HBO-ICT: TICT-VKCLIM-17

Cloud Infrastructure & Management Workbook

Contents

1 Introduction:

“Cloud Infrastructure Technology” is best studied by doing. That's also what we're going to do this course and this workbook will help. You will find in this document all assignments belonging to this course. Each lesson has about the same structure:

- **Lesson assignment (optional)**: This is an assignment that you do in class, to put theory into practice.
- **Week assignment**: These assignments can already be done at home, but you do not have to: You do these assignments to practice what your teacher explains during the lesson. It's too much work to finish during the lessons, so finish it at home. Moreover, don’t remove the result! You can use part of it for your final assignment!
- **Theoretical Exam**: Study all information that is marked under “**Exam Material**” very well, because a written exam is part of this course. The questions from the exam are derived from this workbook, the presentations, the book and the information on SharePoint.
- **Other Information**: Containing additional information to better understand the subjects.

We assume you’ll use your own laptop, more details below.

Theory and examination

The student manual describes how this course was built. The planning for this six-week course can be found below:

Lesson	Chapter Book (Kavis)	Theme
Week 1, lesson 1	1-2-3	Introduction
Week 1, lesson 2	5	IaaS
Week 2, lesson 1		Office Automation
Week 2, lesson 2	4	Architecture
Week 3, lesson 1		Authentication / Federation
Week 3, lesson 2	12	Storage | Cloud Management (Techn)
Week 4, lesson 1	6	Interfacing
Week 4, lesson 2		Containerization
Week 5, lesson 1	7-8-9-10-13	Cloud Security
Week 5, lesson 2	11-14-15	ITSM
Week 6, lesson 1	-	Recht
Week 6, lesson 2	16	Other | Future

Preparation

1.1.1 Introduction

During the labs we'll be using CentOS 7 (Linux) and VMware Workstation on your own laptop, unless specified otherwise.

Although there are several options, we prefer the following configuration:
Your own laptop with >=8GB (16GB is preferred, Openstack is quite demanding!) memory, virtualization support, an internal HD and/or SSD and "VMware Workstation 12 or 14 for Windows".

You are responsible for creating the right environment, but we will help a little. If you choose a different configuration such as Linux, or using Apple operating systems, this is not forbidden, but you will not get support from your teacher and you must make sure that you can perform all tasks.
Any questions? Ask your teacher ...

1.1.2 Amazon AWS Educate

Most public cloud providers require you to present a credit card before using their infrastructure. Therefore, most of our labs are based upon using your laptop in combination with Openstack or other software. However, limited usage of the Amazon AWS is possible because the HU is an Amazon accredited institution. At the end of this workbook you can find what is needed to acquire credits on AWS. These credits can be used for your end assignment and/or labs in week 5. If you like, you can already sign on (see 2.1.59).

More information about the course can be found in the student manual.

2 Assignments

Week 1, Lesson 1: Openstack (Install)

OpenStack is an open-source software platform for cloud computing, mostly deployed as infrastructure-as-a-service (IaaS), whereby virtual servers and other resources are made available to end users. OpenStack embraces a modular architecture to provide a set of core services that facilitates scalability and elasticity:

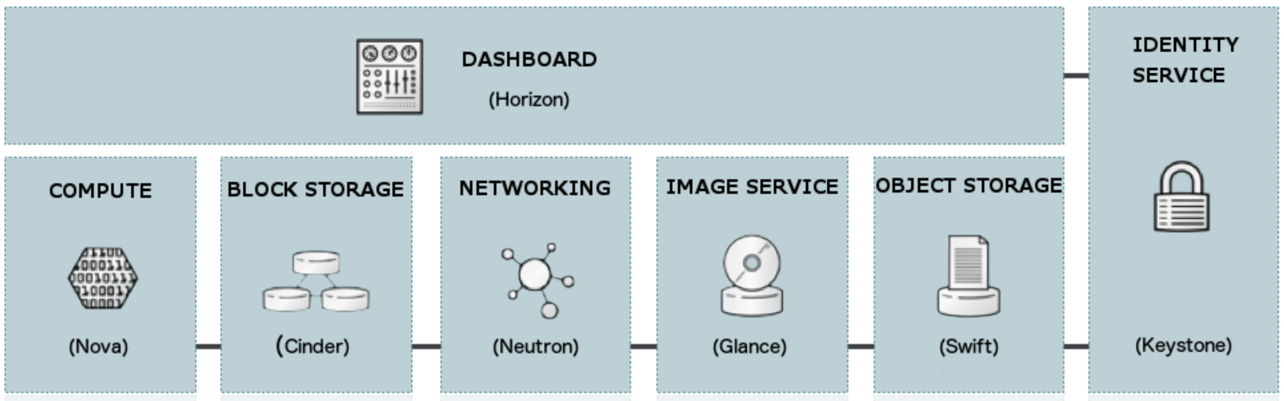
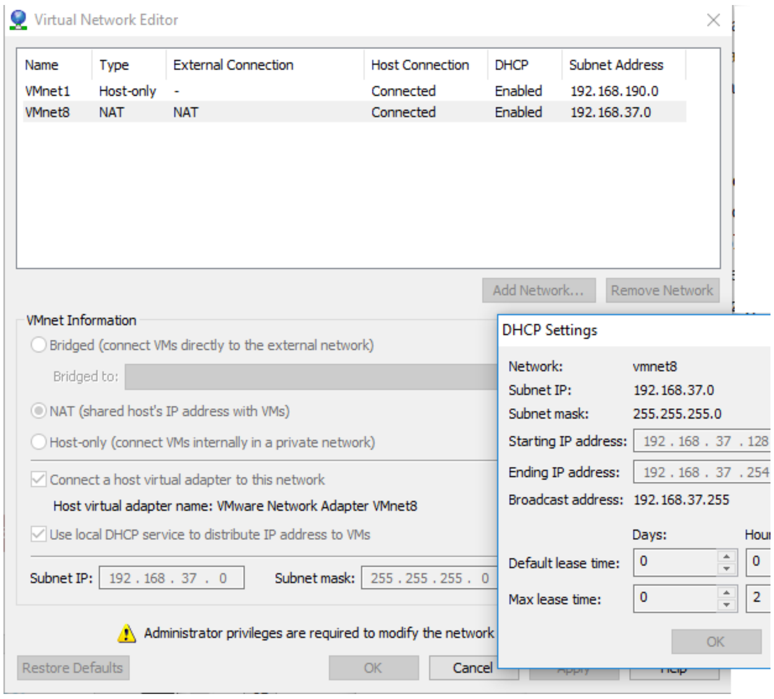


FIGURE 1: SOURCE: HTTPS://DOCS.OPENSTACK.ORG/SECURITY-GUIDE/_IMAGES/MARKETECTURE-DIAGRAM.PNG

This week we will install Openstack on our laptop using VMware Workstation. In the first lesson, it is installed, in the second lesson you will practice some cloud key concepts in this environment. It is advised to do exactly as is written below because while using Openstack there are many options and you will not become an Openstack expert after this lab.

2.1.1 Prepare your host and VM

- **Laptop:** >60 GB free space, >=8GB RAM (16GB recommended), i5 at least and an SSD is recommended.
- **VMware Workstation 12.5.6 or later. Network setup:** VMnet8 (NAT, gateway=192.168.37.2, DHCP scope 192.168.37.128-192.168.37.254), Vmnet1 (Host-only with 192.168.190.1 as host virtual adapter).



Name	Type	External Connection	Host Connection	DHCP	Subnet Address
VMnet1	Host-only	-	Connected	Enabled	192.168.190.0
VMnet8	NAT	NAT	Connected	Enabled	192.168.37.0

VMnet Information
- Bridged (connect VMs directly to the external network)
  Bridged to:
- [✓ NAT (shared host's IP address with VMs)]
- Host-only (connect VMs internally in a private network)

[✓] Connect a host virtual adapter to this network
Host virtual adapter name: VMware Network Adapter VMnet8
[✓] Use local DHCP service to distribute IP address to VMs

Subnet IP: 192 . 168 . 37 . 0 Subnet mask: 255 . 255 . 255 . 0

Administrator privileges are required to modify the network

DHCP Settings

Network:	vmnet8
Subnet IP:	192.168.37.0
Subnet mask:	255.255.255.0
Starting IP address:	192 . 168 . 37 . 128
Ending IP address:	192 . 168 . 37 . 254
Broadcast address:	192.168.37.255

	Days:	Hou
Default lease time:	0	0
Max lease time:	0	2

- **Create a New Virtual Machine (I will install OS later):**
    - **CentOS7:** CentOS-7-x86_64-DVD-1611.iso
    - **Memory:** >=7168MB
    - **Hardware → Processors:** check “Virtualize Intel VT-x/EPT or AMD-V/RVI” (nested virtualization)
    - **Network Adapter:** NAT
    - **Create a disk:** 60GB
    - **Another Network Adapter:** Custom Vmnet1 (Host-only)
    - **Both NICs** connected and connect at power on

- **Install version of CentOS7:**
    - **Version** CentOS-7-x86_64-DVD-1611
    - **Partitioning:** Manual, followed by “Automatic Creation” followed by “delete /home partition”
    - **Software selection:** minimal
    - **Hostname:** packstack.openstack.local
    - **Two NICs** (ens33 / ens34):
        - NAT = ens33 = 192.168.37.100/24
        - VMnet1 = ens34 = 192.168.190.100/24
    - **NTP “ON”**
    - **Begin installation, followed by creating a user account and:**
    - **Selinux:** permissive
    - **/etc/ssh/sshd_config:** PermitRootLogin yes → use putty as ssh-client for the rest of the lab.
    - **yum update**
    - **yum install nano telnet tcpdump**
    - **hostnamectl set-hostname packstack.openstack.local** (in case you forgot to set the hostname during installation)
    - **cat > /etc/hosts**

      ```
      127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4 packstack
      ::1         localhost localhost.localdomain localhost6 localhost6.localdomain6 packstack
      192.168.190.100 packstack packstack.openstack.local
      192.168.190.101 compute compute.openstack.local
      ```
    - **ip route** contains at least (check the interfaces used):
        - a default route (192.168.37.2)
        - a route to the 192.168.37.0 network
        - a route to the 192.168.37.190 network
    - **systemctl disable firewalld**
    - **systemctl stop firewalld**
    - **systemctl disable NetworkManager**
    - **systemctl stop NetworkManager**

- **Modification to the network:** we could use tools on a higher level, but let’s modify the configuration in the directory /etc/sysconfig/network-scripts directly this time:
    - **[root@packstack2 network-scripts]# cat ifcfg-ens33**, should be something like:

      ```
      NAME=ens33
      DEVICE=ens33
      TYPE=Ethernet
      BOOTPROTO=none
      DEFROUTE=yes
      ```

```
ONBOOT=yes
IPADDR=192.168.37.100
PREFIX=24
DNS1=8.8.8.8
GATEWAY=192.168.37.2
```

- **[root@packstack2 network-scripts]# cat ifcfg-ens34**, should be something like:

```
NAME=ens34
DEVICE=ens34
TYPE=Ethernet
BOOTPROTO=none
ONBOOT=yes
IPADDR=192.168.190.100
PREFIX=24
```

- **Prepare for openstack installation:**
  - **reboot**
  - **yum install -y centos-release-openstack-ocata** (installs the repository)
  - **yum update -y**
  - **yum install -y openstack-packstack** (download the software)

- **We would strongly suggest to shutdown the host and create a snapshot now! (name: CleanInstall).**

2.1.2 Install Openstack using Packstack RDO

RDO stands for "**R**PM **D**istribution of **O**penStack". It is a community-supported distribution of OpenStack that was launched by Red Hat in 2013. It is one of the methods to install Openstack, feel free to use another method, but in that case: you're on your own!
The software is installed by creating an answer file for all possible questions during installation. You should modify this file for a number of items followed by the installation (that took me 20 minutes on a fast laptop!).

- **packstack --gen-answer-file=rdo.txt**
- **Modify rdo.txt** so that**:**

```
> CONFIG_DEFAULT_PASSWORD=secure
> CONFIG_CONTROLLER_HOST=192.168.190.100
> CONFIG_COMPUTE_HOSTS=192.168.190.100
> CONFIG_NETWORK_HOSTS=192.168.190.100
> CONFIG_STORAGE_HOST=192.168.190.100
> CONFIG_SAHARA_HOST=192.168.190.100
> CONFIG_AMQP_HOST=192.168.190.100
> CONFIG_MARIADB_HOST=192.168.190.100
> CONFIG_MONGODB_HOST=192.168.190.100
> CONFIG_REDIS_HOST=192.168.190.100
> CONFIG_KEYSTONE_LDAP_URL=ldap://192.168.190.100
> CONFIG_MARIADB_PW=secure
> CONFIG_KEYSTONE_ADMIN_PW=secure
> CONFIG_KEYSTONE_DEMO_PW=secure
> CONFIG_LBAAS_INSTALL=y
> CONFIG_NEUTRON_ML2_TYPE_DRIVERS=vxlan,local,flat
> CONFIG_NEUTRON_OVS_BRIDGE_IFACES=br-ex:ens33
```

- **Apply your modified answerfile** (take a cup of coffee):
  packstack --answer-file=rdo.txt

- Ready? See if you can visit the website: http://192.168.190.100/dashboard

- **In order to make the network setting persistent create an OVSBridge en OVSPort:**

```
cat > /etc/sysconfig/network-scripts/ifcfg-br-ex
NAME=br-ex
DEVICE=br-ex
DEVICETYPE=ovs
TYPE=OVSBridge
BOOTPROTO=static
IPADDR=192.168.37.100
NETMASK=255.255.255.0
GATEWAY=192.168.37.2
DNS1=8.8.8.8
ONBOOT=yes
IPV6INIT=no

cat > /etc/sysconfig/network-scripts/ifcfg-ens33
NAME=ens33
DEVICE=ens33
ONBOOT=yes
IPV6INIT=no
DEVICETYPE=ovs
TYPE=OVSPort
OVS_BRIDGE=br-ex
```

- **Reboot** (takes 5 minutes for me after a reboot before I could login via the website)
- **source keystonerc_admin** (see in the file what happens!)
- **ovs-vsctl show (is ens33 interface part of the br-ex?)**
- **Again, this would be a fine moment to shutdown the host and create a snapshot! (PackStackInstalled).**

2.1.3 Configure Openstack Network and first VM

- **First, we want to clean up a little:** Login to the website as admin and:
  - Delete the demo project
  - Delete the demo user
  - Goto the Admin and delete: the router and both networks because we want to create the infrastructure ourselves.

- **ovs-vsctl show**
  - See three bridges? Which?

    - To which bridge is ens33 connected?

- **Next, we'll create a public network:** that will contain the floating IPs the we need to connect from the outside world (the LAN for instance where this box is connected to).
    - Admin → System → Networks → Create Network
        - Name=public, project=admin, provider=Flat, Physical Network=extnet , State=UP, Shared=yes, External Network=yes → Submit
    - Click on the just created network "public" and create a new subnet:
        - SubnetName=public-subnet, Network Address=192.168.37.0/24 (your LAN), IPv4, Gateway=192.168.37.2
        - Subnet Details: Disable DHCP, instead allocate a pool of addresses outside the scope of the DHCP server of VM Workstation, e.g.: 192.168.37.50,192.168.37.70 with DNS=8.8.8.8 → Create.

- **Next, we'll create a private network within our single project "Admin":** that will contain the private IPs of our VMs. These IPs can connect to the outside world.
    - Project → Network → Networks → Create Network
        - Name=private → next
        - Subnet-name=private-subnet, Network Address=10.1.0.0/24 (for instance), IPv4, Gateway=10.1.0.1 (first address in the network) → next
        - Subnet Details: Enable DHCP, Pool: 10.1.0.10,10.1.0.30, DNS=8.8.8.8 → Create.

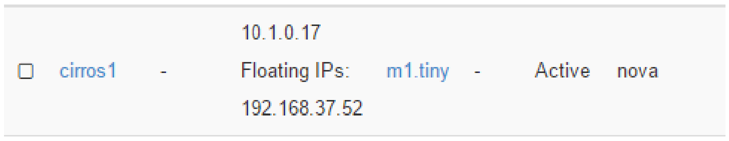
- **Now, we have to connect both networks using a router.**
    - Project → Network → Routers → Create Router
        - Name=router1, External Network=public → Create
        - Next, we connect the router to both networks. Click on "router1", tab "interfaces":
            - There is an interface on the external network already, e.g. 192.168.37.57
            - router1 → add interface: subnet = private and IP is 10.1.0.1 → Submit

- **Click on Project → Network → Network Topology: Does this make sense to you?**

- **Next: allow data flowing to the network (open the firewall):**
    - Click on Project → Network → Security Groups → manage rules of the default security group
    - Do you see what is allowed outgoing? (all traffic)
    - What is allowed incoming? (only traffic from VMs in the same security group)
    - We want to allow management from your local pc, so add a rule:
        - All ICMP, Ingress, CIDR, 0.0.0.0/0 → Add
        - SSH, CIDR, 0.0.0.0/0 → Add

- **Network is ready! Next step is to spawn a new VM in our admin project**. We use a standard image called "Cirros". It is a very small Debian like Linux version (12MB!). Great for testing:
    - Project → Compute → Images → Launch
    - Details: Name=Cirros1
    - Source: Use cirros
    - Flavor: m1.tiny (we'll explain this later)
    - Networks: private
    - Launch Instance

- **Make it available from the outside by Associating it with a Floating IP** from the Pool (Allocate IP if needed)
  Result should be something like this:



☐	cirros1	-	10.1.0.17 Floating IPs: 192.168.37.52	m1.tiny	-	Active	nova

- **Test if it works, that is:**
  - From your host:
    - Ping the floating ip
    - Ssh to your instance, username=cirros / password=cubswin:)
  - From your instance:
    - Ping www.klm.nl

- **This is what you've been creating(see below). See the result using "ip a" on your Openstack VM and "ovs-vsctl show".**

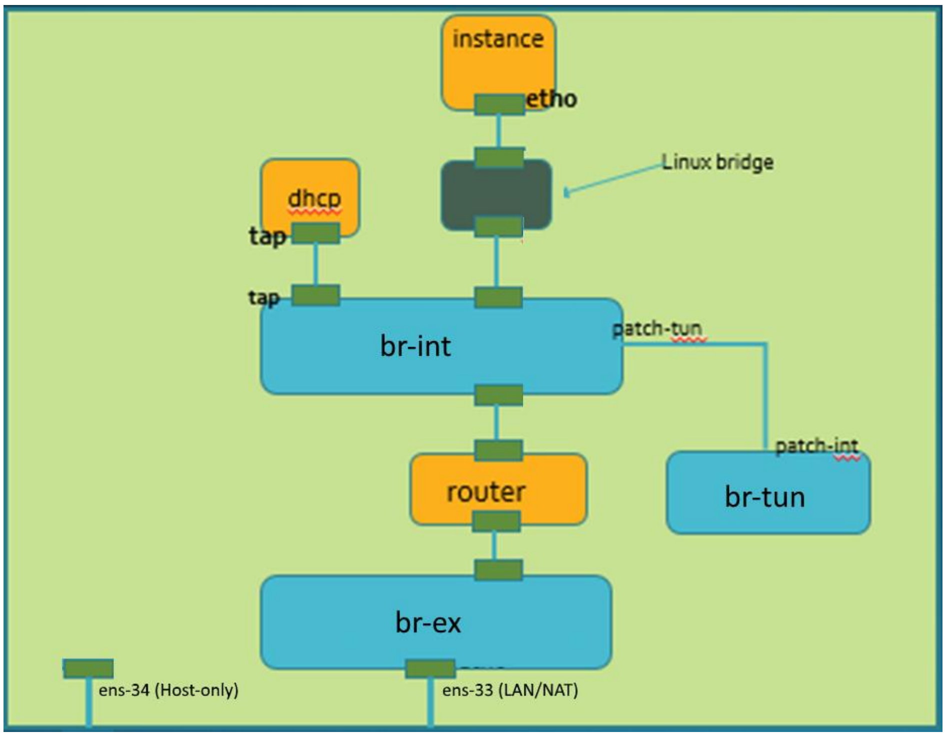


FIGURE 2: MODIFIED FROM: [HTTPS://SUNNYNETWORK.WORDPRESS.COM/2016/03/28/LAB-12DEPLOYING-OPENSTACK-USING-PACKSTACK-ALLINONE/](https://sunnynetwork.wordpress.com/2016/03/28/lab-12deploying-openstack-using-packstack-allinone/)

- **Another snapshot (after shutting down the host)** (PackStackInstalledAndWorking).

2.1.4 Exam material

How to prepare for the exam this week?

- **Learn** all the information to be found in the powerpoint presentations.
- **Read** the information to be found below under the heading "information sources".

- **Be able** to install Openstack using the packstack method resulting in functional cirros VMs, network connectivity (ssh to VM using floating IP, Internet available from VM), functional website (horizon).
- **Be able** to understand the answer-file for packstack.
- **Study the Book (Architecting the cloud): chapter 1, 2, 3 and 5.**

2.1.5 Other information

- RDO Packstack: https://www.rdoproject.org/install/quickstart/
- RDO Packstack advanced: https://www.rdoproject.org/networking/neutron-with-existing-external-network/
- Nice Openstack skills: https://www.youtube.com/watch?v=eOlIB323c8s

Week 1, Lesson 2: 'Configure Openstack'

Ok, the software is installed and your private cloud is operational and serving an instance. This lesson, is not so "step-by-step" and is meant to make you more familiar with this complicated environment. At the end of the lesson you've created your own image, installed on your own volume, managed using your own SSH keypairs using your own environment (flavor). In other words, after this lesson, you know the basics of Openstack.

2.1.6 The basics of Openstack

While it boots

Booting can take a while, Openstack us using about 4.8GB of memory before you can login and start your instances. Once you can login, follow memory usage in megabytes using "`watch -n 5 free -m`" from the command line.
Start your cirros instance after login.

Managing Openstack from the command line

Some tasks and command line options to be found below. Which command is needed to get the results below? Execute all commands from your Openstack VM.

Command #	Task Description
_________	Do this before anything else to prevent authentication errors
_________	Show all running instances
_________	show all your networks including the id
_________	show the details of a certain network id
_________	show a list of all images
_________	show the log of the console of your running instance
_________	show the volumes available
_________	list all environments available to host a server
_________	show all running compute services
_________	show a list of all users

#	Command
1	`nova list --host packstack.openstack.local`
2	`source keystonerc_admin`
3	`openstack compute service list`
4	`openstack user list`
5	`openstack network show <id>`
6	`openstack image list`
7	`openstack volume list`
8	`openstack flavor list`
9	`openstack console log show cirros1`
10	`openstack network list`

Flavors

We'll be creating a new flavor

- Cirros is not very demanding. Log in and see what amount of memory is used: _____________
- Based on that information create a new flavor using the GUI with the amount of memory found * 2, using the name "nano".

- Start a new cirros instance using that particular flavor.
- Create a second flavor "nano2" using the command line and output of the commands:
  - `openstack flavor list`
  - `openstack flavor create --help`
  - What command did you use for this? ____

- Spawn an new instance in your project using the flavor "nano2".
  - Logon to the console using the GUI. (Use Firefox if Chrome is not responding well).
  - Logon using ssh from the Openstack VM in VMware Workstation to see if this works.

- Extra challenge (optional): add an extra floating ip to the just create instance using the command line. Hint: two commands needed, both starting with "openstack" .

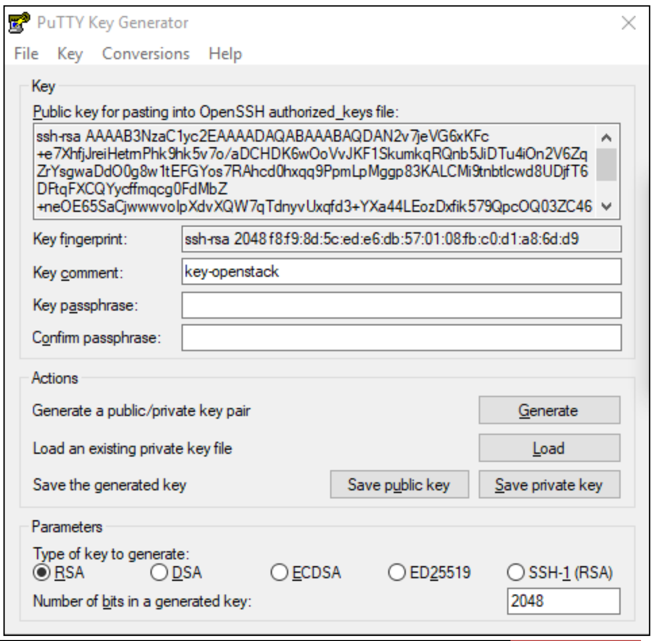
SSH Key Management

One way to enlarge security on your OpenStack cloud is to set up security options that go beyond password-based authentication. The most common way is to use the OpenStack Dashboard, Horizon, to set up public/private key pairs to properly protect the instance at launch time.

In addition: Wouldn't it be wonderful if we were able to login automatically without typing cirros | cubswin:) each time? Let's use SSH-key management and **inject** your ssh-keys on boot:

- In the left-side navigation menu, click Project → Computer → Key Pairs. In the main portion of the screen, click the tab labeled Key Pairs, and choose to Create Key Pair. The Create Key Pair dialog will prompt you to supply a key pair name before downloading a private key (pem format) to your client. (as an alternative, use cli: `openstack keypair create <keyname>`)
- Save the key (copy and paste using nano?) to the right local directory and file, e.g.: `~/.ssh/myprivkey.pem`
- `chmod 600 ~/.ssh/myprivkey.pem` (why?!)
- Create a new cirros instance called "simpleManaged" and **use the just created Key Pair**.
- Start the instance and provide it with a floating ip.
- Now, try to login using SSH and your key pair from your Openstack VM in VMware Workstation:
- `$ ssh -i ~/.ssh/myprivkey.pem cirros@floating_ip_address`

Even more easy is to use putty from your windows(?) host. The downside of PuTTY is that it doesn't like the *.pem format OpenStack gives you, in which the public and private key are together; instead you must separate them using the PuTTYgen client. Let's try this:



Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDAN2v7jeVG6xKFc
+e7XhfjJreiHetmPhk9hk5v7o/aDCHDK6wOoVvJKF1SkumkqRQnb5JiDTu4iOn2V6Zq
ZrYsgwaDdO0g8w1tEFGYos7RAhcd0hxqq9PpmLpMggp83KALCMi9tnbtlcwd8UDjfT6
DRtqFXCQYycffmqcg0FdMbZ
+neOE65SaCjwwwvolpXdvXQW7qTdnyvUxqfd3+YXa44LEozDxfik579QpcOQ03ZC46
```

Key fingerprint: ssh-rsa 2048 f8:f9:8d:5c:ed:e6:db:57:01:08:fb:c0:d1:a8:6d:d9

Key comment: key-openstack

Key passphrase:

Confirm passphrase:

Generate a public/private key pair

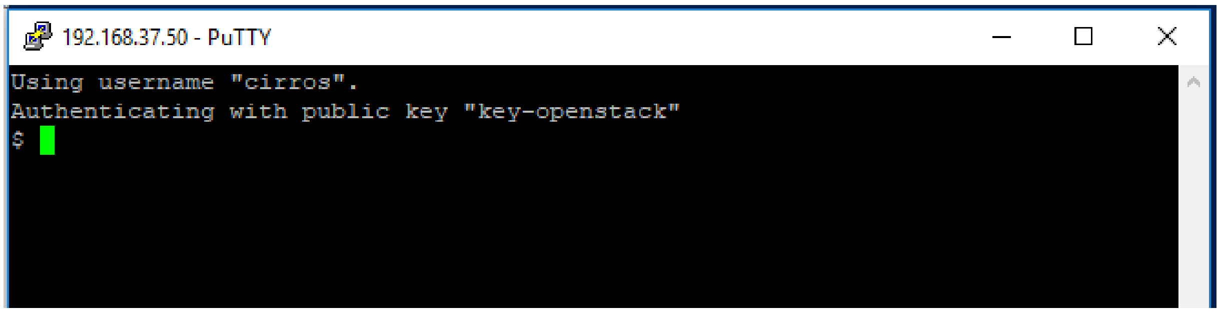
Load an existing private key file

Save the generated key

Type of key to generate:
[✓ RSA] DSA ECDSA ED25519 SSH-1 (RSA)

Number of bits in a generated key: 2048

- **Download PuTTYgen** from the main downloads page: https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html
- Run puttygen.exe.
- Click the Load button and specify "All files *.*" as the file format so it can find the *.pem file.
- Load the key pair you downloaded from OpenStack earlier. You'll see the information in the puttygen window.
- Optionally add a passphrase.
- Click "Save Private Key" and save the file where you'll be able to find it.

- Now you can add the private key to a connection in PuTTY:
- **Download PuTTY** from the main downloads page. Run putty.exe.
- Specify the hostname or IP for the instance (you can find this information on the instances page once it's launched).
- In the left-hand panel, expand "Connection->SSH" and click "Auth".
- Click "Browse" and find the *.ppk file you saved from puttygen.exe.
- Click "Open" to connect to the instance.
- 

```
Using username "cirros".
Authenticating with public key "key-openstack"
$
```

Finally launch a new instance with you own key, flavor and cirros image using the command line and the information below.

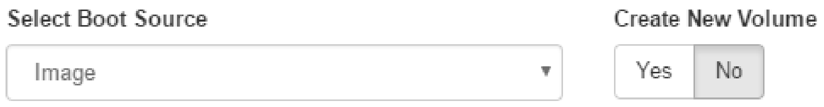
- `openstack image list`
- `openstack flavor list`
- `openstack keypair list`
- `openstack network list`
- Slow laptop with little memory? Stop cirros1 first before continuing, followed by:
```
openstack server create --flavor <myflavorid> --image <myimageid> --
security-group default --key-name <mykeyname> --nic net-
id=<privatenetworkid>  myCirrosServer
```

You could create a new snapshot now if you like (name: DoneWithCirros).

Images
Until now, we've been using the cirros image. That's great for testing purposes, but a real IaaS service should provide other images as well. Using the "Glance" service and some tooling we can accomplish this. Let's go:

The simplest way to obtain a virtual machine image that works with OpenStack is to download one that someone else has already created. Most of the images contain the cloud-init package to support the SSH key pair and user data injection. Because many of the images disable SSH password authentication by default, boot the image with an injected key pair. You can SSH into the instance with the private key and default login account:

- Download an image. E.g.: https://docs.openstack.org/image-guide/obtain-images.html . Choose a Fedora image (https://alt.fedoraproject.org/cloud/) Note: Because we are using KVM, we recommend using the images in qcow2 format,
- If needed (just in case you receive errors), modify the `nova.conf` preventing timeouts using the input provided below (and reboot afterwards):
```
block_device_allocate_retries = 60 (default) to 300
block_device_allocate_retries_interval = 3(default) to 10
block_device_creation_timeout = 10(default) to 300
```
- Goto Admin → Images → "Create Image" and add your image.
    - Notes: Architecture=x86_64
- If needed, create your own flavor with 4GB disk and 1GB RAM (maybe less is also possible, but no less than 256MB).
- From there, you should be able to use the new image to launch a new instance, but do not create a new volume, see image below:


Select Boot Source: Image
Create New Volume: Yes [✓ No]

- Associate it with an floating ip.
- You cannot login using username/password. Can you login using ssh and keys instead?
```
ssh -i ~/.ssh/<yourkey> fedora@<floatingip>
```
- Shutdown the instance, because you are using "nested virtualization" and that's slow of course.

This was real IaaS. However, an IaaS provider will sometimes provide its own modified images. Logical next step is to create an instance of our own. Because we have experience using CentOS7, let's create our own system that is containing a webserver and is "cloud ready":

First, we have to download the iso:

- Navigate to the CentOS mirrors page.
- Click one of the HTTP links in the right-hand column next to one of the mirrors.
- Click the folder link of the CentOS version that you want to use. For example, 7/.
- Click the isos/ folder link.
- Click the x86_64/ folder link for 64-bit images.
- Click the netinstall ISO image that you want to download. For example, CentOS-7-x86_64-NetInstall-1611.iso is a good choice because it is a smaller image that downloads missing packages from the Internet during installation.

Next, install the OS as a VMware VM (1024MB memory / Single VMDK-file of maximum size: 4 GB). Some remarks:

- Default Ethernet setting = ON
- IPv4 Settings Method = DHCP
- Default hostname is fine, later we install the cloud-init package which sets the host name on boot.
- Installation source. Depending on the version of CentOS, the net installer requires the user to specify either a URL or the web site and a CentOS directory that corresponds to one of the CentOS mirrors, eg: `http://mirror.centos.org/centos/7/os/x86_64/`
- Partitioning: Default is fine (LVM with three partitions /boot / swap).
- Installation option: minimal install

- Set the root password
- Installation finished? Login as root and proceed:
- To enable the hypervisor to reboot or shutdown an instance, you must install and run the acpid service on the guest system.
  - `yum install acpid`
  - `systemctl enable acpid`
- An instance must interact with the metadata service to perform several tasks on start up. For example, the instance must get the ssh public key and run the user data script. To ensure that the instance performs these tasks:
  - `yum install cloud-init`
- The account varies by distribution. On CentOS-based virtual machines, the account is called centos. You can change the name of the account used by cloud-init by editing the /etc/cloud/cloud.cfg file and adding a line with a different user, but that is probably not necessary because the default user will be centos I presume. More details: https://docs.openstack.org/image-guide/centos-image.html
- In order for the root partition to properly resize, install the cloud-utils-growpart package, which contains the proper tools to allow the disk to resize using cloud-init:
  - `yum install cloud-utils-growpart`
- Disable the zeroconf route: For the instance to access the metadata service, you must disable the default zeroconf route:
  - `echo "NOZEROCONF=yes" >> /etc/sysconfig/network`
- For the nova console-log command to work properly on CentOS 7, you might need to do the following steps:
  - Edit the /`etc/default/grub` file and configure the GRUB_CMDLINE_LINUX option. Delete the `rhgb quiet` and add `console=tty0 console=ttyS0,115200n8` to the option.
  - For example: `GRUB_CMDLINE_LINUX="crashkernel=auto rd.lvm.lv=cl/root rd.lvm.lv=cl/swap console=tty0 console=ttyS0,115200n8"`
- Run the following command to save the changes:
  - `grub2-mkconfig -o /boot/grub2/grub.cfg`
- Install Apache and makes sure that the service is enabled.
- The operating system records the MAC address of the virtual Ethernet card in locations such as `/etc/sysconfig/network-scripts/ifcfg-ens33` during the instance process. However, each time the image boots up, the virtual Ethernet card will have a different MAC address, so this information must be deleted from the configuration file (if needed).
- The storage devices in KVM need the right drivers, in this case the Virtio Block Device drivers:
  - `vi /etc/dracut.conf`
    uncommenting the `add_drivers+=` line and adding the virtio modules like this:
    `add_drivers+="virtio virtio_blk virtio_net virtio_pci"`
  - followed by the creation of a new initramfs as root:
    `dracut -f`
- We're ready: `poweroff`

Finally, convert the VMware file format to an image file in qcow2-format:

- Copy your vmdk-file to the Openstack Linux host.
- Run the following command on the Openstack Linux host to convert a vmdk image file to a qcow2 image file:
  - `qemu-img convert -f vmdk -O qcow2 image.vmdk image.qcow2`

- From there, you should be able to create a new image and launch a new instance, just like you launched Fedora. Note: flavor used, must be equal or larger than the size of your vmdk (5 GB). Memory: 512 should be enough.
- Add a floating IP, can you login using your keys, username “centos”?
- What’s the hostname of your centos system?

2.1.7 Exam material

How to prepare for the exam this week?

- **Learn** all the information to be found in the PowerPoint presentations.
- **Read** the information to be found below under the heading “information sources”.
- **Be able** to configure Openstack by doing these labs and understanding the command in general.
- **Study the Book (Architecting the cloud): chapter 1, 2, 3 and 5.**

2.1.8 Other information

- Managing SSH key-pairs: https://www.mirantis.com/blog/openstack-security-tip-create-a-keypair-for-accessing-vms/
- Volumes: https://blog.rackspace.com/laying-cinder-block-volumes-in-openstack-part-1-the-basics
- Images CentOS: https://docs.openstack.org/image-guide/centos-image.html
- Virtio: https://devops.profitbricks.com/tutorials/migrate-a-vmware-virtual-machine-running-linux-to-profitbricks/

Week 2, Lesson 1: 'DaaS Proposition'

In this lab, no Openstack exercises! Instead of building your own private cloud, sometimes it makes sense to use a public cloud. For instance, one could use a public cloud provider to host your companies' desktops, called "DaaS". In this lab, you'll compare different DaaS providers and collect information about the different offers. This information will be useful for the end assignment and the exam and it will prepare yourself for a job as system administrator / manager.

2.1.9 Daas Questions

Below some questions about DaaS providers. The first are more or less general questions, followed by a use case. Note: Part of the end assignment will be another use case.

1. Name three different DaaS Providers. In other words, create your own short list.

2. Create a table and answer all questions for each provider found:
    a. "General"
        i. Is this offer SBC or VDI
        ii. Client OS or Server OS?
        iii. Is the desktop persistent or not?
    b. "Software"
        i. Is it possible to use your own client applications?
        ii. Is MS Office included, if not what is the alternative?
    c. "Hardware"
        i. How many vCPU's can be used
        ii. How much memory?
        iii. How much storage is provided?
        iv. USB-support available?
        v. Printer redirection?
        vi. Which Display Protocol is used?
        vii. What about bandwidth
    d. Costs
        i. What are the costs per user/per hour? Startup cost? Minimal number of users required?
        ii. Price clients
    e. Management and security
        i. What about Disaster Recovery?
        ii. How scalable is this solution?
        iii. What about privacy/encryption. How is this solution secured?
        iv. Management: what management tasks should you do, and what tasks will be done by your provider?
        v. What is the SLA for this offer?
        vi. Do you need additional licenses as customer?
    f. Specific for THIS provider
        i. What are the advantages
        ii. What are the disadvantages

3. Case: Describe (maximum two pages) the solutions for the "Kwintox Case Study" described below. It is important that "AWS Workspaces" is part of the solution. More information: https://aws.amazon.com/workspaces/ . In addition, use the information available on Sharepoint, being:

a. White Paper Daas: Whitepaper_daas_020914.pdf
b. Office365 versus Google Apps: WPMicrosoftOffice365vsGoogleAppsforBusinessBDM.pdf

"Kwintox" is a company in the chemical sector. It has an office with 100 users and 20 salesmen that are always on the road. The company trades chemicals and their main applications are hosted in the cloud and available via a web browser. Currently everybody is using a laptop, but the hardware is nearly 5 years old and the manager is considering a DaaS solution. What would you suggest as being their Cloud consultant? From the 12 employees, sales people are using their desktop 4 hours / day and other personal about 6 hours/day. Beside their desktop and web browser, managing documents, e-mail and communication among colleagues is also important. BYOD would be nice.

2.1.10 Exam material

How to prepare for the exam this week?

- **Learn** all the information to be found in the PowerPoint presentations.
- **Read** the information to be found below under the heading "information sources".
- **Be able** to configure Openstack by doing these labs and understanding the command in general.
- **Study** Whitepaper_daas_020914.pdf.

2.1.11 Other information

- **Read** WPMicrosoftOffice365vsGoogleAppsforBusinessBDM.pdf about differences between MS Office 365 and Google Apps.

Week 2, Lesson 2: 'Architecture'

In this lab exercise we are going to work on a case study that will teach us using a pragmatic approach realizing a business architecture.

2.1.12 Background information

D. Jeeling (called Dar by his friends) is the son of an Indian tea planter who is studying at a well-known University of Applied Sciences in The Netherlands. During his summer holidays Dar's father (again) complains about the low prices he and his fellow planters get for their tea leaves year after year. Dar's father knows that Dar is 'doing' things that allow people to use the Cloud to sell products and services and has also understood from the newspapers that the Cloud can help people like himself in getting better prices for their products. Dar should understand that a better price for the tea leaves would make the costs of the study of Dar more bearable for his family. Dar decides to pick up this "challenge" for his family and after he has returned to the Netherlands asks his fellow students to help him putting "Tea in the Cloud". You, as one of his best friends will of course be helping Dar to create the Business Architecture for "Tea in the Cloud".

Regarding the tea business Dar has shared the following background information with you:

- Tea planters normally sell their tea to tea traders who deliberately are vague about the current market value of the tea leaves. They mainly make money because tea planters have no idea about the market value of their tea. Tea planters are very focused on the quality of the tea and not at all on selling tea.
- Tea planters don't share the prices they get for their tea leaves with fellow planters. They don't want other people to know that they get paid for several reasons.
- Some tea traders trade their tea using (local) physical auctions mostly held on tea markets.
- Indian law requires traders and buyers on physical auctions to be registered because tea is traded at the world spot market (like oil, coffee, gold, etc.) and therefor has a special trade status.

2.1.13 Case Study Questions

Identify the Problem Statement:

1. What problem are Dar and you trying to solve?
2. What are the (main) business drivers for this problem?

Evaluate the User Characteristics:

1. Who are the users?
2. What are the user needs?
3. What are the user characteristics?

Identify the Requirements:

1. Identify at least 3 business requirements
2. Identify at least 5 technical requirements

Visualize Consumer Experience:

1. What devices can and will be used by the users?

Identify Project Constraints:

1. Identify at least 2 non-artificial constraints
2. Identify at least 1 artificial constraint

Design a high-level Cloud Based architecture solving Dar's (and your) problem. Make a design and a powerpoint to present your design make sure that all your identified requirements are met and that all project constraints identified by you are covered as well. Your design including presentation will be part of the deliverables at the end of this course and will be judged accordingly.

2.1.14 OpenStack workload & redundancy

In this exercise, we are going to run an OpenStack configuration containing 2 compute nodes. For this purpose, we will extend the initial PackStack installation with an additional node, our second compute node running in a separate machine. On one of these nodes we are going to run a small CirrOS images (an instance) running a very simplistic application (an editor with a file open). As soon the simplistic application is up and running we will perform a live migration of the instance from one compute node to the other compute node running on a different virtual machine (computer).

What do you expect happens with the editor as soon as the instance is migrate from one of the virtual machines to the other virtual machine (from one of the compute nodes to the other compute node)?

____

____

Can you explain why you are expecting this?

____

Before we add the second compute node to our PackStack configuration we first have to remove our setup in the current OpenStack environment. Remove **ALL** floating ip's, routers, instances, volumes, networks and routes (do not delete Images and Flavors). Actually, you have to delete everything you have changed since the original installation as otherwise the addition of an additional compute node might fail.

Now you have to create a additional virtual machine with the following configuration details:

1. 4096 MB of memory;
2. A Host-Only network adapter with IP;
3. A NAT network adapter with IP address 192.168.37.101;
4. A disk of 20 GB;
5. Enabled nested virtualization;

The machine needs to have compute.openstack.local as its hostname an needs to run the latest version of CentOS-7 and the PackStack and OpenStack installation packages installed (do **NOT** install PackStack and/or OpenStack, PackStack will do that for you) including all necessary updates. Make sure you can access the virtual machine with SSH. When the second virtual machine is up and running make sure you have full network access (you can reach your host, the internet, the 192.168.190.0 network and the 192.168.37.0 network, resolve the hostnames compute, compute.openstack.local, packstack and packstack.openstack.local).

Configure a 2 compute node OpenStack environment:

1. Log in (e.g. by using ssh) on your first virtual machine (packstack.openstack.local);
2. Change the value for CONFIG_COMPUTE_HOSTS in your PackStack answer file by adding our new virtual machine to it. It should now become:

```
CONFIG_COMPUTE_HOSTS=192.168.190.100,192.168.190.101
```

3. Apply your modified answerfile (and again take a cup of coffee);
4. Ready? Visit http://192.168.190.100/dashboard ;

Where can you see that you actually have a second compute node?

____

Now you have to setup an cloud environment with the following properties:

1. A public network with some floating IP's;
2. A private network connected through via a router to the public network;
3. An instance connected to the private network that has access to the internet and can be accessed from your host (your PC running VMware);

How did you check your instance is fully accessible?

____

For the actual migration, you have to login on the first PackStack node. Use ssh to get access to this (virtual) computer.

What command do you use to find the ID of your instance?

____

What is the ID of your instance?

____

What command do you use to find out on which compute node your instance is running?

____

On what node is your instance running?

____

What command do you use to see on which hosts you have compute nodes running?

____

What is the hostname of the compute node that is **NOT** running your instance?

____

Now we have to check if the node that you will migrate to has sufficient resources to run your instance. Run the following command:

```
openstack host show <your_hostname>
```

and check if the node is capable of running your instance.

Before we actually migrate we have to fulfill the conditions needed to verify the first question about what we expect to happen during the migration. For this purpose we open an ssh connection (e.g. putty) to our instance and open an editor on our Cirros instance. Enter some data (text), save the file, keep the file in the editor and make some modifications in your file which you should **NOT** save. Do not close the ssh (putty) connection and keep the editor open so that you can see what happens.

Now it is time to migrate. You can execute the migration by running the following command:

```
openstack server migrate <your_instance_id> --live <your_hostname>
```

What command can you use to confirm that your instances has been migrated?

____

Now wait for the migration to finish (repeat above command).

What happened with your ssh (putty) connection?

____

What happened with the editor?

____

What happened with the changed data in the editor?

____

What happened with to saved data?

____

Was this what you originally expected?

____

2.1.15 Exam material

How to prepare for the exam this week?

- **Learn** all the information to be found in the PowerPoint presentations.
- **Read** the information to be found below under the heading “information sources”.
- **Be able** to configure what you’ve learned doing these labs and understanding the commands in general.
- **Study** chapters 4 and 13 of the book “Architecting the Cloud”.

2.1.16 Other information

- Links / info related to extra information (not for the exam)
- https://www.youtube.com/watch?v=zCX-N1H8Vps (a short recap on functional and non-functional requirements)

Week 3, Lesson 1: ‘Authentication / Federation’

In this lab you will authorize a simple web service using one or more well-known public cloud authorization API providers of your own choice.

2.1.17 Background information

One of the cloud services that is used most often, is authentication through well-known public cloud authorization API providers, for example Facebook, Google, Amazon, Twitter, LinkedIn, etc. in order to authorize use of other cloud services through OAuth.

So even though we will not be building an actual software-as-a-service solution in this lab (for now we just focus on the authorization bit), we will authorize access to a Python web application through one or more cloud authorization API’s. Authorizing through the Google and Amazon API’s has been thoroughly tested for the specific setup of this lab, other API’s not so much (but in general they are known to work with Authomatic, the library we have selected to ease OAuth functionality under the Python programming language, as demonstrated in its live demo: http://authomatic-example.appspot.com/ (this is an external site, so be careful what permissions you give it; even though they share source code, you never know what kind of malicious code might actually be running!).

In the first two paragraphs there is a lot of guidance to get the web application up and running, configuring the chosen authorization API’s will be more dependent on your own research and exploration based on the API-specific implementation documentation although we do provide pointers.

2.1.18 Setting up the software

We will be using the Flask web application framework, OpenSSL connections because most authorization API’s refuse (with reason!) to accept plain HTTP requests and redirects, and Authomatic for handling the actual authorization.
So for this exercise you need an environment with Python2 with the flask, OpenSSL and authomatic modules. These are all available through PIP, or on a CentOS 7 VM you could install the following packages (and their dependencies) using yum: `sudo yum python-flask pyOpenSSL` and then install just authomatic through PIP (`sudo python -m pip install authomatic`).
For this example you should open port 5000 in the local firewall to be able to connect to your webservice.

To test whether your Flask install is successful (and to see how easy it is to build a simple web service using it), copy the following code into a Python file:

```
from flask import Flask
app = Flask(__name__)

@         ("/")
def hello():



if __name__ == "__main__":
    app.run(host='0.0.0.0', ssl_context='adhoc')
```

Then run it with your local Python interpreter and connect to port 5000 over HTTPS to see it work (so if your IP address is 1.2.3.4, you connect to https://1.2.3.4:5000). Do not use the localhost (127.0.0.1) address for this. You should disregard any certificate warnings unless you want to create an actual certificate (in which case you will have to use a different ssl_context parameter in the `app.run()` invocation in the code above.

Now we are ready to test another trick that will come in handy later on, as for example Google will not allow you to authenticate from and to an IP address in a private range (which is very likely the case if you are using a VM or working from home): we use the NIP.IO wildcard DNS service to turn the local IP address into a fully qualified hostname. So in our example, you would connect to https://1.2.3.4.nip.io:5000 to check if this works (again disregarding certificate warnings).

2.1.19 Set up Authomatic

We use a slightly modified version of the Authomatic example code provided on the Authomatic website (at http://authomatic.github.io/authomatic/examples/flask-simple.html) which only uses data common to all authorization API's and has a broader choice of authorization API's than the example code.

So enter the following files on your system:

Configuration file `config.py`*:*

```
import authomatic
from authomatic.providers import oauth2, oauth1

CONFIG = {
    'twitter': { # Your internal provider name

        # Provider class
        'class_': oauth1.Twitter,

        # Twitter is an AuthorizationProvider so we need to set several other properties
too:
        'consumer_key': '########################',
        'consumer_secret': '########################',
    },

   'facebook': {

        'class_': oauth2.Facebook,

        # Facebook is an AuthorizationProvider too.
        'consumer_key': '########################',
        'consumer_secret': '########################',

        # But it is also an OAuth 2.0 provider and it needs scope.
        'scope': ['user_about_me', 'email', 'publish_stream'],
    },

    'amazon': {
        'class_': oauth2.Amazon,
        'consumer_key': '########################',
        'consumer_secret': '########################',
        'id': authomatic.provider_id(),
        'scope': oauth2.Amazon.user_info_scope,
    },

    'google': {
        'class_': oauth2.Google,
        'consumer_key': '########################',
        'consumer_secret': '########################',
        'id': authomatic.provider_id(),
        'scope': oauth2.Google.user_info_scope + [
            'https://www.googleapis.com/auth/calendar',
            'https://mail.google.com/mail/feed/atom',
            'https://www.googleapis.com/auth/drive',
            'https://gdata.youtube.com'],
        '_apis': {
            'List your calendars': ('GET',
'https://www.googleapis.com/calendar/v3/users/me/calendarList'),
            'List your YouTube playlists': ('GET',
'https://gdata.youtube.com/feeds/api/users/default/playlists?alt=json'),
        },
    },

    'linkedin': {
        'class_': oauth2.LinkedIn,
        'consumer_key': '##########',
        'consumer_secret': '##########',
        'id': authomatic.provider_id(),
        'scope': oauth2.LinkedIn.user_info_scope + ['rw_nus', 'r_network'],
        '_name': 'LinkedIn',
        '_apis': {
            'List your connections': ('GET',
'https://api.linkedin.com/v1/people/~/connections'),
        },
    }
}
```

Later on we will replace the `consumer_key` and `consumer_secret` fields with actual values from the identity provider API's you choose to use.

Main program file `main.py`*:*

```
from flask import Flask, render_template, request, make_response
from authomatic.adapters import WerkzeugAdapter
from authomatic import Authomatic
import authomatic
import logging

from config import CONFIG

# Instantiate Authomatic.
authomatic = Authomatic(CONFIG, 'your secret string', report_errors=False)

app = Flask(__name__, template_folder='.')
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/login/<provider_name>/', methods=['GET', 'POST'])
def login(provider_name):
    # We need response object for the WerkzeugAdapter.
    response = make_response()
    # Log the user in, pass it the adapter and the provider name.
    result = authomatic.login(WerkzeugAdapter(request, response), provider_name)

    # If there is no LoginResult object, the login procedure is still pending.
    if result:
        if result.user:
            # We need to update the user to get more info.
            result.user.update()

        # The rest happens inside the template.
        return render_template('login.html', result=result)

    # Don't forget to return the response.
    return response

# Run the app on port 5000 on all interfaces, accepting only HTTPS connections
if __name__ == '__main__':
    app.run(debug=True, ssl_context='adhoc', host='0.0.0.0', port=5000)
```

On the 10th line you replace *your secret string* with an actual secret string of your own choosing, it is not important what it actually is.

Base webpage `base.html`*:*

```
<!DOCTYPE html>
<html>
    <head>
        <title>Authomatic Flask Example</title>
    </head>
    <body>
        {% block body %}{% endblock %}
    </body>
</html>
```

Main webpage containing links to the login handlers `index.html`*:*

```
{% extends "base.html" %}
{% block body %}

    <!--Links to the login handler-->
    Login with <a href="login/amazon">Amazon</a>.<br />
    Login with <a href="login/google">Google</a>.<br />
    Login with <a href="login/linkedin">LinkedIn</a>.<br />
    Login with <a href="login/twitter">Twitter</a>.<br />
    Login with <a href="login/facebook">Facebook</a>.<br />

{% endblock %}
```

The actual authorization handler `login.html`*:*

```
{% extends "base.html" %}
{% block body
```

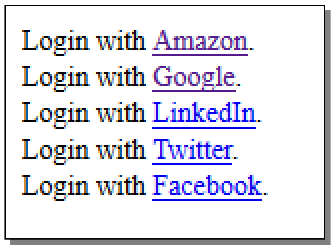
```
    <a href="/">Home</a>

    {% if result.error %}
        <h2>Damn that error: {{ result.error.message }}</h2>
    {% endif %}

    {% if result.user %}
        <h1>Hi {{ result.user.name }}</h1>
        <h2>Your id is: {{ result.user.id }}</h2>
        <h2>Your email is: {{ result.user.email }}</h2>
    {% endif %}

{% endblock body %}
```

You can now run the main.py file to see if the main page works properly (view it on the same link you used in the test in the last paragraph, so if your IP address is 1.2.3.4, go to https://1.2.3.4.nip.io:5000 (again disregarding certificate warnings) and you should see this simple page:


Login with Amazon.
Login with Google.
Login with LinkedIn.
Login with Twitter.
Login with Facebook.

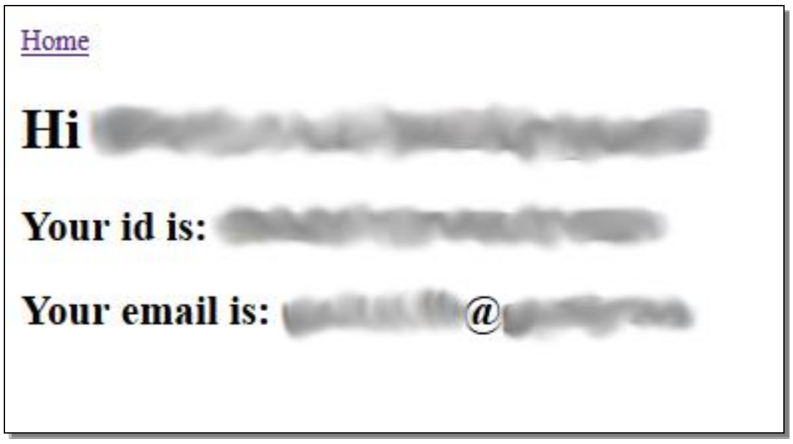
2.1.20 Set up the authorization API('s)

It is now time to setup the authorization API. .

The common steps are:

- Select which API you want to use. The code above is readymade for Amazon, Google, LinkedIn, Twitter and Facebook, if you feel adventurous you can select another one and modify the code accordingly (see Authomatic documentation);
- Get a developer account for the authorization API's provider (if you do not have one yet);
- Create a 'web application' profile with a recognizable name in the authorization API dashboard according to the relevant documentation:
    - Login with Amazon: https://developer.amazon.com/docs/login-with-amazon/web-docs.html;
    - Google Sign-In: https://developers.google.com/identity/sign-in/web/devconsole-project;
    - Sign In with LinkedIn: https://developer.linkedin.com/docs/oauth2;
    - Facebook Login: https://developers.facebook.com/docs/facebook-login/web;
    - Apps on Twitter: https://developer.twitter.com/en/docs/basics/getting-started;
- Set the right 'allowed (JavaScript) origin' (in our example https://1.2.3.4.nip.io:5000) and 'allowed return/redirect URL/URI' (in our example https://1.2.3.4.nip.io:5000/login/*<provider>*, for example https://1.2.3.4.nip.io:5000/login/google);
- The API generates a 'Client ID' and a 'Client Secret' for your application. Copy the client ID to the 'consumer_key' field of the part of the configuration that corresponds with the chosen API and the client secret tot the corresponding 'consumer_secret' field;
- Grant your app the needed permissions (for Google

- You are now (or after a restart of `main.py`) ready to test the authorization against the selected API. If it is successful you should see a page that looks like this (picture is anonymized):


Home

Hi

Your id is:

Your email is: @

If you do not get logged in or an error is shown, retrace your steps;
- If you like to, you can expand the page with more information, but for our purposes the experiment can be considered succesful;
- You can try to enable one or more additional authorization API's if you like.

2.1.21 Exam material

How to prepare for the exam this week?

- **Learn** all the information to be found in the PowerPoint presentations.
- **Read** the information to be found below under the heading "Other information".
- **Be able** to configure what you've learned doing these labs and understanding the commands in general.
- **Study** the following PDF documents that you can find on SharePoint in the "Theory" folder for this lesson:
    - A Survey Paper on Social Sign-On Protocol OAuth 2.0.pdf
    - OAuth_-_The_Big_Picture.pdf

2.1.22 Other information

- Dummy's guide for the Difference between OAuth Authentication and OpenID.pdf (SharePoint)
- Authomatic documentation: https://authomatic.github.io/authomatic/index.html
- Information on how to connect to well-known cloud authorization API providers:
    - Google Sign-In: https://developers.google.com/identity/sign-in/web/devconsole-project
    - Login with Amazon: https://developer.amazon.com/docs/login-with-amazon/web-docs.html
    - Sign In with LinkedIn: https://developer.linkedin.com/docs/oauth2
    - Facebook Login: https://developers.facebook.com/docs/facebook-login/web

Week 3, Lesson 2: 'Storage / Cloud Management (Techn.)'

In this lab exercise we are going to use a NoSQL database. We have selected MongoDB, a document orientated document type database. We first will install a MongoDB environment locally on our desktop. Within this development we will run through a number of exercises to learn how MongoDB is used.

2.1.23 Install MongoDB

Follow the steps on the following webpage to install MongoDB CE (Community Edition). For the exercises in this lab session you can use a Linux (CentOS / Ubuntu / macOS) based installation of MongoDB. In later exercises (when building a simple REST interface) we will use a Linux (CentOS / Ubuntu / macOS) based environment.

https://docs.mongodb.com/manual/installation/#tutorial-installation

After you have installed the environment you should setup the environment according to the instructions and started the MongoDB service.

2.1.24 The `mongo` shell

Import the Example dataset as instructed on the following webpage (**NOTE**: make sure your MongoDB service is running).

https://docs.mongodb.com/getting-started/shell/import-data/

Start a `mongo` shell

Start a mongo shell and test if your database is up and running.

See also: https://docs.mongodb.com/getting-started/shell/client/

```
>use test
>db.restaurants.find()
```

The mongo shell should provide you a list of (JSON formatted) restaurant records like the one below.

```
{ "_id" : ObjectId("59db291353122d83ff0a74a7"), "address" : { "building" : "351",
"coord" : [ -73.98513559999999, 40.7676919 ], "street" : "West   57 Street", "zipcode" :
"10019" }, "borough" : "Manhattan", "cuisine" : "Irish", "grades" : [ { "date" :
ISODate("2014-09-06T00:00:00Z"), "grade" : "A", "score" : 2 }, { "date" : ISODate("2013-
07-22T00:00:00Z"), "grade" : "A", "score" : 11 }, { "date" : ISODate("2012-07-
31T00:00:00Z"), "grade" : "A", "score" : 12 }, { "date" : ISODate("2011-12-
29T00:00:00Z"), "grade" : "A", "score" : 12 } ], "name" : "Dj Reynolds Pub And
Restaurant", "restaurant_id" : "30191841" }

... more records ...

{ "_id" : ObjectId("59db291353122d83ff0a74ba"), "address" : { "building" : "129-08",
"coord" : [ -73.839297, 40.78147 ], "street" : "20 Avenue", "zipcode" : "11356" },
"borough" : "Queens", "cuisine" : "Delicatessen", "grades" : [ { "date" : ISODate("2014-
08-16T00:00:00Z"), "grade" : "A", "score" : 12 }, { "date" : ISODate("2013-08-
27T00:00:00Z"), "grade" : "A", "score" : 9 }, { "date" : ISODate("2012-09-
20T00:00:00Z"), "grade" : "A", "score" : 7 }, { "date" : ISODate("2011-09-
29T00:00:00Z"), "grade" : "A", "score" : 10 } ], "name" : "Sal'S Deli", "restaurant_id"
: "40361618" }
>
```

You now have a working mongo database filled with some data that we will use for the rest of the exercises.

Now insert a new record with the following data:

```
RED ROOSTER HARLEM
American Restaurant
130 Lenox Ave, New York, NY 10027
USA
Latitude: 40.808057
Longitude: -73.944914
```

And print the data using the pretty() function. You result should look like the output below.

```
>db.restaurants.find({"address.zipcode":"10027"}).pretty()
{
        "_id" : ObjectId("************************"),
        "address" : {
                "building" : "351",
                "coord" : [
                        -73.94491390000002,
                        40.8080566
                ],
                "street" : "130 Lenox Ave",
                "zipcode" : "10027"
        },
        "borough" : "Harlem",
        "cuisine" : "American",
        "grades" : [
                {
                        "date" : ISODate("2017-09-06T00:00:00Z"),
                        "grade" : "A",
                        "score" : 2
                }
        ],
        "name" : "RED ROOSTER",
        "restaurant_id" : "90191868"
}
>
```

Update the address with the following data:

```
310 Lenox Ave
```

Again, pretty print the data and verify your output.

2.1.25 Mongo and Python

NOTE: This exercise is also partly a preparation for the lab exercises that belongs to Week4-Lesson1 (Interfacing). Here we will use python to integrate MongoDB into a web server that provides a RESTful interaface.

Now we are going to use MongoDB in combination with Python. First we have to make sure that we have installed the mongodb package for Python. Python is one of the official supported drivers for MongoDB.

For instructions on how to install the python driver see:

https://docs.mongodb.com/ecosystem/drivers/python/

NOTE: make sure you select an installation that fits with your preferred python environment.

Create the following python program in your python environment:

```
import pymongo
import pprint

from pymongo import MongoClient
client = MongoClient()

db = client.test
collection = db.restaurants

pprint.pprint(collection.find_one())
```

This should result in the following result:

```
{u'_id': ObjectId('59db291353122d83ff0a74a7'),
 u'address': {u'building': u'351',
              u'coord': [-73.98513559999999, 40.7676919],
              u'street': u'West   57 Street',
              u'zipcode': u'10019'},
 u'borough': u'Manhattan',
 u'cuisine': u'Irish',
 u'grades': [{u'date': datetime.datetime(2014, 9, 6, 0, 0),
              u'grade': u'A',
              u'score': 2},
             {u'date': datetime.datetime(2013, 7, 22, 0, 0),
              u'grade': u'A',
              u'score': 11},
             {u'date': datetime.datetime(2012, 7, 31, 0, 0),
              u'grade': u'A',
              u'score': 12},
             {u'date': datetime.datetime(2011, 12, 29, 0, 0),
              u'grade': u'A',
              u'score': 12}],
 u'name': u'Dj Reynolds Pub And Restaurant',
 u'restaurant_id': u'30191841'}
```

You now know your python/MongoDB is working. Now it is time to start with some real work.

2.1.26 The case

We now want you to create a loyalty program service application for the Dutch market. Dutch people love loyalty programs (collecting coupons) something that already dates back to the 19th century. Our current client is a large producer of coffee that wants to migrate from paper coupons to an electronic savings program.

The basic idea is that a customer registers her/himself (contact details) and can enter a code that is printed on a product of our coffee producer as an alternative to cut and safe paper coupons. This code can of course only be used once and has a certain value. In a later (second) release the coffee producer wants to introduce QR codes as an extra service next to typing in the code.

You have been asked to create a database together with the necessary functions (code) that allows for the registration of customers together with the registration of codes. We advise you to make a small design (architecture) for this customer loyalty service as well.

Make sure that you have functions to create / insert, read / select, update and delete records in your database.

As result we expect a python program that tests the python functions you have created. These python functions will be used in a next lesson to create a web interface for the customer loyalty service.

NOTE: Carefully store the functions and programs you have created as you will need them in a next lab exercise.

2.1.27 Exam material

How to prepare for the exam this week?

- **Learn** all the information to be found in the PowerPoint presentations.
- **Read** the information to be found below under the heading "information sources".
- **Be able** to configure what you've learned doing these labs and understanding the commands in general.
- **Study** chapters 8 and 12 of the book "Architecting the Cloud".

2.1.28 Other information

- Links / info related to extra information (not for the exam)
- An introduction in MongoDB: https://docs.mongodb.com/manual/introduction/

Week 4, Lesson 1: 'Interfacing with the cloud'

During this week's lab exercise we are going to create a RESTful web API based on the work we already had to do for the customer loyalty service / application. You have to use a Linux (CentOS / Ubuntu / macOS) based environment to execute this week's exercises.

Next to MongoDB we are also going to use Flask and Flask-RESTful for this exercise.

2.1.29 Install Flask and FlaskRESTful (in your python environment)

Install Flask:
http://flask.pocoo.org

Install Flask-RESTful:
https://flask-restful.readthedocs.io/en/latest/installation.html)

Test the following python webserver application:

```python
from flask import Flask
from flask_restful import Resource, Api

app = Flask(__name__)
api = Api(app)

class HelloWorld(Resource):
    def get(self):
        return {'hello': 'world'}

api.add_resource(HelloWorld, '/')

if __name__ == '__main__':
    app.run(debug=True)
```

In order to test you should run this program and use curl or your browser to see what is returned by the webserver. Curl will return:

```
$ curl http://localhost:5000/
{
    "hello": "world"
}
$
```

Your environment is now ready to create your own RESTful web service (server).

2.1.30 The Case

The (Mobile) App Developers of our coffee producers would like to start developing the App and would like to know from us how the API for the database will work. They already know that we will provide a RESTful API and would like to know how they should use the API.

Your first task is to provide the interface details (REST Client endpoints, URLs, arguments, etc.)

Please complete the table below:

URL	**Data**	**Comment**
`http://<host>:port/<path>/<id>`	`name = <name>`	`Register a customer`
		`Remove a customer`
		`Update customer data`
		`Verify code`

Because the developers now have your API you can expect that within a few weeks they want to test your RESTful API. You now have to make sure there is something available for them to test.

Create a python application that implements your RESTful API. Use MongoDB as your database (See Week3-Lesson2) and use Flask-RESTful to implement your API. Use the Postman application to test your RESTful API against the specifications you have provided to the App developers.

We expect to see documented test results for all the endpoints that are defined in your RESTful API (make use of Postman).

2.1.31 In the Cloud

You have been using a local installation of the MongoDB database. You could have used a Cloud based version of MongoDB as well. MongoDB Atlas is a Database as a Service (DaaS) offering the same functionality as your local installation.

1. Can you briefly explain what will change in your RESTful server application?
2. Can you motivate where you prefer to run your RESTful server application?
3. Can you briefly explain if and why or why not this fits into an OpenStack (like) environments?

2.1.32 Exam material

How to prepare for the exam this week?

- **Learn** all the information to be found in the PowerPoint presentations.
- **Read** the information to be found below under the heading “information sources”.
- **Be able** to configure what you’ve learned doing these labs and understanding the commands in general.
- **Study** chapters 6, 14 and 16 of the book “Architecting the Cloud”.

2.1.33 Other information

- As a SN(C)E engineer you will most likely have to deal with RESTful APIs. These APIs are quite commonly used to configure cloud infrastructures. It is even very likely that you will be implementing RESTful APIs for high level configuration purposes (on top of the already existing RESTful API).

Week 4, Lesson 2: ‘Containerization’

This week’s first exercise will make you orchestrate containers. We will be using Docker Swarm (https://docs.docker.com/engine/swarm/).

2.1.34 Setup Docker Swarm

You will need the following:

- four Linux (CentOS) hosts as (small) virtual machines which can communicate over a network, with Docker installed
- Docker Engine 1.12 or later installed on all the hosts
- the IP address of the manager machine
- open ports between the hosts (virtual machines)
    1. TCP port 2377 for cluster management communications
    2. TCP and UDP port 7946 for communication among nodes
    3. UDP port 4789 for overlay network traffic

2.1.35 Create a swarm

```
$ docker-machine ssh <MANAGER-IP>
$ docker swarm init --advertise-addr <MANAGER-IP>
```

add workers and a manager to this swarm and view (on the manager node!) information about nodes:

```
$ docker node ls
ID                           HOSTNAME  STATUS  AVAILABILITY  MANAGER STATUS
<id4>                        <work3>   Ready   Active
<id3>                        <work2>   Ready   Active
<id2>                        <work1>   Ready   Active
<id1> *                      <man1>    Ready   Active        Leader
```

2.1.36 Use the swarm

We want you to deploy a service with 1 running instance to a Docker swarm. You are free to choose any service from Docker as long as it is capable of providing some external output (your swarm will use routing mash and will have published ports).

Please write down the commands you used, why you used the command and show the output of the commands when executing the following actions:

1. Draw a diagram of your swarm configuration;
2. Show the list of running services;
3. Show the details of the running services in JSON format;
4. Show which nodes are running the service;
5. Scale your service to 7 instances;
6. Downgrade your service to a stable lower release because the current release of the service you selected has contains a critical security leak;
7. One of nodes needs hardware maintenance. Please make this node available, physically reboot the virtual machine and return the node to the swarm and make sure it is running an instance of your service again. Is your configuration capable of hardware maintenance on the master node?

8. Reduce the number of instances in your swarm so that one node is not running an instance of your service. Access the output on the published ports on the node that is not running an instance of your service. Are you getting any result?

2.1.37 Setup SaltStack

Install three CentOS server instances in your VMware environment similar to the server you have installed for PackStack/OpenStack. You however only need one (1) NIC interface (host-only) and a smaller footprint for each server instance. Furthermore make sure your installed servers are updated to the latest stable status.

The server characteristics for this lesson are:

- OS: CentOS 7
- Memory: 2048MB
- Disk: 20 GB
- NIC: host-only

The 3 servers should be installed as follows:

1. SaltStack-master:
   - hostname: master
   - IP: 192.168.37.10
   - users: root & non-root sudo user

2. SaltStack-minion1:
   - hostname: minion1
   - IP: 192.168.37.10
   - users: root & non-root sudo user

3. SaltStack-minion2:
   - hostname: minion2
   - IP: 192.168.37.10
   - users: root & non-root sudo user

Note: Make sure all hosts are reachable by each instance of a server and can also be reached by their hostnames. Furthermore you will have to access the systems through the non-root sudo user (so do not use root!)

On the salt master host install:

```
curl -L https://bootstrap.saltstack.com -o install_salt.sh
sudo sh install_salt.sh -P -M
```

Change the salt configuration file /etc/salt/master as follows:
Find the following line:

```
#interface: 0.0.0.0
```

and replace by:

```
interface: 192.168.37.10
```

On each salt minion host install:

```
curl -L https://bootstrap.saltstack.com -o install_salt.sh
sudo sh install_salt.sh -P
```

Change the salt configuration file /etc/salt/master as follows:
Find the following line:

```
#master: salt
```

and replace by:

```
master: master
```

We are now ready to (re)start salt by running the following commands:

```
pkill salt-master
salt-master –d
```

On the minions run:

```
pkill salt-minion
salt-minion -d
```

The first time the daemon is stopped (it should be running as well). The second time your restart and update the daemon. If you don’t get the message the daemon was stopped during the first command you just started it (as it apparently was not running).

Now you have perform some key management to enable the minions on the master. Type the following command on the salt master:

```
salt-key –L
```

You now should see:

```
Accepted Keys:
Denied Keys:
Unaccepted Keys:
minion1.saltstack.local
minion2.saltstack.local
Rejected Keys:
```

To accept the minions type the following command on the salt master:

```
salt-key –A
```

If you now list the keys on the master again you should see:

```
Accepted Keys:
Denied Keys:
Unaccepted Keys:
minion1.saltstack.local
minion2.saltstack.local
Rejected Keys:
```

As a final step we verify is everything is running by sending a command to the minions. Run the following command on the salt master:

```
Salt ‘*’ test.ping
```

This should return 'True' for all minions connected to and accepted by the salt master.
Now our salt configuration with one master and two minions is ready.

2.1.38 A Salt exercise

2.1.38.1 Setting up the Salt State Tree

States are stored in text files on the master and transferred to the minions on demand via the master's File Server. The collection of state files make up the State Tree.

To start using a central state system in Salt, the Salt File Server must first be set up. Edit the master config (/etc/salt/master) file (file_roots) and uncomment the following lines:

```
file_roots:
  base:
    - /srv/salt
```

Restart the Salt master in order to pick up this change:

```
pkill salt-master
salt-master -d
```

2.1.38.2 Prepare the top file

On the master, in the directory uncommented in the previous step, (/srv/salt by default), create a new file called top.sls and add the following:

```
base:
  '*':
    - webserver
```

The top file is separated into environments. The default environment is base. Under the base environment a collection of minion matches is defined; for now simply specify all hosts

2.1.38.3 Create a SLS file

In the same directory as the top file, create a file named webserver.sls containing the following:

```
httpd:                  # ID declaration
  pkg:                  # state declaration
    - installed         # function declaration
```

The first line, called the ID declaration, is an arbitrary identifier. In this case it defines the name of the package to be installed.
The second line, called the State declaration, defines which of the Salt States we are using. In this example, we are using the pkg state to ensure that a given package is installed.
The third line, called the Function declaration, defines which function in the pkg state module to call.

2.1.38.4 Install the package

Next, let's run the state we created. Open a terminal on the master and run:

```
salt '*' state.apply
```

Our master is instructing all targeted minions to run state.apply. When this function is executed without any SLS targets, a minion will download the top file and attempt to match the expressions within it. When the minion does match an expression the modules listed for it will be downloaded, compiled, and executed.
Once completed, the minion will report back with a summary of all actions taken and all changes made.

Your output should contain:

```
minion2.saltstack.local:
----------
          ID: httpd
    Function: pkg.installed
      Result: True
     Comment: The following packages were installed/updated: httpd
     Started: 22:02:25.101731
    Duration: 45909.212 ms
     Changes:
              ----------
              apr:
                  ----------
                  new:
                      1.4.8-3.el7
                  old:
              apr-util:
                  ----------
                  new:
                      1.5.2-6.el7
                  old:
              httpd:
                  ----------
                  new:
                      2.4.6-67.el7.centos.6
                  old:
              httpd-tools:
                  ----------
                  new:
                      2.4.6-67.el7.centos.6
                  old:
              mailcap:
                  ----------
                  new:
                      2.1.41-2.el7
                  old:

Summary for minion2.saltstack.local
------------
Succeeded: 1 (changed=1)
Failed:    0
------------
Total states run:     1
Total run time:  45.909 s
minion1.saltstack.local:
----------
          ID: httpd
    Function: pkg.installed
      Result: True
     Comment: The following packages were installed/updated: httpd
     Started: 22:02:25.104744
    Duration: 50267.416 ms
```

```
     Changes:
              ----------
              apr:
                  ----------
                  new:
                      1.4.8-3.el7
                  old:
              apr-util:
                  ----------
                  new:
                      1.5.2-6.el7
                  old:
              httpd:
                  ----------
                  new:
                      2.4.6-67.el7.centos.6
                  old:
              httpd-tools:
                  ----------
                  new:
                      2.4.6-67.el7.centos.6
                  old:
              mailcap:
                  ----------
                  new:
                      2.1.41-2.el7
                  old:

Summary for minion1.saltstack.local
------------
Succeeded: 1 (changed=1)
Failed:    0
------------
Total states run:     1
Total run time:  50.267 s
```

2.1.38.5 Call multiple states

You can specify multiple State declaration under an ID declaration. For example, a quick modification to our webserver.sls to also start Apache if it is not running:

```
httpd:
  pkg.installed: []
  service.running:
    - require:
      - pkg: httpd
```

Try stopping Apache before running state.apply once again and observe the output. How can you see that the httpd has started?

2.1.38.6 Require other states

We now have a working installation of Apache so let's add an HTML file to customize our website. It isn't exactly useful to have a website without a webserver so we don't want Salt to install our HTML file until Apache is installed and running. Include the following at the bottom of your webserver/init.sls file:

```
httpd:
  pkg.installed: []
  service.running:
    - require:
      - pkg: httpd

/var/www/html/index.html:                   # ID declaration
  file:                                     # state declaration
    - managed                               # function declaration
    - source: salt://webserver/index.html   # function arg
    - require:                              # requisite declaration
      - pkg: httpd                          # requisite reference
```

- **ID declaration:** In this example it is the location we want to install our custom HTML file.
- **State declaration:** This example uses the Salt file state.
- **Function declaration.**:The managed function will download a file from the master and install it in the location specified.
- **Function arg**: The declaration which, in this example, passes the source argument to the managed function.
- **Requisite declaration.**
- **Requisite reference:** which refers to a state and an ID. In this example, it is referring to the ID declaration from our first example. This declaration tells Salt not to install the HTML file until Apache is installed.

Next, create the index.html file and save it in the webserver directory:

```
<!DOCTYPE html>
<html>
    <head><title>Salt rocks</title></head>
    <body>
        <h1>This file brought to you by Salt</h1>
    </body>
</html>
```

Last, call state.apply again and the minion will fetch and execute the highstate as well as our HTML file from the master using Salt's File Server:

```
salt '*' state.apply
```

Verify that Apache is now serving your custom HTML by accessing the webserver on the minions.

2.1.39 Exam material

How to prepare for the exam this week?

- **Learn** all the information to be found in the PowerPoint presentations.
- **Read** the information to be found below under the heading “information sources”.
- **Be able** to configure what you’ve learned doing these labs and understanding the commands in general.

2.1.40 Other information

- All about Docker (https://www.docker.com/).

Week 5, Lesson 1: ‘Cloud Security’ / Google Cloud Platform / AWS

This lesson is about cloud security, but to avoid becoming too theoretical the lab is accompanied by a practical assignment using the Google Cloud Platform. This is a greate opportunity for students getting some experience related to public clouds, because for using a public cloud, normally a credit card is needed…

2.1.41 Disaster Recovery and Public Cloud providers

Introduction:

Selecting the best Disaster Recovery options for your situation can be difficult: It depends on the service provided by your cloud provider and of course your needs with reference to RPO, RTO and Value of the cloud services to the Cloud consumer. Part of the end assignment might be selecting a disaster recovery scenario and a cloud provider. The information that you gather during this task can be used as input for the end assignment if you like.

Task 1: What DR scenario would you choose?

Collect information about the three public cloud providers mentioned below and process the information so that it can be visualized in a table.

Consider the following case: The value of the services provided is high, RPO and RTO are short (e.g. 1 hour at most) and the service provided is a three-tier web application (Web server for presentation | Application Server for business logic | Database Server like SQL for storage).

First question: What scenario would you choose? (Cold / Warm standby (passive), Active/Active, or….). Describe the scenario in three sentences.

Task 2: Fill the table

Public cloud providers to be considered: *Amazon AWS, Microsoft Azure* and *Google Cloud Platform*.

The Information that you collect and present in a table must be related to:

- Services already provided out-of-the-box by the cloud provider (do you have to create the whole solution by yourself or has the cloud provider created some functionality);
- Costs (estimate the costs roughly);
- DNS (how do customers know that they should use the disaster environment);
- Load Balancing (maybe related to DNS?);
- Regions/Availability Zones/Datacenters (is this relevant);
- SQL Databases (e.g. information about users like addresses);
- File Storage (e.g. images uploaded by customers);
- Automation (e.g. automated the whole DR process);
- Failure detection (how can you know);

Table: Should be something like the table presented below (Max 2x A4-format).

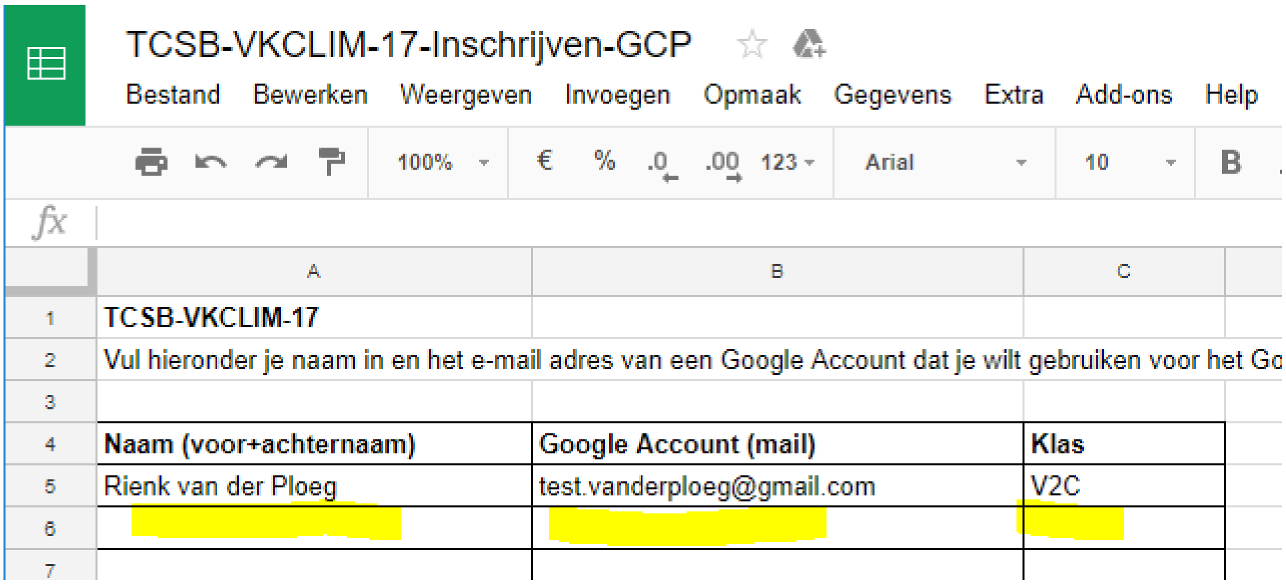
Item	Amazon AWS	Microsoft Azure	Google Cloud Platform
Provided Services			
Costs			
DNS			
…etc…			

2.1.42 Google Public Cloud

In this lab we'll use the Google Public Cloud to create a secure Virtual Machine. Purpose of this lab is that you'll experience how easy it is to create a new VM. Moreover, you will see differences and similarities between Openstack and the Google Cloud Platform. Ready?!

Step 1: Onboarding

Add your name, google account (create one if needed!) and class to the spreadsheet:
https://docs.google.com/spreadsheets/d/1brbMGsTUMhgbSBe3RrNG8kpBjxwtAys_bPpVEDATOnI/edit?usp=sharing



	A	B	C
1	TCSB-VKCLIM-17		
2	Vul hieronder je naam in en het e-mail adres van een Google Account dat je wilt gebruiken voor het Go		
3			
4	**Naam (voor+achternaam)**	**Google Account (mail)**	**Klas**
5	Rienk van der Ploeg	test.vanderploeg@gmail.com	V2C
6			
7			

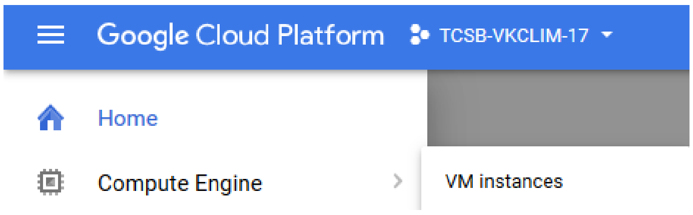
You teacher should give you access to the Google Cloud. Be careful user rights are valid for all VMs within the project, including VMs of other students. Be careful and don't touch VMs you did not create and stop VMs once your ready (don't delete them). After you finish this lab, create screenshots and stop your VMs. Don't wait too long: This environment might not be available for you next week!

Step 2: Log on

- **Log on** to the cloud platform: https://console.cloud.google.com (https://console.cloud.google.com/user-preferences/languages?project=tcsb-vkclim-17&pli=1)
- **Choose English (United States)**

Step 3: Create a VM using the console

- **Create a new VM instance:**



- **Some remarks:**
  - **Name instance** "instance-<firstname><lastname>"
  - **No service account**

    - **CentOS7** is fine.
    - **Zone:** Europe if fine.
    - **Choose a cheap instance, e.g: "micro"** with 0.6 GB memory.
    - **Costs:** about 5 or 6 dollars / month?

Step 4: Google Cloud Shell

- **Create another VM** with the same specs, but this time **using the Google Cloud Shell.** You can use this information if you like**:** https://cloud.google.com/sdk/gcloud/reference/compute/instances/create
- **Some remarks:**
    - **Name instance** "instance-gcloud-<firstname><lastname>"
    - **Machine-type:** 'f1-micro'

Step 5: Create a webserver.

- Logon from putty using SSH to one of you newly create VMs.
- Install a webserver, e.g. via something like: https://cloud.google.com/compute/docs/tutorials/basic-webserver-apache
- You should be able to connect to your website from the Internet via a public IP. Is this possible?

Step 6: Screenshots

- Create a few screenshots for your end assignment proofing:
    - Connectivity using putty
    - Google cloud shell command
    - Your running instances (chrome screenshot of your running VMs on https://console.cloud.google.com
- Stop all your VMs! (delete is not needed)

2.1.43 AWS

Make sure that you have a working AWS Educate Starter Account: See: 2.1.59
Create the same VM (CentOS7 with an Apache Webserver) using AWS EC2.

2.1.44 Exam material

How to prepare for the exam this week?

- **Learn** all the information to be found in the PowerPoint presentations.
- **Read** the information to be found below under the heading "Other information".
- **Be able** to configure what you've learned doing these labs and understanding the commands in general.
- **Study chapters 7, 8, 9, 10 and 13 of the book "Architecting the Cloud**".

2.1.45 Other information

Links / info related to extra information (not for the exam):

- OWASP 2017: https://github.com/OWASP/Top10/blob/master/2017/OWASP%20Top%2010%202017%20RC2%20Final.pdf
- DR AWS: https://cloudacademy.com/blog/disaster-recovery-using-aws
- DR Azure: https://docs.microsoft.com/en-us/azure/architecture/resiliency/disaster-recovery-azure-applications & https://docs.microsoft.com/en-us/azure/best-practices-availability-paired-regions

- DR GCP: https://cloud.google.com/solutions/disaster-recovery-cookbook

Week 5, Lesson 2: 'ITSM / Metrics'

In a cloud environment people pay for what they use. The cloud service provider must provide the right metrics. In this lab, we'll investigate the SLA/KPIs of Amazon and see what we can provide in our own OpenStack Private cloud environment.

2.1.46 SLA Amazon for IaaS (VMs)

Examine Amazon's SLA for the use of virtual machines: https://aws.amazon.com/ec2/sla/ (?)
Compare its commitments and service credits with the Microsoft Azure SLA.

1. What's the difference?

   ____

2. What should the customer do if a VM is unavailable and he/she would like to receive a Credit Request?

   ____

3. What's the difference between a Credit Request and a Refund?

   ____

4. Mention three reasons to any unavailability that Amazon has excluded from its SLA:
   a. ____
   b. ____
   c. ____

2.1.47 Costs (VMs)

Use the Rackspace calculator (https://www.rackspace.com/calculator) to determine the costs for the following case description:

A company would like to hosts it website using Rackspace's public cloud. It is important that all content remain in Europe. The website is using a MySQL database, a backup service is needed, and in case the website will get many users: a load balancer should be in place. We'll manage the environment ourselves.

1. Costs: ____

   ____
   ____
   ____
   ____
   ____

2. Issues: What's the problem if you would like to specify the cost of this infrastructure in advance?

   ____
   ____
   ____
   ____
   ____

2.1.48 Metrics (OpenStack)

Metrics are provided out-of-the-box, using ceilometer and other components/tooling.

Usage Information provided by Horizon

Use horizon (website) to get the following information without additional components:

1. Total of VCPU Usage (Hours) last week as a total per project: ______________________

2. The usage in hours of all your VMs (per VM) in the admin project:

   ____

   ____

   ____

Metrics provided by Ceilometer
Usage information is probably not containing enough detail to send an invoice to your customers. Therefore, we have to use the telemetry services of OpenStack provided by ceilometer. Don't forget to "dot source" the `keystonerc_admin` file before continuing this lab:
Use one of the following commands to answer the questions below. Maybe you'll need multiple commands, find the parameters needed using google or using: `command <help> <subcommand>`

```
ceilometer sample-list --meter METER_NAME
ceilometer resource-list
ceilometer resource-show <recourceid>
ceilometer meter-list
ceilometer meter-list --query resource=<recourceid>
ceilometer event-list
ceilometer statistics
ceilometer statistics --meter METER_NAME
ceilometer statistics --meter cpu_util --query 'resource_id=<recourceid>;timestamp>2015-
01-12T13:00:00;timestamp<=2015-01-13T14:00:00'
ceilometer query-samples -f '{"=": {"counter_name":"cpu_util"}}'
ceilometer -d query-samples -f '{"and": [{"=": {"counter_name":"cpu_util"}}, {">":
{"counter_volume": 2.00}}, {"<": {"counter_volume": 8.00}}] }'
```

1. What meters are available regarding the network?
2. What is the difference between the meters disk.read.bytes / disk.read.bytes.rate / disk.read.requests ?
3. What is the difference between a gauge type and a cumulative meter?
4. What is the resource id from your first VM?
5. What meters are available for this VM?
6. Fetch all samples that are available for this specific VM
7. What is the delta (in minutes) between the different cpu meter samples?
8. Fetch all samples of today regarding the cpu.
9. Fetch the average cpu load of today for this VM.
10. Fetch the total read bytes for this VM today.

2.1.49 Rating and Billing (OpenStack)
Now that we can measure usages, it is possible to send an invoice to our customers. First step for this is to rate the difference components. The OpenStack way to do this is using CloudKitty, but this is not production ready. Therefore, we'll use simple scripts provided by Rackspace and the author of this document to do the billing of your admin project.
Follow the steps needed below, or read the manual on https://github.com/rackerlabs/openstack-usage-report

First step: install the software:
We'll need git to download the script from the Internet:

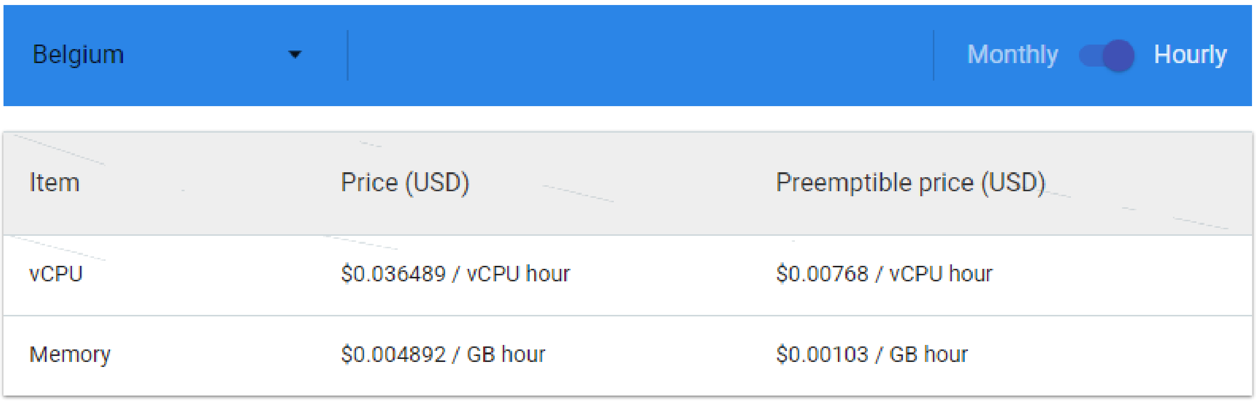
```
git clone https://github.com/rackerlabs/openstack-usage-report
cd openstack-usage-report/
mkdir -p /etc/usage
cp sample-usage.yaml /etc/usage/usage.yaml
cp sample-report.yaml /etc/usage/report.yaml
python setup.py develop

Modify /etc/usage/usage.yaml so that it resembles your admin configuration. You'll find
the data you need in the file ~/keystonerc_admin
```

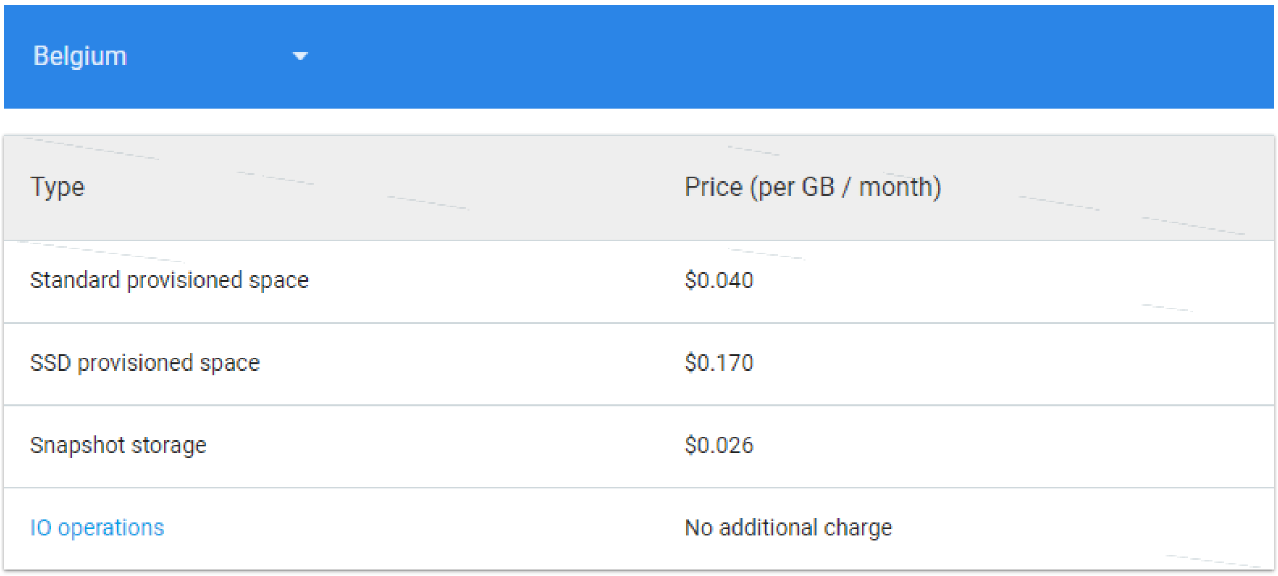
Next step: do the rating:
Create your first report by executing the command: `usage-report --today`
Result to be found in /`etc/usage/reports/`
Load the report in excel (using the right country settings). How much money did you spent today?

Let's make this better and more realistic by:

- modifying currency (EUR),
- modification of the item_rate. Use the information provided from the images below (real data information). If you need more data, visit the Google page providing prices: https://cloud.google.com/compute/pricing?hl=nl#custommachinetypepricing



Item	Price (USD)	Preemptible price (USD)
vCPU	$0.036489 / vCPU hour	$0.00768 / vCPU hour
Memory	$0.004892 / GB hour	$0.00103 / GB hour



Type	Price (per GB / month)
Standard provisioned space	$0.040
SSD provisioned space	$0.170
Snapshot storage	$0.026
IO operations	No additional charge

FIGURE 3: SOURCE: HTTPS://CLOUD.GOOGLE.COM/COMPUTE/PRICING?HL=NL#CUSTOMMACHINETYPEPRICING

Now, create a new report, based upon the new settings. How much money did you spent today?

Final step: create your first invoice:
Use the script `create_invoice.py` provided on SharePoint to convert the csv to an ASCII-invoice.
Read the script first and **modify** it so that it suits your needs!
What is the total price that your customer for the admin project has to pay for this month? Tip: first use the `usage-report` script with the right arguments, followed by running the `create_invoice.py` script to generate the invoice.

Disclaimer: The purpose of these scripts is to show you what can be done. It cannot be used in a production environment of course.

This concludes the lab of this lesson. We were able to: collect metrics, rate items and convert these items to an invoice. Seems that we got all the information to create our own public service provider today!

2.1.50 Exam material

How to prepare for the exam this week?

- **Learn** all the information to be found in the PowerPoint presentations.
- **Read** the information to be found below under the heading “information sources”.
- **Be able** to configure what you’ve learned doing these labs and understanding the commands in general.
- **Study** chapters 11, 14 and 15 of the book “Architecting the Cloud”.

2.1.51 Other information

- Managing_Resource_Accountability-OpenStack.pdf (SharePoint)
- `create_invoice.py` (SharePoint)

Week 6, Lesson 1: 'Recht'

Deze les wordt gegeven in het Nederlands, omdat Nederlands recht van toepassing is.

2.1.52 Lab with title

Zoek op internet een actuele casus, waarbij PRIVACY een rol speelt.

Beantwoord de volgende vragen:

- Beschrijf de situatie
- Wat is de RECHTSVRAAG?
- Welke partijen spelen
- Wat zijn de meningen?
- Welke rechtsregels spelen er?
- Hoe moet je die in deze causus uitleggen/toepassen?
- Wat is jouw mening, op basis van de rechtsregels. En Waarom?

Bereid een presentatie voor (ca. 5-10 min).

2.1.53 Exam material

How to prepare for the exam this week?

- **Learn** all the information to be found in the PowerPoint presentations.

2.1.54 Other information

- Links / info related to extra information (not for the exam)

Week 6, Lesson 2: ‘Exam training / Future / Guest Lesson / Datacenter visit’

At the moment of writing it is unclear what will be done during this lesson. Might be a combination of two guest lessons and a datacenter visit, or something else. There is a possibility that these activities are scheduled at another date/time then your standard schedule.

2.1.55 Guest Lesson

You must attend the guest lessons

2.1.56 Data Center Visit

You must attend the data center visit if this is organized.

2.1.57 Exam material

How to prepare for the exam this week?

- **Attend** guest lessons and datacenter visit if applicable.
- If you cannot attend ask your teacher about the consequences (e.g. **replacement assignment**).

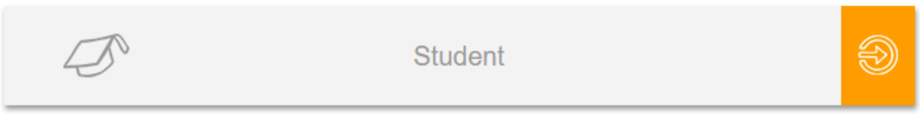
2.1.58 Other information

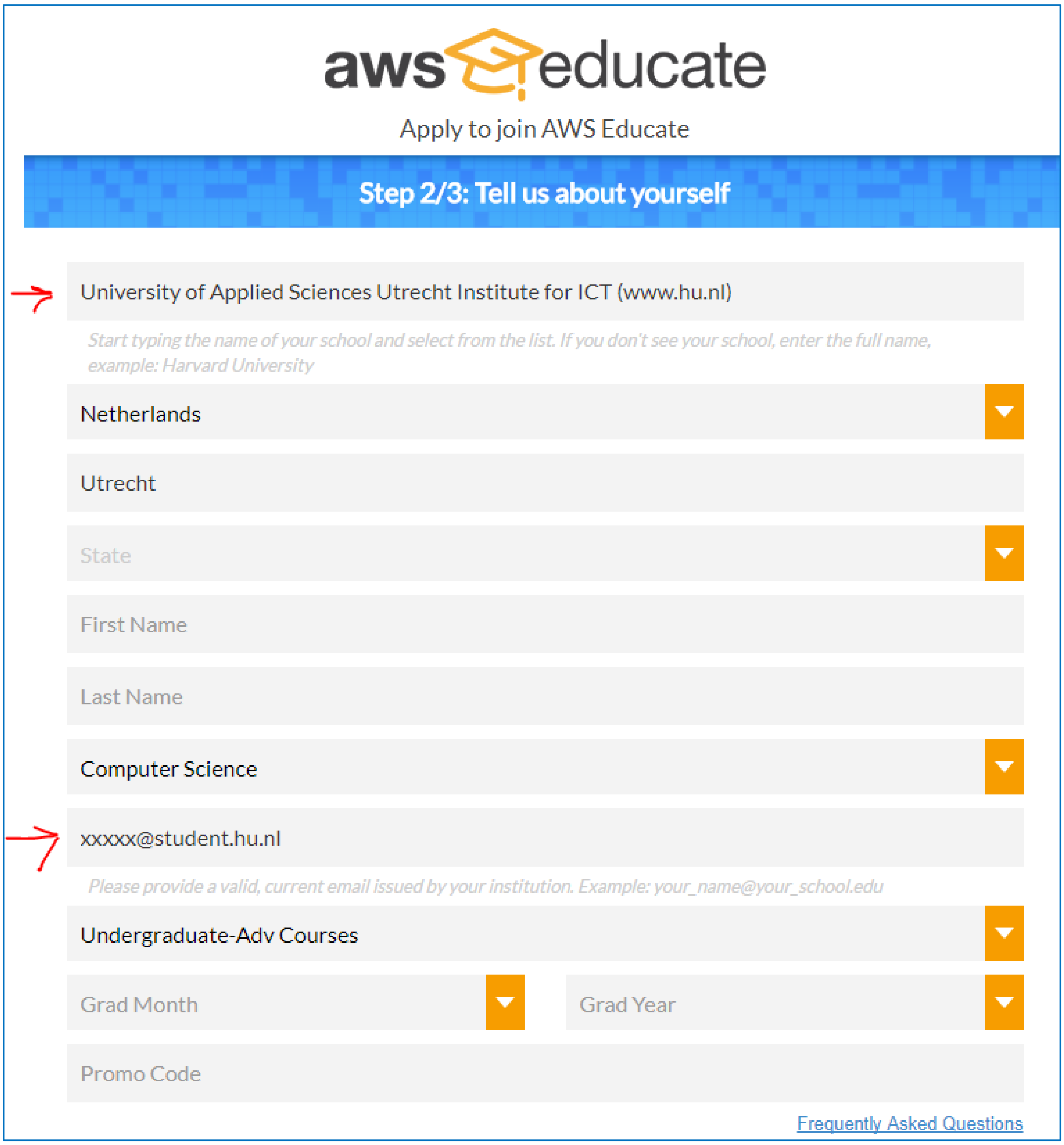
- Links / info related to extra information (not for the exam)

Attachment: Amazon AWS onboarding

2.1.59 Onboarding on AWS Educate (Starter Account: free yearly credits)

Visit https://www.awseducate.com/registration




aws educate

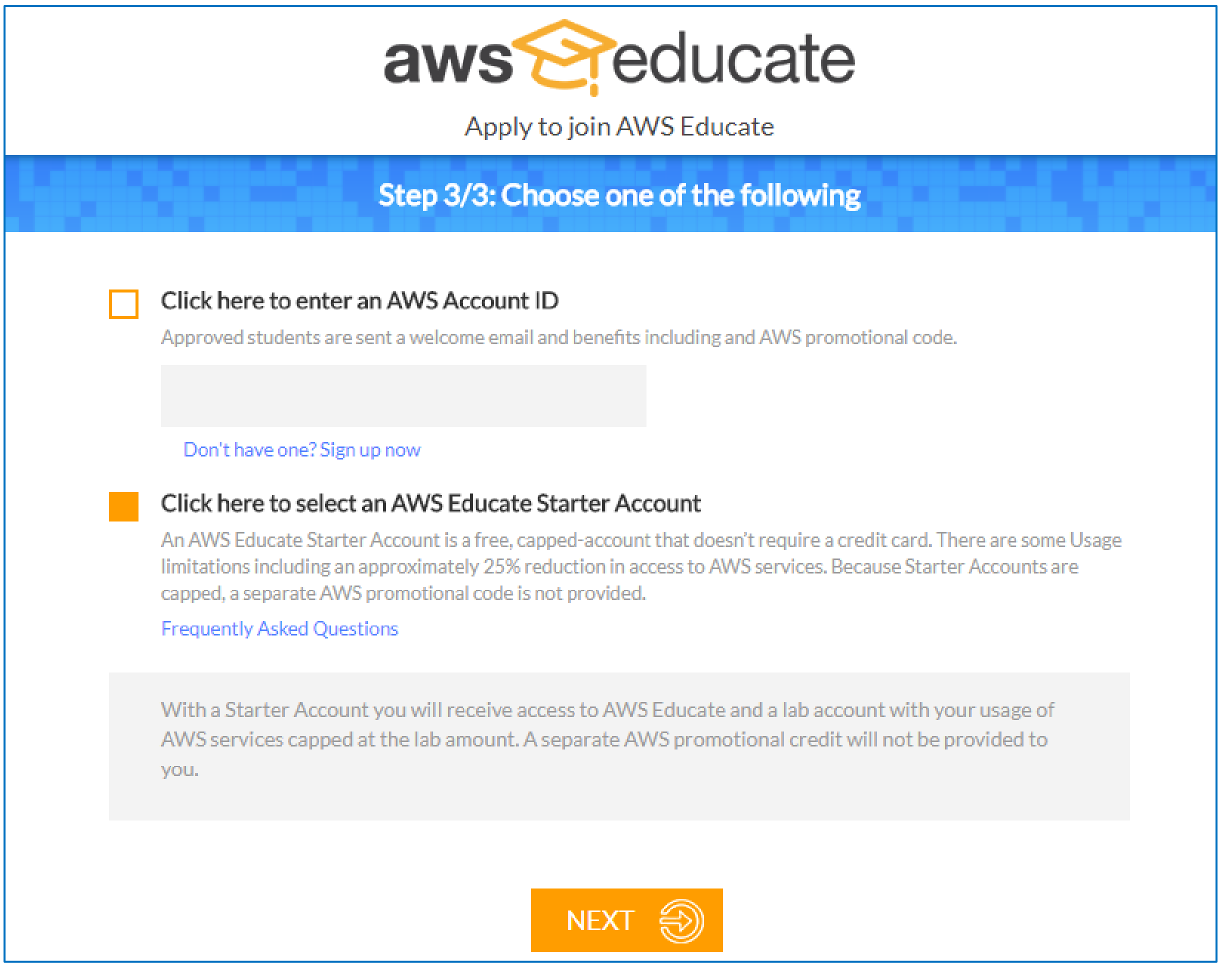
Apply to join AWS Educate

Step 2/3: Tell us about yourself

University of Applied Sciences Utrecht Institute for ICT (www.hu.nl)
Start typing the name of your school and select from the list. If you don't see your school, enter the full name, example: Harvard University
Netherlands
Utrecht
State
First Name
Last Name
Computer Science
xxxxx@student.hu.nl
Please provide a valid, current email issued by your institution. Example: your_name@your_school.edu
Undergraduate-Adv Courses
Grad Month / Grad Year
Promo Code

Frequently Asked Questions

Be careful and select our university and your HU-e-mail address (otherwise you won't get 75$ credits)!
In addition: select Grad Year as "current year + 3" (important not to select current year or next year).


Apply to join AWS Educate

Step 3/3: Choose one of the following

☐ Click here to enter an AWS Account ID

Approved students are sent a welcome email and benefits including and AWS promotional code.

Don't have one? Sign up now

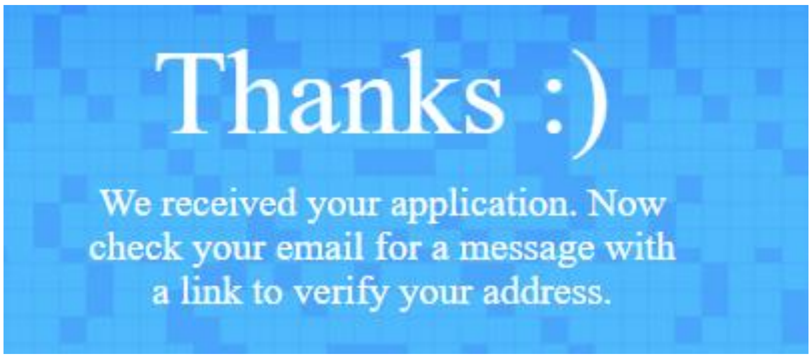
■ Click here to select an AWS Educate Starter Account

An AWS Educate Starter Account is a free, capped-account that doesn't require a credit card. There are some Usage limitations including an approximately 25% reduction in access to AWS services. Because Starter Accounts are capped, a separate AWS promotional code is not provided.

Frequently Asked Questions

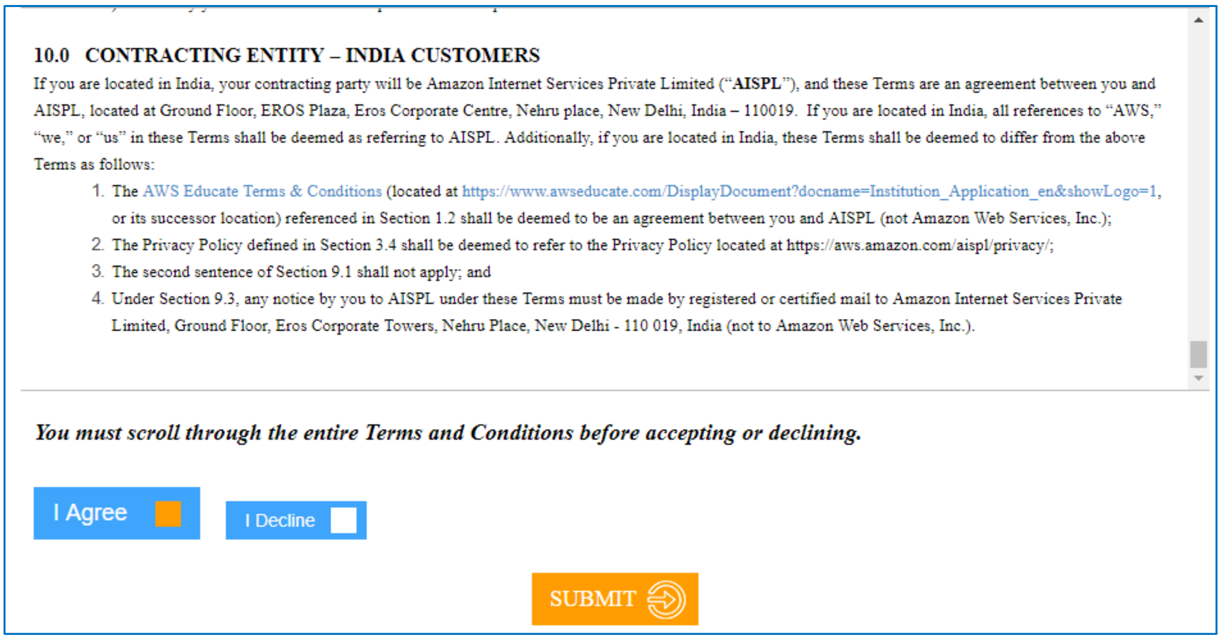
With a Starter Account you will receive access to AWS Educate and a lab account with your usage of AWS services capped at the lab amount. A separate AWS promotional credit will not be provided to you.

Select starter Account! (otherwise a credit card is needed)


Thanks :)

We received your application. Now check your email for a message with a link to verify your address.

Click link in your e-mail



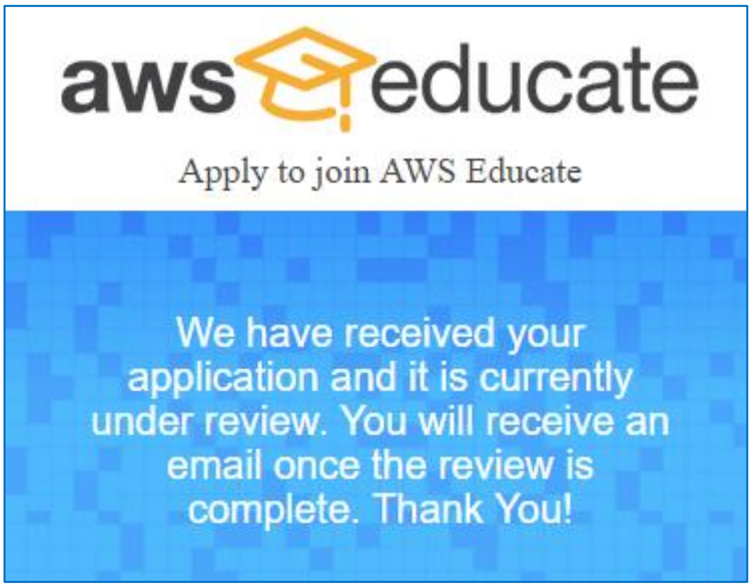
10.0 CONTRACTING ENTITY – INDIA CUSTOMERS

If you are located in India, your contracting party will be Amazon Internet Services Private Limited ("**AISPL**"), and these Terms are an agreement between you and AISPL, located at Ground Floor, EROS Plaza, Eros Corporate Centre, Nehru place, New Delhi, India – 110019. If you are located in India, all references to "AWS," "we," or "us" in these Terms shall be deemed as referring to AISPL. Additionally, if you are located in India, these Terms shall be deemed to differ from the above Terms as follows:

1. The AWS Educate Terms & Conditions (located at https://www.awseducate.com/DisplayDocument?docname=Institution_Application_en&showLogo=1, or its successor location) referenced in Section 1.2 shall be deemed to be an agreement between you and AISPL (not Amazon Web Services, Inc.);
2. The Privacy Policy defined in Section 3.4 shall be deemed to refer to the Privacy Policy located at https://aws.amazon.com/aispl/privacy/;
3. The second sentence of Section 9.1 shall not apply; and
4. Under Section 9.3, any notice by you to AISPL under these Terms must be made by registered or certified mail to Amazon Internet Services Private Limited, Ground Floor, Eros Corporate Towers, Nehru Place, New Delhi - 110 019, India (not to Amazon Web Services, Inc.).

You must scroll through the entire Terms and Conditions before accepting or declining.

Accept the terms



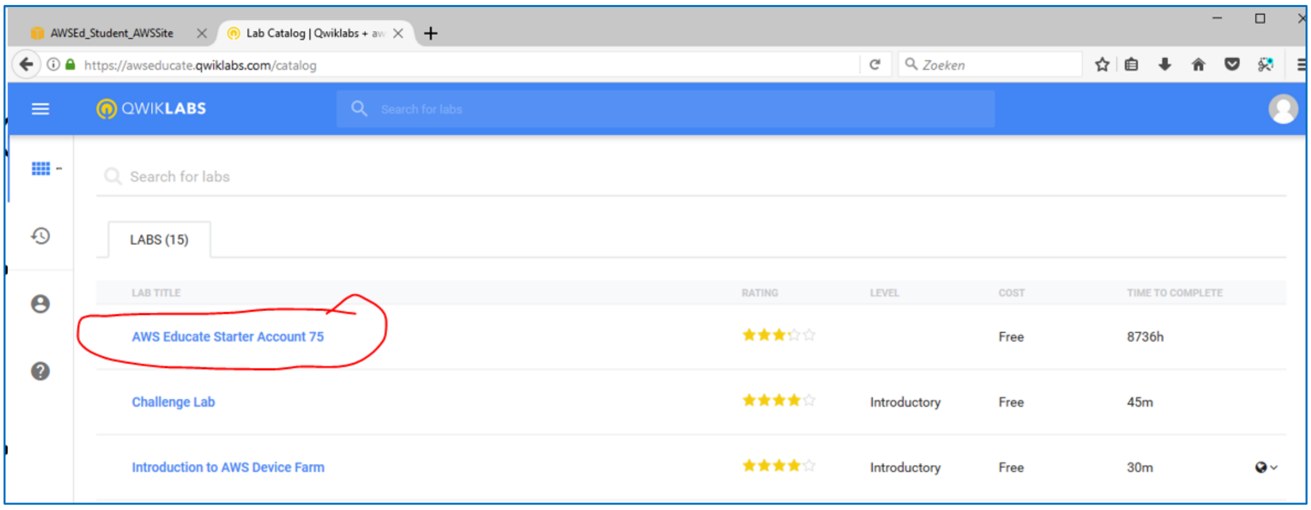
Apply to join AWS Educate

We have received your application and it is currently under review. You will receive an email once the review is complete. Thank You!

Review can take a few business days.

If you are accepted: Read any applicable qwikLABs terms and conditions governing the Starter Account, determine whether you are eligible and decide whether to accept the terms, and choose Submit to accept the terms. (If you are ineligible for the account or do not want to accept the terms, do not choose Submit and do not move forward with account creation.)

You will receive welcome email to set a password and enable access to your Starter Account.

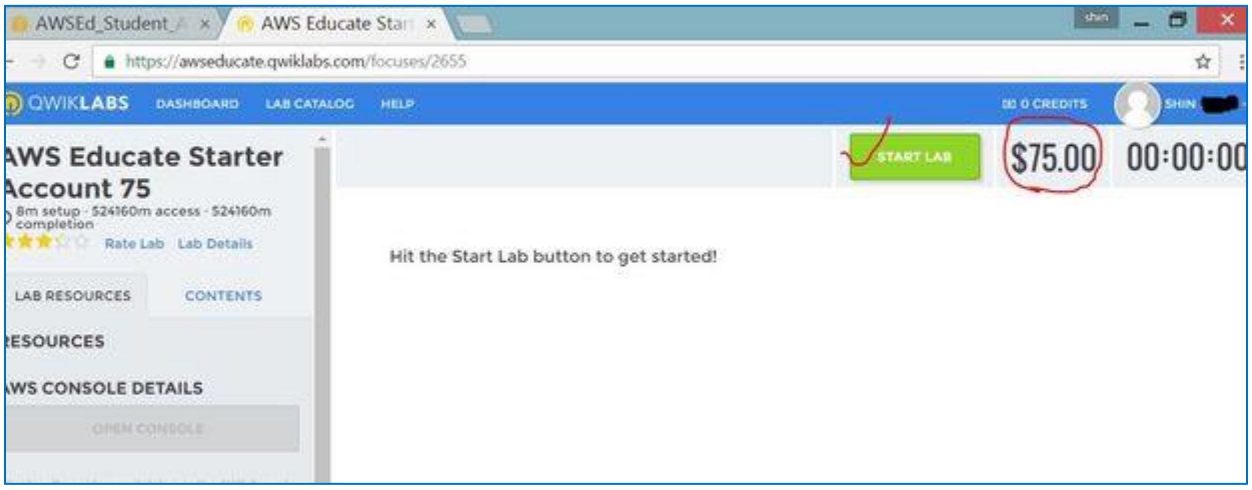


https://awseducate.qwiklabs.com/catalog

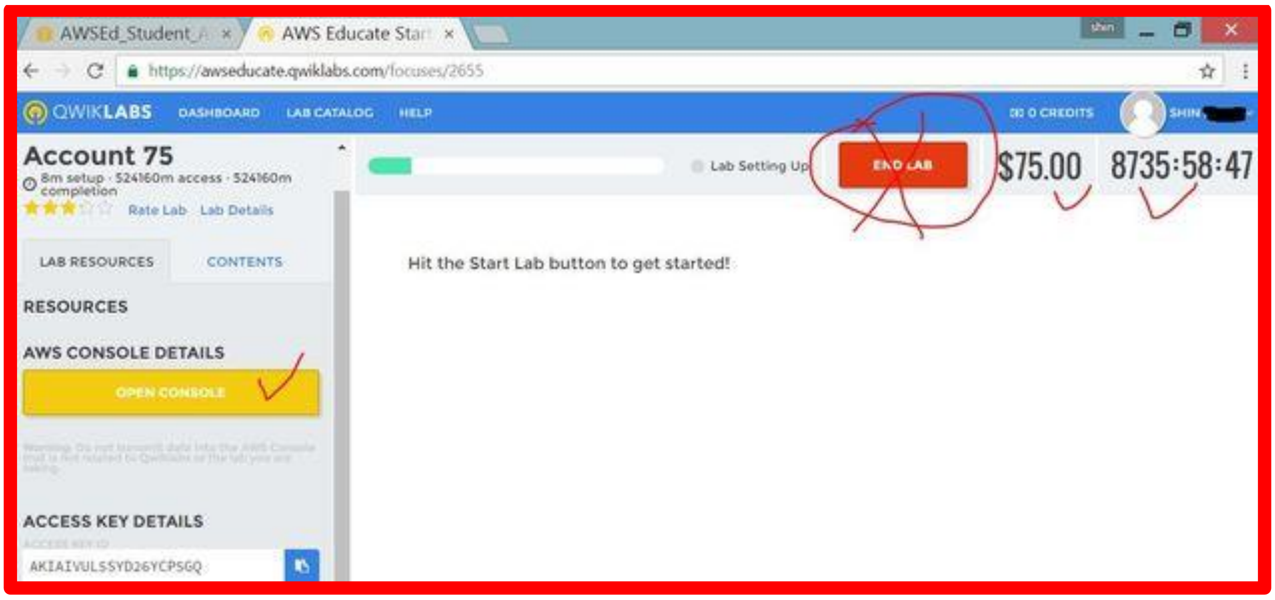
LABS (15)

LAB TITLE	RATING	LEVEL	COST	TIME TO COMPLETE
AWS Educate Starter Account 75	★★★☆☆		Free	8736h
Challenge Lab	★★★★☆	Introductory	Free	45m
Introduction to AWS Device Farm	★★★★☆	Introductory	Free	30m

One of the available labs is the “AWS Educate Starter Account 75”, this is free amazon!



Choose Start Lab, and then choose Go to Console.



When you finish working with your Starter Account, close your browser tab.

Important: If you choose End Lab, you will lose access to your Starter Account. Do not choose End Lab unless you no longer want to use your Starter Account.
Always stop your Virtual Machines and other resources used after usage. 75 dollars is not much!

Account credit is automatically added to your Starter Account once every 12 months from the date your application was approved until you graduate or are no longer eligible for AWS Educate.