



Network Security: Set up SSM for SSH Access

We'll cover the following ^

- Objective
- Steps
- Set up SSM for SSH access

Objective#


Make our instances inaccessible from the internet.

Steps#

- Set up SSM for SSH access.

In this section, we're going to make our EC2 instances inaccessible from the internet. The instances will be able to reach the internet using a NAT gateway (<https://docs.aws.amazon.com/vpc/latest/userguide/vpc-nat-gateway.html>), but the network will not allow anything from the internet to reach the instances without going through the load balancer.



 Once we make our instances use a NAT gateway to connect to the internet, an additional data transfer charge (currently \$0.045/GB in us-east-1 (<https://aws.amazon.com/vpc/pricing/>)) will apply to all traffic that transits the gateway. This includes traffic to other AWS services.

NOTE: It is very important to note that the prices mentioned above are only for the purpose of understanding cost comparisons. These prices are subject to change anytime by AWS. Therefore, most current prices should be referenced for final business decisions.

To avoid the extra charge, most AWS services can be configured to expose an endpoint that doesn't pass through the internet. This can be done via Gateway VPC Endpoints (<https://docs.aws.amazon.com/vpc/latest/userguide/vpce-gateway.html>) or Interface VPC Endpoints (AWS PrivateLink) (<https://docs.aws.amazon.com/vpc/latest/userguide/vpce-interface.html>).

Set up SSM for SSH access#

One thing that won't work after we lock down our hosts is EC2 Instance Connect. In order to acquire SSH access to our instances, we'll have to use AWS Systems Manager Session Manager (SSM) (<https://docs.aws.amazon.com/systems-manager/latest/userguide/session-manager.html>).

Let's start by adding two new managed policies to the IAM role used by our instances.



```
InstanceRole:
  Type: "AWS::IAM::Role"
  Properties:
    AssumeRolePolicyDocument:
      Version: "2012-10-17"
      Statement:
        Effect: Allow
        Principal:
          Service:
            - "ec2.amazonaws.com"
        Action: sts:AssumeRole
    ManagedPolicyArns:
      - arn:aws:iam::aws:policy/CloudWatchFullAccess
      - arn:aws:iam::aws:policy/service-role/AmazonEC2RoleforAWSCodeDeploy
      - arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
      - arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy
    Policies:
      - PolicyName: ec2DescribeTags
        PolicyDocument:
          Version: 2012-10-17
          Statement:
            - Effect: Allow
              Action: 'ec2:DescribeTags'
              Resource: '*'
    Tags:
      - Key: Name
        Value: !Ref AWS::StackName
```

stage.yml

Line #15 and #16: New policies required to allow SSM access.

Let's check it in, and deploy.

```
git add stage.yml
git commit -m "Add SSM policies to hosts"
git push
```



terminal

```
./deploy-infra.sh
...
```



terminal



Note: All the code has been already added and we are pushing it on our repository as well.

This code requires the following API keys to execute: ^

username	Not Specified...
AWS_ACCESS_KEY_ID	Not Specified...
AWS_SECRET_ACCESS_KEY	Not Specified...
AWS_REGION	us-east-1
Github_Token	Not Specified...

[Edit](#)[Import Values from JSON](#)


```
{
  "name": "aws-bootstrap",
  "version": "1.0.0",
  "description": "",
  "main": "server.js",
  "scripts": {
    "start": "node ./node_modules/pm2/bin/pm2 start ./server.js --name hello_aws",
    "stop": "node ./node_modules/pm2/bin/pm2 stop hello_aws",
    "build": "echo 'Building...'"
  },
  "dependencies": {
    "pm2": "^4.2.0"
  }
}
```

To connect from our local terminal, we can install the SSM plugin for the AWS CLI (<https://docs.aws.amazon.com/systems-manager/latest/userguide/session-manager-working-with-install-plugin.html>) while the CloudFormation changes are deploying.



```
$ curl "https://s3.amazonaws.com/session-manager-downloads/plugin/latest/mac/sessionmanager-bundle.zip"
...
$ unzip sessionmanager-bundle.zip
...
$ sudo ./sessionmanager-bundle/install -i /usr/local/sessionmanagerplugin -b /usr/bin
```

terminal

 Sometimes it can take a while for the updated IAM role to take effect on instances that are already running. If your instances are failing to connect via SSM, you can try terminating your instances and letting the ASG replace them with fresh ones.

We should now be able to open a shell on a remote host with the AWS CLI. Once the connection goes through, we are connected as `ssm-user`, not our familiar `ec2-user`. So before we do anything else, we must switch to `ec2-user` and change into the appropriate home directory.

```
$ aws ssm start-session --profile awsbootstrap --target i-07c5a5b5907d43ca7
Starting session with SessionId: josh-0024a17ad7747b5e6
sh-4.2$ sudo su ec2-user
[ec2-user@ip-10-0-192-31 bin]$ pwd
/usr/bin
[ec2-user@ip-10-0-192-31 bin]$ cd ~
[ec2-user@ip-10-0-192-31 ~]$ pwd
/home/ec2-user

[ec2-user@ip-10-0-52-140 ~]$ # admin commands here

[ec2-user@ip-10-0-52-140 ~]$ exit
exit
sh-4.2$ exit
exit

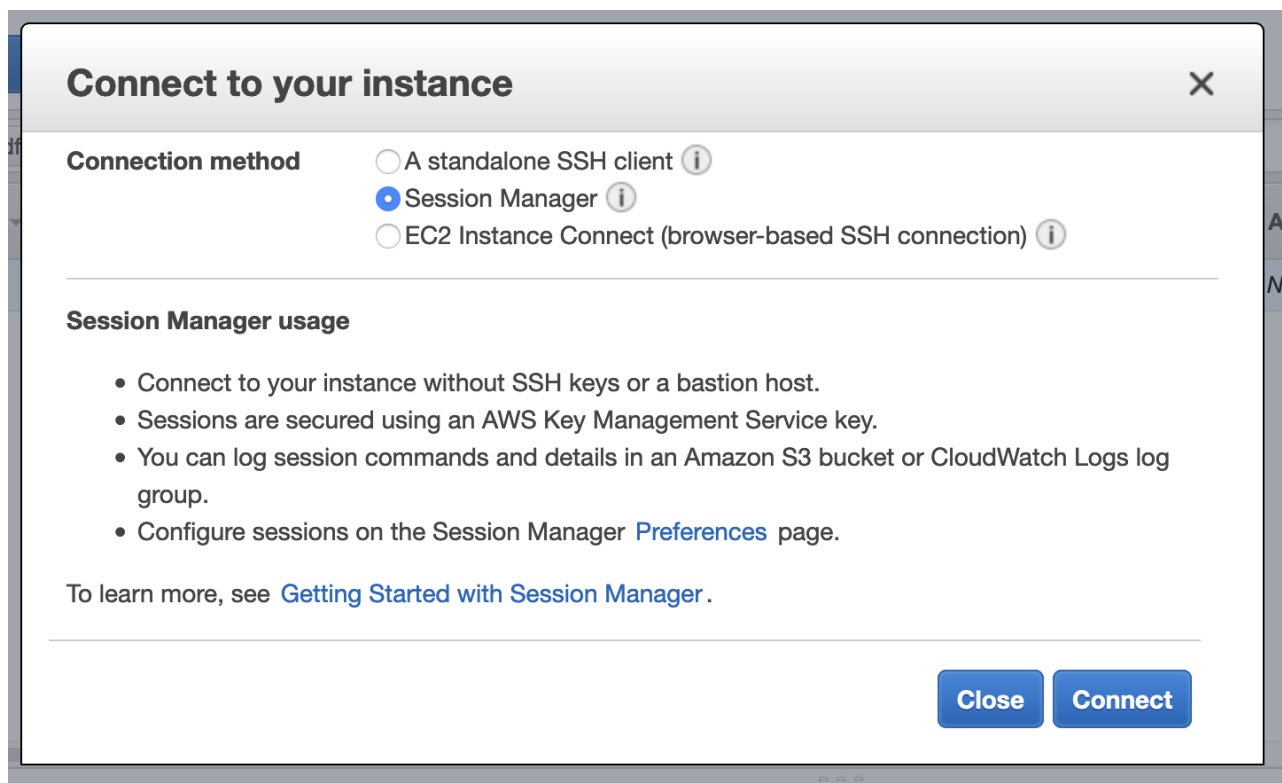
Exiting session with sessionId: josh-0024a17ad7747b5e6.
```

terminal



Line #1: Replace `i-07c5a5b5907d43ca7` with an instance ID from your fleet.


We can also continue to use the AWS console to SSH to our instances. But now we have to choose *Session Manager* instead of *EC2 Instance Connect*.

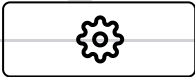


SSM Connection

In the next lesson, we will add private subnets and NAT Gateway for our application.

[← Back](#)[Next →](#)[HTTPS: Make the Application Speak H...](#)[Network Security: Add Private Subnet...](#)

 Mark as Completed



Report an Issue