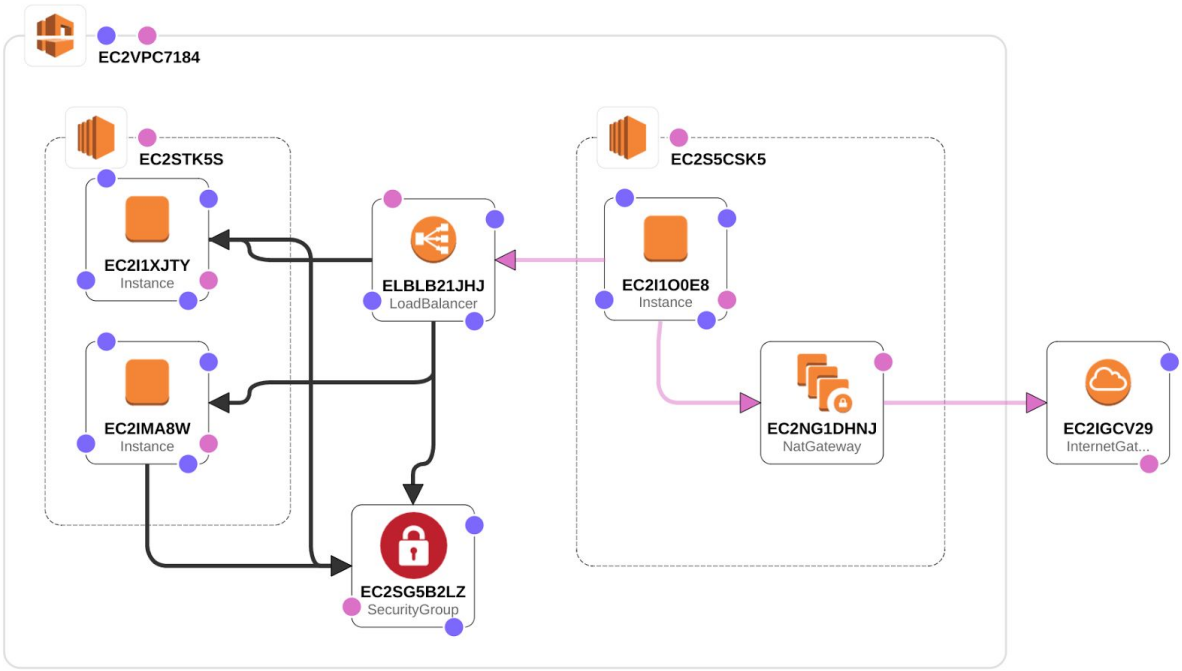
My solution is based on openVPN server installed on an Ubuntu instance. The VM is created inside VPC within a subnet that is connected to the internet. All inbound connections are restricted and only the port 1194 is opened for the openVPN server. The solution is secured because a VPN tunnel is the only way how to reach this instance.

As you can see on the picture below, Ubuntu machine is within the same VPC as the private instances that are running Magical. Therefore the connectivity is guaranteed on a local area network level and the Ubuntu instance is able to establish connection with a Magical service.



John and Joanna can access the instance that is running openVPN server via openVPN client. Their computers will be constantly connected, so it does not matter whether they are at the office or at home.

With this way the instances running inside the private subnet stay secured. If we add an internal load balancer it will be guaranteed that only a healthy instance will be used for the Magical process and at the same time more instances running Magical can be added to the load balancer in order to improve the scalability of the system.

John and Joanna now can establish connection with the Ubuntu instance e.g. using ssh. Rsync command can be used for a transfer of a large file. It is secured as VPN provides a good level of security (also ssh is applied). The file is very large and therefore the transfer must be done in the way that tolerates connection problems.

If a connection or other issue appears, rsync can resume an interrupted transfer. More user friendly solution is mentioned in the bonus point.

File transfer:

-r recurse into directories

-z is compression

-P progress bar and resume of interrupted transfers

rsync -rzP -e "ssh -i /key_pair.pem" filePath user@address:

The file is now located inside the VPN server and it can be easily transferred to the machines that are running Magical. AWS LAN is very fast so it should not be a problem.

Additionally, an encrypted EBS volume can be attached to the VPN server to protect sensitive data inside AWS. There is also a possibility to programmatically attach this EBS volume to the instances that are running Magical.

Magical command can be triggered using ssh or as described in the bonus point.

Alternatively, in order to make the whole process easier, the openVPN server instance can be configured to automatically forward all the traffic into an instance that is running Magical. This was my original idea. The script for iptables configuration is in the folder “port_forwarding”. The only problem is that such solution does not take advantage of the load balancer.

Code:

The code is available here: https://github.com/Ludek2/curve_stack

An AWS CloudFormation template is in the folder aws_stack_template.

The script for a fast openVPN server installation and keys generation is located in the folder “vpn/server”. After successful execution of this script, openVPN server will be running and keys for the clients can be generated using tool easy-rsa located in /etc/openvpn/easy-rsa/.

The script for iptables configuration is in the folder “port_forwarding”. It is useful when using just one VM that is running magical, but if a load balancer is involved, this solution is not relevant. I am including it anyway, just as an additional example of my work.

Possible problems:

- How to avoid full disk and clean large files from an instance when no longer needed, maybe we could write some script for an automatic clean after Magical process is done
- A single openVPN server may be a point of failure. It might be wise to run 2 openVPN servers and connect the clients to both of them, so we can use another server if one instance is unhealthy, ideally each VPN server should be located in a different availability zone

Bonus points:

Automated solution for John and Joanna:

The best way is to run a web server on the instance that is running openVPN server. John and Joanna would be able to reach this web server over a secured VPN channel and upload the file. The server would run the Magical process for them on the instances that are within a private subnet. The only problem would be the size of the file. Some javascript tool could be used e.g. resumable.js. This would be probably a prefered way as the alternative with rsync command is not user friendly.

Safety of sensitive documents on a Mac:

- Strong password, changed regularly
- If the computer is not used the system must logout the user after a short time and ask for a password
- Store sensitive files in an encrypted disk image
- If a sensitive file is deleted, use the “secure empty trash”
- Use FileVault to encrypt a startup disk
- Use open firmware password protection

Auditability and compliance:

- All the details about the infrastructure, methods and tools should be documented