

# DSC 102: Systems for Scalable Analytics

## Programming Assignment 1

### 1 Introduction

In this assignment, you will be using Dask library to explore task parallelism on multiple cores on a cluster of machines. You will be performing feature exploration, data consistency checks, and computing several descriptive statistics about the data to build intuition for feature engineering for the next assignment.

### 2 Dataset Description

You are provided with the Amazon Reviews dataset with the *reviews* and *products* tables as CSV files. The schemas are provided in Table 1. The goal for the final assignment is to predict user's star rating. Thus, "overall" is our label.

(A) Column name	Column description	Example	(B) Column name	Column description	Example
reviewerID	ID of the reviewer	A32DT10X9WS4D0	asin	ID of the product	143561
asin	ID of the product	B003VX9DJM	salesRank	sales rank information	{'Movies & TV': 376041}
reviewerName	name of the reviewer	Slade	imUrl	url of the product image	http://g-ecx.images-amazon.com/31mC.jpg
helpful	helpfulness rating of the review	[0, 0]	categories	list of categories the product	[['Movies & TV', 'Movies']]
reviewText	text of the review	this was a gift for my friend who loves touch lamps.	title	name of the product	Everyday Italian (with Giada de Laurentiis)
overall	rating of the product	1	description	description of the product	3Pack DVD set - Italian Classics
summary	summary of the review	broken piece	price	price in US dollars	12.99
unixReviewTime	summary of the review	1397174400	related	related products (also bought, also viewed, bought together, buy after viewing)	{'also_viewed': ['B0036FO6SI', '000014357X'], 'buy_after_viewing': ['B0036FO6SI', 'B000KL8ODE']}
reviewTime	time of the review (raw)	04 11, 2014	brand	brand name	1

Table 1: (A) *Reviews* table and (B) *Products* table

### 3 Tasks

You will compute several descriptive statistics for both *reviews* and *products* table as follows:

- Q1. Get percentage of missing values for all columns in the *reviews* table and the *products* table
- Q2. Find pearson correlation value between the price and ratings
- Q3. Get mean, standard deviation, median, min, and max for the price column in the *products* table
- Q4. Get number of products for each super-category (the first entry in the "categories" column in the products table).
- Q5. Check (Return 1 or 0) if there are any dangling references to the product ids from the *reviews* table to *products* table. Return 1 if there are dangling references.

Q6. Check (1 or 0) if there is any dangling reference between product ids in the related column and “asin” of the *product* table. Return 1 if there are dangling references.

A code stub with function signature for this task has been provided to you. The input to the function is the *reviews* CSV file and the *products* CSV file. The output is a json file (saved as **results\_PA1.json**). The schema has been shared with you (**OutputSchema\_PA1.json**). You will write your answers to each sub-task as values. **\*\*Do not modify any keys of the json file\*\***. We will time the execution of the function PA1.

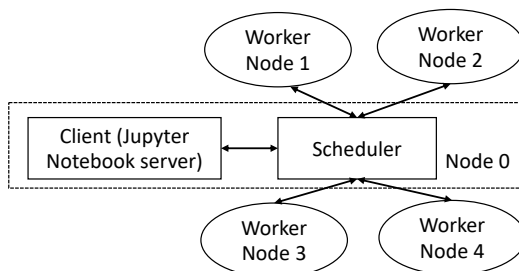
We have shared with you the “development” dataset and our accuracy results. Our code’s runtime on 4 nodes and 1 node are roughly 220s and 870s respectively. You can use this to validate your results and debug your code. The final evaluation will happen on separate held-out test sets. The runtime and the speedup numbers will be different for the held-out test set.

## 4 Deliverables

Submit your source code as <YOUR-TEAM-ID>.py on Canvas. Your source code must confirm to the function signatures provided to you. Make sure that your code is writing results to **results\_PA1.json**.

## 5 Setup

1. You will be launching 5 EC2 *Spot* instances, where you will create one instance for setting up the client (jupyter notebook server) and scheduler, and four instances for running workers.



2. Follow the below steps for launching EC2 spot instances.

a. Access your AWS account using single sign-on ID: <https://ets-apps.ucsd.edu/dsc102wi21-custom-aws/>. Credentials for CLI / API usage can be retrieved using a modified URL: <https://ets-apps.ucsd.edu/dsc102wi21-custom-aws?mode=env>.

b. We have setup the Dask environment on an AMI with name “dsc102-dask-distributed-environment-public”<sup>1</sup>. Go to “AMIs” (under “Images”) in your EC2 dashboard, select public images, and then search by name to find it. Select this AMI. See Figure 1 and Figure 2.

---

<sup>1</sup>The Dask version here is different from PA0’s

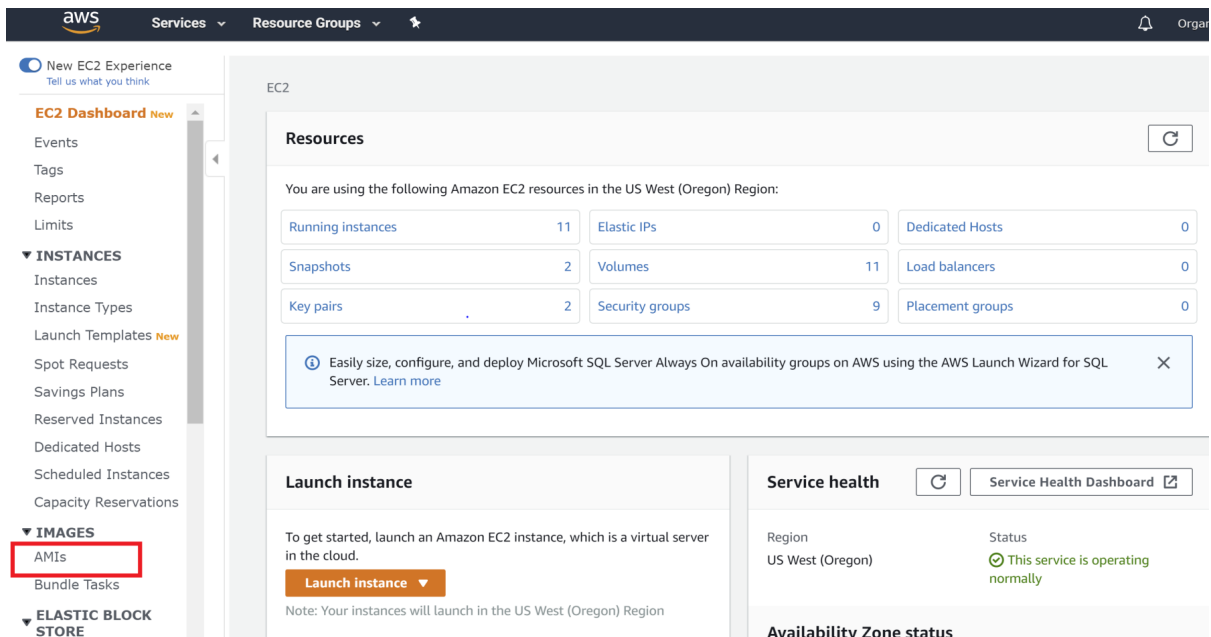


Figure 1

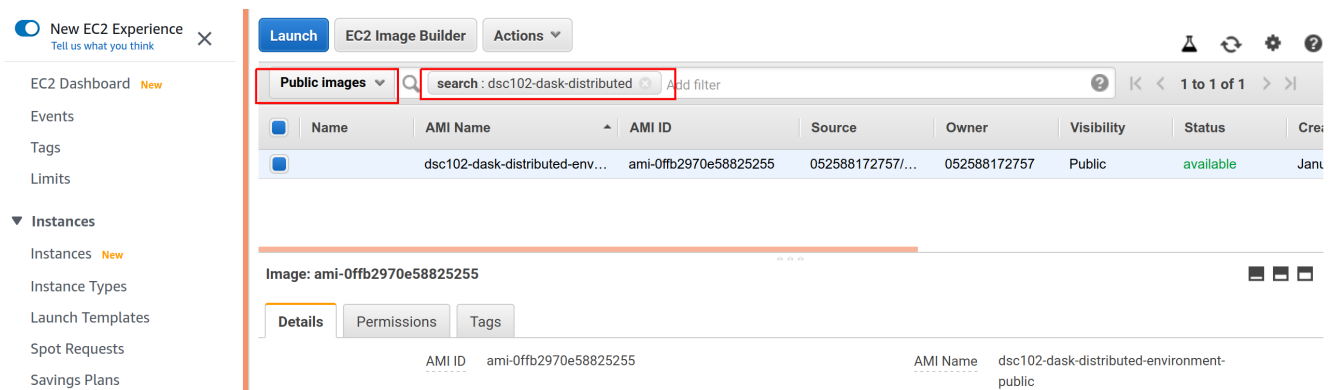


Figure 2

c. Now, you will be launching EC2 *Spot* instances with 100 GB storage. Follow the steps below.

Select the AMI and click Launch. In the next page, put **5** in the “Number of instances” box and put a check on “Request Spot instances” box. Put “maximum price” as 0.0557 (If you see a different “Current price”, then put whatever the current price is). Create a new security group. Retain other fields unchanged. Finally, after pressing the “Launch” button, add a key pair and download this locally. This will allow you to SSH into the instances. See Figure 3 to Figure 7. At the end, you should be able to see one instance in your dashboard.

Services

Resource Groups

OrganizationAccountAccessRol...

Oregon

Support

1. Choose AMI

2. Choose Instance Type

3. Configure Instance

4. Add Storage

5. Add Tags

6. Configure Security Group

7. Review

### Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Filter by:

All instance types

Current generation

Show/Hide Columns

Currently selected: t2.xlarge (Variable ECUs, 4 vCPUs, 2.3 GHz, Intel Broadwell E5-2686v4, 16 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GiB)	EBS-Optimized Available	Network Performance	IPv6 Support
<input type="checkbox"/>	General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	General purpose	t2.micro	1	1	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	General purpose	t2.small	1	2	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	General purpose	t2.large	2	8	EBS only	-	Low to Moderate	Yes
<input checked="" type="checkbox"/>	General purpose	t2.xlarge	4	16	EBS only	-	Moderate	Yes
<input type="checkbox"/>	General purpose	t2.2xlarge	8	32	EBS only	-	Moderate	Yes
<input type="checkbox"/>	General purpose	t3a.nano	2	0.5	EBS only	Yes	Up to 5 Gigabit	Yes

Cancel

Previous

Review and Launch

Next: Configure Instance Details

Figure 3

1. Choose AMI

2. Choose Instance Type

3. Configure Instance

4. Add Storage

5. Add Tags

6. Configure Security Group

7. Review

### Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances

5

Launch into Auto Scaling Group

You may want to consider launching these instances into an Auto Scaling Group to help you maintain application availability and for easy scaling in the future. [Learn how Auto Scaling can help your application stay healthy and cost effective](#).

Purchasing option

Request Spot instances

Current price

Availability Zone	Current price
us-west-2a	\$0.0557
us-west-2b	\$0.0557
us-west-2c	\$0.058

Maximum price

\$ 0.0557

Persistent request

Persistent request

Request valid to

Any time

Network

vpc-f58df88d (default)

Create new VPC

Subnet

No preference (default subnet in any Availability Zone)

Create new subnet

Auto-assign Public IP

Use subnet setting (Enable)

Placement group

Add instance to placement group

Capacity Reservation

Cancel

Previous

Review and Launch

Next: Add Storage

Figure 4

## Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more](#) about storage options in Amazon EC2.

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encrypted
Root	/dev/sda1	snap-086bdb64e02e07440	100	General Purpose SSD (gp2)	300 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypt

[Add New Volume](#)

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. [Learn more](#) about free usage tier eligibility and usage restrictions.

Figure 5

aws Services Resource Groups

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

### Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group: ☒ Create a new security group ☐ Select an existing security group

Security group name:

Description:

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	Custom 0.0.0.0/0	e.g. SSH for Admin Desktop

[Add Rule](#)

**Warning**

Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

[Cancel](#) [Previous](#) [Review and Launch](#)

Figure 6

aws Services Resource Groups

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

### Step 7: Review Instance Launch

Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.

**Improve your instances' security.** Your security group rules allow all traffic. Your instances may be accessible from any IP address. We recommend setting security group rules to allow access from known IP addresses only.

**Your instance configuration is not eligible for the free usage tier.** To launch an instance that's eligible for the free usage tier, check the instance type and root volume.

**AMI Details**

dsc102-dask-environment - ami-0808f706175e12be2

Root Device Type: ebs Virtualization type: hvm

**Instance Type**

Instance Type	ECUs	vCPUs	Memory (GB)
t2.xlarge	Variable	4	16

**Security Groups**

**Select an existing key pair or create a new key pair**

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about removing existing key pairs from a public AMI.

Create a new key pair

Key pair name:

[Download Key Pair](#)

You have to download the **private key file** (\*.pem file) before you can continue. Store it in a secure and accessible location. You will not be able to download the file again after it's created.

[Cancel](#) [Launch Instances](#)

[Cancel](#) [Previous](#) [Launch](#)

Figure 7

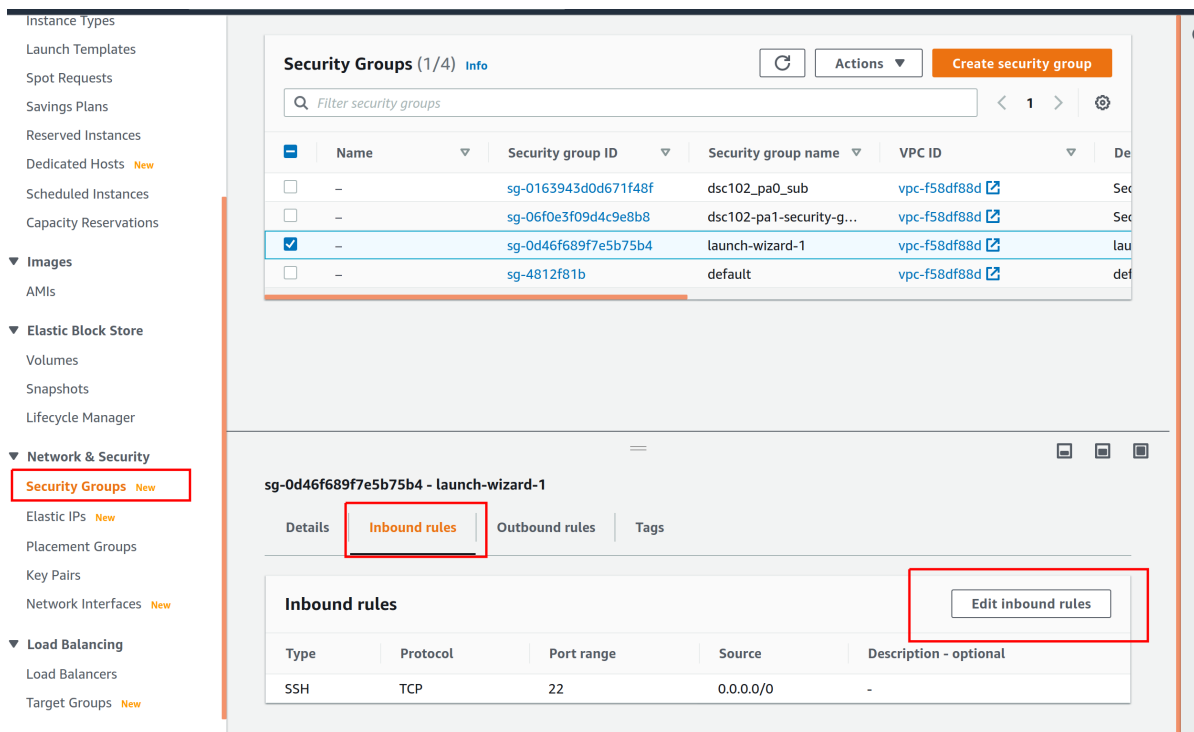


Figure 8

3. Once the EC2 instances are launched, go to the security group of the instances (under “Network & Security” in the left panel) and add two rules (under “Inbound”). (a) A rule with type “SSH”, and source as “0.0.0.0/0”. This rule will allow you to SSH to each of the machines. (b) A rule with type “All TCP”, and source as “Custom” with its group id. This rule will allow workers and scheduler to communicate with each other as in Figure 1.

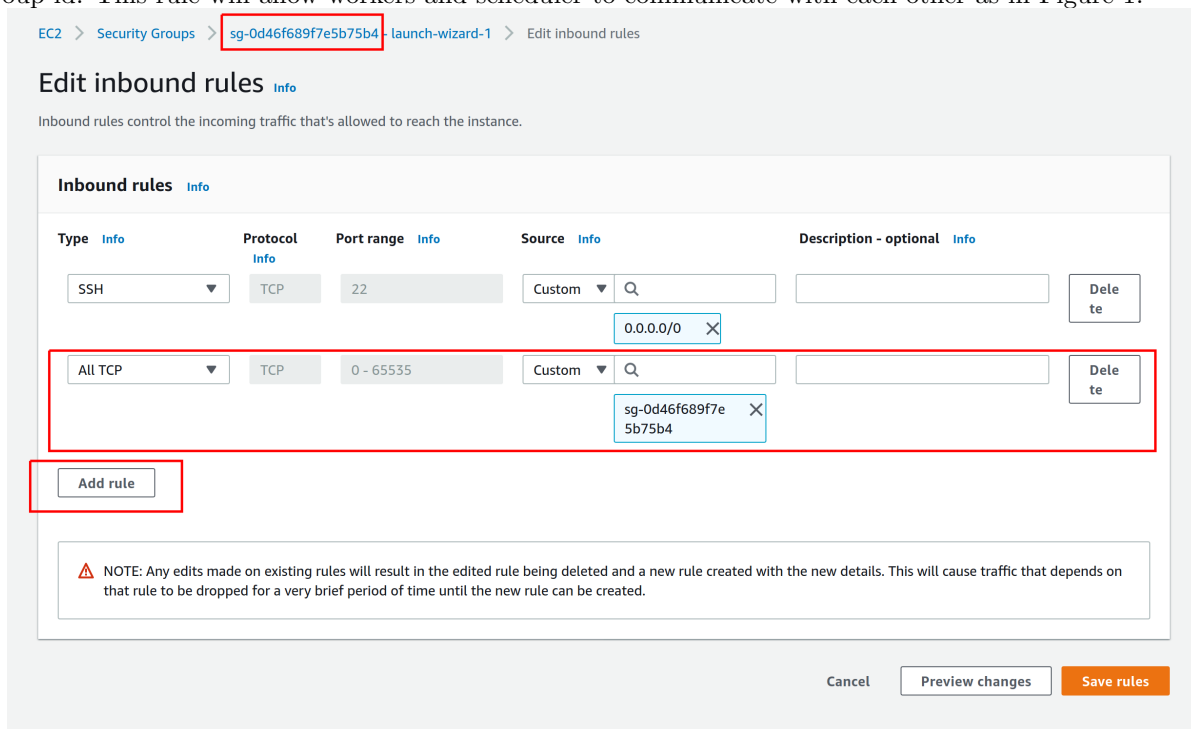


Figure 9

4. In this step, you will start the jupyter notebook server on the client, 1 scheduler process, and 4 worker processes.

- a. Change permission of the ssh keyfile to make sure your private key file isn't publicly viewable: `chmod 400 <keyfilename>.pem`. Linux and Mac users in particular will need the `chmod`.
- b. SSH into one of the nodes using command: `ssh -i "dask-key.pem" ubuntu@<ip-address-of-EC2-instance>`. Activate the dask environment with command: `source dask_env/bin/activate`. Start jupyter notebook server on one terminal with: `jupyter notebook --port=8888` and start the scheduler on another terminal with command: `dask-scheduler --host 0.0.0.0` This will run the scheduler on port 8786 and dashboard on port 8787. `<address-of-scheduler>` is given at `tcp://172.31.31.176:8786` in the Figure 10 below.

```

(dask_env) ubuntu@ip-172-31-31-176:~$
(dask_env) ubuntu@ip-172-31-31-176:~$
(dask_env) ubuntu@ip-172-31-31-176:~$ dask-scheduler --host 0.0.0.0
distributed.scheduler - INFO - -----
distributed.scheduler - INFO - To route to workers diagnostics web server please install jupyter-server-proxy: pip install jupyter-server-proxy
distributed.scheduler - INFO - Local Directory: /tmp/scheduler-0r61ig5q
distributed.scheduler - INFO - -----
distributed.scheduler - INFO - Clear task state
distributed.scheduler - INFO - Scheduler at: tcp://172.31.31.176:8786
distributed.scheduler - INFO - dashboard at: :8787

```

Figure 10

- c. Open a new terminal and SSH to jupyter notebook using: `ssh -i "dask-key.pem" ubuntu@<ip-address-of-EC2-instance> -L 8000:localhost:8888`. '-L' will port forward any connection to port 8000 on the local machine to port 8888 on `<ip-address-of-EC2-instance>`. Type in `jupyter notebook list` to get the token/password for the jupyter notebook. Open your browser and go to `localhost:8000` and paste the token. You can write your code here using jupyter notebook. To see dashboard on localhost port 8001 use command: `ssh -i "dask-key.pem" ubuntu@<ip-address-of-EC2-instance> -L 8001:localhost:8787`.
- d. SSH into other 4 nodes and activate the dask environment. Start workers with command: `dask-worker <address-of-scheduler>:8786 --nprocs 4`. `<address-of-scheduler>` was displayed with command `dask-scheduler --host 0.0.0.0`. `nprocs` option will make sure that all the cores of the machine are used. In the scheduler terminal screen, you will be able to see the connected workers.
5. The data and files are available from the s3 bucket (`s3://dsc102-public`). This contains the function signature (`PA1.py`), datasets (`user_reviews.csv` and `products.csv`), schema of expected output (`OutputSchema_PA1.json`), and the expected result on the development dataset (`results_PA1.json`). Make sure that data is available in the same path where scheduler and workers are running. Refer to step 5 given in the getting started guide of PA0 for more details.
6. Open the dashboard and click on "Workers" to double check if all workers are connected and you are now ready to code up.
7. Terminate EC2 instances once you are done. Remember when you terminate an EC2 instance you lose all the data, therefore we suggest you use a private GitHub repo to routinely push your work (code, logs and other smallfiles) and pull your repo whenever you create a new instance to resume your work.