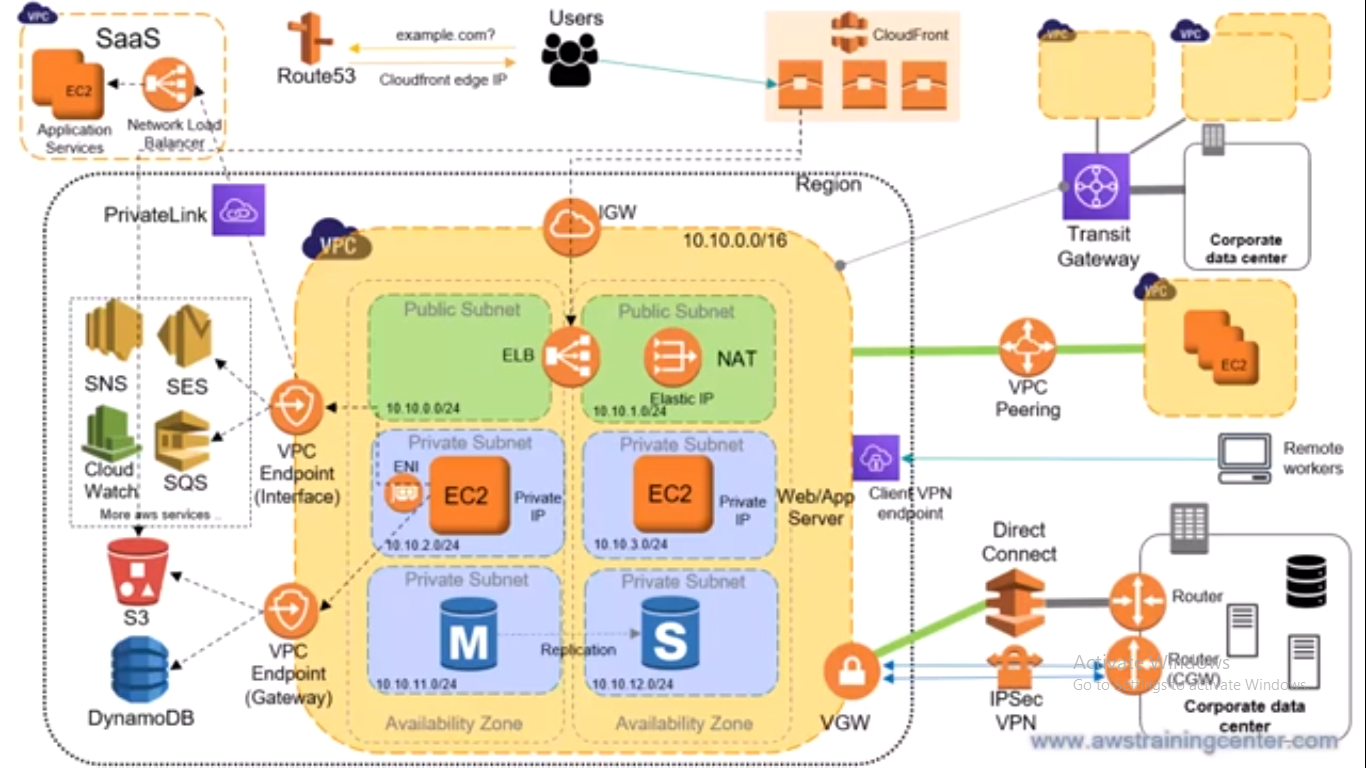
**Introduction to AWS Networking**

****

**VPC**

1. Availability zone nothing for private subnet

2. DNS hostname assign in VPC->Edit hostname->enable->Edit resolution->enable

EC2 DNS name will appear in instances

3. ec2metadata are

**EC2 AMI(its just like making our own OS)**

Create virtual machine within AWS EC2

Machine image is template we can make multiple instance deploy

Launch as many as instance from AMI

Can launch instances as many different AMI you need

Image can customise using EC2 user data

2 type AMI- public AMI and private AMI-personal use package user data run faster boot time

Copy AMI in other region EC2 dash->AMI->actions->destination region or other AWS account

EC2 dash->AMI->actions->launch instances

Register or deregister specially for snapshot and pub n pvt

**Inspector**

Create EC2 instances and then assesment in Inspector,

Deselect on checkbox for all instances tag name for EC2 instances

**EC2**

Placement group->instances grouping into production and development group

Shutdown->stop

Protect against accidental termination(tick)

Tenancy->shared run shared hardware instance

(this below step will directly install while running below)

User data->as text->#!/bin/bash

->Apt update –y

->Apt install apache -y

Ssh connect

ubuntu@ip-172-31-0-223:~$ ls

To switch to root user

ubuntu@ip-172-31-0-223:~$ sudo su

root@ip-172-31-0-223:/home/ubuntu# service apache2 status

root@ip-172-31-0-223:/home/ubuntu# apt install apache2 –y

ubuntu page will be seen

Autoscaling(Target group, Configuration, Autoscaling)

First launch **configuration** and then create **autoscalling**

1.Configuration

AMI

Instance type

Create new SG

SSH-TCP-22-Anywhere

Custom TCP-TCP-80-Anywhere

Key pair->existing or create new key for connection with server

2.Autoscalling

VPC

Select all subnet in VPC

Enable loadbalancing->

**Target group**-> Loadbalancer need target group to connect with Autoscaling

(Target group health checks on targets and tells loadbalancer where to direct traffic to EC2, fixed IP, AWS lambda )

Create target group->instances->group name->add in autoscaling

Healthchecks-300 sec

Group size ->desired,min,max capacity

Scalling policy->target tracking scaling policy->Name, Metric type->cpu utilisation->instances reed->300

Add notifications->

After above ssh in all EC2 instances cli and install and generate load a stress utility->

**sudo amazon-linux-extras install epel –y**

**sudo yum install stress –y**

**stress –c 8** (c is for cpu and processes 8)

\*\*just for saving money cpu utilisation reaches 50% new instances created

For target group individually EC2-Autoscalling group->Automatic scalling->Add policy->Add target group

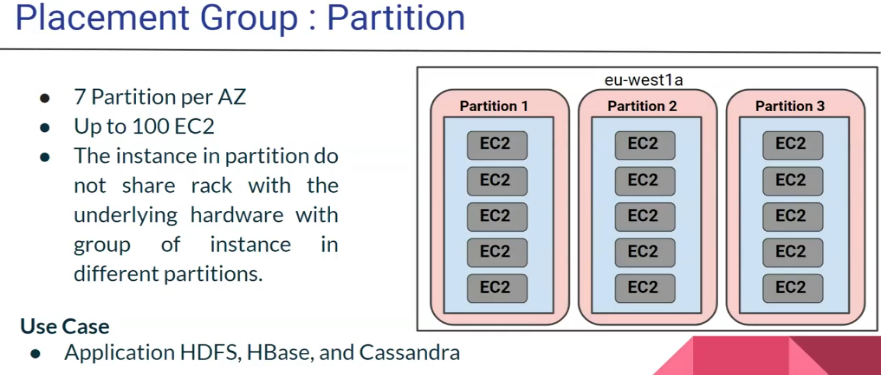
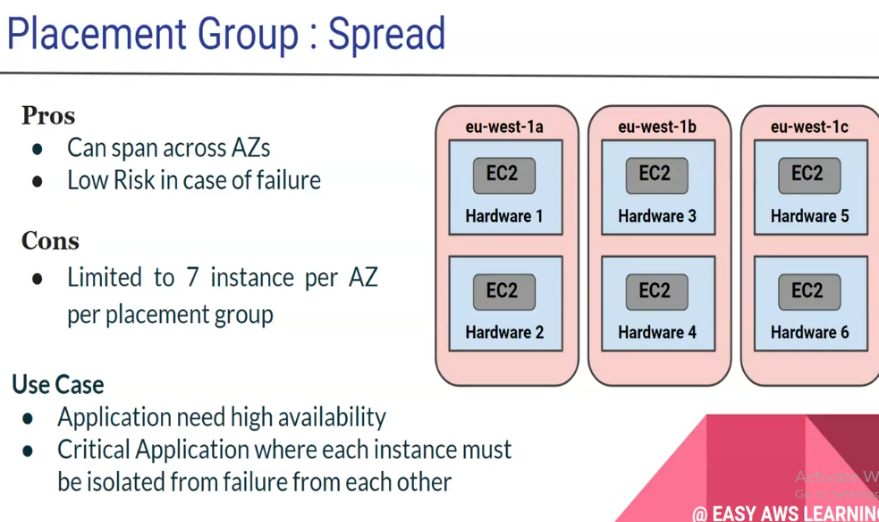
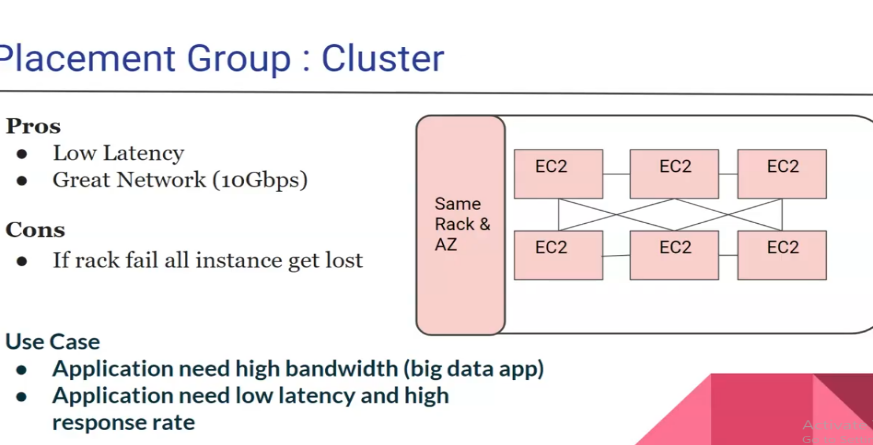
Check for logs->ASG->activity->logs

Placement group

Cluster- low latency, high bandwidth group in single AZ place in same racks upto 10gb but if racks fail all instances will fail and lost all but high performance rate

Spread- places small group of instances across distinct underlying hardware AZ(upto 7 instances per AZ per palcement group), each instance separated from failure of other

Partition-spread instance across different partition(rely on sets of racks) within AZ(upto 100 instances and 7 partition per AZ) and instance in partition do not share rack with the underlying with group of instance in different partitions.(deploy on particular instance)



ENI(Elastic Network Interface)

They are used to enable network connectivity for your instances and having more than one of them connected to your instance allows it to communicate on two different subnets.

Suppose there are two subnet1 and subnet2 subnet1 instances want to communicate with subnet2 then it is used

Create ENI->select ENI->Actions->attach instances subnet 2

See in instances Network interfaces-> eth 0 and eth1

**EBS**

EC2 for EBS

1. create EBS in EC2 gp2 delete on termination 2 gb

2. make EBS vol in linux <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-using-volumes.html>

3. for cross AZ for EBS connection with EC2 create snapshot(pending)

4. ec2-user@ip-172-31-26-45 $ lsblk

$ sudo file -s /dev/xvdf

$ sudo mkfs -t xfs /dev/xvdf (filesystem)

$ sudo mkdir /data (dir)

$ sudo mount /dev/xvdf /data (mount point)

$ lsblk

------this will create directory test and data mount in separate vol of 2GB created------

$ ls /data (check for data)

$ cd /data/

$ sudo mkdir test

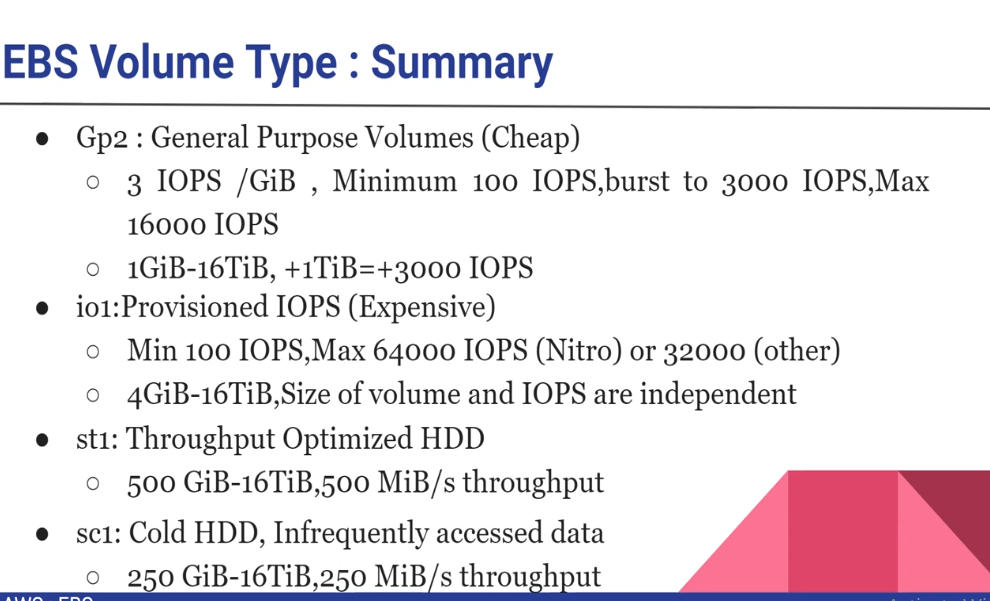
$ sudo touch test/testing.txt

$ ls test/testing.txt

Automatically mount an attached volume after reboot do following things but

Volume in EBS

Detach n then attach other instances in vol



EBS Snapshot

Snapshot is a point-in-time backup of your volume

Saves Incremental copy backups block changes

do not backup EBS during application handling lots of traffic load

Snapshots stored in S3

If making snapshot not necessary to detach volume

Max 100000 snapshots per region if required more request to S3

Can copy snapshot across AZ or region

Can make AMI from snapshot

Snapshots can be automated using AWS data lifecycle manager

Snapshots(below volume)-Actions->we can create volume, image

Snapshot->image->we can update size or vol type

AMI config and EBS vol can be updated

Snapshot can be copy in other region

Snapshot can be encrypted after created->snapshot->copy->encrypt

Note: vol and snapshot created are also not encrypted

From Snapshot vol created then it is encrypted otherwise it will not encypted

EBS lifecycle policy

Its only for automation of snapshot copy into other region

Create lifecycle policy->policytype->EBS,resource->vol, des,tags name created in snapshots, retention period(deletion period keeping data upto 10 days etc) ,this can be enabled or disabled

Encryption in vol

Vol and snapshot created are also not encrypted

From Snapshot vol created if snapshot is encrypted then vol will be encrypted otherwise it will not encrypted

We get option of encryption while creating vol

EBS migration to other AZ

Create snapshot on volume in AZ

Create volume on snapshot and define different AZ and attach this vol to ec2

EBS in Windows User

Remote RDP connect

Vol->2Gb->online->initialise

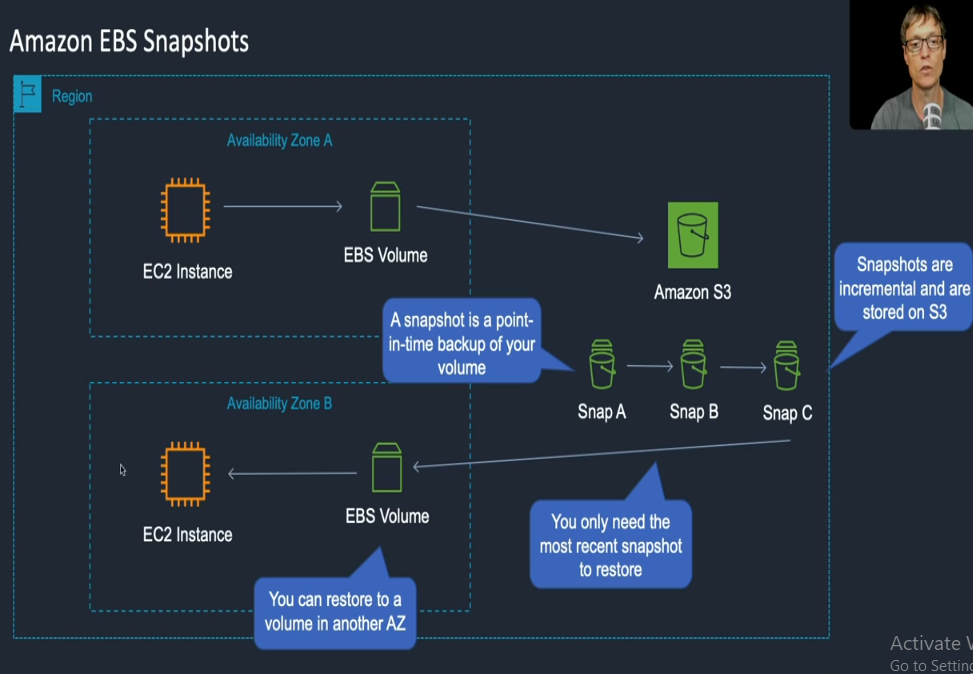
Instance Store volume

These are physically attach to EC2 host server and offer high performance and EBS vol attached over network

Instance store vol are ephemeral(non-persistent) but main dimerit of this is that if instance is stopped then it will be lost so only for temporary data

EBS Snapshots and AMI

If EC2 is in other AZ and if EBS vol snapshot data to be stored in other AZ only possible through S3



Demo: Create 2 EC2 in different AZ

AMI

From root vol of instances create snapshots and from this snapshot create AMI

AMI is to create same instances in different AZ

Snapshots->create image->name AMI

AMI->launch instances in other AZ

EFS

-Create two instances in different AZ and EFS with same SG

-SG create separately

-SG contain Http,https,ssh,RDP and create then add NFS(custom-select sg group name)

-EFS->Network->manage->SG it should be same for both EC2 instance add to all AZ

-EFS->attach follow below

-install in amazon linux 2 cmd

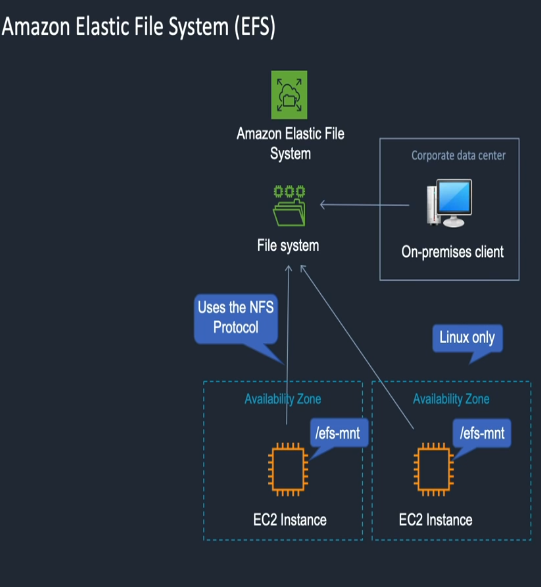
**sudo yum –y install amazon-efs-utils**

**sudo mkdir /mnt/efs**

**sudo mount –t efs –o tls fs-50918781:/ /mnt/efs**

**sudo touch test-file**

**ls**



**Automation on AWS**

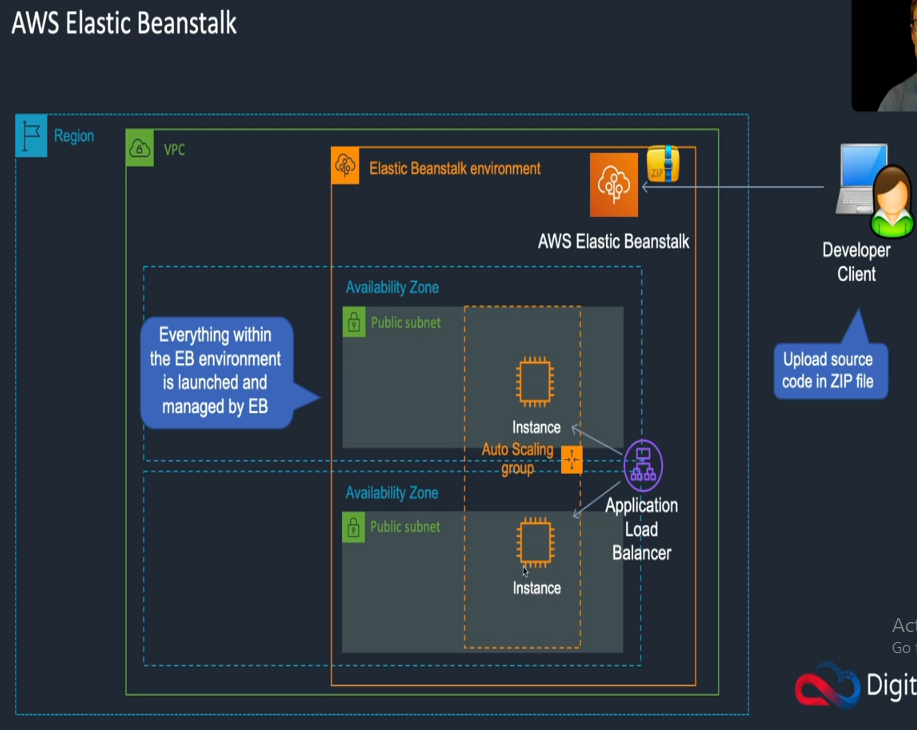
**CloudFormation**

Creates and config resources according to the template

Creates stacks and template on it, template ready then create stacks

Follow doc sample templates->application framework and services->for large application

**ElasticBeanstalk**

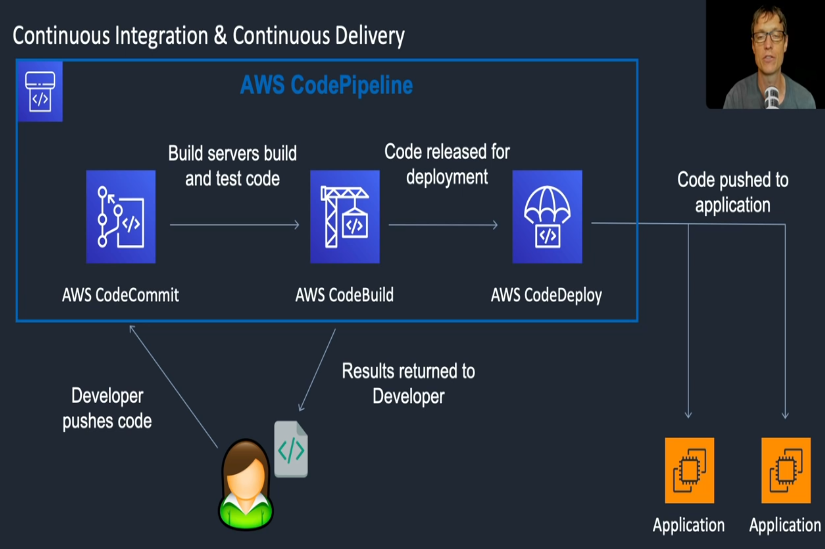
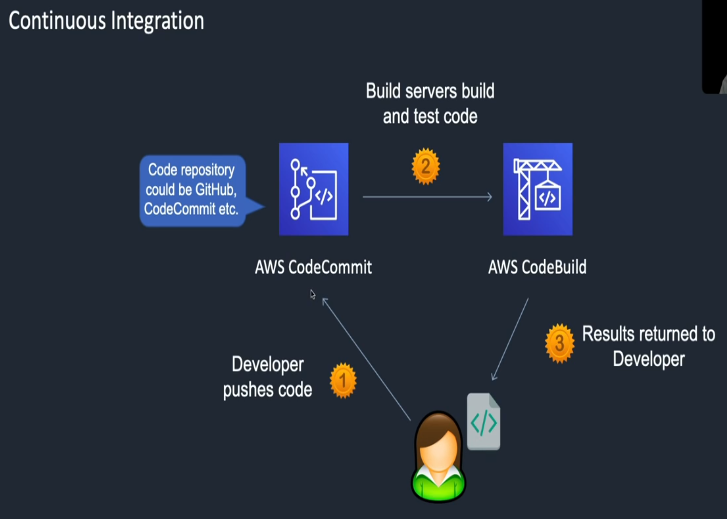


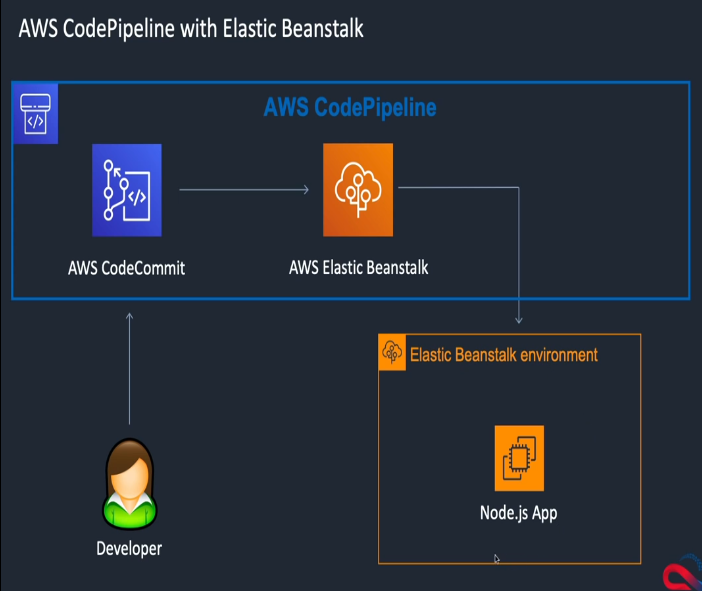
Elasticbeanstalk ->environment->configure more options->DB, ELB, High availability(mainly leave all create from outside)

Upload website or doc in Ebeanstalk

EB->environment->web app->platform

**DevOps**



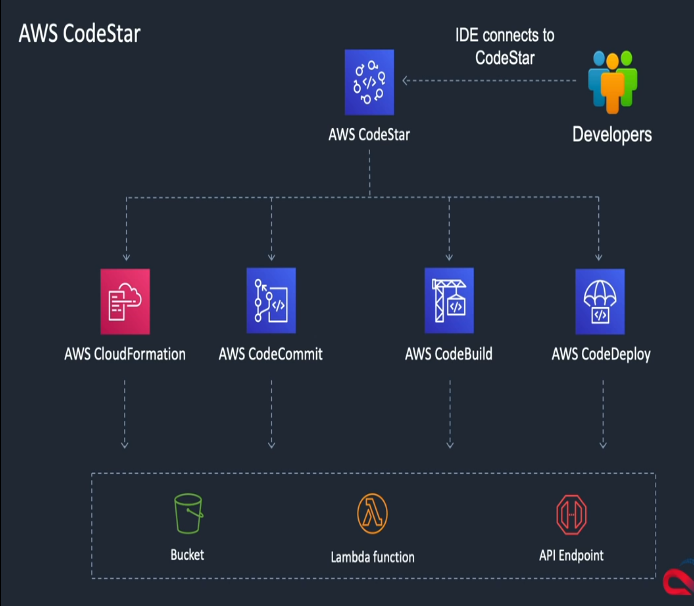


CodeCommit

CC->Repository->repo name->install git on your machine and gitclone repository in git cli

Get user credentials from IAM->Https Git Credentials user and password paste in git cli

CodeStar Project

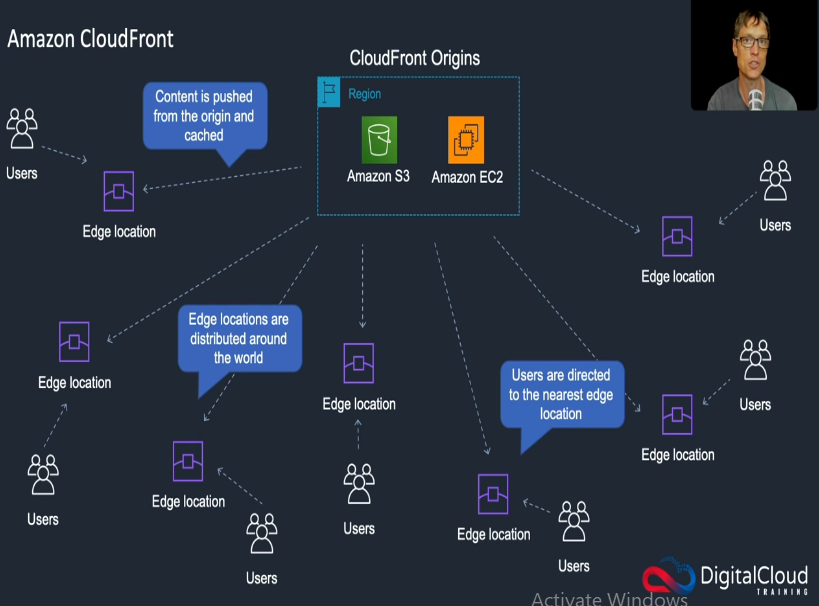


R53

Domain name

Hosted Zones->create records->CNAME(routes traffic to other domain name),A-type

**Create amazon Cloudfront with S3 static website**



Create bucket in S3 and need public access to bucket upload website each file like videos and data will have auto generated link

Create CloudFront

Origin\_domain->s3 bucketlink(appear automatically),

Origin path->folder file stored,

Origin\_id->S3bucket name,

Restrict access,

Redirect to https,

Distribution settings->Edge location->create

**Domain name** will be created in CF distributions to access this CF

Domain name eg:domainname(in cf distribution)/index.html

SSl/TLS

CM->get ssl trhough mail

CF->Distribution Settings->CNAME=example.net,SSl,Default root Object=index.html

Create records for DNS->type=A-Ipv4,alias=yes,alias\_target=CF name,Routing policy=simple

CLI >dig example.net

On windows ~nslookup

>Example.net

**Containers and Serverless Computing**

ECS runs your containers on acluster of Amazon EC2 virtual machine instances pre-installed with docker. It handles installing container, scaling, monitoring and managing these instances through both an API and the AWS management Console

**S3**



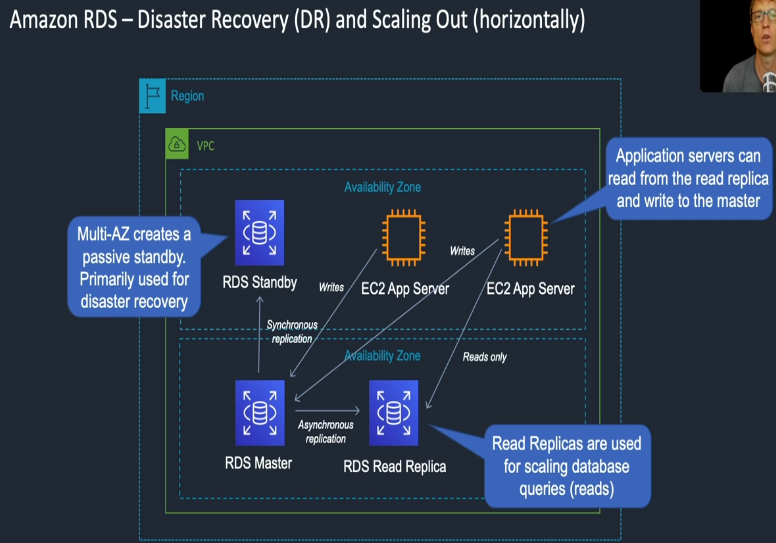
-Remove public access and permission for read,write make public

-Bucket policy:

-aws account permission

-make public access

**RDS**



-add SG MySQL/Aurora->custom->sg select

-multiple AZ select for multiple zone

-Actions->add replica ->it will have different endpoint to DB created

-Automated backups->retention period

-Snapshots create snap of db

Install WP with RDS

DynamoDb

**S3/EBS/EFS**

S3->write once read many times object storage, 5TB, availability zone may different EC2 ans S3, not fit for DB

EBS->Block storage, write many and read many times OS and for frequent database operation, 16 TB,availability zone should be same of EBS and EC2

EFS->unlimited storage, can be used in diffferent availability zone, mount on on-prem using VPN or direct connect, mount on multiple EC2 instance

**S3 Deploy static website**

Create bucket and Remove public access

On bucket->permission ->upload static website->enable

Upload web->make public

**ELB**

Application LB

Better to create new security groups for load balancer

Configure health check->ping protocol->http,ping port->80, ping path->health\_check.html

Response timeout->5 sec, interval->30 sec, unhelathy threshold->2(check 2 times for system is unhealthy), helathy threshold->3(check 3 times then report system is healthy)

Add all EC2 instances in VPC all subnets

Enable cross AZ

Enable connection draining->300 sec default(for healthy instances checks for 300 sec then redirect to other instances)

\*If loadbalancer->instances->OutofService->then ssh in that instance

$ Service httpd status

$ Service httpd start

Loadbalancer need target group to connect with Autoscaling

Target group in Step 4

**NACL**(Network ACL)

VPC->Security->NACL->when VPC created by default NACL created

NACL works on subnet level but Security groups works on instances level so traffic is blocked on NACL then it will not reached to Security group

Eg->rule-99 deny port 80 so ip in ec2 will not open

Inbound->rule->99>100 99 is greater in priority

**IAM**

Roles

In ssh ubuntu@ip:**sudo su**

root@ip:/home/ubuntu# **aws s3 ls**

result: unable to locate credentials

make roles for s3 full access and attach to EC2

root@ip:/home/ubuntu# aws s3 ls

result is below

**SSH into Private EC2 from Public EC2 ip**

-to ssh on pvt instance we have to ssh through public ip

-pvt instance group 22 port everywhere permission added for entering pvt instance we can restrict

-we can restrict permission in pvt instance through pub instance

-public EC2 SSH connect

Ubuntu@ip:$ sudo su

root@ip:/home/ubuntu #ls

root@ip:/home/ubuntu #ec2metdata(public instance info)

now copy pvt key pair to pub instance

user:~/Downloads$ scp –i test.pem(pub) test.pem(pvt) ubuntu@ip ()of pub EC2):/home/ubuntu/

now on public ec2

root@ip:/home/ubuntu# ls

test.pem

now connect pvt ec2 ssh –i “test.pem” ubuntu@ip-pvt instance

root@ip:/home/ubuntu# ssh –i “test.pem” ubuntu@ip pvt instance

ubuntu@ip:~$ sudo su

root@ip-pvt:/home/ubuntu# ec2metdata

**Cloudtrail**

**KMS**

Give access to IAM user and even role

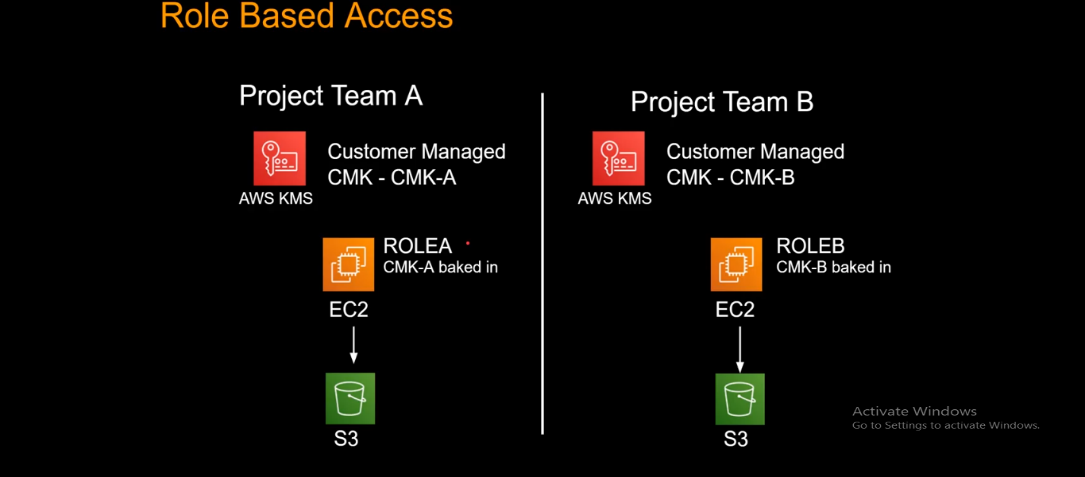
Add KMS

Apply to login IAM user(KMS permission)->S3->Permission->Encryption

AWS managed keys

Customer managed keys->keyactions->schedule key deletion

Custom key stores



**Lambda in PHP**

Arn from command prompt in last

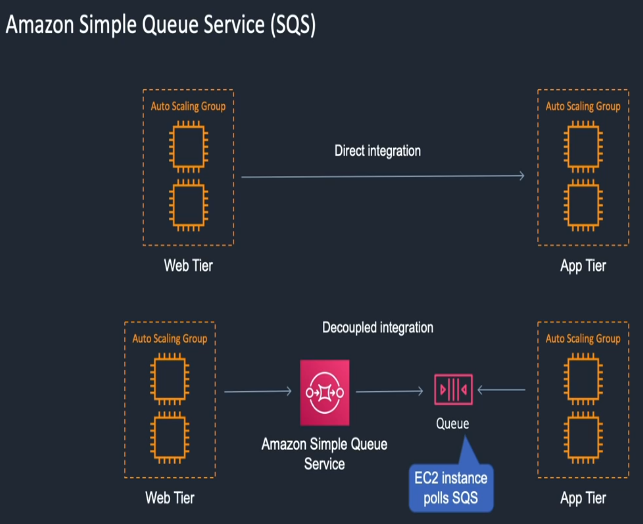
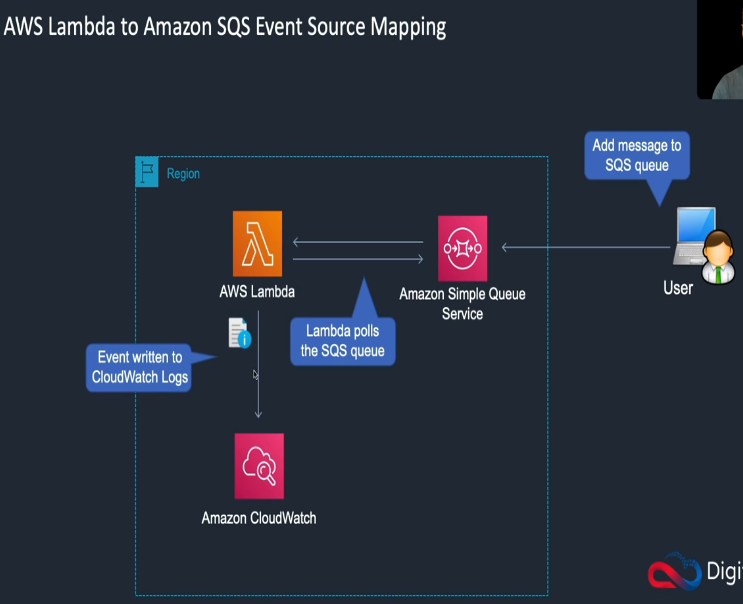
"LayerVersionArn": "arn:aws:lambda:ap-south-1:049185135366:layer:PHP-73-vendor:1",

Goto :https://blog.kukiel.dev/posts/AWS-PHP-LAMBDA-CLOUD-9.html

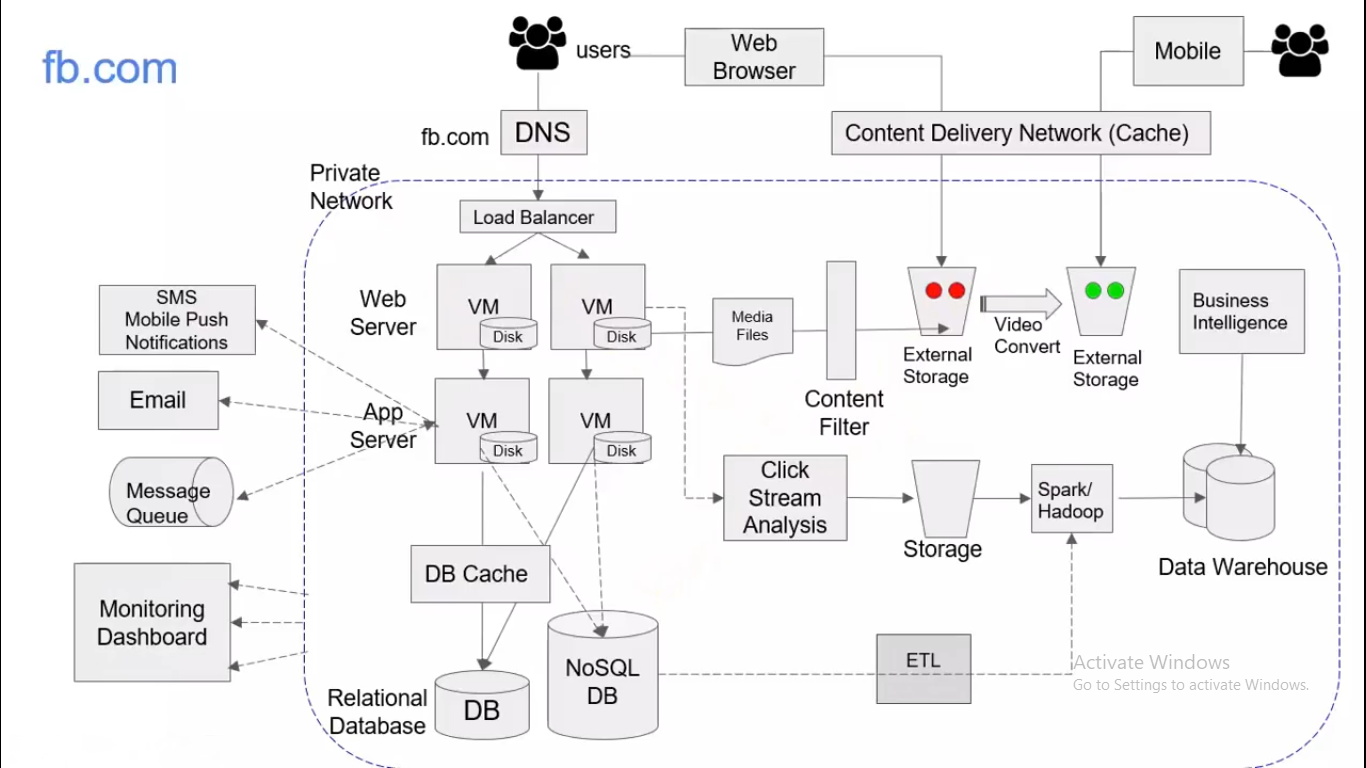
**Lambda with SQS+SNS+CW**

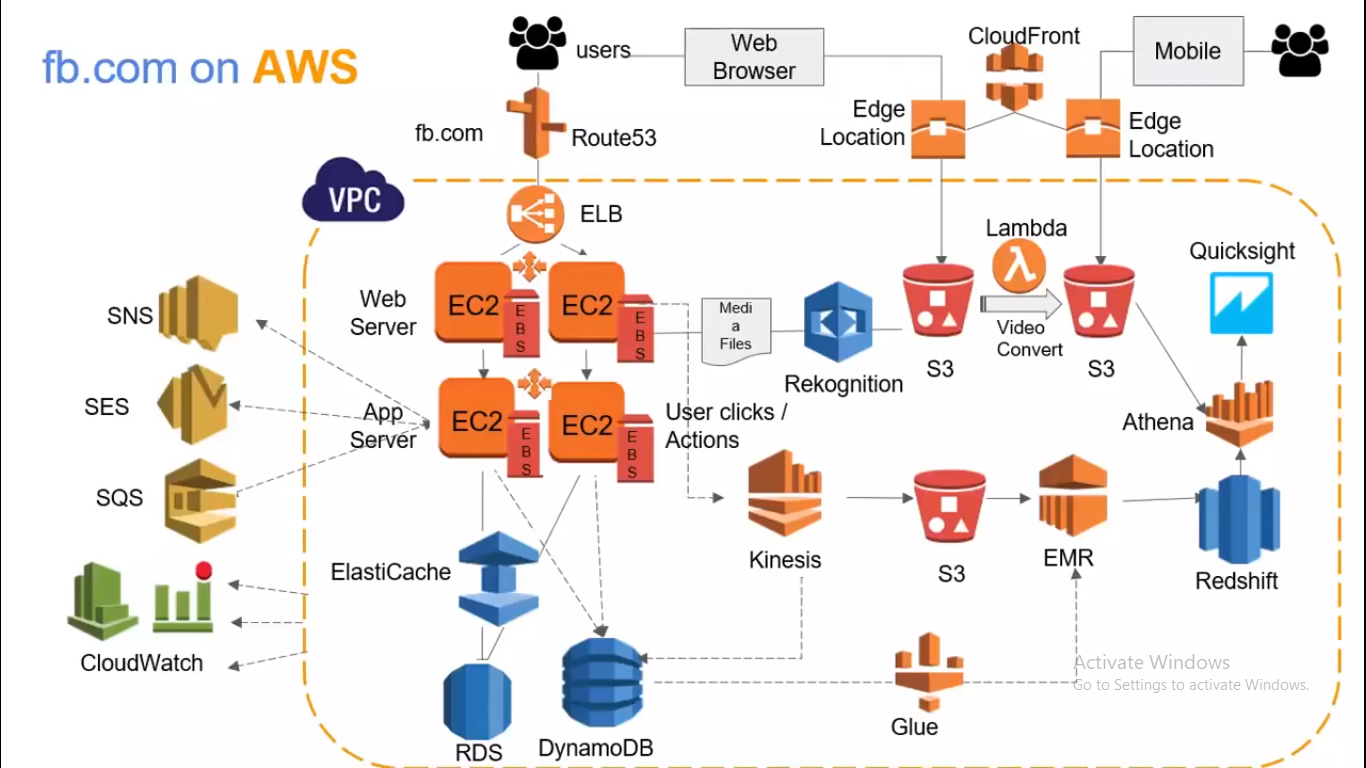
Lambda->permission->IAM->add policy->AWSLambdaSQSQueueExecutionRole

SQS->Lambda triggers🡪add->send and receive messages



**Introduction to AWS services**



****

**NoSql-**scalable database generally used for big data

Dbcache**(ElastiCache)-**Heavy read operations handling

Hadoop(**EMR**-Elastic MapReducing)-find meaning of data aggregation and sorting, jobs like spark and preling jobs

**ETL**(Extract,Transform,Load)(**Glue**)- Transformation refers to the cleansing and aggregation that may need for data to prepare it for analysis from NoSQL to EMR

Data Warehousing(**Redshift**)-find what trending what type of user region they come from and the secure electronic storage of information by a business or other organization

Business intelligence(**Athena**)-to pull data() from redshift send to S3

**Quicksight**-to view result from Athena(ie. Graph, charts, insight of data to take business decisions)

CDN(cache)(**Cloudfront**)-Cloudfront store cache data in Edge location

Monitoring Dashboard(**CloudWatch**)-Monitoring all this infrastructure

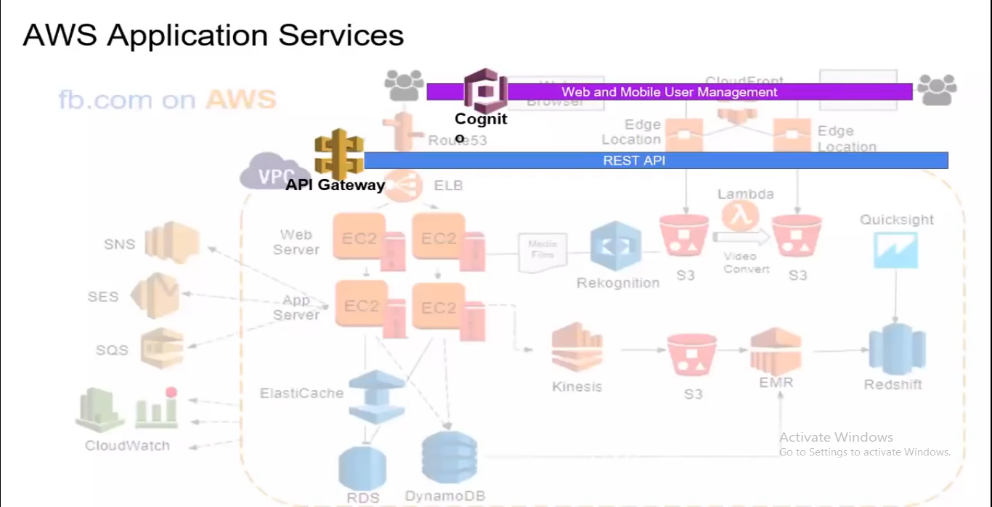
**IAM-**Provides security,Authentication,Authorization

**KMS-**Encrypt data in storage EBS, S3, EMR, Redshift etc

**ACM(**Amazon Certifcate Manager)-SSL enabled connection digital certificate to secure communication deploy on loadbalancer or cloudfront

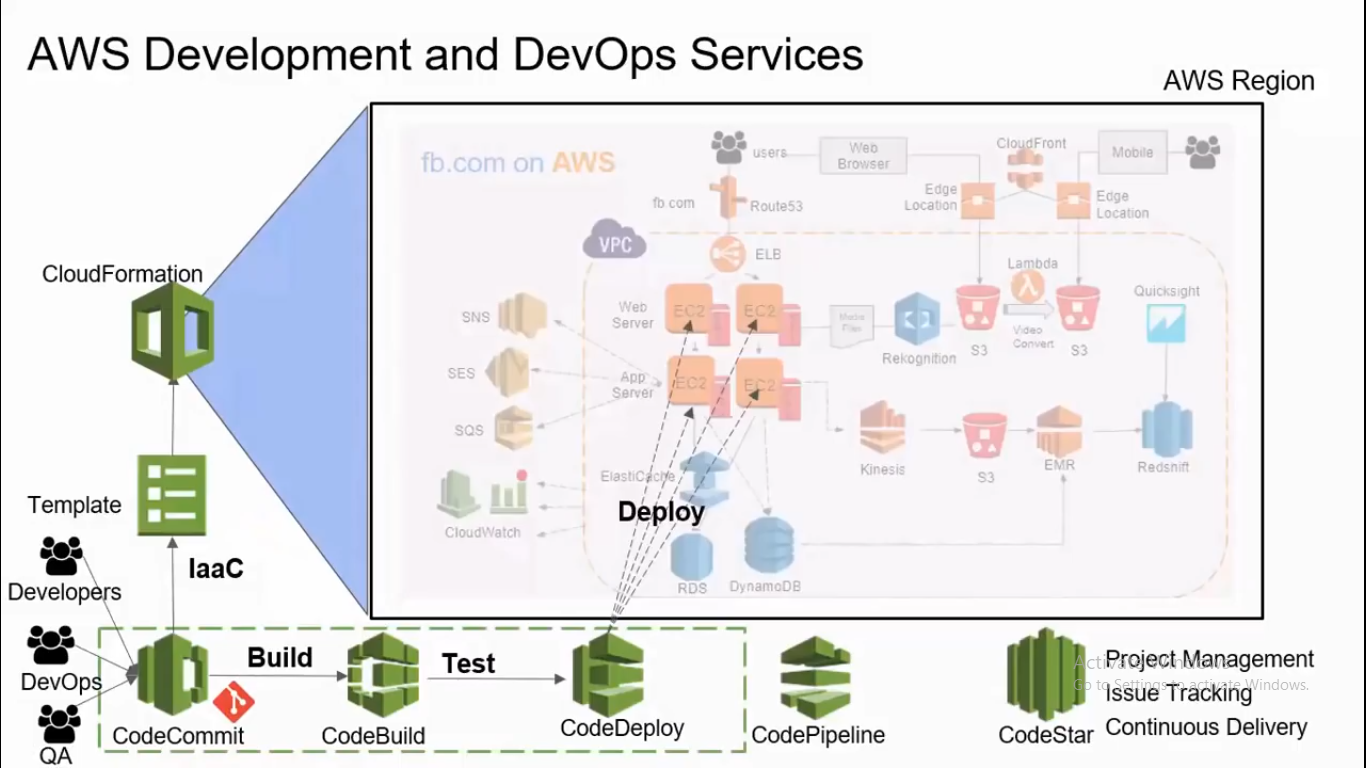
**WAF(**Web Application Firewall**)**-Prevents from attacks like sql ingestion, deployed on ELB, Cloudfront, VPC, API

**Inspector**-Puts agents in EC2 instances scan for unknown vulnerability checking and report to take action



**Cognito**-manage application identity signup application manage accesses

**DevOps**

****

**Cloudformation**-Creates template in json/yaml format creates structure above from scratch which provision our infrastructure from scratch

**IAAC**-Infrastructure as Code-codecommit takes that template and develop in cloudformation

**Codecommit**-git repository

**Codebuild**-AWS will take that code and build unit test produce artifacts(exe,binary executable)

**CodeDeploy**-Put this on EC2 machines

**CodePipeline**-Above three will be used integrated

**CodeStar**-Integrate for testing with testing tools jeera or others

**Implement laravel project to github**

Laravel aws name same in aws and github

1. **Elastic Benstalk**->create new **Application Name** and php version select create project **Environment name**
2. Github repository to connect **DeveloperTools**->**CodePipeline->**add github repository and branch
3. Choose pipeline setting
4. Add source stage
5. Add build stage->skip this
6. Add deploy stage

AWSElasticBeanstalk

Region->mumbai

Application\_name->laravelaws

Environment name->laravelaws-env(this is environment name created in ElasticBenstalk)

1. Review
2. To delete environment list for **Cloudformation**->**stacks**->Delete all environment
3. Goto pipeline deploy->click on pipeline execution ID
4. Elastic Benstalk-> Environment\_name->configuration->software->edit->document\_root=/public

Error:5000 error

Sol:gitignore comment .env file and **git add . and git commit –m”.env file added” git push –u origin master** in vscode

Goto pipeline deploy->**release change** button(changes will appear)

Error:connection refused in AWS

Sol: Elastic Benstalk(or EC2)->configuration->database

**Add Instance class=**db.t2.micro(free version other will be charged),username,password

from Elastic Benstalk->configuration->database ->get

Endpoint: [aab0fxpd9lc2jh.cfacsaa1gwhc.ap-south-1.rds.amazonaws.com:3306](https://ap-south-1.console.aws.amazon.com/rds/home?region=ap-south-1#dbinstances:id=aab0fxpd9lc2jh)

Username: larav6admin Password:12345678(directly if not available and forgot then go to modify)

**Amazon RDS->**databases->click on **DB Identifier->configuration->**get DB name

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE=lara6

DB\_USERNAME=root

DB\_PASSWORD=

Change to **endpoint** DB\_host and username and DB\_name

DB\_CONNECTION=mysql

DB\_HOST=aab0fxpd9lc2jh.cfacsaa1gwhc.ap-south-1.rds.amazonaws.com

DB\_PORT=3306

DB\_DATABASE=ebdb

DB\_USERNAME=larav6admin

DB\_PASSWORD=

AWS\_ACCESS\_KEY\_ID=AKIA25NQKQMN63JDHZDE

AWS\_SECRET\_ACCESS\_KEY=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

AWS\_DEFAULT\_REGION=ap-south-1

AWS\_BUCKET=awss3pj

AWS\_URL=https://awss3pj.s3.ap-south-1.amazonaws.com

Vscode:**php artisan migrate**

**php artisan key:generate**

Error: PDOException::("SQLSTATE[HY000] [2002] A connection attempt failed because the connected party did not properly respond after a period of time, or established connection failed because connected host has failed to respond.)

Sol:RDS->databases->database\_identifier->connectivity & security->security group-rules->EC2 Security Group-inbound(click on link)->Security Groups->inbound rules->edit inbound rules->add rule->

**Type**=MySQL/Aurora

**Source**=Anywhere Or Source=MyIP(if having dedicated IP)

**Save rules**

Error:Network->Headers->Server(nignix)

Sol:.htaccess->files don’t work

https://stackoverflow.com/questions/61640771/any-aws-eb-laravel-route-getting-404-not-found-nginx-1-16-1

**EC2(pending)**

1. Instances->Launch instances

Steps

1. Amazon linux 2 AMI(64bit \*86)
2. Automatically choosed (Choose instance according to size requirement)
3. IP and IAM role can be changed
4. Add storage size can be changed
5. Tag eg:AWS demo
6. Add rule->eg:HTTPS and HTTP->**Launch**->create new pair->key pair name(awsdemo)->download->launch

**S3**

1. Check for pricing in nav tag
2. Create Bucket
3. Name n region->versioning,encryption
4. Configure options
5. Set permission->public access(user can access file here)and manage system permission->don’t give (don’t allow permission to anyone)
6. Review
7. Now upload document
8. Select files->Manage users,don’t allow public permission to anyone
9. Set permissions->encryption
10. Set properties
11. Review
12. Config->filesystems.php->copy all s3 credentials to .env

AWS\_ACCESS\_KEY\_ID=AKIA25NQKQMN63JDHZDE

AWS\_SECRET\_ACCESS\_KEY=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

AWS\_DEFAULT\_REGION=ap-south-1

AWS\_BUCKET=awss3pj

AWS\_URL=https://awss3pj.s3.ap-south-1.amazonaws.com (region and bucket entered)

Note:AWS\_Bucket name from S3 bucket

Refer S3 document for **url** and region from **region and endpoints**

Laravel error:league\Filesystem\AWSS3v#\AwsS3Adaptor

Sol:laravel doc->filesystem->Driver Prerequisites->Amazon S3: composer require **league/flysystem-aws-s3-v3 ~1.0**

or

**composer require league/flysystem-aws-s3-v3 ~1.0 --dev**

1. Dropdown(AKS)->My security credentials->Create User->eg:demo-user
   1. **Programmatic access**->id users(selected) and **AWS Management** Console access->eg:uploaded file given access it will give access to see that file and change and AWS management password of AKS will be used
   2. **Attach existing policies**->search (S3)->AmazonS3FullAccess(select)(read,write)
   3. Add tags skip
   4. Review

6. after in laravel

Route::post('upload', function(){

    request()->file('file')->store(

        'my-file',

        's3'

    );

    return back();

})->name('upload');

  <div class="card-body">

    <form action="{{route('upload')}}" method="post" enctype="multipart/form-data">

                  @csrf

                  <input type="file" name="file" class="form-control">

                  <input type="submit" class="btn btn-primary">

         </form>

  </div>

//upload doc in AWS

error:call to member function is null()

sol:in form add enctype="multipart/form-data"

7. Don’t forgot to give permission to acces file in AWS under **Permission->**edit-> pass grantee

**SES in Laravel**

* 1. SMTP credentials->create IAM user n download password will not be seen later u have to create new user
  2. **Verifyemail addresses(**add email address side bar**)**->from which to send email

3. SMTP credentials->get port number, host(going back after creating user),tls

MAIL\_DRIVER=smtp

MAIL\_HOST=email-smtp.ap-south-1.amazonaws.com

MAIL\_PORT=465

MAIL\_USERNAME=AKIA25NQKQMNZ5ZFOZWY

MAIL\_PASSWORD=BEDAIrUw0dAoZo5d8pxiGvwhBrhuK0/2Zn78hDl4bQu7

MAIL\_ENCRYPTION=tls

MAIL\_FROM\_ADDRESS=[abhi144k@gmail.com](mailto:abhi144k@gmail.com)

4. php artisan make:mail Hellomail and views->mail

**Deploy Laravel in AWS**

1. **Domains**->Registered domains->Register Domain->create all info and search for domain

Hosted domain->container records

1. **Elastic IP address**->host domain to ip address i.e in AWS **EC2**

1. EC2->Network & Security->Elastic Ips->Allocate New Elastic IP->Allocate(go back IP generated)->Edit give name

2. Copy IPaddress->search route 53->dns management->hosted zone->create hosted zone->create record->

Record\_name->

Record\_type->A-routes to Ipv4

Value->IP address(from EC2)->**create records**

**Note:**two record created here for [www.delete.com](http://www.delete.com) and delete.com

1. **Security groups(**allow our instance access to internet so EC2 connected to provide virtual firewall in bound and inbound traffic information**)->**search **EC2->**network n security->security groups->default security groups r there but it allows all traffic access to server poerts its insecure create new

Add security group,description

->add rules->three rules->

Type->HTTP and HTTPs

Source->Anywhere

Type->SSH

Source->MyIP

Provide name to security group

1. **Create RDS** instance->create database

choose DB creation method->standard create

Engine op->MySQL

Version->

Templates->Production

Settings->DB instance identifier->(eg:production-database), auto generate password(tick)

DB instance class->Burstable classes(include t class)->db.t3.micro

Storage->storage type->General purpose(SSD)->allocated storage and storage autoscalling when reach threshold

Availability & durability->do not allow

Public access->no->create DB->instance\_endpoint,master\_user,password

1. **Creating EC2** instance->EC2->**launch instance**->configure instance->auto-assign public IP (enable/disable if don’t want elastic enable it IP address)->configure security->select existing security group/create new as created above in step 2. (selected eg:public website internet access)->launch->eg:hello(created)(Note:download instance pair don’t lost it other not able to join instances)and name instances in edit(helloinstanceweb)

->public ip address**(if IP vaccant)**->so create EC2->network n security->Elastic IP->click on created ip->action->associate elastic ip address->

Instance(selected)->Instance (generated automatically)->associate

1. **Requirement gathering**->git clone

Domain\_name=delete.com

Ip address=3.6.13.176

Instance\_endpoint=production-database.cfacsaa1gwhc.ap-south-1.rds.amazonaws.com

Master\_username=admin

Password=12345678

APP\_NAME=Laravel

APP\_ENV=production

APP\_KEY=

APP\_DEBUG=false

APP\_URL=http://localhost

Note:APP\_DEBUG=false

DB\_CONNECTION=mysql

DB\_HOST=production-database.cfacsaa1gwhc.ap-south-1.rds.amazonaws.com

DB\_PORT=3306

DB\_DATABASE=delete

DB\_USERNAME=admin

DB\_PASSWORD=12345678

Php artisan key:generate

Php artisan migrate

1. **Configure server Laravel project**

**Full laravel Deployment on AWS(Full A-Z)**

1.**ElasticBenstalk**

1. Create Application and Environment in Root user and then Create IAM user

2.**SSH(Secure Shell) ie IAM**

1. **Programmatic access**->id users(selected) and **AWS Management** Console access->eg:uploaded file given access it will give access to see that file and change and AWS management password of AKS will be used->save->attach existing policy directly->search(awscodecommit)-> [AWSCodeCommitPowerUser](https://console.aws.amazon.com/iam/home#/policies/arn%3Aaws%3Aiam%3A%3Aaws%3Apolicy%2FAWSCodeCommitPowerUser)->next->save->download csv file->close
2. Users>ABHISHEK KS>.ssh and gitbash

type> **ssh-keygen**

Enter file in which to save the key (/c/Users/ABHISHEK K S/.ssh/id\_rsa): filename

And other step skip enter

Type >**find . ! –name . –prune –type f –name ‘l\*’**

Type> filename.pub(open in doc) is public key and filename is private key

Copy from filename.pub and paste public key to IAM->users->security credentials->**upload SSH public key** and get SSH public key id(later to be used)

Type>**cat ~/.ssh/lara1307.pub** (will public key)

Signin again as IAM user click on link generated on 5th step because u cannot login as root user for ssh and https following step below or on link(HTTP/SSH/HHTPs(GRC))

Eg: Users with AWS Management Console access can sign-in at: <https://750378451639.signin.aws.amazon.com/console>

Username:lara1307 password:created while creating user

Do following(1st create repository in 3rd step below):

SSH keyID copy in gitbash folder type following

.ssh>**vi config**

Host git-codecommit.\*.amazonaws.com

User APKA25NQKQMN62UOPYLEB

IdentityFile ~/.ssh/awslearn

**>:wq**

**>chmod 600 config**

APKA25NQKQMN62OPYLEB

**3. Initialise codecommit repository**

1. **AWS->CodeCommit**: Devloper tools->codecommit->create repository(click on repository name)

Do following(1st create repository in 3rd step below):

SSH keyID copy in gitbash folder type following

.ssh>**vi config**

Host git-codecommit.\*.amazonaws.com

User APKA25NQKQMNSKQEHZ#LO

IdentityFile ~/.ssh/awslearn

APKA25NQKQMNSKQEH#

ZLO

**>Esc :wq**

**>chmod 600 config**

2. goto command directory of project->

Id = 750378678173

**git init**

**git remote add origin ssh://git-codecommit.ap-south-1.amazonaws.com/v1/repos/lara1307**

**git add .**

**git commit –m “initial commit”**

**git push –u origin main**

**4. Connect repo to Elastic Benstalk with Codepipeline**

1. CodeCommit->Pipeline->Get started->**create** **Pipeline->**select AWSCodepipeline-

>AWSCloudwatch Events(default)

>Deploy->Add Deploy stage->AWS ElasticBenstalk->create Pipeline

**5.** **Configure ElasticBeanstalk for laravel**

ElasticBeanstalk->Environment(link click)

->Configuration->software(Edit)->Document root(**/public**)(as for index.php in laravel public folder)

->Environment Variables->copy from .env file paste in environment variables

APP\_KEY=base64:yexA7iuUjMdu4sAt4qFKarzFh/B+HnZrwJmPQXhiJVk=

->Apply

->This allow us to use file in production in development

Config->app.php->

'key' => env('APP\_KEY'),//comment this use below

'key' => array\_key\_exists('APP\_KEY', $\_SERVER) ? $\_SERVER['APP\_KEY'] : env('APP\_KEY'),

**6. Connect to RDS**

ElasticBeanstalk->Configuration->Database(edit)->

Username and password to be set in laravel .env file->save

Click on Endpoints in RDS->

Error: PDOException::("SQLSTATE[HY000] [2002] A connection attempt failed because the connected party did not properly respond after a period of time, or established connection failed because connected host has failed to respond.)

Sol:RDS->databases->database\_identifier->connectivity & security->security group-rules->EC2 Security Group-inbound(click on link)->Security Groups->inbound rules->edit inbound rules->add rule->

**Type**=MySQL/Aurora

**Source**=Anywhere Or Source=MyIP(if having dedicated IP)

**Save rules**

DB\_CONNECTION=mysql

DB\_HOST=aa179x54eozd98k.cfacsaa1gwhc.ap-south-1.rds.amazonaws.com

DB\_PORT=3306

DB\_DATABASE=ebdb

DB\_USERNAME=lara1307

DB\_PASSWORD=12345678

**php artisan migrate**

Elastic Beanstalk configuration->project name->configuration->software(edit)->Environment Properties->add following

DB\_CONNECTION=mysql

DB\_HOST=aa179x54eozd98k.cfacsaa1gwhc.ap-south-1.rds.amazonaws.com

DB\_PORT=3306

DB\_DATABASE=ebdb

DB\_USERNAME=lara1307

DB\_PASSWORD=12345678

Config->database.php->in mysql->

change from

            'url' => env('DATABASE\_URL'),

            'host' => env('DB\_HOST', '127.0.0.1'),

            'port' => env('DB\_PORT', '3306'),

            'database' => env('DB\_DATABASE', 'forge'),

            'username' => env('DB\_USERNAME', 'forge'),

            'password' => env('DB\_PASSWORD', ''),

To

'url' => array\_key\_exists('DATABASE\_URL', $\_SERVER) ? $\_SERVER['DATABASE\_URL'] : env('DATABASE\_URL'),

'host' => array\_key\_exists('DB\_HOST', $\_SERVER) ? $\_SERVER['DB\_HOST'] : env('DB\_HOST'),

'port' => array\_key\_exists('DB\_PORT', $\_SERVER) ? $\_SERVER['DB\_PORT'] : env('DB\_PORT'),

'database' => array\_key\_exists('DB\_DATABASE', $\_SERVER) ? $\_SERVER['DB\_DATABASE'] : env('DB\_DATABASE'),

'username' => array\_key\_exists('DB\_USERNAME', $\_SERVER) ? $\_SERVER['DB\_USERNAME'] : env('DB\_USERNAME'),

'password' => array\_key\_exists('DB\_PASSWORD', $\_SERVER) ? $\_SERVER['DB\_PASSWORD'] : env('DB\_PASSWORD'),

**7.Nginx setup**

App->public->.htaccess(as server reboot we loose this setup so broken application created under nginx)

Create->.platform(folder)->nginx(folder)->nginx.conf

user                    nginx;

error\_log               /var/log/nginx/error.log warn;

pid                     /var/run/nginx.pid;

worker\_processes        auto;

worker\_rlimit\_nofile    32153;

events {

    worker\_connections  1024;

}

http {

    include       /etc/nginx/mime.types;

    default\_type  application/octet-stream;

    log\_format  main  '$remote\_addr - $remote\_user [$time\_local] "$request" '

                      '$status $body\_bytes\_sent "$http\_referer" '

                      '"$http\_user\_agent" "$http\_x\_forwarded\_for"';

    include       conf.d/\*.conf;

    map $http\_upgrade $connection\_upgrade {

        default     "upgrade";

    }

    server {

        listen        80 default\_server;

        access\_log    /var/log/nginx/access.log main;

        client\_header\_timeout 60;

        client\_body\_timeout   60;

        keepalive\_timeout     60;

        gzip                  off;

        gzip\_comp\_level       4;

        gzip\_types text/plain text/css application/json application/javascript application/x-javascript text/xml application/xml application/xml+rss text/javascript;

        # Do not include the Elastic Beanstalk generated locations

        # include conf.d/elasticbeanstalk/\*.conf;

        # Move Elastic Beanstalk healthd.conf content here

        if ($time\_iso8601 ~ "^(\d{4})-(\d{2})-(\d{2})T(\d{2})") {

            set $year $1;

            set $month $2;

            set $day $3;

            set $hour $4;

        }

        access\_log /var/log/nginx/healthd/application.log.$year-$month-$day-$hour healthd;

        # Move Elastic Beanstalk php.conf content here

        root /var/www/html/public;

        index index.php index.html index.htm;

        # This is an additional configuration

        location / {

            try\_files $uri $uri/ /index.php?$query\_string;

            gzip\_static on;

        }

        location ~ \.(php|phar)(/.\*)?$ {

            fastcgi\_split\_path\_info ^(.+\.(?:php|phar))(/.\*)$;

            fastcgi\_intercept\_errors on;

            fastcgi\_index  index.php;

            fastcgi\_param  QUERY\_STRING       $query\_string;

            fastcgi\_param  REQUEST\_METHOD     $request\_method;

            fastcgi\_param  CONTENT\_TYPE       $content\_type;

            fastcgi\_param  CONTENT\_LENGTH     $content\_length;

            fastcgi\_param  SCRIPT\_NAME        $fastcgi\_script\_name;

            fastcgi\_param  REQUEST\_URI        $request\_uri;

            fastcgi\_param  DOCUMENT\_URI       $document\_uri;

            fastcgi\_param  DOCUMENT\_ROOT      $document\_root;

            fastcgi\_param  SERVER\_PROTOCOL    $server\_protocol;

            fastcgi\_param  REQUEST\_SCHEME     $scheme;

            fastcgi\_param  HTTPS              $https if\_not\_empty;

            fastcgi\_param  GATEWAY\_INTERFACE  CGI/1.1;

            fastcgi\_param  SERVER\_SOFTWARE    nginx/$nginx\_version;

            fastcgi\_param  REMOTE\_ADDR        $remote\_addr;

            fastcgi\_param  REMOTE\_PORT        $remote\_port;

            fastcgi\_param  SERVER\_ADDR        $server\_addr;

            fastcgi\_param  SERVER\_PORT        $server\_port;

            fastcgi\_param  SERVER\_NAME        $server\_name;

            # PHP only, required if PHP was built with --enable-force-cgi-redirect

            fastcgi\_param  REDIRECT\_STATUS    200;

            fastcgi\_param  SCRIPT\_FILENAME  $document\_root$fastcgi\_script\_name;

            fastcgi\_param  PATH\_INFO $fastcgi\_path\_info;

            fastcgi\_pass   php-fpm;

        }

    }

}

**8.Connect to Cloudwatch**

1. To start with new IAM user for log watch following things to be done

maxbanton/cwh github

**composer require maxbanton/cwh:^2.0**

Create IAM user account programmatic account(in same IAM user console) not in AKS

new IAM user->Programmatic access->Attach existing policy->**cloudwatchfullaccess**->

if you prefer to use a separate programmatic IAM user (recommended) or want to define a policy, make sure following permissions are included:

1. CreateLogGroup
2. CreateLogStream
3. PutLogEvents
4. PutRetentionPolicy
5. DescribeLogStreams
6. DescribeLogGroups

After creating cloud watch IAM paste following in .env file

AWS\_ACCESS\_KEY\_ID=AKIA25NQKQMNZQOUXKOO

AWS\_SECRET\_ACCESS\_KEY=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

AWS\_DEFAULT\_REGION=ap-south-1

AWS\_BUCKET=

AKIA25NQKQMN2LKLLWL6E

FJyZw0G/RR84I+LMs2i3a0NvCG6Nulg7hxe7JrYfR

2. config->logging.php

**composer require maxbanton/cwh:^1.0** or ^2.0//only for laravel 8 below it, add manually

add logging.php after errorlog

 'null' => [

        'driver' => 'monolog',

        'handler' => NullHandler::class,

    ],

    'emergency' => [

        'path' => storage\_path('logs/laravel.log'),

    ],

    'cloudwatch' => [

      'driver' => 'custom',

      'via' => \App\Logging\CloudWatchLoggerFactory::class,

      'sdk' => [

        'region' => array\_key\_exists('AWS\_DEFAULT\_REGION', $\_SERVER) ? $\_SERVER['AWS\_DEFAULT\_REGION'] : env('AWS\_DEFAULT\_REGION', 'ap-south-1'),

        'version' => 'latest',

        'credentials' => [

          'key' => array\_key\_exists('AWS\_ACCESS\_KEY\_ID', $\_SERVER) ? $\_SERVER['AWS\_ACCESS\_KEY\_ID'] : env('AWS\_ACCESS\_KEY\_ID'),

          'secret' => array\_key\_exists('AWS\_SECRET\_ACCESS\_KEY', $\_SERVER) ? $\_SERVER['AWS\_SECRET\_ACCESS\_KEY'] : env('AWS\_SECRET\_ACCESS\_KEY'),

        ]

      ],

      'retention' => env('CLOUDWATCH\_LOG\_RETENTION', 30),

      'level' => env('CLOUDWATCH\_LOG\_LEVEL', 'info')

    ],

],

3..env

LOG\_CHANNEL=stack

To

LOG\_CHANNEL=cloudwatch

After creating cloudwatch IAM paste following in .env file

AWS\_ACCESS\_KEY\_ID=AKIA25NQKQMN2LKLWL6E

AWS\_SECRET\_ACCESS\_KEY=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

AWS\_DEFAULT\_REGION=ap-south-1

AWS\_BUCKET=

Note:log IAM is same IAM user

4. create folder in App->Logging(folder)->CloudWatchLoggerFactory.php

<?php

namespace App\Logging;

use Monolog\Logger;

use Maxbanton\Cwh\Handler\CloudWatch;

use Aws\CloudWatchLogs\CloudWatchLogsClient;

class CloudWatchLoggerFactory

{

    /\*\*

     \* Create a custom Monolog instance.

     \*

     \* @param  array  $config

     \* @return \Monolog\Logger

     \*/

    public function \_\_invoke(array $config)

    {

        $sdkParams = $config["sdk"];

        $tags = $config["tags"] ?? [ ];

        $name = $config["name"] ?? 'cloudwatch';

        // Instantiate AWS SDK CloudWatch Logs Client

        $client = new CloudWatchLogsClient($sdkParams);

        // Log group name, will be created if none

        $groupName = config('app.name') . '-' . config('app.env');

        // Log stream name, will be created if none

        $streamName = config('app.name');

        // Days to keep logs, 14 by default. Set to `null` to allow indefinite retention.

        $retentionDays = $config["retention"];

        // Instantiate handler (tags are optional)

        $handler = new CloudWatch($client, $groupName, $streamName, $retentionDays, 10000, $tags);

        // Create a log channel

        $logger = new Logger($name);

        // Set handler

        $logger->pushHandler($handler);

        return $logger;

    }

}

5.register route

Route::get('/log',function(){

    Log::info('Hey');

    return view('welcome');

});

Php artisan serve

6.IAM user and then **AWS**->**CloudWatch** console->log->logsGroups(Laravel-local->App\_name)

7.goto main IAM user of project ElasticBeanstalk->configuration->software(edit)->Environment properties->add

APP\_ENV=local/production

APP\_KEY=base64:yexA7iuUjMdu4sAt4qFKarzFh/B+HnZrwJmPQXhiJVk=

APP\_DEBUG=false

LOG\_CHANNEL=cloudwatch

AWS\_ACCESS\_KEY\_ID=AKIA25NQKQMN2LKLWL6E

AWS\_SECRET\_ACCESS\_KEY=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

AWS\_DEFAULT\_REGION=ap-south-1

DB\_HOST=aa179x54eozd98k.cfacsaa1gwhc.ap-south-1.rds.amazonaws.com

DB\_PORT=3306

DB\_DATABASE=ebdb

DB\_USERNAME=lara1307

DB\_PASSWORD=12345678

Note:**local** is for local server and production is for **production**(eg:AWS) server

and **debug** be false in production

8. config->logging.php->

    'default' => env('LOG\_CHANNEL', 'stack'),

To

|  |
| --- |
|  |
| 'default' => array\_key\_exists('LOG\_CHANNEL', $\_SERVER) ? $\_SERVER['LOG\_CHANNEL'] : env('LOG\_CHANNEL', 'cloudwatch'), |
|  |

**9.Route R53 register(DNS)**

As default DNS created by ElasticBeanstalk(Environment) first get domain name from outside source or create in AWS

1.AWS->**R53** console->Domain->search for domain and create->

Checkbox->Automatically renew domain(checked) and other term n condition after this check records in hosted zones. And goto namecheap.com get domain

**Or**

2.AWS->**R53** console->Hosted zones->Create Hosted zones->Create Record(automatically generated no change required)->create record(only if autogenerated record not available).

Search for domain

3.AWS->**Certificate Manager** console->**Request Certificate**(Provision certificates**)**->Request Certificate

->Add domain Eg:www.lara1307.com, lara1307.com, \*.lara1307.com

->validation method->DNS

->Tags->skip

->Review

->Validation->Domain (dropdownarrow)->**Create Record in Route in 53**->**CNAME** will be created in record(directly to domain otherwise it has to be copy n paste)

Check->R53->Hosted zones->eg:lara1307.com->below it have created

**Note**:Certificate issuing may take upto 30 minutes

4. AWS->**EBS**->eg:lara1307->config->**Load balancer**(edit)->Listeners(add listeners)->

->port->443(HTTPS)

->HTTPS

->SSL\_certificate

->Default->default->add

->apply

----Pending-------

5. AWS->**EC2** Console->Load Balancer->Create(auto generated click any or create/EC2 instance creation –Pending---)

->Listeners->

->HTTP:80->view/edit rules

->edit(pencil)->pencil mark

->delete->Then->RedirectTo->following below

->HTTPS::443(port)

->RedirectTo->HTTPS,Original Hostpath,301->click(tick)->update

->Copy name of Load Balancer And goto R53 below step 6

6.AWS->R53->Hosted Zone name(click)->create record->Alias

->Alias to Application and classic Load Balancer or paste Load balancer name, Region->paste Load balancer name(---Pending---)

**10.AWS->SES**

1. **SMTP Settings->Create my SMTP Credentials**->Get **SMTP** username and password(change default IAM user to your choice eg:**awslearnses** then click dropdown)->create
2. **Verifyemail addresses(**add email address side bar**)**->from which to send email

3. SMTP Settings->get port number, host(going back after creating user),tls

MAIL\_DRIVER=smtp

MAIL\_HOST=email-smtp.ap-south-1.amazonaws.com

MAIL\_PORT=465

MAIL\_USERNAME=AKIA25NQKQ#MNZ5ZFOZWY

MAIL\_PASSWORD=BEDAIrUw0dAoZo5d8pxiGvwhBrhuK0/2Zn78hDl4bQu7

MAIL\_ENCRYPTION=tls

MAIL\_FROM\_ADDRESS=[abhi144k@gmail.com](mailto:abhi144k@gmail.com)

MAIL\_FROM\_NAME= 'Abhishek'

4.**Domains**->**verify** **domain**->generate DKIM settings(checked)->email receiving records(checked)->hosted zones(checked)->Use Route 53

5. **php artisan make:mail Testmail** and **views**->**mail.blade.php**

Route::get('/mail', function(){

    Mail::to('abhi144k@gmail.com')->send(new TestMail());

    return view('welcome');

});

6.ElasticBeanstalk->lara1307->configuration->Environment properties->add

MAIL\_DRIVER=smtp

MAIL\_HOST=email-smtp.ap-south-1.amazonaws.com

MAIL\_PORT=587

MAIL\_USERNAME=AKIA25NQKQMNQZR2HEFW

MAIL\_PASSWORD=BK074gkHKvrSNE62+GIw390Ht+AnCoO8la3iqtc3OhyB

MAIL\_ENCRYPTION=tls

MAIL\_FROM\_ADDRESS=abhi144k@gmail.com

MAIL\_FROM\_NAME='Abhishek'

7.config->mail.php->add on top after return

'default' => array\_key\_exists('MAIL\_MAILER', $\_SERVER) ? $\_SERVER['MAIL\_MAILER'] : env('MAIL\_MAILER'),

    'mailers' => [

        'smtp' => [

           'transport' => 'smtp',

           'host' => array\_key\_exists('MAIL\_HOST', $\_SERVER) ? $\_SERVER['MAIL\_HOST'] : env('MAIL\_HOST'),

           'port' => array\_key\_exists('MAIL\_PORT', $\_SERVER) ? $\_SERVER['MAIL\_PORT'] : env('MAIL\_PORT'),

           'encryption' => array\_key\_exists('MAIL\_ENCRYPTION', $\_SERVER) ? $\_SERVER['MAIL\_ENCRYPTION'] : env('MAIL\_ENCRYPTION'),

            'username' => array\_key\_exists('MAIL\_USERNAME', $\_SERVER) ? $\_SERVER['MAIL\_USERNAME'] : env('MAIL\_USERNAME'),

            'password' => array\_key\_exists('MAIL\_PASSWORD', $\_SERVER) ? $\_SERVER['MAIL\_PASSWORD'] : env('MAIL\_PASSWORD'),

            'timeout' => null,

            'auth\_mode' => null,

        ],

        'ses' => [

            'transport' => 'ses',

        ],

        'mailgun' => [

            'transport' => 'mailgun',

        ],

        'postmark' => [

            'transport' => 'postmark',

        ],

        'sendmail' => [

            'transport' => 'sendmail',

            'path' => '/usr/sbin/sendmail -bs',

        ],

        'log' => [

            'transport' => 'log',

            'channel' => env('MAIL\_LOG\_CHANNEL'),

        ],

        'array' => [

            'transport' => 'array',

        ],

    ],

8. **git push** and **php artisan serve**

**11. Backup DB on S3**

1. Use Spatie/installation-and-setup Laravel 5.8 and above

**composer require spatie/laravel-backup**

**php artisan vendor:publish –provider="Spatie\BackupServiceProvider"**

config->app.php->add

**Spatie\Backup\BackupServiceProvider::class,**

**php artisan backup:run --help**

**php artisan backup:run**

**php artisan backup:run --only-db**

check for DB file storage**->app->laravel**

config->database.php

 'mysql' => [

            'driver' => 'mysql',

            'url' => array\_key\_exists('DATABASE\_URL', $\_SERVER) ? $\_SERVER['DATABASE\_URL'] : env('DATABASE\_URL'),

            'host' => array\_key\_exists('DB\_HOST', $\_SERVER) ? $\_SERVER['DB\_HOST'] : env('DB\_HOST', '127.0.0.1'),

            'port' => array\_key\_exists('DB\_PORT', $\_SERVER) ? $\_SERVER['DB\_PORT'] : env('DB\_PORT', '3306'),

            'database' => array\_key\_exists('DB\_DATABASE', $\_SERVER) ? $\_SERVER['DB\_DATABASE'] : env('DB\_DATABASE', 'forge'),

            'username' => array\_key\_exists('DB\_USERNAME', $\_SERVER) ? $\_SERVER['DB\_USERNAME'] : env('DB\_USERNAME', 'forge'),

            'password' => array\_key\_exists('DB\_PASSWORD', $\_SERVER) ? $\_SERVER['DB\_PASSWORD'] : env('DB\_PASSWORD', ''),

            'unix\_socket' => env('DB\_SOCKET', ''),

            'charset' => 'utf8mb4',

            'collation' => 'utf8mb4\_unicode\_ci',

            'prefix' => '',

            'prefix\_indexes' => true,

            'strict' => true,

            'engine' => null,

            'dump' => [

                'dump\_binary\_path' => 'C:\wamp64\bin\mysql\mysql5.7.26\bin'

               //'dump\_binary\_path' => env('DB\_MYSQLDUMP\_PATH'),

                //'dump\_binary\_path' => '/path/to/the/binary',//C:\wamp64\bin\mysql\mysql5.7.26\bin

                //'dump\_binary\_path' => 'C:\wamp64\bin\mysql\mysql5.7.26\bin', // only the path, so without `mysqldump` or `pg\_dump`

               //'use\_single\_transaction',

                //'timeout' => 60 \* 5, // 5 minute timeout

            ],

            'options' => extension\_loaded('pdo\_mysql') ? array\_filter([

                PDO::MYSQL\_ATTR\_SSL\_CA => env('MYSQL\_ATTR\_SSL\_CA'),

            ]) : [],

        ],

**Error:** mysqldump error will be solved

            'dump' => [

                'dump\_binary\_path' => env('DB\_MYSQLDUMP\_PATH'),

               // 'dump\_binary\_path' => 'C:\wamp64\bin\mysql\mysql5.7.26\bin/', // only the path, so without `mysqldump` or `pg\_dump`

               'use\_single\_transaction',

                'timeout' => 60 \* 5, // 5 minute timeout

            ],

Error: if mismatch following error will occur keep repeating installation,

"league/flysystem-aws-s3-v3": "~1.0",

        "maxbanton/cwh": "^1.0",

        "spatie/db-dumper": "^2.21",

        "spatie/laravel-backup": "^6.11"

public accessable in database instance enabled error will resolve

2. **composer require league/flysystem-aws-s3-v3:~1.0**

This will connect to S3 to store all data on s3 instead on EC2, so its necessary to store all data separately DB will not lost.

3. This S3 will be accessable by **IAM** users(use same IAM used in SES) eg:**awslearnses** same IAM user or create different and give it permission->Attachexisting policies->**amazonS3fullaccess**->give permission

4.Goto->S3 console->create **Bucket(**awslearnbkt**)**

>Block all public access

>Bucket versioning enable

>Default Encryption enable type(select any one)

5.Goto .env add bucket name

AWS\_BUCKET= awslearnbkt

6.config->backup.php->under destination=[] change

'disks' => ['local'],

To

'disks' => ['s3',],

'monitor\_backups' => ['disks' => ['s3'], ]

7. **php artisan backup:run --only-db** and cache and config clear regularly

8.Goto ElasticBeanstalk->config->Environment Properties->add Bucket name

AWS\_BUCKET= awslearnbkt

9.config->filesystem.php->

 'driver' => 's3',

'key' => array\_key\_exists('s3\_ACCESS\_KEY\_ID', $\_SERVER) ? $\_SERVER['s3\_ACCESS\_KEY\_ID'] : env('s3\_ACCESS\_KEY\_ID'),

'secret' => array\_key\_exists('s3\_SECRET\_ACCESS\_KEY', $\_SERVER) ? $\_SERVER['s3\_SECRET\_ACCESS\_KEY'] : env('s3\_SECRET\_ACCESS\_KEY'),

'region' => array\_key\_exists('s3\_DEFAULT\_REGION', $\_SERVER) ? $\_SERVER['s3\_DEFAULT\_REGION'] : env('s3\_DEFAULT\_REGION'),

'bucket' => array\_key\_exists('s3\_BUCKET', $\_SERVER) ? $\_SERVER['s3\_BUCKET'] : env('s3\_BUCKET'),

'url' => array\_key\_exists('s3\_URL', $\_SERVER) ? $\_SERVER['s3\_URL'] : env('s3\_URL'),

'endpoint' => array\_key\_exists('s3\_ENDPOINT', $\_SERVER) ? $\_SERVER['s3\_ENDPOINT'] : env('s3\_ENDPOINT'),

10..env

s3\_ACCESS\_KEY\_ID=AKIA825NQKQMN3WDVZOFL

s3\_SECRET\_ACCESS\_KEY=4dLvHyL8MK1JAsJyOJk7SU36M47Tr+R8m9CGOdAM

s3\_DEFAULT\_REGION=ap-south-1

s3\_BUCKET=awslearnbkt

11. Create **IAM** user separately for **programmatic** **access**->Add inline **policy** to IAM users

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": "s3:\*",

"Resource": [

"arn:aws:s3:::your-bucket-name",

"arn:aws:s3:::your-bucket-name/\*"

],

"Condition": {}

}

]

}

13. config->filesystem.php

 'mail' => [

            'to' => '',//abhi144k@gmail.com',

            'from' => [

                'address' => env('MAIL\_FROM\_ADDRESS', 'hello@example.com'),

                'name' => env('MAIL\_FROM\_NAME', 'Example'),

            ],

        ],

**12. Crons**

1. Do backup every interval of time or every day crons used

2. **AWS->Documentation->AWS EBS->Developer Guide**

Create **.ebextensions(folder)/cronjob.config(file)**

//-----start-----

files:

"/etc/cron.d/schedule\_run":

mode: "000644"

owner: root

group: root

content: |

\* \* \* \* \* root . /opt/elasticbeanstalk/support/envvars && /usr/bin/php /var/www/html/artisan schedule:run 1>> /dev/null 2>&1

commands:

remove\_old\_cron:

command: "rm -f /etc/cron.d/\*.bak"

//-----end-----

3.console->kernel.php->add

protected function schedule(Schedule $schedule)

    {

        // $schedule->command('inspire')->hourly();

        $schedule->command('backup:run --only-db')->dailyAt('4:00');

    }

**13. SMS**

1. **composer require aws/aws-sdk-php-laravel**

2. **php artisan vendor:publish --provider="Aws\Laravel\AwsServiceProvider"**

AKIA25NQKQMNR5FZC7NJ

WHD5iqG8JyqIpbL361mMx9dBJv4wx7u5RqE+KTkW