# Óptimo uso del clúster y de la GPU con Run AI

NetApp Solutions

NetApp
April 19, 2024

This PDF was generated from https://docs.netapp.com/es-es/netapp-solutions/ai/osrunai_run_ai_installation.html on April 19, 2024. Always check docs.netapp.com for the latest.

# Tabla de contenidos

Óptimo uso de clúster y de la GPU con Run:AI . . . 1
Ejecutar:instalación de AI . . . 1
Ejecutar:Paneles de IA y vistas . . . 1
Creación de proyectos para equipos de ciencia de datos y asignación de GPU . . . 2
Enviar trabajos en ejecución:AI CLI . . . 3
Se obtiene un uso elevado del clúster . . . 6
Asignación de GPU fraccionaria para las cargas de trabajo menos exigentes o interactivas . . . 7
Utilización óptima del clúster gracias a la asignación de GPU por encima de una cuota . . . 8
Equidad en la asignación de recursos básicos . . . 10
Justicia por exceso de cuotas . . . 11
Guardar datos en un volumen persistente aprovisionado por Trident . . . 13

# Óptimo uso de clúster y de la GPU con Run:AI

En las siguientes secciones se ofrecen detalles sobre la instalación de Run:AI, escenarios de prueba y resultados realizados en esta validación.

Validamos el funcionamiento y el rendimiento de este sistema mediante herramientas estándar del sector, incluidas las pruebas de rendimiento TensorFlow. El conjunto de datos ImageNET se utilizó para entrenar ResNet-50, que es un famoso modelo de red neuronal convolucional (CNN) DL para la clasificación de imágenes. RESNET-50 ofrece un resultado de entrenamiento preciso con un tiempo de procesamiento más rápido, lo que nos permitió generar una demanda suficiente sobre el almacenamiento.

## Ejecutar:instalación de AI

Para instalar Run:AI, lleve a cabo los siguientes pasos:

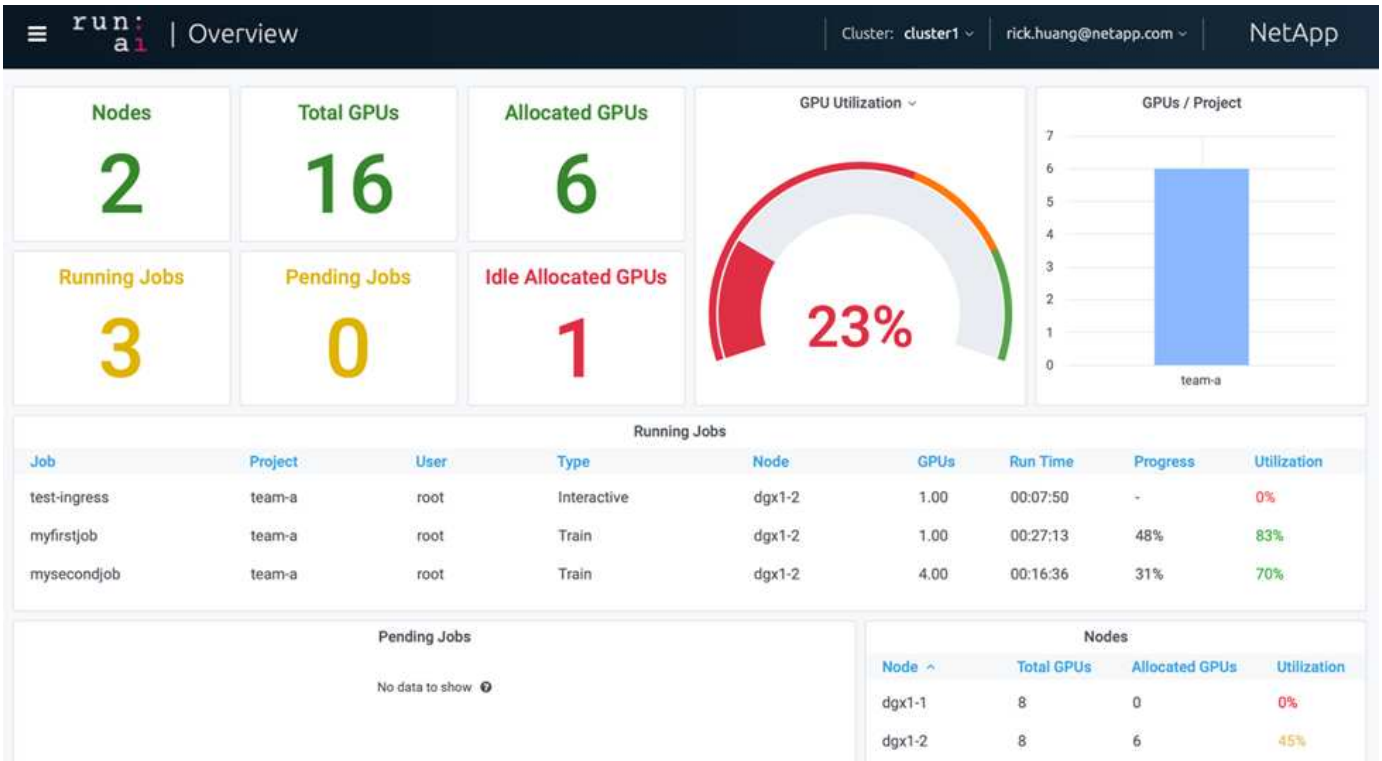
1. Instale el clúster de Kubernetes con DeepOps y configure la clase de almacenamiento predeterminada de NetApp.
2. Prepare los nodos de GPU:
   a. Compruebe que los controladores de NVIDIA están instalados en los nodos de la GPU.
   b. Compruebe que `nvidia-docker` se instala y configura como el tiempo de ejecución de docker predeterminado.
3. Ejecución de instalación:AI:
   a. Inicie sesión en el "Ejecución: IU de administración de IA" para crear el clúster.
   b. Descargue el creado `runai-operator-<clustername>.yaml` archivo.
   c. Aplique la configuración del operador al clúster de Kubernetes.

   ```
   kubectl apply -f runai-operator-<clustername>.yaml
   ```

4. Compruebe la instalación:
   a. Vaya a. "https://app.run.ai/".
   b. Vaya a la consola Overview.
   c. Compruebe que el número de GPU de la parte superior derecha refleja el número esperado de GPU y los nodos de GPU forman parte de la lista de servidores. Para obtener más información acerca de la puesta en marcha de Run:IA, consulte "Instalar Run:IA en un clúster de Kubernetes en las instalaciones" y.. "Instalación de la CLI Run:AI".

## Ejecutar:Paneles de IA y vistas

Después de instalar Run:AI en su clúster de Kubernetes y configurar los contenedores correctamente, verá las siguientes consolas y vistas en "https://app.run.ai" en su navegador, como se muestra en la siguiente figura.




Hay 16 GPU en total en el clúster proporcionados por dos nodos DGX-1. Puede ver el número de nodos, el total de GPU disponibles, las GPU asignadas con cargas de trabajo, el número total de trabajos en ejecución, los trabajos pendientes y las GPU asignadas inactivas. En el lado derecho, el diagrama de barras muestra las GPU por proyecto, que resume cómo usan los distintos equipos el recurso de clúster. En el medio se encuentra la lista de trabajos actualmente en ejecución con detalles de trabajo, incluidos el nombre del trabajo, el proyecto, el usuario, el tipo de trabajo, El nodo en el que se ejecuta cada trabajo, el número de GPU asignados para ese trabajo, el tiempo de ejecución actual del trabajo, el progreso del trabajo en porcentaje y el uso de la GPU para ese trabajo. Tenga en cuenta que el clúster está infrautilizado (uso de la GPU al 23%) porque solo hay tres trabajos en ejecución enviados por un único equipo (`team-a`).

En la siguiente sección, mostramos cómo crear varios equipos en la pestaña proyectos y asignar GPU para cada equipo con el fin de maximizar el uso del clúster y gestionar los recursos cuando hay muchos usuarios por clúster. Los escenarios de prueba imitan entornos empresariales en los que los recursos de memoria y GPU se comparten entre cargas de trabajo de entrenamiento, inferencia e interactivas.

# Creación de proyectos para equipos de ciencia de datos y asignación de GPU

Los investigadores pueden enviar cargas de trabajo a través de la CLI Run:AI, Kubeflow o procesos similares. Para agilizar la asignación de recursos y crear prioridades, Run:AI introduce el concepto de proyectos. Los proyectos son entidades de cuota que asocian un nombre de proyecto con la asignación y las preferencias de la GPU. Se trata de una forma sencilla y cómoda de gestionar varios equipos de ciencia de datos.

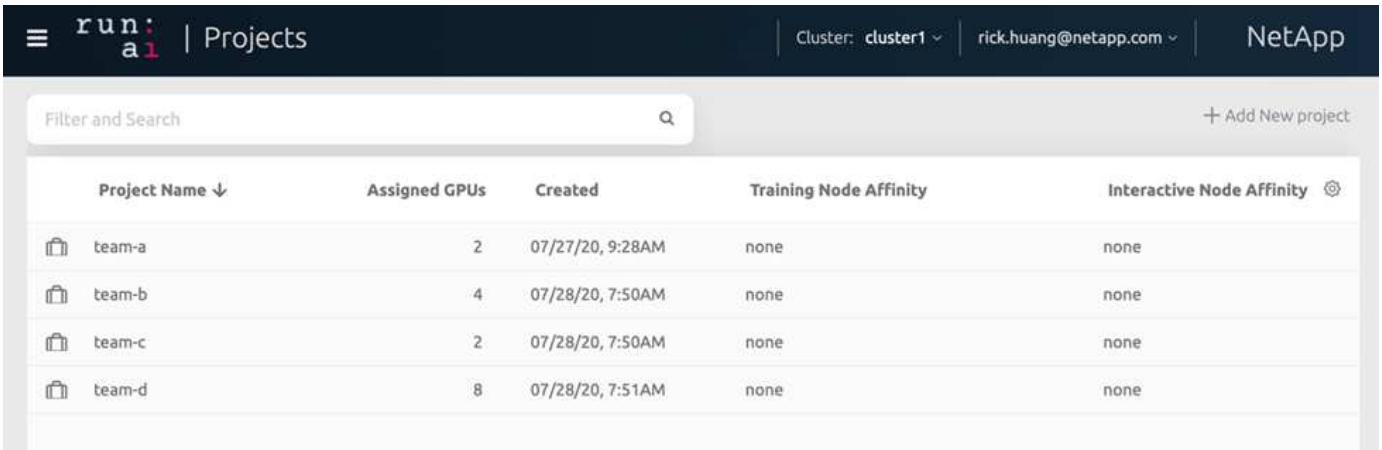
Un investigador que presenta una carga de trabajo debe asociar un proyecto con una solicitud de carga de trabajo. El programador Run:AI compara la solicitud con las asignaciones actuales y el proyecto, y determina si la carga de trabajo puede asignarse recursos o si debe permanecer en estado pendiente.

Como administrador del sistema, puede establecer los siguientes parámetros en la ficha Ejecutar:proyectos

AI:

- **Proyectos modelo.** establecer un proyecto por usuario, establecer un proyecto por equipo de usuarios y establecer un proyecto por proyecto de organización real.
- * Cuotas del proyecto.* cada proyecto está asociado con una cuota de GPU que se pueden asignar para este proyecto al mismo tiempo. Se trata de una cuota garantizada en el sentido de que se garantiza que los investigadores que utilizan este proyecto obtengan este número de GPU, independientemente del estado del clúster. Por lo general, la suma de la asignación de proyectos debe ser igual al número de GPU del clúster. Más allá de eso, un usuario de este proyecto puede recibir un exceso de cuota. Mientras no se utilicen las GPU, un investigador que usa este proyecto puede obtener más GPU. Demostramos escenarios de prueba de exceso de cuota y consideraciones de equidad en "Utilización óptima del clúster gracias a la asignación de GPU por encima de una cuota", "Equidad en la asignación de recursos básicos", y. "Justicia por exceso de cuotas".
- Cree un proyecto nuevo, actualice un proyecto existente y elimine un proyecto existente.
- **Limite los trabajos para que se ejecuten en grupos de nodos específicos**. Puede asignar proyectos específicos para que se ejecuten solo en nodos específicos. Esto resulta útil cuando el equipo de proyecto necesita hardware especializado, por ejemplo, con suficiente memoria. Como alternativa, un equipo de proyecto podría ser el propietario de hardware específico que se adquirió con un presupuesto especializado o cuando sea necesario dirigir las cargas de trabajo de creación o interactivas para trabajar en hardware más débil y dirigir la formación de mayor duración o las cargas de trabajo sin supervisión a nodos más rápidos. Para ver los comandos para agrupar nodos y establecer la afinidad de un proyecto específico, consulte "Ejecute:Documentación de IA".
- **Limitar la duración de los trabajos interactivos**. Los investigadores se olvidan frecuentemente de cerrar trabajos interactivos. Esto podría conducir a una pérdida de recursos. Algunas organizaciones prefieren limitar la duración de los trabajos interactivos y cerrarlos automáticamente.

La siguiente figura muestra la vista proyectos con cuatro equipos creados. A cada equipo se le asigna un número diferente de GPU para dar cuenta de diferentes cargas de trabajo, con un número total de GPU igual al de las GPU disponibles totales en un clúster que consta de dos DGX-1s.




# Enviar trabajos en ejecución:AI CLI

En esta sección se ofrecen detalles sobre los comandos básicos Run:AI que puede utilizar para ejecutar cualquier trabajo con Kubernetes. Se divide en tres partes según el tipo de carga de trabajo. Las cargas de trabajo de IA/ML/AP se pueden dividir en dos tipos genéricos:

- **Sesiones de formación desatendida**. Con estos tipos de cargas de trabajo, el científico de datos prepara una carga de trabajo de ejecución automática y la envía para su ejecución. Durante la ejecución, el cliente puede examinar los resultados. Este tipo de carga de trabajo se utiliza a menudo en la producción o cuando el desarrollo de modelos se encuentra en una etapa en la que no se requiere intervención humana.
- **Sesiones interactivas de construcción**. Con este tipo de cargas de trabajo, el científico de datos abre una sesión interactiva con Bash, Juppyter Notebook, PyCharm remoto o IDE similares y accede directamente a los recursos de la GPU. Incluimos un tercer escenario para ejecutar cargas de trabajo interactivas con puertos conectados a fin de revelar un puerto interno al usuario del contenedor.

## Cargas de trabajo de formación desatendida

Después de configurar los proyectos y asignar GPU, puede ejecutar cualquier carga de trabajo de Kubernetes usando el siguiente comando en la línea de comandos:

```
$ runai project set team-a runai submit hyper1 -i gcr.io/run-ai-
demo/quickstart -g 1
```

Este comando inicia un trabajo de entrenamiento desatendido para el equipo a con una asignación de una única GPU. El trabajo se basa en una imagen de docker de muestra, `gcr.io/run-ai-demo/quickstart`. Nosotros nombramos el trabajo `hyper1`. A continuación, puede supervisar el progreso del trabajo ejecutando el siguiente comando:

```
$ runai list
```

En la siguiente figura se muestra el resultado del `runai list` comando. Los Estados típicos que puede ver incluyen los siguientes:

- `ContainerCreating`. El contenedor docker se está descargando del repositorio en la nube.
- `Pending`. El trabajo está a la espera de ser programado.
- `Running`. El trabajo se está ejecutando.

```
~> runai list
Showing jobs for project team-a
NAME    STATUS   AGE  NODE                                    IMAGE                        TYPE         PROJECT  USER   GPUs
hyper1  Running  11s  gke-dev-yaron1-gpu-4-pool-154f511d-5nk5  gcr.io/run-ai-demo/quickstart  Train        team-a   yaron  1
```

Para obtener un estado adicional en su trabajo, ejecute el siguiente comando:

```
$ runai get hyper1
```

Para ver los registros del trabajo, ejecute el `runai logs <job-name>` comando:

```
$ runai logs hyper1
```

En este ejemplo, debería ver el registro de una sesión DL en ejecución, incluyendo la época de entrenamiento actual, ETA, valor de la función de pérdida, precisión y tiempo transcurrido para cada paso.

Puede ver el estado del clúster en la interfaz de usuario Run:AI en "https://app.run.ai/". En Paneles > Descripción general, puede supervisar el uso de la GPU.

Para detener esta carga de trabajo, ejecute el siguiente comando:

```
$ runai delte hyper1
```

Este comando detiene la carga de trabajo de entrenamiento. Puede verificar esta acción ejecutando `runai list` de nuevo. Para obtener información detallada, consulte "iniciar cargas de trabajo de formación sin supervisión".

## Cargas de trabajo de compilación interactiva

Después de configurar proyectos y asignar GPU, puede ejecutar una carga de trabajo de compilación interactiva utilizando el siguiente comando en la línea de comandos:

```
$ runai submit build1 -i python -g 1 --interactive --command sleep --args
infinity
```

El trabajo se basa en un pitón de imagen de Docker de muestra. Nombramos el edificio de trabajo 1.

> La `-- interactive` indicador significa que el trabajo no tiene inicio ni fin Es responsabilidad del investigador cerrar el trabajo. El administrador puede definir un límite de tiempo para los trabajos interactivos después de que el sistema los termine.

La `--g 1` El indicador asigna una única GPU a este trabajo. El comando y el argumento proporcionados son `--command sleep—args infinity`. Debe proporcionar un comando o el contenedor se inicia y, a continuación, sale inmediatamente.

Los siguientes comandos funcionan de forma similar a los comandos descritos en Cargas de trabajo de formación desatendida:

- `runai list`: Muestra el nombre, el estado, la edad, el nodo, la imagen, Proyecto, usuario y GPU para trabajos.
- `runai get build1`: Muestra el estado adicional en la generación de trabajos 1.
- `runai delete build1`: Detiene la carga de trabajo interactiva build1.para obtener un shell bash en el contenedor, el siguiente comando:

```
$ runai bash build1
```

Esto proporciona un shell directo en el equipo. A continuación, los científicos de datos pueden desarrollar o afinar sus modelos dentro del contenedor.

Puede ver el estado del clúster en la interfaz de usuario Run:AI en "https://app.run.ai". Para obtener

información detallada, consulte "iniciar y utilizar cargas de trabajo de compilación interactivas".

### Cargas de trabajo interactivas con puertos conectados

Como extensión de las cargas de trabajo de compilación interactiva, puede revelar puertos internos al usuario del contenedor al iniciar un contenedor con la CLI Run:AI. Esto resulta útil para entornos cloud, trabajar con Juppyter Notebooks o conectarse a otros microservicios. "Entrada" Permite el acceso a los servicios de Kubernetes desde fuera del clúster de Kubernetes. Puede configurar el acceso mediante la creación de una colección de reglas que definan qué conexiones entrantes llegan a qué servicios.

Para mejorar la gestión del acceso externo a los servicios de un clúster, sugerimos que instalen los administradores de clúster "Entrada" Y configurar LoadBalancer.

Para utilizar Ingress como tipo de servicio, ejecute el siguiente comando para establecer el tipo de método y los puertos al enviar la carga de trabajo:

```
$ runai submit test-ingress -i jupyter/base-notebook -g 1 \
  --interactive --service-type=ingress --port 8888 \
  --args="--NotebookApp.base_url=test-ingress" --command=start-notebook.sh
```

Una vez que el contenedor se haya iniciado correctamente, ejecute `runai list` para ver la `SERVICE URL(S)` Con el que acceder al Cuaderno de Jupyter. La dirección URL está compuesta por el punto final de entrada, el nombre del trabajo y el puerto. Por ejemplo, consulte https://10.255.174.13/test-ingress-8888.

Para obtener información detallada, consulte "iniciar una carga de trabajo de compilación interactiva con puertos conectados".

## Se obtiene un uso elevado del clúster

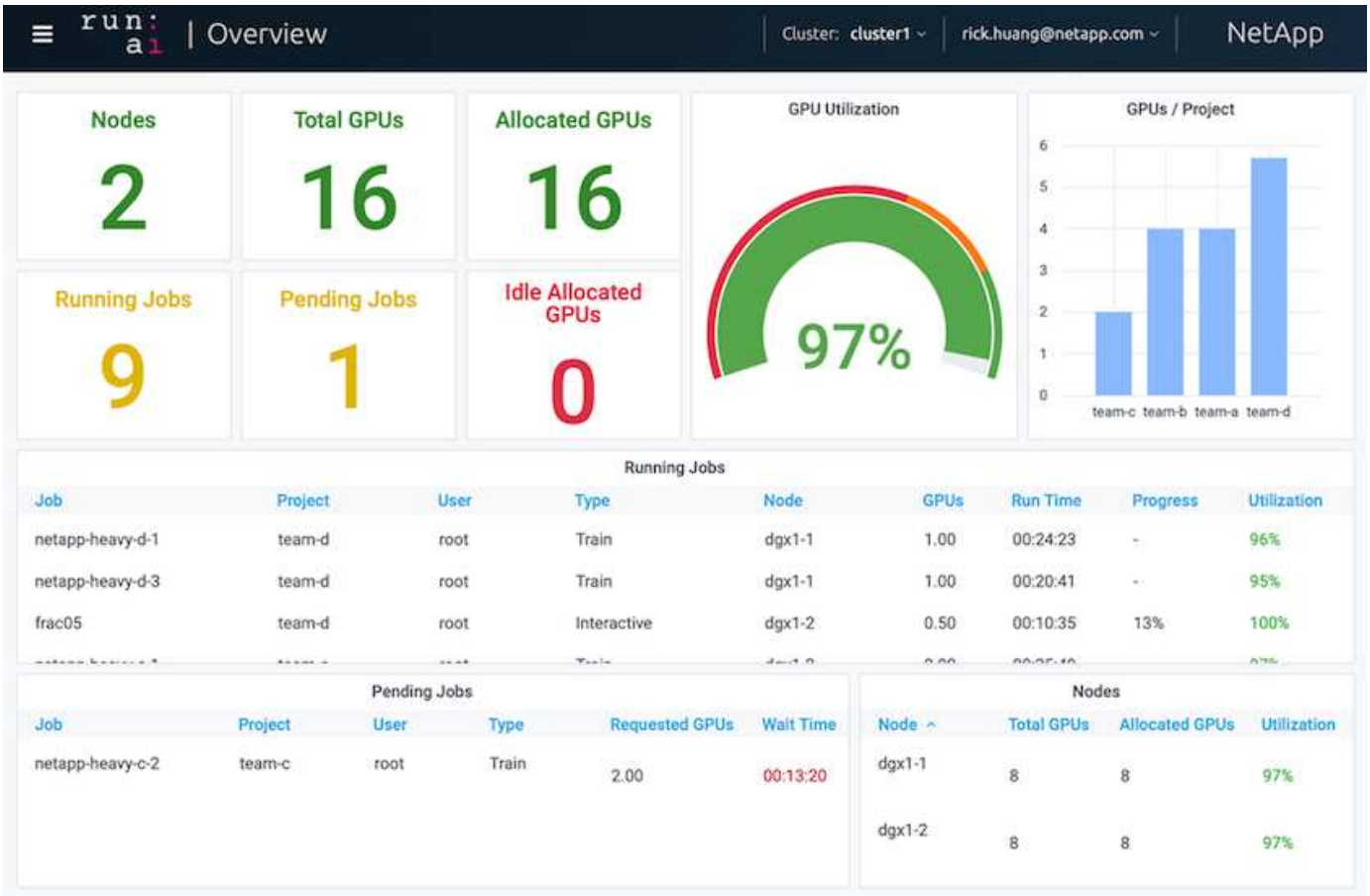
En esta sección, emulamos un escenario realista en el que cuatro equipos de ciencia de datos envían sus propias cargas de trabajo para demostrar la solución de orquestación Run:AI que logra un uso elevado del clúster mientras mantiene la priorización y el equilibrio de los recursos de la GPU. Empezamos utilizando la prueba de rendimiento ResNet-50 descrita en el apartado "RESNET-50 con resumen de la prueba de rendimiento de conjunto de datos ImageNET":

```
$ runai submit netapp1 -i netapp/tensorflow-tf1-py3:20.01.0 --local-image
--large-shm  -v /mnt:/mnt -v /tmp:/tmp --command python --args
"/netapp/scripts/run.py" --args "--
dataset_dir=/mnt/mount_0/dataset/imagenet/imagenet_original/" --args "--
num_mounts=2"  --args "--dgx_version=dgx1" --args "--num_devices=1" -g 1
```

Ejecutamos la misma prueba de rendimiento ResNet-50 como en la "NVA-1121". Usamos la bandera `--local-image` para contenedores que no residen en el repositorio de docker público. Montamos los directorios `/mnt` y.. `/tmp` En el nodo DGX-1 del host a. `/mnt` y.. `/tmp` al contenedor, respectivamente. El conjunto de datos se encuentra en NetApp AFFA800 con el `dataset_dir` argumento que apunta al directorio. Ambas `--num_devices=1` y.. `-g 1` Significa que asignamos una GPU para este trabajo. El primero es un argumento para el `run.py` script, mientras que el último es un indicador para el `runai`

`submit` comando.

La siguiente figura muestra un panel de información general del sistema con un uso del 97 % de la GPU y las dieciséis GPU disponibles asignadas. Puede ver fácilmente cuántas GPU se asignan a cada equipo en el gráfico de barras GPU/proyecto. El panel trabajos en ejecución muestra los nombres de los trabajos en ejecución actuales, el proyecto, el usuario, el tipo, el nodo, Las GPU consumidas, tiempo de ejecución, progreso y detalles de uso. Se muestra una lista de las cargas de trabajo que están en cola con el tiempo de espera en trabajos pendientes. Finalmente, el recuadro Nodes ofrece cifras de GPU y utilización de nodos DGX-1 individuales en el clúster.




## Asignación de GPU fraccionaria para las cargas de trabajo menos exigentes o interactivas

Cuando investigadores y desarrolladores trabajan en sus modelos, ya sea en las etapas de desarrollo, ajuste de hiperparámetros o depuración, estas cargas de trabajo suelen requerir menos recursos computacionales. Por lo tanto, es más eficiente aprovisionar GPU y memoria fraccionarias de modo que la misma GPU se pueda asignar simultáneamente a otras cargas de trabajo. Ejecutar:la solución de orquestación de IA proporciona un sistema de uso compartido de GPU fraccionario para las cargas de trabajo en contenedores en Kubernetes. El sistema admite cargas de trabajo que ejecutan programas CUDA y se adapta especialmente a tareas de IA ligeras como la inferencia y la creación de modelos. El sistema de GPU fraccionaria proporciona a los equipos de ciencia de datos e ingeniería de IA la capacidad de ejecutar varias cargas de

trabajo simultáneamente en una única GPU. De este modo, las empresas pueden ejecutar más cargas de trabajo, como visión informática, reconocimiento de voz y procesamiento de lenguaje natural en el mismo hardware, con lo que se reducen los costes.

Ejecutar:el sistema de GPU fraccionaria de IA crea de manera efectiva GPU lógicas virtualizadas con su propia memoria y espacio de computación que los contenedores pueden utilizar y acceder como si fueran procesadores independientes. De este modo, es posible ejecutar varias cargas de trabajo en contenedores en paralelo y en la misma GPU sin interferir entre sí. La solución es transparente, sencilla y portátil y no requiere ningún cambio en los contenedores en sí.

Una usecasa típica podría ver dos a ocho trabajos ejecutándose en la misma GPU, lo que significa que podría trabajar ocho veces más con el mismo hardware.

Para el trabajo `frac05` perteneciente al proyecto `team-d` En la siguiente figura podemos comprobar que el número de GPU asignadas era de 0.50. Esto es verificado por el `nvidia-smi` Comando, que muestra que la memoria de la GPU disponible para el contenedor era de 16,255 MB: La mitad de las 32 GB por GPU V100 en el nodo DGX-1.

```
root@run-deploy:~# runai bash frac05 -p team-d
root@frac05-0:/workload# nvidia-smi
Tue Jul 28 15:17:03 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 450.51.05    Driver Version: 450.51.05    CUDA Version: 11.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  On   | 00000000:07:00.0 Off |                    0 |
| N/A   57C    P0   240W / 300W |  15525MiB / 16255MiB |    100%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A       156      C   python3                         15525MiB |
+-----------------------------------------------------------------------------+
```

## Utilización óptima del clúster gracias a la asignación de GPU por encima de una cuota

En esta sección y en las secciones "Equidad en la asignación de recursos básicos", y. "Justicia por exceso de cuotas", Hemos diseñado escenarios de pruebas avanzados para demostrar las capacidades de orquestación Run:AI para la administración de

cargas de trabajo complejas, la programación preventiva automática y el aprovisionamiento de GPU con exceso de cuota. Hemos hecho esto para lograr un uso elevado de los recursos de clúster y optimizar la productividad de los equipos científicos de datos a nivel empresarial en un entorno de IA de ONTAP.

Para estas tres secciones, defina los siguientes proyectos y cuotas:

| Proyecto | Cuota |
|---|---|
| equipo a | 4 |
| equipo-b | 2 |
| equipo-c | 2 |
| equipo d | 8 |

Además, utilizamos los siguientes contenedores para estas tres secciones:

- Portátil Jupyter: `jupyter/base-notebook`
- Ejecutar:AI Quickstart: `gcr.io/run-ai-demo/quickstart`

Definimos los siguientes objetivos para este escenario de prueba:

- Muestre la simplicidad del aprovisionamiento de recursos y cómo los recursos se abstraen de los usuarios
- Mostrar cómo los usuarios pueden aprovisionar fácilmente fracciones de una GPU y un número entero de GPU
- Muestre cómo el sistema elimina los cuellos de botella de computación al permitir que equipos o usuarios hagan uso de su cuota de recursos si hay GPU libres en el clúster
- Muestre cómo se eliminan los cuellos de botella en la canalización de datos utilizando la solución de NetApp cuando se ejecutan tareas con un gran procesamiento como el contenedor de NetApp
- Muestra cómo se ejecutan los diversos tipos de contenedores mediante el sistema
  - Portátil Jupyter
  - Ejecute:contenedor de IA
- Muestra una alta utilización cuando el clúster está lleno

Para obtener información detallada sobre la secuencia de comandos real ejecutada durante la prueba, consulte "Detalles de las pruebas para la sección 4.8".

Cuando se envían las 13 cargas de trabajo, puede ver una lista con los nombres de los contenedores y las GPU asignadas, como se muestra en la siguiente figura. Contamos con siete cursos de formación y seis trabajos interactivos, que simulan cuatro equipos de ciencia de datos, cada uno con sus propios modelos en ejecución o en desarrollo. Para trabajos interactivos, los desarrolladores individuales están utilizando Jupyter Notebooks para escribir o depurar su código. Por lo tanto, es adecuado aprovisionar fracciones de GPU sin usar demasiados recursos de clúster.

```
root@run-deploy:~# runai list -A
NAME           STATUS   AGE  NODE    IMAGE                          TYPE        PROJECT  USER  GPUs  CREATED BY CLI  SERVICE URL(S)
b-4-gg         Running  2m   dgx1-2  gcr.io/run-ai-demo/quickstart  Train       team-b   root  2     true
c-5-g          Running  2m   dgx1-2  gcr.io/run-ai-demo/quickstart  Train       team-c   root  1     true
c-4-gg         Running  2m   dgx1-1  gcr.io/run-ai-demo/quickstart  Train       team-c   root  2     true
b-3-g          Running  2m   dgx1-1  gcr.io/run-ai-demo/quickstart  Train       team-b   root  1     true
c-3-g02        Running  2m   dgx1-1  gcr.io/run-ai-demo/quickstart  Interactive team-c   root  0.2   true
d-1-gggg       Running  2m   dgx1-2  gcr.io/run-ai-demo/quickstart  Train       team-d   root  4     true
c-2-g03        Running  2m   dgx1-1  gcr.io/run-ai-demo/quickstart  Interactive team-c   root  0.3   true
c-1-g05        Running  2m   dgx1-1  gcr.io/run-ai-demo/quickstart  Interactive team-c   root  0.5   true
a-2-gg         Running  3m   dgx1-1  gcr.io/run-ai-demo/quickstart  Train       team-a   root  2     true
b-2-g04        Running  3m   dgx1-2  gcr.io/run-ai-demo/quickstart  Interactive team-b   root  0.4   true
a-1-g          Running  3m   dgx1-1  gcr.io/run-ai-demo/quickstart  Train       team-a   root  1     true
b-1-g06        Running  3m   dgx1-2  gcr.io/run-ai-demo/quickstart  Interactive team-b   root  0.6   true
a-1-1-jupyter  Running  3m   dgx1-1  jupyter/base-notebook          Interactive team-a   root  1     true            http://10.61.218.134/a-1-1-jupyter,
https://10.61.218.134/a-1-1-jupyter
```

Los resultados de este escenario de prueba muestran lo siguiente:

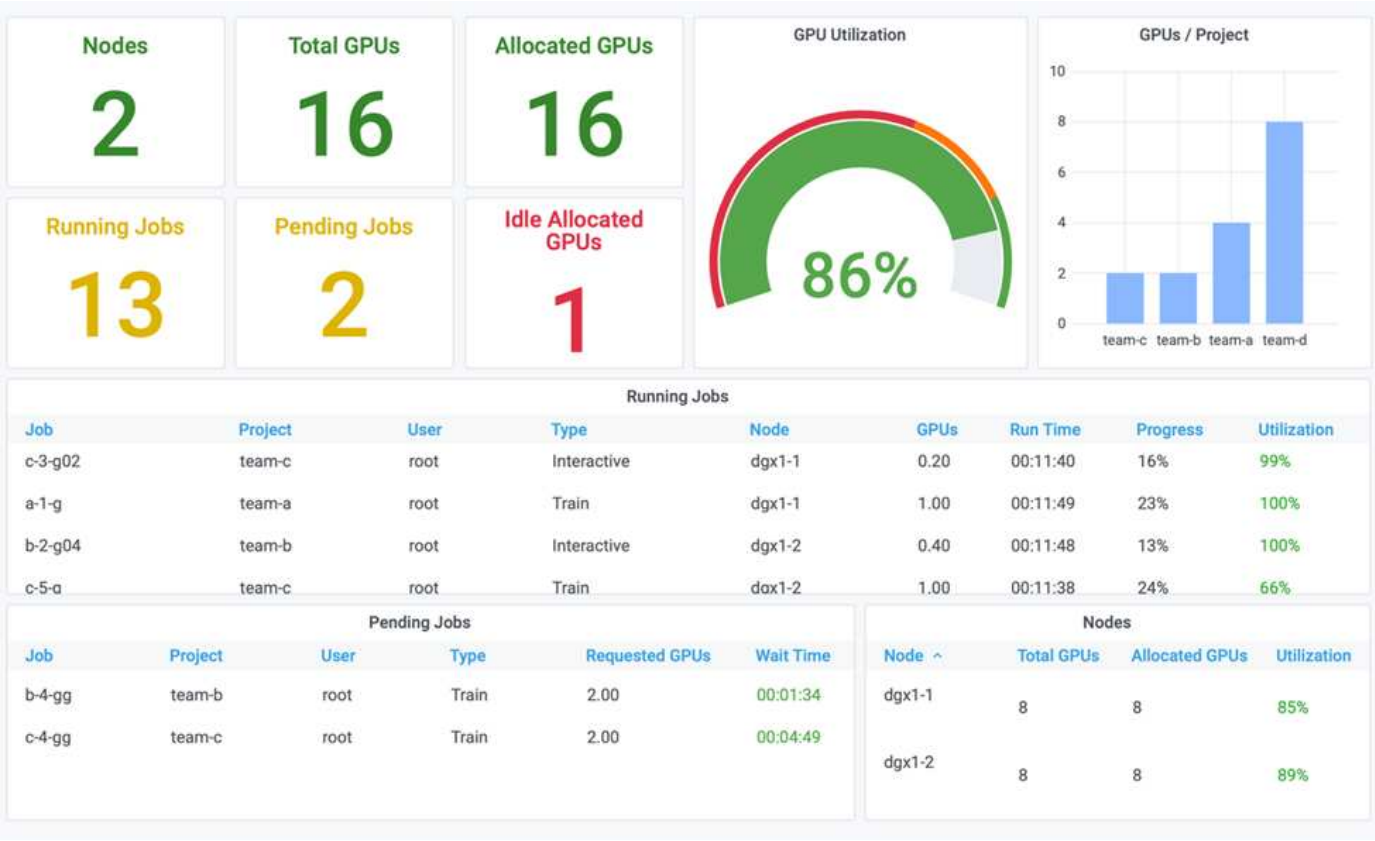
- El clúster debe estar lleno: Se utilizan 16/16 GPU.
- Uso elevado de clúster.
- Más experimentos que las GPU debido a la asignación fraccionaria.
- `team-d` no está utilizando toda su cuota; por lo tanto, `team-b` y.. `team-c` Puede utilizar más GPU para sus experimentos, lo que acelera el plazo de innovación.

# Equidad en la asignación de recursos básicos

En esta sección, lo mostramos cuando `team-d` Solicita más GPU (están por debajo de su cuota), el sistema coloca en pausa las cargas de trabajo de `team-b` y.. `team-c` y los mueve a un estado pendiente de manera equitativa.

Para obtener información detallada, incluidos los envíos de trabajos, las imágenes contenedoras utilizadas y las secuencias de comandos ejecutadas, consulte la sección "Detalles de las pruebas para la sección 4.9".

La siguiente figura muestra el uso resultante del clúster, las GPU asignadas por equipo y los trabajos pendientes debido al equilibrio de carga automático y a la programación preventiva. Podemos observar que cuando el número total de GPU solicitado por todas las cargas de trabajo de equipo supera el total de GPU disponibles en el clúster, el algoritmo de integridad interno de Run:AI pone en pausa un trabajo por cada una `team-b` y.. `team-c` porque han cumplido con su cuota de proyecto. Esto proporciona una utilización elevada del clúster global, mientras que los equipos de ciencia de datos siguen trabajando con las limitaciones de recursos definidas por un administrador.




Los resultados de este escenario de prueba demuestran lo siguiente:

- **Equilibrio de carga automático.** el sistema equilibra automáticamente la cuota de las GPU, de modo que cada equipo utiliza ahora su cuota. Las cargas de trabajo en pausa pertenecen a equipos que se encontraban por encima de su cuota.
- **Pausa de uso compartido justo.** el sistema elige detener la carga de trabajo de un equipo que estaba por encima de su cuota y luego detener la carga de trabajo del otro equipo. Ejecutar:la IA tiene algoritmos internos de justicia.

# Justicia por exceso de cuotas

En esta sección, ampliamos el escenario en el que varios equipos envían cargas de trabajo y superan su cuota. De esta manera, mostramos cómo el algoritmo de equidad de Run:AI asigna recursos de clúster según la proporción de cuotas predefinidas.

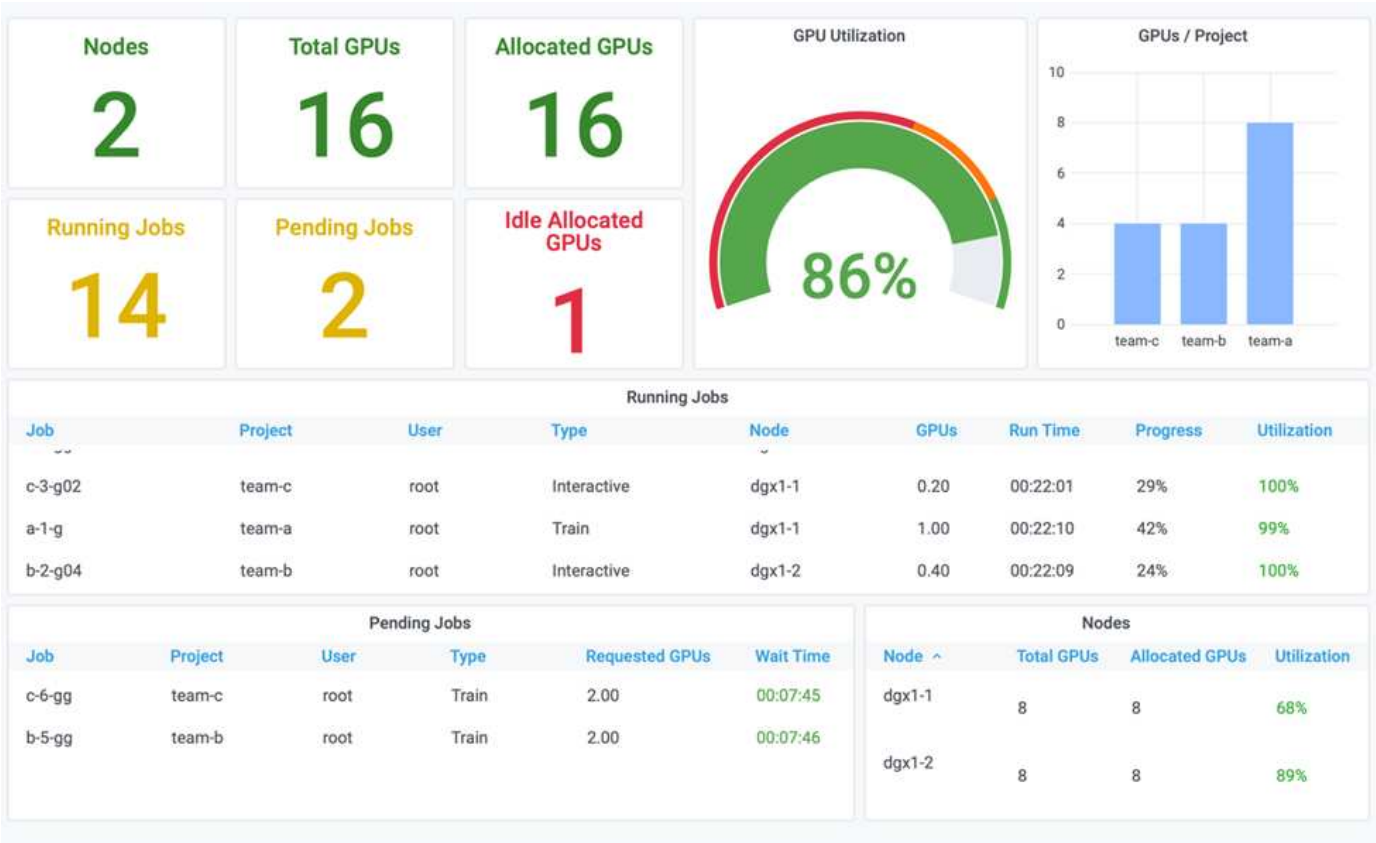
Objetivos de este escenario de prueba:

- Muestra el mecanismo de cola cuando varios equipos solicitan GPU a través de su cuota.
- Muestre cómo distribuye el sistema una cuota del clúster entre varios equipos que se encuentran por encima de su cuota en función de la proporción entre sus cuotas, de modo que el equipo con la cuota mayor obtenga una parte mayor de la capacidad de reserva.

Al final de "Equidad en la asignación de recursos básicos", hay dos cargas de trabajo en cola: una para `team-b` y uno para `team-c`. En esta sección, ponemos en cola las cargas de trabajo adicionales.

Para obtener información detallada, incluidos los envíos de trabajos, las imágenes contenedoras utilizadas y las secuencias de comandos ejecutadas, consulte "Detalles de la prueba para la sección 4.10".

Cuando todos los trabajos se someten según la sección "Detalles de la prueba para la sección 4.10", el panel del sistema muestra eso `team-a`, `team-b`, y. `team-c` Todos tienen más GPU que su cuota predefinida. `team-a` Ocupa cuatro GPU más que su cuota de software predefinida (cuatro), mientras que `team-b` y.. `team-c` Cada uno ocupa dos GPU más que su cuota blanda (dos). La proporción de GPU asignadas por encima de una cuota es igual a la de su cuota predefinida. Esto se debe a que el sistema utilizó la cuota preconfigurada como referencia de prioridad y se aprovisionó de manera acorde cuando varios equipos soliciten más GPU, por lo que superó su cuota. Este equilibrio de carga automático proporciona justicia y priorización cuando los equipos de ciencia de datos empresariales están involucrados activamente en el desarrollo y la producción de modelos de IA.




Los resultados de este escenario de prueba muestran lo siguiente:

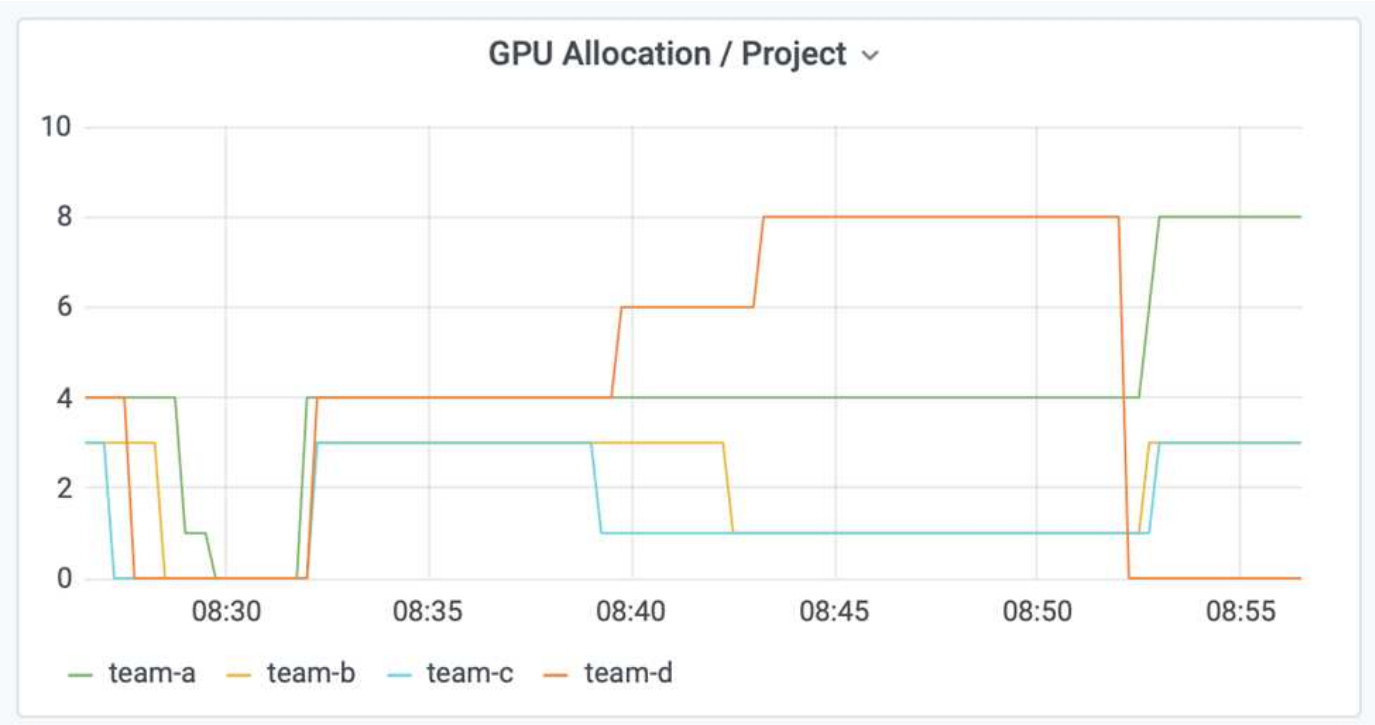
- El sistema empieza a deponer las cargas de trabajo de otros equipos en cola.
- El orden de la depuesta en cola se decide de acuerdo a los algoritmos de justicia, tal que `team-b` y.. `team-c` Obtenga la misma cantidad de GPU de cuota superior (ya que tienen una cuota similar) y. `team-a` Obtiene una cantidad doble de GPU, ya que su cuota es dos veces superior a la de `team-b` y.. `team-c`.
- Toda la asignación se realiza automáticamente.

Por lo tanto, el sistema debe estabilizarse en los siguientes estados:

| Proyecto | GPU asignadas | Comentar |
|---|---|---|
| equipo a | 8/4 | Cuatro GPU a lo largo de la cuota. Cola vacía. |
| equipo-b | 4/2 | Dos GPU por encima de la cuota. Una carga de trabajo en cola. |

| Proyecto | GPU asignadas | Comentar |
|---|---|---|
| equipo-c | 4/2 | Dos GPU por encima de la cuota. Una carga de trabajo en cola. |
| equipo d | 0/8 | No utiliza GPU en absoluto, sin cargas de trabajo en cola. |

La siguiente figura muestra la asignación de GPU por proyecto a lo largo del tiempo en el panel Run:AI Analytics de las secciones "Utilización óptima del clúster gracias a la asignación de GPU por encima de una cuota", "Equidad en la asignación de recursos básicos", y. "Justicia por exceso de cuotas". Cada línea de la figura indica el número de GPU aprovisionadas para un equipo de ciencias de datos en cualquier momento. Podemos observar que el sistema asigna dinámicamente las GPU de acuerdo con las cargas de trabajo enviadas. Esto permite a los equipos revisar las cuotas cuando hay GPU disponibles en el clúster y, después, adelantarse a los trabajos según sea necesario, antes de alcanzar, por fin, un estado estable para los cuatro equipos.




# Guardar datos en un volumen persistente aprovisionado por Trident

Trident de NetApp es un proyecto de código abierto totalmente compatible diseñado para ayudarle a satisfacer las sofisticadas demandas de persistencia de sus aplicaciones en contenedores. Puede leer y escribir datos en un volumen persistente (VP) de Kubernetes aprovisionado por Trident con la ventaja añadida de la organización en niveles de datos, el cifrado, la tecnología Snapshot de NetApp, el cumplimiento de normativas y el alto rendimiento que ofrece el software de gestión de datos ONTAP de NetApp.

## Reutilizar EVs en un espacio de nombres existente

En el caso de proyectos de IA de mayor tamaño, es posible que sea más eficiente que diferentes contenedores para leer y escribir datos en el mismo PV de Kubernetes. Para reutilizar una solicitud de volumen persistente de Kubernetes (PVC), el usuario ya debe haber creado una RVP. Consulte "Documentación de Trident de NetApp" Para obtener más detalles sobre la creación de una RVP. A continuación se muestra un ejemplo de reutilización de un PVC existente:

```
$ runai submit pvc-test -p team-a --pvc test:/tmp/pvc1mount -i gcr.io/run-
ai-demo/quickstart -g 1
```

Ejecute el siguiente comando para ver el estado del trabajo `pvc-test` para el proyecto `team-a`:

```
$ runai get pvc-test -p team-a
```

Debe ver el montaje PV /tmp/pvc1montado a. `team-a` trabajo `pvc-test`. De este modo, se pueden leer varios contenedores desde el mismo volumen, lo que resulta útil cuando hay varios modelos competidores en desarrollo o producción. Los científicos de datos pueden crear un conjunto de modelos y, a continuación, combinar los resultados de las predicciones por voto mayoritario u otras técnicas.

Utilice lo siguiente para acceder al shell del contenedor:

```
$ runai bash pvc-test -p team-a
```

A continuación, puede comprobar el volumen montado y acceder a los datos dentro del contenedor.

Esta capacidad de reutilización de RVP funciona con volúmenes FlexVol de NetApp y volúmenes ONTAP FlexGroup de NetApp, lo que permite a los ingenieros de datos disfrutar de opciones de gestión de datos más flexibles y sólidas para aprovechar su Data Fabric con tecnología de NetApp.

**Información de copyright**

Copyright © 2024 NetApp, Inc. Todos los derechos reservados. Imprimido en EE. UU. No se puede reproducir este documento protegido por copyright ni parte del mismo de ninguna forma ni por ningún medio (gráfico, electrónico o mecánico, incluidas fotocopias, grabaciones o almacenamiento en un sistema de recuperación electrónico) sin la autorización previa y por escrito del propietario del copyright.

El software derivado del material de NetApp con copyright está sujeto a la siguiente licencia y exención de responsabilidad:

ESTE SOFTWARE LO PROPORCIONA NETAPP «TAL CUAL» Y SIN NINGUNA GARANTÍA EXPRESA O IMPLÍCITA, INCLUYENDO, SIN LIMITAR, LAS GARANTÍAS IMPLÍCITAS DE COMERCIALIZACIÓN O IDONEIDAD PARA UN FIN CONCRETO, CUYA RESPONSABILIDAD QUEDA EXIMIDA POR EL PRESENTE DOCUMENTO. EN NINGÚN CASO NETAPP SERÁ RESPONSABLE DE NINGÚN DAÑO DIRECTO, INDIRECTO, ESPECIAL, EJEMPLAR O RESULTANTE (INCLUYENDO, ENTRE OTROS, LA OBTENCIÓN DE BIENES O SERVICIOS SUSTITUTIVOS, PÉRDIDA DE USO, DE DATOS O DE BENEFICIOS, O INTERRUPCIÓN DE LA ACTIVIDAD EMPRESARIAL) CUALQUIERA SEA EL MODO EN EL QUE SE PRODUJERON Y LA TEORÍA DE RESPONSABILIDAD QUE SE APLIQUE, YA SEA EN CONTRATO, RESPONSABILIDAD OBJETIVA O AGRAVIO (INCLUIDA LA NEGLIGENCIA U OTRO TIPO), QUE SURJAN DE ALGÚN MODO DEL USO DE ESTE SOFTWARE, INCLUSO SI HUBIEREN SIDO ADVERTIDOS DE LA POSIBILIDAD DE TALES DAÑOS.

NetApp se reserva el derecho de modificar cualquiera de los productos aquí descritos en cualquier momento y sin aviso previo. NetApp no asume ningún tipo de responsabilidad que surja del uso de los productos aquí descritos, excepto aquello expresamente acordado por escrito por parte de NetApp. El uso o adquisición de este producto no lleva implícita ninguna licencia con derechos de patente, de marcas comerciales o cualquier otro derecho de propiedad intelectual de NetApp.

Es posible que el producto que se describe en este manual esté protegido por una o más patentes de EE. UU., patentes extranjeras o solicitudes pendientes.

LEYENDA DE DERECHOS LIMITADOS: el uso, la copia o la divulgación por parte del gobierno están sujetos a las restricciones establecidas en el subpárrafo (b)(3) de los derechos de datos técnicos y productos no comerciales de DFARS 252.227-7013 (FEB de 2014) y FAR 52.227-19 (DIC de 2007).

Los datos aquí contenidos pertenecen a un producto comercial o servicio comercial (como se define en FAR 2.101) y son propiedad de NetApp, Inc. Todos los datos técnicos y el software informático de NetApp que se proporcionan en este Acuerdo tienen una naturaleza comercial y se han desarrollado exclusivamente con fondos privados. El Gobierno de EE. UU. tiene una licencia limitada, irrevocable, no exclusiva, no transferible, no sublicenciable y de alcance mundial para utilizar los Datos en relación con el contrato del Gobierno de los Estados Unidos bajo el cual se proporcionaron los Datos. Excepto que aquí se disponga lo contrario, los Datos no se pueden utilizar, desvelar, reproducir, modificar, interpretar o mostrar sin la previa aprobación por escrito de NetApp, Inc. Los derechos de licencia del Gobierno de los Estados Unidos de América y su Departamento de Defensa se limitan a los derechos identificados en la cláusula 252.227-7015(b) de la sección DFARS (FEB de 2014).

**Información de la marca comercial**

NETAPP, el logotipo de NETAPP y las marcas que constan en http://www.netapp.com/TM son marcas comerciales de NetApp, Inc. El resto de nombres de empresa y de producto pueden ser marcas comerciales de sus respectivos propietarios.