

# AWS training: Ray cluster deployment

EC2 setup (free tier)

Installation

Credentials

Launch Ray cluster

EFS

Setup

IAM Role and EC2 Instance Profile

Accessing S3

Amazon CloudWatch

Train ML model

## EC2 setup (free tier)

The screenshot displays the AWS Management Console interface. The left-hand navigation pane shows the 'Instances' section under 'EC2 Global View'. The main content area is divided into several sections: 'Resources' (listing various EC2 resources like Instances, Capacity Reservations, etc.), 'Launch instance' (a button to start a new instance), 'Service health' (AWS Health Dashboard), and 'Account attributes' (Default VPC). The 'Resources' section shows a table of resources, with 'Instances (running)' highlighted in red. The 'Launch instance' section is also visible. The 'Service health' section shows the 'AWS Health Dashboard' link. The 'Account attributes' section shows the 'Default VPC' link. The 'Instances' section at the bottom shows a table of instances, with the 'Launch Instances' button highlighted in red.

**Resources**

You are using the following Amazon EC2 resources in the Asia Pacific (Sydney) Region:

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

**Launch instance**

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

**Service health**

[AWS Health Dashboard](#)

**Account attributes**

[Default VPC](#)

**Instances**

Find Instance by attribute or tag (case-sensitive)

Instance state:  Clear filters

All states

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
No matching instances found						

**Select an instance**

▼ Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Q Search our full catalog including 1000s of application and OS images

Quick Start

Amazon Linux

macOS

Ubuntu

Windows

Red Hat

SUSE Linux

Debi

aws

Mac

ubuntu

Microsoft

Red Hat

SUSE

debi

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Amazon Linux 2023 AMI

Free tier eligible

▼ Summary

Number of instances Info

1

Software Image (AMI)

Amazon Linux 2023 AMI 2023.6.2...read more

ami-0d6560f3176dc9es0

Virtual server type (instance type)

t2.micro

Firewall (security group)

New security group

Storage (volumes)

1 x volume(s) 8 GiB

Cancel

Launch instance

Preview code

Identity and Access Management (IAM)

Q Search IAM

Dashboard

▼ Access management

User groups

Users

Roles

Policies

Identity providers

Account settings

Root access management New

Users (0) Info

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

Q Search

User name

Path

Groups

Last activity

MFA

Password age

Console

No resources to display

Create user

Step 1 Specify user details

Step 2 Set permissions

Step 3 Review and create

Step 4 Retrieve password

Specify user details

User details

User name

ray\_test

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and \* . , \_ @ \_ - (hyphen)

☒ Provide user access to the AWS Management Console - optional

If you're providing console access to a person, it's a best practice to manage their access in IAM Identity Center.

Are you providing console access to a person?

User type

☐ Specify a user in Identity Center - Recommended

We recommend that you use Identity Center to provide console access to a person. With Identity Center, you can centrally manage user access to their AWS accounts and cloud applications.

☒ I want to create an IAM user

We recommend that you create IAM users only if you need to enable programmatic access through access keys, service-specific credentials for AWS CodeCommit or Amazon Keyspaces, or a backup credential for your account access.

Identity and Access Management (IAM)

Q Search IAM

Dashboard

▼ Access management

User groups

Users

Roles

Policies

Identity providers

Account settings

Root access management New

▼ Access reports

Create user group

Name the group

User group name

Enter a meaningful name to identify this group.

ray\_test

Maximum 128 characters. Use alphanumeric and \* + = \_ . - ' characters.

Add users to the group - Optional (1/1) Info

An IAM user is an entity that you create in AWS to represent the person or application that uses it to interact with AWS.

Q Search

User name

Groups

Last activity

Creation time

☒ ray\_test

0

None

Now

Identity and Access Management (IAM)

ray\_test

Summary

ARN: [arn:aws:iam::794038232598:user/ray\\_test](#)

Created: January 03, 2025, 19:56 (UTC+13:00)

Console access: [Enabled without MFA](#)

Last console sign-in: [Never](#)

Access key 1: [Create access key](#)

Permissions policies (1)

Permissions are defined by policies attached to the user directly or through groups.

Filter by Type: All types

Identity and Access Management (IAM)

ray\_test

Summary

ARN: [arn:aws:iam::794038232598:user/ray\\_test](#)

Created: January 03, 2025, 19:56 (UTC+13:00)

Console access: [Enabled without MFA](#)

Last console sign-in: [Never](#)

Access key 1: [Never used. Created today.](#)

Access key 2: [Create access key](#)

Security credentials

Console sign-in

Console sign-in link: [https://794038232598.signin.aws.amazon.com/console](#)

Console password: Updated 5 minutes ago (2025-01-03 19:56 GMT+13)

## Installation

```
1 poetry add boto3 botocore ray[default]
```

## Credentials

```
1 # setup AWS credentials using environment variables
2 export AWS_ACCESS_KEY_ID=foo
3 export AWS_SECRET_ACCESS_KEY=bar
4 export AWS_SESSION_TOKEN=baz
5
6 # alternatively, you can setup AWS credentials using ~/.aws/credentials file
7 echo "[default]
8 aws_access_key_id=foo
9 aws_secret_access_key=bar
10 aws_session_token=baz" >> ~/.aws/credentials
```

## Launch Ray cluster

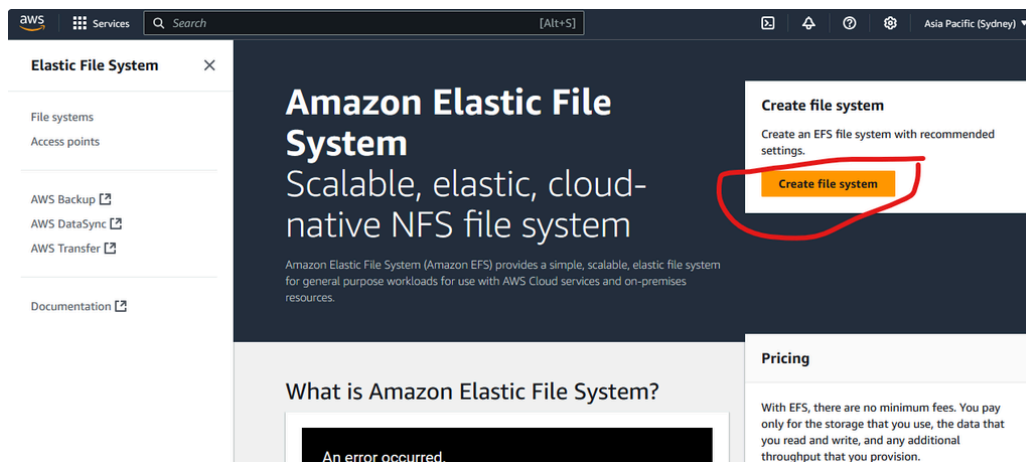
```
1 # Download the example-full.yaml
2 wget https://raw.githubusercontent.com/ray-project/ray/master/python/ray/autoscaler/aws/example-full.yaml
3
```

```

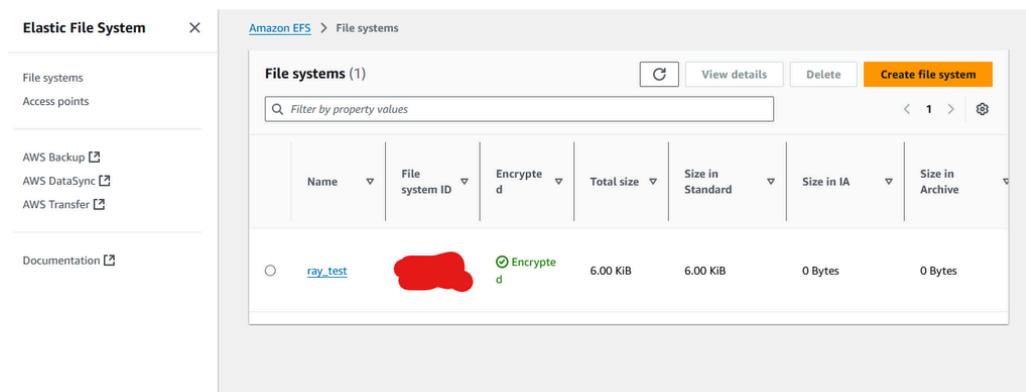
4 # Create or update the cluster. When the command finishes, it will print
5 # out the command that can be used to SSH into the cluster head node.
6 ray up example-full.yaml
7 # This will take a while...
8
9 #####
10
11 # Jump into the cluster
12 # Get a remote shell on the head node.
13 ray attach example-full.yaml
14
15 #####
16
17 pwd
18 ls
19
20 # Try running a Ray program.
21 python -c 'import ray; ray.init()'
22 exit
23
24 # Tear down the cluster.
25 ray down example-full.yaml

```

## EFS



The screenshot shows the AWS Elastic File System console. On the left is a navigation menu with options like 'File systems', 'Access points', 'AWS Backup', 'AWS DataSync', 'AWS Transfer', and 'Documentation'. The main content area has a header 'Amazon Elastic File System' with the tagline 'Scalable, elastic, cloud-native NFS file system'. Below this is a 'Create file system' button, which is highlighted with a red circle. To the right of the button is a 'Pricing' section. At the bottom, there is a black box with the text 'An error occurred.'



The screenshot shows the AWS Elastic File System console with a list of file systems. The left navigation menu is the same as in the previous screenshot. The main content area shows a table of file systems. The first file system is named 'ray\_test' and has a redacted file system ID. The table has columns for Name, File system ID, Encrypted, Total size, Size in Standard, Size in IA, and Size in Archive. The 'ray\_test' file system is 6.00 KiB in total size, 6.00 KiB in Standard size, and 0 Bytes in IA and Archive sizes. The 'Create file system' button is also visible in the top right of the table area.

Name	File system ID	Encrypted	Total size	Size in Standard	Size in IA	Size in Archive
ray_test	[REDACTED]	Encrypted	6.00 KiB	6.00 KiB	0 Bytes	0 Bytes

**Elastic File System** X

File systems  
Access points

AWS Backup [↗](#)  
AWS DataSync [↗](#)  
AWS Transfer [↗](#)

Documentation [↗](#)

Amazon EFS > File systems > fs-016466b1e6ef7fb10

**ray\_test (fs-016466b1e6ef7fb10)** Delete Attach

**General** Edit

Amazon resource name (ARN)  
arn:aws:elasticfilesystem:ap-southeast-2:794038232598:file-system/fs-016466b1e6ef7fb10

Performance mode  
General Purpose

Throughput mode  
Elastic

Lifecycle management  
Transition into Infrequent Access (IA): 30 day(s) since last access  
Transition into Archive: 90 day(s) since last access

Automatic backups  
Enabled

Encrypted  
893932a9-a360-4b33-9c54-09372ed1b3b3 (aws/elasticfilesystem)

File system state  
Available

DNS name  
fs-016466b1e6ef7fb10.efs.ap-southeast-2.amazonaws.com

Replication overwrite protection  
-

**Attach** X

Mount your Amazon EFS file system on a Linux instance. [Learn more](#)

☒ Mount via DNS ☐ Mount via IP

Using the EFS mount helper:

```
sudo mount -t efs -o tls [redacted] / efs
```

Using the NFS client:

```
sudo mount -t nfs4 -o nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresport[redacted].efs.ap-southeast-2.amazonaws.com:/ efs
```

See our user guide for more information. [Learn more](#)

Close

## Setup

Replace `{{FileSystemId}}` below

```
1 # Note You need to replace the {{FileSystemId}} with your own EFS ID before using the config.
2 # You may also need to modify the SecurityGroupIds for the head and worker nodes in the config file.
3
4 setup_commands:
5   - sudo kill -9 `sudo lsof /var/lib/dpkg/lock-frontent | awk '{print $2}' | tail -n 1`;
6     sudo pkill -9 apt-get;
7     sudo pkill -9 dpkg;
8     sudo dpkg --configure -a;
9     sudo apt-get -y install binutils;
10    cd $HOME;
11    git clone https://github.com/aws/efs-utils;
12    cd $HOME/efs-utils;
13    ./build-deb.sh;
14    sudo apt-get -y install ./build/amazon-efs-utils*deb;
15    cd $HOME;
16    mkdir efs;
17    sudo mount -t efs {{FileSystemId}}:/ efs;
18    sudo chmod 777 efs;
```

## IAM Role and EC2 Instance Profile

By default, Ray nodes in a Ray AWS cluster have full EC2 and S3 permissions (i.e. `arn:aws:iam::aws:policy/AmazonEC2FullAccess` and `arn:aws:iam::aws:policy/AmazonS3FullAccess`). This is a good default for trying out Ray clusters but you may want to change the permissions Ray nodes have for various reasons (e.g. to reduce the permissions for security reasons). You can do so by providing a custom `IamInstanceProfile` to the related `node_config`:

```
1 available_node_types:
2   ray.worker.default:
3     node_config:
4       ...
5     IamInstanceProfile:
6       Arn: arn:aws:iam::YOUR_AWS_ACCOUNT:YOUR_INSTANCE_PROFILE
7
```

### Accessing S3

In various scenarios, worker nodes may need write access to an S3 bucket, e.g., Ray Tune has an option to write checkpoints to S3 instead of syncing them directly back to the driver.

If you see errors like “Unable to locate credentials”, make sure that the correct `IamInstanceProfile` is configured for worker nodes in your cluster config file. This may look like:

```
1 available_node_types:
2   ray.worker.default:
3     node_config:
4       ...
5     IamInstanceProfile:
6       Arn: arn:aws:iam::YOUR_AWS_ACCOUNT:YOUR_INSTANCE_PROFILE
7
```

You can verify if the set up is correct by SSHing into a worker node and running

```
1 aws configure list
2
```

You should see something like

	Name	Value	Type	Location
2	----	-----	----	-----
3	profile	<not set>	None	None
4	access_key	*****XXXX	iam-role	
5	secret_key	*****YYYY	iam-role	
6	region	<not set>	None	None
7				

Please refer to this [discussion](#) for more details on accessing S3.

## Amazon CloudWatch

Create `cloudwatch-basic.yaml`

```
1 # cloudwatch-basic.yaml
2
3 provider:
4   type: aws
5   region: us-west-2
6   availability_zone: us-west-2a
```

```

7   # Start by defining a `cloudwatch` section to enable CloudWatch integration with your Ray cluster.
8   cloudwatch:
9     agent:
10      # Path to Unified CloudWatch Agent config file
11      config: "cloudwatch/example-cloudwatch-agent-config.json"
12    dashboard:
13      # CloudWatch Dashboard name
14      name: "example-dashboard-name"
15      # Path to the CloudWatch Dashboard config file
16      config: "cloudwatch/example-cloudwatch-dashboard-config.json"
17
18  auth:
19    ssh_user: ubuntu
20
21  available_node_types:
22    ray.head.default:
23      node_config:
24        InstanceType: c5a.large
25        ImageId: ami-0d88d9cbe28fac870 # Unified CloudWatch agent pre-installed AMI, us-west-2
26        resources: {}
27    ray.worker.default:
28      node_config:
29        InstanceType: c5a.large
30        ImageId: ami-0d88d9cbe28fac870 # Unified CloudWatch agent pre-installed AMI, us-west-2
31        IamInstanceProfile:
32          Name: ray-autoscaler-cloudwatch-v1
33        resources: {}
34    min_workers: 0

```

```

1  mkdir cloudwatch
2  cd cloudwatch
3  wget https://raw.githubusercontent.com/ray-project/ray/master/python/ray/autoscaler/aws/cloudwatch/example-
  cloudwatch-agent-config.json
4  wget https://raw.githubusercontent.com/ray-project/ray/master/python/ray/autoscaler/aws/cloudwatch/example-
  cloudwatch-dashboard-config.json
5
6  cd ..
7  ray up cloudwatch-basic.yaml

```

## Train ML model

```

1  ray up example-full.yaml
2
3  #####
4
5  # Jump into the cluster
6  # Get a remote shell on the head node.
7  ray attach example-full.yaml
8
9  #####
10
11 python -c """
12 import math
13 import torch
14 import gpytorch
15 from matplotlib import pyplot as plt
16 import ray

```

```

17
18 ray.init()
19
20 @ray.remote
21 def get_data():
22     train_x = torch.linspace(0, 1, 100)
23     # True function is sin(2*pi*x) with Gaussian noise
24     train_y = torch.sin(train_x * (2 * math.pi)) + torch.randn(train_x.size()) * math.sqrt(0.04)
25     test_x = torch.linspace(0, 1, 51)
26
27     return train_x, train_y, test_x
28
29
30 # We will use the simplest form of GP model, exact inference
31 class ExactGPModel(gpytorch.models.ExactGP):
32     def __init__(self, train_x, train_y, likelihood):
33         super(ExactGPModel, self).__init__(train_x, train_y, likelihood)
34         self.mean_module = gpytorch.means.ConstantMean()
35         self.covar_module = gpytorch.kernels.ScaleKernel(gpytorch.kernels.RBFKernel())
36
37     def forward(self, x):
38         mean_x = self.mean_module(x)
39         covar_x = self.covar_module(x)
40         return gpytorch.distributions.MultivariateNormal(mean_x, covar_x)
41
42 @ray.remote
43 def get_model():
44     # initialize likelihood and model
45     likelihood = gpytorch.likelihoods.GaussianLikelihood()
46     model = ExactGPModel(train_x, train_y, likelihood)
47
48     # Find optimal model hyperparameters
49     model.train()
50     likelihood.train()
51
52     # Use the adam optimizer
53     optimizer = torch.optim.Adam(model.parameters(), lr=0.1) # Includes GaussianLikelihood parameters
54
55     # "Loss" for GPs - the marginal log likelihood
56     mll = gpytorch.mlls.ExactMarginalLogLikelihood(likelihood, model)
57
58     return model, likelihood, mll, optimizer
59
60 train_x, train_y, test_x = ray.get(get_data.remote())
61 model, likelihood, mll, optimizer = ray.get(get_model.remote())
62
63 @ray.remote
64 def train_model(training_iter, model, mll, optimizer):
65
66     for i in range(training_iter):
67         # Zero gradients from previous iteration
68         optimizer.zero_grad()
69         # Output from model
70         output = model(train_x)
71         # Calc loss and backprop gradients
72         loss = -mll(output, train_y)
73         loss.backward()
74         print('Iter %d/%d - Loss: %.3f lengthscale: %.3f noise: %.3f' % (

```



```

75         i + 1, training_iter, loss.item(),
76         model.covar_module.base_kernel.lengthscale.item(),
77         model.likelihood.noise.item()
78     ))
79     optimizer.step()
80
81     return model, mll, optimizer
82
83 training_iter = 5
84 model, mll, optimizer = ray.get(train_model.remote(
85     training_iter, model, mll, optimizer
86 ))
87
88 @ray.remote
89 def eval_model(model, likelihood, train_x, train_y, test_x):
90     # Get into evaluation (predictive posterior) mode
91     model.eval()
92     likelihood.eval()
93
94     # Test points are regularly spaced along [0,1]
95     # Make predictions by feeding model through likelihood
96     with torch.no_grad(), gpytorch.settings.fast_pred_var():
97         observed_pred = likelihood(model(test_x))
98
99     with torch.no_grad():
100         # Initialize plot
101         f, ax = plt.subplots(1, 1, figsize=(4, 3))
102
103         # Get upper and lower confidence bounds
104         lower, upper = observed_pred.confidence_region()
105         # Plot training data as black stars
106         ax.plot(train_x.numpy(), train_y.numpy(), 'k*')
107         # Plot predictive means as blue line
108         ax.plot(test_x.numpy(), observed_pred.mean.numpy(), 'b')
109         # Shade between the lower and upper confidence bounds
110         ax.fill_between(test_x.numpy(), lower.numpy(), upper.numpy(), alpha=0.5)
111         ax.set_ylim([-3, 3])
112         ax.legend(['Observed Data', 'Mean', 'Confidence'])
113
114         f.savefig('/tmp/gp_plot.png')
115
116 ray.get(eval_model.remote(
117     model, likelihood, train_x, train_y, test_x
118 ))
119
120 ray.shutdown()
121 """
122
123 ls /tmp
124 exit
125
126 #####
127
128 # Tear down the cluster.
129 ray down example-full.yaml

```