

HPC as a service on AWS – User Guide

More and more Weizmann Research Labs start using **Amazon Web Services (AWS)** cloud platform. This user guide will walk you through the authentication and authorization processes in the AWS, how you can easily create a new working environment on the cloud, how to upload & download your data and most important how run your hpc workloads on AWS?

- [AWS Environment](#)
- [HPC Development Environment](#)
- [Upload\Download data from Weizmann to AWS](#)
- [HPC Tool](#)

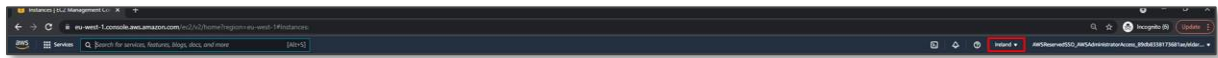
① **Please note:** Throughout this user guide the phrases “EC2 Instance”, “Server”, “Machine” and “Linux Machine” are used interchangeably and aim for a virtual machine that is deployed in the AWS cloud.

AWS Environment

Getting started

Onboard to AWS

- ① **Please note:** All the existing **AWS resources** are located under **the Ireland region (eu-west-1)**. The region in the AWS console may be found on the top right section.



Login to the AWS console

Access the AWS Console

1. Access the user portal: If you have already accepted the invitation from AWS Single Sign-On, use your **User portal URL** and enter the **username** to login.
2. Locate the invitation email:

Email subject: [welcome email](#)

Sender address: IT Weizmann

Dear #USERNAME#,

You have joined the #GROUPNAME# lab on the AWS HPC cluster.

Belonging to a member lab provides you with the privilege of running different HPC cluster jobs on the dedicated AWS Cloud account.

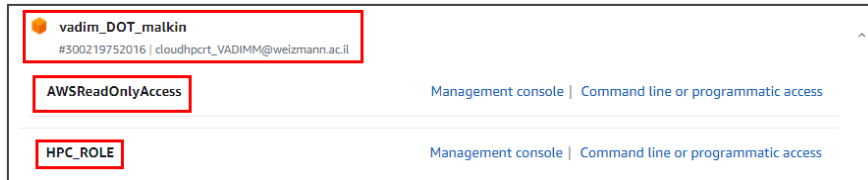
To login to your aws account please use the following link:

[AWS Console](#)

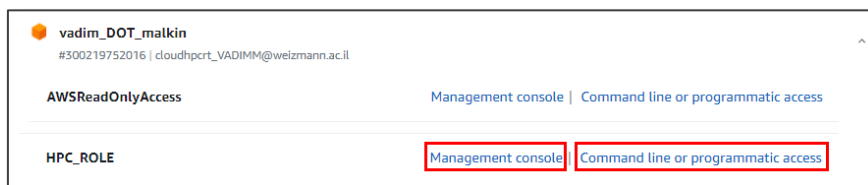
3. Click on the **URL** indicated in the email to navigate to the user portal.
4. Enter your **institutional username and password** and click on **Login**.



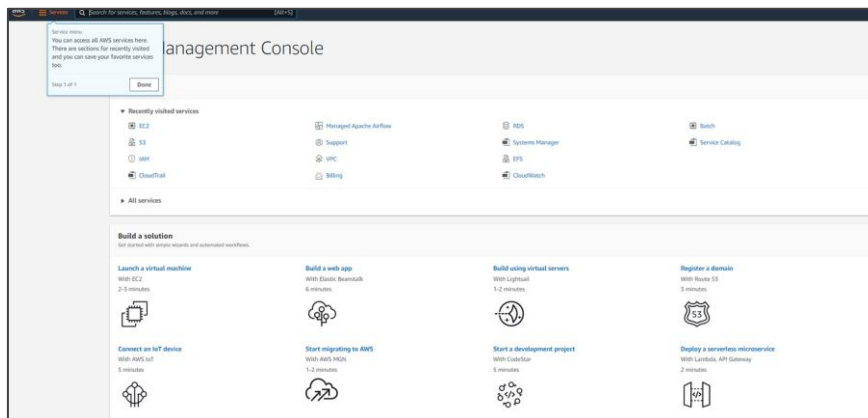
- Click and choose your **lab account**.
- Click and choose the **required permission level** to access with:
 - ➔ **AWSReadOnlyAccess** – Allows viewing only (Role 1).
 - ➔ **HPC_ROLE** – Allows building resources (Role 2).



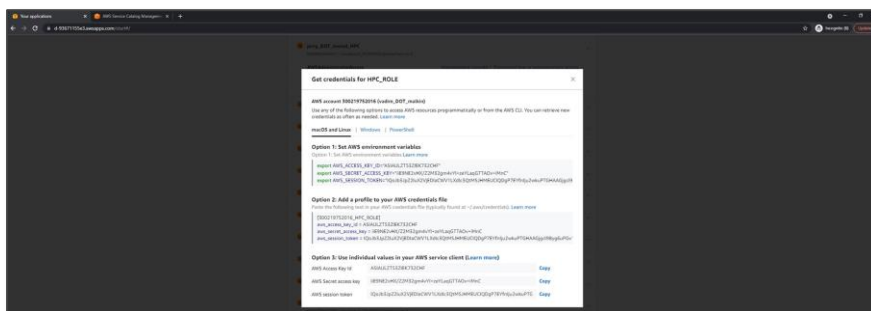
- As needed, click on **Management console** or **Command line or programmatic access**.



- When clicking on **Management console**, the following screen will appear:



- When clicking on **Command line or programmatic access**, the following screen will appear, this are the temporary credentials that you can use on your terminal , see [programmatic access to AWS CLI](#):



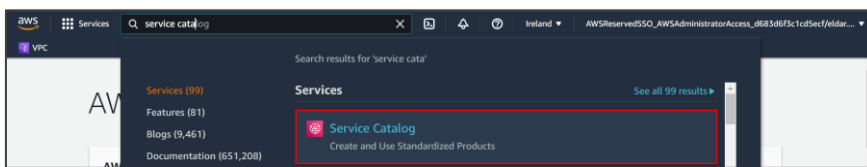
HPC Development Environment

The HPC development Environment is an EC2 Instance (a virtual machine) in your laboratory dedicated AWS account. The instance is an amazon Linux 2 server, which you will connect directly from your local terminal (e.g., Putty, Mobaxterm, iTerm, etc). Within the EC2 Instance, you will use AWS CLI to interact with the different AWS services. In addition, the EC2 Instance includes utilities and integrations required to easily interact with your S3 Bucket, EFS Filesystem, and configuring and running HPC jobs on AWS Batch.

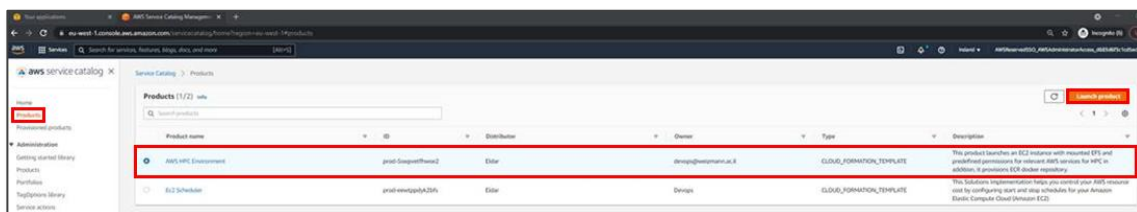
Getting started

Provision the AWS HPC Environment

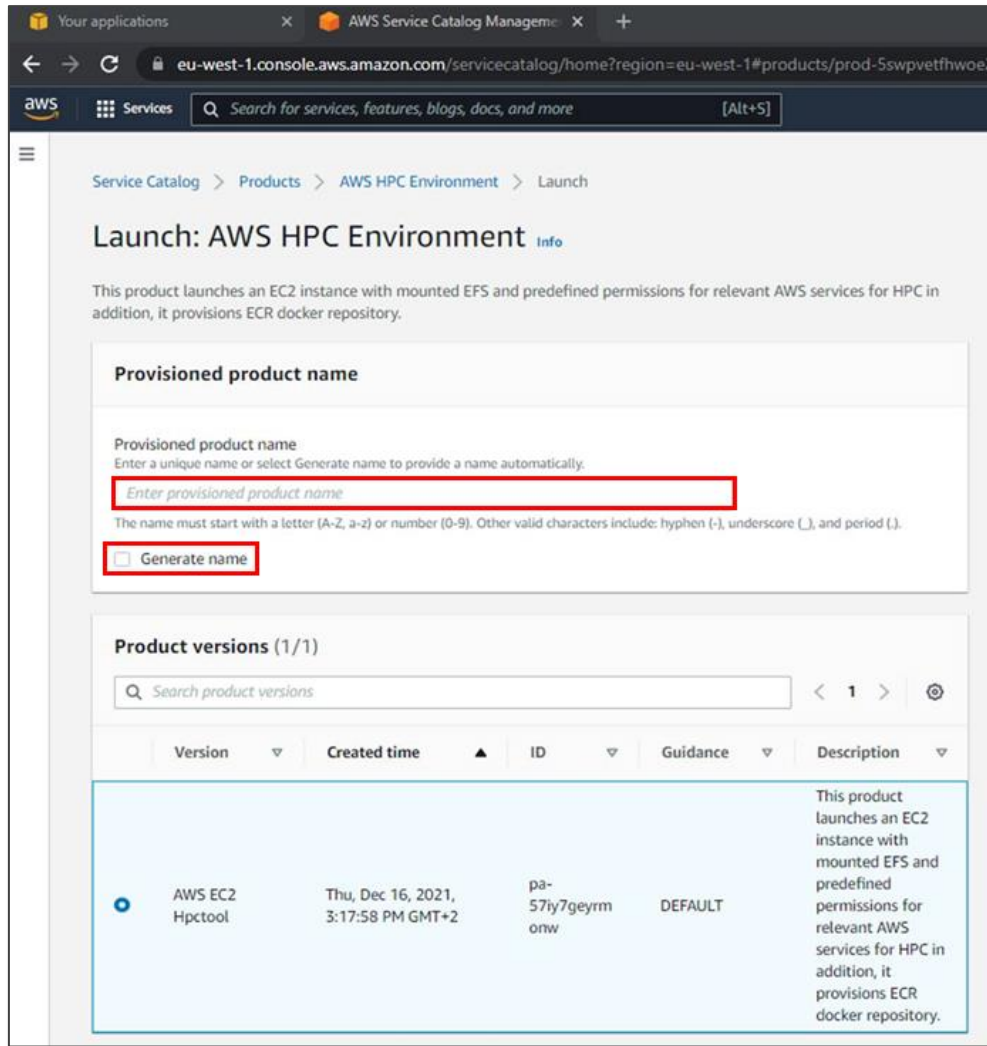
1. Open the **AWS console** to access your account (for further instructions, see [Access the AWS console](#)).
2. Navigate to the **AWS Service Catalog** and open the [Products](#) page (for further instructions, see [Access AWS Service Catalog Console](#)).
3. On the **Products** page, select the **AWS HPC environment** product.



4. Then, on the top right corner, click on **Launch product**.



5. **Launch: AWS HPC Environment** screen will appear.
6. Type a name for the **provisioned product** or click on **Generate name**.



7. Then, the product's form will be loaded.

Parameters

BaseDockerImage
The base docker image used for build HPC default image

Default value can be overridden.

HPCProjectName
The project name which hpctool will prepare

Default value can be overridden.

InstanceName
Linux EC2 server name this will be use as a value for the Name Tag

InstanceSchedule
Choose the periods where your development environment will be running. Outside of that period the server will be shutdown.

Default value can be overridden.

InstanceType
EC2 instance type

Default value can be overridden.

JDCCommand
The command which will be used in job definition

Default value can be overridden.

KeyPairName
Only required for SSH connection method. Please specify an EC2 key pair name which is defined in this account.

OwnerEmail
Please enter your email

ServerImageType
Server image type hpc tool will build AMI for

Default value can be overridden.

8. Fill out the form with the **required information**.

Parameter Name	Description	Constraint	Example
BaseDockerImage	This is an optional value to provide the name of the public docker image to integrate it to your AWS HPC environment. The default value is docker image of amazonlinux, leave it this way if you don't have any other docker image.	In this field you can use any Unicode character up to 256 chars.	Development server
HPCProjectName	This parameter suggest a project name for your HPC use case, all the provisioned resources are tagged with the project name. It is recommended to use a user friendly project name like name of the user, who uses the environment.	In this field you can use any Unicode character up to 256 chars.	Development server
InstanceName	This will be the value for the tag "Name" which is associated with the	In this field you can use any Unicode	Development server

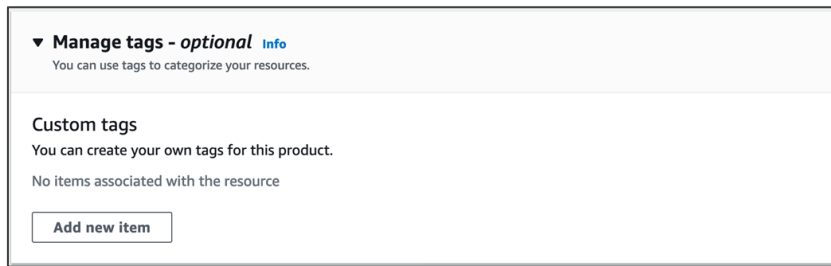


	EC2 Instance (please note that this is not the server domain name). This field will help you to identify the instance in the AWS console or AWS CLI.	character up to 256 chars.	
InstanceSchedule	If this parameter is set to other value than "No-Schedule" the server will be stopped (shutdown) at any period outside of the specified one. E.g., stop-on-7pm, will shut-down/stop the server every day at 7pm.	Currently, there are only two options available: "No-Schedule" "stop-on-7pm". If another schedule is required, please contact IT.	Development server
EC2 Instance Type	Instance types comprise varying combinations of CPU, memory, storage, and networking capacity and give you the flexibility to choose the appropriate mix of resources for your needs. If you are not sure which to choose, use the default t3.small. Checkout the list of the different instance types and their details .	Must be a value from the allowed values listed in the "drop down list".	t3.small
JDCCommand	This parameter will store the command that you are planning to execute on the cluster, relevant if you specified the preferred Docker image. You may leave this field as default and modify the command later.	Name of an AWS System Manager Parameter Store, which holds the AMI-Id string.	
KeyPairName	Choose the method to connect to the EC2 Instance. It's recommended to use SSM Session Manager unless you have specific need for SSH connection. If you choose SSH you will have to specify a valid EC2 Key Pair name under "SSH Key Pair Name" parameter. If you choose SSMSessionManager, see Working with the EC2 Instance for guidance how to connect to the instance using SSM Session Manager.	Can be either "SSMSessionManager" Or "SSH".	Empty
Owner email	Please enter your email in this field, once the ec2 server will be ready –	Must be a valid Weismann Institute of	name@weismann.ac.il



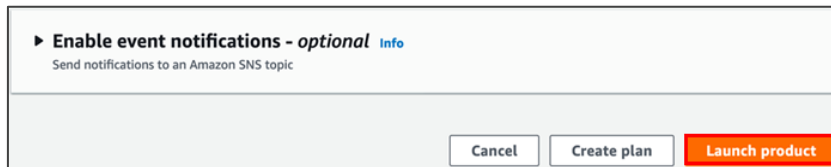
	you will get an email notification to start your work.	Science email address.	
ServerImageType	Please choose what is your hpc use case type CPU or GPU.	Must be a value from the allowed values listed in the "drop down list".	sg-079abf62f6e86cd06

9. With the **Manage Tags** option, you can add additional tags to the server. Use this option only when it is relevant to your operation.

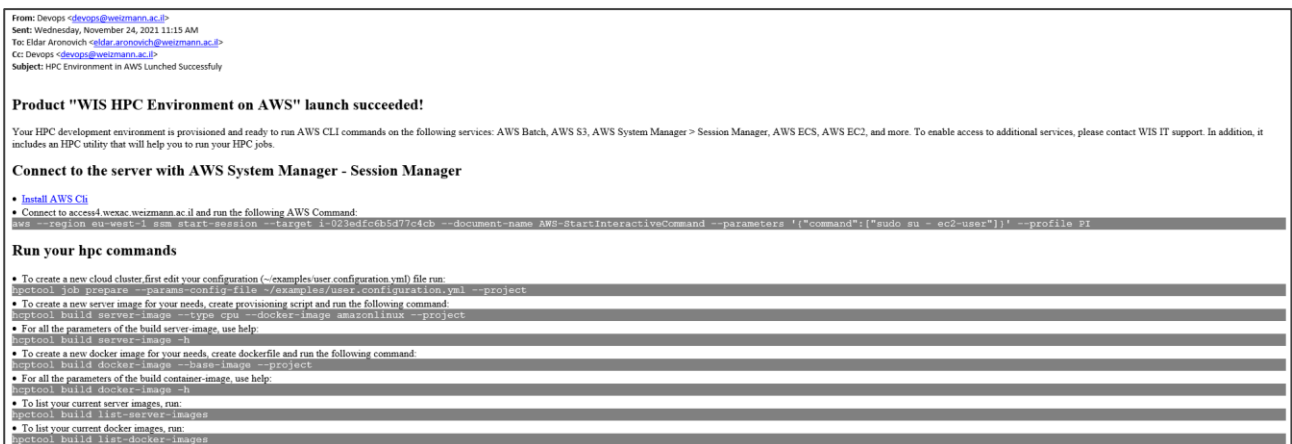


10. The **Enable event notifications** option is not enabled and should be ignored. You can follow the progress of the launch on the AWS Service Catalog product.

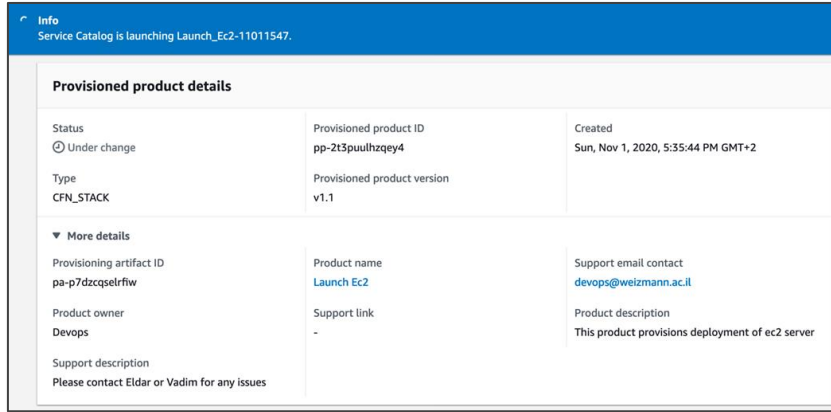
11. After completing the form, click on **Launch product** on the bottom right corner.



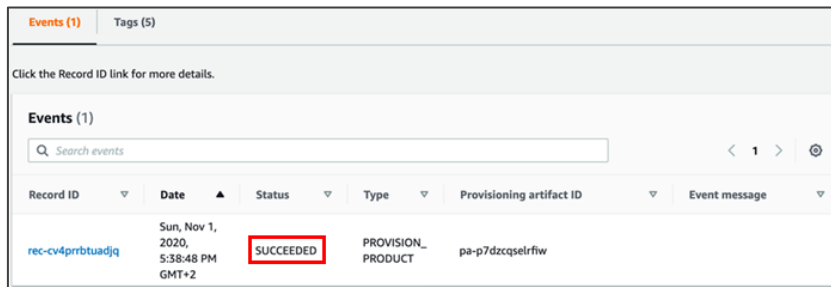
12. Upon successful provisioning of the product, you will receive an **email** with further instructions on connecting to your **new EC2 server**.



- For advanced users: Next, the **Provisioned product details** screen will appear. On the upper panel, you can see the status of the provisioned product. Once the status is set to **Available** the provisioning succeeded.

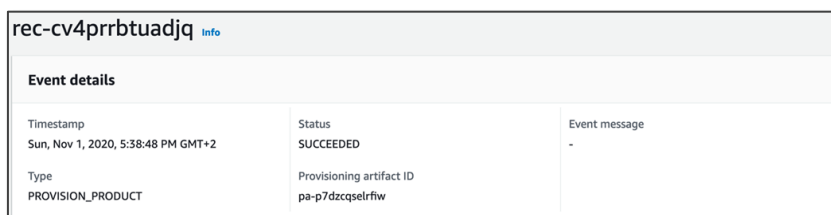


On the lower panel, you can see the **Events** panel, which lists the current and historical actions applied to this provisioned product. When the **product provisioning** (event type **PROVISION_PRODUCT**) is successful, the event status will be updated to **SUCCEEDED**.



❶ **Please note:** The provisioning should take up to 15 minutes. At the end, you will see the result under **Status**.

- Once the product launch completed successfully, see [Working with the EC2 instance](#).
- To see the server details, in the AWS Service Catalog, navigate to **provisioned products**.
- Select the server you own.
- On the **provisioned product** page, scroll down to the **events** panel and select the last event (event type **PROVISION_PRODUCT**).



18. On the event page lower panel, **Outputs**, you will find a list of details that will be required for you to connect to the server.

Key	Value	Description
CloudformationStackARN	arn:aws:cloudformation:eu-west-1:300219752016:stack/SC-300219752016-pp-2t3puulhzqey4/e8692730-1c57-11eb-b69e-02af5a91499d	The ARN of the launched C
InstanceId	i-09ad763221a32e898	
InstanceName	dronen	
SSMConnectCommandLinux	aws --region eu-west-1 ssm start-session --target i-09ad763221a32e898 --document-name AWS-StartInteractiveCommand --parameters [{"command":["sudo su - ec2-user"]}]	
SSMConnectCommandWindows	Start-SSMSession -Region eu-west-1 -Target i-09ad763221a32e898	

Programmatic access to AWS APIs

There are 2 configurations to work with the cloud, using AWS CLI: **WEXAC access servers** or **Command line or programmatic access**.

WEXAC access servers

1. Connect to either one of these two **WEXAC** access servers, **access3.wexac.weizmann.ac.il** or **access4.wexac.weizmann.ac.il**, with your WIS username\password.

```
eldara@LT22080255:~$ ssh eldara@access4.wexac.weizmann.ac.il
eldara@access4.wexac.weizmann.ac.il's password:
```

2. After connecting, a message will show that AWS authentication is established.

```
Last login: Sun May 31 13:05:58 2020 from 132.77.124.80
Authenticated to AWS.
NOTE: The authentication is temporary and will expire at 2020-05-31 14:05:31+00:00
Please contact the friendly admin team in case of a problem.
[eldara@access4 ~]$
```

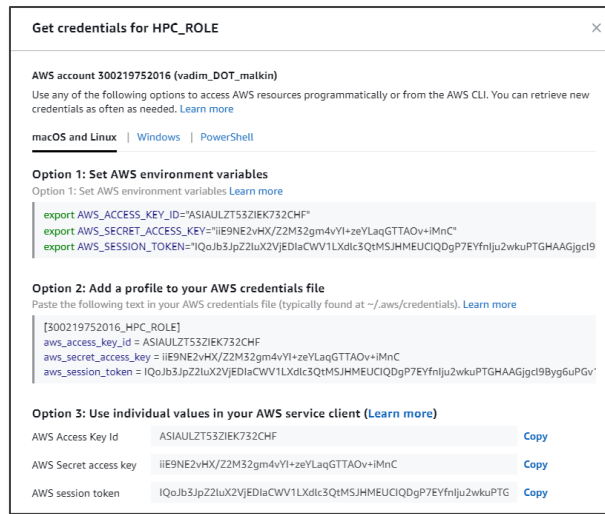
3. In case of authentication issues, a message will show that there is an error.

```
ERROR: Could not authenticate to AWS platform.
Please ignore if you are not using AWS services.
```

In this case, send an email to hpc@weizmann.ac.il or contact **Eldar Aronovich** or **Vadim Malkin**.

Command line or programmatic access

1. After clicking on **Command line or programmatic access**, this window will appear:



2. Click on **Option 1** and then a **Copy** button will appear.
3. To copy your **temporary credentials**, click on **Copy** and a message will appear: "Copied!"
4. Paste your **credentials** in your **Linux terminal**.

```
e1dara@LT22080255:~$ export AWS_SECRET_ACCESS_KEY="kp561Hyg51t3jEc1sYKDVzmkQx8Tpyr8vvgTkak2"
e1dara@LT22080255:~$ export AWS_SESSION_TOKEN="IQoJb3JpZ2luZXZlEDlaCWV1LXdlc3QtMSJHMEUCIQDgP7EYfnJlU2wkuPTGHAAGjgcl9Byy6uPGv'
21xPM/3lVJvkrHvCnInZmgPSKQgg7D8X8rjLFmH9sIFcH4Q38mbP1KQK7K580Jyc2Dn15zoojgrXkePaSkncppsc/MVDI9MFSVengV2517hPKZS88pYnSuUHVgF3LyZ/9d5VYIPe019u7hMVCW9wHhR7+h8Nq1NOBmFb3yus/xAToTCAGL941x1aZnsyzrK6F0y10H880Q/DU1sxeXSSbmb6TMD
xAR/7ikLmUyRed9BrsAz68OACTPKUxQ8do7Y7InGBxL/latpDDJ94a0BjgmAT9p1EtrPkcfceF7Xc0T4Ykusq2804qMGPabkY0Rr1gkAlqHIZyWlXGnGnfhq1LLM4yVQTQ7n/gzk35Qwq54S/vrRgblnaYKz3rHvXxF6dacf/ay0sUfV81FXwrt1h8cnu1f7q7ABz3o3tqprFCA8b58xhZqg
C0mInppBHTQ8BPFY398UzVSDfsLKZmsdXtdgfs60XQhXevQc"
```

5. To validate that your credentials are properly working with AWS, run on your terminal the following AWS CLI command: **aws s3 ls**
6. Make sure that you get an output: **pi-youraccountid-shared-bucket**

```
e1dara@LT22080255:~$ aws s3 ls
2021-03-17 21:13:53 pi-177224236425-shared-bucket
```

Install AWS CLI

Please select the instructions according to your local machine operating system:

- [AWS CLI - Linux](#)
- [AWS CLI - macOS](#)
- [AWS CLI - Windows](#)
- [AWS CLI - Docker image](#)

Install SSM Session Manager Plugin

Please select the instructions according to your local machine operating system:

- [Session Manager plugin on Windows](#)



- [Session Manager plugin on macOS](#)
- [Session Manager plugin on Linux](#)
- [Session Manager plugin on Ubuntu Server](#)

Connect to your EC2 instance

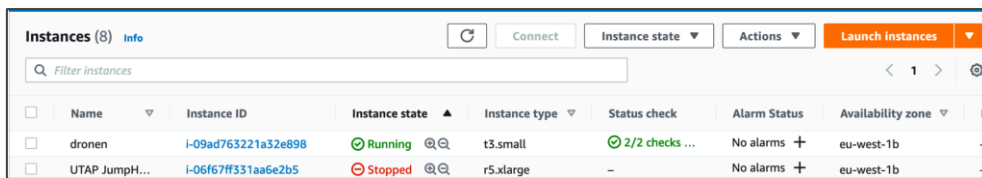
Your EC2 Instance is provisioned and ready to run AWS CLI commands on the following services: AWS Batch, AWS S3, AWS System Manager > Session Manager, AWS ECS, AWS EC2, and more. To enable access to additional services, please contact **WIS IT support**.

To connect the EC2 Instance you have two main alternatives, using **AWS System Manager > Session Manager** or connect via **SSH using an EC2 Key Pair** (the latter option is available only if the “EC2 Key Pair” parameter was provided when launching the HPC Development).

Before connecting, make sure you have the [server details](#).

Connect using Session Manager via AWS Console

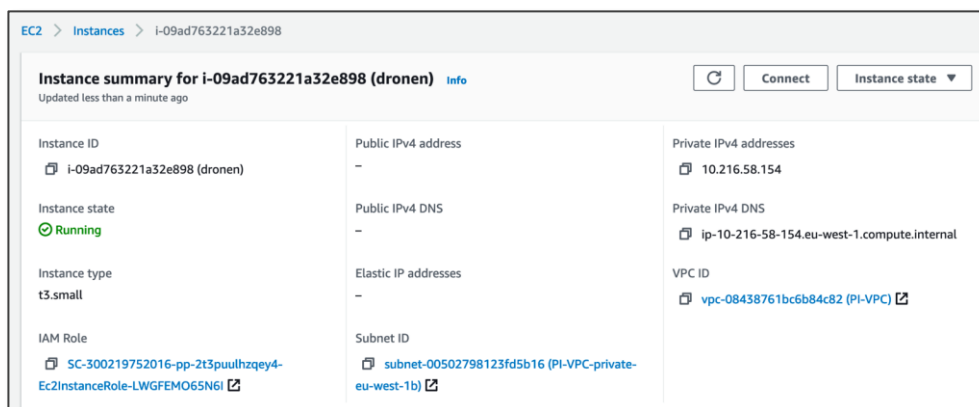
1. Open the **AWS Console** of your account and navigate to **AWS EC2 Service**.



Name	Instance ID	Instance state	Instance type	Status check	Alarm Status	Availability zone
dronen	i-09ad763221a32e898	Running	t3.small	2/2 checks ...	No alarms +	eu-west-1b
UTAP JumpH...	i-06f67ff331aa6e2b5	Stopped	r5.xlarge	-	No alarms +	eu-west-1b

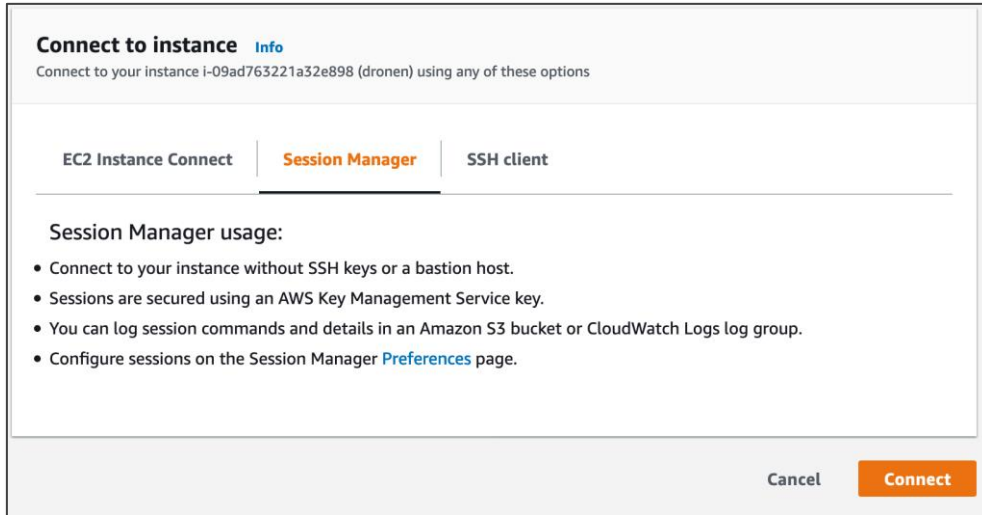
2. Use the **Name** column to identify your instance and select it by click on the **instance id**.

3. On the **instance** page (right upper corner), click on **connect**.



Instance ID	Public IPv4 address	Private IPv4 addresses
i-09ad763221a32e898 (dronen)	-	10.216.58.154
Instance state	Public IPv4 DNS	Private IPv4 DNS
Running	-	ip-10-216-58-154.eu-west-1.compute.internal
Instance type	Elastic IP addresses	VPC ID
t3.small	-	vpc-08438761bc6b84c82 (PI-VPC)
IAM Role	Subnet ID	
SC-300219752016-pp-2t3puulhzqey4-Ec2InstanceRole-LWGFEMO65N6i	subnet-00502798123fd5b16 (PI-VPC-private-eu-west-1b)	

4. Select **Session Manager** Tab and click **Connect** on the right lower corner.



5. A **new tab** will be opened in your browser with a new session to your instance.



6. Use this session to run commands on the server.

Connect using Session Manager via AWS CLI

1. [Install AWS CLI](#).
2. [Install AWS SSM Session Manager Plugin](#).
3. On your terminal, run the following **AWS Command**. You can find the full command with the instance-id in the provisioned product [outputs](#) section in the console.

Linux/MasOs:

```
aws --region eu-west-1 ssm start-session --target [instance-id] --document-name AWS-StartInteractiveCommand --parameters '{"command":["sudo su - ec2-user"]}'
```

Windows:

```
Start-SSMSession -Region eu-west-1 -Target [instance-id]
```

Connect using SSH over Session Manager

With this option you will have the exact same experience as regular SSH connection, but your instance must be associated with the public key and you have to own the private key. See [Create an EC2 SSH Key Pair](#) to prepare the required SSH key pair. As prerequisite, the AWS CLI and the SSM Manager plugin should both be installed, as explained in the previous section [Connect using Session Manager via AWS CLI](#).

1. On your machine, open the **SSH configuration file**:

Linux/MasOs:

The SSH configuration file is typically located at `~/.ssh/config`.

```
# SSH over Session Manager
```

```
host i-* mi-*
```

```
ProxyCommand sh -c "aws --region eu-west-1 ssm start-session --target %h --document-name
```

```
AWS-StartSSHSession --parameters 'portNumber=%p'"
```

Windows:

The SSH configuration file is typically located at `C:\Users\username\.ssh\config`.

```
# SSH over Session Manager
```

```
host i-* mi-*
```

```
ProxyCommand C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe "aws --region
```

```
eu-west-1 ssm start-session --target %h --document-name AWS-StartSSHSession --parameters
```

```
portNumber=%p"
```

2. Now **SSH** to your instance.

```
ssh -i /path/my-key-pair.pem ec2-user@instance-id
```

3. Now you can also copy files using SCP.

```
scp -i /path/my-key-pair.pem /path/SampleFile.txt ec2-user@instance-id:~
```

Create an EC2 SSH Key Pair

AWS uses public-key cryptography to secure the login information for your instance. A Linux instance, has no password to use for SSH access; you use a key pair to log in to your instance securely. You specify the name of the key pair when you create your compute environment, then provide the private key when you log in using SSH.

If you haven't created a key pair already, you can create one using the Amazon EC2 console.

To create a key pair:

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>
 2. From the navigation bar, select a **Region** for the key pair. You can select any Region that's available to you, regardless of your location: however, key pairs are specific to a Region. For example, if you plan to launch an instance in the US West (Oregon) Region, you must create a key pair for the instance in the same Region.
 3. In the navigation pane, choose **Key Pairs, Create Key Pair**.
 4. In the **Create Key Pair** dialog box, for **Key pair name**, enter a name for the new key pair , and choose **Create**. Choose a name that you can remember, such as your IAM user name, followed by `-key-pair`, plus the Region name. For example, `me-key-pair-uswest2`.
 5. The private key file is automatically downloaded by your browser. The base file name is the name you specified as the name of your key pair, and the file name extension is `.pem`. Save the private key file in a safe place.
- ① **Please note:** This is the only chance for you to save the private key file. You'll need to provide the name of your key pair when you launch an instance and the corresponding private key each time you connect to the instance.
6. If you will use an SSH client on a Mac or Linux computer to connect to your Linux instance, use the following command to set the permissions of your private key file so that only you can read it.

```
$ chmod 400 your_user_name-key-pair-region_name.pem
```

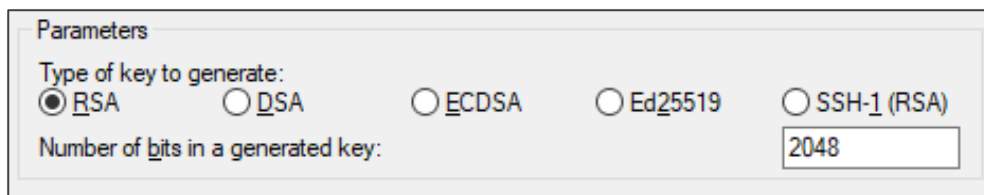
For more information, see [Amazon EC2 Key Pairs](#) in the *Amazon EC2 User Guide for Linux Instances*.

To connect to your instance using your key pair:

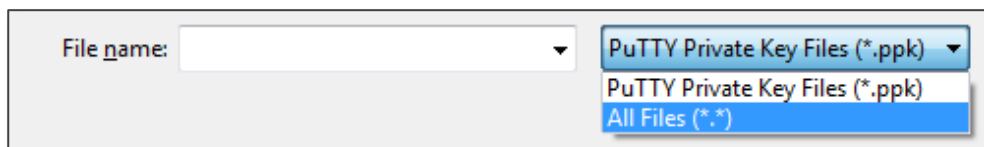
To connect to your Linux instance from a computer running Mac or Linux, specify the `.pem` file to your SSH client with the `-i` option and the path to your private key. To connect to your Linux instance from a computer running Windows, you can use either MindTerm or PuTTY. If you plan to use PuTTY, you'll need to install it and use the following procedure to convert the `.pem` file to a `.ppk` file.

To prepare to connect to a Linux instance from Windows using PuTTY (Optional):

1. Download and install PuTTY from <http://www.chiark.greenend.org.uk/~sgtatham/putty/>. Be sure to install the entire suite.
2. Start PuTTYgen (for example, from the **Start** menu, choose **All Programs, PuTTY, and PuTTYgen**).
3. Under **Type of key to generate**, choose **RSA**. If you're using an earlier version of PuTTYgen, choose **SSH-2 RSA**.



4. Choose **Load**. By default, PuTTYgen displays only files with the extension `.ppk`. To locate your `.pem` file, choose the option to display files of all types.



5. Select the private key file that you created in the previous procedure and choose **Open**. Choose **OK** to dismiss the confirmation dialog box.
6. Choose **Save private key**. PuTTYgen displays a warning about saving the key without a passphrase. Choose **Yes**.
7. Specify the same name for the key that you used for the key pair. PuTTY automatically adds the `.ppk` file extension.

Know your Linux environment

The created EC2 Instance is configured to be ready for immediate work against the AWS and specifically HPC environments. Following are some of the main configurations included with the Linux machine.

- **AWS CLI:** Once you are connected to the instance (either via SSM Session Manager or directly via SSH), you can now interact with AWS services using AWS CLI or Any AWS SDKs (for the latter, you might be required to install the SDK package first). Using AWS CLI is already set for you. You are not required to install it or to set any credentials for authentication.
- **Mounted datastores:** the ec2 Instance includes two mount points. The first one is `"/efs"` which mounts the EFS File System, and the second is `"/s3"` which mounts a dedicated S3 bucket.
- **Docker CLI:** the docker cli enables you to manage and run containers on the EC2 Instance.
- **hpctool:** the tool is installed on the machine and is available to use immediately.
- **Proxy settings:** the EC2 Instance has network connectivity to selected AWS Services (EC2, SSM, S3, EFS, ECS, ECR, CloudFormation) and the Weizmann Institute of Science (WIS) private network. In order for the EC2 Instance to communicate with the public internet (E.g., downloading packages, pulling source code from GitHub, pulling docker images from DockerHub, etc.) it is required to connect via the WIS private network. Thus, the `HTTP_PROXY` and `HTTPS_PROXY` environment variables are set to point the WIS network http proxy.

Upload\Download data from Weizmann to AWS

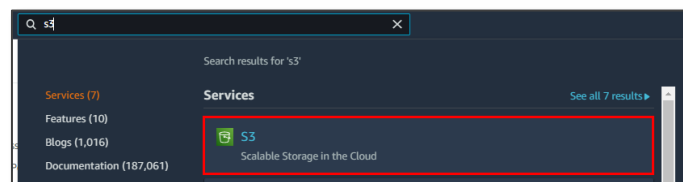
The synchronization of the data from WIS to the cloud and back, can be done using these storage services:

- ➔ **S3 – Amazon Simple Storage Service (Amazon S3)** is an object storage service offering industry-leading scalability, data availability, security, and performance. Customers of all sizes and industries can store and protect any amount of data for virtually any use case, such as data lakes, cloud-native applications, and mobile apps. With cost-effective storage classes and easy-to-use management features, you can optimize costs, organize data, and configure fine-tuned access controls to meet specific business, organizational, and compliance requirements.
- ➔ **EFS – Amazon EFS** is a simple, server-less, set-and-forget, elastic file system that makes it easy to set up, scale, and cost-optimize file storage in the AWS Cloud.

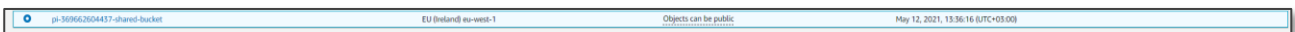
To upload or download files using **S3** service, you may use the **AWS Console** or the **AWS CLI commands**.

AWS Console

7. Choose the **S3** service in your AWS Console.



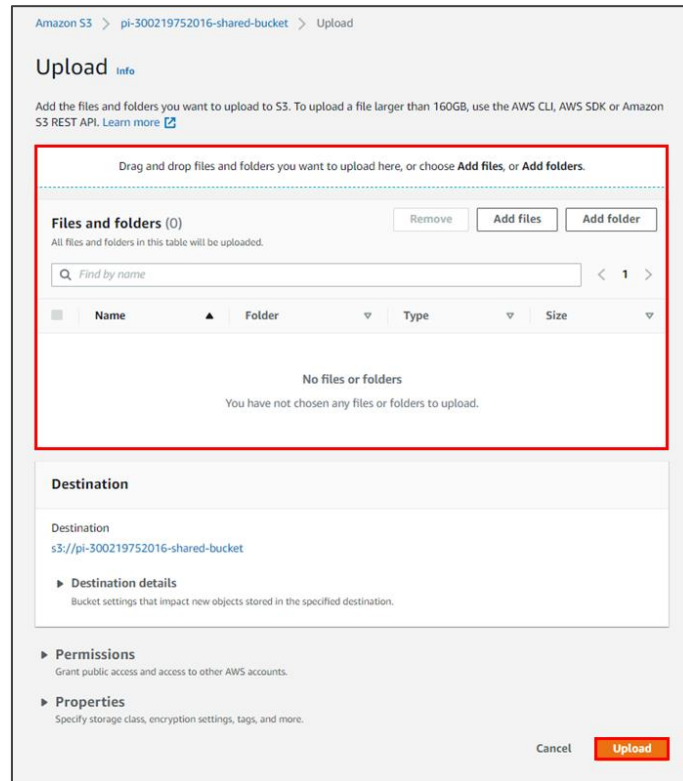
8. Click on the **bucket** you would like to upload file or directory.



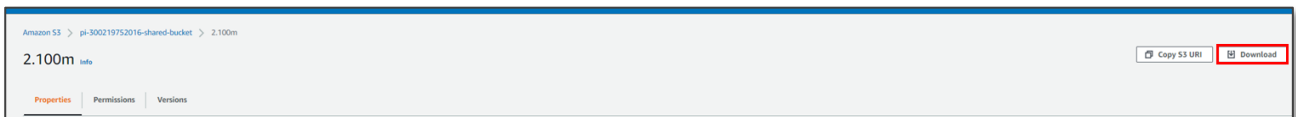
9. Click the **Upload** button.



10. Choose the **folder** of the files you would like to upload and click **Upload**.



11. To download the file using the console navigate to **S3**, choose the **bucket** and the required file.
12. Then, press on it and click on **Download**.



AWS CLI commands

1. Upload to S3 -

- ➔ Get your **programmational credentials** – please refer the user to the relevant section.
- ➔ Verify the credentials by running **AWS S3 ls** (command to list you s3 buckets).
- ➔ Run the following command on the source machine (i.e. local machine or the EC2 Instance) to copy files or folder to an S3 bucket:

`aws --region eu-west-1 s3 sync [local-path] s3://[bucket-name]/directory/` (**sync local to S3**)

`aws s3 cp -recursive [local-path] s3://[bucket-name]/directory/` (**copy local to S3**)

2. Download from S3 -

- ➔ Get your **programmational credentials** – please refer the user to the relevant section.
- ➔ Verify the credentials by running **AWS S3 ls** (command to list you s3 buckets).

- Run the following command on the source machine (i.e. local machine or the EC2 Instance) to copy files or folder to an S3 bucket:

```
aws --region eu-west-1 s3 sync s3://[bucket-name]/directory/ [local-path] (sync S3 folder to local drive)
```

```
aws s3 cp --recursive s3://[bucket-name]/directory/ [local-path] (Copy s3 folder to local drive)
```

To upload or download files to **EFS** using the **AWS CLI** commands:

- Upload -

```
scp -i /path/my-key-pair.pem /path/SampleFile.txt ec2-user@instance-id:~
```

- Download -

```
scp -i /path/my-key-pair.pem / ec2-user@instance-id:/efs/directory local-path/
```

HPC Tool

HPC Job defines business logic or processing procedure on a designated set of data and results in output, the processed data. Jobs are executed on compute resources, e.g., bare-metal servers, virtual machines, containers, or even server-less compute. The job execution duration and the compute resources consumed vary according to the processing complexity and the amount of data. Jobs are usually run within a batch of jobs, and those batches are managed by a job scheduler. The scheduler manages which jobs run where and when.

AWS offers AWS Batch as a fully managed service that schedules jobs, and as such, helps you to run batch computing workloads of any scale. AWS Batch automatically provisions compute resources and optimizes the workload distribution based on the quantity and scale of the workloads. With AWS Batch, there's no need to install or manage batch computing software, so you can focus your time on analyzing results and solving problems.

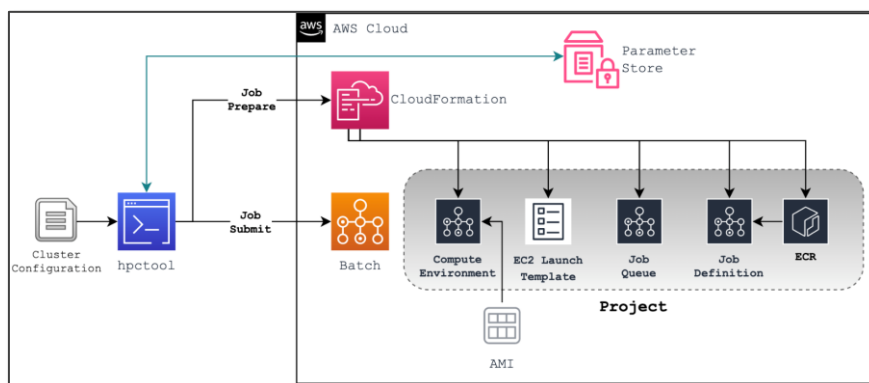
hpctool is a command-line tool that enables the user to run HPC workloads/jobs on [AWS Batch](#). Running jobs on AWS Batch involves different infrastructure settings, for example creating an AWS Batch compute environment. To create compute environment, one is required to identify appropriate subnet, security group, amazon machine image, and launch template that mounts different storage types. To ease the experience of creating the different resources required to run jobs, we have created the *hpctool*. With *hpctool* the user provides

values in a scientist-friendly configuration file and prepares a project into which he then submits jobs. In addition, hpctool include capabilities to create [amazon machine images](#), and [docker images](#), in the context of your project.

hpctool project

The hpctool project is a set of configurations and resources that defines the environment in which jobs are executed. The project state is kept in the AWS Environment, thus the hpctool CLI is essentially stateless, while the project is stateful. Most of the hpctool operations are in the context of a project. When submitting a job, a project name is required for the hpctool, so it can execute the job in the correct environment. Once created, a project includes different AWS Batch resources, an Amazon Machine Image to be used for the compute nodes, a configuration (Launch template) for the compute nodes, and a dedicated Elastic Container Repository to host the project docker images.

The project name must be unique across the same AWS Account (LAB). However, many projects may exist in parallel.



Setting up

If you already launched an [HPC Development Environment](#) and you are able to [connect to your EC2 Instance](#), then you are mostly ready to use hpctool, continue with [Getting started](#). Alternately, you can use your local station to run hpctool to manage your projects, submit jobs, build docker images, and build Amazon machine images. If you use your local station and would like to access the shared elastic filesystem (EFS) you will have to follow the procedure to mount EFS to an on-premises server, see [Upload/Download data from Weizmann to AWS](#) for more details.

Getting started

Before submitting jobs, there are few preparation steps:

1. Upload **raw data** to either **S3** or **EFS**. See [Data Transfer](#) for more details.
2. Create a **configuration file**. See [Cluster Configuration file](#) or use the example file located under `/home/ec2-user/examples/` directory

3. Run the command to create the project:

```
hpctool job prepare --project myProject --config-file /path/to/myConfig.ini
```

4. Use *hpctool* to build a docker image that will execute your script/program.

```
hpctool build docker-image --project myProject --script /path/to/myProgram.py
```

5. Run the command to submit job:

```
hpctool job submit --project myProject
```

6. Run the command to list the job/s:

```
hpctool job list --project myProject --command python myProgram.py
```

Author the job business logic

When authoring the job script/program consider the following:

1. When submitting jobs with AWS Batch the runtime environment is a docker container.
2. There are two available permanent storage volumes for your container:

Host	Container*
<code>/usr/ec2-user/s3</code>	<code>/s3</code>
<code>/usr/ec2-user/efs</code>	<code>/efs</code>

(*) The container mount point is configurable using the configuration ini file.

3. If multiple instances of the job will be run in parallel, using the `--array` modifier, the script can use the `AWS_BATCH_JOB_ARRAY_INDEX` to distinguish the job instances.
4. The job will have access to different AWS Services, as following: Amazon S3, AWS System Manager, Amazon EC2, and Amazon CloudWatch.

Data Transfer

Transfer files to/from Amazon S3

In order to use AWS CLI command for S3 you need to have programmatic credentials set in your terminal/PowerShell. To get your credentials navigate to the [WIS AWS User Portal](#) and copy the programmatic credentials. Paste the credentials to your terminal/PowerShell and run

AWS CLI or S3Fuse. If you are using the [HPC development environment](#) you already have the credentials set up.

1. Use AWS CLI:

→ To upload files from a machine (on-premises machine or the EC2 Instance) to S3 bucket, use the following command:

```
aws --region eu-west-1 s3 sync [local-path] s3://[bucket-name]/[project-name]/
```

→ To download files from S3 bucket to a machine (on-premises machine or the EC2 Instance), use the following command:

```
aws --region eu-west-1 s3 sync s3://[bucket-name]/[project-name]/ [local-path]
```

2. Use AWS CLI:

s3fs allows Linux, macOS, and FreeBSD to mount an S3 bucket via FUSE. s3fs preserves the native object format for files, allowing use of other tools like AWS CLI.

→ A Using the [HPC development environment](#) you have the s3 bucket already mounted to the EC2 Instance which you are [connecting](#) to and you can write/read to the /home/ec2-user/s3/ directory.

→ To use on-premises (or another server) follow the instruction <https://github.com/s3fs-fuse/s3fs-fuse#installation>

Transfer files to/from Amazon Elastic File-System (EFS)

1. Using the [HPC development environment](#) you have the EFS file-system already mounted to the EC2 Instance which you are [connecting](#) to and you can write/read to the /efs directory.

2. To use on-premises (or another server) follow the [Walkthrough: Create and mount a file system on-premises with AWS Direct Connect and VPN](#)

Transfer files to/from on-premises machine to EC2 Instance

To move files directly between your on-premises machine to your dedicated EC2 Instance you can use SCP over SSH.

To create the SSH session to the EC2 Instance, you have 2 options:

1. SSH connection over SSM Session
2. SSH connection using KeyPair



Download the HPC Tool

1. **AWS credentials** – the hpctool artefact is stored on an S3 Bucket in the AWS Cloud. In order to access the S3 bucket and download the artefact to your Linux environment you need to have permissions.

→ Download to your **provisioned EC2 Instance**:

The EC2 instance is provisioned with built in AWS credentials. Thus, you can run the following AWS CLI commands directly on the instance.

→ Download to your **local machine** (laptop/PC):

In this case you need to setup credentials in the current terminal you are running the commands at. To set the credentials, navigate to the [AWS user portal](#).

2. Download the **latest release of hpctool**.

Using AWS CLI: `aws s3 cp s3://_____bucket____/hpctool/latest/filename /tmp/`. Using AWS Console: login to the AWS S3 console and identify the _____ bucket. Browse the objects to locate the *hpctool* package under *hpctool/latest/*

3. Installation:

Run `pip3 install /tmp/filename`

4. Verify the installation:

Run `hpctool`

Cluster Configuration File

```
[queue]

# The name of the jobs queue. This name will appear as an AWS Tag on the actual
resource.
name = <string>

# The priority of the job queue. Job queues with a higher priority (or a higher
integer value for the priority parameter) are evaluated first when associated
with the same compute environment. Priority is determined in descending order.
For example, a job queue with a priority value of 10 is given scheduling
preference over a job queue with a priority value of 1.
priority = <int>

[clusterX]

# The name of the cluster queue. This name will appear as an AWS Tag on the AWS
Batch Compute Environment.
name = <string>

# You can choose either to use EC2 On-Demand Instances and EC2 Spot
Instances. You can optionally set a maximum price so that Spot Instances only
launch when the Spot Instance price is under a specified percentage of the On-
Demand price.
type = <EC2 | SPOT>

# The allocation strategy to use for the compute resource if not enough
instances of the best fitting instance type can be allocated. This might be
because of availability of the instance type in the Region or Amazon EC2 service
limits. For more information, see Allocation Strategies in the AWS Batch User
Guide.
allocation_strategy = <BEST_FIT | BEST_FIT_PROGRESSIVE |
SPOT_CAPACITY_OPTIMIZED>

# Choose yes if you want your cluster to run jobs on GPUs
gpu = <no | yes>

min_vcpus = <int>

desired_vcpus = <int>

max_vcpus = <int>

server_size = optimal | m5.xlarge, r4.large, t3.2xlarge, p3.xlarge

server_image = ami-1234567890

ssh_keypair = <string>

# The order of cluster. Compute environments are tried in ascending order. For
example, if two compute environments are associated with a job queue, the
compute environment with a lower order integer value is tried for job placement
first.
order = <int>

[job_definition]

# The name of the jobs queue. This name will appear as an AWS Tag on the actual
resource.
```

```

name = <string>

#
runtime_parameters = <key,value;key,value;...>

# attempts, (onStatusReason,onReason,onExitCode,action)
retry_strategy =
<int>,<string>;<string>;<string>;RETRY|EXIT,<string>;<string>;<string>;RETRY|EXIT,...

#
retry_strategy_spot = <no | yes>

#
job_timeout = <seconds>

[container]

#
docker_image_uri = <string>

#
vcpus = <int>

#
gpus = <int>

#
memory = <int>

#
entrypoint = ["/bin/sh", "-c"]

#
command = ["/efs/bin/time", "job_script.sh"]

#
environment = key,value;key,value;...

# (host source path, name)
volumes = <src path>,<name>;<src path>,<name>;,...

# (src_volume_name,container_path)
mounts =
<src_volume_name>,<container_path>;<src_volume_name>,<container_path>;,...

[data]

#
efs = <efs-filesystem-id>

#
s3 = <bucketname>

[notification]

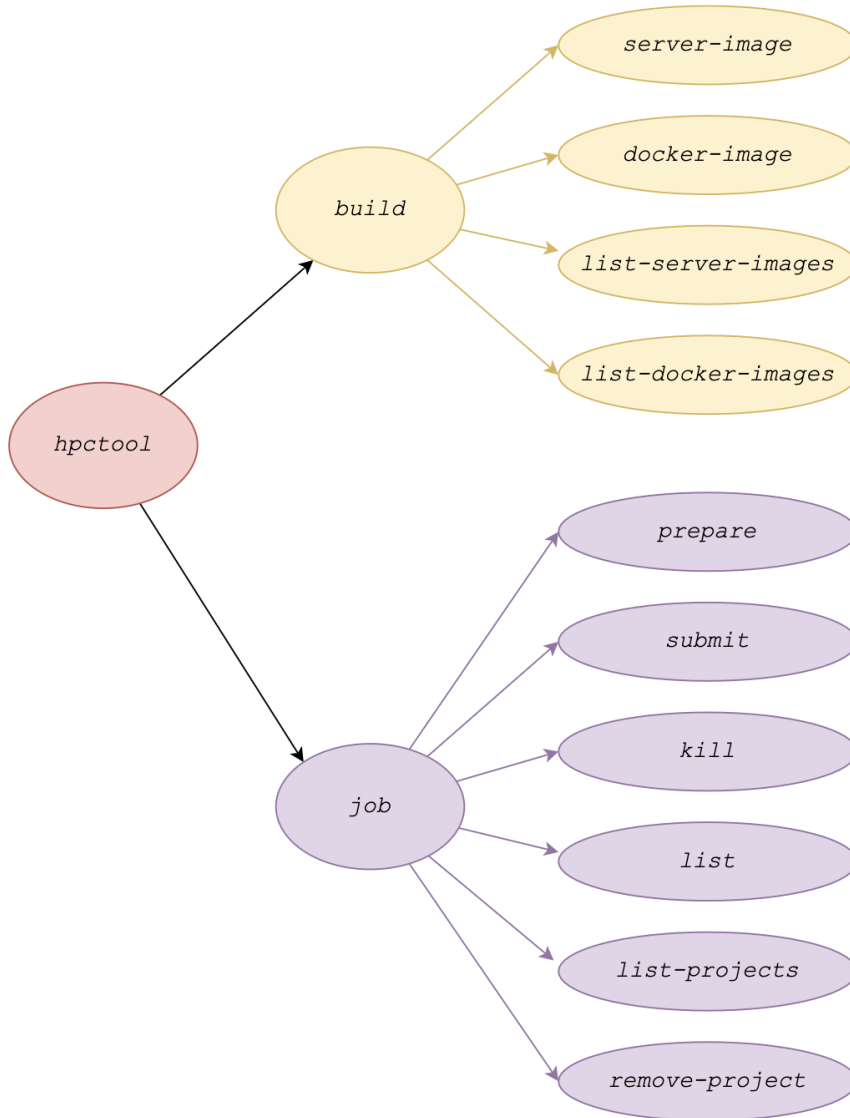
#
enabled = <no | yes>

#
email_address = foo@bar.com

```

Commands

The command line structure:



Hpctool

```
usage: hpctool [-h] [-d] [-q] [-v] {build,job} ...
```

A Cli tool to submit and manage HPC jobs with AWS Batch as backend

optional arguments:

```
-h, --help      show this help message and exit
-d, --debug     full application debug mode
-q, --quiet     suppress all console output
-v, --version   show program's version number and exit
```

sub-commands:

```
{build,job}
  build         build controller
```

job job controller

Usage: hpctool [sub-command] [args] [options]

Build server-image

```
hpctool build server-image usage: hpctool build server-image [-h] [--project PROJECT] [--name NAME] [--type {cpu,gpu}] [--base-image BASE_IMAGE] [--use-latest]
```

```
                                [--script SCRIPT] [--docker-image DOCKER_IMAGE] [--no-proxy] [--no-efs] [--no-s3] [--disk-size DISK_SIZE]
```

optional arguments:

```
-h, --help                show this help message and exit
--project PROJECT         The name of the project
--name NAME              Name for the created server image
--type {cpu,gpu}        Whether the image is CPU or GPU optimized
--base-image BASE_IMAGE the AMI id of an image to build on
--use-latest             if indicated, using the latest image labels with the
project name as base
--script SCRIPT          The full path of a shell script to configure the image
--docker-image DOCKER_IMAGE
                        The uri of a container image to be cached on the server
image
--no-proxy              if indicated, no proxy settings will be embedded to the
script
--no-efs                if indicated, no efs mount prerequisites installation
will be embedded to the script
--no-s3                 if indicated, no s3 mount prerequisites installation
will be embedded to the script
--disk-size DISK_SIZE  The disk size in GB for the built ami
```

Build docker-image

```
usage: hpctool build docker-image [-h] --project PROJECT [--base-image BASE_IMAGE] [--target-repository TARGET_REPOSITORY] [--script SCRIPT]
                                [--entrypoint ENTRYPOINT] [--tag TAG] [--
docker-file DOCKER_FILE]
```

optional arguments:

```
-h, --help                show this help message and exit
--project PROJECT         The name of the project
--base-image BASE_IMAGE  The source docker image. defaults to latest in local ecr
repository
--target-repository TARGET_REPOSITORY
                        The ECR repository to store the docker image; defaults
to the dedicated ECR previously created
--script SCRIPT          File path to shell script to use as part of docker image
build
--entrypoint ENTRYPOINT The entrypoint for the job container
--tag TAG                the container image tag.
--docker-file DOCKER_FILE
                        File name and path to a Dockerfile to use for container
image
build.
```

List Server images

```
usage: hpctool build list-server-images [-h] [--name-prefix NAME_PREFIX]

optional arguments:
  -h, --help            show this help message and exit
  --name-prefix NAME_PREFIX
                        Amazon Machine Image Name prefix to filter by.
```

List docker images

```
usage: hpctool build list-docker-images [-h] --project PROJECT [--repo REPO] [--count COUNT]

optional arguments:
  -h, --help            show this help message and exit
  --project PROJECT     The name of the project
  --repo REPO           Amazon ECR repository name.
  --count COUNT         Number of the docker images to list
```

Job prepare

```
usage: hpctool build docker-image [-h] --project PROJECT [--base-image BASE_IMAGE] [--target-repository TARGET_REPOSITORY] [--script SCRIPT]
                                     [--entrypoint ENTRYPOINT] [--tag TAG] [--docker-file DOCKER_FILE]

optional arguments:
  -h, --help            show this help message and exit
  --project PROJECT     The name of the project
  --base-image BASE_IMAGE
                        The source docker image. defaults to latest in local ecr repository
  --target-repository TARGET_REPOSITORY
                        The ECR repository to store the docker image; defaults to the dedicated ECR previously created
  -h, --help            show this help message and exit
  --project PROJECT     The name of the project
  --job-name JOB_NAME   The name of the job
  --job-timeout JOB_TIMEOUT
                        timeout duration after which AWS Batch terminates your jobs if they have not finished. If a job is terminated due to a timeout, it is not retried. The minimum value for the timeout is 60 seconds
  --job-retries-spot    When indicated and compute is SPOT, retries set with spot error
  --job-queue JOB_QUEUE
                        The job queue where the job is submitted
  --job-definition JOB_DEFINITION
                        The job definition used by this job
  --ami AMI             The Amazon Machine Image (AMI) ID used for instances launched in the compute environment.
  --ec2-types EC2_TYPES
                        The instances types that can be launched.
  --max-vcpus MAX_VCPUS
                        The maximum number of Amazon EC2 vCPUs that a compute environment can reach.
```



```

--ssh-key-pair SSH_KEY_PAIR
    The Amazon EC2 key pair that is used for instances
launched in the compute environment. You can use this key pair to
log in to your instances with SSH.
--cost-model {ON-DEMAND,SPOT}
    The Amazon EC2 key pair that is used for instances
launched in the compute environment. You can use this key pair to
log in to your instances with SSH.
--image-uri IMAGE_URI
    The image used to start a container. This string is
passed directly to the Docker daemon. Images in the Docker Hub
registry are available by default. Other repositories
are specified with repository-url/image:tag.
--vcpus VCPUS
    The number of vCPUs reserved for the container
--memory MEMORY
    The memory hard limit (in MiB) present to the container
--gpus GPUS
    The number of physical GPUs to reserve for the container
--command COMMAND [COMMAND ...]
    The command to send to the container that overrides the
default command from the Docker image or the job definition
--environment ENVIRONMENT
    The environment variables to send to the container. You
can add new environment variables, which are added to the
container at launch, or you can override the existing
environment variables from the Docker image or the job
definition.
--params-config-file PARAMS_CONFIG_FILE
    File with submit parameters
--generate-skeleton-config-file
    If indicated, a skeleton configuration file will be
rendered; Note: the actual command will not be executed if this
option indicated
--yes
    If indicated, will automatically execute the
cloudformation stack.

```

Job submit

```

usage: hpctool job submit [-h] --project PROJECT [--job-name JOB_NAME] [--job-
timeout JOB_TIMEOUT] [--array ARRAY] [--vcpus VCPUS]
    [--memory MEMORY] [--gpus GPUS] [--command ...]

optional arguments:
-h, --help            show this help message and exit
--project PROJECT     The name of the project
--job-name JOB_NAME  The name of the job
--job-timeout JOB_TIMEOUT
    timeout duration after which AWS Batch terminates your
jobs if they have not finished. If a job is terminated due to
a timeout, it is not retried. The minimum value for the
timeout is 60 seconds
--array ARRAY        submit multiple jobs with the same configuration
--vcpus VCPUS       The number of vCPUs reserved for the container
--memory MEMORY     The memory hard limit (in MiB) present to the container
--gpus GPUS        The number of physical GPUs to reserve for the container
--command ...       The command to send to the container that overrides the
default command from the Docker image or the job definition

```

Job kill

```
usage: hpctool job kill [-h] [--project PROJECT] [--job-name-prefix
JOB_NAME_PREFIX] [--job-queue JOB_QUEUE] [--job-id JOB_ID]

optional arguments:
  -h, --help            show this help message and exit
  --project PROJECT     The name of the project
  --job-name-prefix JOB_NAME_PREFIX
                        Substring of the job name
  --job-queue JOB_QUEUE
                        The job queue in scope
  --job-id JOB_ID      The job to delete
```

Job list

```
usage: hpctool job list [-h] [--project PROJECT] [--job-name-prefix
JOB_NAME_PREFIX] [--job-queue JOB_QUEUE]

optional arguments:
  -h, --help            show this help message and exit
  --project PROJECT     The name of the project
  --job-name-prefix JOB_NAME_PREFIX
                        Substring of the job name
  --job-queue JOB_QUEUE
                        The job queue in scope
```

Job list-projects

```
usage: hpctool job list-projects [-h]

optional arguments:
  -h, --help            show this help message and exit
```

Job remove-project

```
usage: hpctool job remove-project [-h] --project PROJECT

optional arguments:
  -h, --help            show this help message and exit
  --project PROJECT     The name of the project
```

Examples

- ➔ Multiple jobs: (showcase both methods on the same usecase)
 - loop over "hpctool job submit" to submit multiple jobs
 - use "hpctool --array" to submit multiple jobs [Tutorial: Using the array job index to control job differentiation - A...](#)
- ➔ EFS FileSystem Management.

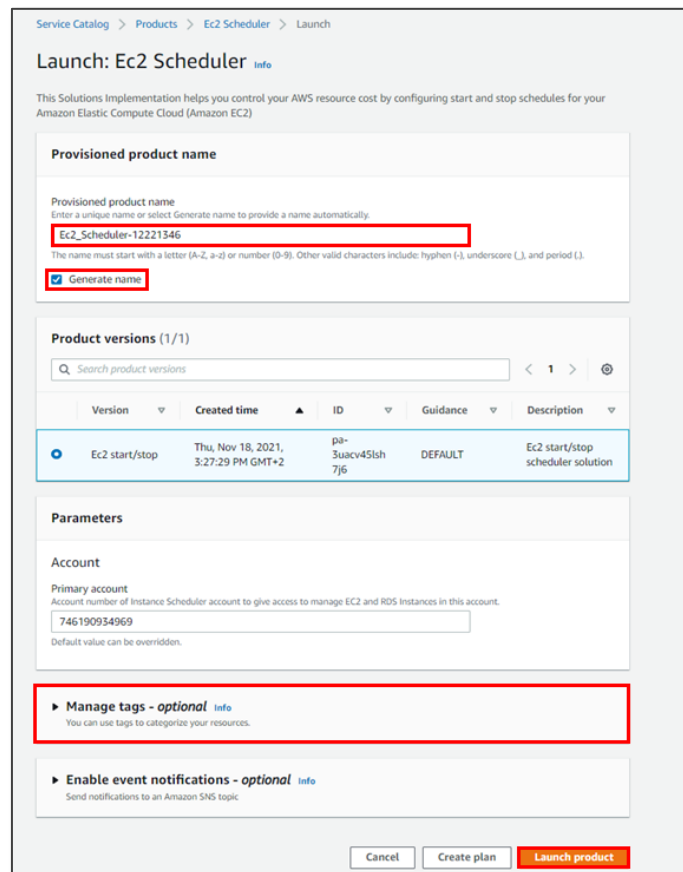
Schedule your instance (EC2 scheduler)

The purpose of this optional service is to start/stop your instance on specific hour based on the scheduler service. To install the service, follow these instructions:

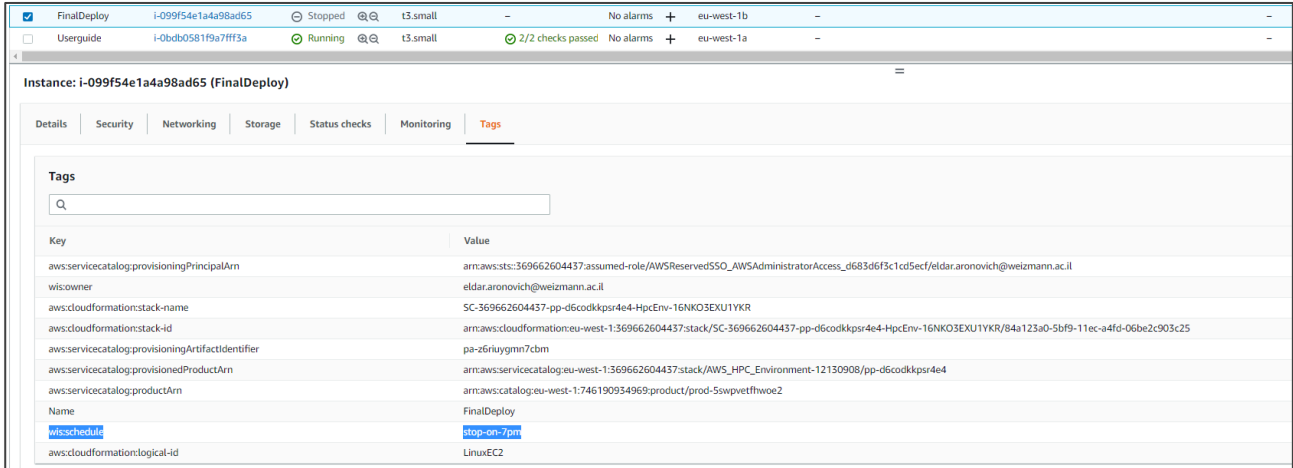
1. Open the **AWS console** to access your account (for further instructions, see [Access the AWS console](#)).
2. Navigate to the **AWS Service Catalog** and open the **Products** page (for further instructions, see [Access AWS Service Catalog Console](#)).
3. On the **Products** page, select the **EC2 Scheduler** product.
4. Then, on the top right corner, click on **Launch product**.



5. Mark the checkbox under **Generate name** or add your name for this product.
6. Click on **Launch product** and don't change any additional parameters.
7. Under **Manage tags**, you have an option to add **additional tags** to the server. Use this option only when it is relevant to your operation.



- Upon successful provisioning of the product, you may tag your instances with the following tags shown below. For example **wis:schedule** – your EC2 server will be automatically turned off on 7pm every day.



The screenshot shows the AWS Management Console interface for an EC2 instance. The instance is named 'FinalDeploy' with ID 'i-099f54e1a4a98ad65'. It is in a 'Stopped' state. The 'Tags' tab is selected, displaying a list of tags with their keys and values.

Key	Value
aws:servicecatalog:provisioningPrincipalArn	arn:aws:sts:369662604437:assumed-role/AWSReservedSSO_AWSAdministratorAccess_d683d6f3c1cd5ecf/eldar.aronovich@weizmann.ac.il
wis:owner	eldar.aronovich@weizmann.ac.il
aws:cloudformation:stack-name	SC-369662604437-pp-d6codkpsr4e4-HpcEnv-16NKO3EXU1YKR
aws:cloudformation:stack-id	arn:aws:cloudformation:eu-west-1:369662604437:stack/SC-369662604437-pp-d6codkpsr4e4-HpcEnv-16NKO3EXU1YKR/84a123a0-5bf9-11ec-a4fd-06be2c903c25
aws:servicecatalog:provisioningArtifactIdentifier	pa-z6riuvgmn7cbm
aws:servicecatalog:provisionedProductArn	arn:aws:servicecatalog:eu-west-1:369662604437:stack/AWS_HPC_Environment-12130908/pp-d6codkpsr4e4
aws:servicecatalog:productArn	arn:aws:catalog:eu-west-1:746190934969:product/prod-5swpvetffwvoe2
Name	FinalDeploy
wis:schedule	stop-on-7pm
aws:cloudformation:logical-id	LinuxEC2

