(372) Demo | Three-tier web app in AWS with VPC, ALB, EC2 & RDS - YouTube

https://www.youtube.com/watch?v=sCBTeMd0Jj4

Transcript:

(00:00) hello and welcome to my youtube channel if you are watching my video for the first time please consider subscribing to the channel i've planned to create more such videos on cloud devops and micro services in the near future in today's video we are going to deploy a 3-d web application on aws so let's start the video before we can actually begin deploying the application components we will need to create the networking base for the same in this use case we will create a vpc three public subnets that is one subnet in each

(00:28) availability zone this subnet will have our internet facing load balancers web servers and jump servers we will then create a private app subnet in each availability zone for application deployment and finally we will create private subnets for db instances once the vpc and subnets are done we will require gateways to configure incoming and outgoing traffic firstly internet gateway needs to be configured with the vpc and then a net gateway needs to be created in any of the public subnets as a best practice it is recommended to

(01:00) create multiple net gateways for redundancy but for this demo we will stick with the single nat gateway people often make the mistake of creating the nat gateway in the private subnet although the nat gateway is used to facilitate outgoing internet traffic for private subnets it needs to reside in the public subnet after the gateways are created we will need to create the corresponding route tables for web app and db subnets once the underlying networking infrastructure to host a three-tier application is created we can start with

(01:30) the application components for this use case route 53 is optional as we can directly access the application via application load balancer then we will create two ec2 instances on both availability zones and install php and apache on it and finally we will create the rds instances here the multi-ac configuration is optional now let's quickly jump into the actual implementation under services section let's go to vpc click on vpcs click on the vpc id you'll notice that the default vpc is yes so we are not going to use that vpc

(02:08) instead we will create our own vpc so we will name the vpc as myvpc the cidr would be zero slash 172.20.0.0.20 we will keep the vp v6 cidr block as is we'll keep the tenancy as default then we'll hit the create vpc button we will quickly verify whether the vpc is created as expected then we will move on to the subnets okay so here also we will notice that the subnets are already created but we will not use that as the subnets are the default ones so we will start the subnet creation so we'll give the name my public

(02:53) web subnet one select the availability zone as us is to a put the vp4 cidr block as 1720 1.0 slash 24 keep the tags as is let's create the second subnet uh put the name as my public web subnet 2. availability zone as us is 2b cidr block as 172 20 2.0 24. now let's create the third public subnet we will keep the same naming convention my public web subnet 3 cidr block as 1720.3.0 slash 24.

(03:42) we'll select the availability zone as 2c will quickly verify whether the subnets are as expected once that is done we will move to creation of app subnets so our naming convention would be my private app subnet one the availability zone would be us is to a cid range would be 172 20 4.0 24 similarly create the second subnet my private app subnet 2 availability zone as 2b cida range as 172 20 5.

(04:43) 0 24. now let's create the third one my private app subnet 3 availability zone as 2c cidr block as 172 20 6.0 24 now let's move to the last year that is the db tier let's put the naming convention as my private db subnet one will select the availability zone as 2a the side range would be 1720 7.0 24 let's create the second db subnet

(05:55) my private db subnet 2 let's select the availability zone as 2b the cidr range would be 1720 8.0 24. now let's create the last subnet my private db subnet 3 availability zone as 2c cidr ranges 172 20 9.0 24. now quickly verify whether all the subnets are created as expected do once verified and now let's click the create subnet button now the subnet creation is in progress

(07:02) now quickly verify whether all the subnets are created okay so once the subnets are created let's move to the creation of route tables so the first route table would be my public route table we will select the vpc that we created and click on the create route table button now let's create the route table for app player my private app route table select the vpc that we created click on the create route table button and finally we will create the route table for db so the naming convention would be my private db route table

(08:14) so quickly verify whether the route tables are created so once the route tables are created we need to associate the route tables with the appropriate subnets okay so we have selected the web route table let's associate the route tables with subnets the web subnets let's do the same for app route table as well and finally the db route table once the route tables are created we need to establish the connectivity for that we will require internet gateway for public web subnet

(09:19) let's name the internet gateway as my internet gateway and click on the create internet gateway button once the internet gateway is created we have to also attach to the vpc so let's do that okay once the internet gateway is created we have to move to nat gateways let's create the net gateway give the name as my nat gateway select the public subnet one for this demo we are going to create the net gateway in only one subnet but as a best practice you should create multiple nat gateways to ensure redundancy also allocate the elastic ip to the net

(09:58) gateway and let's create the nat gateway so this will take some time so our nat gateway is created so once the internet gateway under net gateway is created we have to make changes in the route tables so that the connectivity to the internet will be established from your internet gateway for the public subnets and net gateway for the private subnets so we will add a route in the public route table for 0.

(10:43) 0.0 0 and will point it to internet gateway similarly we will make changes to the app route table we will create a route for app route table we'll add 0.0.0.0 for nad gateway and then finally we will do the same for db subnet as well or the db route table as well so we'll add the same entry 0.

(11:24) 0.0.0 0 and point it to net gateway so this completes our networking configuration so we have created vpc's route table subnets internet gateway net gateway we have associated them now let's start deploying the application components so we will go to ec2 instance click on the instances click on the launch instances we'll select the amazon linux 2mi we will select t2 micro we will select the vpc that we created we will select the public subnet because this would be our jump server select the enable for auto assign ip storage we can keep as is tags we can add a name tag

(12:17) we can give the name as my jump server will create a new security group we'll keep the same naming convention my jump server hg now let's click on launch create a new key pair my key pair dot bam click on launch instances please ensure that you have downloaded the pam key now let's create the php instances select the amazon linux 2mi keep the instance type as t2 micro select the vpc that we created select the app subnet

(13:25) we can keep the storage as it is now let's add the tag name my php app server one will create a new security group for the app layer so give the name as my php app server hd we will allow the ssh connection for this app server only from the jump server so let's remove the entry and put the my jump server is g click on launch we already have the key pair click on launch instances this will take some time meanwhile the instance is creating we will create the other php server so we will follow the similar configuration we'll select the ami as

(14:23) amazon linux we'll select the instance type as t2 micro we'll select the vpc that we created we will select the other app subnet will keep the storage as it is we'll add the tag name my php app server 2 the security group is already created we will select that security group the key is already there so we will acknowledge and launch the instances so all of the servers are in running state now let's try to connect to the jump server from our local machine you can get the connectivity details from the connect button

(15:24) we have already downloaded the key pair we will give the appropriate permissions to the key pair and let's try to the now let's try to connect to the server so we have logged into the jump server now let's try to copy our ssh key give the appropriate permission to the ssh key now we have to connect to the php app server click on the php app server one click on the connect button you will get the example connectivity paste that in the jump server now we are logged into our app server now we have to install php and apache on

(16:26) the tab server so let's follow the documentation from aws i will provide the link to the documentation in the description so let's follow all the steps on the app server so we'll do a yam update then we'll install the required packages now install our apache server we do not require mariadb because we are going to use the rds instance so our apache is installed

(17:34) we'll start the apache server now we will enable the apache service we will verify whether the apache service is created now let's check whether the httpd service is enabled and it is working so yeah it works now let's give appropriate permission to our ec2 user change the ownership of our www folder

(19:09) so as a sample application we will install php myadmin so we'll follow all the steps required to install phpmyadmin so we are restarting the apache service we are restarting the php fpm service now let's navigate to html folder download the phpmyadmin package so once the packages are downloaded we will create a directory for phpmyadmin and we will enter that files and we will remove the php my admin tar file so for now we are done with the configuration of phpmyadmin at a later stage when rds is created we will make

(20:01) certain changes to the config file so once the app instances are created we will move towards creation of load balancer so let's create load balancer select application load balancer give the name as my alb scheme should be internet facing because we want to expose the load balancer to the outside world select the vpc select the availability zones as public subnets now go to the security group settings create a new security group give the name as my albsg source000 is fine create a new target group give the name

(20:50) as my alb php tg target type is instances let's keep the port as 80 now register the targets we have already created the php servers now let's select those and click add to register button now create the album so we have now successfully created the load balancer now check the target group let's see whether the instances are in the healthy state right now they are in the initial state as the target registration is in process by the time our instances are getting registered in the target group we need to allow the load balancer

(21:44) security group in our app server security group so let's make the changes click on edit inbound rules add a rule select port 80 and select the security group of alb so this will establish connectivity between your alb and instances so now let's go to load balancer section and copy the dns name paste the dns name in the browser and let's see whether the test page is appearing yeah it does now we are able to see the apache test page now we have to check whether the load balancing is happening as expected so we have to create an index.html file

(22:33) in the where www.html folder in both the php app servers to validate whether the requests are going to both the app servers we will put a sample text in the index.html page as php server 1 and php server 2. once the files are saved in both the servers we will navigate to index.html on the web browser to check whether the requests are flowing to both the app servers so we can see on the screen that it is going to php server 1 and php server 2.

(23:11) so now we have created two layers of the three tier architecture the web layer and app layer now let's move to the third layer that is the db layer now let's navigate to rds click on the subnet groups as we have created separate subnets for db instances give the name as my db subnet group we can copy the same and put it in the description choose the vpc that we have created choose the availability zones 2 and 2b we have to select the subnets that were created for db and click on create so once our subnet groups are created we

(23:54) will move to creation of the actual db instance so let's create database select standard create mysql click the free tier give the instance name as my db we'll keep the master username as is let's put the password confirm the password keep the instance class as t2 micro disable the storage auto scaling in connectivity we will select the vpc that we created the dbsubnet group is selected by default we will create a new security group name that is mydbhg and will create the database once our database is created we will

(24:54) need to enable the connectivity between your app server and the db server so we have to go to the security group of the db and we have to add one entry for the php app servers so in the inbound rules we will add a rule for 3306 and in the source we will put the app server security group and we will delete the default rule that was created and let's save the rules the creating will take some time so by the time it is in backup stage let's copy the endpoint we will go to the php server

(25:58) we will navigate to the phpmyadmin folder under the phpmyadmin folder we will rename the file config.map.inc.php to to config.inc.php now let's open that file search the host search for host replace localhost with the copied hostname of the rds instance that we just created save the file and on the browser navigate to phpmyadmin and check whether the application is running or not now once you punch in your credentials you'll see the error failed to set session cookie this happens because phpmyadmin is a stateful application the

(26:55) session is stored on individual servers since the load balancing is configured as round robin requests are going to both the servers as a workaround what we can do is go to target groups select our target group go to the attribute tab and enable session stickiness with the load balancer generated cookie this way your request will flow to only one server and in case of a failure it will be redirected to the other server once the changes are made we can open the incognito mode of the browser and check the phpmyadmin now you will be

(27:32) able to login to the application now let's do a quick recap of all the resources we created first of all we created the vpc then the corresponding subnets then the route tables then the internet gateway and attached it to the vpc then we created the nad gateway in the public subnet then we created the jump server and the php app servers then we moved on to the load balancer we created a target group and a load balancer and finally we created the rds instance thanks for watching if you like this video please hit the like button and

(28:25) subscribe to my channel