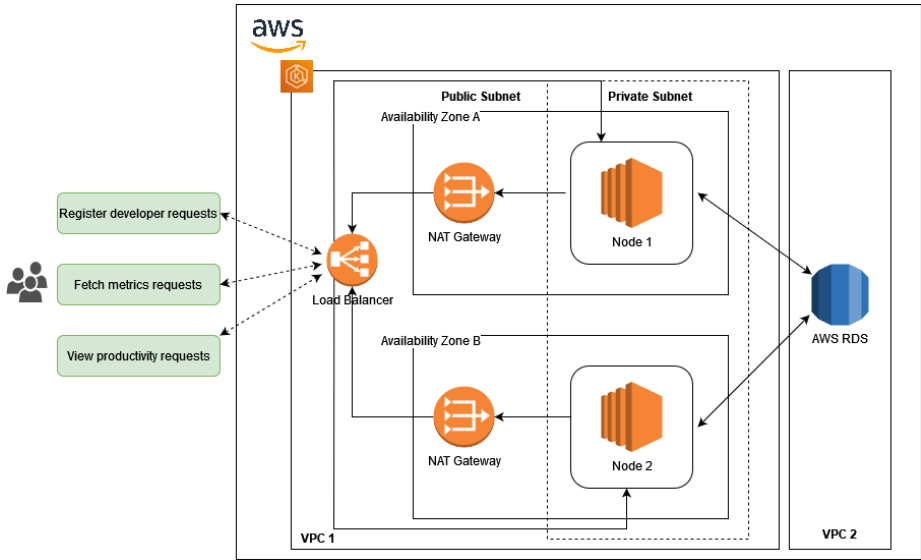
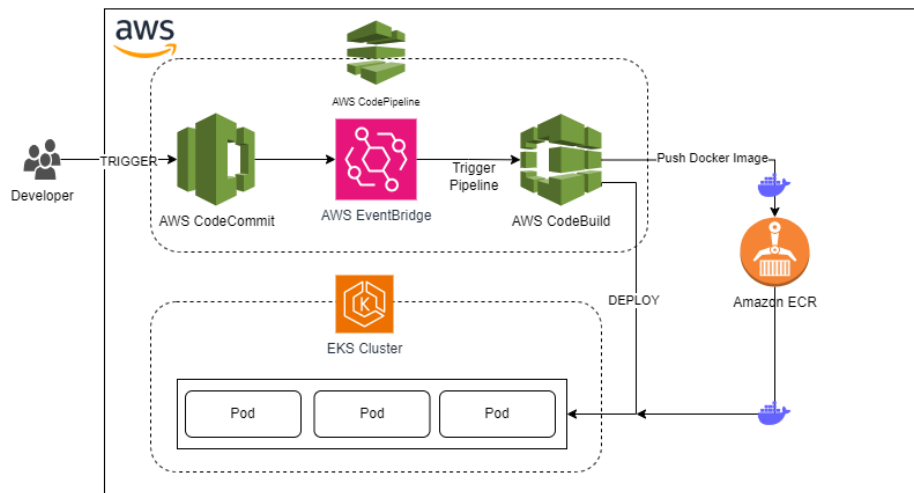


DeveloperIQ Productivity Tracker System

Runbook name	DeveloperIQ System Runbook
Runbook description	<p>Runbook to demonstrate the deployment of developerIQ system which contains three microservices:</p> <ol style="list-style-type: none">Developer Management microservice: This microservice facilitates user-level operations, including registering, retrieving, updating and deleting developer details. It connects to AWS RDS database to store the details.Metrics Management microservice: This microservice fetches the productivity metrics for a respective developer using GitHub REST API and stores them in connected RDS instance. The service fetch three main metrics 'number of commits,' 'number of pull requests' and the 'number of issues.'Productivity/Dashboard Management microservice: Responsible for visualizing a dashboard to easily monitor the productivity of developers. It displays a bar chart to understand the productivity of each developer via metrics and also visualizes their productivity percentage via a pie chart.
Owner	Sahdiya Mariyam Suhan
Version	v 1.0
Version date	12/11/2023
On this page	

Architecture






Support contacts

Expertise level	Team	Contact info
Developer and Owner	Sahdiya Mariyam	sahdiya.20231624@iit.ac.lk

Process

	Step instructions	Execution location	Run environments	Run Instructions	Documentation
1	Create the microservice application for metrics management service	AWS	Amazon SageMaker	<div> <i>i</i> Follow the folder structure according to the folder structure in the source code link attached </div>	https://github.com/Mariyam73/developerIQ-productivity-tracker/tree/main/Source%20Codes Connect your Github account
2	Install and configure AWS CLI	Local Machine	Command Prompt/ Powershell CLI		<div> Install or update the latest version of the AWS CLI - AWS Command Line Interface </div> <div> Configure the AWS CLI - AWS Command Line Interface </div>
3	Install kubectl	Local Machine	Command Prompt/ Powershell CLI		<div> Installing or updating kubectl - Amazon EKS </div>

4	Install aws-iam-authenticator	Local Machine	Command Prompt/ Powershell CLI		Installing aws-iam-authenticator - Amazon EKS
5	Install eksctl	Local Machine	Command Prompt/ Powershell CLI		Getting started with Amazon EKS – eksctl - Amazon EKS
6	Create the EKS Cluster	AWS	AWS CLI in local machine	eksctl create cluster --region ap-southeast-1 --node-type t3.small --nodes 2 --nodes-min 1 --nodes-max 4 --name developer-cc-Cluster --kubeconfig=C:/KubernetesCluster/kube-config.yaml	
7	Create Amazon RDS database instance with MySQL engine	AWS	AWS RDS		Create and Connect to a MySQL Database with Amazon RDS
8	Connect to RDS Workbench	Local Machine	MySQL Workbench	<div>  If the connection does not get established edit the inbound rules to allow all traffic. </div>	Create and Connect to a MySQL Database with Amazon RDS
9	Create ECR repositories to store docker images	AWS	AWS ECR	Create separate repositories for each microservice	Creating a private repository - Amazon ECR
10	Create CodeCommit Repository and clone the codes with the repository	AWS	AWS CodeCommit	Create a dedicated repository for each microservice and create two branches 'dev' and 'prod'	Create an AWS CodeCommit repository - AWS CodeCommit


11	Create CloudFormation Stack to create CodePipeline	AWS	AWS CloudFormation, AWS CodeBuild, AWS CodePipeline	On creation of stack the codepipeline gets created automatically.	Creating a stack on the AWS CloudFormation console - AWS CloudFormation <hr/> CFT Template
12	Deploy the service	Local Machine	Command Prompt/ Powershell CLI	kubectl apply -f deployment.yml	All deployment commands <hr/> Deployment file template

 The [cft file](#) containing configurations for CodePipeline setup should be modified as follows:

- Change the account ID with your account ID
- Change the name of the code repository according to your code repository name

 The [deployment files](#) containing the configurations for deployment should be modified as follows:

- Change the uri of the docker images by copying the uri from your ECR repository

 The [buildspec file](#) containing the commands to build and push docker images should be modified as follows:

- Navigate to ECR repository and select 'View push command'
- Copy the command under 'Retrieve an authentication token and authenticate your Docker client to your registry' and replace the second line under post build commands (`aws ecr get-login-password..`)
- Modify the uri of the repository according to the uri of your repository in the fourth and fifth commands (`docker tag, docker push`)