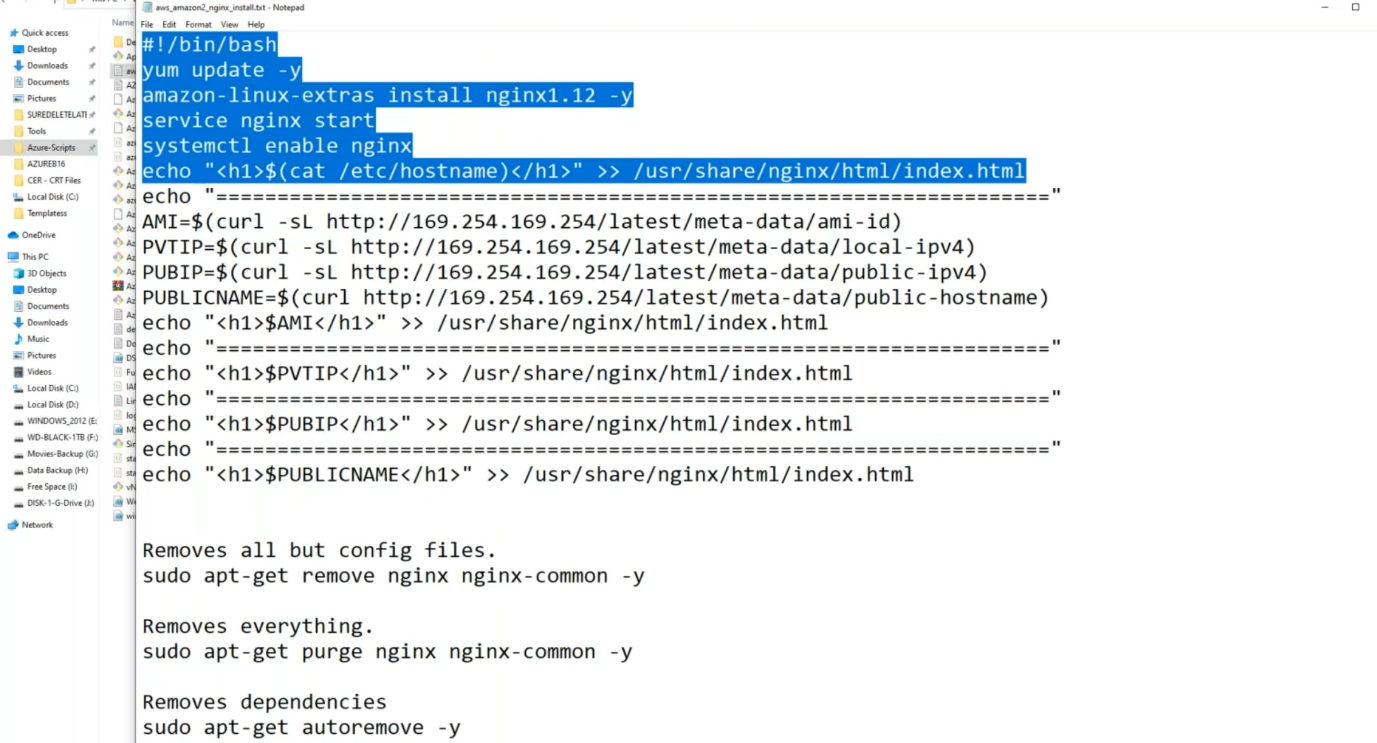
**17.EC2-AMI-Packer-Inspector-Hardening**

--- in this session, we will discuss about below topics.

1. Understanding AMI.
2. scanning aws AMI using inspector.
3. Mitigate the issue using ansible patching.
4. Creating a custom AMI – manual
5. Automating AMI creation using packer.

**scanning aws AMI using inspector.**

--- provision an ec2 instance using below data.



**#! /bin/bash**

**yum update -y**

**amazon-linux-extras install nginx1.12 -y**

**service nginx start**

**systemctl enable nginx**

**echo " <h1>$ (cat /etc/hostname)</h1>" >> /usr/share/nginx/html/index.html**

**aws inspector installation**

--- **Reference** - <https://docs.aws.amazon.com/inspector/v1/userguide/inspector_installing-uninstalling-agents.html#install-linux>

**# Download the inspector using below command**

--- **curl -O** [**https://inspector-agent.amazonaws.com/linux/latest/install**](https://inspector-agent.amazonaws.com/linux/latest/install)

**# Give executable permissions**

--- chmod 777 install

**# Execute the script**

--- ./install

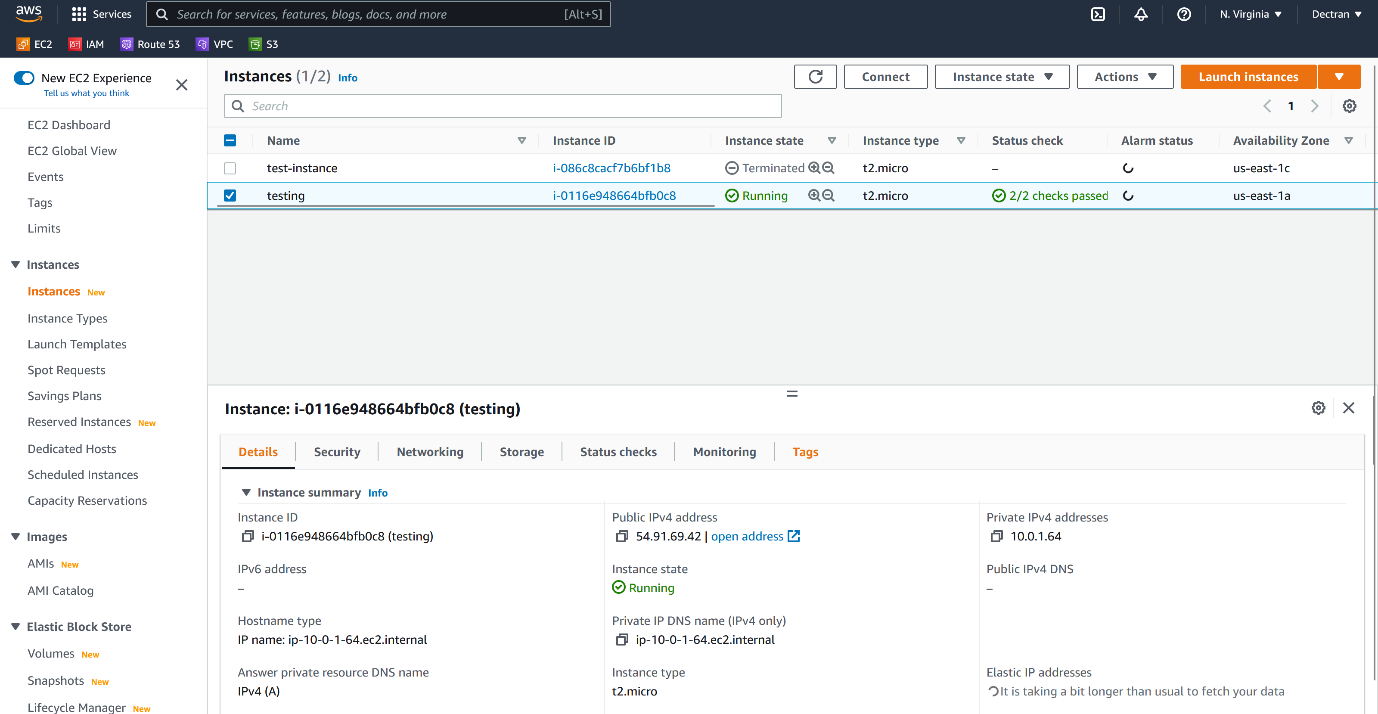
--- **note** – installation is completed.

--- **note** – now we will scan the instance using aws inspector.

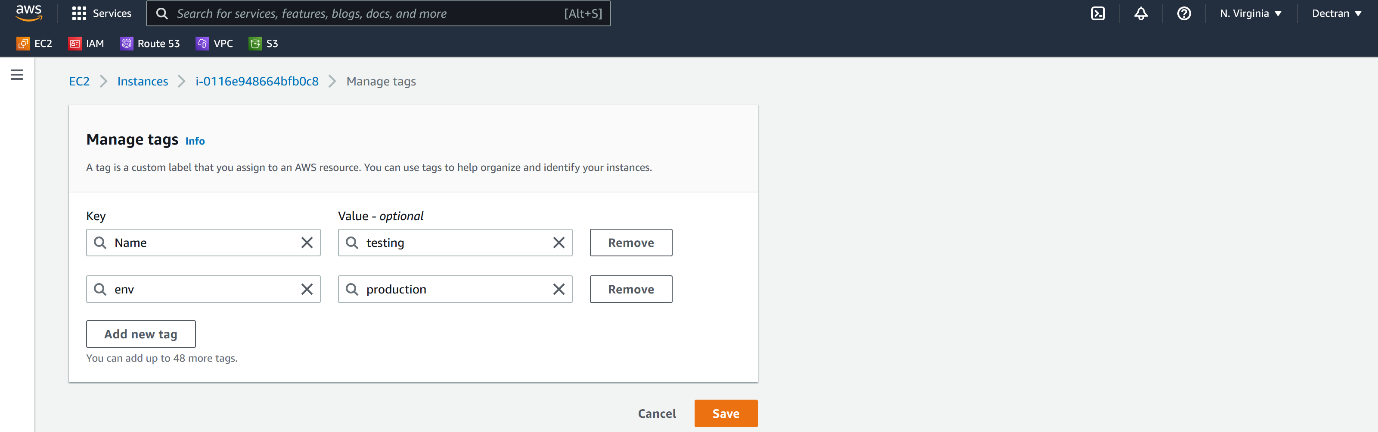
**AWS inspector**

--- now, we must tell the aws inspector that which instance it must scan.

--- aws inspector work on the tags so, we need to tag the instance.

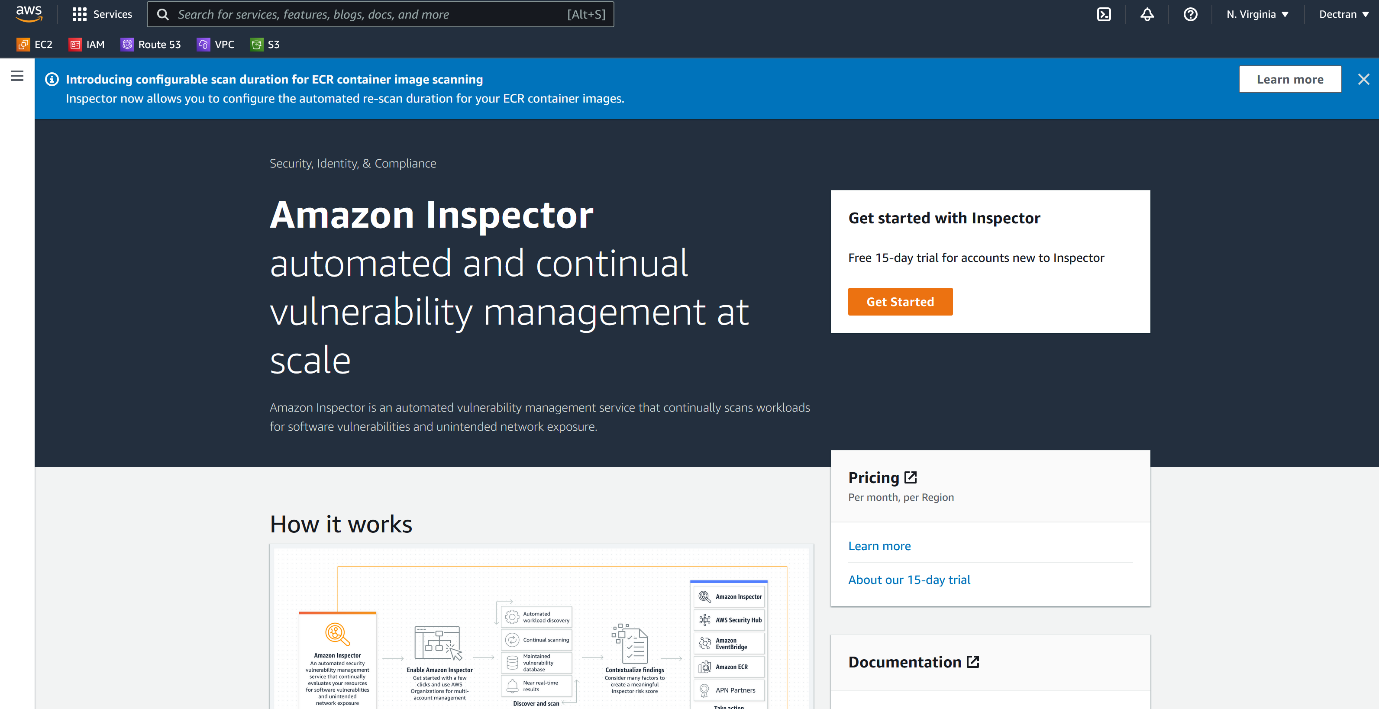


--- click on tags



--- click on save.

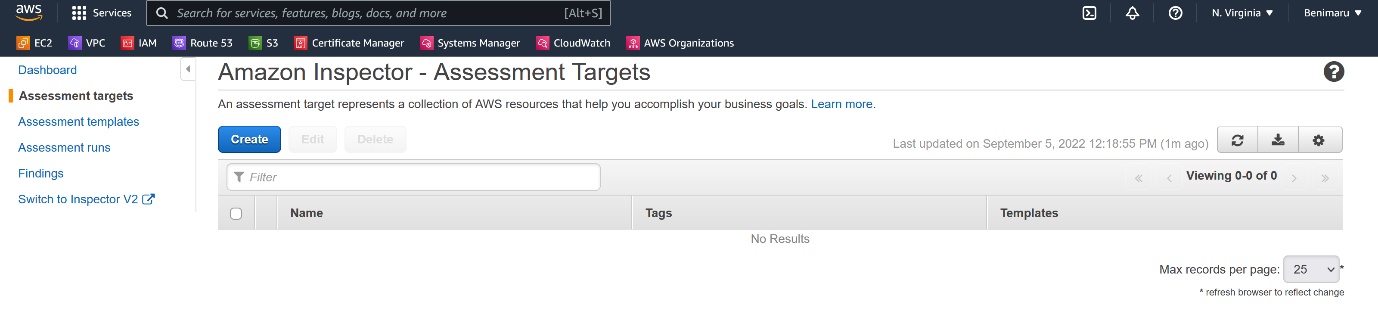
--- now go to the aws inspector.

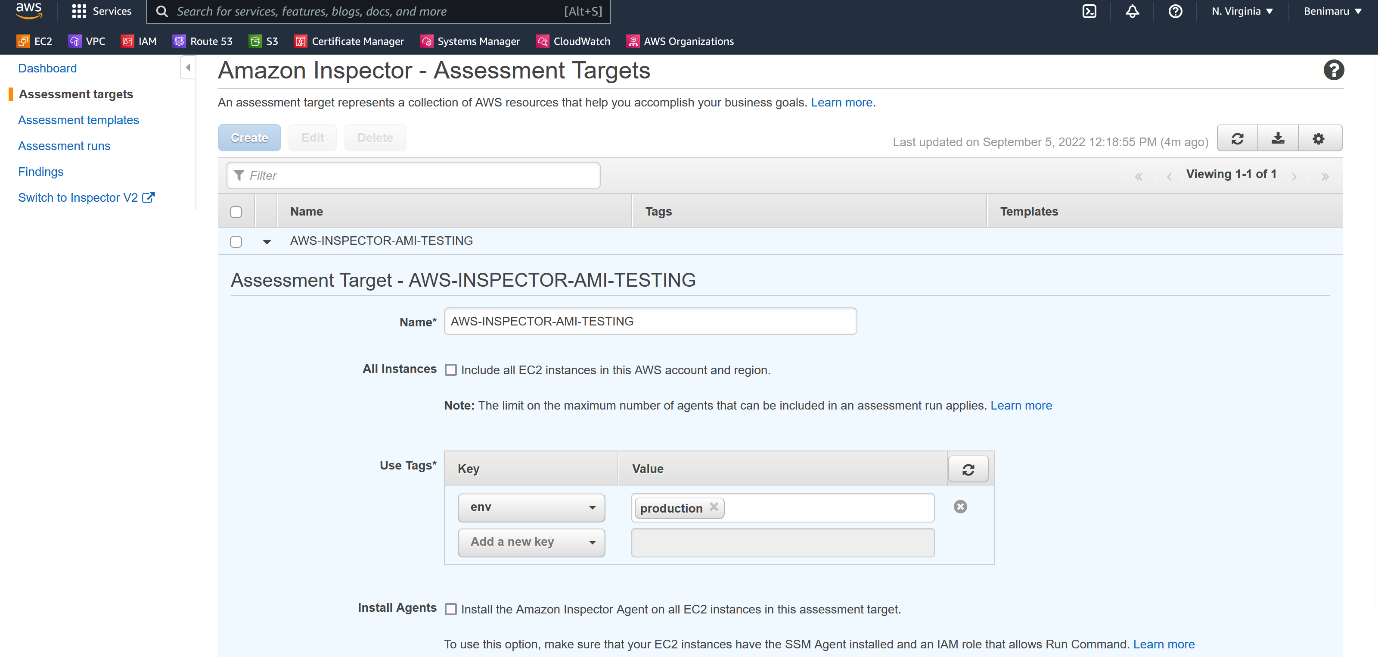


--- Click on get started.

**Create assessment targets**

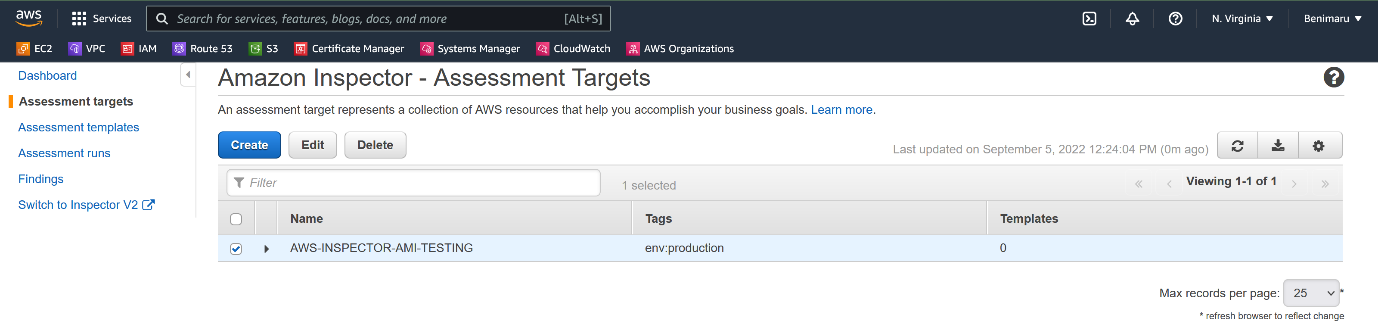
--- **note** – by doing this, we are telling inspector that to which it needs to scan.

--- click on create



--- uncheck the install agent box, because we already installed the agent on the target.

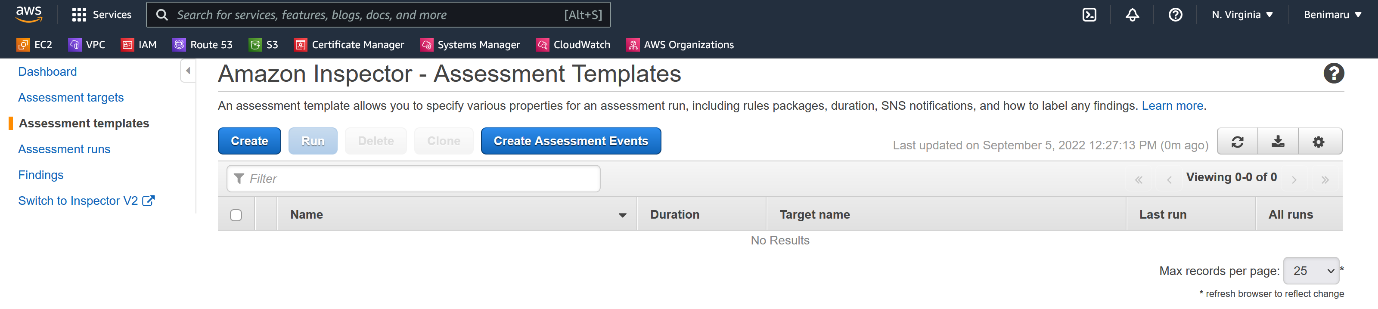
--- click on save.



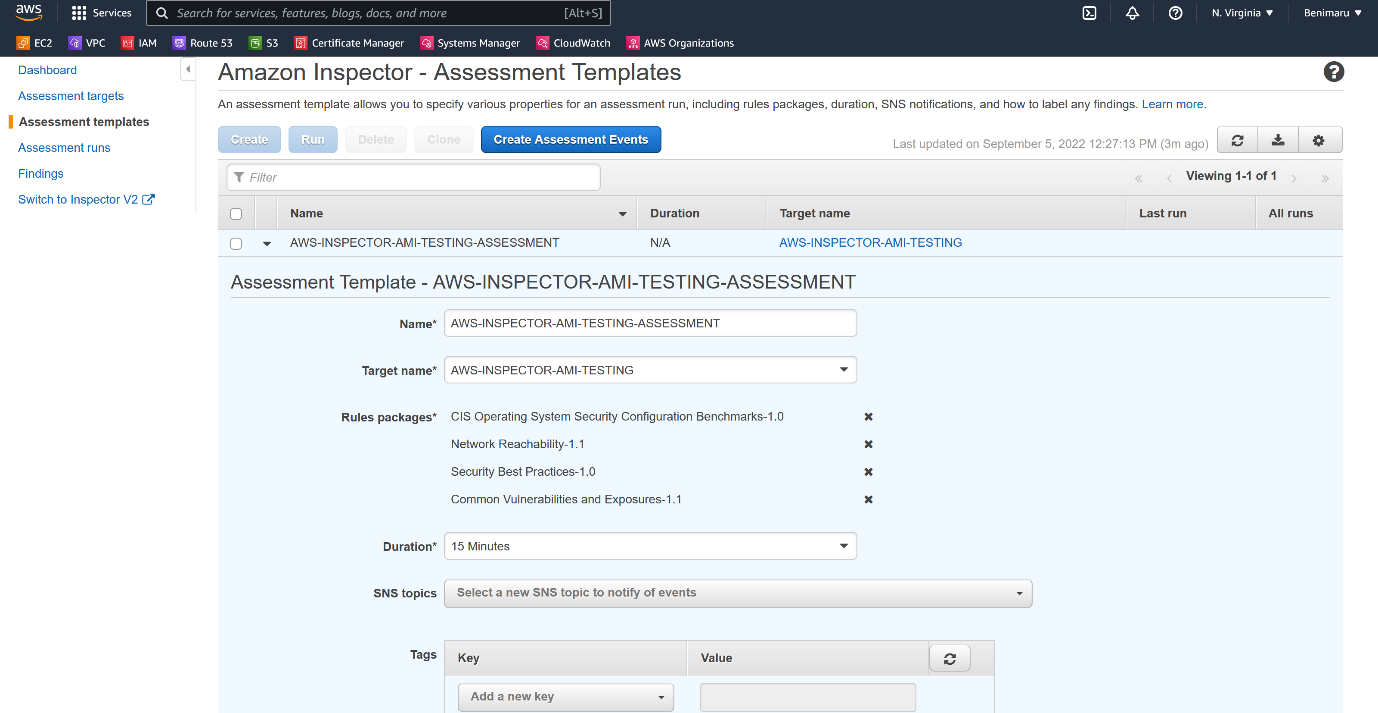
--- assessment targets got created.

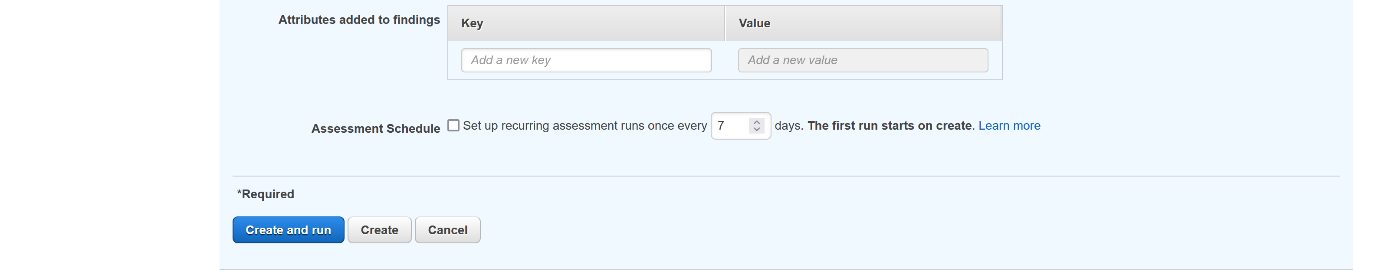
**Assessment templates**

--- **note** – what to scan.



--- click on create.





--- uncheck the assessment schedule.

--- rules packages **1. CIS Operating System Security Configuration Benchmarks-1.0**

**2. Network Reachability-1.1**

**3. Security Best Practices-1.0**

**4. Common Vulnerabilities and Exposures-1.1**

--- click on create and run.

**Mitigate the issue using ansible patching (Patching amazon linux2 )**

--- **note** – we will use ansible tool to patch our instance.

**# Use amazon linux2 and install nginx, ansible and git.**

--- amazon-linux-extras install epel

--- yum update -y

--- yum install nginx -y

--- yum install ansible -y

--- yum install git -y

--- git clone <https://github.com/prabhu9652/cis-amazon-linux-2.git> /root/roles/cis-amazon-linux-2

--- cd /root/roles/cis-amazon-linux-2

**# Create a playbook and execute**

--- playbook.yml

**---**

**- hosts: localhost**

**connection: local**

**gather\_facts: true**

**become: yes**

**roles:**

**- cis-amazon-linux-2**

**# Execute the playbook using below command**

--- ansible-playbook playbook.yml

--- **note** – now the ansible will patch the instance.

**Package use case**

--- the packer will automate below steps.

1. Create a server
2. Login into the server
3. Install necessary packages or apps or harden
4. Stop the server
5. Create ami out of the server
6. Delete the server

--- note – packer is a 3rd party tool,

# Clone the repository

--- git clone <https://github.com/mavrick202/terraformsingleinstance.git>

--- note – in this repository, you will find 2 files for packer.

--- packer-vars.json

{

    "aws\_access\_key": "xxx",

    "aws\_secret\_key": "yyy",

    "region": "us-east-1",

    "source\_ami": "ami-04505e74c0741db8d",

    "instance\_type": "t2.micro",

    "vpc\_id": "vpc-0eac6d4d55868caa8",

    "subnet\_id": "subnet-0d124b5eb12011584"

}

--- packer.json

{

    "\_comment" : "Create a AWS AMI ith AMZ Linux 2018 with Java and Tomcat",

    "variables": {

      "aws\_access\_key": "",

      "aws\_secret\_key": "",

      "region": "",

      "source\_ami":"",

      "instance\_type":"",

      "vpc\_id": "",

      "subnet\_id": ""

    },

    "\_comment1" : "packer build -var \"aws\_secret\_key=foo\" template.json",

    "\_comment2" : "packer build -var-file packer-vars.json template.json",

    "builders": [{

      "access\_key": "{{user `aws\_access\_key`}}",

      "secret\_key": "{{user `aws\_secret\_key`}}",

      "type": "amazon-ebs",

      "region": "{{user `region`}}",

      "source\_ami": "{{user `source\_ami`}}",

      "instance\_type": "{{user `instance\_type`}}",

      "ssh\_username": "ubuntu",

      "ami\_name": "DevOpsClass-B18-Build-{{isotime | clean\_resource\_name}}",

      "vpc\_id": "{{user `vpc\_id`}}",

      "subnet\_id": "{{user `subnet\_id`}}",

      "tags": {

        "Name": "DevOpsClass-B18-Build-{{isotime | clean\_resource\_name}}"

      }

    }],

    "provisioners": [{

      "type": "shell",

      "inline": [

        "sleep 30",

        "sudo apt update -y",

        "sudo apt install nginx -y",

        "sudo apt install git -y",

        "sudo git clone https://github.com/mavrick202/webhooktesting.git",

        "sudo rm -rf /var/www/html/index.nginx-debian.html",

        "sudo cp webhooktesting/index.html /var/www/html/index.nginx-debian.html",

        "sudo cp webhooktesting/style.css /var/www/html/style.css",

        "sudo cp webhooktesting/scorekeeper.js /var/www/html/scorekeeper.js",

        "sudo service nginx start",

        "sudo systemctl enable nginx",

        "curl https://get.docker.com | bash"

        ]

    },

    {

            "type": "file",

            "source": "docker.service",

            "destination": "/tmp/docker.service"

    },

    {

        "type": "shell",

        "inline": [

            "sudo cp /tmp/docker.service /lib/systemd/system/docker.service",

            "sudo usermod -a -G docker ubuntu",

            "sudo systemctl daemon-reload",

            "sudo service docker restart"

        ]

    }

]

  }

# validate packer

--- packer validate -var-file packer-vars.json packer.json

# Build ami

--- packer build -var-file packer-vars.json packer.json

--- **note** – go to the aws console and find the ami