# Ansible Practitioner 36-39

{ PREVIOUS LAB REMAINING }

HOST :

[root@control cloud\_user]# curl 169.254.169.254/latest/user-data

#!/bin/bash

/bin/echo 'E3mXx9v%' | /bin/passwd root —stdin

/bin/echo 'E3mXx9v%' | /bin/passwd cloud\_user —stdin

/bin/echo "StrictHostKeyChecking no" >> /etc/ssh/ssh\_config

sed -i s/requiretty/\!requiretty/ /etc/sudoers

hostnamectl set-hostname control

yum install -y epel-release

yum install -y ansible sshpass

yum install -y python-boto python-boto3

/sbin/useradd ansible

/bin/echo 'E3mXx9v%' | /bin/passwd ansible —stdin

/bin/echo "ansible ALL=(ALL) NOPASSWD: ALL" >> /etc/sudoers

sudo -u ansible /bin/mkdir -p /home/ansible/.ssh

sudo -u ansible /bin/ssh-keygen -q -N "" -f /home/ansible/.ssh/id\_rsa

sudo -u ansible sshpass -p 'E3mXx9v%' ssh-copy-id -i /home/ansible/.ssh/id\_rsa.pub ansible@localhost

echo "localhost" >> /etc/ansible/hosts

ACCESS=AKIARNRH2BBMPEZ3TOVF

SECRET=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

echo "export AWS\_ACCESS\_KEY\_ID='KEYACCESS'" > /home/ansible/accesstemplate

echo "export AWS\_SECRET\_ACCESS\_KEY='KEYSECRET'" > /home/ansible/secrettemplate

echo "#!/bin/sh" >> /home/ansible/keys.sh

sed -e "s|KEYACCESS|$ACCESS|" /home/ansible/accesstemplate >> /home/ansible/keys.sh

sed -e "s|KEYSECRET|$SECRET|" /home/ansible/secrettemplate >> /home/ansible/keys.sh

echo "export AWS\_REGION='us-east-1'" >> /home/ansible/keys.sh

chmod 700 /home/ansible/keys.sh

rm -f /home/ansible/accesstemplate /home/ansible/secrettemplate

echo "AWS\_ACCESS\_KEY\_ID: KEYACCESS" > /home/ansible/accesstemplate

echo "AWS\_SECRET\_ACCESS\_KEY: KEYSECRET" > /home/ansible/secrettemplate

sed -e "s|KEYACCESS|$ACCESS|" /home/ansible/accesstemplate >> /home/ansible/keys.yml

sed -e "s|KEYSECRET|$SECRET|" /home/ansible/secrettemplate >> /home/ansible/keys.yml

echo "AWS\_REGION: us-east-1" >> /home/ansible/keys.yml

chmod 600 /home/ansible/keys.yml

rm -f /home/ansible/accesstemplate /home/ansible/secrettemplate

chown ansible.ansible /home/ansible/\*

echo "finished" > /home/cloud\_user/.done

[root@control cloud\_user]# sudo cat /etc/os-release

NAME="CentOS Linux"

VERSION="7 (Core)"

ID="centos"

ID\_LIKE="rhel fedora"

VERSION\_ID="7"

PRETTY\_NAME="CentOS Linux 7 (Core)"

ANSI\_COLOR="0;31"

CPE\_NAME="cpe:/o:centos:centos:7"

HOME\_URL="<https://www.centos.org/>“

BUG\_REPORT\_URL="<https://bugs.centos.org/>“

CENTOS\_MANTISBT\_PROJECT="CentOS-7"

CENTOS\_MANTISBT\_PROJECT\_VERSION="7"

REDHAT\_SUPPORT\_PRODUCT="centos"

REDHAT\_SUPPORT\_PRODUCT\_VERSION="7"

[root@control cloud\_user]# ansible —version

ansible 2.9.27

config file = /etc/ansible/ansible.cfg

configured module search path = [u'/root/.ansible/plugins/modules', u'/usr/share/ansible/plugins/modules']

ansible python module location = /usr/lib/python2.7/site-packages/ansible

executable location = /bin/ansible

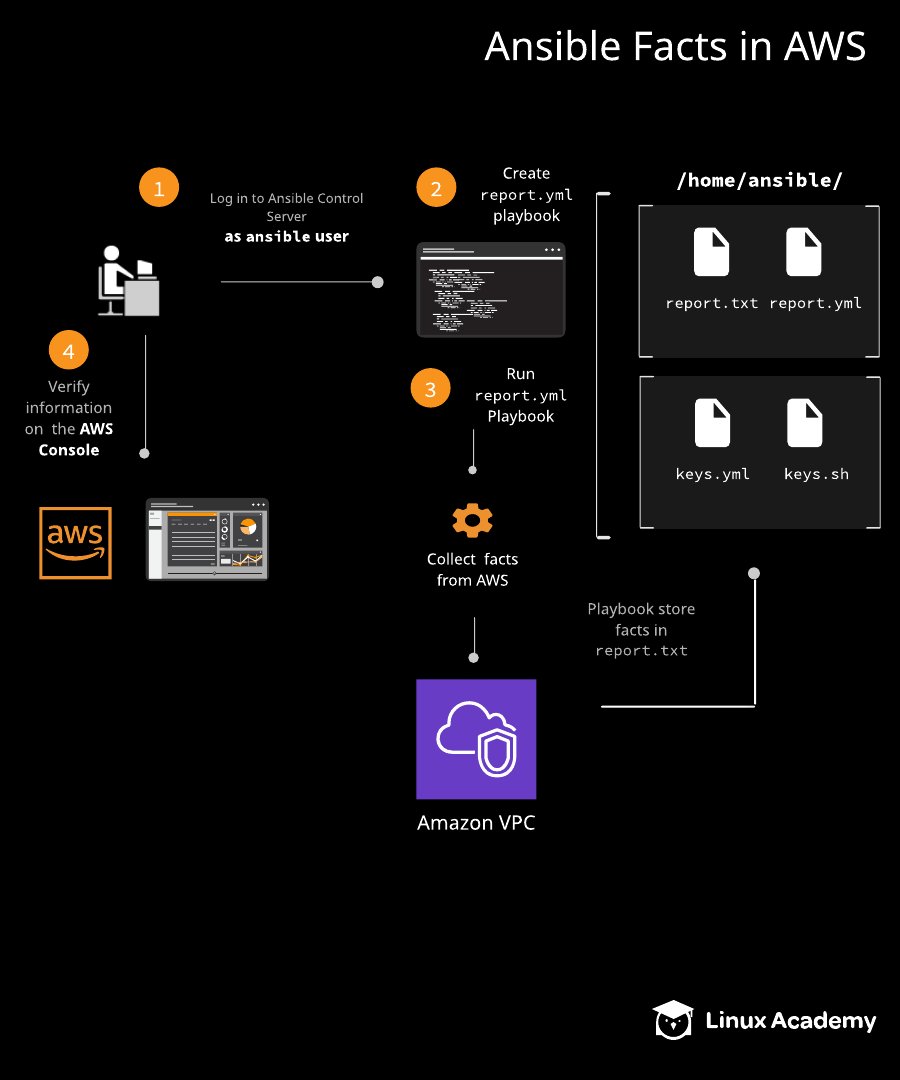
python version = 2.7.5 (default, Nov 16 2020, 22:23:17) [GCC 4.8.5 20150623 (Red Hat 4.8.5-44)]

[root@control cloud\_user]#

CFN TEMPLATE :

{  
 "AWSTemplateFormatVersion" : "2010-09-09",  
 "Description" : "Manipulating EC2 Instances with Ansible",  
  
 "Mappings" : {  
  
   "SubnetConfig" : {  
     "VPC"     : { "CIDR" : "10.0.0.0/16" },  
     "Public1"  : { "CIDR" : "10.0.1.0/24" }  
   }  
 },  
  
 "Resources" : {  
  
   "VPC" : {  
     "Type" : "AWS::EC2::VPC",  
     "Properties" : {  
  
       "EnableDnsSupport" : "true",  
       "EnableDnsHostnames" : "true",  
       "CidrBlock" : { "Fn::FindInMap" : [ "SubnetConfig", "VPC", "CIDR" ]},  
       "Tags" : [  
        { "Key" : "Name", "Value" : "LinuxAcademy" },  
         { "Key" : "Application", "Value" : { "Ref" : "AWS::StackName" } },  
         { "Key" : "Network", "Value" : "VPC" }  
       ]  
     }  
   },  
  
  
   "PublicSubnet1" : {  
     "Type" : "AWS::EC2::Subnet",  
         "Properties" : {  
       "VpcId" : { "Ref" : "VPC" },  
       "AvailabilityZone" : {  
         "Fn::Select" : [ "0", { "Fn::GetAZs" : "" } ]  
       },  
       "CidrBlock" : { "Fn::FindInMap" : [ "SubnetConfig", "Public1", "CIDR" ]},  
       "Tags" : [  
         { "Key" : "Application", "Value" : { "Ref" : "AWS::StackName" } },  
         { "Key" : "Network", "Value" : "Public1" }  
       ]  
     }  
   },  
  
   "InternetGateway" : {  
     "Type" : "AWS::EC2::InternetGateway",  
     "Properties" : {  
       "Tags" : [  
         { "Key" : "Application", "Value" : { "Ref" : "AWS::StackName" } },  
         { "Key" : "Network", "Value" : "Public" }  
       ]  
     }  
   },  
  
   "GatewayToInternet" : {  
      "Type" : "AWS::EC2::VPCGatewayAttachment",  
      "Properties" : {  
        "VpcId" : { "Ref" : "VPC" },  
        "InternetGatewayId" : { "Ref" : "InternetGateway" }  
      }  
   },  
  
   "PublicRouteTable" : {  
     "Type" : "AWS::EC2::RouteTable",  
     "Properties" : {  
       "VpcId" : { "Ref" : "VPC" },  
       "Tags" : [  
         { "Key" : "Application", "Value" : { "Ref" : "AWS::StackName" } },  
         { "Key" : "Network", "Value" : "Public" }  
       ]  
     }  
   },  
  
   "PublicRoute" : {  
     "Type" : "AWS::EC2::Route",  
     "DependsOn" : "GatewayToInternet",  
     "Properties" : {  
       "RouteTableId" : { "Ref" : "PublicRouteTable" },  
       "DestinationCidrBlock" : "0.0.0.0/0",  
       "GatewayId" : { "Ref" : "InternetGateway" }  
     }  
   },  
  
   "PublicSubnetRouteTableAssociation1" : {  
     "Type" : "AWS::EC2::SubnetRouteTableAssociation",  
     "Properties" : {  
       "SubnetId" : { "Ref" : "PublicSubnet1" },  
       "RouteTableId" : { "Ref" : "PublicRouteTable" }  
     }  
   },  
  
   "PublicNetworkAcl" : {  
     "Type" : "AWS::EC2::NetworkAcl",  
     "Properties" : {  
       "VpcId" : { "Ref" : "VPC" },  
       "Tags" : [  
         { "Key" : "Application", "Value" : { "Ref" : "AWS::StackName" } },  
         { "Key" : "Network", "Value" : "Public" }  
       ]  
     }  
   },  
  
   "InboundHTTPPublicNetworkAclEntry" : {  
     "Type" : "AWS::EC2::NetworkAclEntry",  
     "Properties" : {  
       "NetworkAclId" : { "Ref" : "PublicNetworkAcl" },  
       "RuleNumber" : "100",  
       "Protocol" : "6",  
       "RuleAction" : "allow",  
       "Egress" : "false",  
       "CidrBlock" : "0.0.0.0/0",  
       "PortRange" : { "From" : "80", "To" : "80" }  
     }  
   },  
  
   "InboundHTTPSPublicNetworkAclEntry" : {  
     "Type" : "AWS::EC2::NetworkAclEntry",  
     "Properties" : {  
       "NetworkAclId" : { "Ref" : "PublicNetworkAcl" },  
       "RuleNumber" : "101",  
       "Protocol" : "6",  
       "RuleAction" : "allow",  
       "Egress" : "false",  
       "CidrBlock" : "0.0.0.0/0",  
       "PortRange" : { "From" : "443", "To" : "443" }  
     }  
   },  
  
   "InboundSSHPublicNetworkAclEntry" : {  
     "Type" : "AWS::EC2::NetworkAclEntry",  
     "Properties" : {  
       "NetworkAclId" : { "Ref" : "PublicNetworkAcl" },  
       "RuleNumber" : "102",  
       "Protocol" : "6",  
       "RuleAction" : "allow",  
       "Egress" : "false",  
       "CidrBlock" : "0.0.0.0/0",  
       "PortRange" : { "From" : "22", "To" : "22" }  
     }  
   },  
  
   "InboundEmphemeralPublicNetworkAclEntry" : {  
     "Type" : "AWS::EC2::NetworkAclEntry",  
     "Properties" : {  
       "NetworkAclId" : { "Ref" : "PublicNetworkAcl" },  
       "RuleNumber" : "103",  
       "Protocol" : "6",  
       "RuleAction" : "allow",  
       "Egress" : "false",  
       "CidrBlock" : "0.0.0.0/0",  
       "PortRange" : { "From" : "1024", "To" : "65535" }  
     }  
   },  
  
   "OutboundPublicNetworkAclEntry" : {  
     "Type" : "AWS::EC2::NetworkAclEntry",  
     "Properties" : {  
       "NetworkAclId" : { "Ref" : "PublicNetworkAcl" },  
       "RuleNumber" : "100",  
       "Protocol" : "6",  
       "RuleAction" : "allow",  
       "Egress" : "true",  
       "CidrBlock" : "0.0.0.0/0",  
       "PortRange" : { "From" : "0", "To" : "65535" }  
     }  
   },  
  
   "PublicSubnetNetworkAclAssociation1" : {  
     "Type" : "AWS::EC2::SubnetNetworkAclAssociation",  
     "Properties" : {  
       "SubnetId" : { "Ref" : "PublicSubnet1" },  
       "NetworkAclId" : { "Ref" : "PublicNetworkAcl" }  
     }  
   },  
  
"EC2SecurityGroup" : {  
     "Type" : "AWS::EC2::SecurityGroup",  
     "Properties" : {  
       "GroupDescription" : "Enable access to the EC2 host",  
       "VpcId" : { "Ref" : "VPC" },  
       "SecurityGroupIngress" : [  
         { "IpProtocol" : "tcp", "FromPort" : "22",  "ToPort" : "22",  "CidrIp" : "0.0.0.0/0" },  
         { "IpProtocol" : "tcp", "FromPort" : "80",  "ToPort" : "80",  "CidrIp" : "0.0.0.0/0" },  
         { "IpProtocol" : "tcp", "FromPort" : "443",  "ToPort" : "443",  "CidrIp" : "0.0.0.0/0" },  
         { "IpProtocol" : "icmp", "FromPort" : "-1",  "ToPort" : "-1",  "CidrIp" : "0.0.0.0/0" }  
       ]  
     }  
   },  
  "SGBaseIngress": {  
     "Type": "AWS::EC2::SecurityGroupIngress",  
     "Properties": {  
       "GroupId": { "Ref": "EC2SecurityGroup" },  
       "IpProtocol": "tcp",  
       "FromPort": "80",  
       "ToPort": "80",  
       "SourceSecurityGroupId": { "Ref": "EC2SecurityGroup" }  
     }  
   },  
   "ansible": {  
     "Type": "AWS::IAM::User",  
     "Properties": {  
       "Path": "/",  
       "ManagedPolicyArns" : [ "arn:aws:iam::aws:policy/AmazonEC2FullAccess" ],  
       "UserName": "ansible"  
     }  
   },  
   "ansiblekey" : {  
   "Type" : "AWS::IAM::AccessKey",  
   "Properties" : {  
      "UserName" : { "Ref" : "ansible" }  
      }  
   },  
   "Node1" : {  
      "Type" : "AWS::EC2::Instance",  
      "Properties" : {  
          "InstanceType" : "t3.micro",  
          "ImageId" : "ami-09eb574578ce81bb9",  
          "Tags" : [ {"Key" : "Name", "Value" : "Leo"}],  
          "UserData"       : { "Fn::Base64" : { "Fn::Join" : ["", [  
              "#!/bin/bash\n",  
              "/bin/echo 'E3mXx9v%' | /bin/passwd cloud\_user --stdin\n",  
              "/bin/echo \"StrictHostKeyChecking no\" >> /etc/ssh/ssh\_config\n",  
              "sed -i s/requiretty/\\!requiretty/ /etc/sudoers\n",  
              "hostnamectl set-hostname node1\n"  
          ]]}},  
          "NetworkInterfaces" : [{  
              "GroupSet"                 : [{ "Ref" : "EC2SecurityGroup" }],  
              "AssociatePublicIpAddress": "true",  
              "DeviceIndex"              : "0",  
              "DeleteOnTermination"      : "true",  
              "SubnetId"                 : { "Ref" : "PublicSubnet1" }  
          }]  
      }  
   },  
   "Control" : {  
       "Type" : "AWS::EC2::Instance",  
       "Properties" : {  
           "InstanceType" : "t3.micro",  
           "ImageId" : "ami-09eb574578ce81bb9",  
           "UserData"       : { "Fn::Base64" : { "Fn::Join" : ["", [  
               "#!/bin/bash\n",  
               "/bin/echo 'E3mXx9v%' | /bin/passwd root --stdin\n",  
               "/bin/echo 'E3mXx9v%' | /bin/passwd cloud\_user --stdin\n",  
               "/bin/echo \"StrictHostKeyChecking no\" >> /etc/ssh/ssh\_config\n",  
               "sed -i s/requiretty/\\!requiretty/ /etc/sudoers\n",  
               "hostnamectl set-hostname control\n",  
  
               "yum install -y epel-release\n",  
               "yum install -y ansible sshpass\n",  
               "yum install -y python-boto python-boto3\n",  
  
               "/sbin/useradd ansible\n",  
               "/bin/echo 'E3mXx9v%' | /bin/passwd ansible --stdin\n",  
               "/bin/echo \"ansible        ALL=(ALL)       NOPASSWD: ALL\" >> /etc/sudoers\n",  
               "sudo -u ansible /bin/mkdir -p /home/ansible/.ssh\n",  
               "sudo -u ansible /bin/ssh-keygen -q -N \"\" -f /home/ansible/.ssh/id\_rsa\n",  
               "sudo -u ansible sshpass -p 'E3mXx9v%' ssh-copy-id -i /home/ansible/.ssh/id\_rsa.pub ansible@localhost\n",  
               "echo \"localhost\" >> /etc/ansible/hosts\n",  
  
               "ACCESS=",  
               {"Ref" : "ansiblekey" },  
               "\n",  
               "SECRET=",  
               {"Fn::GetAtt" : [  
                   "ansiblekey",  
                   "SecretAccessKey"  
                 ]  
               },  
               "\n",  
  
               "echo \"export AWS\_ACCESS\_KEY\_ID='KEYACCESS'\" > /home/ansible/accesstemplate\n",  
               "echo \"export AWS\_SECRET\_ACCESS\_KEY='KEYSECRET'\" > /home/ansible/secrettemplate\n",  
               "echo \"#!/bin/sh\"  >> /home/ansible/keys.sh\n",  
               "sed -e \"s|KEYACCESS|$ACCESS|\" /home/ansible/accesstemplate  >> /home/ansible/keys.sh\n",  
               "sed -e \"s|KEYSECRET|$SECRET|\" /home/ansible/secrettemplate  >> /home/ansible/keys.sh\n",  
               "echo \"export AWS\_REGION='us-east-1'\"  >> /home/ansible/keys.sh\n",  
               "chmod 700 /home/ansible/keys.sh\n",  
               "rm -f /home/ansible/accesstemplate /home/ansible/secrettemplate\n",  
  
               "echo \"AWS\_ACCESS\_KEY\_ID: KEYACCESS\" > /home/ansible/accesstemplate\n",  
               "echo \"AWS\_SECRET\_ACCESS\_KEY: KEYSECRET\" > /home/ansible/secrettemplate\n",  
               "sed -e \"s|KEYACCESS|$ACCESS|\" /home/ansible/accesstemplate  >> /home/ansible/keys.yml\n",  
               "sed -e \"s|KEYSECRET|$SECRET|\" /home/ansible/secrettemplate  >> /home/ansible/keys.yml\n",  
               "echo \"AWS\_REGION: us-east-1\"  >> /home/ansible/keys.yml\n",  
               "chmod 600 /home/ansible/keys.yml\n",  
               "rm -f /home/ansible/accesstemplate /home/ansible/secrettemplate\n",  
  
               "chown ansible.ansible /home/ansible/\*\n",  
               "echo \"finished\" > /home/cloud\_user/.done\n"  
           ]]}},  
           "NetworkInterfaces" : [{  
               "GroupSet"                 : [{ "Ref" : "EC2SecurityGroup" }],  
               "AssociatePublicIpAddress": "true",  
               "DeviceIndex"              : "0",  
               "DeleteOnTermination"      : "true",  
               "SubnetId"                 : { "Ref" : "PublicSubnet1" }  
           }]  
       }  
   }  
  },  
 "Outputs" : {  
    "pubIpAddress1" : {  
       "Description" : "Public IP of Ansible Control Node",  
       "Value" : { "Fn::GetAtt" : [ "Control", "PublicIp" ] }  
     },  
     "privIpAddress3" : {  
       "Description" : "Private IP of Ansible Control Node",  
       "Value" : { "Fn::GetAtt" : [ "Control", "PrivateIp" ] }  
     }  
 }  
}

## Ansible Facts in AWS



Ansible Facts in AWS  
Introduction  
For just about every Ansible module that performs an AWS task, there is a corresponding module for collecting facts regarding the related AWS component. A thorough understanding of the AWS principles in Ansible can help with implementing automation. This exercise promotes exploration of the facts provided for various AWS-related modules.  
  
Instructions  
To prepare for a possible security audit, we have been tasked with building some automation that can make a list of firewall rules on a target EC2 instance by tag, as well as show affiliated VPC and Subnet information. We will be developing the automation using Ansible in an AWS sandbox environment. We can gather the needed information from Ansible facts.  
  
This is the information that needs to be in the report:  
  
The VPC ID  
The subnet ID affiliated with the VPC  
The ip\_permissions fact of the Security Group affiliated with an EC2 instance that has the tag Name with the value Leo  
From the Ansible Control node:  
  
Create and edit /home/ansible/report.yml to write the required information to the file /home/ansible/report.txt.  
The file /home/ansible/report.txt should resemble the following:  
  VPC ID: vpc-0058f82290a1b4f91  
  Subnet ID: subnet-046390cbfd8fa8747  
  Security Group Rule Set: [{u'from\_port': 80, u'ip\_protocol': u'tcp', u'to\_port': 80, u'ip\_ranges': [{u'cidr\_ip': u'0.0.0.0/0'}], u'prefix\_list\_ids': [], u'ipv6\_ranges': [], u'user\_id\_group\_pairs': [{u'group\_id': u'sg-054ddc7992d607a8c', u'user\_id': u'041840987519'}]}, {u'from\_port': 22, u'ip\_protocol': u'tcp', u'to\_port': 22, u'ip\_ranges': [{u'cidr\_ip': u'0.0.0.0/0'}], u'prefix\_list\_ids': [], u'ipv6\_ranges': [], u'user\_id\_group\_pairs': []}, {u'from\_port': 443, u'ip\_protocol': u'tcp', u'to\_port': 443, u'ip\_ranges': [{u'cidr\_ip': u'0.0.0.0/0'}], u'prefix\_list\_ids': [], u'ipv6\_ranges': [], u'user\_id\_group\_pairs': []}, {u'from\_port': -1, u'ip\_protocol': u'icmp', u'to\_port': -1, u'ip\_ranges': [{u'cidr\_ip': u'0.0.0.0/0'}], u'prefix\_list\_ids': [], u'ipv6\_ranges': [], u'user\_id\_group\_pairs': []}]  
Run the playbook /home/ansible/report.yml to validate your work.  
The Ansible control node has been configured and already has Ansible installed. The control node also has a system user named ansible configured with SSH access keys and necessary system privileges.  
  
An IAM user ansible has been created on the provided AWS sandbox account. The access keys for the ansible IAM user are stored in /home/ansible/keys.sh and /home/ansible/keys.yml for whichever authentication method we prefer. The ansible IAM user has appropriate permissions to perform the required task.  
  
The default Ansible inventory has been configured to include the Ansible control host as localhost.  
  
Logging In  
Use the hands-on lab page to get the public IP of the cloud server we need to log into (making sure to use cloud\_user as a username), then switch to the ansible user. The password for the two users is the same.  
  
Create and Edit /home/ansible/report.yml and Add Ansible Tasks That Output the Required Values into report.txt  
Create and edit the playbook such that it resembles the following:  
  
- hosts: localhost  
  gather\_facts: no  
  vars\_files:  
    - /home/ansible/keys.yml  
  tasks:  
    - name: Get VPC facts  
      ec2\_vpc\_net\_facts:  
        aws\_access\_key: "{{ AWS\_ACCESS\_KEY\_ID }}"  
        aws\_secret\_key: "{{ AWS\_SECRET\_ACCESS\_KEY }}"  
        region: "{{ AWS\_REGION }}"  
      register: vpc\_facts  
    - name: Add line to report.txt  
      lineinfile:  
        path: /home/ansible/report.txt  
        line: "VPC ID: {{ vpc\_facts.vpcs[0].vpc\_id }}"  
        create: yes  
  
    - name: Get VPC Subnet Facts  
      ec2\_vpc\_subnet\_facts:  
        aws\_access\_key: "{{ AWS\_ACCESS\_KEY\_ID }}"  
        aws\_secret\_key: "{{ AWS\_SECRET\_ACCESS\_KEY }}"  
        region: "{{ AWS\_REGION }}"  
        filters:  
          vpc-id: "{{ vpc\_facts.vpcs[0].vpc\_id }}"  
      register: subnet\_facts  
    - name: Add line to report.txt  
      lineinfile:  
        path: /home/ansible/report.txt  
        line: "Subnet ID: {{ subnet\_facts.subnets[0].subnet\_id }}"  
  
    - name: Get EC2 instance facts  
      ec2\_instance\_facts:  
        aws\_access\_key: "{{ AWS\_ACCESS\_KEY\_ID }}"  
        aws\_secret\_key: "{{ AWS\_SECRET\_ACCESS\_KEY }}"  
        region: "{{ AWS\_REGION }}"  
        filters:  
          tag:Name: "Leo"  
      register: ec2\_facts  
  
    - name: Get Security Group facts  
      ec2\_group\_facts:  
        aws\_access\_key: "{{ AWS\_ACCESS\_KEY\_ID }}"  
        aws\_secret\_key: "{{ AWS\_SECRET\_ACCESS\_KEY }}"  
        region: "{{ AWS\_REGION }}"  
        filters:  
          group-id: "{{ ec2\_facts.instances[0].security\_groups[0].group\_id }}"  
      register: security\_group\_facts  
    - name: Add line to report.txt  
      lineinfile:  
        path: /home/ansible/report.txt  
        line: "Security Group Rule Set: {{ security\_group\_facts.security\_groups[0].ip\_permissions }}"  
Run the Modified /home/ansible/report.yml to Validate That the Playbook Successfully Generates the Report  
Run the following command:  
ansible-playbook /home/ansible/report.yml  
Conclusion  
Now if we run cat report.txt, we're going to see all of the information that we've been asked to get. We can hand this text file over to who needs it, because we're done. Congratulations!

Additional Resources  
To prepare for a possible security audit, we have been tasked with building some automation that can make a list of firewall rules on a target EC2 instance by tag, as well as show affiliated VPC and Subnet information. We will be developing the automation using Ansible in an AWS sandbox environment. We can gather the needed information from Ansible facts.  
  
This is the information that needs to be in the report:  
  
The VPC ID  
The subnet ID affiliated with the VPC  
The ip\_permissions fact of the Security Group affiliated with an EC2 instance that has the tag Name with the value Leo  
From the Ansible Control node:  
  
Create and edit /home/ansible/report.yml to write the required information to the file /home/ansible/report.txt.  
The file /home/ansible/report.txt should resemble the following:  
  VPC ID: vpc-0058f82290a1b4f91  
  Subnet ID: subnet-046390cbfd8fa8747  
  Security Group Rule Set: [{u'from\_port': 80, u'ip\_protocol': u'tcp', u'to\_port': 80, u'ip\_ranges': [{u'cidr\_ip': u'0.0.0.0/0'}], u'prefix\_list\_ids': [], u'ipv6\_ranges': [], u'user\_id\_group\_pairs': [{u'group\_id': u'sg-054ddc7992d607a8c', u'user\_id': u'041840987519'}]}, {u'from\_port': 22, u'ip\_protocol': u'tcp', u'to\_port': 22, u'ip\_ranges': [{u'cidr\_ip': u'0.0.0.0/0'}], u'prefix\_list\_ids': [], u'ipv6\_ranges': [], u'user\_id\_group\_pairs': []}, {u'from\_port': 443, u'ip\_protocol': u'tcp', u'to\_port': 443, u'ip\_ranges': [{u'cidr\_ip': u'0.0.0.0/0'}], u'prefix\_list\_ids': [], u'ipv6\_ranges': [], u'user\_id\_group\_pairs': []}, {u'from\_port': -1, u'ip\_protocol': u'icmp', u'to\_port': -1, u'ip\_ranges': [{u'cidr\_ip': u'0.0.0.0/0'}], u'prefix\_list\_ids': [], u'ipv6\_ranges': [], u'user\_id\_group\_pairs': []}]  
Run the playbook /home/ansible/report.yml to validate your work.  
The Ansible control node has been configured and already has Ansible installed. The control node also has a system user named ansible configured with SSH access keys and necessary system privileges.  
  
An IAM user ansible has been created on the provided AWS sandbox account. The access keys for the ansible IAM user are stored in /home/ansible/keys.sh and /home/ansible/keys.yml for whichever authentication method we prefer. The ansible IAM user has appropriate permissions to perform the required task.  
  
The default Ansible inventory has been configured to include the Ansible control host as localhost.  
  
Learning Objectives  
0 of 2 completed  
  
Create and Edit `/home/ansible/report.yml` and Add Ansible Tasks That Output the Required Values into `report.txt`  
  
After logging into the EC2 instance, run su - ansible to become the ansible user. The password is the same as it is for cloud\_user.  
  
Create and edit the playbook such that it resembles the following:  
  
- hosts: localhost  
gather\_facts: no  
vars\_files:  
- /home/ansible/keys.yml  
tasks:  
- name: Get VPC facts  
 ec2\_vpc\_net\_facts:  
   aws\_access\_key: "{{ AWS\_ACCESS\_KEY\_ID }}"  
   aws\_secret\_key: "{{ AWS\_SECRET\_ACCESS\_KEY }}"  
   region: "{{ AWS\_REGION }}"  
 register: vpc\_facts  
- name: Add line to facts.txt  
 lineinfile:  
   path: /home/ansible/facts.txt  
   line: "VPC ID: {{ vpc\_facts.vpcs[0].vpc\_id }}"  
  
- name: Get VPC Subnet Facts  
 ec2\_vpc\_subnet\_facts:  
   aws\_access\_key: "{{ AWS\_ACCESS\_KEY\_ID }}"  
   aws\_secret\_key: "{{ AWS\_SECRET\_ACCESS\_KEY }}"  
   region: "{{ AWS\_REGION }}"  
   filters:  
     vpc-id: "{{ vpc\_facts.vpcs[0].vpc\_id }}"  
 register: subnet\_facts  
- name: Add line to facts.txt  
 lineinfile:  
   path: /home/ansible/facts.txt  
   line: "Subnet ID: {{ subnet\_facts.subnets[0].subnet\_id }}"  
  
- name: Get EC2 instance facts  
 ec2\_instance\_facts:  
   aws\_access\_key: "{{ AWS\_ACCESS\_KEY\_ID }}"  
   aws\_secret\_key: "{{ AWS\_SECRET\_ACCESS\_KEY }}"  
   region: "{{ AWS\_REGION }}"  
   filters:  
     tag:Name: "Leo"  
 register: ec2\_facts  
  
- name: Get Security Group facts  
 ec2\_group\_facts:  
   aws\_access\_key: "{{ AWS\_ACCESS\_KEY\_ID }}"  
   aws\_secret\_key: "{{ AWS\_SECRET\_ACCESS\_KEY }}"  
   region: "{{ AWS\_REGION }}"  
   filters:  
     group-id: "{{ ec2\_facts.instances[0].security\_groups[0].group\_id }}"  
 register: security\_group\_facts  
- name: Add line to facts.txt  
 lineinfile:  
   path: /home/ansible/facts.txt  
   line: "Security Group Rule Set: {{ security\_group\_facts.security\_groups[0].ip\_permissions }}"  
Run the Modified `/home/ansible/report.yml` to Validate That the Playbook Successfully Generates the Report  
  
Log into the Ansible control node as the ansible user.  
Run the following command:  
ansible-playbook /home/ansible/report.yml

HOST SERVER  :

[root@control cloud\_user]# curl 169.254.169.254/latest/user-data

#!/bin/bash

/bin/echo 'XG2lEtK^' | /bin/passwd root —stdin

/bin/echo 'XG2lEtK^' | /bin/passwd cloud\_user —stdin

/bin/echo "StrictHostKeyChecking no" >> /etc/ssh/ssh\_config

sed -i s/requiretty/\!requiretty/ /etc/sudoers

hostnamectl set-hostname control

yum install -y epel-release

yum install -y ansible sshpass

yum install -y python-boto python-boto3

/sbin/useradd ansible

/bin/echo 'XG2lEtK^' | /bin/passwd ansible —stdin

/bin/echo "ansible ALL=(ALL) NOPASSWD: ALL" >> /etc/sudoers

sudo -u ansible /bin/mkdir -p /home/ansible/.ssh

sudo -u ansible /bin/ssh-keygen -q -N "" -f /home/ansible/.ssh/id\_rsa

sudo -u ansible sshpass -p 'XG2lEtK^' ssh-copy-id -i /home/ansible/.ssh/id\_rsa.pub ansible@localhost

echo "localhost" >> /etc/ansible/hosts

echo "AWS\_ACCESS\_KEY\_ID: KEYACCESS" > /home/ansible/accesstemplate

ACCESS=AKIAVKBQB3XPHWYNH4W2

SECRET=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

echo "export AWS\_ACCESS\_KEY\_ID='KEYACCESS'" > /home/ansible/accesstemplate

echo "export AWS\_SECRET\_ACCESS\_KEY='KEYSECRET'" > /home/ansible/secrettemplate

echo "#!/bin/sh" >> /home/ansible/keys.sh

sed -e "s|KEYACCESS|$ACCESS|" /home/ansible/accesstemplate >> /home/ansible/keys.sh

sed -e "s|KEYSECRET|$SECRET|" /home/ansible/secrettemplate >> /home/ansible/keys.sh

echo "export AWS\_REGION='us-east-1'" >> /home/ansible/keys.sh

chmod 700 /home/ansible/keys.sh

rm -f /home/ansible/accesstemplate /home/ansible/secrettemplate

echo "AWS\_ACCESS\_KEY\_ID: KEYACCESS" > /home/ansible/accesstemplate

echo "AWS\_SECRET\_ACCESS\_KEY: KEYSECRET" > /home/ansible/secrettemplate

sed -e "s|KEYACCESS|$ACCESS|" /home/ansible/accesstemplate >> /home/ansible/keys.yml

sed -e "s|KEYSECRET|$SECRET|" /home/ansible/secrettemplate >> /home/ansible/keys.yml

echo "AWS\_REGION: us-east-1" >> /home/ansible/keys.yml

chmod 600 /home/ansible/keys.yml

rm -f /home/ansible/accesstemplate /home/ansible/secrettemplate

touch /home/ansible/report.txt

chown ansible.ansible /home/ansible/\*

echo "finished" > /home/cloud\_user/.done

[root@control cloud\_user]# sudo cat /etc/os-release

NAME="CentOS Linux"

VERSION="7 (Core)"

ID="centos"

ID\_LIKE="rhel fedora"

VERSION\_ID="7"

PRETTY\_NAME="CentOS Linux 7 (Core)"

ANSI\_COLOR="0;31"

CPE\_NAME="cpe:/o:centos:centos:7"

HOME\_URL="<https://www.centos.org/>“

BUG\_REPORT\_URL="<https://bugs.centos.org/>“

CENTOS\_MANTISBT\_PROJECT="CentOS-7"

CENTOS\_MANTISBT\_PROJECT\_VERSION="7"

REDHAT\_SUPPORT\_PRODUCT="centos"

REDHAT\_SUPPORT\_PRODUCT\_VERSION="7"

[root@control cloud\_user]# ansible —version

ansible 2.9.27

config file = /etc/ansible/ansible.cfg

configured module search path = [u'/root/.ansible/plugins/modules', u'/usr/share/ansible/plugins/modules']

ansible python module location = /usr/lib/python2.7/site-packages/ansible

executable location = /bin/ansible

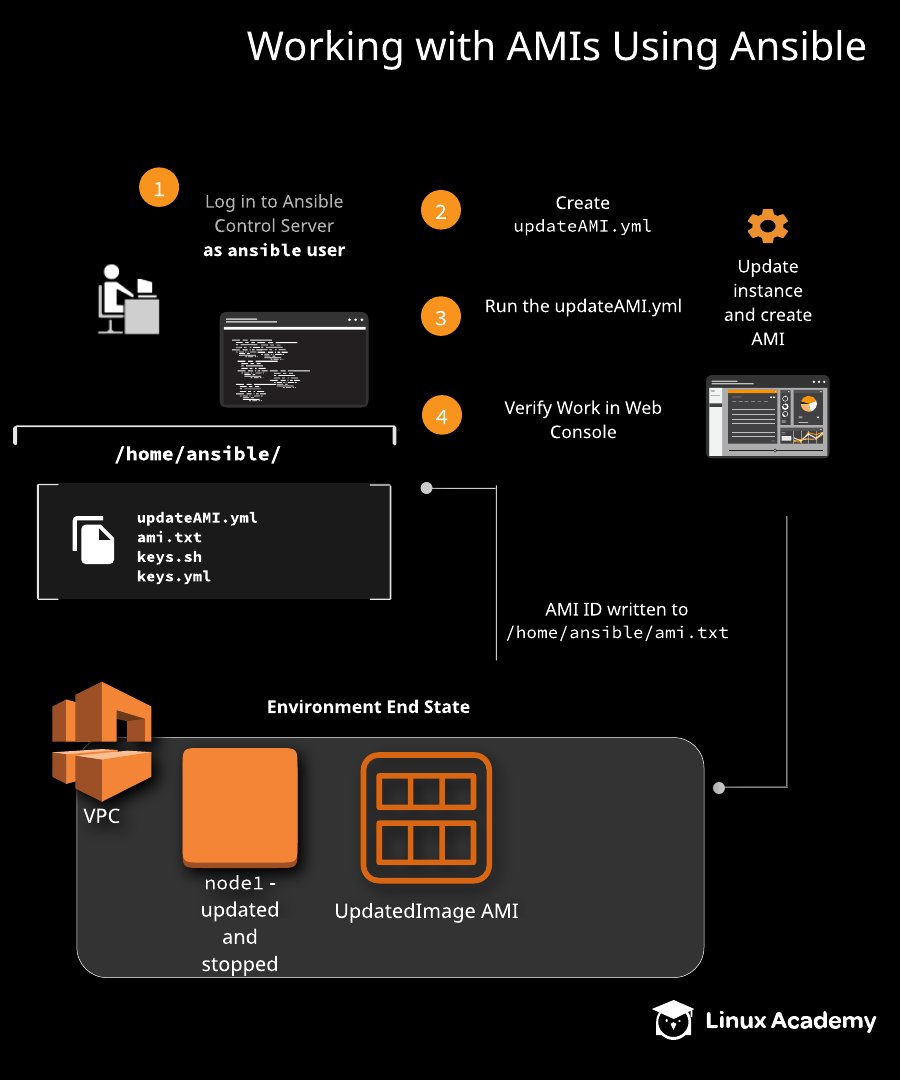
python version = 2.7.5 (default, Nov 16 2020, 22:23:17) [GCC 4.8.5 20150623 (Red Hat 4.8.5-44)]

[root@control cloud\_user]#

CFN

{  
 "AWSTemplateFormatVersion" : "2010-09-09",  
 "Description" : "Ansible Facts in AWS",  
  
 "Mappings" : {  
  
   "SubnetConfig" : {  
     "VPC"     : { "CIDR" : "10.0.0.0/16" },  
     "Public1"  : { "CIDR" : "10.0.1.0/24" }  
   }  
 },  
  
 "Resources" : {  
  
   "VPC" : {  
     "Type" : "AWS::EC2::VPC",  
     "Properties" : {  
  
       "EnableDnsSupport" : "true",  
       "EnableDnsHostnames" : "true",  
       "CidrBlock" : { "Fn::FindInMap" : [ "SubnetConfig", "VPC", "CIDR" ]},  
       "Tags" : [  
        { "Key" : "Name", "Value" : "LinuxAcademy" },  
         { "Key" : "Application", "Value" : { "Ref" : "AWS::StackName" } },  
         { "Key" : "Network", "Value" : "VPC" }  
       ]  
     }  
   },  
  
  
   "PublicSubnet1" : {  
     "Type" : "AWS::EC2::Subnet",  
         "Properties" : {  
       "VpcId" : { "Ref" : "VPC" },  
       "AvailabilityZone" : {  
         "Fn::Select" : [ "0", { "Fn::GetAZs" : "" } ]  
       },  
       "CidrBlock" : { "Fn::FindInMap" : [ "SubnetConfig", "Public1", "CIDR" ]},  
       "Tags" : [  
         { "Key" : "Application", "Value" : { "Ref" : "AWS::StackName" } },  
         { "Key" : "Network", "Value" : "Public1" }  
       ]  
     }  
   },  
  
   "InternetGateway" : {  
     "Type" : "AWS::EC2::InternetGateway",  
     "Properties" : {  
       "Tags" : [  
         { "Key" : "Application", "Value" : { "Ref" : "AWS::StackName" } },  
         { "Key" : "Network", "Value" : "Public" }  
       ]  
     }  
   },  
  
   "GatewayToInternet" : {  
      "Type" : "AWS::EC2::VPCGatewayAttachment",  
      "Properties" : {  
        "VpcId" : { "Ref" : "VPC" },  
        "InternetGatewayId" : { "Ref" : "InternetGateway" }  
      }  
   },  
  
   "PublicRouteTable" : {  
     "Type" : "AWS::EC2::RouteTable",  
     "Properties" : {  
       "VpcId" : { "Ref" : "VPC" },  
       "Tags" : [  
         { "Key" : "Application", "Value" : { "Ref" : "AWS::StackName" } },  
         { "Key" : "Network", "Value" : "Public" }  
       ]  
     }  
   },  
  
   "PublicRoute" : {  
     "Type" : "AWS::EC2::Route",  
     "DependsOn" : "GatewayToInternet",  
     "Properties" : {  
       "RouteTableId" : { "Ref" : "PublicRouteTable" },  
       "DestinationCidrBlock" : "0.0.0.0/0",  
       "GatewayId" : { "Ref" : "InternetGateway" }  
     }  
   },  
  
   "PublicSubnetRouteTableAssociation1" : {  
     "Type" : "AWS::EC2::SubnetRouteTableAssociation",  
     "Properties" : {  
       "SubnetId" : { "Ref" : "PublicSubnet1" },  
       "RouteTableId" : { "Ref" : "PublicRouteTable" }  
     }  
   },  
  
   "PublicNetworkAcl" : {  
     "Type" : "AWS::EC2::NetworkAcl",  
     "Properties" : {  
       "VpcId" : { "Ref" : "VPC" },  
       "Tags" : [  
         { "Key" : "Application", "Value" : { "Ref" : "AWS::StackName" } },  
         { "Key" : "Network", "Value" : "Public" }  
       ]  
     }  
   },  
  
   "InboundHTTPPublicNetworkAclEntry" : {  
     "Type" : "AWS::EC2::NetworkAclEntry",  
     "Properties" : {  
       "NetworkAclId" : { "Ref" : "PublicNetworkAcl" },  
       "RuleNumber" : "100",  
       "Protocol" : "6",  
       "RuleAction" : "allow",  
       "Egress" : "false",  
       "CidrBlock" : "0.0.0.0/0",  
       "PortRange" : { "From" : "80", "To" : "80" }  
     }  
   },  
  
   "InboundHTTPSPublicNetworkAclEntry" : {  
     "Type" : "AWS::EC2::NetworkAclEntry",  
     "Properties" : {  
       "NetworkAclId" : { "Ref" : "PublicNetworkAcl" },  
       "RuleNumber" : "101",  
       "Protocol" : "6",  
       "RuleAction" : "allow",  
       "Egress" : "false",  
       "CidrBlock" : "0.0.0.0/0",  
       "PortRange" : { "From" : "443", "To" : "443" }  
     }  
   },  
  
   "InboundSSHPublicNetworkAclEntry" : {  
     "Type" : "AWS::EC2::NetworkAclEntry",  
     "Properties" : {  
       "NetworkAclId" : { "Ref" : "PublicNetworkAcl" },  
       "RuleNumber" : "102",  
       "Protocol" : "6",  
       "RuleAction" : "allow",  
       "Egress" : "false",  
       "CidrBlock" : "0.0.0.0/0",  
       "PortRange" : { "From" : "22", "To" : "22" }  
     }  
   },  
  
   "InboundEmphemeralPublicNetworkAclEntry" : {  
     "Type" : "AWS::EC2::NetworkAclEntry",  
     "Properties" : {  
       "NetworkAclId" : { "Ref" : "PublicNetworkAcl" },  
       "RuleNumber" : "103",  
       "Protocol" : "6",  
       "RuleAction" : "allow",  
       "Egress" : "false",  
       "CidrBlock" : "0.0.0.0/0",  
       "PortRange" : { "From" : "1024", "To" : "65535" }  
     }  
   },  
  
   "OutboundPublicNetworkAclEntry" : {  
     "Type" : "AWS::EC2::NetworkAclEntry",  
     "Properties" : {  
       "NetworkAclId" : { "Ref" : "PublicNetworkAcl" },  
       "RuleNumber" : "100",  
       "Protocol" : "6",  
       "RuleAction" : "allow",  
       "Egress" : "true",  
       "CidrBlock" : "0.0.0.0/0",  
       "PortRange" : { "From" : "0", "To" : "65535" }  
     }  
   },  
  
   "PublicSubnetNetworkAclAssociation1" : {  
     "Type" : "AWS::EC2::SubnetNetworkAclAssociation",  
     "Properties" : {  
       "SubnetId" : { "Ref" : "PublicSubnet1" },  
       "NetworkAclId" : { "Ref" : "PublicNetworkAcl" }  
     }  
   },  
  
"EC2SecurityGroup" : {  
     "Type" : "AWS::EC2::SecurityGroup",  
     "Properties" : {  
       "GroupDescription" : "Enable access to the EC2 host",  
       "VpcId" : { "Ref" : "VPC" },  
       "SecurityGroupIngress" : [  
         { "IpProtocol" : "tcp", "FromPort" : "22",  "ToPort" : "22",  "CidrIp" : "0.0.0.0/0" },  
         { "IpProtocol" : "tcp", "FromPort" : "80",  "ToPort" : "80",  "CidrIp" : "0.0.0.0/0" },  
         { "IpProtocol" : "tcp", "FromPort" : "443",  "ToPort" : "443",  "CidrIp" : "0.0.0.0/0" },  
         { "IpProtocol" : "icmp", "FromPort" : "-1",  "ToPort" : "-1",  "CidrIp" : "0.0.0.0/0" }  
       ]  
     }  
   },  
  "SGBaseIngress": {  
     "Type": "AWS::EC2::SecurityGroupIngress",  
     "Properties": {  
       "GroupId": { "Ref": "EC2SecurityGroup" },  
       "IpProtocol": "tcp",  
       "FromPort": "80",  
       "ToPort": "80",  
       "SourceSecurityGroupId": { "Ref": "EC2SecurityGroup" }  
     }  
   },  
   "ansible": {  
     "Type": "AWS::IAM::User",  
     "Properties": {  
       "Path": "/",  
       "ManagedPolicyArns" : [ "arn:aws:iam::aws:policy/AmazonEC2ReadOnlyAccess", "arn:aws:iam::aws:policy/AmazonVPCReadOnlyAccess" ],  
       "UserName": "ansible"  
     }  
   },  
   "ansiblekey" : {  
   "Type" : "AWS::IAM::AccessKey",  
   "Properties" : {  
      "UserName" : { "Ref" : "ansible" }  
      }  
   },  
   "Node1" : {  
      "Type" : "AWS::EC2::Instance",  
      "Properties" : {  
          "InstanceType" : "t3.micro",  
          "ImageId" : "ami-09eb574578ce81bb9",  
          "Tags" : [ {"Key" : "Name", "Value" : "Leo"}],  
          "UserData"       : { "Fn::Base64" : { "Fn::Join" : ["", [  
              "#!/bin/bash\n",  
              "/bin/echo 'XG2lEtK^' | /bin/passwd cloud\_user --stdin\n",  
              "/bin/echo \"StrictHostKeyChecking no\" >> /etc/ssh/ssh\_config\n",  
              "sed -i s/requiretty/\\!requiretty/ /etc/sudoers\n",  
              "hostnamectl set-hostname node1\n"  
          ]]}},  
          "NetworkInterfaces" : [{  
              "GroupSet"                 : [{ "Ref" : "EC2SecurityGroup" }],  
              "AssociatePublicIpAddress": "true",  
              "DeviceIndex"              : "0",  
              "DeleteOnTermination"      : "true",  
              "SubnetId"                 : { "Ref" : "PublicSubnet1" }  
          }]  
      }  
   },  
   "Control" : {  
       "Type" : "AWS::EC2::Instance",  
       "Properties" : {  
           "InstanceType" : "t3.micro",  
           "ImageId" : "ami-09eb574578ce81bb9",  
           "UserData"       : { "Fn::Base64" : { "Fn::Join" : ["", [  
               "#!/bin/bash\n",  
               "/bin/echo 'XG2lEtK^' | /bin/passwd root --stdin\n",  
               "/bin/echo 'XG2lEtK^' | /bin/passwd cloud\_user --stdin\n",  
               "/bin/echo \"StrictHostKeyChecking no\" >> /etc/ssh/ssh\_config\n",  
               "sed -i s/requiretty/\\!requiretty/ /etc/sudoers\n",  
               "hostnamectl set-hostname control\n",  
               "yum install -y epel-release\n",  
               "yum install -y ansible sshpass\n",  
               "yum install -y python-boto python-boto3\n",  
  
               "/sbin/useradd ansible\n",  
               "/bin/echo 'XG2lEtK^' | /bin/passwd ansible --stdin\n",  
               "/bin/echo \"ansible        ALL=(ALL)       NOPASSWD: ALL\" >> /etc/sudoers\n",  
               "sudo -u ansible /bin/mkdir -p /home/ansible/.ssh\n",  
               "sudo -u ansible /bin/ssh-keygen -q -N \"\" -f /home/ansible/.ssh/id\_rsa\n",  
               "sudo -u ansible sshpass -p 'XG2lEtK^' ssh-copy-id -i /home/ansible/.ssh/id\_rsa.pub ansible@localhost\n",  
  
               "echo \"localhost\" >> /etc/ansible/hosts\n",  
               "echo \"AWS\_ACCESS\_KEY\_ID: KEYACCESS\" > /home/ansible/accesstemplate\n",  
  
               "ACCESS=",  
               {"Ref" : "ansiblekey" },  
               "\n",  
               "SECRET=",  
               {"Fn::GetAtt" : [  
                   "ansiblekey",  
                   "SecretAccessKey"  
                 ]  
               },  
               "\n",  
  
               "echo \"export AWS\_ACCESS\_KEY\_ID='KEYACCESS'\" > /home/ansible/accesstemplate\n",  
               "echo \"export AWS\_SECRET\_ACCESS\_KEY='KEYSECRET'\" > /home/ansible/secrettemplate\n",  
               "echo \"#!/bin/sh\"  >> /home/ansible/keys.sh\n",  
               "sed -e \"s|KEYACCESS|$ACCESS|\" /home/ansible/accesstemplate  >> /home/ansible/keys.sh\n",  
               "sed -e \"s|KEYSECRET|$SECRET|\" /home/ansible/secrettemplate  >> /home/ansible/keys.sh\n",  
               "echo \"export AWS\_REGION='us-east-1'\"  >> /home/ansible/keys.sh\n",  
               "chmod 700 /home/ansible/keys.sh\n",  
               "rm -f /home/ansible/accesstemplate /home/ansible/secrettemplate\n",  
  
               "echo \"AWS\_ACCESS\_KEY\_ID: KEYACCESS\" > /home/ansible/accesstemplate\n",  
               "echo \"AWS\_SECRET\_ACCESS\_KEY: KEYSECRET\" > /home/ansible/secrettemplate\n",  
               "sed -e \"s|KEYACCESS|$ACCESS|\" /home/ansible/accesstemplate  >> /home/ansible/keys.yml\n",  
               "sed -e \"s|KEYSECRET|$SECRET|\" /home/ansible/secrettemplate  >> /home/ansible/keys.yml\n",  
               "echo \"AWS\_REGION: us-east-1\"  >> /home/ansible/keys.yml\n",  
               "chmod 600 /home/ansible/keys.yml\n",  
               "rm -f /home/ansible/accesstemplate /home/ansible/secrettemplate\n",  
  
               "touch /home/ansible/report.txt\n",  
               "chown ansible.ansible /home/ansible/\*\n",  
               "echo \"finished\" > /home/cloud\_user/.done\n"  
           ]]}},  
           "NetworkInterfaces" : [{  
               "GroupSet"                 : [{ "Ref" : "EC2SecurityGroup" }],  
               "AssociatePublicIpAddress": "true",  
               "DeviceIndex"              : "0",  
               "DeleteOnTermination"      : "true",  
               "SubnetId"                 : { "Ref" : "PublicSubnet1" }  
           }]  
       }  
   }  
  },  
 "Outputs" : {  
    "pubIpAddress1" : {  
       "Description" : "Public IP of Ansible Control Node",  
       "Value" : { "Fn::GetAtt" : [ "Control", "PublicIp" ] }  
     },  
     "privIpAddress3" : {  
       "Description" : "Private IP of Ansible Control Node",  
       "Value" : { "Fn::GetAtt" : [ "Control", "PrivateIp" ] }  
     }  
  
  }  
}

## Working with AMIs Using Ansible



Working with AMIs Using Ansible  
Introduction  
Image creation in AWS provides for both simplified system management and improved deployment performance. Ansible can be leveraged to automate AMI upkeep, and we will be doing just that in this exercise!  
  
Instructions  
In an effort to save some time, you have decided to automate the process for maintaining AMIs using Ansible. You have a deployed EC2 instance on which the AMI is to be based. The instance is configured for access from your Ansible control node using the hostname node1.  
  
Prior to creating the AMI, the playbook needs to make the following changes to node1:  
  
Install the latest updates using yum.  
Use the lineinfile module to update the file /home/ansible/image.txt with the line Image updated <current.date.use.ansible.facts>.  
From the Ansible Control node:  
  
Create the playbook /home/ansible/updateAMI.yml to perform the following tasks:  
Update all packages on node1 using the yum module.  
Insert the line "Image updated <current.date>" into the file on /home/ansible/image.txt on "node1".  
Stop the EC2 instance for node1 using AWS.  
Create a new AMI based on node1.  
Write the AMI ID to the file /home/ansible/ami.txt.  
Run the playbook /home/ansible/updateAMI.yml.  
Verify your work in the AWS Web Console.  
The Ansible control node has been configured and already has Ansible installed. The control node also has a system user named ansible configured with SSH access keys and necessary system privileges.  
  
An IAM user ansible has been created on the provided AWS sandbox account. The access keys for the ansible IAM user are stored in /home/ansible/keys.sh and /home/ansible/keys.yml for whichever authentication method we prefer. The ansible IAM user has appropriate permissions to perform the required task.  
  
The default Ansible inventory has been configured to include the Ansible control host as localhost.  
  
Logging In  
Use the hands-on lab page to get the public IP of the cloud server we need to log into (making sure to use cloud\_user as a username), then switch to the ansible user. The password for the two users is the same.  
  
Create /home/ansible/updateAMI.yml, and Add an Ansible Play that Updates the Software on node1 and Updates /home/ansible/image.txt  
In our solution, we'll complete the required steps and add a task for collecting facts from EC2 metadata, which will help us collect the instance ID of node1.  
  
Edit the playbook such that it resembles the following:  
  
- hosts: node1  
  become: yes  
  tasks:  
    - name: Update packages  
      yum:  
        name: "\*"  
        state: latest  
    - name: Update image.txt  
      lineinfile:  
        path: /home/ansible/image.txt  
        line: "Image updated {{ ansible\_date\_time.date }}"  
    - name: Gather facts  
      ec2\_metadata\_facts:  
Edit the /home/ansible/updateAMI.yml and Add an Ansible Play that Collects the instance\_id of node1, stops node1, Creates the AMI as Described in the Instructions, and Stores the AMI ID in the File /home/ansible/ami.txt  
We will be using the facts we collected in the first play to satisfy the objectives for this task.  
  
Edit the playbook such that it resembles the following:  
  
- hosts: localhost  
  tasks:  
    - name: Stop node1  
      local\_action: ec2  
      args:  
        region: us-east-1  
        state: stopped  
        instance\_id: "{{ hostvars['node1'].ansible\_ec2\_instance\_id }}"  
        wait: yes  
    - name: Create AMI  
      local\_action: ec2\_ami  
      args:  
        state: present  
        instance\_id: "{{ hostvars['node1'].ansible\_ec2\_instance\_id }}"  
        name: UpdatedImage  
      register: ami\_output  
    - name: Write AMI info to file  
      lineinfile:  
        create: yes  
        path: /home/ansible/ami.txt  
        line: "{{ ami\_output.image\_id }}"  
Run /home/ansible/updateAMI.yml to Perform the Required Tasks and Then Log into the AWS Console to Verify Your Work  
Before we can run this playbook, we've got to set a few environment variables. We can do that by sourcing a file that's sitting in our home directory:  
  
source keys.sh  
Now run the following command: ansible-playbook /home/ansible/updateAMI.yml  
  
Log into the AWS Console, and in the EC2 Dashboard (find it by searching for EC2 in the Find Services search box) confirm the new instance's existence and state.  
  
It might be best to wait a bit before checking. Once everything is finished processing though, we'll see a Leo instance that's stopped, and a new one that is running.  
We can also look down at AMIs (down in the Images section of the menu on the left side of the screen) and see our new UpdatedImage there.  
  
Conclusion  
We've done it. Using Ansible, we've automated the process of updating our AMIs. Congratulations!

Additional Resources  
NOTE: After starting the lab, wait 1 to 2 minutes before trying to log into the instances.  
  
In an effort to save some time, you have decided to automate the process for maintaining AMIs using Ansible. You have a deployed EC2 instance on which the AMI is to be based. The instance is configured for access from your Ansible control node using the hostname "node1".  
  
Prior to creating the AMI, the playbook needs to make the following changes to "node1":  
  
Install the latest updates using the yum module.  
Use the lineinfile module to update the file /home/ansible/image.txt with the line "Image updated <current.date.use.ansible.facts>".  
From the Ansible Control node:  
  
Create the playbook /home/ansible/updateAMI.yml to perform the following tasks:  
Update all packages on node1 using the yum module.  
Insert the line "Image updated <current.date>" into the file on /home/ansible/image.txt on "node1".  
Stop the EC2 instance for node1 using AWS.  
Create a new AMI based on node1.  
Write the AMI ID to the file /home/ansible/ami.txt.  
Run the playbook /home/ansible/updateAMI.yml.  
Verify your work in the AWS Web Console.  
The Ansible control node has been configured for you and has already had Ansible installed. The control node also has a system user named ansible configured with ssh access keys and necessary system privileges.  
  
An IAM user called ansible has been created on the provided AWS sandbox account. The access keys for the ansible IAM user are stored in /home/ansible/keys.sh and /home/ansible/keys.yml for which ever authentication method you prefer. The ansible IAM user has appropriate permissions to perform the required task.  
  
The default Ansible inventory has been configured to include a the Ansible control host as 'localhost' and "node1".  
  
Learning Objectives  
0 of 3 completed  
  
Create `/home/ansible/updateAMI.yml`, and Add an Ansible Play that Updates the Software on `node1` Then Updates `/home/ansible/image.txt`, as Described in the Instructions  
  
In our solution, we'll complete the required steps and add a task for collecting facts from EC2 metadata, which will help us collect the instance ID of node1.  
  
Edit the playbook such that it resembles the following:  
  
- hosts: node1  
  become: yes  
  tasks:  
    - name: Update packages  
      yum:  
        name: "\*"  
        state: latest  
    - name: Update image.txt  
      lineinfile:  
        path: /home/ansible/image.txt  
        line: "Image updated {{ ansible\_date\_time.date }}"  
    - name: Gather facts  
      ec2\_metadata\_facts:  
Edit the `/home/ansible/updateAMI.yml` and Add an Ansible Play that Collects the instance\_id of `node1`, stops `node1`, Creates the AMI as Described in the Instructions, and Stores the AMI ID in the File `/home/ansible/ami.txt`  
  
We will be using the facts we collected in the first play to satisfy the objectives for this task.  
  
Edit the playbook such that it resembles the following:  
  
- hosts: localhost  
  tasks:  
    - name: Stop node1  
      local\_action: ec2  
      args:  
        region: us-east-1  
        state: stopped  
        instance\_id: "{{ hostvars['node1'].ansible\_ec2\_instance\_id }}"  
        wait: yes  
    - name: Create AMI  
      local\_action: ec2\_ami  
      args:  
        state: present  
        instance\_id: "{{ hostvars['node1'].ansible\_ec2\_instance\_id }}"  
        name: UpdatedImage  
      register: ami\_output  
    - name: Write AMI info to file  
      lineinfile:  
        create: yes  
        path: /home/ansible/ami.txt  
        line: "{{ ami\_output.image\_id }}"  
Run `/home/ansible/updateAMI.yml` to Perform the Required Tasks and Then Log into the AWS Console to Verify Your Work  
  
Change some environment variables:  
source keys.sh  
Run the playbook:  
ansible-playbook /home/ansible/updateAMI.yml  
Log into the AWS Console and confirm the new EC2 instance:  
Search for "EC2" in the AWS console search and select the EC2 dashboard.  
Confirm the AMI was updated:  
Select AMIs from the menu on the left.

HOST :

[root@control cloud\_user]# curl 169.254.169.254/latest/user-data

#!/bin/bash

/bin/echo 'S[W=u|3E' | /bin/passwd root —stdin

/bin/echo 'S[W=u|3E' | /bin/passwd cloud\_user —stdin

/bin/echo "StrictHostKeyChecking no" >> /etc/ssh/ssh\_config

sed -i s/requiretty/\!requiretty/ /etc/sudoers

hostnamectl set-hostname control

/bin/echo 10.0.1.49 node1>> /etc/hosts

yum install -y epel-release

yum install -y ansible sshpass

yum install -y python-boto python-boto3

yum install -y awscli

/sbin/useradd ansible

/bin/echo 'S[W=u|3E' | /bin/passwd ansible —stdin

/bin/echo "ansible ALL=(ALL) NOPASSWD: ALL" >> /etc/sudoers

sudo -u ansible /bin/mkdir -p /home/ansible/.ssh

sudo -u ansible /bin/ssh-keygen -q -N "" -f /home/ansible/.ssh/id\_rsa

sudo -u ansible sshpass -p 'S[W=u|3E' ssh-copy-id -i /home/ansible/.ssh/id\_rsa.pub ansible@localhost

sudo -u ansible sshpass -p 'S[W=u|3E' ssh-copy-id -i /home/ansible/.ssh/id\_rsa.pub ansible@node1

echo "localhost" >> /etc/ansible/hosts

echo "node1" >> /etc/ansible/hosts

ACCESS=AKIARKS4V3WBXT44X6O5

SECRET=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

mkdir /home/ansible/.aws

echo "export AWS\_ACCESS\_KEY\_ID='KEYACCESS'" > /home/ansible/accesstemplate

echo "export AWS\_SECRET\_ACCESS\_KEY='KEYSECRET'" > /home/ansible/secrettemplate

echo "#!/bin/sh" >> /home/ansible/keys.sh

sed -e "s|KEYACCESS|$ACCESS|" /home/ansible/accesstemplate >> /home/ansible/keys.sh

sed -e "s|KEYSECRET|$SECRET|" /home/ansible/secrettemplate >> /home/ansible/keys.sh

echo "export AWS\_REGION='us-east-1'" >> /home/ansible/keys.sh

chmod 700 /home/ansible/keys.sh

rm -f /home/ansible/accesstemplate /home/ansible/secrettemplate

echo "AWS\_ACCESS\_KEY\_ID: KEYACCESS" > /home/ansible/accesstemplate

echo "AWS\_SECRET\_ACCESS\_KEY: KEYSECRET" > /home/ansible/secrettemplate

sed -e "s|KEYACCESS|$ACCESS|" /home/ansible/accesstemplate >> /home/ansible/keys.yml

sed -e "s|KEYSECRET|$SECRET|" /home/ansible/secrettemplate >> /home/ansible/keys.yml

echo "AWS\_REGION: us-east-1" >> /home/ansible/keys.yml

chmod 600 /home/ansible/keys.yml

rm -f /home/ansible/accesstemplate /home/ansible/secrettemplate

echo "aws\_access\_key\_id: KEYACCESS" > /home/ansible/accesstemplate

echo "aws\_secret\_access\_key: KEYSECRET" > /home/ansible/secrettemplate

echo "[default]" >> /home/ansible/.aws/credentials

echo "[default]" >> /home/ansible/.aws/config

echo "output = json" >> /home/ansible/.aws/config

echo "region = us-east-1" >> /home/ansible/.aws/config

sed -e "s|KEYACCESS|$ACCESS|" /home/ansible/accesstemplate >> /home/ansible/.aws/credentials

sed -e "s|KEYSECRET|$SECRET|" /home/ansible/secrettemplate >> /home/ansible/.aws/credentials

chmod 600 /home/ansible/.aws/credentials

chmod 600 /home/ansible/.aws/config

rm -f /home/ansible/accesstemplate /home/ansible/secrettemplate

chown ansible.ansible /home/ansible/\*

chown ansible.ansible /home/ansible/.aws

chown ansible.ansible /home/ansible/.aws/\*

echo "finished" > /home/cloud\_user/.done

[root@control cloud\_user]# sudo cat /etc/os-release

NAME="CentOS Linux"

VERSION="7 (Core)"

ID="centos"

ID\_LIKE="rhel fedora"

VERSION\_ID="7"

PRETTY\_NAME="CentOS Linux 7 (Core)"

ANSI\_COLOR="0;31"

CPE\_NAME="cpe:/o:centos:centos:7"

HOME\_URL="<https://www.centos.org/>“

BUG\_REPORT\_URL="<https://bugs.centos.org/>“

CENTOS\_MANTISBT\_PROJECT="CentOS-7"

CENTOS\_MANTISBT\_PROJECT\_VERSION="7"

REDHAT\_SUPPORT\_PRODUCT="centos"

REDHAT\_SUPPORT\_PRODUCT\_VERSION="7"

[root@control cloud\_user]# ansible —version

ansible 2.9.27

config file = /etc/ansible/ansible.cfg

configured module search path = [u'/root/.ansible/plugins/modules', u'/usr/share/ansible/plugins/modules']

ansible python module location = /usr/lib/python2.7/site-packages/ansible

executable location = /bin/ansible

python version = 2.7.5 (default, Nov 16 2020, 22:23:17) [GCC 4.8.5 20150623 (Red Hat 4.8.5-44)]

[root@control cloud\_user]#

CFN :

 "AWSTemplateFormatVersion" : "2010-09-09",  
 "Description" : "Working with AMIs using Ansible",  
  
 "Mappings" : {  
  
   "SubnetConfig" : {  
     "VPC"     : { "CIDR" : "10.0.0.0/16" },  
     "Public1"  : { "CIDR" : "10.0.1.0/24" }  
   }  
 },  
  
 "Resources" : {  
  
   "VPC" : {  
     "Type" : "AWS::EC2::VPC",  
     "Properties" : {  
  
       "EnableDnsSupport" : "true",  
       "EnableDnsHostnames" : "true",  
       "CidrBlock" : { "Fn::FindInMap" : [ "SubnetConfig", "VPC", "CIDR" ]},  
       "Tags" : [  
        { "Key" : "Name", "Value" : "LinuxAcademy" },  
         { "Key" : "Application", "Value" : { "Ref" : "AWS::StackName" } },  
         { "Key" : "Network", "Value" : "VPC" }  
       ]  
     }  
   },  
  
  
   "PublicSubnet1" : {  
     "Type" : "AWS::EC2::Subnet",  
         "Properties" : {  
       "VpcId" : { "Ref" : "VPC" },  
       "AvailabilityZone" : {  
         "Fn::Select" : [ "0", { "Fn::GetAZs" : "" } ]  
       },  
       "CidrBlock" : { "Fn::FindInMap" : [ "SubnetConfig", "Public1", "CIDR" ]},  
       "Tags" : [  
         { "Key" : "Application", "Value" : { "Ref" : "AWS::StackName" } },  
         { "Key" : "Network", "Value" : "Public1" }  
       ]  
     }  
   },  
  
   "InternetGateway" : {  
     "Type" : "AWS::EC2::InternetGateway",  
     "Properties" : {  
       "Tags" : [  
         { "Key" : "Application", "Value" : { "Ref" : "AWS::StackName" } },  
         { "Key" : "Network", "Value" : "Public" }  
       ]  
     }  
   },  
  
   "GatewayToInternet" : {  
      "Type" : "AWS::EC2::VPCGatewayAttachment",  
      "Properties" : {  
        "VpcId" : { "Ref" : "VPC" },  
        "InternetGatewayId" : { "Ref" : "InternetGateway" }  
      }  
   },  
  
   "PublicRouteTable" : {  
     "Type" : "AWS::EC2::RouteTable",  
     "Properties" : {  
       "VpcId" : { "Ref" : "VPC" },  
       "Tags" : [  
         { "Key" : "Application", "Value" : { "Ref" : "AWS::StackName" } },  
         { "Key" : "Network", "Value" : "Public" }  
       ]  
     }  
   },  
  
   "PublicRoute" : {  
     "Type" : "AWS::EC2::Route",  
     "DependsOn" : "GatewayToInternet",  
     "Properties" : {  
       "RouteTableId" : { "Ref" : "PublicRouteTable" },  
       "DestinationCidrBlock" : "0.0.0.0/0",  
       "GatewayId" : { "Ref" : "InternetGateway" }  
     }  
   },  
  
   "PublicSubnetRouteTableAssociation1" : {  
     "Type" : "AWS::EC2::SubnetRouteTableAssociation",  
     "Properties" : {  
       "SubnetId" : { "Ref" : "PublicSubnet1" },  
       "RouteTableId" : { "Ref" : "PublicRouteTable" }  
     }  
   },  
  
   "PublicNetworkAcl" : {  
     "Type" : "AWS::EC2::NetworkAcl",  
     "Properties" : {  
       "VpcId" : { "Ref" : "VPC" },  
       "Tags" : [  
         { "Key" : "Application", "Value" : { "Ref" : "AWS::StackName" } },  
         { "Key" : "Network", "Value" : "Public" }  
       ]  
     }  
   },  
  
   "InboundHTTPPublicNetworkAclEntry" : {  
     "Type" : "AWS::EC2::NetworkAclEntry",  
     "Properties" : {  
       "NetworkAclId" : { "Ref" : "PublicNetworkAcl" },  
       "RuleNumber" : "100",  
       "Protocol" : "6",  
       "RuleAction" : "allow",  
       "Egress" : "false",  
       "CidrBlock" : "0.0.0.0/0",  
       "PortRange" : { "From" : "80", "To" : "80" }  
     }  
   },  
  
   "InboundHTTPSPublicNetworkAclEntry" : {  
     "Type" : "AWS::EC2::NetworkAclEntry",  
     "Properties" : {  
       "NetworkAclId" : { "Ref" : "PublicNetworkAcl" },  
       "RuleNumber" : "101",  
       "Protocol" : "6",  
       "RuleAction" : "allow",  
       "Egress" : "false",  
       "CidrBlock" : "0.0.0.0/0",  
       "PortRange" : { "From" : "443", "To" : "443" }  
     }  
   },  
  
   "InboundSSHPublicNetworkAclEntry" : {  
     "Type" : "AWS::EC2::NetworkAclEntry",  
     "Properties" : {  
       "NetworkAclId" : { "Ref" : "PublicNetworkAcl" },  
       "RuleNumber" : "102",  
       "Protocol" : "6",  
       "RuleAction" : "allow",  
       "Egress" : "false",  
       "CidrBlock" : "0.0.0.0/0",  
       "PortRange" : { "From" : "22", "To" : "22" }  
     }  
   },  
  
   "InboundEmphemeralPublicNetworkAclEntry" : {  
     "Type" : "AWS::EC2::NetworkAclEntry",  
     "Properties" : {  
       "NetworkAclId" : { "Ref" : "PublicNetworkAcl" },  
       "RuleNumber" : "103",  
       "Protocol" : "6",  
       "RuleAction" : "allow",  
       "Egress" : "false",  
       "CidrBlock" : "0.0.0.0/0",  
       "PortRange" : { "From" : "1024", "To" : "65535" }  
     }  
   },  
  
   "OutboundPublicNetworkAclEntry" : {  
     "Type" : "AWS::EC2::NetworkAclEntry",  
     "Properties" : {  
       "NetworkAclId" : { "Ref" : "PublicNetworkAcl" },  
       "RuleNumber" : "100",  
       "Protocol" : "6",  
       "RuleAction" : "allow",  
       "Egress" : "true",  
       "CidrBlock" : "0.0.0.0/0",  
       "PortRange" : { "From" : "0", "To" : "65535" }  
     }  
   },  
  
   "PublicSubnetNetworkAclAssociation1" : {  
     "Type" : "AWS::EC2::SubnetNetworkAclAssociation",  
     "Properties" : {  
       "SubnetId" : { "Ref" : "PublicSubnet1" },  
       "NetworkAclId" : { "Ref" : "PublicNetworkAcl" }  
     }  
   },  
  
"EC2SecurityGroup" : {  
     "Type" : "AWS::EC2::SecurityGroup",  
     "Properties" : {  
       "GroupDescription" : "Enable access to the EC2 host",  
       "VpcId" : { "Ref" : "VPC" },  
       "SecurityGroupIngress" : [  
         { "IpProtocol" : "tcp", "FromPort" : "22",  "ToPort" : "22",  "CidrIp" : "0.0.0.0/0" },  
         { "IpProtocol" : "tcp", "FromPort" : "80",  "ToPort" : "80",  "CidrIp" : "0.0.0.0/0" },  
         { "IpProtocol" : "tcp", "FromPort" : "443",  "ToPort" : "443",  "CidrIp" : "0.0.0.0/0" },  
         { "IpProtocol" : "icmp", "FromPort" : "-1",  "ToPort" : "-1",  "CidrIp" : "0.0.0.0/0" }  
       ]  
     }  
   },  
  "SGBaseIngress": {  
     "Type": "AWS::EC2::SecurityGroupIngress",  
     "Properties": {  
       "GroupId": { "Ref": "EC2SecurityGroup" },  
       "IpProtocol": "tcp",  
       "FromPort": "80",  
       "ToPort": "80",  
       "SourceSecurityGroupId": { "Ref": "EC2SecurityGroup" }  
     }  
   },  
   "ansible": {  
     "Type": "AWS::IAM::User",  
     "Properties": {  
       "Path": "/",  
       "ManagedPolicyArns" : [ "arn:aws:iam::aws:policy/AmazonEC2FullAccess" ],  
       "UserName": "ansible"  
     }  
   },  
   "ansiblekey" : {  
   "Type" : "AWS::IAM::AccessKey",  
   "Properties" : {  
      "UserName" : { "Ref" : "ansible" }  
      }  
   },  
   "Node1" : {  
      "Type" : "AWS::EC2::Instance",  
      "Properties" : {  
          "InstanceType" : "t3.micro",  
          "ImageId" : "ami-09eb574578ce81bb9",  
          "Tags" : [ {"Key" : "Name", "Value" : "Leo"}],  
          "UserData"       : { "Fn::Base64" : { "Fn::Join" : ["", [  
            "#!/bin/bash\n",  
            "/bin/echo 'S[W=u|3E' | /bin/passwd cloud\_user —stdin\n",  
            "/bin/echo \"StrictHostKeyChecking no\" >> /etc/ssh/ssh\_config\n",  
            "/usr/bin/hostnamectl set-hostname node1\n",  
            "/sbin/useradd ansible\n",  
            "/bin/echo 'S[W=u|3E' | /bin/passwd ansible —stdin\n",  
            "echo \"Image created 2019-07-07\" >> /home/ansible/image.txt\n",  
            "chown ansible.ansible /home/ansible/*\n",*  
 *"/bin/echo \"ansible        ALL=(ALL)       NOPASSWD: ALL\" >> /etc/sudoers\n"*  
 *]]}},*  
 *"NetworkInterfaces" : [{*  
 *"GroupSet"                 : [{ "Ref" : "EC2SecurityGroup" }],*  
 *"AssociatePublicIpAddress": "true",*  
 *"DeviceIndex"              : "0",*  
 *"DeleteOnTermination"      : "true",*  
 *"SubnetId"                 : { "Ref" : "PublicSubnet1" }*  
 *}]*  
 *}*  
 *},*  
 *"Control" : {*  
 *"Type" : "AWS::EC2::Instance",*  
 *"Properties" : {*  
 *"InstanceType" : "t3.micro",*  
 *"ImageId" : "ami-09eb574578ce81bb9",*  
 *"UserData"       : { "Fn::Base64" : { "Fn::Join" : ["", [*  
 *"#!/bin/bash\n",*  
*"/bin/echo 'S[W=u|3E' | /bin/passwd root —stdin\n",*  
*"/bin/echo 'S[W=u|3E' | /bin/passwd cloud\_user —stdin\n",*  
 *"/bin/echo \"StrictHostKeyChecking no\" >> /etc/ssh/ssh\_config\n",*  
 *"sed -i s/requiretty/\\!requiretty/ /etc/sudoers\n",*  
 *"hostnamectl set-hostname control\n",*  
 *"/bin/echo ",*  
 *{ "Fn::GetAtt" : [ "Node1", "PrivateIp" ] },*  
 *"\tnode1>> /etc/hosts\n",*  
  
 *"yum install -y epel-release\n",*  
 *"yum install -y ansible sshpass\n",*  
 *"yum install -y python-boto python-boto3\n",*  
 *"yum install -y awscli\n",*  
  
 *"/sbin/useradd ansible\n",*  
*"/bin/echo 'S[W=u|3E' | /bin/passwd ansible —stdin\n",*  
 *"/bin/echo \"ansible        ALL=(ALL)       NOPASSWD: ALL\" >> /etc/sudoers\n",*  
 *"sudo -u ansible /bin/mkdir -p /home/ansible/.ssh\n",*  
 *"sudo -u ansible /bin/ssh-keygen -q -N \"\" -f /home/ansible/.ssh/id\_rsa\n",*  
 *"sudo -u ansible sshpass -p 'S[W=u|3E' ssh-copy-id -i /home/ansible/.ssh/id\_rsa.pub ansible@localhost\n",*  
 *"sudo -u ansible sshpass -p 'S[W=u|3E' ssh-copy-id -i /home/ansible/.ssh/id\_rsa.pub ansible@node1\n",*  
 *"echo \"localhost\" >> /etc/ansible/hosts\n",*  
 *"echo \"node1\" >> /etc/ansible/hosts\n",*  
  
 *"ACCESS=",*  
 *{"Ref" : "ansiblekey" },*  
 *"\n",*  
 *"SECRET=",*  
 *{"Fn::GetAtt" : [*  
 *"ansiblekey",*  
 *"SecretAccessKey"*  
 *]*  
 *},*  
 *"\n",*  
  
 *"mkdir /home/ansible/.aws\n",*  
  
 *"echo \"export AWS\_ACCESS\_KEY\_ID='KEYACCESS'\" > /home/ansible/accesstemplate\n",*  
 *"echo \"export AWS\_SECRET\_ACCESS\_KEY='KEYSECRET'\" > /home/ansible/secrettemplate\n",*  
 *"echo \"#!/bin/sh\"  >> /home/ansible/keys.sh\n",*  
 *"sed -e \"s|KEYACCESS|$ACCESS|\" /home/ansible/accesstemplate  >> /home/ansible/keys.sh\n",*  
 *"sed -e \"s|KEYSECRET|$SECRET|\" /home/ansible/secrettemplate  >> /home/ansible/keys.sh\n",*  
 *"echo \"export AWS\_REGION='us-east-1'\"  >> /home/ansible/keys.sh\n",*  
 *"chmod 700 /home/ansible/keys.sh\n",*  
 *"rm -f /home/ansible/accesstemplate /home/ansible/secrettemplate\n",*  
  
 *"echo \"AWS\_ACCESS\_KEY\_ID: KEYACCESS\" > /home/ansible/accesstemplate\n",*  
 *"echo \"AWS\_SECRET\_ACCESS\_KEY: KEYSECRET\" > /home/ansible/secrettemplate\n",*  
 *"sed -e \"s|KEYACCESS|$ACCESS|\" /home/ansible/accesstemplate  >> /home/ansible/keys.yml\n",*  
 *"sed -e \"s|KEYSECRET|$SECRET|\" /home/ansible/secrettemplate  >> /home/ansible/keys.yml\n",*  
 *"echo \"AWS\_REGION: us-east-1\"  >> /home/ansible/keys.yml\n",*  
 *"chmod 600 /home/ansible/keys.yml\n",*  
 *"rm -f /home/ansible/accesstemplate /home/ansible/secrettemplate\n",*  
  
 *"echo \"aws\_access\_key\_id: KEYACCESS\" > /home/ansible/accesstemplate\n",*  
 *"echo \"aws\_secret\_access\_key: KEYSECRET\" > /home/ansible/secrettemplate\n",*  
 *"echo \"[default]\" >> /home/ansible/.aws/credentials\n",*  
 *"echo \"[default]\" >> /home/ansible/.aws/config\n",*  
 *"echo \"output = json\" >> /home/ansible/.aws/config\n",*  
 *"echo \"region = us-east-1\" >> /home/ansible/.aws/config\n",*  
 *"sed -e \"s|KEYACCESS|$ACCESS|\" /home/ansible/accesstemplate  >> /home/ansible/.aws/credentials\n",*  
 *"sed -e \"s|KEYSECRET|$SECRET|\" /home/ansible/secrettemplate  >> /home/ansible/.aws/credentials\n",*  
 *"chmod 600 /home/ansible/.aws/credentials\n",*  
 *"chmod 600 /home/ansible/.aws/config\n",*  
 *"rm -f /home/ansible/accesstemplate /home/ansible/secrettemplate\n",*  
  
 *"chown ansible.ansible /home/ansible/*\n",  
               "chown ansible.ansible /home/ansible/.aws\n",  
               "chown ansible.ansible /home/ansible/.aws/\*\n",  
               "echo \"finished\" > /home/cloud\_user/.done\n"  
           ]]}},  
           "NetworkInterfaces" : [{  
               "GroupSet"                 : [{ "Ref" : "EC2SecurityGroup" }],  
               "AssociatePublicIpAddress": "true",  
               "DeviceIndex"              : "0",  
               "DeleteOnTermination"      : "true",  
               "SubnetId"                 : { "Ref" : "PublicSubnet1" }  
           }]  
       }  
   }  
  },  
 "Outputs" : {  
    "pubIpAddress1" : {  
       "Description" : "Public IP of Ansible Control Node",  
       "Value" : { "Fn::GetAtt" : [ "Control", "PublicIp" ] }  
     },  
     "privIpAddress3" : {  
       "Description" : "Private IP of Ansible Control Node",  
       "Value" : { "Fn::GetAtt" : [ "Control", "PrivateIp" ] }  
     }  
 }  
}