

## Option A (recommended): Remote dev on Nebius via SSH (Cursor edits remote repo)

### What you get

- Cursor is “local UI”, but your code **executes on Nebius** (GPU, models, deps).
- Zero bandwidth pain sending clips back/forth.
- Simplest operationally for a competition sprint.

### How

1. Create a Nebius VM and enable SSH access (Nebius Compute quickstart).
2. Use Cursor’s **Remote SSH** to open the repo on that VM (same as VS Code Remote SSH workflows; Nebius supports SSH-based dev setups).
3. Clone your project + Cosmos repos on the VM (you already planned this).
4. Run Cosmos Reason2 inference locally on the VM GPU.

This pattern matches what you want: “edit locally, heavy duty runs on Nebius” — because your editor is local but the runtime is remote.

---

## Option B: Local dev runs locally, Nebius hosts an inference API (vLLM server)

### What you get

- Your local machine runs the “agent loop” code.
- Nebius runs a long-lived **Reason2 inference server**.
- Your local code sends video paths/bytes to the remote API and receives structured responses.

This is great if you want to test logic locally without always SSH-ing into Nebius, but it adds more networking + deployment surface.

NVIDIA explicitly recommends deploying Reason2 with **vLLM** and provides a concrete server command.

Nebius also publishes practical guidance for serving LLMs with vLLM on their infra.

---

## Recommended setup for your project: Hybrid of A + B

Do **Option A first** (it's the lowest risk).

If you later want a clean “API boundary”, layer in **Option B** once the agent works.

That keeps you finishable and judge-readable.

---

## Concrete steps to implement Option B (Nebius-hosted inference)

### 1) Provision Nebius GPU VM

- Follow Nebius “create VM / SSH keys” quickstart.
- Pick a GPU with enough VRAM for Reason2:
  - Start with **Cosmos-Reason2-2B** unless you have lots of VRAM headroom. (Reason2 repo documents GPU memory expectations; bigger models need more).
- Pricing info is public if you need quick cost sanity checks.

### 2) Install Reason2 runtime on Nebius

Use NVIDIA’s official Reason2 install path (uv + dependencies).

Then authenticate to pull model weights (typically via Hugging Face).

### 3) Start the Reason2 vLLM server on Nebius

NVIDIA provides the server command format in the Reason2 reference. Example (adjust model id as needed):

- Important flags from NVIDIA:
  - `--allowed-local-media-path` to control what local files can be read
  - `--max-model-len` to avoid OOM
  - `--media-io-kwarg`s for video frame behavior
  - `--port 8000`

#### 4) Secure access: don't expose port 8000 publicly

Do **SSH tunneling** instead of opening the port to the internet.

From your laptop:

```
ssh -L 8000:localhost:8000 ubuntu@<NEBIUS_VM_IP>
```

Now your local machine can call:

- `http://localhost:8000` (tunneled to Nebius)

This avoids firewall/ingress complexity and is “competition safe”.

#### 5) Local dev calls the remote model

Your local `reasoning/cosmos_reasoner.py` can hit the tunneled endpoint. vLLM's CLI serving is standard and well documented.

---

## What I'd do for Cosmos Cookoff (fast + robust)

### Phase 1 (today): Option A only

- Cursor Remote SSH into Nebius VM

- Everything runs there
- You record results and build evaluation artifacts

### **Phase 2 (only if needed): Add vLLM server**

- Makes your agent code cleaner
- Lets you keep your local machine “thin”

This approach keeps you aligned with your locked project constraints: inference-only reasoning agent, no training, deterministic constraint layer.

---

### **Decision rule: A vs B**

- If you’re still building core modules → **Option A (Remote SSH dev)** is best.
- If your core loop is stable and you want clean separation → add **Option B (API server)**.