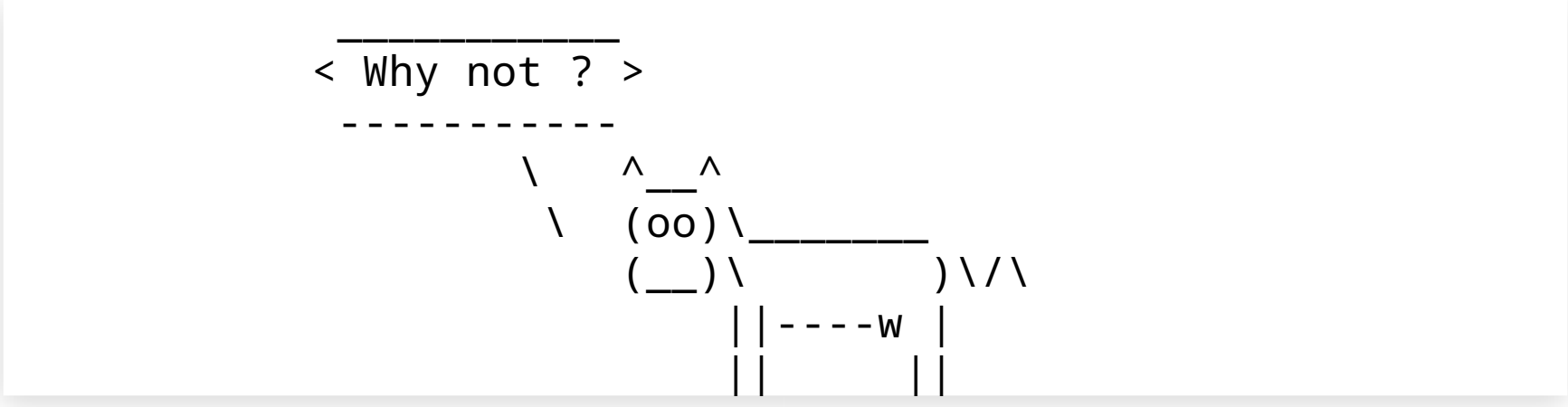
Asking a local LLM about my Ansible playbooks



@ CFGMGMTCAMP | FEBRUARY 2026

with David Moreau-Simard

$ cat dmsimard.yml

```
- name: David Moreau-Simard
  hosts:
    - fosstodon.org/@rfc2549
    - fosstodon.org/@ara
  vars:
    job: Architect at OVHcloud
    profile: ['sysadmin', 'dev/ops', 'CI/CD', 'SRE']
  roles:
    - Ansible user since version ~1.8 (2014?)
    - Created ara in 2016 to make my life easier
    - Was in the Ansible community team at Red Hat
    - Probably use Ansible for too many things
    - Part time contributor (between dev ops and dad ops)
```

Ever used them ?

ChatGPT

Claude

Gemini

Copilot

Ansible VS Code Extension

Red Hat

Ansible Lightspeed

with IBM watsonx Code Assistant

- Big, capable models
- Online in the cloud
- Probably not free
- Security, privacy, confidentiality ?
- Digital sovereignty

What about hosting models locally ?


AMD

https://frame.work/ca/en/desktop

Before we start

- Runtimes
- Registries
- Models
- Integrations

RUNTIMES

https://github.com/ggml-org/llama.cpp

https://lmstudio.ai/

https://ollama.com/

REGISTRIES

https://huggingface.co/

https://lmstudio.ai/models

https://ollama.com/library

MODELS

https://huggingface.co/openai/gpt-oss-120b

https://huggingface.co/Qwen/Qwen3-Coder-30B-A3B-Instruct

https://huggingface.co/zai-org/GLM-4.7-Flash

https://huggingface.co/mistralai/Devstral-Small-2-24B-Instruct-2512

INTEGRATIONS

https://www.home-assistant.io/

https://n8n.io/

~~clawdbot~~ ~~moltbot~~ https://github.com/openclaw/openclaw

What about **local** models ?

They know about Ansible

Explain like I'm five:
why might a system administrator be interested in using ansible ?

`mistralai/devstral-small-2-2512`

Imagine you have 100 toy boxes, each with different toys inside.
Every day, you need to open each box, take out some toys, put in new ones, and make sure everything is clean.

Now, imagine instead of doing this by hand for every single box, you could write a little note that says:
"Take out the red car, put in the blue truck, and wipe the dust off."

Then, you give this same note to a robot who does all 100 boxes for you—fast, without mistakes, and while you sip lemonade.

That's Ansible.
It lets the sysadmin write one set of instructions (playbooks) that automatically handles many servers (toy boxes), saving time and headaches. 🚀

8.45 tok/sec • 154 tokens • 0.40s to first token • Stop reason: EOS Token Found

They know about Python, Jinja and YAML, too.

Write a short and simple python script that uses jinja to render yaml.

qwen/qwen3-coder-30b

python Copy

```python
#!/usr/bin/env python3
from jinja2 import Template
import yaml

template_str = """
database:
  host: {{ db_host }}
  port: {{ db_port }}
  name: {{ db_name }}
  credentials:
    username: {{ username }}
    password: {{ password }}
"""

data = {
    'db_host': 'localhost',
    'db_port': 5432,
    'db_name': 'myapp',
    'username': 'admin',
    'password': 'secret'
}

template = Template(template_str)
rendered = template.render(data)
print(yaml.dump(yaml.safe_load(rendered), default_flow_style=False))
```

This script creates a Jinja2 template with YAML structure, renders it with sample data, and outputs properly formatted YAML.

58.27 tok/sec • 165 tokens • 0.09s to first token • Stop reason: EOS Token Found

Can they troubleshoot playbooks ?

```
 ------------------------
< PLAY [Failed playbook] >
 ------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||


 ------------------------
< TASK [Gathering Facts] >
 ------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||

ok: [localhost]
 -------------
< TASK [fail] >
 -------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||

[ERROR]: Task failed: Action failed: This is a failed playbook
Origin: /home/rfc2549/dev/git/ansible-community/ara/tests/integration/failed.yaml:8:7
6   hosts: localhost
7   tasks:
8     - fail:
        ^ column 7

fatal: [localhost]: FAILED! => {"changed": false, "msg": "This is a failed playbook"}
 ------------
< PLAY RECAP >
 ------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||

localhost                  : ok=1    changed=0    unreachable=0    failed=1    skipped=0    rescued=0    ignored=0
```

failed.txt

What's going on here ?

`zai-org/glm-4.7-flash`

✓ Chosen context injection strategy: 'inject-full-content'. All content can fit into the context

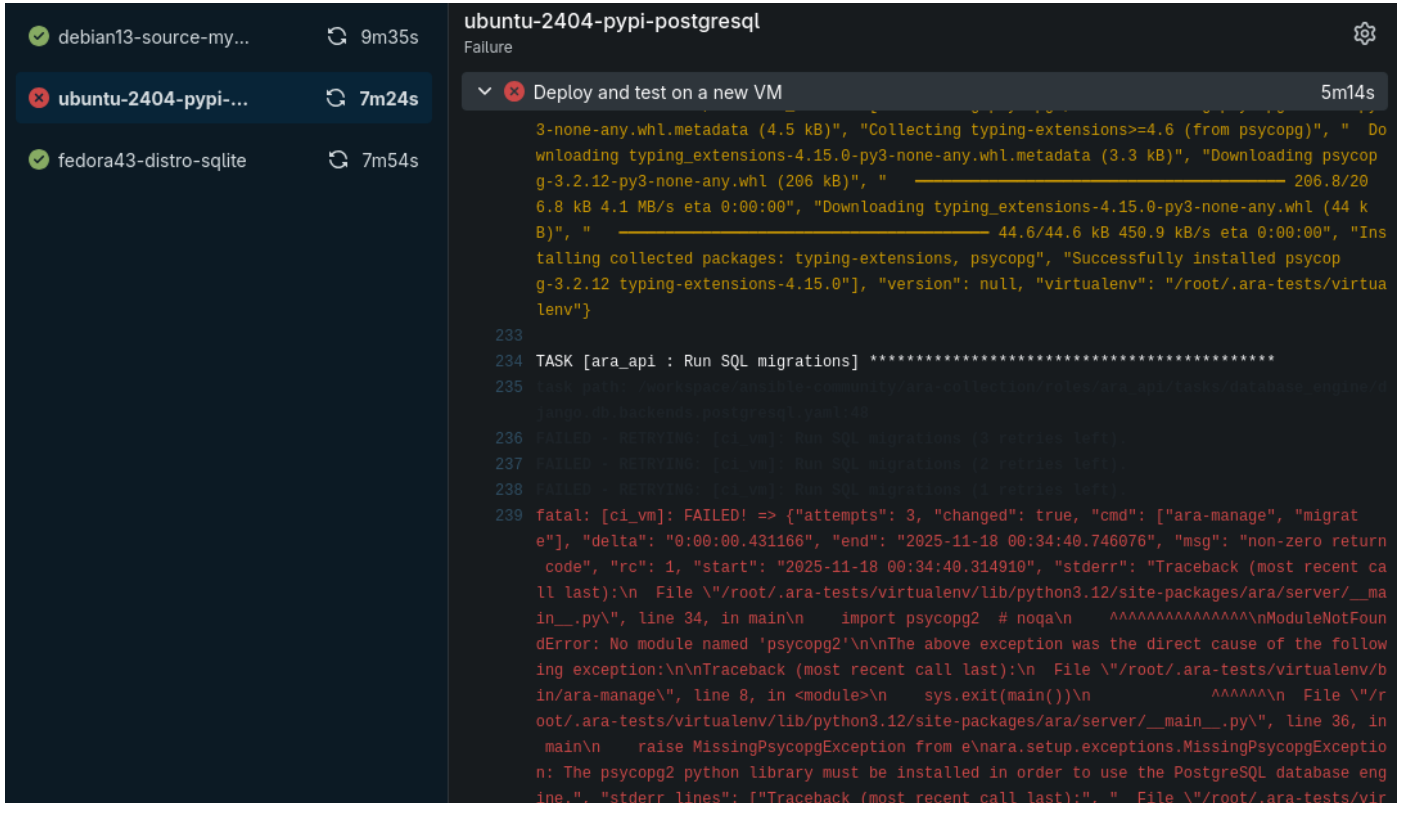
✓ Done predicting

› Thought for 15.30 seconds

Your playbook gathered facts successfully, but then it explicitly called the `fail` module on line 8 to stop execution with the message "This is a failed playbook". It's not a crash; it's a feature.

48.35 tok/sec • 791 tokens • 0.56s to first token • Stop reason: EOS Token Found

What about something more **complicated** ?



https://codeberg.org/ansible-community/ara-collection/pulls/81

https://codeberg.org/ansible-community/ara-collection/actions/runs/38/jobs/1/attempt/1

https://ara.readthedocs.io/en/latest/contributing.html#ci-jobs-and-how-they-work

long-failed.txt

Can you find out what's happening here ?

gpt-oss-120b

✓ Chosen context injection strategy: 'inject-full-content'. All content can fit into the context

✓ Done predicting

› Thought for 2.13 seconds

What the playbook is doing

1. **Spin-up a CI VM** – an Ubuntu 24.04 instance (`ci_vm` at 172.25.0.90).
2. **Install build tools & Python dev stack** (gcc, make, python3-dev, ...).
3. **Pull a PostgreSQL container** with `podman`.
4. **Deploy the ARA API role** – create a virtualenv, install `ara[server]` from PyPI, set up config files and cron jobs.
5. **Install DB-engine dependencies** – for PostgreSQL it installs the OS packages (`postgresql` , `postgresql-server-dev-16` , etc.) and then runs:

   yaml Copy

   ```yaml
   - name: Ensure psycopg(3) is installed
     pip:
       name: psycopg
   ```

6. **Run migrations** with `ara-manage migrate` .

Where it blows up

The migration step fails three times with this traceback:

```
ModuleNotFoundError: No module named 'psycopg2'
...
MissingPsycopgException: The psycopg2 python library must be installed in order to use the PostgreSQL database
engine.
```

- You have *psycopg 3* (`pip install psycopg`) in the virtualenv.
- ARA expects the legacy *psycopg2* module name.

Hence the "missing psycopg2" error, even though a PostgreSQL driver is present.

How to fix it

1. **Install the legacy package** (or its binary wheel) instead of/alongside `psycopg`:

   yaml Copy

   ```yaml
   - name: Ensure psycopg2 is installed
     pip:
       name: "psycopg2-binary>=2.9,<3"
   ```

   The `-binary` variant avoids needing the PostgreSQL client headers at build time.
2. **Or pin ARA to a version that supports psycopg 3** (if such a release exists). Check the ARA changelog; as of 1.7.3 it still expects `psycopg2`.
3. **Update the role variables** (if the role allows overriding the driver name) – some forks let you set `ara_api_postgresql_packages` to include `python3-psycopg2`. Add that to your vars file.
4. **Re-run the playbook** after adjusting the pip task; migrations should now succeed and the API will start listening on port 8000.

Quick sanity check

Run this manually inside the VM's virtualenv:

bash Copy

```bash
source /root/.ara-tests/virtualenv/bin/activate
python -c "import psycopg2, psycopg"
```

If `psycopg2` raises an ImportError while `psycopg` works, you've reproduced the exact problem.

TL;DR

The playbook installs `psycopg` (psycopg 3) but ARA still imports `psycopg2`. Install `psycopg2-binary` (or downgrade to a psycopg2-compatible version of ARA) and the migration step will stop throwing "MissingPsycopgException".

Why did the PostgreSQL container refuse to play nice? Because it heard there was no psycopg in the room! 🚀

28.53 tok/sec • 845 tokens • 318.60s to first token • Stop reason: EOS Token Found

Ouch !

```
CONTEXT: 60680 / 100000
```

Ouch !

```
CONTEXT: 60680 / 100000
```

Can this be **easier** ?

ARA Records Ansible and makes it **easier** to understand and troubleshoot.

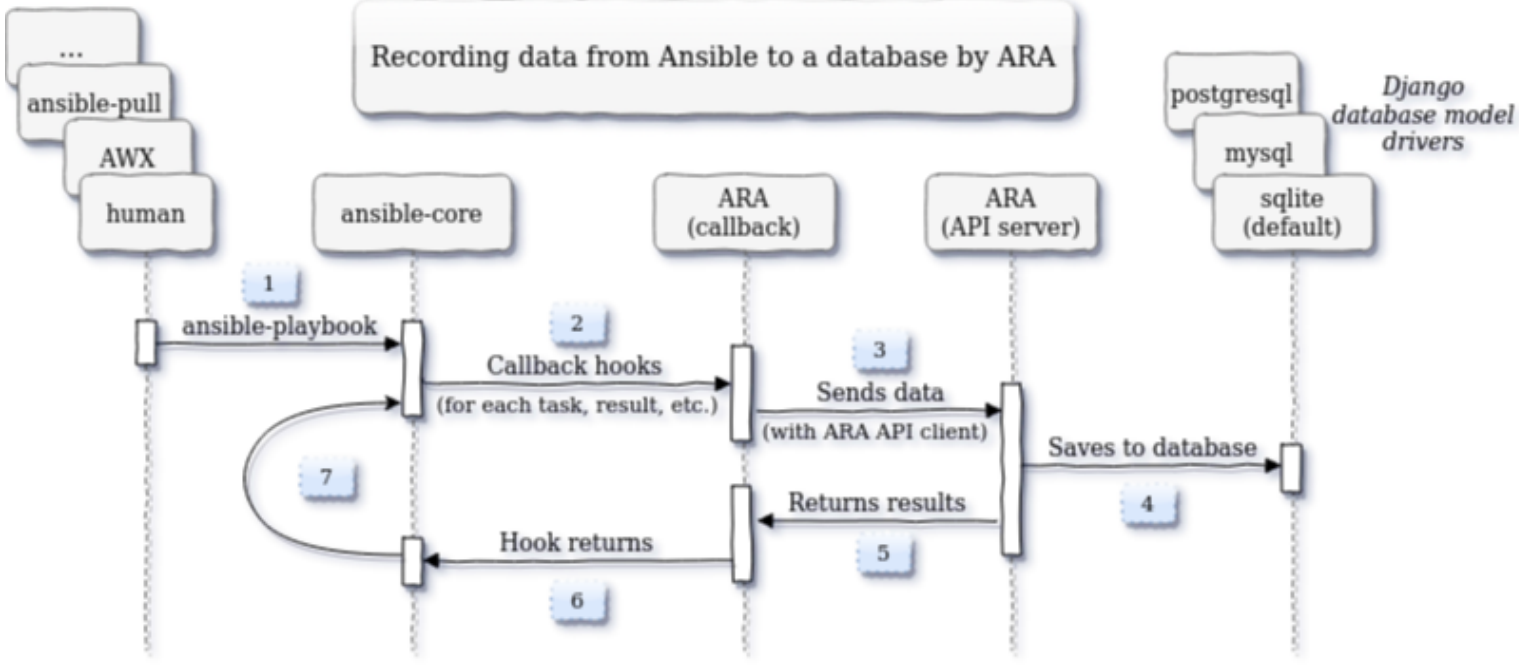
ara

It is another recursive acronym with a focus on simplicity.

GETTING STARTED

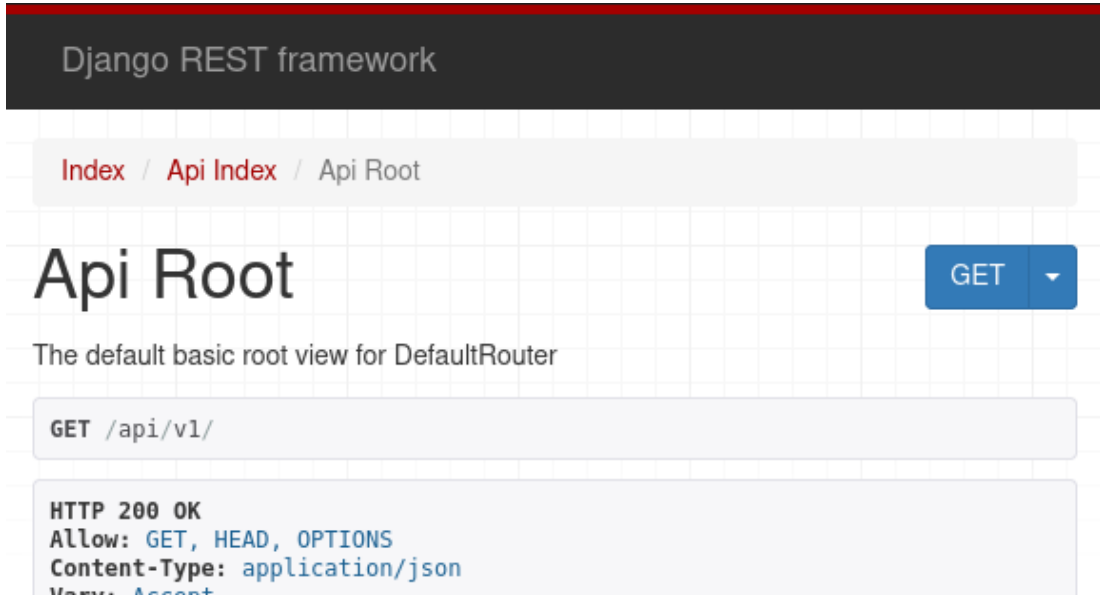
```
# Install ansible with ara (including API server dependencies)
python3 -m pip install --user ansible "ara[server]"

# Configure Ansible to enable ara
export ANSIBLE_CALLBACK_PLUGINS="$(python3 -m ara.setup.callback_plugins)"

# Run an Ansible playbook as usual
ansible-playbook playbook.yml

# Check out the CLI
ara playbook list
ara host list

# or the UI at http://127.0.0.1:8000
ara-manage runserver
```

How ARA Records Ansible

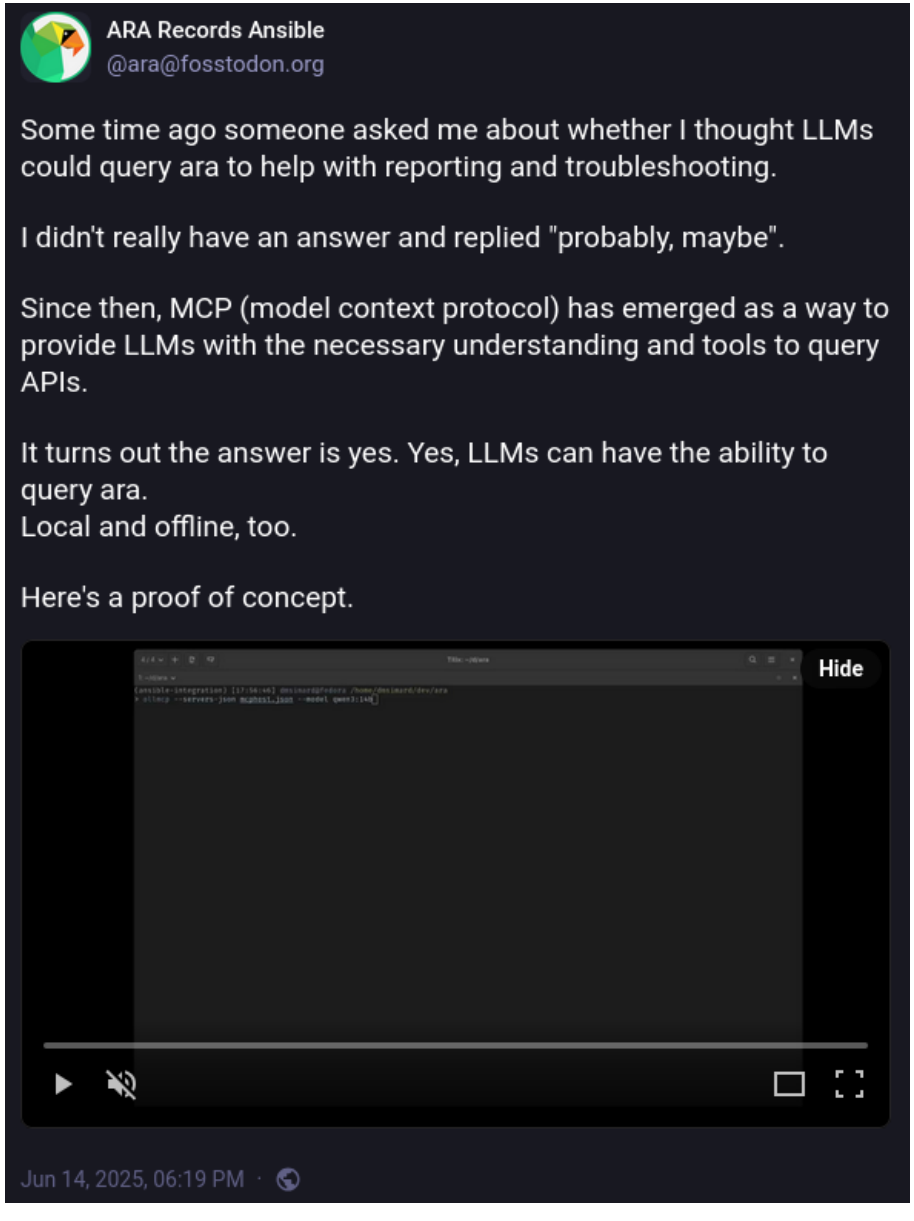


It has an API

https://demo.recordsansible.org/api/v1/

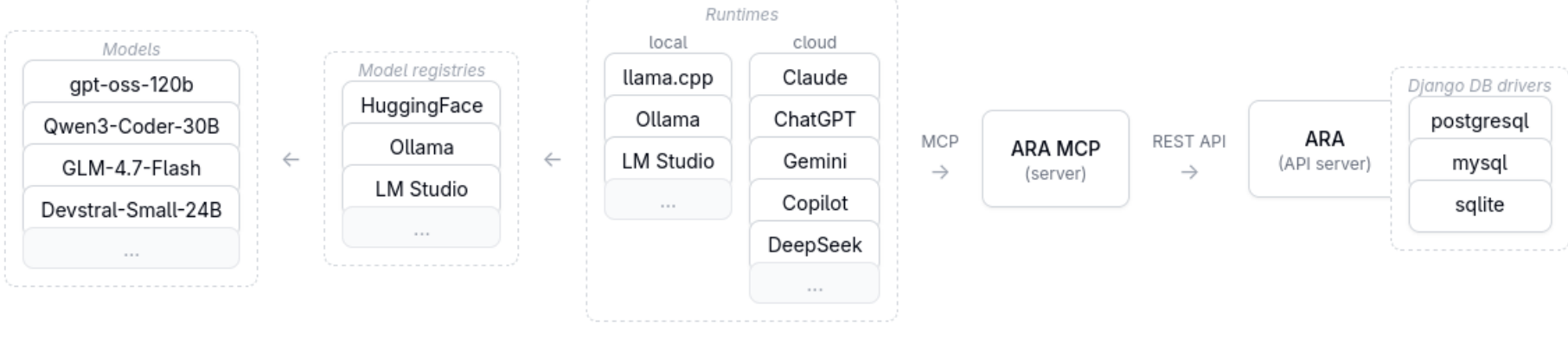


What if we gave models access to this API ?



https://fosstodon.org/@ara/114684037878435782

How MCP works

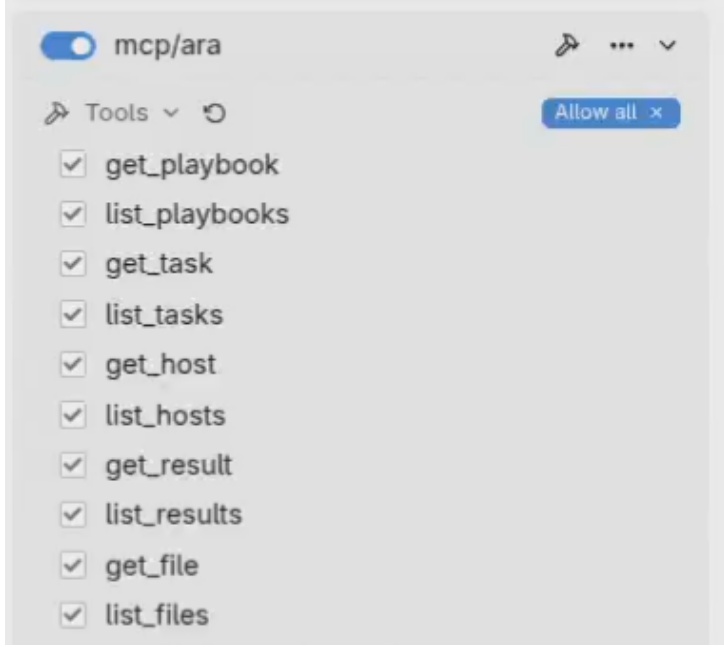


Models ← Model Registries ← Runtimes → MCP Server → ARA API → Database

MCP configuration

```
{
  "mcpServers": {
    "ara": {
      "command": "/path/to/python",
      "args": [
        "/path/to/ara/contrib/mcp/mcp_server.py"
      ],
      "env": {
        "ARA_API_SERVER": "https://demo.recordsansible.org"
      }
    }
  }
}
```

MCP Tools


mcp/ara
Tools
Allow all
get_playbook
list_playbooks
get_task
list_tasks
get_host
list_hosts
get_result
list_results
get_file
list_files

Demo: Using an MCP server for ara.

Next steps

- Wrap it up, submit to /contrib
- Consider usage in ara's own CI pipelines
- MCP vs Skill ?
- Other use cases ?

THANK YOU !

QUESTIONS ?

For more information:

- https://codeberg.com/ansible-community/ara
- https://ara.recordsansible.org/blog
- https://ara.recordsansible.org/presentations
- https://fosstodon.org/@ara