ADEV Lab

Experiments :-

1. To setup AWS Cloud9 IDE. Launch AWS cloud9 IDE & perform collaboration
2. To build your application using AWS codebuild & deploy on S3/SEBS using AWS Codepipeline.
3. To understand the kubernetes cluster arch., install & spin up a kubernetes cluster on linux m/c / cloud platforms.
4. To install kubectl & execute kubectl commands to manage the kubernetes cluster & deploy your 1st kubernetes app.
5. To understand terraform lifecycle & install it on linux m/c. (Installation)
6. To build change, destroy AWS/GCP/ Microsoft azure / Digital ocean infrastructure using terraform.
7. To understand static analysis SAST process & learn to integrate Jenkins SAST to SonarQube/GitLab (Installati
8. Create a Jenkins CICD pipeline with SonarQube / GitLab Integration to perform a static analysis of the code to detect bugs, code smells & Security vulnerabilities on a sample web/app like Java, python

3-4 Kubernetes
5-6 Terraform
7-8 Jenkins/SonarQube
9-10 Nagios
11-12 AWS Lambda

Page No.
Date

9. To understand Continous monitoring, installation & Configuration of Nagios Core, Nagios plugins & NRPE (Nagios Remote plugin Executor) on linux m/c
10. To perform port, Service monitoring, Window/Linux Server monitoring using Nagios
11. To undestand AWS Lambda, its workflow, various functions & create your first lambda functions using py/Java/Node.js
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.

Exp 1 - AWS cloud9

AWS Codestar

AWS Code Commit → AWS Code Build → AWS Code Deploy

AWS Code pipeline

- AWS cloud9 – A cloud IDE for writing, debugging & running code.

- Steps -

1) Login with AWS root account
2) open IAM [Identity & access management]
   Add 2 users – Select password
3) Create group.
   grp name – ____
   grp policy – AWS cloud9 ~~Environment~~ Member User
   * ↳ copy link
4) open cloud9
   ↳ paste the URL of copied IAM user
   Login as IAM user1
5) Create Environment
   Developer tools
6) Create new file on AWS CLI

If Particular file is not running then stopped last running file

Page No. 2
Date

7) Write any code & try to run it.
8) For collaborating this file with other user
   paste IAM user URL in incognito tab
9) Login there as a IAM user2
10) open cloud 9
11) Move to IAM user1
    - click on share
    - Enter username - IAM user 2
12) Move to IAM user2
    - click on shared with you
    - open IDE

Exp 2 - AWS Code Pipeline

- CI/CD
  Continous Integration / Continous Development
- Automate the complete workflow from build, test, packaging, ~~deploy~~ deploy which will be triggered when there are any changes to an exsisting app.
- CI/CD
  New Code → | Src | build test | Stage | Prod → Production-Ready Code
  Pipeline
- Advantages -
  1) frequent releases
  2) Low risk
  3) Increased productivity

Commit → Trigger build → Build → Notify (stagewise) → Run test → Notify → Deliver build Enviro.
Change [Repository]
EB

Git, AWS CodeCommit, Bitbucket

- EBS - Elastic Block Storage
- S3 - Simple Storage Service
  Storage Component - Bucket
- EB - Elastic Bean Stalk - used for deploy web app
  - Single instance
  - Load balancing & Scaling
  Two tiers / Web server / Worker
  - gives unique domain name

Implementation —

1) Via the AWS Management Console
2) Via EB CLI
3) Via AWS Toolkit for Eclipse & VS IDE

Steps - Root Account

1) Create environment using Elastic Beanstalk [Applications]
   - Application name ——
   - Platform Tomcat
   - Sample App

   click on your website after successfully creating enviroment → Congratulation ! HTML Page.

2) Ready Code in the respository
   - go to github → Create 2 file → buildspec.yml, index.html

3) Create AWS Code pipeline
   - go to CodePipeline
   - Create Pipeline
   - Pipeline name ——
   - Source Provider github Version 2
   - Connect to github (1st time)
     - Connection name - github mail id
     - Authorize AWS ✓
     - Connector Install
   - Repository name
   - Branch name
   - Build provider - AWS CodeBuild
   - ↳ Create project
     - OS - Ubuntu
     - Runtime - Standard
     - Image - 4.0

4) Deploy
- deploy Provider – AWS E Beanstalk
- Application name – SanyuktaCodeBuildDemoWeb
- Enviro name – .env

After Done....
→ click on Link which is there on Deploy
→ Now on your Application – click on URL

5) Go to github
→ edit index.html then commit changes
→ you'll seee Source ✓
↓
Build ✓
↓
Deploy ✓

Version 2

Deletion :– 1] EC2 · Instances
· Volumes

2] Codepipeline · pipeline

3] ElasticBeanstalk · Environment – Actions
· Applications

Exp 3 - Kubernetes Cluster SpinUP

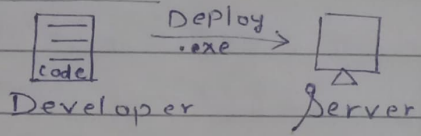
Google - open Source

▷ **Kubernetes** - Used to managed Container [K8S]

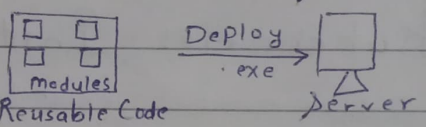
- Traditional deployment
- Virtualized deployment
- Container deployment

need for microservices - Handle Scalability

① Traditional -

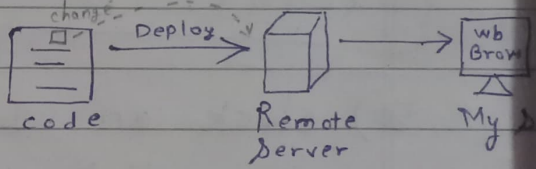


② Modular -

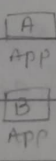


Monolithic

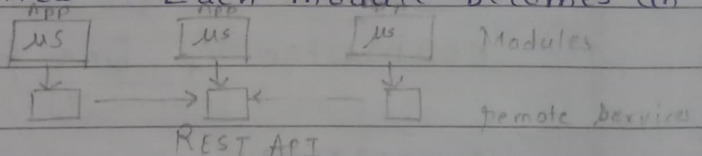
③ Web Application -



Microservices



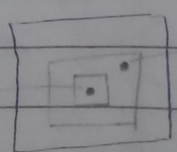
④ Microservices - Each module becomes an app



→ Saving time for deployment. faster
→ Scale separately
→ multiple tech stack

This μs are deployed in **Containers**

Container -



μs

▷ Creation of this Container using **docker**

▷ Management of this container using Kubernetes, dockerSworm

isolated Context which the app can run along its environment

Delete → 1) Instances
2) Security groups
3) Volumes
4) Key Pairs

Page No. 9
Date

exp4 Steps -

1) Go to EC2
→ instances → Launch an instance

x [Not date] 1. No. of instances + 1
[Out out to the old experience]
[Remain in old experience]

2) choose an AMI

- Ubuntu
- 20.04 LTS
- No. of inst = 2
- HTTP ✓
- HTTPS ✓
- SSH ✓

- Ubuntu Server 20.04 LTS (HVM)
- t2.micro
- No. of instances — 2
- ssh — Anywhere IPV4
- HTTP — Anywhere IPV4

3) ☑ Select an existing key Pair
- Create a new key pair
- RSA
- Key pair name ——— → Download key
- Launch instances.

4) Edit name of an instances.
☐ K8SMaster
☐ K8SWorker

5) click on 1 instance at a time then click on Connect So open 2 Separate windows (Tabs) One for K8SMaster & One for K8SWorker

connect

☑ K8SMaster

- Connect

☑ K8SWorker

- Connect

[EC2 Instance Connect]

6) K8S Master | K8S Worker

K8S Master	K8S Worker
> clear	> clear
> Sudo su	> Sudo su
Set the Server as M/W node > hostnamectl Set-hostname Master-node	> hostnamectl Set-hostname Worker-node
> Sudo su	> sudo su
> tput Setaf 02	> tput setaf 01

• tput Setaf 02 • tput Setaf 01

(A) Docker Installation

Install:
> apt-get update
> apt-get install docker.io -y
> docker --version

Start & enable:
> Systemctl enable docker
> Systemctl status docker • q press
> Systemctl start docker.

(B) Kubernetes Installation

Add K8S signing key:
> Curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add

To check the Software's authenticity for downloading K8S

Add S/W Repository:
> apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"
↑ space

To add K8S to the default repo

If errors occurs

> Sed -i -e '(entry no.)d' /etc/apt/sources.list

(Residue Error)

> apt-get update

>> [command run again]

Kubernetes Installation Tools

> apt-get install Kubeadm Kubelet Kubectl -y

> apt-mark hold Kubeadm Kubelet Kubectl.

> Kubeadm Version

(C) Kubernetes Deployment

— Begin K8S deploy.. by disabling swap memory

> swapoff --a

>

>

- ☑ K8s Worker
  → Security
  → click on link ——— (launch-wizard-1)
  → Edit Inbound Rules

Add Rule	Custom TCP	TCP	6443	Anywhere IPV4

Save Rule

K8s Master

> kubeadm init --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=all

> mkdir -p $HOME/.kube — Create directory for cluster

> cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

> chown $(id-u): $(id-g) $HOME/.kube/config

> kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

> kubectl get pods --all-namespaces — Verify the comm

You have to copy that Para:
Then you can join ——
kubeadm join ——
-- discovery ——

Kubeadm join msg
(Used to join Worker node to cluster)

Paste th

K8s Mas

> kube

K8S Worker

> Paste that Para here as Command

K8s Master

> kubectl get nodes

o/p -

master-node
Worker-node

} Part of cluster & Ready to run workloads

onfig

"Pod Network"
Allow Comm bet^n different nodes in the cluster

kube-flannel.yml

Verify the communication, Running

Exp 5. Terraform Installation on linux.

- cloud computing →
  SAAS – Software As A Service
  PAAS – Platform As A Service
  IAAS – Infrastructure As A Service

- Infrastructure as a Code.
  - AWS ×
  - Azure
  - GCP

  get the infrastructure in Script like code.
- Extension → .tf

AWS	Terraform
↓	↓
7 Steps to launch instance	Just write Code in Script

- Terraform → · cloud computing provider
  · keep infrastructure in Compliant (stdzed) state.
  · Automate infrastructure.

- Kubernetes Vs Terraform
  open-Source Infrastruc.
  Container in Script like Code

- Terraform language → HashiCorp Configuration Language (HCL)

- Terraform Workflow → 1. Scope 2. Author 3. Initialize 4. Plan 5. Apply

- Commands [components] → Tf
  1. Init
  2. Plan
  3. apply
  4. destroy

Steps:

1) Go to EC2 → instances → launch an instance [opt out to old exp]
2) choose an AMI
   - Ubuntu Server 20.04 LTS (HVM)
   - t2.micro
   - no. of instances 1
   - Everything Same.
3) Select key pair
   - Terraform RSA
4) Launch Instance.
   - rename - hostPC-terraform
   - Connect
   - EC2 instance Connect.
5) Commands
   - → Sudo su
   - → Wget Paste
   - → apt install unzip
   - → unzip terraform...386.zip
   - → ll
   - → terraform or terraform -v
     Not install show → Set Path
   - → mv terraform /usr/local/bin/
   - → terraform ✓

- go to terraform official website
- Downloads
- 386 Linux
- Copy link address

EXP 4. Install Kubectl & execute Kubectl commands to manage K8S clusters.

K8S Master

> Kubectl create deployment nginx --image=nginx

> Kubectl expose deploy nginx --port 80 --target-port80 --type Nodeport

> kubectl get Services

Ports

K8S

nginx 80:nginx_port/TCP

>30000

~~> Kubectl create deployment nginx --image=nginx~~

> Kubectl Scale --current-replicas=1 --replicas=2 deployment/nginx

> kubectl get pods

> Kubectl describe deployment/nginx

K8S Worker

> docker ps

K8s Master

> Kubectl delete Service nginx
> Kubectl get Services
> Kubectl delete deployment nginx
> Kubectl get deployments

Exp 6: To build, change, destroy AWS infra using terraform

- go to EC2
- Launch instance
  → choose AMI – Ubuntu 20.04
  → 1 instance
  → Security group ☐ SSH ☐ terraform
  → key pair – Terraform.psm
  → launch instance

rename → terraform – demo

- ☑ teraform – demo
  Connect instance EC2
- EC2 instance Connect [Command]

Terraform Installation

```
> clear
> Sudo su
> wget  Paste link
> apt install unzip
> unzip terraform...386.zip
> terraform (NoT found)
> mv terraform /usr/local/bin/
> terraform ✓
```

- go to official site of terraform
- Downloads
- 386 Linux
- Copy link

Build Terraform

```
> touch main.tf   Create a file named main
> nano main.tf    Edit a file
```

ADD

go to google Search for
"Terraform on AWS"
↳ AWS documentation
• Example usage
usage [Script]

- go to IAM
- Add user → User name → Next
  Access key - Programmatic access
  Set permission → Attach existing policy
  ☑ Administrator

```
Provider "aws" {
    region     = "us-east-1"
    access-key = "          "
    Secret-key = "          "
}
```

User → Access Key ID, Secret access key

```
resource "aws_instance"
         "terraform-VIT" {
    ami           = "          "
    instance_type = "t2.micro"
}
```

ami-number

ctrl + S Save it
ctrl + X Exit

```
> terraform init      Initialize the terraform
> terraform plan -lock=false
> terraform apply -lock=false
  Enter a value: yes
```

- For confirming the infra is Created or not
  - → go to EC2
  - → check the instance. (new as per main file)

Change Terraform

- Edit the instance name

```
> nano main.tf
```

> Repeat it

change

```
Provider {
}
resources {
}
tags = {
}   Name = "Terraform-DemoInfra"
```

ctrl + S
ctrl + X

Destroy Terraform

> terraform destroy

Enter a value: yes

Exp 7 : To understand Static analysis SAST process & integrate Jenkins

- **Jenkins :-**

- open source automation server which helps to build, test & deploy the software by facilitating Continous Integration & Continous Delivery (CI/CD)
- Devops tool writted in Java
- Used to implement CI/CD workflows called as pipelines
- Runs in a Servlet container → Apache Tomcat

Jk Same as Aws Codepipeline

- pipeline → collection of jobs that brings s/w from version control into hands of end-users by using aut. Tools.

Installation of Jenkins →

- Go to EC2
- Launch an instance
  - → step 1 - AMI Ubuntu Server 20.04 LTS
  - → step 2 - t2.micro
  - → No. of inst. 1
  - → Security group — name - Jenkins, Description - -11- — Edit Inbound Rules
- Edit name - Jenkins-Demo
- Connect
  - ↳ EC2 instance Connect

Anywhere IPV4	HTTPS
Anywhere IPV4	SSH
Anywhere IPV4	HTTP
Anywhere IPV4	8081 Custom TCP

1. Install Java
> Sudo su
> apt update.
> java --version
> apt install openjdk-11-jre
> java --version

2. Install Jenkins
> curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo tee \
/usr/share/keyrings/~~jen~~ jenkin-keyring.asc
> /dev/null

> echo deb _____ /dev/null
> apt-get update
> apt-get install jenkins

3. Enable, ~~Start~~, ~~Status~~ Jenkins
> systemctl enable jenkins
> Systemctl start jenkins
> systemctl status jenkins
> systemctl edit jenkins

To make PID visible

```
[Service]
Environment="JENKINS_PORT=8081"
```

ctrl+s
ctrl+x
> systemctl restart jenkins

Edit Inbound rule

8080 → P/W invisible
8081 → P/W ~~Visible~~

Page No. 24
Date

- go to Search bar, Enter :

  public ip address : 8081

- Administrator password

  you'll get this on your command terminal

  // Please use follow. passw to Proceed to install :

  xyPqtmnoP

- Install Suggested plugins.
- Create account using your own credentials.

  Jenkins Dashboard

- SonarQube :-

- open - source platform developed by SonarSource for continous inspection of code quality to perform automatic reviews with static analysis of code to detect bugs, code smells & security vulnerabilities on 2ot prog. lang
- features →
  1) detect bugs
  2) code smells
  3) Security vulnerability
  4) Centralize quality (All projects in 1 Place)
  5) Clean code

Installation of SonarQube →

t2.small
₹3 per 2 Hrs.

- Go to EC2
- Launch an instance
  - → Step 1 - AMI Ubuntu Server 20.04
  - → Step 2 - choose instance type
    t2.small (2 GB Memory)
  - → No. of instance - 1
  - → Security group - sonarqube

custom TCP	TCP	9000
SSH	TCP	22

  - → key pair - Sonarkey
- Edit instance name - SonarQube-Demo
- Connect
  ↳ EC2 instance Connect

1. Install Java

```
> Sudo su
> Sudo -i
> apt update -y        (space)
> apt install openjdk-11-jre-headless y
> java --version
> apt install net-tools   } optional
> netstat -ntlp
```

2. Install SonarQube

Need zip file

Community Edition

```
> wget https://binaries.sonarsource.com ____.zip
> mv Sonarqube-9.6.1.59531.zip /opt/
> cd /opt/
> unzip Sonarqube-9.6.1.59531.zip
> apt install unzip
> unzip S_____-11-_____.zip
```

non-root user

```
> ls -la
> adduser Sonaradmin
    Enter password :
    Full name : Enter
                1
                Enter
```

root

```
> ls -la
> chown -R sonaradmin:sonaradmin
  Sonarqube-9.6.1.5931[illegible]
```

sonaradmin

```
> ls -la
> cd Sonarqube-9.6.1.59531/
> ls -la
> cd bin
> ls -la
```

> cd linux-x86-64
> ls -la

Sonar.sh

VIMP
> su sonaradmin
> ./sonar.sh start
> ./sonar.sh status

Go to search bar Enter:

PublicIPaddress : 9000

- Create account using credentials

SonarQube Dashboard — username - admin, Password - admin — Login

> netstat -ntlp

> ./sonar.sh stop

→ goto volume
Sonar-volume ☑ Detach Volume
Stop instance

Summary –

1) Launch instance —— AMI
Edit-name [Jenkins-Demo]
no. of inst. 2
t2.micro
Security -group — Jenkins
HTTP TCP 8081
8082
8083
Key-pair – Jenkins

2) Launch instance —— t2.small
All same
Edit-name [Sonarqube -Demo]
Security -group — Sonarqube
HTTP
SSH
Custom Tcp 9000
custom Tcp 9092
Key-pair – Sonarqube

3) Jenkins-Demo

▷ Connect EC2 instance
↳ Run Commands
↳ Create account & go to dashboard

SonarQube-Demo

▷ Connect EC2 instance
↳ Run Commands
↳ Create account & go to dashboard

Exp 7

1. go to Gitlab.com
   - Register [create account]
   - Sign in
   - Dashboard —

① Create → New Project/Repository
Create blank project ←

project name — SampleProjectForDemoTesting
Visibility level — Private 🔒
Project Config — ☑ Initialize Repo
☑ Enable Static...

Create project

2. Integration :-

1. Integrating SonarScanner to Jenkins

1> go to jenkins dashboard
2> New item +
item name JenkinsPipelineDemo
Select pipeline
OK
3> Configuration Save
4) Go to username → Credentials
• Stores from parent

System V | (global) → click

Add Credential

→ Kind — secret text
→ Scope — global
→ Secret — [ ] ← Pasted
→ ID — SonarToken
→ Description — SonarToken

Create

(9)

NOTEPAD
SonarToken
GitLabToken
GitLab Webhook
Jenkins Token

Page No. 30
Date

5> Go to SonarQube dashboard
- go to administrator
- My account
- Security

Tokens

Name — SonarToken
Type — Global
Expires in — 30 days

Generate

Copy

6> Go to GitLab dashboard
- go to username
- preferences
- User Settings → Access Tokens
  - Token Name — GitLabToken
  - Select Scope — ☑ Select all

Create Personal access Token

→ Your new personal access Token [ ] c

(2)
→ Kind — Secret Text
→ Scope — global
→ Secret — [ ] Pasted
→ ID — GitLabToken
→ Description — -"-

Manage Jenkins
1. Manage Credentials
2. Manage plugins
3. Global Tool config
4. Config System

7) Go to Manage Jenkins
   → Manage Credentials ... (All Tokens here)
   → plugin Manager

UPdate | Avaliable | Installed | Advanced

- search SonarQube Scanner

☑ Sonar ....
Select it

☑ GitLab
☑ GitLab API
☑ GitLab Authentication
☑ GitLab Branch Source

Install without Restart

→ Global Tool Configuration
- SonarQube Scanner

Add SQS
→ name — SonarQube Scanner
☑ Install automatically.
Save

8) Go to Configure System
→ SonarQube Servers
☑ Enviro. Variables
ADD SQ
Name — SonarQube
Server URL — https://54.89.114.14:9000
Save
(URL of SonarQube dashboard)

→ Server authent Token
- Select SonarToken

→ GitLab
- Connection name — GitLab

- GitLab host URL — https://gitlab.com
- Credentials — GitLab Token

GitLab Servers
→ Webhook ☑ Manage webhooks

Save

EXP 8: Jenkins CICD Pipeline

9) go to project JenkinsPipelineDemo
↳ Configure
- Build Triggers
☑ Build when a change is pushed to GitLab

GitLab Webhook URL: [ ] Copied!

ADVANCED

- Secret Token

[ ] Generate
Copied! (Jenkins Token)

- pipeline
→ Definition — pipeline Script from SCM
→ SCM — Git → Repository URL [ ]

- go to gitlab dashboard
- click on username
- Repository — SampleProj for Demo Testing
Clone → clone with https - Copied! *

failed!

Pasted

https://user group : Pasted @'gitlab.com/awscode/projectName.git

- Branch to build — main

- Repository Browser — GitLab
- projectURL — ★ Pasted (without .git i.e. https://gitlab.com/user-group/project)

Save

10) Go to SonarQube Dashboard
- → Projects
- → Manually
- Name — SonarQube Demo Project

setup
- CI — with Jenkins
- Devops platform — GitLab

configure Analysis

→ Create a pipeline Job

continue

Do it for 1st then Continue

- ▷ go to gitlab
- ▷ your username
- ▷ your project (Repo)
- ▷ go to settings
- ▷ click on webhooks
- ▷ → Paste Webhook URL — URL
- ▷ → Paste Jenkins Token — Secret Token
- ☑ Push Events ▲ main — Trigger
- ☑ Select all
- Add webhook

→ Create a GitLab Webhook — continue
→ Create a Jenkinsfile
① other (for JS, TS, Go...)

② Create a Sonar-Project - Properties

Copied!

- ▷ go to gitlab
- ▷ your username
- ▷ your Repository → SampleProjectForDemoTesting
- ▷ This directory → new file

1. SampleWebpageForTesting.html (code)

```
<html>
<title> Sanyukta Adhate </title>
Sample Web Page for jenkins
CICD pipeline & analysis
using SonarQube
</html>
```

commit

2. Pasted! ~~.html~~ - new file
3. Jenkinsfile - new file → copy-Paste Script inside this file.

SonarScanner

Match with SonarQubeScanner Name.

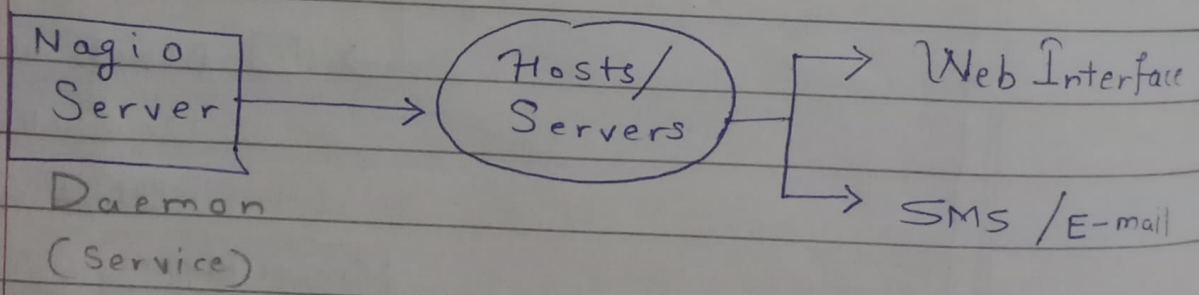
Done

→ go to Console o/p SS ✓

Exp 9-10 : Installation of Nagios & NRPE Plugin

Nagios :-

- Used for continous monitoring of System, applications, Services in a DevOps culture.
- It can alert technical staff of Problem if there is event failure. before outages affect end users.
- It is open-source.



Nagios Server is running on a host & plugins interact with local & all the remote hosts that need to be monitored.

Localhost
Remote host (other PC)

NRPE - Nagios Remote Plugin Executor

Exp 11-12 : Aws Lambda

AWS lambda :-

- Serverless compute service that lets you run your code without worrying abt managing the server.
- Can run application or backend service using AWS lambda with zero administration.
- Code which you run on AWS is called a lambda function
- Supports programming lang:
  Java, python, C#, node.js, Ruby

- Serverless → It has server but just don't require to ~~manage~~ manage any servers
  1) no server
  2) flexible scaling
  3) High avaliab
  4) No idle capacity.

▷ AWS S3, cloudWatch, IAM

STEPS :-

1> open AWS dashboard
- Search for Lambda ☆ S3 ☆

STEP 1: Create IAM Role

2> go to ~~lambda~~ IAM
↳ Roles
- Create Role
  - AWS Service
  - Lambda — Aws Use case
  - 1) lambda_full ☑
  - 2) S3 - full ☑
  - 3) cloudWatchFull ☑
  - } Add permission

RoleName – AWS Lambda Demo
Description – Allows lambda function to call AWS Service on your behalf

Create Role

Step2: ~~Create Bucket~~ Create a bucket in AWS S3 to upload image. Ensure region of bucket is same as that of lambda function.

3) go to S3
→ Buckets
→ Create bucket → bucket Name – Sanyuktademo bucket
→ AWS Region – US-east-1
→ ☐ Block all public access
Unmarked
☑ I acknowledge

Create Bucket

Sanyuktademobucket → ⊞ Upload L Wait
After 5

Step3: Create a lambda function using node.js

4) go to lambda
→ Functions
→ Create function → Use a blueprint
→ S3 – blueprints

Get S3 object nodejs

Configure → Function Name – Sanyuktademo Lambda Function S3
Execution Role – Use an exsisting Role

S3 ★ General Config
Timeout — 20 sec

Page No. 39
Date

Exsisting role
(Search) AWSLambdaDemo

↳ S3 trigger
• Bucket — Sanyuktademobucket

[Bucket Region = Lambda function Region → Same]

☑ I acknowledge

[Create Function]

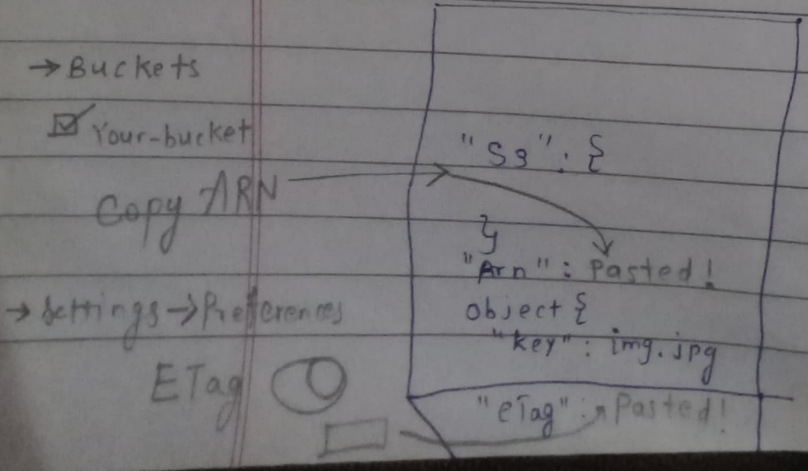
5) In lambda, go to TEST
↳ Test Event
↳ Event name — SanyuktaDemoEvent

All Same
Save — upload

6) In lambda, go to monitors
↳ view logs in cloudWatch

≠ To customize image

7) ① In lambda, go to TEST
↳ Test Event
• Event name — SanyuktaDemoEvent
• code



→ Code

↳ index.js

```
console.log
("An img has been
added");
```

Save

Deploy | Test

Step 4: Create a trigger to invoke creation of logs in CloudWatch when an image is uploaded in the specified bucket. Test the code & view the logs in CloudWatch.

Clean up Resources –

1) delete lambda function
2) delete IAM policy
   IAM Roles.
3) delete S3 Bucket