02. [50p] The compute engine

In this exercise, we will instantiate a virtual machine using the **gcloud compute** engine. This may not be as straightforward as you expect. The reason for this is that there are many aspects to consider. For example, in what datacenter do we want our instance to reside. Do we want a public IP address assigned to it? Looking at the gcloud compute instances create [https://cloud.google.com/sdk/gcloud/reference/compute/instances/create] command, it may first appear intimidating. Let's take it step-by-step and discover what we need to create a VM. With each step, make sure to write down the parameters that you'll need later.

[5p] Task A - Enabling the compute service

Google Cloud offers a number of services, none of which are enabled by default. One of them is the **compute** service (i.e.: `compute.googleapis.com`) that lets us create VM instances. When running a command that requires a certain service that was not previously enabled, you get a prompt asking you if you want to enable it then and there. In this case, we'll do it manually. Note that this may take a bit of time, up to a couple of minutes.

```
# get full list of available services
$ gcloud services list --available

# enable the compute service
$ gcloud services enable compute.googleapis.com
```

[5p] Task B - Configuring a SSH public key

Once your VM is up and running (at *Task G*), you will want to be able to SSH into it. For this to happen, you will have to configure a public SSH key. This key will be automatically copied into *~/.ssh/authorized_keys* at creation. If you *still* don't have a SSH keypair generated, now's a good a time as any (see ssh-keygen [https://www.ssh.com/academy/ssh/keygen]).

This is one annoying aspect of Google Cloud that I wish they'd change. You can't choose your default username for the VMs you create. In stead, they derive a username from your account's gmail address. After you run the next command, look for the **username** variable in the output and write it down.

```
# upload your public SSH key to gcloud
# --ttl 0 means that the key does not have an expiration date
$ gcloud compute os-login ssh-keys add --ttl 0 --key-file ~/.ssh/id_rsa.pub
```

[5p] Task C - Selecting a base image

First things first. What OS do we want to run on our machine? A Windows server [https://www.microsoft.com/en-us/windows-server]? Maybe CentOS [https://www.centos.org/]? Nay – let's look for something familiar, namely Ubuntu. Make sure to take note of the *PROJECT* and *FAMILY* columns:

```
# list available base VM images
$ gcloud compute images list
```

[5p] Task D - Selecting a region

All cloud providers worth their salt will offer you a number of physical locations (datacenters) where to deploy your instance. Locality is very important when offering web services. Normally, this is a difficult task. Can you imagine YouTube running on a single server somewhere in the US and you accessing it from SEA? Many people use Content Delivery Networks (CDN) [https://www.cloudflare.com/learning/cdn/what-is-a-cdn/] for this task. Even DigitalOcean, a rather important cloud provider uses CloudFlare as a proxy for their HTTP servers.

CDNs offer many advantages. The reason why DitigalOcean is using CloudFlare despite having the resources themselves is Distributed Denial of Service (DDoS) protection [https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/]. This protection however, comes at a cost that is not necessarily monetary in nature. CDNs usually have access to the private communication between you and said HTTP server, even if you are using HTTPS [https://www.cloudflare.com/learning/ssl/why-is-http-not-secure/]. Why? Because they need to perform deep packet introspection in order to classify malicious traffic as such.

You can read up on Google's regions and zones [https://cloud.google.com/compute/docs/regions-zones]. When working with your own funds and not with free tier accounts [https://cloud.google.com/free/docs/gcp-free-tier] or education credits, you will want to consult their regional pricing model [https://cloud.google.com/compute/all-pricing]. Usually, US-based datacenters are much cheaper.

```
# show available regions
$ gcloud compute regions list

# show zones in selected region
$ gcloud compute zones list --filter ${REGION}
```

[5p] Task E - Selecting a machine type

When selecting the number of Virtual CPUs (vCPU) and RAM for your VM, you will have to choose from a list of presets. These presets may vary depending on the region.

It is not unusual for cloud providers to limit the number of vCPUs that you may reserve, especially for personal accounts (i.e.: not organizations). AWS for example automatically imposed a 128 vCPU limit (across all VMs registered under an account) some time ago, for default users. This parameter was automatically set at account creation. So people who created AWS accounts a while back may have a 1024 vCPUs limit instead. The reason for this is to limit the losses that they may incur from a bad actor that registers a credit card with say, $5 and no intention to actually pay for their resource usage when charged. In AWS's case, this limit can be increased by contacting support. Since RAM is an inexpensive resource in comparison, it's not usually a limiting factor.

```
# show available flavors for your selected zone
$ gcloud compute machine-types list --zones "${YOUR_ZONE}"
```

[5p] Task F - Selecting a storage option

Storage options refer to HDDs/SSDs and can be separated from the instance you create. Meaning that you can delete the instance and still keep your storage image intact, in case you may want to mount it to another instance in the future (think moving a USB stick between PCs). For this exercise, we won't dive into this specific use-case and delete the virtual storage device together with the instance (at the end of the lab). A 10G disk should be *more* than sufficient. Again, the storage device selection is specific to each zone.

```
# select storage type for selected zone
$ gcloud compute disk-types list --zones "${YOUR_ZONE}"
```

[5p] Task G - Creating the instance

Finally, it's time to start up our VM. In addition to all the parameters that you've selected until now, you will also have to give the instance a name.

```
# create the instance
$ gcloud compute instances create ${INSTANCE_NAME} \
        --image-family   ${IMAGE_FAMILY}           \
        --image-project  ${IMAGE_PROJECT}          \
        --zone           ${ZONE}                   \
        --machine-type   ${FLAVOR}                 \
        --boot-disk-type ${DISK_TYPE}              \
        --boot-disk-size ${DISK_SIZE}              \
        --metadata enable-oslogin=TRUE

# show active instances
$ gcloud compute instances list
```

[5p] Task H - Connecting to your instance

Using the username that was generated for you (see Task B) and the Public (External) IP address of your VM, connect to it via SSH. The Ubuntu image comes with a pre-configured SSH server. But remember that after you created the instance, it still requires ~1m to boot and start up the services.

```
# connect to your instance
$ ssh ${USERNAME}@${PUBLIC_IP}
```

In case you lost your private SSH key in the meantime (how?), you can also connect using **gcloud compute ssh**, which will generate a Google Cloud-specific SSH key for you. The normal method above is preferable to this.

[10p] Task I - Configuring the firewall

Google Cloud implements a software firewall with a few default rules that allow, for example, incoming SSH connections. If you want to run a server on your VM and contact it from a public network, you will need to add a new rule, whitelisting it.

Without getting into too much detail regarding networking stuff, IP addresses represent a network interface on your machine. In contrast to an IP address, a port does not have a hardware correspondent. It's a 16-bit number that represents an endpoint on your machine. This endpoint can be used by *a single process at a time* to either listen for incoming connections or establish outgoing connections. Some port numbers have consecrated meanings (e.g.: 22 - SSH, 80 - HTTP, 443 - HTTPS) and are reserved for servers. Why servers and not clients? Because the clients initiate connections and need to know where to find the servers. The servers don't care what port the clients are running on; anything will do. Port numbers are split into three categories:

- **0 - 1023** : system / well-known ports – don't fuck with these!
- **1024 - 49151** : reserved ports – usually used for servers of any kind
- **49152 - 65535** : ephemeral ports – used for dynamic connections

Next, we want to start a **netcat** server on the gcloud instance and send a text message from our localhost. The **netcat** server will run on port 8989, so we want to whitelist it. The **tcp** part in the commands below refers to the TCP protocol [https://en.wikipedia.org/wiki/Transmission_Control_Protocol]. You don't need to understand what protocols are. These will be covered in the 3rd year Local Networks [https://ocw.cs.pub.ro/courses/rl] course. For the sake of this task, mentioning it is unavoidable. If you want to know more, ask your assistant.

In the following commands *[localhost]* means that the command should be executed on your computer, and *[gcloud]* means that the command should be executed on the VM, over SSH.

```
# display current firewall rules
[localhost]$ gcloud compute firewall-rules list

# add a new firewall rule
[localhost]$ gcloud compute firewall-rules create ${SOME_RULE_NAME} --allow tcp:8989

# start a netcat server in listening mode (-l)
[gcloud]$ nc -l 8989

# send some data to the server
[localhost]$ echo "Hello world!" | nc ${PUBLIC_IP} 8989
```

These firewall rules are not specific to your VM, but to your account! They apply to all your VM instances.

Linux has a built-in firewall system called **iptables**. If you want to create instance-specific firewall rules, you can use this. We won't be covering **iptables** in this lab, but it's nice to know that it's there.