Upload a VHD image of the Nexthink Appliance to Azure

Introduction

In order to support the installation of Nexthink on Azure, Nexthink provides a VHD (Virtual hard disk) file of the Nexthink Appliance and a set of scripts and installation packages to automate the installation process, some manual steps are required and it is recommended to have some minimal knowledge of a Linux environment in order to setup correctly the Nexthink Appliance. Furthermore, it is **mandatory** to apply the Security Hardening guide once your appliance is running, because it will be Internet facing (beginning from 6.17, this step can be automated).

Setup the VHD upload environment in Docker

At the minute there is no UI in the new Azure Portal to deploy a VHD, you need to setup a docker container on a standard Nexthink Appliance:

- on the Appliance, enable the extras repository with the following command:

```
sudo yum-config-manager --enable extras
```

- Install Docker on your Appliance as described on https://docs.docker.com/engine/installation/linux/docker-ce/centos/ (Use the "Install using the repository section" skipping the optional section 3. of this procedure)
- **overwrite the file /usr/lib/systemd/system/docker.service with the content:**
- **(192.168.1.5** will be the binding interface for docker**)**

```
[Unit]
Description=Docker Application Container Engine
Documentation=https://docs.docker.com
After=network-online.target firewalld.service
Wants=network-online.target

[Service]
Type=notify

ExecStart=/usr/bin/dockerd --bip=192.168.1.5/24
ExecReload=/bin/kill -s HUP $MAINPID

LimitNOFILE=infinity
LimitNPROC=infinity
LimitCORE=infinity

TimeoutStartSec=0
Delegate=yes
KillMode=process
Restart=on-failure
StartLimitBurst=3
StartLimitInterval=60s

[Install]
WantedBy=multi-user.target
```

- *sudo systemctl daemon-reload*

- *sudo systemctl start docker.service*
- *if you have an existing docker container (i.e: previous version of Azure client container), we recommend to remove it:*

```
sudo docker rm $(docker ps -a -q)
```

- **Install the docker container for Azure published by Microsoft:**

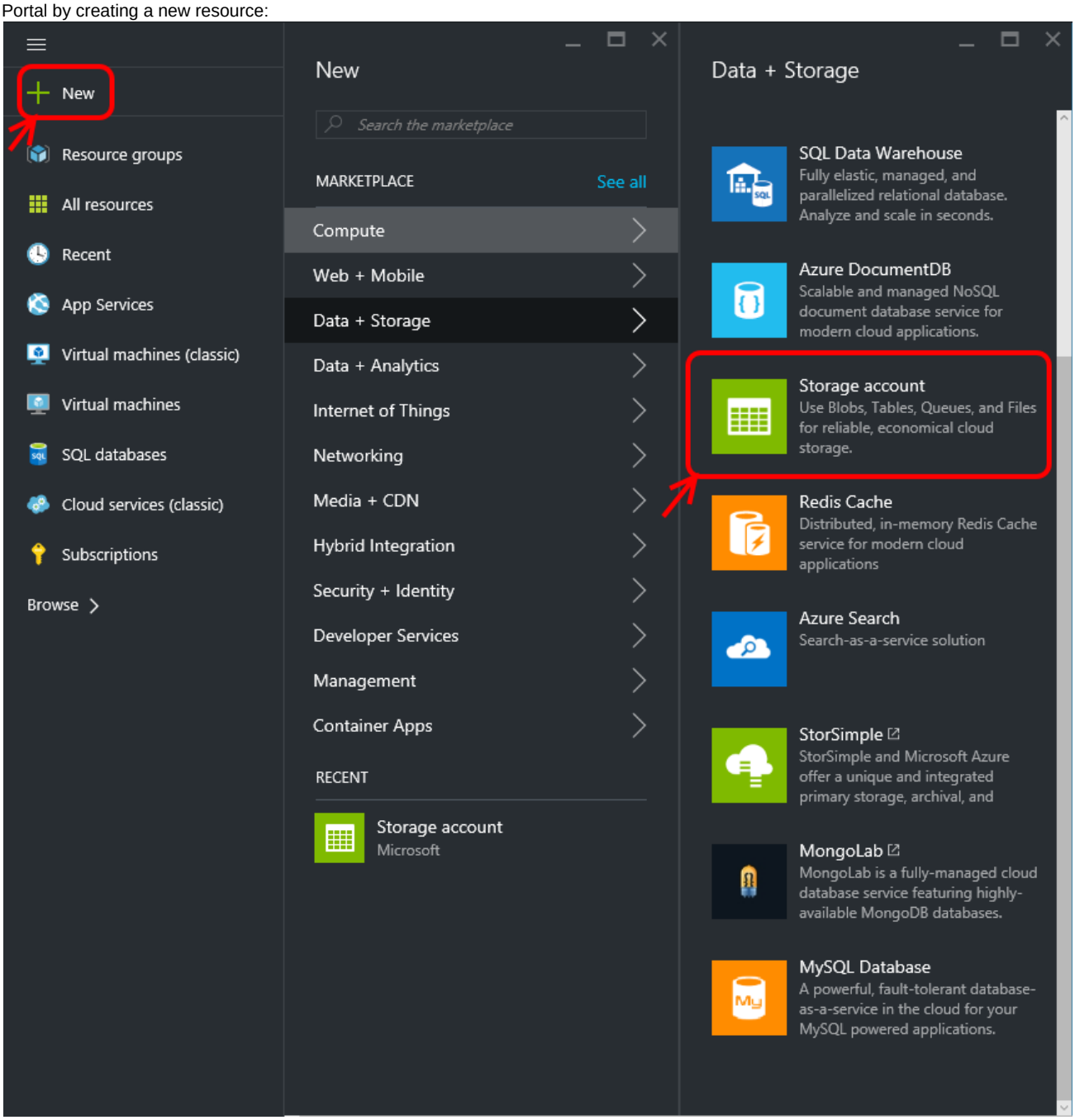
```
sudo docker run -it --name azure-cli azuresdk/azure-cli-python:latest
```

- **WARNING: Here the azure-cli is just an alias to your container, you can name it whatever you want, but docker-azure will have to be replaced by this new alias in the "docker exec" commands**

Setup your Azure account

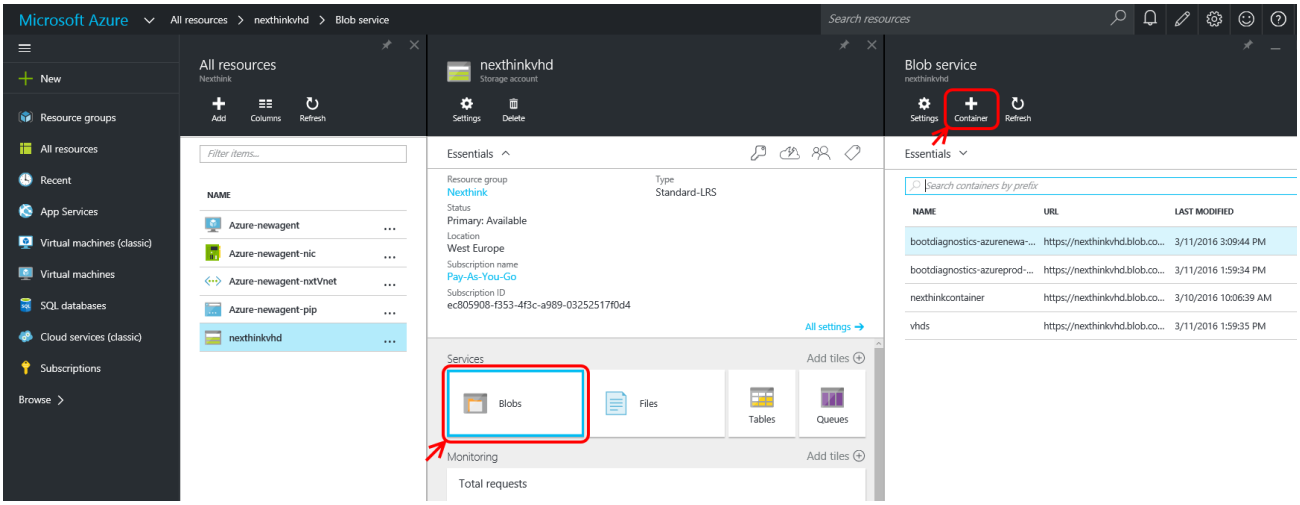
- Your Azure account must have at least the 'Contributor' role in order to proceed with the installation,please check with the IT department of your organization to gain the appropriate rights.
- In order to access to Azure Portal, we recommend using Internet Explorer or Microsoft Edge, you may need to 'allow azure to use local storage' in your browser' if you are notified so

- Make sure you have a storage account for your subscription, and a blob container for this storage account , you can do that in the Azure Portal by creating a new resource:



- The storage container should be created with 'Account kind': General purpose

- The next step is to create a blob container in this storage account, click on 'All resources', browse you storage account and select 'add container':



- Choose a name , select Blob access type and press 'Create button'
- **NB:** This is a basic first setup if you don't have already any Azure environment, in some infrastructures, it is possible to have several Resource groups depending on the resource type, see the script description below to configure that

Scripts definitions

Nexthink for Azure comes with a set of scripts which purpose is to facilitate and automate the deployment to Azure they can be called with a set of parameters, but they also have hard-coded properties to modifiy for the properties that are not supposed to change often

- createVMinAzure.sh hard-coded properties:
    - `RESOURCE_GRP_VM: the Resource group you create/use in Azure UI for storing the VM`
    - `RESOURCE_GRP_NETWORK: the Resource group you create/use in Azure UI for storing the network elements`
    - `RESOURCE_GRP_STORAGE: the Resource group you create/use in Azure UI for storing the storage elements`
    - `STORAGE_ACCOUNT_NAME: the Storage account you created in Azure UI (must have been created in the RESOURCE_GRP_STORAGE)`
    - `BASE_VHD: name of the VHD you upload to your storage account (ex: nxtazure-6.8.1.1.vhd)`
    - `CONTAINER_NAME: the blob container where you want to upload the VHD`
    - `LOCATION: the location of the datacenter where the VM will be hosted`
    - `VNET_PREF: address space of the vnet in case you don't specify an existing one.`
    - `SNET_PREF: address space of the subnet in case you don't specify an existing one.`
- **NB: in case you wish to use only oen Resource group, then you may set RESOURCE_GRP_VM, RESOURCE_GRP_NETWORK and RESOURCE_GRP_STORAGE to the same value**
- The script also has CLI parameters:
    - `--vmName: name of the VM you will create`
    - `--storageKey: the key that allows upload in your storage account, it should be displayed in the Container properties in the Azure UI:`
    - `--vmSize: sizing of the VM, please find more details at https://azure.microsoft.com/en-us/documentation/articles/virtual-machines-size-specs/#standard-tier-a-series`
    - `--vnetName: name of the vnet you want to use (if the name doesn't exist, it will try to create a vnet with this name)`
    - `--subnetName: name of the subnet you want to use (if the name doesn't exist, it will try to create a subnet with this name)`
- This script has 2 Major steps:
    - Setup the environment of the VM (create a subnet (this can be optional depending on your environment), configure the external IP, create a network interface).
    - Create the VM from the uploaded VHD.
- addDataDisk.sh hard coded properties:

    - `RESOURCE_GRP: the Resource group you created in Azure UI`
    - `RESOURCE_GRP_STORAGE: the Resource group you create/use in Azure UI for storing the storage elements (put same value as RESOURCE_GRP if you have only one resource group)`
    - `LOCATION: where the storage disk will be stored`
- The script also has CLI parameters:
    - `--vmName=name of the VM you will create`
    - `--diskSize=size of the disk we want to deploy (in GB)`

Prepare the docker container

- Make sure all the following files are copied to a single folder called per say *azureDirectory/*
    - *addDataDisk.sh*
    - *createVMinAzure.sh*
- *all the docker commands must be executed as root user, you can go to root using sudo -s or sudo -i*
- After the default properties of those scripts were edited, the scripts and components must be copied to the docker container using the command

```
docker cp azureDirectory/  azure-cli:/tmp/
```

- now login to Azure

```
docker exec azure-cli az login -u <your azure login>
```

- ...and enter your password

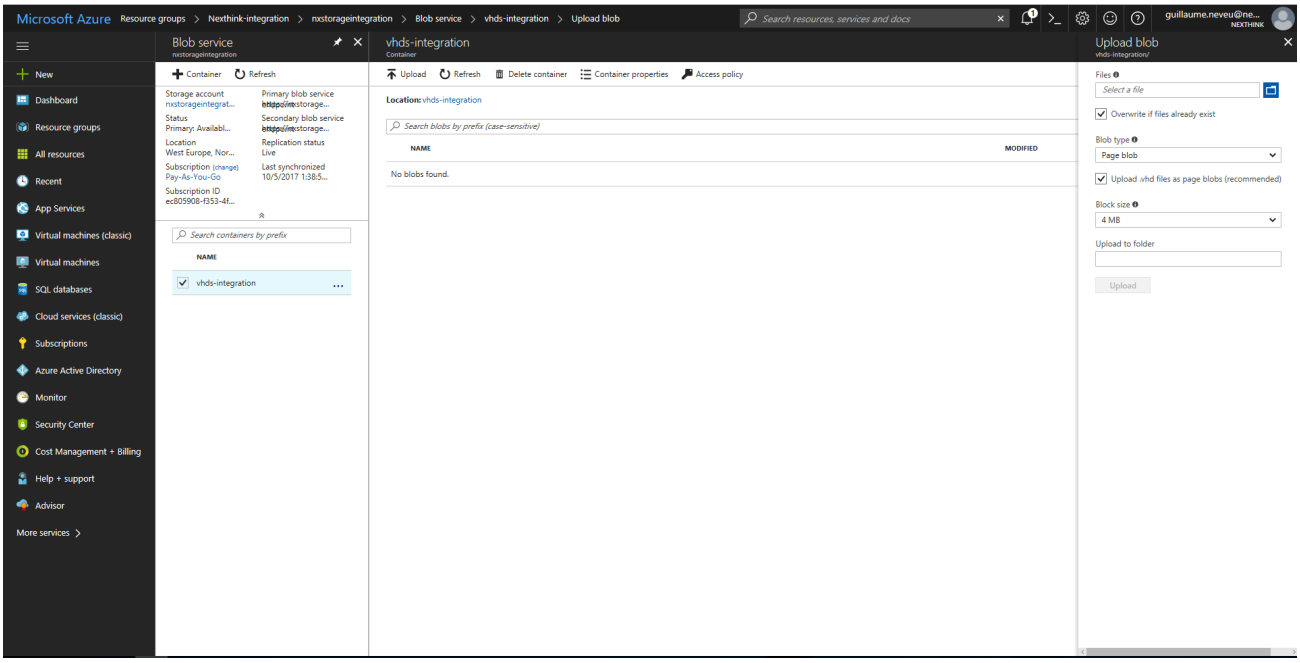
If the above command throws a login error you can try the interactive login method:

```
docker exec azure-cli az login
```

Then, follow the instructions displayed. You will need to have access to a web browser with Internet connection.

Upload the Vhd

- To upload the VHD to azure, the easiest is to go through the Web UI of Azure:
- In the resource group you use for storage go to Resource_group_name > storage_account_name > container_name and click on Upload button in the top bar:



- Make sure Blob type is set to Page blob in the advanced properties and click on the Upload button

Instantiate one or several new VMs

- Using the VM creation script will instantiate an Azure managed disk for your VM that will be dedicated, therefore the base vhd you uploaded (and for which you set the variable BASE_VHD in the script) will be used used as a template for all the VMs you want to create.
- However, if for whatever reason, you would like to backup an existing VHD, this can be achieved with the command:

```
docker exec azure-cli az storage blob copy start --source-
uri=https://<your storage account name>.blob.core.windows.net/<the
source container name>/nxtazure-6.X.vhd \
                --destination-container=<the target container name> --
destination-blob=<target Vhd name, for ex: engineAsiaAppliance.vhd> --
account-name=<storage account name> --account-key=<storage account
key>
```

- Now we can instantiate the VM, let's say you want to create the VM from the vhd "engineAsiaAppliance.vhd" from the previous example and use the existing vNet "myVnet" and existing subnet "mySubnet":

```
docker exec azure-cli /bin/bash /tmp/azureDirectory/createVMinAzure.
sh --vmName=engineAsiaAppliance --storageKey=<storage account key> --
vmSize=<size of the VM> --vnetName=myVnet --subnetName=mySubnet
```

- if the vnet "myVnet" and the subnet "mySubnet" don't exist yet, they will be created
- It is worth noticing that by default all machines in same Vnet will be able to connect to each other, if you wish to put machines on different Vnet, you need to ensure the connecitivty between them, this is not covered in this documentation but highly documented by Microsoft (please review this article)

Add a data disk to the VM

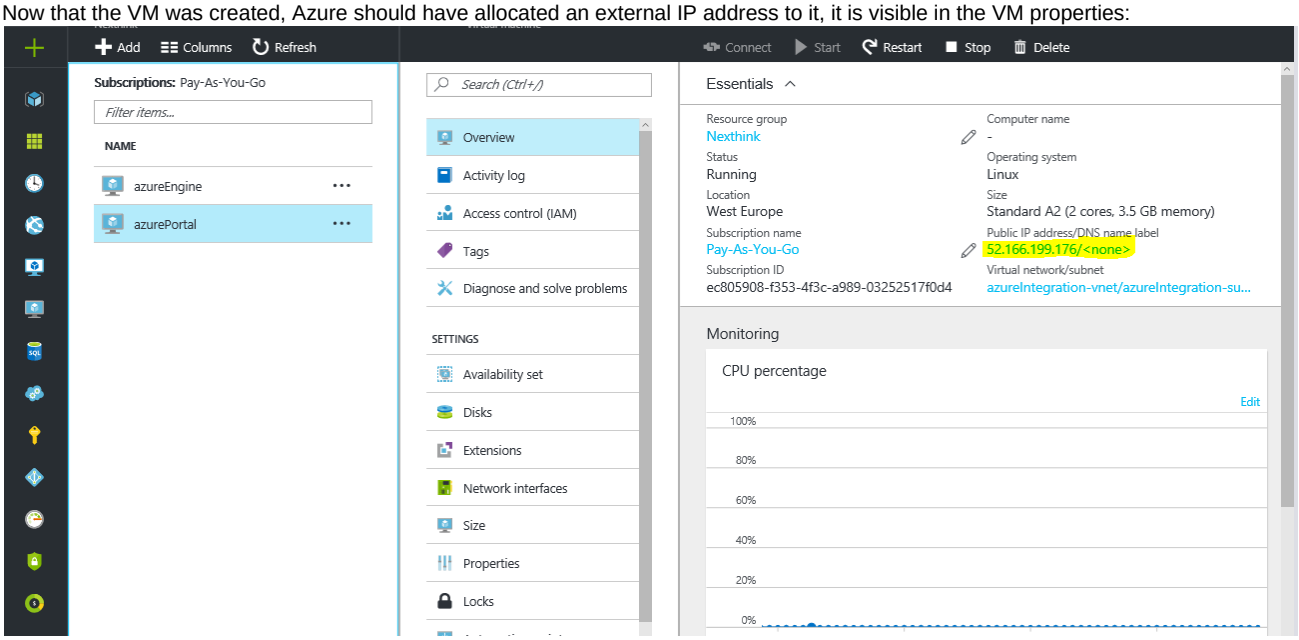
- The VHD containing the OS disk was built with a minimal size as Azure recommends to operate with a separate data disk for the important data. The disk we will attach will contain the Nexthink databases, and basically everything related to Nexthink components

```
docker exec azure-cli /bin/bash /tmp/azureDirectory/addDataDisk.sh --
vmName=engineAsiaAppliance --diskSize=100
```

- This will create a managed storage disk of 100GB, called engineAsiaAppliance-datadisk, once the script completed, it will be visible in your storage resource group

Format the data disk to the VM

- Now that the VM was created, Azure should have allocated an external IP address to it, it is visible in the VM properties:



- Copy the following files to this machine (the default password of the nexthink account is still 123456):

```
scp formatDataDisk.sh Nexthink-offline-install-6.X.tgz nexthink@<VM public
IP>:/tmp/
```

Alternatively, you can use your favorite SCP client in order to connect to the Newly created Nexthink VM on Azure, and then copy the files (e.g. WinSCP).

- Now login in SSH to the public IP of the VM

```
ssh nexthink@<VM public IP>
```

Alternatively, you can use your favorite SSH client in order to connect via SSH (e.g. PuTTY).

- Now that the data disk was created, you can check it by running the command '*lsblk*', a new disk /dev/sdc should appear having the size you specified:

```
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
fd0      2:0    1    4K  0 disk
sda      8:0    0    6G  0 disk ------> this should be the OS disk
sda1   8:1    0 1000M  0 part /boot
sda2   8:2    0    5G  0 part /
sdb      8:16   0  135G  0 disk ------> this should be swap disk
automatically allocated by Azure
sdb1   8:17   0  135G  0 part /mnt/resource
```

```
sdc      8:32   0   40G  0 disk ------> here is your data disk
sr0     11:0    1  1.1M  0 rom
```

- The disk is created but it still has to be partitioned before being used, to do so login in SSH to the public IP of the VM:

```
sudo /tmp/formatDataDisk.sh /dev/sdc
```

- This script is doing the following:
  - create an empty partition on /dev/sdc using fdisk tool
  - create a new LVM volume group and add the partition /dev/sdc1 to this group
  - create a logical volume taking 100% of the free space of the newly created volume group
  - create an EXT4 filesystem on this new logical volume
  - add an entry in /etc/fstab file and mount the disk on /var/nexthink

- If everything completed normally, now when running the command *df -ah*, you should see the size of the data disk you created allocated in */var/nexthink*: (30 GB in the below example)

```
Filesystem                      Size  Used Avail Use% Mounted on
rootfs                             -     -     -    - /
sysfs                              0     0     0    - /sys
proc                               0     0     0    - /proc
devtmpfs                       1.7G     0  1.7G   0% /dev
securityfs                         0     0     0    - /sys/kernel/security
tmpfs                          1.7G     0  1.7G   0% /dev/shm
devpts                             0     0     0    - /dev/pts
tmpfs                          1.7G  8.3M  1.7G   1% /run
tmpfs                          1.7G     0  1.7G   0% /sys/fs/cgroup
cgroup                             0     0     0    - /sys/fs/cgroup
/systemd
pstore                             0     0     0    - /sys/fs/pstore
cgroup                             0     0     0    - /sys/fs/cgroup/cpu,
cpuacct
cgroup                             0     0     0    - /sys/fs/cgroup
/memory
cgroup                             0     0     0    - /sys/fs/cgroup
/devices
cgroup                             0     0     0    - /sys/fs/cgroup
/perf_event
cgroup                             0     0     0    - /sys/fs/cgroup
/cpuset
cgroup                             0     0     0    - /sys/fs/cgroup
/freezer
cgroup                             0     0     0    - /sys/fs/cgroup
/hugetlb
cgroup                             0     0     0    - /sys/fs/cgroup
/net_cls
cgroup                             0     0     0    - /sys/fs/cgroup/blkio
configfs                           0     0     0    - /sys/kernel/config
/dev/mapper/nxt-root           2.8G  2.1G  461M  83% /
systemd-1                          -     -     -    - /proc/sys/fs
/binfmt_misc
hugetlbfs                          0     0     0    - /dev/hugepages
```

```
debugfs                              0     0     0    - /sys/kernel/debug
mqueue                               0     0     0    - /dev/mqueue
/dev/sda2                          969M   55M  848M   7% /boot
/dev/mapper/nxtdatapool-nxtdata     30G  244M   28G   1% /var/nexthink
binfmt_misc                          0     0     0    - /proc/sys/fs
/binfmt_misc
/dev/sdb1                          133G  2.1G  124G   2% /mnt/resource
tmpfs                              345M     0  345M   0% /run/user/1000
```

Configure static private IP

Although the public IP is created as static by default, we need to have a default dynamic private IP address for the VHD installation process. It as to be changed to Static private IP as a DHCP lease expiration for either Portal or Engine could result in problems with the federation or connectivity between the Portal and the Engine.
In order to do so:

- Select your VM from the Virtual machines tab
- got to Networking
- select the network interface of the machine
- select "IP configurations"
- select the default ip configuration assigned to this interface
- select "Static" and pick an IP address that belongs to the space of the vnet the network interface is connected (if you wish so can actually keep the same IP as the one that was given by the DHCP server)
- click on Save and wait few minutes for Azure to reconfigure the VM
- NB: you only set here
- **/!\WARNING/!** : Do not edit the network configuration from the Webconsole as you would do on an onpremise installation, this can result in a total loss of connectivity of the machine, thus making it completely unusable as we only have SSH access to it

Install Nexthink on the VM

- Transfer the installation package '*Nexthink-offline-install-6.X.tgz*' to the VM using your favorite SCP client. Please visit Nexthink Product Downloads page to get the installation package **for Cloud**.
- now, unpack the installation package:

```
tar -xzvf Nexthink-offline-install-6.X.tgz
```

- the script '*installNexthinkOnAzure.sh*' takes two optional parameters: -p to install the Portal, -e to install the engine, depending on what you want on this Appliance:

```
sudo sh installNexthinkInCloud.sh -p -e
```

- Nexthink is now installed, you may wish to check if the components are up & running:
  - Engine:

```
nxinfo info
```

  - Portal:

```
```

```
sudo systemctl status nxportal
```

- Or the Portal by connecting to https://<your public ip> (please note that Portal can take some that to start, you can monitor the logs in */var /nexthink/portal/logs*).
- As written in the security hardening guide, you can now change the default password for the Webconsole and the Portal and make sure that all the steps of the guide are strictly followed.

Important configuration notes:

Compared to a standard installation of Nexthink, the fact that the Appliance(s) is(are) facing Internet on one side and facing an internal network on another side must be taken into account. Notably regarding the Portal Engine Configuration. The Internal IP/DNS Name of the machines must be used when configuring:

- Internal and External DNS in the webconsole parameters section
- Portal IP/hostname in the Engine's webconsole
- Engine hostname in the Portal Engine configuration tab **(here it must be a hostname such as this host will be resolved as the Engine's internal IP address by the Portal machine and it will be resolved as engine's external IP address on the machine using the Finder, this is important so that the Finder can have access to the engine - You can also configure the /etc/hosts file on the Portal machine to make the engine hostname resolve to the internal IP of the engine or use DNS servers).**

Security Hardening

Now that your Appliance is facing Internet, it is **mandatory** to change the password of the root account as well as for the nexthink account, this can be done using the CLI:

```
sudo passwd root
sudo passwd nexthink
```

Or simply after the first login on the webconsole, it will ask you to change the default webconsole password and also the password of the nexthink support account.
At this stage, main Nexthink ports are closed by default, thus you have to apply the security hardening, starting with 6.17, this step is automated, from the CLI , you must run:

```
sudo nxhardening
```

Current limitations

With the current version of Nexthink Product, we have the current limitations with respect to installations on Microsoft Azure:

- Some of the files that the Nexthink Solution is using are stored in a specific folder and, depending on the customer configuration, could end up in being very big (especially for the appliance running the Portal). As Azure comes with predefined configurations, it might happen that the customer needs to choose (and pay) for a configuration that is not adapted in terms of RAM and CPU just in order to cope with the requirement in terms of disk size.