Laboratory Practice Report

Practice 1

September 07, 2023

Computer Systems Engineering

Cloud Architecture

Prof. M.S. Rodolfo Luthe Ríos

Marco Ricardo Cordero Hernández

is727272@iteso.mx

ITESO

Universidad Jesuita
de Guadalajara

Abstract

As an introduction to the capabilities that cloud services can offer, a first practical objective has been set in which several computing *infrastructure* instances are intended to be created and modified to specific requirements in order to satisfy theoretical but realistic user needs.

To state further specific details, the main goal of the current activity consists of the deployment of a couple of cloud computing instances that will communicate between them over a virtual remote environment. In the current context, these computers will serve no other purposes than demonstrative ones, this is intended as an entry point for future reference and as the beginning of cloud exploration.

To achieve completion of this practice, *Amazon Web Services* [1], a self proclaimed comprehensive and broadly adopted cloud, it's being used as main platform to implement the actions required to fulfill previously stated objectives.

Documentation and relevant steps evidence it's found within following contents with explicit statements that describe what they're intending to represent. The purpose of doing so it's for anyone reviewing this activities as some form of guide for possible replication along with other learning activities.

State of the Art

In modern times, results are needed as soon as possible, leaving small and medium enterprises in disadvantage in contrast of multimillionaire corporations with edge technology, but a high cost. This situation evidently is not suitable for emerging businesses, leaving them with two options: either buying, installing and maintaining physical infrastructure, or switching to the novelties found within cloud technologies. Now, the latter won't be a universal solution, but the scope of this discussion goes beyond the purpose of the current development.

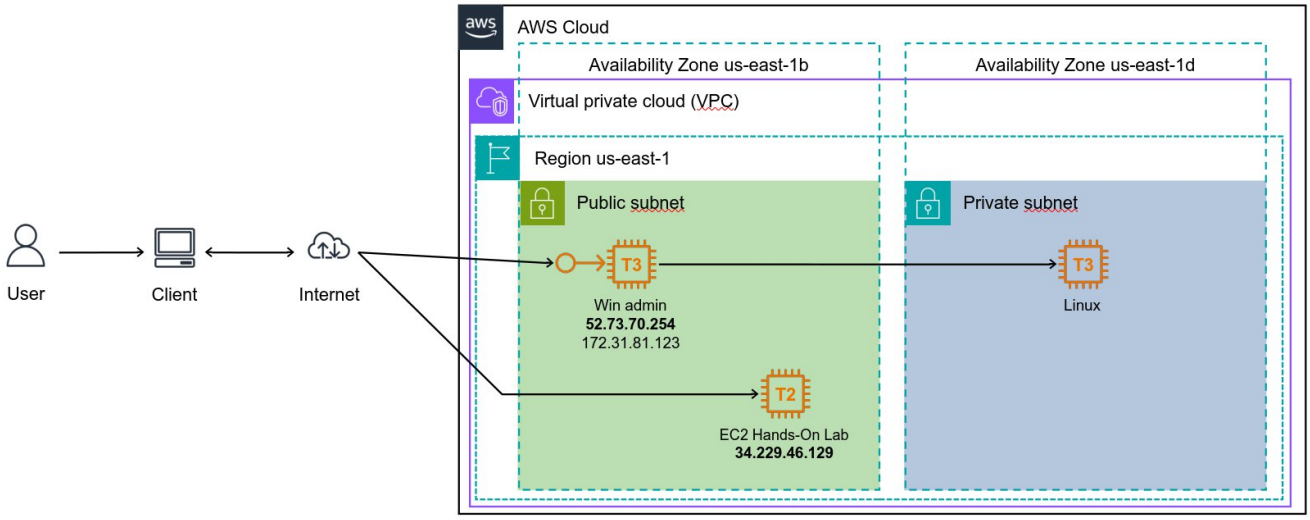
In the described context, cloud seems suitable, as an out-of-the-box solution that requires little to no complex configuration. To be more specific, *cloud computing*, which, according to Singh and Wen [2], refers to both the applications delivered as services over the Internet and the hardware and systems software in the data centers that provide those services. To further expand the stated concept, the computer distribution model of remote resources access would often be referred as *Infrastructure as a Service* [IaaS], formerly defined as infrastructure engineered, deployed and maintained by IaaS vendors, such that their clients can get on business quickly and effectively without any initial investment or tiresome installations [3].

Different clouds provide different means of operations which will ultimately achieve the same goal for educational, research, or more commonly, production purposes. Different trainings are needed for each one of them, but, as stated, the same exact results can be achieved through every single one of them. Alongside this fact, security it's also a common feature, which, nowadays it's not an option anymore, but a necessity, a requirement. Most cloud providers will offer different layers of security, enabling users with the possibility of going as secure as they want, or in any case, as insecure.

For the purpose of this development, several cryptographic applications are involved, such as key pairs, a method of data encryption which involves a public key, used to encrypt data, and a private key, used to decrypt data [4]. Also, secure means of communication it's used to communicate between a bastion host and non-accessible via internet instances, being this SSH [Secure Shell Environment], a network communication protocol that enables two computers to communicate and share data, securely, between them, by encrypting what it's being transmitted [5].

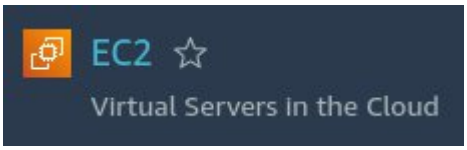
Diagram

The following architecture it's proposed as a graphic for solution for the stated goals.



Practice Development

As stated before, the objective of this practice is to demonstrate the usage of cloud computing. For this, Amazon Compute Cloud [EC2] [6] will be the main service in order to achieve this.



Note: All IP's (public and private), along with DNS and similar information it's shown. In a real context, this would suppose a security threat, however, in the current context, this shouldn't be an issue.

Windows Instance

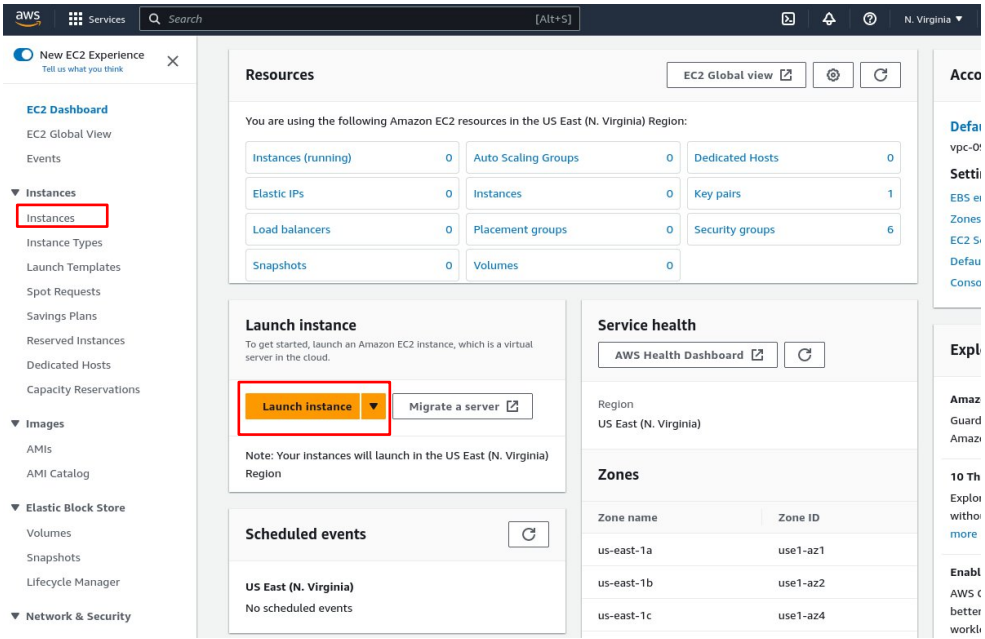
First, a Windows bastion host has to be created and initialized. This instance will interface with following components, which won't be able to reach internet connection.

Before this, to get access to this initial host and for further connections, a pair of keys have to be obtained. In the current academic context, these keys can be accessed at any time through the learning portal, however, in a real environment, these files would be available only once; this is to ensure security at a user level.

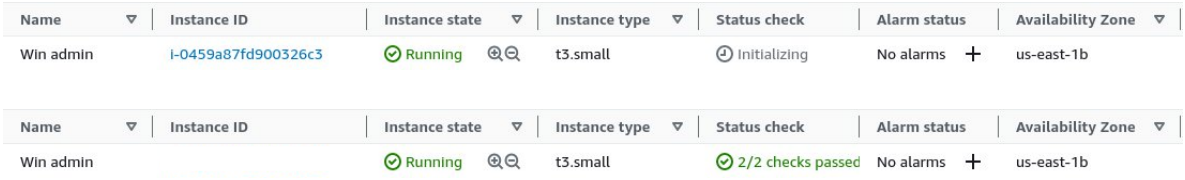
Once this is obtained, the Windows instance has to be made with the following features:

- Name: Win admin
- Operating system: Microsoft Windows Server 2022 Base (latest for free tier)
- Instance type: t3.small
- Login: *vockey* (usage of keys previously downloaded)
- New security group for network
- Storage: 30 Gb (maximum for free tier)
- Volume type: gp2

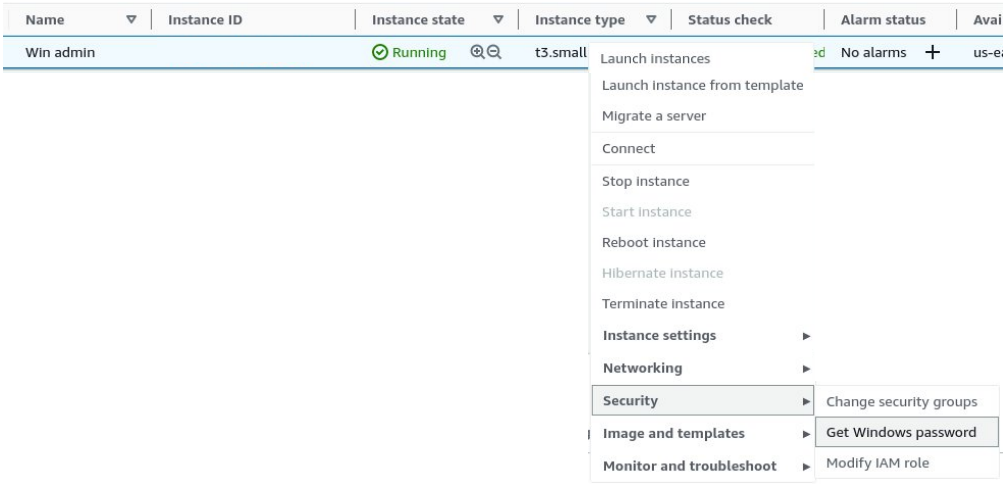
The EC2 panel, accessed through the AWS console, will be used. In this view, multiple information can be spotted, such as the region, availability zones, and, evidently, at least two ways of launching new instances.



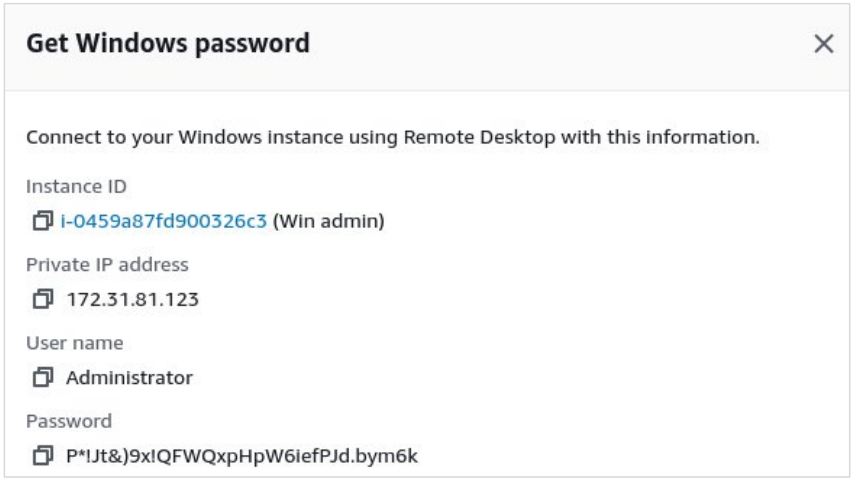
Once the instance it's created, it should be listed inside the instance viewer. The process of resource allocation takes some time, so the first time any instance it's started, it could take a while before it's fully functional.



Once it's running, remote access has to be verified. This instance, being a Windows server instance, can be easily accessed with RDP [Remote Desktop Protocol] [7]. For this to function, the password for this connection has to be obtained; easily achieved directly through instance options.

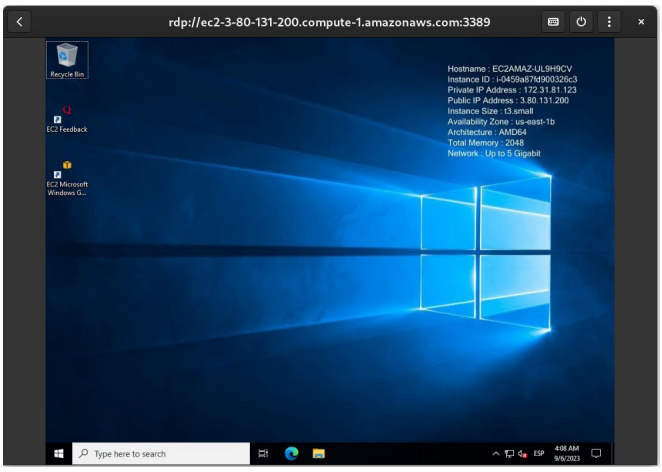


The console will prompt for the private key previously acquired. Once entered, the credentials will be shown.



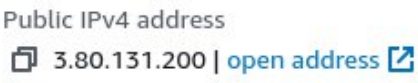
To finally connect to the instance, a remote desktop client has to be used. AWS also offers the fast option to download the RDP client file, which is useful as the public DNS won't always be the same.

Once inside the instance, a normal Windows desktop environment should be available for the user to use as they decide.

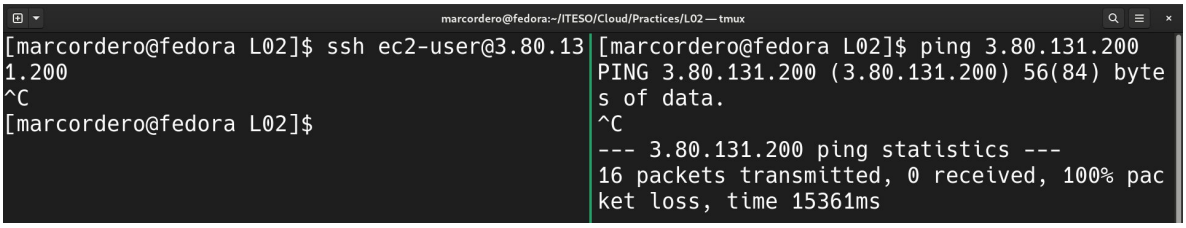


Elastic IP assignation

Consider the following scenario: One should be able to connect to the Windows instance via it's public IPv4 address after rebooting.

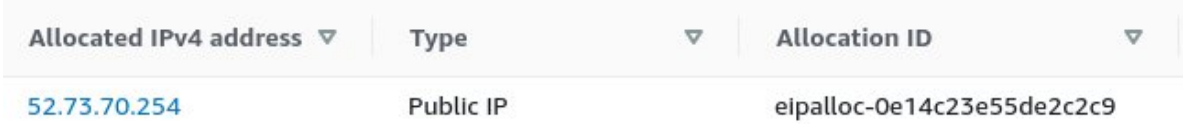


But the following attempts to do so fail, or at least no significant response was sent in relatively long period of time.



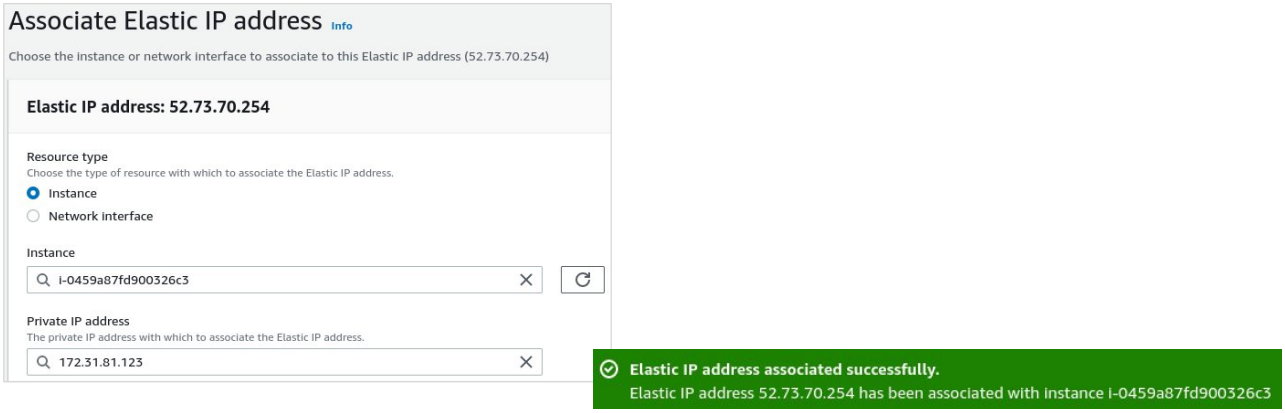
Multiple issues could be occurring here, such as the omission of an identity file, or an non-existent inbound rule for ICMP, but, what shouldn't file it's the RDP connection, as it's already configured from before; however, it would also fail.

To fix this, an elastic IP allocation has to be made. This can be made through the same EC2 section inside the AWS console. This process will take an static address from the Amazon's pool of IPv4 addresses.



Once the address it's allocated, it has to be associated with the bastion host. This can be done within the same section, inside the address actions.

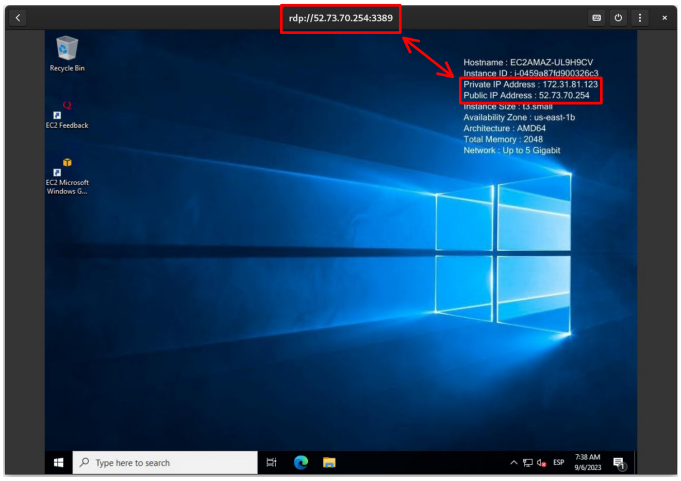
Note: to associate an elastic IP address, the target instance has to be stopped.



As it can be seen, a suggested private IP address has also been assigned. Although this is completely optional [8], it provides with a greater level of detail for a real world architecture.

Note: elastic IP's have no cost *when their associated instance it's running*, otherwise, a small **hourly** fee it's charged. The same applies for non-associated addresses.

If this address assignation wouldn't been done, the previous RDP connection wouldn't work anymore when rebooting the instance, as another IP from AWS Address pool would be assigned to it. Now, if this connection would be a regular action, the same RDP configuration would still be valid.



Linux instance

Once the Windows bastion host has been created and accessed, a subsequent Linux instance without internet access will be configured.

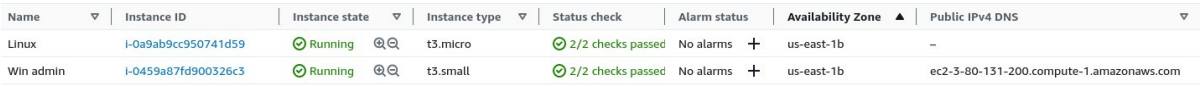
These are its features:

- Name: Linux
- Operating system: Amazon Linux 2 AMI (CentOS) *or* Amazon Linux 2023 (latest) AMI (Fedora) [Selected]

- Instance type: t3.micro
- Login: *vockey* (usage of keys previously downloaded)
- New security group for network
- Storage: 30 Gb (maximum for free tier)
- Volume type: gp2

The key modification for this instance will be the “Auto-assign public IP”change to “Disable”; this will ensure that it won’t have internet access, thus, only allowing communication through SSH. This option can be found at the “Network settings” region, but it would only be visible when editing it.

Now, two instances should be visible.

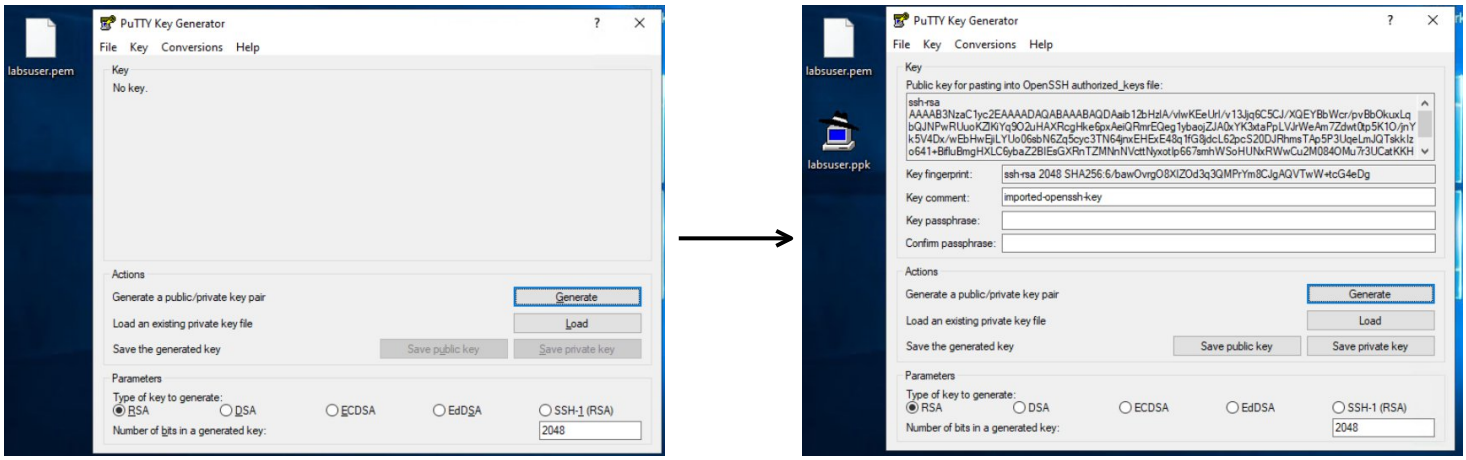


Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
Linux	i-0a9ab9cc950741d59	Running	t3.micro	2/2 checks passed	No alarms	us-east-1b	–
Win admin	i-0459a87fd900326c3	Running	t3.small	2/2 checks passed	No alarms	us-east-1b	ec2-3-80-131-200.compute-1.amazonaws.com

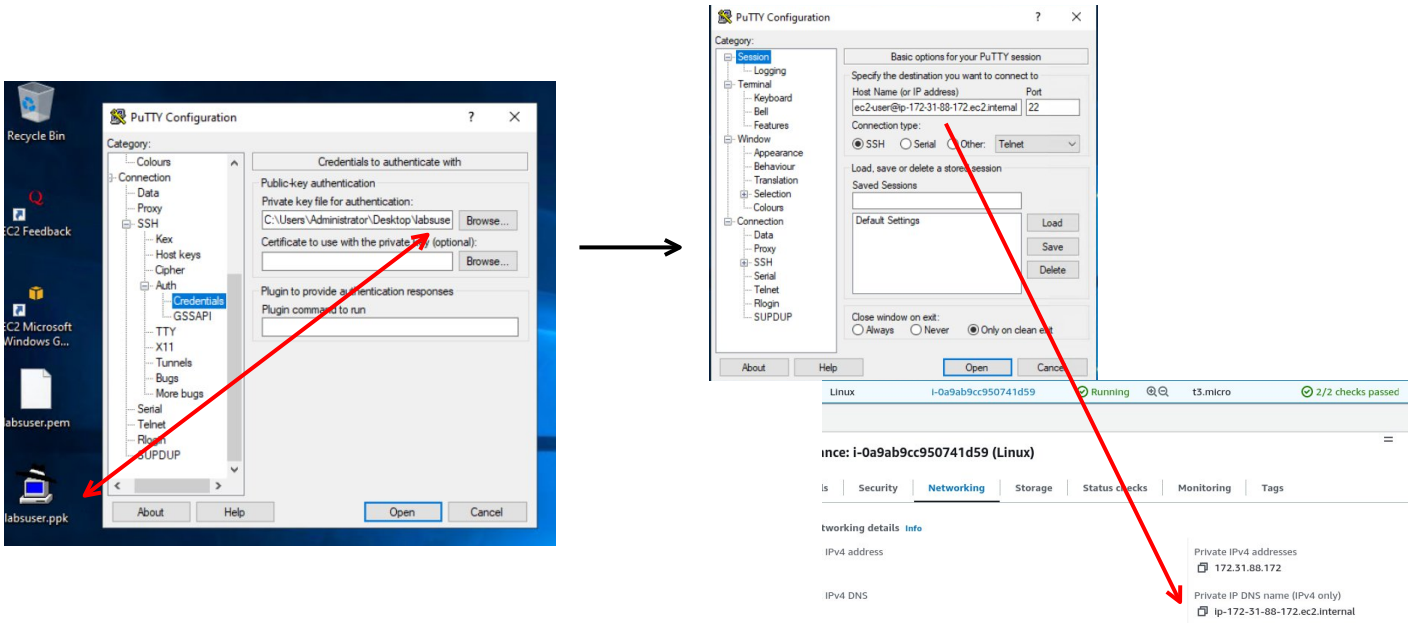
Connection between instances

As previously stated, to be able to access the Linux instance, an SSH connection has to be established from the Windows instance. By default, this operating systems has a built-in SSH client, however, this might be inconvenient for some users, as it’s a pure command-line interface [CLI] tool. To avoid this issue, PuTTY, an SSH client, can be used for ease of control over remote connections.

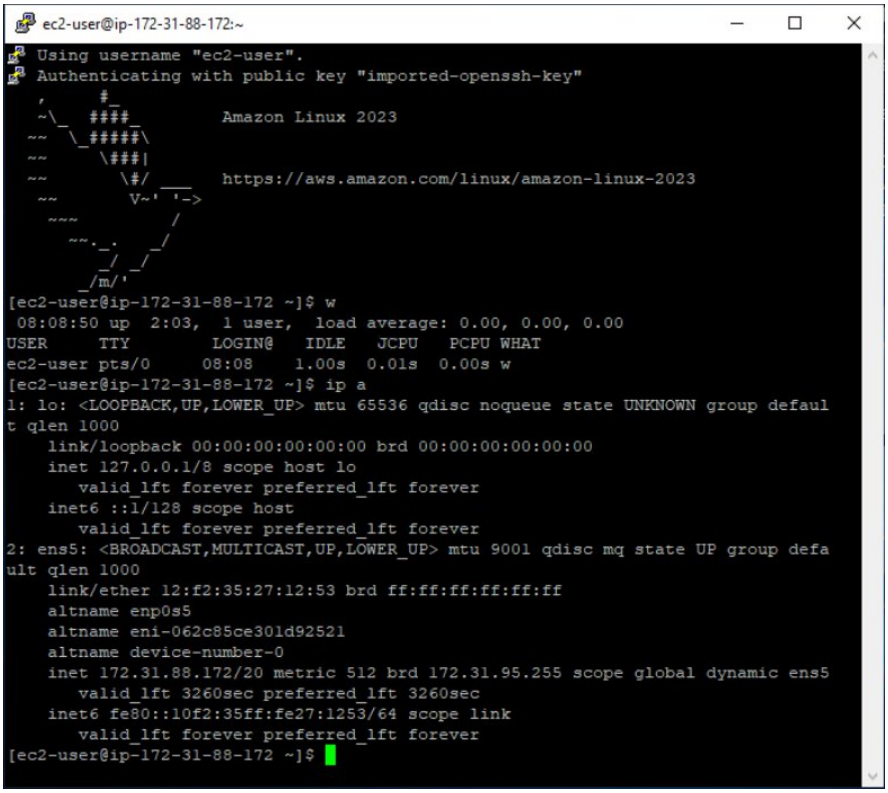
By default, PuTTY doesn’t natively support .pem format provided by AWS, so this file has to be converted to .ppk format. Fortunately, PuTTY comes with PuTTYgen, a tool for key file format conversion.



Afterward, PuTTY can be used to connect via SSH to the Linux instance. First, browse for the recently created private key file and select it. Then, input connection data (user and private dns) to access.

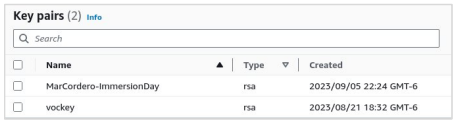


And just like that, SSH connection is established.



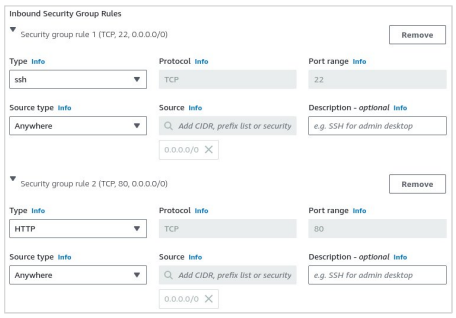
EC2 Hands-On Lab

As an additional exercise for the current activity, a practical laboratory walkthrough will be reviewed. The steps are very similar to the previous instance creation, so they will be only described textually:



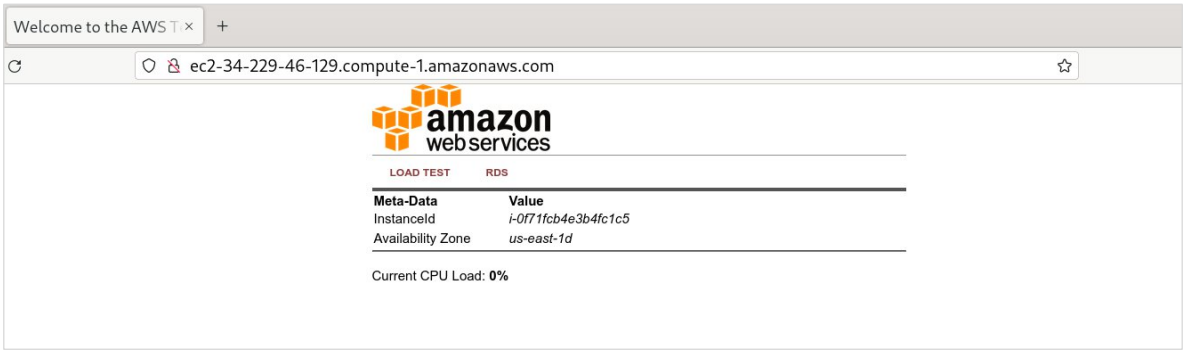
1. Create of a *new* key pair
2. Pick a Web Server Instance (Amazon Linux 2 AMI)
3. Pick t2.micro as instance size
4. Include Apache & PHP install script
5. Create SSH and HTTP security groups
6. Select the new key pair
7. Launch the instance

After the previous steps are made, a web server has been "deployed", meaning that it can be accessed directly through a normal web browser. This can be made by navigating to the public IPv4 DNS provided automatically.



A known issue can and most surely will appear, being this an apparent connection issue, in which the web page supposed to load won't show after a long waiting time. As most things in the IT industry, this can be solved by rebooting the instance.

The web view would look like the following:



The .pem file alteration for read-only permission and SSH connection (made through console) would

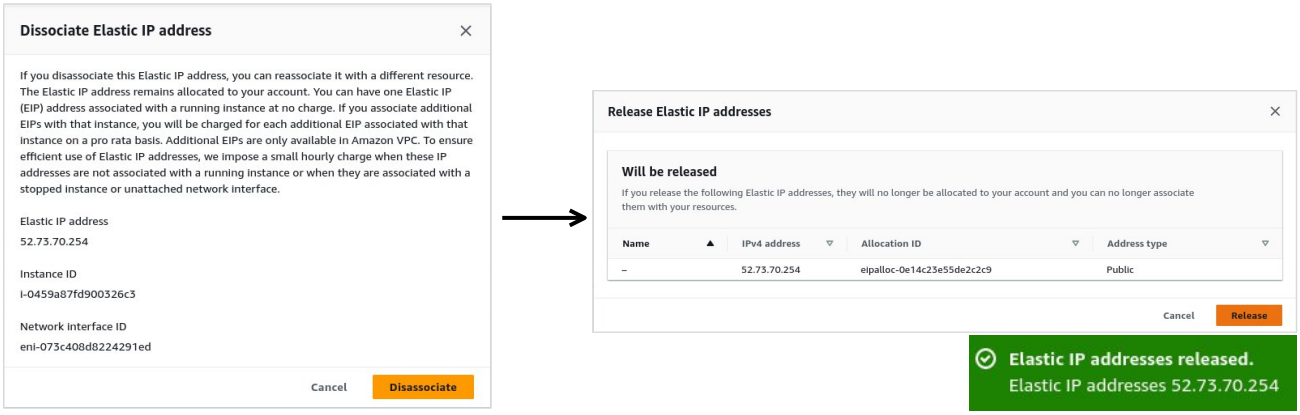
```
[marcordero@fedora L02]$ chmod 400 MarCordero-ImmersionDay.pem
[marcordero@fedora L02]$ ssh -i MarCordero-ImmersionDay.pem ec2-user@34.229.46.129
The authenticity of host '34.      .129 (34.      .129)' can't be established.
ED25519 key fingerprint is SHA256:CsqJfy/OR0qO/3DxPgs9LvAW0ZOI/Yv95EoIJaybCN4.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '34.      .129' (ED25519) to the list of known hosts.
Last login: Wed Sep  6 04:51:52 2023 from fixed-187

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-40-32 ~]$
[ec2-user@ip-172-31-40-32 ~]$ pwd
/home/ec2-user
```

Problems and Solutions

About elastic IP releasing

As seen, an elastic IP had to be allocated for the Windows instance; also, as stated, elastic IP holding would result in its payment when the instance associated it's not in use. Surely this wouldn't cause a problem for a single IP, but, in a real context, this could result in unexpected budget consumption. However, in this context and for unnecessary pay avoidance, is better to release the solicited address. This of course will render unusable the previously made RDP connection.

First, the elastic IP has to be disassociated from its service, in this case, the bastion host instance, which in turn, has to be turned stopped. Then, by accessing its menu, the address can be released.



Elastic IP's fees shouldn't be no more, at least for now.

About file transfer

At some point in development, a .pem file transfer had to be made. Several attempts of achieving this without accessing the AWS console within the bastion host were made to no avail. The ultimate solution was the usage of another cloud service, more specifically a remote file hosting service.

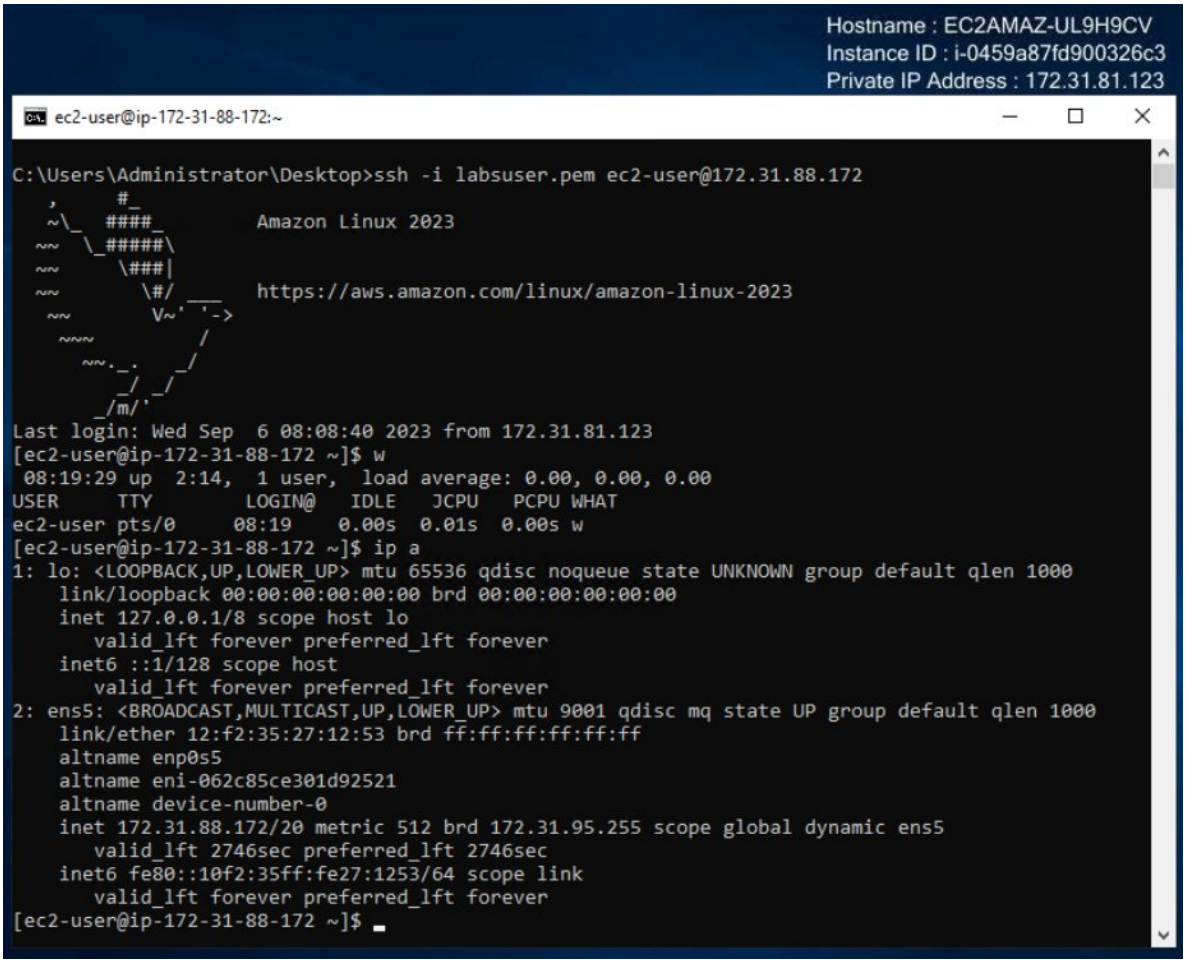
As many other things, this isn't the only way of solving a problem, however, this was the easiest one in this occasion.

Experiments and Results

What if PuTTY wasn't needed?

Often at administrative servers (mostly on Linux ones), graphic environments won't be available by default, leaving the user only with a CLI. In Windows, the closest to a CLI operation environment it's

the “Command Prompt” [CMD], which, as previously mentioned, comes with an SSH client integrated. An added benefit of this it’s the avoidance of .pem file conversion.



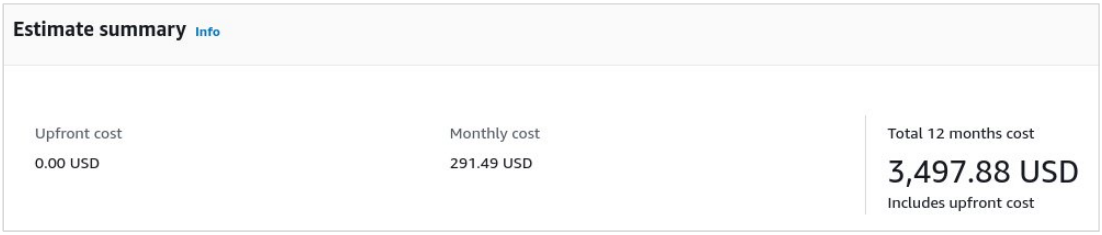
This is just another way of solving the same problem. Even the configurations for this connection can be saved for later usage, as it does RDP clients or like PuTTY with its session saving feature.

What if there was a computer overkill?

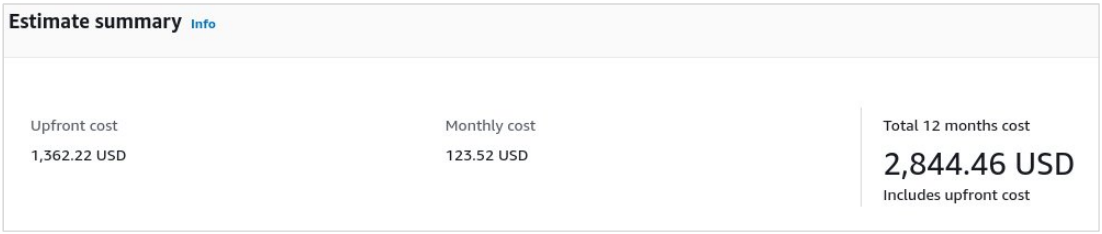
In summary, the costs wouldn’t differ by much.

Consider the following cases: if a single instance was to be configured as a M7i.xlarge with 100 GB storage and 24/7 usage...

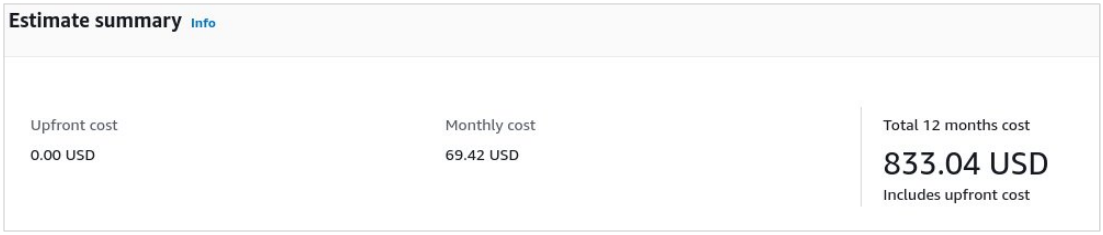
- What would the monthly On-Demand cost be?
- What would the annual On-Demand cost be?



- What would the 1 Year Partial Upfront Reserved cost be?

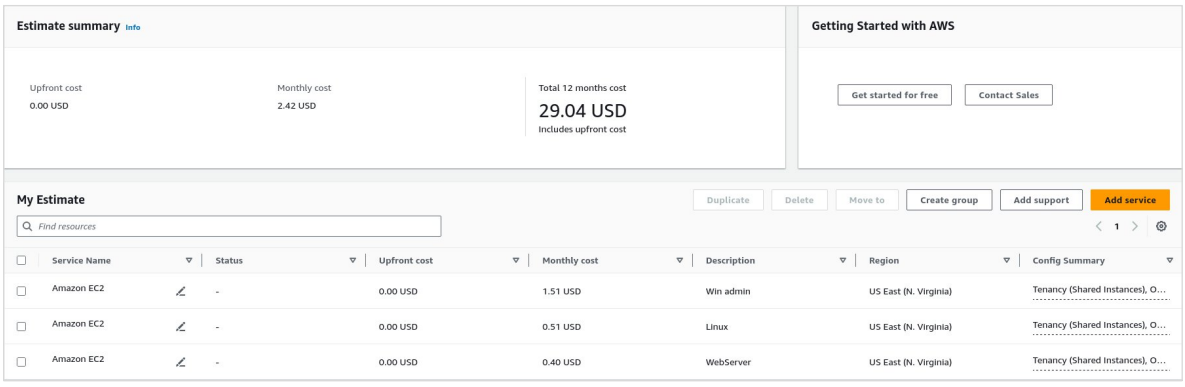


- What would the annual On-Demand cost be, but with 8h/5d usage?



Budget Justification

Consulting over the AWS price calculator, the total implementation for 8 hours a week cost comes to this:



Theoretically, this should be seen as a over the top calculation, as two of the instances shown will be *terminated*, and the other one shouldn't even reach the maximum amount of estimated hours.

Conclusions

Through this practice, several concepts of cloud usage and manipulation has been learned. Most of them were not unknown, however, the process to implement them in a somewhat real environment resulted in such an interesting exercise, and an altogether new experience.

One topic of interest is the amount of time taken to implement the required solution. Approximately 3 hours were taken in total to finish completely the instance deployment, however, as this activity was a first timer, this time would likely reduce in future similar tasks. Nonetheless, this time is but a fraction of an on-premise solution, as the estimated completion time would round between 2 days and up to a week to fully apply security measures, host initialization, license acquirement, and similar requests.

At this point in the career, a broader picture can be seen when dealing with problems, together with a deeper understanding of more concepts and their underlying fundamental concepts. This kind of practice provokes the thought of new solutions, use cases, and scenarios on production, real life implementations that could be achieved by providing a cloud backbone as operative means.

Surely the few instances deployment, along with the concepts that could be analyzed in order to achieve it, will be useful in the near future as the cornerstone of bigger problem solutions.

Bibliography

[1] Aws.amazon.com, 'Cloud computing with AWS'. [Online]. Available: https://aws.amazon.com/what-is-aws.

[2] Sohan Singh Yadav and Zeng Wen Hua, "CLOUD: A computing infrastructure on demand," 2010 2nd International Conference on Computer Engineering and Technology, Chengdu, China, 2010, pp. V1-423-V1-426, doi: 10.1109/ICCET.2010.5486068.

[3] N. Junath and G. Shanmugarathinam, "A Cloud service and conceptual modeling of IAAS," *IET Chennai 3rd International on Sustainable Energy and Intelligent Systems (SEISCON 2012)*, Tiruchengode, 2012, pp. 1-4, doi: 10.1049/cp.2012.2193.

[4] Aws.amazon.com, 'Step 2: Create a Key Pair'. [Online]. Available: https://docs.aws.amazon.com/servicecatalog/latest/adminguide/getstarted-keypair.html.

[5] University College London. 'What is SSH and how do I use it?'. [Online]. Available: https://www.ucl.ac.uk/isd/what-ssh-and-how-do-i-use-it.

[6] Aws.amazon.com, 'Amazon EC2'. [Online]. Available: https://aws.amazon.com/es/ec2/.

[7] Microsoft. 'Understanding the Remote Desktop Protocol (RDP)'. [Online]. Available: https://learn.microsoft.com/en-us/troubleshoot/windows-server/remote/understanding-remote-desktop-protocol.

[8] Aws.amazon.com, 'Elastic IP addresses'. [Online]. Available: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-ip-addresses-eip.html.