Lab Documentation

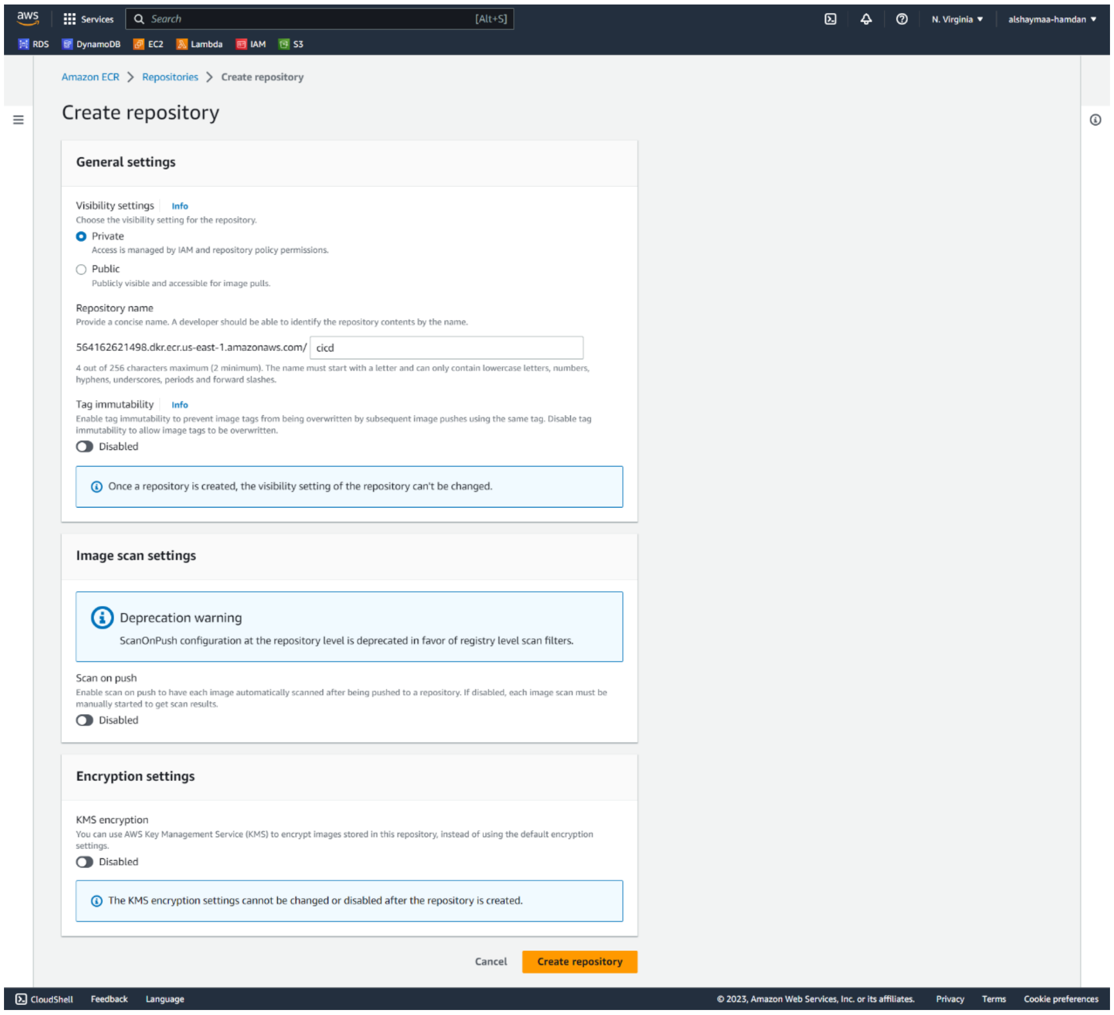
Pushing image to AWS ECR and run it using docker in AWS EC2 instance.

Step 1: Create AWS ECR

Amazon Elastic Container Registry (ECR) is an AWS managed container image registry service that is secure, scalable, and reliable.

Amazon ECR supports private repositories with resource-based permissions using AWS IAM.

- Open AWS Console, then go to "ECR" service.
- Click on "Create Repository"
- On the Repository settings page fill in its name and make it private.


Services
Search [Alt+S]
N. Virginia
alshaymaa-hamdan
RDS DynamoDB EC2 Lambda IAM S3
Amazon ECR > Repositories > Create repository
Cancel
Create repository
CloudShell Feedback Language
© 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Create repository

General settings

Visibility settings Info
Choose the visibility setting for the repository.
[✓ Private] Access is managed by IAM and repository policy permissions.
Public Publicly visible and accessible for image pulls.

Repository name
Provide a concise name. A developer should be able to identify the repository contents by the name.
564162621498.dkr.ecr.us-east-1.amazonaws.com/ cicd
4 out of 256 characters maximum (2 minimum). The name must start with a letter and can only contain lowercase letters, numbers, hyphens, underscores, periods and forward slashes.

Tag immutability Info
Enable tag immutability to prevent image tags from being overwritten by subsequent image pushes using the same tag. Disable tag immutability to allow image tags to be overwritten.
Disabled

Once a repository is created, the visibility setting of the repository can't be changed.

Image scan settings

Deprecation warning
ScanOnPush configuration at the repository level is deprecated in favor of registry level scan filters.

Scan on push
Enable scan on push to have each image automatically scanned after being pushed to a repository. If disabled, each image scan must be manually started to get scan results.
Disabled

Encryption settings

KMS encryption
You can use AWS Key Management Service (KMS) to encrypt images stored in this repository, instead of using the default encryption settings.
Disabled

The KMS encryption settings cannot be changed or disabled after the repository is created.

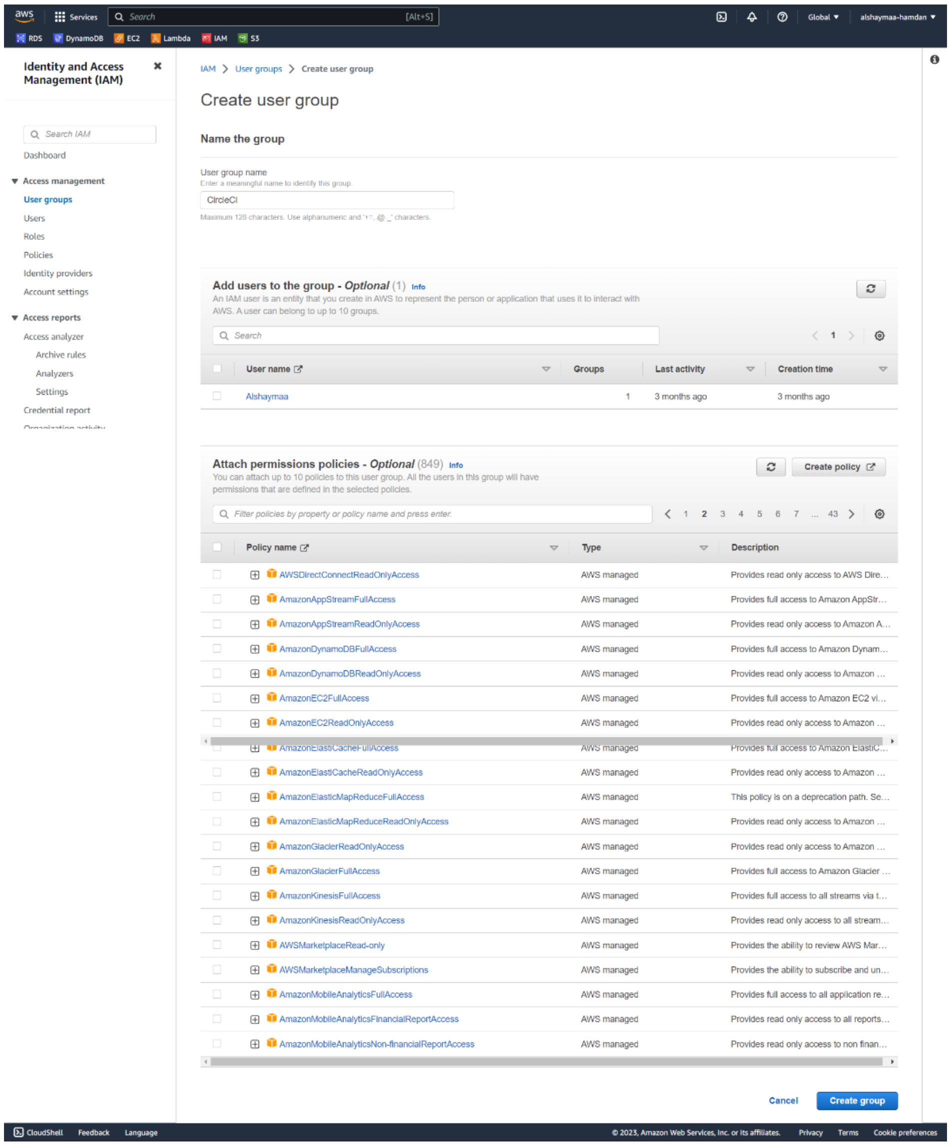
- Click "Create repository"

Step 2: Give an IAM User access to our ECR

AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources. With IAM, you can centrally manage permissions that control which AWS resources users can access. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources.

By default, users and roles don't have permission to create or modify Amazon ECR resources.

- Create IAM User Group
  IAM user group is a collection of IAM users. User groups let you specify permissions for multiple users, which can make it easier to manage the permissions for those users.
  - Go to "IAM", then click on "User groups", then "Create group".
  - Give the group a name, then click on "Create group".



Create user group

Name the group

User group name
Enter a meaningful name to identify this group.
CircleCI
Maximum 128 characters. Use alphanumeric and '+=,.@-_' characters.

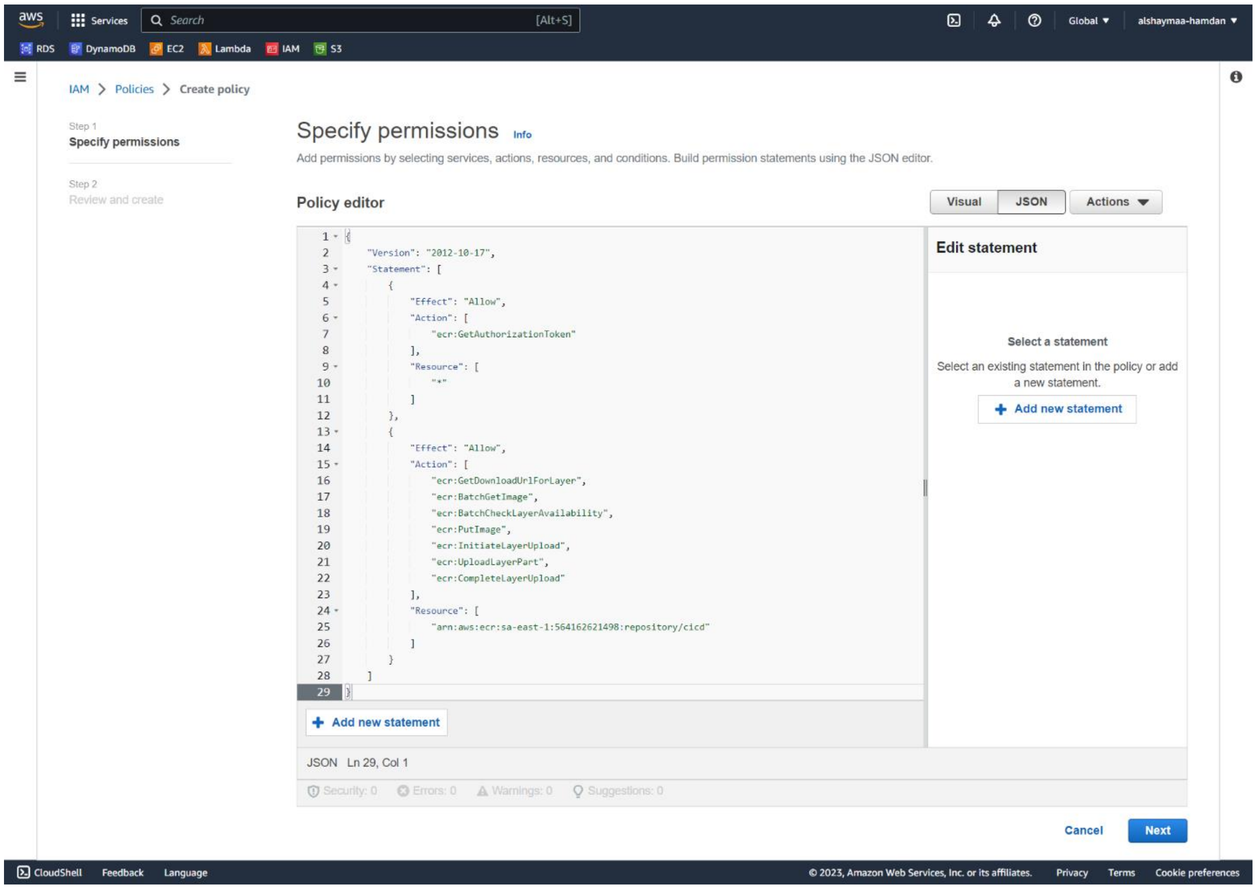
Add users to the group - *Optional* (1) Info
An IAM user is an entity that you create in AWS to represent the person or application that uses it to interact with AWS. A user can belong to up to 10 groups.

User name	Groups	Last activity	Creation time
Alshaymaa	1	3 months ago	3 months ago

Attach permissions policies - *Optional* (849) Info
You can attach up to 10 policies to this user group. All the users in this group will have permissions that are defined in the selected policies.

Policy name	Type	Description
AWSDirectConnectReadOnlyAccess	AWS managed	Provides read only access to AWS Dire…
AmazonAppStreamFullAccess	AWS managed	Provides full access to Amazon AppStr…
AmazonAppStreamReadOnlyAccess	AWS managed	Provides read only access to Amazon A…
AmazonDynamoDBFullAccess	AWS managed	Provides full access to Amazon Dynam…
AmazonDynamoDBReadOnlyAccess	AWS managed	Provides read only access to Amazon …
AmazonEC2FullAccess	AWS managed	Provides full access to Amazon EC2 vi…
AmazonEC2ReadOnlyAccess	AWS managed	Provides read only access to Amazon …
AmazonElastiCacheFullAccess	AWS managed	Provides full access to Amazon ElastiC…
AmazonElastiCacheReadOnlyAccess	AWS managed	Provides read only access to Amazon …
AmazonElasticMapReduceFullAccess	AWS managed	This policy is on a deprecation path. Se…
AmazonElasticMapReduceReadOnlyAccess	AWS managed	Provides read only access to Amazon …
AmazonGlacierReadOnlyAccess	AWS managed	Provides read only access to Amazon …
AmazonGlacierFullAccess	AWS managed	Provides full access to Amazon Glacier …
AmazonKinesisFullAccess	AWS managed	Provides full access to all streams via t…
AmazonKinesisReadOnlyAccess	AWS managed	Provides read only access to all stream…
AWSMarketplaceRead-only	AWS managed	Provides the ability to review AWS Mar…
AWSMarketplaceManageSubscriptions	AWS managed	Provides the ability to subscribe and un…
AmazonMobileAnalyticsFullAccess	AWS managed	Provides full access to all application re…
AmazonMobileAnalyticsFinancialReportAccess	AWS managed	Provides read only access to all reports…
AmazonMobileAnalyticsNon-financialReportAccess	AWS managed	Provides read only access to non finan…

- Create "Policy" to attach it to "CircleCI" group.
  IAM policies define permissions for an action regardless of the method that you use to perform the operation.
  - Click "Create policy", then add the permissions in JSON format



IAM > Policies > Create policy

Step 1
Specify permissions

Step 2
Review and create

Specify permissions

Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.

Policy editor

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:GetAuthorizationToken"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage",
                "ecr:BatchCheckLayerAvailability",
                "ecr:PutImage",
                "ecr:InitiateLayerUpload",
                "ecr:UploadLayerPart",
                "ecr:CompleteLayerUpload"
            ],
            "Resource": [
                "arn:aws:ecr:sa-east-1:564162621498:repository/cicd"
            ]
        }
    ]
}
```

JSON Ln 29, Col 1

Security: 0 Errors: 0 Warnings: 0 Suggestions: 0

Edit statement

Select a statement

Select an existing statement in the policy or add a new statement.

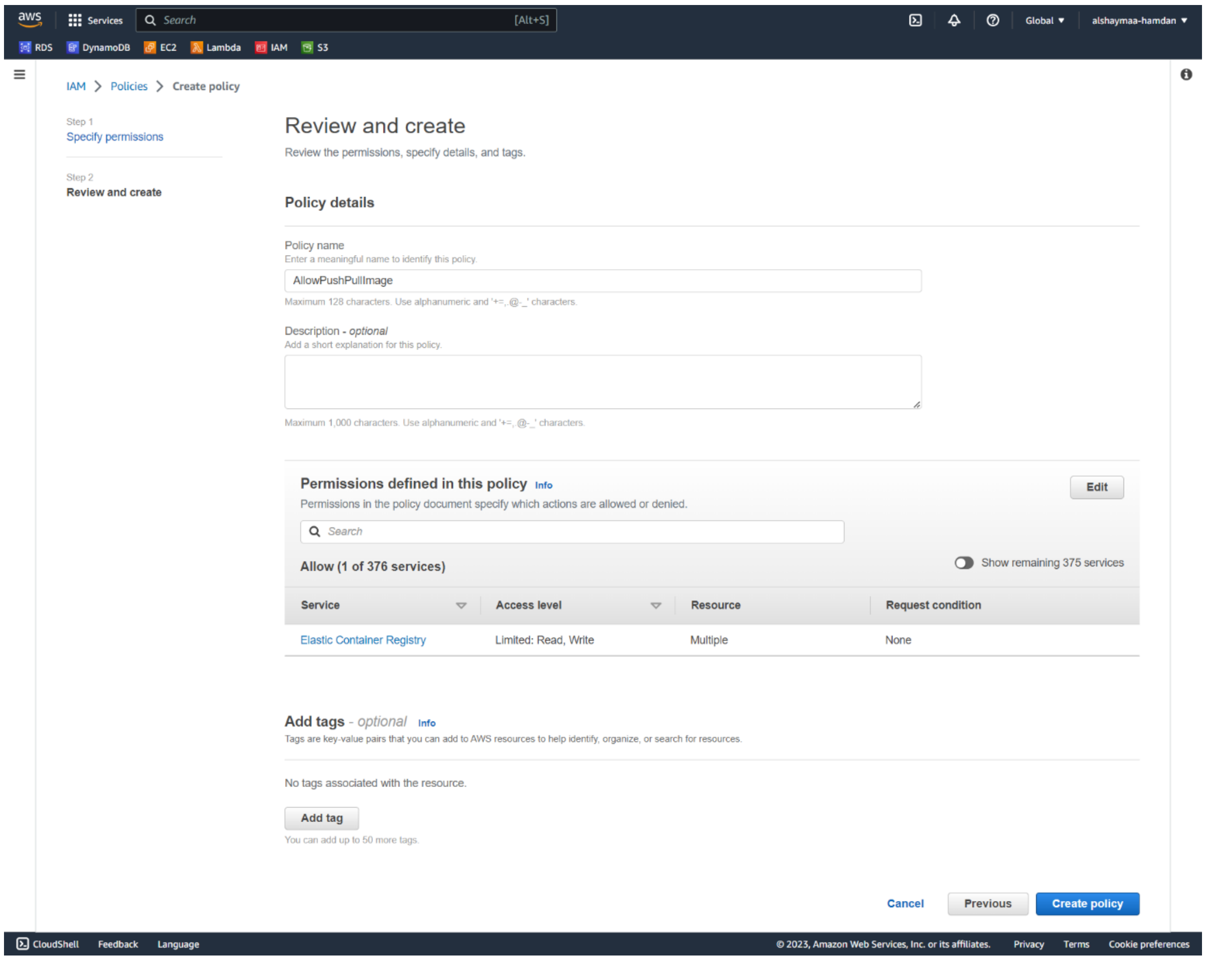
© 2023, Amazon Web Services, Inc. or its affiliates.

- Permissions:

{ "Version": "2012-10-17", "Statement": [{ "Effect": "Allow", "Action": ["ecr:GetAuthorizationToken"], "Resource": ["*"] },	Amazon ECR requires that users have permission to make calls to the ecr:GetAuthorizationToken API through an IAM policy before they can authenticate to a registry and push or pull any images from any Amazon ECR repository.
{ "Effect": "Allow", "Action": ["ecr:GetDownloadUrlForLayer", "ecr:BatchGetImage", "ecr:BatchCheckLayerAvailability",	grant a user in your AWS account access to "cicd" ECR repository. Also, allow the user to push, pull, and list images

"ecr:PutImage", "ecr:InitiateLayerUpload", "ecr:UploadLayerPart", "ecr:CompleteLayerUpload"], "Resource": ["arn:aws:ecr:us-east-1:564162621498:repository/cicd"] }] }	(arn:aws:ecr:{region}:{account_id}:repository/{repo_name})

- Name the policy, then click "Create policy"



Review and create

Review the permissions, specify details, and tags.

Policy details

Policy name
Enter a meaningful name to identify this policy.
AllowPushPullImage
Maximum 128 characters. Use alphanumeric and '+=,.@-_' characters.

Description - *optional*
Add a short explanation for this policy.
Maximum 1,000 characters. Use alphanumeric and '+=,.@-_' characters.

Permissions defined in this policy Info
Permissions in the policy document specify which actions are allowed or denied.

Allow (1 of 376 services) — Show remaining 375 services

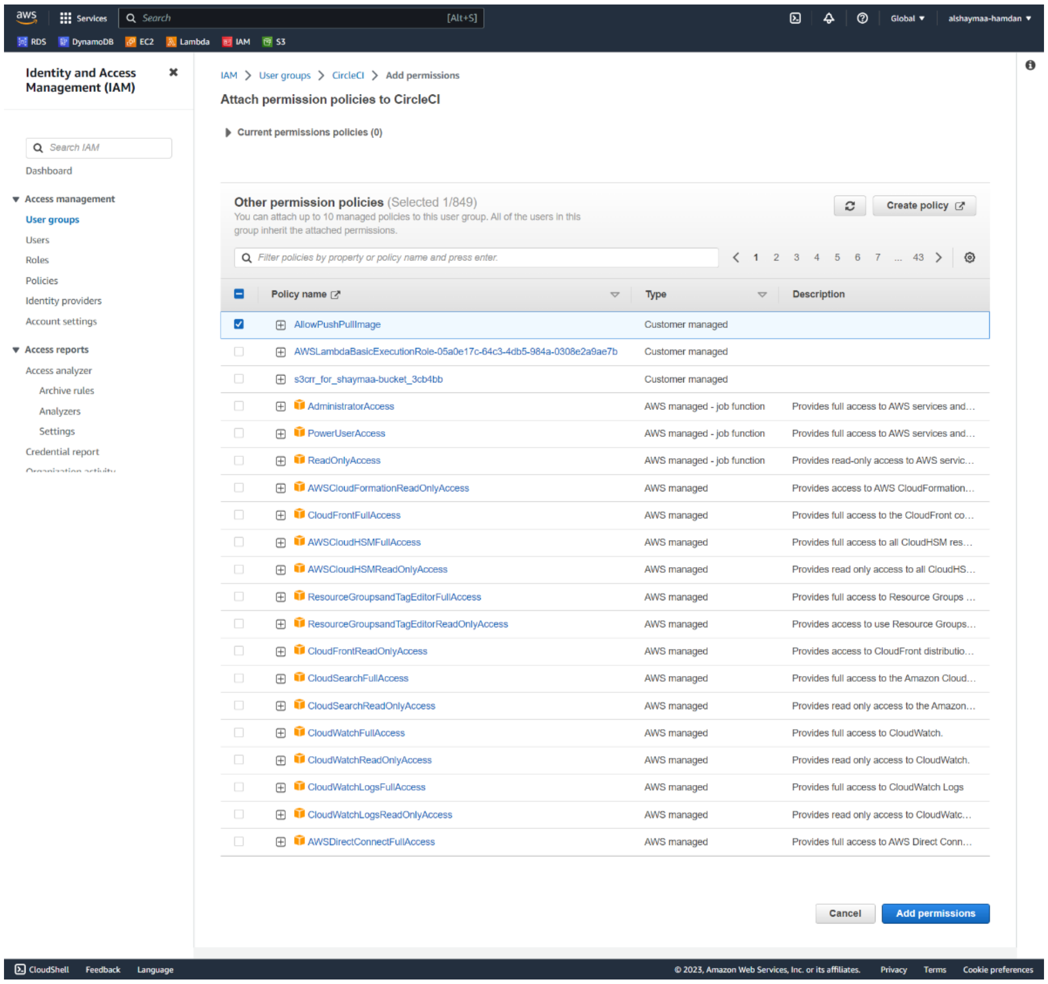
Service	Access level	Resource	Request condition
Elastic Container Registry	Limited: Read, Write	Multiple	None

Add tags - *optional* Info
Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.

No tags associated with the resource.

Add tag
You can add up to 50 more tags.

- Attach "AllowPushPullImage" policy to "CircleCI" group.
  - Go to "User groups", then choose "CircleCI" group, click on "Permissions", "Add Permissions", "Attach Policy", then choose "AllowPushPullImage" Policy.



IAM > User groups > CircleCI > Add permissions

Attach permission policies to CircleCI

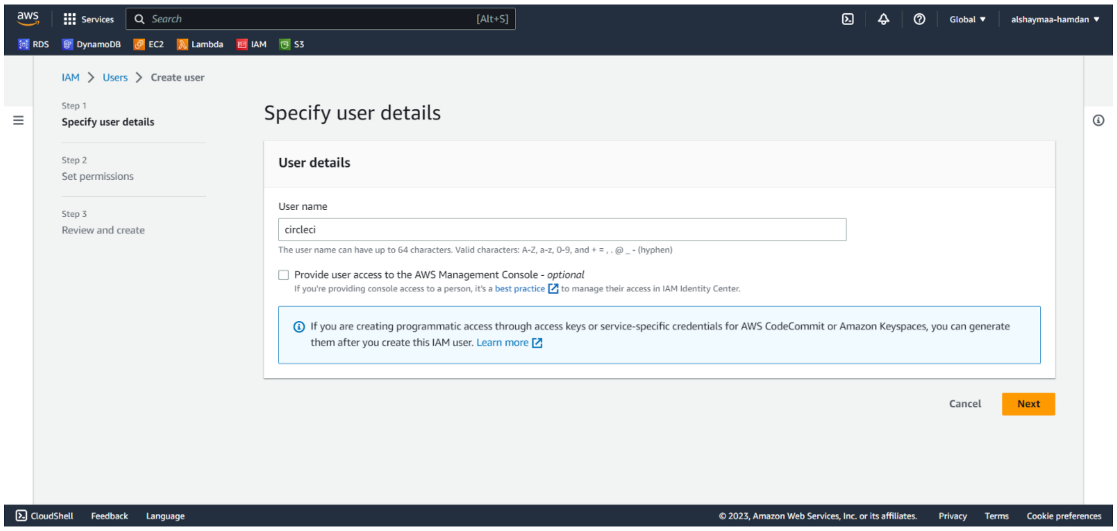
Current permissions policies (0)

Other permission policies (Selected 1/849)

You can attach up to 10 managed policies to this user group. All of the users in this group inherit the attached permissions.

	Policy name	Type	Description
✓	AllowPushPullImage	Customer managed	
	AWSLambdaBasicExecutionRole-05a0e17c-64c3-4db5-984a-0308e2a9ae7b	Customer managed	
	s3crr_for_shaymaa-bucket_3cb4bb	Customer managed	
	AdministratorAccess	AWS managed - job function	Provides full access to AWS services and…
	PowerUserAccess	AWS managed - job function	Provides full access to AWS services and…
	ReadOnlyAccess	AWS managed - job function	Provides read-only access to AWS servic…
	AWSCloudFormationReadOnlyAccess	AWS managed	Provides access to AWS CloudFormation…
	CloudFrontFullAccess	AWS managed	Provides full access to the CloudFront co…
	AWSCloudHSMFullAccess	AWS managed	Provides full access to all CloudHSM res…
	AWSCloudHSMReadOnlyAccess	AWS managed	Provides read only access to all CloudHS…
	ResourceGroupsandTagEditorFullAccess	AWS managed	Provides full access to Resource Groups …
	ResourceGroupsandTagEditorReadOnlyAccess	AWS managed	Provides access to use Resource Groups…
	CloudFrontReadOnlyAccess	AWS managed	Provides access to CloudFront distributio…
	CloudSearchFullAccess	AWS managed	Provides full access to the Amazon Cloud…
	CloudSearchReadOnlyAccess	AWS managed	Provides read only access to the Amazon…
	CloudWatchFullAccess	AWS managed	Provides full access to CloudWatch.
	CloudWatchReadOnlyAccess	AWS managed	Provides read only access to CloudWatch.
	CloudWatchLogsFullAccess	AWS managed	Provides full access to CloudWatch Logs
	CloudWatchLogsReadOnlyAccess	AWS managed	Provides read only access to CloudWatc…
	AWSDirectConnectFullAccess	AWS managed	Provides full access to AWS Direct Conn…

- Create User
  IAM policies define permissions for an action regardless of the method that you use to perform the operation.
    - Go to "Users", give the user a name then add it to "CircleCI" group.



IAM > Users > Create user

Step 1 Specify user details
Step 2 Set permissions
Step 3 Review and create

Specify user details

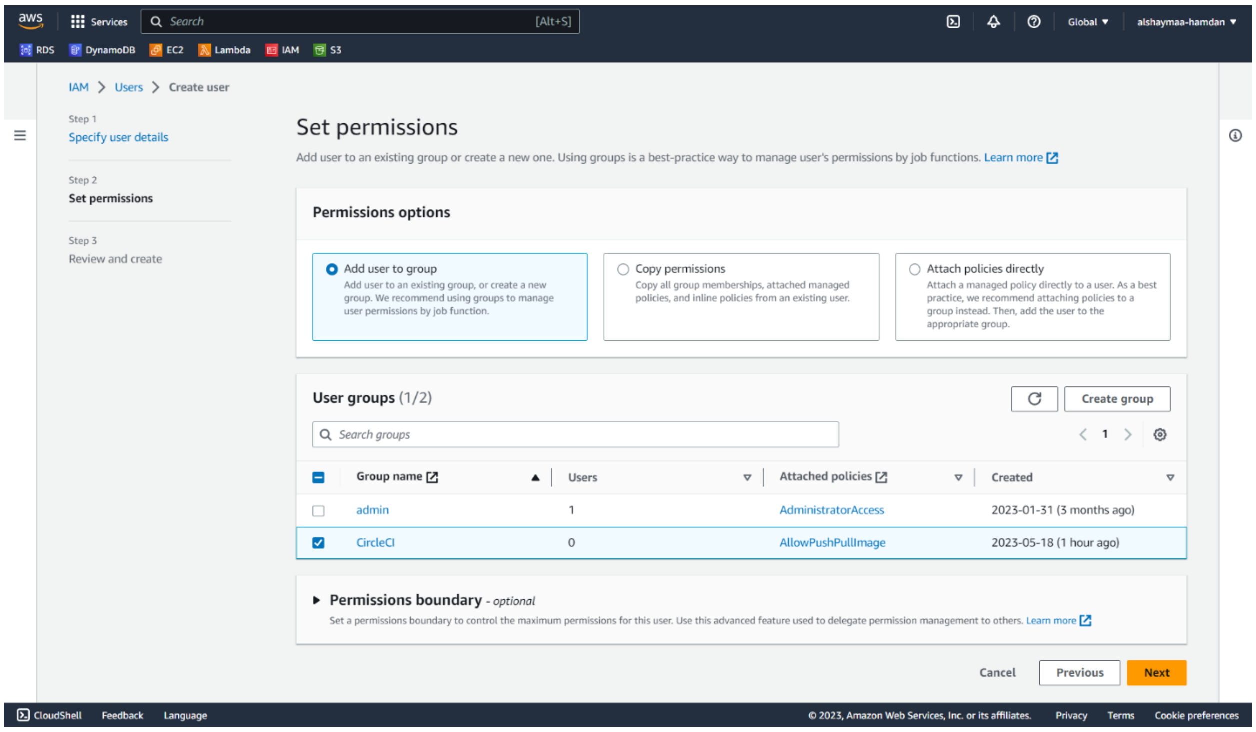
User details

User name: circleci

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , . @ _ - (hyphen)

☐ Provide user access to the AWS Management Console - *optional*
If you're providing console access to a person, it's a best practice to manage their access in IAM Identity Center.

If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. Learn more



Set permissions

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

Permissions options

- ● Add user to group — Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.
- ○ Copy permissions — Copy all group memberships, attached managed policies, and inline policies from an existing user.
- ○ Attach policies directly — Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

User groups (1/2)

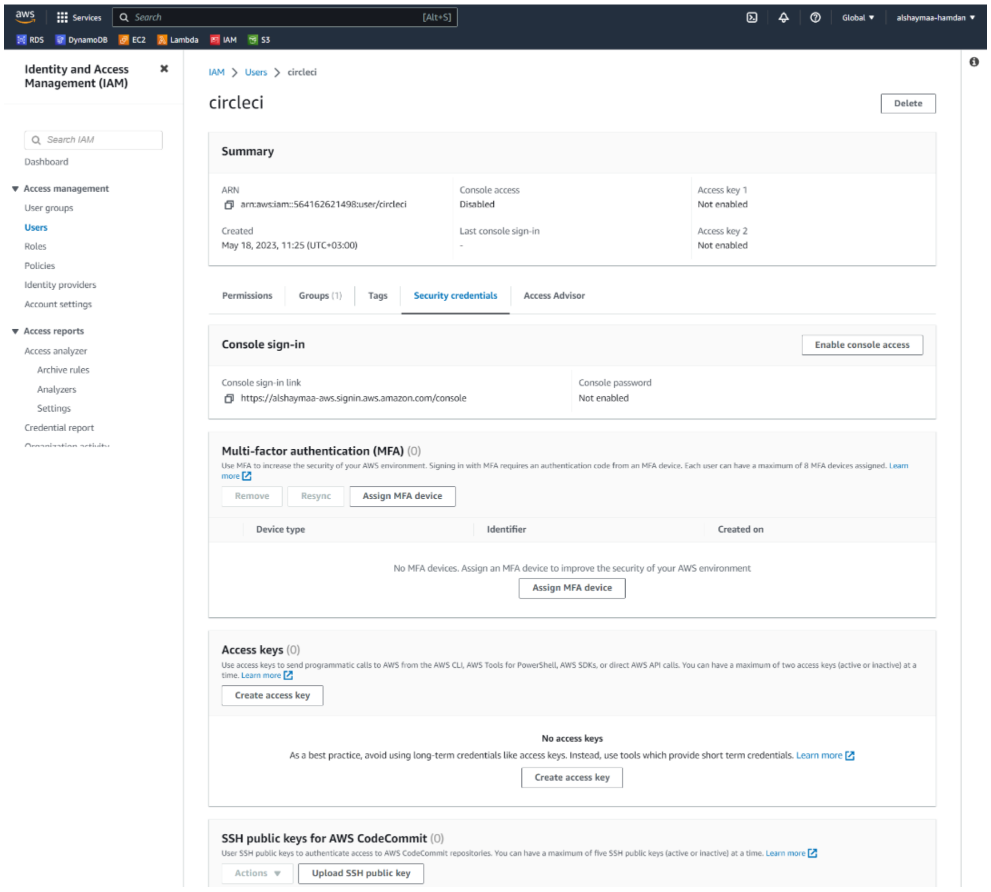
	Group name	Users	Attached policies	Created
☐	admin	1	AdministratorAccess	2023-01-31 (3 months ago)
✓	CircleCI	0	AllowPushPullImage	2023-05-18 (1 hour ago)

Permissions boundary - *optional*

Set a permissions boundary to control the maximum permissions for this user. Use this advanced feature used to delegate permission management to others. Learn more

Create group
Search groups
Cancel
Previous
Next
CloudShell Feedback Language
© 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

- Click "Create user", then click on the user you created "circleci"
- Go to "Security credentials", then create "Access Key"



circleci

Summary

ARN arn:aws:iam::564162621498:user/circleci	Console access Disabled	Access key 1 Not enabled
Created May 18, 2023, 11:25 (UTC+03:00)	Last console sign-in -	Access key 2 Not enabled

Console sign-in

Console sign-in link: https://alshaymaa-aws.signin.aws.amazon.com/console

Console password: Not enabled

Multi-factor authentication (MFA) (0)

Use MFA to increase the security of your AWS environment. Signing in with MFA requires an authentication code from an MFA device. Each user can have a maximum of 8 MFA devices assigned. Learn more

Device type	Identifier	Created on

No MFA devices. Assign an MFA device to improve the security of your AWS environment

Access keys (0)

Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. Learn more

No access keys

As a best practice, avoid using long-term credentials like access keys. Instead, use tools which provide short term credentials. Learn more

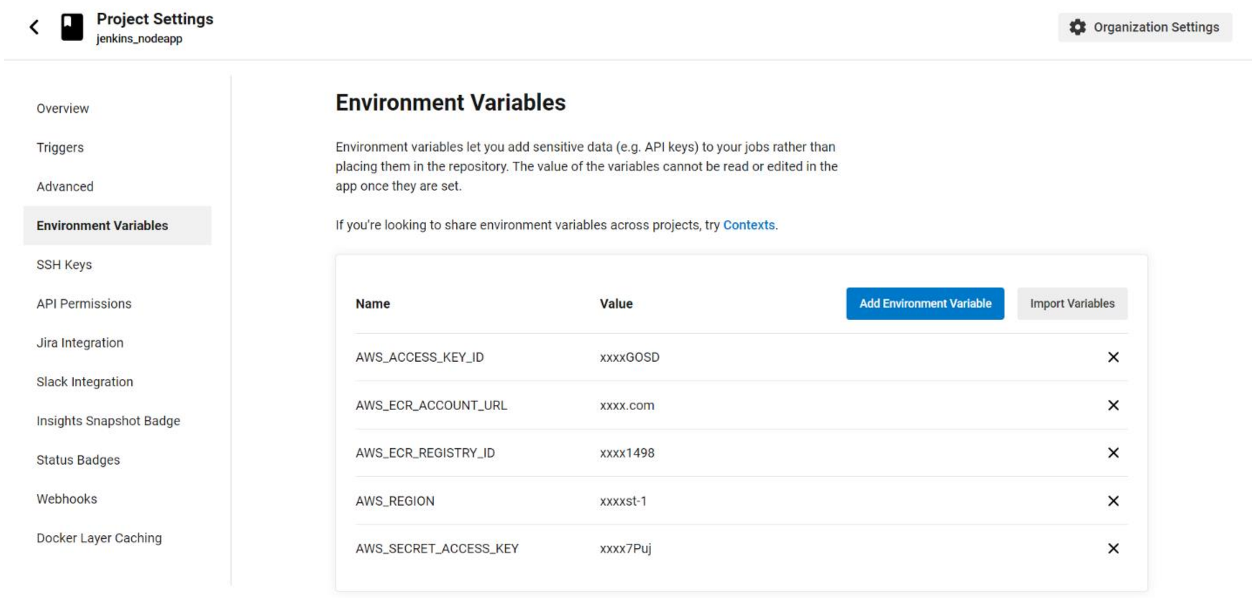
SSH public keys for AWS CodeCommit (0)

User SSH public keys to authenticate access to AWS CodeCommit repositories. You can have a maximum of five SSH public keys (active or inactive) at a time. Learn more

- Choose "Application running outside AWS"
- Don't forget to save the "Access key", click "Download .csv file"

Step 3: Push the image to ECR using CircleCI

- Choose which GitHub project you want to setup
- Add "AWS ECR" credentials to "CircleCI"
    - Create Env-var in your "Project Settings"



Environment Variables

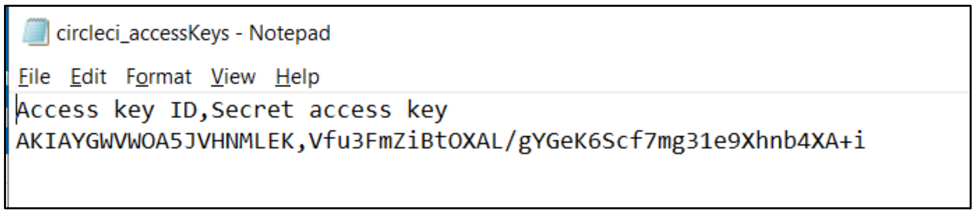
Environment variables let you add sensitive data (e.g. API keys) to your jobs rather than placing them in the repository. The value of the variables cannot be read or edited in the app once they are set.

If you're looking to share environment variables across projects, try Contexts.

Name	Value
AWS_ACCESS_KEY_ID	xxxxGOSD
AWS_ECR_ACCOUNT_URL	xxxx.com
AWS_ECR_REGISTRY_ID	xxxx1498
AWS_REGION	xxxxst-1
AWS_SECRET_ACCESS_KEY	xxxx7Puj

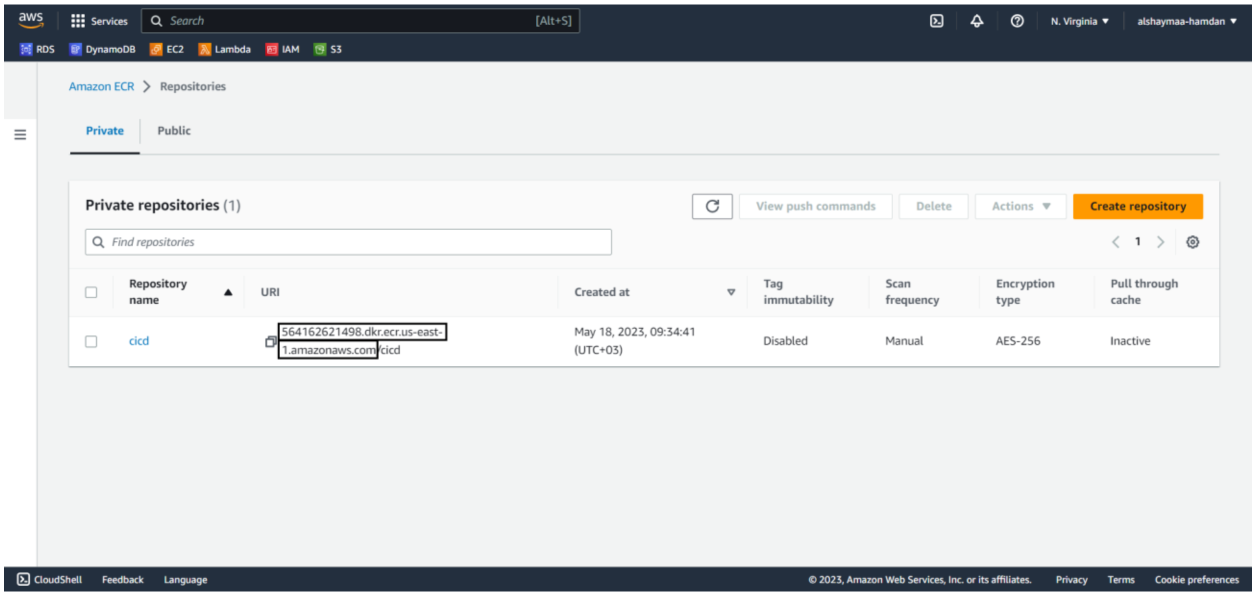
AWS_ACCESS_KEY_ID: open the access key .csv file that you saved in Step 2. copy the ID and add it to this env-var.

AWS_SECRET_ACCESS_KEY: Copy the secret access key and add it to this env-var.



```
Access key ID,Secret access key
AKIAYGWVWOA5JVHNMLEK,Vfu3FmZiBtOXAL/gYGeK6Scf7mg31e9Xhnb4XA+i
```

AWS_ECR_ACCOUNT_URL: get it from ECR "cicd" repository.



Amazon ECR > Repositories

Private | Public

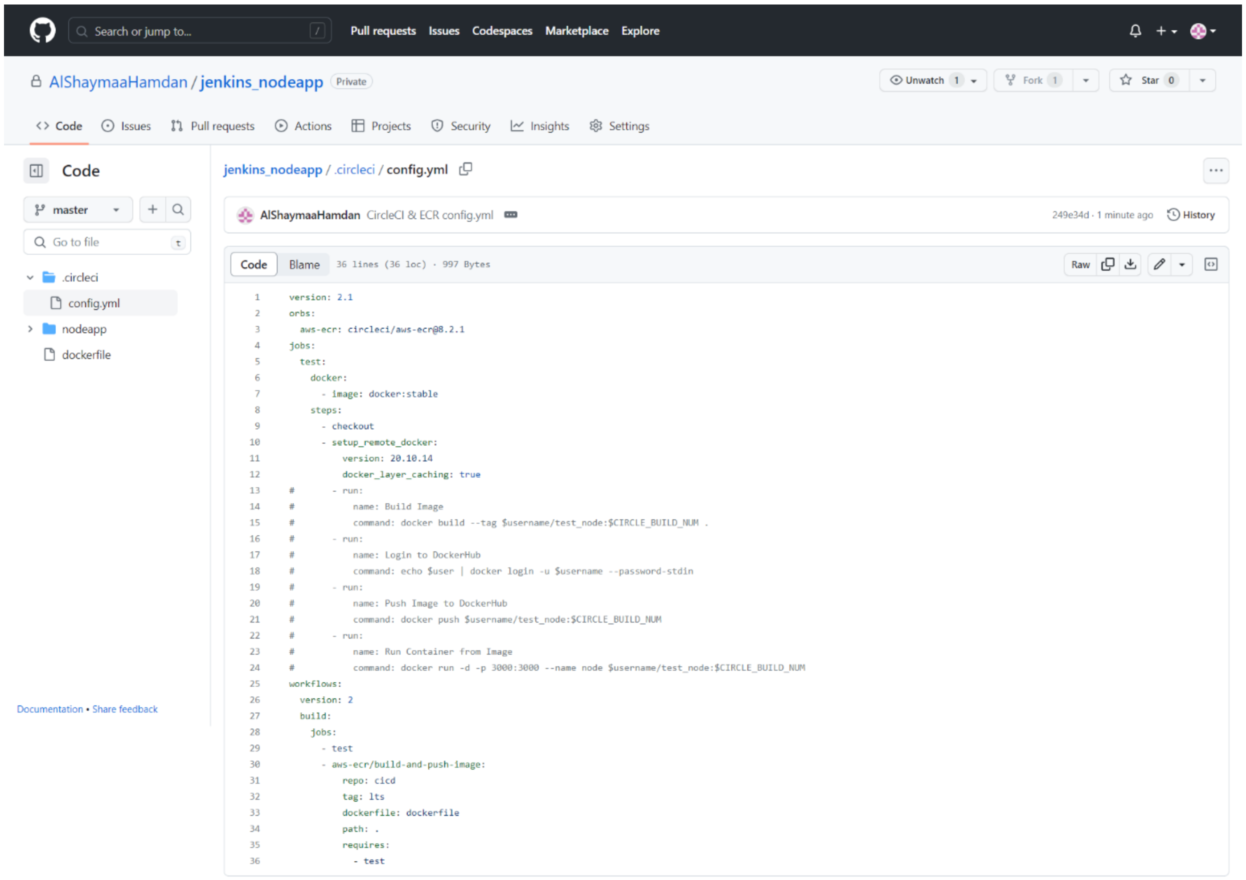
Private repositories (1)

	Repository name	URI	Created at	Tag immutability	Scan frequency	Encryption type	Pull through cache
☐	cicd	564162621498.dkr.ecr.us-east-1.amazonaws.com/cicd	May 18, 2023, 09:34:41 (UTC+03)	Disabled	Manual	AES-256	Inactive

AWS_ECR_REGISTRY_ID: The 12 digit AWS id associated with the ECR account.

AWS_REGION: AWS ECR Region, ex. us-east-1

- Create CircleCI Config file
    - On github repo add “config.yml” file in “.circleci” folder



AlShaymaaHamdan / jenkins_nodeapp

jenkins_nodeapp / .circleci / config.yml

AlShaymaaHamdan CircleCI & ECR config.yml

36 lines (36 loc) · 997 Bytes

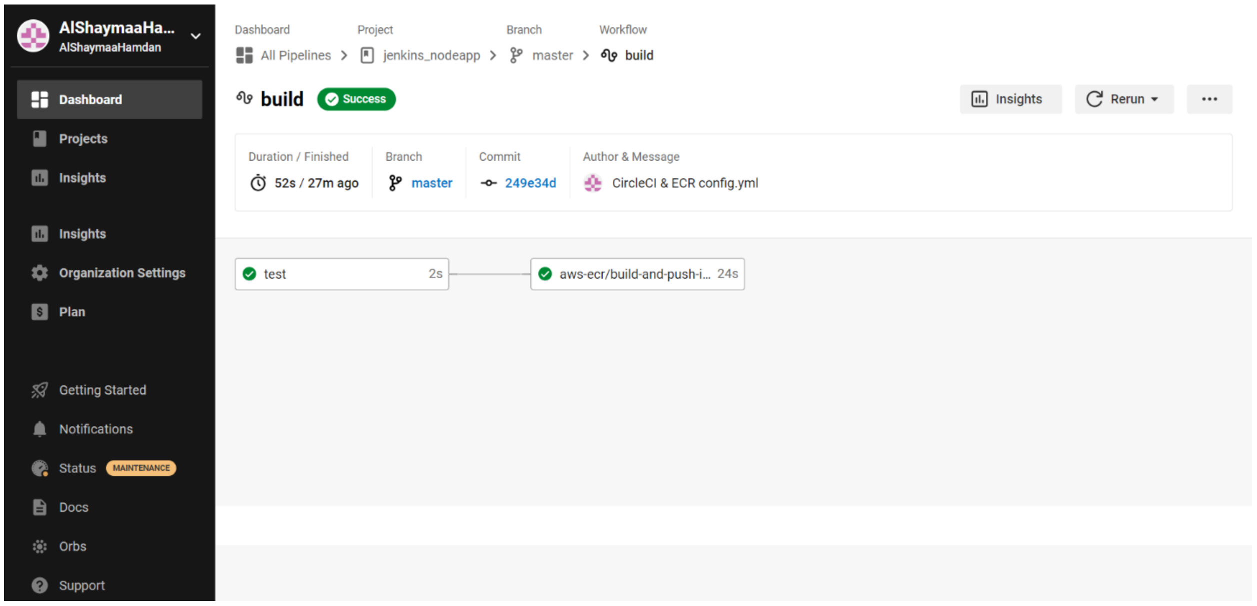
```yaml
version: 2.1
orbs:
  aws-ecr: circleci/aws-ecr@8.2.1
jobs:
  test:
    docker:
      - image: docker:stable
    steps:
      - checkout
      - setup_remote_docker:
          version: 20.10.14
          docker_layer_caching: true
#       - run:
#           name: Build Image
#           command: docker build --tag $username/test_node:$CIRCLE_BUILD_NUM .
#       - run:
#           name: Login to DockerHub
#           command: echo $user | docker login -u $username --password-stdin
#       - run:
#           name: Push Image to DockerHub
#           command: docker push $username/test_node:$CIRCLE_BUILD_NUM
#       - run:
#           name: Run Container from Image
#           command: docker run -d -p 3000:3000 --name node $username/test_node:$CIRCLE_BUILD_NUM
workflows:
  version: 2
  build:
    jobs:
      - test
      - aws-ecr/build-and-push-image:
          repo: cicd
          tag: lts
          dockerfile: dockerfile
          path: .
          requires:
            - test
```

config.yml content:

version: 2.1
orbs:
  aws-ecr: circleci/aws-ecr@8.2.1
jobs:
  test:
    docker:
      - image: docker:stable
    steps:
      - checkout
      - setup_remote_docker:
          version: 20.10.14
          docker_layer_caching: true
	CircleCI Orbs are shareable packages of CircleCI configuration you can use to simplify your builds.
	ecr orb is to Build images and push them to the Amazon Elastic Container Registry.

workflows:
  version: 2
  build:
    jobs:
      - test
      - aws-ecr/build-and-push-image:
          repo: cicd
          tag: lts
          dockerfile: dockerfile
          path: .
          requires:
            - test
	simple-build-and-push
	Log into AWS, build and push image to Amazon ECR. Requires environment variables for AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY And others just as we created before.
	
	# Name of an Amazon ECR repository
	# docker image tags to build and push
	# Name of dockerfile to use.
	# Path to the directory containing your Dockerfile and build context.

- After committing config.yml file changes a new pipeline is triggered in CircleCI



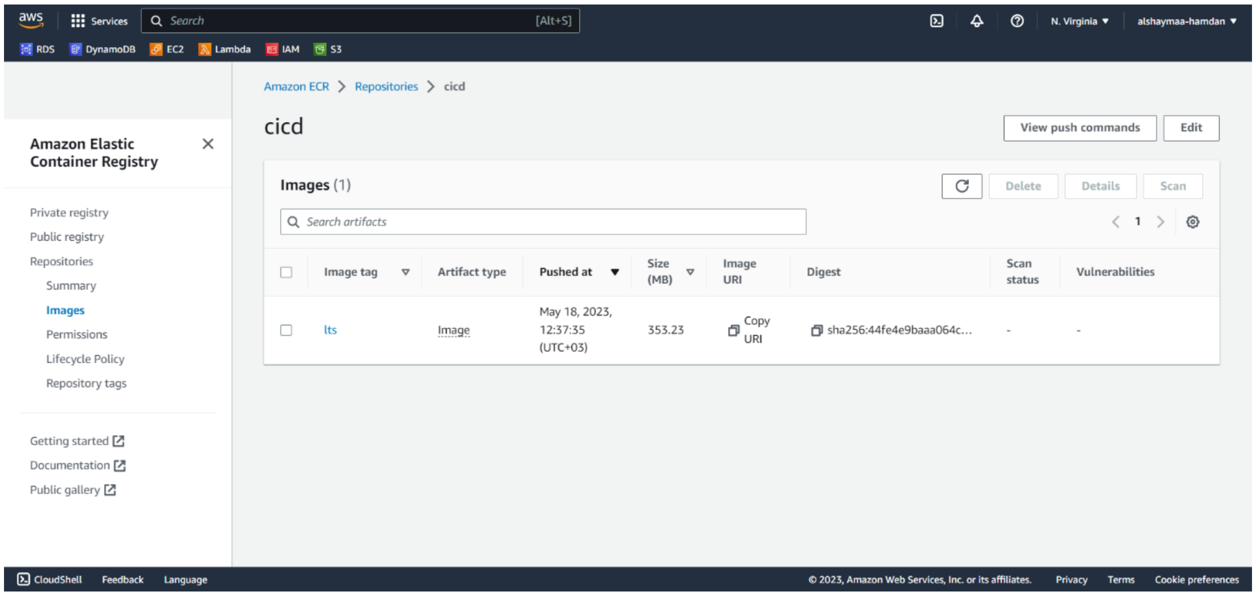
Dashboard: All Pipelines > Project: jenkins_nodeapp > Branch: master > Workflow: build

build Success

Duration / Finished	Branch	Commit	Author & Message
52s / 27m ago	master	249e34d	CircleCI & ECR config.yml

test 2s → aws-ecr/build-and-push-i... 24s

- After the pipeline finish running and its status is success, check ECR “cicd” repository, it should contain a new image with tag “lts”



Amazon ECR > Repositories > cicd

cicd

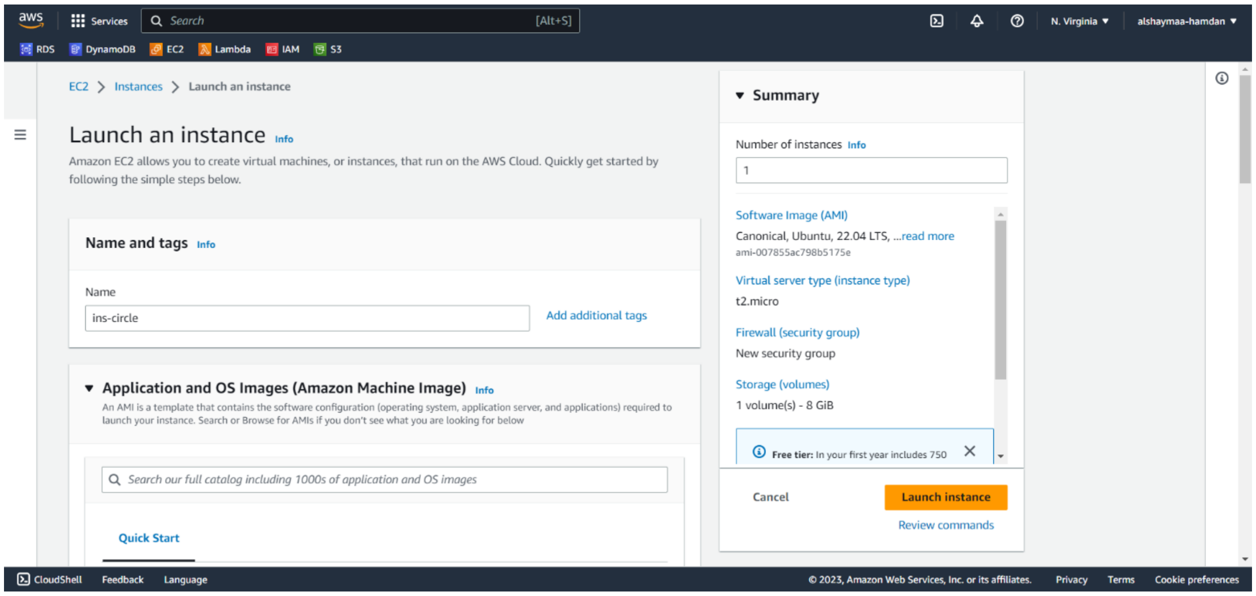
Images (1)

	Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest	Scan status	Vulnerabilities
☐	lts	Image	May 18, 2023, 12:37:35 (UTC+03)	353.23	Copy URI	sha256:44fe4e9baaa064c...	-	-

Step 4: Deploy image to AWS EC2

Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the Amazon Web Services (AWS) Cloud. Using Amazon EC2 eliminates your need to invest in hardware up front, so you can develop and deploy applications faster.

- Create EC2 instance
  - Go to “EC2” Service in AWS Console
  - Click “Launch instance”,



EC2 > Instances > Launch an instance

Launch an instance

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags

Name: ins-circle

Application and OS Images (Amazon Machine Image)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Summary

Number of instances: 1

Software Image (AMI): Canonical, Ubuntu, 22.04 LTS, ...read more ami-007855ac798b5175e

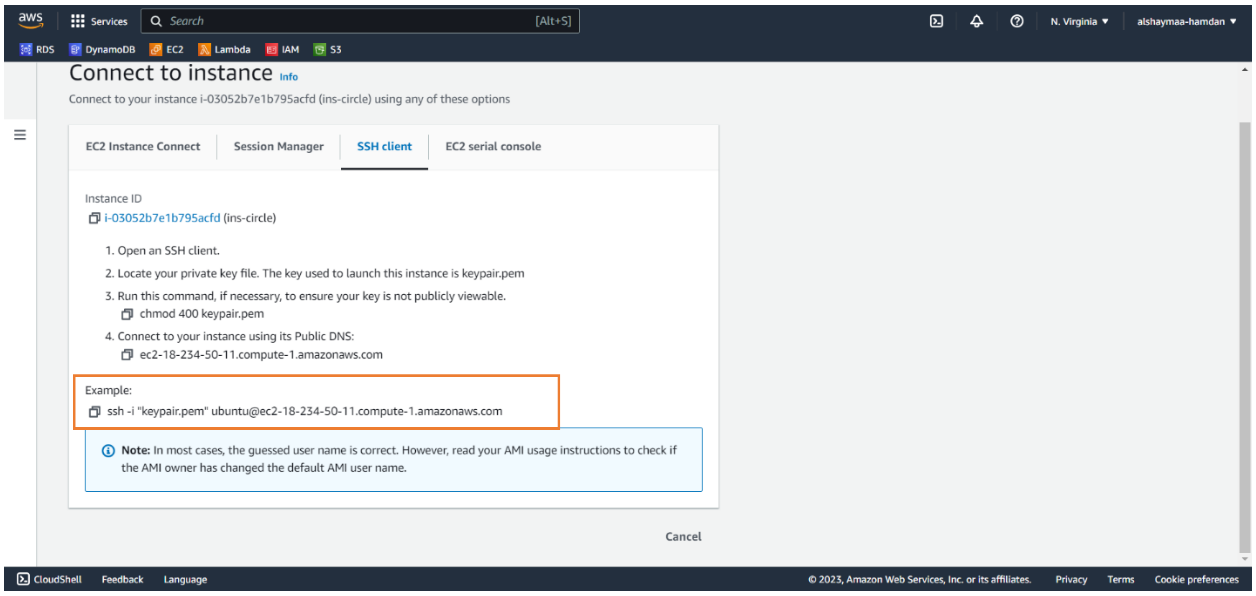
Virtual server type (instance type): t2.micro

Firewall (security group): New security group

Storage (volumes): 1 volume(s) - 8 GiB

Free tier: In your first year includes 750

  - Choose Ubuntu OS Image, instance type “t2.micro”, choose a key pair or create a new one, and choose the security group or create a new one
  - Wait until the instance is running.
- Connect to “ins-circle” instance
  - Choose the instance, click “Connect”, go to SSH Client, take the command from this page:



Connect to instance Info

Connect to your instance i-03052b7e1b795acfd (ins-circle) using any of these options

EC2 Instance Connect | Session Manager | SSH client | EC2 serial console

Instance ID

i-03052b7e1b795acfd (ins-circle)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is keypair.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
   chmod 400 keypair.pem
4. Connect to your instance using its Public DNS:
   ec2-18-234-50-11.compute-1.amazonaws.com

Example:

```
ssh -i "keypair.pem" ubuntu@ec2-18-234-50-11.compute-1.amazonaws.com
```

Note: In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

  - Run this command in a terminal (cd to where “keypair.pem” is located):

```
ssh -i "keypair.pem" ubuntu@ec2-18-234-50-11.compute-1.amazonaws.com
```

  - This command will connect to “ins-circle” instance and use it the way that you'd use a computer sitting in front of you
  - Install "AWS Cli” on the instance

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
aws --version
```

  - Install “Docker” on the instance

```
sudo apt-get update
sudo apt-get install ca-certificates curl gnupg
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/etc/apt/keyrings/docker.gpg
```

```
sudo chmod a+r /etc/apt/keyrings/docker.gpg
```

```
echo \
  "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
  "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

```
sudo apt-get update
```

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

- Generate an "ssh key" to connect the instance with "CircleCI" account
  - Run this command on your terminal to create "ssh key" (replace to your email)

```
ssh-keygen -t rsa -b 4096 -C "alshaymaa.aref@hotmail.com"
```

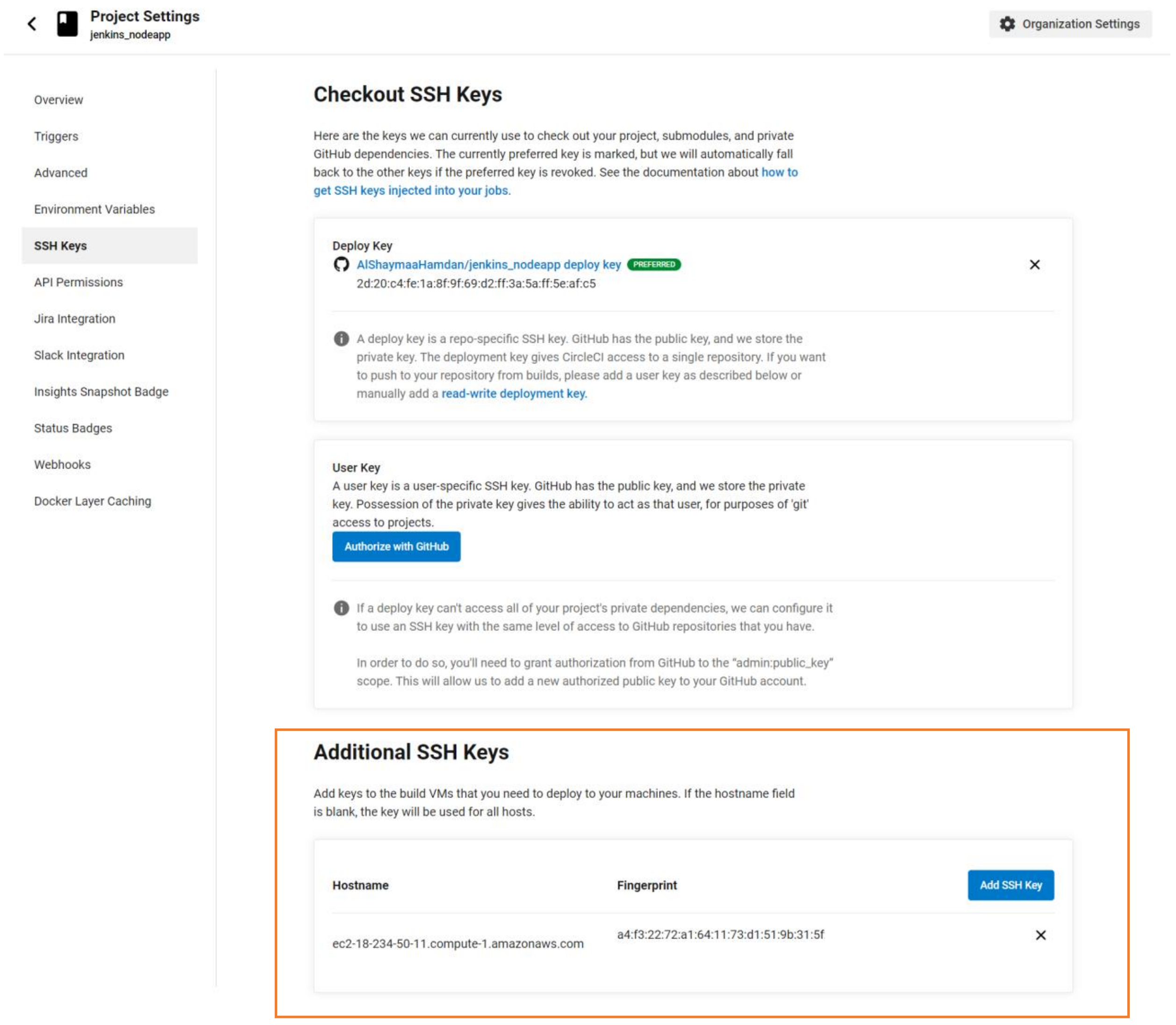
  Will generate ssh key pair: id_rsa, id_rsa.pub
  - ssh to the instance using keypair.pem key to add the new ssh key

```
ssh -i "keypair.pem" ubuntu@ec2-18-234-50-11.compute-1.amazonaws.com
```

  - move id_rsa.pub to /home/ubuntu/.ssh folder in the instance
  - copy id_rsa.pub content and and paste it in /home/ubuntu/.ssh/authorized_keys file
- configure aws cli:

aws configure	Insert AWS ACCESS KEY ID and Secret ACCESS KEY and Region

-
- Configure "CircleCI" to deploy the application over SSH
  - In the "CircleCI" application, go to your project's settings by clicking the the Project Settings button
  - On the Project Settings page, click on "SSH Keys".
  - Scroll down to the "Additional SSH Keys" section.
  - Click the "Add SSH Key" button.
  - In the "Hostname" field, enter the key's associated host.
  - In the "Private Key" field, paste the SSH key you are adding (content of id_rsa)
  - Click the "Add SSH Key" button.



Project Settings
jenkins_nodeapp

Organization Settings

Overview
Triggers
Advanced
Environment Variables
SSH Keys
API Permissions
Jira Integration
Slack Integration
Insights Snapshot Badge
Status Badges
Webhooks
Docker Layer Caching

Checkout SSH Keys

Here are the keys we can currently use to check out your project, submodules, and private GitHub dependencies. The currently preferred key is marked, but we will automatically fall back to the other keys if the preferred key is revoked. See the documentation about how to get SSH keys injected into your jobs.

Deploy Key
AlShaymaaHamdan/jenkins_nodeapp deploy key PREFERRED
2d:20:c4:fe:1a:8f:9f:69:d2:ff:3a:5a:ff:5e:af:c5

A deploy key is a repo-specific SSH key. GitHub has the public key, and we store the private key. The deployment key gives CircleCI access to a single repository. If you want to push to your repository from builds, please add a user key as described below or manually add a read-write deployment key.

User Key
A user key is a user-specific SSH key. GitHub has the public key, and we store the private key. Possession of the private key gives the ability to act as that user, for purposes of 'git' access to projects.

Authorize with GitHub

If a deploy key can't access all of your project's private dependencies, we can configure it to use an SSH key with the same level of access to GitHub repositories that you have.

In order to do so, you'll need to grant authorization from GitHub to the "admin:public_key" scope. This will allow us to add a new authorized public key to your GitHub account.

Additional SSH Keys

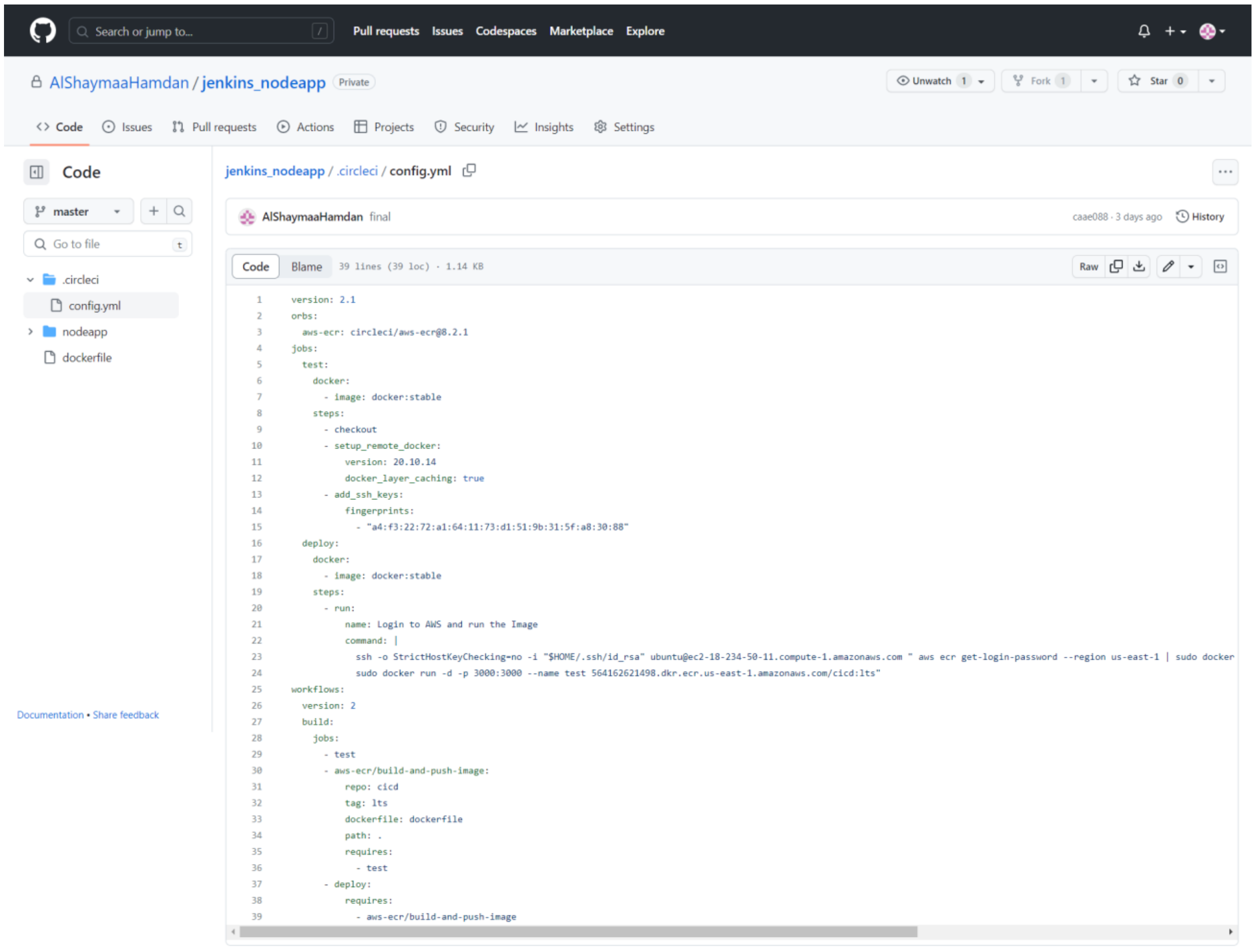
Add keys to the build VMs that you need to deploy to your machines. If the hostname field is blank, the key will be used for all hosts.

Hostname	Fingerprint
ec2-18-234-50-11.compute-1.amazonaws.com	a4:f3:22:72:a1:64:11:73:d1:51:9b:31:5f

Add SSH Key

Step 5: (Connect to EC2 instance to deploy the application) using CircleCI

- Edit CircleCI config.yml file



```yaml
version: 2.1
orbs:
  aws-ecr: circleci/aws-ecr@8.2.1
jobs:
  test:
    docker:
      - image: docker:stable
    steps:
      - checkout
      - setup_remote_docker:
          version: 20.10.14
          docker_layer_caching: true
      - add_ssh_keys:
          fingerprints:
            - "a4:f3:22:72:a1:64:11:73:d1:51:9b:31:5f:a8:30:88"
  deploy:
    docker:
      - image: docker:stable
    steps:
      - run:
          name: Login to AWS and run the Image
          command: |
            ssh -o StrictHostKeyChecking=no -i "$HOME/.ssh/id_rsa" ubuntu@ec2-18-234-50-11.compute-1.amazonaws.com " aws ecr get-login-password --region us-east-1 | sudo docker
            sudo docker run -d -p 3000:3000 --name test 564162621498.dkr.ecr.us-east-1.amazonaws.com/cicd:lts"
workflows:
  version: 2
  build:
    jobs:
      - test
      - aws-ecr/build-and-push-image:
          repo: cicd
          tag: lts
          dockerfile: dockerfile
          path: .
          requires:
            - test
      - deploy:
          requires:
            - aws-ecr/build-and-push-image
```

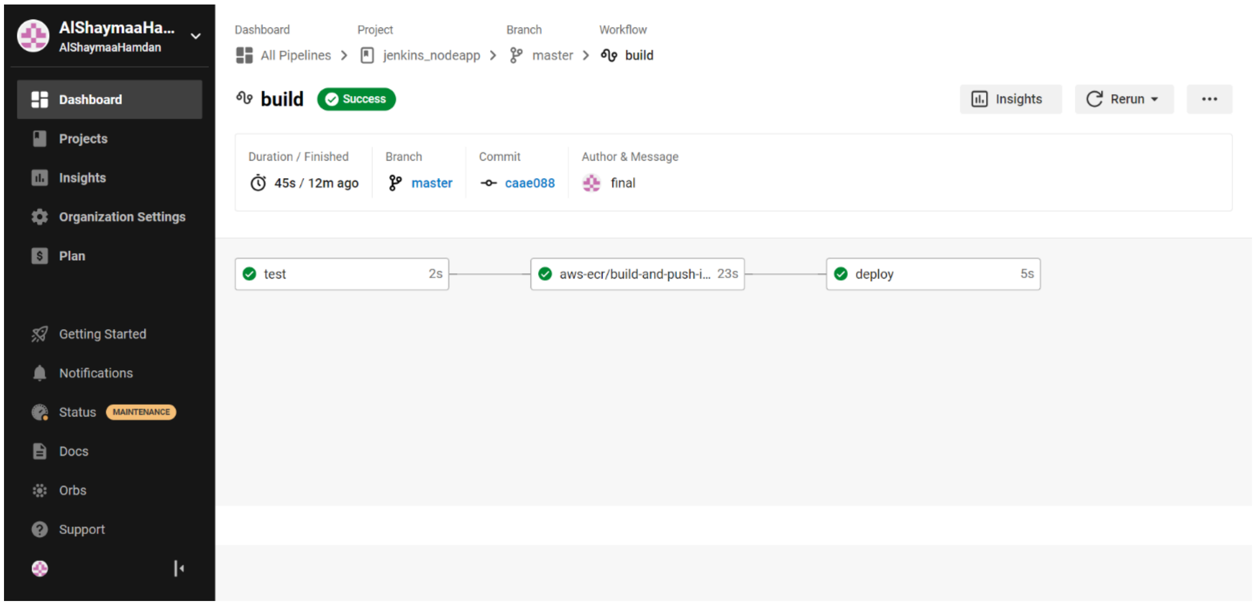
config.yml Content:

```yaml
version: 2.1
orbs:
  aws-ecr: circleci/aws-ecr@8.2.1
jobs:
  test:
    docker:
      - image: docker:stable
    steps:
      - checkout
      - setup_remote_docker:
          version: 20.10.14
          docker_layer_caching: true
      - add_ssh_keys:
          fingerprints:
            - "a4:f3:22:72:a1:64:11:73:d1:51:9b:31"
```

Even though all CircleCI jobs use ssh-agent to automatically sign all added SSH keys, you must use the add_ssh_keys key to actually add keys to a container.
You can have the fingerprint from CircleCI Additional SSH keys

deploy: docker: - image: docker:stable steps: - run: name: Login to AWS and run the Image command: | ssh -o StrictHostKeyChecking=no -i "$HOME/.ssh/id_rsa" ubuntu@ec2-18-234-50-11.compute-1.amazonaws.com " aws ecr get-login-password --region us-east-1 | sudo docker login --username AWS --password-stdin 564162621498.dkr.ecr.us-east-1.amazonaws.com && sudo docker run -d -p 3000:3000 --name test 564162621498.dkr.ecr.us-east-1.amazonaws.com/cicd:lts"	Create a new job called "deploy" ssh to ec2 instance, use the key id_rsa. When connected to EC2 do some commands: 1. Login to AWS with region us-east-1 that contains the created ECR. aws ecr get-login-password : To retrieve a password to authenticate to a registry 2. Run container from the image that was pushed to ECR
workflows: version: 2 build: jobs: - test - aws-ecr/build-and-push-image: repo: cicd tag: lts dockerfile: dockerfile path: . requires: - test - deploy: requires: - aws-ecr/build-and-push-image	run test job then push the image to ECR then run deploy job.

- After committing config.yml file changes a new pipeline is triggered in CircleCI

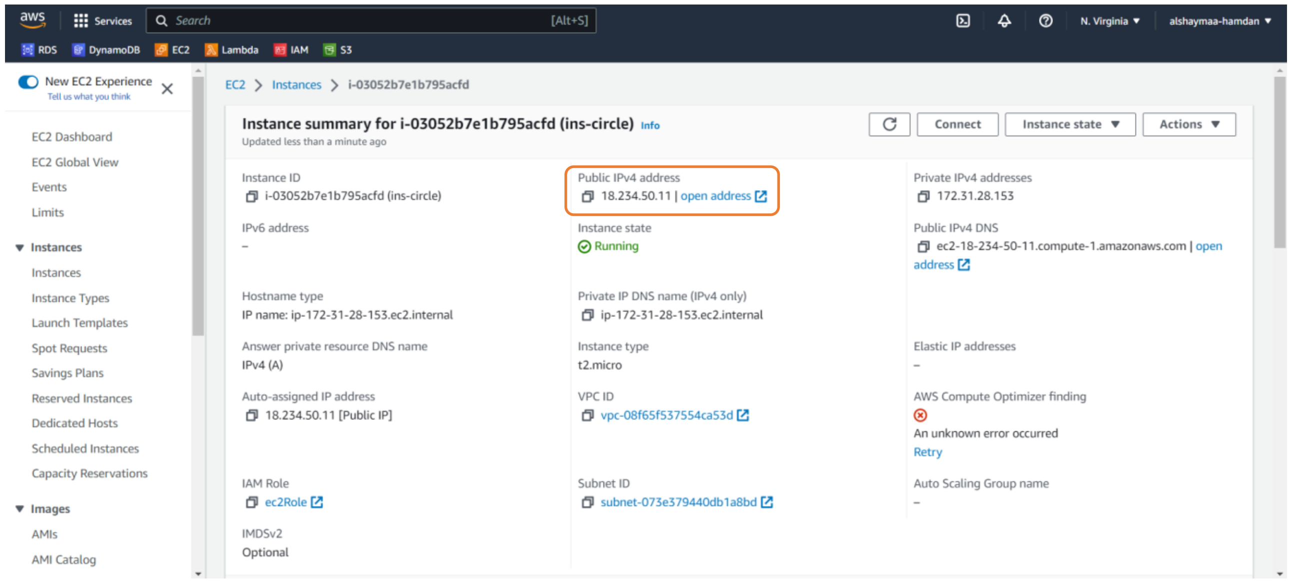


45s / 12m ago | master | caae088 | final

test 2s — aws-ecr/build-and-push-i... 23s — deploy 5s

Step 6: Check the Deployment

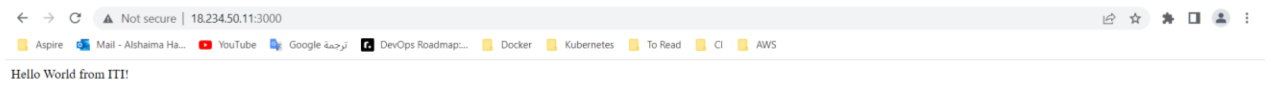
- Browse to “public IP:3000”
  Get the public IP from EC2 “ins-circle” details



Instance summary for i-03052b7e1b795acfd (ins-circle) Info
Updated less than a minute ago

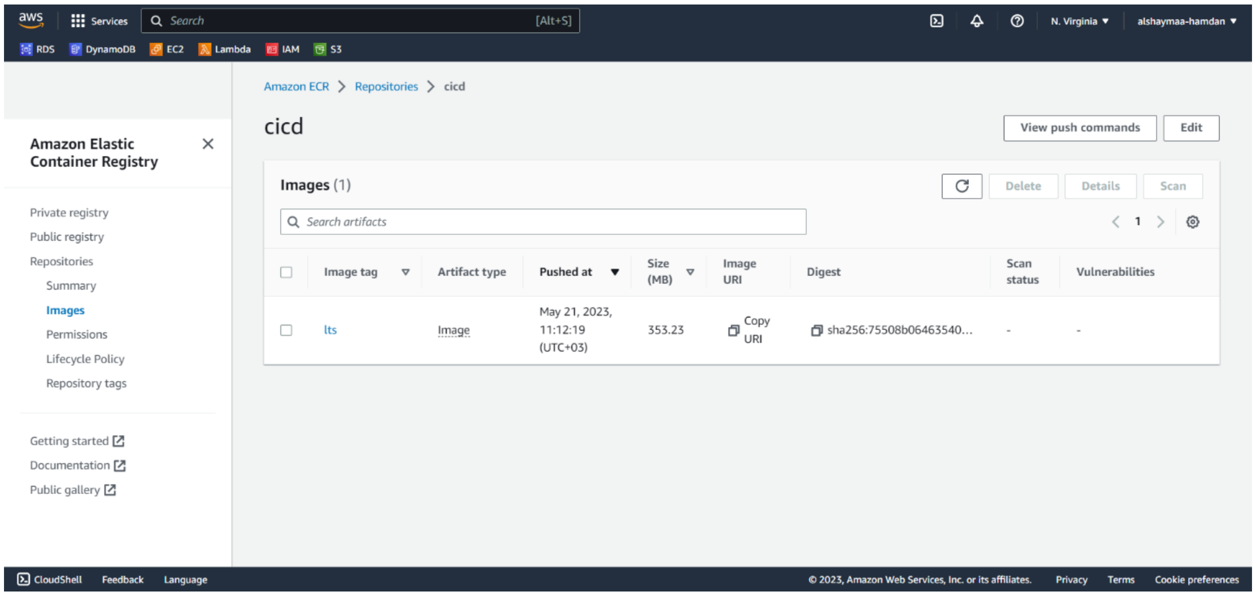
Instance ID i-03052b7e1b795acfd (ins-circle)	Public IPv4 address 18.234.50.11 | open address	Private IPv4 addresses 172.31.28.153
IPv6 address –	Instance state Running	Public IPv4 DNS ec2-18-234-50-11.compute-1.amazonaws.com | open address
Hostname type IP name: ip-172-31-28-153.ec2.internal	Private IP DNS name (IPv4 only) ip-172-31-28-153.ec2.internal	
Answer private resource DNS name IPv4 (A)	Instance type t2.micro	Elastic IP addresses –
Auto-assigned IP address 18.234.50.11 [Public IP]	VPC ID vpc-08f65f537554ca53d	AWS Compute Optimizer finding An unknown error occurred Retry
IAM Role ec2Role	Subnet ID subnet-073e379440db1a8bd	Auto Scaling Group name –
IMDSv2 Optional		

After browsing to the address the page should appear like this.



Hello World from ITI!

- Check the image was pushed to ECR



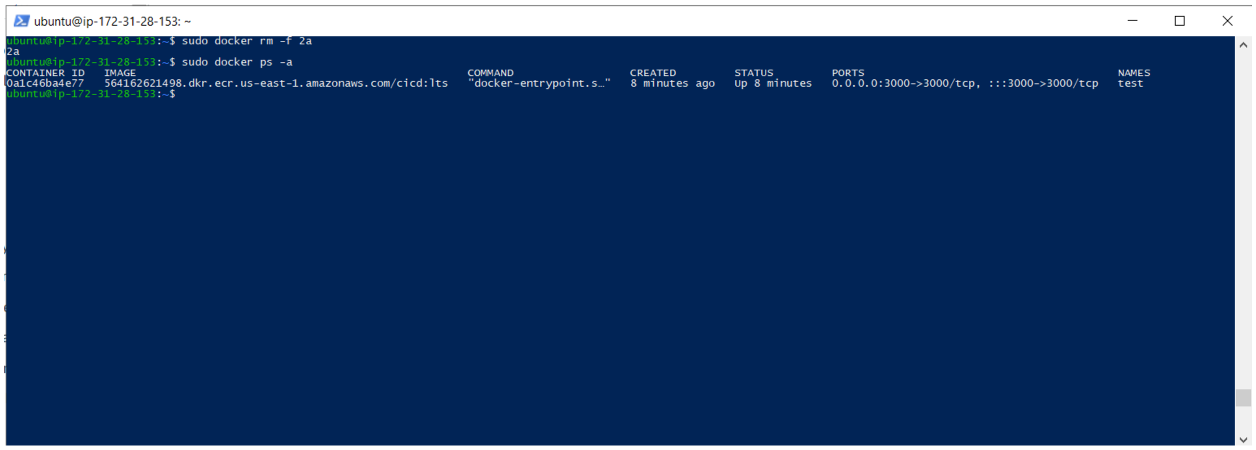
Amazon ECR > Repositories > cicd

cicd

Images (1)

	Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest	Scan status	Vulnerabilities
☐	lts	Image	May 21, 2023, 11:12:19 (UTC+03)	353.23	Copy URI	sha256:75508b06463540...	-	-

- Check the container is running in the ec2 instance



```
ubuntu@ip-172-31-28-153:~$ sudo docker rm -f 2a
2a
ubuntu@ip-172-31-28-153:~$ sudo docker ps -a
CONTAINER ID   IMAGE                                                COMMAND                  CREATED         STATUS         PORTS                                       NAMES
0a1c46ba4e77   564162621498.dkr.ecr.us-east-1.amazonaws.com/cicd:lts   "docker-entrypoint.s…"   8 minutes ago   Up 8 minutes   0.0.0.0:3000->3000/tcp, :::3000->3000/tcp   test
ubuntu@ip-172-31-28-153:~$
```