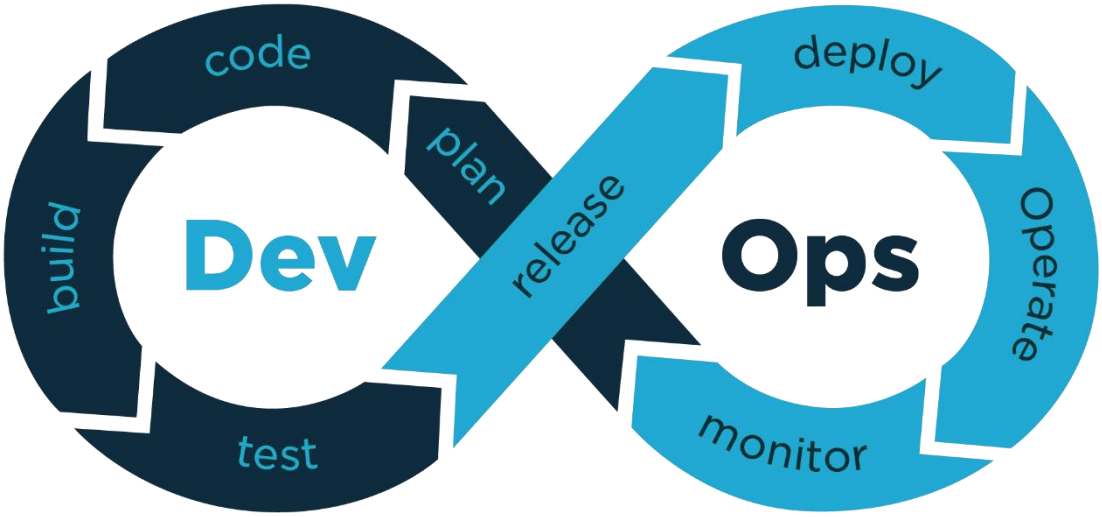


DevOps Project 101

Note : If you are looking for complete source code of this project, then please look at the last page of this document.

Contact : projectdevops101@gmail.com

Table of Contents

1.1 Goal :

This DevOps project is based on a real time DevOps project implemented in Banking and Finance domain.

Goal of Devops 101 Project is to deploy the website on nginx on AWS using terraform, gitlab, nginx, docker and using Gitlab CI CD tool for Devops pipeline.

We will create an IAAC (Infrastructure as a code) with the help of terraform for devops pipeline.

Terraform will use AWS s3 as it's backend to keep the state of terraform. The script setup_pipeline_state.sh will first check if the s3 bucket already exists, if it exists it will fetch the terraform state. If it does not exist (for the first time), it will create a new bucket with name starting with **dev-app-tf-state.**

We will will deploy the website on your EC2 machines with stages such as DEV, TEST, ACCEPTANECE, PRODUCTION.

You can deploy the Website on DEV, TEST, ACCEPTANCE, PRODUCTION environment with the help of a button in Gitlab UI.

AWS launch configuration is used for the launch configuration of EC2 instance.

Auto Scaling group will take care if by any chance any EC2 instance goes down it will launch a new EC2 instance with exact same configuration and your website will be automatically up within minutes. You can test this by going to Ec2 dashboard and terminating an instance. Within few seconds a new Ec2 instance will get launched.

Gitlab pipeline is used as a DevOps pipeline.

Gitlab-ci.yml configuration file is used for the Devops pipeline configuration. In that file devops pipeline is configured.

We have mentioned 9 stages.

Codeanalysis stage will check if our terraform code is correctly formatted. If it is not correctly formatted the stage will fail.

Dev_deploy stage will create all required resources in AWS. For e.g.

Security groups, elastic load balancer, autoscaling group, Aws launch configuration, ec2 instance, VPC and subnets.

Everything under the folder terraform -> dev-app will get triggered with terraform_deploy_ec2.sh file and it will launch the resources.

Dev_destroy stage will destroy all the resources created by our code in AWS account. Same has been implemented for other environments such a DEV, ACCEPTANCE and PRODUCTION.

Docker image for implementing gitlab runner (Please find source code in dockerfile under Runner folder in project). We use this image as a gitlab runner. We have mentioned our gitlab runner under .gitlab.ci file under CONTAINER_IMAGE value.

User_data.sh file under terraform -> dev-app will work as a userdata for our ec2 instance, means when ec2 instance will get launched user_data.sh file will install all required software's to run the website on ec2 machine. You can see the user_data.sh file logs by right clicking on ec2 instance -> Instance settings -> Get system logs.

1.2 Tools used:

1) Terraform
2) Amazon Web Services.
3) Gitlab (as a ci cd tool and as a Version control system).
4) Docker (As a Gitlab runner)
5) AWS Services
    a) EC2
    b) S3
    c) Security groups
    d) AWS EC2 launch configurations
    e) Auto scaling groups
    f) VPC and subnets
    g) Elastic load balancer
    h) Key Management Service
    i) IAM users and roles

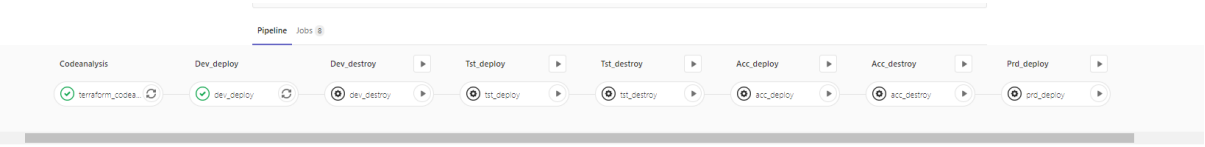
1.3 Tools required for you to deploy the project :

1) Gitlab.com Account (Free tier)
2) AWS Account (Free tier)
3) Intellije IDEA (Community edition 2018 or 2019)
4) Git (on your local machine)

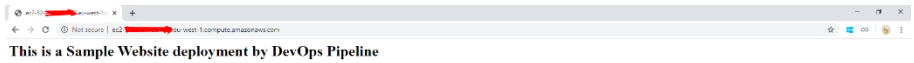
1.4 Output :

The below Devops pipeline will get created in your (free tier)own gitlab account. You will be deploy sample site of centos on your ec2 (AWS free tier machine) instance in your own AWS account.

1) DevOps pipeline in Gitlab account to deploy website on your AWS account.



2) Sample website deployed on your ec2 machine.



Section 2 : Deployment

2.1 Prerequisite

Please fulfill below requirement :

1) Gitlab.com Account (Free tier)

Create an account on https://gitlab.com

Please create a new account with new mail Id to avoid conflicts if you already have one.

2) AWS Account (Free tier)

Create a Amazon Web services account.

Please create a new account (https://aws.amazon.com/console/)with new mail Id to avoid conflicts if you already have one.

3) Intellije IDEA (Community edition 2018 or 2019)

Please download and install Intellije IDEA (2018 or 2019) but **ONLY Community edition**. As enterprise edition will keep you asking for the licence.
(https://www.jetbrains.com/idea/download/#section=windows)

Please download the software according to your operating system.

4) Git (on your local machine)

According to your operating system install git on your local machine(https://www.atlassian.com/git/tutorials/install-git)

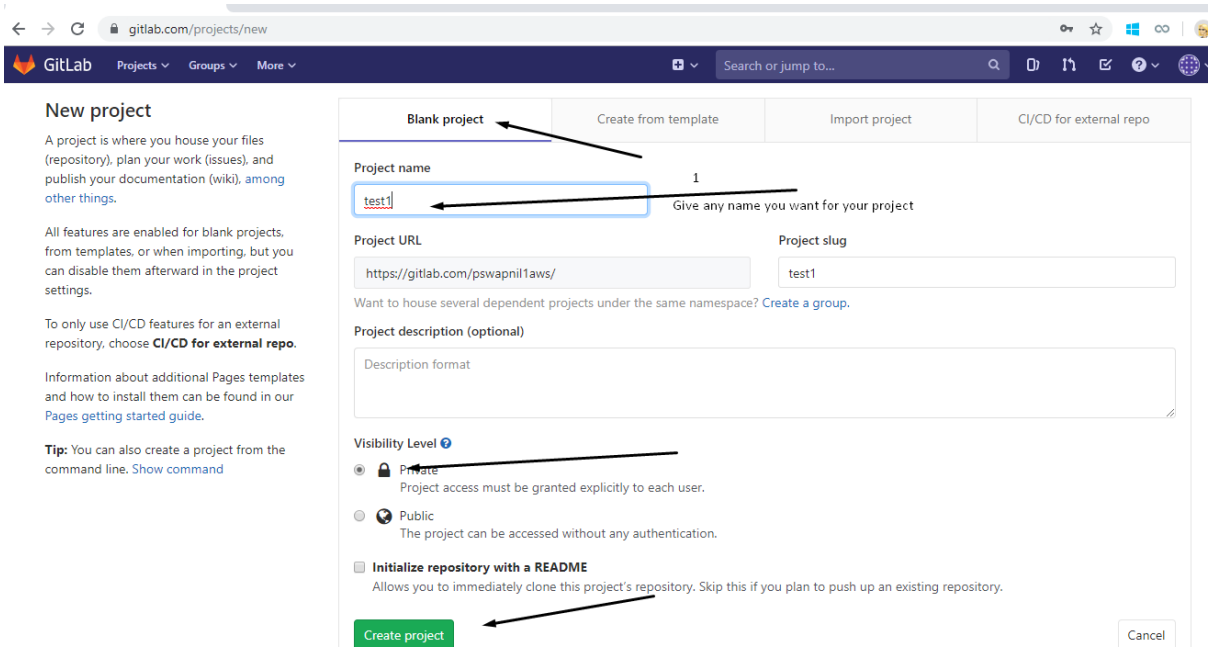
2.2 Steps to follow :

2.2.1 Copy the project.

Please copy the project in your local machine.

And import the project directory DevOps-101 in Intellije Idea in your local machine.

Create a new project in your gitlab account



2.2.2 Create a keypair

To be able to login to ec2 machine, Lets create a keypair in AWS console.

Go to your AWS console -> Services -> EC2 -> EC2 Dashboard -> Under that Select Key Pairs under Network & Security and select Create Key Pair.

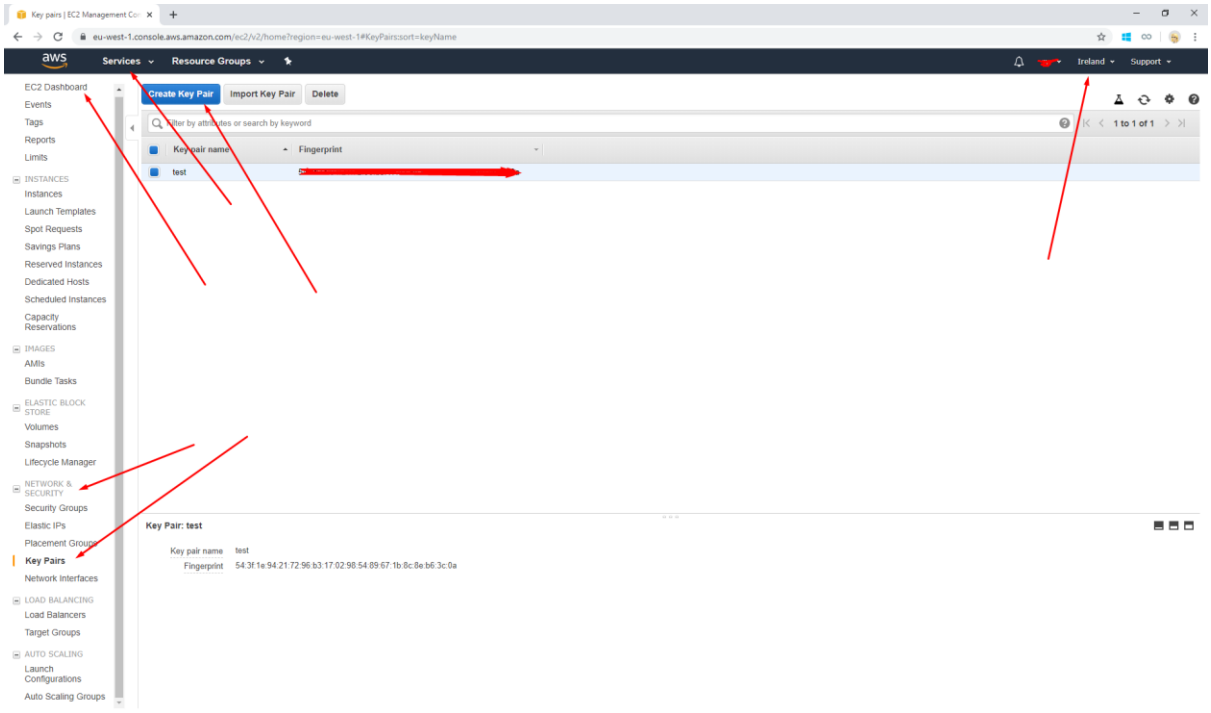
Give a name test to that keypair. If you give a different name to keypair, The code will not work.

Keypair will get downloaded at your local machine.

Please save this test.pem file at safe location.

You can use this key later to login to your EC2 machine.

Refer the image below.



2.2.3 Adding tag in vpc

Login to your AWS account and go to VPC section.

See below image for reference.

Please make sure the region on top is Ireland.

Under your "Your VPCs" You will see one default VPC.

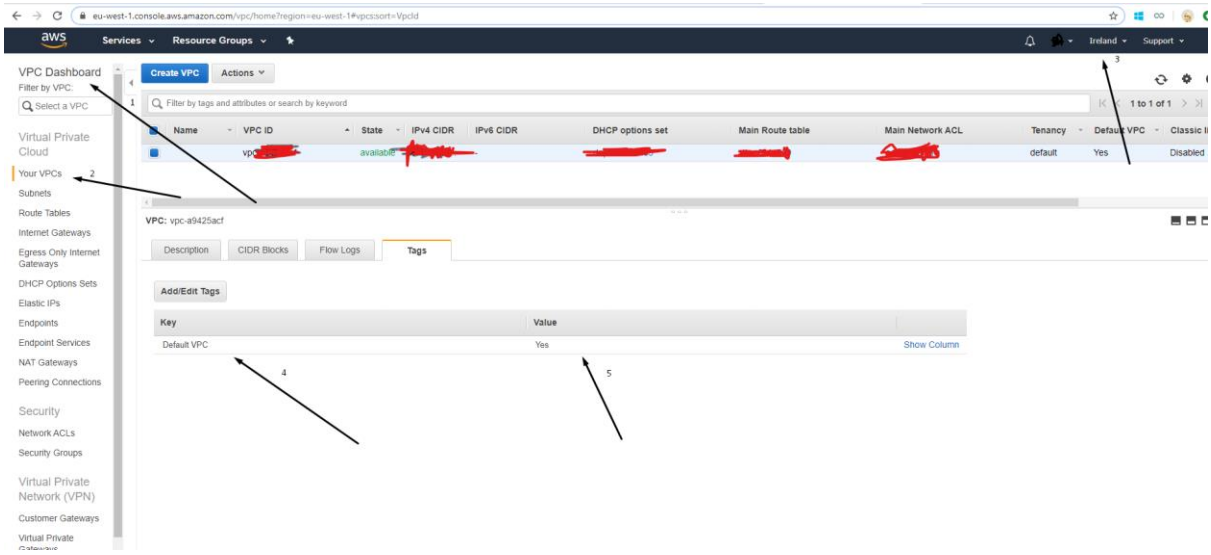
Select that VPC. And under the tags section enter below key and value.

Tag in vpc

Key : "Default VPC"

Value : "Yes"

Note : Do not include “” in tags.



2.2.4 Adding tag in subnet

Login to your AWS account.

Please make sure the region on top is Ireland.

Please refer two below images for reference.

Go to VPC dashboard then go to subnets.

You will see 3 default subnets over there.

Now we need to add tags to any 2 subnets amongst them.

Select any subnet and add following tag.

Tag in subnets

Key : Zone

Value : data

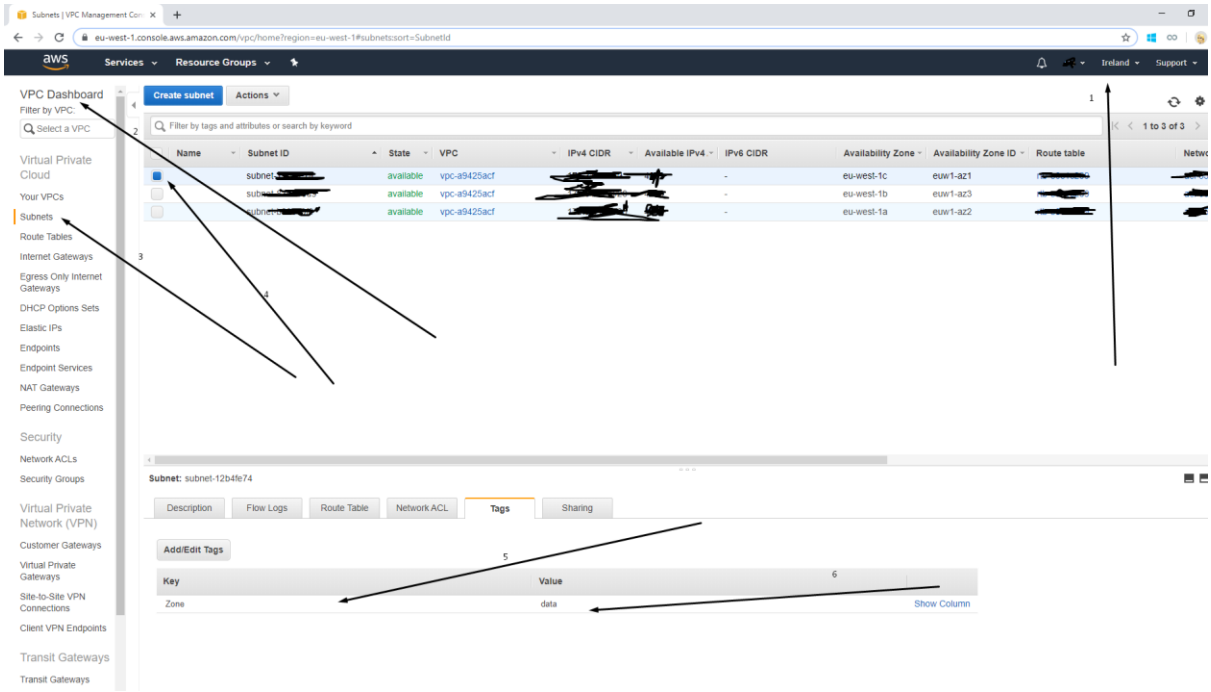
Now select any other subnet from remaining two subnets and

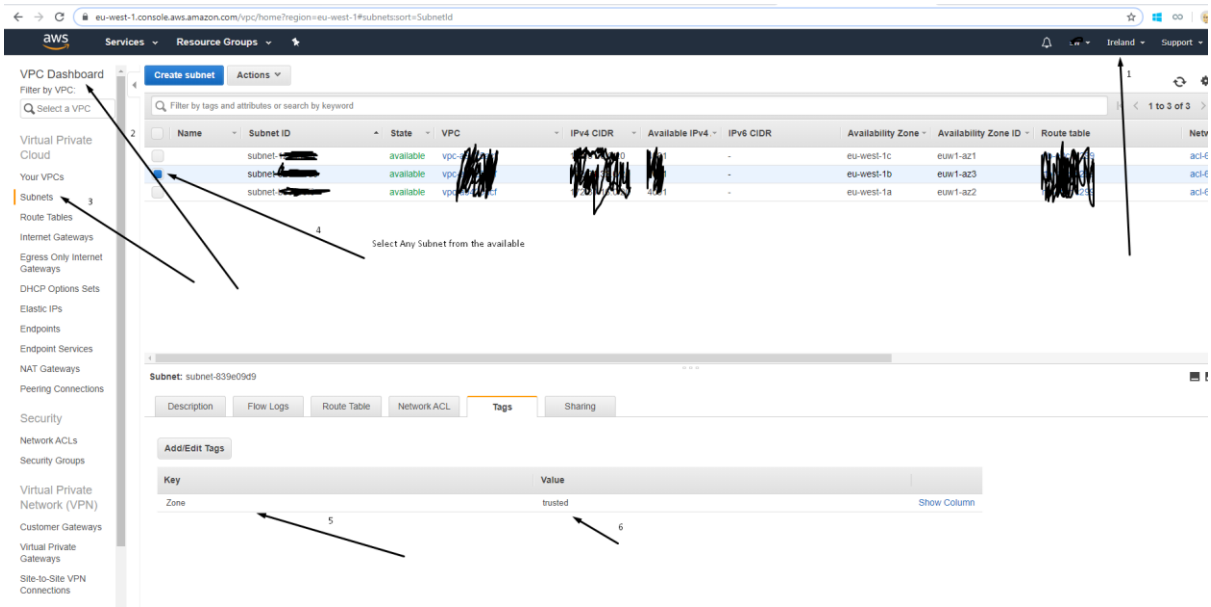
Add the following tag.

Tag in subnet.

Key : Zone

Value : trusted





2.2.5 Adding IAM User :

Create a user in AWS to be used for our terraform code.

Go to your AWS account and go to IAM service, select the option create user.

Give username as user3 only.

NOTE : If you give any other name to user then the code will not work as we have hardcoded username as user3 in our project.

SO MAKE SURE YOU GIVE THE NAME AS USER3

Refer the image below.

Give below permissions to user user3(Refer below image)

To attach permission go to IAM -> Users -> User3 -> Add permissions -> select attach existing policies directly -> In search option search all below 8 permissions and attach it to user3.

AmazonEC2FullAccess
ElasticLoadBalancingFullAccess
AmazonEC2ContainerRegistryFullAccess
AmazonS3FullAccess
IAMUserChangePassword
AmazonVPCFullAccess
AmazonEC2ContainerRegistryPowerUser
AWSKeyManagementServicePowerUser

Once you create the IAM user at the end it will show a Download option of your aws_access_key_id and aws_secret_access_key
credentials file. Please download this file and save at a safe location.

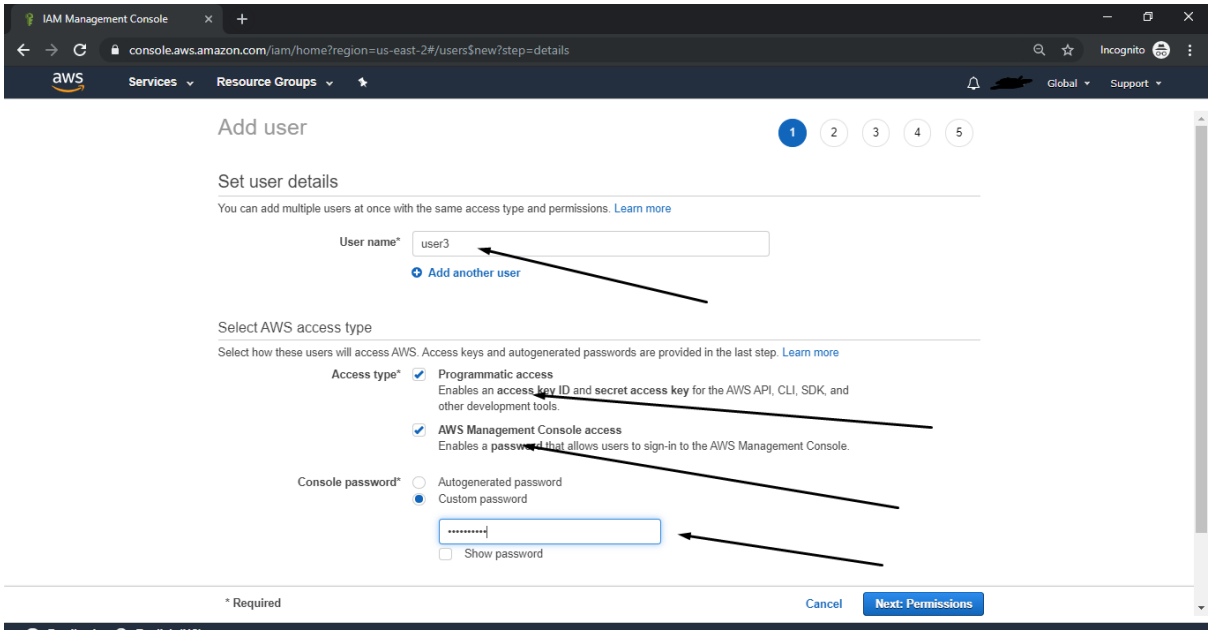
Now copy aws_access_key_id and aws_secret_access_key and
replace it in setup_pipeline_state.sh.

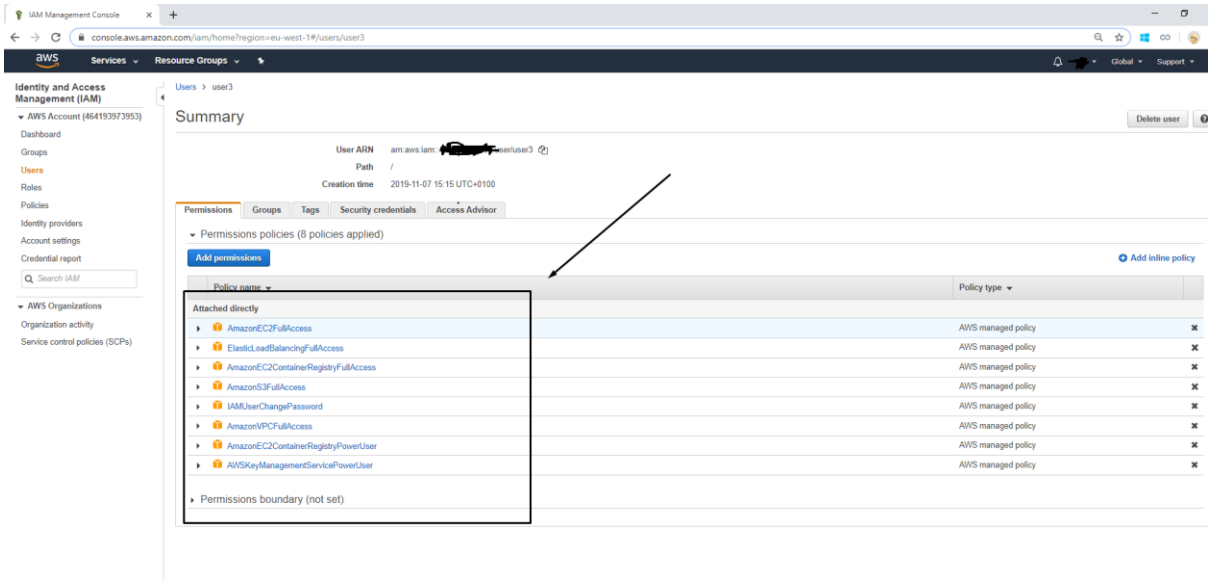
Replace *<YOUR-ACCESS_KEY> and <YOUR-SECRET-KEY>*
With your actual values.

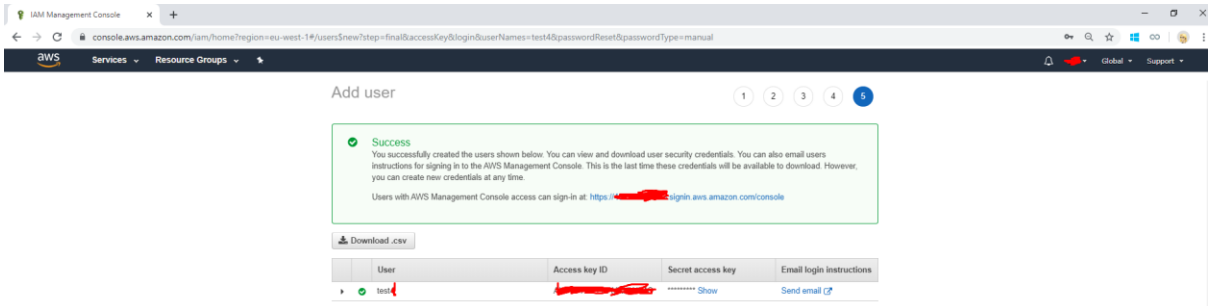
And also in file provider.tf file under setup -> state-storage-bucket -> provider.tf

Replace the values of **<YOUR-ACCESS_KEY> and <YOUR-SECRET-KEY>**

With your actual values from the downloaded file.


IAM Management Console
console.aws.amazon.com/iam/home?region=us-east-2#/users$new?step=details
Incognito
aws
Services
Resource Groups
Global
Support
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*
user3
Add another user
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
Show password
* Required
Cancel
Next: Permissions


IAM Management Console
console.aws.amazon.com/iam/home?region=eu-west-1#/users/user3
aws
Services
Resource Groups
Global
Support
Identity and Access Management (IAM)
AWS Account (464193973953)
Dashboard
Groups
Users
Roles
Policies
Identity providers
Account settings
Credential report
Search IAM
AWS Organizations
Organization activity
Service control policies (SCPs)
Users > user3
Summary
Delete user
User ARN
Path
Creation time
2019-11-07 15:15 UTC+0100
Permissions
Groups
Tags
Security credentials
Access Advisor
Permissions policies (8 policies applied)
Add permissions
Add inline policy
Policy name
Policy type
Attached directly
AmazonEC2FullAccess
ElasticLoadBalancingFullAccess
AmazonEC2ContainerRegistryFullAccess
AmazonS3FullAccess
IAMUserChangePassword
AmazonVPCFullAccess
AmazonEC2ContainerRegistryPowerUser
AWSKeyManagementServicePowerUser
AWS managed policy
Permissions boundary (not set)



2.2.6 Adding IAM Role :

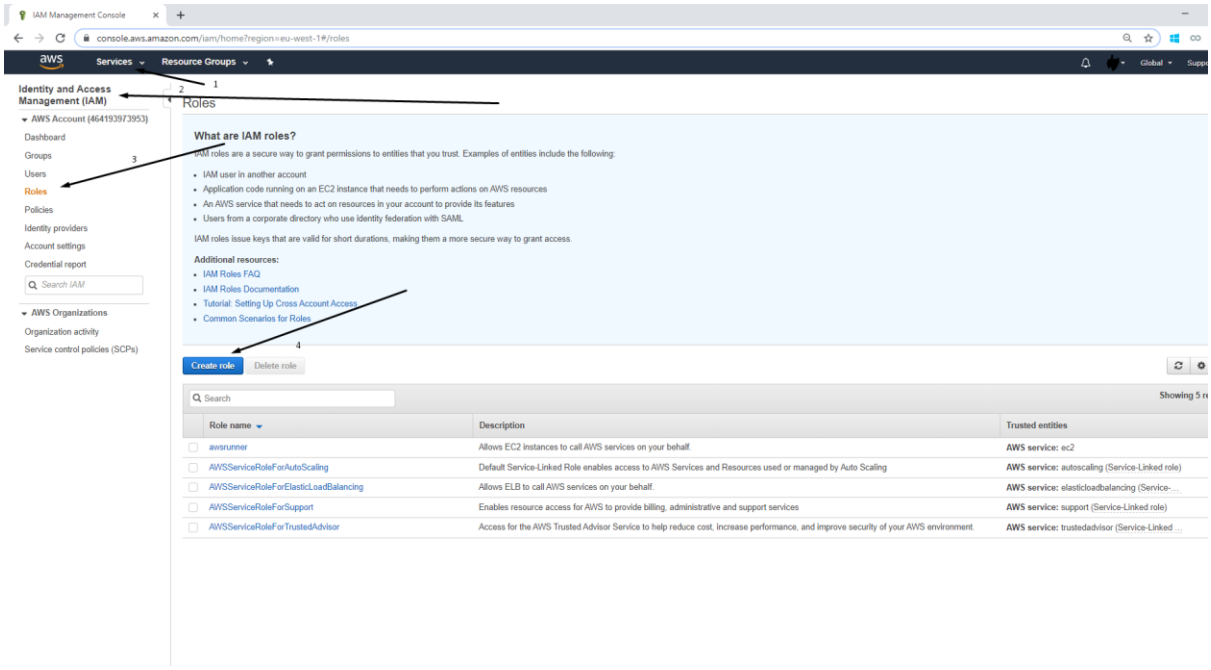
Create role awsrunner and give below permission.

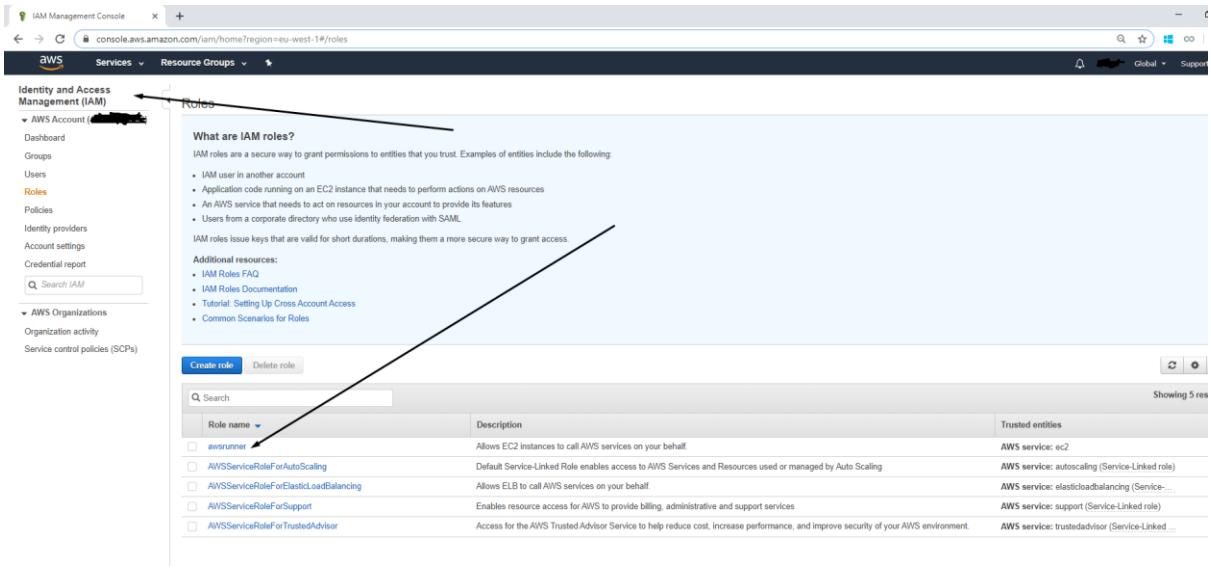
Please keep the role name as awsrunner only else code will not work.

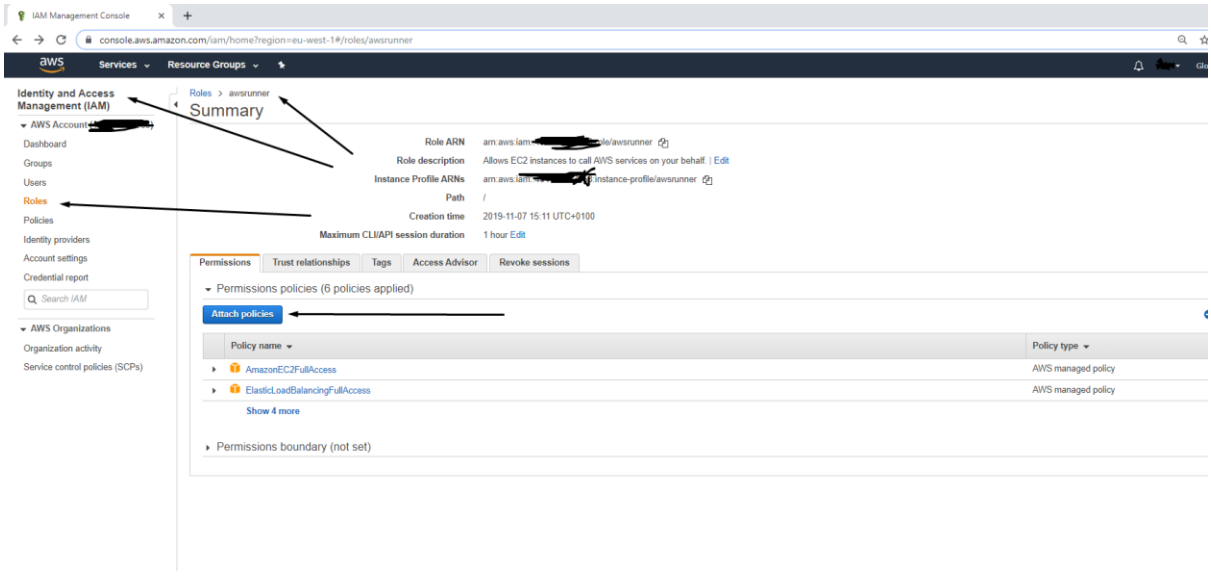
Replace the role arn and aws access and secret key in code.

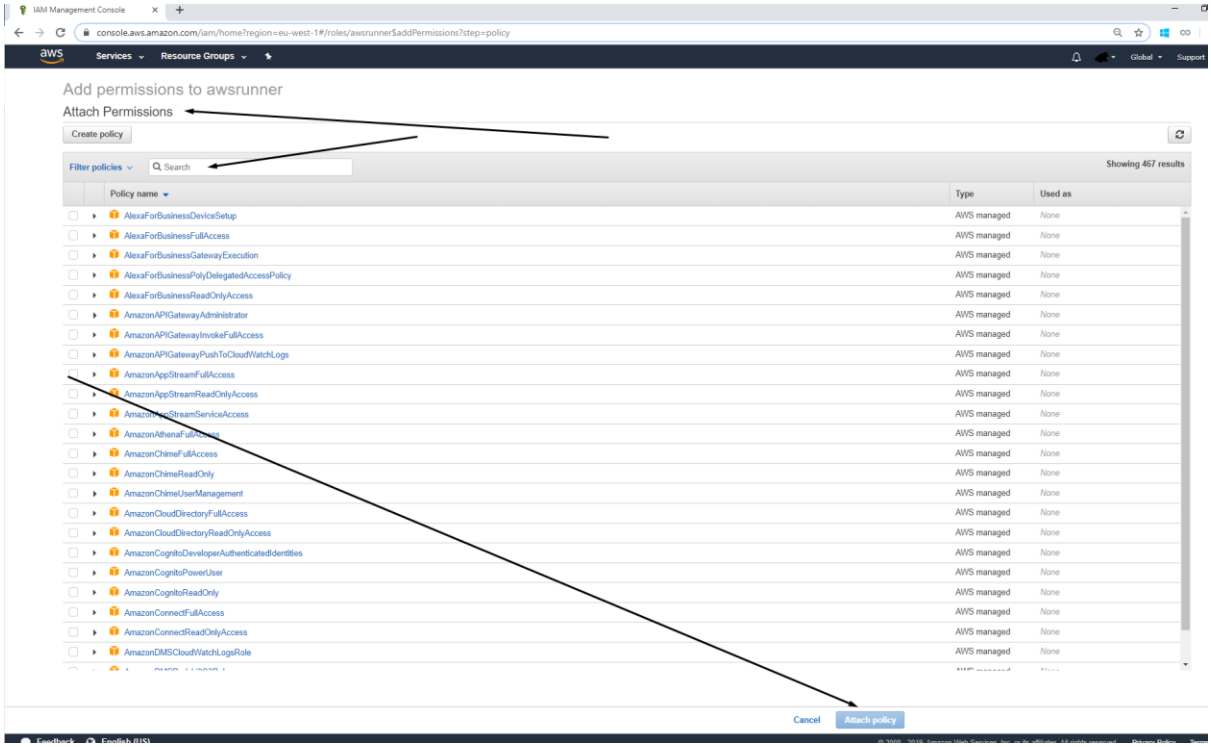
To attach permission go to IAM -> Roles -> click on awsrunner -> Attach policies -> In search option search all below 6 policies and attach it to role awsrunner.

AmazonEC2FullAccess
ElasticLoadBalancingFullAccess
AmazonS3FullAccess
AdministratorAccess
AmazonVPCFullAccess
AWSKeyManagementServicePowerUser


IAM Management Console
console.aws.amazon.com/iam/home?region=eu-west-1#/roles
Services
Resource Groups
Global
Identity and Access Management (IAM)
AWS Account (4641939739S3)
Dashboard
Groups
Users
Roles
Policies
Identity providers
Account settings
Credential report
Search IAM
AWS Organizations
Organization activity
Service control policies (SCPs)
1
2
3
4
Roles
What are IAM roles?
IAM roles are a secure way to grant permissions to entities that you trust. Examples of entities include the following:
IAM user in another account
Application code running on an EC2 instance that needs to perform actions on AWS resources
An AWS service that needs to act on resources in your account to provide its features
Users from a corporate directory who use identity federation with SAML
IAM roles issue keys that are valid for short durations, making them a more secure way to grant access.
Additional resources:
IAM Roles FAQ
IAM Roles Documentation
Tutorial: Setting Up Cross Account Access
Common Scenarios for Roles
Create role
Delete role
Search
Role name
Description
Trusted entities
awsrunner
Allows EC2 instances to call AWS services on your behalf.
AWS service: ec2
AWSServiceRoleForAutoScaling
Default Service-Linked Role enables access to AWS Services and Resources used or managed by Auto Scaling
AWS service: autoscaling (Service-Linked role)
AWSServiceRoleForElasticLoadBalancing
Allows ELB to call AWS services on your behalf.
AWS service: elasticloadbalancing (Service-...
AWSServiceRoleForSupport
Enables resource access for AWS to provide billing, administrative and support services
AWS service: support (Service-Linked role)
AWSServiceRoleForTrustedAdvisor
Access for the AWS Trusted Advisor Service to help reduce cost, increase performance, and improve security of your AWS environment.
AWS service: trustedadvisor (Service-Linked ...


IAM Management Console
console.aws.amazon.com/iam/home?region=eu-west-1#/roles
Services
Resource Groups
Global
Identity and Access Management (IAM)
Dashboard
Groups
Users
Roles
Policies
Identity providers
Account settings
Credential report
Search IAM
AWS Organizations
Organization activity
Service control policies (SCPs)
Roles
What are IAM roles?
IAM roles are a secure way to grant permissions to entities that you trust. Examples of entities include the following:
IAM user in another account
Application code running on an EC2 instance that needs to perform actions on AWS resources
An AWS service that needs to act on resources in your account to provide its features
Users from a corporate directory who use identity federation with SAML
IAM roles issue keys that are valid for short durations, making them a more secure way to grant access.
Additional resources:
IAM Roles FAQ
IAM Roles Documentation
Tutorial: Setting Up Cross Account Access
Common Scenarios for Roles
Create role
Delete role
Search
Role name
Description
Trusted entities
awsrunner
Allows EC2 instances to call AWS services on your behalf.
AWS service: ec2
AWSServiceRoleForAutoScaling
Default Service-Linked Role enables access to AWS Services and Resources used or managed by Auto Scaling
AWS service: autoscaling (Service-Linked role)
AWSServiceRoleForElasticLoadBalancing
Allows ELB to call AWS services on your behalf.
AWS service: elasticloadbalancing (Service-...
AWSServiceRoleForSupport
Enables resource access for AWS to provide billing, administrative and support services
AWS service: support (Service-Linked role)
AWSServiceRoleForTrustedAdvisor
Access for the AWS Trusted Advisor Service to help reduce cost, increase performance, and improve security of your AWS environment.
AWS service: trustedadvisor (Service-Linked ...


IAM Management Console
console.aws.amazon.com/iam/home?region=eu-west-1#/roles/awsrunner
Services
Resource Groups
Identity and Access Management (IAM)
Dashboard
Groups
Users
Roles
Policies
Identity providers
Account settings
Credential report
Search IAM
AWS Organizations
Organization activity
Service control policies (SCPs)
Roles > awsrunner
Summary
Role ARN
Role description
Allows EC2 instances to call AWS services on your behalf. | Edit
Instance Profile ARNs
Path
Creation time
2019-11-07 15:11 UTC+0100
Maximum CLI/API session duration
1 hour Edit
Permissions
Trust relationships
Tags
Access Advisor
Revoke sessions
Permissions policies (6 policies applied)
Attach policies
Policy name
Policy type
AmazonEC2FullAccess
AWS managed policy
ElasticLoadBalancingFullAccess
AWS managed policy
Show 4 more
Permissions boundary (not set)


IAM Management Console
console.aws.amazon.com/iam/home?region=eu-west-1#/roles/awsrunner$addPermissions?step=policy
Services
Resource Groups
Add permissions to awsrunner
Attach Permissions
Create policy
Filter policies
Search
Showing 467 results
Policy name
Type
Used as
AlexaForBusinessDeviceSetup
AlexaForBusinessFullAccess
AlexaForBusinessGatewayExecution
AlexaForBusinessPolyDelegatedAccessPolicy
AlexaForBusinessReadOnlyAccess
AmazonAPIGatewayAdministrator
AmazonAPIGatewayInvokeFullAccess
AmazonAPIGatewayPushToCloudWatchLogs
AmazonAppStreamFullAccess
AmazonAppStreamReadOnlyAccess
AmazonAppStreamServiceAccess
AmazonAthenaFullAccess
AmazonChimeFullAccess
AmazonChimeReadOnly
AmazonChimeUserManagement
AmazonCloudDirectoryFullAccess
AmazonCloudDirectoryReadOnlyAccess
AmazonCognitoDeveloperAuthenticatedIdentities
AmazonCognitoPowerUser
AmazonCognitoReadOnly
AmazonConnectFullAccess
AmazonConnectReadOnlyAccess
AmazonDMSCloudWatchLogsRole
AWS managed
None
Cancel
Attach policy

2.2.7 update MY_UNIQUE_BUCKET

Update MY_UNIQUE_BUCKET in state_pipeline_state.sh.

Go to Intellije where you have imported the project and now open the file under scripts -> setup_pipeline_state.sh

In file setup_pipeline_state.sh change the value of variable

MY_UNIQUE_BUCKT to your name.

Why are we doing this step?

Because every single s3 bucket in AWS accounts is unique. We can not create same bucket names in different accounts as well. So to differentiate between the same buckets we are attaching a variable to bucket name. Remove the text devops-101 and put your name over there.

For e.g. your name is john, then change the line in file setup_pipeline_state.sh as below.

```
export MY_UNIQUE_BUCKET=john
```

This way we are differentiating between all same names for the s3 buckets.

Also Open the file backend_dev.tfvars file under folder terraform -> dev-app

Change the below lines with your unique name.

```
bucket     = "dev-app-tf-state-dev-devops-101"
kms_key_id = "alias/dev-app-tf-state-dev-devops-101"
```

For e.g like this.

```
bucket     = "dev-app-tf-state-dev-john"
kms_key_id = "alias/dev-app-tf-state-dev-john"
```

Save both files.

Make sure in both the files you use same and exact name. Else code will not work.

2.2.8 Create a s3 bucket for logs

Create a s3 bucket for logs and attach a policy

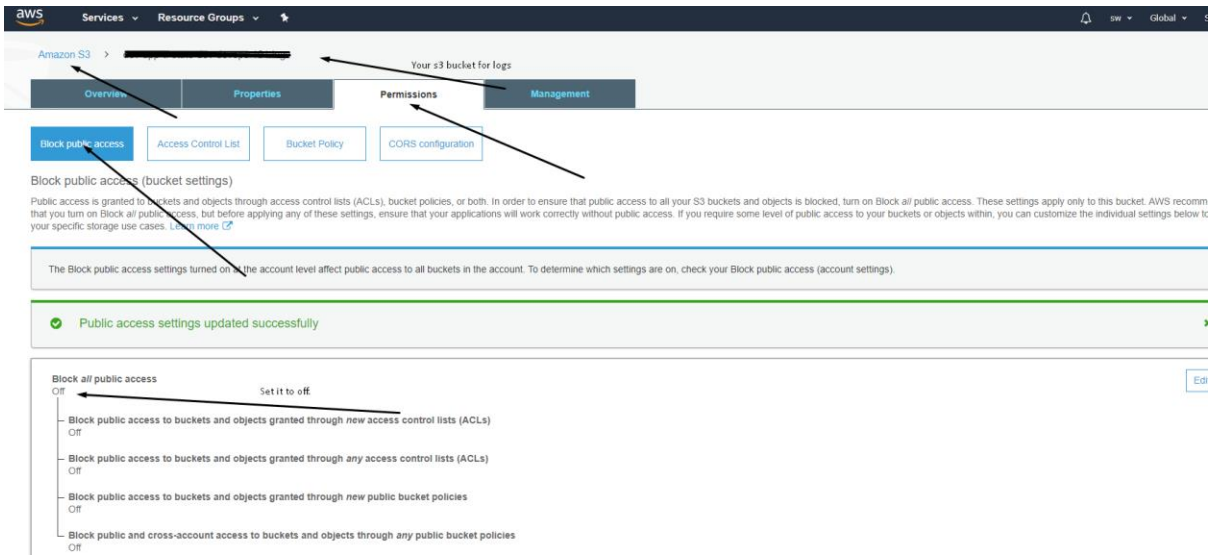
Create a S3 bucket for logs in the same region Ireland (eu-west-1)

Keep the name format as dev-app-tf-state-dev-MY_UNIQUE_BUCKET-logs.

Replace the MY_UNIQUE_BUCKET with the name you have given in setup_pipeline_state.sh file.

For e.g. John.

Make Block public access to OFF.



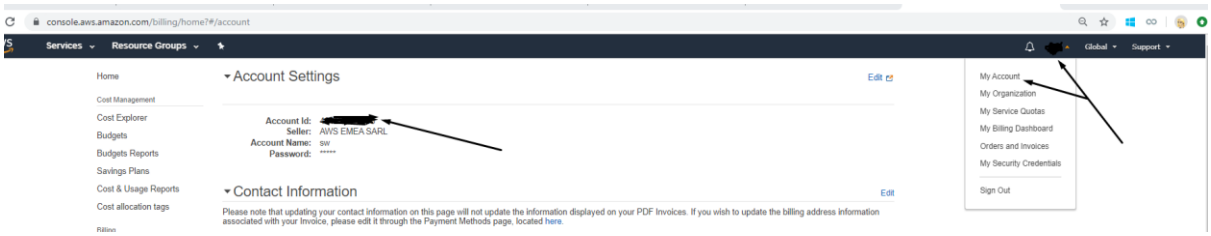
Attach the below policy for your bucket.

```
{
    "Version": "2012-10-17",
    "Id": "AWSConsole-AccessLogs-Policy-1572879529428",
    "Statement": [
        {
            "Sid": "AWSConsoleStmt-1572879529428",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::156460612806:root"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::dev-app-tf-state-dev-devops-101-logs/prefix/AWSLogs/<Change_AC_id_here>/*"
        }
    ]
}
```

Change the account id here.

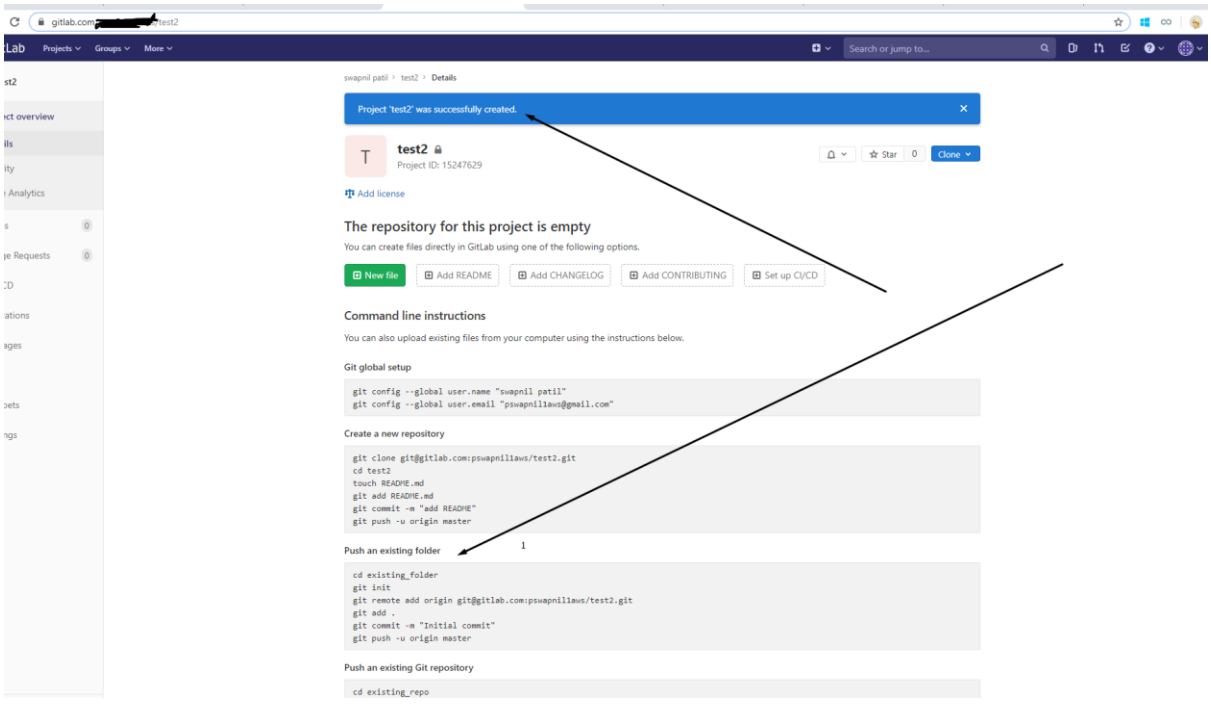
Find your AC id her :

In your AWS account -> Top right corner -> Your account name -> My account -> Account id



2.2.9 Commit and push the code to master branch

Now follow the instructions written under Push an existing folder



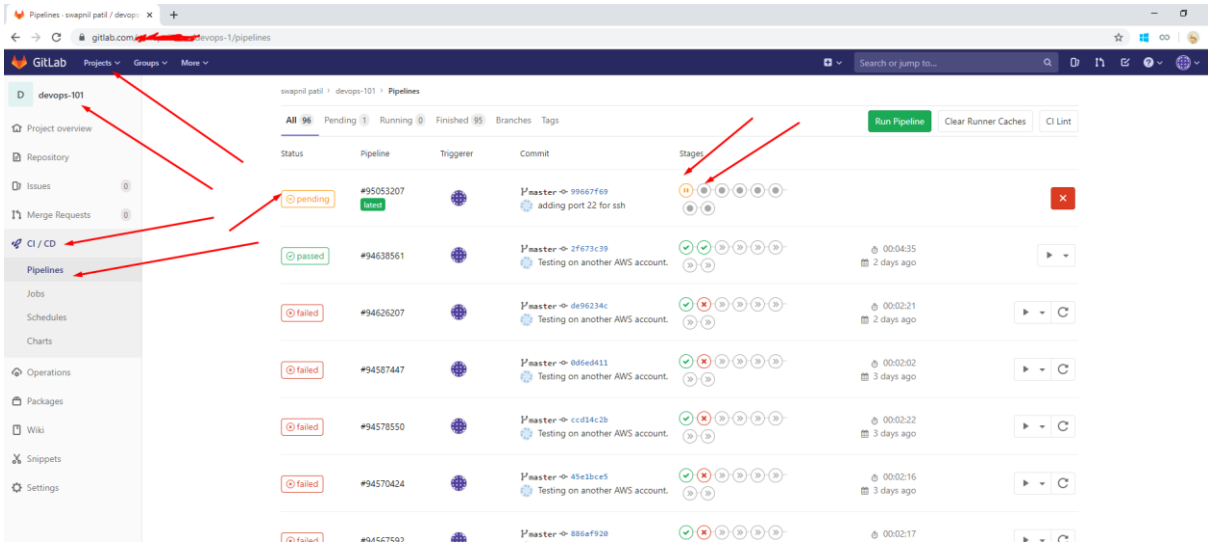
Section 3 : Output

3.1 : Gitlab Pipeline.

Now go to your gitlab account -> Projects -> Your Project Name ->

CI / CD -> Pipelines -> Click on the pipelines.

In stages you will see the logs. (Refer image below)



3.2 AWS EC2 :

Go to your AWS Account

Go to Services -> EC2 -> Instances -> You will see an instance name with dev-app-dev-master-dev-master -> select that instance -> Below

that you will see a description -> Under description you will see field Public DNS (IPv4)-> Copy the value

3.3 Sample Website :

PLEASE PASTE AFTER 15 MINUTES once you trigger the pipeline.

And with only http.

E.g. http://<Public DNS(IPv4)>

As it takes time to launch the ec2 instance and install required software's.

You will be able to see sample web site.

Section 4: Cost and Support

4.1 Cost of the Project :

In the attached folder named DevOps-101, you can find the sample Code of the project.

If you are looking for the complete source code of the below is the cost sheet.

Complete DevOps 101 Project

- Including one hour Skype support if project is not working.

Project Name	Cost		
Project DevOps-101	USD : 55 Dollars.	EUR : 50 Euros.	INR : 4000 Rs.

4.2 Support for the deployment :

If according to given instructions you are not able to deploy the Project, Please send us a screen shot and we will guide you (Only after purchase of project).

If the issue still persist we will take one hour skype session and we will make this project work on your local machine and AWS/Gitlab account.

4.3 Questions or Feedback:

If you have any Questions or feedback/suggestions, then feel free to email at projectdevops101@gmail.com