

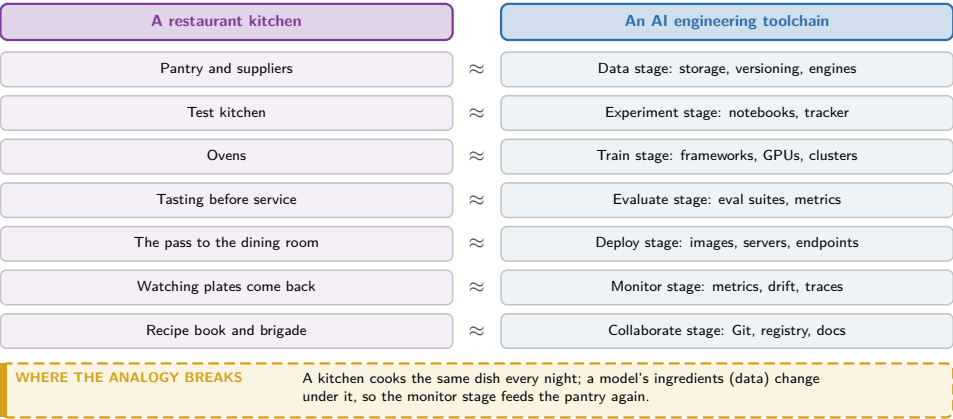


The AI Engineer's Toolbox Map

Tools and platforms | One recommended default and its alternatives for each stage (data, experiment, train, evaluate, deploy, monitor, collaborate), the artefact each stage hands to the next, coverage of the main tools and a stack by team size (opinionated, as of 2026; the full catalogue by task lives in resources 08)

#Toolbox #Stack #MLOps #AsOf2026

→ Why a map



A professional kitchen is a good model of an ML toolchain

Know first	Level	Where
What each MLOps stage is for	must	mlops 00-05
Python environment with uv	must	programming 08
Training loop, checkpoints	helpful	programming 06
Containers	helpful	sheet 05 of this folder

Prerequisites: what this map assumes

📘 INFO

How to read this sheet. Each stage gets one **default** (start here, change only for a reason) and **alternatives** keyed to the reason. resources 08 lists every tool by task; this sheet chooses, and says which artefact links the stages.

📖 One default per stage

Stage	Default	Alternatives, and when	Hand-off
Data	Parquet on object storage, queried with Polars or DuckDB; DVC or Delta for versions	pandas (< 1 GB); Spark/Databricks (TB, many writers); Ray Data (GPU or unstructured); dbt (SQL transforms); Label Studio (labels)	versioned Parquet / Delta table
Experiment	JupyterLab or VS Code + MLflow tracking; Optuna for search	Weights & Biases (hosted UI, reports); ClearML, Comet; marimo (reproducible notebooks); Ray Tune (search on a cluster)	run ID with params, metrics, artefacts
Train	PyTorch with Lightning or the Hugging Face Trainer on one GPU node; scikit-learn / XGBoost / LightGBM for tables	torchrun + FSDP or DeepSpeed (multi-node); Ray Train (Python orchestration); managed training jobs (SageMaker AI, Vertex AI)	checkpoint + registered model version
Evaluate	pytest suite with scikit-learn metrics; Inspect or promptfoo for LLM apps	lm-evaluation-harness (benchmarks); DeepEval, ragas (RAG); Evidently (data and model reports)	eval report, pass/fail gate
Deploy	FastAPI in a Docker image (classic models); vLLM server (LLMs)	BentoML (packaging); KServe (Kubernetes); Ray Serve (composed Python); Triton (multi-framework GPU); managed endpoints	image digest + endpoint
Monitor	Prometheus + Grafana (service and GPU); Evidently (drift); Langfuse (LLM traces)	Arize Phoenix; NannyML (performance without labels); Datadog or cloud monitoring	dashboards, alerts, retrain trigger
Collaborate	Git + GitHub (pull requests, Actions), uv.lock, model registry	GitLab; Hugging Face Hub (sharing models); Quarto (reports); DVC remotes (data sharing)	reviewed commit, release tag

Defaults and alternatives by stage (opinionated, as of 2026). The last column is what the next stage receives

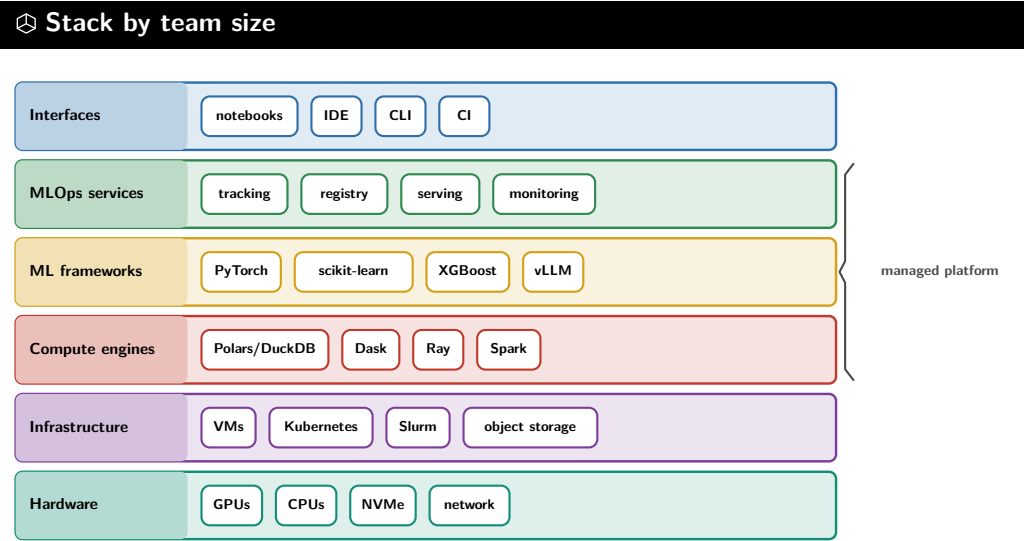
Tool	Data	Experiment	Train	Evaluate	Deploy	Monitor	Collaborate
DATA ENGINES							
Polars	✓	✗	✗	⚠	✗	✗	✗
DuckDB	✓	✗	✗	⚠	✗	✗	✗
DVC data and pipeline versioning	✓	⚠	✗	✗	✗	✗	✓
TRACKING AND REGISTRY							
MLflow open source, Linux Foundation	✗	✓	✗	✓	⚠	⚠	✓
Weights & Biases SaaS, Weave for LLM traces	✗	✓	✗	✓	✗	⚠	✓
COMPUTE							
PyTorch + Lightning	✗	⚠	✓	⚠	✗	✗	✗
Ray Data, Train, Tune, Serve	✓	⚠	✓	⚠	✓	✗	✗
Databricks (Spark) managed lakehouse platform	✓	✓	⚠	⚠	✓	⚠	✓
RUN AND WATCH							
Docker	✗	⚠	⚠	✗	✓	✗	✓
KServe	✗	✗	✗	✗	✓	⚠	✗
Prometheus + Grafana	✗	✗	✗	✗	✗	✓	⚠
Evidently	⚠	✗	✗	✓	✗	✓	✗
Langfuse LLM traces, prompts, evals	✗	⚠	✗	✓	✗	✓	⚠
GitHub	✗	✗	✗	⚠	⚠	✗	✓

✓ yes / full ⚠ partial ✗ no / none

Which stages each common tool covers (yes = core use, partial = possible but not its strength). No single tool covers all seven

 **Glue beats features**

Pick tools that hand off through **open artefacts**: Parquet/Delta for data, MLflow run IDs and model URIs, OCI images by digest, OpenTelemetry traces, Prometheus metrics. Then any single tool can be swapped without touching its neighbours.



The layers of an ML platform, bottom to top. Managed platforms (sheet 09) sell the middle three as one product

Layer	Solo / 1-3 people	One product team	Platform for many teams
Compute	one GPU VM or workstation; Colab/Kaggle	managed training jobs or a small Ray cluster	Kubernetes + Kueue, KubeRay, or Slurm
Data	Parquet + Polars/DuckDB	lakehouse table (Delta/Iceberg) + DuckDB or Spark	Databricks or Spark on Kubernetes, catalogue
Tracking	MLflow local file store	MLflow server or W&B team	MLflow/W&B with SSO, registry with approvals
Serving	FastAPI or vLLM in Docker on one VM	managed endpoint or KServe	KServe/Ray Serve with canary, autoscaling
Monitoring	logs + nvidia-smi	Prometheus/Grafana + Evidently	DCGM exporter, SLOs, on-call
People	nobody runs a platform	part of one engineer	a dedicated platform team

Three stack sizes; move right only when the left one hurts

Worked example: choosing and sizing

Y

A three-person team builds a churn model and an LLM support assistant on AWS. Pick a default per stage, then size one year of tracked artefact storage.

Given: R = runs per week = 50; A = artefacts per run (checkpoint, plots, sample predictions) = 200 MB; W = weeks per year = 52; k = share of runs kept after cleanup = 10 %; Raw storage S = R × A × W; kept storage S_k = S × k

Y

Step 1: Data?

Parquet in S3, Polars and DuckDB, DVC for versions
tables of a few GB; no cluster needed

T

Y

Step 2: Experiment and train?

VS Code + MLflow server; LightGBM for churn; managed fine-tuning only if
prompting fails
one tracker for both projects

T

Y

Step 3: Evaluate?

pytest metrics for churn; promptfoo suite for the assistant, run in CI
a regression fails the pull request

T

Y

Step 4: Deploy and monitor?

FastAPI image on a managed container service; LLM through a provider API;
Langfuse traces; Evidently weekly drift report
no GPU serving to operate

T

Y

Step 5: Raw storage?

$S = 50 \times 200 \text{ MB} \times 52 = 520,000 \text{ MB} \approx 520 \text{ GB}$
every run kept

T

Y

Step 6: After cleanup?

$S_k = 520 \text{ GB} \times 0.10 \approx 52 \text{ GB}$
keep registered models and the best runs; delete the rest after 30 days

T

A one-VM-scale stack (S3 + Polars/DuckDB + MLflow + FastAPI + Langfuse + Evidently), about 520 GB of artefacts a year without cleanup and about 52 GB with a 10 % retention rule

Check: If checkpoints grow to 2 GB (a fine-tuned 1 B model in bf16), raw storage is $50 \times 2 \text{ GB} \times 52 = 5.2 \text{ TB}$; retention rules become mandatory

T

A three-person team picks its stack and sizes artefact storage

Data and compute

1. Polars
2. DuckDB
3. Ray Data
4. pandas on data > RAM

Training and tuning

5. Lightning / HF Trainer
6. Optuna
7. Ray Train

Serving and ops

8. vLLM
9. KServe
10. OTEL GenAI traces
11. TorchServe (limited maintenance)

Developer tools

12. uv
13. Dev containers
14. marimo
15. Unpinned requirements.txt

● no change ▲ moved in (toward Adopt) ▼ moved out ○ new

One team's radar (opinion, not a standard): adopt = default now, trial = pilot on one project, assess = prototype, hold = do not start new work with it

Myths and recap

MYTH OR FACT?

"One end-to-end platform covers every stage best"

MYTH: Every platform is strong in some stages and average in others
The tool matrix above has no full row; platforms win on integration, not on each stage.

MYTH OR FACT?

"Pick the tool with the most features"

MYTH: Pick the one your team can operate and leave
Operations time and exit cost dominate after the first month.

MYTH OR FACT?

"Open source is free"

MYTH: Licence-free, not operations-free
Self-hosted MLflow, Kubeflow or Langfuse need upgrades, backups, SSO and on-call.

Three tool-choice beliefs that cost money

Before adding a tool to the stack

✓ baseline / mandatory □ recommended – verify in your context

- ✓ It reads and writes an open format the neighbours already use
- ✓ Someone is named to upgrade and back it up
- ✓ Its licence fits commercial use (check AGPL, BSL, SSPL)
- **SHOULD** SSO and audit logs exist if more than one team uses it
- **COULD** A removal plan is written: what replaces it, what data moves

Remember

01 Default first, alternative for a stated reason
data size, team size, cloud, licence

02 Stages connect through artefacts
Parquet, run ID, checkpoint, image digest, dashboard

03 No tool covers all seven stages well
integration is what platforms sell

04 Stack size follows team size
VM → product team → platform team

05 Count operations hours, not licence cost
upgrades, backups, on-call

Q1 Default data stack for a few GB of tables?

A Parquet on object storage, queried with Polars or DuckDB, versioned with DVC or Delta.

Q2 Default tracker and registry?

A MLflow (open source); W&B when a hosted UI and reports matter more than self-hosting.

Q3 When leave one-node training for Ray Train or torchrun multi-node?

A When the model or the data no longer fits one node's GPUs or time budget.

Q4 Hand-off artefact from deploy to monitor?

A The image digest and endpoint: monitoring must know exactly which build serves.

Q5 Default for serving an open-weight LLM?

A A vLLM server (OpenAI-compatible API) in a container.

Q6 Which tool shows LLM traces, prompt versions and evals?

A Langfuse (open source) or Arize Phoenix; LangSmith as SaaS.

Q7 Raw artefact storage for 50 runs/week × 200 MB over 52 weeks?

A 520 GB; about 52 GB if only 10 % of runs are kept.

Q8 What is the hold ring of a tech radar?

A Do not start new work with the tool; migrate away when convenient.

Q9 Why is TorchServe on hold?

A The project entered limited maintenance in 2025; prefer vLLM, Triton, KServe runtimes or Ray Serve.