

Adopting Neptune.AI at ALCF

Riccardo Balin
Postdoctoral Appointee
Argonne National Laboratory, LCF

May 19th, 2022

Outline

- Introduction
- Account options and payment
- Signing up
- Installation of client library
- Example of Logging a Training Run
- Comparison to Weights & Biases

Introduction to Neptune.AI

- Neptune is a metadata store for MLOps
- Provides users and teams the ability to log, store, display, organize, compare and query all model-building metadata in a single place
- Website: <https://neptune.ai/>
- Documentation: <https://docs.neptune.ai/>
- GitHub page: <https://github.com/neptune-ai>
- Useful examples to get started: <https://github.com/neptune-ai/examples>







Introduction to Neptune.AI

General Features

- Log and display ML metadata and training data
 - Code, config files and hyperparameters
 - Performance metrics and loss/accuracy curves
 - Hardware consumption and console log (stdout and stderr)
 - Model weights and model checkpoints (e.g., saved as a .pt file)
 - Any chart or visualization (e.g., plots produced with matplotlib)
 - Training data as an artifact (file path and hash)
- Compare experiments and models
 - Web UI allows autodiff of all metadata between runs
- Monitor ML runs live from web UI
- Query metadata from UI and programmatically
- Integrates with most ML frameworks for automatic logging
- Works as SaaS or with on-premise deployment (keep metadata local)



Account Options for Teams Using Neptune.AI

 Individual	 Team	 Organization	 Custom
\$0/month <small>+ usage above free quota</small>	\$150/month <small>+ usage above quota</small>	\$600/month <small>+ usage above quota</small>	<small>starts from</small> \$1200/month
Sign up	Try Team Free for 2 weeks	Contact Us	Contact Us
<ul style="list-style-type: none">◦ 1 member◦ 200 monitoring hours/month*	<ul style="list-style-type: none">◦ Unlimited members◦ 1500 monitoring hours/month*◦ Email and chat support (best effort)	<ul style="list-style-type: none">◦ Unlimited members◦ 6000 monitoring hours/month*◦ User access management◦ Email and chat support (1 work day)	<ul style="list-style-type: none">◦ Unlimited members◦ 12000 monitoring hours/month◦ Volume discounts including unlimited usage◦ Single sign-on◦ Customizable contract and SLA◦ Dedicated support and onboarding

Account Options for Teams Using Neptune.AI

Notes




- Can add storage and monitoring hours for a fee (\$10 per 100 hours)
- Unused monitoring hours roll over month-to-month, but get reset every year (for individual account at least)
- Team plans offered for free (with larger base quotas) to academia, research and education groups, and non-profit organizations
- Accounts are **charged based on usage, NOT number of users**
 - Charged based on monitoring hours consumed by all users as a whole
 - Add infinite users at very small additional cost if usage is negligible
 - Could be attractive option for ALCF, average cost of adding users to account is likely small

Signing Up to Neptune.AI

- Click Sign up button on home page
- Select how to sign up (I chose GitHub)
- Select what type of account (individual or team)
- Answer a few questions and you're done!

Create a free account

Monitor and keep track of ML experiments

 Google  GitHub 

OR USE EMAIL

Email address

Type your email address, e.g. user@example.com

Password

Use 8 or more characters with a mix of letters, numbers and symbols.

Create your secure password

☐ I agree to [Terms of Service](#) and [Privacy Policy](#)

Next

neptune.ai

Metadata store Product Customers Pricing Resources Sign up Login

Experiment tracking and model registry for production teams

Log, store, query, display, organize, and compare all your model metadata in a single place



Sign up See docs

Signing Up to Neptune.AI

- Logging in gets to to your home space in the user interface

The screenshot displays the Neptune.AI user interface for a user named 'rickybalin' (INDIVIDUAL). The top navigation bar includes the Neptune logo, the user name, and links for 'Help center', a notification bell, and a profile icon. Below the navigation bar, the 'Projects' tab is selected, showing a search bar, a 'Sort by' dropdown set to 'Recently visited', and a list of projects. A dashed blue box highlights a '+ New project' button. Two projects are listed: 'testALCF' (Public) and 'example-project-tensorflow-keras' (Private). Each project card shows the user name, project name, description, and statistics (4 experiments, 0 datasets, 1 person for 'testALCF'; 10 experiments, 0 datasets, 1 person for 'example-project-tensorflow-keras').





rickybalin **INDIVIDUAL**





Sharing work results with others? [Try Team for 14 days for free!](#) [Help center](#)  

Projects ² People Subscription Settings

Search Sort by **Recently visited** ▼

+ New project

rickybalin testALCF Public
Test and learn how to use Neptune for ALCF systems
4  0  1  

rickybalin example-project-tensorflow-keras Private
Example project using TensorFlow / Keras.
10  0  1  

Installation of Client Library

- Using Pip on ThetaGPU
 - Install Neptune client simply with *pip install neptune-client*
 - Can load my env with *conda activate /grand/datascience/balin/NeptuneAI/testPipInstall/env/nept*
- Neptune also allows an on-premise deployment
 - Metadata is stored on local machines or private cloud
 - Can be deployed as Kubernetes Cluster or VM running Kubernetes under the hood
 - There is a free 30 day on-premise trial
 - More information and instructions at <https://docs.neptune.ai/administration/on-premises-deployment>

Example Training Run

- Source code for example found at `/grand/datascience/balin/NeptuneAI/testPipInstall/src`
- Simple serial training of 1 layer ANN on random data
- Modifying code to log metadata with Neptune API is simple
 - Import Neptune module

```
# Neptune
import neptune.new as neptune
```

- Initialize Neptune logging, and include source code files

```
# Initialize Neptune logging
run = neptune.init(
    project="rickybalin/testALCF",
    api_token="eyJhcGlYWRkcmVzcyI6Imh0dHBzOi8vYXBwLm5lcHRlbnUuYWkiLCJhcGlfdXJsIjoiaHR0cGVuZS5haSI6ImFwaV9rZXkiOiJhMjllYmRkMSIlMzI2LTQ0NzctOWE2MS05M2MwNmZlZWZhYzkiOiJkaWZlcnQiOjE2OTQ0NDg0In0=",
    source_files=["*.py"],
) # my credentials taken from the project I created on my Neptune account
```

Example Training Run

- Modifying code to log metadata with Neptune API is simple
 - Log some training hyperparameters

```
# Log training parameters
train_params = {'n_epochs': args.Nepochs, 'mini_batch': args.batch,
                'learning_rate': args.learning_rate, 'n_samples': args.nSamples}
model_params = {'n_neurons': args.nNeurons, 'n_inputs': args.nInputs, 'n_outputs': args.nOutputs}
system_params = {'device': args.device}
run['train_params'] = train_params
run['model_params'] = model_params
run['system_params'] = system_params
```

- Pass Neptune run object to training function

```
model, timeStats = trainNN(inputs, outputs, args, logger_conv, run)
```

- Log loss and accuracy at each epoch

```
# Log loss and accuracy to Neptune
run["training/loss"].log(loss)
run["training/acc"].log(acc)
```

Example Training Run

- Modifying code to log metadata with Neptune API is simple
 - Upload model checkpoint

```
# Upload model to Neptune  
run["model"].upload('./NNmodel.pt')
```

- Stop Neptune logging at end of program

```
# Stop Neptune logging  
run.stop()
```

- Log training data as an artefact (I did not do this since I generated training data within program)

```
# Log training data on Neptune  
# If loaded data from file: run["dataset/train_data"].upload("./data/train_data.csv")  
# Can also ave dataset versions as Neptune artifacts with the track_files() method  
run["dataset/train_data"].track_files('data/train_data.csv')
```

Example Training Run

- Visualize and compare experiments on web UI

The screenshot displays a web application interface for managing and visualizing training runs. The top navigation bar includes the user profile 'rickybalin' (INDIVIDUAL), a sharing notice, a help center link, and a share button. The main navigation menu shows 'Runs' (4), 'Models', 'Project metadata', 'Notebooks' (0), 'Settings', and 'Trash' (2). The 'Runs' section is active, showing a list of runs under the 'testALCF' project.

On the left, there is a sidebar with a 'Runs table' section and a 'Horizontal split' section. The 'Runs table' section contains a search bar, a 'Save' button, a 'Save as new' button, and a 'Show suggested columns' toggle. The 'Horizontal split' section contains a 'Compare runs' button.

The main content area displays a table of runs. The table has a 'Pinned columns' section and a 'Suggested columns' section. The 'Pinned columns' section includes 'Id', 'Creation Time', 'Owner', 'Monitoring Time', and 'Tags'. The 'Suggested columns' section includes '..._inputs', '...neurons', and '...outputs'. The table shows four runs, all owned by 'rickybalin'.

PINNED COLUMNS					SUGGESTED COLUMNS		
Id	Creation Time	Owner	Monitoring Time	Tags	..._inputs	...neurons	...outputs
TES-6	2022/05/10 11:08:16	rickybalin	<1m		6	20	6
TES-5	2022/05/10 11:05:29	rickybalin	<1m		6	20	6
TES-4	2022/05/10 11:03:17	rickybalin	<1m		6	20	6
TES-2	2022/05/10 10:57:38	rickybalin	<1m		6	20	6

Example Training Run

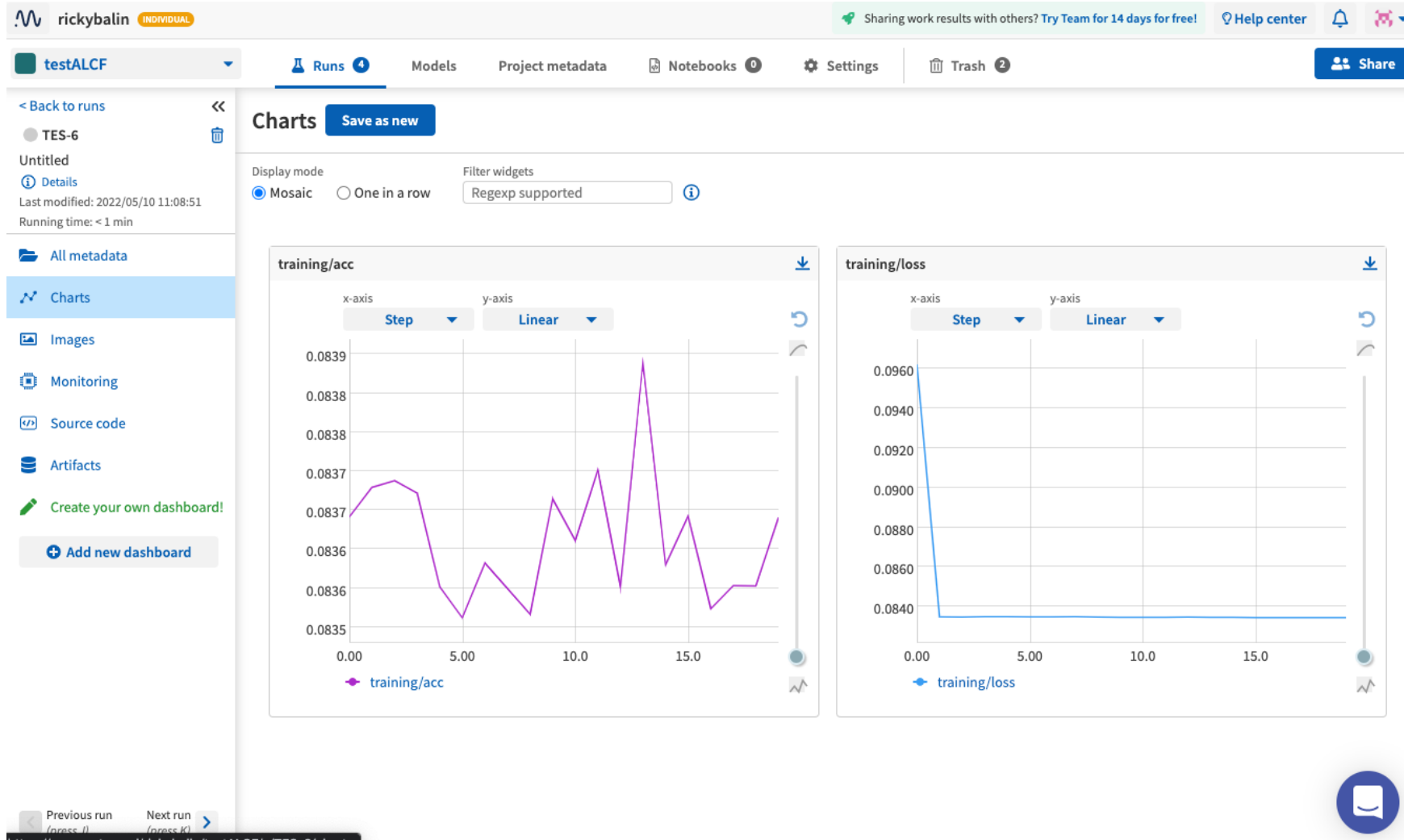
- Click on run name to visualize metadata logged

The screenshot shows the DVC web interface for a user named 'rickybalin'. The top navigation bar includes links for 'Runs' (4), 'Models', 'Project metadata', 'Notebooks' (0), 'Settings', and 'Trash' (2). The left sidebar shows the 'testALCF' project and various metadata views. The main content area displays the 'TES-6' run details, including its last modified time and running time. A table lists the metadata entries, with 'train_params' selected. The right panel is currently empty, prompting the user to 'Select a field to preview'.

Name	Preview
model	1.89KB, pt
model_params	
monitoring	
source_code	
sys	
system_params	
train_params	
training	

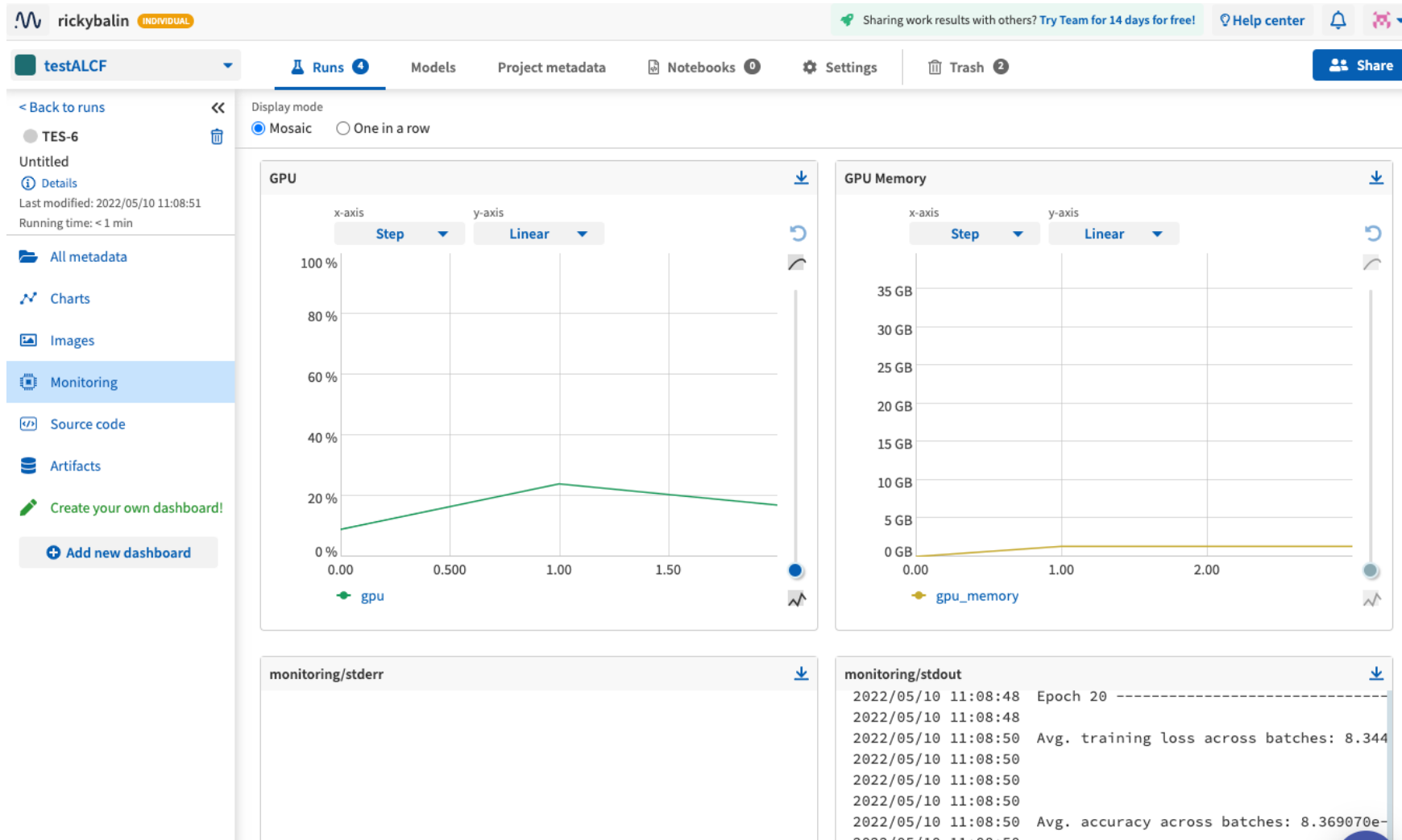
Example Training Run

- Click on run name to visualize metadata logged



Example Training Run

- Click on run name to visualize metadata logged



Example Training Run

- Select what runs to compare clicking on the eye next to the run name

The screenshot displays the ALCF training run comparison interface. The top navigation bar includes the user name 'rickybalin' (INDIVIDUAL), a sharing prompt 'Sharing work results with others? Try Team for 14 days for free!', and links to 'Help center', a notification bell, and a share icon. The main navigation bar shows 'testALCF' as the selected project, with tabs for 'Runs' (4), 'Models', 'Project metadata', 'Notebooks' (0), 'Settings', and 'Trash' (2). A 'Share' button is also present.

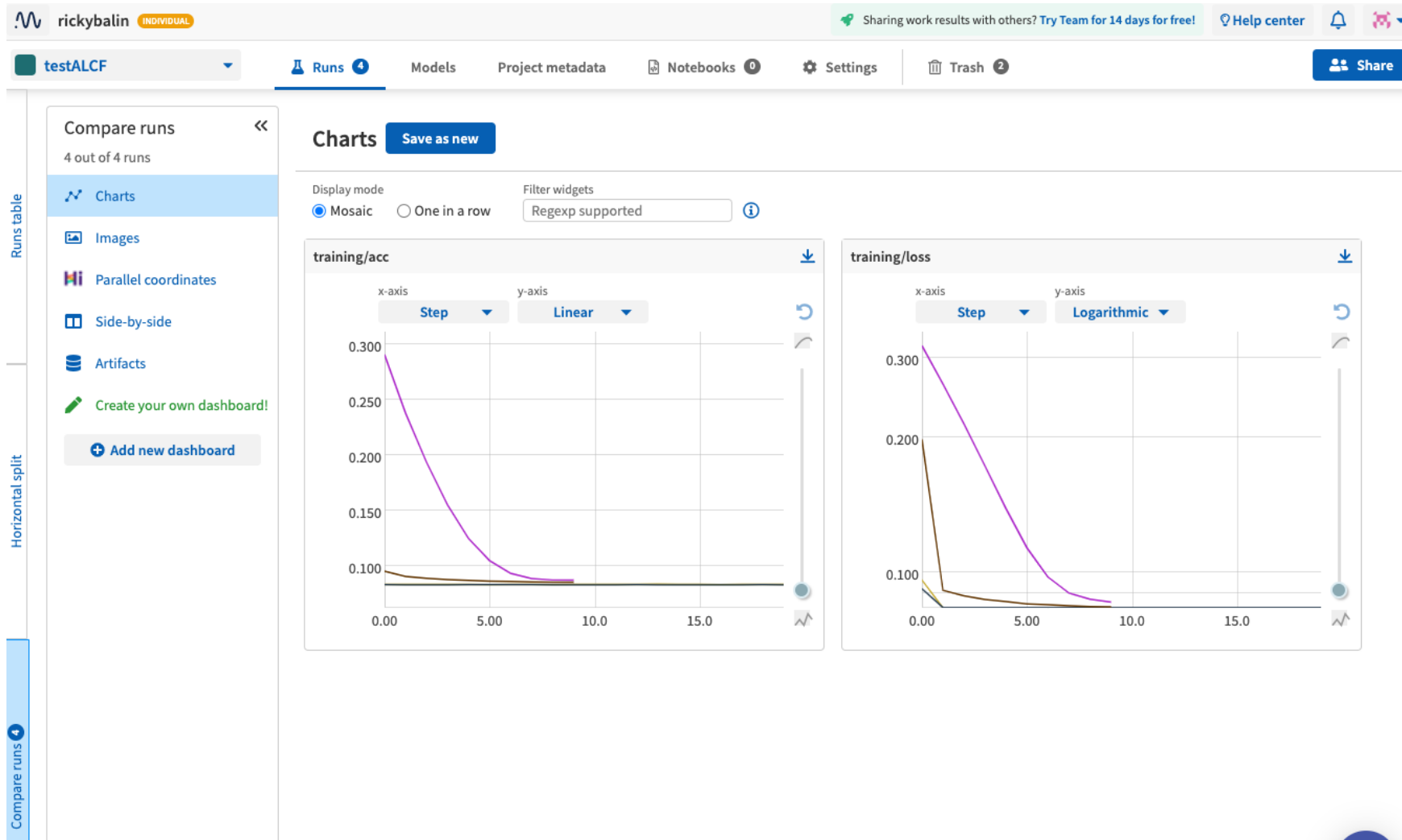
The 'Compare runs' sidebar on the left shows '4 out of 4 runs' selected. It includes options for 'Charts', 'Images', 'Parallel coordinates', 'Side-by-side' (selected), and 'Artifacts'. There is also a 'Create your own dashboard!' link and an 'Add new dashboard' button.

The main table displays metrics for four runs: TES-6, TES-5, TES-4, and TES-2. The 'train_params/n_samples' row is highlighted, showing values of 100000, 100000, 10000, and 1000 respectively. The 'train_params/n_samples' row is also highlighted in the table.

		TES-6	TES-5	TES-4	TES-2
monitoring/memory	variance	0.780228	0.000437601	0	0
Ping Time		2022/05/10 11:08:51	2022/05/10 11:06:29	2022/05/10 11:03:22	2022/05/10 10:57:40
Running Time		35.245	59.365	4.515	1.923
Size		26574	26609	21608	21607
system_params/device		cuda	cpu	cpu	cpu
train_params/n_epochs		20	20	10	10
train_params/n_samples		100000	100000	10000	1000
training/acc	average	0.0836626	0.0833182	0.0880695	0.146586
training/acc	last	0.0836907	0.0833009	0.0855745	0.0875393
training/acc	max	0.0838869	0.0834384	0.0955805	0.290921
training/acc	min	0.0835621	0.083221	0.0855745	0.0875393
training/acc	variance	6.23499e-9	3.25437e-9	0.00000878516	0.00468547
training/loss	average	0.0840961	0.0839683	0.0973014	0.158063

Example Training Run

- Select what runs to compare clicking on the eye next to the run name



Comparison with W&B

Notable similarities

- Allow on-premise deployment
- Minimal changes to code for logging
- Distributed training support
- Web UI to navigate through experiments and metadata

Notable differences

- Neptune has limited TensorBoard support
- Neptune does not have interactive project level reports
- Neptune has more features to organize and search experiment metadata

Backup Slides

Charging Monitoring Hours

- Monitoring hours are the time spent actually logging metadata via the Neptune API
- Browsing, exploring, querying the metadata with the web UI or programmatically does not accumulate monitoring hours
- If logging instances are separated > 10 minutes in clock time, instances logged individually with 1 sec cost
- If logging instances are separated < 10 minutes in clock time, logging time is the entire between two instances
- Within first and last logging instance separated < 10 minutes, can log infinitely many times at no additional cost

