https://www.linkedin.com/in/neel-darji-3a5a8661/

AWS IAM Project

(AWS+ Ansible) Automated user migration and management of AWS Identity and Access Management (IAM) resources using Ansible

Project executed by: Neel Darji

- **Project Definition:** A software company A got merged into another company B and as AWS solution engineer, I was asked to migrate all users from company A to company B.

  There was 100+ users that needed to be migrated and in addition to migrating these users, it will be necessary to associate permissions to groups and users must have the MFA (Multi-factor authentication) enabled, adding an additional layer for the security.

- **Solution: Ansible**

  - **What is Ansible?**
    - Ansible is a software tool that provides simple but powerful automation for cross-platform computer support. It is primarily intended for IT professionals, who use it for application deployment, updates on workstations and servers, cloud provisioning, configuration management, intra-service orchestration, and nearly anything a systems administrator does on a weekly or daily basis. Ansible doesn't depend on agent software and has no additional security infrastructure, so it's easy to deploy.

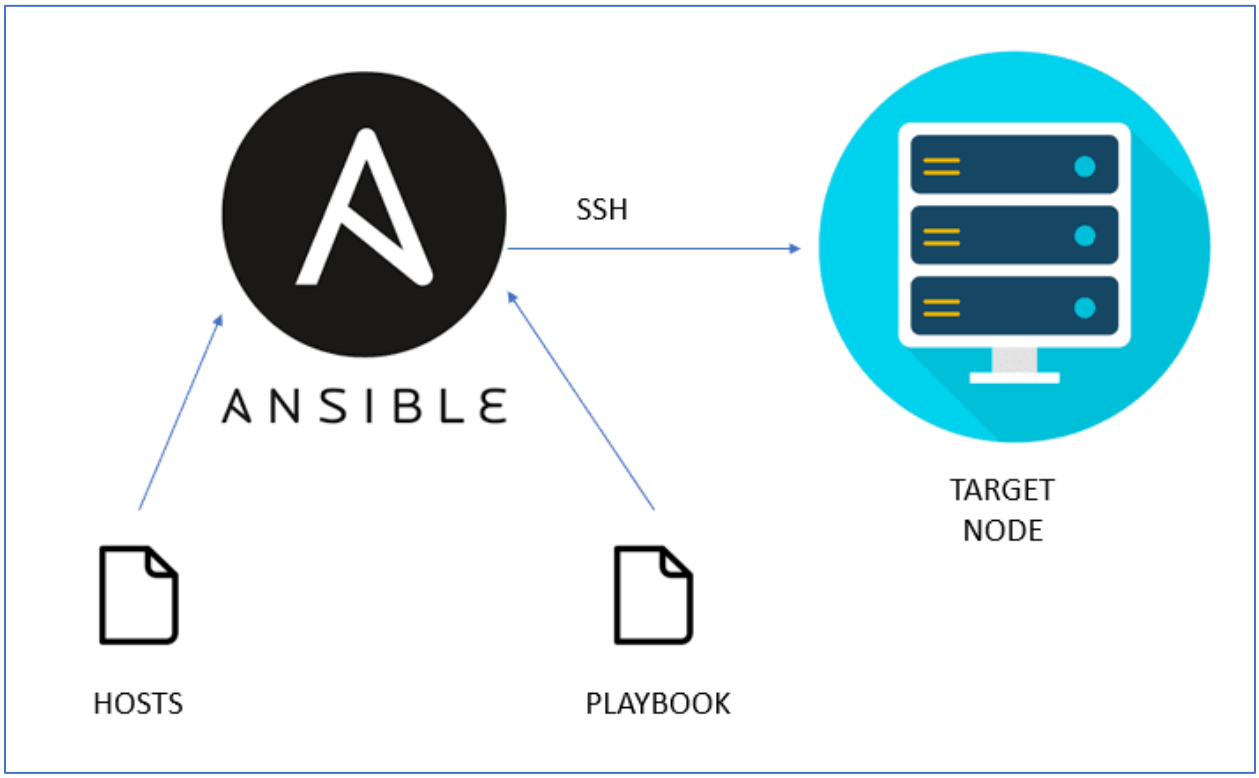
    - Because Ansible is all about automation, it requires instructions to accomplish each job. With everything written down in simple script form, it's easy to do version control. The practical result of this is a major contribution to the "infrastructure as code" movement in IT: the idea that the maintenance of server and client infrastructure can and should be treated the same as software development, with repositories of self-documenting, proven, and executable solutions capable of running an organization regardless of staff changes.

    - While Ansible may be at the forefront of automation, systems administration, and DevOps, it's also useful to everyday users. Ansible allows you to configure not just one computer, but potentially a whole network of computers at once, and using it requires no programming skills. Instructions written for Ansible are human-readable. Whether you're entirely new to computers or an expert, Ansible files are easy to understand.

  - **How Ansible works?**
    - In Ansible, there are two categories of computers: **the control node and managed nodes**. The control node is a computer that runs Ansible. There must be at least one control node, although a backup control node may also exist. A managed node is any device being managed by the control node.

    - Ansible works by connecting to nodes (clients, servers, or whatever you're configuring) on a network, and then sending a small program called an

Ansible module to that node. Ansible executes these modules over SSH and removes them when finished. The only requirement for this interaction is that your Ansible control node has login access to the managed nodes. SSH keys are the most common way to provide access, but other forms of authentication are also supported.



Project Implementation:

Phase-1: Connect Ansible node to AWS environment

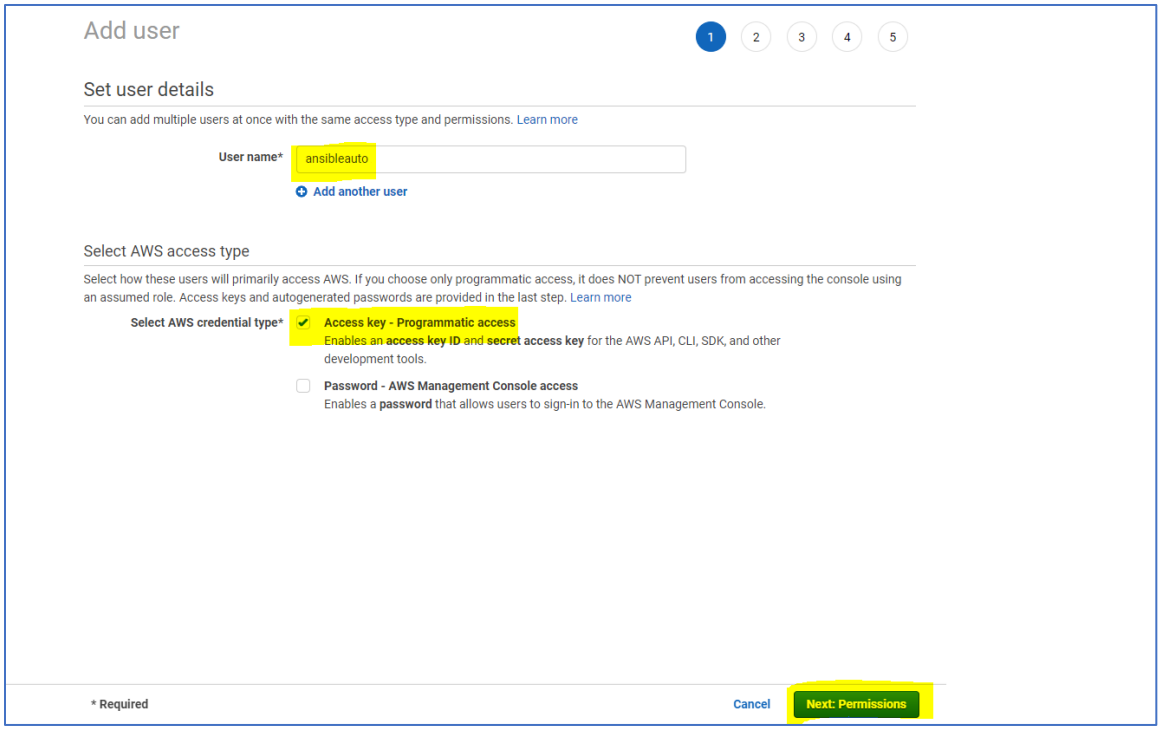
We need to deploy users and groups in AWS, so how can Ansible communicate with AWS?

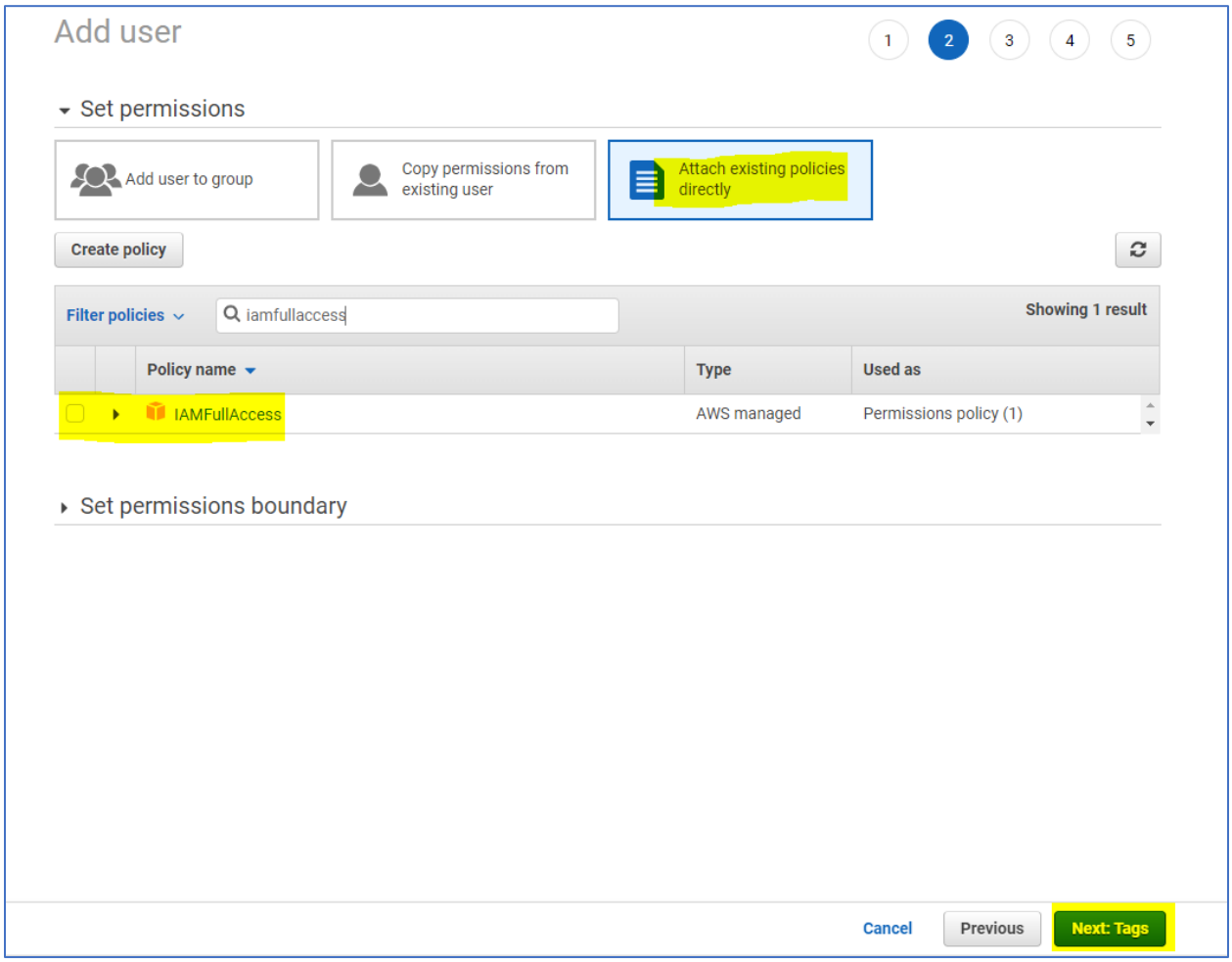
- **Creation of AWS user** - We need to configure our Ansible control node for the job and provide appropriate AWS credentials. For this, we will configure a new IAM user in the AWS console that will allow Ansible to connect to the console programmatically.
- **Assigning IAM permission** - We will also assign the Ansible user the role IAMFullAccess to allow IAM tasks.

Please follow below steps to achieve above tasks:

- Log into the AWS console using your credentials.
- Search for IAM in the Find Services search box and select the IAM that shows up in the popup box.
- Select Users in the left menu.
- Click Add User at the top of the page.
- Provide the username “ansibleauto” and check the box next to Programmatic access for access type.
- Click Next: Permissions.
- Select Attach existing policies directly and search for IAMFullAccess using the filter policies search box.
- Check the box next to IAMFullAccess.
- Click Next: Tags, then Next: Review, and lastly, after ensuring your configurations are correct, click Create user.
- Click Show under Secret access key to reveal the secret access key for the “ansibleauto” user or download the csv file.

  Note: Download this CSV file and keep it in safe place as we will need this after.


Add user
1 2 3 4 5
Set user details
You can add multiple users at once with the same access type and permissions. Learn more
User name*
ansibleauto
Add another user
Select AWS access type
Select how these users will primarily access AWS. If you choose only programmatic access, it does NOT prevent users from accessing the console using an assumed role. Access keys and autogenerated passwords are provided in the last step. Learn more
Select AWS credential type*
Access key - Programmatic access
Enables an access key ID and secret access key for the AWS API, CLI, SDK, and other development tools.
Password - AWS Management Console access
Enables a password that allows users to sign-in to the AWS Management Console.
* Required
Cancel
Next: Permissions


Add user
1 2 3 4 5
Set permissions
Add user to group
Copy permissions from existing user
Attach existing policies directly
Create policy
Filter policies
iamfullaccess
Showing 1 result
Policy name
Type
Used as
IAMFullAccess
AWS managed
Permissions policy (1)
Set permissions boundary
Cancel
Previous
Next: Tags

Add user

1 2 3 4 5

Review

Review your choices. After you create the user, you can view and download the autogenerated password and access key.

User details

User name	ansibleauto
AWS access type	Programmatic access - with an access key
Permissions boundary	Permissions boundary is not set

Permissions summary

The following policies will be attached to the user shown above.

Type	Name
Managed policy	IAMFullAccess

Tags

No tags were added.

Cancel Previous Create user

Add user

1 2 3 4 5

Success

You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.

Users with AWS Management Console access can sign-in at: https://859215391636.signin.aws.amazon.com/console

Download .csv

User	Access key ID	Secret access key
ansibleauto	AKIA4QDJ2NOKC7OOL6FZ	********* Show

Close

Phase-2: Prepare Ansible control node

- Create EC2 VM instance / you can use Microsoft Play store to download Ubuntu and use that Ubuntu as Ansible Control node.
- Once OS is ready, install following packages:
  - Python
  - Boto
  - boto3
  - awscli
  - ansible

Below are the commands to execute to install above packages:

```
$ sudo apt update
```

```
$ sudo apt install python3-pip
```

```
$ pip3 install boto boto3
```

```
$ pip3 install awscli
```

```
$ pip3 install ansible
```

Now, if you remember, we need to encrypt credentials for the AWS user we created in Phase 1.

Credential Encryption - These user's credentials need to be protected, so they will be supplied by means of an encrypted Ansible vault.

Please follow below steps for this:

- Log into the Ansible control node with your username. In my case it is "neel".
- Open /home/neel/keys.yaml using a text editor such as Vim and replace each place holder with the appropriate value.
- Save the file and quit.

neel@Q91BBRV2: ~

```
— — # AWS Credential information
AWS_ACCESS_KEY_ID: AKIA4QDJ2NOKGD4WQ6OL
AWS_SECRET_ACCESS_KEY:  qxhwyklHGRHeO8dNG5tSy7rqaor1M+vzcNEpWze5
AWS_REGION: EAST US
$

~
~
```

- Now, run “ansible-vault encrypt” command to encrypt this key.yaml file.
- Enter the password so you can decrypt it later in case you need it.

```
neel@Q91BBRV2:~$ ansible-vault encrypt keys.yaml
New Vault password:
Confirm New Vault password:
```

Phase-3: Create Playbooks and configure csv having user + group + password details

Excellent!! Now, we have infrastructure ready to connect Ansible node to AWS infra.

Create playbooks – What is this playbook???
Playbook is where we place the tasks. You can divide multiple tasks inside a role or create it in a single task file.

For simplicity, I have created 2 playbooks:

1. Groups – iam.yaml
2. Users – user_create.yaml

Let’s create 1st playbook for creating groups in AWS:

```
--- # create group in AWS using Ansible
- hosts: localhost
  vars_files:
    - /home/neel/keys.yaml
  tasks:
  - name: Create a group and attach a managed policy using its ARN
    community.aws.iam_group:
        aws_access_key: '{{ AWS_ACCESS_KEY_ID }}'
        aws_secret_key: '{{ AWS_SECRET_ACCESS_KEY }}'
        region: '{{ AWS_REGION }}'
        name: DBA01
        managed_policies:
         - arn:aws:iam::859215391636:policy/ForceMFAAllUsers
```

```
      - arn:aws:iam::aws:policy/AmazonRDSFullAccess
      - arn:aws:iam::aws:policy/IAMUserChangePassword
     state: present

- name: Create a group and attach a managed policy using its ARN
  community.aws.iam_group:
    aws_access_key: '{{ AWS_ACCESS_KEY_ID }}'
    aws_secret_key: '{{ AWS_SECRET_ACCESS_KEY }}'
    region: '{{ AWS_REGION }}'
    name: CloudAdmin01
    managed_policies:
      - arn:aws:iam::859215391636:policy/ForceMFAAllUsers
      - arn:aws:iam::awe:policy/IAMUserChangePassword
      - arn:aws:iam::aws:policy/AdministratorAccess
    state: present

- name: Create a group and attach a managed policy using its ARN
  community.aws.iam_group:
    aws_access_key: '{{ AWS_ACCESS_KEY_ID }}'
    aws_secret_key: '{{ AWS_SECRET_ACCESS_KEY }}'
    region: '{{ AWS_REGION }}'
    name: LinuxAdmin01
    managed_policies:
      - arn:aws:iam::859215391636:policy/ForceMFAAllUsers
      - arn:aws:iam::aws:policy/AmazonEC2FullAccess
      - arn:aws:iam::aws:policy/IAMUserChangePassword
    state: present
```

```
- name: Create a group and attach a managed policy using its ARN
  community.aws.iam_group:
    aws_access_key: '{{ AWS_ACCESS_KEY_ID }}'
    aws_secret_key: '{{ AWS_SECRET_ACCESS_KEY }}'
    region: '{{ AWS_REGION }}'
    name: Trainees01
    managed_policies:
     - arn:aws:iam::arn:aws:iam::859215391636:policy/ForceMFAAllUsers
     - arn:aws:iam::aws:policy/ReadOnlyAccess
     - arn:aws:iam::aws:policy/IAMUserChangePassword
    state: present

- name: Create a group and attach a managed policy using its ARN
  community.aws.iam_group:
    aws_access_key: '{{ AWS_ACCESS_KEY_ID }}'
    aws_secret_key: '{{ AWS_SECRET_ACCESS_KEY }}'
    region: '{{ AWS_REGION }}'
    name: NetworkAdmin01
    managed_policies:
      - arn:aws:iam::859215391636:policy/ForceMFAAllUsers
      - arn:aws:iam::aws:policy/AmazonVPCFullAccess
      - arn:aws:iam::aws:policy/IAMUserChangePassword
    state: present
```

```yaml
---  # create group in AWS using Ansible
- hosts: localhost
  vars_files:
    - /home/neel/keys.yaml
  tasks:
  - name: Create a group and attach a managed policy using its ARN
    community.aws.iam_group:
        aws_access_key: '{{ AWS_ACCESS_KEY_ID }}'
        aws_secret_key: '{{ AWS_SECRET_ACCESS_KEY }}'
        region: '{{ AWS_REGION }}'
        name: DBA01
        managed_policies:
         - arn:aws:iam::859215391636:policy/ForceMFAAllUsers
         - arn:aws:iam::aws:policy/AmazonRDSFullAccess
         - arn:aws:iam::aws:policy/IAMUserChangePassword
        state: present

- name: Create a group and attach a managed policy using its ARN
    community.aws.iam_group:
       aws_access_key: '{{ AWS_ACCESS_KEY_ID }}'
       aws_secret_key: '{{ AWS_SECRET_ACCESS_KEY }}'
       region: '{{ AWS_REGION }}'
       name: CloudAdmin01
       managed_policies:
         - arn:aws:iam::859215391636:policy/ForceMFAAllUsers
         - arn:aws:iam::awe:policy/IAMUserChangePassword
         - arn:aws:iam::aws:policy/AdministratorAccess
       state: present

- name: Create a group and attach a managed policy using its ARN
    community.aws.iam_group:
       aws_access_key: '{{ AWS_ACCESS_KEY_ID }}'
       aws_secret_key: '{{ AWS_SECRET_ACCESS_KEY }}'
       region: '{{ AWS_REGION }}'
       name: LinuxAdmin01
       managed_policies:
         - arn:aws:iam::859215391636:policy/ForceMFAAllUsers
         - arn:aws:iam::aws:policy/AmazonEC2FullAccess
         - arn:aws:iam::aws:policy/IAMUserChangePassword
       state: present
```

Now, create CSV file having users, respective groups, and password:

neel@Q91BBRV2: ~

```
Username,Groupname,Password
test1,DBA01,Changeme123456!
test2,LinuxAdmin01,Changeme123456!
test3,DBA01,Changeme123456!
```

Now, let's create 2nd Playbook for creating users:

```
--- # create user
- hosts: localhost
  vars_files:
        - /home/neel/keys.yaml
  tasks:
  - name: reading the csv file
    read_csv:
        path: username.csv
        register: user_list
        delegate_to: localhost
  - name: display user_list data
      debug:
          var: user_list.list

  - name: Create two new IAM users with API keys
    community.aws.iam:
        aws_access_key: '{{ AWS_ACCESS_KEY_ID }}'
        aws_secret_key: '{{ AWS_SECRET_ACCESS_KEY }}'
```

region: '{{ AWS_REGION }}'

iam_type: user

name: "{{ item.Username }}"

groups: "{{ item.Groupname }}"

password: "{{ item.Password }}"

state: present

loop: "{{ user_list.list }}"

neel@Q91BBRV2: ~

```
---  # create user
- hosts: localhost
  vars_files:
     - /home/neel/keys.yaml

  tasks:

- name: reading the csv file
    read_csv:
      path: username.csv
    register: user_list
    delegate_to: localhost

- name: display user_list data
    debug:
      var: user_list.list

- name: Create two new IAM users with API keys
    community.aws.iam:
        aws_access_key: '{{ AWS_ACCESS_KEY_ID }}'
        aws_secret_key: '{{ AWS_SECRET_ACCESS_KEY }}'
        region: '{{ AWS_REGION }}'
        iam_type: user
        name: "{{ item.Username }}"
        groups: "{{ item.Groupname }}"
        password: "{{ item.Password }}"
        state: present
    loop: "{{ user_list.list }}"
~
```

Phase-4: Run the playbooks

Execute following command:

1. Create groups with Playbook-1 – “iam.yaml”

ansible-playbook --ask-vault-pass iam.yaml

```
root@Q91BBRV2:/home/neel# ansible-playbook --ask-vault-pass iam.yaml
Vault password:
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'

PLAY [localhost] ***********************************************************************************************************************************************************************

TASK [Gathering Facts] *****************************************************************************************************************************************************************
ok: [localhost]

TASK [Create a group and attach a managed policy using its ARN] ************************************************************************************************************************
ok: [localhost]

TASK [Create a group and attach a managed policy using its ARN] ************************************************************************************************************************
ok: [localhost]

TASK [Create a group and attach a managed policy using its ARN] ************************************************************************************************************************
ok: [localhost]

TASK [Create a group and attach a managed policy using its ARN] ************************************************************************************************************************
ok: [localhost]

TASK [Create a group and attach a managed policy using its ARN] ************************************************************************************************************************
ok: [localhost]

PLAY RECAP *****************************************************************************************************************************************************************************
localhost                  : ok=6    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

2. Create users with Playbook-2 – “create_user.yaml”

```
root@Q91BBRV2:/home/neel# ansible-playbook --ask-vault-pass user_create.yaml
Vault password:
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'
[DEPRECATION WARNING]: community.aws.iam has been deprecated. The iam module is based upon a deprecated version of the AWS SDKs and is deprecated in favor of the iam_user, iam_group and iam_role
modules. Please update your tasks. This feature will be removed from community.aws in version 3.0.0. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.

PLAY [localhost] ***********************************************************************************************************************************************************************

TASK [Gathering Facts] *****************************************************************************************************************************************************************
ok: [localhost]

TASK [reading the csv file] ************************************************************************************************************************************************************
ok: [localhost]

TASK [display user_list data] **********************************************************************************************************************************************************
ok: [localhost] => {
    "user_list.list": [
        {
            "Groupname": "DBA01",
            "Password": "Changeme123456!",
            "Username": "test1"
        },
        {
            "Groupname": "LinuxAdmin01",
            "Password": "Changeme123456!",
            "Username": "test2"
        },
        {
            "Groupname": "DBA01",
            "Password": "Changeme123456!",
            "Username": "test3"
        }
    ]
}

TASK [Create two new IAM users with API keys] ******************************************************************************************************************************************
changed: [localhost] => (item={'Username': 'test1', 'Groupname': 'DBA01', 'Password': 'Changeme123456!'})
changed: [localhost] => (item={'Username': 'test2', 'Groupname': 'LinuxAdmin01', 'Password': 'Changeme123456!'})
changed: [localhost] => (item={'Username': 'test3', 'Groupname': 'DBA01', 'Password': 'Changeme123456!'})
[DEPRECATION WARNING]: The 'iam' module has been deprecated and replaced by the 'iam_user', 'iam_group' and 'iam_role' modules'. This feature will be removed from community.aws in version 3.0.0.
Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.

PLAY RECAP *****************************************************************************************************************************************************************************
localhost                  : ok=4    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Identity and Access Management (IAM)

Search IAM

Dashboard

Access management
User groups
Users
Roles
Policies
Identity providers
Account settings

Access reports
Access analyzer
Archive rules
Analyzers
Settings
Credential report
Organization activity
Service control policies (SCPs)

IAM > Users

Users (9) Info

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

Delete | Add users

Find users by username or access key

User name	Groups	Last activity	MFA	Password age	Active key age
ansibleauto	None	13 minutes ago	None	None	Yesterday
barbara.brown	CloudAdmin	4 days ago	None	4 days ago	-
brian.garcia	LinuxAdmin	4 days ago	None	4 days ago	-
jane.doe	DBA	4 days ago	Virtual	4 days ago	-
michael.scott	NetworkAdmin	Never	None	4 days ago	-
neel.darji	Trainees	Never	None	4 days ago	-
test1	DBA01	Never	None	13 minutes ago	-
test2	LinuxAdmin01	Never	None	13 minutes ago	-
test3	DBA01	Never	None	13 minutes ago	-

Identity and Access Management (IAM)

Search IAM

Dashboard

Access management
User groups
Users
Roles
Policies
Identity providers
Account settings

Access reports
Access analyzer
Archive rules
Analyzers
Settings
Credential report
Organization activity
Service control policies (SCPs)

IAM > User groups

User groups (10) Info

A user group is a collection of IAM users. Use groups to specify permissions for a collection of users.

Delete | Create group

Filter User groups by property or group name and press enter

Group name	Users	Permissions	Creation time
CloudAdmin	Loading	Loading	5 days ago
CloudAdmin01	Loading	Loading	21 hours ago
DBA	Loading	Loading	5 days ago
DBA01	Loading	Loading	21 hours ago
LinuxAdmin	Loading	Loading	5 days ago
LinuxAdmin01	Loading	Loading	21 hours ago
NetworkAdmin	Loading	Loading	5 days ago
NetworkAdmin01	Loading	Loading	21 hours ago
Trainees	Loading	Loading	5 days ago
Trainees01	Loading	Loading	21 hours ago

Identity and Access Management (IAM)

Dashboard

Access management
User groups
Users
Roles
Policies
Identity providers
Account settings

Access reports
Access analyzer
Archive rules
Analyzers
Settings
Credential report
Organization activity
Service control policies (SCPs)

User ARN: arn:aws:iam::859215391636:user/test1

Path: /

Creation time: 2022-06-14 15:01 ADT

Permissions | Groups (1) | Tags | Security credentials | Access Advisor

Permissions policies (3 policies applied)

Add permissions | Add inline policy

Policy name	Policy type
Attached from group	
ForceMFAAllUsers	Managed policy from group DBA01
AmazonRDSFullAccess	AWS managed policy from group DBA01
IAMUserChangePassword	AWS managed policy from group DBA01

Permissions boundary (not set)

Generate policy based on CloudTrail events

You can generate a new policy based on the access activity for this user, then customize, create, and attach it to this role. AWS uses your CloudTrail events to identify the services and actions used and generate a policy. Learn more