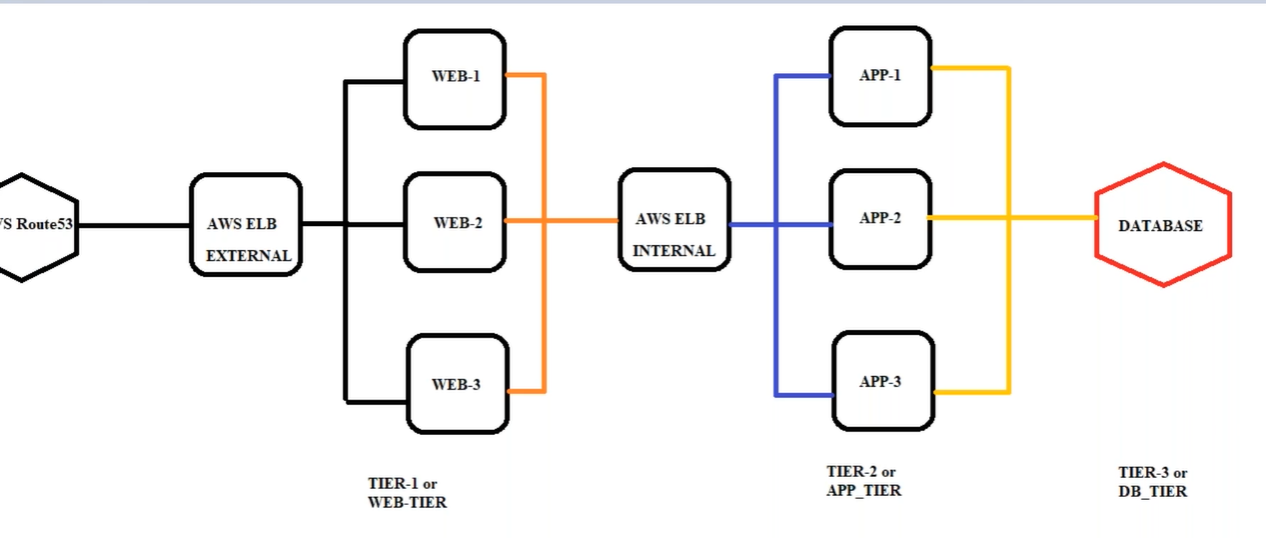
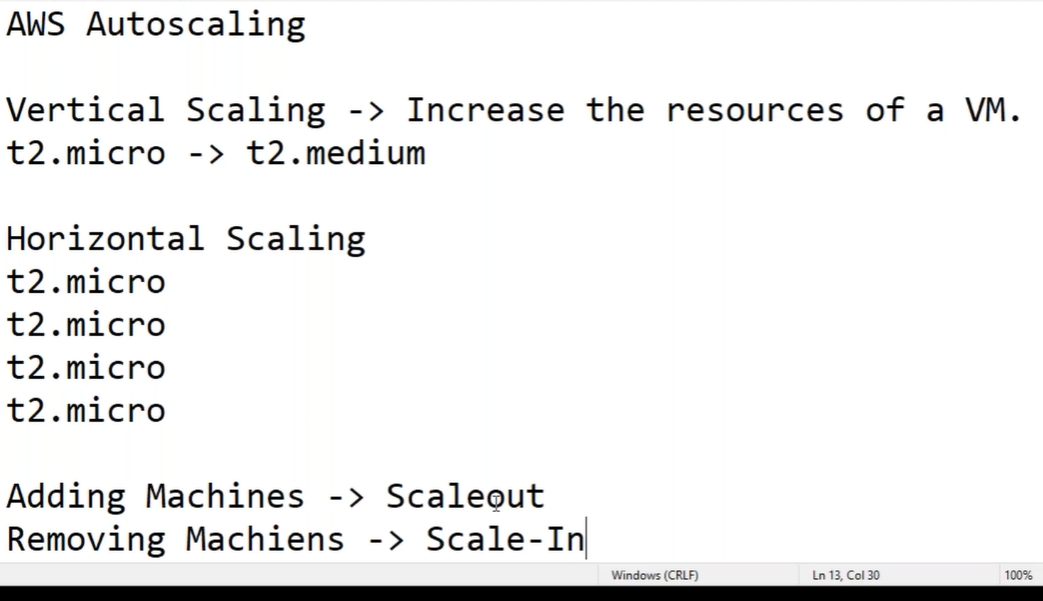
Class 22

**AWS Autoscaling**

We have 3tire architecture in Servers setup, traffic comes from internet



🡺 When we create Load Balancer, we have internet-facing(external) & internal



**AWS Autoscaling**: scaling means increasing the resources, whether it is capacity {(t2. micro [1cpu, 1gb ram]) to (t2. medium [2cpu, 4gb ram])} or instances (web server 1, web server 2, web server 3,) when load or usage of web server or App server is increasing

There are two types of Autoscaling:

1, Vertical Scaling

2, Horizontal scaling

**1, Vertical Scaling**:

Vertical Scaling means increasing the instance type or instance capacity like {(t2. micro [1cpu, 1gb ram]) to (t2. medium [2cpu, 4gb ram])} when Load is increasing on server.

* So, if we use Vertical Scaling, here the **Problem** is we need to stop the instance and then increase the capacity and start (we get down time of server), that may lead to increase load on other web servers 2 & 3.
* Vertical Scaling is used only for Database server, we can’t increase on the fly

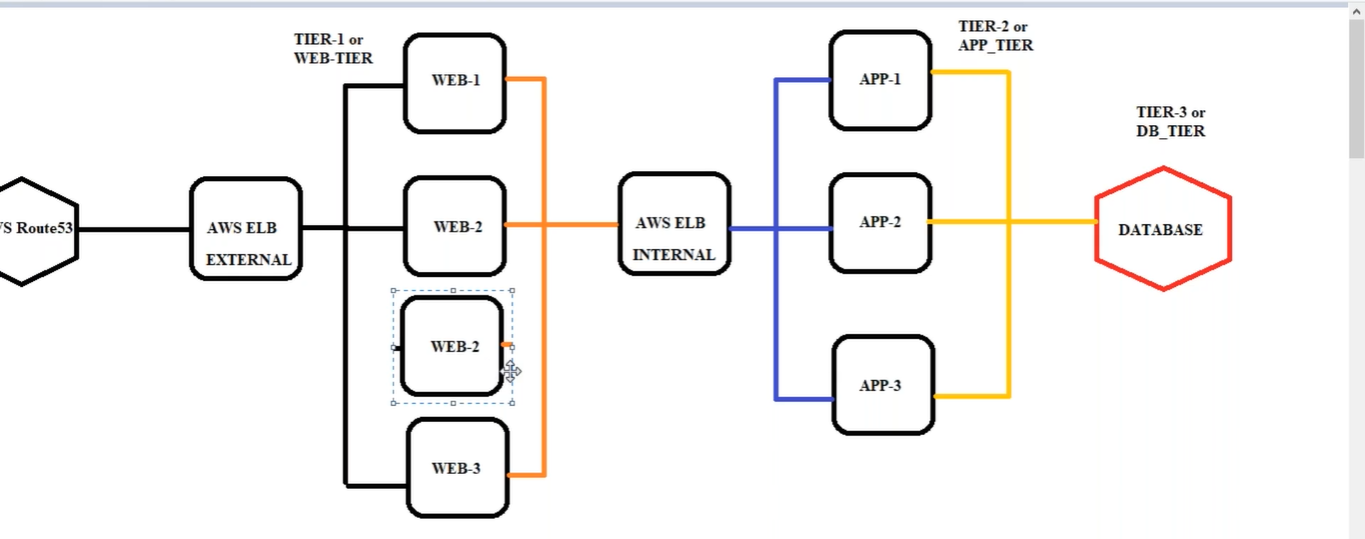
**Note**: To resolve this issue we go with Horizontal scaling

**2, Horizontal scaling**:Here instead of increasing capacity of instance, we increase Machines or servers using AMI image, on the fly.

* Adding Machines -> Scale out

Removing machines -> scale In

* We keep Alarms, when Load increase servers increase and when load decrease server comes to initial number. Eg: if 60% of Load increase one server increase, when Load increases to 70% another machine increase

Lab:

1, Ec2 instance -> ubuntu 18.4 -> “imageserver”

2, Putty -> ubuntu@ ->

3, go to <https://github.com/mavrick202> -> Repositories -> ansibletemplatetesting -> ubuntushellscript ->

Copy each ansible command line (2 -11 except 3)

2, putty -> past each ansible command line (2 -11 except 3)

**Note**: ansible - Linux based configuration tool

**Expectation**: my expectation by executing these commands is to check Load Balancing is done properly

in Autoscaling

|  |  |
| --- | --- |
| 1, | #!/bin/bash |
| 2, | apt update |
| 3, | apt install software-properties-common -y |
| 4, | apt-add-repository --yes --update ppa:ansible/ansible |
| 5, | apt update |
| 6, | apt install ansible -y (python installation) |
| 7, | apt install nginx -y (nginx installation) |
| 8, | apt install python-apt -y (python apt installation) |
| 9, | apt install git -y |
| 10, | apt install stress -y (stress installation- Load on cpu if it reaches to certain limit) |
| 11, | git clone -b aws-autoscaling https://github.com/mavrick202/ansibletemplatetesting.git /myrepo  (clone or copy ansibletemplatetesting folder in /myrepo)  16, ansible-playbook /myrepo/playbook.yaml |

1, Ec2 instance -> “imageserver” -> RC-> image and template -> create image (AMI created) -> image name: Autoscallingimagev1

4, Ec2 -> AMI -> RC -> Launch -> in box #!/bin/bash

ansible-playbook /myrepo/playbook.yaml

Testing is perfect

5, Ec2 -> Load Balancer -> Create Load Balancer -> Network Load Balancer -> Name: Autoscaling-NLB

-> select VPC & Availability Zones -> Target group: New target group -> Name: Autoscaling-TG

Now go to description and copy DNS name of Load Balancer

6, Route53 -> Hosted zones -> Create hosted zone -> Domain name: dheerajpalvai.xyz -> copy NS – DNS

Servers -> create record -> www -> IP -> paste DNS name of Load Balancer-> CNAME

7, godaddy.com -> My Products -> DNS -> Nameservers -> paste NS – DNS Servers

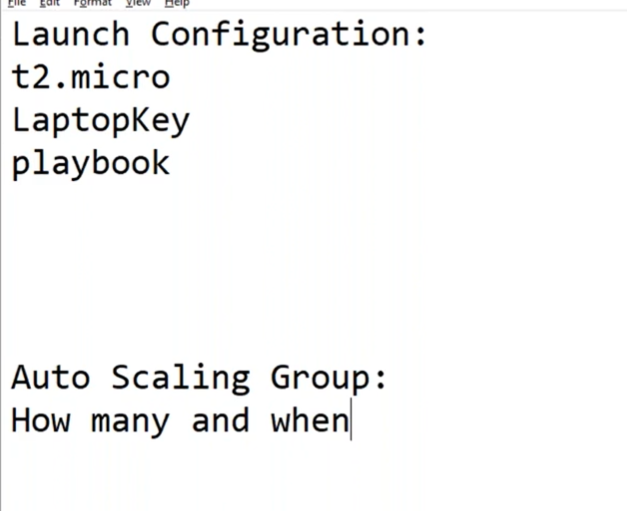
Q, How do you troubleshoot your server?

Goto instance -> Rc -> monitor and troubleshooting ->get system log

And check ansible in server through putty-> cat /myrepo/playbook.yaml -> and check src is from

/myrepo or not

**Details we give in Launch Configuration & Autoscalling**



8, launch configuration:

Ec2 -> launch configuration -> AWSB26-launch configuration -> AMI select -> Instance type: t2.micro

-> Advanced details: User data: #!/bin/bash

ansible-playbook /myrepo/playbook.yaml

->security group: allow all -> key pair

9, Auto Scaling Groups:

Auto Scaling group name: AWSB26-ASG -> switch to launch configuration -> VPC & 3 Subnets -> Enable load balancing & Choose a target group for your load balancer -> Health checks: ELB -> 30 sec -> Maximum capacity: 3 -> Key: Name Value: Autoscalingserver

[here one server Deployed automatically]

9, In EC2 -> Auto Scaling groups -> AWSB26-ASG ->

Details -> {we gave Desired capacity =1, that’s why 1 server deployed automatically, if we change it

to 3 then 3 machines deploy automatically}

Activity -> Activity history -> {we can see here machines deployed successfully}

Automatic scaling -> Add Policy -> Policy type: simple scaling

Scaling policy name: CPUMORETHAN50 (Lode on cup > 50%)

Take the action:2 (that means Load > 50% add 2 machines)

CloudWatch alarm: Create a CloudWatch alarm (click on it)

{cloud alarm will tell you when load reaches to 50}

\* Create a CloudWatch alarm-> select metric-> EC2 -> By Auto Scaling Group Graph search ->

CPUUtilization (in Metric Name) [select a metric]

* Step 1  
  **Specify metric and conditions**
* Step 2  
  Configure actions
* Step 3  
  Add name and description
* Step 4  
  Preview and create

Top of Form

Step1:

Metric name: CPUUtilization

AutoScalingGroupName: AWSB26-ASG

Statistic: Maximum

Period: 1 minute

Greater/Equal

>= threshold

50

Note: {That means CPU maximum utilization for period of 1 minute which is greater than or equal to 50}

{Alarm should ring and trigger and auto install of 2 machines}

Step 2:

Send a notification to…

[dheerajpalvai@gmail.com](mailto:dheerajpalvai@gmail.com)

step 3:

Alarm name: CPU > 50

Step 4:

Create alarm

* Same steps with cpu < 30:

So, come back to Automatic scaling and set CPU > 50 Add2 units

CPU < 30 Remove 2 units

For 30 seconds each.

2, Putty -> RC on top of putty window -> Duplication session

First window: stress

Copy Example: stress and past and give 30m (that means putting load for 30 minutes on morethan cpu 50)

Second window: htop (shows utilization of window)

**Chain process:**

**Stress increase on instance**

**|**

**Cloud watch alarm rings**

**|**

**Goes and tell to Autoscaling (then increases machines)**

**|**

**And those added machines add in Load balancer**

**Data flow and Lab:**

1, Godaddy (DNS servers)

|

2, Route53 (DNS servers) (DNS name)

|

3, Loadbalancer{Target group} (DNS name)

|

4, AutoScaling{Launch configuration} (configure Load balancer in Autoscalling)

|

5, AMI (from instance)