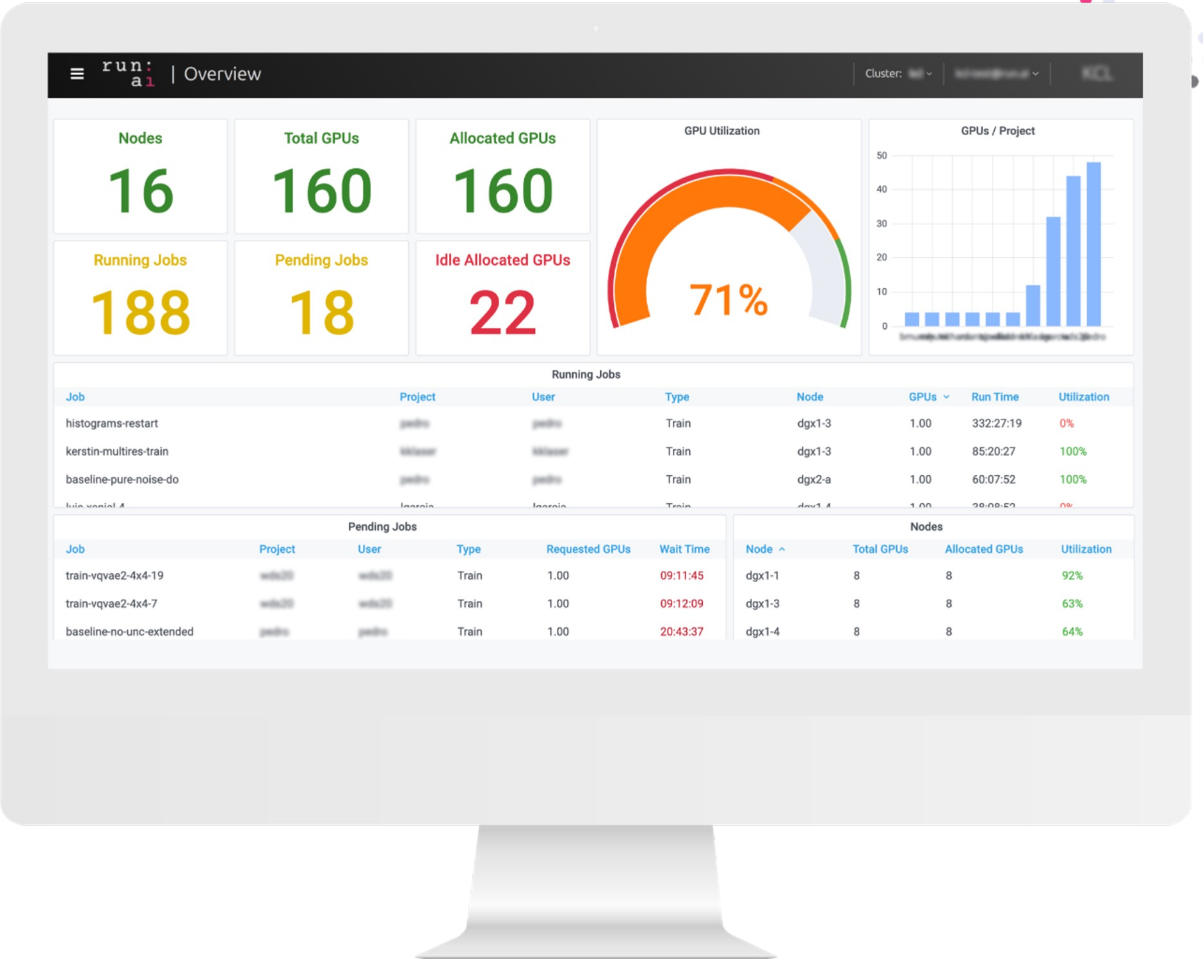
run:ai

Administrator Training

Gain visibility and control over AI workloads to increase GPU utilization

Run:AI brings HPC capabilities to Kubernetes with batch scheduling and GPU virtualization, enabling seamless distributed training and full utilization of GPU resources.



Cluster: [redacted] [redacted] [redacted]

Nodes	Total GPUs	Allocated GPUs
16	160	160

Running Jobs	Pending Jobs	Idle Allocated GPUs
188	18	22

Running Jobs

Job	Project	User	Type	Node	GPUs	Run Time	Utilization
histograms-restart	[redacted]	[redacted]	Train	dgx1-3	1.00	332:27:19	0%
kerstin-multires-train	[redacted]	[redacted]	Train	dgx1-3	1.00	85:20:27	100%
baseline-pure-noise-do	[redacted]	[redacted]	Train	dgx2-a	1.00	60:07:52	100%
luis-xenial-4	lgarcia	lgarcia	Train	dgx1-4	1.00	38:08:52	0%

Pending Jobs

Job	Project	User	Type	Requested GPUs	Wait Time
train-vqvae2-4x4-19	[redacted]	[redacted]	Train	1.00	09:11:45
train-vqvae2-4x4-7	[redacted]	[redacted]	Train	1.00	09:12:09
baseline-no-unc-extended	[redacted]	[redacted]	Train	1.00	20:43:37

Nodes

Node	Total GPUs	Allocated GPUs	Utilization
dgx1-1	8	8	92%
dgx1-3	8	8	63%
dgx1-4	8	8	64%

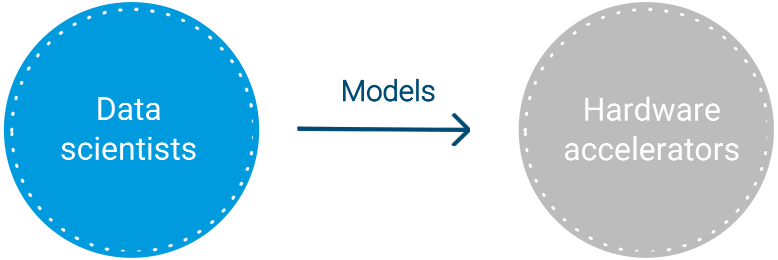
Concepts

Basics & Assumptions
Locality → Global considerations

- At the heart of Run:AI is the premise that “optimization” requires finding the “right resources for your Job”. With this assumption in mind, the researcher is no longer **permanently** assigned a **local** machine.

- Instead, upon request, Run:AI will allocate resources on different machines according to your needs taking into account global considerations

The Run:AI Vision
Full Hardware Abstraction



Basics & Assumptions
Containers & Images

- To be able to abstract the resource location, Run:AI uses docker images to instantiate containers on the right machine

- It is assumed that you are already familiar with docker images and are using them today.

Basics & Assumptions
Shared Storage

- As a researcher, you use data: training data, scripts, interim checkpoints, docker image, etc.
- To be able to abstract the resource location, your data must be stored in a location which is *shared by all machines in a uniform way*. You can no longer rely on data being stored on the local machine
- If not already there, as part of the Run:AI implementation, your IT department will make such a shared location available

Basic Run:AI Concepts
Projects

- As a researcher, each request for cluster resources should be accompanied by a reference project. Without a project, resources cannot be allocated

- Depending on your organization's preference, projects can be modeled as **individuals**, as **teams** of people (e.g. team-ny) or as actual **business** activities (e.g. ct-scan-2020)

Basic Run:AI Concepts
Guaranteed Quotas

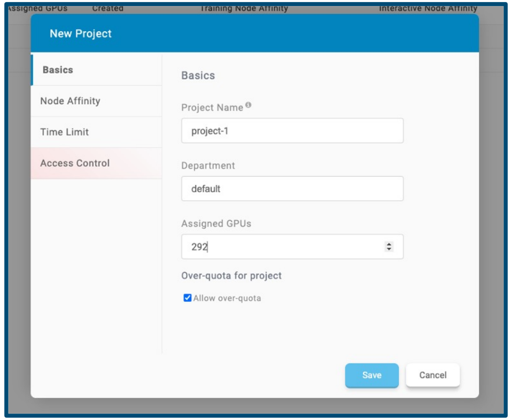
- Projects are assigned with a *guaranteed quota* of GPUs
- Projects can go over quota and consume more GPUs than assigned to them
- The Run:AI scheduler preempts and queues over-quota workloads when there are not enough resources to run under-quota workloads, taking into account fairness and priorities
- For more information on the Run:AI scheduler, including over-quota fairness, preemption, priorities, bin packing, elasticity and more, see here.

Projects

- Project are the most granular level to setup a GPU Quota for a Researcher (or group of..)
- Each Job is always associated with a Project.

Project

Assigned GPUs
Allow Over Quota
Node Affinity
Time Limit for Interactive Jobs



Basics

Project Name: project-1

Department: default

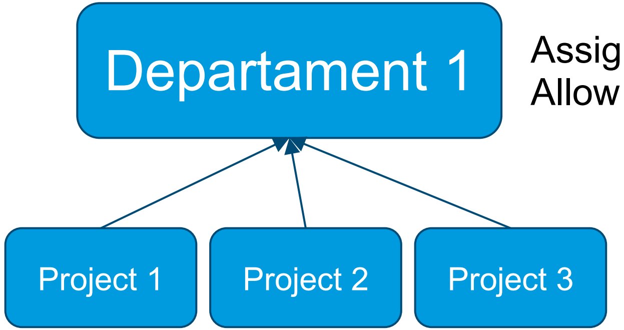
Assigned GPUs: 292

Over-quota for project

☑ Allow over-quota

Departaments

- Departments create a second hierarchy of resource allocation:
  - A Project is associated with a single Department. Multiple Projects can be associated with the same Department.
  - A Department, like a Project is associated with a Quota.
  - A Department quota supersedes a Project quota.



Assigned GPUs
Allow Over Quota

Assigned GPUs
Allow Over Quota
Node Affinity
Time Limit for Interactive Jobs

Basic Run:AI concepts:

Run:AI can schedule interactive “build” workloads, unattended “train” workloads and “inference” workloads

Build

- **Development & debugging**
- **Interactive** sessions
- **Short** cycles
- Performance is **less** important
- **Low** GPU utilization

Training

- **Model Training**
- **Remote, unattended** execution
- **Long** workloads
- **Throughput** is highly important
- **High** GPU utilization

Inference

- **Run Model in Production**
- **Services multiple users**
- **Typically low GPU memory requirements**
- **Performance is key**

Basic Run:AI Concepts
Build (Interactive)

- Build workloads are meant for interactive work.
- Build workloads cannot extend beyond *guaranteed* quota (as they cannot be stopped automatically).
- It is the responsibility of the researcher to stop a build workload.
- The Run:AI scheduler will usually not preempt a build workload with two notable exceptions:
  - The administrator has set a duration limit on interactive jobs for your project
  - The researcher has used the flag –preemptible

- See Quickstart on how to run a build job.

Basic Run:AI Concepts
Training

- Run:AI allows non-interactive training workloads to extend beyond *guaranteed* quotas and into *over-quota* as long as computing resources are available.
- To achieve this flexibility, the system needs to be able to safely stop a training workload and restart it again later. This means that:
  - The docker image should have an entrypoint instruction that initiates the training automatically upon restart.
  - Highly recommended: save ‘checkpoints’ frequently and allow the training to restart from the latest checkpoint
- See Quickstart on how to run a training job

Basic Run:AI Concepts
Inference

- Run:AI allows submitting inference workloads.
- Inference workloads are considered production workloads and thus take precedence over training workloads.
- It is the responsibility of the researcher to stop an inference workload.
- The Run:AI scheduler will usually not preempt a inference workloads.
- Inference workloads cannot extend beyond *guaranteed* quota (as they cannot be stopped automatically).

- See Quickstart on how to run a inference workload

Basic Run:AI Concepts
Scheduler Fairness

The Run:AI scheduler determines fairness between multiple over-quota Projects according to their GPU quota.

The fairness works according to the relative portion of the GPU quota for each Project. To further illustrate that, suppose that:

- Project A has been allocated with a quota of 3 GPUs.
- Project B has been allocated with a quota of 1 GPU.

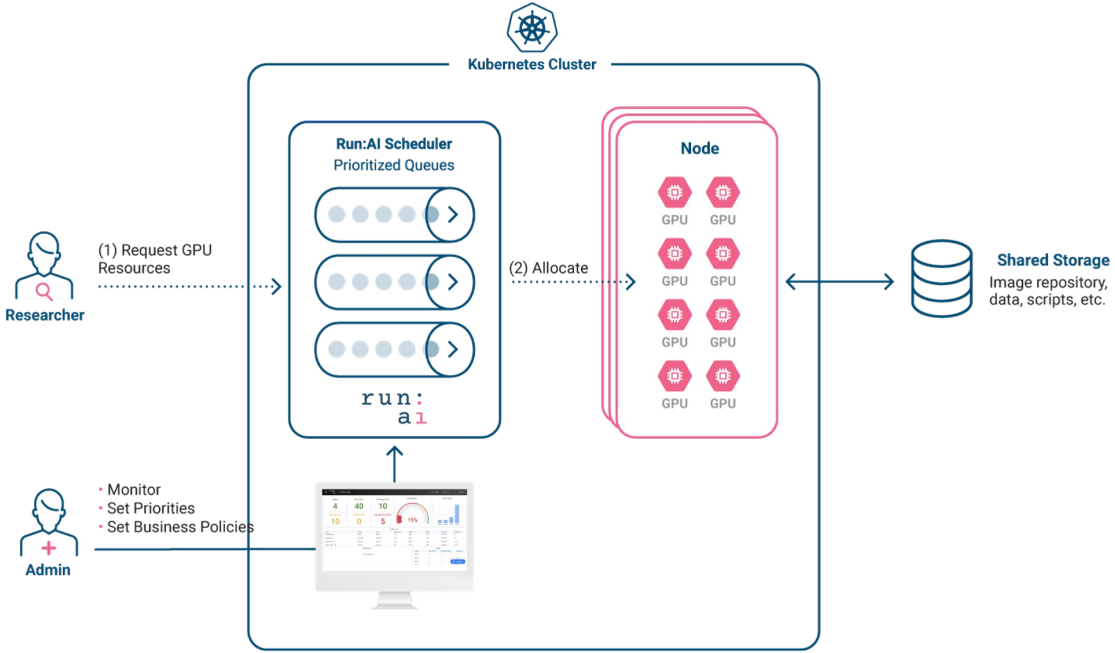
Then, if both Projects go over quota, Project A will receive 75% (=3/(1+3)) of the idle GPUs and Project B will receive 25% (=1/(1+3)) of the idle GPUs.

This ratio will be recalculated every time a new Job is submitted to the system or an existing Job ends.

Components

Architecture

- Installed over a Kubernetes Cluster
- Researchers request and manage resources via CLI (or others)
- Administrators monitor and set priorities via UI
- Multi-Cluster Architecture

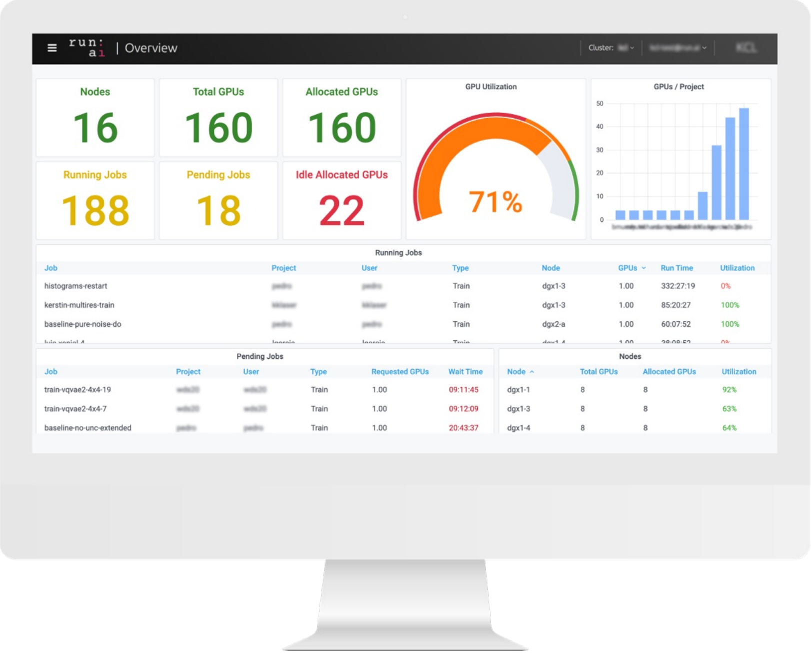


The Run:AI Components
The Run:AI Administrative User Interface

Designed to be used by IT and occasionally by researchers.
Goals are:

- Show holistic view of system resources (nodes, jobs etc). Both current status and long term status
- Allocate resources to researchers via projects
- Resolve conflicts

See: app.run.ai



Nodes 16
Total GPUs 160
Allocated GPUs 160
Running Jobs 188
Pending Jobs 18
Idle Allocated GPUs 22

Running Jobs

Job	Project	User	Type	Node	GPUs	Run Time	Utilization
histograms-restart	[redacted]	[redacted]	Train	dgx1-3	1.00	332:27:19	0%
kerstin-multires-train	[redacted]	[redacted]	Train	dgx1-3	1.00	85:20:27	100%
baseline-pure-noise-do	[redacted]	[redacted]	Train	dgx2-a	1.00	60:07:52	100%
luis-xenial-4	lgarcia	lgarcia	Train	dgx1-4	1.00	38:08:52	0%

Pending Jobs

Job	Project	User	Type	Requested GPUs	Wait Time
train-vqvae2-4x4-19	[redacted]	[redacted]	Train	1.00	09:11:45
train-vqvae2-4x4-7	[redacted]	[redacted]	Train	1.00	09:12:09
baseline-no-unc-extended	[redacted]	[redacted]	Train	1.00	20:43:37

Nodes

Node	Total GPUs	Allocated GPUs	Utilization
dgx1-1	8	8	92%
dgx1-3	8	8	63%
dgx1-4	8	8	64%

The Run:AI Components
The Command Line interface

Designed to be used by the researcher in order to do things like:

- Run and delete workloads
- View list of workloads and their status
- View list of available and allocated GPUs
- Access workloads via bash and view online logs
- Similar to Docker API

Docker Syntax

```
nvidia-docker run --shm-size 16G -it -
-rm -e HOSTNAME=`hostname` -v
/raid/public/my_datasets:/root/dataset
nvcr.io/nvidia/pytorch
```

Run:AI Syntax

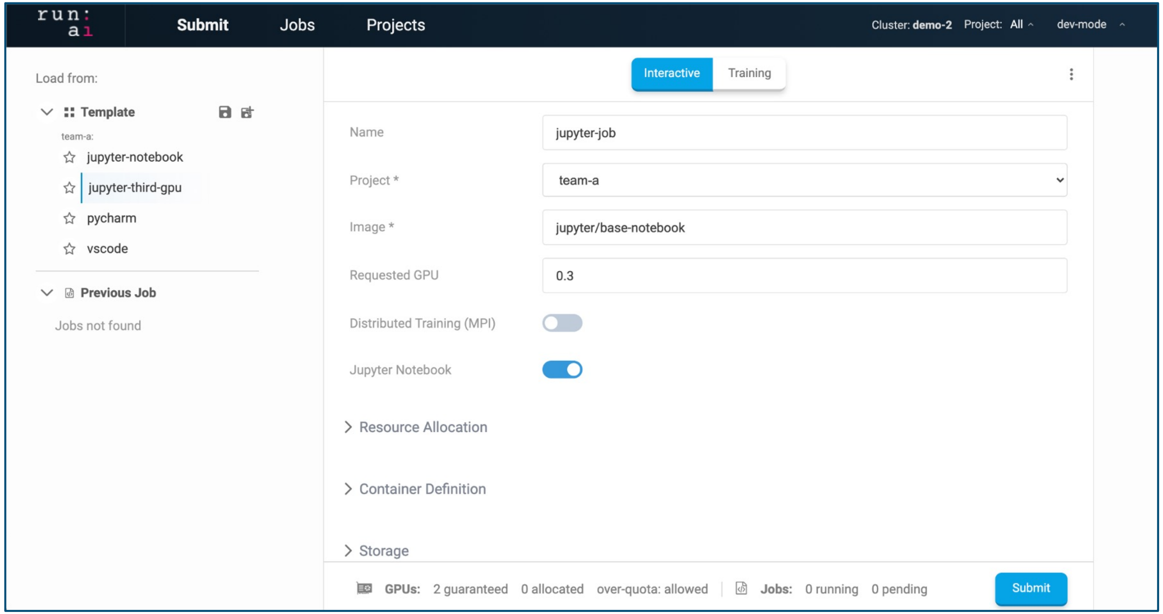
```
runai submit myjob --large-shdm -e
HOSTNAME=`hostname` -v
/raid/public/my_datasets:/root/dataset -i
nvcr.io/nvidia/pytorch
```

See Run:AI CLI reference

The Run:AI Components
The Run:AI Researcher User Interface

Designed to be used by Researchers and/or Data Scientists.

- Submit Jobs
- Use pre-configured templates



Name: jupyter-job

Project *: team-a

Image *: jupyter/base-notebook

Requested GPU: 0.3

Distributed Training (MPI): off

Jupyter Notebook: on

GPUs: 2 guaranteed 0 allocated over-quota: allowed | Jobs: 0 running 0 pending

Admin UI - Overview

Login here: app.run.ai

Main components

- Dashboards - monitor the cluster
- Jobs - monitor specific jobs
- Nodes - monitor nodes
- Projects - manage projects
- Clusters - install clusters
- Users - manage users

Administrative Responsibilities

Installation

The Kubernetes Configuration File

- Kubernetes is accessible via a “config” file.
- The default file location is ~/.kube/config (can be overridden via env-var)
- The initial file provided by Kubernetes installation provides full admin rights. You can and should create a variant of the file with selective access (see task: *Configure Role-based authorization* below)
- Each Researcher should have his/her own copy of the file.

Installing Run:AI
Setup Researcher Command Line Interface (CLI)

For ease of upgrade, we strongly recommend to install the CLI on a dedicated jumpbox

Prerequisites:

- **kubectl** & **helm** binaries
- Kubernetes configuration file (specific, not admin profile)
- **runai** CLI installation (mac or linux)

For further details see: link

Limit Jobs to run on Specific Node Groups

A frequent use case is to assign specific Projects to run only on specific nodes (machines). This can happen for various reasons. Examples:

- The project team needs specialized hardware (e.g. with enough memory).
- The project team is the owner of specific hardware which was acquired with a specialized budget.
- We want to direct build/interactive workloads to work on weaker hardware and direct longer training/unattended workloads to faster nodes.

(see https://docs.run.ai/Researcher/scheduling/the-runai-scheduler/#node-affinity)

Grouping Nodes

To set node affinities, you must first annotate nodes with labels. These labels will later be associated with Projects. Each node can only be annotated with a single name.

To get the list of nodes, run: `kubectl get nodes`

To annotate a specific node with the label "dgx-2", run:

```
kubectl label node <node-name> run.ai/type=dgx-2
```

You can annotate multiple nodes with the same label

Run:AI APIs

Run:AI provides a number of possible APIs to access the system programmatically:

API	Description
Researcher REST API	Provides APIs to submit and manage Jobs, list Projects etc
Administrator REST API	Provides APIs to manage Projects, Departments, Users, Clusters etc.
Kubernetes API	In addition to the above two, you can access the Kubernetes cluster directly by sending YAMLs or Kubernetes APIs.
Metrics API	API for retrieving usage metrics from Run:AI. Used for creating custom dashboards and alerts

Administrative Responsibilities

Authentication & Authorization

Installing Run:AI

Cluster Install - Post Install Tasks

Task	Description
Authentication & Authorization **(*)**	Understand how authentication and authorization work. Specifically configure Researcher Authentication. Optionally, configure Single Sign-on.
Set Node roles **(*)**	Designate specific nodes for Run:AI, for Run:AI system, and for GPU/CPU worker nodes
Allow network access to containers	Add a load balancer to allow researchers to connect to containers
User identity in container	Understand how to control the identity of the user within the container so as to provide the right access to container resources (such as file systems)
Secure docker registry	Setup connectivity to a secured docker registry
Upgrade Cluster	Upgrade the Run:AI software on a cluster.

(*) see separate slide

Installing Run:AI
Users

- By default, Run:AI comes with an internal user system, based on keycloak.
- **Single Sign-on:** It is possible to connect Run:AI with the organization’s Identity Provider (IdP). Run:AI supports any IdP using the SAML protocol. Examples: Auth0, Google, Salesforce.

When using Single Sign-on, you can:

- Map groups from the organization’s directory to Run:AI roles
- Map UID/GID defined in the directory straight into user containers

For more information see: http://docs.run.ai/admin/runai-setup/advanced/sso/

Installing Run:AI

Set Node Roles

Setting roles to specific nodes is needed when you want to:

- Dedicate one or more nodes to Run:AI software.
- Machine learning frequently requires jobs that require CPU but not GPU. You may want to direct the scheduling of these jobs to dedicated nodes that do not have GPUs, so as not to overload these machines.
- Your Kubernetes cluster contains nodes that should not be used by Run:AI.

Use the command `runai-adm set node-role`

(see https://docs.run.ai/admin/runai-setup/advanced/node-roles/?h=node+roles)

Installing Run:AI

Authentication

There are two “domains” that need to be controlled:

- The Run:AI User interface
- The Run:AI GPU Cluster

The first typically (though not always) resides on the cloud.

The second is actually about protecting access to a Kubernetes cluster. This *Researcher authentication* must be configured at the cluster level. See here

Installing Run:AI
Role-based Authorization

The Run:AI User interface allows the definition of *Roles* such as “administrator” and “researcher”.

A researcher can be mapped into specific Run:AI projects (Kubernetes namespaces). This roles are propagated into the Kubernetes cluster and enforced such that a researcher can only see/act on assigned projects

Administrative Responsibilities

Day to Day Maintenance

Day to day Maintenance
Monitoring

Task	Description
Monitor cluster operation	Find Resource usage abnormalities and take actions

Day to Day Maintenance
Business Continuity

Task	Description
Plan for disaster recovery	Learn what to back-up and how to restore data
Node downtime	Safely remove a node from the cluster and/or handle node unplanned downtime.
Monitor cluster health	Connect Run:AI to an alert management system for monitoring cluster health

Day to Day Maintenance
Configuration Tasks

Task	Description
Manage Users **(*)**	Add, Update, Delete Administrators and Researchers.
Manage Projects (and Departments) **(*)**	Create Projects and set Project quotas.
Create and maintain Docker images	Assist Researchers with the creation of Docker images
Upgrade Cluster	Upgrade the Run:AI software on a cluster.

Day to day Maintenance
Advanced

Task	Description
Working with secrets	Propagate sensitive information into Researchers' code
Working with templates **(*)**	Create and maintain templates which simplify CLI and Researcher UI usage
Create Node Groups	Group nodes into 'types' to be used for constraining projects to specific node types

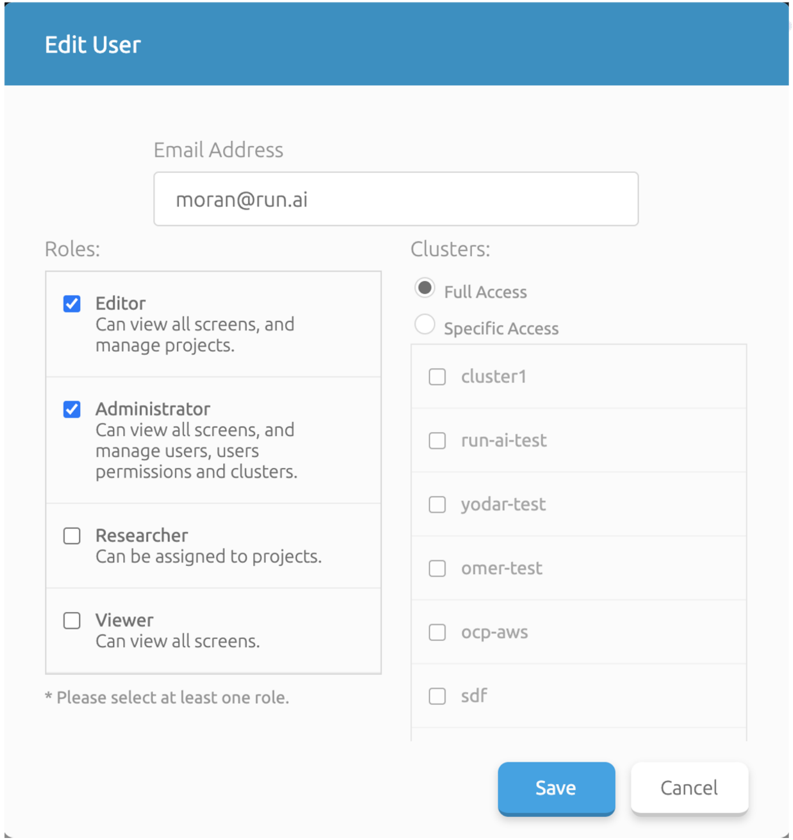
Day to Day Management
Manage Admin UI Users

The Administrative User interface can be configured to add or remove users and to change user roles.

Users can have multiple roles:

- Administrator - can change users and settings
- Editor - can edit objects in Admin UI
- Researcher - can be assigned to Projects

For further details see [link]()



Edit User

Email Address: moran@run.ai

Roles:

☑ Editor
Can view all screens, and manage projects.

☑ Administrator
Can view all screens, and manage users, users permissions and clusters.

☐ Researcher
Can be assigned to projects.

☐ Viewer
Can view all screens.

* Please select at least one role.

Clusters:

● Full Access
○ Specific Access

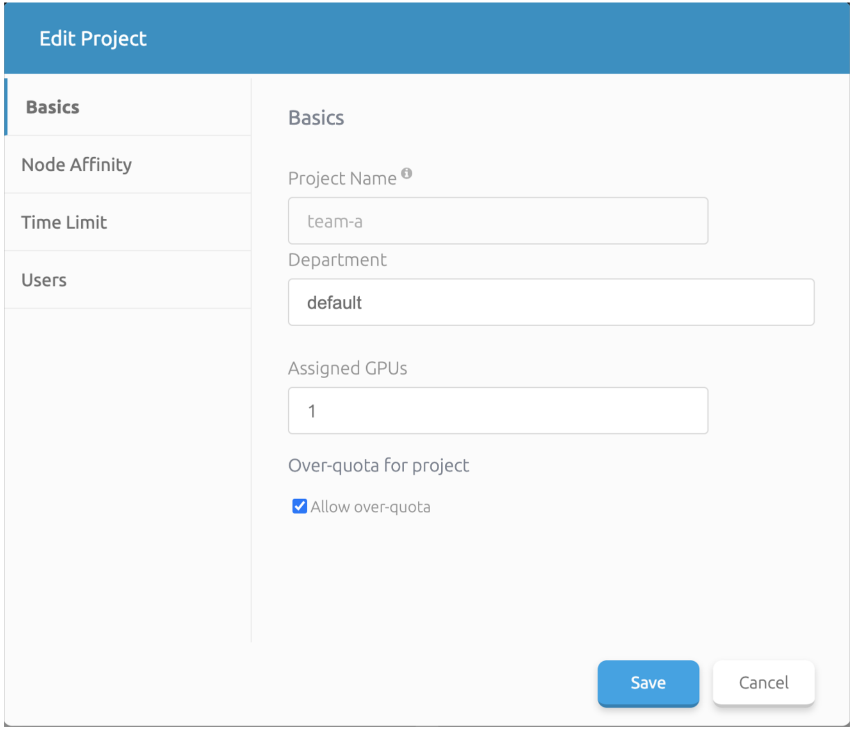
☐ cluster1
☐ run-ai-test
☐ yodar-test
☐ omer-test
☐ ocp-aws
☐ sdf

Day to Day Management
Manage Projects

To setup projects see link. You can set:

- Project name
- GPU Allocation
- Affinity to specific node groups
- Time limitation for interactive jobs

Optional: Departments serve as a second Project hierarchy



Basics

Project Name: team-a

Department: default

Assigned GPUs: 1

Over-quota for project

☑ Allow over-quota

Day to Day Management
Create Templates

The CLI and Researcher UI have a templating mechanism that is useful for shortening the command line. There are two kinds of templates:

- Named templates (e.g. *runai submit…. --template batch_ops*)
- Default Administrative template.

For further details see link

Integration

Run:AI APIs

Run:AI provides a number of possible APIs to access the system programmatically:

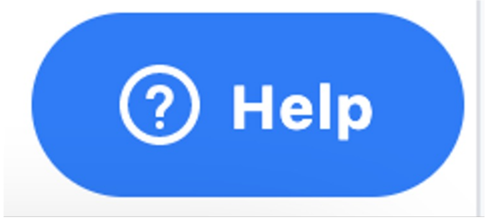
API	Description
Researcher REST API	Provides APIs to submit and manage Jobs, list Projects etc
Administrator REST API	Provides APIs to manage Projects, Departments, Users, Clusters etc.
Kubernetes API	In addition to the above two, you can access the Kubernetes cluster directly by sending YAMLs or Kubernetes APIs.
Metrics API	API for retrieving usage metrics from Run:AI. Used for creating custom dashboards and alerts

How to Get Help

Write to support@run.ai

or

Use the “feedback” button on app.run.ai



run:ai

Thank You

Contact: support@run.ai