

Using VPC Infrastructure in the IBM Cloud

A Beginners Guide

James Belton
Global IT Architect for IBM Cloud Customer Success

May 2019

Table of Contents

Disclaimer	3
Introduction	4
Terminology.....	5
Region.....	5
Zone	5
VPC	5
Subnet.....	5
Classic Access	5
Public Gateways	6
Virtual Server Instance (VSI)	6
Floating IP Address.....	6
Access Control List (ACL).....	6
Security Groups	6
Load Balancer as a Service (LBaaS).....	6
VPN	7
SSH Key	7
VPC Architecture	8
Create a VPC	9
Step 1 - Create your VPN	9
Step 2 – Create an SSH Key	11
Step 3 - Add a Second Subnet	13
Step 4 – Create a Security Group	15
A quick recap.....	18
Step 5 – Create Your Webservers.....	19
Create the servers	19
Provide a Floating IP Address	22
Connect to the servers via SSH.....	23
Use Yum to update the servers	24
Install a webserver using Yum.	25
Create a Simple Webpage (and start the webserver)	25
Open the webpages from your Browser	26
Step 6 – Create and Configure a Load Balancer	27
Next Steps	33
Conclusion	34

Disclaimer

The information here is given in good faith and is intended to provide a helping hand in getting started with VPC services in IBM Cloud. It is intended to complement the official documentation and other resources that you can find at <https://cloud.ibm.com/vpc/overview> and is in no way a replacement.

Any services which you create following this guide are at your own cost and you are responsible for the security of the services which you provision.

Please use this document at your own risk. Any views and information expressed are my own.

Introduction

IBM introduced VPC Infrastructure to their cloud in early 2019, where it was initially available to early adopters only, going on to general availability for all on 31st May. This paper is a Beginner's Guide to VPC Infrastructure, to help you get started with this important new offering.

What is VPC Infrastructure? Well, think of VPC – or Virtual Private Cloud – as having your own private datacenter in a public cloud, in which you can create your own network subnets, create virtual machines to run on those subnets, along with storage and load balancers for your applications. Security is built into the VPC – external access to the VPC's subnets and resources are controlled through Access Control Lists and virtual servers can be further secured using Security Groups, which act like firewalls. Alternatively, you can choose to have no external access – in or out – at all.

The VPC is all set up in software, there is no hardware that you need to provision or configure. While there has always been the ability to create private subnets in IBM Cloud, this has only been achievable through provisioning the appropriate hardware, such as Vyatta gateways. While this is still a valid approach and is recommended for certain workloads, this route means it takes time to first provision the hardware and then more time to set up and configure it. There are costs associated with the routing hardware too. With VPC, everything is virtual – there is no hardware to provision and configure, which means getting started is both much quicker and lower in cost.

So, this paper is something of a beginner's guide to VPC. Hopefully, it will help you get started in understanding some of the terminology so that you can tell the difference between an ACL, a Floating IP address and a Security Group as well as get you on the way to creating your first VPC. Following this guide, you should end up with a VPC that has a couple of subnets, virtual servers running on them with a very simple web page, fronted by a load balancer, demonstrating how a highly available web application can be set up and running in about 30 minutes using VPC Infrastructure on IBM Cloud. Note that to complete this set up, you will need an IBM Cloud account, backed by a credit card or subscription and that there are costs associated with certain elements that you will create.

If you want to know more about the different account types and creating an IBM Cloud account, then I recommend that you watch a couple of videos that I have created and are available on YouTube:

<https://youtu.be/lv5mvTCY5bc>

<https://youtu.be/rSjuK78AuR0>

If you like these, then you may be interested in the rest of the series (The IBM Cloud Foundation Skill Series), which you can find here:

<https://www.youtube.com/playlist?list=PLmesOgYt3nKCfsXqx-A5k1bP7t146U4rz>

Hopefully you'll find this guide easy to use and follow. If you have any trouble with a particular section or spot any errors, please let me know. Note also that this is not a substitute for the official documentation, which you can always find at

<https://cloud.ibm.com/docs/infrastructure/vpc?topic=vpc-about>

Terminology

Let's start with a bit of terminology. Sprinkled through this document you're going to see references to certain 'components' which make up the VPC, so I thought it would be good to explain up front what these things are.

Region

A region is a geographical area covered by the IBM Cloud, into which you can deploy apps, services and other IBM Cloud resources. There are currently six regions (Dallas / us-south, Frankfurt / eu-de, Tokyo / jp-tok, London / eu-gb, Sydney / au-syd and Washington DC / us-east). At launch, three of these regions are available for VPC implementations (Dallas, Frankfurt and Tokyo), while the others will be rolled out over the following months. Each region is then made up of one or more zones.

Zone

A zone is a physical data centre within a region. These host components, such as compute, storage, networking as well as the power and cooling elements that they use to host and run applications. Zones are isolated from one another, ensuring no shared single point of failure. A region will have one or more zones within it.

VPC

This stands for Virtual Private Cloud and refers to the virtual environment that consists of all the other elements. You can create, by default, up to five VPCs in your IBM Cloud account, though this is a soft limit. A VPC is always attached to a single Region (a VPC cannot span regions) but you can create subnets in a VPC which reside in multiple zones within that region, providing high availability for your applications.

Subnet

A subnet is a segment of network with a specific IP address range to which resources such as virtual servers can be attached. A VPC can contain up to 15 subnets by default and each subnet is created into a particular zone. The zones in which a VPC's subnets can be created is determined by the region in which the VPC is created. For example, if a VPC is created in the Dallas Region, then subnets for that VPC can be created in the Dallas01, Dallas02 and Dallas03 zones.

A subnet can use a default range of IP addresses provided by IBM Cloud, or it can use a customer-determined range of IP addresses, commonly termed 'Bring Your Own IP' or BYOIP.

Classic Access

If you have resources that you have, or plan to create, using 'Classic Infrastructure' – that is virtual server instances outside of the VPC Infrastructure service or bare metal instances – that you want to access the services in your VPC, then by granting Classic Access, a software router will be created that provides this access through the internals of the IBM Cloud. We won't be setting this up as part of this paper.

Public Gateways

A public gateway is a virtual gateway router placed on the edge of a VPC subnet that provides access to the Internet. Placing a public gateway on a subnet is optional but is required if direct external access from a subnet is needed.

Virtual Server Instance (VSI)

A VSI is a virtual compute instance that runs inside the VPC, with a network interface that is attached to a particular subnet. These are ordered through the catalog.

Floating IP Address

A floating IP address is a public IP address that can be ordered and attached to a virtual server instance (VSI) from a pool, providing direct access to that VSI from the internet, assuming a public gateway exists on the subnet. Floating IP addresses are also assigned to the public gateway and any public load balancers that might be ordered in the VPC. You can attach and detach public IP addresses from VSIs but until they are returned to the pool, floating IP addresses are charged for. You can only use public IP addresses from this pool, you cannot bring your own public IP addresses.

Access Control List (ACL)

An ACL is a list of rules that govern access to a subnet. While a subnet must have only one ACL associated with it, an ACL can be associated to multiple subnets.

A subnet must have an ACL. The ACL itself is made up of rules that allow or deny inbound and outbound access to the subnet based on the source address, source ports, destination IP, destination port and the protocol being used. The default ACL is quite open, allowing any traffic in and any traffic out but this can be customized to specific needs.

Security Groups

A security group is another list of rules which control traffic to and from virtual server instances which are placed into the group and they act as virtual firewalls. A security group with no rules blocks everything, so you need to allow traffic by creating rules. Again, you can create rules based on protocol, source and ports.

Load Balancer as a Service (LBaaS)

A load balancer is a device which listens for connection requests from an external source and then passes on those connections to a specific host from a pool of hosts, based on a set of rules. For example, a website may be hosted on three servers to provide high availability as well as performance. Rather than connecting directly to one particular web server, a connection request hits the load balancer and the load balancer decides which web server should handle the request, thus spreading load for better performance. The load balancer is also able to detect if one or more of the web servers is down ('unhealthy') and in doing so, will not pass connections to that unhealthy server.

A load balancer will have a front end listener that listens for connections and will be associated to a pool of servers that can service that request.

VPN

A VPN or Virtual Private Network is a means to create a secure connection between two endpoints over a public network, such as the internet. The VPC service provides the ability to create a VPN between the VPC and a customer network, either via the Internet or through an existing Direct Link connection.

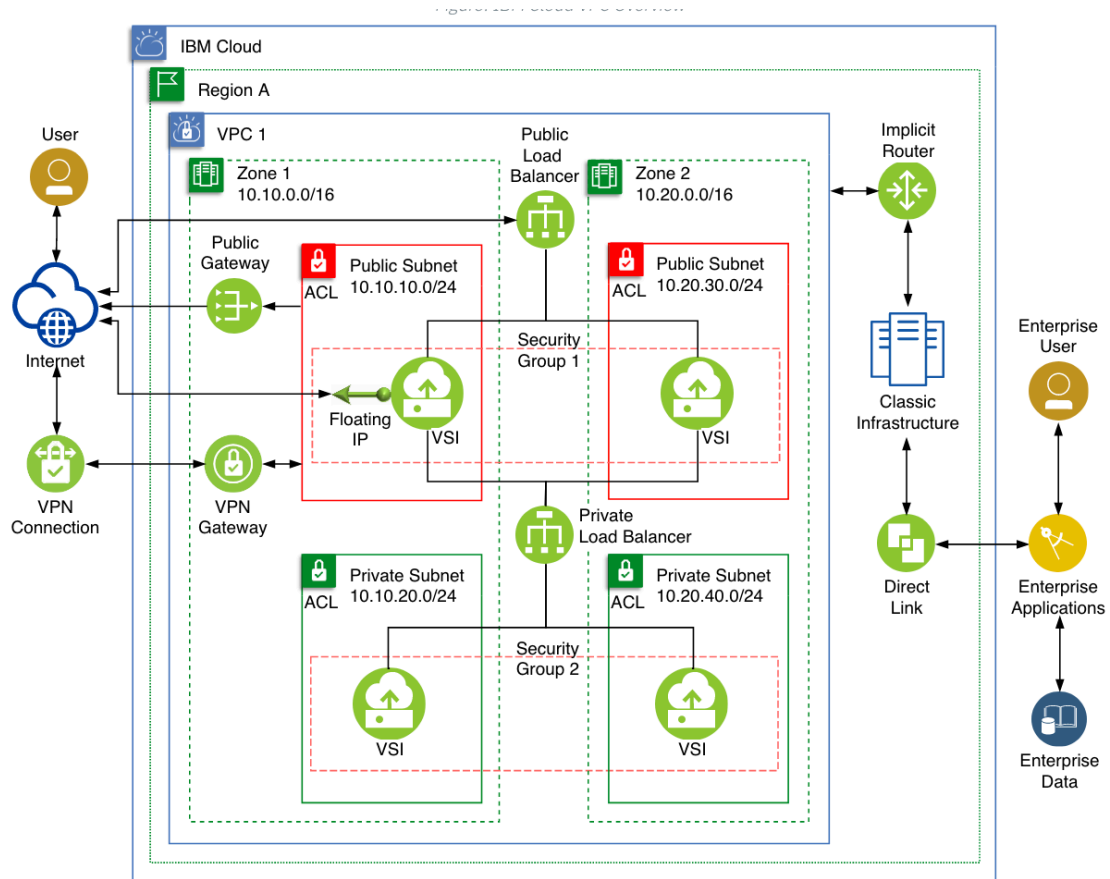
SSH Key

Connections to virtual server instances in a VPC is done via ssh (secure shell) and to increase security, such connections require an SSH Key pair. The public key will reside on the server, while the private key is needed on the computer that is trying to connect. Without the key, the connection will fail. Private keys need to be protected and should never be installed onto servers which can be publicly accessed as this represents a significant security risk.

VPC Architecture

The following diagram shows the architecture of a VPC and is sourced from the IBM Cloud VPC documentation which can be found at

<https://cloud.ibm.com/docs/infrastructure/vpc?topic=vpc-about>



This shows a VPC which has been created in Region A.

The VPC has a total of four subnets, two of which are public-facing (shown red) and two of which are private. These subnets are spread across two zones (data centres), with a public and private subnet located in each.

While each subnet has its own ACL, which determines access from the public gateway and VPNs, the virtual servers in the public subnets share a common Security Group, as do the virtual servers in the private subnets. These determine access rights via the load balancers. The public subnets have a public load balancer, which is accessible from the internet. This is then able to connect to the virtual server instances in the public subnets via their private IP addresses. In turn, these servers are able to access the private load balancer, which can then access the virtual servers in the private subnets.

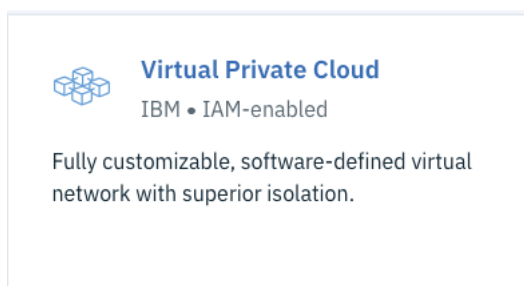
Create a VPC

In this section, I'm going to guide you through the steps of creating a VPC which will contain a couple of subnets, on which will run two virtual servers. These virtual servers will both run a webserver, serving a simple web page. Each of the webpages will be slightly different, for the purposes of showing the load balancer working, which we will also create.

So, let's get started. As previously mentioned, this assumes that you have an IBM Cloud account, are reasonably familiar with how it works and that you can create paid-for resources within that account. **Note, that by following these instructions you will have an environment that contains resources that you will need to pay for.** If you are trying to use these instructions before May 31st 2019, then please note you will not have access to the VPC service unless you have signed up and been accepted as an early adopter.

Step 1 - Create your VPN

From the catalog, click VPC Infrastructure from the left hand menu to bring up the VPC Infrastructure catalog offerings. You'll see six tiles, click the one marked 'Virtual Private Cloud'



The next screen is used to set up the VPC.

Name Provide a unique name for your VPC. As a guide, it's best to use a naming convention which provides meaning, so that you can quickly identify what the VPC is for. For example, the VPC might relate to a particular project or system, so reflect this in the name that you provide. For the purposes of this exercise, I have called mine **frank-webapp-demo**, where 'frank' refers to the location (Frankfurt) and 'webapp-demo' refers to the purpose of the VPC.

Resource Group Access to manage your VPC is determined through Identity and Access Management (IAM). You therefore need to specify the resource group that should be associated with the VPC and the access rights. For the purposes of this exercise, I am going to use the 'default' resource group. For more general information about IAM and Resource Groups watch this video: <https://youtu.be/w2AyDVS2SSM>

Tags These are optional but are useful for grouping resources together. More on tags in this video: https://youtu.be/Y6U_OG0L3eI

VPC default Access Control List Here, you are able to set your default ACL for your VPC. If this is your first VPC in the region, then this drop down list will show 'create new default (allow all)'.

Default Security Group A default security group will also be created for the VPC and you can choose here whether to allow ssh access as well as pings to virtual server instances by default. Keep these ticked for the time being.

Classic Access Ticking this box will create a router which will connect your VPC to services which you might have running or intend to create in Classic Infrastructure. For this example, leave this box unticked.

New Subnet for VPC This is where you create your first subnet on the VPC. This will also determine the region in which your VPC resides. The subnet will need a name (again, use a naming convention that allows you to easily identify the subnet and its purpose). Then choose the location (zone) of the subnet. At launch, you can choose between zones in the regions Dallas, Frankfurt and Tokyo. For this exercise, I am going to use Frankfurt 1 for the subnet, which gives rise to the subnet name I have chosen of 'frank1-webapp-sn'.

IP Range This is where I determine the IP Address range for my subnet. At this point, you can only choose from the '10.x.x.x' range shown by the wizard and those are the ranges we shall use in this exercise. However, when creating subsequent subnets, you can use your own ranges. For now, leave these values at the defaults.

Subnet Access Control List leave this set at the default of 'Use VPC Default'.

Public Gateway we need the VPC to be able to connect to the internet, so slide the switch to Attached.

You should now have a screen similar on the left hand side to the one shown below.

New virtual private cloud

Name

frank-webapp-demo

Resource group

default

Tags

Examples: env:dev, version-1

VPC default access control list

Create new default (Allow all)

Default security group

☒ Allow SSH

☒ Allow ping

Classic access

☐ Enable access to classic resource

New subnet for VPC

Name

frank1-webapp-sn

Location

Dallas
Dallas 1

Frankfurt
Frankfurt 1

Tokyo
Tokyo 1

IP range

10.243.0.0/24

Address prefix

10.243.0.0/18

Number of addresses

256

Address space

10.243.0.0 to 10.243.63.255

Subnet access control list

Use VPC default

Public gateway

Attaching a public gateway will allow all attached resources to communicate with the public Internet.

Detached

Attached

Click the ‘Create Virtual Private Cloud’ button to the right to continue.

You’ll then see a screen similar to this, indicating that your VPC and first subnet have been created and are running.

Virtual private cloud

REGIONS

Frankfurt

New virtual private cloud

Status	Name	Resource group	Subnets	Default ACL	Default Security Group	
Available	frank-webapp-demo	default	1	allow-all-network-acl-e96de5e4-d741-4f09...	knoll-endnote-skedaddle-scouting-tremor	...

Data updated 16 seconds ago

Step 2 – Create an SSH Key

You’ll need an SSH key to be able to ssh and connect to your virtual server instances. There’s two steps to this.

First you need to create your key pair, if you don’t already have one. The public part of the key goes onto the server that you are trying to connect to, while the private part stays on the machine that you are connecting from (e.g. your PC or Mac). If you already have a key pair, then you’ll find it in your home directory in the hidden folder .ssh. The key itself will be called something like id_rsa.pub.

In this example, I'm going to create a new key called **frank-sshkey** using the `ssh-keygen` command on my Mac computer:

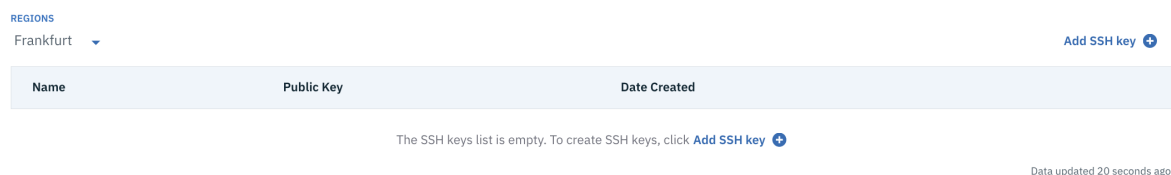
```
jamesbelton — root@webserver01-dallas1:~ — -bash — 108x26
[James-MBP-2:~ jamesbelton$ pwd
/Users/jamesbelton
[James-MBP-2:~ jamesbelton$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/jamesbelton/.ssh/id_rsa): /Users/jamesbelton/.ssh/frank-sshkey
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/jamesbelton/.ssh/frank-sshkey.
Your public key has been saved in /Users/jamesbelton/.ssh/frank-sshkey.pub.
The key fingerprint is:
SHA256: xMa2BvobDJ0CA8DHfELtFKJZ0qWLG5YKFp0lh+3MPV8 jamesbelton@James-MBP-2.lan
The key's randomart image is:
+----[RSA 2048]-----+
| =0*B...          |
| +^=..+ 0         |
| .#+0= . *        |
| = 00.+ * .       |
| .. ++00 SE       |
|      =0..        |
|      +.          |
|      0           |
|      .           |
|      .           |
+-----[SHA256]-----+
James-MBP-2:~ jamesbelton$
```

You can see here that I've created the key with a passphrase (password) so that when I use the key, I get added protection because I will also need to enter a password. By running the command above, I have two new files on my computer:

/Users/jamesbelton/.ssh/frank-sshkey (my private key) and;
/Users/jamesbelton/.ssh/frank-sshkey.pub (my public key)

Next, we need to associate that with the VPC. From the VPC screen above (or navigate to <https://cloud.ibm.com/vpc/network/vpcs> if you have closed your browser), click SSH Keys in the left hand menu, under Compute.

SSH keys for VPC



You should see a screen similar to that above, assuming you have no keys already added. To add a new key, click the Add SSH Key link.

This is a fairly simple screen to complete. First, give your key a name. Again, this should be meaningful. I'm going to use the name **frank-sshkey** as it's the key that I'm going to use in Frankfurt. Note that for added security, think about using different SSH keys for different environments / server types but in this instance, because it's a simple demo, we'll just use the one.

As my VPC is in Frankfurt, I'm going to select Frankfurt for my region.

The next box, headed Public Key is where you need to paste in the contents of your public key file. In my case, using the steps above, my public key file is on my Mac computer and is called:

`/Users/jamesbelton/.ssh/frank-sshkey.pub`

I can use a text editor (e.g. vi or atom) to open the file or similarly I can use 'cat' on the command line to get the contents, which will look something like this:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADMSPBAAABAQDVnCFK1AzbXrHKLsH8zQ7mdv1bvROWj5i6kNKv
kfC0I5KGvDKMBBKisix0or6UBVDGxGXTrbGXCaMc6ABCC7NMF0KkNFgg2PYINvakYrj5
kY6FBu0cQjWnv0vNai+JjrG1m58bAeUrWC4lJ6G3tc40LcjrKjb13ebJMpKkieD30YUY
DXu/1f2LB2riwc1iPwXzjyosk91+V4JBRmpf4ZMwYFUwV+lgCAFpAz0+0l7Q2QQ0s0q2
sOPNeZlY0R0VkhS/oEcmEpG53DWMkw+uYGPca7CvRx2JB2f7+ElNa70Mjk5xT7JfbkJ
6DUmnDvg7vP8eG8yXAM6XrVxN9+T06gb jamesbelton@Jamess-MBP-2.lan
```

Copy and paste this into the appropriate box. You should end up with a screen like this:

Add SSH key

Name

frank-sshkey

Region

Dallas Frankfurt Tokyo

Public key [How do I get a public key?](#)

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADMSPBAAABAQDVnCFK1AzbXrHKLsH8zQ7mdv1bvROWj5i6kNKv
kfC0I5KGvDKMBBKisix0or6UBVDGxGXTrbGXCaMc6ABCC7NMF0KkNFgg2PYINvakYrj5
kY6FBu0cQjWnv0vNai+JjrG1m58bAeUrWC4lJ6G3tc40LcjrKjb13ebJMpKkieD30YUY
DXu/1f2LB2riwc1iPwXzjyosk91+V4JBRmpf4ZMwYFUwV+lgCAFpAz0+0l7Q2QQ0s0q2
sOPNeZlY0R0VkhS/oEcmEpG53DWMkw+uYGPca7CvRx2JB2f7+ElNa70Mjk5xT7JfbkJ
6DUmnDvg7vP8eG8yXAM6XrVxN9+T06gb jamesbelton@Jamess-MBP-2.lan
```

API </> Cancel Add SSH key

Click Add SSH Key and you'll then return to a screen with the SSH Key listed.

Now click Subnets from the left hand menu.

Step 3 - Add a Second Subnet

You should be looking at a list of the subnets in your account at this stage – if not, then navigate to <https://cloud.ibm.com/vpc/network/subnets>. You should only see one subnet

right now – that being the subnet you created when you created your VPN. It's now time to create a second one. Click New Subnet.

[New subnet](#) 

You'll see a screen that looks pretty familiar from the Subnet section of the create VPC screen. Give the subnet a name, using your naming convention. For this example, I'm going to call mine **frank2-webapp-sn** – I'm using 'frank2' as this is the location / zone in which I am going to create my subnet.

Make sure that the Virtual Private Cloud dropdown shows the correct VPC and in Location, choose a zone which is different to the zone chosen for the first subnet. Change the Public Gateway to attached and leave the other settings as they are. You should have something like this:

New subnet for VPC

Name

frank2-webapp-sn

Virtual private cloud

frank-webapp-demo

Location



IP range

10.243.64.0/24

Address prefix

10.243.64.0/18

Number of addresses

256

Address space 10.243.64.0 to 10.243.127.255

Subnet access control list

VPC Default(allow-all-network-acl-e96de5e4-d741-4f09-9175-387e1b99fc9d)

Public gateway

Attaching a public gateway will allow all attached resources to communicate with the public Internet.

Detached  Attached

The reason for creating this second subnet is so that we can then create virtual machines in a second zone, meaning that we are building high availability into our application.

When you're ready, click Create Subnet to the right hand side of the screen.

The second subnet should only take 30 seconds or so to provision, and you should end up looking at a screen like this:

Subnets for VPC

REGIONS
Frankfurt ▾ New subnet +

Status	Subnets	Virtual Private Cloud	Location	IP Range	Public Gateway	
● Available	frank1-webapp-sn	frank-webapp-demo	Frankfurt 1	10.243.0.0/24	158.177.184.97	...
● Available	frank2-webapp-sn	frank-webapp-demo	Frankfurt 2	10.243.64.0/24	161.156.80.91	...

Items per page: 10 ▾ | 1-2 items < >

Data updated 15 seconds ago

So this tells me that I have two subnets, one in Frankfurt 1 and the other in Frankfurt 2. The first subnet has an IP address range of 10.243.0.0/24 and the second has an address range of 10.243.64.0/24. Both are connected to a public gateway and so have internet access.

Step 4 – Create a Security Group

We're now going to create a security group. The purpose of this is to secure the servers that we are going to create in the next step. At the moment, any traffic is allowed into the VPC subnets, but we want to be more cautious with the traffic we allow into our servers.

Click Security Groups on the left hand menu or navigate to <https://cloud.ibm.com/vpc/network/securityGroups>. You'll see listed the default security group (it'll have a pretty random looking name – mine at this stage is called knoll-endnote-scouting-skedaddle-tremor) and you can either repurpose this or create a new one. I'm going to create a new one.

Click New Security Group.

Give the security group a name. Like everything else, use a meaningful name that fits a naming convention. Since this is going to be used on my Frankfurt web servers, I'm going to call mine **frank-webserver-sg**.

I'm going to add my first inbound rule and that will be so that I can ping a server. To do this, I click 'Add Rule' above the Inbound Rule list. Then I select ICMP from the list of protocols and leave Value and Source Type as Any. Then click Save.

Add Inbound Rule
×

Protocol

ICMP ▾

Value
☒ Any ☐ Type and code

Source type
☒ Any ☐ IP address ☐ CIDR block ☐ Security group

I'm then going to add a second rule, this time so that I can ssh to any servers (though note that to do so, a server will need a floating IP address assigned). Since I don't want any old computer to be able to ssh to my hosts I'm going to restrict access to just the IP address of my mac (type "what's my IP address" into Google if you're not sure what your public IP address is – note that if your computer doesn't have a fixed IP address, this can change after you reboot).

Click 'Add Rule' again. Ensure that Protocol is set to TCP and the Port Range is set to Port Min – 22 and Port Max – 22, Port 22 being the port that servers listen on for SSH connections.

For source type, select IP Address and then fill in your IP address in the space below. You should have something like this:

Add Inbound Rule ×

Protocol

TCP ▼

Port

☐ Any ☒ Port range

Port min	Port max
22	22

Source type

☐ Any ☒ IP address ☐ CIDR block ☐ Security group

91.123.123.59

Then click Save.

Next, we need to create rule for our web traffic, so create a rule for incoming TCP on port 80 and another rule for TCP on port 443.

Both will look similar to this:

Add Inbound Rule



Protocol

TCP

Port

☐ Any ☒ Port range

Port min

80

Port max

80

Source type

☒ Any ☐ IP address ☐ CIDR block ☐ Security group

Lastly, we need to create an outbound rule. Right now, traffic can come in but all traffic outbound is blocked. Let's create a single 'allow all' outbound rule. This time, click the Create Rule link next to the Outbound Rules part of the table.

Set protocol to ALL and Source Type to 'any' as follows and then click OK.

Add Outbound Rule



Protocol

ALL

Source type

☒ Any ☐ IP address ☐ CIDR block ☐ Security group

You should now have a screen that looks like the following:

Name

Virtual private cloud

frank-webserver-sg

frank-webapp-demo

Rules

Inbound rules

Add rule +

Protocol	Source type	Source	Value	
TCP	Any	0.0.0.0/0	Ports 443-443	⊖
TCP	Any	0.0.0.0/0	Ports 80-80	⊖
TCP	IP address	91.125.115.59	Ports 22-22	⊖
ICMP	Any	0.0.0.0/0	Type: Any, Code: Any	⊖

Items per page: 10 | 1-4 items

< >

Outbound rules

Add rule +

Protocol	Destination type	Destination	Value	
ALL	Any	0.0.0.0/0	—	⊖

Items per page: 10 | 1-1 items

< >

Click Create Security Group.

A quick recap

At this point, we should have:

- A VPC
- Two Subnets with public gateways
- An Access Control List (created by default)
- An SSH Key
- A Security Group

Let's have a quick look at the Access Control List, so that we understand it.

On the left hand menu, click Access Control Lists or click through to <https://cloud.ibm.com/vpc/network/acl>. You should see one ACL listed. Click on the link that forms the name of the ACL.

You should see something similar to the following screenshot:

All access control lists for VPC



allow-all-network-acl-e96de...

Id: 7c7ecc73-c5ef-4dce-878b-9ba36e9eb8c6

[View docs](#)

Inbound rules

[Add rule](#)

Rule Priority	Allow/Deny	Protocol	Source	Destination	Value	
1	Allow	ALL	Any	Any	—	...

Outbound rules

[Add rule](#)

Rule Priority	Allow/Deny	Protocol	Source	Destination	Value	
1	Allow	ALL	Any	Any	—	...

Basically, this allows All protocols from any source access to any destination to come into the attached subnets. Similarly, it allows all outbound traffic. In a real-life scenario, we would want to close this down a bit but for this exercise, we'll keep these settings and use our Security Groups to protect our servers.

Speaking of servers, let's create those next.

Step 5 – Create Your Webservers

In this step, we're going to:

- create a couple of virtual servers
- give them both Floating IP Addresses
- connect to them via ssh
- update them via yum
- install a web server
- create a simple webpage
- make sure we can connect to the webpage

Create the servers

If you're not already in the VPC section of IBM Cloud, then click through to <https://cloud.ibm.com/vpc/network/vpcs> and then click Virtual Server Instances on the left hand menu.

At this point, the list of virtual servers will be blank – click 'New Instance' and the create Virtual Server Instance form will be displayed. We're going to create two pretty standard virtual machines for this workshop – one of which will be on one of our subnets and the other will be on our second subnet. By doing this, we can demonstrate that the website we build is highly available, when accessed via a load balancer.

We need to give our first server a name. Again, use a naming convention that gives meaning. In this example, I'm going to use the name **webserver1-frank1** since this tells me what server it is (webserver number 1) and where it is (Frankfurt1).

Make sure the name of your VPC is shown in the Virtual Private Cloud drop-down and then choose your location. Here, I'm choosing Frankfurt1 because that's where my first subnet is located (note that if I were to choose Frankfurt3, I would get a message in the Network

Interfaces section further down the screen, informing me that I don't have a subnet there and that to continue, I'd need to create one).

Next we need to choose an operating system image. I'm going to choose CentOS.

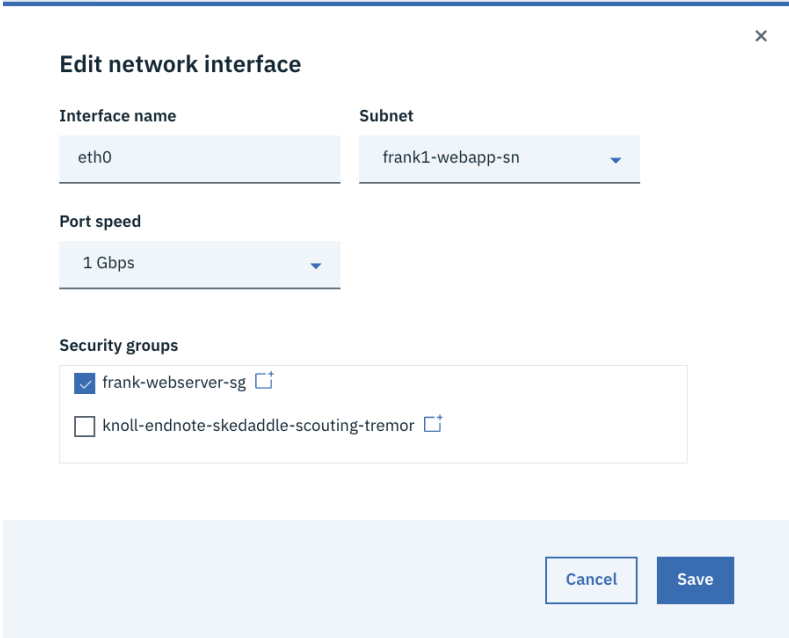
Then we need to select the VSI capacity. Since this is just a demo, I'm going to choose the standard Balanced profile (2vCPUs and 8GB RAM).

Next, we need to select our SSH Key. Select the name of the SSH Key that you created in Step 2 above. **It is important that you do not delete your private key** as once the server has been created, you will not be able to connect to it using ssh without the private key.

We'll leave the user data field blank but in a later update of this paper, I'll detail how to use this to run startup scripts on the server, which is a pretty useful thing to do.

We'll leave the boot volume as is. Note that this is encrypted and will be automatically deleted when the VSI is deleted. We'll also leave the Attached Storage as is (i.e. no further attached storage) but you have the option with attached storage to either delete along with the VSI or leave it in place, so that the same storage can be attached to another machine in the future.

For the Network Interface, we need to make a change to set the correct security group. Click the pencil icon to the right of the Network Interface table. In the screen that pops up, uncheck the name of the default security group and check the name of the Security Group that you created in Step 4. It should then look something like this:



The screenshot shows a modal window titled "Edit network interface" with a close button (X) in the top right corner. The window contains the following fields and controls:

- Interface name:** A text input field containing "eth0".
- Subnet:** A dropdown menu showing "frank1-webapp-sn".
- Port speed:** A dropdown menu showing "1 Gbps".
- Security groups:** A list of two security groups with checkboxes:
 - ☒ frank-webserver-sg
 - ☐ knoll-endnote-skedaddle-scouting-tremor
- Buttons:** "Cancel" and "Save" buttons at the bottom right.

This will ensure that you can connect to the server from your local machine and that HTTP and HTTP traffic can access the server. Click Save to save the change to the network interface.

You can now create the virtual server instance by pressing the 'Create' button on the right hand side. Note that doing this will incur charges, details of which are above the button, for example:

Order summary United States ▾

Virtual server instance	\$0.11/hr
2 vCPUs	
8 GB RAM	
CentOS	
1 Gbps	
Boot volume	\$0.02/hr
100 GB	
Network interface	provided

Subtotal	\$93.78
Sustained usage discount ⓘ	\$6.01
Estimated monthly	\$87.77

[Create virtual server instance](#)

[View docs](#) 📄

[Get sample API call](#) </>

Note that charges are hourly, so if you complete this workshop and then delete the instance, your charges will be minimalized.

After pressing the create button, you'll be taken back to the Virtual Server Instances screen, which will show the VSI you just created in a pending state:

Virtual server instances for VPC

REGIONS
Frankfurt ▾ [New instance](#) ⓘ

Status	Name	Virtual Private Cloud	Private IP	Floating IP	
● Pending	webserver1-frank1	frank-webapp-demo	10.243.0.9	–	...

Items per page: 10 ▾ | 1-1 items Data updated 11 seconds ago

We then need to create the second server instance. Press the New Instance button again.

We now create an identical server – **however** – this time, I am going to name my webserver webserver1-frank2 and place it in Frankfurt2, where it will sit on my second subnet. Check that you are using the same SSH Key and remember to place this server into the correct Security Group in the Network Interface section.

Press create again (noting the charges) and you'll be returned to the Virtual Server Instances screen again, which should now show two servers:

Virtual server instances for VPC

REGIONS
Frankfurt ▼ New instance +

Status	Name	Virtual Private Cloud	Private IP	Floating IP	
● Powered On	webserver1-frank1	frank-webapp-demo	10.243.0.9	—	...
● Pending	webserver1-frank2	frank-webapp-demo	10.243.64.8	—	...

Items per page: 10 ▼ | 1-2 items < >

Data updated 8 seconds ago

Provide a Floating IP Address

So that we can connect to each of the servers, we're going to give them both a Floating IP address. Without this, we cannot connect to either server directly – though there are other ways which are outside the current scope of this paper. Again, subsequent updates will look at this. Note that Floating IP Addresses do attract a cost but we'll use them sparingly and remove them as soon as possible.

Assigning a floating IP Address to a server is simple. From the Virtual Server Instances screen (<https://cloud.ibm.com/vpc/compute/vs>), click the link / name of the first server. Under the Network Instances section to the right, click 'Reserve' under the Floating IP heading in the table.


Network interfaces i

Interface	Subnet Name	Private IP	Floating IP	Security Groups
eth0	frank1-webapp-sn	10.243.0.9	Reserve +	frank-webserver-sg

A few seconds later, a public IP address will appear.

Click the 'All Instances for VPC' link at the top left of the screen (above the server name) to return to your list of two servers and repeat for the second server:

All instances for VPC



webserver1-frank1
● Powered On

Frankfurt 1

Instance details

Your list of servers should now look similar to this, complete with Floating IP addresses (note, if you don't see any addresses, try refreshing your browser).

Virtual server instances for VPC

REGIONS
Frankfurt ▼ New inst

Status	Name	Virtual Private Cloud	Private IP	Floating IP	
● Powered On	webserver1-frank1	frank-webapp-demo	10.243.0.9	158.177.184.71	
● Powered On	webserver1-frank2	frank-webapp-demo	10.243.64.8	161.156.80.152	

Items per page: 10 ▼ | 1-2 items < >

We're now ready to connect to the servers via SSH.

Connect to the servers via SSH

To connect to the first server, first open a command prompt / terminal window. If you are using Windows, then you may need to install ssh (if so, try this guide:

https://docs.microsoft.com/en-us/windows-server/administration/openssh/openssh_install_firstuse). If you are using a Mac or Linux machine, then you'll already have ssh installed.

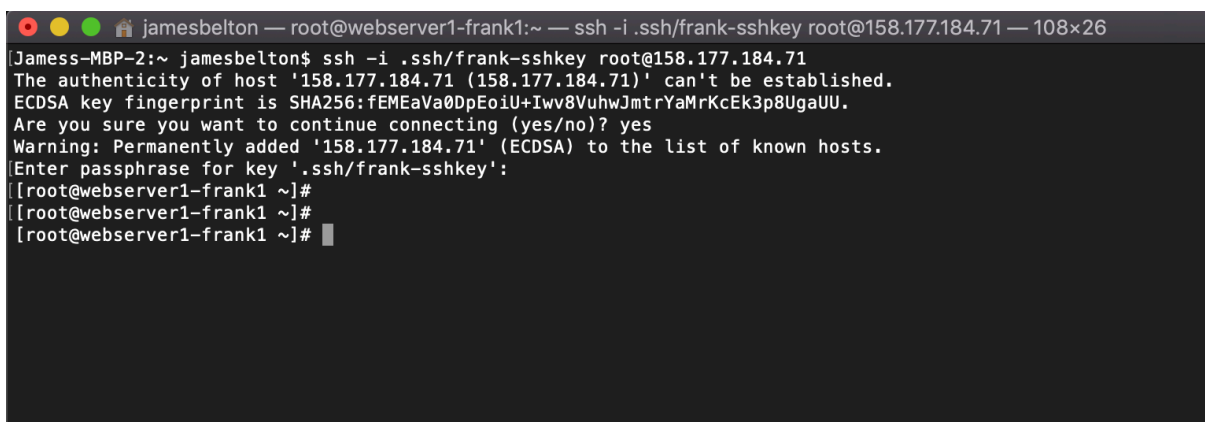
From the command prompt, type:

```
ssh -i <<private-key>> root@<<server-ip>>
```

Where <<private-key>> is the path and filename of your private key file and <<server-ip>> is the Floating IP address of your server. For example:

```
ssh -i .ssh/frank-sshkey root@158.177.184.71
```

You may see a message about the authenticity of the host not being established – this is normal, reply 'yes' to the prompt. Note also that you will need to supply the password for the ssh key that you used when you created the key.



```
jamesbelton — root@webserver1-frank1:~ — ssh -i .ssh/frank-sshkey root@158.177.184.71 — 108x26
[James-MBP-2:~ jamesbelton$ ssh -i .ssh/frank-sshkey root@158.177.184.71
The authenticity of host '158.177.184.71 (158.177.184.71)' can't be established.
ECDSA key fingerprint is SHA256:fEMEaVa0DpEoiU+Iwv8VuhwJmtrYaMrKcEk3p8UgaUU.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '158.177.184.71' (ECDSA) to the list of known hosts.
[Enter passphrase for key '.ssh/frank-sshkey':
[root@webserver1-frank1 ~]#
[root@webserver1-frank1 ~]#
[root@webserver1-frank1 ~]#
```

The process of connecting will look similar to the screenshot above.

If you are unable to connect, check that your IP address hasn't changed (this may have happened if you have rebooted your local machine after you set up the security group in Step 4 – basically, make sure that your IP matches that in the security group rule for access to port 22.

Open another command prompt / terminal window and connect to the other server.

Note that you should also be able to ping the servers from your host on the Floating IP address. When connected to one of the servers, you should also be able to ping the other on its private IP address (it's '10.x.x.x' address) but you won't be able to ssh from one to the

other. This is because the ACL rules allow all traffic to and from the subnets but the Security Group rules only allow traffic to the virtual servers on port 22 from your host (IP address).

Use Yum to update the servers

It's good practice to make sure the servers have the latest kernel patches and so in installed. To do this, type 'yum update -y' at the command prompt when connected to each server:

```
jamesbelton — root@webserver1-frank1:~ — ssh -i .ssh/frank-sshkey root@158.177.184.71 — 108x26
James-MBP-2:~ jamesbelton$ ssh -i .ssh/frank-sshkey root@158.177.184.71
The authenticity of host '158.177.184.71 (158.177.184.71)' can't be established.
ECDSA key fingerprint is SHA256:fEMeVa0DpEoiU+Iwv8VuhwJmtrYaMrKcEk3p8UgaUU.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '158.177.184.71' (ECDSA) to the list of known hosts.
[Enter passphrase for key '.ssh/frank-sshkey':
[root@webserver1-frank1 ~]#
[root@webserver1-frank1 ~]#
[root@webserver1-frank1 ~]# yum update -y
```

```
jamesbelton — root@webserver1-frank1:~ — ssh -i .ssh/frank-sshkey root@158.177.184.71 — 108x26
James-MBP-2:~ jamesbelton$ ssh -i .ssh/frank-sshkey root@158.177.184.71
The authenticity of host '158.177.184.71 (158.177.184.71)' can't be established.
ECDSA key fingerprint is SHA256:fEMeVa0DpEoiU+Iwv8VuhwJmtrYaMrKcEk3p8UgaUU.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '158.177.184.71' (ECDSA) to the list of known hosts.
Enter passphrase for key '.ssh/frank-sshkey':
[root@webserver1-frank1 ~]#
[root@webserver1-frank1 ~]#
[root@webserver1-frank1 ~]# yum update -y
Loaded plugins: fastestmirror
Determining fastest mirrors
base                                     | 3.6 kB  00:00:00
extras                                 | 3.4 kB  00:00:00
updates                                | 3.4 kB  00:00:00
(1/4): base/7/x86_64/group_gz          | 166 kB  00:00:05
(2/4): extras/7/x86_64/primary_db      | 200 kB  00:00:05
(3/4): base/7/x86_64/primary_db       | 6.0 MB  00:00:05
(4/4): updates/7/x86_64/primary_db    | 5.0 MB  00:00:05
```

The update will run and probably take a few minutes to complete.

```
jamesbelton — root@webserver1-frank1:~ — ssh -i .ssh/frank-sshkey root@158.177.184.71 — 108x26
nss-util.x86_64 0:3.36.0-1.1.el7_6          openssl.x86_64 0:2.4.44-21.el7_6
openssl.x86_64 1:1.0.2k-16.el7_6.1         openssl-libs.x86_64 1:1.0.2k-16.el7_6.1
polkitcoreutils.x86_64 0:2.5-29.el7_6.1     polkit.x86_64 0:0.112-18.el7_6.1
postfix.x86_64 2:2.10.1-7.el7               procps-ng.x86_64 0:3.3.10-23.el7
python.x86_64 0:2.7.5-77.el7_6              python-firewall.noarch 0:0.5.3-5.el7
python-libs.x86_64 0:2.7.5-77.el7_6         python-linux-procfs.noarch 0:0.4.9-4.el7
python-perf.x86_64 0:3.10.0-957.12.2.el7     python-urlgrabber.noarch 0:3.10-9.el7
rpm.x86_64 0:4.11.3-35.el7                  rpm-build-libs.x86_64 0:4.11.3-35.el7
rpm-libs.x86_64 0:4.11.3-35.el7             rpm-python.x86_64 0:4.11.3-35.el7
rsyslog.x86_64 0:8.24.0-34.el7               selinux-policy.noarch 0:3.13.1-229.el7_6.12
selinux-policy-targeted.noarch 0:3.13.1-229.el7_6.12
shadow-utils.x86_64 2:4.1.5.1-25.el7_6.1    setup.noarch 0:2.8.71-10.el7
systemd.x86_64 0:219-62.el7_6.6             sudo.x86_64 0:1.8.23-3.el7
systemd-libs.x86_64 0:219-62.el7_6.6        systemd-python.x86_64 0:219-62.el7_6.6
systemd-sysv.x86_64 0:219-62.el7_6.6        tar.x86_64 2:1.26-35.el7
teamd.x86_64 0:1.27-5.el7                    tuned.noarch 0:2.10.0-6.el7_6.3
tzdata.noarch 0:2019a-1.el7                  util-linux.x86_64 0:2.23.2-59.el7_6.1
vim-minimal.x86_64 2:7.4.160-5.el7          wget.x86_64 0:1.14-18.el7_6.1
wpa_supplicant.x86_64 1:2.6-12.el7          xfsprogs.x86_64 0:4.5.0-19.el7_6
yum.noarch 0:3.4.3-161.el7.centos            yum-plugin-fastestmirror.noarch 0:1.1.31-50.el7
zlib.x86_64 0:1.2.7-18.el7

Replaced:
grub2.x86_64 1:2.02-0.65.el7.centos.2      grub2-tools.x86_64 1:2.02-0.65.el7.centos.2

Complete!
[root@webserver1-frank1 ~]#
```


Install a webserver using Yum.

Next, we're going to install the httpd web server. This is really simple, all you need to do is type `yum install -y httpd` into the terminal window when connected to each of the servers:

```
jamesbelton — root@webserver1-frank1:~ — ssh -i .ssh/frank-sshkey root@158.177.184.71 — 108x26

[root@webserver1-frank1 ~]# yum install -y httpd
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
Resolving Dependencies
--> Running transaction check
--> Package httpd.x86_64 0:2.4.6-89.el7.centos will be installed
--> Processing Dependency: httpd-tools = 2.4.6-89.el7.centos for package: httpd-2.4.6-89.el7.centos.x86_64
--> Processing Dependency: /etc/mime.types for package: httpd-2.4.6-89.el7.centos.x86_64
--> Processing Dependency: libaprutil-1.so.0()(64bit) for package: httpd-2.4.6-89.el7.centos.x86_64
--> Processing Dependency: libapr-1.so.0()(64bit) for package: httpd-2.4.6-89.el7.centos.x86_64
--> Running transaction check
--> Package apr.x86_64 0:1.4.8-3.el7_4.1 will be installed
--> Package apr-util.x86_64 0:1.5.2-6.el7 will be installed
--> Package httpd-tools.x86_64 0:2.4.6-89.el7.centos will be installed
--> Package mailcap.noarch 0:2.1.41-2.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package                Arch          Version              Repository           Size
=====
Installing:
httpd                  x86_64        2.4.6-89.el7.centos  updates              2.7 M
Installing for dependencies:
=====
```

```
jamesbelton — root@webserver1-frank1:~ — ssh -i .ssh/frank-sshkey root@158.177.184.71 — 108x26

Total                                                                527 kB/s | 3.0 MB  00:00:05
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : apr-1.4.8-3.el7_4.1.x86_64                          1/5
  Installing : apr-util-1.5.2-6.el7.x86_64                        2/5
  Installing : httpd-tools-2.4.6-89.el7.centos.x86_64             3/5
  Installing : mailcap-2.1.41-2.el7.noarch                        4/5
  Installing : httpd-2.4.6-89.el7.centos.x86_64                  5/5
  Verifying  : httpd-tools-2.4.6-89.el7.centos.x86_64            1/5
  Verifying  : mailcap-2.1.41-2.el7.noarch                        2/5
  Verifying  : httpd-2.4.6-89.el7.centos.x86_64                  3/5
  Verifying  : apr-1.4.8-3.el7_4.1.x86_64                        4/5
  Verifying  : apr-util-1.5.2-6.el7.x86_64                       5/5

Installed:
httpd.x86_64 0:2.4.6-89.el7.centos

Dependency Installed:
apr.x86_64 0:1.4.8-3.el7_4.1    apr-util.x86_64 0:1.5.2-6.el7    httpd-tools.x86_64 0:2.4.6-89.el7.centos
mailcap.noarch 0:2.1.41-2.el7

Complete!
[root@webserver1-frank1 ~]#
```

Create a Simple Webpage (and start the webserver)

We now have a web server but we need some content to serve, so we're going to create a very simple webpage on each of the servers. The idea here is to replicate a website that would be installed over two web servers, to show that the website is highly available. Normally, the website would therefore be identical, but in this case, we're going to make them slightly different so that you can see how the load balancer is working in the next step.

To create your webpages:

1. cd into the directory `/var/www/html`
2. using an editor (e.g. vi or vim) create a file called `index.html`
3. on the first server, use the following code in your `index.html` file:

```
<html>
<head>
  <title> webserver1-frank1 </title>
</head>
```

```
<body>
  <h1> This is webserver1-frank1 </h1>
</body>
</html>
```

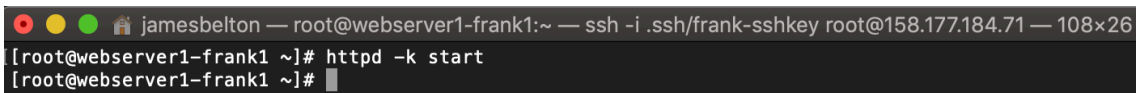
4. on the second server, use the following code in your index.html file:

```
<html>
<head>
  <title> webserver1-frank2 </title>
</head>
<body>
  <h1> This is webserver1-frank2 </h1>
</body>
</html>
```

Note that the two files are essentially the same, only the names of the webserver are changing.

5. Save the files.

Next, we need to start the webserver on each server. To do this, type `httpd -k start`

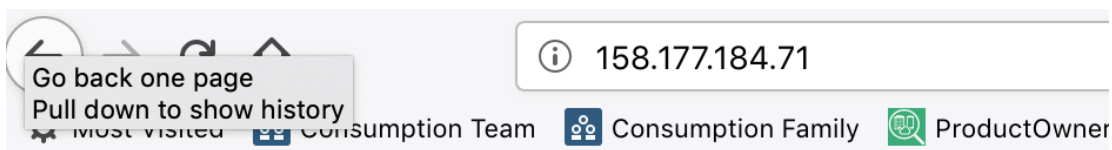


```
jamesbelton — root@webserver1-frank1:~ — ssh -i .ssh/frank-sshkey root@158.177.184.71 — 108x26
[root@webserver1-frank1 ~]# httpd -k start
[root@webserver1-frank1 ~]#
```

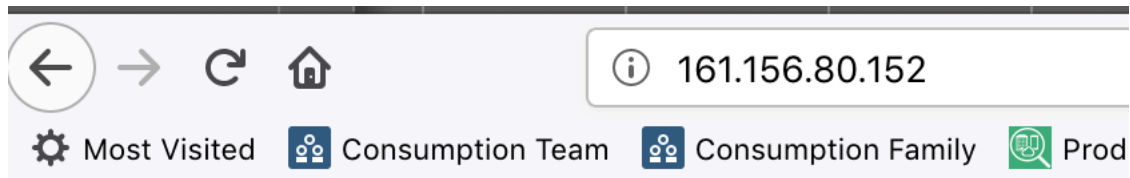
And that's the webserver and website set up. Next we need to make sure we can see the pages.

[Open the webpages from your Browser](#)

Open your favourite browser and simply navigate to the Floating IP address of your servers – use a separate browser tab for each. You should see output like the following:



This is webserver1-frank1



This is webserver1-frank2

OK, so now we have our servers created and set up. We don't really want to access them separately by their IP addresses, so let's set up a load balancer.

Step 6 – Create and Configure a Load Balancer

In this step, we're going to create a load balancer and use it to connect to the website. This will require us to create a front-end listener and a back-end pool but more on that as we go along.

First, click Load Balancers from the left hand menu or click through to <https://cloud.ibm.com/vpc/network/loadBalancers>.

The list of load balancers will be empty, so click New Load Balancer

New load balancer 

There's essentially three tasks to complete on the screen that appears.

The first part of the screen walks us through creating the load balancer itself. First, give the load balancer a name, again using an appropriate naming convention. I'm going to call mine 'frankfurt-lb'.

Checking that the right VPC name is shown in the Virtual Private Cloud drop-down, choose the resource group that you want to associate with the Load Balancer. This can be the same resource group or a different one to that which you used for the VPC in general. Again, I'm going to use my 'default' resource group for this example. I can also add some tags but I'm not going to at this stage.

Next, I want to have a Public load balancer, since I want traffic from the outside world to be able to access it and of course, I want the load balancer in the Frankfurt region.

The top half of the screen now looks like this:

All load balancers for VPC

New load balancer for VPC

Name	Virtual private cloud	Resource group
frankfurt-lb	frank-webapp-demo	default

Tags: ⓘ
Examples: env/dev, version-1

Type
☒ Public ☐ Private

Region
Frankfurt

Next, I need to say which subnets I want the load balancer to be able to serve traffic to – or in other words, be connect to. In this case, I want it to be able to connect to servers on both of the subnets that I have created. So, open up the drop-down and check both of the fields displayed:

Subnets ⓘ

2 x Subnets

- ☒ frank1-webapp-sn (Frankfurt 1)
- ☒ frank2-webapp-sn (Frankfurt 2)

Next, we create a Back-End Pool. A back-end pool is basically the pool of servers that requests will be directed to, along with some rules. Click 'New Pool'.

Give the pool a name – I've chosen to use **frank-be-pool**.

New back-end pool

Name	Protocol	Method	Session stickiness	
frank-be-pool	HTTP	Round robin	None	
Health check path	Health protocol	Interval (sec)	Timeout (sec)	Max retries
/	HTTP	5	2	2

Cancel

Create

Now, this pool is going to be used to serve HTTP traffic as it's sitting in front of our webserver. We'll use Round Robin for the method, which basically means that the load balancer will send the first connection to the first server, the second to the second, the third to the first again, the fourth to the second and the fifth... well, you get the picture. Now you can also use weighted and least connected but for this simple example, we'll stick to round

robin. Session stickiness lets us say whether or not to use the same webserver for all connections coming from a particular source, which is useful where some session information is stored. However, for this simple example (and to show that it works), we'll leave that as 'None'.

We'll leave the health check path as-is too but what this is essentially saying is that the load balancer will check each webserver using HTTP, every 5 seconds to check that it's still alive. If the check fails twice, then it will deem the webserver as failed and stop sending traffic to it.

Click 'Create' to create the back-end pool.

When you've created the back-end pool, you'll see the word 'attach' appear in the table, as in the screenshot below:

Back-end pools

[New](#)

Name	Protocol	Health Path	Instances
frank-be-pool	HTTP	/	Attach +

Click 'Attach'.

Now, from the panel that appears, select the first subnet and then the server on the subnet.

Attach instances

Subnet: 10.243.0.0/24 (Frankfurt 1) [selected], 10.243.64.0/24 (Frankfurt 2)

Instance: Select an instance

Port: 80

Buttons: Cancel, Attach

Attach instances

Subnet: 10.243.0.0/24 (Frankfurt 1) [selected]

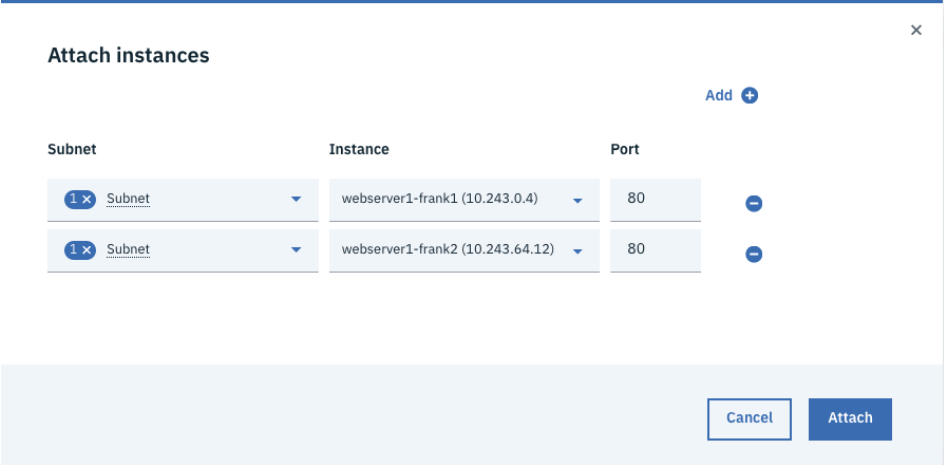
Instance: webserver1-frank1 (10.243.0.4) [selected]

Port: 80

Buttons: Cancel, Attach

Make sure the port is set to 80.

Then click Add and add the server on the second subnet in the same way.



The 'Attach instances' dialog box shows a table with two columns: 'Subnet' and 'Instance'. The 'Subnet' column has two entries, both labeled 'Subnet'. The 'Instance' column has two entries: 'webserver1-frank1 (10.243.0.4)' and 'webserver1-frank2 (10.243.64.12)'. The 'Port' column has two entries, both labeled '80'. There are minus signs in the rightmost column of the table. At the bottom right, there are 'Cancel' and 'Attach' buttons.

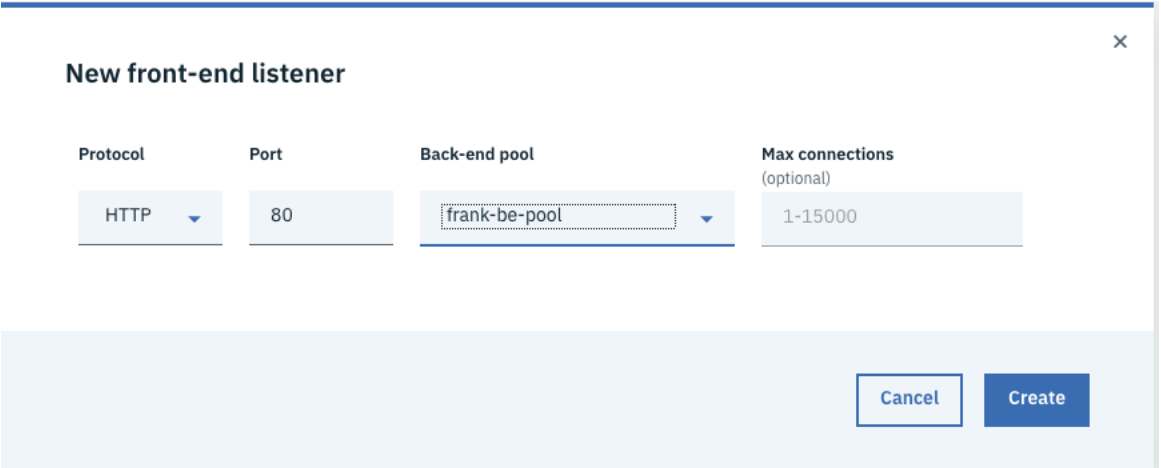
Subnet	Instance	Port
Subnet	webserver1-frank1 (10.243.0.4)	80
Subnet	webserver1-frank2 (10.243.64.12)	80

Click Attach.

Next, we need to create a Front-End listener.

Click New Listener.

In the panel that displays, set the Protocol to HTTP and the Port to 80. The back-end pool value should show the name of the back-end pool that you just created. If it doesn't then select it. Don't worry about adding a value for Max Connections but if you have a website which can handle a certain number of connections at a time, then you can set a value here. Say you set it to 100 and 100 connections are connected. The 101st attempted connection will be refused.



The 'New front-end listener' dialog box shows four fields: 'Protocol' (HTTP), 'Port' (80), 'Back-end pool' (frank-be-pool), and 'Max connections (optional)' (1-15000). At the bottom right, there are 'Cancel' and 'Create' buttons.

Protocol	Port	Back-end pool	Max connections (optional)
HTTP	80	frank-be-pool	1-15000

Click Create.

Your Load Balancer page should now look something like this:

All load balancers for VPC

New load balancer for VPC

Name
frankfurt-lb

Virtual private cloud
frank-webapp-demo

Resource group
default

Tags:
Examples: env:dev, version-1

Type
☒ Public ☐ Private

Region
Frankfurt

Subnets
2 x Subnets

Back-end pools

New pool +

Name	Protocol	Health Path	Instances
frank-be-pool	HTTP	/	2

Front-end listeners

New listener +

Protocol	Port	SSL Certificate	Back-end Pool
HTTP	80		frank-be-pool

To the right-hand side, note the Order Summary. Again, load balancers do incur a charge so if you are following this exercise, this is another point where you are committing to spending some money! If you are happy, click the Create Load Balancer button.

Order summary United States ▼

Load balancer instance
Service usage hours
Data processed - \$0.009/GB

Estimated monthly \$20.16

[Create load balancer](#)

[View docs](#)

[Get sample API call](#)

You'll land back on the Load Balancers for VPC page and see that your Load Balancer is creating. This can take a few minutes to complete

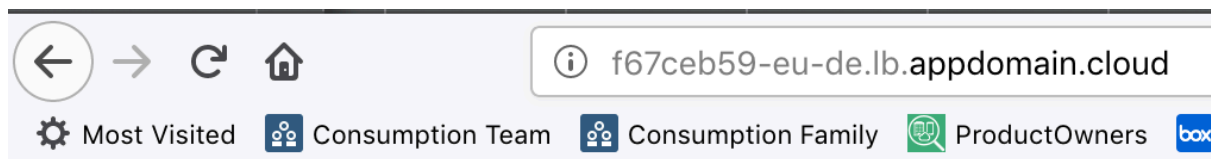
Load balancers for VPC

REGIONS
Frankfurt

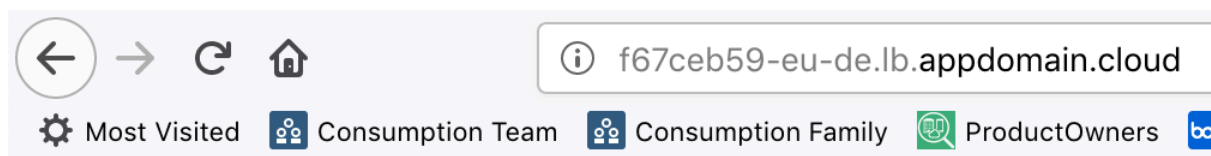
Status	Name	Resource Group	Type	Hostname	Region
● Creating	frankfurt-lb	default	Public	f67ceb59-eu-de.lb.appdomain.cloud	Frankfurt

Once the Load Balancer shows as 'Active', its time to test it. On the same screen you'll see a value for the load balancer under the 'Hostname' heading. This is the public hostname of your load balancer. Copy it and paste it into a browser.

You should see the webpage from your first webserver:



Reload the page (hold shift and click the reload button).




The page reloads but from the second server.

The load balancer is correctly set up and is working.

If you want to further test, try stopping one of the server instances and refreshing the web browser. You'll notice that you then only see the remaining web server.

You can also see the 'health' status of the pool by clicking on the name of the load balancer from the Load Balancers screen:


Health status

Pool Name	Health Status	Listener Protocol	Listener Port
frank-be-pool	 1 / 2	HTTP	80

Here, you can see that 1 of the two servers in the pool is operational.

If you restart the server (note you may need to attach to the server via ssh and restart the webserver too), then eventually, the health will return to 'green'.

Health status

Pool Name	Health Status	Listener Protocol	Listener Port
frank-be-pool	 2 / 2	HTTP	80

Congratulations! You have now set up a Virtual Private Cloud, running a highly available website over two zones.

Next Steps

You can of course build out from this example to create further subnets and further webhosts. You could also build out private subnets that contain hosts that run databases which underpin the website (see the architecture diagram in the VPC Architecture section of this document).

However, if you are finished with this exercise and want to stop your spend, then I would suggest that you:

Delete your load balancer

Click the three horizontal buttons at the end of the Load Balancer table row and click Delete (note you will need to confirm the action in a panel that then pops up):

Load balancers for VPC

regions		New load balancer ⓘ				
Frankfurt ▾						
Status	Name	Resource Group	Type	Hostname	Region	
 Active	frankfurt-lb	default	Public	f67c0b59-eu-de-lb.appdomain.cloud	Frankfurt	...
						Delete

Release the Floating IPs on your Servers

From the left hand menu, click Floating IPs (or click <https://cloud.ibm.com/vpc/network/floatingIPs>). In the table, click the three horizontal buttons next to the first server name and click 'Release'. Repeat for the other server.

Floating IPs for VPC

regions
Frankfurt

Reserve floating IP

Status	Address	Location	Associated Device	
Associated	161.156.80.91	Frankfurt 2	Public gateway on frank-webapp-demo	...
Associated	158.177.184.97	Frankfurt 1	Public gateway on frank-webapp-demo	...
Associated	158.177.184.71	Frankfurt 1	webserver1-frank1 - eth0	...
Associated	161.156.80.152	Frankfurt 2	webserver1-frank2 - eth0	...

Items per page: 10 | 1-4 items

Unassociate
Release
Copy UUID

Delete your Virtual Server Instances

Click Virtual Server instances on the left-hand menu. From the table, click the horizontal buttons in the row of the first server and click Delete. Repeat for the second server.

Virtual server instances for VPC

regions
Frankfurt

New Instance

Status	Name	Virtual Private Cloud	Private IP	Floating IP	
Powered On	webserver1-frank1	frank-webapp-demo	10.243.0.4	—	...
Powered On	webserver1-frank2	frank-webapp-demo	10.243.64.12	—	...

Items per page: 10 | 1-2 items

Stop
Reboot
Delete

If you want to delete the rest of your VPC, then visit the Subnets page and delete the two subnets that you have created and finally, visit the VPCs screen and delete the VPC itself.

Conclusion

Hopefully, you've found this document useful and you've been able to create the worked example. If you have any issues, then please leave a comment and let me know.

Over the next weeks, I'll try and update the document as needed and possibly throw in some more examples, for example a private Subnet with a database.

Thanks for reading!

James Belton

May 2019