Personas & what they can do

Who works in OpenShift AI, and what each role can do with the platform.

⬇ Download as PDF

RHOAI is multi-tenant and role-based: a **cluster admin** stands up and governs the platform, while **data scientists, engineers, and app developers** self-serve inside isolated *Data Science Projects*. The roles below overlap in practice — treat them as hats one person may wear, not rigid job titles.

Data Scientist

Explores data and builds/experiments with models. The primary day-to-day user.

What they can do:

- Create and work inside **Data Science Projects** (isolated namespaces)
- Launch **Workbenches** — Jupyter/VS Code/RStudio with curated PyTorch/CUDA images
- Attach a **GPU** to a workbench via a hardware profile; add persistent storage (PVC)
- Connect to data — S3/object buckets, databases, via data connections
- Experiment, train, and **fine-tune** models (e.g. LoRA/QLoRA)
- Track runs, params, and metrics with **MLflow**
- Build and run **pipelines** visually (Elyra) or with the `kfp` SDK
- Register models to the **Model Registry** and deploy them for testing

---

Key features: Workbenches · Data connections · MLflow · Pipelines · Model Registry

ML / AI Engineer

Turns experiments into robust, scalable training and serving.

What they can do:

- Run **distributed / multi-GPU training** with Kubeflow **Trainer v2** (`TrainJob`) and **Ray**
- Queue and prioritize GPU jobs fairly with **Kueue**
- Optimize/quantize models for inference
- Deploy and scale **model serving** with **KServe** (vLLM for LLMs)
- Deploy pre-optimized **NVIDIA NIM** microservices
- Manage model versions and promotion in the Model Registry
- Expose inference endpoints (REST/gRPC) for applications

Key features: Trainer v2 · Ray · Kueue · KServe/vLLM · NIM · Model Registry

MLOps / Platform Engineer

Automates and operationalizes the ML lifecycle for reliability and repeatability.

What they can do:

- Build automated **Data Science Pipelines** (train → eval → register → deploy) and schedule them
- Set up pipeline servers backed by object storage
- Manage the **Model Registry** lifecycle — staging, promotion, lineage
- Configure **Kueue** quotas, cluster/local queues, and cohorts
- Wire up model **monitoring** and automated retraining
- Integrate with Git and CI/CD for GitOps-style ML delivery

Key features: Pipelines · Model Registry · Kueue · Monitoring · GitOps

GenAI / LLM Engineer

Builds LLM-based systems — RAG, agents, tool calling — on top of served models rather than training from scratch. The role RHOAI 3.x grew to accommodate.

What they can do:

- Serve LLMs with **KServe + the vLLM runtime**; pick quantisation (FP8 / MXFP4 / AWQ) and trade weights against KV-cache for context length
- Build **RAG** systems — embeddings, vector store, retrieval, grounded generation
- Compose **agents** on **Llama Stack** — tool calling, MCP tools, sessions
- Apply input/output safety with the **Guardrails** orchestrator
- Benchmark and regression-test models with **LMEval**
- Right-size a model to the available GPU — the practical ceiling is VRAM, not parameters

---

Key features: Llama Stack · KServe/vLLM · Guardrails · LMEval · Model Registry

Data Engineer

Provides clean, reliable data and features for training and inference.

What they can do:

- Configure **data connections** (S3/object storage, databases)
- Build data-prep and ingestion steps as pipeline stages
- Define and serve reusable features with the **Feast** feature store
- Manage datasets on cluster storage (ODF: block, file, object)

Key features: Data connections · Feast · Pipelines · ODF storage

Application Developer

Builds applications that consume AI — without managing the ML themselves.

What they can do:

- Call deployed **inference endpoints** (REST/gRPC) from applications
- Build **RAG / GenAI apps** against served LLMs or NIM endpoints
- Consume models as stable, versioned services managed by others
- Integrate model APIs into OpenShift-hosted apps

Key features: KServe endpoints · NIM · Model serving APIs

Cluster / OpenShift Administrator

Installs, secures, and operates the platform itself.

What they can do:

- Install and upgrade the **RHOAI operator** and configure the `DataScienceCluster` (which components are enabled)
- Enable **GPUs**: Node Feature Discovery, NVIDIA GPU Operator, hardware/accelerator profiles
- Provide **storage** (ODF) and object storage for workbenches and pipelines
- Manage **multi-tenancy**, RBAC, and Data Science Project isolation
- Configure platform dependencies — Service Mesh, cert-manager, the Data Science Gateway
- Set resource **quotas** and integrate NVIDIA NIM/NGC
- Handle disconnected/air-gapped installs and security/compliance

---

Key features: Operator/DSC · GPU Operator · ODF · RBAC · Service Mesh · Quotas

AI Product / Model Owner (Governance)

Owns model value, risk, and approval — accountable for what ships.

What they can do:

- Review model performance, **bias, drift, and explainability** with **TrustyAI**
- Govern model promotion and approvals via the Model Registry
- Track which models are deployed, where, and by whom
- Apply responsible-AI and compliance guardrails before production

---

Key features: TrustyAI · Model Registry · Monitoring · Governance

How well RHOAI fits each role

The personas above are what the platform *hosts*. They are not served equally well — RHOAI's centre of gravity is the model lifecycle, so roles whose job *is* that lifecycle get the most out of it.

Role	Core of the job on RHOAI	Fit
ML / AI Engineer	Training → registry → serving → monitoring	★★★
MLOps / Platform Engineer	Automation, quotas, GitOps delivery	★★★
Cluster Administrator	GPUs, storage, multi-tenancy, operators	★★★
GenAI / LLM Engineer	RAG, agents, guardrails, LLM eval	★★★
Data Scientist	Notebooks, experiments, fine-tuning	★★★
AI Product / Model Owner	Bias, drift, approval gates	★★☆
Data Engineer	Orchestration, features, distributed ETL	★★☆
Application Developer	Consumes endpoints; doesn't operate the platform	★☆☆

Data science is well served, but at the *“here is your notebook and your GPU”* level — the platform does not help you do statistics. Data engineering sits awkwardly in the middle: strong compute and orchestration, no ingestion and no catalogue.

Where RHOAI stops

Worth being explicit about, because these are scope decisions rather than gaps:

- **Analytics engineer / BI** — no dbt, no semantic layer, no BI tool. Feast is the nearest thing, and it is not close. That is Superset/dbt/Looker territory.
- **Business or "citizen" analyst** — no AutoML, no low-code builder; the dashboard is operator-facing.
- **Data platform primitives** — no ingestion, no CDC, no data catalogue or lineage beyond the model registry. RHOAI orchestrates and computes; sourcing and cataloguing data is assumed to exist elsewhere.

So: roughly six roles served well, two partially, and explicitly not the analytics side of the data organisation.

The team shape it assumes

RHOAI assumes a split. A **platform team** operates the cluster — operators, GPUs, quotas, storage, gateway — and **project teams** consume it through Data Science Projects with namespace-scoped RBAC. Multi-tenancy is the mechanism: each team gets a project with its own workbenches, pipeline server, model servers and GPU allocation.

The through-line that justifies the platform is that every persona shares one substrate — same GPUs, same projects, same RBAC, same storage. A feature Feast materialises is the feature KServe serves.

Proven on this cluster

Which labs demonstrate which persona:

Persona	Labs
ML / MLOps Engineer	`pipeline-lab`, `model-serve-demo`, `mlflow`
Platform / Cluster Admin	GPU time-slicing, distributed training, the opp-ai and gpu-02 builds
GenAI / LLM Engineer	`rag-lab`, `agent-lab`, `guardrails-lab`, capstone

Persona	Labs
Model Owner / Governance	`trustyai-lab`, `eval-lab` (LMEval), model registry
Data Engineer	`data-eng-lab` — the one lab in this row

The distribution is itself informative: the work clusters in the platform and GenAI rows, which is the shape of an **AI platform engineer** rather than a data scientist. See Disciplines for why those are different things.

Add a persona by copying a `<div class="persona">…</div>` block with an `<h2 id="…">` — the sidebar updates automatically.