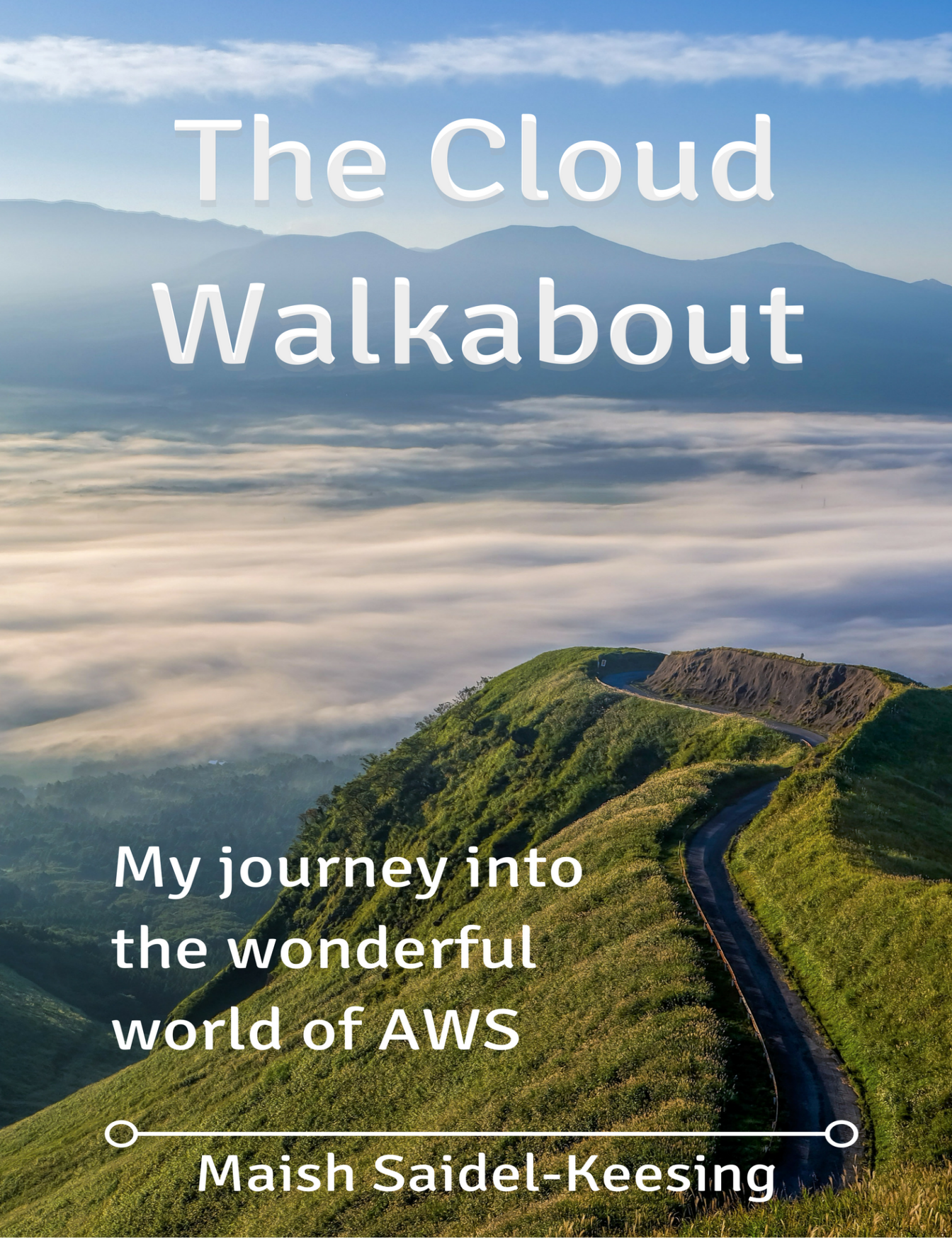

The Cloud Walkabout
My journey into the wonderful world of AWS
Maish Saidel-Keesing

The Cloud Walkabout

My journey into the world of AWS

Maish Saidel-Keesing

This book is for sale at http://leanpub.com/cloudwalkabout

This version was published on 2017-11-23

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

© 2017 Maish Saidel-Keesing

Tweet This Book!

Please help Maish Saidel-Keesing by spreading the word about this book on Twitter!

The suggested tweet for this book is:

I just purchased ”The Cloud Walkabout” http://cloudwalkabout.com by @maishsk.

The suggested hashtag for this book is #cloudwalkabout.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

#cloudwalkabout

Contents

Dedications

Although my mother will never be able to read this, I hope that this book with continue to bring you nachat.

For my wife and 3 daughters. Thank you for continuous support, and believing in me.

Acknowledgements

First and foremost I would like to thank Hashem Yitbarach for gifts and opportunities he has continuously brought my way over the years and in my career.

Danja, Noa, Michal and Avital. You were was surprised when I decided to embark on another book journey. Thank you for your support, and allowing me to pursue my dreams once more.

To my friends and colleagues in Hercules team. Courtney, David, Ezra, Mark, Moshe, Raanan, Ram, Sam, and Udi. You are most talented group of people I have ever had the pleasure to work with, a team that is so creative in so many ways and a team that cares so much about what they do. We achieved the impossible (even when I had my doubts) and I am honored to be a part this group that changes the world, over and over again. Each and every single one of you deserve to be here with me in this book - as it is as much yours, as it is mine.

To Efrat, Ronnen and Shelly - my trusted advisors from AWS. Thank you for putting up with our ridiculous requests at all hours of the day, helping us putting out fires, and escorting us in our journey.

Why did I want to write this book?

At the beginning of 2017 - I dived into a project that required us to migrate a large number of instances from an OpenStack cloud to AWS (Amazon Web Services). This required me to get acquainted with the AWS services that we were already using and adapt rapidly to a whole new world.

I have been writing on my blog for over eight years and I have already co-authored two other technical books, but I feel that when solely focusing on the technical aspect of a certain subject, you miss out on the human aspect of the story. The Why, the How and What the heck happened.

Over the years while attending countless conferences - I learned that the presenters who actually tell a story - those are the presentations that you remember. They are personal - the presenters expose themselves and share - and they share from their heart. That is why these presentations are so good, and they stick.

The first technical novel I read was The Phoenix Project by Gene Kim, Kevin Behr and George Spafford (ISBN: 0988262592)[1] and for the first time - I enjoyed reading a story about IT and what problems they were facing. I could relate to the situations throughout the book, I had been in that position, many times. If you have not yet read the book - I highly recommend you do so.

When starting out my AWS journey I was looking for how do I accomplish this - or what is the best way to get that to work - and what I found out there was a whole lot of AWS documentation (which is welcome) and number of blog posts, and of course the usual number of technical books.

I felt that the technical terms, books and white papers out there and information is great - but I was missing the background behind it all. Why did this happen? What was the problem they were trying to solve? As I said before, a good part of the delivery is the story behind it.

[1] https://www.amazon.com/Phoenix-Project-DevOps-Helping-Business/dp/0988262592

And then this book was born.

What is with this walkabout thing?

> Walkabout historically refers to a rite of passage during which Indigenous male Australians would undergo a journey during adolescence, typically ages 10 to 16, and live in the wilderness for a period as long as six months to make the spiritual and traditional transition into manhood.
>
> (Source: Wikipedia)

I wanted to write this book to document a journey - from a beginner - to what some might perceive as an expert (although I have never considered myself an expert in anything). I learn, I share and I gain more knowledge from others that learn like me.

Is this a rite of passage as the term actually depicts? Perhaps not - but it certainly was a process I underwent - and during this process I learned a significant amount of interesting and new parts of the AWS world - that I was not aware even existed.

Is it spiritual? Probably not - but there are things of beauty that leave you standing in awe - at the way all the nuts and bolts come together in AWS, and how seamless the whole system is to the end user.

Without further ado, fasten your seat belts, and get ready for an unforgettable ride!

About the Author

Maish Saidel-Keesing is a world renowned Cloud Architect who has been in the IT industry for the past 17 years. He is the co-author of two books on enterprise architecture:

VMware vSphere Design Guide[2]

OpenStack Architecture Design Guide[3]

He is also the owner of the popular blog Technodrone[4] and most of the time he is lurking on Twitter (@maishsk[5]).

[2] https://www.amazon.com/VMware-vSphere-Design-Forbes-Guthrie/dp/0470922028

[3] http://technodrone.blogspot.com/2014/08/the-openstack-architecture-design-guide.html

[4] http://technodrone.blogspot.com

[5] https://twitter.com/maishsk

Book Conventions

The code examples that are in this book are written in a number of languages (Ansible, bash, python) and are available on Github[6] for easier consumption.

In the book there are blocks of text that are emphasized throughout the book in the following way.

This is tip will provide additional information

Warning - to let you know “here there be dragons..”

[6] https://github.com/maishsk/cloudwalkabout_examples

This is a Leanpub book

I would really appreciate your feedback on the book. The book is in active development and I will be updating the book regularly. Updates will of course be based on feedback that I receive from you - so that is why it really important for me to hear what you think about the content.

If you find any errors, corrections or if you would like to see a section on a specific topic, please reach out to me and share.

My DM's on Twitter (@maishsk[7]) are open. You can also share your thoughts on the book's Feedback Page[8] or open an issue on the Issues section[9] of the book's example repository[10].

[7] https://twitter.com/maishsk
[8] https://leanpub.com/cloudwalkabout/feedback
[9] https://github.com/maishsk/cloudwalkabout_examples/issues
[10] https://github.com/maishsk/cloudwalkabout_examples

Introduction

Welcome to the Cloud Walkabout. This book will not be a technical manual of how your should do thing one or what is the best way to accomplish thing two.

It is story, some of it is fiction, some of it really happened.

Who is this book for?

I cannot say that this book is catered for a specific part of the technical community, because inside you will find something for people of all levels, from beginner to expert all the way to those who are upper management and finance. It is safe to say that if you are looking to this book to find detailed technical examples of how to deploy your applications in AWS - then you should probably stop right here - because this is not what this book is about.

That being said - there will be technical explanations, with detailed diagrams and examples that you can use - so if you are a geek like me then you will find a lot of useful information in the book.

How is this book divided?

These are the three stages of my journey.

1. Crawl
2. Walk
3. Fly

Crawl

One bright morning I was called into my managers office - and as I stepped in wondering what I have now screwed up - or if my boss is going to give me that talk about, "Sorry we have to downsize, there is not enough work…" and as my mind starts to take a long hard look at my past, and how to plan my future, the following happens. "Our current cloud solution is no longer viable, and upper management want us to move off of what we are using - and migrate everything to AWS. And it has to be done in the next three months."

First thing I did was breathe a sigh of relief. Phew. I was not being fired. The next thing I did was say to myself, "Crap, how the hell are we going to do this - I have no expertise in AWS"

Now I had the task of understanding what this AWS thing really is. Of course I know a bit about AWS - being the market leader and innovator in the public cloud, but I did not really know exactly how it worked. So the first part was research. And a lot of it.

This is the *crawl* phase.

Understanding what EC2 is. How does a VPC work? Network segmentation. Building AMI's. Which flavors should I be using? Deploying a few instances to test the waters, all of these and more are the start of your journey.

Walk

You now have the basic understanding of what you would like your VPC and deployments to look like. You have started to deploy some of your instances in AWS. Now it is time to start looking at some of the more advanced topics and 'Enterprisy' things.

Central Management. How do you synchronize code to AWS? Understanding cost - and where should you start your optimizations to start saving money (who wouldn't love to get their CFO off their back?). Expanding beyond a single region. Load Balancing - and what do you do when AWS' ELB does not serve your purpose? All of this and more.

Here is where you walk the *walk*.

Fly

The term that most of you are probably accustomed to is Crawl, Walk. Run.

Why did I choose fly? The answer to that is simple. One you are at the stage where you have your infrastructure deployed on AWS, running (hopefully like a well oiled machine), this is the stage where you start to look for ways to improve your deployments, improve the way you operate, and streamline your day-to-day operations.

With the possibilities that AWS provides, the services that you have available at your fingertips - will allow you to soar to new heights, develop software that you never thought was possible, in ways you never imagined.

This is where you learn to *fly*.

Part I - Crawl

If all you can do is crawl, start crawling.

(Rumi)

Chapter 1 - The Basics of EC2

People come to the cloud, partly because of the flexibility and the ease with which you can deploy your solution. But to put it in layman's terms, and in plain English - we need a place to run our applications and running your applications means you need a computer running in the cloud.

That is why I am going to start with AWS EC2 as the first and most basic part of this book.

Elastic Compute Cloud (one *E* and two *C*'s - hence the name EC2) is the basic pillar of running your application in the cloud. When you need a instance (the cloudy way of saying a computer or a VM) you go to EC2 to hook you up.

Being the IT geek that I am, I know that I obviously need to have connectivity to a network of some sort in order for a cloud instance to be of any use to me - so the first thing I set out to understand was, AWS networking.

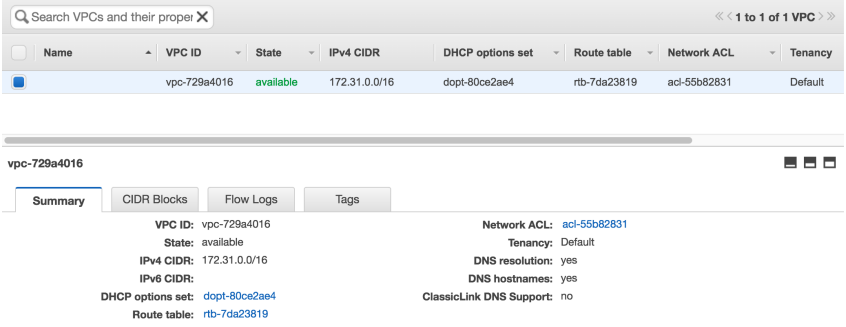
VPC Networking

First things first. Let us understand the concept of a Public IP address and a Private IP and the differences between them. A Public IP is of course one that is accessible from the public internet and a Private IP is not accessible from the outside - it is only accessible from within the network that the instance exists. Let me take an analogy from everyday life that should be easy to understand.

You live in a house on 123 ACME st. in Dudesville. Your house has two floors each with 3 rooms. You want to press the order button for a mirror for your bedroom wall from Amazon, you are prompted to enter your shipping address. It would not enter your mind to enter "my bedroom on the second floor", because no-one knows where that is. You need to provide a shipping address that everyone knows how to find (that is your Public IP) and not which room you want it delivered to (your Private IP). But when you want to get your kids to clean up their bedroom or put away their

laundry, you would not ask to put their laundry away in 123 ACME st. because they are already inside the house - and they know where their room is in the house, how to get there and where they need to go. Your house is your private network - you know how to get from room to room (from one computer to another) and you never need to use your street address (Public IP) when you are inside the house.

By default Amazon creates a VPC for you in every region (I will get into regions and availability zones later on) - and this VPC is tagged as the *default* VPC (You do not control the IP address range of the VPC).



AWS Default VPC

Hold on a Minute..

What is a VPC? An AWS Virtual Private Cloud is your small (or large - depending on your needs) part of real estate in AWS (and this real estate is free of charge). This your boundary,going back to the example above, this would be your plot of land where you can build your house. Will it be a simple caravan? Perhaps a 2 story house or maybe a 60 room palace? You decide - it all depends on your means and your dreams - you can decide. The same goes for your VPC design, it can be practically anything you want (with some limitations)

AWS CIDR block limitations

These are the current limitations[11] on AWS CIDR blocks in a VPC:

10.0.0.0 - 10.255.255.255 (10/8 prefix)

172.16.0.0 - 172.31.255.255 (172.16/12 prefix)

192.168.0.0 - 192.168.255.255 (192.168/16 prefix)

Public vs. Private Networks

The next concept that is important to understand is the difference between a private and a public network.

A public network is a subnet that will require you to attach a Public IP an instance on that subnet - in order for you to access the rest of the world (download patches, files) and in order for the rest of world to access your instance (serve web pages, for example).

A private network will never be directly accessible from the outside world - meaning instances deployed on a private network - can never be accessed directly from the outside world. You cannot attach a public IP address to an instance on a private network (actually you can manually do this - but it won't work).

Going back to the house analogy. Your house is built of an exterior and an interior. The exterior has walls and some of these walls have doors and windows - to allow people, air and light into your house. You go through the door - to come into your house and out the door to get to work (that is the public network). But for an interior room inside the house with no exterior walls - you cannot access it from the outside - without coming in through one of the exterior walls, doors or windows. You cannot make an entrance directly from your garden to an interior room. It is physically not possible. You can become creative - by digging tunnels under your room to the garden (why you want to is another whole discussion) or creating a chute to allow air or light in - but that requires more detailed planning and thought.

[11] http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Subnets.html#vpc-sizing-ipv4

ELB's

When using an Elastic Load Balancer (ELB) you can direct traffic from the outside world to an instance on a private network. I will get into more detail on ELB's in a later chapter.

External Interfaces

Connectivity is everything and connectivity is achieved through a number of connections or gateways. For me this was quite difficult to grasp in the beginning - it is important to understand the differences between them and when each one should will suit your needs.

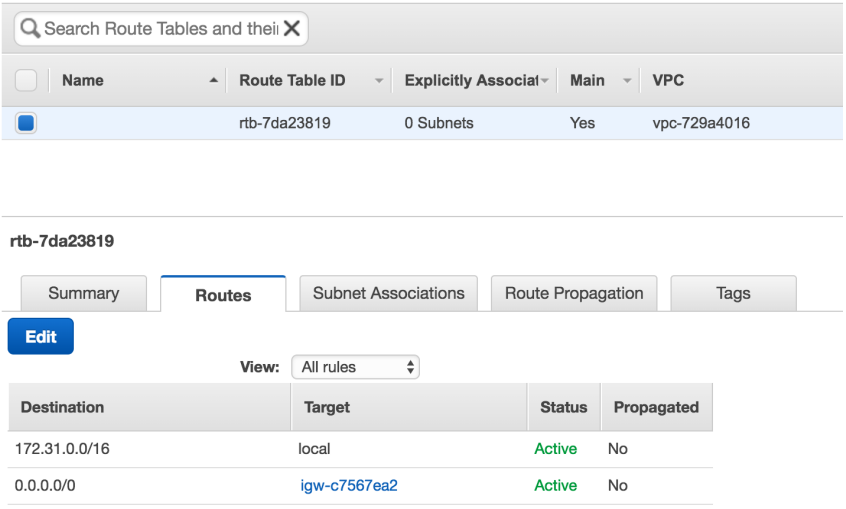
The Internet Gateway (IGW)

This is connection from your VPC to the world. All traffic from your VPC outside of the AWS cloud - flows through this interface.

There are a couple of points that you need to understand I want to make sure that they are clear because they are crucial.

1. There can be only one (I loved the Highlander when I was a kid). One Internet gateway per VPC.
2. Your IGW does not have an IP address (at least not one that your are exposed to - AWS does not have public information on how traffic is routed through your IGW to your resources). You cannot ping it, you cannot manage it, you cannot scale it - basically - it is a single line in your routing table - and that is it. AWS takes care of everything else.
3. Anything that is routed through the IGW **must** have a public IP address.

I want to explain a bit more about this last point. A paragraph or two ago, we discussed the public subnet. The default route for a public subnet - is the IGW in your VPC. See the diagram below.



Default Route Table

When I wrote that instances on the public subnet must have a public IP address - this is precisely the reason why. You can consider these instances to be sitting basically on the same network (or if it makes it easier to understand - on the same hub - but I am sure it is not an actual physical hub) as the IGW.

Trusting AWS

In the beginning I found it hard to accept that I am not going understand precisely how everything works - because I very comfortable with managing my own resources all the way - from compute to network to storage, so understanding how each packet flows is not something I am afraid of. When moving to a platform like AWS - you will need to relinquish some control - and trust that someone else is handling it for you.

Rolling along..

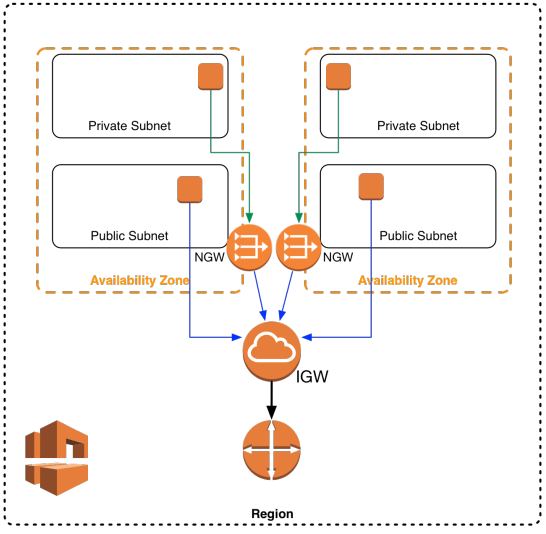
NAT Gateways (NGW)

Instances on your private network will probably need to access things on the internet (patches, packages and the what-nots) but as I stressed before they are not connected to the internet - nor to the public subnet - and here is where the NGW comes in.

A NAT Gateway is an AWS service (again not something you can physically manage) deployed for you. To dumb it down a little - this is an instance that has two network interfaces. One on the **public subnet** and second on the **private subnet**. Traffic from an instance on a private subnet goes out through the NGW, and comes back in same way.

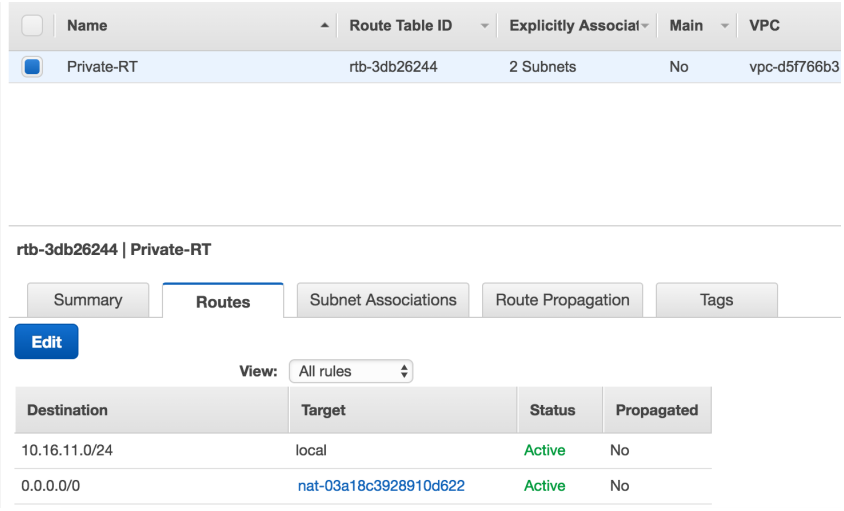
Let me add some important points you should be aware of.

1. You might have noticed that the section title mentioned Gateways - in plural. Unlike the IGW - you can have multiple NGW's in your VPC. How many exactly? As many as you want - but you can only route traffic from your subnet to a single NGW.
2. NGW's are dependent on a single availability zone. If an availability zone goes down (and yes it happens) then your NGW is also down.
3. How many NGW's should you deploy? One per availability zone (AZ). If you have one AZ (and you should ***never*** have a single AZ), then one NGW. If your solution deployed across four AZ's then you should create 4 NGW's.



NAT Gateway Traffic Diagram

Routing your traffic through the NGW - is also done through the routing table of your VPC.



NAT Gateway Routing

When creating the routes - make sure that the subnet and the NGW are in the same AZ - otherwise you will start wondering why your are getting paged at 03:43 when the shit hits the fan - and nothing is working.

NAT Gateway Instances

NAT Gateway instance are available - but I have not found a use for them. The NGW service is by far easier to use, it scales, and zero management on my part.

VPN Gateway

In my experience there are certain cases where you actually need to connect your VPC to an network somewhere on the outside. It could be another VPC outside of your AWS region, it could be different cloud provider or it could be a service in your physical datacenter.

You could always set up a point-to-point VPN from a single instance a public subnet - but that doesn't scale - at all. In comes a site-to-site VPN (as a Service - of course).

I am not going to go into the details of how you create a VPN gateway - the AWS documentation[12] does a great job of explaining precisely how to do this. I am mainly interested in the networking aspect of this gateway.

The VPN Gateway (VPGW) is another destination (similar that of the IGW) in your VPC that you can route traffic through over a VPC connection to the location of your choice.

Ponder on these points.

1. The VPGW service of AWS - is redundant (from the public perspective - you have two distinct IP addresses for the connection)
2. To provide a fully redundant solution - you will need a highly available endpoint on your side

[12] https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_VPN.html

Before I go onto the next part of my EC2 journey, a quick recap

- There are public and private subnets
- There are 3 ways to route traffic in and out of your VPC - each of these has a valid use case.
  - Internet Gateway (IGW)
  - NAT Gateway (NGW)
  - Virtual Private Gateway (VPGW)

EC2 Instances

OK. Now that I have that part out of the way - let me get into the intricate and mind boggling world of AWS compute instances (don't worry there is a whole lot more besides the actual compute in AWS EC2).

Types and Families

I love explaining in ways that we can all relate to in our day to day lives. We all have clothes. And to make it simple - let me take a shirt as an example. Small, Medium, Large and Extra Large - assume these are the sizes that you choose from when walk into a store. I use either a Large or an XL - depending on the country that the store is in. I cannot do a medium - otherwise when I lift up my arms - people run aways in horror - because my belly button is out there. With a Small - I have trouble getting it over my head and I have trouble breathing - because it is so tight.

EC2 instances have the same kind of sizing - and these are your *Instance Types*. The service provided by AWS - dictates (you cannot create your own) what types are available - and they will differ from one instance family to the next.

Families? What is an Instance Family? Back to my real life example. You have your sizes (Large or Extra Large - remember?), but what kind of shirt do I want to buy? A tank top - without sleeves? A crew-cut? A v-neck? A Polo shirt? Buttons? Long sleeves? Short sleeves? With graphics? Without? What color? All these are choices you will make when walking into a store.

Instance families are the range of products you can choose from when walking into the store at AWS.

General Purpose	Compute Optimized	Memory Optimized	Accelerated Computing	Storage Optimized
T2	C5	X1	P3	I3
M4	C4	R4	P2	D2
M3	C3	R3	G3	
			F1	

As you can see there is a great deal of choice in the 'store' - and each of these have a valid use case (otherwise I assure you that AWS would not offer them). Do they fit every single use case that you have - probably not - but then the question arises - what should I do if I cannot find precisely what I need?

My answer to that question is a simple one. Compromise. Either go up or down in the amount of resources you are using. From my experience the increments are well balanced out with the current instance types you have available to you today (AWS instance types change as time goes on). And if you need more CPU than RAM - move to Compute Optimized family - the price difference between them is zero. An m4.large and a c4.large are the same price - the c4 has faster CPU's and the m4 has double the RAM and slower CPU's. Seek and you will find.

AWS for some reason makes it difficult to compare instance prices.

AWS has all the information available on their site - and recently they extended their pricing API, but for the those who want to have a spreadsheet with all the instances in all the regions - with the options to sort and compare and filter - there is nothing like the invaluable resource http://ec2instances.info[13]

AMI's

Amazon Machine Images - are one of the fundamental pillars upon which you will your build solution on AWS.

[13] http://ec2instances.info

We have an image that we use in house for all our deployments - based upon Centos 6.x and 7.x operating systems (but this will be true for what ever OS you choose.)

You can use the Amazon Linux AMI - which offered by AWS themselves, and this includes several obvious benefits, for example, the AMI is maintained by AWS, they provide the support for it, the AMI is constantly updated - and critical patches and updates are automatically applied. The AWS tools are embedded into the image and there is no need to install software to interact with the AWS endpoints.

Importing/Exporting Images.

AWS allows you to import existing images you might have and would like to use also in the cloud. It goes without saying that it is in their best interest - but I would have hoped that they would have made it easier to import AMI's. The documentation is clear and concise - but for someone that is barely starting out with AWS - to start fiddling around with CLI commands is not something that I would recommend for the faint of heart.

When importing the AMI's you will not know if they will work until you try and boot them up, and if you made a mistake while packaging the virtual disk - then you will need to upload the files again, until you get it right. The upside is that upload traffic to S3 is free - and you pay for storage consumed by your files. But uploading GB's of files can take time - a whole lot of time. The total amount of time will depend on your location, your bandwidth, and the size the files that you are dealing with. The smaller your images, and faster the pipe you have - the less amount of time it will take, so don't go and create images that are 80GB in size the sole reason being, that was the original size when you deployed on bare metal. The larger they are, the longer it will take for your instances to start up.

The other option you have - and this is the route I would suggest you take - is to build your own images - directly from AMI's that already exist in AWS. You could choose to build it upon Amazon Linux - but it does not have to be. All the major Linux vendors have official images that you can use from the Amazon Marketplace.

A Word of Warning

Make sure you use approved images from the Marketplace. There are a lot of community images - which people from the AWS community have uploaded - and many people use - but I would compare this to a hotel room. I would only stay at a hotel after I have read its reviews and feedback and I know that they have a good name. Otherwise you never know what kind of bed bugs and other malware you will inherit from the previous tenant. My 0.02 Shekels.

I would not be true to myself if I didn't actually have some code in the first chapter, so I am going to show you how can do this with a simple Ansible playbook.

Ansible as my tool of Choice

In the initial stages of the project - for reasons that I prefer not to disclose, the decision was to go with Ansible as the deployment tool. Would it have been my tool of choice if I had to do it all over again - I am not so sure. There are a number of tools out there. I have grown to love and hate Ansible both at the same time. It is what it is.

The example below will assume you already have the following in place:

1. An AWS account
2. A Linux machine with Ansible installed (version >= 2.2)
3. A VPC - with at least one public subnet
4. A public keypair defined in the region

```
---
- name: Connect to AWS
  hosts: localhost
  connection: local
  gather_facts: True
  vars_prompt:
    - name: "vpc_id"
      prompt: "Please enter your VPC ID (vpc-xzy12345)"
      private: no
    - name: "region"
      prompt: "Please enter your AWS region (for example us-east-1|eu-west-1\
|ca-central-1)"
      private: no
    - name: "subnet_name"
      prompt: "Please enter name of a public subnet in your VPC
      private: no
    - name: "keypair"
      prompt: "Please enter name of your SSH keypair
      private: no

  vars:
    instance_type: t2.small
    component_name: centos_image
    use_public_ip: "yes"
    instance_count: 1
    volume_type: gp2
    root_disk_size: 2
    component_rules:
      - proto: tcp
        from_port: 22
        to_port: 22
        cidr_ip: "0.0.0.0/0"

  tasks:
    - name: Find CentOS 7 AMI
      ec2_ami_find:
        name: "CentOS Linux 7 x86_64 HVM EBS*"
        region: "{{ region }}"
```

```
39          sort: name
40          sort_order: descending
41          sort_end: 1
42        register: centos_image_7
43
44      - name: Create Component Specfic SG.
45        ec2_group:
46          name: "{{component_name}}_SG"
47          description: "Security group for {{component_name}} purposes"
48          vpc_id: "{{ vpc_id }}"
49          region: "{{ region }}"
50          rules: "{{component_rules}}"
51          rules_egress:
52            - proto: all
53              cidr_ip: 0.0.0.0/0
54          state: present
55        register: component_sg
56
57      - name: Tag the security group with a name
58        local_action:
59          module: ec2_tag
60          resource: "{{component_sg.group_id}}"
61          region: "{{ region }}"
62          state: present
63          tags:
64            Name: "{{component_name}}_SG"
65
66      - name: Deploy CentOS 7 instance
67        ec2:
68          region: "{{ region }}"
69          key_name: "{{ keypair }}"
70          image: "{{ centos_image_7.results[0].ami_id }}"
71          wait: yes
72          wait_timeout: 300
73          group_id: "{{component_sg.group_id}}"
74          count: 1
75          monitoring: no
76          instance_tags:
```

```
77            Name: centos7_instance_temp
78          vpc_subnet_id: "{{ subnet_name }}"
79          assign_public_ip: "{{ use_public_ip }}"
80          instance_type: "{{ instance_type }}"
81          volumes:
82            - device_name: /dev/sda1
83              volume_type: "{{volume_type}}"
84              volume_size: "{{ root_disk_size }}"
85              delete_on_termination: True
86        register: ec2
87
88      - name: Add new instance to host group
89        add_host:
90          groupname: serversToInstall
91          hostname: '{{ item }}'
92          ansible_ssh_user: centos
93          ansible_ssh_private_key_file: "/Users/msaidelk/.ssh/id_rsa"
94        with_items:
95          - "{{ ec2.instances[0].public_ip }}"
96        register: ec2_ip
97
98      - name: Wait for the instances to boot by checking the ssh port
99        wait_for:
100         port=22
101         delay=60
102         timeout=300
103         state=started
104         host = "{{ item.add_host.host_name }}"
105       with_items: "{{ ec2_ip.results }}"
106
107 ## Instance Configuration
108 - name: Configure nodes
109   hosts: serversToInstall
110   user: centos
111   become: yes
112   become_method: sudo
113   gather_facts: True
114
```

```
115   pre_tasks:
116     - name: Generalize image
117       shell: |
118           yum update -y
119           yum install -y cloud-init cloud-utils cloud-utils-growpart dos2uni\
120 x vim iotop wget
121           service rsyslog stop
122           service auditd stop
123           logrotate -f /etc/logrotate.conf
124           rm -f /var/log/*-???????? /var/log/*.gz
125           rm -f /var/log/dmesg.old
126           rm -rf /var/log/anaconda
127           cat /dev/null > /var/log/audit/audit.log
128           cat /dev/null > /var/log/tuned/tuned.log
129           cat /dev/null > /var/log/boot.log
130           cat /dev/null > /var/log/cloud-init-output.log
131           cat /dev/null > /var/log/wtmp
132           cat /dev/null > /var/log/lastlog
133           cat /dev/null > /var/log/grubby
134           rm -rf /var/log/dmesg
135           rm -rf /tmp/*
136           rm -rf /var/tmp/*
137           rm -f ~root/.bash_history
138           unset HISTFILE
139           rm -rf ~root/.ssh/
140           rm -f ~root/anaconda-ks.cfg
141           poweroff
142
143 ### AMI Creation
144 - name: Image creation
145   hosts: localhost
146   connection: local
147   gather_facts: True
148   vars:
149     component_name: centos_image
150
151   tasks:
152     - name: Wait for the instance to shutdown by checking the ssh port
```

```
153       wait_for:
154         port=22
155         delay=120
156         timeout=120
157         state=stopped
158         host="{{ hostvars['localhost']['ec2']['instances'][0]['public_ip'] }}\
159 }"
160
161     - name: Pause (wait for the instance to stop)
162       pause:
163         seconds: 45
164
165     - name: Create image from CentOS7 instance
166       ec2_ami:
167         region: "{{ region }}"
168         instance_id: "{{ hostvars['localhost']['ec2']['instances'][0]['id'] \
169 }}"
170         name: "Cloud_Walkabout_CentOS_7.0"
171         description: "The Cloud Walkbout Centos 7.0 image"
172         tags:
173           Name: "Cloud_Walkabout_CentOS_7.0"
174         wait: yes
175       register: centos7_image_name
176
177 ### Clean Up
178     - name: Terminate Temporary Instances
179       ec2:
180         state: absent
181         region: "{{ region }}"
182         instance_ids: "{{ item }}"
183       with_items:
184       - "{{ hostvars['localhost']['ec2']['instances'][0]['id'] }}"
185
186     - name: Remove Component Specfic SG.
187       ec2_group:
188         vpc_id: "{{ vpc_id }}"
189         region: "{{ region }}"
190         name: "{{component_name}}_SG"
```

```
191         description: "Security group for {{component_name}} purposes"
192         state: absent
```

Let me explain what this playbook does - step by step.

7-19: Collection of variables from the user (vpc_id, region, network etc.) *22-27:* Hard coded variables for instance creation *28-32:* To allow SSH into the instance for configuration from anywhere *35-42:* Query AWS for the Approved Centos 7 AMI *44-55:* Create a security group for the instance to allow SSH *57-64:* Tag the Security group with a Name (I will go into tagging in the next section) *66-86:* Deploy the instance *88-96:* Add the instance into the in-memory inventory for further processing *88-105:* Wait for AWS to actually bring the instance up *116-140:* Run a shell script to generalize the image *151-161:* wait for the instance to stop *163-172:* Create an AMI from the instance *174-189:* Clean up

Secure Access to your Instances

When deploying instances on AWS - and even more so - when you deploy them on a public network - you should take extreme caution how much you expose your instance to outside world. Here I opened SSH to 0.0.0.0/0 which is not a good security practice (but for the purposes of this book - I cannot foresee what your public IP address will be). Regardless of the fact how permissive the access is, you will see that of course - you will need the proper SSH keys to login to the newly provisioned instance. Since this instance is short-lived and terminated immediately after the AMI is created - this not that big of a security risk.

One last word on exporting images out of AWS. The documentation[14] explicitly states that you cannot export images out of AWS - that you did not actually import into AWS in the first place. An image that was created based on an AWS AMI that you do not own (as in the example above) will have to stay in AWS - you cannot take it out. This reminds me of the Hotel California lyrics "*You can check out any time you want - but you can never leave.*"

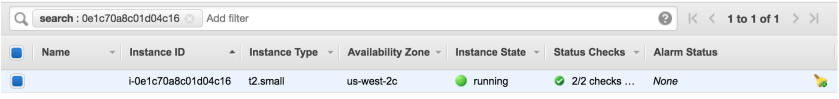
[14] http://docs.aws.amazon.com/vm-import/latest/userguide/vmexport.html

Tagging

I cannot stress this enough - and it something that you have to implement from day one. Without it - you will get lost in your journey to AWS - and I cannot convey to you how fast it will happen. Tag. Tag. Tag.

Tag everything!

Ok after I have scared the living hell out of you - let me get back to the basics. You have the option of attaching a key value pair to almost any and every resource in AWS. What that key value pair is - is entirely up to you. AWS has a number of basic values that are already created but not populated - for example the Instance ***Name***.



EC2 instance Name tag

What are these used for? Mainly making heads and tails of what you have in your cloud. Imagine to have to remember instance_ids in order to find something in your cloud? I would prefer to search by name - it is so much easier.

In the second part of this book I will go into some more details about how you should be tagging your resources - and some of the best practices that worked for me - and tools that you can use. For the time being - tag what you can - and start thinking about how you want to mark and identify the resources you are deploying in AWS. The earlier you can come up with a clear plan of how you want to tag - and what information you need to know about your deployments - the better you will be off in the long run.

Here are some leading questions that will help you along and start the discussion (even if it is only with yourself)

1. Do I need to who deployed/owns the instance? (Owner)
2. Do I need to know which project it belongs to (Project)
3. Do I need to differentiate between Production, Staging and Test (Environment)
4. Do I need to tag data classification (Secret/TopSecret/Clear)

5. Do I need to know what the component is used for (Web/FrontEnd/Database)

Congratulations for staying with me so far. I know that it is a lot to take in for your first chapter and your first part of the journey into AWS. Next up we will go into some more details about connecting environments.