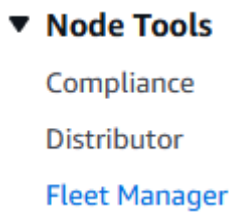
SSM_SysOps

SSM is an AWS service that is really useful for running operations and easily managing a fleet of instances. SSM is also useful for security as we don't need to open an SSH port or give permissions to a system administrator because each instance's SSM agent handles the communication securely.

In order to get started with Systems Manager, we first need to register some instances within it. To do this, go to the SSM service and select "Node Tools" -> "Fleet Manager".

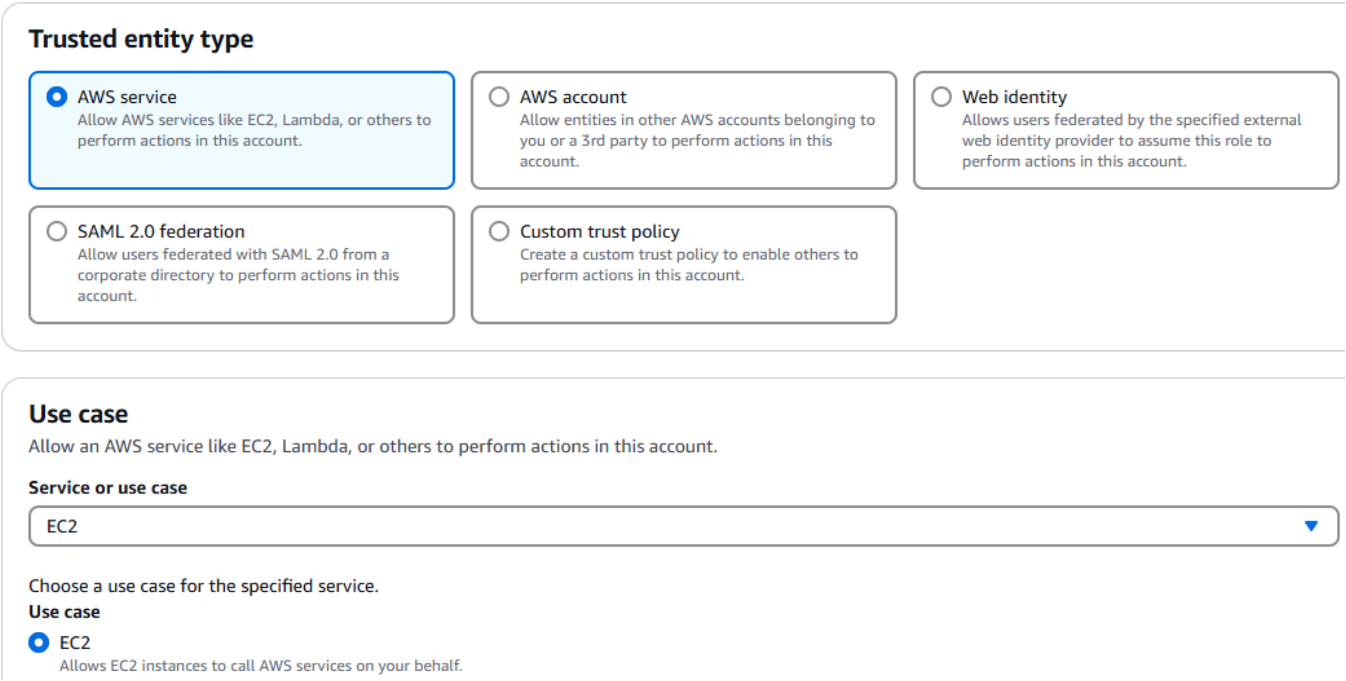


As we can see, there isn't a managed instance yet, so we need to go to the "EC2" service and start a new instance. The instance must meet these two requirements:

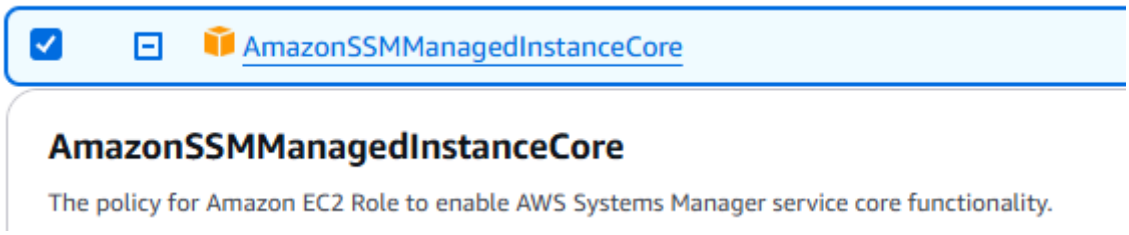
- It must have the SSM agent installed and running.
- It must have the correct IAM permissions to talk to the SSM service.
  Thus, a good choice for this is to use the Amazon Linux 2023 AMI.

When creating the instances, we are going to create a new IAM profile to be able to access the SSM service:

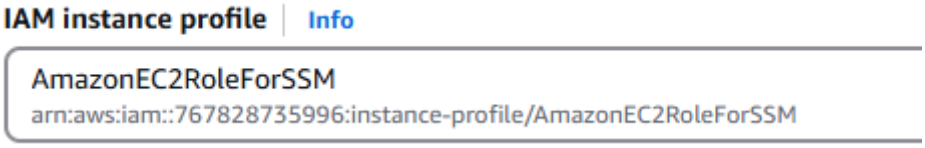
We will allow a service (EC2) to perform actions on this account. Then we need to select the EC2 service and click on the EC2 use case.



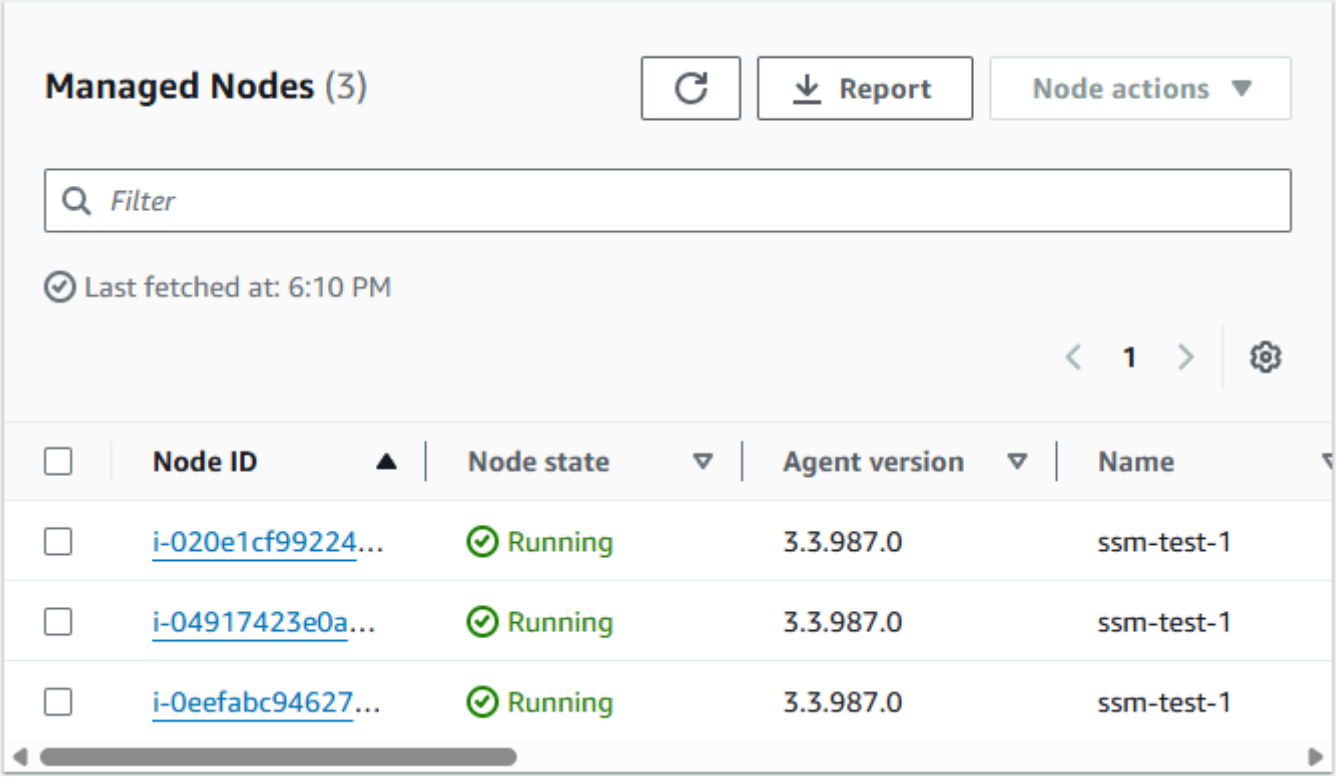
Then, add the `AmazonSSMManagedInstanceCore` which enables the core service functionality:



Now, before we launch the instances, we must assign the role to them:



Finally, change the "Number of instances" to 3 and click "Launch instance". As we are using the Amazon Linux 2023 AMI and the correct IAM permissions, the instances should appear in the Fleet Manager screen on the SSM service when they are finished launching.

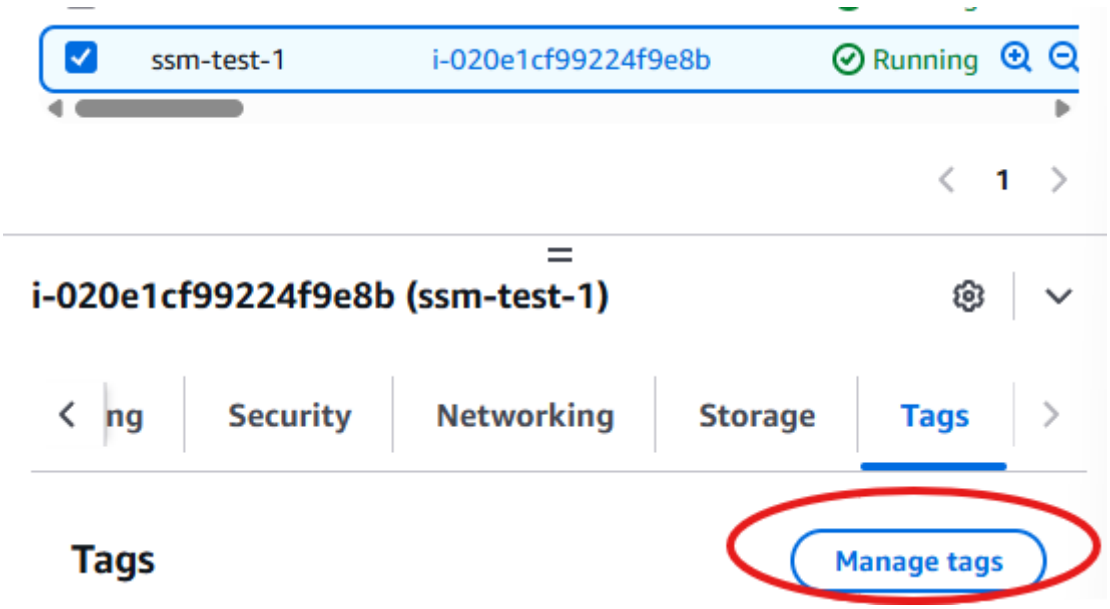


We can see the same three instances that appear in the "Instances" dashboard on the EC2 service. What's more, we can also see the SSM Agent version installed on them.

Tag Resources

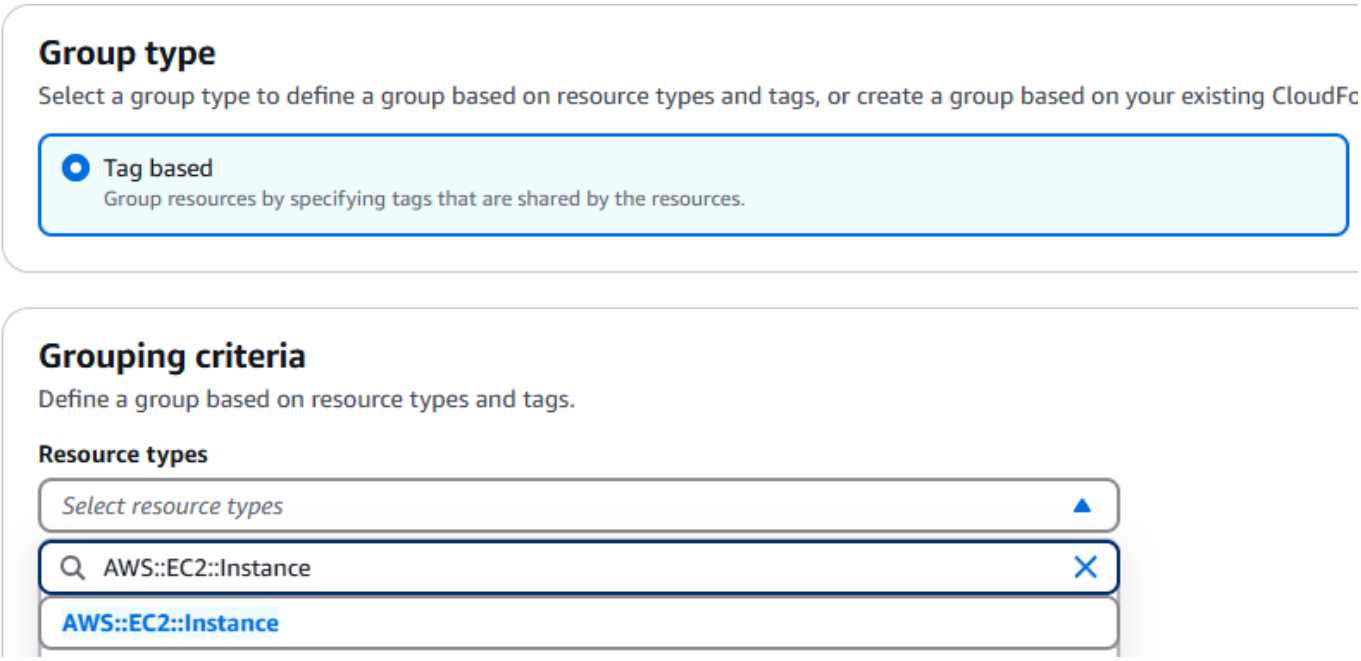
Tags are useful because we can operate SSM at group level, so we can perform some actions on all instances in a group at once.

To create a new tag, we need to select the instance in the Instances dashboard in the EC2 service and then click on Tags -> Manage Tags.

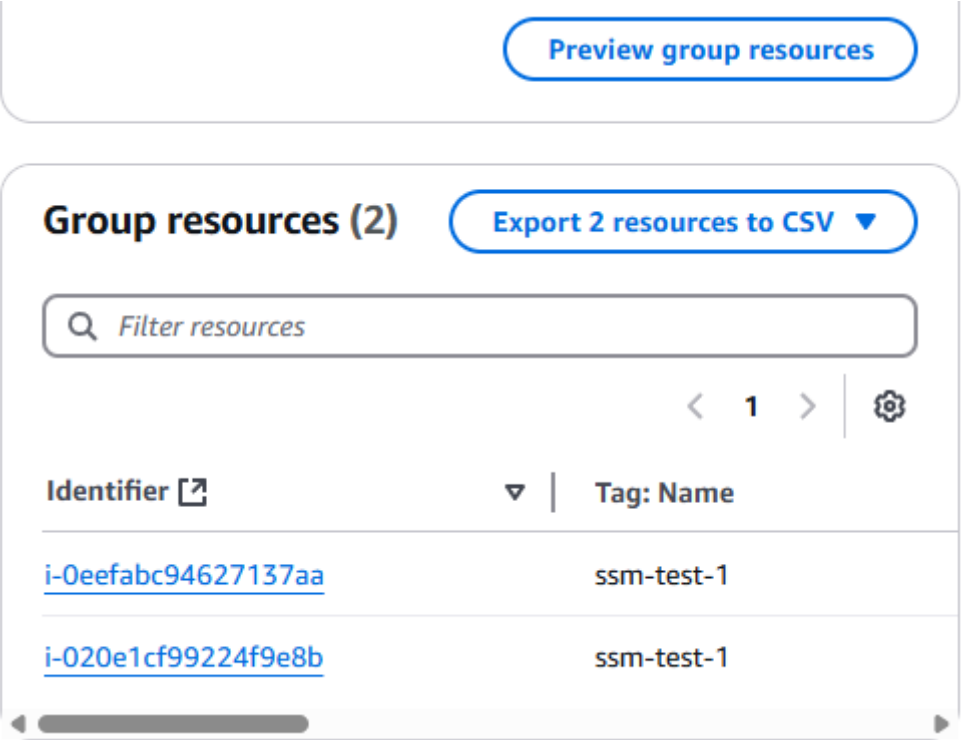


Tags are tuples of key pair values. We can add two new tags to our instance:

Now tag another instance with the same values, changing the `Environment` tag to `Prod`, and the last instance with the `Environment` tag set to `Dev` and the `Team` tag set to `Ops`. We can now create a tag-based resource group in the Resource Groups service. We will manage the resource type "EC2 Instance".



Now, in the Tags section, add the `Environment` tag to be equal to `Dev`. If we click on "Preview Group Resources", we can see that there are two instances in our resource group.



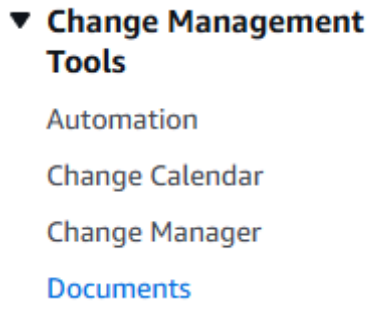
Similarly, we can create another resource group for the `Prod` environment, and a final resource group for the `Engineering` team tag.

SSM - Documents

Documents are the core of SSM. They are JSON or YAML files that define

- Parameters
- A set of steps or **actions** (*what the document does*)
  And then you have the document executed by a specific service.

To manage these documents, we go to the SSM service. In the latest version of the AWS console, we find the documents under the "Change Management Tools" section:



For example, we can search for the `AWS-RunPatchBaseline` document, which is owned by Amazon. In `Content`, we can see all the different commands that are executed for each action, and in `Details`, we can see all the different parameters that the document defines.

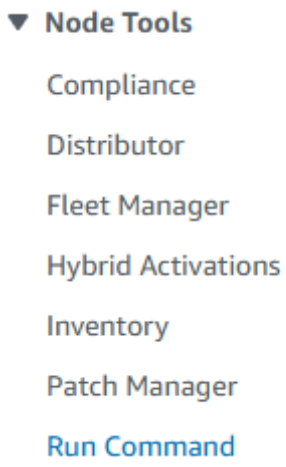
We can create two different types of document:

- **Command or Session**: Runs the entire document or command across a fleet of EC2 instances.
- Automation

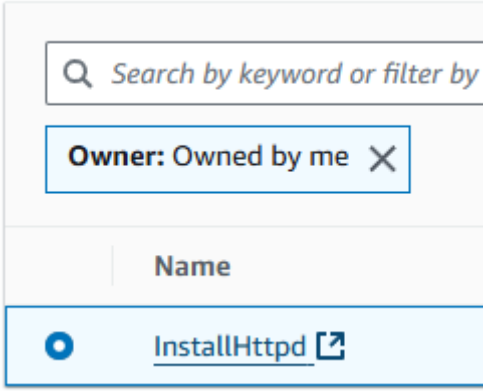
Now let us create a new 'command' document that will install `httpd`. Set the document type to Command and the target type to `/AWS::EC2::Instance`. Then, in the content section, add the following:

```
{
  "schemaVersion": "2.2",
  "description": "Command Document to install httpd on Amazon Linux 2023",
  "parameters": {
    "Message": {
      "type": "String",
      "description": "Message to be displayed on the website.",
      "default": "Hello World"
    }
  },
  "mainSteps": [
    {
      "action": "aws:runShellScript",
      "name": "installHttpd",
      "inputs": {
        "runCommand": [
          "sudo yum update -y",
          "sudo yum install -y httpd",
          "sudo systemctl start httpd",
          "sudo systemctl enable httpd",
          "echo \"{{Message}} from $(hostname -f)\" >
/var/www/html/index.html"
        ]
      }
    }
  ]
}
```

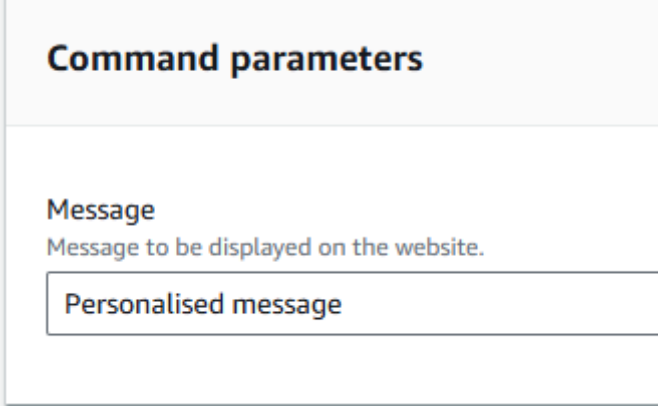
Now, the new document should appear in the "Owned by me" section. Finally, we can go to "Node Tools" > "Run Command" and run the command document.



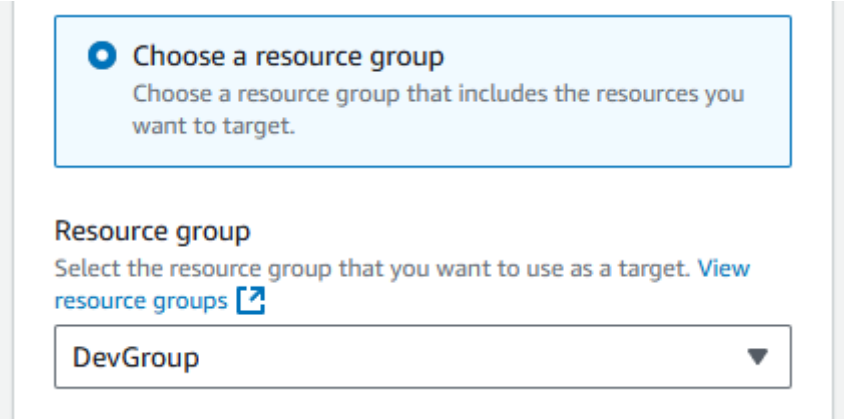
Select the command document you want to run:



We can configure the message to be displayed on the website because we defined it as a parameter earlier:



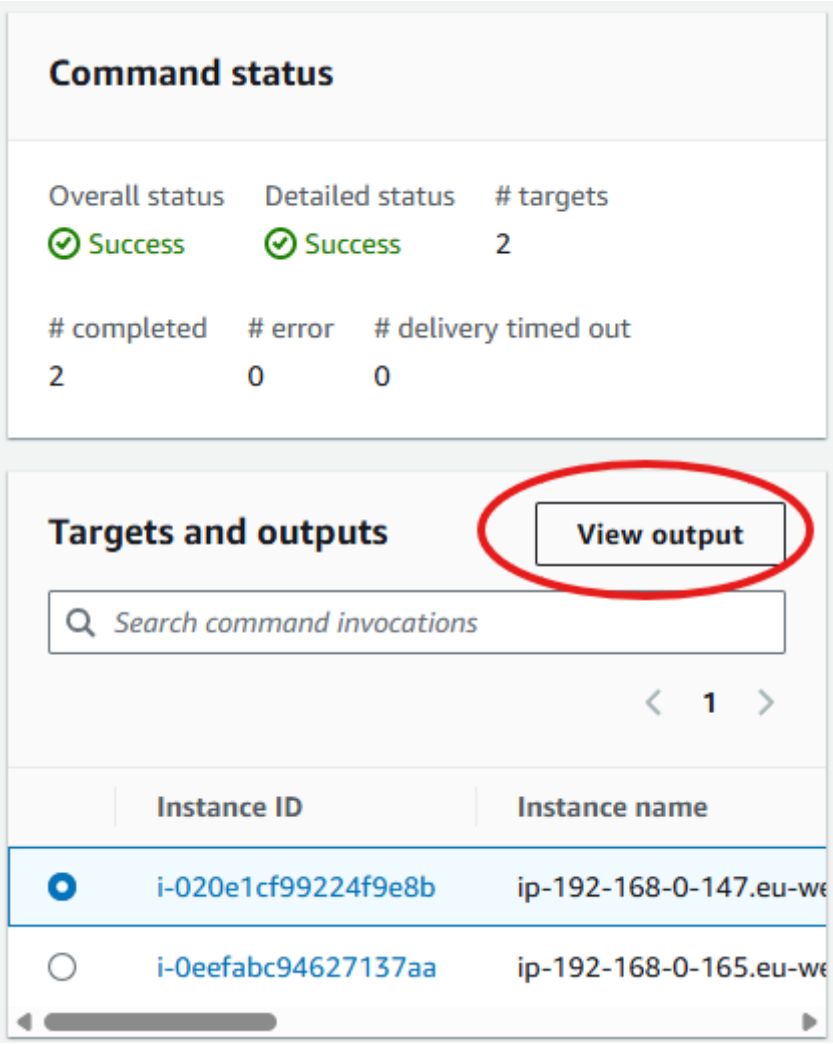
Now the resource groups we created earlier come in handy as we can decide to run this command document on a specific resource group.



Since we only have 3 instances, we'll tell SSM to run the task on one instance at a time, and since this is a really simple command, we don't expect any errors, so I'll set the error threshold to 0.

We can also choose to send the output to either AWS CloudWatch or AWS S3. Note that in order to log the output to a log group in CloudWatch, that log group must exist. In addition, the instance must have sufficient permissions to log events to CloudWatch. To do this, add the `CloudWatchAgentAdminPolicy` and the `CloudWatchAgentServerPolicy` permission policies to the IAM role.

Once we click run, we can see the running status and also see the output and log messages.



If you now copy the public IP of the instance and navigate to it, you should see the following message:

```
Personalised message from <hostname>
```