

Order of SETUP:

1. AWS Beanstalk
2. RDS DB Instance
3. AWS CI / CD Pipeline that will continuously deploying artifact to the Beanstalk environment

**AWS Beanstalk:**

This is a platform as a service for running applications workload easily.You don’t need to setup EC2 instances or Load Balancer or Autoscaling groups. You can get your application up and running within minuets

STEPS:

1. Login to your AWS and go to services
2. search for Elastic Beanstalk and click on create Beanstalk
3. Application Name = vprofile-app
4. scroll down to platform section
5. platform = Tomcat (all other options should be default values)
6. scroll down to Application code section
7. check sample application
8. click on configure more options (work with all the sections professionally like below)
9. click on edit under software box
   1. proxy server = Nginx
   2. click to enable S3 log storage (You can unselect it becos of the charges)
   3. Scroll down to Instance log streaming to CloudWatch Logs section
   4. Click to enable Log streaming (You can unselect it becos of the charges)
   5. click on save
10. click on edit under Instance box
11. Root volume type = General purpose SSD
12. Size = 8 GB
13. scroll down to EC2 Security group
14. EC2 Security Group = create one and return back here to select it, or if you don’t select it will create its’ own security group by default
15. click on save
16. click on edit under Capacity
17. Environment type = Load balanced (This is for production, as single instance isn’t enough)
18. Instances = 2min & 8max
19. scroll down to instance type = t2.micro (But if it is production, select a bigger type)
20. AMI ID = leave the default (change only if you know how to create AMI for Beanstalk, check the documentation guide)
21. Availability Zones = select any
22. placement = select all the availability zones showing at the same time
23. Scaling triggers section(metric = NetworkOut) ….. very good for web pages, it helps in tracking the load
24. scroll down and click save
25. click on edit under Load Balancer section
26. Load Balancer type = Application Load Balancer
27. Processes = check the default disabled port 80

click on actions and click on edit

scroll down to Path = /login

scroll down and enable Stickiness policy

1. Access log files = It will be changed later
2. scroll down and click on save
3. click on edit under Rolling updates and deployment section
4. Deployment policy = Rolling (Good for Production to prevent downtime especially during updates)
5. Batch size = 25% (4 instance = 25%, 1 instance out of 2 at a time = 50%, at a time to be shutdown for upgrade)
6. traffic split = 10, traffic splitting evaluation time = 5mins
7. leave everything as defaults, scroll down and click on save
8. click on edit under Security section
9. EC2 key pair = create key pair, return back here and select it
10. IAM instance profile = select the IAM role created (aws-elasticbeanstalk-ec2-role)
11. scroll down and click on ok
12. Scroll down and click on create App after you are done configuring settings professionally

**RDS & APP Setup on Beanstalk:**

STEPS:

1. click on services and search for RDS
2. click on create database, check standard create
3. click on MySQL
4. version = MySQL 5.6.34 (it supports free tier)
5. Templates = Free Tier
6. scroll down to settings section
7. DB instance identifier = vprofile-bean-rds
8. Master username = admin
9. check Auto generate a password
10. scroll down to DB instance size
11. check Burstable classes, select db.t2.micro from the drop down
12. scroll down, leave all default settings for free tier
13. click on create a new security group
14. VPC S.G name = vprofile-bean-rds-sg
15. scroll down to Additional configuration
16. initial database name = Accounts
17. leave the defaults and click on Create database
18. in the next page, click on View credential details. Top right corner
19. And the copy the info and paste in MS word to save
20. wait for a moment for it to become healthy
21. Go to Beanstalk and check the Health state if it is ok
22. click on the Beanstalk Environment name to check the health state
23. copy the application access link above and paste in the word file to keep for accessing
24. Go to the EC2 under service, you should see two instances that the Beanstalk has created
25. make sure you attached a keypair to it so as you can do SSH
26. select one of the Instance, scroll down and click on Security
27. scroll down and click on the security groups link to correct the wrong rule
28. HTTP / TCP / 80 / AWS EBLoadBalancer
29. SSH / TCP / 22 / myIP
30. copy the ID of the security group of Beanstalk instance to Word file
31. click on save rules
32. Go to the RDS security group and edit the Inbound rules
33. MySQL/Aurora / TCP / 3306 / paste the ID of the security group of Beanstalk instance from word file / Allow Beanstalk instance to connect RDS on 3306
34. click on save rule
35. copy the public IP of the Beanstalk Instance to SSH
36. ssh –i Downloads/vpro-bean-key.pem ec2-user@paste the public IP
37. sudo –i
38. yum install git mysql –y
39. git clone <https://github.com/devopshydclub/vprofile-repo.git> (To clone the source code)
40. cd vprofile-repo/
41. ls
42. git checkout vp-rem
43. ls src/main/resources/db\_backup.sql
44. mysql –h (copy & paste the end of the RDS here) –u admin –p (paste the password of the RDS from the word file) accounts
45. if you are logged in it means it is working
46. Bye
47. mysql –h (copy & paste the end of the RDS here) –u admin –p (paste the password of the RDS from the word file) accounts < src/main/resources/db\_backup.sql
48. logout
49. logout
50. cd /tmp/
51. mkdir beanapp
52. cd beanapp/
53. git clone <https://github.com/devopshydclub/vprofile-repo.git>
54. cd vprofile-repo/
55. ls
56. git checkout vp-rem
57. vim src/main/resources/application.properties
58. Edit the file
59. change:

jdbc.url = db01 to the endpoint of the RDS

jdbc.username = type the RDS username from word

jdbc.password = type the RDS password from word

provide the IP and port of all other services

:wq

1. mvn install
2. cp target/vprofile-v2.war ~/Desktop/
3. Go to your Beanstalk and click on Environment
4. click on VprofileApp-env
5. click on application versions on the left pane
6. you should see the sample Application
7. click on upload

version label = vprofile-app-test

click on choose file button (vprofile-v2.war), you should find it on the Desktop (line 61 saved it there)

1. click on Upload
2. you should see it under the Application Versions
3. check on it (vprofile-app-test) and click on Actions
4. Then click on Deploy
5. Go to Environments and click on the link to view in a browser if the Artifact was deployed

AWS CI / CD Pipeline that will continuously deploying artifact to the Beanstalk environment

**CODE COMMIT:**

AWS Code Commit is a replacement of Github. It is a version control service hosted by Amazon Web Services that you can use to privately store and manage assets (such as documents, source code, and binary files) in the cloud.

Git is a distributed version control system for tracking changes in source code during software development.

STEPS:

1. Make sure your Region is N.Virginia. Becos some regions don’t have Code Artifacts on the left pane
2. Click on services and search for Code Commit
3. Click on AWS Developer Tools
4. On the left pane, click on CodeCommit, then click on Repositories
5. Create Repositories:

* Repository name: vprofile-code-repo
* check Enable Amazon CodeGuru
* click on create

1. Create an IAM User with codecommit rights

* Go to services and search for IAM
* click on Add user
* user name: vprofile-code-admin
* click on programmatic access
* click on Attach policy access
* click on create policy (We only need permission for the repository we created)
* Under Service = search for Codecommit
* click to check All codecommit actions
* Under Resources = check specific
* click on Add ARN

Region = your region

Repository name = vprofile-code-repo

click on Add

click on review Policy

Name = vprofile-code-admin-repo-fullaccess

click on create policy

* Return back to Attach existing policies
* search for the created policy & check it (vprofile-code-admin-repo-fullaccess)
* click on next
* click on create user
* Download the .csv file and click on close
* Your new IAM user would show, click on it (vprofile-code-admin)
* click on the Security credentials Tab, scroll down and Delete the Access key (we don’t need it)
* scroll down and click on Upload SSH public key
* open GitBash
* ssh-keygen.exe
* /c/Users/imran/.ssh/vpro-codecommit\_rsa
* cd .ssh
* ls
* cat vpro-codecommit\_rsa.pub (Copy the key shown)
* Return back to your AWS, and paste the public when you click on Upload SSH public key
* click on Upload SSH public key, copy the SSH key ID
* return back to GitBash
* pwd (/c/Users/imran/.ssh)
* vim config

Host git-codecommit.\*.amazonaws.com

User (paste the SSH key ID)

IdentityFile ~/.ssh/ vpro-codecommit\_rsa

:wq

1. ls –l
2. chmod 600 config
3. cd
4. SSH to the code commit service

ssh –v git-codecommit.us-east-2.amazonaws.com

1. cd /tmp/
2. Go to CodeCommit, click on repositories and then click on your respository
3. click on the SSH Tab and click Clone URL
4. Go to GitBash
5. git clone ssh://git-codecommit.us-east-1.amazonaws.com/vl/repos/vprofile-code-repo
6. cd vprofile-code-repo/
7. ls
8. cd /f/
9. git clone <https://github.com/devopshydclub/vprofile-project.git>
10. cd vprofile-project/
11. ls
12. cat .git/config
13. git branch –a (check the branch you want to push to code commit)
14. git checkout master
15. git branch –a | grep –v HEAD | cut –d ‘ /’ –f3 | grep –v master > /tmp/branches
16. cat /tmp/branches
17. for i in ‘cat /tmp/branches’: do echo $i:done
18. for i in ‘cat /tmp/branches’: do git checkout $i:done
19. git branch –a
20. git fetch - -tags
21. git remote rm origin
22. git remote add origin (the SSH url of your CodeCommit repository )
23. cat .git/config
24. git push origin - -all
25. git push - -tags
26. Go to the CodeCommit repository to see the changes

**CODE BUILD:**

AWS CodeBuild is a fully managed build service that compiles source code, runs tests, and produces software packages that are ready to deploy. It eliminates the need for you to set up, configure, and scale your own build servers.

With CodeBuild, you can build code in nearly any programming language, including Java, Python, Ruby, Node.js, and Go. You can also use CodeBuild to build Docker images, and it integrates with other AWS services such as CodeCommit, CodePipeline, and CodeDeploy.

To use CodeBuild, you define a build specification in a YAML file, which specifies the build environment, the source code location, the build commands, and the output artifacts. When you start a build, CodeBuild creates a Docker container based on the specified environment and executes the build commands. You can monitor the progress of the build in the CodeBuild console or through AWS APIs.

CodeBuild is a scalable and flexible solution that can handle builds of any size and complexity. It also supports custom build environments, which allow you to use your own Docker images to build your code. CodeBuild is pay-as-you-go and charges based on the duration of your builds and the amount of compute and storage resources used.

STEPS:

1. Go to Developer Tools and click on CodeCommit
2. Click on Repositories and click on your created Repository name
3. click on the second drop down to change branch to vp-rem
4. click on src and click on main
5. click on resources and then click on application.properties
6. Take note of the codes as you will need them to edit codes in step 8 line 14 using VS code editor
7. you can find these codes in the folder (buildspec.yaml)
8. Use the YAML file in the folder to edit
9. Use info in step 6 above to edit
10. Line 22: Please note that the RDS password would be placed after the admin123
11. Line 24: Please note that the RDS endpoint would be placed after the 3306/
12. save the Buildspec.yaml
13. Search for S3Bucket and create an S3 in the same region as your project
14. Still in your AWS Developer tools, click on Code Build
15. click on Get started and click on Create project

* Project name: vprofile-Build
* Source provider = AWS CodeCommit
* Repository = vprofile-code-repo
* Branch = vp-rem
* Environment Image = Managed image
* Operating system = Ubuntu
* Runtime = standard
* image = aws/codebuild/standard:3.0
* Image version = Always use the latest image for this runtime version
* Environment type = Linux
* service role = New service role
* Role name = codebuild-vprofile-Build-service-role
* Click on Additional Configuration
* VPC = you can select your VPC
* Build specifications = insert build commands (click on switch editor …. to edit the config file)
* paste the codes in the Buildspec.yaml from the folder
* Scroll down to Artifacts section
* Type = Amazon S3
* Bucket name = select your created Bucket make sure it is in the same region
* Artifact packaging = None
* Under Log section, click to check CloudWatch Logs
* Group name = vprofile-cicd-project
* stream name = buildlogs
* check S3 logs
* click on Create build project
* If it gives you an error that a role or policy already exist, simply delete the role or policy or the best option is to change the name in role name above and click on create build project
* Once it is created, click on Start Build

**BUILD, DEPLOY & CODE PIPELINE**

AWS CodePipeline is a fully managed continuous delivery service that helps you automate the release of software updates. It enables you to model, visualize, and automate the steps required to release your software, from building and testing to deploying and updating.

With CodePipeline, you create a pipeline that defines the workflow for your software release process. You can use pre-built integrations with other AWS services such as CodeCommit, CodeBuild, CodeDeploy, and Elastic Beanstalk to define the stages in your pipeline, or you can create your own custom integrations with third-party tools.

CodePipeline automates the build, test, and deploy phases of your software release process, and provides real-time feedback on the status of each stage. You can view the progress of your pipeline in the CodePipeline console or through AWS APIs.

CodePipeline is a flexible and customizable solution that supports a variety of deployment scenarios, including rolling updates, canary deployments, and blue-green deployments. It also integrates with AWS Identity and Access Management (IAM) to provide fine-grained access control for your pipeline resources.

CodePipeline is a pay-as-you-go service that charges based on the number of active pipelines, the duration of your pipeline executions, and the number of pipeline actions.

**STEPS:**

* Once it is created, click on Start Build
* Wait a little bit for a successful Build
* It is time to Deploy, Under Developer Tools
* click on Pipeline on the Left pane and click on Getting Started
* click on create Pipeline
* Pipeline name = vprofile-cicd-pipeline
* Service role = New service role
* Role name = AWSCodePipelineServiceRole-us-west-2-vprofile-cicd-pipeline
* click on Advanced settings
* artifact store = custom location
* Bucket = select your s3 Bucket making sure it is in the same region
* Encryption key = Default AWS Managed key
* click on next
* source provider = AWS CodeCommit
* Repository name = select your repo name (vprofile-code-repo)
* Branch name = vp-rem
* Change detection options = Amazon CloudWatch Events
* Output artifact format = CodePipeline default
* click on next
* Build provider = AWS CodeBuild
* project name = select your project name
* Build type = Single build
* click on next
* Deploy provider = AWS Elastic Beanstalk
* Region = Select same region
* Application name = select your App Names
* Environment name = select your Environment Name
* click on Next
* verify all settings and click on create Pipeline

The steps in Red explains how to add Test and Approvals to your Pipeline if you wish:

* click on edit tab beside Notify (We didn’t include the Test stage)
* Just beneath the build Job, click on Add Stage
* Stage name = Test
* Under Test, click on Add action group
* Action name = Unit Test
* Action provider = AWS CodeBuild
* Region = Select same region
* Input artifacts = BuildArtifact
* Project name = vprofile-build
* scroll down and click on ok
* Just beneath the Test, click on Add Stage (To add minor approval)
* Stage name = Approval
* Under Approval, click on Add action group
* Action name = approval
* Action provider = manual approval
* SNS topic ARN = Monitoring-Team
* URL for review = The Beanstalk link
* scroll down and click on done
* Scroll up or Down and click on Save

CleanUp of Resources to prevent Billing:

1. Remove the RDS, Beanstalk from security groups by removing the Inbound rules
2. Delete the RDS (uncheck create final snapshot and check I acknowledge that upon instance…..)
3. Go to Beanstalk and click on environment to delete the env you created
4. You can leave the AWS Pipeline becos it is not charged unless you use them