# Cumulus

Liquid AI Compute

# Our Team

**Suryaa Rajinikanth**
Co-Founder

- Helped build and scale TensorDock, distributed GPU marketplace
- Worked on critical infrastructure at **Palantir**

**Veer Shah**
Co-Founder

- Senior Engineer at AiRANACULUS
- Building networking infrastructure for space/military communication
- 2+ years of customer facing experience

# Developers Want Change

**In the past week…**

- We've reached out to 5+ startups and labs
- Validated strong demand for:
  - Elastic GPU scaling
  - Cheaper spot pricing
  - Easier deployment
  - Outcome oriented infrastructure

Tandem

Georgia Tech Research Institute

AiRANACULUS™

FIRST

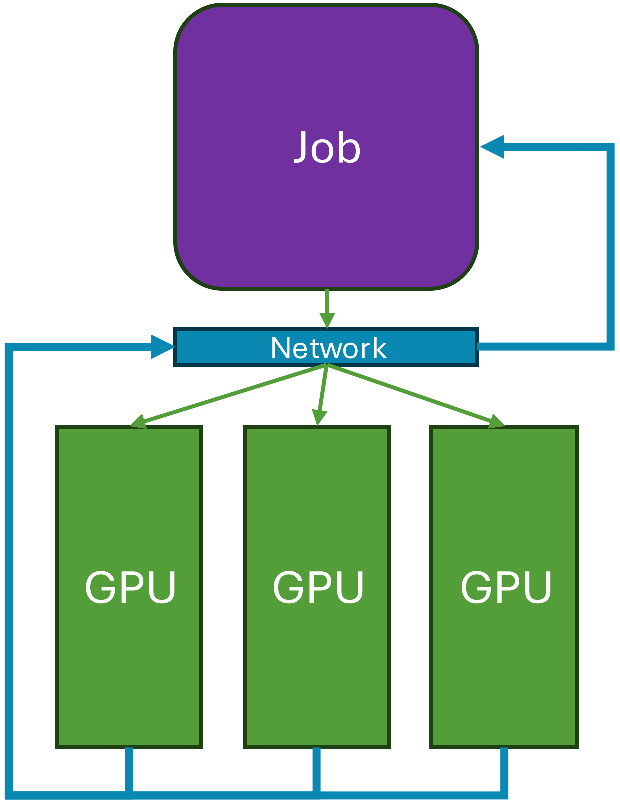
# Firms will optimize their compute for cost & performance.

# The GPU Cloud Crisis

Cumulus

**Today's GPU clouds are rigid and inefficient:**

- Expensive, inflexible pricing
- Hard to set up or scale without DevOps expertise
- No visibility into runtime tradeoffs:
  - 1× high-end GPU vs. 2× mid-tier GPUs?
  - Peak-time vs. off-peak costs?
  - Unknown until you pay for it.
- Users waste money, and providers waste GPUs.



1) Intercept calls from job
2) Split Across Pool GPUs
3) Aggregated and Returned

*Minimal code changes required; Elastic Computing Model*

# Developers Are Unsatisfied

**Adithya**

AI @ Petra Security

- Receives new data to fine tune model but must batch for **efficiency**
- Spends too much on **scarce GPU** compute

**Kartik**

NLP @ GT

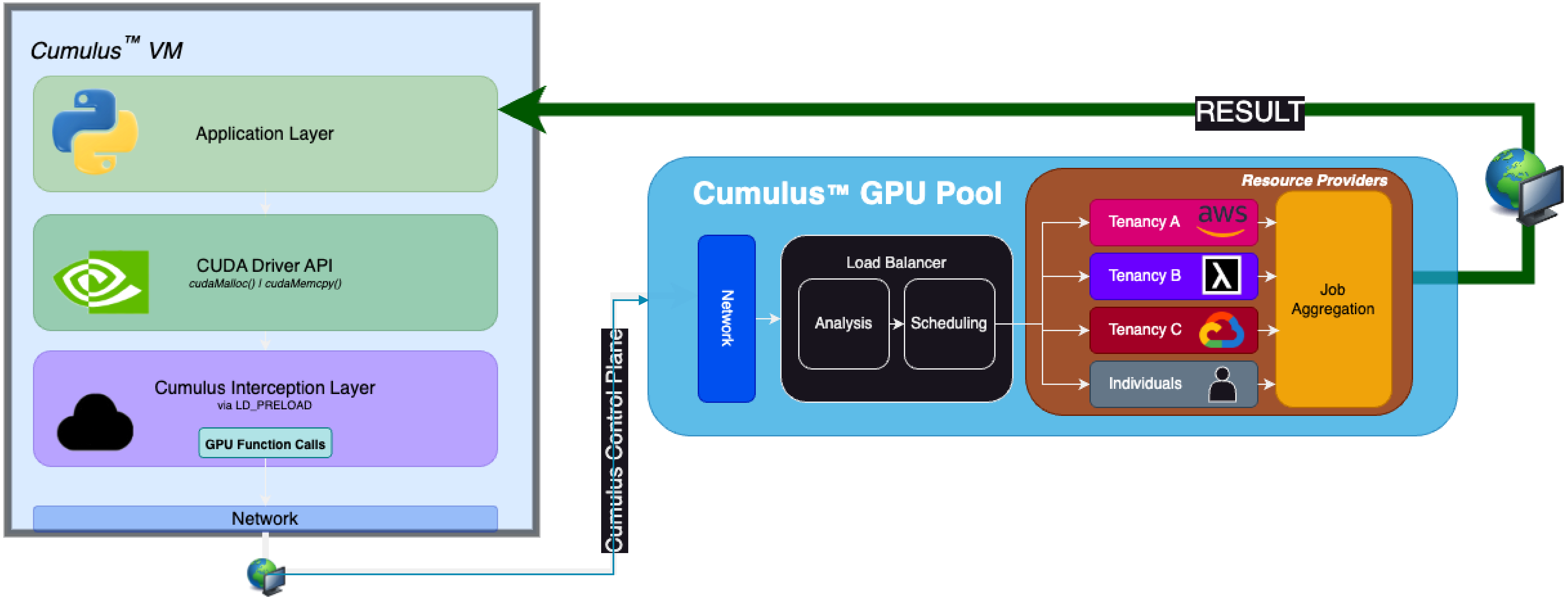
- Students and researchers wait in **long queues** on GPU cluster
- Progress constrained by **compute**

**Arjun**

Co-Founder @ Tandem

- Doesn’t want to deal with infrastructure to fine tune models
- Constrained by **complexity & cost**

# Product Workflow



# Our Unique Advantages

## 1. Democratized Supply

- Leverages all cloud providers and unused consumer cards

## 2. Intelligent Orchestration

- Intercepts low-level CUDA calls from your workload
- Analyzes task characteristics in real-time
- Dynamically route to optimal GPU combination

## 3. Adaptive Routing

- Migrates between GPUs live to optimize cost/performance
- Cache results to minimize re-computation

## 4. Outcome-Based Pricing

- Set your budget and performance requirements
- Charged per-compute-second, not per-GPU-hour

# Example Use Case

Cumulus

| Platform | Specs & Cost | Setup & Reliability | Performance & Automation |
|---|---|---|---|
| **Traditional Cloud (AWS)** | • 1× H100 @ \$2.00/hr × 4hr = \$8.00<br>• Manual provisioning & configuration | • Requires DevOps expertise<br>• Limited automation | • Expensive<br>• No parallelization |
| **Vast.ai** | • 1× RTX 4090 @ \$0.40/hr × 8hr = \$3.20<br>• - Manual GPU selection | • User handles setup<br>• Reliability varies | • No built-in optimization<br>• Slowest |
| **Cumulus** | • Auto-routed to 6× RTX 4090s @ \$0.40/hr × 4.5hr = \$1.80 | • Declarative interface<br>• No manual setup required | • Parallelized<br>• Automated routing, load balancing, optimization |

**Result: 78% cheaper than AWS, faster than Vast.ai, zero DevOps**

# GPU Rental Is Exploding

- Global GPU shortage projected to last **through 2027+**.
- AI demand up **10× year-over-year**.
- Yet **average GPU utilization < 40%**.

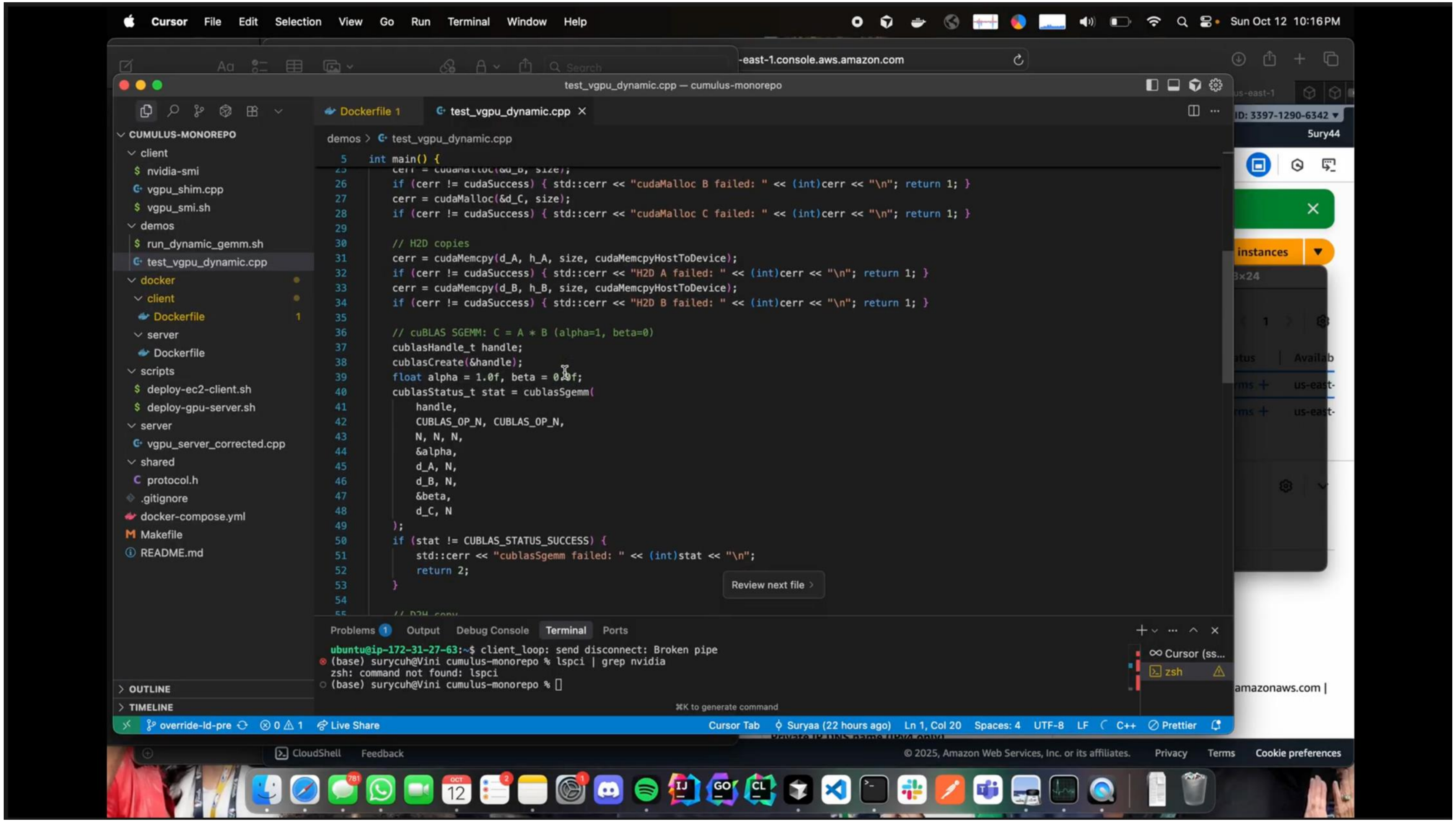
Potential **$49B+**

*Market by 2032*

# Our Solution

| Problem | Solution |
|---|---|
| Static GPU allocation | Elastic, phase-aware scaling |
| Fixed pricing | Dynamic, budget and time based pricing |
| Idle GPUs | Shared tenancy & live migration |
| Fragmented supply | Unified GPU pool of individual hosts |
| Manual setup | One declarative interface |

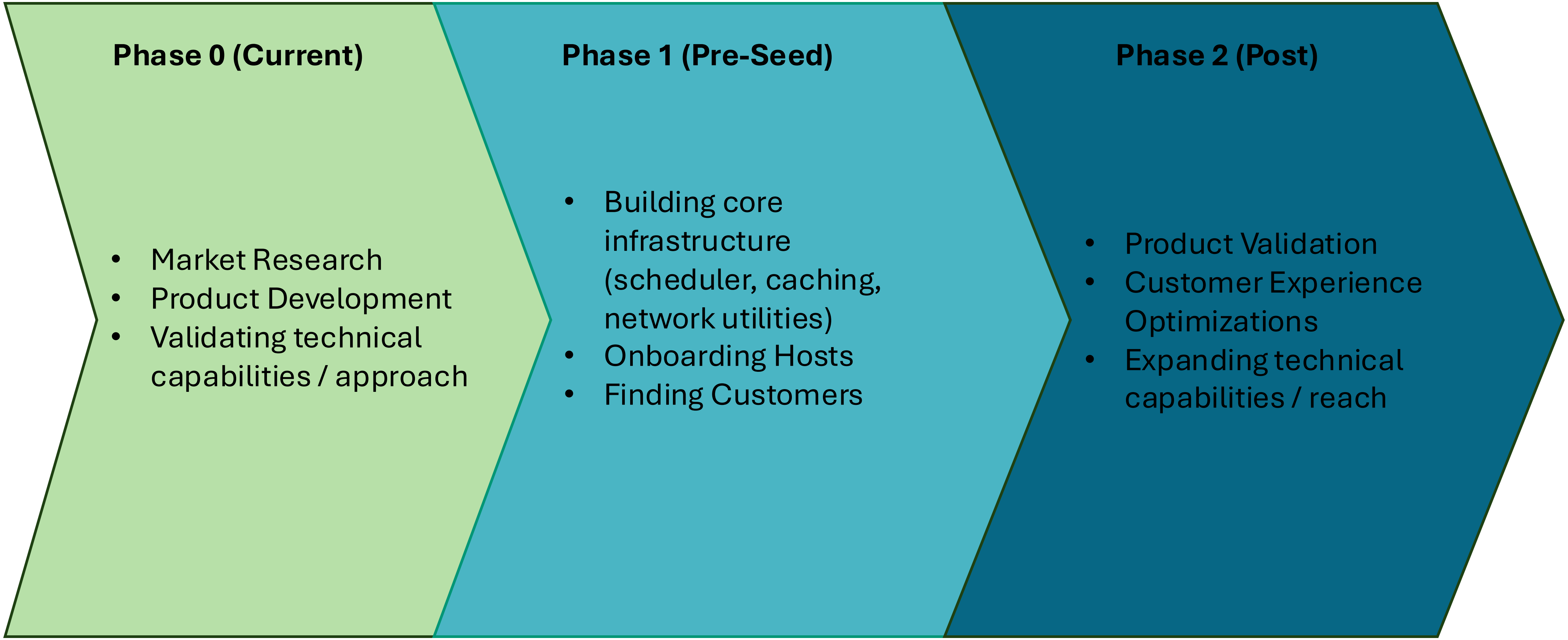
Cumulus is the next generation of liquid, fair, and flexible GPU compute.



```cpp
int main() {
    cerr = cudaMalloc(&d_B, size);
    if (cerr != cudaSuccess) { std::cerr << "cudaMalloc B failed: " << (int)cerr << "\n"; return 1; }
    cerr = cudaMalloc(&d_C, size);
    if (cerr != cudaSuccess) { std::cerr << "cudaMalloc C failed: " << (int)cerr << "\n"; return 1; }

    // H2D copies
    cerr = cudaMemcpy(d_A, h_A, size, cudaMemcpyHostToDevice);
    if (cerr != cudaSuccess) { std::cerr << "H2D A failed: " << (int)cerr << "\n"; return 1; }
    cerr = cudaMemcpy(d_B, h_B, size, cudaMemcpyHostToDevice);
    if (cerr != cudaSuccess) { std::cerr << "H2D B failed: " << (int)cerr << "\n"; return 1; }

    // cuBLAS SGEMM: C = A * B (alpha=1, beta=0)
    cublasHandle_t handle;
    cublasCreate(&handle);
    float alpha = 1.0f, beta = 0.0f;
    cublasStatus_t stat = cublasSgemm(
        handle,
        CUBLAS_OP_N, CUBLAS_OP_N,
        N, N, N,
        &alpha,
        d_A, N,
        d_B, N,
        &beta,
        d_C, N
    );
    if (stat != CUBLAS_STATUS_SUCCESS) {
        std::cerr << "cublasSgemm failed: " << (int)stat << "\n";
        return 2;
    }

    // D2H copy
```

```
ubuntu@ip-172-31-27-63:~$ client_loop: send disconnect: Broken pipe
(base) surycuh@Vini cumulus-monorepo % lspci | grep nvidia
zsh: command not found: lspci
(base) surycuh@Vini cumulus-monorepo %
```

# Product Roadmap



**Phase 0 (Current)**

- Market Research
- Product Development
- Validating technical capabilities / approach

**Phase 1 (Pre-Seed)**

- Building core infrastructure (scheduler, caching, network utilities)
- Onboarding Hosts
- Finding Customers

**Phase 2 (Post)**

- Product Validation
- Customer Experience Optimizations
- Expanding technical capabilities / reach

# Cumulus

Liquid AI Compute