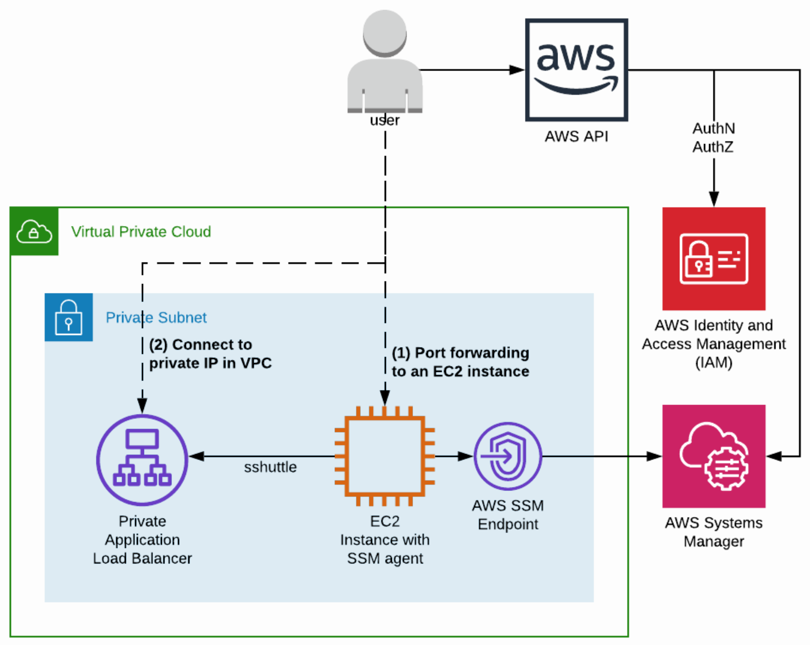
CarriageReturn.Nl About the author

Tunneling into VPC

Jan 3, 2021 by Petri Kallberg

In Where is my bastion host? and EC2 Instance Connect vs. SSM Session Manager I wrote about how to connect to EC2 instances inside a VPC without having to run a bastion host exposed to Internet but using the AWS API as the contact point. While this arrangement has many advantages over running your own server/service there is still one important use-case it can not do.

Above lets you get shell access or forward a port to EC2 instance (1), but connecting to any other resource inside VPC (2) e.g. internal load balancer and RDS database from your desktop isn't possible, or it would require extra trickery such as sshuttle I'm going to explain in this post.


user
AWS API
AuthN
AuthZ
Virtual Private Cloud
Private Subnet
(2) Connect to private IP in VPC
(1) Port forwarding to an EC2 instance
AWS Identity and Access Management (IAM)
sshuttle
Private Application Load Balancer
EC2 Instance with SSM agent
AWS SSM Endpoint
AWS Systems Manager

Some details are left off from the diagram, e.g. ALB requires min. 2 subnets/AZs.

Getting SSH work with Session Manager

Lets start by configuring `ssh` to work over SSM Session Manager connection.

Add the following into `.ssh/config`

```
Host i-*.* mi-*.*
  ProxyCommand bash -c "aws ssm start-session --target $(echo %h|cut -d'.' -f1) -
```

NOTE1: Pre-requisite for this is to have `aws ssm start-session` working as described earlier.
NOTE2: `i-*.*` will match EC2 instance IDs and `mi-*.*` is for managed on-prem VMs.

Now you can use `ssh` to connect instances via Session Manager.

```
% ssh ec2-user@i-abcdef01234567890.eu-central-1
Last login: Wed Dec 30 08:23:41 2020 from localhost

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

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-10-0-0-91 ~]$
```

Above may first seem like complicating things that were already working. It is possible to connect instance with aws -cli and session manager plugin without involving ssh. And that doesn't require local user or ssh key to be deployed either. So why bother?

With above configuration, not just ssh but all tools depending on ssh are now working over the Session Manager connection. E.g You can copy fiels to/from EC2 instances with scp or get Ansible connection without a direct network connection to instance. And as the name suggests, sshuttle is also build upon ssh!

Digging a tunnel into VPC

For a demo, I created an ALB with a HTTP listener returning a fixed HTTP 200 response. I also created Route53 private zone, attached to the VPC, and added an alias `alb.vpc.internal` for the ALB.

The Beauty of sshuttle is that all the installed components will be on on your localhost, the only requirement for the remote host is python interpreter. There is a long list of supported installation methods on the sshuttle Github -page. As long as you have either Linux or MacOS you are covered.

Once you have sshuttle installed, opening the tunnel into VPC can be as simple as

```
% sshuttle --remote ec2-user@i-abcdef01234567890.eu-central-1 --auto-nets
```

Now I can connect to internal ALB from my desktop

```
% dig internal-VpcInternalLB-104985830.eu-central-1.elb.amazonaws.com +short
10.0.0.13
10.0.1.15
% curl http://internal-VpcInternalLB-104985830.eu-central-1.elb.amazonaws.com/
HelloWorld!
```

Finishing touches

Above demo did work only because I used AWS provided public DNS entry for the ALB. To resolve names from private hosted zones, e.g. alb.vpc.internal, you can use sshuttle option `---dns` that will forward the DNS queries through the EC2 instance.

Unless you used your default ssh key for ec2-user when the instance was provisioned you should also specify the private key when opening the tunnel. While there is no specific option in sshuttle for this, you can simply add it to cmd sshuttle is using for connecting to remote host, `--ssh-cmd 'ssh -i /path/to/.ssh/privkey'`

And finally, if `--auto-nets` doesn't detect the correct networks to be tunneled, you can specify addresses and ports manually. It is possible to define both included and excluded ranges.

```
positional arguments:
  IP/MASK[:PORT[-PORT]]...
                        capture and forward traffic to these subnets (whitespace
  ...
optional arguments:
  ...
  -x IP/MASK[:PORT[-PORT]], --exclude IP/MASK[:PORT[-PORT]]
                        exclude this subnet (can be used more than once)
  -X PATH, --exclude-from PATH
                        exclude the subnets in a file (whitespace separated)
```

Sshuttle vs. AWS Client VPN

AWS Client VPN is managed VPN service that does solve the same problem as sshuttle & session manager combination. If you are looking for long-term solution connecting to private resources in VPCs, I would consider AWS Client VPN as tunneling parameters could be centrally managed and overall experiense is more end-user friendly than sshuttle. For ad-hoc development and testing, sshuttle is quicker to setup and provides secure connections into VPC without exposing any servers to internet.

Resources

- sshuttle and AWS Systems Manager Session Manager by Jim Lamb, shows how to use SSH transparently with SSM Session Manager, and how that also enables `sshuttle` to work with Session Manager.
- sshuttle - A VPN for the Lazy is bit more condenced version how to use sshuttle and how it works.
- Using sshuttle as a service by Mike Reider, shows how to run sshuttle as a service and automatically setup regularly used tunnels.
- sshuttle source code in Github.

Did you find this post useful?
Or do you have constructive feedback?