**😊 AWS Session Manager**

AWS Systems Manager Session Manager provides secure and auditable remote shell access to your instances within AWS without the need for SSH or RDP (Remote Desktop Protocol) connections. It allows you to start interactive sessions with your instances directly from the AWS Management Console or through the AWS CLI, SDKs, or APIs.

Here are some key features and use cases of AWS Systems Manager Session Manager:

1. **Secure Access:** Session Manager uses AWS Identity and Access Management (IAM) policies to control access to instances, ensuring that only authorized users can initiate sessions. It eliminates the need for managing SSH keys or exposing RDP ports to the internet, reducing the attack surface.
2. **Auditability:** All session activity is logged to AWS CloudTrail, providing a comprehensive audit trail of who accessed which instances, when, and what commands were executed. This helps with compliance and security auditing requirements.
3. **Encryption:** Session Manager encrypts all session data using TLS (Transport Layer Security), ensuring that communication between the client and the instance remains secure.
4. **Fine-Grained Access Control:** IAM policies can be used to granularly control which users or roles have permission to start sessions with specific instances. You can also restrict the commands that users are allowed to execute during sessions.
5. **Centralized Management:** Session Manager is integrated with AWS Systems Manager, providing a centralized location to manage all your instances, including EC2 instances, on-premises servers, and virtual machines.
6. **No Bastion Hosts:** Session Manager eliminates the need for bastion hosts or jump servers, simplifying network architecture and reducing management overhead.
7. **Session Logging and Recording:** You can enable session logging and recording to capture session output, including commands entered and responses received, for further analysis or troubleshooting.
8. **Access to Instances in Private Subnets:** Session Manager can be used to access instances located in private subnets without requiring direct internet connectivity or the need for a bastion host.

**😁 Use cases of Session Manager**

AWS Systems Manager Session Manager provides secure and auditable remote shell access to your instances within AWS without the need for SSH or RDP (Remote Desktop Protocol) connections. It allows you to start interactive sessions with your instances directly from the AWS Management Console or through the AWS CLI, SDKs, or APIs.

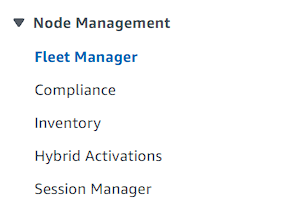
Here are some key features and use cases of AWS Systems Manager Session Manager:

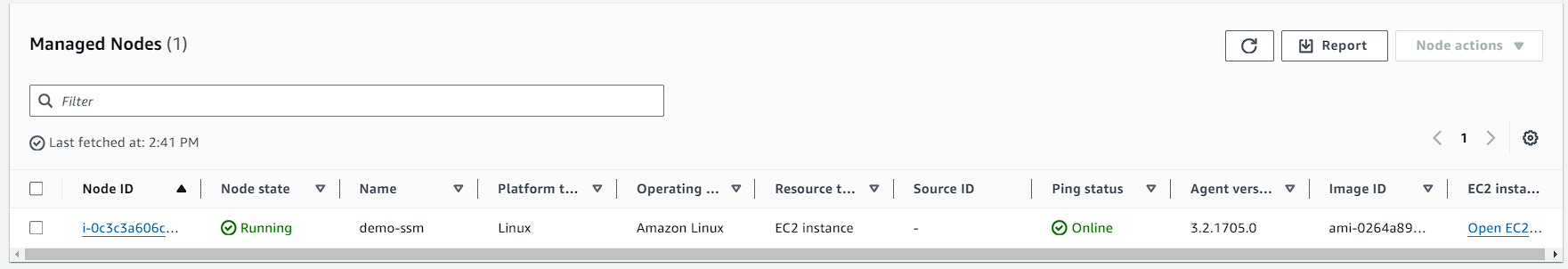
1. **Secure Access:** Session Manager uses AWS Identity and Access Management (IAM) policies to control access to instances, ensuring that only authorized users can initiate sessions. It eliminates the need for managing SSH keys or exposing RDP ports to the internet, reducing the attack surface.
2. **Auditability:** All session activity is logged to AWS CloudTrail, providing a comprehensive audit trail of who accessed which instances, when, and what commands were executed. This helps with compliance and security auditing requirements.
3. **Encryption:** Session Manager encrypts all session data using TLS (Transport Layer Security), ensuring that communication between the client and the instance remains secure.
4. **Fine-Grained Access Control:** IAM policies can be used to granularly control which users or roles have permission to start sessions with specific instances. You can also restrict the commands that users are allowed to execute during sessions.
5. **Centralized Management:** Session Manager is integrated with AWS Systems Manager, providing a centralized location to manage all your instances, including EC2 instances, on-premises servers, and virtual machines.
6. **No Bastion Hosts:** Session Manager eliminates the need for bastion hosts or jump servers, simplifying network architecture and reducing management overhead.
7. **Session Logging and Recording:** You can enable session logging and recording to capture session output, including commands entered and responses received, for further analysis or troubleshooting.
8. **Access to Instances in Private Subnets:** Session Manager can be used to access instances located in private subnets without requiring direct internet connectivity or the need for a bastion host.

**😁 To begin with the Lab:**

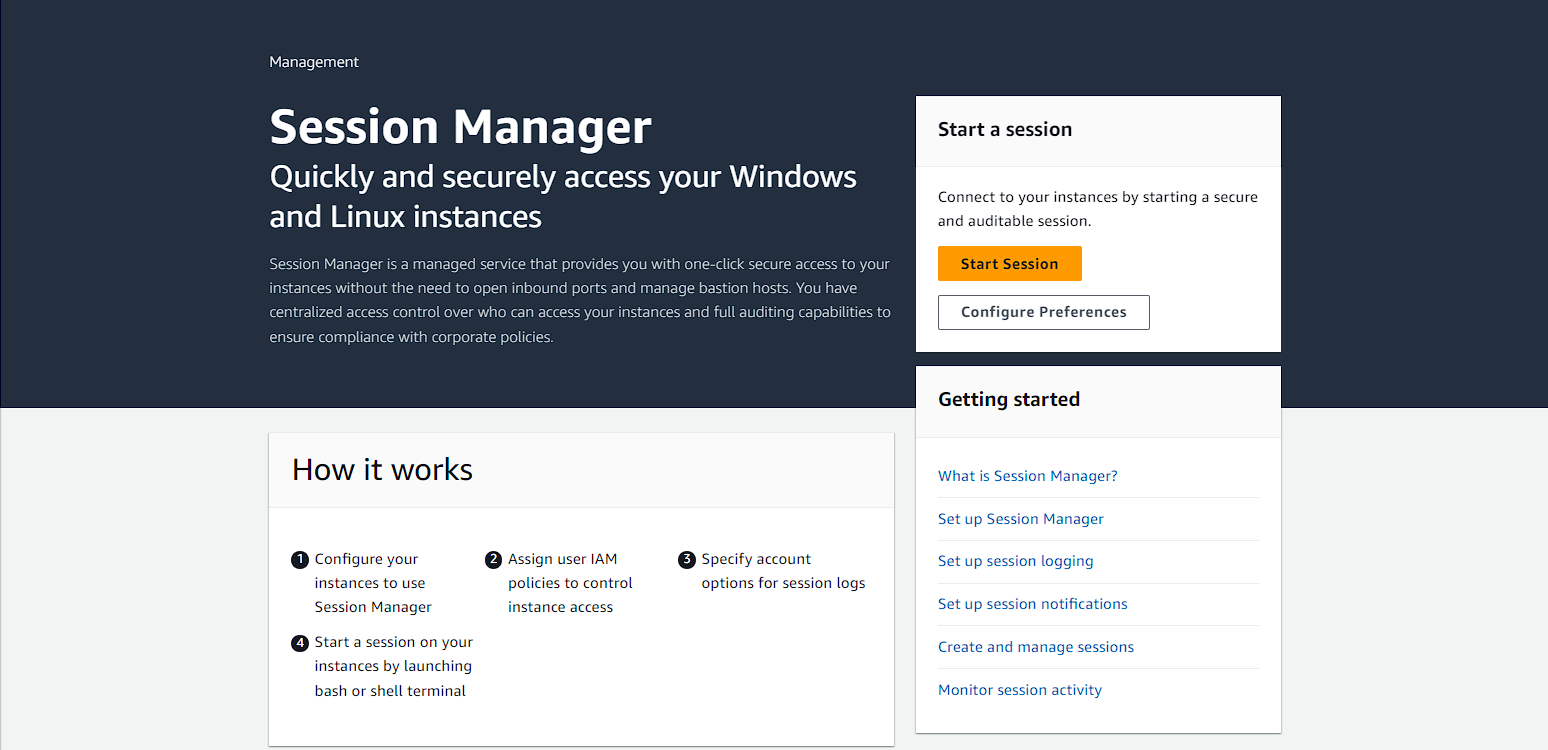
**😎 Step 1: Session Manager**

1. In AWS Console navigate to Systems Manager. But before that first go to fleet manager and check that your instance is up and running.

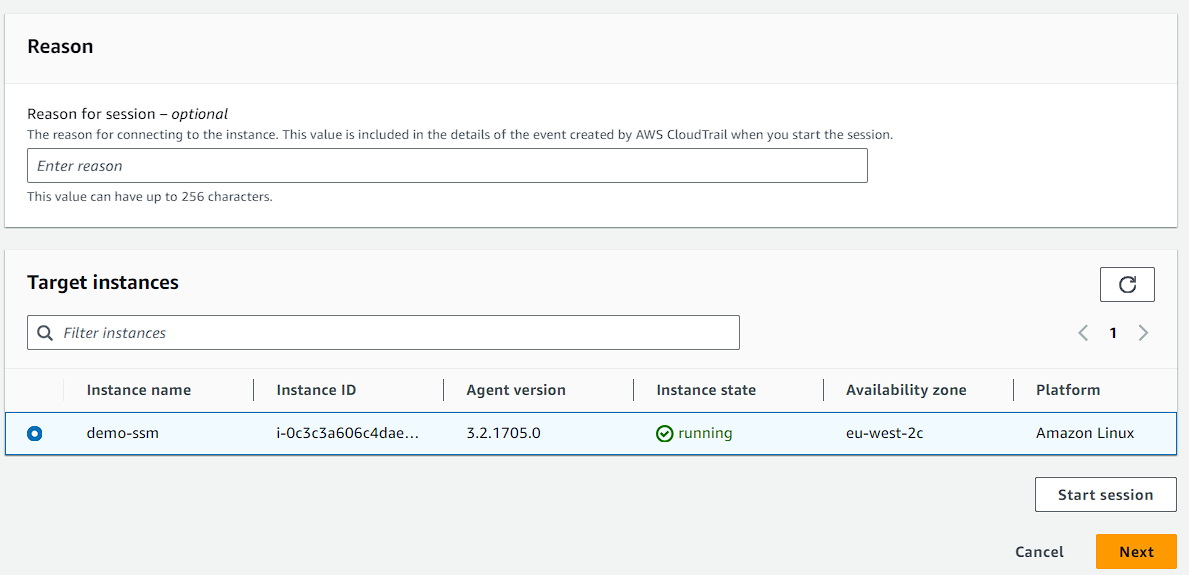




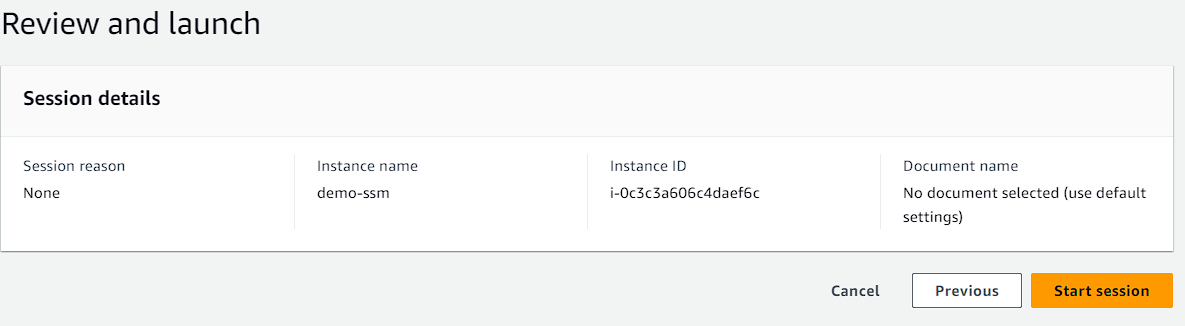
1. From the dashboard of session manager, you need to click on Start Session.



1. Now you need to select your instance and click on next.



1. Then jump to Start session and start it.



1. Here you can see that you are in the session manager.



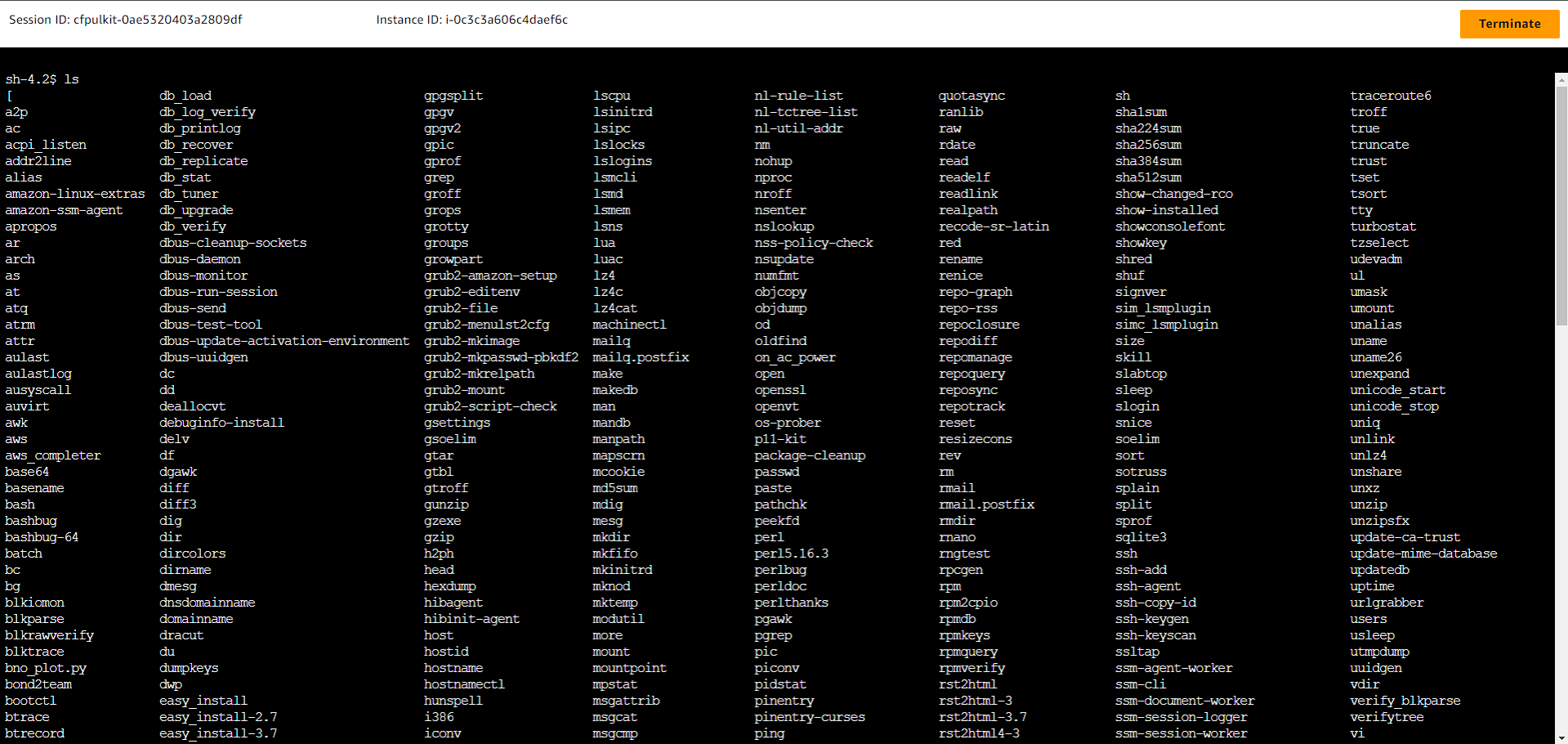
1. Here you run some commands you can see all the information.

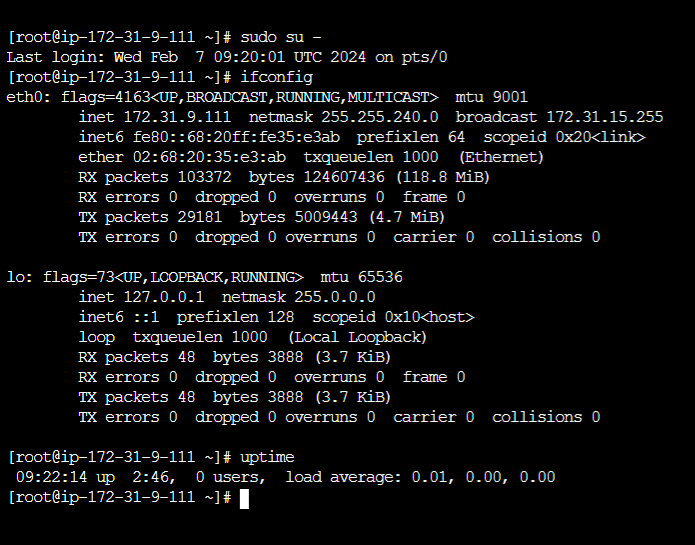
**ls**

**sudo su –**

**ifconfig**

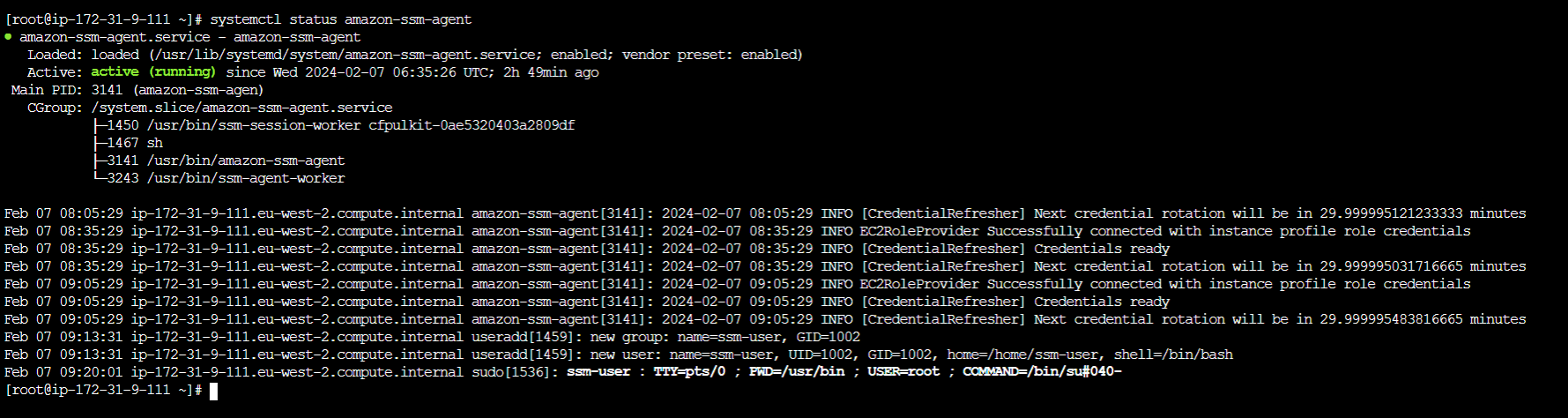
**uptime**





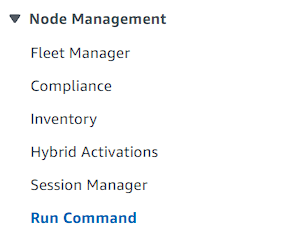
1. Now check whether the system manager package is installed in the session or not as usually, by default it is already installed in your session but if not, you can download it by going through the documentation. To check the system manager installation run the following command:

**systemctl status amazon-ssm-agent**

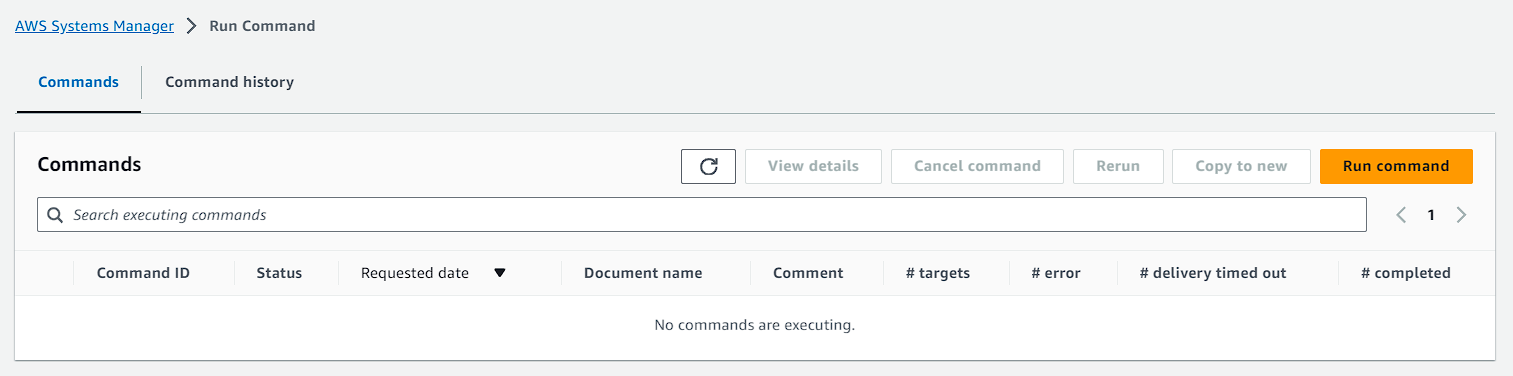
****

**😎 Step 2: Run Command**

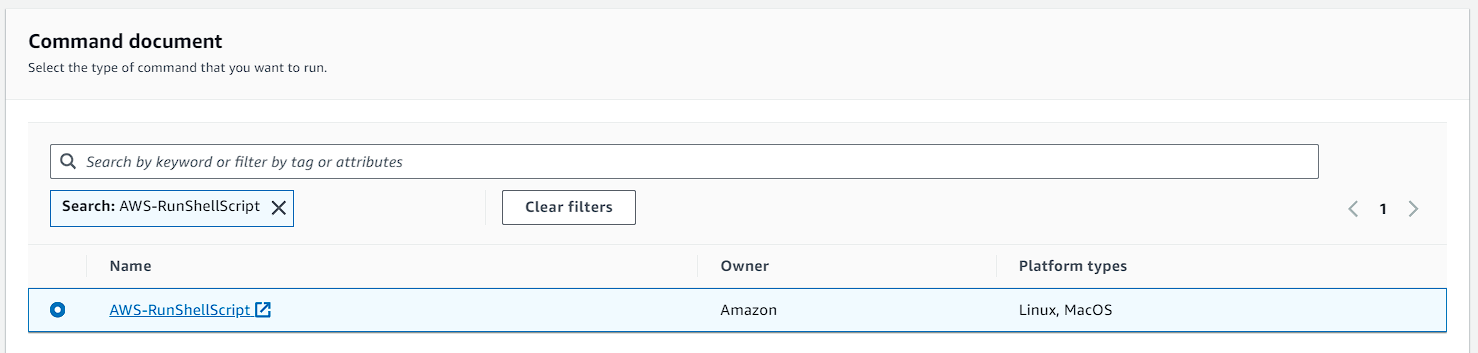
1. Now you are going to learn about the run command, for the you need to go to run command option which is available under Session manager option.



1. Now you need to click on Run command.



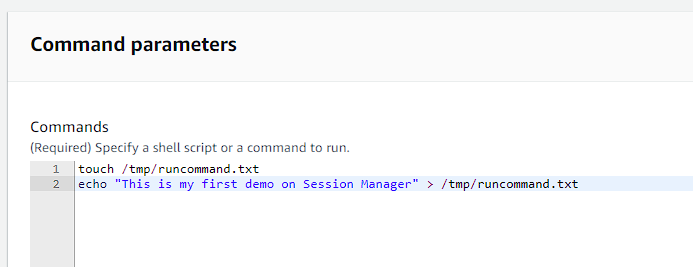
1. Then after you need to search **AWS-RunShellScript** select it



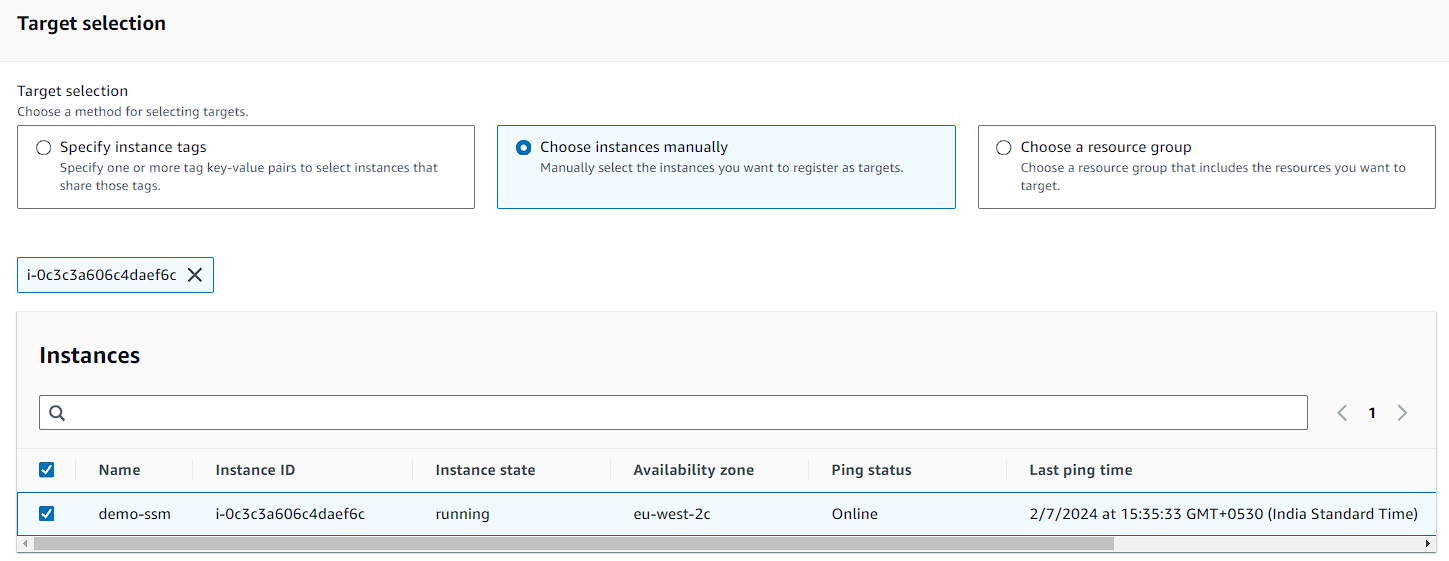
1. After that on the command parameters you need to write this because it is a mandatory field. Based on this command the run command feature will work.

**touch /tmp/runcommand.txt**

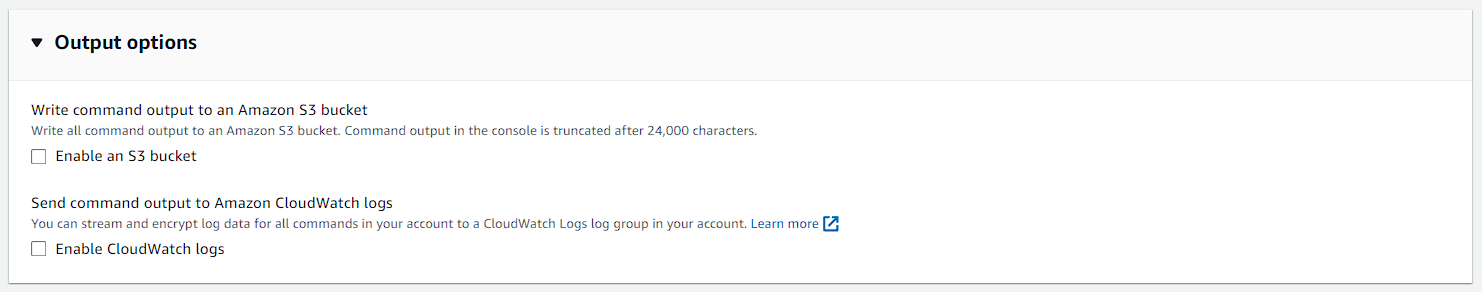
**echo "This is my first demo on Session Manager" > /tmp/runcommand.txt**



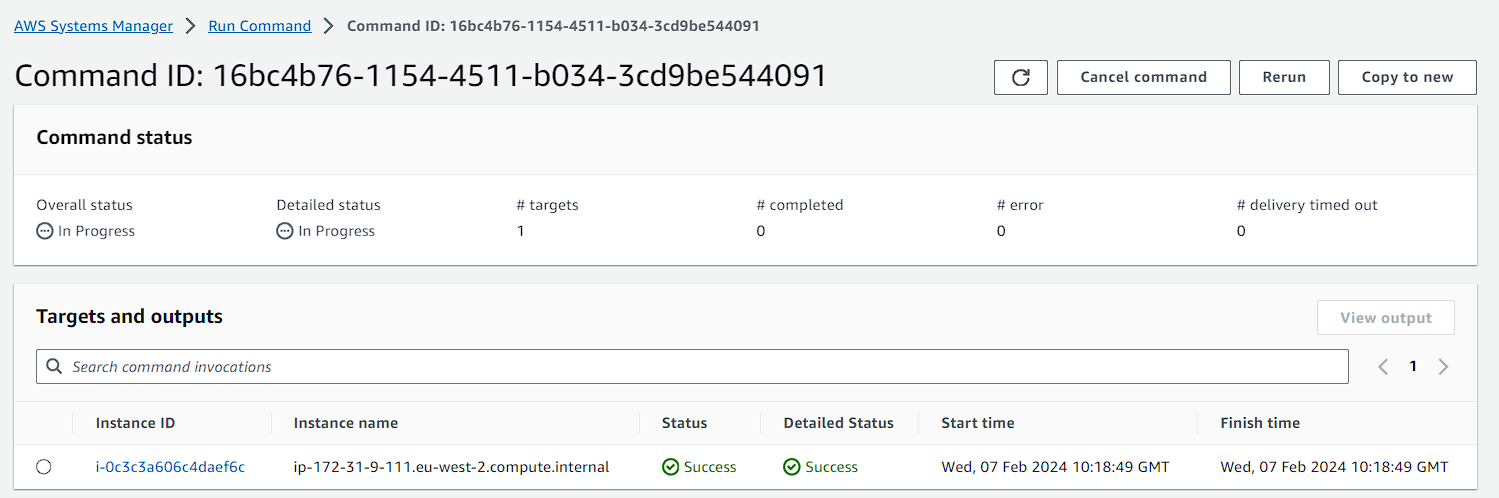
1. After that you need to come down to the Target selection option.
2. Here you need to select choose instances manually and choose your instance.



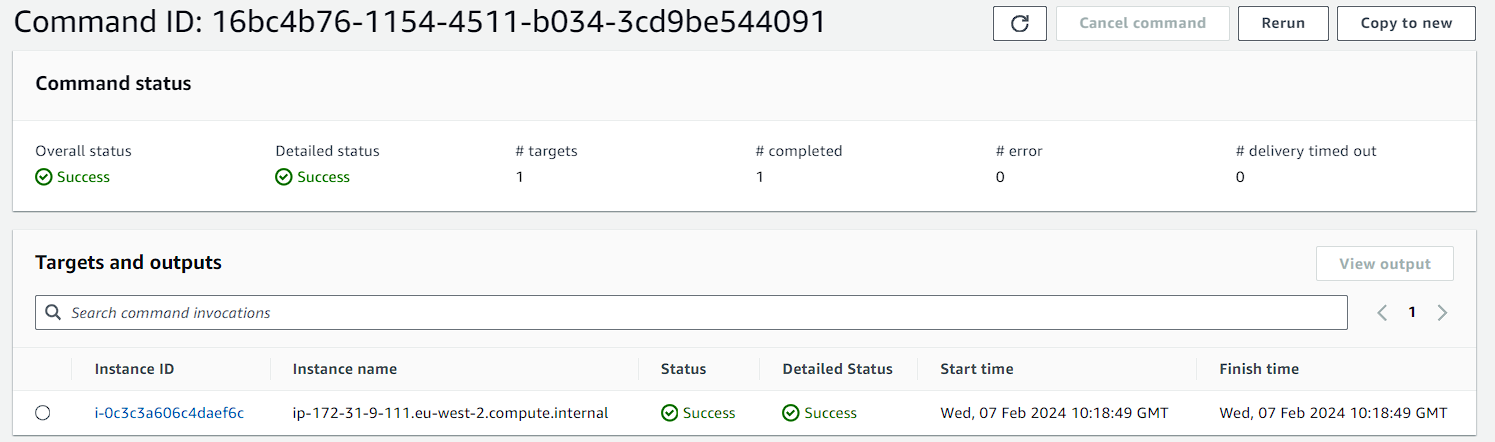
1. After that in the output options deselect Enable an S3 bucket.
2. Then just run your command



1. Now you can see that the overall status is in progress. You need to wait for some time.



1. So, after waiting for several seconds it is in the success state.



1. Now go back to EC2 and connect your instance or login to your instance.
2. There you need to run some to check whether the run command has performed its work or not.

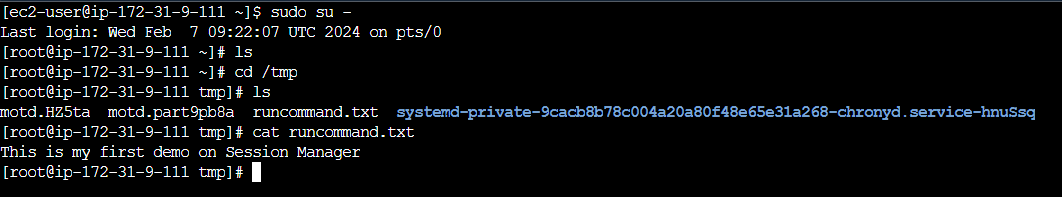
**sudo su –**

**ls**

**cd /tmp**

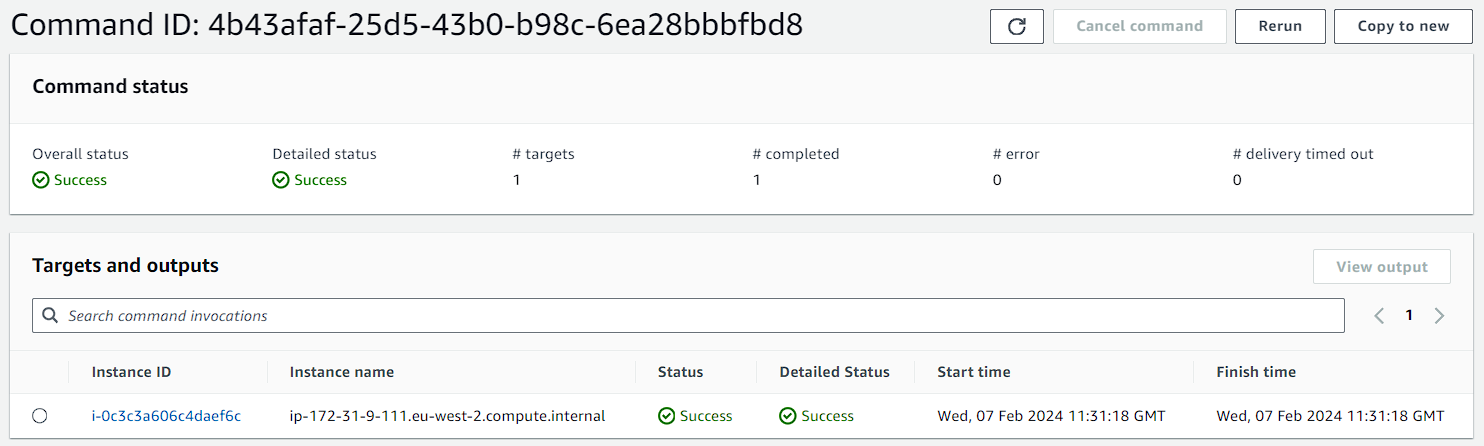
**ls**

**cat runcommand.txt**

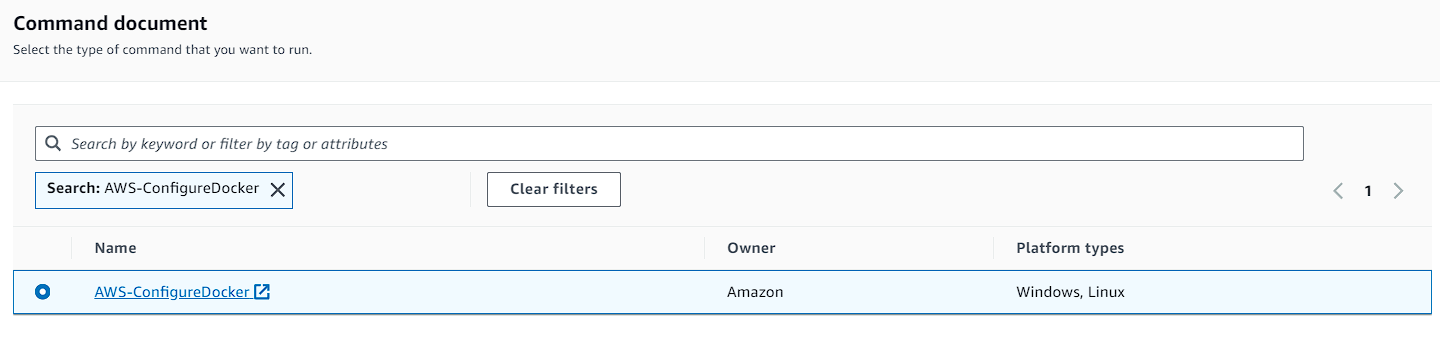


1. Now if you want to install HTTPD on your instance. Then you can also do that the steps are same you just need to write this code.
2. Below you can see that the command status is a success which means that HTTPD is now installed on our instance.

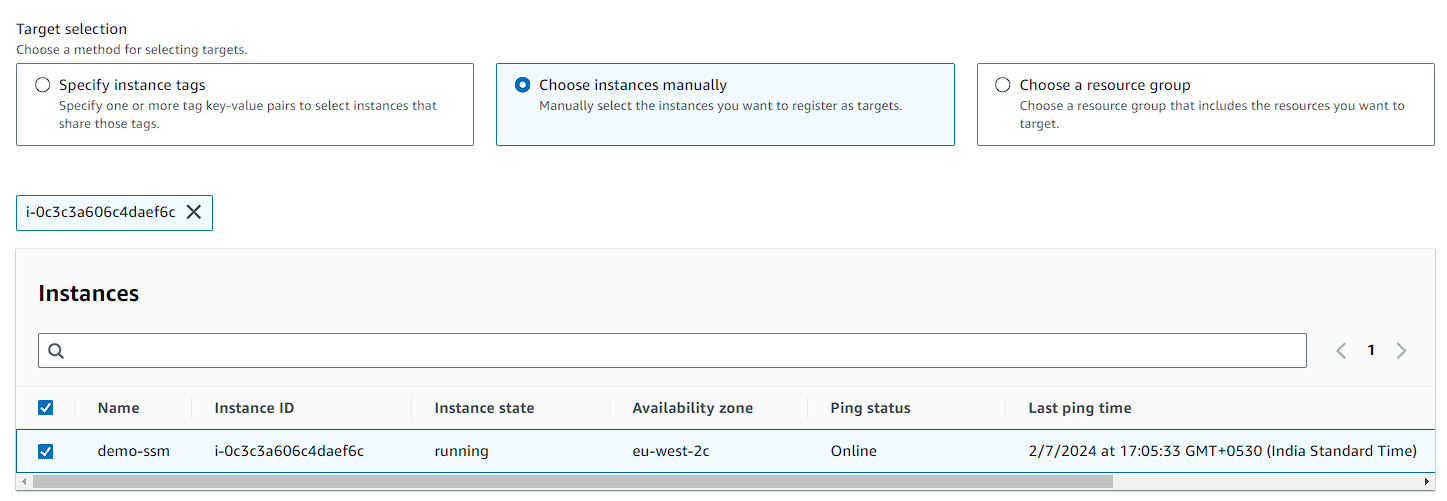
**rpm -qa l grep httpd**



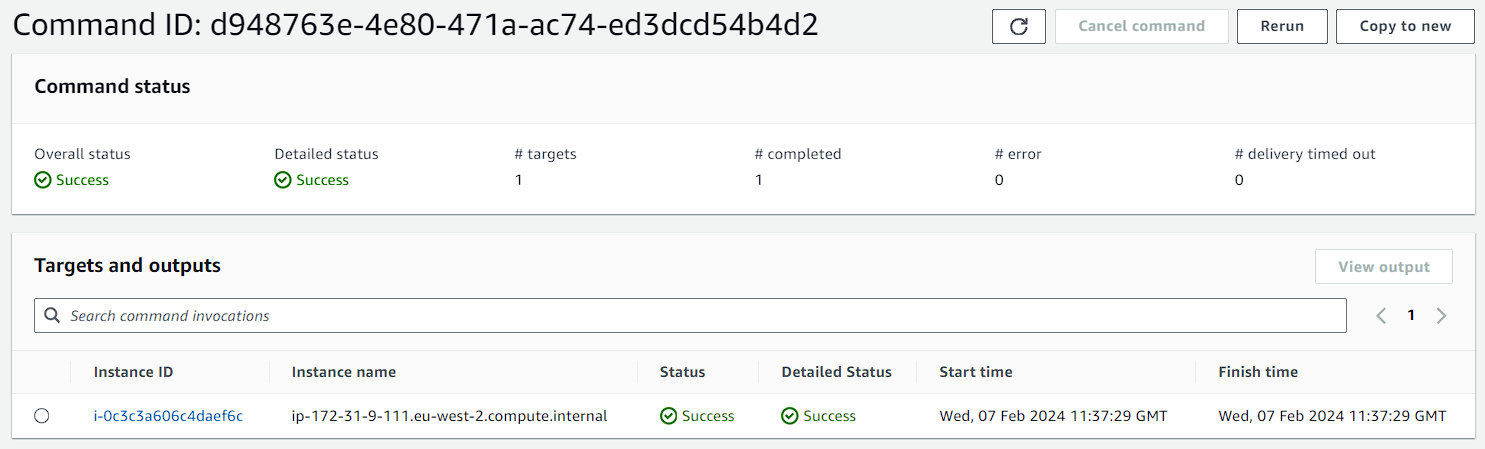
1. Now you are going to configure docker on your instance.
2. For that again you need to go and click on Run command.
3. Choose this option accordingly.



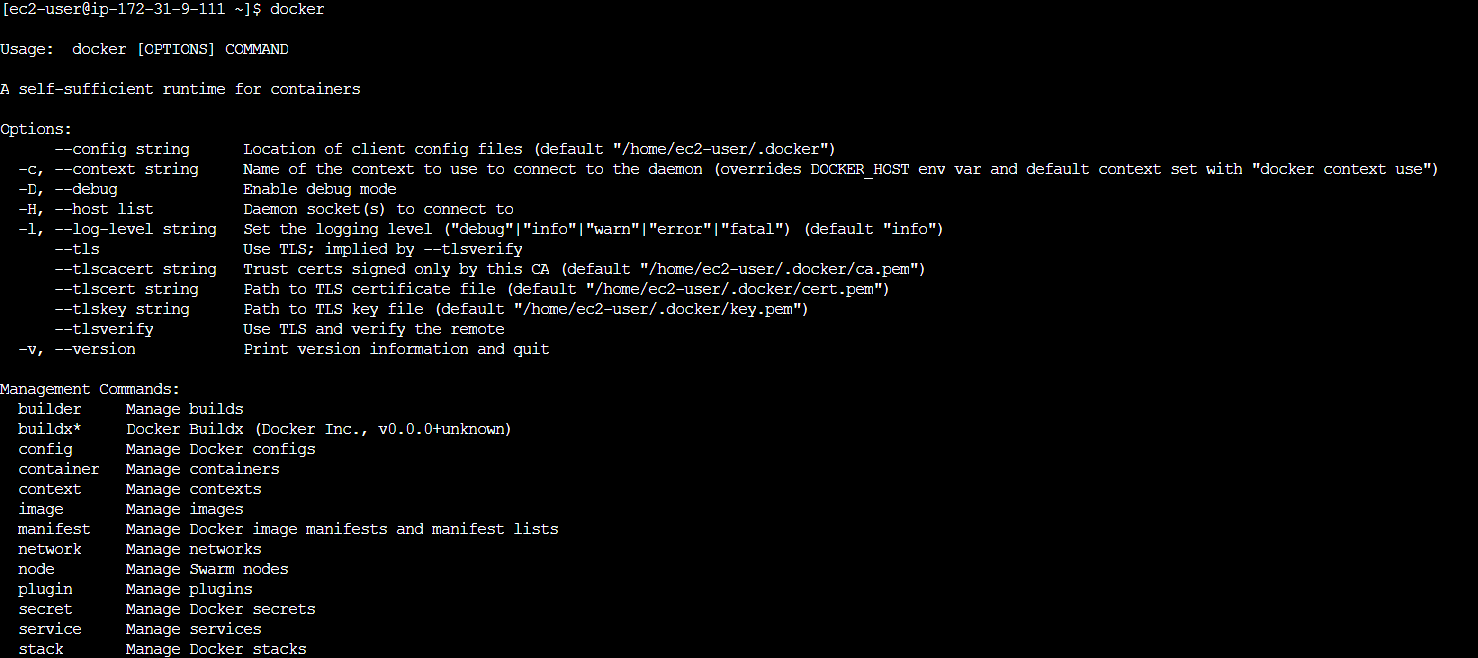
1. After that you just need to select your instance and click on run command. No code or command is required this time.
2. Now wait for some time.



1. After some time, you can see that the command is a success.



1. If you back to your EC2 session and run docker over there you can see the contents.



1. Now if you run a command to check the docker status, you will see that it is up and running.

**systemctl status docker**

