Practical Worksheet 5

Version: 1.0 Date: 12/04/2018 Author: David Glance

## Learning Objectives

1. Networking and VPC
2. KMS Key Management System – creating keys and using the key for symmetric encryption
3. Using AES Encryption

## Technologies Covered

Ubuntu

AWS

AWS VPC

Networking

Python/Boto scripts

VirtualBox

**Note**: Do this from your VirtualBox VM – if you do it from any other platform (Windows, Mac – you will need to resolve any potential issues yourself)

## Background

The aim of this lab is to write a program that will:

[1] Understand how to configure different network arrangements to gain and control access to computers and other networked resources

[2] Understand IP addressing and CIDR and the meaning of TCP and UDP ports

## Networking

## [Step 1] Configure inbound IP on VirtualBox VM

This can be done in a number of ways, but we are going to use NAT port mapping. When a VM is created in VirtualBox, it defaults to creating a single NAT interface

[1] The VM does not have to be stopped for this but it isn’t a bad idea to make changes when it is stopped.

[2] In the VirtualBox Manager, select the VM you want to configure, then click Settings (Golden Gear Cog) and Network. Adapter 1 should be configured as NAT. Click on Advanced and then Port Forwarding. Set up 2 rules:

[a] Use the localhost host IP 127.0.0.1 and host port 2222 and map that to Guest Port 22

[b] Add a similar rule mapping Host Port 8080 to Guest Port 80

[3] **Testing!** You can test the NAT’d ports by running your docker app and seeing if you can access it from your computer – the url will be <http://127.0.0.1:8080>

If you want to try ssh to the VM, you can install sshd as follows:

sudo apt install tasksel

sudo tasksel install openssh-server

start the ssh service by:

sudo service ssh start

you can stop it using:

sudo service ssh stop

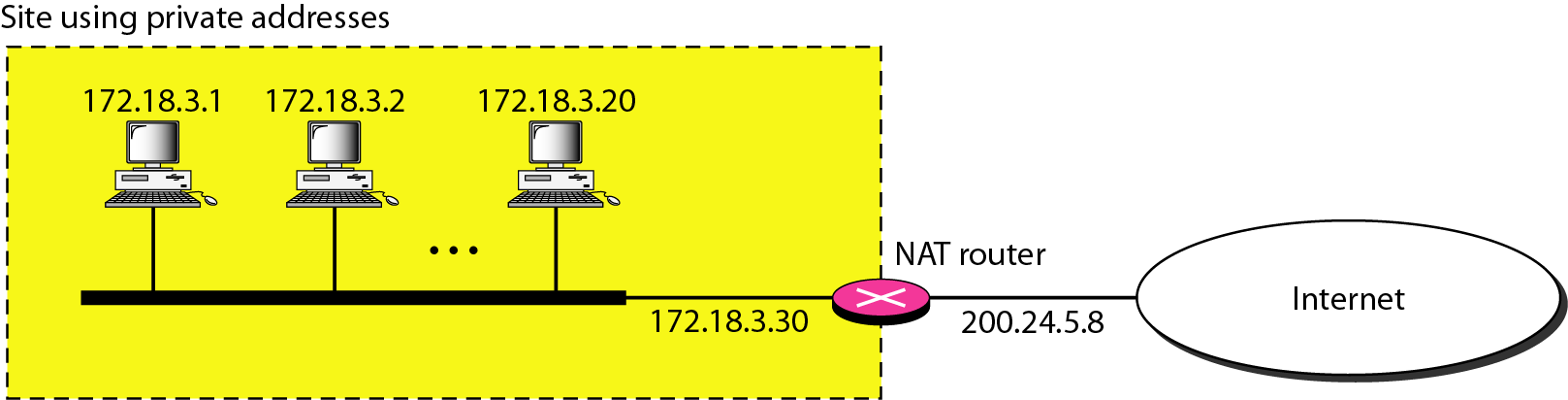
To ssh to the VM, open a terminal on your PC (or use Putty) and ssh as

ssh -p 2222 <usermame>@127.0.0.1

You should be prompted for your password

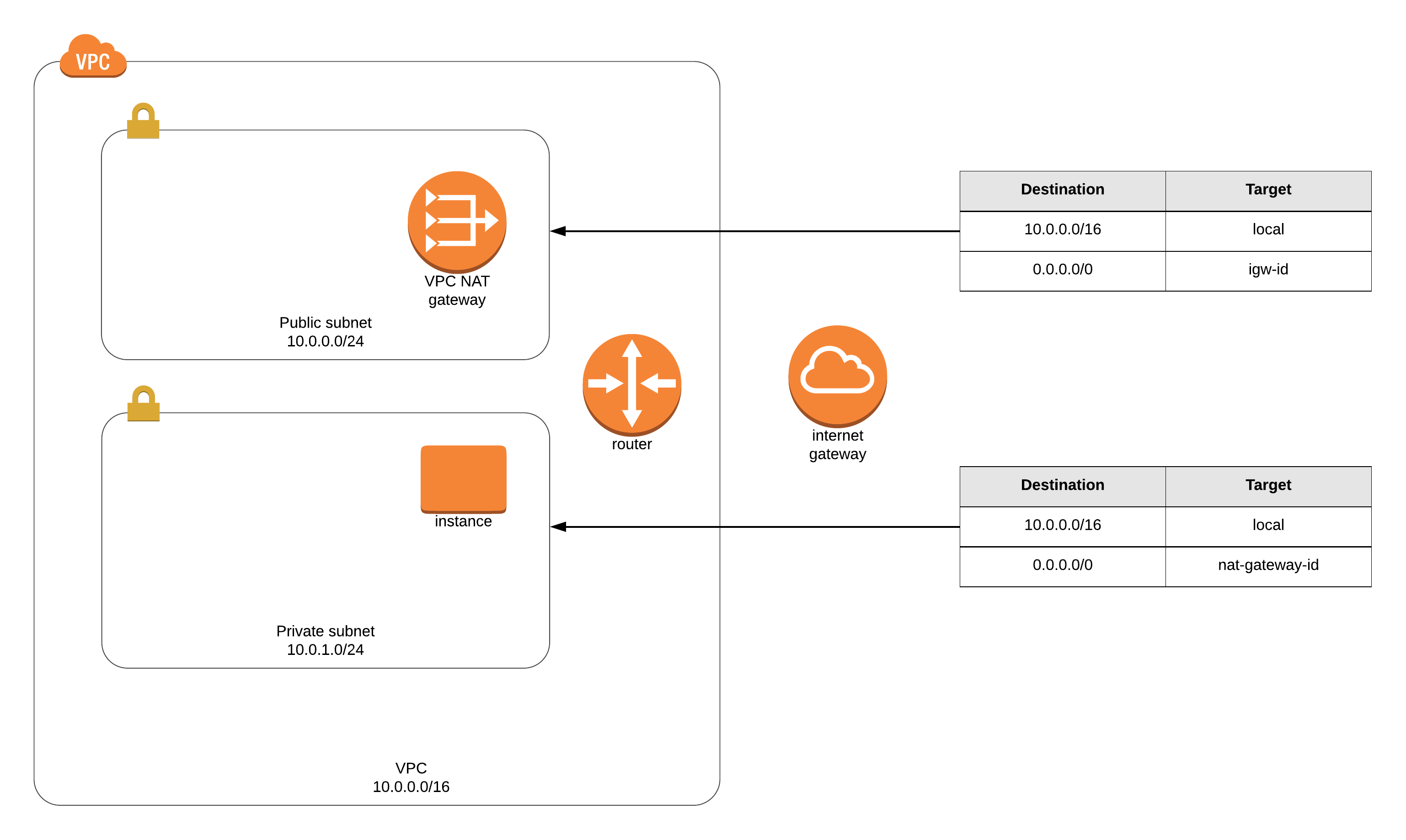
## [Step 2] Creating a NAT’d network

The aim of this part of the lab is to create a private network that has access to the internet but the internet has no direct access to any of the machines. Visually, this looks like:



This is very similar to how your home network will be set up. What we are going to do however is to set up a NAT gateway on AWS.

To do this, we are going to create a network as shown below:



[1] **Create a VPC**: Choose a name for the VPC like LabVPC and assign an IPv4 address range – this is a private IP address range – something like 172.30.0.0/16 – we don’t need 64k IP addresses that this gives us but it makes it easier to subnet. Ignore IPv6

[2] **Create two subnets**: One for the public subnet (LabPublic) with the address range 172.30.0.0/24 and the other for the private subnet (LabPrivate) with the address range 172.30.1.0/24

[3] **Create an internet gateway** to connect the VPC to the internet – this happens in two stages – create the internet gateway and then select and attach to the VPC **NOTE** this takes a bit of time to show up as being “attached”

[4] **Create NAT gateway**: Associate it with the LabPublicSubnet and allocate an Elastic IP – this is the gateways public IP address to communicate with the internet

[5] **Create two route tables**

[a] LabDefaultRouteTable – associated with the LabVPC. Once created, associate the LabPublicSubnet to it and in Routes, add the default route of 0.0.0.0/0 with a target of the internet gateway.

[b] LabPrivateRouteTable associated with the LabVPC. Once created, associate the LabPrivateSubnet and in Routes, add the default route of 0.0.0.0/0 with a target of the NAT gateway

You should be all set now to create 2 ec2 micro instances. 1 in the public subnet and the other in the private one.

Once you have created the machines, you should be able to ssh only to the machine in the public subnet – even if the instance in the private subnet has a public IP. This is because the ec2 instance in the private subnet can only communicate out via the NAT gateway and this doesn’t allow inbound traffic. However, you can still get to the private instance via the public instance. To do this, you can use the internal IP addresses – when you select an EC2 instance in the console, it will provide those details.

[6] Verify you can ssh to the instance in the public subnet but not the private one from your machine

[7] ssh to the private instance from the public instance using the internal IP address of the machine

[8] Once on the private machine, verify your public IP address by using the command:

dig +short myip.opendns.com @resolver1.opendns.com

This IP address should be the same as the public IP address assigned to your NAT gateway.

## [Step 3] Optional: Write a python program that creates the network above and adds the instances to the different subnets.

## Submission and Quiz

No submission – respond to the quiz

## Respond to the Quiz

[1] Select the option that is false. A NAT gateway or interface is:

[A] Avoids machines needing public IP addresses by allowing them to use private IP addresses

[B] Can only allow machines to access the internet and can’t allow machines on the private network to be accessed from the internet

[C] Needs a public IP address and a way for it itself to access the internet

[D] Works by substituting its own public IP address for the internal IP address of the machines on its private network

[2] Is the following statement true or false: A subnet associated with a VPC needs to allow at least 64 hosts

[A]True

[B] False

[3] Which component of the network is responsible for allocating IP addresses to machines on the network?

[A] VPC

[B] Subnet

[C] Route Table

[D] NAT Gateway