

OpenMPI: how to realize a cluster of Google virtual instances

20.10.2021

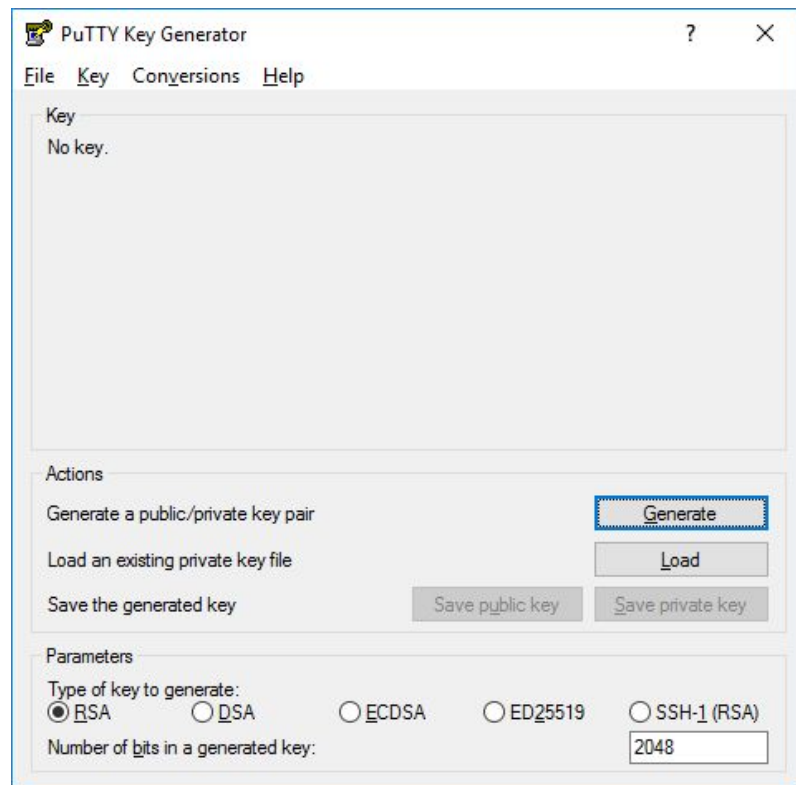
Luigi Santangelo (luigi.santangelo@unipv.it)

Before starting: creating a SSH key (using Linux)

- `mkdir myGoogleKey`
- `cd myGoogleKey`
- `ssh-keygen -t rsa -b 4096 -f ./id_rsa`
 - The system will create the private key and ask for protecting it using a password. Leave empty for no password. If provided, don't forget the password, it will be asked at login time
 - At the end, two files are created: `id_rsa` (the private key) and `id_rsa.pub` (the public key)
 - Keep safe both files as everybody could get access to your virtual instance

Before starting: creating a SSH key (using Win)


- download PuttyGen from <https://www.puttygen.com/>
- start the tool
- generate a RSA key
- save and keep safe public and private keys



Building a Virtual Instance (used as template)

- Log in <https://console.cloud.google.com> using your institutional email credentials
- Select Compute Engine > Virtual Instances
- Create a new instance having the following configuration:
 - name: node1
 - region: us-central1
 - cpu: 2
 - memory: 8GB

The screenshot displays the configuration page for a new virtual instance in the Google Cloud Platform console. The instance is named 'node1' and is permanent. It is located in the 'us-central1 (Iowa)' region, specifically in the 'us-central1-a' zone. The machine configuration is set to 'General-purpose' with 'E2' series and 'e2-standard-2' machine type, which provides 2 vCPUs and 8 GB of memory. The configuration is summarized in a table at the bottom.

	vCPU	Memory	GPUs
	2	8 GB	-

Building a Virtual Instance (used as template)

- Create a new instance having the following configuration:
 - OS: centos
 - Version: 8
 - Boot Disk: Standard
 - Size: 50GB

Boot disk

Select an image or snapshot to create a boot disk; or attach an existing disk. Can't find what

[Public images](#) [Custom images](#) [Snapshots](#) [Existing disks](#)

Operating system
CentOS

Version
CentOS 8

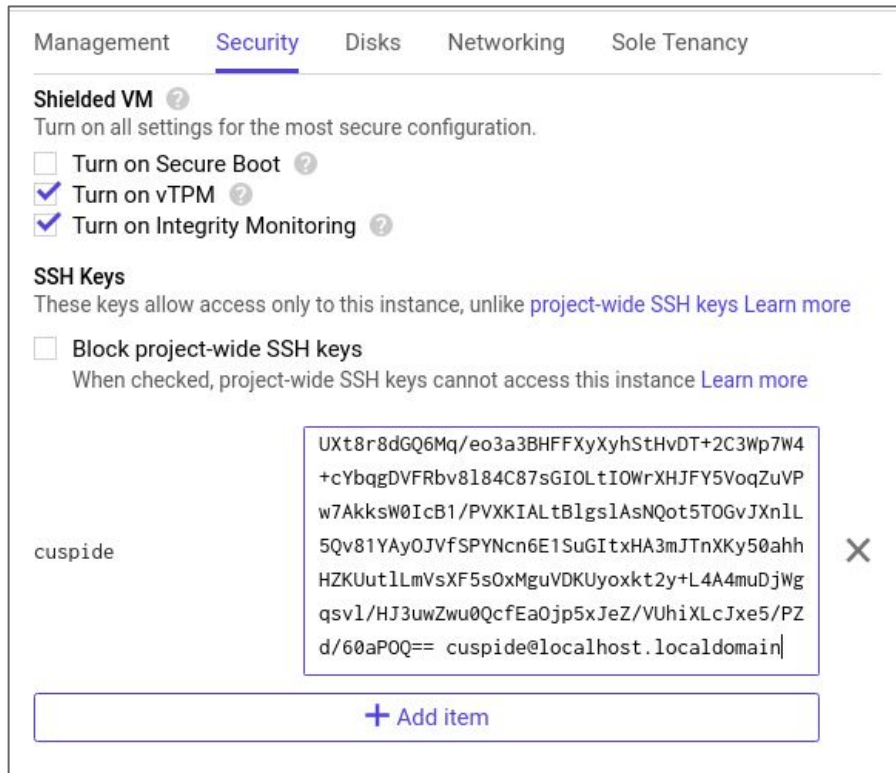
x86_64 built on 20201014, supports Shielded VM features

Boot disk type [?](#) **Size (GB)** [?](#)

Standard persistent disk 50

Building a Virtual Instance (used as template)

- Using a text editor, open the public key created before (id_rsa.pub), copy the content and paste it into the right field (Security Tab)
- Take a look at the username assigned to the key (which is the same username who created the key)
- Let's select the Create button to build the virtual instance.
- The VI is started up straightaway.



Management **Security** Disks Networking Sole Tenancy

Shielded VM [?](#)
Turn on all settings for the most secure configuration.

Turn on Secure Boot [?](#)
 Turn on vTPM [?](#)
 Turn on Integrity Monitoring [?](#)

SSH Keys
These keys allow access only to this instance, unlike [project-wide SSH keys](#) [Learn more](#)

Block project-wide SSH keys
When checked, project-wide SSH keys cannot access this instance [Learn more](#)

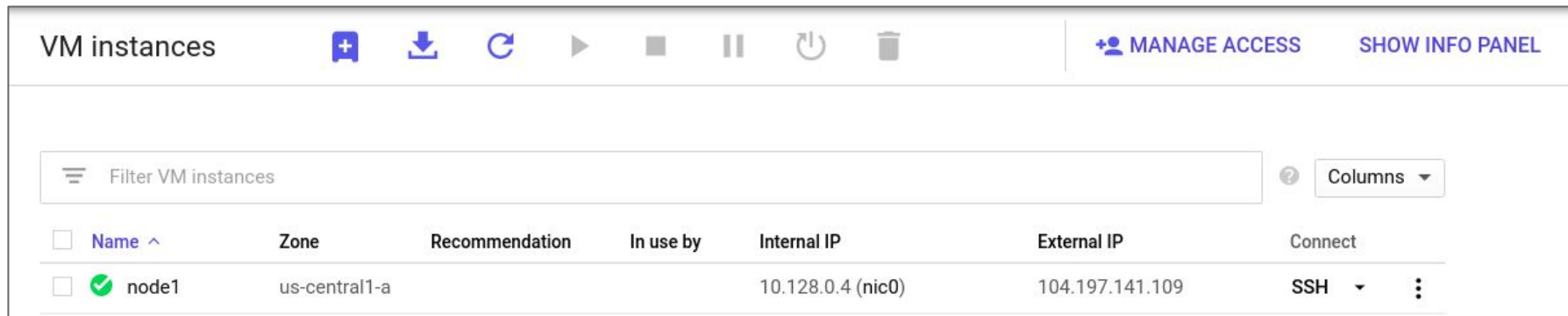
cuspid

```
UXt8r8dGQ6Mq/eo3a3BHFFXyXyhStHvDT+2C3Wp7W4
+cYbqgDVFRbv8l84C87sGIOLtIOWrXHJFY5VoqZuVP
w7Akksw0IcB1/PVXKIALtB1gs1AsNQt5T0GvJXn1L
5Qv81YAy0JVfSPYNcn6E1SuGItxHA3mJTnXKy50ahh
HZKUut1LmVsXF5s0xMguVDKUyoxkt2y+L4A4muDjWg
qsv1/HJ3uwZwu0QcfEa0jpxJeZ/VUhiXLCJxe5/PZ
d/60aPQQ== cuspid@localhost.localdomain
```

[+ Add item](#)

Getting an access to the virtual instance

- Using the Dashboard, let's take a look to the virtual instance. The green button means it is running
- The Virtual Instance is assigned to an external IP. Take note of that and keep in mind that it is going to stay the same as long as the virtual instance is left running. After that, the address might change



VM instances

[+](#) [↓](#) [↺](#) [▶](#) [■](#) [⏸](#) [🔄](#) [🗑](#) [+ MANAGE ACCESS](#) [SHOW INFO PANEL](#)

Filter VM instances Columns ▾

<input type="checkbox"/> Name ^	Zone	Recommendation	In use by	Internal IP	External IP	Connect
<input type="checkbox"/> ✔ node1	us-central1-a			10.128.0.4 (nic0)	104.197.141.109	SSH ▾ ⋮

Getting an access to the virtual instance

- Using your shell, run the following command:
 - `ssh -l cuspide -i ./id_rsa 104.197.141.109`
- Where:
 - `cuspide`: is the username showed in the security section
 - `id_rsa`: is the name of the private key created at the beginning
 - `104.197.141.109`: the is virtual instance IP address showed by the dashboard
- If everything went well, you are inside your remote virtual instance. You can see that the prompt is different as it is something similar to `cuspide@node1`

Download and install OpenMPI

- `sudo su`
- `yum install wget`
- `yum install perl`
- `yum install gcc`
- `yum install gcc-c++`
- `yum install make`
- `mkdir /usr/local/openMPI`
- `cd ~`
- `mkdir openMPI`
- `cd openMPI`
- `wget https://download.open-mpi.org/release/open-mpi/v4.0/openmpi-4.0.5.tar.gz`
- `tar -xvzf openmpi-4.0.5.tar.gz`

Download and install OpenMPI

- `cd openmpi-4.0.5`
- `mkdir build`
- `cd build`
- `../configure --prefix=/usr/local/openMPI`
- `make all install`
- `exit` (getting back to the non-admin user)
- `vi ~/.bashrc`
 - `export PATH=$PATH:/usr/local/openMPI/bin`
- `ssh-keygen -t rsa -b 4096`
- copy public key into `.ssh/authorized_keys`

The first program

```
#include <stdio.h>
#include <string.h>
#include <mpi.h>

const int MAX_STRING = 100;

int main(void)
{
    char greeting[MAX_STRING];
    int comm_sz;    /* Numero di processi */
    int my_rank;   /* Rango dei processi */
    int q = 0;

    MPI_Init(NULL, NULL);
    MPI_Comm_size(MPI_COMM_WORLD, &comm_sz);
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);

    if (my_rank != 0) {
        sprintf(greeting, "0- Greetings from process %d of %d!", my_rank, comm_sz);
        printf("Prima dell'invio: %d\n", my_rank);
        MPI_Send(greeting, strlen(greeting)+1, MPI_CHAR, 0, 0, MPI_COMM_WORLD);
        printf("Dopo l'invio: %d\n", my_rank);
    } else {
        printf("A - Greetings from process %d of %d!\n", my_rank, comm_sz);
        for (q = 1; q < comm_sz; q++) {
            MPI_Recv(greeting, MAX_STRING, MPI_CHAR, q, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
            printf("B - %s\n", greeting);
        }
    }
    MPI_Finalize();
    return 0;
}
```

Compiling and running the first application

- vi hostfile
 - localhost slots=4
- mpicc 01.c -o 01.o
- mpirun --hostfile hostfile -np 4 01.o

```
[cuspide@node1 srcOpenMPI]$ mpirun --hostfile hostfile -np 4 01.o
A - Greetings from process 0 of 4!
B - 0- Greetings from process 1 of 4!
B - 0- Greetings from process 2 of 4!
B - 0- Greetings from process 3 of 4!
Prima dell'invio: 1
Dopo l'invio: 1
Prima dell'invio: 2
Dopo l'invio: 2
Prima dell'invio: 3
Dopo l'invio: 3
[cuspide@node1 srcOpenMPI]$ █
```

Create the cluster

- Stop the running virtual instance
- Select and Open the Virtual Instance
- Click on “Create Machine Image” button
- Set “template” as name
- Create the image

Create a machine image

Name *

Name is permanent

Description

Source VM instance *

Location

Multi-regional

Regional

Select location

Create the cluster

- From Compute Engine > Machine images, select the template called as “template” and select “Create instance”
- Set the new instance name as node2
- Do the same for node2, node3 and node4

The screenshot displays the Google Cloud Platform interface for the Compute Engine service, specifically the Machine images page. The left-hand navigation pane is expanded to show 'Machine images' under the 'Virtual machines' category. The main content area features a table with the following data:

	Name	Source instance	Machine type	Actions
<input checked="" type="checkbox"/>	template	node1	e2-standard-2	

Create the cluster

Start all nodes and note that each Virtual Instance has got its own external IP as well as the Internal IP. This last one will be used to connect the virtual instance to each others

Filter VM instances						Columns
Name	Zone	Recommendation	In use by	Internal IP	External IP	Connect
<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> node1	us-central1-a			10.128.0.4 (nic0)	104.197.141.109	SSH
<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> node2	us-central1-a			10.128.0.5 (nic0)	34.123.43.212	SSH
<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> node3	us-central1-a			10.128.0.6 (nic0)	35.225.5.66	SSH
<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> node4	us-central1-a			10.128.0.7 (nic0)	35.224.178.150	SSH

Try the cluster interconnection

- Get an access to the first node (node1):
 - `ssh -l cuspide -i ./id_rsa 104.197.141.109`
- Try to connect using ssh to all other virtual instances using private network:
 - `ssh 10.128.0.4`
 - `ssh 10.128.0.5`
 - `ssh 10.128.0.6`
 - `ssh 10.128.0.7`
- Modify the hostfile
 - `10.128.0.4 slots=2`
 - `10.128.0.5 slots=2`
 - `10.128.0.6 slots=2`
 - `10.128.0.7 slots=2`
- Run the application again
 - `mpirun --hostfile hostfile -np 8 01.o`