0.3 SETUP `simple-watsonx-enviroment`

Now we'll set up your **watsonx.ai sandbox**: a clean Python 3.11 + Jupyter environment that knows how to talk to Granite / Llama models hosted on IBM watsonx.ai.

You can run it **locally** (virtualenv) or via **Docker** with minimal fuss.

GOAL

By the end of this lab you will:

- Have the **`simple-watsonx-enviroment`** repo cloned.
- Provide **IBM Cloud credentials** via a `.env` file.
- Run `notebooks/watsonx_quickstart.ipynb` and generate text with a Granite model.
- Understand how this environment relates to the **RAG accelerator** you'll use on Day 2–3.

REPOSITORY OVERVIEW

The repo layout looks like:

```
simple-watsonx-enviroment/
├── Dockerfile
├── Makefile
├── pyproject.toml
├── .env.sample
├── notebooks/
│   └── watsonx_quickstart.ipynb
└── scripts/
    ├── mac/
    ├── ubuntu/
    └── windows/
```

Key components:

- **`Dockerfile`** Builds a container with:
  - Python 3.11.
  - Jupyter.
  - `ibm-watsonx-ai` SDK.
  - `langchain-ibm` for LLM integration.
- **`Makefile`** Offers shortcuts like:
  - `make install` – local venv + Jupyter kernel.
  - `make build-container` – build Docker image.

STEP 1 – CLONE THE REPOSITORY

From your main workshop folder:

```
cd ~/projects/watsonx-workshop   # or your path
git clone https://github.com/ruslanmv/simple-watsonx-enviroment.git
cd simple-watsonx-enviroment
```

You now have both env repos side by side.

STEP 2 – CONFIGURE `.env` (CREDENTIALS)

This is the most important step: teaching the environment how to authenticate to watsonx.ai.

1. Copy the sample file:

```
cp .env.sample .env
```

2. Edit `.env` with your IBM Cloud details:

```
# Preferred variables
IBM_CLOUD_API_KEY=your_api_key_here
IBM_CLOUD_URL=https://us-south.ml.cloud.ibm.com
IBM_CLOUD_PROJECT_ID=your_project_id_here

# Compatibility aliases (optional)
WATSONX_APIKEY=${IBM_CLOUD_API_KEY}
WATSONX_URL=${IBM_CLOUD_URL}
PROJECT_ID=${IBM_CLOUD_PROJECT_ID}
```

Where to find these values:

- **IBM_CLOUD_API_KEY**
  - IBM Cloud console → Manage → Access (IAM) → API keys.
- **IBM_CLOUD_URL**

STEP 3 – CHOOSE SETUP PATH

OPTION A – LOCAL (VIRTUALENV)

From the repo root:

```
make install
```

This will:

- Create a virtual environment.
- Install Python dependencies from `pyproject.toml`.
- Register a Jupyter kernel, e.g. **“Python 3.11 (watsonx-env)”**.

Start Jupyter:

```
jupyter notebook
```

Then choose the **watsonx-env** kernel when opening notebooks.

OPTION B – DOCKER (RECOMMENDED FOR TEAM CONSISTENCY)

From the repo root:

```
make build-container
```

STEP 4 – RUN `watsonx_quickstart.ipynb`

Time to confirm that credentials + environment are correct.

1. Open Jupyter (local or container).
2. Navigate to `notebooks/`.
3. Open `watsonx_quickstart.ipynb`.
4. Run the cells in order.

A typical pattern inside the notebook looks like:

```
import os
from dotenv import load_dotenv
from ibm_watsonx_ai import APIClient, Credentials
from ibm_watsonx_ai.foundation_models import ModelInference
from ibm_watsonx_ai.metanames import GenTextParamsMetaNames as GenParams

load_dotenv()

api_key = os.getenv("IBM_CLOUD_API_KEY") or os.getenv("WATSONX_APIKEY")
url = os.getenv("IBM_CLOUD_URL") or os.getenv("WATSONX_URL")
project_id = os.getenv("IBM_CLOUD_PROJECT_ID") or os.getenv("PROJECT_ID")

credentials = Credentials(url=url, api_key=api_key)
client = APIClient(credentials=credentials, project_id=project_id)

model_id = "ibm/granite-13b-instruct-v2"
prompt = "Write a short story about a robot who wants to be a chef."

params = {
    GenParams.DECODING_METHOD: "greedy",
    GenParams.MAX_NEW_TOKENS: 200
```

If everything is configured correctly, you'll see model output printed in the notebook.

OPTIONAL: LANGCHAIN INTEGRATION

If you prefer LangChain style:

```
from langchain_ibm import WatsonxLLM
from dotenv import load_dotenv
import os

load_dotenv()
api_key = os.getenv("IBM_CLOUD_API_KEY") or os.getenv("WATSONX_APIKEY")
url = os.getenv("IBM_CLOUD_URL") or os.getenv("WATSONX_URL")
project_id = os.getenv("IBM_CLOUD_PROJECT_ID") or os.getenv("PROJECT_ID")

llm = WatsonxLLM(
    model_id="ibm/granite-13b-instruct-v2",
    url=url,
    apikey=api_key,
    project_id=project_id,
    params={"decoding_method": "greedy", "max_new_tokens": 128},
)

print(llm.invoke("Give me 3 study tips for Python."))
```

We’ll build on this pattern in later labs.

CONNECTION TO THE `accelerator/` PROJECT

The **accelerator** inside `watsonx-workshop/accelerator/` is where you'll build a **production-like RAG service**:

- **Core RAG logic**:
  - `rag/retriever.py`
  - `rag/pipeline.py`
  - `rag/prompt.py`
- **API**:
  - `service/api.py` – FastAPI app exposing `POST /ask`.
  - `service/deps.py` – holds configuration (URL, API key, project, index names).
- **Tools**:
  - `tools/chunk.py`, `tools/extract.py`, `tools/embed_index.py`, `tools/eval_small.py`
- **UI**:
  - `ui/app.py` – Streamlit front-end.

The patterns you used in `watsonx_quickstart.ipynb`:

REFERENCE NOTEBOOKS IN `labs-src/` AND `accelerator/assets/notebook/`

Once your environment is stable, it's worth quickly skimming some reference notebooks:

RAG & VECTOR DB EXAMPLES (`labs-src/`)

- **Elasticsearch + LangChain** `use-watsonx-elasticsearch-and-langchain-to-answer-questions-rag.ipynb`
- **Elasticsearch Python SDK** `use-watsonx-and-elasticsearch-python-sdk-to-answer-questions-rag.ipynb`
- **Chroma + LangChain** `use-watsonx-chroma-and-langchain-to-answer-questions-rag.ipynb`

These will inspire your implementation of:

- RAG pipelines in Day 2 labs.
- `retriever.py` & `pipeline.py` in the accelerator.

ACCELERATOR NOTEBOOKS (`accelerator/assets/notebook/`)

- Ingestion & indexing:
  - `Process_and_Ingest_Data_into_Vector_DB.ipynb`

TROUBLESHOOTING

401 / 403 – AUTHENTICATION ERRORS

- Verify:
  - `IBM_CLOUD_API_KEY` is correct.
  - You pasted the whole key (no trailing spaces).
  - You're using the correct `IBM_CLOUD_URL` for your region.
  - The project ID is valid and you have access.

"PROJECT NOT FOUND" / 404

- Double-check the **Project ID** in the watsonx.ai UI.
- Ensure you're using the right region and project/space type.

`.env` NOT LOADING

- Make sure `.env` is in the repo root (same folder as `Makefile`, `Dockerfile`).
- Ensure the notebook calls `load_dotenv()` at the top.
- If running via Docker, confirm `--env-file .env` is passed.

JUPYTER KERNEL MISSING

- Re-run:

CHECKLIST

Before moving to the final Day 0 step:

- ✅ `simple-watsonx-enviroment` cloned.
- ✅ `.env` configured with:
  - API key
  - URL
  - Project/space ID
- ✅ Dependencies installed (local venv or Docker image).
- ✅ `watsonx_quickstart.ipynb` runs and returns a Granite response.
- ✅ You know where the `accelerator/` project is and can open its notebooks.

Next up: we'll run a **combined verification** of both environments.