CODESTAR-Build,test,deploy the javaspring application in Ec2.

Working with Projects in AWS CodeStar:

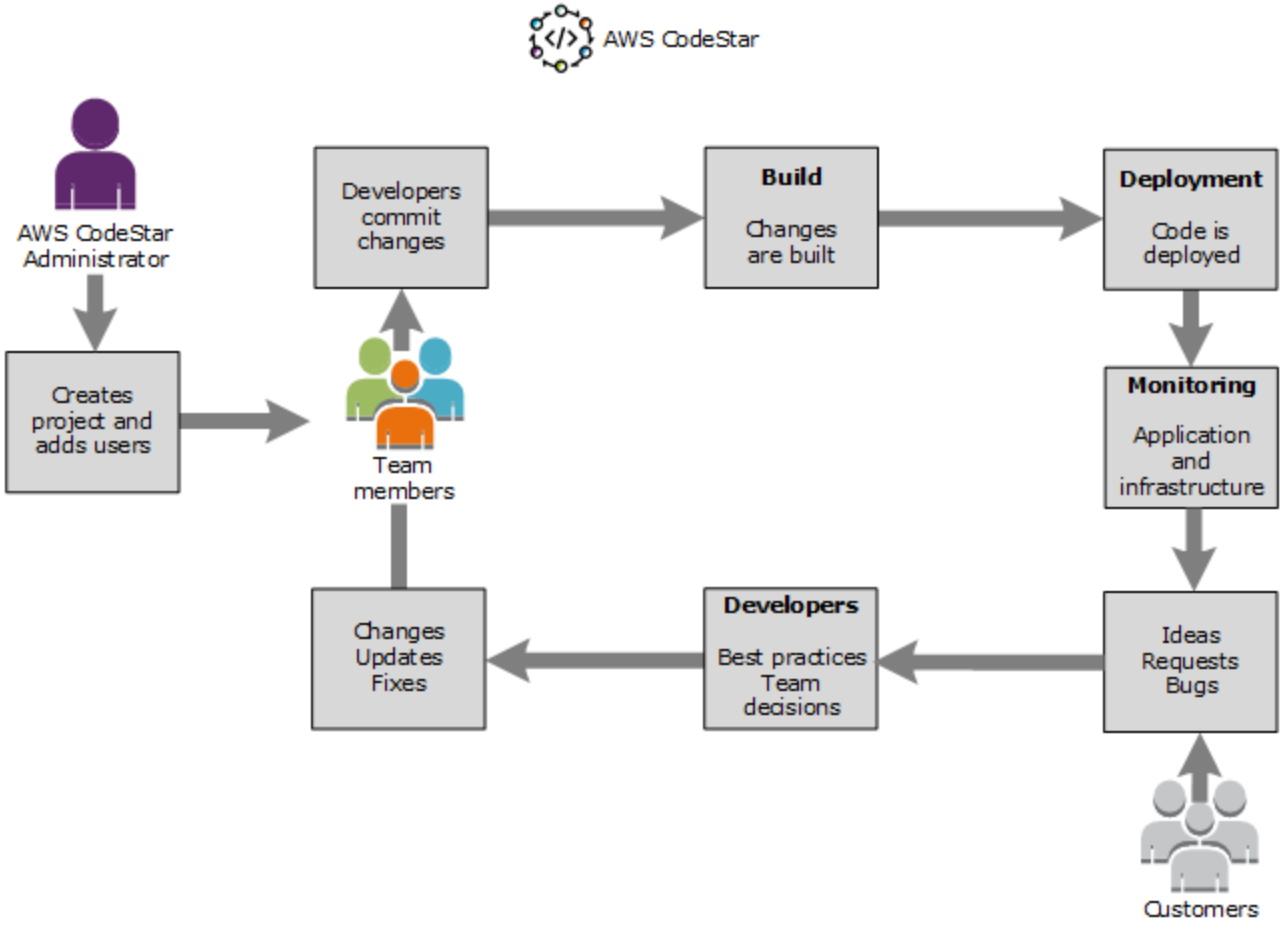
When you use an AWS CodeStar project template, you can quickly create a project that is already configured with the resources you need, including:

- Source repository
- Build environment
- Deployment and hosting resources
- Programming language

The template even includes sample source code so you can start working with your project right away.

After you have a project, you can add or remove resources, customize your project dashboard, and monitor progress.

The following diagram shows a basic workflow in an AWS CodeStar project.



1. A developer with the `AWSCodeStarFullAccess` policy applied creates a project and adds team members to it. Together they write, build, test, and deploy code.
2. The project dashboard provides tools that can be used in real time to view application activity and monitor builds, the flow of code through the deployment pipeline, and more.
3. The team uses the team wiki tile to share information, best practices, and links.
4. They integrate their issue-tracking software to help them track progress and tasks.
5. As customers provide requests and feedback, the team adds this information to the project and integrates it into their project planning and development. As the project grows, the team adds more team members to support their code base.

Create a Project(java app in ec2) in AWS CodeStar:

You use the AWS CodeStar console to create a project. If you use a project template, it sets up the required resources for you. The template also includes sample code that you can use to start coding.

To create a project, sign in to the AWS Management Console with an IAM user that has the `AWSCodeStarFullAccess` policy or equivalent permissions.

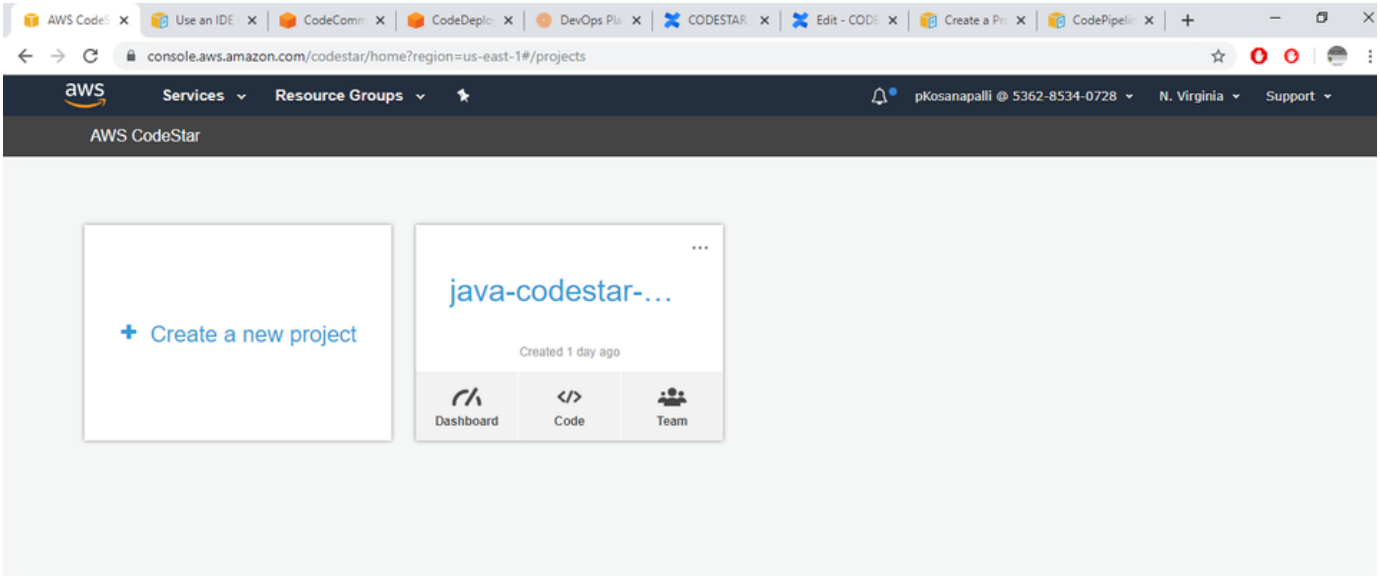
You must complete the steps in How CodeStar - Getting started- how to create project in codestar in AWS before you can complete the procedures in this topic.

Build, test, package& deploy the javaspring application in ec2:

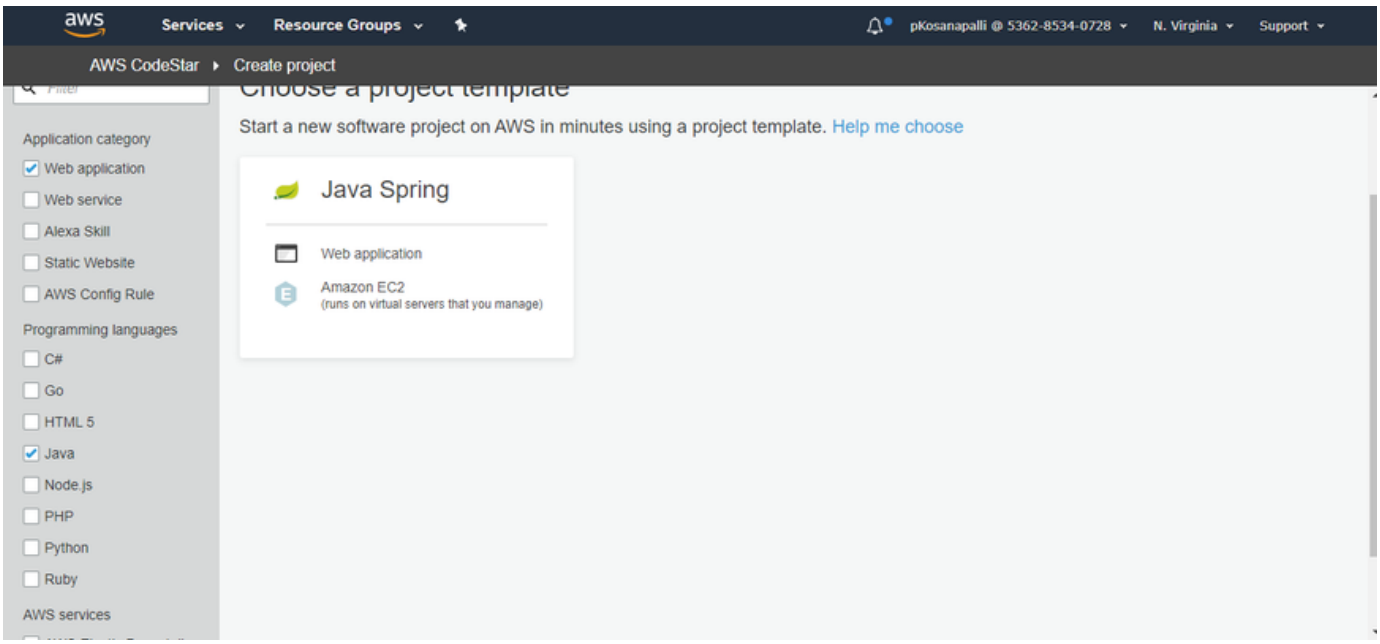
Use the AWS CodeStar console to create a project.

To create a project in AWS CodeStar

- Sign in to the AWS Management Console, and then open the AWS CodeStar console at https://console.aws.amazon.com/codestar/.
- On the **AWS CodeStar** page, choose **Create a new project**. (If you are the first user to create a project, choose **Start a project**.)



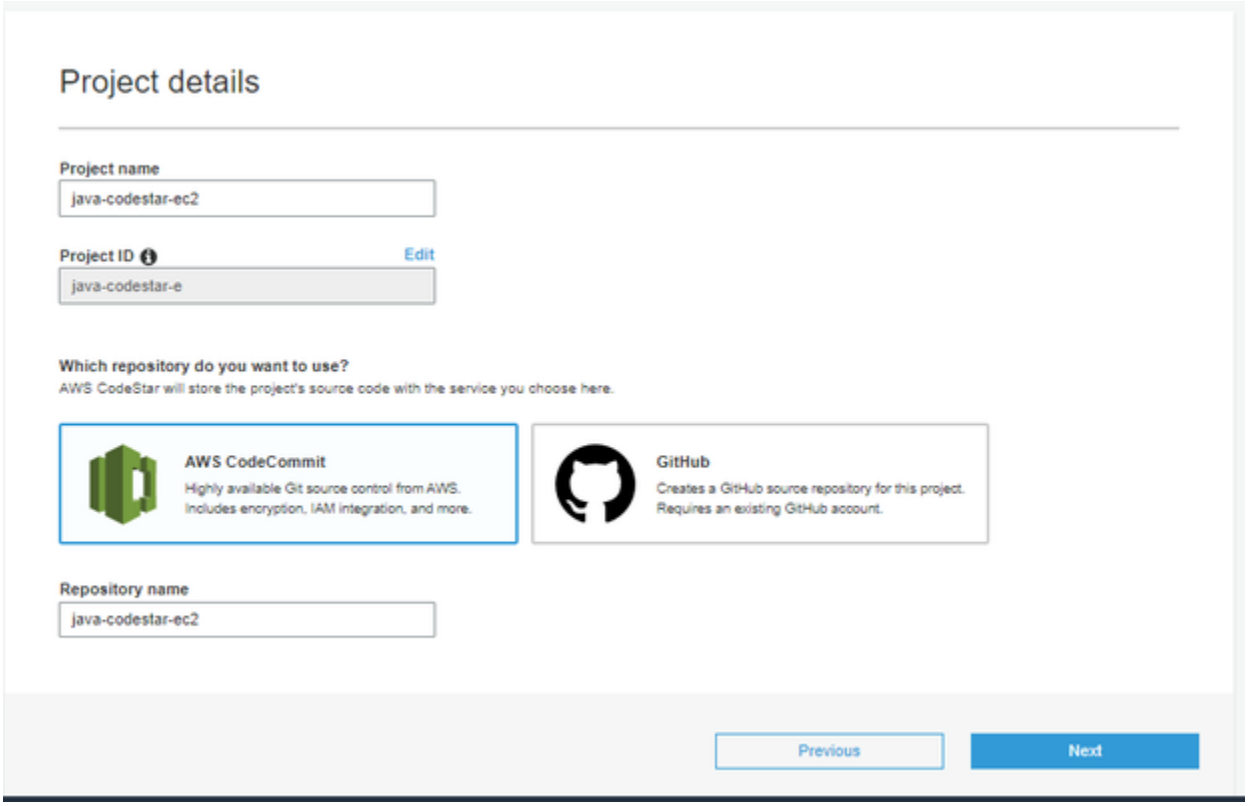
- On the **Choose a project template** page, choose the project type from the list of AWS CodeStar project templates. You can use the filter bar to narrow your choices. For example, for a web application project written in java spring to be deployed to Amazon EC2 instances, select the **Web application**, **java** and **Amazon EC2** check boxes. Then choose from the templates available for that set of options.



- In **Project name**, enter a name for the project, such as `My First Project`. The ID for the project is derived from this project name, but is limited to 15 characters.

  For example, the default ID for a project named `My First Project` is `java-codestar-ec2`. This project ID is the basis for the names of all resources associated with the project. AWS CodeStar uses this project ID as part of the URL for your code repository and

for the names of related security access roles and policies in IAM. After the project is created, the project ID cannot be changed. To edit the project ID before you create the project, choose **Edit**. Project IDs must be unique for your AWS account in an AWS Region.



- Choose the repository provider, **AWS CodeCommit** or **GitHub**.
- If you chose **AWS CodeCommit**, for **Repository name**, accept the default AWS CodeCommit repository name, or enter a different one.
- If you chose **GitHub**, choose **Connect with GitHub.** go to this documention CODESTAR integration with github .
- Choose **Next**. You can see the project details i.e codecommit, codebuild, code deploy, codepipeline, cloudwatch. these services are default created.

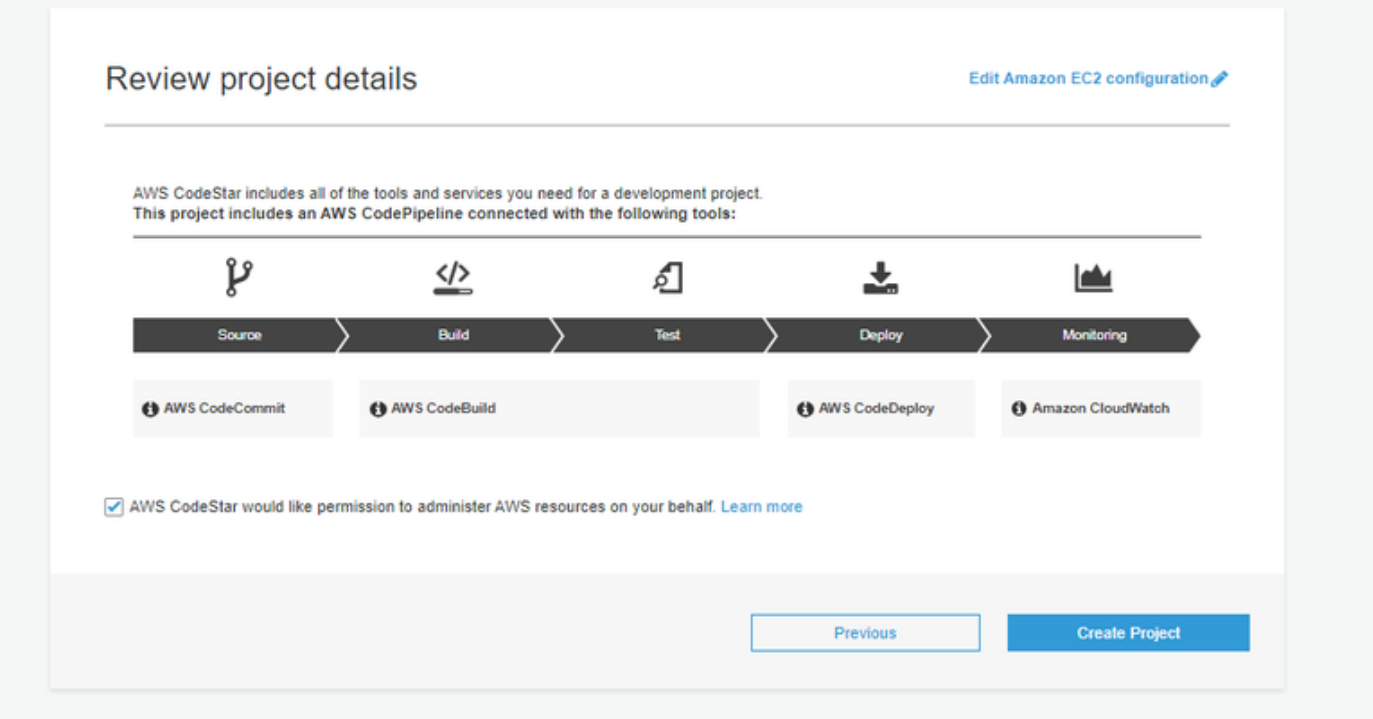
code commit- creates new repository.

codebuild - creates default codebuild service contains source, build environment(docker image,environment,service role),variables, build spec,artifacts, logs.

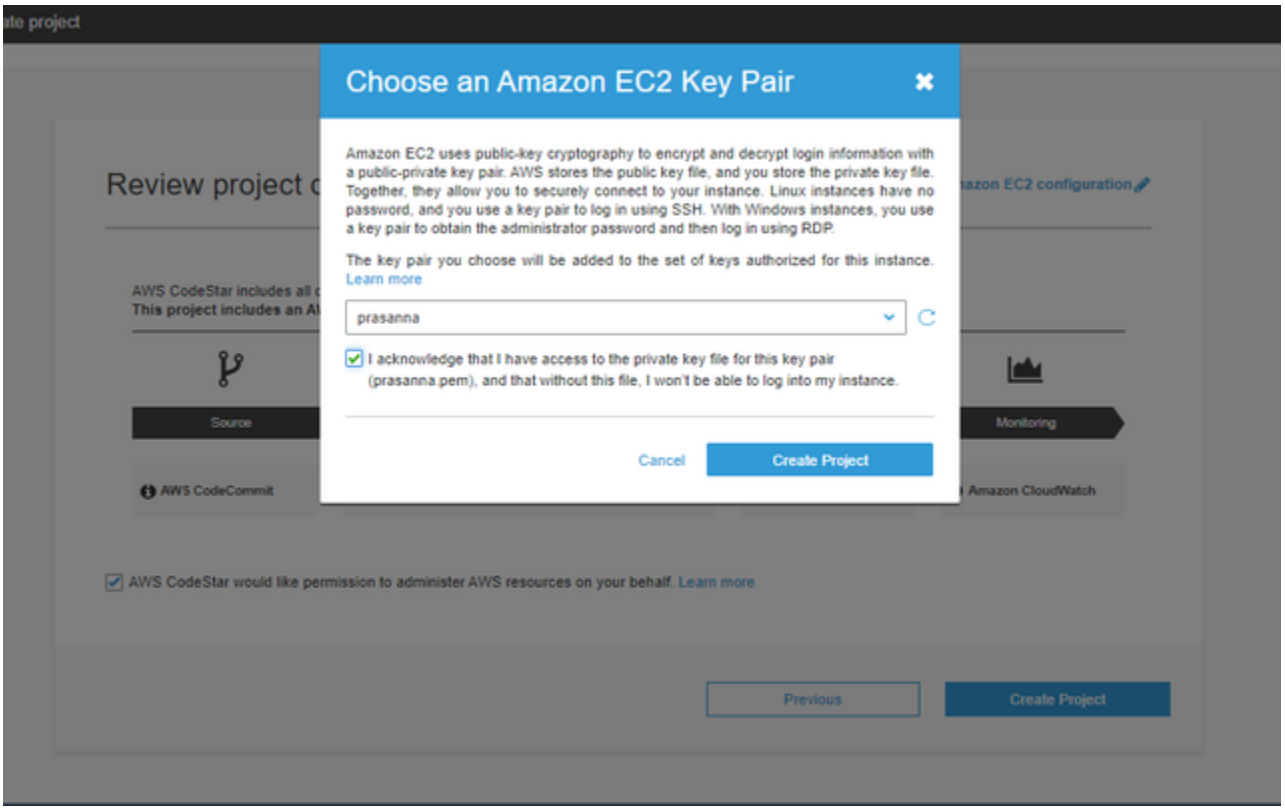
codedeploy - deployments, deployment groups (service role, compute platform.)

codepipeline- source,build,deploy.

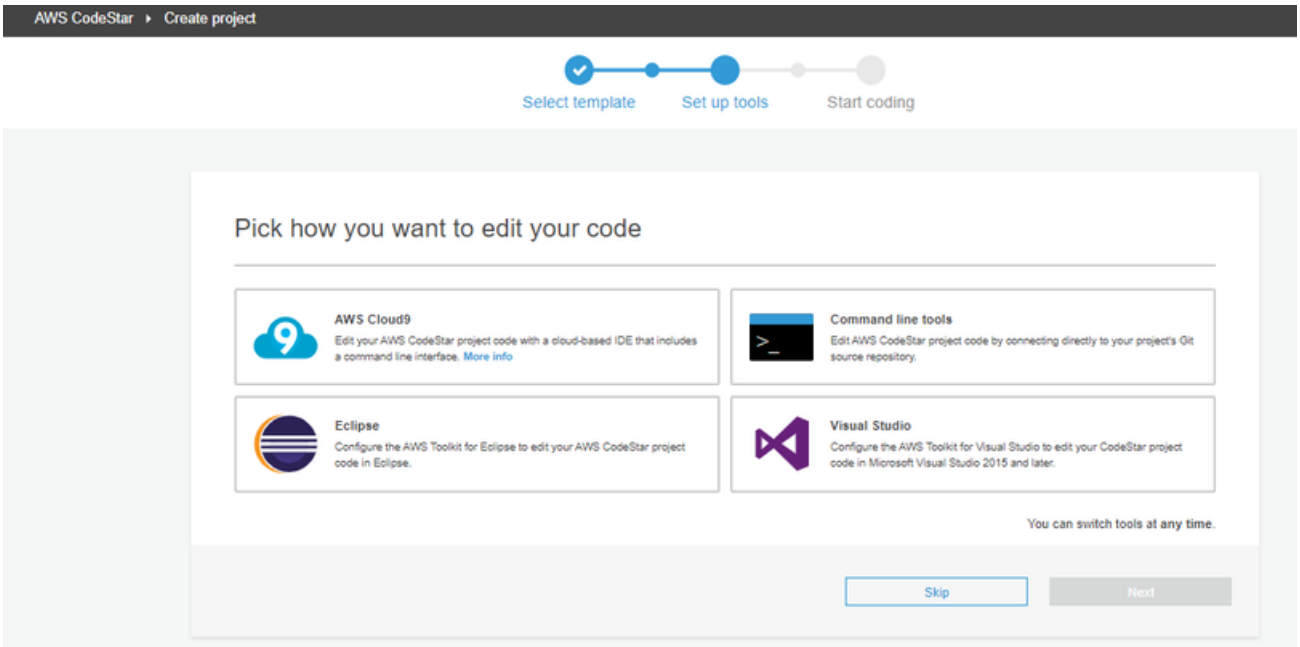
- Review the resources and configuration details.



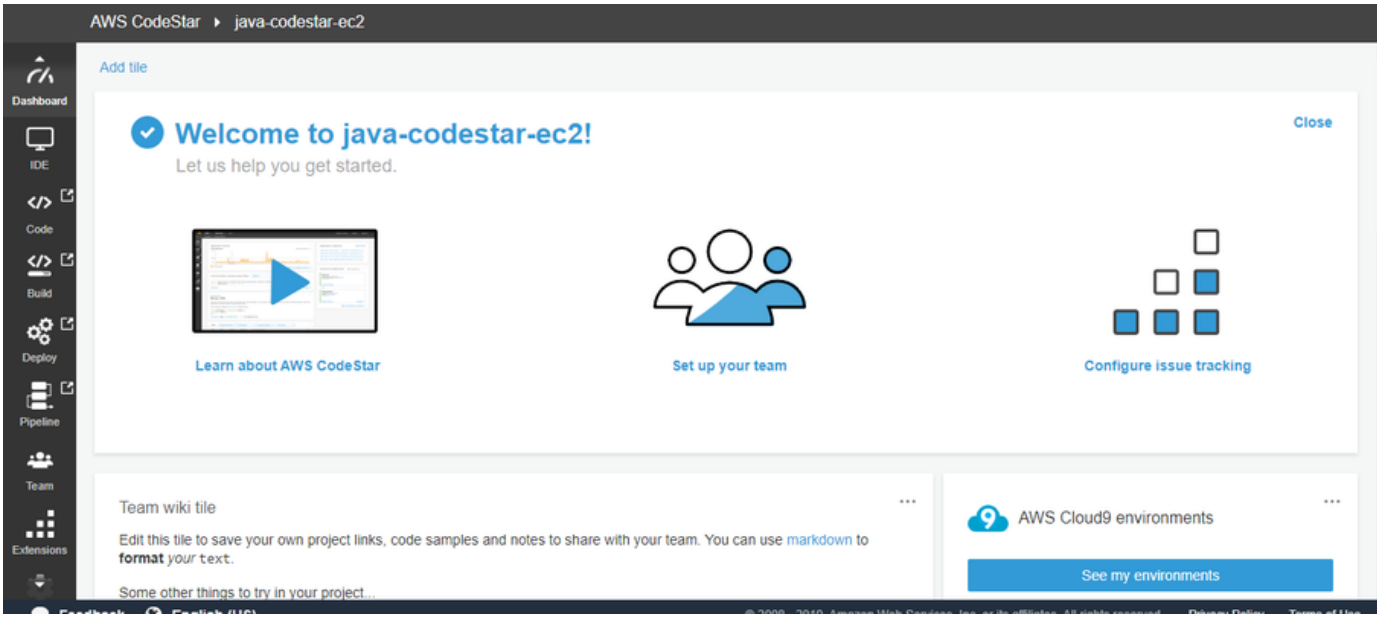
- In **Choose an Amazon EC2 Key Pair**, choose the Amazon EC2 key pair you created in keypairs.
- Select **I acknowledge that I have access to the private key file for this key pair**, and then choose **Create project**.



- It might take a few minutes to create the project (including the repository). After your project has a repository, you can use the **Set up tools** page to configure access to it, or you can choose **Skip** and configure access later.



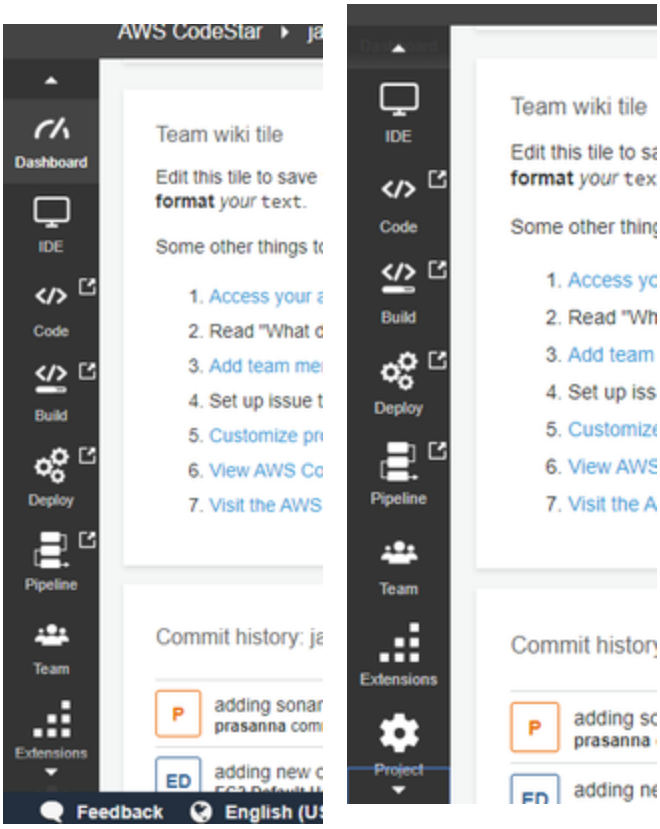
- After your project has been created, you can use the links on the **Welcome** tile to configure other items, such as your user profile in AWS CodeStar.



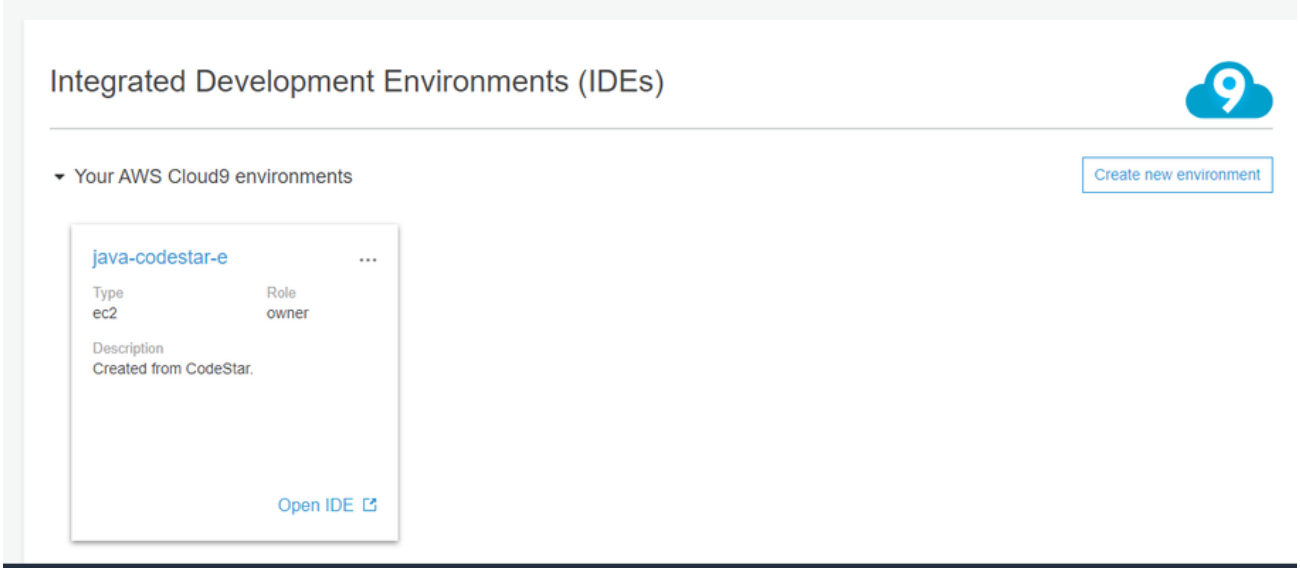
- While your project is being created, you can add team members or configure access to your project repository from the command line or your favorite IDE.
- You can connect your favorite IDE in any one of IDE's AWS cloud9, Eclipse, Vs code, AWS cli., for complete setup with IDE's, see the documentions here Codestar integrate with IDEs- ECLIPSE, CLOUD9,VS CODE.

Lets see the content of Welcome board of codestar project:

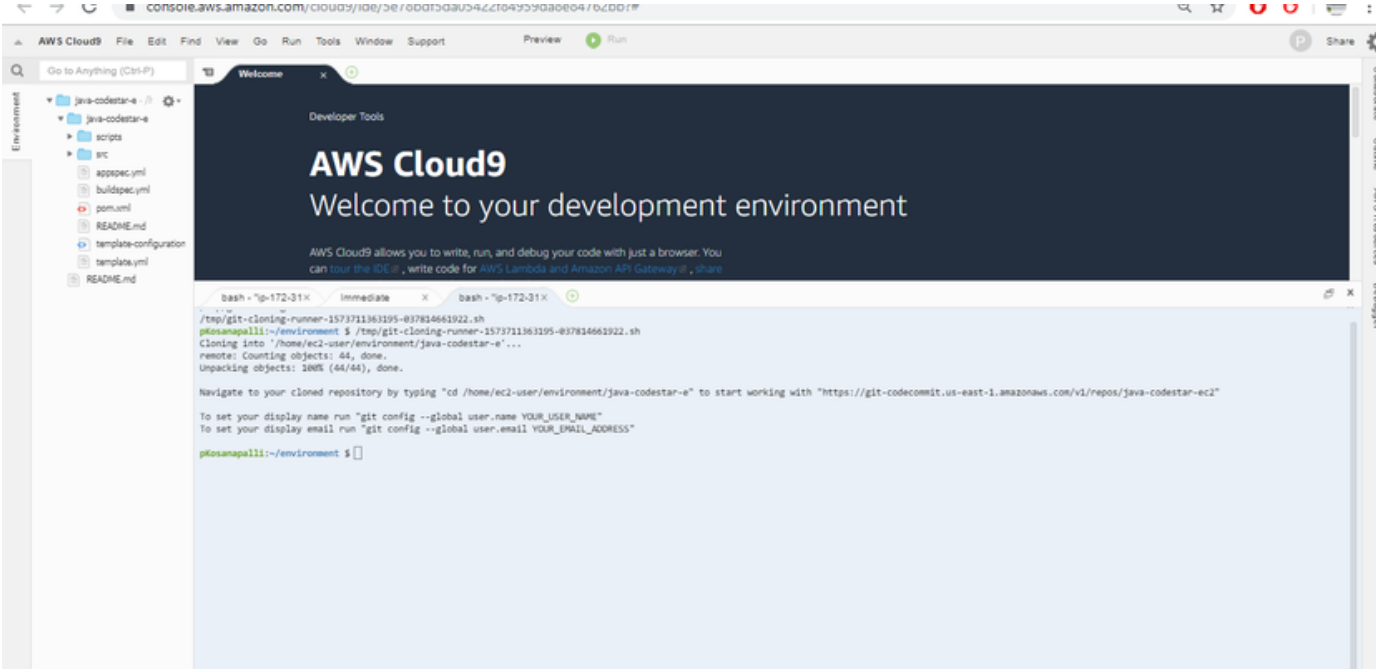
- first you can observe welcome page dashboard of the codestar project, which contains all collaborative services of aws and some of features of codestar.
- On the left side of dashboard you can observe some labels contains IDE, code, build, Deploy,pipeline, team,extensions, project see the below pics.



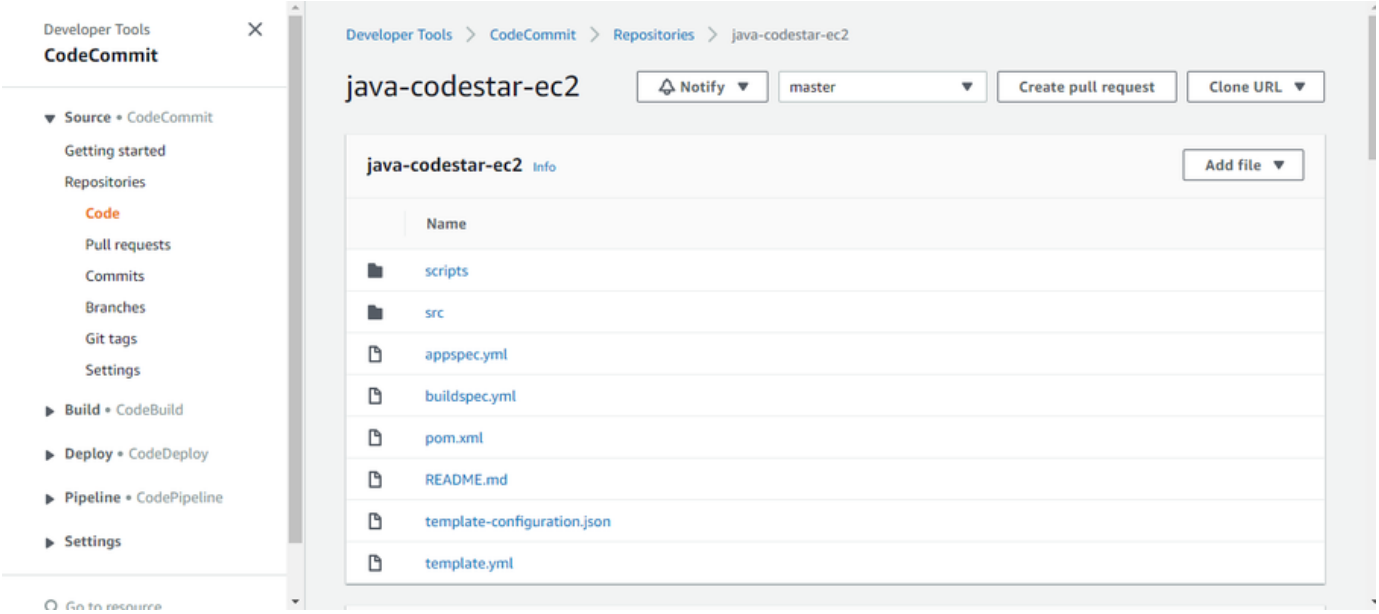
- when ever you click the IDE, you can go to new page, that configure the favorite IDE setup.



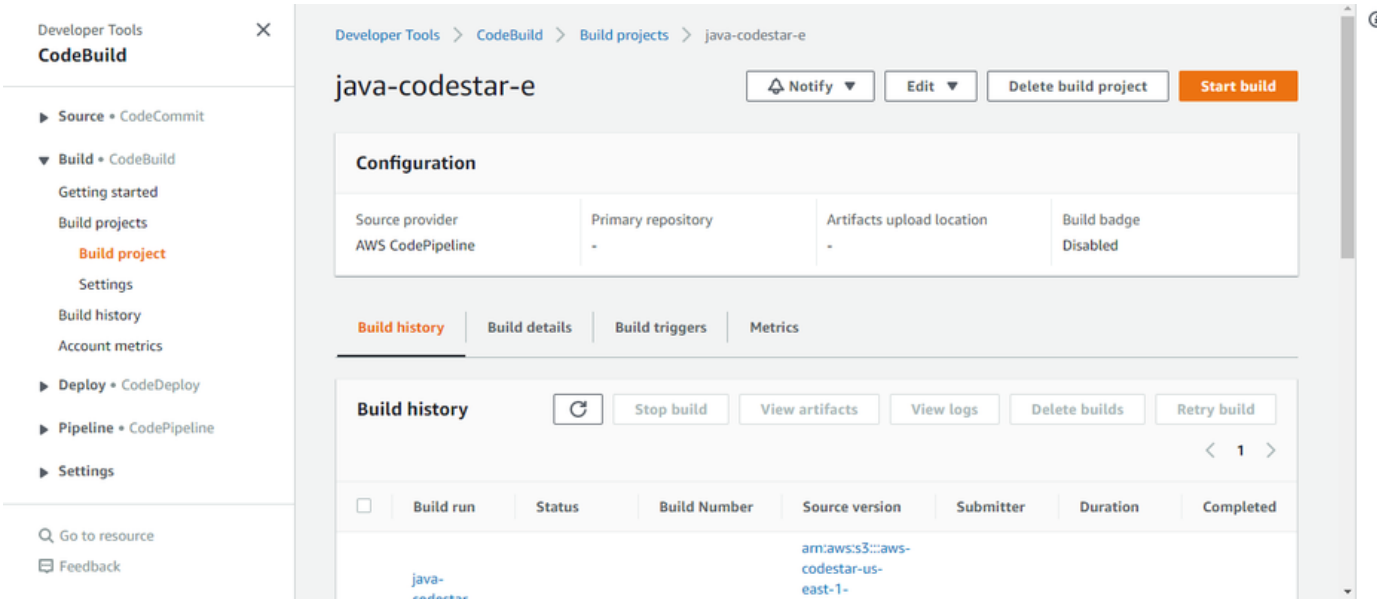
- You can observe the different IDEs setup, right now im configured the cloud9 environment for my code editing.You can connect your favorite IDE in any one of IDE's AWS cloud9, Eclipse, Vs code, AWS cli., for complete setup with IDE's, see the documentions here Code star integrate with IDEs- ECLIPSE, CLOUD9,VS CODE.
- im opted cloud9 IDE, you can open by clicking open IDE environment.



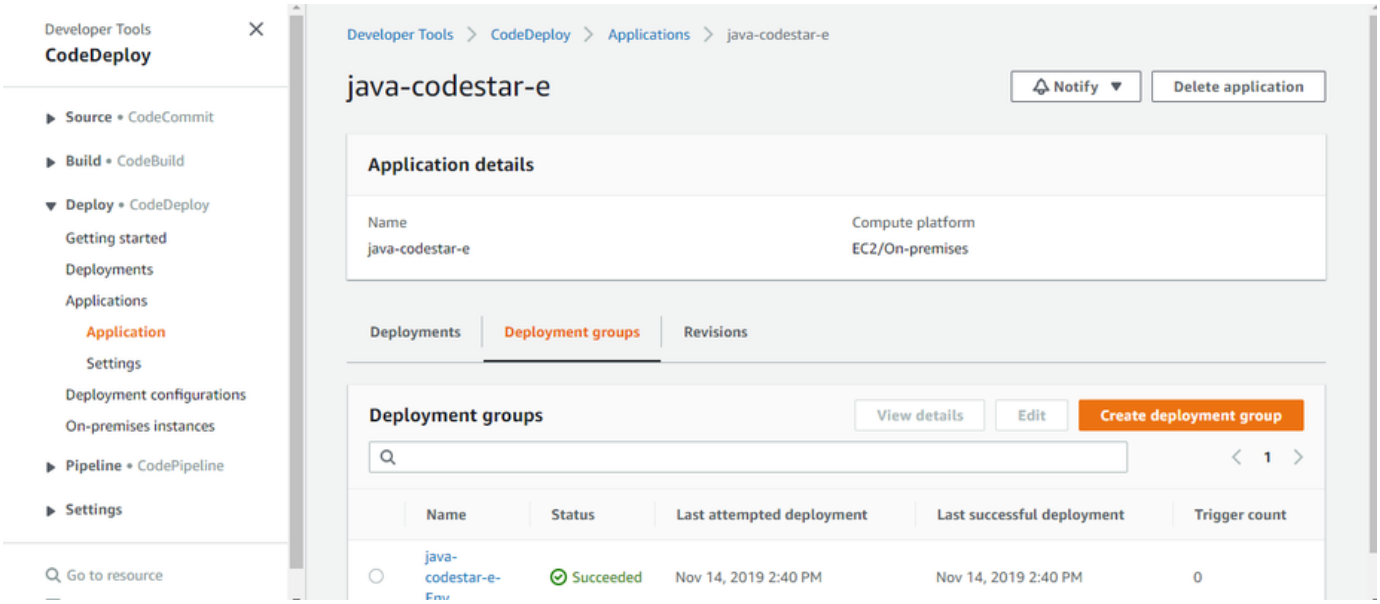
- Now a new instance created &configured for our environment, it contains CLI and repository of your codestar. Through out our document we are using Cloud9 editor.
- Go and see what are things in the instance. this is similar to ubuntu machine in AWS.
- On the left side of dashboard you can observe some labels name code, you click the code, you can go to new page, codestar is configure with new repo name javaspring-ec2.



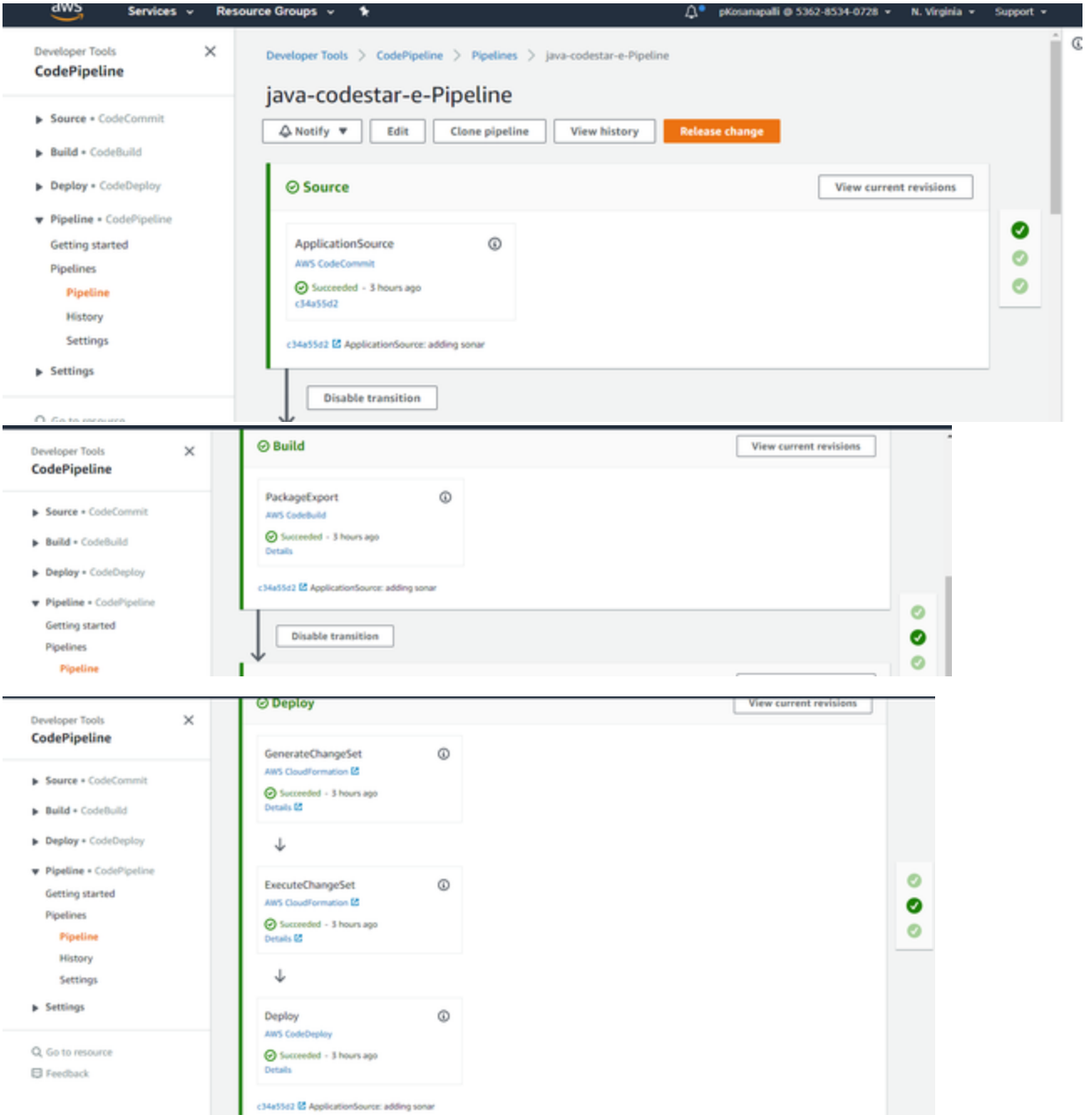
- Now you can clone the code in IDE and make changes, push it to repo. Same you can observe in the repository.
- On the left side of dashboard you can observe some labels name Build, you click the Build, takes to new page, codestar is configure with new codebuild name javaspring-ec2.
- See the codebuild - creates default codebuild service contains source, build environment(docker image,environment,service role),variables, build spec,artifacts, logs, build history.



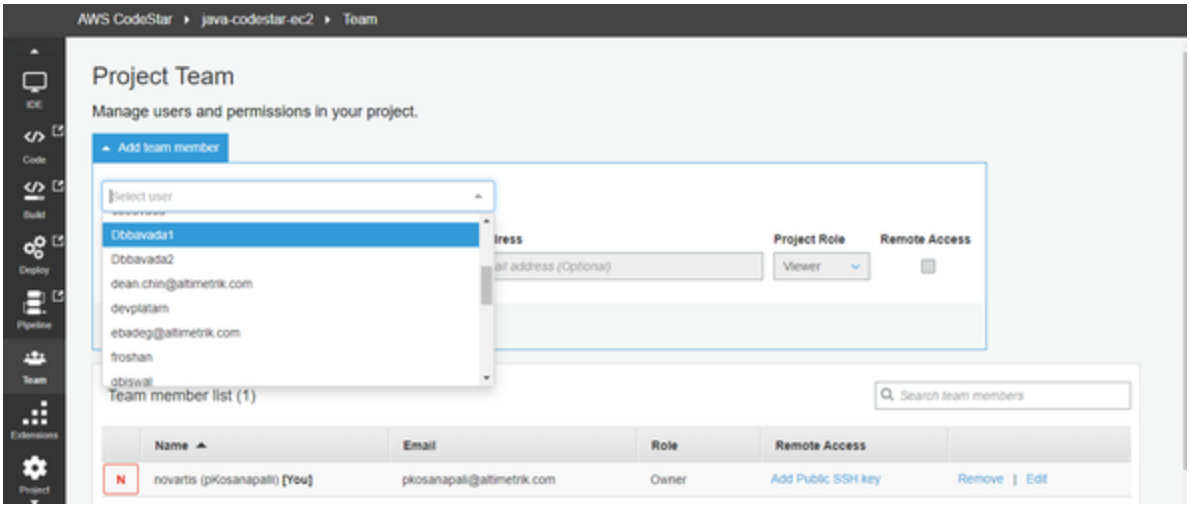
- On the left side of dashboard you can observe some labels name Deploy, you click the Deploy, takes to new page, codestar is configure with new codedeploy name javaspring-ec2. Codedeploy contains - deployments, deployment groups (service role, compute platform.)

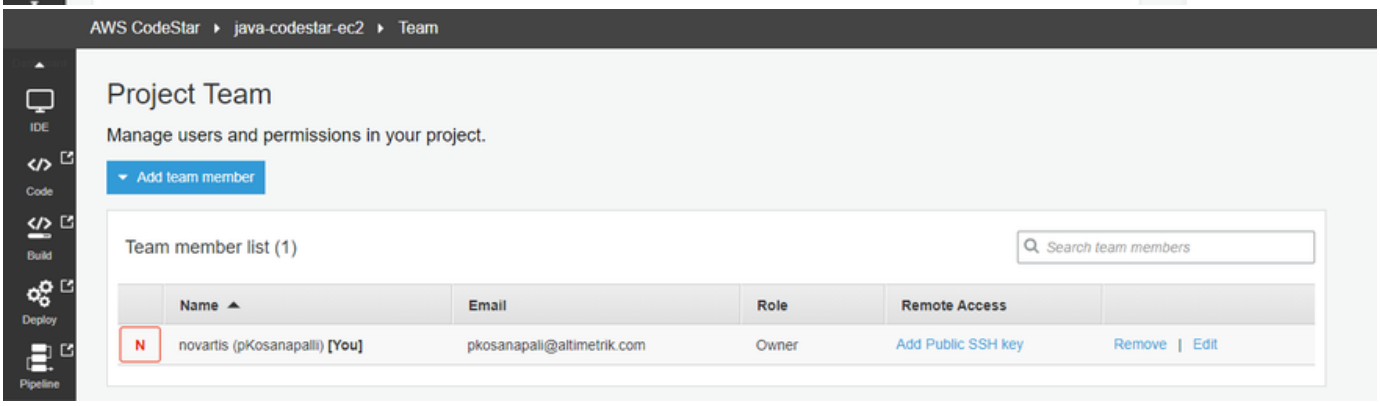


- On the left side of dashboard you can observe some labels name Pipeline, you click the pipeline, takes to new page, codestar is configure with new codepipeline name javaspring-ec2. Codepipeline contains integrations of code commit, codebuild, code deploy.

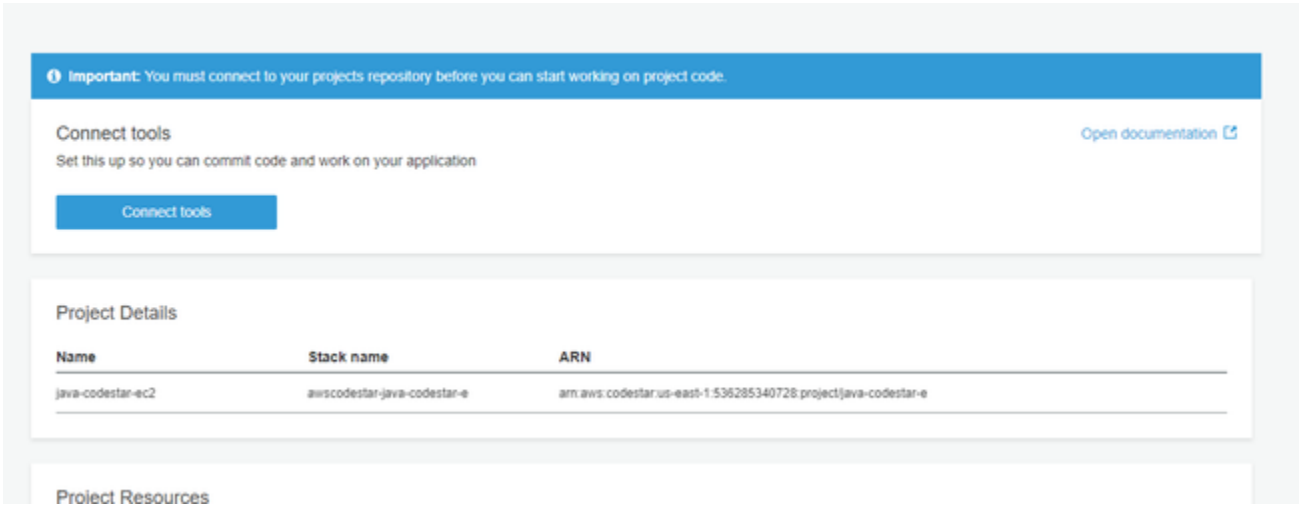


- On the left side of dashboard you can observe some labels name Team, you click the Team, takes to new page, codestar is configure with new feature, add team mates and restrict them the role.





- On the left side of dashboard you can observe some labels name extensions, you click the extensions, codestar is configure with new feature, integrations i.e, github, jira. for complete setup for integration see this codestar- How to integrate with JIRA.
- On the left side of dashboard you can observe some labels name project details, you click this, you can observe the project details, project resources, connected tools.

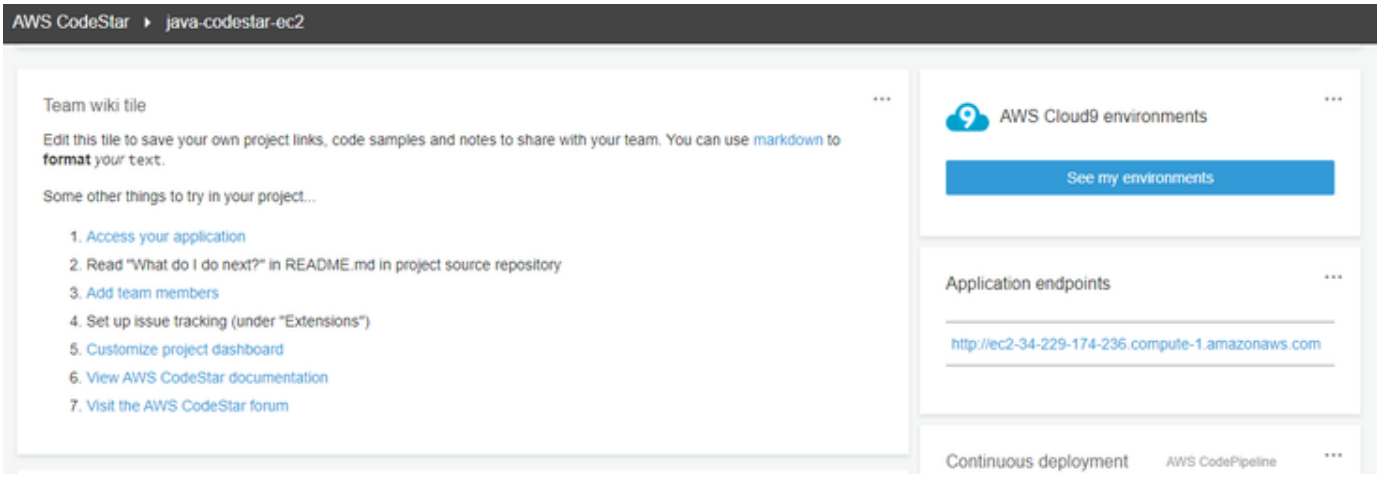


Project Resources

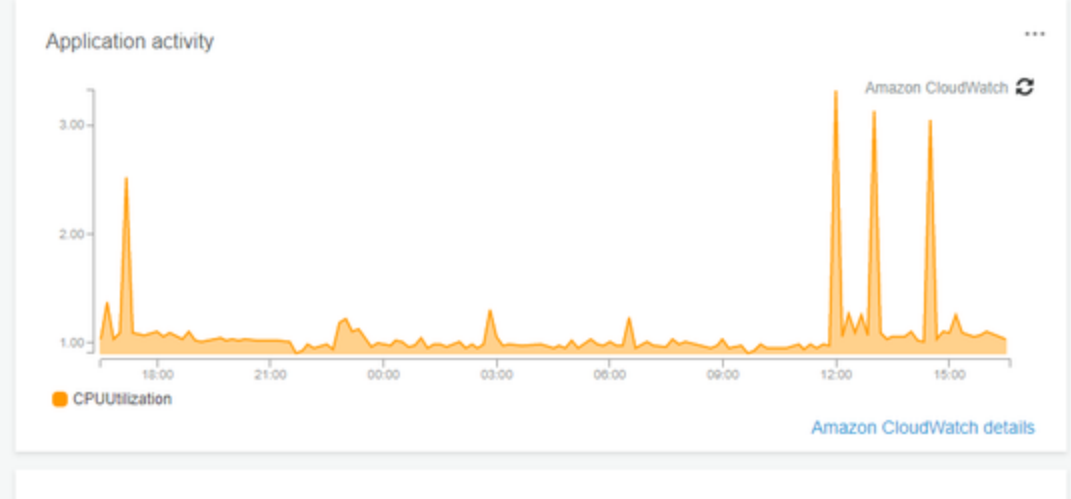
Type	Name	ARN
AWS Cloud9	environment:5e78bdf5da05422f84959da8...	arn:aws:cloud9:us-east-1:536285340728:environment:5e78bdf5da05422f84959da8e84762bb
AWS CloudFormation	stack/awscodestar-java-codestar-e/fa8968...	arn:aws:cloudformation:us-east-1:536285340728:stack/awscodestar-java-codestar-e/fa896890-05ff-11ea-8929-12b44...
AWS CloudFormation	stack/awscodestar-java-codestar-e-infrastr...	arn:aws:cloudformation:us-east-1:536285340728:stack/awscodestar-java-codestar-e-infrastructure/a01a8280-0600-11...
AWS CodeBuild	project/java-codestar-e	arn:aws:codebuild:us-east-1:536285340728:project/java-codestar-e
AWS CodeCommit	java-codestar-ec2	arn:aws:codecommit:us-east-1:536285340728:java-codestar-ec2
AWS CodeDeploy	deploymentgroup:java-codestar-e/java-cod...	arn:aws:codedeploy:us-east-1:536285340728:deploymentgroup:java-codestar-e/java-codestar-e-Env
AWS CodeDeploy	application:java-codestar-e	arn:aws:codedeploy:us-east-1:536285340728:application:java-codestar-e
AWS CodePipeline	java-codestar-e-Pipeline	arn:aws:codepipeline:us-east-1:536285340728:java-codestar-e-Pipeline

AWS IAM	role/CodeStarWorker-java-codestar-e-Tool...	arn:aws:iam::536285340728:role/CodeStarWorker-java-codestar-e-ToolChain
AWS IAM	role/CodeStarWorker-java-codestar-e-Web...	arn:aws:iam::536285340728:role/CodeStarWorker-java-codestar-e-WebApp
AWS IAM	policy/CodeStar_java-codestar-e_Permissi...	arn:aws:iam::536285340728:policy/CodeStar_java-codestar-e_PermissionsBoundary
AWS IAM	role/CodeStarWorker-java-codestar-e-Clou...	arn:aws:iam::536285340728:role/CodeStarWorker-java-codestar-e-CloudFormation
Amazon EC2	instance/i-05fdf4633c533c4e5	arn:aws:ec2:us-east-1:536285340728:instance/i-05fdf4633c533c4e5
Amazon EC2	security-group/sg-050bf849e4230a47d	arn:aws:ec2:us-east-1:536285340728:security-group/sg-050bf849e4230a47d
Amazon S3	aws-codestar-us-east-1-536285340728-jav...	arn:aws:s3:::aws-codestar-us-east-1-536285340728-java-codestar-e-pipe

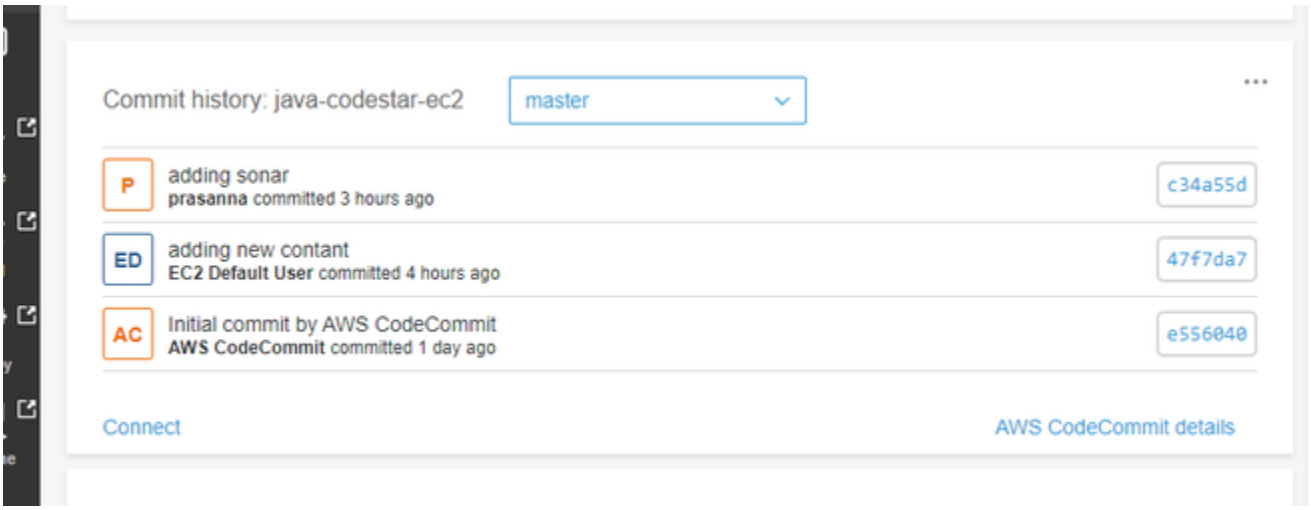
- this are the resources using for codestarproject. All resource under code-star (s3, codepipeline,ec2) is configured with cloud-formation (stored under secret) for complete background of architecture of CODESTAR is documented separately- CODESTAR- background architecture.
- you can see the corresponding resource dashboard by clicking the corresponding urls in the pics.
- come back to thedashboard of codestar javaspring app project. you can see dashboard with some blog views, which are configured to view title. they are as below.



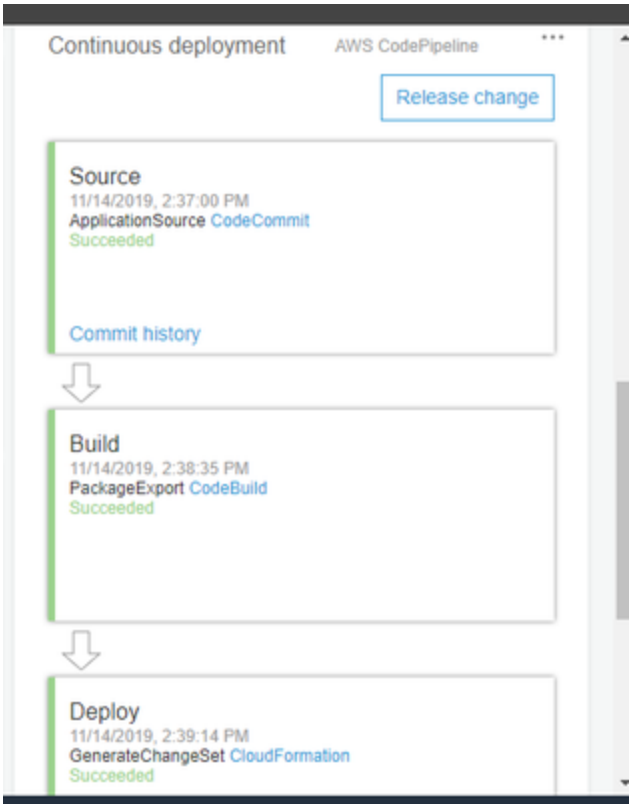
- AWS cloud9 environment you can redirect to IDE dashboard.
- Application endpoints- which are the configured for application output dashboard you can see the changes here by clicking the URL.
- Application activity - which can be monitor by the AWS cloudwatch for the application, click the cloudwatch details for more details.



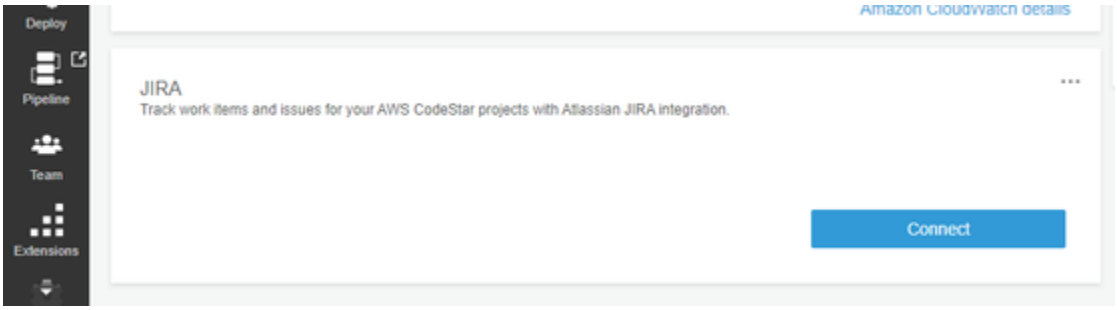
- commit history- view latest changes of codecommit in this project.



- continuous deployment- dashboard view for code pipeline.



- you can configure the Jira issue with code star. for more details see codestar- How to integrate with JIRA.

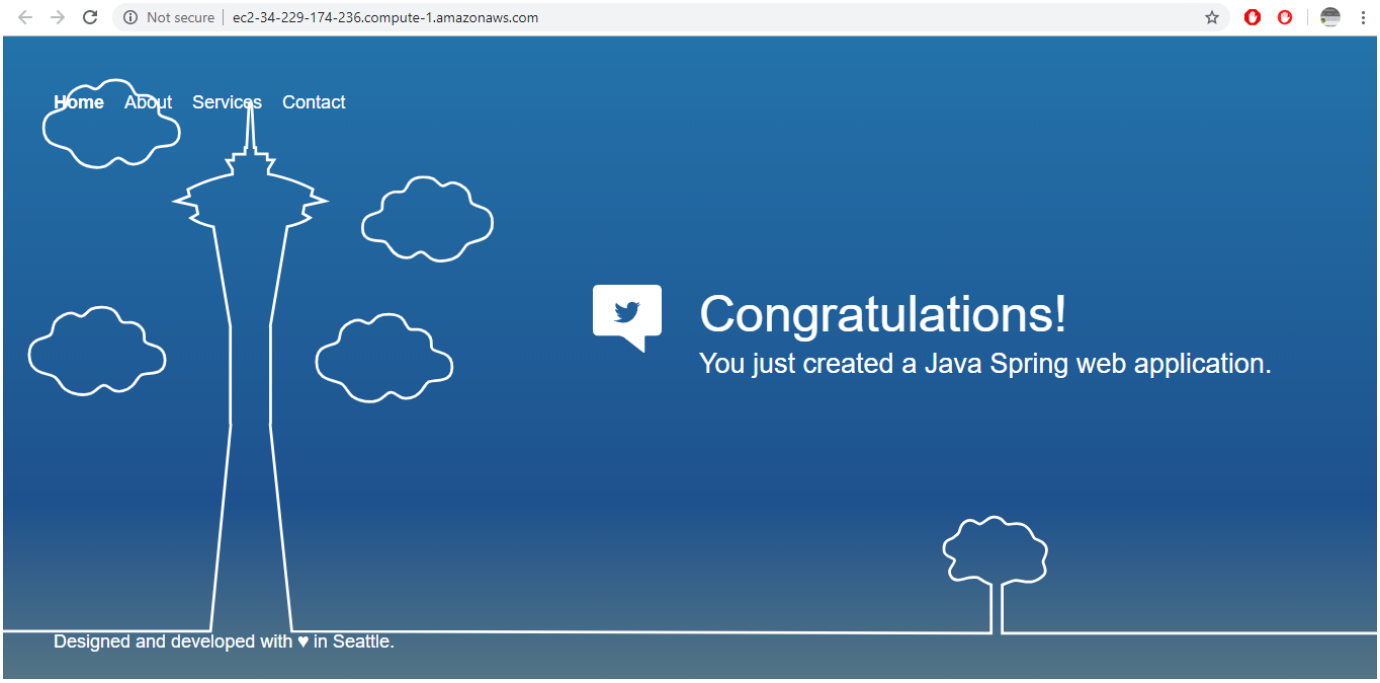


Build, test, package& deploy the javaspring application in ec2:

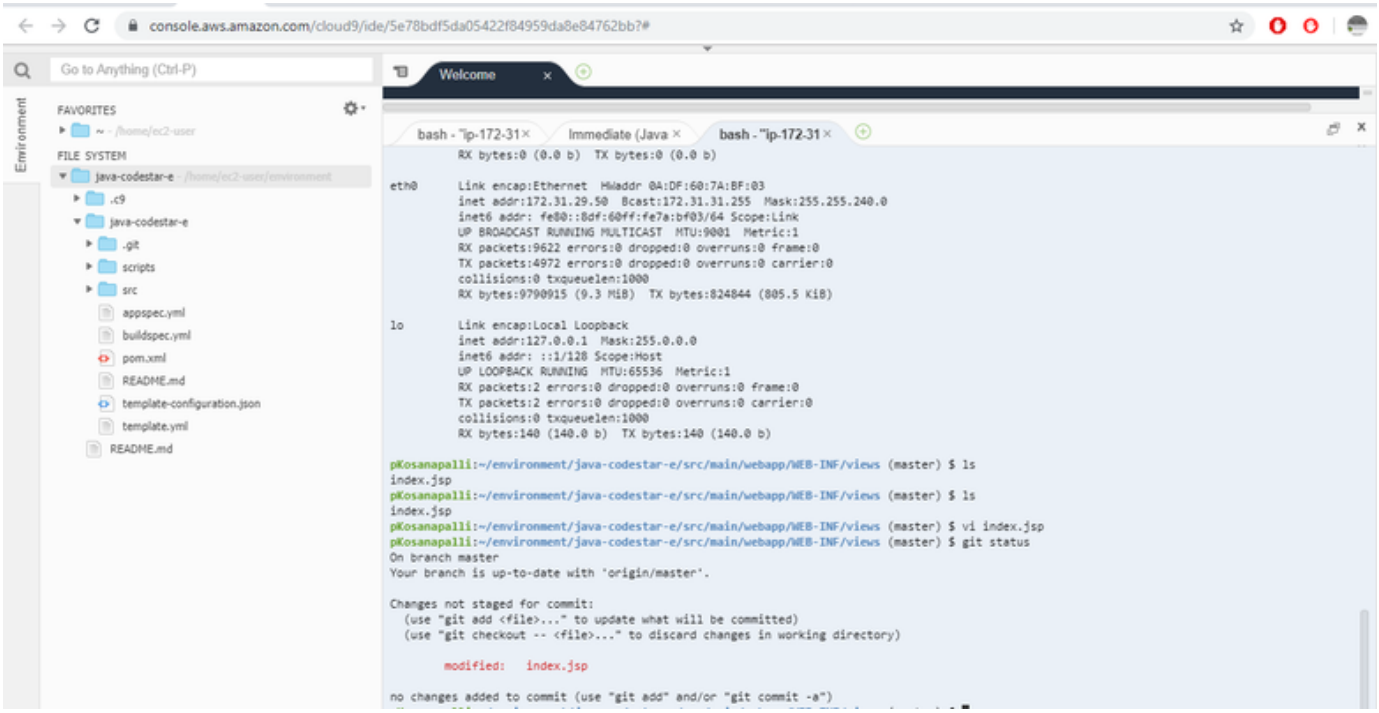
- Now we have created a javaspring project in codestar.
- Go to project explore click the endpoint(url of application), redirect to new url of your application, this url can be configure based on our requirement in the DNS service.

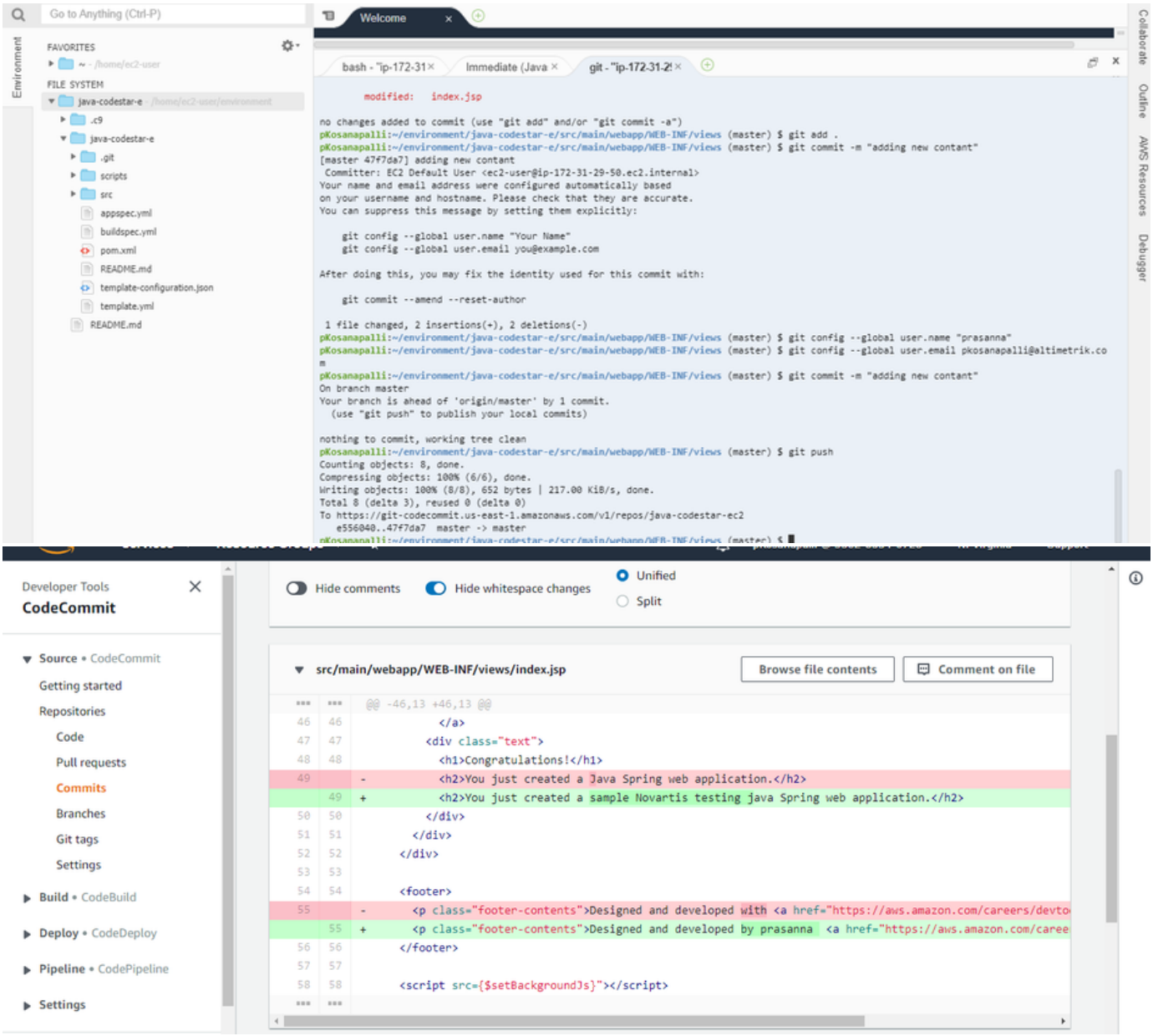
```
http://ec2-34-229-174-236.compute-1.amazonaws.com/
```

- you can able to see the new java-spring application.

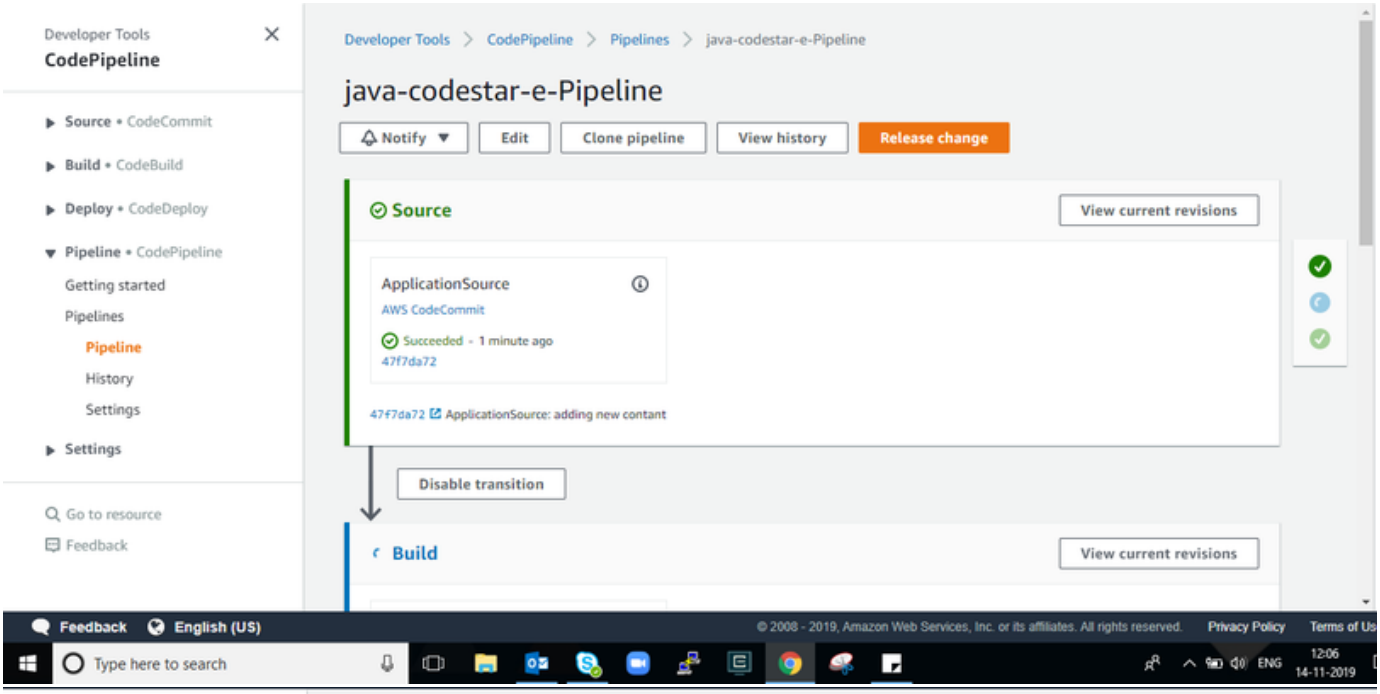


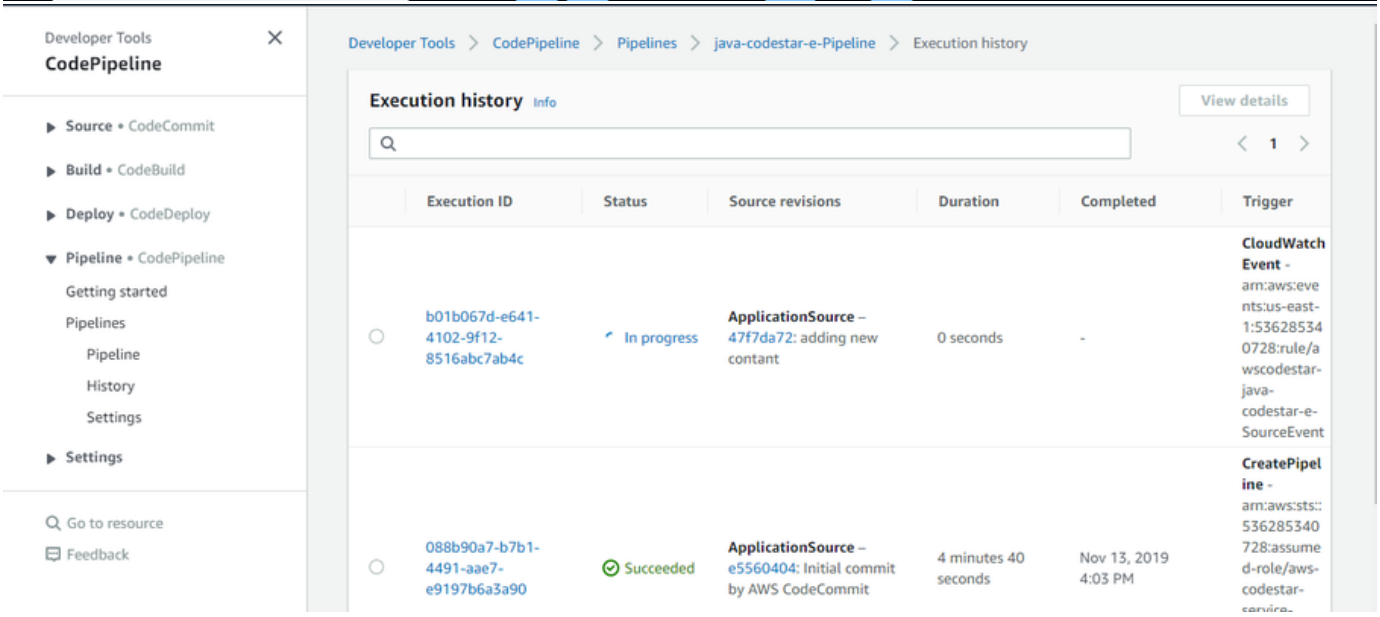
- Now go to the Cloud9 editor and modify the required changes and pull the changes to codestar. for more go to Codestar integrate with IDEs- ECLIPSE, CLOUD9,VS CODE.



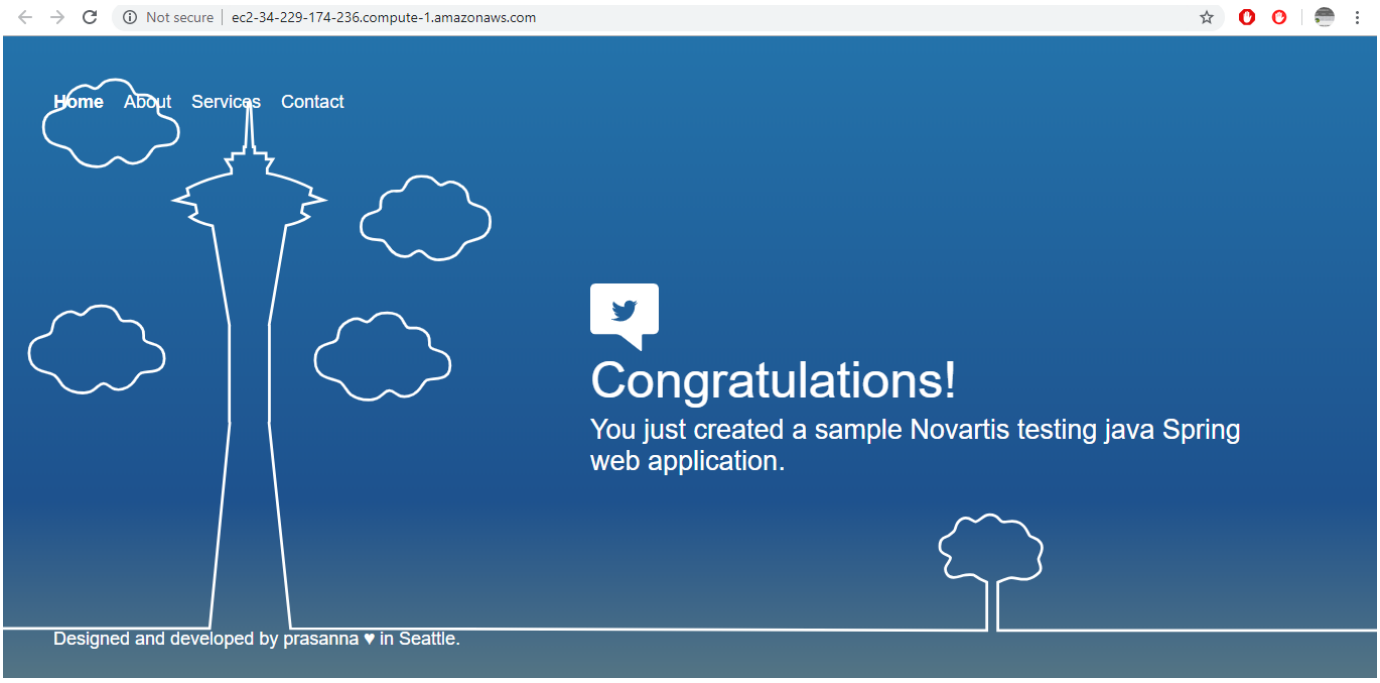


- Now a new release is triggered in the code pipeline with latest changes. We have configured our code pipeline like this. Go to pipeline and check this.





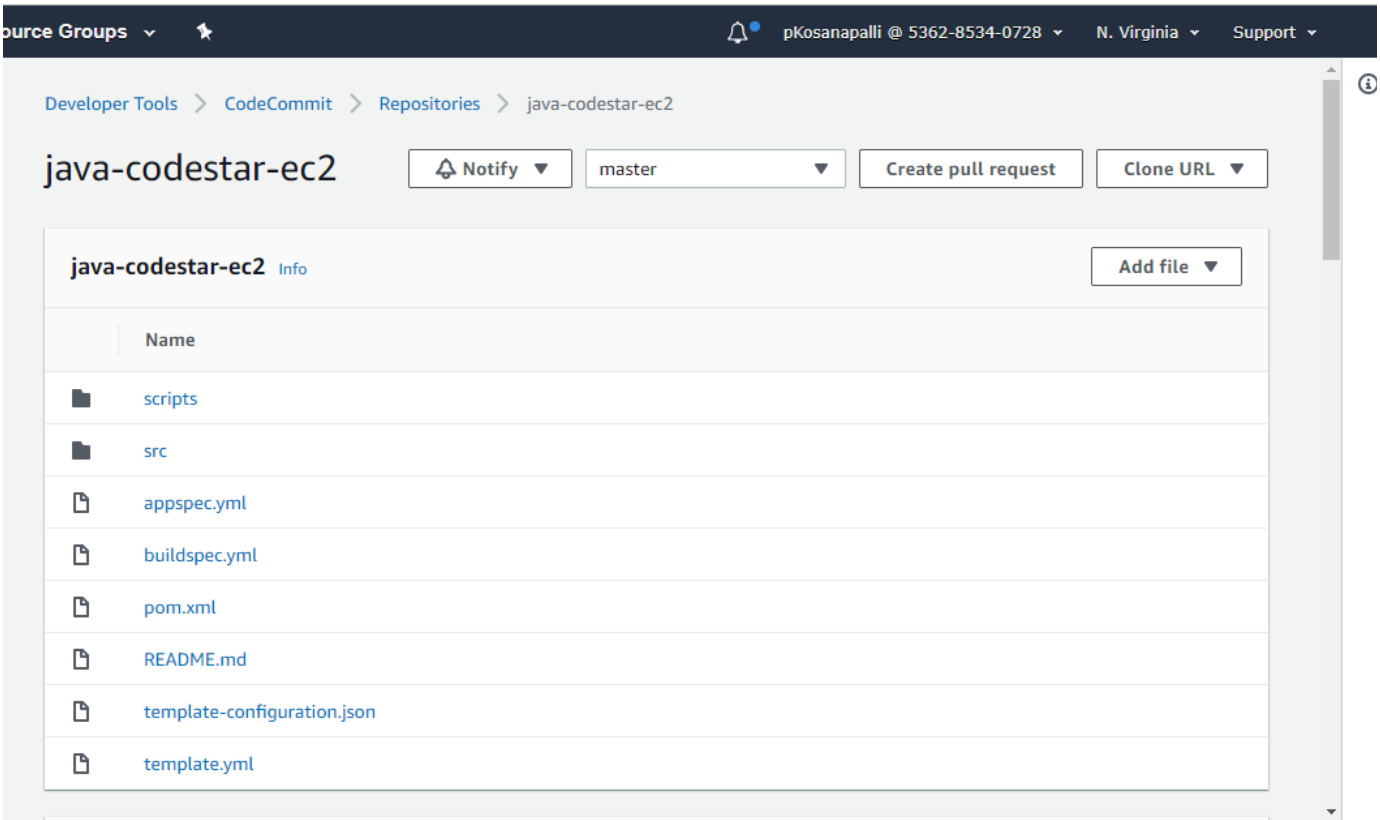
- Go to the Url of the application, see the changes we made.



- Now we have done the build, test, deploy the java-spring application.

How we build,test,deploy the javaspring application in EC2:

- First lets see whats things in code commit(SCM), go to code commit & see the repo.



- lets see what it is

```
This sample code contains a simple Java web application deployed by AWS
CodeDeploy and AWS CloudFormation to an Amazon EC2 server.
What's Here
This sample includes:
README.md - this file
appspec.yml - this file is used by AWS CodeDeploy when deploying the web
application to EC2
buildspec.yml - this file is used by AWS CodeBuild to build the web
application
pom.xml - this file is the Maven Project Object Model for the web
application
src/main - this directory contains your Java service source files
src/test - this directory contains your Java service unit test files
scripts/ - this directory contains scripts used by AWS CodeDeploy when
installing and deploying your application on the Amazon EC2 instance
template.yml - this file contains the description of AWS resources used
by AWS CloudFormation to deploy your infrastructure
template-configuration.json - this file contains the project ARN with
placeholders used for tagging resources with the project ID
```

- lets see the code individually etc.

```
{
  "Tags":
  {

"awscodestar:projectArn":"arn:$PARTITION$:codestar:$AWS_REGION$:$ACCOUNT
_ID$:project/$PROJECT_ID$"
  }
}
----
all resources under this codestar project will be created with this
naming configuration.
Name:java-codestar-e
Project ARN:
arn:aws:codebuild:us-east-1:536285340728:project/java-codestar-e
```

Code deploy-infrastructure automation with cloud formation(ec2& tomcat,java):-

- Once after selecting the application type and deployment, we need to configure the required resources for codestar project i.e s3, codepipeline, ec2 instance.For that purpose we need to create cloud formation stack for creating the ec2 instance and configurations,

```
this is cloud formation stack for creating the ec2 instance with
specified default parameters and security group also.
AWSTemplateFormatVersion: 2010-09-09
```

```
Conditions:
  UseSubnet: !Not [!Equals [!Ref 'SubnetId', subnet-none]]
  IsBurstableInstanceType: !Equals [!Select [0, !Split ['.', !Ref
InstanceType]], t2]
Transform:
- AWS::CodeStar
Parameters:
  ProjectId:
    Type: String
    Description: AWS CodeStar project ID used to name project resources
and create roles.
  InstanceType:
    Type: String
    Description: The type of Amazon EC2 Linux instances that will be
launched for this project.
  WebAppInstanceProfile:
     Type: String
     Description: The IAM role that will be created for the Amazon EC2
Linux instances.
  ImageId:
    Type: String
    Description: The Amazon EC2 Linux instance Amazon Machine Image
(AMI), which designates the configuration of the new instance.
  KeyPairName:
    Type: String
    Description: The name of an existing Amazon EC2 key pair in the
region where the project is created, which you can use to SSH into the
new Amazon EC2 Linux instances.
  VpcId:
    Type: String
    Description: The ID of the Amazon Virtual Private Cloud (VPC) to use
for Amazon EC2 instances.
  SubnetId:
    Type: String
    Description: The name of the VPC subnet to use for Amazon EC2
instances launched for this project.
  Stage:
    Type: String
    Description: The name for a project pipeline stage, such as Staging
or Prod, for which resources are provisioned and deployed.
    Default: ''
Resources:
  WebApp01:
    Description: The installation and configuration commands this
project will use to create instances that support this sample web
application.
    Properties:
      CreditSpecification:
        CPUCredits: !If [IsBurstableInstanceType, unlimited, !Ref
'AWS::NoValue']
```

```
      IamInstanceProfile: !Ref 'WebAppInstanceProfile'
      ImageId: !Ref 'ImageId'
      InstanceType: !Ref 'InstanceType'
      KeyName: !Ref 'KeyPairName'
      NetworkInterfaces:
      - AssociatePublicIpAddress: true
        DeviceIndex: 0
        GroupSet:
        - !Ref 'WebAppSG'
        SubnetId: !If
        - UseSubnet
        - !Ref 'SubnetId'
        - !Ref 'AWS::NoValue'
      Tags:
      - Key: Environment
        Value: !Sub '${ProjectId}-WebApp${Stage}'
      - Key: Name
        Value: !Sub '${ProjectId}-WebApp${Stage}'
      UserData:
        Fn::Base64:
          Fn::Sub: |
            #!/bin/bash -ex
            # Install the AWS CodeDeploy Agent.
            cd /home/ec2-user/
            wget
https://aws-codedeploy-${AWS::Region}.s3.amazonaws.com/latest/codedeploy
-agent.noarch.rpm
            yum -y install codedeploy-agent.noarch.rpm
            # Install the Amazon CloudWatch Logs Agent.
            wget
https://s3.amazonaws.com/aws-cloudwatch/downloads/latest/awslogs-agent-s
etup.py
            wget
https://s3.amazonaws.com/aws-codedeploy-us-east-1/cloudwatch/codedeploy_
logs.conf
            wget
https://s3.amazonaws.com/aws-codedeploy-us-east-1/cloudwatch/awslogs.con
f
            chmod +x ./awslogs-agent-setup.py
            python awslogs-agent-setup.py -n -r ${AWS::Region} -c
./awslogs.conf
            mkdir -p /var/awslogs/etc/config
            cp codedeploy_logs.conf /var/awslogs/etc/config/
            service awslogs restart
    Type: AWS::EC2::Instance
  WebAppSG:
    Description: The default Amazon EC2 security group that will be
created for the Amazon EC2 Linux instances.
    Type: AWS::EC2::SecurityGroup
    Properties:
```

```
      GroupDescription: Enable HTTP access via port 80 and SSH access
via port 22.
      SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: '80'
        ToPort: '80'
        CidrIp: 0.0.0.0/0
      - IpProtocol: tcp
        FromPort: '22'
```

```
        ToPort: '22'
        CidrIp: 0.0.0.0/0
      VpcId: !Ref 'VpcId'
```

- Now we have Ec2 instance for deployment, now we need to configure the instance for java app deployment in tomcat. After instace created this script runs includes(shell scripts).

```
version: 0.0 (this will deploy the s3 jar file in tomcat)
os: linux
files:
  - source: target/ROOT.war (build code is stored)
    destination: /home/ec2-user/javaapp (deployment target in ec2
instance)
hooks:
  AfterInstall:
    - location: scripts/install_dependencies(installing tomcat and java)

      timeout: 300
      runas: root
    - location: scripts/codestar_remote_access(configuring the remote
access)
      timeout: 300
      runas: root
    - location: scripts/start_server(service restart)
      timeout: 300
      runas: root
```

- The scripts are below install priority wise.

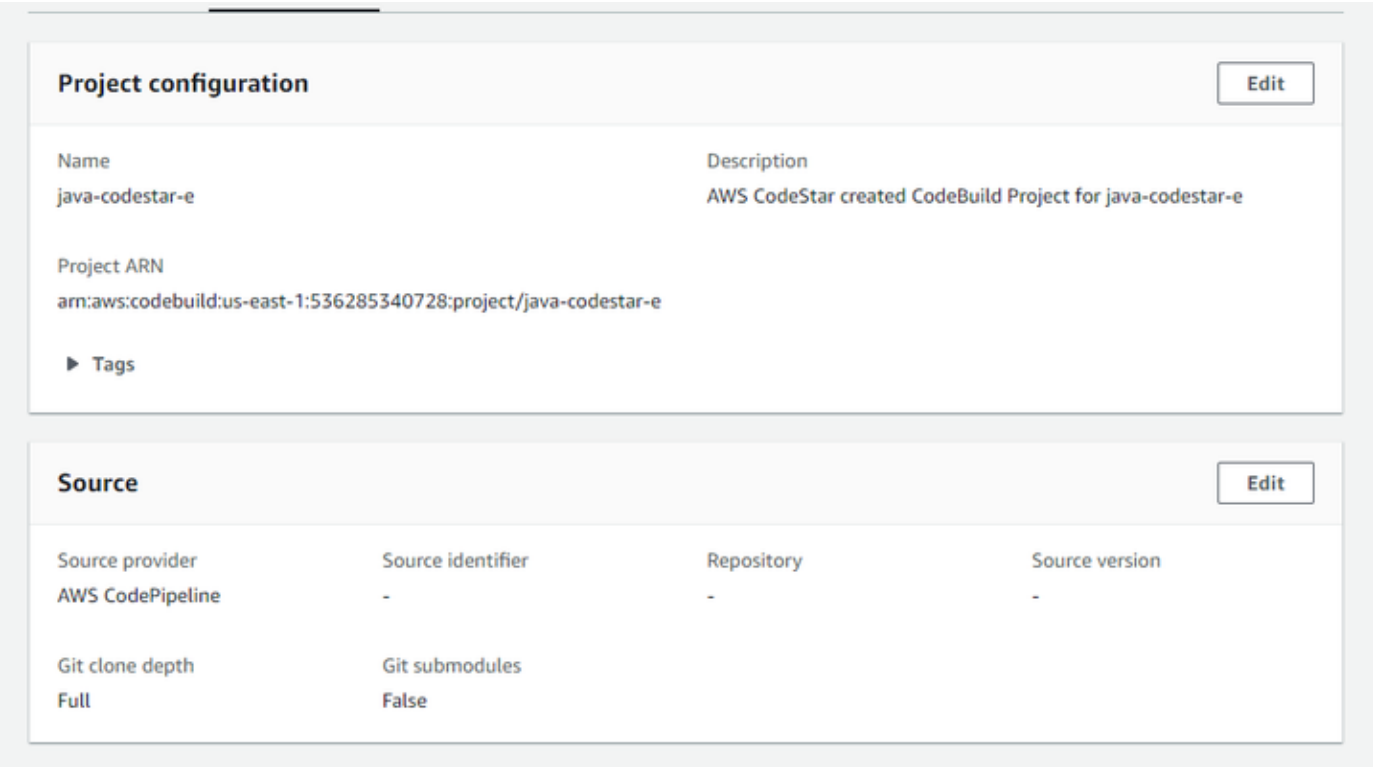
java-codestar-ec2/scripts/codestar_remoteaccess	java-codestar-ec2/scripts/install_dependencies
#!/bin/bash # Install AWS CodeStar remote access management. # Allows project members to remotely access Amazon EC2 instances running Linux associated with the project. wget -O /usr/local/bin/get_authorized_keys https://awscodestar-templates-common.s3.amazonaws.com/us-east-1/get_authorized_keys chmod 755 /usr/local/bin/get_authorized_keys sed -i '/AuthorizedKeysCommand /s/.*/AuthorizedKeysCommand /usr/local/bin/get_authorized_keys/g' /etc/ssh/sshd_config sed -i '/AuthorizedKeysCommandUser /s/.*/AuthorizedKeysCommandUser root/g' /etc/ssh/sshd_config /etc/init.d/sshd restart yum update -y aws-cfn-bootstrap yum install -y aws-cli # Install pip and python dev libraries. yum install -y python27-devel python27-pip gcc pip install boto3 pip install pycryptodome	#!/bin/bash cd /home/ec2-user/javaapp wget https://s3.amazonaws.com/dhqs-mirror-iad/ap .tar.gz mkdir /opt/tomcat tar xvf apache-tomcat-8*tar.gz -C /opt/tomcat --strip cd /opt/tomcat sed -i 's|port="8080"|port="80"|g' conf/server.xml rm -rf webapps/ROOT mv /home/ec2-user/javaapp/ROOT.war webapps/ yum -y install java-1.8.0-openjdk-1.8.0.171 alternatives --set java /usr/lib/jvm/jre-1.8.0-openjdk

- Till now we have done deployment configurations in ec2 instance. What about build part, let see codebuild.

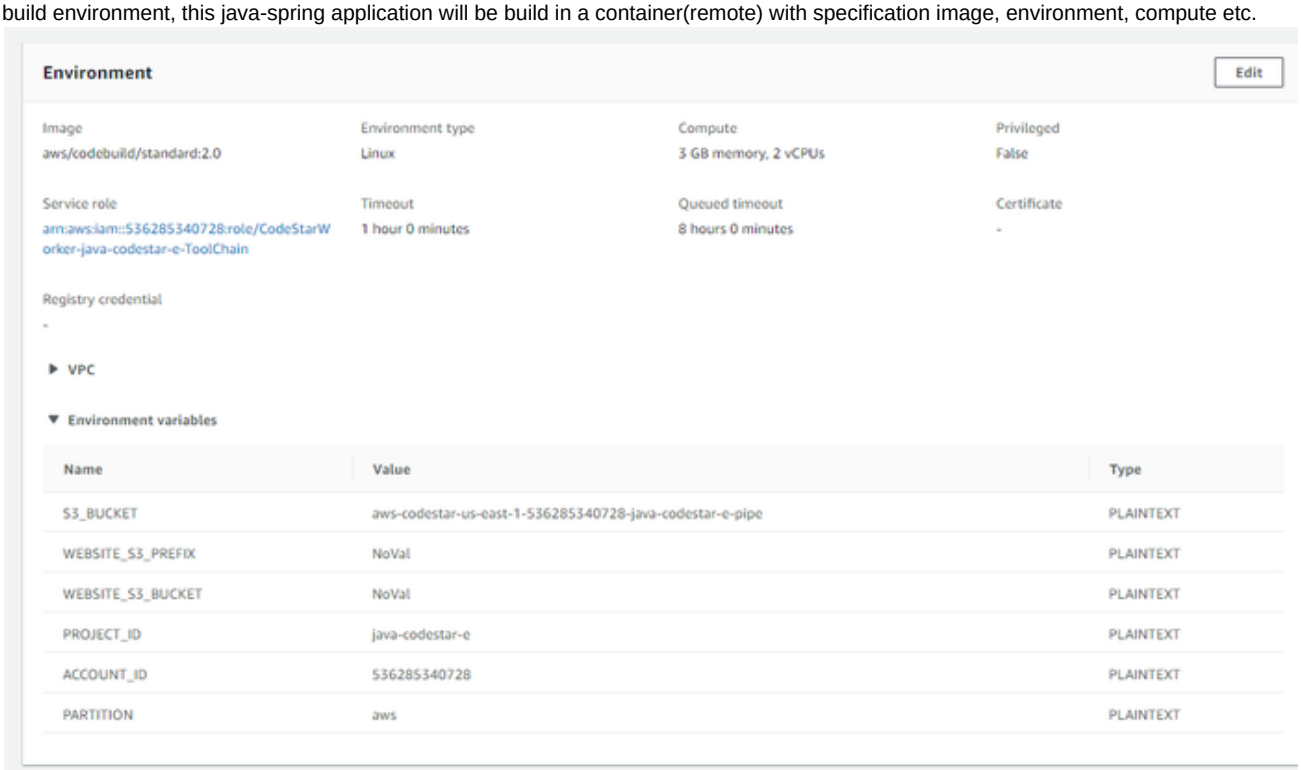
Code-build - build the java app specifications:-

- Code-build- the code will be build in containers and store the artifacts in s3 instance. So lets see the specifications in code build.
- project configurations- related to name,Project ARN, description,tags.

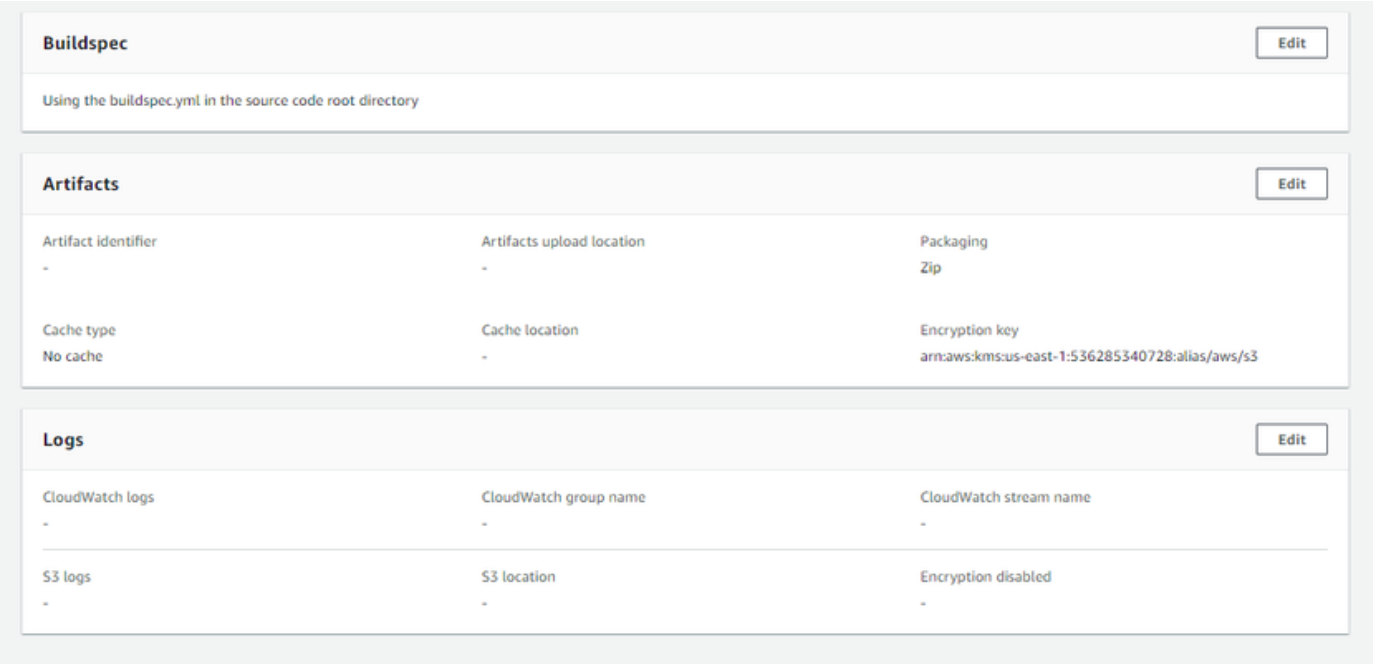
- Source- its taken default code pipeline which is reflected by code-star.



- build environment, this java-spring application will be build in a container(remote) with specification image, environment, compute etc.



- build spec- this will allow you to build and archive the application.
- Artifacts- this will store the archived files in the s3 bucket.
- logs - Cloudwatch is configure for monitoring the application.



- build steps is mentioned in the builspec.yml.
- This application is build with maven tool and sonar code quality. give the following commands to run.

```
version: 0.2

phases:
  install:
    runtime-versions:
      java: openjdk8
    commands:
      # Upgrade AWS CLI to the latest version
      - pip install --upgrade awscli
  pre_build:
    commands:
      - mvn clean compile test
  build:
    commands:
      - mvn package
      - mvn sonar:sonar
-Dsonar.host.url=http://novartis.devops.altimetrik.io:9000
-Dsonar.login=admin -Dsonar.password=admin

  post_build:
    commands:
      # Do not remove this statement. This command is required for AWS
CodeStar projects.
      # Update the AWS Partition, AWS Region, account ID and project ID
in the project ARN in template-configuration.json file so AWS
CloudFormation can tag project resources.
      - sed -i.bak
's/\$PARTITION\$/'${PARTITION}'/g;s/\$AWS_REGION\$/'${AWS_REGION}'/g;s/\
$ACCOUNT_ID\$/'${ACCOUNT_ID}'/g;s/\$PROJECT_ID\$/'${PROJECT_ID}'/g'
template-configuration.json
artifacts:
  type: zip
  files:
    - 'appspec.yml'
    - 'template.yml'
    - 'scripts/*'
    - 'target/ROOT.war'
    - 'template-configuration.json'
```

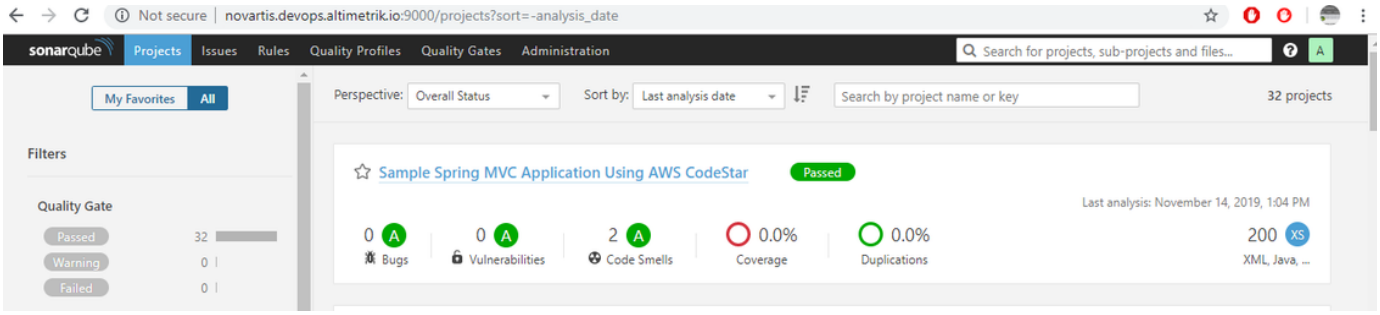
- lets look the code maven commands passing for build, test, package, code quality and archiving the files.

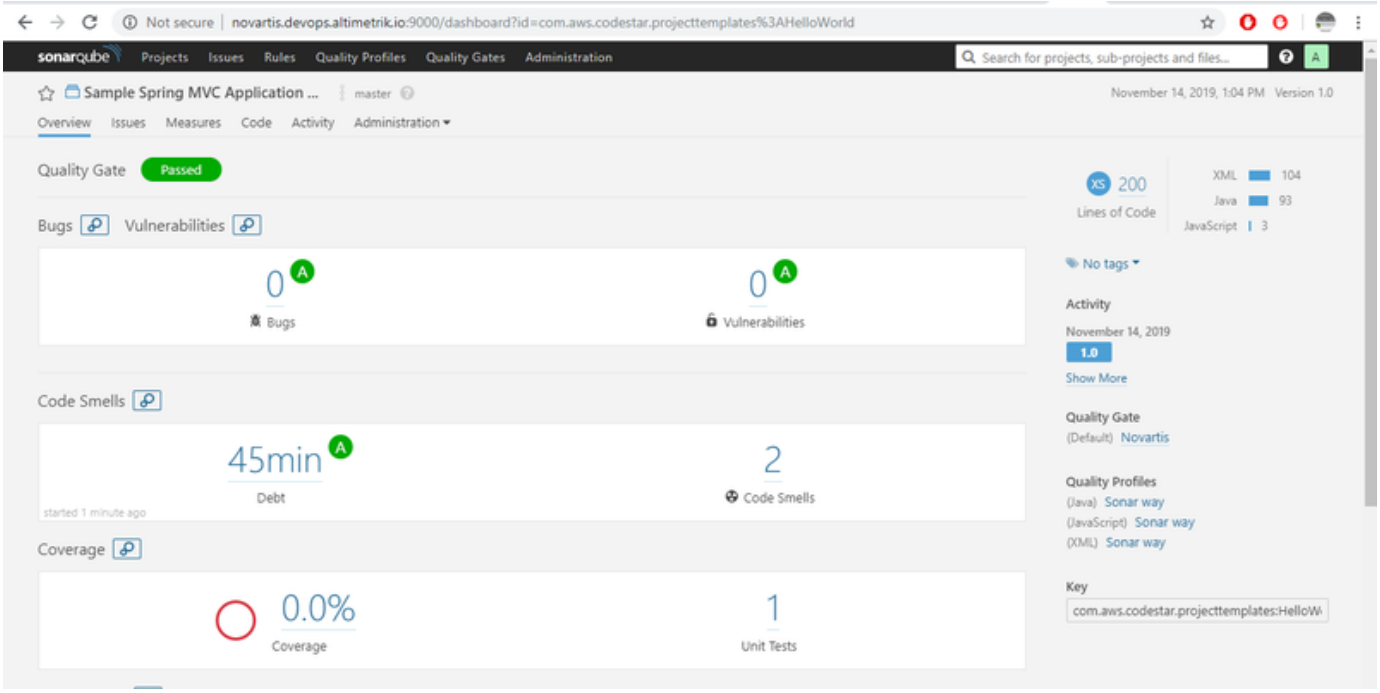
```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.aws.codestar.projecttemplates</groupId>
```

```
    <artifactId>HelloWorld</artifactId>
    <version>1.0</version>
    <packaging>war</packaging>
    <name>Sample Spring MVC Application Using AWS CodeStar</name>
    <properties>
        <junit.platform.version>1.2.0</junit.platform.version>
        <junit.jupiter.version>5.2.0</junit.jupiter.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>5.0.7.RELEASE</version>
        </dependency>
    </dependencies>
    <build>
        <pluginManagement>
            <plugins>
                <plugin>
                    <groupId>org.apache.maven.plugins</groupId>
                    <artifactId>maven-war-plugin</artifactId>
                    <version>3.2.2</version>
                    <configuration>

<warSourceDirectory>src/main/webapp</warSourceDirectory>
                        <warName>ROOT</warName>
                        <failOnMissingWebXml>false</failOnMissingWebXml>

                    </configuration>
                </plugin>
                <plugin>
                    <groupId>org.apache.maven.plugins</groupId>
                    <artifactId>maven-surefire-plugin</artifactId>
                    <version>2.22.0</version>
                    <dependencies>
                        <dependency>
                            <groupId>org.junit.platform</groupId>

<artifactId>junit-platform-surefire-provider</artifactId>
                            <version>${junit.platform.version}</version>

                        </dependency>
                    </dependencies>
                </plugin>
            </plugins>
        </pluginManagement>
```

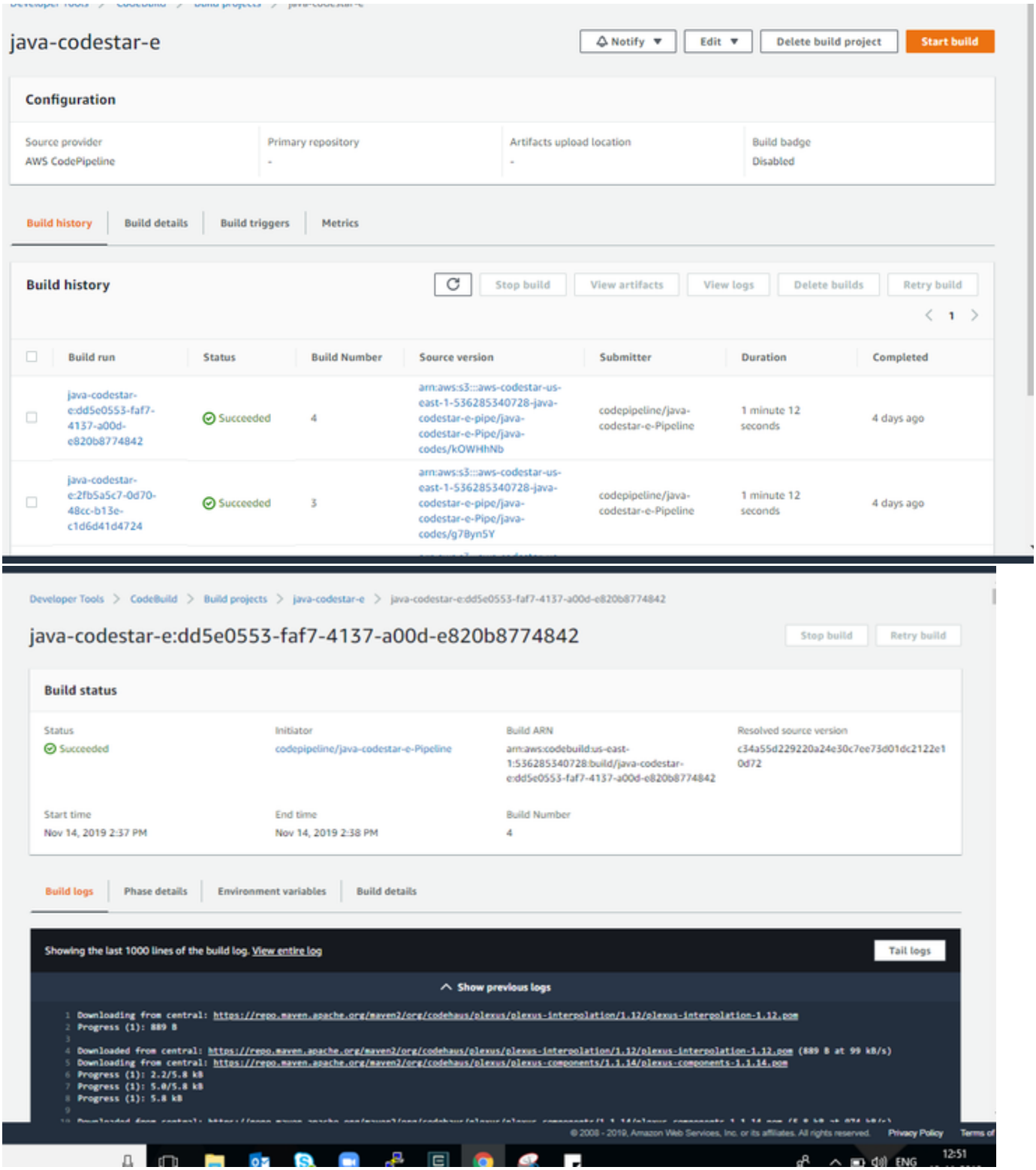
```
        <finalName>ROOT</finalName>
    </build>
</project>
```

- for code quality coverage, sonar is generating the results to sonar instance.





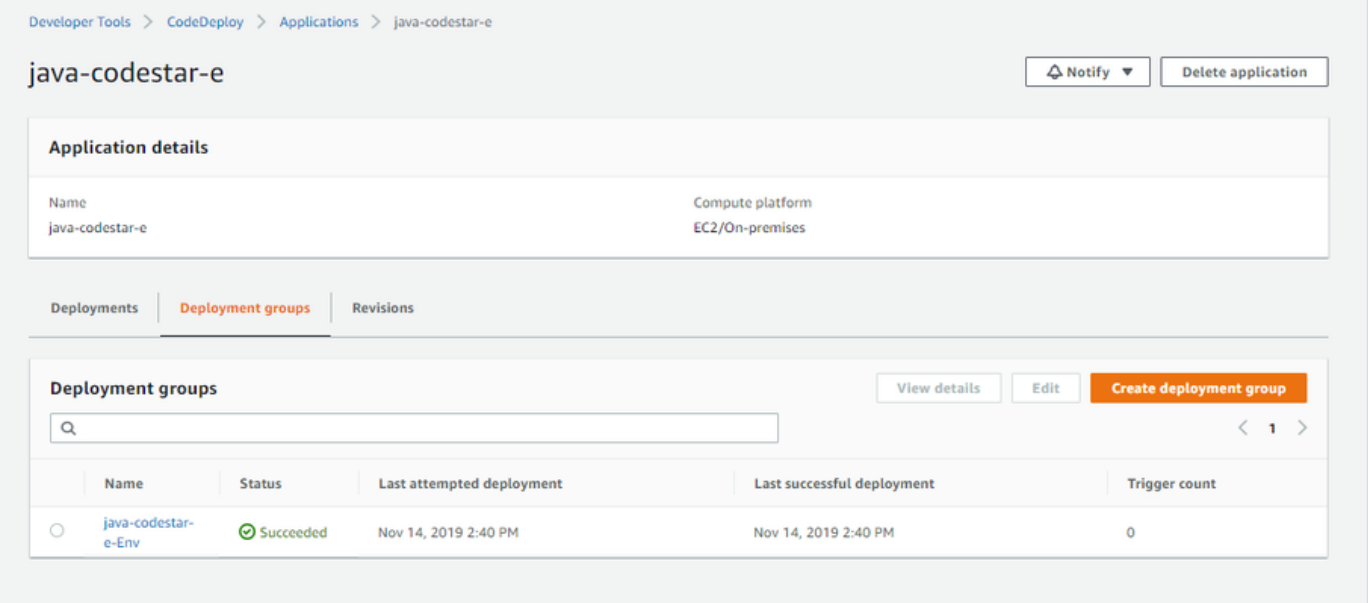
- you can able to see the build details and builds history.



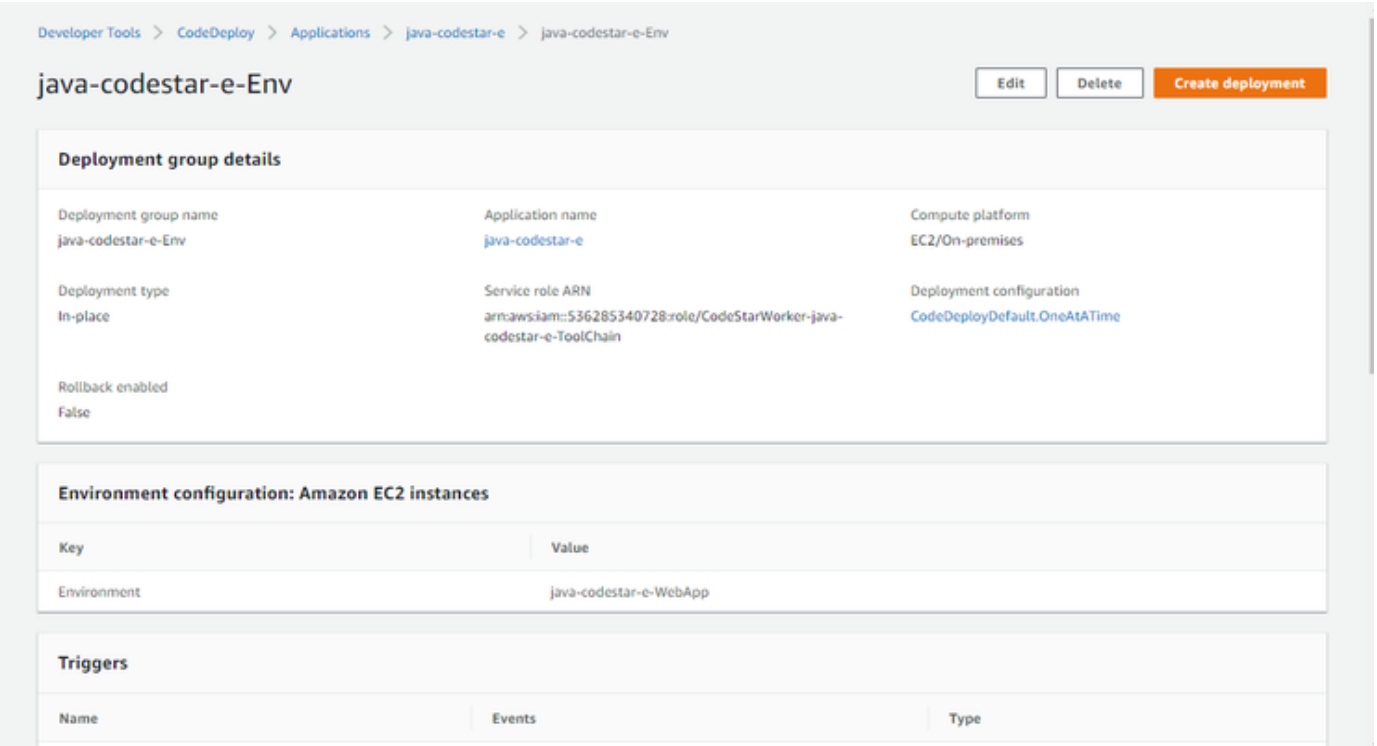
- Artifacts will be store in a S3 bucket configure by cloud formation see the codestar project resources.

Code deploy- configurations:

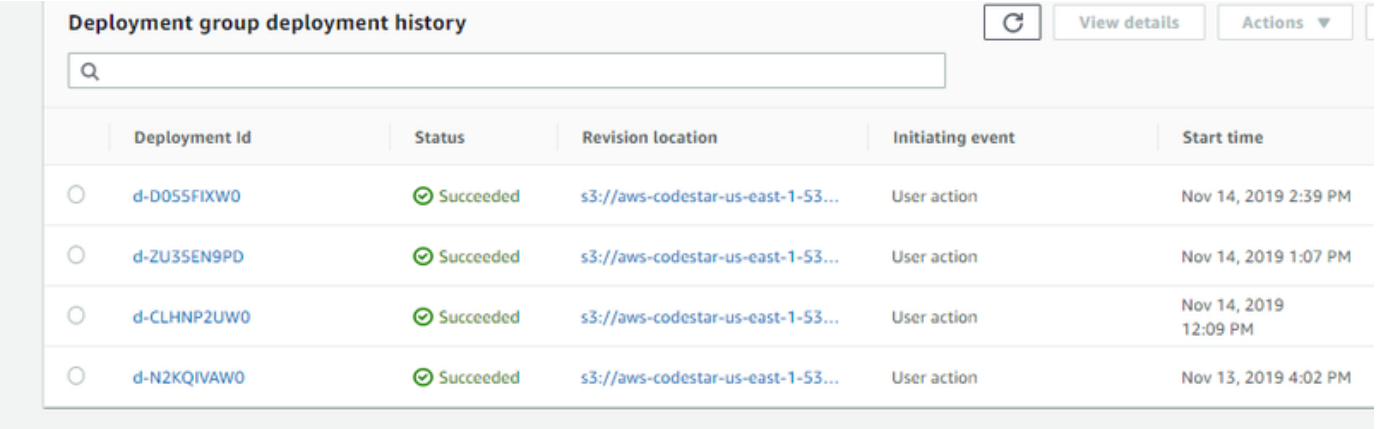
- lets go to code deploy configurations, deployment group details i.e compute platform



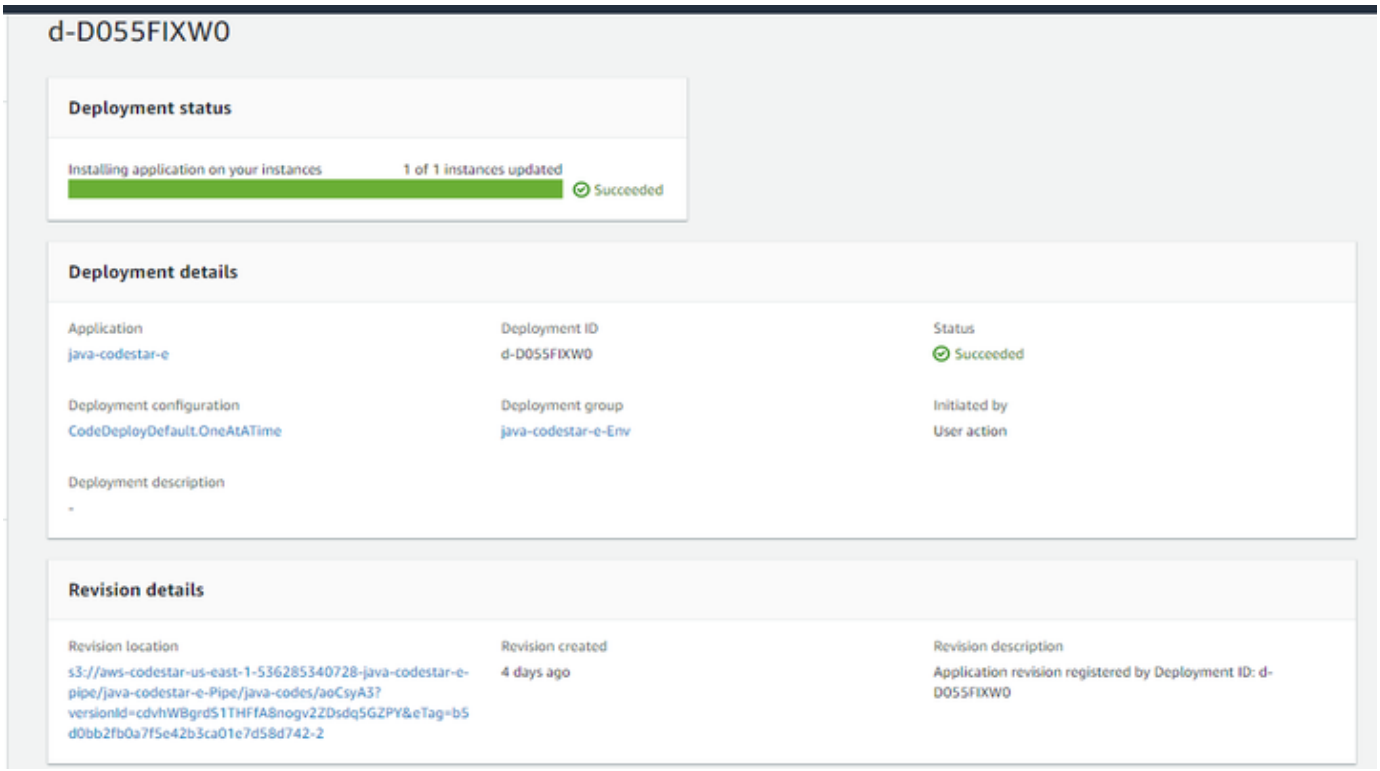
- deployment group details- compute platform(tag of cloudformation ec2).



- deployment revisions are taking from s3 bucket (archived files of build).

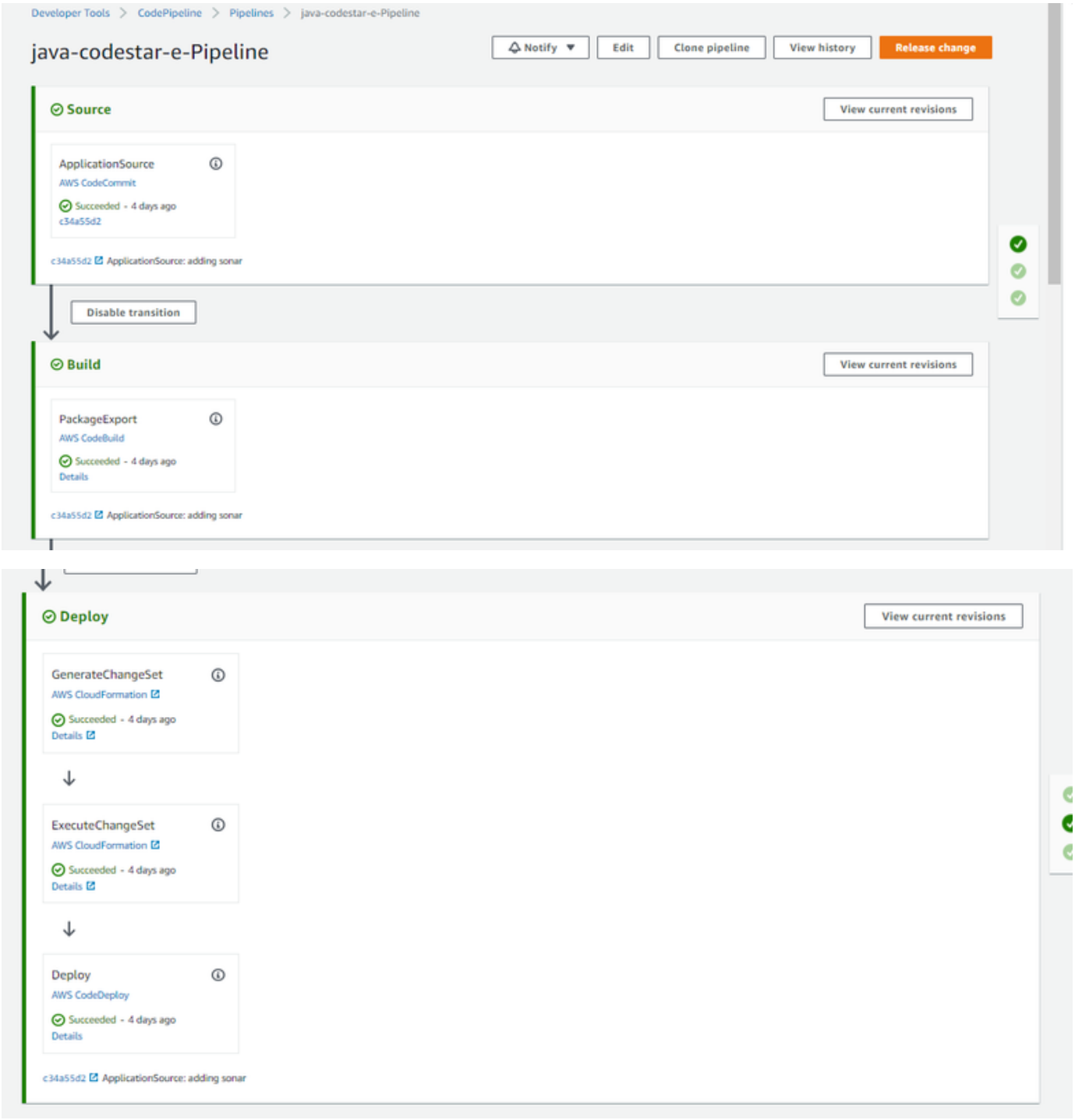


- Go to each deployment id, you can see the configurations.

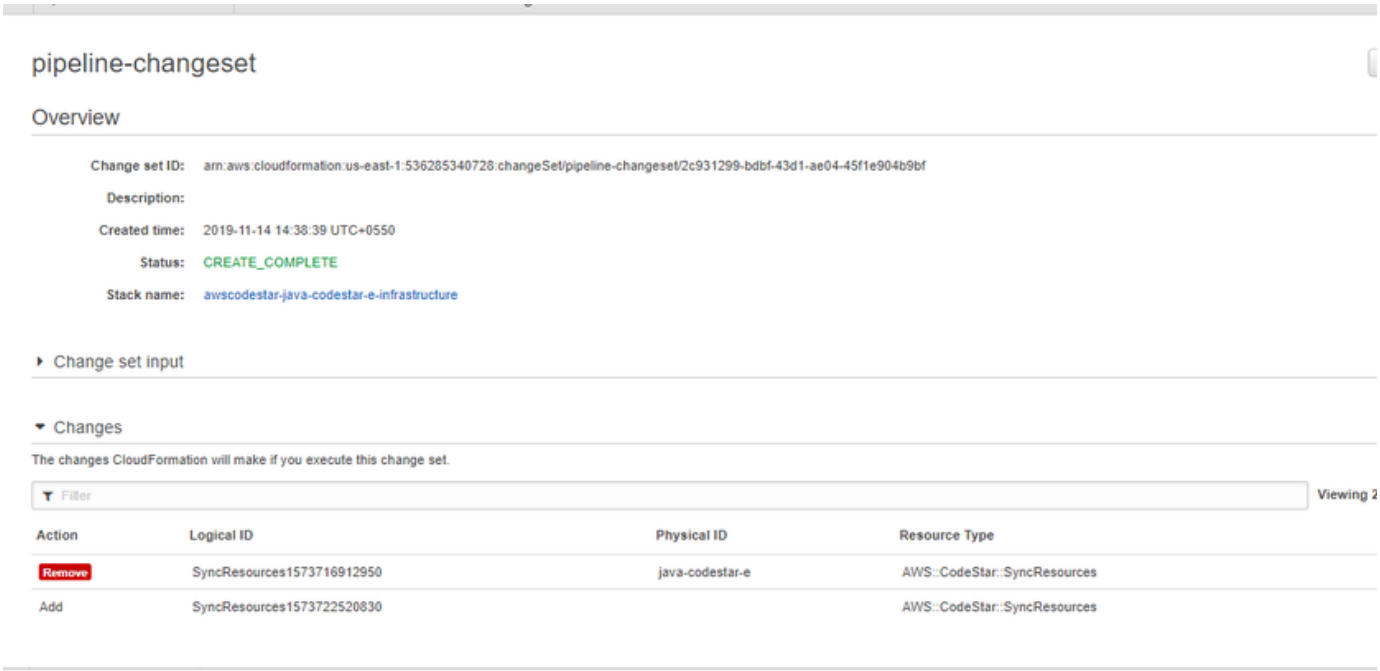


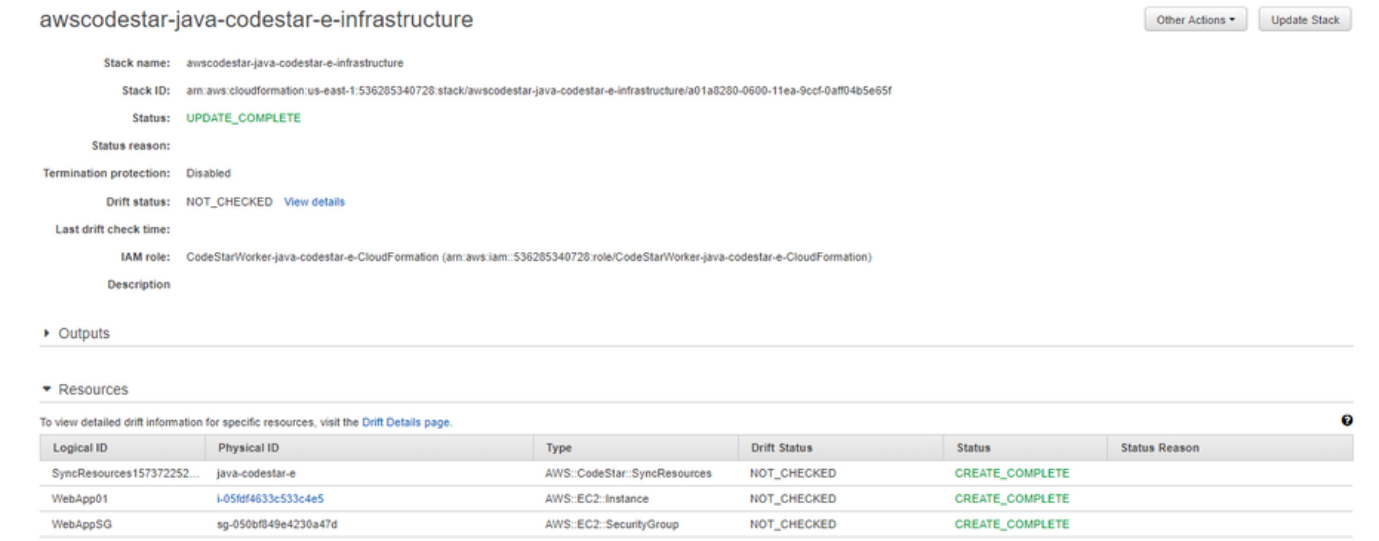
Code pipeline- configurations:

- Go to code star project code-pipeline see the stages.



- you can click each stage, it will be redirect.
- AWS cloudformation for creating ec2 instance.



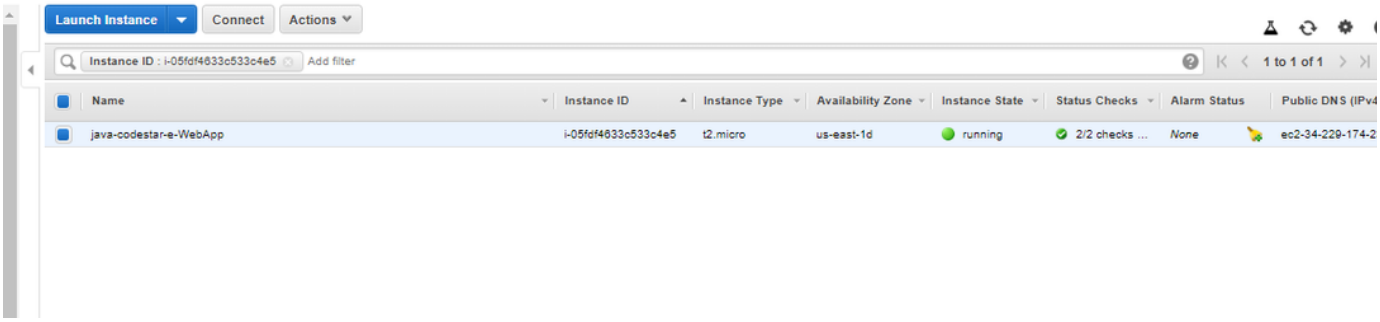


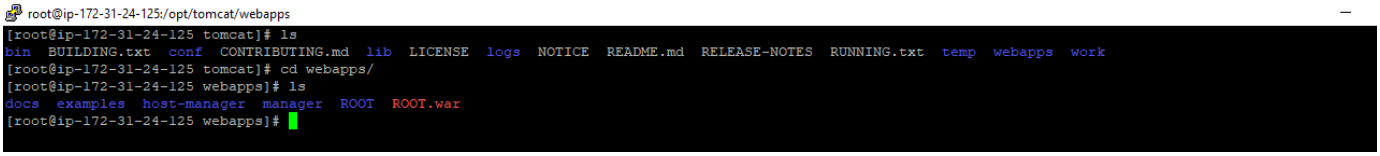
- AWS code deploy- code will be deploy here.
- click the release change in pipeline, the trigger will be started.
- The triggers can be also configure based on over requirement at specific stage level code build/code-deploy.
- we can add the stages at any specified point and need to configure the task.

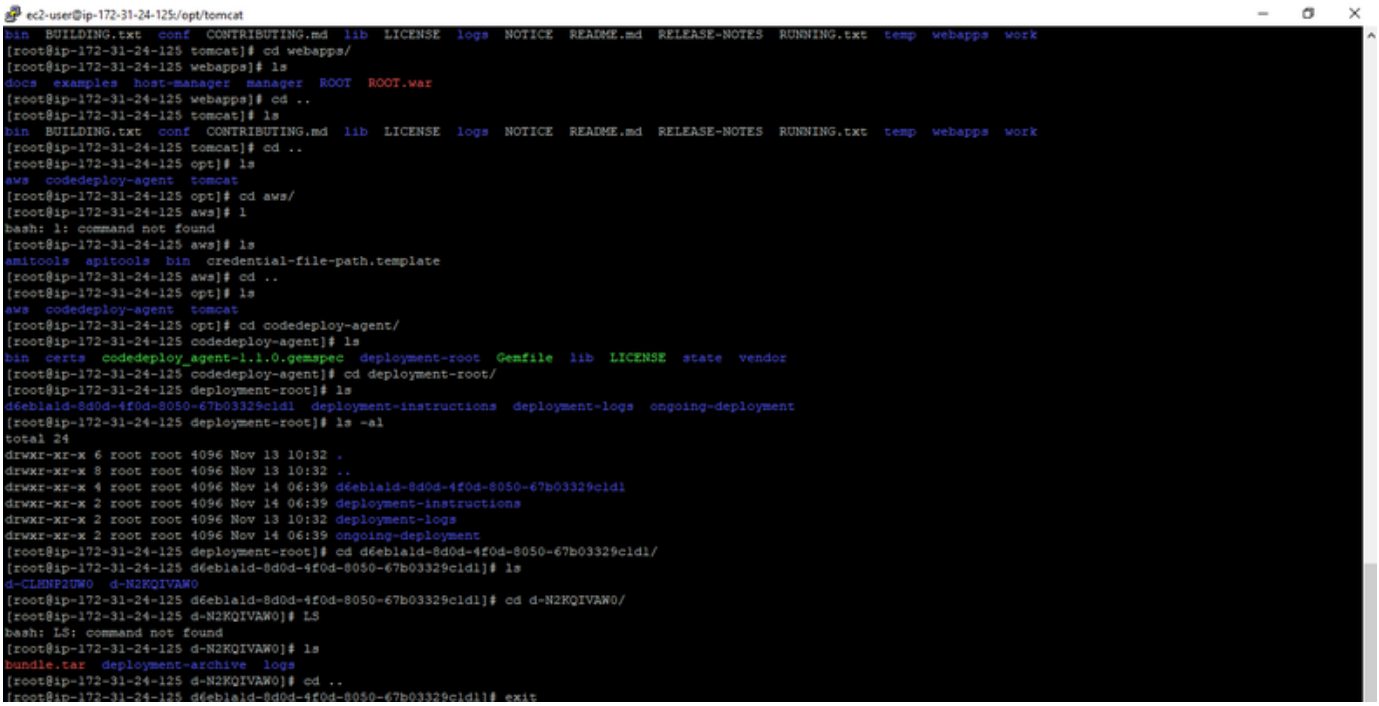
Note: What's next ? How to debug the application

How to debug the code star project application:

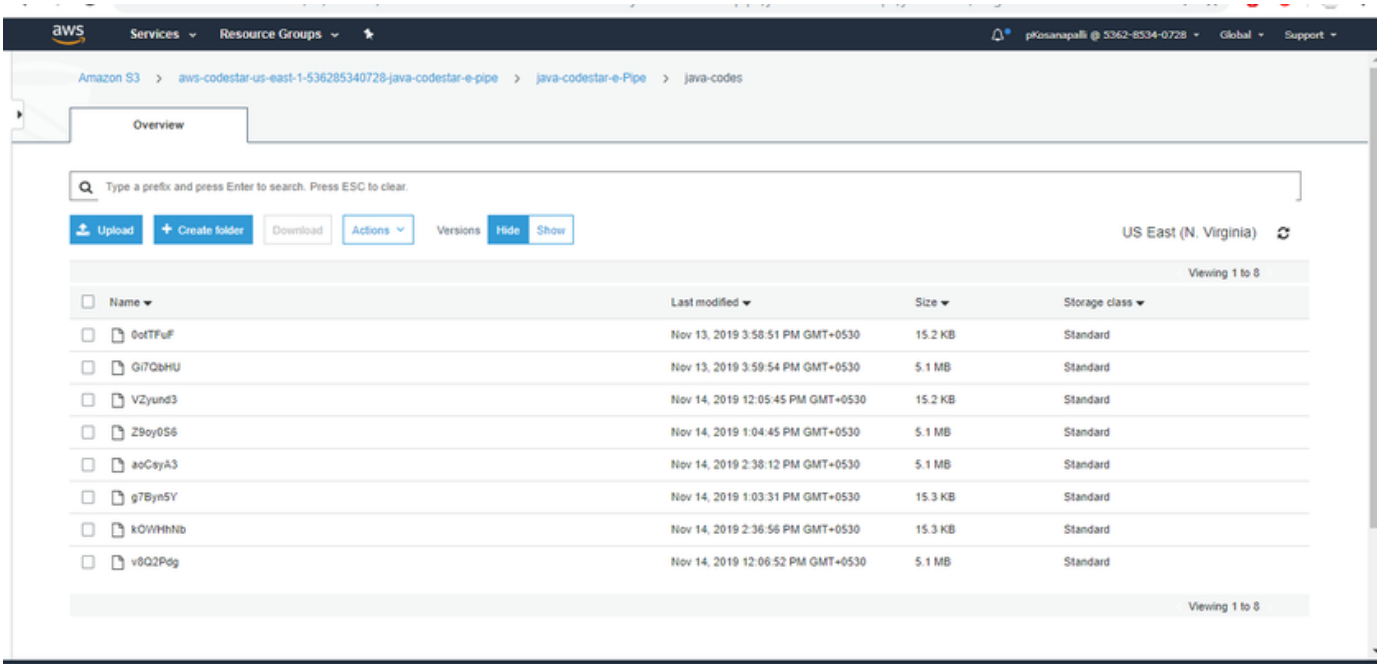
- Go to project dashboard project resources click the corresponding resources.
- click the ec2 instaces, redirect console and try to login to the instance.



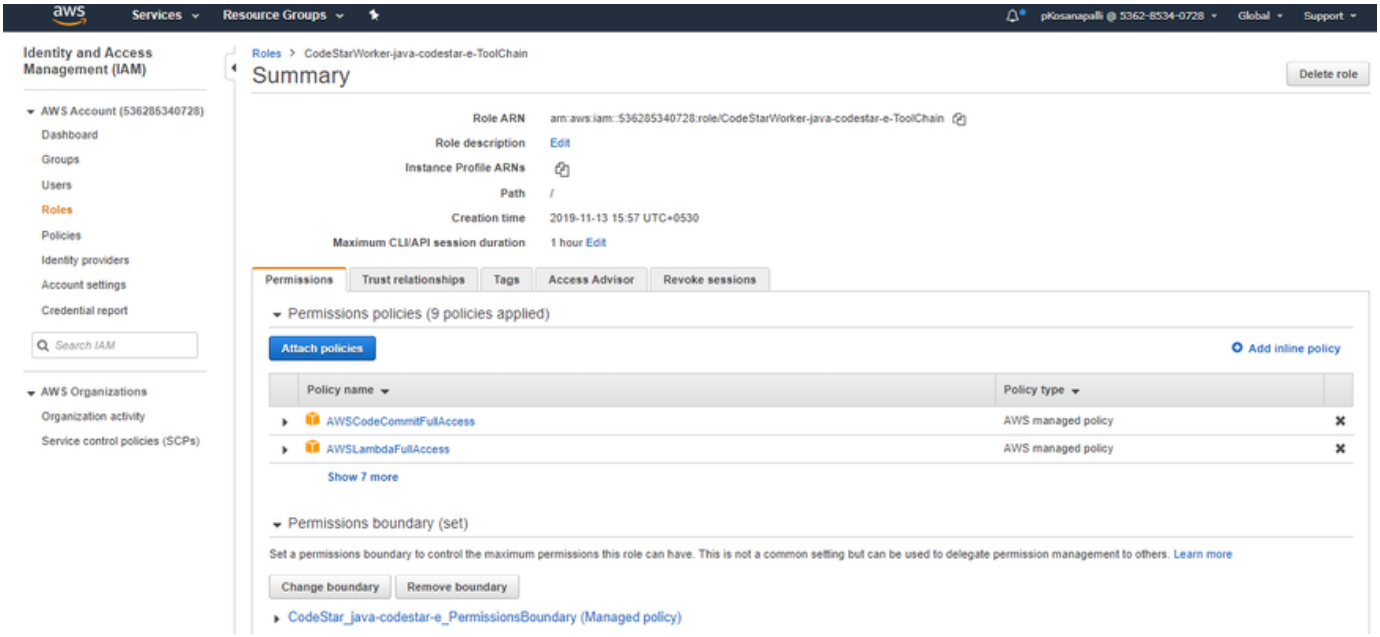




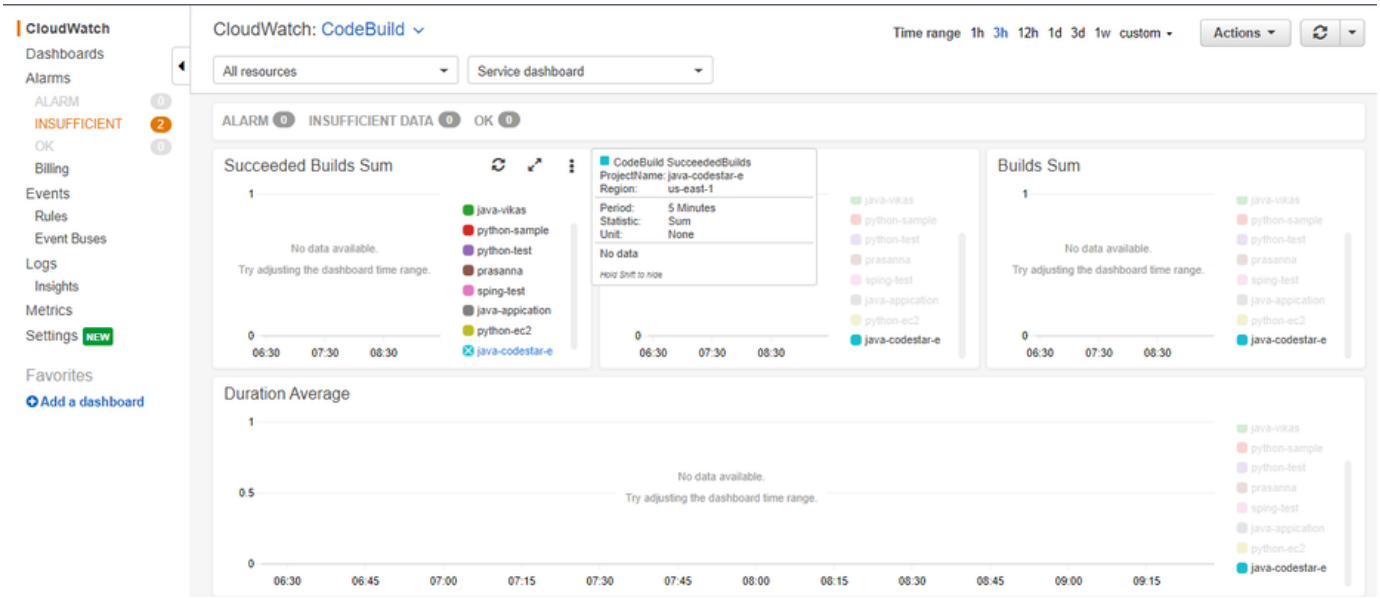
- Here we can observe the tomcat web-apps root directory, that is the war file of what we accessing the application. we can debugg if any issues facing in the application.
- Code build will store the artifacts in s3. The same artifacts will be taken as source for deployment purpose in code deploy.



- For any role based access go to IAM configurations, debug if any issues faced.



- go to cloudwatch for monitoring the code-build.



that done !!!

whats next ???

lets try the java-spring application in fargate and python django too.