VPC SSM Endpoints

This post is the second in a series about alternatives to bastion hosts in each of the major cloud providers. Although there are other methods for accessing EC2 instances in AWS, Session Manager is the best match for our requirements, which are:

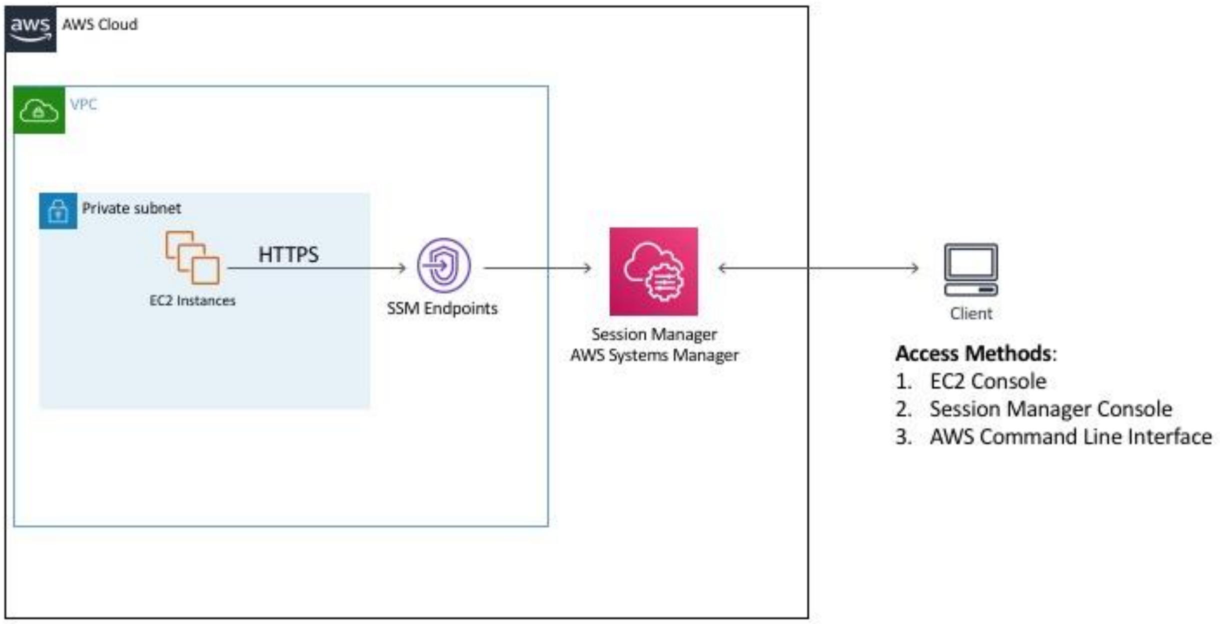
- Does not require public IP addresses for the VMs.
- Eliminates the possibility of SSH multiplexing attacks.
- Consolidates access (such as SSH or RDP) to IAM credentials.
- Captures metadata and full session logs (when possible) for remote access.

The Session Manager service is provided as a component of AWS Systems Manager. To use Session Manager, you must set up AWS Systems Manager. While Session Manager supports Linux and Windows instances, we will focus on using it with Elastic Compute Cloud (EC2) Linux instances running in AWS. If you have a hybrid environment, which also contains VMs in your own facility, then you can still use this solution with some additional steps that we do not cover in this post.

Why AWS Session Manager?

Session Manager can be used to access instances within private subnets that allow no ingress from the internet. This is made possible by the Systems Manager (SSM) agent running on the EC2 instances, which pushes traffic to the Session Manager service. The configuration shown in the diagram below provides an example of how it can be configured, which will be explained in more detail in the sections below.

The figure shows that the EC2 instances are not at all accessible from the internet, even by our authorized client. The SSM agent starts the session for us, and pushes traffic to the interface endpoints we’ve created in the VPC for the SSM services. The Session Manager service acts as the intermediary, providing the client with a shell for an EC2 instance.



The client machine is outside of AWS, and is being used by someone with valid user credentials for the AWS environment. The client can create a session in two different parts of the AWS console, or via the AWS Command Line Interface (CLI). It is possible to restrict how users launch sessions, so you could choose to only allow them to invoke a connection via the AWS CLI (which requires the installation of the Session Manager Plugin) if your users don't have the ability to log into the console, or the other way around.

AWS SSM provides the ability to establish a shell on your systems through its native service, or by using it as a tunnel for other protocols, such as Secure Shell (SSH). The advantages of using SSM without tunneling SSH are:

- It will log the commands issued during the session, as well as the results. If you use SSH within Session Manager, then this will all be encrypted.
- No need to manage SSH keys.
- Shell access is completely contained within Identity and Access Management (IAM) policies, so there is one central choke point to control that access.

Now that you have an idea of how AWS SSM works in general, we'll take a closer look at it by breaking down the details into the following categories:

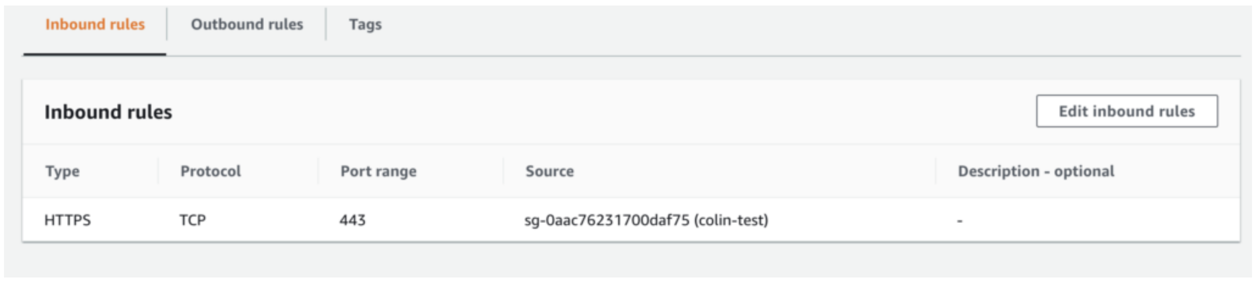
1. Networking Considerations

2. Virtual Machine Configuration
3. Identity Management
4. Logging

Networking Considerations

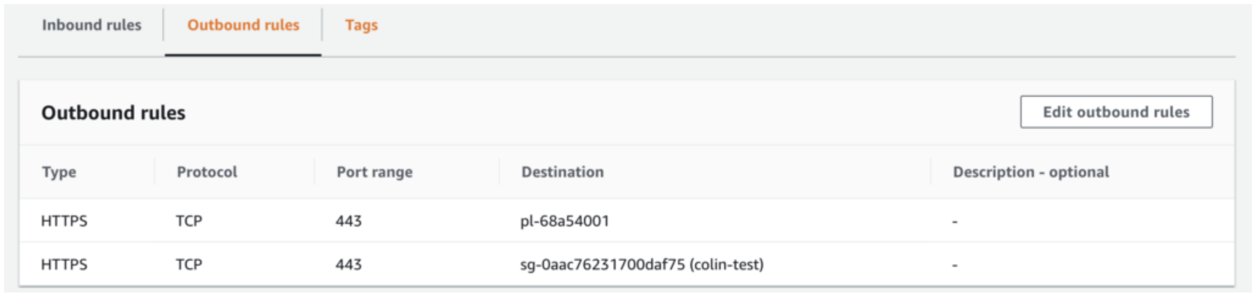
As shown in our diagram above, the SSM agent actually uses HTTPS. The amount of network traffic that needs to be allowed is minimal. If you follow the steps below, then it is not necessary to open port 22 to ingress traffic, even within the VPC:

1. Create VPC interface endpoints in the target VPCs. This makes use of AWS PrivateLink, so the connection traffic between the target EC2 instances and the Session Manager service does not traverse the internet. Create the endpoints for the following service names (substitute your own region in the names below, such as 'us-west-2'):
   - ***com.amazonaws.[region].ssm***
   - ***com.amazonaws.[region].ssmmessages***
2. Configure the Security Groups and Network ACLs that protect the target EC2 instances to allow HTTPS egress traffic (443) from the instances to the Systems Manager endpoints. It is not necessary to allow any ingress traffic to the EC2 instances. The way we recommend to configure this is to add the endpoints and EC2 instances to a security group, and only allow the traffic within the members of that security group:



Inbound rules

Type	Protocol	Port range	Source	Description - optional
HTTPS	TCP	443	sg-0aac76231700daf75 (colin-test)	-



Outbound rules

Type	Protocol	Port range	Destination	Description - optional
HTTPS	TCP	443	pl-68a54001	-
HTTPS	TCP	443	sg-0aac76231700daf75 (colin-test)	-

The configuration above shows that the security group allows HTTPS between network interfaces that are members of the security group. In addition, it allows outgoing HTTPS only to the prefix list (pl-68a54001), which contains our gateway to S3

Virtual Machine Configuration

Complete the prerequisites

The prerequisites for Systems Manager must be satisfied on each EC2 instance, which includes installing the Systems Manager (SSM) agent. The Amazon Linux images already include the agent, and you can install it manually on other Linux systems. The agent allows a lot more than just connecting with Session Manager. More information about its capabilities is available here. If you want to use it on a system that is not currently supported, you can always download the code and modify it yourself. However, AWS will not support a modified version of the agent.

Grant instances access to AWS

Each EC2 instance must have permission to make certain API calls to the Session Manager Service. All necessary permissions are available in the Amazon managed IAM policy, **AmazonSSMManagedInstanceCore**. However, if you only want to apply the permissions necessary for Session Manager by creating a custom profile, there is more information on how to do that here.

Identity Management

The Systems Manager agent will create a local user on both Windows and Linux-based EC2 instances, called ‘ssm-user’. If the user is not created, it may be due to incorrect permissions applied to the instance profile. The ssm-user will be added to the sudoers file in Linux and added to the Administrators group on Windows instances. Since the ssm-user is a privileged user, AWS recommends that you further restrict the permissions around using Systems Manager. Session access can actually be restricted using instance tags and the commands can be restricted within a SSM document.

Using this method to access the instances does not require any management of SSH keys. Since everything is managed through the Session Manager, access is only based on IAM policies. When configuring Session Manager access for your end-users, you can limit which instances they access, whether it’s through the command line or the console, and if they are allowed to tunnel SSH or just use the regular Session Manager shell access. More information about how to configure the IAM policies is available here. Some of the AWS policy templates show how to restrict access to specific EC2 instances, but that will probably not be a scalable approach if you really want to implement this for your organization. Instead, you can refer to templates available here to see how to configure the policies to allow access based on tags. You will also want to make sure you add permissions for users to terminate only the sessions they started, as shown here. Otherwise, a user may be able to terminate another user's session.

Logging

CloudTrail

Without configuring anything, AWS CloudTrail will collect basic information about sessions. When someone launches a remote access session with Session Manager, SSM will log an event named, "StartSession." This event will include a number of interesting things, such as:

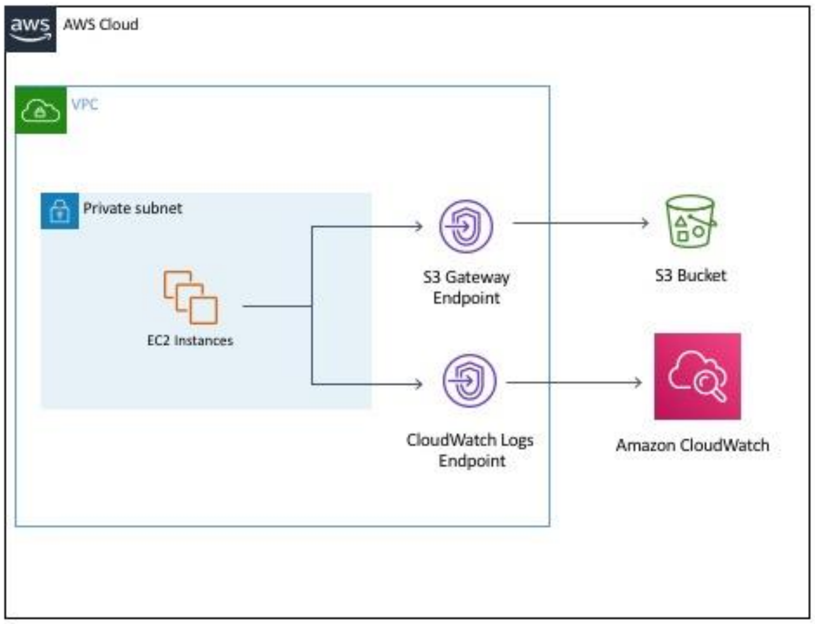
- The username that launched the session
- In some cases, whether the user was authenticated with multi-factor authentication (MFA)
- The originating IP address
- The unique ID of the target EC2 instance
- The session ID

Additional information about what events are logged by CloudTrail is available here. The events in CloudTrail are useful for getting a sense of who's logging into your instances and when it's happening. However, it does not provide you with any information about what they are doing once they've established a session.

Session data logs

If you are interested in capturing the commands issued during each session, it is possible to do this directly from Session Manager. As mentioned in the overview of AWS Session Manager, SSM uses HTTPS to establish sessions, but it is also possible to use it as a tunnel for other protocols, such as SSH or RDP. Be aware that if you choose to use the tunneling option, then AWS cannot provide full session logs for those connections. The sections below will cover the case where you are using SSM directly and full session logs are available.

The diagram below shows an example configuration for logging full session data either to CloudWatch or S3.



Session data in CloudWatch

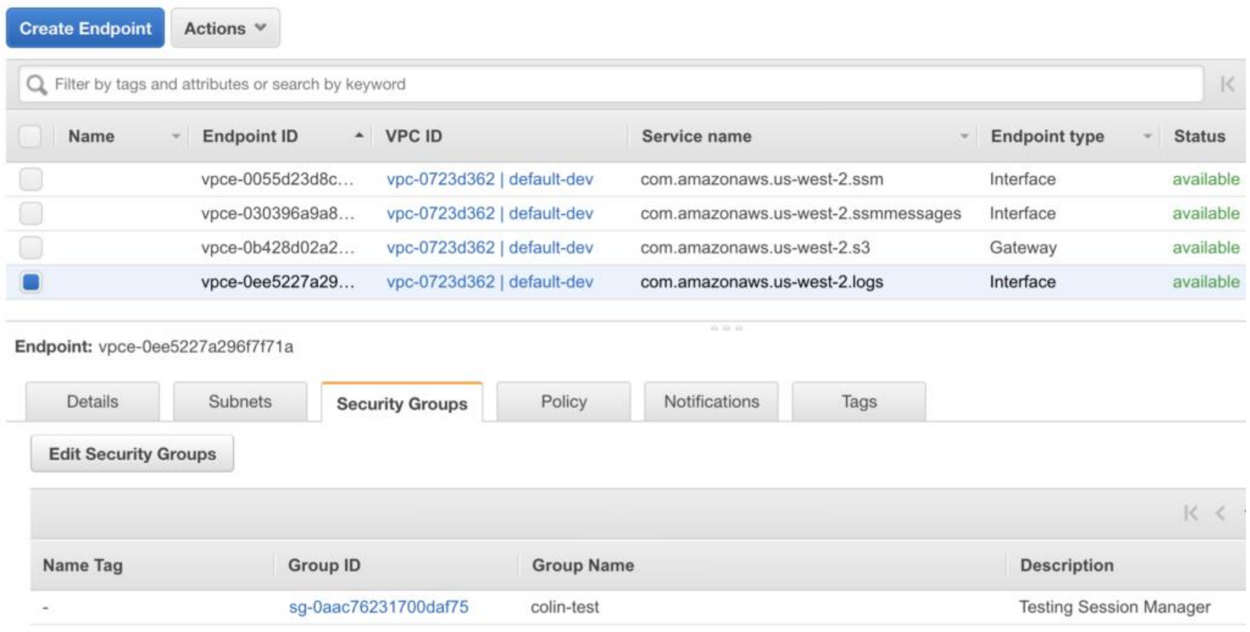
There are a couple of methods that can be used to save session logs to CloudWatch. One requires very little additional configuration by using the SSM agent to send logs to CloudWatch. Although this is the easiest method, it's being deprecated by AWS. The other method is to install and configure the CloudWatch agent on your EC2 instances to send the logs to CloudWatch. In the sections below we'll cover the prerequisite steps necessary no matter which agent you use, and then cover each method in more detail.

Universal steps for CloudWatch

The first step is to select a CloudWatch log group to use for session data. If you don't already have one to use, you must create a log group.

Depending on your network configuration, you may need to use a VPC endpoint for the CloudWatch Logs service. We've chosen to use an endpoint (shown in the diagram above). The endpoint for CloudWatch Logs has the following name (substitute your own region in the name below, such as 'us-west-2'): ***com.amazonaws.[region].logs***

Once the endpoint has been created, it can be added to a security group with the EC2 instances that allow HTTPS traffic (the security group rules are shown in the Network Considerations section). No modifications are required to the security group to allow any additional traffic, and now your EC2 instances will be able to send logs to CloudWatch. The screenshot below shows the endpoint in our VPC console, and the fact that it's a member of our security group:



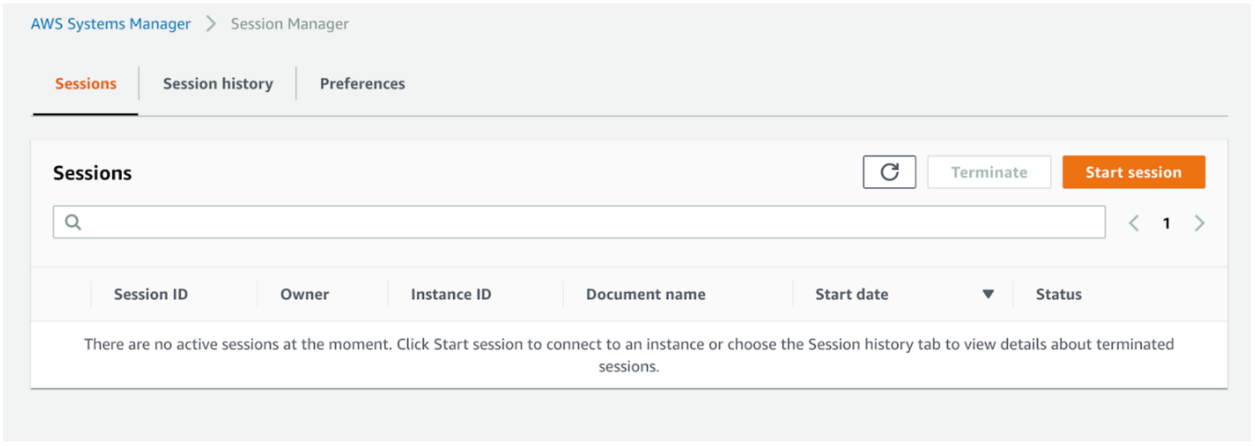
Name	Endpoint ID	VPC ID	Service name	Endpoint type	Status
	vpce-0055d23d8c...	vpc-0723d362 | default-dev	com.amazonaws.us-west-2.ssm	Interface	available
	vpce-030396a9a8...	vpc-0723d362 | default-dev	com.amazonaws.us-west-2.ssmmessages	Interface	available
	vpce-0b428d02a2...	vpc-0723d362 | default-dev	com.amazonaws.us-west-2.s3	Gateway	available
	vpce-0ee5227a29...	vpc-0723d362 | default-dev	com.amazonaws.us-west-2.logs	Interface	available

Endpoint: vpce-0ee5227a296f7f71a

Name Tag	Group ID	Group Name	Description
-	sg-0aac76231700daf75	colin-test	Testing Session Manager

Once you have CloudWatch set up, you must set the Session Manager Preferences to use the log group:

1. Go to the Systems Manager section of the AWS Console, and select "Session Manager". It will look like this:

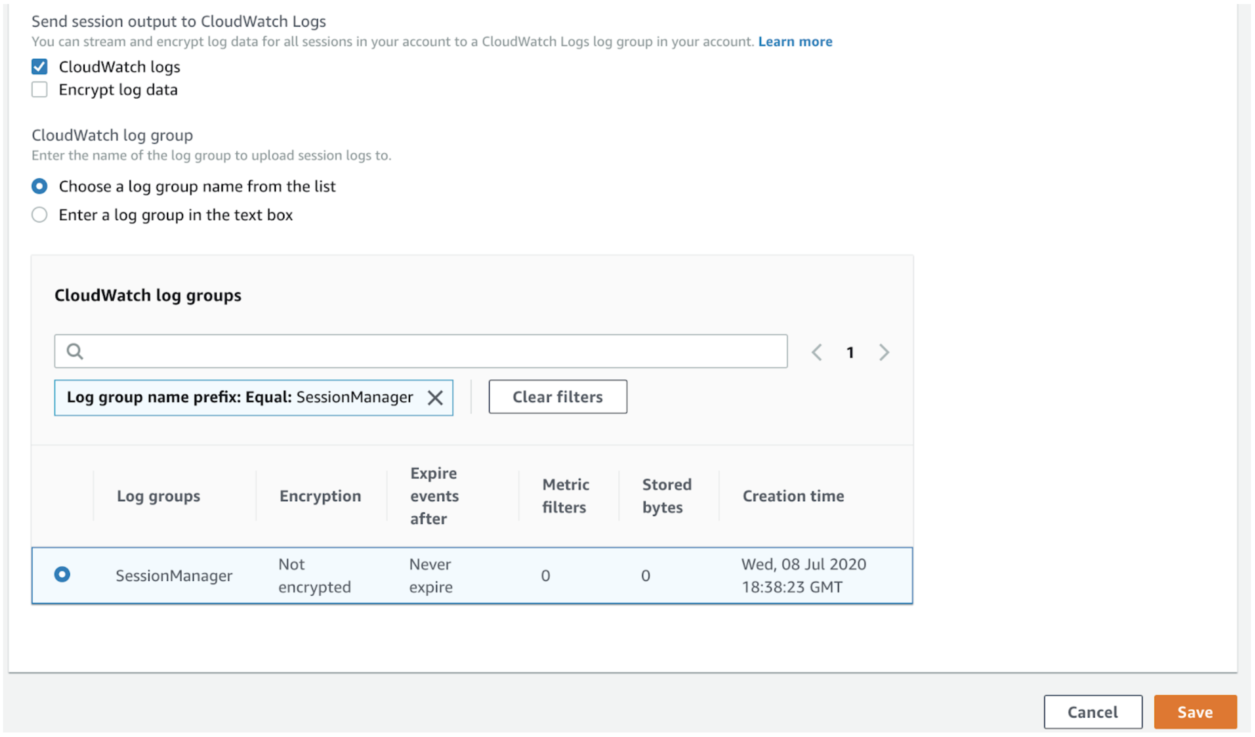


Session ID	Owner	Instance ID	Document name	Start date	Status

There are no active sessions at the moment. Click Start session to connect to an instance or choose the Session history tab to view details about terminated sessions.

2. Select the "Preferences" tab and click the "Edit" button.

3. Under the "Send session output to CloudWatch Logs" section, check the box that says "CloudWatch logs."
4. Under the "CloudWatch log group" section, select the name of the log group you will use for session data. It should look something like this:



Send session output to CloudWatch Logs

You can stream and encrypt log data for all sessions in your account to a CloudWatch Logs log group in your account. Learn more

[✓] CloudWatch logs

[ ] Encrypt log data

CloudWatch log group

Enter the name of the log group to upload session logs to.

● Choose a log group name from the list

○ Enter a log group in the text box

CloudWatch log groups

Log group name prefix: Equal: SessionManager

	Log groups	Encryption	Expire events after	Metric filters	Stored bytes	Creation time
●	SessionManager	Not encrypted	Never expire	0	0	Wed, 08 Jul 2020 18:38:23 GMT

5. Click the "Save" button.
6. Now, we need to give the EC2 instances enough permissions to write events to the CloudWatch Log group. This documentation shows the IAM permissions that should be granted to the instance profile in order to allow SSM to write session logs to CloudWatch or an S3 bucket.