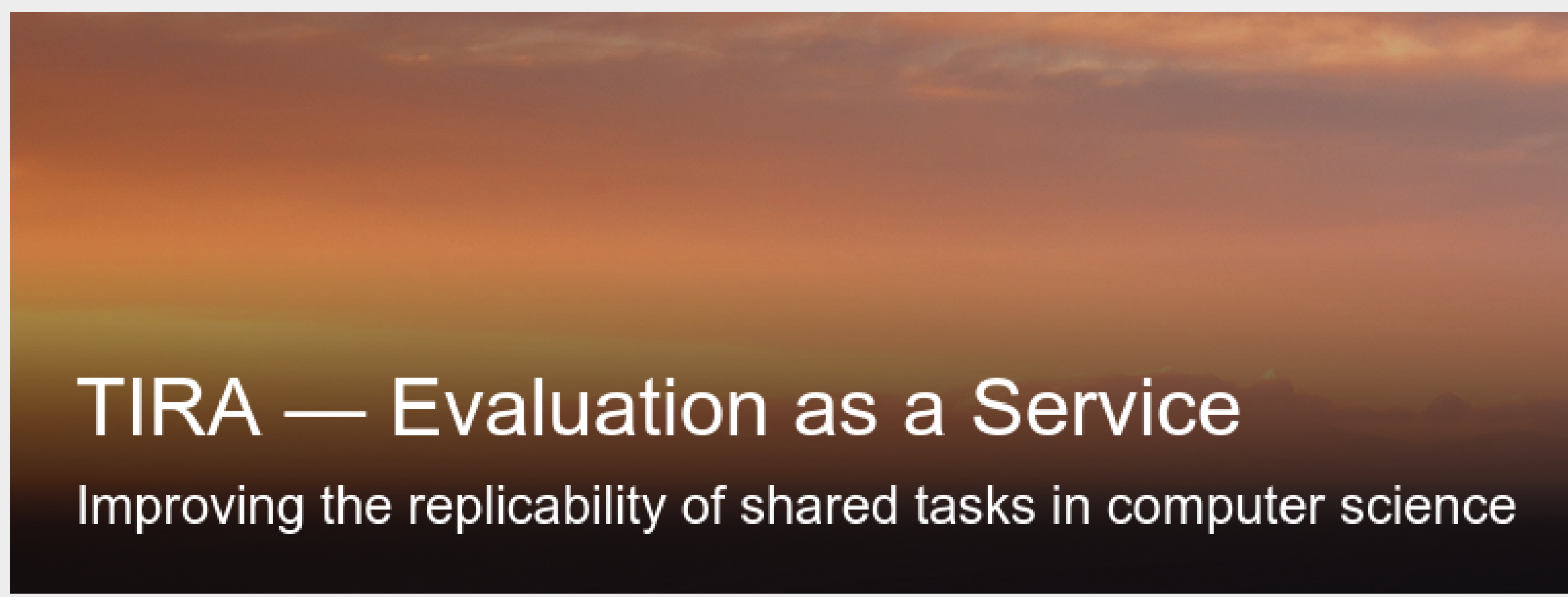


# Continuous Integration for Reproducible Shared Tasks with TIRA.io



TIRA in a Nutshell:

- Software Submissions + Blinded Experiments
- Sandboxing for execution on confidential data
- Complete export of shared task archive



<https://github.com/tira-io>

## Immutable Software Submissions

Implemented in Docker + Git CI/CD

- Shared task = git repository
- Software execution = commit

Technology Stack

- Image registry via Gitlab
  - Storage 12.4 PB HDD via a Ceph cluster (78 nodes)
- Kubernetes cluster for software execution
  - 130 nodes from a shared cluster (1,620 CPU cores, 25.4 TB RAM)
  - 24 dedicated GeForce GTX 1080 GPUs
- Adding additional runners is simple: E.g., add a runner on your laptop

## The Perspective of a Participant in a Shared Task

### Step 1: Implement Approach in Docker Image

- Docker image must be self-contained
- No internet access during execution

### Step 2: Local Testing

Participants can test their software locally

- Input: Public validation data
- Mirrors cloud execution and sandboxing

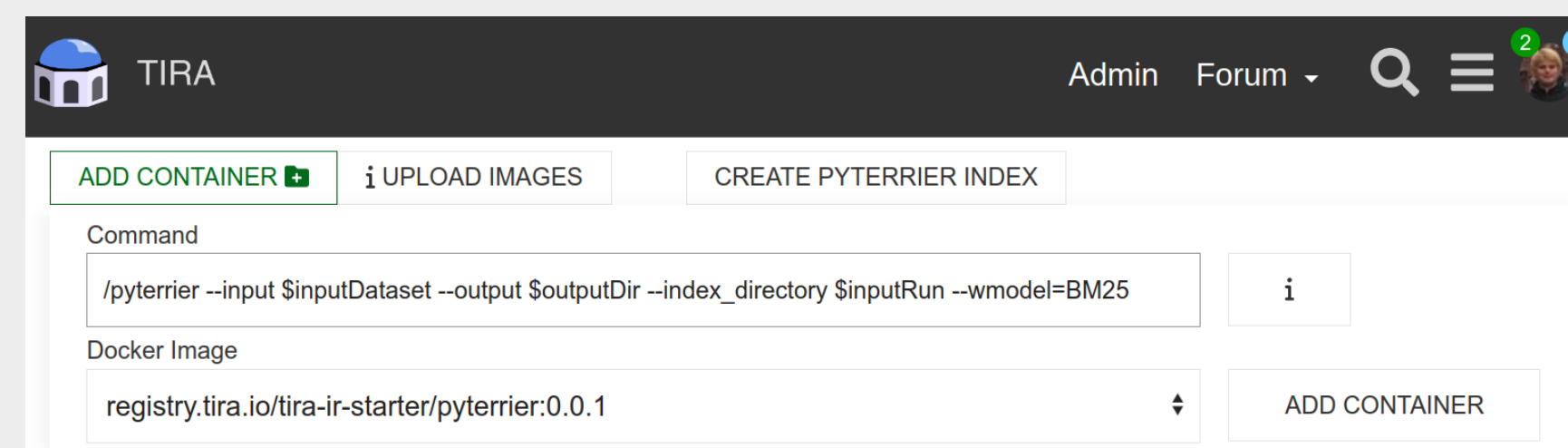
```
tira-run \
  --output-directory <OUTPUT> \
  --input-directory <INPUT> \
  --image <DOCKER-IMAGE> \
  --command <COMMAND>
```

### Step 3: Upload Image

- Each team has a dedicated image registry
- Upload via `docker push`

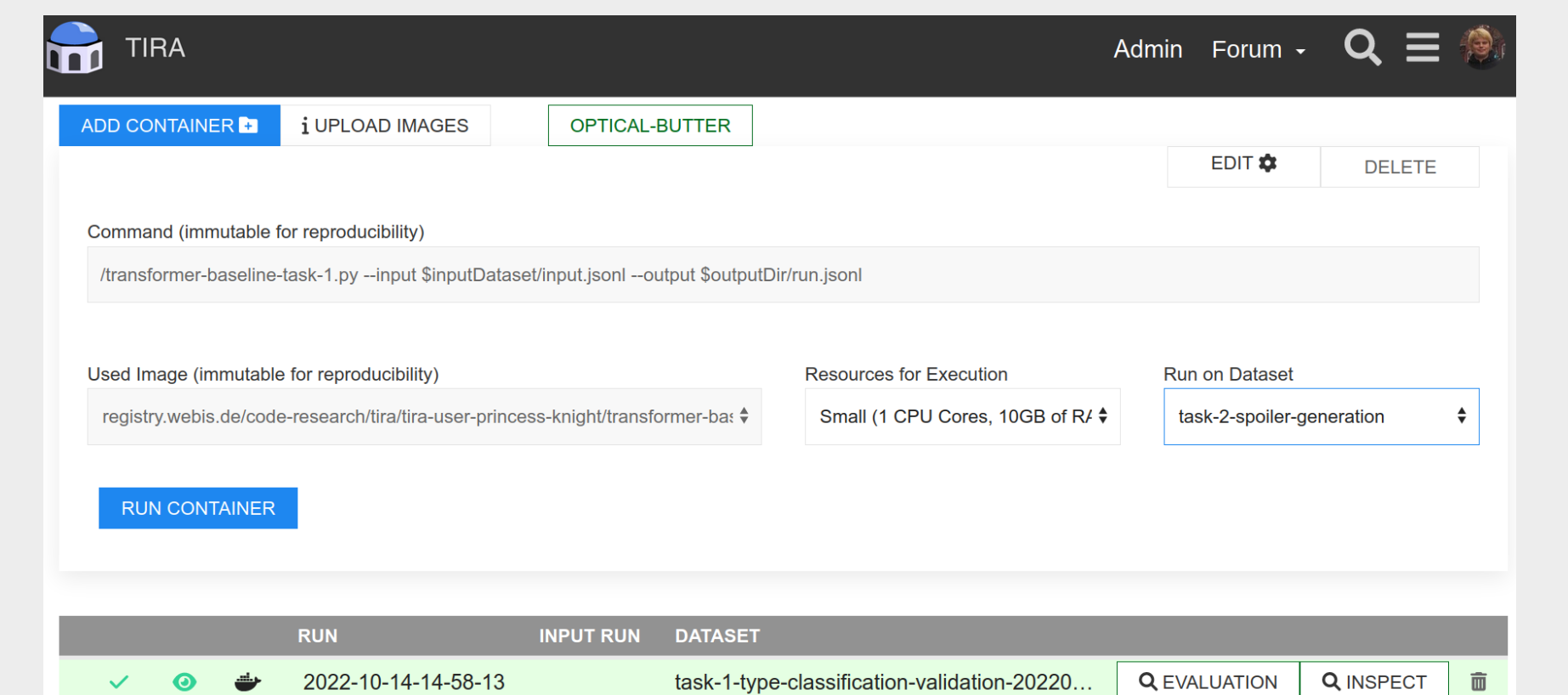
### Step 4: Configure Immutable Software

- Software = Docker image + command
- Immutability by retagging images
- Documentation: Paper + Description



### Step 5: Run Software

- Parallel software executions possible
- Validation vs. test executions
- Resources for execution can be specified
- E.g., CPU, GPU, etc.



## Organization of Shared Tasks with TIRA

### Requirements to organize a task in TIRA

Data:

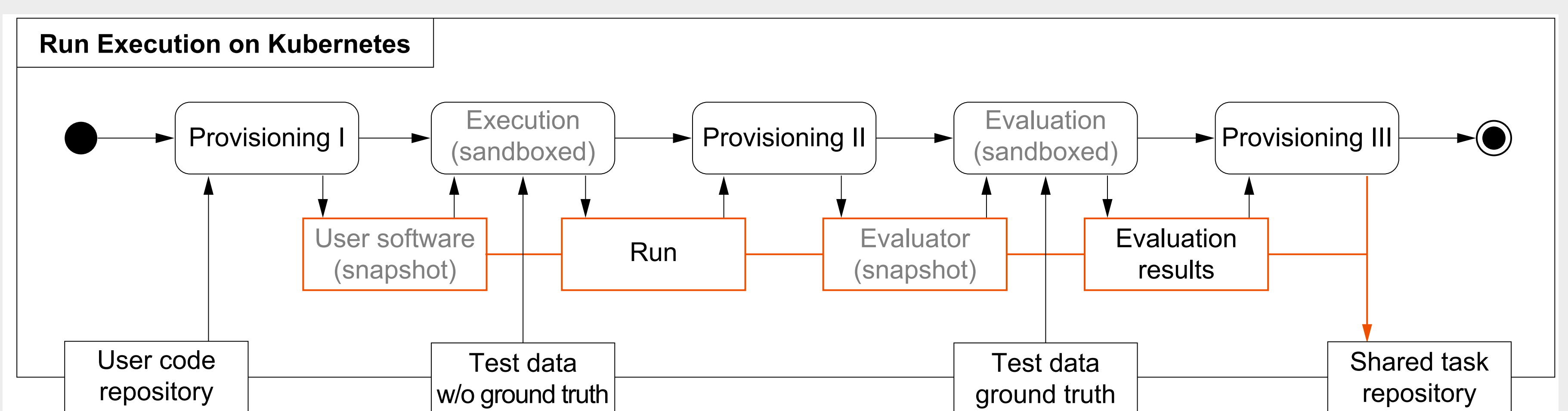
- Public validation data + ground truth
- Private test inputs

Evaluator:

- Docker image that evaluates runs
- Input: Run + ground truth
- Output: Evaluation scores
- We have a collection of standard evaluators

Baseline:

- Docker image with a baseline
- Might serve as starting point for participants
- We have a collection of standard baselines



### Workflow

- Provisioning I (trusted): Branch and clone repository + copy test data.
- Provisioning II (trusted): Persist run files and logs + copy the test ground truth.
- Provisioning III (trusted): Persist evaluation results and logs + merge branch.

### TIRA at SemEval 2023

Task	Reg.	Active	Software	Largest Image	D. in Top-10
ValueEval	91	41	7	66 GB	10%
Clickbait	83	31	21	47 GB	90%

### Want to organize a shared task in TIRA?

We would be very happy to help you!  
 You only need 15 minutes to import your task!

## Post-Hoc Reproducibility Experiments

### Git repository of the shared task can be published after the task

- Repository is fully self-contained (metadata, runs, logs, etc.)
- No Lock-in effect (`tira-run` is only syntactic sugar around Docker)
- Repeat, replicate, and reproduce in one line of code

```
import tira
df = tira.load_data('<dataset-name>')
predictions, evaluation = tira.run(
    '<task-name>/<user-name>/<software-name>',
    data=df, evaluate='<evaluator-name>'
)
```

### Future Work Enabled By Docker Submissions

- Docker images resulting from shared tasks enable creative reuse/hacking
- Creative reuse of SOTA submission: `values.args.me`
- Inject code, models, oracle functions, ...

Try it out :)

Set up your shared task in TIRA in 15 minutes:

