KLE Society's
KLE Technological University

DISTRIBUTED AND CLOUD COMPUTING

Course Code: 20ECSC305

CICD PIPELINE ON KUBERNETES USING AWS

TEAM - 02

Shreeya Goggi	**01fe19bcs045**	**140**
Rashmi Kiragi	**01fe19bcs057**	**150**
Renuka Talwar	**01fe19bcs068**	**159**
Sahana Bhasme	**01fe19bcs072**	**163**

COURSE TEACHER
Priyadarshini Patil

SCHOOL OF COMPUTER SCIENCE & ENGINEERING
KLE TECHNOLOGICAL UNIVERSITY
HUBLI–580 031 (India)

Academic year 2021-22

Abstract

Kubernetes,also known as K8s, is an open-source system for automating deployment, scaling, and management of containerized applications across multiple hosts.It groups containers that make up an application into logical units for easy management and discovery. Kubernetes builds upon a decade and a half of experience at Google running production workloads at scale using a system called Brog, combined with best-of-breed ideas and practices from the community.Kubernetes is hosted by the Cloud Native Computing Foundation. Amazon, Google, IBM,Microsoft, Oracle, Red Hat, SUSE and VMWare offer Kubernetes-based platforms or infrastructure as a service (IaaS) that deploy Kubernetes.

Introduction

Why Kubernetes ?

Kubernetes allows us to deploy cloud-native applications anywhere and manage them everywhere.As most modern software developers can attest, containers have provided us with dramatically more flexibility for running cloud-native applications on physical and virtual infrastructure. Containers package up the services comprising an application and make them portable across different compute environments, for both dev/test and production use. With containers, it’s easy to quickly ramp application instances to match spikes in demand. And because containers draw on resources of the host OS, they are much lighter weight than virtual machines. This means containers make highly efficient use of the underlying server infrastructure.

How are Kubernetes different ?

Traditional deployment era:

Early on, organizations ran applications on physical servers. There was no way to define resource boundaries for applications in a physical server, and this caused resource allocation issues. For example, if multiple applications run on a physical server, there can

be instances where one application would take up most of the resources, and as a result, the other applications would underperform. A solution for this would be to run each application on a different physical server. But this did not scale as resources were underutilized, and it was expensive for organizations to maintain many physical servers.

Virtualized deployment era:

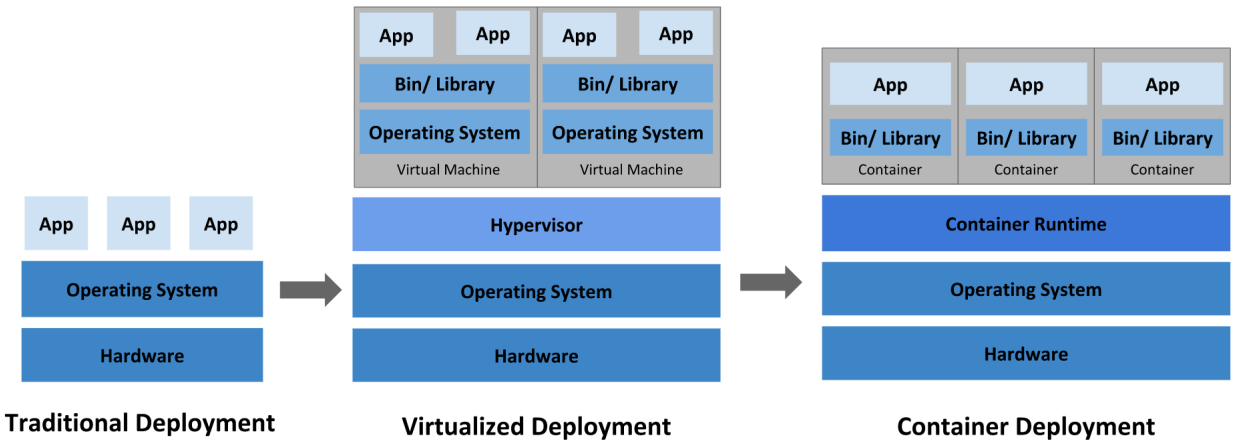
As a solution, virtualization was introduced. It allows you to run multiple Virtual Machines (VMs) on a single physical server's CPU. Virtualization allows applications to be isolated between VMs and provides a level of security as the information of one application cannot be freely accessed by another application.

Virtualization allows better utilization of resources in a physical server and allows better scalability because an application can be added or updated easily, reduces hardware costs, and much more. With virtualization you can present a set of physical resources as a cluster of disposable virtual machines.

Each VM is a full machine running all the components, including its own operating system, on top of the virtualized hardware.

Container deployment era:

Containers are similar to VMs, but they have relaxed isolation properties to share the Operating System (OS) among the applications. Therefore, containers are considered lightweight. Similar to a VM, a container has its own filesystem, share of CPU, memory, process space, and more. As they are decoupled from the underlying infrastructure, they are portable across clouds and OS distributions.



Benefits of Containers

- Agile application creation and deployment: increased ease and efficiency of container image creation compared to VM image use.
- Continuous development, integration, and deployment: provides for reliable and frequent container image build and deployment with quick and efficient rollbacks (due to image immutability).
- Dev and Ops separation of concerns: create application container images at build/release time rather than deployment time, thereby decoupling applications from infrastructure.
- Observability: not only surfaces OS-level information and metrics, but also application health and other signals.
- Environmental consistency across development, testing, and production: Runs the same on a laptop as it does in the cloud.
- Cloud and OS distribution portability: Runs on Ubuntu, RHEL, CoreOS, on-premises, on major public clouds, and anywhere else.
- Application-centric management: Raises the level of abstraction from running an OS on virtual hardware to running an application on an OS using logical resources.

- Loosely coupled, distributed, elastic, liberated micro-services: applications are broken into smaller, independent pieces and can be deployed and managed dynamically – not a monolithic stack running on one big single-purpose machine.
- Resource isolation: predictable application performance.
- Resource utilization: high efficiency and density.

Gaps And Challenges

Enterprises are quickly discovering new Kubernetes use cases to increase agility and application performance. While Kubernetes provides more freedom to run applications across a variety of infrastructures, broader adoption means a proliferation of new clusters and increased complexity in how containers are run, monitored and managed.While managing one Kubernetes cluster is not trivial, trying to manage multiple Kubernetes clusters on hybrid or multiple clouds becomes exponentially more difficult.There are a number of common challenges organizations run into when attempting to efficiently rein in multiple clusters.Developers should keep the following in mind when developing a strategy to overcome these three common governance challenges in Kubernetes deployments:

Lack of Visibility and Management:

As the number of clusters grows and spreads, managing and tracking their activity and growth becomes increasingly difficult. It is also harder and more time-consuming to troubleshoot any problems that may arise; if different software is involved, a single solution cannot be applied to each version. Lack of centralized governance and visibility into what's happening within the Kubernetes environment negatively impacts application availability and performance and, ultimately, the organization's bottom line.

Problems also arise if a cluster goes down unexpectedly—troubleshooting problems requires time and resources, and if there are dozens of potential software versions in use, managing all of them across the organization is increasingly difficult. Unlike planned downtime, when teams are generally able to understand what is needed to mitigate impacts for customers and projects, unplanned downtime makes this preparation impossible. Operators must be able to consistently administer, manage and obtain insights about their infrastructure.

Operational Complexity and Overhead:

Spreading numerous Kubernetes clusters across different business units creates challenges in user identity tracking—especially if users onboard, offboard or change teams. Operators lose the flexibility to define user roles, responsibilities and privileges to ensure the right people are performing the right tasks within the environment. They also face difficulties identifying role violations, assessing governance risks and performing compliance checks. When more time is spent chasing issues or putting out fires, there is less time for efficient operations.

Empowering Both Developers and Operators:

As developers work to implement Kubernetes, there must be a balance between the developer's independence and the operator's ability to easily manage the policies and procedures necessary to maintain the overall system. While the autonomy of developers is crucial, this freedom shouldn't come at the expense of the environment's security. Multiple solutions across multiple clusters often lead to more opportunities for attacks to slip through the cracks unnoticed. The alternative approach, where there is more consistency for a few select solutions, allows standardization across organizational clusters. The challenge comes when addressing the fine line between a developer's ability to innovate and an operator's ability to maintain the necessary procedures and governance.

Problem Statement

To Build a CI/CD Pipeline for Microservices on Kubernetes.

Objectives

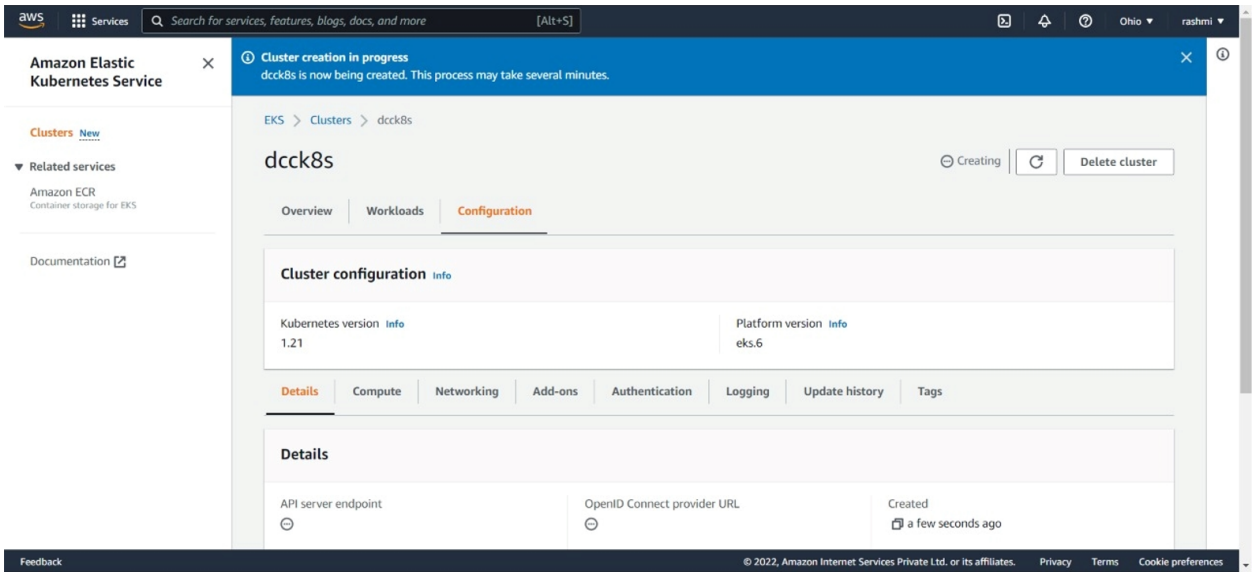
- To build and deploy services independently.
- Code changes that pass the CI process are automatically deployed to a production-like environment.
- A new version of a service can be deployed side by side with the previous version.

Tools

- AWS EC2 instance
- AWS EKS
- Jenkins
- Codecommit pipeline
- AWS ECS
- Github repository
- Kubectl
- Eksctl
- AWS cli

Program implementation plan

To set up a Kubernetes cluster consisting of one Master node and one or more worker nodes.



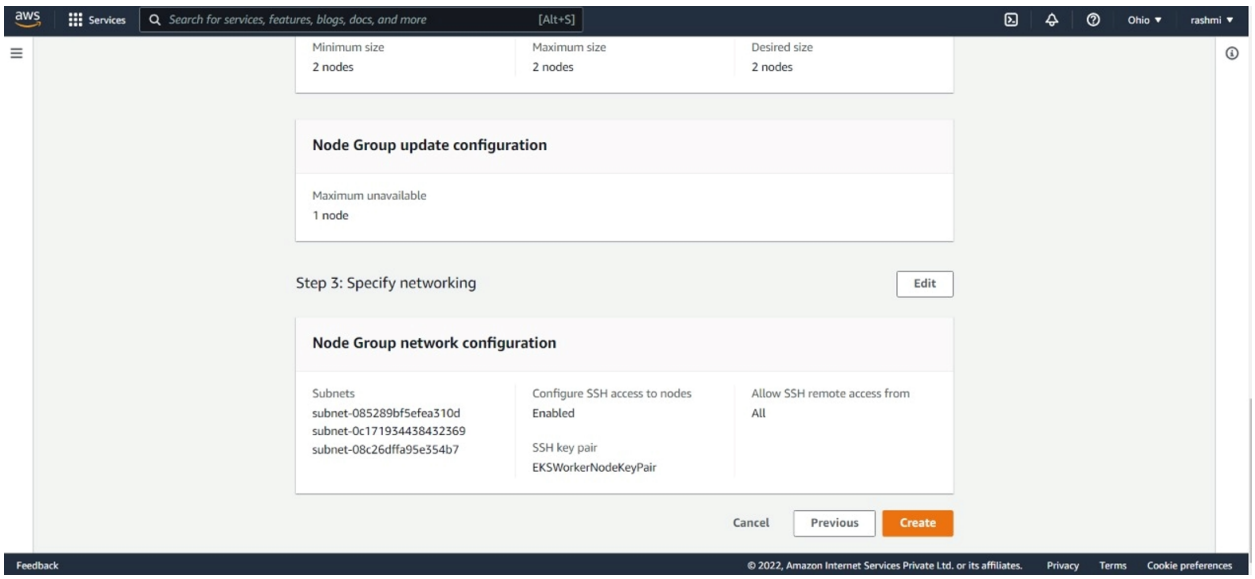
RSA public key generator

```
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data:
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUM1ekNDQWMrZ0F3SUJBZ0lCQURBTkJna
3Foa2lHOXcwQkFRc0ZBREFWTVJNd0VRWURWUVFERXdwcmRXSmwKY201bGRHVnpNQjRYRFRJeU
1EUXhNekE0TWpVME1Gb1hEVE15TURReE1EQTRNalUwTUZvd0ZURVRNQkVHQTFVRQpBeE1LYTN
WaVpYSnVaWFJsY3pDQ0FTSXdEUVlKS29aSWh2Y05BUUVCQlFBRGdnRVBBRENDQVFvQ2dnRUJB
S2g5CnlDV1U5WGE0ZmhZUk1MVHdzbk56R1VXNjlialpaY0tsb3JKZmxUMzFIdHF4eFRTTFhtM
DdNWUs0UTNWL3YydloKak0vS0ZKWEJsN1plUVhvNjAzTzdpbk85OGQzMkVQSkZGcUZlNlVqT1
lVSXVDcFV1c1RYTHJUeGd0bWZTVDRmbgpIbXEvRHJpVDIxemFzc3I1YkQ3MzU0NkZPL2w3OXN
LUklWdDBTVnhNYk1LTXF1V2FFMjMvYkhLa3JKNXN2MlFPClA2NTRnVGg4UzFsaDhHY2hlZXN4
UEpiRTY0aGdHcXUvcWJFbll1ZnF4UjN3Y0N3OHN2amhBSkJKck5jZlE5VGkKa2xLNEVJSnBib
jdJUEVNYU1UcHlFUWlDMVNQV1lQRDJxaEpCd2RCMnM3L2pLL2tRV1E3S1RocHMxTk5KU1pwQw
plUU9tenUvcHhRWVJVb0gxQ3YwQ0F3RUFBYU5DTUVBd0RnWURWUjBQQVFIL0JBUURBZ0trTUE
4R0ExVWRFd0VCCi93UUZNQU1CQWY4d0hRWURWUjBPQkJZRUZCZmozTi9MOFZsbFlOZGpWeE1Q
djRJcE05Z0tNQTBHQ1NxR1NJYjMKRFFFQkN3VUFBNElCQVFCVmd1VmJDT0hXUUxmZGllM0pYM
FVnL3ZZckRKVVowT3BHWFM2SXVZT01ZRkwwcndtegpROWY3TFFJb0VDVlRHeW52RERqOTRyT0
xZMVFGUHIydENiZHM0WXZpdHpCYzZPNFBTeWozUi9Ed1p1WGtWUFloCkVITnB3UXV4SnFrRFh
HNjhQcXp2WUQ3MURQUUJ5NXYzQ254U0dqTUxRclRpVTNINFJMREVXNWdXS1VMNWE1RFMKaWgr
OGxOY2J3V3BjdUFjelRONWxWKzl2Q2hrWDhNaDYvUzQzS3l6a1d1U0h1K1Nsb0x1MUZlNWtlU
UI5WUozYwoyTDE3N3d2OE9qMEE2c1dVOUcxcU4rZU1IWmZnUnFNUjh6V0ZmNFpTWTZlL3dZYz
RqV0I2WndqSmdaMloxQ1JZCm1wZ05RQkVsMG4rWXpOVjRHRjk2RGxVNmc4T2hHNk82OENveQo
tLS0tLUVORCBDRVJUSUZJQ0FURS0tLS0tCg==
    server: https://6491FBC870261581C2DB82F94B9F5E72.gr7.us-east-
2.eks.amazonaws.com
  name: arn:aws:eks:us-east-2:803947141637:cluster/dcck8s
contexts:
- context:
    cluster: arn:aws:eks:us-east-2:803947141637:cluster/dcck8s
    user: arn:aws:eks:us-east-2:803947141637:cluster/dcck8s
  name: arn:aws:eks:us-east-2:803947141637:cluster/dcck8s
current-context: arn:aws:eks:us-east-2:803947141637:cluster/dcck8s
```

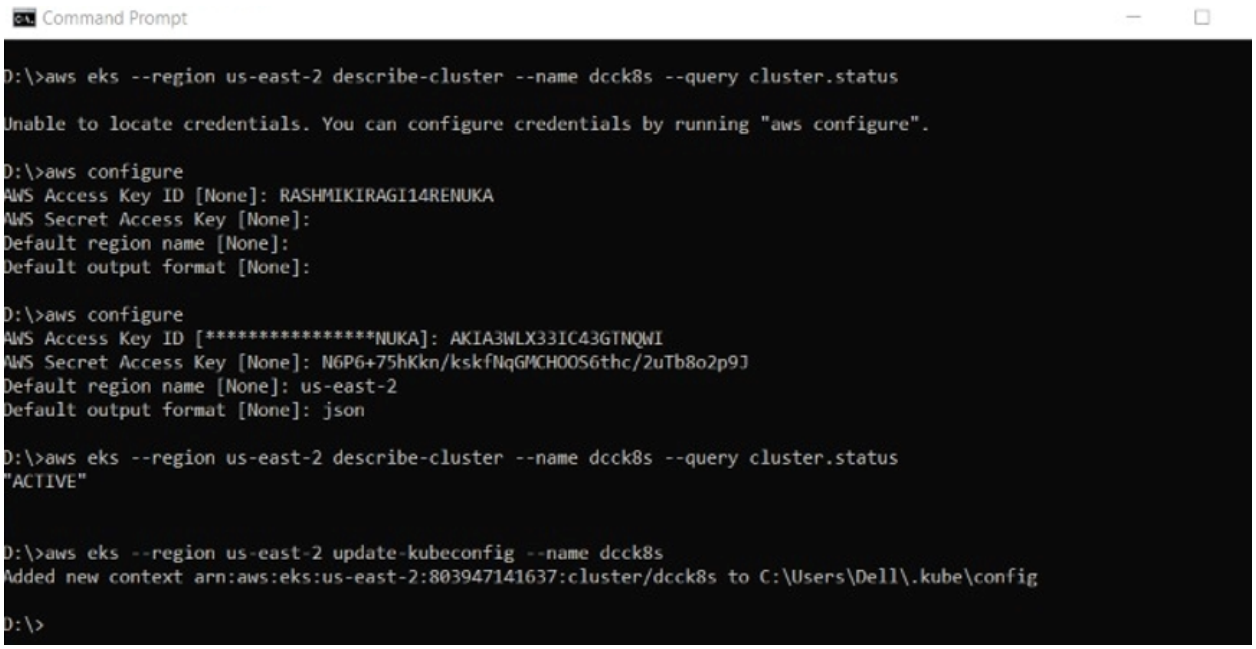
Creation of worker nodes

Command Prompt

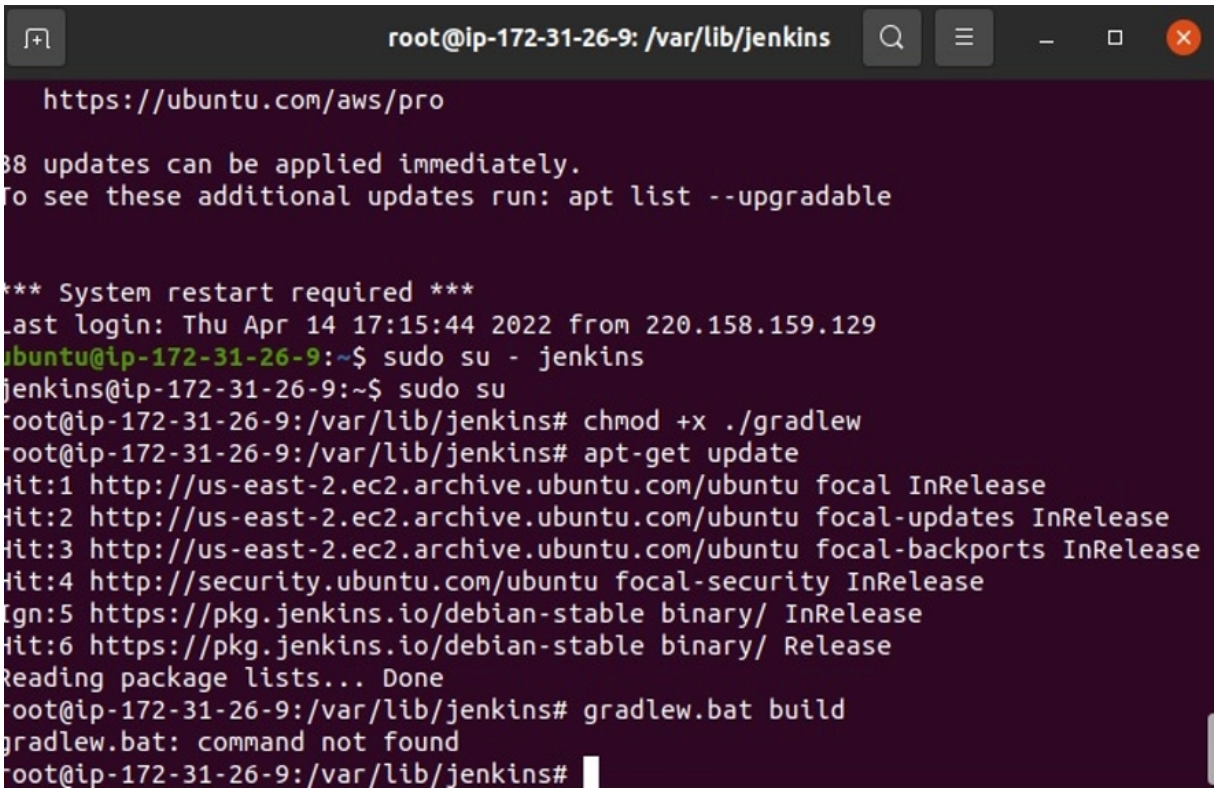
```
:\>aws configure
WS Access Key ID [****************NUKA]: AKIA3WLX33IC43GTNQWI
WS Secret Access Key [None]: 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
efault region name [None]: us-east-2
efault output format [None]: json

:\>aws eks --region us-east-2 describe-cluster --name dcck8s --query cluster.status
ACTIVE"


:\>aws eks --region us-east-2 update-kubeconfig --name dcck8s
dded new context arn:aws:eks:us-east-2:803947141637:cluster/dcck8s to C:\Users\Dell\.kube\config

:\>kubectl get svc
AME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
ubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   87m

:\>kubectl get nodes --watch
AME                                          STATUS   ROLES    AGE     VERSION
p-172-31-0-230.us-east-2.compute.internal    Ready    <none>   3m11s   v1.21.5-eks-9017834
p-172-31-26-23.us-east-2.compute.internal    Ready    <none>   3m24s   v1.21.5-eks-9017834
p-172-31-26-23.us-east-2.compute.internal    Ready    <none>   5m31s   v1.21.5-eks-9017834
p-172-31-0-230.us-east-2.compute.internal    Ready    <none>   5m32s   v1.21.5-eks-9017834

:\>kubectl get nodes
AME                                          STATUS   ROLES    AGE     VERSION
p-172-31-0-230.us-east-2.compute.internal    Ready    <none>   6m44s   v1.21.5-eks-9017834
p-172-31-26-23.us-east-2.compute.internal    Ready    <none>   6m57s   v1.21.5-eks-9017834
```



Configuring AWS cli

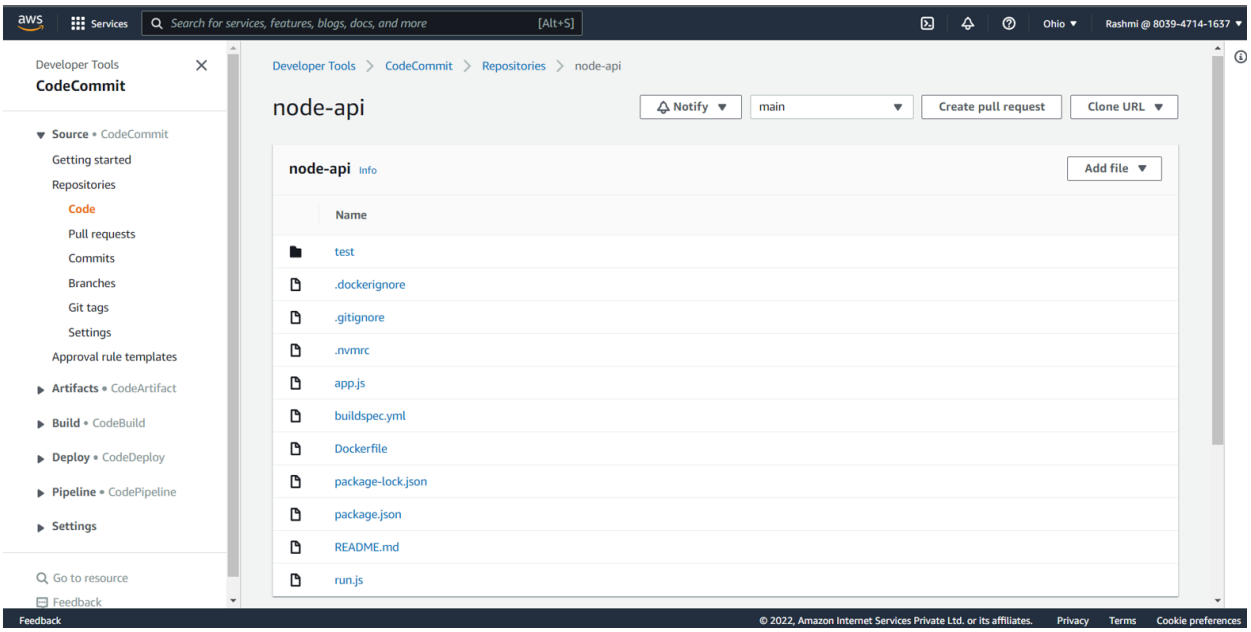


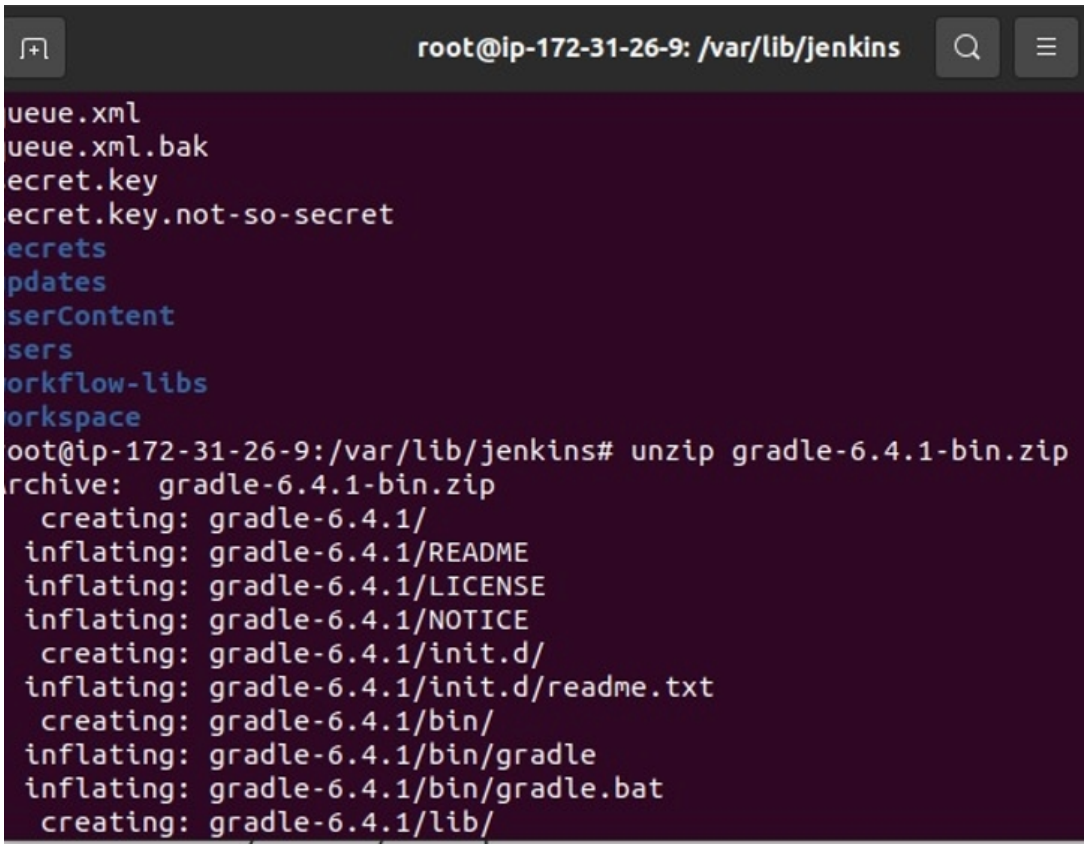
Creating CI/CD pipeline using jenkins

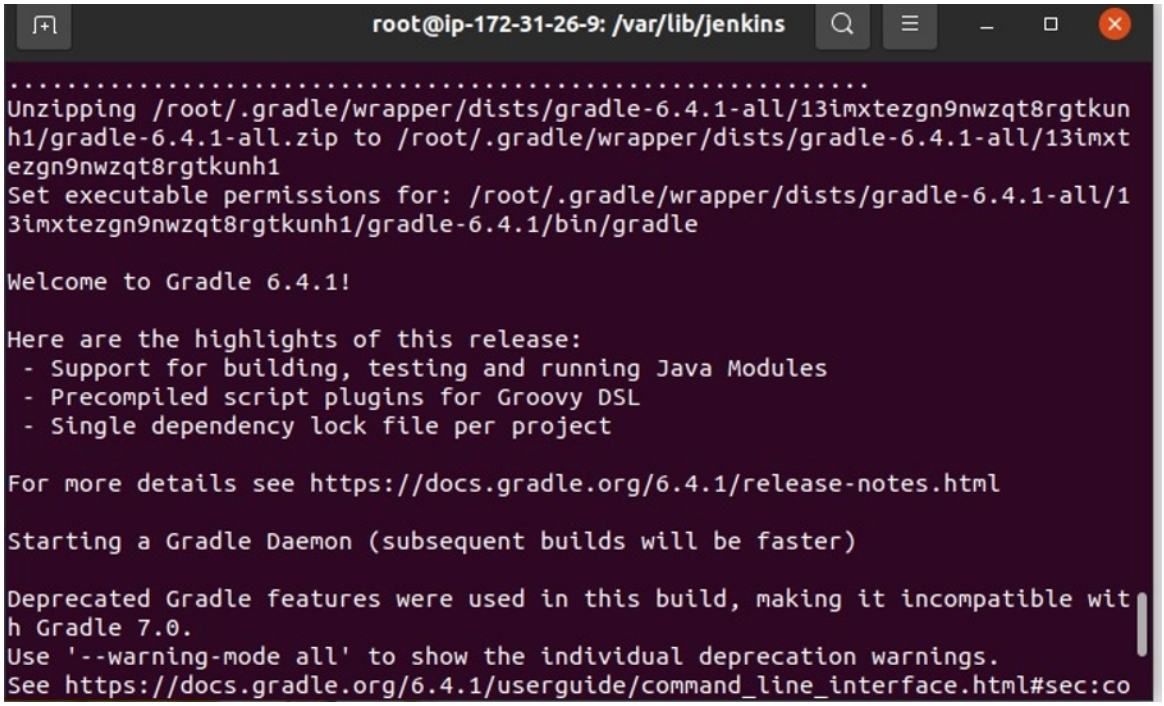


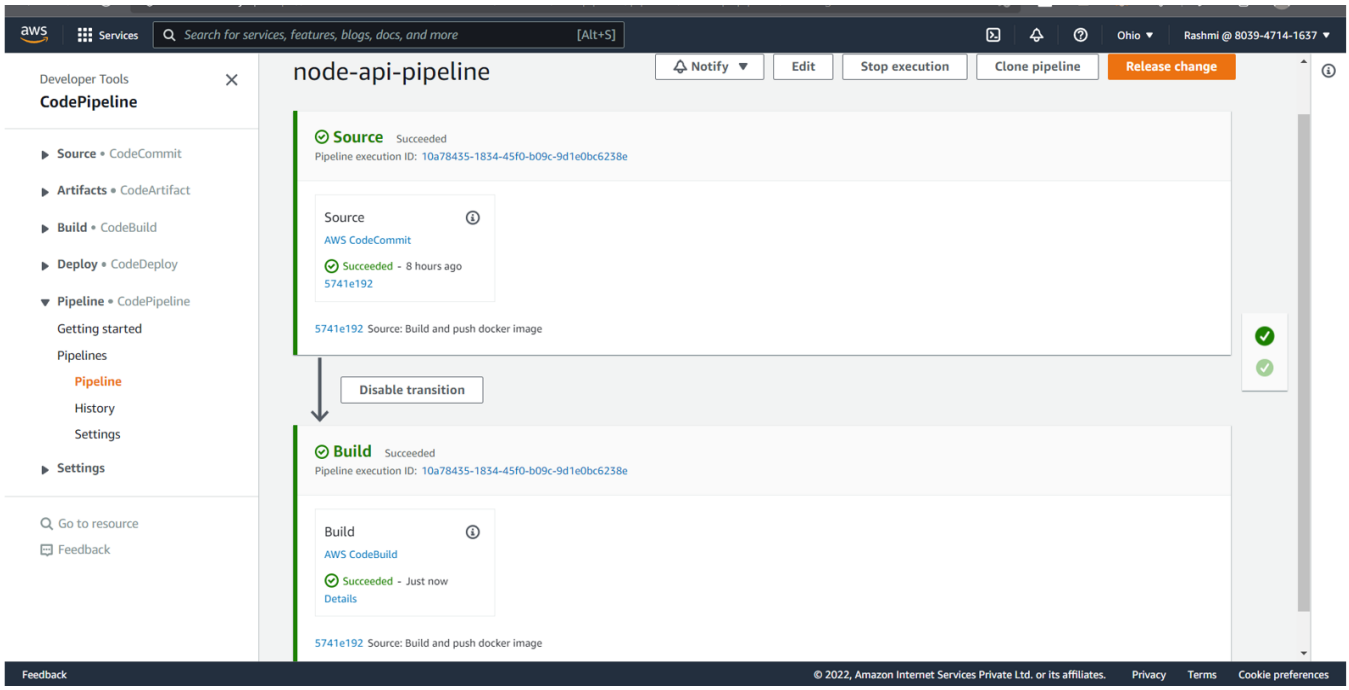
Pushing the application to Github repository and compiling using gradle and building

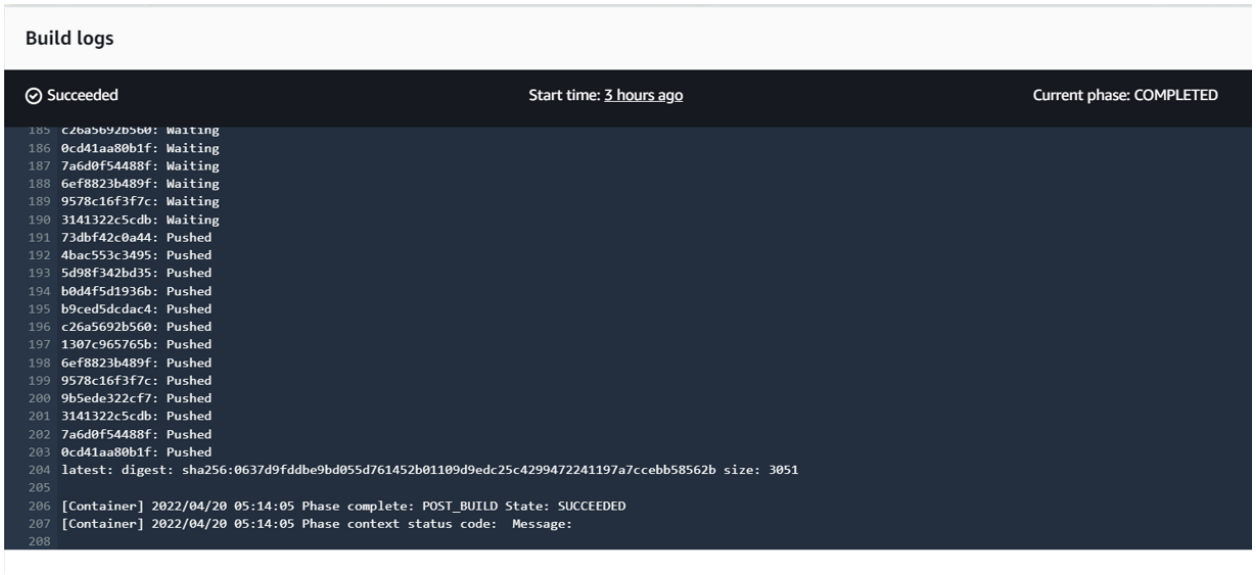
```
$ ls

Hp@rashmik MINGW64 ~/node-api (master)
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)

Hp@rashmik MINGW64 ~/node-api (master)
$ cd ..

Hp@rashmik MINGW64 ~
$ git clone ssh://APKA3WLX33ICU2IIF7ES@git-codecommit.us-east-2.amazonaws.com/v1/repos/node-api
fatal: destination path 'node-api' already exists and is not an empty directory.

Hp@rashmik MINGW64 ~
$ git clone ssh://APKA3WLX33ICU2IIF7ES@git-codecommit.us-east-2.amazonaws.com/v1/repos/node-api
Cloning into 'node-api'...
remote: Counting objects: 35, done.
Receiving objects: 100% (35/35), 32.45 KiB | 2.70 MiB/s, done.

Hp@rashmik MINGW64 ~
$ cd node-api

Hp@rashmik MINGW64 ~/node-api (main)
$ code .

Hp@rashmik MINGW64 ~/node-api (main)
$ git add buildspec.yml

Hp@rashmik MINGW64 ~/node-api (main)
$ git commit -m "Add buildspec"
[main 5d9c05e] Add buildspec
 1 file changed, 16 insertions(+)
 create mode 100644 buildspec.yml

Hp@rashmik MINGW64 ~/node-api (main)
$ git push remote main
fatal: 'remote' does not appear to be a git repository
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.

Hp@rashmik MINGW64 ~/node-api (main)
$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 392 bytes | 196.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
To ssh://git-codecommit.us-east-2.amazonaws.com/v1/repos/node-api
   13c9731..5d9c05e  main -> main

Hp@rashmik MINGW64 ~/node-api (main)
$
```


aws
Services
Search for services, features, blogs, docs, and more
[Alt+S]
Ohio
Rashmi @ 8039-4714-1637
Developer Tools
CodeCommit
Source • CodeCommit
Getting started
Repositories
Code
Pull requests
Commits
Branches
Git tags
Settings
Approval rule templates
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Settings
Go to resource
Feedback
Developer Tools > CodeCommit > Repositories > node-api
node-api
Notify
main
Create pull request
Clone URL
node-api Info
Add file
Name
test
.dockerignore
.gitignore
.nvmrc
app.js
buildspec.yml
Dockerfile
package-lock.json
package.json
README.md
run.js
Feedback
© 2022, Amazon Internet Services Private Ltd. or its affiliates.
Privacy
Terms
Cookie preferences

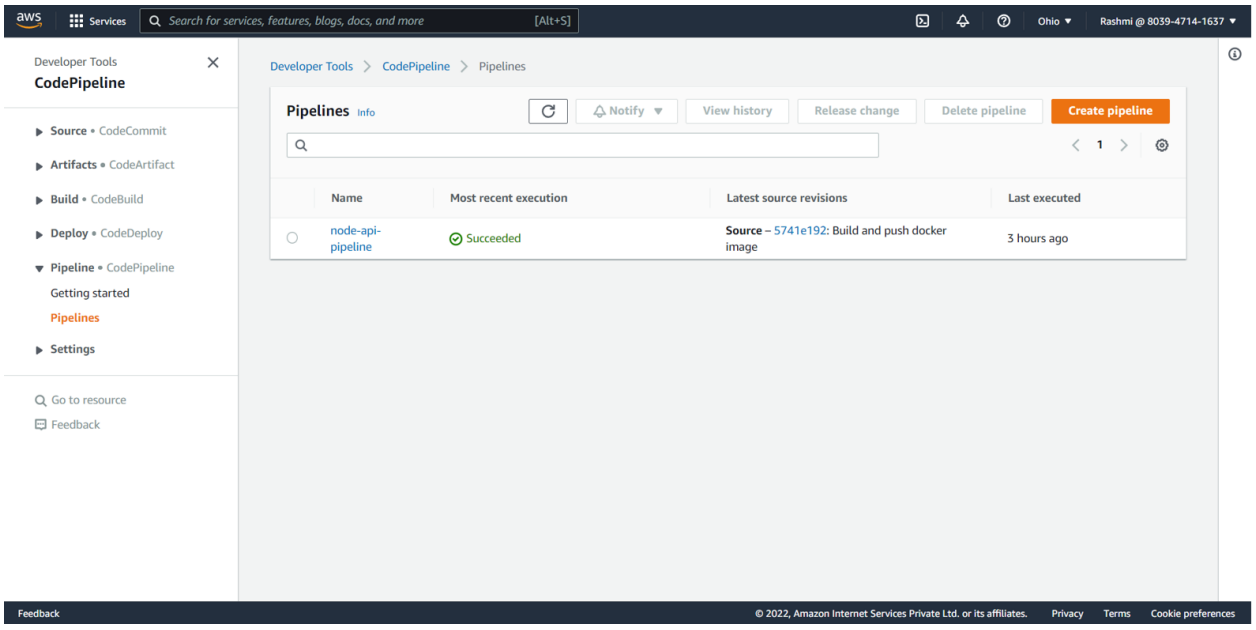

root@ip-172-31-26-9: /var/lib/jenkins
ueue.xml
ueue.xml.bak
ecret.key
ecret.key.not-so-secret
ecrets
pdates
serContent
sers
orkflow-libs
orkspace
oot@ip-172-31-26-9:/var/lib/jenkins# unzip gradle-6.4.1-bin.zip
rchive: gradle-6.4.1-bin.zip
creating: gradle-6.4.1/
inflating: gradle-6.4.1/README
inflating: gradle-6.4.1/LICENSE
inflating: gradle-6.4.1/NOTICE
creating: gradle-6.4.1/init.d/
inflating: gradle-6.4.1/init.d/readme.txt
creating: gradle-6.4.1/bin/
inflating: gradle-6.4.1/bin/gradle
inflating: gradle-6.4.1/bin/gradle.bat
creating: gradle-6.4.1/lib/


root@ip-172-31-26-9: /var/lib/jenkins
.....
Unzipping /root/.gradle/wrapper/dists/gradle-6.4.1-all/13imxtezgn9nwzqt8rgtkun
h1/gradle-6.4.1-all.zip to /root/.gradle/wrapper/dists/gradle-6.4.1-all/13imxt
ezgn9nwzqt8rgtkunh1
Set executable permissions for: /root/.gradle/wrapper/dists/gradle-6.4.1-all/1
3imxtezgn9nwzqt8rgtkunh1/gradle-6.4.1/bin/gradle
Welcome to Gradle 6.4.1!
Here are the highlights of this release:
- Support for building, testing and running Java Modules
- Precompiled script plugins for Groovy DSL
- Single dependency lock file per project
For more details see https://docs.gradle.org/6.4.1/release-notes.html
Starting a Gradle Daemon (subsequent builds will be faster)
Deprecated Gradle features were used in this build, making it incompatible wit
h Gradle 7.0.
Use '--warning-mode all' to show the individual deprecation warnings.
See https://docs.gradle.org/6.4.1/userguide/command_line_interface.html#sec:co

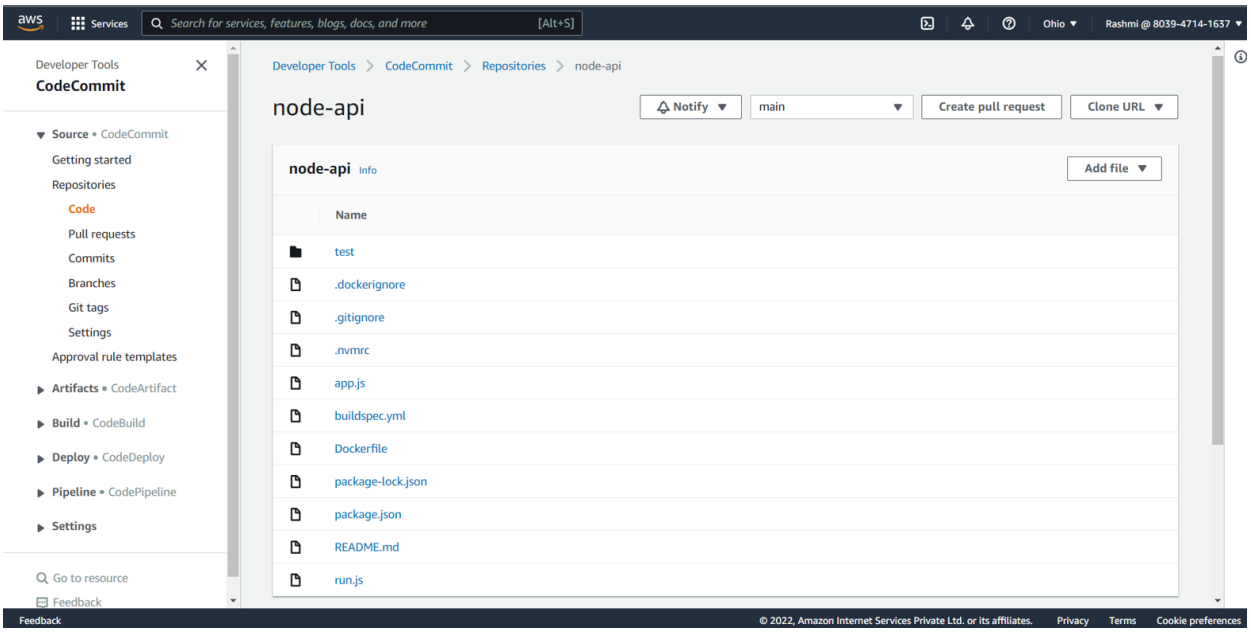

aws
Services
Search for services, features, blogs, docs, and more
[Alt+S]
Ohio
Rashmi @ 8039-4714-1637
Developer Tools
CodePipeline
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Getting started
Pipelines
Pipeline
History
Settings
Settings
Go to resource
Feedback
node-api-pipeline
Notify
Edit
Stop execution
Clone pipeline
Release change
Source Succeeded
Pipeline execution ID: 10a78435-1834-45f0-b09c-9d1e0bc6238e
Source
AWS CodeCommit
Succeeded - 8 hours ago
5741e192
5741e192 Source: Build and push docker image
Disable transition
Build Succeeded
Pipeline execution ID: 10a78435-1834-45f0-b09c-9d1e0bc6238e
Build
AWS CodeBuild
Succeeded - Just now
Details
5741e192 Source: Build and push docker image
Feedback
© 2022, Amazon Internet Services Private Ltd. or its affiliates.
Privacy
Terms
Cookie preferences

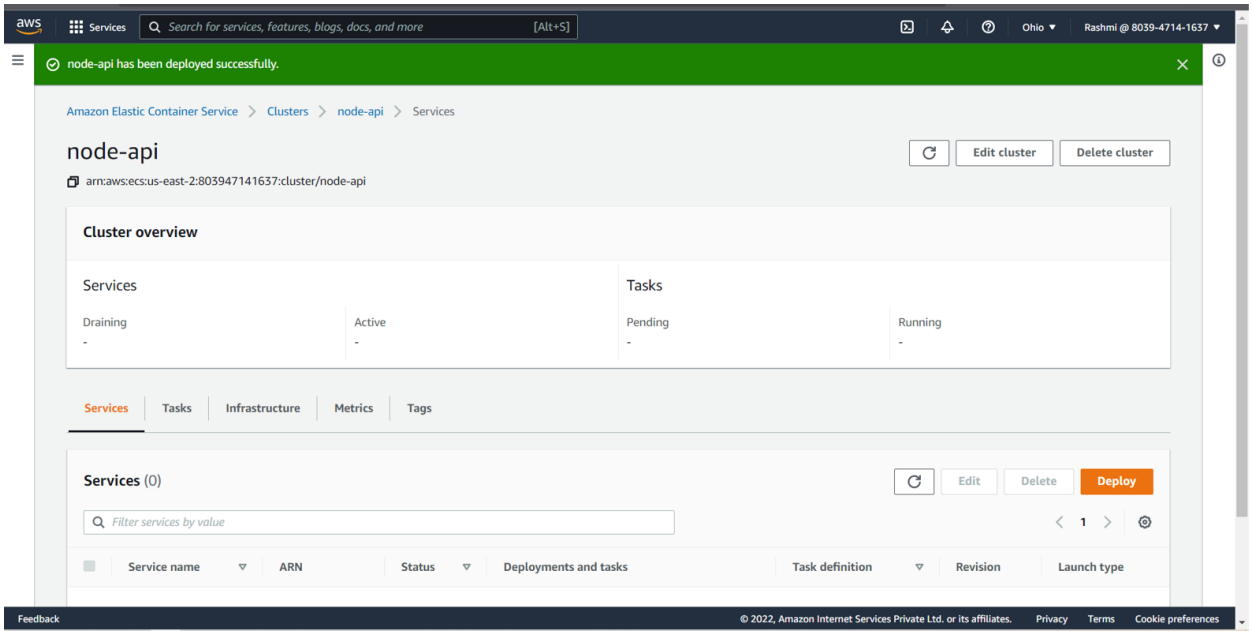


Configuring a codecommit pipeline service to deploy



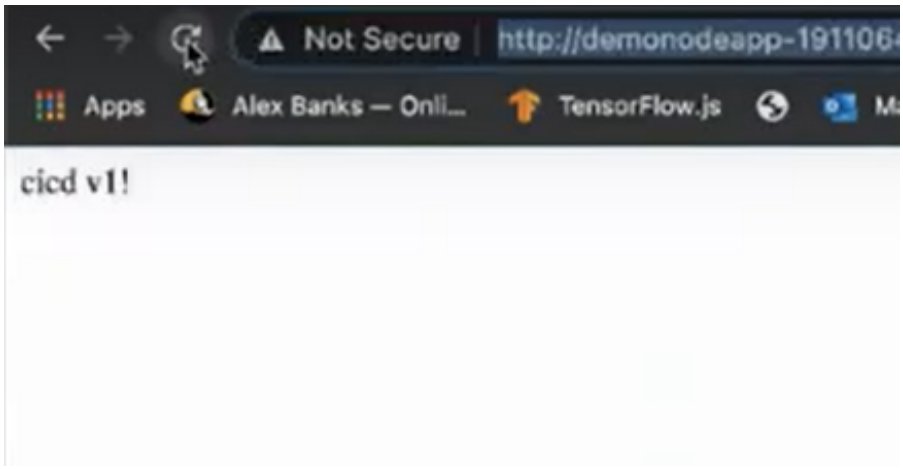
Deployment


Developer Tools
CodeCommit
Source • CodeCommit
Getting started
Repositories
Code
Pull requests
Commits
Branches
Git tags
Settings
Approval rule templates
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Settings
Developer Tools > CodeCommit > Repositories > node-api
node-api
Notify
main
Create pull request
Clone URL
node-api Info
Add file
Name
test
.dockerignore
.gitignore
.nvmrc
app.js
buildspec.yml
Dockerfile
package-lock.json
package.json
README.md
run.js


node-api has been deployed successfully.
Amazon Elastic Container Service > Clusters > node-api > Services
node-api
arn:aws:ecs:us-east-2:803947141637:cluster/node-api
Edit cluster
Delete cluster
Cluster overview
Services
Draining
Active
Tasks
Pending
Running
Services
Tasks
Infrastructure
Metrics
Tags
Services (0)
Edit
Delete
Deploy
Filter services by value
Service name
ARN
Status
Deployments and tasks
Task definition
Revision
Launch type

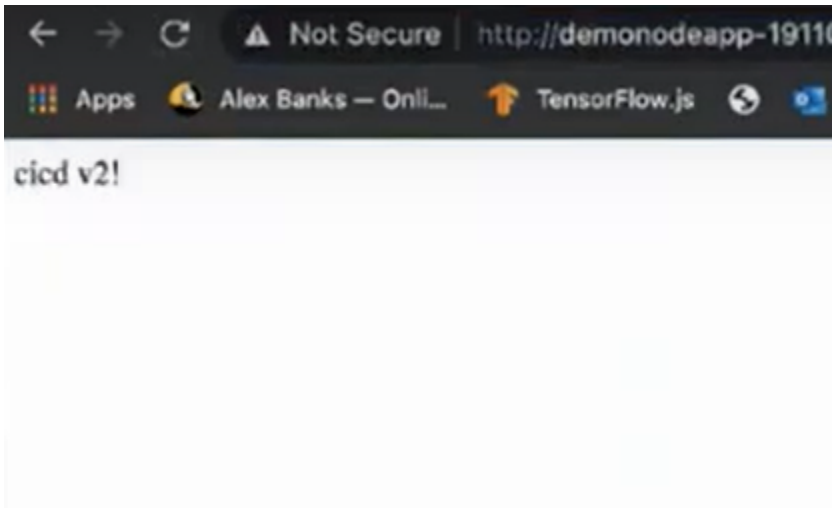
Deployed application

Version v1



Version v2

Any changes in the code can be immediately seen in the web page due to the CI/CD pipeline.



References

(59) Kubernetes Tutorial for Beginners [FULL COURSE in 4 Hours] - YouTube

Getting started with Amazon EKS - Amazon EKS

What is IAM? - AWS Identity and Access Management (amazon.com)

AWS CodePipeline Documentation (amazon.com)

AWS CodeCommit Documentation (amazon.com)

(59) Build a CI/CD pipeline in AWS for container applications - YouTube

(59) CI/CD With AWS ECS + CodePipeline + CodeCommit + CodeBuild | DevOps With AWS Ep7 - YouTube

(59) AWS EKS CI/CD: Cluster Set-up, App containerization & Deployment, CodePipeline| DevOps With AWS Ep10 - YouTube

(59) AWS EKS CI/CD: Cluster Set-up, App containerization & Deployment, CodePipeline| DevOps With AWS Ep10 - YouTube

awscicd/app at main · abohmeed/awscicd (github.com)