UbiHPC Club 4.0: Introducción al desarrollo de software para vehículos aéreos no tripulados (UAVs) inteligentes -- Sección 3

Leonardo Camargo Forero, Ph.D

CEO **UbiHPC**
www.ubihpc.com

Agenda

- Tutorial de Amazon Web Services (pendiente sesión 2)
- High Performance Robotic Computing
- Desarrollo de software para cluster de UAVs
- Preguntas

Agenda del curso

Sección 1 (4 horas)

- Revisión de aplicaciones actuales con UAVs y aplicaciones futuras inmediatas
- Emprendimiento
- Tutorial de Python general
- Tutorial de Git y GitHub

Sección 2 (4 horas)

- Tutorial de ArduPilot SITL (Software In The Loop)
- Tutorial de DroneKit

Sección 3 (4 horas)

- Tutorial de Amazon Web Services
- High Performance Robotic Computing
- Desarrollo de software para cluster de UAVs

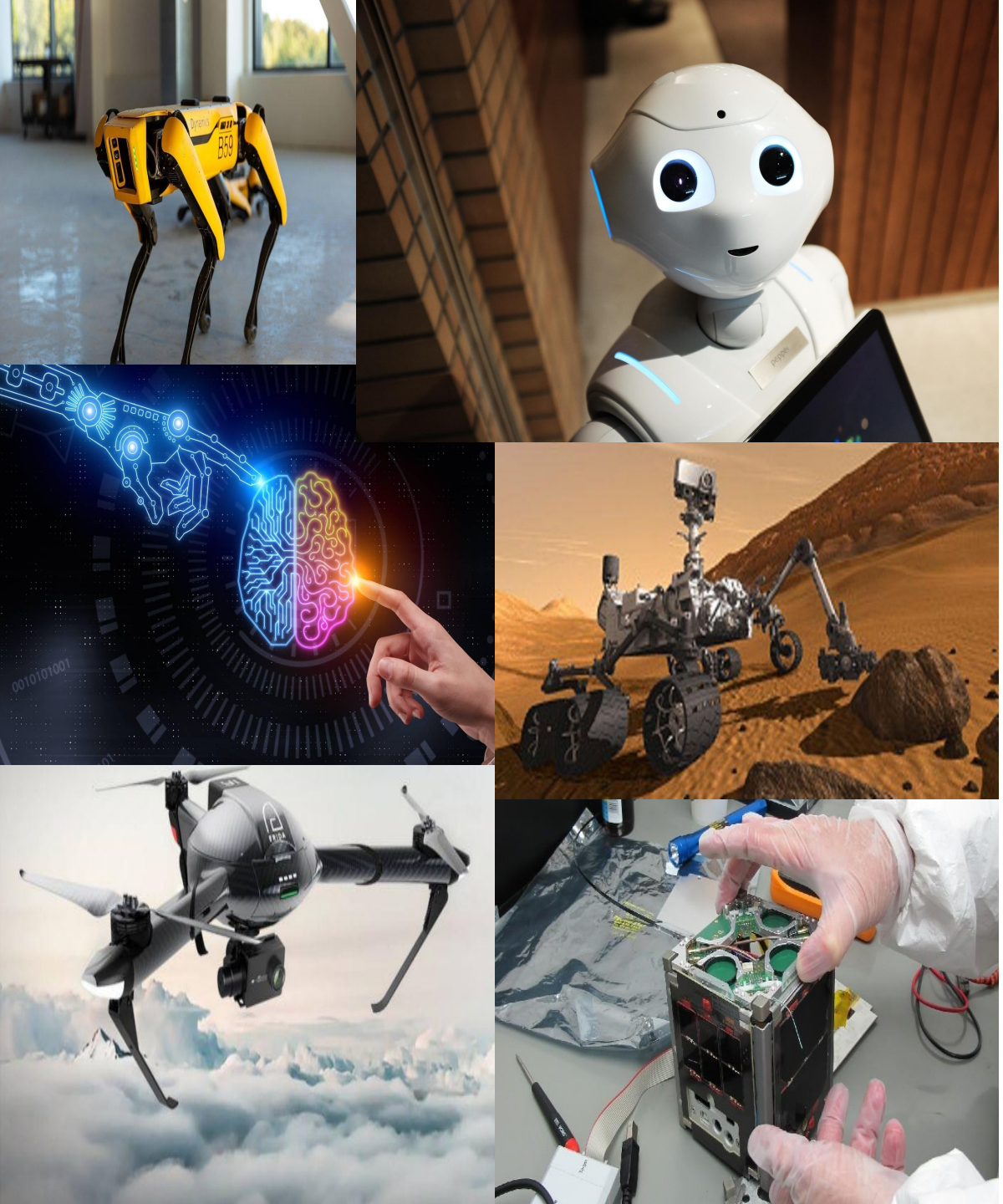
Sesión 4 (3 horas)

- Introducción a Deep Learning y Visión por computador con OpenCV.

Charla aplicaciones con UAVs y oportunidades en Universitat Politécnica de Catalunya

Charla integración de UAVs en espacio aéreo y oportunidades en Cranfield university

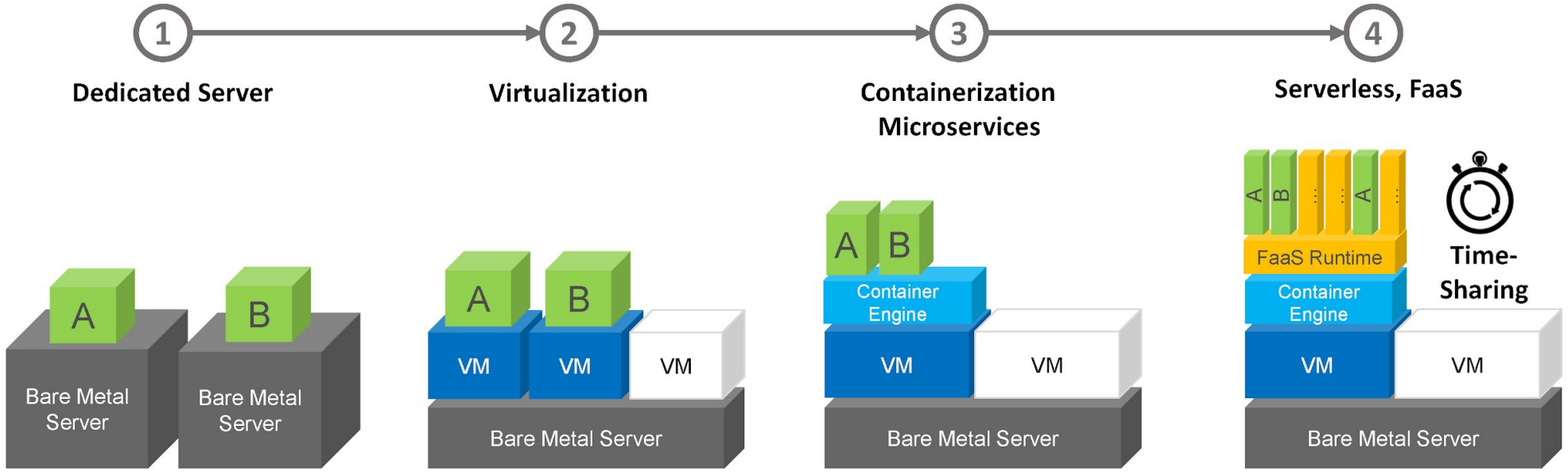
Charla SubT, Landing on an asteroid using computer vision y oportunidades en NASA Jet Propulsion Laboratory



Tutorial de AWS

4

Tutorial de AWS



In case of dedicated servers applications (A, B) are deployed on physical servers. In consequence, the servers are often over dimensioned and have inefficient utilization rates.

Machine virtualization is mainly used to consolidate and isolate applications on virtual machine instead of dedicated servers. This increases the application density on bare metal servers but the virtual machine images (deployment unit) are very large.

To pragmatically operate more than one application per virtual machine, containerization established as a trend. A container starts faster than a virtual machine and shares the operating system with other containers, thus reducing deployment unit sizes and increasing application density per virtual machine.

But a container still requests a share of CPU, memory, and storage – even if the provided service is hardly requested. It is more resource efficient, if services would consume resources only if there are incoming requests. FaaS runtime environments enable that services can timeshare a host. However, this involves to follow a serverless architecture style.

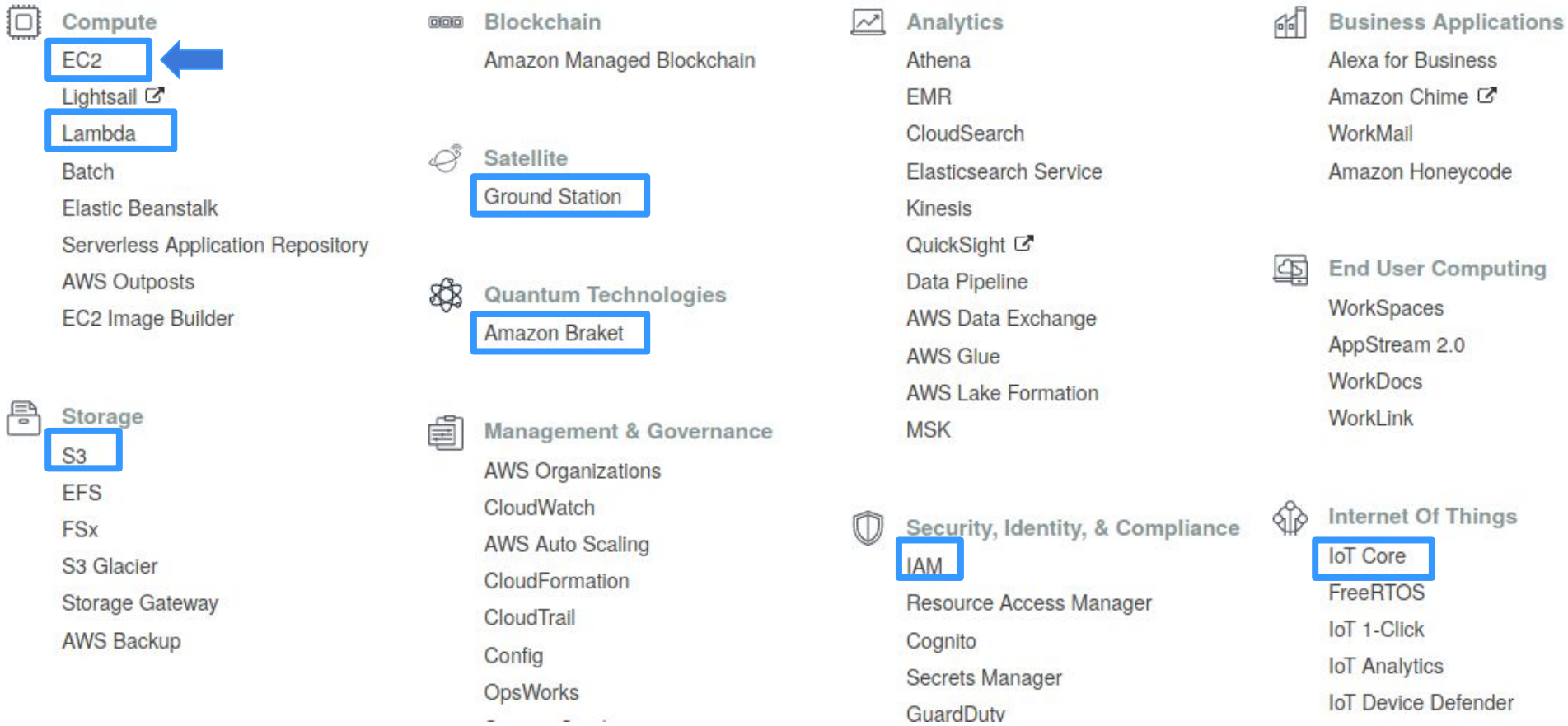
Tutorial de AWS

Amazon Web Services

- Costos (servidores, administración, personal)
- Escalabilidad
- Seguridad
- Resiliencia
- Multitud de servicios

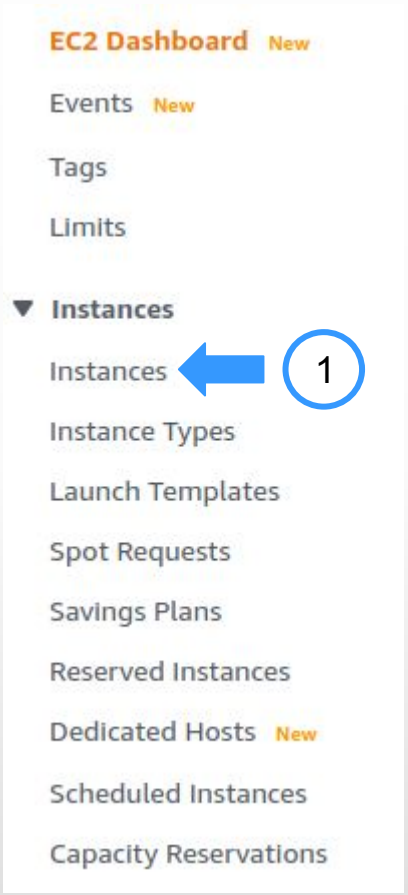
Tutorial de AWS

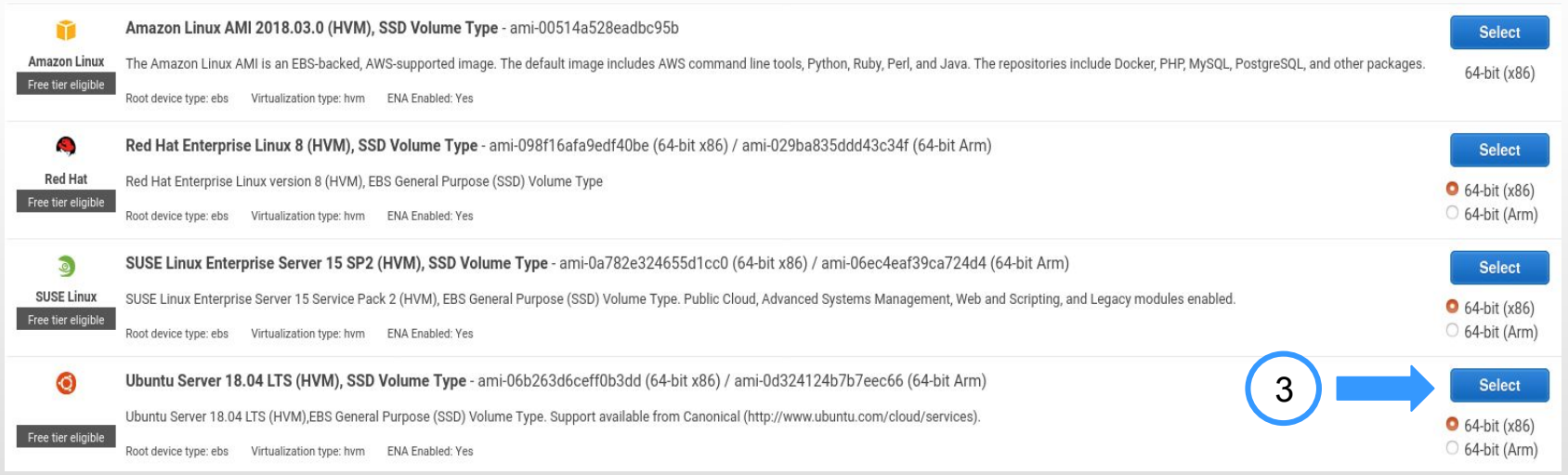
Tutorial de AWS



UbiHPC

Tutorial de AWS





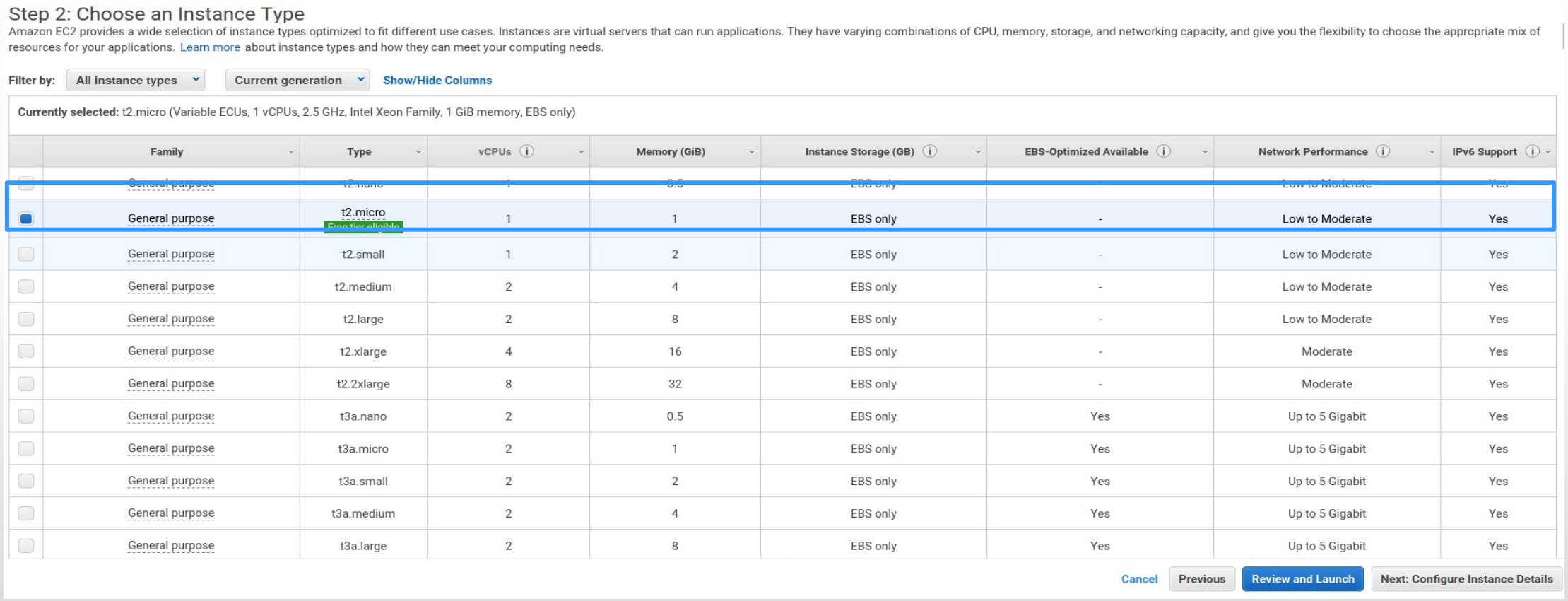
UbiHPC

Tutorial de AWS

	vCPU	ECU	Memory (GiB)	Instance Storage (GB)	Linux/UNIX Usage
General Purpose - Current Generation					
a1.medium	1	N/A	2 GiB	EBS Only	$0.0255 per Hour
a1.large	2	N/A	4 GiB	EBS Only	$0.051 per Hour
a1.xlarge	4	N/A	8 GiB	EBS Only	$0.102 per Hour
a1.2xlarge	8	N/A	16 GiB	EBS Only	$0.204 per Hour
a1.4xlarge	16	N/A	32 GiB	EBS Only	$0.408 per Hour
a1.metal	16	N/A	32 GiB	EBS Only	$0.408 per Hour
t4g.nano	2	N/A	0.5 GiB	EBS Only	$0.0042 per Hour
t4g.micro	2	N/A	1 GiB	EBS Only	$0.0084 per Hour
t4g.small	2	N/A	2 GiB	EBS Only	$0.0168 per Hour
t4g.medium	2	N/A	4 GiB	EBS Only	$0.0336 per Hour
t4g.large	2	N/A	8 GiB	EBS Only	$0.0672 per Hour
t4g.xlarge	4	N/A	16 GiB	EBS Only	$0.1344 per Hour
t4g.2xlarge	8	N/A	32 GiB	EBS Only	$0.2688 per Hour
t3.nano	2	Variable	0.5 GiB	EBS Only	$0.0052 per Hour
t3.micro	2	Variable	1 GiB	EBS Only	$0.0104 per Hour
t3.small	2	Variable	2 GiB	EBS Only	$0.0208 per Hour
t3.medium	2	Variable	4 GiB	EBS Only	$0.0416 per Hour

GPU Instances - Current Generation					
p3.2xlarge	8	31	61 GiB	EBS Only	$3.06 per Hour
p3.8xlarge	32	97	244 GiB	EBS Only	$12.24 per Hour
p3.16xlarge	64	201	488 GiB	EBS Only	$24.48 per Hour
p2.xlarge	4	16	61 GiB	EBS Only	$0.90 per Hour
p2.8xlarge	32	97	488 GiB	EBS Only	$7.20 per Hour
p2.16xlarge	64	201	732 GiB	EBS Only	$14.40 per Hour
g4dn.xlarge	4	N/A	16 GiB	125 GB NVMe SSD	$0.526 per Hour
g4dn.2xlarge	8	N/A	32 GiB	225 GB NVMe SSD	$0.752 per Hour
g4dn.4xlarge	16	N/A	64 GiB	225 GB NVMe SSD	$1.204 per Hour
g4dn.8xlarge	32	N/A	128 GiB	900 GB NVMe SSD	$2.176 per Hour
g4dn.12xlarge	48	N/A	192 GiB	900 GB NVMe SSD	$3.912 per Hour
g4dn.16xlarge	64	N/A	256 GiB	900 GB NVMe SSD	$4.352 per Hour
g3.4xlarge	16	58	122 GiB	EBS Only	$1.14 per Hour
g3.8xlarge	32	97	244 GiB	EBS Only	$2.28 per Hour
g3.16xlarge	64	201	488 GiB	EBS Only	$4.56 per Hour
g3s.xlarge	4	13	30.5 GiB	EBS Only	$0.75 per Hour

Tutorial de AWS



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.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)

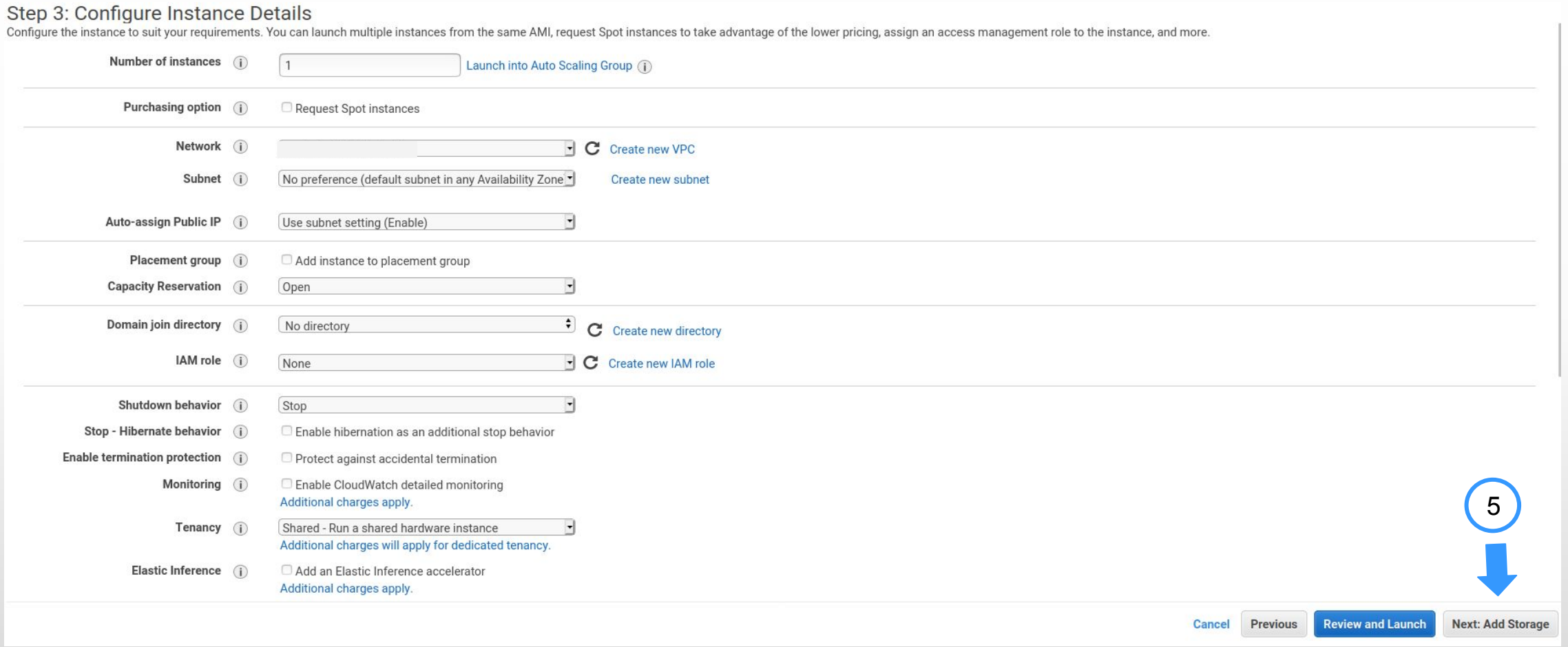
	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
	General purpose	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
	General purpose	t2.small	1	2	EBS only	-	Low to Moderate	Yes
	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
	General purpose	t2.large	2	8	EBS only	-	Low to Moderate	Yes
	General purpose	t2.xlarge	4	16	EBS only	-	Moderate	Yes
	General purpose	t2.2xlarge	8	32	EBS only	-	Moderate	Yes
	General purpose	t3a.nano	2	0.5	EBS only	Yes	Up to 5 Gigabit	Yes
	General purpose	t3a.micro	2	1	EBS only	Yes	Up to 5 Gigabit	Yes
	General purpose	t3a.small	2	2	EBS only	Yes	Up to 5 Gigabit	Yes
	General purpose	t3a.medium	2	4	EBS only	Yes	Up to 5 Gigabit	Yes
	General purpose	t3a.large	2	8	EBS only	Yes	Up to 5 Gigabit	Yes

Cancel | Previous | Review and Launch | Next: Configure Instance Details

4

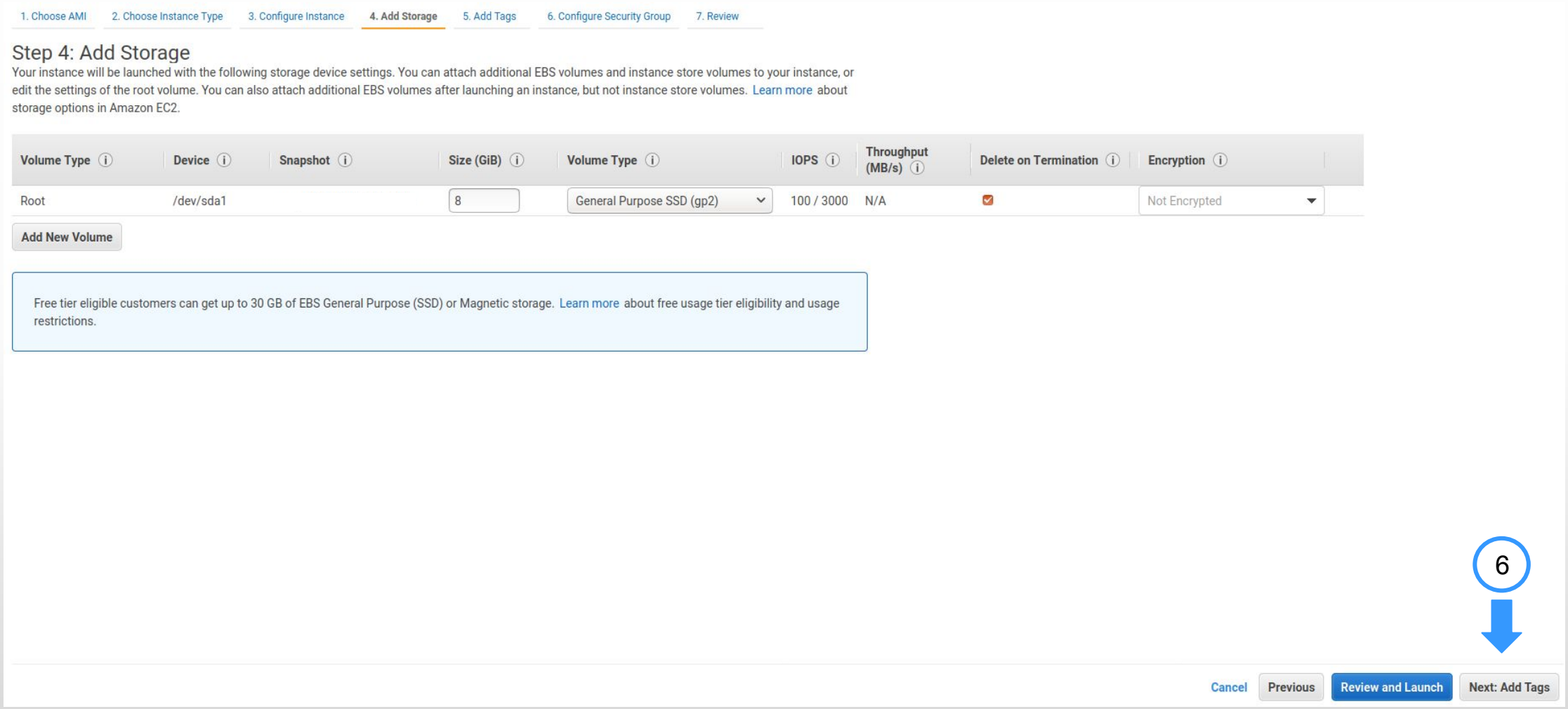
UbiHPC

Tutorial de AWS

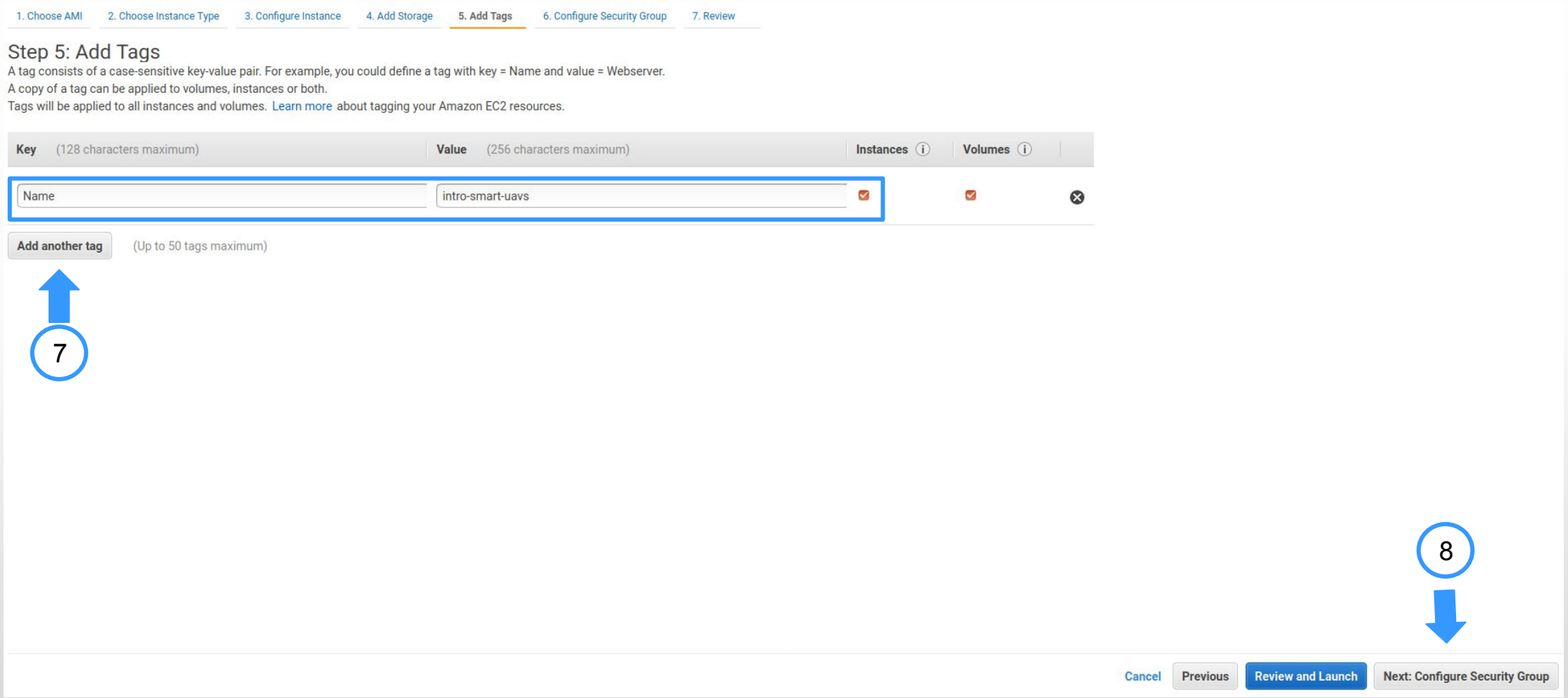


UbiHPC

Tutorial de AWS



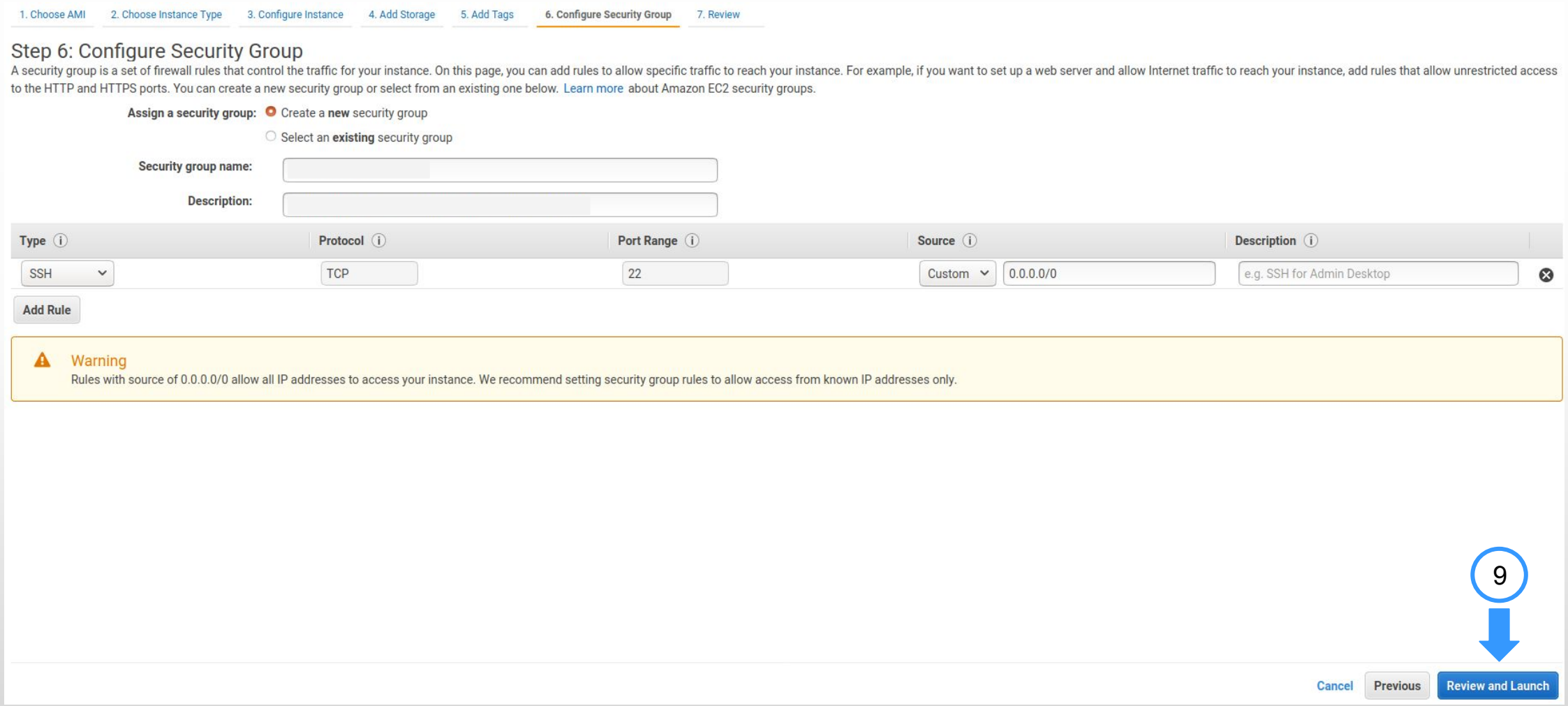
Tutorial de AWS



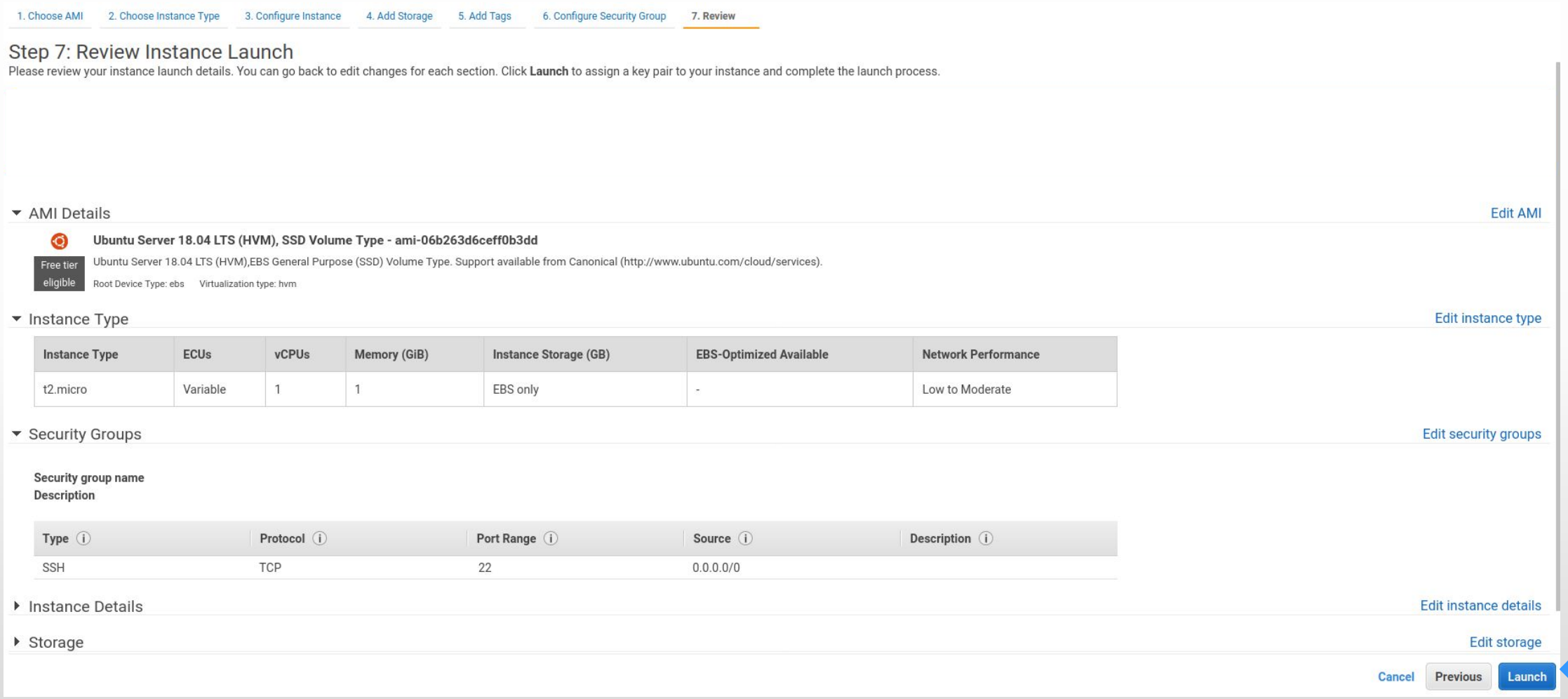
UbiHPC

UbiHPC

Tutorial de AWS

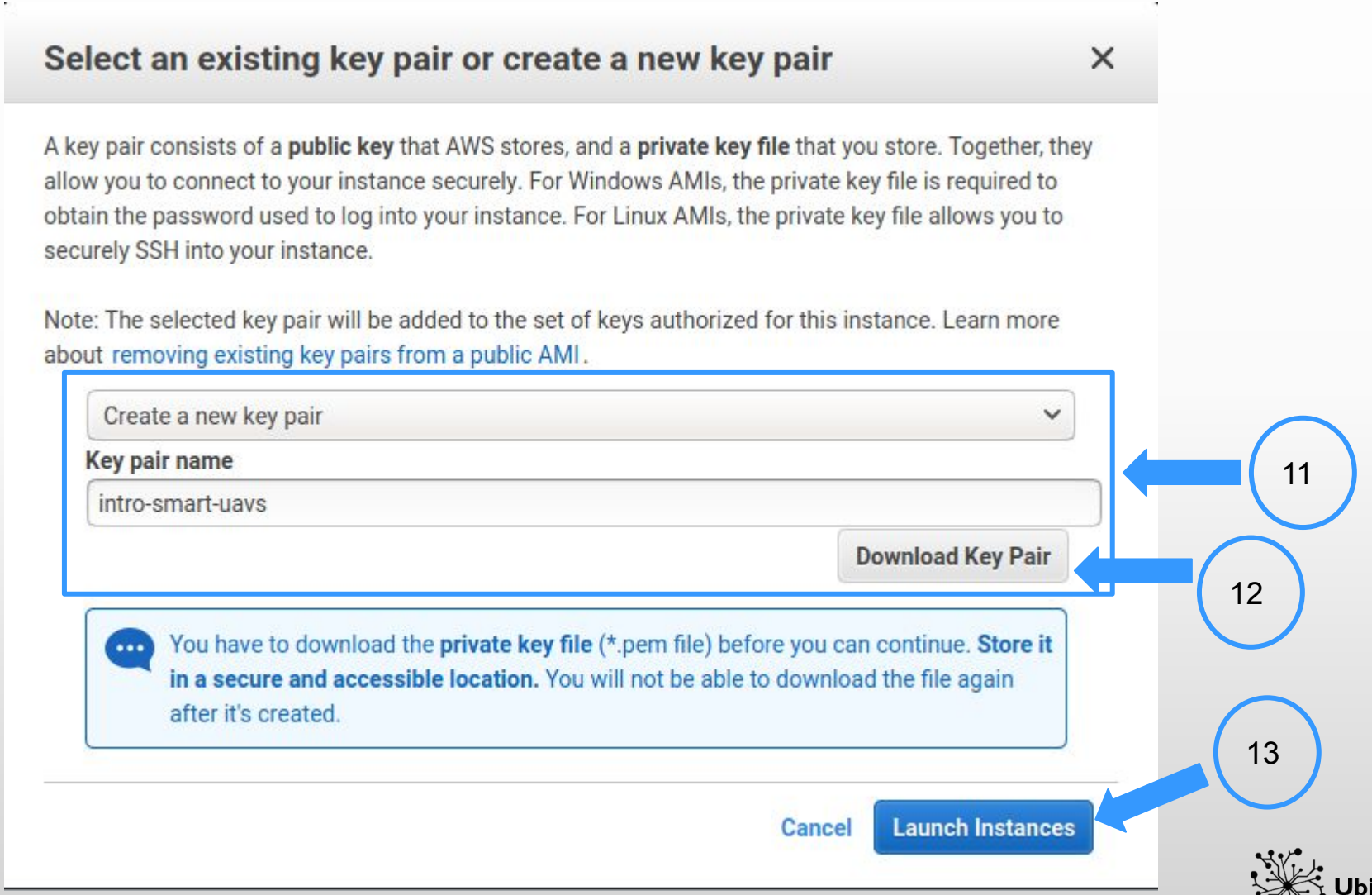


Tutorial de AWS

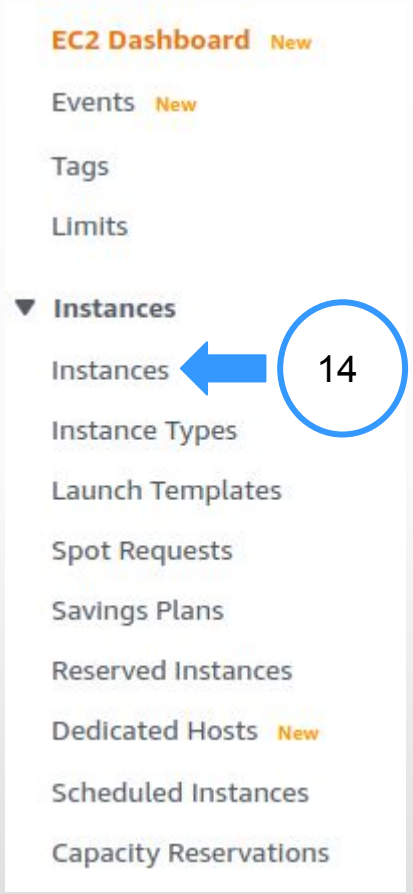


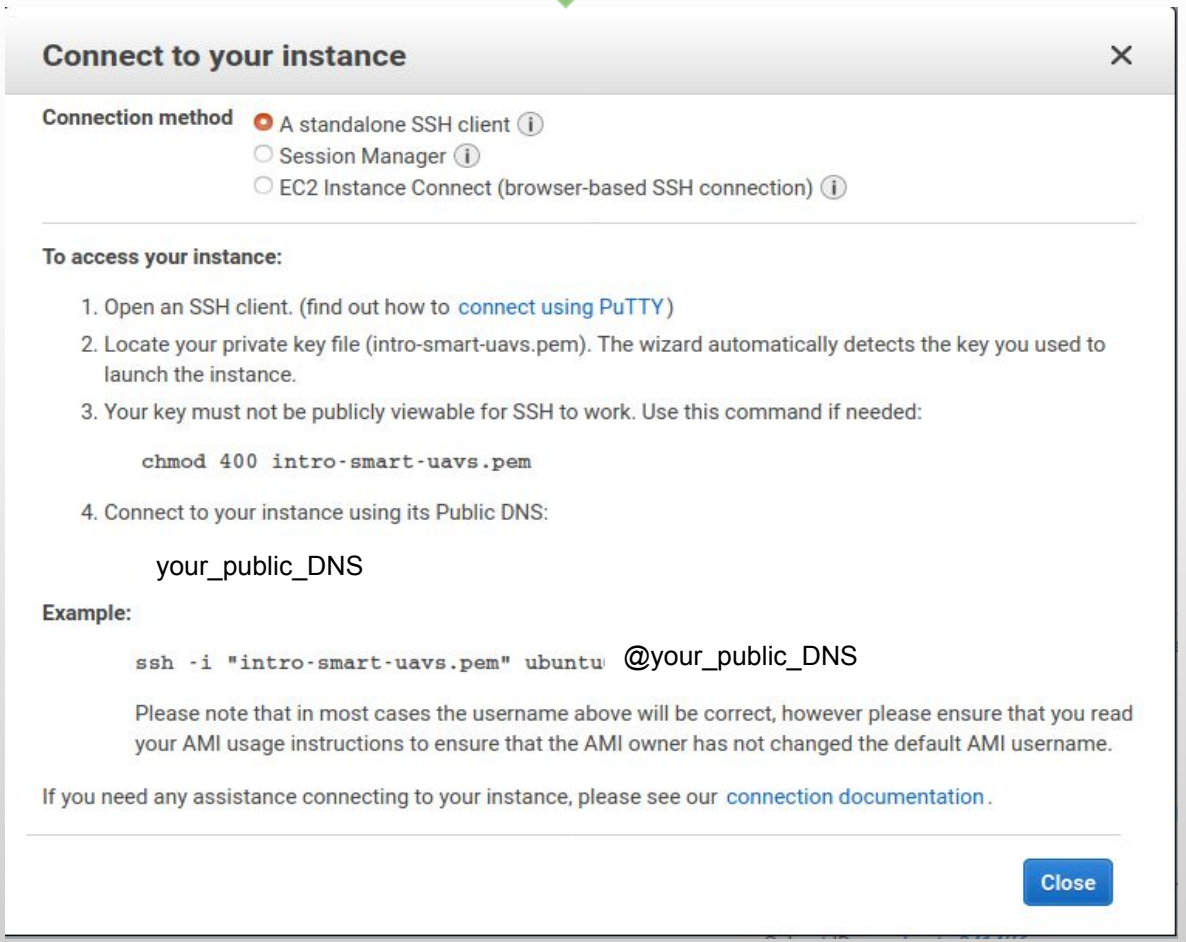
UbiHPC

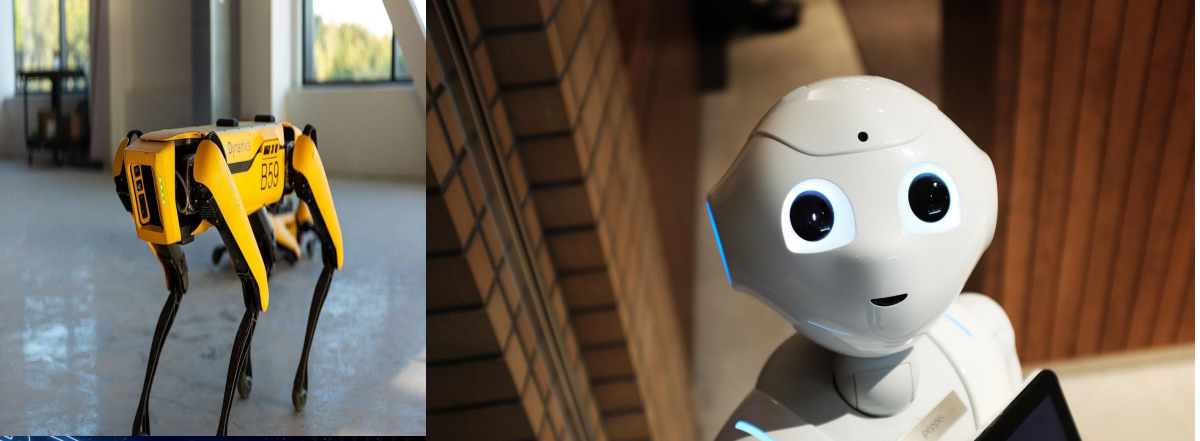
Tutorial de AWS



Tutorial de AWS










High Performance Robotic Computing

UbiHPC

High Performance Computing
a.k.a. Supercomputing

¿Para qué se usa la supercomputación?

La supercomputación está en todos lados

Porqué usar la supercomputación?

Simulación

Diseño de aeronaves
Computational Fluid Dynamics (CFD)
Orígenes del universo
Inteligencia Artificial
Cyber-heart living heart (25M de variables)
Is Cancer written in our DNA?
Boeing lighter 787
Disney-Pixar
Avatar
Facebook/email keyword analysis

Demasiada complejidad computacional

Incluso un PC poderoso

Miles de años para obtener resultados

Supercomputación

Qué es la supercomputación (HPC)?

Distribuir

Paralelizar

Concurrir

Optimizar

Supercomputador / cluster

Computación en Grid

BOINC

Computación oportunista

GPGPU

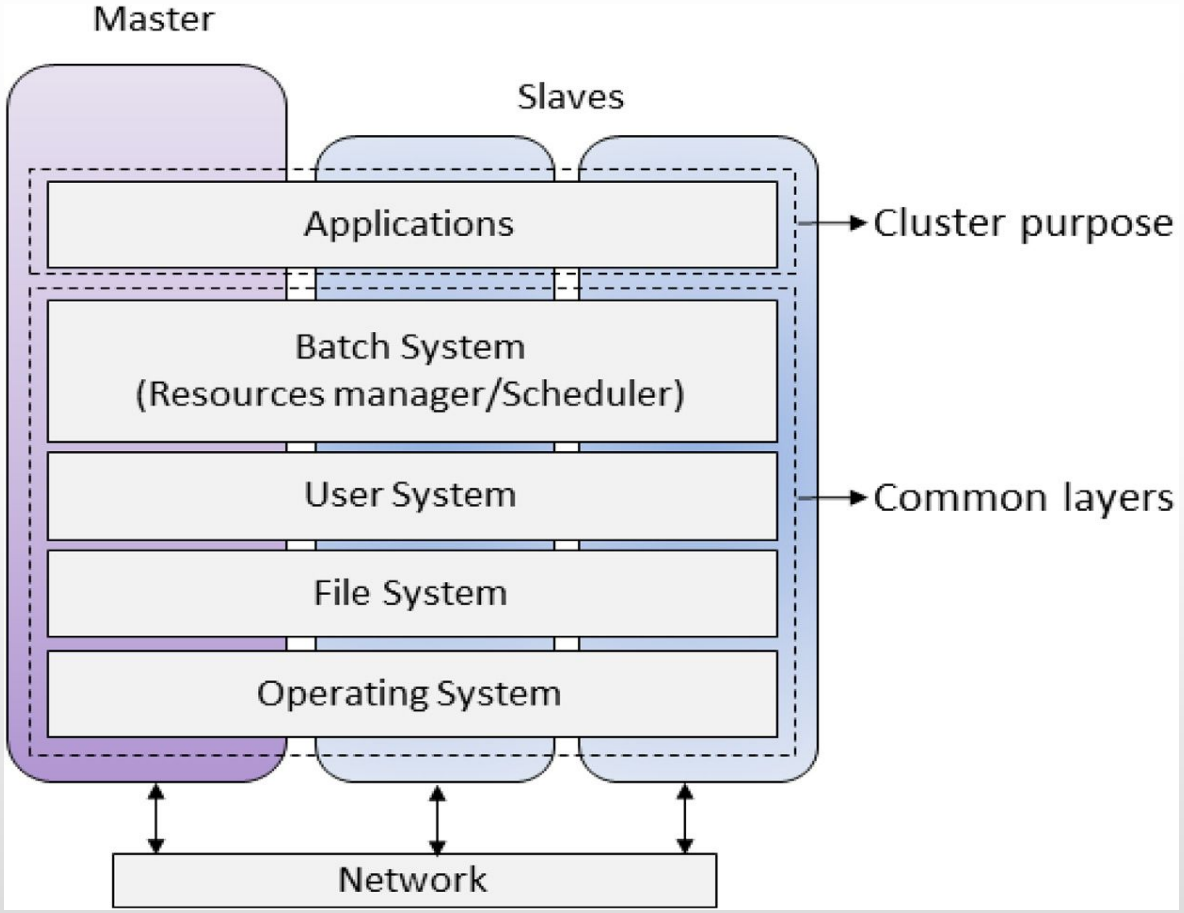
MPI

NVIDIA CUDA

HPC = High Performance Computing

23

Supercomputador o cluster HPC

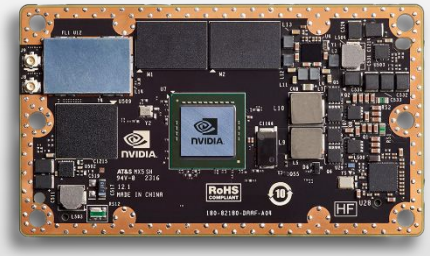


Supercomputador

Arquitectura software

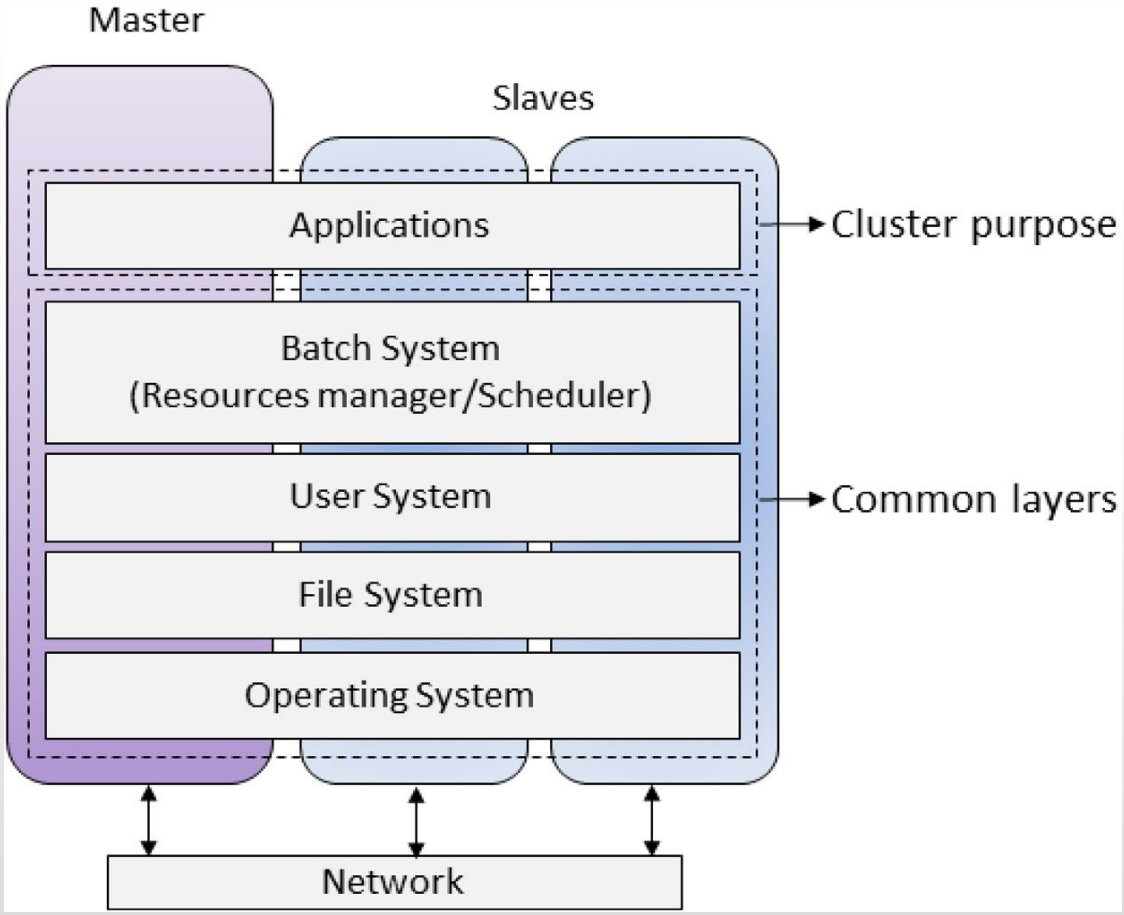
UbiHPC

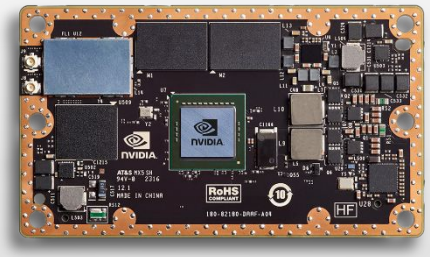
High Performance Robotic Computing



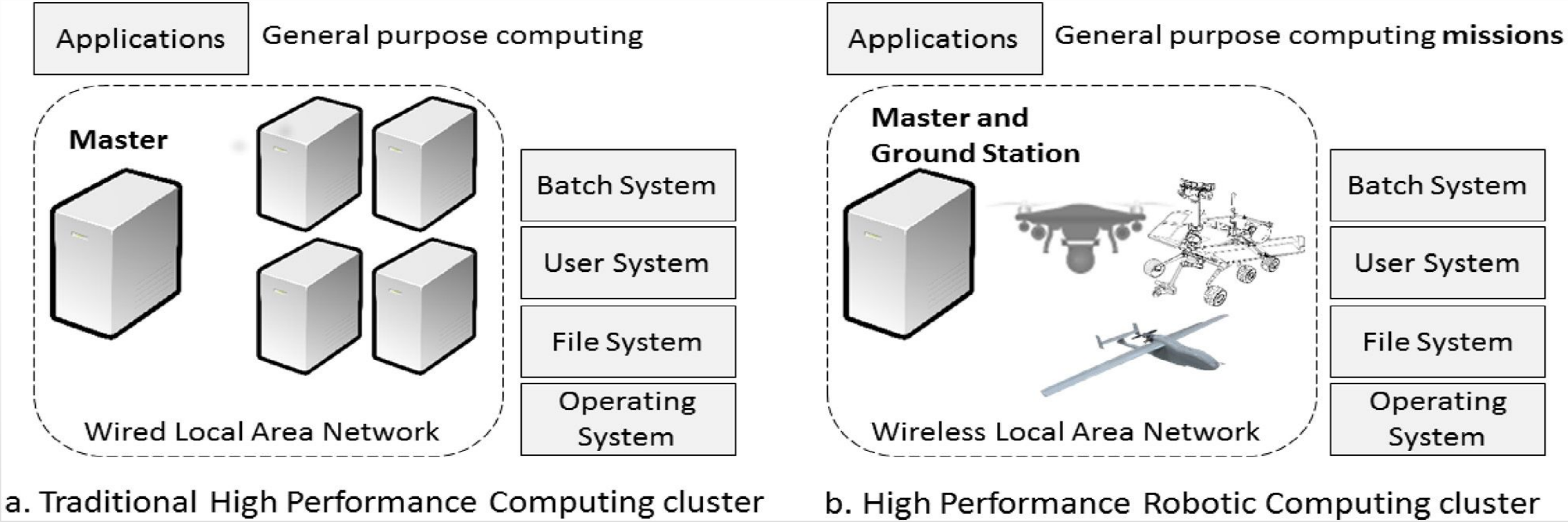
UbiHPC

High Performance Robotic Computing

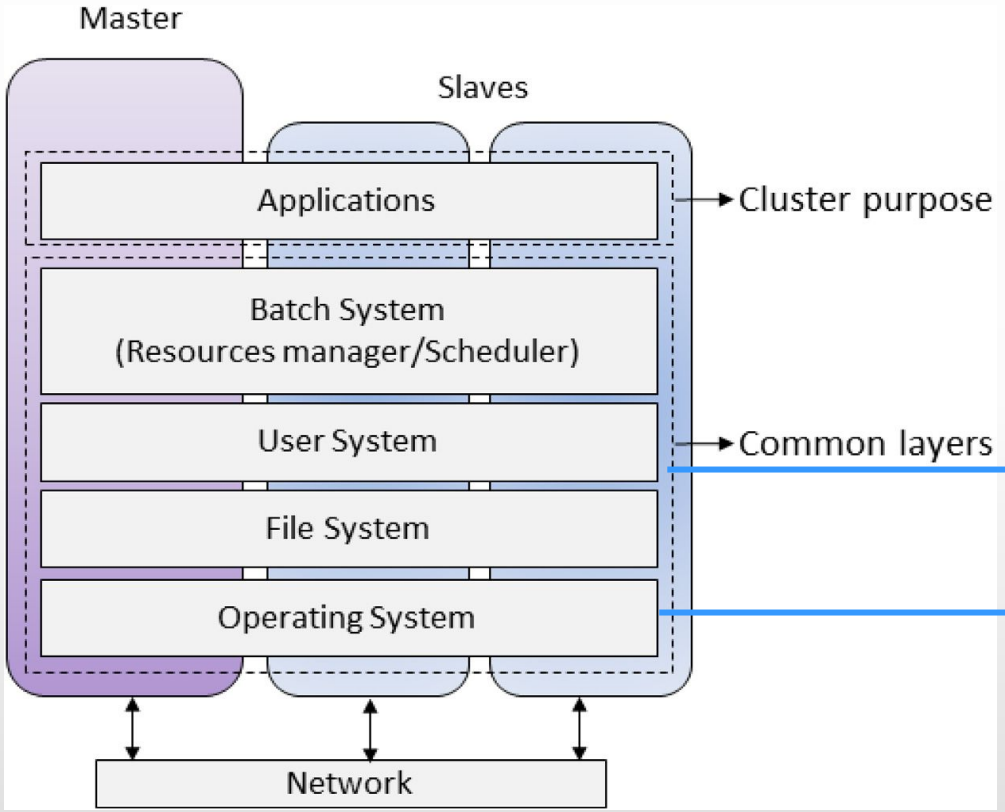




High Performance Robotic Computing



High Performance Robotic Computing



General configuration

Software Installation (as root or with sudo)

```
apt-get install openssh-server nfs-common
apt-get install python-mpi4py openmpi-common openmpi-bin openmpi-doc libopenmpi-dev
```

Add hosts to /etc/hosts , one line per host (as root or with sudo)

```
<IP>  <FQDN>  <Hostname>
```

Create home directory for HPRC cluster's users (as root or with sudo)

```
mkdir -p /export/home
```

Create user (as root or with sudo)

```
useradd -u <user_ID> -m -d /export/home/<username> -s /bin/bash <username>
passwd <username>
su <username>
```

Only in the master node: Generate SSH keys for and configure passwordless authentication

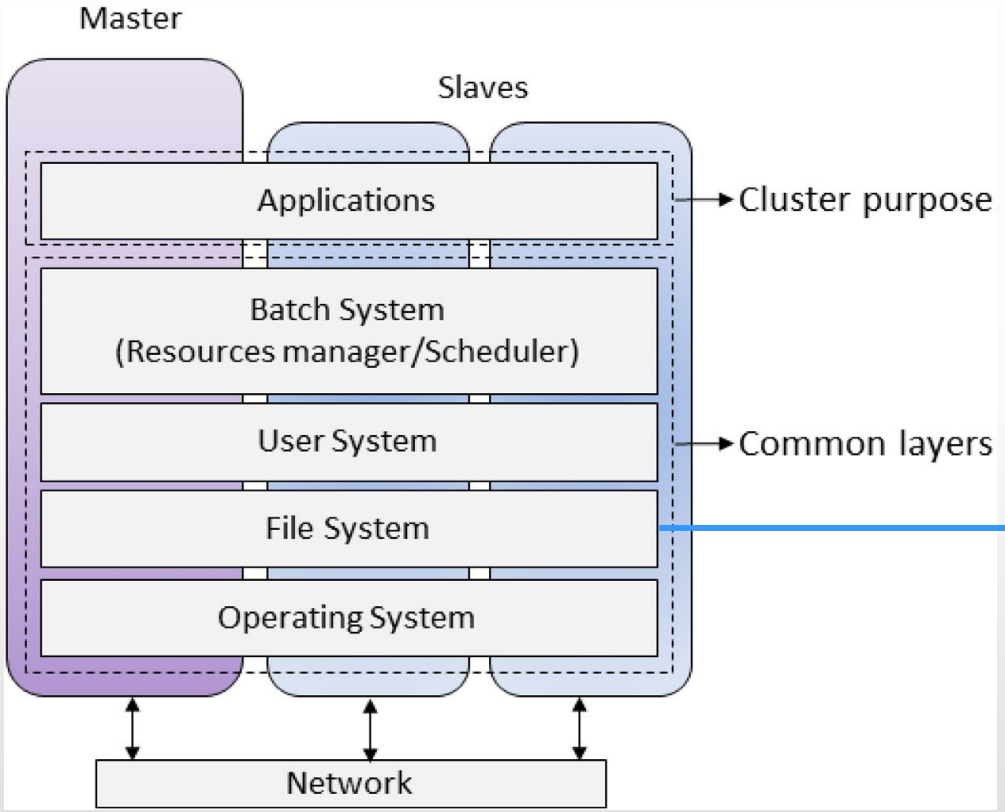
```
ssh-keygen -t rsa
cd ~/.ssh
cat id_rsa.pub >> authorized_keys
chmod 600 authorized_keys
```

https://github.com/leonardocfor/HPRC-Cluster-deployment

Previous
Next

UbiHPC

High Performance Robotic Computing



File system

File system Installation: In a HPRC cluster, the file system can be any technology traditionally used in Supercomputing. For a simple cluster, Network File System (NFS) will suffy. For this example, the folder /export/home will be shared from the master node. Such folder corresponds to the user space for this configuration

Master node

```
apt-get install nfs-kernel-server
```

Edit the file */etc/exports* and add the following lines:

```
/export/home <Network/Mask>(rw,no_root_squash)
```

Execute the following commands

```
service rpcbind restart
/etc/init.d/nfs-kernel-server restart
exportfs
```

Slave nodes

Edit the file */etc/fstab* and add the following lines:

```
<SERVER_IP>:/export/home /export/home    nfs    defaults,proto=tcp,port=2049    0 0
```

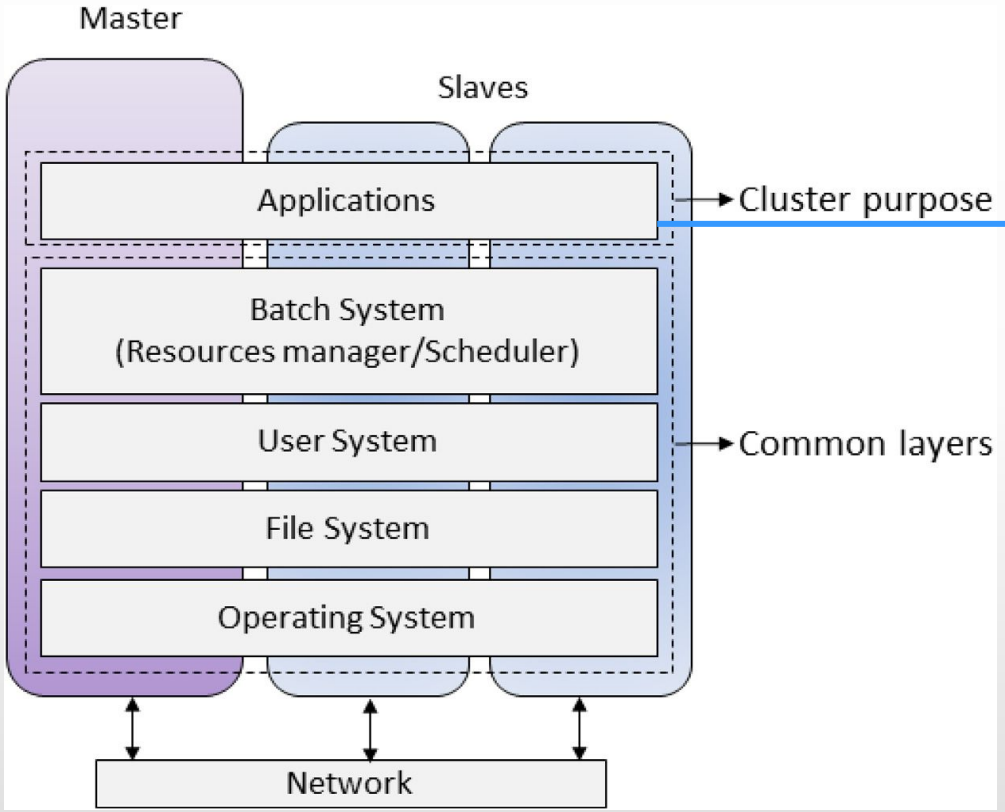
Mount NFS shared folder

```
mount <SERVER_IP>:/export/home /export/home
```

https://github.com/leonardocfor/HPRC-Cluster-deployment

UbiHPC

High Performance Robotic Computing



Traditional supercomputing (HPC) examples

Hello world with mpi4py

PI computation with mpi4py

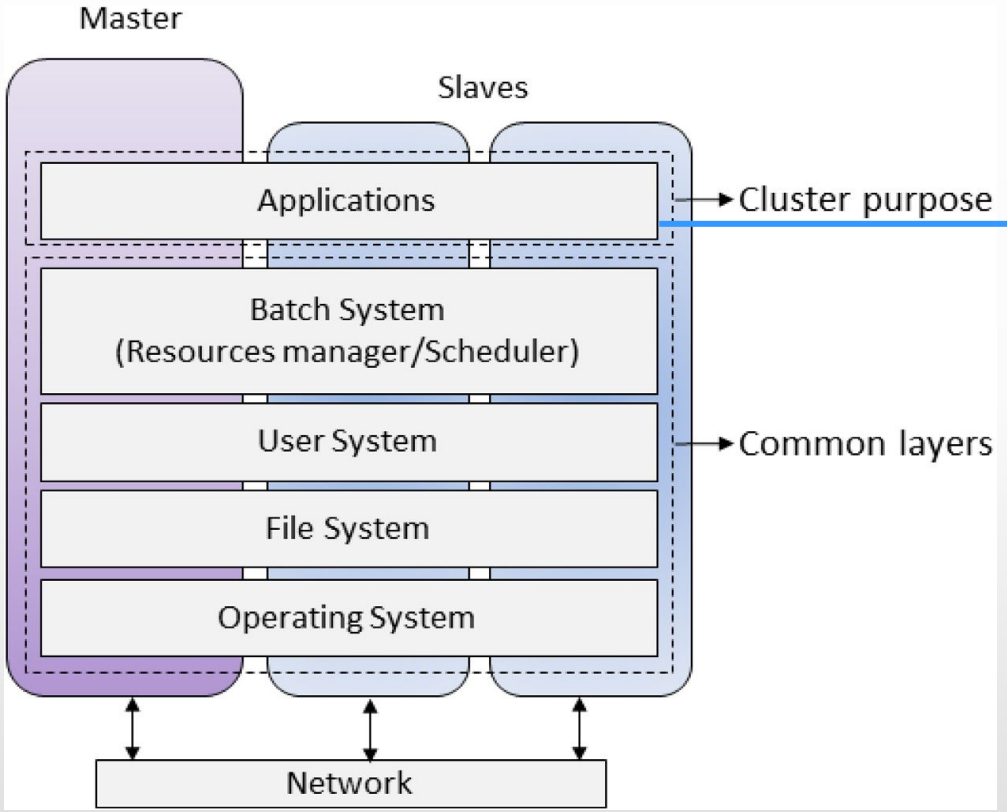
To execute these examples use the following command:

```
mpirun -np <X> --hostfile <hostfile> python <pythonSoftware>

where,

X = Quantity of MPI parallel processes
<pythonSoftware> The corresponding software
<hostfile> is a text file with the hostnames of the computers in your cluster, e.g.
  <Hostname1>
  <Hostname2>
  <Hostname3>
```

https://github.com/leonardocfor/HPRC-Cluster-deployment

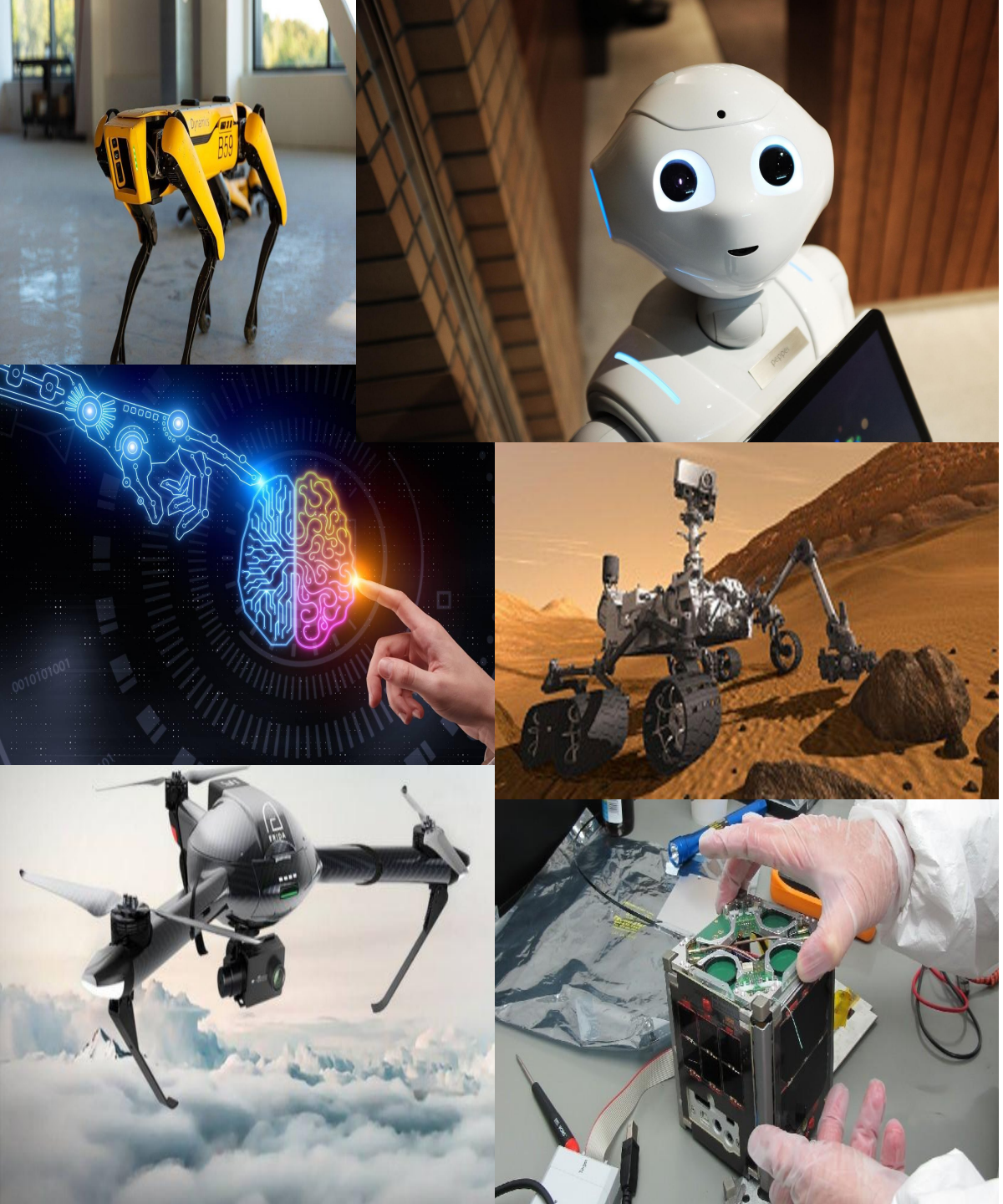
UbiHPC

High Performance Robotic Computing




```python
#!/usr/bin/env python
"""
Parallel Hello World
"""

from mpi4py import MPI
import sys

size = MPI.COMM_WORLD.Get_size()
rank = MPI.COMM_WORLD.Get_rank()
name = MPI.Get_processor_name()

sys.stdout.write(
    "Hello, World! I am process %d of %d on %s.\n"
    % (rank, size, name))
```

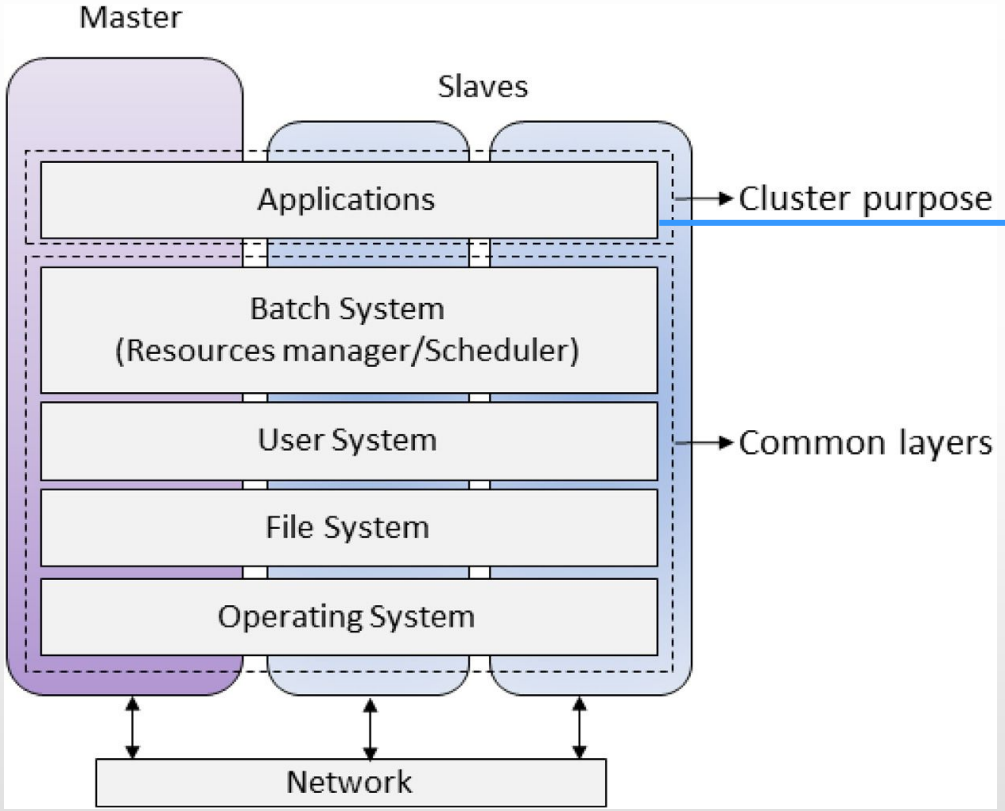
```
mpirun -np <X> --hostfile <hostfile> python <pythonSoftware>

where,

X = Quantity of MPI parallel processes
<pythonSoftware> The corresponding software
<hostfile> is a text file with the hostnames of the computers in your cluster, e.g.
  <Hostname1>
  <Hostname2>
  <Hostname3>
```



Desarrollo de software para cluster de UAVs

UbiHPC

Desarrollo de software para cluster de UAVs



Usage

To execute the software, launch the following command:

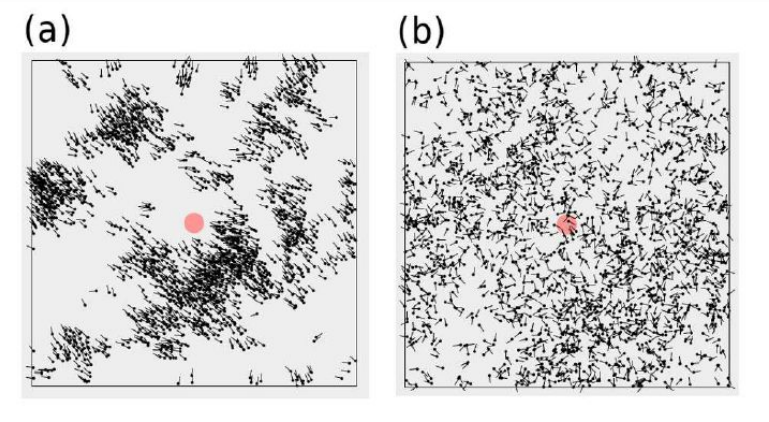
```
mpirun -np X --rankfile <rankfile> python parallelVicsek.py -v <vehicleType> -s <simTime> -t <telemet

Where

X = Quantity of vehicles
vehicleType = <drone, rover>
simTime = Simulation time in seconds
telemetryFolder = Shared folder in the HPRC cluster
```

Finally the rankfile is used to execute a single MPI process per vehicle (HPRC node). An example is the following:

```
rank 0=<hostname_vehicle_1> slot=0
rank 1=<hostname_vehicle_2> slot=0
rank 2=<hostname_vehicle_3> slot=0
rank 3=<hostname_vehicle_4> slot=0
rank 4=<hostname_vehicle_5> slot=0
```


UbiHPC

Desarrollo de software para cluster de UAVs



Modelo Vicsek

1. Cada entidad (por ejemplo, pájaro, UAV) vuela con velocidad constante y dirección (rumbo) igual al promedio de las direcciones de sus vecinos. Una entidad *i* se considera vecina de la entidad *j* si en el tiempo *t*, *i* y *j* están a una distancia menor o igual que un radio seleccionado.

2. El ruido en el sistema. Se consideran dos tipos de ruido: *extrínseco* e *intrínseco*.
   a. Extrínseco: Se refiere al ruido causado por el medio ambiente. En los sistemas biológicos, podría relacionarse con las entidades, a las que les podría suceder, que en determinado momento, no pudieren determinar correctamente si otra entidad es un vecino. En sistemas artificiales, el ruido extrínseco puede estar relacionado con problemas de precisión del GPS.
   b. Intrínseco: Se refiere a la decisión de las entidades de moverse en cierta dirección incluso si comprenden completamente la dirección de sus vecinos. En sistemas artificiales, esto podría relacionarse con errores de cómputo en tiempo de ejecución.

Desarrollo de software para cluster de UAVs

Ingresar al maestro usando el usuario (<usuario_cluster> que crearon para el despliegue del cluster

1. Clonar el repositorio intro_to_smart_uavs:

```
cd
git clone https://github.com/<usuario_git>/intro_to_smart_uavs.git
cd intro_to_smart_uavs/session3
```

2. Crear directorio para almacenar el código fuente:

```
mkdir -p code
cd code
touch _init_.py
mkdir instructor_code my_code
touch instructor_code/_init_.py my_code/_init_.py
```

3. Clonar código multi-robot-vicsek

```
cd instructor_code
git clone https://github.com/leonardocfor/multi-robot-vicsek
```

Desarrollo de software para cluster de UAVs

4. Crear .gitignore

cd $HOME/intro_to_smart_uavs
nano .gitignore → session3/code/instructor_code

5. Configurar PATH y PYTHON_PATH

cd
nano .bashrc
Al final del archivo
export PYTHONPATH=$PYTHONPATH:$HOME/intro_to_smart_uavs/session3/code/my_code
export PATH=$PATH:/home/<usuario_sudo>/ardupilot/Tools/autotest
Cerrar archivo
source .bashrc

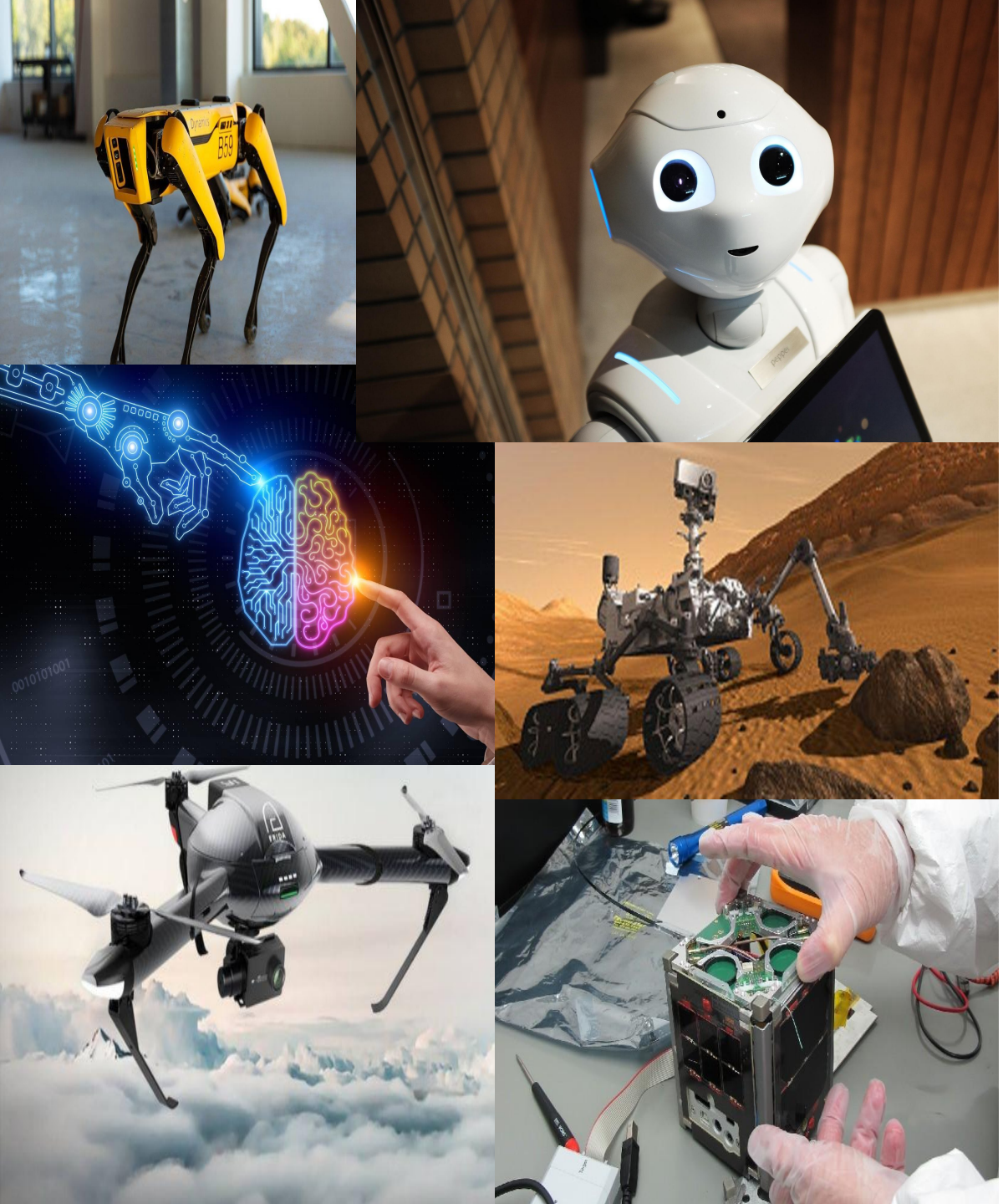
Desarrollo de software para cluster de UAVs

6. A little hack

sudo chown -R <usuario_cluster> /home/<usuario_sudo>/ardupilot
Partir terminator en dos ventanas
Ventana 1:
sim_vehicle.py -v ArduCopter --map --console --out 127.0.0.1:14551 --out 127.0.0.1:14552
Ventana 2:
python3
>>> from dronekit import connect, VehicleMode
>>> vehicle = connect('127.0.0.1:14550',wait_ready=True)
>>> vehicle.mode = VehicleMode('GUIDED')
>>> vehicle.armed = True; vehicle.simple_takeoff(30)

7. Code Along

cd $HOME/intro_to_smart_uavs/session3/code
atom .



Para la siguiente sección

Previous
Next

Para la siguiente sección

Instalar Jupyter notebook Windows: https://test-jupyter.readthedocs.io/en/latest/install.html
Instalar OpenCV python Windows: https://docs.opencv.org/master/d5/de5/tutorial_py_setup_in_windows.html

Gracias

Leonardo Camargo Forero, Ph.D

leonardo@ubihpc.com
www.ubihpc.com

UbiHPC