

Workshop Unit 5

Simulation and RL

Chenyu Yang

19.10.2026



1. Github and Project collaboration
2. SSH and Cloud-computing platform
3. Isaacsim and usd
4. RL and domain randomization



GitHub and project collaboration

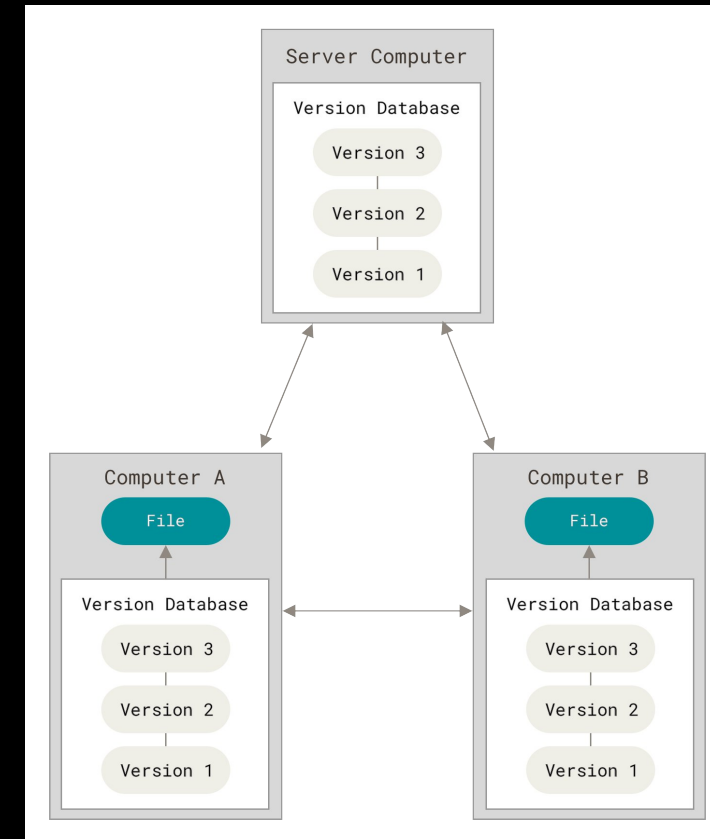
Git: version control on your computer

Record project history, compare versions, and develop changes on branches.

GitHub: a shared home for the project

Host Git repositories, track tasks, and discuss changes through pull requests.

Today: change → commit → push → review → merge



Everyone has a local history. Sharing happens when repositories synchronize.

Figure: Pro Git • Chacon & Straub • CC BY-NC-SA 3.0



A repository is a project with a history

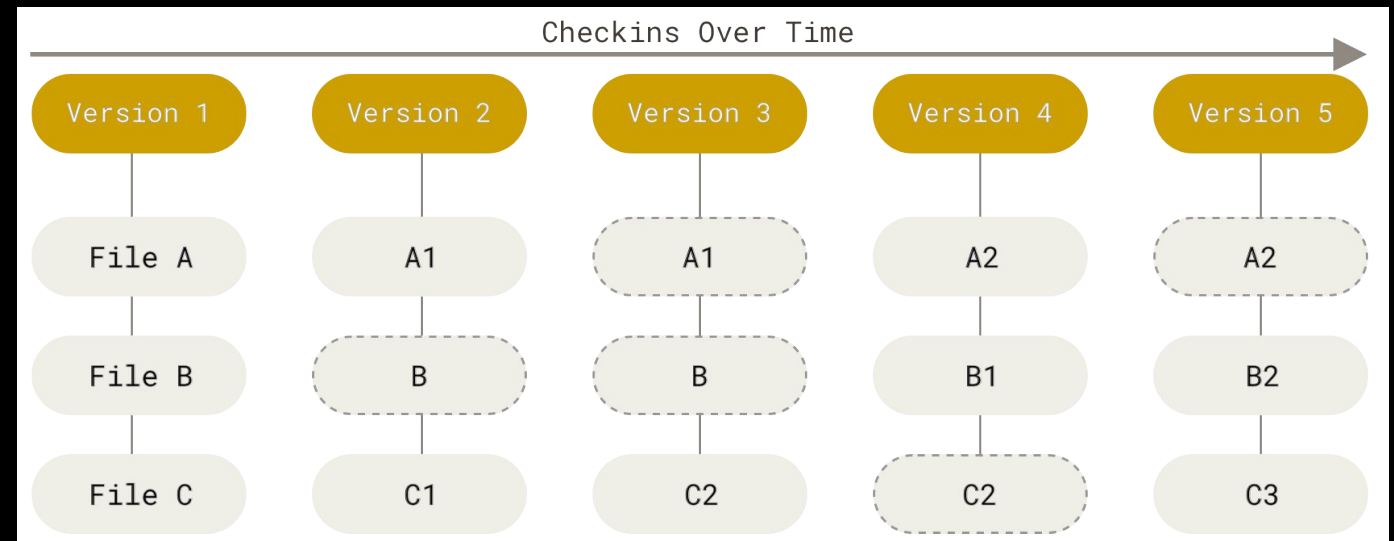
Repository (repo)

Project files plus the history stored in the .git directory.

Commit

A saved snapshot with an author, a message, and a unique ID.

```
git log --oneline  
git show <commit-id>
```



Read left to right: each column is a saved project version.

Use a clear commit message: “Document simulator setup”, rather than “updates”.

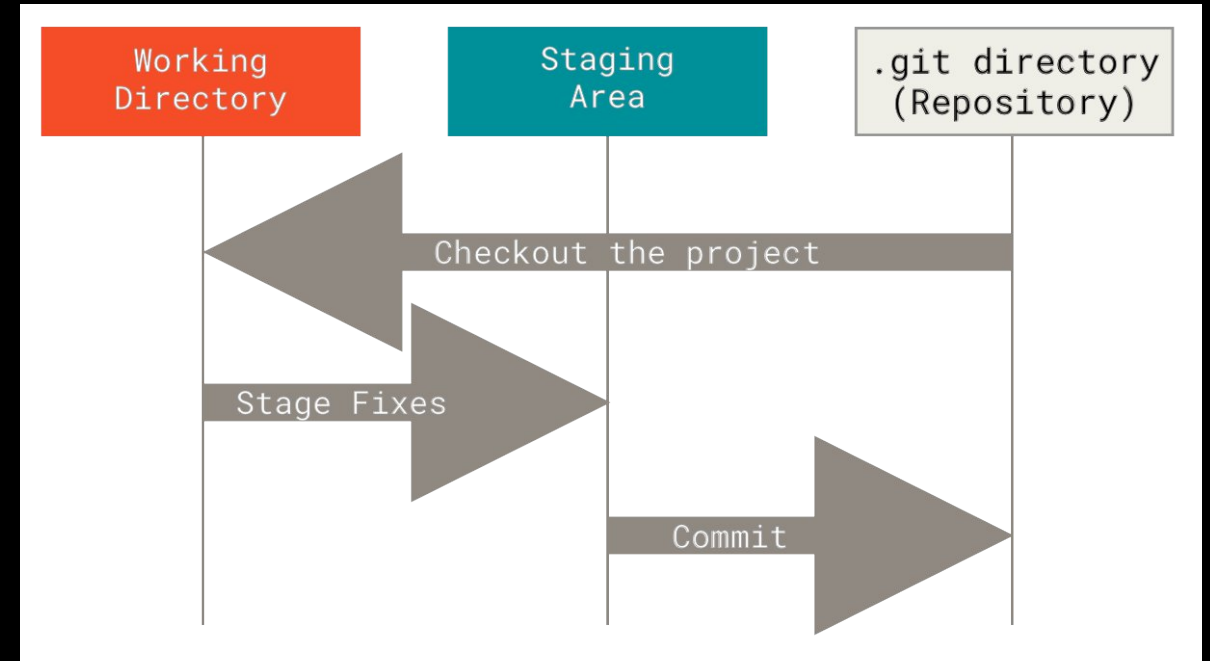
[Figure: Pro Git • Chacon & Straub • CC BY-NC-SA 3.0](#)



The local loop: edit → stage → commit

- 1 Edit files in your working tree.
- 2 Stage the changes you want to save.
- 3 Commit one logical improvement.

```
git status
git diff
git add README.md
git diff --staged
git commit -m "Document setup"
```



Stage = choose the contents of the next commit.

Saving a file, making a commit, and publishing it are three separate actions.

Figure: Pro Git • Chacon & Straub • CC BY-NC-SA 3.0



Get the project: clone and fork

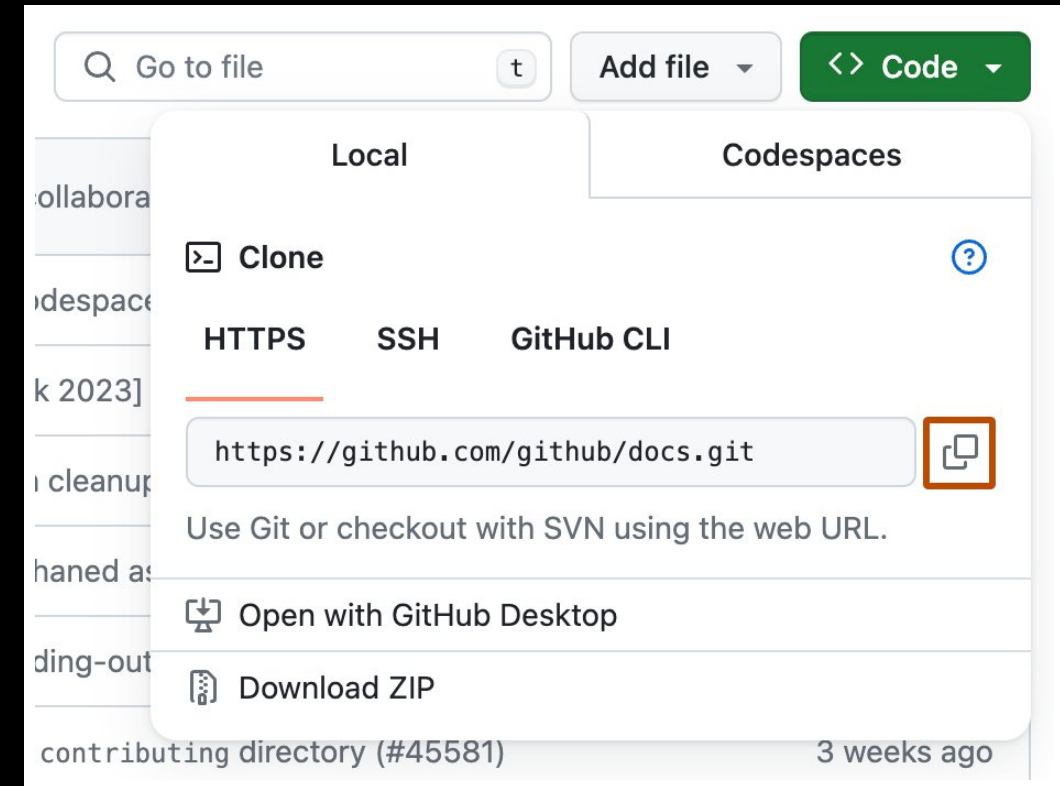
Clone → a local working copy

Copy an existing repository and its history to your laptop or cloud machine.

Fork → your own GitHub copy

Useful when you cannot push to the original project. Then clone your fork.

```
git clone <url> robot-project
cd robot-project
git remote -v
```



Copy the repository URL from Code. “origin” normally names the repository you cloned.

[Screenshot: GitHub Docs • CC BY 4.0](#)



Use a branch for each focused change

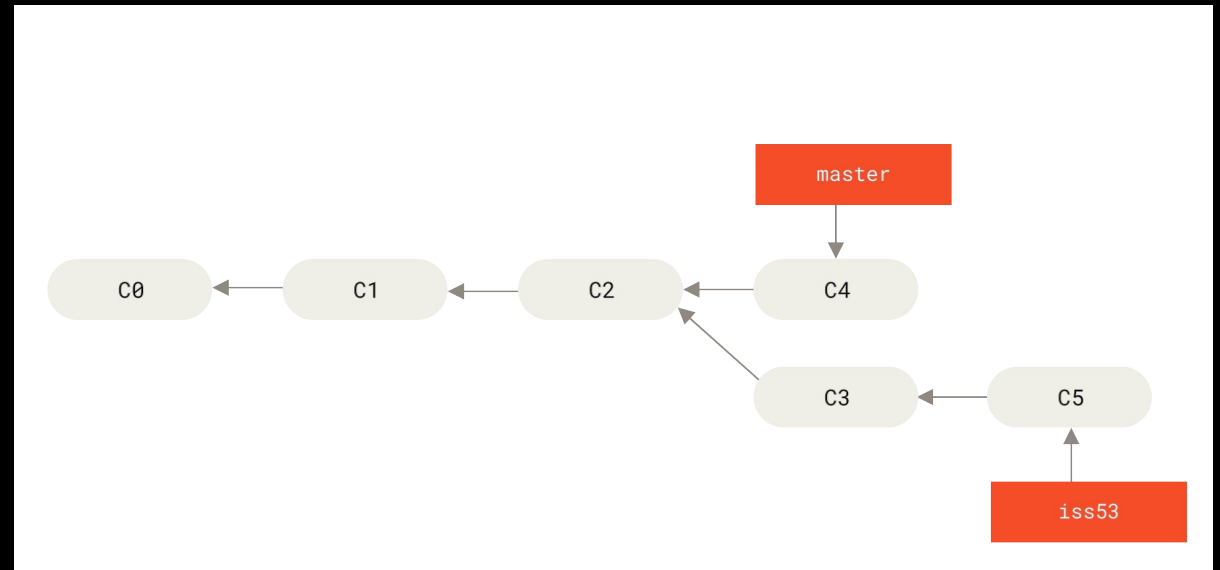
main = the shared baseline

Keep the team's main branch usable.

A topic branch = one task

A branch points to its latest commit.
New commits advance that branch.

```
git switch main
git pull --ff-only
git switch -c docs/setup
```



Book labels: master ≈ main; iss53 ≈ our feature branch.

Start from an up-to-date main, then develop and commit your change on the topic branch.

[Figure: Pro Git • Chacon & Straub • CC BY-NC-SA 3.0](#)



Synchronize: push, fetch, and pull

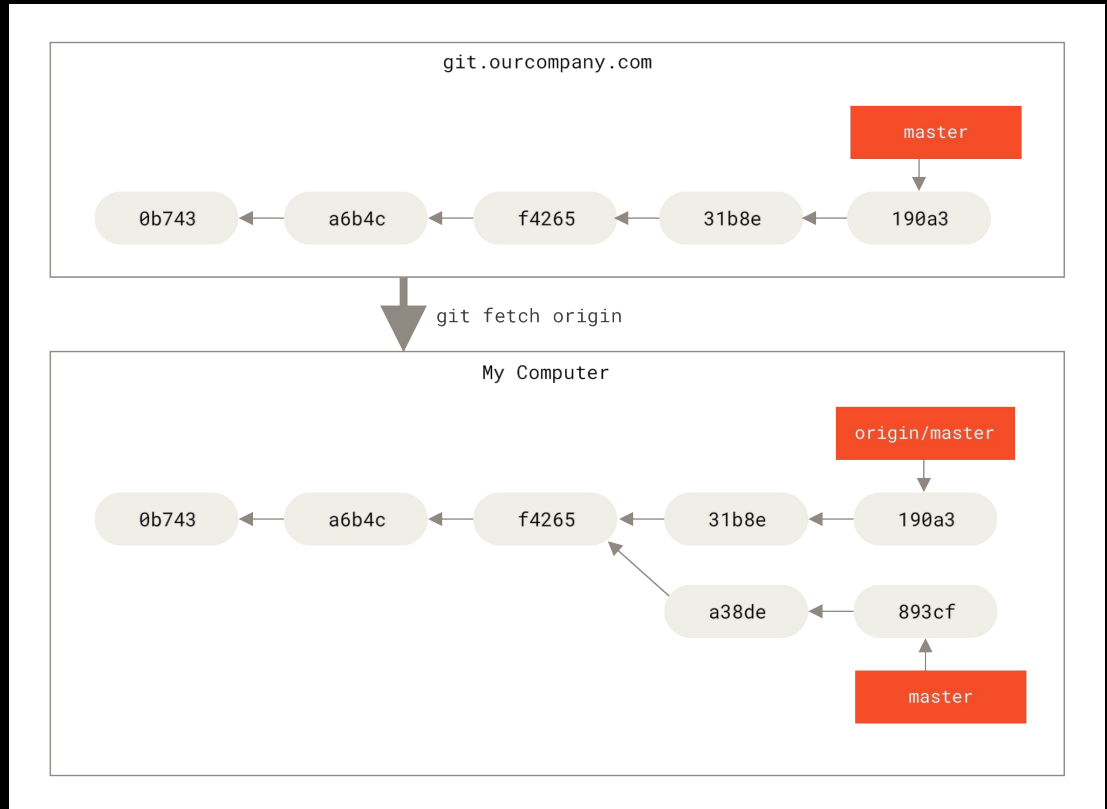
Push: publish your local commits.

```
git push -u origin docs/setup
```

Fetch: download the latest history.
Your working files stay as they are.

```
git fetch origin
```

Pull = fetch + integrate
into your current branch.



A commit is local. A push publishes it. A pull request proposes integrating it.

Figure: Pro Git • Chacon & Straub • CC BY-NC-SA 3.0



Coordinate the work before changing code

Define the task in an issue

State the goal, owner, and what “done” means before starting implementation.

Make progress visible

Use a Project board to track tasks and avoid duplicate work.

Example issue #12

“Document simulator setup”

Done: a teammate can follow the steps.

Create new issue

github/OctoArcade

Add a title

Assignee (a) Labels (l) Projects (p) Milestone (m)

Leave a comment

Cancel Create more Create

One focused issue → one branch → one reviewable pull request.

[Screenshot: GitHub Docs • CC BY 4.0](#)



Open a pull request to propose the change

 octo-repo had recent pushes less than a minute ago

Compare & pull request

Choose the direction

base: main ← **compare: docs/setup**

Explain why, what changed, and how you checked it. Link the issue. Request a teammate's review; use Draft if the work is unfinished.

EXAMPLE PR DESCRIPTION

Document simulator setup

Why: unblock first-time setup.

Change: add steps to README.

Checked: teammate ran the steps.

Closes #12

A pull request is a proposal plus discussion, checks, and review. It is not yet a merge.

[Screenshot: GitHub Docs • CC BY 4.0](#)

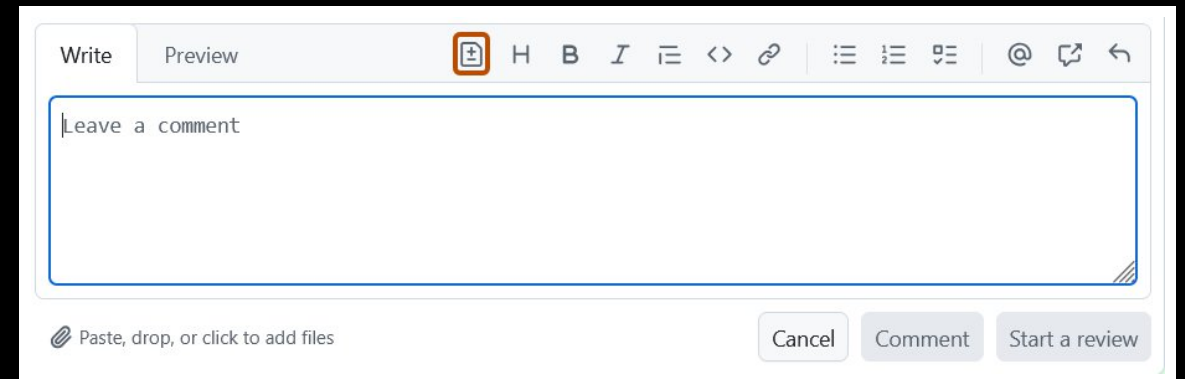
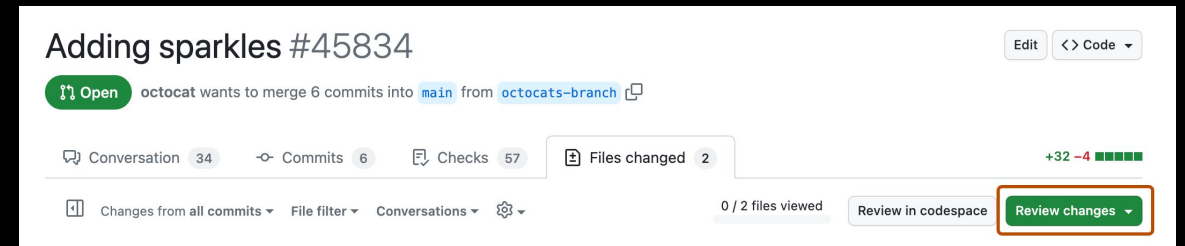


Review the change, then improve it together

Read Files changed and ask:

- Does this solve the stated task?
- Is the change clear and focused?
- Do the tests or setup steps work?

Leave specific, constructive feedback.
The author fixes it, commits, and
pushes to the same branch.



Comment / Approve / Request changes

Review is a conversation. New pushes update the existing pull request.

[Screenshot: GitHub Docs](#) • CC BY 4.0



Resolve conflicts when changes overlap



This branch has conflicts that must be resolved

Use the [web editor](#) or the [command line](#) to resolve conflicts.

Conflicting files

content/issues/tracking-your-work-with-issues/linking-a-pull-request-to-an-issue.md

Resolve conflicts

Git needs a decision from the team.

- 1 Read both versions and agree on intent.
- 2 Edit the final contents; remove markers.
- 3 Test, stage, commit, and push.

Example: two teammates edit the same setup instruction in README.md.

```
git switch docs/setup
git fetch origin
git merge origin/main
# If README.md conflicts:
# edit it, remove markers, test
git add README.md
git commit
git push
```

Keep both authors' intent. A resolved conflict still needs a behavior check.

[Screenshot: GitHub Docs • CC BY 4.0](#)



Merge, then bring everyone up to date

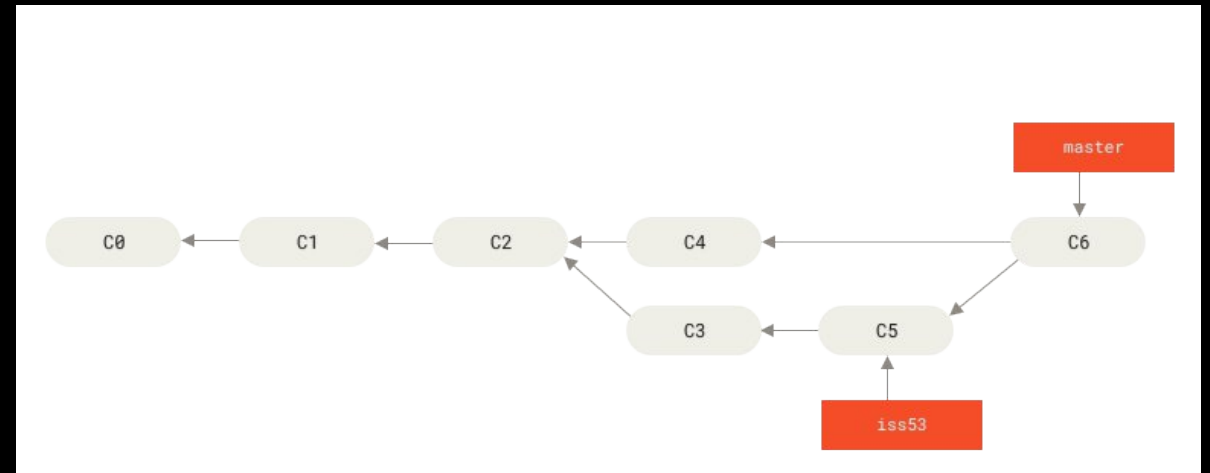
Before merging

Reviews addressed, relevant checks passing, and conflicts resolved.

After merging

The change is part of main.
Delete the finished remote branch.

```
git switch main  
git pull --ff-only
```



Example: a merge commit joins both histories.
The book calls the main branch “master”.

Merge on GitHub; pull on each computer. Review approval alone does not update main.

[Figure: Pro Git • Chacon & Straub • CC BY-NC-SA 3.0](#)



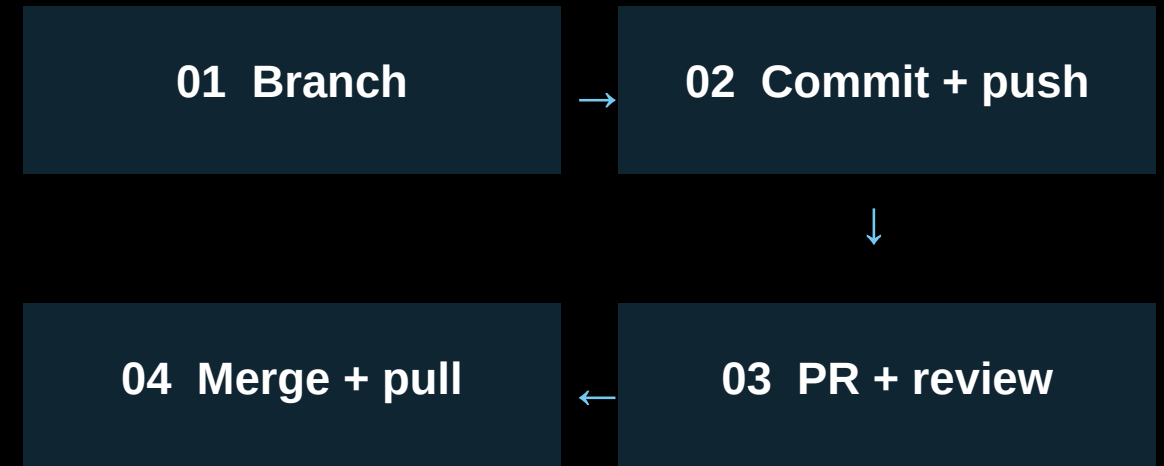
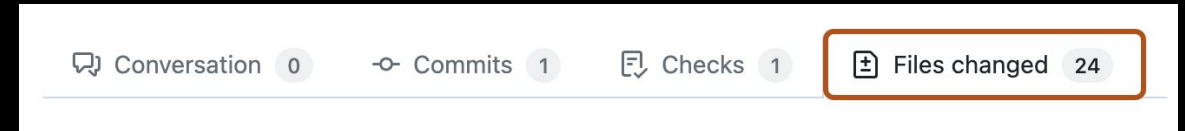
Pair exercise: your first reviewed change

Author

Create a branch, improve README.md, commit, push, and open a PR. Link the issue and describe your check.

Reviewer

Request one useful improvement. Review the update, approve, and merge. Both teammates pull the new main.



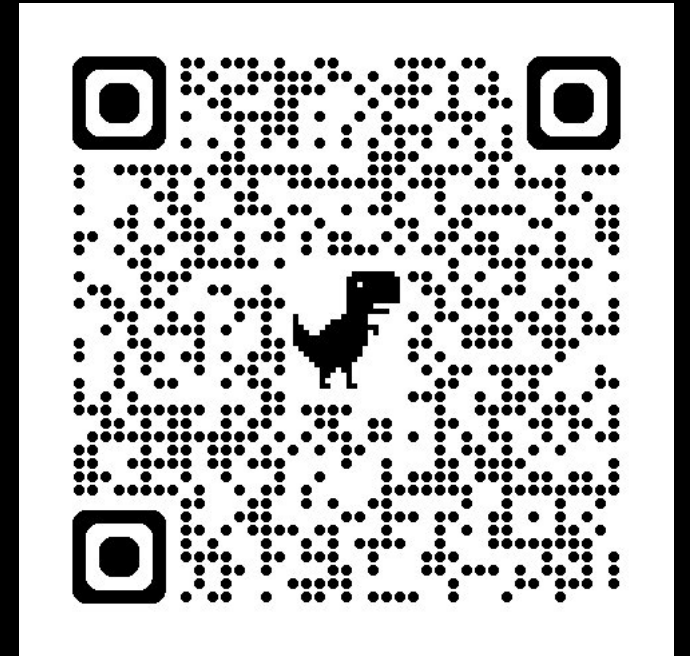
Done = a merged PR, a useful review conversation, and the updated main on both machines.

[Screenshot: GitHub Docs](#) • CC BY 4.0

Announcement



**Please fill in the form
for your github account
for access to the code
and compute**





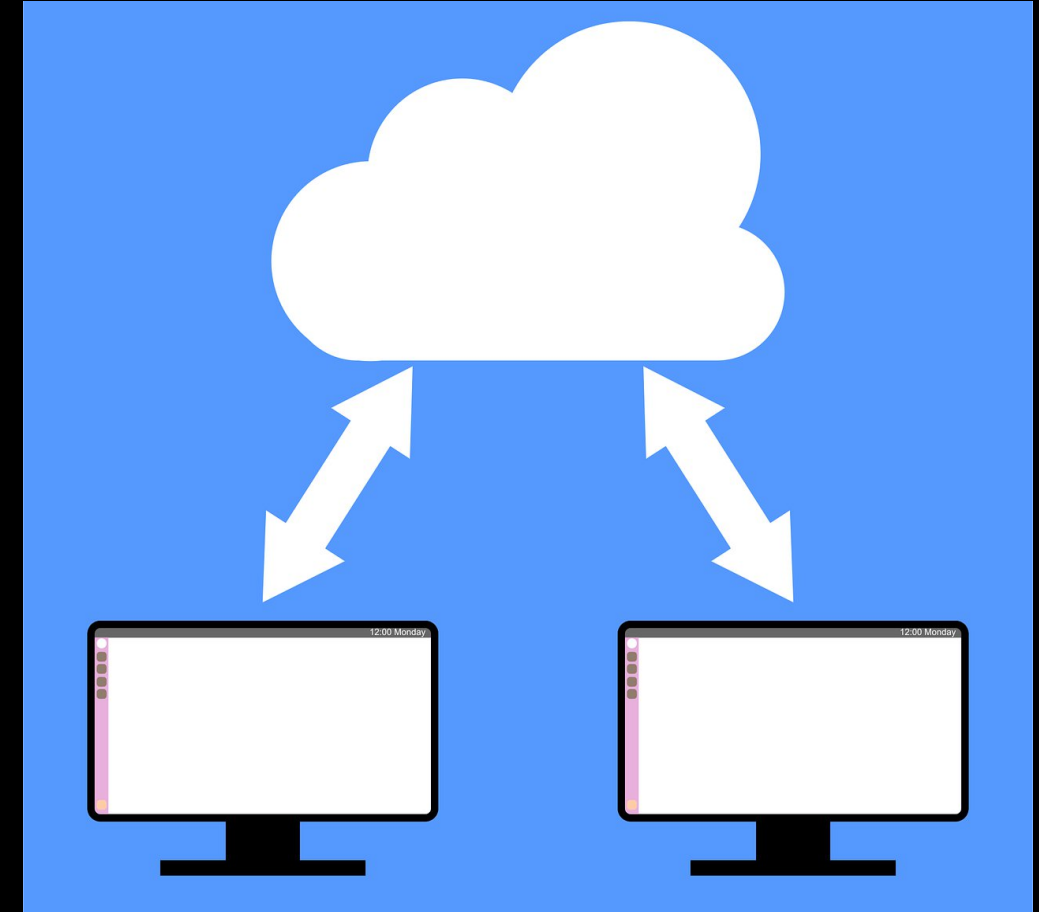
SSH and Cloud-computing platform

Personal computing instance with powerful GPUs to use for training and simulation

Accessible remotely via SSH

Turns off only after lack of usage. You can leave it “running” for hours at a time.

If you have the power locally, you’re welcome to use it



Connecting through SSH



Create your ssh key with [ssh-keygen](#)

SSH into your instance

ssh username@ip

Config for convenient usage:

Add the following into ~/.ssh/config

Example

Host <name>

HostName <ip>

User <user name>

IdentityFile <the ssh key you created>

IdentitiesOnly yes

Copy the files from/to the server

scp source destination

rsync -avzP source destination

- **a: Archive**
- **v: Verbose**
- **z: Zip**
- **P: Progress + partial**

Host microagi

HostName 10.5.13.126

User yangcyself

IdentityFile ~/.ssh/microagi

IdentitiesOnly yes

Vscode Remote



1. Install Conda with Miniforge

Follow the Miniforge guide on your new GPU instance.

01

Download

Linux x86_64 installer

Check: `conda --version`

02

Install

In your home directory

03

Open

A new terminal

[Open the official Miniforge installation guide](https://github.com/conda-forge/miniforge) →

github.com/conda-forge/miniforge



2. Create a Python environment

Use Python 3.11 for this workshop.



```
conda create -n env_isaaclab python=3.11
conda activate env_isaaclab
```

Install Isaac Sim, Isaac Lab, and faive_lab in this environment.

[Open the Conda environment guide](#) →

[docs.conda.io · Managing environments](https://docs.conda.io/en/latest/Managing%20environments.html)



3. Install Isaac Sim

Follow “Installing Isaac Sim” in the official guide.

ISAAC SIM

5.1.0

Simulator runtime

PYTORCH

2.7.0 + cu128

GPU learning library

Choose Linux x86_64. Keep env_isaaclab active.

[Open the official Isaac Sim installation steps](#) →

[Isaac Lab 2.3.2 docs · Installation using Isaac Sim Pip Package](#)



4. Install flatdict before Isaac Lab

Prevents the “No module named pkg_resources” error.

```
python -m pip install "pip>=25.3"
printf 'setuptools<81\n' > "$HOME/isaac-build-constraints.txt"
export PIP_BUILD_CONSTRAINT="$HOME/isaac-build-constraints.txt"
python -m pip install "flatdict==4.0.1"
```

Wait for success. Keep this terminal open for the next step.

[Why this works: pip build constraints](#) →

[pip documentation · Constrain setuptools in the temporary build environment](#)



5. Install Isaac Lab

Follow “Installing Isaac Lab”; use release v2.3.2.

```
git clone --branch v2.3.2 \
  https://github.com/isaac-sim/IsaacLab.git
cd IsaacLab
./isacclab.sh --install rsl_rl
```

Keep the flatdict build constraint active in this terminal.

[Open the official Isaac Lab installation steps](#) →

[Isaac Lab 2.3.2 docs · Installing Isaac Lab](#)



6. Check that the simulator starts

Run the empty-scene example from your IsaacLab folder.

```
./isaacsim.sh -p \
  scripts/tutorials/00_sim/create_empty.py --headless
```

Wait for startup, then stop with Ctrl+C.

[Open the official verification steps](#) →

[Isaac Lab docs · Verifying the Isaac Lab installation](#)



7. Set up Git LFS

Git LFS downloads the large assets used by faive_lab.

```
conda install -c conda-forge git git-lfs  
git lfs install
```

Set up Git LFS before cloning the project.

[Open the official Git LFS setup guide](#) →

[docs.github.com](https://docs.github.com/en/github/working-with-github-lfs) · Installing Git Large File Storage



8. Clone faive_lab and its assets

With GitHub SSH access ready, clone inside ~/RWR.

```
git clone --branch rwr2026 \
  git@github.com:srl-ethz/faive_lab.git
cd faive_lab
git lfs pull
```

git lfs pull downloads the assets for this checkout.

[Open the faive lab workshop branch](#) →

github.com/srl-ethz/faive_lab · rwr2026 · Repository access required



9. Install the project

From the faive_lab folder, install the project package.

```
python -m pip install -e source/faive_lab
```

Next: follow the project README and run a small training test.

[Open the project setup instructions](#) →

[faive_lab · rwr2026 README](#)



Isaac Lab and IsaacSim

Isaac Lab is a robotics research platform built on IsaacSim

GPU Accelerated physics simulation with PhysX

Massively parallel environments

RL **How does this help train policies better?**

Works with URDF, MJCF and CAD



Robot description files



URDF

```
<joint type="revolute" name="right_thumb_abd">
  <parent link="right_thumb_mp"/>
  <child link="right_thumb_abd_jointbody"/>
  <origin xyz="6.938893903907228e-18 0.0 0.0" rpy="-1.22173 0.0 0.0"/>
  <axis xyz="0.0 0.34202085907197166 0.9396923602752502"/>
  <limit lower="-1.08211" upper="0.0" effort="100" velocity="100"/>
</joint>
<joint type="fixed" name="right_thumb_abd_offset">
  <parent link="right_thumb_abd_jointbody"/>
  <child link="right_thumb_pp"/>
  <origin xyz="-0.03 -0.0 -0.0" rpy="0.0 0.0 0.0"/>
</joint>
<link name="right_thumb_ip">
  <inertial>
    <origin xyz="0.00135 2e-05 0.01383" rpy="0.0003700000000000001 0.032630000
    <mass value="0.01402"/>
    <inertia ixx="1.7e-06" iyy="1.64e-06" izz="5.9e-07" ixy="0" ixz="0" iyz="0
  </inertial>
  <visual name="right_visual_thumb_ip_mesh">
    <origin xyz="0.0012998948492342203 7.159773213648307e-06 0.013619877706625
    <geometry>
      <mesh filename="package://orcahand_description/assets/urdf/right/visual
    </geometry>
    <material name="white"/>
  </visual>
  <collision name="right_collision_thumb_ip_mesh">
    <origin xyz="0.0013560375396256981 0.00012710825491742077 0.01344632766073
    <geometry>
      <mesh filename="package://orcahand_description/assets/urdf/right/collis
    </geometry>
```

USD

```
orcahand_right (defaultPrim)
├── right_forearm
├── right_tower
├── joints
│   ├── right_tower_fixed
│   ├── right_wrist
│   ├── right_index_abd
│   ├── right_index_mcp
│   ├── right_index_pip
│   ├── right_middle_abd
│   ├── right_middle_mcp
│   ├── right_middle_pip
│   ├── right_pinky_abd
│   ├── right_pinky_mcp
│   ├── right_pinky_pip
│   ├── right_ring_abd
│   ├── right_ring_mcp
│   ├── right_ring_pip
│   ├── right_thumb_cmc
│   ├── right_thumb_abd
│   ├── right_thumb_mcp
│   └── right_thumb_dip
├── right_carpals
├── visuals
├── collisions
├── right_index_ap
├── right_index_pp
├── right_index_fingertip_assembly
├── right_middle_ap
├── right_middle_pp
├── right_middle_fingertip_assembly
├── right_pinky_ap
├── right_pinky_pp
├── right_pinky_fingertip_assembly
├── right_ring_ap
├── right_ring_pp
├── right_ring_fingertip_assembly
├── right_thumb_tp
└── right_thumb_ap
```

Isaac sim remote streaming



Local Client

- Download stream client on your local device

[Isaac Sim 5.1.0 Download](#)

- Check the Live Visualization IP from the collab.microtrack.ai and paste into the client
- Click connect

Remote Server

- Start streaming on the remote server
`isaacsim isaacsim.exp.full.streaming --no-window`

Live visualization (Isaac Sim)

Run an Isaac Sim / Isaac Lab **demo** script with `--livestream 2` (Isaac Sim WebRTC Streaming Client over WARP and connect to:

10.4.48.29

Stream a *demo* (scripts/demos/*.py), not a training script — training requires a relaunch. WebRTC ports are opened from the WARP tunnel by research

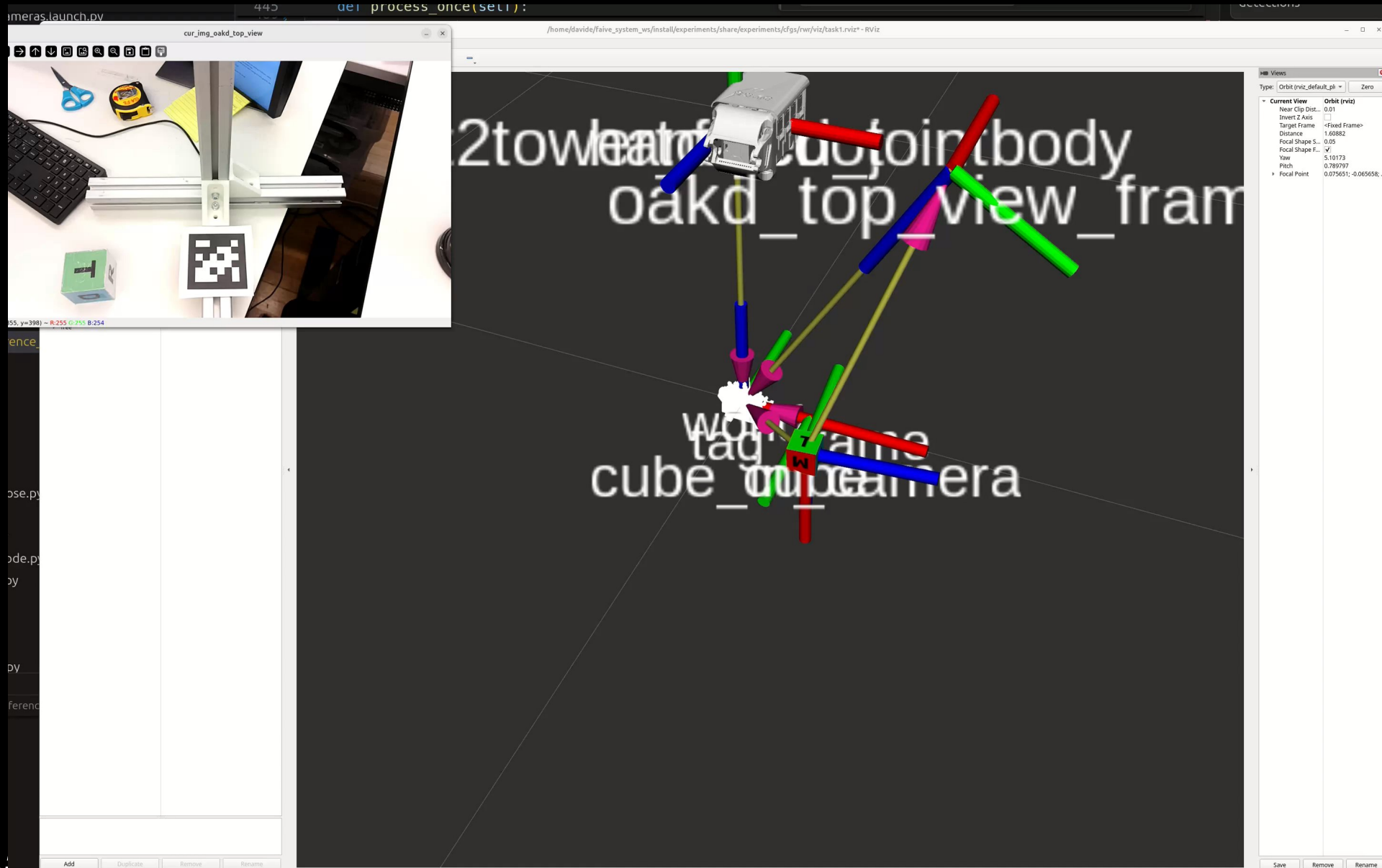
Isaac Sim WebRTC Streaming Client

Server:

Resolution:

Connect

RVIZ





FAIVE LAB

Our own environment for
IsaacLab-based projects

Includes all the necessary setup
for the ORCA Hand

Isolated environment for RL
Training

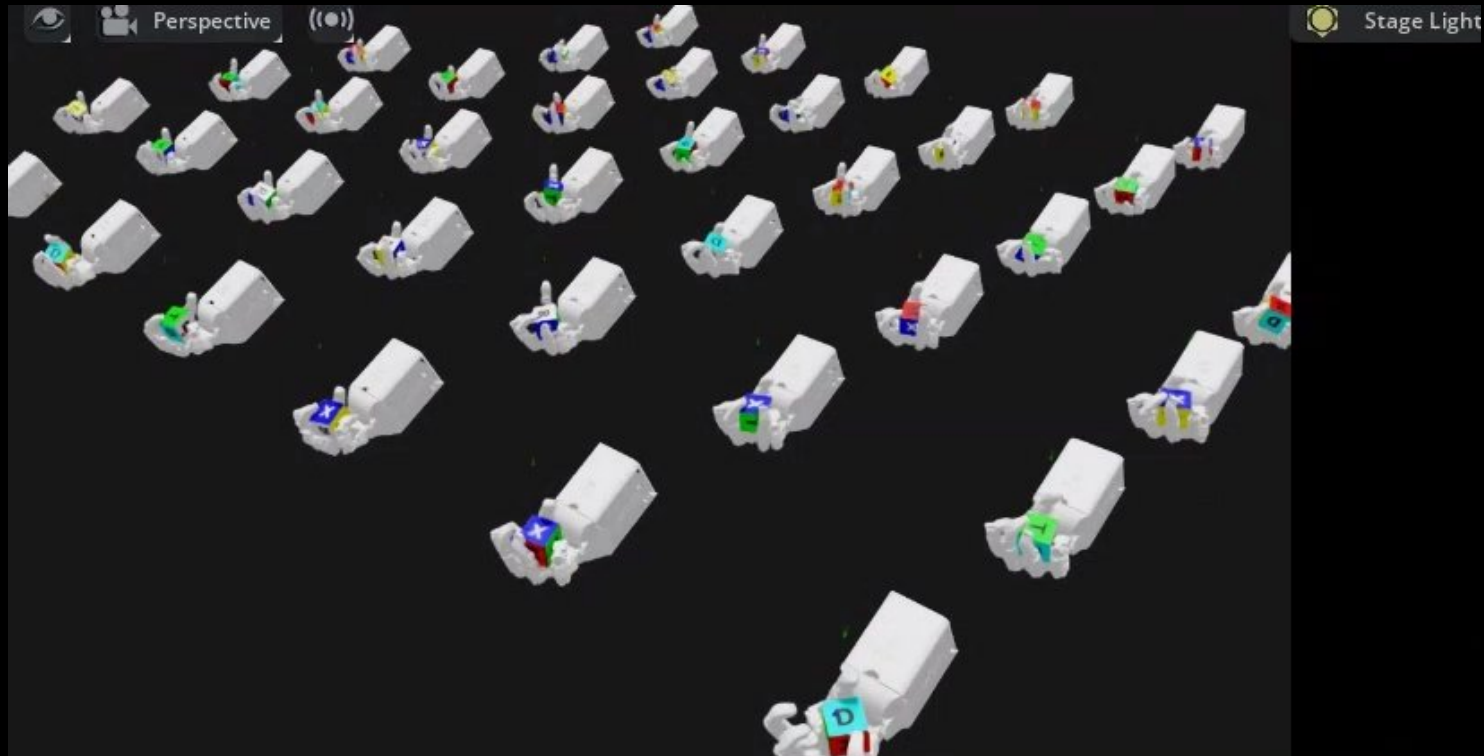
Fork it from GitHub on your AWS
instance and follow the
installation guide





Training a policy

```
python3 scripts/rsl_rl/train.py --task=orca-inhand-repose-v0 --logger wandb --  
headless --num_envs=8192 --video --video_interval=4000
```





Viewing the simulation

Option 1:

Run your training without

--headless

Let IsaacSim launch and show the “gym” environment

Connect with Amazon DCV remote desktop to view

The simulation doesn't need to be visible for it to be running

Be patient, it takes a while to load.

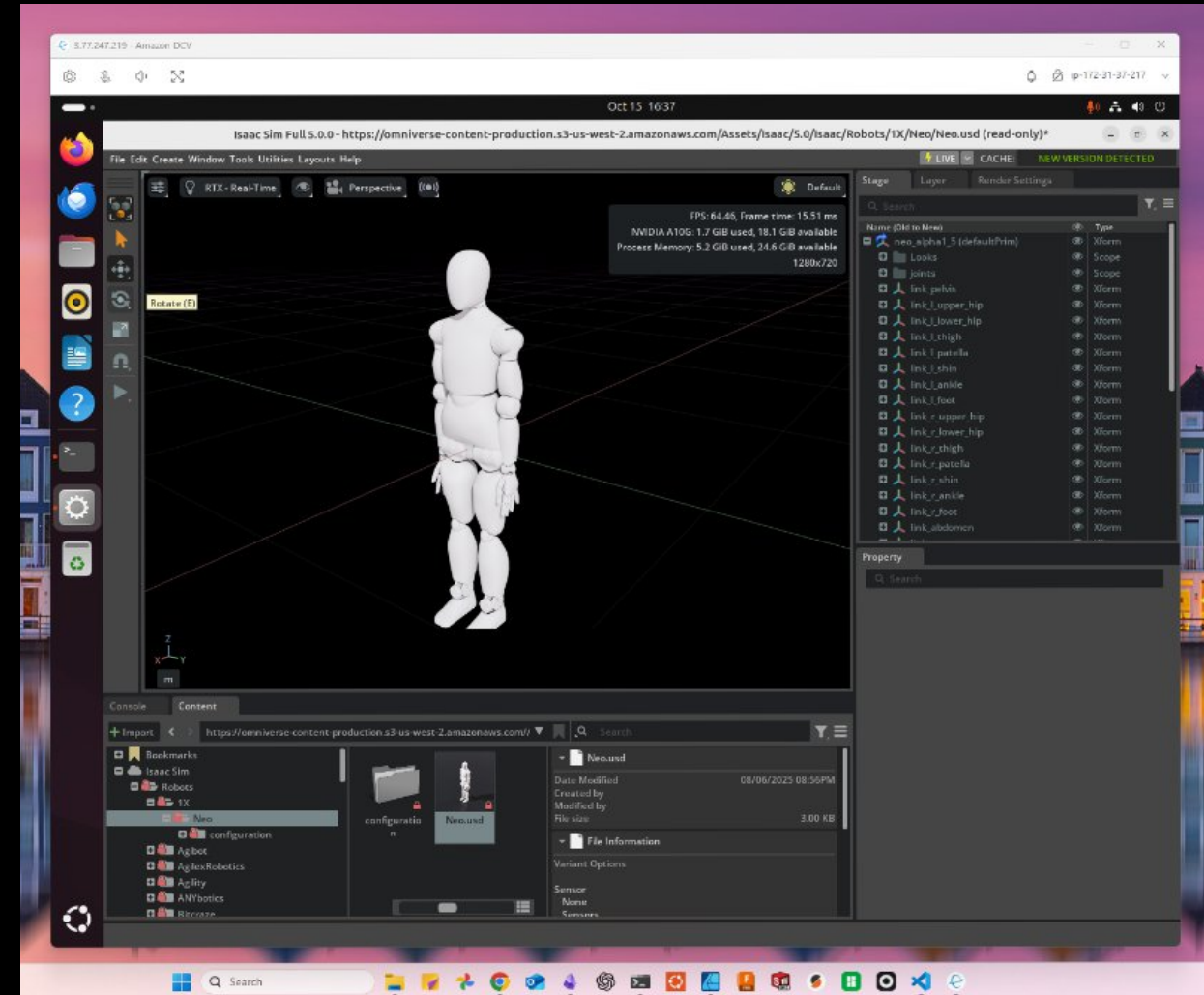
Option 2:

Record videos every N episodes

--video=True

--video_interval=2000

Transfer the videos through SCP or view them through VSCode SSH





Logging and Monitoring with Weights and Bias

AI Developer platform for tracking training

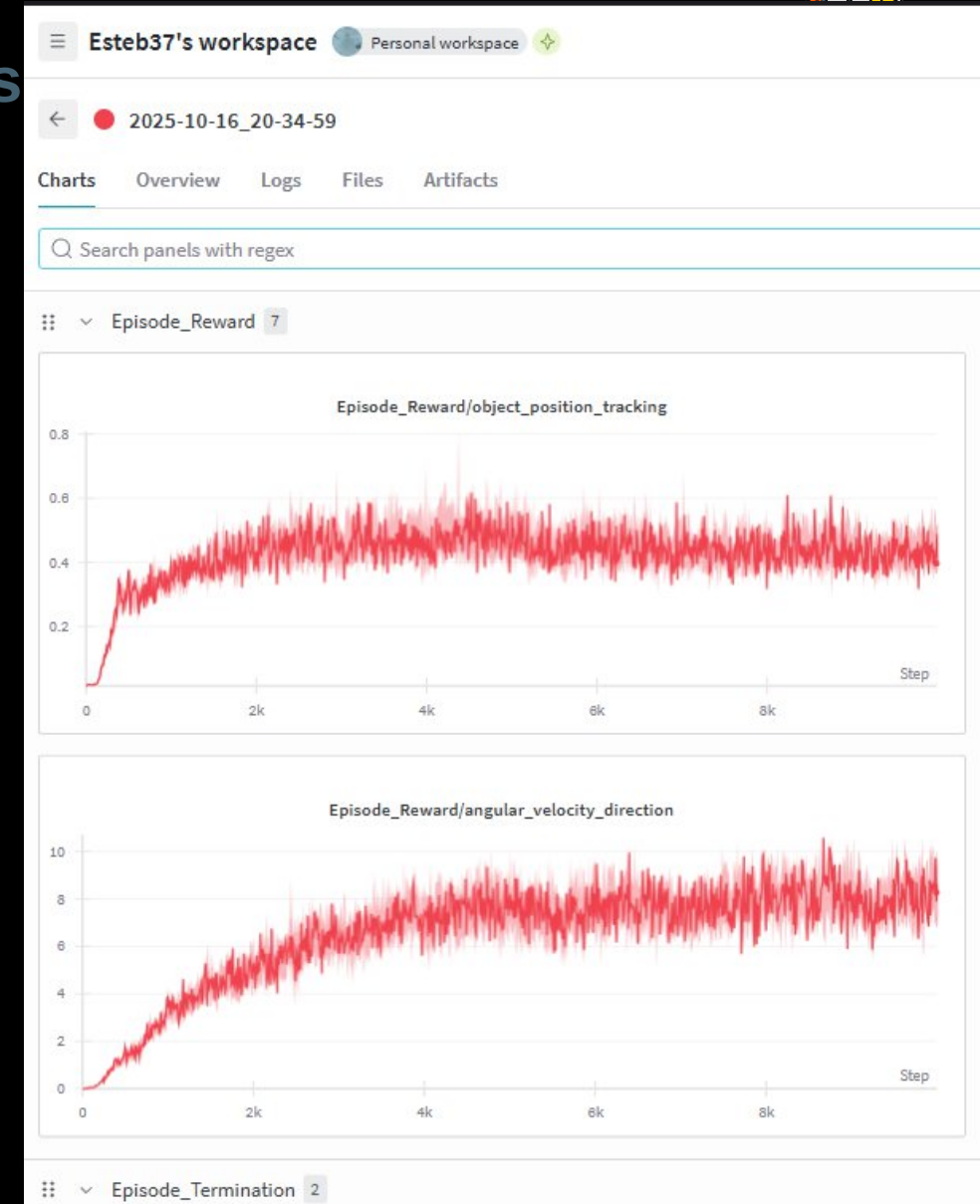
View terminal log history with timestamps

View progress in rewards, losses, mean episode length, etc

Share data with your team

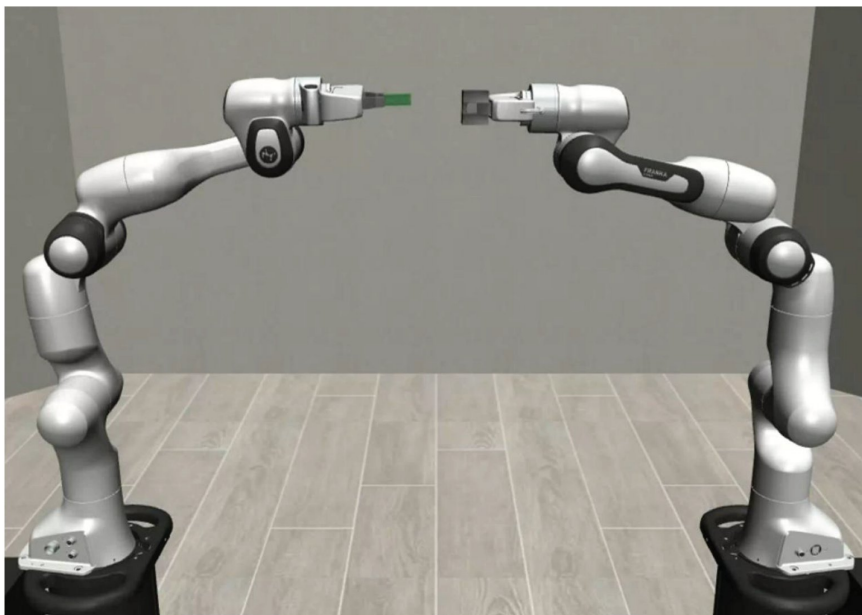
Compare different runs

wandb.ai

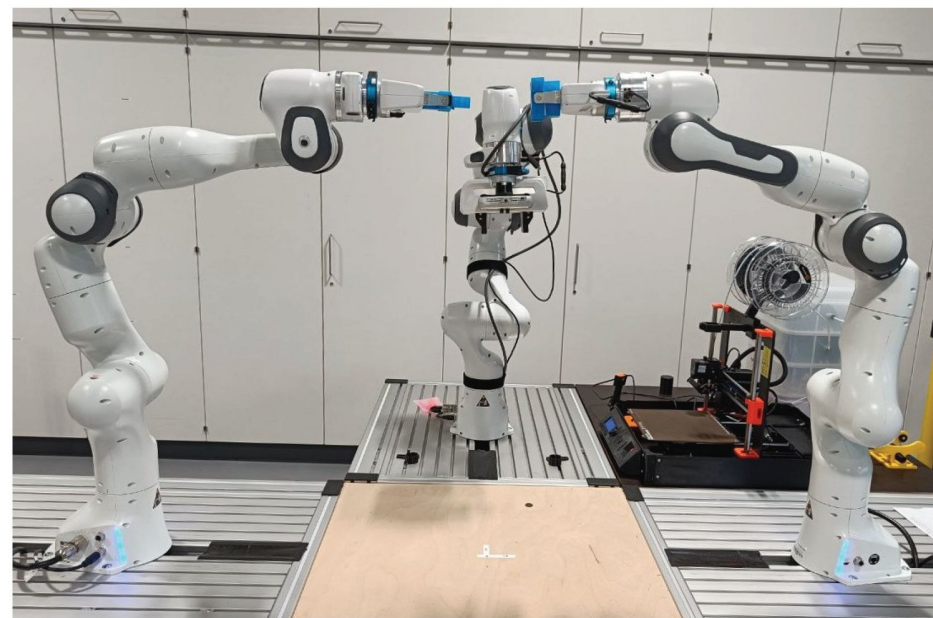




Sim2Real Gap



(a)



(b)



Domain Randomization



System Identification

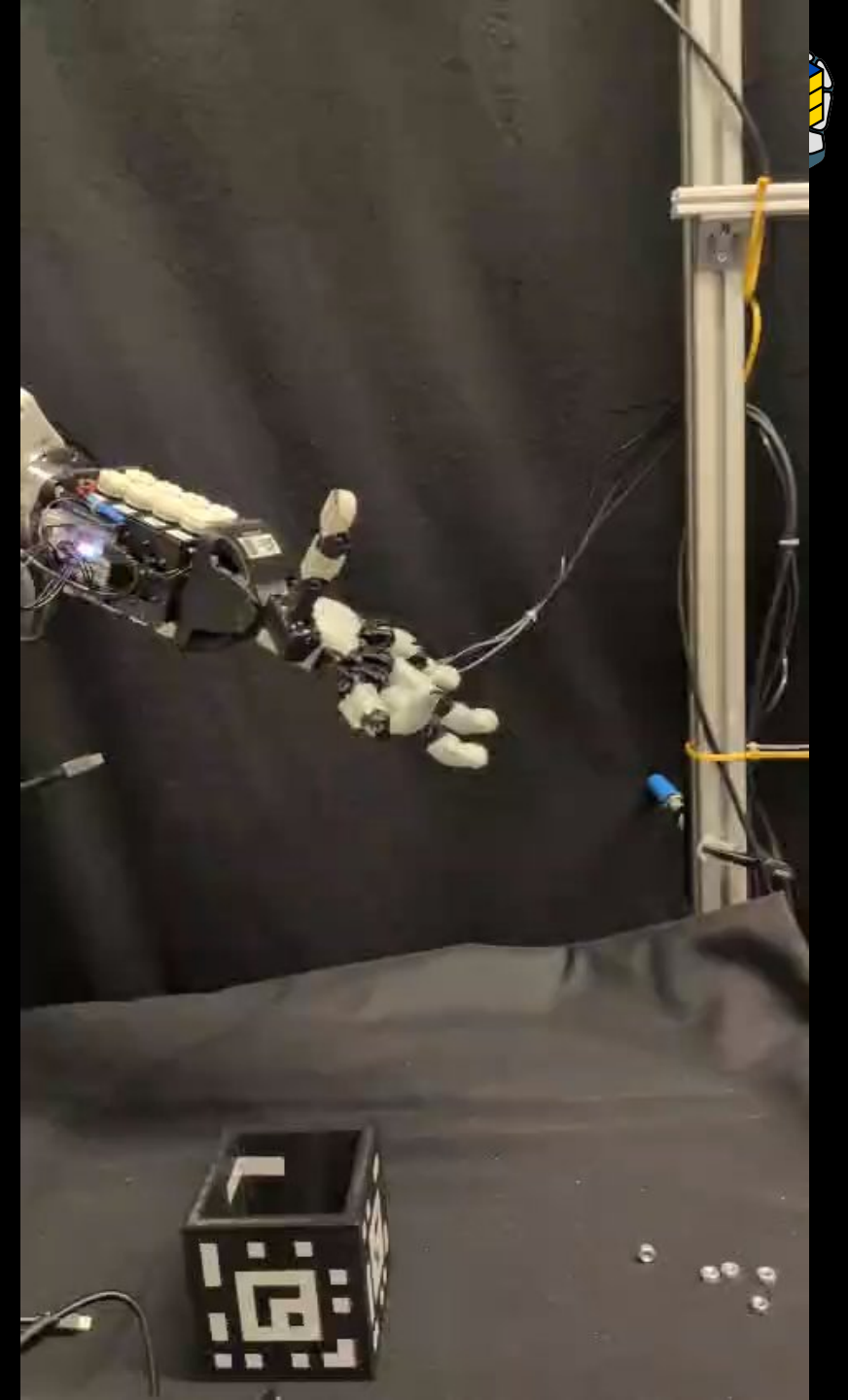
Running experiments on both your real hardware and your simulated software to find values on physical characteristics

Friction

Damping

Stiffness

The resulting simulation captures the real dynamics more accurately





Real World Deployment

Your training session will produce a PyTorch model

Inference Node

We will give you a guide on how to deploy this model on the workstation after you've connected your own hands





Task 1 breakthrough and evaluation

1. Setup training of a proprioception-only in-hand cube reorientation around the z-axis
2. Deploy the trained policy to the physical Orca hand
3. Implement domain randomization and system identification to improve sim2real transfer
4. Train a policy that uses the cube pose from the camera and deploy it to the real world
5. (bonus) Train and deploy a policy that can reorient the cube to any given pose

Deadline: November 09th

Each team will showcase both the proprioception-only policy and the cube pose feedback policy during the task 1 showcase.

Each deployed policy will be evaluated with a score that is a function of average rotational speed and amount of rotation before falling. Score will be averaged between 10 runs

Minimum passing criteria for this task: showcase *at least one* policy that rotates the cube around z axis for at least 10 seconds. The maximum grade is achievable also without the (bonus) subtask.



**FULL TASK DOCUMENT AND
CODE WILL BE RELEASED
TONIGHT.
BE PATIENT**