

OpenClaw-RL Tutorial

Complete Setup Guide

Train an LLM from Your Own Conversations
Using Reinforcement Learning

Deploy on RunPod with 3–4× H100 80GB GPUs

Qwen3-4B · GRPO · SGLang · Megatron-LM

Made by Vizuara AI Labs

<https://vizuara.ai>

<https://github.com/VizuaraAILabs/OpenClaw-RL-Tutorial>

Last updated: March 2026

Contents

| | |
|--|-----------|
| 1 Overview | 3 |
| 1.1 How It Works | 3 |
| 1.2 Architecture | 3 |
| 1.3 What You Will Need | 4 |
| 2 Part 1: RunPod Setup | 5 |
| 2.1 Step 1: Create a RunPod Pod | 5 |
| 2.2 Step 2: SSH into Your Pod | 5 |
| 2.3 Step 3: Run the Setup Script | 5 |
| 3 Part 2: Launch the Training Pipeline | 7 |
| 3.1 Step 4: Download the Deploy Script | 7 |
| 3.2 Step 5: Set Your API Key | 7 |
| 3.3 Step 6: Launch the Server | 7 |
| 3.4 Step 7: Verify Everything Works | 7 |
| 3.5 Step 8: Start Chatting | 7 |
| 3.6 Step 9: Run Before/After Benchmarks | 8 |
| 3.7 Step 10: Monitor Training | 8 |
| 4 Part 3: Dashboard Setup | 9 |
| 4.1 Step 11: Start the Metrics Server (on the Pod) | 9 |
| 4.2 Step 12: Configure the Dashboard (on Your Local Machine) | 9 |
| 4.3 Step 13: Run the Dashboard | 9 |
| 4.4 Dashboard Features | 10 |
| 5 Part 4: WhatsApp Integration (Optional) | 11 |
| 5.1 Step 14: Install Node.js 22 | 11 |
| 5.2 Step 15: Install the OpenClaw CLI | 11 |
| 5.3 Step 16: Run the Onboarding Wizard | 11 |
| 5.4 Step 17: Configure the SGLang Provider | 11 |
| 5.5 Step 18: Link Your WhatsApp | 12 |
| 5.6 Step 19: Set Up the Environment | 12 |
| 5.7 Step 20: Start the Gateway | 12 |
| 6 Part 5: Resuming After Pod Restart | 14 |
| 6.1 Step A: Restart the Pod | 14 |
| 6.2 Step B: Update the Pod ID in Three Files | 14 |
| 6.2.1 File 1: Dashboard Environment | 14 |
| 6.2.2 File 2: OpenClaw Config | 14 |
| 6.2.3 File 3: Agent Model Cache (Critical!) | 14 |
| 6.3 Step C: Restart Everything | 15 |
| 7 Training Details | 16 |
| 7.1 Hyperparameters | 16 |
| 7.2 How PRM Scoring Works | 16 |
| 8 Cost Estimate | 17 |
| 9 deploy.sh Command Reference | 17 |

| | |
|--|-----------|
| 10 File Structure | 18 |
| 11 Troubleshooting | 19 |
| 12 Port Reference | 19 |
| 13 Quick Reference: Complete Workflow | 19 |
| 14 Credits | 20 |

1 Overview

This guide walks you through deploying a complete reinforcement learning pipeline that trains a language model from your own conversations in real time. By the end of this guide, you will have:

- A Qwen3-4B model running on RunPod GPUs, learning from your chats via GRPO
- A real-time dashboard showing training metrics, scores, and conversations
- (Optional) A WhatsApp integration so your everyday messages train the model

1.1 How It Works

Training Loop

1. You send a message (via chat client, dashboard, or WhatsApp)
2. The **Rollout GPU** generates a response using SGLang
3. The **PRM GPU** evaluates the response quality (3 independent votes)
4. The **Actor GPUs** train on the scored response using GRPO
5. Updated weights are pushed back to the Rollout GPU
6. The next response is better

Everything runs asynchronously—the model keeps serving while training runs in the background. A training step is triggered after every **32 conversation samples**.

1.2 Architecture

| Component | What It Does | Framework | GPU |
|----------------|--|-------------|---------|
| Rollout | Serves the model via OpenAI-compatible API | SGLang | GPU 2 |
| PRM | Evaluates response quality (3 independent votes) | SGLang | GPU 3 |
| Actor | Trains model weights using scored responses | Megatron-LM | GPU 0–1 |
| Orchestrator | Coordinates all components across GPUs | Ray + Slime | — |
| API Server | Proxies chat, captures conversations, triggers PRM | FastAPI | — |
| Metrics Server | Aggregates training data for the dashboard | FastAPI | — |

Table 1: System components and their roles

| Setup | Actor | Rollout | PRM | Tensor Parallelism |
|---------|--------|---------|-------|--------------------|
| 4× H100 | 2 GPUs | 1 GPU | 1 GPU | TP=2 |
| 3× H100 | 1 GPU | 1 GPU | 1 GPU | TP=1 |

Table 2: GPU allocation by setup

1.3 What You Will Need

| Requirement | Details |
|---------------------|---|
| RunPod account | With access to H100 80GB GPUs |
| GPUs | 3–4× NVIDIA H100 80GB (or equivalent) |
| HuggingFace account | To download Qwen3-4B model weights |
| Local machine | Node.js 18+ (for dashboard), SSH client |
| Cost | ~\$30–45 for a full 3–4 hour session |

No machine learning experience is required. All scripts are ready to run.

2 Part 1: RunPod Setup

2.1 Step 1: Create a RunPod Pod

Log in to <https://runpod.io> and create a new GPU pod with these exact settings:

| Setting | Value |
|-----------------|--|
| GPU | 3–4× NVIDIA H100 80GB |
| Container Image | <code>slimerl/slime:latest</code> |
| Container Disk | 50 GB |
| Network Volume | 100 GB (mount at <code>/workspace</code>) |
| Start Command | <code>bash -c "sleep infinity"</code> |
| HTTP Ports | 30000, 8080, 8265 |

Critical

You **must** use `slimerl/slime:latest` as the container image. It has all dependencies pre-installed (Python 3.12, CUDA 12.9, PyTorch 2.9, SGLang, Ray, Megatron-LM with custom patches). **Do NOT pip install anything**—it will break compiled CUDA extensions.

Critical

You **must** set `bash -c "sleep infinity"` as the Start Command. Without this, the container will crash immediately on startup.

2.2 Step 2: SSH into Your Pod

Once the pod is running, find the SSH connection details in the RunPod dashboard (click “Connect” on your pod).

```
ssh root@<POD_IP> -p <PORT> -i ~/.ssh/id_ed25519
```

Replace `<POD_IP>` and `<PORT>` with the values from RunPod.

Tip

RunPod uses an SSH relay server, so `scp` may not work. To transfer files, use `curl` to download from a URL, or use the RunPod web terminal.

2.3 Step 3: Run the Setup Script

On the pod, download and run the setup script:

```
curl -s0 https://raw.githubusercontent.com/VizuaraAILabs/OpenClaw-RL-Tutorial/main/
      scripts/setup.sh
bash setup.sh
```

This script will:

1. Verify you have 3+ GPUs and the correct Docker image

2. Clone the [OpenClaw-RL](#) repository
3. Download the Qwen3-4B-Thinking-2507 model (~8 GB)
4. Create checkpoint and results directories
5. Generate pre-configured launch scripts for 3-GPU and 4-GPU setups

Wait for: [setup] Setup complete!

3 Part 2: Launch the Training Pipeline

3.1 Step 4: Download the Deploy Script

```
curl -s0 https://raw.githubusercontent.com/VizuaraAILabs/OpenClaw-RL-Tutorial/main/deploy.sh
```

3.2 Step 5: Set Your API Key

Choose any secret key. This is used to authenticate requests to the inference server.

```
export SGLANG_API_KEY="pick-any-secret-key"
```

Warning

Remember this key—you will need it when configuring the dashboard and WhatsApp integration.

3.3 Step 6: Launch the Server

```
bash deploy.sh launch
```

This starts the full pipeline: Ray cluster, SGLang inference, PRM scoring, and Megatron-LM training. Startup takes **3–5 minutes**.

Wait for: [OpenClaw] your model is fired up and ready to roll

3.4 Step 7: Verify Everything Works

```
bash deploy.sh test
```

This runs 4 checks:

1. **Health check** — server is responding
2. **List models** — qwen3-4b is loaded
3. **Chat completion** — model can generate responses
4. **GPU utilization** — GPUs are active

All 4 should show PASS. If any fail, wait another minute and retry.

3.5 Step 8: Start Chatting

```
bash deploy.sh chat
```

This opens an interactive multi-turn chat session. Type your messages and press Enter. Commands:

- quit or exit — end the session

- `new` — start a fresh conversation (new session ID)

Alternatively, use `curl` directly:

```
curl http://localhost:30000/v1/chat/completions \
-H "Content-Type: application/json" \
-H "Authorization: Bearer $SGLANG_API_KEY" \
-d '{"model": "qwen3-4b",
  "messages": [{"role": "user", "content": "Hello!"}],
  "max_tokens": 512}'
```

3.6 Step 9: Run Before/After Benchmarks

Run the benchmark before and after training to measure improvement:

```
# Save pre-training baseline
bash deploy.sh benchmark

# ... have 30-50 conversations to fill training batches ...

# Save post-training results
bash deploy.sh benchmark
```

The benchmark sends 10 standardized prompts and records responses, latency, and token counts. Results are saved to JSONL files in the results directory.

3.7 Step 10: Monitor Training

```
# Watch training logs in real-time
bash deploy.sh monitor

# View the Ray dashboard (open in browser)
# https://<POD_ID>-8265.proxy.runpod.net
```

Training triggers after **32 conversation samples** are collected (one batch).

4 Part 3: Dashboard Setup

The dashboard is a Next.js web app that runs on your **local machine** and connects to the RunPod metrics server.

4.1 Step 11: Start the Metrics Server (on the Pod)

Open a **second terminal** connected to the pod and run:

```
cd /workspace/OpenClaw-RL
pip install -q fastapi unicorn
unicorn pod-server.metrics_server:app \
--host 0.0.0.0 --port 8080
```

Verify it is running:

```
curl http://localhost:8080/health
```

Expected response: {"status": "ok"}

Tip

To run the metrics server in the background so it persists if you close the terminal:

```
nohup unicorn pod-server.metrics_server:app \
--host 0.0.0.0 --port 8080 \
> /tmp/metrics_server.log 2>&1 &
```

4.2 Step 12: Configure the Dashboard (on Your Local Machine)

Clone the tutorial repo and set up the dashboard:

```
git clone https://github.com/VizuaraAILabs/OpenClaw-RL-Tutorial.git
cd OpenClaw-RL-Tutorial/dashboard
cp .env.example .env.local
```

Edit `.env.local` with your pod's URLs. Find your pod ID in the RunPod dashboard URL.

```
RUNPOD_BASE_URL=https://<YOUR-POD-ID>-8080.proxy.runpod.net
SGLANG_BASE_URL=https://<YOUR-POD-ID>-30000.proxy.runpod.net/v1
SGLANG_API_KEY=pick-any-secret-key
MODEL_NAME=qwen3-4b
```

Replace `<YOUR-POD-ID>` with your actual pod ID (e.g., `nz8djvv2khxq0w`).

4.3 Step 13: Run the Dashboard

```
npm install
npm run dev
```

Open <http://localhost:3000> in your browser.

4.4 Dashboard Features

- **Live training metrics** — sample count, batch progress, training steps, average PRM score
- **Score chart** — real-time PRM score visualization (last 50 samples)
- **GPU monitor** — utilization, memory, and temperature for all GPUs
- **Conversation feed** — live stream of chat exchanges with their PRM scores
- **Chat panel** — chat with the model directly from the dashboard
- **Demo mode** — works without a pod connection (generates synthetic data for preview)

5 Part 4: WhatsApp Integration (Optional)

This section connects your WhatsApp to the training pipeline so your everyday conversations contribute to model training. This uses [OpenClaw](#), an open-source WhatsApp AI gateway.

5.1 Step 14: Install Node.js 22

OpenClaw requires Node.js 22. Install it using `fnm` (Fast Node Manager):

```
# Install fnm
curl -fsSL https://fnm.vercel.app/install | bash

# Restart your terminal, then:
fnm install 22
fnm use 22

# Verify
node --version
# Should show v22.x.x
```

Tip

You can also use `nvm` or download Node.js 22 directly from <https://nodejs.org>. Any method works as long as `node --version` shows v22.x.x.

5.2 Step 15: Install the OpenClaw CLI

```
npm install -g openclaw
```

Verify the installation:

```
openclaw --version
```

5.3 Step 16: Run the Onboarding Wizard

```
openclaw onboard
```

Follow the interactive prompts to set up your OpenClaw gateway configuration. This creates the configuration files in `~/.openclaw/`.

5.4 Step 17: Configure the SGLang Provider

Edit your OpenClaw config to point to your RunPod pod. The config file is typically at `~/.openclaw/openclaw.json`.

```
{
  "providers": [
    {
```

```

    "name": "sglang",
    "type": "openai",
    "baseUrl": "https://<YOUR-POD-ID>-30000.proxy.runpod.net/v1",
    "apiKey": "pick-any-secret-key",
    "models": [
        {
            "name": "qwen3-4b",
            "maxTokens": 4096,
            "temperature": 0.6
        }
    ],
    "defaultModel": "sglang:qwen3-4b"
}

```

Replace <YOUR-POD-ID> with your actual RunPod pod ID and use the same API key you set in Step 5.

Tip

A complete example config file is included in the repository at `openclaw-config-example.json`.

5.5 Step 18: Link Your WhatsApp

```
openclaw pair
```

This displays a QR code in your terminal. On your phone:

1. Open WhatsApp
2. Go to **Settings** → **Linked Devices**
3. Tap **Link a Device**
4. Scan the QR code displayed in your terminal

5.6 Step 19: Set Up the Environment

Source the environment script to configure OpenClaw paths:

```
source scripts/openclaw-env.sh
```

Tip

Add this line to your `~/.zshrc` or `~/.bashrc` so it loads automatically on every new terminal:

```
source ~/path/to/openclaw-rl-tutorial/scripts/openclaw-env.sh
```

5.7 Step 20: Start the Gateway

```
openclaw start
```

Messages you send and receive on WhatsApp will now flow through the model on RunPod. Each conversation is automatically captured, PRM-scored, and used for training.

Verify the gateway is running:

```
openclaw gateway status
```

Tip

To view gateway logs in real time:

```
openclaw logs --follow
```

6 Part 5: Resuming After Pod Restart

When you stop and restart your RunPod pod (e.g., to pause billing overnight), the pod ID may change. You must update the pod URL in **three places**.

Critical

This is the most common source of errors. If you skip updating `models.json` (step 2 below), you will get “404 status code (no body)” errors from WhatsApp.

6.1 Step A: Restart the Pod

Go to the RunPod dashboard, find your stopped pod, and click **Start/Resume**. Wait for it to boot. Note the **new pod ID** from the URL.

Warning

Always use **Stop** (not Terminate) to preserve your network volume. Terminate deletes the pod entirely, though the network volume survives separately.

6.2 Step B: Update the Pod ID in Three Files

6.2.1 File 1: Dashboard Environment

Edit `dashboard/.env.local`:

```
RUNPOD_BASE_URL=https://<NEW-POD-ID>-8080.proxy.runpod.net
SGLANG_BASE_URL=https://<NEW-POD-ID>-30000.proxy.runpod.net/v1
```

6.2.2 File 2: OpenClaw Config

Edit `~/.openclaw/openclaw.json` — update the `baseUrl` field:

```
"baseUrl": "https://<NEW-POD-ID>-30000.proxy.runpod.net/v1"
```

6.2.3 File 3: Agent Model Cache (Critical!)

Edit `~/.openclaw/agents/main/agent/models.json` — update the `baseUrl` field.

Critical

OpenClaw caches the model configuration in this separate file. If you only update `openclaw.json`, the agent will still use the old URL from `models.json` and you will get “404 status code (no body)” errors.

You can use `sed` to update it quickly:

```
sed -i '' \
"s|https://[a-z0-9]*-30000.proxy.runpod.net/v1|https://<NEW-POD-ID>-30000.proxy.
runpod.net/v1|g" \
~/.openclaw/agents/main/agent/models.json
```

6.3 Step C: Restart Everything

On the pod:

```
# Start the training pipeline
export SGLANG_API_KEY="your-key"
bash deploy.sh launch

# Start the metrics server (after pipeline is ready)
cd /workspace/OpenClaw-RL
nohup unicorn pod-server.metrics_server:app \
--host 0.0.0.0 --port 8080 \
> /tmp/metrics_server.log 2>&1 &
```

On your local machine:

```
# Start the dashboard
cd dashboard && npm run dev

# Start the WhatsApp gateway
source scripts/openclaw-env.sh
openclaw start
```

7 Training Details

7.1 Hyperparameters

| Parameter | Value |
|---------------------|---|
| Base Model | Qwen3-4B-Thinking-2507 |
| RL Algorithm | GRPO (Group Relative Policy Optimization) |
| Learning Rate | 1e-6 (constant schedule) |
| KL Penalty | 0.02 (low-variance KL) |
| Clip Range | 0.2 / 0.28 |
| Batch Size | 32 samples per training step |
| Max Response Length | 8,192 tokens |
| Context Length | 32,768 tokens |
| PRM Evaluations | 3 independent votes per response |
| Optimizer | Adam (CPU-offloaded, precision-aware) |
| Weight Decay | 0.1 |

Table 3: GRPO training hyperparameters

7.2 How PRM Scoring Works

The Process Reward Model (PRM) is another copy of the same Qwen3-4B model running on a dedicated GPU. For each assistant response:

1. The PRM receives the full conversation context.
2. It generates **3 independent evaluations** ($M=3$).
3. Each evaluation votes **-1** (poor), **0** (neutral), or **1** (good).
4. The majority vote becomes the final score.
5. Scores are normalized to 0.0–1.0 for the dashboard using the formula: $display = (raw+1)/2$.
6. This score becomes the reward signal for GRPO training.

The PRM can judge in two ways:

- **Multi-turn:** reads the user’s follow-up message to infer if the response was helpful
- **Single-turn:** evaluates the response quality directly when there is no follow-up

Additionally, explicit user sentiment (e.g., “thanks!”, “that’s wrong”) overrides the LLM-based PRM via regex pattern matching, providing a direct reward signal.

8 Cost Estimate

| Phase | Time | Cost (4× H100 @ ~\$10/hr) |
|--|----------------|---------------------------|
| Setup + model download | 15 min | ~\$2.50 |
| Server startup | 5 min | ~\$0.80 |
| Fill first training batch (32 samples) | 30–60 min | ~\$5–10 |
| Extended training (50+ conversations) | 2–3 hrs | ~\$20–30 |
| Total for a full session | 3–4 hrs | ~\$30–45 |

Table 4: Estimated costs for a full training session

Tip

Use RunPod’s **Stop** (not Terminate) to keep your network volume between sessions. You will only pay for storage (\$0.10/GB/month). Your model checkpoints and training data persist on the network volume.

9 deploy.sh Command Reference

| Command | What It Does |
|---------------------------------------|---|
| <code>bash deploy.sh setup</code> | Clone repo + download Qwen3-4B model (~8 GB) |
| <code>bash deploy.sh launch</code> | Start the RL training server (3–5 min) |
| <code>bash deploy.sh test</code> | Run 4-step verification |
| <code>bash deploy.sh chat</code> | Interactive multi-turn chat |
| <code>bash deploy.sh benchmark</code> | Run 10 test prompts (before/after comparison) |
| <code>bash deploy.sh monitor</code> | Watch training logs in real-time |
| <code>bash deploy.sh stop</code> | Shut down server + Ray cluster |

Table 5: Available deploy.sh commands

10 File Structure

```
openclaw-rl-tutorial/
|-- deploy.sh                                # One-script deployment orchestrator
|-- openclaw-config-example.json               # Example OpenClaw provider config
|-- LICENSE
|-- README.md
|
|-- pod-server/
|   |-- openclaw_api_server.py                # API proxy + PRM scoring
|   +-+ metrics_server.py                    # Metrics aggregator for dashboard
|
|-- scripts/
|   |-- setup.sh                            # One-command pod setup
|   |-- run_4gpu.sh                         # Launch script (4x H100, TP=2)
|   |-- run_3gpu.sh                         # Launch script (3x H100, TP=1)
|   |-- chat.py                             # Interactive chat client
|   |-- benchmark.py                        # Before/after comparison tool
|   +-+ openclaw-env.sh                     # Shell env for WhatsApp integration
|
|-- dashboard/
|   |-- src/
|   |   |-- app/                             # Pages + API routes
|   |   |-- components/                     # UI components
|   |   |-- hooks/                          # Polling hooks
|   |   |-- stores/                         # Zustand state management
|   |   +-+ lib/                            # Types, constants, demo data
|   |-- package.json
|   +-+ .env.example                      # Template for .env.local
|
+-+ assets/
    +-+ dashboard.png                     # Dashboard screenshot
```

11 Troubleshooting

| Problem | Solution |
|---|---|
| Container crashes on start | Set Start Command to <code>bash -c "sleep infinity"</code> |
| <code>ModuleNotFoundError</code> | Wrong Docker image. Must be <code>slimerl/slime:latest</code> |
| Server not responding on port 30000 | Wait 3–5 min for startup. Check with <code>curl localhost:30000/healthz</code> |
| <code>train_async.py</code> not found | You must <code>cd /workspace/OpenClaw-RL/slime</code> before launching |
| Path errors about <code>/absolute/path/to/</code> | Upstream launch script has placeholder paths—use our pre-configured scripts |
| SCP not working with RunPod SSH | RunPod uses an SSH relay. Use <code>curl</code> or heredocs to transfer files |
| WhatsApp “404 status code (no body)” | Update <code>baseUrl</code> in both <code>openclaw.json</code> AND <code>models.json</code> |
| PRM scores all 0.5 | Model may need warmup. Have more conversations with clear good/bad signals |
| Dashboard shows “Demo Mode” | Check <code>.env.local</code> URLs match your current pod ID. Restart with <code>npm run dev</code> |
| Training not starting | Need 32 samples to fill a batch. Check count on the PRM record file |
| GPU out of memory | Reduce <code>-rollout-max-context-len</code> or use 4 GPUs instead of 3 |

Table 6: Common issues and solutions

12 Port Reference

| Port | Service | Purpose |
|-------|-------------------------------------|--|
| 30000 | <code>openclaw_api_server.py</code> | Inference API (wraps SGLang) |
| 8080 | <code>metrics_server.py</code> | Dashboard data (metrics, GPU, conversations) |
| 8265 | Ray Dashboard | Training job monitoring and logs |

Table 7: Port map on RunPod

13 Quick Reference: Complete Workflow

Here is the complete workflow from start to finish:

1. Create RunPod pod (3–4× H100, `slimerl/slime:latest`)
2. SSH into the pod
3. Run `bash setup.sh` (clone repo + download model)
4. Run `bash deploy.sh launch` (start training server)

5. Run `bash deploy.sh test` (verify 4/4 pass)
6. Run `bash deploy.sh benchmark` (save pre-training baseline)
7. Start metrics server: `uvicorn pod-server.metrics_server:app -host 0.0.0.0 -port 8080`
8. *On local machine:* Set up `dashboard/.env.local` and run `npm run dev`
9. *On local machine (optional):* Install OpenClaw, configure WhatsApp
10. Have 30–50 conversations (via chat, dashboard, or WhatsApp)
11. Run `bash deploy.sh benchmark` (save post-training results)
12. Run `bash deploy.sh stop` and stop the pod in RunPod console

14 Credits

- [OpenClaw-RL](#) by Gen-Verse — the GRPO training framework
- [Slime](#) by THUDM (Tsinghua University) — async RL orchestration
- [SGLang](#) — fast LLM serving
- [Megatron-LM](#) by NVIDIA — distributed training
- [OpenClaw](#) — WhatsApp AI gateway

Made by Vizuara AI Labs

<https://vizuara.ai>

MIT License | 2025