How to build and deploy project from Bitbucket to AWS Code Deploy

Integration of Bitbucket with AWS CodeDeploy

This document is to explain how to integrate Bitbucket with AWS CodeDeploy. As we are aware that Bitbucket is used to store the source code. Bitbucket has come up with new services like Bitbucket Pipeline and Deploy to AWS services. With the help of these two added services one can build the code and deploy the code to other cloud services like AWS, Azure where they keep their production servers.

Once the developer commits the code in Bitbucket the Pipeline service activates and creates a build. Once the build is successful, we can deploy the build directly to AWS CodeDeploy. So that the changes done in the application will reflect on the go.

This document explains on how to configure Bitbucket with AWS CodeDeploy.

Procedure to configure Bitbucket with AWS CodeDeploy.
Prerequisite:

1. Bitbucket account
2. AWS account
3. Working source code.

Procedure:

Step 1: Login into Bitbucket and upload Source Code

1. Login into Bitbucket account and upload the source code. At the time of creating this document I am uploaded a Java-maven based project into Bitbucket.

Step2: Install add-ons for Bitbucket.

1. Once logged into Bitbucket go to user settings -> INTEGRATIONS AND FEATURES -> Find Integrations and here select Pipelines and AWS Codedeploy and install them.

Step3: Configure Bitbucket Pipeline

1. Select the repository on which you want to configure Pipelines.
2. On the left pane select Pipelines, a new window opens in the right side. Here click **Enable Pipeline** button.
3. A new windows will be opened, here choose a template to create .yml file. As my project is **Java-Maven** project I have chosen it. Once we select the template the sample code of bitbucket-pipelines.yml will be displayed click **Next** button.
4. Now we can edit this and export bitbucket-pipeline.yml file so that a new container will be created and our code will be build on this container in Bitbucket itself.
5. Now commit on this or source files, then Bitbucket Pipeline will automatically build and give the build result whether success or failure message.

Step4: Create Bitbucket Service Role in AWS

1. Now login into AWS and select service IAM service.
2. Now select Roles.
3. Click button **Create new role.**
4. Select role type as **AWS Service Role** -> **AWS CodeDeploy** and press Select
5. Now attach policy **AWSCodeDeployRole** and press Next Step
6. Now give the role name as **BitbucketServiceRole** and press create role button.
7. Once the role is created select the role and go to Trust relationships options and update the policy as below

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "ec2.amazonaws.com",
          "codedeploy.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Note: This role is used to connect the AWS CodeDeploy service with AWS EC2 service.

Step5: Now create EC2CodeDeploy role

1. Now go to IAM services and select Roles in it.
2. Click **create new role** button.
3. Select Role type as **Amazon EC2**.
4. Now simply press Next Step button (do not select any policy)
5. Now give an name to this role as **EC2CodeDeploy** and create a new role.
6. Once the role is created open the role. Here in Permissions tab select **Inline Policies**. Here select create role policy button-> Custom policy->Policy name :EC2CodeDeploy and add below policy.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```
Note : This role is to connect AWS EC2 with AWS S3 services.

Step6: Now create BitbucketCodeDeployAddon role

1. Login into Bitbucket and select the repository which we want to deploy to AWS.
2. In the settings of the repository we can see Codedeploy settings.
3. Here we can see AWS **Account ID** and **External ID** note these ID's.
4. Now login into AWS and Select IAM service.
5. Create a new role.
6. Select role type: Role for cross-account access -> provide access between AWS account and 3rd party AWS account.
7. Now enter the Account ID and external ID we got from Bitbucket repository settings.
8. Create a role **BitbucketCodeDeployAddon**.
9. Now select the role and in permissions tab Inline Policies create a new policy and add below code to the policy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListAllMyBuckets",
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "codedeploy:*"
      ],
      "Resource": "*"
    }
  ]
```

}

Note: This role is to connect Bitbucket with AWS Codedeploy and S3, the above policy is avaliable in Bitbucket codedeploy settings options.

Step7: Create an new instance and install codedeploy agent and software's needed.

1. Login into AWS and go to EC2 service.
2. Select Launch EC2 instance
3. Choose AMI here select free tier Ubuntu server and press select.
4. Choose instance type as t2.micro
5. Now open configuration and here in IAM role select **EC2Codedeploy** role
6. Add storage of 8GB
7. Now add tags key: name and value: BitbucketEC2
8. Now select the security group if you already have or create one. This security group allows the traffic which we release from certain ports.
9. Once every thing is done launch the instance.

Note: When we attach EC2codedeploy role our EC2 instance can access S3 service.

Step8: Create S3 bucket

1. Login into AWS and go to S3 service.
2. Create a new bucket(Ex: javatestcode)
3. Now in bucket create a file (Ex: sourcecode)
4. Select the code and we can see the key of file on right side, save it.

Step9: Configure AWS CodeDeploy

1. Login into AWS and select CodeDeploy service.
2. First create an Application. Here give application name(Ex:Bitbucket) and Deployment group name (Ex: Bitbucket-Test).
3. Deployment type: In-place deployment

4. Add Instances key:name value the instance we created above.
5. Service role ARN give **BitbucketServiceRole** and create an application.

Step10: Configure Bitbucket Codedeploy settings

1. In bitbucket codedeploy settings request for role ARN and region.
2. Here give the ARN of BitbucketCodeDeployAddon and region as ap-south-1(this region must be of S3 bucket region)
3. Now it will ask for Application name (Ex:Bitbucket) and S3 bucket where we want to store code in S3(Ex: javatestcode).
4. Save the settings.
5. Now open the commits in Bitbucket repository and select latest commit. In the right side we can see Deploy to AWS. If we select this option then we can push code from bitbucket to AWS Codedeploy.

Note: The above procedure just builds the code in Pipeline and gives success or failure result. When we click **deploy to aws** this will simply push source code from bitbucket to S3 and from S3 to code deploy agent in EC2 instance. Thus we can simply move source code from Bitbucket to AWS code deploy.

Deploy Pipeline build to AWS code deploy

In the above case we simply moved the source code from Bitbucket to AWS code deploy but if you build a project in .jar or .war file we need to place the war file in tomcat webapps folder so that it will be accessible through web browser. To do this

Step1: Follow all the steps above.

Step2: Create a user account in AWS and assign permission to the user so that we can we can deploy build war file in pipeline to aws codedeploy directly.

1. Login into AWS account and go to service IAM.
2. Select users option on the left side and click **Add user** button.
3. Give user name (Ex:Bitbucketuser) and access type programmatic access.

4. Next in permission leave default and click Next:Review
5. Finally click create user.
6. Once user account is created its security credentials file will be created. Save the file.(Bitbucketuser.csv)
7. Now go to users and select (Ex:Bitbucketuser) user and in permission tab click Add Inline policy.
8. Now set permissions Custom policy->select
9. Policy name (Ex:Bitbucketuser) and policy doc as below

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "autoscaling:*",
        "codedeploy:*",
        "ec2:*",
        "elasticloadbalancing:*",
        "iam:AddRoleToInstanceProfile",
        "iam:CreateInstanceProfile",
        "iam:CreateRole",
        "iam:DeleteInstanceProfile",
        "iam:DeleteRole",
        "iam:DeleteRolePolicy",
        "iam:GetInstanceProfile",
        "iam:GetRole",
        "iam:GetRolePolicy",
        "iam:ListInstanceProfilesForRole",
        "iam:ListRolePolicies",
        "iam:ListRoles",
        "iam:PassRole",
        "iam:PutRolePolicy",
        "iam:RemoveRoleFromInstanceProfile",
        "s3:*"
      ],
      "Resource": "*"
    }
  ]
}
```

Step3: Edit the bitbucket-pipeline.yml file.

Now we need to add commands to bitbucket-pipeline.yml file so that when a build starts the commands in the file will be executed and builds the war file and deploys it to aws codedeploy service. Mine is a java-maven project so my .yml will is like below

```
# This is a sample build configuration for Java – Maven.
# Check our guides at https://confluence.atlassian.com/x/zd-5Mw for more examples.
# Only use spaces to indent your .yml configuration.
# -----
# You can specify a custom docker image from Docker Hub as your build environment.
image: maven:3.3.9-jdk-7

pipelines:
  default:
    - step:
        caches:
          - maven
        script: # Modify the commands below to build your repository
         - apt-get update && apt-get install -y python-dev
         - curl -O https://bootstrap.pypa.io/get-pip.py
         - python get-pip.py
         - pip install awscli
         - mvn -f /pom.xml -B verify # -B batch mode makes Maven less verbose
         - export AWS_ACCESS_KEY_ID=<Bitbucketuser ID>
         - export AWS_SECRET_ACCESS_KEY=<Bitbucketuser secret key>
         - export AWS_DEFAULT_REGION=ap-south-1 <region where S3 bucket present>
         - aws deploy push --application-name <application-name> --s3-location s3://<bucket name>/<key> --ignore-hidden-files
         - aws deploy create-deployment --application-name <application name> --s3-location bucket=<bucket name> ,key=<key>,bundleType=zip --deployment-group-name <group name>
```

Note: The above .yml file will build the .war file using maven build tool. Then it will install python, pip and finally awscli (amazon cmd line). Then aws cmds will deploy project + build war file to S3. Then aws codedeploy takes the file from S3 to EC2 instance.

Step4: Create and Edit appspec.yml file in source code.

Since we need to place the war file from project_name/targert/project.war(this is the path where maven puts the build war file) to /opt/tomcat7/webapps folder we need to update the same to Bitbucket pipeline so update the same in **appspec.yml file**. My sample file is like below

```
version: 0.0
os: linux
files:
  - source: /<project>/target/<project.war>
     destination: /opt/tomcat7/webapps
```

The EC2 instance check appspec.yml file and copies the .war file.
Now open the web browser enter the public IP address of EC2 instance
Ex: publicip:8080/project_name

This should display our web app.