Homepage

Website Improvements Needed: The existing graphics don't need to be changed. This additive to the existing website.

Navigation bar

Pages:

- Product
- Docs
- Blog
- About Us
- Login

Top Right Button: Login (app.juicelabs.co)

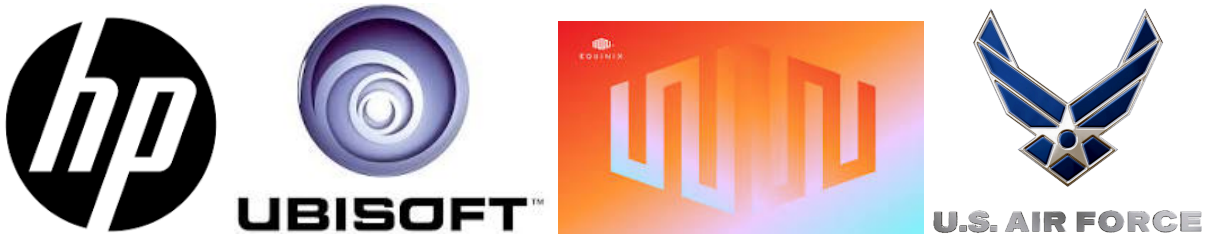
Hero Section:

- **Headline:** "Ephemeral Compute for AI and Graphics"
- **Subheading:** "Juice's GPU-over-IP technology enables seamless, on-the-fly access to your GPUs wherever your applications require them, without any code modifications."
- **Demo Video:** Embed a demonstration video to provide visitors with a visual understanding of Juice's capabilities.

[Demo Video: https://www.youtube.com/watch?v=Kt75NmR8938]

[Call to Action Button - **Free Trial**]

(Scrolling Carousel)
Trusted By (logo): - Remove “Trusted By”



TESTIMONIALS (Remove “Testimonials”)

- “Juice is a complete game changer for us. Being able to connect to remote GPU natively and share resources across workloads allows for massive savings and innovative new deployments.” - **Thomas, Ubisoft**
- “We see a great opportunity to partner on achieving our longer-term objectives related to end-user computing and the paradigm shift of AI… We are impressed by the technological advancements [Juice] has achieved in the computing space.” – **Jarrett Simmerson, DELL CTO**
- We see immediate benefit… not only in terms of GPU utilization, but in terms of how we build out resources and design from the storage layer to the GPU layer.” - **Chris Simmons, CSO Cambridge Computing**
- **Pure Storage** - (Enrique, sales engineer/systems engineer)
- “With appropriate resources we believe Juice Labs will unlock the potential of Private AI on a global scale.” **Equinix**
- "Like my brain, I still can't comprehend how it works so well” -- **Max Taranov, IT Engineer**

Call-to-Action Button: Include a prominent button labeled "Get a Demo"

Product Page

Similar to the homepage without the big green splash and xlarge font.

Section: Not Another GPU Cloud, Your tailored GPU Cloud

[Demo Video: https://www.youtube.com/watch?v=Kt75NmR8938]

- **Heading:** "What Is Juice and Why Is It Different from other Serverless GPU Solutions?"
- **Content:**
    - **Introduction:**
      Juice offers a revolutionary approach to GPU utilization, distinct from traditional serverless GPU services.
    - **Key Differentiators:**
        - **Local Applications, Remote GPU**
        - **Works on Prem, works in the cloud**
        - **Can fractionalize GPU on the fly, no resetting systems like vGPU or MIG**
        - **No Code Changes, just run it**
        - **No k8s needed**
        - **Cross OS Functionality**
            - **Windows <> Linux**
            - **MacOS Support via Container**
                - **Run NVIDIA GPU on Macbooks**
        - **On-Premises and Cloud Deployment:** Juice supports both on-premises and cloud environments, offering unparalleled flexibility compared to conventional serverless GPU services that are often confined to specific cloud providers.
        - **Seamless Integration:** Unlike typical serverless GPU solutions that may require code adjustments or specific frameworks, Juice integrates effortlessly into your existing workflows without necessitating code changes.
        - **Dynamic Resource Allocation:** Juice provides the flexibility to allocate entire GPUs or micro-fractions on an application-by-application basis, optimizing resource utilization and cost-effectiveness.
        - **High Performance:** By enabling near-100% GPU utilization through efficient sharing and pooling across networks, Juice ensures superior performance for AI and graphics workloads.
        - **Scalability:** Juice allows for the independent scaling of GPU resources across data centers, facilitating efficient resource balancing and maximum performance.
    - **Conclusion:**
      By leveraging Juice's GPU-over-IP technology, you gain a versatile, high-performance solution that seamlessly integrates into your existing infrastructure, setting it apart from traditional serverless GPU offerings.

Section: How could this possibly be performant?! What about latency

- [Performance, network latency graphic
    - Placeholder graph
- [Two sentence answer]
    - Whitepaper link for learn more

Compatibility	Windows	Linux	Cross OS Compatibility	Mac*
OS	✅	✅	✅	✅

OS Compatibility Notes:

- "Cross OS Compatibility" refers to software or tools that function seamlessly across multiple operating systems.
- Mac* indicates that some tools or features may have limited support or functionality on macOS.

Supported CUDA Libraries:

- Pytorch
- Tensorflow

Supported graphics software:

- DirectX 10-12
- OpenGL
- OpenCL
- Vulkan

Full list of features:

- **Orchestration with or without K8's**
  - **Pools**
  - **User Access Control**
  - **Tags and Taints**
  - **Dashboards (coming soon!)**
    - **GPU utilization**
    - **Session management**
- **Product interfaces**
  - **Command Line Interface**
  - **Pip install juice**
    - **Add GPU over IP access natively to your applications**
  - **SDK - (coming soon!)**
  - **Web application (coming soon!)**

Documentation Page

Juice User Guide

Juice GPU Trial User Guide

System Requirements

Windows

Both Client and GPU Agent are supported on Windows 10/11.

Linux

The Client and GPU Agent are supported on Debian based distributions on version 11 and greater, for example Ubuntu 20.04 and greater. Other distributions with a similar release date or newer should be supported as well.

For running within WSL on Windows, only the Client is currently supported (not the GPU Agent).

Preconditions for running a Client app

The following are necessary to *use* a remote GPU:

Windows

- Windows 10/11

If you're running on a machine without a GPU and drivers installed, e.g. a VM, then you must install the Vulkan Runtime. Download and install from https://sdk.lunarg.com/sdk/download/latest/windows/vulkan-runtime.exe

Linux

- Ubuntu 20.04 or greater, Debian 11 or greater
- The following packages must be installed:
    - libatomic1
    - libnuma1

These can be installed by running `apt update && apt install libatomic1 libnuma1`

Preconditions for running a GPU agent

The following are necessary to *host* a remote GPU:

On all platforms an NVIDIA GPU with an installed driver version 535 or greater, and a CUDA driver version 12.2 through to 12.4.

Windows

- Windows 10/11

Linux

- Ubuntu 20.04 or greater, Debian 11 or greater
- The following packages installed:
    - libatomic1
    - libnuma1
    - libvulkan1
    - libgl1
    - libglib2.0-0

These can be installed by running `apt update && apt install libvulkan1 libgl1 libglib2.0-0 libatomic1 libnuma1`

Accessing the Trial

1. Contact will@juicelabs.co if you need an email invitation to the trial.

2. You should have received an email with the subject “Set up your Juice GPU account”. Follow the instructions in that email to set your password. Please create a new password for this trial.

3. Access this download screen:

Windows

4. Click to download the Windows release.

5. Double-click the downloaded file to install Juice GPU for Windows. Restart if instructed.

6. See below regarding using the desktop app.

Linux

4. Click to download the Linux release. Depending on your Linux setup, it might be necessary to download the file on a different system, then copy the file to your intended Linux client system.

5. Once the file is on your Linux client system, run `tar -xf [name of file]` in the directory you want the application to run from.

6. See below regarding using the CLI application and sharing a GPU.

Using the Desktop App (Windows)

1. Open the “Juice Desktop” app (look in `C:\Program Files\Juice GPU` if you have trouble finding it).

2. Click the Login button, and log in with your email address and password.

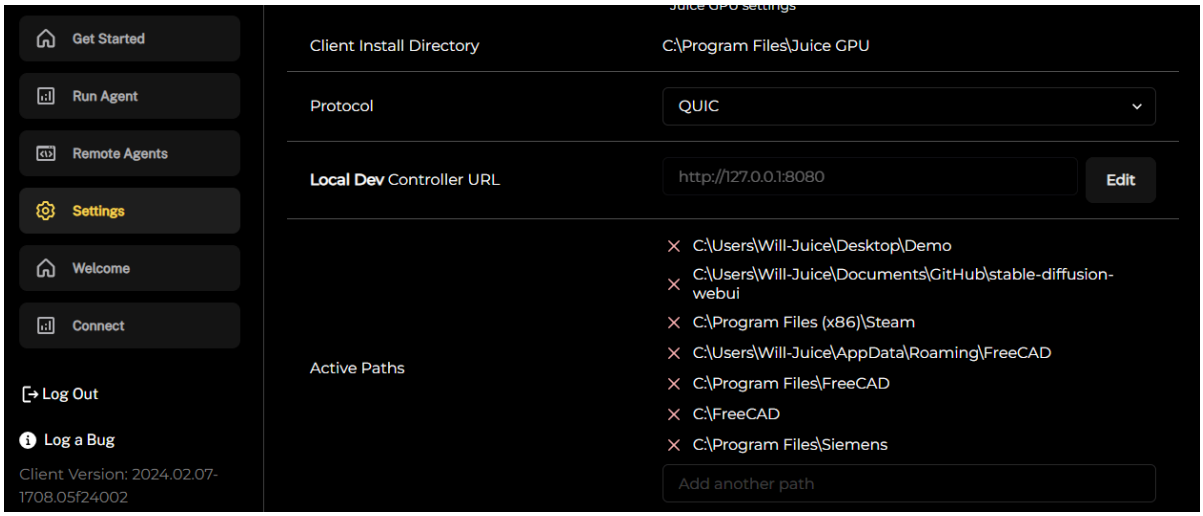
3. At times during the Trial period, a new version of the desktop app may become available. If you see a message at the top of the app window prompting you to do so, please Update and Install the latest version.

4. The main navigation in the desktop app is via the vertical bar on the left. The rest of this section of the guide will cover the functions of each element in the nav.

Settings - “Active Paths”

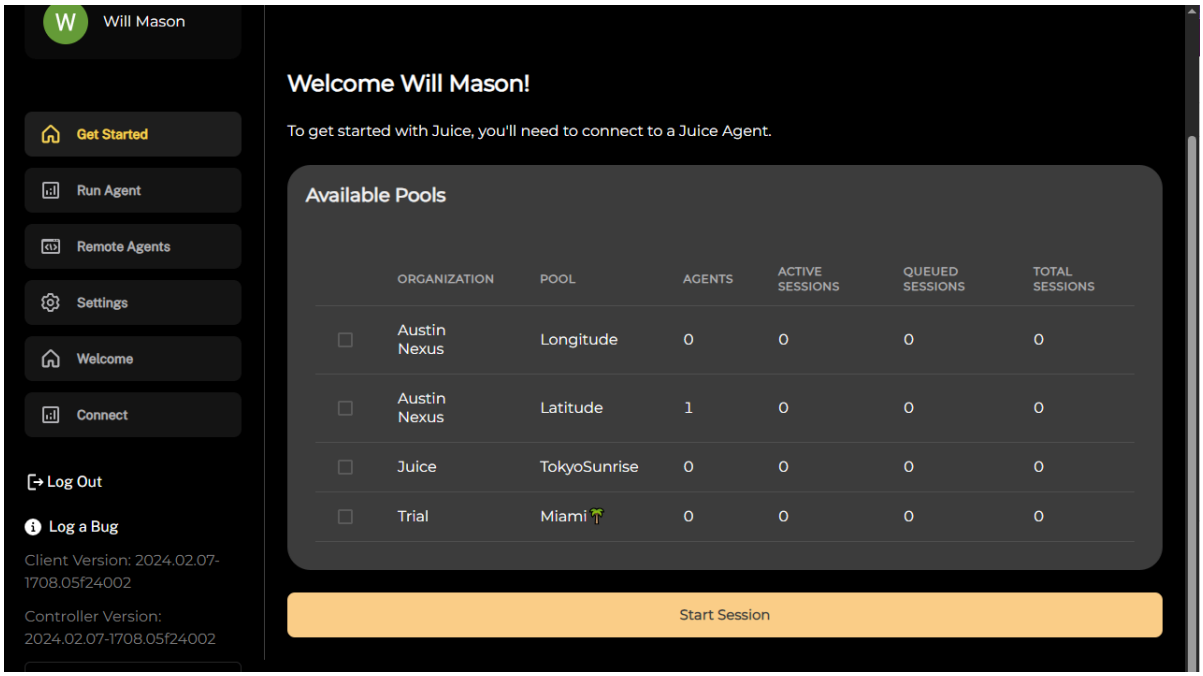
Any application that is listed here - or that has a parent folder listed here - will be launched via remote GPU instead of the local GPU, if there is a remote GPU connected.

Add specific applications or folders by either typing/pasting a path:



Getting Started

This section allows for connection to a remote GPU, or displays any existing connection to a remote GPU.



Pools

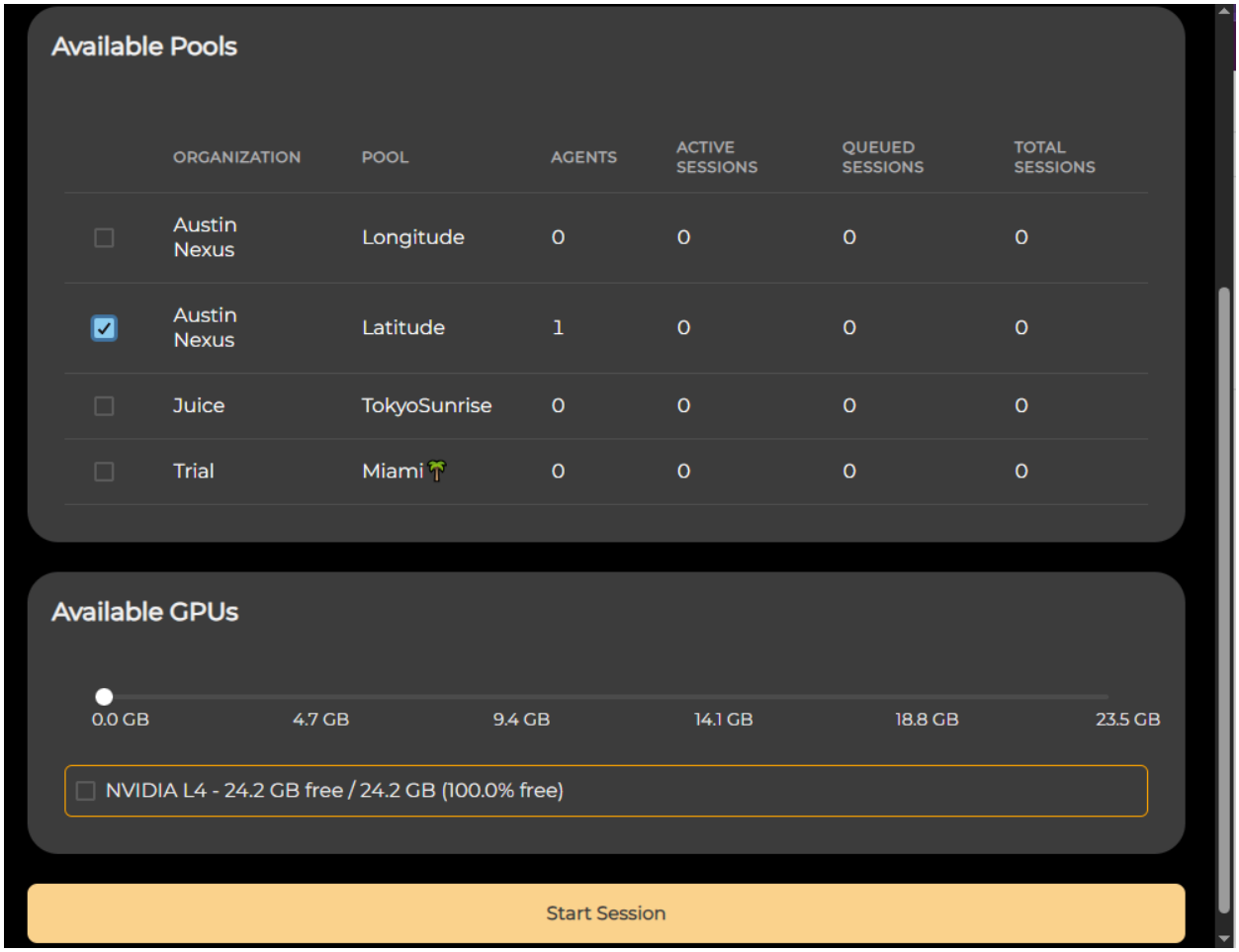
A pool is a collection of remote GPUs that have been offered up for use, combined with a set of users who are permitted access to those remote GPUs.

You might have been invited to other pools by communication with operations@juicelabs.co or by other colleagues (see below regarding administration). If not, you will have only your own pool called “My Pool” at first, and it will not have any GPUs in it by default.

Connecting to a remote GPU

If there *are* remote GPUs available to you inside any of the pools you have access to:

1. Check the box next to the GPU pool you would like to connect to, like this:

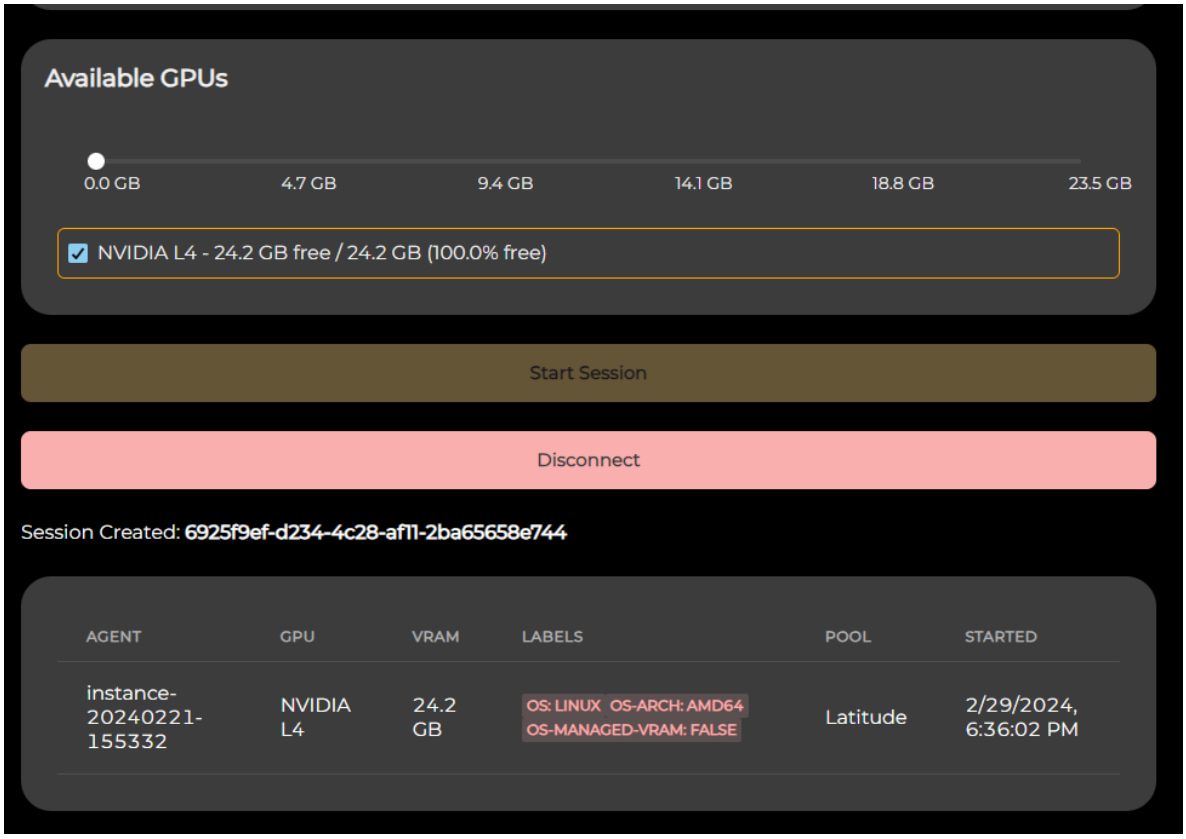


Clicking the small refresh button  *in the upper right will ensure you have the most current view of available GPUs, Pools, and Host Machines.*

2. Select one GPU-type panel (e.g. NVIDIA RTX A6000 shown with orange outline indicating it as selected).

3. Click the Start Session button.

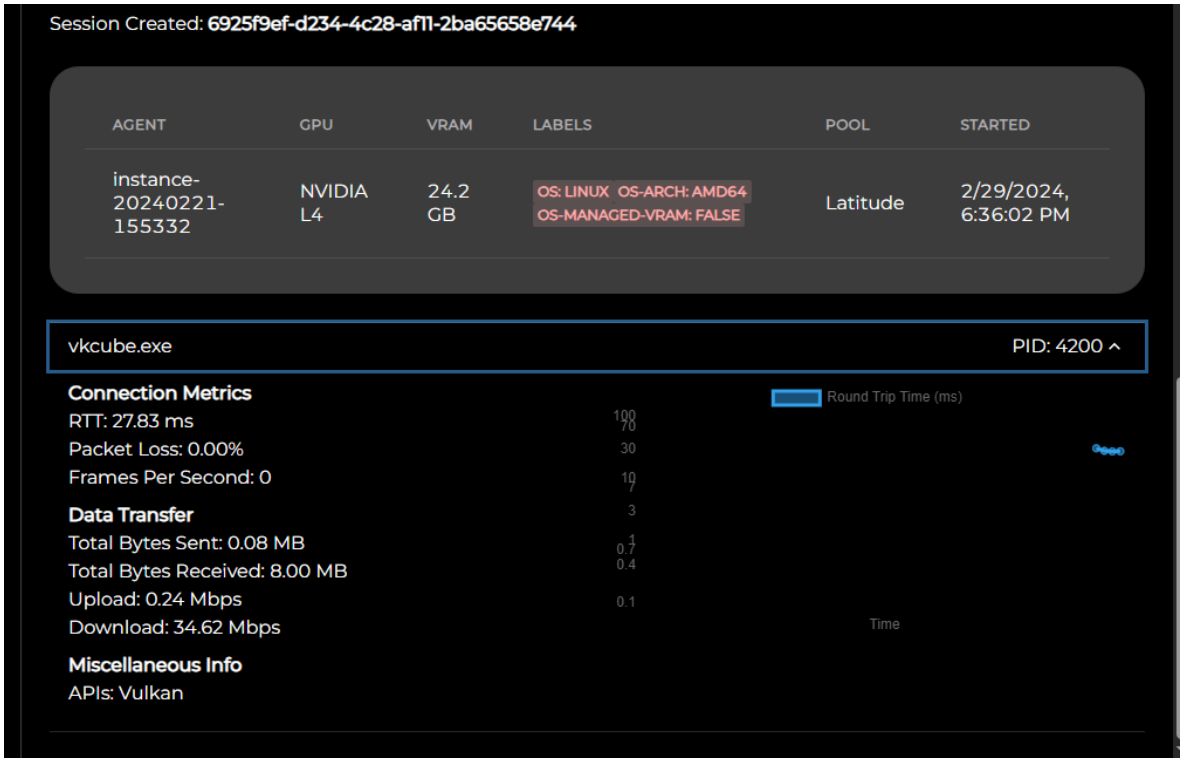
4. You should now see a screen indicating that you are connected to a remote GPU, like this:



Running an application via remote GPU

While your system is connected to a remote GPU, any application which has its path listed in the Active Paths in the Settings tab will run via the remote GPU instead of any local GPU.

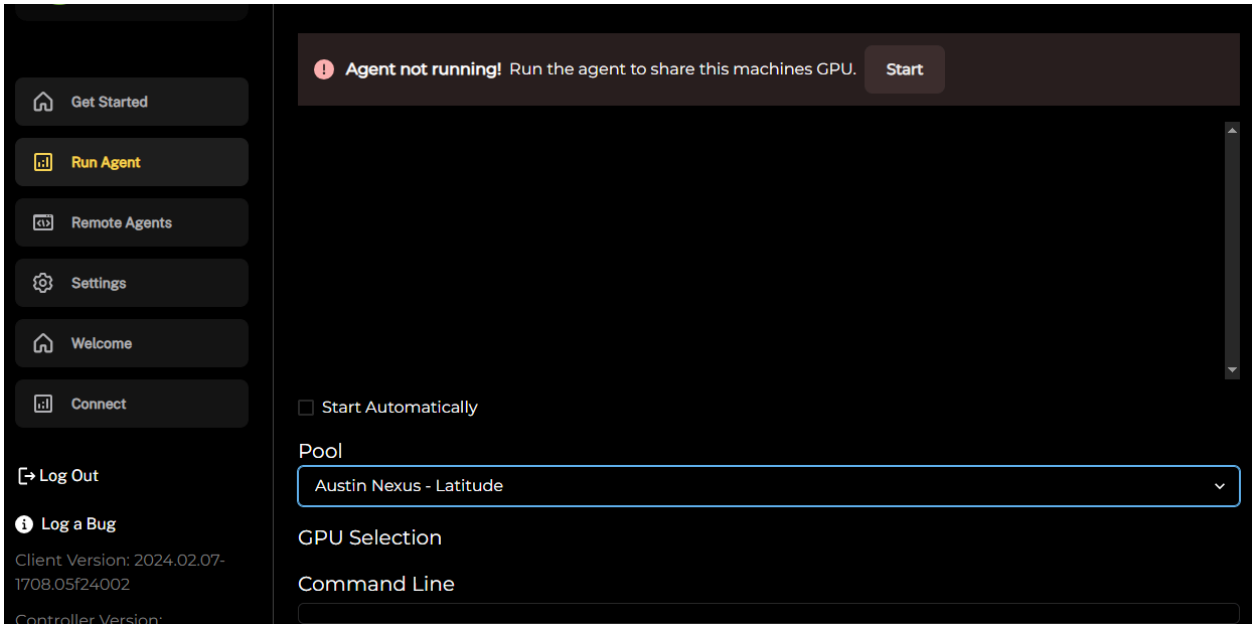
While running applications via remote GPU the “Getting Started” tab displays application telemetry in a section at bottom. Expanding an application row using the arrow at right will reveal real-time performance metrics for that application, e.g. "vkcube.exe" shown here:



“Run Agent” Nav

“Run Agent” tab

To make the GPU(s) from the system that is running this desktop app available to a pool as remote GPU(s):



1. Select the GPU(s) you want to share in GPU selection

2. Select a pool to add the GPU(s) to

3. Click the “Start” button

You will see a screen indicating that your GPU is shared in that pool, a button to stop sharing, and an option to automatically do the same sharing action whenever the app is started up. You will also see the console output for the GPU server.

“Settings” Nav

For the purposes of this Trial, please do not change any other settings here without discussing via operations@juicelabs.co.

“Feedback” Nav

Your mechanism for reporting issues, ideas, or any other feedback to the Trial team.

“Account” Nav

The account under which you’re logged in, and the version of the desktop app you are running.

“Logout” Nav

Logs you out of the desktop app.

Using the CLI app (Linux, Windows, WSL)

`juice` is a command-line interface (CLI) tool designed to facilitate various operations and configurations for users. It provides a wide range of commands and options to manage configurations, contexts, agents, pools, organizations, sessions, runs, users, and authentication.

> NOTE: In this document every example command will be shown as `./juice` as it works on Linux. For Windows, substitute `juice.exe` in place of `./juice`.

This document will focus on the main capabilities and primary features of the CLI app. For

Help within the app

At any point you can get in-context help by appending `--help`, or just `-h`. For example, `./juice -h` gives you top-level help, while something like `./juice session -h` gives you help with sessions.

Logging in

1. `./juice login`

2. Scan the QR code using your phone, and log in.

3. After a few moments, the CLI interface will show you as “Logged in”.

Running an application against a remote GPU

The most straightforward way to run an application is to use the `run` command, like this:

```
./juice run <application> [<arg>...]
```

The `run` command encapsulates several steps: it requests a session (connecting you to a remote GPU), runs the application using that remote GPU, then releases the session, freeing that remote GPU back into its pool.

Try `run -h` for more advanced capabilities, including the ability to narrow your remote GPU target(s) by pool(s), session(s), or even specific GPUs.

Managing sessions manually

Beyond the simple elegance of the `run` command, you can also manage sessions yourself. You might want to do this if you want to:

- keep connected to the same GPU over a period of time and over multiple application runs
- connect to a specific GPU
- establish several sessions simultaneously to multiple different GPUs

`./juice session request` will connect you to a GPU in one of your pools, and you will remain connected until you `session release` (or of course until something else happens to sever the connection, like the GPU agent being disconnected for example).

Though you can maintain multiple sessions simultaneously, at any given time you have one and only one “active” session. You can change which session is active using `session activate <session>` .

The `release` command will release the active session by default. You can also release a specific session using `session release <session>` .

In order to specifically reference any session, you will need to populate `<session>` with the correct ID. The ID looks something like `a1350d37-e84c-4bd1-882c-0fb41e844d81` and is returned when you `request`. You can also use `session list` to see current sessions, including their IDs.

Choosing a specific GPU

If there is a specific GPU that you want to connect to:

1. Use `gpu list` to see all GPUs available to you in any of your pools.
2. Look for a GPU that is not currently in use, by looking for the key-value pair `"usedvram":0`
3. Copy the `uuid` value of the GPU that you want to connect to.

Specify the GPU’s ID for the run command using `run --gpu <uuid>` or by requesting a session using `session request --gpu <uuid>`

Multi-GPU Support

Juice allows you to take advantage of multiple GPUs on a single host. This enables you to run more demanding workloads or to run multiple applications concurrently on a single machine.

Using the CLI:

To request a session with multiple GPUs, use the --num-gpus flag when creating a session or running a workload. For example:

```
juice run --num-gpus 2 python
```

content_copyUse code with caution.Bash

This command requests an agent that has 2 available GPUs.

Important Considerations:

- To use multiple GPUs, you must have a host device with multiple GPUs installed.
- The Juice agent running on the host will expose all GPUs by default.
- The effectiveness of utilizing multiple GPUs is highly workload dependent. For example, some workloads may be able to leverage multiple GPUs for faster processing, while others may not benefit from this.

Example:

Here is a trivial example of using PyTorch to print out the device properties for a multi-GPU system:

```
import torch

# Check the number of GPUs available
print(torch.cuda.device_count())

# Print the properties of each GPU
print(torch.cuda.get_device_properties(0))
print(torch.cuda.get_device_properties(1))
```

content_copyUse code with caution.Python

General Best Practices for Multi-GPU Workloads:

- **LLMs:** For LLMs, using frameworks like llama.cpp (https://github.com/ggerganov/llama.cpp) can help split the model layers across multiple GPUs, enabling larger models to be loaded and used. The oobabooga UI (https://github.com/oobabooga/oobabooga) can manage multiple backends, including llama.cpp, and utilize multiple GPUs effectively.
- **Image Generation:** Explore frameworks like Stable Diffusion, which might offer multi-GPU support in more recent versions. Look for updates and documentation on their GitHub repository or official website.

- **Fine Tuning:** A good starting point for fine tuning models is Stable Diffusion's documentation (https://huggingface.co/docs/diffusers/v0.9.0/en/training/text2image). You can use this training with tools like AUTOMATIC1111/stable-diffusion-webui for a user-friendly experience.
- **Training:** Consider replicating training benchmarks like ResNet, YOLOv8, GPT2, or even the MLPerf training benchmark to test the performance of multiple GPUs for your specific training needs.

Remember, the best approach for utilizing multiple GPUs will depend on your specific workload and application requirements.

Sharing a GPU (Linux, WSL)

1. On the download screen from Accessing the Trial, scroll to the bottom and click the “Linux” tab.

2. Copy the command line code.

3. Paste and run the code from inside the directory where you extracted the file on setup.

4. While this agent is running, GPUs on this system are available within your default pool. If you would like more advanced pooling, see below regarding administration.

More advanced concepts

Higher-level administration

Juice has richer capabilities for administration of users, pools, and permissions, but for this Trial, please Juice for more details.

Advanced parameters

Run command options

- `--disable-cache`: Disables caching
- `--disable-compression`: Disables compression

- `--ephemeral`: Always requests a new session for the application
- `--match-labels <string>`: Comma separated list of key=value pairs (see below)
- `--no-version-check`: Disables version checking when allocating the session requirements
- `--on-connection-error <string>`: When a connection error happens, [fail, continue] (default "fail")
- `--on-queue-timeout <string>`: When a queue timeout happens, [fail, continue] (default "fail")
- `--pool-ids <string>`: Comma separated list of pool ids from which to allocate the session resources, when empty any available pool you have access to is used (use `pool list` to get necessary pool ids)
- `--queue-timeout <uint>`: Maximum number of seconds to wait for a GPU
- `--tolerates <string>`: Comma separated list of key=value pairs (see below)

Global options

- `--log-file`: Specify a file to log CLI output. By default, logs are printed to stderr.
- `--log-level`: Set the logging level (error is the default). Acceptable values include debug, info, warn, error, and fatal.
- `--quiet`: Run the CLI in quiet mode, suppressing most output.
- `--no-banner`: Suppress the display of the startup banner.
- `--controller`: Specify the controller option for advanced configurations.

Labels and Match Labels

The concept of Labels and Match Labels has been taken from Kubernetes. To learn more about this concept from Kubernetes, please see https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/.

In essence, the Agent can be started with `--labels <key=value,...>` to provide a set of labels for the Agent. Then when creating a session, the `juice run` command takes `--match-labels <key=value,...>`. These match labels indicate to the controller when selecting an Agent for the session request to only consider Agents that have every match label specified.

Taints and Tolerates

The concept of Taints and Tolerates has been taken from Kubernetes. To learn more about this concept from Kubernetes, please see https://kubernetes.io/docs/concepts/scheduling-eviction/taint-and-toleration/.

In essence, the Agent can be started with `--taints <key=value,...>` to provide a set of taints for the Agent. Then when creating a session, the `juice run` command takes `--tolerates <key=value,...>`. The taints specified when creating an Agent indicate to the controller that it can only assign session requests to the Agent if the session request was created with the complete set of taints in the `--tolerates <key=value,...>` option.

Known Issues

- Tokens on agents expire after 30 days
- Multi-GPU scenarios not supported
- Running an Agent inside a Docker container requires host networking, which is not supported in some environments (macOS & VM-based containers) WSL sharing GPUs has the same limitations as Docker WSL is subnetted with the host machine.
- DX11/DX12/OpenGL/Vulkan are the graphics API that are supported. Most (almost all) graphical software *should* work out of the box. CUDA 10-12.4 is supported for compute. Pytorch and Tensorflow binaries *should* work out of the box.
- Game resolutions beyond 4K may have questionable performance due to video encoder/decoder limitations and will be addressed with a software encoder in the future.

Troubleshooting / FAQ

I can't connect to my Agent
Ensure that (1) the agent is running on the correct Pool (2) the agent has no active sessions (check agent status from that Pool's page on the web) (3) the user you are trying to connect with has the **Create Session** permission for that pool.

If there's a session from another user on the agent, currently the only way to reset it is to restart the agent.

My application is not starting
Check the telemetry in the desktop app - see if we're uploading data to the agent. If data is being uploaded, please wait. Depending on the application and the network connection this can take many minutes.

My application is not using the Juice GPU
Verify that (1) you have an active session in the desktop client and (2) you've correctly set the **Active Path** for your executable in Settings. Verify that you have the following Registry Keys set

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session
Manager\AppCertDLLs

JuiceAppCert = C:\Program Files\Juice GPU\JuiceAppCert64.dll
```

```
JuiceAppCertWow64 = C:\Program Files\Juice GPU\x86\JuiceAppCert32.dll
```

And verify that those files exist on your machine. You may need to add **C:\Program Files\Juice GPU** to your allowlist in in Windows Defender or other Antivirus software.

My Agent isn't receiving connections (not working)
Verify from the output that the agent is running. You may have to add RenderWin.exe to the **Windows Firewall** rules for incoming UDP connections if you weren't prompted to do so.

If you are running an agent from a Docker container, make sure you are using **network=host**

My Agent is shutting down when my ssh connection is closed
In order to have a linux agent run persistently, launch it using the "daemonize" command (you may need to install this with *sudo apt install daemonize*)

Ex. `daemonize [Path/to/Juice]/agent …`

Blog

Blog content already in webflow it just needs better formatting.

Misc Data

Compatibility	Windows	Linux	Cross OS Compatibility	Mac*
OS	✅	✅	✅	✅

OS Compatibility Notes:

- "Cross OS Compatibility" refers to software or tools that function seamlessly across multiple operating systems.
- Mac* indicates that some tools or features may have limited support or functionality on macOS.

Supported CUDA Libraries:

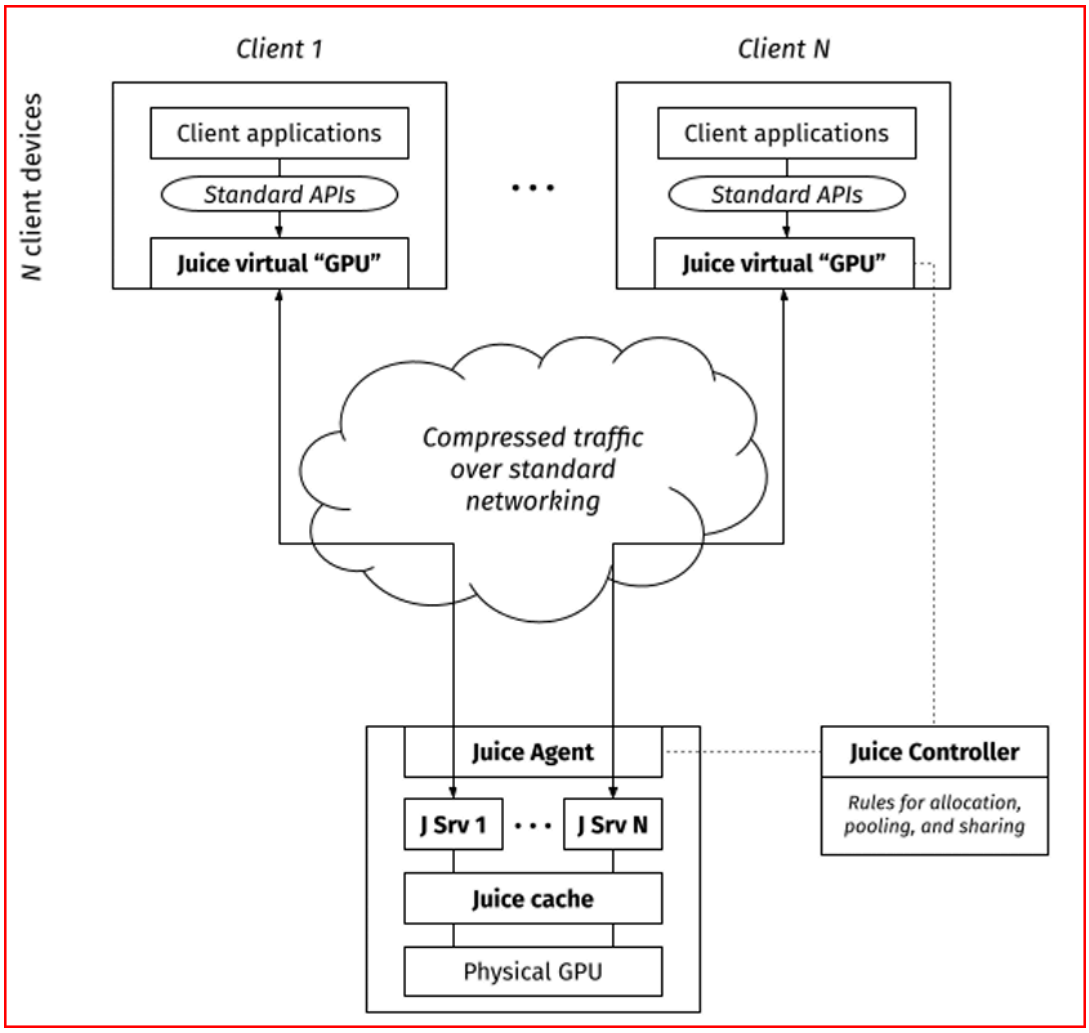
- Pytorch
- Tensorflow

Supported graphics software:

- DirectX 10-12
- OpenGL
- OpenCL
- Vulkan

Full list of features:

- **Orchestration with or without K8's**
  - **Pools**
  - **User Access Control**
  - **Tags and Taints**
  - **Dashboards (coming soon!)**
    - **GPU utilization**
    - **Session management**
- **Product interfaces**
  - **Command Line Interface**
  - **Pip install juice**
    - **Add GPU over IP access natively to your applications**
  - **SDK - (coming soon!)**
  - **Web application (coming soon!)**



Optimizing GPU Resources Across Secure Environments with Juice

The Challenge

Organizations struggle to manage GPU resources across teams and projects with varying security requirements. Siloed GPUinstallations lead to underutilization and in efficient scaling. This results in significant cost overhead and difficulty meeting peak demands while maintaining strict data separation and compliance.

The Juice Solution

Juice revolutionizes GPU resource management by enabling secure sharing across different security environments.Our solution creates a unified GPU pool using VLAN technology for dynamicallocation.This approach maintains security compliance through virtual air-gapping at the network layer,optimizing utilization and adapting to various security protocols.

Key Benefits

Juice's solution dramatically reduces hardware expenses and improves performance by ensuring all projects have access to necessary GPU resources. It enhances security while enabling resource sharing, offers easy scalability, and decreases energy consumption. Importantly, it helps organizations meet various regulatory requirements across different projects and teams.

How It Works

Juice's GPU-over-IP technology seamlessly connects a dedicated network of GPU boxes to separate compute and storage systems for each security level. VLAN switching dynamically allocates GPUs to different environments as needed, maximizing the investment while maintaining the highest levels of security and compliance across all operations.