**Code pipeline Setup for Deploying CloudFormation templates**

**Introduction to Cloudformation :**

**AWS CloudFormation enables you to create and provision AWS infrastructure deployments predictably and repeatedly. It helps you leverage AWS products such as Amazon EC2, Amazon Elastic Block Store, Amazon SNS, Elastic Load Balancing, and Auto Scaling to build highly reliable, highly scalable, cost-effective applications in the cloud without worrying about creating and configuring the underlying AWS infrastructure. AWS CloudFormation enables you to use a template file to create and delete a collection of resources together as a single unit (a stack).**

**AWS CloudFormation is a service that helps you model and set up your AWS resources so that you can spend less time managing those resources and more time focusing on your applications that run in AWS. You create a template that describes all the AWS resources that you want (like Amazon EC2 instances or Amazon RDS DB instances), and CloudFormation takes care of provisioning and configuring those resources for you. You don't need to individually create and configure AWS resources and figure out what's dependent on what; CloudFormation handles that.**

**So we can also setup a CI-CD pipeline for AWS cloudformation. So, whenever a stack is updated it automatically starts the pipeline and the resources will deployed. For CI-CD pipeline, we are going to use Codepipeline. Let us see the basics of code pipeline below**

**Introduction to Codepipeline :**

**AWS CodePipeline is a fully managed**[**continuous delivery**](https://aws.amazon.com/devops/continuous-delivery/)**service that helps you automate your release pipelines for fast and reliable application and infrastructure updates. CodePipeline automates the build, test, and deploy phases of your release process every time there is a code change, based on the release model you define. This enables you to rapidly and reliably deliver features and updates. You can easily integrate AWS CodePipeline with third-party services such as GitHub or with your own custom plugin. With AWS CodePipeline, you only pay for what you use.**

**Benifits of Codepipeline :**

**Rapid delivery :**

**AWS CodePipeline automates your software release process, allowing you to rapidly release new features to your users. With CodePipeline, you can quickly iterate on feedback and get new features to your users faster.**

**Automating your build, test, and release process allows you to quickly and easily test each code change and catch bugs while they are small and simple to fix. You can ensure the quality of your application or infrastructure code by running each change through your staging and release process.**

**Configurable workflow :**

**AWS CodePipeline allows you to model the different stages of your software release process using the console interface, the AWS CLI, AWS CloudFormation, or the AWS SDKs. You can easily specify the tests to run and customi ze the steps to deploy your application and its dependencies.**

**Easy to integrate**

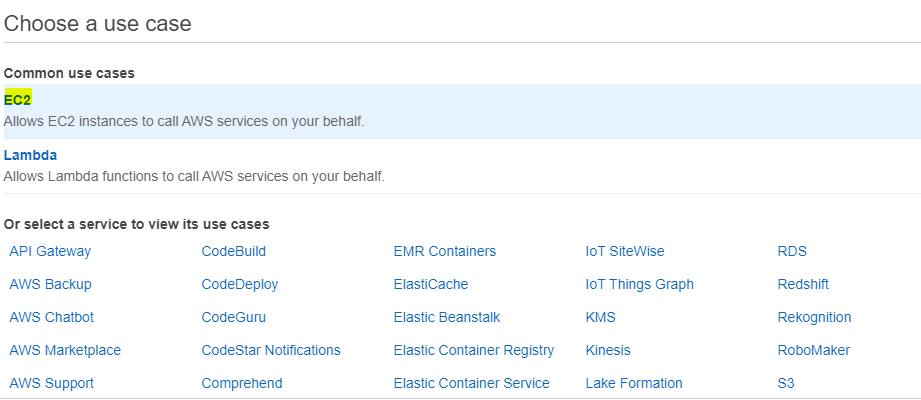
**AWS CodePipeline can easily be extended to adapt to your specific needs. You can use our pre-built plugins or your own custom plugins in any step of your release process. For example, you can pull your source code from GitHub, use your on-premises Jenkins build server, run load tests using a third-party service, or pass on deployment information to your custom operations dashboard.**

**In-Order to automate cloudformation deployment with codepipeline, follow the below steps :**

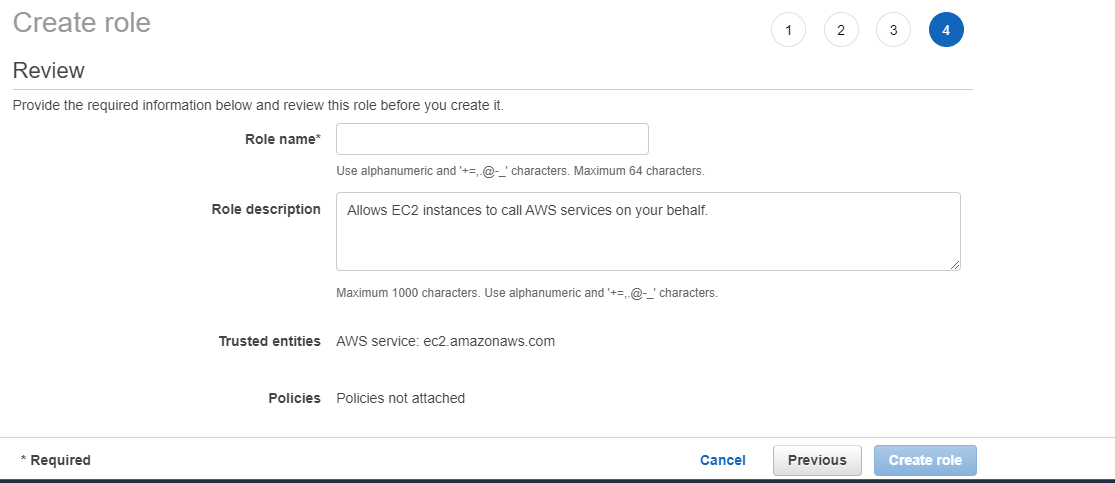
**Step 1 :**

**In order to create a code pipeline, we want to create a service role.**

* **Go to IAM -> Roles -> create role**
* **Click on Allow EC2 (Common use cases) as trusted identity as we donot have codepipeline as trusted identity. Click next**

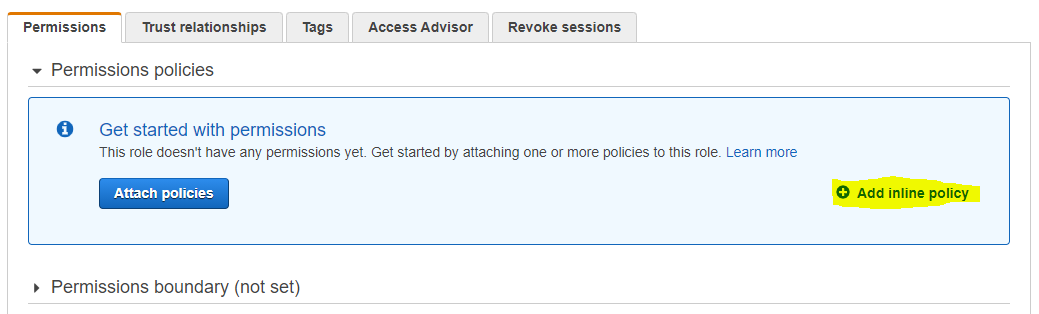


* **Do not attach any permissions. Click on next**
* **Click on review**
* **Enter the role name (Codepipeline role)**



**Now the role will be created.**

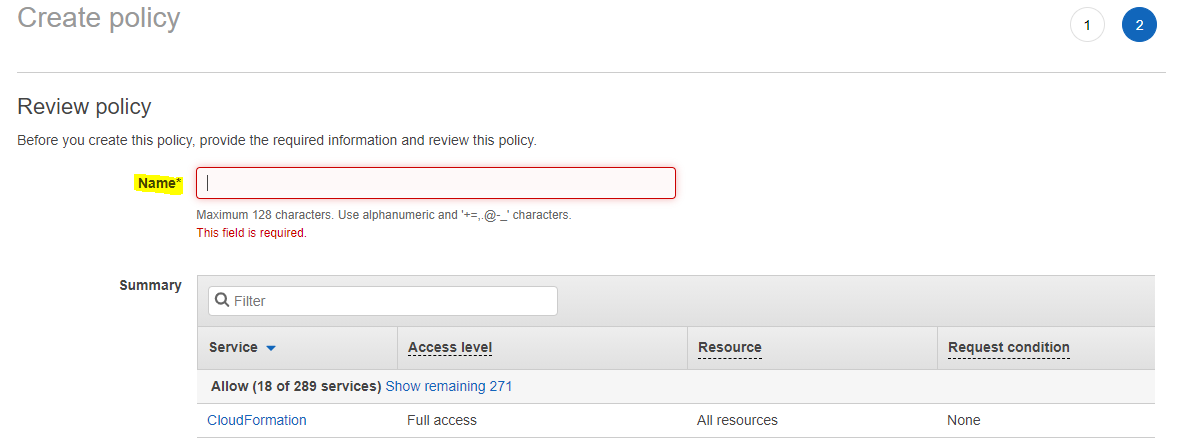
* **Go to IAM and open the code pipeline role created. Under permission, click on “Add Inline policy”**



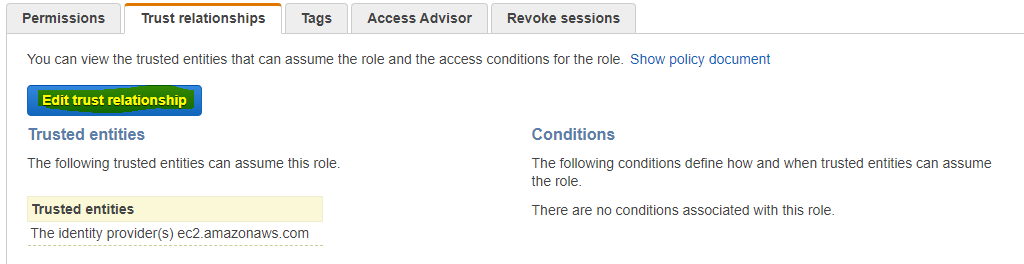
* **Click on json and add the below policy**

|  |  |
| --- | --- |
|  | { |
|  | "Statement": [ |
|  | { |
|  | "Action": [ |
|  | "s3:GetObject", |
|  | "s3:GetObjectVersion", |
|  | "s3:GetBucketVersioning" |
|  | ], |
|  | "Resource": "\*", |
|  | "Effect": "Allow" |
|  | }, |
|  | { |
|  | "Action": [ |
|  | "s3:PutObject" |
|  | ], |
|  | "Resource": [ |
|  | "arn:aws:s3:::codepipeline\*", |
|  | "arn:aws:s3:::elasticbeanstalk\*" |
|  | ], |
|  | "Effect": "Allow" |
|  | }, |
|  | { |
|  | "Action": [ |
|  | "codecommit:CancelUploadArchive", |
|  | "codecommit:GetBranch", |
|  | "codecommit:GetCommit", |
|  | "codecommit:GetUploadArchiveStatus", |
|  | "codecommit:UploadArchive" |
|  | ], |
|  | "Resource": "\*", |
|  | "Effect": "Allow" |
|  | }, |
|  | { |
|  | "Action": [ |
|  | "codedeploy:CreateDeployment", |
|  | "codedeploy:GetApplicationRevision", |
|  | "codedeploy:GetDeployment", |
|  | "codedeploy:GetDeploymentConfig", |
|  | "codedeploy:RegisterApplicationRevision" |
|  | ], |
|  | "Resource": "\*", |
|  | "Effect": "Allow" |
|  | }, |
|  | { |
|  | "Action": [ |
|  | "elasticbeanstalk:\*", |
|  | "ec2:\*", |
|  | "elasticloadbalancing:\*", |
|  | "autoscaling:\*", |
|  | "cloudwatch:\*", |
|  | "s3:\*", |
|  | "sns:\*", |
|  | "cloudformation:\*", |
|  | "rds:\*", |
|  | "sqs:\*", |
|  | "ecs:\*", |
|  | "iam:PassRole" |
|  | ], |
|  | "Resource": "\*", |
|  | "Effect": "Allow" |
|  | }, |
|  | { |
|  | "Action": [ |
|  | "lambda:InvokeFunction", |
|  | "lambda:ListFunctions" |
|  | ], |
|  | "Resource": "\*", |
|  | "Effect": "Allow" |
|  | }, |
|  | { |
|  | "Action": [ |
|  | "opsworks:CreateDeployment", |
|  | "opsworks:DescribeApps", |
|  | "opsworks:DescribeCommands", |
|  | "opsworks:DescribeDeployments", |
|  | "opsworks:DescribeInstances", |
|  | "opsworks:DescribeStacks", |
|  | "opsworks:UpdateApp", |
|  | "opsworks:UpdateStack" |
|  | ], |
|  | "Resource": "\*", |
|  | "Effect": "Allow" |
|  | }, |
|  | { |
|  | "Action": [ |
|  | "cloudformation:CreateStack", |
|  | "cloudformation:DeleteStack", |
|  | "cloudformation:DescribeStacks", |
|  | "cloudformation:UpdateStack", |
|  | "cloudformation:CreateChangeSet", |
|  | "cloudformation:DeleteChangeSet", |
|  | "cloudformation:DescribeChangeSet", |
|  | "cloudformation:ExecuteChangeSet", |
|  | "cloudformation:SetStackPolicy", |
|  | "cloudformation:ValidateTemplate", |
|  | "iam:PassRole" |
|  | ], |
|  | "Resource": "\*", |
|  | "Effect": "Allow" |
|  | }, |
|  | { |
|  | "Action": [ |
|  | "codebuild:BatchGetBuilds", |
|  | "codebuild:StartBuild" |
|  | ], |
|  | "Resource": "\*", |
|  | "Effect": "Allow" |
|  | } |
|  | ], |
|  | "Version": "2012-10-17" |
|  | } |

* **Click on Review and enter the Policy Name and create the policy.**



* **Then Under our codepipeline role, click on “Trusted Relationship “**



* **Replace the policy with the below policy and save it.**

{

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

"Statement": [

{

"Effect": "Allow",

"Principal": {

"Service": "codepipeline.amazonaws.com"

},

"Action": "sts:AssumeRole"

}

]

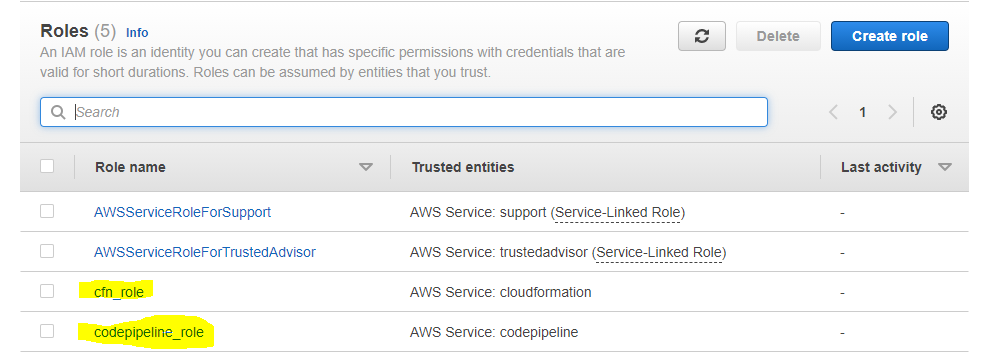
}

* **We need to create another role for the cloudformation to assume. So**

**Goto Roles -> create role -> click on cloudformation**

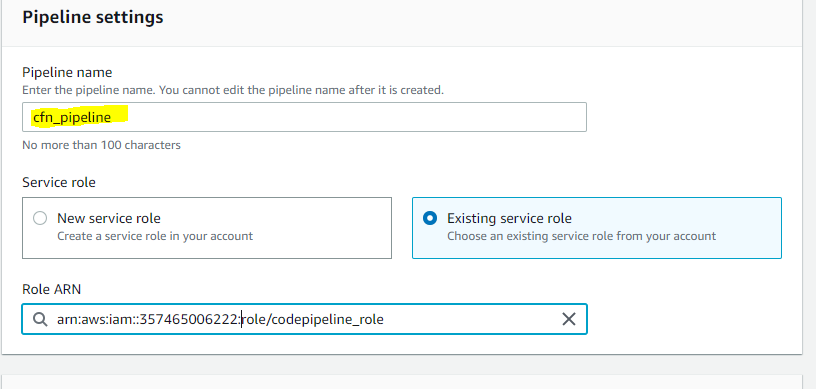


* **In Permission add the policies of the resources which we need to create.**
* **Click on Next and enter the role name. Click on create Role and role will be created.**

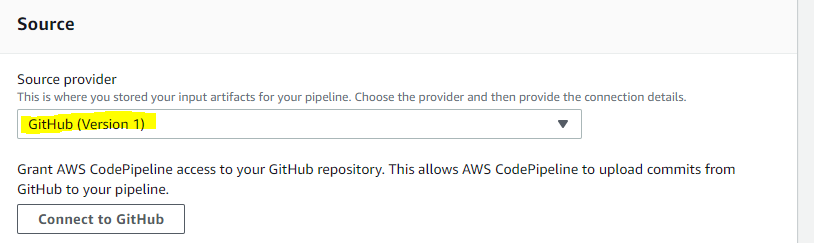


* **Go to Codepipeline.**

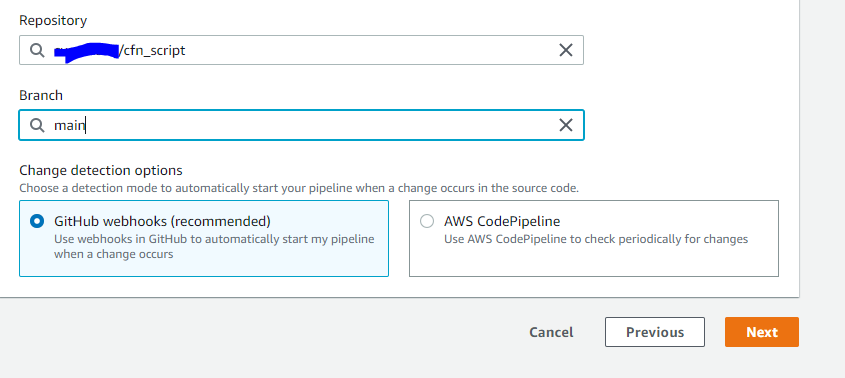
**Create pipeline –> Enter the pipeline Name and click on existing role -> select the arn of pipeline role**



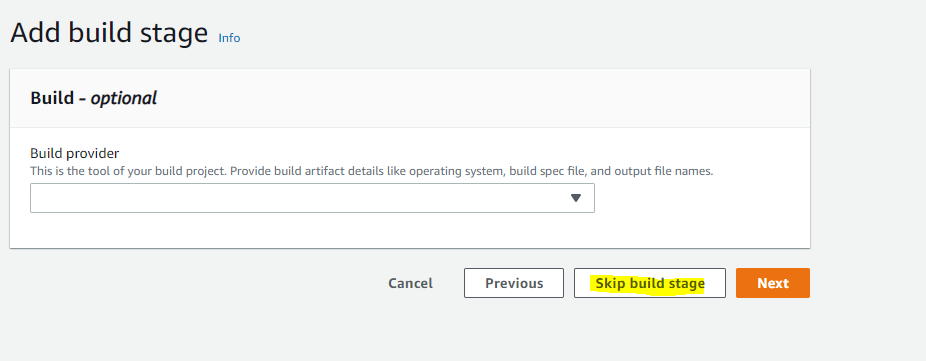
* **Click on Next and you will be redirected to Source.**
* **Under Source select the source where your code resides**



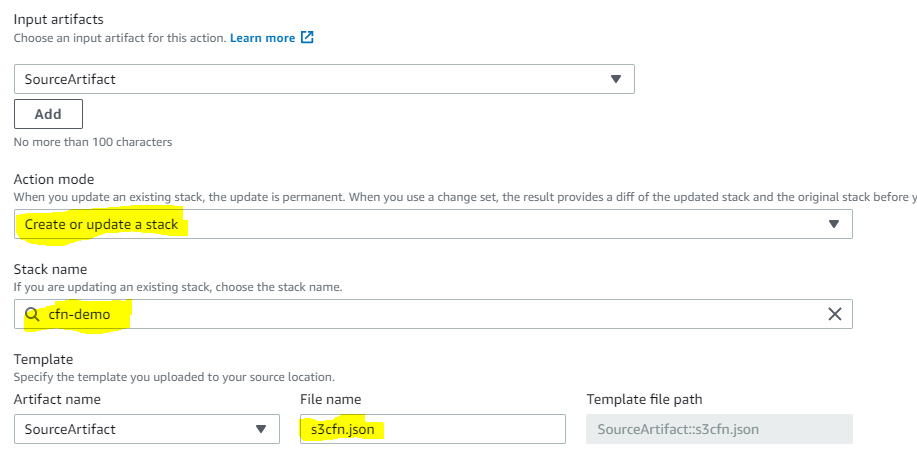
* **Enter the repo name and branch**



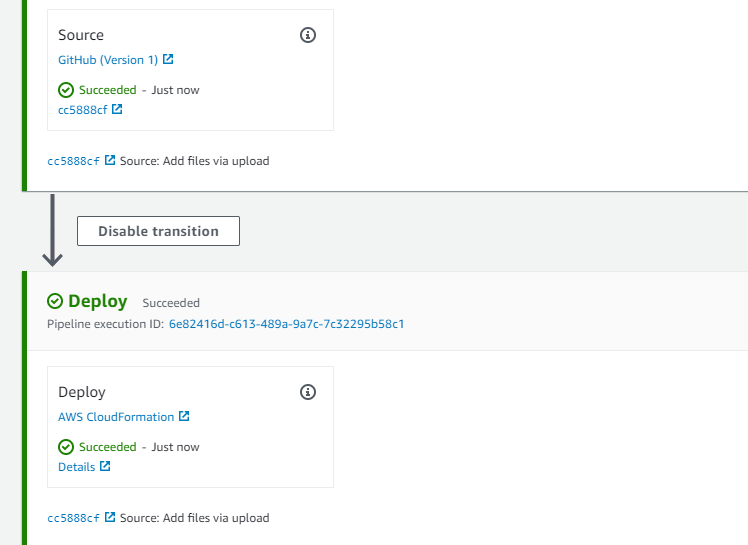
* **Click on next and as we don’t have any build stage, click on “Skip Build”.**



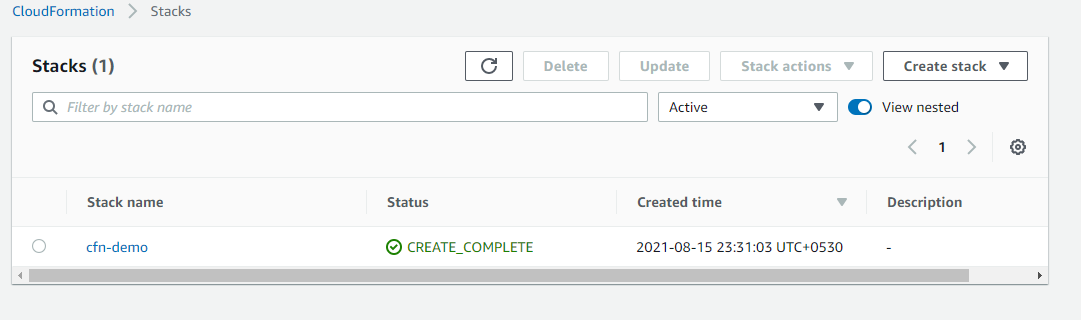
* **Click on Next and you will be moved to Deploy stage.**
* **Under Deploy, Enter the deploy provider as cloud formation, Action mode as “create or update stack”, stack name as required, artifact name should be “source artifact” and filename should be the cloudformation script file name(For Demo purpose we have create a script for creating s3 bucket). Under role enter the cloudformation role created.**



* **Click on Next and create pipeline.**



* **Go to CloudFormation console and check if the stack has been created/updated.**



**Thus Codepipeline for deploying CloudFormation can be done.**