Basic Introduction to gcloud

Nick Thompson

January 28, 2016

Goals of this discussion

Goals of this discussion

- Learn how to rent CPUs and RAM from google

Goals of this discussion

- Learn how to rent CPUs and RAM from google
- Learn how to rent persistent disks

Goals of this discussion

- Learn how to rent CPUs and RAM from google
- Learn how to rent persistent disks
- Learn how to control expenses

Goals of this discussion

- Learn how to rent CPUs and RAM from google
- Learn how to rent persistent disks
- Learn how to control expenses
- Learn how to manage permissions

Goals of this discussion

- Learn how to rent CPUs and RAM from google
- Learn how to rent persistent disks
- Learn how to control expenses
- Learn how to manage permissions
- Learn how to create teams of identical VMs

Goals of this discussion

- Learn how to rent CPUs and RAM from google
- Learn how to rent persistent disks
- Learn how to control expenses
- Learn how to manage permissions
- Learn how to create teams of identical VMs
- Learn how to autoscale nodes based on load

What we won't discuss, but should

What we won't discuss, but should

- The Python/Go/Node bindings

What we won't discuss, but should

- The Python/Go/Node bindings
- The REST (representational state transfer) interface

What we won't discuss, but should

- The Python/Go/Node bindings
- The REST (representational state transfer) interface
- gcloud sql

What we won't discuss, but should

- The Python/Go/Node bindings
- The REST (representational state transfer) interface
- gcloud sql
- Google Container Engine

What we won't discuss, but should

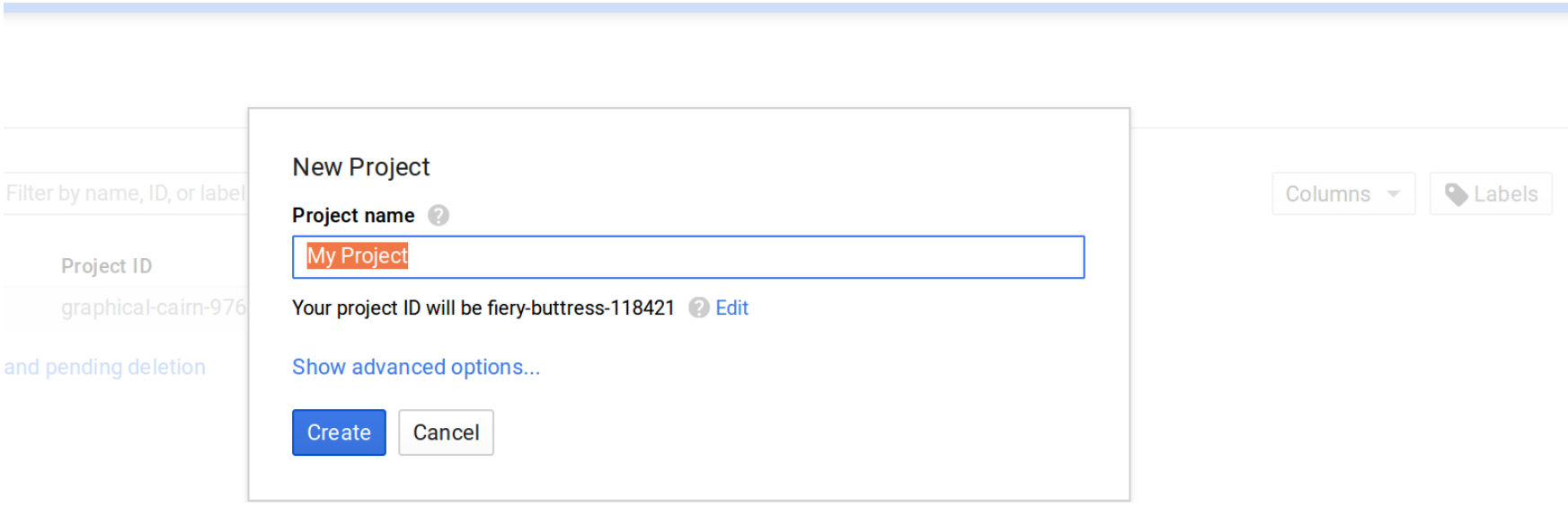
- The Python/Go/Node bindings
- The REST (representational state transfer) interface
- gcloud sql
- Google Container Engine
- Lots more . . .

Getting started:

```
$ curl https://sdk.cloud.google.com | bash
$ exec -l $SHELL
$ gcloud init
$ gcloud auth login
$ firefox https://console.developers.google.com
```

You don't need a gmail account to use gcloud, but you must enter an email into gcloud that then becomes your google account login.

Starting a Project



Starting a Project

Each project has

Starting a Project

Each project has

- its bill separated in the invoice (1 billing account $\mapsto$ many project accounts)

Starting a Project

Each project has

- its bill separated in the invoice (1 billing account $\mapsto$ many project accounts)
- its own VMs

Starting a Project

Each project has

- its bill separated in the invoice (1 billing account $\mapsto$ many project accounts)
- its own VMs
- its own persistent disks

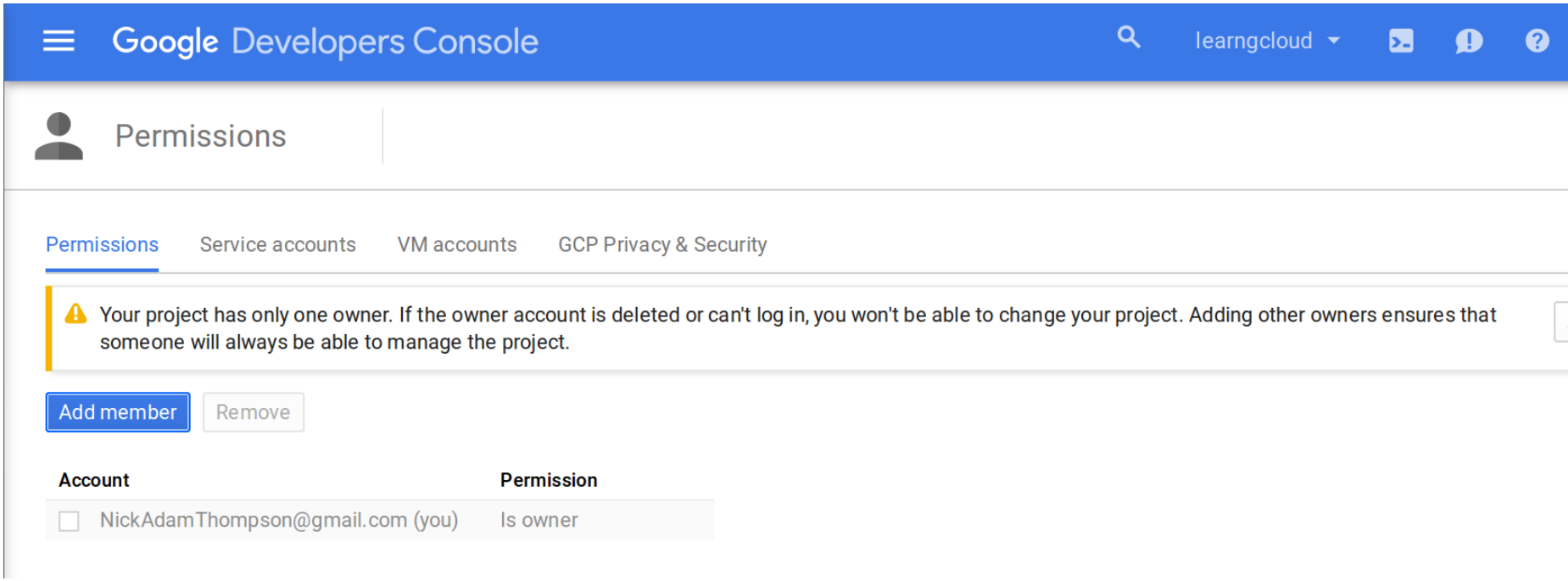
Starting a Project

Each project has

- its bill separated in the invoice (1 billing account $\mapsto$ many project accounts)
- its own VMs
- its own persistent disks
- its own users with their associated permissions.

Starting a Project

Multiple people can be listed as project owners, or given read access, or read/write access to the project:



Note that people who only have read access to the project nonetheless have root access to all the VMs!

Permissions

The permission levels for a project are

Permissions

The permission levels for a project are

- Owners-who can add/remove team members, and rent resources, and are root on resources

Permissions

The permission levels for a project are

- Owners-who can add/remove team members, and rent resources, and are root on resources
- Editors-who can rent resources, and are root on resources

Permissions

The permission levels for a project are

- Owners-who can add/remove team members, and rent resources, and are root on resources
- Editors-who can rent resources, and are root on resources
- Viewers-who can't rent resources, but are root on resources

Starting a Project

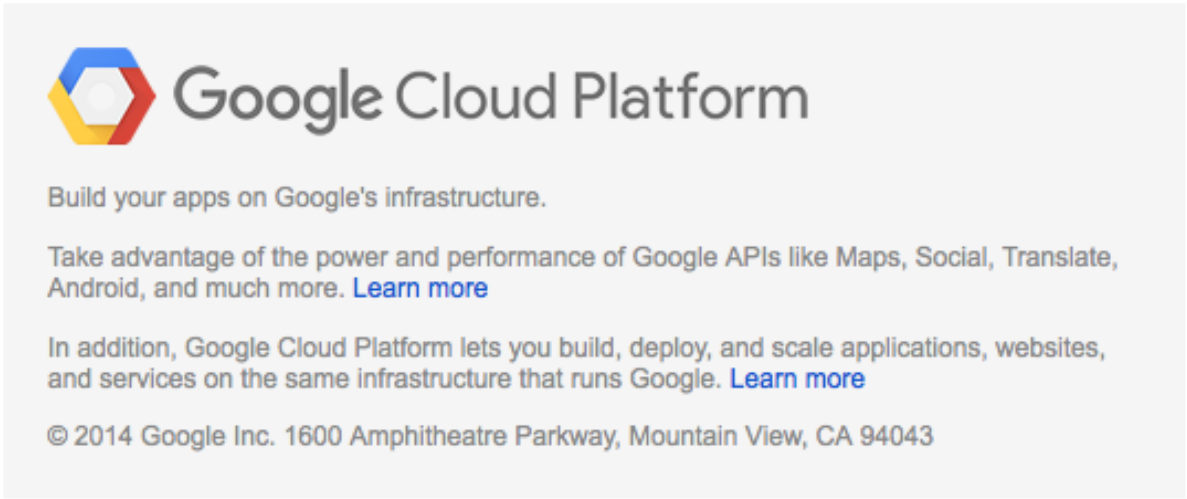
When you grant someone permission to view/edit/co-own your project, they receive an email asking them if they want to join:

Hello, 张洁,

I invite you to join the Google Developers Console project "myfirstproject". Please click this link to accept my invitation:

https://console.developers.google.com/project/learngcloud-1184/rsvp/?account=zjzjhn@gmail.com

Thanks,
Nick Thompson
NickAdamThompson@gmail.com



gcloud user accounts

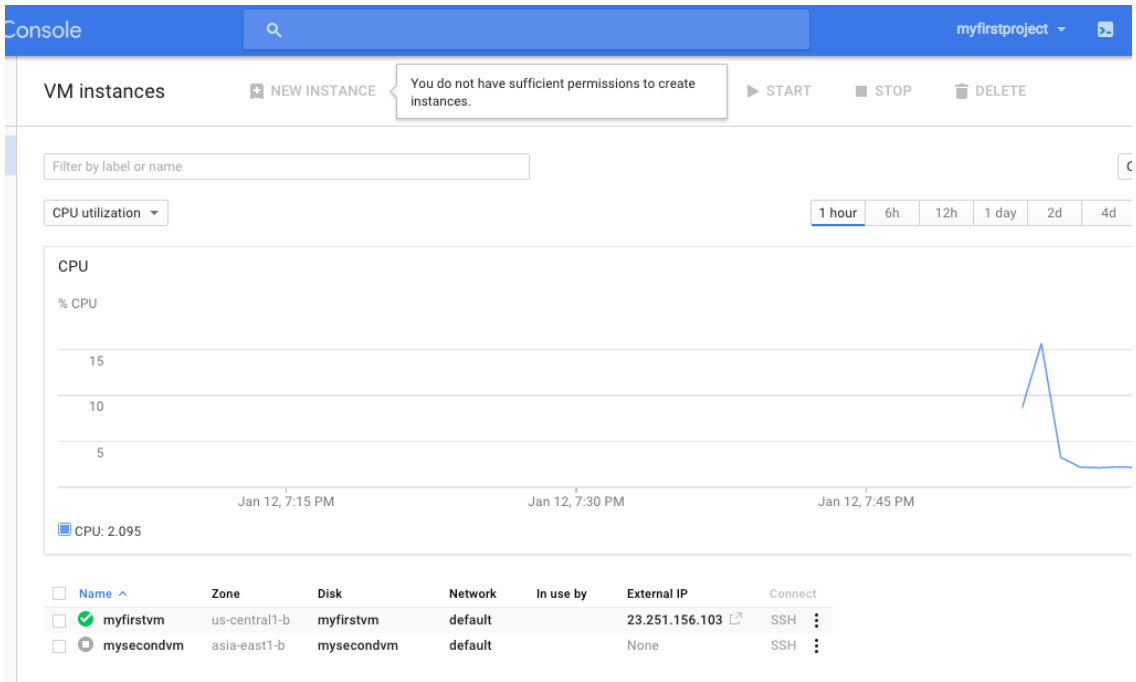
gcloud user accounts are in beta, and seem to be evolving rapidly. To see more, use

```
$ gcloud beta compute users -h
```

or visit the docs.

gcloud user accounts

Users with read-only access to your project have root on your VMs, but they can't launch new VMs on your dime:



Starting a project

You need tell the gcloud command-line tool that you've created a new project:

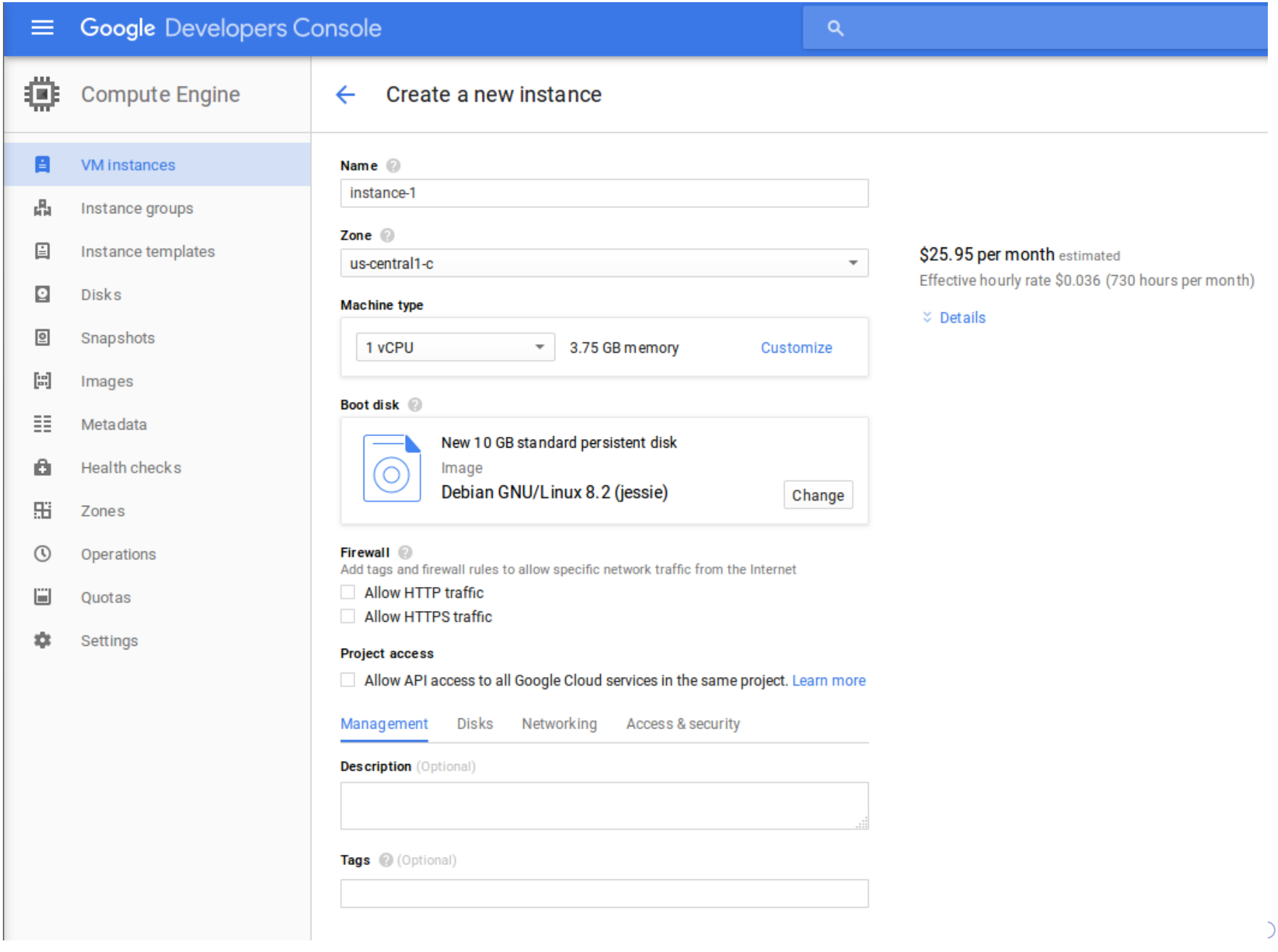
```
$ gcloud config list
[core]
account = nickadamthompson@gmail.com
disable_usage_reporting = True
project = graphical-cairn-97618
[meta]
active_config = default
```

If the `project` field has the wrong value, you need to set it:

```
$ gcloud config set project learngcloud-1184
```

Note that you don't set the project *name*, you set the project *ID*.

Setting up an instance



Setting up an instance

Things to choose at this point:

Setting up an instance

Things to choose at this point:

- Number of cores (min 1/2, max 32)

Setting up an instance

Things to choose at this point:

- Number of cores (min 1/2, max 32)
- amount of RAM (min 0.6GB, max unlimited, it seems, but 200GB is supported for sure)

Setting up an instance

Things to choose at this point:

- Number of cores (min 1/2, max 32)
- amount of RAM (min 0.6GB, max unlimited, it seems, but 200GB is supported for sure)
- size of disk, SSD (max 10TB) or spinning (max 10TB)

Setting up an instance

Things to choose at this point:

- Number of cores (min 1/2, max 32)
- amount of RAM (min 0.6GB, max unlimited, it seems, but 200GB is supported for sure)
- size of disk, SSD (max 10TB) or spinning (max 10TB)
- Operating system (Ubuntu, Centos, CoreOS), or choose a VM snapshot

Setting up an instance

Things to choose at this point:

- Number of cores (min 1/2, max 32)
- amount of RAM (min 0.6GB, max unlimited, it seems, but 200GB is supported for sure)
- size of disk, SSD (max 10TB) or spinning (max 10TB)
- Operating system (Ubuntu, Centos, CoreOS), or choose a VM snapshot
- Firewall rules

Setting up an instance

Things to choose at this point:

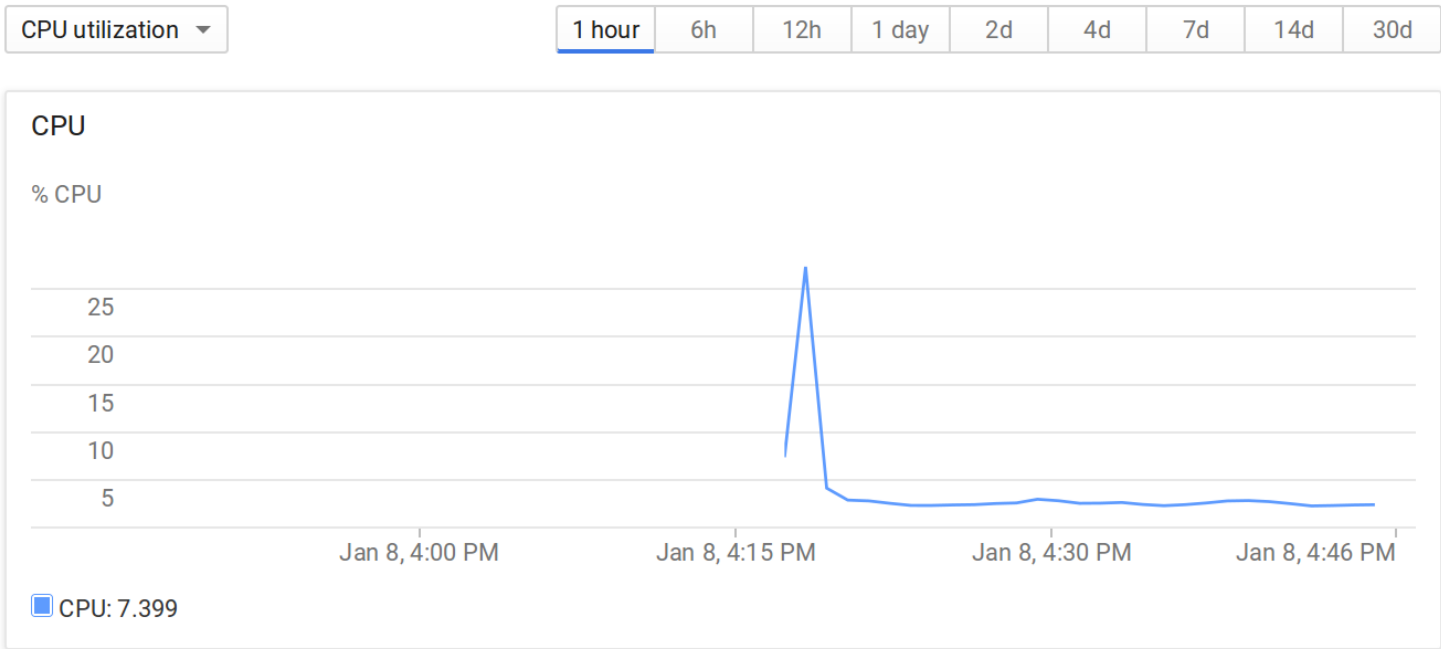
- Number of cores (min 1/2, max 32)
- amount of RAM (min 0.6GB, max unlimited, it seems, but 200GB is supported for sure)
- size of disk, SSD (max 10TB) or spinning (max 10TB)
- Operating system (Ubuntu, Centos, CoreOS), or choose a VM snapshot
- Firewall rules
- Whether to use static or ephemeral IP addresses (static IPs cost money!)

Setting up an instance

Aside: If you need more than 10TB of disk space, you need to fill out the Google Compute Engine Quota Change Request Form.

Setting up an instance

Once you create a VM, you'll be assigned an external IP address and can see the load on your server:



Name	Zone	Disk	Network	In use by	External IP	Connect
myfirstvm	us-central1-b	myfirstvm	default		104.154.88.253	SSH

Setting up an instance

To access your VM, use:

```
$ gcloud compute ssh myfirstvm --zone us-central1-b
```

(Don't choose the wrong zone or else your instance won't be found!)
This is really a wrapper script around the IP address of your instance:

```
$ ssh -i ~/.ssh/google_compute_engine 104.154.88.253
```

Setting up an instance

For every console action, there is a equivalent gcloud command. So, for example, to set up an instance, you could type

```
$ gcloud compute instances create mysecondvm \
      --image ubuntu-15-10 --zone us-central1-b
```

This is useful for scripting.

Exercise

Why doesn't the following code work?

```
local$ gcloud compute instances create vm1 --zone us-central1-b
local$ gcloud compute ssh vm1
vm1$ gcloud compute instances create vm2 --zone us-central1-b
ERROR: (gcloud.compute.instances.create) Failed to find image for alias
 - Insufficient Permission
```

Solution

You are logged into gcloud, and can rent resources. However, your VM isn't you, and it doesn't have permission to rent resources on your behalf.
You can solve this problem by authorizing your VM to be able to rent resources using a *service account*.

Service Accounts

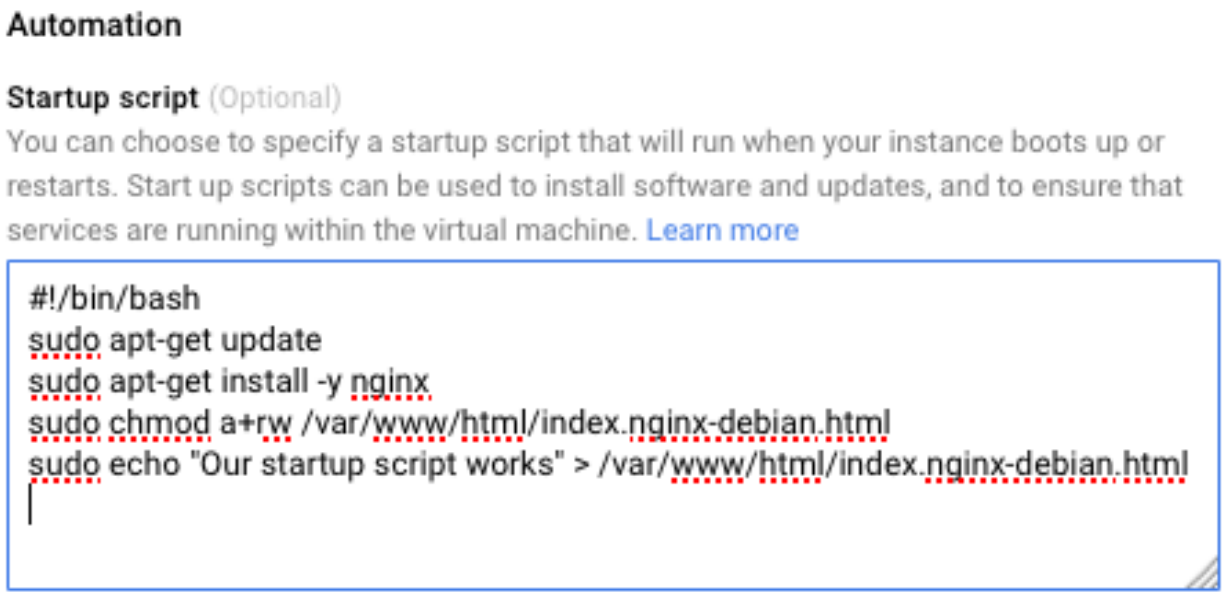
To authorize a VM to rent resources on your behalf, use the *scopes* tag:

```
$ gcloud compute instances create vm-scoped \
    --scopes compute-rw --zone us-central1-b
```

This gives your vm permission to spawn new vms.

Startup Scripts

To get an image configured for autoscaling, we need to specify what sort of software we need on the image before it's creation. We can do this via a "startup script":



Startup Scripts

It works!

```
$ curl 130.211.135.33
Our startup script works
```

However, this is not a good design; as the startup script is not in source control.

Startup Scripts

Once your startup script is in source control, you can easily deploy a new instance via:

```
$ gcloud compute instances create wstartup \
   --metadata-from-file startup-script=script.sh \
   --zone=us-central1-b  --tags "http-server"
Created [https://www.googleapis.com/compute/v1/projects/learngcloud-118
NAME     ZONE          MACHINE_TYPE  PREEMPTIBLE INTERNAL_IP EXTERNAL_I
wstartup us-central1-a n1-standard-1             10.240.0.5  104.197.59
$ curl 104.197.59.116
Our startup script works
```

Startup Scripts

Note that you can ssh into your machine before the startup script has finished!
To see if your startup script has finished, or to debug your startup script:

```
$ gcloud compute ssh wstartup --zone=us-central1-b
wstartup$ cat /var/log/startupscript.log
...
ubuntu startupscript: Finished running startup script /var/run/google.s
```

Regions and Zones

Google divides the world into regions:

```
$ gcloud compute regions list
NAME          CPUS          DISKS_GB     ADDRESSES RESERVED_ADDRESSES ST
asia-east1       0.00/24.00     10/10240      0/23      0/7            UP
europe-west1     0.00/24.00      0/10240      0/23      0/7            UP
us-central1      4.00/24.00     40/10240      4/23      0/7            UP
us-east1         0.00/24.00      0/10240      0/23      0/7            UP
```

Regions and Zones

And each region is divided into multiple zones

```
$ gcloud compute zones list
NAME            REGION        STATUS NEXT_MAINTENANCE TURNDOWN_DATE
asia-east1-a    asia-east1    UP
asia-east1-b    asia-east1    UP
asia-east1-c    asia-east1    UP
europe-west1-b  europe-west1  UP
europe-west1-d  europe-west1  UP
europe-west1-c  europe-west1  UP
us-central1-c   us-central1   UP
us-central1-a   us-central1   UP
us-central1-f   us-central1   UP
us-central1-b   us-central1   UP
us-east1-c      us-east1      UP
us-east1-b      us-east1      UP
us-east1-d      us-east1      UP
```

Regions and Zones

A zone is essentially single datacenter, so two instances in a zone can communicate very quickly with one another:

```
vmcentral1b-1$ ping vmcentral1b-2 # ping VM in same datacenter/zone:
rtt min/avg/max/mdev = 0.308/0.394/0.841/0.104 ms
vmcentral1b-1$ ping vmcentral1a # Ping VM in same region, different zon
rtt min/avg/max/mdev = 0.571/0.673/1.210/0.099 ms
vmcentral1b-1$ ping vm-in-asia
rtt min/avg/max/mdev = 154.738/154.878/155.326/0.442 ms
```

So within-zone communication is fastest, within-region is fast, between-region is slow. (Note how your instance names are resolved via DNS!)

Regions and Zones

Things you rent from Google are classified by their scope in the global/regional/zonal hierarchy. For instance,

Regions and Zones

Things you rent from Google are classified by their scope in the global/regional/zonal hierarchy. For instance,

- Images, VM snapshots, firewall rules, and buckets are global resources

Regions and Zones

Things you rent from Google are classified by their scope in the global/regional/zonal hierarchy. For instance,

- Images, VM snapshots, firewall rules, and buckets are global resources
- Addresses are regional resources

Regions and Zones

Things you rent from Google are classified by their scope in the global/regional/zonal hierarchy. For instance,

- Images, VM snapshots, firewall rules, and buckets are global resources
- Addresses are regional resources
- Instances and their boot disks are zonal resources

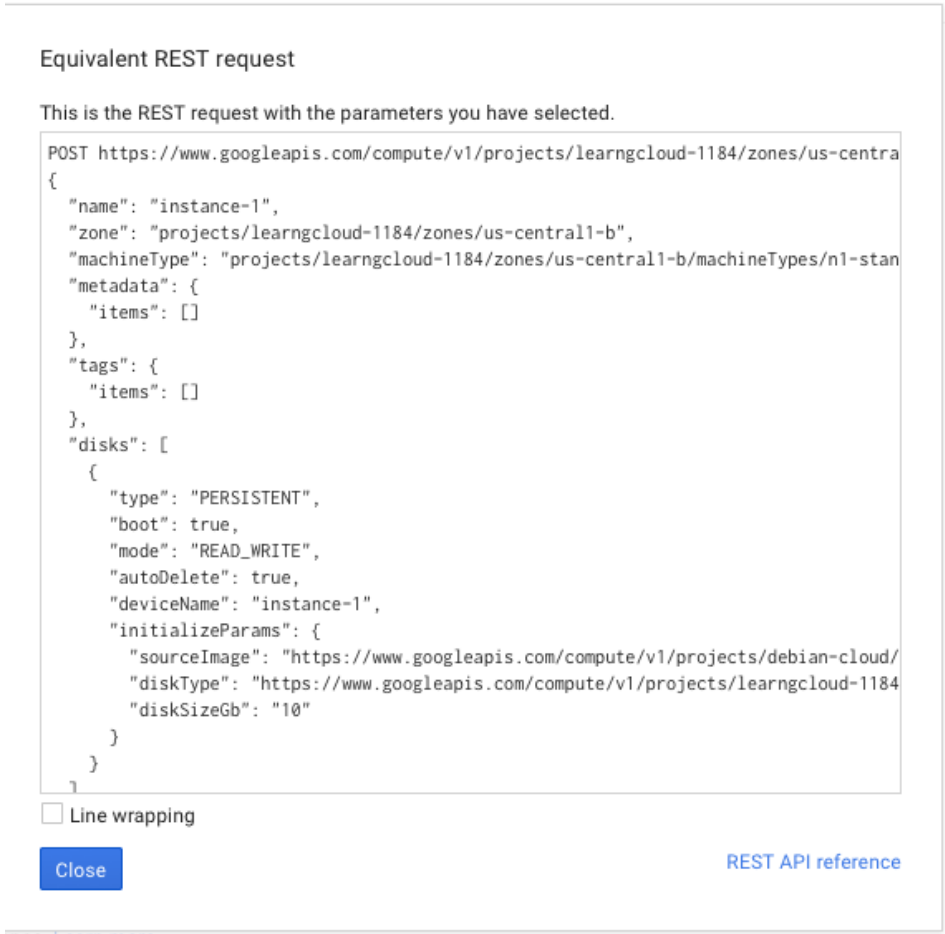
Exercise

Why does google need to ask us for the zone for almost every command?

```
$ gcloud compute instances describe vm2
For the following instances:
 - [vm2]
choose a zone:
 [1] asia-east1-a
 ...
```

Solution

Everything action in gcloud is achieved via REST (representational state transfer).
This is a POST/GET/DELETE/PUT request to an html endpoint, and the endpoint url contains the zone!



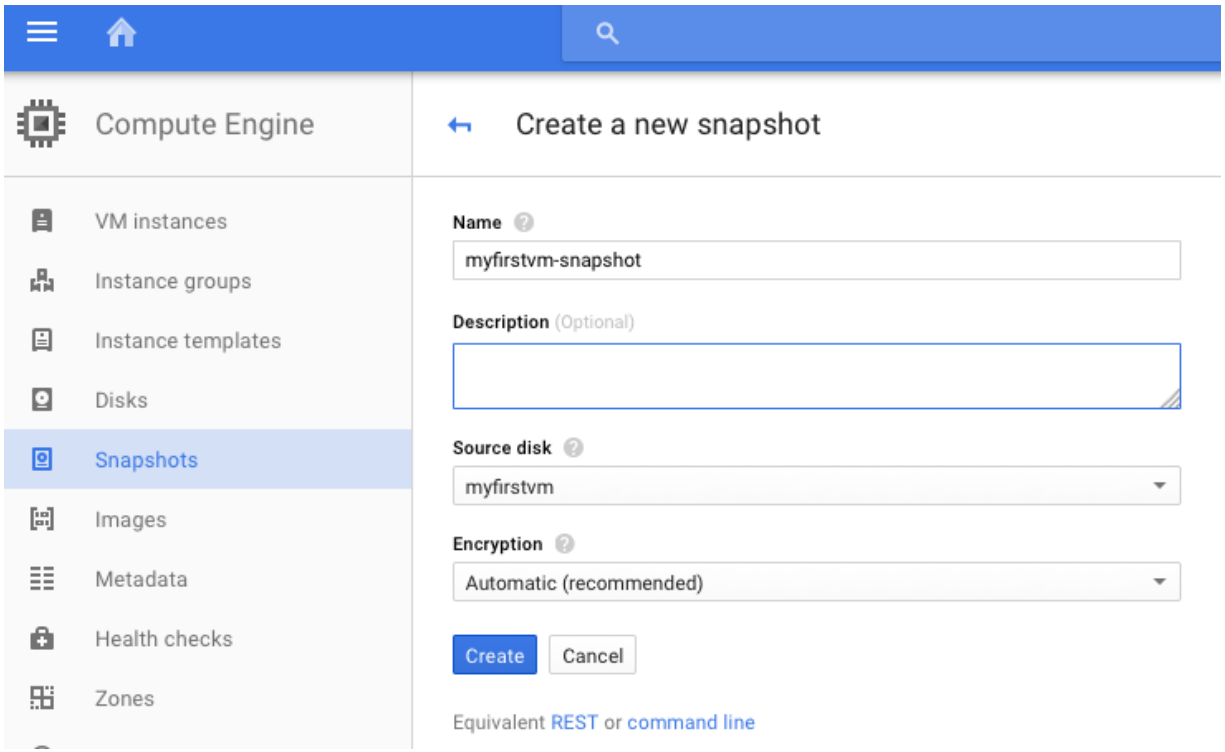
Snapshotting an instance

Once you have installed your favorite software on your VM, you might want to snapshot it:

```
myfirstvm$ sudo apt-get update
myfirstvm$ sudo apt-get install -y gcc g++ emacs git
myfirstvm$ sudo touch /opt/hello.txt && sudo chmod a+rw /opt/hello.txt
myfirstvm$ echo "Hello from GCE!" >> /opt/hello.txt
myfirstvm$ exit
localhost$ gcloud compute disks snapshot "myfirstvm" \
               --zone "us-central1-b" --snapshot-names "firstvmsnapshot"
```

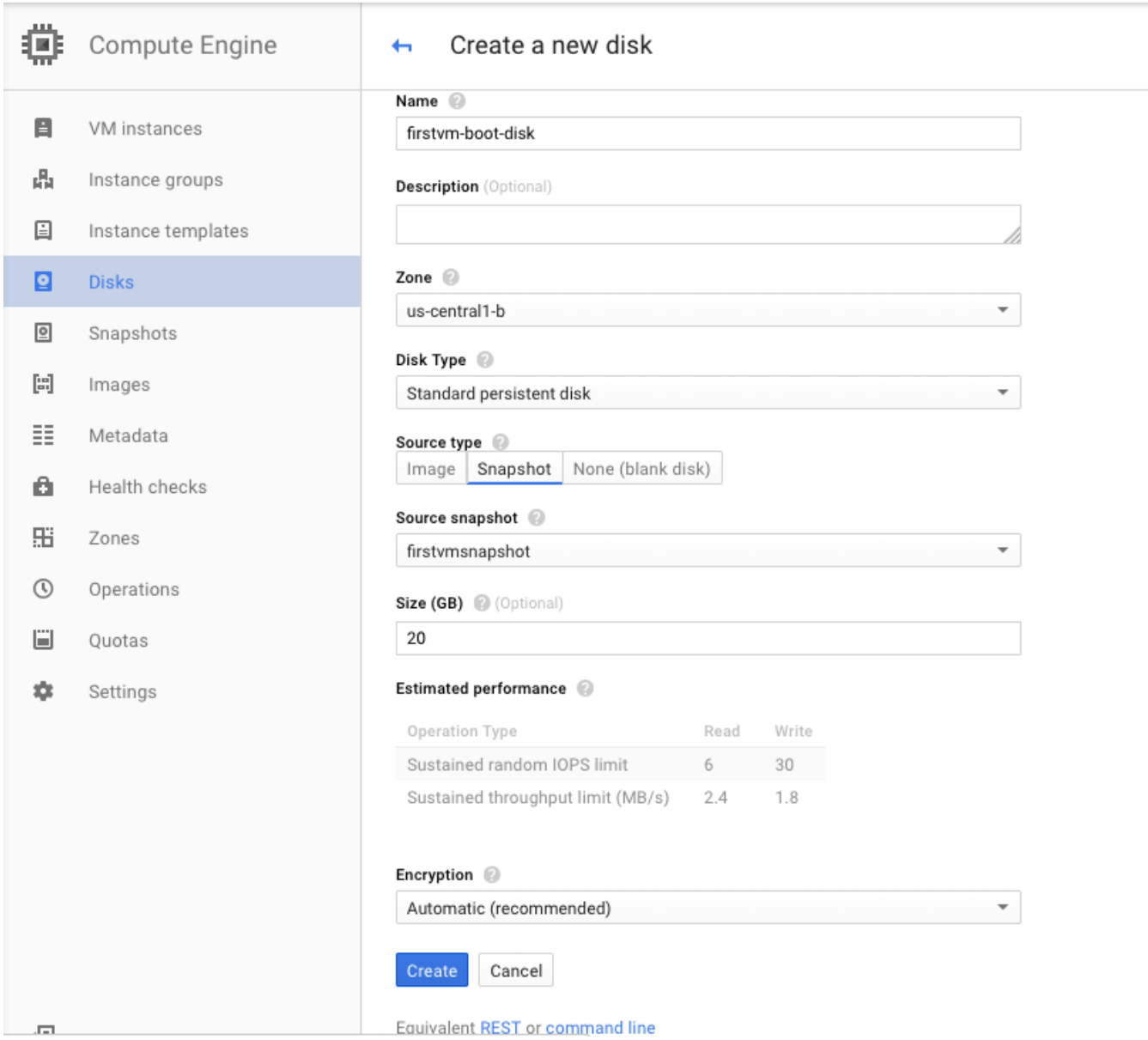
Snapshotting an instance

Of course, using the console GUI is a bit easier:



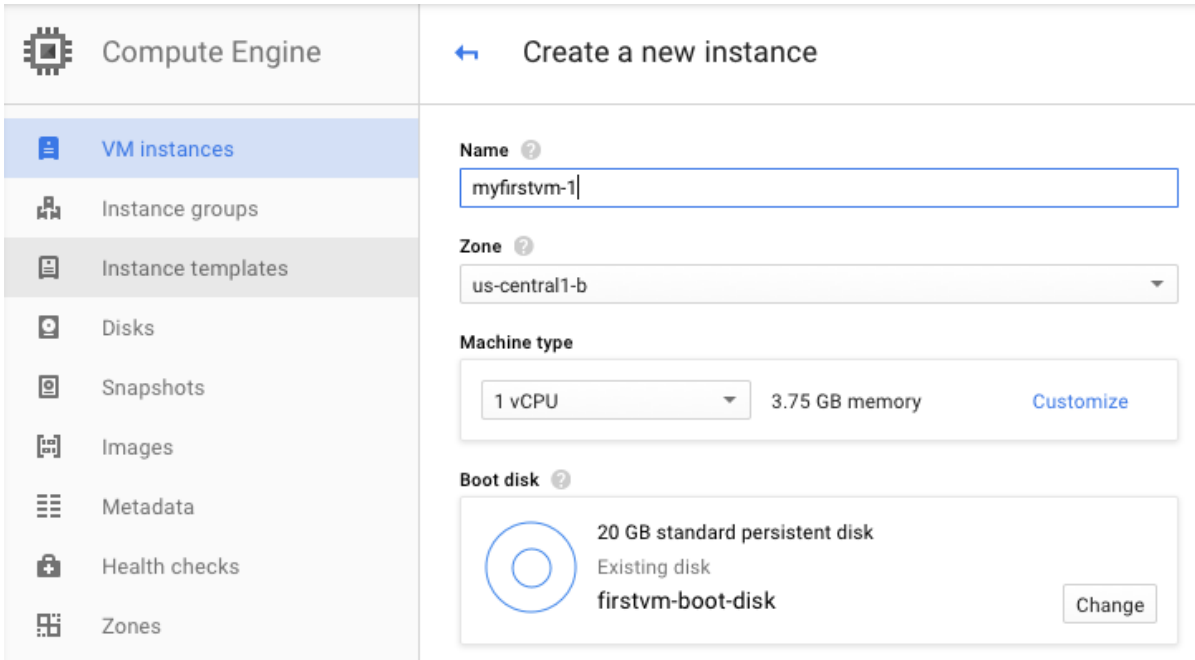
Snapshotting an instance

Once you have a snapshot, you can create a boot disk from it:



Snapshotting an instance

And once you have a boot disk, you can create a VM from it:



Replicating VMs

A boot disk can only be attached in RW mode to a single instance. To attach a boot disk in read-only mode to many instances, use

```
$ gcloud compute instances create vm-{1..3} --zone us-central1-b \
   --disk name=firstvm-boot-disk,boot=yes,mode=ro
```

A read-only boot disk is a pain! This is how we get identical snapshots with read/write boot disks:

```
$ gcloud compute disks create vm-{1..5}  \
    --source-snapshot "vm-snapshot"  --zone us-central1-b
$ for i in {1..5}; do gcloud compute instances create vm-$i \
    --zone=us-central1-b --disk name=vm-$i,mode=rw,boot=yes; done
```

Exercise

You created an n1-standard-1 VM to serve a website. However, since this VM is handling the load with very little CPU usage, you realize you can save money by using a shared-CPU instance.
Exercise: Use persistent boot disks to migrate your server from an n1-standard-1 instance to an f1-micro instance.

Exercise

Step 1: Create the standard instance:

```
$ gcloud compute instances create myserver \
    --zone=us-central1-b --machine-type=n1-standard-1 \
    --tags=http-server
Created [https://www.googleapis.com/compute/v1/projects/learngcloud-118
NAME     ZONE          MACHINE_TYPE  PREEMPTIBLE INTERNAL_IP EXTERNAL_I
myserver us-central1-b n1-standard-1             10.240.0.3  130.211.12
```

Exercise

Step 2: Start an nginx-server on this instance:

```
$ gcloud compute ssh myserver --zone=us-central1-b
myserver$ sudo apt-get update && sudo apt-get install -y nginx
myserver$ sudo chmod a+rw /var/www/html/index.nginx-debian.html
myserver$ echo "Hola!" > /var/www/html/index.nginx-debian.html
myserver$ exit
$ curl 130.211.120.11 # Make sure it works!
Hola!
```

Exercise

Step 3: Delete your instance, but keep the boot disk!

```
$ gcloud compute instances delete myserver \
   --keep-disks boot --zone=us-central1-b
 $ gcloud compute disks list
NAME              ZONE          SIZE_GB TYPE        STATUS
myserver          us-central1-b 10      pd-standard READY
```

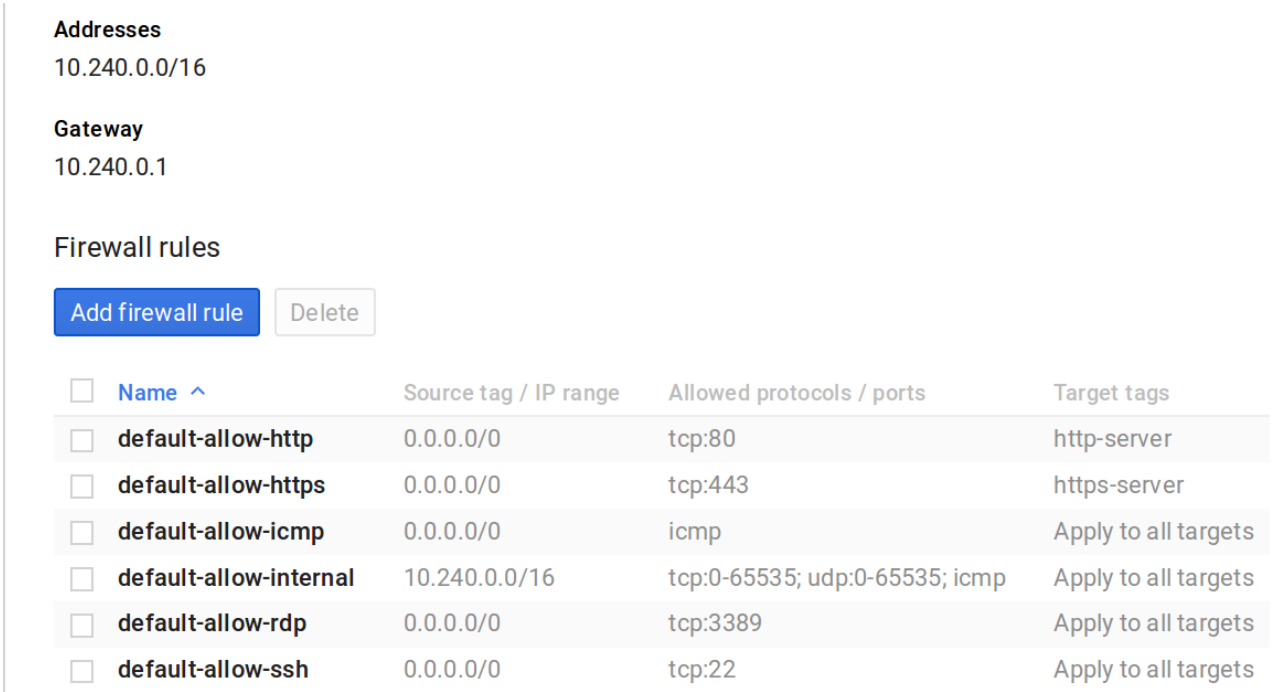
Exercise

Step 4: Create a new instance from the boot disk:

```
$ gcloud compute instances create smaller-server \
--zone=us-central1-b --machine-type=f1-micro \
    --tags=http-server --disk name=myserver,boot=yes,mode=rw
Created [https://www.googleapis.com/compute/v1/projects/learngcloud-118
NAME            ZONE          MACHINE_TYPE INTERNAL_IP EXTERNAL_IP     S
smaller-server us-central1-b f1-micro     10.240.0.3  130.211.120.111 R
$ curl 130.211.120.111
Hola!
```

Editing Firewalls

Google cloud has a set of default firewall rules that you might like to edit:



Addresses
10.240.0.0/16

Gateway
10.240.0.1

Firewall rules

Add firewall rule | Delete

Name	Source tag / IP range	Allowed protocols / ports	Target tags
default-allow-http	0.0.0.0/0	tcp:80	http-server
default-allow-https	0.0.0.0/0	tcp:443	https-server
default-allow-icmp	0.0.0.0/0	icmp	Apply to all targets
default-allow-internal	10.240.0.0/16	tcp:0-65535; udp:0-65535; icmp	Apply to all targets
default-allow-rdp	0.0.0.0/0	tcp:3389	Apply to all targets
default-allow-ssh	0.0.0.0/0	tcp:22	Apply to all targets

Note that all ports are available via the internal ip addresses 10.240.0.0/16

```
$ gcloud compute firewall-rules list
NAME                    NETWORK SRC_RANGES     RULES
default-allow-http      default 0.0.0.0/0      tcp:80
default-allow-https     default 0.0.0.0/0      tcp:443
default-allow-icmp      default 0.0.0.0/0      icmp
default-allow-internal default 10.240.0.0/16 tcp:0-65535,udp:0-65535,ic
default-allow-ssh       default 0.0.0.0/0      tcp:22
```

Editing Firewalls

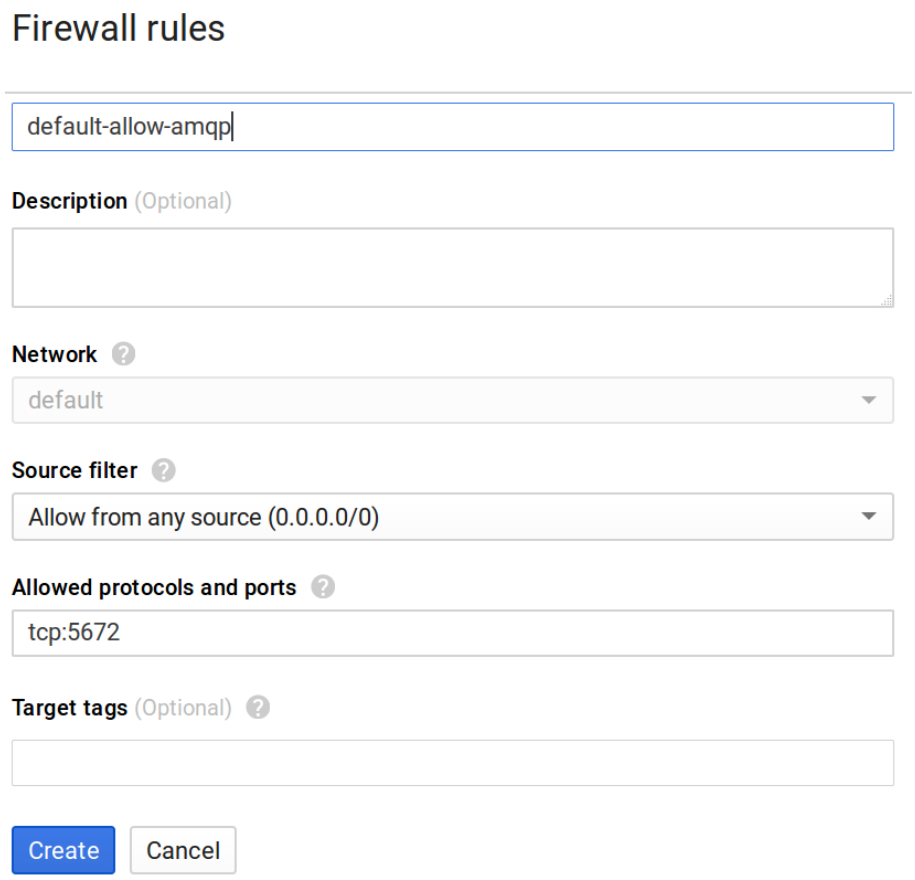
The default rules are of course reflected in the port scan:

```
$ nmap -p 0-10000 104.154.88.253

Starting Nmap 6.47 ( http://nmap.org ) at 2016-01-08 17:43 CST
Nmap scan report for 253.88.154.104.bc.googleusercontent.com (104.1
Host is up (0.040s latency).
Not shown: 9997 filtered ports
PORT     STATE  SERVICE
22/tcp   open   ssh
80/tcp   closed http
443/tcp  closed https
3389/tcp closed ms-wbt-server
```

Editing firewalls

This allows traffic from AMQP:



Editing firewalls

Command line to allow AMQP on port 5672 from any source IP, use the command

```
$ gcloud compute firewall-rules \
   create default-allow-amqp --allow tcp:5672 \
   --target-tags rabbit-server
Created [https://www.googleapis.com/compute/v1/projects/learngcloud-118
NAME                NETWORK SRC_RANGES RULES    SRC_TAGS TARGET_TAGS
default-allow-amqp default 0.0.0.0/0  tcp:5672
```

Creating a firewall rule applies it to all live instances in the project, unless you give it a target-tag.

Target Tags

Target tags give us an idea of what our instance is going to be used for. This tag allows traffic on port 80/443:

```
$ gcloud compute instances create webserver \
 --zone us-central1-b --tags http-server,https-server
```

Our custom amqp firewall rule will be applied to this server:

```
$ gcloud compute instanes create rabbit \
--zone us-central1-b --tags rabbit-server
```

Networking

Each project supports a $2^{16} = 65,536$ address internal network:

```
$ gcloud compute networks list
NAME    IPV4_RANGE    GATEWAY_IPV4
default 10.240.0.0/16 10.240.0.1
```

Network Load balancing

Let's set up a couple servers to learn how to load balance. First, our startup script.sh:

```
#!/bin/bash
sudo apt-get update
sudo apt-get install -y nginx emacs
sudo chmod a+rw /var/www/html/index.nginx-debian.html
sudo echo "Hello from $(hostname)" > /var/www/html/index.nginx-debian.h
```

Network Load balancing

Now create your servers with this startup script:

```
$ gcloud compute instances create server-{1..3} \
    --tags http-server --zone us-central1-a \
    --metadata-from-file startup-script=script.sh
```

Network Load Balancing

Now let's set up a "target-pool", a group of server that responds to requests:

```
$ gcloud compute target-pools create \
    mypool --region us-central1
$ gcloud compute target-pools add-instances \
    mypool --instances server-{1..3} --zone us-central1-a
$ gcloud compute forwarding-rules create \
  frontend-forwarding --target-pool mypool --region us-central1
NAME      REGION      IP_ADDRESS     IP_PROTOCOL TARGET
frontends us-central1 104.154.36.151 TCP         us-central1/targetPool
$ for i in {1..100}; do curl 104.154.36.151; done
Welcome to server-1!
Welcome to server-1!
Welcome to server-3!
Welcome to server-2!
Welcome to server-1!
```

Network Load Balancing

What happens if one of our servers stops?

```
$ for i in {1..100}; do curl 104.154.44.41; done
Hello from server-1!
Hello from server-3!
curl: (7) Failed to connect to 104.154.44.41 port 80: Connection refuse
```

The load balancer doesn't know anything about our servers, so it can't tell if it should send traffic there or not. Let's educate our load balancer:

Educating your load balancer

We need our load balancer to know that the server on port 80 should be available. Therefore, we create a health-check for it:

```
$ gcloud compute http-health-checks create frontend-check
Created [https://www.googleapis.com/compute/v1/projects/learngcloud-118
NAME           HOST PORT REQUEST_PATH
frontend-check      80   /
$ gcloud compute target-pools add-health-checks mypool \
   --http-health-check frontend-check --region=us-central1
Updated [https://www.googleapis.com/compute/v1/projects/learngcloud-118
```

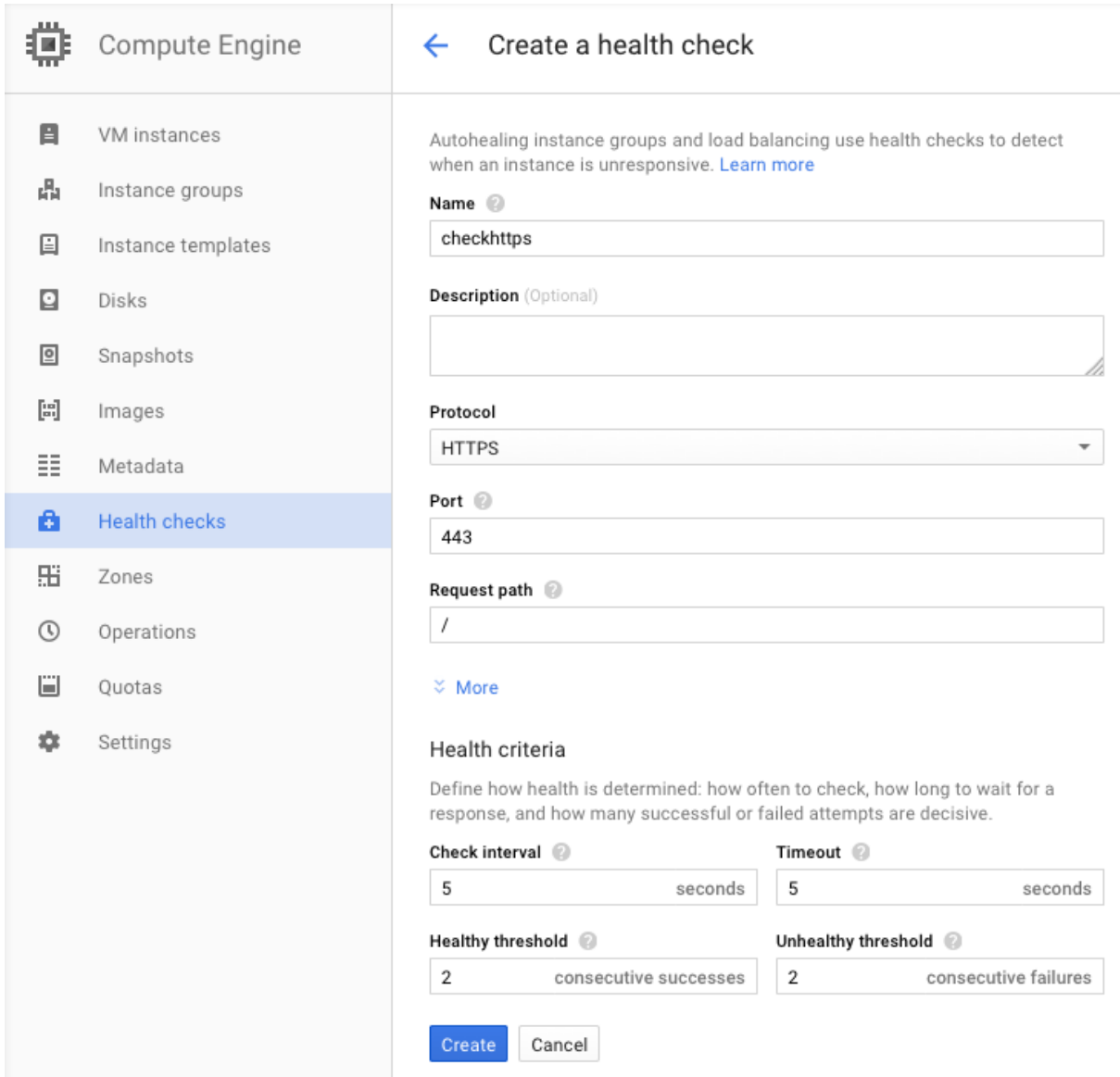
Network Load Balancer with Health Checks

Now we shouldn't see many failed requests:

```
$ for i in {1..100}; do curl 104.154.44.41; done
Hello from server-1!
Hello from server-3!
Hello from server-1!
Hello from server-1!
Hello from server-3!
Hello from server-3!
Hello from server-3!
Hello from server-1!
Hello from server-3!
Hello from server-1!
Hello from server-3!
Hello from server-1!
Hello from server-1!
Hello from server-3!
Hello from server-3!
```

Educating your load balancer

For those who don't like the command line, of course we can create the health check in the console:



Session Affinity

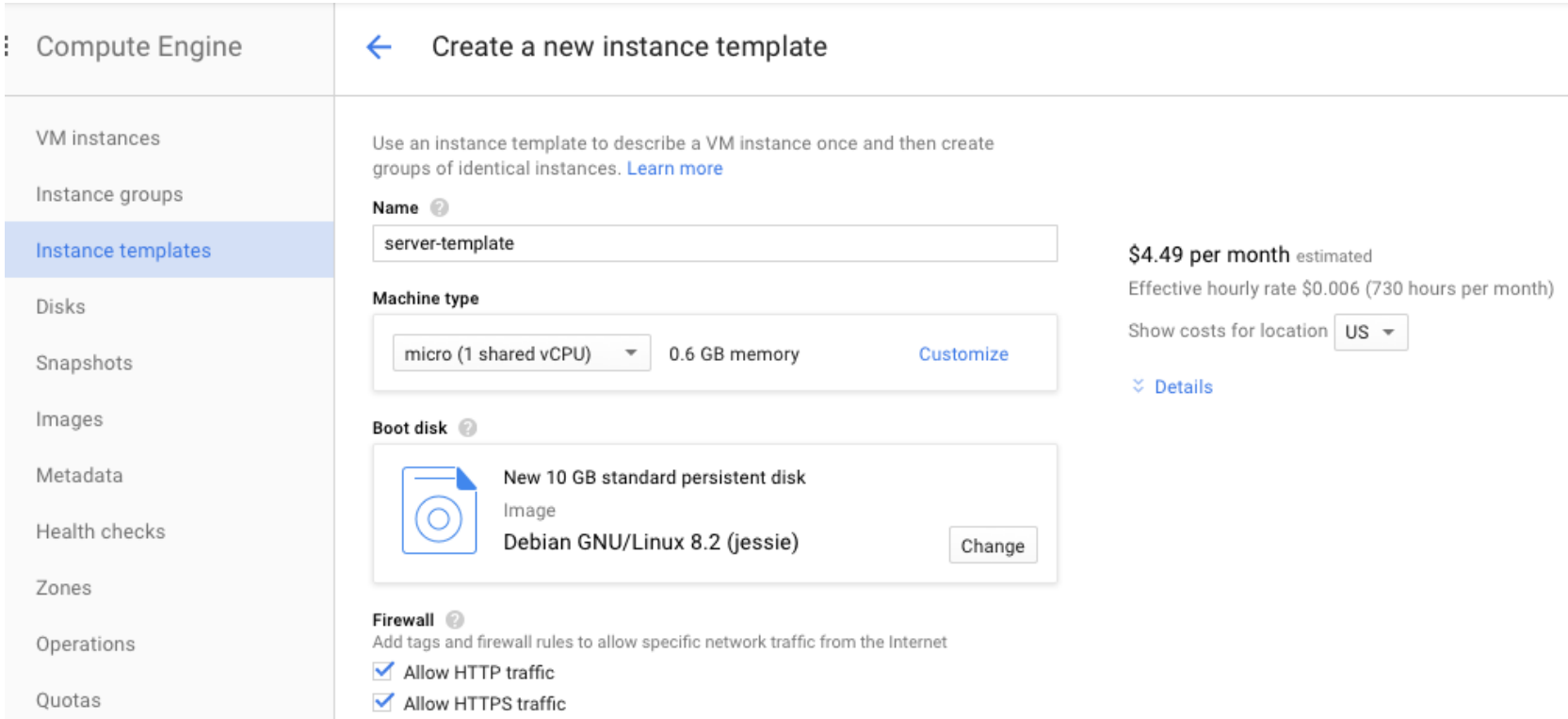
What happens if we don't want each http request from a given client to go to a different backend server? We need to give *session affinity* to our target pool:

```
$ gcloud compute target-pools create mypoolwsession \
    --session-affinity CLIENT_IP --region us-central1 \
    --health-check check80
```

This ensures that all traffic from a single IP winds up at the same backend server, as long as that server is serving traffic on port 80 successfully.

Instance Templates

Before autoscaling, we need to create an *instance template*, which describes how big we want our nodes, what OS, and perhaps a startup script. Remember, these nodes must be identical.



You can also use a custom image instead of an instance template; this is preferred for complex deployments.

Instance Templates

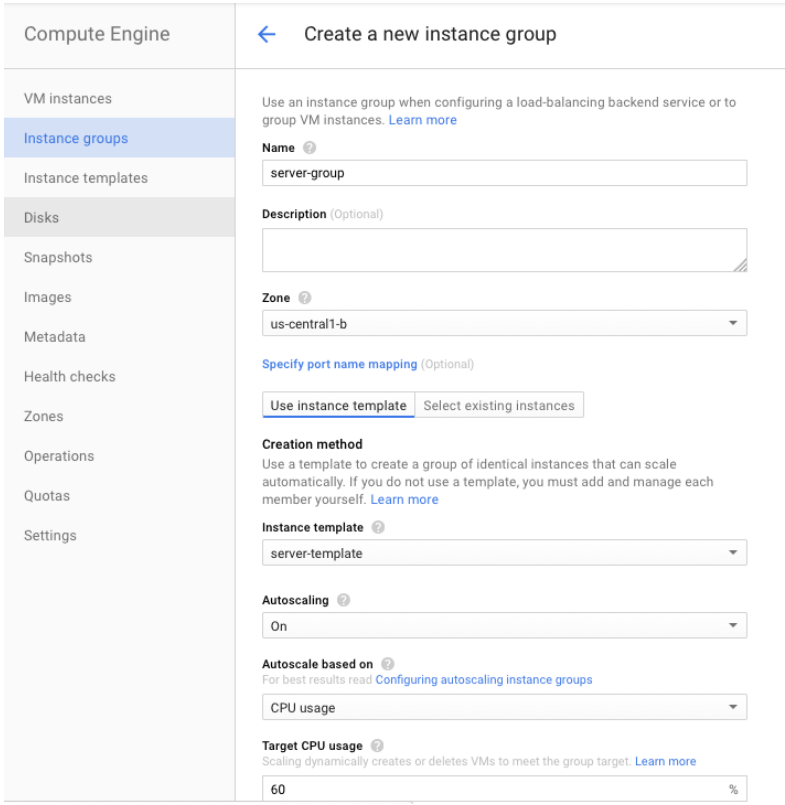
To create the instance template from the command line using a startup script, use:

```
$ gcloud compute instance-templates create my-instance-template \
  --machine-type=f1-micro --scopes=compute-rw,storage-full \
   --tags=http-server \
    --metadata-from-file startup-script=server_startup.sh
```

Question 1: Why didn't we need to specify a zone here?
Question 2: Why do we need to tag this with "http-server"?

Instance Groups

Once we have an instance template, we can create an autoscaling instance group:



Instance Groups

Command-line for creating instance groups:

```
$  gcloud compute instance-groups managed create my-instance-group \
   --zone us-central1-b --template my-instance-template \
    --base-instance-name my-instance-group --size 1
```

Autoscaling an Instance Group

Command line:

```
$ gcloud compute instance-groups managed \
  set-autoscaling "my-instance-group" \
  --zone "us-central1-b" --cool-down-period "60" \
  --max-num-replicas "4" --min-num-replicas "1" \
  --target-cpu-utilization "0.6"
```

This should now start autoscaling based on load!

Zonal Data Disks

In addition to creation of boot disks, you can create zonal data disks:

```
$ gcloud compute disks create "data-disk" \
   --size=200 --zone=us-central1-b
Created [https://www.googleapis.com/compute/v1/projects/learngcloud-118
NAME      ZONE          SIZE_GB TYPE        STATUS
data-disk us-central1-b 200     pd-standard READY
```

Zonal Data Disks

We can now mount this on an instance:

```
$ gcloud compute instances create datavm \
   --scopes=compute-rw --zone=us-central1-b
$ gcloud compute ssh datavm --zone=us-central1-b
datavm$ gcloud compute instances attach-disk datavm \
        --disk data-disk --zone=us-central1-b
datavm$ sudo mkdir -p /mnt/pd0
datavm$ ls -l /dev/disk/by-id/google*
datavm$ sudo /usr/share/google/safe_format_and_mount -m "mkfs.ext4 -F"
              /dev/disk/by-id/google-persistent-disk-1 /mnt/pd0
datavm$ lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda      8:0    0   10G  0 disk
??sda1   8:1    0   10G  0 part /
sdb      8:16   0  200G  0 disk /mnt/pd0
```

Zonal Data Disks

Zonal data disks can only be mounted read/write on a single instance. However, they can be mounted read-only on many instances:

```
$ gcloud compute instances attach-disk datavm \
   --disk data-disk --mode ro
$ gcloud compute ssh datavm
datavm$ lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
 sda      8:0    0   10G  0 disk
-->sda1   8:1    0   10G  0 part /
 sdb      8:16   0  200G  1 disk
$ sudo /usr/share/google/safe_format_and_mount \
    -m "mkfs.ext4 -F" /dev/sdb /mnt/pd2
```

Zonal Data Disks

Resizing zonal data disks is super-easy:

```
$ gcloud beta compute disks resize data-disk --size 250GB
```

It's easier to manage one huge disk than tons of small ones, and throughput is higher on larger disks.

Zonal Data Disks

Exercise: Attach a zonal data disk (read-only) to a preemptible f1-micro instance during instance creation. Use a startup script to mount the volume.

Solution

Let's begin with our startup script, call it "attach_disk.sh":

```
#!/bin/bash
stat /dev/disk/by-id/google-my-pd
if [ $? -ne 0 ]; then
    echo "Block device not found."
    exit 1;
fi
sudo mkdir /mnt/pd0
sudo /usr/share/google/safe_format_and_mount \
    -m "mkfs.ext4 -F" /dev/disk/by-id/google-my-pd /mnt/pd0
```

Solution

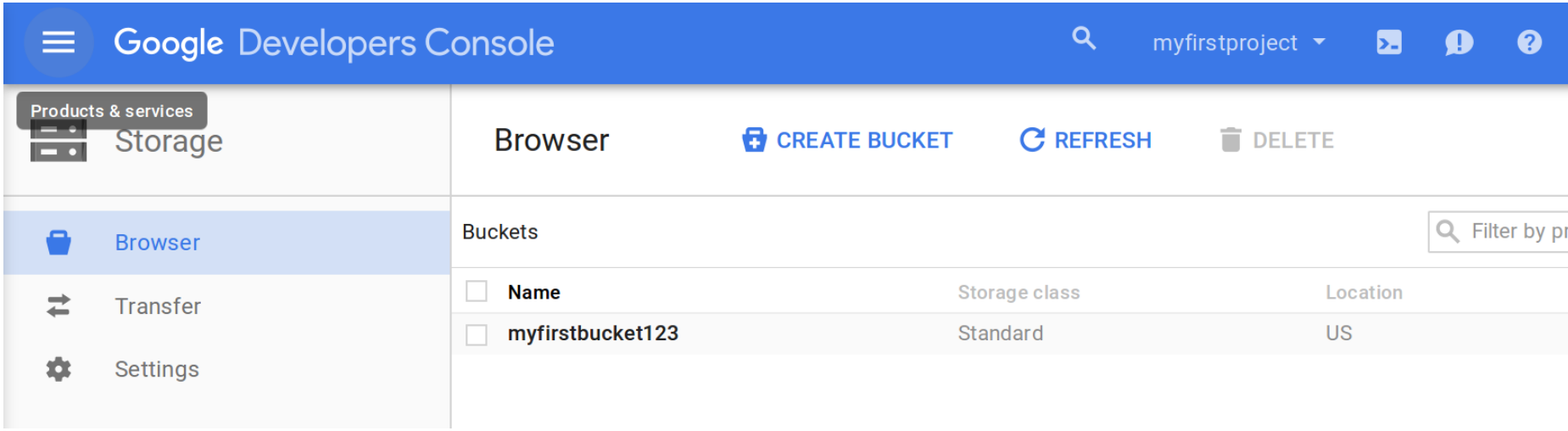
Now create the instance:

```
$ gcloud compute instances create testing123 \
    --disk name=data-disk,boot=no,mode=ro,device-name=my-pd
    --machine-type f1-micro --preemptible
    --metadata-from-file startup-script=attach_disk.sh
```

(Notice the association between the mount point and the device name?)

Persistent Disks

If you need storage that is not chained to a particular zone, then you need to use "buckets":



Persistent Disks

In order to manage persistent disks from the command line, use the gsutil utility:

```
$ pip install gsutil
$ gsutil mb gs://myfirstbucket
Creating gs://myfirstbucket/...
ServiceException: 409 Bucket myfirstbucket already exists.
$ gsutil mb gs://myfirstbucket123
Creating gs://myfirstbucket123/...
$ gsutil cp talk.log gs://myfirstbucket123
Copying file://talk.log [Content-Type=application/octet-stream]...
Uploading   gs://myfirstbucket123/talk.log:                              43
$ gsutil ls
gs://myfirstbucket123/
$ gsutil ls -l gs://myfirstbucket123
44993  2016-01-09T21:19:30Z  gs://myfirstbucket123/talk.log
TOTAL: 1 objects, 44993 bytes (43.94 KiB)
```

Bucket names need to be globally unique across all of gcloud!

Persistent Disks

- gcloud buckets are encrypted on disk
- gcloud buckets are accessible outside zones (i.e. have global scope)
- Uploads to gcloud buckets are atomic and considered successful once the info is stored in multiple datacenters.

Persistent Disks

Mounting a bucket to an instance takes a little work (instructions). First let's create a startup script called can-bucket.sh:

```
#!/bin/bash
sudo apt-get update
sudo apt-get install -y fuse
sudo curl -L -O \
https://github.com/GoogleCloudPlatform/gcsfuse/releases/download/v0.15.
sudo dpkg --install gcsfuse_0.15.0_amd64.deb
```

Persistent Disks

Now let's create some instances which have permission to view our bucket:

```
$ gcloud compute instances create can-bucket-{1..3} \
    --zone=us-central1-b --scopes=storage-full \
    --metadata-from-file startup-script=can-bucket.sh
```

The "scopes=storage-full" gives our instances permission to mount our buckets.

Persistent Disks

```
$ gcloud compute ssh can-bucket-1 --zone=us-central1-b
can-bucket-1$ mkdir mount_point
can-bucket-1$ gcsfuse myfirstbucket123 mount_point
can-bucket-1$ touch mount_point/file.txt
```

That'll do it! Try it on your other instances!

Persistent Disks

A few more gsutil examples:

```
$ gsutil rsync gs://myfirstbucket123/  `pwd`
Building synchronization state...
Starting synchronization
Copying gs://myfirstbucket123/talk.log...
Downloading file:///home/NAThompson/talk.log:                    43.94
$ gsutil rsync `pwd` gs://myfirstbucket123/foo
Building synchronization state...
Starting synchronization
...
$ gsutil du -h gs://myfirstbucket123
43.94 KiB   gs://myfirstbucket123/talk.log
```

HTTP Load Balancing

Our network load balancer was a regional resource. What if we want low latency connections to anyone in the world?
We'll do this by setting up *HTTP load balancing*.

Exercise

Construct a load-balancing, fault tolerant, autoscaling webserver using an http load balancer.

Solution

Since we want to demonstrate that the autoscaler works under heavy traffic, let's have nginx serve a large image; here's some from NASA:
http://earthobservatory.nasa.gov/IOTD/view.php?id=77627
Put whatever image you chose in your bucket; I called mine "nasa.jpg"

```
$ gsutil cp nasa.jpg gs://myfirstbucket123
```

Solution

Now we need to create a startup script; call it autoscaler_startup.sh:

```
#!/bin/bash
sudo apt-get update
sudo apt-get install -y nginx
sudo chmod a+rw /var/www/html/index.nginx-debian.html
sudo echo "<!DOCTYPE html>
<html>
<head>
<title>$(hostname)</title>
</head>
<body>
<img src='nasa.jpg'>
</body>
</html>" > /var/www/html/index.nginx-debian.html
sudo gsutil cp gs://myfirstbucket123/nasa.jpg /var/www/html/nasa.jpg
```

Solution

Now let's create an instance template:

```
$ gcloud compute instance-templates create server-template \
--machine-type f1-micro --scopes=storage-full --tags http-server \
--metadata-from-file startup-script=autoscaler_startup.sh
```

(Why do we need the tag http-server? Why do we need the scopes=storage-full? Why don't we need –zone?)

Solution

Now create the base instance of the autoscaling group:

```
$ gcloud compute instance-groups managed create usc1b \
  --template server-template --size 1 --base-instance-name usc1b \
  --zone us-central1-b
$ gcloud compute instance-groups managed create euw1b \
  --template server-template --size 1 --base-instance-name euw1b \
  --zone europe-west1-b
```

Solution

Make your instance group autoscaling:

```
$ gcloud compute instance-groups managed set-autoscaling euw1b \
  --zone europe-west1-b --cool-down-period 180 --max-num-replicas 5 \
  --min-num-replicas 1 --target-load-balancing-utilization 0.2
$ gcloud compute instance-groups managed set-autoscaling usc1b \
  --zone us-central1-b --cool-down-period 180 --max-num-replicas 5 \
  --min-num-replicas 1 --target-load-balancing-utilization 0.2
```

Solution

Announce that your autoscaling groups are listening on port 80:

```
$ gcloud compute instance-groups set-named-ports usc1b \
--zone us-central1-b --named-ports http:80
$ gcloud compute instance-groups set-named-ports euw1b \
--zone europe-west1-b --named-ports http:80
```

Solution

Create your health check for fault-tolerance:

```
$ gcloud compute http-health-checks create check80
```

Solution

Create a backend service:

```
$ gcloud compute backend-services create web-service \
--http-health-check check80
```

Solution

Add your autoscaling groups to your backend service:

```
$ gcloud compute backend-services add-backend web-service \
  --instance-group usc1b --zone us-central1-b
$ gcloud compute backend-services add-backend web-service \
  --instance-group euw1b --zone europe-west1-b
```

Solution

Create an URL map:

```
$ gcloud compute url-maps create web-map \
--default-service web-service
```

and a target http proxy:

```
$ gcloud compute target-http-proxies \
create web-proxy --url-map web-map
```

Now create a forwarding rule:

```
$ gcloud compute forwarding-rules create http-rule \
--global --target-http-proxy web-proxy --port-range 80
```

Solution

The following will request the image over and over, and should trip the autoscaler:

```
$ while true; do wget --no-parent --accept=jpg \
--mirror http://130.211.15.20/; rm -rf 130.211.15.20; done;
```

Preemptible Instances

You can easily halve your bill by using preemptible compute instances:

```
$ gcloud compute instances create preempt-me --preemptible
```

The instance can be shutdown by Google at any time, and is guaranteed to be shutdown in 24 hours.

Preemptible Instances

In order to shutdown gracefully, gcloud allows you to define a shutdown script, which is a set of actions performed when the computer shuts down:

```
$ gcloud compute instances create preempt-me --preemptible \
 --metadata-from-file shutdown-script=shutdown.sh
```

Preemptible Instances

Example shutdown script:

```
#!/bin/bash
echo "Shutdown requested at $(date)"
echo "Attempting to gracefully shutdown nginx:"
nginx -s quit #Keeps serving open requests, doesn't accept new.
```

Preemptible Instances

We can test our shutdown script by stopping the instance in the console. The result should show:

```
$ cat /var/log/shutdownscript.log
Jan 28 04:48:46 preempt3 shutdownscript: Running shutdown script /var/r
Jan 28 04:48:46 preempt3 shutdownscript: Shutdown requested at Thu Jan
Jan 28 04:48:46 preempt3 shutdownscript: Attempting to gracefully shutd
Jan 28 04:48:46 preempt3 shutdownscript: Finished running shutdown scri
```

Estimating Costs

gcloud gives a nice website for estimating costs:
https://cloud.google.com/products/calculator

References

- Google Compute Engine, by Marc Cohen, Kathryn Hurley, and Paul Newson
- Building Your Next Big Thing With Google Cloud Platform, by Jose Gonzales and S. Krishnan.