PA1: AWS and Dask Setup

Rohit Ramaprasad

Contents

1 Introduction

This manual will cover how to setup multiple instances on AWS and start the Dask scheduler and workers for Programming Assignment 1.

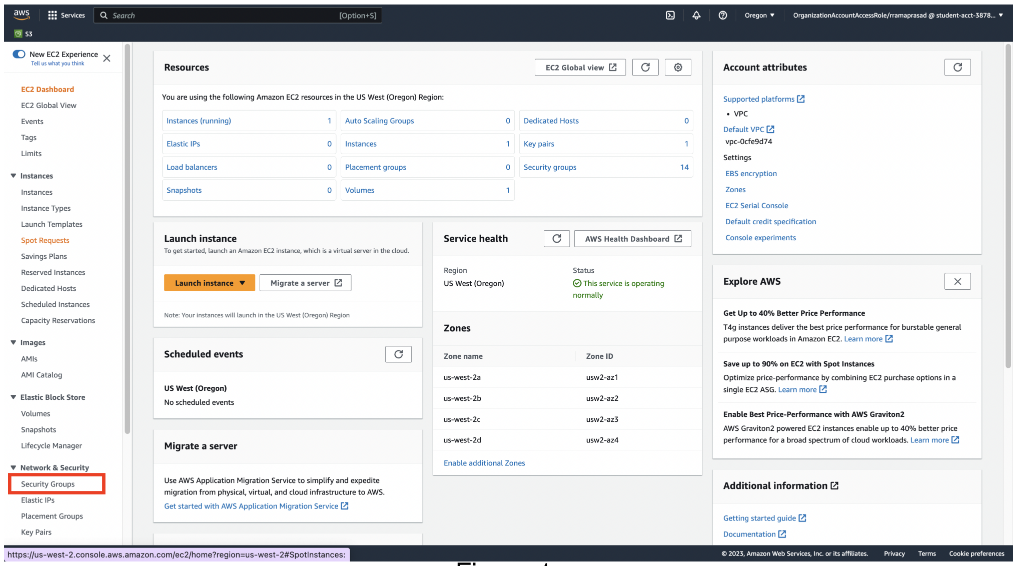
For this assignment, you will be running your code on both a single EC2 instance and also on a cluster of five EC2 instances to measure the *speedup*. There are points for how much speedup your code achieves when moving from 1 to 5 instances. Refer the grading rubric for more details.

2 AWS Setup

The AWS setup is very similar to PA0. For launching multiple instances, there are three differences. First, we will specify 5 instances instead of 1. One of these instances will run our Jupyter Notebook and Dask Scheduler, and the remaining 4 instances will run our Dask workers. Second, each of these instances will have 100GB SSD storage instead of 40GB. Third, we will create a new security group for our 5 instances that allow each of the instances to communicate with each other. Follow the below steps one by one.

a. Access your ETS account using single sign-on ID: `https://ets-apps.ucsd.edu/individual/DSC204A_SP23_A00/`. To open the AWS console click "Click here to access AWS" at the bottom of the page. To get your AWS credentials for CLI / API usage click "Generate API Keys (for CLI/scripting)".

b. Open AWS Dashboard. We will first create a new security group so that we can apply it to all of our instances later. Click on "Security Groups" on the left menu.



Resources

You are using the following Amazon EC2 resources in the US West (Oregon) Region:

Instances (running)	1	Auto Scaling Groups	0	Dedicated Hosts	0
Elastic IPs	0	Instances	1	Key pairs	1
Load balancers	0	Placement groups	0	Security groups	14
Snapshots	0	Volumes	1		

Account attributes

Supported platforms
- VPC

Default VPC
vpc-0cfe9d74

Settings
EBS encryption
Zones
EC2 Serial Console
Default credit specification
Console experiments

Launch instance
To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.

Note: Your instances will launch in the US West (Oregon) Region

Service health

Region: US West (Oregon)
Status: This service is operating normally

Zones

Zone name	Zone ID
us-west-2a	usw2-az1
us-west-2b	usw2-az2
us-west-2c	usw2-az3
us-west-2d	usw2-az4

Enable additional Zones

Scheduled events

US West (Oregon)
No scheduled events

Migrate a server

Use AWS Application Migration Service to simplify and expedite migration from physical, virtual, and cloud infrastructure to AWS.
Get started with AWS Application Migration Service

Explore AWS

Get Up to 40% Better Price Performance
T4g instances deliver the best price performance for burstable general purpose workloads in Amazon EC2. Learn more

Save up to 90% on EC2 with Spot Instances
Optimize price-performance by combining EC2 purchase options in a single EC2 ASG. Learn more

Enable Best Price-Performance with AWS Graviton2
AWS Graviton2 powered EC2 instances enable up to 40% better price performance for a broad spectrum of cloud workloads. Learn more

Additional information

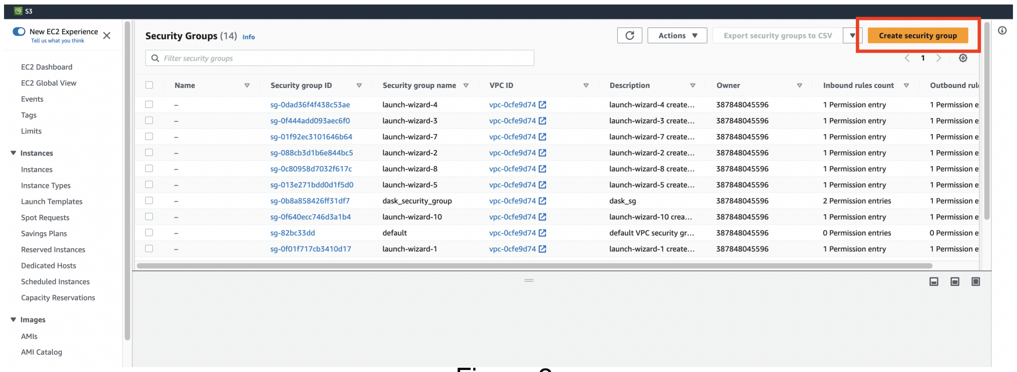
Getting started guide
Documentation

https://us-west-2.console.aws.amazon.com/ec2/home?region=us-west-2#SpotInstances:

© 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Figure 1:

c. Click on create security group.

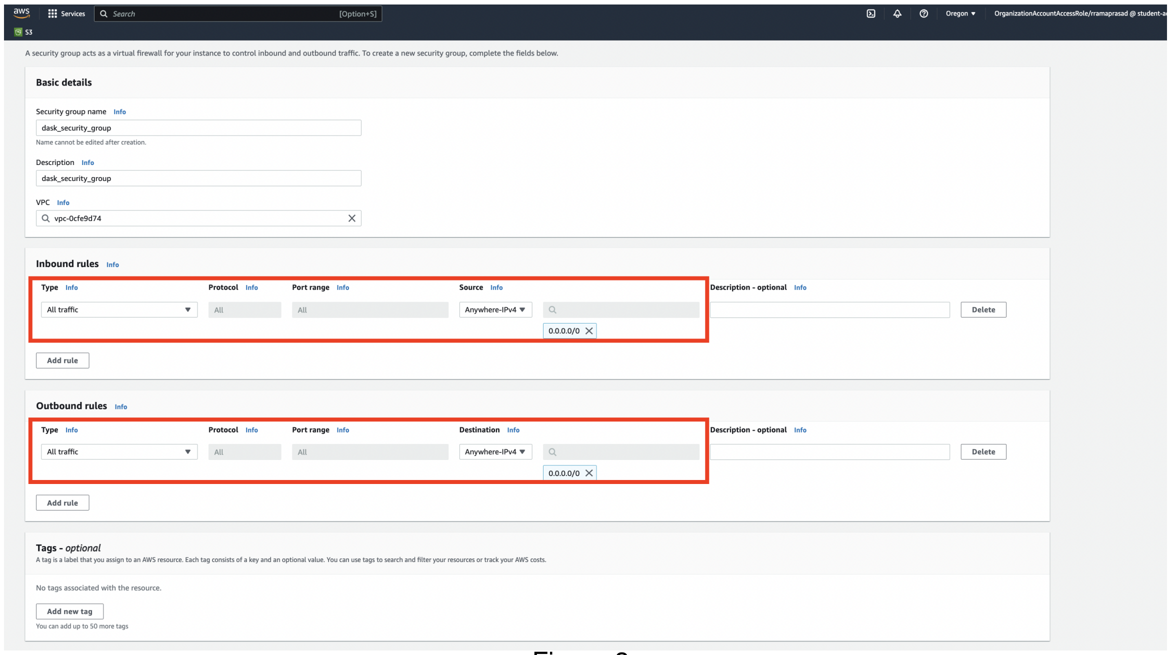


Security Groups (14) Info

Name	Security group ID	Security group name	VPC ID	Description	Owner	Inbound rules count	Outbound rul
–	sg-0dad36f4f438c53ae	launch-wizard-4	vpc-0cfe9d74	launch-wizard-4 create...	387848045596	1 Permission entry	1 Permission e
–	sg-0f444add093aec6f0	launch-wizard-3	vpc-0cfe9d74	launch-wizard-3 create...	387848045596	1 Permission entry	1 Permission e
–	sg-01f92ec3101646b64	launch-wizard-7	vpc-0cfe9d74	launch-wizard-7 create...	387848045596	1 Permission entry	1 Permission e
–	sg-088cb3d1b6e844bc5	launch-wizard-2	vpc-0cfe9d74	launch-wizard-2 create...	387848045596	1 Permission entry	1 Permission e
–	sg-0c80958d7032f617c	launch-wizard-8	vpc-0cfe9d74	launch-wizard-8 create...	387848045596	1 Permission entry	1 Permission e
–	sg-013e271bdd0d1f5d0	launch-wizard-5	vpc-0cfe9d74	launch-wizard-5 create...	387848045596	1 Permission entry	1 Permission e
–	sg-0b8a858426ff31df7	dask_security_group	vpc-0cfe9d74	dask_sg	387848045596	2 Permission entries	1 Permission e
–	sg-0f640ecc746d3a1b4	launch-wizard-10	vpc-0cfe9d74	launch-wizard-10 crea...	387848045596	1 Permission entry	1 Permission e
–	sg-82bc33dd	default	vpc-0cfe9d74	default VPC security gr...	387848045596	0 Permission entries	0 Permission e
–	sg-0f01f717cb3410d17	launch-wizard-1	vpc-0cfe9d74	launch-wizard-1 create...	387848045596	1 Permission entry	1 Permission e

Figure 2:

d. Choose any name and description for your security group. Change both the inbound and outbound rules to have "Type=All Traffic", "Source/Destination = Anywhere-IPv4". See image below.


A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

Basic details

Security group name: dask_security_group
Name cannot be edited after creation.

Description: dask_security_group

VPC: vpc-0cfe9d74

Inbound rules

Type	Protocol	Port range	Source		Description - optional
All traffic	All	All	Anywhere-IPv4	0.0.0.0/0	

Outbound rules

Type	Protocol	Port range	Destination		Description - optional
All traffic	All	All	Anywhere-IPv4	0.0.0.0/0	

Tags - *optional*
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

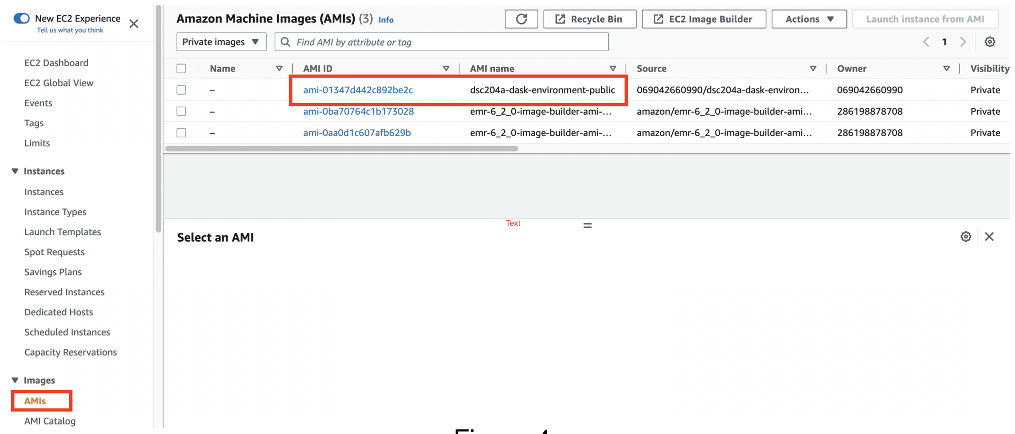
No tags associated with the resource.

You can add up to 50 more tags

Figure 3:

e. Lastly, click on "Create Security Group" at the bottom right.

f. Now, we will create our 5 instances which will use this new security group. We have setup the Dask environment on an AMI with the name “dsc204a-dask-environment-public.” Go to “AMIs” (under “Images”) in your EC2 dashboard, select private images, and then search by name to find it. Select this AMI. See Figure 4.


Amazon Machine Images (AMIs) (3)

Private images

Name	AMI ID	AMI name	Source	Owner	Visibility
–	ami-01347d442c892be2c	dsc204a-dask-environment-public	069042660990/dsc204a-dask-environ...	069042660990	Private
–	ami-0ba70764c1b173028	emr-6_2_0-image-builder-ami-...	amazon/emr-6_2_0-image-builder-ami...	286198878708	Private
–	ami-0aa0d1c607afb629b	emr-6_2_0-image-builder-ami-...	amazon/emr-6_2_0-image-builder-ami...	286198878708	Private

Select an AMI

Figure 4:

g. After selecting the AMI, click “Launch Instance from AMI" as shown below.

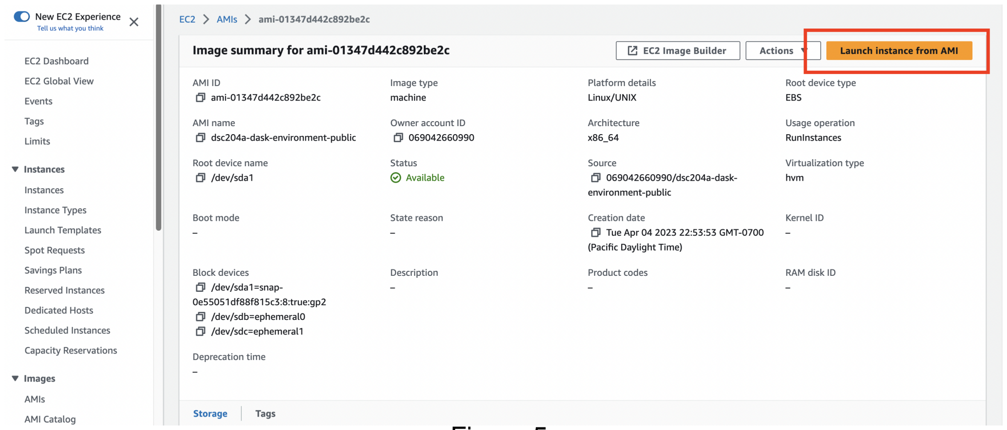


Image summary for ami-01347d442c892be2c

AMI ID ami-01347d442c892be2c	Image type machine	Platform details Linux/UNIX	Root device type EBS
AMI name dsc204a-dask-environment-public	Owner account ID 069042660990	Architecture x86_64	Usage operation RunInstances
Root device name /dev/sda1	Status Available	Source 069042660990/dsc204a-dask-environment-public	Virtualization type hvm
Boot mode –	State reason –	Creation date Tue Apr 04 2023 22:53:53 GMT-0700 (Pacific Daylight Time)	Kernel ID –
Block devices /dev/sda1=snap-0e55051df88f815c3:8:true:gp2 /dev/sdb=ephemeral0 /dev/sdc=ephemeral1	Description –	Product codes –	RAM disk ID –
Deprecation time –			

Figure 5:

h. Now, strictly follow the below instructions to launch the EC2 Spot instances.

a. Give any name for your instance.

b. Number of instances to launch is 5.

c. The instance type is “t2.xlarge”.

d. Create a new key pair for SSH’ing to your instance later. The private key will be downloaded to your local machine.

e. Under “Network Settings", click “Select Existing Security Group" and choose the name of the security group you just created, which in my case is “dask_security_group".

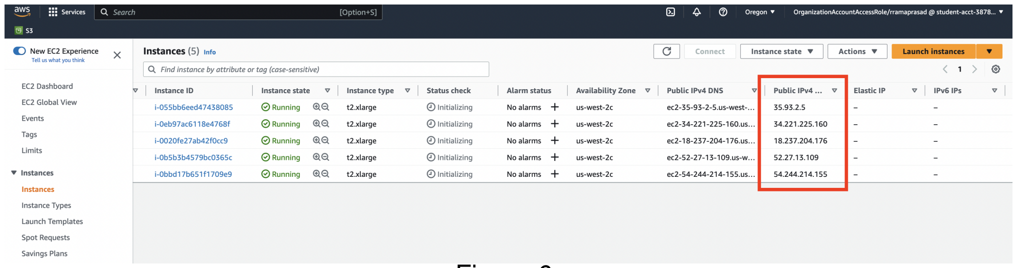
f. Choose 100GB SSD gp2 storage.

g. Open advanced details. Select “Request Spot Instances”. Then click on “customize” just on the right. Open the dropdown for “Request type” and select “One-time”.

h. Lastly, click “Launch Instance”.

3 Dask Setup

After launching 5 instances, your AWS dashboard must look like the figure below. You can view the IP addresses of the different instances inside the red box.



Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP	IPv6 IPs
i-055bb6eed47438085	Running	t2.xlarge	Initializing	No alarms	us-west-2c	ec2-35-93-2-5.us-west-...	35.93.2.5	–	–
i-0eb97ac6118e4768f	Running	t2.xlarge	Initializing	No alarms	us-west-2c	ec2-34-221-225-160.us...	34.221.225.160	–	–
i-0020fe27ab42f0cc9	Running	t2.xlarge	Initializing	No alarms	us-west-2c	ec2-18-237-204-176.us...	18.237.204.176	–	–
i-0b5b3b4579bc0365c	Running	t2.xlarge	Initializing	No alarms	us-west-2c	ec2-52-27-13-109.us-w...	52.27.13.109	–	–
i-0bbd17b651f1709e9	Running	t2.xlarge	Initializing	No alarms	us-west-2c	ec2-54-244-214-155.us...	54.244.214.155	–	–

Figure 6:

3.1 First Instance Setup

We will run Jupyter Notebook and the Dask scheduler on the first of our 5 EC2 instances. As you will see, we will SSH into this instance 5 times on 5 different terminals to execute different commands. If you are comfortable with tmux you can avoid creating these many terminals, but for sake of simplicity, I will avoid using tmux. The IP address of my first instance is `35.95.2.5` as seen in Figure 6. **Replace this IP with the IP of your first instance in the commands below.**

a. Terminal 1 - (Dataset transfer from S3 bucket into our EC2 SSD)

1. Open a terminal on your machine and SSH into the instance -
```
ssh -i "my-key.pem" ubuntu@35.95.2.5
```
2. Copy your AWS API Keys (from the link on ETS dashboard) and paste it in the terminal.
3. Start dataset transfer -
```
aws s3 sync s3://dsc204a-pa1 .
```

b. Terminal 2 - (Starting jupyter notebook)

1. Open a terminal on your machine and SSH into the instance -
```
ssh -i "my-key.pem" ubuntu@35.95.2.5
```
2. Activate the dask environment
```
source dask_env/bin/activate
```
3. Start jupyter notebook on port 8888 -
```
jupyter-notebook --port 8888
```

c. Terminal 3 - (Starting Dask Scheduler)

1. Open a terminal on your machine and SSH into the instance -
```
ssh -i "my-key.pem" ubuntu@35.95.2.5
```
2. Activate the dask environment
```
source dask_env/bin/activate
```
3. Start Dask scheduler
```
dask scheduler --host 0.0.0.0
```

d. Terminal 4 - (Port forward jupyter notebook)

1. Open a terminal on your machine and run the below command. This will forward the process on EC2's port 8888 (jupyter notebook) to your computer's port 8888.

`ssh -i "my_key.pem" ubuntu@35.95.2.5 -L 8888:localhost:8888`

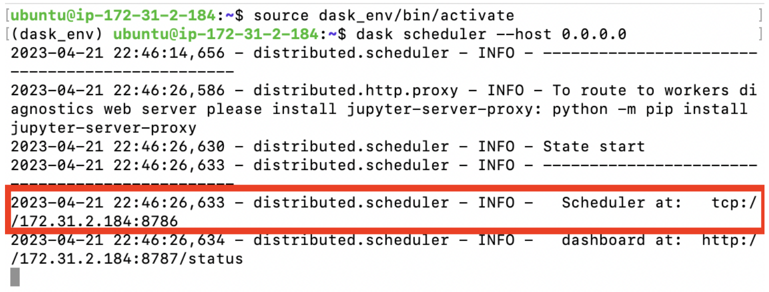
e. Terminal 5 - (Port forward dask dashboard)

1. Open a terminal on your machine and run the below command. This will forward the process on EC2's port 8787 (dask dashboard) to your computer's port 8787.

`ssh -i "my_key.pem" ubuntu@35.95.2.5 -L 8787:localhost:8787`

3.2 Remaining 4 Instances Setup

We will run our Dask workers on the remaining 4 instances. We need a way for the workers to communicate with the scheduler and this is achieved by passing in the IP address of the scheduler to each worker. This is done in terminal 2 below. We can find the address of the Dask scheduler from the output in terminal 3 of our first EC2 instance. In my case, the scheduler address is `tcp://172.31.2.184:8786`. See figure below.



```
ubuntu@ip-172-31-2-184:~$ source dask_env/bin/activate
(dask_env) ubuntu@ip-172-31-2-184:~$ dask scheduler --host 0.0.0.0
2023-04-21 22:46:14,656 - distributed.scheduler - INFO - -----------------------
------------------------
2023-04-21 22:46:26,586 - distributed.http.proxy - INFO - To route to workers di
agnostics web server please install jupyter-server-proxy: python -m pip install
jupyter-server-proxy
2023-04-21 22:46:26,630 - distributed.scheduler - INFO - State start
2023-04-21 22:46:26,633 - distributed.scheduler - INFO - -----------------------
------------------------
2023-04-21 22:46:26,633 - distributed.scheduler - INFO -   Scheduler at:   tcp:/
/172.31.2.184:8786
2023-04-21 22:46:26,634 - distributed.scheduler - INFO -   dashboard at:  http:/
/172.31.2.184:8787/status
```

Figure 7:

We will launch 2 terminals for each of our 4 EC2 instances, making a total of 8 terminals. In the first terminal we will run the command to download our dataset from the S3 bucket to the EC2 instance (similar as done for instance 1 above). In the second terminal we will start our Dask workers.

Repeat the below two steps for each of your 4 EC2 instances. The only difference is the IP address you will use to SSH. I have shown the steps for my second EC2 instance below.

a. Terminal 1 - (Dataset transfer from S3 bucket into our EC2 instance)

1. Open a terminal on your machine and SSH into the instance -
`ssh -i "my-key.pem" ubuntu@34.221.225.160`

2. Copy your AWS API Keys (from the link on ETS dashboard) and paste it in the terminal.

3. Start dataset transfer -
`aws s3 sync s3://dsc204a-pa1 .`

b. Terminal 2 - (Starting Dask Worker)

1. Open a terminal on your machine and SSH into the instance -
`ssh -i "my-key.pem" ubuntu@34.221.225.160`

2. Activate the dask environment
`source dask_env/bin/activate`

3. Start 4 Dask worker processes on this machine. Replace scheduler address with yours.
`dask worker tcp://172.31.2.184:8786 --nworkers 4`

3.3 Sanity Check

After completing the above steps, when you visit the Dask dashboard at `http://localhost:8787`, you should see 16 workers in total. See figure below.

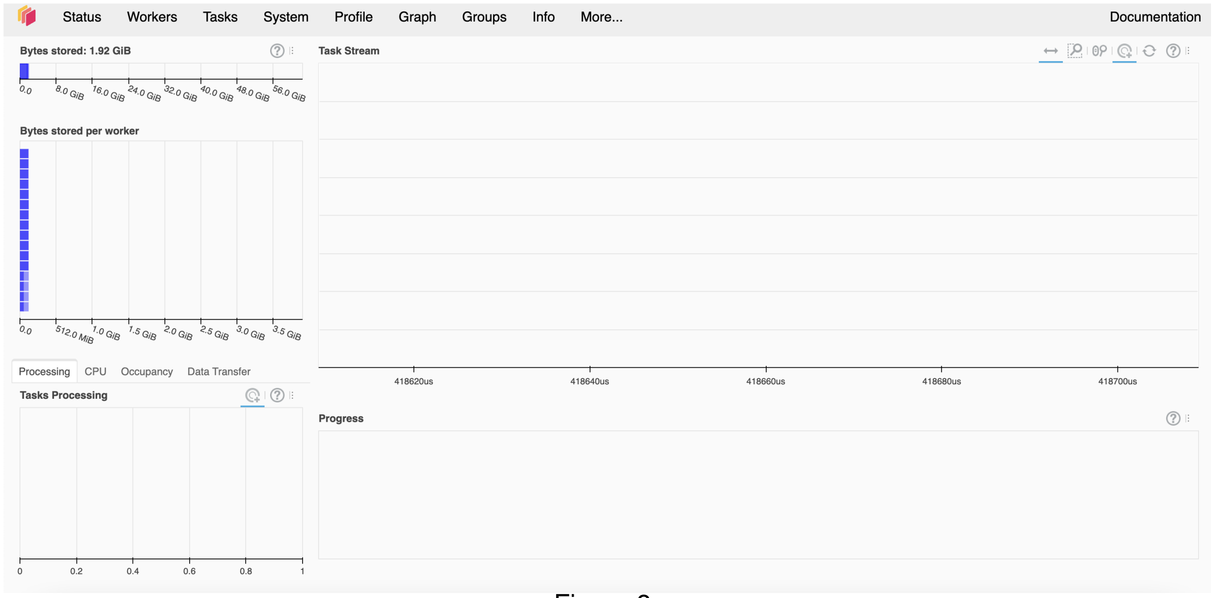
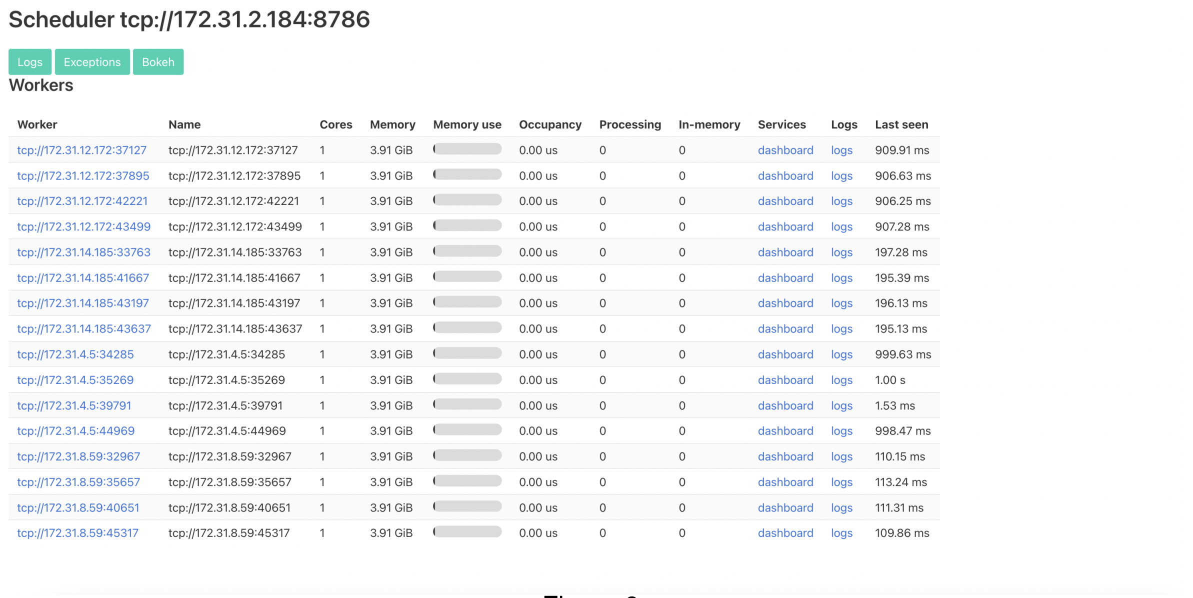


Figure 8:



Scheduler tcp://172.31.2.184:8786

Workers

Worker	Name	Cores	Memory	Memory use	Occupancy	Processing	In-memory	Services	Logs	Last seen
tcp://172.31.12.172:37127	tcp://172.31.12.172:37127	1	3.91 GiB		0.00 us	0	0	dashboard	logs	909.91 ms
tcp://172.31.12.172:37895	tcp://172.31.12.172:37895	1	3.91 GiB		0.00 us	0	0	dashboard	logs	906.63 ms
tcp://172.31.12.172:42221	tcp://172.31.12.172:42221	1	3.91 GiB		0.00 us	0	0	dashboard	logs	906.25 ms
tcp://172.31.12.172:43499	tcp://172.31.12.172:43499	1	3.91 GiB		0.00 us	0	0	dashboard	logs	907.28 ms
tcp://172.31.14.185:33763	tcp://172.31.14.185:33763	1	3.91 GiB		0.00 us	0	0	dashboard	logs	197.28 ms
tcp://172.31.14.185:41667	tcp://172.31.14.185:41667	1	3.91 GiB		0.00 us	0	0	dashboard	logs	195.39 ms
tcp://172.31.14.185:43197	tcp://172.31.14.185:43197	1	3.91 GiB		0.00 us	0	0	dashboard	logs	196.13 ms
tcp://172.31.14.185:43637	tcp://172.31.14.185:43637	1	3.91 GiB		0.00 us	0	0	dashboard	logs	195.13 ms
tcp://172.31.4.5:34285	tcp://172.31.4.5:34285	1	3.91 GiB		0.00 us	0	0	dashboard	logs	999.63 ms
tcp://172.31.4.5:35269	tcp://172.31.4.5:35269	1	3.91 GiB		0.00 us	0	0	dashboard	logs	1.00 s
tcp://172.31.4.5:39791	tcp://172.31.4.5:39791	1	3.91 GiB		0.00 us	0	0	dashboard	logs	1.53 ms
tcp://172.31.4.5:44969	tcp://172.31.4.5:44969	1	3.91 GiB		0.00 us	0	0	dashboard	logs	998.47 ms
tcp://172.31.8.59:32967	tcp://172.31.8.59:32967	1	3.91 GiB		0.00 us	0	0	dashboard	logs	110.15 ms
tcp://172.31.8.59:35657	tcp://172.31.8.59:35657	1	3.91 GiB		0.00 us	0	0	dashboard	logs	113.24 ms
tcp://172.31.8.59:40651	tcp://172.31.8.59:40651	1	3.91 GiB		0.00 us	0	0	dashboard	logs	111.31 ms
tcp://172.31.8.59:45317	tcp://172.31.8.59:45317	1	3.91 GiB		0.00 us	0	0	dashboard	logs	109.86 ms

Figure 9: