Transcript Summary Process

I will be providing you transcript of 6 videos of a devops project that i done using a course on udemy one by one . I want you to make notes for me out of them by analyzing briefly , add or exempt details for my better learning and also to revise what i did in that
so should i start giving you all the transcript in one go but it will be a long prompt may exceed the limit or should i give one by one

It's best to provide the transcripts one by one. This way, I can analyze each one thoroughly and create detailed notes for you. Please go ahead and share the first transcript when you're ready.

Welcome to the project.

This is an AWS cloud computing project.

The name of this project is lift and shift application workload, and we're going to lift our application,

the vprofile and shift it on AWS cloud.

Let's understand about this project, In previous project, we have seen a multi-tier Web application

stack vprofile by using Vagrant .In this project, we are going to host and run it on AWS cloud

for production.

And we're going to use a lift and shift strategy for this.

After going through this project, you will learn how to run application workload on AWS cloud

by using lift and shift strategy.

Let's understand the scenario.

We have, let's say, application services, which are running on physical or virtual machines, there

could be various services like databases, Postgres, Oracle, application services like Tomcat, even

LAMP Stack, DNS services and various kinds of services that powers our application.

And we have all this work in our data center.

So, so many servers running varieties of services on your local data centre.

To manage all this, you will need multiple teams working around the clock.

We will need virtualization team for running virtualization platform, data center, operations team

for data centre & related operations work, monitoring team to monitor 24/7 and sys admin team,

of course.

Managing all these services, servers and teams is complex.

It becomes more complex if you want to scale up or scale down, which needs to be done very regularly.

There's a huge cost for procuring all these resources and also regular maintenance cost.

Most of the processes in this will be manual.

If you have a virtualization layer on top of it, it is possible to automate those things, but it's

really difficult to do it and also to maintain it.

And not to mention all these things are very time consuming.

Solution to all this problem is to have a cloud computing setup, so instead of running our workload

in our data center, we run it on a cloud computing platform that we don't pay for the upfront cost

for procuring the resource. We pay as we go.

Consuming infrastructure as a service.

Just like electricity.

We get flexibility, it's elastic in nature, we can scale out or scale in and really control our cost.

So managing infrastructure becomes easier.

And most important, we can do automation, we can automate each and every step and process to avoid

human errors and save our time, of course.

We're using AWS, cloud computing, and we'll see what all the services that we are going to use

in this project, starting with ec2 instances.

Ec2 instances will be our VMs for Tomcat, rabbitmq, Memcache and mysql servers. We will be

also using elastic load balancer, which will be replacement of our engine service.

Will be using auto scaling service, which will automatically scale out and scale in our ec2 instances,

which will automatically control our resources and also our cost.

For storage, if you'll be using S3 or EFS so and also Route 53 for a private DNS service.

Along with these we will be using few more services like IAM, ACM, EBS etc

Let's make sure our objectives are clear.

We want a flexible infrastructure, we want to pay as we go model.

We don't want to be upfront cost.

We would like to modernize our application more effectively by using AWS services.

And we also want automation.

Infrastructure as a code.

Now we will see architectural design of AWS set up that we will be creating.

We are using ec2 instances, elastic load balancer.

Auto scaling.

S3 for storage.

Amazon certificate manager and Route 53 service.

Well, this is the stack from our previous project.

We had all the service on virtual machines on our computer Nginx, Apache, Tomcat, RabbittMQ,

Memcache and MySQL, we are going to shift this stack on AWS Cloud.

Once we have our stack on AWS Cloud, our architectural design will be like this.

Users will access our website by using a URL and that URL we be pointing are told, will be point to an end point.

This entry will be mentioned in GoDaddy DNS.

User browsers or the app will use this end point then to connect to the load balancer, by using https

Certificate for https encryption will be mentioned in ACM Amazon Certificate Manager Service.

So user will access application load balancer endpoint.

Our load balancer will be in a security group and will only allow https traffic.

And then our application load balancer, will route the request to Tomcat instances. Apache tomcat

service, will be running on some set of instances which will be managed by our auto scaling group.

So as for high or low load, these instances capacity will be scaled out or scaled in.

These ec2 instances where Tomcat is running, will be in a separate security group and will only allow

traffic on Port 8080 only from a load balancer.

We know application, our vprofile application sits on Tomcat instance.

We have seen this in our previous project.

And our application needs backend servers, which are MySQL, Memcache and RabbitMQ. Information

of backend services or the backend server IP address will be mentioned in Route 53 private DNA zone.

So Tomcat instances will access back server with a name which will be mentioned in Route 53.

Private dns where the private IP address of our background servers will be mentioned.

These backend ec2 instances, which will be running my sql, RabbitMQ Memcache will be in

a separate security group.

So the AWS. sources, which are in use over here are first Amazon certificate manager for a certificate

application load balancer.

Set of ec2 instances for Tomcat, Memcache Rabbit, MQ and my sql, three separate security groups.

Amazon Route 53 for DNS Private Zones.

And also, there's Amazon S3 bucket to store our software artifacts.

Now, I recommend you pause the video and watch this architectural diagram once again.

Will start the execution now, and this is the flow of execution, first, we will log into our AWS

account.

We're going to key pairs, which will use to login to our ec2 instances.

We create security groups for load balancer, Tomcat and backend services. We will launch instances with

user data, which will be our bash scripts. We will update IP to name mapping in Route 53.

We're going to then build our application from source code, this will do it on our local machine,

on our laptop.

And we upload our artifact to a s3 bucket. From S3 bucket we will download our artifact to ec2

instance, where tomcat service will be running.

Then we'll set up a load balancer with https connection.

Will map our elastic load balancer and point to a website in GoDaddy DNS and will verify. Once we verified

our entire set up, then we'll build an auto scaling group for a Tomcat instances.

So now it's time to dive into AWS.

Notes on the Lift and Shift Application Workload Project

Project Overview

- **Objective**: Migrate the vProfile multi-tier web application from a local data center to AWS using a lift and shift strategy.
- **Goal**: Learn how to run application workloads on AWS cloud.

Background

- **Previous Setup**: Multi-tier Web application (vProfile) hosted using Vagrant.
- **New Setup**: Host and run the vProfile application on AWS for production.

Scenario

- **Current Environment**: Application services running on physical/virtual machines in a data center.
  - Services include: Databases (Postgres, Oracle), Application services (Tomcat, LAMP Stack), DNS services, etc.
  - Challenges: High complexity, multiple teams required, manual processes, high costs, difficult scaling.
- **Solution**: Cloud computing setup (AWS).

  - Benefits: Pay-as-you-go model, elasticity, automation, reduced management complexity, cost control.

AWS Services Used

- **Compute**: EC2 instances for Tomcat, RabbitMQ, Memcache, and MySQL servers.
- **Load Balancing**: Elastic Load Balancer (ELB) to replace Nginx service.
- **Scaling**: Auto Scaling service to manage EC2 instances based on load.
- **Storage**: S3 or EFS for storage.
- **DNS**: Route 53 for private DNS services.
- **Security**: IAM for access management, ACM for certificates, EBS for block storage.
- **Automation**: Infrastructure as Code (IaC).

Project Objectives

1. Create a flexible infrastructure.
2. Utilize a pay-as-you-go model.
3. Modernize application using AWS services.
4. Implement automation through Infrastructure as Code.

AWS Architectural Design

1. **User Access**: Users access the application via a URL pointing to an endpoint in GoDaddy DNS.
2. **Load Balancer**:
   - Uses HTTPS encryption with certificates from ACM.
   - Allows only HTTPS traffic.
3. **Application Servers**:
   - Requests routed to Tomcat instances managed by an auto-scaling group.
   - Instances in a security group allowing traffic on port 8080 from the load balancer only.
4. **Backend Services**:
   - Tomcat accesses backend servers (MySQL, Memcache, RabbitMQ) using private DNS names in Route 53.
   - Backend EC2 instances in separate security groups.

Execution Flow

1. **Login to AWS**: Access AWS account and create key pairs for EC2 instance access.
2. **Security Groups**: Create security groups for load balancer, Tomcat, and backend services.
3. **Launch Instances**: Deploy EC2 instances using user data (bash scripts).
4. **Route 53 Setup**: Update IP to name mapping in Route 53.
5. **Build Application**: Compile the application source code on a local machine.
6. **Artifact Storage**: Upload the artifact to an S3 bucket and download it to the Tomcat EC2 instance.
7. **Load Balancer Setup**: Configure the load balancer with HTTPS.
8. **DNS Configuration**: Map the ELB endpoint to the website in GoDaddy DNS.
9. **Verification**: Verify the setup and build an auto-scaling group for Tomcat instances.

Next Steps

- **Action**: Dive into AWS and start the execution process.

These notes provide a detailed summary of the project, highlighting the main objectives, architectural design, and execution steps for better understanding and revision.

Is it better this way or i think you should take inputs of all transcript first

It's up to you, but here are the benefits of each approach:

One-by-One Transcripts

- **Detailed Analysis**: Each section can be thoroughly analyzed and understood.
- **Immediate Feedback**: You get immediate notes and can provide feedback or ask questions before moving on to the next part.
- **Manageable Chunks**: Easier to process and less overwhelming.

All Transcripts at Once

- **Comprehensive View**: Allows for a holistic understanding of the entire project before summarizing.
- **Contextual Connections**: Easier to see how different parts of the project connect and relate to each other.
- **Streamlined Process**: Might be faster overall as all information is available at once.

If you want detailed, immediate notes and feedback, proceed one by one. If you prefer a comprehensive summary and can wait until all transcripts are provided, then sharing all at once might be better.
Let me know how you would like to proceed!

one by one is better

Great! Please share the next transcript, and I'll continue making detailed notes for you.

Udemy
DevOps Beginners to Advanced with Projects
0:08 / 11:40
Transcript
Imran: Let's dig in.

I am logged into my AWS account.

I'm in North Virginia region.

And I also have a certificate

which I'll be using for a HTTPS connection on load balancer.

I have mentioned the prerequisite video,

how to create this certificate

and do its validation from your domain provider.

I have a GoDaddy domain, groopyh.in,

and I have CNAME validation in this DNS zone.

Okay, let's go back to AWS console.

I'll go to EC2 service.

The first thing that we'll do

we'll create security groups.

First, we'll create security group for our load balancer.

So let's go to Security Groups.

Create Security Group.

Let's give a name.

Let's give some description.

And let's add the rules.

Inbound rules.

First we'll say port 80 is allowed from anywhere.

We are publicly hosting our website.

And we'll remove this rule later

because we'll only have HTTPS login.

So HTTPS 443, allowed from anywhere again.

And let's save this.

Next we'll create security group for our Tomcat instance.

Vprofile-app-sg.

Let's add a rule.

So our Tomcat service will be running on port 8080

so let's give that port, and we'll allow traffic

only from our load balancer.

So here we'll select the security group

of the load balancer.

We'll give some description.

"Allow traffic from ELB."

And let's save this.

Now, before we move ahead and create our third

security group which is for backend services,

let's take a look again at the architectural diagram

and understand why we are specifying specific ports

in our backend security group.

So by now we have two security group.

One is for the application load balancer.

So this is going to allow connection on port 80 and 443

so user can connect from outside.

So it's allowed from anywhere.

The Tomcat instance security group

or application security group allows port 8080

from the load balancer security group.

So our load balancer can connect

to the instance on port 8080.

That's why we gave the rule port 8080

allowed from load balancer security group.

Our vprofile application will be living in the Tomcat

instance and it'll connect to the backend services.

Our backend services will be inside

the backend security group.

We have a MySQL, Memcache, RabbitMQ.

Now this is very essential for you to understand

when you work in a real-time environment,

your application will be having backend services

and there will be a configuration file

inside your application created by the developers.

So let's take a look at a vprofile source code

which has this backend configuration file.

Just keep in mind when I'm showing you the source code,

the configuration file, that will be in the Tomcat instance.

Artifact will be deployed in the Tomcat instance

and that will have that configuration file that will provide

information to connect to backend services.

Just keep this point in mind

when I'm showing you the configuration file.

Okay, so github.com/hkcoder/vprofile-project.

Just look.

You don't need to go into the source code for now.

Just take a look.

So this is our project source code, right?

SRC.

In this you have main.

In this you have resources.

And here you have application.properties file.

We have seen this file previously in the previous projects.

For now, just take a look at the port numbers.

So for MySQL, db01 host name and port number 3306.

So it's going to connect to MySQL service on port 3306.

So the security group of backend should allow port 3306

from the application security group.

Likewise, we have memcache and RabbitMQ.

So memcache 11211, and RabbitMQ 5672.

So we are configuring these three backend services, right?

So in the backend security group, we have to allow

these ports from the application security group

because our vprofile application

will be in the application security group.

It needs to connect to backend services.

So backend security group should allow these three ports.

Now, I get this question many times

when people work in real-time that they're not able

to connect to backend services.

And most of the time I have seen the problem

is with the firewall or wrong port number specified.

So I thought to explain this concept, once again,

in the middle of this lecture.

If you still have doubts, rewind the lecture, watch again.

And after completing this project, once again,

come back to this lecture and watch this again.

This is very, very important for real-time.

Not for this project.

We will execute this project and complete it successfully,

but for real-time and also for the upcoming projects

in this course, you will definitely need

to know this information.

So, one more security group.

This security group will be for our backend services

RabbitMQ, memcache and MySQL.

So we'll name it vprofile-backend-SG.

And let's add rule.

So we'll be running MySQL service over here.

So let's select MySQL, 3306 port.

Allowed only from our application security group.

Let's give a description.

We have two other sources

that will be running in the security group.

Memcache and RabbitMQ.

Memcache will be running on port 11211

and RabbitMQ will be running on port 5672.

11211.

Allowed from our security group, application security group.

And port 5672, again from application security group.

So Tomcat, which is in application security group,

can connect to our backend services on these ports.

Let's also give some description.

And now let's save this.

Okay, we need to make one more change in our security group.

In our backend security group we have three services

and these services will be also interacting with each other.

So we are going to give a rule over here

all traffic from itself, its own security group ID.

That's the final rule.

Let's save it.

So these rules will make our application functional

but if we want to do SSH to the instances

that we are going to launch, we need to add one more rule,

22 from my IP in both the security group.

Let's go to the app security group first.

Edit Inbound Rules.

Add rule.

22 from my IP.

And your IP might change daily.

So if you are accessing in the next day

you make sure you update your IP once again.

We will add here one more rule,

port 8080 from my IP.

So if you want to directly access the app server from the browser rather than from the load balancer

for troubleshooting purpose, later, when you need.

So you can do 8080 from my IP and you can directly access

the app server public IP on port 8080

and you should get the application webpage.

Okay, save this rule.

And now let's come to the backend security group.

Backend security group.

Edit inbound rule.

Add rule.

22 from my IP.

So we can SSH to all the backend machines when we need it.

Once again, you need to update it regularly

if your public IP is changing regularly.

Save Rules.

And in any case, if you make any mistake in adding the rule

we can come back and make the changes.

Now we will create a login key pair

which we'll use to log into our EC2 instances.

So go to Key Pairs, Create Key Pair.

Let's give a name, vprofile-prod-key.

I'm going to use Git Bash for login.

So I will use PEM format.

If you're going to use PUTTY then use PPK format.

If you want to import your own key, you can do that.

But we'll see these kinds

of security approach in other projects.

Notes on Setting Up Security Groups and Key Pairs

AWS Region and Prerequisites

- **Region**: North Virginia.
- **Certificate**: Pre-existing certificate for HTTPS connection on the load balancer, validated via CNAME in GoDaddy DNS (groopyh.in).

Security Groups Creation

1. **Load Balancer Security Group**:
    - **Name**: Not specified.
    - **Description**: Not specified.
    - **Inbound Rules**:
        - Port 80 (HTTP): Allowed from anywhere (will be removed later).
        - Port 443 (HTTPS): Allowed from anywhere.
2. **Tomcat Instance Security Group**:
    - **Name**: vprofile-app-sg.
    - **Description**: "Allow traffic from ELB."
    - **Inbound Rules**:
        - Port 8080: Allowed only from the load balancer security group.
3. **Backend Services Security Group**:
    - **Name**: vprofile-backend-SG.
    - **Inbound Rules**:
        - Port 3306 (MySQL): Allowed from the application security group.
        - Port 11211 (Memcache): Allowed from the application security group.
        - Port 5672 (RabbitMQ): Allowed from the application security group.
    - **Additional Rule**:
        - All traffic from itself (own security group ID) to allow backend services to interact with each other.

Configuration File Reference

- **Source Code Location**: github.com/hkcoder/vprofile-project
- **File**: `application.properties`
- **Port Numbers**:
    - MySQL: Port 3306.
    - Memcache: Port 11211.
    - RabbitMQ: Port 5672.

SSH and Direct Access Rules

1. **Tomcat Instance Security Group**:

- **Additional Rules**:
  - Port 22 (SSH): Allowed from your IP (needs regular updating).
  - Port 8080: Allowed from your IP (for direct browser access and troubleshooting).

2. **Backend Services Security Group**:
   - **Additional Rule**:
     - Port 22 (SSH): Allowed from your IP (needs regular updating).

Key Pair Creation

- **Name**: vprofile-prod-key.
- **Format**: PEM (for Git Bash); PPK (for PUTTY).
- **Option**: Can import your own key if preferred.

Summary

In this part of the project, we set up the necessary security groups and key pairs for our AWS environment:

- Created security groups for the load balancer, Tomcat instances, and backend services with specific inbound rules.
- Ensured proper port configurations for backend services based on the application's configuration file.
- Added SSH and direct access rules for easier management and troubleshooting.
- Generated a key pair for secure login to EC2 instances.

This setup ensures that our environment is secure and properly configured for the next steps of the project.

Instructor: Okay, so in this lecture

we are going to launch EC2 instances for all our services.

But before we do that,

I want you to verify one thing which is

security group outbound rules.

Inbound rules we have edited,

but many people make mistake

of editing outbound rules instead of inbound rules,

and when you do that outbound rule different than this,

what you're seeing right now,

if you're seeing outbound rule different than this, then your instance will not get any internet.

We will use user data scripts

to download packages from the internet

and your provisioning will fail

if your outbound rule is not like this.

So all traffic should go to anywhere from the instance

in the outbound rule,

and an inbound rule should be saying

what we discussed while creating the security group,

so you can check that as well.

Make sure it is for all the security group that we created.

Backend, outbound rule, edit outbound rule should be same,

what you're seeing on the screen.

And also verify once again the inbound rule

just to be careful, just to be sure that you did it correct.

Okay, after this we will clone the source code.

Before we clone the source code,

I want you to remove the previous source code.

Mine was in this location.

In a vprofile-project local setup we clone the source code

and I want you to delete that source code.

The reason because we made some changes into that,

and when you clone the latest source code,

you get the latest changes as well.

Then go to your browser, go to the url,

github.com/hkhcoder/vprofile-project

and here you will see a branch, aws-LiftAndShift.

That is the branch we will be using in this project.

So click on this dropdown code, HTTPS, copy the URL,

open your VS code,

click on that source control button, clone repository,

paste your repository there and hit enter.

Select the location.

I'm going to save it at this location, F hkhcoder,

select destination and wait for a minute or so to clone

and check few things in our source code

it's going to take a moment.

Okay, after a few seconds you should see here main branch.

Click on that.

And now you should see here other branches.

You have to select AWS lift and shift for this project.

Select it.

And now you should see a folder here, the user data.

Click on this.

Now, these are the same script

we used in vprofile project setup local with (indistinct).

One script is different, tomcat_ubuntu.sh.

We are going to install tomcat

or we're going to run tomcat on an Ubuntu instance.

And in Ubuntu you can install tomcat very easily.

After you install tomcat nine (indistinct) and you'll have system CTL file and everything,

you don't need to do the other kind of setup.

For code build and deploy process

that we did in previous project

through manually or through scripts,

that we will not do in the instance,

we are going to build the artifact locally on our computer,

on our laptop, push it to S3 bucket,

and from S3 bucket we're going to fetch in the instance.

And why we are doing it outside

and not in the instance directly,

that you will understand in the CICD projects,

but always keep in mind in real time on the servers,

we don't build the application on the server directly,

we build it outside, we version it,

we validate it and then we deploy it.

Now, we will be doing that

process CICD automation process later.

This will be manual build and deploy process.

But anyways, it will be out of the instance

and not in the instance, not in the server.

So we are going to start with the MySQL server

and it's the same script except the firewall rule,

which I want to talk now.

So our instance has a security group, which is a firewall, plus we are going to set up one more firewall

at the operating system level.

Firewalld is a service on Red hat based OS,

sent to us on Oracle Linux, Amazon Linux, Alma Linux,

all Red Hat based OS.

Our RPM based OS will have a firewall called firewalld.

That's the name of the firewall.

It's pretty simple, start and enable,

and we are going to add the port 3306 and reload.

If you start the service and you do not add the rule,

by default, all the traffic is blocked,

same like security group.

So now we have firewall at two levels,

at OS level and also at the security group level.

The rest, all the scripts are same from previous projects,

so let's go and start launching instance starting from db.

So come to instances and say launch instance.

Let's give the name tag.

I'm going to give vprofile-db01.

I recommend you keep the same.

Add additional tag.

We'll give one more tag, which is project and project name.

Ah, if I can type today.

Okay, so in real time you need to follow the standards

set by your organization for tagging.

Sometimes they're also called as naming conventions.

You may have tags like project environment,

owner and different kinds of tags.

If there are no standards set anyways,

make sure you add some tags

to identify your instances easily.

Belongs to which project,

what is the environment production?

Naming conventions, many people think it's optional.

It is optional but it is very, very important,

very, very essential to find your instances

or to sort your instances.

Okay, come down.

Okay, for this one we will go

with Amazon Linux 2023 AMI.

Amazon Linux is same as CentOS or AlmaLinux

or Red Hat Enterprise Linux.

And there's Amazon Linux 2023 AMI

and then there is Amazon Linux Two.

So we will be using 2023 AMI for this project.

So make sure it's selected.

By default, it is going to select Amazon Linux 2023 AMI.

Now, let's come down and select the instance type.

For this instance which we will be using for database,

T2 micro is enough for our project and it is also free.

So the AMI that we selected is free tier and the instance type is also free tier.

Make sure if you see a different instance type

it is T2 micro free tier eligible, select that one.

Key pair, the login key,

we are going to select vprofile-prod-key

that we created earlier.

And in network settings, select security group.

DB goes to backend security group.

Be careful.

Backend security group.

And then advanced details

and you come all the way down in user data

and let's go to the VS code

and copy our MySQL.sh script.

Make sure you copy the entire script

including the first line and paste it there

and say launch instance.

Okay.

So it takes time to launch the instance

and run the user data script.

So instead of waiting, we'll launch all the other instances

and then we can wait for some time.

Let's go to launch instance once again.

This time we will launch (indistinct).

Vprofile-mc01.

Add additional tag, add new tag, project, vprofile and come down.

AMI, the same Amazon Linux 2023 AMI

will be selected by default.

If it's not there, make sure you select this AMI.

Then come down.

Instance type, T2 micro.

If there is any other instance type,

make sure you select T2 micro from the list.

Key pair, vprofile-prod-key.

Security group, backend.

Once again, remember, backend.

RabbitMQ, Memcached, database, backend security group.

Advanced details.

User data.

Let's get the script for Memcached.

So copy the entire script including the first line.

Okay.

Always whenever you copy paste the user data script,

the shell script basically,

make sure the first line is this.

If this line does not exist then the script won't run.

Launch instance.

Now, RabbitMQ.

So launch instance, once again.

To tag name, name of the instance, project v profile.

AMI.

Once again, same, Amazon Linux 2023 AMI,

and instance type, T2 micro.

Key pair, vprofile-prod-key.

Security group,

this one also goes in the backend at security group.

Careful once again, backend security group.

Come down.

Advanced details.

Let's come to VS Code.

RabbitMQ.

Copy the entire script and paste it there.

Launch instance.

Okay, one more last instance pending.

App01.

This one will be with Ubuntu OS.

So launch instance.

Vprofile-app01.

Project vprofile, and select Ubuntu here.

Ubuntu comes in the quick start itself.

Ubuntu server 22 HVM SSD volume free tier eligible.

There are other Ubuntu AMIs also.

So make sure you select this one, okay?

If you click on dropdown,

you will see other Ubuntu server images.

Okay, be careful, select this one.

Now, every time I'm saying be careful

that is because for some AMI

you need to pay subscription fees

and some AMIs doesn't work for our need.

That's why.

Come down, T2 micro,

it's anyways selected t2 micro by default.

Key pair, vprofile-prod-key.

Security group.

Okay, tell me, which security group

apps that I will go into.

Yeah, right, app, security group.

The name itself we created, right?

It's app security group.

Come down, advanced details,

and it's pretty simple script to provision this instance.

Tomcat Ubuntu, a set of commands to install JDK and tomcat.

Remember our service name will be tomcat nine.

A specific name of the service will be, okay?

In local setup we have seen the service name was just tomcat

because we named it like that.

Here in Ubuntu when you install tomcat nine,

the service name is tomcat nine.

So system CTLs start tomcat nine,

system CTLs stop tomcat nine like that.

Okay, so if you did it very quickly like me...

Actually, most of the time I was pausing my recording

so I was not that fast,

but anyways, if you did it fast, wait for 10 minutes

because the instance comes up, right?

The initializing health check will be done.

Two by two health check.

One is on the hardware,

the other is for the virtual machine.

After that completes,

the user data script will start getting executed

and the execution will take some time.

So wait for around 10 minutes

before we log in and check.

So I'm going to pause my recording

for 10 minutes and come back.

Okay, after waiting for some time,

I'm going to check log into each and every instance

and check whether the service is running or not.

Yeah, refresh this one,

so it will show you the health check as passed,

otherwise it'll be showing the old information.

Okay.

So copy the IP address.

Oh, before that, select the instance and click on connect.

Need to find out the username to log in, right?

So go to SSH client, EC2-user.

That's the username, it's EC2-user.

Okay, coming back to the instance, vprofile-db,

I'm logging into db, make sure you selected db,

copy the public ip, open your Git Bash,

SSH -I, downloads, and vprofile-prod-key,

that's where my login key is saved.

Username, EC2-user, (indistinct) and paste the IP address,

shift insert in Git Bash.

Yes.

Okay, pseudo -I.

Let's first check the service status,

system CTL status, MariaDB.

And it says active runnings or user data script worked.

Make clear the screen.

I'll also show you how you can verify your user data script.

So just Google retrieve user data EC2.

So you'll get this document work with instance user data.

I gave wrong spelling of retrieve.

Okay, retrieve instance user data from the documentation.

So, in this URL, okay, copy this URL,

go to your instance and all you have to do is say curl

and paste that URL.

So on every instance at this IP address,

you can access the user data script.

See that?

So if you think you made some mistake in user data script,

you can check it later also.

Okay, also then MySQL,

login into database and verify MySQL -u admin.

We had created the user admin -padmin123,

the password and the database name accounts.

Hit enter.

Should be able to log in.

Show tables and you should see the tables

created in the accounts database.

This is good, it's working fine.

Let's log out and let's log into the other instance,

which is Memcached.

Let's get the Memcached public IP.

Select MC01, copy it's public IP and paste right there.

Yes.

Sudo -i.

Again, if you want to verify the user data script,

we can do it with that URL.

I just want to check the service.

Okay, I'll show another way of checking, right?

So when we start any service, network service,

it runs on a port number, Memcached 11211, MySQL 3306.

So run the command ss -tunlp,

pipe grep 11211.

Not greo, grep.

Yeah, okay.

11211, that's the Memcached port number,

default port number.

So if the service is running,

there will be a Memcached process

and that process would have opened this port number.

See that?

11211, and you should see here 0.0.0.0.

That means listening from all the IP from outside also.

So one command you're verifying the process is running,

it's listening, and it's listening

on all the IP addresses, right?

Okay, let's log out from this and go to RabbitMQ.

Copy the RMQ zero and public IP.

And this one I leave up to you

how you are going to verify this one.

You want to run system CTL, you want to check the process,

your choice, right?

I'm just going to run system CTL status.

All right.

You should know the port number, right, to verify?

So if you don't know the port number of any service,

just simply Google or use Chat GPT if you want to.

Check the default port.

RabbitMQ-server.

Okay, it's active, it's running well and good.

Let's log out, we are all done here.

Okay, last instance, tomcat.

App01, let's get the public IP.

Shift insert.

Yes.

Oh yes.

Okay, that's Ubuntu instance, right?

And the username is definitely not easy (indistinct) user.

The username is Ubuntu.

You can check by clicking on the connect button,

but I know it's definitely Ubuntu.

I used this AMI many times.

See that?

It will log in.

Okay, as I told you, tomcat package name was tomcat nine

and the service name is also tomcat nine.

System CTL, status, tomcat nine.

Okay?

That's the service name.

Also, one more thing,

if you remember when we did a local setup on the VM,

the home directory was USR local tomcat.

But when we installed through package manager,

the home directory of tomcat will be var/lib/tomcat9.

Here, you should have the web apps folder.

And in the web apps folder you should have route

that is a default tomcat application.

Okay.

Now, in Vagrant we used host manager plugin

to automatically add ETC host rules.

No, we're definitely not going to do ETC host file here.

ETC host file is local.

Only the machine knows the name to IP mapping.

We are going to do better.

We are going to use route 53 DNS servers.

Let me go directly there and talk about it.

So search for a route 53 in AWS console.

AWS route 53, very, very popular DNS servers

and very easy to configure.

We need to create the zone, which will be your domain name.

And then in that domain you will have different hosts

and that host records will have the IP or the c name.

I'll show you.

Click on create hosted zone.

We will give our zone name as vprofile.n,

and you can give the same

because we are going to use private hosted zone.

This is not going to be public,

it is not going to be resolved from the internet.

It's going to be in the same VPC of the region,

AWS region where we have instance only there,

this will be resolved, not outside.

You need to select the region.

So I'm using us-east-1.

I think you are also using the same one, us-east-1.

And VPC.

Just have one VPC in that.

VPC is the network in that region.

So only in this network you would be able

to resolve the DNS host names.

Again, in a moment you will see what I mean.

Create hosted zone and create record.

Click on create record.

Simple routing.

That means from the name, the IP will be resolved,

no extra algorithms, no other things.

The many fancy and amazing algorithms,

routing rules that you can set

but not going into that now.

Simple routing.

Go next.

Define simple record.

Okay, so this is how it is going to be.

So starting with db01.

This is the host name that we are giving.

So db01.vprofile.inshoot, resolve to the IP address

of our DB instance.

So I'm going to open AWS console in another tab,

log back in, make sure you're in the right region.

EC2.

Okay, so db01, private IP, not the public IP,

private IP address.

Once again, private IP, copy that.

And we'll paste it here.

Okay, so this name will be resolved with this IP address

and it is not local like ETC host file, right?

Not in the OS itself.

It will be resolved in this entire VPC.

Define simple records.

Okay.

Create records.

So many times you have to click now.

Okay, another record.

Simple routing.

Next.

Define simple record.

Next instance, mc01 and the IP address,

get the IP address of mc0

and make sure you're copying the right IP.

Most of the time it's always copy paste mistake.

Private IP.

Paste it.

MC01.vprofile.in will be resolving to this IP address.

Create records.

Okay, one more.

Simple routing.

Go next.

Define simple routing.

RMQ01 and get the IP address of rapid MQ instance.

Once again, private IP.

Now, you think I'm stressing too much on this,

but many people make the mistake of using public

to private and private to public.

It's a very common mistake.

Create records.

Okay, so we only need these three

because app server is going to connect

to these backend services, in application.properties file.

Remember, in vprofile project,

application.properties file?

There we have the backend server information database,

Memcached and RabbitMQ.

Only for that, we are creating this record.

So that is all in this lecture.

Next lecture, we will build the artifact in our laptop,

upload to S3 bucket and deploy to the app server.

See you in the next lecture.

Summary and Key Points:

1. **Security Group Outbound Rules**:
   - Ensure outbound rules in your security groups allow all traffic to go anywhere.
   - Verify inbound rules to match the specific needs of your application.
2. **Clone the Latest Source Code**:
   - Remove old source code and clone the latest from the repository (`github.com/hkhcoder/vprofile-project`) using the `aws-LiftAndShift` branch.
   - Use the `tomcat_ubuntu.sh` script for installing Tomcat on Ubuntu instances.
3. **Building and Deploying Outside Instances**:
   - Build artifacts locally on your laptop, push them to an S3 bucket, and then fetch them into instances.
   - Understand the significance of building outside the instance, which will be detailed in later CICD projects.
4. **Firewall Setup**:
   - Ensure that firewall rules are set up both at the OS level (using `firewalld` for Red Hat-based OS) and the security group level.
   - Add necessary ports and reload the firewall service.
5. **Launching EC2 Instances**:
   - **Database Instance (MySQL)**:
     - Use `Amazon Linux 2023 AMI`.
     - Instance type: `t2.micro`.
     - Security group: `backend`.
     - Copy and paste `MySQL.sh` script in user data.
   - **Memcached Instance**:
     - Use `Amazon Linux 2023 AMI`.
     - Instance type: `t2.micro`.
     - Security group: `backend`.
     - Copy and paste `Memcached.sh` script in user data.
   - **RabbitMQ Instance**:
     - Use `Amazon Linux 2023 AMI`.
     - Instance type: `t2.micro`.
     - Security group: `backend`.
     - Copy and paste `RabbitMQ.sh` script in user data.
   - **Application Instance (Tomcat on Ubuntu)**:
     - Use `Ubuntu Server 22 HVM SSD Volume`.
     - Instance type: `t2.micro`.
     - Security group: `app`.
     - Copy and paste `tomcat_ubuntu.sh` script in user data.
6. **Instance Verification**:
   - Log into each instance using SSH and verify the services are running.
   - Use `systemctl` commands to check the status of services.
   - Verify ports with the `ss -tunlp` command.
7. **Setting Up Route 53 DNS**:

   - Create a private hosted zone in Route 53 for the domain `` `vprofile.in` ``.
   - Add records for `` `db01` ``, `` `mc01` ``, and `` `rmq01` `` with their respective private IP addresses.
   - This allows resolving names within the VPC.
8. **Next Steps**:
   - In the next lecture, you will build artifacts locally, upload them to an S3 bucket, and deploy them to the app server.

Steps to Follow:

1. **Verify Security Groups**:
   - Ensure outbound rules allow all traffic.
   - Double-check inbound rules for correctness.
2. **Clone the Latest Code**:
   - Remove old code from your local setup.
   - Clone the latest code from the specified GitHub repository and branch.
3. **Launch EC2 Instances**:
   - Follow the specific instructions for each type of instance (DB, Memcached, RabbitMQ, and App).
   - Use the provided scripts in user data during instance launch.
4. **Check Service Status**:
   - SSH into each instance and verify the service is running.
   - Use appropriate commands (`` `systemctl` ``, `` `ss -tunlp` ``) to ensure services are up and running.
5. **Configure Route 53**:
   - Create a private hosted zone for `` `vprofile.in` ``.
   - Add DNS records for each backend service with their private IP addresses.

By following these steps, you will ensure that all services are properly set up and configured, ready for the next stage of building and deploying artifacts.

Udemy
DevOps Beginners to Advanced with Projects
Lecture thumbnail
0:01 / 15:46
Transcript
Now it's time to build our artifact.

Upload it to S3 Bucket, and from there, fetch to the EC2 instance to the tomcat EC2 instance.

So we need to do a few things on our local machine and a few things on on AWS.

We need to create an S3 bucket and an Iam role.

And an Iam user.

We'll get to before that.

Let's open git bash from vs code.

Mac OS user you can same way open your terminal.

So hit Ctrl shift P.

Mac os will be using command shift P.

Control shift P and search for select default profile.

Okay, type that.

So select default profile and click on it.

And from here, select Git bash.

Now Gitbash becomes the default terminal.

That means here we can execute the commands.

Now in The View.

Select terminal.

And you should see here.

Git bash terminal over here.

Now, before that, we need to make change to the application properties file.

So get into src folder main.

Resources.

Application.properties file.

Okay.

This one.

Make sure the path is correct.

Double click on that.

Let me close this terminal for now.

So.

Database, right?

Zero one But we have the complete name as profile dot in which will be resolved to the IP address from

Route 53.

User username and password did not change in user data script, so that remains same.

Here also zero one and RMK zero one also.

Okay.

Make sure you save this.

Ctrl s or go here and click on Save.

And now you can build the artifact.

So view.

Terminal.

Nats git bash terminal do you should be in the same folder here.

You should see pom.xml.

And here you have to run mvn install now before we run npm install.

I hope you had already installed maven and JDK in prerequisites.

So mvn hyphen version run that command.

Check.

It should be Maven three and the Java version should be 11.

In the Prerequisites lecture we had installed Maven and Java first we installed.

Java.

And then we installed Maven.

Also we installed AWS CLI, so you should have AWS CLI.

Also if you just type and hit enter.

You should get this output helper like that.

So make sure CLI Maven and JDK are all installed and then make sure you are in this particular folder

and say mvn install to build the artifact.

It is going to take some time.

So I'm going to pause my recording and come back.

Okay.

My build is success.

If I do ls I should see the target folder or I can see it from here.

Also target and this profile v2 dot var.

Now this artifact contains the application properties file and this is the archive of this folder.

So if you expand this v profile hyphen v2, go to web hyphen inf classes application.properties.

You should see db0 one v profile dot n and same for C01 and rmq zero one.

And let me close this one.

This was just to show.

Just just to show you.

Let me close this.

Okay, so we have the artifact.

Now it's time to push this artifact to S3 bucket.

And it is not possible without the authentication.

So we are going to create an Iam user and also S3 bucket.

Okay, let's go to Iam service.

And go to users, click on add users.

I will give this name as S3 admin.

And we don't need any console access for this one.

We need the access key and secret key.

So go next, attach policies directly, select that and search for S3.

Full access, Amazon S3, Full access.

So we are going to use this user's access key to access the S3 bucket.

So we need S3 full access because we're going to copy data to it.

Next and just create user.

Okay, After that, click on the user and go to security credentials.

Come down and say create access keys.

Command line interface.

I understand the risk.

Next.

Create access key.

Now, in no way or no case, you should expose your access key and secret key to anybody.

You should not show it to anybody.

You should definitely not put it on GitHub repository, especially the public repositories.

It's way too dangerous.

There are Autobots.

Sorry, I had to say the bots.

Not the Autobots.

So there are bots on the internet which scans for such kind of data.

And they use it.

They will use these keys to access your account and will be doing Bitcoin mining.

Or if it's a bigger company account then.

There will be also ransom.

And the ransom also needs to be paid with Bitcoin.

So I don't know who to blame Bitcoin or access key.

But anyways, be careful you will be seeing my access key, but as soon as the recording is completed

I'm going to delete those keys so you will not be able to use it.

So download CSV file.

And to download it which will have the access key and secret key.

Done.

Now this user has just S3 full access.

But if it has administrator access and these keys are exposed then.

You'll get huge amount of bill in few hours.

Okay, So I'm going to.

Store this access key or configure my key with this access key.

So I'm going to say AWS configure.

It's going to ask the access key and the secret key.

So I got my access key from the CSV file.

And the secret key.

The second one comma separated.

That's the secret key.

I told you I had to show you, but I'm going to delete as soon as the recording is completed.

Region us East one.

So that's the default region for me and that's the region I am using.

If you don't see your region name here, just type that like I'm going to type it once again.

Us hyphen east hyphen one output format Json.

Now, I could have done the same thing from the VS code terminal here also.

Okay.

Now I am going to create S3 bucket from command line.

So AWS S3, MDB.

S3, Colon slash, slash and the bucket name.

The bucket name needs to be unique.

You cannot give the same what am giving because when I give, I create that bucket, I get that name.

You, you lose because you are late.

Okay.

Just kidding.

So.

S3 bucket gives you a endpoint that can be resolved on the Internet.

So definitely you cannot have duplicate endpoints, right?

You don't never have duplicate URLs.

So that's why the bucket name needs to be unique.

So you get your own unique URL to access the S3 bucket.

So I'm going to name it as coder.

And I will give I will say hyphen, Wipro hyphen, artifact Arts, I'll say.

Okay.

My bucket is created.

Now if you get an error bucket name already exists, then add some numbers to make it unique.

Let's copy our artifact.

So from the same folder.

AWS S3 CP target folder we have profile v2 dot var and we're going to copy it to the S3 bucket.

So S3, Colon slash, slash and the bucket name.

Bucket Name.

Slash.

So it's copying the artifact to the S3 bucket.

You can go and check from the AWS console.

Also, you should have an S3 bucket and once the copy is completed, you should see your artifact there.

Okay.

Upload completed.

Okay.

In the S3, you see my S3 bucket and you should see the artifact.

Now the final step is to download this artifact into our EC2 instance Tomcat.

EC2 instance.

Right, that one.

So we will SSH to this instance, install aws cli and fetch our artifact from S3 bucket.

But how will it authenticate to the S3 bucket?

Like we from our laptop we used aws CLI and used Iam access keys.

Those access keys gave us gave us authentication to the account.

So we are able to push the artifact to the S3 bucket.

We can do same.

In this instance, we can install and set the access key, but there is a better way.

And that is true Iam roles.

We can create a role like we create a user, we can create a role.

Can go to Im and click on roles.

We create a role and we can attach this role to our instance.

So.

I'm sorry.

So come to Rolls and click on Create Role.

So let's keep service and select EC2 here.

Common use case because we are going to attach it to our EC2 instance and go next.

Search for the policy like we did.

Same for the user.

Iam user.

Same for the role we need to attach policy.

So.

S3 full access I'm going to give.

And click on next.

Give role a name, So I'll say Shkoder.

Hyphen v profile hyphen S3.

And come down and say create role.

Okay, the role is created.

Let's go to our instance.

Make sure you refresh it.

Select your ab0 an instance.

Action.

Security modify Iam role?

And select your role code or profile S3 There, that's mine.

And you can see update rule.

So instead of access keys, we are giving roles to the instance.

Select on that and let's do SSH.

Make sure your instance security group 22 allowed from my IP is there.

If you are doing it on a second day, your IP will change.

So update your my IP.

Okay.

Ssh hyphen.

I login key is in downloads.

And the username is ec2 hyphen user.

No, it's ubuntu.

Sorry.

All right, it's ubuntu instance.

Ubuntu is the user.

Okay, let's be the root user.

Clear the screen.

Let's run apt update.

And then install aws cli.

So apt.

Install aws cli hyphen VI.

Okay, I get this message.

New kernel available.

I'll just hit space bar to see.

Okay.

And use tab and hit spacebar once again.

Okay, CLI is installed now you can test it, you can see AWS S3 if your Iam role, which is attached to this instance, has S3 privilege, you should be able to list all your S3 buckets so you can copy

AWS, S3, CP.

S3 Colon slash slash your bucket name slash profile hyphen v two dot var not C2V2 dot.

There and I'm going to copy it to the temp folder.

Okay.

That was faster, right?

Because S3 bucket and the instance is in the same region.

So the speed is very fast.

Okay, it's time to deploy this.

We have to stop the Tomcat nine service.

And we have to remove the default application.

Var lib Tomcat nine web apps, root capital r o t remove that and we'll copy our artifact from temp.

We profile v2 dot var to var lib.

Tomcat nine web apps with the name root dot var.

Make sure you give root dot var at the end.

Otherwise you will see a you won't see any page because you removed the root application.

Also default application.

And if you copy it with a different name, you will not see anything in the Tomcat web page.

So copy it with this name and let's start the Tomcat nine service.

And now this should extract.

Our artifact.

With the name root there extracted.

So we are done.

Just one validation.

So cat var lib tomcat nine.

Web apps.

Route.

And in that we have Web-inf classes application dot properties file.

Now this file is the same file we edited in our local machine.

This should be in the artifact.

Right?

Right.

See that?

Everything is good.

Okay, so that is all.

In this lecture, we have built the artifact locally, upload it to S3 Bucket and from the S3 bucket

we have fetched the artifact to our Tomcat instance.

Next lecture we will see load balancer how to configure load balancer for this Tomcat instance and also

access it from Https secure endpoint.

This lecture outlines the process of building, uploading, and deploying an artifact using AWS services, specifically S3 and EC2. Here is a step-by-step summary of the actions taken:

1. **Preparation on Local Machine:**
    - Ensure Maven, JDK, and AWS CLI are installed.
    - Open Git Bash in VS Code.
    - Navigate to the `` `src/main/resources/application.properties` `` file and update it with the necessary information.
    - Build the artifact using `` `mvn install` ``.
2. **AWS Setup:**
    - Create an S3 bucket and an IAM user with S3 full access.
    - Configure AWS CLI with the IAM user's access key and secret key.
3. **Uploading Artifact to S3:**
    - Use `` `AWS S3 CP` `` command to copy the artifact to the S3 bucket.
4. **Setting Up EC2 Instance:**
    - SSH into the EC2 instance.
    - Install AWS CLI on the EC2 instance.
    - Create and attach an IAM role with S3 full access to the EC2 instance.
5. **Downloading and Deploying Artifact from S3:**
    - Use the `` `AWS S3 CP` `` command to copy the artifact from the S3 bucket to the EC2 instance.
    - Stop Tomcat service, remove default application, and deploy the new artifact as `` `root.war` ``.
    - Start Tomcat service and validate the deployment by checking the `` `application.properties` `` file in the deployed artifact.
6. **Next Steps:**
    - The next lecture will cover configuring a load balancer and securing the endpoint with HTTPS.

This process involves multiple AWS services and commands, ensuring the artifact is correctly built, securely uploaded, and deployed efficiently on an EC2 instance.

Udemy
DevOps Beginners to Advanced with Projects
Lecture thumbnail
0:08 / 13:26
Transcript
Instructor: Okay, our application server

is also up and running.

Backend services are up and running.

Now it's time to set up a load balancer.

So let's go to EC2, and you have to scroll down in the load balancing section.

Come to the target groups,

and we will first create a target group.

Target group, which is basically the group of instance

that you want the traffic to be routed on.

So load balancer will route the traffic to the target group.

So we say create target group

and we have instances.

Come down, let's give this target group a name.

We will say vprofile-app-TG and port.

Now this is the port number

on which the instance is running the service.

So we are running Tomcat service on the instance,

and it listens to port 8080 by default.

So make sure you change it to 8080.

Coming down health check.

Now target group is going to perform the health check

of the instance, which is basically

just access the instance.

So our application runs slash login.

So that means the IP of the instance slash login.

That's where the view profile application listens on.

So you need to do slash login advance health check.

Now here the protocol is http

and the default port for http is 80,

but we are running Tomcat on that.

It is using the http protocol, but the port number is 8080.

So you have to override the port here

and you have to say 8080.

Make sure you confirm these things.

8080 in the override slash login here.

And this is the port number again, 8080,

on which the load balancer will route the traffic

to the instance.

Make sure they don't change.

Its 8080 and slash login.

Now, healthy threshold, you can reduce this.

So check how many times it's like, we'll say two times.

If it's healthy, then declare the instances.

Unhealthy, sorry, healthy.

For two times it's unhealthy,

then declare the instances unhealthy.

So coming down, let's say next.

Now you need to select the instance.

So we have vprofile-app0,

and this is the instance we need to add in the target group,

so put a check mark on that.

Make sure it's the correct instance.

Make sure the port number shows here 8080

and click on include as spending below.

Make sure you click this one.

Then you will see your instance here,

and then you are good to create the target group.

Okay, the target group is created.

Now we will create a load balancer,

which is going to route the traffic to this target group.

So go to load balancers

and click on create load balancer.

Now we need to use application load balancer.

We have http and https traffic,

and for that the best load balancer

is application load balancer.

Now you have option of network load balancer here as well.

This is for a very high volume of traffic

and it is expensive.

We have classic load balancer also,

which is the favorite of many, including mine

because it's easy to set up and easy to maintain.

But we'll stick to application load balancer,

and we'll go, oh sorry, yeah.

Here we have to click on create.

Let's give load balancer a name.

Let's name it as vprofile-prod-elb.

You can give any name of your choice,

just make sure you're able to recognize it

with the name later.

In the network mapping, you have to select the zone where you want the load balancer to route traffic to.

So we'll select all these zones over here,

and you have to make sure that the instance zone

is definitely selected where your instance is living.

So you can go to the instance,

select the instance, and check the zone.

At least that zone should be selected.

And minimum two zones you have to select.

So anyways, we have selected all the zones.

Security group.

We have to select the ELB security group.

This is the security group we created for the load balancer.

Make sure you select that.

Make sure it's selected.

Now, listeners and routing.

So on which port it should listen.

We saying listen on port 80

and route the traffic to our target group.

We also want to configure https connection.

So we are going to say add listener.

In the protocol, select https.

By default, it's going to take the port 443

and route the traffic to the target group.

So now we basically have two listeners,

one on port 80 and one on port 443.

Now this one, https 443, will work

only if you created the certificate.

This we did in the prerequisite lecture

in the AWS setup lecture.

Prerequisite section AWS setup lecture.

If you have not created the certificate,

if you have not purchased the domain,

you can skip this part.

You can remove this one because that is mandatory.

You need to have the certificate.

The certificate we are going to select over here

from ACM, select the certificate.

Once again, if you have not created the certificate,

if you have not purchased the domain,

then you need to remove this one.

Just keep http port 80.

Come down and say create load balancer.

Click on view load balancer.

So it is going to take some time.

This is in the provisioning state.

It'll take some time to become available.

By the time, we are going to go to our domain provider.

For me it's GoDaddy.

We, I have purchased domain in GoDaddy.

We did this in the perquisites lecture.

Again, if you have not purchased the domain, just watch.

But if you have the domain, copy this DNS name over here

and go to your domain provider.

As I said, mine is GoDaddy.

So I'm going to go to my domain.

You have to go into my products section.

This UI may change a little bit,

but log into your GoDaddy account.

Go to my product, find your domain, click on your domain.

Okay, here you have quick links, manage domain.

You can click over here or you can go to the domain,

and on just near your domain name

you should see here setting manage DNS.

You need to go into manage DNS.

So UI may change.

Basically you need to get into manage DNS section.

Okay, now you have to add a new record.

Now we want our website to be properly accessible

with a proper url.

So domain is, my domain is hkhinfo.xyz,

and I'm going to hear select type cname.

CNAME means name to name mapping.

So we are going to give a name to our website,

and okay, yeah, so I'm going to say vprofile app.

It's your choice.

You can give anything.

In the value, you need to paste

the load balancer endpoint, okay?

No http or anything over here.

Just the endpoint, starting with the name

of the load balancer ending with .com.

No full stop, no nothing.

Okay, cname, right, and click on save.

Okay, now it may, it'll show success,

but it'll take time to publish this url on the internet.

Like 10 minutes, 15 minutes, half an hour like that.

Depends on where you now reside in the world.

If you are in U.S., it will be very quick.

Asia Pacific, it's going to take some time.

So wait some time.

By the time we can also check

with the load balancer endpoint.

So we have the load balancer endpoint.

Copy that and we are going to put it

in the browser and hit enter.

Okay, so when you do it with the load balancer, ul,

by default it goes with http traffic.

You can also do it with https, but this is not going to be

with the certificate.

As I told you, we are waiting.

So by the time, we are also checking with

https colon slash the url,

and it will show you your connection is not private.

It's able to access, but it is saying

the certificate is different.

Here, the domain is amazonaws.com.

We can still go advance, and we can say proceed.

So it's showing the page, the login page,

but https, you see it's showing that cross mark.

This will be gone when we exit with the proper url

that we have updated in GoDaddy.

But for some reason, if you are not able to see this,

you're getting some errors,

like 503 gateway timeout, these kinds of errors.

Then you make sure you check your target group.

Select your target group.

Go to targets and make sure the instance is healthy.

If it is showing unhealthy, there's a huge chance

you have given the wrong security group rule

for your application security group.

The other chance is that you have given wrong port number

or your configured wrong health check.

Make sure you check the health checks slash login

on port 8080

and the target is routing the request on port 8080.

Also go to the application security group.

Port 8080 should be allowed

from load balancer security group.

You can also enable 8080 from anywhere

and wait five, 10 minutes and then check again.

Also, you need to see whether your instance

is healthy or not.

So go to your instance.

First of all, you need to go to the security group

of the instance, application security group.

Edit inbound rule.

Once again, this is application security group, okay?

You have to say 8080, add one rule 8080.

Allow from my IP, I already did this.

You see this rule that we have added, right?

8080 allowed from the load balancer security group.

Also add one more rule, 8080 allowed from my IP.

See if this, go back to your instance,

get your instance a public IP, put it in the browser.

Colon 8080.

This is to validate directly whether your instance

is serving the webpage or not, right?

If this is coming and from load balance

you're not able to see, then huge chance security group

or the target group port number you have made some mistake.

But if from the instance IP also it is not coming

on port 8080, that means your Tomcat service itself

is not running.

So you need to set up the instance once again.

Okay, by the time we're talking

I think the url should be published.

So vprofile app, right?

So let's open the browser.

Http colon slash slash vprofile app.hkhinfo.xyz,

your domain name.

Okay, mine is hkhinfo.xyz.

So whatever name you have given, just try to access it.

See, this works, this simply translates this name

to the load balancer endpoint.

Okay, we can also check through https.

Wait, it's already https here.

Okay, so it's https, and I don't get any error.

That means my certificate is good, able to access.

Now let's login.

Username, admin_vp,

and the password is also the same, admin_vp.

Okay, I'm able to log in.

If you're getting any error over here,

check the backend security group.

You need to allow 3306

from the application security group.

Or you can say 3306 allowed from anywhere,

just to make it work.

Okay, so this validates that application

is able to connect to the database.

If you not able to log in, then you need to check

in the database side, database security group,

the database has created the tables or not.

Let's check RabbitMQ, click on RabbitMQ.

RabbitMQ initiated.

This is good.

It's able to connect to RabbitMQ.

Now let's go and check mem Cache.

So click on all users.

So this is going to access the all user index

from the database, and I'm going to click on any user id.

Click on any user id.

You should see a message here.

Data is from DB and data is inserted in Cache.

So mem Cache is also working.

Go back and if you click on the same instance,

sorry, same user ID once again,

then it should load faster

and you should see another message,

data is from Cache.

So that is also working.

So we have validated our application.

Click on this visual part, vprofile over here.

So overall our application is working,

but we need to go to the next level.

We need to set up auto-scaling group

for our app zero one instance, our Tomcat instance.

We are going to set up auto-scaling group,

that we'll see in the next lecture.

So join me in the next lecture.

This lecture from the Udemy course "DevOps Beginners to Advanced with Projects" walks you through the setup of an application load balancer (ALB) on AWS, and then goes on to configure the necessary DNS settings for routing traffic to your application. Here's a summary of the key steps covered:

1. **Creating a Target Group**:
    - Navigate to EC2 and the load balancing section.
    - Create a target group named `` `vprofile-app-TG` ``.
    - Set the port to 8080 (default port for Tomcat).
    - Configure health checks on the path `` `/login` `` using HTTP and port 8080.
    - Add your application instance (`` `vprofile-app0` ``) to the target group.
2. **Creating an Application Load Balancer**:
    - Choose Application Load Balancer (ALB) for HTTP and HTTPS traffic.
    - Name the ALB (`` `vprofile-prod-elb` ``) and select appropriate availability zones.
    - Attach the correct security group to the ALB.
    - Configure listeners for HTTP (port 80) and HTTPS (port 443).
    - For HTTPS, select the SSL certificate created in the prerequisites.
3. **Configuring DNS Settings**:
    - Use your domain provider (e.g., GoDaddy) to add a CNAME record.
    - The CNAME should map your custom subdomain (e.g., `` `vprofileapp` ``) to the ALB endpoint.
4. **Validation**:
    - Test the load balancer endpoint directly to ensure traffic routing works.
    - Validate the HTTPS connection to check SSL certificate functionality.
    - Ensure your application instance is healthy by checking the target group and security groups.
5. **Troubleshooting**:
    - If facing errors, check the security group rules and health checks.
    - Verify the application instance's public IP and Tomcat service status.
    - If still encountering issues, revisit the DNS settings and ensure proper propagation.
6. **Final Validation**:
    - Access your application via the custom URL to ensure it resolves correctly.
    - Test logging into the application and interacting with various services (e.g., RabbitMQ, memcached).

The lecture concludes with an introduction to setting up an auto-scaling group, which will be covered in the next session.

Instructor: Okay, so the last thing that we need

to do is create the auto-scaling group

for our Tomcat instance.

Right, app01.

We wanted to automatically scale out

and scale in based on the load.

We need total three things

to set up the complete auto-scaling group.

One is the AMI of this instance.

So, we need to create the AMI of this instance.

Second is the launch template.

The information about how to launch the instance.

That is launch template.

Third is the auto-scaling group itself.

Based on the launch template or information provided

from the launch template, it is going to launch the instance

from the AMI, which is specified in the launch template.

So once again, three thing.

AMI, launch template, auto-scaling group.

First thing we need to do is we need to create the AMI

of this instance.

So, let's say Action, Instance state...

Oh, sorry.

Image and Create image.

I'll give it a name.

vprofile-app-image,

and you can give some description,

and I will say Create image.

This is going to take some time.

The AMI is getting created, it is in the Pending status,

so this is going to take some time.

So, wait until the status of the AMI becomes available.

Once it is available to use,

we will create a launch template

and then we will move to the auto-scaling group.

Okay, so after waiting for five minutes or so,

the AMI is available now.

Now we will create launch template

which will be used in the auto-scaling group.

So, let's go to Launch Templates.

Okay, so here, we are at Launch Templates.

Now auto-scaling group is going to create

or delete instances, or maintain the number

of instances based on the information that we provide to it,

which we will do next.

But if it's launching the instance,

how will it know which AMI to use,

which security group to use, which key pair to use?

All that information is provided by the launch template.

So in the auto-scaling group, we specify,

this is the launch template you need to use

when you want to launch an instance.

So, let's click on Create launch template.

Let's give it a name.

vprofile-app-LC.

Description. I'm going to give the same thing.

Okay, Application and OS Images.

AMI, basically, so we created already the AMI,

so we're going to say My AMI,

and you should be able to see your AMI over here.

If you have multiple AMIs,

make sure you select the right one that we just created

and come down.

It is almost similar to launching an instance

because it's the same thing.

Launch template will be used to launch the instance.

So instance type, we will keep it as a t2.micro, free tier.

Key pair.

The same key pair that we used to launch the instance,

vprofile-prod-key.

Security group. Select an existing security group.

vprofile app security group.

Coming down.

Resource tags.

Let's give tag, Name.

Oh, I need to do proper case.

Name.

Value. I'm going to put vprofile-app.

So here, we don't give app01, 02,

because auto-scaling group is going to maintain the number

of instance.

So here, the number of the instance does not matter, really.

So give this tag to the Instances,

and also, I want to give it to the Volumes.

So, the volumes of this instance

will also should have this tag,

also, let's give it to the Network interfaces.

So it's easy to identify when you have tags.

We can give one more tag over here.

I will say Project.

vprofile.

Same for the volumes, network interface.

Okay, come in the Advanced details. Let's come down.

Now we have created an IAM role

that we attach to our app01 instance.

When this instance gets launched,

we want these instances also to have that IAM role.

So I'll say instance IAM profile

and find your role, s3 role.

Select that,

and that is all we need.

And we can just say Create launch template.

Okay, so the launch template is created,

and we're going to specify this launch template

in the auto-scaling group.

Now let's go to the auto-scaling group.

Click on Create Auto Scaling group.

Let's give auto scaling group a name.

vprofile-app-ASG,

and let's select the launch template.

Scroll down and go to next.

Okay, so let's keep the VPC, the default VPC,

and let's select all the zones that we have in this VPC.

I'm going to select on all the zones,

so that means you can launch the instance in any of the zone

or you can select specific zones also.

And then going next,

so load balancer, we have a load balancer.

We want our instance to be in the load balancer

after they are launched.

So we can select here,

"Attach to an existing load balancer."

So we basically select here the target group,

not the load balancer.

So once the instance is launched by the auto-scaling group, it is going to add it under this target group.

So, let's select our target group.

Coming down to the health check,

so auto-scaling group, we'll do a basic health check,

which is the instance health check two by two,

whether the instance is running or not, that's all.

It is not going to check

whether the service in the instance is running or not.

So for us, there is the Tomcat service.

If you want to check the status

if the Tomcat service is also running or not,

there is no direct method over here,

but we have this method

which is looking at the load balancer health check.

Now our load balancer will do the health check

of the instance on port 8080 at /loginurl.

So basically, it's checking the Tomcat service is running

or not.

So based on that health check,

auto-scaling group can decide

whether this instance is healthy or unhealthy.

So, that's what this check mark means.

"Turn on Elastic Load Balancing health checks."

And it's also recommended.

Let's come down. Let's go to Next.

Now here, we need to decide the capacity.

How many instance you want for the sake of cost.

Keep one as the desired instance

and minimum instance as one.

Maximum capacity, you can increase

just for the sake of learning.

Anyways, the auto-scaling group is not going

to launch more instances

because our instance will not have any load.

So, this is to maintain the capacity.

If we want to scale out or scale in based on the policies,

you need to go for Scaling policies.

So by default, it's none.

We are going to say tracking scaling policy.

So, set the scaling policy.

And you have different metrics over here,

CPU utilization is the most common one,

and network out for web application.

These are two very popular metrics for the web application,

http, https-based.

Any application you have,

CPU utilization and network out are the most used one.

We'll stick to CPU utilization

that if CPU utilization goes more than 50 overall.

So in one single instance,

if the one single instance CPU utilization goes more than 50 then it is going to decide to launch extra instance,

and maximum, it can launch four instance.

That's what we have specified.

Okay, that simple it is.

Sure, there are many complicated policies also

that you can set

with customized policy, but we are not getting into that.

There is a scale-in protection option

that if you want the instance to not get terminated

if it's unhealthy, then you can put this check mark on.

We don't want this.

This is usually used, you know,

auto-scaling group will terminate the instance

if it's unhealthy and launch a replacement for it.

But you might need to troubleshoot

and log into the instance,

and see what was the actual problem why it went unhealthy.

In such cases, you need to enable scale-in protection.

So, auto-scaling group will not terminate

the unhealthy instance,

but it won't divert any traffic to it.

It will launch a replacement,

but it won't terminate the unhealthy instance.

That's what this check means. Let's not put this check mark.

And let's go Next.

Notification.

Now if you follow the initial AWS setup lectures,

you must have created a billing alarm.

So, you can use that SNS Topic over here.

If you don't see anything in this dropdown,

that means you did not create the billing alarm

and you don't have any topic to send email to.

So, auto-scaling group can send these four events.

Launch, Terminate, Fail to launch, Fail to terminate

to the topic that you have provided.

I'm going to keep that,

but if you don't have this, it's okay.

You can remove this and go Next.

Tag. Let's add some tag.

Name.

Let's say vprofile-app.

And I will say Project,

name as vprofile,

and we'll go Next.

So, carefully review all the settings

that you have selected.

If you are good with all the settings that you selected,

you can say Create Auto Scaling group

and wait for almost 10 minutes or so

for it to launch the instance, add in the target groups.

It's going to take some time.

So, wait until all those things are completed

and then we'll take a look what actually happened.

-: Okay, so I was waiting for like more than 10 minutes,

and in this time, auto-scaling group has launched instance.

It has launched two instances, the desired capacity changed

to two and then it deleted the instance.

So, it was figuring out how to maintain the capacity,

and I was just waiting and watching

until it comes to one single instance.

So you also need to do the same thing, wait for some time,

and in this time, it is going to launch instance,

terminate instance, and it will take 10 to 15 minutes

to become completely stable.

Now in the Instances section, if you go,

and let's search with app.

Okay, vprofile-hyphen-app.

So we have vprofile-app01,

this instance we created earlier to create the AMI.

We don't need this one, so you should terminate this one.

So you can say Instance state, Terminate instance.

Terminate.

And this is also going to take some time.

The instance will get terminated.

It's going to remove from the target group,

so this process takes some time.

So we have vprofile-app, this is in Running state.

We have one more vprofile-app,

but you see it's in the Terminated state.

Now if you go to the Target Groups.

In the Targets, you can see instances.

Now app01, this one will become unhealthy,

and we can remove it later from the target group.

And this is the instance that was created

by the auto-scaling group.

Now if you have multiple instances in your target group,

they are healthy,

you don't want to terminate the other instances.

So you need to do one thing

before you check the vprofile application from browser,

before that you need to enable one setting.

This is specifically for vprofile application.

So, click on your target group, vprofile-app-tg,

that's the name of my target group,

and come to the Attributes section.

Okay, once again. Target Group, Attributes, click on Edit.

Scroll down, and you have to enable this setting Stickiness.

Now stickiness means if the user is forwarded

to an instance, let's say we name that instance as app01,

and we have one more instance, app02.

Now the user is routed to app01.

That user will always get routed to app01 for a period of, we can specify the time here, one day.

Right, if there is no stickiness,

when the user refresh or try to log in,

the load balancer might route the request

to app02 then app01, so it may...

Not it may. It will do that if there is no stickiness.

By default, the algorithm is round-robin.

So first request goes to first instance,

the second goes to the second,

and third goes to the first instance again,

the fourth goes to the second instance.

So, it's round-robin by default.

But that will be problem for vprofile application-specific.

Okay? This is only for vprofile application.

I'm not saying you need to do this for every application

that you see in real time.

Okay, so enable Stickiness and Save changes.

You can read more about stickiness online.

Now there is a complete process

how it stores a cookie in the user's browser.

And based on that cookie, it is routed to the instance,

so on and so forth.

But that will be out of scope of this course,

so I'm sticking to only this much.

Okay, so come back to target groups while we are doing this,

if you go to the target group targets.

Okay, we just have only one single instance now.

The old instance is gone.

You can get the load balancer a URL

and check it from the browser for now.

I will do that in the next lecture.

That is all in this session.

Summary of Steps to Create an Auto-Scaling Group for a Tomcat Instance

1. **Create the AMI**:
   - Go to your instance in the AWS Management Console.
   - Select the instance, go to "Actions" > "Image" > "Create Image".
   - Give it a name (e.g., `` `vprofile-app-image` ``) and a description, then create the image.
   - Wait until the AMI is available.
2. **Create the Launch Template**:
   - Go to the EC2 dashboard and select "Launch Templates" > "Create launch template".
   - Provide a name (e.g., `` `vprofile-app-LC` ``) and description.
   - Select the AMI created in the previous step.
   - Choose the instance type (e.g., `` `t2.micro` ``).
   - Select the key pair used to launch the original instance.
   - Choose the appropriate security group (e.g., `` `vprofile app security group` ``).
   - Add resource tags for easy identification (e.g., `` `Name: vprofile-app` ``, `` `Project: vprofile` ``).
   - Attach the IAM role used by the original instance (e.g., `` `s3 role` ``).
   - Create the launch template.
3. **Create the Auto-Scaling Group**:
   - Go to the EC2 dashboard and select "Auto Scaling Groups" > "Create Auto Scaling group".
   - Name the group (e.g., `` `vprofile-app-ASG` ``).
   - Select the launch template created in the previous step.
   - Choose the default VPC and select all availability zones.
   - Attach to an existing load balancer by selecting the target group.
   - Enable Elastic Load Balancing health checks.
   - Set the desired, minimum, and maximum capacities (e.g., 1 for minimum and desired, 4 for maximum).
   - Set the scaling policy (e.g., CPU utilization above 50%).
   - Optionally, configure notifications and tags.
   - Review the settings and create the auto-scaling group.
4. **Wait for Instances to Launch**:
   - Monitor the auto-scaling group as it launches instances and adjusts the desired capacity.
   - Verify that the instances are being added to the target group and are in a healthy state.
   - Terminate the original instance that was used to create the AMI.
5. **Configure Load Balancer Stickiness** (specific to vprofile application):

  - Go to the target group in the Load Balancer settings.
  - Enable stickiness to ensure that user sessions are routed to the same instance.
6. **Verify the Setup**:
  - Use the load balancer's URL to check the application from a browser.
  - Ensure that the application is functioning as expected with the auto-scaling configuration.

These steps will create an auto-scaling group for your Tomcat instance, ensuring that the application can scale in and out based on the load.

Instructor: Okay, so wait until the Auto Scaling Group

is completely stabilized.

You see the instance in the target group.

It shows healthy.

If everything is good, then let's check the URL once again.

If you're already logged in from the same browser,

then you should see the login page, sorry, the dashboard.

If you are not logged in, you will see the login page.

Anyways, this is the validation

that we wanted to do previously.

We validated everything, RabbitMQ, Memcached,

and the login as well.

Now let's summarize. Let's see what all we did.

Let's go back to the architectural diagram again.

If you have come this far, then really good job.

So you now as a user access your app through a URL

that points to the load balancer endpoint.

With HTTPS connection,

you access your application load balancer endpoint.

The certificate is an ACM, Amazon Certificate Manager.

Application load balancer are in a security group

that allows only 4,43 HTTPS traffic

which then forwards the request to Tomcat EC2 instance

on port 8080, which is in another security group.

For the backend servers that it accesses with name,

its private IP mapping is given in the private DNS zone.

Our backend servers are in one security group,

Memcached, RabbitMQ, and MySQL.

And now whenever we want,

we can upload a new artifact to S3 bucket

and download it to our Tomcat EC2 instances.

Now that's not such an efficient way of deploying artifact.

The right way of deploying artifact

we're going to learn in CI/CD projects,

which will be completely automated.

If you wish to create Auto Scaling Group

for Memcached, RabbitMQ, and database, you can do that.

It's very easy to do.

You can just select your instance action, instance setting,

attach to Auto Scaling Group,

and you can give a new Auto Scaling Group name over here,

but there are really better ways to do these things in AWS.

Instead of using Auto Scaling Group and EC2 instances,

we can use some PaaS and SaaS services

which we're going to see in the next project.

We are going to refactor our application stack.

So from EC2 instances, from virtual machines,

we'll start migrating to AWS managed services.

And that is all in this project. Thank you for watching.

I will see you in the next project. Keep practicing.

(upbeat music)

Click the "Create a new note" box, the "+" button, or press "B" to make your first note.

Summary of Final Steps and Next Steps in the Project

Final Validation and Summary

1. **Validate Auto Scaling Group**:
   - Ensure the auto-scaling group is stable and the instance in the target group shows as healthy.
   - Check the application URL. If already logged in, you should see the dashboard. Otherwise, you will see the login page.
   - Validate all components: RabbitMQ, Memcached, and the login functionality.
2. **Review the Architecture**:
   - **User Access**:
     - Users access the application through a URL that points to the load balancer endpoint with HTTPS.
     - The certificate is managed by ACM (Amazon Certificate Manager).
   - **Application Load Balancer**:
     - The load balancer is in a security group allowing only HTTPS traffic on port 443.
     - It forwards requests to the Tomcat EC2 instance on port 8080, which is in another security group.
   - **Backend Servers**:
     - Backend servers (Memcached, RabbitMQ, MySQL) are in a private DNS zone and a specific security group.
   - **Artifact Deployment**:
     - Artifacts can be uploaded to an S3 bucket and downloaded to the Tomcat EC2 instances.
     - For more efficient artifact deployment, CI/CD projects will automate this process.
3. **Auto Scaling for Backend Servers**:
   - If needed, you can create auto-scaling groups for Memcached, RabbitMQ, and MySQL.
   - Use instance settings to attach to a new auto-scaling group.
   - AWS offers better ways to manage these services using PaaS and SaaS, which will be covered in the next project.
4. **Next Project**:
   - Refactor the application stack to use AWS managed services instead of EC2 instances and auto-scaling groups.
   - Transition from virtual machines to more efficient AWS managed services.

Key Takeaways

- **Auto Scaling**:
  - Ensures the application can scale based on load.
  - Validates the health of instances and maintains desired capacity.
- **Architecture**:
  - Secure and scalable architecture using load balancers and security groups.
- **Deployment**:
  - Current manual artifact deployment through S3.
  - Future automated deployment with CI/CD pipelines.
- **Next Steps**:
  - Learn and implement AWS managed services for better efficiency and management.

Conclusion

- Practice the concepts learned in this project.
- Prepare for the next project focusing on AWS managed services and advanced deployment techniques.

Note: The upcoming projects will enhance your knowledge of AWS services and improve the efficiency of your application deployment and management. Keep practicing and stay tuned for the next project.

Out of all the transcripts i ve given you above give me a comprehensive notes what i did and how i did

Comprehensive Notes on the Project

Objective

The goal was to set up a scalable and resilient application architecture using AWS services. This involved configuring an Auto Scaling Group (ASG) for a Tomcat instance, integrating various AWS services, and preparing for future refactoring to managed services.

Steps Taken

1. Create an AMI (Amazon Machine Image)

- **Action**:
  - From the EC2 Dashboard, select the instance you want to create an AMI from.
  - Go to `Actions` -> `Instance State` -> `Image and Create Image`.
  - Name the image (e.g., `vprofile-app-image`), add a description, and create the image.
- **Outcome**:
  - The AMI is created and will be available once its status changes to `Available`.

2. Create a Launch Template

- **Purpose**: Define how instances should be launched.
- **Action**:
    - Navigate to `` `Launch Templates` `` and click `` `Create launch template` ``.
    - Provide a name (e.g., `` `vprofile-app-LC` ``) and description.
    - Select the previously created AMI from the `` `Application and OS Images` ``.
    - Specify instance type (`` `t2.micro` ``), key pair (`` `vprofile-prod-key` ``), and security group (`` `vprofile app security group` ``).
    - Add tags (e.g., `` `Name: vprofile-app` ``, `` `Project: vprofile` ``).
    - Attach the IAM role (`` `s3 role` ``) used by the instance.
    - Click `` `Create launch template` ``.
- **Outcome**:
    - A launch template is created to define instance configuration details.

3. Create an Auto Scaling Group (ASG)

- **Purpose**: Automatically manage the number of instances based on load.
- **Action**:
    - Go to `` `Auto Scaling Groups` `` and click `` `Create Auto Scaling group` ``.
    - Name the ASG (e.g., `` `vprofile-app-ASG` ``), and select the previously created launch template.
    - Configure the VPC and availability zones.
    - Attach to an existing load balancer's target group.
    - Enable Elastic Load Balancing health checks to ensure instances are healthy based on the load balancer's checks.
    - Set initial capacity: 1 desired instance, 1 minimum instance, and configure scaling policies.
    - Review settings and create the ASG.
- **Outcome**:
    - The ASG is created and configured to launch and manage instances automatically based on specified policies.

4. Instance Management

- **Action**:
    - After creating the ASG, monitor instance status. The ASG will launch and terminate instances based on load and scaling policies.
    - Terminate the initial instance (e.g., `` `vprofile-app01` ``) used for AMI creation.
- **Outcome**:
    - The ASG should stabilize to the desired number of instances, showing healthy status in the target group.

5. Validate and Test

- **Action**:
    - Check the application URL to validate if the application is accessible.
    - Ensure the target group shows the instance as healthy.
    - If logged in, verify if you see the dashboard; otherwise, the login page should be visible.
- **Outcome**:
    - Successful validation of application access through the load balancer.

6. Enable Stickiness (For Specific Applications)

- **Purpose**: Ensure that users are consistently routed to the same instance.

- **Action**:
  - Go to the target group settings.
  - Edit attributes to enable stickiness.
  - Save changes.
- **Outcome**:
  - Stickiness is enabled, helping maintain user sessions consistently with the same instance.

Summary of Architecture

1. **User Access**:
   - Users access the application through a load balancer endpoint using HTTPS.
   - SSL certificates are managed by ACM.
2. **Load Balancer**:
   - Security group allows only HTTPS traffic on port 443.
   - Forwards requests to Tomcat EC2 instances on port 8080.
3. **Backend Services**:
   - Services like Memcached, RabbitMQ, and MySQL are secured in specific security groups and accessed using private DNS zones.
4. **Artifact Deployment**:
   - Artifacts are uploaded to an S3 bucket and downloaded to EC2 instances.
5. **Future Improvements**:
   - Transition to AWS managed services for better scalability and efficiency.
   - Implement CI/CD pipelines for automated artifact deployment.

Next Steps

1. **Refactor to Managed Services**:
   - Move from EC2 instances to AWS managed services (e.g., ECS, EKS, RDS).
2. **Explore CI/CD**:
   - Learn and implement CI/CD practices to automate deployment and improve efficiency.

By following these steps, you successfully set up a scalable application environment on AWS, validated its functionality, and prepared for future improvements.

ChatGPT can make mistakes. Check important info.