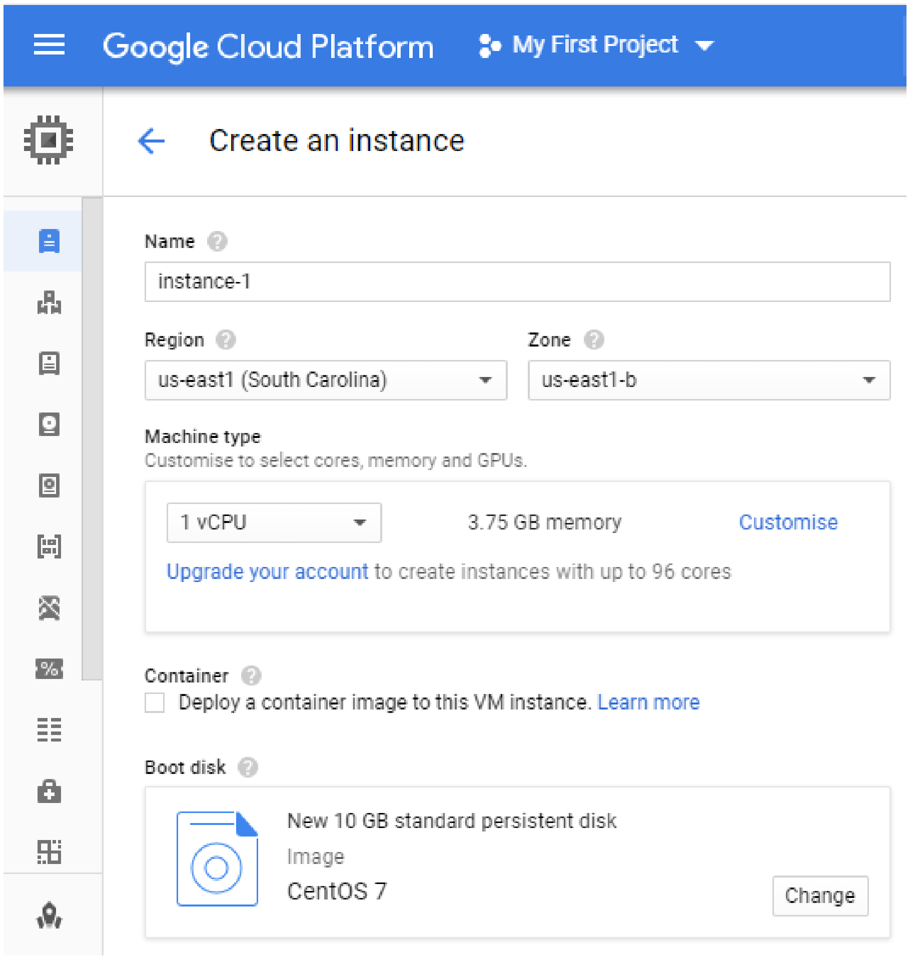
Uploading a Shiny app to a server

1. Creating a sever instance

For this example, we use Google Cloud (https://console.cloud.google.com/freetrial). Amazon Web Services (https://aws.amazon.com/) or Microsoft Azure (https://azure.microsoft.com/) are all comparable solutions, all with free credit offers for new accounts. The only advantage of Google is that it doesn't require additional software downloads to interact with a Linux server.

Begin by setting up your account on the above URL using any Google account (Google Mail, YouTube, etc.). Having done this, navigate to "VM Instances" (for Virtual Machine) from the main navigation (https://console.cloud.google.com/compute/instances). Here you can create a virtual server to host your code. Having selected create you should see a screen like the below:



Name: instance-1

Region: us-east1 (South Carolina) Zone: us-east1-b

Machine type
Customise to select cores, memory and GPUs.

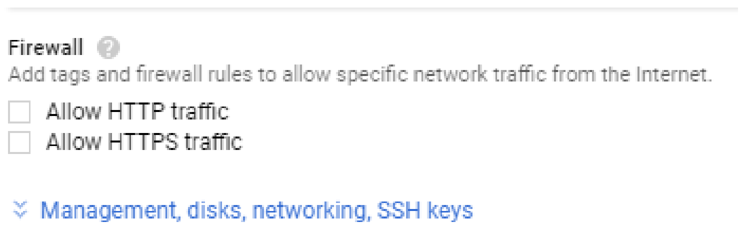
1 vCPU 3.75 GB memory

Upgrade your account to create instances with up to 96 cores

Container
Deploy a container image to this VM instance. Learn more

Boot disk
New 10 GB standard persistent disk
Image
CentOS 7

There are various options and customisations available here, far too many to go into, so just the key parts are highlighted. Naming the instance is useful if you are maintaining several VMs. You can also change the machine specifics in terms of CPUs (computing power – although it is worth noting that "out-of-the-box" R, and therefore Shiny, is a single-core application), RAM (amount of data that can be held in memory) and storage space. In most cases the standard settings are fine and it is possible to upgrade later. You should also select your "flavour" of Linux. My preference is CentOS 7, which is what the below instructions will be based on (although Ubuntu can also be used). Further down is a link labelled "Management, disks, networking, SSH keys". We'll need to check to allow HTTP traffic (so users can visit the app) and expand the options.


Firewall

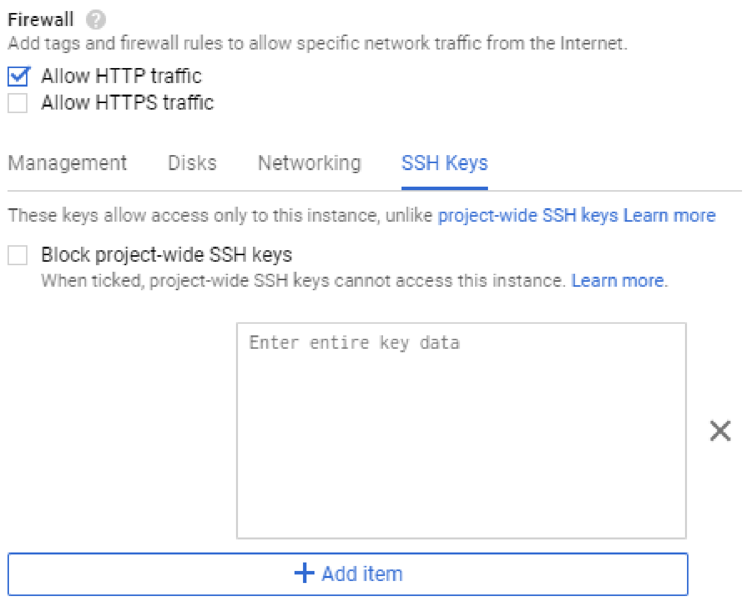
Add tags and firewall rules to allow specific network traffic from the Internet.

☐ Allow HTTP traffic

☐ Allow HTTPS traffic

Management, disks, networking, SSH keys

We'll be adding an SSH key (basically a code-based password) so that we can securely log into the instance for transferring files and similar tasks. You can either use a pre-existing SSH, or one can be generated (for Windows: https://www.ssh.com/ssh/putty/windows/puttygen; and for Mac: https://docs.joyent.com/public-cloud/getting-started/ssh-keys/generating-an-ssh-key-manually/manually-generating-your-ssh-key-in-mac-os-x). Having created the key you need to paste it directly into the Google Console in the space below:


Firewall

Add tags and firewall rules to allow specific network traffic from the Internet.

☑ Allow HTTP traffic

☐ Allow HTTPS traffic

Management Disks Networking SSH Keys

These keys allow access only to this instance, unlike project-wide SSH keys Learn more

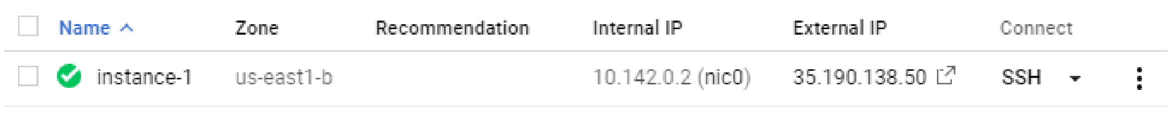
☐ Block project-wide SSH keys

When ticked, project-wide SSH keys cannot access this instance. Learn more.

Enter entire key data

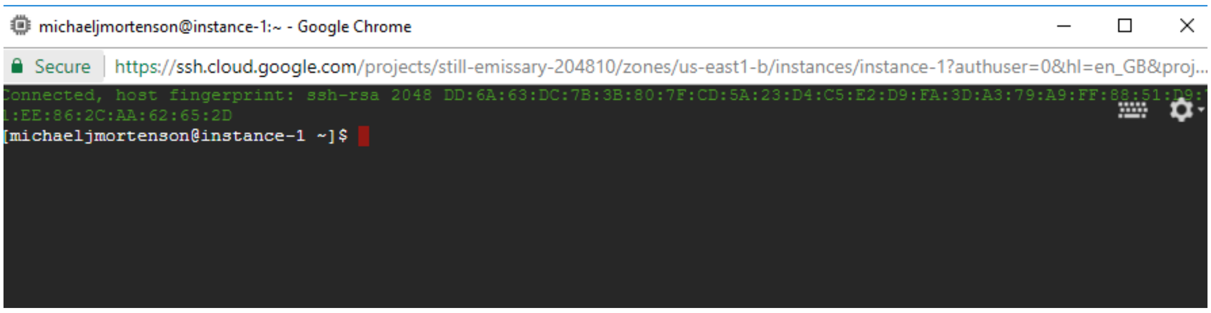
+ Add item

Having completed this step click on “Create” and your virtual machine will be built. This may take a short amount of time, but when finished you will see a green tick next to your instance name:



Name ^	Zone	Recommendation	Internal IP	External IP	Connect
✓ instance-1	us-east1-b		10.142.0.2 (nic0)	35.190.138.50	SSH ▾

One of the main advantages of Google is that we can now connect directly to the instance by selecting “Open in browser window” under the dropdown menu by “SSH”. This should open up a pop-up window with the interface to our machine:



michaeljmortenson@instance-1:~ - Google Chrome

Secure | https://ssh.cloud.google.com/projects/still-emissary-204810/zones/us-east1-b/instances/instance-1?authuser=0&hl=en_GB&proj...

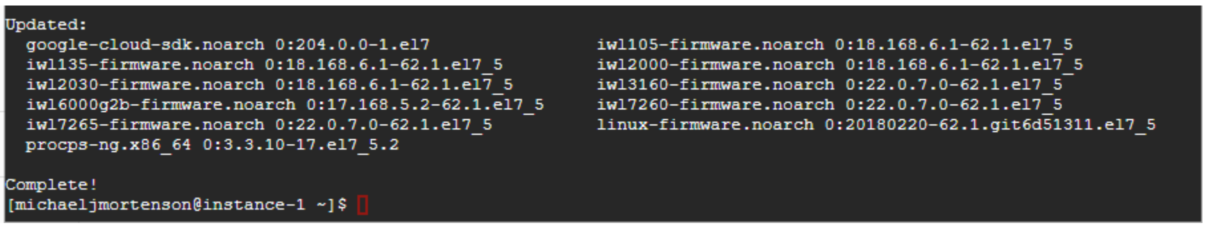
```
onnected, host fingerprint: ssh-rsa 2048 DD:6A:63:DC:7B:3B:80:7F:CD:5A:23:D4:C5:E2:D9:FA:3D:A3:79:A9:FF:88:51:D9:
1:EE:86:2C:AA:62:65:2D
[michaeljmortenson@instance-1 ~]$
```

From here we can start to configure our machine and install the necessary programs. To begin with we will make sure everything is up to date using the following commands:

```
sudo yum install epel-release
```

```
sudo yum update
```

All being well you will be shown a total download size (in my practice run its 90mb) and prompted if this is OK. “y” (for yes) will then download any additional packages. Once finished you will see a message saying “Complete!” and the command prompt will be back:



```
Updated:
  google-cloud-sdk.noarch 0:204.0.0-1.el7                iwl105-firmware.noarch 0:18.168.6.1-62.1.el7_5
  iwl135-firmware.noarch 0:18.168.6.1-62.1.el7_5         iwl2000-firmware.noarch 0:18.168.6.1-62.1.el7_5
  iwl2030-firmware.noarch 0:18.168.6.1-62.1.el7_5        iwl3160-firmware.noarch 0:22.0.7.0-62.1.el7_5
  iwl6000g2b-firmware.noarch 0:17.168.5.2-62.1.el7_5     iwl7260-firmware.noarch 0:22.0.7.0-62.1.el7_5
  iwl7265-firmware.noarch 0:22.0.7.0-62.1.el7_5          linux-firmware.noarch 0:20180220-62.1.git6d51311.el7_5
  procps-ng.x86_64 0:3.3.10-17.el7_5.2

Complete!
[michaeljmortenson@instance-1 ~]$
```

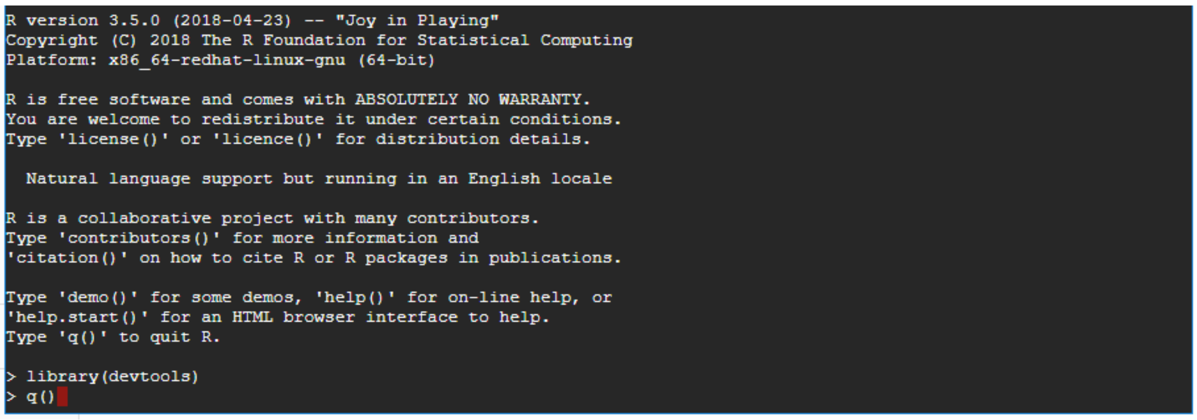
Next step is installing R and Shiny, with the following commands:

```
sudo yum install libcurl-devel

sudo yum install openssl-devel

sudo yum install wget

sudo yum install R

sudo su - -c "R -e \"install.packages(c('curl', 'shiny',
'rmarkdown', 'devtools', 'RJDBC', 'shinydashboard', 'plotly', 'DT',
'dplyr', 'e1071', 'rpart'), repos='http://cran.rstudio.com/')\""
```

Again, you'll be prompted to agree to the downloads with a "y" for yes and there may be a small delay. We'll test everything worked by opening up R (just type in "R" into the prompt), and trying to bring a library in memory using the following command:

```
library(devtools)
```

All being well you will see no error message after the command, which shows everything is working as it should be (afterwards you can exit R using the command "q()" – q for quit – and select "n" to say you don't want to save your workspace):



```
R version 3.5.0 (2018-04-23) -- "Joy in Playing"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-redhat-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> library(devtools)
> q()
```

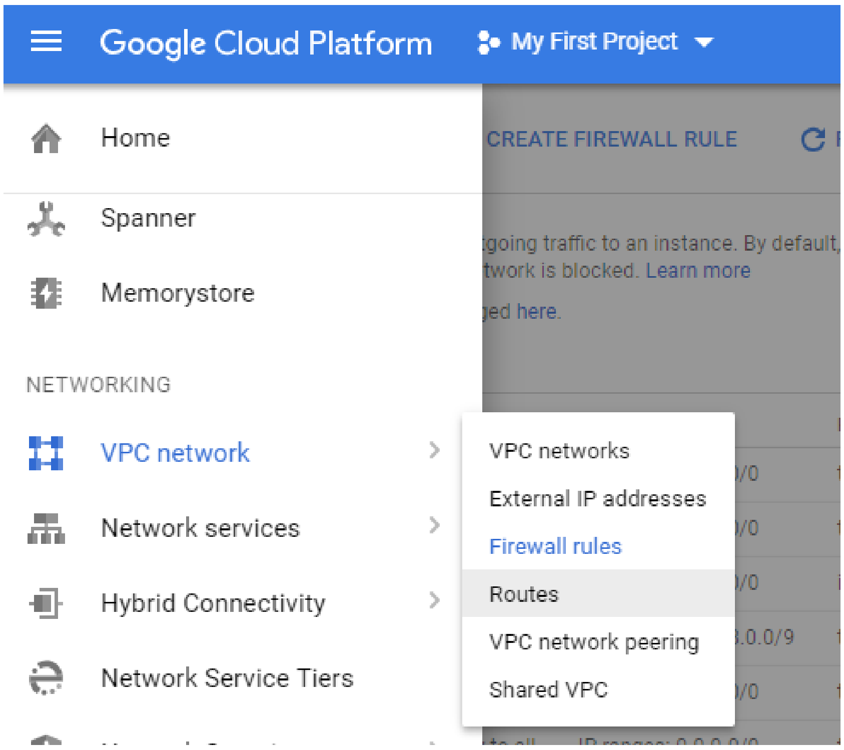
At this point we're ready to install Shiny Server. Note: the below code will change from time to time with new releases, so its worth checking the website (https://www.rstudio.com/products/shiny/download-server/) for the latest URLs:

```
wget https://download3.rstudio.org/centos6.3/x86_64/shiny-server-
1.5.7.907-rh6-x86_64.rpm

sudo yum install --nogpgcheck shiny-server-1.5.7.907-rh6-x86_64.rpm
```

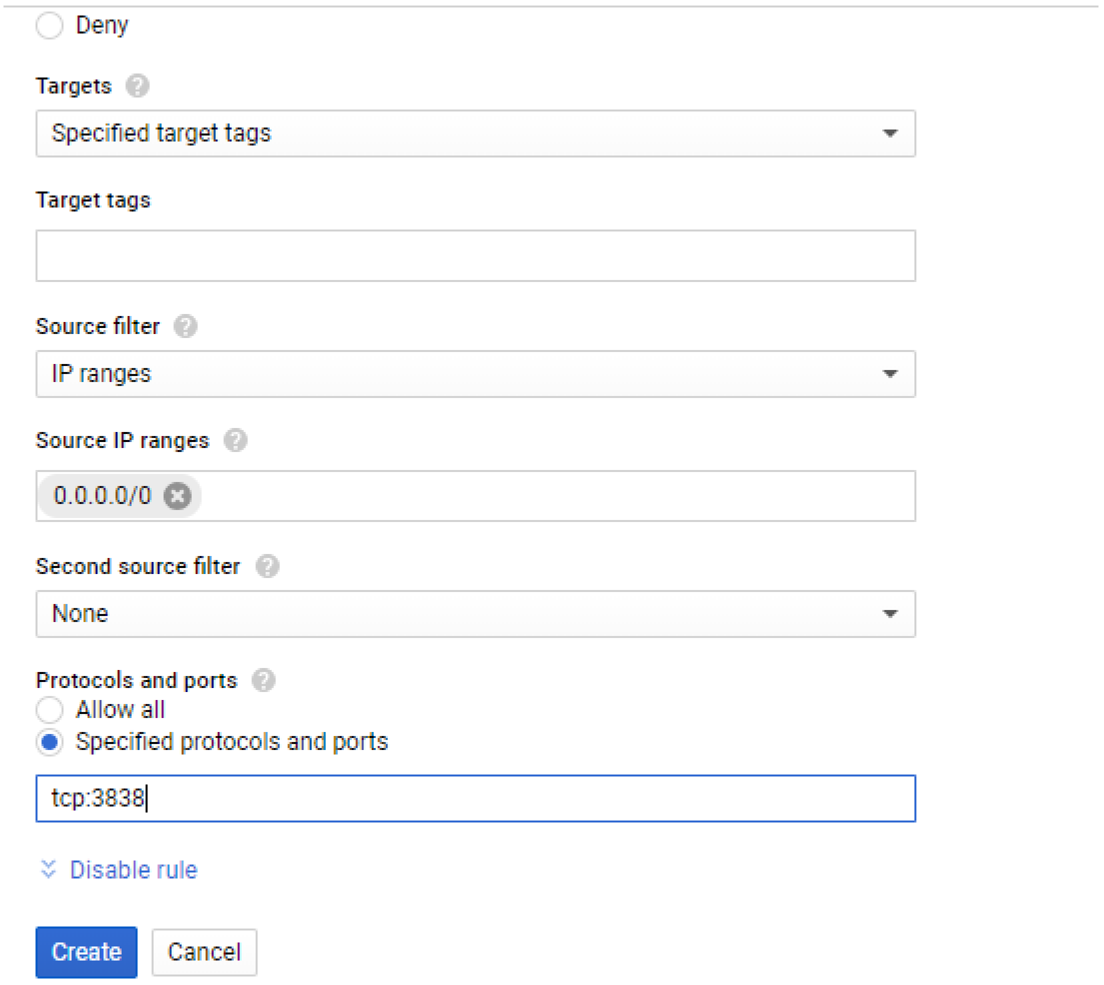
Finally, we want to start Shiny Server and enable it to run whenever the server starts (i.e. make it run permanently). We can do this using systemctl and the following commands:

```
sudo systemctl start shiny-server

sudo systemctl enable shiny-server
```

At this point everything should be installed and working! We just need to enable access for users to see the app. To do this return to the Google Console and create a "firewall rule", i.e. enable access to port 3838 where Shiny Server serves content. Navigate to "VPC network" from the main menu ☰ and then "Firewall rules".



We want to "CREATE FIREWALL RULE" from the top menu. Fill in the form including a name for the rule. At the "Targets" option we need to tell Google which VMs we want this to apply to, either by picking specific instances or just "All instances in the network" (if you only use the account to run Shiny apps). We also need to specify who the rule applies to. In "Destination filter" select "IP ranges". Here we can either grant access only to specific IP addresses or address ranges (e.g. make the app only available to someone in a client's office), or we can allow access to anyone. Assuming the latter the "Destination IP ranges" box will be filled with "0.0.0.0/0". Finally we want to tell Google which port this applies to. In "Protocols and ports" select "Specified protocols and ports" and then type in "tcp:3838" (this means we provide normal http access to port 3838 where Shiny runs).



Deny

Targets: Specified target tags

Target tags:

Source filter: IP ranges

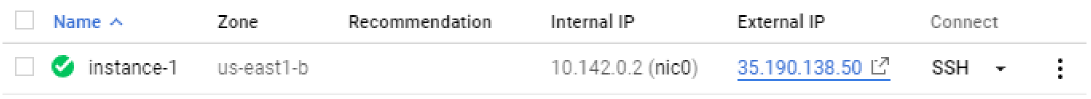
Source IP ranges: 0.0.0.0/0

Second source filter: None

Protocols and ports
Allow all
[✓ Specified protocols and ports]

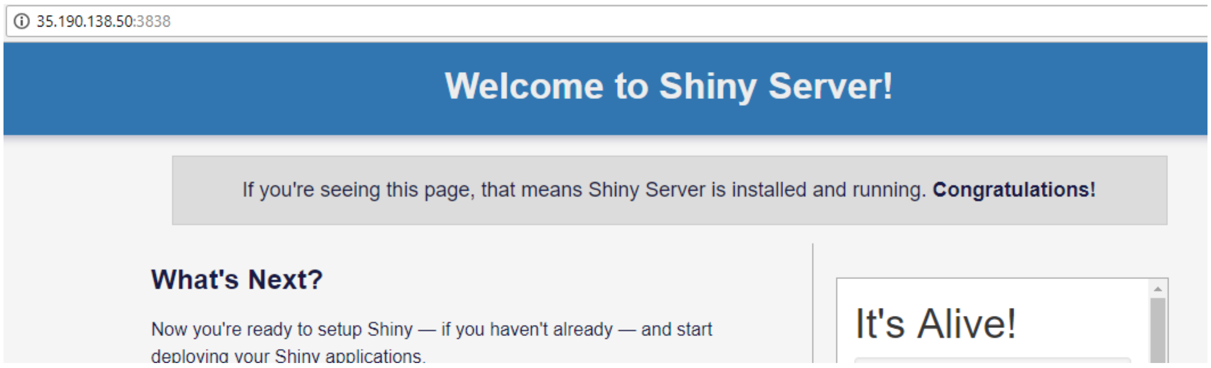
tcp:3838

To test this, go back to the Google Console and click on the external IP (35.190.138.50 in this case).



Name	Zone	Recommendation	Internal IP	External IP	Connect
instance-1	us-east1-b		10.142.0.2 (nic0)	35.190.138.50	SSH

This will throw a "page cannot be found error" as it will assume port 80, which has no content on. However, if you add to the end of the URL ":3838" then you should get access and see the page below (so the full URL is now http://35.190.138.50:3838 - replacing 35.190.138.50 with the IP assigned to your machine).


35.190.138.50:3838

Welcome to Shiny Server!

If you're seeing this page, that means Shiny Server is installed and running. **Congratulations!**

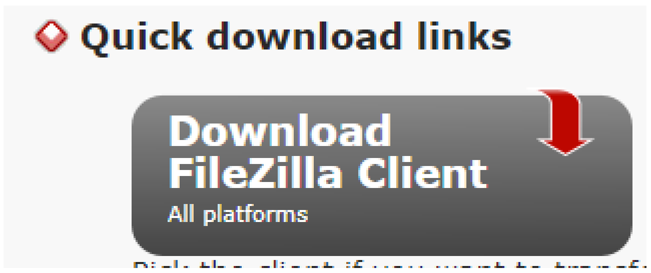
What's Next?

Now you're ready to setup Shiny — if you haven't already — and start deploying your Shiny applications.

It's Alive!

2. Uploading your app to the server

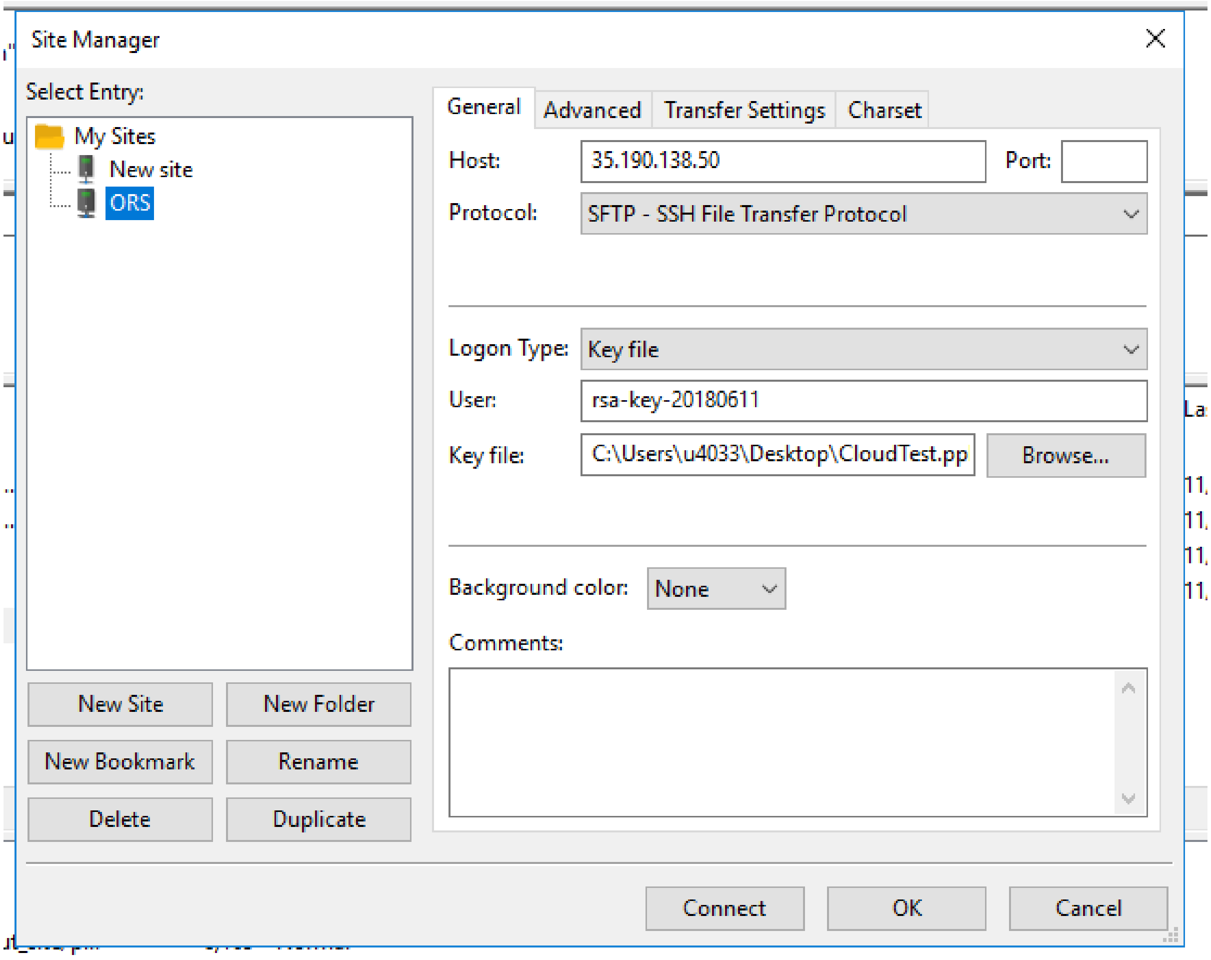
Now that everything is running, we want to be able to move our app online rather than the test. Fortunately, this quite easy to do using an FTP client, in this case we use FileZilla (https://filezilla-project.org). From this URL, download the latest version of the software from this link:



FileZilla allows us to easily, drag 'n drop files from our local machine to the VM. Firstly, however, we need to give permission to write to the relevant folder on the VM. In the browser access to the machine we run the following command (warning: we need to reverse this later on for security reasons!):

```
sudo chmod 777 -R /srv/shiny-server
```

We can now start transferring. In FileZilla select "File > Site Manager" which will produce the below interface . The "Host" is the IP address of the instance (without the port information ":3838") and "Protocol" should be set as SFTP. The "Logon Type" is "Key file" (as we will use the key we generated earlier).


Site Manager

Select Entry:

- My Sites
  - New site
  - ORS

General | Advanced | Transfer Settings | Charset

Host: 35.190.138.50 Port:

Protocol: SFTP - SSH File Transfer Protocol

Logon Type: Key file

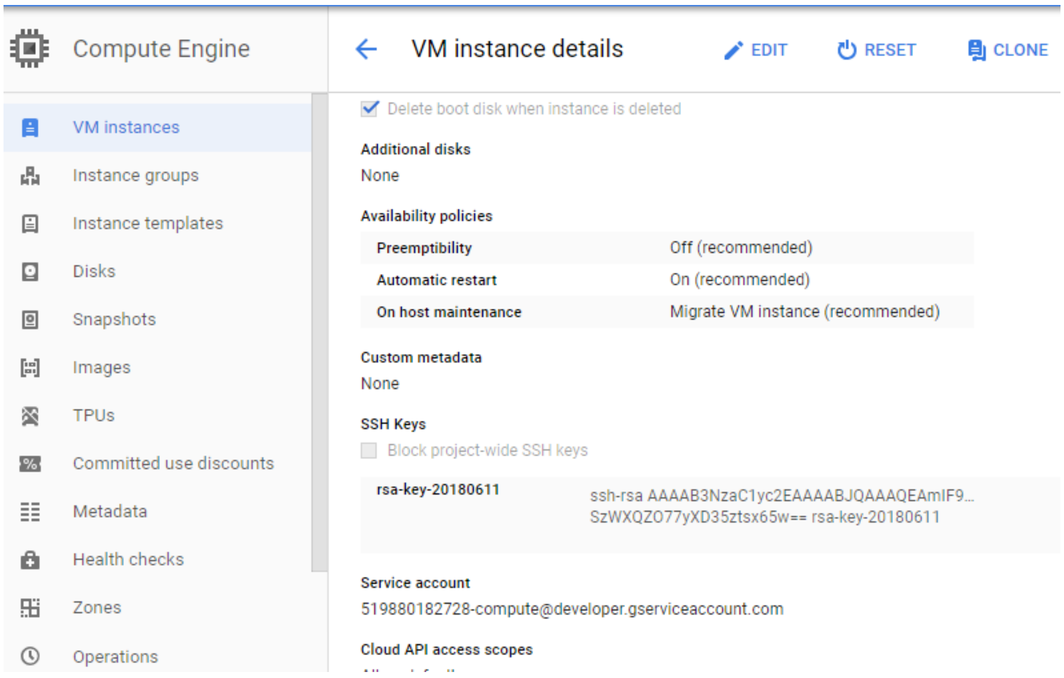
User: rsa-key-20180611

Key file: C:\Users\u4033\Desktop\CloudTest.pp

Background color: None

Comments:

For user, we need to refer back to the instance, and click through to instance details (by clicking on instance name). Scrolling down we can find the name of the SSH key we set up earlier:



✓ Delete boot disk when instance is deleted

Additional disks
None

Availability policies

Preemptibility	Off (recommended)
Automatic restart	On (recommended)
On host maintenance	Migrate VM instance (recommended)

Custom metadata
None

SSH Keys
Block project-wide SSH keys

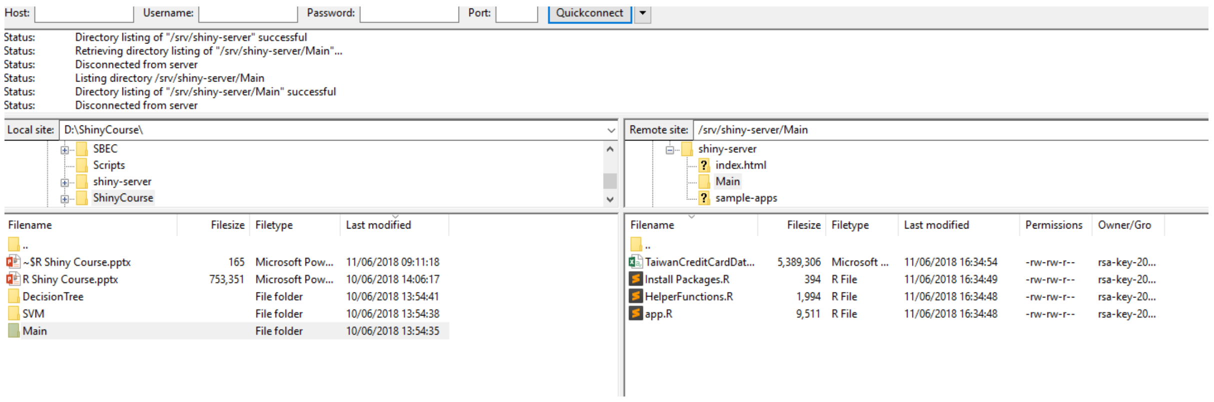
rsa-key-20180611	ssh-rsa AAAAB3NzaC1yc2EAAAABJQAAAQEAmIF9... SzWXQZO77yXD35ztsx65w== rsa-key-20180611

Service account
519880182728-compute@developer.gserviceaccount.com

Cloud API access scopes

Finally, we can select the local version of the SSH key created earlier as “Key File”.

Now we should be connected to the server so that the left hand side of our interface is the local machine we are working on, and the right hand side is the VM instance:



```
Status:     Directory listing of "/srv/shiny-server" successful
Status:     Retrieving directory listing of "/srv/shiny-server/Main"...
Status:     Disconnected from server
Status:     Listing directory /srv/shiny-server/Main
Status:     Directory listing of "/srv/shiny-server/Main" successful
Status:     Disconnected from server
```

Local site: D:\ShinyCourse\

- SBEC
- Scripts
- shiny-server
- ShinyCourse

Filename	Filesize	Filetype	Last modified
..			
~$R Shiny Course.pptx	165	Microsoft Pow...	11/06/2018 09:11:18
R Shiny Course.pptx	753,351	Microsoft Pow...	10/06/2018 14:06:17
DecisionTree		File folder	10/06/2018 13:54:41
SVM		File folder	10/06/2018 13:54:38
Main		File folder	10/06/2018 13:54:35

Remote site: /srv/shiny-server/Main

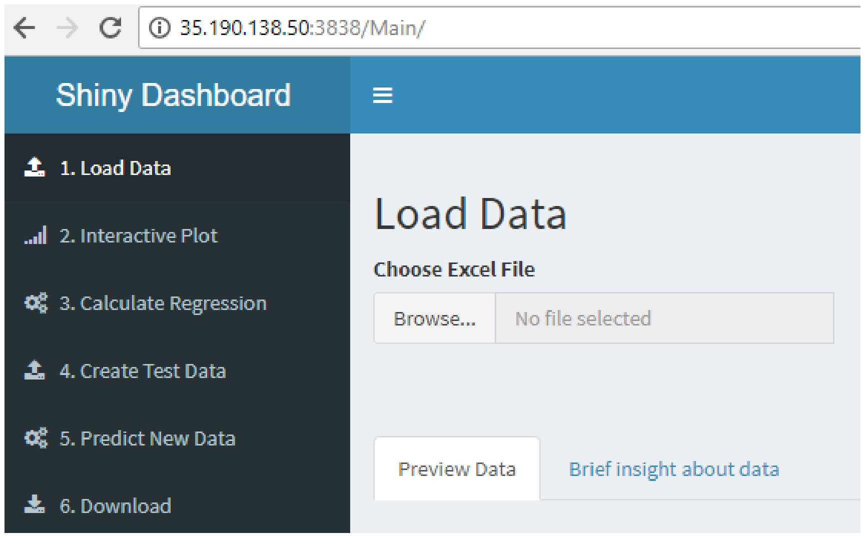
- shiny-server
  - index.html
  - Main
  - sample-apps

Filename	Filesize	Filetype	Last modified	Permissions	Owner/Gro
..					
TaiwanCreditCardDat...	5,389,306	Microsoft ...	11/06/2018 16:34:54	-rw-rw-r--	rsa-key-20...
Install Packages.R	394	R File	11/06/2018 16:34:49	-rw-rw-r--	rsa-key-20...
HelperFunctions.R	1,994	R File	11/06/2018 16:34:48	-rw-rw-r--	rsa-key-20...
app.R	9,511	R File	11/06/2018 16:34:48	-rw-rw-r--	rsa-key-20...

On the left we want to navigate to the folder where our app sits (in this case it is in a folder called “Main”). On the right, we navigate to the top level (by clicking on the .. symbol), and then into /srv/shiny-server. Here we can simply drag the folder (“Main”) from the left over to the right and the files will transfer. Now we should be ready to server our app! We need to change the permission back to secure, and then reboot Shiny Server:

```
sudo chmod 755 -R /srv/shiny-server

sudo systemctl restart shiny-server
```

Now we can navigate to our app by simply adding the folder name to the end of our URL (e.g. http://35.190.138.50:3838/Main/):



35.190.138.50:3838/Main/
Shiny Dashboard
1. Load Data
2. Interactive Plot
3. Calculate Regression
4. Create Test Data
5. Predict New Data
6. Download
Load Data
Choose Excel File
Browse...
No file selected
Preview Data
Brief insight about data