



LAB MANUAL OF

Advance DevOps Lab

Code: ITL504

Class: TE Information Technology

Semester: V (Rev-2019 ‘C’ Scheme)

Sachin Barahate
Lab Incharge

H.O.D
(Dr. Pradip Mane)

College Vision

- To provide an environment to educate, encourage and explore students by facilitating innovative research, entrepreneurship, opportunities and employability to achieve social and professional goals.

College Mission

- To foster entrepreneurship & strengthen industry institute interaction to enhance career opportunities for the employability of students.
- To encourage collaborations with industries and academic institutes in terms of projects & internships by creating area for Research and Development.
- To build up appropriate moral and ethical skills and to promote holistic development of students through various academic, social and cultural activities

Department Vision

- To impart quality education in the field of Information Technology to meet the challenging needs of the society and industry.

Department Mission

- To provide quality education to students by including Problem Solving, Teamwork and Leadership Skills to achieve their goals in the field of Information Technology.
- To develop skilled IT professionals with moral principles and empower them in lifelong learning.
- To educate students for global development including entrepreneurship, employability and the ability to apply technology to real life problems.

Program Educational Objectives (PEO)

- Graduates will be successful with sound foundation in engineering fundamentals, trending technologies and entrepreneurship.
- Graduates will be able to identify and solve real world problems.
- Graduates will become ingenious and responsible citizens by demonstrating ethics with nurtured professional attitude

Program Specific Outcomes (PSO)

- Develop efficient IT based solutions by applying and integrating various domains like Artificial Intelligence, IoT, Computer Networks and Security to solve real time problems.
- Apply technical knowledge in the field of Information Technology to achieve successful career and to pursue higher studies for future endeavors.

Program Outcomes (POs)

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate complex engineering problems reaching substantiated conclusions using principles of Computer Engineering.
3. **Design / development of solutions:** Design / develop solutions for complex engineering problems and design system components or processes that meet the specified needs with

- appropriate consideration for the society.
4. **Conduct investigations of complex problems:** Use knowledge for the design of experiments, analysis, interpretation of data, and synthesis of the information to provide valid conclusions.
 5. **Modern tool usage:** Create, select and apply appropriate techniques and modern engineering tools, including predictions and modeling to complex engineering activities with an understanding of the limitations.
 6. **The engineer and society:** Apply the knowledge to assess social issues and the responsibilities relevant to engineering practices.
 7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in social and environmental contexts, and demonstrate the knowledge for sustainable development.
 8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
 9. **Individual and teamwork:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
 10. **Communication:** Communicate effectively such as being able to comprehend and write effective reports and design documentation, make effective presentations.
 11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management skills and apply the skills to manage projects effectively.
 12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Lab Objectives

The Lab experiments aims:

1. To understand DevOps practices and cloud native environments to achieve continuous software delivery pipelines and automated operations that address the gap between IT resources and growing cloud complexity.
2. To Use Kubernetes services to structure N-tier applications.
3. To be familiarized with Infrastructure as code for provisioning, compliance, and management of any cloud infrastructure, and service.
4. To understand that security and speed in software development are not inversely-related objectives Internalizing the contribution of tools and automation in DevSecOps
5. To understand various troubleshooting techniques by monitoring your entire infrastructure and business processes
6. To understand how software and software-defined hardware are provisioned dynamically.

Lab Outcomes

On successful completion, of course, learner/student will be able to:

1. To understand the fundamentals of Cloud Computing and be fully proficient with Cloud based DevOps solution deployment options to meet your business requirements
2. To deploy single and multiple container applications and manage application deployments with rollouts in Kubernetes
3. To apply best practices for managing infrastructure as code environments and use terraform to define and deploy cloud infrastructure.
4. To identify and remediate application vulnerabilities earlier and help integrate security in the development process using SAST Techniques.

5. To use Continuous Monitoring Tools to resolve any system errors (low memory, unreachable server etc.) before they have any negative impact on the business productivity
6. To engineer a composition of nano services using AWS Lambda and Step Functions with the Serverless Framework

Prerequisite: Operating System, Linux Administration, Java /Web Application Programming, Software Engineering, Cloud Computing and DevOps Ecosystem.

Hardware & Software Requirements:

Hardware Requirements	Software Requirements	Other Requirements
PC With following Configuration 1. Intel i3 core or above 2. 4 GB RAM or above 3. 500 GB HDD 4. Network interface card	1. Linux / Windows Operating system 2. VIRTUAL BOX/ VMWARE	1. Internet Connection for installing additional packages 2. GitHub account 3. AWS free tier account

Lab /syllabus:

Sr. No.	Module	Detailed Content	Hours	LO Mapping
0	Prerequisite	Knowledge of Linux Operating system, installation and configuration of services and command line basics, Basics of Computer Networks, Software Development Life cycle, Cloud Computing and DevOps Ecosystem.	02	--
I	Introduction to Devops on Cloud	<p>Learn about various cloud services and service providers, also get the brief idea of how to implement DevOps over Cloud Platforms.</p> <ul style="list-style-type: none"> • Introduction to high availability architecture and auto-scaling • Set up the DevOps infrastructure on the cloud • Work and set up IDE on Cloud9 • Deploy projects on AWS using Code Build, CodeDeploy, and CodePipeline <p>Self-Learning Topics: AWS CodeStar</p>	04	LO1
II	Container Orchestration using Kubernetes	<p>In this module, you will learn how Kubernetes automates many of the manual processes involved in deploying, managing, and scaling containerized applications.</p> <ul style="list-style-type: none"> • Install and configure Kubernetes • Spin Up a Kubernetes Cluster • Check the Nodes of Your Kubernetes Cluster • Installing kubectl to manage cluster 	04	LO1, LO2

		<p>and deploy Your First Kubernetes Application</p> <p>Self-Learning Topics:</p> <ul style="list-style-type: none"> • Using Services and Ingresses to Expose Deployments • Perform logging, monitoring, services, and volumes in Kubernetes. 		
III	Infrastructure Automation with Terraform	<p>In this module you will learn, Infrastructure as code for provisioning, compliance, and management of any cloud infrastructure, and service.</p> <ul style="list-style-type: none"> • Introduction to Infrastructure as Code with Terraform • Install, Build, change and Destroy Infrastructure using Terraform. <p>Self-Learning Topics: Terraform</p> <ul style="list-style-type: none"> • Create Resource Dependencies • Provision Infrastructure • Define Input Variables, Query Data with output and store remote state 	04	LO1, LO3
IV	DevSecOps: Static Application Security Testing (SAST)	<p>In this module, you will learn to identify and remediate application vulnerabilities earlier and help integrate security in the development process using tools like SonarQube / Gitlab /</p> <ul style="list-style-type: none"> • Perform static analysis on application source code and binaries. • Spot potential vulnerabilities before deployment • Analysis of java / web-based project • Jenkins SonarQube / Gitlab Integration <p>Self-Learning Topics: Snyk, OWASP ZAP, Analysis Core Plugin</p>	04	LO1, LO4
V	DevSecOps: Continuous Monitoring	<p>In this module, you will learn to detect, report, respond to the attacks and issues which occur within the infrastructure.</p> <ul style="list-style-type: none"> • Introduction to Continuous Monitoring • Introduction to Nagios • Installing Nagios • Nagios Plugins (NRPE) and Objects • Nagios Commands and Notification • Monitoring of different servers using Nagios <p>Self-Learning Topics: Splunk, Snort, Tenable</p>	04	LO1, LO5
VI	NoOps: Serverless	Computing	04	LO1, LO6

	<p>code and deploy it without ever needing to configure or manage underlying servers.</p> <ul style="list-style-type: none"> • AWS Lambda - Overview and Environment Setup . • Building and Configuring the Lambda function (NODEJS/PYTHON/JAVA) • Creating & Deploying using AWS Console/CLI • Creating & Deploying using Serverless Framework <p>Self-Learning Topics: AWS Lambda</p> <ul style="list-style-type: none"> • Create a REST API with the Serverless Framework 	
--	--	--

Textbooks:

1. AWS Certified SysOps Administrator Official Study Guide: Associate Exam by Stephen Cole (Author), Gareth Digby (Author), Chris Fitch (Author), Steve Friedberg (Author), Shaun Qual
2. AWS Certified Solutions Architect Official Study Guide: Associate Exam by Joe Baron
3. Terraform: Up & Running - Writing Infrastructure as Code, Second Edition by Yevgeniy Brikman , O'Reilly
4. Kubernetes: Up and Running - Dive into the Future of Infrastructure, Second Edition by Brendan Burns,O'Reilly
5. Going Serverless with AWS Lambda: Leveraging the latest services from the AWS cloud by Ajay Pherwani , Shroff/X-Team;
6. Learning Nagios, Packt Publishing.

References:

1. Learning Aws - Second Edition: Design, build, and deploy responsive applications using AWS by Amit Shah Aurobindo Sarkar
2. Mastering Aws Lambda by Yohan Wadia Udita Gupta

List of Experiments

Sr. No	Experiment Title	Lab Outcome
1	To understand the benefits of Cloud Infrastructure and Setup AWS Cloud9 IDE, Launch AWS Cloud9 IDE and Perform Collaboration Demonstration.	LO1
2	To Build Your Application using AWS Code Build and Deploy on S3 / SEBS using AWS Code Pipeline, deploy Sample Application on EC2 instance using AWS Code Deploy.	LO1
3	To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud Platforms.	LO1, LO2
4	To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy Your First Kubernetes Application.	LO1, LO2
5	To understand terraform lifecycle, core concepts/terminologies and install it on a Linux Machine.	LO1, LO3
6	To Build, change, and destroy AWS / GCP /Microsoft Azure/ Digital Ocean infrastructure Using Terraform.	LO1, LO3
7	To understand Static Analysis SAST process and learn to integrate Jenkins SAST to Sonar Qube/Git Lab.	LO1, LO4
8	Create a Jenkins CICD Pipeline with Sonar Qube / Git Lab Integration to perform a static analysis of the code to detect bugs, code smells, and security vulnerabilities on a sample Web / Java / Python application.	LO1, LO4
9	To Understand Continuous monitoring and Installation and configuration of Nagios Core, Nagios Plugins and NRPE (Nagios Remote Plugin Executor) on Linux Machine.	LO1, LO5
10	To perform Port, Service monitoring, Windows/Linux server monitoring using Nagios.	LO1, LO5
11	To understand AWS Lambda, its workflow, various functions and create your first Lambda functions using Python / Java / Nodejs.	LO1, LO6
12	To create a Lambda function which will log “An Image has been added” once you add an object to a specific bucket in S3.	LO1, LO6

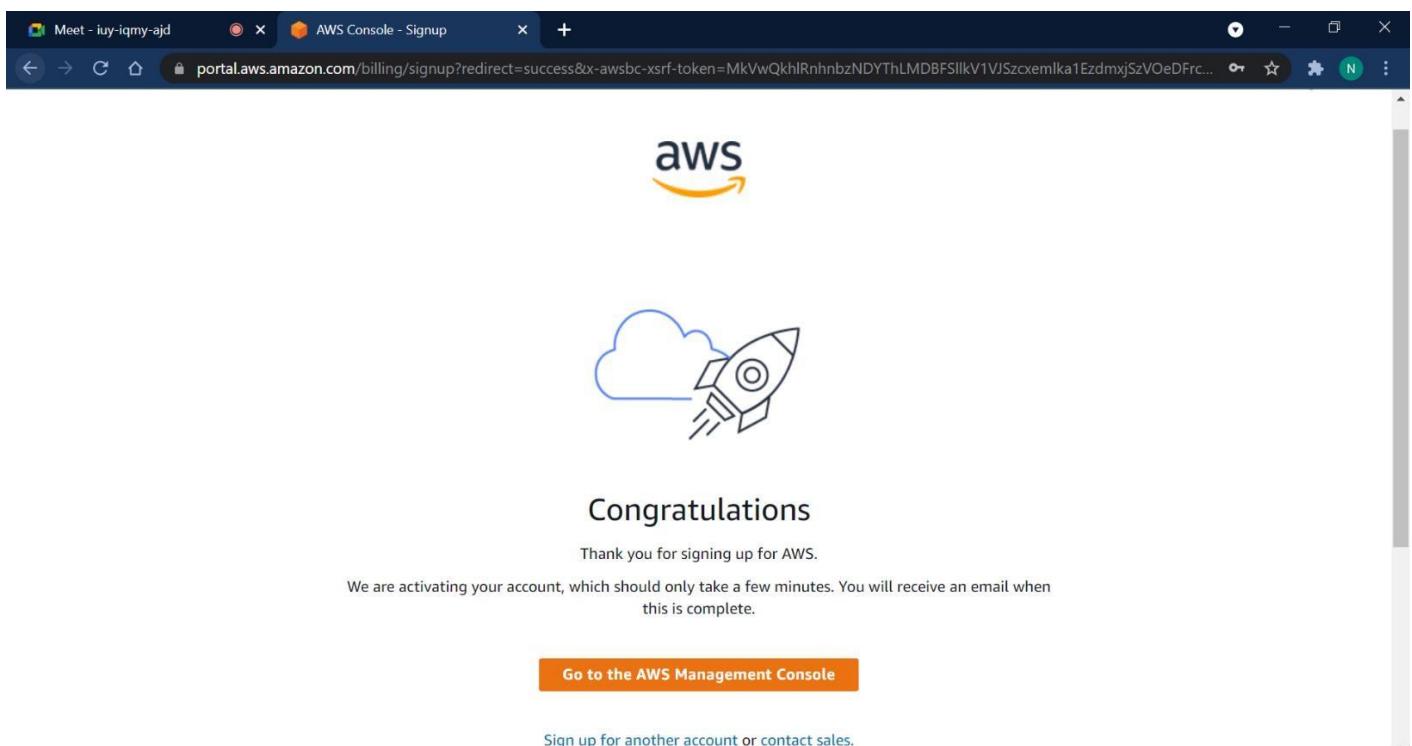
Experiment no. 1

Aim: To understand the benefits of Cloud Infrastructure and Setup AWS Cloud9 IDE, Launch AWS Cloud9 IDE and Perform Collaboration Demonstration.

Theory:

Cloud infrastructure is a term used to describe the components needed for cloud computing, which includes hardware, abstracted resources, storage, and network resources. Think of cloud infrastructure as the tools needed to build a cloud. In order to host services and applications in the cloud, you need cloud infrastructure. Cloud infrastructure is made up of several components, each integrated with one another into a single architecture supporting business operations. A typical solution may be composed of hardware, virtualization, storage, and networking components. As a term, cloud infrastructure can be used to describe a complete cloud computing system - once all the pieces are put together - as well as the individual technologies themselves.

Results: Account created



2. AWS Cloud9

The screenshot shows the AWS Cloud9 homepage. At the top, there's a search bar and navigation links for services, user arshadsamani, and support. The main heading is "AWS Cloud9" with the subtext "A cloud IDE for writing, running, and debugging code". Below this, a paragraph explains the service's purpose: "AWS Cloud9 allows you to write, run, and debug your code with just a browser. With AWS Cloud9, you have immediate access to a rich code editor, integrated debugger, and built-in terminal with preconfigured AWS CLI. You can get started in minutes and no longer have to spend the time to install local applications or configure your development machine." To the right, there's a call-to-action box titled "New AWS Cloud9 environment" with a "Create environment" button. On the left, under "Developer Tools", there's a section titled "How it works" with a sub-section "Create an AWS Cloud9 development environment on a new Amazon EC2 instance or reuse an existing one with the AWS Cloud9 Quick Start AMI". The bottom of the page includes standard footer links for feedback, language selection (English US), copyright notice (© 2008 - 2021), and privacy policy.

3. Creating Environment

The screenshot shows the "Name environment" step of the AWS Cloud9 creation wizard. The left sidebar shows "Step 1: Name environment" is selected, with "Configure settings" as the next step. The main form is titled "Environment name and description". It has a "Name" field containing "npm-user" and a "Description - Optional" field containing "This will appear on your environment's card in your dashboard. You can update it at any time in your environment settings." The description field has a character limit of 200 characters and a note that it's optional. At the bottom, there are "Cancel" and "Next step" buttons, with the "Next step" button being orange. The bottom of the page includes standard footer links for feedback, language selection (English US), copyright notice (© 2008 - 2021), and privacy policy.

Meet - iuy-iqmy-ajd Create a new environment

us-east-2.console.aws.amazon.com/cloud9/home/create

AWS Services Search for services, features, marketplace products, and docs [Alt+S]

arshadsamani Ohio Support

AWS Cloud9 > Environments > Create environment

Step 1 Name environment

Step 2 Configure settings

Step 3 Review

Review

Environment name and settings

Name: npm-user

Description: This is my first nodejs project in AWS

Environment type: EC2

Instance type: t2.micro

Subnet:

Platform: Amazon Linux 2 (recommended)

Cost-saving settings: After 30 minutes (default)

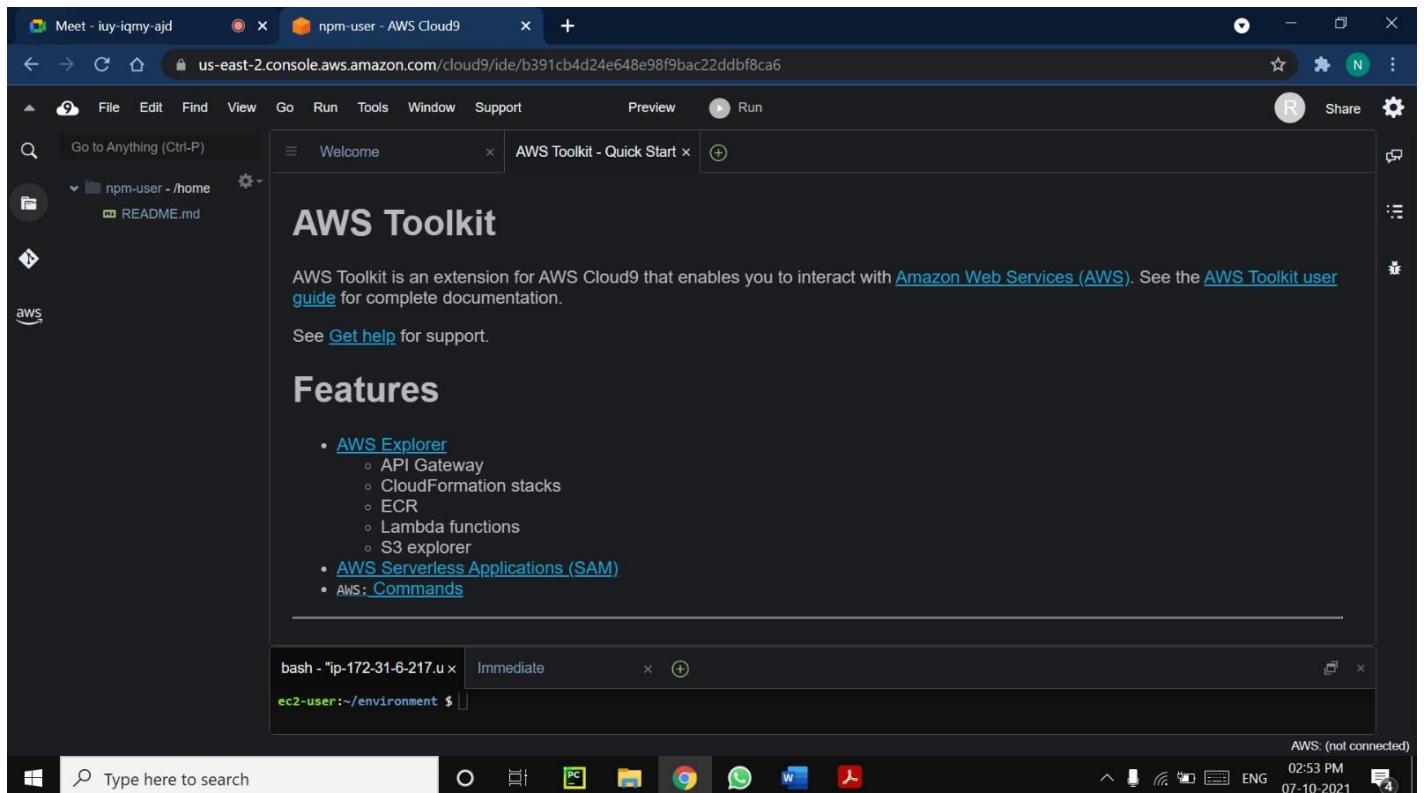
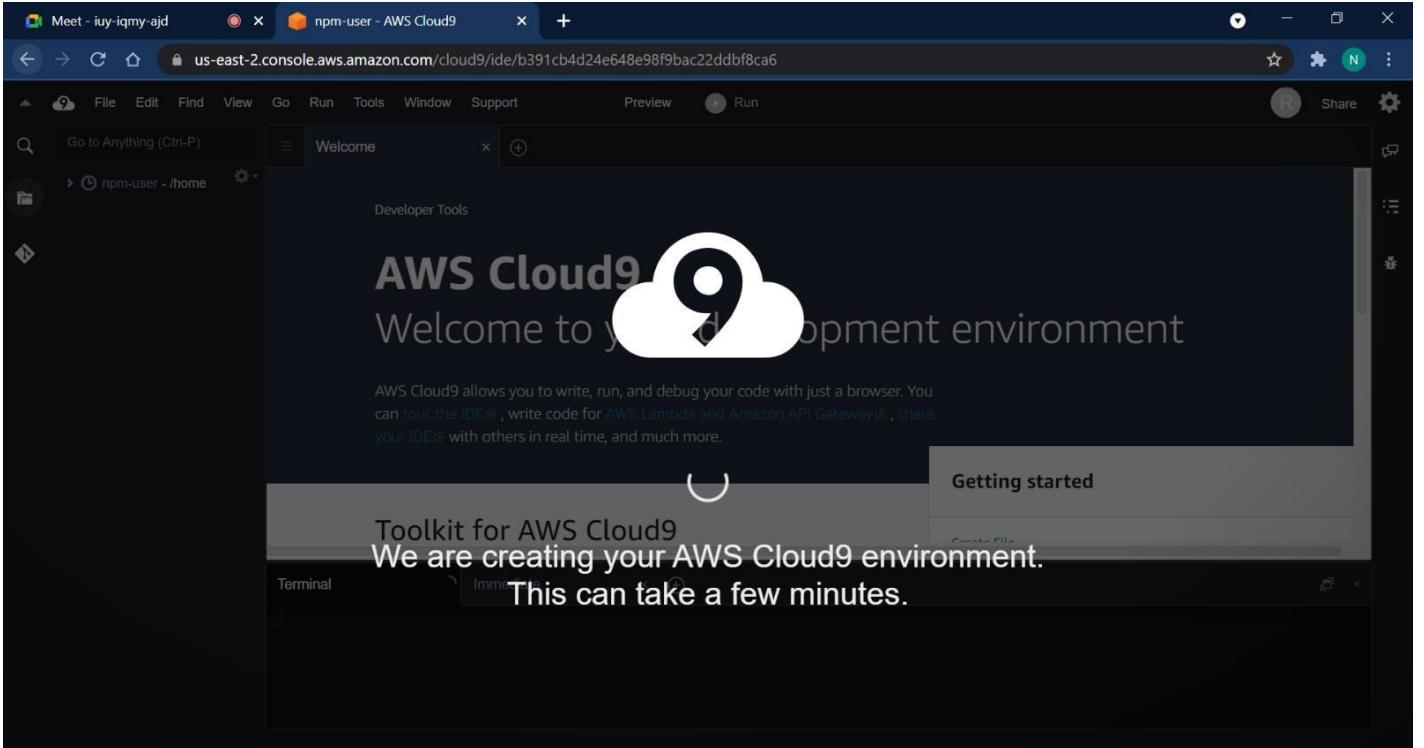
IAM role: AWSServiceRoleForAWSCloud9 (generated)

We recommend the following best practices for using your AWS Cloud9 environment

- Use **source control and backup** your environment frequently. AWS Cloud9 does not perform automatic backups.
- Perform regular **updates of software** on your environment. AWS Cloud9 does not perform automatic updates on your behalf.
- Turn on **AWS CloudTrail** in your AWS account to track activity in your environment. [Learn more](#)
- Only share your environment with **trusted users**. Sharing your environment may put your AWS access credentials at risk. [Learn more](#)

Cancel Previous step **Create environment**

Feedback English (US) © 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences



Conclusion: We successfully created an account on AWS and also created an environment on Cloud9 IDE.

Experiment No :2

Title: To Build Your Application using AWS Code Build and Deploy on S3 / SEBS using AWS Code Pipeline, deploy Sample Application on EC2 instance using AWS Code Deploy.

Solution: In this experiment I am Building a Java application.

Step 1: Create the source code

1. In an empty directory on your local computer or instance, create this directory structure.

(root directory name)

```
``-- src
    |-- main
        |   ``-- java
        ``-- test
            ``-- java
```

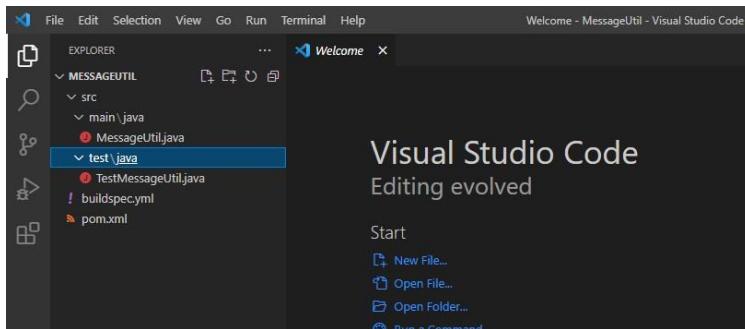
2. Using a text editor of your choice, create this file, name it MessageUtil.java, and then save it in the src/main/java directory.

This class file creates as output the string of characters passed into it. The MessageUtil constructor sets the string of characters. The printMessage method creates the output. The salutationMessage method outputs Hi! followed by the string of characters.

3. Create this file, name it TestMessageUtil.java, and then save it in the /src/test/java directory.
4. Create this file, name it pom.xml, and then save it in the root (top level) directory.

Step 2: Create the buildspec file.

Create this file, name it buildspec.yml, and then save it in the root (top level) directory.



Step 3: Create two S3 buckets

Name	AWS Region	Access	Creation date
first-demo-input-bucket	US East (Ohio) us-east-2	Bucket and objects not public	October 8, 2021, 19:21:49 (UTC+05:30)
first-demo-output-bucket	US East (Ohio) us-east-2	Bucket and objects not public	October 8, 2021, 19:22:34 (UTC+05:30)

Step 4: Upload the source code and the buildspec file

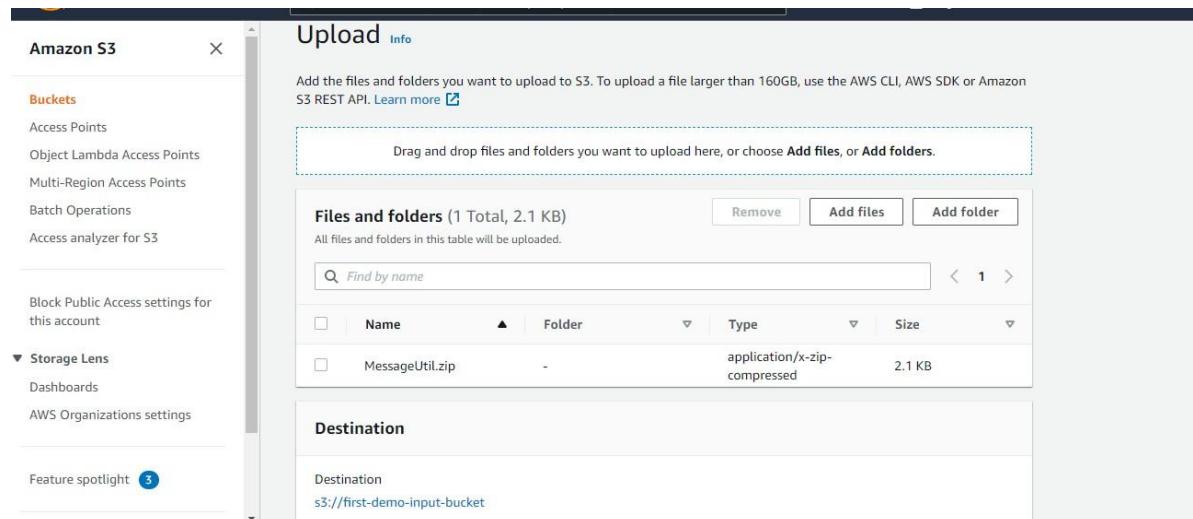
In this step, you add the source code and build spec file to the input bucket.

Using your operating system's zip utility, create a file named `MessageUtil.zip` that includes `MessageUtil.java`, `TestMessageUtil.java`, `pom.xml`, and `buildspec.yml`.

The `MessageUtil.zip` file's directory structure must look like this.

MessageUtil.zip

```
-- pom.xml  
|-- buildspec.yml  
`-- src  
    |-- main  
    |   |-- java  
    |       |-- MessageUtil.java  
    `-- test  
        |-- java  
            `-- TestMessageUtil.java
```



Step 5: Create the build project

In the CodeBuild service create new build Project

Create build project

Project configuration

Project name
cloud-for-developer-java-build

A project name must be 2 to 255 characters. It can include the letters A-Z and a-z, the numbers 0-9, and the special characters - and _.

Description - *optional*

Build badge - *optional*
 Enable build badge

Enable concurrent build limit - *optional*
Limit the number of allowed concurrent builds for this project.
 Restrict number of concurrent builds this project can start

Source

Add source

Source 1 - Primary

Source provider
Amazon S3

Bucket
first-demo-input-bucket

S3 object key or S3 folder
MessageUtil.zip

Source version - *optional* Info
Enter the version ID of the object that represents the build input ZIP file.

Environment

Environment image
 Managed image Use an image managed by AWS CodeBuild
 Custom image Specify a Docker image

Operating system
Amazon Linux 2

The programming language runtimes are now included in the standard image of Ubuntu 18.04, which is recommended for new CodeBuild projects created in the console. See Docker Images Provided by CodeBuild for details.

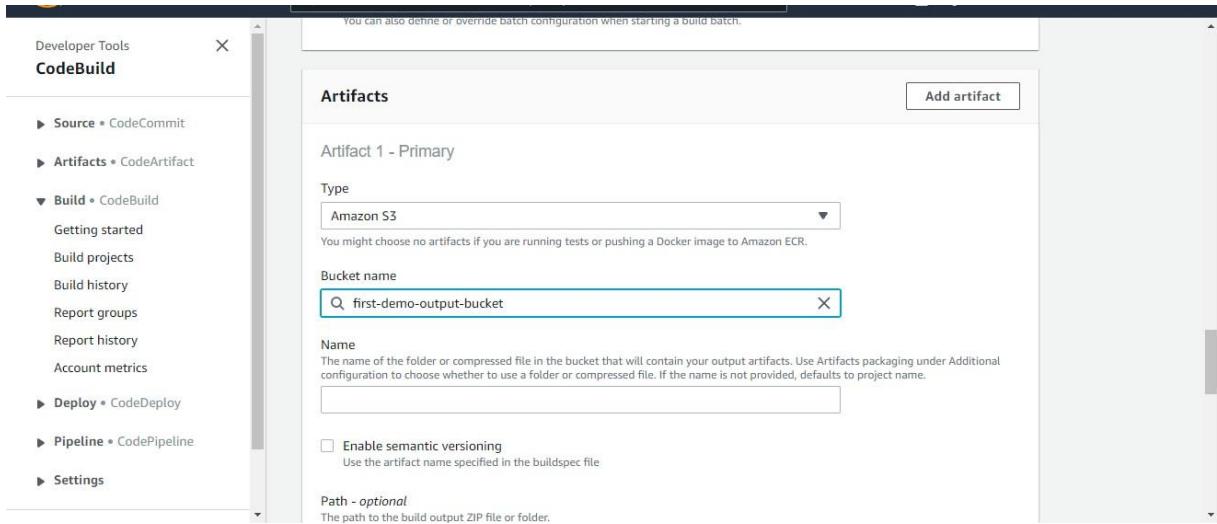
Runtime(s)
Standard

Image
aws/codebuild/amazonlinux2-x86_64-standard:2.0

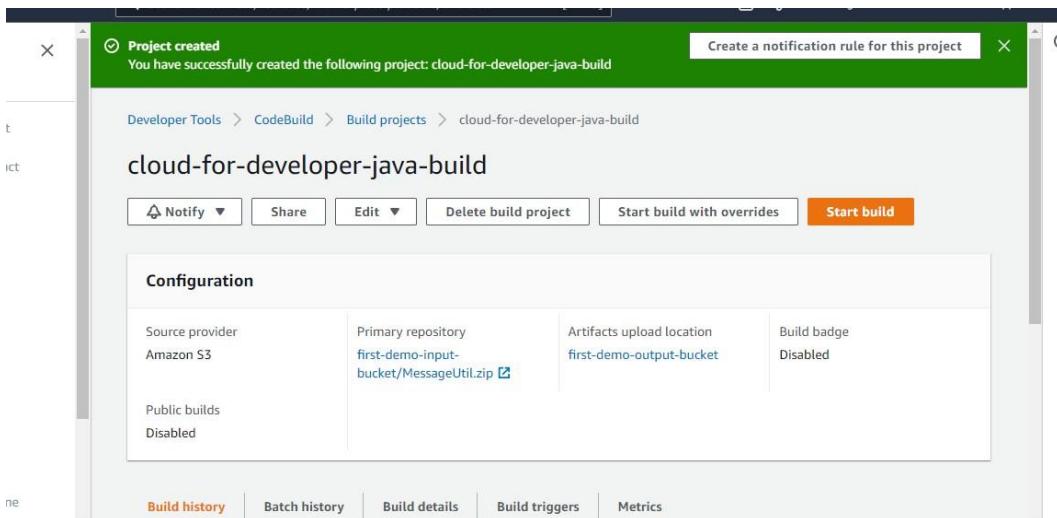
Image version
Always use the latest image for this runtime version

Environment type
Linux

Privileged
 Enable this flag if you want to build Docker images or want your builds to get elevated privileges



Step 6: Start Build by clicking on start Build



Step 7: Check Status success or fail

The screenshot shows the AWS CodeBuild build status page for build ID 38fb0675be0. A green header bar at the top indicates "Build started" and "You have successfully started the following build: cloud-for-developer-java-build:39c362d0-1c5a-4506-a219-38fb0675be0". Below this, the build ID "38fb0675be0" is displayed. There are two buttons: "Stop build" (disabled) and "Retry build". The "Build status" section contains the following details:

Status	Initiator	Build ARN
Succeeded	root	arn:aws:codebuild:us-east-2:960934202107:build/cloud-for-developer-java-build:39c362d0-1c5a-4506-a219-38fb0675be0
Resolved source version	Start time	End time
first-demo-input-bucket/MessageUtil.zip	Oct 8, 2021 7:51 PM (UTC+5:30)	Oct 8, 2021 7:52 PM (UTC+5:30)
Build number		
2		

- **Deploy code using CodeDeploy**

Step 1: Select Sample Deployment.

The screenshot shows the "Get started with AWS CodeDeploy" wizard. At the top, there is a message: "You have opted out of the updated AWS CodeDeploy console experience. Give us feedback on why you are using the old experience." Below this is a blue button: "Return to the new console experience". On the left, there are two navigation links: "Step 1: Get started" and "Step 2: Choose a deployment type". The main content area is titled "Get started with AWS CodeDeploy" and includes a question mark icon. It describes AWS CodeDeploy as helping to quickly deploy applications to Amazon EC2 instances or on-premises instances. It suggests starting by creating a deployment using a sample application or skipping to a custom deployment. Two options are shown:

- Sample deployment**
Recommended for new AWS CodeDeploy users.
- Custom deployment**
Recommended if you already have instances and an application to deploy.

At the bottom right, there are "Cancel" and "Next" buttons.

Step 2: Choose Deployment Type.

The screenshot shows the AWS CodeDeploy Management console at the URL <https://console.aws.amazon.com/codedeploy/home?region=us-east-1#/first-run/deployment-type>. The page title is "Choose a deployment type". A sidebar on the left lists steps: Step 1: Get started, Step 2: Choose a deployment type (which is selected), Step 3: Create blue/green deployment. The main content area has two options: "Blue/green deployment" (selected) and "In-place deployment". The "Blue/green deployment" section describes replacing instances in the deployment group with new ones. The "In-place deployment" section describes updating instances with the latest revision while keeping them online. At the bottom are "Cancel", "Previous", and "Next" buttons. The status bar at the bottom includes links for Feedback, English (US), Privacy Policy, and Terms of Service.

Step 3: Configure Instances

The screenshot shows the AWS CodeDeploy Management console at the URL <https://console.aws.amazon.com/codedeploy/home?region=us-east-1#/first-run/instance-settings>. The page title is "Configure instances". A sidebar on the left lists steps: Step 1: Get started, Step 2: Choose a deployment type, Step 3: Configure instances (selected), Step 4: Name your application, Step 5: Select a revision, Step 6: Create a deployment group, Step 7: Select a service role, Step 8: Choose a deployment configuration, Step 9: Review deployment details. The main content area shows configuration options: Operating system (Amazon Linux selected, Windows Server available), Instance type (t2.micro selected), Key pair name (Select an existing key pair name dropdown), and Tag key and value (Name: CodeDeployDemo, Value:). Below these are instructions to launch instances via AWS CloudFormation. At the bottom are "Launch instances", "Required", "Cancel", "Previous", "Skip", and "Next" buttons.

Step 4: Name Your Application.

The screenshot shows the AWS CodeDeploy Management console at the URL <https://console.aws.amazon.com/codedeploy/home?region=us-east-1#/first-run/application>. The page title is "Name your application". A sidebar on the left lists steps from 1 to 9. Step 4 is highlighted. The main form has an "Application name*" field containing "DemoApplication". Below it is a "Required" label and buttons for "Cancel", "Previous", and "Next". At the bottom, there are links for "Feedback", "English (US)", and copyright information.

Step 5: Create Deployment Group

The screenshot shows the AWS CodeDeploy Management console at the URL <https://console.aws.amazon.com/codedeploy/home?region=us-east-1#/first-run/deployment-group>. The page title is "Create a deployment group". A sidebar on the left lists steps from 1 to 9. Step 6 is highlighted. The main form has a "Deployment group name*" field containing "DemoTest". Below it is a "Add instances" section with a table for "Search by tags". The table has two rows: row 1 with "Name" as key and "CodeDeployDemo" as value, and row 2 which is empty. A note below the table says "Total matching Amazon EC2 instances: 3".

Step 6: Select Service Role

The screenshot shows the AWS CodeDeploy console interface. The URL is https://console.aws.amazon.com/codedeploy/home?region=us-east-1#/first-run/service-role. The top navigation bar includes links for Dashboard, Checklist, Gotowebinars, Google Analytics, and Continuous Deployments. The main content area has a message: "You have opted out of the updated AWS CodeDeploy console experience. Give us feedback on why you are using the old experience." Below this is a "Return to the new console experience" button. To the left, a sidebar lists steps from 1 to 9. The current step, "Step 7: Select a service role", is highlighted. The main form asks to "Select a service role that allows AWS CodeDeploy to work with other dependent AWS services on your behalf during a deployment." It features two dropdown menus: "Service role*" set to "Use an existing service role." and "Role name*" set to "CodeDeployRole". A note says "*Required". At the bottom are "Cancel", "Previous", and a large blue "Next" button.

Step 7: Choose Deployment Configuration

The screenshot shows the "Choose a deployment configuration" step. It starts with a general description: "A deployment configuration determines the rate at which instances are deployed to and specifies the number or percentage of instances that must remain available at any time during a deployment. Choose from a list of default deployment configurations or create a custom configuration." Two radio buttons are shown: "Use a default deployment configuration" (selected) and "Create a custom deployment configuration". Below are three configuration options: "One at a time" (selected), "Half at a time", and "All at once". Each option has a "Deployment behavior:" section and an "Example:" section. The "One at a time" example states: "You deploy your application to three instances. This configuration will deploy to one instance at a time." It includes two radio buttons: "Succeeds if all three instances succeed" (selected) and "Fails after any instance fails". The "Select" button is located below the examples. The "Half at a time" example states: "You deploy your application to three instances. This configuration will deploy to one instance at a time." It includes two radio buttons: "Succeeds if two or more instances succeed" (selected) and "Fails if two or more instances fail". The "Select" button is located below the examples. The "All at once" section is partially visible.

Step 8: Deploy Application

You have opted out of the updated AWS CodeDeploy console experience. Give us feedback on why you are using the old experience.

[Return to the new console experience](#)

Deployments

View, diagnose, and manage your deployments.

[Create deployment](#)

Deployment ID	Type	Compute ...	Application	Deployme...	Revision L...	Start...	End ...	Initiating event	Status	Actions
d-VX1LH3UNY	In-place	EC2/On-pr...	DemoAppl...	DemoFleet	s3://aws-c...	Mar ...			Created	Stop

Details

Deployment ID: d-VX1LH3UNY
Initiated by: user
Deployment configuration: CodeDeployDefault.OneAtATime
Minimum healthy hosts: All instances except 1
Revision location: s3://aws-codedeploy-us-east-1/samples/latest/SampleApp_1_linux.zip

Instances

Installing application on your instances
0 of 3 instances updated
[View all instances](#)

Step 9: Check Status

→ <https://console.aws.amazon.com/codedeploy/home?region=us-east-1#/deployments/d-VX1LH3UNY>

Apps e! Dashboard < Edureka... Checklist Gotowebinars Google Analytics Continuous Deploy...

[Return to the new console experience](#)

Deployment: d-VX1LH3UNY

✓ Deployment Succeeded

Deployment progress

Installing application on your instances

3 of 3 instances updated

► Deployment details

▼ Instance activity

Instance ID	Start time	End time	Duration	Status	Most recent event	Events
I-0d8a70844b4594537	Mar 8, 2019 2:56:58 PM UTC	Mar 8, 2019 2:57:06 PM UTC	8 secs	Succeeded	ValidateService	View event

Result: The CodeBuild and CodeDeploy operation is successfully performed. This Experiment helped me to understand the basic of AWS cloud9 and other services of AWS.

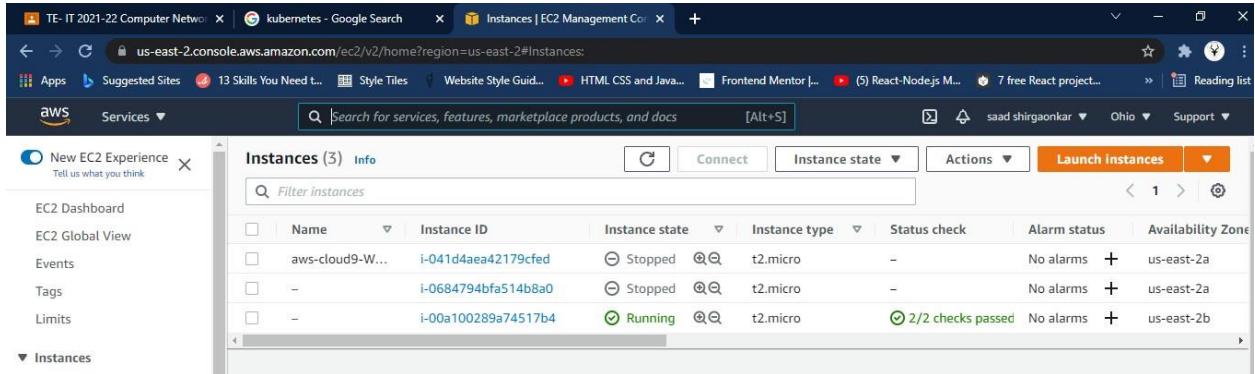
Conclusion: All instance to demonstrate the CodeBuild and CodeDeploy were successfully created and executed.

Experiment No: 3

Title: To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud Platforms.

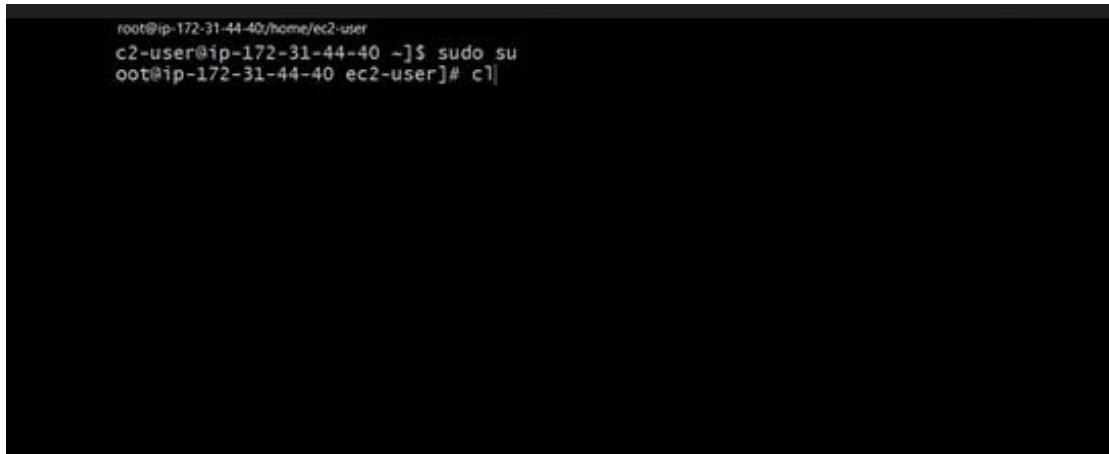
Solution:

Step 1: Create a new or use an existing Cloud instance.



	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
<input type="checkbox"/>	aws-cloud9-W...	i-041d4aea42179cfed	Stopped	t2.micro	-	No alarms	+ us-east-2a
<input type="checkbox"/>	-	i-0684794bfa514b8a0	Stopped	t2.micro	-	No alarms	+ us-east-2a
<input type="checkbox"/>	-	i-00a100289a74517b4	Running	t2.micro	2/2 checks passed	No alarms	+ us-east-2b

Step 2: Take Super user Access. Type Sudo su.



```
root@ip-172-31-44-40:~$ sudo su
root@ip-172-31-44-40 ~]$
```

Step 3: Type the shown Command

```

root@ip-172-31-44-40/home/ec2-user#
root@ip-172-31-44-40 ec2-user]# free -h
              total        used        free      buff/cache   available
m:       3.9G       89M       3.6G      408K      184M       3.6G
ap:          0B         0B         0B
root@ip-172-31-44-40 ec2-user]# swapoff -a
root@ip-172-31-44-40 ec2-user]# cat <>EOF > /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF

```

Step 4: Installation of Docker and enable docker

```

net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF
root@ip-172-31-44-40 ec2-user]# yum install docker
Last metadata expiration check: 0:00:00 ago
Processing Dependencies: ext4, suggestions, langpacks, priorities, update-motd
  solving dependencies
> Running transaction check
--> Package docker.x86_64 0:19.03.13ce-1.amzn2 will be installed
> Processing Dependency: runc >= 1.0.0 for package: docker-19.03.13ce-1.amzn2.x86_64
> Processing Dependency: containerd >= 1.3.2 for package: docker-19.03.13ce-1.amzn2.x86_64
> Processing Dependency: pigz for package: docker-19.03.13ce-1.amzn2.x86_64
> Processing Dependency: libcgroup for package: docker-19.03.13ce-1.amzn2.x86_64
> Running transaction check
--> Package containerd.x86_64 0:1.4.1-2.amzn2 will be installed
--> Package libcgroup.x86_64 0:0.41-21.amzn2 will be installed
--> Package pigz.x86_64 0:2.3.4-1.amzn2.0.1 will be installed
--> Package runc.x86_64 0:1.0.0-0.1.20200826.gitff819c7.amzn2 will be installed
> Finished Dependency Resolution

dependencies Resolved

=====
package           Arch      Version           Repository      Size
=====
stalling:
  docker           x86_64    19.03.13ce-1.amzn2      amzn2extra-docker  37 M
stalling for dependencies:
  containerd        x86_64    1.4.1-2.amzn2          amzn2core-docker   24 M
  libcgroup         x86_64    0.41-21.amzn2         amzn2core          66 k
  libt2k            x86_64    2.3.4-1.amzn2.0.1     amzn2core          81 k
  runc              x86_64    1.0.0-0.1.20200826.gitff819c7.amzn2 3.7 M

transaction Summary:
=====
stall 1 Package (>4 Dependent packages)
total download size: 65 M
stalled size: 270 M
this ok [y/d/N]: y
unloading packages:
=====
stalled size: 270 M
this ok [y/d/N]: y
unloading packages:
=====
(5): libcgroup-0.41-21.amzn2.x86_64.rpm | 66 kB 00:00:00
(5): pigz-2.3.4-1.amzn2.0.1.x86_64.rpm | 81 kB 00:00:00
(5): containerd-1.4.1-2.amzn2.x86_64.rpm | 24 MB 00:00:00
(5): runc-1.0.0-0.1.20200826.gitff819c7.amzn2.x86_64.rpm | 3.7 MB 00:00:00
(5): docker-19.03.13ce-1.amzn2.x86_64.rpm | 37 MB 00:00:01
=====
total
running transaction check
running transaction test
transaction test succeeded
running transaction
installing : runc-1.0.0-0.1.20200826.gitff819c7.amzn2.x86_64
installing : containerd-1.4.1-2.amzn2.x86_64
installing : pigz-2.3.4-1.amzn2.0.1.x86_64
installing : libcgroup-0.41-21.amzn2.x86_64
installing : docker-19.03.13ce-1.amzn2.x86_64
verifying : docker-19.03.13ce-1.amzn2.x86_64
verifying : libcgroup-0.41-21.amzn2.x86_64
verifying : pigz-2.3.4-1.amzn2.0.1.x86_64
verifying : containerd-1.4.1-2.amzn2.x86_64
verifying : runc-1.0.0-0.1.20200826.gitff819c7.amzn2.x86_64
stalled:
  docker.x86_64 0:19.03.13ce-1.amzn2
dependency Installed:
  containerd.x86_64 0:1.4.1-2.amzn2  libcgroup.x86_64 0:0.41-21.amzn2  pigz.x86_64 0:2.3.4-1.amzn2.0.1  runc.x86_64 0:1.0.0-0.1.20200826.gitff819c7.amzn2
complete!
root@ip-172-31-44-40 ec2-user]# systemctl enable docker
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr/lib/systemd/system/docker.service.

```

Step 5: Configure Kubernetes packages.

```
root@ip-172-31-44-40:~# cat <<EOF > /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-e17-x86_64
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
EOF
root@ip-172-31-44-40:~#
```

Step 6: installation of Kubelet, kubeadm, kubectl.

Step 7: initialize Kubelet.

```
[root@ip-172-31-44-40 ec2-user]# systemctl status kubelet
● kubelet.service - kubelet: The Kubernetes Node Agent
   Loaded: loaded (/usr/lib/systemd/system/kubelet.service; disabled; vendor preset: disabled)
   Active: inactive (dead)
     Docs: https://kubernetes.io/docs/
[root@ip-172-31-44-40 ec2-user]# kubeadm init
[07 11:09:02.904822 5366 configset.go:348] WARNING: kubeadm cannot validate component configs for API groups [kubelet.config.k8s.io kubeproxy.config.k8s.io]
[init] Using Kubernetes version: v1.19.4
[init] Running pre-flight checks
[WARNING IsDockerSystemdCheck] detected "cgroupfs" as the Docker cgroup driver. The recommended driver is "systemd". Please follow the guide at http://kubernetes.io/docs/setup/cri/
[WARNING FileExisting-tc] tc not found in system path
[WARNING Service-kubelet] kubelet service is not enabled, please run 'systemctl enable kubelet.service'
[flight] Pulling images required for setting up a Kubernetes cluster
[flight] This might take a minute or two, depending on the speed of your internet connection
[flight] You can also perform this action in beforehand using 'kubeadm config images pull'
```

Step 8:

This command used to make master as a worker in Single node cluster

```
kubectl taint nodes --all node-role.kubernetes.io/master-
```

To apply pods network communication, we use calico network

```
kubectl apply
```

```
-f https://docs.projectcalico.org/v3.9/manifests/calico.yaml
```

to see the list of Pods we will use this command

```
kubectl get pods --all-namespaces
```

Result: The Installation of Kubernetes on EC2 cluster is successfully performed. The installed version of Kubernetes is Kubernetes version: v1.22.2

Conclusion: This experiment helped me to get a knowledge of Kubernetes. Kubernetes was successfully installed.

Experiment no. 4

Aim: To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy Your First Kubernetes Application.

Kubernetes Components:

When you deploy Kubernetes, you get a cluster.

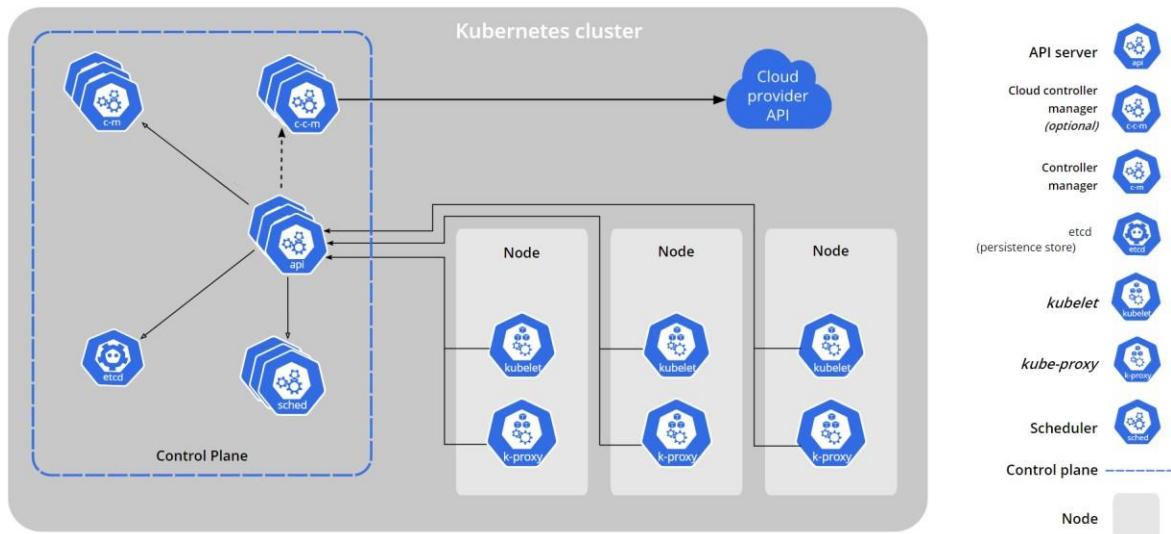
A Kubernetes cluster consists of a set of worker machines, called nodes, that run containerized applications. Every cluster has at least one worker node.

The worker node(s) host the Pods that are the components of the application workload.

The control plane manages the worker nodes and the Pods in the cluster. In production environments, the control plane usually runs across multiple computers and a cluster usually runs multiple nodes, providing fault-tolerance and high availability.

This document outlines the various components you need to have for a complete and working Kubernetes cluster.

Here's the diagram of a Kubernetes cluster with all the components tied together.



Outputs:

1. Starting with Kubernetes.

The screenshot shows the Amazon EKS console interface. On the left, there's a sidebar with navigation links for Amazon Container Services, Amazon ECS, Amazon EKS (which is selected), and Amazon ECR. The main content area features a large title 'Elastic Kubernetes Service (Amazon EKS)' followed by the subtitle 'Fully managed Kubernetes control plane'. Below the title, a paragraph explains that Amazon EKS is a managed service for using Kubernetes on AWS. To the right, there's a call-to-action button labeled 'Add cluster' with a dropdown menu. At the bottom right, there's a 'Pricing' section. The footer includes standard AWS links like Feedback, English (US), Privacy Policy, Terms of Use, and Cookie preferences.

2. Selecting EKS

The screenshot shows the IAM Management Console. In the search bar at the top, 'EKS' is typed. A list of AWS services is displayed, with 'EKS' highlighted. At the bottom of the screen, there are buttons for '* Required', 'Cancel', and 'Next: Permissions'.

3. Select EKS - Cluster

The screenshot shows the IAM Management Console interface. The URL in the address bar is `console.aws.amazon.com/iam/home#/roles$new?step=type`. The page title is "Select your use case". A list of options is shown, with "EKS - Cluster" highlighted in blue:

- EKS**
Allows EKS to manage clusters on your behalf.
- EKS - Cluster**
Allows access to other AWS service resources that are required to operate clusters managed by EKS.
- EKS - Connector**
Allows access to other AWS service resources that are required to connect to external clusters
- EKS - Fargate pod**
Allows access to other AWS service resources that are required to run Amazon EKS pods on AWS Fargate.
- EKS - Fargate profile**
Allows EKS to run Fargate tasks.
- EKS - Nodegroup**
Allow EKS to manage nodegroups on your behalf.

At the bottom, there are buttons for "* Required", "Cancel", and "Next: Permissions".

4. Creating Role

The screenshot shows the IAM Management Console interface. The URL in the address bar is `console.aws.amazon.com/iam/home#/roles$new?step=review&selectedService=AmazonEKS&selectedUseCase=AmazonEKSClusterPolicy`. The page title is "Create role". The "Review" step is active, indicated by a blue circle with the number 4. The role information is as follows:

- Role name***: eksrole
- Role description**: Allows access to other AWS service resources that are required to operate clusters managed by EKS.

Below the role details, it says "Maximum 1000 characters. Use alphanumeric and '+-=,@-_-' characters." Under "Trusted entities", it lists "AWS service: eks.amazonaws.com".

At the bottom, there are buttons for "Policies", "Cancel", "Previous", and "Create role".

5. Role has been created

The screenshot shows the AWS IAM Management Console. On the left, there's a sidebar with options like 'Identity and Access Management (IAM)', 'Access management', 'Access reports', and 'CloudWatch Metrics'. The main area displays a message: 'Introducing the new Roles list experience' and 'The role eksrole has been created.' Below this, a table lists three existing roles: 'AWSServiceRoleForAWSCloud9', 'AWSServiceRoleForOrganizations', and 'AWSServiceRoleForSupport'. A 'Create role' button is visible at the top right of the table.

6. Cloud Formation

The screenshot shows the AWS CloudFormation console. The sidebar includes 'Stacks', 'Designer', and 'Registry'. The main area shows a table for 'Stacks (1)'. One stack is listed: 'aws-cloud9-npm-user-b391cb4d24e648e98f9bac22ddbf8ca6' with a status of 'CREATE_COMPLETE' and a creation time of '2021-10-07 14:50:23 UTC+0530'.

Stack name	Status	Created time	Description
aws-cloud9-npm-user-b391cb4d24e648e98f9bac22ddbf8ca6	CREATE_COMPLETE	2021-10-07 14:50:23 UTC+0530	-

The screenshot shows the AWS CloudFormation 'Specify template' step. At the top, there are three options: 'Template is ready' (selected), 'Use a sample template', and 'Create template in Designer'. Below this, the 'Specify template' section is titled 'Template source'. It says 'Selecting a template generates an Amazon S3 URL where it will be stored.' There are two radio buttons: 'Amazon S3 URL' (selected) and 'Upload a template file'. An 'Amazon S3 URL' input field contains the value <https://amazon-eks.s3.us-west-2.amazonaws.com/cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml>. A 'View in Designer' button is located next to the URL. At the bottom right are 'Cancel' and 'Next' buttons.

The screenshot shows the AWS CloudFormation 'Specify stack details' step. On the left, a sidebar lists steps: Step 1 'Specify template' (selected), Step 2 'Specify stack details' (highlighted in blue), Step 3 'Configure stack options', and Step 4 'Review'. The main area is titled 'Specify stack details' and contains a 'Stack name' section with an input field containing 'eks-service'. A note below says 'Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-)'. To the right, there are four parameter sections: 'PublicSubnet01Block' (CidrBlock for public subnet 01 within the VPC, value 192.168.0.0/18), 'PublicSubnet02Block' (CidrBlock for public subnet 02 within the VPC, value 192.168.64.0/18), 'PrivateSubnet01Block' (CidrBlock for private subnet 01 within the VPC, value 192.168.128.0/18), and 'PrivateSubnet02Block' (CidrBlock for private subnet 02 within the VPC, value 192.168.192.0/18). At the bottom right are 'Cancel', 'Previous', and 'Next' buttons.

7. Stack is created

CloudFormation - Stack eks-service | Kubernetes On AWS | AWS Kube | +

CloudFormation / Stacks / eks-service

Stack info | **Events** | Resources | Outputs | Parameters | Template | Change sets

Stacks (2)

Filter by stack name

Active View nested

eks-service

2021-10-28 15:44:45 UTC+0530

CREATE_COMPLETE

aws-cloud9-npm-user-b391cb4d24e648e98f9ba

c22dbf8ca6

2021-10-07 14:50:23 UTC+0530

CREATE_COMPLETE

Events (68)

Search events

Timestamp	Logical ID	Status	Status reason
2021-10-28 15:48:21 UTC+0530	eks-service	CREATE_COMPLETE	-
2021-10-28 15:48:20 UTC+0530	PrivateRoute02	CREATE_COMPLETE	-
2021-10-28 15:48:05 UTC+0530	PrivateRoute01	CREATE_COMPLETE	-

Feedback English (US) © 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

8. Cluster created

Amazon EKS | Kubernetes On AWS | AWS Kube | +

Amazon Container Services X

Amazon ECS

Clusters

Task definitions

Amazon EKS

Clusters

Amazon ECR

Repositories

EKS > Clusters

Clusters (1) Info

Filter cluster by name, status, kubernetes version, or provider

Cluster name	Status	Kubernetes version	Provider
arshadcluster	Active	1.21	EKS

Feedback English (US) © 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

9. Installing kubectl

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Writing web request
Writing request stream... (Number of bytes written: 16736610)
```

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Try the new cross-platform PowerShell

PS C:\Windows\system32> curl -o kubectl.exe.sha256 https://amazon-eks.s3.us-west-2.amazonaws.com/1.21.2/2021-07-05/bin/windows/amd64/kubectl.exe.sha256
PS C:\Windows\system32> curl -o kubectl.exe.sha256 https://amazon-eks.s3.us-west-2.amazonaws.com/1.21.2/2021-07-05/bin/windows/amd64/kubectl.exe.sha256
PS C:\Windows\system32> Get-FileHash kubectl.exe

Algorithm      Hash                                         Path
----          ----                                         ---
SHA256         B5EC989B31219241F205D1E2861F35A4C4515BC70DCC307225A0736EAFF80C6  C:\Windows\system32\kubectl.exe

PS C:\Windows\system32>
```

10. Succesfully Installed

```
Administrator: Windows PowerShell
PS C:\Windows\system32> kubectl version --short --client
Client Version: v1.21.2-13+d2965f0db10712
PS C:\Windows\system32>
```

```

Administrator: Windows PowerShell
PS C:\Windows\system32> C:\Windows\system32\kubectl.exe
kubectl controls the Kubernetes cluster manager.

Find more information at: https://kubernetes.io/docs/reference/kubectl/overview/

Basic Commands (Beginner):
  create      Create a resource from a file or from stdin.
  expose      Take a replication controller, service, deployment or pod and expose it as a new Kubernetes Service
  run         Run a particular image on the cluster
  set         Set specific features on objects

Basic Commands (Intermediate):
  explain     Documentation of resources
  get         Display one or many resources
  edit        Edit a resource on the server
  delete      Delete resources by filenames, stdin, resources and names, or by resources and label selector

Deploy Commands:
  rollout    Manage the rollout of a resource
  scale      Set a new size for a Deployment, ReplicaSet or Replication Controller
  autoscale   Auto-scale a Deployment, ReplicaSet, StatefulSet, or ReplicationController

Cluster Management Commands:
  certificate Modify certificate resources.
  cluster-info Display cluster info
  top          Display Resource (CPU/Memory) usage.
  cordon      Mark node as unschedulable
  uncordon    Mark node as schedulable
  drain       Drain node in preparation for maintenance
  taint       Update the taints on one or more nodes

Troubleshooting and Debugging Commands:
  describe    Show details of a specific resource or group of resources
  logs        Print the logs for a container in a pod
  attach      Attach to a running container
  exec        Execute a command in a container
  port-forward Forward one or more local ports to a pod
  proxy       Run a proxy to the Kubernetes API server
  cp          Copy files and directories to and from containers.
  auth        Inspect authorization
  debug       Create debugging sessions for troubleshooting workloads and nodes

Advanced Commands:
  diff        Diff live version against would-be applied version

```

```

NAME          TYPE        CLUSTER-IP      EXTERNAL-IP      PORT(S)
AGE
kubernetes   ClusterIP  10.100.0.1   <none>           443/TCP        22m
PS C:\Users\snbha> kubectl get pods

No resources found in default namespace.

PS C:\Users\snbha> kubectl get all

NAME          STATUS        ROLES      AGE        VERSION
RT(S)  AGE
service/kubernetes   ClusterIP  10.100.0.1   <none>           443/TCP        22m
PS C:\Users\snbha> kubectl get pods --watch
error: the server doesn't have a resource type "nods"
PS C:\Users\snbha> kubectl get nodes --watch
NAME          STATUS        ROLES      AGE        VERSION
ip-172-31-79-10.ec2.internal  NotReady    <none>    0s       v1.18.9-eks-d1db3c
ip-172-31-79-10.ec2.internal  NotReady    <none>    0s       v1.18.9-eks-d1db3c
ip-172-31-79-10.ec2.internal  NotReady    <none>    0s       v1.18.9-eks-d1db3c
ip-172-31-40-253.ec2.internal NotReady    <none>    0s       v1.18.9-eks-d1db3c
ip-172-31-40-253.ec2.internal NotReady    <none>    0s       v1.18.9-eks-d1db3c
ip-172-31-40-253.ec2.internal NotReady    <none>    0s       v1.18.9-eks-d1db3c
ip-172-31-79-10.ec2.internal Ready      <none>    20s      v1.18.9-eks-d1db3c
ip-172-31-79-10.ec2.internal Ready      <none>    20s      v1.18.9-eks-d1db3c
ip-172-31-40-253.ec2.internal Ready      <none>    20s      v1.18.9-eks-d1db3c
ip-172-31-40-253.ec2.internal Ready      <none>    20s      v1.18.9-eks-d1db3c
ip-172-31-79-10.ec2.internal Ready      <none>    30s      v1.18.9-eks-d1db3c
ip-172-31-40-253.ec2.internal Ready      <none>    30s      v1.18.9-eks-d1db3c
PS C:\Users\snbha>

```

Experiment No: 5

Title: To understand terraform lifecycle, core concepts/ terminologies and install it on a Linux machine.

Solution:

Step 1: Download the compatible installation file from the official website.

istry Publishing



macOS

64-bit | Arm64



FreeBSD

32-bit | 64-bit | Arm



Linux

32-bit | 64-bit | Arm | Arm64



OpenBSD

Step 2: Unzip the file.

```
# root@ip-172-31-85-113:/home/ec2-user/Terra
[root@ip-172-31-85-113 Terra]# unzip terraform_0.11.11_linux_amd64.zip
Archive:  terraform_0.11.11_linux_amd64.zip
  inflating: terraform
[root@ip-172-31-85-113 Terra]# ls
terraform  terraform_0.11.11_linux_amd64.zip
[root@ip-172-31-85-113 Terra]# |
```

Step 3: Create the Environment variable path for the Terraform.

```
[root@ip-172-31-85-113 ~]# pwd  
/home/ec2-user/Terra  
[root@ip-172-31-85-113 Terra]# cd ..  
[root@ip-172-31-85-113 ec2-user]# pwd  
/home/ec2-user  
[root@ip-172-31-85-113 ec2-user]# terrafrom  
bash: terrafrom: command not found  
[root@ip-172-31-85-113 ec2-user]# terraform  
bash: terraform: command not found  
[root@ip-172-31-85-113 ec2-user]# PATH=$PATH:/home/ec2-user/Terra  
[root@ip-172-31-85-113 ec2-user]# |
```

Terraform is Successfully installed.

Result: The Experiment helped us to terraform lifecycle, core concepts/ terminologies. Terraform was successfully installed on a Linux machine.

Conclusion: The Terraform Installation is Successfully performed.

Experiment No: 6

Title: To Build, change, and destroy AWS / GCP /Microsoft Azure/ DigitalOcean infrastructure Using Terraform.

Solution:

1. Build AWS Infrastructure.

Step 1: Create a new AWS Access Key

The screenshot shows the AWS IAM Access Keys page. The left sidebar has a 'Multi-factor authentication (MFA)' section and a 'Access keys (access key ID and secret access key)' section, which is expanded. Below the sidebar, there is a note about using access keys for programmatic calls. A callout box highlights a note about root user access keys providing unrestricted access to the entire account. At the bottom of the page is a 'Create New Access Key' button.

The screenshot shows the 'Your Security Credentials' page. It includes a note about managing credentials for the AWS account. A modal window titled 'Create Access Key' is open, displaying a success message: 'Your access key (access key ID and secret access key) has been created successfully.' It also contains instructions to download the key file and a note about key rotation. Below the modal is a table with columns: Created, Access Key ID, Last Used, Last Used Region, Last Used Service, Status, and Actions. A 'Create New Access Key' button is at the bottom.

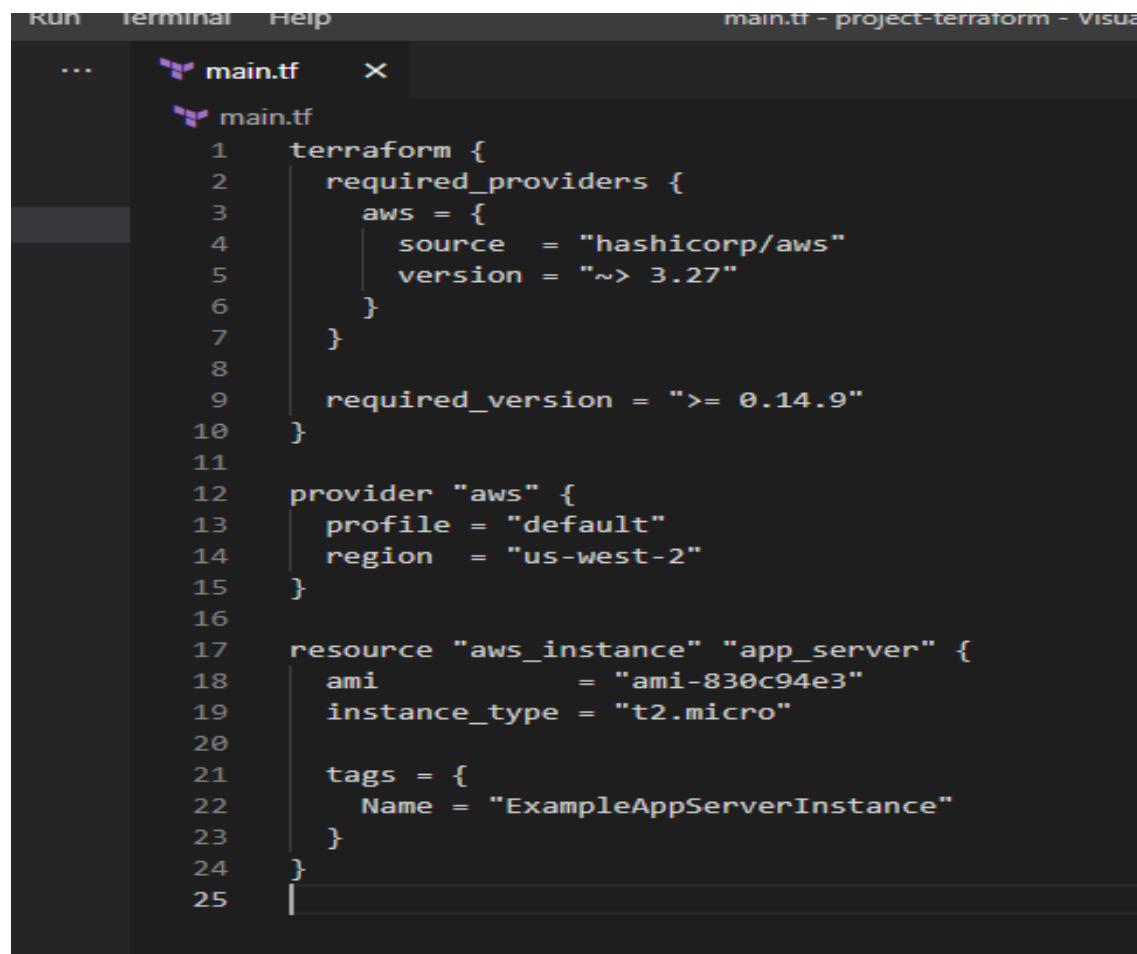
Step 2: Create a new Terraform project

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\ACER> cd project-terraform
PS C:\Users\ACER\project-terraform> aws configure
AWS Access Key ID [None]: AKIA57PBEJL52YSUHKY5
AWS Secret Access Key [None]: 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
Default region name [None]:
Default output format [None]:
PS C:\Users\ACER\project-terraform>
```

Step 3: Create a new .tf file and enter the following code.



The screenshot shows a Visual Studio Code interface with a dark theme. The title bar reads "main.tf - project-terraform - Visual Studio Code". The left sidebar shows a tree view with a root node "main.tf" expanded, revealing a sub-node "main.tf". The main editor area displays the following Terraform configuration code:

```
1  terraform {
2      required_providers {
3          aws = {
4              source  = "hashicorp/aws"
5              version = "~> 3.27"
6          }
7      }
8
9      required_version = ">= 0.14.9"
10 }
11
12 provider "aws" {
13     profile = "default"
14     region  = "us-west-2"
15 }
16
17 resource "aws_instance" "app_server" {
18     ami           = "ami-830c94e3"
19     instance_type = "t2.micro"
20
21     tags = {
22         Name = "ExampleAppServerInstance"
23     }
24 }
25 }
```

Step 4: Initialize Terraform.

```
Windows PowerShell
Default output format [None]:
PS C:\Users\ACER\project-terraform> terraform init

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 3.27"...
- Installing hashicorp/aws v3.62.0...
- Installed hashicorp/aws v3.62.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
PS C:\Users\ACER\project-terraform>
```

Step 5: Apply the setting

```
Windows PowerShell
Commands will detect it and remind you to do so if necessary.
PS C:\Users\ACER\project-terraform> terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# aws_instance.app_server will be created
+ resource "aws_instance" "app_server" {
    ami                               = "ami-830c94e3"
    arn                             = (known after apply)
    associate_public_ip_address      = (known after apply)
    availability_zone                = (known after apply)
    cpu_core_count                   = (known after apply)
    cpu_core_per_core                = (known after apply)
    disable_api_termination          = (known after apply)
    ebs_optimized                    = (known after apply)
    get_password_data                = false
    host_id                          = (known after apply)
    id                                = (known after apply)
    instance_initiated_shutdown_behavior = (known after apply)
    instance_state                   = (known after apply)
    instance_type                    = "t2.micro"
    ipv4_address_count               = (known after apply)
    ipv4_addresses                   = (known after apply)
    key_name                         = (known after apply)
    monitoring                       = (known after apply)
    outpost_arn                      = (known after apply)
    password_data                    = (known after apply)
    placement_group                  = (known after apply)
    primary_network_interface_id     = (known after apply)
    private_dns                       = (known after apply)
    private_ip                        = (known after apply)
    public_dns                        = (known after apply)
    public_ip                         = (known after apply)
    secondary_private_ips            = (known after apply)
}
```

```

+ kms_key_id          = (known after apply)
+ tags                = (known after apply)
+ throughput          = (known after apply)
+ volume_id           = (known after apply)
+ volume_size         = (known after apply)
+ volume_type         = (known after apply)
}
}

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

aws_instance.app_server: Creating...
aws_instance.app_server: Still creating... [10s elapsed]
aws_instance.app_server: Still creating... [20s elapsed]
aws_instance.app_server: Still creating... [30s elapsed]
aws_instance.app_server: Still creating... [40s elapsed]
aws_instance.app_server: Still creating... [50s elapsed]
aws_instance.app_server: Creation complete after 57s [id=i-0324ff4ec22dbe076]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\ACER\project-terraform> -

```

AWS Instance is successfully created using Terraform

2. Change Infrastructure.

Step 1: Change the AMI in your .tf file you created before as shown below.

```

resource "aws_instance" "app_server" {
  ami           = "ami-08d70e59c07c61a3a"
  instance_type = "t2.micro"

  tags = {
    Name = "ExampleAppServerInstance"
  }
}

```

Step 2: Apply the Settings.

```
PS C:\Users\ACER> cd project-terraform
PS C:\Users\ACER\project-terraform> terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with:
+ create

Terraform will perform the following actions:

# aws_instance.app_server will be created
+ resource "aws_instance" "app_server" {
  + ami                               = "ami-08d70e59c07c61a3a"
  + arn                             = (known after apply)
  + associate_public_ip_address      = (known after apply)
  + availability_zone                = (known after apply)
  + cpu_core_count                  = (known after apply)
  + cpu_threads_per_core            = (known after apply)
  + disable_api_termination        = (known after apply)
  + ebs_optimized                   = (known after apply)
  + get_password_data               = false
  + host_id                          = (known after apply)
  + id                                = (known after apply)
  + instance_initiated_shutdown_behavior = (known after apply)
  + instance_state                  = "t2.micro"
  + instance_type                   = (known after apply)
  + ipv6_address_count              = (known after apply)
  + ipv6_addresses                  = (known after apply)
  + key_name                         = (known after apply)
  + monitoring                       = (known after apply)
  + outpost_arn                     = (known after apply)
  + password_data                   = (known after apply)
  + placement_group                 = (known after apply)
  + primary_network_interface_id    = (known after apply)
  + private_dns                      = (known after apply)
  + private_ip                       = (known after apply)
  + public_dns                       = (known after apply)
  + public_ip                        = (known after apply)
  + secondary_private_ips           = (known after apply)
  + security_groups                 = (known after apply)
  + source_dest_check               = true
  + subnet_id                        = (known after apply)
  + tags
    + "Name" = "ExampleAppServerInstance"
}
```

Step 3: Check Result

```
+ volume_size          = (known after apply)
+ volume_type           = (known after apply)
}
}

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

aws_instance.app_server: Creating...
aws_instance.app_server: Still creating... [10s elapsed]
aws_instance.app_server: Still creating... [20s elapsed]
aws_instance.app_server: Still creating... [30s elapsed]
aws_instance.app_server: Still creating... [40s elapsed]
aws_instance.app_server: Creation complete after 45s [id=i-084b44a37c4dccba9]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\ACER\project-terraform>
```

AWS Instance is successfully updated using Terraform

3. Delete the Instance.

Step 1: Type Destroy command

```
Windows PowerShell
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
- destroy

Terraform will perform the following actions:

# aws_instance.app_server will be destroyed
- resource "aws_instance" "app_server" {
  ami                               = "ami-08d70e59c07c61a3a" -> null
  arn                             = "arn:aws:ec2:us-west-2:960934202107:instance/i-084b44a37c4dccba9" -> null
  associate_public_ip_address      = true -> null
  availability_zone                = "us-west-2b" -> null
  cpu_core_count                   = 1 -> null
  cpu_threads_per_core            = 1 -> null
  disable_api_termination         = false -> null
  ebs_optimized                    = false -> null
  get_password_data               = false -> null
  hibernation                      = false -> null
  id                                = "i-084b44a37c4dccba9" -> null
  instance_initiated_shutdown_behavior = "stop" -> null
  instance_state                   = "running" -> null
  instance_type                     = "t2.micro" -> null
  ipv6_address_count              = 0 -> null
  ipv6_addresses                   = [] -> null
  monitoring                       = false -> null
  primary_network_interface_id    = "eni-0746a73d991e4d885" -> null
  private_dns                       = "ip-172-31-21-171.us-west-2.compute.internal" -> null
  private_ip                         = "172.31.21.171" -> null
  public_dns                        = "ec2-18-237-82-17.us-west-2.compute.amazonaws.com" -> null
  public_ip                          = "18.237.82.17" -> null
  secondary_private_ips            = [] -> null
  security_groups                  = [
    - "default",
  ] -> null
  source_dest_check                = true -> null
  subnet_id                         = "subnet-0e004374299b5dfb4" -> null
  tags                               = {
    - "Name" = "ExampleAppServerInstance"
  } -> null
}
```

Step 2: Confirm the destruction

```
Windows PowerShell
- cpu_credits = "standard" -> null
}

- enclave_options {
  - enabled = false -> null
}

- metadata_options {
  http_endpoint           = "enabled" -> null
  http_put_response_hop_limit = 1 -> null
  http_tokens             = "optional" -> null
}

- root_block_device {
  - delete_on_termination = true -> null
  - device_name           = "/dev/sda1" -> null
  - encrypted              = false -> null
  - iops                   = 100 -> null
  - tags                   = {} -> null
  - throughput              = 0 -> null
  - volume_id              = "vol-07cc397d57371a404" -> null
  - volume_size             = 8 -> null
  - volume_type             = "gp2" -> null
}
}

Plan: 0 to add, 0 to change, 1 to destroy.

Do you really want to destroy all resources?
Terraform will destroy all your managed infrastructure, as shown above.
There is no undo. Only 'yes' will be accepted to confirm.

Enter a value: yes

aws_instance.app_server: Destroying... [id=i-084b44a37c4dccba9]
aws_instance.app_server: Still destroying... [id=i-084b44a37c4dccba9, 10s elapsed]
aws_instance.app_server: Still destroying... [id=i-084b44a37c4dccba9, 20s elapsed]
aws_instance.app_server: Still destroying... [id=i-084b44a37c4dccba9, 30s elapsed]
aws_instance.app_server: Destruction complete after 37s

Destroy complete! Resources: 1 destroyed.
PS C:\Users\ACER\project-terraform>
```

Result: The AWS Instance using Terraform was successfully created, changed and destroyed.

Conclusion: The Experiment helped me to understand the use and importance of terraform. The modules of Experiments were successfully performed.

Experiment No : 7

Title: To understand Static Analysis SAST process and learn to integrate Jenkins SAST to SonarQube/GitLab.

Solution:

Step 1: SonarQube Instance (Type the shown command)

```
$ docker run -d -p 9000:9000 sonarqube
```

Step 2: Generate User Token

The screenshot shows the 'Tokens' section of the SonarQube interface. It includes instructions for generating tokens to increase security. A 'Generate Tokens' button is present, along with a success message about a token named 'Jenkins' being created. Below this, a table lists the token details: Name (Jenkins), Last use (Never), and Created (October 19, 2020). A 'Revoke' button is also visible. At the bottom, there's a 'Change password' form with fields for Old Password*, New Password*, and Confirm Password*.

Step 3: Add necessary plugin

The screenshot shows the Jenkins 'Plugins' page. A search bar at the top has 'python' entered. The results list the 'Python Code Quality and Security' plugin, which is version 2.13 (build 7236) and is installed. The plugin page includes links to 'Homepage', 'Issue Tracker', and 'Uninstall'. Below the main list, it says '1 shown'.

Step 4: Configuring SonarQube in Codebase (Type the shown command)

```
sonar.projectKey=movie-crud-flask
```

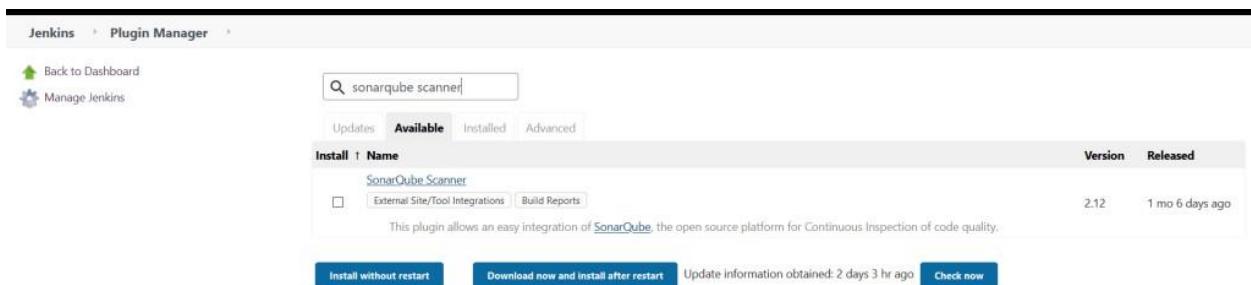
```
sonar.projectName=movie-crud-flask
```

```
sonar.projectVersion=1.0
```

```
sonar.projectBaseDir=.
```

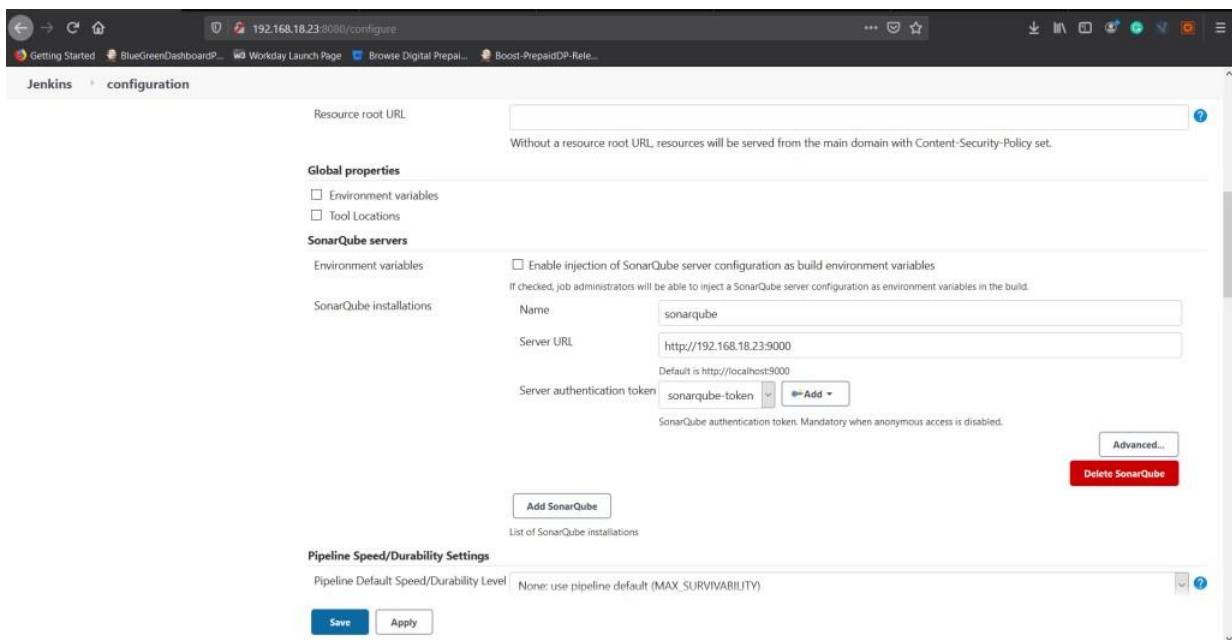
```
sonar.python.bandit.reportPaths=/report/banditResult.json
```

Step 5: Install SonarQube plugin in Jenkins.



The screenshot shows the Jenkins Plugin Manager interface. A search bar at the top contains the text "sonarqube scanner". Below the search bar, tabs for "Updates", "Available", "Installed", and "Advanced" are visible, with "Available" being the active tab. A table lists the "SonarQube Scanner" plugin, which is version 2.12 and was released 1 month and 6 days ago. The plugin description states: "This plugin allows an easy integration of [SonarQube](#), the open source platform for Continuous Inspection of code quality." At the bottom of the table are buttons for "Install without restart" (highlighted in blue), "Download now and install after restart", and "Check now".

Step 6: Tool Configuration SonarQube Scanner



The screenshot shows the Jenkins configuration page for the SonarQube Scanner tool. The "Resource root URL" field is empty with a placeholder: "Without a resource root URL, resources will be served from the main domain with Content-Security-Policy set." Under "Global properties", there are checkboxes for "Environment variables" and "Tool Locations", both of which are unchecked. In the "SonarQube servers" section, a new server named "sonarqube" is being configured. The "Name" field is "sonarqube", the "Server URL" field is "http://192.168.18.23:9000", and the "Server authentication token" dropdown contains "sonarqube-token" with an "Add" button next to it. A note below the token field says: "SonarQube authentication token. Mandatory when anonymous access is disabled." At the bottom of the configuration page are "Save" and "Apply" buttons.

The screenshot shows the Jenkins Global Tool Configuration page. Under the 'SonarQube Scanner' section, there is a 'SonarQube Scanner installations' table. A new entry is being added, with the 'Name' field set to 'sonarqube'. Below the table, there is an 'Install from Maven Central' section with a dropdown menu showing 'Version SonarQube Scanner 4.5.0.2216'. At the bottom of the page are 'Save' and 'Apply' buttons.

Step 7: SonarQube Scanner in Jenkins Pipeline

Add following code.

```
stage ("sonar-publish"){\n    steps {\n        echo "=====Performing Sonar Scan=====\n        sh "${tool("sonarqube")}/bin/sonar-scanner\n    }\n}
```

Step 8: Execute the code.

The image displays two screenshots illustrating the integration of Jenkins SAST with SonarQube.

Jenkins Pipeline Screenshot:

- The top part shows the Jenkins pipeline for the "movie-db-pipeline" project. It includes a summary card with build details (Branch: -, Commit: -, Duration: 2m 18s, Started by user Prabhu Vignesh, Last Commit: 2 days ago) and a pipeline graph showing stages: Start, Git checkout, Python Flask Prepare, Unit Test, Python Bandit Security Scan, Dependency Check with Pyth..., Static Analysis with python-taint, sonar-publish, and End. All stages are marked as green (Passed).
- The bottom part shows the log output for the "sonar-publish" stage, which includes commands like "Performing Sonar Scan", "sonarqube", and the execution of "/var/lib/jenkins/tools/hudson.plugins.sonar.SonarRunnerInstallation/sonarqube/bin/sonar-scanner". The duration for this stage is 1m 12s.

SonarQube Analysis Screenshot:

- The screenshot shows the SonarQube interface for the "movie-crud-flask" project. The dashboard provides an overview of the code quality metrics:
 - Quality Gate:** Passed (Green)
 - Reliability (Bugs):** 6 Bugs (Yellow)
 - Vulnerabilities:** 0 (Green)
 - Hotspots Reviewed:** 0.0% (Red)
 - Coverage:** 1 A (Green)
 - Duplications:** 0.0% (Green)
 - Code Smells:** 0.0% (Red)
 - Python, CSS, ...:** 169 (Blue)
- Filters:** On the left, there are filters for Quality Gate (Passed, Failed), Reliability (Bugs), Security (Vulnerabilities), and Security Review (Security Hotspots).
- Message:** A yellow warning message at the bottom states: "Embedded database should be used for evaluation purposes only. The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine."
- Footer:** Includes links to SonarQube technology information and community resources.

Result: The Experiment is successfully performed and the integration of Jenkins SAST to SonarQube/GitLab was done successfully.

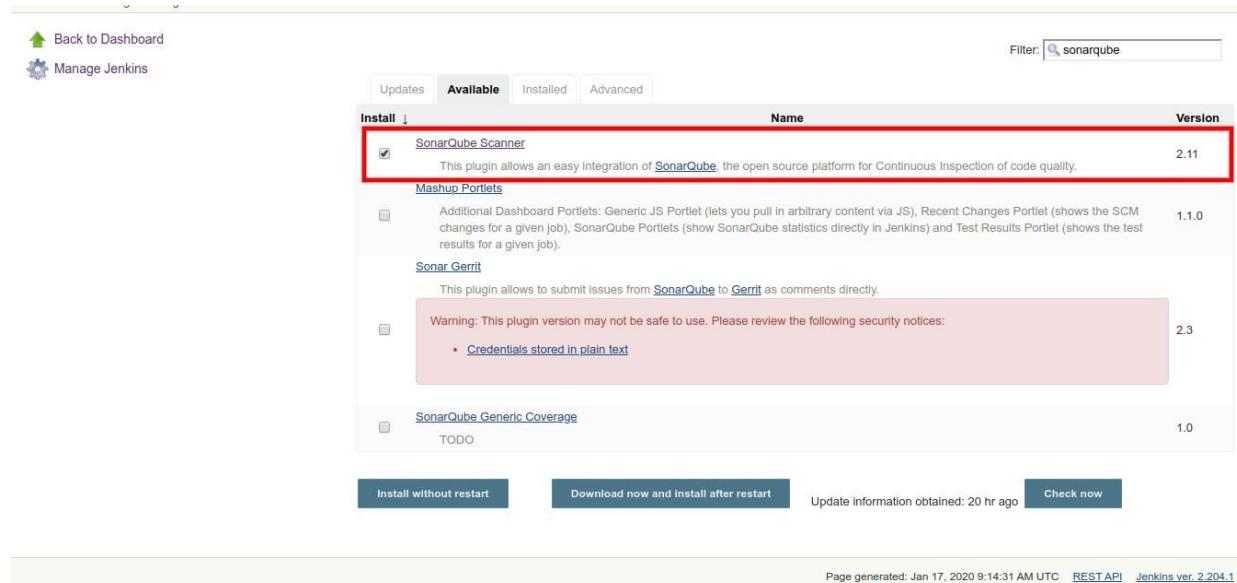
Experiment No: 8

Title: Create a Jenkins CICD Pipeline with SonarQube / GitLab Integration to perform a static Analysis of the code to detect bugs, code smells, and security vulnerabilities on a sample Web / Java / Python application.

Solution:

Step 1: Integrating SonarQube and GitLabs in Jenkins

to integrate SonarQube. Go [Dashboard > Manage Jenkins > Manage Plugins](#) and search for SonarQube

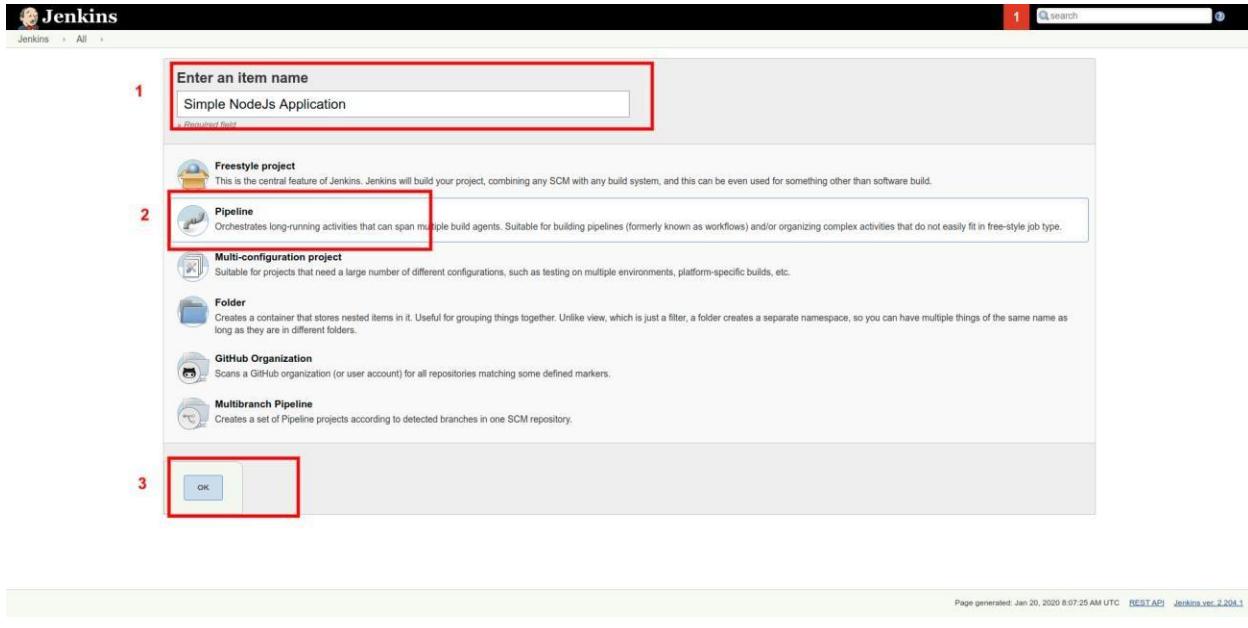


The screenshot shows the Jenkins Manage Plugins interface. The 'Available' tab is selected, and the search bar at the top right contains the text 'sonarqube'. A single plugin, 'SonarQube Scanner', is listed. It has a checked checkbox under 'Install' and a red border around it, indicating it is selected for installation. The plugin details show it allows for an easy integration of SonarQube and provides statistics and test results portlets. A warning message states that the plugin version may not be safe to use due to stored credentials in plain text. Other plugins listed include 'Mashup Portlets', 'Sonar Gerrit', and 'SonarQube Generic Coverage'. At the bottom, there are buttons for 'Install without restart' and 'Download now and install after restart', along with a status message 'Update information obtained: 20 hr ago' and a 'Check now' button. The footer of the page includes links for 'Page generated: Jan 17, 2020 9:14:31 AM UTC', 'REST API', and 'Jenkins ver. 2.204.1'.

Make sure you restart Jenkins once the plugin is successfully installed.

Step 2: Create a New Job

Go to Jenkins' Dashboard and click on the "New Item" link. Now, enter the item name and select *Pipeline* option as shown in the figure:



Step 3: Pipeline

```

1- pipeline {
2-   agent any
3-   tools nodejs "nodeenv"
4-   stages {
5-     stage("Code Checkout from GitLab") {
6-       steps {
7-         git branch: 'master',
8-         credentialsId: 'gitlab_access_token',
9-         url: 'http://[REDACTED]:18888/root/test-project.git'
10-      }
11-    }
12-    stage("Code Quality Check via SonarQube") {
13-      steps {
14-        script {
15-          def scannerHome = tool 'sonarqube';
16-          withSonarQubeEnv('sonarqube-container') {
17-            sh "[${toolHome}/sonarqube]/bin/sonar-scanner \
18-          }"
19-        }
20-      }
21-    }
22-  }
23-}

```

Step 4 - Run Pipeline

Now, as you have saved the pipeline script, it's time to build your application in Jenkins. Go to **Dashboard > YOUR PROJECT > Build Now.**

The screenshot shows the Jenkins interface for a pipeline named "Simple NodeJS Application". The left sidebar provides navigation options for managing pipelines. The main content area displays the pipeline's configuration and status.

If your build runs successfully, you will be able to see the time taken by each stage, in Stage View

Result: The Creation of a Jenkins CICD Pipeline with SonarQube Integration was successfully created.

Conclusion: The Experiment was successfully performed.

Experiment No: 9

Aim: To Understand Continuous monitoring and Installation and configuration of Nagios Core, Nagios Plugins and NRPE (Nagios Remote Plugin Executor) on Linux Machine.

Solution:

Step by Step method for installing Nagios in Amazon Linux

Step 1: Install Prerequisite Software

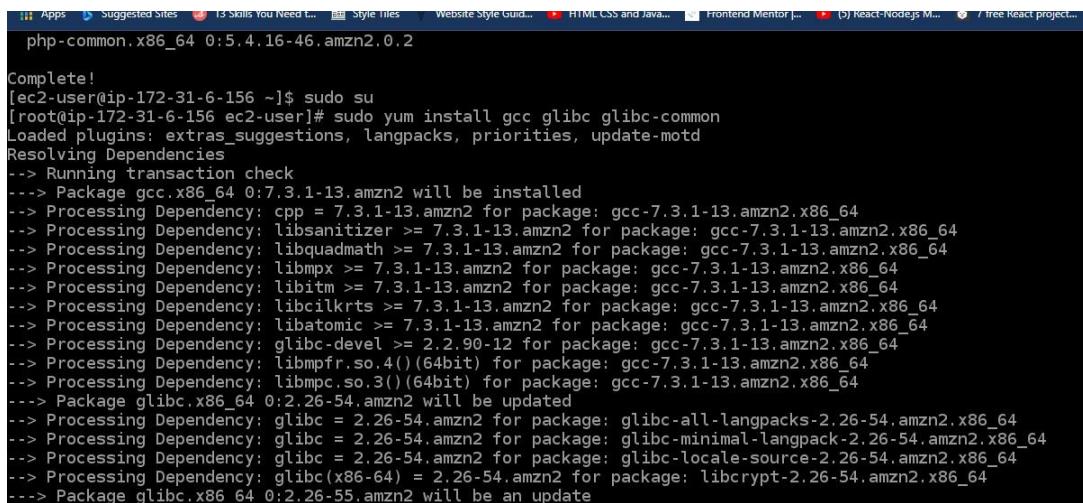
Nagios requires the following packages are installed on your server prior to installing Nagios:

Apache

PHP

GCC compiler

GD development libraries



```
php-common.x86_64 0:5.4.16-46.amzn2.0.2
Complete!
[ec2-user@ip-172-31-6-156 ~]$ sudo su
[root@ip-172-31-6-156 ec2-user]# sudo yum install gcc glibc glibc-common
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Resolving Dependencies
--> Running transaction check
--> Package gcc.x86_64 0:7.3.1-13.amzn2 will be installed
--> Processing Dependency: cpp = 7.3.1-13.amzn2 for package: gcc-7.3.1-13.amzn2.x86_64
--> Processing Dependency: libasanitizer >= 7.3.1-13.amzn2 for package: gcc-7.3.1-13.amzn2.x86_64
--> Processing Dependency: libquadmath >= 7.3.1-13.amzn2 for package: gcc-7.3.1-13.amzn2.x86_64
--> Processing Dependency: libmpx >= 7.3.1-13.amzn2 for package: gcc-7.3.1-13.amzn2.x86_64
--> Processing Dependency: libitm >= 7.3.1-13.amzn2 for package: gcc-7.3.1-13.amzn2.x86_64
--> Processing Dependency: libcilkkrts >= 7.3.1-13.amzn2 for package: gcc-7.3.1-13.amzn2.x86_64
--> Processing Dependency: libatomic >= 7.3.1-13.amzn2 for package: gcc-7.3.1-13.amzn2.x86_64
--> Processing Dependency: glibc-devel >= 2.2.90-12 for package: gcc-7.3.1-13.amzn2.x86_64
--> Processing Dependency: libmpfr.so.4()(64bit) for package: gcc-7.3.1-13.amzn2.x86_64
--> Processing Dependency: libmpc.so.3()(64bit) for package: gcc-7.3.1-13.amzn2.x86_64
--> Package glibc.x86_64 0:2.26-54.amzn2 will be updated
--> Processing Dependency: glibc = 2.26-54.amzn2 for package: glibc-all-langpacks-2.26-54.amzn2.x86_64
--> Processing Dependency: glibc = 2.26-54.amzn2 for package: glibc-minimal-langpack-2.26-54.amzn2.x86_64
--> Processing Dependency: glibc = 2.26-54.amzn2 for package: glibc-locale-source-2.26-54.amzn2.x86_64
--> Processing Dependency: glibc(x86-64) = 2.26-54.amzn2 for package: libcrypt-2.26-54.amzn2.x86_64
--> Package glibc.x86_64 0:2.26-55.amzn2 will be an update
```

```

libitm.x86_64 0:7.3.1-13.amzn2          libmpc.x86_64 0:1.0.1-3.amzn2.0.2
libmpx.x86_64 0:7.3.1-13.amzn2          libquadmath.x86_64 0:7.3.1-13.amzn2
libsanitizer.x86_64 0:7.3.1-13.amzn2    mpfr.x86_64 0:3.1.1-4.amzn2.0.2

Updated:
glibc.x86_64 0:2.26-55.amzn2           glibc-common.x86_64 0:2.26-55.amzn2

Dependency Updated:
glibc-all-langpacks.x86_64 0:2.26-55.amzn2   glibc-locale-source.x86_64 0:2.26-55.amzn2
glibc-minimal-langpack.x86_64 0:2.26-55.amzn2  libcrypt.x86_64 0:2.26-55.amzn2

Complete!
[root@ip-172-31-6-156 ec2-user]# sudo yum install gd gd-devel
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Resolving Dependencies
--> Running transaction check
--> Package gd.x86_64 0:2.0.35-27.amzn2 will be installed
--> Processing Dependency: libfontconfig.so.1()(64bit) for package: gd-2.0.35-27.amzn2.x86_64
--> Processing Dependency: libXpm.so.4()(64bit) for package: gd-2.0.35-27.amzn2.x86_64
--> Processing Dependency: libX11.so.6()(64bit) for package: gd-2.0.35-27.amzn2.x86_64
--> Package gd-devel.x86_64 0:2.0.35-27.amzn2 will be installed
--> Processing Dependency: zlib-devel for package: gd-devel-2.0.35-27.amzn2.x86_64
--> Processing Dependency: libpng-devel for package: gd-devel-2.0.35-27.amzn2.x86_64
--> Processing Dependency: libjpeg-devel for package: gd-devel-2.0.35-27.amzn2.x86_64
--> Processing Dependency: libXpm-devel for package: gd-devel-2.0.35-27.amzn2.x86_64

```

Step 2: Create Account Information

You need to set up a Nagios user. Run the following commands:

- sudo adduser -m nagios
- sudo passwd nagios

Type the new password twice.

```

expat-devel.x86_64 0:2.1.0-12.amzn2
fontconfig-devel.x86_64 0:2.13.0-4.3.amzn2
freetype-devel.x86_64 0:2.8-14.amzn2.1
libSM.x86_64 0:1.2.2-2.amzn2.0.2
libX11-common.noarch 0:1.6.7-3.amzn2.0.2
libXau.x86_64 0:1.0.8-2.1.amzn2.0.2
libXext.x86_64 0:1.3.3-3.amzn2.0.2
libXpm-devel.x86_64 0:3.5.12-1.amzn2.0.2
libjpeg-turbo-devel.x86_64 0:2.0.90-2.amzn2.0.5
libuuid-devel.x86_64 0:2.30.2-2.amzn2.0.5
libxcb-devel.x86_64 0:1.12-1.amzn2.0.2
zlib-devel.x86_64 0:1.2.7-18.amzn2

fontconfig.x86_64 0:2.13.0-4.3.amzn2
fontpackages-filesystem.noarch 0:1.44-8.amzn2
libICE.x86_64 0:1.0.9-9.amzn2.0.2
libX11.x86_64 0:1.6.7-3.amzn2.0.2
libX11-devel.x86_64 0:1.6.7-3.amzn2.0.2
libXau-devel.x86_64 0:1.0.8-2.1.amzn2.0.2
libXpm.x86_64 0:3.5.12-1.amzn2.0.2
libXt.x86_64 0:1.1.5-3.amzn2.0.2
libpng-devel.x86_64 2:1.5.13-8.amzn2
libxcb.x86_64 0:1.12-1.amzn2.0.2
xorg-x11proto-devel.noarch 0:2018.4-1.amzn2.0.2

Complete!
[root@ip-172-31-6-156 ec2-user]# sudo adduser -m nagios
[root@ip-172-31-6-156 ec2-user]# sudo passwd nagios
Changing password for user nagios.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
Sorry, passwords do not match.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[root@ip-172-31-6-156 ec2-user]#

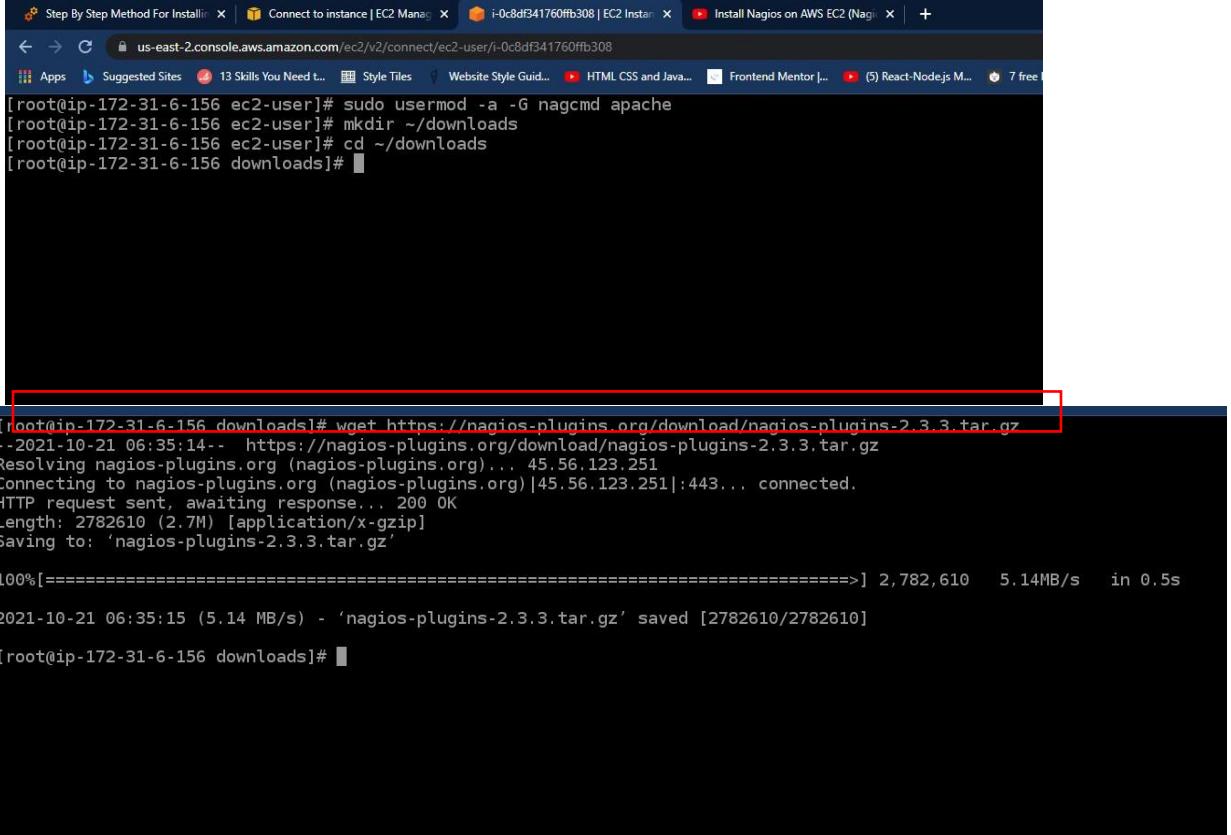
```

- sudo groupadd nagcmd
- sudo usermod -a -G nagcmd nagios
- sudo usermod -a -G nagcmd apache

Step 3: Download Nagios Core and the Plugins

Create a directory for storing the downloads.

```
mkdir ~/downloads  
  
cd ~/downloads  
  
wget http://prdownloads.sourceforge.net/sourceforge/nagios/nagios-4.0.8.tar.gz  
  
wget http://nagios-plugins.org/download/nagios-plugins-2.0.3.tar.gz
```



The screenshot shows a terminal window on an AWS EC2 instance. The terminal session starts with root privileges, as indicated by the '#'. It first creates a directory named 'downloads' in the user's home directory. Then, it changes into that directory. Finally, it runs the 'wget' command to download the Nagios Plugins tarball from the official website. The download progress is shown, indicating a speed of 5.14MB/s over a 0.5s period.

```
[root@ip-172-31-6-156 ~]# sudo usermod -a -G nagcmd apache  
[root@ip-172-31-6-156 ~]# mkdir ~/downloads  
[root@ip-172-31-6-156 ~]# cd ~/downloads  
[root@ip-172-31-6-156 downloads]#  
  
[root@ip-172-31-6-156 downloads]# wget https://nagios-plugins.org/download/nagios-plugins-2.3.3.tar.gz  
--2021-10-21 06:35:14-- https://nagios-plugins.org/download/nagios-plugins-2.3.3.tar.gz  
Resolving nagios-plugins.org (nagios-plugins.org)... 45.56.123.251  
Connecting to nagios-plugins.org (nagios-plugins.org)|45.56.123.251|:443... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 2782610 (2.7M) [application/x-gzip]  
Saving to: 'nagios-plugins-2.3.3.tar.gz'  
  
100%[=====] 2,782,610 5.14MB/s in 0.5s  
2021-10-21 06:35:15 (5.14 MB/s) - 'nagios-plugins-2.3.3.tar.gz' saved [2782610/2782610]  
[root@ip-172-31-6-156 downloads]#
```

Step 4: Compile and Install Nagios

- Extract the Nagios source code tarball.
 - tar zxvf nagios-4.0.8.tar.gz
 - cd nagios-4.0.8

```

Saving to: 'nagios-plugins-2.0.3.tar.gz'
100%[=====] 2,659,772 7.96MB/s in 0.3s
2021-10-21 06:36:22 (7.96 MB/s) - 'nagios-plugins-2.0.3.tar.gz' saved [2659772/2659772]

[root@ip-172-31-6-156 downloads]# tar zxvf nagios-4.0.8.tar.gz
nagios-4.0.8/
nagios-4.0.8/.gitignore
nagios-4.0.8/ChangeLog
nagios-4.0.8/INSTALLING
nagios-4.0.8/LEGAL
nagios-4.0.8/LICENSE
nagios-4.0.8/Makefile.in
nagios-4.0.8/README
nagios-4.0.8/README.asciidoc
nagios-4.0.8/THANKS
nagios-4.0.8/UPGRADING
nagios-4.0.8/base/
nagios-4.0.8/base/.gitignore
nagios-4.0.8/base/Makefile.in
nagios-4.0.8/base/broker.c
nagios-4.0.8/base/checks.c
nagios-4.0.8/base/commands.c
nagios-4.0.8/base/config.c

```

```

nagios-4.0.8/worker/Makefile.in
nagios-4.0.8/worker/ping/
nagios-4.0.8/worker/ping/.gitignore
nagios-4.0.8/worker/ping/Makefile.in
nagios-4.0.8/worker/ping/worker-ping.c
nagios-4.0.8/xdata/
nagios-4.0.8/xdata/.gitignore
nagios-4.0.8/xdata/Makefile.in
nagios-4.0.8/xdata/xcddefault.c
nagios-4.0.8/xdata/xcddefault.h
nagios-4.0.8/xdata/xodtemplate.c
nagios-4.0.8/xdata/xodtemplate.h
nagios-4.0.8/xdata/xpddefault.c
nagios-4.0.8/xdata/xpddefault.h
nagios-4.0.8/xdata/xrddefault.c
nagios-4.0.8/xdata/xrddefault.h
nagios-4.0.8/xdata/xsddefault.c
nagios-4.0.8/xdata/xsddefault.h
[root@ip-172-31-6-156 downloads]# cd nagios-4.0.8
[root@ip-172-31-6-156 nagios-4.0.8]#

```

- Run the configuration script with the name of the group which you have created in the above step.

➤ ./configure --with-command-group=nagcmd

```

nagios-4.0.8/tap/tests/todo/makewrite.in
nagios-4.0.8/tap/tests/todo/test.c
nagios-4.0.8/tap/tests/todo/test.pl
nagios-4.0.8/tap/tests/todo/test.t
[root@ip-172-31-6-156 nagios-4.0.8]# ./configure --with-command-group=nagcmd
checking for a BSD-compatible install... /bin/install -c
checking build system type... x86_64-unknown-linux-gnu
checking host system type... x86_64-unknown-linux-gnu
checking for gcc... gcc
checking for C compiler default output file name... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking whether make sets $(MAKE)... yes
checking for strip... /bin/strip

```

- Compile the Nagios source code.

➤ make all

```

Web Interface Options:
-----
    HTML URL: http://localhost/nagios/
    CGI URL: http://localhost/nagios/cgi-bin/
Traceroute (used by WAP): /bin/traceroute

Review the options above for accuracy. If they look okay,
type 'make all' to compile the main program and CGIs.

[root@ip-172-31-6-156 nagios-4.0.8]#
[root@ip-172-31-6-156 nagios-4.0.8]# make all
cd ./base && make
make[1]: Entering directory `/root/downloads/nagios-4.0.8/base'
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o nagios.o nagios.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o broker.o broker.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o nebmods.o nebmods.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o ../common/shared.o ../common/shared.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o nerd.o nerd.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o query-handler.o query-handler.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o workers.o workers.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o checks.o checks.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o config.o config.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o commands.o commands.c

```

- Install binaries, init script, sample config files and set permissions on the external command directory.
- sudo make install
- sudo make install-init
- sudo make install-config
- sudo make install-commandmode

```

web interface

*** Support Notes ***

If you have questions about configuring or running Nagios,
[root@ip-172-31-6-156 nagios-4.0.8]#
[root@ip-172-31-6-156 nagios-4.0.8]#
[root@ip-172-31-6-156 nagios-4.0.8]#
[root@ip-172-31-6-156 nagios-4.0.8]# sudo make install
cd ./base && make install
make[1]: Entering directory `/root/downloads/nagios-4.0.8/base'
make install-basic
make[2]: Entering directory `/root/downloads/nagios-4.0.8/base'
/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/bin
/bin/install -c -m 774 -o nagios -g nagios nagios /usr/local/nagios/bin
/bin/install -c -m 774 -o nagios -g nagios nagiostats /usr/local/nagios/bin
make[2]: Leaving directory `/root/downloads/nagios-4.0.8/base'
make[2]: Entering directory `/root/downloads/nagios-4.0.8/base'
/bin/strip /usr/local/nagios/bin/nagios
/bin/strip /usr/local/nagios/bin/nagiostats
make[2]: Leaving directory `/root/downloads/nagios-4.0.8/base'
make[1]: Leaving directory `/root/downloads/nagios-4.0.8/base'
cd ./cgi && make install

```

```

cp -rf contrib/exfoliation/images/* /usr/local/nagios/share/images
*** Exfoliation theme installed ***
NOTE: Use 'make install-classicui' to revert to classic Nagios theme

make[1]: Leaving directory `/root/downloads/nagios-4.0.8'
make install-basic
[root@ip-172-31-6-156 nagios-4.0.8]# sudo make install-init
/bin/install -c -m 755 -d -o root -g root /etc/rc.d/init.d
/bin/install -c -m 755 -o root -g root daemon-init /etc/rc.d/init.d/nagios

*** Init script installed ***

[root@ip-172-31-6-156 nagios-4.0.8]# sudo make install-config
/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc
/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc/objects
/bin/install -c -b -m 664 -o nagios -g nagios sample-config/nagios.cfg /usr/local/nagios/etc/nagios.cfg
/bin/install -c -b -m 664 -o nagios -g nagios sample-config/cgi.cfg /usr/local/nagios/etc/cgi.cfg
/bin/install -c -b -m 660 -o nagios -g nagios sample-config/resource.cfg /usr/local/nagios/etc/resource.cfg
/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/templates.cfg /usr/local/nagios/etc/objects/templates.cfg
/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/commands.cfg /usr/local/nagios/etc/objects/commands.cfg
/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/contacts.cfg /usr/local/nagios/etc/objects/contacts.cfg

```

```

calhost.cfg
/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/windows.cfg /usr/local/nagios/etc/objects/windows.cfg
/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/printer.cfg /usr/local/nagios/etc/objects/printer.cfg
/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/switch.cfg /usr/local/nagios/etc/objects/switch.cfg

*** Config files installed ***

Remember, these are *SAMPLE* config files. You'll need to read
the documentation for more information on how to actually define
services, hosts, etc. to fit your particular needs.

[root@ip-172-31-6-156 nagios-4.0.8]#
[root@ip-172-31-6-156 nagios-4.0.8]#
[root@ip-172-31-6-156 nagios-4.0.8]#
[root@ip-172-31-6-156 nagios-4.0.8]#
[root@ip-172-31-6-156 nagios-4.0.8]# sudo make install-commandmode
/bin/install -c -m 775 -o nagios -g nagcmd -d /usr/local/nagios/var/rw
chmod g+s /usr/local/nagios/var/rw

*** External command directory configured ***

[root@ip-172-31-6-156 nagios-4.0.8]#

```

Step 5: Customize Configuration

Change E-Mail address with nagiosadmin contact definition you'd like to use for receiving Nagios alerts.

➤ sudo vim /usr/local/nagios/etc/objects/contacts.cfg

```
              _|_(_|_) / Amazon Linux 2 AMI
              __\_\_|_
https://aws.amazon.com/amazon-linux-2/
3 package(s) needed for security, out of 9 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-6-156 ~]$ sudo vim /usr/local/nagios/etc/objects/contacts.cfg
```

Step 6: Configure the Web Interface

➤ sudo make install-webconf

```
[root@ip-172-31-6-156 nagios-4.0.8]#
[root@ip-172-31-6-156 nagios-4.0.8]#
[root@ip-172-31-6-156 nagios-4.0.8]#
[root@ip-172-31-6-156 nagios-4.0.8]# sudo make install-webconf
/bin/install -c -m 644 sample-config/httpd.conf /etc/httpd/conf.d/nagios.conf
*** Nagios/Apache conf file installed ***
[root@ip-172-31-6-156 nagios-4.0.8]#
```

Create a nagiosadmin account for logging into the Nagios web interface. Note the password you need it while login to Nagios web console.

- sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin // Type Password Twice
- sudo service httpd restart //Restart Service

Step 7: Compile and Install the Nagios Plugins

- Extract the Nagios plugins source code tarball.

- cd ~/downloads
- tar zxvf nagios-plugins-2.0.3.tar.gz
- cd nagios-plugins-2.0.3
- Compile and install the plugins.
- ./configure --with-nagios-user=nagios --with-nagios-group=nagios
- make
- sudo make install

```
config.status: creating po/Postrl1y
config.status: creating po/Makefile
    --with-apt-get-command:
        --with-ping6-command: /sbin/ping6 -n -U -w %d -c %d %s
        --with-ping-command: /bin/ping -n -U -w %d -c %d %s
            --with-ipv6: yes
            --with-mysql: no
            --with-openssl: no
            --with-gnutls: no
        --enable-extra-opts: yes
            --with-perl: /bin/perl
    --enable-perl-modules: no
        --with-cgiurl: /nagios/cgi-bin
    --with-trusted-path: /bin:/sbin:/usr/bin:/usr/sbin
        --enable-libtap: no
[root@ip-172-31-6-156 nagios-plugins-2.0.3]# ./configure --with-nagios-user=nagios --with-nagios-group=nagios
```

```
[root@ip-172-31-6-156 nagios-plugins-2.0.3]#  
[root@ip-172-31-6-156 nagios-plugins-2.0.3]#  
[root@ip-172-31-6-156 nagios-plugins-2.0.3]# sudo make install
```

```
[root@ip-172-31-6-156 nagios-plugins-2.0.3]#  
[root@ip-172-31-6-156 nagios-plugins-2.0.3]#  
[root@ip-172-31-6-156 nagios-plugins-2.0.3]#  
[root@ip-172-31-6-156 nagios-plugins-2.0.3]#  
[root@ip-172-31-6-156 nagios-plugins-2.0.3]# sudo chkconfig --add nagios
```

Step 8: Start Nagios

Add Nagios to the list of system services and have it automatically start when the system boots.

- sudo chkconfig --add nagios
- sudo chkconfig nagios on

Verify the sample Nagios configuration files.

- sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

```
[root@ip-172-31-6-156 nagios-plugins-2.0.3]#  
[root@ip-172-31-6-156 nagios-plugins-2.0.3]# sudo chkconfig --add nagios  
[root@ip-172-31-6-156 nagios-plugins-2.0.3]# sudo chkconfig nagios on  
[root@ip-172-31-6-156 nagios-plugins-2.0.3]#  
[root@ip-172-31-6-156 nagios-plugins-2.0.3]#  
[root@ip-172-31-6-156 nagios-plugins-2.0.3]# sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
```

Nagios Core 4.0.8
Copyright (c) 2009-present Nagios Core Development Team and Community Contributors
Copyright (c) 1999-2009 Ethan Galstad
Last Modified: 08-12-2014
License: GPL

Website: http://www.nagios.org
Reading configuration data...
 Read main config file okay...
 Read object config files okay...

Running pre-flight check on configuration data

If there are no errors, start Nagios.

- sudo service nagios start

```
Checked 0 service groups.
Checked 1 contacts.
Checked 1 contact groups.
Checked 24 commands.
Checked 5 time periods.
Checked 0 host escalations.
Checked 0 service escalations.
Checking for circular paths...
Checked 1 hosts
Checked 0 service dependencies
Checked 0 host dependencies
Checked 5 timeperiods
Checking global event handlers...
Checking obsessive compulsive processor commands...
Checking misc settings...

Total Warnings: 0
Total Errors: 0

Things look okay - No serious problems were detected during the pre-flight check
[root@ip-172-31-6-156 nagios-plugins-2.0.3]# [root@ip-172-31-6-156 nagios-plugins-2.0.3]# sudo service nagios start
Starting nagios (via systemctl): [ OK ]
[root@ip-172-31-6-156 nagios-plugins-2.0.3]#
```

- ❖ The Nagios service has been started.

Result: This Experiment helped me to understand Continuous monitoring and Installation and configuration of Nagios Core, Nagios Plugins and NRPE. The Experiment was successfully completed.

Conclusion: The modules of the experiment were successfully executed. The Installation of Nagios core was performed.

Experiment No: 10

Aim: To perform Port, Service monitoring, Windows/Linux server monitoring using Nagios.

Solution:

1. Adding a remote Linux Host to Nagios Server

Step1: Install Nagios NRPE

```
# sudo apt install nagios-nrpe-server nagios-plugins
```

```
root@ubuntu-nagios-client:~# 
root@ubuntu-nagios-client:~# 
root@ubuntu-nagios-client:~# sudo apt-get install nagios-nrpe-server nagios-plugins
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  grub-pc-bin libnumual
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  libarchive13 libavahi-client3 libavahi-common-data libavahi-common3 libcurl4 libdbi1 libjansson4 libldb1
  libmysqlclient20 libnet-snmp-perl libpq5 libpython-stdlib libpython2.7 libpython2.7-minimal libpython2.7-stdlib
  libssensors4 libsmclient libsnmp-base libsnmp30 libtalloc2 libtdb1 libtevent0 libtirpc1 libwbclient0
  monitoring-plugins monitoring-plugins-basic monitoring-plugins-common monitoring-plugins-standard mysql-common
  python python-crypto python-ldb python-minimal python-samba python-talloc python-tdb python2.7
  python2.7-minimal rpcbind samba-common samba-common-bin samba-libs smbclient snmp
Suggested packages:
  lrzip cups-common libcrypt-des-perl libdigest-hmac-perl libio-socket-inet6-perl lm-sensors snmp-mibs-downloader
  icinga | icinga2 nagios-plugins-contrib fping postfix | sendmail-bin | exim4-daemon-heavy | exim4-daemon-light
  qstat xinetd | inetd python-doc python-tk python-crypto-doc python-gpgme python2.7-doc binutils binfmt-support
  heimdal-clients cifs-utils
```

Step 2: Configure NRPE by opening its configuration file in /etc/nagios/nrpe.cfg

```
# SERVER ADDRESS
# Address that nrpe should bind to in case there are more than one interface
# and you do not want nrpe to bind on all interfaces.
# NOTE: This option is ignored if NRPE is running under either inetd or xinetd

server_address=10.149.0.53
```

Step 3: Add Nagios server IP address in the ‘allowed_hosts’ attribute,

```
# Note: The daemon only does rudimentary checking of the client's IP
# address. I would highly recommend adding entries in your /etc/hosts.allow
# file to allow only the specified host to connect to the port
# you are running this daemon on.

# NOTE: This option is ignored if NRPE is running under either inetd or xinetd
allowed_hosts=127.0.0.1, 192.168.1.50
```

Save and exit the configuration file.

Step 4: Restart NRPE service and verify its status

- # systemctl restart nagios-nrpe-server

```
root@ubuntu-nagios-client:~#
root@ubuntu-nagios-client:~# systemctl restart nagios-nrpe-server
root@ubuntu-nagios-client:~#
root@ubuntu-nagios-client:~# systemctl status nagios-nrpe-server
● nagios-nrpe-server.service - Nagios Remote Plugin Executor
   Loaded: loaded (/lib/systemd/system/nagios-nrpe-server.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2019-10-31 21:13:42 UTC; 13s ago
     Docs: http://www.nagios.org/documentation
 Process: 6592 ExecStopPost=/bin/rm -f /var/run/nagios/nrpe.pid (code=exited, status=0/SUCCESS)
 Main PID: 6595 (nrpe)
    Tasks: 1 (limit: 4395)
   CGroup: /system.slice/nagios-nrpe-server.service
           └─6595 /usr/sbin/nrpe -c /etc/nagios/nrpe.cfg -f
```

2. Configure Nagios Server to monitor Linux host

Step 1: Log in to Nagios Server and install EPEL (Extra packages for Enterprise Linux) package.

- # dnf install <https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm>

Install NRPE plugin on the server

- # dnf install nagios-plugins-nrpe -y

Step 2: Uncomment the line below in the configuration file

- cfg_dir=/usr/local/nagios/etc/servers

```
# You can also tell Nagios to process all config files (with a .cfg  
# extension) in a particular directory by using the cfg_dir  
# directive as shown below:  
  
cfg_dir= /usr/local/nagios/etc/servers  
cfg_dir=/usr/local/nagios/etc/printers  
cfg_dir=/usr/local/nagios/etc/switches  
cfg_dir=/usr/local/nagios/etc/routers
```

Step 3: Create a configuration directory

- # mkdir /usr/local/nagios/etc/servers

Step 4: create client configuration file

- # vim /usr/local/nagios/etc/servers/ubuntu-host.cfg

Copy and paste the configuration below to the file.

```
define host{  
  
    use          linux-server  
  
    host_name    ubuntu-nagios-client  
  
    alias        ubuntu-nagios-client  
  
    address      10.128.0.53  
  
}
```

```
define hostgroup{

    hostgroup_name      linux-server

    alias              Linux Servers

    members            ubuntu-nagios-client

}
```

```
define service{

    use                local-service

    host_name          ubuntu-nagios-client

    service_description SWAP Usage

    check_command      check_nrpe!check_swap

}


```

```
define service{

    use                local-service

    host_name          ubuntu-nagios-client

    service_description Root / Partition

    check_command      check_nrpe!check_root

}
```

```

}

define service{

    use          local-service

    host_name    ubuntu-nagios-client

    service_description  Current Users

    check_command   check_nrpe!check_users

}

define service{

    use          local-service

    host_name    ubuntu-nagios-client

    service_description  Total Processes

    check_command   check_nrpe!check_total_procs

}

define service{

    use          local-service

    host_name    ubuntu-nagios-client

    service_description  Current Load

    check_command   check_nrpe!check_load

}

```

Step 5: Save and exit the configuration file.

Verify that there are no errors in Nagios configuration

- # /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

Step 6: Open port 5666 which is used by NRPE plugin on the firewall of the Nagios server.

```
[root@centos-8 servers]#
[root@centos-8 servers]# firewall-cmd --permanent --add-port=5666/tcp
success
[root@centos-8 servers]# firewall-cmd --reload
success
[root@centos-8 servers]# █
```

Step 7: Go to your Linux host (Ubuntu 18.04 LTS) and allow the port on UFW firewall.

```
root@ubuntu-nagios-client:~#
root@ubuntu-nagios-client:~# ufw allow 5666/tcp
Rule added
Rule added (v6)
root@ubuntu-nagios-client:~# ufw reload
Firewall reloaded
root@ubuntu-nagios-client:~#
root@ubuntu-nagios-client:~#
root@ubuntu-nagios-client:~# ufw status
Status: active

To                         Action      From
--                         ----      --
5666/tcp                   ALLOW      Anywhere
5666/tcp (v6)              ALLOW      Anywhere (v6)
```

Result: The Setup to perform monitoring of Port, Service monitoring, Windows/Linux server monitoring using Nagios is successfully done.

This experiment helped me to get knowledge of Working of Nagios.

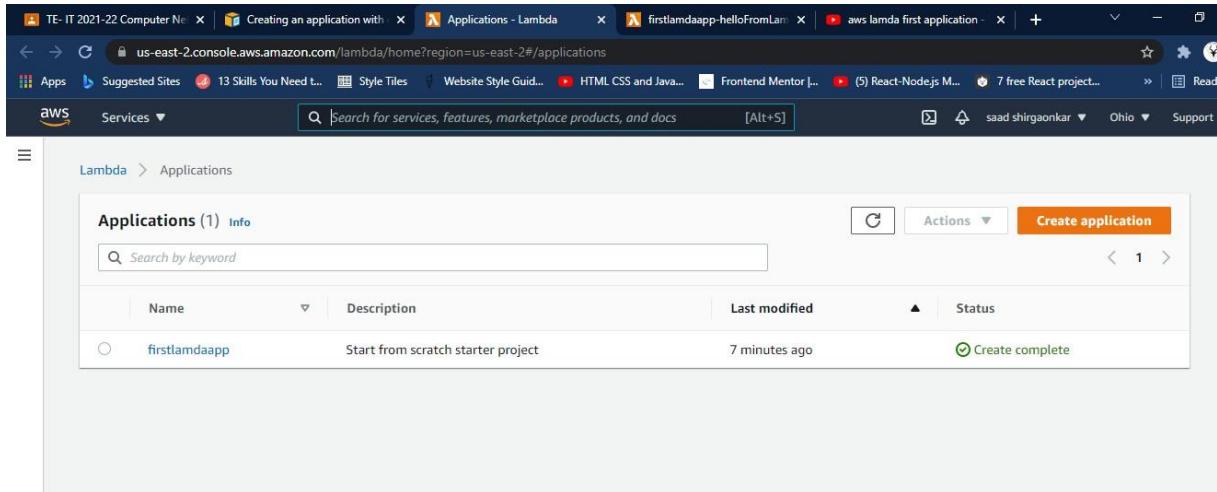
Conclusion: The modules of the experiment were successfully performed. The monitoring setup was successfully done.

Experiment No: 11

Aim: To understand AWS Lambda, its workflow, various functions and create your first Lambda functions using Python / Java / Nodejs.

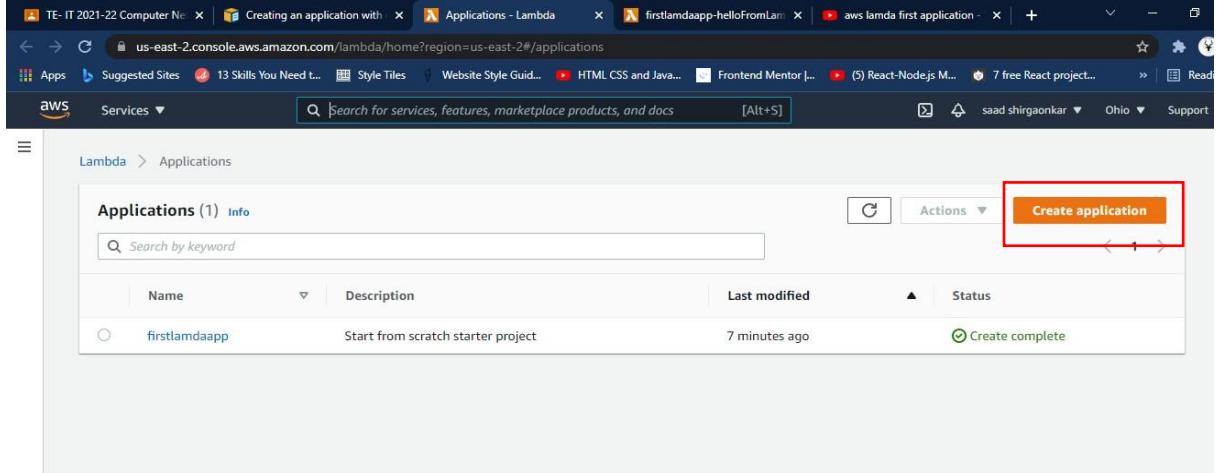
Solution:

Step 1: Open the Lambda console



The screenshot shows the AWS Lambda Applications console. The URL in the browser is `us-east-2.console.aws.amazon.com/lambda/home?region=us-east-2#/applications`. The page displays a table titled "Applications (1) Info". The table has columns: Name, Description, Last modified, and Status. There is one entry: "firstlambdaapp" with the description "Start from scratch starter project", last modified "7 minutes ago", and status "Create complete". A red box highlights the "Create application" button at the top right of the table header.

Step 2: Choose Create application.



This screenshot is identical to the previous one, showing the AWS Lambda Applications console with the "firstlambdaapp" application listed. The "Create application" button is again highlighted with a red box.

Step 3: Choose Author from scratch.

AWS Lambda

Queue processing

Use an AWS Lambda function to process messages from an Amazon SQS queue. With Amazon SQS, you can offload tasks from one component of your application by sending them to a queue and processing them asynchronously. Lambda polls the queue and invokes your function.

Made by: AWS

Uses: Lambda, SQS

Runtime: Node.js 14.x

Other options

AWS Serverless Application Repository

Deploy an application from the AWS Serverless Application Repository (pipeline not included).

Author from scratch

Create a repository and pipeline to use with your own application.

Author from scratch

Step 4: Configure application settings and Choose Create..

AWS Lambda

Configure your application

Application details

Name
firstlambdaapp

Use only lowercase letters, numbers, or hyphens. The maximum length is 20 characters.

Language

Runtime
Node.js 14.x

Template format [Info](#)

AWS SAM (YAML)

AWS CDK (TypeScript)

Feedback English (US) ▾

© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.

Privacy Policy

Terms of Use

Cookie preferences



AWS Lambda

Source control service

Choose where to create your application's Git repository.

CodeCommit

Create a repository in your AWS account. Manage SSH keys and HTTP credentials for users in the IAM console.

CodeStar Connections

Create a private repository in your GitHub account using CodeStar Connections.

Repository name
firstlambdaapp

The maximum length is 100 characters.

Permissions [Info](#)

Create roles and permissions boundary

Lambda needs permission to create IAM roles for the resources that support your application. It also needs permission to create a permissions boundary that limits the permissions that can be granted to Lambda functions by modifying execution roles in the application template. [Learn more](#)

Create

Your Lambda Application is Successfully created

The screenshot shows the AWS CloudFormation console with two main sections: 'Resources (2)' and 'Infrastructure (9)'. In the 'Resources' section, there is one Lambda Function named 'HelloFromLambdaFunction'. In the 'Infrastructure' section, there are nine resources including CloudFormationRole, CodeBuildProject, CodeCommit Repo, PermissionsBoundaryPolicy, ProjectPipeline, S3Bucket, SourceEvent, and ToolChainRole.

Logical ID	Physical ID	Type	Last modified
HelloFromLambdaFunction	firstlambdaapp-helloFromLambdaFunction-jITnbyR97pOe	Lambda Function	43 seconds ago

Logical ID	Physical ID	Type	Last modified
CloudFormationRole	firstlambdaapp-us-east-2-CloudFormationRole	IAM Role	3 minutes ago
CodeBuildProject	firstlambdaapp	CodeBuild Project	3 minutes ago
CodeCommit Repo	4b09e8c9-d0a4-4d02-bba6-94b58d317f42	CodeCommit Repository	4 minutes ago
PermissionsBoundaryPolicy	arn:aws:iam::960934202107:policy/firstlambdaapp-us-east-2-PermissionsBoundary	IAM ManagedPolicy	4 minutes ago
ProjectPipeline	firstlambdaapp-Pipeline	CodePipeline Pipeline	3 minutes ago
S3Bucket	aws-us-east-2-960934202107-firstlambdaapp-pipe	S3 Bucket	3 minutes ago
SourceEvent	firstlambdaapp-SourceEvent	Events Rule	2 minutes ago
ToolChainRole	firstlambdaapp-us-east-2-ToolChain	IAM Role	3 minutes ago

Hello world Example in Node.js using lambda

The screenshot shows the AWS Lambda code editor with the file 'hello-from-lambda' selected. The code is a simple Node.js function that returns a string 'Hello from Lambda!'. The code editor interface includes a toolbar with File, Edit, Find, View, Go, Tools, Window, Test, Deploy, and a status bar indicating 'Changes deployed'.

```
1 /**
2  * A Lambda function that returns a string.
3  */
4 exports.helloFromLambdaHandler = async () => {
5   // If you change this message, you will need to adjust tests in hello-from-lambda.test.js
6   const message = 'Hello from Lambda!';
7
8   // All log statements are written to CloudWatch by default. For more information, see
9   // https://docs.aws.amazon.com/lambda/latest/dg/nodejs-prog-model-logging.html
10  console.log(message);
11
12  return message;
13};
14
```

Result: The Creation of lambda Application is successfully done.

Conclusion: The importance of AWS Lambda, its workflow, various functions were learned in this experiment and the first lambda application is successfully done.

Experiment No: 12

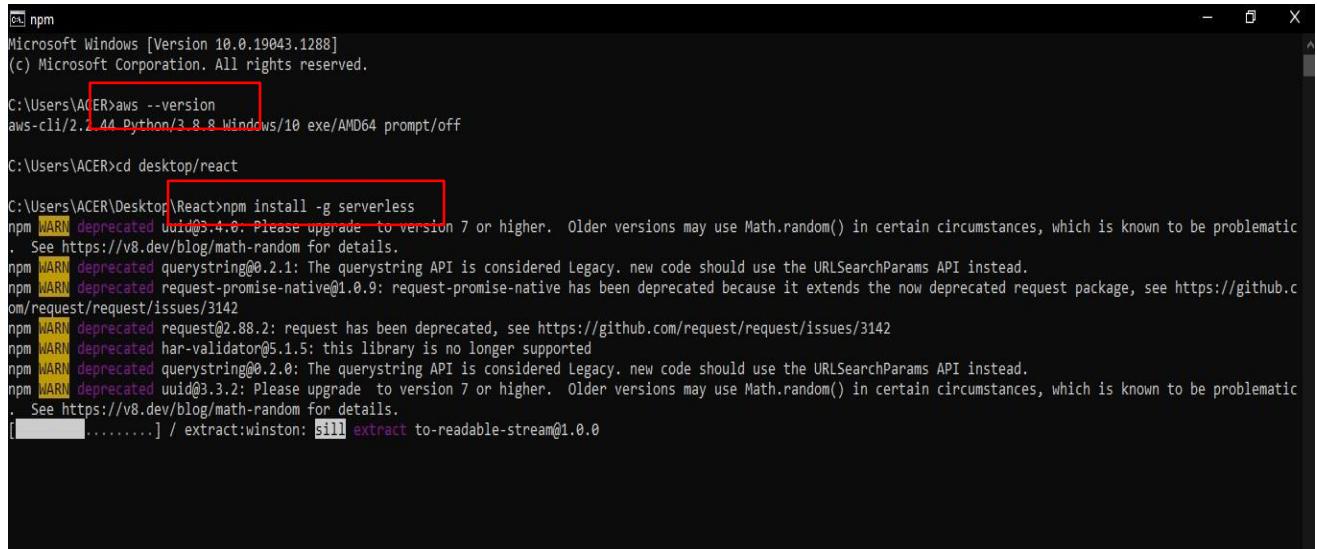
Aim: To create a Lambda function which will log “An Image has been added” once you add an object to a specific bucket in S3.

Solution:

Required Tools

- Node JS 12
- Serverless
- AWS CLI

Step 1: Install AWS CLI and Serverless Framework.



```
npm
Microsoft Windows [Version 10.0.19043.1288]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ACER>aws --version
aws-cli/2.1.44 Python/3.8.8 Windows/10 exe/AMD64 prompt/off

C:\Users\ACER>cd desktop/react

C:\Users\ACER\Desktop\React>npm install -g serverless
npm WARN deprecated uuid@3.4.8: Please upgrade to version 7 or higher. Older versions may use Math.random() in certain circumstances, which is known to be problematic
. See https://v8.dev/blog/math-random for details.
npm WARN deprecated querystring@0.2.1: The querystring API is considered Legacy. new code should use the URLSearchParams API instead.
npm WARN deprecated request-promise-native@1.0.9: request-promise-native has been deprecated because it extends the now deprecated request package, see https://github.com/request/request/issues/3142
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
npm WARN deprecated querystring@0.2.0: The querystring API is considered Legacy. new code should use the URLSearchParams API instead.
npm WARN deprecated uuid@3.3.2: Please upgrade to version 7 or higher. Older versions may use Math.random() in certain circumstances, which is known to be problematic
. See https://v8.dev/blog/math-random for details.
[.....] / extract:winston: sill extract to-readable-stream@1.0.0
```

Step 2: Run the following command to generate sample code with serverless.

```

C:\Windows\system32\cmd.exe
> protobufjs@6.11.2 postinstall C:\Users\ACER\AppData\Roaming\npm\node_modules\serverless\node_modules\protobufjs
> node scripts/postinstall

> aws-sdk@2.1012.0 postinstall C:\Users\ACER\AppData\Roaming\npm\node_modules\serverless\node_modules\aws-sdk
> node scripts/check-node-version.js

> serverless@2.64.1 postinstall C:\Users\ACER\AppData\Roaming\npm\node_modules\serverless
> node ./scripts/postinstall.js

  Serverless Framework successfully installed!
  To start your first project run "serverless".

npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@~2.3.2 (node_modules\serverless\node_modules\chokidar\node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: numpy@6.3.5 (node_modules\serverless\node_modules\numpy):
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: numpy@6.3.5 install: 'prebuild-install || node-gyp rebuild'
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: Exit status 1

+ serverless@2.64.1
added 598 packages from 410 contributors in 75.976s

C:\Users\ACER\Desktop\React>sls create --template hello-world
Serverless: Generating boilerplate...
Serverless: Successfully generated boilerplate for template: "hello-world"
Serverless: NOTE: Please update the "service" property in serverless.yml with your service name
C:\Users\ACER\Desktop\React>

```

Step 3: Install dependencies

```

C:\Windows\system32\cmd.exe
C:\Users\ACER\Desktop\React\imageupload-lambda>sls create --template hello-world
Serverless: Generating boilerplate...
Serverless: Successfully generated boilerplate for template: "hello-world"
Serverless: NOTE: Please update the "service" property in serverless.yml with your service name
C:\Users\ACER\Desktop\React\imageupload-lambda>npm init -y
Wrote to C:\Users\ACER\Desktop\React\imageupload-lambda\package.json:

{
  "name": "imageupload-lambda",
  "version": "1.0.0",
  "description": "",
  "main": "handler.js",
  "scripts": {
    "test": "echo \'Error: no test specified\' && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}

C:\Users\ACER\Desktop\React\imageupload-lambda>npm install busboy && uuid && jimp && aws-sdk
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN imageupload-lambda@1.0.0 No description
npm WARN imageupload-lambda@1.0.0 No repository field.

+ busboy@0.3.1
added 3 packages from 1 contributor and audited 3 packages in 0.865s
found 0 vulnerabilities

'uuid' is not recognized as an internal or external command,
operable program or batch file.
C:\Users\ACER\Desktop\React\imageupload-lambda>

```

Step 4: Create the function to decode the multipart/form-data

```

formParser.js - Notepad
File Edit Format View Help
const Busboy = require('busboy');

module.exports.parser = (event, fileZise) =>
  new Promise((resolve, reject) => {
    const busboy = new Busboy({
      headers: {
        'content-type': event.headers['content-type'] || event.headers['Content-Type']
      },
      limits: {
        fileZise
      }
    });
    const result = {
      files: []
    };
    busboy.on('file', (fieldname, file, filename, encoding, mimetype) => {
      const uploadFile = {}
      file.on('data', data => {
        uploadFile.content = data
      });
      file.on('end', () => {
        if (uploadFile.content) {
          uploadFile.filename = filename
          uploadFile.contentType = mimetype
          uploadFile.encoding = encoding
          uploadFile.fieldname = fieldname
          result.files.push(uploadFile)
        }
      })
    })
    busboy.on('field', (fieldname, value) => {
      result[fieldname] = value
    })
  })

```

Step 5: Function that will process and upload the images to S3

```

fileUploaderHome.js - Notepad
File Edit Format View Help
"use strict";
const AWS = require("aws-sdk")
const uuid = require("uuid/v4")
const Jimp = require("jimp")
const s3 = new AWS.S3()
const formParser = require("./formParser")

const bucket = process.env.Bucket
const MAX_SIZE = 4000000 // 4MB
const PNG_MIME_TYPE = "image/png"
const JPEG_MIME_TYPE = "image/jpeg"
const JPG_MIME_TYPE = "image/jpg"
const MIME_TYPES = [PNG_MIME_TYPE, JPEG_MIME_TYPE, JPG_MIME_TYPE]

module.exports.handler = async event => {
  const getErrorMessage = message => ({ statusCode: 500, body: JSON.stringify( message )})

  const isAllowedfile = (size, mimeType) => { // some validation code }

  const uploadToS3 = (bucket, key, buffer, mimeType) =>
    new Promise((resolve, reject) => {
      s3.upload(
        { Bucket: bucket, Key: key, Body: buffer, ContentType: mimeType },
        function(err, data) {
          if (err) reject(err);
          resolve(data)
        }
      )
    })
  }

  const resize = (buffer, mimeType, width) =>
    new Promise((resolve, reject) => {
      Jimp.read(buffer)
        .then(image => image.resize(width, Jimp.AUTO).quality(70).getBufferAsync(mimeType))
        .then(resizedBuffer => resolve(resizedBuffer))
        .catch(error => reject(error))
    })
}

```

Step 6: Build serverless.yml file.

```
File Edit Selection View Go Run Terminal Help  
serverless.yml - Visual Studio Code  
! serverless.yml x  
C:\Users\ACER\Desktop\React\imageupload-lambda> ! serverless.yml  
1 service: file-UploaderService-foqc-home  
2 custom:  
3   bucket: lambda-test-foqc-file-home  
4 provider:  
5   name: aws  
6   runtime: nodejs12.x  
7   region: us-east-1  
8   stackName: fileUploaderHome  
9   apiGateway:  
10    binaryMediaTypes:  
11      - '*/*'  
12    iamRoleStatements:  
13      - Effect: "Allow"  
14    Action:  
15      - "s3:PutObject"  
16      - "s3:GetObject"  
17    Resource:  
18      - "arn:aws:s3:::${self:custom.bucket}/*"  
19 functions:  
20   UploadFileHome:  
21     handler: fileUploaderHome.handler  
22     events:  
23       - http:  
24         path: upload  
25         method: post  
26         cors: true  
27     environment: Bucket: ${self:custom.bucket}  
28 resources:  
29   Resources:  
30     StorageBucket:  
31       Type: "AWS::S3::Bucket"  
32       Properties:
```

Step 7: Deploy the code using the following command.

- sls deploy --stage=test

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19043.1288]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ACER\Desktop\React\imageupload-lambda>sls deploy --stage=test

Serverless Error -----
Cannot parse "serverless.yml": bad indentation of a mapping entry in "C:\Users\ACER\Desktop\React\imageupload-lambda\serverless.yml" (27:24)

 24 |           path: upload
 25 |           method: post
 26 |           cors: true
 27 |           environment: Bucket: ${self:custom.bucket}
-----^
 28 | resources:
 29 | Resources:

Get Support -----
  Docs:      docs.serverless.com
  Bugs:      github.com/serverless/serverless/issues
  Issues:    forum.serverless.com

Your Environment Information -----
  Operating System:      win32
  Node Version:         14.17.5
  Framework Version:   2.64.1
  Plugin Version:      5.5.0
  SDK Version:         4.3.0
  Components Version:  3.17.1
```

Step 8: Test Your API

POST https://aa3jr8d1w6.execute-api.us-east-1.amazonaws.com/test/upload

Params Authorization Headers **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

KEY	VALUE
<input checked="" type="checkbox"/> file	photo-1510915228340-29c85a43dcfe.jpeg 
Key	Value

Response

Step 9: To conclude, in case you need to remove the service, run the following command.

- `sls remove --stage=test`

Step 10: Check the result in S3 Bucket.

The screenshot shows the AWS S3 console interface. On the left, there's a sidebar with 'Buckets' selected. The main area displays an 'Account snapshot' with a link to 'View Storage Lens dashboard'. Below it is a table titled 'Buckets (3) Info' showing three buckets:

Name	AWS Region	Access	Creation date
aws-us-east-2-960934202107-firstlambdaapp-pipe	US East (Ohio) us-east-2	Objects can be public	October 22, 2021, 15:36:18 (UTC+05:30)
first-demo-input-bucket	US East (Ohio) us-east-2	Bucket and objects not public	October 8, 2021, 19:21:49 (UTC+05:30)
first-demo-output-bucket	US East (Ohio) us-east-2	Bucket and objects not public	October 8, 2021, 19:22:34 (UTC+05:30)

Result: The Image upload with AWS lambda function is successfully completed.

Conclusion: The experiment helped me to understand the importance of AWS Lambda and its use. The modules of the experiment have been successfully completed.