AI Masterclass 2025

May 12-16, Trinity College Dublin

Welcome!

This is me

Diego Maniloff

dmanilof@redhat.com

UNATA instacart nextdoor Hugging Face Red Hat

Not a typo

Diego Maniloff

dmanilof@redhat.com

UNATA instacart nextdoor Hugging Face Red Hat

Masterclass Outline

From scratch to an advanced RAG agent with guardrails

```python
# Unit 1
agent = Agent(
    llama_stack_client,
    model="my-model",
    instructions="You are a helpful assistant that can use tools to answer questions.",
    tools=["basic-rag-tool"],
)
```

Masterclass Outline

From scratch to an advanced RAG agent with guardrails

```
# Unit 1
agent = Agent(
    llama_stack_client,
    model="my-model",
    instructions="You are a helpful assistant that can use tools to answer questions.",
    tools=["basic-rag-tool"],
)
```

Masterclass Outline

From scratch to an advanced RAG agent with guardrails

```python
# Unit 1
agent = Agent(
    llama_stack_client,
    model="my-model",
    instructions="You are a helpful assistant that can use tools to answer questions.",
    tools=["basic-rag-tool"],
)
```

Masterclass Outline

From scratch to an advanced RAG agent with guardrails

```
# Unit 2
agent = Agent(
    llama_stack_client,
    model="my-model",
    instructions="You are a helpful assistant that can use tools to answer questions.",
    tools=["basic-rag-tool", "advanced-rag-tool"],
)
```

Masterclass Outline

From scratch to an advanced RAG agent with guardrails

```
# Unit 2
agent = Agent(
    llama_stack_client,
    model="my-model",
    instructions="You are a helpful assistant that can use tools to answer questions.",
    tools=["basic-rag-tool", "advanced-rag-tool"],
)
```

Masterclass Outline

From scratch to an advanced RAG agent with guardrails

```
# Unit 2
agent = Agent(
    llama_stack_client,
    model="my-model",
    instructions="You are a helpful assistant that can use tools to answer questions.",
    tools=["basic-rag-tool", "advanced-rag-tool"],
)
```

Masterclass Outline

From scratch to an advanced RAG agent with guardrails

```
# Unit 3
agent = Agent(
    llama_stack_client,
    model="my-model",
    instructions="You are a helpful assistant that can use tools to answer questions.",
    tools=["basic-rag-tool", "advanced-rag-tool"],
    input_shields=["guardrails::my-filter"],
    output_shields=["guardrails::my-filter"],
)
```

Masterclass Outline

From scratch to an advanced RAG agent with guardrails

```
# Unit 3
agent = Agent(
    llama_stack_client,
    model="my-model",
    instructions="You are a helpful assistant that can use tools to answer questions.",
    tools=["basic-rag-tool", "advanced-rag-tool"],
    input_shields=["guardrails::my-filter"],
    output_shields=["guardrails::my-filter"],
)
```

Masterclass Outline

From scratch to an advanced RAG agent with guardrails

```
# Unit 3
agent = Agent(
    llama_stack_client,
    model="my-model",
    instructions="You are a helpful assistant that can use tools to answer questions.",
    tools=["basic-rag-tool", "advanced-rag-tool"],
    input_shields=["guardrails::my-filter"],
    output_shields=["guardrails::my-filter"],
)
```

Unit 1
Introduction to Llama Stack

Unit 1 Goals

- Understand the core concepts of Llama Stack
- Get our development environment working
- Prepare for Units 2 and 3
- Start your Masterclass Project

Intros

What we're doing today

1. Why Llama Stack?
2. Environment Setup
3. Our Own (tiny) Stack
4. Core Concepts
5. 🎹 Let's use our stack
6. 🎹 Let's break our stack
7. Takeaways
8. 🎹 Implement RAG from scratch
9. Agents
10. Agents on Llama Stack
11. Execution model
12. 🎹 An agent that can find its location
13. 🎹 An agent that can search the web
14. 🎹 An agent that can read files

Format

1. 📽️📽️📽️, then 🎹
2. 📽️📽️, then 🎹 🎹
3. 📽️, then 🎹 🎹 🎹

Don’t worry, we’ll take breaks!

- 8:30 - 9am: Breakfast
- 10am: Morning coffee break
- 12-1pm: Lunch
- 2:30pm: Afternoon coffee break

What is Llama Stack and why is it a good idea?

AI System Design

AI System Design

I just need model inference, `curl https://api.openai.com/v1/chat/completions` *and I'm done*

AI System Design

AI System Design

- ~~I just need model inference,~~ ~~`curl https://api.openai.com/v1/chat/completions`~~ ~~and I'm done~~

AI System Design

- ~~I just need model inference, `curl https://api.openai.com/v1/chat/completions` and I'm done~~
- That's not very useful. Let's connect the model to our data. **We need retrieval.**

AI System Design

- ~~I just need model inference,~~ ~~`curl https://api.openai.com/v1/chat/completions`~~ ~~and I'm done~~
- That's not very useful. Let's connect the model to our data. **We need retrieval.**
- Hmm, but exactly how useful is this? **We need evaluation and metrics.**

AI System Design

- ~~I just need model inference, `curl https://api.openai.com/v1/chat/completions` and I'm done~~
- That's not very useful. Let's connect the model to our data. **We need retrieval.**
- Hmm, but exactly how useful is this? **We need evaluation and metrics.**
- But wait! Will it leak private information? **We need safety guardrails and content filtering.**

AI System Design

- ~~I just need model inference, `curl https://api.openai.com/v1/chat/completions` and I'm done~~
- That's not very useful. Let's connect the model to our data. **We need retrieval.**
- Hmm, but exactly how useful is this? **We need evaluation and metrics.**
- But wait! Will it leak private information? **We need safety guardrails and content filtering.**
- …

AI System Design

- ~~I just need model inference, `curl https://api.openai.com/v1/chat/completions` and I'm done~~
- That's not very useful. Let's connect the model to our data. **We need retrieval.**
- Hmm, but exactly how useful is this? **We need evaluation and metrics.**
- But wait! Will it leak private information? **We need safety guardrails and content filtering.**
- …
- **We need agents**

AI System Design

- ~~I just need model inference, `curl https://api.openai.com/v1/chat/completions` and I'm done~~
- That's not very useful. Let's connect the model to our data. **We need retrieval.**
- Hmm, but exactly how useful is this? **We need evaluation and metrics.**
- But wait! Will it leak private information? **We need safety guardrails and content filtering.**
- …
- **We need agents**
- **We need tool calling**

AI System Design

- ~~I just need model inference, `curl https://api.openai.com/v1/chat/completions` and I'm done~~
- That's not very useful. Let's connect the model to our data. **We need retrieval.**
- Hmm, but exactly how useful is this? **We need evaluation and metrics.**
- But wait! Will it leak private information? **We need safety guardrails and content filtering.**
- …
- **We need agents**
- **We need tool calling**
- **We need monitoring**

AI System Design

- ~~I just need model inference, `curl https://api.openai.com/v1/chat/completions` and I'm done~~
- That's not very useful. Let's connect the model to our data. **We need retrieval.**
- Hmm, but exactly how useful is this? **We need evaluation and metrics.**
- But wait! Will it leak private information? **We need safety guardrails and content filtering.**
- …
- **We need agents**
- **We need tool calling**
- **We need monitoring**
- …

AI System Design

- ~~I just need model inference, `curl https://api.openai.com/v1/chat/completions` and I'm done~~
- That's not very useful. Let's connect the model to our data. **We need retrieval.**
- Hmm, but exactly how useful is this? **We need evaluation and metrics.**
- But wait! Will it leak private information? **We need safety guardrails and content filtering.**
- …
- **We need agents**
- **We need tool calling**
- **We need monitoring**
- …
- How do we deploy this whole thing?

What is Llama Stack and why is it a good idea?

What is Llama Stack and why is it a good idea?

Opinionated Organization

What is Llama Stack and why is it a good idea?

Helpful Organization

What is Llama Stack and why is it a good idea?

Organization

Organization

What's there to organize?

- Inference
- Retrieval
- Evaluation
- Safety
- Agents
- Tool calling
- Monitoring
- Deployment
- … and more!

Dev Environment

Environment Setup

Setting up the local development environment

Run the ollama server

```
ollama pull llama3.2:3b-instruct-fp16
ollama serve
```

Clone the masterclass repo

```
git clone https://github.com/trustyai-explainability/ai-masterclass-2025.git
cd ai-masterclass-2025
```

Build and run the stack

```
uv run --with "llama-stack" llama stack build --config=distributions/masterclass-minimal/build.yaml
```

```
source masterclass_minimal/bin/activate
llama stack run distributions/masterclass-minimal/run.yaml
```

Environment Setup

What happened?

```
$ tree distributions
distributions
├── masterclass-agents
│   ├── build.yaml
│   └── run.yaml
└── masterclass-minimal
    ├── build.yaml
    └── run.yaml
```

Distribution config files (from our repo)

```
$ ls -l masterclass_minimal
total 16
-rw-r--r--   1 diego  staff    43B May  9 09:27 CACHEDIR.TAG
drwxr-xr-x  39 diego  staff   1.2K May  9 09:27 bin
drwxr-xr-x   3 diego  staff    96B May  9 09:27 lib
-rw-r--r--   1 diego  staff   176B May  9 09:27 pyvenv.cfg
```

Pyton virtual environment (generated by the build command)

```
$ tree ~/.llama/distributions/masterclass_minimal
/Users/diego/.llama/distributions/masterclass_minimal
├── kvstore.db
├── masterclass_minimal-build.yaml
```

Distribution directory (generated by the build command)

Our Own (tiny) Stack

Our Own Stack

🙌🏻 We just ran our first stack!

```
cat distributions/masterclass-minimal/run.yaml
```

```yaml
version: "2"
image_name: masterclass_minimal
apis:
  - inference
  - telemetry
providers:
  inference:
    - provider_id: ollama
      provider_type: remote::ollama
      config:
        url: ${env.OLLAMA_URL:http://localhost:11434}
  telemetry:
    - provider_id: meta-reference
      provider_type: inline::meta-reference
      config:
        service_name: ${env.OTEL_SERVICE_NAME:}
      sinks: ${env.TELEMETRY_SINKS:console,sqlite}
      sqlite_db_path: ${env.SQLITE_STORE_DIR:~/.llama/dist
models:
  - metadata: {}
    model_id: meta-llama/Llama-3.2-3B-Instruct
    provider_id: ollama
    model_type: llm
    provider_model_id: llama3.2:3b-instruct-fp16 # from ol
server:
  host: localhost
  port: 8321
```

Core Concepts

Core Concepts

- Api
- Provider
- Resource
- Distributions

https://llama-stack.readthedocs.io/en/latest/concepts/index.html

```yaml
version: "2"
image_name: masterclass_minimal
apis:
  - inference
  - telemetry
providers:
  inference:
    - provider_id: ollama
      provider_type: remote::ollama
      config:
        url: ${env.OLLAMA_URL:http://localhost:11434}
  telemetry:
    - provider_id: meta-reference
      provider_type: inline::meta-reference
      config:
        service_name: ${env.OTEL_SERVICE_NAME:}
      sinks: ${env.TELEMETRY_SINKS:console,sqlite}
      sqlite_db_path: ${env.SQLITE_STORE_DIR:~/.llama/dist
models:
  - metadata: {}
    model_id: meta-llama/Llama-3.2-3B-Instruct
    provider_id: ollama
    model_type: llm
    provider_model_id: llama3.2:3b-instruct-fp16 # from ol
server:
  host: localhost
  port: 8321
```

Core Concepts

- Api
- Provider
- Resource
- Distributions

> An API is a collection of endpoints.

https://llama-stack.readthedocs.io/en/latest/concepts/index.html

```yaml
version: "2"
image_name: masterclass_minimal
apis:
  - inference
  - telemetry
providers:
  inference:
    - provider_id: ollama
      provider_type: remote::ollama
      config:
        url: ${env.OLLAMA_URL:http://localhost:11434}
  telemetry:
    - provider_id: meta-reference
      provider_type: inline::meta-reference
      config:
        service_name: ${env.OTEL_SERVICE_NAME:}
      sinks: ${env.TELEMETRY_SINKS:console,sqlite}
      sqlite_db_path: ${env.SQLITE_STORE_DIR:~/.llama/dist
models:
  - metadata: {}
    model_id: meta-llama/Llama-3.2-3B-Instruct
    provider_id: ollama
    model_type: llm
    provider_model_id: llama3.2:3b-instruct-fp16 # from ol
server:
  host: localhost
  port: 8321
```

Core Concepts

- Api
- Provider
- Resource
- Distributions

> An API is a collection of endpoints.

> Providers implement APIs …

https://llama-stack.readthedocs.io/en/latest/concepts/index.html

```yaml
version: "2"
image_name: masterclass_minimal
apis:
  - inference
  - telemetry
providers:
  inference:
    - provider_id: ollama
      provider_type: remote::ollama
      config:
        url: ${env.OLLAMA_URL:http://localhost:11434}
  telemetry:
    - provider_id: meta-reference
      provider_type: inline::meta-reference
      config:
        service_name: ${env.OTEL_SERVICE_NAME:}
      sinks: ${env.TELEMETRY_SINKS:console,sqlite}
      sqlite_db_path: ${env.SQLITE_STORE_DIR:~/.llama/dist
models:
  - metadata: {}
    model_id: meta-llama/Llama-3.2-3B-Instruct
    provider_id: ollama
    model_type: llm
    provider_model_id: llama3.2:3b-instruct-fp16 # from ol
server:
  host: localhost
  port: 8321
```

Core Concepts

- Api
- Provider
- Resource
- Distributions

> An API is a collection of endpoints.

> Providers implement APIs …

> … and provide resources.

https://llama-stack.readthedocs.io/en/latest/concepts/index.html

```yaml
version: "2"
image_name: masterclass_minimal
apis:
  - inference
  - telemetry
providers:
  inference:
    - provider_id: ollama
      provider_type: remote::ollama
      config:
        url: ${env.OLLAMA_URL:http://localhost:11434}
  telemetry:
    - provider_id: meta-reference
      provider_type: inline::meta-reference
      config:
        service_name: ${env.OTEL_SERVICE_NAME:}
      sinks: ${env.TELEMETRY_SINKS:console,sqlite}
      sqlite_db_path: ${env.SQLITE_STORE_DIR:~/.llama/dist
models:
  - metadata: {}
    model_id: meta-llama/Llama-3.2-3B-Instruct
    provider_id: ollama
    model_type: llm
    provider_model_id: llama3.2:3b-instruct-fp16 # from ol
server:
  host: localhost
  port: 8321
```

Core Concepts

- Api
- Provider
- Resource
- Distributions

An API is a collection of endpoints.

Providers implement APIs …

… and provide resources.

A distribution is a collection of specific providers (+ config).

https://llama-stack.readthedocs.io/en/latest/concepts/index.html

```yaml
version: "2"
image_name: masterclass_minimal
apis:
  - inference
  - telemetry
providers:
  inference:
    - provider_id: ollama
      provider_type: remote::ollama
      config:
        url: ${env.OLLAMA_URL:http://localhost:11434}
  telemetry:
    - provider_id: meta-reference
      provider_type: inline::meta-reference
      config:
        service_name: ${env.OTEL_SERVICE_NAME:}
      sinks: ${env.TELEMETRY_SINKS:console,sqlite}
      sqlite_db_path: ${env.SQLITE_STORE_DIR:~/.llama/dist
models:
  - metadata: {}
    model_id: meta-llama/Llama-3.2-3B-Instruct
    provider_id: ollama
    model_type: llm
    provider_model_id: llama3.2:3b-instruct-fp16 # from ol
server:
  host: localhost
  port: 8321
```

Core Concepts

APIs map to resources

Inference, Eval and Post Training	Model
Safety	Shield
Tool Runtime	ToolGroup
DatasetIO	Dataset
VectorIO	VectorDB
Scoring	ScoringFunction
Eval	Model and Benchmark

https://llama-stack.readthedocs.io/en/latest/concepts/index.html

 Let’s use our stack

Let’s use our stack

From the CLI

```
llama-stack-client --help
```

```
llama-stack-client providers list
```

```
llama-stack-client models list
```

```
llama-stack-client inference chat-completion --message "hi"
```

```
ChatCompletionResponse(
    completion_message=CompletionMessage(
      content='How can I assist you today?',
      role='assistant',
      stop_reason='end_of_turn',
      tool_calls=[]
    ),
    logprobs=None,
    metrics=[
      Metric(metric='prompt_tokens', value=12.0, unit=None),
      Metric(metric='completion_tokens', value=17.0, unit=None),
      Metric(metric='total_tokens', value=29.0, unit=None)
    ]
)
```

Let’s use our stack

From the Python client

```
import rich
from llama_stack_client import LlamaStackClient

client = LlamaStackClient(base_url="http://0.0.0.0:8321")
models = client.models.list()
rich.print(models)
```

```
[
    Model(
        identifier='meta-llama/Llama-3.2-3B-Instruct',
        metadata={},
        api_model_type='llm',
        provider_id='ollama',
        provider_resource_id='llama3.2:3b-instruct-fp16',
        type='model',
        model_type='llm'
    )
]
```

client.ipynb

```
import rich

response = client.inference.chat_completion(
    model_id="meta-llama/Llama-3.2-3B-Instruct",
    messages=[
        {"role": "system", "content": "You are a friendly a
        {"role": "user", "content": "Write a two-sentence p
    ],
)
rich.print(response)
```

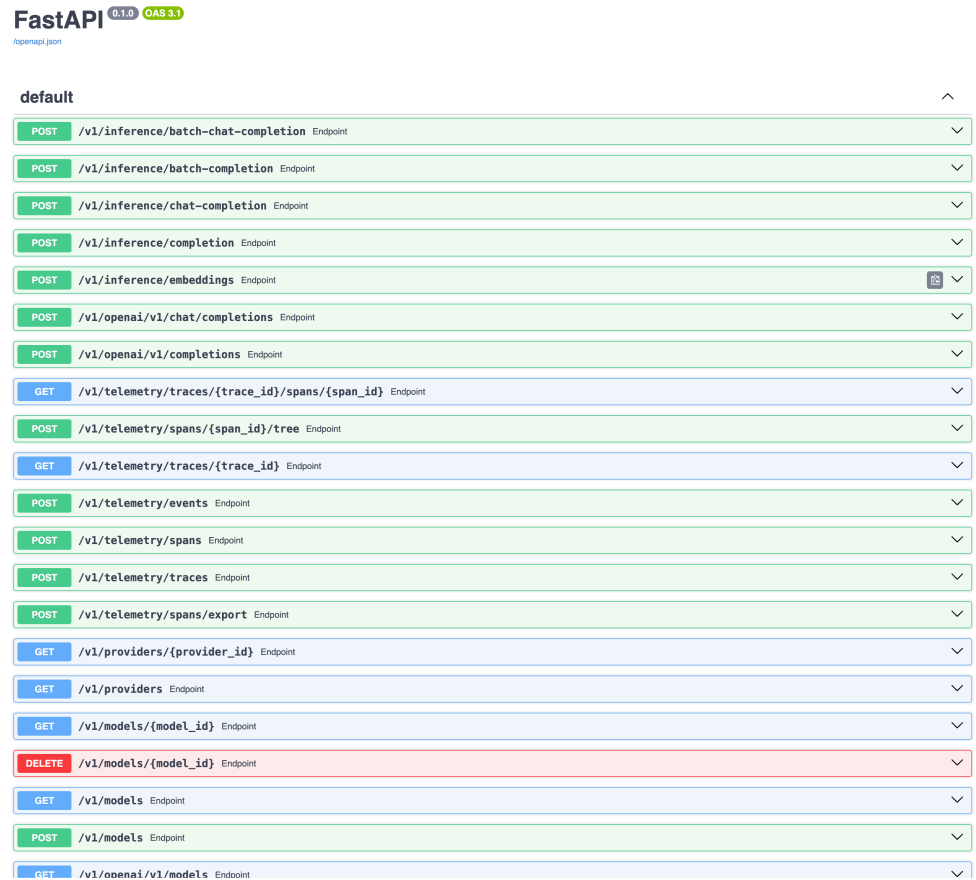
```
ChatCompletionResponse(
    completion_message=CompletionMessage(
        content='Here is a short poem about a llama:\n\nWit
        role='assistant',
        stop_reason='end_of_turn',
        tool_calls=[]
    ),
    logprobs=None,
    metrics=[
        Metric(metric='prompt_tokens', value=30.0, unit=Non
        Metric(metric='completion_tokens', value=37.0, unit
        Metric(metric='total_tokens', value=67.0, unit=None
    ]
)
```

Let’s use our stack

From the REST API

http://localhost:8321/docs

```
curl -X POST http://localhost:8321/v1/inference/chat
  -H "Content-Type: application/json" \
  -d '{"model_id": "meta-llama/Llama-3.2-3B-Instruct
```



 Let’s break our stack

Add stuff to our stack

Let's brake some things …

- Add an OpenAI provider in the provider section

```
- provider_id: openai
  provider_type: remote::openai
  config:
    api_key: ${env.OPENAI_API_KEY:}
```

- Add an OpenAI model in the model section

```
- model_id: openai/gpt-4o
  provider_id: openai
  provider_model_id: openai/gpt-4o
```

Add stuff to our stack

Let's brake more things …

- Add a tool runtime provider

```
$ llama stack list-providers | grep -i tool
| tool_runtime   | inline::rag-runtime             | blobfile,chardet,pypdf,tqdm,numpy,scikit-learn,scipy,nltk,sentencepi
| tool_runtime   | remote::bing-search             | requests
| tool_runtime   | remote::brave-search            | requests
| tool_runtime   | remote::model-context-protocol  | mcp
| tool_runtime   | remote::tavily-search           | requests
| tool_runtime   | remote::wolfram-alpha           | requests
```

Add stuff to our stack

Let's brake even more things …

- Add the agents Api

```
version: "2"
image_name: masterclass_minimal
apis:
  - inference
  - telemetry
  - agents
providers:
[ ... ]
```

The providers registry

The providers registry

Different providers have different dependencies

```
  File "/Users/diego/src/ai-masterclass-2025/masterclass_minimal/lib/python3.12/site-packages/llama_stack/distribution/s
    impls = await resolve_impls(run_config, provider_registry or get_provider_registry(run_config), dist_registry)
            ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Users/diego/src/ai-masterclass-2025/masterclass_minimal/lib/python3.12/site-packages/llama_stack/distribution/r
    return await instantiate_providers(sorted_providers, router_apis, dist_registry)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Users/diego/src/ai-masterclass-2025/masterclass_minimal/lib/python3.12/site-packages/llama_stack/distribution/r
    impl = await instantiate_provider(provider, deps, inner_impls, dist_registry)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Users/diego/src/ai-masterclass-2025/masterclass_minimal/lib/python3.12/site-packages/llama_stack/distribution/r
    impl = await fn(*args)
           ^^^^^^^^^^^^^^^
  File "/Users/diego/src/ai-masterclass-2025/masterclass_minimal/lib/python3.12/site-packages/llama_stack/providers/remo
    from .openai import OpenAIInferenceAdapter
  File "/Users/diego/src/ai-masterclass-2025/masterclass_minimal/lib/python3.12/site-packages/llama_stack/providers/remo
    from llama_stack.providers.utils.inference.litellm_openai_mixin import LiteLLMOpenAIMixin
  File "/Users/diego/src/ai-masterclass-2025/masterclass_minimal/lib/python3.12/site-packages/llama_stack/providers/util
    import litellm
ModuleNotFoundError: No module named 'litellm'
```

The providers registry

Different providers have different dependencies

```
  File "/Users/diego/src/ai-masterclass-2025/masterclass_m
    import litellm
ModuleNotFoundError: No module named 'litellm'
```

llama-stack/llama_stack/providers/registry/inference.py

```python
def available_providers() -> list[ProviderSpec]:
    return [
        [ ... ],
        remote_provider_spec(
            api=Api.inference,
            adapter=AdapterSpec(
                adapter_type="openai",
                pip_packages=["litellm"],
                module="llama_stack.providers.remote.
                config_class="llama_stack.providers.r
                provider_data_validator="llama_stack.
            ),
        ),
    ]
```

The providers registry

Apis can depend on other apis

```
Traceback (most recent call last):
  [ ... ]
  File "/Users/diego/.local/share/uv/python/cpython-3.12.7-macos-aarch64-none/lib/python3.12/asyncio/runners.py", line 1
    return runner.run(main)
           ^^^^^^^^^^^^^^^^
  File "/Users/diego/.local/share/uv/python/cpython-3.12.7-macos-aarch64-none/lib/python3.12/asyncio/runners.py", line 1
    return self._loop.run_until_complete(task)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Users/diego/.local/share/uv/python/cpython-3.12.7-macos-aarch64-none/lib/python3.12/asyncio/base_events.py", li
    return future.result()
           ^^^^^^^^^^^^^^^
  File "/Users/diego/src/llama-stack/llama_stack/distribution/stack.py", line 226, in construct_stack
    impls = await resolve_impls(run_config, provider_registry or get_provider_registry(run_config), dist_registry)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Users/diego/src/llama-stack/llama_stack/distribution/resolver.py", line 136, in resolve_impls
    return await instantiate_providers(sorted_providers, router_apis, dist_registry)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Users/diego/src/llama-stack/llama_stack/distribution/resolver.py", line 245, in instantiate_providers
    deps = {a: impls[a] for a in provider.spec.api_dependencies}
                ~~~~~^^^
KeyError: <Api.safety: 'safety'>
```

The providers registry

Apis can depend on other apis

```
Traceback (most recent call last):
  [...]
  File "/Users/diego/src/llama-stack/llama_stack/distribut
    deps = {a: impls[a] for a in provider.spec.api_depende
                ~~~~~^^^
KeyError: <Api.safety: 'safety'>
```

```
cat ~/src/llama-stack/llama_stack/providers/registry/agent
```

llama-stack/llama_stack/providers/registry/agents.py

```python
def available_providers() -> list[ProviderSpec]:
    return [
        InlineProviderSpec(
            api=Api.agents,
            provider_type="inline::meta-reference",
            pip_packages=[
                "matplotlib",
                "pillow",
                "pandas",
                "scikit-learn",
            ]
            + kvstore_dependencies(),
            module="llama_stack.providers.inline.agen
            config_class="llama_stack.providers.inlin
            api_dependencies=[
                Api.inference,
                Api.safety,
                Api.vector_io,
                Api.vector_dbs,
                Api.tool_runtime,
                Api.tool_groups,
            ],
        ),
    ]
```

External Providers

Providers that live outside of the main codebase

Llama Stack supports two types of external providers:

- Remote Providers: Providers that communicate with external services (e.g., cloud APIs)
- Inline Providers: Providers that run locally within the Llama Stack process

```
external_providers_dir: /etc/llama-stack/providers.d/
```

```
adapter:
  adapter_type: custom_ollama
  pip_packages:
  - ollama
  - aiohttp
  config_class: llama_stack_ollama_provider.config.OllamaImplConfig
  module: llama_stack_ollama_provider
api_dependencies: []
optional_api_dependencies: []
```

https://llama-stack.readthedocs.io/en/latest/providers/external.html

Takeaways

Concepts

- Llama Stack is a way to organize your LLM applications
- Providers are the main building blocks of Llama Stack
- You can group providers and resources into distributions

Dev gotchas

- There's a lot going on! Even in our minimal disto
- Providers have their own (pip) dependencies
- Apis can depend on other apis
- The build command specifies the providers list so you can create a virtual environment with the right dependencies

(Optional)
Alternative Dev Env Setup

Alternative Env Setup

Single virtualenv

- Our repo has a pyproject.toml file for `uv`

```
cd ~/src
git clone https://github.com/trustyai-explainability/ai-mas
git clone https://github.com/meta-llama/llama-stack.git # t
```

```
cat pyproject.toml
```

```
uv sync
source .venv/bin/activate
```

- You can then incrementally update your venv via `--print-deps` and `uv add`

```
llama stack build \
  --config distributions/masterclass-minimal/build.yaml \
  --image-type venv \
  --print-deps
```

```
uv add [ ... ]
```

- As you `uv add` stuff, your `pyproject.toml` will update itself
- The llama-stack dependency is configured a local path and editable

```
1   [project]
2   name = "ai-masterclass-2025"
3   version = "0.1.0"
4   description = "AI Masterclass 2025 (for the dev setup
5   readme = "README.md"
6   requires-python = "≥3.12"
7   dependencies = [
8       "llama-stack",
9   ]
10
11  [tool.uv.sources]
12  llama-stack = { path = "../llama-stack", editable = t
```

Alternative Env Setup

VSCode / Cursor integration

- Our repo also has a `launch.json` file for the VSCode debugger
- Your can use this to run the stack with the debugger attached, and set breakpoints in the stack code

```
cat .vscode/launch.json
```

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "LS debug -- minimal-run.yaml",
      "type": "python",
      "request": "launch",
      "module": "llama_stack.cli.llama",
      "args": ["stack", "run", "distributions/masterc
      "console": "integratedTerminal",
      "justMyCode": false,
      "cwd": "${workspaceFolder}",
      "envFile": "${workspaceFolder}/.env",
      "env": {
        "PYTHONPATH": "${workspaceFolder}"
      }
    },
    {
      "name": "LS debug -- agents-run.yaml",
      "type": "python",
      "request": "launch",
      "module": "llama_stack.cli.llama",
      "args": ["stack", "run", "distributions/masterc
      "console": "integratedTerminal",
      "justMyCode": false,
      "cwd": "${workspaceFolder}",
      "envFile": "${workspaceFolder}/.env",
```

 Implement RAG from scratch

Implement RAG from scratch

- Update or create a new distribution
  - masterclass-agents/run.yaml
- Register a vector database from the client
- Implement a basic RAG pipeline in a notebook
  - Insert a few documents
  - Query them
  - Generate response with context
- (No Agents for now)

Implement RAG from scratch

```
vector_db_id = "my-vector-db"
client.vector_dbs.register(
    vector_db_id=vector_db_id,
    embedding_model="all-MiniLM-L6-v2",
    embedding_dimension=384,
)
```

```
client.tool_runtime.rag_tool.insert(
    documents=documents,
    vector_db_id=vector_db_id,
    chunk_size_in_tokens=512,
)
```

```
client.tool_runtime.rag_tool.query(
    content=prompt, vector_db_ids=[vector_db_id]
)
```

```
db_response = client.vector_io.query(
    vector_db_id=vector_db_id,
    query=prompt,
)
```

```
f"""
Please answer the given query using the context below.

QUERY:
{prompt}

CONTEXT:
{prompt_context}
"""
```

```
response = client.inference.chat_completion(
    messages=messages,
    model_id="meta-llama/Llama-3.2-3B-Instruct",
)
```

Implement RAG from scratch (Agent version)

```
rag_agent = Agent(
    client,
    model="meta-llama/Llama-3.2-3B-Instruct",
    instructions="You are a helpful assistant",
    tools = [
        {
          "name": "builtin::rag/knowledge_search",
          "args" : {
            "vector_db_ids": [vector_db_id],
          }
        }
    ],
)
```

rag.ipynb

Agents

What is an agent?

Some definitions

From Hugging Face's Agents Course:

> *An Agent is a system that leverages an AI model to interact with its environment in order to achieve a user-defined objective. It combines reasoning, planning, and the execution of actions (often via external tools) to fulfill tasks.*

From Anthropic's Building Effective Agents:

> - *Workflows are systems where LLMs and tools are orchestrated through predefined code paths.*
> - *Agents, on the other hand, are systems where LLMs dynamically direct their own processes and tool usage, maintaining control over how they accomplish tasks.*

Tools

From Hugging Face's Agents Course

- LLMs can only receive text inputs and generate text outputs. They have no way to call tools on their own.
- When we talk about providing tools to an Agent, we mean teaching the LLM about the existence of these tools and instructing it to generate text-based invocations when needed.
- From the user's perspective, it appears as if the LLM directly interacted with the tool, but in reality, it was the Agent that handled the entire execution process in the background.
- How do we give tools to an LLM?

Agents on Llama Stack

Basic concepts

- Agent Configuration
  - Model
  - Instructions
  - Tools
  - Shields
- Session
  - conversation thread with an agent
- Turn
  - a single interaction within a session
- Step
  - individual actions within a turn (inference, tool execution, shield calls, etc)

```
agent = Agent(
    llama_stack_client,
    model="my-model",
    instructions="You are a helpful assistant that can use
    tools=["my-tool", "my-other-tool"],
    input_shields=["guardrails::my-filter"],
    output_shields=["guardrails::my-filter"],
)
```

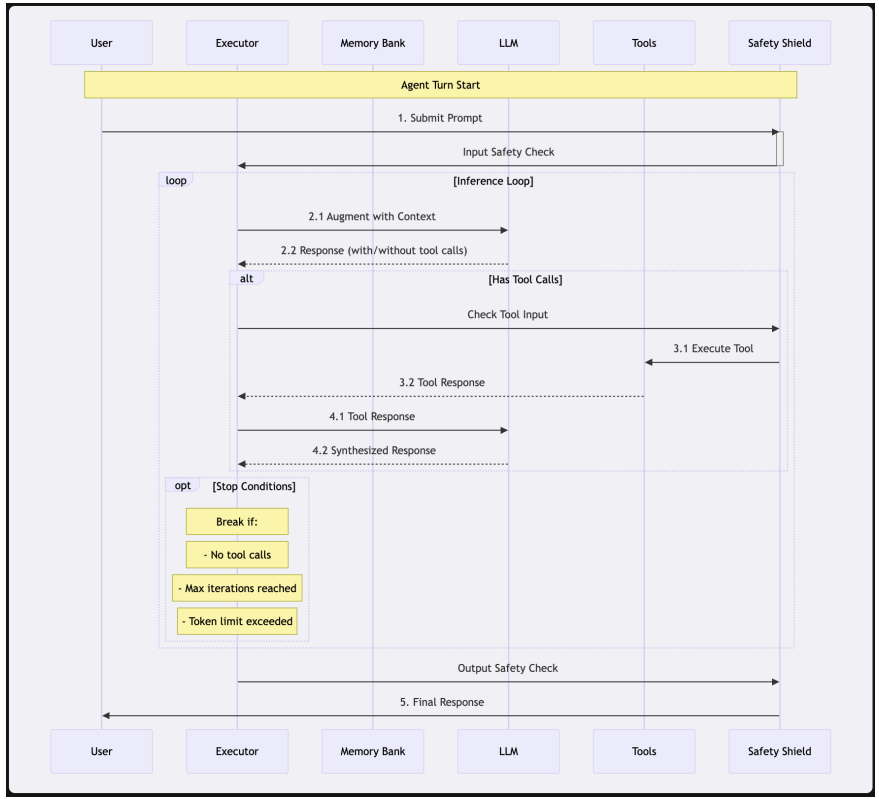
```
session_id = agent.create_session("test-session")
```

```
response = agent.create_turn(
  messages=[{"role": "user", "content": prompt}],
  session_id=session_id,
)
```

Agents on Llama Stack

Execution model

1. Initial Safety Check
2. Inference Loop
   - Augment with context
   - Generate a response, potentially with tool calls and safety checks
   - Repeat
3. Final Safety Check



Execution model, details

llama_stack/providers/inline/agents/meta_reference/agent_instance.py

```
async def run(self, session_id, turn_id, input_messages, sampling_params, stream, documents):
    # 1. Input Safety Checks
    if len(self.input_shields) > 0:
        # Run input safety shields
        async for res in self.run_multiple_shields_wrapper( ... ):
            yield res

    # 2. Main Agent Execution
    async for res in self._run( ... ):
        if isinstance(res, CompletionMessage):
            final_response = res
            break
        else:
            yield res

    # 3. Output Safety Checks
    if len(self.output_shields) > 0:
        # Run output safety shields
        async for res in self.run_multiple_shields_wrapper( ... ):
            yield res
```

 An agent that can find its location

An agent that can find its location

A tool to get a user's location

```python
from llama_stack_client.lib.agents.client_tool import client_tool

@client_tool
def get_location(query: str = "location"):
    """
    Provide the location upon request.

    :param query: The query from user
    :returns: Information about user location
    """
    import geocoder

    try:
        g = geocoder.ip('me')
        if g.ok:
            return f"Your current location is: {g.city}, {g.state}, {g.country}"
        else:
            return "Unable to determine your location"
    except Exception as e:
        return f"Error getting location: {str(e)}"
```

An agent that can find its location

Create an agent to use the `get_location` tool

```
agent = Agent(
    client,
    model="meta-llama/Llama-3.2-3B-Instruct",
    instructions="You are a helpful assistant.",
    tools=[get_location],
)
```

```
user_prompts = ["Where am I?"]
session_id = agent.create_session("test-session")
for prompt in user_prompts:
    rich.print(f"User> {prompt}")

    response = agent.create_turn(
        messages=[{"role": "user", "content": prompt}],
        session_id=session_id,
        stream=True,
    )
    for log in AgentEventLogger().log(response):
        log.print()
```

tool.ipynb

Supported Models

Verify your `identifier` and `provider_resource_id`

```
$ llama-stack-client models list

Available Models
```

model_type	identifier	provider_resource_id	metadata	provider_id
llm	meta-llama/Llama-3.2-3B-Instruct	llama3.2:3b-instruct-fp16		ollama

```
Total models: 1
```

- When using an agent with tools, the prompt gets modified depending on the model family.
- Supported models
- llama_stack/providers/remote/inference/ollama/models.py

 An agent that can search the web

An agent that can search the web

```
agent = Agent(
    client,
    model="meta-llama/Llama-3.2-3B-Instruct",
    instructions="You are a helpful assistant.",
    tools=["builtin::websearch"],
)
```

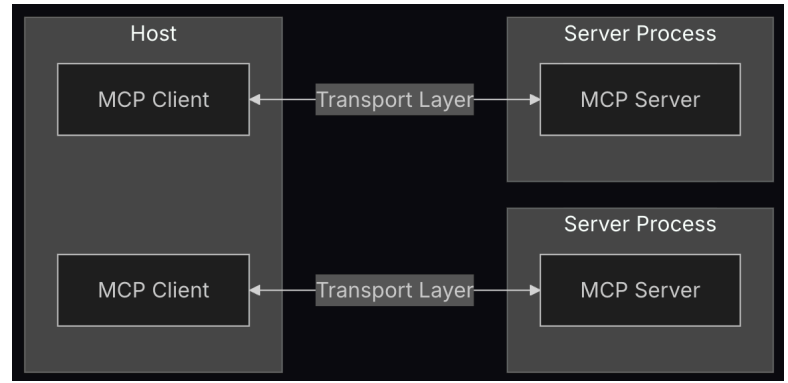
- Give your previous agent a tool to search the web
- Combine it with the `get_location` tool
- Use it to provide answers that are relevant to the user's location
- Pick from the Brave API or the Tavily API (https://tavily.com/)

 An agent that can read files

Model Context Protocol

https://modelcontextprotocol.io/introduction

> MCP is an open protocol that standardizes how applications provide context to LLMs.



- Hosts are LLM applications that initiate connections
- Clients maintain 1:1 connections with servers, inside the host application
- Servers provide context, tools, and prompts to clients
- Transport
  - Stdio
  - HTTP/SSE

Model Context Protocol

Spin up a local MCP server

```
npx -y supergateway --port 8000 --stdio 'npx -y @modelcontextprotocol/server-filesystem my-local-files'
```

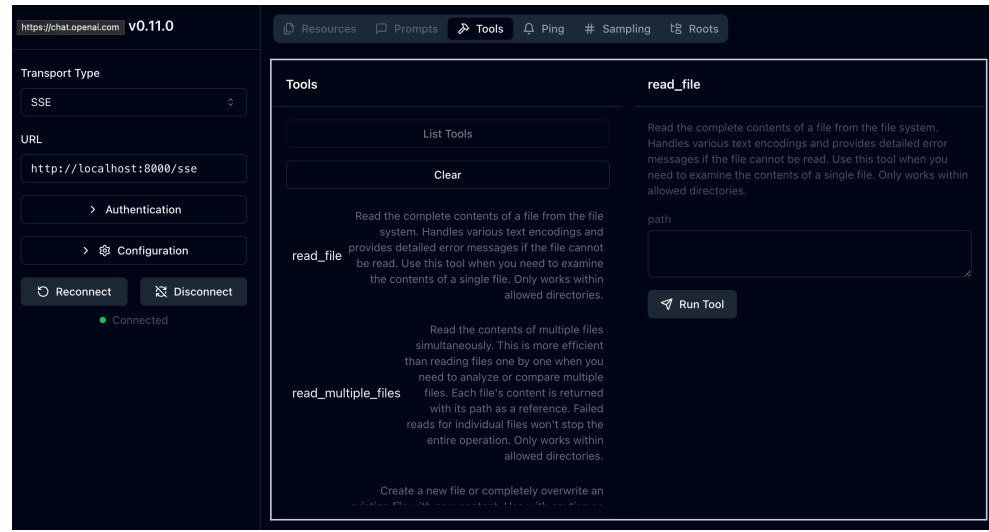
```
[supergateway] Starting...
[supergateway] Supergateway is supported by Supermachine (hosted MCPs) - https://supermachine.ai
[supergateway]   - outputTransport: sse
[supergateway]   - Headers: (none)
[supergateway]   - port: 8000
[supergateway]   - stdio: npx -y @modelcontextprotocol/server-filesystem my-local-files
[supergateway]   - ssePath: /sse
[supergateway]   - messagePath: /message
[supergateway]   - CORS: disabled
[supergateway]   - Health endpoints: (none)
[supergateway] Listening on port 8000
[supergateway] SSE endpoint: http://localhost:8000/sse
[supergateway] POST messages: http://localhost:8000/message
[supergateway] Child stderr: Secure MCP Filesystem Server running on stdio

[supergateway] Child stderr: Allowed directories: [ '/Users/diego/src/ai-masterclass-2025/my-local-files' ]
```

Model Context Protocol

Inspect the MCP server

```
npx -y @modelcontextprotocol/inspector
Starting MCP inspector...
⚙️ Proxy server listening on port 6277
🔍 MCP Inspector is up and running at http://127.0.0
```



An agent that can read files

```
agent = Agent(
    client,
    model="meta-llama/Llama-3.2-3B-Instruct",
    instructions="You are a helpful assistant.",
    tools=["mcp::filesystem"],
)
```

- Create a local directory with your files for testing
- Setup the filesystem MCP server
- Add a MCP provider to your stack
- Create an MCP agent

An agent that can read files

mcp.ipynb

 A “reasoning and acting” (ReAct) agent

A “reasoning and acting” (ReAct) agent

ReAct agents are more flexible and can handle more complex tasks

llama_stack_client/lib/agents/react/prompts.py

```
You are an expert assistant who can solve any task using tool calls.
You will be given a task to solve as best you can.
To do so, you have been given access to the following tools: <<tool_names>>

You must always respond in the following JSON format:
{
    "thought": $THOUGHT_PROCESS,
    "action": {
        "tool_name": $TOOL_NAME,
        "tool_params": $TOOL_PARAMS
    },
    "answer": $ANSWER
}
[ ... ]
This Thought/Action/Observation can repeat N times, you should take several steps when needed.
[ ... ]
You can use the result of the previous action as input for the next action.
```

A “reasoning and acting” (ReAct) agent

ReAct agents are more flexible and can handle more complex tasks

Learn more about ReAct agents:

- https://www.ibm.com/think/topics/react-agent
- ReAct: Synergizing Reasoning and Acting in Language Models

A “reasoning and acting” (ReAct) agent

```
from llama_stack_client.lib.agents.react.agent import ReActAgent
```

- Compare a basic agent with a chain of prompts vs a ReAct agent
- Should be able to ask "Are there any weather-related risks in my area?"
  - figure out it first needs to call the `get_location` tool;
  - and then use the `builtin::websearch` tool.

A “reasoning and acting” (ReAct) agent

react.ipynb

Thank You!