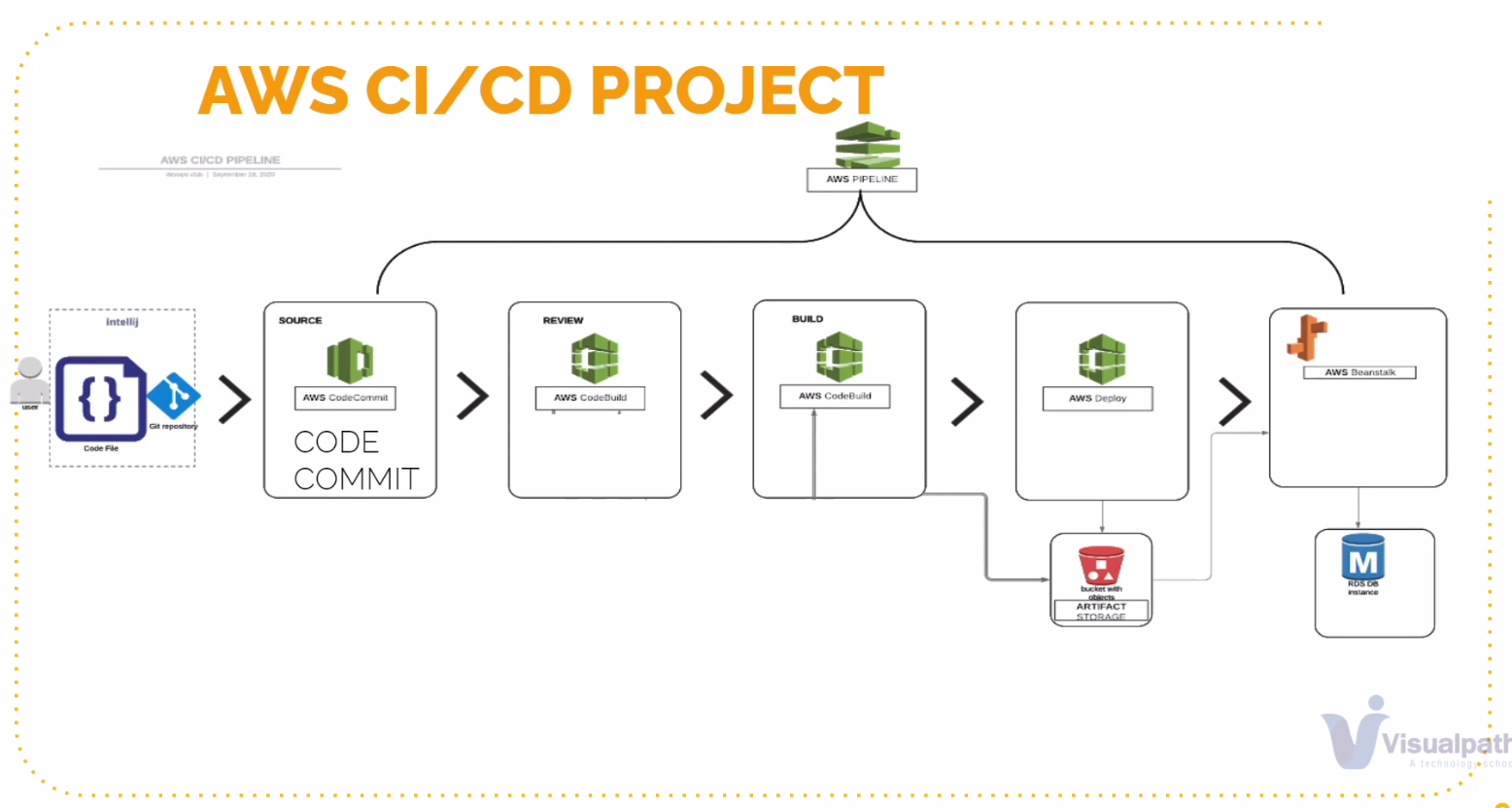
Beanstalk CI/CD Project



Beanstalk. Platform as service. No need to install EC2 instances, Load Balancers, monitoring, auto scaling. For Application.

RDS. Platform as service. For Database.

EC2. Infrastructure as service. Need build everything VPC setup, auto scaling, volumes, LB.

Just upload code to Beanstalk.BS automatically handles deployment, auto scaling, load balancing, EC2 instances, health monitoring. No additional charge for BS, pay only for the AWS resources needed to store and run your applications. Just create artefact, upload and run it. Service for Deploying & Scaling applications deployed with Java, PHP, Docker, and python, etc.

Application (project name). Environment. Dev, QA, Staging, Prod.

Environment. Instances, LB, Auto scaling, S3 buckets, Cloud watch alarm.

Create application (vprofile-app)

Platform.

Tomcat

Software. Proxy services. Apache/Nginx. In front of tomcat.

S3.To save logs/archives. Not free of use. Requires additional charge.

Cloud watch. Stream logs.

Instances. SSD=8GB. Security groups (web-sg), allow SSH from my IP.

EC2 instances. Read only. No change inside. They just replaced.

Capacity. Load balanced (in Prod). Min 2, Max 4.

Availability zones. To provide high availability. Distributed among multiple zones.

Scaling triggers. For Auto scaling. CPU utilization, Network out (web apps)

Load Balancer. App LB. Health check /login. Ports 80 or 8080.

Access logs. To store in S3 bucket.

Application Deployments (Rolling updates and deployments)

All at once. All instances upgrades at once/together. Will be down time during upgrade.

If 2 instances. Upgrades both together. So will be down time.

**Rolling**. For **production**. Extra charge from extra instances.

If 2 Instance, Batch size = 50%. 1 instance at a time. Upgrades one by one.

If 4 Instance, Batch size = 25%. 1 instance at a time. Upgrades one by one.

Rolling with additional batch.

If 2 Instances. Batch size = 50%. Enough Capacity.

Launches 3rd instance (new one), upgrades it & deletes 1st old one.

Launches 4th instance (new one), upgrades it & deletes 2nd old one.

Immutable. The most safest, fastest & expensive one.

If 10 Instances. Launches, 10-upgraded Instances & routes traffic there. If all is fine, deletes old 10 ones.

Traffic splitting. New one. For Bleu/green deployment.

Traffic split = 10%. Time = 5 min.

Every 5 min 10% traffic goes to new instances. 90% on old, 10% on new. If issues, effects only 10% users.

After 5 min, if all ok. 80% old, 20% new. After 5min, 70% old, 30% new … Till 100% on new instances.

Can also managed by Route53.Swap option of BS.

Security. Key-pair for EC2 instances (vpro-bean-key).

Managed Updates

Monitoring.

Notifications (email address)

Network. VPC, subnets, zones.

Database RDS. Not recommend for Production (Make RDS separate. Integrate RDS with BS). If deletes Beanstalk, RDS will be deleted.

VPRO-BS-APP

LB web access link

Configuration (software, instance, capacity, Load Balance, Rolling updates and deployments, security, monitoring, updates, notification, NET, DB)

Logs

Health

Monitoring

Alarms

Managed updates

Events

RDS. App setup

Create RDS. MySQL (v5.6.34), free tier. Auto generate-pass

Create Sec.Group. Additional configuration. Add accounts DB.

Store credentials

admin

Only4123123RDS

EC2 Security groups. Allow SSH from my IP.

On RDS SG, allow 3306 from EC2 instances SG.

Connect EC2.

Install Git. Git clone <https://github.com/devopshydclub/vprofile-project.git>

Install mysql

Test connectuon to RDS

mysql -h vprofile-rds.ckhqvjcx3xzq.us-east-1.rds.amazonaws.com -u admin -pOnly4123123RDS

Upload SQL-Conf to RDS

mysql -h vprofile-rds.ckhqvjcx3xzq.us-east-1.rds.amazonaws.com -u admin -pOnly4123123RDS accounts < src/main/resources/db\_backup.sql

Locally. Git clone. Update SQL configuration file (SQL link). MVN install. Make artefacts.

Bean Stalk. Upload artefact to Beanstalk. Deploy artefact.

Fix errors

Load Balancer. Health check. Path (/login). Tick Stickiness policy. Save. Apply

**08012023**

AWS commit version control system

Code Commit. Create repository (vpro-repo). Code Gury like code analyses.

Use SSH to connect.

Create policy (vpro-repo-policy) for access repository (vpro-repo)

Create IAM user (vpro-repo-admin) and attach policy (vpro-repo-policy). Upload SSH pub key.

Locally. Generate SSH keys. Create SSH config file. Chmod 600 config. Connect to repo via SSH.

<https://docs.aws.amazon.com/codecommit/latest/userguide/setting-up-ssh-unixes.html>

git branch -a | grep -v HEAD | cut -d '/' -f3 | grep -v master > ~/temp/branches

for i in `cat ~/temp/branches`; do echo $i;done

Jorge method.

Birinshi ozim tusinip alowim kk. Teach SMB with simple words. Interview process.

CODE BUILD (Artefacts coped to S3 bucket)

Create build (like MVN)

Communicate with RDS. Insert build command. SQL Application.conf file.

Cloud watch (to see logs if failure happens)

BUILD. DEPLOY (Artefacts sent to Beanstalk)

Create pipeline (like Jenkins). Use Cloud watch.

Cleaning. Delete RDS, SG, Beanstalk (Env, App)