---------What is three tire architecture---------

When we host Three tire application =>

There are three layer is approach called 3-tire architecture.

* First layer is web server=> it is act as reverse proxy
* Second layer is application server=> it has business logic
* Third layer is database server=> it provide database



Application Server->

* In application server there is logic only not a storage
* It has a request for data to database server and get response from them.
* User can interact with application server and application server connect to database server for data

Database ->

* For storage we can use s3 or database
* But in s3 we save static data
* For save dynamic data or tabular format data we need to create new database using RDS service

RDS=> RDS relational database server

* It used create the database server in aws
* It is a Paas(platform as a service) because it provide installing database ,maintaining database ,and configuration also.



Create RDS=>

* Search RDS in aws Services
* Click on create databases then create database
* Click on standard create
* Click on MySql
* In templets we choose free tire
* In setting enter DB instance name
* Then enter username and password
* Then choose cpu and RAM =>
* if we choose 4cpu and 16GB RAM RDS automatically create EC2 instance with 4 cpu and 16 gb RAM in any availability zone, we cant see this ec2 instance and we can not login in that ,it on call by database software that is called as PaaS.
* If we create RDS in one availability zone and this availability zone is not access then we cant access the RDS also
* To resolve above problem in aws enabling multi AZ(multiple availability zone) concept if we enable muti AZ then RDS created in multiple availability zone and one availability zone is expired then another availability zone are accessed.
* If we create RDS in 1a zone it act as a primary and 1b act as a standby if we access primary then standby is not working if expire the primary then automatically activate standby it continue the work.
* There is two ec2 instances one is primary and standby RDS has know which is primary and which is standby
* Click on create database
* It was created and there is a endpoint it is a url which we use to connect with RDS for that it take some time.
* Then create the security group and attach it
* For it search security group and click on it
* In security group creation time add name of security group and description.
* In inbound click on add rule
* Select type MySQL
* In source select allow for anywhere
* Then click on create.
* Comeback in RDS console add group
* In monitoring we check all information about RDS like cpu utilization, storage if any issue ocured then check the monitoring and fix the issue

Connect RDS to local mysql=>

* Copy the endpoint url of RDS
* Open MySQL of local machine and enter connection name
* Add endpoint url in hostname
* Enter user name of RDS mysql end enter password which is DB identifier in RDS
* Click on test connection

Proxy Server(ngnix) or web server=>

Proxy server and web server almost same



* Proxy server gate request from user and send it to application server
* It is not actually present it only forward request to backend.
* If we use this we protect ip address of application server and RDS from internet because proxy server don’t directly connect with database
* There are two type of proxy forward proxy and reverse proxy
* If client knows he gate data from which actual server(application) is known as forward proxy
* If client don’t know about application server he know about only proxy server then it known as reverse proxy.

When we host Three tire application =>

There are three layer is approach called 3-tire architecture.

* First layer is web server=> it is act as reverse proxy
* Second layer is application server=> it has business logic
* Third layer is database server=> it provide database

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Deploy 3-tire application by creating 3-tire infrastucture

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

There are three layers =>

* Webserver
* Application server
* Rds database

To create three tire infrastructure first=>

* Create VPC first

In that enter name of VPC and enter CIDR is 10.0.0.0/16 for ipV4

* Create subnet private 2 and public 2 in different availability zone
* Enter subnet name
* Select vpc which is you created
* Select availability zone
* Eneter CIDR 10.0.0.0/24 or 10.0.1.0/24 or 10.0.2.0/24, 10.0.3.0/24
* Click on create subnet
* Like vise create 4 vpc 2 for private 2 for public
* Create internet gateway
* Create nat gateway

4 Create Rout Table

* Create 4 rout table for each subnet
* Click on create rout table
* Enter name like private rout table
* Select vpc
* Click on create rout table then it was created
* Then click on this rout table id
* Go to the subnet association in that go to the edit subnet association silect our subnet
* Then click on rout and click on edit rout
* Select gateway like natgatway for private rout table and internet for public rout table
* Same process follow for 4 rout table
* Create security group
* Public security group=> in that select vpc which is you created and add inbound rule select HTTP and access anywhere
* Private security group=> aslo select vpc and add inbound rule port no 8000 or 8081
* Create sg for RDS=> select vpc and add inbound rule add rule select MYSQL and post no 3306
* So that wise we created 3 SG
* Create RDS=>
* Go to RDS
* Click on create database
* Select database
* Enter credentials
* Select VPC
* Slect subnet
* Select security group which is created for RDS
* Click on additional configuration and enter database name
* Create servers means EC2 instances

1.web server=>

* Select machine which is free
* Select type t2 micro
* While creating that select vpc
* Select public subnet
* Enable setings
* Select public security group

2. create bootstrap server as web server same configuration

3.application server=> create it using private subnet and private security group

-------------------Here our 3-tire application is created.------------------------

Deploy Application=>

1. connect bootstrap server
2. create key.pem using copying content of private key
3. connect application server using priate ip address and key.pem

* apt install git
* get filr in server: git clone <https://github.com/kadamsagar039/new_chatapp>
* cd new\_chatap
* update server: apt update -y
* install liabrarys: sudo apt-get install -y python3-pip
* sudo apt-get install -y python3-venv
* sudo apt-get install libmysqlclient-dev
* install environment: python3.8 -m venv env
* activate encironment: source env/bin/activate
* pip3.8 freeze
* pip3.8 install wheel
* pip3.8 install -r requirements.txt
* in that file add our RDS configuration: nano settings.py
* python3.8 manage.py migrate
* pip3.8 install mysqlclient
* pip3.8 install django-environ
* python3.8 manage.py migrate
* telnet chatappdemo.cnuddw0eyqax.ap-south-1.rds.amazonaws.com 3306
* run our application: gunicorn --workers 3 --bind 10.0.1.183:8081 fundoo.wsgi:application
* run our applicatin in background : nohup gunicorn --workers 3 --bind 10.0.1.183:8081 fundoo.wsgi:application &

Configuration for webserver

1 connect web server

2 install nginx

3 cd /etc/nginx/

4 cd sites-available/

5 create file: cat >chatapp

6 chmod 744 chatapp

7 nano chatapp

=> add following text in that and add private ip address of application server

server {

listen 80;

server\_name \_;

location = /favicon.ico { access\_log off; log\_not\_found off; }

location / {

include proxy\_params;

proxy\_pass [http://10.0.0.0:8081](http://10.0.0.0:8081/);

}

}

8 cd sites-enabled/

9 rm -rf default

10 ln -s /etc/nginx/sites-available/chatapp /etc/nginx/sites-enabled/chatapp

11 nginx -s reload

here server configuration is done so copy web server public ip address and hit it on browser

Load Balancing=>

* If we host our website in server and incoming traffic of this website is increased but this server configuration don’t has a capacity to handle this traffic.
* In this case cpu utilization and memory utilization automatically increased and cross the deadline and server getting crashed .
* To resolve this problem in aws using load balancing
* In load balancing we setup multiple server in backend so traffic of this website is distributed in backend multiple servers.

Auto Scaling=>

* if any condition incoming traffic of this website lots of high and backend servers also cant handle this traffic .
* in that condition backend severs(load balancer ) crashed.
* And creating new backend server is gating time.
* To resolve the above problem in aws using auto scaling
* In Auto scaling we set the cpu utilization criteria for eg. We set 70 % cpu utilization if cpu utilization reach the 70% then aws create automatically new server with same configuration as a old server.
* And distribute load on that server in that way it can avoid server crash.

Create Load Balancer=>

* Go to load balancer
* Click on create load balancer
* There two load balancer application load balancer and classic load balancer , application load is a enhance version it give the feature of port forwarding means if you open any other port but your website is in other port then you also map.
* Click on preferred HTTPS and click on continue
* Enter the name
* Select internet facing if your website is in internet ,or select internal and add vpc
* In listener select http or https
* Choose vpc
* Select subnet (if there is subnets in different availability zone then its very good )

Load balancer=>

* it takes request from client and forward it to web server which is highly available
* it may be there is multiple web server but load balancer forward request to highly available server
* load balancer act publically by using load balancer we protect expose web server also from internate