classmate
Date 09/01/2024
Page

LAMP stacks to deploy static/dynamic application

Open terminal - ctrl + Alt + T
username@hostname
→ hostname:
↳ Gives the hostname
→ ip a
Different network connected to system
enp2s0 - network interface card name (LAN)
ip address - 172.16.51.43/24
lo - loop back : call same system
docker
→ clear - clear screen
→ cat /etc/os-release : linux (Ubuntu) version
→ pwd : present working directory
→ cd / → root

master = 07
windows_slave = 65

- All executable commands are present in bin folder
- High level administrator commands ~~or~~ executables are present in sbin.
- Linux kernel & boot-loader - boot
- All shared modules/libraries across multiple users - usr
  pip3 show awscli
  → Version, name, path.
  → whereis python3 / which python3
- Softwares which are 3-party and have their own configurations - opt
- All users details - home
- External disks, floppy details - dev
- External mounting - mnt
- root user home folder - root
- Temporary variables / logs - var
- Temporary session data - tmp
- Configuration system details - etc

sudo vi /etc/hostname
i - enter
esc + :wq → save
esc + :q! → without save quit.
esc + d → delete
esc + yy → paste

sudo vi /etc/hosts
Lists IP addresses and all details

cd ~ / cd / cd home (full path) → To go to home

apt → package manager - install / update / detail any app
→ sudo apt-get update -y - updating all servers
→ sudo nano /etc/apt/sources.list
Link to all ubuntu sources
in ubuntu it will be ~~def~~ deb file

→ sudo apt-get install apache2 -y
→ sudo systemctl status apache2 → checking status
check status restart service enable service
start service stop service disable service

→ sudo service apache2 status → checking status
→ Send request to server
http://172.16.51.23:80
→ sudo nano /etc/apache2/sites-available/000-default.conf
DocumentRoot → default folder for apache
/var/www/html
→ sudo mkdir myweb
html5 up or FreeHTML5.com
→ sudo cp Downloads/html5up.zip . → current location
→ sudo mkdir code
→ sudo cp html5up.zip code/.

→ cd code
→ sudo unzip html5up.zip
→ sudo rm -rf html5up.zip
→ sudo cp * /var/www/html/myweb1.
→ sudo nano /etc/apache2/sites-available/000-default.conf
Change DocumentRoot to myweb
→ Restart apache2

PHP server:-
webserver graphical interface
→ sudo apt-get install php libapache2-mod-php -y
↳ Interface for php to render apache2
↳ Interface for apache2 to render php

Github - source code/version code management
git - tool to perform operations on github

→ git clone URL
→ sudo nano /etc/apache2/mods-enabled/dir.conf

10/01/2023 MEAN stack
- MongoDB, Angular & node.js
MERN stack
- MongoDB, React, node.js

→ sudo apt-get install mysql-server mysql-client php-mysql -y
↓
Library that PHP needs to perform CRUD operations on mysql server

→ clone git repository

→ Set up front-end as following same steps as bef
→ Set up mysql database.
→ Connection string – Connecting the application with the database (host)
→ sudo mysql -u root -p
Password not needed.
→ create database ecommerce;
→ Create user 'nidhi'@'localhost' IDENTIFIED BY 'passwo
Create user 'nidhi'@'172.16.51.23' identified by 'passwo
create user 'nidhi'@'%' indentified by 'password';
'MSis@123
→ Grant All privileges on *.* To 'msis'@'localhost';
everything/all db

→ Grant all privileges on ecommerce.* To 'msis'@'localhs

→ Go to /var/www/html/ecommerce
→ Go to mysql
→ use ecommerce;
→ source onlineshop.sql; → copies tables/data from onlines hoppe.sql to ecommerce database.

→ For connection string we need to provide host, database username and password details
→ In the ecommerce folder check for db.php or config.php file.
→ sudo nano db.php
→ edit those 4 details in the file.

Github and git :

- .gitignore - If we want to ignore few files to copy or pull out of networks
- Tags - organize file based on tags
- Our own repo - origin
- If we fork a repo to ours - upstream

step1: Create a repo
step2: Go to code → copy the URL
step3: ls -a → To show hidden files
step4: sudo mkdir mybankapp
step5: cd mybankapp
step6: sudo git init → local repo (working directory)
step7: cd .git → All objects are stored as BLOBs
step8: cd ..
step9: sudo git remote add/set origin "URL"
can be used

Linking git and local repo
step10: sudo git pull origin main
step11: sudo git branch (branch name)
step12: sudo git branch -r (branch in remote)
step13: sudo git branch -a (both branches name)
step14: sudo git checkout main
Create a new branch or shift from one branch to oth
step15: sudo git branch -M main, master
step16: sudo touch index.html
step17: sudo cat >> index.html
step18: sudo ls -la
step19: Change ownership/permissions
sudo chown -R ms1s:ms1s *
step20: cat >> index.html → Write to file
step21: cat index.html → Read
step22: sudo git status

23) sudo git add index.html
24) sudo git status
25) sudo git commit index.html "-m "First commit"
26) If we get unknown authority error
sudo git config --global user.email "email id"
sudo git config -global user.name "user name"
27) Push to repo
sudo git push origin main
Copy for auth token for password.
28) sudo git log
29) sudo git cat-file, "commited" -p
Tree structure under .git hidden folder

sudo chown -R msis:msis /home/sois/mybanksapp
→ permission for all files
sudo git add -A (multiple files)
sudo git commit -a -m "Added files"

Create a new branch:-
→ sudo git branch userreg
→ sudo git branch
→ sudo git checkout userreg

Merging a branch:-

→ Go to branch to which you want to merge
→ sudo git merge userreg
→ sudo git rebase userreg

1) stage diff
2) Revert
3) Rebase

Continuous Integration:

Jenkins:-
jar, war, ear → package

Standalone server - Wherever jenkins is installed, the same server is used for build.

Source → How are we building it → what are tools used to build → Result → Post build

Build automatically - webhooks creation

Build steps.
Goals - clean compile validate test install package
Code review metrics
goals - pmd:pmd checkstyle:checkstyle findbugs:findbugs

ssh jenkins@172.16.51.121
50is@123

cd /var/lib/jenkins
cd workspace
cd Java_deploy
cd target
java -jar jarname

23/01/2024 Deploy war, ear to tomcat server:-

Jsp-deploy - project
Build steps - install package
Deploy war/ear (plugin) Deploy to container
**/*.war → path
name of the application - context path

Container : Select remote server (Tomcat or Glassfish)
- Tomcat 8.x remote
Credentials - Jenkins
Tomcat URL

Stop jenkins
sudo systemctl stop jenkins

Install tomcat (tomcat9)
sudo apt-get install tomcat9 -admin

sudo vi /etc/tomcat9/tomcat-users.xml
<user username="jenkins" password="sois@123"
roles="manager-script"/>
User with this role only can deploy an application from 3rd party application or from remote server

Manage Jenkins
Manage credentials
→ System
→ Global credentials
→ There give username & password

Node.js application

2) Clone git repository (npm - node package manager)
sudo nano server.js
sudo apt-get install nodejs npm -y
node --version
npm --version
sudo npm install - Reads package.json & install dependencies
sudo node server.js (Running in front end) (foreground)
sudo npm -g install pm2 (to run in background) (boots)
↳ to access the package anywhere

sudo pm2 start server.js
sudo pm2 stop/restart server.js

Set up a slave:-

1) Manage Jenkins
2) Security
3) Agents - Fixed or random = 9000
   port number
4) save
5) Manage Jenkins
6) Nodes
7) New node
8) Ubuntu-ip-server
9) Permanent agent - create
10) No. of executors = 1
11) Remote root directory
    /home/msis/jenkins
12) Save

If curl command has localhost

1) Manage jenkins
2) Configure system
3) Change Jenkins URL to ip
4) Run both curl & Java commands in ~~master~~ slave

Restrict where this project can be run. - select slave

~~Pedit~~

Deploy PHP application:-

Publish over SSH - Plugin for PHP file

Sudo nano /etc/mysql/mysql.conf.d

bind address to 0.0.0.0

CI asp.net application deploy on windows:-

→ MSBuild
→ Internet Integrated Server (IIS)
→ Solution file (.sln)

→ package manager - nuget

Create a windows node:-

root directory
C:/Users/MSIS/Folder-name - remote working directory

Plugins:-

Build a visual - MSBuild
Testing - MSTest

Adding tools repository:-

MSBuild → add the tool
Add this path in slave machine configuration
Go to slave node
configure
Tool Locations
Git → add git path of slave
MSBuild → add path

Creating project:-

→ Restrict where to run
→ Git repo
→ Build steps:- execute windows batch command
nuget.exe restore AsCalsMaster.sln
asp.net/c# package manager

MSBuild – Build a visual studio project

MSBuild – GS-Server

file = Asp.sln

/t ; clean; build; package /p:PackageFileName = "C:/Users/Jenkins/workspace/Calculator/aspcalapp.zip"

Execute windows batch command

– Application

Run test cases

Windows package installer

→ chocolatey / choco

→ winget

Docker:-

An open source computer program used for containing

Step 1: Install docker
sudo apt-get install docker.io
Docker demon is installed.
RunC, containerD & CLI installed,
low level high level
runtime software
Image is the template (read-only), which has info on container

Step 2: sudo docker run hello-world

Step 3:- sudo docker pull Centos:latest
If we don't give username, then it pulls from public docker hub community

Step 4:- sudo docker image ls
→ To list all images

Step 5:- sudo docker rmi $(sudo docker image ls -a -q / sudo docker ps -aq -q)
→ Removes all images

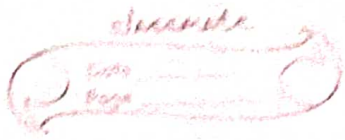
Step 6:- sudo docker pull centos:latest

Step 7:- Run or start a container
sudo docker run --name centos -it -d centos:latest (docker image)
interact with sudo terminal

-d = detached mode (in background)
Creates 64bit container ID

Step 8:- sudo docker ps
→ Lists the processes that are currently running

Step 9:- sudo docker ps -a
→ Lists all processes



Step10: sudo docker exec -it centos bash

cat /etc/os-release → to check OS details

yum update
↳ yet another update manager

Step11:- sudo docker rmf centos

Step11: sudo docker stop centos - stop container
Step12: sudo docker rm -f centos - remove container

sudo docker pull ubuntu:latest
sudo docker run (name ubuntu) -it ubuntu:latest
→ Directly inside container

→ apt-get update
→ cp to bookalbum in in git a copy for sed command
→ cat /etc/apt/source-list
→ apt-get update
→ apt-get install apache2
→ service apache2 status
→ service apache start
→ cafe static website git
→ apt-get install git
→ cd cafe-static-website
→ cp -rf * /var/www/html.
→ exit from container (exit)
→ capture current status container
sudo docker commit ubuntu mycafewebsite:v1
→ sudo docker image ls (host)
→ sudo docker run --name mycafeApp -it -d -p 4000:80 (host port to container port) mycafewebsite:v1

→ sudo docker ps
→ Restart the apache2 server
→ sudo docker exec -it mycafeapp bash
→ service apache2 start
→ exit

Push it to Hub :-

→ sudo docker image ls
→ sudo docker tag IMAGEID HUBNAME / mycafewebsite
copy the id of the image

→ sudo docker login
→ sudo docker push sreedocker123/mycafewebsite:V1

Docker file :-

→ sudo git clone website
→ cd cafe-website
→ sudo nano dockerfile
- Edit the file
- Dockerfile contains english statements with all steps
- Front-end = non-interactive
- ADD (cp) ./var/www/html/
- Expose 80
- EntryPOINT (cmd) apachectl -D FOREGROUND

Build an image :-

-f name of dockerfile
any name

sudo docker build -t sreedocker123/mycafewebsite:v2 .
sudo docker run -it -d -p 4001:80 sreedocker123/mycafewebsite:V2

classmate
Date 31/01/2024
Page

Containerization:

→ clone the git basic-php-website (sudo rm -rf for rpm if exists)
→ sudo nano Dockerfile
→ sudo docker run --name booksa
→ sudo docker build -t bookalbum:v1 .
→ sudo docker run --name bookalbum -it -d -p 4000:80 bookalbum:v1
→ sudo docker rm -f bookalbum → remove container if
→ Go to browser & check

NodeJS docker Demo :-

→ Git nodeJS docker repo
→ cd nodejsDockerDemo
→ ls
→ sudo nano package.json → check for dependency
→ sudo nano Dockerfile → change the port
→ sudo docker build -t nodeapp:v1 .

Create a CI/CD pipeline in Jenkins:-

→ Create a freestyle project
→ Restrict to run on slave (which should have docker imstalled) → Git link
→ Build step
  • Docker (cloud Bees) → To push to hub
  • Dockerhub name - sreedocker123/cafedockerapp
  • Tag - v1
  • Add credentials of dockerhub
→ sudo dake run --name ms2s -it -d -p 4007580 nedhsrao/cafedockerdemo:v1

sudo groupadd docker
sudo usermod -aG docker $USER

Date 05/02/2024
Page

Data persistence :-

Data persistance even after the deletion/removal of container/docker

→ git - docker-volume-nodejs-app
→ clone to your system
→ cd docker-volume-nodejs-app
→ ls
→ sudo nano Dockerfile
→ Build an image :- sudo docker build -t bankapp:v1 .
→ Run a container :- sudo docker run --name bankapp -it -d -p 3000:80 bankapp:v1
→ sudo docker ps → To check the status
→ 172.16.151.43:80 3000
→ Data is generated. Go & see inside the container
→ Go inside container: sudo docker exec -it bankapp bash
→ cd feedback
→ ls → All data stored under /app/feedback

Case 2:- When the container is accidently deleted.

→ sudo docker rm -f bankapp → delete the bankapp
→ Try to run the same container :- sudo docker run --name bankapp -it -d -p 3000:80 bankapp:v1
→ Copy the /app/feedback or using docker volume for data persistence
→ Create a folder in the host : sudo mkdir mybankvolume
→ copy the complete path where docker volume created
→ sudo docker run --name bankapp -it -d -p 3000:80 -v /home/sais/docker-volume-nodejs-app/mybankvolume:/app/feedback bankapp:v1
→ Go to browser : 172.16.51.43:3000
→ Data is stored in mybankvolume

→ cd mybankvolume
→ touch demo.txt
→ sudo chown -R msis:msis demo.txt
→ cat > demo.txt (bind modes)

Docker managed volumes or Named volumes :-

→ sudo docker volume create dockerbankvolume
→ sudo docker run --name bankapp -it -d -p 3001:80
-v dockerbankvolume:/app/feedback bankapp:v1
→ Get details of docker managed volumes :- sudo docker
inspect dockerbankvolume
→ change/switch to root user :- see root
→ cd /var/lib/docker/volumes/dockerbankvolume/

Anonymous volumes :-
→ We will not create a docker volume
→ But bind a docker volume while running the
container
→ This will automatically creates the dockervolume
→ cd /var/lib/docker/volume/NAME_OF_DOCKER_VOLUME
→ cd _data
→ ls

Docker compose :-

→ Run multiple containers together and bind it togethe

→ git clone company-addressbooks
→ cd company-addressbook
→ ls
→ cd src
→ cd database
→ ls

→ sudo nano Database.php
→ cd ..
→ sudo nano Dockerfile
→ sudo nano Dockerfile_db
→ sudo nano docker-compose.yml
→ sudo docker-compose up --build -d → background

→ 172.16.51.43:8001
→ sudo docker ps
→ sudo docker exec -it mysql bash
→ ls
→ sudo cd docker-entrypoint-initdb.d → sources the database
→ ls
→ mysql -u user -p
→ password
→ show databases;
→ use database;
→ exit
→ cd /var/lib/mysql
→ cd database
→ ls

07/01/2024 Docker-compose:

→ Php-crud-employeeapp
→ git clone php-crud-employee-app
→ cd php-crud-employee-app
→ ls
→ sudo nano docker-compose.yml
→ sudo docker-compose up --build -d
→ sudo docker exec -it container_id bash
→ cd docker-entrypoint-initdb.d/
→ ls
→ cd dump.sql
→ sudo mysql -u user1 -p

cd /var/lib/mysql
ls

Jenkins :-

-> php crup app : Freestyle project
-> Build steps - Docker compose build step
-> start all services

20/05/2024 Building a docker image using Jenkins :-

1) Select git
2) Add git repo URL
3) Build step - Docker build & publish plugin
4) We need to mention Dockerhub credentials, tag & image name & provide Dockerfile name
5) Build step - Execute docker command.
6) pull image
   nidhisraol/mycafe website/final
6) Build step - execute docker command
   create container
   Image name : nidhisrao/mycafewebsite
   command :- it -d
   container name : mycafe appdemo
   Advanced mode :-
   Post binding : 4000:80

7) Build step - Execute docker command
   Start container(s)
   name → container name

Modifying the system deamon file so that the system accept the external command

a) ExecStart = /usr/bin/dockerd -H fd:// -H tcp://127.0.0.1:2375
By default docker command listens to internal commands
b) cd /lib/systemd/system
c) nano docker.service
d) docker restart

Docker compose file :- Docker compose-php-crud-file

~~Build step : Docker compose Build step~~
~~Docker-compose file : ./public/docker/docker-compose.yml~~

Build step
→ Select Docker compose build step
→ In command ~~se~~ select run all services
→ If we want to give the custom path then select "Use Custom Docker Compose File"
In Docker Compose File - give the path
./public/docker/docker-compose.yml

Kubernetes :-

POD - Basic unit
API server - Gate keeper

Components to be installed :-
kubectl
kubeadm
kubernetes-cni
kubelet - primary component in worker node
container runtime engine (docker)

1) sudo kubeadm init → In master
   → Pull images of scheduler, API server & all services & starts the services as container
2) Get the token
3) In the slave do →
   kube join token (add the token got from master)

21/02/2024 Remediation

Deployment file - Instruction for creating/deploying POD and the no. of replicas to run.
→ Deployment is the higher level abstraction
In this we will define we want to create one POD & no. of replicas.
→ Service - to expose the containered application to outside world.
→ Replication set - kubernetes object
→ If we want to delete any kubernetes object kubectl should send request to API server.

Date 21/02/2021
Page

Create & expose deployment :- (Imperative way)

→ ssh chefserver@172.16.51.53
→ Log in to kubernetes master
→ Get details - sudo kubectl get nodes
The node having control-pane role is the master
→ kubectl get pods
Get pods running in default namespaces
→ sudo kubectl get pods --all namespaces
Get pods running in all namespaces.
→ Create a deployment, in that create a pod & run container
sudo kubectl create deployment myngnix --image=nginx:latest --replicas=2
Two pods with myngix will be created. Replicaset manager will manage it.
→ Get details of deployment
sudo kubectl get deployments

→ User created pods, if no namespace provided will be created in the default namespace.
→ Replicas will be created in different hosts.
sudo kubectl get pods

→ Create the service - expose the deployment.
3 types of services
→ Load balancer - Distribute the requests
→ node port - converts or redirect port 80 to 30000 to 60000
→ ~~External Name~~ Cluster IP - Object/Application accessible only within cluster
sudo kubectl expose deployment myngix --type=LoadBalancer --port=80

→ sudo kubectl get services
or
sudo kubectl get svc
} Get service details

classmate
Date 21/02/2024
Page

→ To know on which worker node the pod is running
sudo kubectl describe pods myngint -dx
ID number

Pull a docker image and publish on kubernetes:-

→ sudo kubectl create deployment cafeapp --image=sudocker123/cafedockerapp:v1 → Replica 1
→ sudo kubectl get deployments
→ sudo kubectl get pods
→ sudo kubectl expose deployment cafeapp --type=NodePort --port=80
→ sudo kubectl get svc

sudo swapoff -a

26/02/2024

Declarative way of using yaml file - yet another markup file:-

→ clone from git
→ cd basic php website
→ ls
→ 2 files → deployment (abstraction - creates a pod)
Inside pod → tell container which image to run
→ sudo nano bookalbum-deployment.yaml (specified here)
(labels have key-value pair)

To run this yaml file:-

→ sudo kubectl apply -f bookalbum-deployment.yaml
create
→ sudo kubectl get deployments
→ sudo kubectl get pods

- sudo nano bookalbum-service.yaml
  contains version, service, labels app: bookalbum (same created pod)

→ sudo kubectl create -f bookalbum_service.yaml
→ sudo kubectl get svc (services) → To get the po
→ Browser → hit the IP address

If rebooted we have to export the .kube/conf file
→

28/02/2024 With Jenkins deploying an application in kubernetes:-

1st stage:-

1) Create book_album_deploy_build_server
2) Restrict on slave
3) Git source code
4) Build steps
   Execute shell
5) Build steps
   Execute shell
   echo "Build is successful"
   echo "Unit & Integration testing started"

2nd stage:-

1) Book_album_deploy_staging_docker
2) Restrict on slave
3) Build when
4) Docker build & publish
   repo name -
   tag -

Stage 3 :

1) Book_album_deploy_prod_kubernetes
2) Restrict where to run
3) Source code
4) Build after test project
5) Build steps (Deploy to Kubernetes)
6) Add config file
7) Config files - bookalbum_deployment.yaml, bookalbum_service.yaml

To add config file :-

→ Manage Jenkins
→ Credentials
→ System
→ ID - MSIS_kubernetes
→ content - copy the .kube/config file content

Plugins install for deployment to kubernetes :-

→ Advanced mechanism
→ ~~Browse for file~~
→ Plugins
→ Advanced
→ Browse the hpis file & select

kube master
username = chefserver
pass = chef@123
ip = 172.16.51.53

slave
ip = 172.16.51.50
user = ansible
soft@123

Continuous monitoring :-

Tools :- Splunk, Amazon CloudWatch, Nagios, Kibana + Elastic + logstash

Nagios Remote Plugins Executor - NRPE

Setting up our infrastructure monitoring :-

→ Copy the Nagios file to /usr/local/nagios
→ cd /usr/local/nagios
→ ls
→ cd libexec
→ ls
→ This will be containing all installed plugins. We need to copy the plugins executables to this folder
→ cd ..
→ ls
→ cd etc/objects → own configuration is present
→ sudo nano commands.cfg

Ex:- command_name check_local_disk
command_line $USER/check_disk -w $ARG1$ -c $ARG2$ -p $ARG3$

command_name check-host-alive
command_line $USER1$/check_ping -H ...

→ cd ..
→ sudo nano MyLinuxHost001.cfg
→ MyLinuxHost001.cfg - Define the host, IP address, services