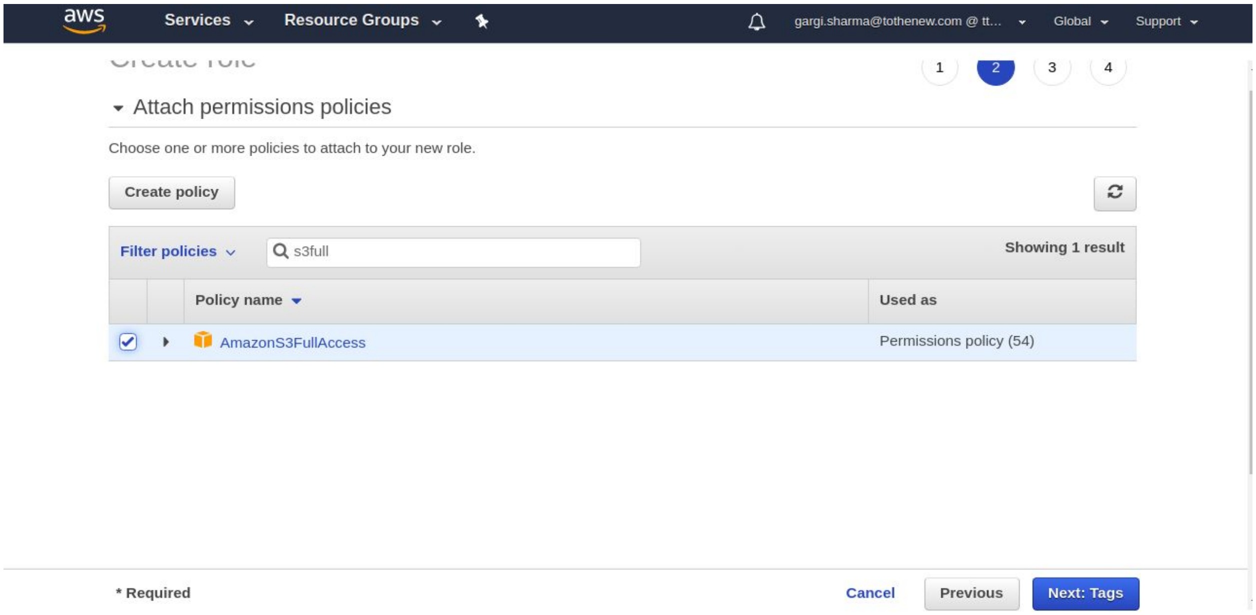
Assessment -10

IAM

Trainee Name : Arun Parmar
Mentor Name : Ravi Kumar
College: UPES

1. Create a role with full access to S3



Attach permissions policies

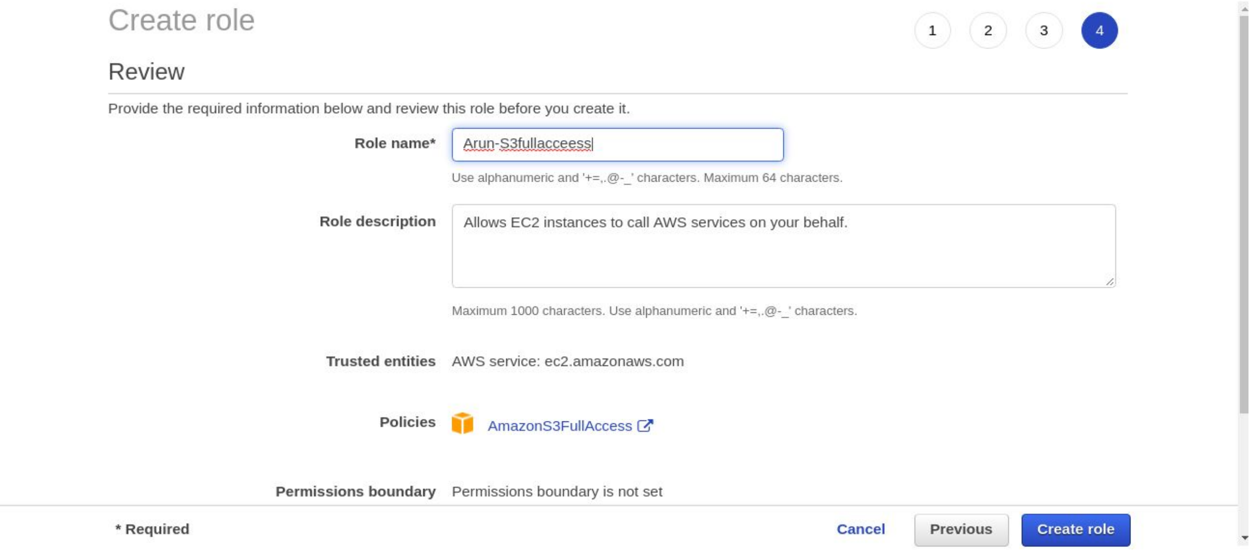
Choose one or more policies to attach to your new role.

s3full

Showing 1 result

	Policy name	Used as
☑	AmazonS3FullAccess	Permissions policy (54)

* Required



Create role

Review

Provide the required information below and review this role before you create it.

Role name* Arun-S3fullacceess

Use alphanumeric and '+=,.@-_' characters. Maximum 64 characters.

Role description Allows EC2 instances to call AWS services on your behalf.

Maximum 1000 characters. Use alphanumeric and '+=,.@-_' characters.

Trusted entities AWS service: ec2.amazonaws.com

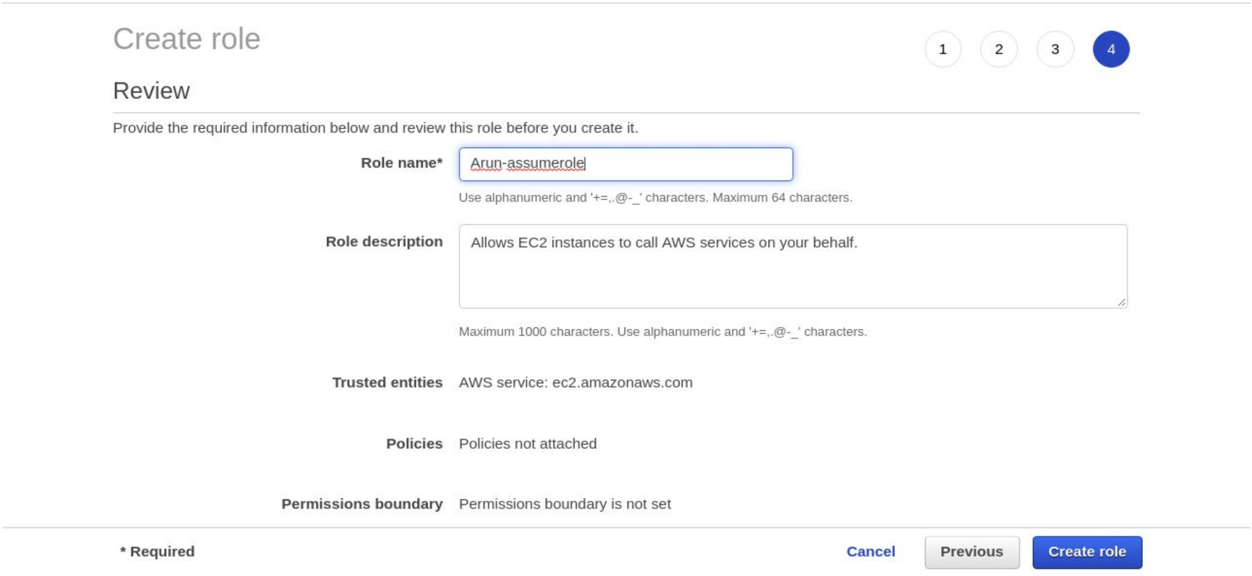
Policies AmazonS3FullAccess

Permissions boundary Permissions boundary is not set

* Required

2. **Create another which has the policy to assume the previous Role.**

Create a new role



Create role

Review

Provide the required information below and review this role before you create it.

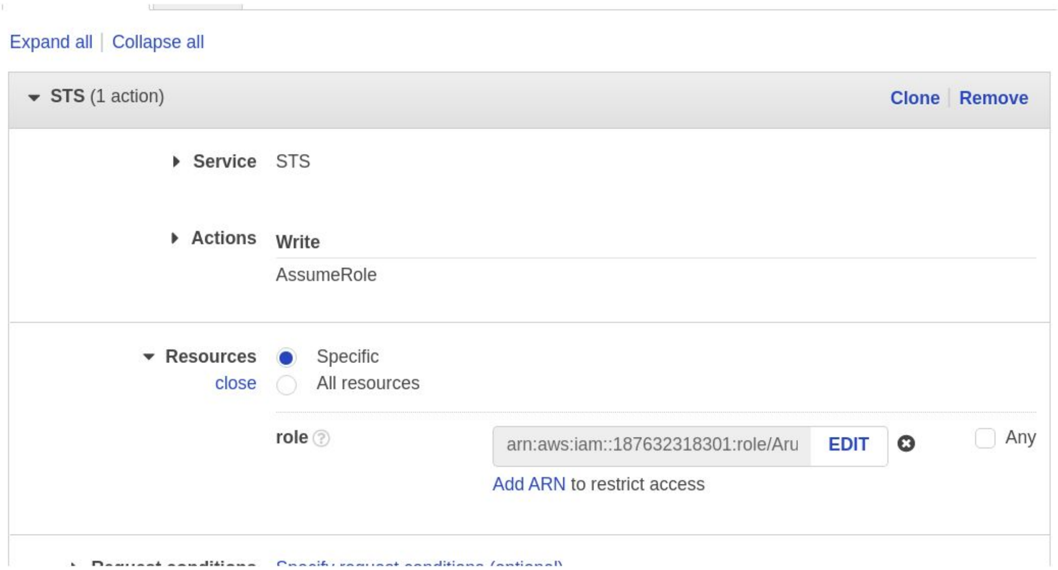
Role name*	Arun-assumerole
	Use alphanumeric and '+=,.@-_' characters. Maximum 64 characters.
Role description	Allows EC2 instances to call AWS services on your behalf.
	Maximum 1000 characters. Use alphanumeric and '+=,.@-_' characters.
Trusted entities	AWS service: ec2.amazonaws.com
Policies	Policies not attached
Permissions boundary	Permissions boundary is not set

* Required

Create a new policy

Select service STS and action assume role

Go to resources(specific) and Copy the ARN of s3 full access and paste



STS (1 action)

Service STS

Actions Write
AssumeRole

Resources

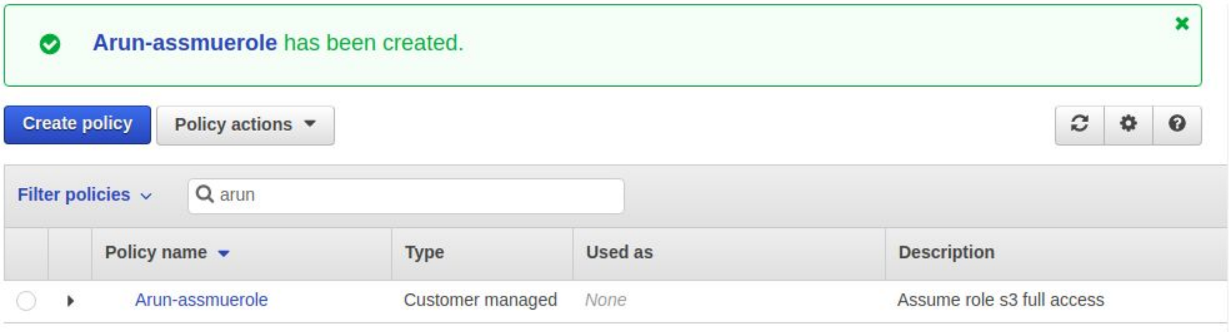
● Specific
○ All resources

role arn:aws:iam::187632318301:role/Aru

□ Any

Add ARN to restrict access

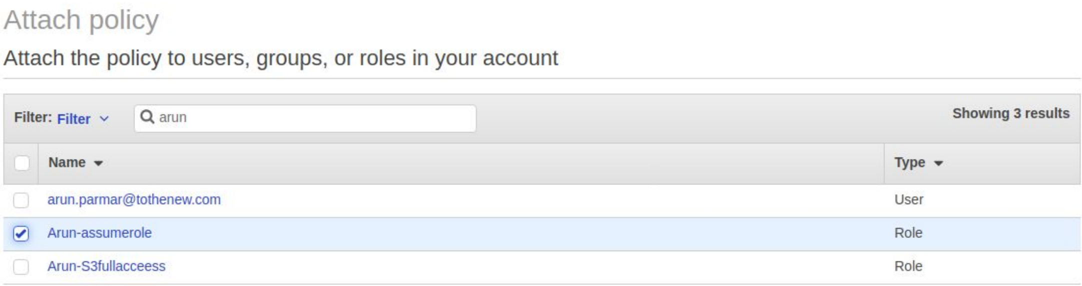
Request conditions Specify request conditions (optional)



Arun-assmuerole has been created.

	Policy name	Type	Used as	Description
○	Arun-assmuerole	Customer managed	None	Assume role s3 full access

Attach this policy to the newly created role



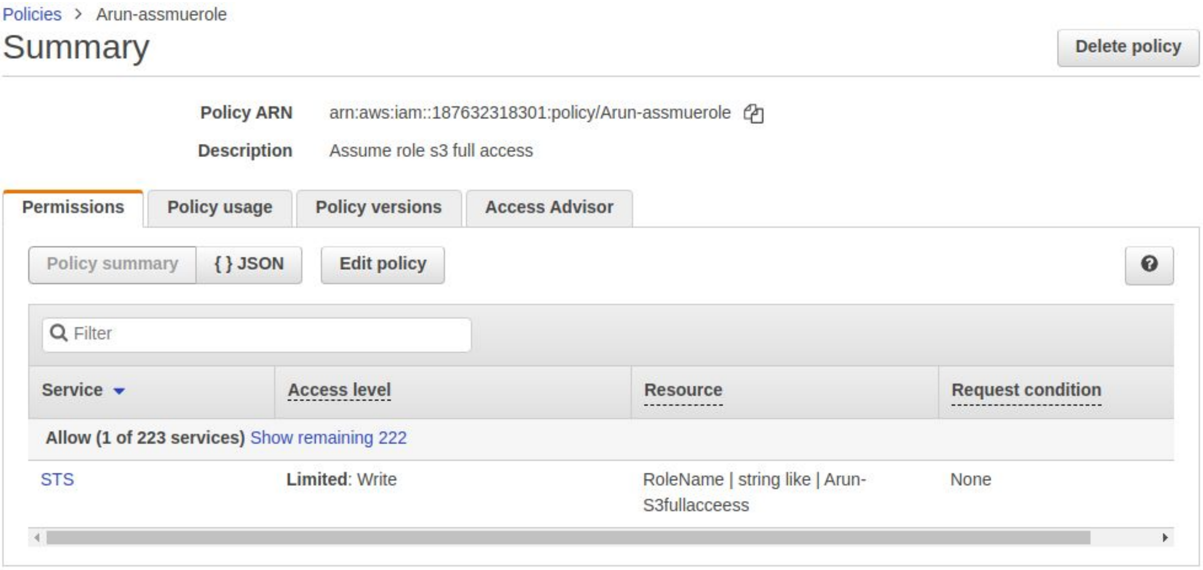
Attach policy

Attach the policy to users, groups, or roles in your account

Showing 3 results

□	Name	Type
□	arun.parmar@tothenew.com	User
☑	Arun-assumerole	Role
□	Arun-S3fullacceess	Role

Now open the newly created role and check for the assume role



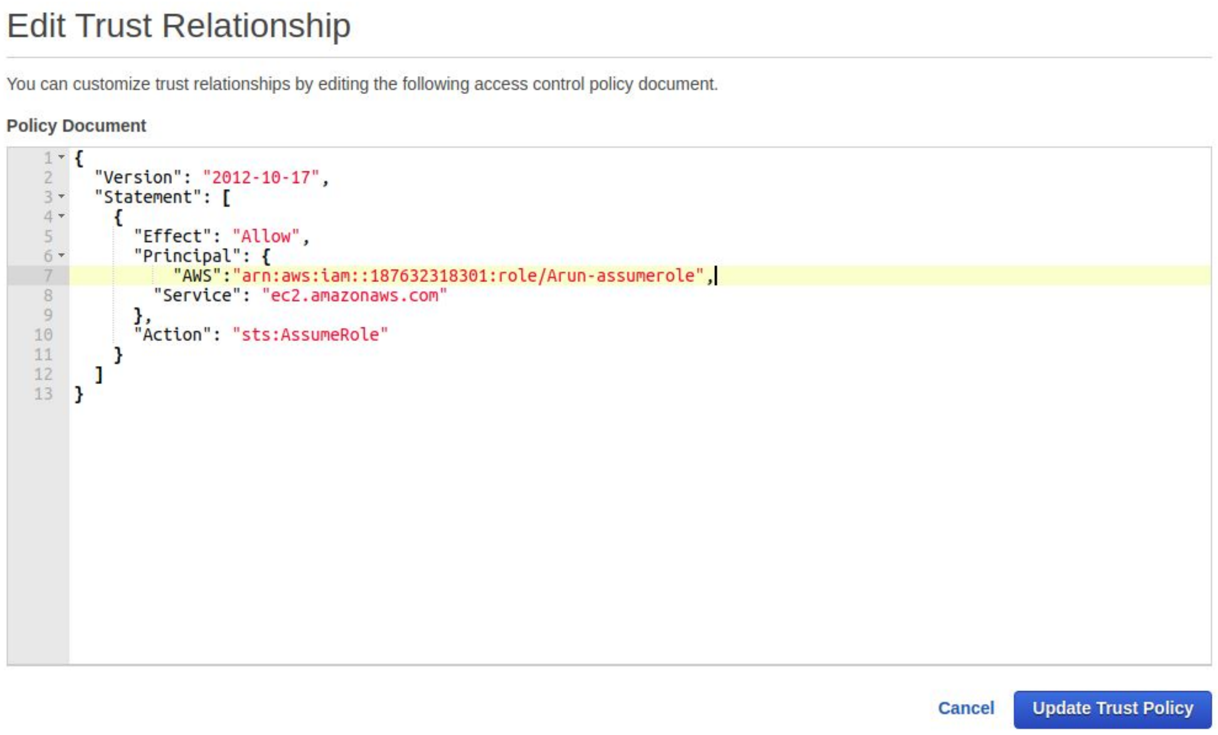
Summary

Policy ARN arn:aws:iam::187632318301:policy/Arun-assmuerole

Description Assume role s3 full access

Service	Access level	Resource	Request condition
Allow (1 of 223 services) Show remaining 222			
STS	Limited: Write	RoleName | string like | Arun-S3fullacceess	None

Go to the newly created role(assumerole-Arun) and copy the ARN. Now go to the old role(ArunS3fullaccess) and edit trust relationships. Then paste the ARN as follows.

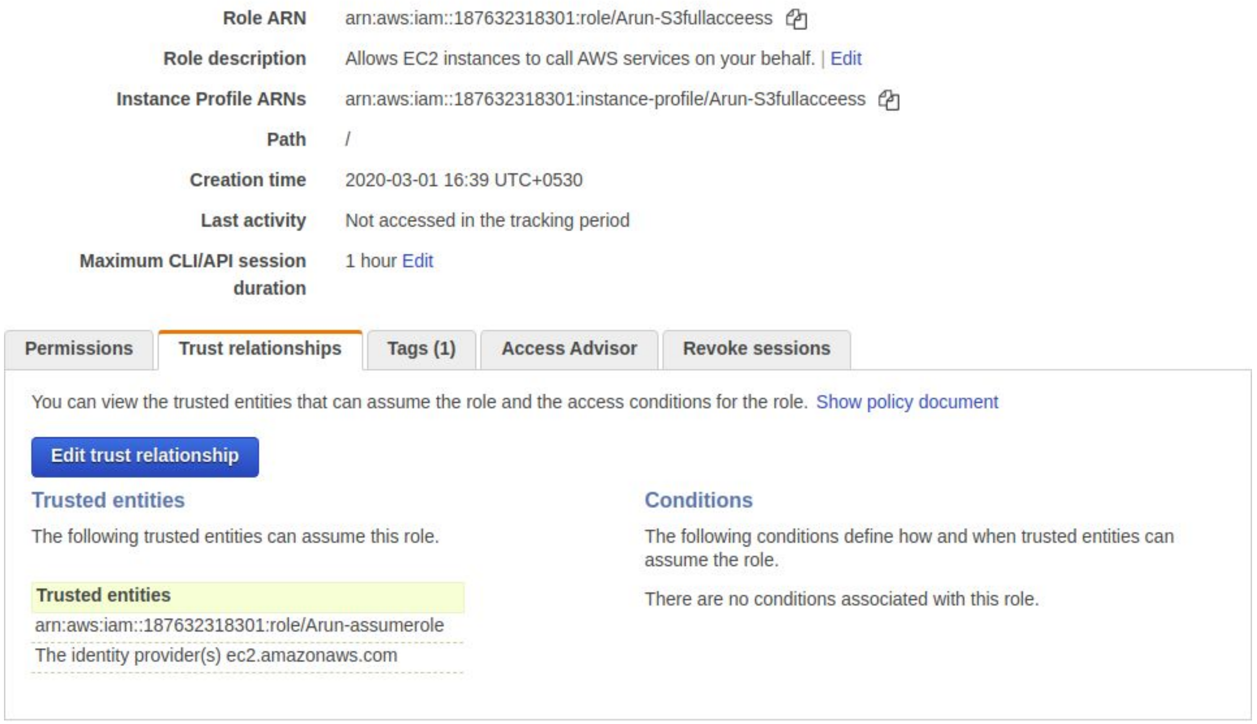


Edit Trust Relationship

You can customize trust relationships by editing the following access control policy document.

Policy Document

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
          "AWS":"arn:aws:iam::187632318301:role/Arun-assumerole",
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```



Role ARN	arn:aws:iam::187632318301:role/Arun-S3fullacceess
Role description	Allows EC2 instances to call AWS services on your behalf. | Edit
Instance Profile ARNs	arn:aws:iam::187632318301:instance-profile/Arun-S3fullacceess
Path	/
Creation time	2020-03-01 16:39 UTC+0530
Last activity	Not accessed in the tracking period
Maximum CLI/API session duration	1 hour Edit

You can view the trusted entities that can assume the role and the access conditions for the role. Show policy document

Trusted entities

The following trusted entities can assume this role.

Trusted entities
arn:aws:iam::187632318301:role/Arun-assumerole
The identity provider(s) ec2.amazonaws.com

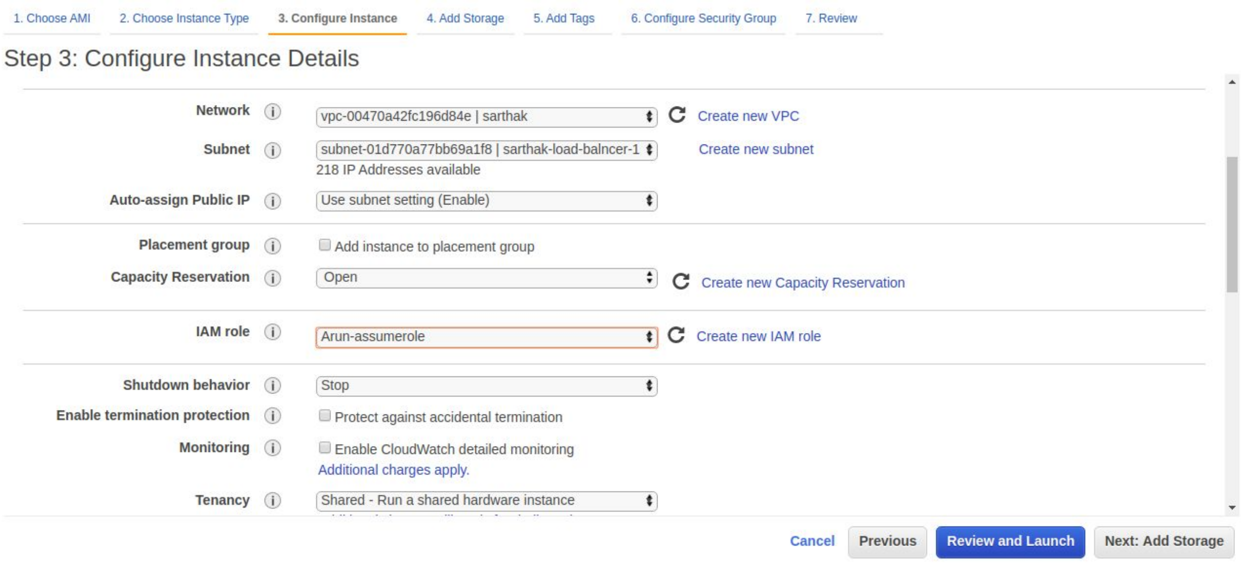
Conditions

The following conditions define how and when trusted entities can assume the role.

There are no conditions associated with this role.

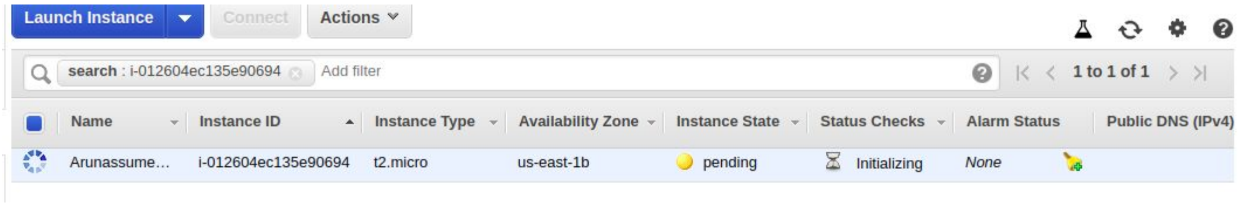
3. Attach this to an instance and get an sts token.

Create a new instance and then attach the new role(Gargi_assume_role)



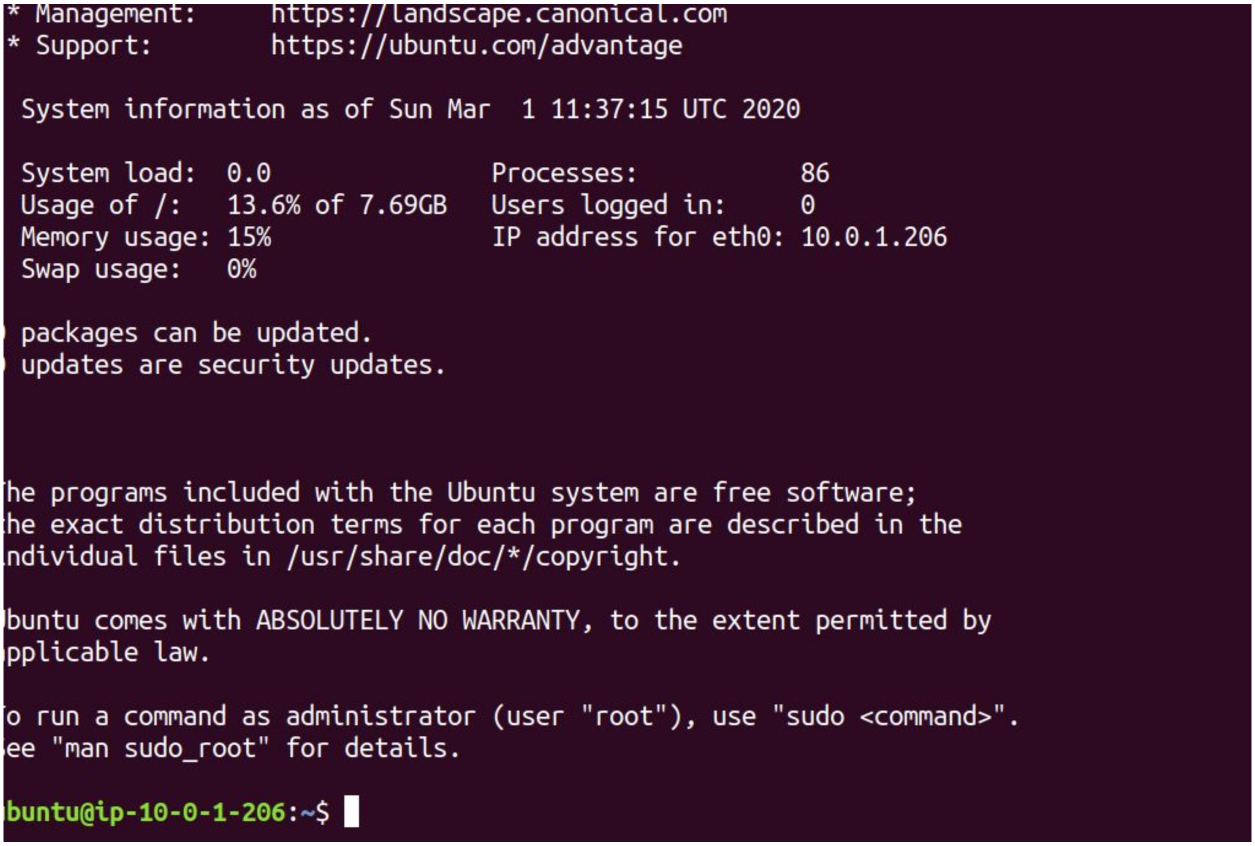
Step 3: Configure Instance Details

Network: vpc-00470a42fc196d84e | sarthak
Subnet: subnet-01d770a77bb69a1f8 | sarthak-load-balncer-1
218 IP Addresses available
Auto-assign Public IP: Use subnet setting (Enable)
Placement group: □ Add instance to placement group
Capacity Reservation: Open
IAM role: Arun-assumerole
Shutdown behavior: Stop
Enable termination protection: □ Protect against accidental termination
Monitoring: □ Enable CloudWatch detailed monitoring
Additional charges apply.
Tenancy: Shared - Run a shared hardware instance

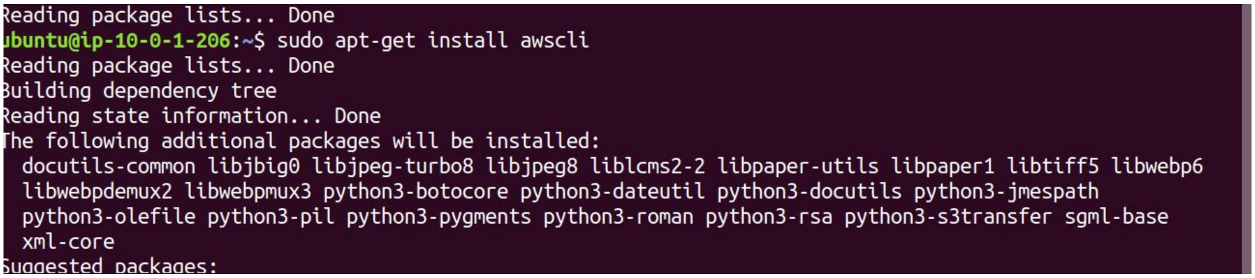


Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)
Arunassume…	i-012604ec135e90694	t2.micro	us-east-1b	pending	Initializing	None	

SSh into the instance and install awscli

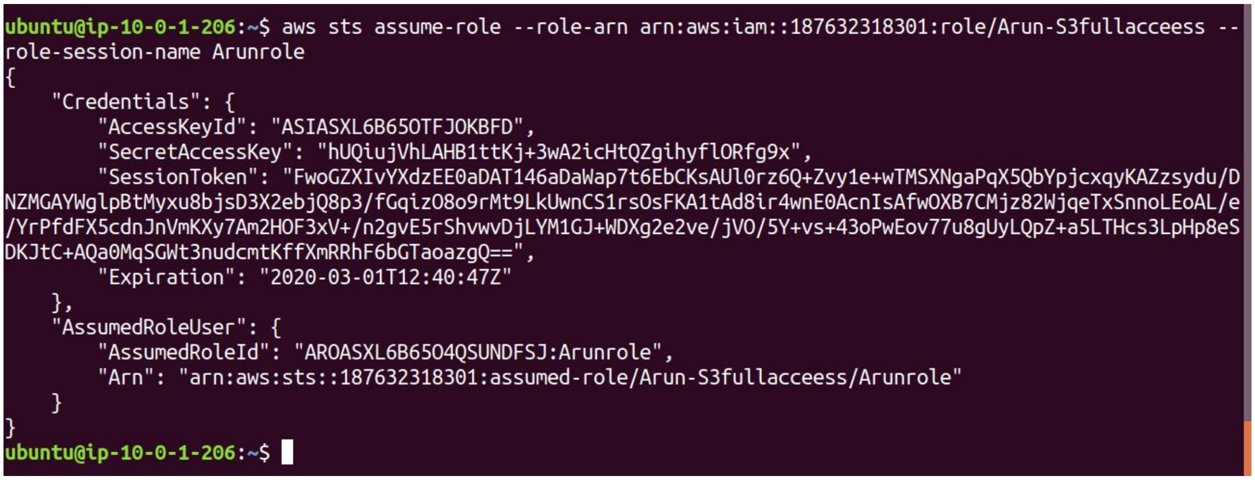


```
* Management:     https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage

 System information as of Sun Mar  1 11:37:15 UTC 2020

 System load:  0.0               Processes:           86
 Usage of /:   13.6% of 7.69GB   Users logged in:     0
 Memory usage: 15%               IP address for eth0: 10.0.1.206
 Swap usage:   0%

 packages can be updated.
 updates are security updates.



he programs included with the Ubuntu system are free software;
he exact distribution terms for each program are described in the
ndividual files in /usr/share/doc/*/copyright.

buntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
pplicable law.

o run a command as administrator (user "root"), use "sudo <command>".
ee "man sudo_root" for details.

buntu@ip-10-0-1-206:~$ 
```



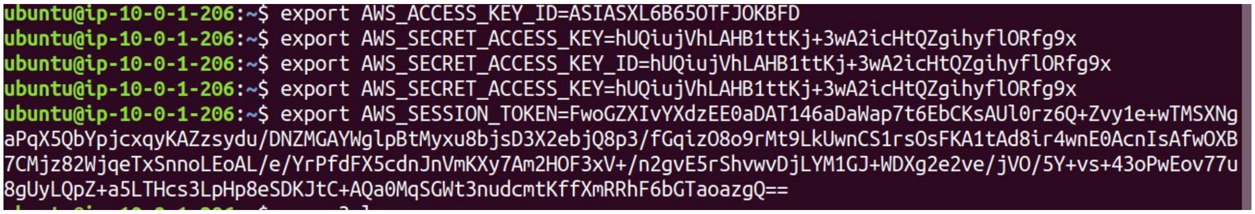
```
Reading package lists... Done
ubuntu@ip-10-0-1-206:~$ sudo apt-get install awscli
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  docutils-common libjbig0 libjpeg-turbo8 libjpeg8 liblcms2-2 libpaper-utils libpaper1 libtiff5 libwebp6
  libwebpdemux2 libwebpmux3 python3-botocore python3-dateutil python3-docutils python3-jmespath
  python3-olefile python3-pil python3-pygments python3-roman python3-rsa python3-s3transfer sgml-base
  xml-core
Suggested packages:
```

Now execute the following command :aws sts assume-role --role-arn arn:aws:iam::187632318301:role/Arun-S3fullacceess --role-session-name Arunrole to generate the sts token.



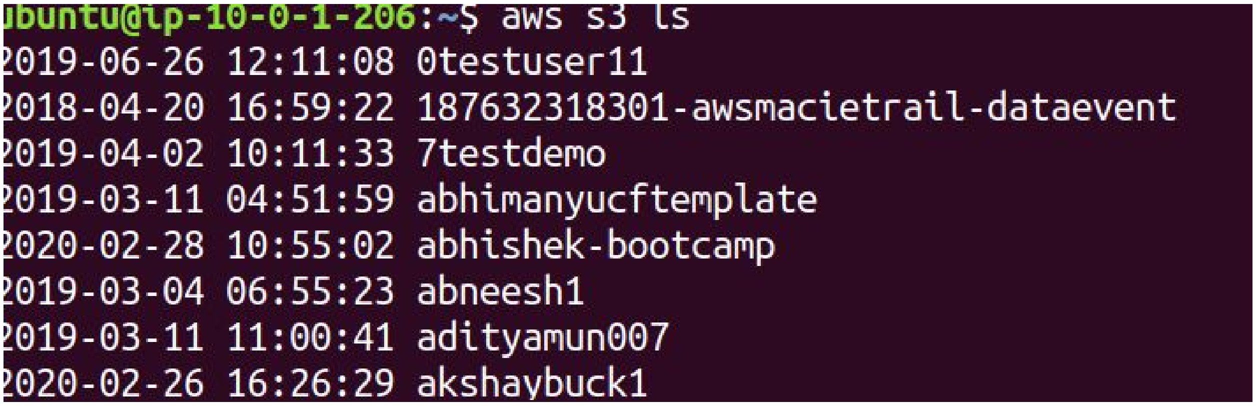
```
ubuntu@ip-10-0-1-206:~$ aws sts assume-role --role-arn arn:aws:iam::187632318301:role/Arun-S3fullacceess --
role-session-name Arunrole
{
    "Credentials": {
        "AccessKeyId": "ASIASXL6B65OTFJOKBFD",
        "SecretAccessKey": "hUQiujVhLAHB1ttKj+3wA2icHtQZgihyflORfg9x",
        "SessionToken": "FwoGZXIvYXdzEE0aDAT146aDaWap7t6EbCKsAUl0rz6Q+Zvy1e+wTMSXNgaPqX5QbYpjcxqyKAZzsydu/D
NZMGAYWglpBtMyxu8bjsD3X2ebjQ8p3/fGqizO8o9rMt9LkUwnCS1rsOsFKA1tAd8ir4wnE0AcnIsAfwOXB7CMjz82WjqeTxSnnoLEoAL/e
/YrPfdFX5cdnJnVmKXy7Am2HOF3xV+/n2gvE5rShvwvDjLYM1GJ+WDXg2e2ve/jVO/5Y+vs+43oPwEov77u8gUyLQpZ+a5LTHcs3LpHp8eS
DKJtC+AQa0MqSGWt3nudcmtKffXmRRhF6bGTaoazgQ==",
        "Expiration": "2020-03-01T12:40:47Z"
    },
    "AssumedRoleUser": {
        "AssumedRoleId": "AROASXL6B65O4QSUNDFSJ:Arunrole",
        "Arn": "arn:aws:sts::187632318301:assumed-role/Arun-S3fullacceess/Arunrole"
    }
}
ubuntu@ip-10-0-1-206:~$ 
```

Now export variables:



```
ubuntu@ip-10-0-1-206:~$ export AWS_ACCESS_KEY_ID=ASIASXL6B65OTFJOKBFD
ubuntu@ip-10-0-1-206:~$ export AWS_SECRET_ACCESS_KEY=hUQiujVhLAHB1ttKj+3wA2icHtQZgihyflORfg9x
ubuntu@ip-10-0-1-206:~$ export AWS_SECRET_ACCESS_KEY_ID=hUQiujVhLAHB1ttKj+3wA2icHtQZgihyflORfg9x
ubuntu@ip-10-0-1-206:~$ export AWS_SECRET_ACCESS_KEY=hUQiujVhLAHB1ttKj+3wA2icHtQZgihyflORfg9x
ubuntu@ip-10-0-1-206:~$ export AWS_SESSION_TOKEN=FwoGZXIvYXdzEE0aDAT146aDaWap7t6EbCKsAUl0rz6Q+Zvy1e+wTMSXNg
aPqX5QbYpjcxqyKAZzsydu/DNZMGAYWglpBtMyxu8bjsD3X2ebjQ8p3/fGqizO8o9rMt9LkUwnCS1rsOsFKA1tAd8ir4wnE0AcnIsAfwOXB
7CMjz82WjqeTxSnnoLEoAL/e/YrPfdFX5cdnJnVmKXy7Am2HOF3xV+/n2gvE5rShvwvDjLYM1GJ+WDXg2e2ve/jVO/5Y+vs+43oPwEov77u
8gUyLQpZ+a5LTHcs3LpHp8eSDKJtC+AQa0MqSGWt3nudcmtKffXmRRhF6bGTaoazgQ==
```

Now we can list all s3 buckets:



```
ubuntu@ip-10-0-1-206:~$ aws s3 ls
2019-06-26 12:11:08 0testuser11
2018-04-20 16:59:22 187632318301-awsmacietrail-dataevent
2019-04-02 10:11:33 7testdemo
2019-03-11 04:51:59 abhimanyucftemplate
2020-02-28 10:55:02 abhishek-bootcamp
2019-03-04 06:55:23 abneesh1
2019-03-11 11:00:41 adityamun007
2020-02-26 16:26:29 akshaybuck1
```

4. **Create a group for "Data Administrator" where the user 'Alice' is a member of this group. This group will prepare the data for the analysis. So Provide the following access to the group.**

   Service: Amazon S3;

   Action:

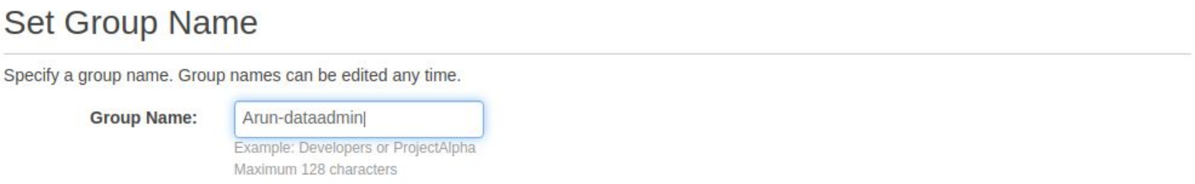
   Get*,

List*,

Put*,

ARN: Input and output Buckets (no conditions)

Ans. Step1: Create group named “dataAdministrator” and attach no policies



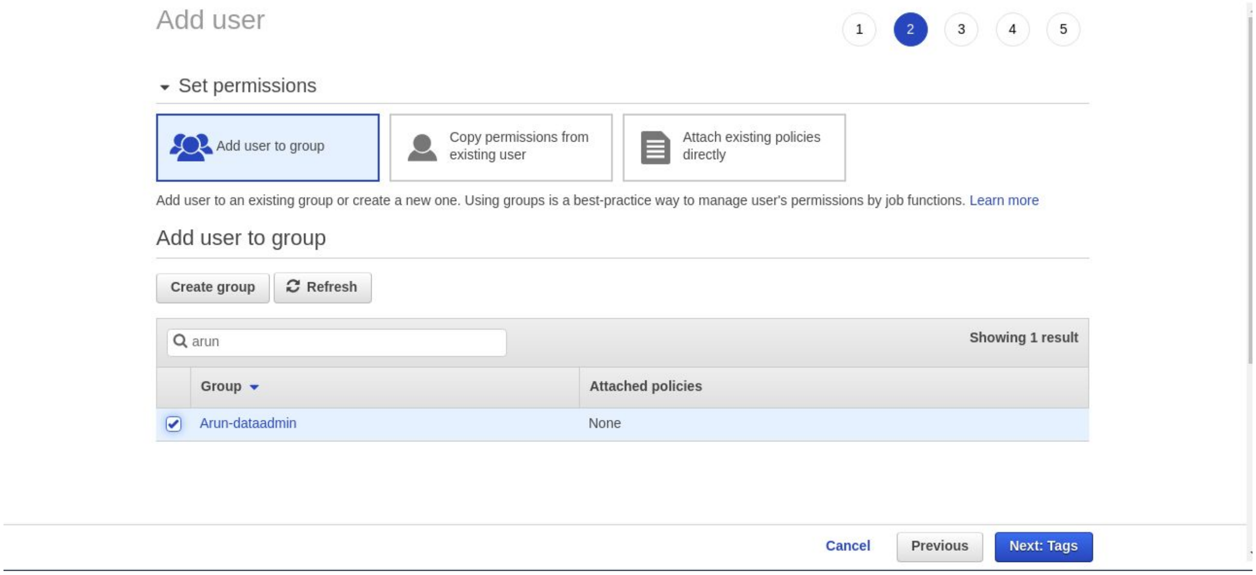
Set Group Name

Specify a group name. Group names can be edited any time.

Group Name: Arun-dataadmin

Example: Developers or ProjectAlpha
Maximum 128 characters

Step2: Create user named Alice and attach it to the group dataAdmin



Add user

Set permissions

Add user to group
Copy permissions from existing user
Attach existing policies directly

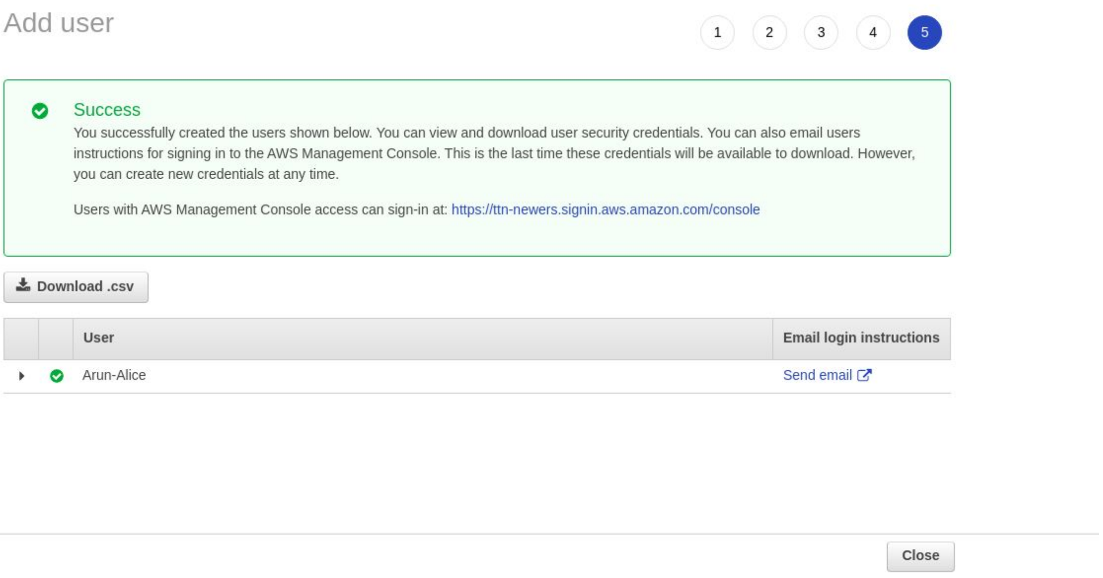
Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. Learn more

Add user to group

Search: arun — Showing 1 result

	Group	Attached policies
☑	Arun-dataadmin	None

Cancel
Previous
Next: Tags



Add user

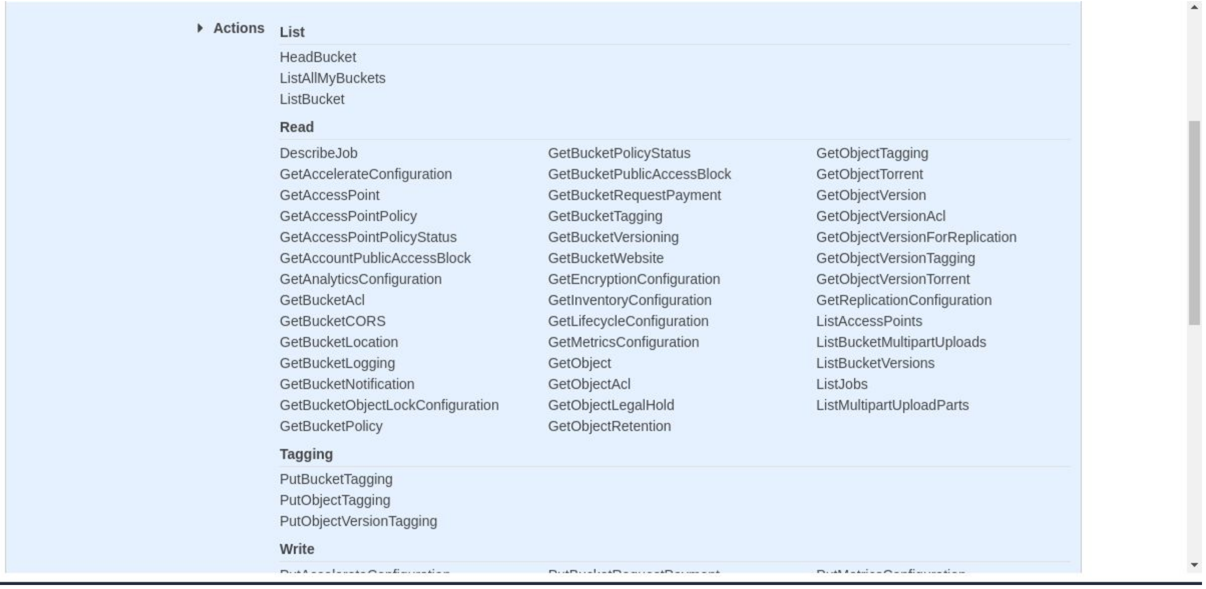
Success

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://ttn-newers.signin.aws.amazon.com/console

		User	Email login instructions
▸	✓	Arun-Alice	Send email

Step3: Create policies and provide get(all),list(all),put(all) and resources(all any)



List

HeadBucket
ListAllMyBuckets
ListBucket

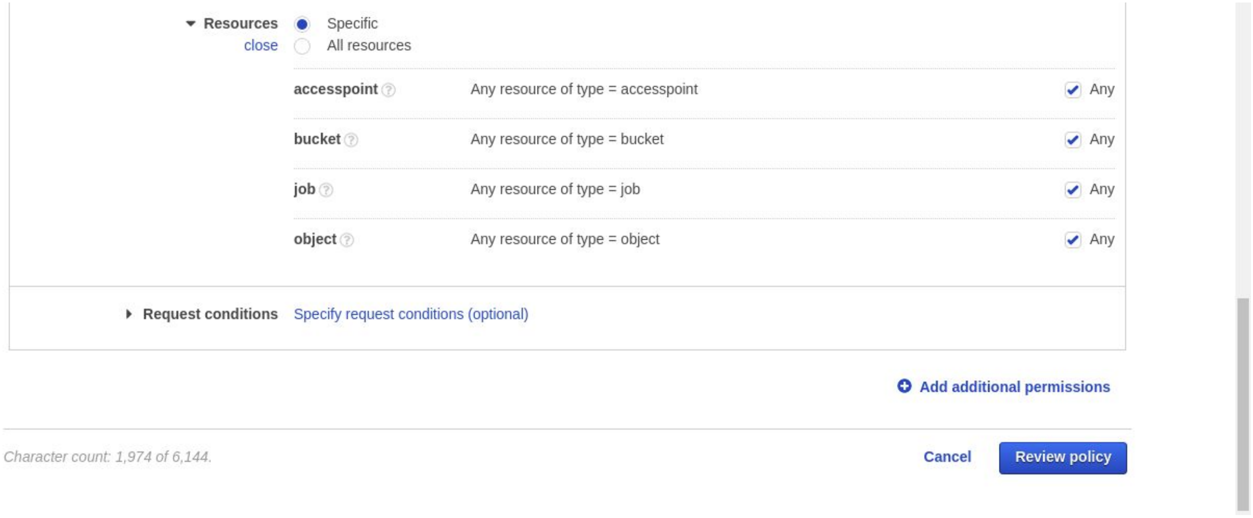
Read

DescribeJob
GetAccelerateConfiguration
GetAccessPoint
GetAccessPointPolicy
GetAccessPointPolicyStatus
GetAccountPublicAccessBlock
GetAnalyticsConfiguration
GetBucketAcl
GetBucketCORS
GetBucketLocation
GetBucketLogging
GetBucketNotification
GetBucketObjectLockConfiguration
GetBucketPolicy
GetBucketPolicyStatus
GetBucketPublicAccessBlock
GetBucketRequestPayment
GetBucketTagging
GetBucketVersioning
GetBucketWebsite
GetEncryptionConfiguration
GetInventoryConfiguration
GetLifecycleConfiguration
GetMetricsConfiguration
GetObject
GetObjectAcl
GetObjectLegalHold
GetObjectRetention
GetObjectTagging
GetObjectTorrent
GetObjectVersion
GetObjectVersionAcl
GetObjectVersionForReplication
GetObjectVersionTagging
GetObjectVersionTorrent
GetReplicationConfiguration
ListAccessPoints
ListBucketMultipartUploads
ListBucketVersions
ListJobs
ListMultipartUploadParts

Tagging

PutBucketTagging
PutObjectTagging
PutObjectVersionTagging

Write

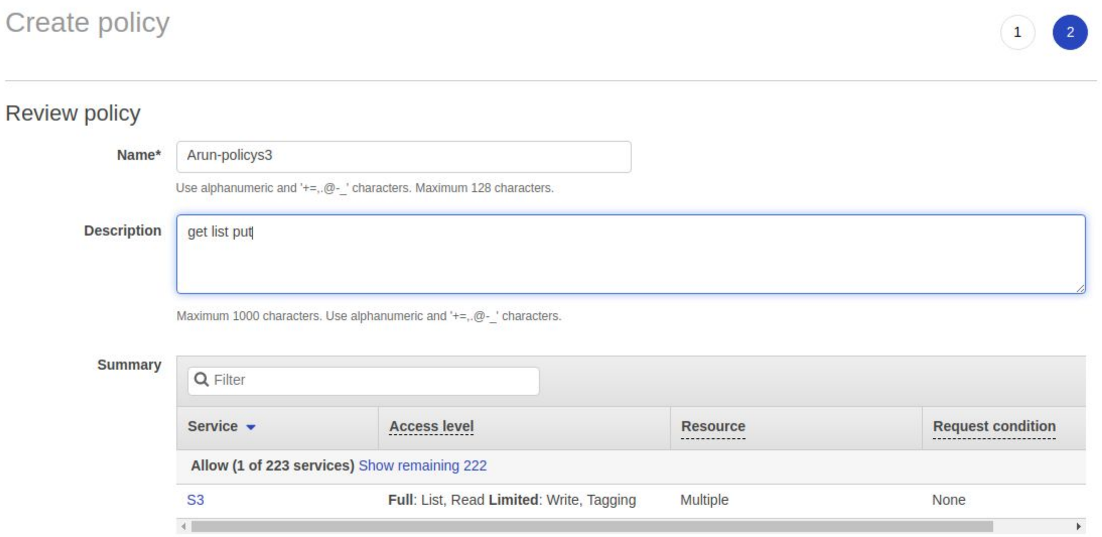


Resources ● Specific
close ○ All resources

accesspoint	Any resource of type = accesspoint	☑ Any
bucket	Any resource of type = bucket	☑ Any
job	Any resource of type = job	☑ Any
object	Any resource of type = object	☑ Any

Request conditions Specify request conditions (optional)

Character count: 1,974 of 6,144.



Review policy

Name* Arun-policys3

Use alphanumeric and '+=,.@-_' characters. Maximum 128 characters.

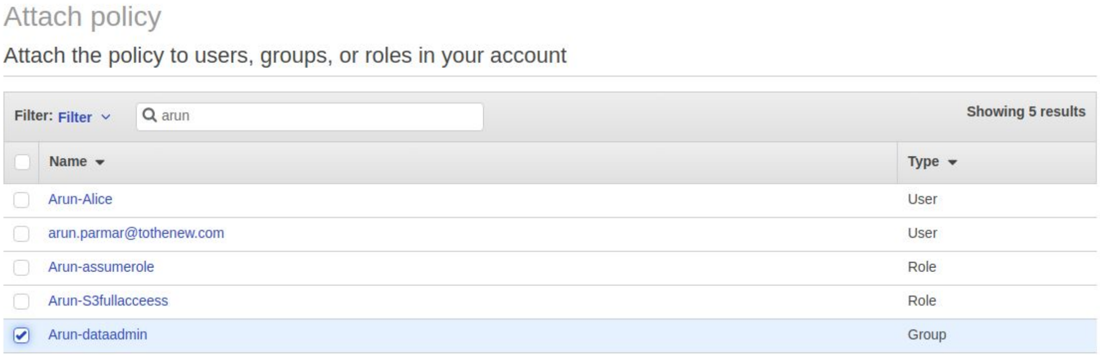
Description get list put

Maximum 1000 characters. Use alphanumeric and '+=,.@-_' characters.

Summary

Service	Access level	Resource	Request condition
Allow (1 of 223 services) Show remaining 222			
S3	**Full**: List, Read **Limited**: Write, Tagging	Multiple	None

Step4:Attach policy to the group

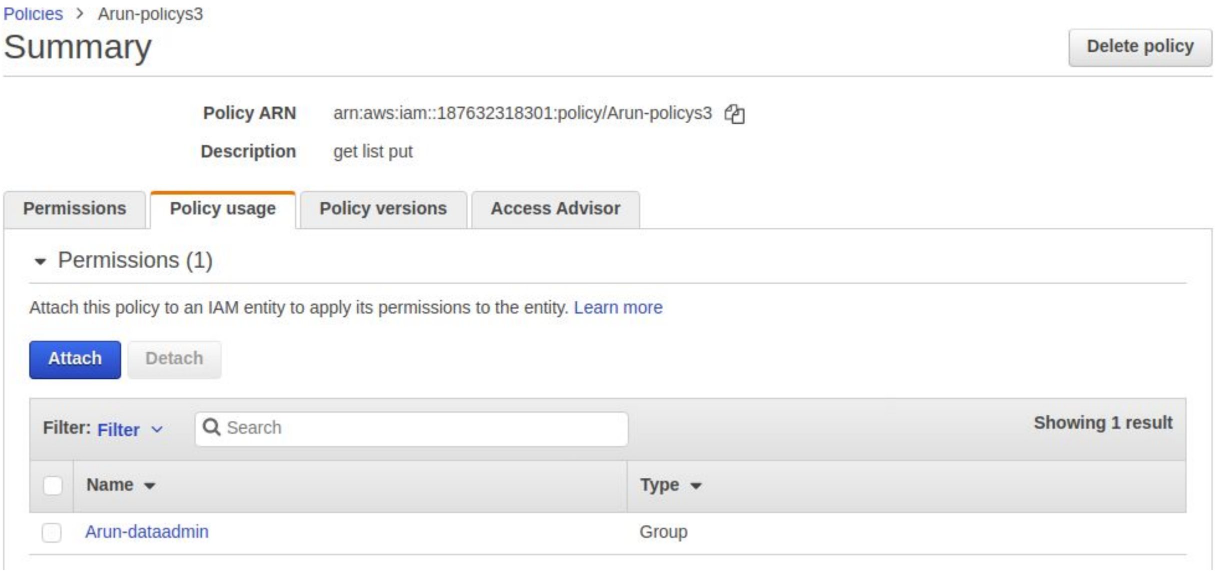


Attach policy

Attach the policy to users, groups, or roles in your account

Filter: Filter arun — Showing 5 results

☐	Name	Type
☐	Arun-Alice	User
☐	arun.parmar@tothenew.com	User
☐	Arun-assumerole	Role
☐	Arun-S3fullacceess	Role
☑	Arun-dataadmin	Group



Policies > Arun-policys3

Summary

Policy ARN arn:aws:iam::187632318301:policy/Arun-policys3

Description get list put

Permissions (1)

Attach this policy to an IAM entity to apply its permissions to the entity. Learn more

Filter: Filter Search — Showing 1 result

☐	Name	Type
☐	Arun-dataadmin	Group

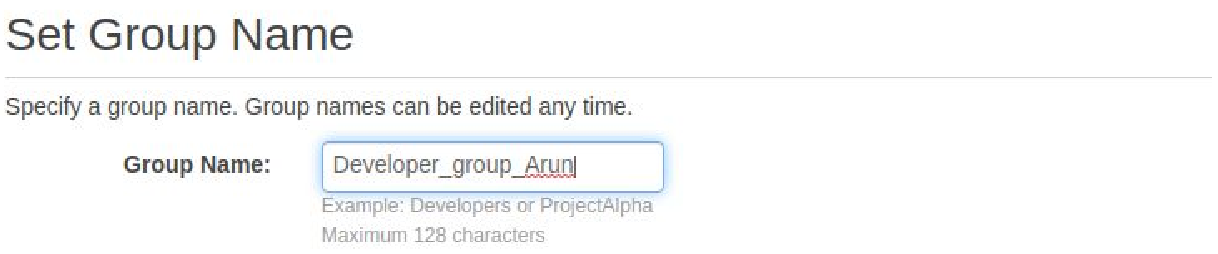
5. **Create a group for the "Developer group " where the user 'bob ' is a member of this group. This group with Test Newly Developed Features for which they require access to EC2 instances. Provide the following access to this group:**

   Service: Amazon EC2

   Action: *Instances, *Volume, Describe*, CreateTags;

   Condition: Dev Subnets only

   Step1: Create group “Develop_group-Arun”



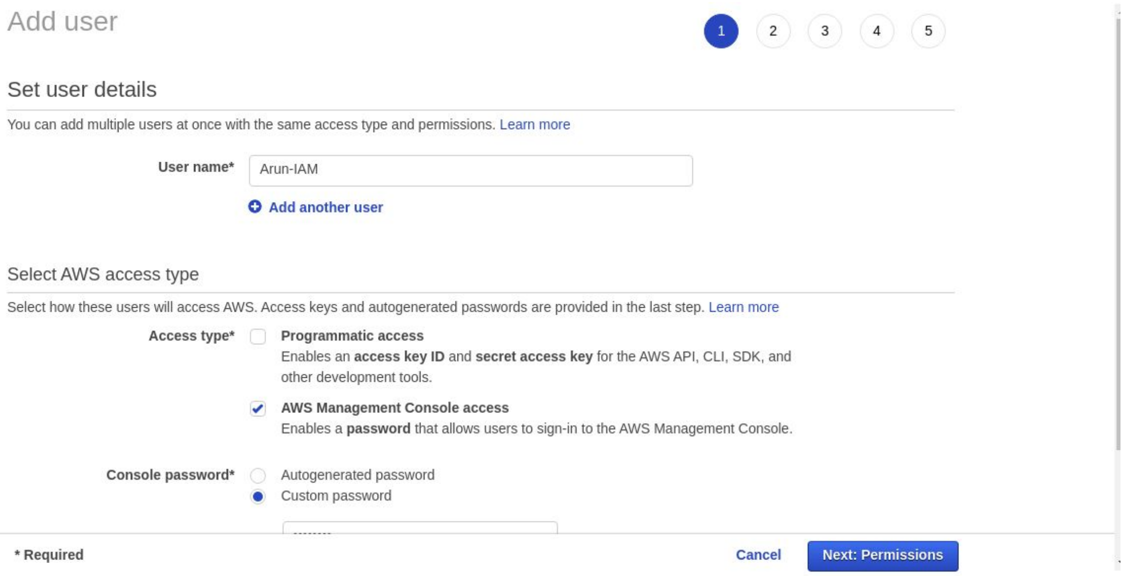
Set Group Name

Specify a group name. Group names can be edited any time.

Group Name: Developer_group_Arun

Example: Developers or ProjectAlpha
Maximum 128 characters

Step2: Create user “Arun-IAM”



Add user

Set user details

You can add multiple users at once with the same access type and permissions. Learn more

User name* Arun-IAM

Select AWS access type

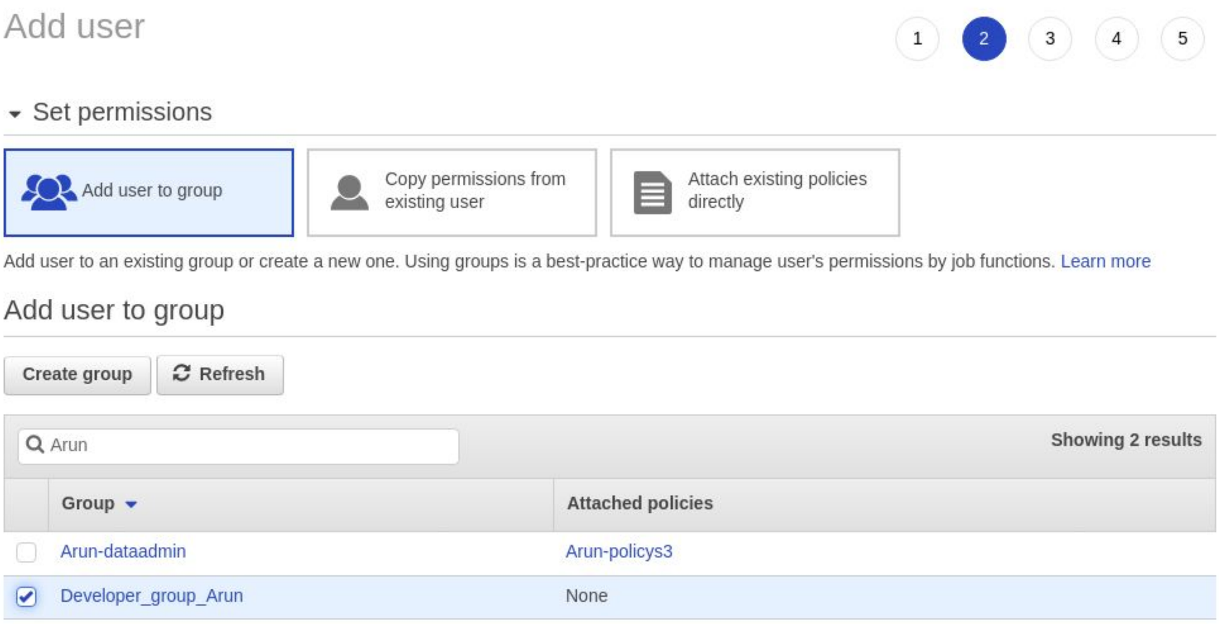
Select how these users will access AWS. Access keys and autogenerated passwords are provided in the last step. Learn more

Access type*
□ **Programmatic access**
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.

☑ **AWS Management Console access**
Enables a **password** that allows users to sign-in to the AWS Management Console.

Console password*
○ Autogenerated password
● Custom password

* Required



Add user

▾ Set permissions

Add user to group
Copy permissions from existing user
Attach existing policies directly

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. Learn more

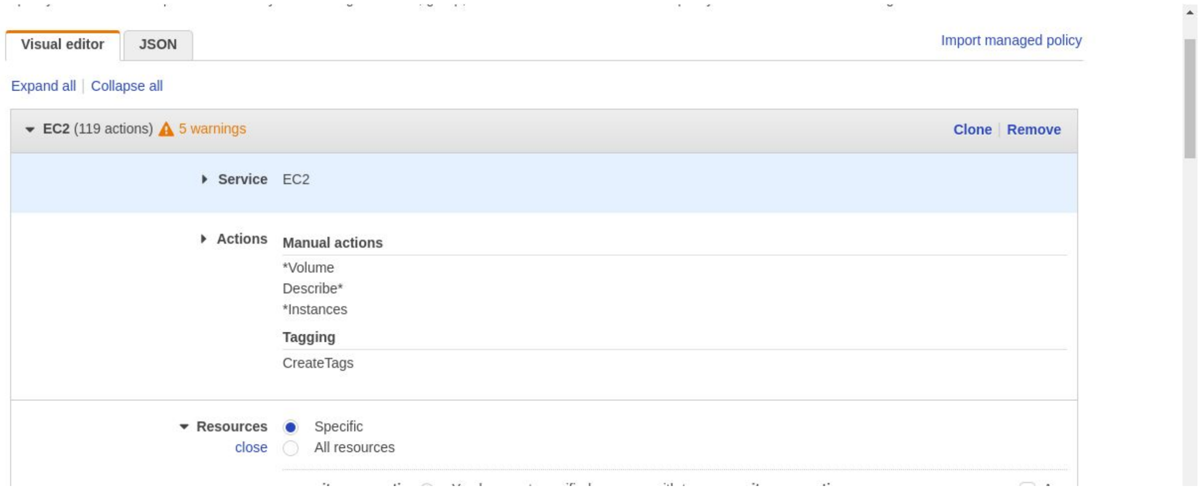
Add user to group

Arun

Showing 2 results

	Group ▾	Attached policies
□	Arun-dataadmin	Arun-policys3
☑	Developer_group_Arun	None

Step3: Create a policy “dev_group_policy” and specify the action and condition as mentioned in the question(Providing arn of Dev subnet)

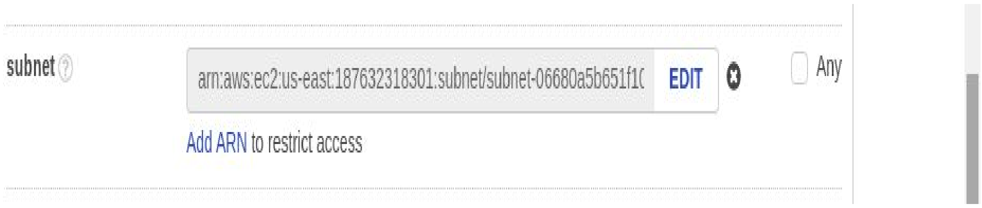


▾ EC2 (119 actions) ⚠ 5 warnings

Service EC2

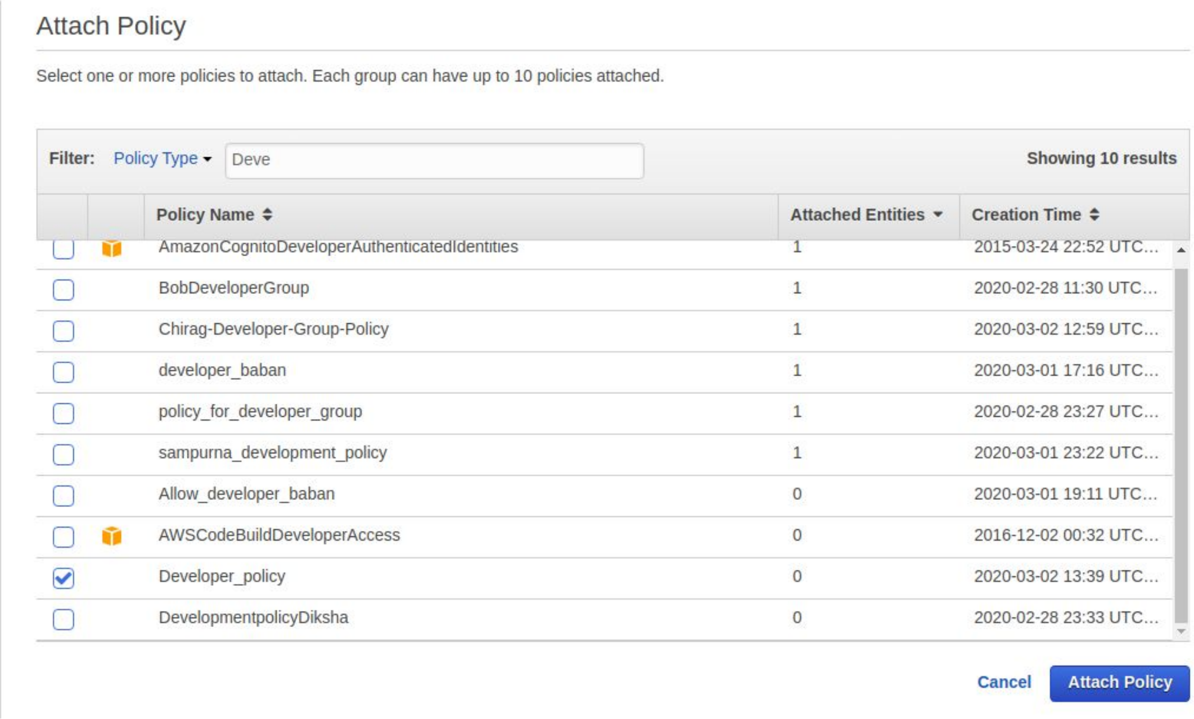
Actions
Manual actions
*Volume
Describe*
*Instances
Tagging
CreateTags

Resources ● Specific
close ○ All resources



subnet: arn:aws:ec2:us-east:187632318301:subnet/subnet-06680a5b651f1(

Step4: Attach the policy to the group



Attach Policy

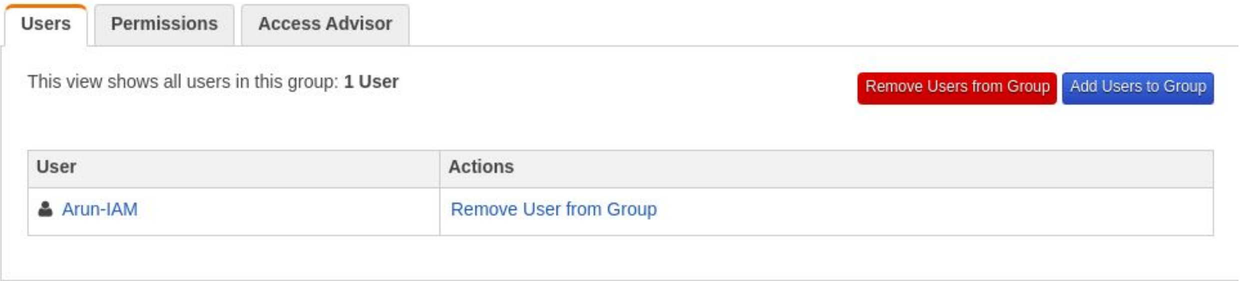
Select one or more policies to attach. Each group can have up to 10 policies attached.

Filter: Policy Type ▾ Deve

Showing 10 results

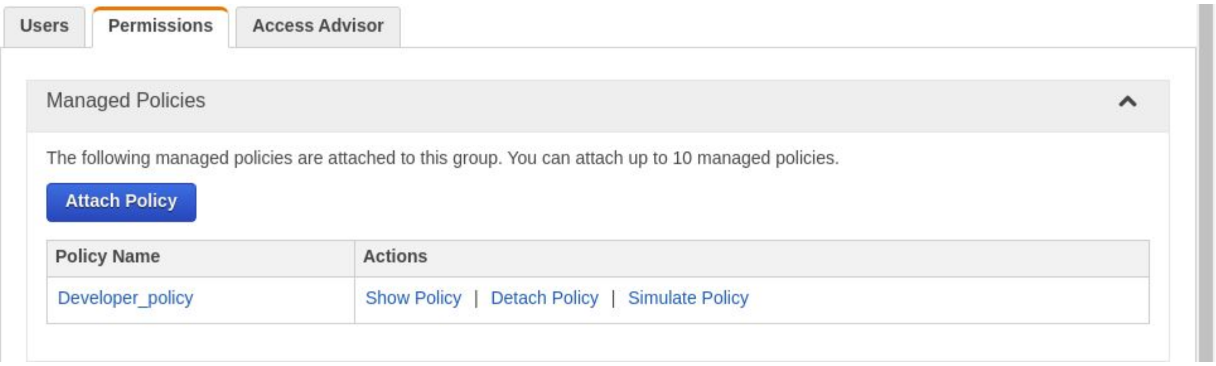
		Policy Name	Attached Entities	Creation Time
☐	▨	AmazonCognitoDeveloperAuthenticatedIdentities	1	2015-03-24 22:52 UTC...
☐		BobDeveloperGroup	1	2020-02-28 11:30 UTC...
☐		Chirag-Developer-Group-Policy	1	2020-03-02 12:59 UTC...
☐		developer_baban	1	2020-03-01 17:16 UTC...
☐		policy_for_developer_group	1	2020-02-28 23:27 UTC...
☐		sampurna_development_policy	1	2020-03-01 23:22 UTC...
☐		Allow_developer_baban	0	2020-03-01 19:11 UTC...
☐	▨	AWSCodeBuildDeveloperAccess	0	2016-12-02 00:32 UTC...
☑		Developer_policy	0	2020-03-02 13:39 UTC...
☐		DevelopmentpolicyDiksha	0	2020-02-28 23:33 UTC...

Step5:Open the group and check for the users and policies attached.



This view shows all users in this group: **1 User**

User	Actions
Arun-IAM	Remove User from Group

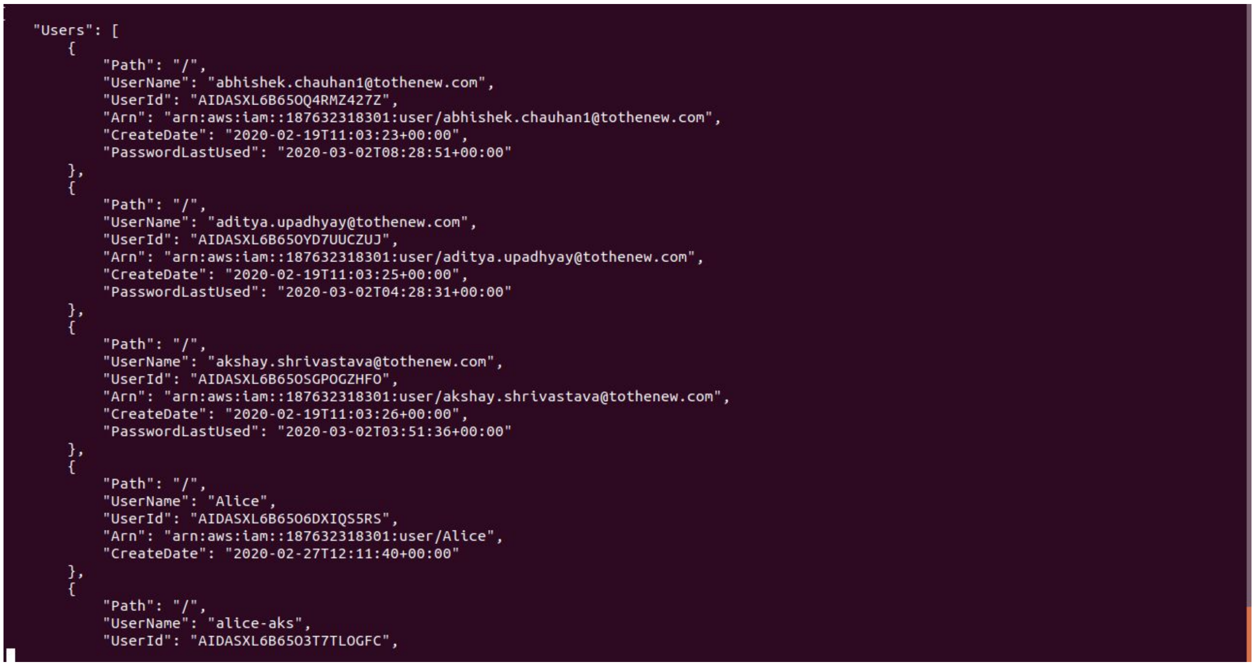


The following managed policies are attached to this group. You can attach up to 10 managed policies.

Policy Name	Actions
Developer_policy	Show Policy | Detach Policy | Simulate Policy

6. **Identify the unused IAM users/credentials using AWS CLI.**

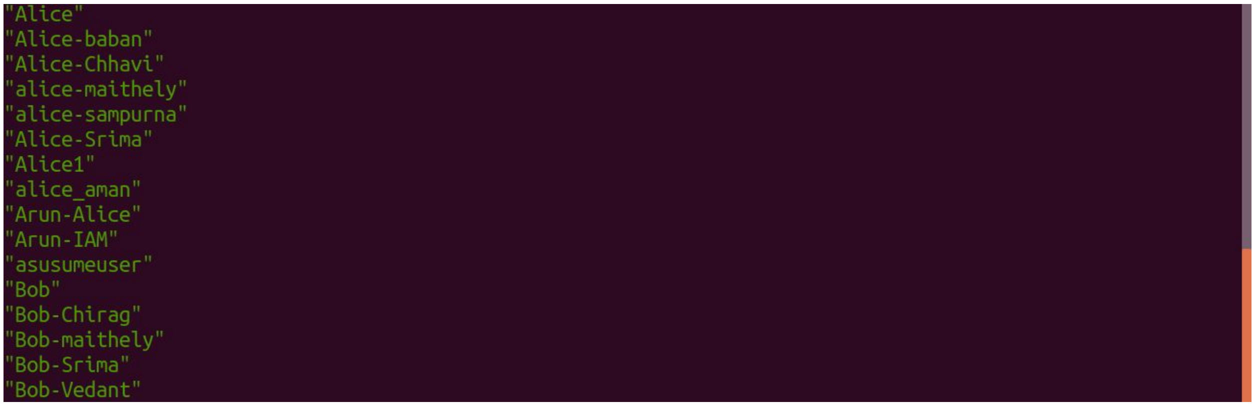
Step1: List all users and Install jq



```
    "Users": [
        {
            "Path": "/",
            "UserName": "abhishek.chauhan1@tothenew.com",
            "UserId": "AIDASXL6B65OQ4RMZ427Z",
            "Arn": "arn:aws:iam::187632318301:user/abhishek.chauhan1@tothenew.com",
            "CreateDate": "2020-02-19T11:03:23+00:00",
            "PasswordLastUsed": "2020-03-02T08:28:51+00:00"
        },
        {
            "Path": "/",
            "UserName": "aditya.upadhyay@tothenew.com",
            "UserId": "AIDASXL6B65OYD7UUCZUJ",
            "Arn": "arn:aws:iam::187632318301:user/aditya.upadhyay@tothenew.com",
            "CreateDate": "2020-02-19T11:03:25+00:00",
            "PasswordLastUsed": "2020-03-02T04:28:31+00:00"
        },
        {
            "Path": "/",
            "UserName": "akshay.shrivastava@tothenew.com",
            "UserId": "AIDASXL6B65OSGPOGZHFO",
            "Arn": "arn:aws:iam::187632318301:user/akshay.shrivastava@tothenew.com",
            "CreateDate": "2020-02-19T11:03:26+00:00",
            "PasswordLastUsed": "2020-03-02T03:51:36+00:00"
        },
        {
            "Path": "/",
            "UserName": "Alice",
            "UserId": "AIDASXL6B65O6DXIQS5RS",
            "Arn": "arn:aws:iam::187632318301:user/Alice",
            "CreateDate": "2020-02-27T12:11:40+00:00"
        },
        {
            "Path": "/",
            "UserName": "alice-aks",
            "UserId": "AIDASXL6B65O3T7TLOGFC",
```

jq(JSON QUERY) is like `sed` for JSON data - you can use it to slice and filter and map and transform structured data with the same ease that `sed`, `awk`, `grep` and friends let you play with text.

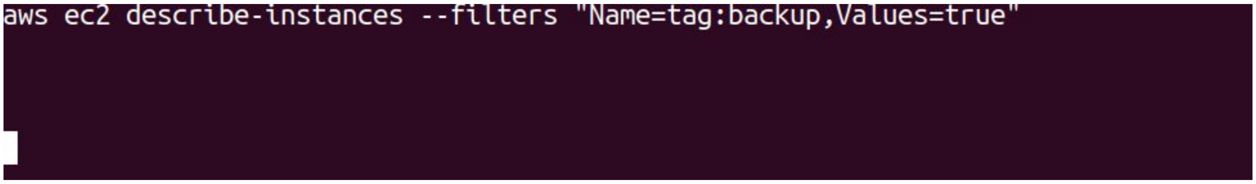
JQ Query: Aws iam list-users | jq '.Users[ ] | select(.PasswordLastUsed==null) | .UserName'



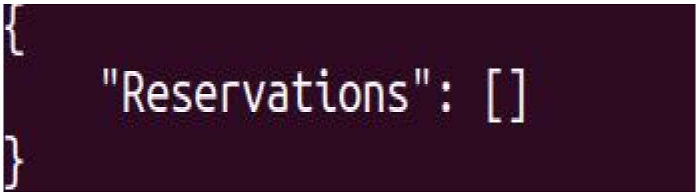
```
"Alice"
"Alice-baban"
"Alice-Chhavi"
"alice-maithely"
"alice-sampurna"
"Alice-Srima"
"Alice1"
"alice_aman"
"Arun-Alice"
"Arun-IAM"
"asusumeuser"
"Bob"
"Bob-Chirag"
"Bob-maithely"
"Bob-Srima"
"Bob-Vedant"
```

7. **Identify all the instances having the tag key-value "backup=true" using AWS CLI.**

Command: aws ec2 describe-instances --filters "Name=tag:backup,Values=true"



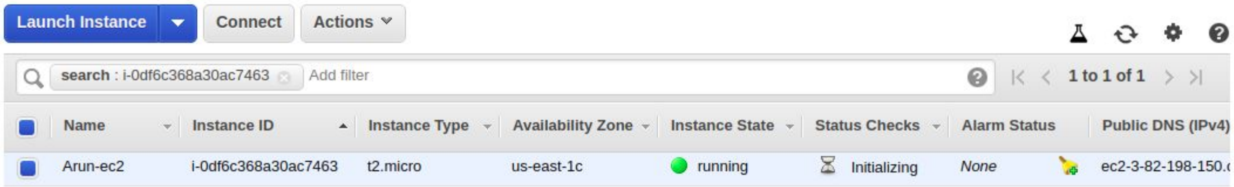
```
aws ec2 describe-instances --filters "Name=tag:backup,Values=true"
```



```
{
    "Reservations": []
}
```

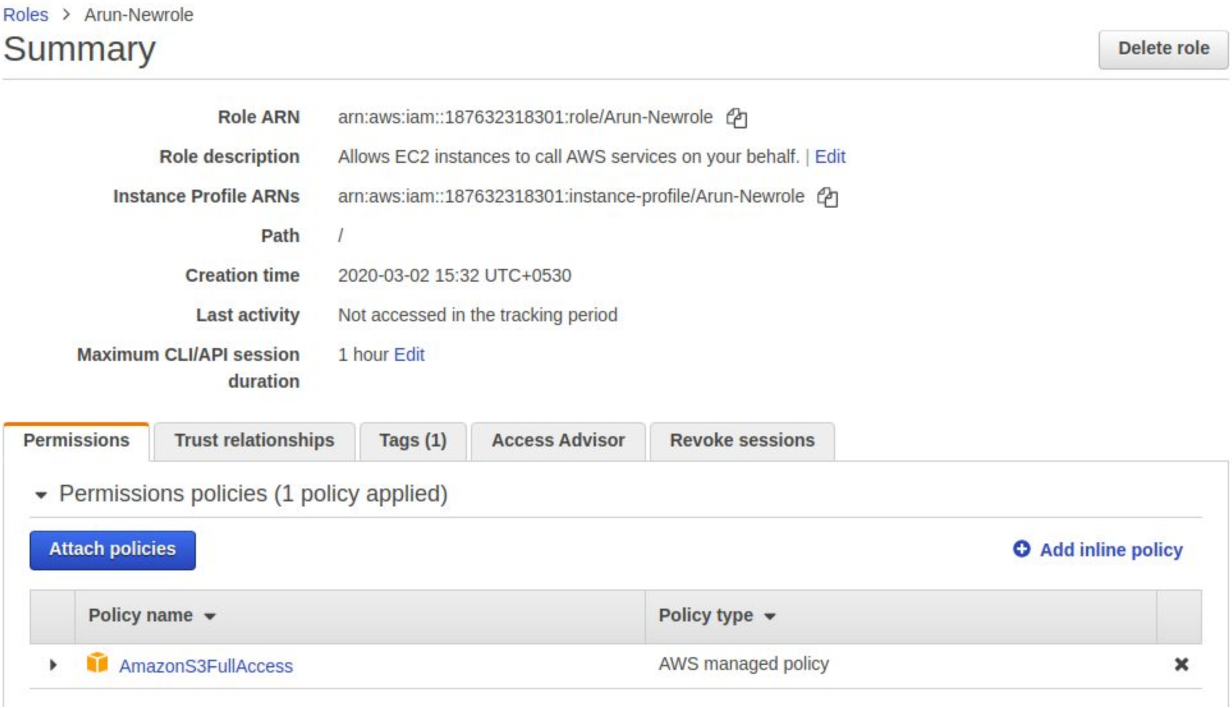
8. **An EC2 Instance hosts a Java-based application that accesses an s3 bucket. This EC2 Instance is currently serving production users. Create the role and assign the role to EC2 instance.**

Step1: Launch an EC2 instance:



	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)
■	Arun-ec2	i-0df6c368a30ac7463	t2.micro	us-east-1c	● running	Initializing	None	ec2-3-82-198-150.c

Step2:Create a role and attach S3 full access policy to it.

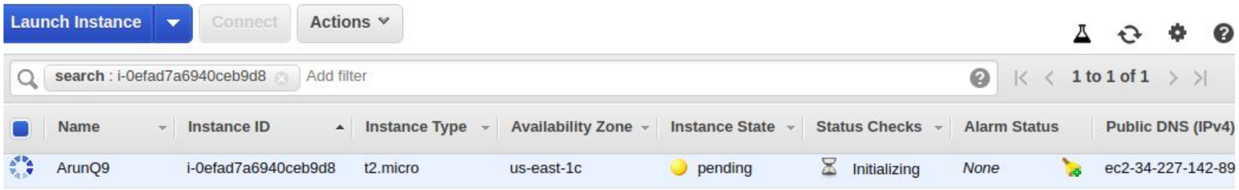


Summary

Role ARN	arn:aws:iam::187632318301:role/Arun-Newrole
Role description	Allows EC2 instances to call AWS services on your behalf. | Edit
Instance Profile ARNs	arn:aws:iam::187632318301:instance-profile/Arun-Newrole
Path	/
Creation time	2020-03-02 15:32 UTC+0530
Last activity	Not accessed in the tracking period
Maximum CLI/API session duration	1 hour Edit

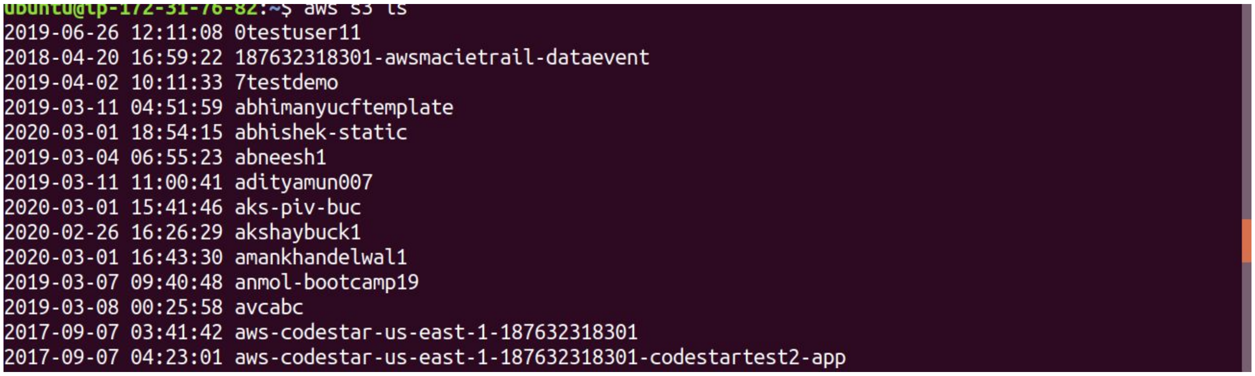
Policy name	Policy type
AmazonS3FullAccess	AWS managed policy

Step3:Attach the above role created to the EC2 instances.



Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)
ArunQ9	i-0efad7a6940ceb9d8	t2.micro	us-east-1c	pending	Initializing	None	ec2-34-227-142-89

Step4: ssh into instance, install awscli and run “aws s3 ls”



```
ubuntu@ip-172-31-76-82:~$ aws s3 ls
2019-06-26 12:11:08 0testuser11
2018-04-20 16:59:22 187632318301-awsmacietrail-dataevent
2019-04-02 10:11:33 7testdemo
2019-03-11 04:51:59 abhimanyucftemplate
2020-03-01 18:54:15 abhishek-static
2019-03-04 06:55:23 abneesh1
2019-03-11 11:00:41 adityamun007
2020-03-01 15:41:46 aks-piv-buc
2020-02-26 16:26:29 akshaybuck1
2020-03-01 16:43:30 amankhandelwal1
2019-03-07 09:40:48 anmol-bootcamp19
2019-03-08 00:25:58 avcabc
2017-09-07 03:41:42 aws-codestar-us-east-1-187632318301
2017-09-07 04:23:01 aws-codestar-us-east-1-187632318301-codestartest2-app
```

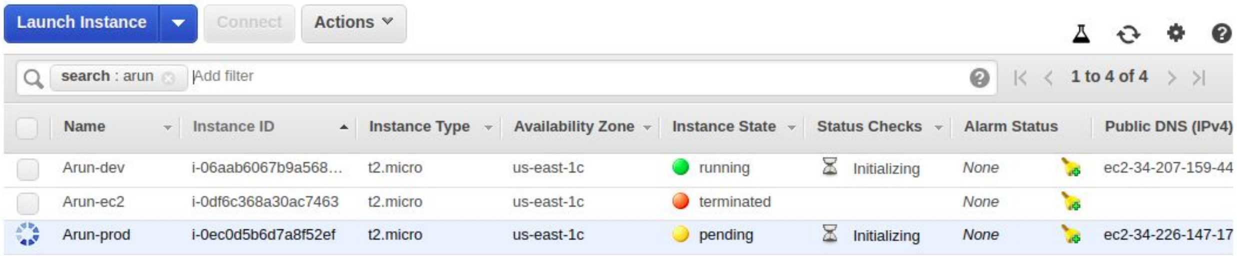
9. **You have both production and development based instances running on your VPC. It is required to ensure that people responsible for the development instances do not have access to work on production instances for better security. Define the tags on**

the test and production servers and add a condition to the IAMPolicy which allows access to specific tags.

ANS.

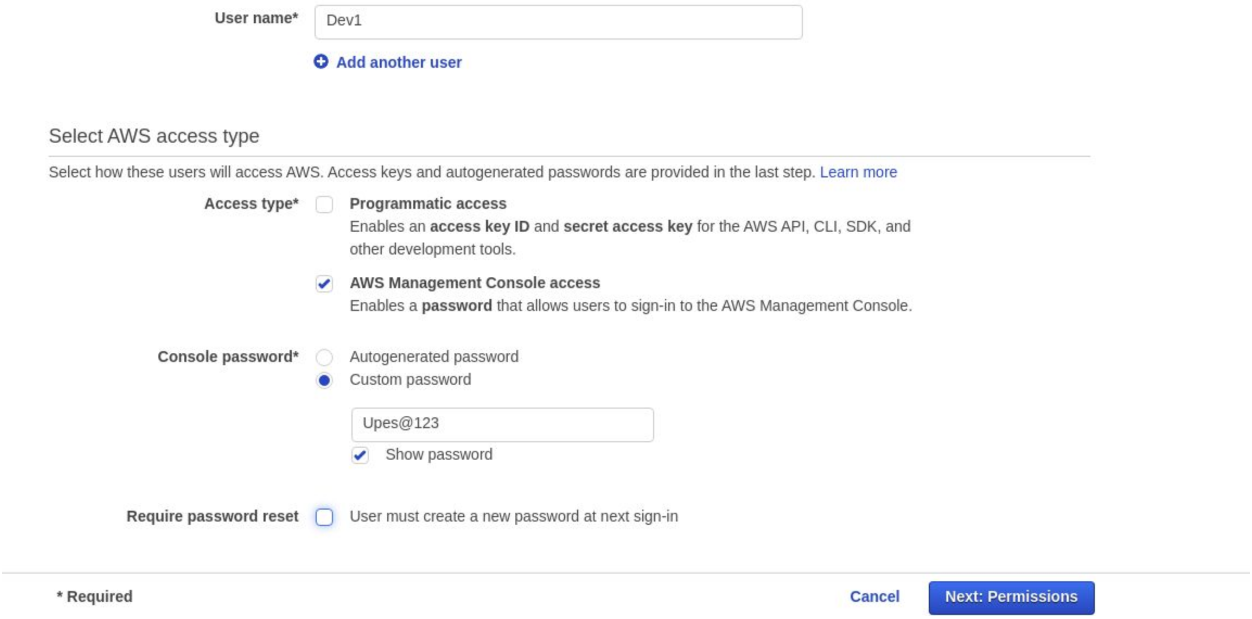
Step1: We will create two instances in the default VPC.

Arun-dev and Arun-prod



	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status		Public DNS (IPv4)
□	Arun-dev	i-06aab6067b9a568…	t2.micro	us-east-1c	🟢 running	⌛ Initializing	None		ec2-34-207-159-44
□	Arun-ec2	i-0df6c368a30c7463	t2.micro	us-east-1c	🔴 terminated		None		
	Arun-prod	i-0ec0d5b6d7a8f52ef	t2.micro	us-east-1c	🟡 pending	⌛ Initializing	None		ec2-34-226-147-17

Step2: Now create two users: Dev1-gargi and Prod1-gargi



User name* Dev1

Select AWS access type

Select how these users will access AWS. Access keys and autogenerated passwords are provided in the last step. Learn more

Access type* □ **Programmatic access**
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.

☑ **AWS Management Console access**
Enables a **password** that allows users to sign-in to the AWS Management Console.

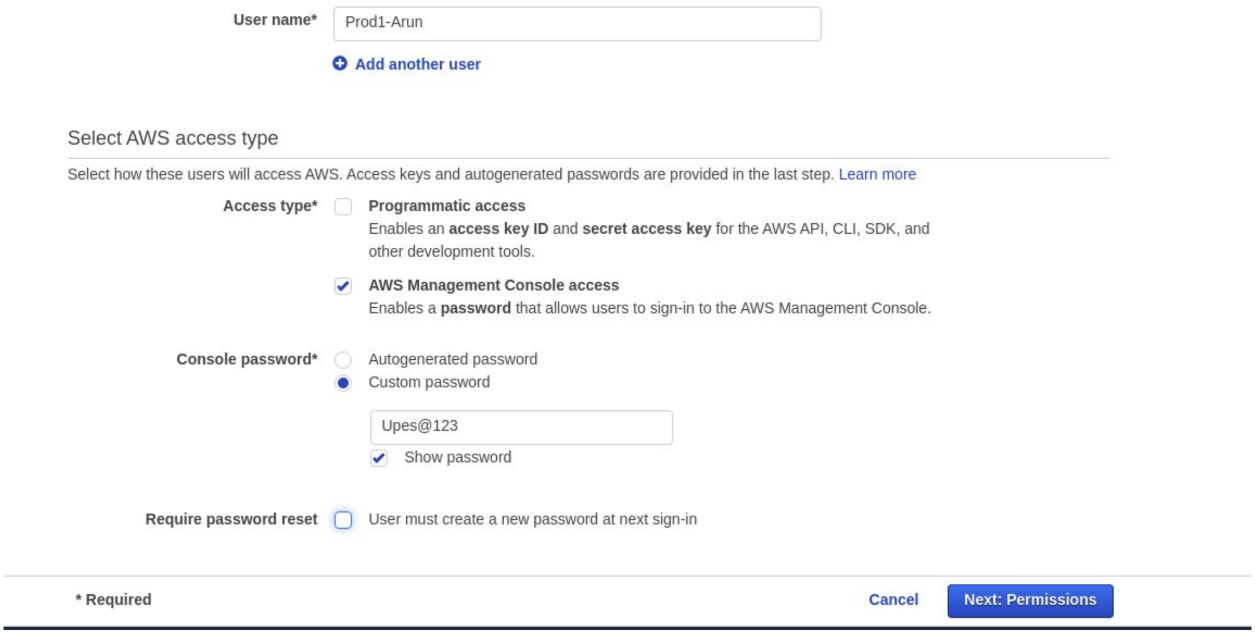
Console password* ○ Autogenerated password
● Custom password

Upes@123

☑ Show password

Require password reset □ User must create a new password at next sign-in

* Required



User name* Prod1-Arun

Select AWS access type

Select how these users will access AWS. Access keys and autogenerated passwords are provided in the last step. Learn more

Access type* □ **Programmatic access**
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.

☑ **AWS Management Console access**
Enables a **password** that allows users to sign-in to the AWS Management Console.

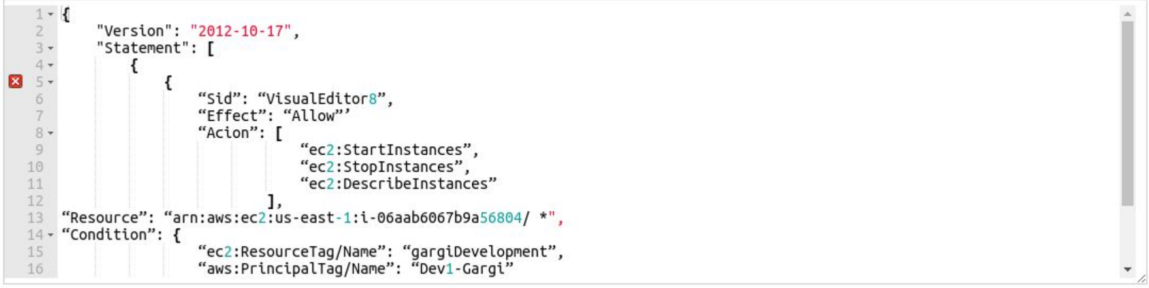
Console password* ○ Autogenerated password
● Custom password

Upes@123

☑ Show password

Require password reset □ User must create a new password at next sign-in

Step3:Now create a policy for the development server

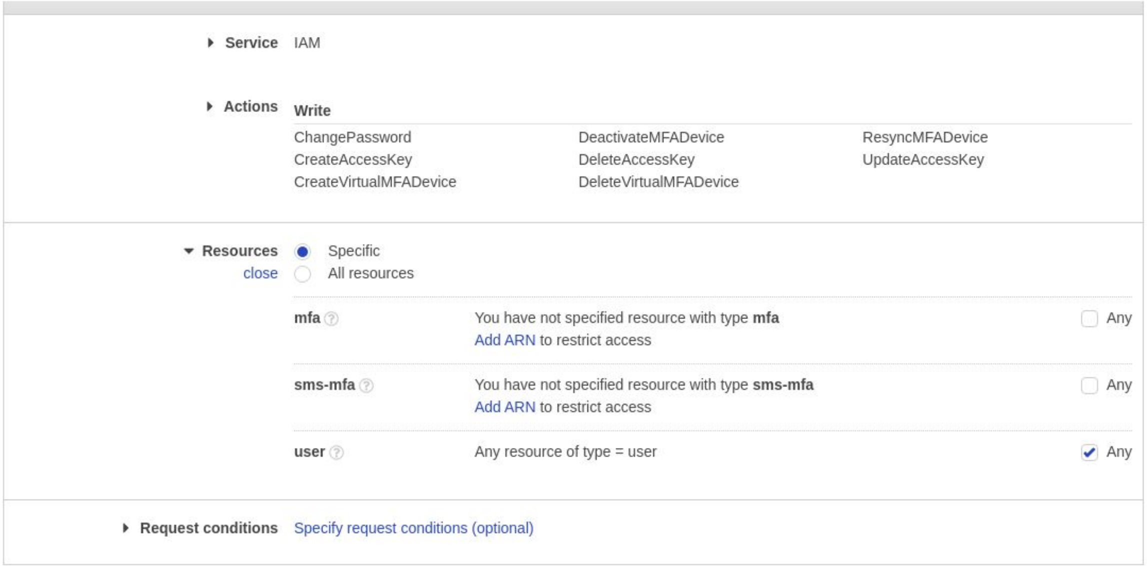


```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            {
                “Sid”: “VisualEditor8”,
                “Effect”: “Allow”’
                “Acion”: [
                            “ec2:StartInstances”,
                            “ec2:StopInstances”,
                            “ec2:DescribeInstances”
                        ],
“Resource”: “arn:aws:ec2:us-east-1:i-06aab6067b9a56804/ *",
“Condition”: {
                “ec2:ResourceTag/Name”: “gargiDevelopment”,
                “aws:PrincipalTag/Name”: “Dev1-Gargi”
```

Similarly do it for production server.

10. **Create a policy for allowing users to set or rotate their credentials, such as their console password, their programmatic access keys, and their MFA devices.**

    STEP 1: Create a policy and set service=IAM and give actions as per the question



Service IAM

Actions Write

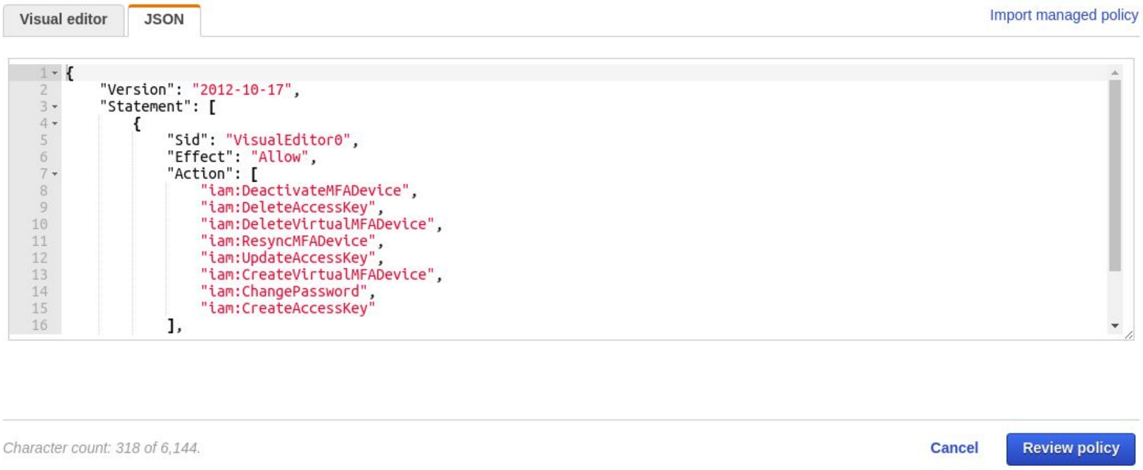
ChangePassword	DeactivateMFADevice	ResyncMFADevice
CreateAccessKey	DeleteAccessKey	UpdateAccessKey
CreateVirtualMFADevice	DeleteVirtualMFADevice	

Resources ● Specific ○ All resources

close

mfa	You have not specified resource with type **mfa** Add ARN to restrict access	□ Any
sms-mfa	You have not specified resource with type **sms-mfa** Add ARN to restrict access	□ Any
user	Any resource of type = user	☑ Any

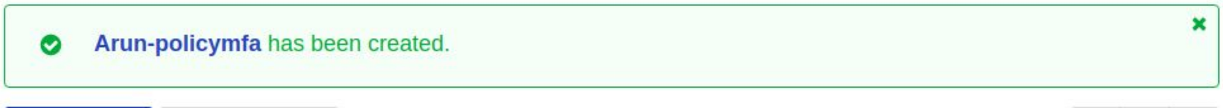
Request conditions Specify request conditions (optional)



```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "iam:DeactivateMFADevice",
                "iam:DeleteAccessKey",
                "iam:DeleteVirtualMFADevice",
                "iam:ResyncMFADevice",
                "iam:UpdateAccessKey",
                "iam:CreateVirtualMFADevice",
                "iam:ChangePassword",
                "iam:CreateAccessKey"
            ],
```

Character count: 318 of 6,144.

Policy has been created.



Arun-policymfa has been created.