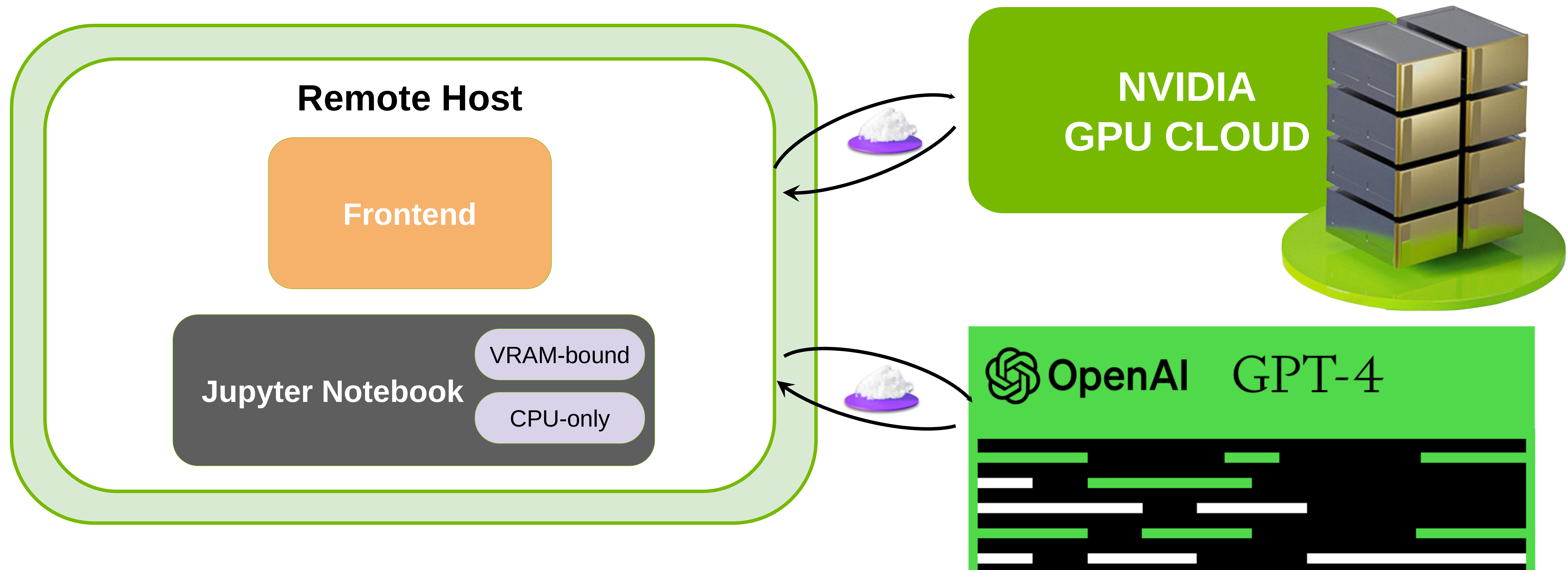
```
Write a Fibonacci method [/INST]
```python
## Implementation of Fibonacci w/
```

Llama

LLM



Large Model Hosting Platforms



Large Model Hosting Platforms



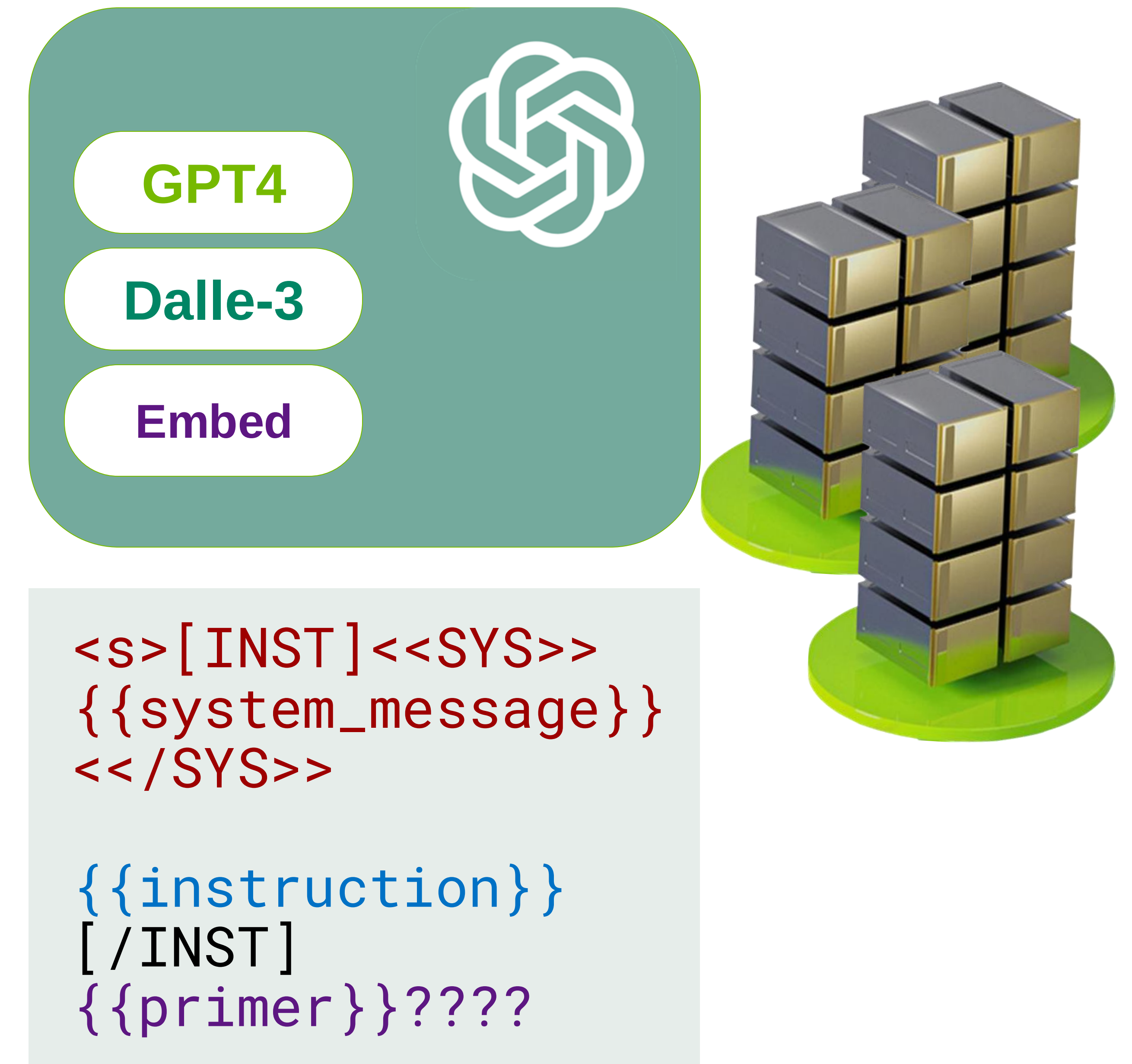
- 1. GPT4/3.5
- 2. Integrated Tools/Multimodal Support
- 3. Models/Query Router Internal
- 4. Path to Local Deployment Less Clear

NVIDIA
GPU CLOUD



- 1. Publicly Available Models
- 2. Models/Query Router Accessible
- 3. Path to Local Deployment/Self-Hosting
- 4. Pieces to Compose Complex Systems

Query Router Access



Query Router Access

```
{  
  "messages": [{  
    "content": "...",  
    "role": "system"  
  }, {  
    "content": "...",  
    "role": "user"  
  }],  
  "model": "gpt-4",  
  "temperature": 0.2,  
  "top_p": 0.7,  
  "max_tokens": 1024,  
  "stream": True  
}
```

/chat/completions

Facilitate
Monitor
Optimize
Load Balance

GPT4

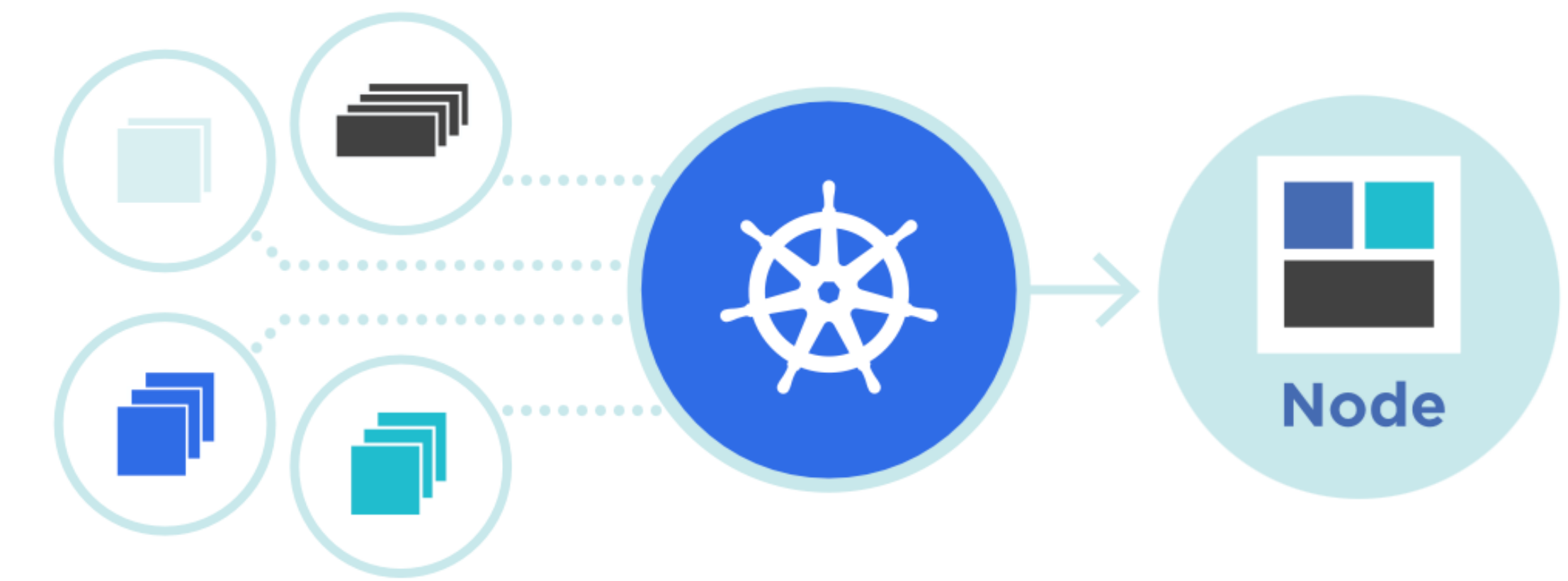
Dalle-3

Embed

```
<s>[INST]<<SYS>>  
{{system_message}}  
<</SYS>>
```

```
{{instruction}}  
[/INST]  
{{primer}}????
```

Query Router Access



```
{  
  "messages": [{  
    "content": "...",  
    "role": "system"  
  }, {  
    "content": "...",  
    "role": "user"  
  }],  
  "model": "gpt-4",  
  "temperature": 0.2,  
  "top_p": 0.7,  
  "max_tokens": 1024,  
  "stream": True  
}
```

/chat/completions

Facilitate
Monitor
Optimize
Load Balance

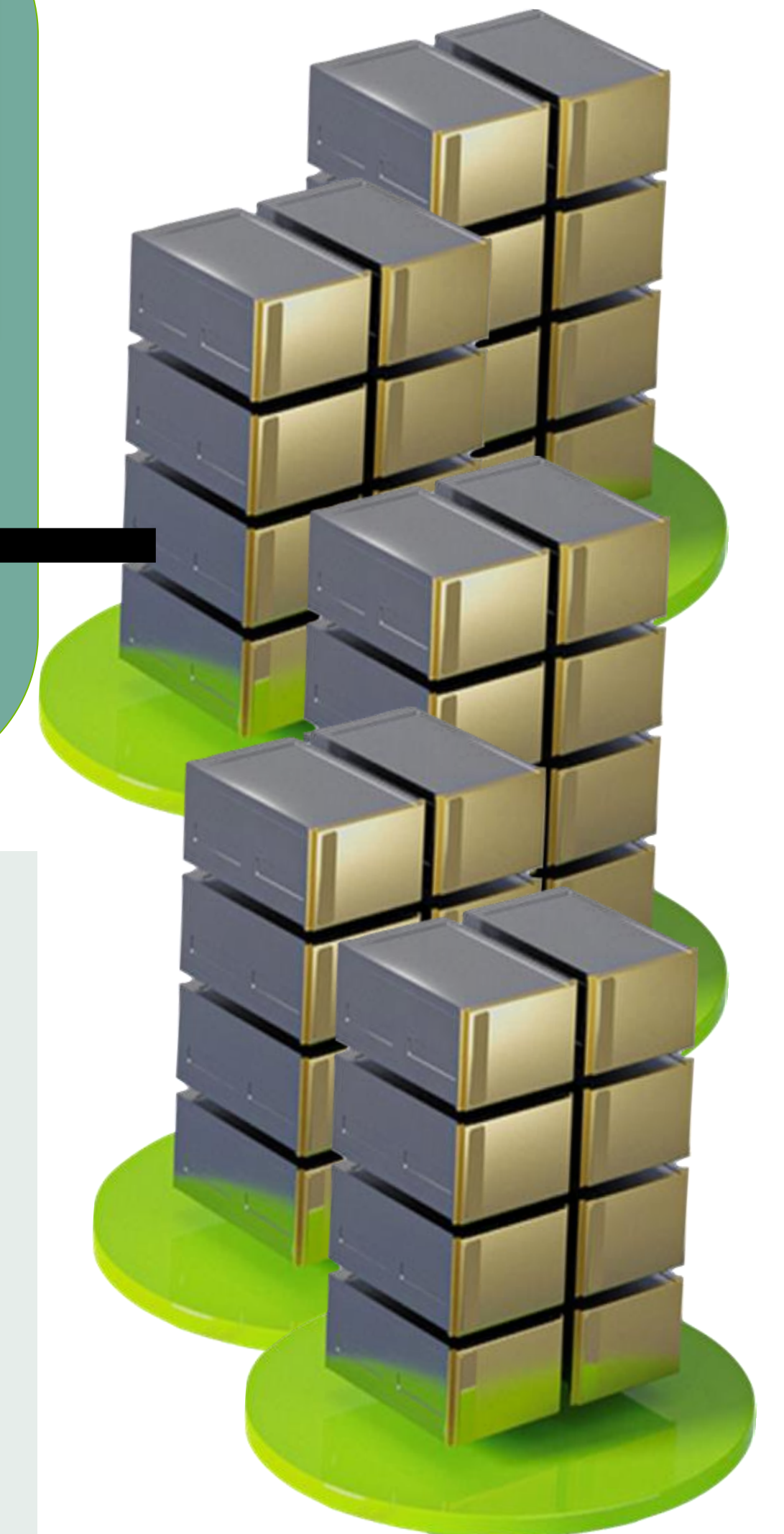
GPT4

Dalle-3

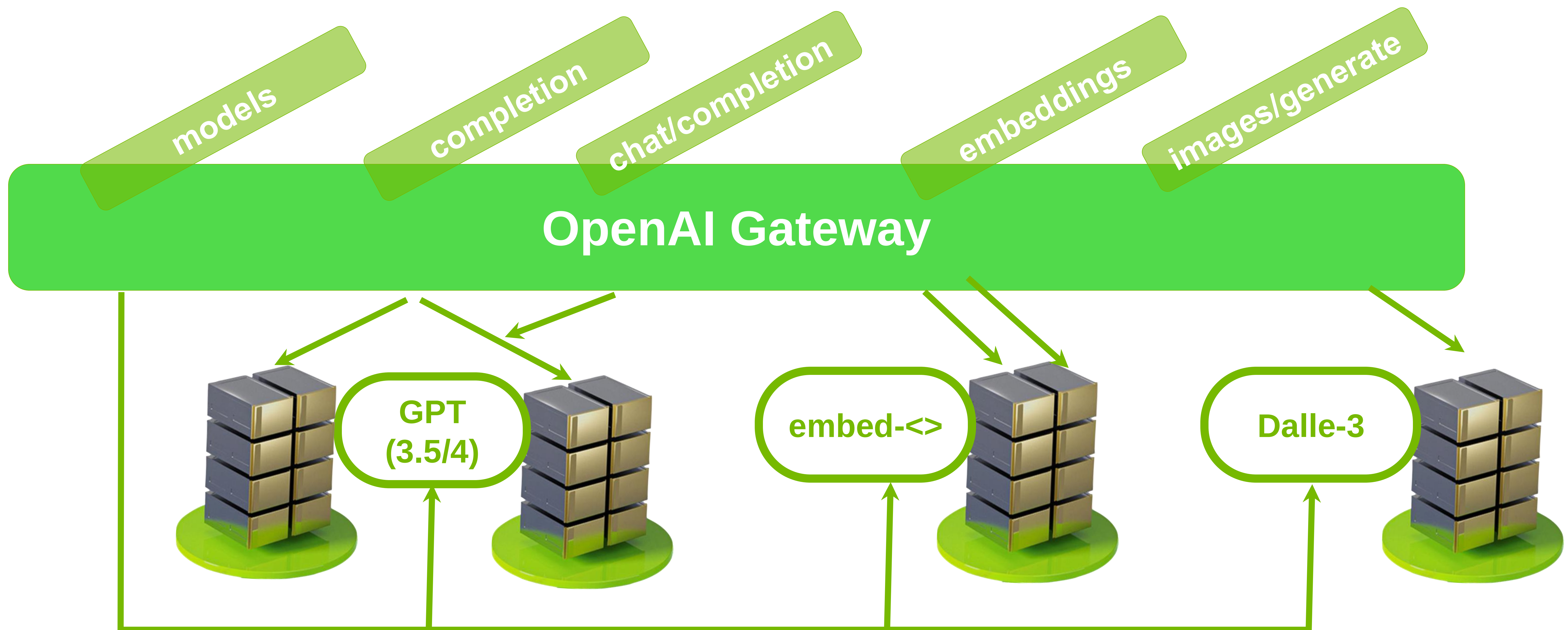
Embed

```
<s>[INST]<<SYS>>  
{{system_message}}  
<</SYS>>
```

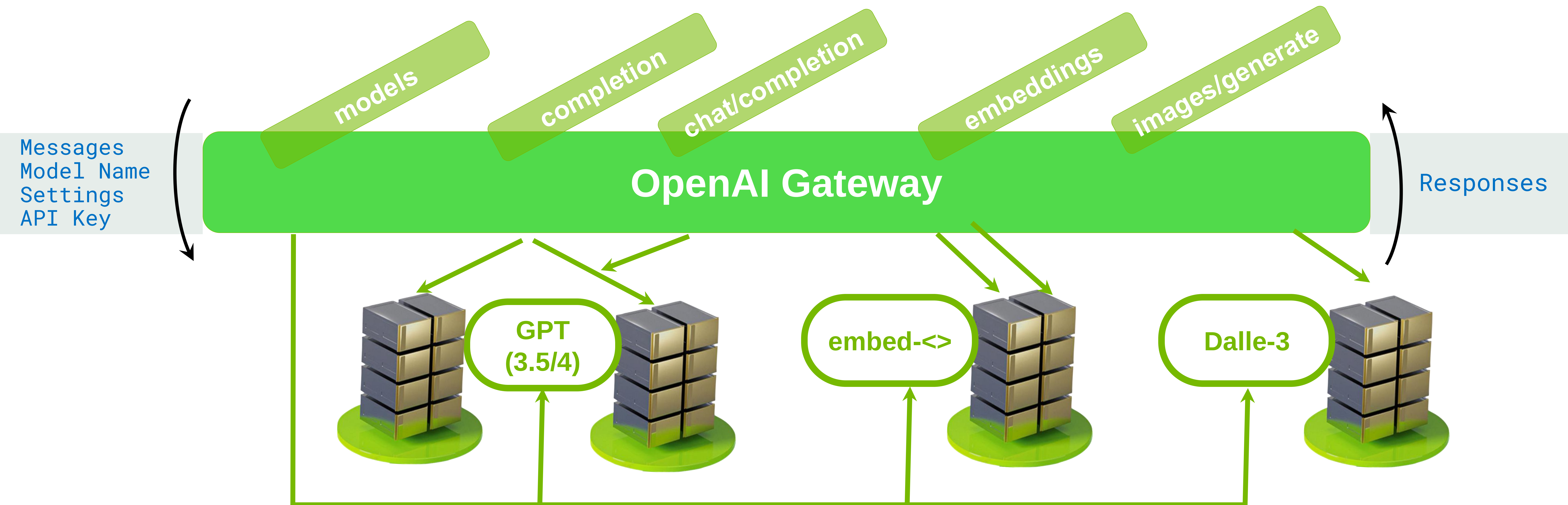
```
{{instruction}}  
[/INST]  
{{primer}}????
```



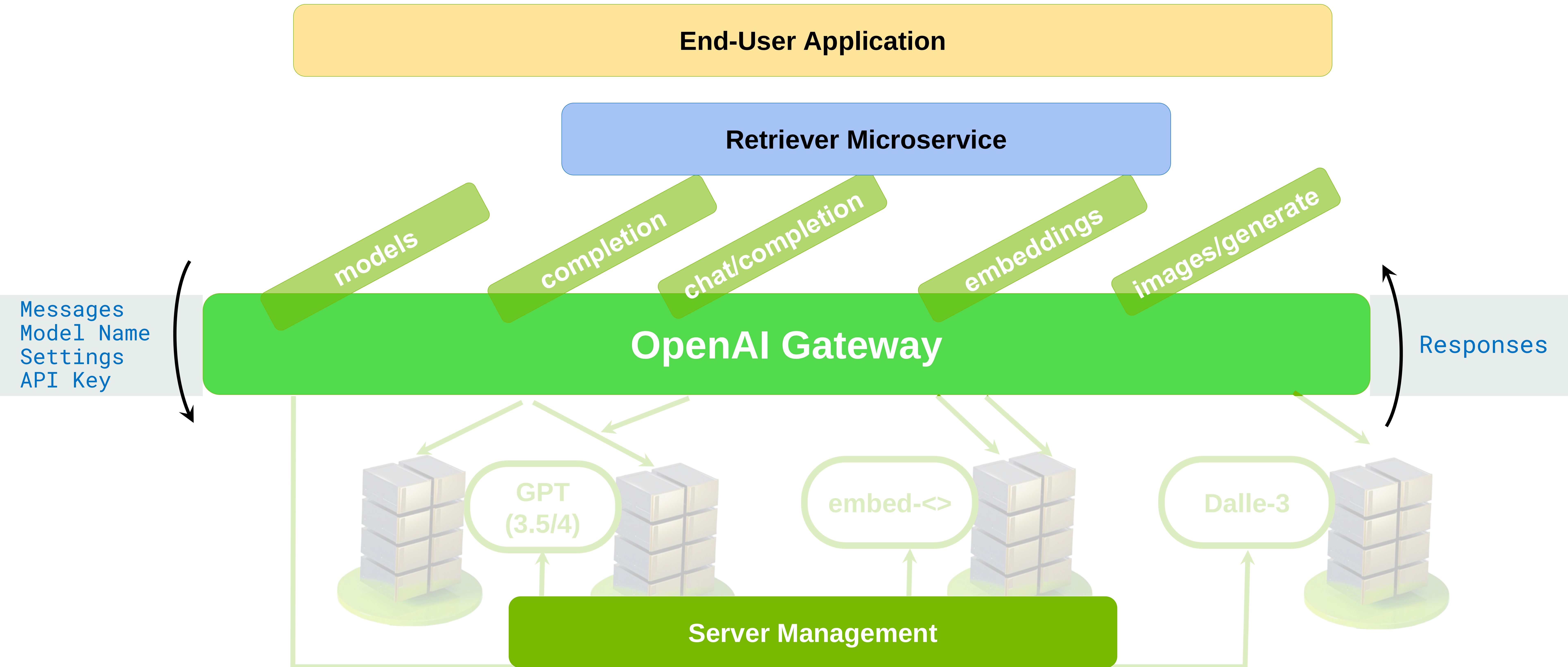
Large Model Hosting Platforms - OpenAI



Large Model Hosting Platforms - OpenAI



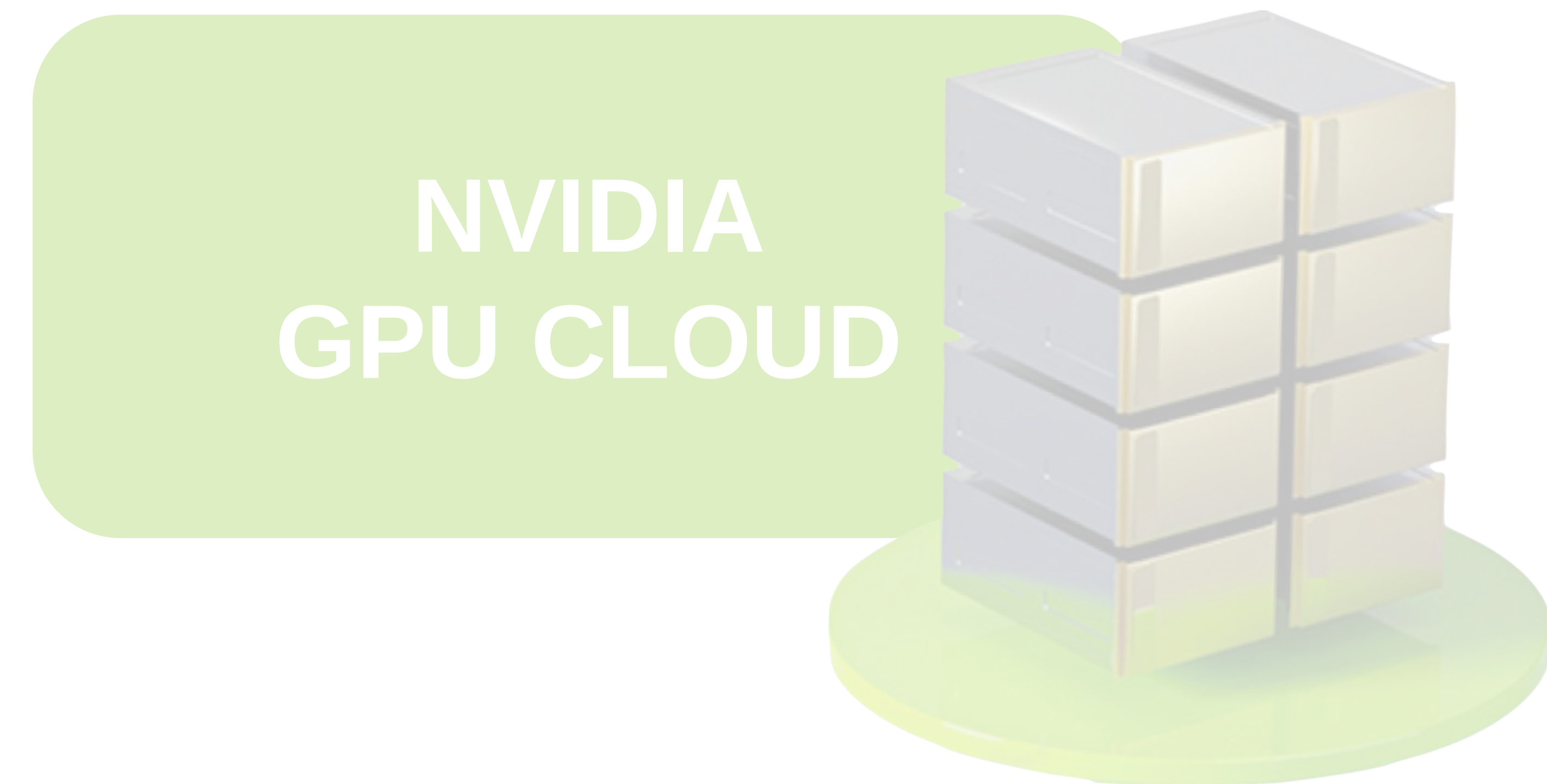
Large Model Hosting Platforms - OpenAI



Large Model Hosting Platforms



- 1. GPT4/3.5
- 2. Integrated Tools/Multimodal Support
- 3. Models/Query Router Internal
- 4. Path to Local Deployment Less Clear

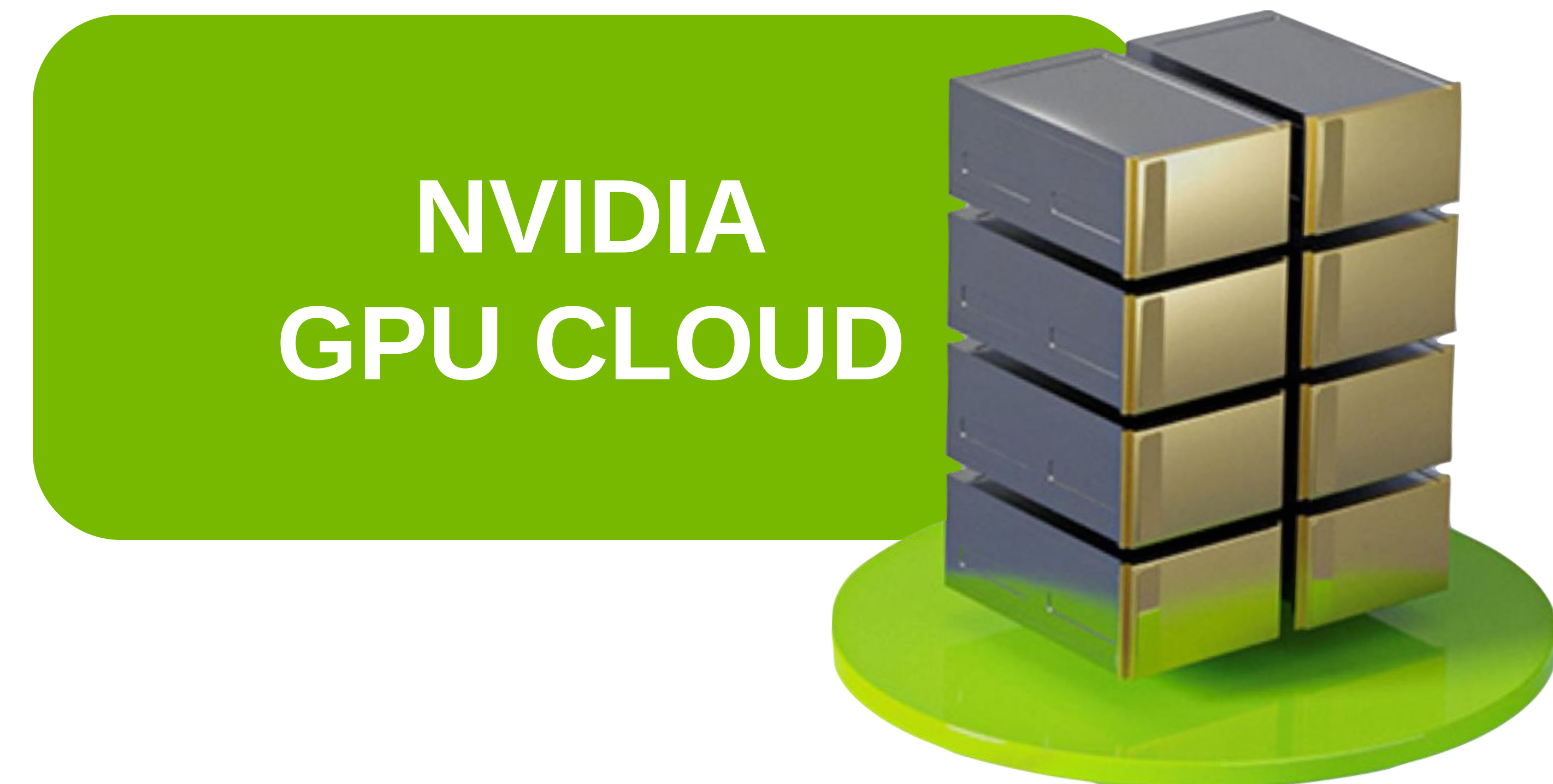


- 1. Publicly Available Models
- 2. Models/Query Router Accessible
- 3. Path to Local Deployment/Self-Hosting
- 4. Pieces to Compose Complex Systems

Large Model Hosting Platforms



- 1. GPT4/3.5
- 2. Integrated Tools/Multimodal Support
- 3. Models/Query Router Internal
- 4. Path to Local Deployment Less Clear



- 1. Publicly Available Models
- 2. Models/Query Router Accessible
- 3. Path to Local Deployment/Self-Hosting
- 4. Pieces to Compose Complex Systems

Query Router Access

```
{
  "messages": [{
    "content": "...",
    "role": "system"
  }, {
    "content": "...",
    "role": "user"
  }],
  "model": "mixtral",
  "temperature": 0.2,
  "top_p": 0.7,
  "max_tokens": 1024,
  "stream": True
}
```

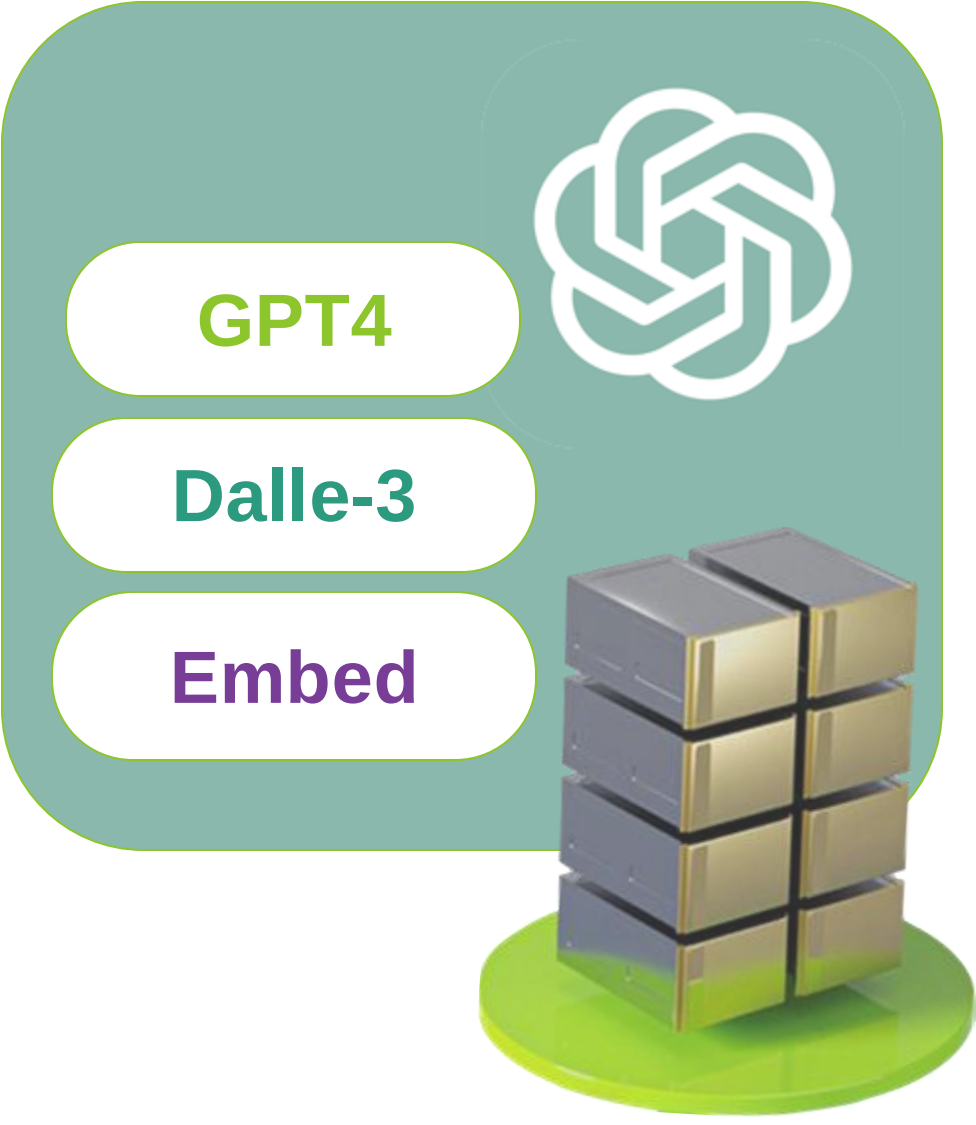
/chat/completions

Facilitate
Monitor
Optimize
Load Balance

<s>[INST]<<SYS>>
{{system_message}}
<</SYS>>

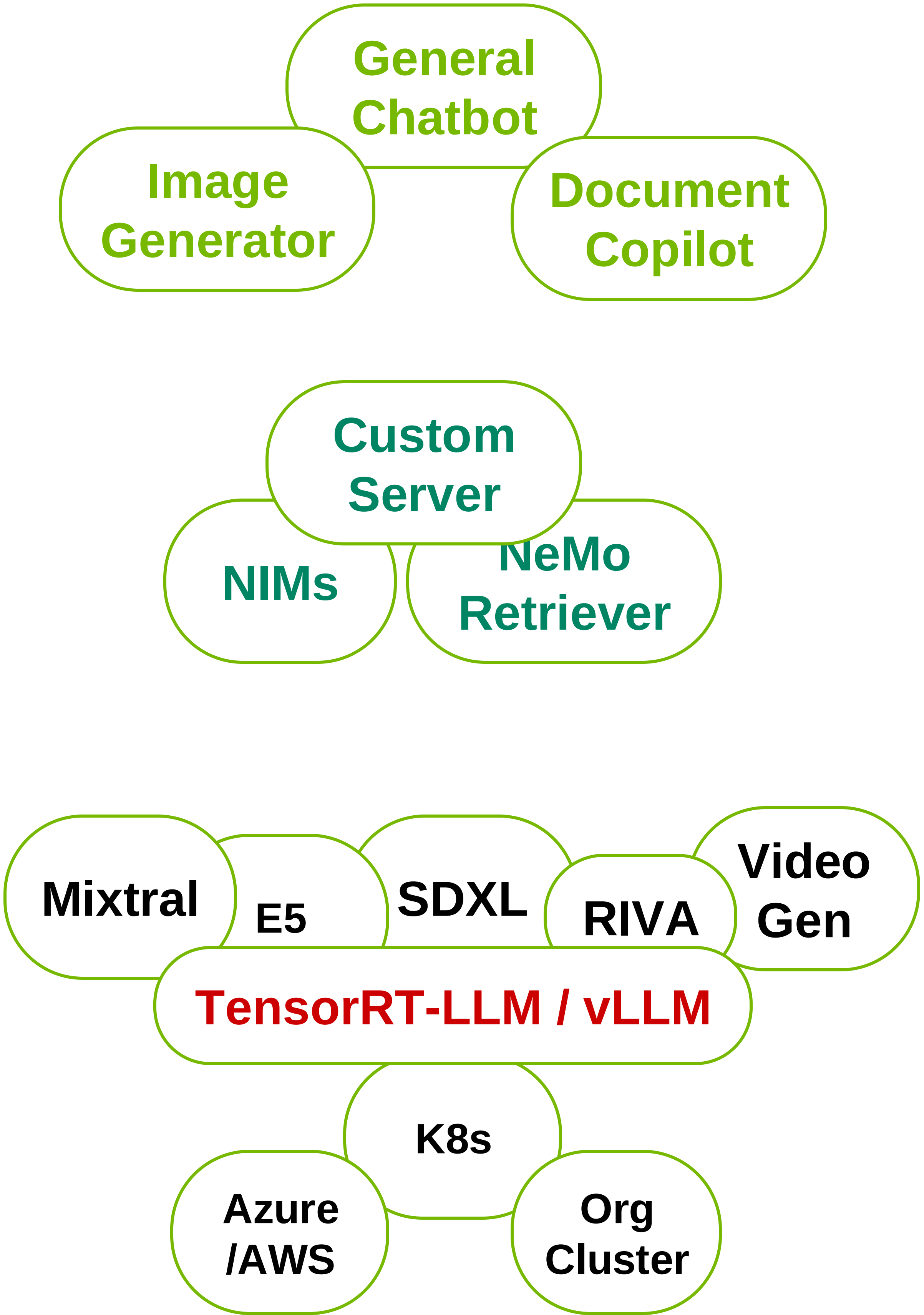
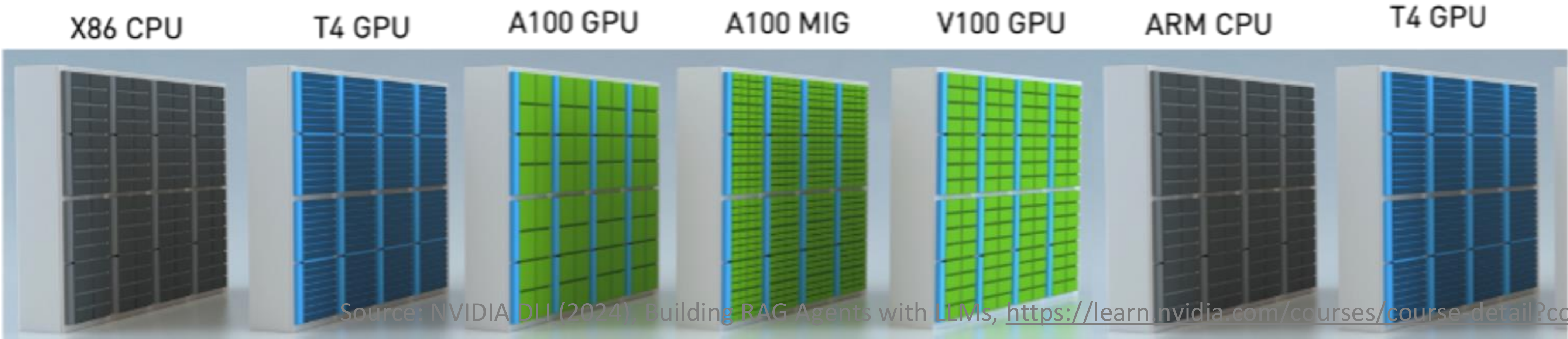
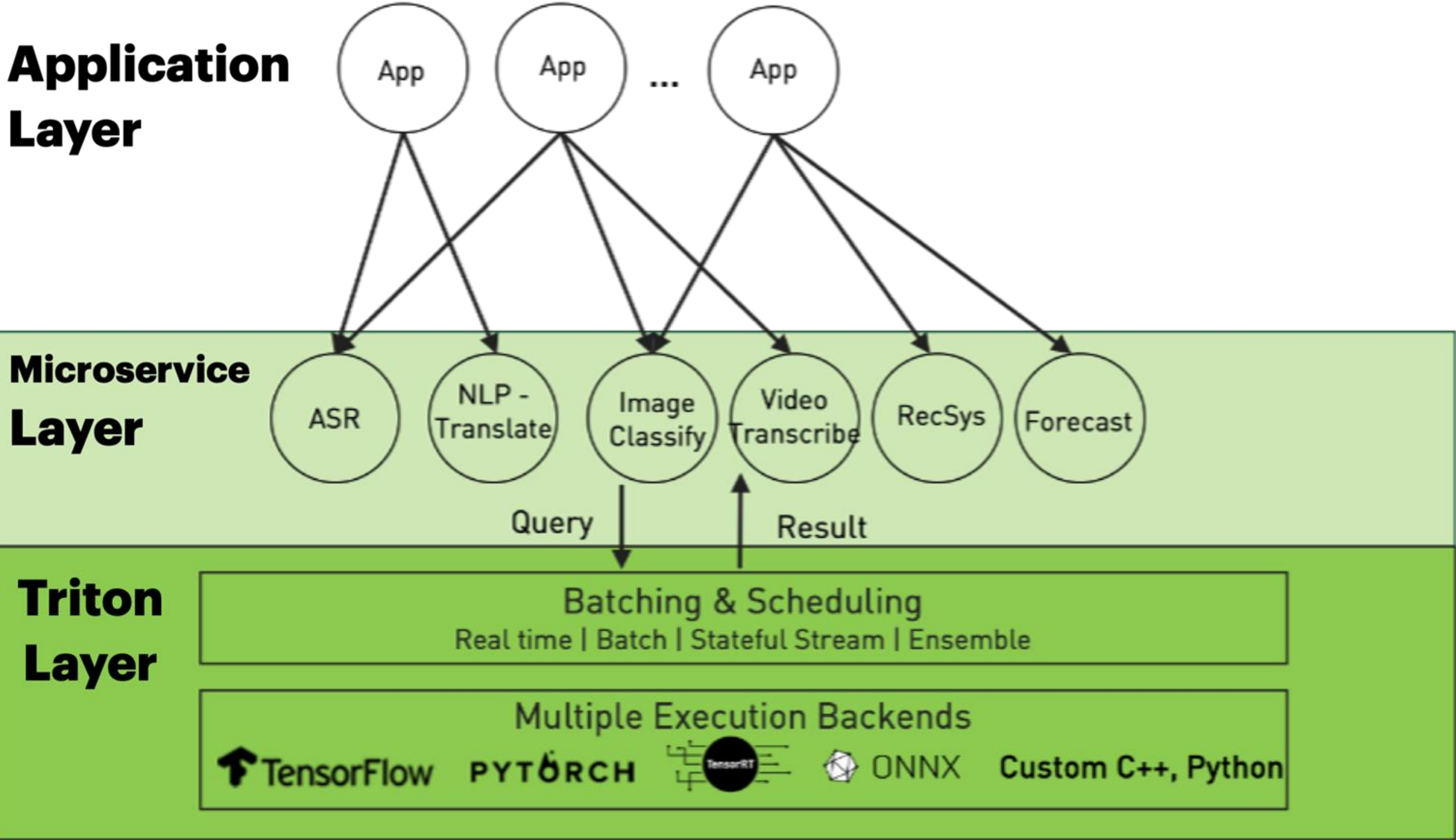
{{instruction}} [/INST]
{{primer}}

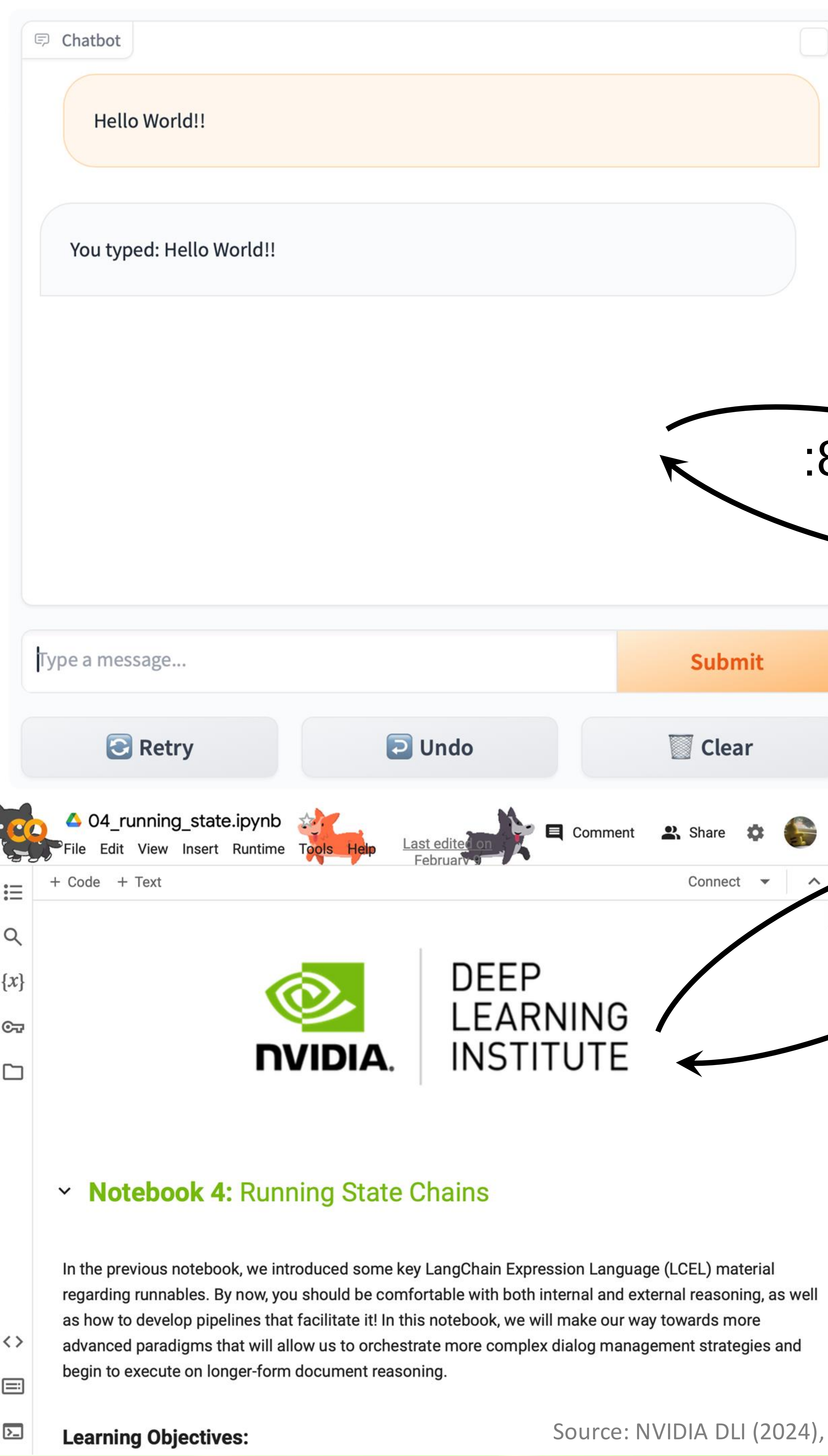
{
 "context": {{context}}
 "model": "query"/"doc"
}



Full Deployment Stack

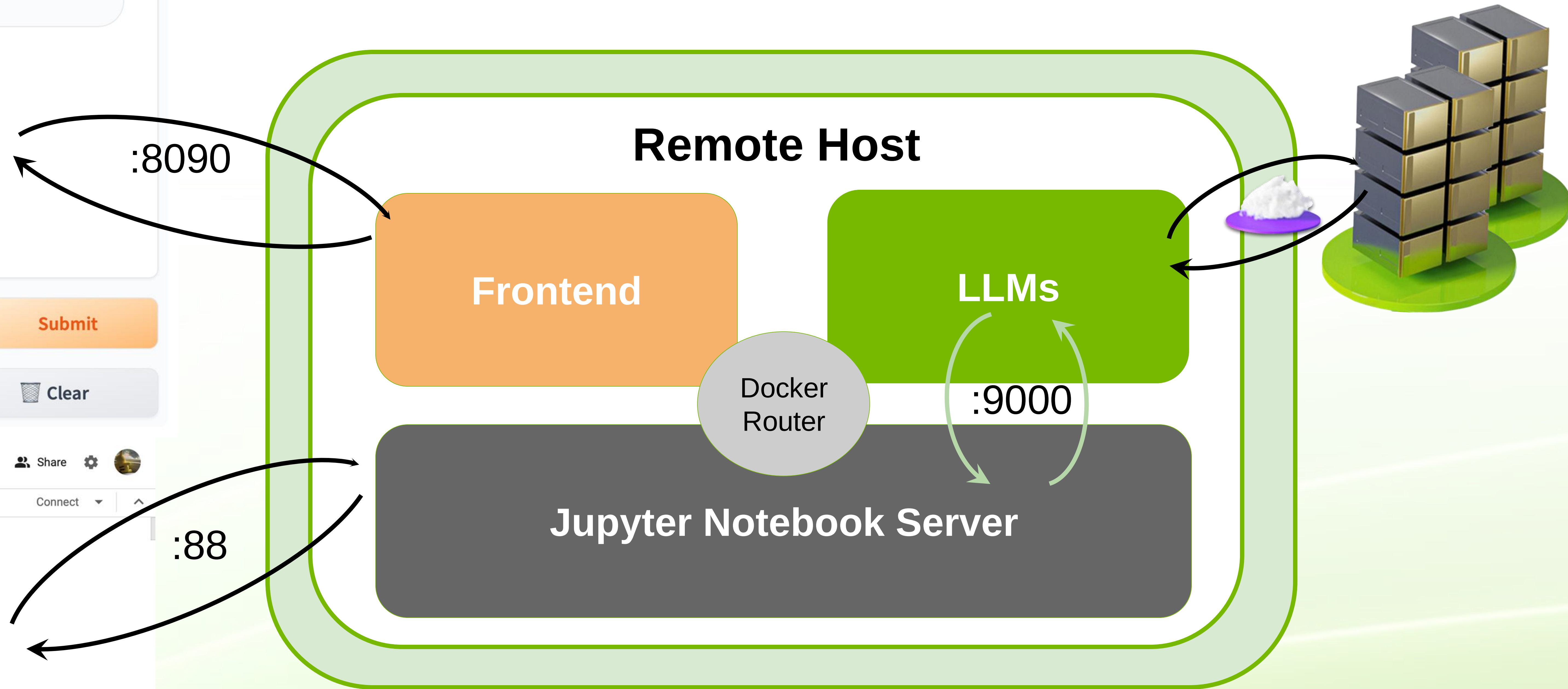
Millions of Apps | Billions of Users | Trillions of Queries



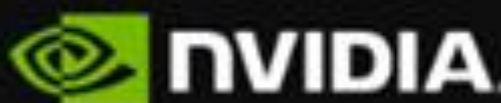


Our Environment

Jupyter Notebooks + Frontend
+ LLM Client



NVIDIA Foundation Model Endpoints



[Explore](#)

[Docs](#)

[Login](#)

Discover

MODELS

Reasoning

Visual Design

Retrieval

Speech

Biology

INDUSTRIES

Gaming

Healthcare

Industrial

Top Open Foundation Models

The leading open models built by the community, optimized and accelerated by NVIDIA's enterprise-ready inference runtime



stabilityai

sdxl-turbo

image generation

nvidia nim



google

gemma-7b

chat

language generation



meta

llama2-70b

chat

language generation

Trending Now

The latest and most popular additions to the list



mistralai /

mixtral-8x7b-instruct-v0.1

An MOE LLM that follows instructions, completes requests, and generates creative text.



microsoft /

kosmos-2

Groundbreaking multimodal model designed to understand and reason about visual elements in images.



NVIDIA

GPU CLOUD

Source: NVIDIA DLI (2024). Building RAG Agents with LLMs, https://learn.nvidia.com/courses/course-detail?course_id=course-v1-DLI+S-FX-15+V1

107



NVIDIA Foundation Model Endpoints



Search NVIDIA AI

< Back

mistralai / **mixtral-8x7b-instruct-v0.1**

An MOE LLM that follows instructions, completes requests, and generates creative text.

advanced reasoning

chat

large language models

text-to-text

A 3D icon of a stack of four GPU modules, each with a gold top and a dark grey base, arranged in a 2x2 grid. The stack is placed on a green circular base.

**NVIDIA
GPU CLOUD**

NVIDIA Foundation Model Endpoints

mistralai / **mixtral-8x7b-instruct-v0.1**

An MOE LLM that follows instructions, completes requests, and generates creative text.

advanced reasoning

chat

large language models

text-to-text

Preview

JSON

Reset Chat



NVIDIA's GPU Technology Conference (GTC) is an annual event that showcases the latest developments in graphics processing units (GPUs) and artificial intelligence (AI). Here are some of the things you can expect to see at GTC:

1. Keynote speeches: GTC typically features keynote speeches from NVIDIA's CEO, Jensen Huang, as well as other industry leaders and experts. These speeches cover the latest trends and breakthroughs in AI, data science, high-performance computing, and graphics.
2. Technical sessions: GTC offers a wide range of technical sessions, including

Type text here...

Python

Node

Shell

Get API Key Copy Code

```
from openai import OpenAI

client = OpenAI(
    base_url = "https://integrate.api.nvidia.com/v1",
    api_key = "$API_KEY_REQUIRED_IF_EXECUTING_OUTSIDE_NGC"
)

completion = client.chat.completions.create(
    model="mistralai/mixtral-8x7b-instruct-v0.1",
    messages=[{"role": "user", "content": ""}],
    temperature=0.5,
    top_p=1,
    max_tokens=1024,
    stream=True
)

for chunk in completion:
    if chunk.choices[0].delta.content is not None:
        print(chunk.choices[0].delta.content, end="")
```



**NVIDIA
GPU CLOUD**

NVIDIA Foundation Model Endpoints

mistralai / **mixtral-8x7b-instruct-v0.1**

An MOE LLM that follows instructions, completes requests, and generates creative text.

advanced reasoning

chat

large language models

text-to-text

Python

Node

Shell

Get API Key Copy Code

```
curl https://integrate.api.nvidia.com/v1/chat/completions \
-H "Content-Type: application/json" \
-H "Authorization: Bearer $API_KEY_REQUIRED_IF_EXECUTING_OUTSIDE_NGC" \
-d '{
  "model": "mistralai/mixtral-8x7b-instruct-v0.1",
  "messages": [{"role": "user", "content": ""}],
  "temperature": 0.5,
  "top_p": 1,
  "max_tokens": 1024,
  "stream": true
}'
```

```
from openai import OpenAI

client = OpenAI(
    base_url = "https://integrate.api.nvidia.com/v1",
    api_key = "$API_KEY_REQUIRED_IF_EXECUTING_OUTSIDE_NGC"
)

completion = client.chat.completions.create(
    model="mistralai/mixtral-8x7b-instruct-v0.1",
    messages=[{"role": "user", "content": ""}],
    temperature=0.5,
    top_p=1,
    max_tokens=1024,
    stream=True
)
```

```
for chunk in completion:
    if chunk.choices[0].delta.content is not None:
        print(chunk.choices[0].delta.content, end="")
```



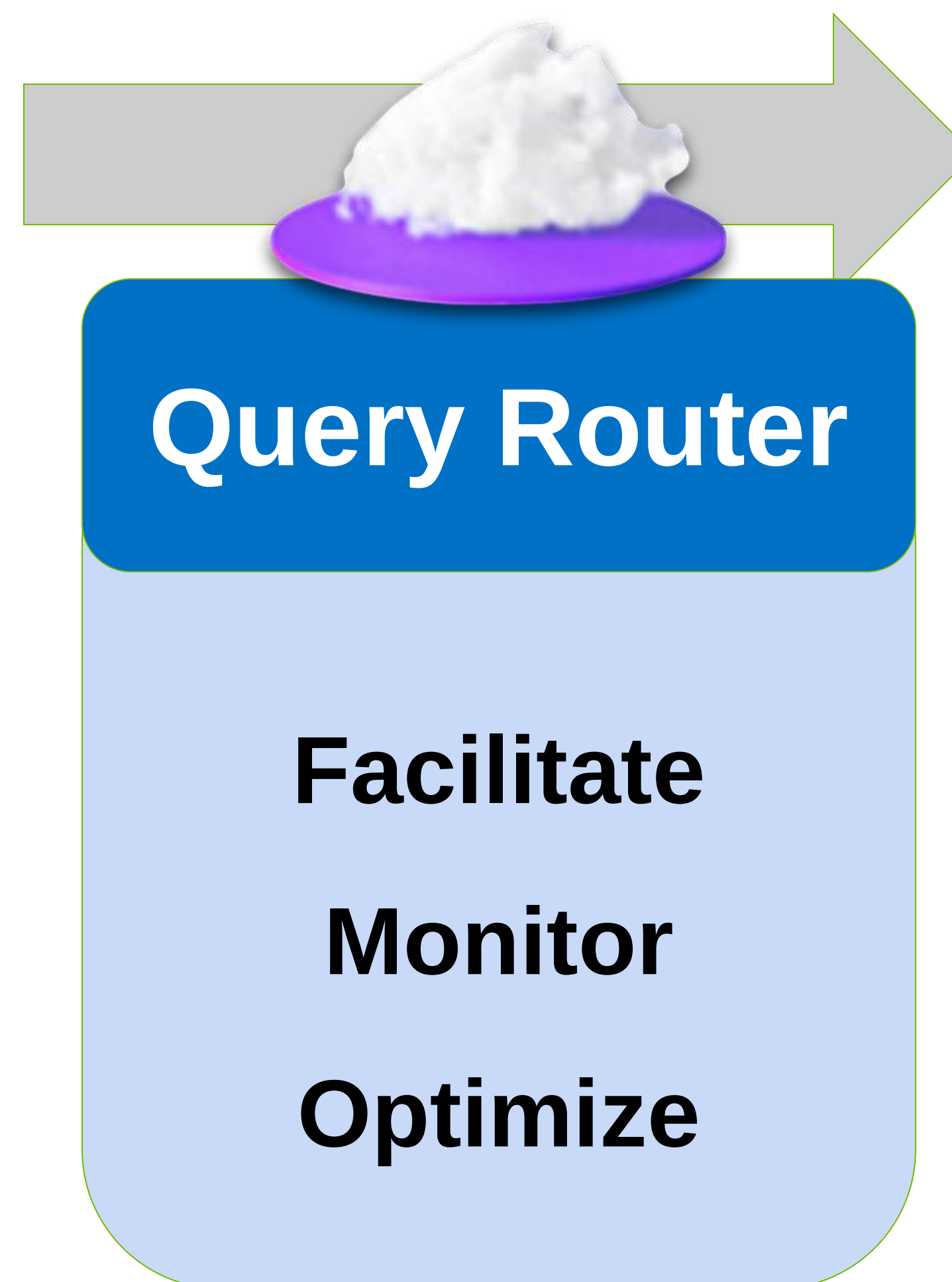
**NVIDIA
GPU CLOUD**

From Raw Requests to LangChain Model



From Raw Requests to LangChain Model

```
> llm("Hello world")  
  ChatMessage(content="hello")  
  
> embedder("Hello")  
  [0.4535437, 0.0800435, ...]
```



```
<s>[INST]<<SYS>>  
{{system_message}}  
<</SYS>>
```

```
{{instruction}} [/INST]  
{{primer}}
```

```
{  
  "context": {{context}}  
  "model": "query"/"doc"  
}
```

Llama Mistral

LLM

E5



LLM Interfaces

The Whole Stack

api.nvcf.
nvidia.com
/v2/nvcf

 NVIDIA

NGC | CLOUD FUNCTIONS

 CATALOG

 CONSOLE

 PRIVATE REGISTRY

 ORGANIZATION

 CLOUD FUNCTIONS

Functions

Create Function

Cloud Functions > Functions

Functions

My Functions

Shared Functions

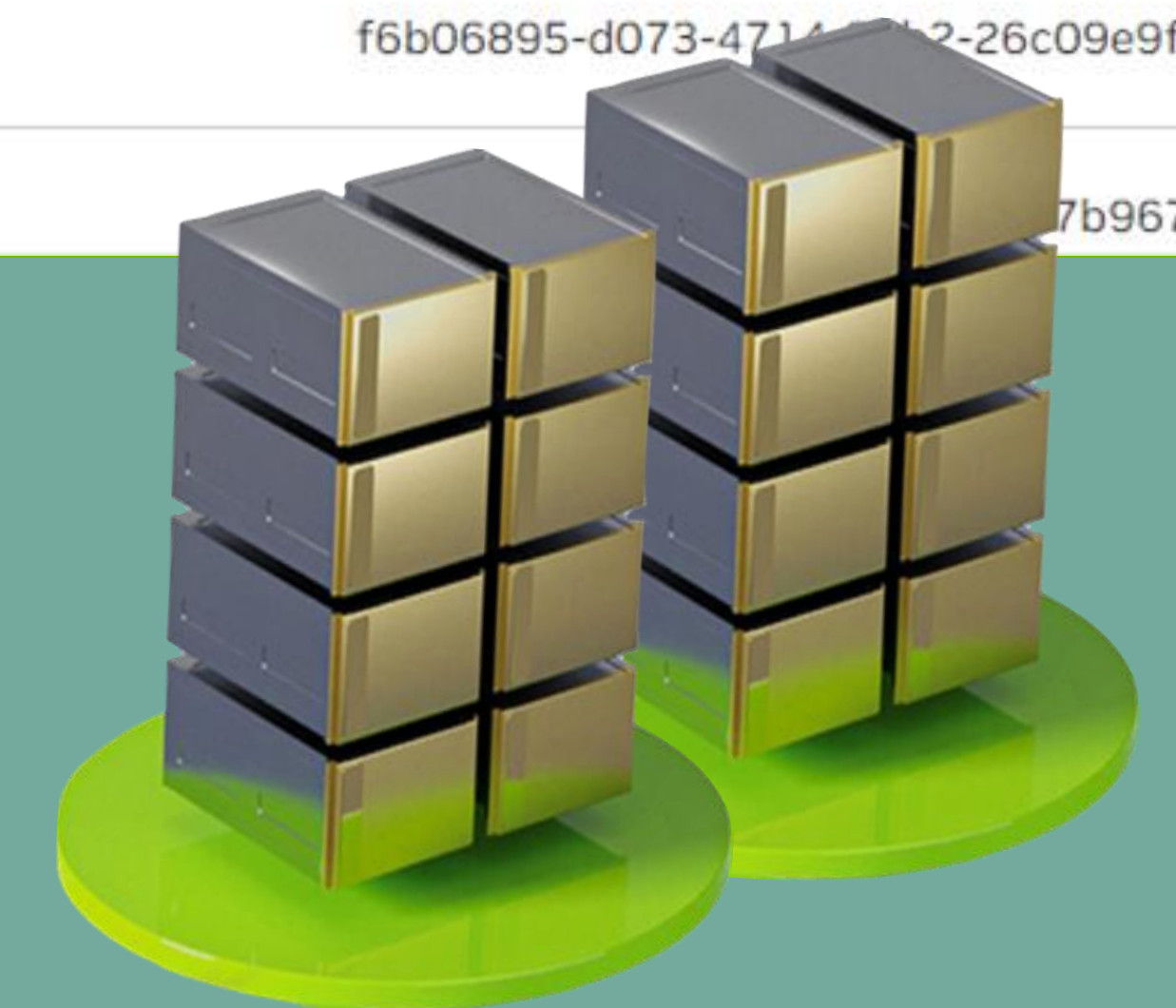
Created by NVIDIA

Function Name	Function ID
ai-llama2-70b 	2fddadfb-7e76-4c8a-9b82-f7d3fab94471
ai-mixtral-8x7b-instruct 	a1e53ece-bff4-44d1-8b13-c009e5bf47f6
ai-mistral-7b-instruct-v2 	d7618e99-db93-4465-af4d-330213a7f51f
ai-codellama-70b 	f6b06895-d073-4714-8112-26c09e9f6597
ai-gemma-7b 	7b9675f9





Scaled
Function
Deployment
Solution



LLM Interfaces

The Whole Stack

health.api.nvidia.com/
ai.api.nvidia.com/

integrate.api.nvidia.com

v1/models
v1/completions
v1/chat/completions
v1/embeddings

api.openai.com

v1/models
v1/completions
v1/chat/completions
v1/embeddings

api.nvcf.
nvidia.com
/v2/nvcf

NVIDIA. NGC | CLOUD FUNCTIONS

CATALOG
CONSOLE
PRIVATE REGISTRY
ORGANIZATION
CLOUD FUNCTIONS
Functions
Create Function

Cloud Functions > Functions

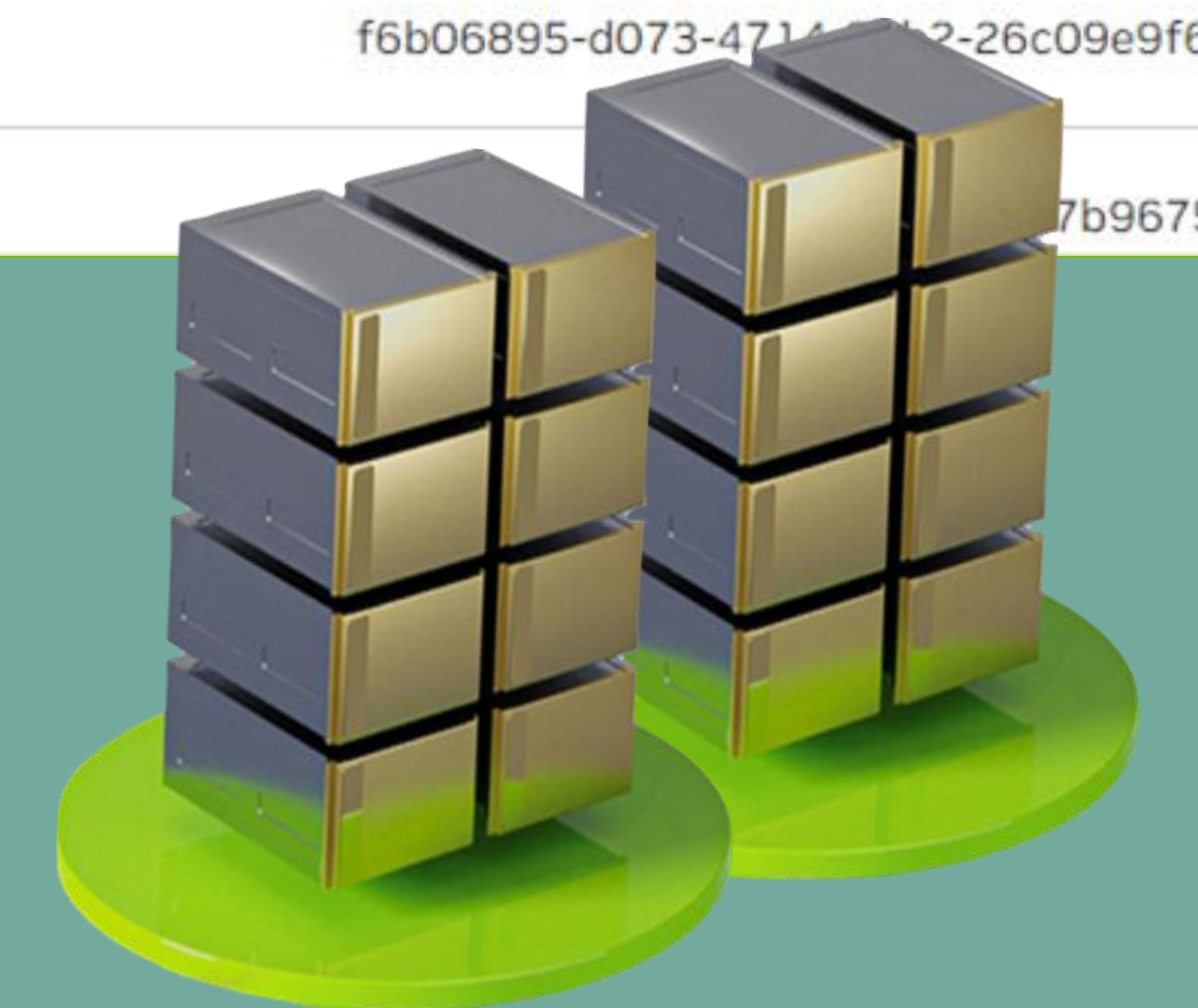
Functions

My Functions Shared Functions Created by NVIDIA

Function Name	Function ID
ai-llama2-70b	2fddadfb-7e76-4c8a-9b82-f7d3fab94471
ai-mixtral-8x7b-instruct	a1e53ece-bff4-44d1-8b13-c009e5bf47f6
ai-mistral-7b-instruct-v2	d7618e99-db93-4465-af4d-330213a7f51f
ai-codellama-70b	f6b06895-d073-4714-b012-26c09e9f6597
ai-gemma-7b	7b9675f9

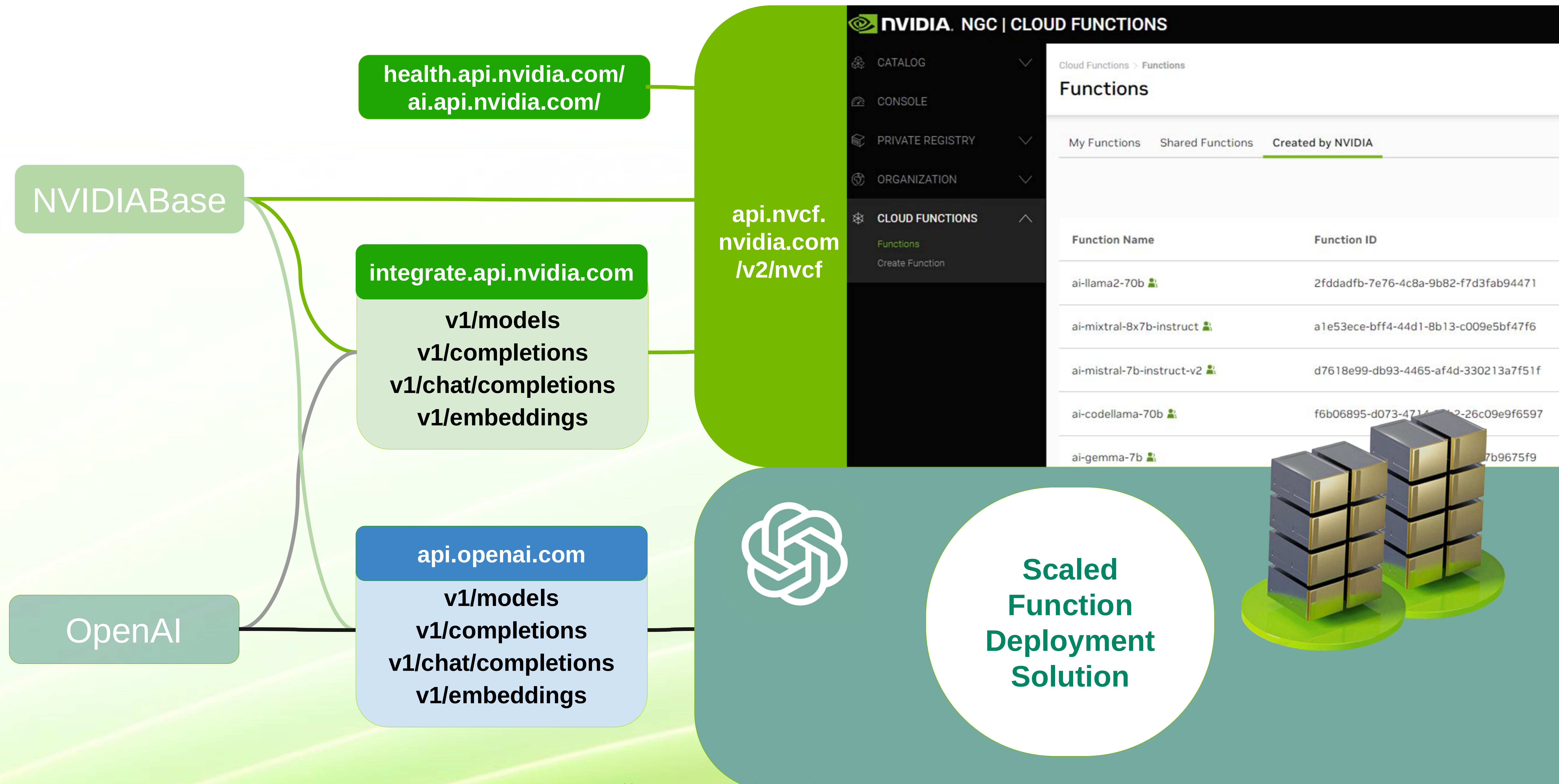


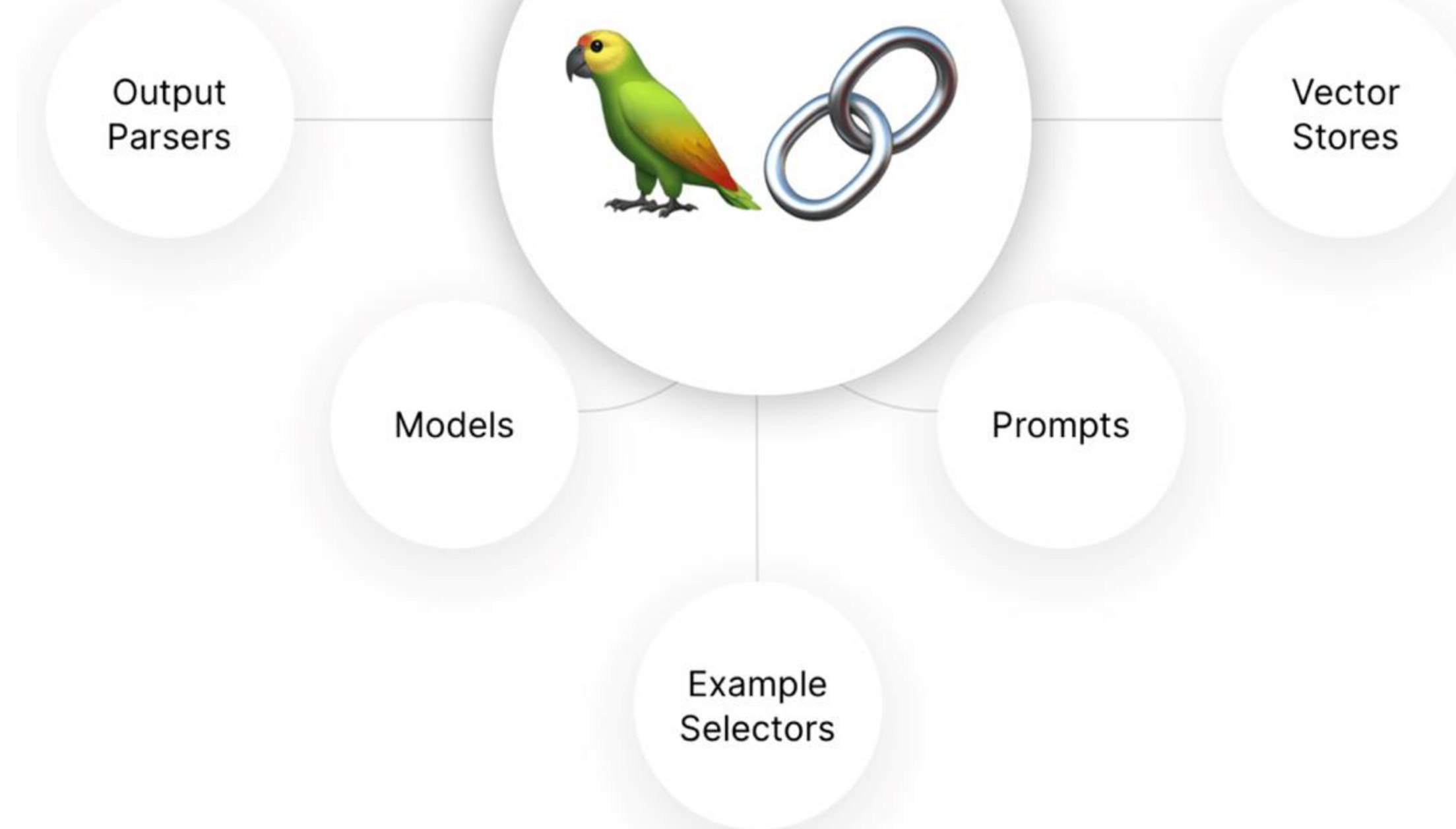
Scaled
Function
Deployment
Solution



LLM Interfaces

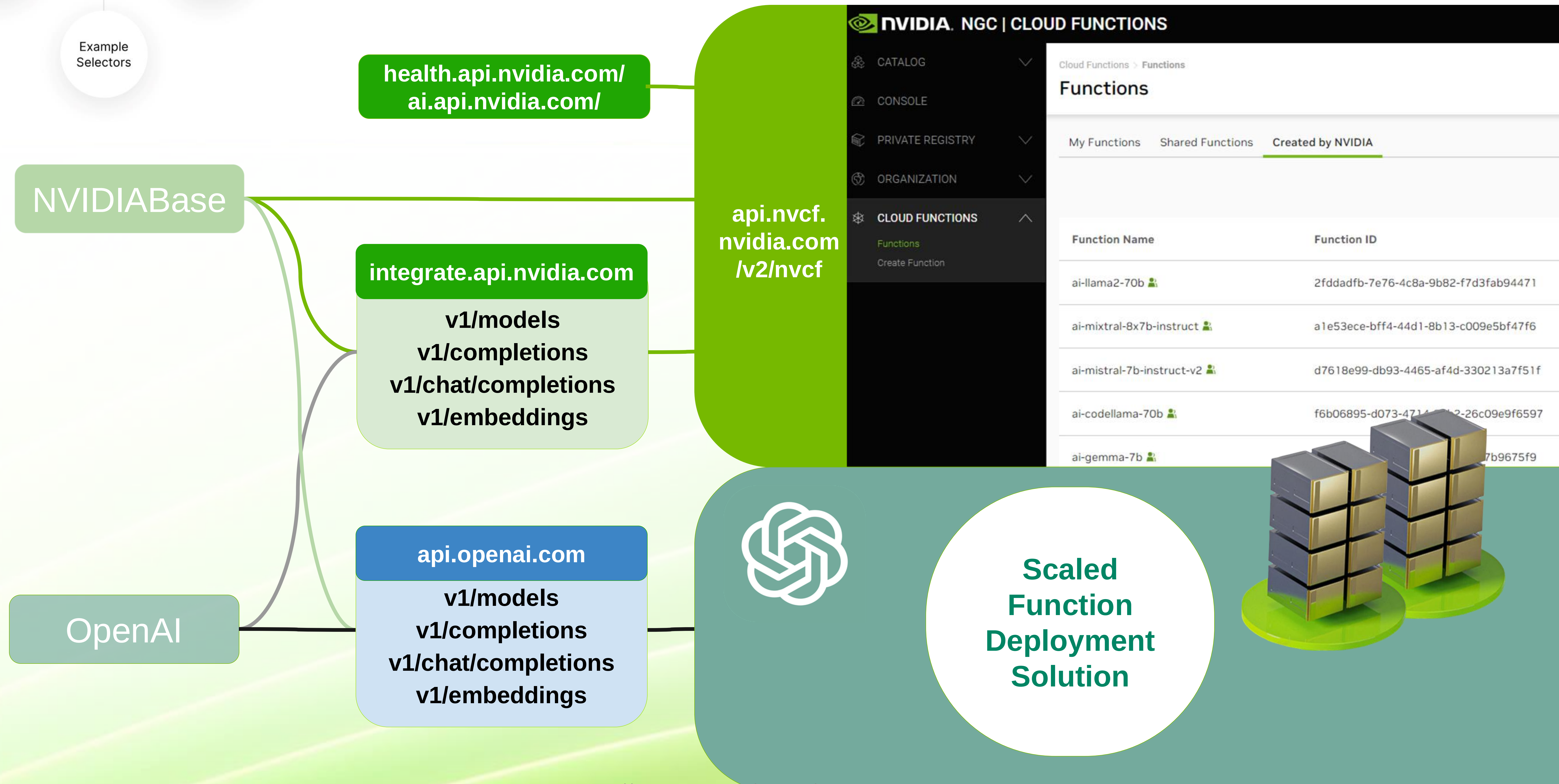
The Whole Stack





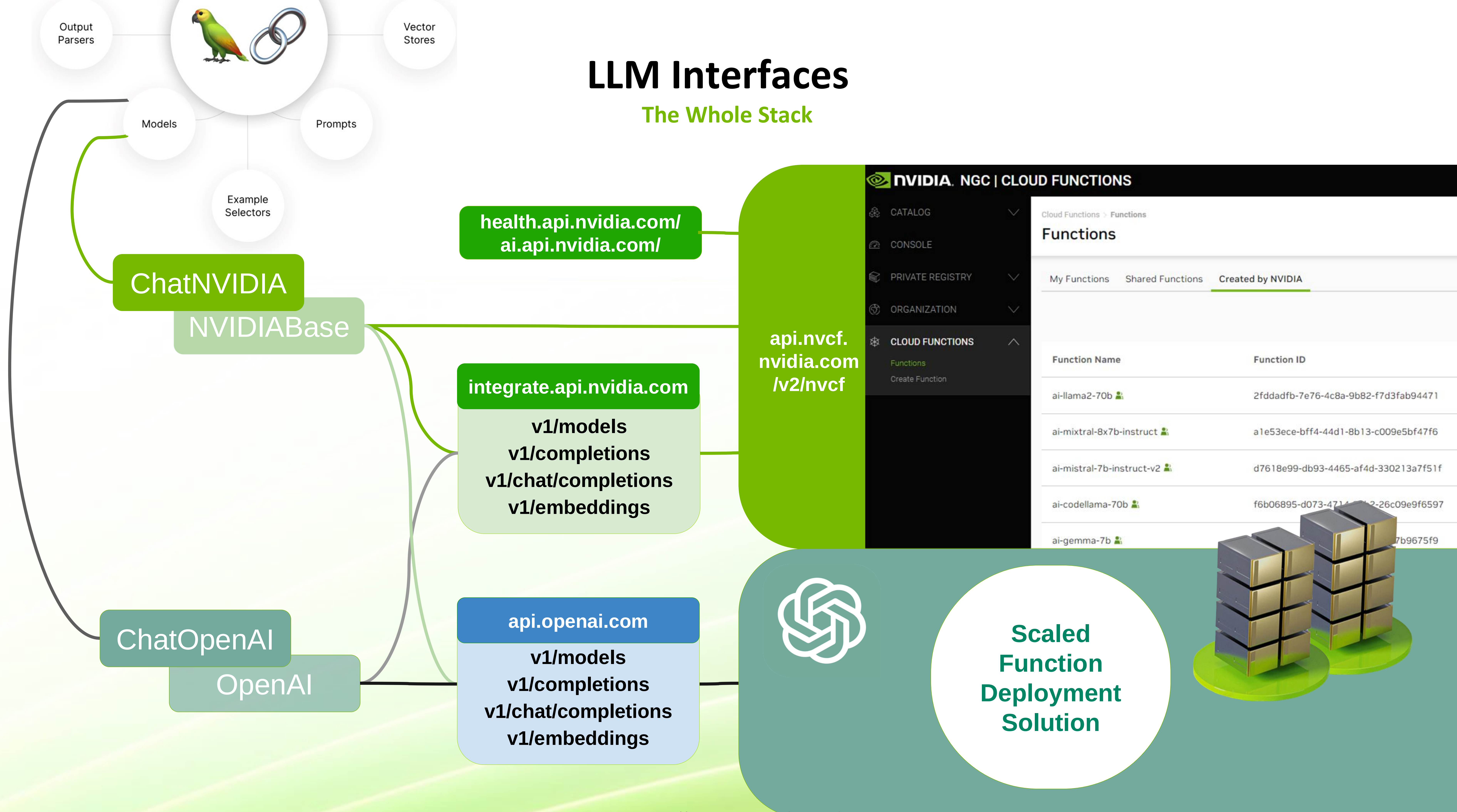
LLM Interfaces

The Whole Stack



LLM Interfaces

The Whole Stack



NVIDIA Building RAG Agents with LLMs

Part I

[0, 1, 2]

Building RAG Agents with LLMs



DEEP
LEARNING
INSTITUTE

This Lab 0 : 05 : 59 / 2 : 00 : 00

Course 1 : 47 : 00 / 32 : 00 : 00



LAUNCH



STOP TASK

Course Slides

Course Slides

Environment and LLMs

Environment [0, 1, 2]

- chatbot
- composer
- docker_router
- frontend
- imgs
- llm_client
- slides

- 00_jupyterlab.ipynb
- 01_microservices.ipynb
- 02_llms.ipynb
- 99_table_of_contents.ipynb

NVIDIA Building RAG Agents with LLMs Part II [3, 4]



Building RAG Agents with LLMs

Part 3: Intro to LangChain

LangChain Structure



Chain Building

Just The LLM



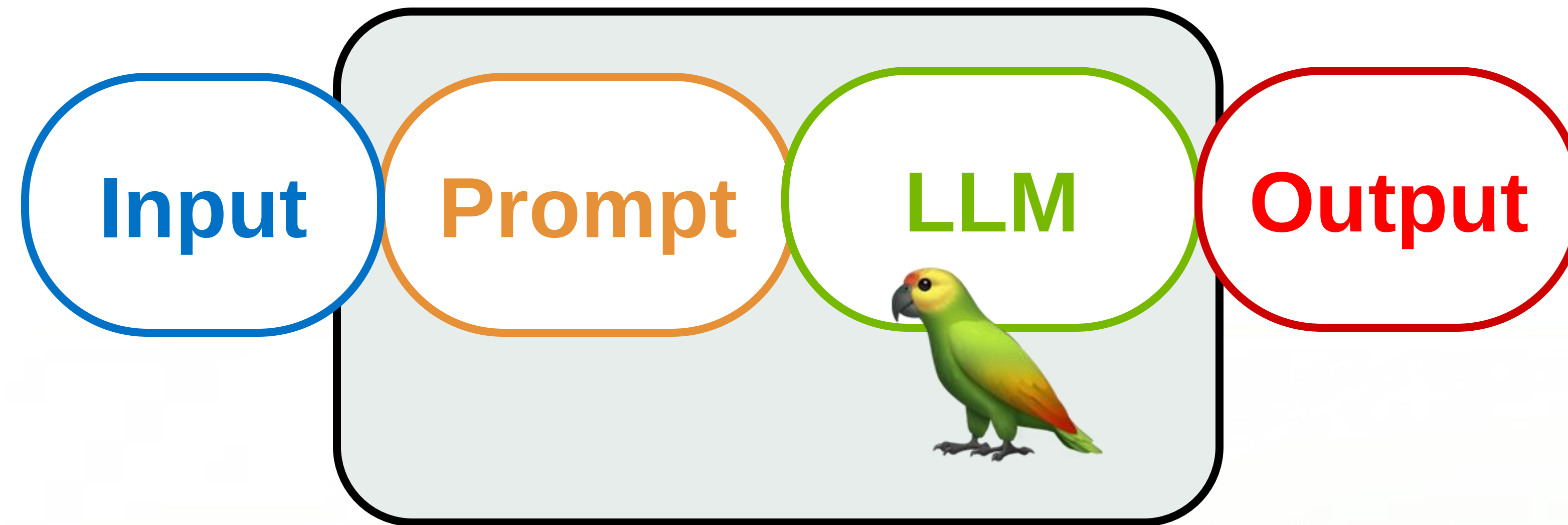
Chain Building

Just The LLM



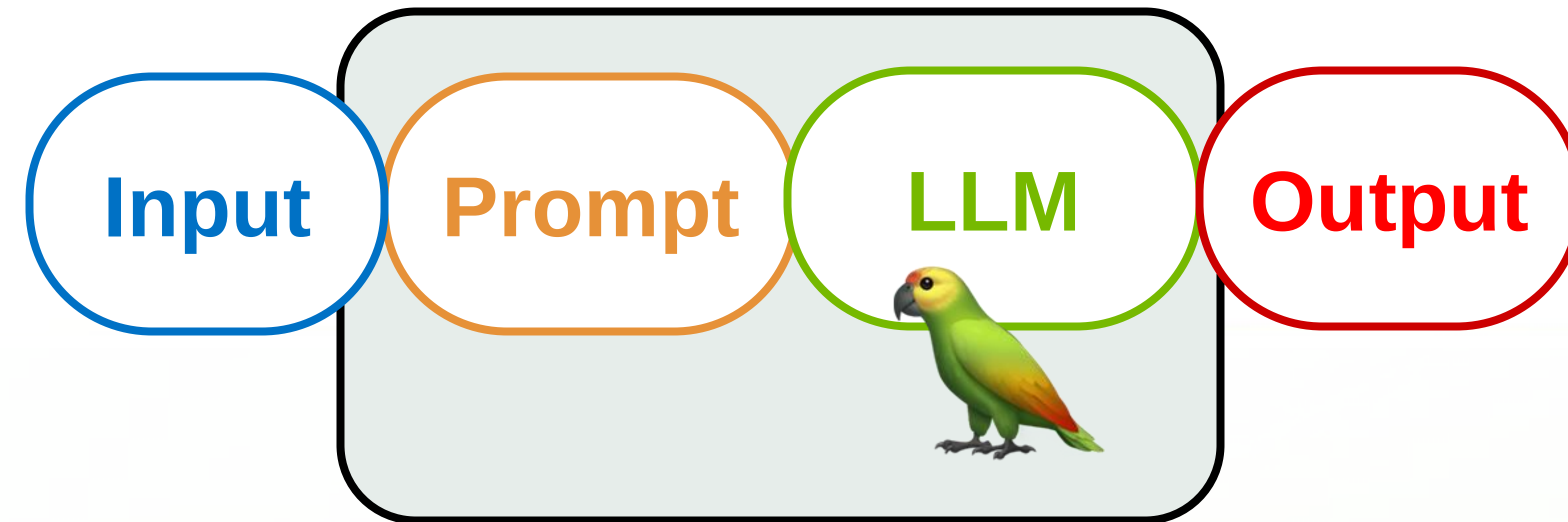
Chain Building

Simple Prompt+LLM Chain



Chain Building

Simple Prompt+LLM Chain

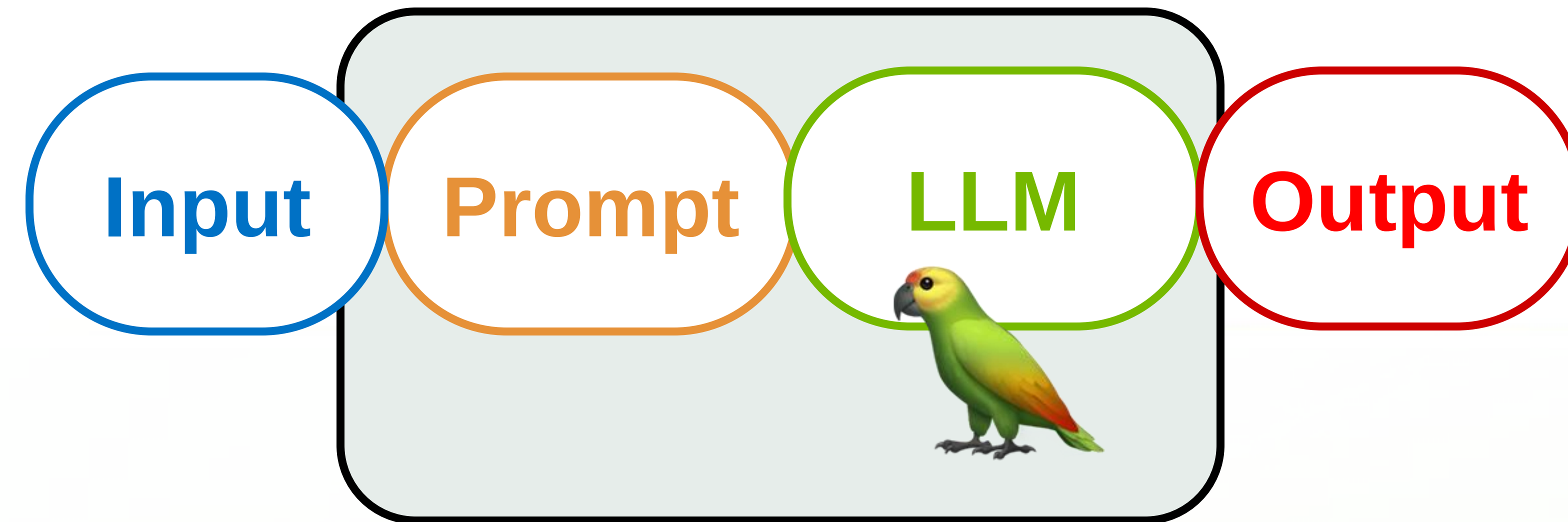


```
chain = prompt | chat  
chain.invoke(inputs)
```



Chain Building

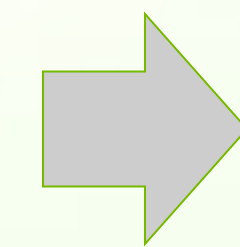
Simple Prompt+LLM Chain



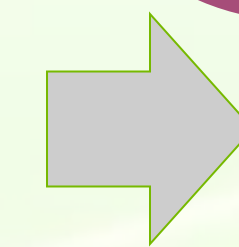
```
## StrOutputParser()  
def get_content(value):  
    return getattr(value, "content", value)
```

```
chain = prompt | chat | get_content  
chain.invoke(inputs)
```

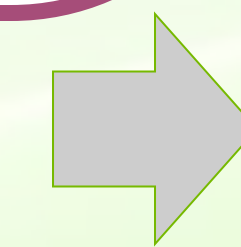
{'input' :
"Hey There"}



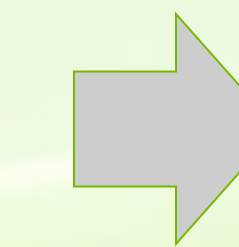
Prompt
Context: {context}
Input: {input}



Hey
There



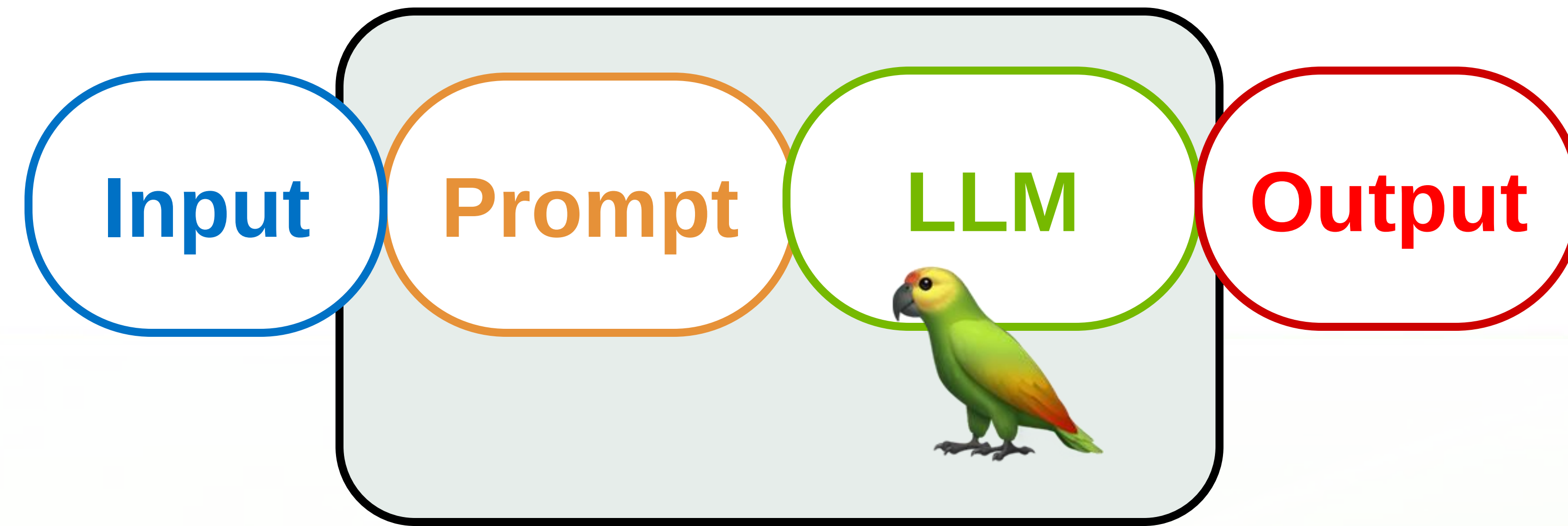
LLM



AI Message("Hello")

Chain Building

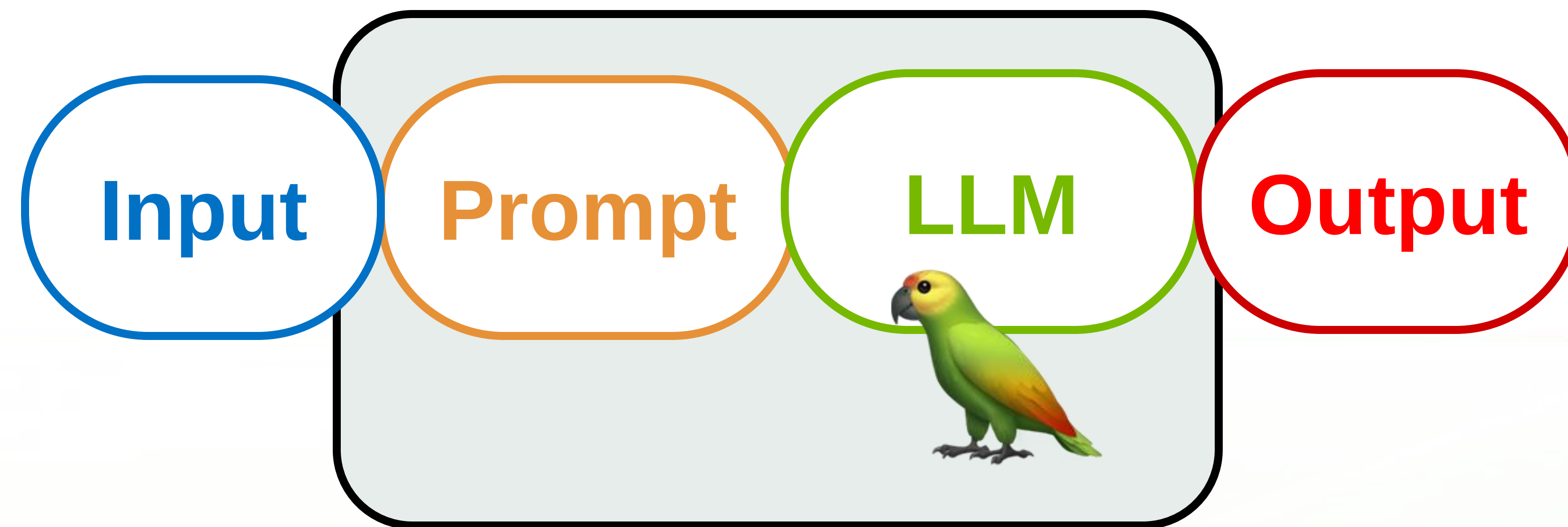
Invoking Runnables



```
chain = prompt | chat | StrOutputParser()
```

Chain Building

Invoking Runnables



```
chain = prompt | chat | StrOutputParser()
```

```
msg = chain.invoke(...)
```

```
for token in chain.stream(...)
```

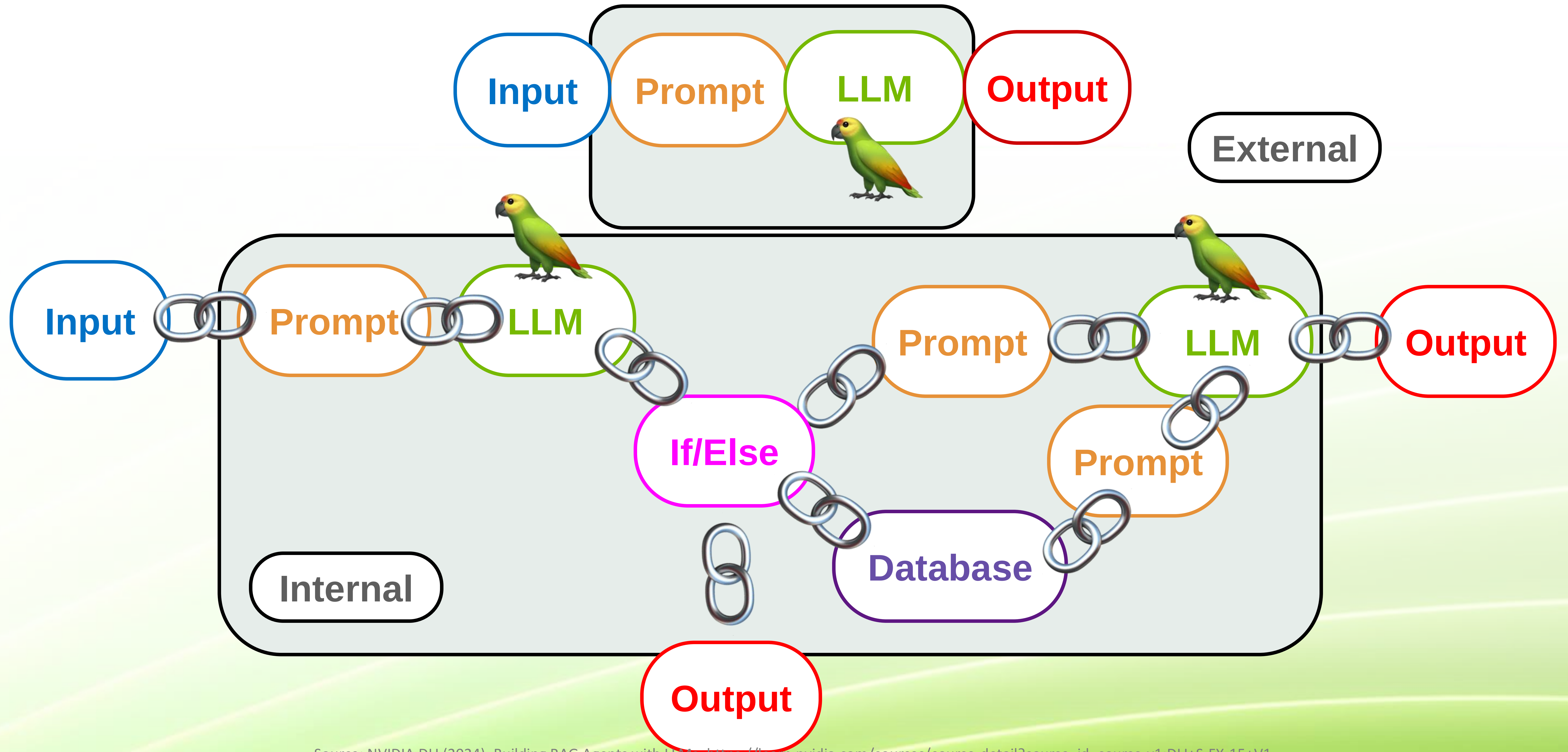
Chain Building

Building Information Pipelines

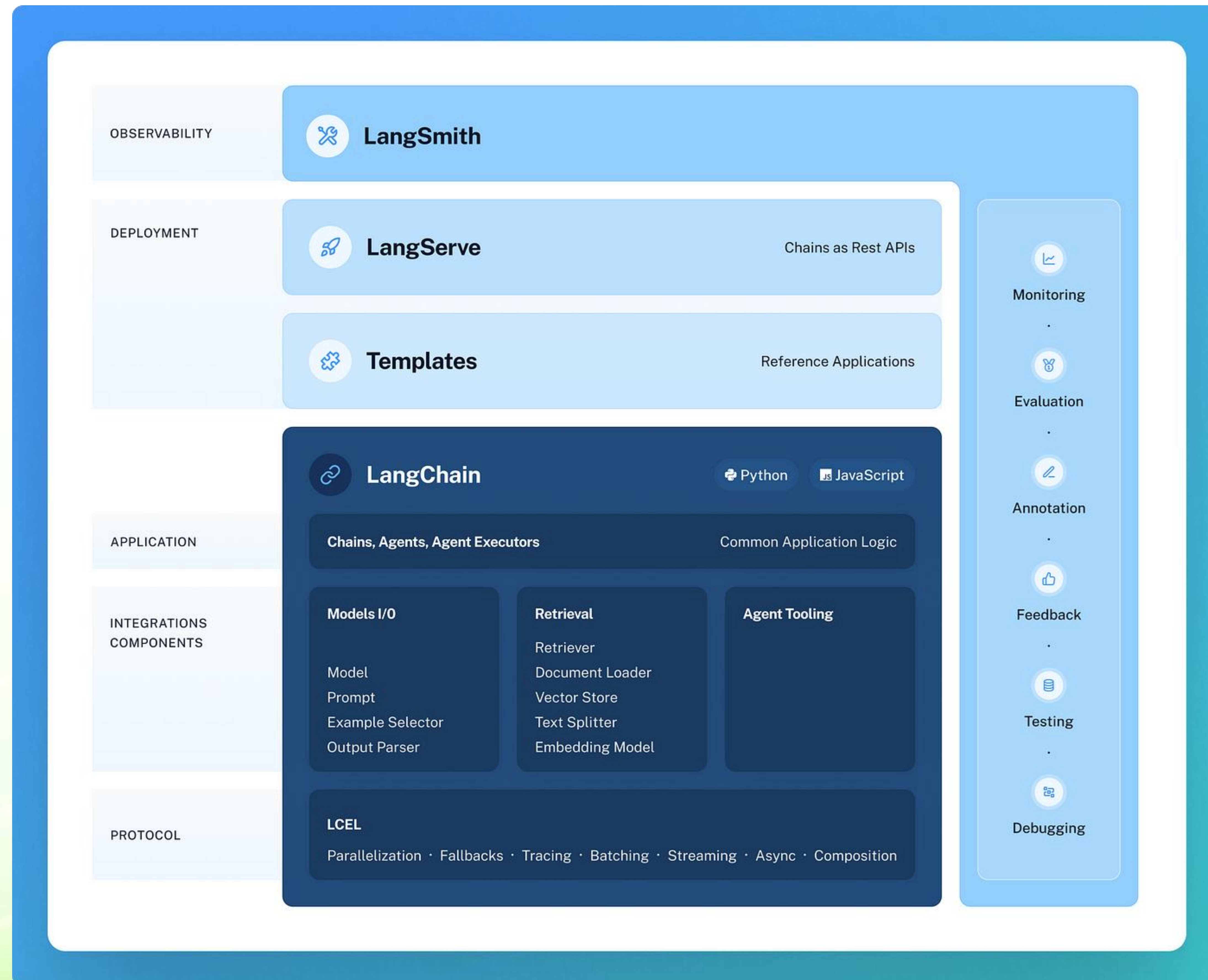


Chain Building

Building Information Pipelines



LangChain Extended Ecosystem



Chatbot

Hello World!!

You typed: Hello World!!

Type a message...

Submit

Retry

Undo

Clear

04_running_state.ipynb

File Edit View Insert Runtime Tools Help

Last edited on February

Comment Share

+ Code + Text

Connect

 **DEEP LEARNING INSTITUTE**

▼ **Notebook 4: Running State Chains**

In the previous notebook, we introduced some key LangChain Expression Language (LCEL) material regarding runnables. By now, you should be comfortable with both internal and external reasoning, as well as how to develop pipelines that facilitate it! In this notebook, we will make our way towards more advanced paradigms that will allow us to orchestrate more complex dialog management strategies and begin to execute on longer-form document reasoning.

Learning Objectives:

Source: NVIDIA DLI (2024), Building RAG Agents with LLMs, https://learn.nvidia.com/courses/course-detail?course_id=course-v1:DLI+S-FX-15+V1

Our Environment

Jupyter Notebooks + Frontend
+ LLM Client

The diagram illustrates the system architecture. A large green rounded rectangle represents the **Remote Host**. Inside this host, there are three main components: an orange **Frontend** box, a green **LLMs** box, and a dark grey **Jupyter Notebook Server** box. A grey circle labeled **Docker Router** is positioned between the Frontend and LLMs boxes. Arrows indicate the flow of data and connections: a curved arrow labeled **:8090** connects the Frontend to the Jupyter Notebook Server; a curved arrow labeled **:9012** connects the Frontend to the Docker Router; a curved arrow labeled **:9000** connects the LLMs to the Docker Router; and a curved arrow labeled **:88** connects the Jupyter Notebook Server to the Docker Router. To the right of the Remote Host, there is a stack of server hardware icons. A cloud icon is positioned between the Remote Host and the server hardware, with arrows indicating a connection between them.

132

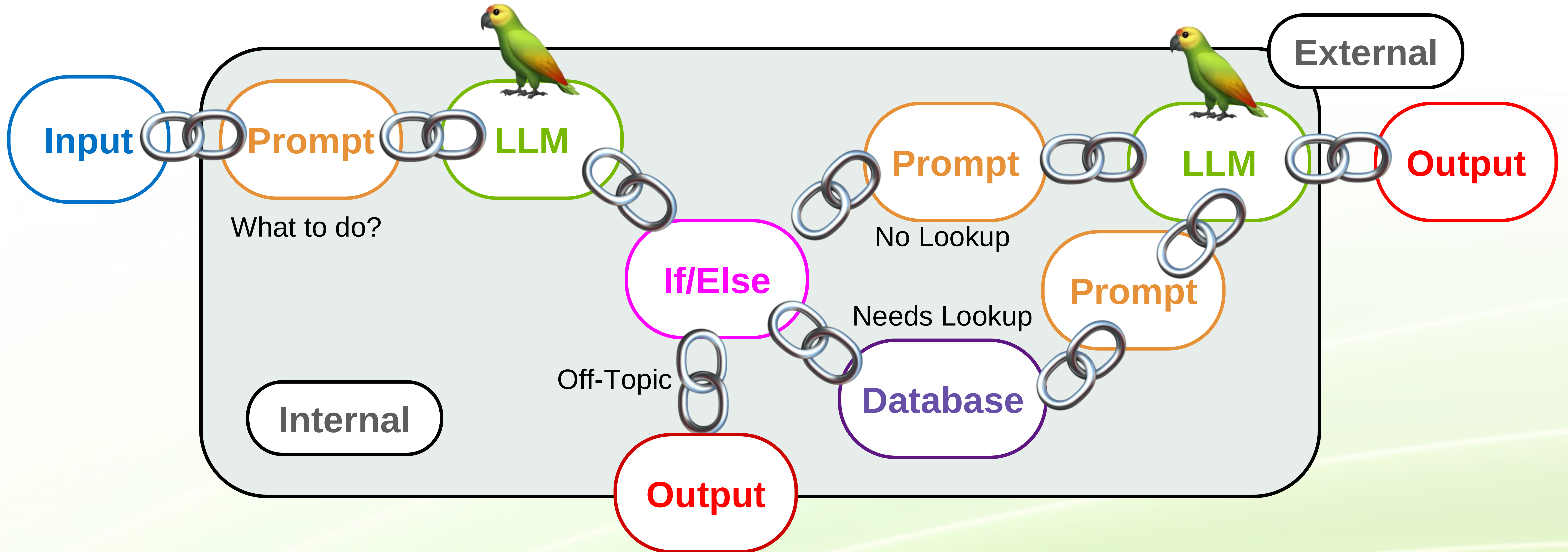


Building RAG Agents with LLMs

Part 4: Running State Chain

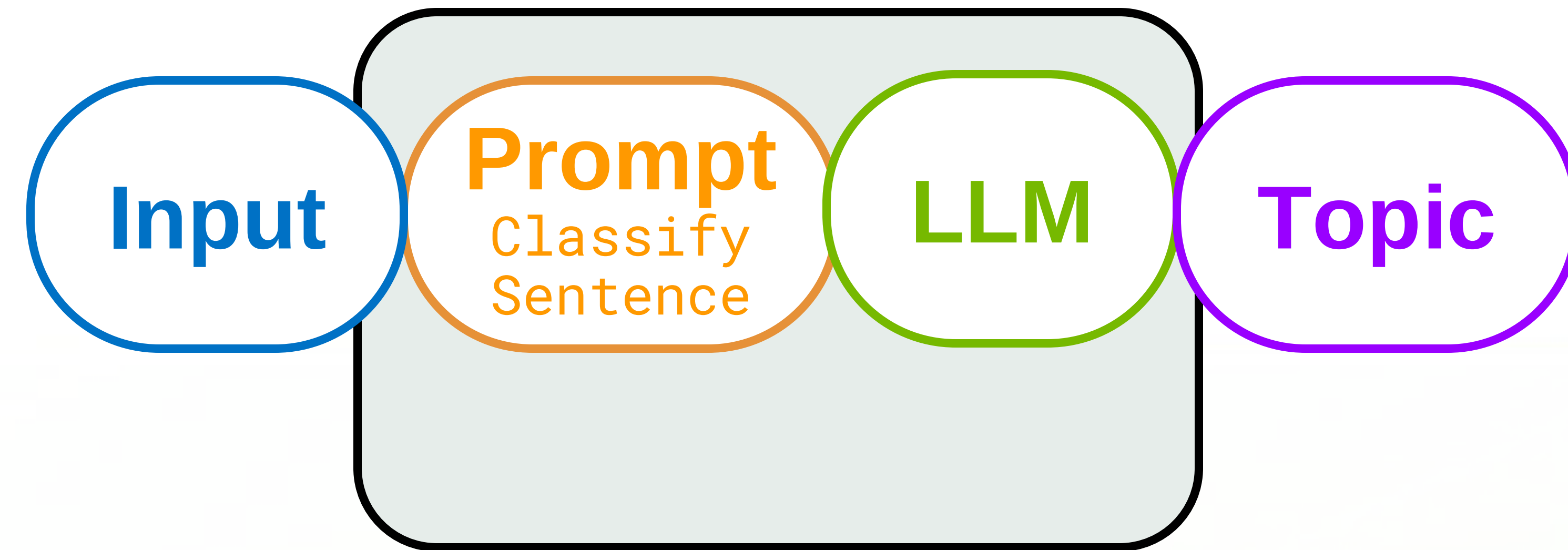
Chain Building

Recall Our Assumptions



Chain Building

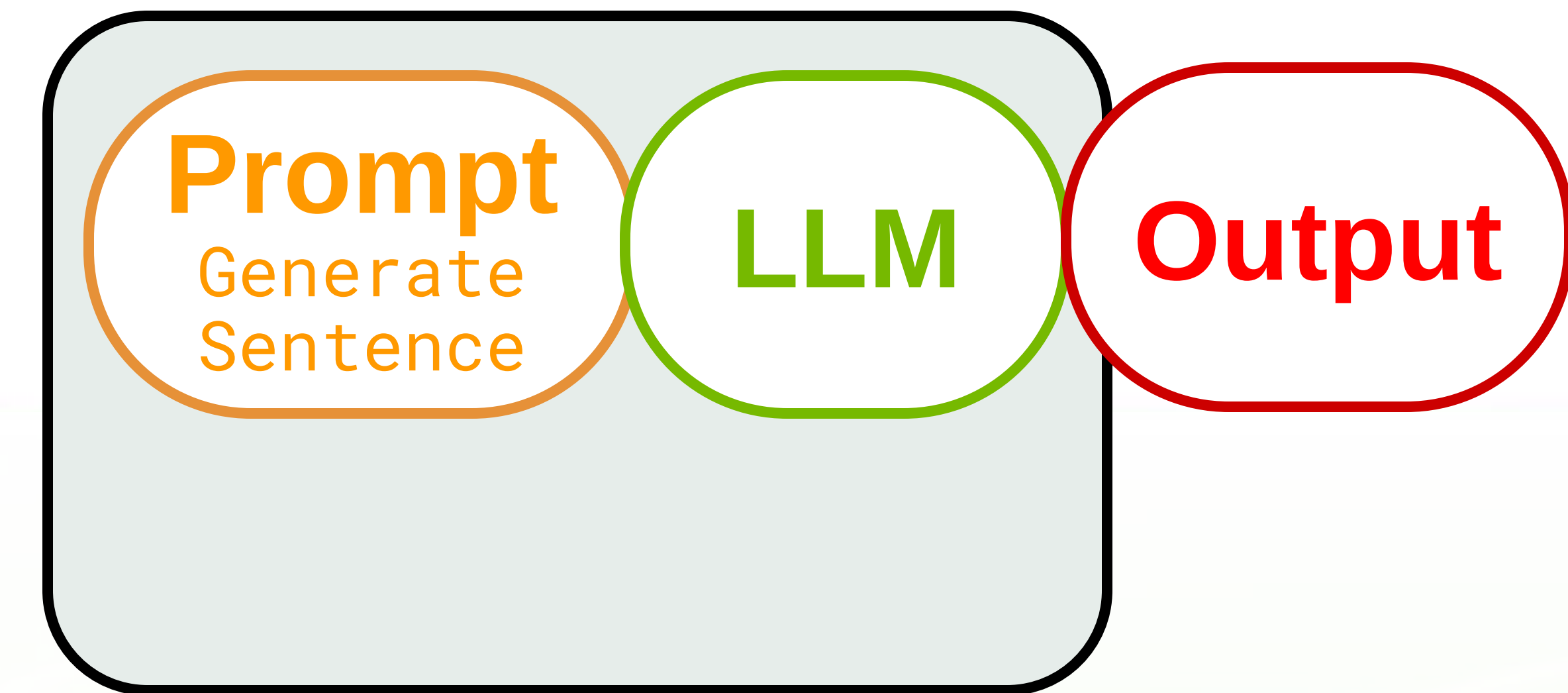
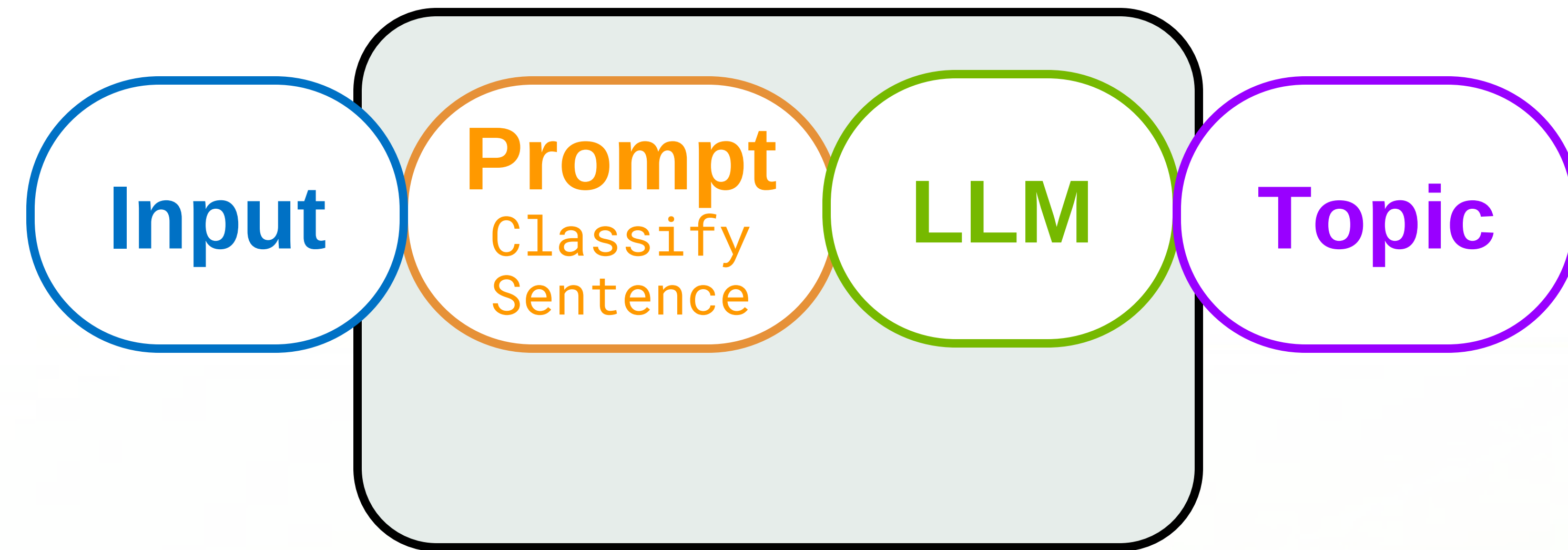
Towards Running State



```
inputs = {'input' : 'sentence'}  
cls_chain = cls_prompt | llm  
topic = cls_chain.invoke(inputs)
```

Chain Building

Towards Running State

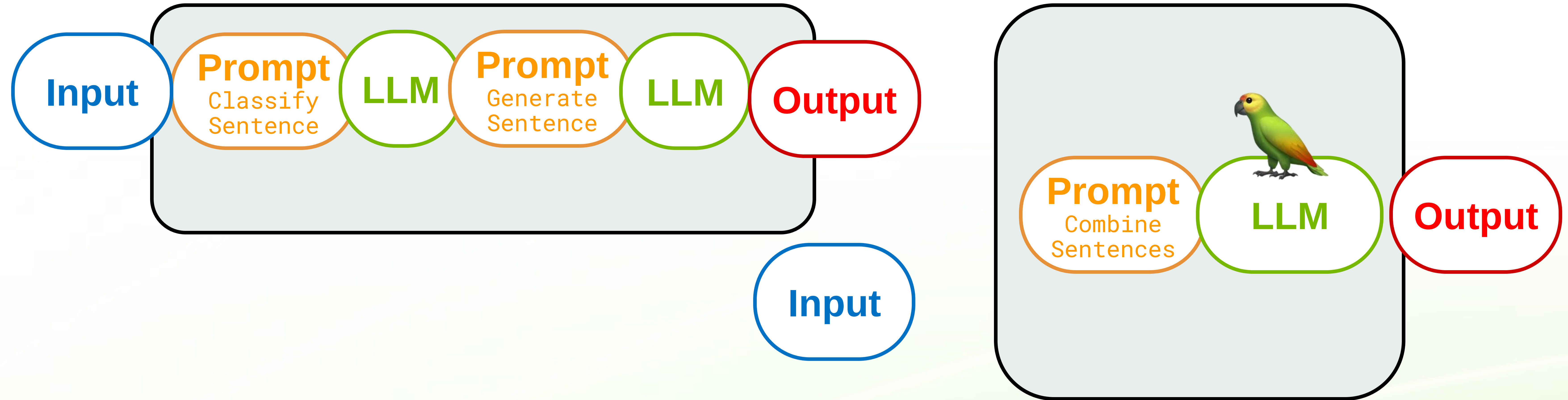


```
inputs = {'input' : 'sentence'}  
cls_chain = cls_prompt | llm  
topic = cls_chain.invoke(inputs)
```

```
gen_chain = out_prompt | llm  
out_chain = cls_chain | gen_chain  
for token in out_chain.stream(inputs):  
    print(token, end="")
```

Chain Building

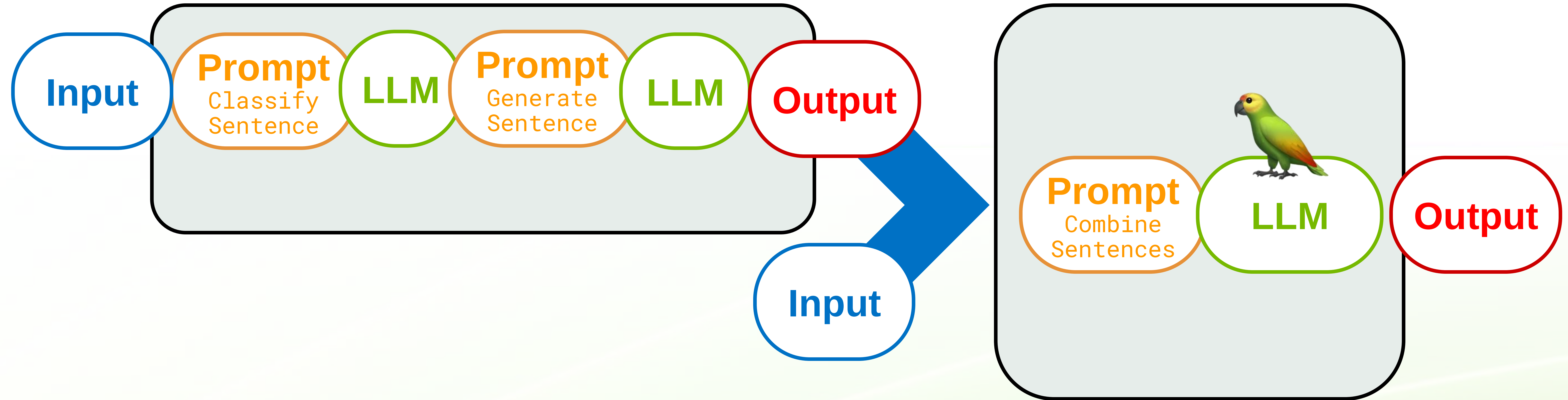
Towards Running State



```
inputs = {'input' : 'sentence'}  
new_sentence = out_chain.invoke(inputs)
```

Chain Building

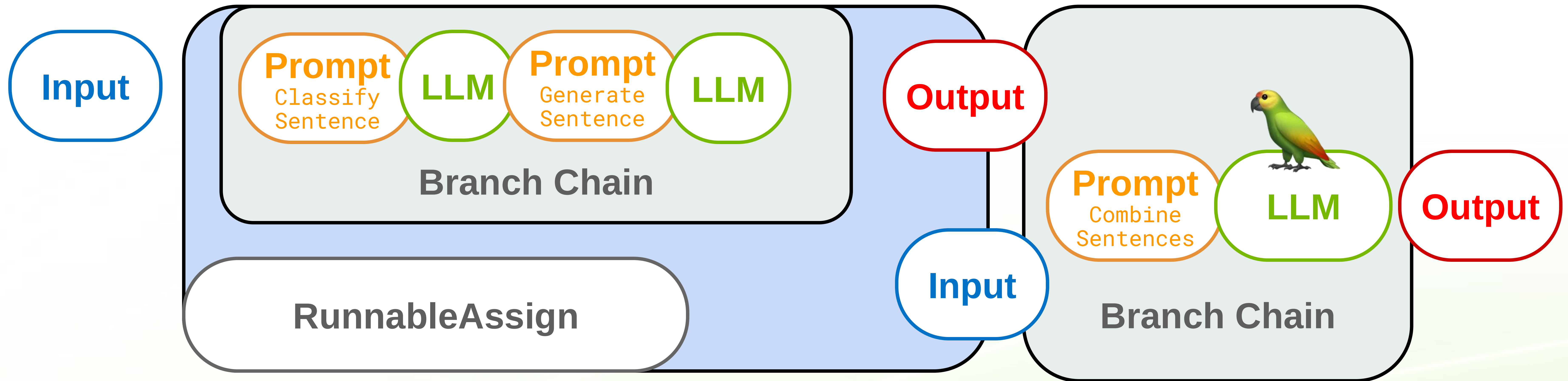
Towards Running State



```
inputs.update({'new' : 'new_sentence'})  
for token in merge_chain.stream(inputs):  
    print(token, end="")
```

Chain Building

Towards Running State



```
out_chain = RunnableAssign({  
    new_sentence : cls_chain | gen_chain  
}) | merge_chain
```

Typical Running State

Regular Fibonacci w/ While Loop

```
def fn():  
    ## Initial State  
    n = 8  
    fib = [0,1]  
  
    ## Running State  
    while len(fib) < n:  
        fib += [fib[-1] + fib[-2]]  
  
    ## Final State  
    print("\nFinal State:", locals())  
  
    ## Variable Pulled From State  
    print("\nFinal Output Variable:", fib)  
    return fib  
  
fn();
```

Initial State: {'n': 8, 'fib': [0, 1]}

Running State: {'n': 8, 'fib': [0, 1, 1]}

Running State: {'n': 8, 'fib': [0, 1, 1, 2]}

Running State: {'n': 8, 'fib': [0, 1, 1, 2, 3]}

Running State: {'n': 8, 'fib': [0, 1, 1, 2, 3, 5]}

Running State: {'n': 8, 'fib': [0, 1, 1, 2, 3, 5, 8]}

Running State: {'n': 8, 'fib': [0, 1, 1, 2, 3, 5, 8, 13]}

Final State: {'n': 8, 'fib': [0, 1, 1, 2, 3, 5, 8, 13]}

Final Output Variable: [0, 1, 1, 2, 3, 5, 8, 13]

Running State Chain Components

Towards LCEL While Loop

```
def get_initial_state():  
    n = 8  
    fib = [0,1]  
    return locals()  
  
initial_state = get_initial_state()  
initial_state  
  
{'n': 8, 'fib': [0, 1]}
```

State

n = 8
fib = [0,1]

Running State Chain Components

Towards LCEL While Loop

```
running_state = {'n': 8, 'fib': [0, 1]}
```

```
running_state['n'] = 7
```

```
running_state['fib'] += [1]
```

```
running_state['msg'] = "Hello"
```

```
{'n': [7], 'fib': [0, 1]}
```

```
{'n': [7], 'fib': [0, 1, 1]}
```

```
{'n': [7], 'fib': [0, 1, 1], 'msg': 'Hello'}
```

State

n = 8

fib = [0,1]

State

n = 7

fib = [0,1,1]

msg="Hello"

Running State Chain Components

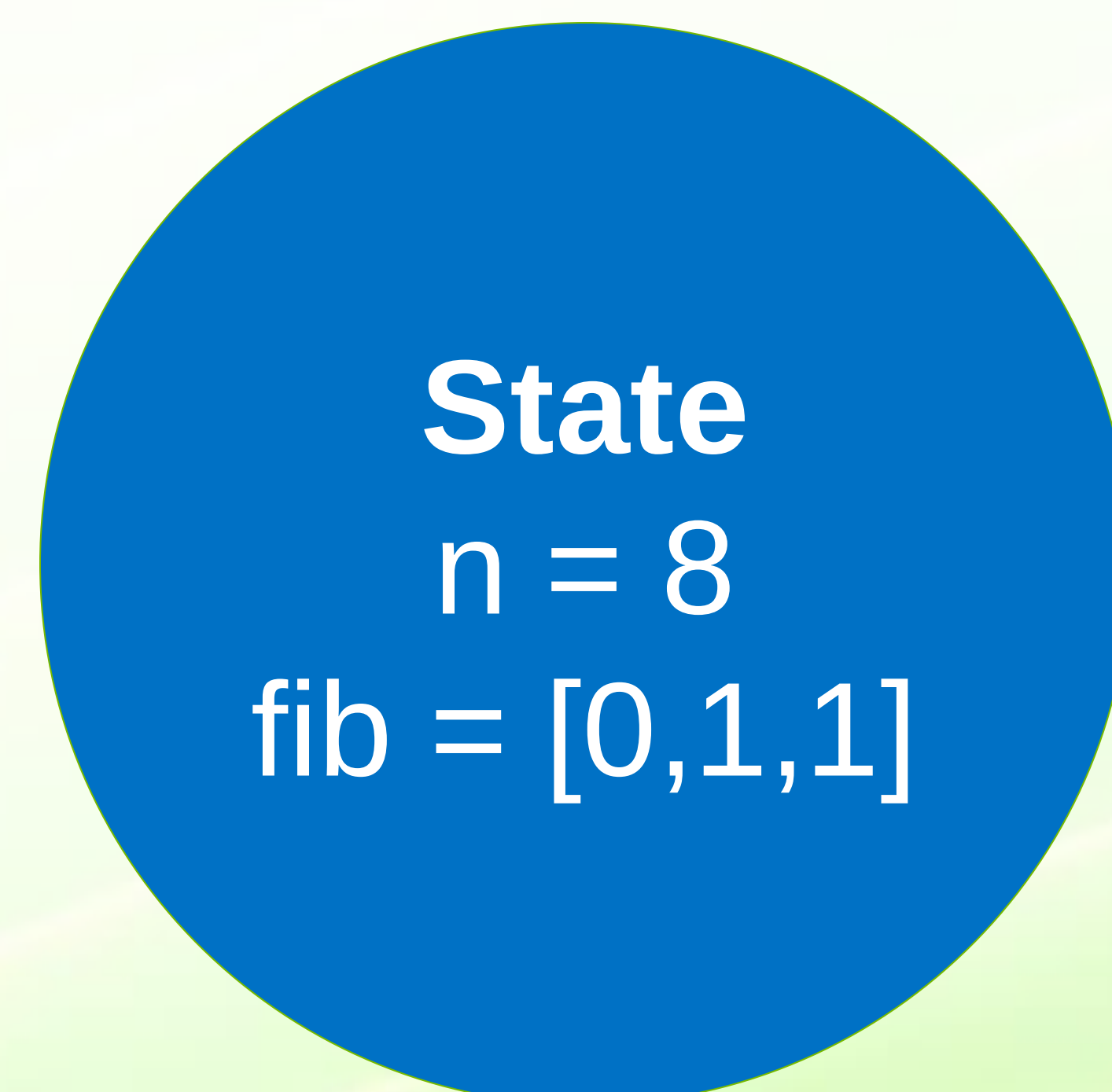
Towards LCEL While Loop

```
from langchain_core.runnables import RunnableLambda

def next_fib_fn(d):
    return d['fib'] + [d['fib'][-1] + d['fib'][-2]]

fib_branch = RunnableLambda(next_fib_fn)
fib_branch.invoke(running_state)

[0, 1, 1, 2]
```



Running State Chain Components

Towards LCEL While Loop

```
from langchain_core.runnables import RunnableLambda

def next_fib_fn(d):
    return d['fib'] + [d['fib'][-1] + d['fib'][-2]]

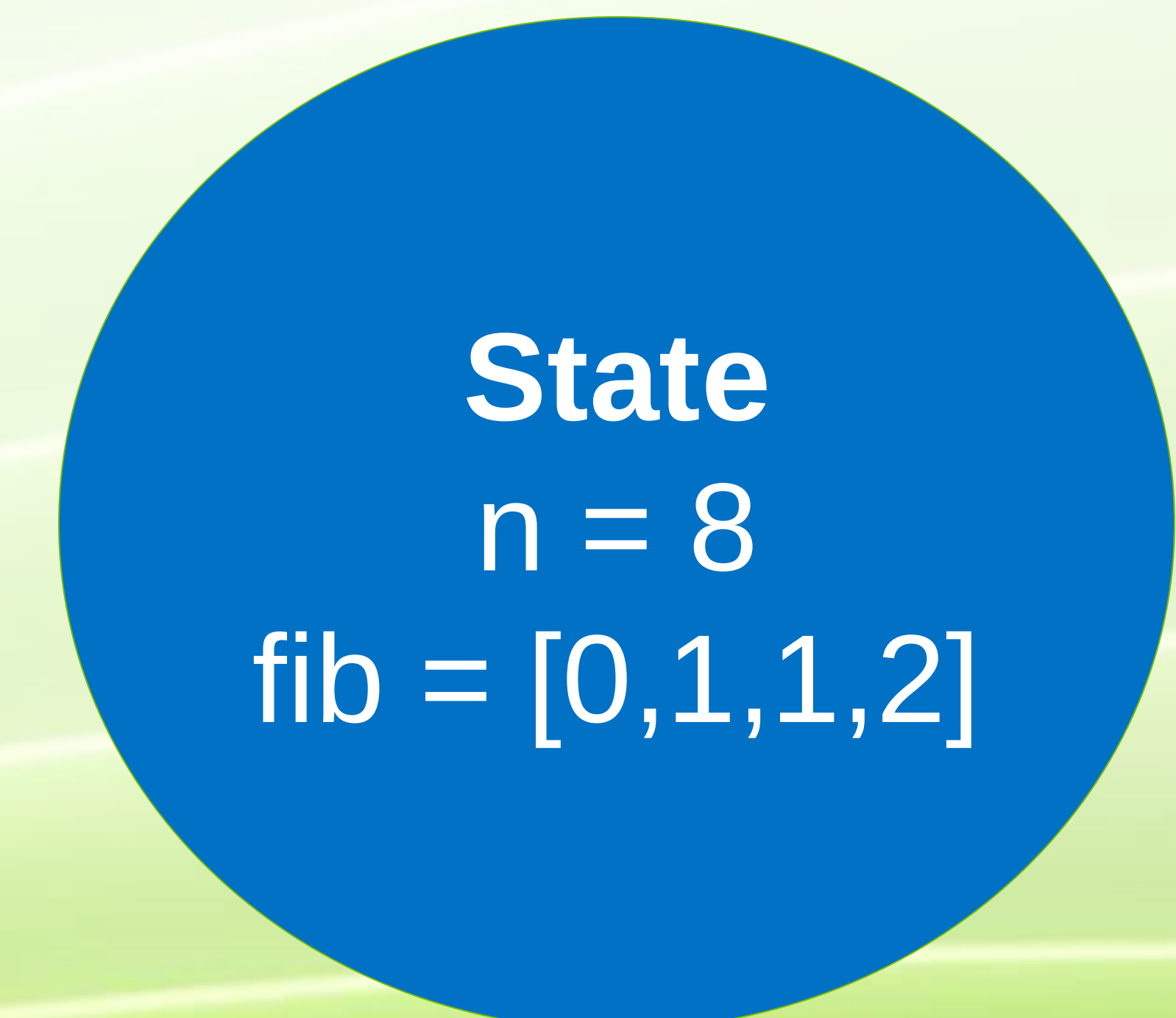
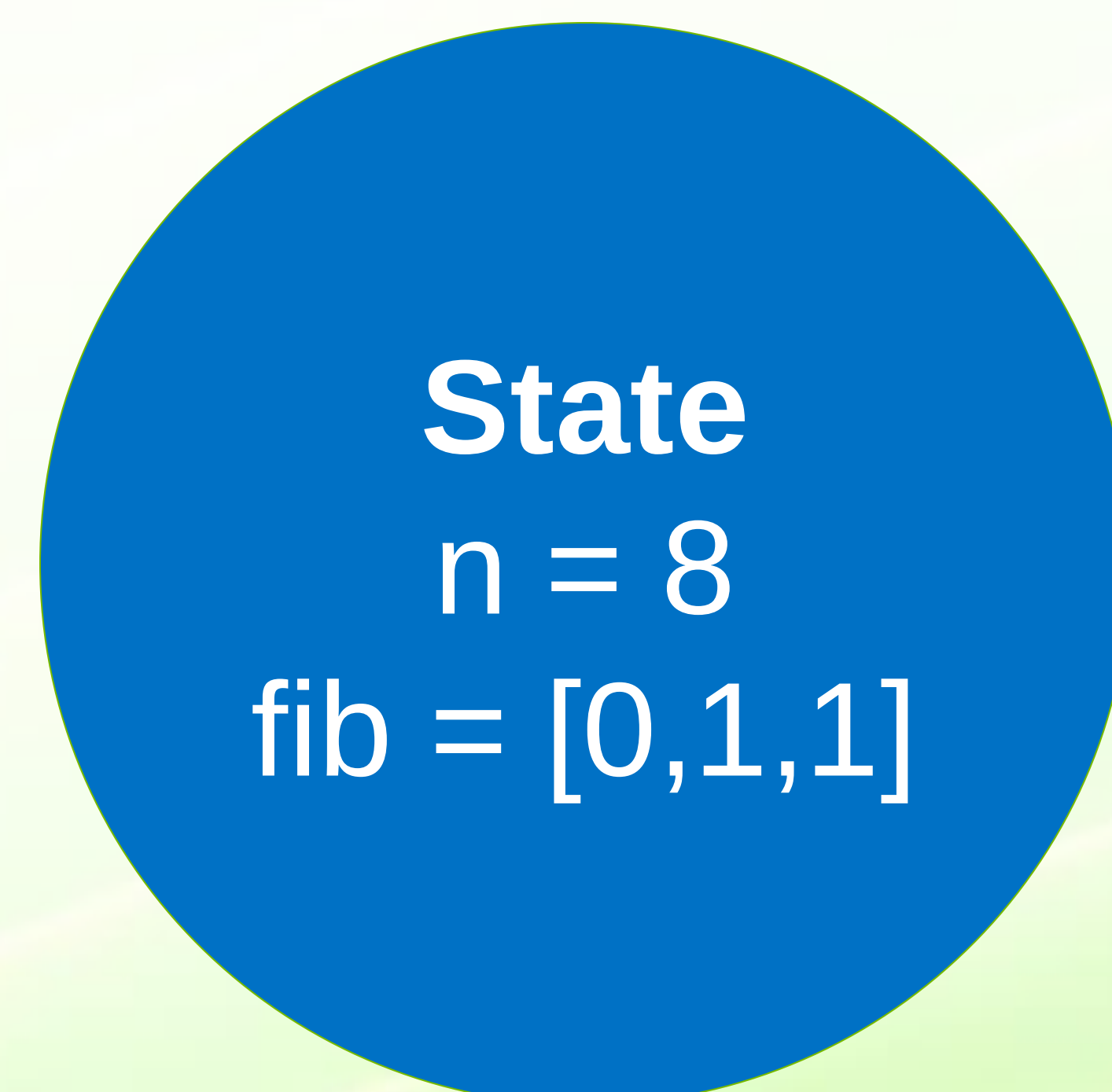
fib_branch = RunnableLambda(next_fib_fn)
fib_branch.invoke(running_state)

[0, 1, 1, 2]
```

```
from langchain_core.runnables import RunnableAssign

rstate_chain = RunnableAssign({'fib' : fib_branch})
running_state = initial_state.copy()
running_state = rstate_chain.invoke(running_state)

running_state: {'n': 8, 'fib': [0, 1, 1, 2]}
running_state: {'n': 8, 'fib': [0, 1, 1, 2, 3]}
```



Running State Chain Components

Towards LCEL While Loop

```
from langchain_core.runnables import RunnableLambda

def next_fib_fn(d):
    return d['fib'] + [d['fib'][-1] + d['fib'][-2]]

fib_branch = RunnableLambda(next_fib_fn)
fib_branch.invoke(running_state)

[0, 1, 1, 2]
```

```
from langchain_core.runnables import RunnableAssign

rstate_chain = RunnableAssign({'fib' : fib_branch})
running_state = initial_state.copy()
running_state = rstate_chain.invoke(running_state)
running_state = rstate_chain.invoke(running_state)

running_state: {'n': 8, 'fib': [0, 1, 1, 2]}
running_state: {'n': 8, 'fib': [0, 1, 1, 2, 3]}
```

State
n = 8
fib = [0,1,1]

State
n = 8
fib = [0,1,1,2]

State
n = 8
fib =
[0,1,1,2, 3]

Final Running State Loop

```
from langchain_core.runnables import RunnableAssign

def next_fib_fn(d):
    fib = d.get('fib')
    return fib + [fib[-1] + fib[-2]]

rstate_chain = RunnableAssign({'fib' : next_fib_fn})

running_state = {
    'n': 8,
    'fib': [0, 1]
}

while len(running_state['fib']) < running_state['n']:
    running_state = rstate_chain.invoke(running_state)

running_state: {'n': 8, 'fib': [0, 1, 1]}
running_state: {'n': 8, 'fib': [0, 1, 1, 2]}
running_state: {'n': 8, 'fib': [0, 1, 1, 2, 3]}
running_state: {'n': 8, 'fib': [0, 1, 1, 2, 3, 5]}
running_state: {'n': 8, 'fib': [0, 1, 1, 2, 3, 5, 8]}
running_state: {'n': 8, 'fib': [0, 1, 1, 2, 3, 5, 8, 13]}
```

Typical Running State Loop

Comparing Typical with Running State

```
def fn():
    ## Initial State
    n = 8
    fib = [0,1]

    ## Running State
    while len(fib) < n:
        fib += [fib[-1] + fib[-2]]

    ## Final State
    print("\nFinal State:", locals())

    ## Variable Pulled From State
    print("\nFinal Output Variable:", fib)
    return fib
```

```
from langchain_core.runnables import RunnableAssign

def next_fib_fn(d):
    fib = d.get('fib')
    return fib + [fib[-1] + fib[-2]]

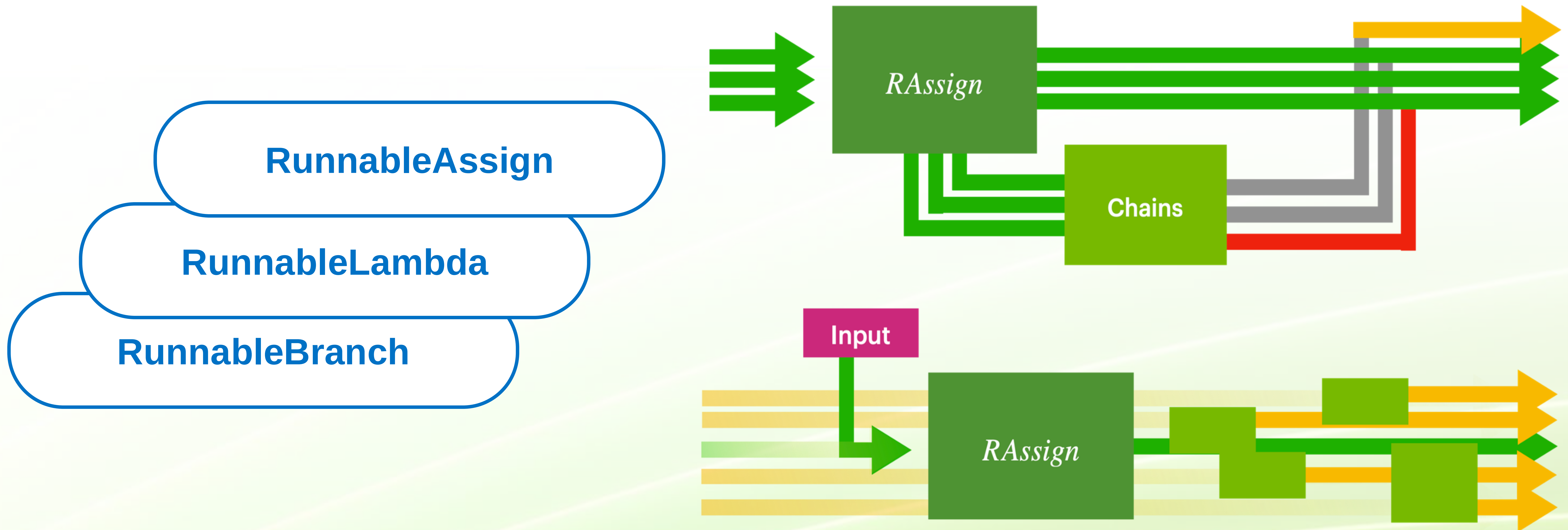
rstate_chain = RunnableAssign({'fib' : next_fib_fn})

running_state = {
    'n': 8,
    'fib': [0, 1]
}

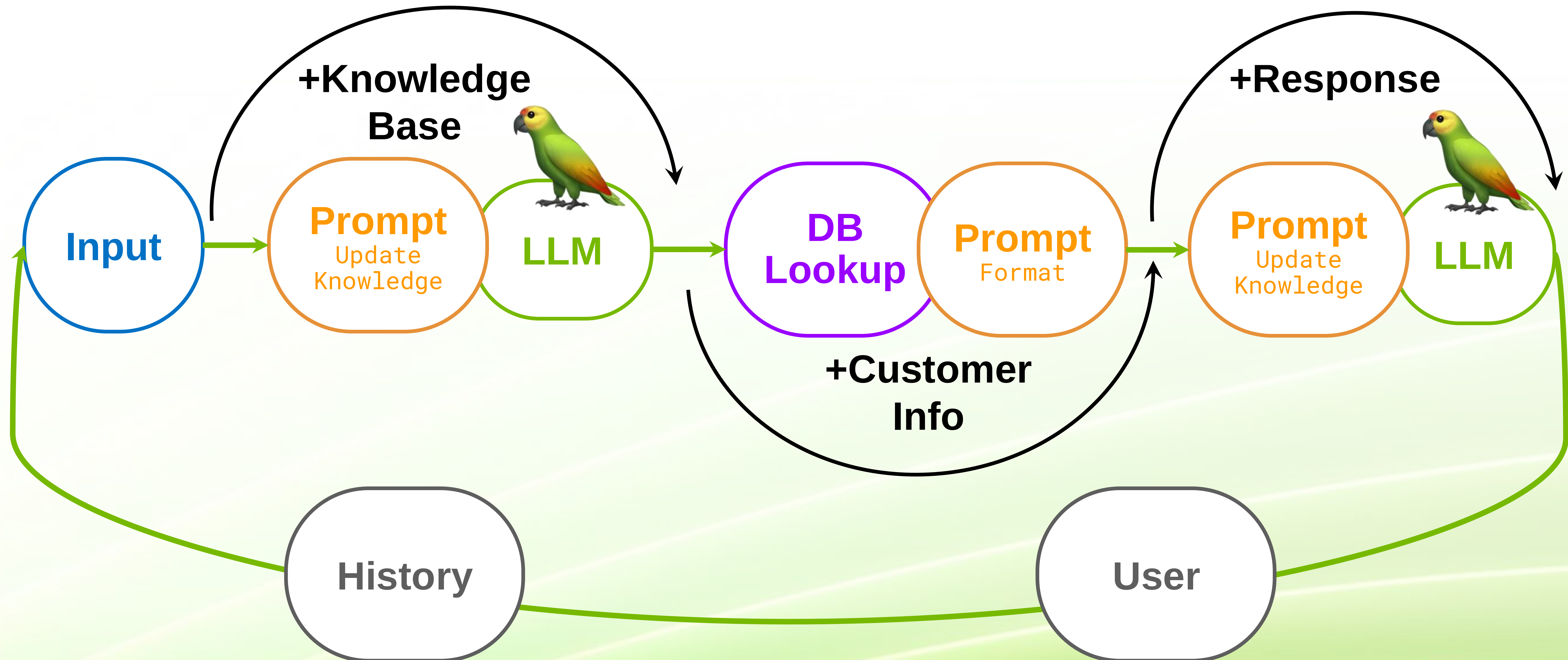
while len(running_state['fib']) < running_state['n']:
    running_state = rstate_chain.invoke(running_state)
```

Final Running State Loop

Big Picture

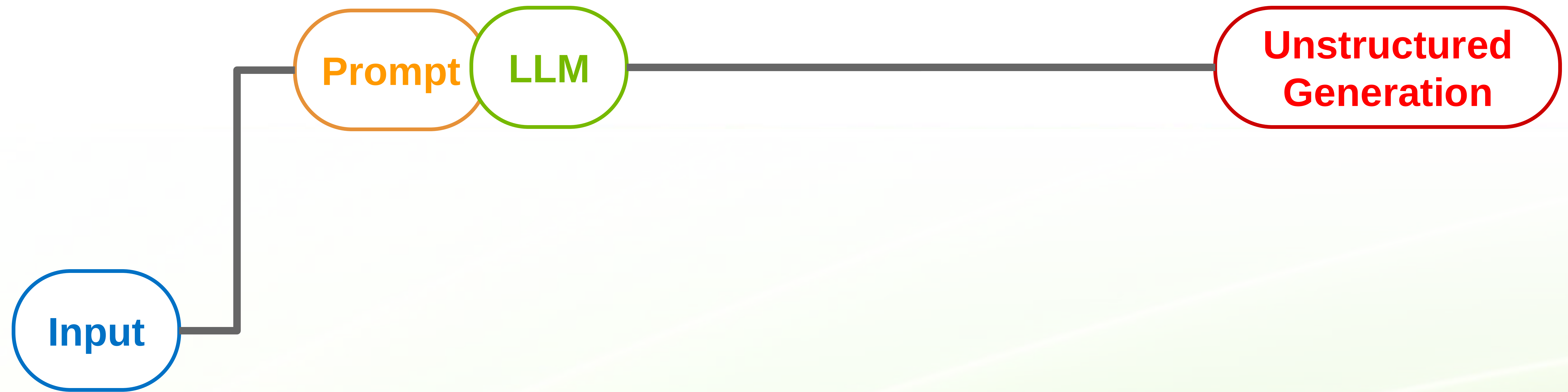


Airline Chatbot



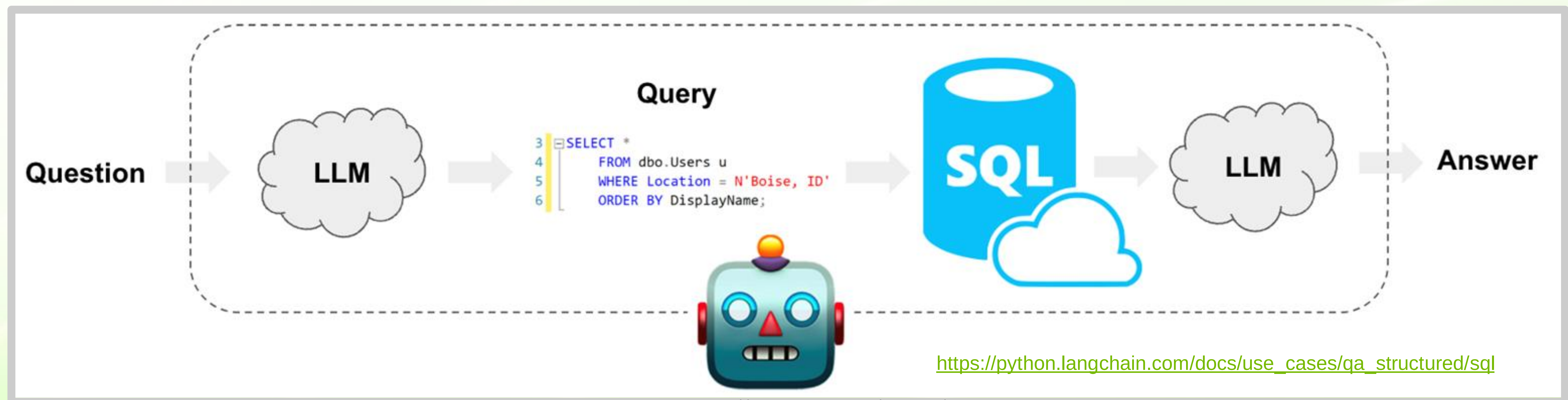
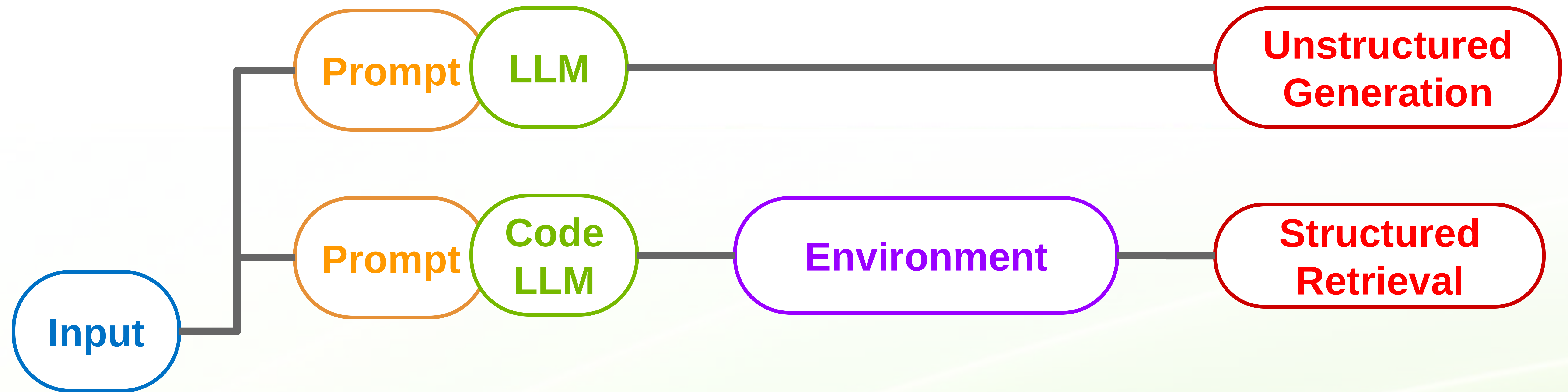
Modern Chain Paradigms

Towards Powerful Running State



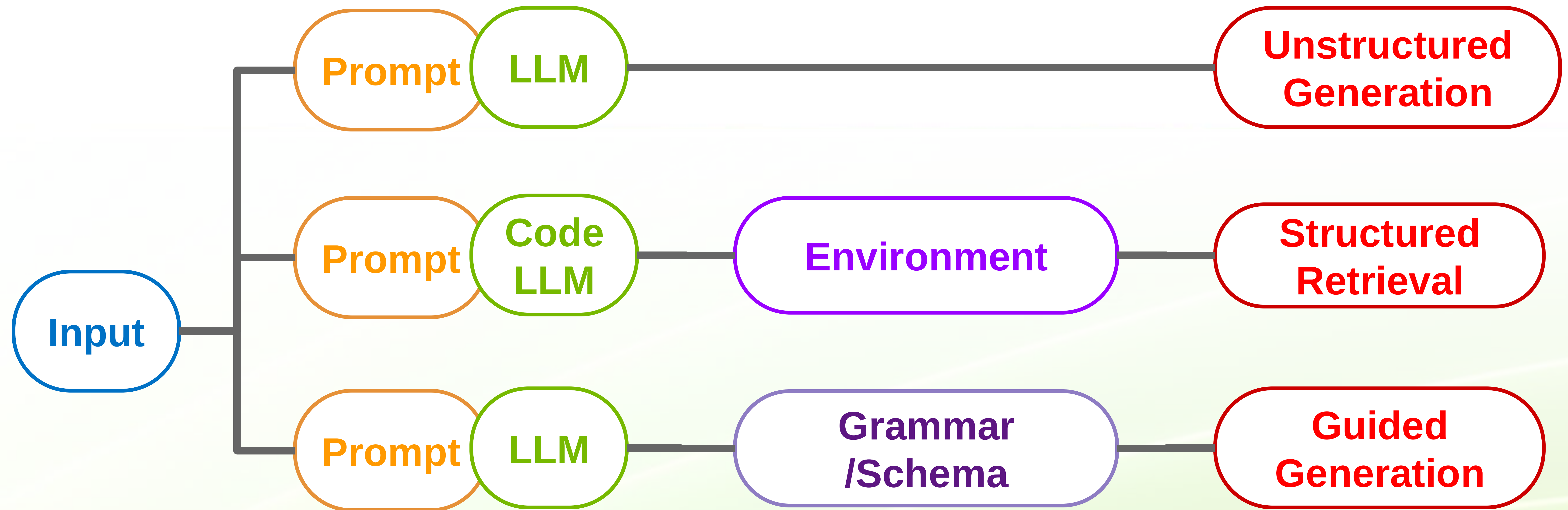
Modern Chain Paradigms

Towards Powerful Running State



Modern Chain Paradigms

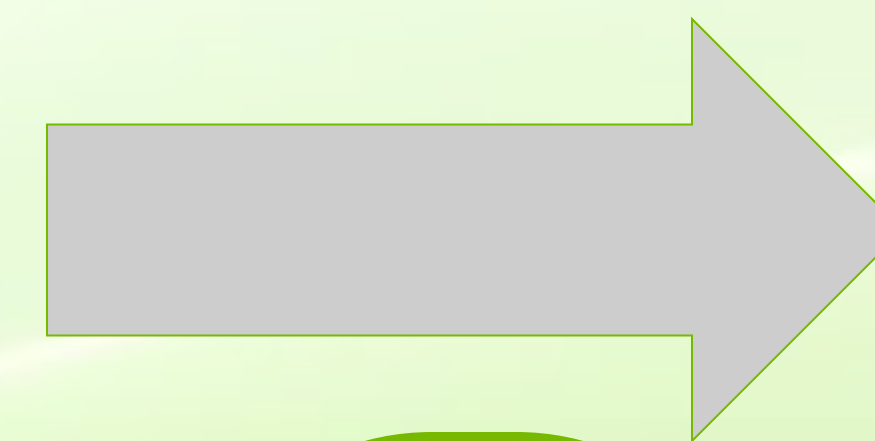
Towards Powerful Running State



```
{"first_name" : "unknown",  
  "last_name" : "unknown",  
  "confirmation" : -1}
```

Update given new info:

"Sure, my name is Jane Doe!"

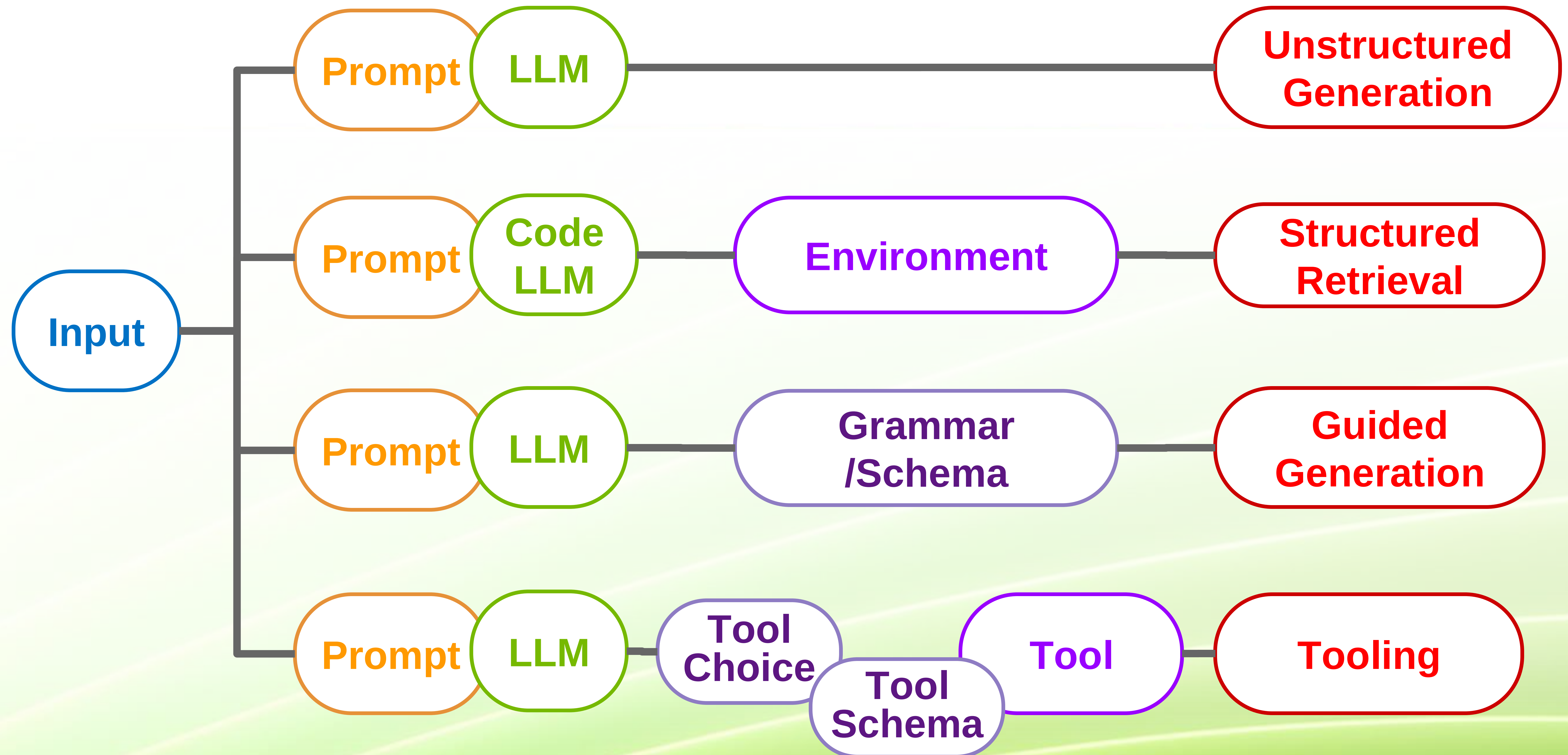


LLM

```
{"first_name" : "Jane",  
  "last_name" : "Doe",  
  "confirmation" : -1}
```

Modern Chain Paradigms

Towards Powerful Running State



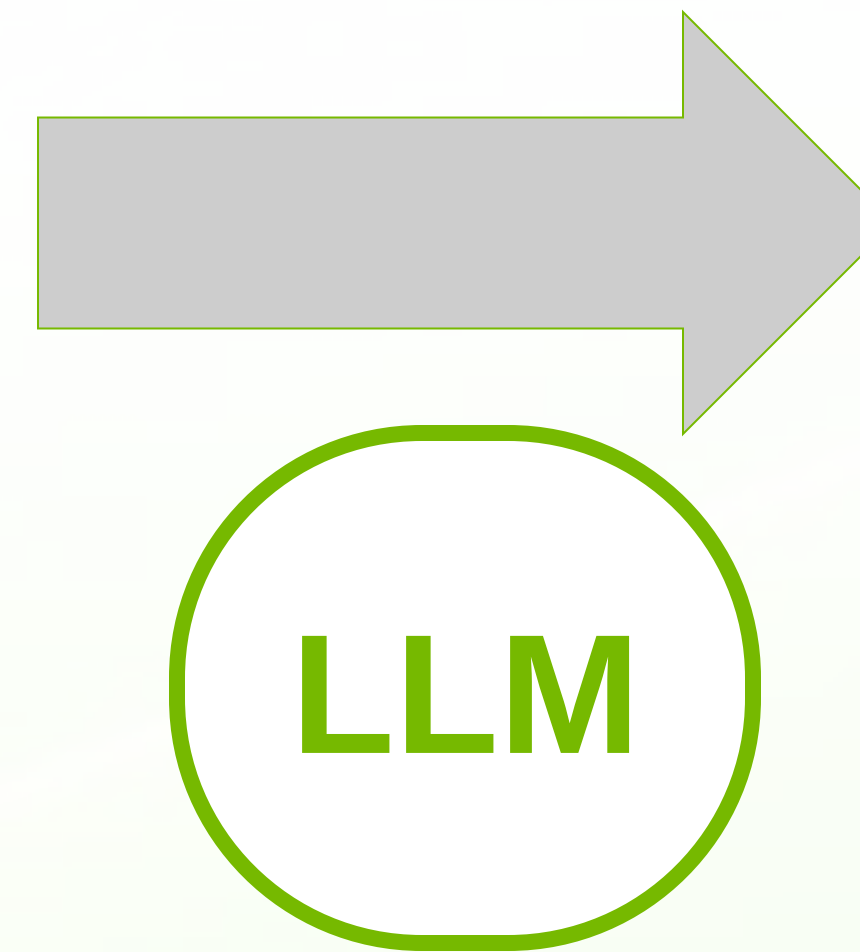
Final Objective

Knowledge Base + Running State Chain

```
{  
  "first_name" : "unknown",  
  "last_name" : "unknown",  
  "confirmation" : -1,  
  ...  
}
```

Update given new info:

"Sure, my name is Jane Doe!"

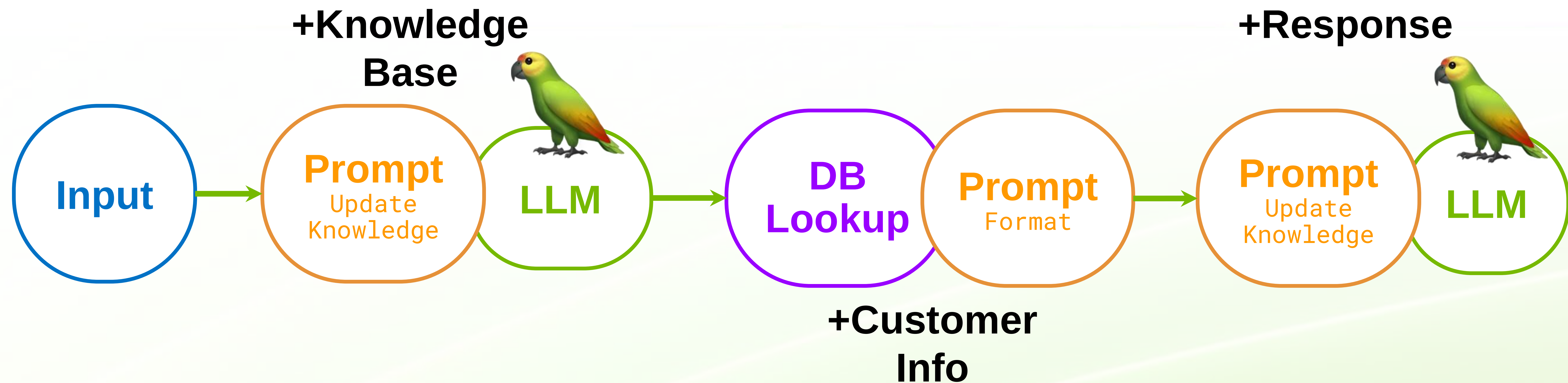


```
{  
  "first_name" : "Jane",  
  "last_name" : "Doe",  
  "confirmation" : -1,  
  ...  
}
```

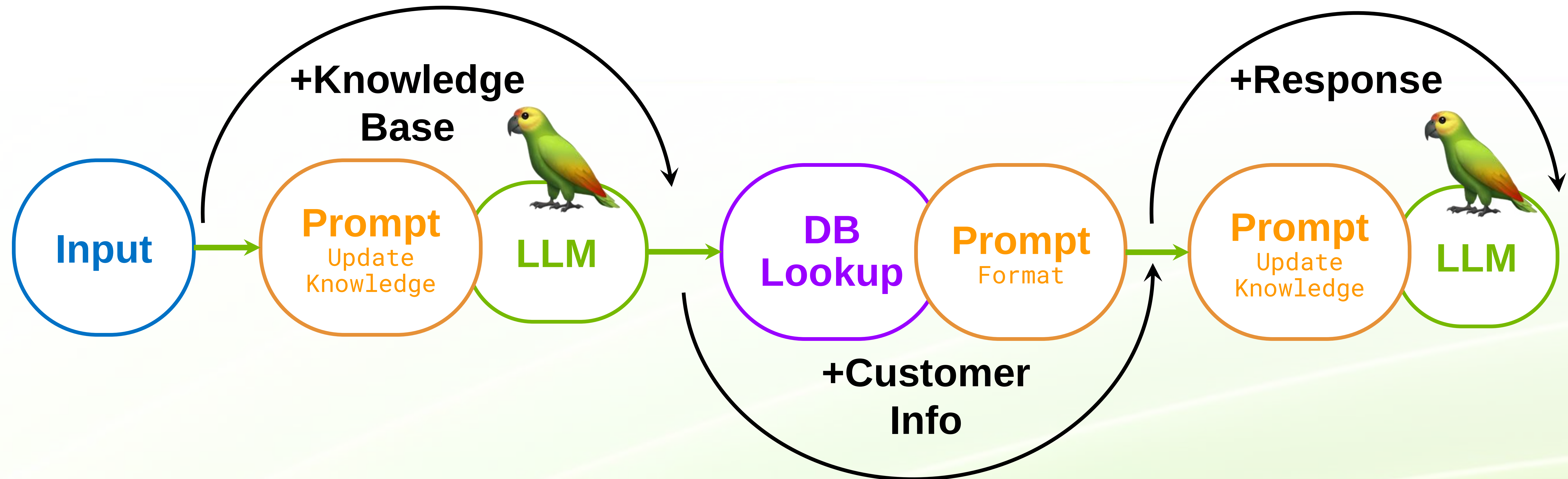
```
## Your goal is to invoke the following through natural conversation
```

```
get_flight_info(  
  "first_name" : "Jane",  
  "last_name" : "Doe",  
  "confirmation" : 12345,  
) -> "Jane Doe's flight from San Jose to New Orleans departs at 12:30 PM tomorrow and lands at 9:30 PM."
```

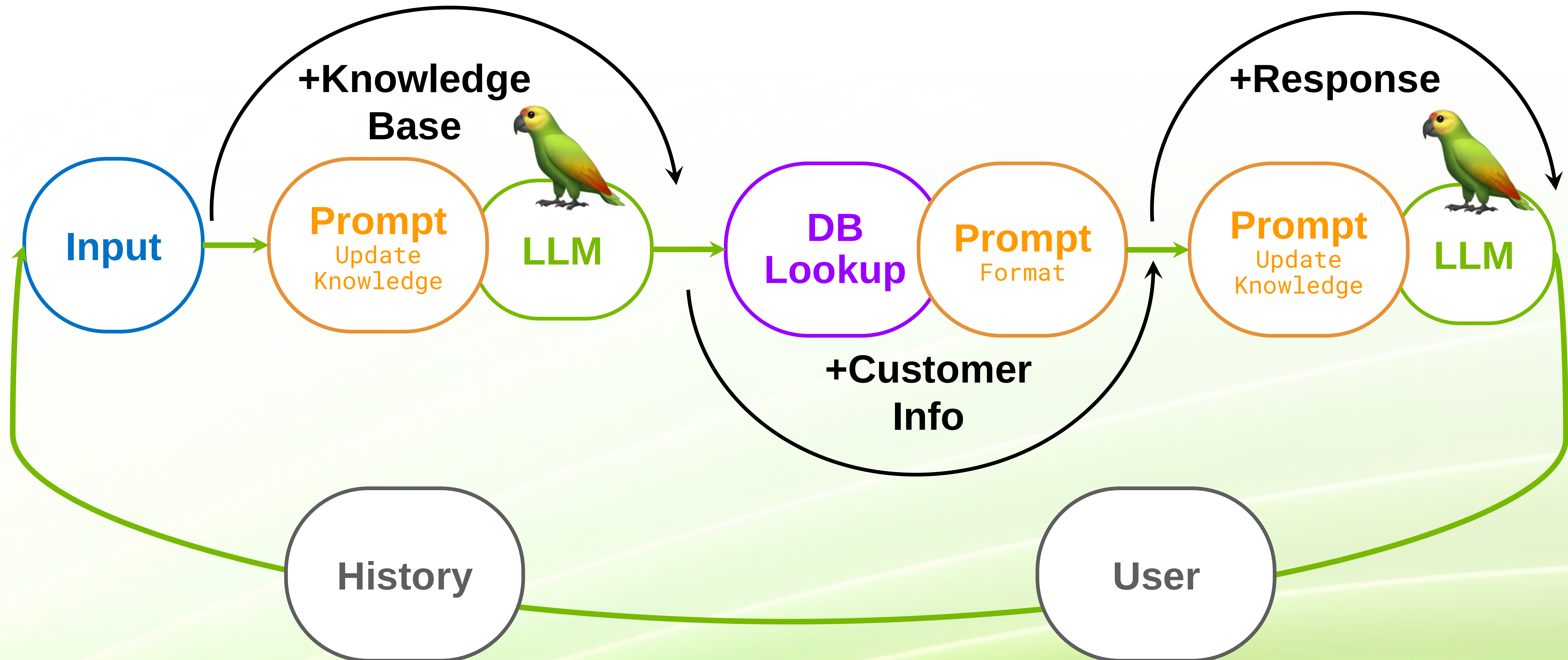
Airline Chatbot



Airline Chatbot



Airline Chatbot



NVIDIA Building RAG Agents with LLMs

Part II

[3, 4]

Building RAG Agents with LLMs

Course Slides

Course Slides

Environment and LLMs

Environment [0, 1, 2]

LangChain

Environment [3, 4]

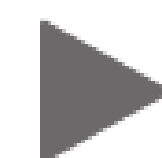


DEEP
LEARNING
INSTITUTE

- chatbot
- composer
- docker_router
- frontend
- imgs
- llm_client
- slides
- solutions
- 00_jupyterlab.ipynb
- 01_microservices.ipynb
- 02_llms.ipynb
- 03_langchain_intro.ipynb
- 04_running_state.ipynb
- 09_langserve.ipynb
- 99_table_of_contents.ipynb

This Lab 0 : 08 : 28 / 2 : 00 : 00

Course 2 : 09 : 22 / 32 : 00 : 00



LAUNCH



STOP TASK

NVIDIA Building RAG Agents with LLMs Part II [5, 6]

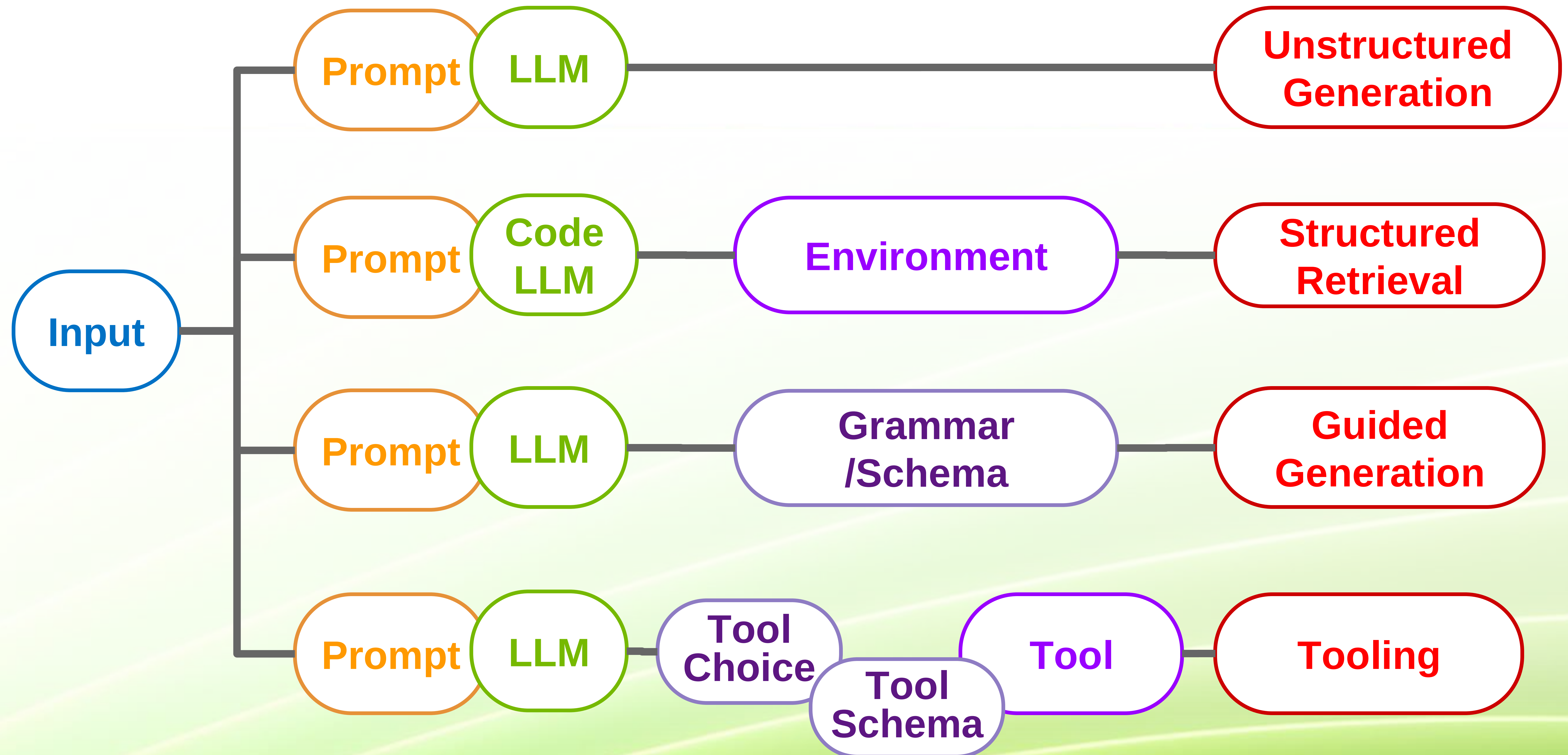


Building RAG Agents with LLMs

Part 5: Working with Documents

Modern Chain Paradigms

Towards Powerful Running State



Document Reasoning

NVIDIA DEVELOPER

Home

Blog

Forums

Docs

Downloads

Training

Technical Blog

Search blog

Filter

Learn the latest advancements in technology and get hands-on training at GTC 2024. March 18-21.

Generative AI / LLMs

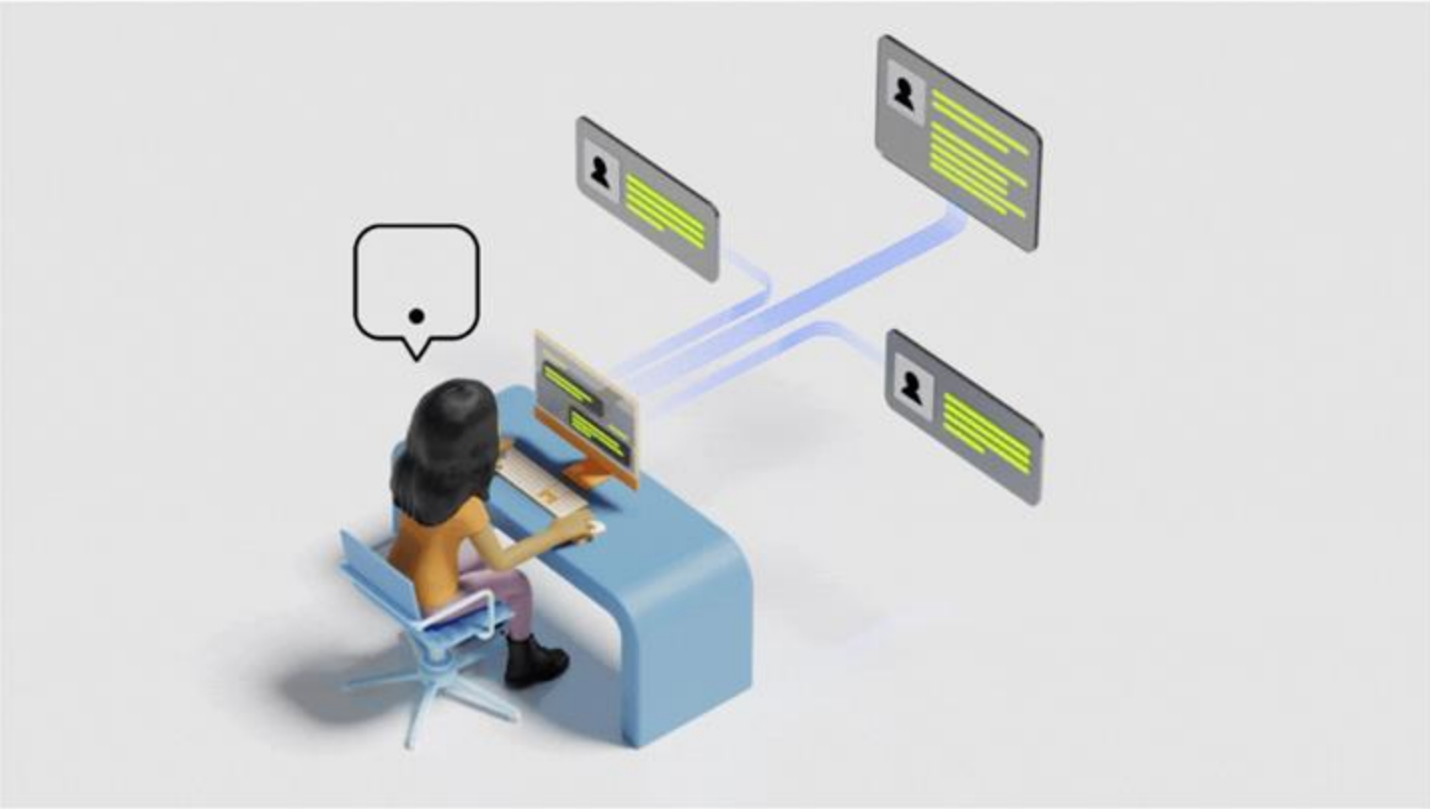
RAG 101: Retrieval-Augmented Generation Questions Answered

Dec 18, 2023

+5 Like

Discuss (1)

By [Hayden Wolff](#), [Daniel Rohrer](#) and [Jiwei Liu](#)



Data scientists, AI engineers, MLOps engineers, and IT infrastructure professionals must consider a variety of factors when designing and deploying a RAG pipeline: from core components like LLM to evaluation approaches.

The key point is that RAG is a system, not just a model or set of models. This system consists of several stages, which were discussed at a high level in [RAG 101: Demystifying Retrieval-Augmented Generation Pipelines](#). All these stages provide opportunities to make design decisions according to your needs.

Here's a list of top questions and answers.

When should you fine-tune the LLM vs. using RAG?

In the world of LLMs, choosing between fine-tuning, Parameter-Efficient Fine-Tuning (PEFT), prompt engineering, and retrieval-augmented generation (RAG) depends on the specific needs and constraints of your application.

- *Fine-tuning* customizes a pretrained LLM for a specific domain by updating most or all of its parameters with a domain-specific dataset. This approach is resource-intensive but yields high accuracy for specialized use cases.
- *PEFT* modifies a pretrained LLM with fewer parameter updates, focusing on a subset of the model. It strikes a balance between accuracy and resource usage, offering improvements over prompt engineering with manageable data and computational demands.
- *Prompt engineering* manipulates the input to an LLM to steer its output, without altering the model's

Generative AI / LLMs

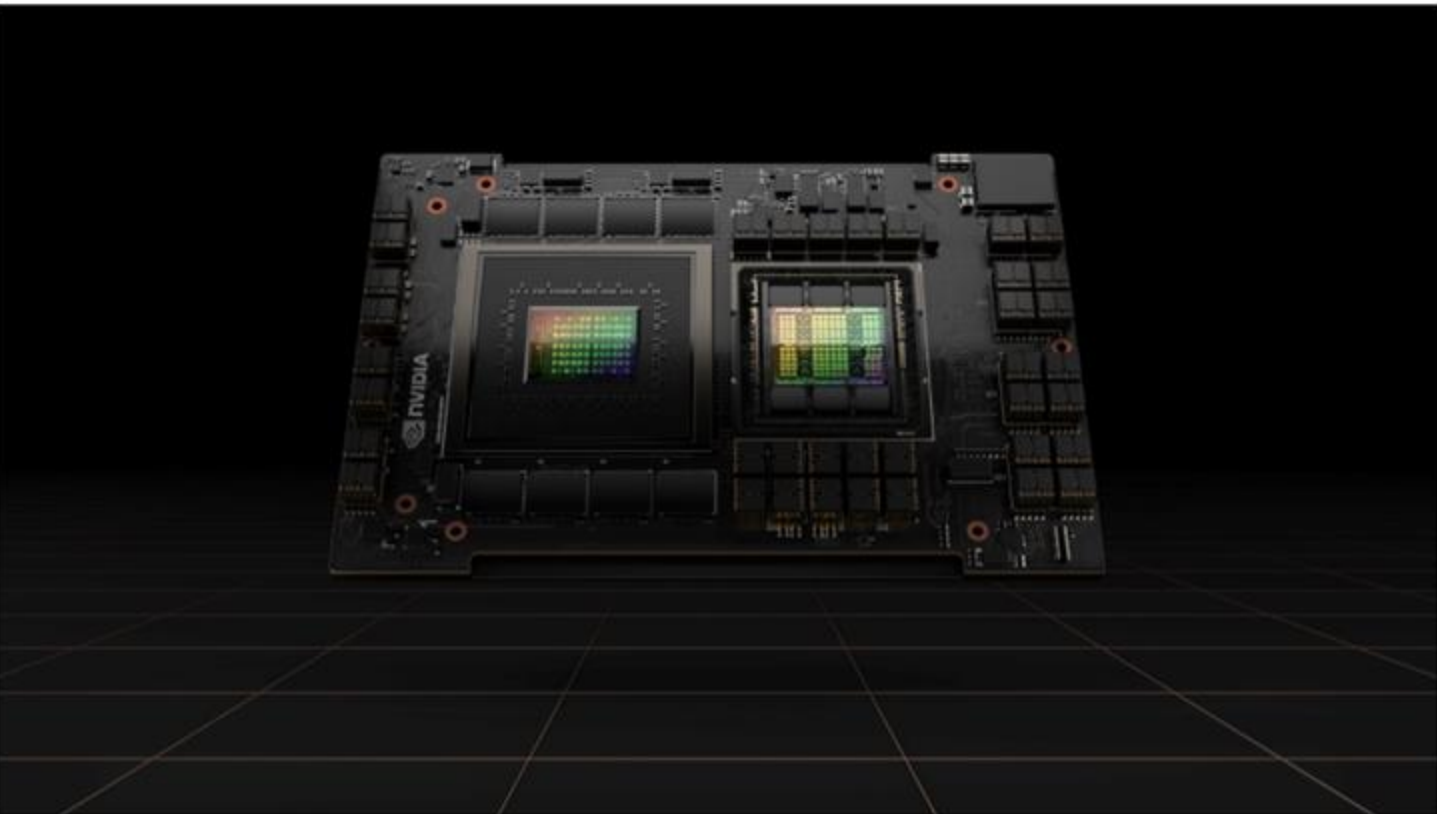
Deploying Retrieval-Augmented Generation Applications on NVIDIA GH200 Delivers Accelerated Performance

Dec 18, 2023

+14 Like

Discuss (2)

By [Rohil Bhargava](#), [Jiwei Liu](#) and [Ivan Goldwasser](#)



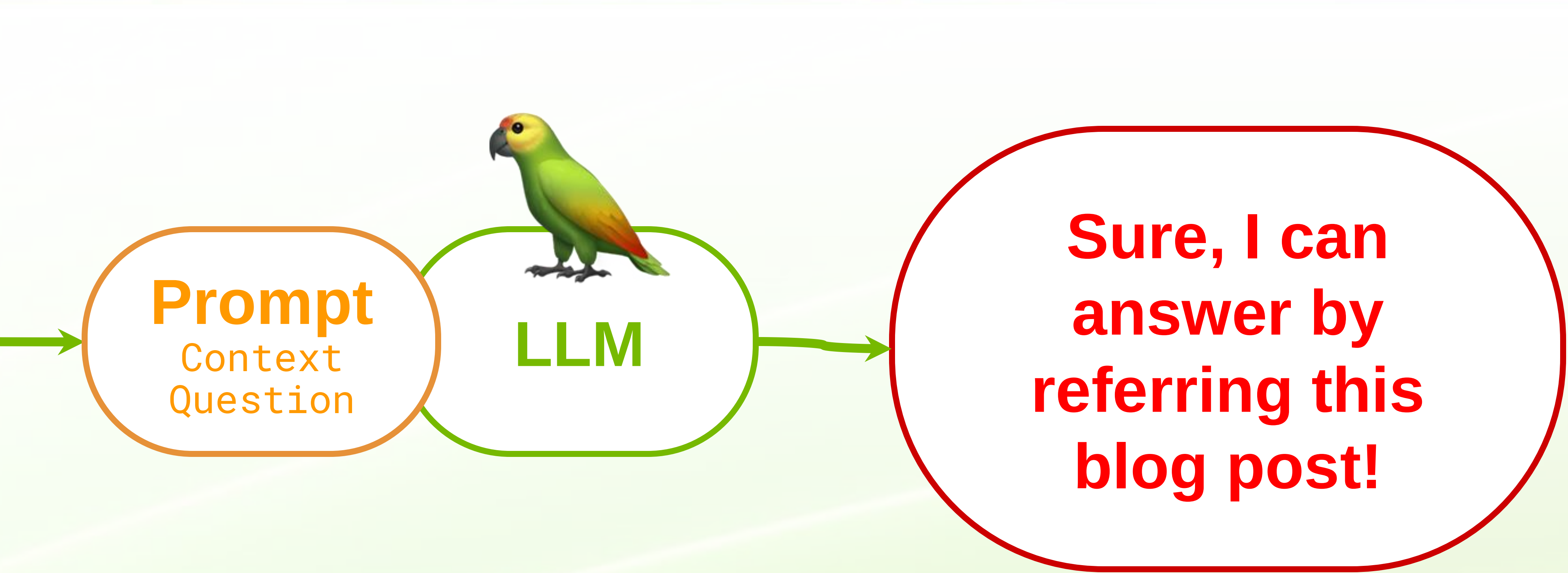
Large language model (LLM) applications are essential in enhancing productivity across industries through natural language. However, their effectiveness is often limited by the extent of their training data, resulting in poor performance when dealing with real-time events and new knowledge the LLM isn't trained on.

Retrieval-augmented generation (RAG) solves these problems. By augmenting models with an external knowledge base, RAG can ground LLMs with relevant data. This enhances the quality of their response, improving accuracy, and reducing hallucinations. RAG not only extends the utility of LLMs but also provides a cheaper alternative to time-consuming re-training runs.

Deploying and scaling RAG applications for tens of thousands to millions of users comes with its own set of challenges, specifically around GPU memory management. Developers need access to state-of-the-art infrastructure with robust memory capabilities that runs real-time RAG applications performantly and within stringent service level agreements (SLAs).

This is where the NVIDIA end-to-end full-stack accelerated computing solution shines. In this post, we discuss how the NVIDIA GH200 Grace Hopper Superchip can help solve these issues.

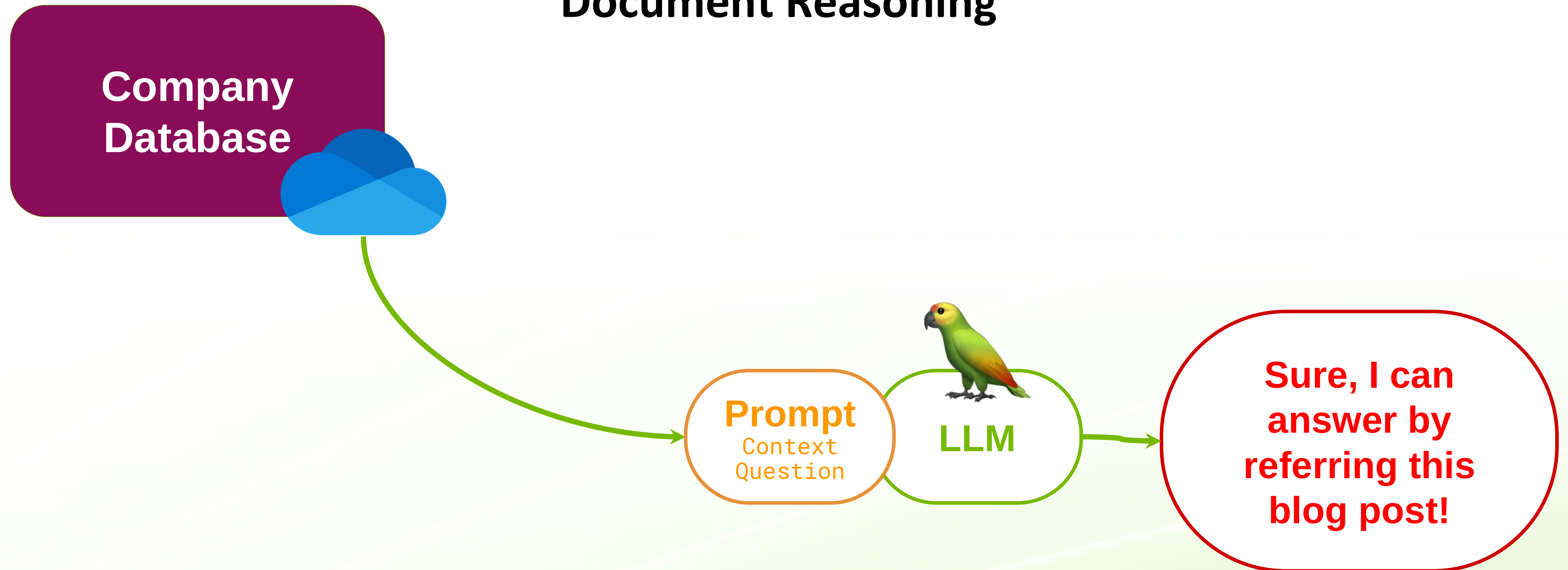
Comparing the GH200 to NVIDIA A100 Tensor Core GPUs, we observed up to a 2.7x increase in speed for embedding generation, 2.9x for index build, 3.3x for vector search time, and 5.7x for Llama-2-70B (FP8).



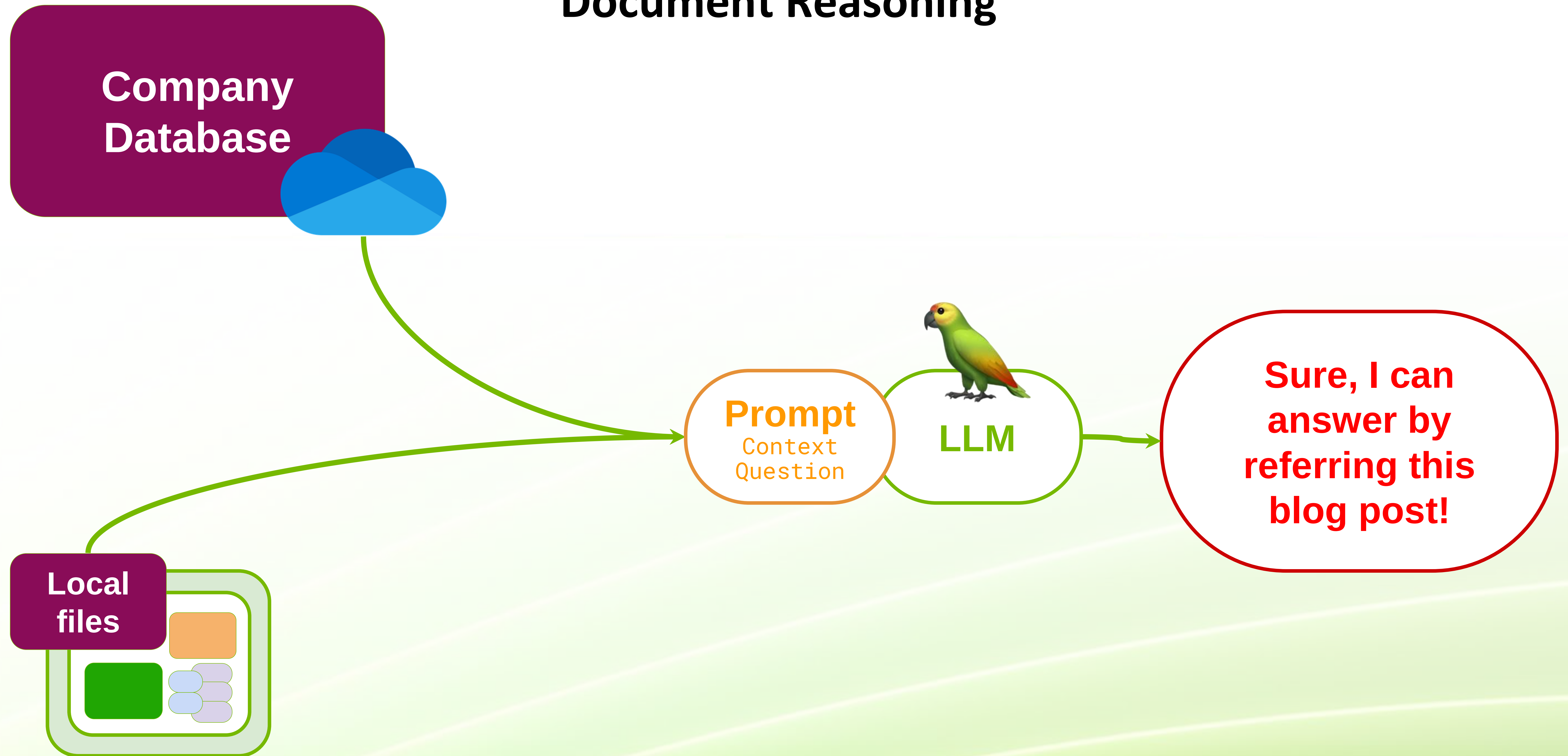
<https://developer.nvidia.com/blog/>

Source: NVIDIA DLI (2024), Building RAG Agents with LLMs, https://learn.nvidia.com/courses/course-detail?course_id=course-v1:DLI+S-FX-15+V1

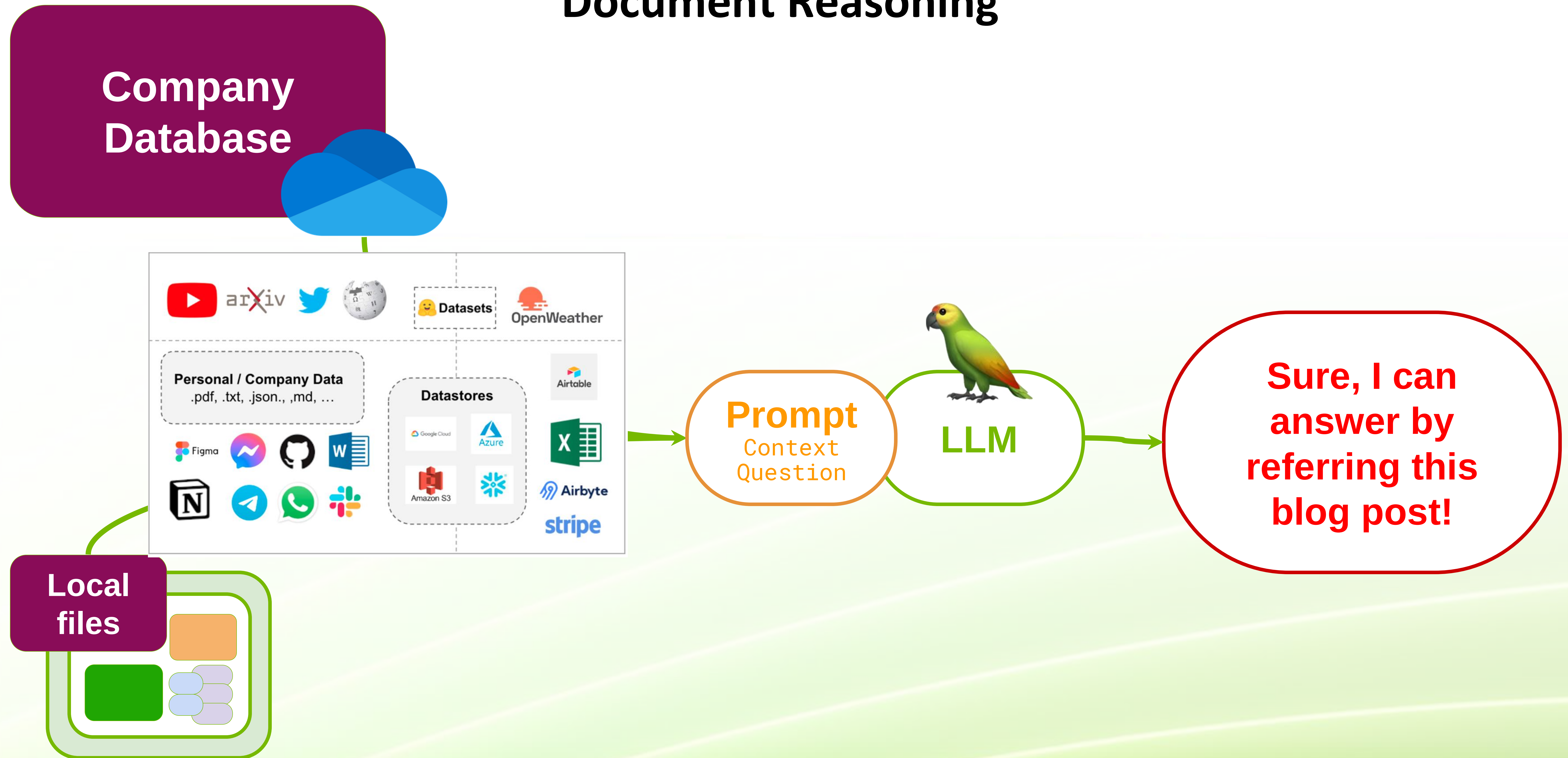
Document Reasoning



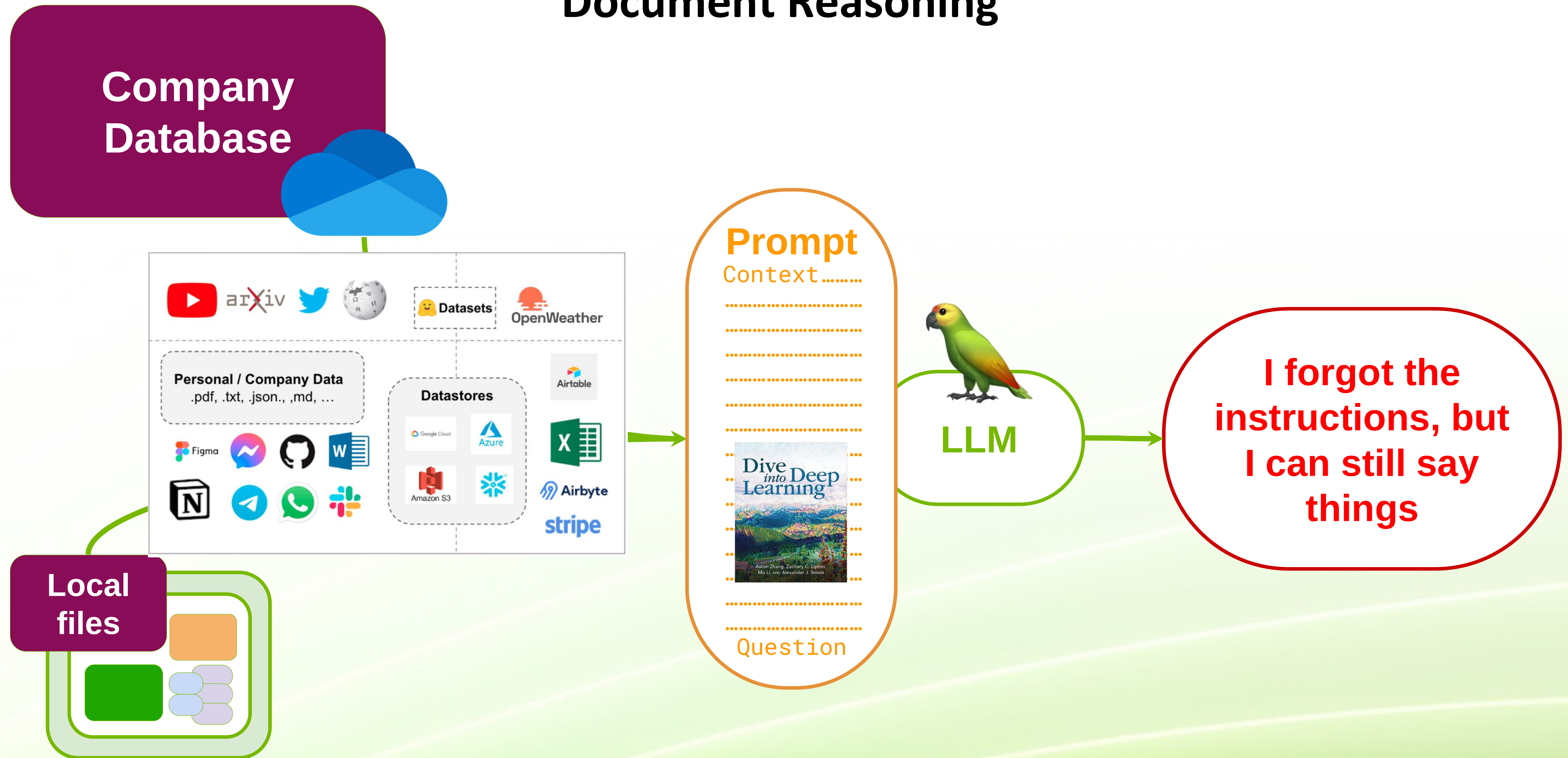
Document Reasoning



Document Reasoning



Document Reasoning



Chunking

arXiv:2307.09288v2 [cs.CL] 19 Jul 2023

LLAMA 2: Open Foundation and Fine-Tuned Chat Models

Hugo Touvron* Louis Martin† Kevin Stone†
Peter Albert Amjad Almahairi Yasmine Babaei Nikolay Bashlykov Soumya Batra
Prajwal Bhargava Shruti Bhosale Dan Bikel Lukas Blecher Cristian Canton Ferrer Moya Chen
Guillem Cucurull David Esiobu Jude Fernandes Jeremy Fu Wenyin Fu Brian Fuller
Cynthia Gao Vedanuj Goswami Naman Goyal Anthony Hartshorn Saghar Hosseini Rui Hou
Hakan Inan Marcin Kardas Viktor Kerkez Madian Khabsa Isabel Kloumann Artem Korenev
Punit Singh Koura Marie-Anne Lachaux Thibaut Lavril Jenya Lee Diana Liskovich
Yinghai Lu Yuning Mao Xavier Martinet Todor Mihaylov Pushkar Mishra
Igor Molybog Yixin Nie Andrew Poulton Jeremy Reizenstein Rashi Rungta Kalyan Saladi
Alan Schelten Ruan Silva Eric Michael Smith Ranjan Subramanian Xiaoqing Ellen Tan Binh Tang
Ross Taylor Adina Williams Jian Xiang Kuan Puxin Xu Zheng Yan Iliyan Zarov Yuchen Zhang
Angela Fan Melanie Kambadur Sharan Narang Aurelien Rodriguez Robert Stojnic
Sergey Edunov Thomas Scialom*

GenAI, Meta

Abstract

In this work, we develop and release Llama 2, a collection of pretrained and fine-tuned large language models (LLMs) ranging in scale from 7 billion to 70 billion parameters. Our fine-tuned LLMs, called LLAMA 2-CHAT, are optimized for dialogue use cases. Our models outperform open-source chat models on most benchmarks we tested, and based on our human evaluations for helpfulness and safety, may be a suitable substitute for closed-source models. We provide a detailed description of our approach to fine-tuning and safety improvements of LLAMA 2-CHAT in order to enable the community to build on our work and contribute to the responsible development of LLMs.

*Equal contribution, corresponding authors: {tscialom, htouvron}@meta.com
†Second author
Contributions for all the authors can be found in Section A.1.

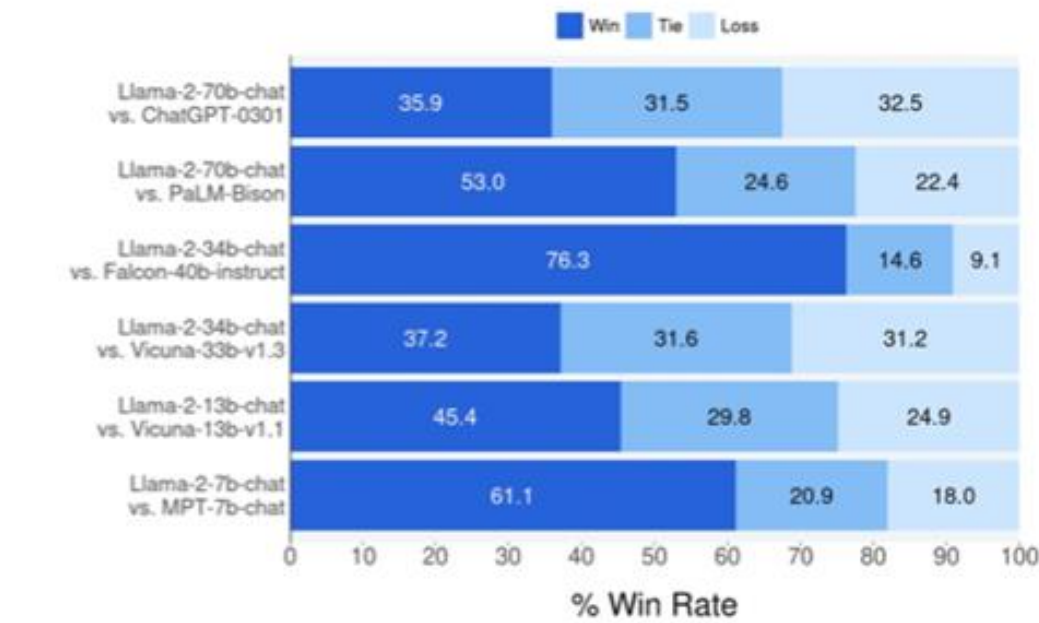


Figure 1: Helpfulness human evaluation results for LLAMA 2-CHAT compared to other open-source and closed-source models. Human raters compared model generations on ~4k prompts consisting of both single and multi-turn prompts. The 95% confidence intervals for this evaluation are between 1% and 2%. More details in Section 3.4.2. While reviewing these results, it is important to note that human evaluations can be noisy due to limitations of the prompt set, subjectivity of the review guidelines, subjectivity of individual raters, and the inherent difficulty of comparing generations.

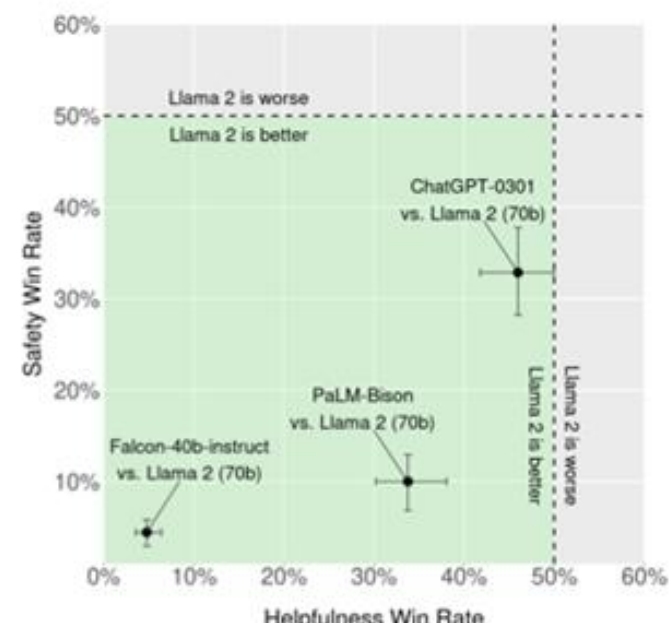


Figure 2: Win-rate % for helpfulness and safety between commercial-licensed base-lines and LLAMA 2-CHAT, according to GPT-4. To complement the human evaluation, we used a more capable model, not subject to our own guidance. Green area indicates our model is better according to GPT-4. To remove ties, we used $win/(win + loss)$. The orders in which the model responses are presented to GPT-4 are randomly swapped to alleviate bias.

1 Introduction

Large Language Models (LLMs) have shown great promise as highly capable AI assistants that excel in complex reasoning tasks requiring expert knowledge across a wide range of fields, including in specialized domains such as programming and creative writing. They enable interaction with humans through intuitive chat interfaces, which has led to rapid and widespread adoption among the general public.

The capabilities of LLMs are remarkable considering the seemingly straightforward nature of the training methodology. Auto-regressive transformers are pretrained on an extensive corpus of self-supervised data, followed by alignment with human preferences via techniques such as Reinforcement Learning with Human Feedback (RLHF). Although the training methodology is simple, high computational requirements have limited the development of LLMs to a few players. There have been public releases of pretrained LLMs (such as BLOOM (Scao et al., 2022), LLaMA-1 (Touvron et al., 2023), and Falcon (Penedo et al., 2023)) that match the performance of closed pretrained competitors like GPT-3 (Brown et al., 2020) and GPT-3.5 (Hoffmann et al., 2022), but none of these models are suitable substitutes for closed “product” LLMs, such as ChatGPT, BARD, and Claude. These closed product LLMs are heavily fine-tuned to align with human preferences, which greatly enhances their usability and safety. This step can require significant costs in compute and human annotation, and is often not transparent or easily reproducible, limiting progress within the community to advance AI alignment research.

In this work, we develop and release Llama 2, a family of pretrained and fine-tuned LLMs, LLAMA 2 and LLAMA 2-CHAT, at scales up to 70B parameters. On the series of helpfulness and safety benchmarks we tested, LLAMA 2-CHAT models generally perform better than existing open-source models. They also appear to be on par with some of the closed-source models, at least on the human evaluations we performed (see Figures 1 and 3). We have taken measures to increase the safety of these models, using safety-specific data annotation and tuning, as well as conducting red-teaming and employing iterative evaluations. Additionally, this paper contributes a thorough description of our fine-tuning methodology and approach to improving LLM safety. We hope that this openness will enable the community to reproduce fine-tuned LLMs and continue to improve the safety of those models, paving the way for more responsible development of LLMs. We also share novel observations we made during the development of LLAMA 2 and LLAMA 2-CHAT, such as the emergence of tool usage and temporal organization of knowledge.

Document Stuffing

LLAMA 2: Open Foundation and Fine-Tuned Chat Models

Hugo Touvron* Louis Martin† Kevin Stone†
Peter Albert Amjad Almahairi Yasmine Babaei Nikolay Bashlykov Soumya Batra
Prajwal Bhargava Shruti Bhosale Dan Bikel Lukas Blecher Cristian Canton Ferrer Moya Chen
Guillem Cucurull David Esiobu Jude Fernandes Jeremy Fu Wenyin Fu Brian Fuller
Cynthia Gao Vedanuj Goswami Naman Goyal Anthony Hartshorn Saghar Hosseini Rui Hou
Hakan Inan Marcin Kardas Viktor Kerkez Madian Khabsa Isabel Kloumann Artem Korenev
Punit Singh Koura Marie-Anne Lachaux Thibaut Lavril Jenya Lee Diana Liskovich
Yinghai Lu Yuning Mao Xavier Martinet Todor Mihaylov Pushkar Mishra
Igor Molybog Yixin Nie Andrew Poulton Jeremy Reizenstein Rashi Rungta Kalyan Saladi
Alan Schelten Ruan Silva Eric Michael Smith Ranjan Subramanian Xiaoqing Ellen Tan Binh Tang
Ross Taylor Adina Williams Jian Xiang Kuan Puxin Xu Zheng Yan Iliyan Zarov Yuchen Zhang
Angela Fan Melanie Kambadur Sharan Narang Aurore Rodriguez Robert Stojanovic

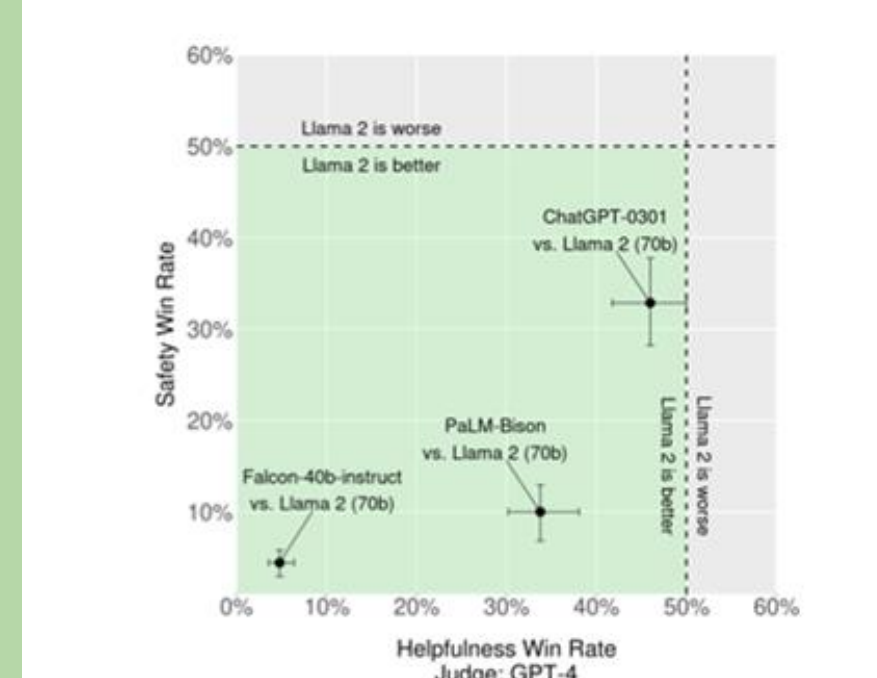


Figure 2: Win-rate % for helpfulness and safety between commercial-licensed base models and LLaMA 2 (70B) based on GPT-4 evaluation. We show the win rate for helpfulness and safety across various models. The orders in the chart are presented to alleviate bias.

1 Introduction

Large Language Models (LLMs) have shown great promise as complex reasoning tasks requiring expert knowledge across a wide range of domains such as programming and creative writing. They enable chat interfaces, which has led to rapid and widespread adoption.

The capabilities of LLMs are remarkable considering the seemingly straightforward nature of the training methodology. Auto-regressive transformers are pretrained on an extensive corpus of self-supervised data, followed by alignment with human preferences via techniques such as Reinforcement Learning with Human Feedback (RLHF). Although the training methodology is simple, high computational requirements have limited the development of LLMs to a few players. There have been public releases of pretrained LLMs (such as BLOOM (Scao et al., 2022), LLaMa-1 (Touvron et al., 2023), and Falcon (Penedo et al., 2023)) that match the performance of closed pretrained competitors like GPT-3 (Brown et al., 2020) and Chinchilla (Hoffmann et al., 2022), but none of these models are suitable substitutes for closed “product” LLMs, such as ChatGPT, BARD, and Claude. These closed product LLMs are heavily fine-tuned to align with human preferences, which greatly enhances their usability and safety. This step can require significant costs to compute and human annotation, and is often not transparent or easily reproducible, limiting progress with the community to advance AI alignment research.

Abstract

In this work, we develop and release Llama 2, a collection of pretrained and fine-tuned large language models (LLMs) ranging in scale from 7 billion to 70 billion parameters. Our fine-tuned LLMs, called LLaMA 2-CHAT, are optimized for dialogue use cases. Our models outperform open-source chat models on most benchmarks we tested, and based on our human evaluations for helpfulness and safety, may be a suitable substitute for closed-source models. We provide a detailed description of our approach to fine-tuning and safety improvements of LLaMA 2-CHAT in order to enable the community to build on our work and contribute to the responsible development of LLMs.

The capabilities of LLMs are remarkable considering the seemingly straightforward nature of the training methodology. Auto-regressive transformers are pretrained on an extensive corpus of self-supervised data, followed by alignment with human preferences via techniques such as Reinforcement Learning with Human Feedback (RLHF). Although the training methodology is simple, high computational requirements have limited the development of LLMs to a few players. There have been public releases of pretrained LLMs (such as BLOOM (Scao et al., 2022), LLaMa-1 (Touvron et al., 2023), and Falcon (Penedo et al., 2023)) that match the performance of closed pretrained competitors like GPT-3 (Brown et al., 2020) and Chinchilla (Hoffmann et al., 2022), but none of these models are suitable substitutes for closed “product” LLMs, such as ChatGPT, BARD, and Claude. These closed product LLMs are heavily fine-tuned to align with human preferences, which greatly enhances their usability and safety. This step can require significant costs to compute and human annotation, and is often not transparent or easily reproducible, limiting progress with the community to advance AI alignment research.

In this work, we develop and release Llama 2, a family of pretrained and fine-tuned LLMs, LLaMA 2 and LLaMA 2-CHAT, at scales up to 70B parameters. On the series of helpfulness and safety benchmarks we tested, LLaMA 2-CHAT models generally perform better than existing open-source models. They also appear to be on par with some of the closed-source models, at least on the human evaluations we performed (see Figures 1 and 3). We have taken measures to increase the safety of these models, using safety-specific data annotation and tuning, as well as conducting red-teaming and employing iterative evaluations. Additionally, this paper contributes a thorough description of our fine-tuning methodology and approach to improving the safety of LLaMA 2-CHAT.



Figure 1: Helpfulness human evaluation results for LLaMA 2-CHAT compared to other open-source and closed-source models. Human raters compared model generations on ~4k prompts consisting of both single and multi-turn prompts. The 95% confidence intervals for this evaluation are between 1% and 2%. More details in Section 3.4.2. While reviewing these results, it is important to note that human evaluations can be noisy due to limitations of the prompt set, subjectivity of the review guidelines, subjectivity of individual raters, and the inherent difficulty of comparing generations.

Prompt

(DOCS 1-4)

Use Them
Please

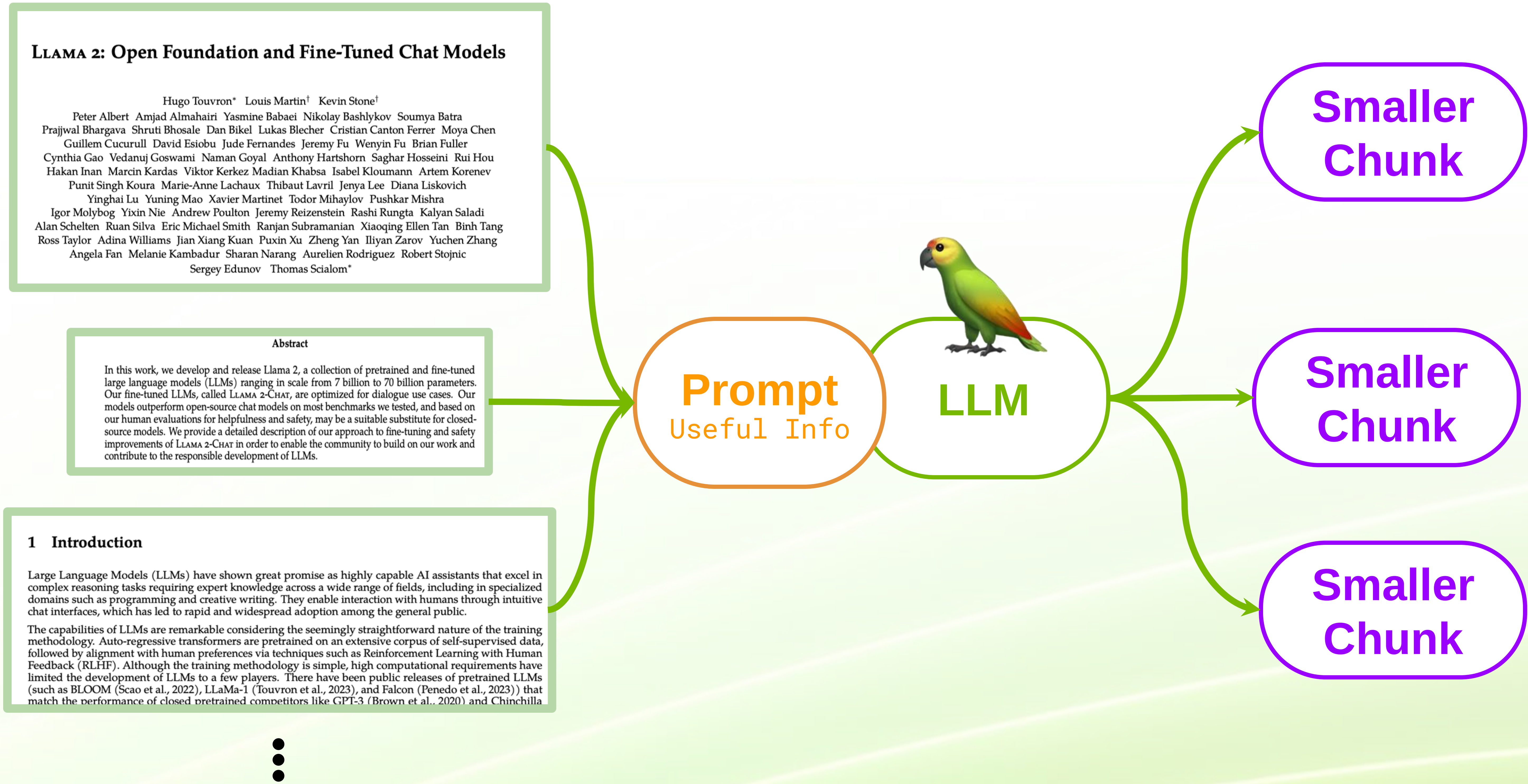
Question



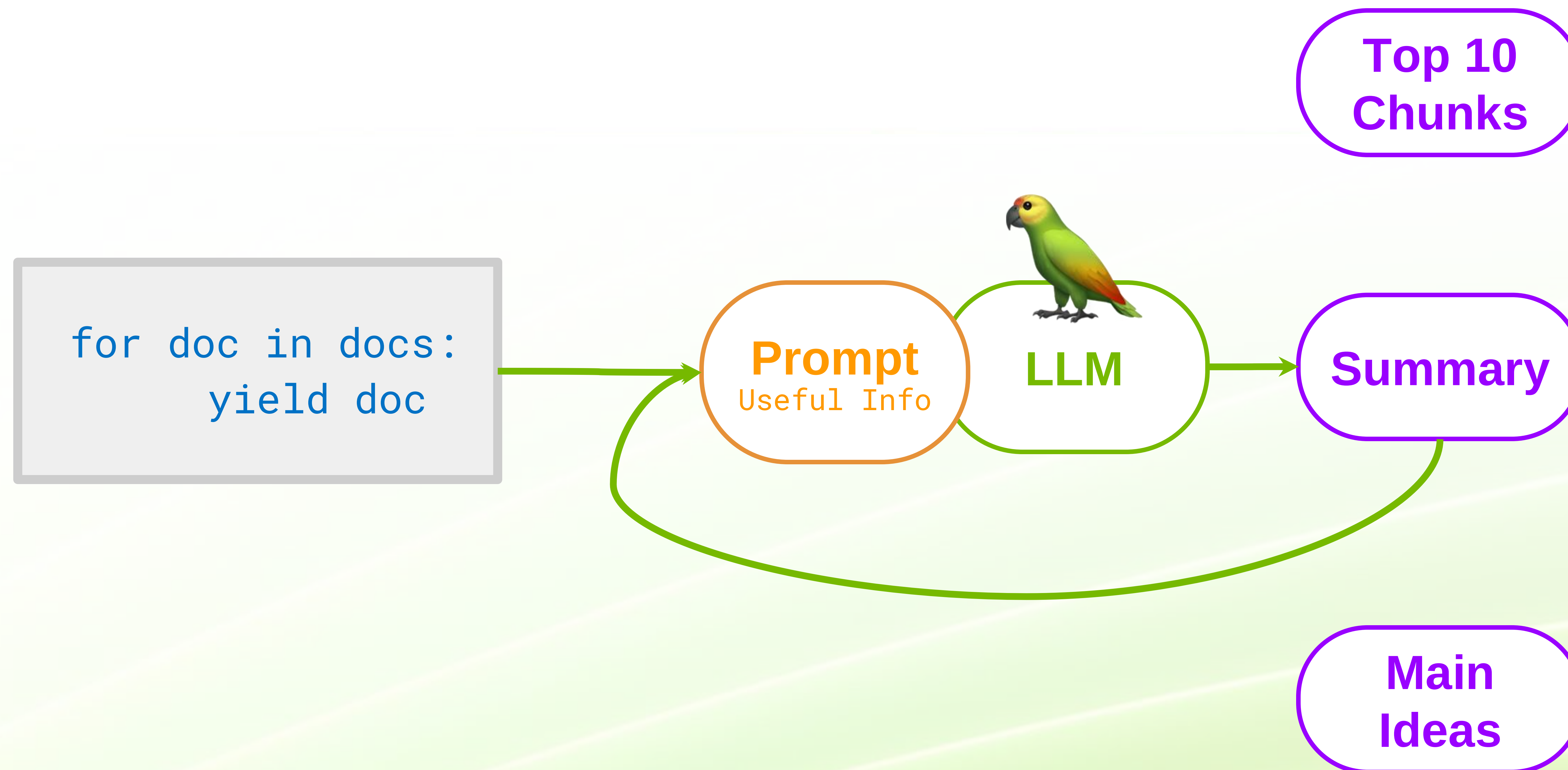
LLM

Oh yes, the intro
tells me
everything! You
see...

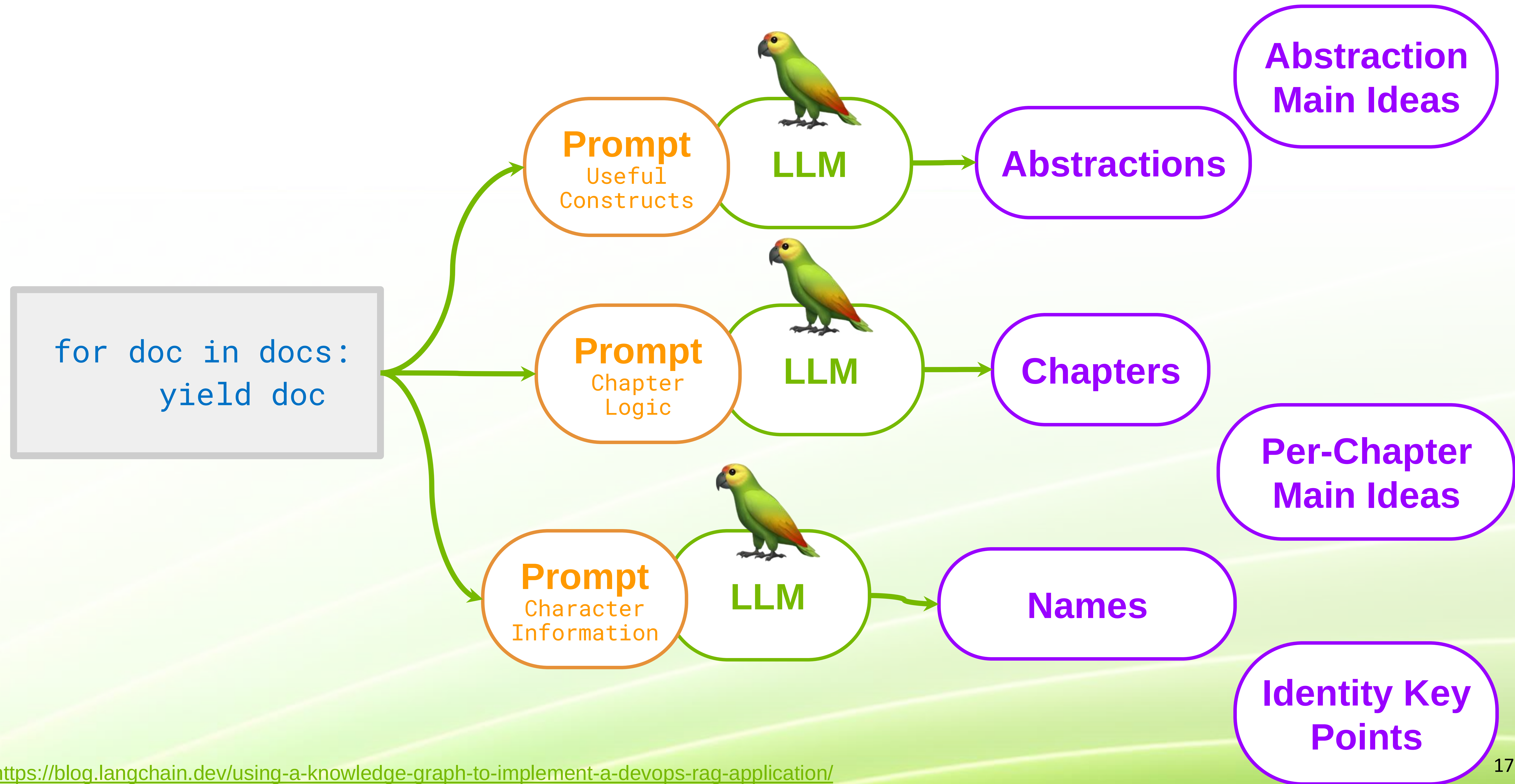
Map Reduce Chain



Refinement Chain



Knowledge Graph Construction

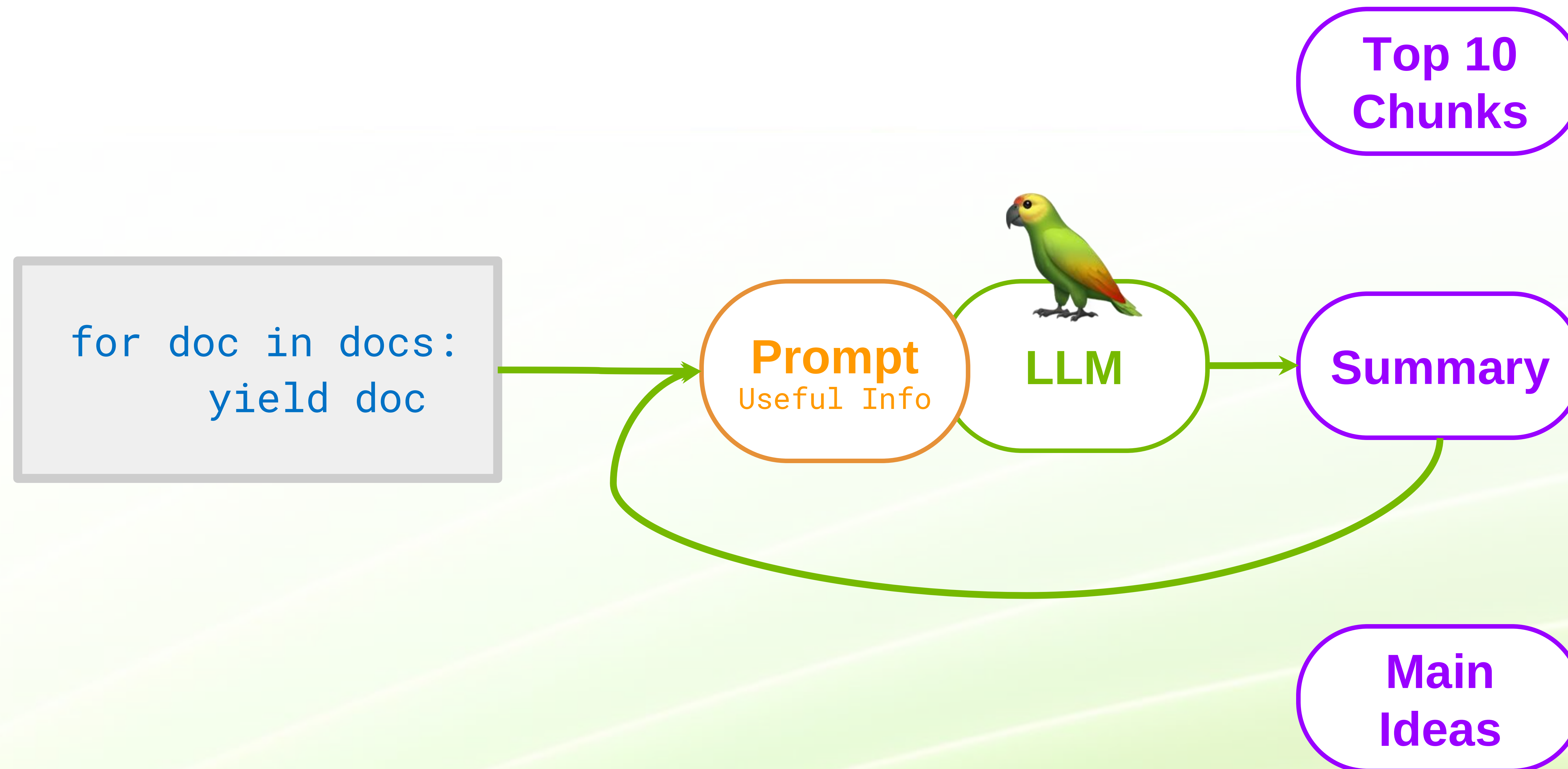


Knowledge Graph Traversal



Refinement Chain

Your Assignment



Refinement Chain

Your Assignment

RunnableLambda

```
for doc in docs:  
    yield doc
```

Prompt
Useful Info



LLM

Summary

Optional Tangent: LangGraph

```
from langchain_openai import ChatOpenAI
from langchain_core.messages import HumanMessage
from langgraph.graph import END, MessageGraph
```

```
model = ChatOpenAI(temperature=0)
```

```
graph = MessageGraph()
```

```
graph.add_node("oracle", model)
```

```
graph.add_edge("oracle", END)
```

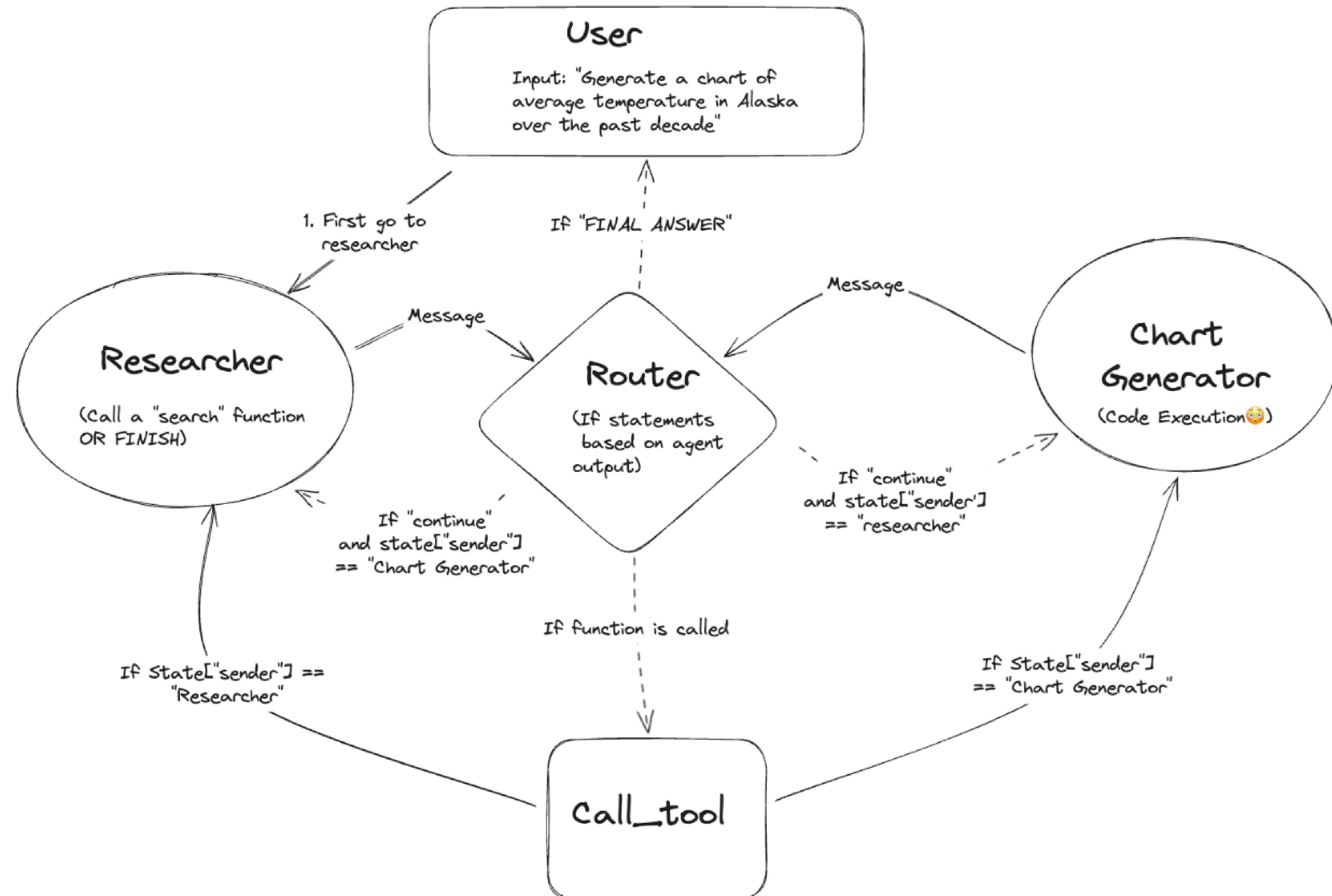
```
graph.set_entry_point("oracle")
```

```
runnable = graph.compile()
```

Let's run it!

```
runnable.invoke(HumanMessage("What is 1 + 1?"))
```

```
[HumanMessage(content='What is 1 + 1?'), AIMessage(content='1 + 1 equals 2.')]
```



<https://python.langchain.com/docs/langgraph/>
<https://blog.langchain.dev/langgraph-multi-agent-workflow/>

Refinement Chain

Your Assignment

RunnableLambda

```
for doc in docs:  
    yield doc
```

Prompt
Useful Info



LLM

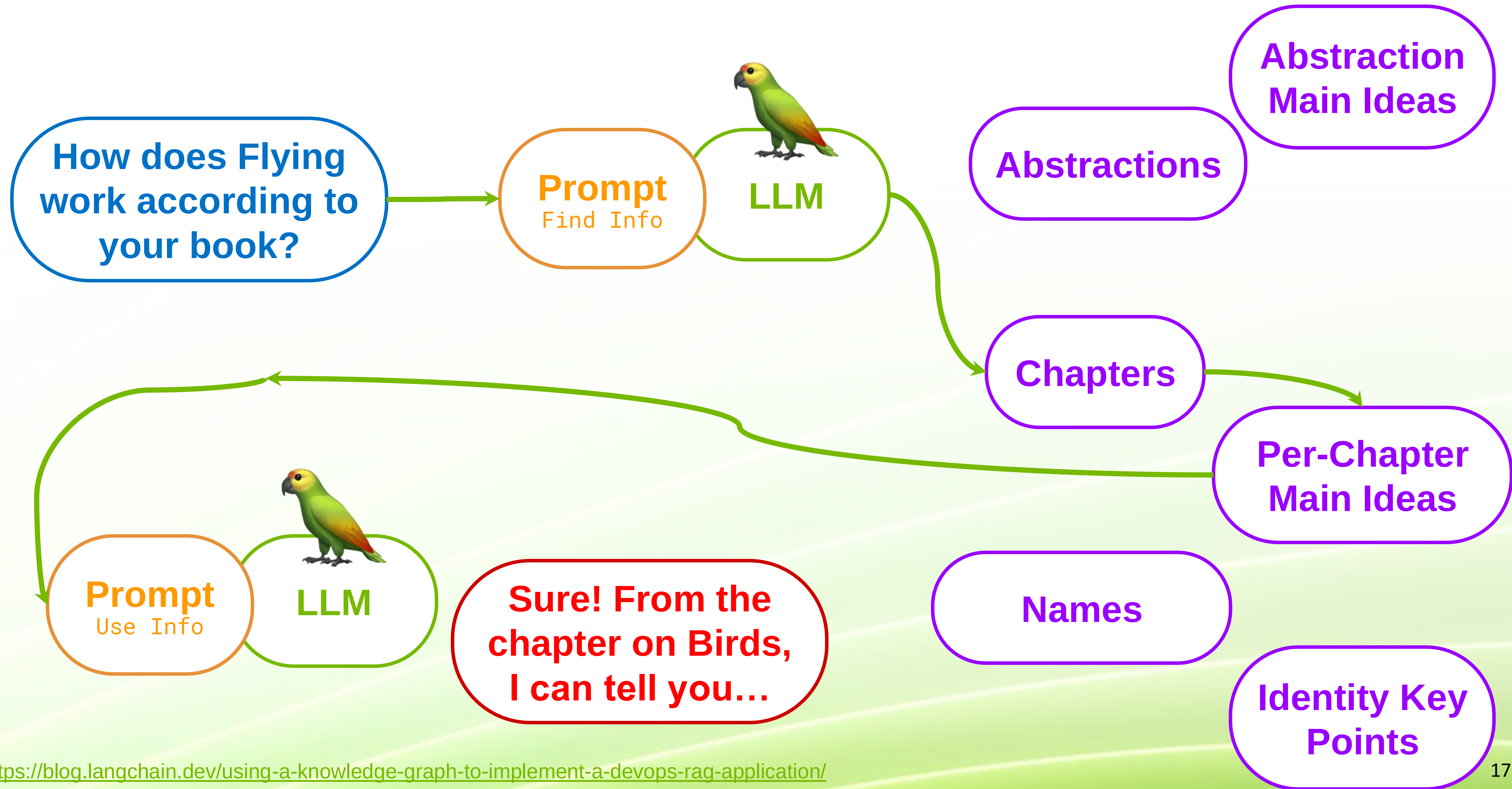
Summary



Building RAG Agents with LLMs

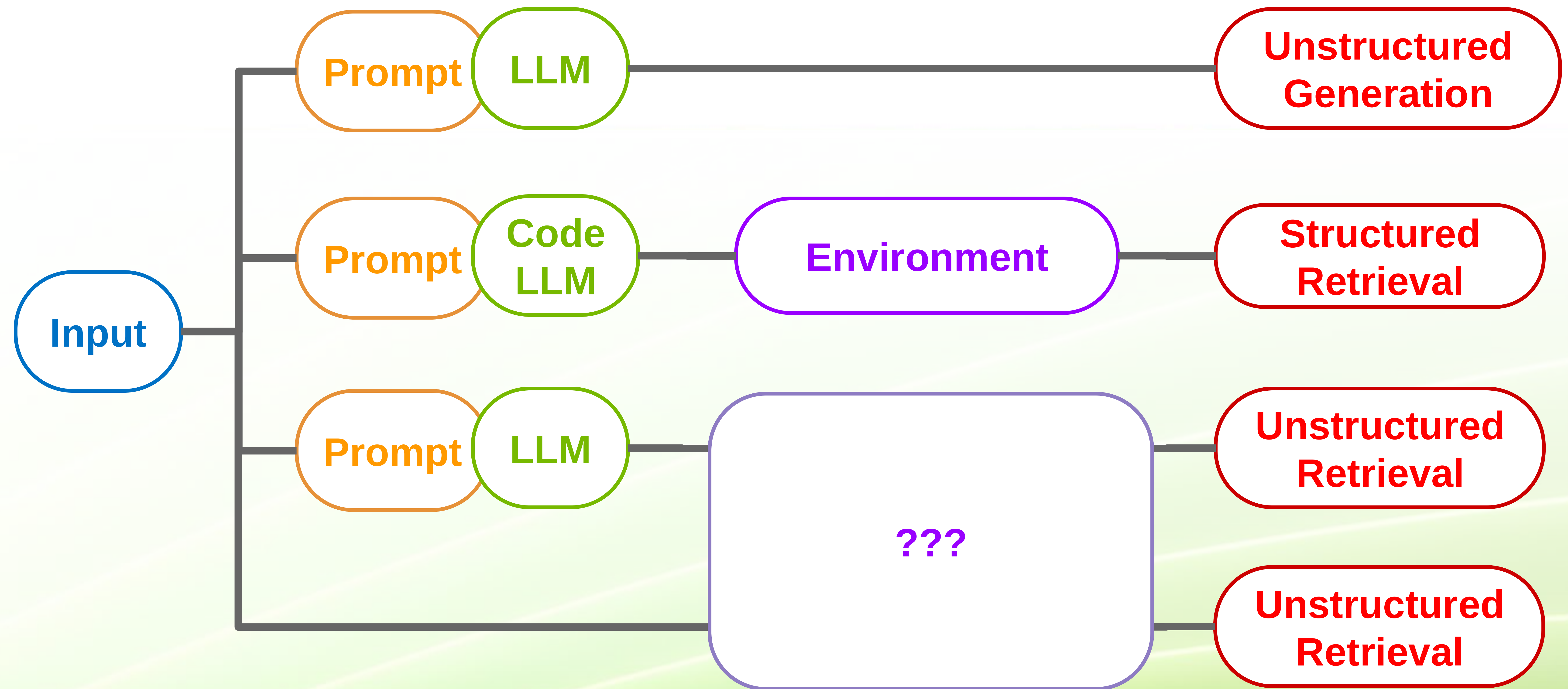
Part 6: Embedding Model for Retrieval

Knowledge Graph Traversal



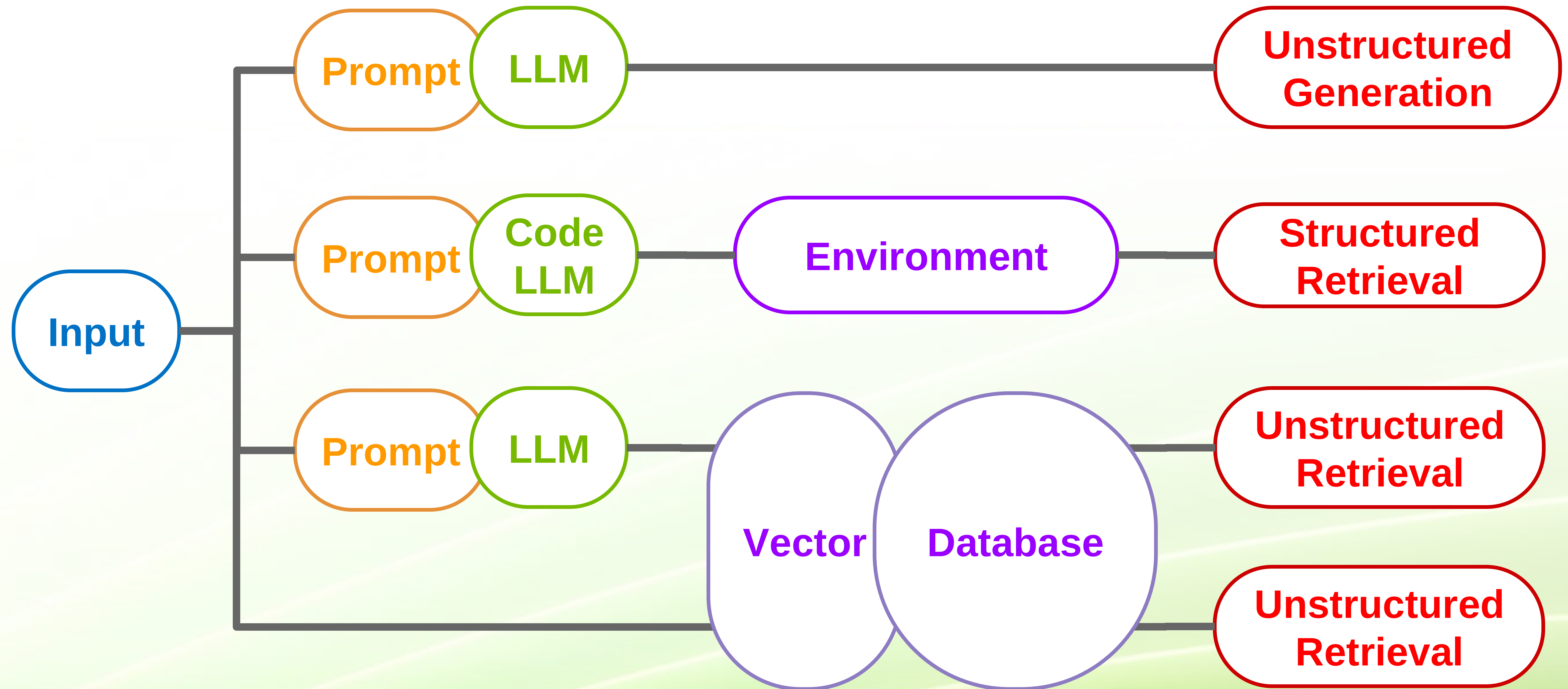
Modern Chain Paradigms

Towards Powerful Running State



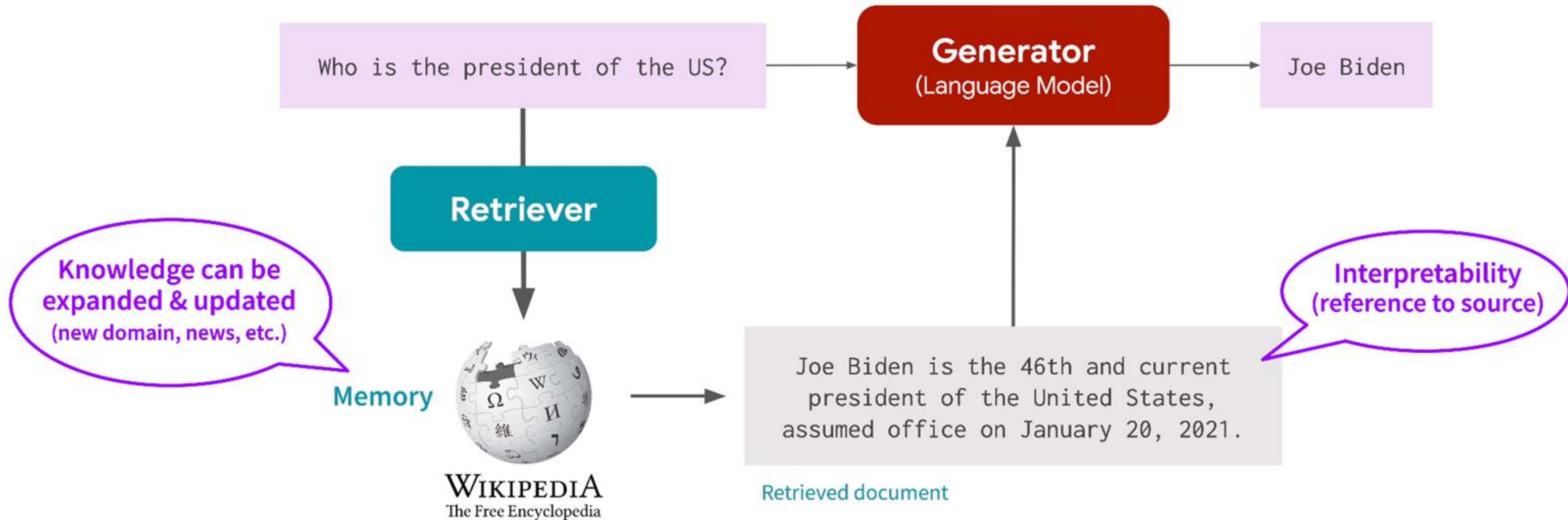
Modern Chain Paradigms

Towards Powerful Running State



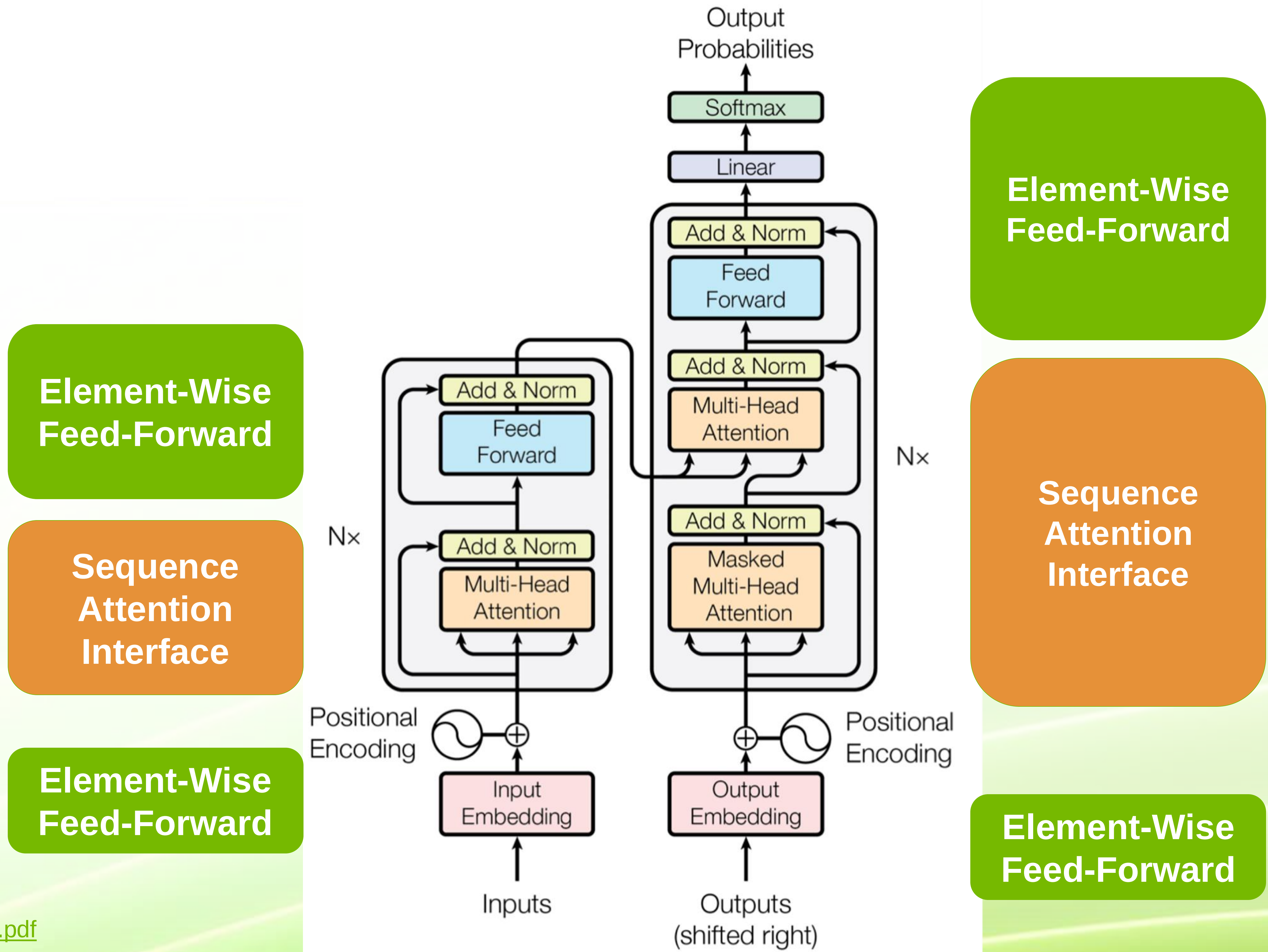
Retrieval-Augmented Generation

Pulling in Information from a Database



Transformer Architecture

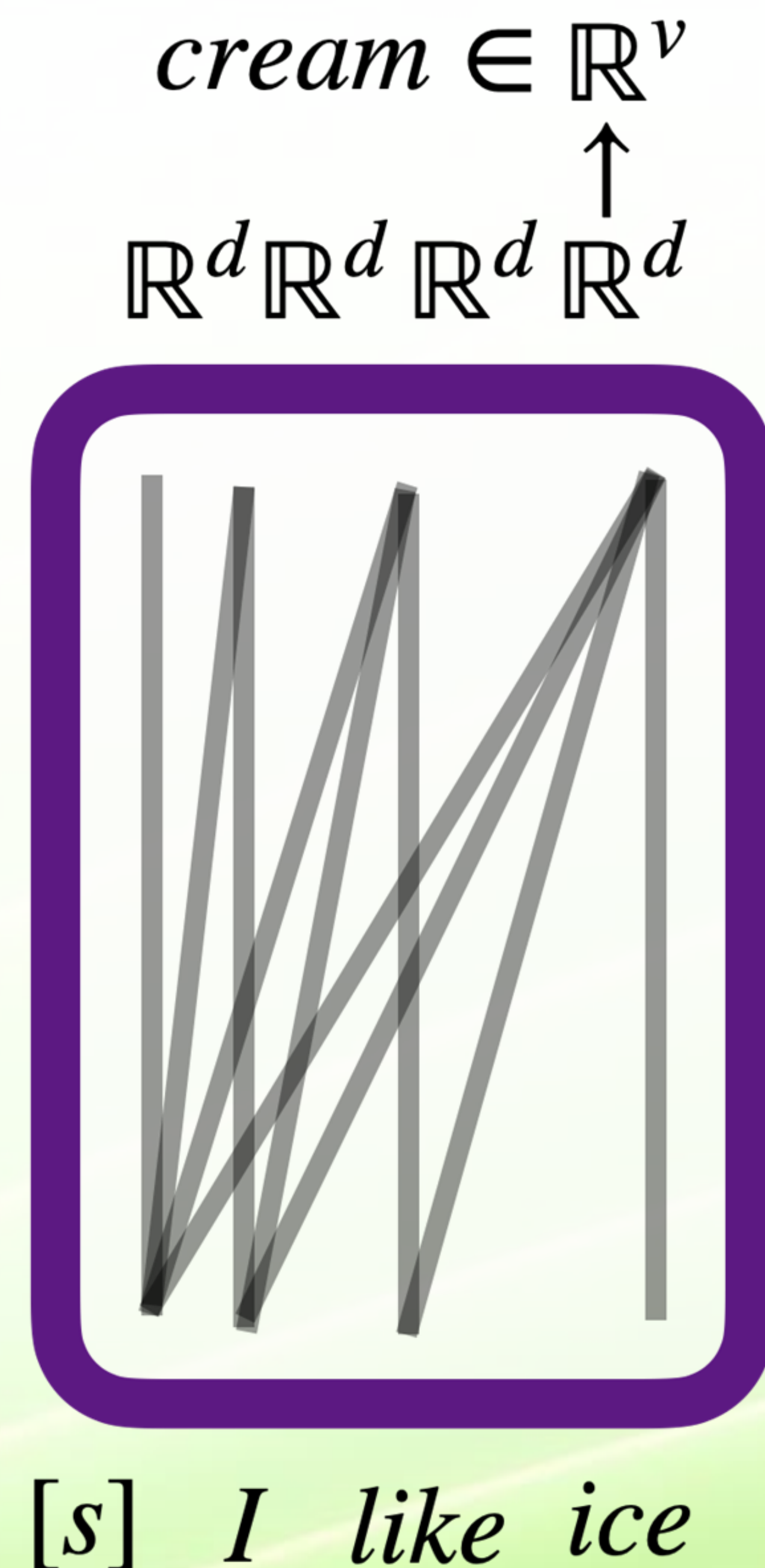
Primary Backbone of LLMs



Transformer Architecture

Autoregressing vs Embedding Flavors

Decoder For
Autoregressive Forecasting
(Common, Parallelizable)



Transformer Architecture

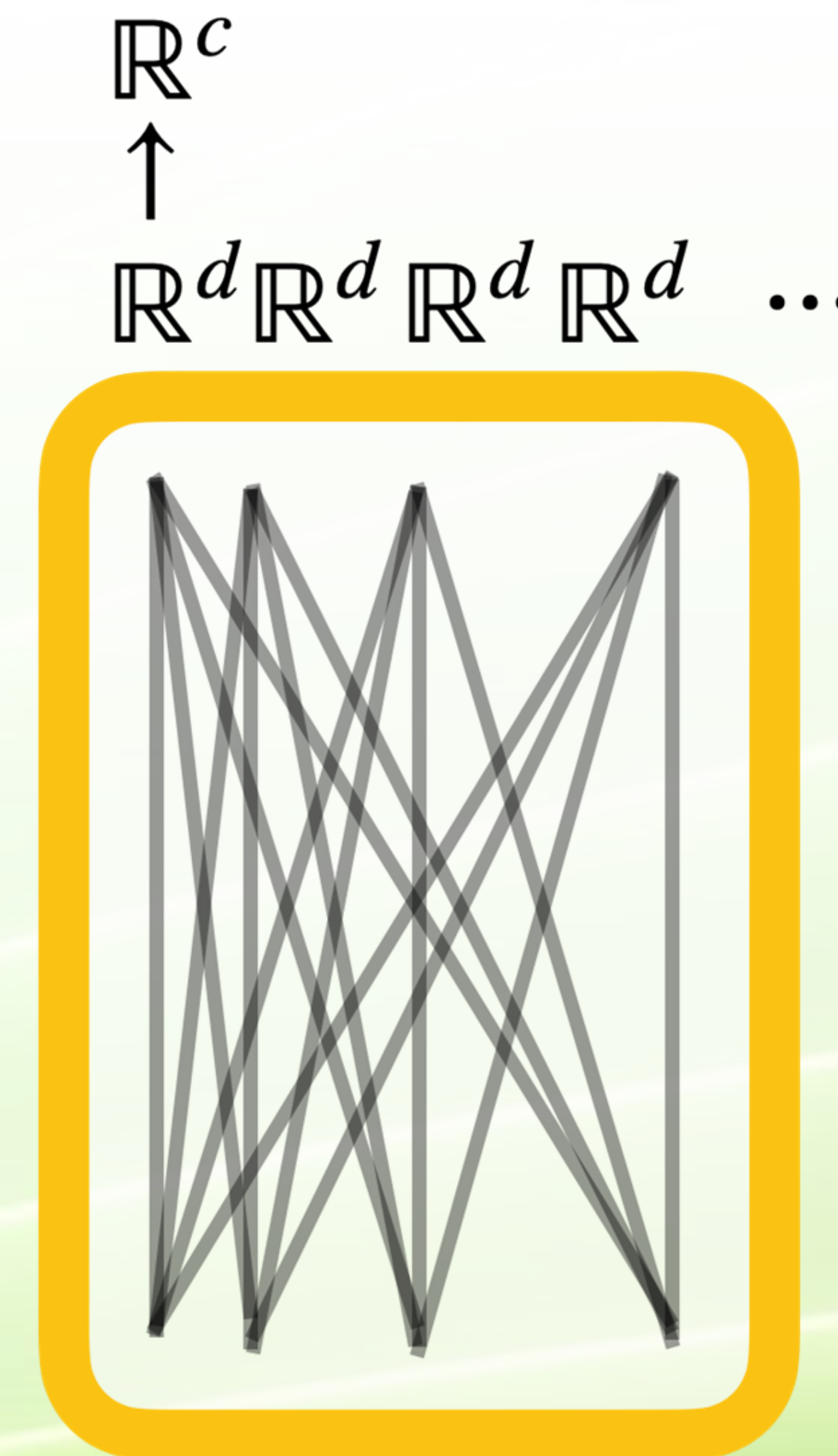
Autoregressing vs Embedding Flavors

Decoder For
Autoregressive Forecasting
(Common, Parallelizable)



$[s]$ *I like ice*

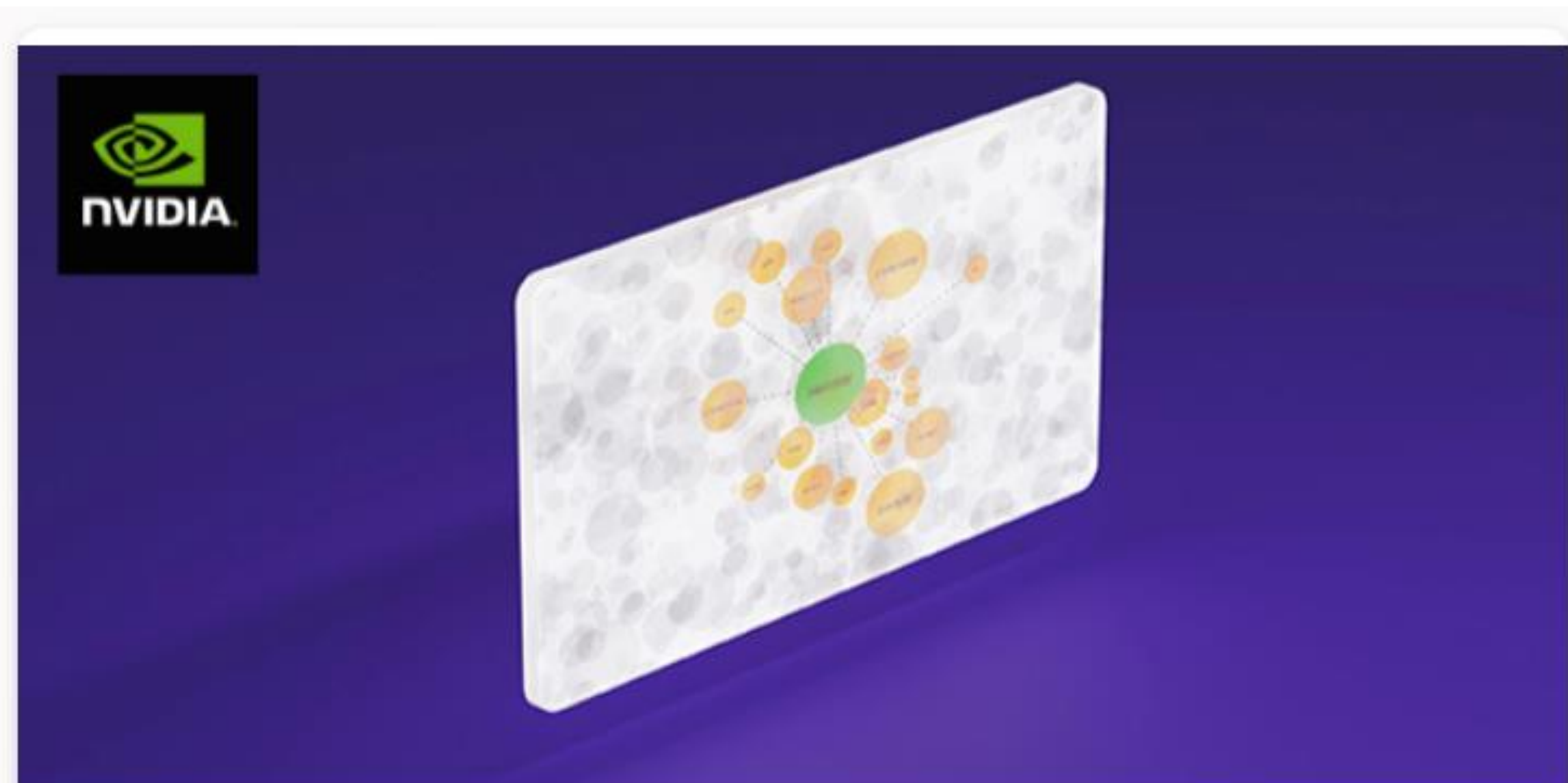
Encoder For
Token/Sequence Prediction
(Common, Fast, Lightweight)



$[CLS]$ *I like ice ...*

Retrieval QA Embedding

Asymmetric Query/Document Model



Description

NVIDIA Retrieval QA Embedding is an embedding model that represents words, phrases, or other entities as vectors of numbers and understands the relation between words and phrases.

Publisher

NVIDIA

```
"data": [  
  {  
    "index": 0,  
    "embedding": [  
      -0.02204068750143051,  
      -0.05545914173126221,  
      0.00420820526778698,  
      0.07957681268453598,  
      0.0136950071901083,  
      -0.005691236350685358,  
      -0.030427008867263794,  
      0.009445813484489918,  
      -0.02140694297850132,  
      -0.010374147444963455,  
      0.0348891019821167,  
      0.01636182703077793,  
      0.04929877817630768,  
      0.04269032925367355,
```

Embedding and Comparing

Querying for Semantically Similar Entries

High-performance computing.

Happy Holidays!

DLSS Gaming Statistics

Any cool video games lately?

Biological vision structure

What's with GPUs these days?

Mitochondria, powerhouse

<https://developer.nvidia.com/blog/accelerating-vector-search-using-gpu-powered-indexes-with-rapids-raft/>

Source: NVIDIA DLI (2024), Building RAG Agents with LLMs, https://learn.nvidia.com/courses/course-detail?course_id=course-v1:DLI+S-FX-15+V1

Embedding and Comparing

Querying for Semantically Similar Entries

High-performance computing.

Happy Holidays!

DLSS Gaming Statistics

Any cool video games lately?

Biological vision structure

What's with GPUs these days?

Mitochondria, powerhouse

Database	
 Text Image Video	▸ [1, 0, 3, 5]
 Text Image Video	▸ [9, 4, 6, 9]
 Text Image Video	▸ [4, 6, 2, 5]
 Text Image Video	▸ [6, 8, 6, 1]
 Text Image Video	▸ [7, 0, 3, 5]
 Text Image Video	▸ [1, 4, 6, 9]
 Text Image Video	▸ [4, 6, 2, 4]

Embedding and Comparing

Querying for Semantically Similar Entries

High-performance computing.

Happy Holidays!

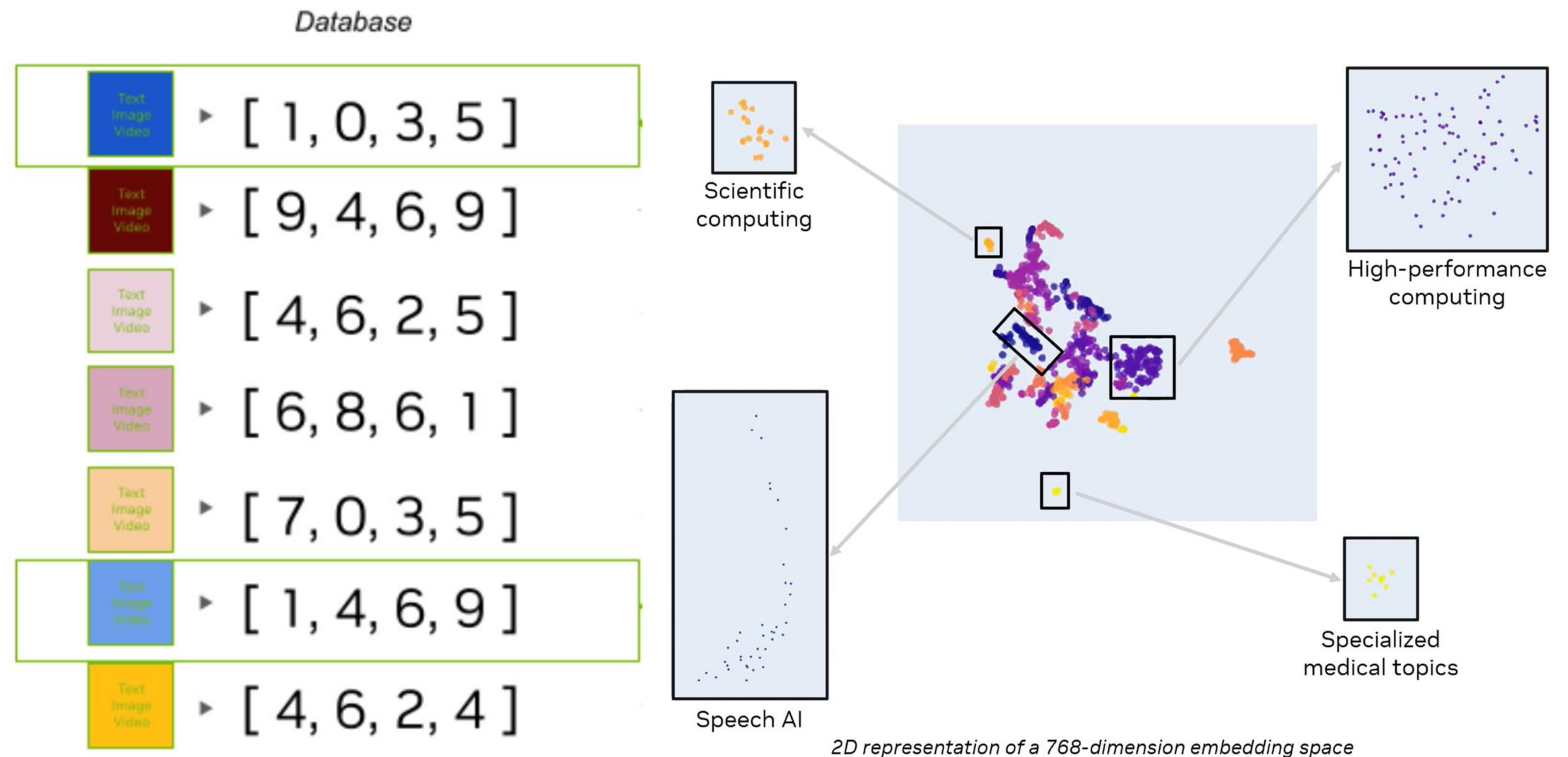
DLSS Gaming Statistics

Any cool video games lately?

Biological vision structure

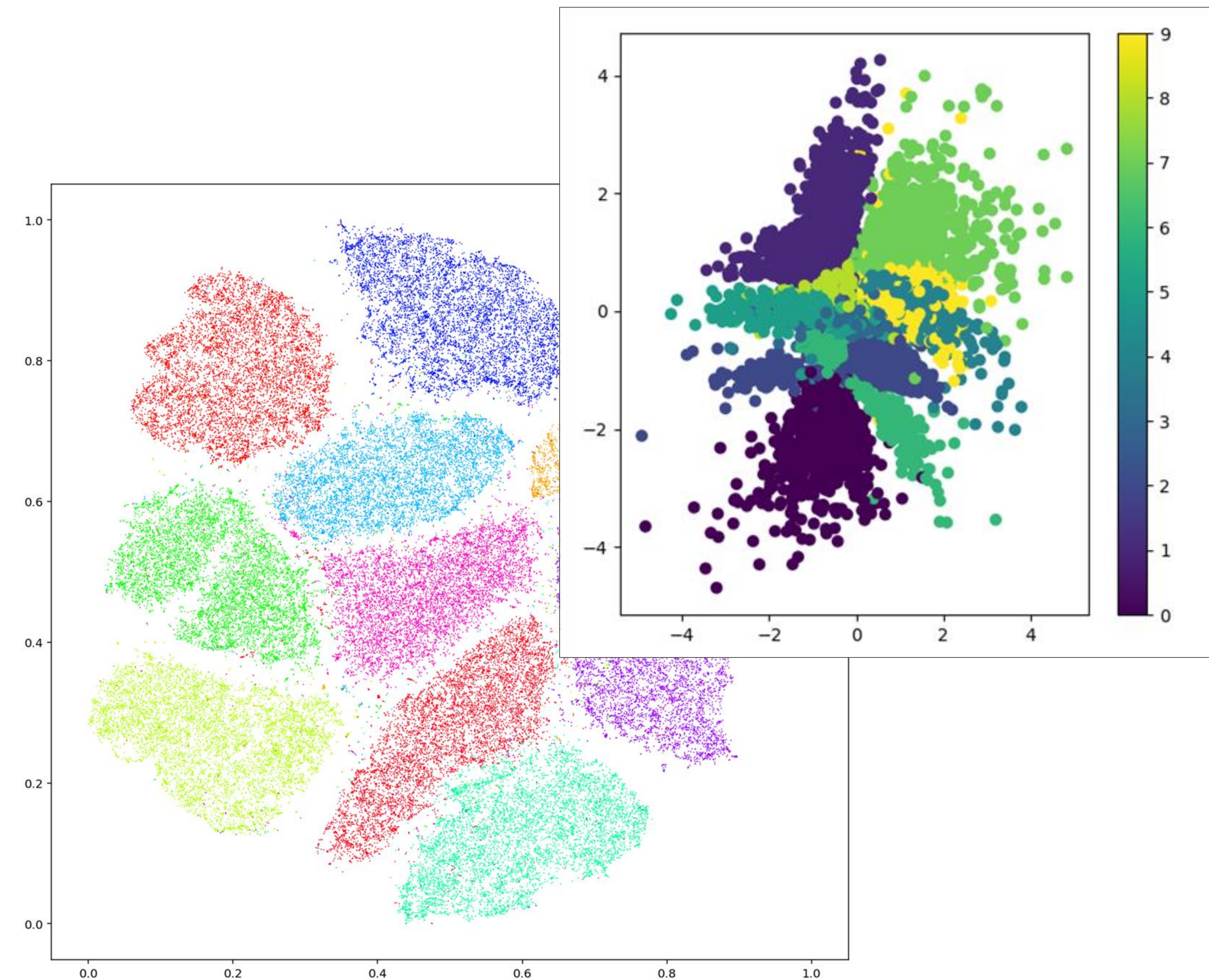
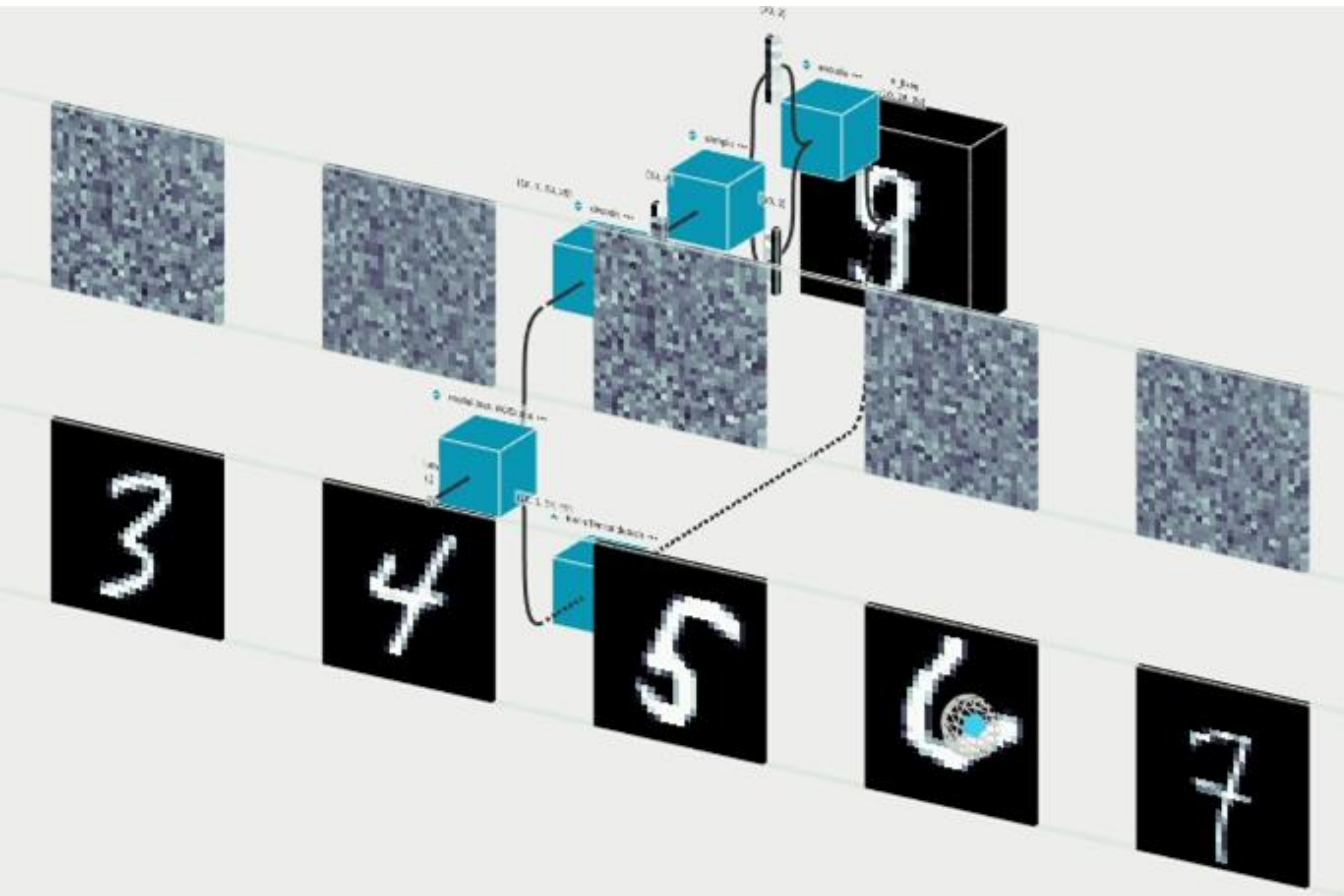
What's with GPUs these days?

Mitochondria, powerhouse



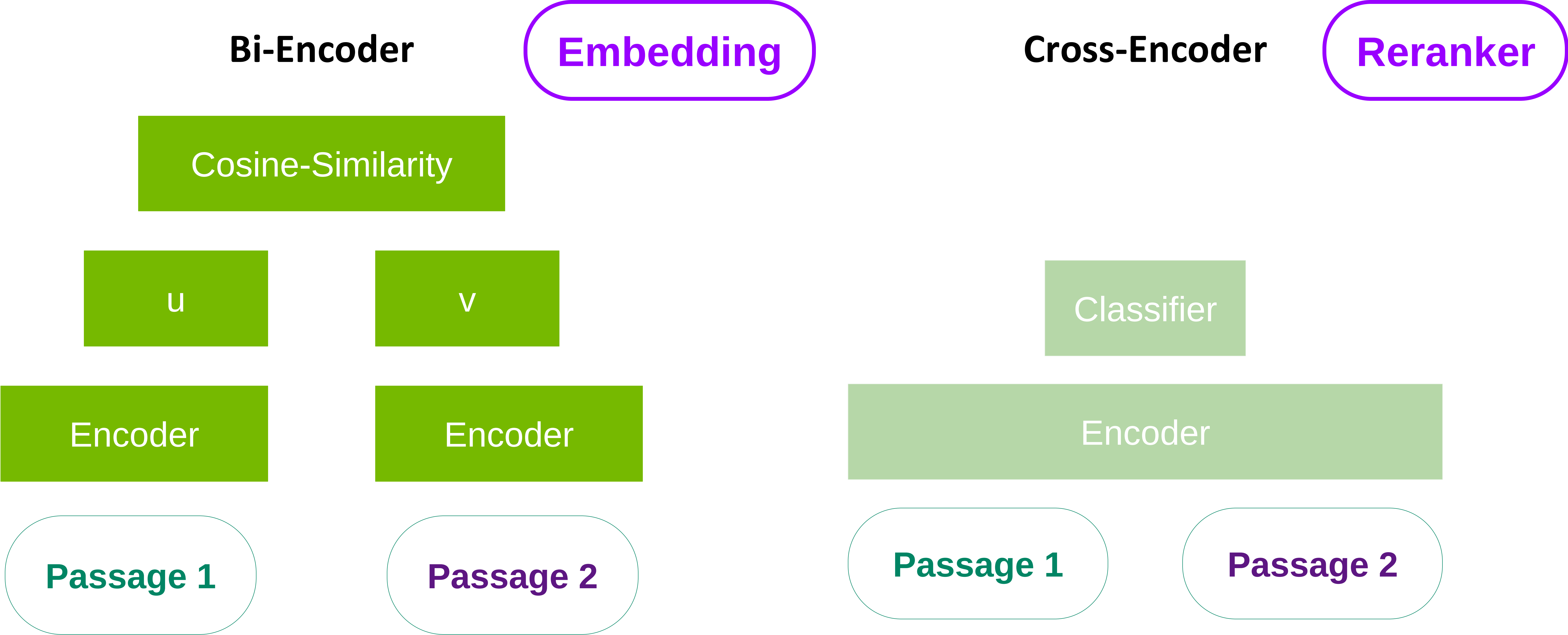
Embedding Spaces

Learned Semantically Dense Values



Language Embedding Schemes

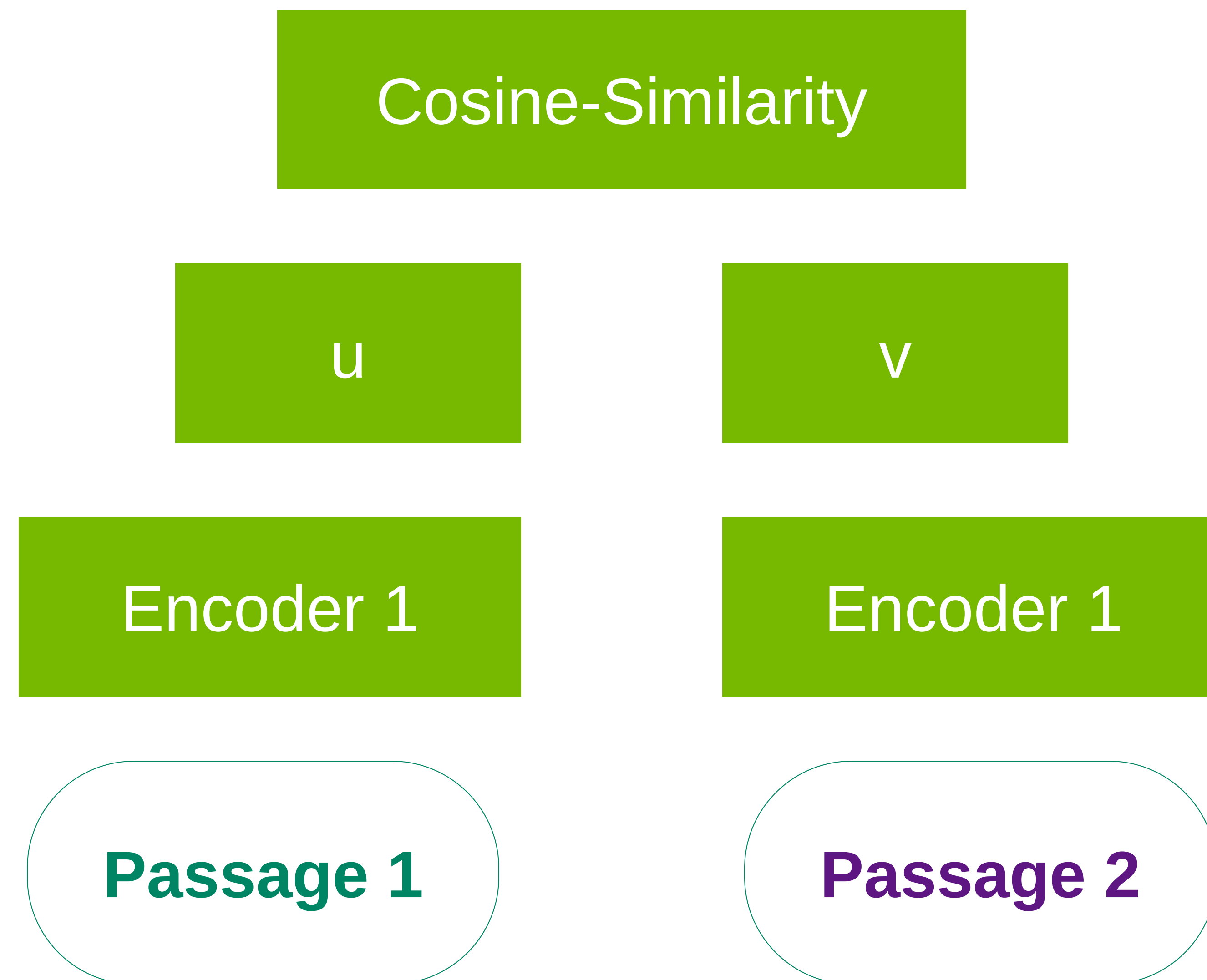
Bi-Encoder versus Cross-Encoder



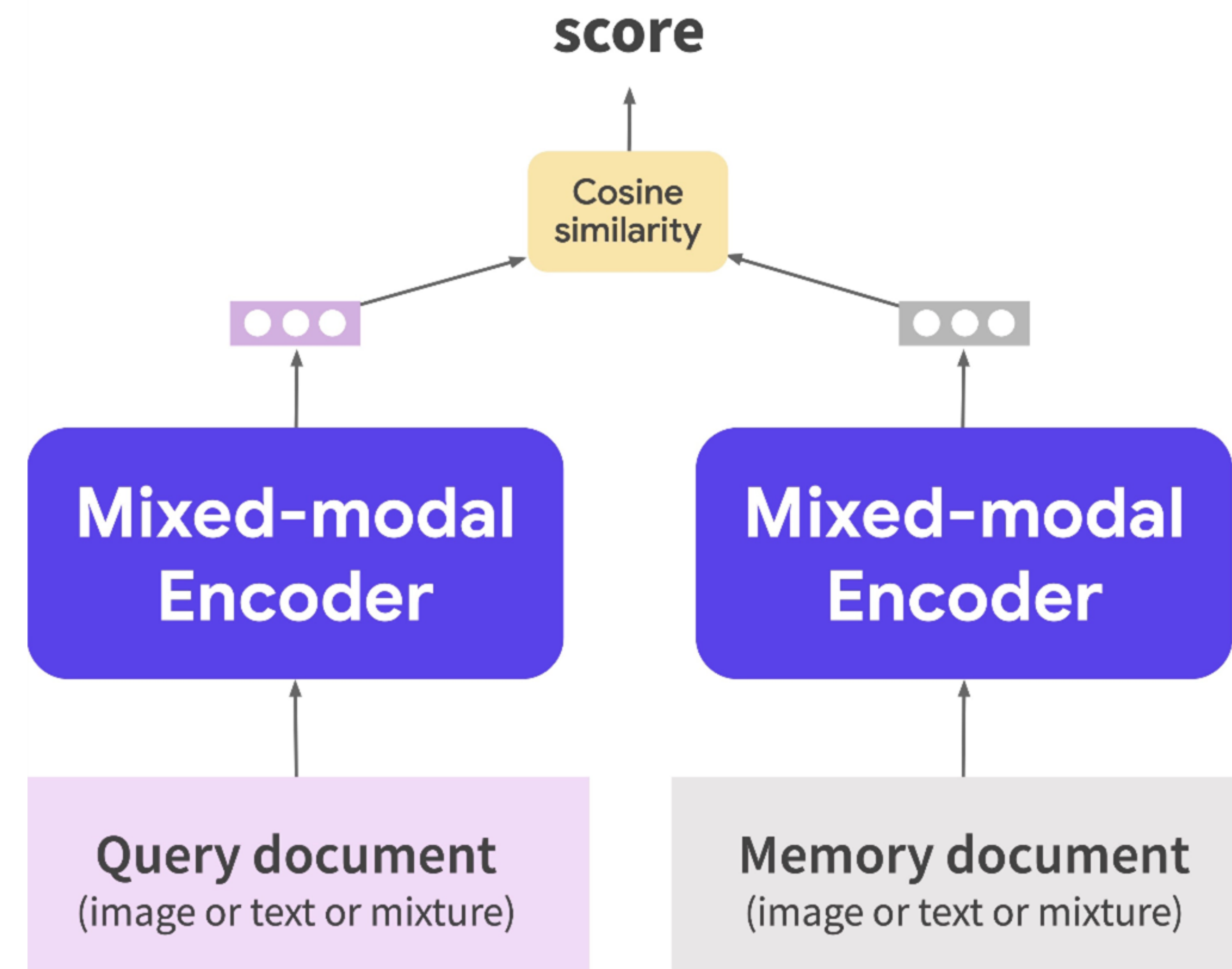
Language Embedding Schemes

Symmetric versus Asymmetric

Symmetric

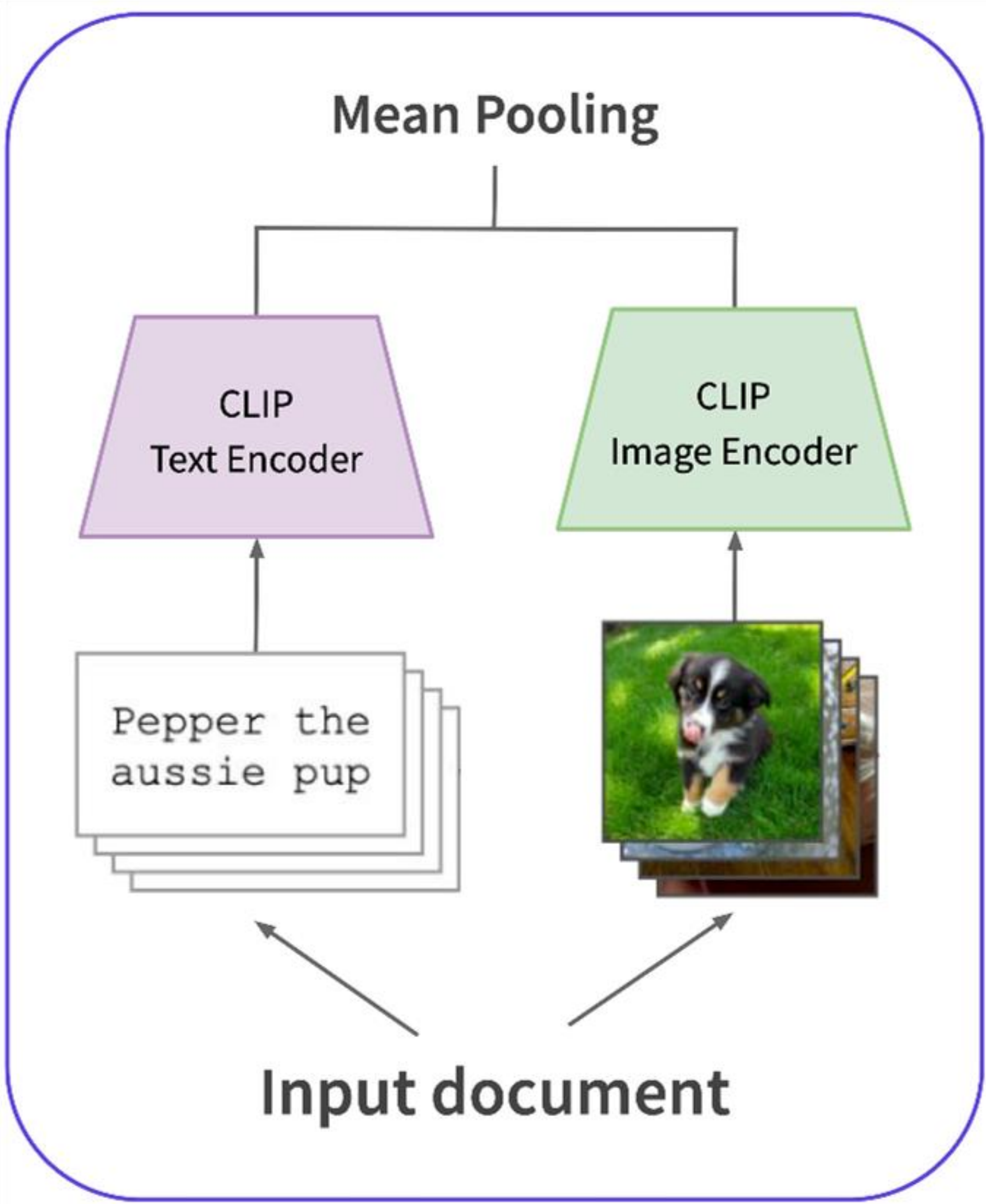
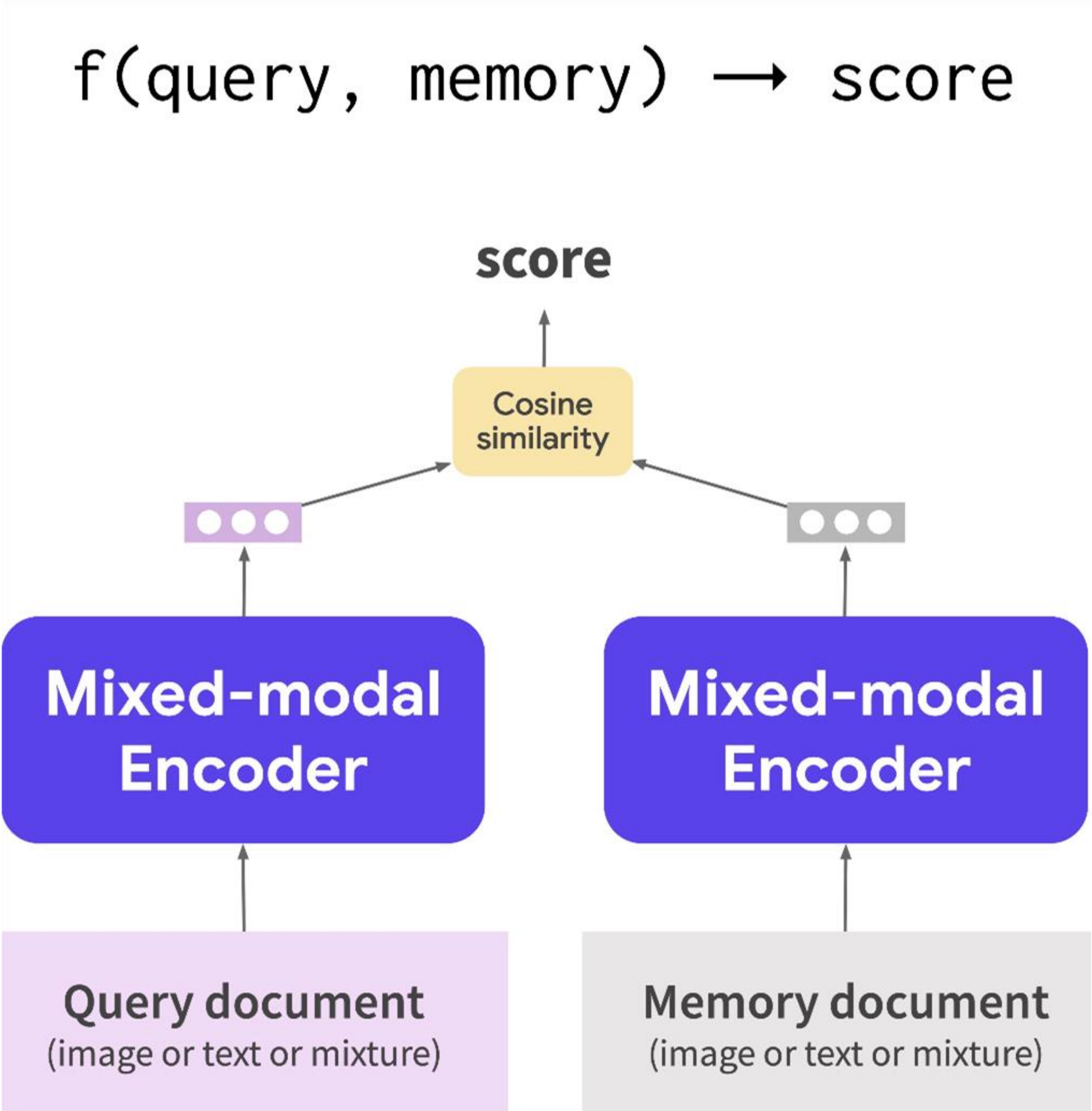


Asymmetric/Generalized



Language Embedding Schemes

Generalized Definition



Embedding and Comparing

Querying for Semantically Similar Entries

High-performance computing.

Happy Holidays!

DLSS Gaming Statistics

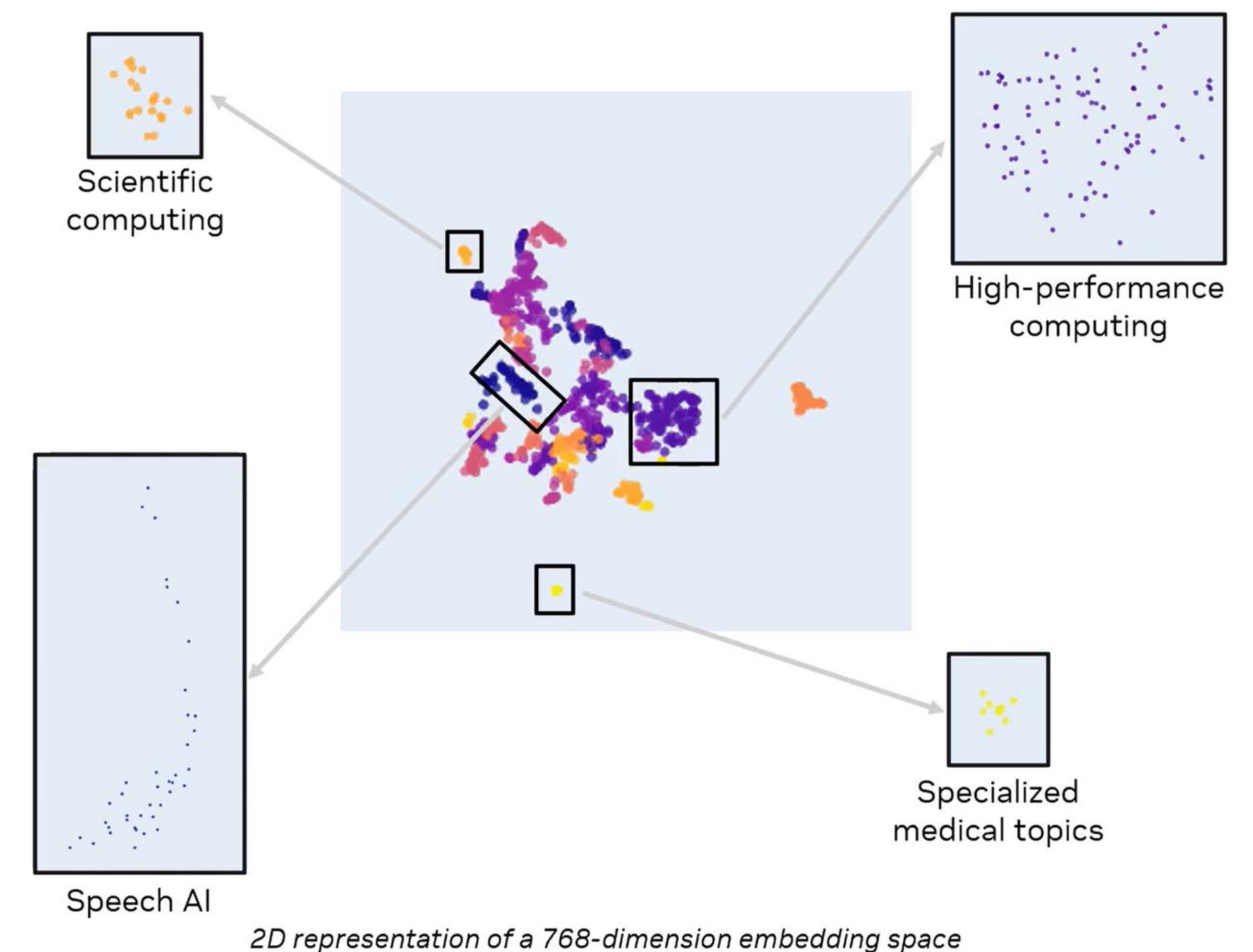
Any cool video games lately?

Biological vision structure

What's with GPUs these days?

Mitochondria, powerhouse

Database	
Text Image Video	[1, 0, 3, 5]
Text Image Video	[9, 4, 6, 9]
Text Image Video	[4, 6, 2, 5]
Text Image Video	[6, 8, 6, 1]
Text Image Video	[7, 0, 3, 5]
Text Image Video	[1, 4, 6, 9]
Text Image Video	[4, 6, 2, 4]





Building RAG Agents with LLMs

Part 6.4: Guardrails

Embedding and Comparing

Querying for Semantically Similar Entries

High-performance computing.

Happy Holidays!

DLSS Gaming Statistics

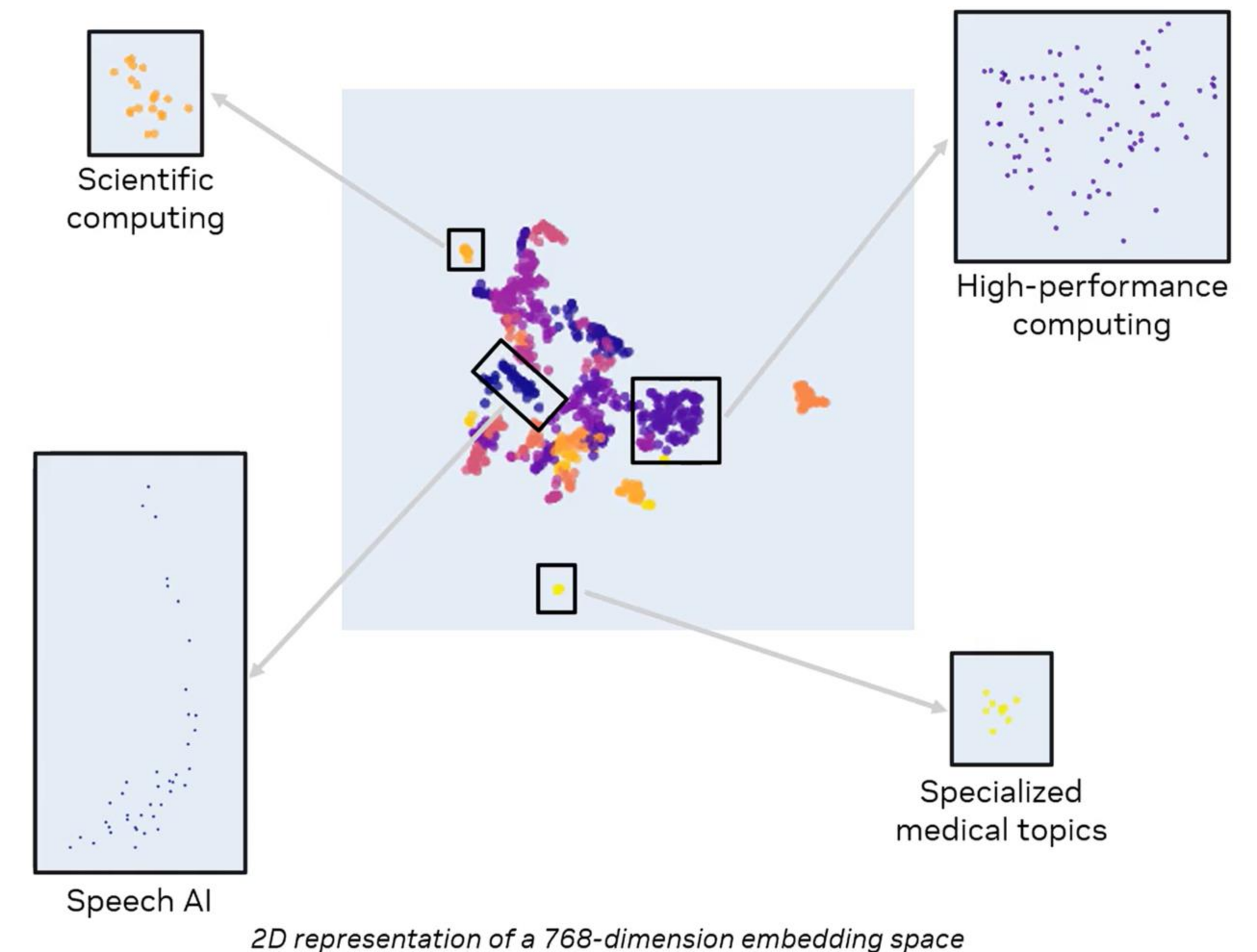
Any cool video games lately?

Biological vision structure

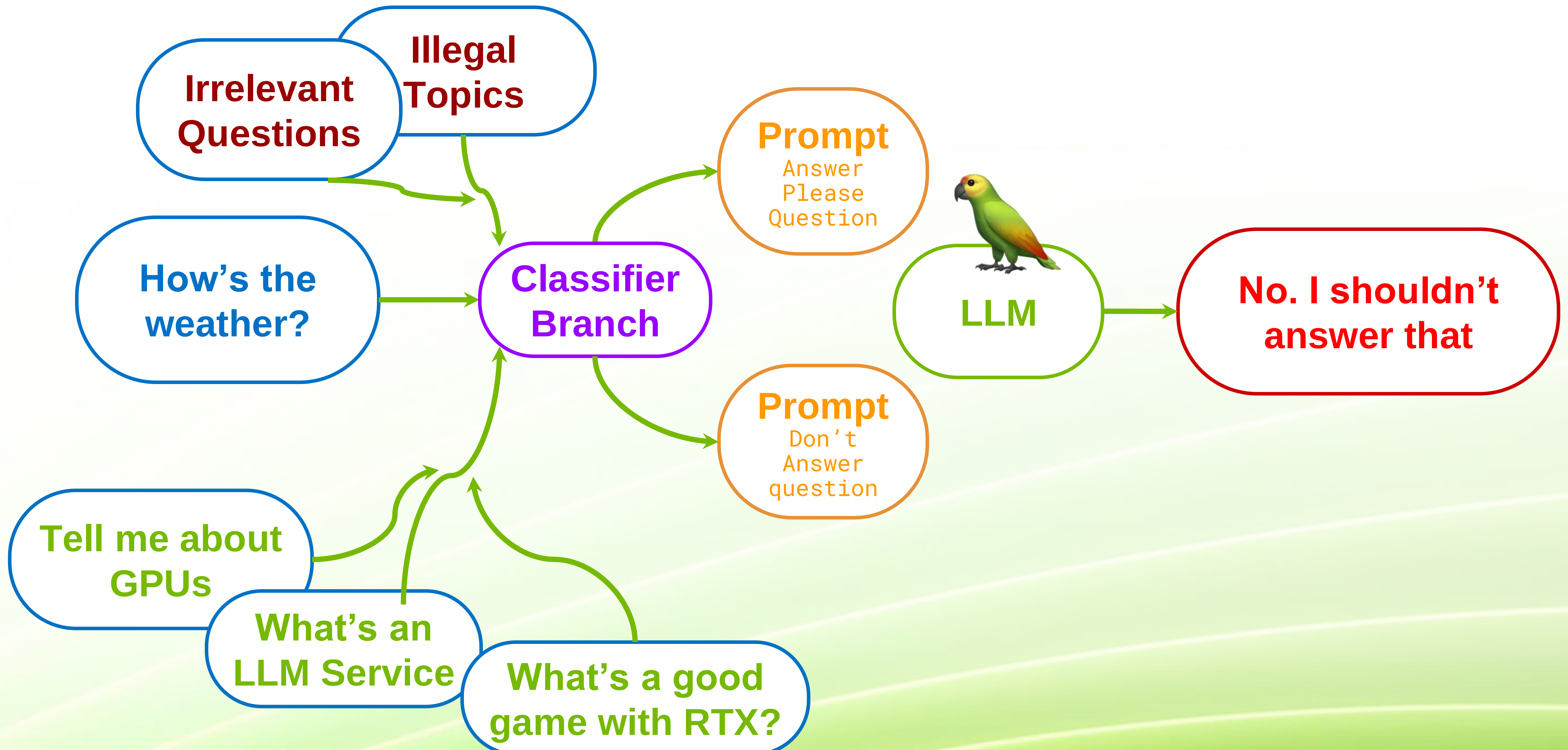
What's with GPUs these days?

Mitochondria, powerhouse

Database	
 Text Image Video	▶ [1, 0, 3, 5]
 Text Image Video	▶ [9, 4, 6, 9]
 Text Image Video	▶ [4, 6, 2, 5]
 Text Image Video	▶ [6, 8, 6, 1]
 Text Image Video	▶ [7, 0, 3, 5]
 Text Image Video	▶ [1, 4, 6, 9]
 Text Image Video	▶ [4, 6, 2, 4]



Semantic Guardrails



Embedding Classification

Classifying with embeddings

**Irrelevant
Questions**

**Illegal
Topics**

**Tell me about
GPUs**

**What's an
LLM Service**

**What's a good
game with RTX?**

<https://developer.nvidia.com/blog/accelerating-vector-search-using-gpu-powered-indexes-with-rapids-raft/>

Embedding Classification

Classifying with embeddings

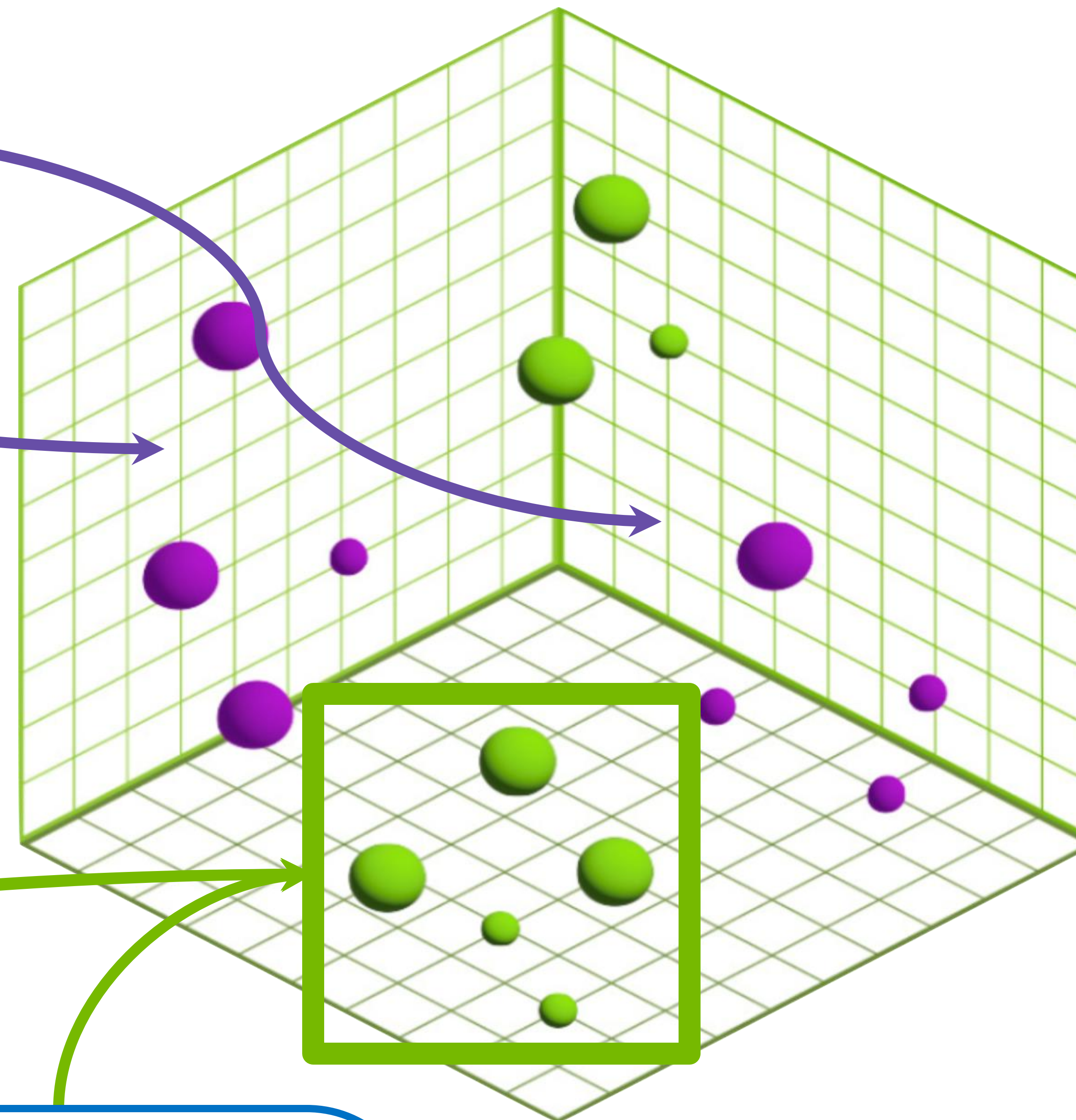
Irrelevant
Questions

Illegal
Topics

Tell me about
GPUs

What's an
LLM Service

What's a good
game with RTX?



<https://developer.nvidia.com/blog/accelerating-vector-search-using-gpu-powered-indexes-with-rapids-raft/>

Embedding Classification

Classifying with embeddings

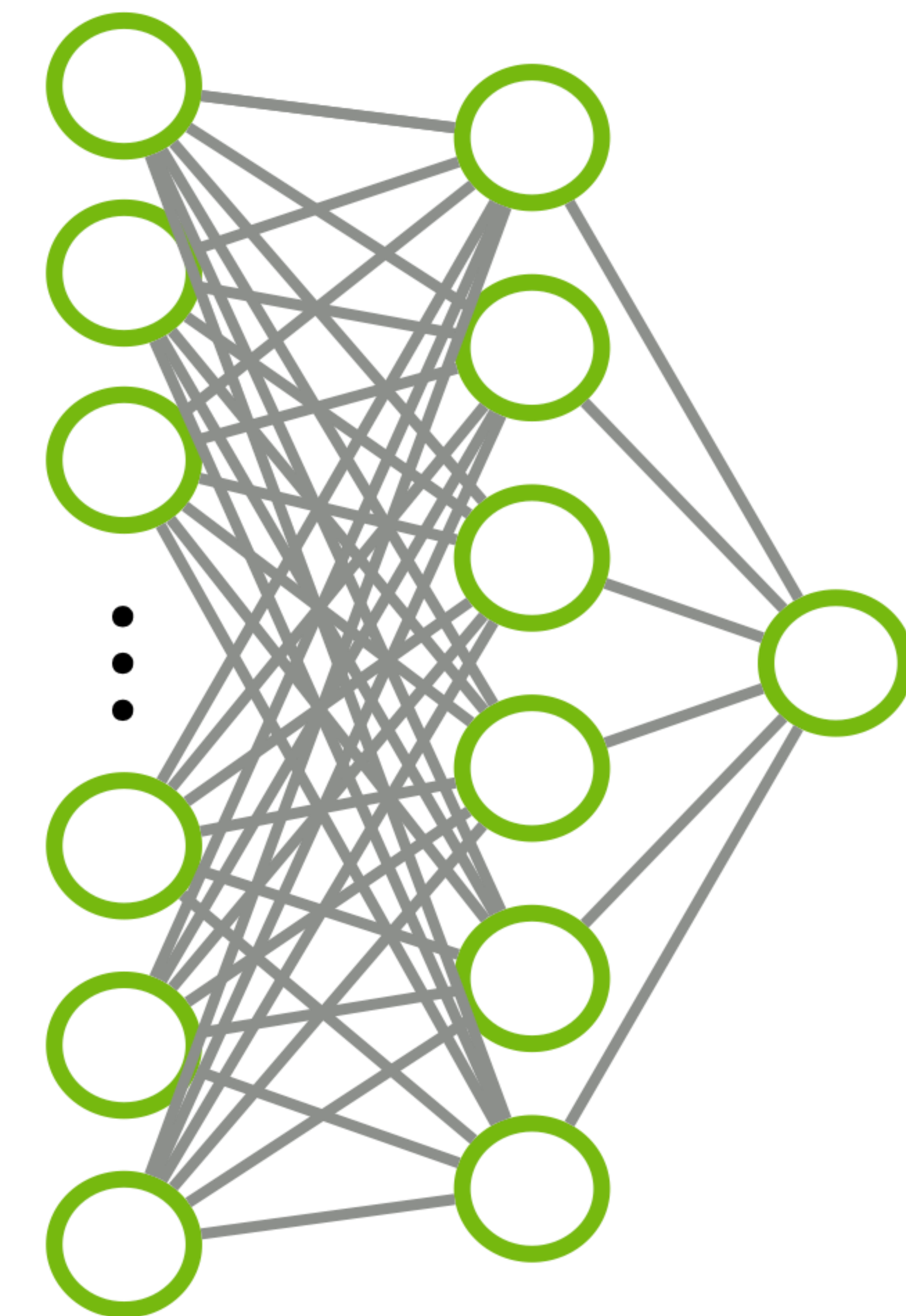
Irrelevant Questions
Illegal Topics

Tell me about GPUs

What's an LLM Service

What's a good game with RTX?

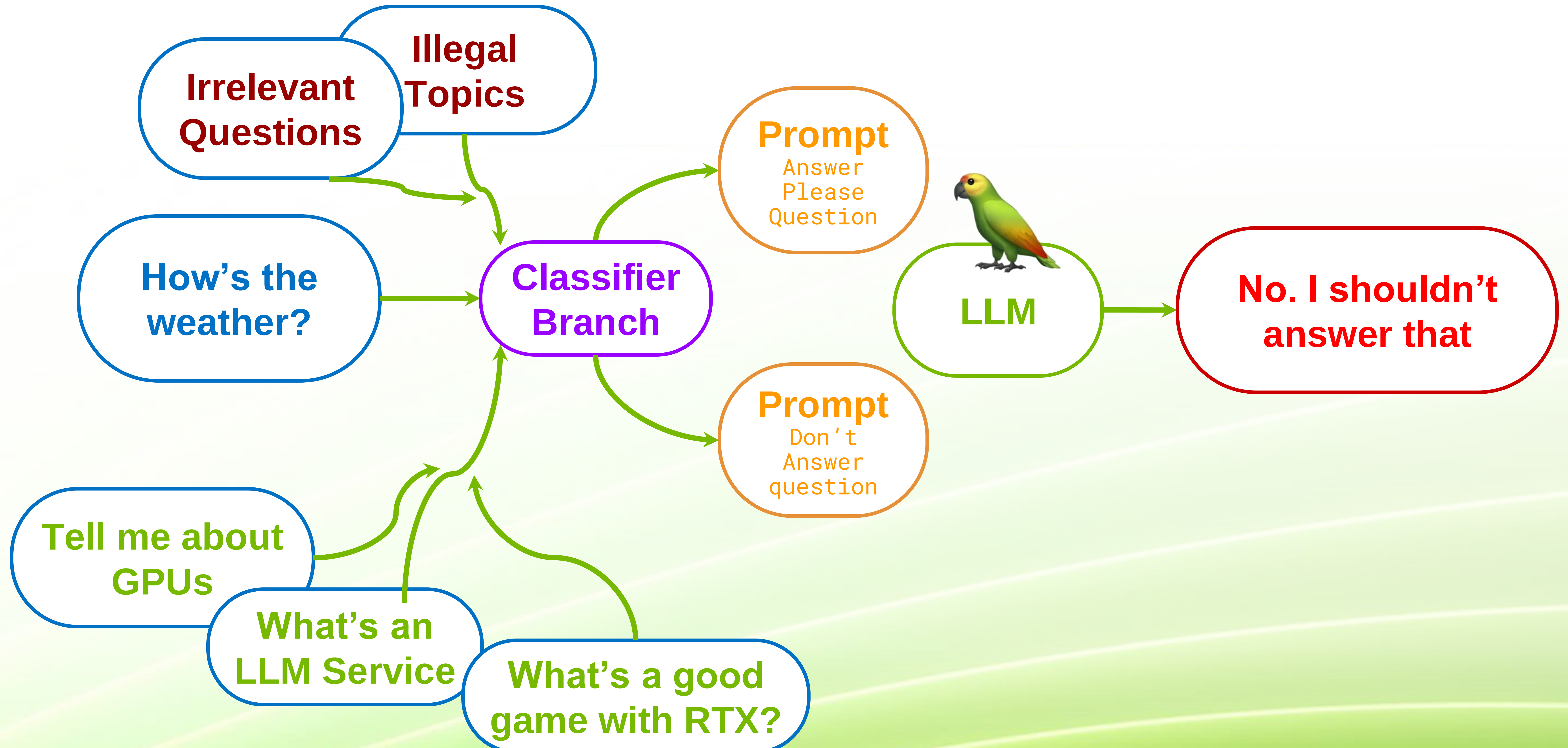
Classification Head



0 or 1?

<https://developer.nvidia.com/blog/accelerating-vector-search-using-gpu-powered-indexes-with-rapids-raft/>

Semantic Guardrails



NVIDIA Building RAG Agents with LLMs

Part II

[5, 6]

Building RAG Agents with LLMs

Course Slides

Course Slides

Environment and LLMs

Environment [0, 1, 2]

LangChain

Environment [3, 4]

Documents and Embeddings

Environment [5, 6]



DEEP
LEARNING
INSTITUTE

This Lab 0 : 08 : 28 / 2 : 00 : 00

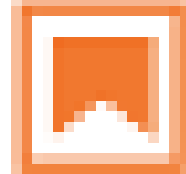
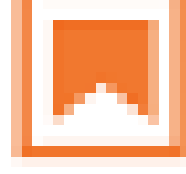
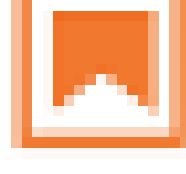
Course 2 : 09 : 22 / 32 : 00 : 00



LAUNCH



STOP TASK

-  00_jupyterlab.ipynb
-  01_microservices.ipynb
-  02_llms.ipynb
-  03_langchain_intro.ipynb
-  04_running_state.ipynb
-  05_documents.ipynb
-  06_embeddings.ipynb

References

- NVIDIA DLI (2024), Building RAG Agents with LLMs,
https://learn.nvidia.com/courses/course-detail?course_id=course-v1:DLI+S-FX-15+V1
- NVIDIA DLI (2024), Generative AI with Diffusion Models,
https://learn.nvidia.com/courses/course-detail?course_id=course-v1:DLI+S-FX-14+V1