JENKINS	➢ Install on Desktop/Local Host:8080: ➢ Install on AWS Linux2 [public ip:8080]: **sudo yum update -y** **sudo wget link from jenkins.io downloads page for linux** **sudo yum upgrade -y** **sudo amazon-linux-extras install java-openjdk11 -y** **sudo yum install jenkins -y** **sudo systemctl enable jenkins** **sudo systemctl start jenkins** **sudo systemctl status jenkins** check in browser with publicIP:8080 Copy password link and cat passwordlink and login into Jenkins dashboard ➢ Install on AWS ubuntu 20.04.2LTS [public ip:8080]: **sudo apt-get update -y** **wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -** **sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'** **sudo apt update** **sudo apt install jenkins** **sudo systemctl start jenkins** **sudo systemctl status jenkins** **sudo systemctl status jenkins** **sudo ufw allow 8080** **sudo ufw allow OpenSSH** **sudo ufw enable** **sudo ufw status** check in browser with publicIP:8080 Copy password link and cat passwordlink and login into Jenkins dashboard **sudo cat /var/lib/jenkins/secrets/initialAdminPassword** {Copy the 32-character alphanumeric password from the terminal and paste it into the **Administrator password** field, then click **Continue}** ==> click the **Install suggested plugins** option, ==> Enter the name and password for your user: **Instance Configuration ==> Save and Finish ==> Jenkins is Ready**

DOCKER	➢ Install on Desktop/Local Host: ➢ Install on AWS Linux2: **sudo yum update -y** **sudo amazon-linux-extras install docker.io -y** **sudo systemctl enable docker** **sudo systemctl start docker** **sudo systemctl status docker** **sudo usermod -a -G docker ec2-user** **docker info** **sudo vi Dockerfile** **docker build -t naren-1 .** (i.e. img name here . means in same folder) **==> docker image ls** **docker login --username Narian318 ==> password:** **docker tag naren-1 "dockerhub Repo name:version tag"** **docker push "dockerhub Repo name:version tag" ==> docker image ls** **docker run -dp 80:80 "new img name"** copy & paste EC2 publicIP or public DNS link in web browser to see website **docker ps ==> docker stop "container id"** ➢ Install on AWS ubuntu 20.04.2LTS: **sudo apt update** **sudo apt install apt-transport-https ca-certificates curl software-properties-common** **curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -** **sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"** **apt-cache policy docker-ce** **sudo apt install docker-ce** **sudo systemctl status docker** **sudo usermod -aG docker ${USER} means ec2-user** **sudo usermod -aG docker "username"** **docker info** **docker run hello-world** **docker pull ubuntu** **docker images** **docker run -it ubuntu** **apt update ==> apt install nodejs ==> node -v**

	docker ps -a or docker ps or docker ps -1 **docker start 1c08a7a0d0e4(container id)** **docker stop quizzical_mcnulty** **docker commit -m "What you did to the image" -a "Author Name" container_id repository/new_image_name** **docker commit -m "added Node.js" -a "sammy" d9b100f2f636 sammy/ubuntu-nodejs** **docker login -u docker-registry-username** **docker tag sammy/ubuntu-nodejs dockerhub username/ubuntu-nodejs** **docker push dockerhub username/docker-image-name** **docker push sammy/ubuntu-nodejs**

ANSIBLE	**As per AOS Note videos:** ➢ Install on AWS Linux2: **sudo yum update -y** **sudo amazon-linux-extras install ansible2 -y** **ansible --version** **sudo vi inventory [add servers pvt ip's] ==> :wq! (save)** **sudo vi nn-playbook.yml [yaml code of ansible playbook for tasks to perform into target servers]** **ansible all --key-file ~/.ssh/id_rsa -i inventory -m ping -u ec2-user [to ping servers from ansible master]** **sudo vi ansible.cfg** **[defaults]** **remote_user = ec2-user** **inventory = inventory** **private_key_file = ~/.ssh/id_rsa ==> :wq! (save)** **ansible-playbook nn-playbook.yml** [to run playbook] Project: Install **Ansible** on **Amazon Linux-2** EC2 [with its default Inventory file] **(user as ansadmin)** on Ansible Master: `sudo su -` [root@ansible-master ~]# `1. yum update -y` `2. useradd ansadmin` `3. passwd ansadmin # give your own password for "ansadmin" user` `4. vi /etc/sudoers # adding ansadmin ALL=(ALL) NOPASSWD:ALL` `5. vi /etc/ssh/sshd_config # No Password Authentication yes` `6. systemctl restart sshd` Note: Repeat the above steps in all Node servers also on Ansible Master: `1. su - ansadmin` `2. sudo yum update -y` `3. ssh-keygen` `4. cd .ssh/` `5. ls -ltr` `6. ssh-copy-id ansadmin@35.174.105.47 (# Pvt IP of Node) # run this same for all 3 nodes` `7. sudo amazon-linux-extras install ansible2 -y` `8. ansible --version` `9. cd /etc/ansible` `10. ls -l` `11. sudo chown -R ansadmin:ansadmin *` `12. vi hosts # adding the all Node servers pvt IPs in [servers] group` `13. vi ansible.cfg # uncomment inv line, add interpreter_python = /usr/bin/python under defaults` `14. ansible all -m ping # to ping all Nodes status` `15. sudo touch naren-file1` `16. ansible all -i hosts -m copy -a "src=naren-file1 dest=/home/ansadmin/" # to copy naren-file1 to all Nodes from master thru adhoc command` `17. sudo vi naren-pb-git.yml # create playbook to install git on Nodes` `18. ansible-playbook naren-pb-git.yml --syntax-check` `19. ansible-playbook naren-pb-git.yml`

	(Git & Ansible) Install on Amazon EC2 Linux2 (user as ec2-user) 1. sudo yum update -y & sudo yum install git -y 2. sudo -i 3. ssh-keygen 4. cd .ssh/ 5. ls -ltr 6. ssh-copy-id root@35.174.105.47 (# Pvt IP of Node) # run this same for all 2 nodes 7. sudo amazon-linux-extras install ansible2 -y 8. ansible --version 9. cd /etc/ansible 10. ls -l 11. sudo chown -R ec2-user:ec2-user * 12. vi hosts # adding the all Node servers pvt IPs in [servers] group 13. vi ansible.cfg # uncomment inv line, add interpreter_python = /usr/bin/python under defaults 14. ansible all -m ping # to ping all Nodes status cd /home ==> git clone github-repo link

MAVEN	Project: Install **Maven** on **Amazon Linux-2** EC2 1. cd /opt 2. wget https://dlcdn.apache.org/maven/maven-3/3.9.3/binaries/apache-maven-3.9.3-bin.tar.gz 3. ls 4. tar -xvzf apache-maven-3.9.3-bin.tar.gz 5. ls 6. mv apache-maven-3.9.3 maven3 7. rm -rf apache-maven-3.9.3-bin.tar.gz 8. export PATH=$PATH:/opt/maven3/bin/

SONARQUBE	Project: Install **Sonarqube** on **Amazon Linux-2** EC2 [with sonar user] 1. Launch EC2 with 22, 80, 9000 ports open from anywhere access 2. sudo yum update -y 3. sudo su - 4. cd /opt ==> yum install wget unzip -y ==> amazon-linux-extras install java-openjdk11 -y 5. wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-8.9.10.61524.zip 6. unzip sonarqube-8.9.10.61524.zip 7. rm -rf sonarqube-8.9.10.61524.zip 8. useradd sonar ==> visudo { adding at root as: sonar ALL=(ALL) NOPASSWD:ALL } 9. chown -R sonar:sonar /opt/sonarqube-8.9.10.61524 10. chmod -R 775 /opt/sonarqube-8.9.10.61524 11. su - sonar 12. cd opt/ 13. cd sonarqube-8.9.10.61524/ 14. cd bin 15. cd linux-x86-64/ 16. ./sonar.sh start [# check the Application with EC2 Public IP:9000 in browser]

TOMCAT	Project: Install **Tomcat** on **Amazon Linux-2** EC2 on Tomcat server (8090 port) 1. yum update -y 2. yum install git wget -y 3. yum install java-1.8* 4. cd /opt 5. wget https://dlcdn.apache.org/tomcat/tomcat-8/v8.5.90/bin/apache-tomcat-8.5.90.tar.gz 6. tar -xvzf apache-tomcat-8.5.90.tar.gz 7. chmod +x -R apache-tomcat-8.5.90 8. cd apache-tomcat-8.5.90/ 9. cd bin 10. pwd 11. chmod +x startup.sh 12. chmod +x shutdown.sh 13. ln -s /opt/apache-tomcat-8.5.90/bin/startup.sh /usr/local/bin/tomcatup 14. ln -s /opt/apache-tomcat-8.5.90/bin/shutdown.sh /usr/local/bin/tomcatdown 15. tomcatup 16. cd 17. tomcatdown 18. tomcatup 19. cd /opt 20. cd apache-tomcat-8.5.90/ 21. cd conf/ 22. vim server.xml 23. tomcatdown 24. tomcatup 25. find / -name | grep context.xml 26. find / -name context.xml 27. vim /opt/apache-tomcat-8.5.90/webapps/host-manager/META-INF/context.xml 28. vim /opt/apache-tomcat-8.5.90/webapps/manager/META-INF/context.xml 29. tomcatdown 30. tomcatup 31. vim tomcat-users.xml 32. cd 33. hostname nn-tomcat-server 34. exec bash 35. cd /opt 36. apache-tomcat-8.5.90/ 37. cd webapps/ 38. ls -ltr 39. date 40. cd webapp 41. cat index.jsp

{GRAFANA+ LOKI+ PROMTAIL}	**Grafana-Loki-Promtail** **sudo apt-get update** **sudo apt install nginx** **sudo apt-get install -y apt-transport-https** **sudo apt-get install -y software-properties-common wget** **sudo wget -q -O /usr/share/keyrings/grafana.key https://apt.grafana.com/gpg.key** **echo "deb [signed-by=/usr/share/keyrings/grafana.key] https://apt.grafana.com stable main" | sudo tee -a /etc/apt/sources.list.d/grafana.list** **# Update the list of available packages** **sudo apt-get update** **# Install the latest OSS release:** **sudo apt-get install grafana** **# Start grafana-server:** **sudo systemctl start grafana-server** **You can now access Grafana by navigating to http://your-ec2-instance-ip:3000** **The default username and password are admin/admin** **sudo apt-get update** **sudo apt-get install docker.io** **sudo usermod -aG docker $USER** **sudo reboot**

	2 sudo mkdir grafana_configs 3 cd grafana_configs 4 sudo wget https://raw.githubusercontent.com/grafana/loki/v2.8.0/cmd/loki/loki-local-config.yaml -O loki-config.yaml 5 sudo wget https://raw.githubusercontent.com/grafana/loki/v2.8.0/clients/cmd/promtail/promtail-docker-config.yaml -O promtail-config.yaml 6 ls 7 docker run -d --name loki -v $(pwd):/mnt/config -p 3100:3100 grafana/loki:2.8.0 --config.file=/mnt/config/loki-config.yaml 8 docker run -d --name promtail -v $(pwd):/mnt/config -v /var/log:/var/log --link loki grafana/promtail:2.8.0 --config.file=/mnt/config/promtail-config.yaml 9 cd .. 10 sudo systemctl restart grafana-server 11 cd grafana_configs 12 vim promtail-config.yaml 13 chmod 777 promtail-config.yaml 14 sudo chmod 777 promtail-config.yaml 15 vim promtail-config.yaml 16 docker ps 17 sudo docker restart promtail-container-id 18 sudo docker restart 5800f350a167 19 sudo apt install stress 20 history

{Magento on EKS}	AWS EKS Kubectl (Magento Install) #Create Cluster: curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" \| tar xz -C /tmp sudo mv /tmp/eksctl /usr/local/bin eksctl create cluster --name magento-cluster aws eks update-kubeconfig --region ap-south-1 --name magento-cluster #Install Helm: sudo yum install openssl -y curl https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 > get_helm.sh chmod 700 get_helm.sh ./get_helm.sh #Setup EBS CSI addon (OIDC IAM) for EKS: oidc_id=$(aws eks describe-cluster --name magento-cluster --query "cluster.identity.oidc.issuer" --output text | cut -d '/' -f 5) aws iam list-open-id-connect-providers | grep $oidc_id \| cut -d "/" -f4 eksctl utils associate-iam-oidc-provider --cluster magento-cluster --approve #IAM Role for eksctl: eksctl create iamserviceaccount \ --name ebs-csi-controller-sa \ --namespace kube-system \ --cluster magento-cluster \ --attach-policy-arn arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy \ --approve \ --role-only \ --role-name AmazonEKS_EBS_CSI_DriverRoleMagento #Adding EBS CSI to EKS: eksctl create addon --name aws-ebs-csi-driver --cluster magento-cluster --service-account-role-arn arn:aws:iam::167613117387:role/AmazonEKS_EBS_CSI_DriverRoleMagento --force #Install Magento thru Helm: helm repo add bitnami https://charts.bitnami.com/bitnami helm install my-release bitnami/magento #Set the Hostname: export APP_HOST=$(kubectl get svc --namespace default my-release-magento --template "{{ range (index .status.loadBalancer.ingress 0) }}{{ . }}{{ end }}") export APP_PASSWORD=$(kubectl get secret --namespace default my-release-magento -o jsonpath="{.data.magento-password}" \| base64 -d) export DATABASE_ROOT_PASSWORD=$(kubectl get secret --namespace default my-release-mariadb -o jsonpath="{.data.mariadb-root-password}" | base64 -d) export APP_DATABASE_PASSWORD=$(kubectl get secret --namespace default my-release-mariadb -o jsonpath="{.data.mariadb-password}" | base64 -d) helm upgrade --namespace default my-release bitnami/magento \

	--set magentoHost=$APP_HOST,magentoPassword=$APP_PASSWORD,mariadb.auth.rootPassword=$DATABASE_ROOT_PASSWORD,mariadb.auth.password=$APP_DATABASE_PASSWORD echo Password : $(kubectl get secret --namespace default my-release-magento -o jsonpath="{.data.magento-password}" | base64 -d) #delete Total Cluster eksctl delete cluster magento-cluster

{EKS+ISTIO+ KIALI+JAIGER}	curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl" curl -LO "https://dl.k8s.io/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256" echo "$(cat kubectl.sha256) kubectl" \| sha256sum --check sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp sudo mv /tmp/eksctl /usr/bin eksctl version eksctl create cluster --name=nn-eks-aug --region=us-west-1 --zones=us-west-1c,us-west-1a --without-nodegroup eksctl utils associate-iam-oidc-provider --region us-west-1 --cluster nn-eks-aug --approve eksctl create nodegroup --cluster=nn-eks-aug --region=us-west-1 --name=eksdemo-ng-public --node-type=t2.medium --nodes=2 --nodes-min=2 --nodes-max=4 --node-volume-size=10 --ssh-access --ssh-public-key=nn-uswest1-key --managed --asg-access --external-dns-access --full-ecr-access --appmesh-access --alb-ingress-access kubectl get pods -n kube-system curl -L https://istio.io/downloadIstio | ISTIO_VERSION=1.18.1 TARGET_ARCH=x86_64 sh - cd istio-1.18.1 ll cd samples ll cd bin cd .. cd bin ll cd .. export PATH=$PWD/bin:$PATH istioctl install --set profile=demo -y kubectl get pods -n default kubectl get namespace kubectl get pods -n istio-system kubectl apply -f https://raw.githubusercontent.com/istio/istio/release-1.18/samples/bookinfo/platform/kube/bookinfo.yaml kubectl get pods -n default kubectl exec "$(kubectl get pod -l app=ratings -o jsonpath='{.items[0].metadata.name}')" -c ratings -- curl -sS productpage:9080/productpage \| grep -o "<title>.*</title>" kubectl get services kubectl label namespace default istio-injection=enabled istioctl analyze kubectl get pods kubectl delete pod details-v1-5ffd6b64f7-dnfpf kubectl delete pod productpage-v1-8b588bf6d-5s46z ratings-v1-5f9699cfdf-wjlqk reviews-v1-569db879f5-wrq8r reviews-v2-65c4dc6fdc-t4wd4 kubectl delete pod reviews-v3-c9c4fb987-zz6ts kubectl get pods cd samples/bookinfo/networking/ kubectl apply -f bookinfo-gateway.yaml kubectl get gateway kubectl get svc istio-ingressgateway -n istio-system export INGRESS_HOST=$(kubectl -n istio-system get service istio-ingressgateway -o jsonpath='{.status.loadBalancer.ingress[0].ip}') export INGRESS_PORT=$(kubectl -n istio-system get service istio-ingressgateway -o jsonpath='{.spec.ports[?(@.name=="http2")].port}') export SECURE_INGRESS_PORT=$(kubectl -n istio-system get service istio-ingressgateway -o jsonpath='{.spec.ports[?(@.name=="https")].port}') echo $SECURE_INGRESS_PORT export INGRESS_HOST=ac62cb469da37487ebccb53635c96ef2-1463963366.us-west-1.elb.amazonaws.com

	export GATEWAY_URL=$INGRESS_HOST:$INGRESS_PORT **echo $GATEWAY_URL** **echo "http://$GATEWAY_URL/productpage"** **cd ..** **cd ..** **cd addons** **kubectl apply -f .** **kubectl get pods -n istio-system** **kubectl port-forward --address 0.0.0.0 svc/kiali 9008:20001 -n istio-system** **kubectl port-forward --address 0.0.0.0 svc/tracing 8008:80 -n istio-system** **history** **eksctl delete nodegroup --cluster=nn-eks-aug --region=us-west-1 --name=eksdemo-ng-public** **eksctl delete cluster --name=nn-eks-aug --region=us-west-1**

Node.js	Launch EC2 {ubuntu 22.04} with 22, 80, 443 ports open from anywhere access **Install NodeJs and npm using nvm:** **sudo su -** **curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.34.0/install.sh | bash** **. ~/.nvm/nvm.sh (Activate nvm)** **nvm install node** **node -v** **npm -v** Install Git and Clone Git Repo from Github **sudo apt-get update -y** **sudo apt-get install git -y** **git --version** **git clone https://github.com/yeshwanthlm/nodejs-on-ec2.git** **cd nodejs-on-ec2** **npm install** **npm start** (==> check the Application with EC2 Public IP in browser)

Django (Py)	**Install Django using python:** Launch EC2 {ubuntu 22.04} with 22, 80, 443, 3000 ports open from anywhere access **sudo apt-get update** **git clone https://github.com/yeshwanthlm/django-on-ec2.git** **cd django-on-ec2** **sudo apt install python3-pip -y** **pip install django** [Download django usig pip] **python3 manage.py makemigrations** **python3 manage.py migrate** [all database migrations] **python3 manage.py createsuperuser** **python3 manage.py runserver** **python3 manage.py runserver 0.0.0.0:3000** ==> check the Application with EC2 Public IP:3000 in browser

Kubectl + eksctl	**AWS EKS cluster setup through (kubectl + eksctl) :** Launch EC2 instance ==> Install kubectl ==> Install eksctl ==> Create IAM Role with admin access ==> attach this IAM role to EC2 Create Cluster & Nodes ==> check Nodes & Pods ==> delete cluster Launch EC2 {Amazon Linux2} with 22, 80, 443, ports open from anywhere access # Steps:- # For kubectl Installation **sudo su** **yum update -y** **curl -o kubectl https://amazon-eks.s3.us-west-2.amazonaws.com/1.19.6/2021-01-05/bin/linux/amd64/kubectl** **chmod +x ./kubectl** **mv ./kubectl /usr/local/bin** **kubectl version --short --client** # For eksctl Installation **curl --silent --location**

	"https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp sudo mv /tmp/eksctl /usr/local/bin eksctl version # For Cluster & Nodes Creation eksctl create cluster --name naren-eks --region us-east-1 --node-type t2.small --zones us-east-1a,us-east-1b kubectl get nodes ==> kubectl get pods eksctl delete cluster naren-eks --region us-east-1

MongoDB	Launch EC2 {Amazon Linux 2023} with 22, 80, 443 ports open from anywhere access **Install MongoDB on Amazon Linux 2023 (Fedora):** sudo dnf update sudo dnf install docker sudo systemctl start docker sudo systemctl enable docker sudo docker pull mongo mkdir ~/mongodb_data sudo docker run -d -p 27017:27017 -v ~/mongodb_data:/data/db --name mongodb mongo sudo docker ps docker exec -it mongodb mongosh db.runCommand({ hello: 1 })

MySQL	**MySQL on Amazon Linux 2023 (Fedora):** sudo wget https://dev.mysql.com/get/mysql80-community-release-el9-1.noarch.rpm sudo ls -lrt sudo dnf install mysql80-community-release-el9-1.noarch.rpm dnf repolist enabled | grep "mysql.*-community.*" sudo dnf install mysql-community-server sudo systemctl start mysqld sudo mysql -V sudo mysql_secure_installation sudo mysql -uroot -p sudo vi /etc/my.cnf ==> skip-grant-tables sudo systemctl restart mysqld sudo mysql_secure_installation mysql -uroot -p CREATE schema my_database; show databases; exit **MySQL on Ubuntu 22.04:** sudo apt update sudo apt install mysql-server systemctl start mysql systemctl enable mysql mysql -u root -p ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'new_password'; sudo mysql_secure_installation mysql -uroot -p

Java Application	Launch EC2 {Amazon Linux 2023} with 22, 80, 443 ports open from anywhere access **Java Application on Amazon Linux 2023(Fedora):** **sudo dnf update -y** **sudo dnf install java-17-amazon-corretto-devel** **java -version** **sudo vi HelloWorld.java** **public class HelloWorld {** **public static void main(String[] args) {** **System.out.println("Hello, World!");** **}** **} ==> :wq!** **javac HelloWorld.java** **java HelloWorld** **Java Application on Ubuntu 22.04:** **sudo apt update** **sudo apt install default-jdk** **java -version** **sudo vi HelloWorld.java** **public class HelloWorld {** **public static void main(String[] args) {** **System.out.println("Hello, World!");** **}** **} ==> :wq!** **javac HelloWorld.java** **java HelloWorld**

Node.Js	Launch EC2 {Amazon Linux 2023} with 22, 80, 443 ports open from anywhere access **Node.Js on Amazon Linux 2023 (Fedora):** **curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.0/install.sh | bash** **nvm --version** **nvm install node** **nvm install 16.0.0** **nvm use node** **nvm use 16.0.0** **node -v** **sudo dnf install nginx -y** **sudo systemctl start nginx** **sudo vi /etc/nginx/conf.d/sameple.com.conf** **server {** **listen 80;** **server_name nodejs.naren-cloudsolutions.click;** **location / {** **proxy_pass http://127.0.0.1:3000;** **proxy_set_header Host $host;** **proxy_set_header X-Real-IP $remote_addr;** **proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;** **proxy_set_header X-Forwarded-Proto $scheme;** **}** **}** **SSL setup for above domain:** **sudo firewall-cmd –add-service=https –permanent** **sudo firewall-cmd –reload** **sudo python3 -m venv /opt/certbot/** **sudo /opt/certbot/bin/pip install --upgrade pip** **sudo /opt/certbot/bin/pip install certbot certbot-nginx** **sudo ln -s /opt/certbot/bin/certbot /usr/bin/certbot** **sudo nano /etc/nginx/nginx.conf** **server {** **listen 80;** **server_name nodejs.naren-cloudsolutions.click;** **return 301 https://$host$request_uri;** **}** **sudo certbot --nginx** **sudo certbot renew [for automatic renewal of https validity]** **(check in browser for https lock symbol)**

Apache	Launch EC2 {Amazon Linux 2023} with 22, 80, 443 ports open from anywhere access **Apache on Amazon Linux 2023 (Fedora):** **sudo dnf update -y** **sudo dnf install httpd mod_ssl php php-mysqlnd -y** **sudo systemctl start httpd** **sudo systemctl status httpd** **sudo systemctl enable httpd** **sudo vi /var/www/html/info.php** **<?php** **phpinfo();** **?>** **sudo dnf install python3 augeas-libs** **sudo python3 -m venv /opt/certbot/** **sudo /opt/certbot/bin/pip install --upgrade pip** **sudo /opt/certbot/bin/pip install certbot certbot-apache** **sudo ln -s /opt/certbot/bin/certbot /usr/bin/certbot** **sudo certbot --apache**