دستورالعمل و راهنمای استفاده از امکانات پردازش سریع Compute Canada

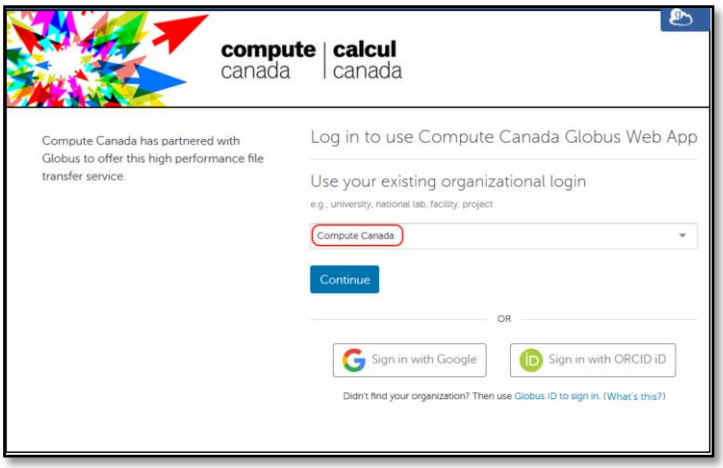
کلیه مطالب این نوشتار با توجه به توضیحات موجود در صفحات Compute Canada Wiki آماده شده است. لطفا برای کسب اطلاعات بیشتر به آدرس زیر مراجعه نمایید:

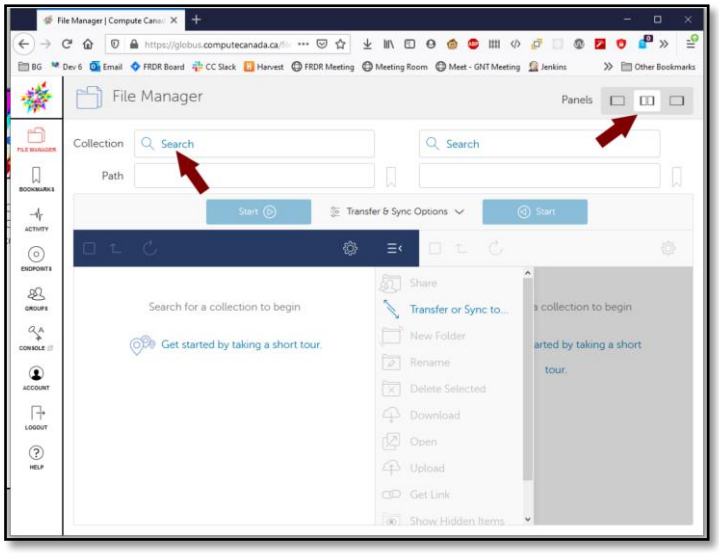
CC Doc (computecanada.ca)

برای بهره‌برداری از خدمات پردازش سریع Compute Canada لازم است به آدرس Compute Canada Please sign in وارد شده و با استفاده از گزینه‌ی Register برای ایجاد یک حساب کاربری اقدام نمایید. در مراحل ایجاد حساب کاربری از شما درخواست می‌شود شناسه‌ای تحت عنوان Compute Canada Role Identifier (CCRI) را وارد نمایید. این شناسه باید با ugi-961-02 مقداردهی شود. پس از ایجاد حساب کاربری، ایمیلی جهت فعال‌سازی حساب به آدرس ایمیل ثبت شده ارسال می‌شود. پس از استفاده از لینک فعال‌سازی موجود در این ایمیل، حساب کاربری در انتظار تایید استاد راهنمای شما (آقای دکتر رمضی) قرار می‌گیرد. بنابراین پس از انجام مراحل ایجاد حساب کاربری، مورد را جهت تایید به ایشان اطلاع دهید.

خدمات پردازش سریع Compute Canada شامل سه بستر به نام‌های Cedar، Graham و Niagara است. برای دسترسی به این بسترها می‌توان از طریق سیستم‌عامل‌های مختلف از جمله **ویندوز و لینوکس** اقدام نمود. **در ادامه روش استفاده از این بسترها در سیستم عامل لینوکس توضیح داده می‌شود.**

برای ارتباط با بستر پردازش سریع ابتدا لازم است از طریق یک نرم‌افزار مدیریت فایل، کلیه‌ی داده‌ها و برنامه‌های مورد نظر به فضای تخصیص یافته به حساب کاربری شما روی بستر پردازش سریع انتقال داده شود و پس از آن از طریق یک نرم‌افزار خط فرمان دستورات اجرایی صادر شوند. راهنمای Compute Canada نرم‌افزار مدیریت فایل **Globus** را پیشنهاد می‌دهد که از طریق آن می‌توانید به انتقال فایل از سیستم شخصی خود به یک سرور از بستر پردازش سریع یا بالعکس و همچنین انتقال فایل بین دو سرور از بسترهای پردازش سریع بپردازید. برای استفاده از این نرم‌افزار مدیریت فایل لازم است ابتدا از طریق حساب کاربری خود در Compute Canada وارد حساب کاربری در Globus شوید و سپس این نرم‌افزار را روی سیستم خود نصب کنید. محیط Globus و نحوه‌ی ورود به آن در شکل‌های زیر قابل مشاهده‌اند:

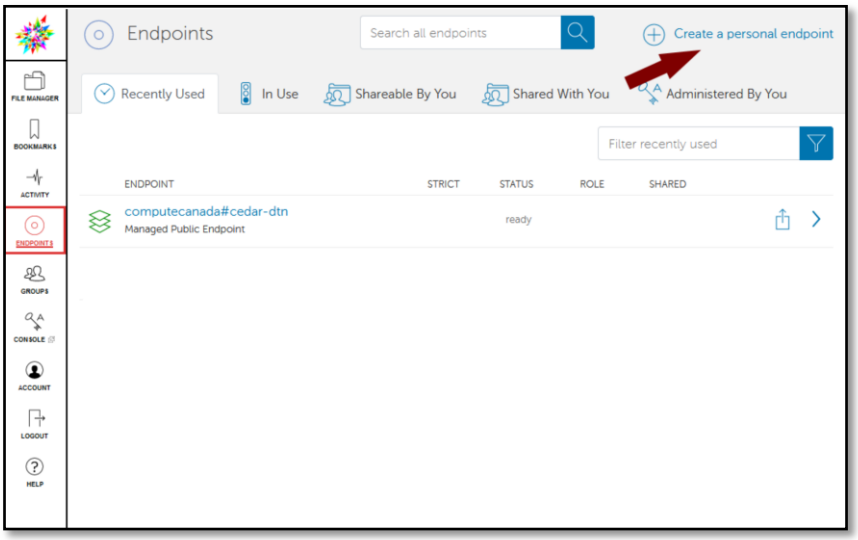




در این صفحه ابتدا باید از طریق گزینه‌ی Collection گزینه‌ی مورد نظر خود را روی بستر پردازش سریع انتخاب کنید. آدرس متناظر با هر بستر در صفحه Wiki مرتبط با نام آن بستر موجود است. برای مثال در مورد Cedar این اطلاعات در عنوان Globus Endpoint در جدول زیر قابل مشاهده است:

Availability: Compute RAC2017 allocations started June 30, 2017
Login node: **cedar.computecanada.ca**
Globus endpoint: **computecanada#cedar-dtn**
System Status Page: **http://status.computecanada.ca/**

برای آنکه Globus به سیستم شخصی کاربر دسترسی داشته باشد باید از طریق گزینه‌ی Create a personal endpoint در صفحه زیر نسخه‌ی مناسب با سیستم عامل مورد استفاده‌ی کاربر را نصب کرد.



اگر با نرم‌افزار مدیریت فایل FileZilla آشنایی دارید به راحتی می‌توانید از این نرم‌افزار به جای نرم‌افزار Globus استفاده کنید. برای این کار آدرس موجود در گزینه‌ی Data transfer node را به عنوان آدرس سرور به همراه اطلاعات

حساب کاربری Compute Canada و شماره پورت ۲۲ در نرم‌افزار FileZilla وارد کنید و به فضای کاربری جهت تبادل فایل‌ها دسترسی یابید.

Availability: In production since June 2017
Login node: **graham.computecanada.ca**
Globus endpoint: **computecanada#graham-dtn**
Data transfer node (rsync, scp, sftp,...): **gra-dtn1.computecanada.ca**

هر کدام از بسترهای پردازش سریع برای ذخیره‌سازی داده‌ها و برنامه‌ها، سه فضای مجزا با فایل سیستم‌های مختلف در اختیار قرار می‌دهد که از لحاظ امکانات ذخیره‌سازی، طول عمر نگهداری اطلاعات و سطح دسترسی با یکدیگر متفاوت هستند. به عنوان نمونه فضای ذخیره‌سازی بستر Graham به صورت زیر است:

Home space 64TB total volume	• Location of home directories. • Each home directory has a small, fixed quota. • Not allocated via RAS or RAC. Larger requests go to Project space. • Has daily backup.
Scratch space 3.6PB total volume Parallel high-performance filesystem	• For active or temporary (`/scratch`) storage. • Not allocated. • Large fixed quota per user. • Inactive data will be purged.
Project space 16PB total volume External persistent storage	• Allocated via RAS or RAC. • Not designed for parallel I/O workloads. Use Scratch space instead. • Large adjustable quota per project. • Has daily backup.

Filesystem	Default Quota	Lustre-based?	Backed up?	Purged?	Available by Default?	Mounted on Compute Nodes?
Home Space	50 GB and 500K files per user[1]	Yes	Yes	No	Yes	Yes
Scratch Space	20 TB and 1M files per user	Yes	No	Files older than 60 days are purged.[2]	Yes	Yes
Project Space	1 TB and 500K files per group[3]	Yes	Yes	No	Yes	Yes
Nearline Space	2 TB and 5000 files per group	Yes	Yes	No	Yes	No

1. ↑ This quota is fixed and cannot be changed.
2. ↑ See Scratch purging policy for more information.
3. ↑ Project space can be increased to 10 TB per group by a RAS request. The group's sponsoring PI should write to technical support to make the request.

لازم به ذکر است که ذخیره‌سازی اطلاعات در پوشه‌ی Scratch از نظر طول عمر محدودیت زمانی دارد و اگر داده‌ها برای مدتی غیرفعال باشند از روی این پوشه حذف می‌شوند. بنابراین بهتر است از این پوشه برای ذخیره‌سازی فایل‌های موقت برای مثال فایل‌های Checkpoint یا فایل‌های خروجی حاصل از اجرای یک Job استفاده شود. **از سوی دیگر برای اجرای یک Job باید فرآیند ارجاع Job از پوشه‌ای غیر از پوشه‌ی Home صورت پذیرد. برای مثال باید از طریق ترمینال ابتدا وارد پوشه‌ی Scratch شده و سپس فرمان اجرای یک Job صادر شود.** برای اطلاعات بیشتر به آدرس Storage and file management - CC Doc (computecanada.ca) مراجعه کنید.

پس از ذخیره‌سازی برنامه‌ها و داده‌ها برای اجرای کد برنامه بر اساس آنکه برنامه‌ی مورد نظر از چه نرم‌افزار یا زبان برنامه‌نویسی استفاده می‌کند، لازم است ابتدا ملزومات اجرای Job فراهم شود. در ادامه به **روند اجرای Job به زبان پایتون** پرداخته می‌شود.

برای اجرای یک Job به زبان پایتون، ابتدا باید محیط مناسب برای اجرا فراهم شود. بدین منظور در محیط ترمینال با استفاده از دستورات مناسب یک محیط مجازی با نسخه‌ی پایتون مورد نظر در یکی از پوشه‌های موجود برای مثال پوشه‌ی Home از فضای کاربری ایجاد می‌شود. توجه به این مسئله لازم است که هر کدام از کتابخانه‌هایی که در پایتون مورد استفاده هستند نسخه‌های متنوعی دارند و گاه نیاز است نسخه‌های مشخصی از هر کتابخانه روی محیط مجازی نصب شوند تا برای عملکرد مناسب محیط با یکدیگر سازگار باشند. از طرفی بستر پردازش سریع برای آنکه بهترین هماهنگی را روی سرورهای خود ایجاد نماید، برای هر کدام از این کتابخانه‌ها فایل‌های نصب سازگار خود را آماده کرده و در اختیار قرار داده است. توصیه‌ی پشتیبانی این بسترها این است که تا حد امکان از همین Wheelهای آماده استفاده شود و از دانلود فایل‌های نصب خودداری شود.

نکته‌ی دیگر اینکه ممکن است برای هر کتابخانه روی ماژول‌های مختلف مجموعه‌ای از نسخ متفاوت در دسترس باشد. بنابراین می‌توان با تغییر ماژول‌هایی که در محیط کار بارگذاری می‌شوند به نسخه‌ی مورد نظر از آن کتابخانه دسترسی پیدا کرد. دقت شود که در نهایت باید یک ماژول مشخص را انتخاب و پیش از ساخت محیط آن را در محیط ترمینال بارگذاری کنید و سپس نسخه‌ی پایتون مورد نظر خود را بارگذاری کرده و محیط مجازی را با مجموعه‌ای از کتابخانه‌های مورد نظر ایجاد نمایید. برای مشاهده‌ی نسخه‌های در دسترس روی هر بستر کافیست در محیط ترمینال از دستور زیر استفاده شود:

```
[name@server ~]$ module avail python
```

این دستور تمام ورژن‌های موجود پایتون را روی ماژول‌هایی که در حال حاضر بارگذاری شده‌اند نشان می‌دهد. برای آنکه لیست تمام ماژول‌هایی که در حال حاضر بارگیری شده‌اند قابل مشاهده باشد از دستور زیر استفاده کنید:

```
[name@server ~]$ module list
```

```
[name@server ~]$ module load python/3.6
```

برای بارگذاری یک ماژول برای مثال نسخه‌ی مشخصی از پایتون از دستور فوق استفاده می شود. در این نمونه، ورژن موجود از پایتون 3.6 برای مثال ورژن 3.6.3 روی محیط بارگیری می‌شود. حال اگر ماژول StdEnv 2020 روی محیط بارگیری شده باشد ورژن 3.6.10 از پایتون 3.6 روی محیط بارگذاری می‌شود و اگر ماژول StdEnv 2018 روی محیط فعال باشد، ورژن 3.6.3 از پایتون 3.6 فعال می‌گردد.

بارگذاری ماژول‌های مناسب StdEnv، Cuda و Cudnn برای کار با GPU نودهای موجود در بستر پردازش سریع الزامی هستند و همه آنها از طریق دستور module load قابل بارگیری هستند.

برای مثال اگر بخواهید محیطی برای کار با GPU از طریق ورژن‌های هماهنگ Cuda و Cudnn در پوشه‌ی Home ایجاد کنید لازم است دستورات زیر به ترتیب اجرا شوند:

```
[name@server ~]$ module load StdEnv/2020 cuda cudnn
[name@server ~]$ module load python/3.6
```

پس از اجرای این دستورات ماژول‌های StdEnv 2020 ، Cuda 11.0، Cudnn 8.0.3 و Python 3.6.10 فعال می‌شوند.

در اجرای دستورات زیر محیطی در پوشه‌ی Home به نام venv ایجاد و فعال می‌شود. دستور سوم pip را به آخرین ورژن موجود ارتقا می‌دهد. گزینه‌های no-download-- و no-index-- برای استفاده از Wheelهای موجود در بستر پردازش سریع استفاده می‌شوند و اگر این گزینه‌ها از دستور حذف شوند و Wheel مورد نظر در بستر پردازش سریع موجود نباشد از PyPI برای نصب کتابخانه استفاده می‌شود. (تضمینی بر کارکرد درست کتابخانه‌هایی که بدون این گزینه‌ها روی محیط نصب می‌شود وجود ندارد.)

```
[name@server ~]$ virtualenv --no-download ~/venv
[name@server ~]$ source ~/venv/bin/activate
(venv)[name@server ~]$ pip install --no-index --upgrade pip
```

برای غیر فعال کردن یک محیط مجازی از دستور زیر استفاده می‌شود:

```
(venv) [name@server ~] deactivate
```

برای نصب هر کدام از کتابخانه‌ها پس از فعال‌سازی محیط مجازی می‌توان از دستور pip استفاده کرد. دستور زیر به عنوان نمونه، آخرین نسخه‌ی موجود در بستر را متناظر با ماژول‌های بارگیری شده برای کتابخانه numpy روی محیط نصب می‌کند.

```
(venv) [name@server ~] pip install numpy --no-index
```

برای مشاهده Wheelهای موجود به آدرس **Available wheels** مراجعه نمایید. همچنین می‌توانید از دستورات زیر استفاده کنید و ورژن‌های موجود را از طریق ترمینال مشاهده نمایید:

```
[name@server ~]$ avail_wheels --name "*cdf*" --all_version
```

دستور فوق تمام ورژن‌های موجود از کتابخانه‌هایی را نمایش می‌دهد که عنوان آنها شامل واژه‌ی cdf است.
اگر گزینه‌ی all_version-- از دستور فوق حذف شود، بالاترین ورژن کتابخانه‌ها نمایش داده می‌شود.
اگر ورژن خاصی از یک کتابخانه مورد نظر است می‌توان به جای گزینه‌ی all_version-- از گزینه‌ی version-- استفاده کرد.

```
[name@server ~]$ avail_wheels --name "*cdf*" --version 1.3
```

برای نمونه دستور فوق تمام ورژن‌های 1.3 از کتابخانه‌هایی را لیست می‌کند که عنوان آنها شامل واژه‌ی cfd است.
اگر هدف مشاهده Wheelهای موجود برای نسخه‌ی مشخصی از پایتون باشد از دستور به صورت زیر استفاده می‌شود:

```
[name@server ~]$ avail_wheels --name "*cdf*" --python 3.6
```

اگر لیست کتابخانه‌های مورد نیاز طولانی باشد و بخواهید با یک دستور تمام این لیست را یکجا برای نصب معرفی کنید می‌توانید این لیست را در یک فایل txt. ذخیره نموده و با استفاده از دستور زیر آنها را نصب کنید:

```
pip install --no-index -r requirements.txt
```

نمونه‌ای از فایل requirement.txt در زیر قابل مشاهده است.

File : requirements.txt

```
absl-py==0.5.0
astor==0.7.1
gast==0.2.0
grpcio==1.17.1
h5py==2.8.0
Keras-Applications==1.0.6
Keras-Preprocessing==1.0.5
Markdown==2.6.11
numpy==1.16.0
protobuf==3.6.1
six==1.12.0
tensorboard==1.12.2
tensorflow-gpu==1.12.0+computecanada
termcolor==1.1.0
Werkzeug==0.14.1
```

در صورتی که نیاز باشد کتابخانه‌ی خاصی روی محیط نصب شود که جزء Wheelهای آماده در بستر پردازش سریع نیست باید ابتدا فایل whl. را از صفحه‌ی PyPI · The Python Package Index دانلود کرده و در فضای مشخصی از بستر قرار دهید و سپس با استفاده از دستور زیر آن را روی محیط نصب کنید. ترجیح این است که عنوان بسته‌ی دانلود شده به `-none-any.whl` خـتم شـود. اگـر ایـن عنـوان بـا مـوارد دیگـری بـرای مثـال `linux_x86_64` یا `manylinux*_x86_64` پایان یافته باشد، تضمینی برای عملکرد صحیح نسخه‌ی نصب شده روی بستر پردازش سریع وجود ندارد.

```
pip install tensorboardX-1.9-py2.py3-none-any.whl
```

مثال فوق برای نصب کتابخانه‌ی tensorboardX از فایل از پیش دانلود شده استفاده می‌کند. تاکید می‌شود که باید از این روش به عنوان آخرین راهکار استفاده شود.

برای اجرای یک برنامه روی بستر پردازش سریع لازم است یک فایل حاوی اطلاعات Job ایجاد شود و مشخصات نود محل اجرای برنامه از نظر تعداد هسته‌های CPU و تعداد GPU، میزان حافظه‌ی مورد نیاز برای اجرای کد و مواردی از این دست در این فایل مشخص شود. برای این منظور یک فایل با پسوند sh. ایجاد کرده و اطلاعات مورد نظر به صورت زیر در آن وارد می‌شود. برای اطلاعات بیشتر و مشاهده انواع تنظیمات برای درخواست‌های مختلف روی بستر Graham به Graham - CC Doc (computecanada.ca) مراجعه کنید.

توضیحات	دستورات
	`#!/bin/bash`

اطلاعات اختیاری جهت دریافت ایمیل در زمان شروع به کار، پایان کار، وقوع خطا در زمان اجرا، ورود مجدد به صف و غیره	`#SBATCH --mail-user=you@some.email.address`
	`#SBATCH --mail-type=BEGIN`
	`#SBATCH --mail-type=END`
	`#SBATCH --mail-type=FAIL`
	`#SBATCH --mail-type=REQUEUE`
	`#SBATCH --mail-type=ALL`

نامی که به Job به صورت اختیاری انتساب داده می‌شود.	`#SBATCH --job-name= jobname`
این گزینه باید حتما به صورت زیر تنظیم شود. **`#SBATCH --account=def-ramazi`**	`#SBATCH --account=def-someuser`

اگر از این گزینه استفاده شود Job یک نود را با تمام منابعش برای خود رزرو می‌کند و نود تخصیص یافته را با سایر Jobها شریک نمی‌شود. حذف این گزینه به برنامه‌ریز بستر پردازش سریع این امکان را می‌دهد که در صورت وجود منابع Jobهای مختلف را روی یک نود اجرا کند. در صورتی می‌توان از این امکان استفاده کرد که برنامه مورد نظر بتواند به خوبی از امکانات یک نود استفاده کند. در غیر این صورت توصیه می‌شود از این عمل خودداری شود.	`#SBATCH --nodes=1`
انتخاب نوع و تعداد GPU در این نمونه یک GPU از نوع Volta 100 درخواست شده است. اگر نوع GPU مشخص نشود به صورت پیش‌فرض از نوع Pascal 100 استفاده می‌شود.	`#SBATCH --gres=gpu:v100:1`
انتخاب تعداد CPU برای هر وظیفه لازم است تعداد CPUهای درخواستی نسبت به تعداد GPUهای انتخابی برای نودهایی که ۲۸ هسته‌ای هستند به مقدار 3.5 یا کمتر مقیاس شود. برای مثال بهتر است درخواست یک GPU با انتخاب حداکثر ۳ هسته CPU برای هر وظیفه همراه باشد.	`#SBATCH --cpus-per-task=3`
مقدار حافظه‌ی مورد نیاز در زمان اجرای برنامه اگر مقدار صفر تخصیص داده شود به معنای مقدار کل حافظه‌ی نود درخواستی است.	`#SBATCH --mem=12G`
`day-hour:minute:second` بیشترین زمان قابل قبول برای اجرای یک Job مدت ۲۸ روز است و بهتر است زمان تقریبی اجرای برنامه محاسبه شده و این گزینه کمی بیشتر از زمان مورد نیاز تنظیم شود. دقت شود که اگر زمان تنظیم شده کوتاه‌تر از مدت زمان اجرای کد باشد، برنامه‌ریز بستر پردازش سریع مقدار این پارامتر را اولویت دانسته و اجرای Job را خاتمه می‌دهد.	`#SBATCH --time=1-00:00:00`

نمونه‌ای از دستورات بارگذاری ماژول‌های مورد نیاز، فعال‌سازی محیط مجازی، ورود به پوشه‌ی حاوی کد برنامه و اجرای برنامه	`module load StdEnv/2020 Cuda Cudnn`
	`module load python/3.6`
	`source ~/venv/bin/activate`
	`cd ~/MyCodeFolder`
	`python myCode.py > output.txt`

پس از ایجاد فایل حاوی اطلاعات Job و انتقال آن به فضای کاربری، در صورت نیاز به تغییرات می‌توان از یک ویرایشگر ساده از طریق ترمینال استفاده نمود. برای مثال می‌توان این فایل‌ها را با دستور nano باز و ویرایش کرد. توجه به این نکته لازم است که فایل Job نباید حاوی کاراکترهای پنهان یا کاراکترهای پایان خط باشد. برای حذف این کاراکترها در فایل می‌توانید از دستور `dos2unix` استفاده کنید. برای اطلاعات بیشتر به Running jobs - CC Doc (computecanada.ca) مراجعه کنید.

برای صدور فرمان اجرای یک Job از طریق ترمینال در لینوکس ابتدا با استفاده از دستور زیر و وارد کردن رمز عبور حساب کاربری به فضای کاربری وارد شوید:

```
ssh -Y username@machine_name
```

به عنوان مثال برای ورود کاربر alamiyan به بستر Graham از دستور زیر استفاده می‌شود:

```
ssh -Y alamiyan@graham.computecanada.ca
```

توجه کنید که برای برقراری تعادل در حجم Loginها روی سرورهای مختلف ممکن است در هر بار ورود از طریق نود متفاوتی وارد شوید. برای اطلاعات بیشتر به SSH - CC Doc (computecanada.ca) مراجعه کنید.

فرض کنید فایل اجرای Job با عنوان run.sh در پوشه‌ی scratch موجود باشد، برای ارسال این Job جهت اجرا به Scheduler از روش زیر استفاده می‌شود:

```
[alamiyan@gra-login2 ~]$ cd scratch
[alamiyan@gra-login2 scratch]$ sbatch run.sh
```

پس از صدور فرمان اجرا، یک شناسه جهت پیگیری Job به آن اختصاص داده می‌شود.

برای مشاهده‌ی وضعیت Job در Scheduler می‌توان از دستور sq استفاده کرد. در خروجی این دستور مشخصات Job ارسالی به همراه وضعیت Job در Scheduler قابل مشاهده است. بسته به میزان امکاناتی که در درخواست اجرای یک Job مشخص شده است ممکن است Job برای مدتی در صف انتظار بماند تا منابع مورد نیاز آزاد و به Job اختصاص یابد. این زمان می‌تواند از چند ثانیه تا چند ساعت به طول بیانجامد.

```
[alamiyan@gra-login2 ~]$ sq
```

برای لغو اجرای یک Job از دستور زیر استفاده می‌شود:

```
[alamiyan@gra-login2 ~]$ scancel 48335486
```

در مثال فوق 48335486 شماره اختصاص یافته به Job است.

در روش sbatch تا زمانی که اجرای Job خاتمه نیافته باشد امکان دسترسی به Job و خروجی کنسول وجود ندارد اما اگر کد برنامه به صورت دوره‌ای نتایج را در فایل‌هایی از فضای حافظه ذخیره کند و Checkpointهایی در حین اجرا تولید شود، به این خروجی‌ها در مسیرهایی که در کد برنامه مشخص شده است بسته به زمان تولید خروجی دسترسی وجود دارد.

گاه ممکن است پیش از اجرای Job به روش sbatch نیاز باشد که کد برنامه اشکال‌زدایی و به اصطلاح Debug شود. برای این مورد تعدادی از نودهای بستر پردازش سریع به عنوان Interactive Node در اختیار کاربران قرار داده شده است تا برای مدتی آنها را درخواست کرده و از آنها استفاده نمایند. برای درخواست این نودها می‌توان از دستور زیر استفاده کرد. این دستور می‌تواند به همراه گزینه‌های مختلفی به‌کار گرفته شود تا امکانات مورد نیاز روی نود مقصد برای اجرای کد برنامه فراهم شود. این گزینه‌ها مشابه با گزینه‌هایی هستند که در فایل Job توضیح داده شد:

```
$ salloc --time=1:0:0 --ntasks=1 --account=def-someuser
salloc: Granted job allocation 1234567
$ ...              # do some work
$ exit             # terminate the allocation
salloc: Relinquishing job allocation 1234567
```

همان‌طور که در مثال فوق قابل مشاهده است پس از تخصیص منابع درخواستی به یک درخواست salloc نیـز شناسـه‌ای بـه درخواست تخصیص داده می‌شود. برای خروج از فضای تخصیص یافته پیش از پایان مدت زمان تخصیص یافته به Job می‌توان از دستور exit استفاده نمود. برای اطلاعات بیشتر به Running jobs - CC Doc (computecanada.ca) مراجعه کنید.